**AN ENGINEERING PROJECT FINAL REPORT**

**ON**

**AGROSPHERE: IOT DRIVEN GREENHOUSE**

**Submitted By**

**Abnish Kumar Yadav (BEX 20/200210)**

**Himal Rawat (BEX 20/200248)**

**Pratham Adhikari (BEX 20/200211)**

**Sushil Poudel (BEX 20/200247)**

**Submitted To**

**The Department of ICT**

**In Partial fulfillment of requirement for the degree of Bachelor of Engineering in Electronics and communication**

****

**Cosmos College of Management Technology (Affiliated to Pokhara University)**

**Tutepani, Lalitpur, Nepal**

**2082 / 04 / 07**

**AN ENGINEERING FINAL PROJECT REPORT**

**ON**

**AGROSPHERE: IOT DRIVEN GREENHOUSE**

**SUBMITTED BY:**

**Abnish Kumar Yadav (BEX 20/200210)**

**Himal Rawat (BEX 20/200248)**

**Pratham Adhikari (BEX 20/200211)**

**Sushil Poudel (BEX 20/200247)**

**Under the Supervision of**

**Er. AMAN ANSARI**

**Submitted To**

**The Department of Information and Communication Technology**

**In Partial fulfilment of the requirement for the degree of**

**Bachelor of Engineering in Electronic and Communication**

**Cosmos College of Management and Technology**

**(Affiliated to Pokhara University)**

**Tutepani, Lalitpur, Nepal**

**2082 / 04 / 07**

**Cosmos College of Management and Technology**

**(Affiliated to Pokhara University)**

**Chapagaun, Lalitpur, Nepal**

**Ref No.:** **Date:**

### CERTIFICATION

This is to notify that the project titled **“AGROSPHERE: IOT DRIVEN GREENHOUSE”** submitted by **Abnish Kumar Yadav, Himal Rawat, Pratham Adhikari and Sushil Poudel** as a partial fulfillment of the requirement of the Bachelor Degree in Electronics and Communication Engineering has been completed under our supervision. We recommend the project for acceptance and approval.

Er. Aman Ansari

### APPROVAL

This final year project report entitled **“AGROSPHERE: IOT DRIVEN GREENHOUSE”** submitted by **Abnish Kumar Yadav, Himal Rawat, Pratham Adhikari** and **Sushil Poudel** as a partial fulfillment of the requirement of the Bachelor Degree in Electronics and Communication Engineering has been examined and accepted by us.

Panel of Examiners

Name: Signature

1. Mr. Aman Ansari ……………………….

(Supervisor)

2. Mr. Hemanta Joshi ……………………….

(External Examiner)

(Universal engineering & science college)

(Department Head of Computer)

3. Mr. Ram Kumar Sharma ……………………….

(Principal)

4. Mr. Ajaya Adhikari ……………………….

(HOD, Department of ICT)

### COPYRIGHT

The author has agreed that the library, University of Pokhara, Cosmos College of Management and Technology, may make this Engineering project report freely available for inspection. Moreover, the author has agreed that the permission for extensive copying of this report for scholarly purpose may be granted by the supervisor the work recorded herein or, in which the project work was done. Copying or publication or any other use of this project report for financial gain without approval of the college and the authors permission I prohibited.

Request for permission to copy or to make any use of the materials in this report in whole or in part should be addressed to:

Principal

Cosmos College of Management and Technology

Pokhara University

### ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to all faculty members who supported and encouraged us, without their guidance this project would not have been possible.

We are deeply grateful to Department of Information and Communications Technology, **Cosmos College of Management and Technology, Tutepani, Lalitpur** for providing us all the consultation and knowledge to conduct this project. We show our sincere gratitude to our teachers for effective suggestion and strengthening in the completion of this proposal.

We would also like to acknowledge the immense contribution of our supervisor **Er. Aman Ansari** for the kind support and guidance during project implementation phase as well as his inspiration to move forward for the completion of project on time.

Sincerely,

**Abnish Kumar Yadav (BEX 20/200210)**

**Himal Rawat (BEX 20/200248)**

**Pratham Adhikari (BEX 20/200211)**

**Sushil Poudel (BEX 20/200247)**

### ABSTRACT

The aim of this project is to design a greenhouse monitoring and controlling system based on the internet of things (IOT). A greenhouse is a covered area where plants grow and cultivate. It is also known as land of controlled crops and plants. Various greenhouse automation equipment like computer software and sensors are connected and used to collect data in the greenhouse environment to boost crop yields. This new innovative technology (IoT or the Internet of Things) makes use of numerous sensors linked to a central greenhouse environment control computer. The greenhouse sensor systems have elements that monitor and control temperature, humidity, soil moisture, illumination and read external weather conditions via a weather station. These are monitored and regulated using a microcontroller which is connected to sensors and external mechanical devices. Climate controlled greenhouses can help any grower get better at quality control or enhance crop yields, with advanced computer technology. There are other manual methods, but farmers who want to make the best of modern technology prefer to integrate the climate control system. This will help seasonal crops to grow throughout the year. A climate control system automates the greenhouse to reach the desired temperature as required by your crops’ growing process. The system monitors and handles humidity, temperature, soil moisture and illumination. This is accomplished by the way of real time sensors, that communicate wirelessly in the greenhouse, via Wi-Fi.

**Key words:** IOT, MQTT, Agrosphere, Sensors etc.

### ABBREVIATION

MQTT Message Queuing Telemetry Transport

IOT Internet of things

LDR Light Dependent Resistor

RF Radio Frequency

RH Relative Humidity

ADC Analog to Digital Converter

WI-FI Wireless Fidelity

LCD Liquid Crystal Display

### Table of contents

[**CERTIFICATION i**](#_Toc204687761)

[**APPROVAL ii**](#_Toc204687762)

[**COPYRIGHT iii**](#_Toc204687763)

[**ACKNOWLEDGEMENT iv**](#_Toc204687764)

[**ABSTRACT v**](#_Toc204687765)

[**ABBREVIATION vi**](#_Toc204687766)

[**Table of contents vii**](#_Toc204687767)

[**Table of figure ix**](#_Toc204687768)

[**List of TABLES xi**](#_Toc204687769)

[**CHAPTER 1 2**](#_Toc204687770)

[**1. INTRODUCTION 2**](#_Toc204687771)

