

MAR IVANIOS COLLEGE (AUTONOMOUS)

Mar Ivanios Vidya Nagar, Nalanchira

Thiruvananthapuram – 695015

B.Sc. Computer Science Major Project Report

SMART RAINWATER HARVESTING SYSTEM WITH AUTOMATED IRRIGATION

A report submitted in partial fulfilment of the requirement for the Sixth Semester
BSc. Computer Science



SUBMITTED BY

Gayathri.A.M **2220803**

Akshaykumar V J **2220814**

Ricky Paul.N **2220825**

Under the guidance of

Dr. Anitha.K.L

DEPARTMENT OF COMPUTER SCIENCE

BSc Computer Science

2025

MAR IVANIOS COLLEGE (AUTONOMOUS)

Mar Ivanios Vidya Nagar, Nalanchira

Thiruvananthapuram – 695015

DEPARTMENT OF COMPUTER SCIENCE



CERTIFICATE

This is to certify that the project entitled “SMART RAINWATER HARVESTING SYSTEM WITH AUTOMATED IRRIGATION” is a bonafide record of the work done by Akshaykumar V J (2220814), Gayathri.A.M (2220803), RickyPaul.N (2220825) in partial fulfilment of the requirements for the award of the Degree of Bachelor of Science in Computer Science by the University of Kerala.

INTERNAL GUIDE

HEAD OF THE DEPARTMENT

EXTERNAL EXAMINERS

1.

2.

ACKNOWLEDGEMENT

With the profound gratitude to the **ALMIGHTY**, I take this chance to thank people who helped us to complete this project.

We take this as a right opportunity to say THANKS to our **PARENTS** who are there to stand with us always with the words “YOU CAN”.

We are thankful to Principal **Dr. MEERA GEORGE, MAR IVANIOS COLLEGE** who gave us the platform to establish ourselves to reach greater heights.

We earnestly thank **Dr. K. OOMMACHAN**, Director, who always encourages us to do novel things and providing all facilities.

We express our sincere gratitude to **Mrs. TINU C PHILIP**, Head, Department of Computer Science, for her valuable guidance and support to execute all incline in learning.

It is our delight to thank our project guide **Dr. ANITHA K.L**, Assistant Professor, Department of Computer Science for her help, support, encouragement, suggestions and guidance throughout the development phases of the project.

We convey our gratitude to all the faculty members of the department who extended their support through valuable comments and suggestions during the reviews.

A great note of gratitude to friends and people who are known and unknown for us and helped in carrying out this project work a successful one.

AKSHAYKUMAR V J

GAYATHRI.A.M

RICKY PAUL.N

ABSTRACT

Smart Rainwater Harvesting System with Automated Irrigation project presents an IoT (Internet of Things) - based smart rainwater harvesting system is designed to collect and manage rainwater from rooftops in a residential street. The system uses a rain sensor to detect rain and provides data to automatically open the lid of a centralized tank, enabling the collection of rainwater from individual households. The acid level in the water is analysed using a pH sensor. The water level is also analysed in real-time. Based on the quantity of water it is pumped into individual household tanks for non-potable uses. The system features real-time water level monitoring using ultrasonic sensors, displayed on an LCD screen and a mobile application. The app provides users with updates on rain detection, tank status, water quality, and quantity pumped into their household tanks. Additionally, the system integrates a drip irrigation system for street plants, promoting sustainable urban gardening practices. The mobile app includes a hierarchical access system, allowing administrators to monitor the entire system and individual users to access information about their household tank. The compact sensor package integrates all necessary sensors, making it a comprehensive and user-friendly solution. The study investigates the quality of harvested rainwater, water usage patterns, and the effectiveness of the smart rainwater harvesting system.

TABLE OF CONTENTS

1. INTRODUCTION.....	1
1.1 PURPOSE OF THE PROJECT.....	1
1.2 OBJECTIVE.....	2
2. LITERATURE SURVEY	3
3. SYSTEM SPECIFICATIONS.....	5
3.1 SOFTWARE REQUIREMENTS.....	5
3.1.1 Arduino IDE	5
3.1.2 Firebase.....	5
3.1.3 Android studio	6
3.1.4 VS Code.....	7
3.2 HARDWARE REQUIREMENTS	7
3.2.1 ESP32 Microcontroller	7
3.2.2 Rain sensor	7
3.2.3 Ultrasonic sensor	7
3.2.4 Soil hygrometer	7
3.2.5 Water pump	8
3.2.6 5V Relay Modules	8
3.2.7 20x4 12C LCD	8
3.3 LANGUAGE DESCRIPTION	8
3.3.1 Kotlin.....	8
3.3.2 Embedded C	9
3.3.3 MySQL	9
3.3.4 PHP.....	10
4. SYSTEM ANALYSIS.....	12
4.1 EXISTING SYSTEM.....	12
4.2 PROPOSED SYSTEM.....	12
4.3 FEASIBILITY STUDY.....	13

4.3.1 Technical Feasibility	13
4.3.2 Social Feasibility	14
4.3.3 Operational Feasibility	14
4.3.4 Economic feasibility	15
5. SYSTEM DESIGN.....	16
5.1 ARCHITECTURE DIAGRAM.....	16
5.2 CIRCUIT DIAGRAM	17
5.3 TABLE DESIGN.....	18
5.4 DATAFLOW DIAGRAM.....	20
5.5 USE CASE DIAGRAM	21
6. MODULE DESCRIPTION	22
6.1 MAIN TANK SYSTEM.....	22
6.2 AUTOMATED WATER COLLECTION MODULE	22
6.3 DRIP IRRIGATION MODULE.....	22
6.4 USER MODULE.....	23
6.5 ADMIN MODULE	23
7. SYSTEM IMPLEMENTATION.....	25
7.1 SOURCE CODE	25
7.1.1 Arduino code	25
7.1.2 Login page	30
7.1.3 Connection establishment.....	31
7.1.4 view_analysis	31
7.1.5 Homefragment.....	33
8. SYSTEM TESTING.....	39
8.1 UNIT TESTING	39
8.2 INTEGRATION TESTING	39
8.3 SYSTEM TESTING.....	39
8.4 VALIDATION TESTING.....	40
8.5 BLACK BOX TESTING	40

8.6 WHITE BOX TESTING	40
8.7 TEST CASES	40
8.7.1 IoT system	40
8.7.2 Mobile Application.....	42
9. CONCLUSION	44
10. FUTURE ENHANCEMENTS	45
11. REFERENCES	46
12. APPENDIX	47
12.1 UI SCREENSHOTS	47
13. USER MANUAL	52
13.1 INTRODUCTION	52
13.2 GETTING STARTED	52
13.2.1 System Components	52
13.4 TURNING ON/OFF THE DEVICES	52
13.4.1 Turning On	52
13.4.2 Turning Off.....	52
13.5 INSTALLATION GUIDE.....	53
13.6 TROUBLESHOOTING	53
13.7 SAFETY AND MAINTENANCE	54

LIST OF FIGURES

Figure 5.1 Architecture Diagram.....	16
Figure 5.2 Circuit Diagram.....	17
Figure 5.4.1 Context Diagram	20
Figure 5.4.2 Level 1 user	20
Figure 5.4.3 Level 1 admin.....	20
Figure 5.5.1 Use case diagram-user	21
Figure 5.5.2 Use case diagram-admin	21
Figure 12.1.1 Splash Page.....	47
Figure 12.1.2 Signup page.....	47
Figure 12.1.3 Login Page	48
Figure 12.1.4 Reset Password	48
Figure 12.1.5 Admin Homepage	49
Figure 12.1.6 Analysis Page.....	49
Figure 12.1.7 View User Page.....	50
Figure 12.1.8 LCD Screen.....	50
Figure 12.1.8 Rain Sensor	51

LIST OF TABLES

Table 1 user_details	18
Table 2 sensor_data	19
Table 3 test_cases_mobile_application	40
Table 4 test_cases_mobile application	42

LIST OF ABBREVIATIONS

IoT: Internet of Things

pH: Potential of Hydrogen

MySQL: My Structured Query Language

I/O: Input/Output

PHP: Hypertext Preprocessor

API: Application Programming Interface

IDE: Integrated Development Environment

ESP32: Expressif32

1. INTRODUCTION

In the modern world, water shortages and ineffective water management are major issues. The Smart Rainwater Harvesting System with Automated Irrigation uses Internet of Things technology to develop an effective and sustainable way to conserve and use water in order to solve the water scarcity related problems.

A centralized rainwater collection tank in this creative system is intended to gather rainfall from the rooftops of several homes along a street. To ensure optimal water capture, the centralized tank opens automatically when it rains. The device continuously checks the pH and tank water level in real time using integrated Internet of Things (IoT) sensors. A mobile application receives this data and uses it to give users information about the availability and quality of water. Everyday usage of the collected rainwater is made possible by its effective distribution to residential water tanks.

The system also has a drip irrigation feature that maximizes water efficiency while boosting urban greenery by automatically watering the street's vegetation with the water from the centralized tank. House gardens might employ the same strategy.

In order to help users properly monitor and manage their water resources, the mobile application also offers statistics, including quality reports, water usage trends, and the amount of rainfall collected. This system provides a comprehensive way to improve water efficiency, lessen reliance on outside water sources, and maintain urban ecosystems by fusing IoT technology with sustainable water management techniques. In addition to encouraging environmental sustainability, the Smart Rainwater Harvesting System gives communities the tools they need to actively solve water-related issues.

1.1 PURPOSE OF THE PROJECT

- **Water Conservation:** By collecting rainfall and reusing it, the main goal is to lessen reliance on conventional water sources like municipal supplies.
- **Sustainable Gardening:** By using the collected rainwater for gardening, plants are given a steady supply of water and the need for tap water is reduced.

