ASSIGNMENT FINAL REPORT

Qualification	**Pearson BTEC Level 5 Higher National Diploma in Computing**		
Unit number and title	**Unit 45: Internet of Things**		
Submission date	December 4th, 2024	**Date Received 1st submission**	December 4th, 2024
Re-submission Date		**Date Received 2nd submission**	
Student Name	Hoang Anh Quy	**Student ID**	BS00311
Class	SE07101	**Assessor name**	Mr. NGUYEN VAN VU

specification. I certify I have clearly referenced any sources and any artificial intelligence (AI) tools used in the work. I understand that making a false declaration is a form of malpractice.	
Student's signature	Quy

Grading grid

P1	P2	P3	P4	P5	P6	P7	P8	M1	M2	M3	M4	M5	M6	D1	D2	D3

❐ Summative Feedback:	❐ Resubmission Feedback:	
Grade:	Assessor Signature:	Date:
Internal Verifier's Comments:		
Signature & Date:		

Contents

List of Figures:

List of Tables:

Introduction.

The Internet of Things (IoT) has revolutionized the way we interact with the physical world by enabling everyday objects to connect, collect, and exchange data through the internet. As a prime example of IoT's impact, smart plant pots with automated watering systems have gained significant attention for their potential to enhance plant care and environmental sustainability. These IoT-powered systems use various sensors to monitor conditions such as soil moisture, temperature, and light, adjusting watering schedules accordingly. In this assignment, we explore the application of IoT in creating a smart, self-watering plant pot system, discussing the underlying architecture, frameworks, hardware, and APIs used to develop such systems. We also identify challenges encountered during the integration of these systems into broader IoT networks and propose solutions to ensure smooth operation and scalability. By examining the key components and their interactions, this paper aims to highlight the benefits of IoT in enhancing everyday life and its potential in the field of automation and smart environments.

ASM Part 1

1 Explore various forms of IoT functionality on electronic platforms. P1

1.1 What is IoT? (Introduction to IoT).

1.1.1 Definition of IoT (Internet of Things).

According to the book Internet of Things (A Hands-on-Approach) written by Arshdeep Bahga and Vijay Madisetti, the Internet of Things (IoT) is a network of physical devices, vehicles, appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity that enable these devices to collect, exchange data, and process data autonomously without human intervention.

In simpler terms, it is a way of connecting everyday objects to the internet, allowing them to send and receive data. This can include everything from thermostats and refrigerators to cars and industrial machinery, sensors, home appliances, vehicles, and many other types of machines, which are integrated with technology to communicate and interact with their surroundings. (Arshdeep Bahga, Vijay Madisetti, 2014), (aws, n.d.)

Figure 1 Definition of IoT (Internet of Things).

1.1.2 Description of how IoT devices connect with each other and share data over the internet.

The process of connecting and sharing data takes place through the following main steps:

1. Data collection: Sensors on IoT devices record information from the environment, such as temperature, humidity, light, or motion.
2. Data connection and transmission: Data is transmitted from the device to a cloud platform or server via an internet connection.
3. Processing and analysis: The cloud platform or data processing system analyzes the collected information to make appropriate decisions or responses.
4. Response: The analysis results are sent back to the device, helping it perform automated actions (e.g., turning on lights, adjusting the temperature, or sending notifications to the user).

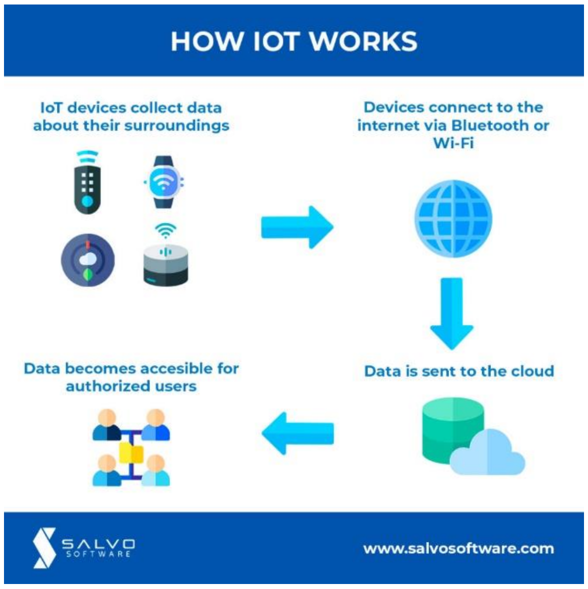

HOW IOT WORKS
IoT devices collect data about their surroundings
Devices connect to the internet via Bluetooth or Wi-Fi
Data is sent to the cloud
Data becomes accesible for authorized users
SALVO SOFTWARE
www.salvosoftware.com

Figure 2 Description of how IoT devices connect with each other and share data over the internet.

IoT devices connect and share data over the internet through a combination of technologies: (Hoi, 2018),

1. Network Connections:
    - Wired Connections: Some devices use wired connections like Ethernet for stable and secure data transfer. This is common for industrial setups where reliability is crucial.
    - Wireless Connections: Most IoT devices rely on wireless connections due to their flexibility and ease of deployment. Common options include:
    - Wi-Fi: Widely available and offers decent range for data transfer within a home or office.
    - Cellular Networks: Provide wider coverage for remote devices, like sensors in agricultural fields.
    - Low-Power Wide Area Networks (LPWAN): Designed for long-range communication with low power consumption, ideal for battery-powered devices. Examples include Bluetooth Low Energy (BLE) and Zigbee.
2. Communication Protocols:

    These protocols define a common language for devices to understand each other. Here are some key players:
    - TCP/IP: The foundation of internet communication, also used by many IoT devices.
    - MQTT (Message Queuing Telemetry Transport): Lightweight messaging protocol ideal for sending small amounts of data from resource-constrained devices.
    - CoAP (Constrained Application Protocol): Similar to HTTP but simplified for use in low- power devices.
3. Data Flow:

    The data flow can involve several components:
    - Sensors: Collect data from the physical environment (temperature, motion, etc.).
    - Device: Processes the sensor data and prepares it for transmission.
    - Gateway (Optional): Acts as a central hub for multiple devices, aggregating data and potentially performing local processing before sending it onwards.
    - Cloud Platform: A remote server that receives, stores, analyzes, and manages the data. This is where complex computations and large-scale data storage happen.
    - Applications: Mobile apps, web interfaces, or other software tools that visualize and interact with the data, allowing users to monitor, control, or gain insights.

1.1.3 Explanation of the role of IoT in modern life.

The Internet of Things (IoT) plays a vast and growing role in modern life, impacting everything from our homes to our cities.

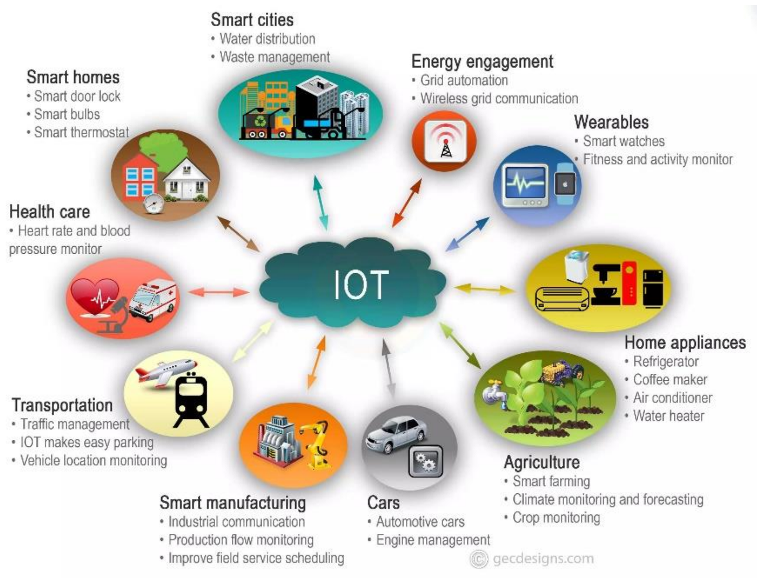


Figure 3 Explanation of the role of IoT in modern life.

Here's a breakdown of its key contributions: (fast, 2023), (Viettelz, 2023)

Convenience and Automation:

- Smart Homes: We can control lights, thermostats, appliances, and even locks remotely using our smartphones or voice assistants. Imagine a coffee maker that starts brewing your coffee pot right before you wake up or a thermostat that adjusts the temperature based on your preferences.

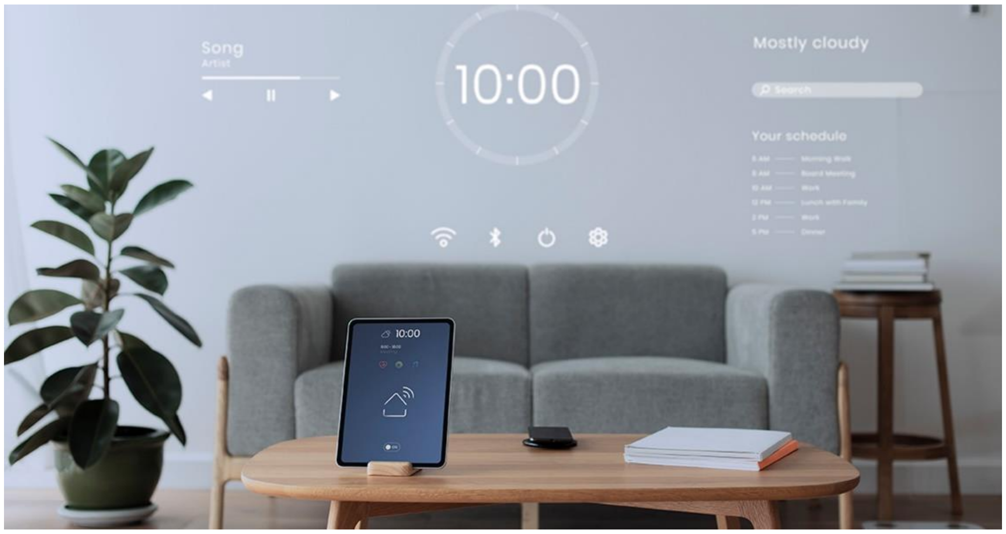


Figure 4 Smart home and office.

- Wearable Technology: Fitness trackers monitor our health, smartwatches display notifications and control music, while smart glasses can provide augmented reality experiences.

Figure 5 Wearable Technology.

Enhanced Efficiency and Productivity:

- Connected Cars: Cars can collect data on traffic conditions, engine performance, and parking availability, leading to smoother commutes and improved fuel efficiency.
- Smart Manufacturing: Factories use IoT sensors to monitor machines, predict maintenance needs, and optimize production lines, leading to increased efficiency and reduced downtime.
- Smart Cities: Traffic lights adjust based on real-time traffic data, while sensors optimize waste collection and energy use in buildings.

Improved Safety and Security:

- Smart Security Systems: Homes and businesses can be equipped with interconnected cameras, motion sensors, and alarms for better security and remote monitoring. Doorbells with cameras can allow you to see who's at your door even when you're not home.
- Wearable Health Monitors: Devices can track heart rate, blood sugar, and other vital signs, allowing for early detection of potential health issues.

Data-Driven Decision Making:

- Retail and Inventory Management: Businesses can track inventory levels with IoT sensors, allowing for automatic reordering and preventing stockouts.
- Agriculture: Farmers can use sensors to monitor soil moisture, crop health, and weather conditions, optimizing irrigation and applying fertilizer precisely where needed.

Figure 6 Smart Agriculture.

1.2 History of IoT (The history and development of IoT)

Figure 7 History of IoT.

1.2.1 Overview of the History of IoT Development Since the 1980s.

The concept of the Internet of Things (IoT) has evolved over decades, transitioning from theoretical ideas to practical applications that influence daily life. Its development began in the 1980s and has gained momentum with technological advancements. (Foote, 2022)Here is an overview:

- 1980s: Early Concepts
    - Early Precursors: While not exactly "internet" connected, the 1800s saw the rise of communication technologies like the telegraph, foreshadowing the idea of interconnected devices. The 1980s continued this trend with the use of remote sensors for industrial automation, laying the groundwork for sensor-based data collection.
    - The Birth of an Idea: In 1982, a group of students at Carnegie Mellon University took things a step further. They modified a Coca-Cola vending machine to connect to the internet, allowing them to monitor its inventory and even check the temperature of the drinks remotely. This is often considered the first internet-connected appliance, sparking imaginations about a future of connected devices.
- 1990s: Conceptual Frameworks

The term "Internet of Things" was first coined by Kevin Ashton in 1999 while working at Procter & Gamble. Ashton described IoT as a system where the internet could sense the physical world through RFID (Radio Frequency Identification) technology.

- 2000s: Initial Implementation
  IoT started gaining traction with advancements in wireless technology, low-cost sensors, and computing power. The launch of IPv6 in 1998-2006 enabled a vast number of devices to connect to the internet. Key applications included smart home devices and industrial IoT (IIoT).
- 2010s: Expansion and Commercialization
  IoT became more accessible due to the proliferation of smartphones, cloud computing, and big data analytics. Platforms like Amazon Alexa and Google Home revolutionized smart home applications. IoT began to impact industries such as healthcare, agriculture, and transportation.
- 2020s: IoT in the Era of AI and 5G
  With the integration of artificial intelligence (AI) and the deployment of 5G networks, IoT systems have become more intelligent and efficient. Edge computing has further enhanced real-time data processing, and IoT is now central to the development of smart cities and autonomous vehicles.

1.2.2 Key milestones in the development of IoT.

Key Milestones in the Development of IoT (Braun, 2019)

- 1982: Introduction of the first internet-connected appliance – a Coca-Cola vending machine at Carnegie Mellon University.
- 1999: Kevin Ashton coins the term "Internet of Things" while proposing RFID technology.
- 2008-2009: The number of devices connected to the internet surpasses the global human population.
- 2011: The World Economic Forum identifies IoT as a key emerging technology.
- 2014: Launch of the Amazon Echo, a major milestone in consumer IoT.
- 2020: Deployment of 5G networks accelerates the adoption of IoT in various sectors.

1.2.3 Technological advancements that have driven the development of IoT.

- Wireless Connectivity: Advances in Wi-Fi, Bluetooth, Zigbee, and cellular technologies like 4G and 5G have enabled seamless device communication.
- Sensor Technology: Miniaturized and cost-effective sensors have made it possible to capture data from the environment with high accuracy.

- Cloud Computing: The availability of scalable cloud platforms allows IoT devices to store, process, and analyze data in real time.
- Big Data and Analytics: The ability to analyze massive datasets has made IoT systems more insightful, enabling predictive maintenance and personalized experiences.
- Artificial Intelligence and Machine Learning: AI enables IoT devices to learn from data, make autonomous decisions, and improve over time, enhancing functionality.
- Edge Computing: Processing data closer to the source reduces latency and improves the efficiency of IoT systems.
- IPv6 Adoption: With an almost infinite number of IP addresses, IPv6 supports the connection of billions of devices.

1.3 Characteristics of IoT (The characteristics and features of IoT).

1.3.1 Connectivity.

This is the foundation of IoT. Devices need a way to connect to the internet and each other to share data and function as a network. Common connection methods include Wi-Fi, Bluetooth, cellular networks, and Low-Power Wide Area Networks (LPWAN).

1.3.2 Real-time Interaction.

Unlike traditional pre-programmed devices, IoT devices can collect and exchange data in real-time. This allows for near-instantaneous responses and adjustments based on the data collected. Imagine a thermostat that automatically adjusts the temperature based on real-time weather data or a fitness tracker that transmits your heart rate live during a workout.

1.3.3 Data Analytics.

The massive amount of data generated by IoT devices needs to be analyzed to unlock its true potential. This data can reveal trends, identify patterns, and provide valuable insights. For instance, data from smart sensors in a factory can be analyzed to predict equipment failure and schedule preventive maintenance.

1.3.4 Automation.

A core benefit of IoT is its ability to automate tasks. By analyzing data and responding accordingly, devices can perform actions without human intervention. Smart irrigation systems automatically water lawns based on moisture levels, while connected lighting systems can adjust brightness based on the time of day.

1.3.5 Scalability.

IoT systems are designed to grow and adapt. The ability to easily add new devices to an existing network is crucial. This scalability allows for implementation in various settings, from small home automation systems to large-scale industrial applications.

1.4 Why use IoT?

1.4.1 Benefits of IoT in enhancing efficiency, reducing costs, and improving quality of life.

1.4.1.1 Enhancing Efficiency

- Automation of Tasks: IoT enables real-time monitoring and automated control of systems, reducing manual intervention and streamlining operations.
- Data-Driven Decisions: IoT devices collect and analyze data, providing actionable insights to optimize workflows and improve efficiency.
- Improved Maintenance: Predictive maintenance using IoT sensors minimizes downtime by detecting issues before they become critical.

1.4.1.2 Reducing Costs

- Energy Savings: Smart energy management systems reduce power consumption by adjusting lighting, heating, and cooling based on occupancy and usage patterns.
- Operational Optimization: IoT enhances supply chain and inventory management, reducing waste and minimizing storage costs.
- Asset Utilization: IoT tracks equipment usage and ensures assets are optimally deployed, avoiding unnecessary expenditures.

1.4.1.3 Improving Quality of Life

- Smart Homes: IoT-enabled devices like smart thermostats, security cameras, and voice assistants enhance convenience, security, and energy efficiency at home.
- Healthcare Monitoring: Wearable devices and remote health monitoring systems provide real-time data, improving patient care and emergency response times.
- Transportation and Mobility: IoT powers smart transportation systems, offering real-time traffic updates, navigation, and efficient public transport scheduling.

1.4.2 How IoT helps optimize business processes and resource management.

1.4.2.1 Streamlined Operations

- IoT devices provide continuous monitoring and tracking of production processes, identifying inefficiencies and improving throughput.
- Connected devices allow for just-in-time manufacturing, reducing excess inventory and waste.

1.4.2.2 Improved Resource Allocation

- IoT systems monitor resource usage (e.g., water, energy, and materials), enabling businesses to allocate resources more efficiently and sustainably.
- Sensors in smart buildings optimize energy usage by adjusting lighting, temperature, and ventilation based on occupancy.

1.4.2.3 Enhanced Supply Chain Management

- IoT enables real-time tracking of shipments and inventory, improving transparency and reducing delays.
- Predictive analytics powered by IoT data helps businesses anticipate demand and adjust supply accordingly.

1.4.2.4 Better Customer Experiences

- IoT provides personalized experiences through real-time data analysis, such as recommending products or services based on customer behavior.
- Automated systems powered by IoT reduce response times and enhance service reliability.

1.4.2.5 Operational Safety

IoT devices monitor working conditions, detect hazards, and ensure compliance with safety standards, reducing accidents and liabilities.

1.5 Application of IoT (Areas of IoT application).

Smart Home: This is a familiar concept for many. It involves connecting household appliances and devices to the internet, allowing for remote control, automation, and monitoring. Examples include:

- Smart Lighting: Control lights remotely, adjust brightness based on preferences or time of day, and even integrate with motion sensors for automatic activation.
- Smart Thermostats: Program heating and cooling schedules for optimal comfort and energy efficiency.

- Smart Security Systems: Monitor your home remotely with connected cameras, door locks, and motion sensors, receive alerts in case of security breaches.
- Smart Appliances: Control your oven, refrigerator, or washer and dryer remotely, receive notifications when cycles are complete, and even optimize energy consumption.

Smart Car: The future of transportation is heavily influenced by IoT. Here are some exciting applications:

- Connected Car Features: Access features like remote lock/unlock, vehicle diagnostics, real- time traffic updates, and even remote engine start.
- Autonomous Driving: Self-driving cars rely heavily on a network of sensors, cameras, and internet connectivity to navigate roads safely and efficiently.
- Advanced Driver Assistance Systems (ADAS): Features like lane departure warnings, blind spot monitoring, and automatic emergency braking utilize sensors and connectivity to enhance safety on the road.
- In-Vehicle Connectivity: Stay connected with Wi-Fi hotspots, access entertainment systems, and receive personalized recommendations based on your location.

Smart Health: IoT is revolutionizing healthcare by enabling remote patient monitoring and improving overall health management. Examples include:

- Wearable Health Monitors: Track heart rate, blood pressure, activity levels, and sleep patterns to provide valuable insights into your health.
- Remote Patient Monitoring: Allow doctors to monitor patients with chronic conditions remotely, facilitating timely intervention and improved care management.
- Medication Management Systems: Smart pill dispensers remind patients to take medications and can even track adherence to treatment plans.
- Connected Medical Devices: Implantable devices like pacemakers can transmit data wirelessly, allowing for real-time monitoring and adjustments by healthcare professionals.

Smart Industry: Manufacturing and industrial processes are undergoing significant transformations with the help of IoT. Here are some key applications:

- Industrial Automation: Production lines can be automated using connected sensors and robots, optimizing efficiency and reducing human error.
- Predictive Maintenance: Sensors can monitor equipment health and performance, allowing for preventive maintenance and minimizing downtime.
- Supply Chain Management: Track goods and inventory in real-time using connected devices, optimize logistics and ensure timely delivery.
- Smart Warehouses: Automated warehouse systems with connected robots and inventory management software improve efficiency and accuracy.

2 Review standard architecture, frameworks, tools, hardware, and APIs available for use in IoT development. P2

2.1 Architecture of IoT.

Figure 8 Architecture of IoT.

The architecture of IoT (Internet of Things) can be visualized as a layered approach, with each layer playing a crucial role in enabling communication and data flow between devices and applications. Here's a breakdown of the four main layers:

2.1.1 Perception Layer:

- This layer forms the foundation, consisting of the physical devices and sensors that gather data from the environment.
- Sensors can be of various types, measuring things like temperature, pressure, motion, light, or even chemical composition.

- Examples of devices in this layer include smart thermostats with temperature sensors, wearables with heart rate monitors, or industrial machines with vibration sensors for monitoring equipment health.

2.1.2 Network Layer:

- This layer is responsible for transmitting the data collected by the sensors to other devices and systems.
- Various communication protocols and technologies are used in this layer, depending on the application and requirements.
- Common options include Wi-Fi, Bluetooth, cellular networks, and Low-Power Wide Area Networks (LPWAN) for long-range communication with low power consumption.

2.1.3 Processing Layer:

- The data collected from the sensors needs to be analyzed and processed to extract meaningful insights.
- This layer can be located on the edge (closer to the sensors) or in the cloud depending on the complexity of processing required and resource limitations of the devices.
- Edge devices may perform basic processing tasks like filtering or aggregation, while the cloud handles more complex computations and large-scale data analysis.

2.1.4 Application Layer:

- This layer interacts with the end-users and provides the actual functionality and value of the IoT system.
- It consists of applications, software tools, and user interfaces that allow users to visualize data, control devices, and interact with the system.
- Examples include mobile apps for controlling smart home devices, dashboards for monitoring industrial processes, or healthcare platforms for analyzing patient data collected by wearables.

2.2 Frameworks of IoT.

Introduce and describe a few frameworks used, including:

2.2.1 Node-RED:

Node-RED is a flow-based development tool for visual programming, developed by IBM for wiring together hardware devices, APIs, and online services. It is built on Node.js, providing a browser-based editor that makes it easy to create IoT applications by connecting various nodes (components) through a drag-and-drop interface.

Support for IoT Development and Deployment:

- **Visual Programming:** Node-RED's visual interface simplifies the process of designing and deploying IoT applications. Users can connect nodes representing different devices, services, and data flows without extensive coding.
- **Node Ecosystem:** Node-RED has a vast library of pre-built nodes that support various IoT protocols (e.g., MQTT, HTTP, WebSockets) and integration with numerous cloud services (e.g., AWS IoT, IBM Watson IoT).
- **Real-time Data Processing:** The framework can handle real-time data streams, allowing for quick prototyping and deployment of IoT solutions.
- **Extensibility:** Developers can create custom nodes to extend Node-RED's functionality,
- enabling tailored solutions for specific IoT use cases.
- **Community and Support:** Node-RED has a strong community and extensive documentation, making it accessible for both beginners and experienced developers.

