CHAPTER 1

INTRODUCTION

Precision farming is one of the most important features in agriculture in a country with a large population, fertile soil and abundant water . This system is suitable for different types of water use such as interpretation water , river water and water discharge. In agricultural situations, water quality management is important due to limited water resource and agriculture dependence on irrigation .It solves the waste water problem by optimizing irrigation based on soil moisture content and ambient temperature to reduce waste water and promotes plant growth using on-site sensors and data analysis.

Banning water is a good price to pay. It works in two places, one is to make the sensor of the microcontroller, the other is to start the water flow. In other countries such as Bangladesh, solar panels have been used to provide environmentally friendly energy. This system uses sensor nodes and wireless communication to provide real- time monitoring and automatic water management and solve water constraints. To achieve water quality, we can also use batteries produced by PV to minimize the water and data analysis.

The integration AI with sensor driven frameworks offers a promising arrangement for the challenges that framing faces due to water shortage. This inventive approach revolution's conventional water system hones by optimizing water utilization and minimizing wastage through sensors, showing the clients almost the precise edit water needs.

The integration AI with sensor driven frameworks offers a promising arrangement for the challenges that framing faces due to water shortage. This inventive approach revolution's conventional water system hones by optimizing water utilization and minimizing wastage through sensors, showing the clients almost the precise edit water needs.

1.1 PROBLEM DEFINITON

AI Driven Sensor for Irrigation and Water Waste Minimization aims to develop a sophisticated system that utilizes AI algorithms and sensor technology to optimize irrigation processes and minimize water wastage in agricultural. An intelligent irrigation system capable of accurately monitors the soil moisture levels, weather conditions and plant requirement in real-time. By analysing the data collected from sensors, the AI algorithm will make timely decisions regarding the optimal timing, duration, temperature, humidity and amount of water to be applied to the crops, ensuring efficient water usage while maximizing yield. This project aims to develop an AI-driven solution for precise water management in piped and micro irrigation networks by predicting dynamic crop water demands, integrating real-time soil moisture data, and automating valve control mechanisms to optimize water usage and crop yield.

1.2 OBJECTIVE OF THE PROJECT

The primary objective of AI driven sensor system is to enhance the smart irrigation and the waste water minimization. Specially, the objectives include:

- Real-time monitoring: Utilizing a arrange of sensors to assemble significant natural and soil information, counting dampness levels, temperature and climate conditions
- AI-powered water forecasting: Utilizing machine learning, analyze sensor information and anticipate the exact water pre-requisites for particular crops at distinctive development stages.
- Data-driven choice making: Giving ranchers with profitable information and to optimize water system methodologies, make strides asset administration and improve wellbeing.

1.3 SIGNIFICANCE OF THE PROJECT

The Significance of the Smart irrigation lies in its potential to address several critical issues in agriculture and water resource management”

- Enhancing the crop yield by ensuring the crops that receive the right amount of water at the right time.
- Efficient irrigation practices not only conserve water but it also reduces the environmental impact associated with excessive water usage.
- The irrigation schedules and minimizing water wastage, farmers can reduce their operational costs associated with water usage and translates to increased profitability and economic for agriculture operations.
- It offers a solution by minimizing water wastage through precise monitoring and intelligent irrigation strategies, thereby conserving various valuable water resources.
- It demonstrated the potential of technology-driven solutions to address complex challenges in farming, paving the way for further innovations and advancements in precision agriculture and water management.

1.4 OUTLINE OF THE PROJECT

A system of smart irrigation aims to tackle the water inefficiencies in irrigation by leveraging cutting-edge technology. Through the strategic placement of sensors including soil moisture, weather and temperature. Data collected is then processed using AI algorithms, which employ machine learning models for predictive modelling of water requirements and optimization algorithms for irrigation scheduling. It’s control unit communicate with actuators and a user interface, enabling real-time adjustments to irrigation settings.

CHAPTER 2

LITERATURE REVIEW

The development of a system using the cost- effective solar energy system for open wells that meets the urgent need for clean drinking water in rural areas. Realizing the problems of electricity usage in these areas, the system uses solar energy to work efficiently. By using glass to optimize the exchange of sunlight, the efficiency of the system is increased and to reduce energy costs and promote environmentally friendly practices. This solar water pump has many applications such as open well water extraction.

A new way to manage water in pipes that could be used in gardens by replacing human interaction with a network of wireless sensors. This also saves the water up to 34 % by using temperature, humidity and its data, and up to 26% by using temperature alone. It also includes the use of real cases and discussed.

The implementation of intelligent irrigation and adjusting the irrigation system is essential in today's agricultural System that control the value of water that is required for the plant. Two models were used to analyze the data: the regression model in SPSS software and the genetic function model in the software. Among the developed countries, India is one of the largest countries in the world today, with a large part of its economy is based on agriculture.

The country also has the traditional methods have been used for irrigation of farm also requires automation to reduce the time and effort required for manual reviews. It also uses deep learning models to classify the captured images into saggy and healthy classes. The fuzzy logic algorithm brings all the parameters together and controls the operating time of the irrigation system. Drip irrigation with the help of IoT and soil moisture sensors has many advantages over traditional irrigation methods . By transmitting water directly to the roots, it helps reduce water waste and improve crop quality .

Using IoT technology and soil moisture sensors can make more efficient and cleaner water planning , help save water and reduce energy costs . It provides instant information about soil moisture that can be used for irrigation purposes. Sensors can be connected to the central control system and the engine can be turned on and off according to the data collected by the sensors.

Address the issue of wasteful water utilization and a lack of automation in conventional irrigation techniques, for which, a Smart Irrigation Management System (SIMS) is being implemented.

A cost-effective smart irrigation system based on microcontrollers is proposed, featuring adjustable parameters to cater to specific soil conditions. The system, comprising humidity sensors, relays, and water pumps, operates in zones. By utilizing soil moisture sensors positioned near plant roots and solar panels for energy, the proposed system aims to increase water savings up to 64%, compared to conventional irrigation methods, thereby enhancing agricultural sustainability and productivity.

By integrating modern sensors and mobile app feedback, the system optimizes water resource utilization through big data analysis. It enhances crop productivity through continuous, remote soil moisture monitoring and precise irrigation, improving water use efficiency and promoting sustainable agriculture.

2.1 DESCRIPTION OF THE METHODOLOGY

The methodology of an AI-driven sensor-based irrigation system encompasses a systematic approach to data collection, analysis, and decision-making aimed at optimizing water usage and promoting efficient irrigation practices in agriculture. At its core, the methodology involves the integration of various sensors, including soil moisture sensors, weather sensors, and evapotranspiration sensors, strategically placed within the irrigation area .

Data collected by these sensors is then processed and analyzed using advanced AI algorithms, which employ machine learning techniques for predictive modeling of water requirements and optimization algorithms for irrigation scheduling.

Data-driven approach enables the system to accurately determine when and how much water should be applied to the crops, taking into account factors such as soil type, crop type, weather forecasts, and evapotranspiration rates.

The central control unit of the system serves as the hub for data processing and decision-making, leveraging the insights derived from the AI algorithms to dynamically adjust irrigation settings in real-time. Through this methodology, farmers can make informed decisions regarding irrigation management, leading to reduced water waste, increased crop yields, and improved agricultural sustainability.

Ongoing monitoring and refinement of the system's methodology are essential to ensure its effectiveness and adaptability to changing environmental conditions and crop requirements, thereby maximizing its impact on water conservation and agricultural productivity.

2.2 SPECIFICATION OF MODEL

The model specifications for an AI-driven sensor-based irrigation system entail several crucial components and functionalities. Firstly, it requires high-precision sensors such as soil moisture sensors, weather sensors, and evapotranspiration (ET) sensors for accurate data collection on soil moisture levels, environmental conditions, and evapotranspiration rates.

The sensors feed data into sophisticated AI algorithms, including machine learning models for predictive water requirement modeling and optimization algorithms for efficient irrigation scheduling.

The system architecture comprises a central control unit with robust processing capabilities, supported by a reliable wireless communication infrastructure for seamless data transmission.

A user-friendly interface, accessible via web or mobile applications, empowers farmers with real-time insights and control over the irrigation system.

