Author 1: Mansour Jatta	**(210032203)**
Author 2: Mustapha Jammeh	**(210032207)**
Author 3: Famara fofana	**(210032202)**
Author 4: Modou lamin Jabang	**(210032201)**
Author 5: Ramise Ndongo	**(210032208)**

Department of technical and vocational education

Islamic University of Technology (IUT)

Gazipur, Bangladesh

PROJECT TITLE

AUTOMATIC WATER LEVEL CONTROLLER For Both Overhead and Underground Tanks Using Arduino Uno software

Approved by:

Dr. Dr. Syed Iftekhar Ali

Supervisor and Professor,

Department of Electrical and Electronic Engineering,
Islamic University of Technology (IUT),
Board bazar, Gazipur-1704.

Date: ……

Acknowledgments

The realization of "Construction of Automatic Water Level Controller for Both Overhead and Underground Tanks" has been made possible through the collaboration and support of several individuals. While their names remain unmentioned, their contributions have been instrumental in bringing this project to fruition.

I extend my sincere appreciation to our supervisor, Dr. Iftekhar, whose guidance and expertise were invaluable in shaping the direction of this project. Dr. Iftekhar's support has been a cornerstone in navigating the complexities of research and development.

To all those who contributed, your impact has not gone unnoticed, and this work stands as a testament to our collective dedication to advancing knowledge in the field.

Thank you for your valuable contributions

ABSTRACT

Controlling water tanks is essential for responsible water management, property protection, and environmental sustainability. Controlling water from overflowing prevents from the followings; water conservation, property damage, flood prevention, mould and mildew, utility costs, environmental impact and legal compliance.

ABSTRACT controlling water tanks is essential for responsible water management, property protection, and environmental sustainability. Controlling water from overflowing prevents from the followings; water conservation, property damage, flood prevention, mould and mildew, utility costs, environmental impact and legal compliance. The research project is geared toward controlling the water level through automatic or manual by monitoring, controlling and switching the tanks level at high and low levels. Automation is crucial in today's fast paced, technology-driven world to stay competitive, efficient, and adaptable to changing market demands. Automation is important for both domestic and industries for the followings; efficiency, cost reduction, scalability, consistency, quality improvement, safety, data analysis, competitive advantage, customer satisfaction and innovation.

When the level of the tank is at low or high level, the sensor sent a signal to the programmable logic controller (PLC). The PLC then send command signal to the pump for its triggering through the relay. The sensors detect the levels of the water in both underground and overhead tanks. When the water level falls below low level threshold (as detected by the sensors), the pump is activated, this set to be **ON** and while the pump is running, it continuously monitors the water level, when the water level rises to a high level (as detected by the sensors), the pump automatically turned off to prevent overfilling or wasting energy and the pump is set to be **OFF**.

List of figures

Table of Contents

CHAPTER 1

INTRODUCTION

1.1. The goal of automatic system control

Is to minimize human involvement, maximize resource utilization, and improve system safety, dependability, and performance. The automated pumping system built into the system is monitored, controlled, and switched by an automation system. Water levels are Arduino Uno-based, which guarantees effective management of water resources and lowers the possibility of shortages or overflow. Based on the logic and sensor inputs, the Arduino Uno controls the water pump by sending signals to it. When it falls or rises above the predetermined low/high level, a signal to start or stop is sent. Implementation of alerts or notifications for anomalous circumstances, including overflow, malfunctioning of a pump, or failure of a sensor. These alerts might be recorded for further examination or forwarded to operators. The program has features for implementation safety, such as emergency stop conditions and overflow protection. They guarantee the system functions safely and guard against harm or mishaps. The Arduino Uno sends a signal to turn on the pump so that the tank is filled when the water level drops below a certain point low level (LL). The pump is signalled to stop by the Arduino Uno when it reaches a new high level (HL). Using the sensors, the Arduino Uno continuously measures the water level and compares it to the predetermined levels. The pump control is adjusted if the actual level deviates from the desired range.

1.1.1. Aim and Objectives

Objective: The main objective of this project is to effectively monitor and regulate the water levels in the overhead tank and the subterranean tank. The goal is to avoid shortages or overflow and to guarantee a steady and reliable supply of water.

Goals:

The following are the project's specific goals:

Water Conservation: To efficiently preserve water resources.