- **Rainwater Collection:** The system uses natural rainfall for irrigation and household use by automatically collecting rainwater using pipelines that directs the water into a storage tank.
- **Effective Storage:** The rainwater is kept in a storage tank. The tank provides an automated lid to control access to the water and keep it hygienic.
- **Cost-Effective:** By using rainwater for gardening and household purposes, the system helps in reducing water bills, saving money and encouraging environment friendly water use.
- **Automation for Convenience:** By automating processes like opening and shutting the tank lid or turning on the watering system, the app may make the entire process more effective and labour-efficient.

1.2 OBJECTIVE

By installing a smart rainwater harvesting system and conducting a pH study, the project's main goal is to improve easy water management. The project intends to maximize the gathering and utilization of rainwater through a centralized water harvesting system using IoT technology and automated monitoring.

The idea uses sensors to track pH in real-time. The technology can notify people if the water is overly acidic because the data is analysed in real-time.

By using an automatic drip irrigation system for the street plants, water wastage can be reduced and greenery can be maintained at a lower cost without the need for labour. The project also intends to solve particular issues with urban water scarcity and environmental sustainability. By integrating quality monitoring and intelligent rainfall collecting, the system enables proactive water management while minimizing the negative environmental effects of water extraction and the strain on municipal water supplies. This encourages a shift to greener urban development and aids in water conservation by empowering consumers to adopt sustainable practices. The ultimate goal of the project is to construct an urban infrastructure that is more resource-efficient supporting sustainable living and environmental care.

2. LITERATURE SURVEY

Rainwater can be captured using the rainwater harvesting system. Generally, rainwater harvesting system is the direct collection of rainwater from roofs and other surfaces to built catchments, the collection of sheet runoff from man-made ground or natural surface catchments and rock catchments for domestic, industry, agriculture and environmental use. Adopting the concept of sustainability and conservation of water resources can help to cope with the global water shortage. A rainwater harvesting system is one of the concepts that can be implemented to meet the water shortage problem [1]. The author have proposed an Internet of Things (IoT) based innovative solution for rainwater harvesting. Harvested rainwater is often contaminated with many things. Here water is classified while harvesting, and for that a designed smart first flush diverter, some sensors and ESP32 SoCs have been used to run the whole system. As a result, the proposed system can be a practical and useful solution for harvesting rainwater in households with ensuring water quality [2]. This paper addresses provide a technique and nanotechnology based on IoT to satisfy the demands for monitoring water quality and providing water (Internet of Things). This study demonstrate a method for collecting rainwater in rural areas as well as an Internet of Things model which is used for monitoring water quality and level. The collected data via sensors are made available for real-time viewing on a website. The proposed system contains an Arduino as the main controller along with a number of sensors used are water level sensor, pH sensor, and water flow. A microcontroller processes the data it receives from the sensors and then uses a wireless connection module to send it to the cloud. [3]. This paper presents an IoT based model for water level and quality monitoring. The data will be collected using sensors and accessed on real time basis through website. The proposed system consist different sensors like water flow sensor, pH sensor, water control valve, water level sensor and a raspberry PI 0W as a core controller. The data received from the sensors is processed by a microcontroller and sent to the cloud communication module [4]. The paper proposes a Smart Rainwater Harvesting system is developed that uses a water treatment approach. The proposed system focuses on treating the unusable water that is collected and treated using sodium hydroxide and soda ash. The entire process is also connected using webpages that helps the user for an effortless. These enhancements make our model much yielding and upgraded compared to the traditional approach. The resultant water collected can be used in domestic, Industrial, economic and also for potability [5]. The paper gives an Earth Observing (EO) information put together methodology with respect to creating techniques to supply data that bolsters progression in water gathering advances of the horticultural croplands

inside the semi-bone-dry conditions. The final word objective of the examination is to help advance harvest profitability (efficiency per unit of land) and water inside the downpour took care of territories of semi-bone-dry conditions bringing about expanded nourishment creation and nourishment security [6]. The work proposed a smart sensor interface device that can sense the water quality parameters and effectively generate data in an online system for showing real-time measures of water quality parameters. It integrates drinking water quality measurement by different types of sensors. These sensors relate to Arduino for the purpose of monitoring the parameters of water quality. For transmitting the values, we create a serial communication between Arduino and NodeMCU which will show the data on an online system (web interface). A QR code will be attached with every water source for accessing it easily by any user. They can scan the QR to get sure whether the water is safe to drink or not. The Government can use this system to get information of area's water quality. This system can be also implemented on agriculture related applications and in industrial fields. The design, development and implementation of an IoT based system will help the authorities take the necessary steps to perform proper solutions for the affected area.

3. SYSTEM SPECIFICATIONS

3.1 SOFTWARE REQUIREMENTS

3.1.1 Arduino IDE

Water quality analysis and the development and implementation of an intelligent rainwater collecting system both heavily rely on the Arduino IDE (Integrated Development Environment). Many smart systems use Arduino-compatible microcontrollers as their central control units, and the Arduino IDE provides an open-source environment for developing and uploading code to these microcontrollers. The Arduino microcontroller serves as the brain of a rainwater harvesting system, handling sensor data and regulating the functioning of pumps, valves, and other mechanical parts. The Arduino IDE enables developers to create code that automates the gathering, storing, and distributing of rainwater in the context of a smart rainwater harvesting system. By linking a number of sensors, including water level, soil moisture, and rain sensors - to an Arduino board, the system can make real-time decisions based on environmental data.

A pH sensor is integrated into the Arduino IDE to monitor pH for water quality analysis. These sensors provide data to the Arduino microcontroller, which evaluates it and can send out alerts if the water pH is too high. The Arduino IDE allows programmers to modify code to meet particular needs, like establishing sensor thresholds and notifying linked devices.

Multiple inputs and outputs can be managed more easily with the help of libraries that make sensor integration simpler in the Arduino IDE. Furthermore, data from the rainwater harvesting and quality monitoring systems may be accessible remotely because of the IDE's support for connection protocols like Wi-Fi and Bluetooth. Because of this connectivity, customers can use a smartphone to track system performance, get warnings, and examine trends.

3.1.2 Firebase

As a real-time database for communication, Firebase is essential to water quality analysis and intelligent rainwater harvesting systems. All of the data gathered by the Internet of Things sensors used in the rainwater harvesting system can be stored in a real-time database offered by Firebase.

The system can instantly update sensor data using Firebase's real-time database, giving users and automated systems access to the most recent data. This is essential for making proactive decisions, such setting off alarms in the event that storage levels fill up or water quality drops below acceptable limits.

The seamless integration of Firebase with mobile applications allows for the development of appealing system interfaces. Users have access to a dashboard that shows real-time water levels, water quality metrics, historical data trends and analytics, helping them make informed decisions about water usage and maintenance.

With Firebase's support for analytics, users can gain insights into water consumption patterns, quality fluctuations, and other critical metrics. This information can be used to optimize the rainwater harvesting system's performance over time.

3.1.3 Android studio

A free and open-source Integrated Development Environment (IDE) made especially for creating Android apps is called Android Studio. It offers an extensive collection of tools for creating, evaluating, and debugging Android apps. Android Studio has emerged as the preferred platform for Android app developers due to its user-friendly interface and extensive feature set. With its many capabilities, such as code editing, debugging, and project management, the IDE is a vital tool for creating Android apps of the finest quality. Android Studio is essential to our IoT-based rainwater harvesting project since it helps us create the mobile application that enhances the system. Through the app, users can track rainwater collection, analyze water quality in real time, and keep an eye on patterns in water usage. The app also sends notifications and alerts when water levels are low or water quality parameters exceed safe limits. Rapid development and prototyping are made possible by Android Studio's powerful capabilities and user-friendly interface, guaranteeing that the app is effective and usable. Our developers may utilize Android Studio to produce a complete mobile application that gives customers real-time statistics and insightful information to help them optimize their water usage.

It reduces waste and encourages sustainable water management techniques by empowering users to make knowledgeable decisions about how much water they consume. Our developers can guarantee a seamless user experience by using Android Studio to make sure the app is scalable, safe, and simple to maintain. We can develop an innovative mobile application that advances the objective of our project, which is to promote effective and sustainable water management techniques, by utilizing Android Studio.

3.1.4 VS Code

Visual Studio Code (VSCode) is a free, open-source code editor developed by Microsoft. It's lightweight yet powerful, supporting multiple programming languages like Python, JavaScript, and C++. With features like intelligent code completion, debugging, and Git integration, it enhances developer productivity. VSCode's extensive marketplace offers numerous extensions for customization, enabling linting, theming, and language support. Its cross-platform compatibility (Windows, macOS, Linux) and built-in terminal make it a versatile tool. Popular for web development, VSCode offers live preview and collaborative coding with Live Share. Its active community and frequent updates ensure continuous improvement and relevance in modern development.

3.2 HARDWARE REQUIREMENTS

3.2.1 ESP32 Microcontroller: The ESP32 is a low-power, low-cost microcontroller developed by Espressif Systems, featuring a dual-core CPU, Wi-Fi, Bluetooth, and a range of peripherals, making it an ideal choice for IoT and robotics projects. With its high-performance capabilities, flexibility, and ease of use, the ESP32 has become a popular choice among makers, hobbyists, and professionals for developing innovative and connected devices.

3.2.2 Rain sensor: A rain sensor is a device that detects the presence of rain or moisture, typically using infrared or capacitive sensing technology to provide a digital or analog output signal. Rain sensors are commonly used in various applications, including weather stations, automatic windshield wipers, and building automation systems, to detect and respond to rainfall or moisture.

3.2.3 Ultrasonic sensor: An ultrasonic sensor uses high-frequency sound waves to measure distance, level, or proximity of objects, and is commonly used in applications such as robotics, automation, and level sensing. Ultrasonic sensors work by emitting ultrasonic waves and measuring the time-of-flight or echo time to calculate the distance or level of the target object.

3.2.4 Soil hygrometer: A soil hygrometer is a device used to measure the moisture content or humidity of soil, typically using electrical resistance or capacitance sensors to provide an accurate reading of soil moisture levels. Soil hygrometers are commonly used in agriculture,

horticulture, and environmental monitoring applications to optimize irrigation schedules, prevent waterlogging, and promote healthy plant growth.