Implementation involves deploying quality hardware components like microcontrollers and actuators, alongside software development utilizing programming languages and frameworks for algorithm implementation and testing.

The benefits of this model include water conservation, increased crop yield, cost savings, and environmental sustainability. Adhering to these specifications ensures the system's effectiveness and reliability in optimizing irrigation practices for sustainable agriculture.

CHAPTER 3

SYSTEM ANALYSIS

System Analysis is about complete understanding of existing systems and finding where the existing system fails. The solution is determined to resolve issues in the proposed system. It defines the system. The system is divided into smaller parts. Their functions and inter relation of these modules are studied in system analysis. Software is required to act as an interface into the digital world by understanding user requests or commands and then translating into actions or recommendations based on the user's understanding of the world.

3.1 EXISTING SYSTEM

The existing irrigation systems lack the real-time sensor data for precise water waste management. The traditional irrigation systems approach is based on the following criteria:

- Timers or Fixed Schedules: Fixed timers or pre-set schedules- based irrigation may be difficult since the farmers has to check for soil moisture levels based on real-time varying weather conditions in person. The variation in weather conditions results in a change in crop water needs which are pre-defined.
- Manual Monitoring: Manual monitoring may result in underwatering or overwatering to the crops in case of any situation where the farmer is unable to visit them.
- Notify about weather conditions: A notification to the farmer on changing weather conditions will be sent which helps them to know about the current weather of their farm from a specific distance. This is difficult in some cases where the farmer in unable to reach the location based on different weather conditions.

Water management in agriculture relies heavily on manual observation and traditional irrigation schedules. In this system, farmers often face challenges in accurately assessing crop water needs, resulting in over-or under-irrigation, which can lead to water waste, decreased productivity, and environmental damage.

Descion-making in the system is based on the analysis of the information provide by the user and it is predefined with optimized actuators or control systems to adjust the water supply in response to AI. It provides the addition aim of using water efficiency, increasing crop yields and reducing the environmental impacts of agricultural interventions, compatibility with existing water infrastructure, energy efficiency and productivity.

3.2 DRAWBACKS

- The initial cost for setting up the sensors, actuators and the central control unit which can be relatively high which poses us a barrier to adoption for small-scale farmers or those with limited resources.
- Implementing and maintaining such a system requires technical expertise in areas like sensor integration, AI algorithm development, and software programming, which may be challenging for some users.
- Continuous operation of sensors, actuators, and communication devices consumes energy,
- To operational costs and environmental impact, especially if powered by non-renewable energy sources.
- Collecting and storing sensitive agricultural data, including crop yields and irrigation schedules, raises concerns about data privacy and the potential for unauthorized access or misuse.

3.3 PROPOSED SYSTEM

The AI-Driven system will integrate AI-driven technology with sensor networks to create a smart irrigation system capable of optimizing water management in piped and micro irrigation setups. Sensors will analyze data , including soil moisture, weather conditions and adjust irrigation schedules and flow rates in real time to ensure optimal moisture levels in the soil while minimizing water waste.

The system aims in making the irrigation system smarter by implementing sensors which helps in gathering the real-time weather conditions and notify the farmers regarding it. Our project possesses the following functionalities:

- Real-time Data Collection: Continuous monitoring on soil moisture, temperature, humidity and even weather forecasts will be done.
- AI-powered physical presence monitoring: AI analyzes the person or animal that enter into the crops and indicate to the farmer may find it easier to protect the crops which is an added advantage. This process is done using the image processing technique where a camera will be continuously monitoring the field.
- Automated Irrigation Control: The sensor-based system controls valves or sprinklers, delivering the precise amount of water needed at the right time by indicating the state of the soil moisture content by which the farmer may control turning on or off the motor using the application.
- Minimized Water Waste: Water will be consumed precisely by providing only the required amount of water to the crops.
- Improved Crop Yields: Healthier growth and potentially higher yields are achieved when crops receive the ideal amount of water.

The proposed system has the following advantages:

- Environmental Sustainability: By reducing water waste and minimizing the environmental impact of irrigation practices, such as soil erosion and water runoff, the system contributes to environmental sustainability and conservation efforts.
- Remote Monitoring & Control: With wireless communication capabilities and cloud-based services, farmers can remotely monitor and control the irrigation system from anywhere, providing greater flexibility and convenience in managing their agricultural operations.
- Adaptability & Flexibility: The system can adapt to changing environmental conditions and crop requirements, allowing for dynamic adjustments to irrigation schedules and settings to accommodate seasonal variations and crop growth stages.
- Improved Farm Management: By integrating with other smart farming technologies, such as crop monitoring and pest detection systems, the irrigation system contributes to a comprehensive farm management, efficiency and productivity.

3.4 FEASIBILITY STUDY

An analysis and evaluation of a proposed project to determine if it is technically feasible, is feasible within the estimated cost, and will be profitable. Feasibility studies are almost always conducted where large sums are at sake. A feasibility study aims to objectively and rationally uncover the strengths and weakness of an existing sensor applications and threats present in the environment, the resources required to carry through, and ultimately the prospects the success of the environment.

3.5 TESTS OF FEASIBILITY

Feasibility study is conducted once the problem is clearly understood. Feasibility is necessary to determine that the proposed system in AI Driven sensor is feasible by considering the technical, operational and economical factors. By having a detailed feasibility study will have a clear-cut view of the proposed system of the AI Sensor System for Irrigation and Water waste Minimization.

Feasibility study encompasses the following things:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

3.5.1 TECHNICAL FEASIBILTY

The technical feasibility of an AI-driven sensor-based irrigation system rests upon the availability and sophistication of sensor technology, which has seen significant advancements in recent years. Soil moisture sensors, weather sensors, and evapotranspiration sensors have become increasingly accurate and affordable, enabling precise monitoring of environmental conditions and soil moisture levels in real-time. Moreover, the rapid progress in AI and machine learning techniques facilitates the development of algorithms for data analysis, predictive modeling, and optimization, allowing for sophisticated irrigation management. The communication infrastructure required for data transmission between sensors, actuators, and the central control unit is readily available through wireless technologies such as Wi-Fi, Bluetooth, and cellular networks, further enhancing the technical feasibility of the system. Additionally, the hardware components necessary for the system, including microcontrollers, single-board computers, and actuators, are widely accessible at affordable.

3.5.2 OPERATIONAL FEASIBILITY

The operational feasibility of an AI-driven sensor-based irrigation system hinges on several key factors aimed at ensuring the system's usability, reliability, and maintainability. Designing an intuitive user interface for the system, accessible via web or mobile applications, is paramount to facilitating ease of use and ensuring that farmers can effortlessly monitor and control the irrigation system's operation. Additionally, streamlining maintenance procedures, such as sensor calibration and software updates, helps minimize downtime and ensures reliable performance.

Providing adequate training and technical support to farmers and agricultural workers is crucial for enhancing the system's operational feasibility, as it equips them with the knowledge and skills necessary to effectively operate and troubleshoot the irrigation system. By addressing these operational considerations, the system can be seamlessly integrated into existing agricultural practices, enhancing its overall feasibility and usability.

3.5.3 ECONOMICAL FEASIBILITY

The economic feasibility of an AI-driven sensor-based irrigation system relies on conducting a comprehensive cost-benefit analysis to evaluate its financial viability and potential returns on investment. Assessing factors such as initial investment costs, operational expenses, water savings, and potential crop yield increases is essential for determining the economic feasibility of implementing the irrigation system.

By quantifying the projected savings in water usage, labour costs, and potential increases in crop yields, farmers can calculate the return on investment (ROI) and gauge the economic viability of adopting the technology. Furthermore, government incentive programs,subsidies or grants offered by agricultural.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS

Processor : i3 Processor

Speed : 3.00 GHZ

RAM : 4 GB

Hard Disk : 500 GB

Monitor : 16’ Color Monitor

Keyboard : Standard 110 keys

Pointing Device : Mouse

Smart Phone : Any type

4.2 SOFTWARE REQUIREMENTS

Operating System : Windows/Linux/Visual studio

Front End : Android Eclipse

Back End : SQL server, java

Web Service : Visual Studio

4.3 TOOLS & FRAMEWORKS

The Blynk Smart Irrigation Control app revolutionizes the way users manage and interact with their irrigation systems. By harnessing the power of IoT and intuitive design, it empowers users to achieve optimal plant growth, conserve water resources, and cultivate healthier landscapes effortlessly.