Constant Supply: To keep the water supply steady and unbroken.

Energy Efficiency: To maximise the water supply system's use of energy.

Decreased Manual Intervention: To reduce the amount of time spent on manual oversight and management.

Notifications and Alerts: To deliver timely notifications and alerts on the status of the system.

Data logging: To capture and preserve pertinent data for examination.

Cost Savings: To lower the water supply's operational expenses.

Impact on the Environment: To make a beneficial contribution to the environment.

Reliability: To guarantee the water supply's dependability and consistency

Safety: To enhance the safety of the water supply system.

Achieving these objectives will result in an automatic water tank level control system that contributes to efficient and sustainable water resource management.

1.1.1.1. Justification

Water conservation, energy efficiency, convenience, cost savings, and enhanced equipment protection are just a few benefits that come with automatic water tank level control systems. They also reduce the possibility of human error by eliminating the need for manual control and can react quickly to leaks or spikes in water demand, preventing damage. Lastly, by running the pump only when necessary, these systems save energy and prolong the life of plumbing systems by using less electricity.

1.1.1.1.1. Scope of the Project

A low-cost automatic water pump was used to achieve this goal, and the automatic water level controller detects and manages water levels in the tank. The purpose of this project was to automate the control of the water pump, ensuring a constant reserve of water in the reservoir. The scope of the design was purposefully kept concise and straightforward to avoid unnecessary complexities that could make it less user-friendly. The system does not include complex peripheral devices to ensure affordability and maintain accuracy.

1.1.1.1.1.1. Constraints

The main challenges that we encountering in this course of this project includes difficulties in finding the project's design and sourcing the required materials and components. Items such as pumps and programming buffers were particularly challenging to locate.

1.1.1.1.1.1.1. **Limitations of the Project**

It is essential to note that this design is limited to controlling a 12V, 5A electric pump and cannot be used to control industrial water pumps exceeding 5 amps.

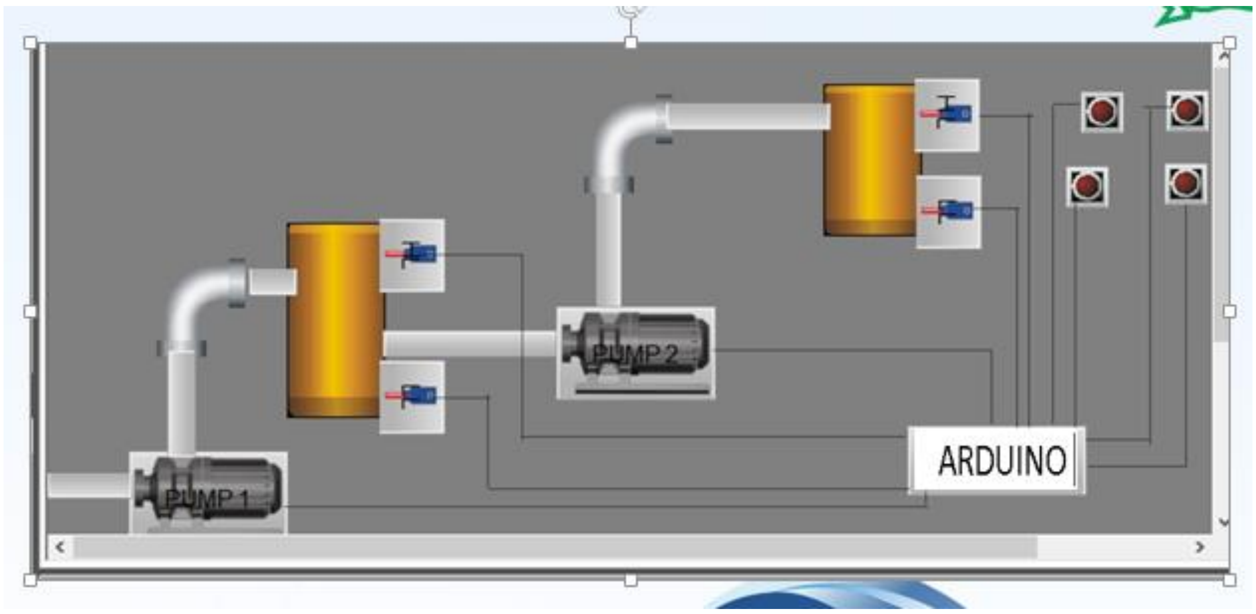


FIG. 1.1. SCHEMATIC DIAGRAM OF THE AUTOMATIC WATER LEVEL CONTROL SYSTEM

CHAPTER 2

METHODOLOGY

2.1. Introduction:

The methodology employed in this project aims to systematically design and implement an automatic water level controller for both overhead and underground tanks. Central to this approach is the utilization of the Arduino Uno microcontroller (see image below). Its versatility and seamless integration with sensor systems make it the ideal choice for this project.

