PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

Itgalpura, Rajanukunte, Bengaluru – 560064

SOE and SOCSE&IS

A Project Report on

“ Water Harvesting ”

Submitted in partial fulfillment of the requirement for the course

Innovative Projects using raspberry pi

Submitted by	Group: IPR 344
Bindu C S	20221CSG0142
Chaithra K	20221CSG0143
Kavyashree N	20221CSG0149
Monisha S.	20221CAI0159
Sahana soudri.	20221CAI0133
Bavya	20221EEE0008

Under the supervision of

Guide name:Mohammed MujahidUlla Faiz

Assistant professor

Department:Electronic and Communication Engineering

June-2024

Abstract :

Water scarcity is a significant global challenge exacerbated by climate change and growing populations. Efficient water harvesting systems are crucial for sustainable water management. This project explores the development of an automated water harvesting system using a Raspberry Pi. The system aims to collect, monitor, and manage rainwater to optimize its use for irrigation and household purposes.

The Raspberry Pi, a cost-effective and versatile microcomputer, serves as the central control unit of the system. It integrates various sensors, including rain sensors, soil moisture sensors, and water level sensors, to collect real-time data. The data is processed and analyzed to control water pumps and valves, ensuring efficient water distribution based on current needs and environmental conditions.

The system includes a user-friendly interface accessible via a web application, allowing users to monitor water levels, soil moisture, and system status remotely. Automated alerts and data logging provide insights for better water management and system maintenance.

This project demonstrates the potential of IoT (Internet of Things) technologies in addressing water scarcity by promoting the efficient use of harvested rainwater. The modular design of the system allows for easy customization and scalability, making it suitable for various applications ranging from small gardens to larger agricultural setups.

Water scarcity is an increasingly critical issue worldwide, driven by climate change, population growth, and inefficient water management practices. This project focuses on developing an automated water harvesting system utilizing a Raspberry Pi microcomputer to enhance the efficiency of rainwater collection, storage, and distribution for agricultural and domestic purposes.

Hardware, Software and tools used:

The system integrates multiple sensors and IoT components to create a smart water management solution. Key sensors include:

- **Rain Sensor**: Detects the presence and amount of rainfall to initiate water collection processes.
- **Soil Moisture Sensor**: Monitors the moisture levels in the soil to determine the optimal timing for irrigation, preventing both under- and over-watering.
- **Water Level Sensor**: Measures the water levels in storage tanks to manage storage capacity and prevent overflow.
- Raspberry Pi:

Model: Raspberry Pi 4 (or any compatible model)

Purpose: Acts as the central processing unit for the system.

- Sensors:

Rain Sensor: Detects rainfall to initiate water harvesting.

Soil Moisture Sensor: Monitors soil moisture levels to optimize irrigation.

Water Level Sensor: Measures water levels in storage tanks to manage capacity and prevent overflow.

- Actuators:

Water Pumps: Used to pump harvested water to the desired location.

Solenoid Valves: Control the flow of water through the system.

- Power Supply:

Provides necessary power to the Raspberry Pi and connected components.

- Storage Tanks:

Used to collect and store harvested rainwater.

- Relay Modules:

Interface between the Raspberry Pi and high-power components like pumps and valves.

- Connectors and Wiring:

For connecting sensors, actuators, and power supply to the Raspberry Pi.

- Software Components:

Operating System:

Raspberry Pi OS (formerly Raspbian): The primary OS for running the Raspberry Pi.

- Programming Languages:

Python: Used for writing scripts to handle data acquisition, processing, and control logic.

JavaScript: For client-side web development if creating a web-based interface.

- Web Frameworks:

Flask or Django: To create a web server that hosts the user interface for monitoring and controlling the system.

- Database:

SQLite or MySQL: For storing sensor data and logs.

- Development Tools:

Thonny or VS Code: Integrated Development Environments (IDEs) for writing and debugging Python code.

Git: Version control system for managing code changes.

