NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

MICROPROCESSORS AND MICROCONTROLLERS

COECC404

Project Report

“SMART IRRIGATION SYSTEM”

Aditya Singla (2023UCS1669),

Himanshu Kumar Bairwa (2023UCS1708),

Peeyush Gupta (2023UCS1717)

SEMESTER-4

CSE SECTION-3

INDEX

ACKNOWLEDGEMENT

We express our sincere gratitude to all those who have contributed to the completion of this project. It has been a journey marked by collaborative efforts, and we are deeply appreciative for the support and assistance we have received.

First and foremost, we extend our heartfelt thanks to our mentors, Prof. Dhananjay V. Gadre and Dr. Konark Sharma, whose unwavering guidance and supervision have been instrumental throughout this endeavour. Their profound insights, constant support and pro-vision of essential information have been invaluable.

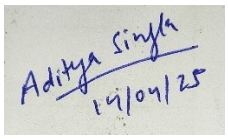
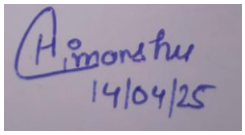
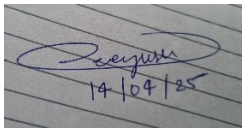
We are indebted to Prof. Dhananjay V. Gadre for his profound mentorship, which has illuminated our path and enriched our understanding of the project intricacies. His willingness to share his extensive knowledge has been a beacon of inspiration, enabling us to navigate through the complexities of the project with confidence.

We extend our heartfelt gratitude to Dr. Konark Sharma for his valuable contributions and steadfast support. His dedication to our growth and his willingness to impact his expertise have been pivotal in shaping our journey and ensuring the successful completion of our tasks.

Furthermore, we also express our deepest gratitude to our parents for their unwavering love, encouragement, and sacrifices. Their support has been the bedrock of our journey and we are profoundly thankful for their guidance and understanding.

CERTIFICATE

This is to certify that **Aditya Singla** (2023UCS1669), **Himanshu Kumar Bairwa** (2023UCS1708) and **Peeyush Gupta** (2023UCS1717) have successfully completed their 'MICROPROCESSOR & MICROCONTROLLER' project 'SMART IRRIGATION SYSTEM' under the guidance of **Prof. Dhananjay V. Gadre** sir and **Dr. Konark Sharma** sir.

S no.	Name	Roll No.	Photo	Sign
1.	Aditya Singla	2023UCS1669		
2.	Himanshu Kumar Bairwa	2023UCS1708		
3.	Peeyush Gupta	2023UCS1717		

OBJECTIVE

The primary objective of this project is to design and implement an efficient, automated, and resource-conserving smart irrigation system using the ESP32 microcontroller. This system will leverage real-time environmental data, such as soil moisture, temperature, and humidity, to optimize water usage for agricultural applications. The system will integrate sensors (soil moisture, temperature, and humidity sensors) with the ESP32 to monitor soil conditions and trigger irrigation only when necessary, reducing water wastage and ensuring plants receive optimal hydration. The system will be remotely accessible through a mobile application or a web interface, enabling users to control and monitor irrigation schedules, enhancing convenience, and improving water management.

Key features could include:

- Real-time monitoring of soil conditions.
- Automated irrigation control based on predefined moisture thresholds.
- Remote control and monitoring via a mobile app or web interface.
- Energy-efficient operation with scheduled irrigation.
- Alerts and notifications for system status, low water levels, or failures.

This project would aim to create an eco-friendly and cost-efficient solution for modern irrigation needs, making use of IoT and automation technologies to contribute to sustainable agriculture.

INTRODUCTION

The Smart Irrigation System is designed to automate the watering process in agricultural fields, gardens, or greenhouses by utilizing real-time sensor data and intelligent control mechanisms. At the core of the system lies the ESP32 microcontroller, which acts as the central processing unit, collecting and processing data from various sensors and controlling the operation of the irrigation components.

The system uses a soil moisture sensor to continuously monitor the moisture content of the soil. When the detected moisture level falls below a predefined threshold, the system automatically activates a water pump to irrigate the soil. Once the soil regains adequate moisture, the pump is turned off, ensuring water is only used, when necessary, thereby reducing waste and conserving this valuable resource.

To enhance functionality and environmental awareness, the system also integrates weather data (either from onboard sensors or external APIs) to make informed decisions about watering, such as skipping irrigation on rainy days. A PIR (Passive Infrared) motion sensor is included to detect the presence of humans or animals near the system, which can be used to trigger alerts, increase security, or temporarily pause irrigation to prevent unintended soaking.

All sensor readings—including soil moisture levels, ambient temperature, humidity, and motion detection status—are displayed in real time on a computer interface, providing users with continuous insight into the system's operation and environmental conditions. This interface can be developed using serial communication or over a Wi-Fi network, enabling remote monitoring and control.

By automating irrigation based on real-time environmental feedback and user-defined thresholds, this system not only increases the efficiency of water usage but also reduces the manual effort required for field maintenance. It represents a significant step toward the integration of IoT technologies in agriculture, promoting smart farming practices that are scalable, sustainable, and easy to deploy.

BASIC IDEA

The Smart Irrigation System is an IoT-based project that uses an ESP32 microcontroller to automate the process of watering plants in gardens or agricultural fields. The system uses a soil moisture sensor to measure the moisture level of the soil. If the moisture level drops below a set value, the ESP32 activates a water pump to supply water to the plants. Once the soil reaches the required moisture level, the pump turns off automatically.

To make the system smarter and more efficient, additional features are included:

- A PIR motion sensor to detect movement near the system (useful for security or safety).
- Weather monitoring (via temperature and humidity sensors or online data) to avoid unnecessary watering when it might rain.
- A real-time display of sensor data on a computer screen using serial communication or a Wi-Fi-based dashboard.

This project helps in saving water, reducing manual effort, and making agriculture more automated and sustainable by using simple and low-cost electronic components.

COMPONENTS

---

1. **ESP32 – Wroom Dev Kit**
Acts as the brain of the system. It reads sensor data, makes decisions based on moisture levels, controls the water pump via the relay, and can send data to a computer or cloud for monitoring.

2. **Soil Moisture Sensor**
Measures the moisture content in the soil. If the soil is too dry, it signals the ESP32 to turn on the water pump.

3. **Relay Module**
Works as a switch to safely control high-power devices like the water pump using low-power signals from the ESP32.

4. **Water Pump**
Responsible for watering the plants. It is activated by the relay when soil moisture is low and turns off when the soil is sufficiently wet.

5. **Temperature and Humidity Sensor (DHT11/DHT22)**
Collects environmental data to monitor weather conditions. This data can be used to make smarter irrigation decisions, like skipping watering on humid or rainy days.

6. **PIR Sensor**
Detects motion around the system. It can be used to trigger alerts, pause irrigation if a person or animal is nearby, or log activity for security.

7. **Power Supply (12V adapter or batteries)**
Provides electrical power to the ESP32 and other components. The choice of power source depends on whether the system is portable or stationary.

8. **Jumper Wires (3 sets)**
Used to connect all the components on the breadboard to the ESP32 and other modules for signal and power transfer.

9. **Breadboard**
A prototyping platform to arrange and connect components without soldering, making it easy to test and modify the circuit.

THEORY

The Smart Irrigation System is a modern agricultural innovation that integrates Internet of Things (IoT) technologies to optimize water usage and reduce manual intervention in the irrigation process. It combines environmental sensing, automation, and remote monitoring to ensure plants receive the right amount of water at the right time.

