Mini Project Report on

### Smart Plant Monitoring System Using Blynk IoT

submitted in partial fulfillment for award of the degree of Bachelor of Technology in Computer Science and Engineering(AI&ML)

**CH. Kaartheeka 22321A6618**

**K Lalitha 22321A6623**

**K Sindhu 22321A6641**

**N Sneha 22321A6642**

Under the esteemed Guidance of Internal Guide

Dr M Seshu Bhavani Associate Professor

CSE (AI& ML)



Bhoj Reddy Engineering College for Women

Department of Computer Science and Engineering(AI&ML)

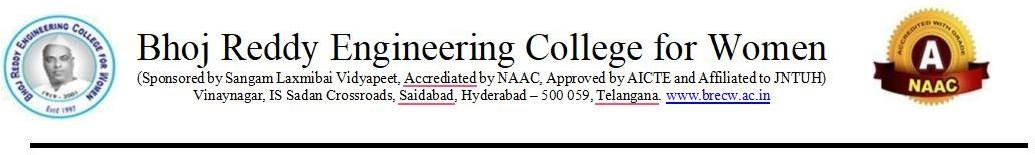
(Sponsored by Sangam Laxmibai Vidyapeet, Accredited by NAAC with A Grade, Approved by AICTE and Affiliated to JNTUH)

Recognised by UGC under section 2(f) of the UGC Act, 1956.

Vinaynagar, IS Sadan Crossroads, Hyderabad – 500 059

Ph: +91-40-2453 7282; Website: [www.brecw.ac.in](http://www.brecw.ac.in/); Email: [principal@brecw.ac.in](mailto:principal@brecw.ac.in)

2024- 2025

**CERTIFICATE**

This is to certify that the Mini Project Report entitled **“Smart Plant Monitoring System Using Blynk IoT”**is a bonafide work carried out

by

**CH Kaartheeka 22321A6618**

**K Lalitha 22321A6623**

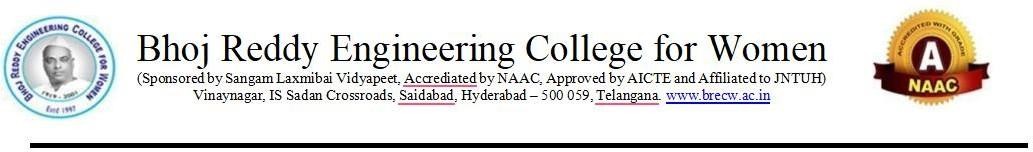
**K Sindhu 22321A6641**

**N Sneha 22321A6642**

in partial fulfillment for award of the degree of Bachelor of Technology in Computer Science and Engineering department from Bhoj Reddy Engineering College for Women,

|  |  |  |  |
| --- | --- | --- | --- |
| Hyderabad, affiliated to (JNTUH). | Jawaharlal | Nehru | Technological University Hyderabad |
| **Internal Guide**  **Dr M Seshu Bhavani** |  |  | **Head of the Department Dr M Seshu Bhavani** |

###### External Examiner

****

**CERTIFICATE**

This is to certify that the Mini Project Report entitled entitled **“Smart Plant Monitoring System Using Blynk IoT”** is a bonafide work carried out by

**CH Kaartheeka 22321A6618**

**K Lalitha 22321A6623**

**K Sindhu 22321A6641**

**N Sneha 22321A6642**

in partial fulfillment for award of the degree of Bachelor of Technology in Computer Science and Engineering department from Bhoj Reddy Engineering College for Women, Hyderabad, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH).

**Internal Guide Head of the Department**

**Dr M Seshu Bhavani Dr M Seshu Bhavani**

**ACKNOWLEDGMENT**

It is our pleasure to express our whole-hearted thanks to Dr. M Seshu Bhavani, Associate Professor & Head of Computer Science and Engineering(AI&ML) Department for providing the necessary facilities to under go this project.

We are thankful to our project coordinator Mr Mohd Basit Mohiuddin, Assistant Professor, Computer Science and Engineering(AI & ML) Department, for giving us this opportunity to do this project.

We are also thankful to our internal guide Ms Dr M Seshu Bhavani, Associate Professor, Computer Science and Engineering Department for her extreme guidance and support in completing this project successfully.

We are also thankful to the staff members of Computer Science and Engineering(AI & ML) ,

my friends and to our parents who helped us in completing this project successfully.

By

Ch Kaartheeka (22321A6618)

KLalitha (22321A6623)

K Sindhu (22321A6641)

N Sneha (22321A6642)

### Contents

|  |  |  |
| --- | --- | --- |
|  |  | Page No |
| List of Figures | | i |
| List of Tables | | ii |
| Abbreviations Abstract | | iii iv |
| **1.** | **Introduction** | **1-3** |
|  | 1.1 Introduction of the project | 2 |
|  | 1.2 Existing system | 2 |
|  | 1.3 Problems in existing system | 2 |
|  | 1.4 Proposed system | 3 |
|  | 1.5 Advantages of proposed system | 3 |
| **2.** | **Requirements Analysis** | **4-8** |
|  | 2.1 Functional requirements | 5 |
|  | 2.2 Non-functional requirements | 6 |
|  | 2.3 Computational resource requirements | 6 |
|  | 2.3.1 Hardware resources | 6 |
|  | 2.3.2 Software resources | 6 |
|  | 2.4 Life cycle model | 7-8 |
| **3.** | **Architecture** | **9-15** |
|  | 3.1 Hardware Architecture  3.1.1 Block Diagram  3.1.2 Description of Hardware components  3.1.3 Arduino ESP8266  3.1.4 Soil moisture  3.1.5 DHT11 Sensor  3.1.6 18650 Battery  3.1.7 Breadboard & Jumper Wires  3.1.8 16x2 I2C LCD  3.1.9 Tactile Button  3.1.10 PIR motion detector  3.1.11 Relay module  3.1.12 Water pump 13 | 8  8  9  10  10  10  11  11  11  12  12  12 |

3.13 circuit diagram 14

|  |  |  |
| --- | --- | --- |
|  | 3.2 Technical Architecture  3.2.1 Data Flow Digram | 15 |
| **4** | **Implementation** | **16 -19** |
|  | 4.1 Technologies  4.1.1 C Programming  4.1.2 Features of C in Embedded Systems | 16 |
|  | 4.2 Technologies and Components Used | 17 |
|  | 4.3 Working of the System | 18-19 |
|  | 4.4 Pseudocode | 20-22 |
| **5.** | **Testing** | **24-25** |
|  | 5.1 Overview  5.1.1 Dimensions of Testing  5.1.2 Stages of Testing  5.1.3 Unit Testing  5.1.4 Integration Testing  5.1.5 System Testing  5.1.6 Acceptance Testing | 24 |
|  | 5.2 Types of Testing  5.2.1 Black box testing  5.2.2 White box Testing | 25 |
|  | 5.3 Test Cases  5.3.1 System Startup  5.3.2 Moister Checking  5.3.3 Water Dispensing  5.3.4 Motion detection  5.3.5 Temperature and humidity  5.3.6 System Shutdown | 26-27 |
| **6.** | **Screenshots** | **28-30** |
|  | 6.1 Blynk IOT Dashboard |  |
|  | 6.2 Practical setup |  |
|  | 6.3 Wet and Dry Soil |  |
|  | 6.4 Soil moisture readings |  |
|  | 6.5 Water flow using water pump |  |
|  | 6.6 LCD Display |  |
|  | 6.7 Digital moisture readings and motion detection and automatic water pump on |  |
| **7** | **Conclusion and Future Scope** | **32** |
|  | 7.1 Conclusion |  |
|  | 7.2 Future Scope |  |
| **8.** | **References** | **34** |

