CERTIFICATE

This is to certify that this project entitled “IoT-Based Smart Water Tank Monitoring System” submitted in partial fulfillment of the degree of Bachelor of Computer Applications done by Mr. Nitesh Roll No. 09524402022 and Mr. Yash Jha Roll No. 35524402022 is an is an authentic work carried out by him at IINTM, New Delhi under my guidance. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief...

Signature of the Student
Nitesh kumar
Yash Jha

Signature of the Guide
Mrs. Priya Tripti

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to everyone who supported me during the development of the **IoT-Based Smart Water Tank Monitoring System** project.

I sincerely thank my mentor/supervisor, Dr. Priya Tripti, for their guidance, feedback, and encouragement throughout this journey. Their expertise played a pivotal role in the success of this project. I am also grateful to **IINTM** for providing the necessary resources and an environment conducive to learning and innovation.

I extend my appreciation to my peers and collaborators for their valuable insights and teamwork, as well as to my family and friends for their unwavering support and motivation. Finally, I am thankful for the online resources and documentation that helped me navigate challenges along the way.

This project has been a remarkable learning experience, and I am deeply appreciative of everyone's contributions and encouragement.

Nitesh

Yash Jha

Synopsis

Project Title: IoT-Based in Smart Water Tank Monitoring System

Abstract

Water management is a crucial issue in both residential and industrial sectors. Traditional water tank monitoring methods are inefficient, leading to water wastage, overflow, and inadequate supply. To address these challenges, this project presents an IoT-Based Smart Water Tank Monitoring System that provides real-time monitoring and automated control of water levels.

The system consists of water level sensors, an IoT-enabled microcontroller, and a cloud-based dashboard or mobile application for remote access. The sensors detect the water level in the tank and send data to the microcontroller, which processes the information and transmits it to the cloud. Users can monitor water levels in real-time, receive alerts for low or overflow conditions, and automate pump operations to optimize water usage.

This smart system enhances water conservation efforts, reduces human intervention, and ensures a continuous water supply with minimal wastage. The implementation of IoT technology makes it cost-effective, scalable, and suitable for various applications, including households, agriculture, and industries.

Introduction

Water management is a critical issue in both residential and industrial areas. Traditional water tank monitoring methods often lead to water wastage, overflow, and inefficient utilization of resources. To address these challenges, this project proposes an IoT-based Smart Water Tank Monitoring System, which provides real-time water level monitoring, automated control, and remote access via the Internet of Things (IoT).

The system integrates water level sensors, microcontrollers (such as Arduino or ESP8266/ESP32), and a cloud-based dashboard to provide users with accurate water level data. Users can receive alerts and control the water pump remotely through a mobile application or web interface. This technology enhances water conservation efforts, prevents wastage, and ensures efficient water distribution.

Objectives:

1. Real-Time Monitoring: Continuously track water levels in the tank using IoT sensors.
2. Automated Alerts: Notify users via mobile apps or SMS when water levels are critically low or high.
3. Leakage Detection: Identify leaks or abnormal water usage to prevent wastage.
4. Smart Control: Automate water pump operations based on predefined water level thresholds.

5. Data Logging & Analytics: Store and analyze water usage patterns for better resource management.
6. Remote Access: Allow users to monitor and control the system from anywhere via an online dashboard.
7. Energy Efficiency: Optimize water usage and reduce unnecessary pump operations to save electricity.

Scope:

1. Real-Time Water Level Monitoring:
    - Use of ultrasonic or float sensors to measure water levels.
    - Data transmission via Wi-Fi or GSM modules to a cloud platform.
2. Automated Alerts & Notifications:
    - Alerts for low water levels, overflow, and leaks.
    - Notifications sent via mobile app, SMS, or email.
3. Remote Access & Control:
    - Mobile app/web dashboard for monitoring water levels.
    - Option to integrate with smart valves for automated refilling.
4. Data Logging & Analysis:
    - Historical data storage for tracking water consumption trends.
    - Predictive analysis for efficient water management.
5. Energy-Efficient & Cost-Effective:
    - Low-power IoT components for minimal energy consumption.
    - Affordable setup with easy scalability for multiple tanks.

System Components:

1. Microcontroller (ESP8266 / Arduino) – Acts as the brain of the system, collecting sensor data and transmitting it to the cloud.
2. Water Level Sensors (Ultrasonic / Float / Capacitive Sensors) – Measures the water level in the tank.
3. Flow Sensor – Monitors water inflow and outflow.
4. Temperature Sensor – Tracks water temperature if needed.

5. Water Quality Sensor (Optional - TDS Sensor) – Measures water purity and contamination levels.
6. LCD Screen – Provides remote monitoring and alerts via notifications.
7. Relay Module (For Motor Control) – Automatically turns the water pump ON/OFF based on the water level.
8. Power Supply (Battery / Adapter / Solar Panel) – Provides power to the system.

Methodology:

The methodology for this project involves the following key steps:

1. Problem Identification:
    - Analyse the common issues faced in traditional water tank management, such as overflow, dry tanks, and water wastage.
    - Identify the need for a smart, automated solution to monitor and control water levels in real time.
2. Hardware Selection & Setup:
    - Microcontroller: Use of ESP8266/ESP32 or Arduino as the main control unit.
    - Sensors:
        - Ultrasonic Sensor (HC-SR04) for water level measurement.
        - Water Flow Sensor to track water consumption.
        - Temperature & pH Sensor (optional) to monitor water quality.
    - Actuators:
        - Relay module to control the water pump based on sensor inputs.

Use Cases:

1. Residential Buildings
    - Alerts users when the water level is low.
    - Automatically turns the pump on/off based on predefined thresholds.
    - Prevents overflow and unnecessary electricity usage.
2. Commercial Buildings & Apartments
    - Helps facility managers monitor and control water usage.
    - Prevents overflows and ensures an uninterrupted water supply.

- Sends notifications to maintenance teams for quick actions.

3. Industrial Use
    - Tracks large-scale water usage and prevents wastage.
    - Detects leaks and abnormalities in water flow.
    - Integrates with industrial automation systems for efficient water management.
4. Agricultural Sector
    - Monitors water levels in irrigation tanks.
    - Automates water distribution to crops based on moisture levels.
    - Reduces water wastage and optimizes resource usage.
5. Smart Cities & Municipal Water Supply
    - Helps in efficient water distribution and management.
    - Provides real-time data for urban planning and water conservation efforts.
    - Reduces manual inspections with automated reporting.