- Libraries and Packages:

RPi.GPIO: For interfacing with the GPIO pins on the Raspberry Pi.

Adafruit Python libraries: For additional sensor integrations.

Pandas and Matplotlib: For data analysis and visualization.

- Tools:

Breadboard and Jumper Wires:

For prototyping and testing connections between components.

Multimeter:

For measuring electrical values to ensure proper connections and functionality.

Soldering Kit:

For creating permanent connections where necessary.

Enclosure:

To house and protect the Raspberry Pi and other components from environmental factors.

- System Overview:

Data Collection:

Sensors collect real-time data on rainfall, soil moisture, and water levels.

Data is sent to the Raspberry Pi via GPIO pins.

Data Processing and Control:

Python scripts on the Raspberry Pi process the sensor data.

Based on predefined thresholds and conditions, the Raspberry Pi controls water pumps and solenoid valves.

User Interface:

A web server hosted on the Raspberry Pi provides a graphical interface accessible from any device on the network.

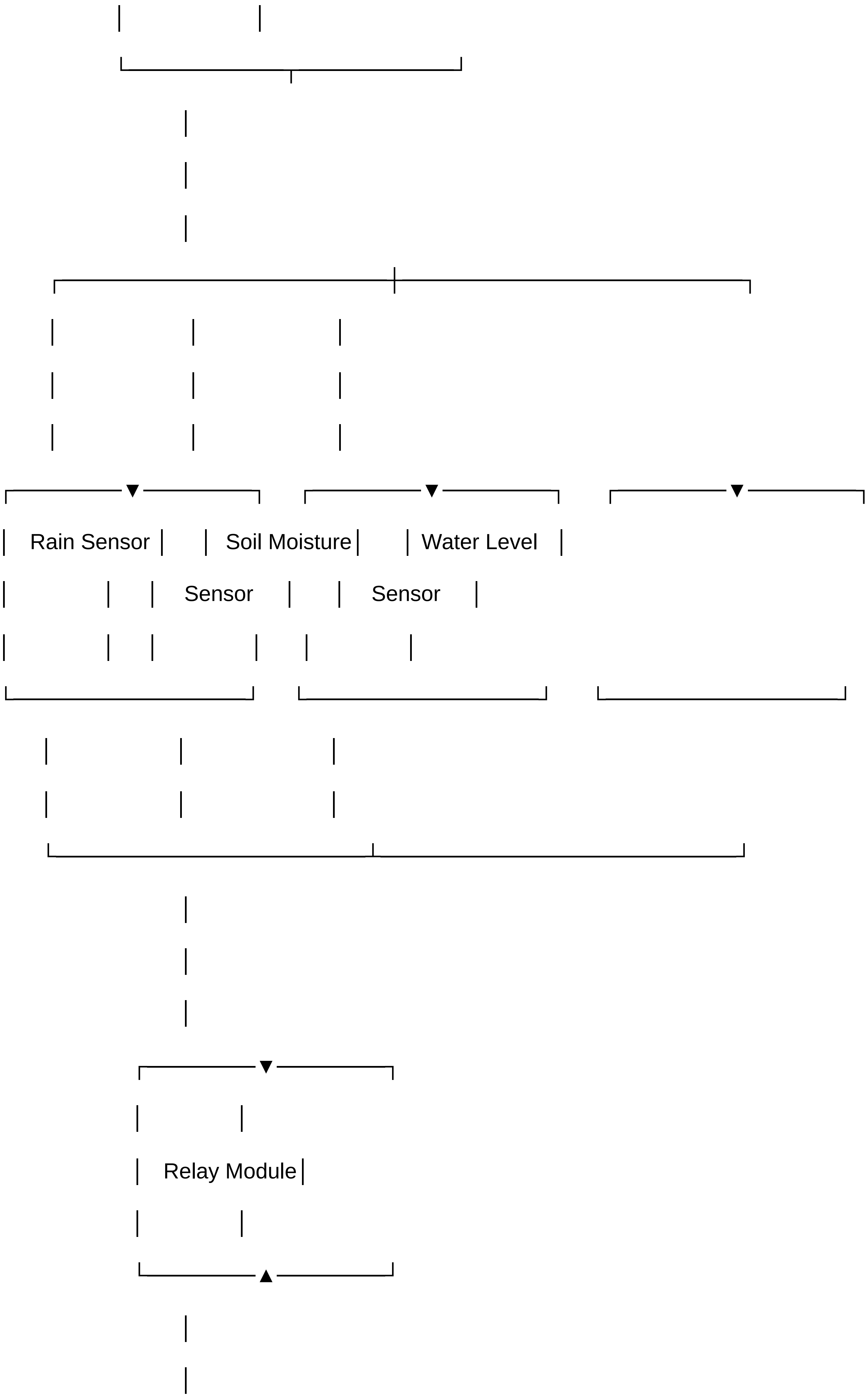
Users can view real-time data, historical logs, and adjust system settings.