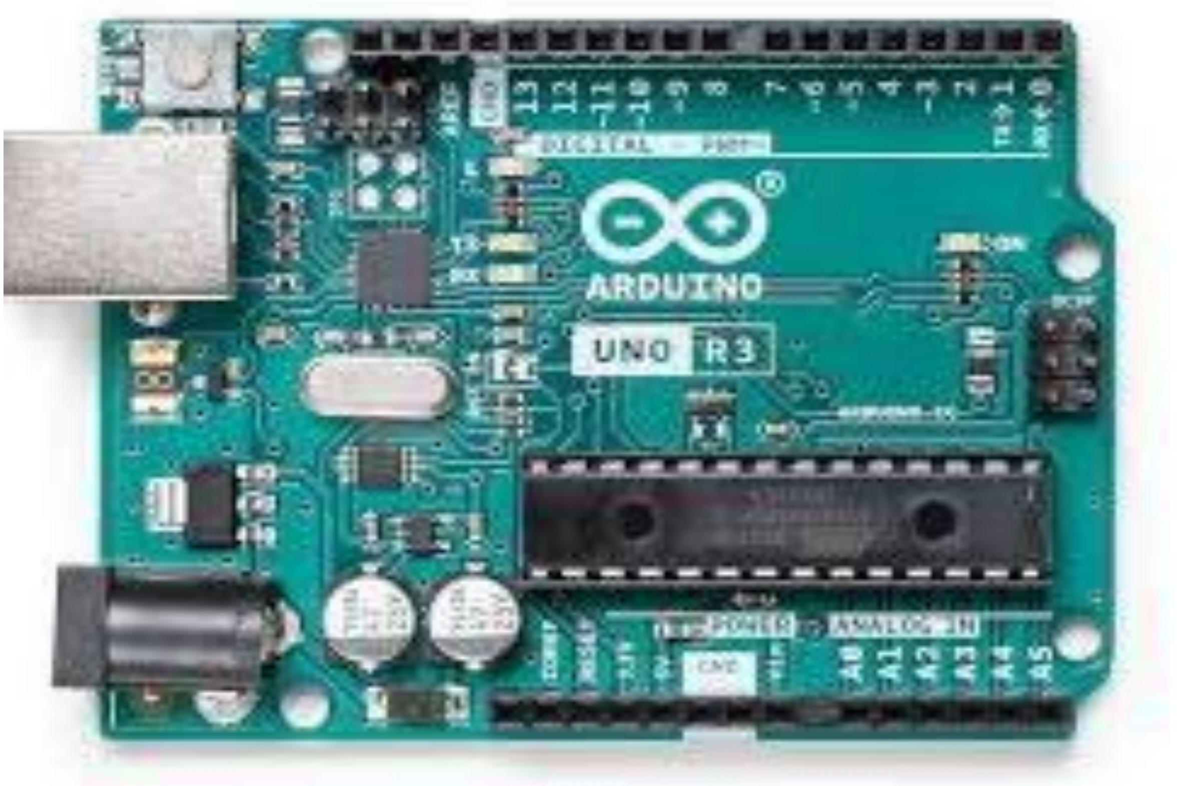


Fig 2.1 Arduino Uno microcontroller

2.1.1. Research Design:

A mixed-methods research design was selected, combining experimental trials with qualitative observations (see image below). The microcontroller plays a critical role in the experimental phase, facilitating real-time data processing and control mechanisms.

2.1.1.1. Participants or Materials:

The study involves participants with expertise in relevant fields, such as electrical engineering and water resource management. This ensures diverse perspectives on the construction process and valuable insights for system optimization.

