A
Major Project Report on

RAIN DETECTION AND MOTOR CONTROL FOR SMART AGRICULTURE

Submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

SUBMITTED BY

J. MADHAN NAYAK	**218R1A0425**
JITENDER PAL SINGH	**218R1A0426**
K. SRINIVAS REDDY	**218R1A0427**
K. MANOJ KUMAR	**218R1A0428**

Under the Esteemed Guidance of

Dr. S. RAMA KISHORE REDDY
Associate Professor

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CMR ENGINEERING COLLEGE
UGC AUTONOMOUS

(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA & NAAC)
Kandlakoya (V), Medchal (M), Telangana – 501401

2024-2025

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA & NAAC)
Kandlakoya (V), Medchal (M), Telangana – 501401

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the Major Project work entitled **"RAIN DETECTION AND MOTOR CONTROL FOR SMART AGRICULTURE"** is being submitted by **J. MADHAN NAYAK** bearing Roll No: **218R1A0425, JITENDER PAL SINGH** bearing Roll No: **218R1A0426, K. SRINIVAS REDDY** bearing Roll No: **218R1A0427, K. MANOJ KUMAR** bearing Roll No: **218R1A0428** in B.Tech IV - II semester, Electronics and Communication Engineering is a record bonafide work carried out by then during the academic year 2024-25. The results embodied in this report have not been submitted to any other University for the award of any degree.

INTERNAL GUIDE

Dr. S. RAMA KISHORE REDDY
Associate Professor

HEAD OF THE DEPARTMENT

Dr. SUMAN MISHRA
Professor

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

We sincerely thank the management of our college **CMR ENGINEERING COLLEGE** for providing required facilities during our project work.

We derive great pleasure in expressing our sincere gratitude to our Principal **Dr. A. S. REDDY** for his timely suggestions, which helped us to complete the project work successfully.

It is the very auspicious moment we would like to express our gratitude to **Dr. SUMAN MISHRA,** Head of the Department, ECE for his consistent encouragement during the progress of this project.

We take it as a privilege to thank our major project coordinator **Dr. T. SATYANARAYANA**, Associate Professor, Department of ECE for the ideas that led to complete the project work and we also thank him for his continuous guidance, support and unfailing patience, throughout the course of this work.

We sincerely thank our major project internal guide **Dr. S. RAMA KISHORE REDDY,** Associate Professor of ECE for guidance and encouragement in carrying out this project work.

DECLARATION

We hereby declare that the project work entitled **"RAIN DETECTION AND MOTOR CONTROL FOR SMART AGRICULTURE"** is the work done by us in campus at **CMR ENGINEERING COLLEGE,** Kandlakoya during the academic year 2024-2025 and is submitted as Major project in partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING FROM JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD.**

J. MADHAN NAYAK	(218R1A0425)
JITENDER PAL SINGH	(218R1A0426)
K. SRINIVAS REDDY	(218R1A0427)
K. MANOJ KUMAR	(218R1A0428)

ABSTRACT

Irrigation is a vital part of Agriculture and motors are used to water the crops. In the rainy season, fields are prone to over irrigation due to running of motor, when the service of motor is not required. This can cause the crop to fail or the yield of crops can be badly affected. The failed crops or the bad yield brings loss to the farmer.

To overcome this issue and improve the yield of the crops, we are using Internet of Things (IoT) which plays a vital role. We are automating the irrigation process and all the other processes related to irrigation thus it now requires minimal human intervention. To do this we are detecting the rain and controlling the motor according to the requirement. In this farmer can even control the motors from home without going to the fields. An email alert about all the events handled is also sent. This project also has a feature where the farmer can even enquire the status of the motor and control it.

To automate the irrigation process and control the motors, we are using Node MCU which is an open source IoT platform. A rain sensor to detect the rain and a soil moisture sensor to detect the moisture level of the soil. Motor is used to irrigate the fields and a relay to drive the connected motor. Blynk IoT cloud is used to enable communication between the device and the user.

If it rains while the motor is ON, then the rain sensor will detect it and turn off the motor, after turning OFF the motor an acknowledgement message is sent to the farmer. The farmer can also check the status of the motor from anywhere and control it. The communication between the motor and various sensors is taken care by the Node MCU. Blynk is used to establish a communication path between the user and the device, it provides email alerts and notifications to the farmer and the farmer can take necessary actions based on the requirements.

Keywords: Node MCU, Rain Detector, Soil Moisture Sensor, Blynk IoT, Motor.

CONTENTS

CHAPTERS	PAGE NO

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This Irrigation System uses Internet of Things (IoT) technology to help farmers efficiently manage their irrigation processes, reducing human intervention and optimizing water usage. The system integrates several sensors, including a rain sensor and a soil moisture sensor, to automatically control the irrigation motor based on real-time environmental data. The rain sensor detects rainfall and automatically turns off the motor if it starts raining, preventing over-irrigation. The soil moisture sensor continuously monitors the moisture levels in the soil, and when the moisture level drops below a set threshold, the system activates the irrigation motor to water the crops. If the soil has adequate moisture, the motor remains off.

At the heart of the system is the Node MCU, a microcontroller that processes data from the sensors and controls the motor. A motor driver is used to control the irrigation motor based on commands from the Node MCU. Additionally, the Blynk IoT enables communication between the system and the farmer. The Blynk sends email alerts to the farmer about critical events, such as the motor being turned off due to rain or when the soil moisture drops too low. The farmer can also use IoT cloud to remotely check the status of the motor and control its operation, turning it on or off as needed.

The system works by continuously monitoring soil moisture and rainfall. If the soil moisture is low, the motor is activated to irrigate the field. If it starts raining, the rain sensor detects the rainfall and sends a signal to shut off the motor, ensuring water is not wasted. The farmer is notified via email about these changes, allowing them to take action if necessary. The ability to monitor and control the system remotely through email offers flexibility and convenience, particularly in areas where farmers may not always be on-site.

This system offers several benefits, including water conservation by preventing over-irrigation, reducing labour by automating the irrigation process, and improving crop yield by ensuring optimal watering conditions. However, challenges such as ensuring a stable power supply, calibrating the sensors for accurate readings, and maintaining reliable internet coverage in rural areas need to be addressed for the system to function effectively.

Overall, the Automated Irrigation System is a smart farming solution that leverages IoT technology to help farmers manage irrigation more efficiently, reduce water wastage, and improve crop yields, ultimately contributing to more sustainable agricultural practices.

1.1 OBJECTIVE OF THE PROJECT

This project aims to design and develop an intelligent irrigation solution that minimizes human intervention while optimizing water usage and crop yield utilization, accomplished by using the Internet of Things (IoT) technology. The system is designed to overcome the problem of over-irrigation, especially during the rainy season, through the use of sensors for rainfall and soil moisture levels. It is through the automation of the control of the irrigation motor from real-time environmental data that the system ensures crops get the right amount of water, at the right time, without wasting water. A major goal is to save some water by not over-irrigating in the first place, through the integration of a rain sensor that turns the irrigation motor off if rain is detected.

Another goal is the remote monitoring and control of the irrigation system over SMS, so that the farmer can check on the status of the motor and adjust from anywhere without necessarily having to be in the field, even when he is busy at another location. This allows saving time while ensuring prompt handling of irrigation tasks. Additionally, SMS notifications are also relayed to the farmer about events occurring in the farm, such as when the motor is shut off by rain or when soil moisture falls below the threshold. This notification keeps him in the information, and hence he can take corresponding action and make decisions at the right time. The project finally aims to ensure optimal irrigating conditions to yield better crop growth.

By automating the irrigation process, it saves labor costs and the management of water resources is streamlined. The system also aims to increase productivity from the farmer's end and is cost-effective and friendly, both for people having scarce resources in rural areas or the lack of technical expertise. The project deals with the integration of IoT and email technology to enhance agricultural productivity in a manner that will favor sustainable farming practices.

1.2 ORGANIZATION OF THE PROJECT

The organization of this project is structured to ensure efficient implementation and functionality of the smart irrigation system. At the core, the NodeMCU (IoT platform) serves as the central control unit, connecting and coordinating all the sensors and communication modules. The project begins with the integration of key sensors: the rain sensor and soil moisture sensor. These sensors are responsible for detecting the environmental conditions necessary for decision-making in the irrigation process. The rain

sensor identifies rainfall in the field, while the soil moisture sensor monitors the water content in the soil to determine when irrigation is required.

Connected to the NodeMCU is the motor that controls the irrigation process. A motor driver is used to regulate the motor's operation, ensuring that it is turned on or off based on sensor inputs. The system is further enhanced with a GSM module, which enables two-way communication with the farmer. This allows the farmer to receive SMS alerts about the system's status, such as when rain is detected or when irrigation has been activated or deactivated. It also provides the farmer with the ability to remotely control the motor and inquire about its status from any location.

The system's architecture is designed to function autonomously, with minimal human intervention. The integration of sensors, NodeMCU, motor control, and cloud communication is organized to work seamlessly in real time, ensuring that irrigation is only carried out when necessary and that the farmer is always informed and in control. This approach not only reduces water wastage but also optimizes crop irrigation, leading to improved yields and better resource management.

CHAPTER 2

LITERATURE SURVEY

Yasin, Hajar M., Subhi RM Zeebaree, and Ibrahim MI Zebari. [1] Suggests the mobile of farms/homeowner can inform him if soil needs watering and in turn the farmer/householder sends an SMS message to the controller to start irrigation, and then send another SMS message to stop the irrigation after receiving an SMS message from the microcontroller that the soil saturated with water and also tells that there are probabilities to take into consideration, one of them is if the water tank is empty, which in this situation the moisture sensor inserted in the tank to inform the homeowner there is no water if empty and in turn the homeowner has sent the SMS message to microcontroller to turn ON the water pump to provide the tank with water. Another probability is rain fall; in this state, the rain drop sensor module is added to the system to stop the irrigation via SMS sent by homeowner.

Guravaiah, Koppala, and S. Srinivasulu Raju. [2] Proposes a wireless sensor based networking system to address water pumping system to crop and feed nutrients to crop. The problem is to automate the process of pumping the water to crop in a garden. Temperature and humidity sensor are used to monitor the temperature in order to initiate the water pumping system, while soil moisture sensors are used for sensing the water level and initiate the process of pumping the water to the crops in the garden. Rain sensor is used to check the rain; if rain is coming it is going to stop all the motors.

Selvi, V. Myvizhi, et al. [3] provides a system using IoT devices such as Node MCU and the several sensors like Soil Moisture, Soil Temperature, Humidity, and Rain drop detection sensor. The sensors are integrated to collect the different parameters of the field. The data are collected from the sensors can be used for the real-time monitoring of agriculture field without the presence of the Farmer in the field.

Barkunan, S. R., V. Bhanumathi, and V. Balakrishnan. [4] suggests a system which is developed based on ARM micro controller combined with GSM module to inform the rain fall level to the farmer and as well as automatically regulates the water irrigation.

Kanupuru, Priyanka, and Uma Reddy Nadig Vijayendra Reddy. [5] proposes motor protection against the dry run, which not only protects the motor from break down but also avoids the unwanted power consumption and also all the live data is in turn uploaded to the cloud so that the user can have a track of the current status of his farm at any time.

Pareja, Michael, and Argel Bandala [6] uses Fuzzy logic to improve the irrigation system

that includes three input parameters, such as soil moisture, soil temperature, and the water level. The combinations of these parameters will produce the time duration to have an efficient flow of water to the crop fields.

2.1 EXISTING SYSTEM

Existing systems for irrigation automation typically rely on basic timers or manually-controlled systems where the farmer must turn on or off the irrigation motor. These systems do not take into account real-time environmental conditions such as rainfall or soil moisture levels. In some cases, automated systems are used, but they often require regular human intervention for adjustments, which can still result in over-irrigation or under-irrigation. Current smart irrigation solutions often rely on weather data and schedules to determine when irrigation should occur, but they lack real-time feedback and communication with the farmer. While some systems integrate moisture sensors to detect soil dryness, they may not have the capability to stop irrigation during rainfall or alert the farmer when issues arise. Furthermore, many systems lack remote control features, leaving farmers unable to monitor or adjust irrigation practices from a distance. Additionally, the use of GSM or internet-based communication is often limited, meaning farmers are not notified of operational events, such as motor malfunctions or rain detection, unless they are physically present in the field. These limitations make it challenging to optimize irrigation practices and ensure the most efficient water usage, leading to potential crop damage, reduced yields, and increased costs.

2.2 PROPOSED SYSTEM

The proposed system aims to automate the irrigation process by integrating real-time environmental monitoring with remote control capabilities to optimize water usage and improve crop yield. It uses an IoT-based platform, specifically the NodeMCU, to connect and manage various sensors and devices. The system incorporates a rain sensor to detect rainfall and a soil moisture sensor to monitor the moisture level of the soil. If rainfall is detected, the rain sensor sends a signal to the NodeMCU to automatically turn off the irrigation motor, preventing over-irrigation. If the soil moisture level falls below a predefined threshold, the system will activate the motor to irrigate the crops. The farmer can control and monitor the irrigation process remotely using a GSM module, which sends SMS alerts for various events, such as rainfall detection, motor activation, or status updates. This feature ensures that farmers are always informed about the condition of their

irrigation system, even when they are away from the field. Additionally, the system allows farmers to remotely check the status of the motor and take necessary actions, such as turning it on or off, improving convenience and efficiency while minimizing the risk of crop damage due to improper irrigation. This proposed solution minimizes human intervention, reduces water wastage, and ensures that crops receive the appropriate amount of irrigation based on real-time environmental conditions.