Whether for residential gardens, agricultural fields, or commercial landscapes,the Blynk app sets a new standard for smart irrigation management.

Microcontroller board is the brain of the system, processing sensor data and controlling the water pump. Popular options include ESP8266 (NodeMCU), ESP32, or Arduino Uno.

Soil moisture sensor is the sensor measures the moisture level in the soil, sending a signal to the microcontroller. Relay module acts as a switch, controlled by the microcontroller to turn the water pump on or off. Water pump, delivers water to your plants based on the system's control.

Blynk app is a visual programming platform for IoT applications. You'll use it to create a user interface for monitoring sensor data and controlling the irrigation system remotely.

Arduino IDE is the Integrated Development Environment where you'll write code to program the microcontroller board and the temperature and humidity sensor (DHT11) sensors provide additional environmental data you can monitor in the Blynk app.

CHAPTER 5

SOFTWARE DESCRIPTION

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and nonfunctional requirements, and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement of the bus information and on route of the local bus service what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

5.1 FRONT END

The Blynk Smart Irrigation Control app is compatible with a wide range of IoT devices, including microcontrollers (e.g., Arduino, Raspberry Pi), sensors, and actuators commonly used in smart irrigation systems. It supports both iOS and Android platforms, ensuring accessibility across various mobile devices.

The Blynk Smart Irrigation Control app is a user-friendly interface designed to facilitate efficient management and monitoring of irrigation systems remotely. Seamlessly integrated with IoT (Internet of Things) technology, it empowers users to control watering schedules, monitor soil moisture levels, and optimize water usage with ease and precision.

Blynk Mobile App Interface is the mobile app serves as the frontend interface for users to interact with the smart irrigation system. Users can view sensor data, control irrigation schedules, and receive alerts through the app.

5.2 BACK END

- Microcontroller (ESP8266): The microcontroller serves as the central processing unit for the smart irrigation system. It collects data from various sensors and controls actuators such as water pumps and valves.
- Sensors: Soil Moisture Sensor: Measures the moisture level in the soil. Weather Sensor is optional and it Measures environmental factors like temperature, humidity, and rainfall. These sensors provide input data to the microcontroller for decision-making.
- Actuators: These actuators are controlled by the microcontroller based on sensor readings and user inputs
- Water Pump: Controls the flow of water to the irrigation system.
- Solenoid Valve: Controls the release of water to specific areas or plants.
- Blynk IoT Platform: Blynk serves as the bridge between the microcontroller and the mobile app.. Blynk provides widgets that allow users to visualize sensor data, control actuators, and set up automation rules. It also handles user authentication, data logging, and push notifications.
- Backend Code (Microcontroller Programming): The microcontroller firmware is programmed to read sensor data, make irrigation decisions based on predefined algorithms or user settings, and communicate with the Blynk cloud server. The code should handle tasks such as initializing sensors, reading sensor values, controlling actuators, and sending/receiving data from the Blynk server.

CHAPTER 6

PROJECT DESCRIPTION

AI Driven Sensor Driven System for irrigation is an online process developed for individuals seeking for the improvement of the agricultural environment.

6.1 OVERVIEW OF THE PROJECT

AI-driven sensor-driven system for irrigation and water waste minimization is to build a model that helps user in reducing the burden of continuous monitoring. One among them is data acquisition and monitoring. Deploy a network of sensors to collect real-time data on crucial environmental factors like soil moisture, temperature, and rain. Integrate weather data feeds (potentially local weather stations) to provide a broader environmental context. Develop a communication system for sensor data transmission. Design a user interface for data visualization, allowing farmers to monitor soil moisture levels, temperature trends, and recent rainfall events. Next is the sensor-based irrigation control. It establishes threshold values for soil moisture based on specific crop types and their water requirements. Program the irrigation system to activate automatically when soil moisture falls below the pre-defined threshold. Utilize timers or flow meters to control the duration and volume of water delivered during each irrigation cycle.

Implement rain shutoff sensors to automatically cease irrigation during rainfall events, preventing overwatering. Water conservation and efficiency includes minimized water waste by eliminating unnecessary irrigation based on real-time soil moisture data. Optimize water delivery based on pre-defined crop needs and actual soil conditions, reducing reliance on guesswork or traditional schedules. This provides farmers with a user-friendly interface for system control, allowing adjustments to threshold values and irrigation schedules.

Prioritized low-power sensors and communication protocols to improve system sustainability and reduce operational costs.

The system is user-friendly and easy to maintain for farmers with varying technical backgrounds. As a result, the sensor-driven irrigation system can significantly improve water management in agriculture. Farmers can benefit from reduced water waste, improved efficiency, and potentially higher crop yields. While this system lacks the predictive capabilities of AI, it offers a cost-effective and data-driven approach to irrigation control.

6.2 MODULE DESCRIPTION

6.2.1 HISTOGRAM EQUALIZATION

A continuous monitoring on the field is done using the YOLO algorithm which helps in detecting the objects in the field at a single shot. Deployment of IP cameras strategically throughout a field to capture real-time movements are implemented. Implementation of a computer vision system running the YOLO algorithm to continuously analyze the capture. YOLO is pre-trained to identify and classify specific objects relevant to field monitoring. This could include any animals or persons and many other object. In each frame, YOLO detects these objects, draws bounding boxes around them, and assigns class labels with confidence scores. The system continuously monitors the output from YOLO, tracking the presence, location, and count of objects of interest. If YOLO detects objects, the system can trigger alerts which will be received to the application. Alerts can be visualized such as on-screen notifications. This enables real-time monitoring of field conditions without the need for constant physical presence. The specific objects YOLO detects depend on the pre-trained model. Training a custom YOLO model might be necessary for field-specific applications. Reliable internet connectivity is crucial for real-time monitoring and remote alerts.

6.2.2 HARDWARE ASSEMBLING

The hardware components such as rain sensor, soil sensor, temperature sensor, a relay, regulator, motor, node MCU and LCD display are assembled such that status of the moisture soil is passed continuously accurately and taken necessary actions when needed.

- Power Supply: Connect the DC power supply to the ESP8266 board according to its specifications (5V).
- Sensors: Connect each sensor's ground pin to the ESP8266's GND pin. Connect the sensor's power pin (5V) to the corresponding power rail on the ESP8266. Connect the sensor's signal pin to a designated analog or digital pin on the ESP8266.
- Relay Module: Connect the relay module's control pin to a digital pin on the ESP8266. Connect the relay's power supply according to the module's specifications. Connect the motor's positive wire to the normally open (NO) terminal of the relay, and the motor's negative wire to ground.
- LCD Display : Connect the LCD display to the ESP866 according to its specific connections . This typically involves data pins, control pins, and a power supply connection.
- Jumper Wires: Use jumper wires to make all the necessary connections between components. A breadboard will be helpful for prototyping the circuit before soldering connections permanently. A voltage regulator might be needed if the motor or relay requires a voltage different from the power supply.

6.2.2 HARDWARE IMPLEMENTATION

Implementing a sensor-based smart irrigation system utilizing soil moisture, rain, and temperature sensors, along with relay modules, regulators, and a motor, controlled by an ESP8266 microcontroller, and providing real-time feedback through an LCD display can be split into certain modules that includes:

- Sensor Integration and Calibration: Integrate soil moisture, rain, and temperature sensors with the ESP8266 microcontroller, ensuring compatibility and proper wiring. Calibrate sensors to accurately measure and report soil moisture levels, rainfall intensity, and ambient temperature.
- Data Acquisition and Processing: Develop firmware for the ESP8266 to collect data from the integrated sensors at regular intervals. Implement algorithms to process sensor data and determine irrigation requirements based on predefined thresholds and conditions.
- Control Mechanism: Utilize relay modules to control the irrigation system's motor and valve, enabling automated water flow regulation. Develop logic to activate the motor and valve based on real-time sensor readings and irrigation schedules.
- Regulation and Feedback: Integrate regulators to manage water flow and pressure within the irrigation system, optimizing efficiency and preventing water waste. Implement feedback mechanisms to monitor system performance and provide alerts or notifications through the LCD display in case of anomalies or irregularities.
- User Interface Design: Design and implement an intuitive user interface on the LCD display to provide real-time feedback on soil moisture levels, rainfall status, and system operation.

