GreenGuardian: Smart Irrigation & Smart Fertilization for Precision Agriculture

Project report submitted for the partial fulfillment of the requirements for the award of the degree

of

Bachelor of Technology

By

SUVADIP MONDAL (Roll No: 10305521038)

Under the supervision of

Dr. Soumya Roy

Associate Professor, Dept. of AEIE

Haldia Institute of Technology

Department of Applied Electronics & Instrumentation Engineering

Haldia, Purba Medinipur, West Bengal, India,

2025

CERTIFICATE

This is to certify that the project report entitled “GreenGuardian: Smart Irrigation & Smart Fertilization for Precision Agriculture” submitted by Suvadip Mondal (roll no: 10305521038) is a bonafide work carried out by them under the supervision of Dr. Soumya Roy and it is approved for the partial fulfillment of the requirement of the Haldia Institute of Technology, Maulana Abul Kalam Azad University Of Technology (MAKAUT) for the award of the degree of Bachelor of Technology (Applied Electronics & Instrumentation Engineering) during the academic year 2024-2025.

Dr. Soumya Roy
Associate Professor
Department of AEIE
[Project Supervisor]

Dr. Uday Maji
Head of the Deaprtmer
Department of AEIE

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to all those who have contributed to the successful completion of our college group project on the topic of GreenGuardian: Smart Irrigation & Smart Fertilization for Precision Agriculture. It has been an incredible journey, and we owe our achievements to the unwavering support and assistance we have received throughout this endeavor.

First and foremost, we extend our heartfelt appreciation to our project supervisor, Dr. Soumya Roy, for his invaluable guidance, constant encouragement, and expertise. His insights and suggestions have been instrumental in shaping our project and refining our ideas. We are immensely grateful for his patience, time, and dedication in overseeing our progress.

We would also like to extend our sincere thanks to the faculty members of Applied Electronics and Instrumentation Engineering Department, whose expertise and knowledge have enriched our understanding of the subject matter. Their insightful lectures and meaningful discussions have inspired us and provided a solid foundation for our project.

We are indebted to our fellow classmates and friends who have been an integral part of this project. Their unwavering support, collaborative efforts, and shared enthusiasm have created an environment conducive to learning and growth. We are grateful for their contributions and the memorable moments we have shared throughout this journey.

Additionally, we would like to express our appreciation to the staff of Haldia Institute of Technology, who have provided us with the necessary resources, facilities, and technical support to execute our project effectively. Their assistance has played a crucial role in the smooth execution of our ideas.

In conclusion, we express our deepest gratitude to everyone who has contributed to the completion of our college group project on GreenGuardian: Smart Power & Resource Management for Precision Agriculture. Your support, guidance, and collaboration have been instrumental in turning our vision into a reality. We are proud of what we have accomplished together and look forward to future endeavors fueled by the knowledge and experiences gained from this project.

ABSTRACT

This report presents a detailed analysis and implementation of a GreenGuardian: Smart Irrigation & Smart Fertilization for Precision Agriculture for Precision Agriculture as the final project for the Bachelor's degree program. Automating agricultural aspects is a mechanical process with or without human intervention in agriculture. This report is consisting of a theoretical and conceptual platform of Recommendation system through integrated models of collecting environmental factors using ESP32-S microcontrollers. Due to which farmers can easily predict, what crop type would be the most suitable for the selected area by collecting the environmental factors and processing them with the trained sub-models of the main of the system.

INDEX

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1.MOTIVATION AND OBJECTIVE

1.1.1. MOTIVATION

In the modern farms and agricultural operations as taken place more totally different than those many decades agone, primarily due to advancements in technology, as well as sensors, devices, machines, and knowledge technology. Today's agriculture habitually uses refined technologies like robots, temperature and wetness sensors, aerial pictures, and GPS technology, and lots of complex IOT devices. These advanced devices in agriculture enable businesses and farmers to be additional profitable, efficient, safer, and more environmentally friendly. The rise of digital agriculture and its connected technologies has opened a wealth of latest knowledge opportunities. Remote sensors and alternative connected devices will gather data twenty-four hours per day over a complete farm or land. These will monitor plant health, soil condition, temperature, humidity, etc. the quantity of information these sensors will generate is overwhelming. This enables farmers to achieve a far better an improved understanding of state of matters on the bottom through advanced technology which will inform them additional regarding their situation more accurately and quickly.

The environmental data that is gathered by remote sensors are processed by algorithms and statistical data which will be understood and helpful to farmers for decision makings and keep track of their farms. The more inputs and statistical data collected, and higher the algorithmic rule is at predicting the outcomes. And the aim is that farmers will use these technologies to attain their goal of improved harvest by creating better selections within the field. By implementing the system of temperature, soil hydrogen ion concentration and soil wetness detection, the information captured are processed with an explicit algorithmic rule and passed to a centralized database that is connected to different modules of the research, so the main system will the data and compared with the provided datasheet. If the measured variables are not at desired level, error signal will be generated which will actuate the required relay.

1.1.2. OBJECTIVE

1. **AI-POWERED SMART IRRIGATION:**

It includes:

- Optimize water usage through precise soil moisture monitoring and weather-based predictions.
- Increase crop yield with targeted irrigation tailored to specific crop requirement.

2. **INTELLIGENT FERTILIZER RECOMMENDATIONS:**

It includes:

- Reduce fertilizer waste with customized recommendations based on crop type and growth stage.
- Enhance crop health through optimate nutrient delivery

1.2. OUTLINE OF THE PROJECT

The outline of this project is to Design AI based tool. Smart management is done to reduce waste. Sensors detect the rain, moisture, humidity, temperature and nutrients present in the soil. The data is recorded and display on the app. If the measured variables are not at desired level, error signal will be generated which will actuate the required relay.

