1. ABSTRACT

Access to clean and safe water is essential for health, agriculture, and environmental sustainability. This project aims to design and develop a low-cost, portable water quality monitoring system using the Arduino Uno microcontroller. The system integrates various sensors to measure key parameters such as pH, temperature, turbidity, and Total Dissolved Solids (TDS), which are critical indicators of water quality. Real-time data from the sensors are processed by the Arduino and displayed on an LCD screen, with optional transmission to a computer or IoT platform for remote monitoring.

The primary objective is to enable continuous and reliable assessment of water conditions, particularly in areas lacking advanced laboratory facilities. Alerts can be triggered when the measured values exceed predefined safe limits, allowing for timely response to contamination. The system is calibrated to ensure accuracy and can be used in both freshwater and wastewater sources. By combining embedded systems and environmental science, this project provides an innovative, scalable, and affordable solution to address global water quality challenges.

1.1. Aims

- To monitor key water quality parameters such as temperature, pH, turbidity, and TDS (Total Dissolved Solids) using appropriate sensors.
- To develop a low-cost, portable water quality monitoring system that can be deployed in remote or underdeveloped areas.
- To interface water quality sensors with Arduino Uno and display real-time data on an LCD screen or transmit it to a computer or cloud platform.

- To alert users of water contamination when sensor readings exceed safe threshold values.
- To promote environmental awareness and public health by providing accessible tools for water quality testing.
- To explore the integration of IoT features such as wireless data transmission (e.g., via Wi-Fi or GSM module) for remote monitoring.

1.2. Objectives of the project

- To design and build a water quality monitoring system using Arduino Uno and compatible sensors.
- To measure essential water parameters such as pH, temperature, turbidity, and TDS in real-time.
- To calibrate and validate sensor data for accuracy and reliability in various water samples.
- To process and display water quality data on an LCD screen or transmit it to a computer/mobile device for analysis.
- To set threshold values for each parameter and generate alerts when water quality deviates from safe standards.
- To provide a cost-effective and scalable solution for water testing in both urban and rural settings.
- To develop a user-friendly interface for data visualization and interpretation.
- To enhance skills in embedded systems, sensor integration, and environmental monitoring.

2. INTRODUCTION

This project aims to address a specific problem by developing a practical and efficient solution through systematic planning, research, and implementation. The primary objective is to apply theoretical knowledge to real-world scenarios, thereby enhancing understanding and skill development in the chosen field. The project involves identifying key requirements, designing a methodology, and executing the proposed solution while adhering to defined goals and constraints. The project began with extensive research into the topic, including a review of existing literature, technologies, and methodologies relevant to the subject. Based on this foundation, a detailed project plan was developed outlining the scope, objectives, and expected outcomes. The work involved multiple phases such as requirement gathering, system design, development, testing, and evaluation. Throughout the process, care was taken to ensure adherence to best practices, industry standards, and ethical considerations.

This report outlines the motivation, objectives, scope, and structure of the project, providing a comprehensive overview of the work undertaken and the results achieved. This report is structured to provide a clear understanding of the project's background, development process, and results. It includes insights into the challenges encountered, the solutions devised, and the lessons learned. Ultimately, this project not only contributes to academic growth but also offers potential benefits for practical implementation in relevant sectors. The following chapters delve deeper into each stage of the project, providing a detailed account of the work accomplished.

The project aimed to improve efficiency, accuracy, and usability within its chosen domain. Emphasis was placed on research, innovation, and application of modern tools and technologies. A structured approach was

followed throughout the development process, including requirement analysis, system design, and functional testing. The team collaborated effectively to divide responsibilities and ensure timely progress. Extensive documentation was maintained at every stage to record challenges, observations, and key findings. The tools and techniques used were selected based on relevance, performance, and scalability. Quality assurance was achieved through rigorous testing and validation procedures. The final output met the defined objectives and demonstrated the practical feasibility of the proposed solution. Throughout the project, skills in problem-solving, teamwork, and technical communication were significantly improved. The experience also helped to bridge the gap between academic knowledge and industrial practices.

2.1. IoT

The Internet of Things, otherwise known as IoT in the simplest sense, refers to the concept of connecting physical devices, machines, software, and objects to the Internet. In a broader sense, it is a dynamic and global network infrastructure, in which intelligent objects and entities are used in conjunction with actuators, electronics, sensors, software and connectivity to enhance connection, collection and data exchange.

This type of network generally has a large number of nodes that interact with the environment and exchange data, whilst reacting to events or triggering actions to exert control or change upon the physical world. By sharing and acting on shared data contributed by individual parts, an IoT system would be greater than the sum of its parts. Each network node is considered smart and consumes little resources such as data processing and data storage power as well as energy consumption

The Internet of Things (IoT) is already deeply integrated into daily life, making tasks more convenient, efficient, and connected. In homes, IoT powers smart devices like voice assistants, smart lights, thermostats, and security systems that can be controlled remotely through mobile apps. In healthcare, wearable fitness trackers and remote monitoring devices help individuals track their health in real time, aiding in early diagnosis and chronic disease management. Transportation has become more intelligent with GPS navigation, real-time traffic updates, and smart vehicle systems that monitor performance and safety. In retail, IoT is used for managing inventory through smart shelves and providing personalized shopping experiences. Even workplaces have adopted IoT for energy-efficient automation, smart meeting rooms, and employee tracking systems.

2.2. Scope of the project

TriSense water analysis using an Arduino Uno equipped with a pH sensor, Turbidity and waterflow sensor is a valuable project, particularly in the context of environmental monitoring and water quality assessment. Here's an overview of the scope:

1. Environmental Monitoring

- Water Quality Assessment: By continuously monitoring pH, turbidity and waterflow rate, you can assess the water quality in real-time.
- This is crucial for detecting pollution events, such as acid rain or industrial discharges.

2. Data Collection and Analysis

- Real-Time Data Logging: The Arduino Uno can collect data in real-time, which can be stored locally on an SD card or sent to a cloud platform for remote access and analysis.
- Trend Analysis: Over time, the collected data can be used to analyse trends, seasonal variations, and the impact of different factors (e.g., rainfall, industrial activities) on river water quality.

3. Public Health and Safety

- Drinking Water Safety: Monitoring pH and turbidity levels is essential for ensuring that water remains within safe limits for human consumption.
- Aquatic Life: The sensors can help track conditions that affect aquatic ecosystems, such as changes in acidity or sediment levels, which can have significant impacts on fish and other organisms.

4. Education and Research

- Educational Projects: This project serves as a hands-on educational tool for students learning about environmental science, electronics, and data analysis.
- Research Applications: Researchers can use the setup to gather data for studies on water quality, pollution, and the effects of human activities on natural water bodies.

5. Community and Policy Engagement

- Community Awareness: Data can be shared with local communities to raise awareness about the importance of water quality and encourage conservation efforts.

- Policy Making: The collected data can inform policymakers and support the creation of regulations to protect water resources.

6. Challenges and Considerations

- Calibration and Maintenance: Regular calibration of sensors is necessary to ensure accuracy.
- The system also needs to be robust enough to handle environmental conditions.

2.3. Project objective

To develop a real-time monitoring system using Arduino Uno that continuously measures and analyses the pH, turbidity and waterflow levels of water, with the aim of assessing water quality, detecting potential pollutants, and providing timely data for environmental management and public safety.

Specific Goals:

1. **Continuous Monitoring:** Implement a system that continuously monitors the pH, turbidity and flowrate of water in real-time.
2. **Data Collection and Logging**: Design and build a data logging system that records pH and turbidity values over time for further analysis. The data should be stored locally and transmitted to a remote server or cloud platform.
3. **Data Visualization:** Create a user-friendly interface or dashboard that displays real-time data and historical trends, enabling easy interpretation and decision-making.
4. **Threshold Detection and Alerts:** Set up threshold limits for pH and turbidity values. The system should trigger alerts when the water quality

deviates from these predefined safe limits, indicating potential pollution or other environmental concerns.