**List of Figure**

| **Fig. No** | | **Figure Name** | **Page No** |
| --- | --- | --- | --- |
| 2.4 | Water Fall Model | | 8 |
| 3.1.1 | Block Diagram | | 9 |
| 3.1.3 | Arduino ESP8266 | | 9 |
| 3.1.4 | Soil Moisture Sensor | | 10 |
| 3.1.5 | DHT11 Sensor | | 10 |
| 3.1.6 | 18650 Battery | | 11 |
| 3.1.7 | Breadboard and Jumper Wires | | 11 |
| 3.1.8 | 16x2 I2C LCD | | 12 |
| 3.1.9 | Tactile Button | | 12 |
| 3.1.10 | PIR Motion Detector | | 13 |
| 3.1.11 | Relay Module | | 13 |
| 3.1.12 | Water Pump | | 14 |
| 3.1.13 | Circuit Diagram | | 14 |
| 3.2.1 | Data Flow Diagram for Smart Plant Monitoring System Using IoT | | 15 |
| 6.1 | Blynk IoT Dashboard | | 28 |
| 6.2 | Practical Setup | | 29 |
| 6.3 | Wet and Dry Soil | | 29 |
| 6.4 | Soil Moisture Readings when Sensor is in Wet Soil | | 29 |
| 6.5 | Water Flow Using Water Pump | | 30 |
| 6.6 | LCD Display | | 30 |
| 6.7 | Digital Moisture Readings, Motion Detection, and Automatic Water Pump Activation | | 40 |

### 

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table.No** | **Table Name** | **Page.No** |
| 5.1 | System Startup | 27 |
| 5.2 | Moisture Checking | 27 |
| 5.3 | Water Dispensing | 27 |
| 5.4 | Motion detection | 27 |
| 5.5 | Temperature and humidity | 27 |
| 5.6 | System Shutdown | 27 |

### Abstract

The **Smart Plant Monitoring System** is an IoT-based solution designed to enhance plant care through the integration of environmental sensors, automation, and wireless connectivity. Utilizing the ESP8266 NodeMCU microcontroller, the system collects real-time data from soil moisture, temperature, humidity, and motion sensors. This data is transmitted to the Blynk IoT cloud platform, allowing users to monitor and control the system remotely through a mobile or web interface. By automating the watering process based on live soil moisture readings, the system ensures efficient water usage, prevents overwatering or underwatering, and promotes healthier plant growth. Additionally, the inclusion of motion detection adds a security layer, alerting users to unauthorized movements around the monitored area. Designed to reduce manual effort, support sustainable water management, and improve plant health, this system is ideal for home gardens, smart agriculture, and greenhouse environments. It exemplifies how the convergence of IoT and automation can lead to intelligent, eco-friendly plant care solutions..

# INTRODUCTION

### Introduction

#### Introduction

The **Smart Plant Monitoring System** is an innovative IoT-based solution developed to intelligently monitor and manage plant health and irrigation. By integrating embedded systems, sensor networks, and cloud services, the system automates routine plant care tasks such as watering, environmental monitoring, and security. At its core, the project utilizes the ESP8266 NodeMCU microcontroller to interface with various sensors and actuators, collecting real-time environmental data including soil moisture, temperature, humidity, and motion. The system processes this data and transmits it to the **Blynk IoT cloud**, enabling users to remotely visualize and control plant conditions through a smartphone or web application. This automation not only optimizes water usage and reduces manual effort but also introduces motion-based alerts to safeguard the plants from human or animal disturbances. The solution is ideal for smart homes, greenhouses, and small-scale agricultural applications aiming for sustainable, tech-enabled farming.

#### Existing System

Conventional plant care systems rely heavily on manual observation and intervention. Gardeners or farmers must inspect soil conditions, determine watering needs, and physically operate irrigation systems. This method is time-consuming, inconsistent, and lacks precision. Traditional systems do not support remote control, real-time data monitoring, or automated decision-making. Additionally, there is no security mechanism to detect disturbances around the plants, leaving them vulnerable to theft or damage..

###### Problems in Existing System

* **High Manual Effort**: Continuous physical presence is required for monitoring and watering.
* **Over/Underwatering**: No real-time moisture detection leads to improper irrigation.
* **Lack of Automation**: Systems are passive and require human decision-making.
* **No Remote Access**: Users must be physically present to interact with the system.
* **Absence of Security**: No alerts or actions against unauthorized access or movement.

#### Proposed System

The proposed **Smart Plant Monitoring System** incorporates smart sensing, cloud communication, and automation to address the drawbacks of traditional plant care. It features:

**ESP8266 NodeMCU** for microcontroller operations and Wi-Fi communication.

**Soil Moisture Sensor** to detect when watering is necessary.

**DHT11 Sensor** to measure ambient temperature and humidity.

**PIR Motion Sensor** to detect motion around the plant and trigger alerts.

**Blynk IoT Platform** for real-time monitoring, mobile notifications, and remote control.

**Relay-controlled DC Pump** that activates automatically based on sensor readings.

###### Advantages of proposed system

 **Automated Plant Watering**: Automatically waters plants when soil moisture drops below a threshold.

 **Water Conservation**: Irrigates only when necessary, minimizing wastage.

 **Real-Time Monitoring**: Displays sensor readings live on Blynk dashboard.

 **Remote Access**: View and control the system from anywhere via mobile or web.

 **Security F** **eature**: Motion detection system alerts users to plant disturbances.

# REQUIREMENT ANALYSIS

## REQUIREMENT ANALYSIS

#### Functional Requirements

The Smart Plant Monitoring System must fulfill the following key functionalities:

1. **Admin**

* Configures the Blynk IoT dashboard .
* Manages user access and system settings.
* Monitors overall system performance, sensor data, and alerts.

1. **User**

* Monitors real-time soil moisture, temperature, and humidity data via the Blynk .
* Controls the water pump manually or lets it operate automatically.
* Receives alerts for low soil moisture, motion detection, or environmental changes.

1. **Application**

* Collects real-time data from sensors and displays it on the Blynk dashboard.
* Sends alerts and notifications for motion detection, and security.
* Allows manual override of the water pump via a mobile app.

#### Non-Functional Requirements

• **Performance** – Real-time data updates with minimal delay for quick decision-making.

• **Fast & Responsive** – Low-latency transmission and instant feedback from sensors.

• **Scalable** – Supports more sensors and expanded monitoring without performance loss.

• **Reliable** – Accurate condition tracking with stable data delivery to the app.

• **Power Efficient** – Optimized for low energy use to extend sensor battery life.

• **User-Friendly** – Clean, intuitive UI with visual dashboards and reports.

• **Sensor-Compatible** – Works with various plant sensors (moisture, temperature, etc.).