At the core of this system lies the ESP32 microcontroller, a low-cost, low-power system-on-chip (SoC) with integrated Wi-Fi and Bluetooth capabilities. The ESP32 serves as the main processing unit, collecting data from various sensors and executing control operations accordingly.

1. Working Principle

The system continuously monitors the soil moisture level using a soil moisture sensor. When the moisture level falls below a predefined threshold (indicating dry soil), the ESP32 activates a water pump through a relay module. The pump irrigates the soil until the moisture level reaches the desired threshold, at which point the ESP32 turns the pump off, thereby automating the irrigation cycle.

2. Environmental Monitoring

A DHT11/DHT22 sensor is used to monitor temperature and humidity. These values can help in:

- Understanding environmental conditions.
- Making future enhancements (e.g., adjusting watering based on weather conditions).

The system may also integrate external weather APIs to further optimize irrigation schedules.

3. Motion Detection

A PIR (Passive Infrared) sensor is added to detect movement near the system. This feature can be used to:

- Trigger security alerts.

- Temporarily pause irrigation to avoid spraying water on animals or humans.

4. Data Display and Monitoring

The system provides real-time monitoring of sensor readings via a computer interface using serial communication or Wi-Fi. This helps users:

- View live data (e.g., moisture level, temperature, humidity).
- Get notified of any issues or sensor thresholds.

5. Power Supply and Circuit Design

The ESP32 and other components are powered using a 12V adapter or battery. A breadboard is used for easy prototyping, and jumper wires connect all components to form the circuit. The use of a relay ensures that high-voltage devices (like pumps) are isolated from the low-voltage control circuit of the ESP32.

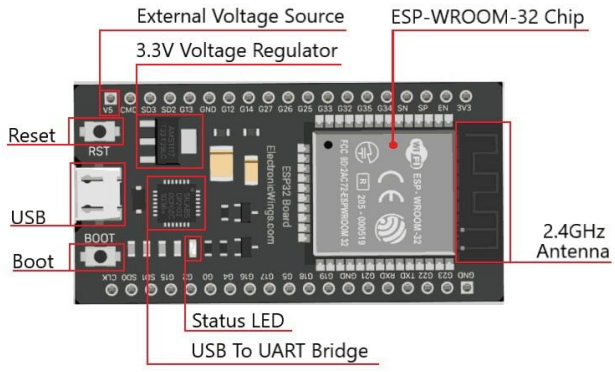


Fig1. Esp32 -Wroom Development Board

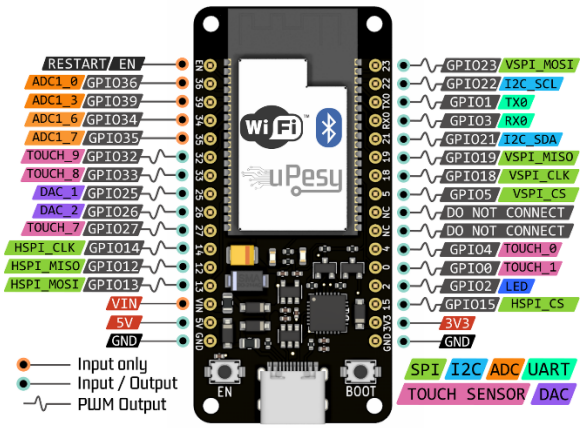


Fig2: Pin Diagram of ESP32

a. V5 (5V Pin): Provides 5V power supply.

b. GND (Ground Pin): Provides the ground reference for the circuit.

c. 3V3 (3.3V Pin): Provides 3.3V power supply.

d. GPIO (General Purpose Input/Output Pins):

- ESP32 has 34 GPIO pins (depending on the variant, not all are available).
- Supports functions like PWM, I2C, SPI, UART, ADC, DAC, and more.

e. ADC Pins (Analog-to-Digital Converter):

- 12-bit ADC resolution.
- Up to 18 channels for analog input (depending on the variant).

f. DAC Pins (Digital-to-Analog Converter):

- 2 DAC pins for analog output.

g. SPI Pins (Serial Peripheral Interface):

- Includes pins for MISO, MOSI, SCK, and CS.

h. I2C Pins (Inter-Integrated Circuit):

- SDA (data) and SCL (clock) for I2C communication.

i. UART Pins (Universal Asynchronous Receiver-Transmitter):

- TXD (Transmit) and RXD (Receive) for serial communication.

j. Touch Pins: Capacitive touch support on several GPIOs.

k. SD Card Pins: Dedicated pins for interfacing with SD cards.

l. External Interrupt Pins: Supports external interrupts on several GPIO pins.

m. Wi-Fi and Bluetooth Antenna Pins: Separate pins for connecting external Wi-Fi/Bluetooth antennas