5. **Data Visualization:** Create a user-friendly interface or dashboard that displays real-time data and historical trends, enabling easy interpretation and decision-making.
6. **Low-Cost and Scalable Solution:** Ensure the system is cost-effective, easy to deploy, and scalable, so it can be implemented in various locations for widespread environmental monitoring.
7. **Educational and Community Impact:** Promote awareness and understanding of water quality issues by making the project accessible as an educational tool for students, researchers, and community members.

3. LITERATURE REVIEW

3.1. Review of Key Research Studies:

1. Ganga River Monitoring by CPCB

The Central Pollution Control Board (CPCB) deployed IoT-based real-time water quality monitoring stations along the Ganga River. These systems use sensors to track parameters like pH, DO (Dissolved Oxygen), and conductivity. Data is transmitted live to central servers to detect pollution sources quickly, aiding government cleanup efforts.

2 Comprehensible River Water Management (Sri Vidyashankar M.H, 2024)

River water valuation is extremely significant in green globalization. To verify (Sri Vidyashankar M.H, 2024) the safe supply of drinking water, the quality must be monitored in real time. This work will focus on designing a

cost-effective system based on IoT for real-time river water assessment at a single location. A variety of sensors are employed to measure the temperature, PH, turbidity, and TDS of the water. The core controller can process the measured values from the sensors.

3. Hyderabad Smart Water Grid

Hyderabad's water board (HMWSSB) incorporated IoT into its water supply system, using sensors to detect leaks and monitor water quality across treatment plants and pipelines. This project improved water distribution efficiency and helped deliver safe drinking water to urban populations.

4. IIT Kanpur's Rural Water Monitoring Kit

IIT Kanpur developed a low-cost, solar-powered IoT kit for monitoring handpump and well water quality in rural Uttar Pradesh. The system uses GSM modules to send alerts about unsafe water, helping villagers report contamination and avoid waterborne diseases.

5. Pune Smart City Water Project

Under the Smart City Mission, Pune Municipal Corporation deployed smart sensors to monitor water quality in its supply system. Real-time data helped reduce water losses and ensured compliance with safety standards, enhancing citizen trust in municipal services.

6. Gujarat Jal Jeevan Mission Monitoring

Gujarat implemented IoT-based systems in rural villages to support the Jal Jeevan Mission. Sensors monitored drinking water quality and flow, ensuring safe, adequate supply to households. The system also generated alerts when parameters like turbidity or residual chlorine deviated from standards.

4. ABOUT PROGRAMMING LANGUAGE

4.1. Introduction to Embedded C

Embedded systems have become an integral part of modern technology, from consumer electronics and automotive systems to industrial machines and medical devices. At the core of these systems lies a specialized form of programming known as Embedded C. It is an extension of the standard C programming language tailored for use in programming microcontrollers and other embedded devices. This report provides a detailed overview of Embedded C, emphasizing its features, applications, and the challenges it presents in embedded system development.

What is Embedded C?

Embedded C is a set of language extensions for the C programming language to support embedded processor development. It enables developers to write efficient and reliable code that directly interacts with the hardware.

Unlike high-level programming languages that abstract the hardware, Embedded C provides low-level control, making it ideal for use in systems with limited memory, processing power, and real-time requirements.

Embedded C is widely used in microcontroller programming, including platforms like AVR, PIC, ARM, and popular development boards such as Arduino and STM32.

4.2. Features of Embedded C

1. Low-Level Hardware Access

One of the main advantages of Embedded C is the ability to manipulate hardware directly. Programmers can access registers, ports, and memory

locations using pointers and specific memory addresses, which is essential for device control.

2. Efficiency and Performance

Embedded systems typically run on limited hardware. Embedded C produces lightweight, efficient, and fast-executing code, which is essential in systems with minimal RAM, ROM, and processing power.

3. Portability

Though Embedded C is closely tied to hardware, it remains relatively portable between platforms with similar architectures. This allows code to be reused across different devices with minimal changes.

4. Real-Time Operation Support

Many embedded applications require real-time response to events (e.g., in robotics, medical devices). Embedded C allows the use of interrupts and timers to implement deterministic behaviour and time-critical operations.

5. Modularity and Code Reusability

Like standard C, Embedded C supports modular code development. Functions and libraries can be reused across different projects, improving maintainability and reducing development time.

4.3. Challenges of Embedded C programming language

1. Limited Debugging Tools

Debugging embedded systems is more complex than debugging desktop applications. Embedded C programs often run on bare-metal

hardware with limited feedback mechanisms, making it difficult to trace bugs without specialized tools like logic analysers or in-circuit debuggers.

2. Memory Constraints

Embedded systems typically have limited RAM and ROM. Programmers must optimize memory usage carefully, and even small memory leaks or inefficient code can lead to system failure or unexpected behaviour.

3. Hardware Dependency

Embedded C code is often tightly coupled with the underlying hardware. This makes it less portable compared to higher-level languages. Any change in microcontroller or peripheral requires significant code modification.

4. Timing and Real-Time Issues

Real-time applications require precise timing. Mistakes in timer settings, delays, or interrupt handling can cause systems to fail, especially in safety-critical applications like automotive or medical devices.

5. SOFTWARE

5.1. Arduino IDE

Single board micro-controller kits are created and manufactured by Arduino, an open-source hardware and software project, user community, and company. Its goods are covered by the GNU General Public License (GPL) or the GNU Lesser General Public License (LGPL). Several types of microprocessors and controllers are used in Arduino board designs. Several extension boards. (shields"), breadboards (for prototyping), and other circuits may be interfaced to the hoards' sets of digital and analogue input/output (1/0) pins.

Fig: - ARDUINO IDE

Key Features:

1. Programming Language:

The Arduino IDE utilizes a simplified version of C++ tailored for Arduino boards. This language abstraction shields users from complex syntax, making it easier to focus on hardware interaction and project logic.

2. User Interface:

The IDE provides an intuitive interface comprising a text editor, a message area for compiling and uploading feedback, and a serial monitor for real-time communication with the connected Arduino board. A toolbar offers quick access to essential functions like uploading sketches and verifying code syntax.

3. Sketches:

Arduino programs, known as sketches, are written and saved within the IDE using the .ino file extension. This structure encourages modular programming and rapid prototyping, essential for iterating on hardware designs.

4. Library Support:

Extensive library support within the IDE allows developers to integrate pre-written code modules for various sensors, actuators, and communication protocols. This capability accelerates development by leveraging tested and optimized code snippets.

5. Board Manager:

The IDE includes a board manager facilitating seamless integration of new Arduino-compatible hardware definitions. This feature supports a wide range of boards beyond the standard Arduino Uno, catering to diverse project requirements.

6**. Cross-Platform Compatibility:**

Available for Windows, macOS, and Linux operating systems, the Arduino IDE ensures broad accessibility across different Internship computing environments, fostering a diverse community of users and developers.

7. Debugging and Testing:

The IDE's serial monitor plays a crucial role in debugging Arduino projects, enabling developers to monitor sensor outputs, debug logic errors, and interact with their projects in real-time via serial communication.

8. Open-Source Foundation:

As open-source software, the Arduino IDE encourages community collaboration and innovation. Its source code is freely available, allowing enthusiasts to modify, enhance, and contribute to its ongoing development

Advantages:

1. Simplicity and Accessibility:

The Arduino IDE is designed to be user friendly, especially for beginners in programming and electronics. Its simplified version of C++ and straightforward interface make it easy to get started with coding for Arduino boards.

2. Integrated Development Environment:

It provides an all-in-one platform for writing, compiling, and uploading code to Arduino boards. This integration streamlines the development process, reducing the need for external tools or complex setup.

5.2. Arduino programming language

1. Based on C/C++:

Arduino uses a simplified version of the C/C++ language. It retains core syntax and structure, making it powerful yet easy to learn. Beginners can pick it up quickly without deep programming knowledge. Advanced users can write optimized and modular code. This balance makes Arduino versatile for all skill levels.

2. Arduino IDE

The Arduino IDE is where you write, compile, and upload code. It features a simple interface designed for electronics projects. It supports syntax highlighting, serial monitoring, and board selection. One-click buttons simplify uploading and debugging. It works on Windows, macOS, and Linux.

3. Sketch Structure

Each Arduino program is called a "sketch". Sketches must include two main functions: setup () and loop (). setup () runs once to initialize settings. loop () runs repeatedly to control the hardware continuously. This simple structure mirrors the operation of embedded systems.