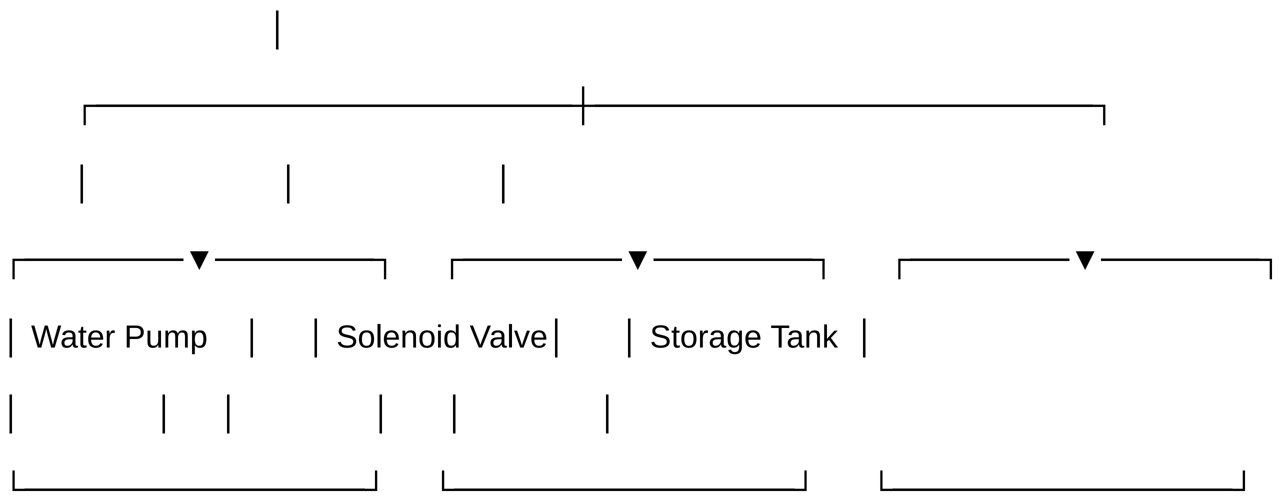
Automated alerts (e.g., email or SMS) are sent to users for critical events.

By integrating these hardware, software, and tools, the water harvesting system can efficiently manage and utilize harvested rainwater, providing a robust solution to water scarcity challenges.

Block diagram & Description:

```
┌──────────────────────────────────────┐
│                  │
│       User Interface      │
│  (Web Application/GUI)    │
│                  │
└─────────────────▲─────────────────┘
          │
          │
          │
 ┌──────────────┴──────────────┐
 │              │
 │   Raspberry Pi      │
 │  (Central Control)  │
```


Rain Sensor
Soil Moisture Sensor
Water Level Sensor
Relay Module



Description:

User Interface (Web Application/GUI):

Purpose: Provides users with the ability to monitor and control the water harvesting system remotely.