#### Computational Resource Requirements

**2.3.1 Hardware Requirements**

1. ESP8266 Node MCU – Microcontroller for data processing & Wi Fi connectivity.
2. Soil Moisture Sensor – Measures soil moisture levels.
3. PIR Motion Sensor – Detects motion near the plants.
4. Relay Module – Controls the water pump.
5. Water Pump – Supplies water when needed.
6. DHT11 Sensor – Measures temperature & humidity.
7. 18650 Battery (or Power Supply) – Provides power to the system.
8. Breadboard & Jumper Wires – For circuit connections.
9. 16X2 I2C LCD – For displaying the values.
10. Tactile Button – Allows users to turn the water pump ON/OFF manually if needed

**2.3.2 Software Requirements**

1. Blynk IoT Platform – For real-time monitoring and remote control.

2. Arduino IDE – Used to program the ESP8266.

3. Blynk Library & ESP8266 WiFi Library – Enables communication with Blynk

### Life Cycle Model

Waterfall Model is a classical software development methodology that was first introduced by Winston W. Royce in 1970. It is a linear and sequential approach to software development that consists of several phases that must be completed in a specific order. The phases include:

**Requirements Gathering and Analysis:** The first phase involves gathering requirements from stakeholders and analyzing them to understand the scope and objectives of the project.

**Design:** Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.

**Implementation:** The implementation phase involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.

**Testing:** In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.

**Deployment:** Once the software has been tested and approved, it is deployed to the production environment.

**Maintenance:** The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time

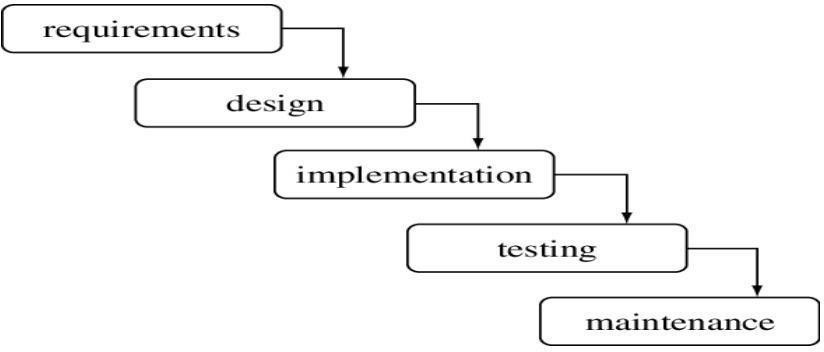


Fig 2.4 Water Fall Model

# ARCHITECTURE

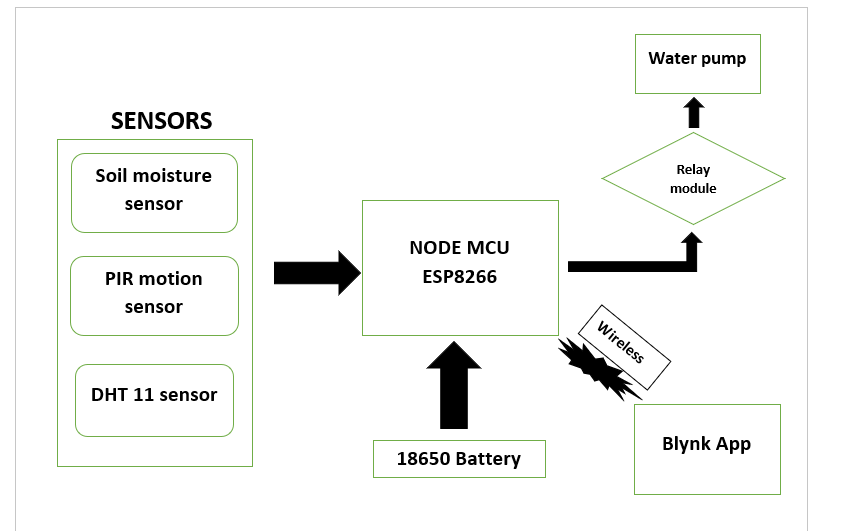
**3 Design and Architecture**

Project architecture represents number of components we are using as a part of our project and the flow of request processing i.e. what components in processing the request and in which order. An architecture description is a formal description and representation of a system organized in a way that supports reasoning about the structure of the system. Architecture is of two types. They are

1. Hardware Architecture
2. Technical Architecture

### Hardware Architecture

### 3.1.1 Block Diagram / Hardware Architecture



Fi g 3.1.1 Block diagram

**3.1.2 Description of hardware components**

**3.1.3 ARDUINO ESP8266:**

The **ESP8266** is a remarkable, budget-friendly Wi-Fi enabled microcontroller chip, designed by Espressif Systems, that has revolutionized the world of embedded systems and the Internet of Things (IoT). What makes it particularly popular is its ability to be programmed using the familiar **Arduino Integrated Development Environment (IDE)**. Essentially, the "Arduino ESP8266" refers to leveraging the robust capabilities of the ESP8266 — which includes a powerful 32-bit RISC processor, built-in Wi-Fi (802.11 b/g/n), and a range of GPIO pins for interacting with sensors and actuators — all while writing code in the user-friendly Arduino language (a simplified C++). This seamless integration, facilitated by the "Arduino core for ESP8266," means that anyone familiar with Arduino can easily add network connectivity to their projects, from creating smart home devices and environmental monitors to building simple web servers, making IoT development incredibly accessible and cost-effective

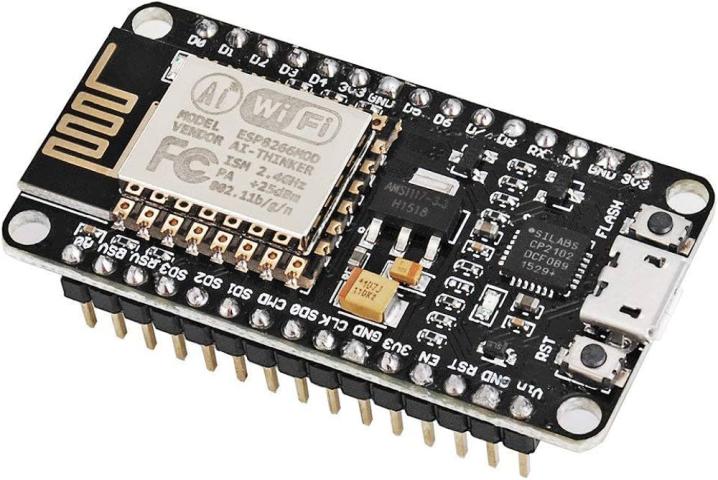


Fig 3.1.3 Arduino ESP8266

**3.1.4 Soil moisture:**

A soil moisture sensor is a device that measures the amount of water in the soil. It helps determine when and how much to water crops, gardens, or lawns, ensuring optimal plant growth while preventing overwatering. These sensors work by measuring a property of the soil that is related to the moisture content, such as electrical resistance or dielectric constant.

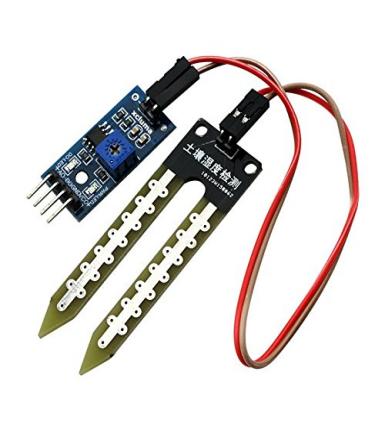
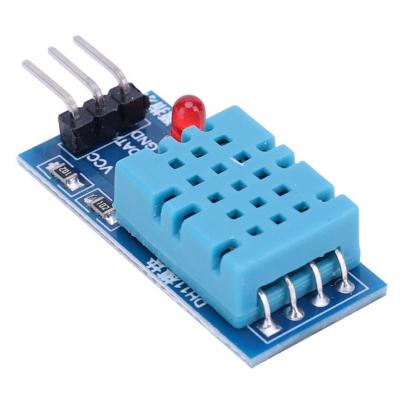


Fig 3.1.4 soil moisture

**3.1.5 DHT11 Sensor:**

Measures temperature and humidity, uses a single-wire digital interface, low-cost, provides accurate readings for basic applications, operates in the range of 0-50°C (temperature) and 20-80% RH (humidity), commonly used in weather stations, smart homes, and IoT projects.