2.3 EMBEDDED INTRODUCTION

A dedicated computer system operating inside a bigger mechanical or electrical system is called an embedded system. Embedded systems, in contrast to general-purpose computers, are created for particular purposes, frequently with real-time processing limitations. In order to monitor and regulate physical processes, these systems are usually integrated with hardware elements including sensors, actuators, and communication modules. Embedded systems are extensively employed in diverse fields such as consumer electronics, automotive systems, and medical equipment because of their dependability, efficiency, and capacity for real-time operations.The embedded system is essential to the seamless and independent operation of the wheelchair guidance system for patients with disabilities that is being suggested. The embedded system is in charge of interpreting data from many sensors, including Bluetooth beacons, RFID, and GPS, to figure out the wheelchair's location and direct it about the hospital. Moreover, it keeps an eye on obstacle detection sensors (such infrared or ultrasonic sensors) to prevent crashes and redirect as needed. By analyzing these real-time data, the embedded system makes sure the wheelchair moves securely and effectively while adapting to environmental changes.

FIG: 2.1 Embedded System

Several essential parts make up the wheelchair guidance platform's embedded system. The brains of the system are the microcontroller or microprocessor, which manages navigation and sensor data processing by putting preprogrammed algorithms into action. Actuators in the wheelchair allow it to move in response to guidance commands that come from the microcontroller.

Furthermore, communication components are added to guarantee that the wheelchair stays linked to the hospital's internal navigation system, which consists of RFID readers and Bluetooth beacons. The wheelchair is able to operate independently because of the coordinated actions of these parts, which are managed by the embedded system.In this context, an embedded system's capacity for real-time processing is among its most crucial features. For the wheelchair to operate smoothly and to protect the user, the system needs to be able to respond quickly to sensor inputs, such as seeing an obstruction or getting an updated route. For instance, the wheelchair's embedded system has to immediately halt it or change its path if a sensor picks up an obstruction in its path. In order to keep the patient updated on their progress, the system is also in charge of continuously tracking thewheelchair's position and providing audio instructions in real time. By guaranteeing safety-critical processes, the embedded system keeps navigation prompt and dependable.There are various benefits of using an embedded system in the autonomous wheelchair navigation system. Initially, it guarantees effective and instantaneous functioning, a crucial aspect of securely navigating hectic medical settings.

Because of its low power consumption, the system is perfect for transportable applications where energy efficiency is crucial, such as wheelchairs. Because embedded systems are small and reasonably priced, they can be easily included into wheelchairs without taking up extra space. They are also the best option for this project since they are highly dependable and easily adaptable for different jobs including voice help, obstacle avoidance, and navigation.

History of embedded systems

Embedded systems have their roots in the massive, costly computers of the 1950s and 60s, which were mostly utilized for data processing and scientific calculations. Early embedded systems were made to operate certain machinery, such industrial and washing machines. The Apollo Guidance Computer (AGC), which was created in the 1960s for the Apollo space missions and allowed the spacecraft to navigate and control flight, is regarded as one

of the first embedded systems.The development of embedded systems underwent a dramatic shift with the advent of microprocessors in the early 1970s. The first microprocessor to be sold commercially was the Intel 4004, which debuted in 1971 and let designers incorporate computing power into smaller devices. As a result of this breakthrough, increasingly sophisticated embedded systems were created, including the first digital watches and household appliances that used microprocessors to improve functionality and control.

Embedded systems became widely used in consumer electronics and industrial applications during the 1980s. PLCs, or programmable logic controllers, were popular in this decade and automated machinery to change production. Furthermore, the development of 8-bit microcontrollers, like the Intel 8051, reduced the cost and increased the accessibility of embedded systems for a range of uses, including home and automotive systems. Embedded systems are widely used in various industries due to the growing desire for more intelligent and responsive products.Significant advances in embedded system technologies, such as the creation of 32-bit microcontrollers and more advanced software tools, were made in the 1990s. Real-time operating systems (RTOS) also gained popularity at this time, allowing embedded computers to react instantly to events and efficiently handle several tasks at once.

The development of networking technologies and the Internet also brought about the introduction of Internet of Things (IoT) devices, which increased the range of applications for embedded systems by enabling data sharing and communication across networks.Embedded systems have grown more potent, small, and networked in the 2000s and beyond. Greater functionality and efficiency were made possible by the emergence of system-on-chip (SoC) designs, which merged several components—such as CPUs, memory, and peripherals—onto a single chip. Numerous applications, such as wearable technology, vehicle safety systems, smart home gadgets, and healthcare equipment, use contemporary embedded systems.Embedded systems may now make decisions based on real-time data and adapt to changing settings thanks to the development of machine learning and artificial intelligence. The future of embedded systems promises to offer even more innovation and integration into daily life as the demand for smart, connected devices grows, especially with the ongoing development of IoT, smart cities, and autonomous systems.

Characteristics of embedded systems

1. Dedicated Functionality:

Inside a larger system, embedded systems are made to carry out particular jobs or functions. In contrast to general-purpose computers, which are capable of handling a wide range of tasks, embedded systems concentrate on specific tasks, guaranteeing optimal performance for their intended.

2. Real-Time Operation:

A lot of embedded systems have to process inputs and generate outputs in a set amount of time because they are operating in real-time. This feature is useful for applications where time is key, including medical devices (like pacemakers) or automotive systems (like anti-lock brake systems), where a delayed response could result in catastrophic failures.

3. Resource Constraints:

Embedded systems usually face severe resource limitations, such as low memory, processing, and energy usage. Because of these constraints, rigorous hardware and software optimization is needed to ensure effective functioning without using up too many resources.

4. Interfacing with the Physical World:

Sensors and actuators are frequently used by embedded systems to communicate with the outside world. Sensors gather environmental data (such as temperature, pressure, and motion), while actuators (such as motors and valves) act on the processed data. Because of this feature, embedded systems can monitor and control external devices.

5. Reliability and Stability:

Reliability is an essential feature of embedded systems, particularly in applications where safety is a top priority, such automotive and healthcare systems. It is anticipated that these systems would function flawlessly for prolonged periods of time. To guarantee stability and robustness in their operation, they go through extensive testing and validation.

6. Low Power Consumption:

In order to prolong their operational life, many embedded systems are made for battery-operated devices. Strategies including power control, sleep modes, and effective processing algorithms are frequently used to reduce energy consumption without sacrificing functionality.

7. Software and Hardware Integration:

Typically, embedded systems consist of both software (operating systems, application code) and hardware (microcontrollers, sensors, and actuators). Specialized functionalities—where the software is particularly matched to the hardware capabilities to achieve optimal performance are made possible by this close integration.

8. Scalability and Modularity:

Although a lot of embedded systems are made for particular uses, they can also be flexible, which makes it possible to add features or updates in the future. Because of its scalability, developers can modify the system to meet changing needs without having to start from scratch.

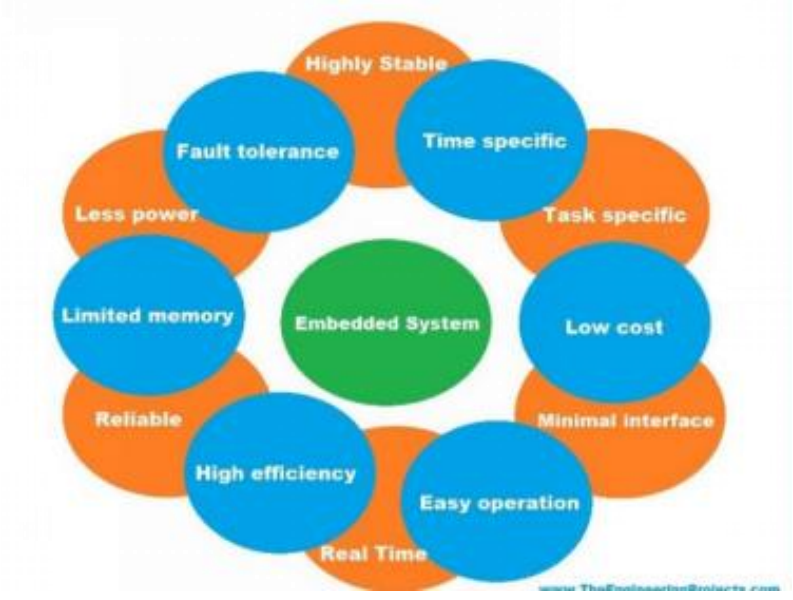

FIG: 2.2 Embedded Characteristics

9. Complexity:

There is a wide range of complexity in embedded systems, from basic devices with a single microcontroller to complicated systems with many processors and a large number of software components. The degree of functionality required and the application requirements determine the complexity.

10. User Interface:

To enable user interaction, a lot of embedded systems have a user interface. This enables users to efficiently manage and monitor the system. It might take the form of straightforward LED indications and buttons or more intricate graphical interfaces on screens.

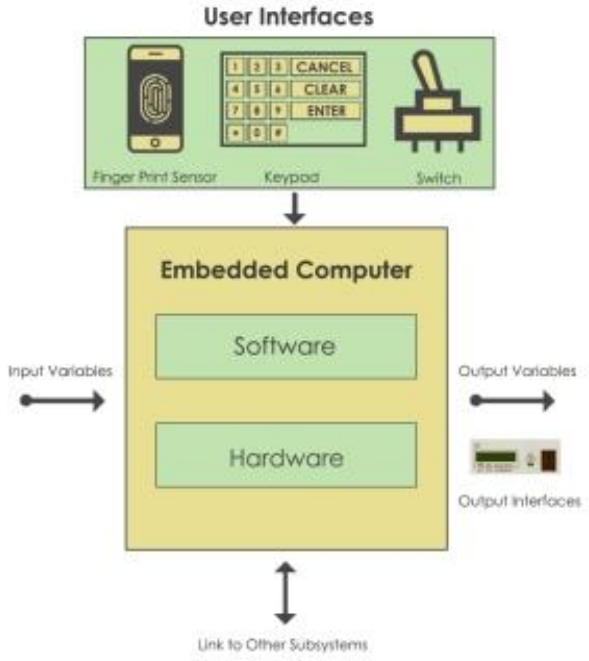

FIG: 2.3 Blocks Of Embedded System

2.3.1 Why Embedded?

Unlike general-purpose computers, embedded systems are made to do specific jobs. They concentrate on carrying out certain tasks inside a bigger system. For instance, a washing machine's microprocessor manages the machine's wash cycles.

Because of their specificity, embedded systems are very dependable for the specific task they are designed to complete.Because they are designed for a specific task, embedded systems are highly efficient in terms of power usage and performance. This is

especially crucial for battery-operated products like wearables, Internet of Things devices, and medical equipment. These systems' efficiency enables prolonged operation with little energy consumption.A lot of embedded systems are made to function in real time, reacting right away to inputs or events. This is crucial in applications like industrial machinery, vehicle safety systems, and medical devices like pacemakers where any delay could lead to failure. For these systems to function smoothly and safely, data must be processed fast and precisely.

Embedded systems work well in devices with limited space since they are frequently small and compact. Their compact size and straightforward hardware layout save expenses, which is a major benefit for home appliances and consumer electronics. These systems don't need the complexity or cost of general-purpose computing gear because they only perform particular functions.Almost every industry uses embedded systems, including consumer electronics, industrial automation, healthcare, and automobiles. They run everything, including sophisticated medical equipment, automobile control systems, and cellphones and smart home appliances. Their capacity to deliver dependable, economical, and efficient performance makes them indispensable in today's technological landscape, spanning multiple sectors.

FIG: 2.4 Embedded Systems Hardware

2.3.2 Design Approaches

An extensive examination of the requirements is the first step in the design of an embedded system.

This entails being aware of the precise duties that the system must carry out, as well as its performance standards, power goals, financial limitations, and any real-time or environmental requirements.This informs the design of the system architecture, which

includes choices for memory, I/O interfaces, communication protocols, hardware platforms (microcontroller, CPU, or FPGA), and other components. Planning also includes the software architecture, which establishes the necessity for an operating system and the organization of the tasks.Embedded system design involves simultaneous development of both hardware and software to ensure they perform flawlessly together. Software design is concerned with creating the required algorithms, control logic, and real-time processes, whereas hardware design is concerned with choosing components, creating circuits, and controlling power. Hardware and software integration is crucial, as the system must function as a cohesive unit to satisfy performance and efficiency goals.

Following design completion, software and hardware prototyping are used to put the system into operation. Thorough testing is then conducted to make sure the system satisfies all criteria, including real-time performance and power efficiency. During testing, any flaws are found and resolved, and the system is adjusted for better performance and dependability. To ensure that it satisfies industry requirements for dependability and safety, the embedded system may additionally go through certification in important applications like automotive or medical systems.

Steps in Embedded System Design Process

The difference steps in the embedded system design flow/flow diagram include the following

1. Need:

This step entails determining if a certain embedded system is necessary. Usually, it entails identifying an issue or chance that an embedded solution can solve.

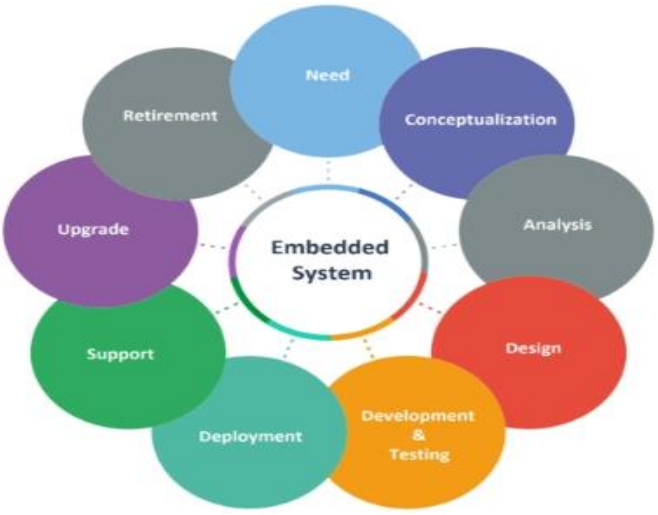

FIG: 2.5 Embedded Design Process Steps

2. Conceptualization:

Rough designs and preliminary concepts are created during this stage. Based on the defined need, the embedded system concept is developed, taking into account possible solutions and viability.