1.3. PURPOSE, SCOPE AND APPLICABILITY

1.3.1. PURPOSE AND SCOPE

The scope of this project includes the development and implementation of an intelligent system for automating irrigation and fertilization using real-time environmental data. The primary goal is to optimize the usage of water and fertilizers based on the current needs of the soil and crops, ultimately leading to better yield and sustainable farming practices.

The project leverages the ESP32-S microcontroller, a variety of environmental sensors, and cloud-based data logging to make informed decisions. This helps reduce human intervention and minimizes resource wastage. The limitations of this project that can be eliminated includes:

1. The enhancement of sensor calibration accuracy and the expansion of data analytics models to cover broader environmental conditions.
2. There might be issue with the components used in making this project

The dataset used for training and evaluation may not cover all possible variations and demographics, potentially limiting the generalization capability of the system.

1.3.2. APPLICABILITY

Few applicability in the project is:

1. This project will enhance the knowledge on embedded systems, sensor interfacing, Internet of Things (IoT), and real-time monitoring.
2. It is applicable in both educational and agricultural sectors.
3. Farmers, agricultural research institutes, and environmental monitoring organizations can benefit from implementing this project to optimize agricultural inputs and promote sustainable practices.

It can also be applied in urban farming, greenhouse automation, and precision gardening. The system can be used to develop mobile-based smart agriculture solutions and training kits for students and hobbyists. Furthermore, this project lays the groundwork for future integration with machine learning and weather forecasting APIs for advanced agricultural intelligence.

1.4. LITERATURE REVIEW

Here is an insight on the Literature Review of this project.

Sunilkumar and Arokiaraj (2024) describes this study uses LGBM to recommend suitable crops based on soil and weather data, enhancing yield and sustainability. It also applies a Random Forest algorithm to predict the ideal type and amount of fertilizer needed, enabling efficient resource use and supporting eco-friendly, data-driven farming decisions for improved agricultural productivity

Thaer Thaher; Isam Ishaq (2020) describe this study presents a smart irrigation system using WSN and IoT technologies. It combines Arduino, XBee, Raspberry Pi, and YL-69 soil moisture sensors to monitor and manage irrigation efficiently. Data is processed and visualized via ThingSpeak, enabling real-time crop monitoring and alerts, while highlighting challenges for future improvements.

Rasha Assaf; Isam Ishaq (2020) describe This project proposes a smart irrigation system for Palestine's agriculture, addressing declining rainfall. It uses an ESP8266 microcontroller to control water and fans based on soil moisture, temperature, and humidity. Data is monitored via the Blynk server, ensuring efficient, automated irrigation based on real-time environmental conditions.

Shekhar et. al. (2017) describe this project presents an intelligent IoT-based automated irrigation system using M2M communication. Soil moisture and temperature data are analyzed using the K-Nearest Neighbor (KNN) machine learning algorithm to predict irrigation needs. The system is fully automated and built using low-cost devices like Arduino Uno and Raspberry Pi 3.

Ogidan et. al. (2019) describe this paper presents a smart irrigation system designed for water-scarce regions. It uses soil moisture and ultrasonic sensors to control pumps via an Arduino microcontroller. The system optimizes water usage by prioritizing irrigation based on crop needs and reservoir levels, demonstrating efficient performance through a lab-scale farm model.

Olugbenga Kayode Ogidan, Abiodun Emmanuel Onile, Oluwabukola Grace Adegboro
Department of Electrical and Electronic Engineering, Elizade University, Ilara Mokin, Nigeria.

GreenGuardian, integrates smart irrigation and fertilization using ESP32-S, real-time sensor data, and mobile app visualization, offering a more holistic and scalable solution. Compared to Sunilkumar and Arokiaraj (2024), who focused on crop and fertilizer prediction using ML, your system uses direct sensor feedback for immediate action. While Thaer Thaher, Rasha Assaf, and Ogidan's systems focus mainly on irrigation, yours combines multi-sensor input (moisture, rain, humidity, temperature, and NPK) for dual resource management. Unlike Shekhar et al.'s ML-driven irrigation with basic sensors, GreenGuardian applies real-time logic with IoT and cloud integration, enhancing adaptability, control, and precision in agriculture.

CHAPTER 2

DESIGN AND IMPLEMENTATION

2.1. PHYSICAL DESIGN

2.1.1. DESIGN METHODLOGY

In this project, a modular design approach is followed. The sensors are interfaced with the ESP32-S microcontroller which processes the sensor data and triggers actuators through relay modules. Data is also transmitted to a cloud platform via Wi-Fi for real-time monitoring. Detail block diagram of the project is shown in figure no.1.

2.1.2. DESIGN OVERVIEW

Designing and implementing the system involves integrating sensors, relays, a buzzer, and the ESP32-S on a PCB and studying their performance. Here's an overview of the design:
Components Needed to design the project:

1. ESP32: The central microcontroller.
2. Soil Moisture Sensor: Measures the volumetric water content in the soil.
3. DHT11 Sensor: A Temperature & Humidity monitoring sensor which monitors ambient weather conditions.
4. Rain Drop Sensor: Detects the presence and intensity of rainfall.
5. NPK Sensor: Measures soil nutrient levels to guide fertilization.
6. Relay Module: Used to control pumps and other actuators.
7. Buzzer: Provides audible alerts based on system events.
8. Power Adapter: Supplies regulated power to the circuit.