Materials include:

- Water level sensors: Ultrasonic sensors for overhead tanks and pressure sensors for under-ground tanks (see images below). These sensors provide accurate measurements of water levels.

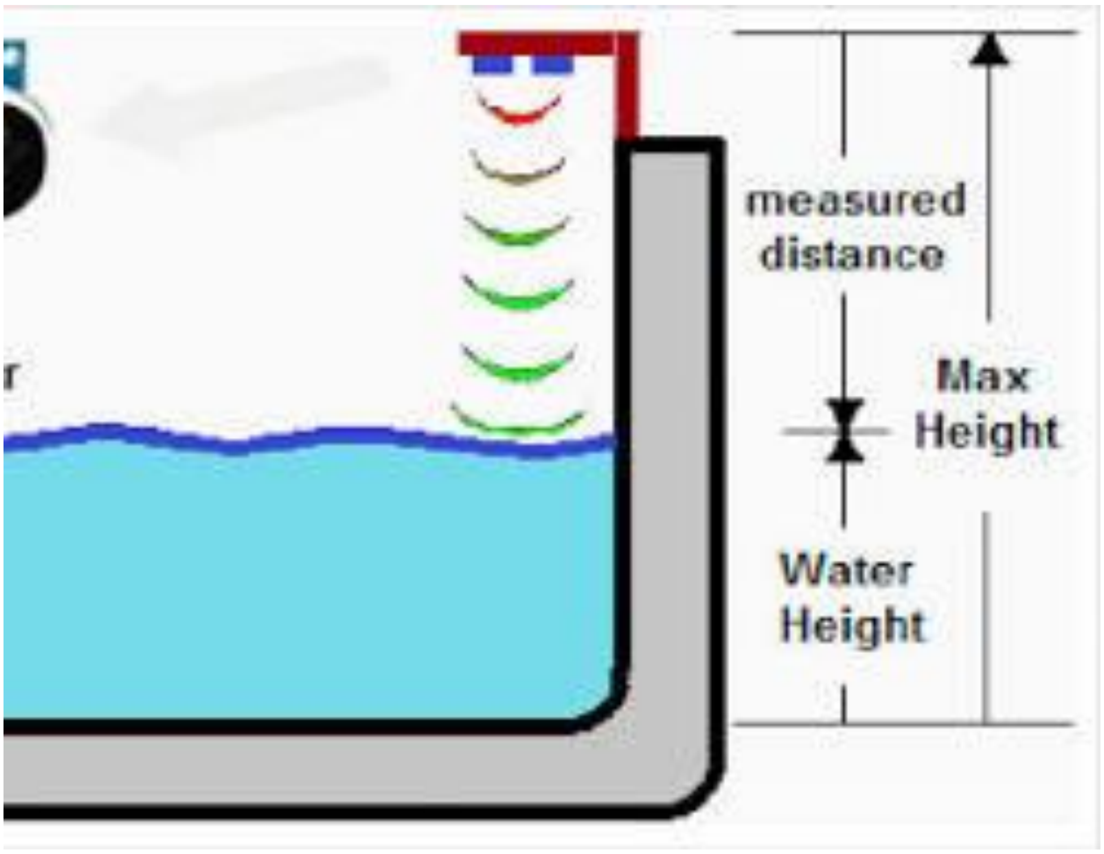


Fig 2.1. Ultrasonic sensor for water level Actuators: Relays to switch the pump on and off based on Arduino signals (see image