[**1.1 Background 2**](#_Toc204687772)

[**1.2 Objectives 2**](#_Toc204687773)

[**1.3 Scope and Limitation 3**](#_Toc204687774)

[**CHAPTER 2 4**](#_Toc204687775)

[**2. LITRATURE REVIEW 4**](#_Toc204687776)

[**CHAPTER 3 5**](#_Toc204687777)

[**3. METHODOLOGY 5**](#_Toc204687778)

[**3.1 Block diagram 5**](#_Toc204687779)

[**3.2 Flowchart 6**](#_Toc204687780)

[**3.3 Circuit diagram 7**](#_Toc204687781)

[**CHAPTER 4 8**](#_Toc204687782)

[**4. COMPONENT DESCRIPTION 8**](#_Toc204687783)

[**CHAPTER 5 18**](#_Toc204687784)

[**5. INTERNET OF THINGS 18**](#_Toc204687785)

[**CHAPTER 6 19**](#_Toc204687786)

[**6. MQTT 19**](#_Toc204687787)

[**6.1 Introduction 19**](#_Toc204687788)

[**6.2 Components 20**](#_Toc204687789)

[**CHAPTER 7 22**](#_Toc204687790)

[**7. INTERFACING 22**](#_Toc204687791)

[**7.1 Dht11 With Arduino 22**](#_Toc204687792)

[**7.2 Soil Moisture With Arduino 23**](#_Toc204687793)

[**7.3 Relay Pump With Arduino 24**](#_Toc204687794)

[**CHAPTER 8 25**](#_Toc204687795)

[**8. WORKING PRINCIPLE 25**](#_Toc204687796)

[**CHAPTER 9 26**](#_Toc204687797)

[**9. OUTCOME 26**](#_Toc204687798)

[**CHAPTER 10 27**](#_Toc204687799)

[**10. RESULT AND DISCUSSION 27**](#_Toc204687800)

[**CHAPTER 11 28**](#_Toc204687801)

[**11. CONCLUSION AND FUTURE ENHANCEMENT 28**](#_Toc204687802)

[**12. REFERENCES 30**](#_Toc204687803)

[**13. APPENDIX 31**](#_Toc204687804)

[**13.1 BUDGET STRUCTURE 31**](#_Toc204687805)

[**13.2 WORKING SCHEDULE 32**](#_Toc204687806)

[**13.3 CODE OF ARDUINO 33**](#_Toc204687807)

[**13.4 ESP-32 CODE 37**](#_Toc204687808)

### Table of figure

[**Figure 3.1: Block diagram 5**](#_Toc204687099)

[**Figure 3.2: Flowchart 6**](#_Toc204687100)

[**Figure 3.3: Circuit diagram 7**](#_Toc204687101)

[**Figure 4.1: Arduino Uno 8**](#_Toc204687102)

[**Figure 4.2:ESP-32 9**](#_Toc204687103)

[**Figure 4.3: Soil sensor 10**](#_Toc204687104)

[**Figure 4.4: DHT11 Sensor 11**](#_Toc204687105)

[**Figure 4.5: PIR sensor 12**](#_Toc204687106)

[**Figure 4.6: Relay module 13**](#_Toc204687107)

[**Figure 4.7: Level shifter 13**](#_Toc204687108)

[**Figure 4.8: Water pump 14**](#_Toc204687109)

[**Figure 4.9: LDR sensor 15**](#_Toc204687110)

[**Figure 4.10: I2C LCD display 16**](#_Toc204687111)

[**Figure 4.11: DC fan 17**](#_Toc204687112)

[**Figure 6.1: MQTT in IOT 19**](#_Toc204687113)

[**Figure 6.2: MQTT dashboard 20**](#_Toc204687114)

[**Figure 7.1: DHT22 with Arduino 22**](#_Toc204687115)

[**Figure 7.2: Soil Moisture with Arduino 23**](#_Toc204687116)

[**Figure 7.3: Relay Pump with Arduino 24**](#_Toc204687117)

### List of TABLES

[**Table 1: DHT22 with Arduino 22**](#_Toc204687538)

[**Table 2: Soil Moisture with Arduino 23**](#_Toc204687539)

[**Table 3: Relay pump with Arduino 24**](#_Toc204687540)

[**Table 4: Budget Structure 31**](#_Toc204687541)

[**Table 5: Working Schedule 32**](#_Toc204687542)

### CHAPTER 1

# INTRODUCTION

## Background

Nepal is a country in the South Asian territory which is also known as agricultural country where most of the people depend on agriculture. It has suitable climate for the production of variety of crops. The agriculture sector in Nepal is still in progress. Most of the people using traditional farming methods which can be wasteful, inefficient and insufficient production of crops. Today’s world needed more health-conscious, producing quality crops. In order to solve the problem and supports in minimal cost and high reliability which holds significant potential for addressing the challenges faced by farmers in Nepal, the greenhouse monitoring and control system is proposed.

Statement of Problem

* Unstable Environmental Conditions: Variations in temperature, humidity, and soil moisture can adversely affect crop health.
* Limited Accessibility: Farmers cannot monitor or control greenhouse conditions remotely, limiting their ability to respond to sudden environmental changes.
* Inefficient Resource Utilization: Excessive or insufficient use of water, light, or ventilation can lead to resource wastage.
* High Labor Dependency: The constant need for human intervention increases labor costs and reduces operational efficiency.
* Unsustainable Practices: Inefficient resource usage and reliance on manual processes make traditional greenhouse systems less sustainable and environmentally friendly.

## Objectives

* Automation: Automatic activation of irrigation, fans, heaters, and lights based on sensor data
* Sustainability: To promote sustainable agricultural practices by optimizing resource usage and reducing dependency on manual labor.
* Real-Time Monitoring: To monitor critical parameters such as temperature, humidity, soil moisture, and light intensity in real-time using sensors.
* Remote Accessibility: To provide farmers with a mobile or web-based interface for remote monitoring and control of greenhouse operations.
* Alert System: To implement notifications or alerts for abnormal conditions (e.g., high temperature, low soil moisture) via SMS, email, or app notifications.
* To integrate ESP-32 with Wi-Fi module or lora for communication.