2.2. BLOCK DIAGRAM

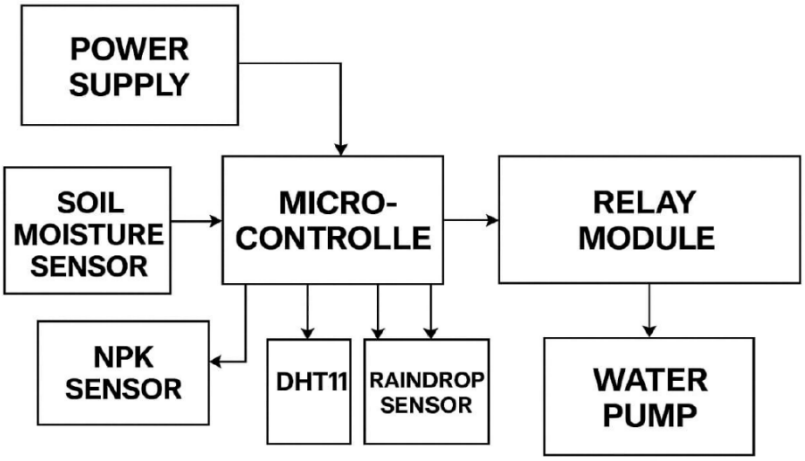

Fig 1: Block Diagram

The block diagram consists of the ESP32-S microcontroller at the centre. Various sensors such as DHT11, Rain Drop Sensor, Soil Moisture Sensor, and NPK Sensor are connected to it. Based on the sensor readings, signals are sent to the relay modules to control water and fertilizer pumps. Data is also uploaded to Firebase and Google Sheets.

2.3. CIRCUIT DIAGRAM

The overall circuit diagram of the project is shown below:

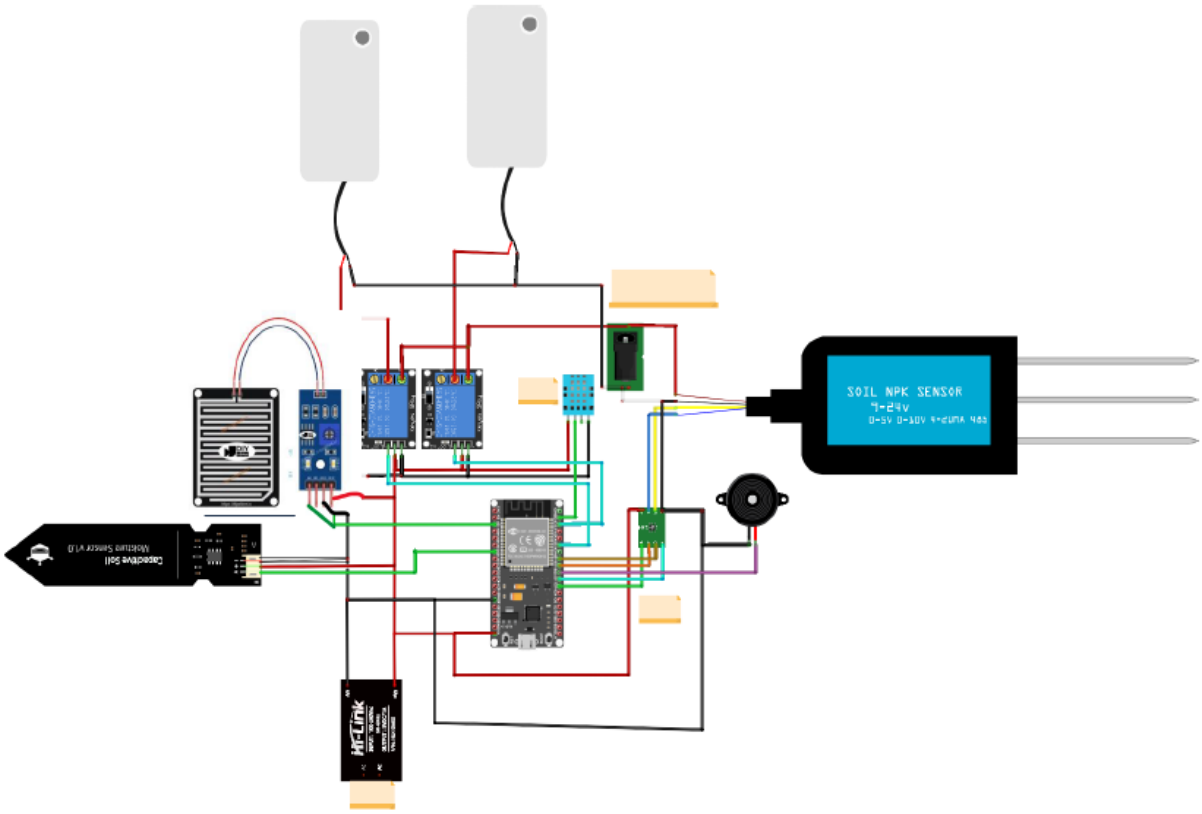

Fig 2: Circuit Diagram

Here, the main objective of this work is as follows:

- The ESP32-S microcontroller mounted on a custom PCB.
- Connections to the DHT11 sensor, Rain Drop sensor, and Soil Moisture sensor
- The NPK sensor connected via Modbus RS485.
- Relay modules wired to pump and buzzer.
- Power supply through a regulated adapter
- All sensors and actuators are properly interfaced with GPIO pins of ESP32-S

2.4. ALGORITHM

STEP 1: Initialize all sensor modules and Wi-Fi connection.

STEP 2: Continuously read data from DHT11, Soil Moisture, Rain Drop, and NPK sensors.

STEP 3: Display sensor values on Serial Monitor and the folder we created.

STEP 4: If soil moisture is below threshold and no rain is detected, turn on irrigation pump via relay.

STEP 5: If NPK values are below threshold, turn on fertilizer pump via relay.

STEP 6: Send sensor values to Serial Monitor and Google Sheets.

STEP 7: Sound buzzer for alerts on abnormal values or in error in connections

STEP 8: Log timestamp with sensor readings.

STEP 9: Repeat data acquisition and control loop.