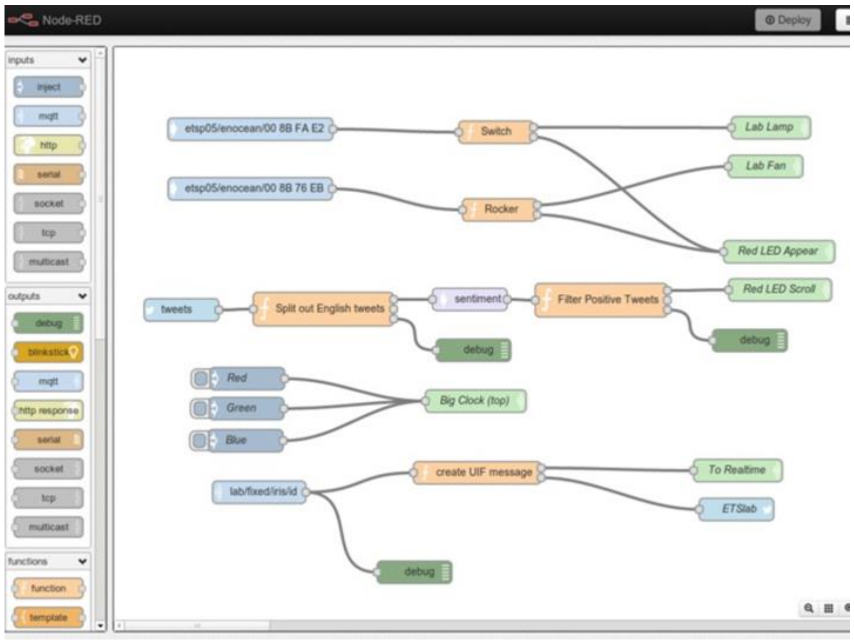


Figure 9 Node-RED.

2.2.2 Kaa.

Kaa is an open-source middleware platform that facilitates the development of end-to-end IoT solutions. It provides a comprehensive set of features for managing connected devices, data collection, and real-time analytics.

Support for IoT Development and Deployment:

- **Device Management:** Kaa offers robust device management capabilities, allowing developers to manage device connectivity, provisioning, and firmware updates.
- **Data Collection and Analytics:** The platform supports data collection from connected devices, enabling real-time analytics and insights through its powerful data management tools.
- **Interoperability:** Kaa supports multiple IoT protocols and standards, making it easy to integrate with various devices and third-party services.
- **Customizable and Scalable:** The platform is highly customizable and scalable, suitable for both small-scale and large-scale IoT deployments.
- **Security:** Kaa includes built-in security features such as device authentication, data encryption, and secure communication channels to protect IoT deployments from threats.
- **Multi-tenancy:** Kaa supports multi-tenancy, allowing different projects and organizations to operate within a single Kaa instance.

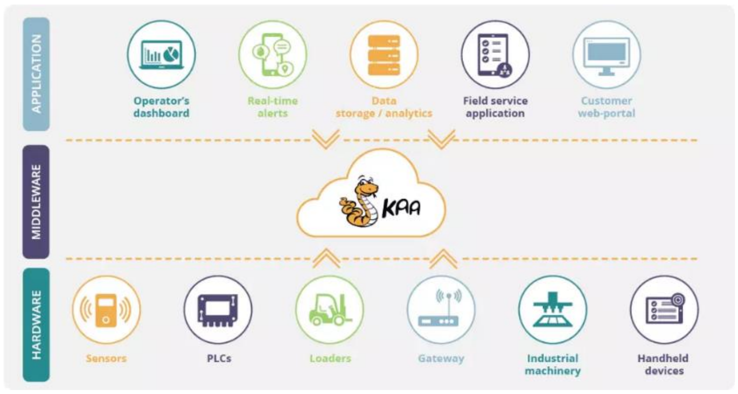


Figure 10 Kaa.

2.2.3 OpenIoT.

OpenIoT is an open-source IoT platform that enables the creation of scalable and flexibleIoT applications. It focuses on providing interoperability and integration of various IoT devices andservices.

Support for IoT Development and Deployment:

- **Sensor Middleware:** OpenIoT acts as middleware for connecting sensors and actuators, providing a standard interface for data collection and control.
- **Cloud Integration:** The framework supports integration with cloud services for data storage, processing, and analytics, leveraging the scalability and reliability of cloud infrastructures.
- **Semantic Data Processing:** OpenIoT includes semantic data processing capabilities, allowing for advanced data analytics and context-aware applications.
- **Service Delivery:** The platform supports the dynamic discovery and composition of IoT services, enabling flexible and adaptive IoT applications.
- **Interoperability:** OpenIoT promotes interoperability between heterogeneous IoT devices and

systems, facilitating seamless integration and communication.

- **Open Standards:** The platform adheres to open standards and protocols, ensuring broad compatibility and reducing vendor lock-in.

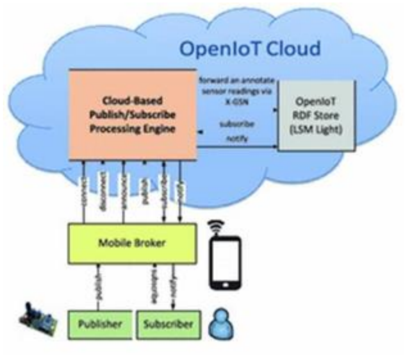


Figure 11 OpenIoT.

2.3 Describe how Arduino/Raspberry Pi board and end device.

2.3.1 Arduino

Arduino: Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's widely used by hobbyists, students, and professionals to create interactive projects. The Arduino platform consists of various boards catering to different needs and complexities of projects.

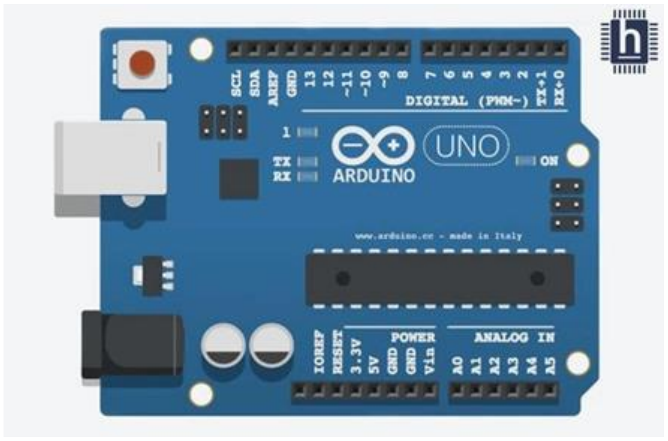


Figure 12 Arduino.

2.3.1.1 Various Arduino Boards:

Arduino Uno: The most popular and widely used board, ideal for beginners. It features an ATmega328 microcontroller.

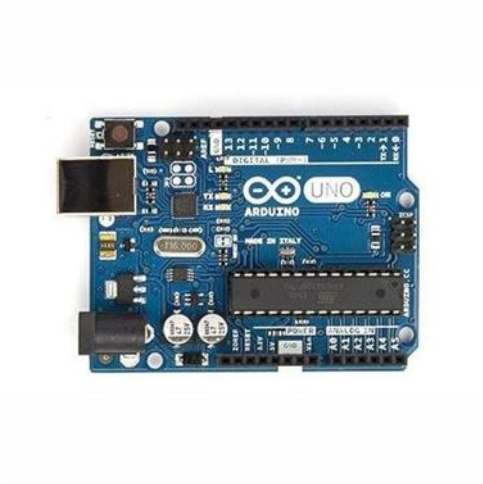


Figure 13 Arduino Uno.

Arduino Mega: Designed for more complex projects, it has more I/O pins and memory, featuringan ATmega2560 microcontroller.

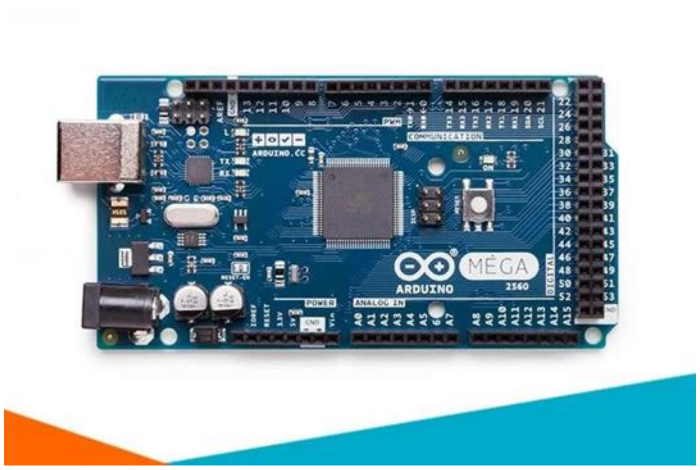


Figure 14 Arduino Mega.

Arduino Nano: A compact board suitable for breadboard prototyping, with an ATmega328microcontroller.

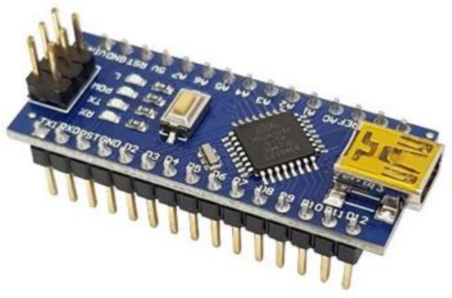


Figure 15 Arduino Nano.

Arduino Leonardo: Features a microcontroller with built-in USB communication, useful forcreating USB devices.

Figure 16 Arduino Leonardo.

Arduino MKR Series: Aimed at IoT projects, these boards come with integrated connectivityoptions like Wi-Fi (MKR WiFi 1010), LoRa (MKR WAN 1300), and GSM (MKR GSM 1400).

Figure 17 MKR Wi-Fi 1010.

2.3.1.2 Using Arduino in IoT Projects.

Programming: Arduino boards are programmed using the Arduino IDE, which uses a simplifiedversion of C++.

Connectivity: For IoT projects, Arduino can be connected to the internet using various shields (e.g.,Wi-Fi, Ethernet) or modules (e.g., ESP8266).

Data Collection and Control: Arduino boards can interface with sensors to collect data and withactuators to control devices, making them ideal for monitoring and automation tasks.

Example Projects: Environmental monitoring systems, home automation, and wearable devices.

2.3.2 Raspberry Pi

Raspberry Pi: Raspberry Pi is a series of small, single-board computers developed by the Raspberry Pi Foundation. Unlike Arduino, a microcontroller-based platform, Raspberry Pi is a full-fledged computer running a Linux-based operating system.

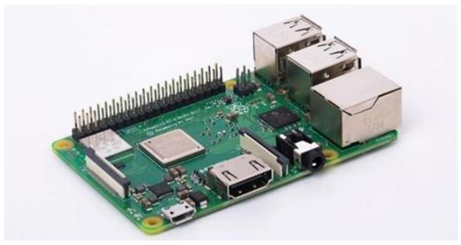


Figure 18 Raspberry Pi.

2.3.2.1 Versions of Raspberry Pi:

Raspberry Pi Model B: The first version, followed by several iterations improving performance andconnectivity.

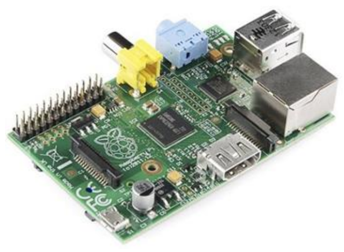


Figure 19 Model B

Raspberry Pi Zero: A smaller, cost-effective version with limited I/O but suitable for compact projects.

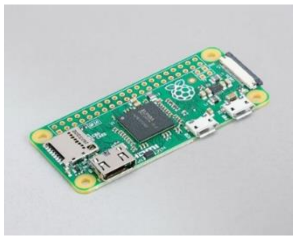


Figure 20 Raspberry Pi Zero.

Raspberry Pi 3 Model B+: Enhanced version with better CPU, Wi-Fi, Bluetooth, and Ethernet.

Figure 21 Pi 3 Model B+.

Raspberry Pi 4 Model B: Latest version with significant improvements, including USB 3.0, moreRAM options, and better CPU and GPU

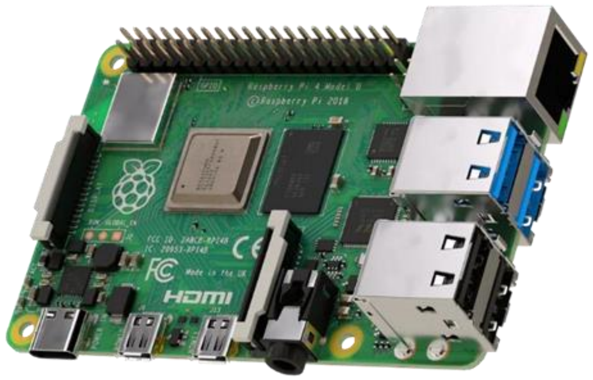


Figure 22 Pi 4 Model B

2.3.2.2 Using Raspberry Pi in IoT Projects:

Operating System: Typically runs Raspbian OS (a Debian-based Linux OS) but can run other OSesas well.

Programming: Supports multiple programming languages such as Python, Java, and C++. Python isparticularly popular for its ease of use.

Connectivity: Built-in Ethernet, Wi-Fi, and Bluetooth enable easy connectivity to the internet andother devices.

Data Processing: With its higher processing power, Raspberry Pi can handle more complex taskslike data analysis and image processing.

Example Projects: Home automation systems, personal web servers, IoT gateways, and AI-powered devices.

2.3.3 End Device: Description of how to connect sensors and actuators to Arduino/Raspberry Pi

- **Connectivity:** Built-in Ethernet, Wi-Fi, and Bluetooth enable easy connectivity to theinternet and other devices.
- **Data Processing:** With its higher processing power, Raspberry Pi can handle more complextasks like data analysis and image processing.
- **Example Projects:** Home automation systems, personal web servers, IoT gateways, and AI-powered devices.

2.3.3.1 Connecting Sensors and Actuators.

2.3.3.1.1 Connecting Sensors and Actuators to Arduino.

- **Analog and Digital Pins:** Arduino boards come with a variety of analog and digital pins thatcan read sensor inputs and control actuators.
- **Libraries:** Arduino IDE offers numerous libraries for interfacing with different sensors and actuators, simplifying the programming process.

Examples:

- **Temperature Sensor (DHT11/DHT22):** Connect the sensor to an analog pin, read thetemperature data, and upload it to the cloud.
- **LED:** Connect an LED to a digital pin and control it via the Arduino program to create asimple lighting system.
- **Servo Motor:** Use PWM (Pulse Width Modulation) pins to control the position of a servomotor for robotics applications.

2.3.3.1.2 Connecting Sensors and Actuators to Raspberry Pi:

- **GPIO Pins:** Raspberry Pi features General Purpose Input/Output (GPIO) pins to connectwith sensors and actuators.
- **Libraries and Tools:** The GPIO pins can be controlled using libraries such as RPi.GPIO forPython, pigpio, or WiringPi.

Examples:

- **Motion Sensor (PIR):** Connect the PIR sensor to GPIO pins to detect motion and triggerevents or notifications.
- **Camera Module:** Use the official Raspberry Pi Camera Module for capturing images andvideos, useful for surveillance or image processing projects.

- **Relays:** Connect relays to GPIO pins to control high-power devices like lights and motorsfrom the Raspberry Pi.

2.4 Sensor.

Sensors are an important component in IoT systems, allowing devices to collect data from their environment. Here, we will introduce common types of sensors used in IoT including temperature, humidity, light, motion, and gas sensors.

2.4.1 Introduction to common sensors in IoT: temperature, humidity, light, motion, and gas sensors.

2.4.1.1 Temperature Sensors

Introduction: Temperature sensors measure the degree of heat or cold in an environment. They are widely used in climate control, weather stations, and industrial processes.

Types:

Thermistors: Resistive devices were resistance changes with temperature. They are inexpensive and have a simple design but are limited to a specific temperature range.

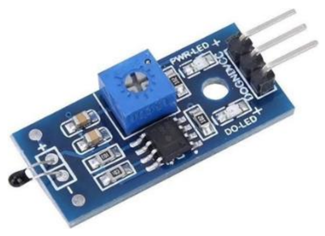


Figure 23 Thermistors.

Thermocouples: Made from two different metals joined together, generating a voltage proportional to temperature. Suitable for high-temperature ranges.

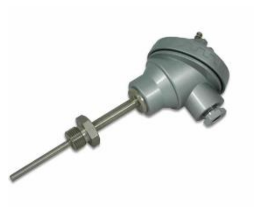


Figure 24 Thermocouples.

Digital Sensors (e.g., DS18B20): Provide digital output directly, often used with microcontrollersand development boards.

Figure 25 Digital Sensors

2.4.1.2 Humidity Sensors

Introduction: Humidity sensors (or hygrometers) measure the amount of water vapor in the air. They are essential for HVAC systems, agriculture, and meteorological stations.

Types:

- **Capacitive:** Measure changes in capacitance due to humidity variations. They are accurate andhave a wide range.

Figure 26 Capacitive.

- **Resistive:** Measure changes in electrical resistance caused by humidity. They are typically lessaccurate than capacitive sensors.

Figure 27 Resistive.

- **Thermal:** Measure changes in thermal conductivity due to humidity. They are less common butuseful in specific applications.

Figure 28 Therma.

2.4.1.3 Light Sensors.

Introduction: Light sensors (or photodetectors) measure the intensity of light in theirenvironment. They are used in smart lighting, security systems, and wearable devices.

Types:

- **Photoresistors (LDR):** Change resistance based on light intensity. They are simple and inexpensivebut less precise.

Figure 29 Photoresistors.

- **Photodiodes:** Convert light into current. They are more accurate and responsive.

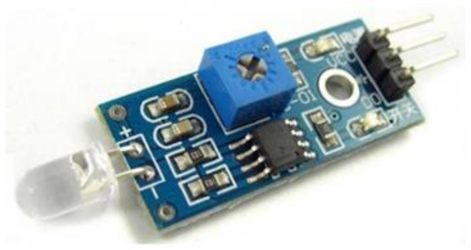


Figure 30 Photodiodes.

- **Phototransistors:** Similar to photodiodes but with amplification, providing higher sensitivity.

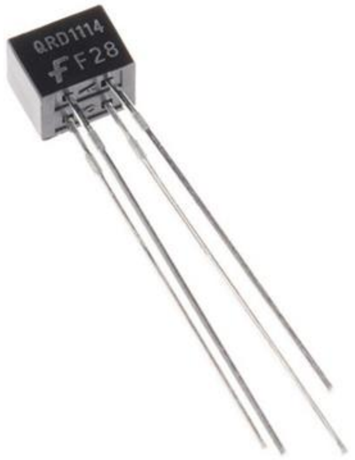


Figure 31 Phototransistors.

2.4.1.4 Motion Sensors

Introduction: Motion sensors detect movement in their vicinity. They are used in security systems, automation, and interactive installations.

Types:

- **Passive Infrared (PIR):** Detects infrared radiation from moving objects. Commonly used in security systems and automatic lighting.

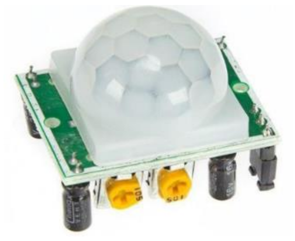


Figure 32 Passive Infrared.

- **Ultrasonic:** Emit ultrasonic waves and measure the reflection to detect motion. Useful for precisedistance measurement.

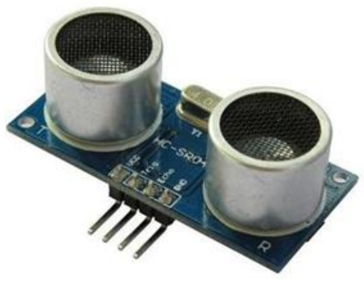


Figure 33 Ultrasonic.

- **Radar:** Use radio waves to detect motion and provide more robust performance in various conditions.

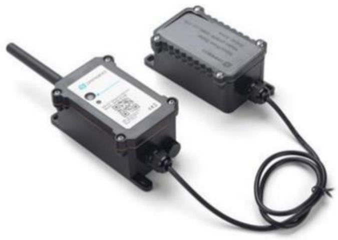


Figure 34 Radar.

2.4.1.5 Gas Sensors

Introduction: Gas sensors detect the presence and concentration of gases in the environment. They are crucial in safety, environmental monitoring, and industrial applications.

Types:

- **Electrochemical:** Measure gas concentration through a chemical reaction producing a current. They are highly sensitive and specific.

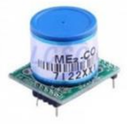


Figure 35 Electrochemical.

- **Metal Oxide:** Change resistance when exposed to gases. They are robust and have a long lifespan.

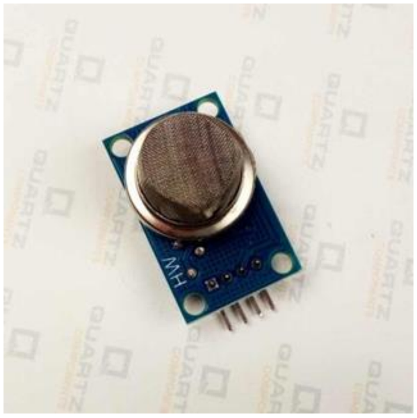


Figure 36 Metal Oxide.

- **Infrared (IR):** Detect gases by measuring absorption of IR light. Suitable for detecting gases like CO2 and methane.

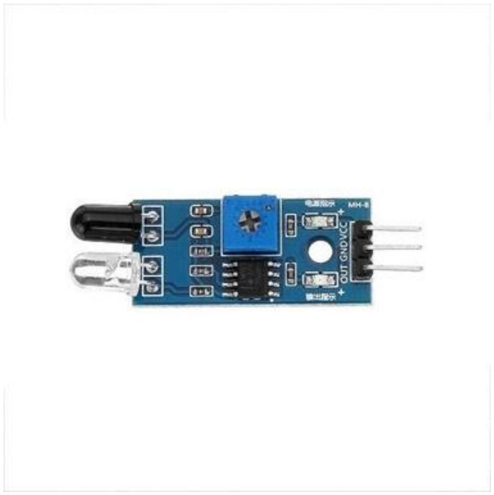


Figure 37 Infrared.

2.4.2 How to select and use different types of sensors in various IoT applications.

2.4.2.1 Determine the Application Requirements.

- Environment: Consider environmental conditions (temperature, humidity, presence of interference) where the sensor will operate.
- Accuracy and Precision: Select sensors that meet the accuracy and precision requirements of your application.
- Range: Ensure the sensor can measure within the necessary range for your application.
- Response Time: Choose sensors with an appropriate response time for the application's needs.

2.4.2.2 Compatibility with Hardware:

- Interfacing: Check if the sensor is compatible with your microcontroller or development board (e.g., Arduino, Raspberry Pi).
- Power Requirements: Ensure the sensor's power requirements match the available power supply.
- Signal Type: Verify if the sensor provides analog or digital output and ensure compatibility with the input pins of your controller.

2.4.2.3 Integration and Calibration:

- Libraries and Drivers: Use available libraries and drivers for your chosen platform to simplify integration.
- Calibration: Calibrate sensors as necessary to maintain accuracy. Some sensors may require periodic recalibration.

2.4.2.4 Data Processing and Communication:

- Data Processing: Implement necessary data processing algorithms to filter and interpret sensor data.
- Communication Protocols: Choose appropriate communication protocols (e.g., I2C, SPI, UART, MQTT) for transmitting data to the cloud or other devices.

2.5 API for use in IoT development

APIs (Application Programming Interfaces) play a crucial role in facilitating communication between IoT devices and central systems. Here, we will introduce three popular APIs used in IoT development including MQTT, CoAP, and HTTP/REST. Besides, we describe how they are used to enable seamless data exchange and control in IoT applications.

2.5.1 MQTT (Message Queuing Telemetry Transport)

2.5.1.1 Introduction.

MQTT is a lightweight messaging protocol designed for constrained devices and low-bandwidth, high-latency, or unreliable networks. It uses a publish/subscribe model, making it ideal for IoT applications that require efficient, real-time communication.

2.5.1.2 How to Use MQTT.

Architecture:

- **Broker:** The central server that manages message distribution. Popular brokers includeMosquitto, HiveMQ, and AWS IoT.

- **Clients:** Devices or applications that publish messages to topics or subscribe to topics to receive messages.

Communication Flow:

- **Publish:** A device (client) sends data to a specific topic on the broker.
- **Subscribe:** Other devices (clients) subscribe to the topic to receive the published data.
- **Quality of Service (QoS):** MQTT supports three levels of QoS to ensure message deliveryreliability:
- **QoS 0:** At most once delivery (no acknowledgment).
- **QoS 1:** At least once delivery (acknowledgment required).
- **QoS 2:** Exactly once delivery (highest reliability).

2.5.2 CoAP (Constrained Application Protocol)

2.5.2.1 Introduction.

CoAP is a specialized web transfer protocol designed for use with constrained nodes and networksin IoT. It is based on the REST model, similar to HTTP, but optimized for low-power and lossy networks.

2.5.2.2 How to Use CoAP.

Architecture:

- **Server:** The device that hosts resources (sensors/actuators) and responds to requests.
- Client: The device or application that sends requests to the server to interact withresources.

Communication Flow:

- **Requests and Responses:** CoAP uses GET, POST, PUT, and DELETE methods similar to HTTPto interact with resources.
- **Observe Option:** Allows clients to observe changes in a resource and receive notificationswhen the resource changes.
- **Message Transmission:** CoAP messages are exchanged using UDP (User DatagramProtocol) to minimize overhead.
- **Security**: CoAP can use DTLS (Datagram Transport Layer Security) to provide secure communication.