Fig 3.1.5 DHT11 sensor

**3.1.6 18650 Battery:**

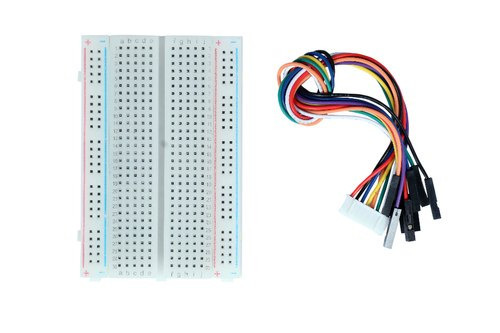
Rechargeable lithium-ion battery, widely used in portable devices, provides high energy density, widely used in projects requiring reliable power, such as IoT systems, smart devices, and robotics.



Fig 3.1.6 18650 battery

**3.1.7 Breadboard & Jumper Wires**:

Breadboard & Jumper Wires: A breadboard allows for quick prototyping of electronic circuits without soldering, while jumper wires are used to make temporary connections between components, enabling easy circuit assembly and testing.

 Fig 3.1.7 breadboard and jumper wires

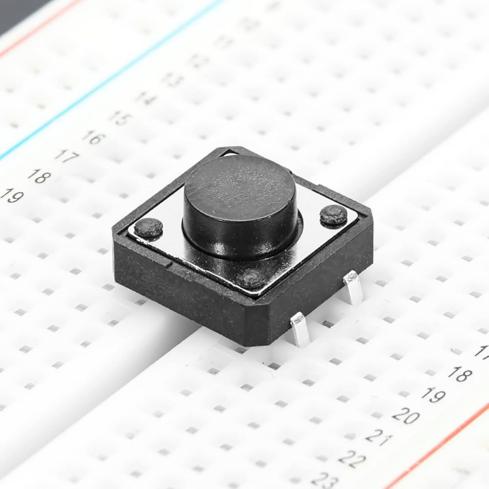
**3.1.8 16x2 I2C LCD:**

A screen that shows 16 characters across 2 lines. Is commonly used in projects to display text or data with microcontrollers like Arduino or ESP8266

 Fig 3.1.8 16x2 I2C LCD

**3.1.9 Tactile Button :**

A small, push-button switch that provides physical feedback when pressed. It is commonly used in circuits to trigger actions or inputs, such as starting or stopping a process in electronics and microcontroller projects.

 Fig 3.1.9 tactile button

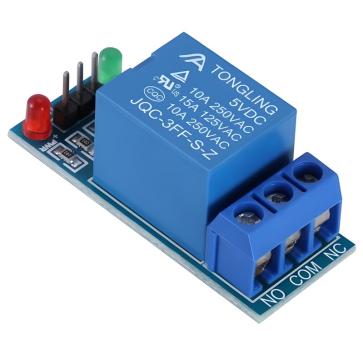
**3.10 PIR motion detector:**

A PIR (Passive Infrared) motion sensor is used to detect movement by sensing infrared (IR) radiation (heat) emitted by objects, typically humans or animals.

Fig 3.10 PIR motion detector

**3.11 Relay module:**

A relay module allows a microcontroller to control a high-voltage water pump by acting as an electrically operated switch. It isolates and safely switches the pump ON/OFF based on signals. This setup is essential in automated irrigation systems.

 Fig 3.11 relay module

**3.12 Waterpump:**

A water pump is a device used to move water from one place to another, commonly powered by electricity or batteries. In automation, it supplies water based on sensor data (. It is often controlled using a relay module connected to a microcontroller.

Fig 3.12 water pump

**Circuit diagram**

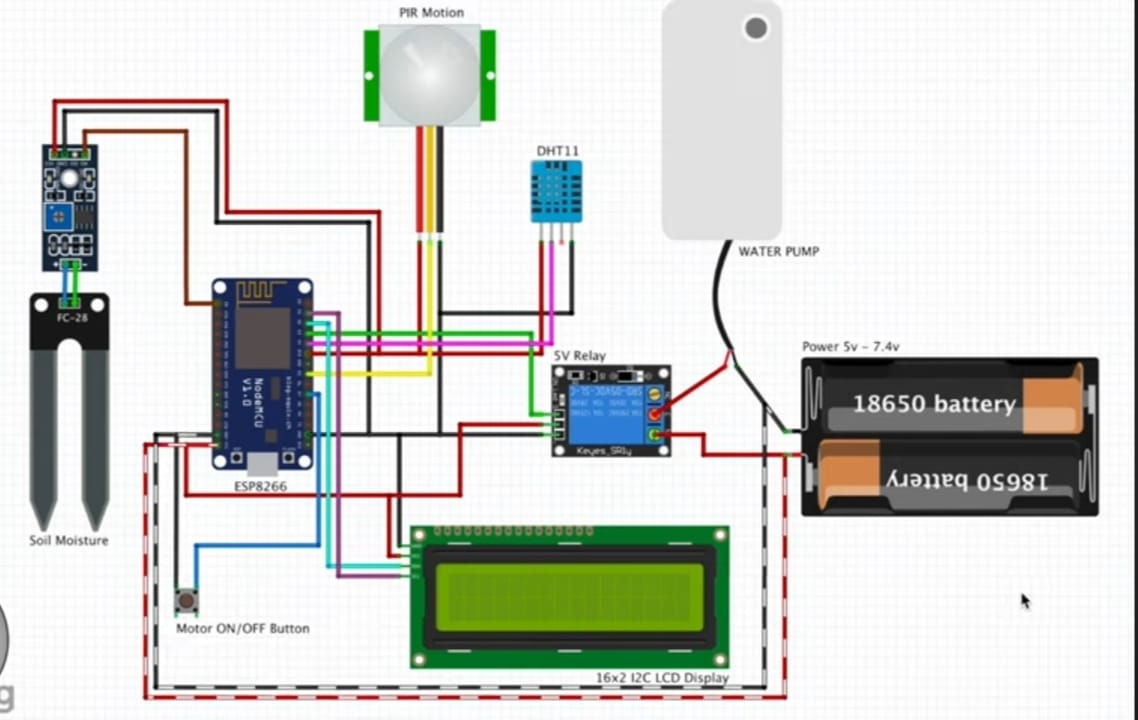


Fig 3.13 circuit diagram

### Technical Architecture:

**3.2.1 Data Flow Diagram:**

A Data Flow Diagram is a graphical representation used to visualize the flow of data within a system. It shows how data moves from one part of a system to another, what processes are involved, and where the data is stored. The primary purpose of a DFD is to model the system's information processes, focusing on how input data is transformed into output data