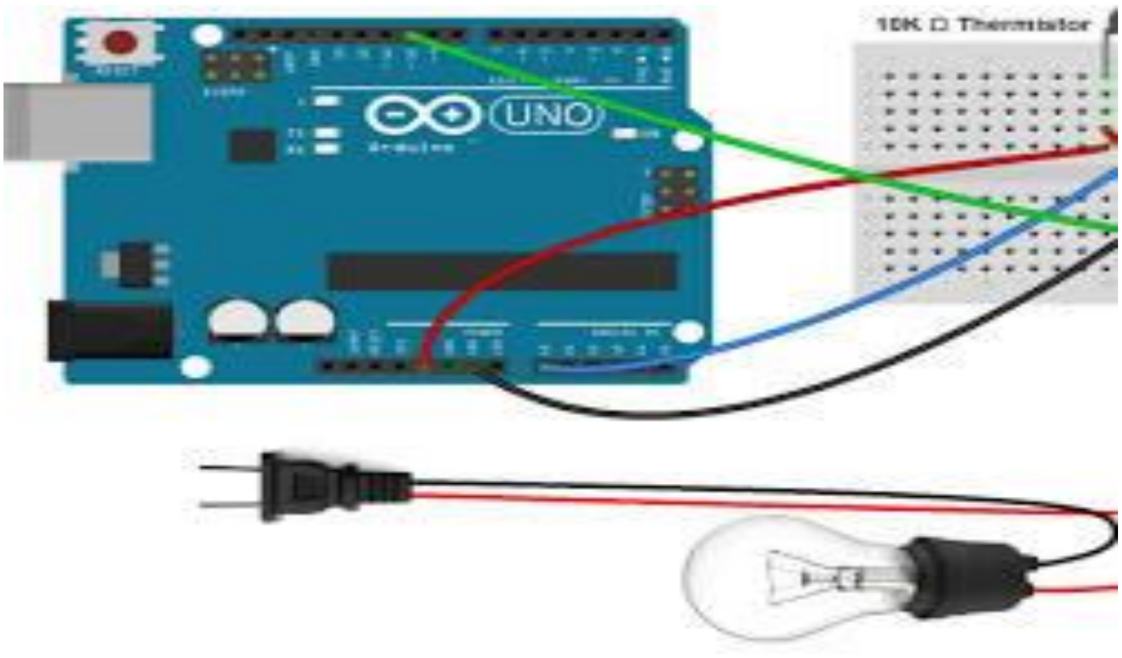


Fig 2.2 Relay for Arduino

- Water pump: Suitable for filling both tanks efficiently.
- Power supply: To power the microcontroller and relays.
- Display unit (optional): LCD or LED display to show water levels in real-time.
- Enclosure (optional): To protect electronics from moisture and dust.

2.1.1.1.1. Procedure:

1. Install water level sensors: Securely mount sensors in the appropriate locations within the tanks, ensuring proper signal transmission.
2. Connect components: Wire all components together following the schematic diagram, and ensure proper power connections.
3. Upload code to Arduino Uno: Implement the custom-designed program, using appropriate libraries for sensor interaction and pump control.
4. Test and calibrate: Verify code functionality and sensor accuracy using a simulated environment or controlled water source. Fine-tune control logic as needed based on testing results.
5. Install controller: Mount the controller unit in a readily accessible and protected location, considering moisture resistance and user interaction

2.1.1.1.1.1. Data Collection:

The Arduino Uno continuously records and processes sensor data in real-time. The microcontroller interfaces with water level sensors, capturing variations and transmitting signals for precise control of water levels. Specific sensor models, such as the HC-SR04 ultrasonic sensor and the MPX4250 pressure sensor, are compatible with Arduino Uno and ensure accurate data acquisition.

2.1.1.1.1.1.1. Data Analysis:

Quantitative data obtained from the Arduino Uno is analysed using statistical methods like descriptive statistics and inferential testing (see image below). This provides real-time insights into water level variations and system performance. Qualitative data from participant observations undergoes thematic analysis, offering a comprehensive understanding of user experiences and potential areas for improvement

2.1.1.1.1.1.1.1. Ethical Considerations:

Ethical approval was obtained from [relevant ethical review board], ensuring participant privacy and data confidentiality. Arduino Uno data collection adheres to privacy and data protection standards, with a focus on responsible and transparent use of technology.

2.1.1.1.1.1.1.1.1. Limitations:

This study has limitations, including:

- Sample size: The number of participants may not be sufficient to generalize findings to a wider population.
- Controlled environment: Testing in a real-world setting may introduce unforeseen challenges and variations.
- Arduino Uno limitations: Technical limitations of the microcontroller, such as processing power or communication range, may need to be considered.

2.1.1.1.1.1.1.1.1.1. Conclusion:

The chosen methodology, incorporating the Arduino Uno microcontroller, provides a robust framework for the construction of the automatic water level controller. Leveraging Arduino's capabilities ensures a systematic and reliable implementation, contributing significantly to the achievement of the project's objectives. This methodology aligns with the project's commitment to precision, efficiency, and the seamless integration of technology in water level management.