2.5.3 HTTP/REST (HyperText Transfer Protocol / Representational State Transfer)

2.5.3.1 Introduction.

HTTP/REST is a widely used web protocol and architectural style for building scalable web services. It is based on standard HTTP methods and is known for its simplicity and ease of integration.

2.5.3.2 How to Use HTTP/REST.

Architecture:

- Server: The central system hosting RESTful APIs that manage resources and handle client requests.
- Client: Devices or applications that send HTTP requests to interact with resources on the server.

Communication Flow:

- Requests and Responses: Clients use standard HTTP methods (GET, POST, PUT, DELETE) to perform CRUD (Create, Read, Update, Delete) operations on resources.
- Endpoints: RESTful APIs expose endpoints (URLs) that represent resources.
- Data Format: Common data formats for communication include JSON (JavaScript Object Notation) and XML (eXtensible Markup Language).
- Security: HTTPS (HTTP Secure) is used to encrypt communication, ensuring data integrity and confidentiality.

ASM part 2.

3 Investigate architecture, frameworks, tools, hardware, and API techniques to develop IoT applications. P3

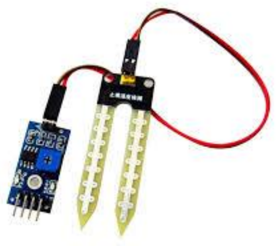
3.1 IoT Architecture.

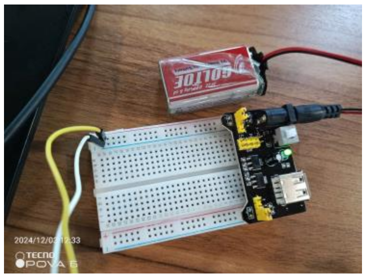
3.1.1 Perception Layer.

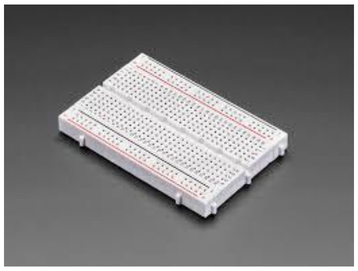
The perception layer is the layer that collects information and interacts directly with the physical environment. This is where sensors and output devices operate to measure and control physical factors such as humidity, temperature, and soil moisture.

The components in the Perception Layer are presented in the table below.

Table 1 The components in the Perception Layer.

Sensors and output devices	Task	Connection	Data collected/Function
DHT11 Sensor (Air Temperature and Humidity)	Measure the temperature (T) and humidity (H) of the air.	The DHT11 sensor is connected to GPIO D5 of ESP8266.	The sensor returns the temperature and humidity values, which will be displayed on the LCD and transmitted to the Blynk application.
Soil Moisture Sensor 	Measure soil moisture	Soil moisture sensor is connected to analog pin (A0) of ESP8266.	Soil moisture data will be converted into percentage value from analog value (0-1023) and sent to Blynk, displayed on LCD.
LEDs (Red, Yellow, Green)	Reflect the status of soil moisture.	The LEDs are connected to the GPIO pins D8, D6 and D7 of ESP8266 in red, green and yellow order respectively.	Red light: Lights up when soil moisture is higher than the high threshold (v3). Yellow light: Lights up when soil moisture is lower than the low threshold (v4). Green light: Lights up when soil moisture is at a normal level.
Relay and Pump	Control water pump based on soil moisture.	Relay is connected to GPIO D0 of ESP8266.	Auto mode: Pump turns on when soil moisture is lower than low threshold (v4), turns

			off when moisture is higher than high threshold (v3). Manual mode: Pump can be turned on/off manually via Blynk application (v9).
LCD_I2C Display 	Display parameters such as temperature, humidity and soil moisture.	LCD_I2C display is connected to ESP8266 via I2C interface (SDA and SCL) with SCL connected to D1, SDA connected to D2, VCC connected to vin (5V).	Display detailed information about T (temperature), H (DHT11 humidity) and M (soil moisture).
M102 Power Supply 	Provide stable 5V and 3.3V power to devices on the breadboard, helping to reduce the load on the ESP8266. Specifically for the relay and pump.	The M102 power supply will be mounted directly on the breadboard, taking power from the battery (DC). Voltage supply: 5V: the positive pole will be connected to the COM pin of the relay, the NC pin of the relay will be connected to the positive pole of the pump, the negative pole of the pump will be connected to the	Ensure stable voltage for devices in the IoT system, avoiding errors or loss of connection due to insufficient power. Support to reduce the load on the ESP8266 module, helping the module operate stably when connecting to Wi-Fi

		negative pole of the power source.	
Breadboard 	Connect and provide communication between devices in the perception system.	Breadboard plays the role of creating a simple, configurable, and reusable physical connection system.	As an intermediate physical component to connect and organize perception devices. Ensure that the M102 power source distributes stable voltage to each sensor, relay, and LED.

3.1.2 Network Layer.

The network layer is responsible for connecting sensors, devices, and applications over the Internet to transmit and receive data. In IoT, the network layer plays a critical role in handling data communication between the Perception Layer and the ApplicationLayer. This layer involves protocols for data transmission, routing, and ensuring connectivity. It can include technologies like MQTT, CoAP, HTTP, and other communication protocols. In addition, it manages data security, as IoT devices often transmit sensitive information. The ESP8266 will take on the role of this network layer, using Wi-Fi to connect to the Internet.

The network layer will include protocols for transmitting data from ESP8266 to Blynk and LCD_I2C display.

- Data will be transmitted from ESP8266 to Blynk application via Wi-Fi, allowing users to control and monitor the status of the system remotely.
- Data can also be displayed on the LCD_I2C display on site for users to monitor directly.
- Through HTTP or MQTT protocol, the system can transmit data to the cloud and can interact with external services and applications.

Components in the network layer:

- ESP8266:

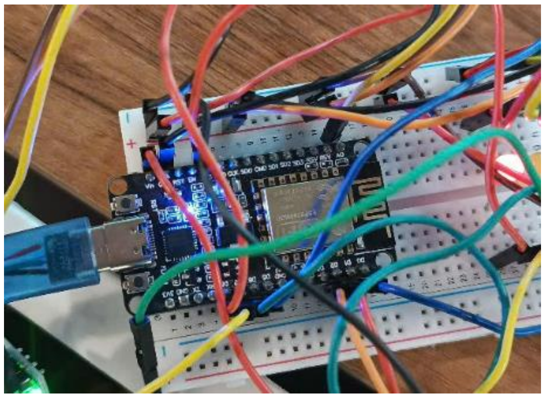


Figure 38 ESP8266

- Task: Connect the IoT system to the Wi-Fi network and transmit data to the Blynk application.
- Function:
    - Network connection: ESP8266 connects to the available Wi-Fi network to send data to the Blynk application.
    - Data transmission: Data from sensors (DHT11, soil moisture) is sent to the Blynk application via HTTP or MQTT protocol.
    - Device control: Commands from the Blynk application (such as turning on/off the pump) are sent to the ESP8266 and control the devices (LEDs, pumps) via GPIO pins.
    - Blynk: Mobile application allows remote monitoring and control. Blynk receives data from the ESP8266 and displays parameters such as temperature, humidity, soil moisture. It also allows users to change the mode from "auto" to "manual" and control the pump.

- Wi-Fi Network:
    - Task: Provide Internet connection to transmit data from ESP8266 to Blynk application and from Blynk to ESP8266.
    - Ensure stable connection: Ensure ESP8266 is always connected to Wi-Fi so that commands and data are transmitted promptly. Wi-Fi used must have Band: 2.4GHz.

3.1.3 Application Layer.

This is the layer where the user will interact directly with the system. The application layer includes the user interface and applications to manage and control the system remotely.

The Blynk application will allow users to:

- View temperature, humidity and soil moisture through gauges and labels:

    - Gauge v0 and v5: Display DHT11 temperature and humidity values as gauge bars.
    - Label v1: Display soil moisture values.
- Switch v2: Switch between Auto and Manual modes. The Auto/Manual mode determines the control of the water pump, allowing users to flexibly adjust the system according to their needs.
- Manually control the water pump (when in Manual mode) via switch v9.
- Set appropriate high and low moisture thresholds for plants via Slider v3 and Slider v4
- Receive notifications when the pump turns on/off or when the moisture reaches a high/low level via Label v6.
    - When the soil moisture is lower than the low threshold (v4), the pump turns on and displays "low moisture and watering".
    - When the soil moisture is higher than the high threshold (v3), the pump turns off and displays "high moisture".
    - When the soil moisture is at a normal level, the pump turns off and displays "normal moisture".

LCD_I2C display:

- Mission: Display parameters such as temperature (T), air humidity (H), and soil moisture (M).
- Function: Continuously update data collected from sensors and display on LCD_I2C display so that users can monitor directly on the spot.

3.2 IoT Frameworks.

3.2.1 Definition Blynk.

Blynk is an IoT platform designed to make developing and managing IoT applications quick and easy. It provides an intuitive and convenient interface for connecting smart devices to the Internet, allowing remote monitoring, control, and interaction via mobile devices or browsers.

3.2.2 Basic Features of Blynk

- User-friendly interface: Provides mobile applications (iOS and Android) with a drag-and-drop interface, helping users create dashboards to monitor and control IoT devices.
- Multi-platform support: Compatible with a variety of microcontrollers (ESP8266, ESP32, Arduino, Raspberry Pi, etc.).
- Diverse connectivity: Supports connection protocols such as Wi-Fi, Ethernet, GSM, LTE, and MQTT.

- Blynk Cloud: Provides free and stable cloud storage services for IoT devices.
- Extensibility: Supports private servers if users want to set up a self-managed system instead of using Blynk Cloud.

3.2.3 Functions

- Flexible dashboard: Users can add widgets such as buttons, sliders, graphs, and display real-time data.
- Remote Device Control: Support monitoring and controlling IoT devices via phone or computer from anywhere in the world.
- Real-Time Data Display: You can easily display real-time data from sensors and IoT devices on your mobile app.
- Instant Notifications: Send notifications via app or email when events occur (e.g., sensors exceed thresholds).
- Multi-Device Integration: Connect and manage multiple devices at the same time on a single account.
- Automation Tools: Support schedulers, event-based automation, and complex control scenarios.
- APIs and Webhooks: Provide RESTful APIs to interact with external systems and integrate third-party applications.
- IoT Application Development Support: Users can customize the interface or integrate Blynk into their own IoT applications.

3.2.4 Overview of Blynk in automatic watering system.

- Blynk App (Mobile/Web Application):
  - Is a user interface in the application layer in IoT architecture.
  - Allows monitoring of data from sensors (soil moisture, temperature, air humidity).
  - Provides the ability to control water pumps and switch between Auto/Manual modes.
- Blynk Cloud:
  - Is an intermediate network layer, ensuring data transmission between ESP8266 and the application.
  - Supports data storage, status synchronization and processing of commands from the application.
- Blynk Library:
  - Is a library installed on ESP8266 to communicate with Blynk Cloud.

    - Supports updating data from sensors and performing control commands (such as turning on/off the pump).
- User Interface on Blynk App: Necessary widgets on Blynk App include:
    - Gauge (V0, V5): Display temperature (V0) and air humidity (V5) from DHT11 sensor.
    - Label (V1): Display soil moisture value after converting to percentage.
    - Slider (V3, V4): set soil moisture threshold for plants.
    - Switch (V2): Switch between Auto and Manual mode.
    - Switch (V9): Manually control water pump when in Manual mode.
    - Label (V6): Display the status of irrigation system.
- Connect to Blynk Cloud
    - Connect to Wi-Fi network and sync data to Blynk Cloud.
    - Send data from sensors (DHT11 and soil moisture) to Cloud.
    - Receive control commands from the app via Blynk Cloud.

3.2.5 How to Use Blynk for Smart Plant Pot IoT Products.

Steps to Integrate Blynk into an IoT Product:

- Set up a Blynk account: Start by creating an account on the Blynk platform (using the Blynk cloud or setting up your own server).
- Create a mobile app: Use the Blynk app builder to design a custom mobile app interface by adding widgets and configuring their functionality.
- Write code: Develop firmware for IoT devices using libraries and code examples provided by Blynk. Connect your system's devices to the Blynk platform using authentication tokens.
- Connect IoT devices: Upload firmware to IoT devices and configure them to communicate with Blynk.
- Monitor and control: Once you have set up the mobile app and devices, you can now monitor and control your IoT devices remotely.

3.3 IoT Tools and Hardware.

3.3.1 IoT Tools.

3.3.1.1 Programming Tools.

Arduino IDE: is an integrated development environment used to program and upload code to Arduino boards. This is a powerful and easy-to-use tool, supporting programming in C/C++ language.

Figure 39 Arduino IDE

- Main components of Arduino IDE:
  - Source Code Area (Code Area):
    - Where to write program code, also known as sketch.
    - Program files have the extension .ino.
  - Tools:
    - Verify/Compile: Check, look up and compile code.
    - Upload: Upload code to the board.
    - New, Open, Save: Manage program files.
    - Serial Monitor: Send and receive data over Serial communication.
  - Message Area (Message Area): Displays errors, compile information and operating status.
  - Serial Monitor Window: Used to communicate with the board over Serial port.
- Role: Programming and downloading source code to ESP8266.
- Features:
  - Support Blynk, DHT, LCD I2C libraries.
  - Friendly and easy-to-use interface.

Supported libraries:

- Blynk Library: Integrate ESP8266 with Blynk App.
- DHT Library: Communicate with DHT11 sensor.
- Wire Library: Support I2C communication for LCD screen.
- LiquidCrystal_I2C: Control 16x2 or 20x4 LCD screen via I2C communication.

3.3.1.2 IoT Platform.

Blynk:

- Role: Remote control and monitoring platform via mobile/web application.
- Features:
    - Display sensor data (Gauge, Label).
    - Control water pump (Switch).
    - Toggle between Auto/Manual mode.

Serial Monitor (on Arduino IDE):

- Role: Monitor sensor data and system status during development and testing.
- Features: Display temperature, humidity, and soil moisture data.

3.3.2 Hardware.

As mentioned and listed in the IoT Architecture section, the hardware of the IoT system includes: ESP8266 NodeMCU, DHT11 sensor, soil moisture sensor, Relay Module (1 channel), LED lights (red, yellow, green), LCD_I2C screen, Water pump 5VDC, Power supply M102 , breadboard and Jumper wires.

3.4 IoT APIs: Blynk.

3.4.1 Blynk API Definition

Blynk API is a powerful toolkit provided by Blynk to connect, communicate, and control IoT devices remotely via a mobile app or web interface. Blynk API allows users to perform operations such as sending and receiving data between IoT devices and Blynk applications, such as reading sensors, sending control signals, and managing device status. This API empowers IoT enthusiasts, engineers, and developers to create custom IoT applications, web interfaces, and mobile apps that can seamlessly interact with and control IoT devices.

3.4.2 Key components of Blynk API.

Blynk HTTP API:

- Enables communication with IoT devices via HTTP commands.
- This API allows users to send HTTP GET or POST requests to Blynk Cloud, to read or write data to virtual pins on their devices.

Blynk WebSocket API:

- WebSocket API is a good choice when you want to transmit real-time data with low latency.
- With WebSocket, you can maintain a constant connection between your device and application without having to send continuous HTTP requests.
- WebSocket is useful for applications that need constantly updated information, such as monitoring sensors or controlling devices instantly.

Blynk MQTT API:

- MQTT is a lightweight and popular protocol for IoT applications, which helps transfer data between devices and servers efficiently.
- Blynk's MQTT API supports communication between IoT devices and applications over the cloud, allowing you to receive and send messages from virtual pins via MQTT topics.

3.4.3 Key features and functions

- Device control: With the Blynk API, you can remotely control hardware devices connected to the Blynk platform. This includes sending commands to IoT devices, turning switches on and off, and adjusting settings from your own apps.
- Data Interaction: The API supports data exchange, allowing you to send and receive data from IoT devices. You can get real-time sensor data, historical data, and more. User Management: The Blynk API supports user account management, allowing you to create, manage, and authenticate users in your IoT apps.
- Project Configuration: You can customize IoT projects, widgets, and user interfaces through the API. This includes configuring the look and behavior of mobile and web apps.
- Notifications: Blynk allows you to set up notifications for events or alerts from IoT devices, and you can trigger actions based on these notifications.
- Authentication and Access: To access the Blynk API, you need an authentication token associated with your Blynk project. This token acts as a security key that allows your applications to interact with your specific project. Authentication ensures that only authorized entities can communicate with your IoT device and data.
- Supported Platforms: The Blynk API is compatible with a wide range of hardware platforms, microcontrollers, and software environments. You can use it with popular devices such as Arduino, Raspberry Pi, ESP8266, etc.

3.4.4 IoT APIs: Blynk for smart plant pot system.

3.4.4.1 Blynk API Features.

Send Sensor Data:

- Send temperature, humidity, and soil moisture data from your device to the Blynk app.
- Display data via widgets like Gauge, Label.

Control Devices Remotely:

- Use the Switch button to turn the pump on/off or switch between Auto/Manual modes.
- Control based on actual sensor data.

Alerts and Notifications:

- Display notifications like: "Low moisture and watering", "High moisture", or "Normal moisture".
- Provide real-time feedback when environmental conditions change.

Process API from Blynk Server:

- Connect to Blynk Cloud Server or Private Server.
- Use HTTP or WebSocket API to access data or send commands.

3.4.4.2 Basic Blynk API for the System

Sending Values from ESP8266 to Blynk: **Blynk.virtualWrite(pin, value);**

Receive commands from the Blynk app: **BLYNK_WRITE(pin) { int value = param.asInt();}**

4 Discuss a specific problem to solve using IoT. P4

4.1 Problem Identification.

Figure 40 Smart automatic watering plant pot system.

Smart potted planters with automatic watering systems aim to solve problems related to plant care, especially in environments where soil moisture cannot be constantly monitored and adjusted. The main problems of smart potted planters with automatic watering systems revolve around accurately managing soil moisture, continuously and easily monitoring parameters, effectively controlling the watering system, and ensuring that users can easily monitor and adjust the system according to their requirements.

Some specific problems that need to be solved are:

- Manual plant care often encounters difficulties in accurately measuring the amount of water needed for plants, leading to water shortage or excess water. Lack of water will cause plants to wilt and not grow, while excess water can cause root rot and plant death.
- Users often have difficulty maintaining the correct watering regime at different times of the day. Many times plant caretakers forget to water or water too much without checking the soil moisture.
- Users may not regularly check the status of plants and soil moisture, resulting in a lack of information about the condition of the plants.
- Users may not immediately recognize the status of soil moisture without having to directly check the sensor or application.

To solve the above problems, a soil moisture sensor can be used to measure soil moisture. When the moisture drops below the low threshold, the system will automatically activate the pump to water the plants. When the moisture exceeds the high threshold, the system will stop watering. Parameters such as temperature, DHT11 humidity, soil moisture are displayed on the LCD_I2C screen, as well as through the Blynk application. This information will help users easily monitor the status of plants and decide to adjust when necessary. Notifications about soil moisture levels will also be sent via Blynk and displayed through alerts, helping users recognize the current soil condition. In addition, LEDs with different colors can be used to signal the status of soil moisture. The yellow LED will light up when the humidity is low, the red light will light up when the humidity is too high, and the green light will light up when the humidity is normal. This makes it easy for users to know the status of the plant without having to directly intervene.

4.2 Problem Context.

4.2.1 Reasons for choosing the product.

In the current context, plant care automation is becoming a popular need for individuals and businesses who have a hobby or require planting plants indoors or in the garden. This product helps reduce the effort of care, especially in maintaining stable soil moisture, avoiding plants from lacking or having too much water.

4.2.2 Describe features and benefits.

- Set the humidity threshold parameters suitable for the type of plant.
- Automatic mode: The system automatically waters the plants when the soil humidity is lower than the allowed threshold, helping the plants stay in good condition without the need for care.
- Manual mode: Users can manually intervene, turning the pump on/off as needed.
- Monitoring via Blynk and LCD_I2C: Provides an easy interface to monitor temperature, humidity and soil moisture right on the phone or LCD screen.
- Color LED: Helps users easily recognize the soil moisture status through different colored LEDs, thereby determining the condition of the plant.
- Display parameters and warnings: Provides warnings about soil moisture that is too high or too low, giving users timely information to adjust.

4.2.3 Product Impact.

- Energy saving: The system helps optimize the amount of water used for plants, minimizing resource consumption.

- Convenient: Automating plant care saves time and effort for growers, while minimizing the risk of forgetting to water plants.
- Increasing crop productivity and health: Plants will always have enough moisture needed to grow well, thereby increasing crop productivity and health.
- Using in urban areas: The system can be applied in areas with water shortages or where there is limited space for planting trees, creating favorable conditions for planting trees in homes, offices or gardens.

5 Employ an appropriate set of tools to develop a plan into an IoT application. P5

5.1 Hardware Configuration.

5.1.1 Circuit diagram of smart automatic watering plant pot system.

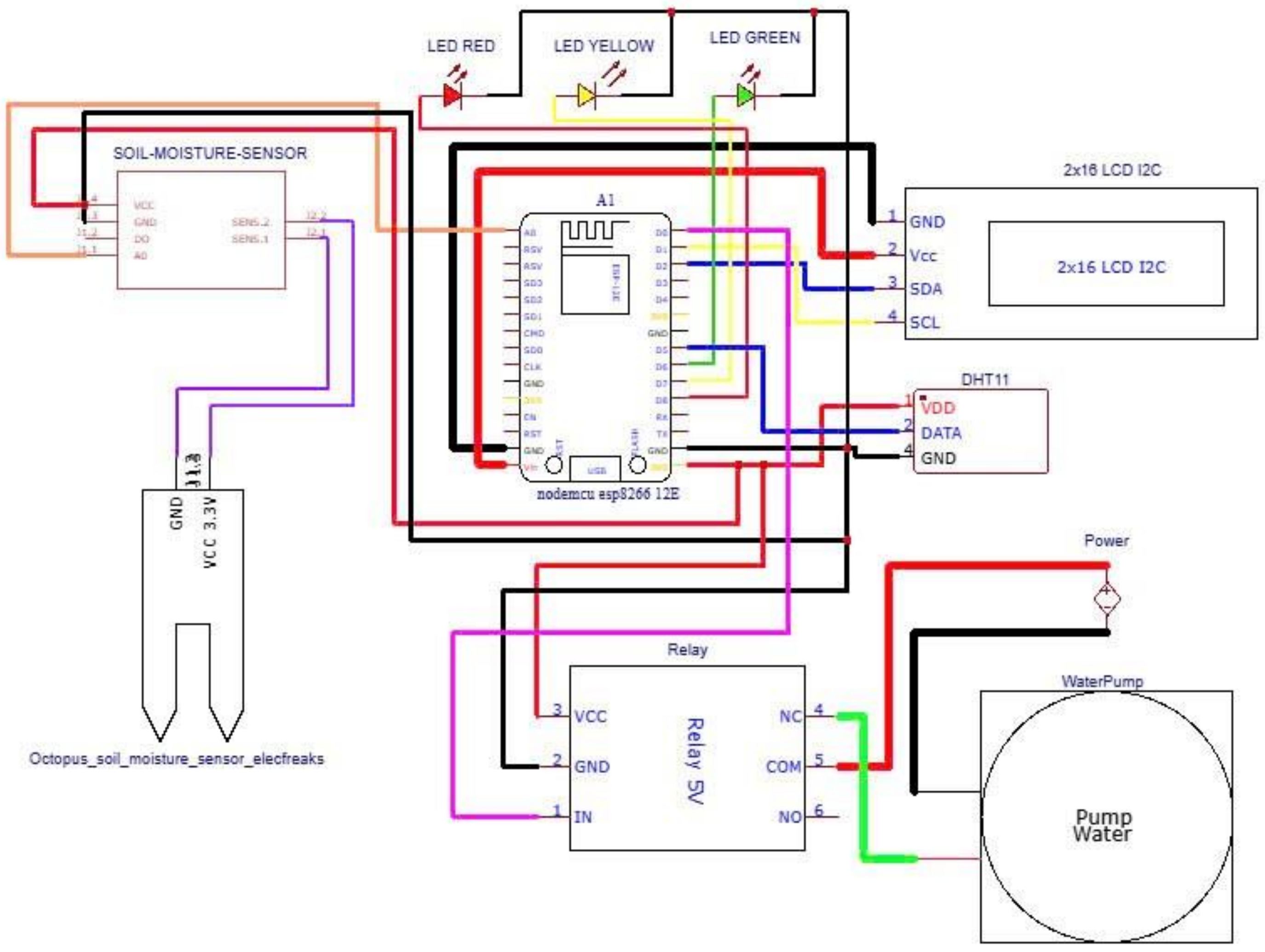


Figure 41 Circuit diagram of smart automatic watering plant pot system.