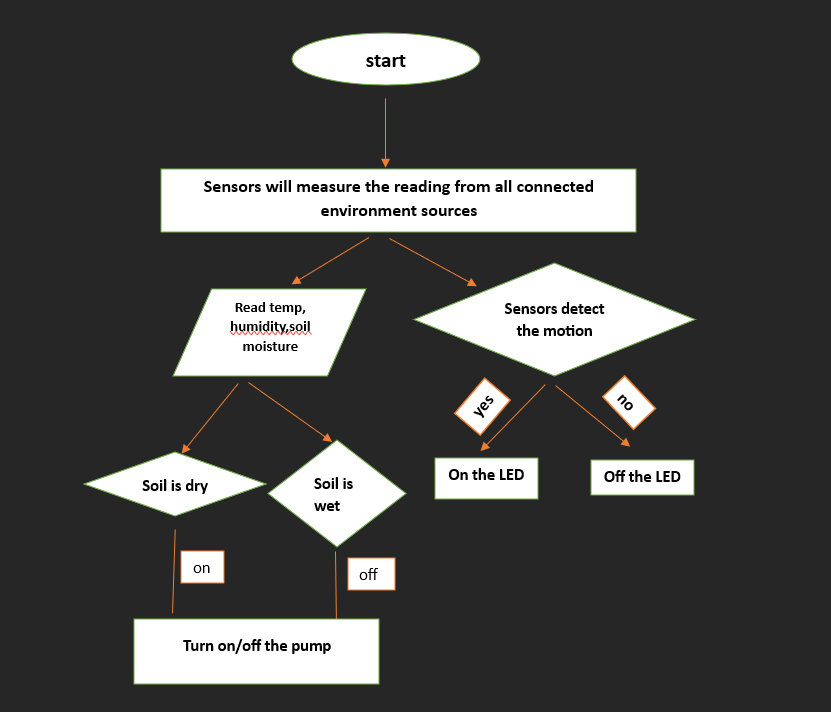
****

Fig 3.2.1 Data Flow diagram for Smart Plant monitoring system Using IoT

# IMPLEMENTATION

## IMPLEMENTATION

### Technologies

This Smart Plant Monitoring System is implemented using **C programming** on the **Arduino IDE**, tailored for the **ESP8266 NodeMCU** microcontroller. The system utilizes IoT and embedded programming principles to create an automated plant care system with real-time monitoring and control capabilities.

* + 1. **C Programming**

C is a powerful procedural programming language commonly used for low-level system development and embedded programming. In this project, C is used to program the NodeMCU board to interact with sensors, actuators, and the Blynk IoT platform.

**4.1.2 Features of C in Embedded Systems**

1. **Low-Level Access**: Allows direct interaction with hardware components such as sensors and pins.
2. **Efficient Execution**: Code runs quickly and is memory-efficient, suitable for microcontroller-based systems.
3. **Portability**: Can be ported across different microcontroller platforms with minor changes.
4. **Modular Structure**: Supports modular programming through functions and libraries.
5. **Real-Time Response**: Ideal for systems requiring immediate feedback and control.

**4.2 Technologies and Components Used**

**Microcontroller**

* **ESP8266 NodeMCU**: A Wi-Fi-enabled microcontroller used for reading sensor data and sending it to the Blynk cloud. It supports OTA updates and low-power operations, making it ideal for IoT applications.

**Sensors and Actuators**

* **DHT11 Sensor**: Measures temperature and humidity; the values are displayed on an LCD and sent to Blynk.
* **Soil Moisture Sensor**: Detects soil water levels and determines the need for irrigation.
* **PIR Motion Sensor**: Detects motion around the plant area for added security.
* **Relay Module**: Controls a water pump for irrigation based on soil moisture.
* **Push Button**: Manual override for relay control.
* **LCD (I2C 16x2)**: Displays real-time temperature, humidity, soil moisture, motion, and water pump status.

**IoT Platform**

* **Blynk Cloud**: Used for remote data monitoring, relay control, and event logging through a mobile application.

**4.3 Working of the System**

The ESP8266 initializes all sensor pins and connects to the Blynk cloud using Wi-Fi credentials.

**DHT11** and **Soil Moisture Sensors** send data every few milliseconds using BlynkTimer.

If soil moisture falls below a set threshold, the relay is triggered to power on the water pump.

The **PIR sensor** continuously checks for motion; if detected, a warning is logged via Blynk.

The user can also **manually control the water pump** via a **mobile app or physical push button**.

Real-time values are displayed on an **LCD screen**, showing temperature (T), humidity (H), soil moisture (S), motion detection (M:ON/OFF), and water pump status (W:ON/OFF)

**Applications of C Programming in Embedded and IoT Systems**

1. **Embedded Systems Programming**: C is ideal for microcontroller programming due to its close-to-hardware nature, making it suitable for interacting with GPIOs, timers, and sensors.
2. **IoT Device Development**: C is widely used in IoT firmware for devices like the ESP8266, handling real-time data acquisition, decision-making, and cloud communication.
3. **Automation Projects**: With its speed and efficiency, C enables automation systems such as smart plant watering, environmental monitoring, and actuator control.
4. **Consumer Electronics**: Many devices (e.g., smart home appliances, wearables) are powered by firmware written in C.
5. **Sensor Interfacing**: C simplifies the integration of various sensors (temperature, humidity, motion) and modules (LCDs, relays) in microcontroller-based setups.
6. **Real-time Systems**: Its deterministic performance makes C suitable for systems needing instant feedback and real-time control, such as motion alerts or irrigation.

**Advantages of C in IoT and Embedded Projects**

1. **Low-Level Hardware Control**: Direct access to hardware registers and memory allows precise control over pins, sensors, and modules.
2. **Efficient Execution**: C programs are fast and consume less memory, making them perfect for resource-constrained microcontrollers like the ESP8266.
3. **Modularity**: Functions, macros, and libraries promote clean and reusable code, easing debugging and future enhancements.
4. **Portability**: C code written for one microcontroller can often be reused for others with minor modifications.
5. **Real-Time Capability**: C supports the creation of highly responsive systems that need minimal delays or lag.

**Disadvantages of C in IoT Projects**

1. **Manual Memory Management**: Unlike high-level languages, C requires careful handling of memory, which can lead to bugs or crashes if mismanaged.
2. **No Built-in Networking Libraries**: For Wi-Fi or cloud communication, developers must rely on third-party libraries (e.g., Blynk or ESP8266WiFi).
3. **Steeper Learning Curve**: C's syntax and debugging process may be challenging for beginners compared to Python or JavaScript.
4. **Less Abstraction**: More lines of code are needed to achieve tasks that are simpler in higher-level languages.

**Why C Stands Out in Embedded Development**

C remains the dominant language for embedded and low-level system development due to its unparalleled control over hardware, efficiency, and portability. In projects like the Smart Plant Monitoring System, C enables direct sensor interfacing, real-time processing, and communication with IoT platforms like Blynk.

Whether you're automating irrigation or monitoring environmental parameters, C offers a balance of speed, control, and reliability unmatched by most modern languages in embedded domains. **C Data Types in Embedded Context**

Like Python, C offers several data types — each suited for different operations and memory usage, especially in microcontroller environments:

**1. Basic Data Types**

* int: Integer values (e.g., int moisture = 30;)
* float: Floating-point numbers (e.g., float temp = 27.5;)
* char: Stores single characters or bytes (e.g., char status = 'A';)
* double: Higher-precision float, rarely used in embedded C due to memory

**2. Derived Data Types**

* **Arrays**: Collection of elements of the same type (e.g., int sensorValues[5];)
* **Pointers**: Stores the address of another variable; heavily used for memory manipulation.
* **Structures** (struct): Custom grouping of multiple variables under one name used for grouping sensor data.
* **Unions**: Similar to structures but shares memory among all members.