PROCEDURE

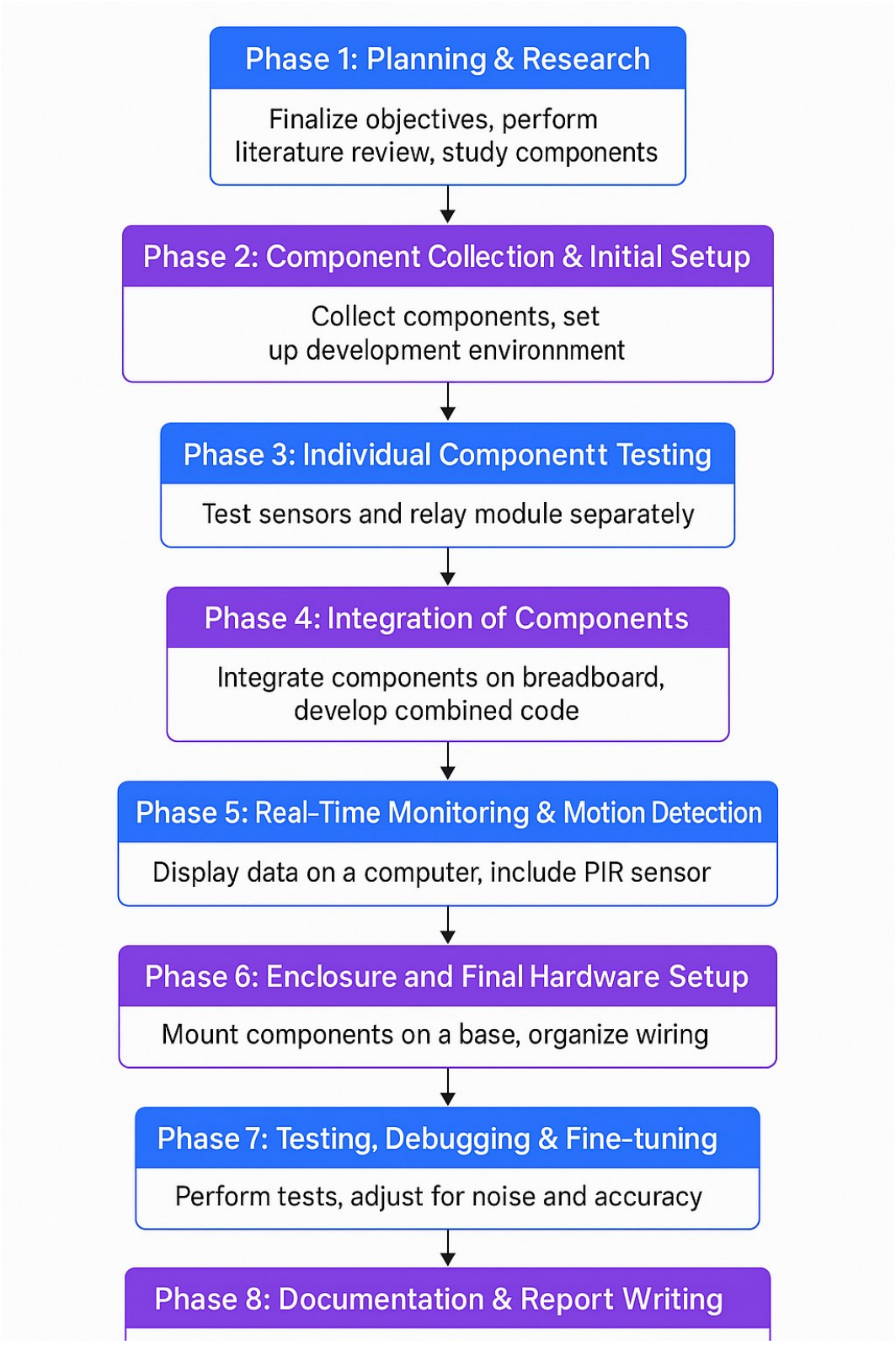


Fig3. Work Flow Diagram

Phase 1: Planning & Research

Goals:

- Finalize the project idea and title.
- Understand the objective, scope, and requirements.
- Perform literature review on existing smart irrigation systems.
- Study ESP32, sensors, and components.

Outputs:

- Project synopsis.
- Component list and cost estimation.
- Block diagram of the proposed system.

Phase 2: Component Collection & Initial Setup

Goals:

- Purchase or collect all required components:
    - ESP32, sensors, relay, pump, breadboard, etc.
- Set up Arduino IDE or Platform IO for ESP32 programming.
- Install necessary libraries and drivers.

Outputs:

- Functional development environment.
- All hardware components ready for use.

Phase 3: Individual Component Testing

Goals:

- Test each sensor separately using ESP32:
    - Soil moisture sensor.

    - DHT11/DHT22.
    - PIR sensor.
    - Relay with pump.
- Confirm proper working through serial monitor.

Outputs:

- Working codes for each module.
- Understanding of sensor data and threshold values.

Phase 4: Integration of Components

Goals:

- Integrate all components on a breadboard.
- Write combined code to:
    - Read sensors.
    - Control pump using relay.
    - Display values on serial monitor.
- Add threshold-based decision logic for pump activation.

Outputs:

- Fully working prototype on breadboard.
- Code for integrated system.

Phase 5: Real-Time Monitoring & Motion Detection

Goals:

- Integrate PIR sensor for motion detection.
- Display real-time values on a computer screen via serial/Wi-Fi.
- Optional: Design a simple dashboard or data logger.

Outputs:

- Enhanced monitoring with motion alerts.
- Stable real-time data display.

Phase 6: Enclosure and Final Hardware Setup

Goals:

- Mount components on a cardboard or acrylic base.
- Neatly organize the wires and sensors.
- Use a water container to simulate real irrigation.

Outputs:

- Physical working model ready for demonstration.

Phase 7: Testing, Debugging & Fine-tuning

Goals:

- Perform field-like tests.
- Adjust moisture threshold and pump runtime.
- Debug sensor noise or unstable readings.

Outputs:

- Stable and reliable system.
- Improved accuracy and responsiveness.

Phase 8: Documentation & Report Writing

Goals:

- Prepare the following:
    - Abstract

    - Introduction
    - Theory
    - Circuit diagrams
    - Code explanation
    - Results and conclusion
- Add photos of the working model.

Outputs:

- Complete project report and presentation slides.

Phase 9: Final Presentation & Demonstration

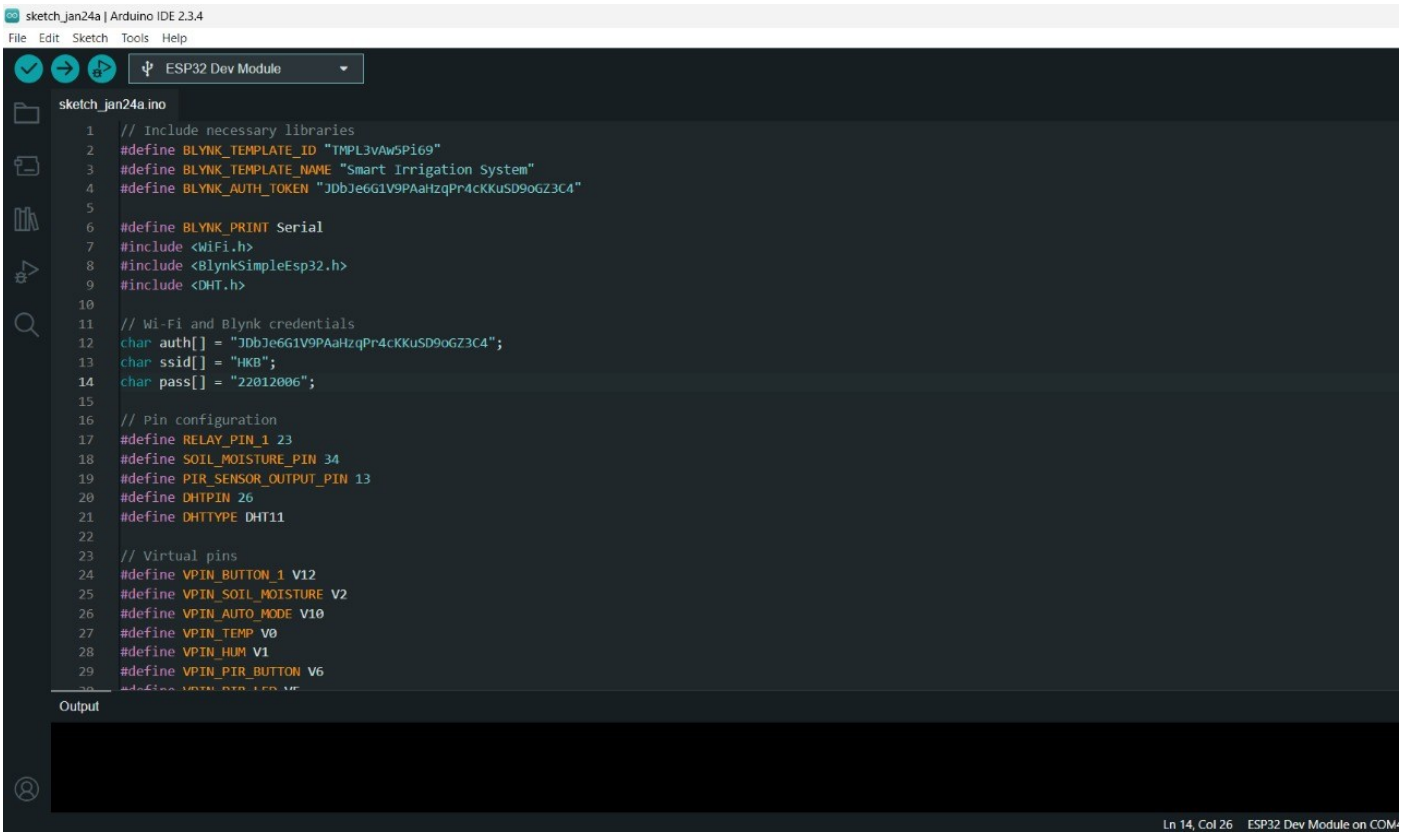
Goals:

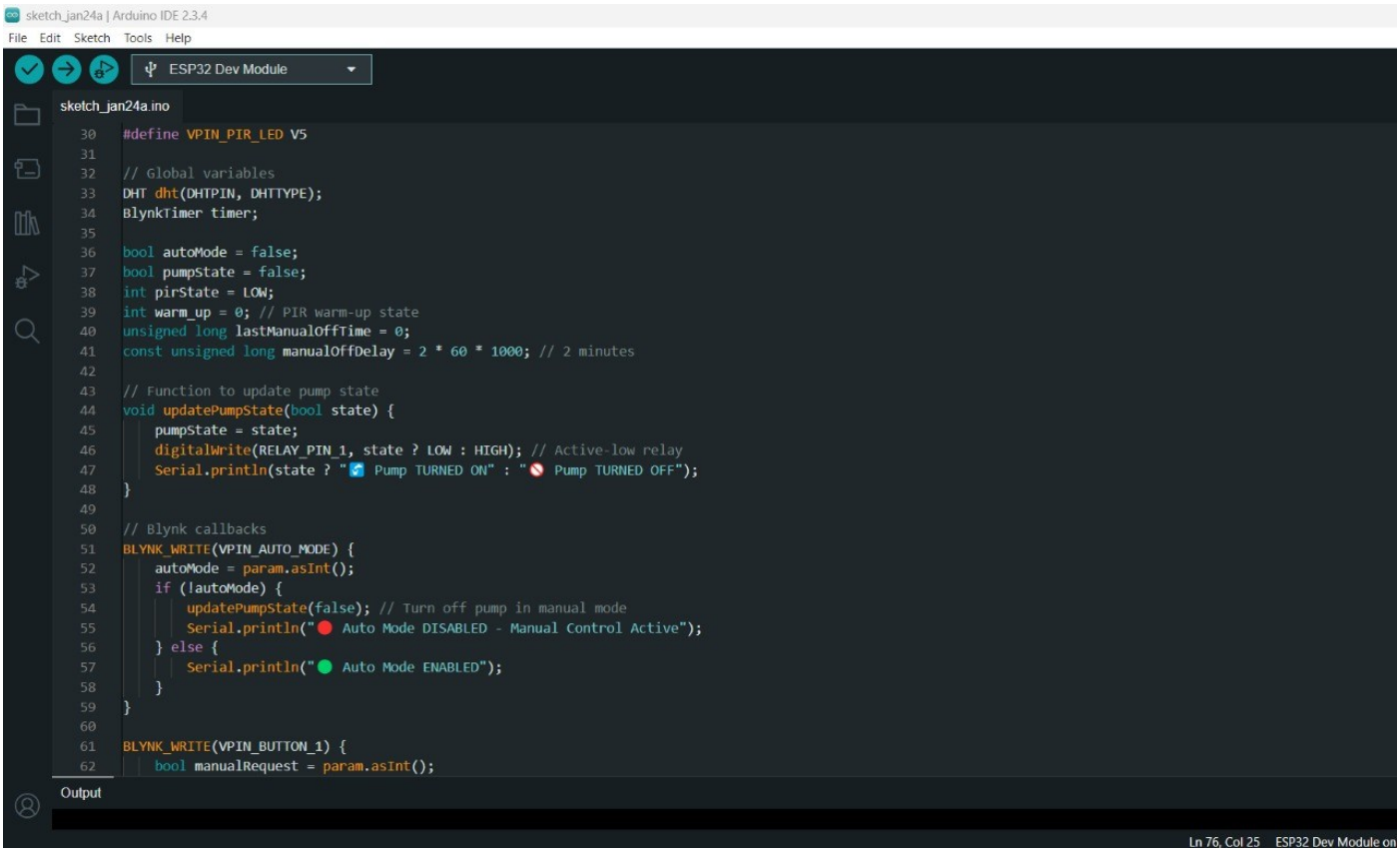
- Rehearse the demonstration.
- Explain the working, application, and advantages.
- Present the report, code, and working model.

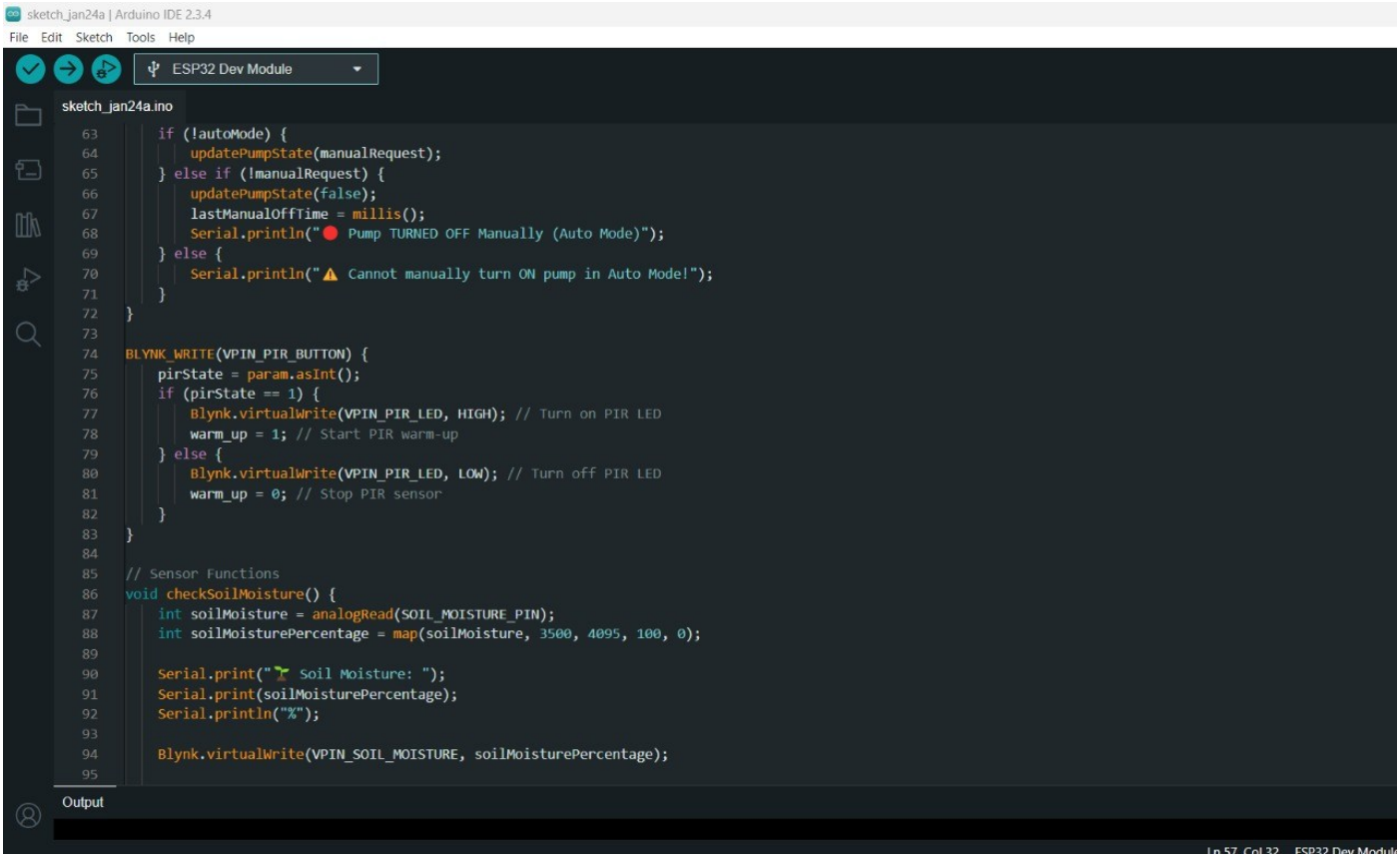
Outputs:

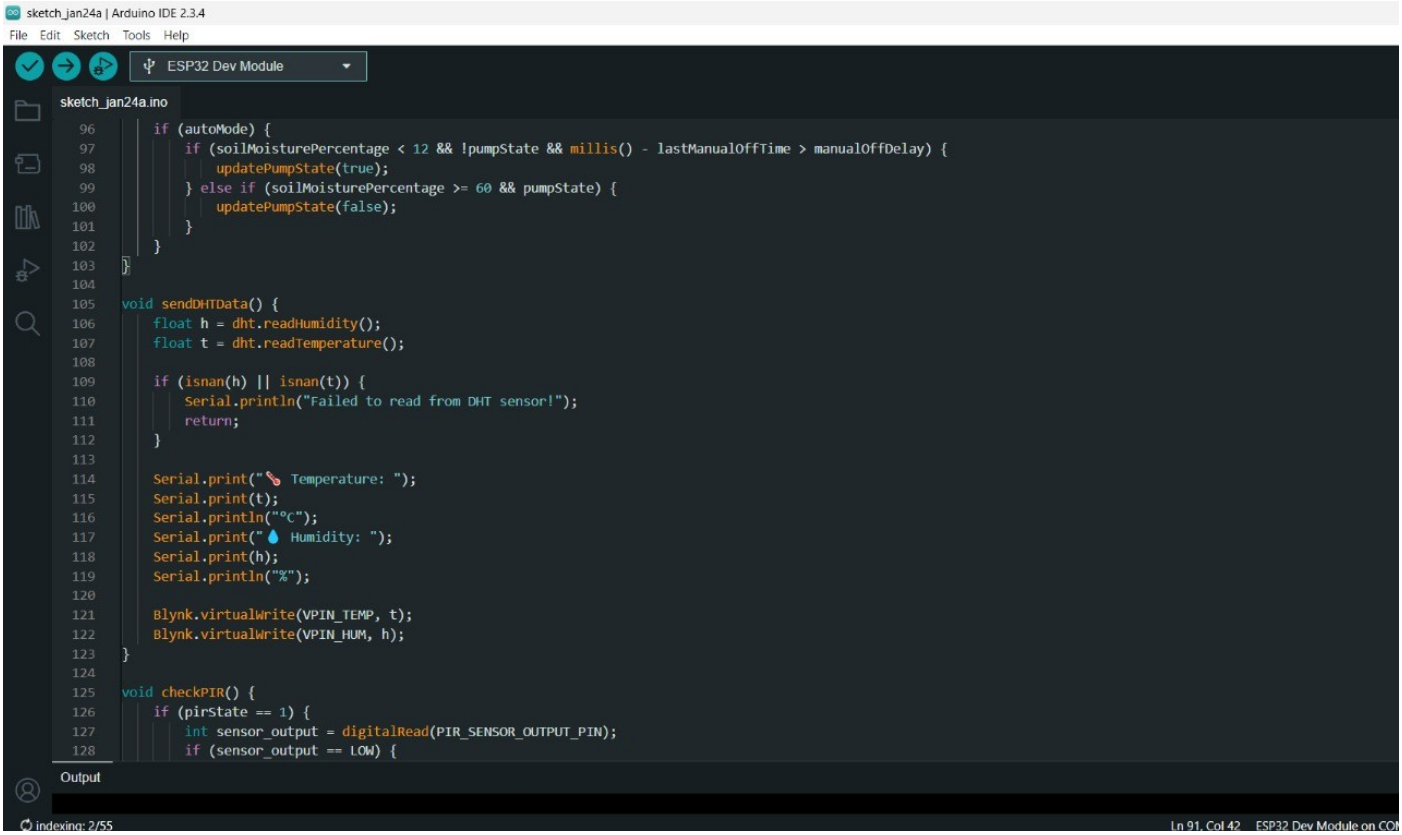
- Successful final evaluation.
- Project submission completed.