2.5. DATA FLOW DIAGRAM

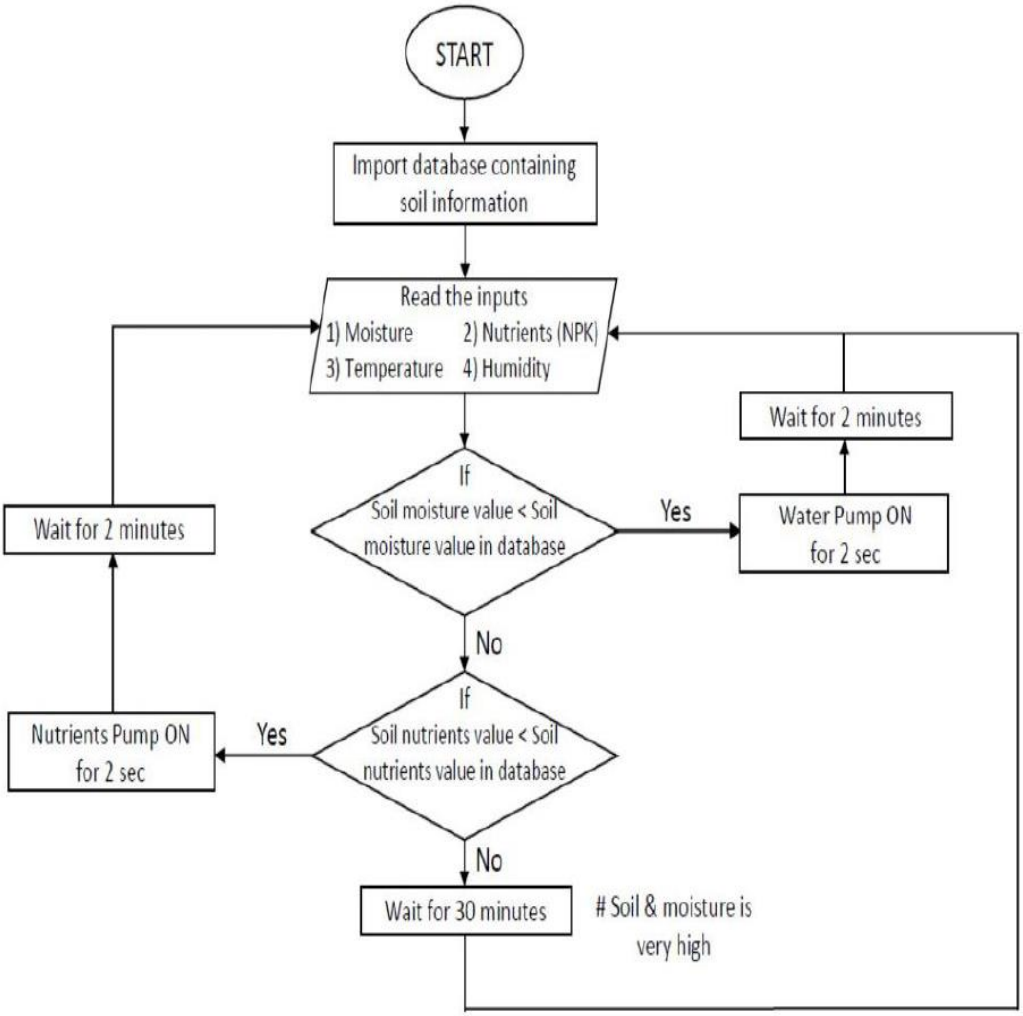

Fig 3: Flow Diagram

2.6. HARDWARE REQUIREMENTS

Hardware components needed for this project are:

1. **ESP32-S**

   The ESP32-S is a dual-core microcontroller, which has built-in Wi-Fi and Bluetooth capabilities. The ESP32S is a powerful and versatile chip that combines performance and connectivity. It can be used for low-power sensor networks which can handle data collection, processing, and communication between sensors and actuators.

Fig 4: ESP32-S

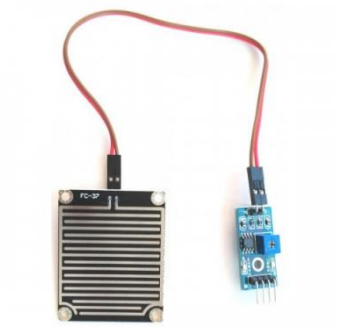

Fig 5: Rain Drop Sensor

2. **RAIN DROP SENSOR**

   The Raindrops Detection sensor module is used for rain detection. It consists of two modules, a rain board that detects the rain and a control module, which compares the analog value, and converts it to a digital value.

3. **NPK SENSOR**

   Soil nitrogen, phosphorus and potassium sensor is suitable for detecting the content of nitrogen, phosphorus and potassium in soil. It can judge the fertility degree of soil by detecting the content of nitrogen, phosphorus and potassium in soil, thus facilitating the systematic evaluation of soil conditions.

Fig 6: NPK Sensor

4. **DHT11 TEMPERATURE AND HUMIDITY SENSOR**
   DHT11 is a popular temperature and humidity based digital sensor. The sensor uses a capacitive humidity sensor and a thermistor-based temperature sensor to measure the ambient humidity and temperature. It monitors ambient temperature and humidity levels. It provides environmental data to fine-tune irrigation and fertilizer schedules based on weather conditions.

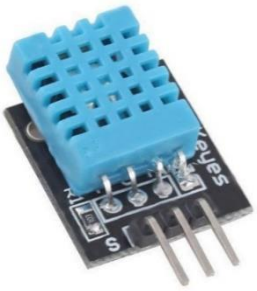

Fig 7: DHT11 Sensor

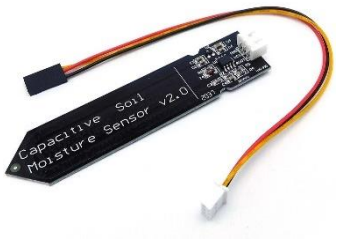

Fig 8: Soil Moisture Sensor

5. **SOIL MOISTURE SENSOR**
   Soil moisture sensors measure the water content in the soil and can be used to estimate the amount of stored water in the soil horizon. Soil moisture sensors do not measure water in the soil directly. Instead, they measure changes in some other soil property that is related to water content in a predictable way.