**3. Custom Types**

typedef: Used to create new data type names.

enum: Used to define named integral constants (e.g., enum status {ON, OFF]

#### Pseudo Code

/\* Connections

Relay. D3

Btn. D7

Soil. A0

PIR. D5

SDA. D2

SCL. D1

Temp. D4

\*///Include the library files

#include <LiquidCrystal\_I2C.h>

#define BLYNK\_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <DHT.h>

//Initialize the LCD display

LiquidCrystal\_I2C lcd(0x3F, 16, 2);

char auth[] = " "; //Enter your Blynk Auth token

char ssid[] = " "; //Enter your WIFI SSID

char pass[] = " "; //Enter your WIFI Password

DHT dht(D4, DHT11);//(DHT sensor pin,sensor type) D4 DHT11 Temperature Sensor

BlynkTimer timer;

//Define component pins

#define soil A0 //A0 Soil Moisture Sensor

#define PIR D5 //D5 PIR Motion Sensor

int PIR\_ToggleValue;

void checkPhysicalButton();

int relay1State = LOW;

int pushButton1State = HIGH;

#define RELAY\_PIN\_1 D3 //D3 Relay

#define PUSH\_BUTTON\_1 D7 //D7 Button

#define VPIN\_BUTTON\_1 V12

//Create three variables for pressure

double T, P;

char status;

void setup() {

Serial.begin(9600);

lcd.begin();

lcd.backlight();

pinMode(PIR, INPUT);

pinMode(RELAY\_PIN\_1, OUTPUT);

digitalWrite(RELAY\_PIN\_1, LOW);

pinMode(PUSH\_BUTTON\_1, INPUT\_PULLUP);

digitalWrite(RELAY\_PIN\_1, relay1State);

Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);

dht.begin();

lcd.setCursor(0, 0);

lcd.print(" Initializing ");

for (int a = 5; a <= 10; a++) {

lcd.setCursor(a, 1);

lcd.print(".");

}}BLYNK\_WRITE(V6)

{PIR\_ToggleValue = param.asInt();

}BLYNK\_CONNECTED() {

// Request the latest state from the server

Blynk.syncVirtual(VPIN\_BUTTON\_1);

}BLYNK\_WRITE(VPIN\_BUTTON\_1) {

Timer.run();

# TESTING

## TESTING

**5.1 Overview**

Software testing is a critical phase in the development lifecycle aimed at ensuring that the system meets its requirements and performs reliably in real-world conditions. For the Smart Plant Monitoring System, testing ensures that sensor data is accurately collected, transmitted, and responded to through automation. Given the system’s role in real-time plant care, testing also guarantees robustness, correctness, and user trust.

**5.1.1 Dimensions of Testing**

The system is tested across multiple dimensions to ensure full functionality:

* Application Layers: Sensors, microcontroller, cloud server (Blynk), and user interface
* Testing Scale: Unit, integration, system, and acceptance
* Testing Types: Functional, performance, security, usability
* Methodologies: White-box for hardware logic; black-box for UI and cloud interaction

**5.1.2 Stages of Testing**

**5.1.3 Unit Testing**

Objective: Validate individual components such as sensors and actuators.  
Examples:

Soil moisture sensor accuracy testing

Temperature and humidity readings validation

Motion sensor triggering under presence

Water pump activation based on threshold moisture levels  
Method: White-box testing

**5.1.4 Integration Testing**

Objective: Ensure different modules interact correctly.  
Examples:

* ESP8266 receiving and processing sensor data
* Data correctly transmitted to Blynk cloud
* Blynk commands triggering hardware (e.g., water pump)  
  Method: Combination of white-box and black-box testing

**5.1.5 System Testing**

Objective: Test the full system workflow in realistic conditions.  
Examples:

* Real-time monitoring from app interface
* Water pump operation based on actual moisture levels
* UI displaying updated sensor values
* Alerts on motion detection  
  Method: Black-box testing

**5.1.6 Acceptance Testing**

Objective: Validate system against user expectations and usability.  
Examples:

* Remote control via Blynk works correctly
* Users can view live environmental data
* Alerts/notifications for movement or moisture thresholds  
  Method: Black-box testing with real-world user feedback

**5.2 Types of Testing**

**5.2.1 Black-box Testing**

Focuses on input-output behavior without internal code visibility.  
Applied in system and acceptance testing to validate UI, alerts, automation, and cloud interaction.

5.2.2 White-box Testing

Tests internal logic and sensor-actuator code on the ESP8266 board.  
Applied in unit testing for accurate condition checks and response logic.

Techniques Used:

* Statement Coverage
* Branch Coverage
* Path Coverage

#### 5.3Test Cases

Table 5.3.1 System Startup

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test Case**  **Description** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | System Startup | Turn on the system | System should initialize successfully | System initialize d | Pass |

Table 5.3.2 Moistuer Checking

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test Case Description** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 2 | Soil Moisture Check | Moisture sensor reading | Sensor should detect soil moisture level | Sensor detected level | Pass |

Table 5.3.3 Water Dispensing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test Case Description** | **Test Data** | **Expectd Result** | **Actual Result** | **Pass/Fail** |
| 3 | Water Dispensing | Dry soil detected | Water pump should activate | Water pump activated | Pass |

Table 5.3.4 Motion detection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test Case**  **Description** | **Test Data** | **Expectd Result** | **Actual Result** | **Pass/Fail** |
| 4 | Motion Detection Check | Any motion  detected | LED in blynk should be  Activated | LED  Blinked | Pass |

Table 5.3.5 Temperature and humidity

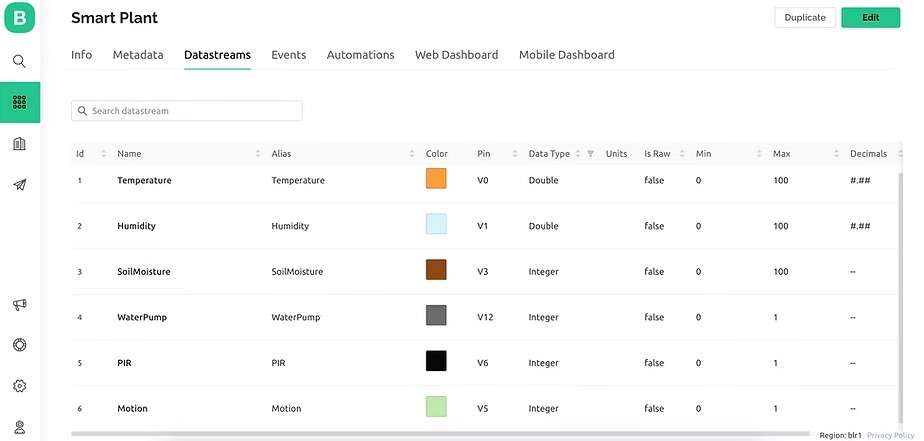
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test Case Description** | **Test Data** | **Expectd Result** | **Actual Result** | **Pass/Fail** |
| 5 | Temp and Humidity check | Automatic temp and humidity readings | Readings should be displayed in Blynk | Blynk shows the measurements | Pass |

Table 5.3.6 System Shutdown

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case**  **Description** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 6 | System Shutdown | Turn off the system | System should shut  down safely | System shut down | Pass |