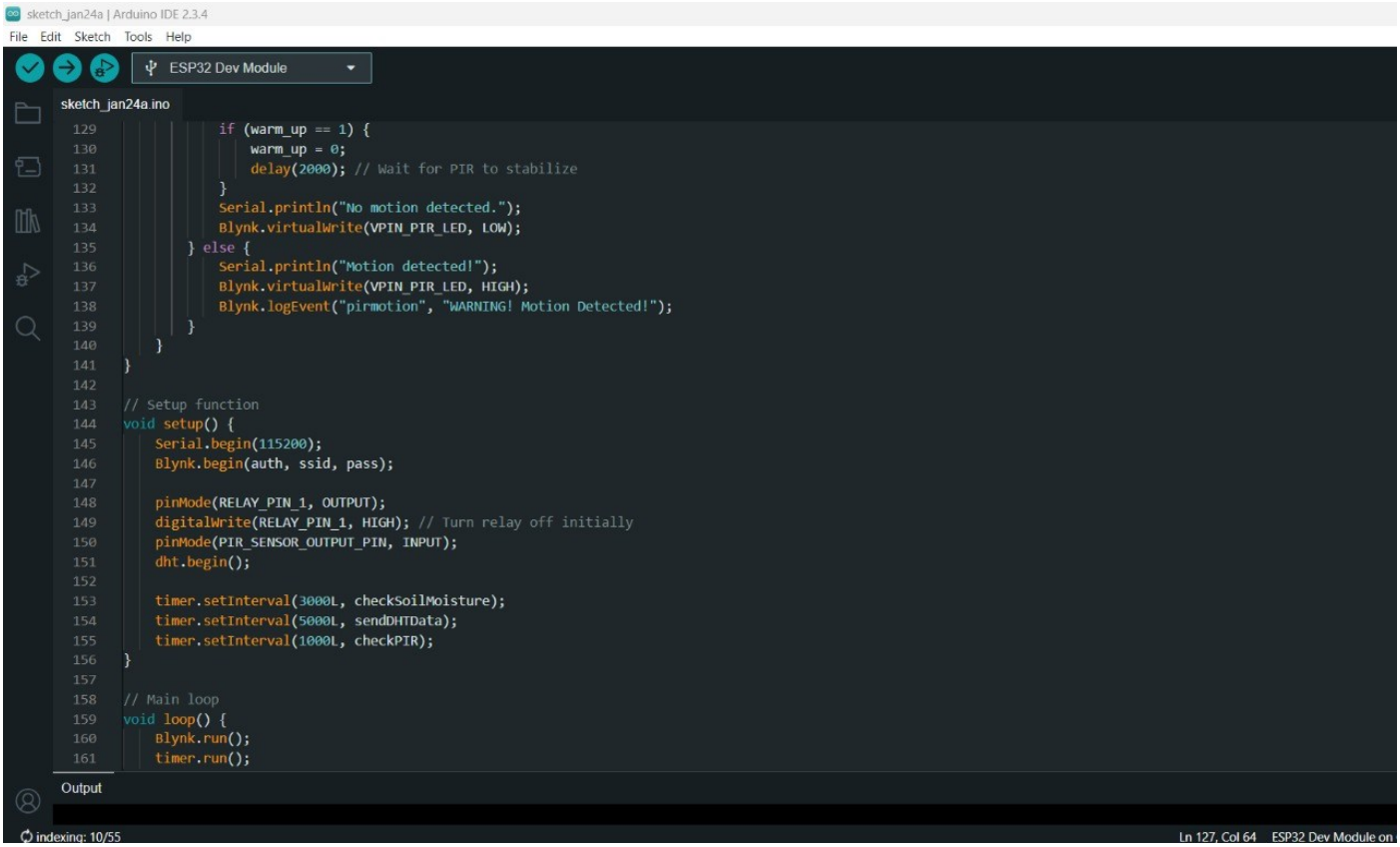
CODING


sketch_jan24a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
ESP32 Dev Module
sketch_jan24a.ino
// Include necessary libraries
#define BLYNK_TEMPLATE_ID "TMPL3vAw5Pi69"
#define BLYNK_TEMPLATE_NAME "Smart Irrigation System"
#define BLYNK_AUTH_TOKEN "JDbJe6G1V9PAaHzqPr4cKKuSD9oGZ3C4"
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>
// Wi-Fi and Blynk credentials
char auth[] = "JDbJe6G1V9PAaHzqPr4cKKuSD9oGZ3C4";
char ssid[] = "HKB";
char pass[] = "22012006";
// Pin configuration
#define RELAY_PIN_1 23
#define SOIL_MOISTURE_PIN 34
#define PIR_SENSOR_OUTPUT_PIN 13
#define DHTPIN 26
#define DHTTYPE DHT11
// Virtual pins
#define VPIN_BUTTON_1 V12
#define VPIN_SOIL_MOISTURE V2
#define VPIN_AUTO_MODE V10
#define VPIN_TEMP V0
#define VPIN_HUM V1
#define VPIN_PIR_BUTTON V6
Output
Ln 14, Col 26 ESP32 Dev Module on COM4


sketch_jan24a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
ESP32 Dev Module
sketch_jan24a.ino
#define VPIN_PIR_LED V5
// Global variables
DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;
bool autoMode = false;
bool pumpState = false;
int pirState = LOW;
int warm_up = 0; // PIR warm-up state
unsigned long lastManualOffTime = 0;
const unsigned long manualOffDelay = 2 * 60 * 1000; // 2 minutes
// Function to update pump state
void updatePumpState(bool state) {
pumpState = state;
digitalWrite(RELAY_PIN_1, state ? LOW : HIGH); // Active-low relay
Serial.println(state ? "🚰 Pump TURNED ON" : "🚫 Pump TURNED OFF");
}
// Blynk callbacks
BLYNK_WRITE(VPIN_AUTO_MODE) {
autoMode = param.asInt();
if (!autoMode) {
updatePumpState(false); // Turn off pump in manual mode
Serial.println("🔴 Auto Mode DISABLED - Manual Control Active");
} else {
Serial.println("🟢 Auto Mode ENABLED");
}
}
BLYNK_WRITE(VPIN_BUTTON_1) {
bool manualRequest = param.asInt();
Output
Ln 76, Col 25 ESP32 Dev Module on


sketch_jan24a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
ESP32 Dev Module
sketch_jan24a.ino
if (!autoMode) {
updatePumpState(manualRequest);
} else if (!manualRequest) {
updatePumpState(false);
lastManualOffTime = millis();
Serial.println("🔴 Pump TURNED OFF Manually (Auto Mode)");
} else {
Serial.println("⚠️ Cannot manually turn ON pump in Auto Mode!");
}
}
BLYNK_WRITE(VPIN_PIR_BUTTON) {
pirState = param.asInt();
if (pirState == 1) {
Blynk.virtualWrite(VPIN_PIR_LED, HIGH); // Turn on PIR LED
warm_up = 1; // Start PIR warm-up
} else {
Blynk.virtualWrite(VPIN_PIR_LED, LOW); // Turn off PIR LED
warm_up = 0; // Stop PIR sensor
}
}
// Sensor Functions
void checkSoilMoisture() {
int soilMoisture = analogRead(SOIL_MOISTURE_PIN);
int soilMoisturePercentage = map(soilMoisture, 3500, 4095, 100, 0);
Serial.print("🌱 Soil Moisture: ");
Serial.print(soilMoisturePercentage);
Serial.println("%");
Blynk.virtualWrite(VPIN_SOIL_MOISTURE, soilMoisturePercentage);
Output


sketch_jan24a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
ESP32 Dev Module
sketch_jan24a.ino
if (autoMode) {
if (soilMoisturePercentage < 12 && !pumpState && millis() - lastManualOffTime > manualOffDelay) {
updatePumpState(true);
} else if (soilMoisturePercentage >= 60 && pumpState) {
updatePumpState(false);
}
}
}
void sendDHTData() {
float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
Serial.print("🌡️ Temperature: ");
Serial.print(t);
Serial.println("°C");
Serial.print("💧 Humidity: ");
Serial.print(h);
Serial.println("%");
Blynk.virtualWrite(VPIN_TEMP, t);
Blynk.virtualWrite(VPIN_HUM, h);
}
void checkPIR() {
if (pirState == 1) {
int sensor_output = digitalRead(PIR_SENSOR_OUTPUT_PIN);
if (sensor_output == LOW) {
Output
indexing: 2/55


sketch_jan24a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
ESP32 Dev Module
sketch_jan24a.ino
if (warm_up == 1) {
warm_up = 0;
delay(2000); // Wait for PIR to stabilize
}
Serial.println("No motion detected.");
Blynk.virtualWrite(VPIN_PIR_LED, LOW);
} else {
Serial.println("Motion detected!");
Blynk.virtualWrite(VPIN_PIR_LED, HIGH);
Blynk.logEvent("pirmotion", "WARNING! Motion Detected!");
}
}
}
// Setup function
void setup() {
Serial.begin(115200);
Blynk.begin(auth, ssid, pass);
pinMode(RELAY_PIN_1, OUTPUT);
digitalWrite(RELAY_PIN_1, HIGH); // Turn relay off initially
pinMode(PIR_SENSOR_OUTPUT_PIN, INPUT);
dht.begin();
timer.setInterval(3000L, checkSoilMoisture);
timer.setInterval(5000L, sendDHTData);
timer.setInterval(1000L, checkPIR);
}
// Main loop
void loop() {
Blynk.run();
timer.run();
Output
indexing: 10/55
Ln 127, Col 64
ESP32 Dev Module on

OBSERVATIONS

Implemented the Hardware Successfully:

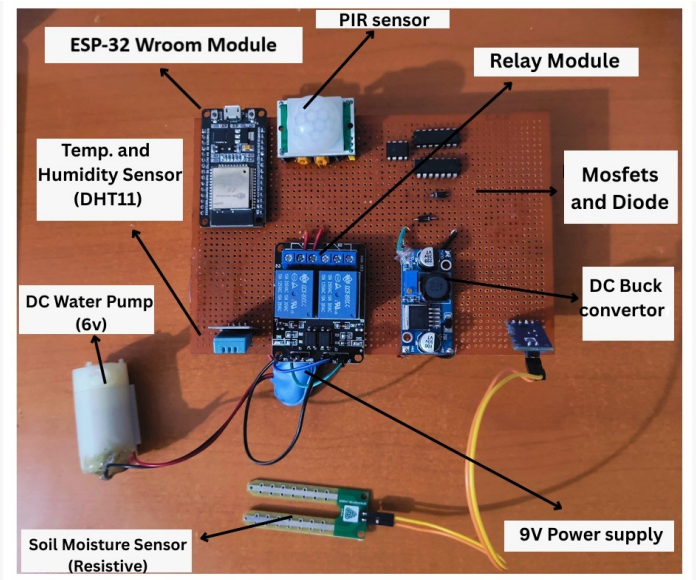
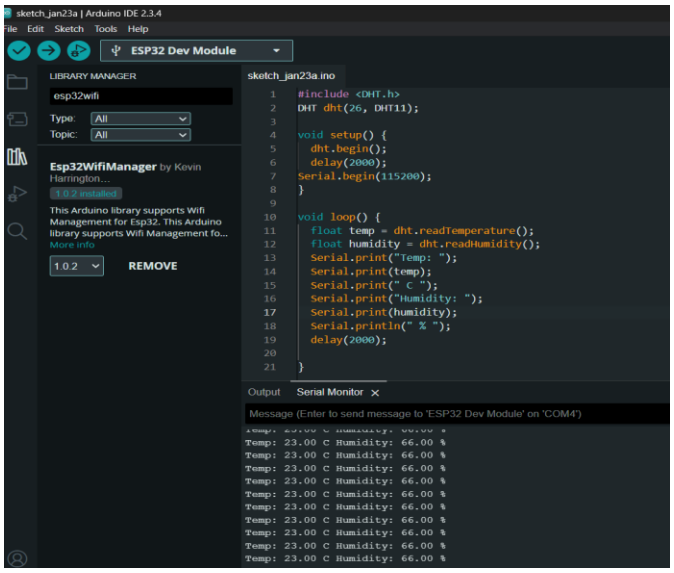
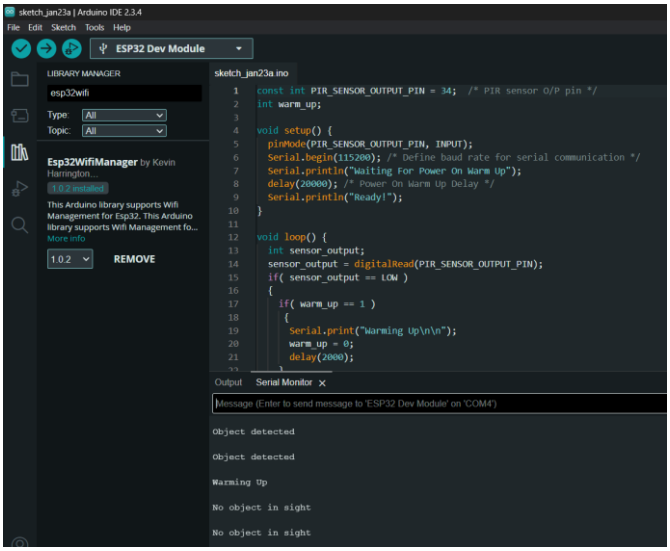


Fig4. Integrated Circuit

Online Simulation:





Interface:

Fig5. Mobile Interface

CHALLENGES FACED

---

1. Driver Issue for Connecting ESP32:
    - Problem: Initially, we faced issues connecting the ESP32 to the computer due to missing or incorrect drivers. The device was not recognized, and the COM port was not visible in the Arduino IDE.
    - Solution: We solved this by installing the correct CP210x USB to UART Bridge VCP drivers (for the ESP32) from the Silicon Labs website. After installation, the ESP32 was properly recognized, and the correct COM port appeared in the IDE, allowing us to upload the code.
2. Port Selection for Sensor and Reading:
    - Problem: At first, we had difficulty selecting the correct COM port for communication with the ESP32, which affected sensor readings and data transmission.
    - Solution: We ensured that the correct COM port was selected in the Tools menu of the Arduino IDE. Additionally, we verified that the correct board (ESP32 DevKitC) was selected, which resolved the issue of incorrect readings and allowed us to interface with the sensors successfully.
3. Correct Library Required:
    - Problem: During the setup, we faced issues with sensor integration because the libraries for the soil moisture sensor and relay module were either missing or outdated.
    - Solution: We identified and installed the correct libraries via the Arduino Library Manager. For example, we used the DHT sensor library for the DHT11 and the Adafruit Soil Moisture Sensor library for the soil moisture sensor. This allowed us to get accurate sensor readings.
4. Assembling Resistors and Other Components Correctly:
    - Problem: Initially, we did not assemble the resistors, MOSFETs, and other components correctly, which caused fluctuating or inaccurate readings from the sensors.
    - Solution: We carefully followed the circuit diagrams to ensure that resistors were correctly placed (e.g., 10kΩ pull-up resistors for the DHT11 sensor). We also ensured that the MOSFETs were connected properly for switching the relay. After re-checking the assembly, the sensor readings became precise, and the system began functioning as expected.
5. By addressing these issues systematically, we were able to troubleshoot and resolve the problems, leading to successful integration and reliable sensor readings for the smart irrigation system.

SIGNIFICANCE OF THE PROJECT

A smart irrigation system powered by ESP32 microprocessor has profound significance due to its ability to address critical challenges in agriculture and gardening while leveraging modern IoT technologies. Below are the key points highlighting its importance:

1. Water Conservation

- Traditional irrigation methods waste up to 50% of water due to inefficiencies like evaporation, runoff, and overwatering. Smart irrigation systems optimize water usage by using real-time data from sensors to apply water only, when necessary, significantly reducing wastage.
- This technology contributes to sustainable farming practices, essential in combating global water scarcity issues.

2. Cost Efficiency

- By minimizing water waste, smart irrigation systems reduce operational costs for farmers and gardeners. Automated watering schedules eliminate the need for manual intervention, saving both time and labour costs.

3. Improved Crop Health

- Precise control over soil moisture ensures that crops receive the optimal amount of water, preventing issues such as overwatering or underwatering. This leads to healthier plants and improved crop yields.
- The system also adapts to changes in weather and soil conditions, ensuring consistent growth cycles.

4. Enhanced Resource Management

- Smart irrigation systems provide a unified view of soil characteristics, including moisture levels and nutrient content, enabling better decision-making for resource allocation.

- Integration with IoT platforms allows remote monitoring and control via mobile apps or cloud-based dashboards, making farm management more efficient.

5. Environmental Sustainability

- By reducing water usage and energy consumption (e.g., through optimized pump operations), smart irrigation systems contribute to environmental sustainability.
- The technology also helps mitigate the impact of climate change by promoting responsible use of natural resources.

6. User-Friendly Automation

- These systems are easy to operate, requiring minimal training. Automated processes reduce human effort while maintaining flexibility for customization based on specific landscape needs.

7. Market Growth Potential

- The smart irrigation market is growing rapidly, with a projected value of $2.1 billion by 2025. This reflects increasing adoption by agribusinesses and farmers worldwide due to its proven benefits in efficiency and productivity.

In summary, a smart irrigation system using ESP32 is significant not only for improving agricultural practices but also for addressing global challenges like water conservation, cost reduction, and sustainable farming. Its integration with IoT makes it a transformative solution for modern agriculture and gardening needs.

CONCLUSION

The development of a smart irrigation system using the ESP32 microprocessor is a significant step toward modernizing agricultural practices and promoting sustainable resource management. This project successfully demonstrates how IoT technology can be leveraged to optimize water usage, reduce costs, and improve crop health through precise and automated irrigation processes.

By integrating sensors, actuators, and real-time data analysis, the system ensures that water is delivered efficiently based on soil moisture levels and environmental conditions. The inclusion of remote monitoring and control capabilities further enhances its usability, making it suitable for both small-scale gardening and large-scale farming applications.

This project not only addresses critical challenges such as water scarcity and inefficient irrigation but also contributes to environmental sustainability by minimizing resource wastage. It provides a scalable solution that can be adapted to various agricultural scenarios, paving the way for smarter and more sustainable farming practices.