3. Analysis:

After the notion is formed, a thorough investigation is carried out. This include researching specifications for things like cost, performance, power consumption, and functionality as well as any limitations like size or operating environment.

4. Design:

Following the conclusion of the investigation, the real design work starts. This comprises both hardware and software design, including selecting the microcontroller or CPU, peripherals, and establishing the system architecture.

5. Development & Testing:

During this stage, hardware prototyping and coding are used to implement the system. To make sure the system satisfies the necessary requirements, testing is done. Real-time performance assessments and functional testing are both included in this.

6. Deployment:

The embedded system is put into use for practical purposes following a successful testing phase. This might entail putting the software on the hardware and producing the system in large quantities.

7. Support:

After the system is put into place, it needs continuous assistance. This include bug fixes, software updates, and making sure the system keeps working the way it was designed to.

8. Upgrade:

Upgrades are frequently required due to shifting requirements and system performance. These can entail updating the system with new features, improving performance, or adjusting it to new developments in technology.

9. Retirement:

An embedded system may eventually come to the end of its useful life because of obsolescence or the emergence of better alternatives. After that, the system is retired and might be replaced with a fresh approach.

Software design is described using architectural description language.

• Data structure and hierarchy;

• Software procedure;

• Control hierarchy;

• Structure division.

User requirements, environment analysis, and system function are all important factors in user interface design. For instance, we take care of other factors on a mobile phone in order to lower the power consumption of mobile phones.

WHO has identified formulations for local preparation of alcohol-based handrubs, which will aid nations and healthcare facilities in implementing systemic change and establishing them as the gold standard for hand hygiene in healthcare. economic, cultural, safe, and logistical.

TABLE: 2.1 Embedded System Design Software Development Activities

Stage	Objective	Key Activities
Analysis of Requirements	Recognize system requirements and specify software needs.	Specify features and time constraints in real time. Identify hardware-software interactions Record the control mechanisms, outputs, and inputs.
Design of Software Architecture	Plan the software's overall structure. Specify the components of the software and their functions.	Create interfaces for the modules. Select operating system (if needed) Create a schedule for design tasks.
Development of Firmware	Develop low-level code to interface with hardware.	Create hardware abstraction layers (HAL), manage memory, power, and bootloader, and write drivers for hardware components.
Development of Application Software	Put high-level features and control logic into practice.	Provide control systems and user interfaces. Put communication protocols into action. Create systems for error handling and fault detection.
Instantaneous Programming	Verify the system satisfies the real-time requirements.	Use multitasking and task scheduling. Create ISRs, or interrupt service routines. Make code more deterministic in real time.
Testing and Troubleshooting	Make sure the software works correctly and	Conduct system-level testing and hardware-in-the-loop (HIL) tests Use

	integrates with the hardware.	debugging tools (ICE, JTAG)
Enhancement of Performance	Software should be optimized for power, memory, and real-time effectiveness.	Reduce memory footprint, maximize power consumption, and enhance task execution and latency for real-time performance
Updating and Maintaining Software	After deployment, give continuing software support	Carry out over-the-air (OTA) updates; Troubleshoot problems and supply fixes; Uphold version control and monitor.

Embedded systems are used in a variety of technology across industries. Some examples include:

Embedded systems are vital to many different industries because they improve automation and offer necessary functions. They are essential to infotainment systems, anti-lock braking systems (ABS), and engine control units (ECUs) in the automotive industry, guaranteeing both driver safety and peak vehicle performance. Embedded systems are used in consumer electronics like wearables, tablets, smart TVs, and smartphones to improve user experiences through touch sensitivity, communication, and real-time data processing.Embedded systems are used in home automation to control smart devices such as security cameras and thermostats, allowing for remote control and automation of household tasks. Embedded systems are used by the healthcare sector in medical devices such as insulin pumps and pacemakers to enable real-time monitoring and potentially life-saving interventions. SCADA systems and programmable logic controllers (PLCs) in industrial automation use embedded technology to effectively monitor and automate industrial processes.To manage data traffic and network connectivity, the telecommunications industry uses embedded systems in routers, modems, and mobile communication towers. They are essential to flight control systems, radar technology, and real-time data processing for navigation and surveillance in aerospace and defense.

Smart metering and renewable energy systems optimize resource consumption and management, which benefits the energy sector. Furthermore, embedded systems are used in robotics to power autonomous robots and drones, allowing for real-time decision-making on challenging jobs. Embedded technology is used in retail point of sale (POS)

systems and automated vending machines to process transactions and manage inventories. Lastly, embedded systems in transportation control railway signalling and traffic control, improving the efficiency and safety of transit systems. Overall, embedded systems significantly contribute to automation, efficiency, and enhanced functionalities across these diverse applications.

FIG: 2.6 Applications of Embedded Systems

2.3.3 Combination of Logic Device

Digital systems and embedded technologies are based on a combination of logic devices that provide the framework for processing, managing, and controlling tasks. To carry out basic Boolean operations, basic logic gates such as AND, OR, and NOT are required. These gates build more intricate circuits that can carry out intricate tasks when combined in different configurations. In order to effectively route data and manage signals in systems, multiplexers and demultiplexers are used to distribute or choose data among various pathways.Next, discrete data bits are stored in memory using flip-flops and latches. Particularly in sequential logic systems, they are indispensable for tasks requiring the synchronization of actions or the maintenance of a state.

Conversely, shift registers play a crucial part in data conversion procedures like converting serial data into parallel form and vice versa. They enable the movement of data left or right. In digital systems, counters are used for timing, frequency division, and the generation of control signals. They are frequently implemented using flip-flops, track sequences, or events.

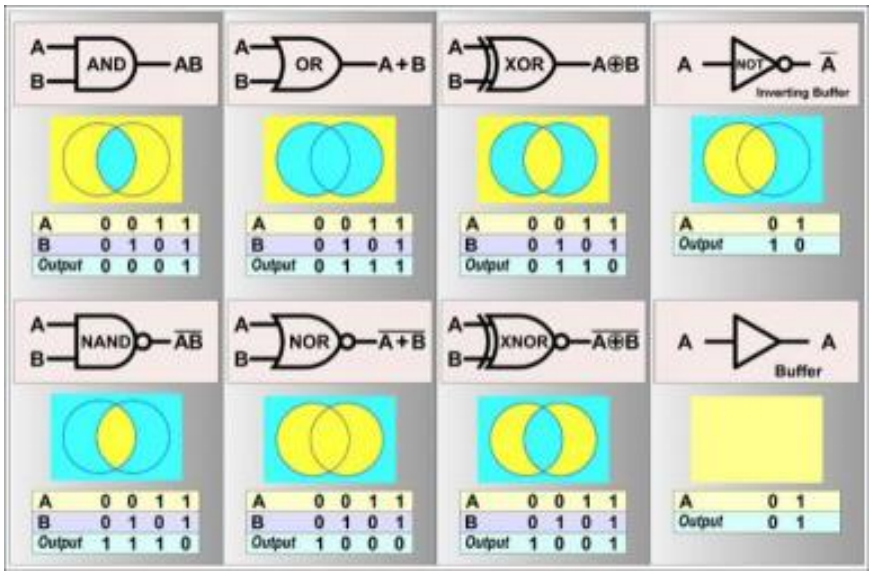

FIG: 2.7 Logic Gates

The transformation of encoded signals into defined outputs and vice versa is handled by encoders and decoders. These devices are frequently used in communication systems to interpret or compress signals, as well as for memory addressing. Engineers can create unique digital circuits using programmable logic devices (PLDs), like FPGAs, by configuring logic blocks to meet particular needs. This flexibility and reconfigurability can be used for a variety of applications.An essential part of microprocessors, the arithmetic logic unit (ALU) combines several logic functions to carry out arithmetic operations like addition and subtraction as well as logic operations like AND, OR, and NOT. To enable data storage and retrieval, logic circuits are combined with memory devices such as RAM (for temporary data storage) and ROM (for permanent instructions).Furthermore, buffers and bus drivers control signal integrity and data flow, especially in larger systems where data must be moved between modules or communication buses. Real-time embedded systems require the synchronization of operations, which is made possible by the use of clocks and timers. Lastly, communication between the analog and digital realms is made possible by analog-to-digital converters (ADCs) and digital-to-analog converters (DACs).

For applications like controlling motors or producing audio signals, DACs convert processed digital data back into an analog form.

ADCs enable systems to process real-world signals like temperature or sound by converting them into a digital form.All things considered, these logic devices are integral to the functioning of contemporary digital technologies because of their ability to integrate complex functions such as data processing, signal control, communication, and memory management in embedded systems. The foundation for the design of embedded systems in a variety of industries, including automotive, telecommunications, and healthcare, is laid by this combination, which allows systems to execute everything from simple calculations to real-time decision-making.

CHAPTER 3
HARDWARE REQUIREMENTS

3.1 HARDWARE

Embedded system hardware

The hardware of an embedded system is made up of a number of integrated parts that cooperate to carry out particular tasks within time constraints. The microcontroller (MCU) or microprocessor (MPU), which functions as the system's brain, is at the heart of every embedded system. The microcontroller is an all-in-one task control solution because it usually consists of the CPU, memory, and peripherals on a single chip. A microprocessor might be employed in more intricate systems, necessitating peripherals and external memory to provide more sophisticated processing powers.Memory, which includes ROM, which houses the firmware or software required for the system's fundamental operations, and RAM, which momentarily stores data while the system is functioning, is one of its most important components. Software that might need to be updated is frequently stored in flash memory, which provides embedded systems with a non-volatile storage option. Another essential piece of hardware is the power supply, which either draws energy directly from the grid or from batteries to power the device. Power management circuits are typically included in embedded systems in order to maximize energy efficiency.

In embedded systems, timers and counters are essential for enabling time-sensitive operations, scheduling work, and controlling task execution delays. These parts are necessary for systems like robotics and industrial automation that need to function in real time. Interfaces for input/output (I/O) allow the system to communicate with external parts like actuators and sensors. GPIO pins, SPI, I2C, UART, and USB are examples of common I/O interfaces that make it easier to communicate with and control different hardware parts. Sensors are essential components of embedded systems that are meant to interact with the real world. They take measurements of things like motion, light, pressure, temperature, and other environmental variables and translate them into signals that the microcontroller can use. Actuators are used in systems to translate electrical signals into mechanical actions, like servo adjustments, valve openings, and motor movement. Real-time monitoring and response of an embedded system to its surroundings is made possible by the integration of sensors and actuators.

An essential part of converting analog signals from sensors into digital data that the microcontroller can process is the analog-to-digital converter, or ADC. A Digital-to-Analog Converter (DAC), which transforms digital outputs into analog signals for controlling motor speed or powering external devices like speakers, is also included in many systems. Systems that need to interface with the analog, real world require these converters.

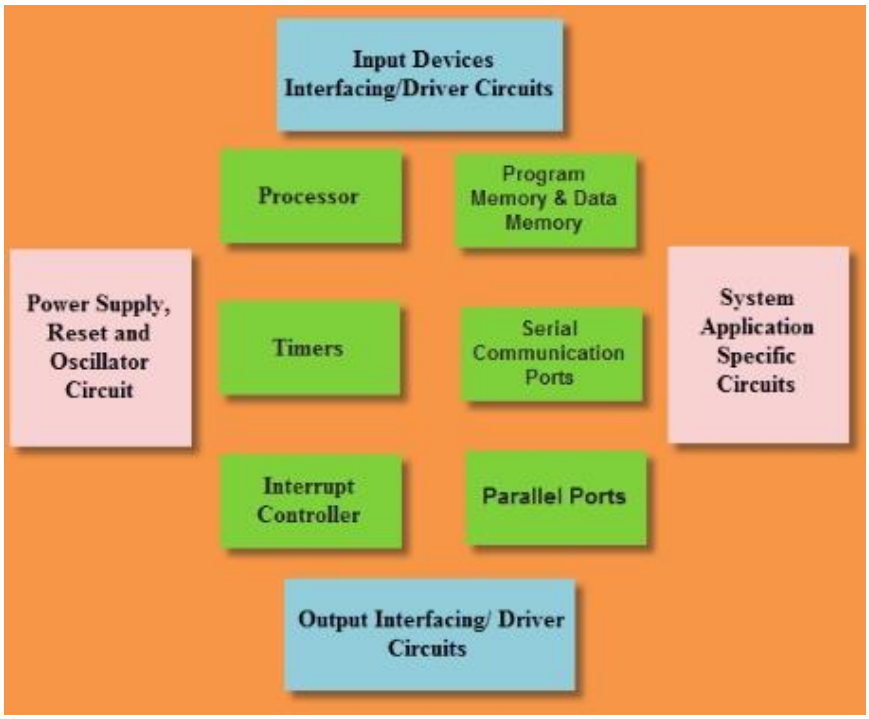

FIG: 3.1 Embedded Systems Hardware Block Diagram

When it comes to facilitating data exchange between an embedded system and other networks or devices, communication interfaces are crucial. While wireless interfaces like Wi-Fi, Bluetooth, Zigbee, and LoRa enable remote control, monitoring, and communication, particularly in IoT devices, wired interfaces like Ethernet, CAN Bus, and RS-232 offer reliable, fast communication in industrial settings.The display unit, which includes LCD or OLED screens and enables users to view system outputs, is another essential component of embedded systems hardware. In applications requiring real-time feedback or data visualization, such as consumer electronics, industrial machinery, or medical devices, these displays are especially helpful. These screens can occasionally be used as touchscreen interfaces, providing input and output capabilities. All of the hardware components are integrated and connected on the printed circuit board (PCB), which is made up of copper traces connecting each component to the other. When it comes to

minimizing electrical interference and maximizing system performance—particularly in high-speed or high-power applications—the PCB layout is essential.