5.1.2 Functions and processing code for smart self-watering plant pot system.

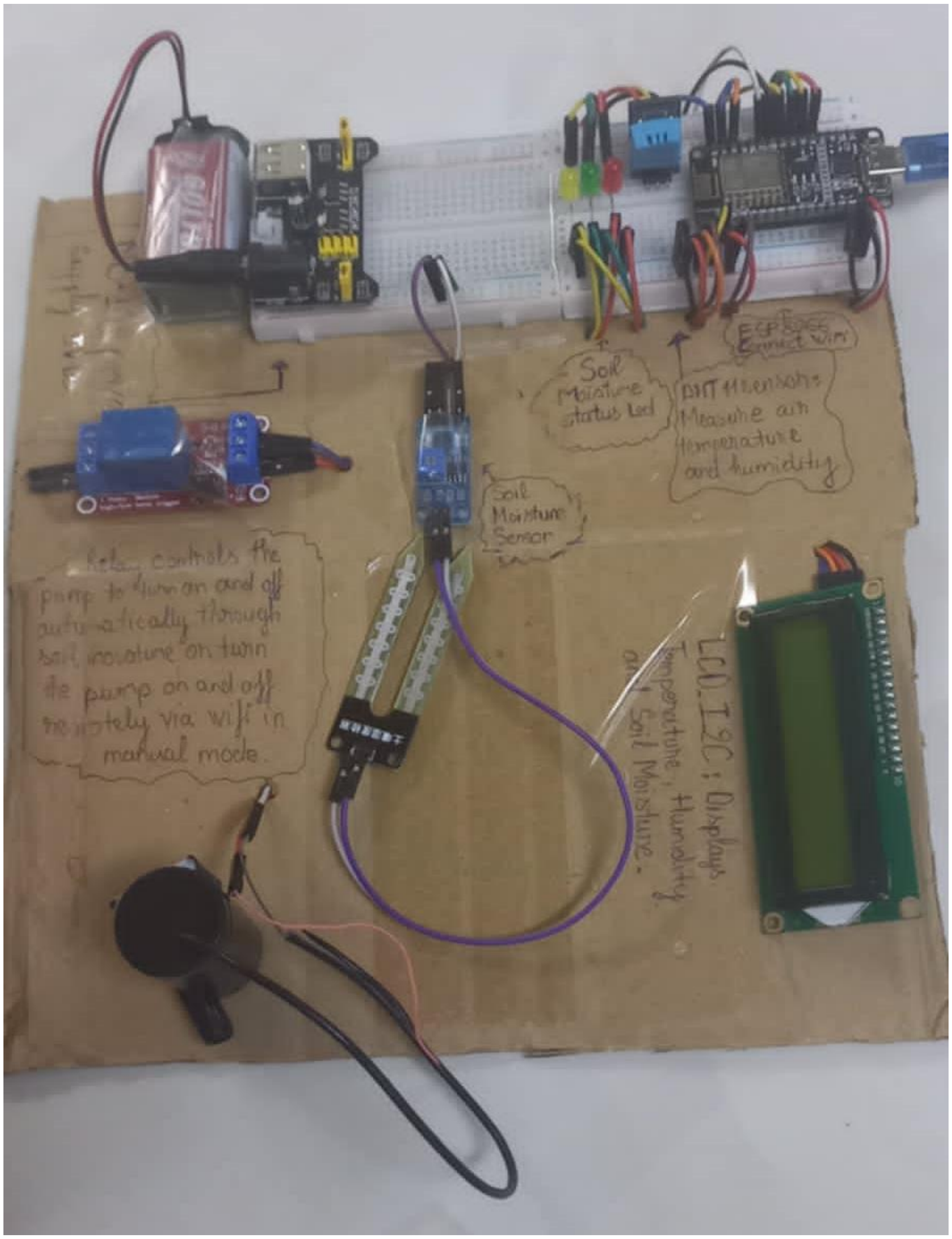

ESP8266 Connect wifi
Soil Moisture status Led
DHT 11 sensors Measure air temperature and humidity
Soil Moisture Sensor
Relay controls the pump to turn on and off automatically through soil moisture or turn the pump on and off remotely via wifi in manual mode.
LCD I2C: Displays Temperature, Humidity and Soil Moisture.

Figure 42 Smart plant pot model with automatic watering.

Link YouTube: https://youtu.be/lpUDix4hNjs?si=hBLoIBu5IkGvps0c

5.1.2.1 Set the moisture threshold limit for plants.

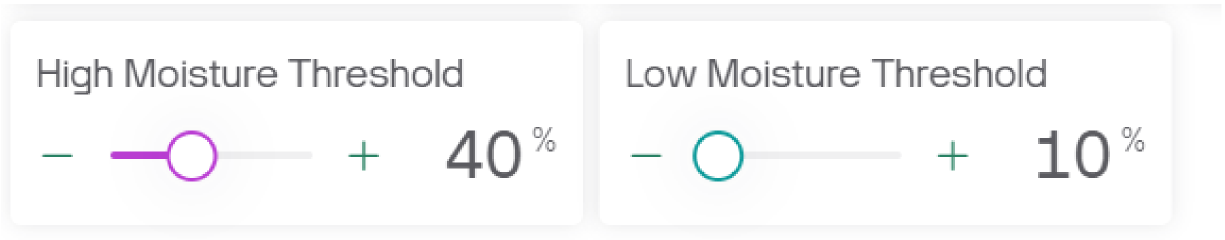
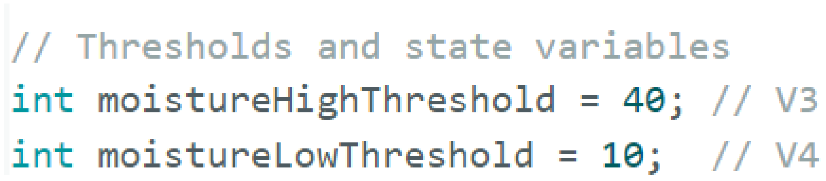
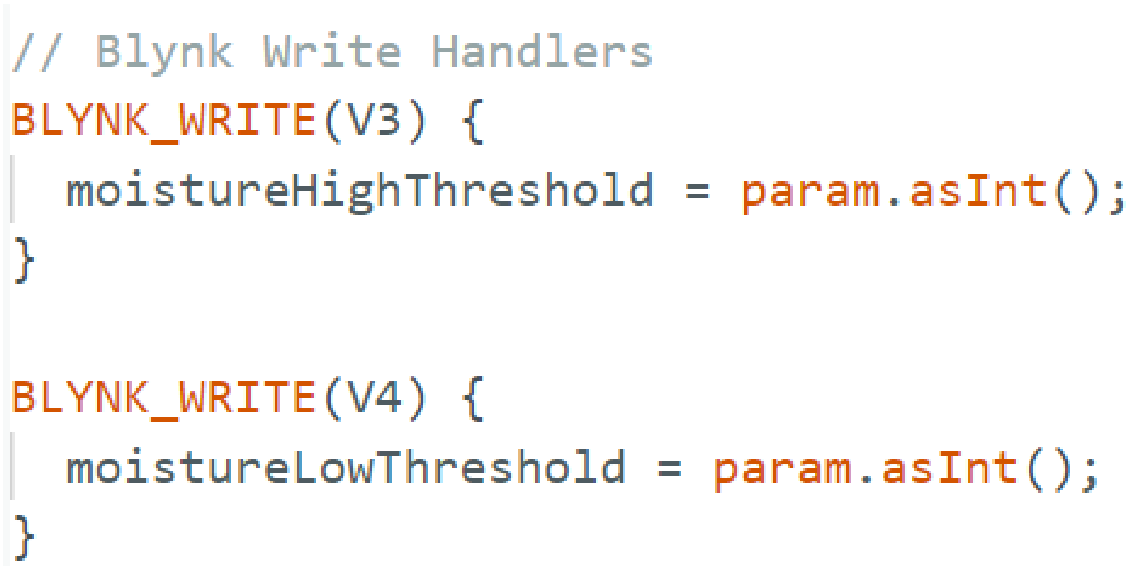


Figure 43 Set the moisture threshold limit for plants 1.



```
// Thresholds and state variables
int moistureHighThreshold = 40; // V3
int moistureLowThreshold = 10;  // V4
```

Figure 44 Set the moisture threshold limit for plants 2.



```
// Blynk Write Handlers
BLYNK_WRITE(V3) {
  moistureHighThreshold = param.asInt();
}

BLYNK_WRITE(V4) {
  moistureLowThreshold = param.asInt();
}
```

Figure 45 Set the moisture threshold limit for plants 3.

The two variables moistureHighThreshold and moistureLowThreshold are used to define the “ideal” moisture range for the soil.

The BLYNK_WRITE functions allow the user to change this threshold value from the Blynk application without reloading the code to the ESP8266.

These thresholds affect how the water pump is controlled and the state of the system (LEDs, notifications).

5.1.2.2 Display parameters of temperature, air humidity and soil moisture on LCD_I2C.

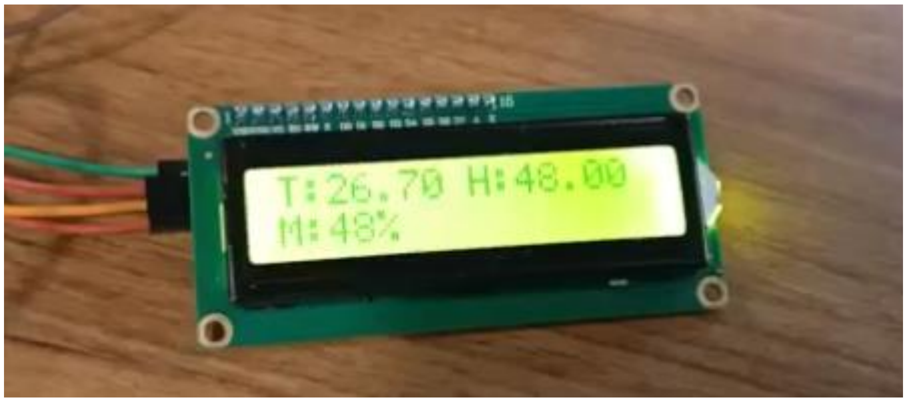
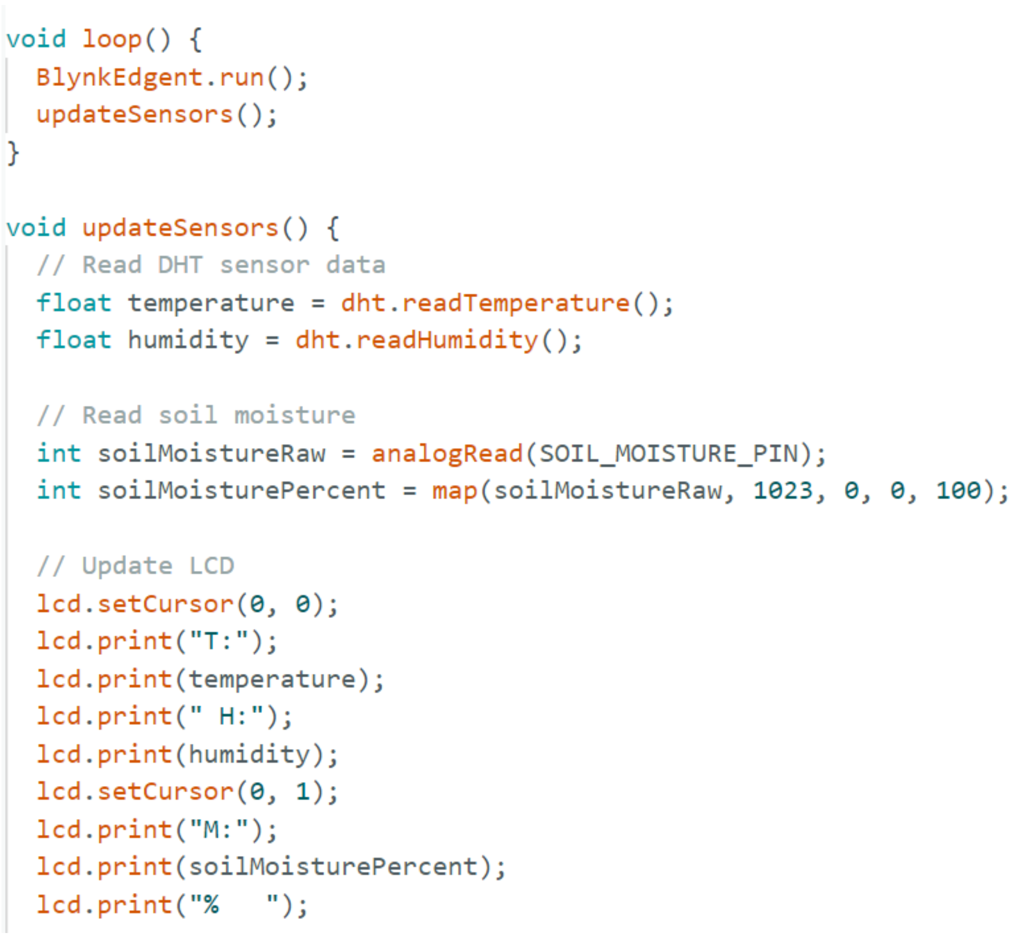


Figure 46 Display parameters of temperature, air humidity and soil moisture on LCD_I2C.



```
void loop() {
  BlynkEdgent.run();
  updateSensors();
}

void updateSensors() {
  // Read DHT sensor data
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Read soil moisture
  int soilMoistureRaw = analogRead(SOIL_MOISTURE_PIN);
  int soilMoisturePercent = map(soilMoistureRaw, 1023, 0, 0, 100);

  // Update LCD
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(temperature);
  lcd.print(" H:");
  lcd.print(humidity);
  lcd.setCursor(0, 1);
  lcd.print("M:");
  lcd.print(soilMoisturePercent);
  lcd.print("%   ");
```

Figure 47 Display parameters of temperature, air humidity and soil moisture on LCD_I2C.

updateSensors() function:

- Reads data from temperature, humidity (DHT11) and soil moisture sensors.
- Displays this information on the LCD screen.

The loop() function ensures these tasks are executed periodically every 2 seconds via timer.run(), maintaining connection and synchronization with Blynk.

5.1.2.3 Automatically water through soil moisture and display moisture status on Blynk interface through Label V6.

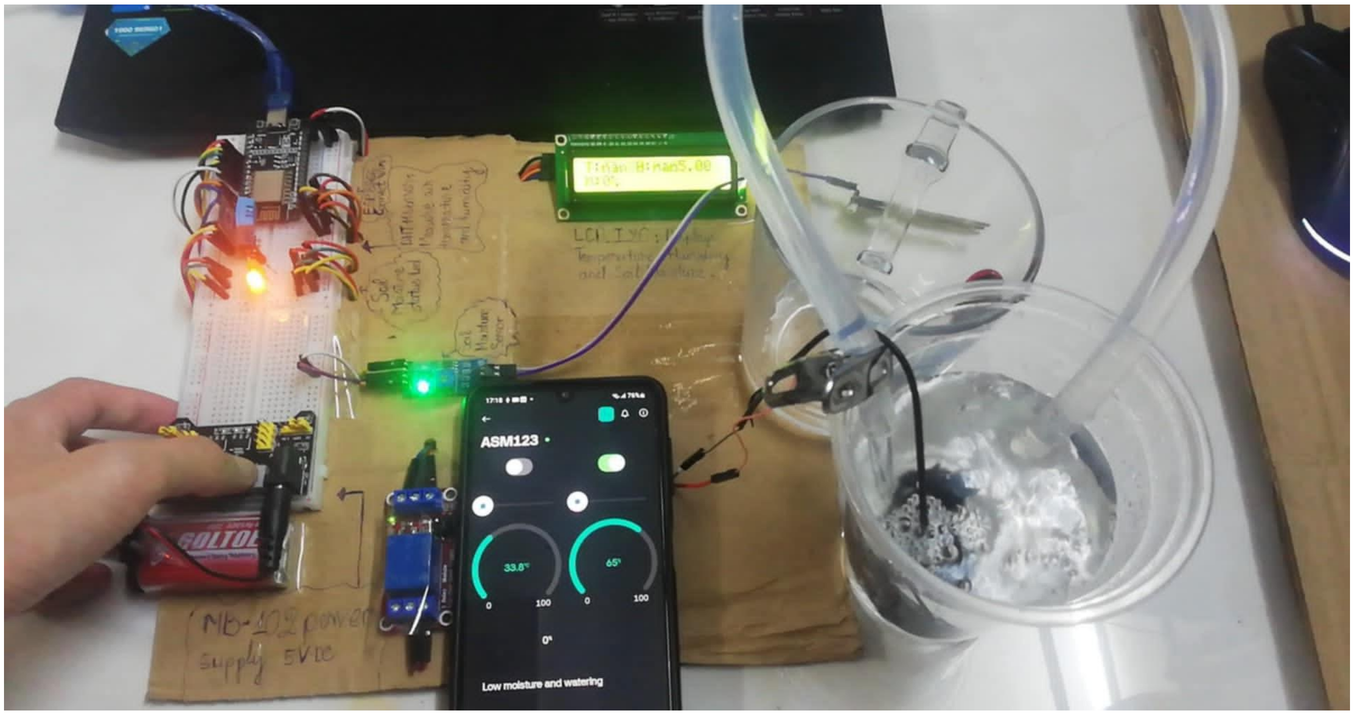


Figure 48 Automatically water through soil moisture and display moisture status on Blynk interface through Label V6.

Low soil moisture, Moisture Status: "Low moisture and watering" message is displayed, and the pump is running.

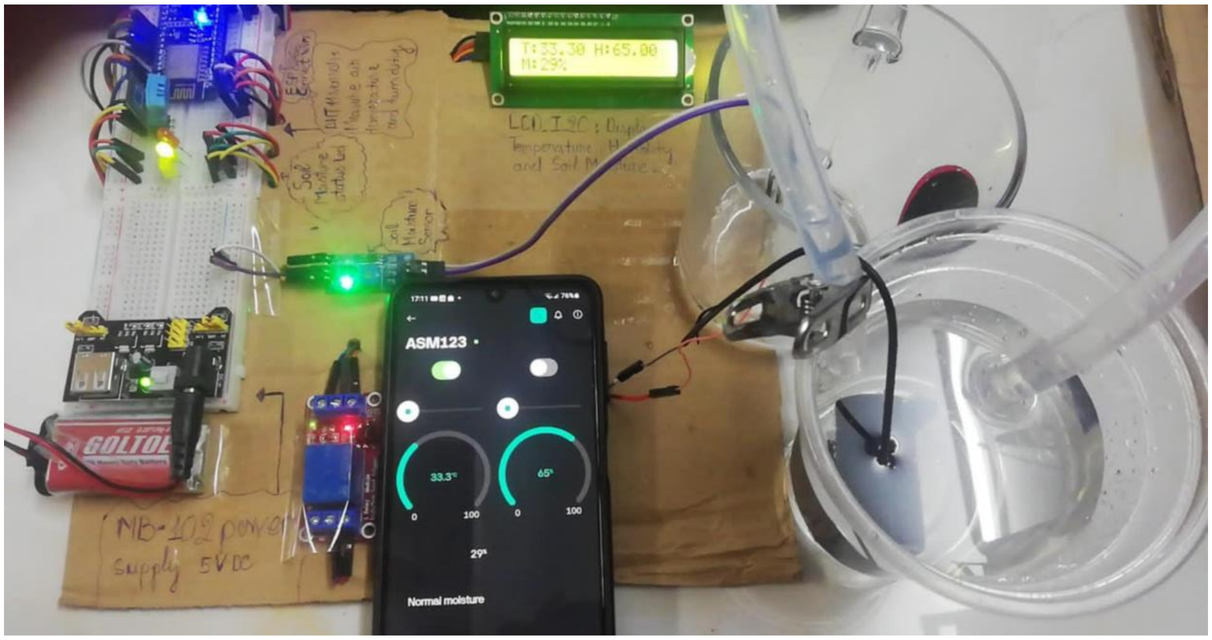


Figure 49 Automatically water through soil moisture and display moisture status on the Blynk interface through Label V6.

Soil moisture is 29%, within the threshold. Moisture Status will display the status line: "Normal moisture", the pump is not running.

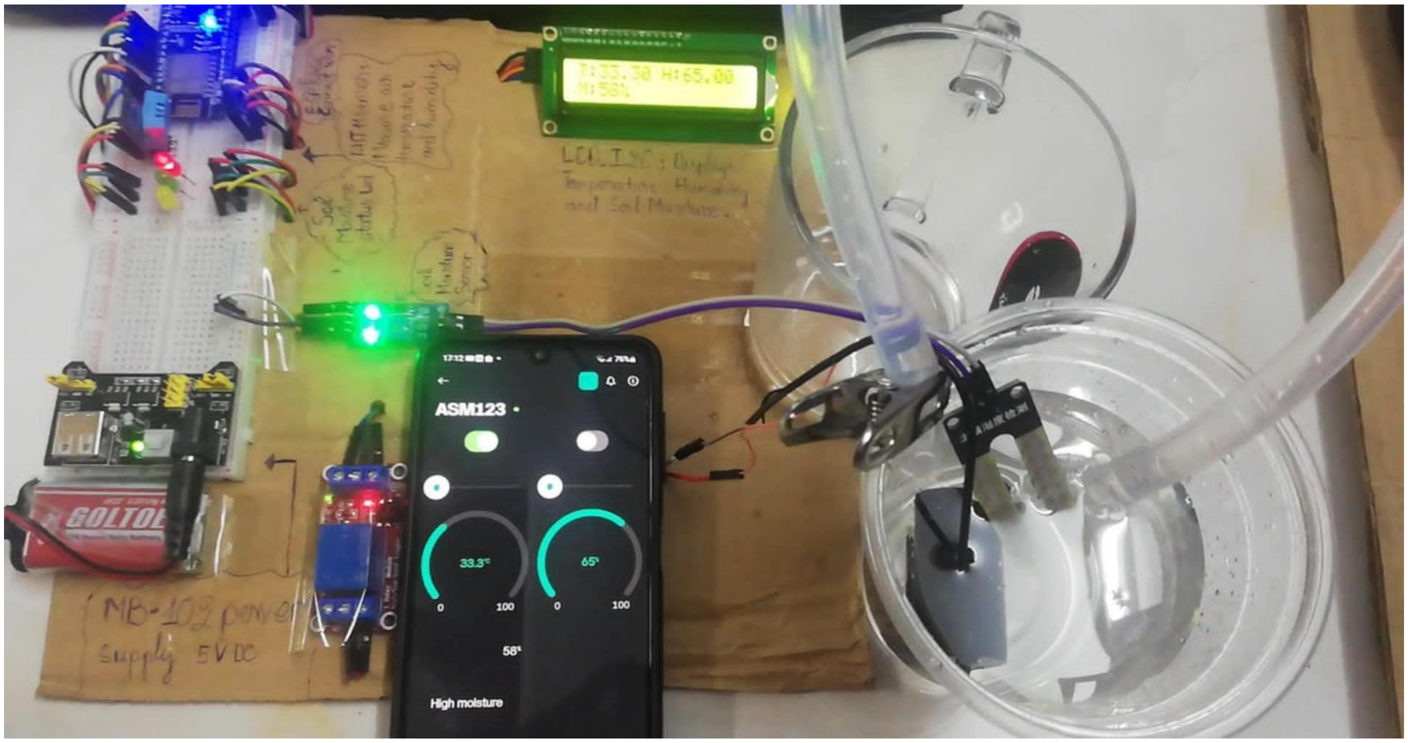
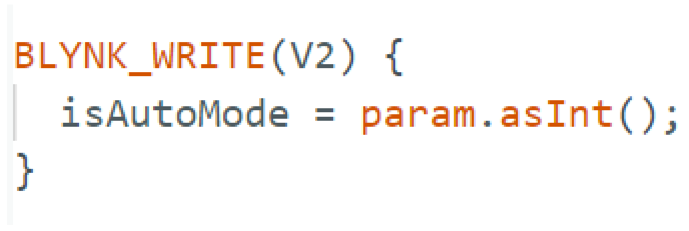


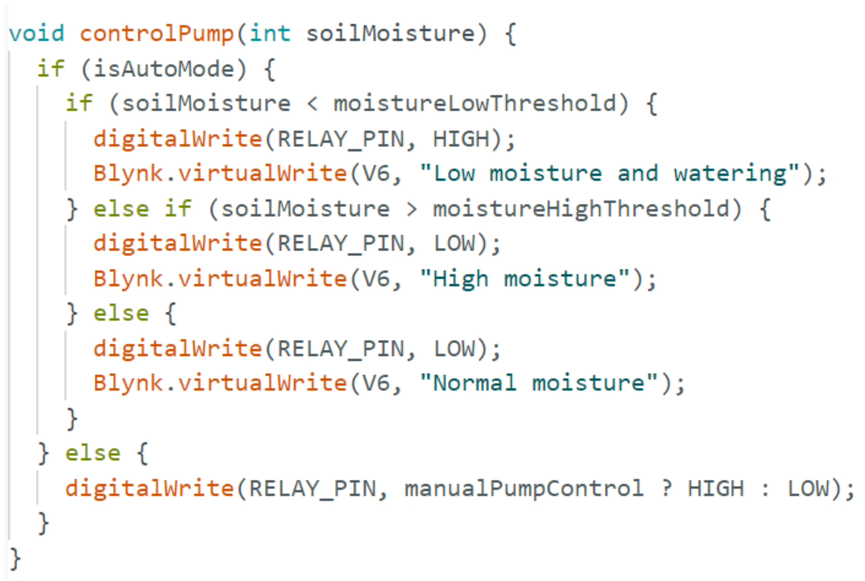
Figure 50 Automatically water through soil moisture and display moisture status on Blynk interface through Label V6.