- Testing and Optimization: Conduct thorough testing of the hardware implementation to validate sensor accuracy, control reliability, and system responsiveness. Optimize firmware algorithms and control logic based on testing results and user feedback to enhance system performance and efficiency. Provide training and educational materials to users on system operation, maintenance, and troubleshooting.

6.2.3 USER APPLICATION

Blynk app is implemented for the following reasons: Remote monitoring and control: Blynk allows to monitor sensor data (soil moisture) and remotely control the irrigation system (activate or deactivate pump) from smartphone.

Visualization: Create a user-friendly interface with virtual widgets like gauges to display soil moisture levels in real-time.

Alerts: Set up notifications on your phone to be alerted when soil moisture falls below a pre-defined threshold, indicating the need for irrigation.

Steps involved in integrating:

- Install Blynk App: Download and install the Blynk app on smartphone.
- Create a Blynk Account: Sign up for a free Blynk account within the app. This provides an authentication token for communication.
- Include Blynk Library: Add the Blynk library for development with NodeMCU.
- NodeMCU Code: Writing code for your NodeMCU

- Initializes communication with the Blynk server using our authentication token and Wi-Fi credentials.
- Reads sensor data (soil moisture) from connected sensors.
- Sends sensor data to specific virtual widgets created on the Blynk app interface.
- Receives commands (turn on/off pump) from buttons or sliders created in the Blynk app and controls the irrigation system accordingly (activating the relay connected to the water pump).
- Blynk App Interface Design: Design Blynk app interface by adding virtual widgets.
- Used a gauge to display the real-time soil moisture level received from the NodeMCU.
- Adding a button or slider to allow remote control of the irrigation system (turning the pump on or off).
- Blynk offers a user-friendly platform for remote monitoring and control, enhancing the functionality of our smart irrigation system. The visual interface provides easy-to-understand data representation and allows for interactive control.

6.3 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. The DFD also provides information about the outputs and inputs of each entity and the process itself.

The DFD breaks down the main process into several subprocesses. Data Collection and Preprocessing handle the raw sensor data, cleaning and preprocessing it for further analysis. Data Analysis and Decision Making process this preprocessed data along with historical data and user preferences .

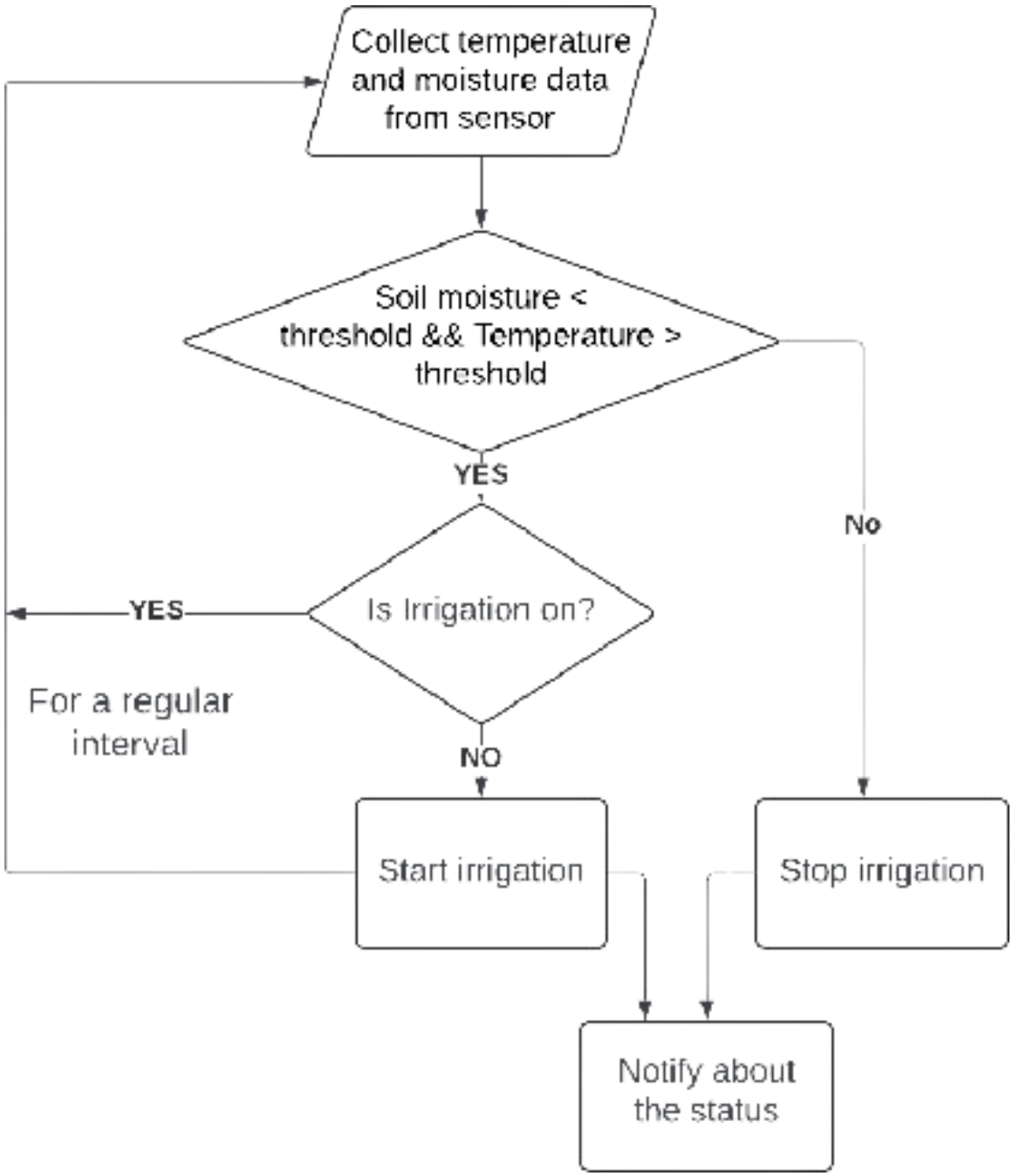

FIG 6.3: Block Diagram of the model

CHAPTER 7

SYSTEM TESTING

One of the most important phases in the development of any software is testing. Testing is a dynamic technique of verification and validation, which ensures that the software meets the expectations of the user. It involves executing an implementation of the software with test data. Test data is the set of data that the system will process as a normal input. A test case is a set of sequential steps to execute a test, operating on a set of predefined inputs to produce certain expected outputs. Testing demonstrates the software functions work according to specifications. In addition data collected from testing provides a good indication of software reliability and quality. Testing results in the deduction of number of errors. In this project critical modules are tested. The fields are tested for various validation methods using the different testing techniques

7.1 TESTING METHODS

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.
- Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.2 TYPES OF TESTING

7.2.1 Unit Testing

Test individual components or modules of the system, such as functions, classes, or methods, to ensure they work correctly. Verify the functionality of each unit and validate inputs and outputs.

7.2.2 Integration Testing

Test the interaction and integration between different modules or components of the system. Validate the flow of data and functionality between various parts of the system.

7.2.3 System Testing

Test the system as a whole to ensure it meets the specified requirements and functions correctly. Verify the system's behaviour, user interfaces, and overall functionality.

7.2.4 User Acceptance Testing (UAT)

The Involve end users or representatives to validate the system's usability and ensure it meets their needs and expectations. Conduct real-world scenarios and gather feedback to assess user satisfaction.

7.2.5 Performance Testing

Test the system's performance under expected workload conditions, such as testing response times, throughput, and resource utilization. 29 Evaluate system scalability and identify potential bottlenecks or performance issues.