# SCREENSHOTS

## SCREENSHOTS

****

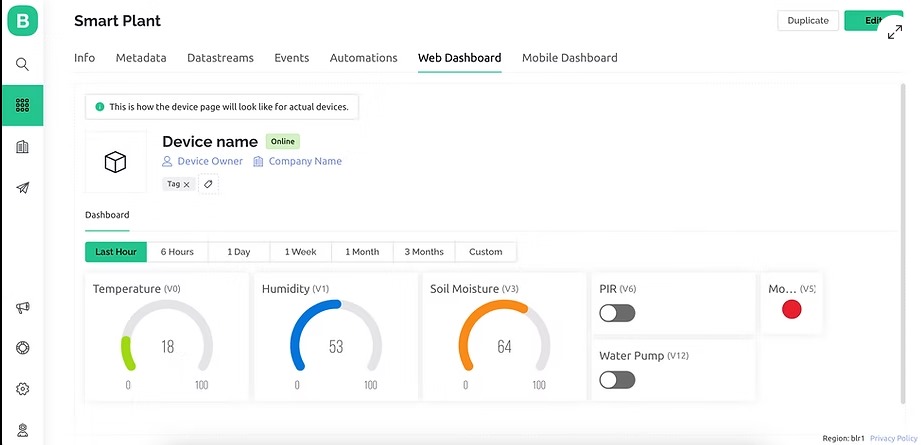
****

Fig 6.1 Blynk IoT dashboard

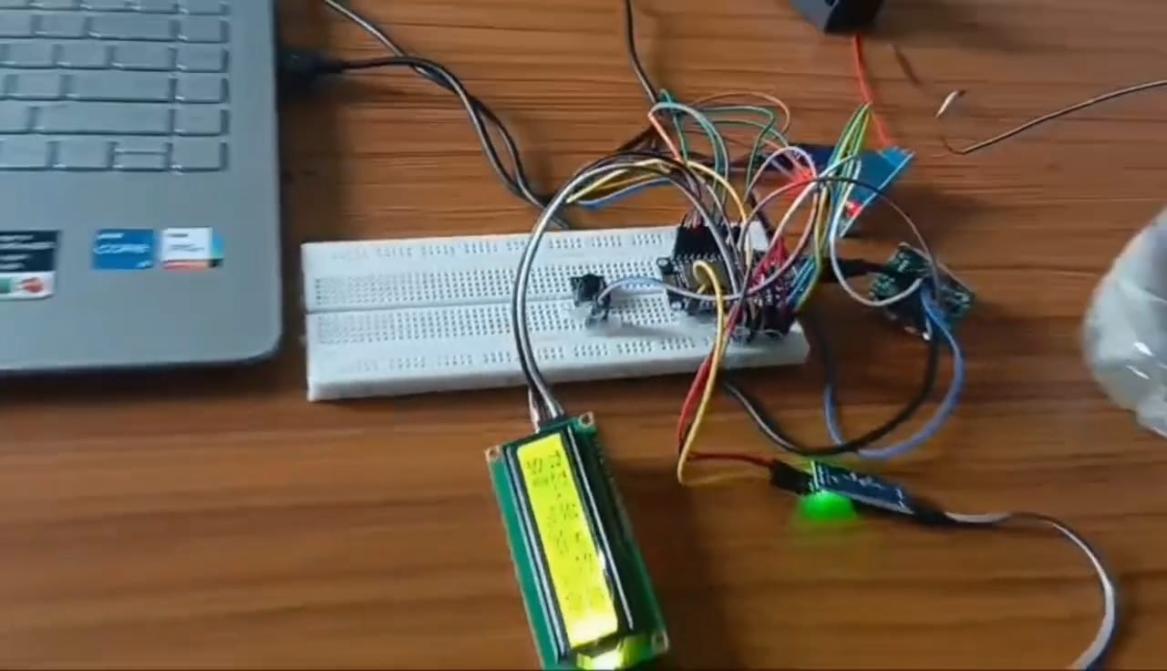


Fig 6.2 Practical set up



Fig 6.3 wet and dry soil



Fig 6.4 Soil moisture readings when soil sensor is kept in wet soil

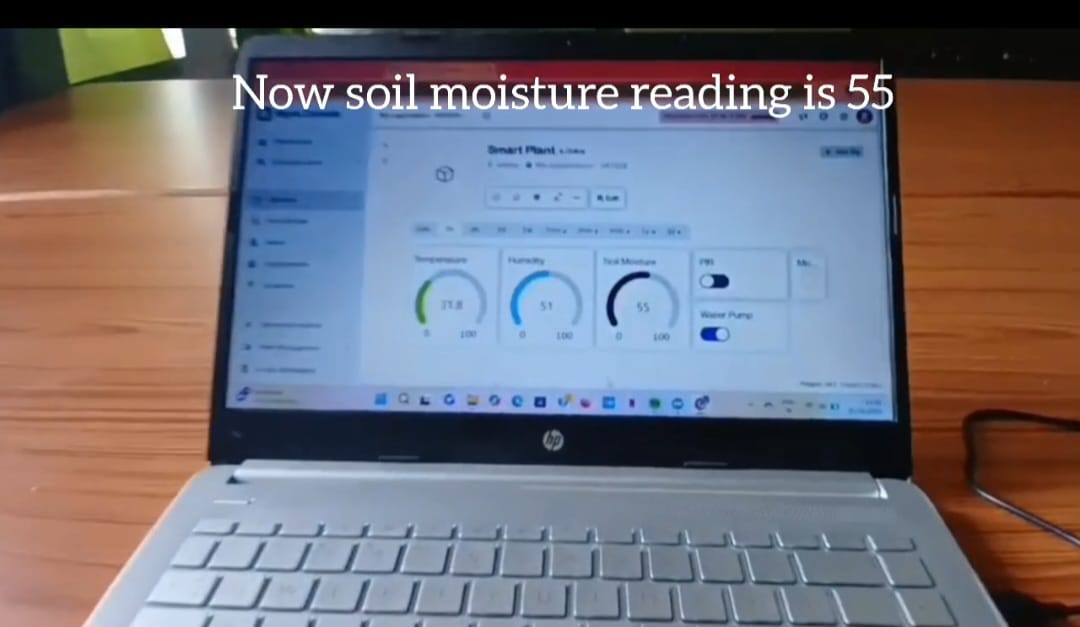
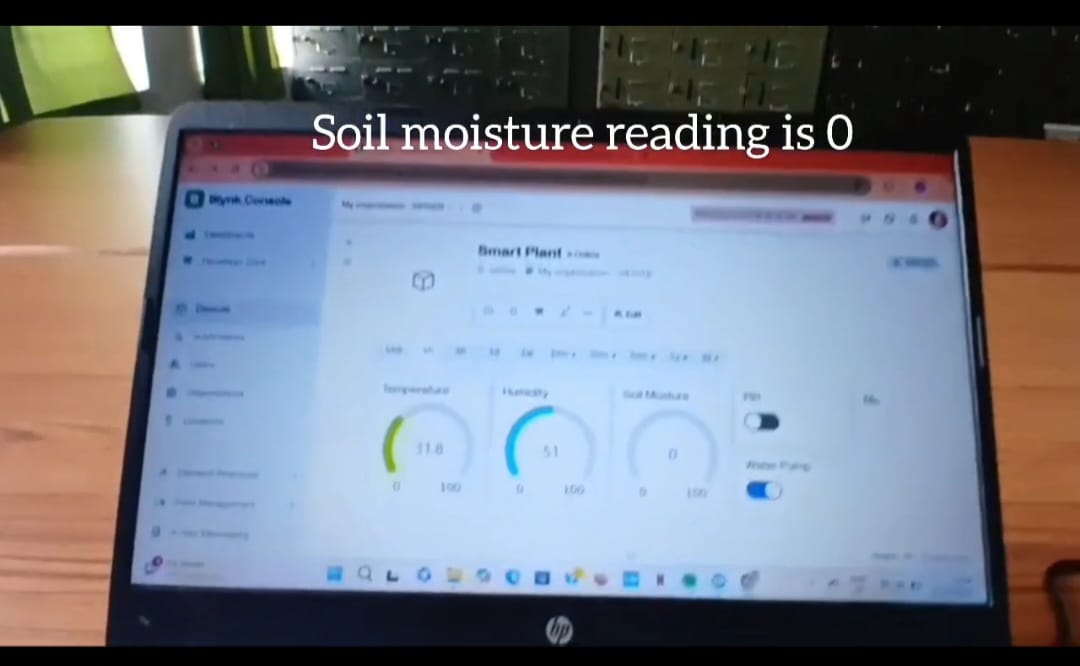
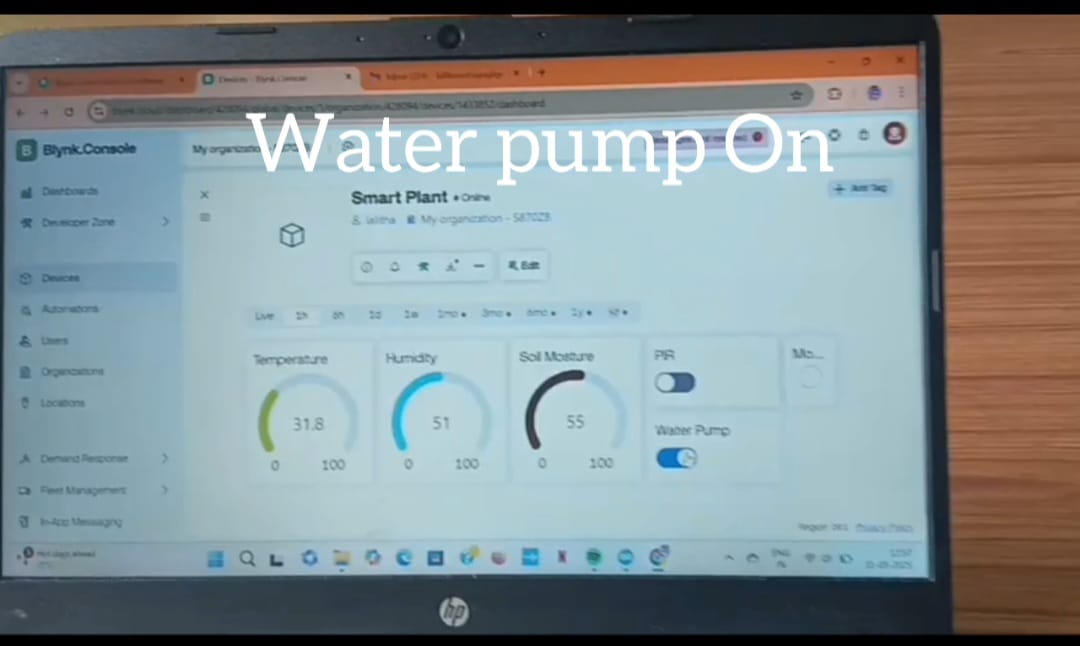
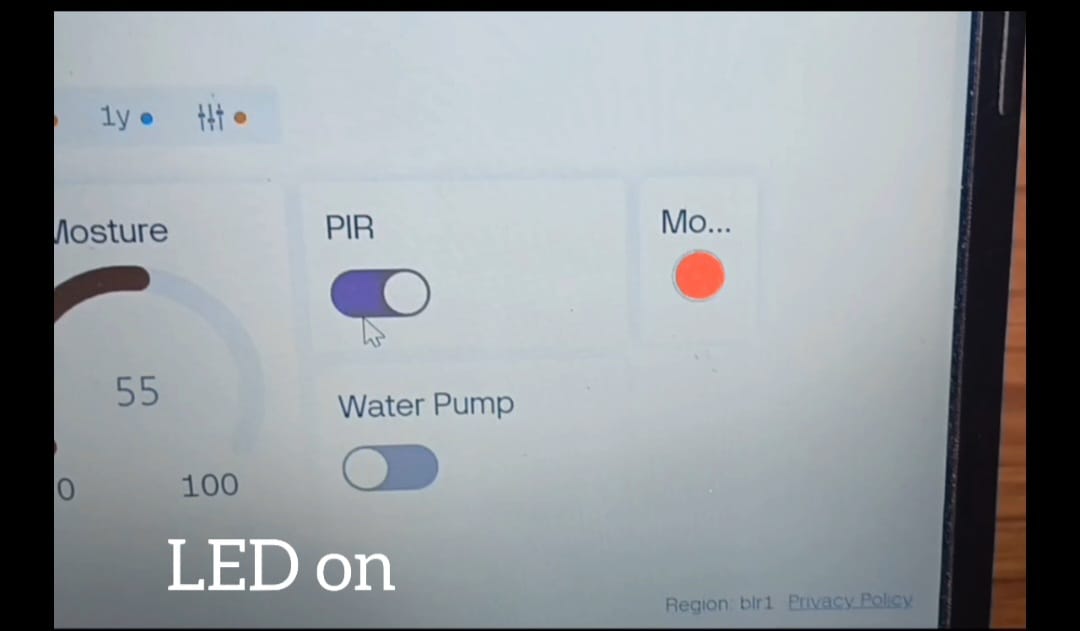


Fig 6.7 Digital moisture readings and motion detection and automatic water pump on [all 4 fig]

CONCLUSION AND FUTURE SCOPE

## CONCLUSION AND FUTURE SCOPE

#### Conclusion

The Smart Plant Monitoring System demonstrates how affordable and accessible sensor technologies can be used to effectively monitor essential environmental parameters such as soil moisture, temperature, humidity, and motion. By automating irrigation and enabling real-time monitoring through a user-friendly mobile application, the system reduces manual labor, improves water efficiency, and enhances plant security. Its adaptability to various environments like home gardens, greenhouses, and small-scale farms makes it a practical solution for modern agriculture..

#### Future Scope

i. The system can be further enhanced by integrating real-time weather data to dynamically adjust irrigation schedules. For example, it can prevent watering if rainfall is forecasted, thus conserving water and resources.