A real-time clock (RTC) is used in systems that need precise timing in order to keep track of time even when the system is powered down. This is especially crucial for systems like surveillance cameras or time-based alarms that record information or take action according to a schedule. Additionally, some embedded systems have fans or heat sinks for cooling, particularly those with high-performance processors that produce a lot of heat.In embedded systems, security modules are becoming more and more crucial, especially for network-connected devices. These modules, which include Trusted Platform Modules (TPMs), guarantee encryption, safe booting, and defense against tampering or unwanted access. They are essential for applications such as cloud-connected IoT devices, medical devices, and financial systems.

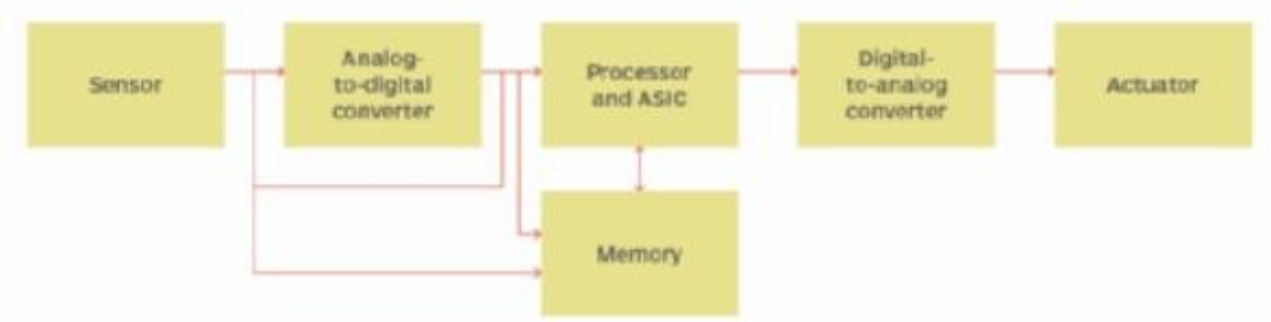

FIG: 3.2 Basic Embedded Structure

Lastly, embedded systems that produce a lot of heat occasionally employ cooling mechanisms like heat sinks and fans, particularly in applications that call for high-speed processing or high-power operation. Maintaining the system at ideal temperatures promotes longevity and dependability.

Typically, an embedded system hardware configuration consists of the following blocks:

1. Processor (microprocessor/microcontroller)
2. RAM, ROM, and Flash memory
3. Devices for Input
4. Devices of Output
5. Counters and Timers

6. Interfaces for Communication
7. Peripherals of the Power Supply (Sensors, Actuators)
8. Bus System

1. Processor (microprocessor/microcontroller)

A microprocessor (MPU) or microcontroller (MCU) is the central component of any embedded system. Because the MCU combines a CPU, memory, and I/O peripherals onto a single chip, it is an affordable option for application control. On the other hand, an MPU is usually utilized in more intricate systems that demand more processing power and usually requires external memory and peripherals.The MCU or MPU's CPU (Central Processing Unit) is in charge of managing data, carrying out instructions that have been stored in memory, and overseeing the system's overall operation. Low power consumption is a feature of some processors, which is crucial for battery-powered devices.

2. Recollection

An embedded system's memory holds both the data and the operating code needed to perform its functions. There are various kinds of memory.Random Access Memory (RAM): During operation, volatile memory is used to store temporary data. It is quick and useful for data manipulation, program execution, and variable storing.Read-Only Memory, or ROM, is a type of non-volatile memory used to store operating software or firmware. Updates and rewrites are possible with EEPROM (Electrically Erasable Programmable ROM) and PROM (Programmable ROM), two types of ROM variations.

Non-volatile memory that can store programs that can be updated or changed is called flash memory. Due to its rewritable nature, firmware storage is one of its common uses.

3. Input Sources

In embedded systems, input devices facilitate the system's ability to obtain data from external sources. These may consist of:

Switches or keypads: Used to enter data manually by users.

Sensors: These are devices that translate physical phenomena (such as light, pressure, temperature, and so on) into electrical signals the system can understand.Communication

ports, such as USB, Ethernet, or UART for serial communication, are used to receive input from other systems or devices.

4. Devices for Output

The system's output devices allow it to communicate with external hardware or send commands:

Displays (LCD/OLED): Give users visual cues.

LED Indicators: Low-tech lights that display the status of the system.

Actuators: They are used to translate electrical signals into mechanical movements, like opening a valve, moving a robotic arm, or running a motor.

Speakers: Produce sound from digital audio signals.

5. Counters and Timers

Timers and counters are frequently used by embedded systems to carry out time-sensitive operations. These hardware components are in charge of scheduling events, producing exact time delays, and measuring intervals of time. In real-time systems where tasks must be completed on a regular basis or within tight time constraints, they are essential.

6. Interfaces for Communication

Embedded systems make use of a variety of communication interfaces to exchange data with external devices or other systems, including:Microcontrollers can be connected to peripherals like sensors or memory devices via the synchronous communication protocol known as SPI (Serial Peripheral Interface).

Another synchronous communication protocol that's frequently used to connect sensors and microcontrollers is called I2C (Inter-Integrated Circuit).The Universal Asynchronous Receiver-Transmitter, or UART, is a serial communication interface that is typically used for serial communication with external devices. It allows data to be sent and received asynchronously.Wireless Interfaces: These enable wireless communication between embedded systems, particularly in Internet of Things (IoT) applications. Examples of these interfaces are Bluetooth, Wi-Fi, and Zigbee.

7. Energy Source

The power supply plays a critical role in supplying the voltage and current required for the embedded system to operate. It might be:

Battery-powered: Frequently found in low-power or portable electronics.

AC-powered: Suitable for stationary or larger systems.

Power management circuits can be incorporated into embedded systems to lower energy consumption and prolong the life of battery-operated devices.

8. Peripherals: Actuators and Sensors

Sensors: These devices identify and quantify various environmental factors, such as light, motion, humidity, pressure, and temperature. Sensor data is sent to the processor, which processes, interprets, and acts upon it.

Actuaries: Transform mechanical motion into electrical energy. Relays, for instance, turn on and off electronic devices, while motors move a robotic arm.

9. Bus System

All of the parts are connected by the bus system, which also enables communication between the peripherals, memory, CPU, and I/O devices. It serves as the embedded system's central nervous system. Among the bus types are:

Data Bus: Transfers information between peripherals and the CPU.Memory addresses that the processor will access to read or write data are transported via the address bus.

Control Bus: Sends control signals to other components to synchronize their actions.Despite their differences, an embedded system's hardware components cooperate to carry out particular tasks effectively and dependably.

Serving as the brain, the processor uses I/O interfaces to interface with input and output devices and manages data that is kept in memory. Timers facilitate the control of time-sensitive operations, and communication interfaces allow the system to communicate with networks or other external devices. The power supply and peripheral devices round out the system by supplying the energy required for operation and facilitating sensor and actuator-based real-world interaction.Simple home appliances to sophisticated industrial

machinery can all be equipped with embedded systems, which can carry out specific tasks in a variety of applications by combining all these parts in a compact and effective way.

3.2 NODE MCU

NodeMCU is an open-source LUA based firmware developed for the ESP8266 wifi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board/kit i.e. NodeMCU Development board.

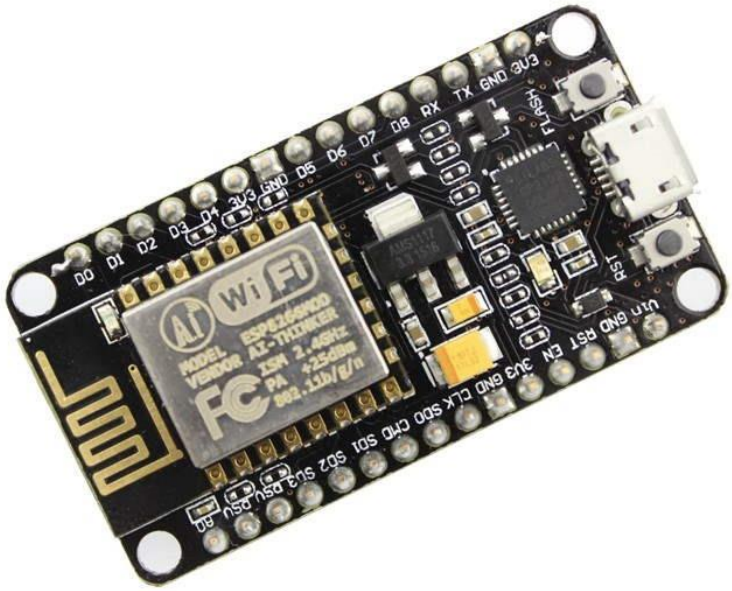

FIG: 3.3 Node MCU

Since NodeMCU is an open-source platform, its hardware design is open for edit/modify/build. NodeMCU Dev Kit/board consists of ESP8266 wifi enabled chip. The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol.

NodeMCU Dev Kit has **Arduino like** Analog (i.e. A0) and Digital (D0-D8) pins on its board. It supports serial communication protocols i.e. UART, SPI, I2C, etc. Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards, etc.

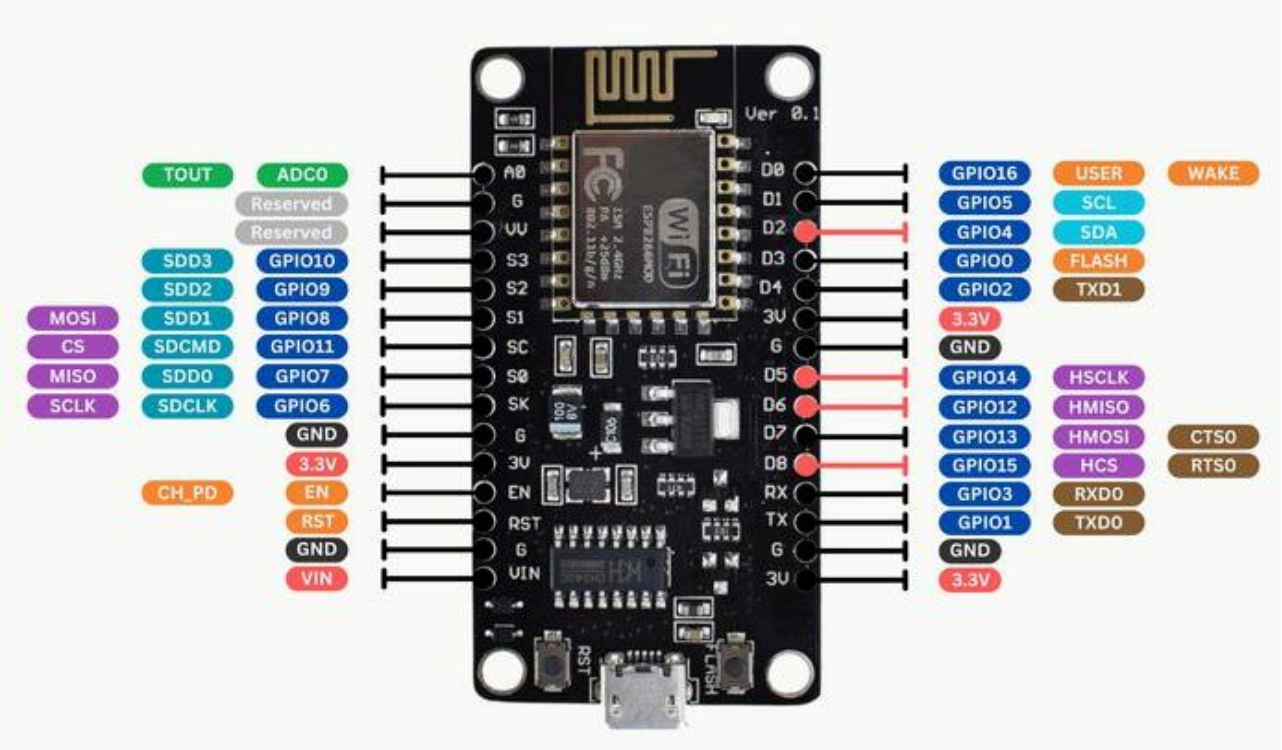

FIG: 3.4 Esp8266 Pin Diagram

Within the ESP8266, a diverse range of pins exist, each serving distinct purposes critical to the operation of microcontroller. These include:

1. Power Pins (VCC, GND)

VIN or **VCC (3V)** is the power input pin, and it requires a stable **3V supply**. The **GND** pin serves as the reference point for the voltage levels within the ESP8266 circuit.

2. General Purpose Input/Output (GPIO) Pins

The ESP8266 features **17 GPIO pins** labeled **GPIO0** to **GPIO15** and **GPIO16**. These pins are versatile and can be configured as either input or output, depending on the application's needs. Each GPIO pin has specific capabilities and can perform multiple functions

3. Special Function Pins

- **ADC (A0):** The ESP8266 includes a **10-bit** analog-to-digital converter (ADC) on the **A0 pin**, allowing the microcontroller to read analog signals from sensors such as

temperature or light sensors. The input range is typically 0 to 1V, so a voltage divider may be required to scale higher voltages.

- **RST (Reset):** This pin is used to reset the microcontroller. Pulling this pin **LOW** will reset the ESP8266, causing it to restart the current program from the beginning.
- **EN (CH_PD):** The enable pin must be pulled **HIGH** (connected to 3V) for the ESP8266 to run. Pulling it **LOW** puts the chip into a power-down state.

4. Power Management Pins

GPIO16 (D0) pin has a special role in power management. It is often used to wake the ESP8266 from deep sleep mode by connecting it to the RST pin. This feature is particularly useful in **battery-powered applications** where power conservation is crucial.