Conclusion

The IoT-Based Smart Water Tank Monitoring System provides an innovative and efficient solution for real-time water management. By integrating IoT sensors, cloud connectivity, and automated control mechanisms, the system optimizes water usage, prevents wastage, and ensures a continuous water supply. This technology is scalable and adaptable for various applications, including residential, commercial, industrial, agricultural, and municipal sectors. Implementing this system can significantly contribute to water conservation efforts and sustainable resource management.

Index

Chapter 1

Smart Water Tank Monitoring System

Introduction

Water is one of the most essential resources for daily life, and its efficient management is crucial to prevent wastage. With increasing demand for water in residential, commercial, and agricultural sectors, ensuring its proper utilization has become a priority. However, traditional water tank systems rely on manual monitoring, which can lead to various challenges such as overflow, shortages, and inefficient usage. These conventional methods often require human intervention, making them time-consuming, unreliable, and prone to human error. Such inefficiencies result in excessive water wastage, increased electricity consumption, and inconvenience to users.

To overcome these challenges, smart water management solutions are gaining popularity. With advancements in the Internet of Things (IoT), automation has made it possible to efficiently monitor and control water usage remotely. IoT-based water monitoring systems provide real-time data on water levels and automate the operation of water pumps, ensuring effective water management.

The Smart Water Tank Monitoring System is an IoT-based project designed to automate water level monitoring and optimize water usage. It uses sensors, microcontrollers, and cloud-based platforms to detect the water level inside a tank and provide users with real-time updates and alerts. The system allows users to remotely monitor and manage their water supply through a mobile application or web dashboard.

This system not only prevents water wastage but also ensures that there is an adequate supply of water at all times. By automating the pump operation, the system eliminates the need for manual intervention, reducing human effort and ensuring a cost-effective, efficient, and sustainable approach to water management.

Key Features of the System

The Smart Water Tank Monitoring System offers several key features, including:

- **Real-time Monitoring** – Displays live water levels via a web or mobile application.
- **Automated Pump Control** – Turns the water pump on or off based on predefined water levels.
- **Alert Notifications** – Sends alerts via SMS, email, or app notifications when water levels are too low or too high.
- **Cloud Integration** – Stores water usage data for long-term analysis and optimization.
- **Energy Efficiency** – Reduces unnecessary pump operation, saving electricity and reducing costs.
- **User-friendly Interface** – Provides a simple and accessible dashboard for monitoring and controlling water levels remotely.

1. Data Collection

Data collection is a crucial step in developing an IoT-based Smart Water Tank Monitoring System, as it ensures accurate measurement, monitoring, and control of water levels. The collected data helps in analysing water usage patterns, optimizing pump operations, and preventing wastage. This chapter discusses the types of data collected, sources of data, data collection methods, and the role of IoT sensors in gathering real-time information.

- **Water Level Data** – Measures the current water level inside the tank.
- **Pump Status Data** – Indicates whether the water pump is ON or OFF.
- **Flow Rate Data** – Tracks the rate at which water is being filled or consumed.
- **Temperature and Humidity Data (optional)** – Helps monitor environmental factors that may affect water quality.

- **Time-stamped Usage Data** – Records when water is used, aiding in consumption analysis.
- **Alert and Notification Logs** – Stores alerts sent to users regarding water level status.

Sources of Data

The data for this system is collected from various sources, including:

- **Ultrasonic Sensors** – Measure water levels in the tank.
- **Float Sensors** – Detect minimum and maximum water levels.
- **Water Flow Sensors** – Measure the flow rate of water entering or leaving the tank.
- **Microcontrollers (Arduino/ESP8266/NodeMCU)** – Process and transmit collected data.
- **Cloud Servers (Firebase, MQTT, Thing speak, etc.)** – Store and manage data for analysis.
- **User Input** – Manual inputs from users to configure settings like threshold levels for water levels.

1.2 Data Collection Methods

The system employs different methods for collecting and transmitting data, such as:

- **Sensor-Based Data Collection**
  - Ultrasonic sensors continuously monitor water levels and send data to the microcontroller.
  - Float sensors detect when the water reaches a certain level and trigger actions accordingly.
  - Flow sensors measure water usage and detect leakages.
- **Automated Data Logging**
  - The microcontroller processes sensor data and logs it to a cloud-based platform for real-time monitoring.
  - Time-stamped data helps in analysing daily, weekly, or monthly water consumption patterns.
- **Wireless Data Transmission**
  - Data is transmitted using Wi-Fi or GSM modules to cloud platforms like Thing Speak, Firebase, or Blynk for remote access.
  - Users receive notifications and alerts on their smartphones or web dashboards.
- **User Feedback and Manual Inputs**
  - Users can manually input threshold levels for alerts (e.g., set minimum and maximum water level limits).
  - User feedback helps improve system efficiency and customization.

1.3 SDLC Model

Introduction to SDLC

The Software Development Life Cycle (SDLC) is a structured approach used in the development of software projects to ensure systematic planning, design, implementation, and maintenance. It helps in efficiently managing project development, reducing errors, and improving quality.