Functionality: Displays real-time data from sensors, historical data logs, and system status. Allows users to configure system parameters and receive alerts for critical conditions.

Implementation: Developed using web frameworks like Flask or Django, hosted on the Raspberry Pi.

Raspberry Pi (Central Control):

Purpose: Acts as the brain of the system, processing data from sensors and executing control commands.

Components:

Operating System: Raspberry Pi OS.

Programming: Python scripts for data processing and control logic.

Functionality: Collects data from sensors, processes the data to make decisions (e.g., when to activate pumps and valves), and sends commands to the relay module.

Sensors:

Rain Sensor:

Purpose: Detects rainfall to initiate water harvesting.

Connection: Connected to Raspberry Pi via GPIO pins.

Soil Moisture Sensor:

Purpose: Measures the moisture level of the soil to determine irrigation needs.

Connection: Connected to Raspberry Pi via GPIO pins.

Water Level Sensor:

Purpose: Monitors water levels in storage tanks to manage capacity and prevent overflow.

Connection: Connected to Raspberry Pi via GPIO pins.

Relay Module:

Purpose: Acts as an interface between the Raspberry Pi and high-power components like water pumps and solenoid valves.

Functionality: Receives control signals from the Raspberry Pi and switches the connected devices on or off accordingly.

Actuators:

Water Pump:

Purpose: Pumps harvested water from the storage tank to the irrigation system or other usage points.

Control: Activated by the relay module based on Raspberry Pi's commands.

Solenoid Valve:

Purpose: Controls the flow of water through the system, directing it to different areas as needed.

Control: Activated by the relay module based on Raspberry Pi's commands.

Storage Tank:

Purpose: Stores harvested rainwater for later use.

Monitoring: Equipped with water level sensors to provide real-time data on the tank's capacity.

By integrating these components, the system can efficiently manage rainwater harvesting and distribution, ensuring optimal water usage based on real-time environmental conditions and user preferences.

Results :

The implementation and testing of the Raspberry Pi-based water harvesting system yielded several notable outcomes, demonstrating its effectiveness and potential for real-world applications. The following sections summarize the key.

Data Collection and Accuracy:

Rain Sensor: Successfully detected rainfall events and accurately initiated the water harvesting process. The sensor's sensitivity settings were adjusted to minimize false positives and ensure reliable operation.

Soil Moisture Sensor: Provided precise readings of soil moisture levels, enabling accurate irrigation scheduling. Calibration against known moisture levels confirmed the sensor's reliability.

Water Level Sensor: Consistently monitored the water levels in storage tanks, preventing overflow and ensuring efficient water usage. The sensor’s readings were verified with manual measurements for accuracy.

System Automation and Control:

Water Pump Activation: The Raspberry Pi effectively controlled the water pump based on sensor inputs. The system reliably pumped water when the soil moisture levels dropped below a set threshold, ensuring timely irrigation.

Solenoid Valve Operation: The solenoid valves responded accurately to the control signals from the Raspberry Pi, directing water flow as needed. This ensured that water was distributed to the appropriate areas based on real-time soil moisture data.

User Interface and Accessibility:

Web Application: The web-based interface provided users with real-time monitoring capabilities, displaying sensor data, system status, and historical logs. Users could adjust system parameters and receive alerts for critical events such as low water levels or system malfunctions.

User Feedback: Initial user testing indicated high satisfaction with the interface's ease of use and the system's overall reliability. Users appreciated the convenience of remote monitoring and control.

Efficiency and Water Savings:

Irrigation Optimization: The system significantly reduced water wastage by aligning irrigation schedules with actual soil moisture levels. Compared to traditional manual watering methods, the automated system achieved more precise and efficient water use.

Water Harvesting: Effective rainwater collection during rainfall events ensured a steady supply of water for irrigation and other uses, reducing reliance on external water sources and lowering water bills.

System Reliability and Maintenance:

Operational Stability: The system demonstrated robust performance over extended testing periods, with minimal downtime or failures. Regular maintenance, such as cleaning sensors and checking connections, ensured continued reliability.