5. SPI Interface

The Serial Peripheral Interface (SPI) is a communication protocol commonly used for interfacing the ESP8266 with flash memory, sensors, and other modules. **GPIO12 (MISO)**, **GPIO13 (MOSI)**, **GPIO14 (SCLK)**, and **GPIO15 (CS)** pins are dedicated to SPI communication, but they can be repurposed if SPI is not being used.

6. I2C Interface

GPIO4 (SDA) & **GPIO5 (SCL)** pins support I2C communication, a popular protocol for interfacing with multiple sensors and modules using just two wires. These I2C pins allow multiple devices to communicate with the ESP8266, making it ideal for complex IoT projects.

7. UART Interface

The Universal Asynchronous Receiver/Transmitter (UART) interface is used for serial communication, essential for debugging and flashing firmware. **GPIO1 (TX pin)** & **GPIO3 (RX pin)** are typically connected to a USB-to-serial adapter when programming the ESP8266.

8. PWM Pins

GPIO12, **GPIO13**, and **GPIO14** pins support Pulse Width Modulation (PWM), which is used to control the brightness of LEDs, speed of motors, and other similar applications. The ESP8266 can generate PWM signals on these pins with up to **10-bit resolution**.

9. External Interrupts

GPIO0, **GPIO2**, **GPIO4**, **GPIO5**, **GPIO12**, **GPIO13**, **GPIO14**, and **GPIO15** pins can also be used as external interrupt sources, allowing the ESP8266 to respond to external events such as a button press.

10. Deep Sleep Wakeup Pin

GPIO16 (D0) pin is connected to the RST pin to wake the ESP8266 from deep sleep mode. This pin is crucial for projects where power efficiency is important, such as battery-operated IoT devices.

Applications of NodeMCU:

1. **IoT (Internet of Things):**NodeMCU is widely used in IoT applications due to its Wi-Fi capabilities. It can connect devices like sensors, actuators, and other IoT components to the internet for remote control and monitoring.
2. **Home Automation:**NodeMCU can be used to control home appliances remotely using Wi-Fi. For instance, lights, fans, doors, and thermostats can be controlled via a mobile app or a web interface..
3. **Smart Agriculture:**NodeMCU can be connected to sensors to monitor environmental conditions like soil moisture, temperature, and humidity. The data can be uploaded to the cloud for real-time analysis.
4. **Wearable Devices:**By integrating NodeMCU with wearable technology, you can create systems for health monitoring like heart rate sensors, fitness trackers, or sleep monitoring devices.
5. **Wireless Communication Systems:**NodeMCU can be used for creating low-cost wireless communication systems that enable remote data transmission and reception.

3.3 DHT 11 SNESOR

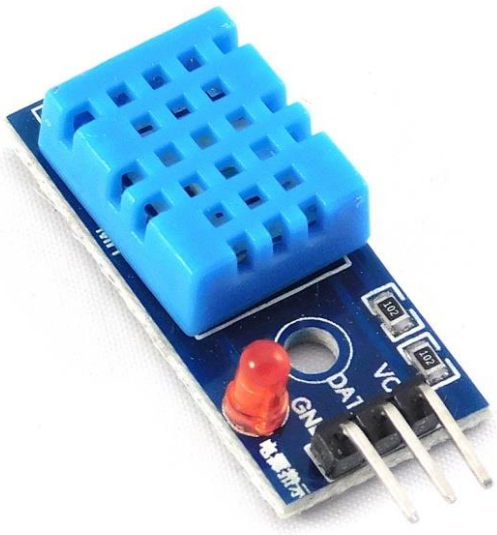

FIG: 3.5 Dht 11 Sensor

The **DHT11 sensor** is a popular digital sensor used for measuring temperature and humidity. It's commonly used in various applications, including weather stations, home automation systems, and environmental monitoring. The sensor is inexpensive, easy to use, and provides reliable readings for many DIY and small-scale projects.

Applications of the DHT11 Sensor:

1. HVAC (Heating, Ventilation, and Air Conditioning) Systems: The DHT11 can be integrated into HVAC systems to monitor and control the temperature and humidity levels of different rooms or areas in a building.

2. Smart Agriculture and Precision Farming: In agriculture, the DHT11 sensor can be used in automated irrigation systems, greenhouses, or soil moisture sensors.

3. Health and Medical Devices: DHT11 can be used in applications that monitor air quality and environmental conditions in hospitals or medical devices, such as in incubators for newborns.

4. Storage and Warehousing: In warehouses or storage facilities for sensitive goods like food, pharmaceuticals, or electronics, the DHT11 sensor can be employed to ensure that

temperature and humidity levels stay within a safe range.

5. Indoor Air Quality Monitoring Systems: The DHT11 sensor can be used in air quality monitoring systems to measure the indoor climate in homes, offices, or schools.

6. Smart Fan Systems: DHT11 sensors can be used in automatic fan systems for homes or server rooms.

7. Educational and DIY Projects: The DHT11 sensor is popular in educational kits for students learning about sensors and environmental monitoring.

3.4 RAIN DETECTOR SENSOR

A **rain detection sensor** is a device used to detect the presence of rain. It works by sensing moisture levels or changes in the environment when it rains and triggering an action, such as activating a wiper, stopping an irrigation system, or sending an alert to the user. These sensors are widely used in various applications, including automotive systems, home automation, irrigation systems, and weather monitoring.

FIG: 3.6 Rain Sensor

Applications of Rain Detection Sensors:

1. **Smart Home Automation:**Automatically closing windows or roofs when rain is detected to protect interiors from water damage.

2. **Smart Irrigation**: Automatically stopping irrigation systems when it detects rain, thus saving water and preventing overwatering.

3. **Agriculture and Farming:**Integrating rain sensors with irrigation systems to prevent watering during rainfall, conserving water and ensuring crops receive the right amount of moisture.

4. **Soil Moisture Monitoring**: Helps farmers determine when additional irrigation is needed based on rainfall data.

5. **Weather Stations:**Used in personal or professional weather stations to monitor rainfall and help provide weather forecasting information.Often used alongside other meteorological instruments to measure total rainfall over time for weather analysis.

3.5 SOIL MOISTURE SENSOR

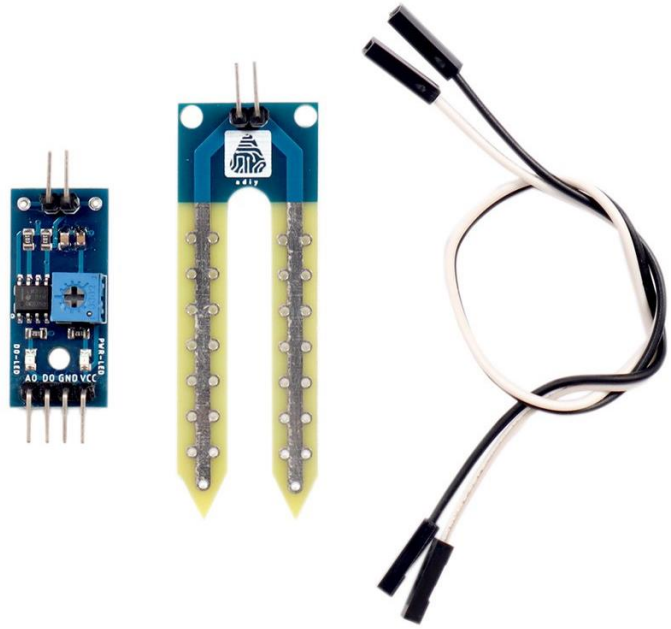

FIG: 3.7 Soil Moisture Sensor

The soil moisture sensor is one kind of sensor used to gauge the volumetric content of water within the soil. As the straight gravimetric dimension of soil moisture needs eliminating, drying, as well as sample weighting. These sensors measure the volumetric water content not directly with the help of some other rules of soil like dielectric constant, electrical resistance, otherwise interaction with neutrons, and replacement of the moisture content.

Applications of Soil Moisture Sensors:

1. **Agriculture and Farming:** Used to automate irrigation systems in agriculture, ensuring crops are watered only when the soil moisture drops below a certain threshold. This prevents both over-watering and under-watering, promoting healthy plant growth and maximizing crop yields.

2. **Precision Agriculture**: In modern precision farming, soil moisture sensors are used to monitor field conditions in real-time, optimizing irrigation and improving resource management for large-scale farms.

3. **Smart Irrigation Systems:** Soil moisture sensors can be integrated into smart garden irrigation systems to maintain optimal moisture levels, preventing the need for constant manual watering.

4. **Landscape Management**: In commercial and residential landscapes, these sensors are used to automate watering systems for lawns, gardens, and flowerbeds, helping maintain optimal soil conditions.

5. **Drought Monitoring**: Soil moisture sensors are used in environmental monitoring to assess drought conditions, helping governments and organizations manage water resources in times of water scarcity.

6. **Flood Prevention**: These sensors can also be part of flood prevention systems, providing early warnings of excessive moisture levels in the soil, which could lead to flooding in certain areas.

7.

CHAPTER 4

SOFTWARE REQUIREMENTS

4.1 SOFTWARE TOOLS

Embedded firmware refers to specialized software designed to run on embedded systems. These systems are typically hardware devices that are not primarily intended for general-purpose computing but rather to perform a dedicated function, such as microcontrollers, sensors, and other electronics. Firmware is typically stored in non-volatile memory (like flash memory) within the device and acts as the interface between the hardware and higher-level software.

Key Characteristics of Embedded Firmware:

1. **Direct Control of Hardware**: Firmware is responsible for directly controlling the hardware's operation, such as configuring sensors, controlling motors, or managing communications protocols.
2. **Real-Time Operation**: Many embedded systems operate in real-time, meaning firmware must ensure timely responses to external events. This is especially important for systems with strict timing constraints, such as industrial control systems or automotive applications.
3. **Resource Constraints**: Embedded systems often have limited processing power, memory, and storage. The firmware is typically optimized for low resource usage, often written in languages like C or assembly.
4. **Long Lifecycle**: Unlike general-purpose software that may change frequently, embedded firmware tends to have a much longer lifecycle. Once deployed, it might not be updated regularly, although some devices support over-the-air (OTA) firmware updates.
5. **Device-Specific**: Firmware is highly tailored to the specific hardware of the device. For example, firmware written for a microcontroller may not work for another microcontroller unless they share similar architectures or peripherals.

Types of Embedded Firmware:

1. **Bootloaders**: Small programs that initialize the hardware when the device is powered on and prepare it to load the main firmware.
2. **Device Drivers**: Code that manages hardware peripherals such as sensors, actuators, displays, etc. It provides an abstraction layer between the hardware and the application software.
3. **Operating System/RTOS (Real-Time Operating System)**: For more complex embedded systems, firmware can include an operating system, such as FreeRTOS or Embedded Linux, to handle multitasking, task scheduling, and resource management.
4. **Application Code**: The actual software that implements the main functionality of the embedded device, such as a thermostat control algorithm, communications protocol handling, or a user interface.

Development Process for Embedded Firmware:

1. **System Design**: Define the requirements of the embedded system, including the hardware, functionality, and performance expectations.
2. **Hardware Integration**: Develop or select the appropriate microcontroller or embedded processor, sensors, actuators, and other peripherals.
3. **Firmware Coding**: Write the firmware in a language appropriate for the hardware, typically C, C++, or assembly for low-level control. Sometimes, higher-level languages like Python (via MicroPython) or Lua are used for simpler systems.
4. **Testing**: Test firmware functionality both in isolation (unit testing) and in integration with the hardware. Emulators or simulators may also be used before actual hardware is available.
5. **Debugging**: Tools like JTAG debuggers, in-circuit emulators (ICE), or serial communication are commonly used to step through code and inspect the system's state.
6. **Deployment & Updates**: Flash the firmware to the embedded device and, if necessary, provide means for remote updates (OTA updates) via wireless communication (Wi-Fi, Bluetooth, etc.).

Examples of Embedded Firmware Applications:

1. **Consumer Electronics**: Smart TVs, digital cameras, smart thermostats, fitness trackers.

2. **Automotive**: ECU (Engine Control Unit) firmware for controlling engine, transmission, ABS, etc.
3. **Industrial Automation**: PLCs (Programmable Logic Controllers), SCADA systems, industrial robots.
4. **Medical Devices**: Pacemakers, infusion pumps, portable diagnostic devices.
5. **IoT Devices**: Smart home devices like lights, thermostats, and security cameras.