## Scope and Limitation

Scope

* + - Remote Control: Cloud-based access for monitoring and controlling system from anywhere

Limitation

* Doesn’t recommend fertilization of crops

### CHAPTER 2

# LITRATURE REVIEW

* **IoT in Agriculture**

IoT technology has transformed agriculture by enabling precise monitoring and management of environmental parameters. Studies highlight its potential to improve crop yields, reduce waste, and provide scalability for various agricultural operations *(Patil & Kale, 2016).*

* **Automation in Greenhouses**

Automation using IoT reduces manual intervention. Research shows that automated irrigation systems based on soil moisture sensors conserve water and ensure consistent plant hydration. Similarly, temperature regulation through fans and heaters, triggered by threshold values, enhances crop productivity *(Ali et al., 2017).*

* **Environmental Monitoring**

Greenhouse environments are sensitive to changes in temperature, humidity, soil moisture, and light intensity. Sensors such as DHT22, soil moisture probes, and light-dependent resistors (LDR) have been proven effective for accurate data collection *(Kumar et al., 2018).*

* **Sustainability in Greenhouse Management**

Studies by *Nair et al. (2018)* demonstrated that IoT-based greenhouse systems contribute to sustainability by optimizing resource use and reducing environmental impact. Automated irrigation systems, for example, minimize water wastage and ensure consistent plant health.

* **IoT Platforms for Data Management**

Cloud platforms such as Thing speak, Firebase, and Blynk enable real-time data visualization and storage. These platforms provide remote access to environmental data, allowing users to monitor and control greenhouse conditions from anywhere *(Sharma et al., 2019).*

### CHAPTER 3

# METHODOLOGY

## Block diagram

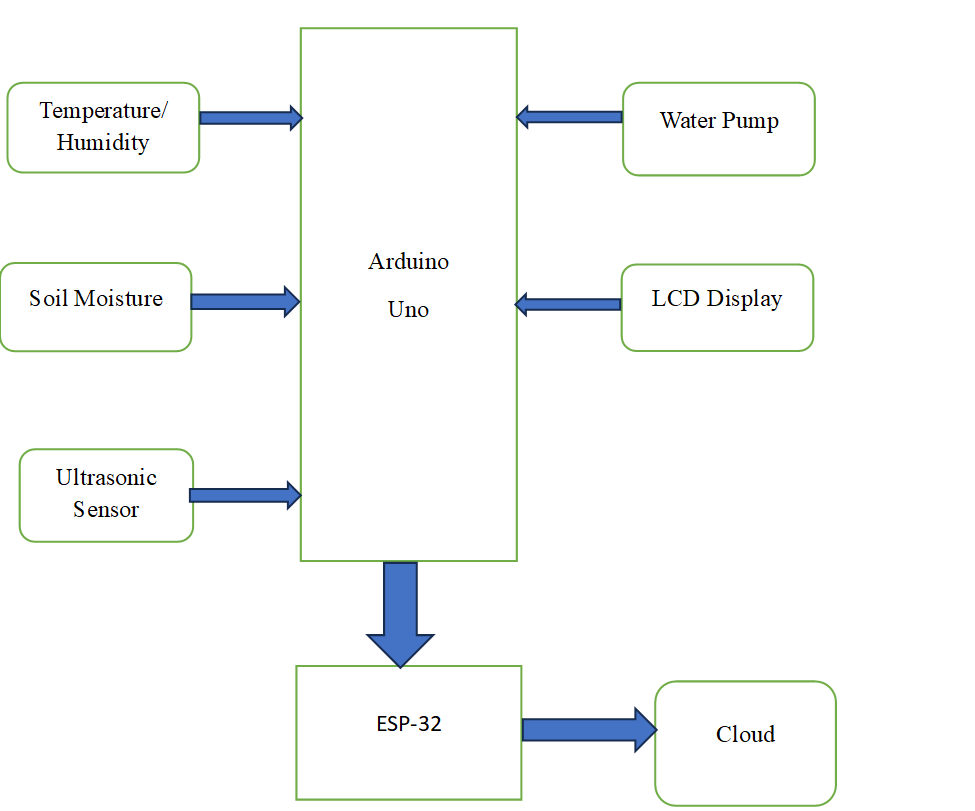


Figure 3.1: Block diagram

## Flowchart

Event triggered

Arduino Uno

Wi-Fi module

Check from Database/Cloud

Actuators

Applications

Figure 3.2: Flowchart

## Circuit diagram

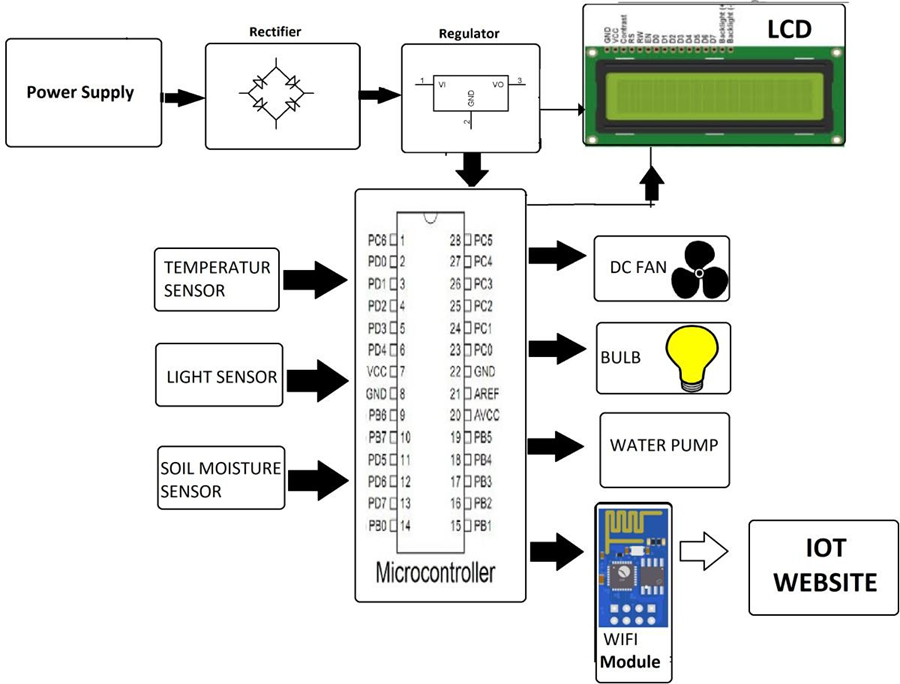


Figure 3.3: Circuit diagram

### CHAPTER 4

# COMPONENT DESCRIPTION

The main component uses in Agrosphere: IoT-Driven Greenhouse are as follows: -

1. **Arduino Uno**

Arduino is an open-source computer hardware and software company, project, and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense, and control objects in the physical world.