Soil moisture is 58%, exceeding the threshold. Moisture Status will display the status line: "High moisture", the pump is not running.



```
BLYNK_WRITE(V2) {
  isAutoMode = param.asInt();
}
```

Figure 51 Automatically water through soil moisture and display moisture status on Blynk interface through Label V6.



```
void controlPump(int soilMoisture) {
  if (isAutoMode) {
    if (soilMoisture < moistureLowThreshold) {
      digitalWrite(RELAY_PIN, HIGH);
      Blynk.virtualWrite(V6, "Low moisture and watering");
    } else if (soilMoisture > moistureHighThreshold) {
      digitalWrite(RELAY_PIN, LOW);
      Blynk.virtualWrite(V6, "High moisture");
    } else {
      digitalWrite(RELAY_PIN, LOW);
      Blynk.virtualWrite(V6, "Normal moisture");
    }
  } else {
    digitalWrite(RELAY_PIN, manualPumpControl ? HIGH : LOW);
  }
}
```

Figure 52 Automatically water through soil moisture and display moisture status on Blynk interface through Label V6.

command: if (isAutoMode)

- Checks whether the system is in automatic mode.
- The variable isAutoMode is adjusted via Virtual Pin V2 in the Blynk application.

In automatic mode, the pump is controlled based on the soil moisture value compared to two thresholds:

- moistureLowThreshold: Low threshold.
- moistureHighThreshold: High threshold.
    - If the soil moisture is lower than the low threshold:
        - Enable the pump: digitalWrite(RELAY_PIN, HIGH).
        - Send a notification to the Blynk application via Virtual Pin V6: "Low moisture and watering".
    - If the soil moisture is higher than the high threshold:

        - Turn the pump off: digitalWrite(RELAY_PIN, LOW).
        - Send a notification to the Blynk application: "High moisture"
    - If the soil moisture is between the high and low thresholds:
        - Turn the pump off: digitalWrite(RELAY_PIN, LOW).
        - Send a notification: "Normal moisture".

5.1.2.4 Soil moisture status through LED lights.

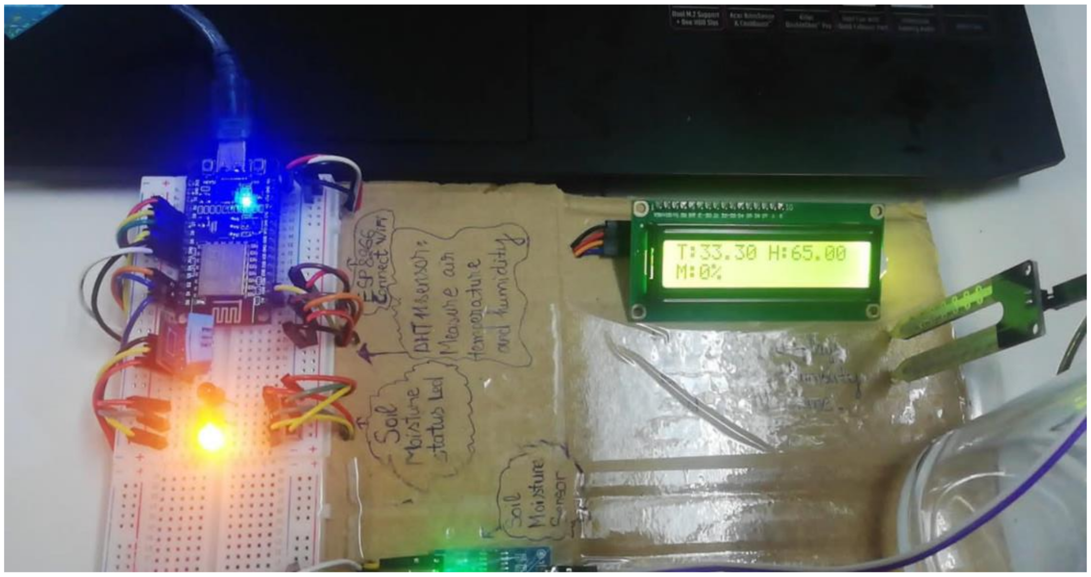


Figure 53 Soil moisture status through LED lights.

Low soil moisture, yellow LED on.

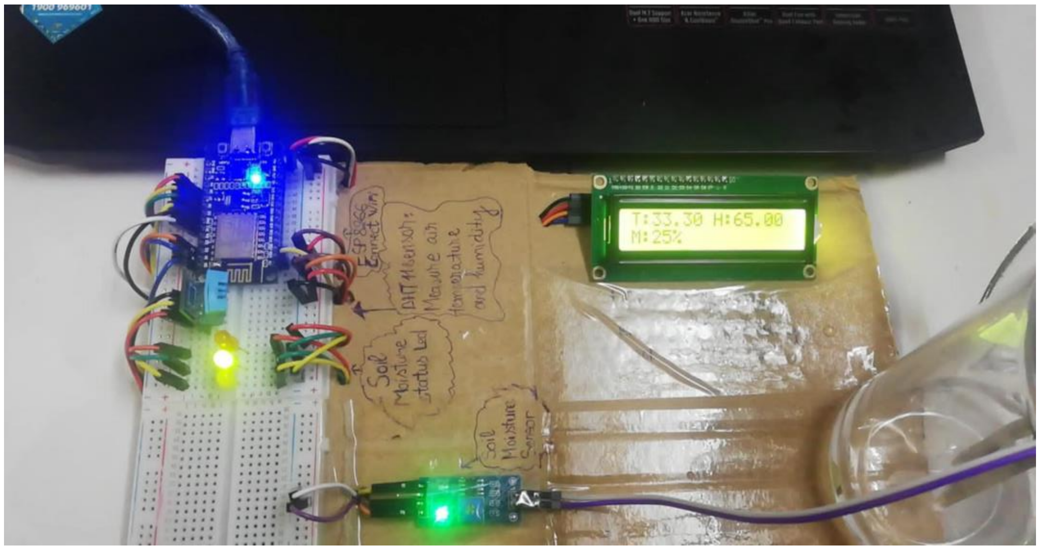


Figure 54 Soil moisture status through LED lights.

Soil moisture is normal within the threshold, and the green LED is on.

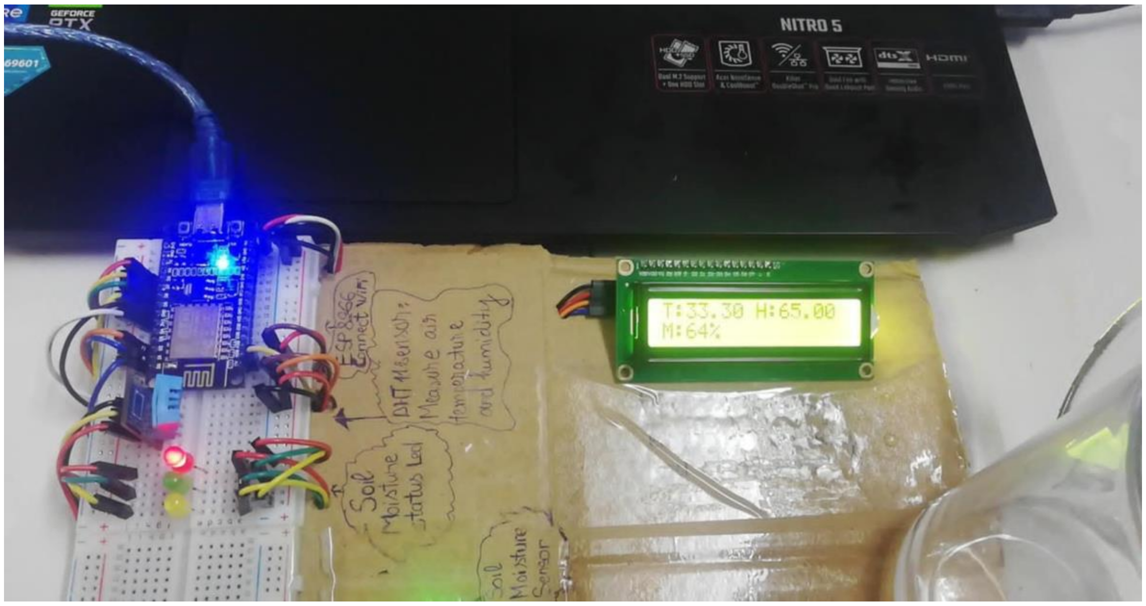
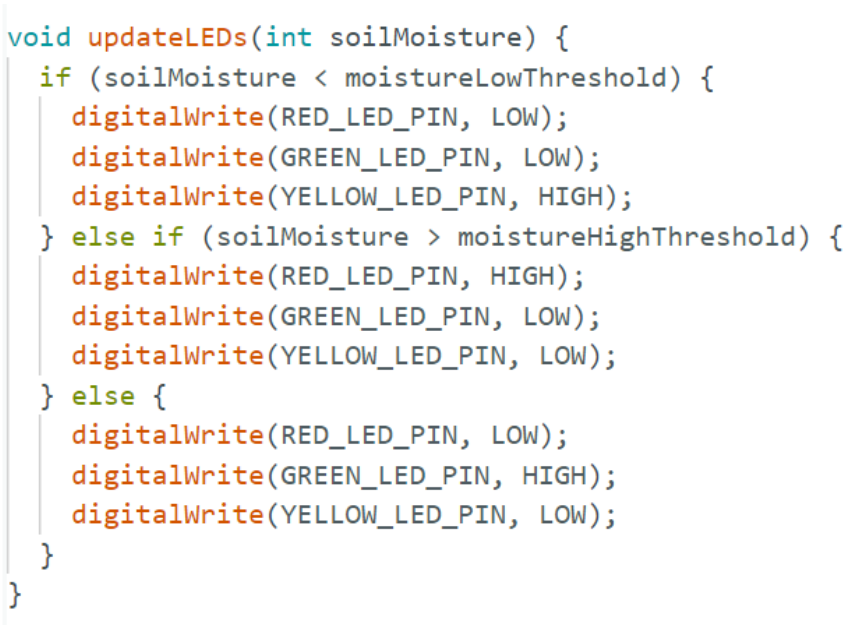


Figure 55 Soil moisture status through LED lights.

Soil moisture is above threshold, and the red LED is on.



```
void updateLEDs(int soilMoisture) {
  if (soilMoisture < moistureLowThreshold) {
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(GREEN_LED_PIN, LOW);
    digitalWrite(YELLOW_LED_PIN, HIGH);
  } else if (soilMoisture > moistureHighThreshold) {
    digitalWrite(RED_LED_PIN, HIGH);
    digitalWrite(GREEN_LED_PIN, LOW);
    digitalWrite(YELLOW_LED_PIN, LOW);
  } else {
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(GREEN_LED_PIN, HIGH);
    digitalWrite(YELLOW_LED_PIN, LOW);
  }
}
```

Figure 56 Soil moisture status through LED lights.

Displays soil moisture status by turning on/off the LEDs:

- Yellow light: Soil is dry (moisture is lower than the low threshold).

- Red light: Soil is too wet (moisture is higher than the high threshold).
- Green light: Normal moisture (between the two thresholds).

Check if soil moisture is below low threshold

- Yellow light on: Indicates soil is too dry. Meaning: Need to pump water.
- Other lights (red, green) are off.

Check if soil moisture is higher than high threshold (moistureHighThreshold)

- Red light on: Indicates soil is too moist.
- Other lights (yellow, green) are off.

If soil moisture is within the normal threshold

- Green light on: Indicates soil moisture is good. Meaning: The system is operating normally, no adjustment is required.
- Other lights (yellow, red) are off.

5.1.2.5 Manual watering.

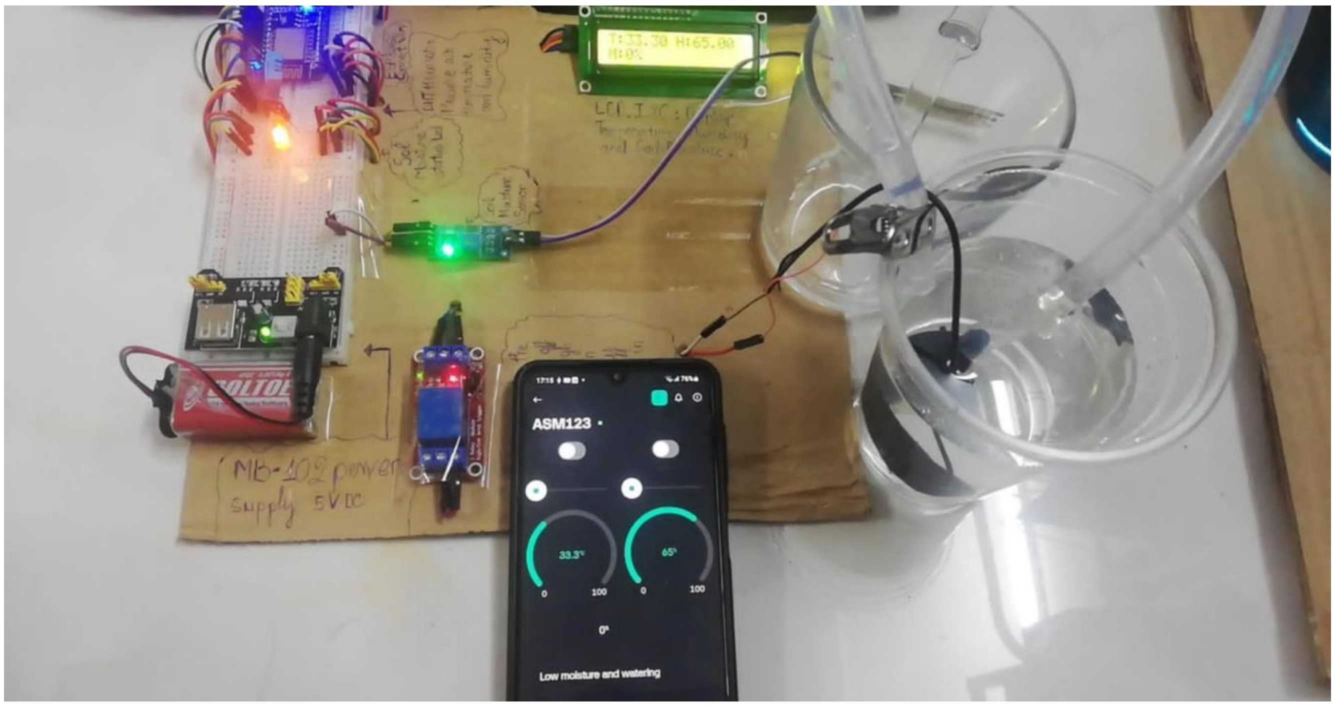


Figure 57 Manual watering.

In Manual mode, when the Pump Manual button is off, the pump will not operate even if the soil moisture is 0%.

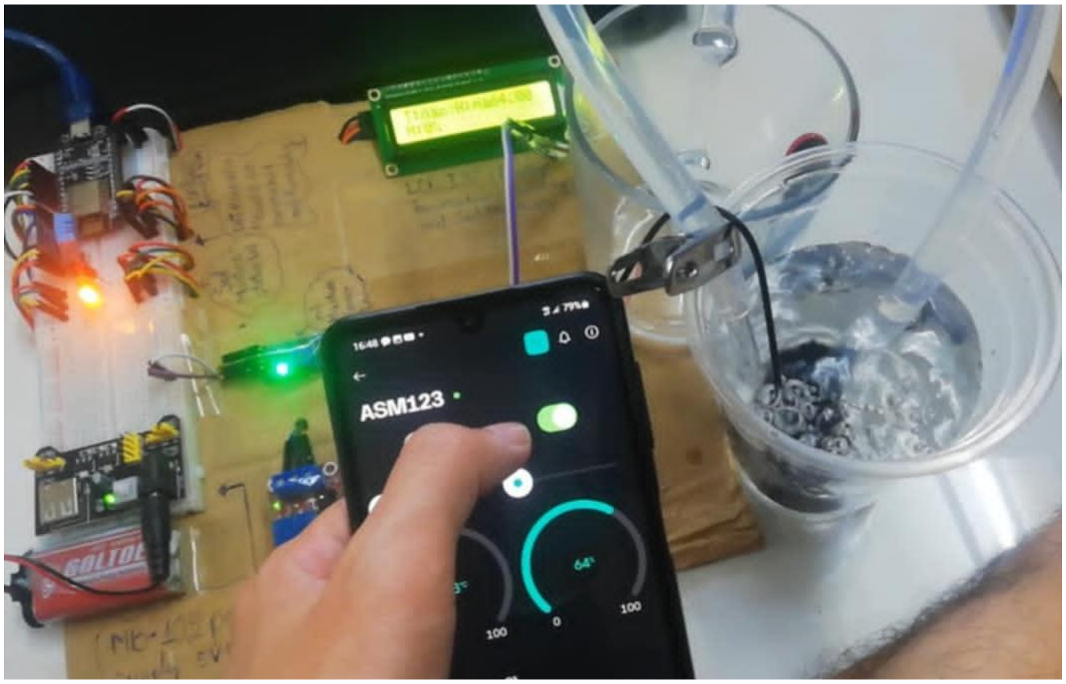
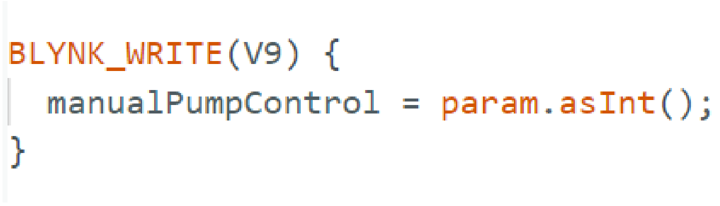


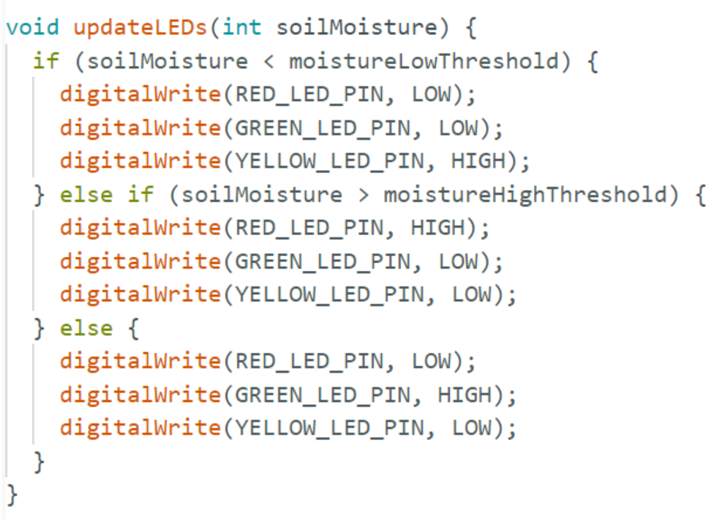
Figure 58 Manual watering.

In Manual mode, when the Pump Manual button is on, the pump will operate.



```
BLYNK_WRITE(V9) {
  manualPumpControl = param.asInt();
}
```

Figure 59 Manual watering.



```
void updateLEDs(int soilMoisture) {
  if (soilMoisture < moistureLowThreshold) {
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(GREEN_LED_PIN, LOW);
    digitalWrite(YELLOW_LED_PIN, HIGH);
  } else if (soilMoisture > moistureHighThreshold) {
    digitalWrite(RED_LED_PIN, HIGH);
    digitalWrite(GREEN_LED_PIN, LOW);
    digitalWrite(YELLOW_LED_PIN, LOW);
  } else {
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(GREEN_LED_PIN, HIGH);
    digitalWrite(YELLOW_LED_PIN, LOW);
  }
}
```

Figure 60 Manual watering.

Manual Mode

when: if (isAutoMode) = False, the system will switch to this command:

else {

digitalWrite(RELAY_PIN, manualPumpControl ? HIGH : LOW);

}

In this mode, the pump is completely controlled by the user via Virtual Pin V9.

manualPumpControl is a boolean variable, representing the state of the pump:

- true (pump on): digitalWrite(RELAY_PIN, HIGH).
- false (pump off): digitalWrite(RELAY_PIN, LOW).

5.2 Software Development.

5.2.1 Blynk Configuration.

5.2.1.1 Register an account and set up the Blynk IOT Template.

Step 1. Visit https://blynk.cloud/ and select Create new account.

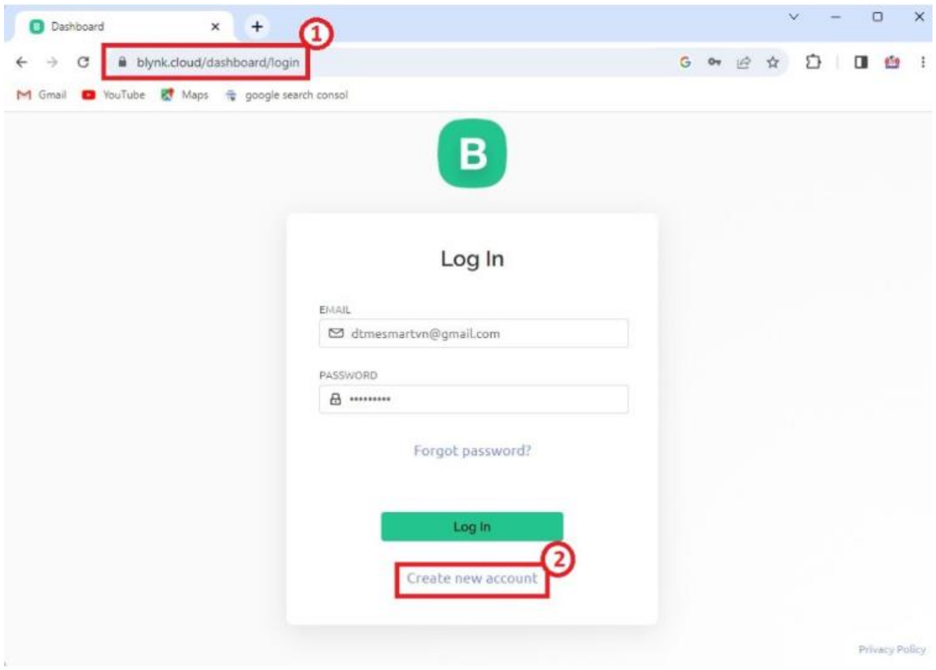
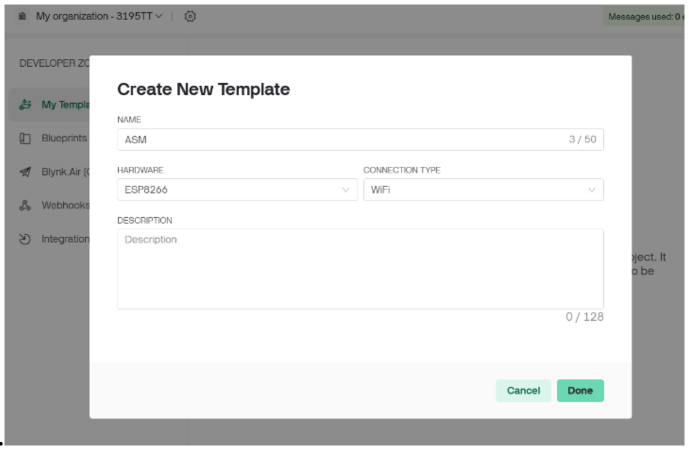


Figure 61 Create new account.

After completing the profile, you will be redirected to the Templates page to create device templates for your IoT projects.

Step 2. Create a new template and enter parameters such as name, version, device type, and connection.