3.2.5 Water pump: A water pump is a mechanical device that uses energy to raise, transfer, or compress water from one location to another, often used in applications such as irrigation, drinking water supply, and wastewater management. Water pumps come in various types, including centrifugal, submersible, and positive displacement pumps, each suited for specific uses and flow rates.

3.2.6 5V Relay Modules: A 5V relay module is an electronic device that uses a small input voltage (5V) to control a high-current output, typically used to switch on/off devices such as motors, pumps, and lights. The 5V relay module is commonly used in microcontroller-based projects, such as Arduino and Raspberry Pi, to control external devices and expand the project's capabilities.

3.2.7 20x4 12C LCD: A 20x4 12C LCD (Liquid Crystal Display) is a type of display module that can show 4 lines of 20 characters each to display text-based information.

3.3 LANGUAGE DESCRIPTION

3.3.1 Kotlin

Kotlin, a modern programming language developed by JetBrains, is gaining traction in the development of smart IoT systems, including applications in rainwater harvesting and water quality analysis. Its concise syntax, interoperability with Java, and support for Android development make Kotlin an ideal choice for building mobile and web applications that manage and monitor smart rainwater systems.

Kotlin can play a critical role in managing IoT devices such as rain sensors, water level detectors, and quality monitoring sensors that are essential to a smart rainwater harvesting system. Kotlin's powerful asynchronous programming capabilities using coroutines enable efficient handling of real-time data collected from sensors, ensuring the system can respond to changes in water levels or quality without latency.

Kotlin's primary use in the development of Android applications, which can provide a user-friendly interface to monitor the performance of a rainwater harvesting system. With Kotlin's clean and readable codebase, developers can quickly create mobile apps that visualize data from the sensors, such as water collection levels, filtration status, and water quality metrics.

With Kotlin's support for cloud-based applications, developers can easily integrate rainwater harvesting systems with Firebase. This integration allows data from local IoT devices to be sent to cloud servers for advanced analytics, storage, and long-term trend analysis.

3.3.2 Embedded C

Embedded C plays a crucial role in the development of a smart rainwater harvesting system and water quality analysis by serving as the primary programming language for the microcontrollers and embedded devices that control and monitor the system. In such systems, Embedded C is used to write the firmware that drives the core functionality, enabling precise control over sensors, actuators, data processing, and communication modules.

In a smart rainwater harvesting system, multiple sensors are employed to measure various parameters such as water level, flow rate, and environmental conditions (like humidity and rainfall). Embedded C is used to program the microcontroller to read data from these sensors. The code ensures accurate and timely data collection.

Embedded C is used to program the system's decision-making logic. Microcontroller can be programmed to trigger certain actions based on sensor readings, such as opening or closing valves, starting or stopping pumps, or diverting water to storage tanks when the water quality meets specific criteria. These automated decisions are crucial for efficient water collection, storage, and usage.

3.3.3 MySQL

MySQL, a widely-used open-source relational database management system, plays a crucial role in smart rainwater harvesting systems and water quality analysis. In the context of

these systems, MySQL serves as a reliable and efficient database solution for storing, managing, and retrieving data collected from various sensors and monitoring devices. It provides a centralized platform to handle large datasets, making it easier to analyze trends, make informed decisions, and ensure the effectiveness of water management processes.

In a smart rainwater harvesting system, data is continuously collected from sensors measuring parameters like rainfall, water levels, flow rates, and water quality indicators such as pH, turbidity, salinity and water level in tank. MySQL databases are used to store this data in a structured format, allowing for easy access and retrieval. The system can handle high frequency data inputs from multiple sources, ensuring that all relevant information is accurately captured and stored.

One of MySQL's advantages in smart rainwater harvesting systems is its flexibility and scalability. As the system evolves or expands, MySQL databases can be easily modified to accommodate new sensors, additional data parameters, or increased data storage requirements.

MySQL's ability to handle complex data structures enables the integration of multiple water sources or harvesting systems, providing a unified database to manage comprehensive water quality and collection data.

3.3.4 PHP

PHP (Hypertext Preprocessor) is a server-side scripting language that plays a pivotal role in creating RESTful APIs (Application Programming Interfaces) for IoT projects, particularly for pushing data from Firebase to a MySQL database. As a server-side scripting language, PHP is executed on the server-side, allowing it to interact with the MySQL database and perform data processing tasks. This enables the creation of dynamic web content and web applications that can efficiently manage and analyse data from various sources.

In the context of this IoT project, PHP is utilized to develop RESTful APIs that provide a standardized interface for data exchange between the Firebase Realtime Database and the MySQL database. This involves creating API endpoints that can handle HTTP requests, such as GET, POST, PUT, and DELETE, to perform CRUD (Create, Read, Update, Delete) operations on the data. PHP's ability to interact with databases using PDO (PHP Data Objects) extension provides a standardized interface for database operations, making it an ideal choice

for this project. Its server-side scripting capabilities, flexibility, and rapid development capabilities make it an ideal choice for this project.

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

The existing systems rely on manual intervention for turning the system on or off.

- Gutter and downspout system: Collects runoff water through a channel and collect it down to grade level.
- Storage tanks: A buried tank where rainwater is stored.
- Automated system collecting and pumping rainwater into individual storage: Uses a water pump to share water with the household tanks.
- Individual drip irrigation system: The use of sensors and controllers help to optimize the irrigation schedule for plants in gardens.

4.2 PROPOSED SYSTEM

Proposed system consists of automated storage tank which collects rainwater across the street. There is a centralized tank which is connected to the terraces of the households in the street. Rainwater collected on these terrace is directed into the main tank where water is collected and stored. A rain sensor is placed on top of the tank lid so that it detects the rain and automatically opens up the lid. The water pH inside the tank is continuously monitored and if any variation is found, then the pH value is updated on the mobile application along with an alert. The water level in the tank is analysed continuously using an ultrasonic sensor so that the amount of water used and collected can be identified and the data can be sent to the database for further storage. As the rain ends, the system shuts the lid automatically. If the tank is full, water is redirected to underground without causing an issue of overflowing. Water is pumped into individual water tanks placed in houses for usage. The compact sensor package integrates all the necessary sensors, making it comprehensive and user friendly solution.

The system also helps in watering the plants/trees in the area with the help of an automated irrigation system. Through the established mobile application, the admin can determine whether the drip irrigation system should operate manually or automatically.

The mobile application is useful in getting the information about the rainwater harvesting system. Data such as the tank lid status (open/closed), rain status (yes/no), water level, water pH, drip irrigation status (on/off) etc can be identified. There will be an admin who can monitor the complete system through the mobile application. When the tank is empty the user gets an alert on the mobile application that the tank is empty and the user should use their household pipeline water. Project result showed the analysis study of water harvesting patterns, usage patterns and the impact of water saving both economically and environmentally.

4.3 FEASIBILITY STUDY

A feasibility study is carried out before the project is developed to assess the likelihood that the suggested system will be successful. A feasibility study is necessary to determine whether creating a new or improved system is compatible with costs, benefits, operation, and technology.

4.3.1 Technical Feasibility

- **Centralized Tank Setup:** It is possible to collect rainwater from several homes along a street and store it in a central tank. In order to guarantee that larger material is filtered out in the early stages, plumbing connections with the proper filters are needed. Pipe and storage infrastructure must be carefully designed to guarantee optimal flow and prevent leaks.
- **Data Aggregation:** Firebase receives the data gathered by these sensors for real-time analysis, and MySQL eventually stores the data.
- **ESP32:** An excellent option for this project is the ESP32. Its integrated Wi-Fi is crucial for transmitting data to the cloud. Additionally, it features many I/O pins for integrating with sensors and other components, and it enables analog input for sensor data.
- **Cloud and database integration:** Firebase allows real-time updates and data tracking and is utilized for real-time data. MySQL is used to store historical data. Firebase can be integrated with PHP APIs.

- **Real-time Monitoring App:** Kotlin can be used to develop a mobile application that provides analytics and notifications along with real-time data monitoring.
- **Drip irrigation system:** With the right configuration, drip irrigation for plants in public gardens is technically possible. ESP32-controlled automated drip system with solenoid valves that open and close in response to current conditions, sensor data, and water supply.

4.3.2 Social Feasibility

- **Handling Water Scarcity:** It can offer a substitute water source, lessening dependency on conventional water sources, and be particularly helpful during droughts or dry seasons.
- **Public Health and Safety:** The initiative makes sure that the collected water is safe for use by monitoring water pH.
- **Accessibility:** The system is made relevant and practical by using a mobile app for real-time monitoring.
- **Scalability and Long-Term Vision:** A wider impact could be achieved by expanding or adapting the centralized rainwater gathering and monitoring system to additional communities.
- **Assistance from Local Organizations and Authorities:** Getting the backing of local government organizations can make things more feasible. If the project fits in with the city's objectives for urban greening or water conservation, authorities may provide funding or other support.

4.3.3 Operational Feasibility

- **Usability:** Because the system automates the gathering, tracking, and distribution of rainfall, it is easy to operate. End users can easily access data.
- **Cost-effectiveness:** The project includes expenses for developing a mobile app, establishing a centralized collection system, and purchasing hardware (sensors, ESP32, valves, tanks). Nonetheless, a large number of these parts are widely accessible and reasonably priced.

- **Data Management and Real-Time Monitoring:** The mobile app can display real-time data that the ESP32 has provided to Firebase for decision-making. If the water quality drops below acceptable levels, real-time alarms can be set off.

4.3.4 Economic feasibility

- **Capital Investment Costs:** Major parts including sensors, storage tanks, and ESP32 must be purchased. The cost of creating and maintaining mobile applications also needs to be taken into account.
- **Financial Potential:** The project might be able to obtain grants or funding from educational, governmental, or non-profit organizations that support inclusive education for individuals with disabilities, making it financially feasible in the long run.
- **Fit for areas with scarce water supplies:** The system can be made freely available to the locals since it can help them.