Figure 4.1: Arduino Uno

Specifications:

* It is an ATmega328P based Microcontroller
* The Operating Voltage of the Arduino is 5V
* The recommended input voltage ranges from 7V to 12V
* Digital input and output pins 14
* Digital input & output pins (PWM)-6
* Analog I/p pins are 6
* The speed of the CLK is 16 MHz

1. **ESP-32**



Figure 4.2:ESP-32

Specification

* + - 1. Processor:
* Dual-core Tensilica Xtensa LX6 processor (can operate independently).
  + - 1. Memory
* 520 KB SRAM.
  + - 1. Wireless Connectivity:
* Wi-Fi: 802.11 b/g/n with a frequency range of 2.4 GHz.
  + - 1. GPIO
* Up to 34 GPIO pins (varies depending on the package Supports multiplexing for multiple peripherals.
  + - 1. Analog Features:
* ADC (Analog-to-Digital Converter): 12-bit resolution. Up to 18 channels.
* DAC (Digital-to-Analog Converter):
* Watchdog timers.
  + - 1. Operating Voltage
* 3.3V typical.
  + - 1. Power Consumption:
* Active mode: ~160 mA.
* Deep sleep mode: ~5 µA

1. **Soil sensor**

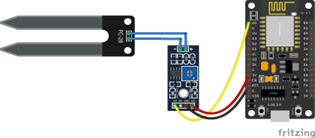


Figure 4.3: Soil sensor

1. Purpose:

* Measures the water content in soil to determine its moisture level.
* Commonly used in agriculture, gardening, and environmental monitoring.

1. Working Principle:

* Operates by measuring the soil's conductivity or capacitance.
* Wet soil conducts electricity better than dry soil, enabling moisture level detection.

1. Interfacing with Microcontroller:

* Connect the analog or digital output pin to the microcontroller (e.g., ESP32).
* Use ADC for precise moisture level readings if using analog output.

1. Operating Voltage:

* Typically operates between 3.3V and 5V, compatible with most microcontrollers.

1. **Humidity /temperature sensor**



Figure 4.4: DHT11 Sensor

Specification

1. Temperature Range:

* It can measure temperatures from -40°C to 80°C with an accuracy of ±0.5°C.

1. Humidity Range:

* It can measure humidity from 0% to 100% with an accuracy of ±2-5%.

1. Operating Voltage:

* The sensor operates with a supply voltage of 3.3V to 6V.

1. Output:

* The DHT22 provides a digital signal to communicate with a microcontroller

1. Interface:

* It uses a single-wire communication protocol, which simplifies the interfacing

1. **PIR sensor**

A Passive Infrared (PIR) sensor detects motion by sensing changes in infrared radiation, typically emitted by humans or animals. These sensors are widely used in security systems, automatic lighting, and various automation projects due to their low cost, ease of use, and low power consumption.



Figure 4.5: PIR sensor

Specification

* **Operating Voltage**: Typically ranges from 4.5V to 20V DC.
* **Output**: Digital signal (High/Low), often 3.3V TTL logic level.
* **Detection Range**: Up to 7 meters, with a field of view around 120°.
* **Power Consumption**: Low, often around 65mA during operation.
* **Operating Temperature**: Generally, between -15°C to +70°C.
* **Delay Time**: Adjustable, commonly between 0.5s to 200s.
* **Trigger Modes**: Repeatable (H) and Non-Repeatable (L) modes.

1. **Relay module**

A 1-channel 5V relay module typically has a 5V DC operating voltage and can switch a single high-powered device.

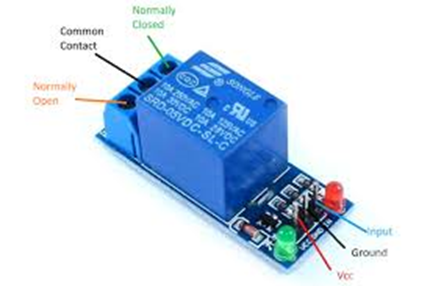


Figure 4.6: Relay module

1. **Level shifter**

A 5V to 3.3V level shifter converts a 5V logic signal to a 3.3V logic signal, and vice versa, allowing components operating at different voltage levels to communicate without damage. This is typically achieved using MOSFETs or voltage dividers, with the choice depending on whether the conversion needs to be unidirectional or bidirectional and the specific application requirements



Figure 4.7: Level shifter

1. **Water pump**



Figure 4.8: Water pump

Specification

* voltage (DC 5V, 12V, etc.)
* power (3W, 5W, etc.)
* flow rate (e.g., 50 GPH, 200 L/H)
* lift height (e.g., 0.8-1m)
* size (e.g., 1.5 x 1.5 x 1.1 inch).

1. **LDR sensor**

An LDR (Light Dependent Resistor), also known as a photoresistor, is a component whose resistance varies with the intensity of light it receives. Essentially, it's a light-sensitive resistor where resistance decreases as light levels increase.



Figure 4.9: LDR sensor

Specification

* **Operating Voltage:** 3V to 5V DC.
* **Operating Current:** Around 15 milliamps.
* **Output:** May offer both digital and analog outputs.
* **Digital Output:** A digital output (DO) provides a HIGH or LOW signal based on a set threshold for light intensity.
* **Analog Output:** An analog output (AO) provides a voltage level proportional to the light intensity.
* **Sensitivity Adjustment:** Many LDR modules include a potentiometer to adjust the sensitivity or trigger level of the digital output.

1. **lcd display**

An I2C LCD display, often a 16x2 configuration, offers a compact way to display text information using only two communication lines (SDA and SCL) for data transfer via the I2C protocol. These displays typically operate at 5V and require minimal current, with the backlight drawing additional current. They are widely used with microcontrollers like Arduino and feature a convenient four-pin connection (VCC, GND, SDA, SCL).