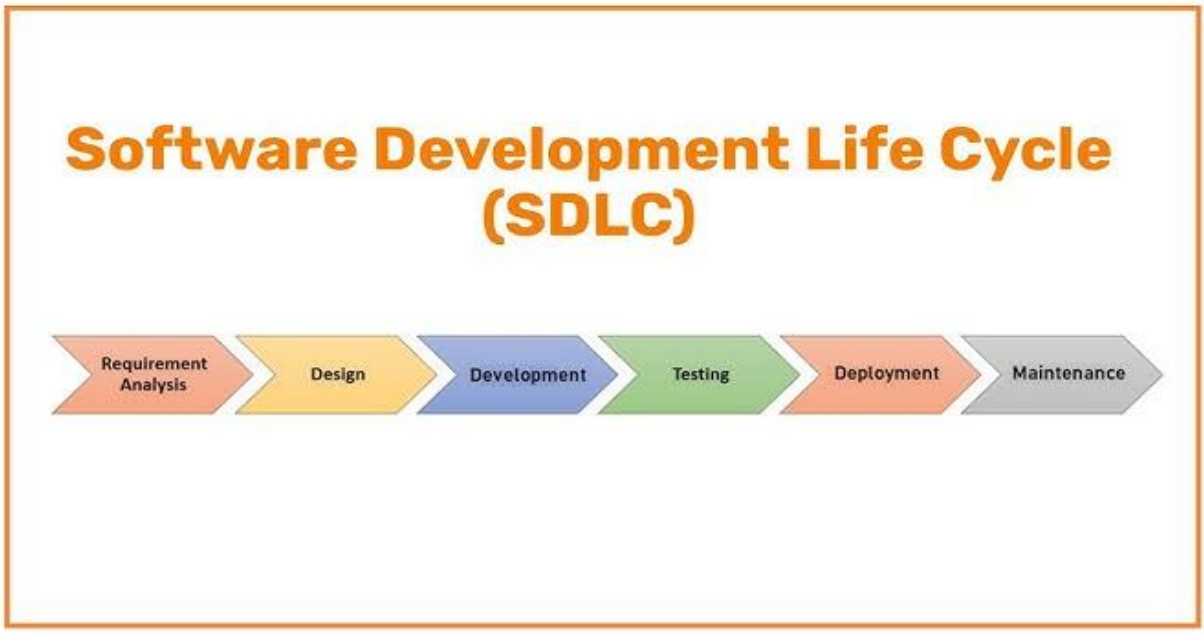
For the Smart Water Tank Monitoring System, an appropriate SDLC model is chosen to ensure smooth development, from requirements gathering to deployment.

SDLC Model Used in the Project

For this project, the Waterfall Model is used due to its sequential and structured approach. The development follows a step-by-step process where each phase is completed before moving to the next.

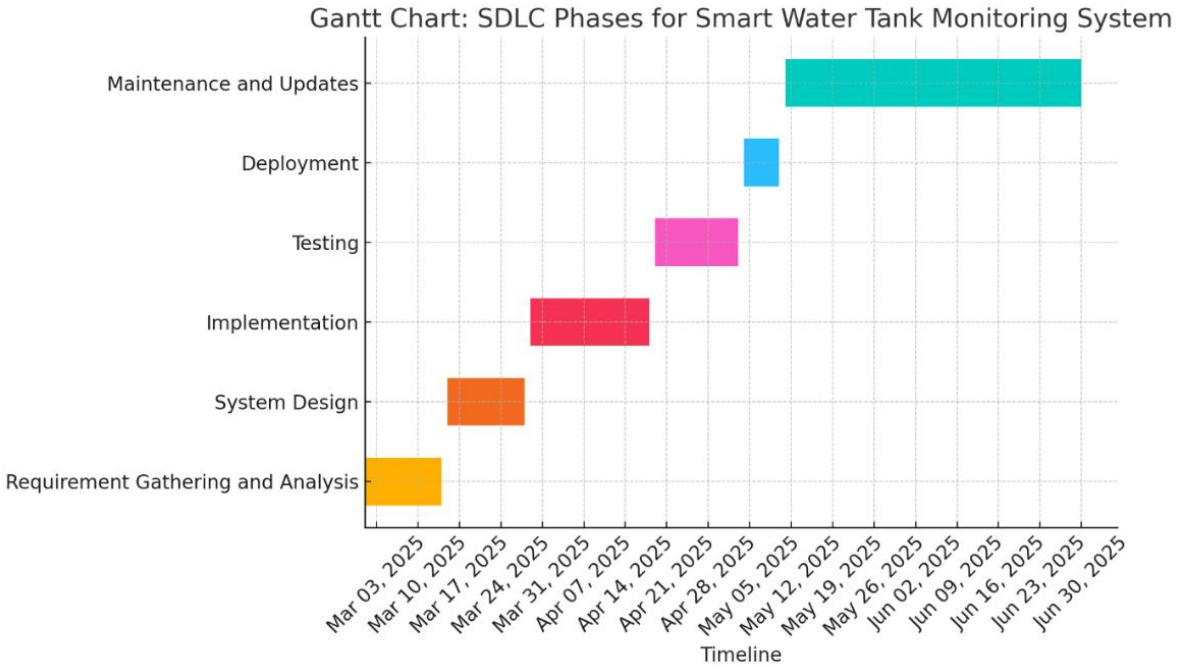
Phases of SDLC in This Project

1. **Requirement Analysis**
   - Identifying the need for an IoT-based water monitoring system.
   - Understanding hardware and software requirements.
   - Defining functional and non-functional requirements.
2. **System Design**
   - Designing the system architecture, including sensor placement, microcontroller selection, and cloud integration.
   - Creating a block diagram of the system components.
   - Planning the user interface for monitoring water levels.
3. **Implementation**
   - Integrating hardware components such as ultrasonic sensors, flow sensors, microcontrollers, and Wi-Fi modules.
   - Writing the software code for sensor data collection and cloud communication.
   - Developing a mobile app or web dashboard for user interaction.
4. **Testing**
   - Conducting unit testing on individual components (sensors, microcontrollers).
   - Performing integration testing to ensure seamless communication between hardware and software.
   - Running system testing to validate functionality and performance.
5. **Deployment**
   - Installing the system in a real-world environment (home, industry, or farm).
   - Configuring cloud connectivity for remote monitoring.
   - Ensuring proper data logging and notifications.
6. **Maintenance & Upgrades**
   - Monitoring system performance and fixing bugs.
   - Updating software for better efficiency and security.
   - Adding new features such as AI-based water usage predictions.



1.4 GANTT CHART

A Gantt chart is a powerful tool for project management, particularly when visually representing a project's timeline and the various phases involved. In the context of the Smart Water Tank Monitoring System, using the Software Development Life Cycle (SDLC) model, the Gantt chart helps break down the project into its key phases, from initiation to completion, and shows the tasks involved in each phase. Here's how the Gantt chart can be used to represent the timeline for this system development:



Chapter 2
Software Requirement Specification (SRS)

2.1 Introduction

The IoT-Based Smart Water Tank Monitoring System is an advanced solution designed to monitor and manage water levels in tanks in real time. By leveraging Internet of Things (IoT) technology, sensors, and automation, the system ensures efficient water usage, prevents overflows, and alerts users when the water level is too low or too high. This system is ideal for households, apartments, industries, and agricultural water management.