Create New Template

NAME: ASM (3 / 50)

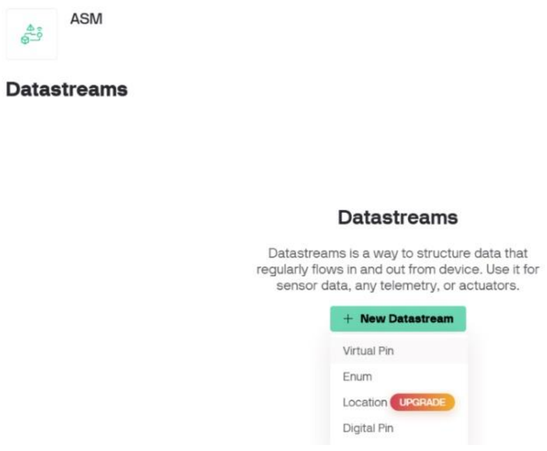
HARDWARE: ESP8266

CONNECTION TYPE: WiFi

DESCRIPTION: Description (0 / 128)

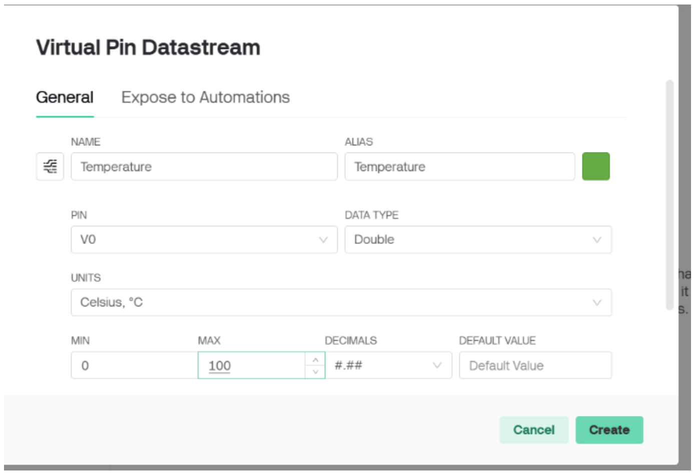
Figure 62 Create a new template.

Step 3. Switch to the Datastream tab to create virtual Pin variables to assign to the control panel and program code.



Datastreams is a way to structure data that regularly flows in and out from device. Use it for sensor data, any telemetry, or actuators.

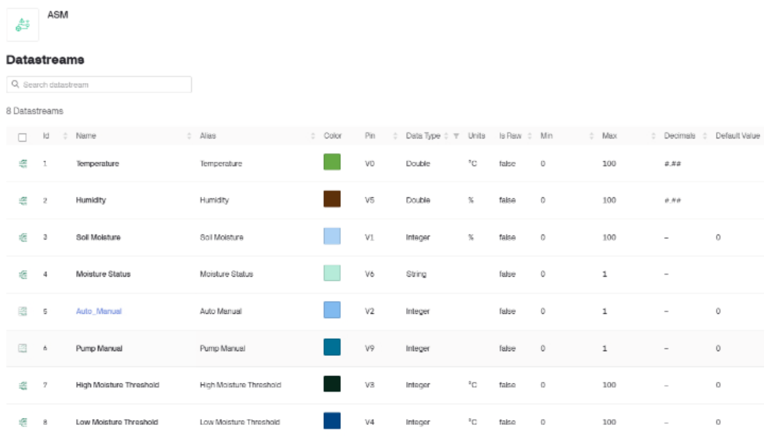
Figure 63 Create virtual Pin variables in Datastream.



Virtual Pin Datastream

NAME	Temperature
ALIAS	Temperature
PIN	V0
DATA TYPE	Double
UNITS	Celsius, °C
MIN	0
MAX	100
DECIMALS	#.##
DEFAULT VALUE	Default Value

Figure 64 Create virtual Pin variables in Datastream.



ASM

Datastreams

8 Datastreams

Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Max	Decimals	Default Value
1	Temperature	Temperature	🟩	V0	Double	°C	false	0	100	#.##	
2	Humidity	Humidity	🟫	V5	Double	%	false	0	100	#.##	
3	Soil Moisture	Soil Moisture	🟦	V1	Integer	%	false	0	100	–	0
4	Moisture Status	Moisture Status	🟩	V6	String		false	0	1	–	
5	Auto_Manual	Auto Manual	🟦	V2	Integer		false	0	1	–	0
6	Pump Manual	Pump Manual	🟦	V9	Integer		false	0	1	–	0
7	High Moisture Threshold	High Moisture Threshold	⬛	V3	Integer	°C	false	0	100	–	0
8	Low Moisture Threshold	Low Moisture Threshold	🟦	V4	Integer	°C	false	0	100	–	0

Figure 65 Create virtual Pin variables in Datastream.

Step 4. Switch to the Web dashboard tab to create a dashboard, displayed on the blynk.cloud web. Drag and drop the required objects into the widget box on the Dashboard screen.

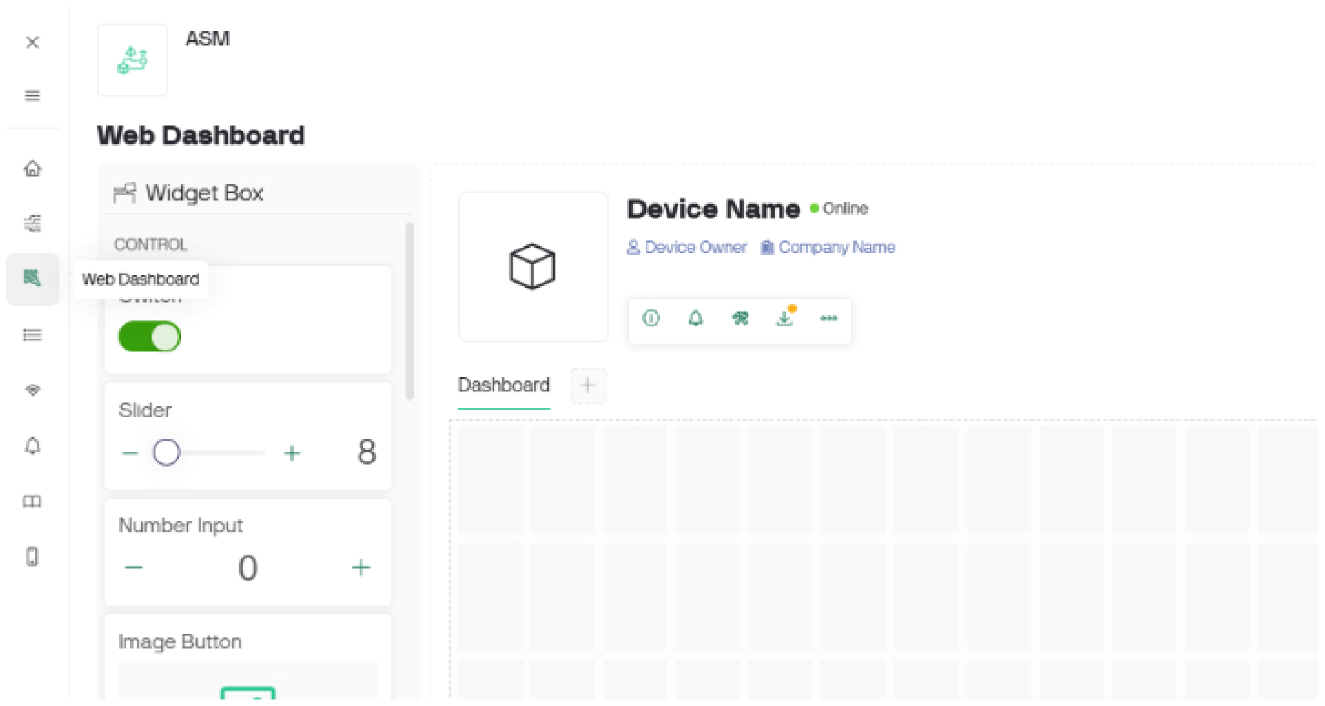


Figure 66 Create a dashboard.

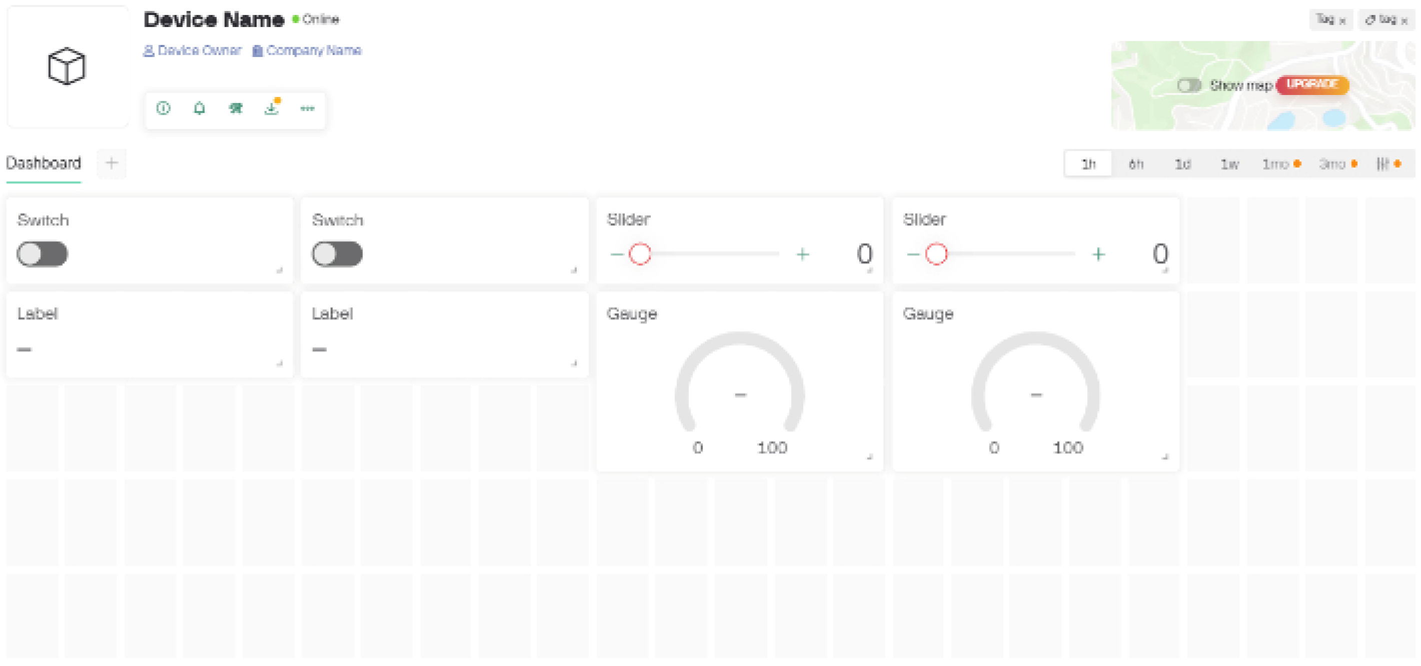


Figure 67 Create a dashboard.

Step 5. Attach the appropriate virtual pin in the datastream to the objects in the web dashboard.

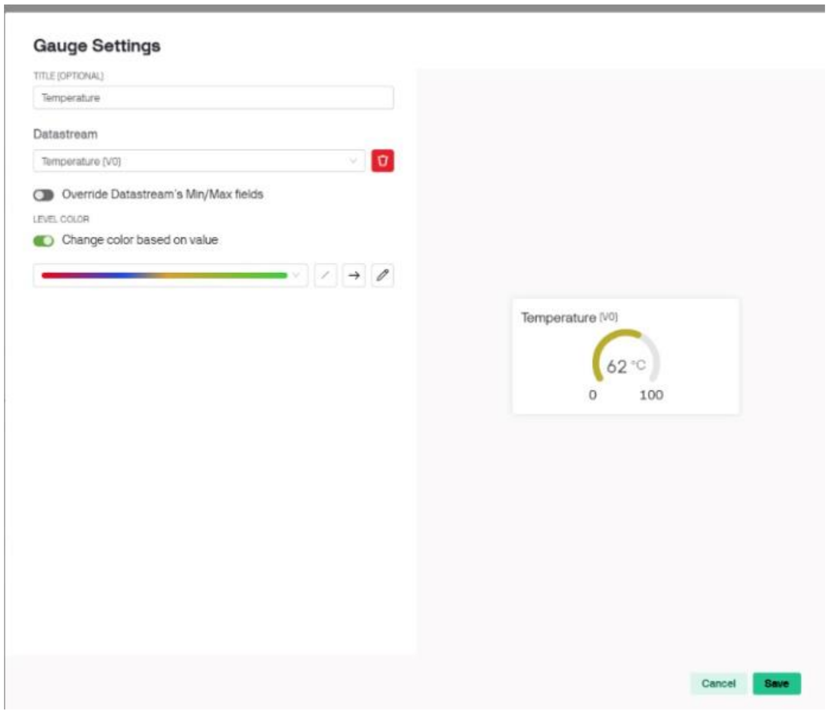


Figure 68 Attach the virtual pin to the objects in the web dashboard.

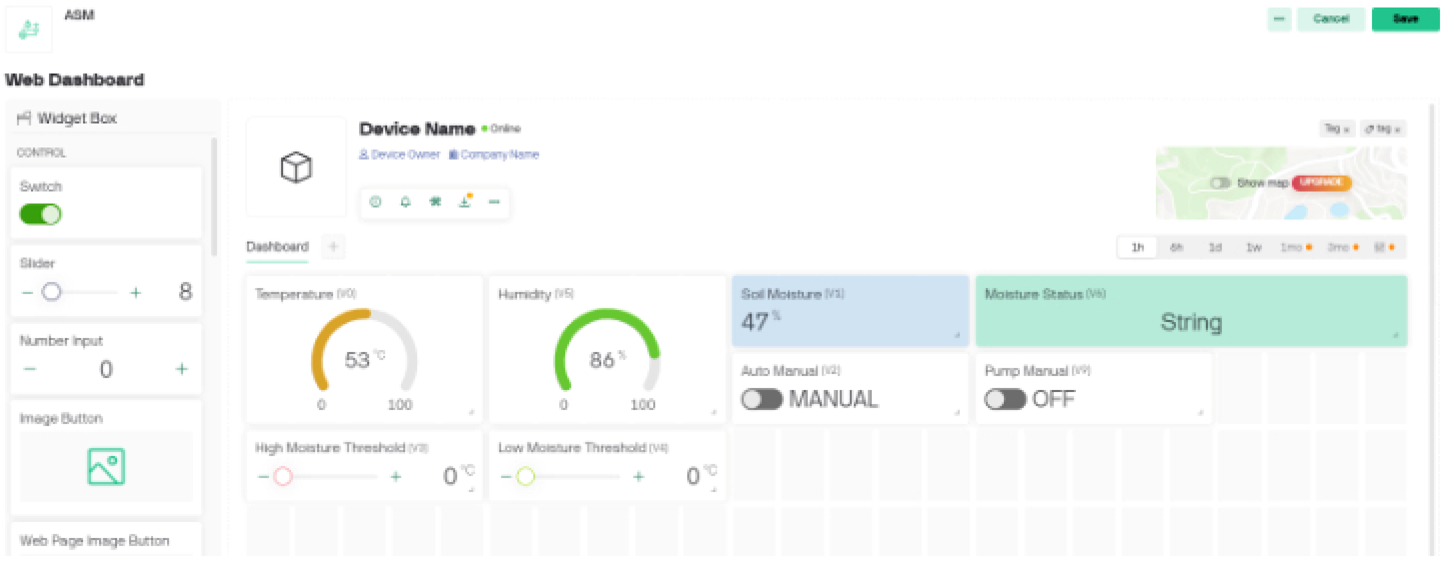


Figure 69 Attach the virtual pin to the objects in the web dashboard.

After creating the Device Templates, you will receive BLYNK_TEMPLATE_ID and BLYNK_DEVICE_NAME at Home tab, two important parameters to connect Blynk IOT and nodemcu esp8266.

5.2.1.2 Load program for nodemcu esp8266 kit.

Step 1. Open the Arduino IDE and select Sketch > Include Library > Manage Libraries. Here download the necessary libraries such as Blynk, DHT, Wire, LiquidCrystal_I2C, ...

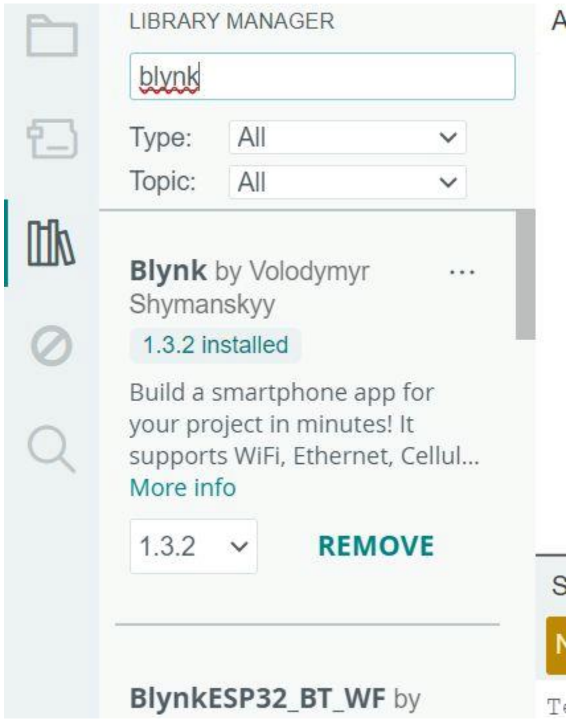

LIBRARY MANAGER
blynk
Type: All
Topic: All
Blynk by Volodymyr Shymanskyy
1.3.2 installed
Build a smartphone app for your project in minutes! It supports WiFi, Ethernet, Cellul...
More info
1.3.2
REMOVE
BlynkESP32_BT_WF by

Figure 70 Install the necessary libraries.

Step 2. Open the Blynk sample code: select File > Examples > Blynk > Blynk.Edgent. Choose Edget_ESP8266.

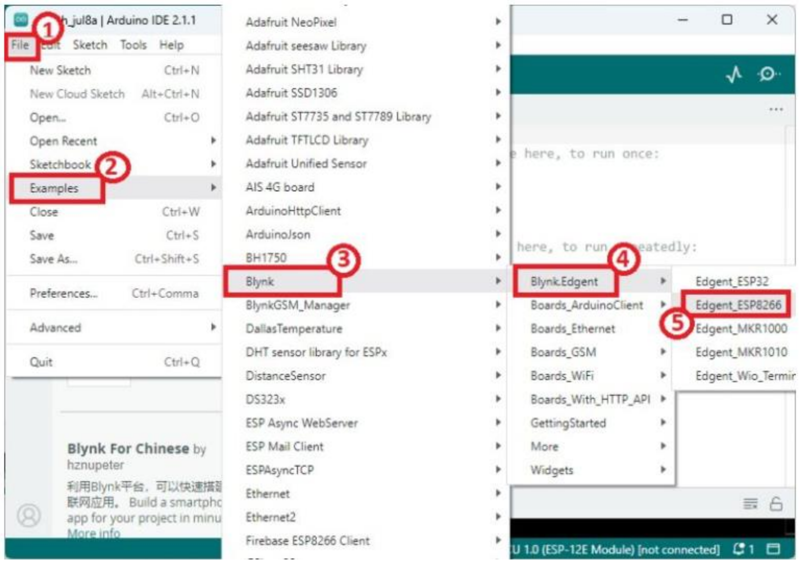
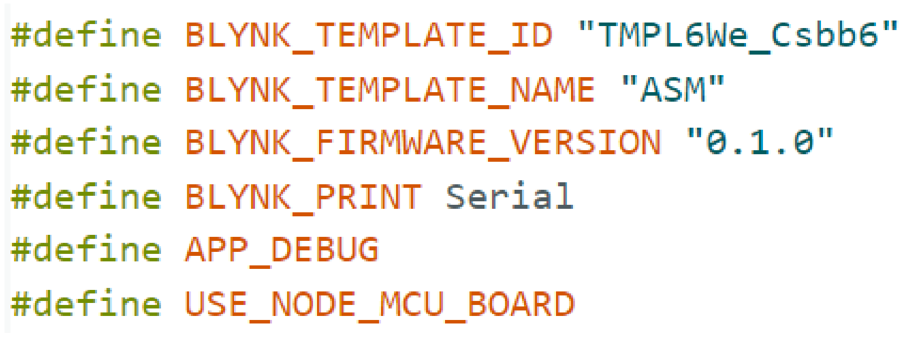


Figure 71 Open the Blynk sample code.

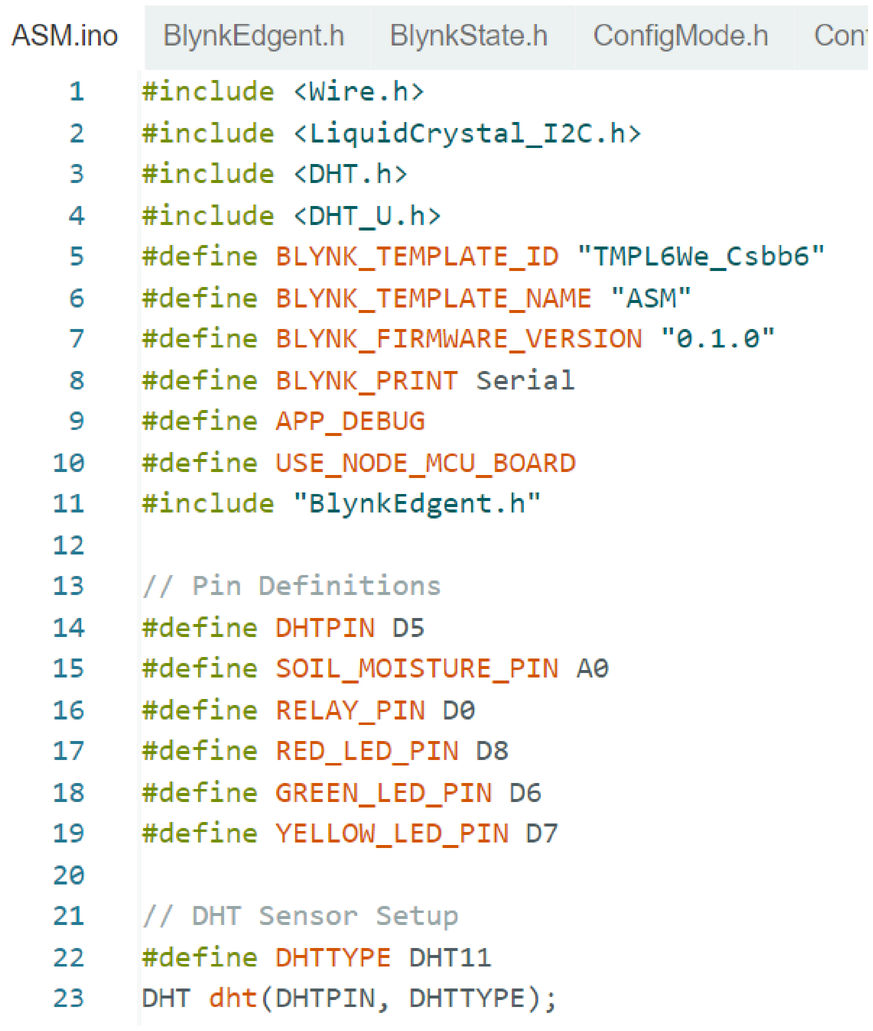
Step 3. Insert BLYNK_TEMPLATE_ID and BLYNK_DEVICE_NAME into the sample code and uncomment the #define USE_NODE_MCU section.



```
#define BLYNK_TEMPLATE_ID "TMPL6We_Csbb6"
#define BLYNK_TEMPLATE_NAME "ASM"
#define BLYNK_FIRMWARE_VERSION "0.1.0"
#define BLYNK_PRINT Serial
#define APP_DEBUG
#define USE_NODE_MCU_BOARD
```

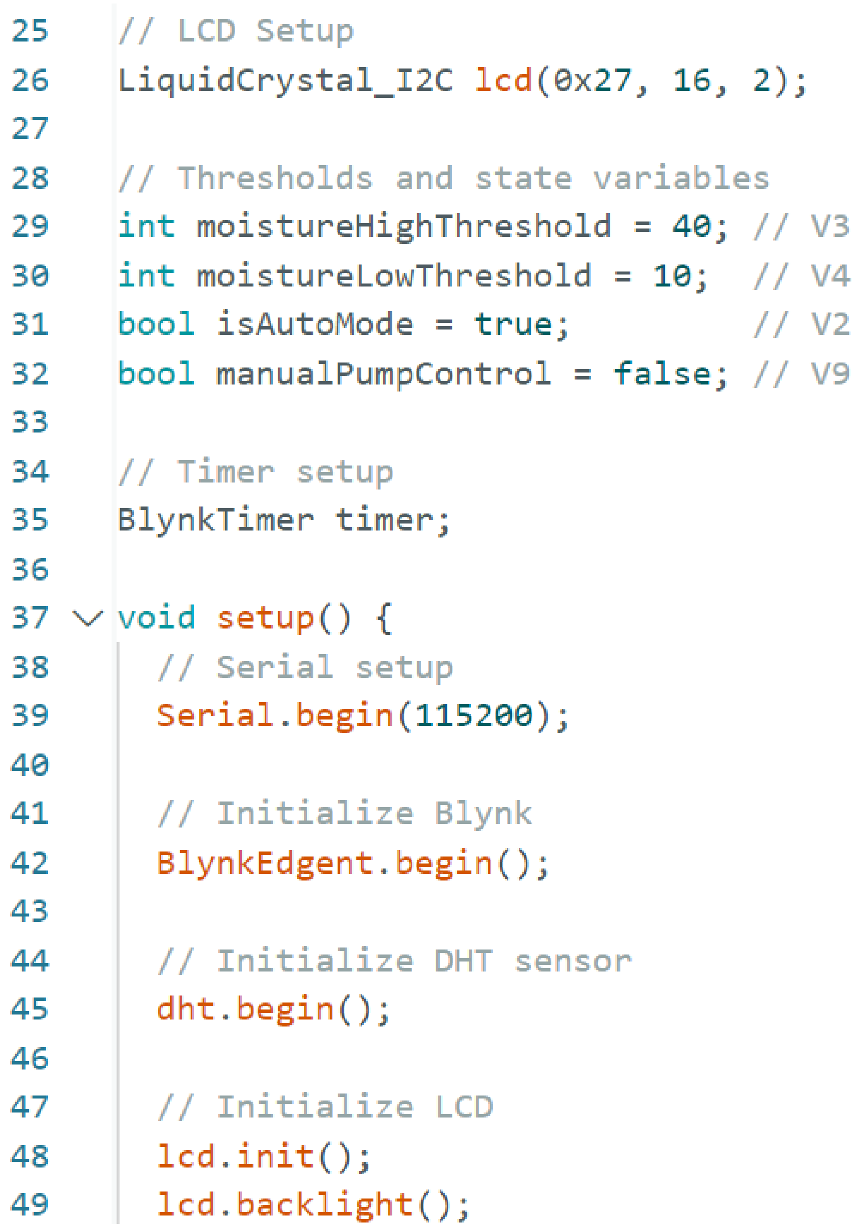
Figure 72 Insert BLYNK_TEMPLATE_ID and BLYNK_DEVICE_NAME.