5. SYSTEM DESIGN

5.1 ARCHITECTURE DIAGRAM

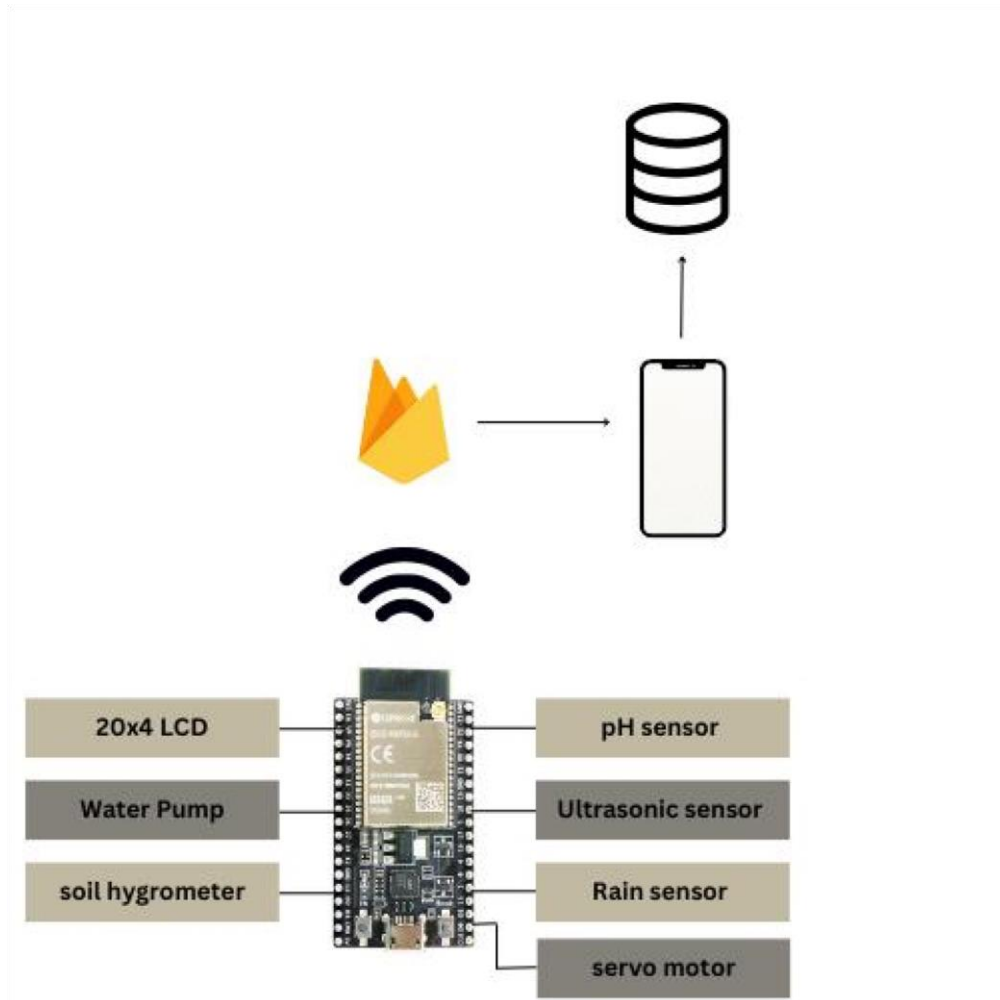


Figure 5.0.1 Architecture Diagram

The ESP32 Microcontroller is one centre point to where all the sensors are connected into. The data from the sensors are collected in and sent to the firebase real-time database through the Wi-Fi module. The data is displayed in the mobile application and then eventually sent through PHP API's to MySQL database for further storage.

5.2 CIRCUIT DIAGRAM

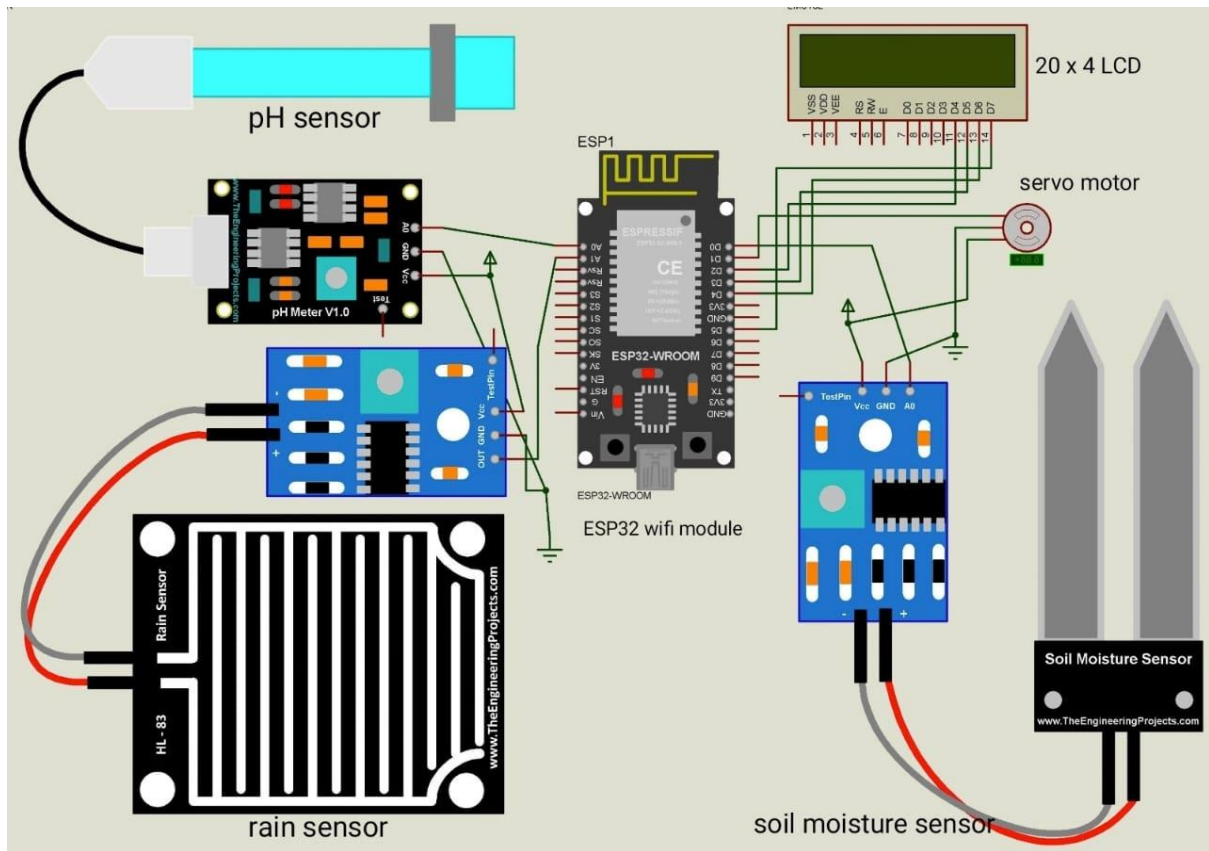


Figure 5.0.2 Circuit diagram

5.3 TABLE DESIGN

Table 1 user_details

Field name	Type	Constraints
userid	int	Primary key
first_name	varchar(30)	NOT NULL
last_name	varchar(30)	NOT NULL
email	varchar(30)	NOT NULL
phone_number	varchar(15)	NOT NULL
password	varchar(15)	NOT NULL

The following table is used to store the personal information of users along with their login credentials.

Table 2 sensor_data

Field name	Type	Constraints
id	int	Primary Key
quantity	decimal(10,2)	NOT NULL
date	date	NOT NULL
time	time	NOT NULL
type	varchar(50)	NOT NULL

Details like the quantity of rainfall based on date and time is stored here. The type refers to if the water is being flown into or out of the tank. This helps in analysing the water usage and water collection details.