In conclusion, this smart irrigation system is an innovative and impactful solution that aligns with global efforts to conserve resources, improve agricultural productivity, and promote eco-friendly technologies. With further enhancements such as solar power integration or advanced nutrient monitoring, it has the potential to revolutionize the way irrigation is managed in the future.

WORK DIARY

Date	Aditya Singla (2023UCS1669)	Himanshu Kumar Bairwa (2023UCS1708)	Peeyush Gupta (2023UCS1717)
09/01/2025	Assigned Analysis on Research Paper and update the project file periodically and help in testing and execution	Assigned with all the hardware related part including the purchasing as well as testing of every part individually	Assigned with the market research part and learning the functioning of the IDE and the coding atmosphere.
16/01/2025	Updated the project file Read and analysed the research paper by “Sandra Millan.” Done the purchasing of Components Required for the project along with Himanshu and Peeyush.	Understood different hardware components, their significance and role in the project along with functionalities Done the purchasing of Components Required for the project along with Aditya and Peeyush.	Done the market research and the purchasing parts with other members and the need of ESP 32 in place of intel 8085. Done the purchasing of Components Required for the project along with Himanshu and Aditya.
Purchasing Completed			
23/01/2025	Did the documentation part and studied about the processors. Also studied the online simulator and got to know more about online environment. Also, in progress to design a User Interface with the help of Peeyush. Also provided necessary documents to the Himanshu to work on his hardware assembling.	Compiled all the hardware components and the materials. Also started working on compiling the project and studied researches and models that are already present. Test all the components and take the necessary action regarding them. Completed 90% of Hardware Assembling of basic model and currently working on Arduino code.	Studied about the intel 8085 processor and the ESP32. Also studied the instruction set and their details of intel 8085 and the ESP 32. Completed material purchasing and the market research. Studying the online simulator environment. Working with Aditya to design a desired user interface. Also helping Himanshu to write a correct Arduino Code.
30/01/2025	Documentation work and online simulation for the results from the existing model.	Started the hardware compiling of the model and tested the components.	Completed the code for the basic model that was just the soil moisture level detector.
06/02/2025	Did the assembling of the circuit of advanced model.	Did the assembling of the circuit of advanced model.	Did the assembling of the circuit of advanced model.
Progress Video Shared			
13/02/2025	Working on Interface Design.	Working on Interface Codes.	Working on Interface Codes.
Completed the project and tested the working of the Model			

COMPARISION WITH EXISTING SOLUTIONS

---

1. Hardware Enhancements:

We integrated a humidity sensor along with soil moisture and temperature sensors for more accurate environmental monitoring.

A relay module is used for efficient pump control, ensuring reliable automation.

We added a PIR sensor for motion detection, which helps in detecting animals that may harm the plants.

To ensure a stable power supply, we included a DC buck converter, allowing the system to run on either a rechargeable battery or direct household power using a 12V AC adapter.

Instead of ESP8266, we used ESP32-WROOM, which provides higher processing power, dual-core capability, more GPIO pins, better connectivity, and lower power consumption—making it ideal for IoT applications.

Circuit Enhancements: We used MOSFETs and diodes to improve power efficiency, reduce noise, and enhance sensor readings' stability, ensuring more accurate control of the system.

2. Software & Arduino Code Enhancements:

The video’s system only worked in auto mode, but plants need precise watering, especially for sensitive crops like fruits and vegetables.

Our system offers both auto and manual modes:

Manual Mode: Users can monitor live values of temperature, humidity, soil moisture, and PIR sensor data and control the pump remotely via Wi-Fi.

Auto Mode: The system intelligently controls the pump based on pre-calibrated sensor thresholds, ensuring optimized water usage.

We collected data from 12+ soil types and fine-tuned sensor thresholds for maximum accuracy.

Unlike the video, which only used the Arduino IDE, we designed a full UI on Blynk for a better user experience.

Our system sends real-time notifications if the water level is low or if motion is detected near the plants.

We optimized the Arduino code for maximum compatibility, ensuring efficient and stable control logic on any device.

Conclusion:

Our project is cost-effective, highly automated, and designed for real-world agricultural needs. The existing solution showcased a basic system, whereas our project includes advanced hardware, circuit optimizations, and intelligent software integration for better efficiency and accessibility.

LEARNING OUTCOME AND FUTURE SCOPE

Learning Outcomes:

1. **Understanding IoT Integration**: The project provides hands-on experience with integrating IoT technologies into agricultural practices, enhancing understanding of how sensors, microcontrollers, and cloud services work together to automate tasks.
2. **Sustainable Resource Management**: Participants learn about the importance of efficient water use and how smart systems can reduce waste while promoting plant health.
3. **Technical Skills Development**: The project develops skills in programming (e.g., using Arduino IDE), circuit design, and troubleshooting, which are essential for IoT-based projects.
4. **Data-Driven Decision Making**: The system demonstrates how real-time data can inform irrigation decisions, improving crop yields and reducing manual intervention.
5. **Environmental Awareness**: It highlights the role of technology in addressing environmental challenges like water scarcity and climate change.

Future Scope:

1. **Advanced Sensor Integration**: Incorporating additional sensors (e.g., pH, TDS) to monitor nutrient levels and further optimize irrigation strategies.
2. **Machine Learning Integration**: Implementing machine learning algorithms to predict optimal irrigation schedules based on historical data and weather forecasts.
3. **Solar Power Optimization**: Enhancing the system with more efficient solar panels and battery management systems for off-grid sustainability.

4. **Expansion to Large-Scale Farming**: Scaling the system for commercial agriculture by integrating multiple ESP32 units and expanding sensor networks.
5. **User-Friendly Interfaces**: Developing more intuitive mobile apps or web dashboards for easier monitoring and control, enhancing user experience.
6. **Integration with Other Smart Farming Technologies**: Combining the irrigation system with other smart farming tools (e.g., precision planting, autonomous farming equipment) to create comprehensive smart agriculture solutions.
7. **Cost Reduction and Accessibility**: Exploring cost-effective manufacturing methods to make the system more accessible to small-scale farmers and gardeners.

By exploring these future directions, the smart irrigation system can continue to evolve and contribute significantly to sustainable agriculture practices worldwide.

BIBLIOGRAPHY

---

1. Pereira, G. P., Chaari, M. Z., & Daroge, F. (2023). IoT-Enabled Smart Drip Irrigation System Using ESP32. *IoT*. Retrieved from Semantic Scholar.
2. Shiva Shankar, J., Palanivel, S., & China Venkateswarlu, S. (2021). IoT-Based Smart Irrigation System Using ESP32 and Adafruit IO. *International Conference on Emerging Technologies: AI, IoT, and CPS for Science & Technology Applications*. CEUR Workshop Proceedings.
3. Circuit Digest Team. (2024). Arduino Smart Irrigation System Using ESP32 and Blynk App. Retrieved from Circuit Digest.
4. Malar, K., & Vaishnavi, R. (2023). IoT-Based Smart Irrigation System Using ESP32. Retrieved from Semantic Scholar.
5. Design of a Smart Hydroponics Monitoring System Using an ESP32 Microcontroller and the Internet of Things. (2023). *PMC*. Retrieved from PubMed Central.
6. NextPCB Support Team. (2024). Smart Irrigation System Using ESP32. Retrieved from NextPCB Blog.
7. Automatic Irrigation Using ESP32 Research Paper. (2024). Retrieved from Academia.edu.
8. IoT-Enabled Smart Drip Irrigation System Using ESP32. (2023). *MDPI*.