7.2.6 Security Testing

Test the system's security measures and vulnerabilities to ensure sensitive data and user information are protected. Identify and address potential security risks, such as unauthorized access or data breaches

7.2.7 Compatibility Testing

Test the system's compatibility across different platforms, devices, and browsers to ensure a consistent user experience. Verify the system's functionality and appearance on various operating systems and screen sizes

7.2.7.1 Regression Testing

`Re-test previously validated functionalities after making changes or adding new features to ensure that existing functionalities are not affected. Detect and address any unintended side effects or regressions in the system.

7.3 TESTING STRATEGY

Implementation is the most crucial stage in achieving a successful system and giving the user's confidence that the new system is effective and workable. Implementation of this project refers to the installation of the package in its real environment to the full satisfaction of the users and operations of the system. Testing is done individually at the time of development using the data and verification is done the way specified in the program specification. In short, implementation constitutes all activities that are required to put an already 30 Test Strategy is also as test approach defines how testing would be carried out.

Test approach has two techniques:

- Proactive - An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.
- Reactive - An approach in which the testing is not started until after design and coding are completed.

CHAPTER 8

SYSTEM IMPLEMENTATION

This section discusses the implementation details declaring each stage to fulfill the main scenario listed above in system description. The purpose of the implementation of the process is to design and create (or fabricate) a system element conforming to that this is an element's design properties and/or requirements. The element is constructed employing appropriate technologies and industry practices. This process bridges the system definition processes and the integration process. Next sub-section clarifies algorithms implemented in the project based on use-cases declared earlier. The implementation of a bus searching app involves several key steps.

8.1 DEFINE REQUIREMENTS

The first step in implementing the app is to define the requirements based on the needs of the users and the business. This involves identifying the features and functionality that the app must have to meet the needs of the users, as well as any technical requirements or constraints.

8.2 DEVELOP A DESIGN

The next step is to develop a design for the app that incorporates the identified requirements. This includes creating wireframes, mockups, and prototypes to visualize the user interface and user experience.

8.3 CHOOSE A DEVELOPMENT PLATFORM

Once the design is finalized, the development platform must be chosen. This could involve choosing between native development for iOS and Android platforms or using cross-platform.

8.4 BUILD AND TEST THE APPLICATION

The development team can build the application using the chosen platform and programming language, while testing the application to ensure it meets the requirements and functions as expected.

8.5 DEPLOY THE APPLICATION

Once the app has been built and tested, it can be deployed to the app stores for iOS and Android, or distributed through an enterprise app store.

8.6 MONITOR AND MAINTAIN

After deployment, the app must be monitored and maintained to ensure it continues to function as expected. This involves ongoing testing, bug fixing and updating the app to meet changing user needs or technical requirements.

8.7 COLLECT AND ANALYZE DATA

Finally, data should be collected and analyzed to understand how the app is being used, identify areas for improvement, and inform future enhancements. Overall, the implementation of a bus searching app involves a systematic approach that requires careful planning, design, development, and maintenance to create a successful product that meets the needs of its users.

CHAPTER 9

CONCLUSION & FUTURE ENHANCEMENT

9.1 CONCLUSION

Integrating into water management at scale represents a revolutionary step in permaculture . Leveraging the power of smart algorithms and advanced technology , this innovation provides accurate water, stability, real-time monitoring and decision-making information and improves efficiency. It is and AI-powered sensor-based system that purifies water according to the specific needs of crops. This optimization reduces water waste, improves resource use, and maximizes crop yields, helping to improve agriculture and food security.

Waste materials and environmental impacts to promote sustainable development. By using water efficiency according to the actual needs of crops, and save water, reduce soil erosion, rescue nutrient leaching, thereby protecting the long-term health of the agricultural ecosystem for production purposes or weather events respond quickly to improve farm management.

The proactive approach reduces lose and optimize productivity and ensures the quality service and farm profitability. By analyzing large amounts of data collected by sensors, the system can identify trends, patterns and relationships to inform strategic decisions that will improve overall agricultural and manufacturing performance. Sensor-based systems represent an agricultural water productivity and data- driven decision making to solve problems of water scarcity and environmental degradation. As global population increases and climate change worsens, such new technologies will play a key role ensures the sustainability and productivity.

9.2 FUTURE ENHANCEMENT

Future enhancements for an AI-driven sensor-based irrigation system involve advancements in technology and methodologies to further improve water management efficiency and agricultural productivity. Firstly, incorporating advanced sensors with enhanced capabilities, such as hyperspectral imaging sensors or drones equipped with multispectral cameras, can provide more detailed and precise data on soil conditions, crop health, and water stress levels. These sensors can offer finer granularity and spatial resolution, allowing for more targeted and efficient irrigation practices.

Secondly, advancements in AI algorithms and machine learning techniques can enable the development of more sophisticated models for predictive analytics and optimization. This includes leveraging deep learning algorithms for more accurate crop yield predictions and reinforcement learning for adaptive irrigation control in dynamic environments. The integrating the irrigation system with emerging technologies like Internet of Things (IoT) and edge computing can enhance real-time data processing and decision-making capabilities. By deploying edge devices at the sensor level, data processing can be decentralized, reducing latency and improving system responsiveness. Furthermore, enhancing the system's scalability and interoperability will be crucial for widespread adoption and integration with other smart farming technologies. This involves developing standardized communication protocols and open-source platforms to facilitate seamless integration with existing agricultural management systems.

CHAPTER 10

APPENDIX

10.1 SOURCE CODE

10.1.1 PYTHON CODE (HISTOGRAM EQUALIZATION)

```
from ultralytics import YOLO
import cv2
import math
import serial
import time
import requests
import base64
import json
import numpy as np
from blynklib import Blynk

#https://blynk.cloud/external/api/update?token=vB-
UzWHaX7ttbrx_xSEbc2_lT_KK6l4f&V1=
BLYNK_SERVER =  "https://blynk.cloud"
BLYNK_AUTH_TOKEN = "vB-UzWHaX7ttbrx_xSEbc2_lT_KK6l4f"
VIRTUAL_PIN = 1

def update_data_to_blynk(value):
    iot_url =
f"{BLYNK_SERVER}/external/api/update?token={BLYNK_AUTH_TOKEN}&V{VIRTUAL
_PIN}={value}"

    try:
        response = requests.get(iot_url)
        if response.status_code == 200:
            print("Data sent successfully to Blynk.")
        else:
            print(f"Failed to send data to Blynk. Status code:
{response.status_code}")
    except Exception as e:
        print(f"Error: {e}")
```

```
# Load pre-trained model for object detection (e.g., Haarcascades for
simplicity)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
# Open webcam
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
# model
model = YOLO("yolo-Weights/yolov8n.pt")
# object classes
classNames = ["person", "bicycle", "car", "motorbike", "aeroplane",
"bus", "train", "truck", "boat",
              "traffic light", "fire hydrant", "stop sign", "parking
meter", "bench", "bird", "cat",
              "dog", "horse", "sheep", "cow", "elephant", "bear",
"zebra", "giraffe", "backpack", "umbrella",
              "handbag", "tie", "suitcase", "frisbee", "skis",
"snowboard", "sports ball", "kite", "baseball bat",
              "baseball glove", "skateboard", "surfboard", "tennis
racket", "bottle", "wine glass", "cup",
              "fork", "knife", "spoon", "bowl", "banana", "apple",
"sandwich", "orange", "broccoli",
              "carrot", "hot dog", "pizza", "donut", "cake", "chair",
"sofa", "pottedplant", "bed",
              "diningtable", "toilet", "tvmonitor", "laptop", "mouse",
"remote", "keyboard", "cell phone",
              "microwave", "oven", "toaster", "sink", "refrigerator",
"book", "clock", "vase", "scissors",
              "teddy bear", "hair drier", "toothbrush"
              ]
#classNames = ["person","cat","dog", "horse", "sheep", "cow",
"elephant", "bear", "zebra", "giraffe",]
try:
    while True:
        success, img = cap.read()
        results = model(img, stream=True)
```

```
        # coordinates
        for r in results:
            boxes = r.boxes
            for box in boxes:
                # bounding box
                x1, y1, x2, y2 = box.xyxy[0]
                x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2) #
convert to int values
                # put box in cam
                cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 255),
3)

                # confidence
                confidence = math.ceil((box.conf[0]*100))/100
                print("Confidence --->",confidence)

                # class name
                cls = int(box.cls[0])
                print("Class name -->", classNames[cls])

                if classNames[cls]=="elephant":
                    update_data_to_blynk(classNames[cls])
                elif classNames[cls]=="bear":
                    update_data_to_blynk(classNames[cls])
                elif classNames[cls]=="giraffe":
                    update_data_to_blynk(classNames[cls])
                elif classNames[cls]=="horse":
                    update_data_to_blynk(classNames[cls])
                elif classNames[cls]=="person":
                    update_data_to_blynk(classNames[cls])
                # object details
                org = [x1, y1]
                font = cv2.FONT_HERSHEY_SIMPLEX
                fontScale = 1
                color = (255, 0, 0)
                thickness = 2

                #gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
                faces = face_cascade.detectMultiScale(img,
scaleFactor=1.3, minNeighbors=5)

                if len(faces) > 0:
                    cv2.putText(img, classNames[cls], org, font,
fontScale, color, thickness)
                    #update_data_to_blynk(classNames[cls])




        cv2.imshow('Webcam', img)
        if cv2.waitKey(1) == ord('q'):
            break
except KeyboardInterrupt:
    cap.release()
    src.close()
    cv2.destroyAllWindows()
```