5.4 DATAFLOW DIAGRAM

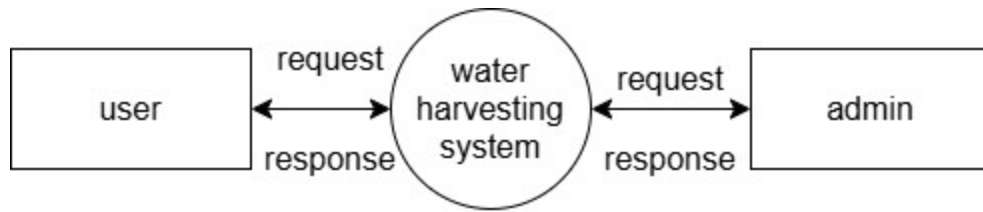


Figure 5.4.1 Context Diagram

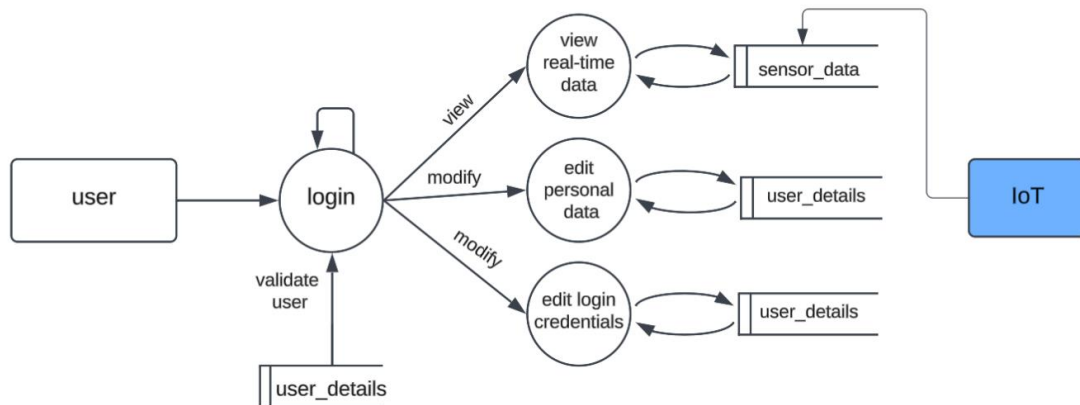


Figure 5.4.2 Level 1 User

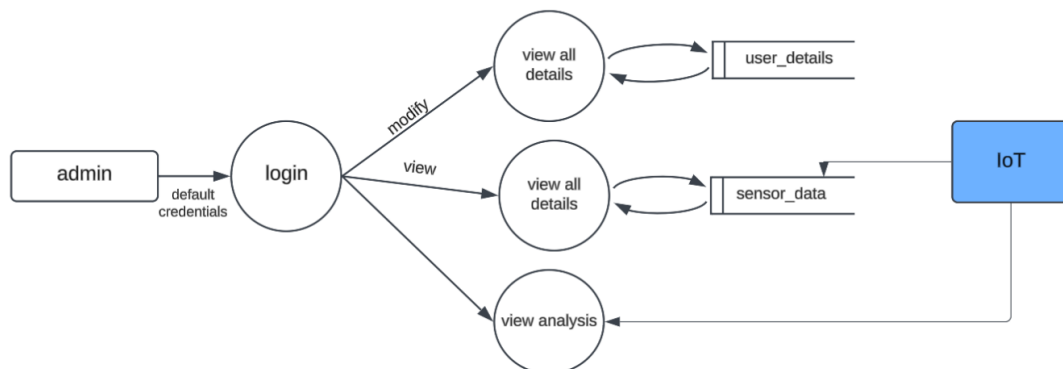


Figure 5.4.3 Level 1 admin

5.5 USE CASE DIAGRAM

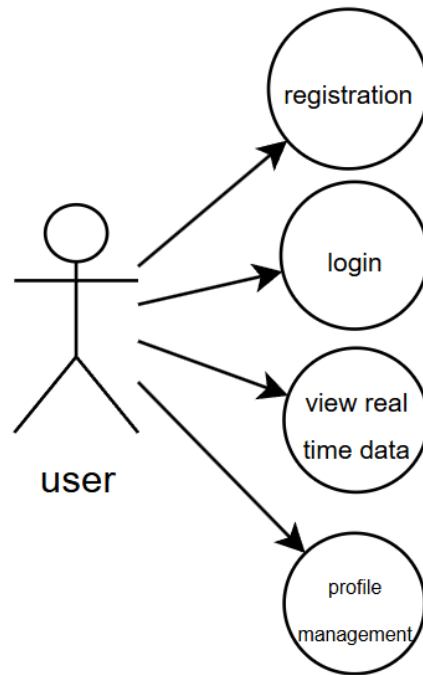


Figure 5.5.1 Use Case diagram - user

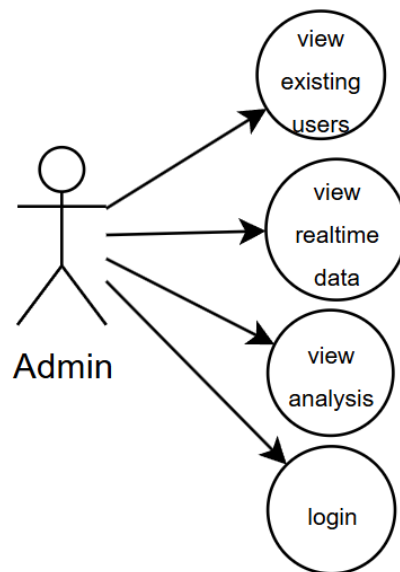


Figure 5.5.2 Use Case diagram - admin

6. MODULE DESCRIPTION

6.1 MAIN TANK SYSTEM

The tank consists of a pH sensor which is connected to the ESP32 helps in updating the water pH value to the mobile application. The real-time water pH inside the tank is analysed and continuously sent to Firebase. The data is also sent to LCD and mobile application and after that it is stored in the MySQL database for further analysis. A water pump is used to pump water into the individual households for personal use. Water level monitoring is also done inside the centralized tank. It consists of an ultrasonic sensor which updates the amount of water in the tank in real-time, this data is sent to the Firebase which is further shown via the mobile application. The real-time water level is also updated to the LCD screen set up in the tank.

6.2 AUTOMATED WATER COLLECTION MODULE

A rain sensor is connected to the lid of the tank which helps in the opening of the tank while it is raining and a servo motor that works on the basis of the value produced by the rain sensor. When the rain is detected the rain sensor produces a digital value for which the decision is made to open the tank lid. After the rain a digital value is produced for which the tank lid closes immediately.

6.3 DRIP IRRIGATION MODULE

The soil hygrometer senses the soil moisture and the data is sent which is collected by microcontroller and is sent to Firebase. Depending upon the value of the soil hygrometer, decision is made whether the drip irrigation system should be initiated or not. The drip irrigation system is connected to the rainwater harvesting tank and it helps in automatic watering of plants in the street. The soil moisture is analysed and when it goes below a particular limit, the water pump gets initiated. The drip irrigation system can be controlled by the admin via the mobile application. Admin has the authority to decide whether the system

should be controlled manually or automatically. The water pump is mainly used to pump the water for the drip irrigation system.

6.4 USER MODULE

The User module is dedicated to the residential users of the street who use the rainwater harvesting system for their individual use. The mobile application provides features to analyse real-time data and analysis report.

- a) Access to mobile application: Every information about the system such as the tank lid status, rainfall sensing, water level in the tank is presented through mobile application. The data collected via the ESP32 is sent to the Firebase and eventually to the mobile application for providing information to the user. Real-time pH, water level in the tank are provided to the users and eventually these data are sent to the MySQL for further storage and analysis.
- b) Modify personal details: Users can access and modify their personal information via the mobile application including their login credentials and personal data. This leads to ease of modification of data.
- c) Real-time data view: Users can view the real-time data produced by the IoT system with an internet connection. This provides transparency and helps user understand the current state of the system.
- d) Notifications and alerts: Notifications are an important part of the application. Notifications for when it is raining, when the tank is full/empty and whether the tank lid is open/close is informed via the application.

6.5 ADMIN MODULE

The admin serves as the higher authority of the whole system, responsible for managing the whole system and monitoring the users.

- a) Access to mobile application: Every information about the system such as the tank lid status, rainfall sensing, water level in the tank and finally the analysis is presented through mobile application. The data collected via the ESP32 is sent to the Firebase and eventually to the mobile application for providing information to the user. The admin is the single person who can access the soil moisture sensor data.
- b) Provision to view users: Admin has the capability view existing users.
- c) Data access: Admin can access all the real-time and stored data about the system.
- d) Notifications and alerts: Notifications are an important part of the application. Notifications for when it is raining, when the tank is full/empty and whether the tank lid is open/close is informed via the application.

7. SYSTEM IMPLEMENTATION

7.1 SOURCE CODE

7.1.1 Arduino code

```
#include <WiFi.h>
#include <LiquidCrystal_I2C.h>
#include <FirebaseESP32.h>
#include <ESP32Servo.h> // Use the ESP32Servo library

// WiFi credentials
#define WIFI_SSID "kiranks"
#define WIFI_PASSWORD "10341069876"

// Firebase credentials
#define FIREBASE_HOST "https://rain-water-evanious-default-rtdb.asia-southeast1.firebaseio.com/"
#define FIREBASE_AUTH "AIzaSyAluc9rezwvcdmL1MdTnp2MOZC-NqvGBhI"

// Firebase configuration and authentication
FirebaseConfig config;
FirebaseAuth auth;

// Firebase object
FirebaseData firebaseData;

// Ultrasonic sensor pins
#define TRIG_PIN 22
#define ECHO_PIN 23

// Soil hygrometer pin
#define SOIL_PIN 35

// Rain sensor pin
```

```

#define RAIN_PIN 34

// pH sensor pin
#define PH_PIN 33
#define waterpump 25
#define pump 26
// Servo motor pin
#define SERVO_PIN 15

// Initialize 20x4 LCD display
LiquidCrystal_I2C lcd(0x27, 20, 4);

// Initialize Servo object
Servo rainLidServo;

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

  // Initialize 20x4 LCD
  Wire.begin(21, 19); // Set SDA to GPIO 21 and SCL to GPIO 19
  lcd.begin(20, 4); // Initialize 20x4 LCD
  lcd.backlight();

  // Connect to Wi-Fi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi...");
  lcd.setCursor(0, 0);
  lcd.print("Connecting to Wi-Fi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to Wi-Fi.");
  lcd.clear();
}

```

```

lcd.setCursor(0, 0);
lcd.print("Wi-Fi Connected!");

// Set up Firebase config
config.host = FIREBASE_HOST;
config.signer.tokens.legacy_token = FIREBASE_AUTH;

// Initialize Firebase
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);

// Set up ultrasonic sensor pins
pinMode(TRIG_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);

// Set up other sensor pins
pinMode(SOIL_PIN, INPUT);
pinMode(RAIN_PIN, INPUT);
pinMode(PH_PIN, INPUT);
pinMode(waterpump, OUTPUT);
pinMode(pump, OUTPUT);

// Attach servo motor to pin
rainLidServo.attach(SERVO_PIN);
rainLidServo.write(0); // Initially keep the lid closed
}

void loop() {
  // Ultrasonic sensor
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

```

```

long duration = pulseIn(ECHO_PIN, HIGH);
float distance = duration * 0.034 / 2;

// Soil hygrometer
int soilMoistureValue = analogRead(SOIL_PIN);

// Rain sensor
int rainValue = analogRead(RAIN_PIN);

// pH sensor
int pHValue = (analogRead(PH_PIN)-4088);

// Display data on the LCD
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Distance: ");
lcd.print(distance);
lcd.print(" cm");

lcd.setCursor(0, 1);
lcd.print("Soil: ");
lcd.print(soilMoistureValue);

lcd.setCursor(0, 2);
lcd.print("Rain: ");
lcd.print(rainValue);

lcd.setCursor(0, 3);
lcd.print("pH: ");
lcd.print(pHValue);

// Push data to Firebase
if (Firebase.setFloat(firebaseData, "/Sensor/Distance", int(distance))) {
    Serial.println("Distance sent to Firebase successfully.");
}