Details of the Equipment's above are in the Appendix.

2.7. SAFETY ISSUES

Safety issues are a critical aspect to consider when designing this project. While this offers numerous benefits, it also raises concerns regarding connections, reading errors, etc.

1. Proper Insulation: Ensure all electrical connections and components are properly insulated to prevent short circuits and electric shocks.
2. Secure Sensors: Ensure sensors are not exposed to excess water
3. Handling Power: Handle the power supply with care to prevent short circuits.
4. Robust Code: Write robust, well-tested code to avoid unexpected behaviors that could cause unsafe conditions.

CHAPTER 3

RESULT ANALYSIS AND DISCUSSION

3.1. RESULTS ANALYSIS

The performance of all the sensors functioned correctly and gave accurate readings when tested.

Writing and Uploading of Code:

Fig 9: Code in Arduino IDE

The code got uploaded successfully in the ESP-WROOM-32 Wi-Fi Microcontroller.

The errors present in the code was elimination for the proper running of the code.

The ESP32-S microcontroller successfully processed data and controlled irrigation and fertilization systems via relay modules. The collected data was transmitted and displayed on the file and also in Google Sheets.

3.2. PERFORMANCE EVALUATION

There was a proper functioning of all the sensors and receiving of the data in the file which we have created using MIT app investor. Where all result is clearly visible on the screen so that one can easily check the proper data for further evaluation.

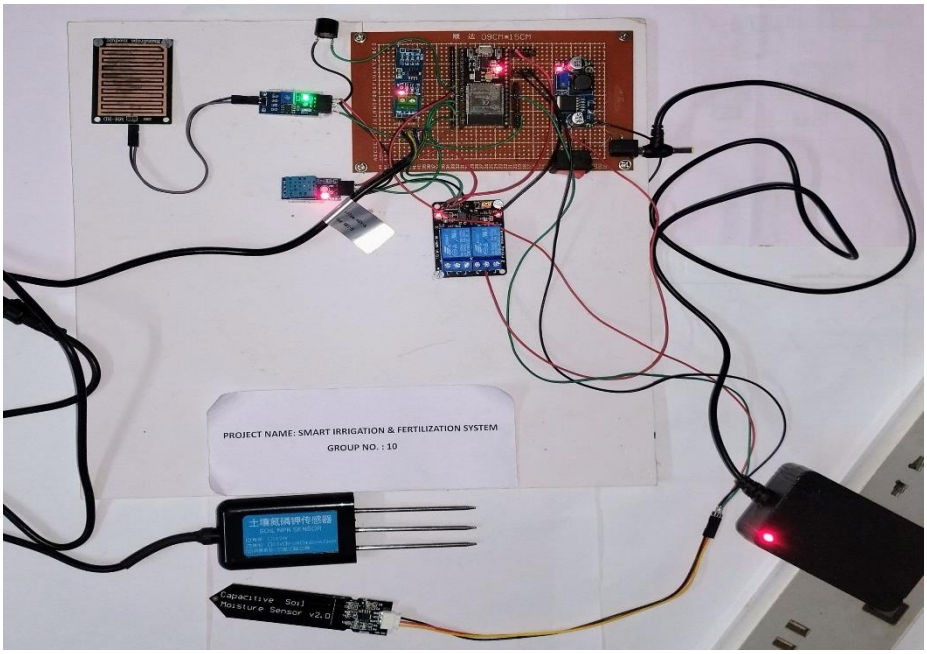

Fig 10: The Working Model of our project

The code is written in Arduino IDE, and is inserted in Microprocessor which control all the sensors and help to receive data. Arduino IDE is also recording data at the Serial Monitor. So, the note of the data which is recorded in the file can be taken.

The system was tested under simulated field conditions. It demonstrated effective irrigation when the soil moisture dropped below a set threshold, and proper nutrient distribution based on NPK values. Data was successfully logged and displayed, proving system reliability and potential scalability.

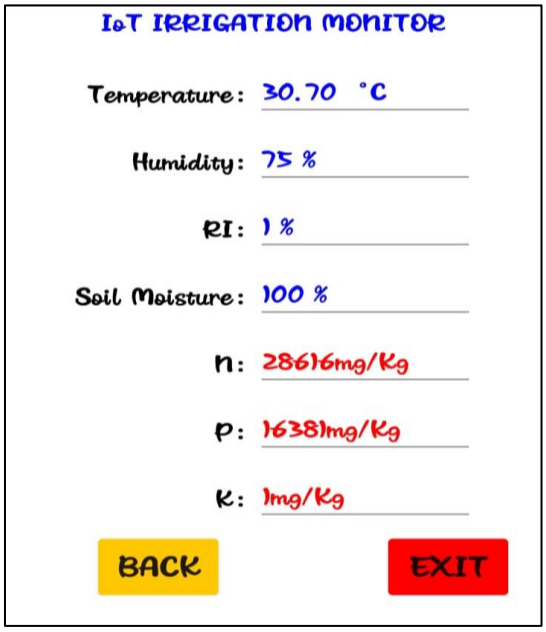

Fig 11: Output showing in the file we created

The old data which we have recorded is also saved in Recorded Data file which can be visible at any time.