10.1.2 EMBEDDED C CODE (HARDWARE)

//e2637iot@awgarstone.com

//Info@2024

#define BLYNK_TEMPLATE_ID "TMPL3CZk8Eegs"

#define BLYNK_TEMPLATE_NAME "ANIMAL DETECTED"

#define BLYNK_AUTH_TOKEN "vB-UzWHaX7ttbrx_xSEbc2_lT_KK6l4f"

char auth[] = BLYNK_AUTH_TOKEN;

```
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>


#include <LCD_I2C.h>

LCD_I2C lcd(0x3F);


#include "DHT.h"      // including the library of DHT11 temperature and humidity sensor

#define DHTPIN D7 // Selecting the pin at which we have connected DHT11

#define DHTTYPE DHT11 // Selecting the type of DHT sensors

DHT dht ( DHTPIN, DHTTYPE ) ;


#define moi1 A0

#define rain1 D5

#define pump1 D0


String readdata;


String person = "person";

int read1, i;
```

```
int act1 = 0, act2 = 0, act3 = 0, act4 = 0, act5 = 0, sp;
String data = "";

char ssid[] = "IOT";
char pass[] = "123456789";

int moi, fire, ldr, rain;
int humidity, temp;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(moi1, INPUT);
  pinMode(pump1, OUTPUT);
  digitalWrite(pump1, HIGH);
  dht.begin (  ) ;
  lcd.begin(); // If you are using more I2C devices using the Wire library use lcd.begin(false)
  lcd.backlight();
  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
  lcd.clear();
```

```
}


void loop() {
  humidity = dht.readHumidity ( ) ;
  temp = dht.readTemperature ( ) ;
  lcd.setCursor(0, 0);
  lcd.print("T:");
  if (temp <= 9) {
    lcd.print("00");
    lcd.print(temp);
  }
  else if (temp <= 99) {
    lcd.print("0");
    lcd.print(temp);
  }
  else if (temp <= 999) {
    lcd.print("");
    lcd.print(temp);
  }
```

```
lcd.setCursor(7, 0);
lcd.print("H:");
if (humidity <= 9) {
  lcd.print("00");
  lcd.print(humidity );
}
else if (humidity <= 99) {
  lcd.print("0");
  lcd.print(humidity );

else if (humidity <= 999) {
  lcd.print("");
  lcd.print(humidity );
}
moi = analogRead(moi1);
moi = map(moi, 0, 1024, 100, 0);
if (moi < 10)
{
  moi = 0;
}

delay(400);
```

```
lcd.setCursor(0, 1);
lcd.print("W:");
if (moi <= 9) {
  lcd.print("00");
  lcd.print(moi);
}
else if (moi <= 99) {
  lcd.print("0");
  lcd.print(moi);
}
else if (moi <= 999) {
  lcd.print("");
  lcd.print(moi);
}
rain = digitalRead(rain1);

if (moi > 60)
{
  Blynk.virtualWrite(V4, "WATER LEVEL HIGH");
  Blynk.logEvent("msg", "WATER LEVEL HIGH");
  lcd.setCursor(0, 0);
```

```
    lcd.print("WATER LEVEL          "); // You can make spaces using well... spaces
    lcd.setCursor(0, 1); // Or setting the cursor in the desired position.
    lcd.print("HIGH           ");
    delay(2000);
    lcd.clear();
   }
   else if (moi > 11 && moi < 30)
   {
    Blynk.virtualWrite(V4, "WATER LEVEL LOW");
    Blynk.logEvent("msg", "WATER LEVEL LOW");
    lcd.setCursor(0, 0);
    lcd.print("WATER LEVEL          "); // You can make spaces using well... spaces
    lcd.setCursor(0, 1); // Or setting the cursor in the desired position.
    lcd.print("LOW           ");
    delay(2000);
    lcd.clear();
   }
   else if (rain == 1)
   {
    Blynk.virtualWrite(V4, "RAIN DETECTED");
```

```
    Blynk.logEvent("msg", "RAIN DETECTED");
    lcd.setCursor(0, 0);
    lcd.print("RAIN         "); // You can make spaces using well... spaces
    lcd.setCursor(0, 1); // Or setting the cursor in the desired position.
    lcd.print("DETECTED             ");
    delay(2000);
    lcd.clear();
   }
   else
   {
    Blynk.virtualWrite(V4, "                    ");
   }
   Blynk.virtualWrite(V2, "W: " + String(moi) + "T: " + String(temp) + "H: " + String(humidity));
   Blynk.run();
  }
  BLYNK_WRITE(V3)
  {
   int button = param.asInt(); // read button
   if (button == 1)
   {
    digitalWrite(pump1, LOW);
```

```
  }
  else
  {
    digitalWrite(pump1, HIGH);
  }
}
BLYNK_WRITE(V1)
{
  String button = param.asString(); // read button

  readdata = button;
  for (i = 0; readdata[i] != '\0'; i++)
  {
    if (readdata[i] == person[i]) {
      act1 = 1;
    }
    else  {
      act1 = 0;
      break;

    }
```

```
}


if (act1 == 1)
{
  Blynk.virtualWrite(V0, "                  ");
}
else
{
  Serial.println(readdata);
  delay(500);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(readdata); // You can make spaces using well... spaces
  Blynk.logEvent("msg", readdata);
  Blynk.virtualWrite(V4, readdata);
  Blynk.logEvent("msg", readdata);
  delay(5000);
  Blynk.virtualWrite(V4, "                  ");
  lcd.setCursor(0, 0);
```