```

```

    } else {
        Serial.println(firebaseData.errorReason());
    }

    if (Firebase.setInt(firebaseData, "/Sensor/Moisture", int(soilMoistureValue))) {
        Serial.println("Soil moisture sent to Firebase successfully.");
    } else {
        Serial.println(firebaseData.errorReason());
    }

    if (Firebase.setInt(firebaseData, "/Sensor/Rain", int(rainValue))) {
        Serial.println("Rain sensor value sent to Firebase successfully.");
    } else {
        Serial.println(firebaseData.errorReason());
    }

    if (Firebase.setInt(firebaseData, "/Sensor/PH", int(pHValue))) {
        Serial.println("pH sensor value sent to Firebase successfully.");
    } else {
        Serial.println(firebaseData.errorReason());
    }

    // Servo motor control
    if (rainValue < 4000) { // Example threshold for rain detection
        rainLidServo.write(90); // Open the lid
    } else {
        rainLidServo.write(0); // Close the lid
    }

    if (soilMoistureValue > 4000) {
        digitalWrite(waterpump, HIGH);
        digitalWrite(pump, HIGH);
    } else {
        digitalWrite(waterpump, LOW);
        digitalWrite(pump, LOW);
    }

```

```

    }
    delay(2000); // Delay before next iteration
}

```

7.1.2 Login page

```

<?php
require 'connection.php';
$phone = $_REQUEST['phone'];
$password = $_REQUEST['password'];
$data = array();
$post = array();
$sql = "SELECT * FROM users WHERE phone_number='$phone' AND
password='$password'";
$result = $con->query($sql);
$count = $result->num_rows;
if($count > 0) {
    $row = $result->fetch_assoc();
    $data[] = array(
        "userid" => ($row['userid'] == null ? "" : $row['userid']),
        "first_name" => ($row['first_name'] == null ? "" : $row['first_name']),
        "last_name" => ($row['last_name'] == null ? "" : $row['last_name']),
        "phone" => ($row['phone_number'] == null ? "" : $row['phone_number']),
        "email" => ($row['email'] == null ? "" : $row['email']),
        "photo" => ($row['photo'] == null ? "" : $base_path.$row['photo']) );
    $post = array(
        "status" => true,
        "message" => "Login Successful",
        "userData" => $data );
} else {
    $post = array(
        "status" => false,
        "message" => "Invalid phone number or password",

```



```

        "userData" => $data
    );
}
echo json_encode($post);
?>

```

7.1.3 Connection establishment

```

<?php
$con=new mysqli("localhost","root","Neontetra@2021#","rain_water");
if(!$con)
{
    die("sql connection error");
}
$server_path=$_SERVER['DOCUMENT_ROOT']."/Project/rain_water/upload";
$base_path="http://campus.sicsglobal.co.in/Project/rain_water/upload/";
?>

```

7.1.4 view_analysis

```

<?php
// Enable error reporting
ini_set('display_errors', 1);
error_reporting(E_ALL);

// Get the current month and year
$currentMonth = date('m');
$currentYear = date('Y');

require 'connection.php';

```

```

// SQL query to calculate the average quantity for 'rain' and 'usage' grouped by date
$sql = "SELECT
    date,
    type,
    AVG(quantity) AS average_quantity
FROM
    sensor_data
WHERE
    MONTH(date) = '$currentMonth' AND YEAR(date) = '$currentYear'
GROUP BY
    date, type
ORDER BY
    date ASC, type ASC";

$result = $con->query($sql);

if ($result->num_rows > 0) {
    $data = [];
    while ($row = $result->fetch_assoc()) {
        $data[] = [
            'date' => $row['date'],
            'type' => $row['type'],
            'average_quantity' => round($row['average_quantity'], 2)
        ];
    }
    echo json_encode(["status" => true, "data" => $data]);
} else {
    echo json_encode(["status" => false, "message" => "No data found for the current
month"]);
}
$con->close();
?>

```

7.1.5 Homefragment

```
package com.project.irhs.fragments

import android.app.AlertDialog
import android.content.Context
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ProgressBar
import android.widget.TextView
import android.widget.Toast
import androidx.fragment.app.Fragment
import com.google.firebase.database.*
import com.project.irhs.R
import com.project.irhs.databinding.FragmentHomeBinding
import kotlinx.coroutines.*
import java.net.HttpURLConnection
import java.net.URL
import java.text.SimpleDateFormat
import java.util.*

class HomeFragment : Fragment() {

    private lateinit var progressBar: ProgressBar
    private lateinit var progressText: TextView
    private val scope = CoroutineScope(Dispatchers.Main + Job())

    private lateinit var binding: FragmentHomeBinding
    private lateinit var database: DatabaseReference

    private var previousWaterLevel: Float = 0f
```

```

private var waterUsage: Float = 0f

private var rainfall: Float = 0f

override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    binding = FragmentHomeBinding.inflate(inflater, container, false)

    progressBar = binding.circularProgressBar
    progressText = binding.progressText

    progressBar.visibility = View.VISIBLE
    progressText.text = "Loading..."

    return binding.root
}

private fun initializeDatabaseListener() {
    database = FirebaseDatabase.getInstance().reference

    database.addValueEventListener(object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {
            try {
                val ph_value = snapshot.child("Sensor/PH").value
                val currentWaterLevel =
snapshot.child("Sensor/Distance").value?.toString()?.toFloatOrNull()
                val moist =
snapshot.child("Sensor/Moisture").value?.toString()?.toIntOrNull()
                val rain = snapshot.child("Sensor/Rain").value?.toString()?.toIntOrNull()

                binding.rain.text = if (rain != null && rain > 4000) "Not Raining" else
                "Raining"
            }
        }
    })
}

```

```

binding.moist.text = if (moist != null && moist > 4000) "Dry" else "Wet"

if (moist != null && moist > 4000) {
    showAlert("Water Pump Alert", "Soil is dry! The water pump has been
turned ON.")
}

binding.phvalue.text = ph_value?.toString() ?: "N/A"
binding.progressText.text = currentWaterLevel?.toString() ?: "N/A"

if (currentWaterLevel != null) {
    val waterLevelDifference = currentWaterLevel - previousWaterLevel

    val isAdmin = arguments?.getBoolean("isAdmin", false) ?: false

    if (isAdmin) { // Only allow admin to send data
        if (rain != null && rain > 4000) {
            waterUsage += waterLevelDifference
            binding.wateringValue.text = "Water Usage: $waterUsage liters"
            sendDataToPhpServer(waterUsage, "usage")
        } else if (rain != null && rain <= 4000) {
            rainfall += waterLevelDifference
            binding.wateringValue.text = "Rainfall: $rainfall mm"
            sendDataToPhpServer(rainfall, "rain")
        }
    }

    previousWaterLevel = currentWaterLevel
}

if (moist != null) {
    binding.wateringValue.text = if (moist > 4000) "Active" else "Disabled"
}

```

```

        if (rain != null) {
            binding.sunImg.setImageResource(if (rain > 4000) R.drawable.rain else
R.drawable.mazha)
            binding.lidImg.setImageResource(if (rain > 4000) R.drawable.closelid
else R.drawable.openlid)
        }

        progressBar.visibility = View.GONE

    } catch (e: Exception) {
        Log.e("FirebaseError", "Error fetching data: ${e.message}")
        Toast.makeText(requireContext(), "Error loading data",
Toast.LENGTH_SHORT).show()
    }
}

override fun onCancelled(error: DatabaseError) {
    progressBar.visibility = View.GONE
    Toast.makeText(requireActivity(), "Failed to read sensor data:
${error.message}", Toast.LENGTH_SHORT).show()
}

})
}

private fun sendDataToPhpServer(quantity: Float, type: String) {
    val currentDate = SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault()).format(Date())
    val currentTime = SimpleDateFormat("HH:mm",
Locale.getDefault()).format(Date())

    val jsonBody = """"
    {
        "quantity": $quantity,
        "type": "$type",

```

```

        "date": "$currentDate",
        "time": "$currentTime"
    }
    """.trimIndent()

    scope.launch(Dispatchers.IO) {
        try {
            val url =
                URL("http://campus.sicsglobal.co.in/Project/rain_water/api/store_sensor_data.php")
            val connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "POST"
            connection.setRequestProperty("Content-Type", "application/json")
            connection.doOutput = true

            connection.outputStream.use { outputStream ->
                outputStream.write(jsonBody.toByteArray())
            }

            val responseCode = connection.responseCode
            if (responseCode == HttpURLConnection.HTTP_OK) {
                Log.d("ServerResponse", "Data sent successfully")
            } else {
                Log.e("ServerResponse", "Failed to send data: $responseCode")
            }

            connection.disconnect()
        } catch (e: Exception) {
            Log.e("HttpError", "Error sending data to PHP server: ${e.message}")
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
    }

```

```
        scope.cancel()
    }

    private fun showAlert(title: String, message: String) {
        AlertDialog.Builder(requireContext())
            .setTitle(title)
            .setMessage(message)
            .setPositiveButton("OK") { dialog, _ -> dialog.dismiss() }
            .show()
    }
}
```


8. SYSTEM TESTING

Testing is the process of finding bugs in a program. It helps improve the quality of the software. It must be done thoroughly and with the help of specialist testers. System testing is the process of checking whether the developed system functions according to the objectives and requirements.

8.1 UNIT TESTING

Unit testing of software applications is performed during the development (coding) phase of an application. The objective of unit testing is to isolate a section of code and verify its correctness. In procedural programming, a unit may be an individual function or procedure. The goal of unit testing is to isolate each part of the program and ensure that the individual components function correctly. Unit testing is usually performed by the developer. It aims to discover errors in the individual modules of the system, whereas integration testing focuses on the decision logic, control flow, recovery procedures, throughput, capacity, and timing characteristics of the entire system.