2.2 Purpose

The purpose of this document is to define the functional and non-functional requirements for the Smart Water Tank Monitoring System. This system provides real-time water level tracking, notifications, and automation to improve water conservation and management.

2.3 Scope

The Smart Water Tank Monitoring System will include the following features:

- Real-Time Water Level Monitoring: Continuously measure and display water levels.
- Automatic Pump Control: Turn the pump ON/OFF based on predefined water levels.
- IoT Connectivity: Send real-time updates to a cloud platform or mobile app.
- User Alerts & Notifications: Notify users via mobile apps, SMS, or email about critical water levels.
- Data Logging & Analysis: Store historical data for future analysis and optimization.

2.4 Definitions, Acronyms, and Abbreviations

- IoT (Internet of Things): A network of smart devices communicating via the internet.
- MCU (Microcontroller Unit): The processing unit controlling the system.
- Wi-Fi Module: ESP8266 or ESP32 used for IoT connectivity.
- HC-SR04 Sensor: An ultrasonic sensor used to measure water levels.
- Relay Module: A switching device to control the pump.

2.5 References

- IEEE 830-1998 Software Requirements Specification Standard
- Embedded System Design & IoT Development Guidelines

2.6 Overview

This document outlines the functional, non-functional, hardware, and software requirements for the IoT-Based Smart Water Tank Monitoring System. It also describes the system's architecture, constraints, dependencies, and user expectations to ensure efficient development.

3. Overall Description of the Proposed System

3.1 Product Perspective

The Smart Water Tank Monitoring System integrates IoT-based real-time monitoring, water level sensors, and automated motor control. The system consists of a microcontroller, ultrasonic sensors, relay module, Wi-Fi module, mobile/web application, and cloud storage for data logging.

3.1.1 System Interfaces

The system interacts through various interfaces:

- User Interface: Mobile app/web dashboard for real-time monitoring and notifications.
- Hardware Interface: Sensors, microcontroller, and relay module.
- Communication Interface: Data transmission via Wi-Fi to a cloud-based platform.

3.1.2 Hardware Interfaces

- Microcontroller: ESP8266, ESP32, or Arduino.
- Ultrasonic Sensor (HC-SR04): Measures water levels.
- Relay Module: Controls the water pump based on sensor data.
- Power Supply: 5V/12V adapter or battery backup for uninterrupted operation.

3.1.3 Software Interfaces

- Mobile/Web App: Displays real-time data and sends alerts.
- Cloud Integration: Blynk, Firebase, or ThingSpeak for remote access.

3.1.4 Communication Interfaces

- Wi-Fi (ESP8266/ESP32): Connects the system to the cloud for real-time monitoring.
- MQTT/HTTP Protocols: Used for data transmission between devices and the cloud.

3.1.5 Operations

- Continuous Monitoring: Tracks water level in real time.
- Pump Automation: Automatically controls the pump based on water level.
- User Notifications: Alerts users via the mobile app when the tank is empty or full.
- Data Logging: Stores water level history for future analysis.

3.1.6 Site Adaptation Requirements

The system is adaptable for homes, apartments, industries, and agriculture, with considerations for tank size, connectivity options, and environmental conditions.

3.2 Product Functions

- Water Level Measurement: Ultrasonic sensors detect and transmit real-time data.
- Pump Control Automation: The pump switches ON/OFF based on water level thresholds.
- Alerts & Notifications: Users receive alerts.

- Remote Monitoring: Users can check the water level from anywhere using an IoT-enabled dashboard.
- Data Storage: Cloud-based logging of historical water level data.

3.3 User Characteristics

The system is designed for various users:

- Homeowners: Monitor water levels and automate pump control.
- Building Managers: Manage water tanks for apartments and commercial buildings.
- Farmers: Optimize irrigation by monitoring water availability.
- Industrial Users: Ensure efficient water management in factories.

3.4 Constraints

- Power Dependency: Requires continuous power supply for operation.
- Internet Connection: Stable Wi-Fi is needed for IoT-based monitoring.
- Sensor Accuracy: Measurement errors may occur due to tank shape or environmental conditions.

3.5 Assumptions and Dependencies

- The user has a stable internet connection for real-time monitoring.
- The system is placed in a secure and dry location to prevent sensor damage.
- Users must configure threshold levels for notifications and pump automation.

3.6 Future Enhancements

Future versions of the system may include:

- AI-based Water Usage Predictions to optimize consumption.
- Battery Backup for operation during power outages.
- Solar-Powered Sensors for energy-efficient monitoring.

4. Specific Requirements

4.1 External Interfaces

4.1.1 User Interfaces

- Web Dashboard: Displays water level, system status, and alerts.
- Push Notifications & Alerts: Notifies users about low or high-water levels.

4.1.2 Hardware Interfaces

- Microcontroller (ESP8266/ESP32/Arduino)
- Ultrasonic Sensor (HC-SR04)
- Relay Module (For Pump Control)
- Power Supply (5V/12V Adapter or Battery Backup)

4.1.3 Software Interfaces

- Arduino IDE for Firmware Development
- Mobile App (Blynk, Firebase, or Custom Web App)
- Cloud Integration (Thing Speak, Firebase, AWS IoT)

4.1.4 Communication Interfaces

- Wi-Fi for IoT Connectivity
- MQTT / HTTP Protocol for Data Transmission

4.2 Performance Requirements

- Real-Time Response: Water level updates every 1-5 seconds.
- Low Power Consumption: Efficient operation for extended usage.
- Reliable Connectivity: Minimum 99% uptime for IoT-based monitoring.

4.3 Design Constraints

- Secure Data Transmission to prevent unauthorized access.

4.4 Software System Attributes

- **Reliability**: Continuous monitoring with minimal downtime.
- **Security**: Encrypted data transmission to ensure privacy.
- **Scalability:** Ability to support multiple tanks.