ii. Artificial Intelligence (AI) and data analytics can be incorporated to analyze trends in plant health, soil quality, and environmental conditions. These insights can support smarter decision-making and efficient resource utilization.

iii. With the continuous evolution of IoT, the system can offer more advanced remote monitoring and control through cloud-based mobile or web platforms, making plant care more accessible, even from distant locations.

REFERENCES

## REFERENCES

#### [1] Divani, Drashti; Patil, Pallavi; Punjabi, Sunil. (2016). *Automated Plant Watering System*.

#### [2] N. Ananthi et al., “IoT Based Smart Soil Monitoring System for Agricultural Production”, *IEEE International Conference on Technological Innovations in ICT*, ITAR 2017.

#### [3] E. Sowmiya, S. Sivaranjani, “Smart System Monitoring on Soil Using Internet of Things”, *International Research Journal of Engineering and Technology (IRJET)*, Volume 4, Issue 2, February 2017.

#### [4] N. Sakthipriya, “An Effective Method for Crop Monitoring Using Wireless Sensor Network”, *Middle East Journal of Scientific Research*, Volume 5, Issue 2, March 2018.

#### [5] K. Shalini, K. Soundarya, “Smart Plant Monitoring System using IoT and Blynk”, *International Journal of Engineering Research & Technology (IJERT)*

Department of CSE(AI&ML), BRECW 34