8.2 INTEGRATION TESTING

Integration testing verifies that different modules of the system interact correctly when integrated. This is done after unit testing to ensure that all units work together as expected. Integration testing is performed to detect errors in data flow between modules. Ensure smooth communication between different system components. Validate the correctness of interfaces and interactions between units.

8.3 SYSTEM TESTING

This testing ensures that all the system components work together properly. After completing the project, system testing was performed based on control flow and correct output. This process verified that the project functioned correctly with system configurations and other dependencies to produce the desired results.

8.4 VALIDATION TESTING

Validation testing ensures that the product meets the client's needs and expectations. It verifies that the software functions correctly in its intended environment. Validation testing can be best demonstrated using the V-Model, where each phase of development is tested to ensure compliance with requirements.

8.5 BLACK BOX TESTING

Black box testing focuses on testing the system without knowledge of its internal workings. The tester evaluates the functionality of the application based on the expected inputs and outputs. Ensures that all functionalities work as per requirements. Helps detect incorrect or missing functionalities. Focuses on user experience and external behaviour.

8.6 WHITE BOX TESTING

White box testing, also known as glass-box testing, tests the internal structure, design, and implementation of the system. It involves examining the code to verify logical flows, error handling, and performance optimization. Tests the internal logic and structure of the code. Identifies hidden errors within the application's source code. Requires knowledge of programming and system architecture.

8.7 TEST CASES

8.7.1 IoT system

Table 3 test_cases_mobile_application

Test case ID	Description	Expected result	Actual result	Pass/Fail
1000	Validate ultrasonic sensor readings	Water level is correctly displayed on the	Water level displayed accurately	Pass

		LCD and mobile app		
1001	Test automatic water pumping	Water is pumped to household tanks when they are empty	Water pumped successfully when required	Pass
1002	Check drip irrigation system	System releases water to plants based on moisture levels.	Water released as per schedule	Pass
1003	Validate lid opening and closing mechanism	Tank lid closes when no rain is detected and opens when rain is detected.	Lid opened when it started raining and closed properly after rain stopped	Pass
1004	Check system response to acidic water	System gives alerts when high/low pH is detected	Alerts was produced when safety went beyond safe limits	Pass
1005	Wi-Fi connectivity test	IoT devices stay connected and transmit data correctly	Devices maintained connectivity and transmitted data properly	Pass

1006	Verify rain sensor detection	System detects rain and opens the tank	System detected rain and opened tank	Pass
1007	Verify pH sensor functionality	Sensor provided accurate pH value within expected range	Sensor provided expected result	Pass

8.7.2 Mobile Application

Table 4 test_cases_mobile application

Test Case ID	Description	Expected result	Actual result	Pass/Fail
0001	Verify user login functionality	User successfully logs in with valid credentials	User logged in successfully	Pass
0002	Test real-time water level updates	Water level updates correctly in the app	Water level updated in real-time	Pass
0003	Check rain detection alert	User receives alerts when rain is detected	User received notification when rain started	Pass
0004	Verify admin login functionality	Admin successfully logs in with default credentials	Admin logged in successfully	Pass

0005	Validate admin dashboard access	Admin can view all system data	Admin was able to access and view all system data	Pass
0006	Security test	Unauthorized users cannot access restricted features	Unauthorized users were unable to access restricted features	Pass
0007	Validate system alerts	Users receive alerts for critical issues (low water level, rain alert, water pump, etc.)	Users received alerts for critical system conditions	Pass

9. CONCLUSION

The Smart Rainwater Harvesting System with Automated Irrigation project aims to improve the sustainable use of rainwater with the help of an automated system. Through this project, our team aims to develop a solution that integrates an IoT system with a mobile application to produce an all-around automated system for sustainable water collection and management. The system includes a centralized water tank which collects the rooftop water and store it then pump to individual households for their own use. The centralized tank also includes a plant watering system which waters the street plants automatically depending upon the soil moisture. The mobile application provides all the details about the tank system such as the rain status, tank lid status, water level, water pH etc. The mobile application helps in getting alerts when the water level goes beyond a particular limit and letting the users know that water may stop at any moment hence switch to the corporation water lines. In conclusion, the project design aims to develop an automated system that can collect, store, and manage rainwater in cities and areas with less rainfall along with promoting urban greenery.

10. FUTURE ENHANCEMENTS

Looking toward the future, several potential enhancements could further elevate the capabilities and usability of the system. Integration of weather forecasting by adding weather APIs can significantly improve its efficiency, sustainability, and water management capabilities and help in scheduling maintenance. It allows users to empty the tank for maximum water capture and for planning usage. Additionally, we can include some more sensors into the system that can check the water quality parameters such as turbidity sensors for measuring the cloudiness or suspected particles, TDS sensors for measuring the concentration of dissolved substances, dissolved oxygen sensors which are important for aerobic bacteria that help in natural water cleaning. We can add a maintenance alert in the mobile app by including a calendar module which can make it easy for the user not to remember about the future maintenance dates and it helps to keep the system maintenance systematic and efficient. Adding a billing system to rainwater that is distributed from the centralized tank ensures fair usage and allows the system to be financially self-sustaining. It enables the user to only pay based on their water consumption and provides financial benefits to the government or authorities responsible for the management of the system. This future enhancement holds the potential to transform smart rainwater harvesting systems with automated drip irrigation systems into more sophisticated and versatile tools for ensuring water conservation.

11. REFERENCES

- [1] V S P Chandrika Kota, Chinna Rao Annepu, Kalyan Dusarlapudi, Sreelatha Edara, "Smart Approach Of Harvesting Rainwater And Monitoring Using Iot", International Research Journal Of Engineering And Technology (IRJET), Vol 10 Issue 8 Pg.662-667, 2020
- [2] Mohammed Golam Sarwar Bhuyan, Must. Asma Yasmi, "Iot Based Smart Rainwater Harvesting And Classifying System Journal Of Emerging Technologies And Innovative Research", Vol 9 Issue 10 Pg.A363-A370, 2022
- [3] Prof. Priyanka Golsalves, Mr. Vivek Modi, Mr. Amaan Jambura, Komal Ayare. "Smart Rainwater Harvesting System", International Research Journal Of Engineering And Technologies, Vol 10 Pg.662-667, 2023
- [4] Shweta Karad, Maryam Merchant, Ashwini Kardili, Vijayata Misra, "Iot Based Real Time Water Monitoring System For Smart City", International Journal Of Innovative Science And Research Technology, Pg.246-251, 2018
- [5] Praveen Talari, Anirudh M, Sairam M, K Varun, Jaba Kumar. "An Integrative Iot Approach For Smart Rainwater Harvesting", International Journal For Innovative Technology And Exploring Engineering, Vol 11, Pg.1-5, 2021
- [6] D.Srinath Reddy, B. Sai Nishanth, C H. Sai Teja, D.Varun Reddy, "Iot Based Automatic Rain Water Harvesting And Irrigation System", International Journal Of Advanced Research In Computer Science, Vol 11, Pg.16-2, 2020
- [7] Md.Mahbubur Rahman, Chinmay Bepery, Mohammad Jamal Hossain, Zahidhassan, "Internet Of Things (Iot) Based Water Quality Monitoring System", International Journal Of Multi Disciplinary And Current Educational Research, Vol 2, Pg.168-, 2020

12. APPENDIX

12.1 UI SCREENSHOTS

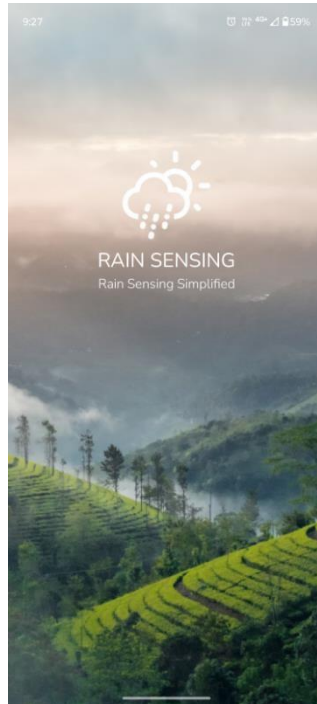


Figure 12.1.1 Splash Page

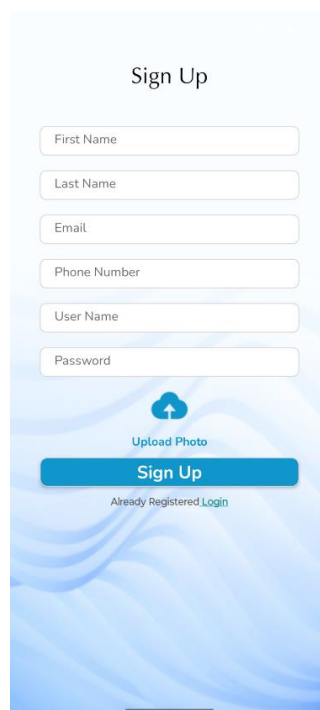
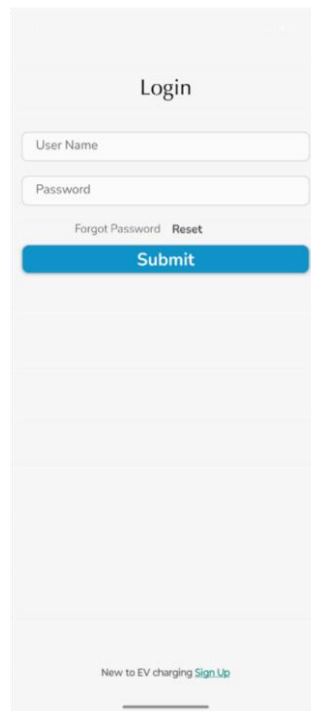
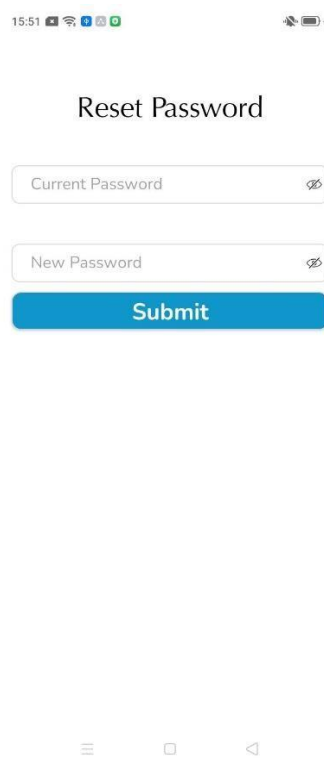
The 'Sign Up' page of the app. It has a light blue background with a subtle wave pattern. The title 'Sign Up' is centered at the top. Below it are seven input fields: 'First Name', 'Last Name', 'Email', 'Phone Number', 'User Name', and 'Password'. Each field is a simple white rectangle with rounded corners. Below the 'Password' field is an 'Upload Photo' section, which includes a blue circular icon with a white upward arrow and the text 'Upload Photo'. At the bottom of the form is a large blue button with the text 'Sign Up' in white. Below the button, there is a link that says 'Already Registered.Login'.