Date_Time	Temperature	Humidity	RI	SMI	N	P	K
5/2/2025 12:48:10	30.2	59	1	31	29540	16381	65535
5/2/2025 12:49:28	30.2	58	1	31	29540	16381	65535
5/23/2025 10:47:44	31	76	53	30	29540	16381	65535
5/24/2025 9:21:58	30	76	59	31	28380	16381	65535
5/24/2025 9:23:20	30	75	54	31	28380	16381	65535
5/24/2025 9:40:38	30.1	75	61	32	29060	16381	65535
5/24/2025 9:41:57	30.2	76	71	31	29060	16381	65535
5/24/2025 9:43:38	30.4	75	27	31	28616	16381	65535
5/24/2025 9:44:55	30.5	75	1	32	28616	16381	65535
5/24/2025 9:46:13	30.5	75	1	32	28616	16381	65535
5/24/2025 9:47:35	30.5	75	1	31	28616	16381	65535
5/24/2025 9:48:55	30.5	75	1	75	28616	16381	65535
5/24/2025 9:50:14	30.6	75	1	70	28616	16381	65535
5/24/2025 9:51:32	30.6	75	1	69	28616	16381	65535
5/24/2025 9:52:50	30.6	75	1	68	28616	16381	65535
5/24/2025 9:54:05	30.6	75	1	68	28616	16381	65535
5/24/2025 9:55:21	30.5	75	1	67	28616	16381	65535
5/24/2025 9:56:40	30.5	75	1	67	28616	16381	65535
5/24/2025 9:57:58	30.5	75	1	67	28616	16381	65535
5/24/2025 9:59:14	30.5	75	1	67	28616	16381	65535
5/24/2025 10:00:34	30.6	75	1	67	28616	16381	65535
5/24/2025 10:01:51	30.6	75	1	66	28616	16381	65535
5/24/2025 10:03:13	30.6	75	1	64	28616	16381	65535
5/24/2025 10:04:41	30.7	75	1	66	28616	16381	65535

Fig 12: Output showing in the Recorded Data

CHAPTER 4

CONCLUSIONS

4.1. APPLICATIONS

Applications of different control actions include:

- Smart irrigation based on soil moisture levels.
- Fertilization control using NPK soil analysis.
- Remote monitoring and control of farm parameters.
- Data logging and analysis for predictive maintenance and resource planning.
- Integration with weather forecasting to schedule watering/fertilizing tasks.
- Real-time alerts and notifications for abnormal field conditions.

4.2. LIMITATIONS OF THE SYSTEM

The limitations which can be occur are as follows:

- Relies on stable internet connectivity for real-time data upload.
- Requires precise sensor calibration.
- Limited by the number of sensors supported by the ESP32-S.
- No machine learning implementation for decision-making.

4.3. FUTURE ASPECTS OF THE PROJECT

The future aspects of the project are:

1. **Optimize Power Usage:**
   - Energy consumption will reduce by approximately 30% through the system's intelligent power management protocols.
   - Battery lifespan of the system's operational units will be extended by an average of 40%, ensuring sustainability and cost-effectiveness.

2. **Increase Coverage:**
   - Add more sensors for broader analysis like pH and EC.
   - Integrating these will offer a holistic view of soil conditions and help in precision farming.

3. **AI Integration:**
   - Use AI for smart predictions and anomaly detection.
   - Algorithms can predict optimal irrigation/fertilization times and detect unusual patterns indicating pest or disease outbreaks.

4. **Cloud Dashboard:**
   - Develop a mobile/web dashboard for easier monitoring.
   - Features can include real-time alerts, performance charts, and remote manual override for pumps.

5. **Solar-Powered Module:**
   - Add solar panel for energy independence.
   - This would make the system viable in remote areas and reduce operational costs.