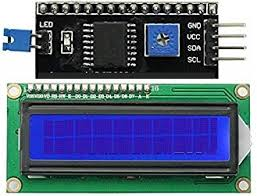


Figure 4.10: I2C LCD display

Specification

* **Display Capacity:** 16 characters x 2 lines.
* **Interface:** I2C (also known as IIC or TWI).
* **Supply Voltage:** Typically, 5V DC.
* **Operating Current:** Low, around 2mA typical at 5V, with backlight adding to the current draw.
* **I2C Address:** Usually 0x27 (default, but can vary).
* **Backlight:** Often blue with white characters, adjustable for contrast.
* **Communication:** Utilizes SDA (Serial Data) and SCL (Serial Clock) lines for communication.

1. **DC fan**

DC fans, commonly used for cooling applications, have a range of specifications that vary by model, but generally include operating voltage, current consumption, airflow, noise level, speed, lifespan, connector type, and dimensions. Typical operating voltages are 5V, 12V, and 24V.



Figure 4.11: DC fan

Specification

* **Operating Voltage:** Most DC fans operate on low voltages, such as 5V, 12V, or 24V.
* **Current Consumption:** This varies based on size and model, but a common range is 0.1A to 0.5A.
* **Airflow:** Measured in CFM (Cubic Feet per Minute), typically ranges from 20 CFM to 70 CFM.
* **Noise Level:** Typically measured in dBA (decibels) and generally falls between 20 dBA and 35 dBA.
* **Speed:** Measured in RPM (Revolutions Per Minute), commonly between 1000 RPM and 3000 RPM.

### CHAPTER 5

# INTERNET OF THINGS

The Internet of Things (IoT) is a technology that connects physical devices like sensors, appliances, and machines to the internet, allowing them to collect, send, and receive data without human involvement.

**Importances of iot**

* Automates manual work
* Saves time and energy
* Improves efficiency and accuracy
* Enables remote monitoring and control
* Supports real-time decision-making

**Component**

* Sensors: Measure physical parameters (e.g., temp, moisture)
* Actuators: Perform actions (e.g., turn on pump/fan)
* Microcontroller: Processes data and connects to Wi-Fi (e.g., ESP32, Arduino)
* Connectivity: Transfers data (Wi-Fi, Bluetooth, Zigbee, etc.)
* Cloud Platform: Stores and analyzes data (e.g., Blynk, Firebase)
* User Interface: to monitor/control remotely

### CHAPTER 6

# MQTT

## Introduction

MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe based messaging protocol designed for resource-constrained devices and low-bandwidth, high-latency networks. It's widely used in the Internet of Things (IoT) for efficient communication between sensors, actuators, and other devices. MQTT operates on a publish-subscribe model, where devices publish messages to specific topics and other devices subscribe to those topics to receive the messages.

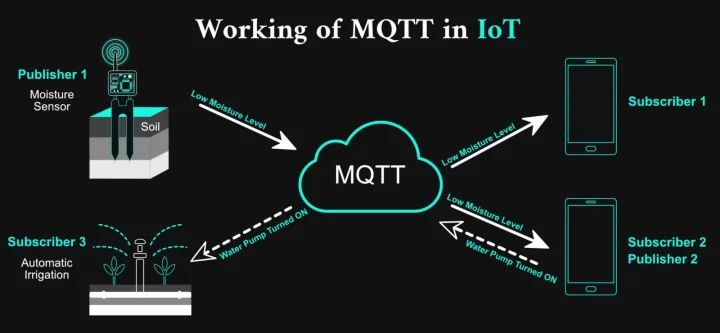


Figure 6.1: MQTT in IOT

## Components

* **Publisher**
* Role: The device or application that sends data (messages).
* Function: Publishes messages to a specific topic.
* Example: A DHT11 sensor with ESP32 publishing temperature to topic greenhouse/temp.
* **Subscriber**
* Role: The device or application that receives data.
* Function: Subscribes to one or more topics to get updates when a new message is published.
* Example: A mobile app subscribed to greenhouse/temp to display temperature.



Figure 6.2: MQTT dashboard

* **Broker**
* Role: The central server that handles all messages.
* Function: Receives messages from publishers and forwards them to the appropriate subscribers.
* Example: Mosquitto, HiveMQ, or EMQX MQTT brokers
* Public MQTT Broker
* **Broker Name**: HiveMQ
* **Address:** broker.hivemq.com
* **Port:** 1883
* **Secure Port (TLS):** 8883

### CHAPTER 7

# INTERFACING

## Dht11 With Arduino

|  |  |
| --- | --- |
| DHT11 | Arduino |
| VCC | 5V |
| GND | GND |
| DATA | Digital pin 2 |