4.2 ARDUINO IDE

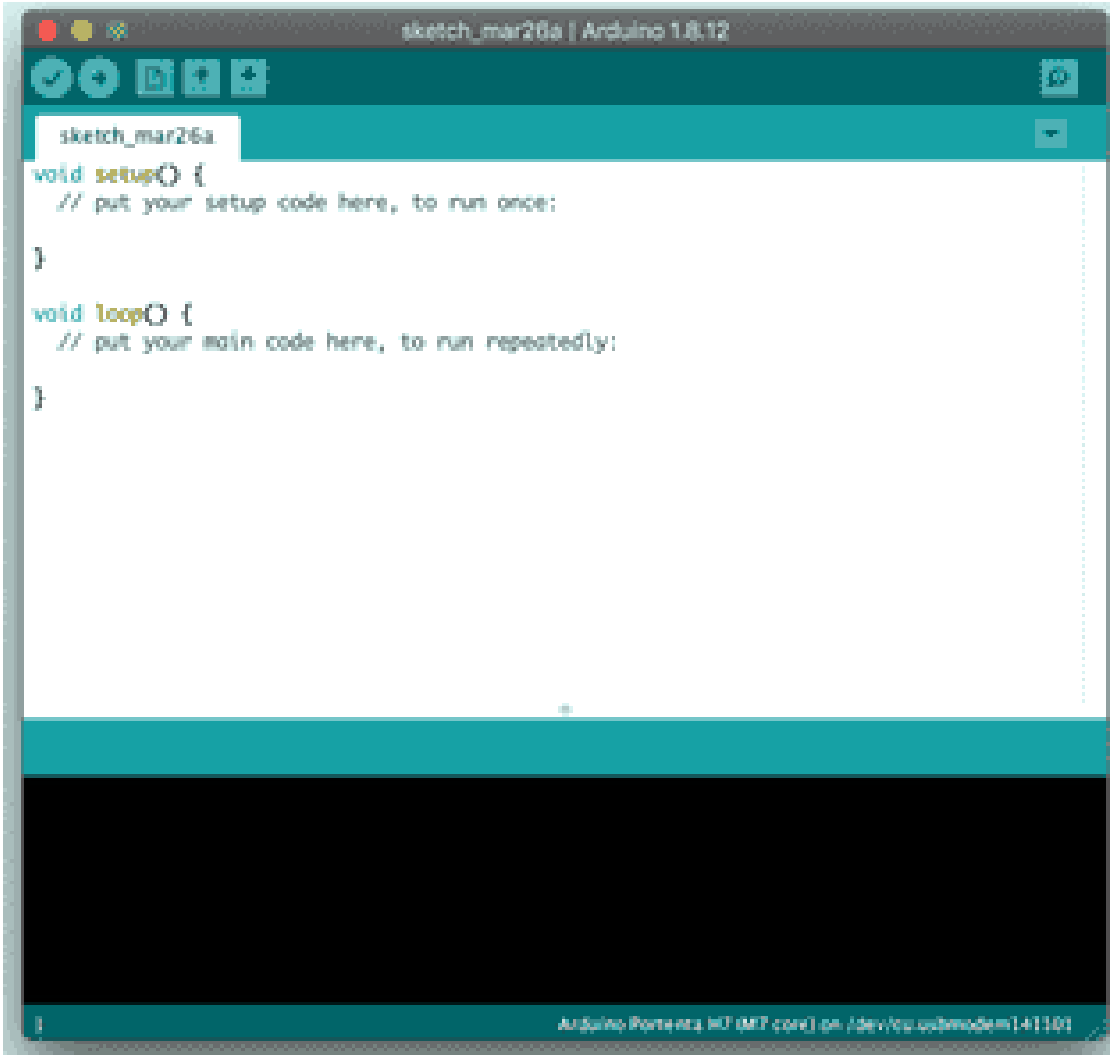

FIG: 4.1 Arduino IDE

The Arduino IDE (Integrated Development Environment) is a popular software tool used to write, compile, and upload code to Arduino boards and other compatible microcontroller-based devices. It simplifies embedded development, especially for

beginners and hobbyists, and has become a key platform for rapid prototyping and educational projects.

Key Features of the Arduino IDE:

1. **Code Editor**: The IDE provides a simple text editor where you can write your code (referred to as a *sketch* in Arduino terminology). The editor highlights syntax to make the code easier to read and debug.
2. **Libraries**: Arduino has an extensive ecosystem of pre-built libraries for interacting with a wide range of hardware like sensors, motors, displays, and communication protocols (e.g., I2C, SPI, Bluetooth, etc.). Libraries simplify hardware integration and reduce development time.
3. **Board Manager**: The IDE allows you to select from a variety of Arduino boards (e.g., Arduino Uno, Mega, Nano, and many others). You can also add support for other microcontrollers and development boards (e.g., ESP8266, ESP32, Teensy) using the *Board Manager*.
4. **Compiler and Uploader**: Once you've written your sketch, the Arduino IDE compiles the code (turns it into machine-readable instructions) and uploads it to the microcontroller on your Arduino board via USB or other communication methods. The IDE uses the GCC toolchain for compiling code.
5. **Serial Monitor**: A built-in Serial Monitor allows you to view data sent from your Arduino board to your computer (such as sensor readings or debugging information). You can also send data from the computer to the Arduino for communication purposes.
6. **Sketches**: Programs written for Arduino are called *sketches*. These are typically written in a simplified version of C/C++ and must include two main functions:

    - setup(): This function is executed once when the program starts. It's used to initialize hardware, set pin modes, and configure other setup tasks.
    - loop(): This function is continuously executed after setup() is finished. It holds the main logic of the program and runs repeatedly as long as the Arduino is powered.

Arduino IDE Workflow:

1. **Writing Code (Sketch)**: In the Arduino IDE, you'll write your code using the C++-like syntax, typically using the predefined functions setup() and loop().

2. **Select Board and Port**: You need to select the correct Arduino board model (e.g., Arduino Uno, Arduino Mega) and the correct communication port to which the board is connected.
3. **Compile Code**: The IDE compiles your code, turning it into machine code that can be understood by the microcontroller on the board.
4. **Upload Code**: After compiling, you upload the code to the board by clicking the "Upload" button in the IDE. The code is transferred via USB or another method (depending on the board).
5. **Monitor Output**: Once the code is running on the Arduino, you can monitor it using the Serial Monitor. This is especially useful for debugging or getting feedback from sensors and other peripherals.

CHAPTER 5

PROJECT IMPLEMENTATION

5.1 BLOCK DIAGRAM

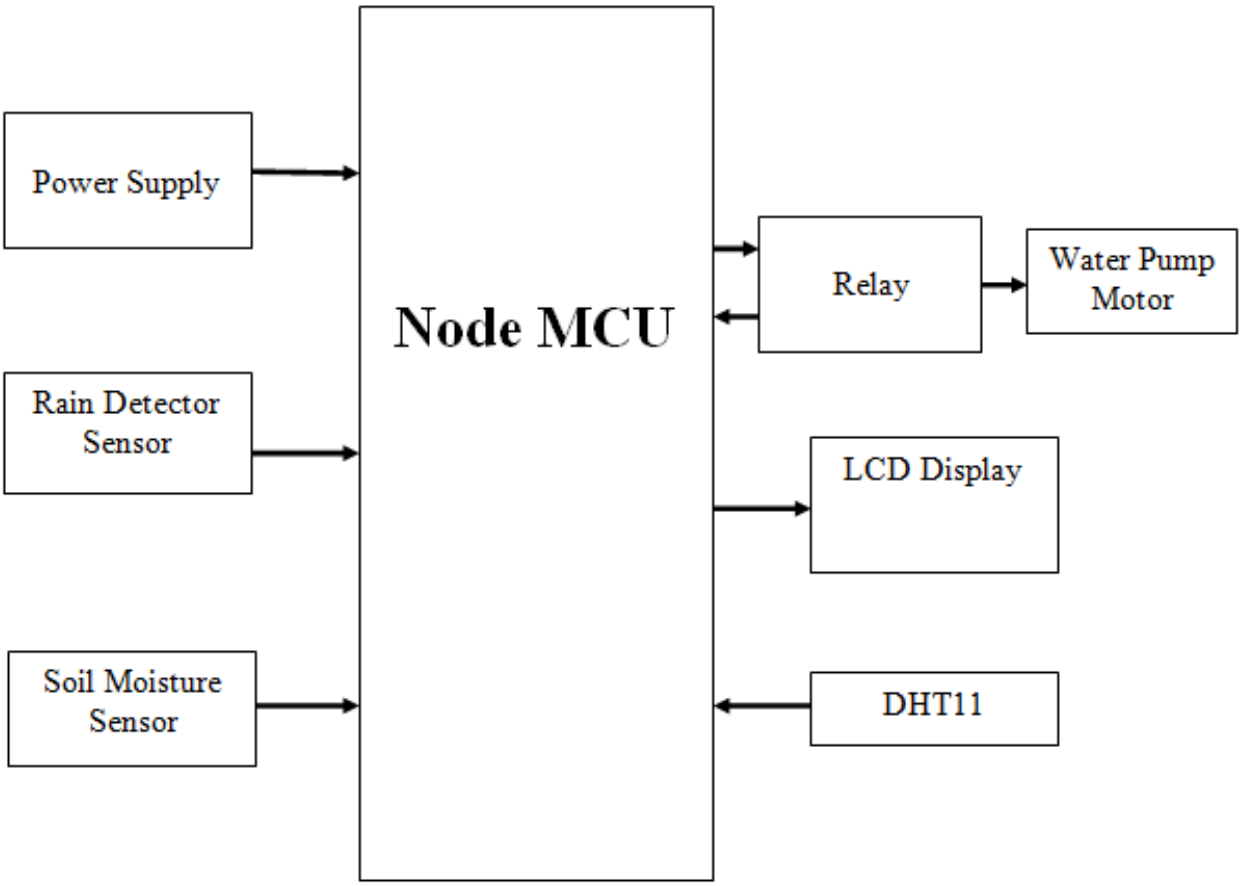

FIG: 5.1 Block Diagram

5.2 FLOW CHART

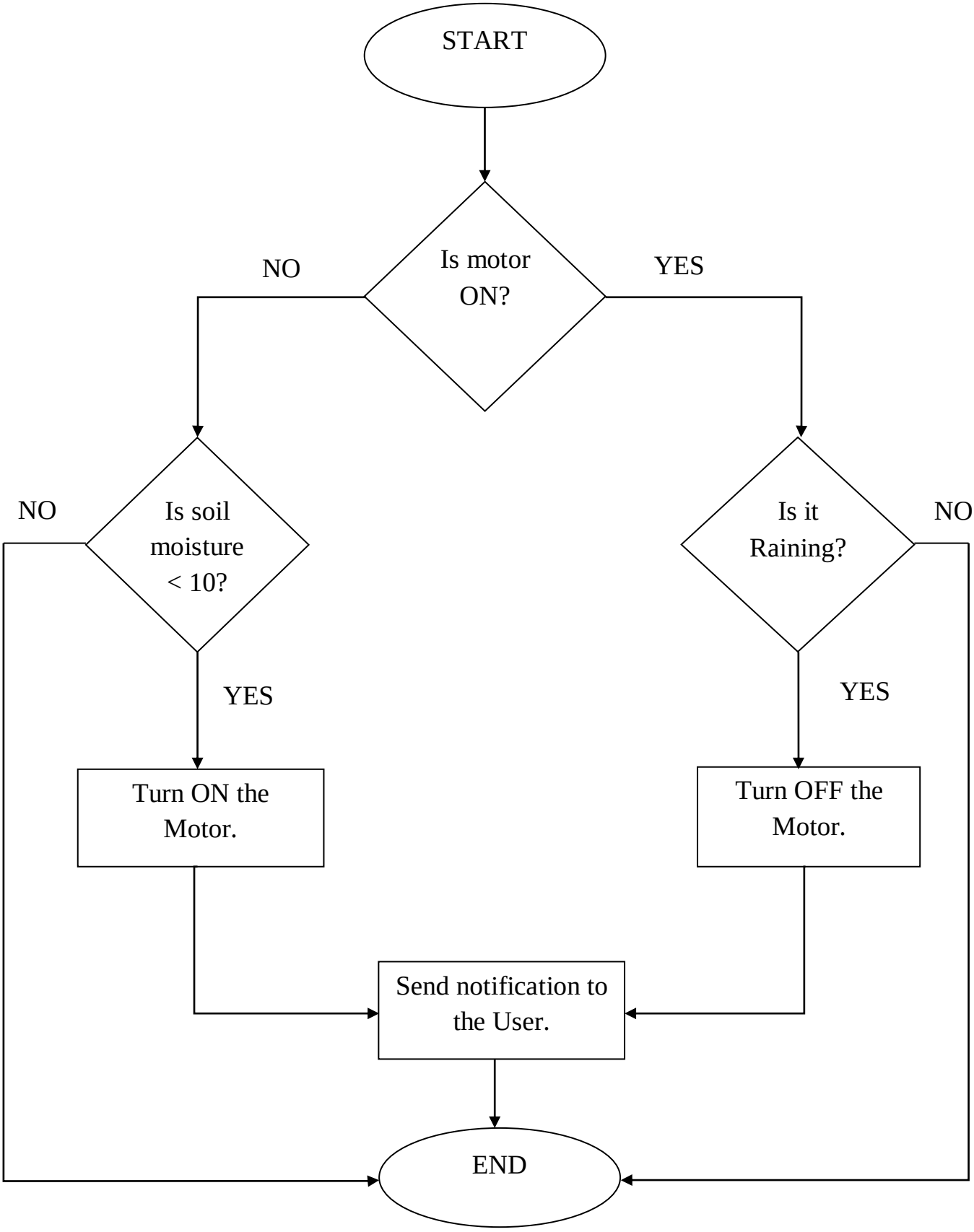

FIG 5.2: Working Flow of the Project

5.3 WORKING

This project is an IoT-based automatic watering system designed to manage the operation of a water pump based on soil moisture levels**,** rain detection, and remote control through the Blynk app**.** The core purpose of the system is to provide efficient irrigation by ensuring that plants are watered only when necessary and preventing overwatering during rainfall. The water pump is controlled by a relay, which functions as an electronic switch, enabling the system to turn the pump on or off automatically depending on the soil's moisture levels, and the user's manual inputs via the Blynk app. The system also integrates a rain sensor that actively detects rainfall, ensuring that the water pump does not operate when it is raining, thereby preventing unnecessary water usage.