4. Pin Configuration

Use pinMode() to set pins as input or output. This function is usually called in the setup () section. Correct pin configuration is crucial for hardware interaction. Digital and analog pins are handled differently.

5. Digital & Analog I/O

Use digitalWrite() and digitalRead() for digital pins, analogRead() reads sensor values from analog pins, analogWrite() sends PWM signals to control outputs like LED brightness.

6. Delay Function

Delay(ms) pauses program execution for a specified time. It's useful for timing events like LED blinking.

7. Serial Communication

Use Serial.begin() to start serial data transfer. Serial.print() sends text to the Serial Monitor for debugging. It's essential for reading sensor values or diagnosing issues. Helps visualize data during development.

8. Libraries

Libraries extend functionality, e.g., Servo.h, Wire.h, LiquidCrystal.h. Include them using #include <library_name.h>. Many are pre-installed or

available via the Library Manager. Great for saving time and avoiding complex low-level code.

5.3. BLYNK

Blynk is a popular Internet of Things (IoT) platform that allows developers and hobbyists to build web and mobile applications for the control and monitoring of connected devices. It's especially useful for projects like water quality analysis, where sensor data needs to be collected, displayed, and monitored remotely.

1. Mobile App: Blynk provides a mobile app (available for iOS and Android) that serves as a control interface for IoT projects. Users can create customizable dashboards (known as projects) by dragging and dropping widgets such as buttons, sliders, graphs, and displays. These widgets allow users to interact with and monitor their IoT devices in real-time. Internship

2. Cloud Server: Blynk operates on a cloud-based infrastructure, facilitating remote connectivity and management of IoT devices. The cloud server handles communication between the mobile app and connected devices, ensuring seamless data exchange and control from anywhere with an internet connection.

3. Hardware Libraries: Blynk offers libraries and example code for popular hardware platforms including Arduino, ESP8266, ESP32, Raspberry Pi, and others. These libraries simplify the integration of sensors, actuators, and other components into IoT projects, enabling rapid prototyping and development.

4. Widget Customization: Widgets in Blynk's mobile app are highly customizable, allowing users to configure their appearance, behaviour, and

interaction logic. This flexibility enables developers to create intuitive user interfaces tailored to specific project requirements.

5. Security: Security is a fundamental aspect of Blynk. It employs industry standard encryption protocols (SSL/TLS) to protect data transmitted between devices and the cloud server. User authentication methods and secure token-based communication ensure the confidentiality and integrity of IoT data.

Why Blynk?

- **User-Friendly Interface:** Intuitive drag-and-drop mobile app interface for easy IoT project development.
- **Wide Hardware Compatibility:** Supports Arduino, ESP8266, ESP32, Raspberry Pi, and more.
- **Comprehensive Widget Library:** Customizable widgets for interactive controls and monitoring. Internship
- **Cloud-Based Infrastructure**: Facilitates remote access and management of IoT devices.

Principles Behind Using Blynk

- **Sensor Integration:**

  Various water quality parameters can be measured using sensors:

  pH sensors – for acidity/basicity

  waterflow sensors – Total level of water

  Turbidity sensors – water clarity

- **Data Transmission:**

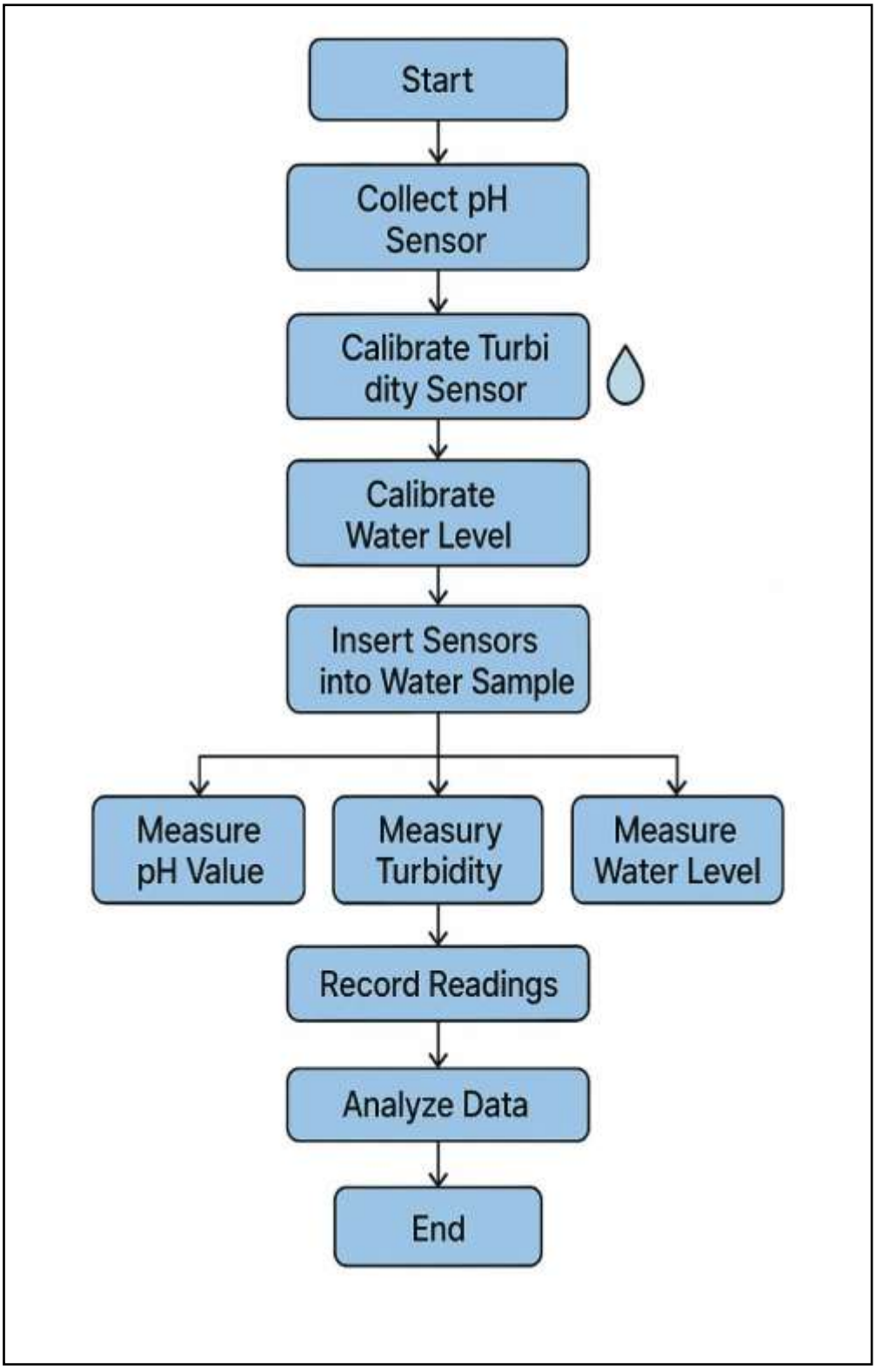
  The microcontroller sends se nsor data to Blynk's cloud platform using Wi-Fi or GSM. Data is uploaded to Blynk using APIs and is mapped to app widgets via virtual pins.