Step 4. The process of writing code for an automatic watering system.



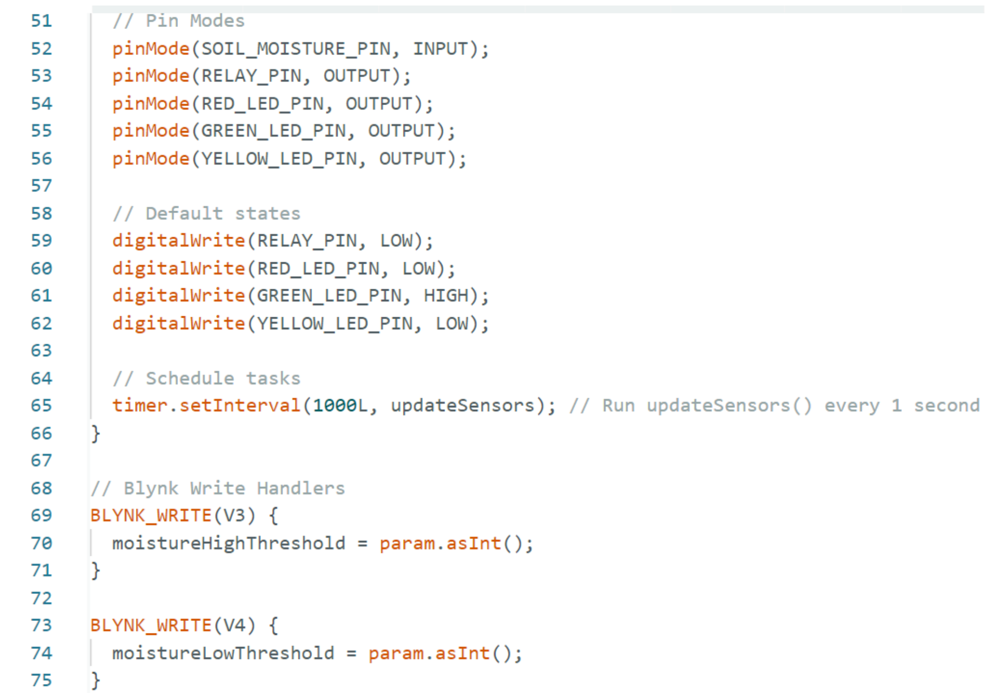
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <DHT_U.h>
#define BLYNK_TEMPLATE_ID "TMPL6We_Csbb6"
#define BLYNK_TEMPLATE_NAME "ASM"
#define BLYNK_FIRMWARE_VERSION "0.1.0"
#define BLYNK_PRINT Serial
#define APP_DEBUG
#define USE_NODE_MCU_BOARD
#include "BlynkEdgent.h"

// Pin Definitions
#define DHTPIN D5
#define SOIL_MOISTURE_PIN A0
#define RELAY_PIN D0
#define RED_LED_PIN D8
#define GREEN_LED_PIN D6
#define YELLOW_LED_PIN D7

// DHT Sensor Setup
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

Figure 73 Full processing code.



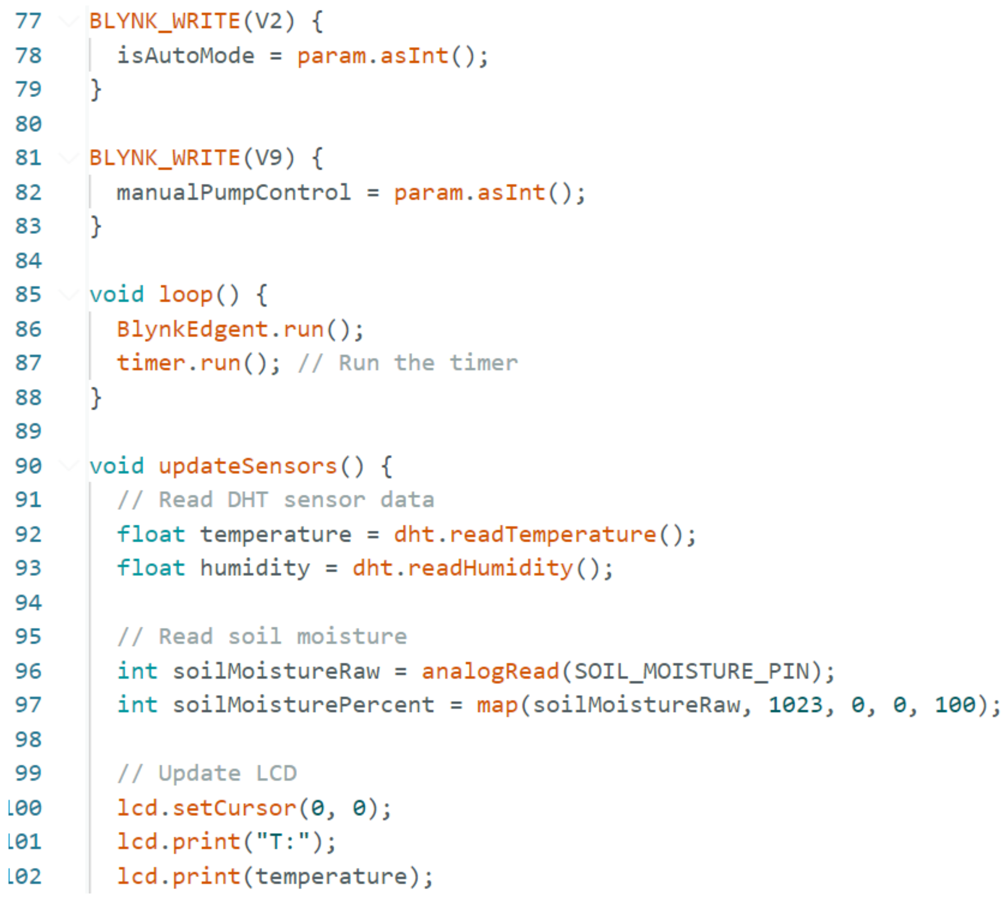
```
// LCD Setup
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Thresholds and state variables
int moistureHighThreshold = 40; // V3
int moistureLowThreshold = 10;  // V4
bool isAutoMode = true;         // V2
bool manualPumpControl = false; // V9

// Timer setup
BlynkTimer timer;

void setup() {
  // Serial setup
  Serial.begin(115200);

  // Initialize Blynk
  BlynkEdgent.begin();

  // Initialize DHT sensor
  dht.begin();

  // Initialize LCD
  lcd.init();
  lcd.backlight();
```

Figure 74 Full processing code.



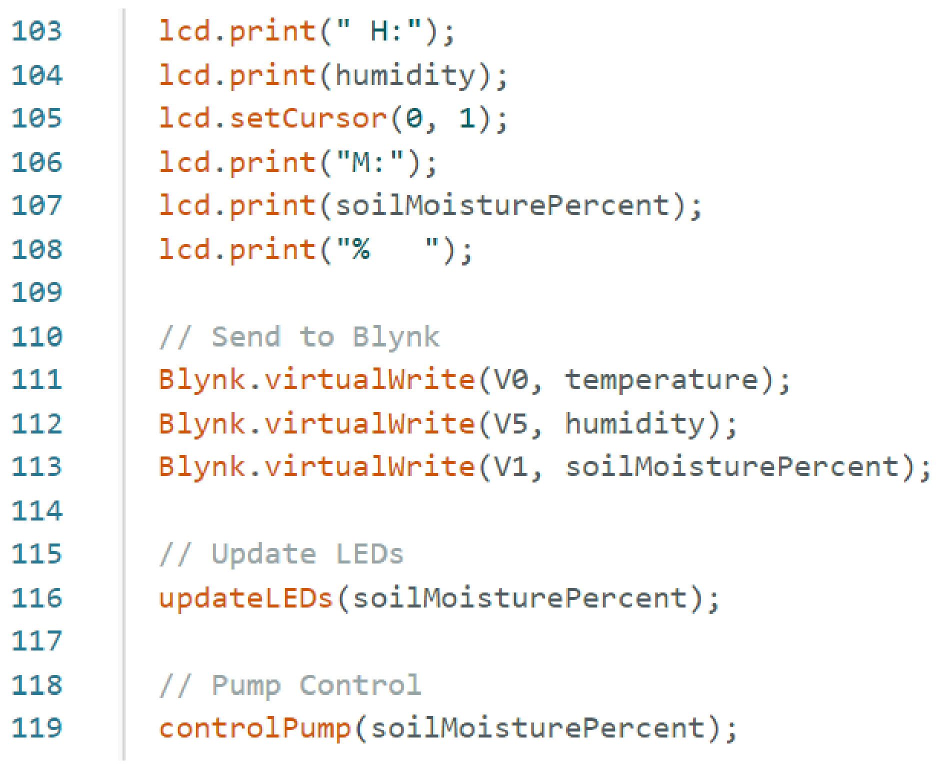
```
  // Pin Modes
  pinMode(SOIL_MOISTURE_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(YELLOW_LED_PIN, OUTPUT);

  // Default states
  digitalWrite(RELAY_PIN, LOW);
  digitalWrite(RED_LED_PIN, LOW);
  digitalWrite(GREEN_LED_PIN, HIGH);
  digitalWrite(YELLOW_LED_PIN, LOW);

  // Schedule tasks
  timer.setInterval(1000L, updateSensors); // Run updateSensors() every 1 second
}

// Blynk Write Handlers
BLYNK_WRITE(V3) {
  moistureHighThreshold = param.asInt();
}

BLYNK_WRITE(V4) {
  moistureLowThreshold = param.asInt();
}
```

Figure 75 Full processing code.



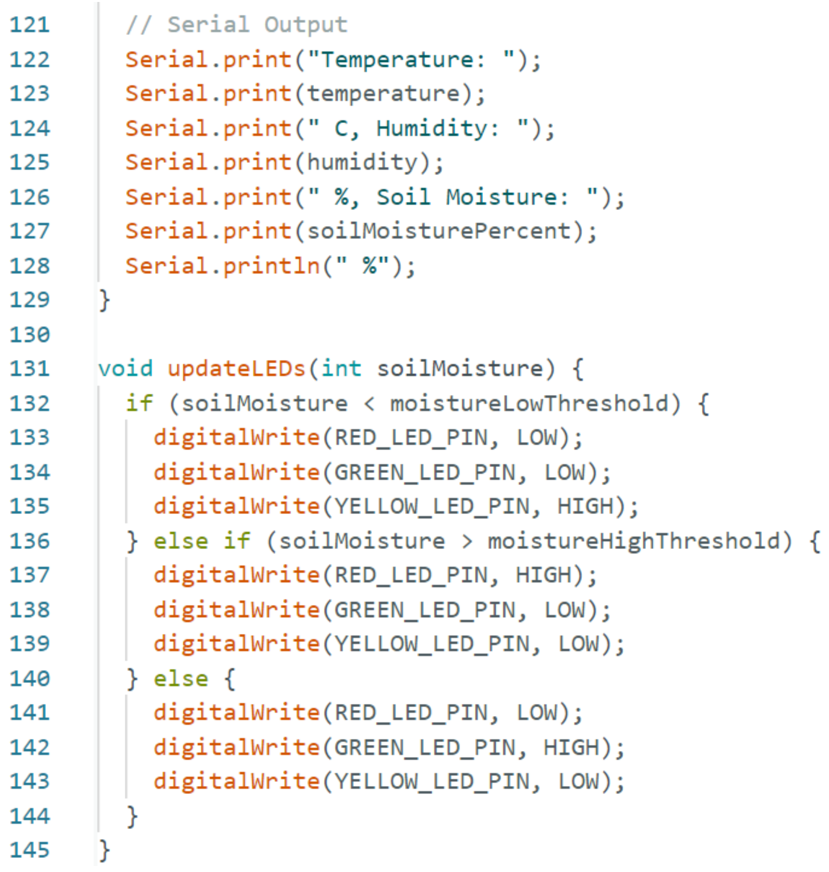
```
BLYNK_WRITE(V2) {
  isAutoMode = param.asInt();
}

BLYNK_WRITE(V9) {
  manualPumpControl = param.asInt();
}

void loop() {
  BlynkEdgent.run();
  timer.run(); // Run the timer
}

void updateSensors() {
  // Read DHT sensor data
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Read soil moisture
  int soilMoistureRaw = analogRead(SOIL_MOISTURE_PIN);
  int soilMoisturePercent = map(soilMoistureRaw, 1023, 0, 0, 100);

  // Update LCD
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(temperature);
```

Figure 76 Full processing code.



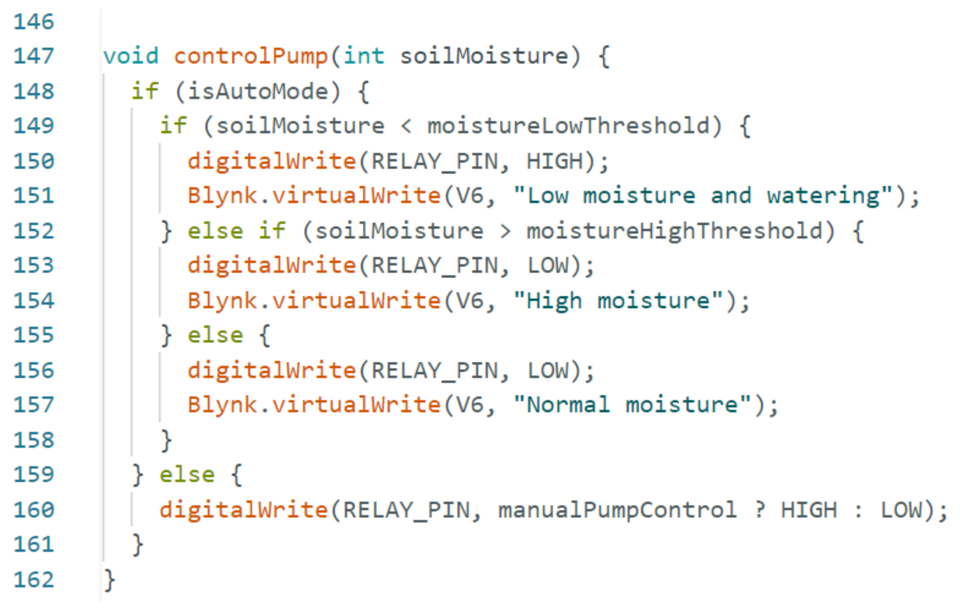
```
  lcd.print(" H:");
  lcd.print(humidity);
  lcd.setCursor(0, 1);
  lcd.print("M:");
  lcd.print(soilMoisturePercent);
  lcd.print("%   ");

  // Send to Blynk
  Blynk.virtualWrite(V0, temperature);
  Blynk.virtualWrite(V5, humidity);
  Blynk.virtualWrite(V1, soilMoisturePercent);

  // Update LEDs
  updateLEDs(soilMoisturePercent);

  // Pump Control
  controlPump(soilMoisturePercent);
```

Figure 77 Full processing code.



```
  // Serial Output
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" C, Humidity: ");
  Serial.print(humidity);
  Serial.print(" %, Soil Moisture: ");
  Serial.print(soilMoisturePercent);
  Serial.println(" %");
}

void updateLEDs(int soilMoisture) {
  if (soilMoisture < moistureLowThreshold) {
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(GREEN_LED_PIN, LOW);
    digitalWrite(YELLOW_LED_PIN, HIGH);
  } else if (soilMoisture > moistureHighThreshold) {
    digitalWrite(RED_LED_PIN, HIGH);
    digitalWrite(GREEN_LED_PIN, LOW);
    digitalWrite(YELLOW_LED_PIN, LOW);
  } else {
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(GREEN_LED_PIN, HIGH);
    digitalWrite(YELLOW_LED_PIN, LOW);
  }
}
```

Figure 78 Full processing code.

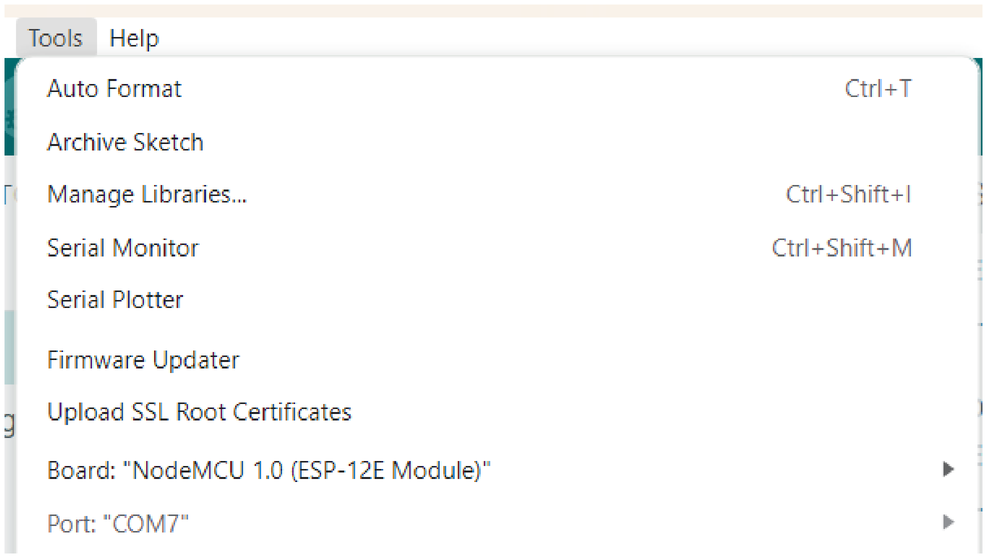


```
void controlPump(int soilMoisture) {
  if (isAutoMode) {
    if (soilMoisture < moistureLowThreshold) {
      digitalWrite(RELAY_PIN, HIGH);
      Blynk.virtualWrite(V6, "Low moisture and watering");
    } else if (soilMoisture > moistureHighThreshold) {
      digitalWrite(RELAY_PIN, LOW);
      Blynk.virtualWrite(V6, "High moisture");
    } else {
      digitalWrite(RELAY_PIN, LOW);
      Blynk.virtualWrite(V6, "Normal moisture");
    }
  } else {
    digitalWrite(RELAY_PIN, manualPumpControl ? HIGH : LOW);
  }
}
```

Figure 79 Full processing code.

5.2.1.3 Load the program into the nodemcu esp8266 kit.

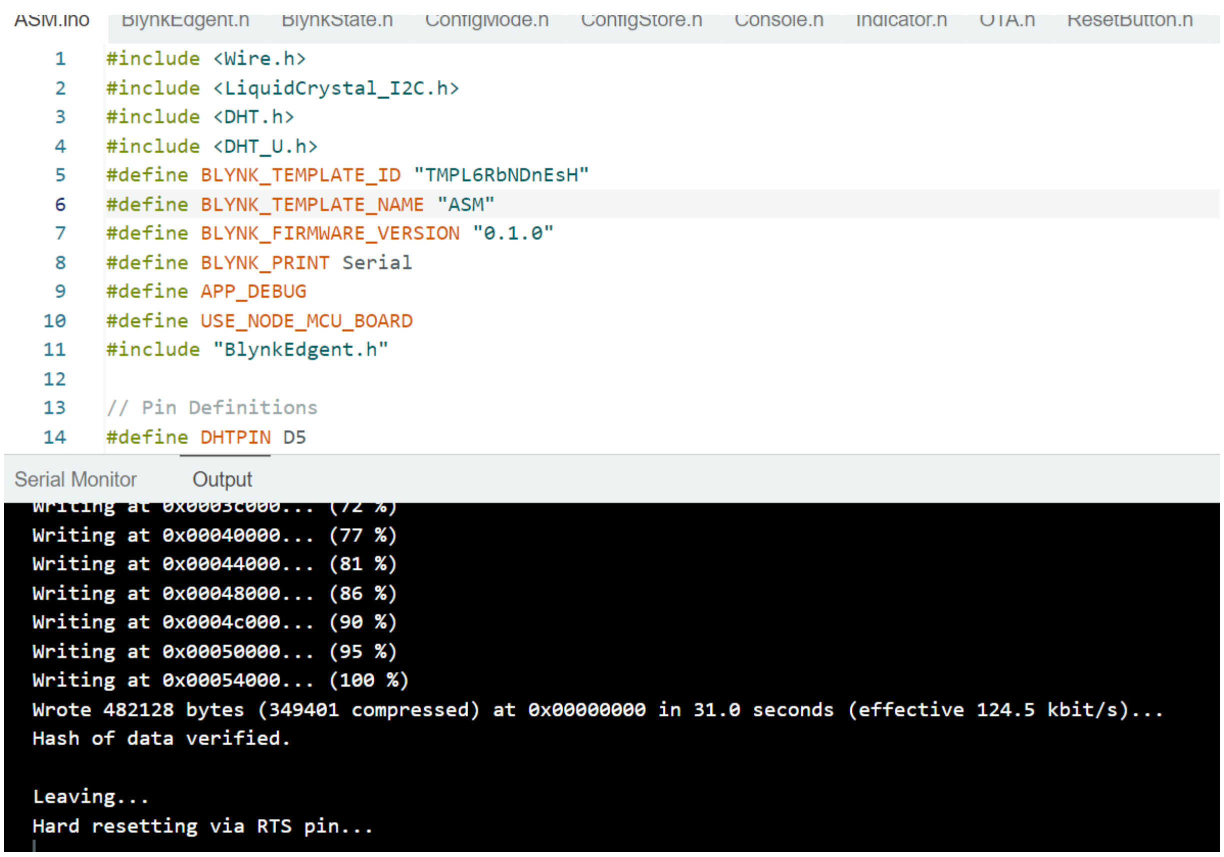
Step 1. After editing the code, go to Tool > Board > ESP8266, select the NODEMCU 1.0 (ESP12E Module) board, and select the COM port connecting to the microprocessor.


Tools
Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates

Board: "NodeMCU 1.0 (ESP-12E Module)"
Port: "COM7"

Figure 80 Select the NODEMCU 1.0 (ESP12E Module) board and COM port.