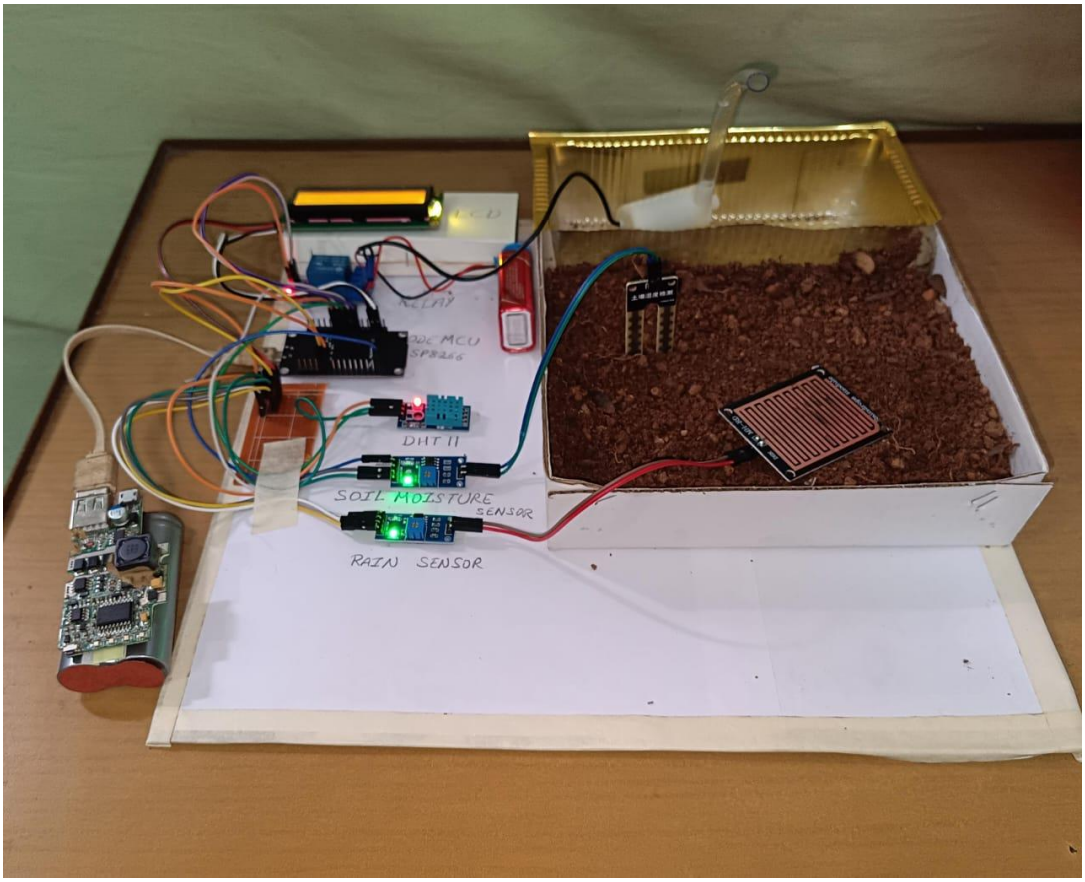

FIG: 5.3 Rain Detection And Motor Control System For Smart Agriculture

Upon powering up the system, the various sensors integrated into the system begin their initialization process. These include the DHT11 sensor**,** which measures both temperature and humidity, giving an idea of the overall environmental conditions. The soil moisture sensor is used to assess the moisture content in the soil, providing crucial

information to determine whether irrigation is needed. The rain sensor**,** which is another important component, detects the presence of rainfall and sends signals to the microcontroller when rain is detected. Once the system is fully initialized, the Blynk app allows users to remotely control the water pump. The app features a virtual button that can be toggled on or off to manually control the pump. Additionally, if the soil moisture is low, indicating that the plants need water, the system automatically activates the water pump to irrigate the soil, without requiring manual intervention. This makes the system both automatic and adaptable to varying environmental conditions.

The rain sensor plays a crucial role in the system's operation by preventing the water pump from running during rain. If the rain sensor detects rainfall, it triggers a signal to the system, alerting it to the presence of rain. In such cases, if the water pump is already on—whether due to low soil moisture or manual activation by the user—the system immediately sends a push notification through the Blynk app to notify the user. The notification might read, “Rain detected. Please turn off the motor,” ensuring that the user is informed in real time. To provide an additional layer of communication, the system is also set up to send an email notification to the user. The email, much like the push notification, alerts the user that rain has been detected and encourages them to turn off the water pump to avoid overwatering the plants. This feature is particularly useful for users who may not be actively monitoring the Blynk app but want to stay informed about the status of their watering system.

The system's flexibility is enhanced by the manual control available via the Blynk app**.** While the system operates automatically by using soil moisture data to trigger the water pump, users can override the system by manually turning the pump on or off via the app at any time. This means the user can choose to water the plants even if the moisture level is sufficient or stop watering if they believe it’s unnecessary. The LCD display continuously shows the real-time readings of soil moisture, temperature, and humidity, providing the user with constant feedback about the current conditions, ensuring that they can make informed decisions about whether to activate the water pump.

In essence, this IoT-based automatic watering system provides an efficient and intelligent solution for managing irrigation needs, making it ideal for home gardens, small farms, or any environment where efficient water usage is essential. The system ensures that the water pump is activated only when necessary—when the soil moisture is low—and automatically deactivated when rain is detected, preventing water wastage. The integration of Blynk app for remote control and monitoring, along with the push notifications and

email alerts for rain detection, makes the system user-friendly and highly convenient. It empowers the user with full control over the watering process, while simultaneously providing real-time updates on the environmental conditions and system status. This combination of automation, real-time communication, and user control results in an efficient, reliable, and easy-to-use watering system that helps users save water, reduce waste, and ensure their plants are always properly watered.

CHAPTER 6

RESULTS

6.1 DISCUSSION OF RESULTS

The project is working properly and giving the desired output.

FIG: 6.1 Running of the Water Pump

The output of LCD screen is observed where the current soil moisture level, temperature and humidity levels are displayed.

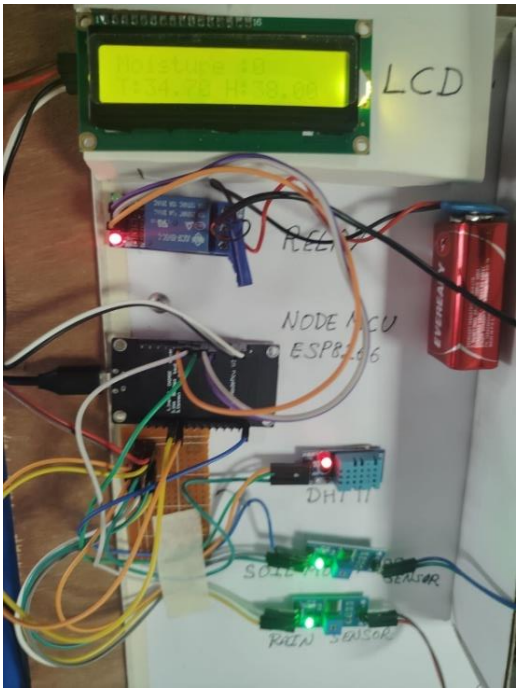

FIG: 6.2 Display of Humidity and Temperature

The mobile app output is observed in which soil moisture level, temperature and humidity level is observed and a switch is present to control the operation of the motor.

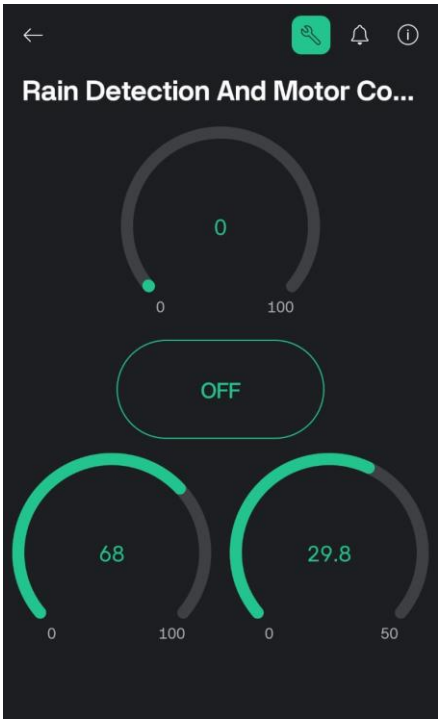

FIG: 6.3 Mobile App ON/OFF of the Motor

The notification alert is received by the user, when it rains outside.

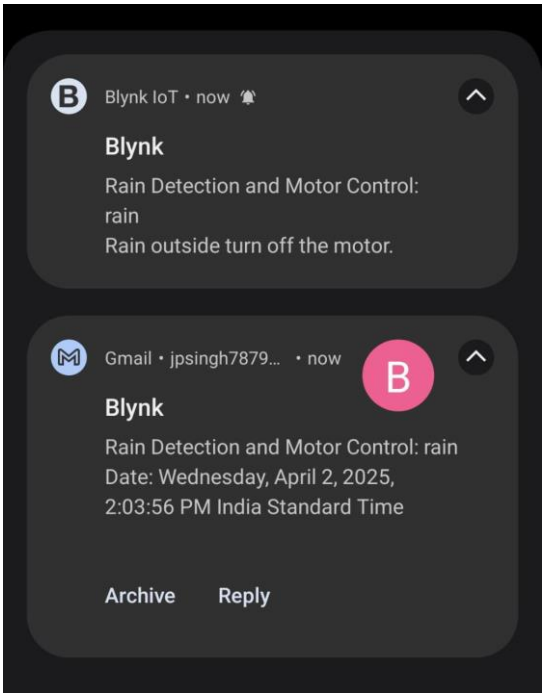

FIG: 6.4 Notification for the User

CHAPTER 7

APPLICATIONS AND ADVANTAGES

7.1 APPLICATIONS

1. Precision Agriculture:

- **Optimized Water Usage:** The system ensures that water is used efficiently by irrigating crops only when needed, based on soil moisture levels and rainfall detection. This reduces water waste, lowers costs, and helps conserve water resources.
- **Improved Crop Yield:** By providing the right amount of water at the right time, the system helps optimize growing conditions, leading to healthier crops and improved yields.
- **Soil Health Monitoring:** The system can be used to monitor soil conditions continuously, enabling farmers to maintain proper soil health, which is critical for sustainable farming.

2. Smart Farming (Remote Monitoring and Control):

- **Remote Monitoring:** Farmers can monitor the irrigation system remotely using a mobile phone, tablet, or computer. This is especially useful for farmers who manage large areas or have multiple fields.
- **Remote Control of Irrigation Systems:** Farmers can turn the irrigation system on or off from anywhere, ensuring they can react quickly to changing weather conditions or crop needs, without needing to visit the field.
- **Alerts and Notifications:** The system sends SMS alerts to farmers about rainfall, soil moisture levels, and irrigation status, allowing them to make timely decisions and avoid crop damage.

3. Water Management in Agriculture:

- **Efficient Water Conservation:** By automatically turning off the irrigation system when it rains or when soil moisture levels are adequate, the system helps conserve water, which is essential for areas facing water scarcity.
- **Cost Savings:** By reducing unnecessary water usage, farmers can save on water bills, electricity costs for running the motor, and other irrigation-related expenses.

4. Sustainable Farming Practices:

- **Environmentally Friendly:** The automated system promotes sustainable farming practices by reducing water wastage, conserving energy, and ensuring that crops receive the optimal amount of water for growth.
- **Climate-Smart Agriculture:** The system can adjust irrigation based on real-time weather conditions, making it more adaptable to climate change and varying environmental factors.

5. Smart Cities and Urban Agriculture:

- **Urban Agriculture:** In cities where space is limited, vertical farms and rooftop gardens can use automated irrigation systems to maintain plant health and productivity without requiring constant manual oversight.
- **Integration with Smart Cities:** The irrigation system could be integrated into smart city frameworks, where data from different sensors (such as weather stations, moisture levels, etc.) could be analyzed for broader water management solutions across urban farming initiatives.

6. Greenhouses and Horticulture:

- **Controlled Environment Agriculture:** For greenhouse farming, where environmental conditions are critical, automated irrigation systems help control watering based on precise environmental parameters. This ensures better plant health and higher yields.
- **Monitoring Plant Growth:** Automated systems can be used to track growth patterns, moisture levels, and other parameters important for horticulture and floriculture businesses.

7. Rural Development and Agriculture in Remote Areas:

- **Automation for Remote Areas:** In rural or remote areas where farmers may not have immediate access to irrigation resources, the automated system ensures efficient water usage without the need for constant manual intervention. This is especially helpful in regions where labor is scarce or expensive.
- **Access to Real-Time Information:** Farmers in remote areas can receive updates on their irrigation system status via SMS, even without internet access. This ensures that they remain connected and informed about the system's performance.

8. Agricultural Research:

- **Data Collection for Research:** The system can collect valuable data on soil moisture, weather conditions, and irrigation efficiency. This data can be used for agricultural

research, helping scientists and researchers develop better irrigation practices or study crop growth in various environmental conditions.

- **Pilot Projects for New Farming Techniques:** Researchers can use such systems to test the efficacy of different irrigation strategies or soil conditions for various crops, contributing to the development of advanced farming technologies.

9. Education and Training in Agriculture:

- **Training Farmers:** The system can be used in agricultural training programs to educate farmers on smart farming practices and the use of IoT technologies in agriculture.
- **Demonstration of IoT Applications:** It provides an excellent demonstration of IoT applications in agriculture for students, researchers, and farmers, showcasing how technology can improve farming practices and outcomes.

10. Commercial Farming and Large-Scale Agricultural Operations:

- **Automated Large-Scale Irrigation:** For commercial farms and large agricultural operations, where managing irrigation systems manually would be cumbersome, this automated system ensures that large areas are irrigated efficiently without human intervention.
- **Centralized Control:** With multiple motors and sensors connected to a central system, commercial farms can manage irrigation systems for large tracts of land from a single location or remotely, optimizing water use and operational efficiency.

7.2 ADVANTAGES

1. Water Conservation:

- **Efficient Water Usage:** The system ensures that water is used only when necessary by detecting soil moisture levels. This helps prevent over-irrigation and ensures that crops receive the right amount of water at the right time.
- **Prevention of Wastage:** The rain sensor helps to automatically turn off the motor when it detects rainfall, avoiding unnecessary watering when it’s already raining. This further reduces water wastage.