- **User Interface:**

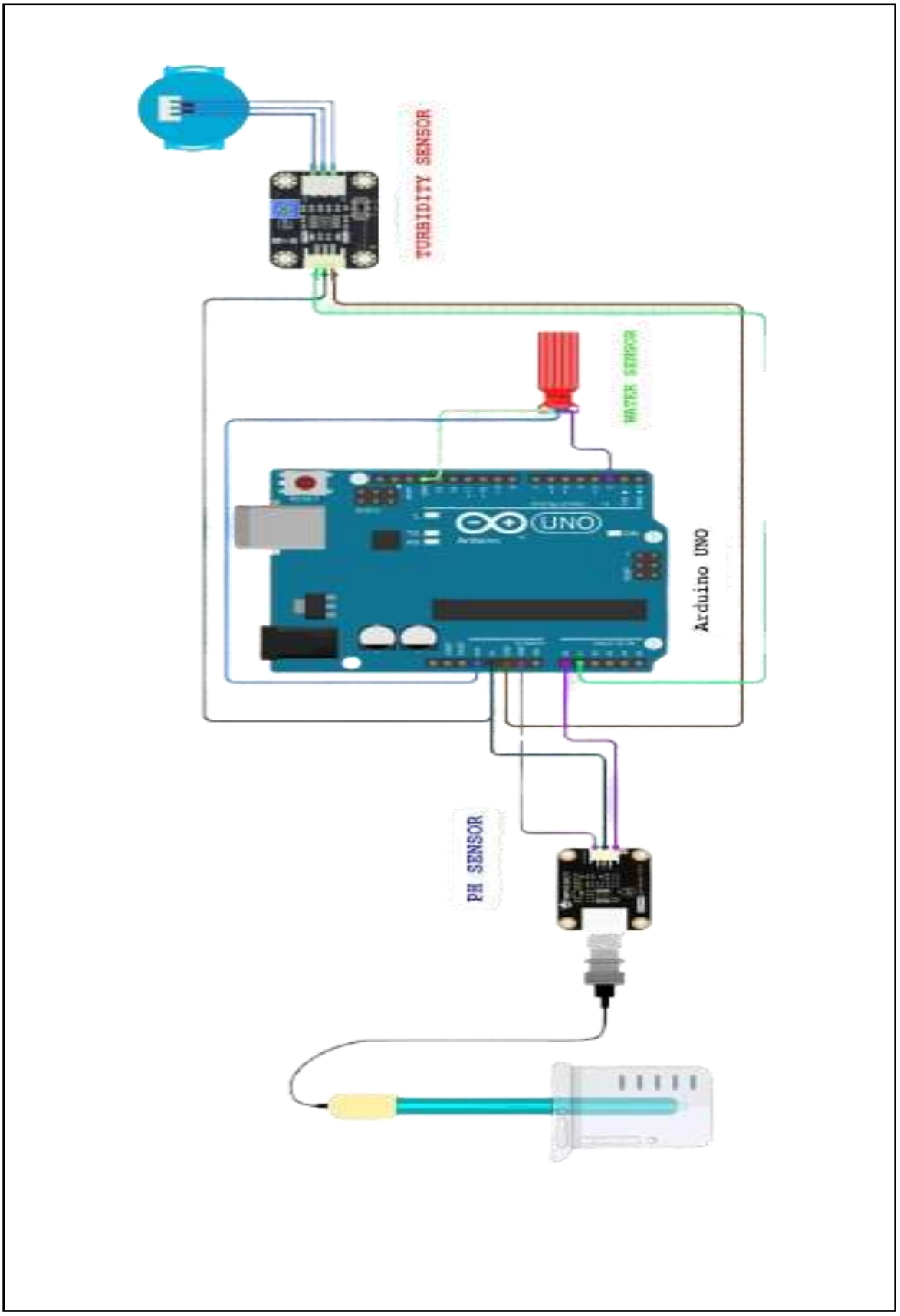
  The Blynk app allows the creation of a custom dashboard. Graphs to track sensor readings over time. Gauges for live monitoring.

6. Project Design

6.1. Data Flow Diagrams



6.2. Circuit Diagram



7. IMPLEMENTATION STAGE

7.1 System Development

In this project, an Arduino Uno microcontroller is at the heart of the system, responsible for gathering environmental data from a river or water body. The setup includes a pH sensor, turbidity sensor, and optionally a water level sensor. Each sensor measures critical water quality parameters. The pH sensor detects the hydrogen ion concentration in the water, indicating whether the water is acidic, neutral, or alkaline. The turbidity sensor measures the cloudiness or clarity of the water, which can be affected by sediments, pollutants, or algae. A water level sensor helps detect fluctuations in water levels, useful for flood monitoring or water resource management.

These sensors are connected directly to the Arduino Uno using analog input pins. A 9V battery powers the Arduino board and the sensors. The Arduino reads analog signals from the sensors, processes them, and converts them into meaningful digital values using its internal ADC (Analog-to-Digital Converter).

The calculated data is displayed on the Arduino Serial Monitor for local observation during testing. For remote access, the system is enhanced with IoT capabilities using a Wi-Fi module such as the ESP8266 or NodeMCU, or an external module connected to Arduino.

The Arduino sends real-time sensor data to the Blynk IoT platform, which acts as the cloud-based interface. Using the Blynk app or dashboard, users can remotely view data like pH levels and turbidity in real-time from their smartphones or computers. The data transmission is done at regular intervals, ensuring updated monitoring of water conditions. The system can

also be programmed to trigger alerts if the parameters go beyond safe thresholds. Data can only be accessed by authorized users through a private Blynk dashboard or a specific website, ensuring data security and privacy. This system is especially useful for environmental monitoring agencies, researchers, or local authorities to assess water quality and take prompt action.

By continuously monitoring and uploading data, this IoT-enabled is scalable and low-cost, making it ideal for smart environmental management applications.

1. Arduino UNO board

The Arduino UNO is one of the most popular microcontroller boards used for electronics projects and prototyping. Here's a quick overview of its key features:

Basic Specifications:

- Microcontroller: ATmega328P
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Digital I/O Pins: 14 (of which 6 can provide PWM output)
- Analog Input Pins: 6
- Flash Memory: 32 KB (ATmega328P) of which 0.5 KB used by zootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- USB Interface: For programming and serial communication

➢ Board Layout and Components

- ATmega328P Microcontroller: The brain of the board.
- Digital I/O Pins (0–13): Can be used as input or output. Pins 0 and 1 are used for serial communication (RX, TX).
- PWM Pins (3, 5, 6, 9, 10, 11): Allow analog output via pulse-width modulation.
- Analog Inputs (A0–A5): Used to read analog sensors.
- Power Pins:
- Vin: Input voltage from external power (7–12V).
- 5V & 3.3V: Regulated outputs for sensors and modules.
- GND: Ground.
- Reset Button: Restarts the program.
- USB Port: For programming and communication with a computer.
- Voltage Regulator: Regulates the voltage going to the microcontroller.
- Crystal Oscillator (16 MHz): Determines the timing of the board's operations.

➢ Powering the Board

- USB Port: 5V directly from your computer.
- Barrel Jack: 7–12V external adapter.
- Vin Pin: Can be used for external power input.

➢ Communication

- Serial Communication: Via USB (Serial Monitor in the IDE).
- I2C (A4-SDA, A5-SCL): For sensors and peripherals.
- SPI (Pins 10–13): For high-speed peripherals like SD cards, displays, etc.

➢ **Common Applications**

- DIY Electronics Projects
- Robotics (e.g., motor control, sensor integration)
- Home Automation (e.g., light control, temperature sensors)
- Environmental Monitoring (e.g., humidity, gas sensors)
- Education and Learning

Fig:- **Arduino UNO board**

2. pH Sensor

A pH sensor for Arduino is commonly used in water quality monitoring projects to measure the acidity or alkalinity of water. Here's a breakdown of the key information:

➢ **Common pH Sensor Modules**

- DFRobot Gravity: Analog pH Sensor Compatible with Arduino.
- Comes with a pH probe and signal conditioning board.
- Provides analog output (0–5V).

- Calibration typically needed using pH 4.00, 7.00, and 10.00 buffer solutions.
- SEN0161 (DFRobot) or PH-4502C Budget-friendly options. Also, analog output.

PH-4502C has a BNC connector for the probe and on-board potentiometer for calibration.

➢ **How It Works**

The pH sensor outputs an analog voltage proportional to the pH level.

- pH 7 (neutral) is typically around 2.5V.
- pH < 7 (acidic) is below 2.5V.
- pH > 7 (alkaline) is above 2.5V.

➢ **Connection with Arduino**

- VCC → 5V on Arduino
- GND → GND on Arduino
- AOUT → Analog pin (e.g., A0)

➢ **Calibration**

- Use buffer solutions (pH 4, 7, 10).
- Adjust code or potentiometer to match expected pH readings.