4.5 Other Requirements

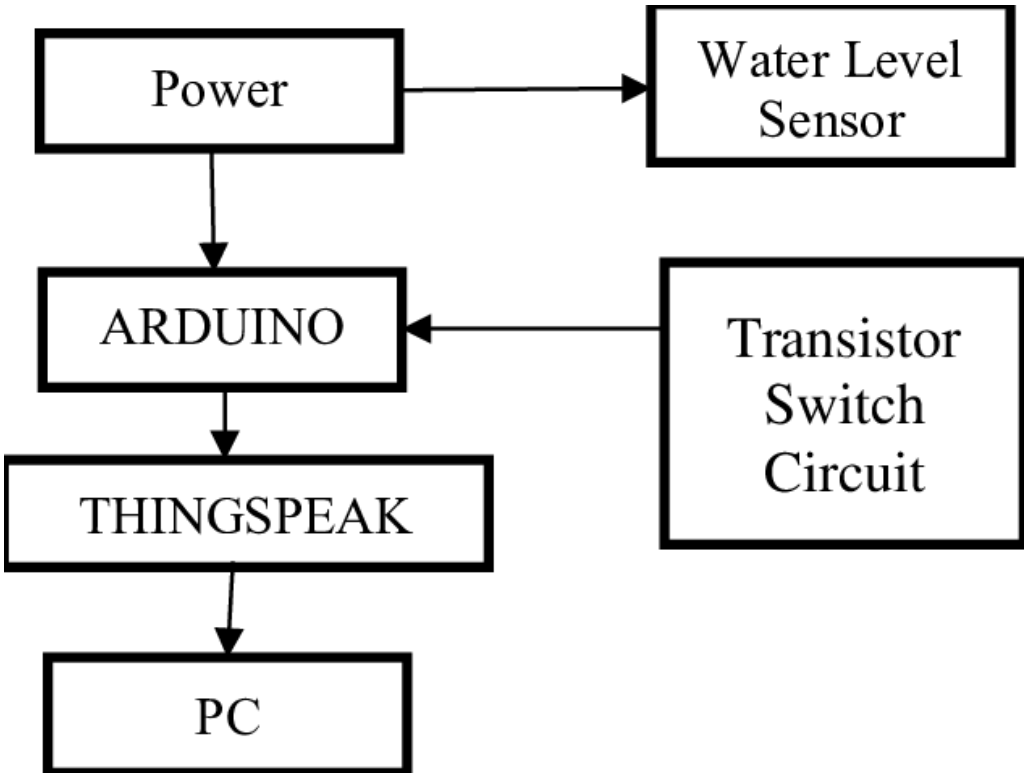
- The system should work in both indoor and outdoor environments.
- Integration with smart home systems for automation.

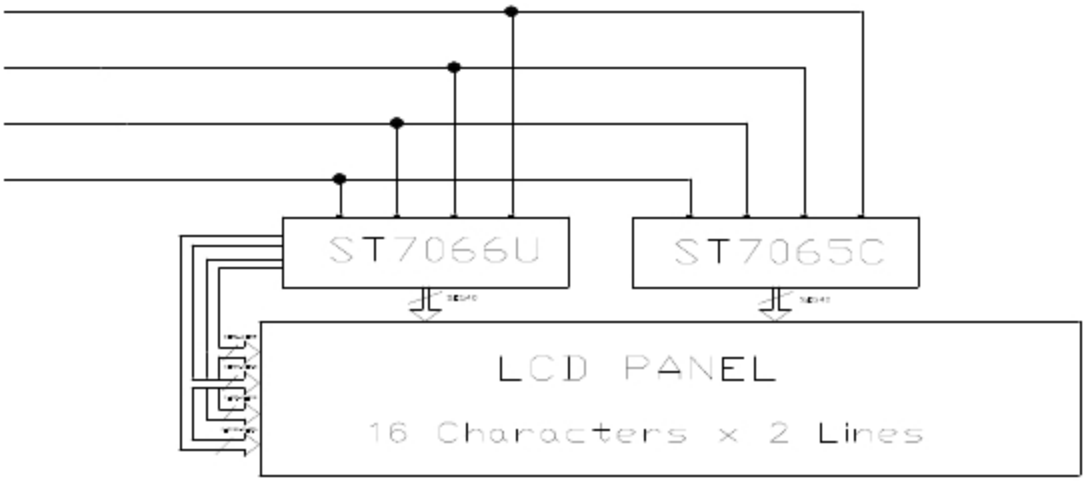
Chapter 3

System Design

3.1 Block Diagram

The block diagram in Figure shows that the power supply is useful for activating the Arduino Mega 2560 and also the water level sensor. This water level sensor uses a sensor strip connected to the transistor switch as an indicator of water level and becomes the data input for Arduino Mega 2560. After the data is input the data will be sent with a WIFI signal to the thingspeak page and the water level display on the runway can be accessed using a PC that is far from the runway. The mechanical design of the water level sensing shown in Figure





3.2 Circuit Diagram

circuit diagram represents a **water level monitoring system** that uses a combination of a **microcontroller (or decoder IC), sensors, an LCD display, and an alert system**. Below is a detailed analysis of the different sections of the circuit.

3.2.1 Water Level Sensors Section

- The left side of the diagram shows multiple **water level probes** labeled **20%, 40%, 60%, 80%, and 100%**.
- These sensors (possibly metallic probes or conductive plates) detect the water level and send signals to the main processing unit.
- The circuit likely works by checking whether the probes are submerged in water, completing an electrical circuit and generating a signal.

3.2.2 Microcontroller/IC (Main Processing Unit)

- The central rectangular block appears to be a **microcontroller or a multiplexer/decoder IC** that takes inputs from the **water level sensors**.
- **Pins labeled A0, A1, A2, A3, A4, and A5** suggest that these are address lines used for selecting input signals.
- It processes the input from the sensors and sends corresponding outputs to the **LCD display**.

3.2.3. LCD Display (16x2) for Water Level Indication

- A **16x2 LCD module (with RS, E, D4-D7 pins labeled)** is used to display the **current water level percentage**.
- The microcontroller sends the processed data to the display, showing levels like **"Water Level: 40%"** or **"Tank Full"**.
- **Resistors (4.7KΩ)** are used for **pull-up/pull-down** connections to ensure proper signal transmission.