10.2 SCREENSHOTS

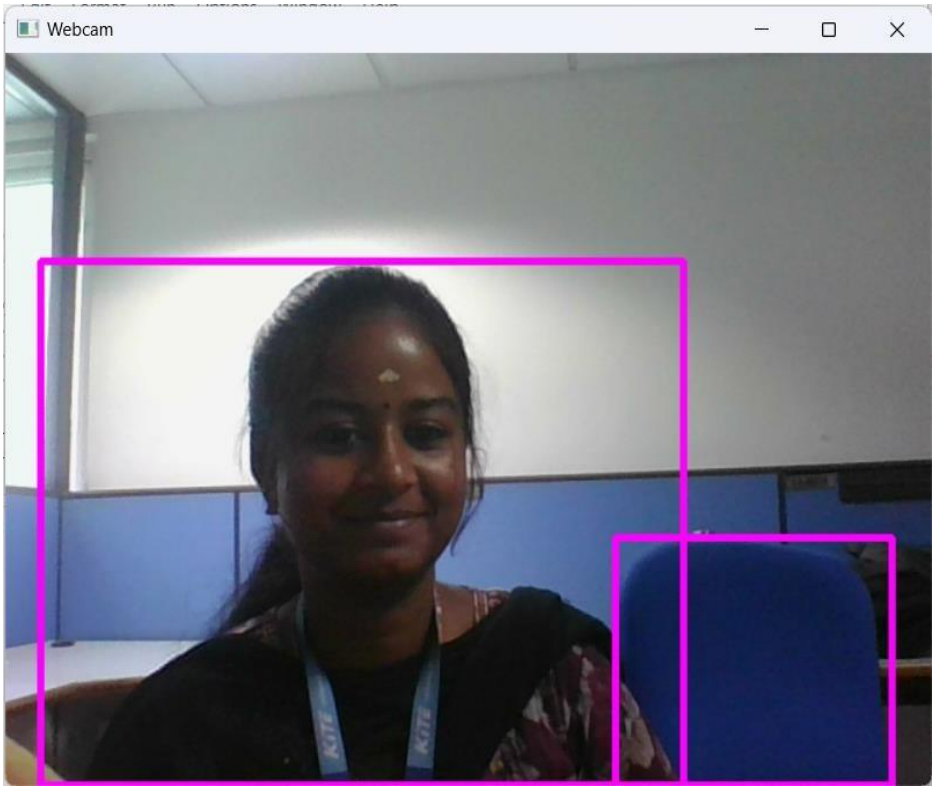

FIG 10.2.1: Histogram Equalizitation

```
Confidence ---> 0.85
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.85
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.51
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.5
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.41
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.31
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.29
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.29
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.26
Class name --> person
```

Ln: 508 Col: 0

FIG 10.2.2: Confidence of the Image

```
Confidence ---> 0.85
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.8
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.59
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.3
Class name --> chair
Confidence ---> 0.29
Class name --> person
Data sent successfully to Blynk.
0: 480x640 4 persons, 1 chair, 236.4ms
Speed: 6.0ms preprocess, 236.4ms inference, 1.8ms postprocess per image at shape
 (1, 3, 480, 640)

Confidence ---> 0.85
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.85
Class name --> person
Data sent successfully to Blynk.
Confidence ---> 0.51
Class name --> person
Data sent successfully to Blynk.
```