Table 1: DHT22 with Arduino

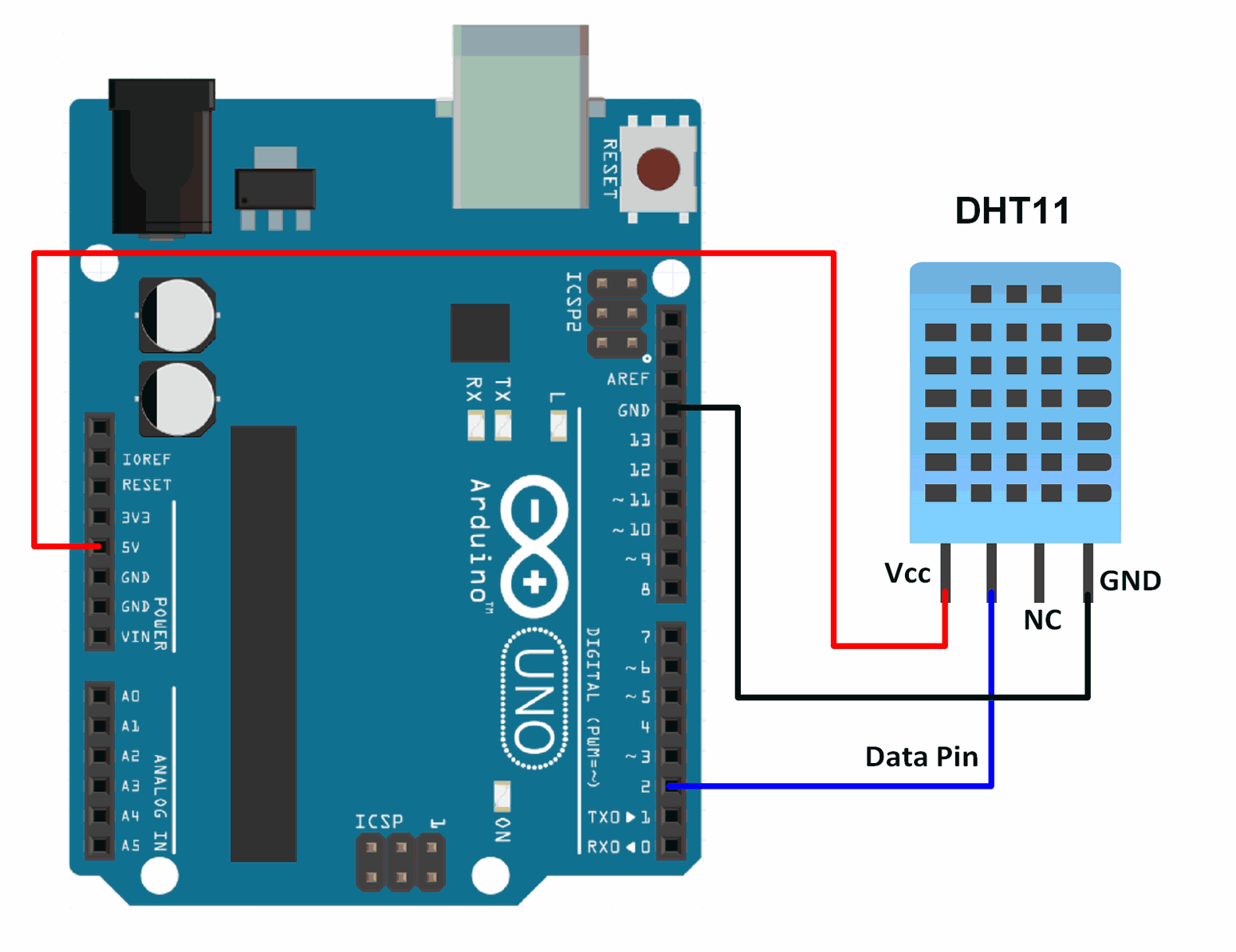


Figure 7.1: DHT22 with Arduino

## Soil Moisture With Arduino

|  |  |
| --- | --- |
| Soil sensor | Arduino Uno |
| VCC | 5V |
| GND | GND |
| A0 | A0 |

Table 2: Soil Moisture with Arduino

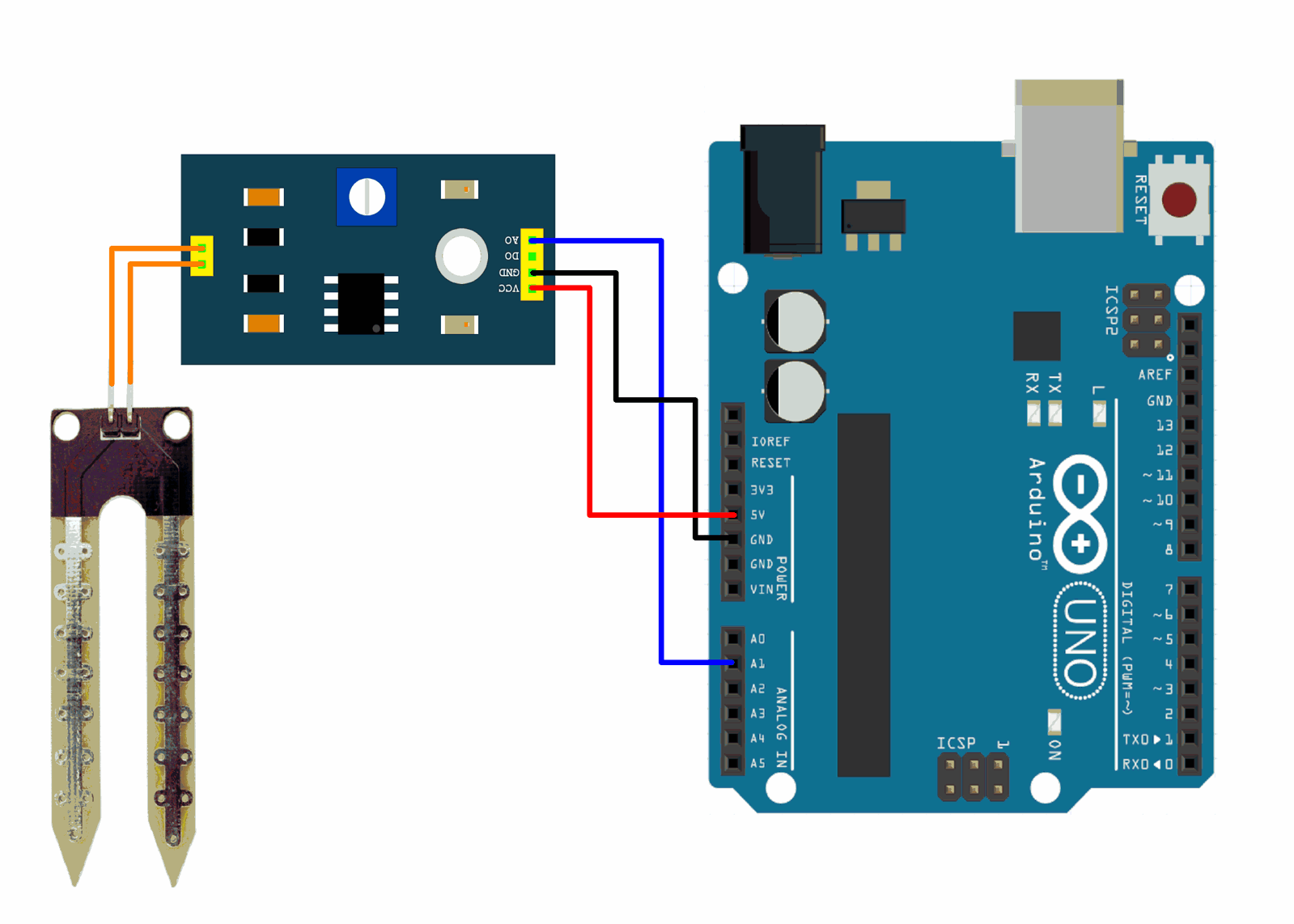


Figure 7.2: Soil Moisture with Arduino

## Relay Pump With Arduino

|  |  |  |  |
| --- | --- | --- | --- |
| Relay | Motor Pump | Arduino | Battery |
| IN |  | D7 |  |
| VCC |  | 5V |  |
| GND | GND | GND |  |
| COM |  |  |  |
| NO |  |  | -ve |
| NC |  |  |  |
|  | +ve |  | VCC |