3.2.4. Transistor (BC547) with LED Alert System

- A **BC547 NPN transistor** is connected with a **resistor (1KΩ)** and an **LED**.
- This part of the circuit serves as an **alarm/indicator mechanism**, turning ON the LED when the tank is **full or empty**.
- The transistor acts as a **switch**, allowing current to flow through the LED when triggered.
- A **battery is used as the power source** for this alert system

Chapter 4

Components and Coding

4.1 Components

1. Microcontroller & Display

- Arduino (Microcontroller)
- 16x2 LCD Display

2. Resistors

- 4kΩ Resistor (For LCD contrast adjustment)
- 1kΩ Resistors (2 units) (For transistor base and LED)

3. Transistor

- BC547 (NPN Transistor)

4. Sensors & Inputs

- Water Level Sensor (with 5 levels: 100%, 80%, 60%, 40%, 20%)

5. Indicators & Output

- Green LED (Motor Indicator)
- Motor (Blower/Fan)

6. Power & Connections

- Ground (GND)
- Power Supply (VCC, not explicitly shown but necessary)

41. Microcontroller & Display

4.1.1 Arduino (Microcontroller)

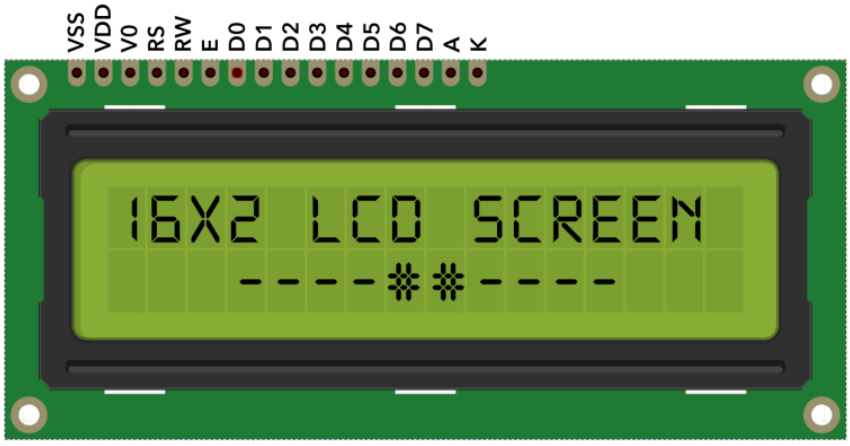
Arduino is an open-source microcontroller board used for electronics projects. It reads input signals from sensors (like the water level sensor in this circuit) and controls outputs (like the LCD display and motor). It processes the data and makes decisions based on programmed instructions.

4.1.2 Features of Arduino in This Circuit:

- Reads water level signals from the sensor.
- Displays the water level percentage on the LCD.
- Controls the motor by turning it ON/OFF based on water level conditions.

4.1.3 16x2 LCD Display

A 16x2 LCD (Liquid Crystal Display) can display 16 characters per line and has 2 lines. It is commonly used in embedded systems to show output data.



No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

4.1.4 Features of the LCD in This Circuit:

- Displays real-time water level readings (e.g., "Water Level: 80%").
- Helps in monitoring the system without needing a computer.
- Connected to the Arduino with multiple pins for data and control.
- Uses a 4kΩ resistor for contrast adjustment.

4.1.5 4kΩ Resistor (For LCD contrast adjustment):

- This resistor is typically used in series with the V0 pin (contrast pin) of an LCD module to control the contrast of the display. Adjusting the contrast is necessary to make the text on the LCD readable under different lighting conditions.

4.1.6 1kΩ Resistors (2 units) (For transistor base and LED):

- **For transistor base:** A 1kΩ resistor is often used in series with the base of a transistor to limit the current flowing into the base and to protect the transistor from excessive current. This ensures proper switching behavior for the transistor.

- **For LED:** A 1kΩ resistor is commonly used in series with an LED to limit the current flowing through the LED, preventing it from burning out. The exact value of the resistor depends on the supply voltage and the LED's forward voltage and current rating.
- **Type:** NPN (Negative-Positive-Negative)
- **Main Uses:**
    - **Switching:** The BC547 can act as a switch to control larger currents with a smaller current at its base. It's often used in low-power applications to switch devices like LEDs or motors.
    - **Amplification:** It can also amplify weak electrical signals in circuits like audio amplifiers.
- **Pin 1 (Collector):** The current flows out from the collector. This pin connects to the load or the output part of the circuit.
- **Pin 2 (Base):** The base controls the transistor's switching. A small current flowing into the base allows current to flow from the collector to the emitter.
- **Pin 3 (Emitter):** The current flows into the emitter. This pin connects to ground or the negative side of the circuit.

Typical Use Cases:

- **Switching on/off LEDs:** Using a current-limiting resistor (like the 1kΩ you mentioned earlier), the base can be controlled by a microcontroller or any logic circuit to switch on/off the LED.
- **Amplifying signals:** In a common-emitter configuration, the BC547 can amplify small signals, useful in audio circuits.

The **Water Level Sensor** you mentioned, with 5 levels (100%, 80%, 60%, 40%, 20%), is likely a type of analog or digital sensor used to detect the water level in a tank or container. These sensors typically provide different outputs based on the level of water in contact with the sensor, which can then be used to control various devices (such as pumps or alarms).

4.1.7 Green LED (Motor Indicator):

- **Purpose:** The green LED is commonly used as an indicator light, often to show that a motor or system is running or operating normally. In your case, it will likely indicate whether the **blower/fan motor** is on or off.
- **Connections:**
    - The LED should be connected in series with a current-limiting resistor (such as the **1kΩ resistor** you mentioned earlier) to avoid damaging the LED.
    - The anode (positive side) connects to the positive voltage supply (such as 5V or 12V depending on your system), while the cathode (negative side) connects to the control circuit, such as a transistor or microcontroller, which can turn it on or off.
- **Usage:** The LED will light up when the motor is running (or the system is active), providing a visual indication to the user. It could be controlled by a transistor or a relay connected to a microcontroller.

4.1.8 Motor (Blower/Fan):

- **Purpose:** The motor is used to drive the blower or fan. It will be responsible for creating airflow in a cooling system, exhaust system, or ventilation setup.