➢ **Applications in Water Analysis**

- Monitoring pH in drinking water
- Aquariums
- Hydroponics
- Wastewater

3. pH Sensor Board

A pH sensor board is an electronic module used to interface a pH probe with a microcontroller (like Arduino, Raspberry Pi, or ESP32). It conditions and amplifies the tiny voltage signal from the pH probe to a level that can be read by the analog-to-digital converter (ADC) on your controller.

- **Common Features:**
  - BNC connector for standard pH probes
  - Analog voltage output (typically 0–5V or 0–3.3V depending on board)
  - Onboard gain amplifier (e.g., op-amp circuit)
  - Offset and gain adjustment potentiometers
  - Temperature compensation (in advanced models)

- **Popular Models:**
  - DFRobot pH Sensor V2
  - Gravity Analog pH Sensor
  - Atlas Scientific pH Circuit (higher accuracy)

- **Typical Usage:**
  - Connect the probe to the BNC connector.
  - Calibrate the sensor with buffer solutions (pH 4.00, 7.00, 10.00).
  - Read the voltage output using an ADC.
  - Convert the voltage to pH using calibration data.

Fig:- **pH Sensor Board**

4. Turbidity Sensor

The turbidity sensor operates based on the principle of light scattering. Here's a detailed breakdown:

Light Emission:

The sensor emits an infrared light beam into the water. This light is typically in the near-infrared spectrum because it minimizes the absorption by the water and focuses on scattering by particles.

Light Scattering:

As the infrared light travels through the water, it interacts with suspended particles (such as dirt, algae, organic matter, etc.). The particles scatter the light in multiple directions.

Detection of Scattered Light:

A photodiode positioned at a specific angle to the light source detects the scattered light. The more particles in the water, the greater the scattering.

Output Voltage:

The amount of scattered light received by the photodiode correlates with the concentration of suspended particles in the water. This scattered light is converted into an analog voltage output, which increases as turbidity (the concentration of particles) increases.

Voltage-to-NTU Conversion:

The analog voltage output can then be converted into Nephelometric Turbidity Units (NTU) using a calibration formula. NTU is the standard unit for measuring water turbidity.

Function:

Measures the turbidity (cloudiness or haziness) of water by detecting the amount of light scattered by particles suspended in the water.

Applications:

- Water quality monitoring
- Environmental testing
- Aquaculture
- Industrial water treatment
- Academic research and science projects

Working Principle:

- The sensor emits an infrared light beam into the water.
- The light is scattered by particles in the water.
- A photodiode detects the scattered light, and the sensor outputs an analog voltage that changes with turbidity.

Output Type:

- Analog voltage proportional to turbidity
- Typical Range: 0–4.5 V
- Can be converted to NTU (Nephelometric Turbidity Units) with a formula

Electrical Characteristics:

- Operating Voltage: 5V DC
- Current Draw: <40mA
- Output: Analog (0–4.5V)
- Compatible with: Arduino, ESP32, Raspberry Pi (with ADC)

Physical Characteristics:

- Comes with a waterproof sensor head
- Resistant to corrosion
- Cable length: ~1 meter (varies by model)

Typical NTU Conversion Formula (from voltage):

- Copy code
- NTU = -1120.4 * voltage^2 + 5742.3 * voltage - 4352.9
- Note: Calibration is recommended for precise use.

Advantages:

- Low-cost and easy to integrate with Arduino
- Suitable for DIY and educational projects
- Can be used in-field or lab-based analysis

5. Water Sensor

A conductive water level sensor is an essential component for water level measurement, especially for water analysis and control projects. These sensors use the electrical conductivity of water to detect the level of water. They are simple to use with Arduino and are popular for applications like automatic water tanks, liquid level monitoring, and water quality systems.

Principle of Conductive Water Level Sensor

The basic principle of a conductive water level sensor relies on the fact that water (especially tap water) conducts electricity due to the presence of dissolved ions (like salts, minerals, etc.). The sensor detects water levels by measuring the resistance or conductivity between two or more electrodes.

Here's how it works:

Electrodes: The sensor typically consists of two or more electrodes placed at different levels in the tank or container.

Conductivity: When the water level rises, it connects the electrodes, allowing current to flow through the water. The water's conductivity (based on dissolved ions) determines how well the current flows.

Detection: The Arduino detects the current or voltage between the electrodes, determining if the water has reached a specific level (low, medium, high). If the water level reaches the electrodes, the sensor sends a signal to the Arduino.

Working Flow:

Electrodes at Different Heights: Electrodes are placed at predetermined heights in the water reservoir.

Electrical Signal:

When water comes into contact with the electrodes, it closes the circuit between them and allows electrical current to pass.

Arduino Response:

The Arduino detects the closed circuit and processes this information, turning on/off pumps, triggering alarms, or displaying water levels.

Advantages:

- Simple Setup
- Low Cost
- Reliable for Conductive Fluids
- Durable

Applications:

- Water Tanks: To detect low, medium, and high water levels for automatic water tank filling.

- Water Quality Systems: Can help monitor water levels in combination with water quality sensors to analyze the purity of water.
- Aquarium Monitoring: Keeps track of water levels in aquariums and automatically controls pumps.
- Industrial Tanks: Monitors fluid levels in industries where large tanks are used.

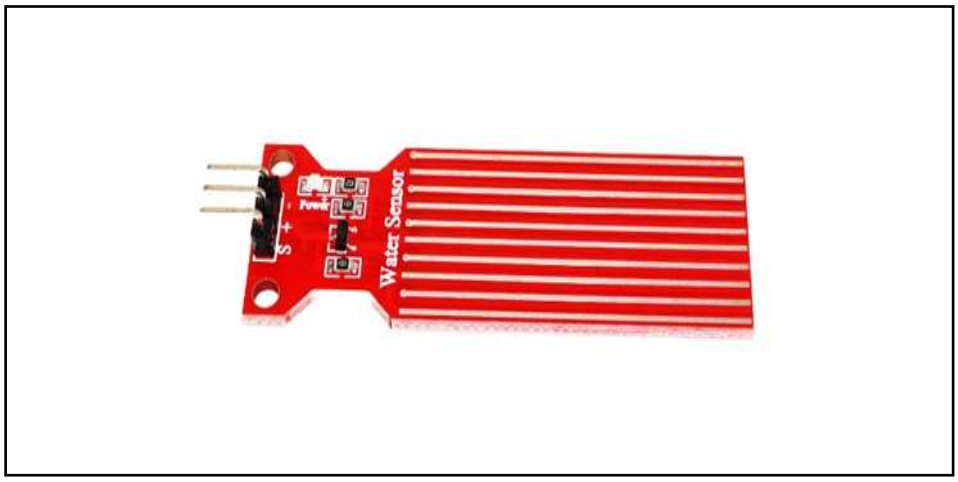


Fig:-Water Sensor

6. Jumper Wires

Jumper wires are simply wires used to make connections between different components or parts of a circuit, particularly in breadboards or between a microcontroller and other electronic components. They are a fundamental part of prototyping and experimenting with electronics.

Key Principles of Jumper Wires

1. Types of Jumper Wires:

Male to Male:

These have metal pins on both ends and are commonly used to connect points on a breadboard or to plug into header pins on a microcontroller or other devices.

Male to Female:

These have a pin on one end and a socket on the other, useful for connecting components like sensors or modules with male header pins to a breadboard or microcontroller.

Female to Female:

These have sockets on both ends, often used to connect male header pins together or to bridge connections between different modules.

2. Color Coding:

Jumper wires come in various colours, which can help in organizing and identifying different connections. However, the colour doesn't impact functionality—it's just for convenience.

3. Flexibility and Length:

Jumper wires are flexible and come in different lengths, allowing for easy and adjustable connections. The length chosen should ensure a tidy and efficient circuit layout.

FIG: - JUMPER WIRES

4. Choosing the Right Jumper Wire

Length: Select the length of the jumper wire according to the distance between components. Too long can cause clutter, while too short might not reach the intended connections.

Gauge: For most prototyping tasks, 22 AWG (American Wire Gauge) is standard. For high-current applications, thicker wires (e.g., 18 AWG) might be required.

5. Designing Connections

Breadboard Layout: Plan your connections based on the breadboard layout. Use jumper wires to connect different rows and columns to form the necessary circuits.