Table 3: Relay pump with Arduino

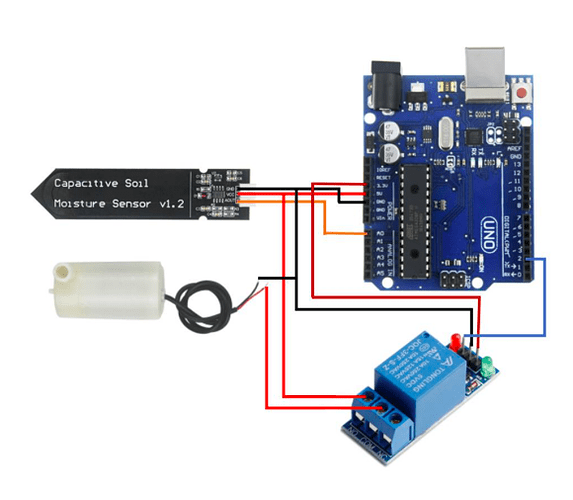


Figure 7.3: Relay Pump with Arduino

### CHAPTER 8

# WORKING PRINCIPLE

The IoT-based greenhouse monitoring system works by using sensors to continuously measure environmental parameters such as temperature, humidity, and soil moisture. These sensor readings are collected by a microcontroller, like an ESP32 or Arduino. The data is then transmitted via Wi-Fi to an IoT platform such as Blynk or MQTT, allowing users to monitor the greenhouse conditions in real-time on their mobile phones. If any sensor reading crosses a set threshold, the system automatically activates connected devices like water pumps or fans using a relay module. Additionally, alerts are sent to the user to notify them of abnormal conditions, ensuring that the greenhouse environment remains optimal for plant growth.

System implementation:

For Example, if we take a Tomato crop, the essential parameters are:

1. The user shall set the temperature suitable for a tomato at 27°C
2. The user shall set the value of Humidity suitable for a tomato at 60%
3. The user shall fix the Light intensity at 80%

With respect to the temperature, humidity and light intensity, when the temperature passes the set optimum threshold value, the relay will perform the required action which is to either bring the temperature down when it’s too high (27°C) by turning on the cooler and turning the heater on when the temperature is too low. Similarly, when the set value for humidity gets higher than 60%, the heater gets turned on. Also, with the light intensity fixed at 80%, if the value happens to get higher, the light is turned off which results in a reduction of light intensity in order to avoid the production of bad crops.

### CHAPTER 9

# OUTCOME

* Real-time monitoring of greenhouse conditions.
* Automated control mechanisms for optimal plant growth.
* Reduced resource consumption and labor costs.
* Remote access to greenhouse data and controls.

### CHAPTER 10

# RESULT AND DISCUSSION

The sensing part of all the greenhouse parameters is being measured by the use of appropriate sensors. The sensed data is being stored in the database by the use of Blynk or MQTT software and also a comparative analysis is done on the parameters based on the sensor data. Mobile alerts are also sent to the concerned people from time to time so that they keep updated on the real time greenhouse environment. The name Greenhouse Monitoring and Control System using IoT is to both sense and monitor the environmental parameters from the sensor and also to stabilize the conditions if the conditions exceed the threshold. Actuators are used to control the parameters based on the sensor input. It can be done in both automatic and manual mode. In the manual mode, the actuators are controlled by the user based on the inputs obtained through SMS which is not implemented in our project. In the automatic mode, based on the database of the previous event the actuators are being controlled.

### CHAPTER 11

# CONCLUSION AND FUTURE ENHANCEMENT

**Conclusion**

The project demonstrated an efficient, remotely accessible greenhouse system using IoT and MQTT. It offers a scalable solution for smart farming, helping monitor and control conditions to improve crop yield and reduce manual work. Future improvements can include automation, AI integration, and solar power support.

**Future enhancement**

* Integrate AI/ML for predictive control of temperature, humidity, and irrigation.
* Add camera-based plant health monitoring.
* Use solar panels for energy efficiency.
* Implement auto-irrigation based on soil moisture thresholds.
* Expand to support multiple greenhouses from a single dashboard.

# **REFERENCES**

|  |  |
| --- | --- |
| [1] | Patil &kale, "IOT in Agriculture," 2016. |
| [2] | Ali et. al, "Automation in Greenhouse," 2017. |
| [3] | Kmar et. al, "Environmental Monitoring," 2018. |
| [4] | Nairet. al, "Sustainability in Greenhouse Management," 2018. |
| [5] | Sharma et. al, "IOT platforms for data management," 2019. |

# APPENDIX

## BUDGET STRUCTURE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.N.** | **Name of Components** | **Price** | **Quantity** | **Total Cost (R.S)** |
| 1 | Arduino Uno | 2000 | 1 | 2000/- |
| 2 | ESP-32 microcontroller | 3000 | 1 | 3000/- |
| 3 | DHT22 Sensor | 500 | 1 | 500/- |
| 4 | Soil Moisture Sensor | 500 | 1 | 500/- |
| 5 | Light Sensor | 300 | 1 | 300/- |
| 6 | Relay Module | 400 | 1 | 400/- |
| 7 | DC fan | 600 | 1 | 600/- |
| 8 | Water pump | 500 | 1 | 500/- |
| 9 | Level Shifter | 300 | 1 | 300/- |
| 10 | PCB | 200 | 1 | 200/- |
| 11 | Wire and Connectors | 10 | 20 | 200/- |
| 12 | 18650 Li Battery | 700 | 3 | 2100/- |
| 13 | Cell Holder | 200 | 1 | 400/- |
| 14 | The outer body of the House | 5000 | 1 | 5000/- |
| **Total (R.S)** | | | | **= 16000/-** |