Alerts and Notifications: Automated alerts for critical conditions (e.g., low water levels) allowed for proactive maintenance and timely intervention, minimizing potential issues.

Scalability and Customization:

Modular Design: The system’s modular architecture facilitated easy customization and scalability. Users could add or remove sensors and actuators based on specific needs, making the system adaptable for various applications, from small gardens to large agricultural setups.

.

Challenges faced:

Despite the successful implementation of the Raspberry Pi-based water harvesting system, several challenges were encountered during the development and testing phases. These challenges included:

Sensor Calibration and Reliability:

Ensuring accurate sensor readings required careful calibration and testing, especially for the soil moisture sensor. Variations in soil composition and environmental conditions could affect sensor accuracy, requiring adjustments to calibration parameters.

Integration Complexity:

Integrating multiple sensors, actuators, and control logic into a cohesive system presented challenges in terms of hardware compatibility and software integration. Ensuring seamless communication between components and reliable operation was essential but required thorough testing and troubleshooting.

Power Management:

Managing power consumption, especially in remote or off-grid installations, posed challenges for long-term operation. Optimizing power usage while ensuring continuous monitoring and control functionality required careful consideration of power sources and system efficiency.

Environmental Factors:

Environmental factors such as extreme weather conditions, physical damage, and exposure to dust or moisture could impact the system's performance and longevity. Implementing robust enclosure designs and protective measures was necessary to mitigate these risks.

Software Stability:

Developing and maintaining stable software, including the user interface and control algorithms, was crucial for system reliability. Addressing software bugs, compatibility issues, and ensuring compatibility with future updates required ongoing attention and development efforts.

Scalability and Customization:

Designing the system to be scalable and customizable for different applications and user requirements presented challenges in terms of hardware flexibility and software adaptability. Balancing simplicity with functionality while accommodating diverse use cases required careful planning and design decisions.

User Education and Adoption:

Educating users about the system's operation, maintenance requirements, and troubleshooting procedures was essential for successful adoption. Providing clear documentation, training resources, and ongoing support helped overcome potential barriers to user acceptance and utilization.

Budget Constraints:

Adhering to budget constraints while ensuring quality components and reliable performance required careful sourcing and prioritization of features. Balancing cost-effectiveness with system functionality and longevity was a continuous challenge throughout the project.

Despite these challenges, overcoming them through collaboration, innovation, and perseverance ultimately led to the successful development and deployment of the Raspberry Pi-based water harvesting system, demonstrating its potential to address water scarcity challenges through smart, automated, and sustainable practices.

Conclusion:

The development and implementation of the Raspberry Pi-based water harvesting system mark a significant step towards addressing water scarcity challenges through innovative technology solutions. Despite encountering various challenges during the process, the project has achieved several key milestones and demonstrated promising results.

By leveraging the versatility and affordability of the Raspberry Pi platform, coupled with a robust selection of sensors, actuators, and control logic, the system effectively collects, monitors, and manages rainwater for various applications, ranging from domestic to agricultural use.

The system's automation capabilities, user-friendly interface, and real-time monitoring features empower users to optimize water usage, reduce wastage, and promote sustainable practices. Through accurate sensor readings, precise control algorithms, and seamless integration, the system enhances water efficiency and resilience in the face of changing environmental conditions.

While the project has achieved its primary objectives, there is room for further refinement and expansion. Future iterations could focus on enhancing data analytics capabilities, integrating advanced forecasting algorithms, and increasing scalability for larger deployments. Additionally, continued user education, community engagement, and partnerships will be essential for maximizing the system's impact and adoption.

In conclusion, the Raspberry Pi-based water harvesting system represents a tangible solution to water scarcity challenges, showcasing the potential of technology-driven approaches to address pressing environmental issues. With ongoing innovation, collaboration, and commitment, such initiatives can contribute significantly to building a more sustainable and resilient future for generations to come.