FIG 10.2.3: Confidence and speed of the Image

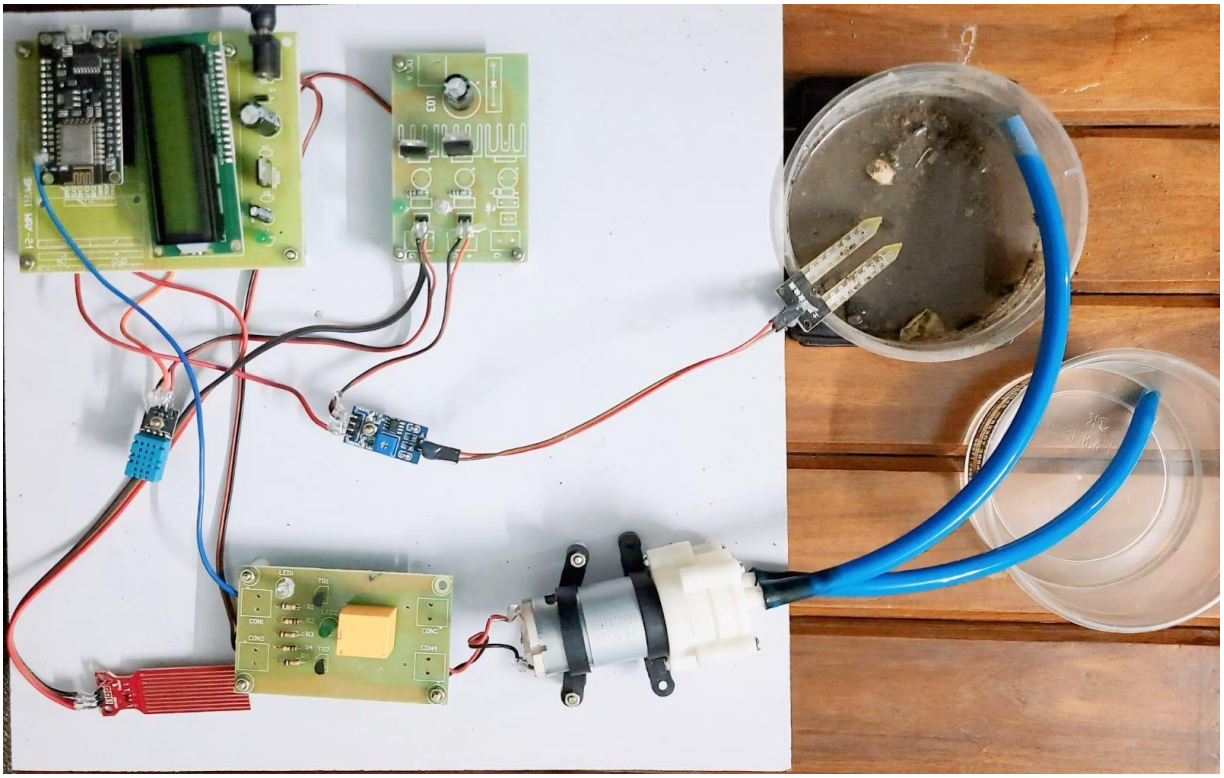

FIG 10.2.4: Hardware Assembly

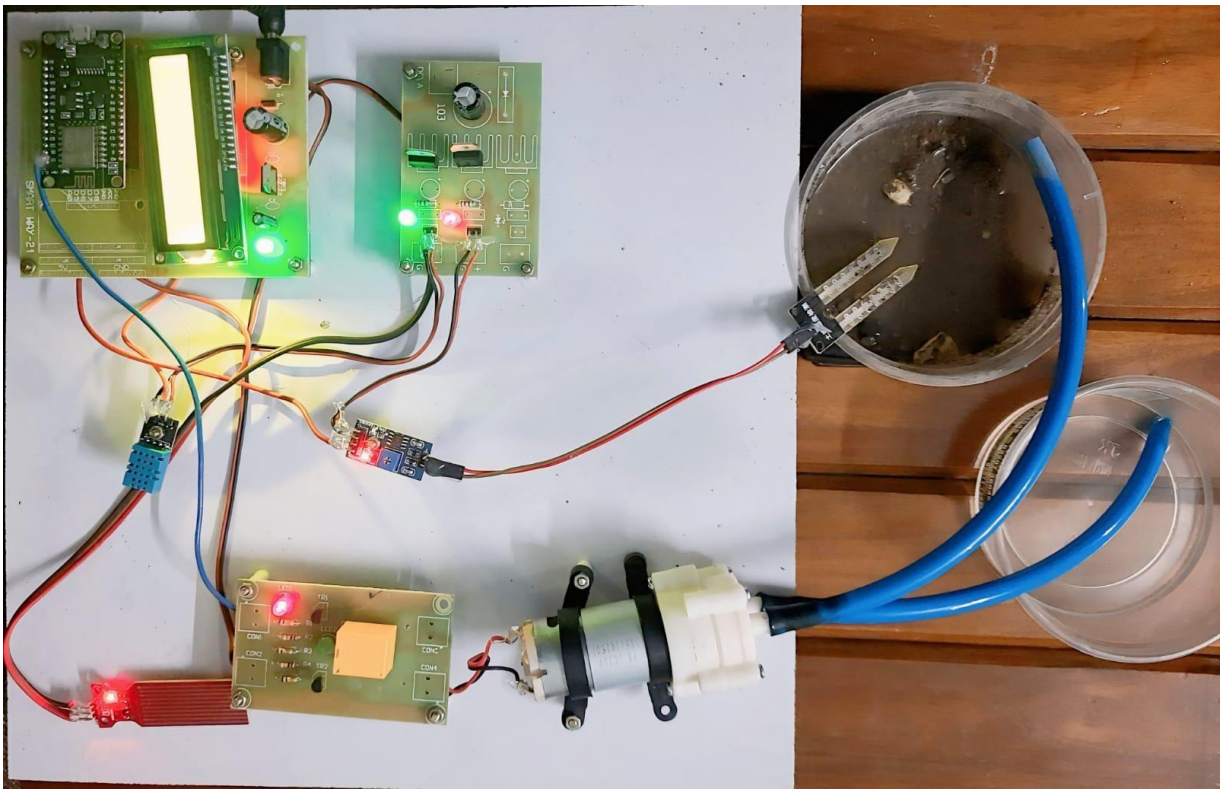

FIG 10.2.5:Hardware Implementation

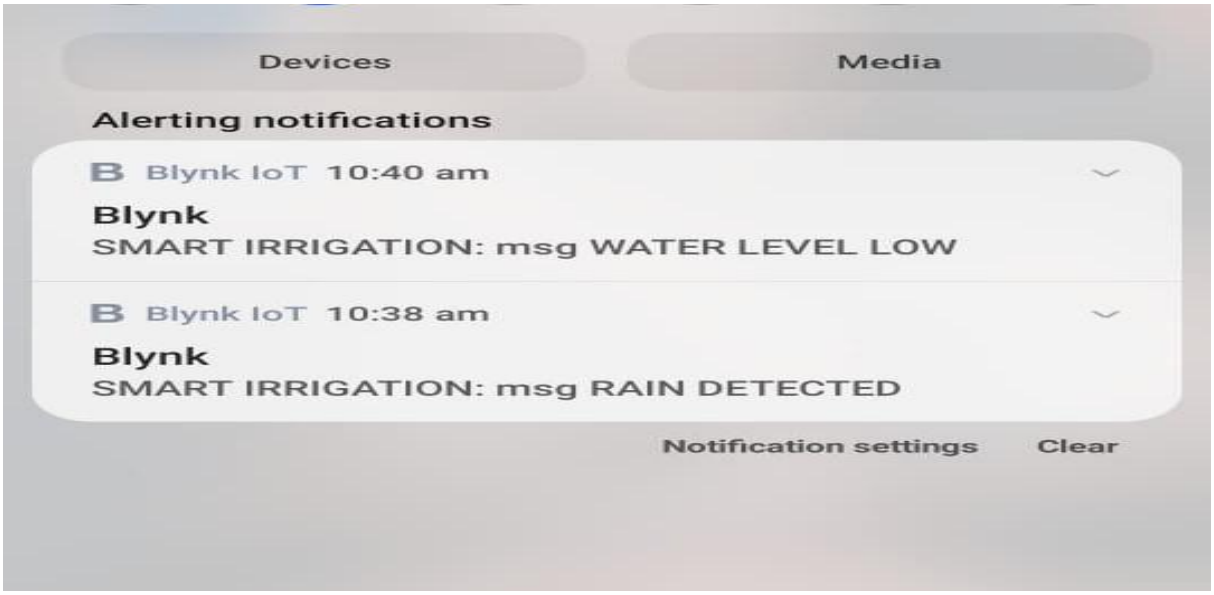

FIG 10.2.6: Mobile notifications pop up

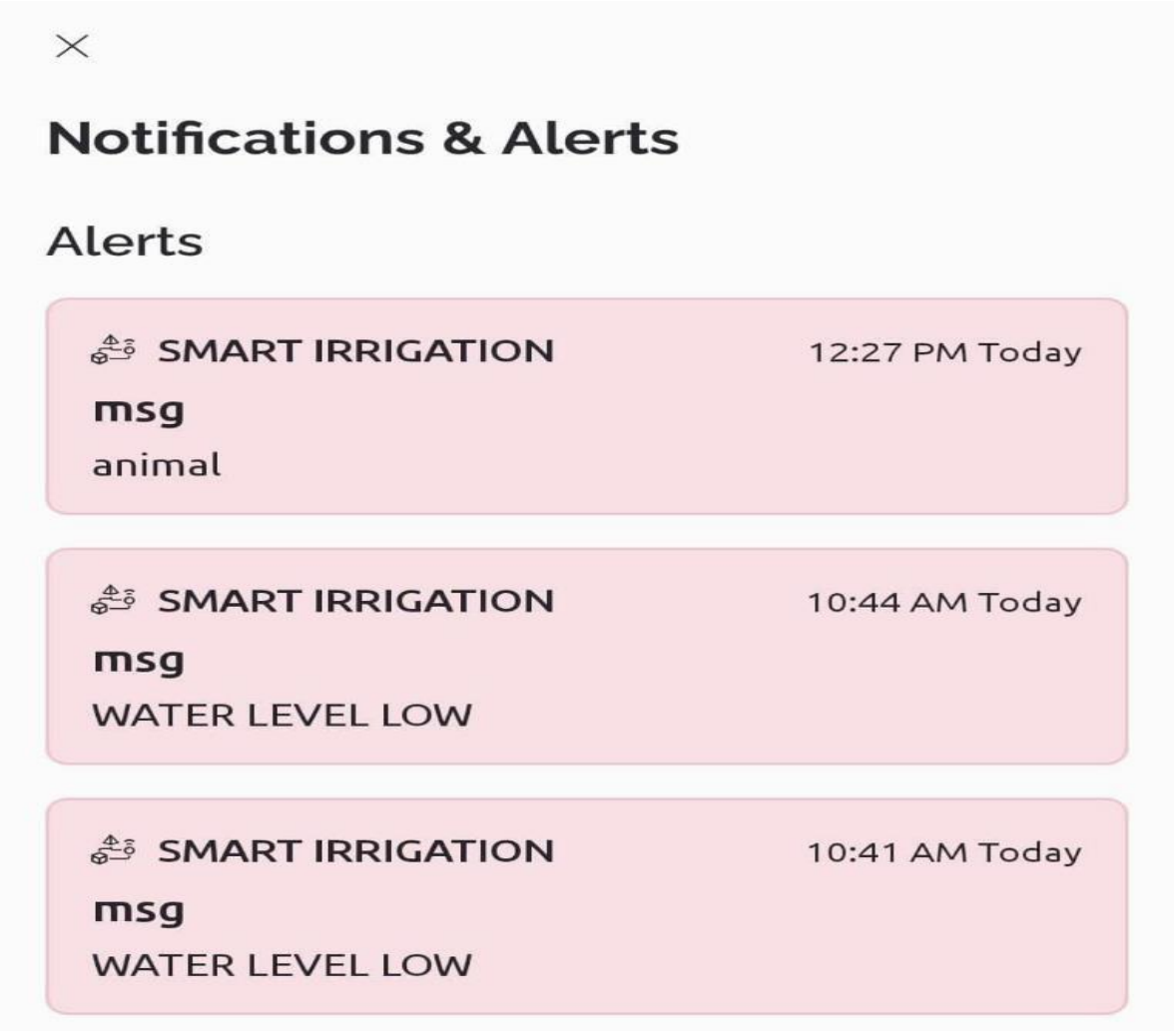

FIG 10.2.7: Notification history in BLYNK APP

CHAPTER 11

REFERENCES

[1] Ali, M., Bhatti, A., & Iqbal, R. (2019). Doorstep collection of household plastic waste: A case study of a UK waste management company. Waste Management & Research, 37(8), 853-860.

[2] Parker, L., Tolfrey, K., & Halog, A. (2021). Plastic waste management in low- and middle-income countries: a systematic review of barriers and enablers. Journal of Environmental Management, 298, 113540.

[3] Hopewell, C., Kosior, E., & Bull, S. (2019). Sustainable plastic waste management towards a circular economy: the role of technology, policy, and society. Science of the Total Environment, 658, 1143-1151.

[4] Nguyen, L., Le, Q. T., & Vu, T. T. (2020). Door-to-door collection of plastic waste in an urban area: A pilot study in Hanoi, Vietnam. Resources, Conservation and Recycling, 161, 104966.

[5] Escobar, N., Zuniga, E., & Mora, E. (2021). Evaluation of a Door-to-Door Plastic Waste Collection Pilot Program in a Rural Community in Costa Rica. Sustainability, 13(4), 2132.

[6] Chavannes, C., Vellinga, T. V., & Breure, A. M. (2020). Do doorstep waste collection schemes increase recycling rates and reduce residual household waste? A case study of Amsterdam. Journal of Cleaner Production, 249, 119345.

[7] Hossain, M. S., & Rahman, M. M. (2019). Improving the recycling rate of plastic waste through doorstep collection in Bangladesh. Resources, Conservation and Recycling, 148, 144-153.