- **Control Circuit:**
    - **Transistor or Relay:** The motor will likely require a higher current than what a microcontroller can supply directly. Therefore, a **transistor** (like the **BC547** you mentioned earlier) or a **relay** is used to switch the motor on and off.
    - **Transistor-based Switching:** The base of the **BC547** would be connected to the control signal (like from a microcontroller or sensor), and the emitter would go to ground. The collector would connect to the negative terminal of the motor, with the positive terminal of the motor connected to the supply voltage.
    - **Motor Driver (if needed):** Depending on the type and voltage of the motor, you might also need a dedicated **motor driver circuit** (like an L298N for DC motors or an H-Bridge) to handle the motor's current requirements.

4.1.8 Ground (GND):

- **Purpose:** The **ground (GND)** is the reference point for the entire circuit. All components in the system need a common ground connection to ensure that their signals and voltage levels are properly referenced to the same point.
- **Connections:**
    - All components (sensors, motor, LEDs, microcontroller, etc.) should be connected to the **GND pin** on the power supply or microcontroller (e.g., Arduino).
    - The **GND of the power supply** connects to the GND of the microcontroller and other components, ensuring that all parts of the circuit have the same voltage reference.
    - For the **transistor (BC547)**, the **emitter** is connected to the ground (GND), which completes the circuit for the base and collector to work correctly.

4.1.9 Power Supply (VCC):

- **Purpose:** The **VCC** (or supply voltage) provides the necessary power to drive all components in the system, including sensors, LEDs, the motor, and the microcontroller.
- **Connections:**
    - **Microcontroller:** The **VCC pin** (usually 5V or 3.3V depending on the microcontroller) connects to the positive rail of your circuit to power the microcontroller and possibly low-power components like sensors or LEDs.
    - **Motor (Blower/Fan):** The motor will likely require a higher voltage (e.g., 12V). So, you'll need a **separate power supply** for the motor. If your motor operates at 12V, then **VCC for the motor** will be connected to a 12V power supply, while the microcontroller and other components use the lower voltage (e.g., 5V or 3.3V).
    - **Transistor (BC547):** The **collector** of the BC547 transistor will connect to the motor, and the **VCC (supply voltage)** for the motor will connect to the motor's positive terminal. The power to the transistor comes from the same supply that powers the motor, ensuring that the transistor can handle the current needed for the motor.

5 General Connection Summary:

- **Power Supply (VCC)** is connected to all components that need a power source (like the motor, microcontroller, and sensors).
- **Ground (GND)** should be connected to all components to provide a common reference point.
- Ensure that different voltage requirements (e.g., 5V for the microcontroller and 12V for the motor) are handled correctly, potentially using separate power supplies or voltage regulators.
- **Motor control**: Use a transistor (like **BC547**) or a relay to switch the motor on and off based on input from sensors or the microcontroller.

Example Setup:

- **VCC (5V or 12V)**: Powers the microcontroller, sensors, and motor (depending on voltage requirements).
- **GND**: Common ground shared by all components.
- **Motor**: Powered separately by **12V**, controlled through the **transistor (BC547)**.
- **LED**: Connected to **VCC** through a current-limiting resistor, with the cathode to **GND**

5. code

```
#include <LiquidCrystal.h>

int level_1   = A1;

int level_2   = A2;

int level_3   = A3;

int level_4   = A4;

int level_5   = A5;

int pin_motor = 8;

int led = 9;

int a;

int b;

int c;

int d;

int e;

int r; //Water Pump status

int z=100;

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup()

{

pinMode(level_1,INPUT);

pinMode(level_2,INPUT);
```

```
pinMode(level_3,INPUT);
pinMode(level_4,INPUT);
pinMode(level_5,INPUT);
pinMode(pin_motor,OUTPUT);
pinMode(led,OUTPUT);
lcd.begin(16, 2);
}
void loop()
{
r=digitalRead(pin_motor);
a=analogRead(level_1);
b=analogRead(level_2);
c=analogRead(level_3);
d=analogRead(level_4);
e=analogRead(level_5);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("MORY");
//lcd.setCursor(0,1);
//lcd.print("Water Level ");
if(e>z && d>z && c>z && b>z && a>z )
{
  digitalWrite(pin_motor,LOW);
  lcd.setCursor(0,0);
  lcd.print("Tank is 100%FULL");
  digitalWrite(led,HIGH);
  delay(500);
  digitalWrite(led,LOW);
  delay(500);
{
  lcd.setCursor(0,0);
  lcd.print("Tank is 80% FULL");
}
```

```
else if(e<z && d<z && c>z && b>z && a>z )
{
  lcd.setCursor(0,0);
  lcd.print("Tank is 60% FULL");
}
else if(e<z && d<z && c<z && b>z && a>z )
{
  lcd.setCursor(0,0);
  lcd.print("Tank is 40% FULL    ");
}
else if(e<z && d<z && c<z && b<z && a>z )
{
  lcd.setCursor(0,0);
  lcd.print("Tank is 20% FULL");
}
else if(e<z && d<z && c<z && b<z && a<z )
{
  digitalWrite(pin_motor,HIGH);
  lcd.setCursor(0,0);
  lcd.print("Tank is EMPTY    ");
}
if(r==LOW)
{
  lcd.setCursor(0,1);
  lcd.print("Pump is (OFF)     ");
}
else
{
  lcd.setCursor(0,1);
  lcd.print("Pump is (ON)      ");
}
  delay(100);
  lcd.clear();
```

```
}
```

CHAPTER: -05

System Testing

INTRODUCTION

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- Meets the requirements that guided its design and development,
- Works as expected,
- Can be implemented with the same characteristics, and satisfies the needs of stakeholders.

Testing can never completely identify all the defects within software. Instead, it furnishes a criticism or comparison that compares the state and behavior of the product against oracles—principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts, comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria. A very fundamental problem with software testing is that testing under all combinations of inputs and preconditions (initial state) is not feasible, even with a simple product. This means that the number of defects in a software product can be very large and defects that occur infrequently are difficult to find in testing. More significantly, non-functional dimensions of quality (how it is supposed to be versus what it is supposed to do)— usability, scalability, performance, compatibility, reliability—can be highly subjective; something that constitutes sufficient value to one person may be intolerable to another.