Figure 12.1.2 Sign up page



A mobile app login screen with a light gray background. At the top, the word "Login" is centered in a dark gray font. Below it are two white input fields with gray borders, labeled "User Name" and "Password". Under the password field, there are two links: "Forgot Password" and "Reset". A prominent blue button with the word "Submit" in white is centered below the links. At the bottom of the screen, a small line of text reads "New to EV charging [Sign Up](#)".

Figure 12.1.3 User Login Page



A mobile app reset password screen. The status bar at the top shows the time "15:51" and various system icons. The title "Reset Password" is centered in a dark gray font. Below the title are two white input fields with gray borders and toggle icons on the right, labeled "Current Password" and "New Password". A blue "Submit" button is centered below the fields. At the bottom, there are three small, faint icons: a hamburger menu, a square, and a left-pointing arrow.

Figure 12.1.4 Reset Password Page

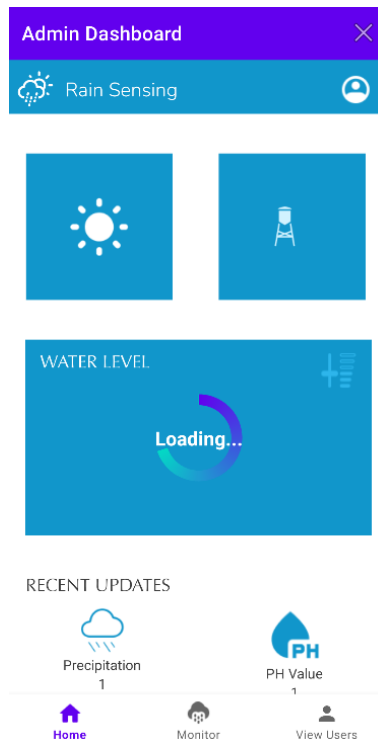


Figure 12.1.5 Admin Home Page

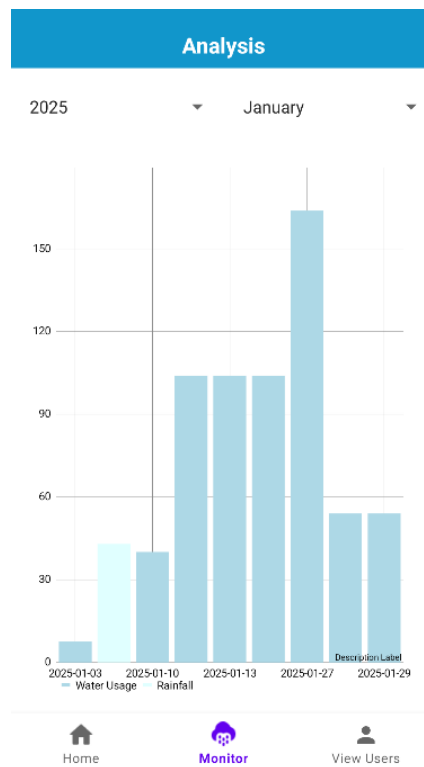


Figure 12.1.6 Analysis page



Figure 12.1.7 View User page



Figure 12.1.8 LCD Monitor

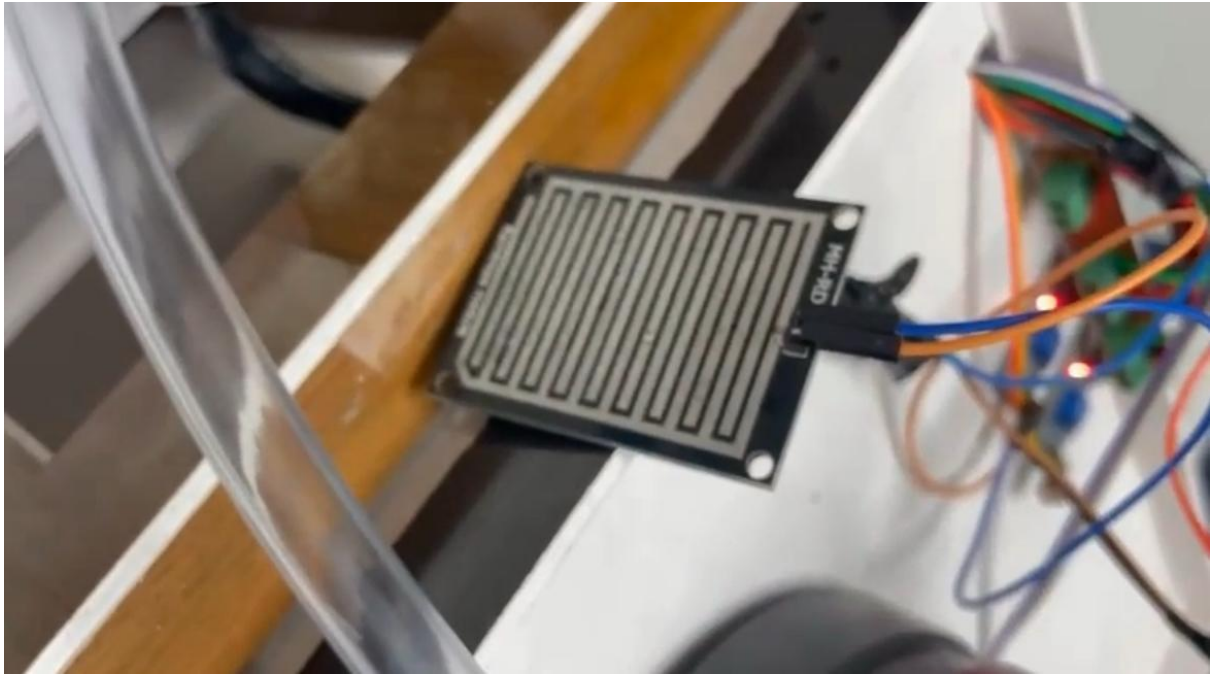


Figure 12.1.9 Rain sensor

13. USER MANUAL

13.1 INTRODUCTION

This system is designed to efficiently collect and manage rainwater from rooftops, analyse its quality, and distribute it for non-potable uses. It also features an automated irrigation system for street plants. The system is IoT-based and provides real-time updates via a mobile app.

13.2 GETTING STARTED

13.2.1 System Components

- **Rain Sensor** – Detects rainfall and signals the tank lid to open.
- **pH Sensor** – Analyses water acidity.
- **Ultrasonic Sensor** – Monitors water levels in the tank.
- **Centralized Tank with Motorized Lid** – Stores rainwater.
- **Household Tanks** – Individual tanks for non-potable water storage.
- **Water Pump** – Transfers water from the centralized tank to household tanks.
- **Drip Irrigation System** – Automates watering of street plants.
- **LCD Display** – Shows real-time water level and status.
- **Mobile Application** – Provides user and admin access to system data.

13.3 TURNING ON/OFF THE DEVICES

13.3.1 Turning On

1. Ensure all components are properly installed and connected.
2. Power on the central control unit and verify sensor connectivity.
3. Check the LCD screen and mobile app to confirm system readiness.

13.3.2 Turning Off

1. Shut off the main power supply to stop the automated working of the system.
2. Shut off the main power supply before performing maintenance works.

3. Ensure the water pump and sensors are inactive before performing any system modifications.

13.4 INSTALLATION GUIDE

Step 1: Setting Up the Centralized Tank

1. Select a location with good rooftop runoff collection potential.
2. Install the centralized tank and connect it to rooftop pipelines.
3. Ensure the motorized lid is properly attached and functional.

Step 2: Installing Sensors

1. Attach the **rain sensor** to an open rooftop area.
2. Place the **pH sensor** inside the centralized tank.
3. Mount the **ultrasonic sensor** at the top of the tank for level monitoring.

Step 3: Connecting the Water Pump

1. Install the water pump near the centralized tank.
2. Connect the pump output to the distribution pipelines leading to household tanks.
3. Ensure proper power supply and test pump functionality.

Step 4: Setting Up the Drip Irrigation System

1. Lay out the drip irrigation network along the street garden.
2. Connect it to the water pump.

Step 5: Mobile App Configuration

1. Download and install the mobile app from the provided source.
2. Create an account and log in.
3. Connect the system to Wi-Fi and sync it with the mobile app.

13.5 TROUBLESHOOTING

- **Lid Not Opening:** Check the rain sensor connection and system power.

- **No Water Flow to Household Tanks:** Ensure the pump is operational and pipelines are not clogged.
- **Incorrect Water Level Readings:** Clean the ultrasonic sensor and check wiring.

13.6 SAFETY AND MAINTENANCE

Regular Maintenance

- Clean **rain sensor** periodically to ensure accurate detection.
- Check **pH sensor** calibration every three months.
- Inspect **water pump** for proper functioning.
- Flush out any debris from the **drip irrigation system**.

Safety Guidelines

- Keep all electrical components away from direct exposure to rain.
- Ensure regular maintenance to prevent leaks and system failures.
- Follow mobile app notifications for system alerts and required actions.