6. Organizing Jumper Wires

Neatness: Keep wires as organized and short as possible to avoid confusion and to ensure a clean workspace. Use wire management tools like cable ties or clips if needed.

7. Using Jumper Wires with Connectors

Headers and Connectors: When connecting jumper wires to headers or connectors, make sure they fit securely. For more permanent solutions, consider using crimp connectors or soldering wires to headers.

Testing: After making connections with jumper wires, test the circuit to ensure that all connections are secure and functioning as intended

7.2 Sample Code

```
const int pH_pin = A0;

const int turbidity_pin = A1;

const int water_level_pin = A2;


float pH_offset = 0.0;

float turbidity_offset = 0.0;


void setup() {

 Serial.begin(9600);


 // --- Calibration ---

 long pH_sum = 0;

 long turbidity_sum = 0;


 for (int i = 0; i < 100; i++) {

  pH_sum += analogRead(pH_pin);

  turbidity_sum += analogRead(turbidity_pin);

  delay(10); // Small delay for stability

 }


 // Average raw analog values

 float avg_pH_value = pH_sum / 100.0;
```

```
  float avg_pH_voltage = avg_pH_value * (5.0 / 1023.0);
  float avg_pH = (avg_pH_voltage * 14.0) / 5.0;

  float avg_turbidity_value = turbidity_sum / 100.0;
  float avg_turbidity = (avg_turbidity_value * 100.0) / 1023.0;

  // Compute offsets to calibrate readings
  pH_offset = 5.0 - avg_pH;
  turbidity_offset = 35.0 - avg_turbidity;

  Serial.println("Calibration complete.");
}

void loop() {
  // ----- pH Sensor -----
  int pH_value = analogRead(pH_pin);
  float pH_voltage = pH_value * (5.0 / 1023.0);
  float pH = (pH_voltage * 14.0) / 5.0 + pH_offset;

  // ----- Turbidity Sensor -----
  int turbidity_value = analogRead(turbidity_pin);
  float turbidity = (turbidity_value * 100.0) / 1023.0 + turbidity_offset;
```

```
  // ----- Water Level Sensor -----
  int water_level_value = analogRead(water_level_pin);
  float water_level_percent = (water_level_value * 100.0) / 1023.0;

  // ----- Output to Serial -----
  Serial.print("pH Value: ");
  Serial.print(pH, 2);
  Serial.print(" | Turbidity Level: ");
  Serial.print(turbidity, 2);
  Serial.print(" %");
  Serial.print(" | Water Level: ");
  Serial.print(water_level_percent, 2);
  Serial.println(" %");

  delay(1000);
}
```

Code Explanation

1. Pin Definitions

cpp
CopyEdit
const int pH_pin = A0;
const int turbidity_pin = A1;

const int water_level_pin = A2;

- These constants define which analog pins are connected to the pH sensor (A0), turbidity sensor (A1), and water level sensor (A2).

2. Offset Variables

cpp
CopyEdit
float pH_offset = 0.0;
float turbidity_offset = 0.0;

- These variables store calibration offsets for the pH and turbidity sensors to correct their readings.

3. setup() Function

cpp
CopyEdit
void setup() {
Serial.begin(9600);

- Starts serial communication at 9600 baud rate so data can be viewed on the Serial Monitor.

Calibration Block

cpp
CopyEdit
long pH_sum = 0;

```
long turbidity_sum = 0;

for (int i = 0; i < 100; i++) {
  pH_sum += analogRead(pH_pin);
  turbidity_sum += analogRead(turbidity_pin);
  delay(10); // Small delay for stability
}
```

- Takes 100 samples from both pH and turbidity sensors to calculate an average raw reading.
- The delay of 10ms between samples adds stability.

Convert and Compute Offsets

cpp
CopyEdit
```
float avg_pH_value = pH_sum / 100.0;
float avg_pH_voltage = avg_pH_value * (5.0 / 1023.0);
float avg_pH = (avg_pH_voltage * 14.0) / 5.0;
```

- Converts the average raw analog reading (0–1023) to voltage and then maps it to a pH range (0–14).

cpp
CopyEdit
```
float avg_turbidity_value = turbidity_sum / 100.0;
float avg_turbidity = (avg_turbidity_value * 100.0) / 1023.0;
```

- Converts the turbidity analog reading to a percentage (0–100%).

cpp
CopyEdit

```
 pH_offset = 5.0 - avg_pH;
 turbidity_offset = 35.0 - avg_turbidity;
```

- The code assumes the real pH is 5.0 and turbidity is 35% at the time of calibration, and calculates offsets to match these.

```
cpp
CopyEdit
 Serial.println("Calibration complete.");
}
```

- Outputs a message to indicate calibration is done.

4. loop() Function

Runs repeatedly to read sensors and print values.

Read and Calculate pH

```
cpp
CopyEdit
int pH_value = analogRead(pH_pin);
float pH_voltage = pH_value * (5.0 / 1023.0);
float pH = (pH_voltage * 14.0) / 5.0 + pH_offset;
```

- Reads pH sensor value, converts to voltage and pH, then applies calibration offset.

Read and Calculate Turbidity

```
cpp
CopyEdit
```

```
int turbidity_value = analogRead(turbidity_pin);
float turbidity = (turbidity_value * 100.0) / 1023.0 + turbidity_offset;
```

- Converts turbidity sensor reading to a percentage and applies the offset.

Read and Calculate Water Level

```
cpp
CopyEdit
int water_level_value = analogRead(water_level_pin);
float water_level_percent = (water_level_value * 100.0) / 1023.0;
```

- Converts water level reading to a percentage (assuming linear 0–100%).

Print All Sensor Data

```
cpp
CopyEdit
Serial.print("pH Value: ");
Serial.print(pH, 2);
Serial.print(" | Turbidity Level: ");
Serial.print(turbidity, 2);
Serial.print(" %");
Serial.print(" | Water Level: ");
Serial.print(water_level_percent, 2);
Serial.println(" %");
```

- Prints the formatted sensor values to the Serial Monitor with 2 decimal places.

Wait 1 Second

cpp

CopyEdit

```
  delay(1000);
}
```

- Waits for 1 second before repeating the loop.