2. Cost Savings:

- **Reduced Water Bills:** By preventing over-irrigation, the system reduces the amount of water consumed, leading to lower water bills.

- **Reduced Electricity Consumption:** The motor is only activated when needed, and unnecessary irrigation is avoided, resulting in savings on electricity used for running the motor.
- **Reduced Labor Costs:** The automation of the irrigation process reduces the need for manual labor, saving time and money for farmers.

3. Improved Crop Yield and Quality:

- **Optimal Watering:** The system ensures that crops receive the right amount of water, avoiding both drought stress and waterlogging. This promotes healthier plants and better yields.
- **Consistent Irrigation:** Automated systems provide a consistent and uniform watering schedule, ensuring that crops are irrigated at the best possible intervals for growth.

4. Remote Monitoring and Control:

- **Remote Management:** The farmer can monitor the irrigation system and control the motor remotely through SMS, which is especially useful for farmers with large fields or who live far from their farms.
- **Instant Alerts and Notifications:** The GSM module sends SMS alerts regarding the system's status, including rain detection, motor status, and other important updates, keeping the farmer informed at all times.

5. Reduction of Human Error:

- **Automated Decision Making:** The system automatically makes decisions based on soil moisture and weather conditions, removing the possibility of human error in irrigation decisions.
- **Consistency in Operation:** Unlike manual irrigation, which can be prone to inconsistency, the automated system ensures that the irrigation process is carried out uniformly.

6. Time and Labor Efficiency:

- **Minimal Human Intervention:** The system reduces the need for constant monitoring, adjustments, and manual watering, freeing up time for farmers to focus on other important tasks or manage multiple farms.
- **Time-Saving:** Farmers no longer need to visit the fields frequently to turn the motor on or off, making it ideal for busy or remote farming operations.

7. Sustainability and Environmental Benefits:

- **Conservation of Natural Resources:** By reducing water wastage and preventing over-irrigation, the system contributes to the sustainable use of water, which is a valuable and finite resource.
- **Eco-Friendly Practices:** Automated irrigation is more environmentally friendly, ensuring that water is distributed efficiently, and reducing the environmental impact of over-irrigation, such as soil erosion or waterlogging.

8. Enhanced Decision-Making:

- **Real-Time Data Collection:** The system collects data on soil moisture, rainfall, and motor status, enabling farmers to make informed decisions based on real-time conditions. This helps in optimizing irrigation schedules and improving farm management.
- **Data-Driven Insights:** The data collected can be analyzed over time to identify trends and optimize irrigation strategies, improving overall farming efficiency.

9. Easy Integration with Other Smart Farming Technologies:

- **Scalability and Integration:** The system can easily integrate with other smart farming technologies such as climate monitoring systems, crop health sensors, and automated fertilization systems, creating a fully automated and efficient farming environment.
- **Future Proofing:** As new IoT devices and sensors are developed, the system can be upgraded or expanded to include additional features, making it adaptable to future technologies.

10. Farmer Empowerment and Convenience:

- **Ease of Use:** The system is user-friendly and does not require technical expertise to operate. Farmers can control and monitor the system via simple SMS commands, even without internet access.
- **24/7 Availability:** The system runs autonomously, providing continuous operation even if the farmer is unavailable or in a remote location.
- **Peace of Mind:** Farmers can have peace of mind knowing that the irrigation system is functioning properly, and any issues (such as rain detection or motor malfunctions) will be communicated to them immediately.

11. Support for Rural and Remote Areas:

- **Accessibility for Remote Areas:** Farmers in remote areas, where access to modern irrigation equipment or internet connectivity might be limited, can still benefit from the system by using SMS-based control and monitoring.
- **Improved Farm Management in Rural Areas:** The automation of irrigation systems can help smallholder farmers in rural areas improve farm management, productivity, and crop quality, boosting their economic viability.

12. Reduced Risk of Crop Damage:

- **Avoids Over-Irrigation and Under-Irrigation:** By using sensors to track moisture levels, the system reduces the risk of over-irrigation (which can lead to waterlogging and root damage) or under-irrigation (which can lead to drought stress and reduced yield).
- **Increased Crop Protection:** By adjusting watering schedules according to weather and soil conditions, the system helps protect crops from adverse conditions that might otherwise damage them.

13. Data-Driven Farming:

- **Smart Farming Approach:** The system provides data-driven insights that help optimize irrigation and farming practices, contributing to a smarter, more sustainable agricultural approach.
- **Trackable Performance:** With continuous data collection and analysis, the farmer can track the performance of the irrigation system over time, identify areas for improvement, and optimize water use in future seasons.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

The IoT-based automated irrigation system is an innovative solution to the challenges faced by farmers in managing irrigation effectively, particularly in regions where water resources are scarce or over-irrigation is a concern. By integrating various components such as a Node MCU, rain sensors, soil moisture sensors, GSM modules, and motor drivers, the system enables the automation of irrigation, ensuring that crops receive optimal water without the risk of wastage.

The system offers several significant advantages, including water conservation, cost savings, improved crop yield, reduced labor, and environmental sustainability. The use of sensors to detect soil moisture and rainfall allows for intelligent, real-time decision-making that ensures water is only used when necessary, thus preventing over-irrigation and optimizing crop health. Furthermore, the integration of SMS alerts and remote control functionality provides farmers with the flexibility to manage irrigation remotely, making the system especially beneficial for large farms or those in remote locations.

This project represents a leap toward smart farming, where technology plays a crucial role in improving productivity and sustainability. By reducing human intervention, minimizing water wastage, and enabling better control over irrigation processes, it enhances efficiency and economic viability for farmers. Moreover, it contributes to sustainable agricultural practices, which are increasingly critical as the world faces challenges like climate change, water scarcity, and growing food demands.

In conclusion, the IoT-based automated irrigation system is a practical and scalable solution that not only addresses the pressing issues of water management in agriculture but also empowers farmers to take more control over their farming practices. The potential for future enhancements and integration with other smart technologies makes this system a valuable tool in modernizing agriculture and improving farm management for years to come.

8.2 FUTURE SCOPE

1. Integration with Weather Forecasting Systems:

- **Predictive Irrigation:** By integrating real-time weather data or forecasts into the system, the irrigation process can be adjusted in anticipation of rainfall or changes in weather conditions. For example, if a rainstorm is expected, the system could preemptively reduce or stop irrigation, optimizing water usage and preventing over-watering.
- **AI-Driven Irrigation Scheduling:** Advanced algorithms can predict the optimal watering schedule based on forecasted weather, historical data, and current soil conditions, making the system even more intelligent.

2. Advanced Soil Monitoring and Fertilization:

- **Soil Nutrient Monitoring:** Adding sensors to monitor soil nutrients like nitrogen, phosphorus, and potassium could enable farmers to tailor irrigation and fertilization schedules based on soil health. This could ensure optimal crop growth and reduce unnecessary fertilization, which is both cost-effective and environmentally friendly.
- **Automated Fertilizer Application:** The system could be integrated with automatic fertilizer dispensers, allowing for the precise delivery of nutrients during irrigation cycles, improving crop yield and reducing fertilizer wastage.

3. Machine Learning and Data Analytics:

- **Predictive Analytics for Crop Health:** By analyzing historical data collected by the system (such as soil moisture, temperature, and weather patterns), machine learning models can predict potential crop diseases, pests, or other growth issues before they arise, allowing the farmer to take proactive measures.

4. Energy Efficiency and Solar Integration:

- **Solar-Powered Systems:** The irrigation system could be powered by solar energy to make it more sustainable, especially in remote areas with limited access to electricity. Solar panels can provide a green energy source, reducing the system's carbon footprint and operating costs.
- **Energy-Efficient Pumps and Motors:** Future iterations of the system could integrate more energy-efficient pumps and motors, reducing the overall energy consumption of the irrigation process.

5. Integration with AI-Powered Crop Monitoring:

- **AI and Image Recognition:** Using AI-powered cameras or imaging systems, the system can detect changes in plant health and water requirements based on visual cues, such as leaf color, size, and shape. This can help automate irrigation adjustments and prevent issues like water stress.
- **Crop Type Detection:** The system could use AI to identify different crop types through sensors and imaging, adjusting watering schedules based on crop-specific needs.

REFERENCES

[1] Yasin, Hajar M., Subhi RM Zeebaree, and Ibrahim MI Zebari. "Arduino based automatic irrigation system: Monitoring and SMS controlling." *2019 4th Scientific International Conference Najaf (SICN)*. IEEE, 2019.

[2] Guravaiah, Koppala, and S. Srinivasulu Raju. "e-Agriculture: irrigation system based on weather forecasting." *2020 IEEE 15th international conference on industrial and information systems (ICIIS)*. IEEE, 2020.

[3] Selvi, V. Myvizhi, et al. "Agriculture Field Monitoring and Motor Control System with Rain Detection." *2024 International Conference on Communication, Computing and Internet of Things (IC3IoT)*. IEEE, 2024.

[4] Barkunan, S. R., V. Bhanumathi, and V. Balakrishnan. "Automatic irrigation system with rain fall detection in agricultural field." *Measurement* 156 (2020): 107552.

[5] Kanupuru, Priyanka, and Uma Reddy Nadig Vijayendra Reddy. "Multi Node Based Smart Monitoring System with Motor Dry Run Avoidance for Sustainable Agriculture." *Instrumentation, Mesures, Métrologies* 20.1 (2021).

[6] Pareja, Michael, and Argel Bandala. "Fuzzy Irrigation System with Rain Detection and Fertilizer Control." *2020 IEEE REGION 10 CONFERENCE (TENCON)*. IEEE, 2020.

[7] Sarathkumar, D., et al. "IOT Based Motor Control and Line Detection for Smart Agriculture." *2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE, 2024.

[8] Sreeram, K., et al. "Smart farming—A prototype for field monitoring and automation in agriculture." *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2017.

[9] Nandurkar, S. R., V. R. Thool, and R. C. Thool. "Design and development of precision agriculture system using wireless sensor network." *2014 First international conference on automation, control, energy and systems (ACES)*. IEEE, 2014.

[10] Kaplun, Dmitrii, et al. "An intelligent agriculture management system for rainfall prediction and fruit health monitoring." *Scientific Reports* 14.1 (2024): 512.

[11] Kadage, A. D., and J. D. Gawade. "Wireless control system for agricultural motor." *2009 Second International Conference on Emerging Trends in Engineering & Technology*. IEEE, 2009.

[12] Obaideen, Khaled, et al. "An overview of smart irrigation systems using IoT." *Energy Nexus* 7 (2022): 100124.

[13] Ragab, Mohammed Ali, et al. "IOT based smart irrigation system." *International Journal of Industry and Sustainable Development* 3.1 (2022): 76-86.

[14] Nawandar, Neha K., and Vishal R. Satpute. "IoT based low cost and intelligent module for smart irrigation system." *Computers and electronics in agriculture* 162 (2019): 979-990.

[15] Pujahari, Rakesh Mohan, Satya Prakash Yadav, and Rijwan Khan. "Intelligent farming system through weather forecast support and crop production." *Application of Machine Learning in Agriculture*. Academic Press, 2022. 113-130.

APPENDIX

PROJECT CODE

```
#define BLYNK_TEMPLATE_NAME "Rain Detection and Motor Control"
//Include the library files
#include <LiquidCrystal_I2C.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
//Initialize the LCD display
LiquidCrystal_I2C lcd(0x27, 16, 2);

char ssid[] = 'WiFi Device";//Enter your WIFI name
char pass[] = "12345678";//Enter your WIFI password

BlynkTimer timer;
bool Relay = 0;

DHT dht(14,DHT11);

//Define component pins
#define sensor A0
#define waterPump 13
boolean flag;
void setup() {
  Serial.begin(9600);
  pinMode(waterPump, OUTPUT);
  digitalWrite(waterPump, HIGH);
  lcd.init();
  lcd.backlight();
  dht.begin();
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
```

```
  lcd.setCursor(1, 0);
  lcd.print("System Loading");
  for (int a = 0; a <= 15; a++) {
    lcd.setCursor(a, 1);
    lcd.print(".");
    delay(500);
  }
  lcd.clear();
  //Call the function
  timer.setInterval(10000L, DHT11sensor);
  timer.setInterval(10000L, soilMoistureSensor);
  timer.setInterval(10000L, rainSensor);
}
//rain sensor
void rainSensor() {
  boolean val = digitalRead(12);
  if(val == false && flag)
  {
    // Serial.println("Its raining");
    Blynk.logEvent("rain");
    Blynk.virtualWrite(V1, 0);
    digitalWrite(waterPump, HIGH);
  }
}

//Get the button value
BLYNK_WRITE(V1) {
  Relay = param.asInt();
  if (Relay == 1) {
    digitalWrite(waterPump, LOW);
    flag = true;
  }

else {
```

```
    digitalWrite(waterPump, HIGH);
    flag = false;
  }
}
//Get the soil moisture values
void soilMoistureSensor() {
  int value = analogRead(sensor);
  value = map(value, 0, 1024, 0, 100);
  value = (value - 100) * -1;

  Blynk.virtualWrite(V0, value);
  lcd.setCursor(0, 0);
  lcd.print("Moisture :");
  lcd.print(value);
  lcd.print(" ");
}

//DHT11 Sensor files..
void DHT11sensor() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  lcd.setCursor(0, 1);
  lcd.print("T:");
  lcd.print(t);
  lcd.setCursor(8, 1);
  lcd.print("H:");
  lcd.print(h);
  Blynk.virtualWrite(V2, h);
  Blynk.virtualWrite(V3, t);
```

```
}
void loop() {
  Blynk.run();//Run the Blynk library
  timer.run();//Run the Blynk timer
}
```