SAMPLE CODE

```
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
#include <HTTPClient.h>
#include "time.h"
#include <WiFi.h>
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 19800;
const int   daylightOffset_sec = 0;

String GOOGLE_SCRIPT_ID =
"AKfycbzsNxslwCZo7spDIjLEYL_W6uEm6aFDh9dbSLTKeoByX00FsxUGgOUvioOMnG
09Qt-3FQ";    // Actually deployment ID

// WiFi Credentials
#define WIFI_SSID "ROBOTICS SOLUTIONS"
#define WIFI_PASSWORD "bana@SJ17"

// Firebase Config
#define API_KEY "AIzaSyD_3Z_VrGSTvrQWmieWBKHori-HsKh8Sdw"
#define DATABASE_URL "https://iot-air-quality-monitor-494d4-default-
rtdb.firebaseio.com/"

// Firebase setup
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

char timeStringBuff[50];
String asString;

#include <ModbusMaster.h>

// Pin definitions
#define MAX485_DE 19
#define MAX485_RE 18
#define RXD2 16
#define TXD2 17

ModbusMaster node;

#include "DHT.h"
#define DHTPIN 23
#define DHTTYPE DHT11
```

```
DHT dht(DHTPIN, DHTTYPE);
int SMI=0;
int AQI = 0;

#define buzzer 5
#define PUMP_W 22
#define PUMP_L 21

void preTransmission() {
  digitalWrite(MAX485_RE, 1);
  digitalWrite(MAX485_DE, 1);
}

void postTransmission() {
  digitalWrite(MAX485_RE, 0);
  digitalWrite(MAX485_DE, 0);
}

void setup() {
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);

  pinMode(MAX485_DE, OUTPUT);
  pinMode(MAX485_RE, OUTPUT);

  pinMode(PUMP_W, OUTPUT);
  pinMode(PUMP_L, OUTPUT);

  digitalWrite(MAX485_DE, 0);
  digitalWrite(MAX485_RE, 0);

  node.begin(1, Serial2); // 1 = Modbus slave address
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);
  dht.begin();
  pinMode(buzzer,OUTPUT);
   beep(3,500);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
     beep(1,100);
  }

  Serial.println("\n☑ WiFi Connected!");
     beep(1,1000);
```

```
  // Firebase Configuration
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  auth.user.email = "swechasinha20@gmail.com";
  auth.user.password = "123456789";
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}

void loop() {
    static bool flag = false;
    struct tm timeinfo;
    if (!getLocalTime(&timeinfo)) {
      Serial.println("Failed to obtain time");
      return;
    }

    //50 chars should be enough
    strftime(timeStringBuff, sizeof(timeStringBuff), " %d %B %Y %H:%M:%S", &timeinfo);
String asString(timeStringBuff);
    asString.replace(" ", "-");
    Serial.print("Time:");
    Serial.println(asString);

  uint8_t result;
  uint16_t data[6];
  result = node.readHoldingRegisters(0x0000, 6); // Read 6 registers starting from 0x0000

  if (result == node.ku8MBSuccess) {
    for (int i = 0; i < 3; i++) {
      data[i] = node.getResponseBuffer(i);  // N, P, K values
    }

    Serial.print("N: "); Serial.print(data[0]); Serial.print(" mg/kg, ");
    Serial.print("P: "); Serial.print(data[1]); Serial.print(" mg/kg, ");
    Serial.print("K: "); Serial.print(data[2]); Serial.println(" mg/kg");
      if(data[0]<50 || data[1] <50 || data[2]<50){
    digitalWrite(PUMP_L,0);
  delay(1000);
  }
  else{
    digitalWrite(PUMP_L,1);
  }
  } else {
    Serial.print("Error: ");
    Serial.println(result);
  }
```

```
delay(2000);
float dhtTemp = dht.readTemperature();
float humidity = dht.readHumidity();
AQI =100- map(analogRead(36),0,4096,0,100);
SMI =100- map(analogRead(32),0,4096,0,100);
if(SMI<50) {
  digitalWrite(PUMP_W,0);
delay(1000);
}
else{
  digitalWrite(PUMP_W,1);
}

Serial.print("DHT11 Temp: ");
Serial.print(dhtTemp);
Serial.println(" °C");
Serial.print("Humidity: ");
Serial.print(humidity);
Serial.println(" %");
Serial.print("RI: ");
Serial.print(AQI);
Serial.println(" %");
Serial.print("SMI: ");
Serial.print(SMI);
Serial.println(" %");
delay(1000);

if (WiFi.status() == WL_CONNECTED) {
   // Send fault data
   if (Firebase.RTDB.setInt(&fbdo, "/IrrigationData/Temperature", dhtTemp))
     Serial.println("☑ Temperature updated");
   else
     Serial.println("❌ Temperature update error: " + fbdo.errorReason());

   if (Firebase.RTDB.setInt(&fbdo, "/IrrigationData/Humidity", humidity))
     Serial.println("☑ humidity updated");
   else
     Serial.println("❌ humidity update error: " + fbdo.errorReason());

   if (Firebase.RTDB.setInt(&fbdo, "/IrrigationData/LPG", SMI))
     Serial.println("☑ SMI updated");
   else
     Serial.println("❌ SMI update error: " + fbdo.errorReason());

     if (Firebase.RTDB.setInt(&fbdo, "/IrrigationData/AQI", AQI))
     Serial.println("☑ RI updated");
   else
```

```
      Serial.println("❌ RI update error: " + fbdo.errorReason());
      if (Firebase.RTDB.setInt(&fbdo, "/IrrigationData/N", data[0]))
      Serial.println("☑ N updated");
    else
      Serial.println("❌ N update error: " + fbdo.errorReason());

      if (Firebase.RTDB.setInt(&fbdo, "/IrrigationData/P", data[1]))
       Serial.println("☑ P updated");
    else
      Serial.println("❌ P update error: " + fbdo.errorReason());

      if (Firebase.RTDB.setInt(&fbdo, "/IrrigationData/K", data[2]))
      Serial.println("☑ K updated");
    else
      Serial.println("❌ K update error: " + fbdo.errorReason());
}

String urlFinal =
"https://script.google.com/macros/s/"+GOOGLE_SCRIPT_ID+"/exec?"+"date=" + asString
+ "&temperature=" + String(dhtTemp) + "&humidity=" + String(humidity) +"&aqi=" +
String(AQI) +"&smi=" + String(SMI) +"&N=" + String(data[0]) +"&P=" + String(data[1])
+"&K=" + String(data[2]);
    Serial.print("POST data to spreadsheet:");
   Serial.println(urlFinal);
   HTTPClient http;
   http.begin(urlFinal.c_str());
   http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
   int httpCode = http.GET();
   Serial.print("HTTP Status Code: ");
   Serial.println(httpCode);
   //---------------------------------------------------------------------
   //getting response from google sheet
   String payload;
   if (httpCode > 0) {
      payload = http.getString();
      Serial.println("Payload: "+payload);
   }
   //---------------------------------------------------------------------
   http.end();
  delay(60000);
}
void beep(int k, int t) {
  for(int i=0 ; i<k ; i++) {
    digitalWrite(buzzer,1); delay(t);
     digitalWrite(buzzer,0); delay(t);
  }
}
```

BIBLIOGRAPHY

[1] S. A and A. A. Jovith, "Smart Irrigation System By Recommending Crop And Fertilizers," 2024 2nd International Conference on Networking and Communications (ICNWC), Chennai, India, 2024, pp. 1-8, doi: 10.1109/ICNWC60771.2024.10537564.

[2] T. Thaher F., I. Ishaq S.: "Cloud-based Internet of Things Approach for Smart Irrigation System: Design and Implementation,". International Conference on Promising Electronic Technologies (ICPET), 2020, pp. 32-37, doi: 10.1109/ICPET51420.2020.00015.

[3] R. Assaf F., I. Ishaq S.: Improving Irrigation by Using a Cloud Based IoT System. International Conference on Promising Electronic Technologies (ICPET), 2020, pp. 28-31, doi: 10.1109/ICPET51420.2020.00014.

[4] Yuthika Shekhar F.: Intelligent IoT Based Automated Irrigation System. International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp. 7306-7320.