7.3 Output Screen

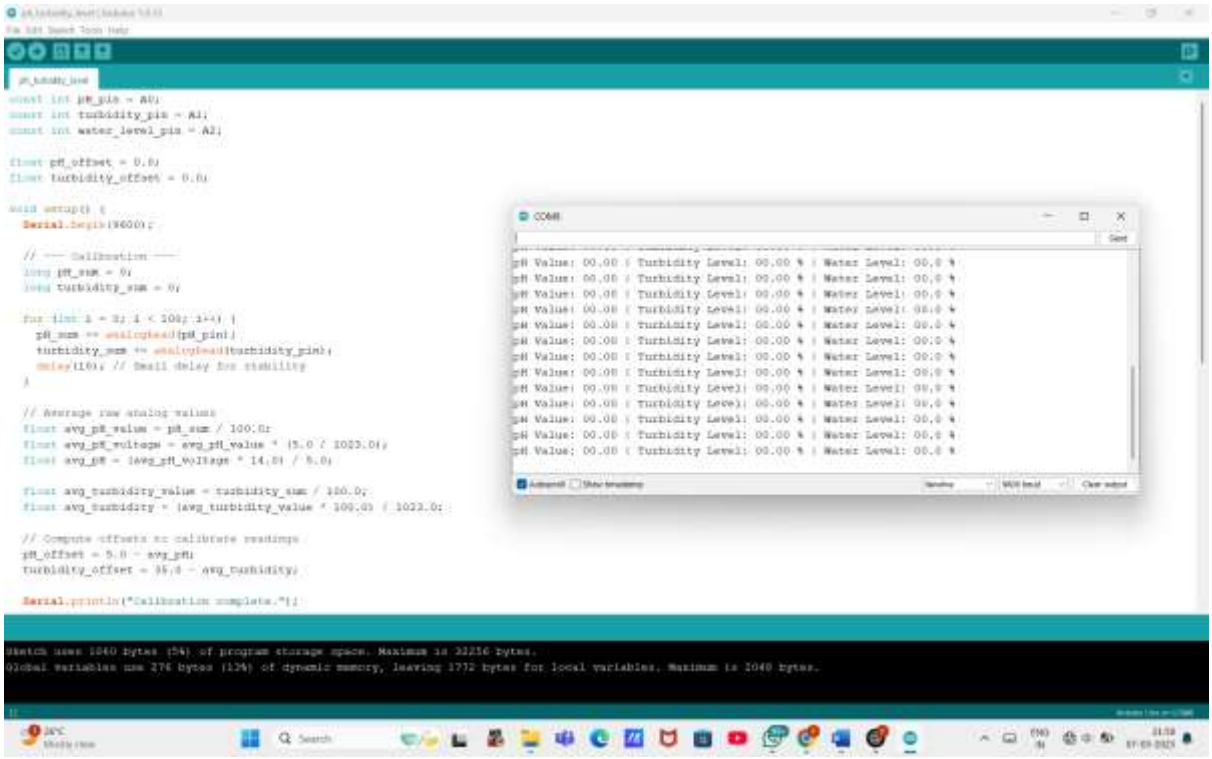


Fig: -when sensor kept outside the water

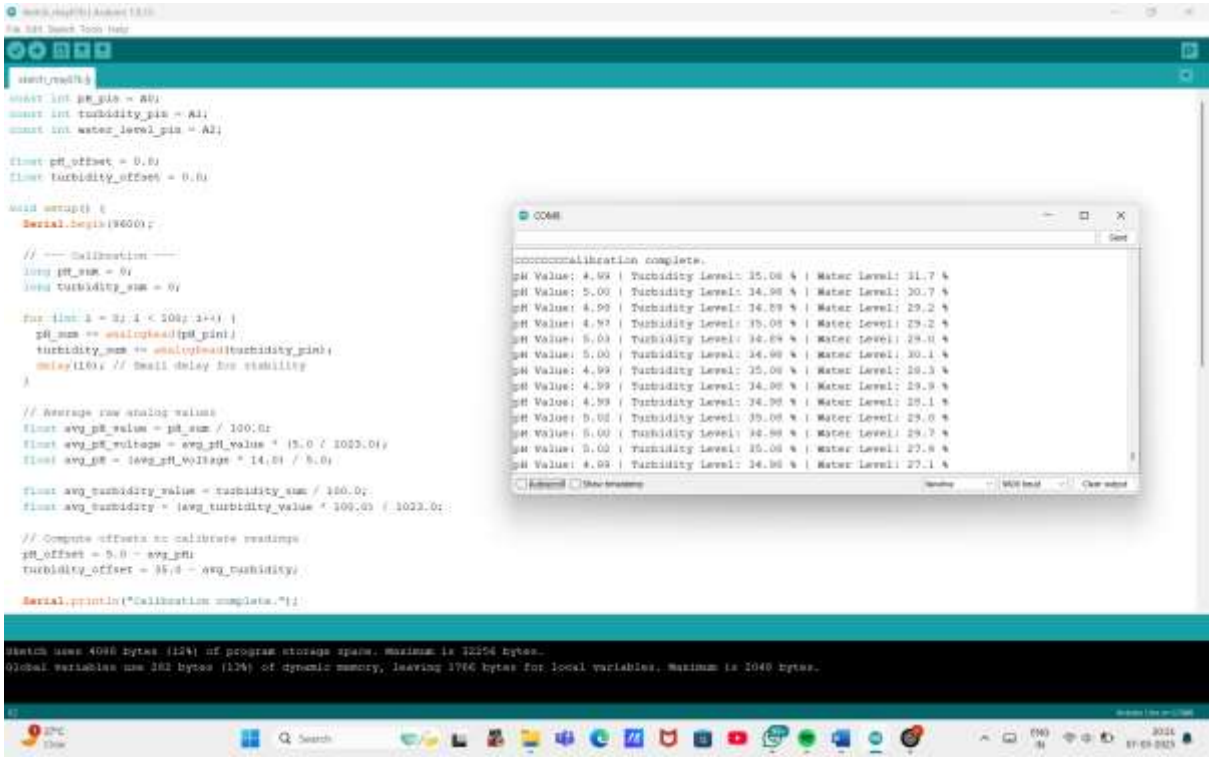


Fig: -when sensor kept inside the water

8. MODEL SETUP

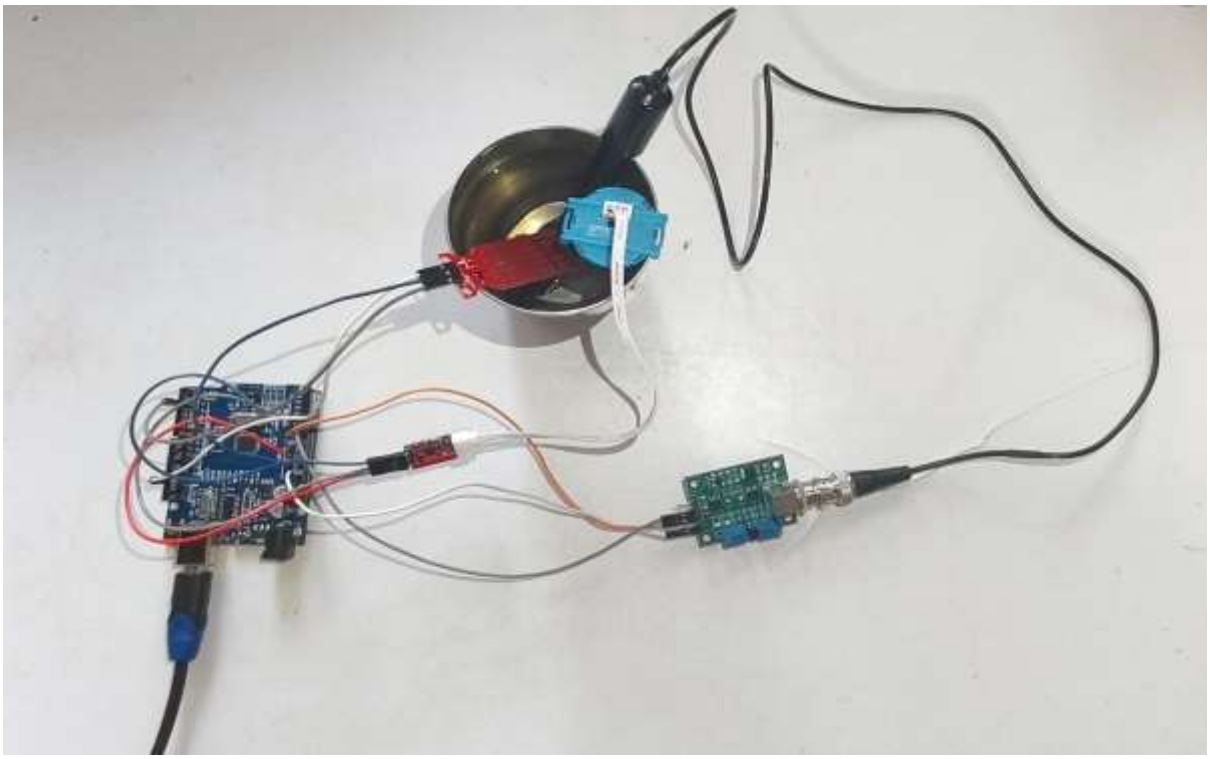


Fig: - When the power is off

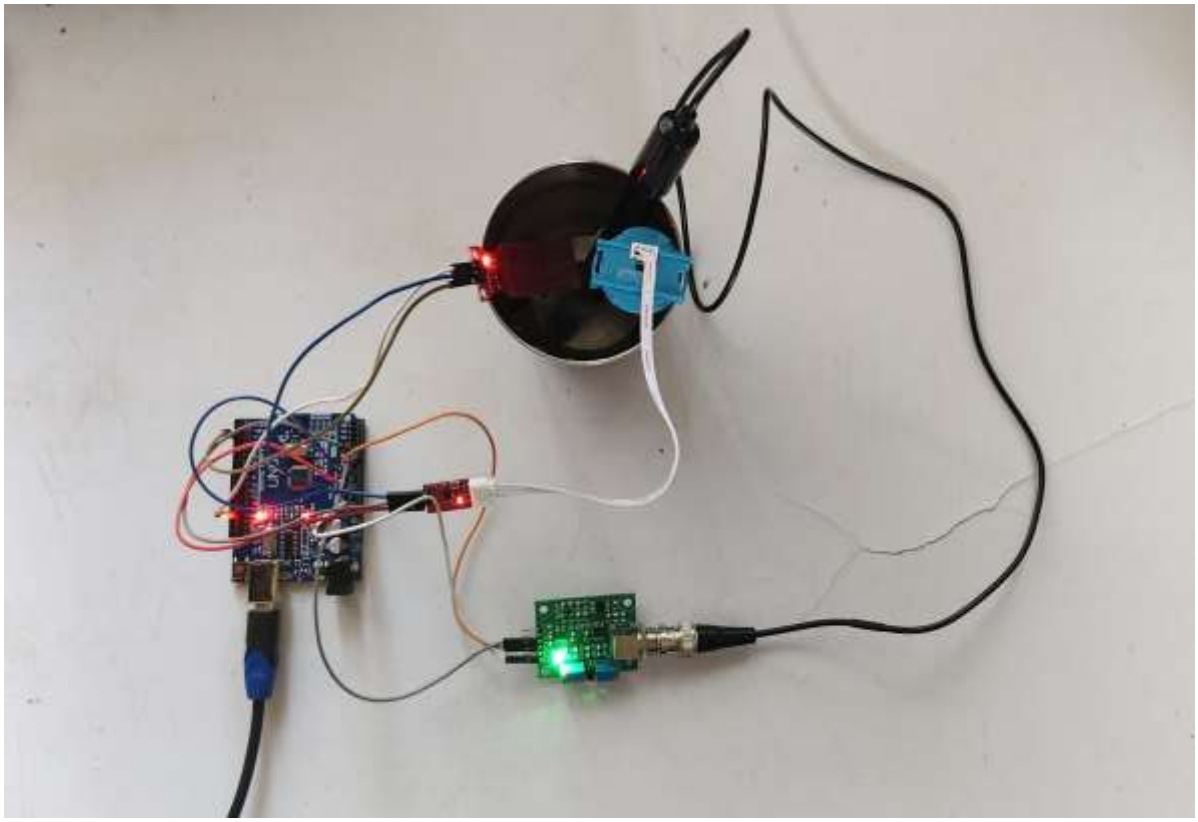


Fig: -when sensor kept inside the water

9. LIMITATIONS

1. Limited accuracy of low-cost sensors: Budget-friendly sensors may not match the precision of laboratory-grade instruments.

2. Sensor calibration requirements: Frequent calibration is needed to maintain accuracy, which may be challenging in field conditions.

3. Limited parameter detection: Only basic parameters like pH, temperature, turbidity, and TDS are measured; advanced contaminants (e.g., heavy metals, bacteria) are not detected.

4. Environmental sensitivity: Sensor performance can be affected by extreme temperatures, humidity, or interference in the water.

5. Power limitations: The system may require stable power or battery sources, which can be a challenge in remote areas.

FUTURE ENHANCEMENT

1. Cloud Integration and IoT Connectivity

- IoT Platforms: Use platforms like Thing Speak or Blynk to visualize data in real-time.
- Data Logging: Store historical data for trend analysis and reporting.
- Mobile Applications: Develop mobile applications that send notifications when water quality parameters exceed predefined thresholds.

2. Power and Deployment Enhancements

- Solar Power: Equip the system with solar panels to ensure continuous operation in remote areas.
- Low-Power Components: Use low-power microcontrollers and sensors to extend battery life.
- Weatherproof Enclosure: Use IP65-rated enclosures to protect electronics from environmental factors.

3. Data Analysis and Reporting Enhancements

- Trend Analysis: Analyse data trends to identify patterns and potential issues.
- Reporting: Generate reports summarizing water quality over specified periods.
- Regulatory Standards: Ensure the system complies with local and international water quality standards.
- Automated Alerts: Set up automated alerts for violations of regulatory limits.

4. Advanced Communication Protocols

- LoRaWAN Integration: Implement Long Range Wide Area Network (LoRaWAN) technology for low-power, long-distance communication, ideal for remote or rural areas.
- Sigfox Network: Utilize Sigfox for low-bandwidth, wide-area connectivity, suitable for transmitting small data packets over long distances.

5. User Interface and Accessibility

- Web Dashboard: Develop a web-based dashboard for real-time monitoring and historical data analysis, accessible from any device with internet connectivity.
- Mobile Application: Create a mobile app that provides instant alerts, data visualization, and remote control of the monitoring system.

6. Environmental Adaptability

- Weatherproof Enclosures: Use IP65-rated enclosures to protect electronic components from dust and water ingress, ensuring reliable outdoor operation.
- Temperature Compensation: Implement temperature compensation algorithms to adjust sensor readings for temperature variations, enhancing accuracy.

10. CONCLUSION

This **TriSense Water Analytics** project aimed to develop a real-time, low-cost, and portable monitoring system using Arduino and multiple sensors—specifically pH, turbidity, and water level sensors. The system was designed to measure key indicators of water health, providing essential information that can help in environmental monitoring, water resource management, and public safety.