Table 4: Budget Structure

## WORKING SCHEDULE

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.N.** | **Tasks** | **Months** | | | | | | |
| **December** | **January** | **February** | **March** | **April** | **May** | **June** |
| 1 | Study on research papers &  Proposal Writing |  |  |  |  |  |  |  |
| 2 | Dataset & Component collection |  |  |  |  |  |  |  |
| 3 | Demonstration and Review |  |  |  |  |  |  |  |
| 4 | Circuit Design & System Implementation |  |  |  |  |  |  |  |
| 5 | Pilot Testing & Debugging |  |  |  |  |  |  |  |
| 6 | System Development(iot) |  |  |  |  |  |  |  |
| 7 | Documentation |  |  |  |  |  |  |  |

Table 5: Working Schedule

## CODE OF ARDUINO

#include <DHT.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// LCD at I2C address 0x27 with 16x2 display

LiquidCrystal\_I2C lcd(0x27, 16, 2);

#define DHTPIN 2

#define DHTTYPE DHT11

#define SOIL\_PIN A0

#define TRIG\_PIN 9

#define ECHO\_PIN 10

#define RELAY\_PIN 7

DHT dht(DHTPIN, DHTTYPE);

// Auto-watering variables

bool autoMode = true;

int dryThreshold = 500; // adjust based on sensor

void setup() {

  Serial.begin(9600); // For UART to ESP32

  dht.begin();

  lcd.begin(16, 2);

  lcd.backlight();

  lcd.print("Greenhouse Ready");

  pinMode(SOIL\_PIN, INPUT);

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, LOW);

}

void loop() {

  // --- Sensor Readings ---

  int soilValue = analogRead(SOIL\_PIN);

  float temp = dht.readTemperature();

  float hum = dht.readHumidity();

  int waterLevel = readUltrasonic();

  // --- Auto-Watering Logic ---

  if (autoMode && soilValue > dryThreshold) {

    digitalWrite(RELAY\_PIN, HIGH); // Pump ON

    sendAlert("SOIL\_DRY");

  } else if (autoMode) {

    digitalWrite(RELAY\_PIN, LOW); // Pump OFF

  }

  // --- Display to LCD ---

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("T:");

  lcd.print(temp);

  lcd.print(" H:");

  lcd.print(hum);

  lcd.setCursor(0, 1);

  lcd.print("S:");

  lcd.print(soilValue);

  lcd.print(" W:");

  lcd.print(waterLevel);

  // --- Send Data to ESP32 via UART ---

  Serial.print("SOIL:");

  Serial.print(soilValue);

  Serial.print(",TEMP:");

  Serial.print(temp);

  Serial.print(",HUM:");

  Serial.print(hum);

  Serial.print(",WATER:");

  Serial.println(waterLevel);

  // --- Receive Command from ESP32 ---

  if (Serial.available()) {

    String cmd = Serial.readStringUntil('\n');

    cmd.trim();

    handleCommand(cmd);

  }

  delay(3000); // Wait 3 seconds

}

// --- Read distance from ultrasonic ---

int readUltrasonic() {

  digitalWrite(TRIG\_PIN, LOW);

  delayMicroseconds(2);

  digitalWrite(TRIG\_PIN, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG\_PIN, LOW);

  long duration = pulseIn(ECHO\_PIN, HIGH);

  int cm = duration \* 0.034 / 2;

  return cm;

}

// --- Handle incoming commands from ESP32 ---

void handleCommand(String cmd) {

  if (cmd == "PUMP\_ON") {

    autoMode = false;

    digitalWrite(RELAY\_PIN, HIGH);

  } else if (cmd == "PUMP\_OFF") {

    autoMode = false;

    digitalWrite(RELAY\_PIN, LOW);

  } else if (cmd == "AUTO\_ON") {

    autoMode = true;

  } else if (cmd == "AUTO\_OFF") {

    autoMode = false;

    digitalWrite(RELAY\_PIN, LOW);

  }

}

// --- Send alert to ESP32 ---

void sendAlert(String msg) {

  Serial.print("ALERT:");

  Serial.println(msg);

}

## ESP-32 CODE

#include <WiFi.h>

#include <PubSubClient.h>

#define RXD2 16  // ESP32 RX to Arduino TX (D1)

#define TXD2 17  // ESP32 TX to Arduino RX (D0)

// WiFi credentials

const char\* ssid = "Name";

const char\* password = "Password";

// MQTT broker

const char\* mqtt\_server = "broker.hivemq.com";

const int mqtt\_port = 1883;

const char\* pubTopic = "greenhouse/data";

const char\* cmdTopic = "greenhouse/cmd";

const char\* alertTopic = "greenhouse/alert";

WiFiClient espClient;

PubSubClient client(espClient);

String incomingUART = "";

void setup() {

  Serial.begin(115200);

  Serial2.begin(9600, SERIAL\_8N1, RXD2, TXD2);  // UART communication

  WiFi.begin(ssid, password);

  Serial.print("Connecting to WiFi");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500); Serial.print(".");

  }

  Serial.println("Connected to WiFi");

  client.setServer(mqtt\_server, mqtt\_port);

  client.setCallback(callback);

  connectToMQTT();

}

void loop() {

  if (!client.connected()) {

    connectToMQTT();

  }

  client.loop();

  // Read from Arduino via UART

  while (Serial2.available()) {

    char c = Serial2.read();

    if (c == '\n') {

      processUART(incomingUART);

      incomingUART = "";

    } else {

      incomingUART += c;

    }

  }

}

// Process incoming UART message from Arduino

void processUART(String data) {

  data.trim();

  if (data.startsWith("ALERT:")) {

    client.publish(alertTopic, data.substring(6).c\_str());

  } else {

    client.publish(pubTopic, data.c\_str());

  }

}

// MQTT callback: handle incoming command

void callback(char\* topic, byte\* payload, unsigned int length) {

  String cmd = "";

  for (int i = 0; i < length; i++) {

    cmd += (char)payload[i];

  }

  cmd.trim();

  // Forward command to Arduino

  Serial2.println(cmd);  // Send to Arduino

  Serial.println("Sent to Arduino: " + cmd);

}

// MQTT connect function

void connectToMQTT() {

  while (!client.connected()) {

    Serial.print("Connecting to MQTT...");

    if (client.connect("ESP32Client")) {

      Serial.println("connected");

      client.subscribe(cmdTopic);

    } else {

      Serial.print("failed, rc=");

      Serial.print(client.state());

      Serial.println(" trying again in 5s");

      delay(5000);

    }

  }

}