5.1 FUNCTIONAL TESTING

In software development, functional testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of functional testing is to ensure that the system performs and behaves as expected under given conditions. It primarily focuses on testing the functionality of the software application by providing appropriate input and verifying the output against the expected results.

Characteristics of Functional Testing

It verifies that the application performs as intended based on the defined functional requirements. It does not concern itself with the underlying code structure.

- It ensures that each function of the software application works in accordance with the requirement specification.
- It is typically conducted manually or using automated testing tools.

Functional Testing Process

1. Requirement Analysis: Identify and understand the functional requirements of the system.

2. Test Case Design: Develop test cases that outline input data, execution steps, and expected outcomes.

3. Test Execution: Run the designed test cases and document the results.

4. Defect Reporting: Log and track any defects or inconsistencies found during testing.

5. Retesting and Regression Testing: Verify fixed defects and ensure new changes have not introduced additional issues.

Types of Functional Testing

Functional testing includes various subtypes, such as:

- Smoke Testing: A preliminary test to check the basic functionality of the software.
- Sanity Testing: A quick, focused test to validate specific functionalities after minor changes.
- Regression Testing: Ensuring new code changes do not negatively impact existing functionalities.
- User Acceptance Testing (UAT): Validating the software from an end-user perspective before deployment.
- Integration Testing: Testing the interfaces and interactions between integrated components or systems.

Benefits of Functional Testing

• Ensures that the software meets business and user requirements.

• Identifies defects early in the development cycle.

• Improves software quality and reliability.

• Provides confidence in the product before its release.

Functional testing is an essential part of the software testing lifecycle, ensuring that an application meets its intended functional requirements. By systematically verifying the expected behavior of software, functional testing contributes to building robust, error-free, and high quality applications that align with business needs.

5.2 User Interface Testing

Not applicable

5.3 Navigation Testing

Not applicable

5.4. Form testing:-

Not applicable

5.5. Database testing:-

Not applicable

Chapter: - 6

Scope of Improvement, Summary, and Conclusions

6.1 Scope of Improvement

While the Smart Water Tank Monitoring System provides an efficient solution for real-time water management, there is always room for enhancements. The following areas highlight possible improvements to further optimize performance, accuracy, and scalability.

6.1.1 Integration of Artificial Intelligence (AI) and Machine Learning (ML)

- **Predictive Analytics:** Implementing AI and ML algorithms can help predict water usage patterns based on historical data. This can optimize pump operation schedules, reducing unnecessary energy consumption.
- **Leakage Detection:** Advanced AI models can detect abnormal water consumption patterns and alert users about possible leaks or faulty plumbing.
- **Smart Decision-Making:** AI-based decision-making can automate pump activation and shutdown, considering weather conditions, user behavior, and real-time water demand.

6.1.2 Sensor Accuracy and Reliability

- **Multi-Sensor Integration:** Instead of relying solely on **ultrasonic sensors**, incorporating **capacitive and flow sensors** can improve accuracy in detecting water levels.
- **Self-Calibrating Sensors:** Some sensors may require regular calibration due to environmental factors such as humidity and temperature. Introducing **self-calibrating sensors** can maintain accurate readings over time.
- **Error Reduction Algorithms:** Implementing software-based filtering techniques, such as **Kalman filtering**, can reduce noise in sensor data and improve measurement accuracy.

6.1.3 Energy Efficiency and Sustainability

- **Solar Power Integration:** Adding **solar panels** to power the system will reduce electricity consumption and make it more environmentally friendly.
- **Low-Power IoT Communication:** Switching to **LoRaWAN (Long Range Wide Area Network) or Zigbee** instead of Wi-Fi can help reduce power consumption and improve connectivity over long distances.

6.1.4 User Interface and Accessibility Improvements

- **Advanced Mobile App & Web Dashboard:**
    - An interactive dashboard with **real-time graphs and historical data analysis** will help users understand their water consumption trends.
    - A **multi-user access system** can be developed for families or institutions to monitor different water tanks from a single interface.
- **Voice Command & Smart Home Integration:**
    - Integrating the system with **Google Assistant or Amazon Alexa** will allow users to check water levels using voice commands.

- Smart home automation features can trigger water pumps based on pre-set voice or mobile commands.

6.1.5 Enhanced Security and Data Protection

- **End-to-End Encryption:** Strong encryption methods (such as **AES-256**) should be used to ensure secure data transmission between the IoT device and the cloud.
- **Multi-Cloud Backup:** Storing data across multiple cloud services (AWS, Google Firebase, Thingspeak) will prevent data loss in case of server failures.

6.1.6 Multi-Tank and Large-Scale Integration

- **Smart City Integration:** The system can be connected to **municipal water supply systems** for efficient distribution in urban areas.
- **Multi-Tank Monitoring:** Enhancing the system to manage **multiple water tanks** in large buildings, industries, or farms will improve scalability.
- **Industrial and Agricultural Applications:**
  - Implementing automated irrigation scheduling in **agriculture** based on soil moisture and weather data.
  - Water management in **factories and commercial buildings** can help reduce wastage and optimize supply.

6.2 Summary

Water is a vital resource, and efficient management is necessary to prevent wastage, shortages, and overuse. Traditional water tank systems require manual monitoring, which can be inefficient and time-consuming. The Smart Water Tank Monitoring System provides an automated, real-time monitoring solution using IoT sensors, microcontrollers, and cloud-based platforms.

This project follows the Waterfall SDLC model, ensuring a structured development process from requirement analysis to implementation, testing, and deployment. The system offers real-time water level updates, automated pump control, mobile notifications, and cloud storage. It benefits households, industries, and agricultural applications by optimizing water consumption and preventing overflow or wastage.

6.3 Conclusion

The Smart Water Tank Monitoring System successfully demonstrates how IoT technology can be leveraged to automate water management, reduce wastage, and improve efficiency. By integrating sensors, microcontrollers, and cloud-based platforms, the system provides real-time water level monitoring, automated pump control, and remote accessibility through a user-friendly interface.

This project addresses common challenges in traditional water management, such as manual monitoring, overflow, and shortages, by offering a fully automated solution. The system is particularly beneficial for households, industries, and agriculture, ensuring optimal water usage and conservation.