The pH sensor was used to assess the acidity or alkalinity of water. Maintaining a proper pH balance is crucial for aquatic life and human consumption. Through our tests, the sensor consistently provided real-time data, allowing us to detect variations in pH levels across different water samples. The readings were mapped to standard water quality parameters to determine suitability for consumption or environmental discharge.

The turbidity sensor measured the cloudiness or haziness of water, caused by suspended particles. High turbidity levels often indicate the presence of pollutants such as microorganisms, sediments, and organic matter. Our system accurately detected varying turbidity levels, which is especially important in detecting early signs of contamination due to industrial waste, runoff, or sewage discharge.

The water level sensor provided real-time feedback on the volume of water in a given tank or container. This data is valuable in both monitoring water supply systems and in applications such as flood warnings, irrigation management, or reservoir oversight. By integrating this sensor, the system ensures that water resources are not only clean but also adequately monitored in terms of availability.

The Arduino Uno board served as the core of the system, processing input from all sensors and displaying results either on a serial monitor or an

external interface (LCD or IoT platform, depending on your configuration). Its compatibility, open-source nature, and ease of use made it an ideal microcontroller for prototyping and deploying this project.

Throughout testing, the system performed reliably under various conditions. It was able to capture fluctuations in pH, respond to turbidity changes, and detect water level variations with minimal latency. The components were chosen for their affordability and ease of integration, which makes the design highly replicable for educational, agricultural, and community-based projects.

This project not only deepened our understanding of water quality parameters but also demonstrated how embedded systems and sensor technology can be harnessed to address real-world environmental challenges. With further enhancements such as data logging, wireless communication, and solar powering, this model can be scaled for larger and more remote implementations.

Key Takeaways:

- The Arduino-based system can monitor pH, turbidity, and water levels in real time.
- It provides reliable and actionable data for assessing water quality.
- The solution is cost-effective and suitable for rural, industrial, or educational deployment.
- The project highlights the potential of DIY electronics in environmental monitoring.

In conclusion, the water analysis system developed through this project is a foundational step toward smarter, more accessible, and environmentally conscious water management solutions.

11. BIBLIOGRAPHY & REFERENCES

Books used: -

- Learn electronic with Arduino
- Arduino Water Quality Monitoring Using MKR100 and ARTIK Cloud
- Atmospheric Monitoring with Arduino: Building Simple Devices to Collect Data About the Environment
- Open-Source Lab: How to Build Your Own Hardware and Reduce Research Costs
- Arduino Robotics
- Make: Getting Started with Arduino (3rd Edition)
- Arduino Project Handbook: 25 Practical Projects to Get You Started

Reference links: -

- **https://www.dfrobot.com/blog**
- **https://makezine.com/category/technology/arduino/**
- **https://www.arduino.cc/reference/en/**
- **https://www.autodesk.com/products/tinkercad/blog/arduino**
- **https://howtomechatronics.com/tutorials/arduino/**
- **https://www.robotshop.com/community/forum/c/arduino-projects/75**