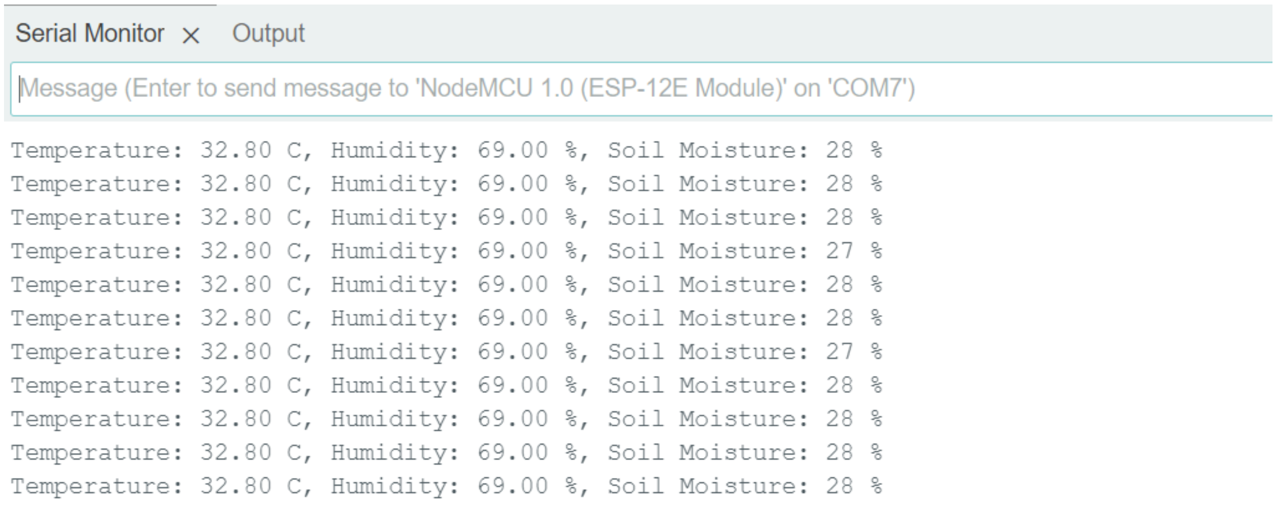
Step 2. Next, go to Sketch, select Upload and wait for the Arduino IDE to compile and download the code. Open Serial monitor port and adjust baud rate to 115200 to see Debug program Blynk IOT and nodemcu esp8266.



```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <DHT_U.h>
#define BLYNK_TEMPLATE_ID "TMPL6RbNDnEsH"
#define BLYNK_TEMPLATE_NAME "ASM"
#define BLYNK_FIRMWARE_VERSION "0.1.0"
#define BLYNK_PRINT Serial
#define APP_DEBUG
#define USE_NODE_MCU_BOARD
#include "BlynkEdgent.h"

// Pin Definitions
#define DHTPIN D5
```

```
Writing at 0x0003c000... (72 %)
Writing at 0x00040000... (77 %)
Writing at 0x00044000... (81 %)
Writing at 0x00048000... (86 %)
Writing at 0x0004c000... (90 %)
Writing at 0x00050000... (95 %)
Writing at 0x00054000... (100 %)
Wrote 482128 bytes (349401 compressed) at 0x00000000 in 31.0 seconds (effective 124.5 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Figure 81 Upload and wait for the Arduino IDE to compile and download the code.


Serial Monitor
Output
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM7')

```
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 27 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 27 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
Temperature: 32.80 C, Humidity: 69.00 %, Soil Moisture: 28 %
```

Figure 82 select Upload and wait for the Arduino IDE to compile and download the code.

5.2.2 Web Apps.

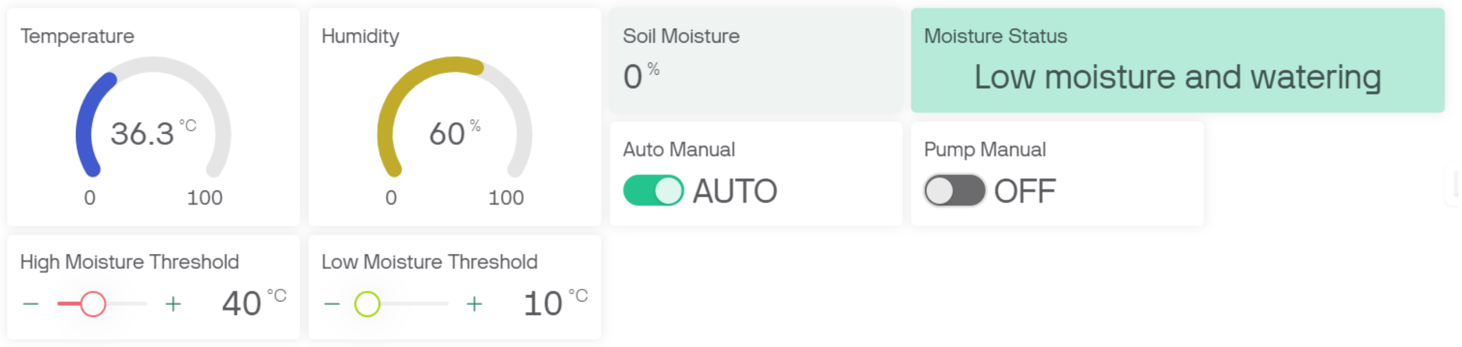
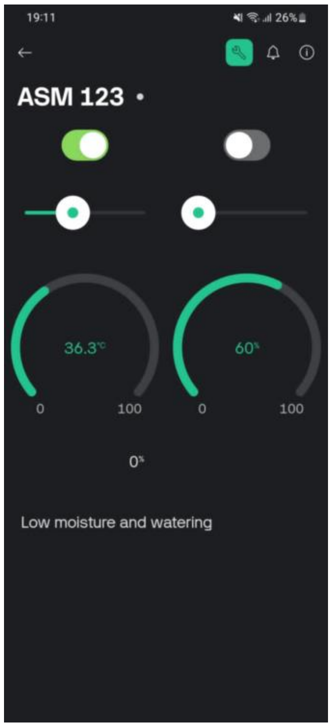


Figure 83 Application interface on Blynk web.

The WED interface includes:

- 2 Gauge displays air temperature and humidity.
- The label shows soil moisture.
- The label shows the soil moisture status.
- Switch to switch back and forth between Auto and Manual modes.
- Switch to turn the pump on and off in Manual mode.
- 2 Slider sets the gloom threshold for the tree.

5.2.3 Interface on Mobile application.



Low moisture and watering

5.3 Connectivity and Communication.

The system uses ESP8266 as the control center to connect components such as sensors, relays, LEDs, LCD_I2C and Blynk application:

- Wi-Fi Communication: ESP8266 connects to Wi-Fi network to communicate with Blynk application, helping users to monitor and control remotely through intuitive interface on smartphone.
- Serial Communication: Data from sensors (DHT11, soil moisture) is transmitted to ESP8266 and displayed on Serial Monitor at baud rate 115200.
- GPIO Control: GPIO pins on ESP8266 connect to sensors, relays, and LEDs to perform control operations and display status.

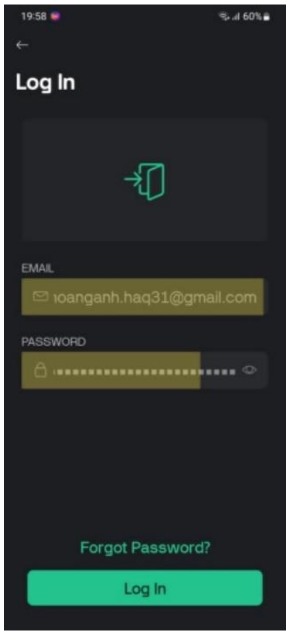
5.3.1 Wi-Fi Connection.

The ESP8266 is configured via the BlynkEdgent.begin() function, where the Wi-Fi SSID and password are set from the configuration in the Blynk application.

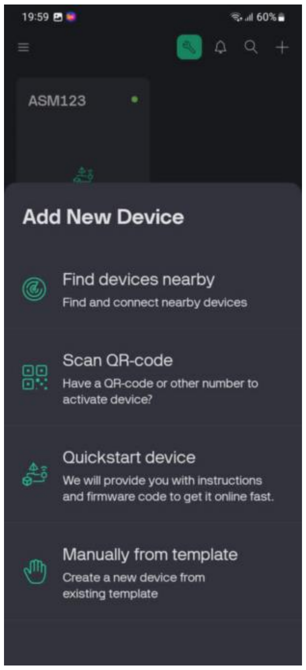
The ESP8266 will automatically connect to the Wi-Fi network to ensure communication.

In systems using the BlynkEdgent library, the SSID (Wi-Fi network name) and Wi-Fi password information are not hardcoded in the source code but set directly from the Blynk application during initial configuration. This allows flexibility when changing Wi-Fi networks without having to update or reload source code to the ESP8266.

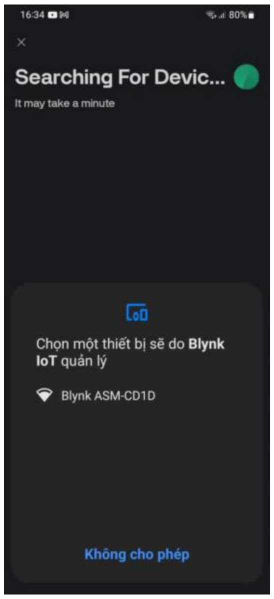
Step 1. Open the Blynk IoT app and log in with your account. Select Add New Device.



Step 2. Select Find devices nearby > Start > Continue and select the device named Blynk and select the Wi-Fi network to connect to.


19:59
ASM123
Add New Device
Find devices nearby
Find and connect nearby devices
Scan QR-code
Have a QR-code or other number to activate device?
Quickstart device
We will provide you with instructions and firmware code to get it online fast.
Manually from template
Create a new device from existing template

When the ESP8266 is first programmed or after pressing the Wi-Fi reset button (on the Blynk app or via hardware), the ESP8266 will create a temporary Wi-Fi network (Access Point - AP), named "Blynk ASM-CD1D"



Step 3. The user connects to the temporary Wi-Fi network created by the ESP8266, then uses the Blynk app to set the SSID and password of the Wi-Fi network that the ESP8266 will connect to.

After the user enters the SSID and password information on the Blynk application, the ESP8266 will:

- Store this information in non-volatile memory (usually flash memory) for use during subsequent boots.
- When the ESP8266 is powered up or restarted, it will automatically connect to the specified Wi-Fi network without requiring any reconfiguration (unless the configuration is reset).

5.3.2 Communicating with Blynk via TCP/IP

With the BlynkEdgent.h Library, the System uses TCP/IP protocol to communicate with the Blynk server. This communication is maintained continuously to send and receive data from the application.

The ESP8266 acts as a client connecting to the Blynk server via the default port.

Virtual Pins are used to communicate data between devices (sensors, pumps) and the Blynk dashboard:

- Read sensor data: Send humidity, temperature values from the device to Blynk.
- Receive control commands: Receive pump status, auto/manual mode from the dashboard.

This communication is two-way communication:

Direction from ESP8266 to Blynk: Data from sensors (temperature, air humidity from DHT11 and soil humidity) is sent to the application via Virtual Pins:

- V0: Display temperature on gauge.
- V5: Display air humidity on gauge.
- V1: Display soil humidity on label.
- Soil humidity and pump status are updated via V6.

Direction from Blynk to ESP8266: The user controls the system via control buttons in the application:

- V2: Switch between Auto and Manual modes.
- V9: Turn the pump on/off manually (in Manual mode).
- V3 and V4: Adjust the high and low soil moisture thresholds.

5.3.3 Serial Communication

ESP8266 uses Serial Communication to display sensor data on Serial Monitor via USB port. This helps to test and debug the system during development, monitor data in real time when connected to a computer. How it works:

Data from the sensor is collected and printed to Serial Monitor at baud rate 115200.

Serial Monitor displays:

- Temperature from DHT11 sensor.
- Air humidity from DHT11 sensor.
- Soil moisture (converted to percent).

5.3.4 I2C (Inter-Integrated Circuit) Communication.

LCD_I2C uses the I2C protocol to communicate with the ESP8266.

ESP8266 communicates with the LCD screen via two wires:

- SDA: Data (Serial Data Line).
- SCL: Clock (Serial Clock Line).
- Default address of LCD_I2C: 0x27.

ESP8266 sends commands and data to LCD_I2C to display:

- Line 1: Temperature and humidity from DHT11.
- Line 2: Soil moisture (percent).
- Data is refreshed every 2 seconds to ensure real-time.

5.3.5 GPIO (General Purpose Input/Output Pins) Connection

ESP8266 controls peripheral devices through GPIO pins.

Table 2 Connection Details.

Device	ESP8266 GPIO Pin	Signal Type
DHT11	D5	Input
Soil Moisture Sensor	A0	Analog Input
Red LED	D8	Output
Yellow LED	D7	Output
Green LED	D6	Output
Relay (Pump)	D0	Output

How it works:

- Input:
    - DHT11 sends digital signals about temperature and humidity.
    - Soil Moisture Sensor sends analog signals about soil moisture level.
- Output:
    - GPIO pins control relays and LEDs based on program logic.
        - Relay turns the pump on/off depending on soil moisture and mode (Auto/Manual).
        - LED displays soil moisture status (low threshold, high threshold, normal).

6 Create a detailed test plan and examine feedback. P6

6.1 Developing the Test Plan.

6.1.1 Test Objectives.

The objectives for developing the test plan are:

- Verify the operation of each hardware component: ESP8266, DHT11, soil moisture sensor, LED, relay, LCD_I2C.
- Ensure correct control logic: Switch between manual and automatic modes, pump on/off logic, display the correct information.
- Ensure Blynk integration works properly: Display data on widgets (gauge, label, switch).
- Measure system accuracy and stability: Read sensors and perform control according to preset thresholds.

6.1.2 Test plan scope.

Hardware (connection and operation of sensors, relays, LEDs).

Software (LED control logic, pump, Auto/Manual mode).

Blynk integration (parameter display, mode switching, pump control).

Display on LCD_I2C.

Check Serial Monitor.

6.2 Executing the Test Plan.

This test plan outlines the testing process for an automated and manual irrigation system. The system uses an ESP8266 microcontroller, DHT11 sensor, soil moisture sensor, relays, and LEDs, controlled via a Blynk application and LCD display. The tests cover a variety of scenarios to ensure that all components function as expected, including sensor readings, LEDs, relay control, and user interface feedback. The results will be evaluated to confirm that the system meets the desired functionality.

Below is a table showing the results after conducting the tests with detailed test cases:

Test case	Description	Step Details	Expected Results	Actual Results	Notes
1	Check DHT11 Sensor Output	1. Run the program. 2. Observe the values on the serial monitor, LCD, and Blynk.	Temperature and humidity values are correctly displayed on the serial monitor, LCD, and widgets V0 and V5.	1st time: fail. 2nd time: pass	DHT11 signal is unstable, with values fluctuating.

2	Read Soil Moisture Value	1. Place the sensor in dry, moist, and submerged soil. 2. Observe the value on serial, LCD, and Blynk.	Soil moisture value is correctly converted to percentage and displayed on serial, LCD (line 2), and widget V1.	Pass	No issues.
3	Control LEDs Based on Soil Moisture	1. Simulate low, high, and medium moisture values.	Yellow LED lights up when moisture is below the low threshold. Red LED lights up when moisture is above the high threshold. Green LED lights up within the normal range.	Pass	No issues.
4	Control Pump in Auto Mode	1. Set switch V2 to Auto. 2. Simulate soil moisture: low, high, and medium.	Pump turns on when moisture is below the low threshold. Pump turns off when moisture is above the high threshold or within the normal range.	Pass	No issues.
5	Display Pump Status on Blynk (V6)	1. Observe the status on V6 when the pump is on or off.	Display: - "Low moisture and watering" when moisture is low. - "High moisture" when moisture is high. - "Normal moisture" when moisture is within range.	Pass	No issues.

6	Control Pump in Manual Mode	1. Set switch V2 to Manual. 2. Use switch V9 to turn the pump on/off.	Pump turns on/off based on the status of switch V9.	Pass	No issues.
7	Switch Between Manual and Auto Mode	1. Toggle switch V2 between Auto and Manual.	The system switches between modes and operates according to the logic of each mode.	Pass	No issues.
8	Check LCD_I2C Display	1. Observe the LCD after running the program.	Line 1: "T: <temp> H: <humidity>" Line 2: "M: <soil moisture>"	Pass	No issues.
9	Display Data on Serial Monitor (115200 baud)	1. Observe the data from the serial monitor.	Correct temperature, humidity, and soil moisture data are displayed on the serial monitor.	Pass	No issues.
10	Check Relay Operation	1. Observe the relay when the pump turns on/off.	The relay is activated correctly based on the pump's status.	Pass	No issues.

6.3 Analyzing Feedback.

Pass: 9/10 Test Cases Passed Out of the 10 test cases, 9 have successfully passed, indicating that the majority of the system's functionality is operating as expected. These tests covered a range of functionalities, including the reading of temperature and humidity from the DHT11 sensor, correct soil moisture readings, proper LED control based on soil moisture levels, and accurate pump operation in both automatic and manual modes. Additionally, the system is displaying the correct information on the LCD and serial monitor, as well as transmitting data to the Blynk app as intended. This demonstrates that the core functionality of the system, such as sensor readings, relay control, and communication with the app, is reliable.

Fail: DHT11 Signal Unstable The failure was observed in the test case related to reading the temperature and humidity values from the DHT11 sensor. The issue identified is the instability of the DHT11 sensor's signal, where the values fluctuate and do not remain consistent over time.

To address this issue and improve the sensor's reliability, several potential solutions are recommended:

- Implement Signal Filtering: The DHT11 sensor’s signal may be susceptible to noise and fluctuations. A moving average filter or debouncing technique could be used to smooth out these fluctuations and provide more stable readings. A moving average filter works by averaging the sensor readings over a set period, effectively reducing random noise. Alternatively, debouncing can be applied to eliminate spurious signals that may be generated by slight fluctuations or glitches in the sensor's output.
- Provide a Stable Power Supply: The DHT11 sensor is known to be sensitive to power fluctuations, which can contribute to unstable readings. It is crucial to ensure that the sensor is provided with a stable and clean power source. A dedicated voltage regulator or a higher-quality power supply could be used to reduce the chances of power instability, which may help maintain more consistent sensor performance.
- Consider Replacing the DHT11 Sensor: The DHT11 sensor, while inexpensive, is not the most accurate or reliable option for temperature and humidity measurements, especially in more critical applications. The DHT22 sensor is a more precise and stable alternative that provides better accuracy over a wider range of temperatures and humidity levels. Switching to a DHT22 could result in more reliable sensor readings and improved overall system performance.

By addressing the signal instability, the system can achieve more consistent performance, ensuring that sensor readings are accurate and reliable for proper decision-making and control of the watering system.

7 Review the IoT application detailing the problems it solves. P7

7.1 Description of the Application.

Smart plant pot application for automatic/manual watering with ESP8266 helps users take care of plants more effectively through the use of DHT11 sensor (measuring air temperature and humidity), soil moisture sensor, and LED lights (red, yellow, green) to display the status of plants and soil. The system can operate automatically or manually, with the following features:

- Automatic mode: Based on soil moisture, the system will automatically control the pump to water the plants when the humidity is below the low threshold (v4) and stop when the humidity exceeds the high threshold (v3).
- LED will light up according to the soil moisture status (red, yellow, green).
- Manual mode: Users can control the pump directly via the button on the Blynk app. In this case, the pump can be turned on/off according to user needs.
- Information about the indicators (temperature, air humidity, soil moisture) is displayed on LCD_I2C and sent to the Blynk application for users to monitor.

This system helps users easily control and care for plants, and provides an effective alternative to traditional manual watering.

7.2 Problems Solved.

The system has solved the following problems:

- The automatic watering system is based on soil moisture sensors, helping to avoid water shortage or excess water. The pump will operate when the soil moisture is low, and stop when the moisture reaches a high threshold, ensuring the appropriate water supply for the plants.
- Energy saving: The system helps to optimize the amount of water used for plants, minimizing the amount of resources consumed.
- The system fully automates watering, helping users not to worry about forgetting to water or watering too much. Users can rest assured that the plants will be watered at the right time without having to be there.
- Automating plant care saves time and effort for growers.
- Increasing crop yield and health: Plants will always have enough moisture needed to grow well, thereby increasing crop yield and health.
- The system displays information about temperature, air humidity and soil moisture directly on the LCD and Blynk application. Users can monitor the condition of plants and soil remotely, even when not near the plants.
- With the integration of soil moisture sensors and the Blynk app, users will receive notifications about soil moisture status in real time, making it easy to recognize when plants need water or are sufficiently moist.

- Use in urban areas: The system can be applied in areas with water shortages or where there is limited space for planting, creating favorable conditions for planting plants in the home, office or garden.

8 Investigate the potential problems the IoT application might encounter when integrating into the wider system. P8

8.1 Integration Issues.

Issues that may occur when integrating an IoT system into a larger system:

- Network Connectivity Issues: The IoT system uses the ESP8266 to connect to the Internet and the Blynk application. If the Wi-Fi connection is unstable or there is a network problem, data from sensors or device status (pumps, LEDs, etc.) may not be sent in time, affecting the automatic operation of the system.
- Software and Hardware Compatibility: The system components (ESP8266, DHT11 sensor, soil moisture sensor, LED, LCD_I2C, etc.) may have compatibility issues if the firmware or control software is not updated or does not match each other. This may result in errors in displaying information on the LCD screen or errors in sensor data.
- Big Data Management: As the system grows and connects to more devices, collecting and processing large amounts of data from sensors (temperature, humidity, etc.) can become difficult to store and process, especially when this data needs to be continuously transmitted over Wi-Fi or via Blynk.
- Security and Privacy Concerns: IoT systems can be vulnerable to attacks if not properly secured. Vulnerabilities can include data sent over the Internet being unencrypted, or remote control of pumps and LEDs being tampered with by malicious actors.
- Scalability Issues: As the system expands to incorporate more devices, scalability issues can arise. Adding sensors and control devices to the Wi-Fi network can degrade system performance if not properly optimized.

8.2 Analysis and Solutions.

8.2.1 Network Connection Issues.

Analysis: An unstable network connection can disrupt data collection and transmission, causing the system to not function as expected, especially when using the Blynk application for monitoring and control.

Solution: Improve the Wi-Fi network and ensure a more stable network connection, such as using a Wi-Fi router with good coverage. You can add an auto-reconnect feature if the connection is lost. Add the ability to temporarily store data and send it after the connection is restored.

8.2.2 Software and Hardware Compatibility.

Analysis: When the components of the system are not compatible with each other, errors may occur or the system may not function properly. For example, sensors or LEDs may not function properly if there is no synchronization between the hardware and software.

Solution: Update the firmware of the ESP8266 and the control software regularly. Ensure software libraries (Arduino libraries for DHT11, LCD_I2C, etc.) are up to date and compatible with each other. Test and validate the operation of each component before integrating into the system.

8.2.3 Big Data Management.

Analysis: As the number of devices and sensors increases, the volume of data collected increases, which can impact the performance and storage capacity of the system.

Solution: Edge processing using systems such as Raspberry Pi or more powerful microcontrollers instead of relying solely on the ESP8266. Improve data processing capabilities and store data on a cloud platform for easy management and retrieval.

8.2.4 Security and Privacy.

Analysis: IoT systems can become targets of attacks if not protected, such as data tampering or pump hijacking.

Solution: Use SSL/TLS encryption for data sent over the Internet to protect privacy and security. Set up strong authentication measures to prevent unauthorized access to the system. Additionally, protect the Wi-Fi system with strong passwords and update the device firmware regularly.

8.2.5 Scalability.

Analysis: Expanding the system to add more sensors and devices can reduce performance if not well optimized.

Solution: Design the system to be scalable, such as using a distributed network architecture or a cloud system to store and process data. Ensure the system can connect and control a large number of devices without reducing performance.

Conclusion.

The integration of IoT technology into smart plant pots demonstrates the vast potential of automation in simplifying everyday tasks while also enhancing the efficiency of plant care. Through the use of sensors and devices such as the ESP8266 and Blynk platform, we have outlined the key steps in developing an intelligent system capable of monitoring and adjusting watering cycles based on real-time environmental conditions. This application not only provides convenience but also promotes sustainability by optimizing water usage for plants. However, as we have discussed, the integration of IoT into a wider system comes with various challenges, such as network connectivity issues, hardware-software compatibility, data management, security concerns, and scalability. Addressing these issues through careful planning and the use of appropriate tools, frameworks, and APIs ensures the system can function efficiently in a larger IoT ecosystem. In conclusion, the smart plant pot system represents just one of many innovative IoT applications that are transforming the way we interact with technology and our environment, offering significant opportunities for both personal and industrial applications.

References

Arshdeep Bahga, Vijay Madisetti, 2014. *Internet of Things (A Hands-on-Approach).* 1st ed. s.l.:VPT.

aws, n.d. *What is IoT (Internet of Things)?.* [Online] Available at: https://aws.amazon.com/vi/what-is/iot/
[Accessed 10 21 2024].

Braun, A., 2019. *History of IoT: A Timeline of Development.* [Online] Available at: https://www.iottechtrends.com/history-of-iot/
[Accessed 21 10 2024].

fast, 2023. *What is IoT? Application of IoT in modern life.* [Online] Available at: https://fast.com.vn/internet-of-things-la-gi-ung-dung-cua-internet-of-things-trong-cuoc-song-hien-dai/
[Accessed 21 10 2024].

Foote, K. D., 2022. *A Brief History of the Internet of Things.* [Online] Available at: https://www.dataversity.net/brief-history-internet-things/
[Accessed 21 10 2024].

Hoi, H. H., 2018. *How do IOT devices connect to the internet?.* [Online] Available at: https://viblo.asia/p/cac-thiet-bi-iot-ket-noi-internet-nhu-the-nao-bWrZnNJrZxw
[Accessed 21 10 2024].

Viettelz, 2023. *What is IoT? The role and application of IoT in life.* [Online] Available at: https://viettelz.com/iot-la-gi/
[Accessed 21 10 2024].