[5] Olugbenga Kayode Ogidan F., Abiodun Emmanuel Onile S., Oluwabukola Grace Adegboro T.: "Smart Irrigation System: A Water Management Procedure". Agricultural Sciences, doi: 10.4236/as.2019.101003

[6] V Gawande, D R K Saikanth, B S Sumithra, A Aravind, "Potential of Precision Farming Technologies or eco-friendly agriculture" *International Journal of Plant & Soil Science,* Vol 35, No. 19 August 2023, DOI: 10.9734/IJPSS/2023/v35i/193528.

[7] R K Raman, A Kumar, S Sarkar, A K Yadav, "Reconnoitering Precision Agriculture and Resource Management" *Indian Research Journal of Extension Education,* Vol 24, No. 1 February 2024.

[8] https://en.wikipedia.org/wiki/Power_managemen.

[9] https://en.wikipedia.org/wiki/ESP32

[10] https://www.researchgate.net/publication/346627389_Crop_Recommendation_System

APPENDIX

1. ESP32-S:

- Uses -
  - Act as the central microcontroller.
  - Handles data collection, processing, and communication between sensors and actuators.
  - Facilitates Wi-Fi connectivity for real-time monitoring and remote control.
- Connections –
  - The ESP32-S is the core microcontroller and should have GPIO pins assigned for all peripherals.
  - Sensors are connected with it.
- Software -
  - *Arduino IDE*: The Arduino Uno is programmed using the Arduino Integrated Development Environment (IDE), which supports a simplified version of C/C++.
  - *MIT App Investor*: Used to develop a custom Android application to view sensor data in real time.
- Programming:
  - *Sketches*: Arduino programs are called sketches. A sketch typically consists of two main functions: setup() and loop().

2. Rain Drop Sensors:

- Uses -
  - Detects rainfall and provides input to the irrigation system.
  - Helps adjust watering schedules to prevent over-irrigation during or after rainfall.
- Connections –
  - *VCC* → Connect to the 3.3V pin of ESP32.
  - *GND* → Connect to the GND of ESP32.
  - *OUT* → Connect to any GPIO pin (e.g., GPIO34) for digital signal reading.
- Working Principle -
  - *Conductive Plate Design*: The sensor module consists of a PCB with conductive tracks (typically a grid-like pattern) that act as a rain detection surface.
  - *Water Conductivity*: When raindrops fall on the sensor plate, water bridges the gaps between the conductive tracks, allowing current to flow through the circuit.
  - *Resistance Change*:
    - Dry surface → High resistance → Low current flow
    - Wet surface → Low resistance → High current flow
  - *Comparator Threshold*: The onboard **potentiometer** allows adjusting the sensitivity by setting a threshold for digital output switching.

3. NPK Sensor:

- Uses -
  - Measures soil nutrient levels (Nitrogen, Phosphorus, Potassium).
  - Provides data for intelligent fertilizer recommendations, optimizing nutrient delivery for crops.
- Connections –
  - *VCC* → Connect to the 5V output of XL6009 (step-up to 5V).
  - *GND* → Connect to GND.
  - *OUT* → Connect to an analog GPIO pin (e.g., GPIO36) of ESP32 for reading sensor data.
- Working Principle -
  - *Ion-Selective Electrodes (ISEs)*: The sensor uses ion-selective electrodes to detect the presence of Nitrogen (N), Phosphorus (P), and Potassium (K) ions in the soil.
  - *Electrochemical Measurement*: Each electrode produces a voltage signal in response to the concentration of a specific nutrient ion in the soil.
  - *Analog to Digital Conversion*: The sensor internally converts the analog voltage values into digital data, representing the nutrient concentration in mg/kg (ppm).
  - *Depth-Based Probing*: The sensor is inserted into the soil at the root zone level, ensuring accurate readings of available nutrients for plant uptake.

4. DHT11 Sensor:

- Uses -
  - Monitors ambient temperature and humidity levels.
  - Provides environmental data to fine-tune irrigation and fertilizer schedules based on weather conditions.
- Connections –
  - *VCC* → Connect to the 3.3V pin of ESP32.
  - *GND* → Connect to GND.
  - *DATA* → Connect to a GPIO pin (e.g., GPIO27). Use a pull-up resistor (4.7kΩ or 10kΩ) between DATA and VCC.
- Working Principle -
  - *Sensing Elements*:
    - *Humidity*: Uses a capacitive humidity sensor where changes in humidity alter the dielectric constant of a polymer layer, affecting capacitance.
    - *Temperature*: Uses a thermistor (a temperature-sensitive resistor) that changes resistance with temperature.
  - *Signal Conversion*: The analog signals from the sensing elements are converted into digital signals using an onboard ADC (Analog-to-Digital Converter).
  - *Digital Communication*: The sensor sends pre-calibrated digital signals to the microcontroller using a single-wire serial interface.

- *Measurement Cycle*: It can measure data every 1 second (once per second). Frequent reads may return the same value.
- *Data Format*: The output includes 8-bit integer values for both humidity and temperature, along with a checksum byte for data integrity.

5. Soil Moisture Sensor:

➢ Uses -
- *Measure Soil Water Content*: Detects moisture levels in the soil to determine irrigation needs.
- *Crop Health Monitoring*: Ensures crops receive optimal water, improving yield and reducing water stress.
- *Water Conservation*: Prevents over-irrigation and conserves water resources.

➢ Connections –
- *VCC* → Connect to the 3.3V pin of ESP32.
- *GND* → Connect to GND.
- *DATA* → Connect to a GPIO pin (e.g., GPIO34) to read moisture levels as a variable voltage.

➢ Working Principle -
- *Electrical Conductivity-Based Detection*: The sensor uses two conducting probes that are inserted into the soil. Moisture in the soil conducts electricity between the probes.
- *Resistance Variation*:
  - Wet soil has low resistance (more conductivity), allowing more current to pass.
  - Dry soil has high resistance (less conductivity), allowing less current to pass.
- *Analog Voltage Output*: Based on the resistance, the sensor outputs an analog voltage.
  - Lower voltage = More moisture
  - Higher voltage = Less moisture