CHAPTER 3

3.1 Building a Smart Water Management System with Prioritized Filling

This chapter delves into the design and construction of an automated water level control system with prioritized filling for two tanks – an underground tank and a roof tank. This system utilizes ultrasonic sensors to monitor water levels, pumps to maintain those levels, and optional SMS alerts to keep you informed.

Prioritized Filling: Keeping the Underground Tank First

The system prioritizes keeping the underground tank full for several reasons. It might be the primary water source, or it could be crucial for maintaining pressure in the pipes throughout your system. This priority ensures the underground tank reaches its full level before the roof tank pump even activates.

How it Works: A Step-by-Step Breakdown

1. **Continuous Monitoring**: Ultrasonic sensors continuously measure the distance between themselves and the water surface in each tank. By subtracting this distance from a pre-defined tank height, the system calculates the current water level.
2. **Underground Tank Takes Priority**: If the underground tank dips below its empty threshold, the system activates the corresponding pump to refill it. Additionally, an SMS alert (optional) can be sent to notify you of this change (e.g., "Underground Tank Empty").
3. **Roof Tank Waits**: Even if both tanks are empty, the roof tank pump remains inactive until the underground tank reaches its full level. The code checks this condition before allowing the roof tank pump to turn on.
4. **Simultaneous Filling:** Once the underground tank is full, the code can be modified to allow both pumps to run concurrently, speeding up the filling process for the roof tank. This feature might not be necessary depending on your specific needs.
5. **Full Tank Reached**: When the roof tank reaches its full level, the system turns off the roof tank pump (if running) and sends another SMS alert (optional) indicating this change (e.g., "Roof Tank Full").

3.1.1 Essential Components for Smart Water Management

- **Ultrasonic Sensors:** These sensors are the eyes of the system, constantly measuring the distance to the water surface and providing crucial data for level calculations.
- **Freewheeling Diodes**: When pumps shut off, they generate a brief voltage spike (back EMF) that can disrupt the Arduino. Freewheeling diodes provide a safe path for this current to dissipate, protecting the Arduino. This is a crucial component to address the Back EMF from Pumps challenge.
- **Buck Converter:** This component steps down the incoming voltage (usually 12V) to the 5V level required by the Arduino and other components like the ultrasonic sensors. Setting the output voltage correctly is vital to avoid damaging sensitive components like the SIM module. This addresses the SIM Module Damage challenge where an incorrect voltage setting can fry the SIM module (e.g., SIM800L).
- **Pumps**: These are the workhorses of the system, responsible for refilling the tanks when water levels drop below the thresholds.
- **GSM Module**: This module adds SMS notification functionality. It allows the system to send alerts to a designated phone number when tanks reach full or empty states. You can customize the message content to include the tank name (underground or roof) and its current level.

Coding the Brains of the Operation

The code for this system will likely utilize conditional statements (like if and else) to check sensor readings and control components based on the defined conditions.

This code checks the ultrasonic reading against pre-defined thresholds for empty and full conditions. It then turns the pump on or off and sends an SMS alert (optional) based on the reading.

Beyond the Basics: Additional Considerations

- Power Supply: Ensure your power supply can handle the combined current draw of all components, especially when both pumps are running simultaneously.

- Sensor Placement: Strategic placement of the ultrasonic sensors is crucial. They need a clear, unobstructed view of the water surface for accurate readings.
- Calibration: The ultrasonic sensor readings might require calibration based on the actual tank dimensions and sensor placement.
- Enclosure (Optional): Housing the system in a weatherproof

3.1.1.1 Challenges:

- **Back EMF from Pumps**: The pumps generate a voltage spike (back EMF) when switched off, which can interfere with the Arduino. This is solved by adding freewheeling diodes across the pumps.
- **SIM Module Damage:** Using an adjustable buck converter with a high voltage output can damage the SIM module (e.g., SIM800L). Ensure the voltage is set correctly for your module.
- **Soldering Issues**: Difficulty soldering bridge rectifiers can occur if the soldering iron tip is not hot enough or the wire is too thick. Pre-tinning the wire and using the right tip temperature can help.

Coding Considerations:

The code will have four main conditions to manage:

1. **Underground Tank Full:** Turns off pump, sends "Underground Tank Full" SMS alert (optional).
2. **Underground Tank Empty**: Turns on underground pump, sends "Underground Tank Empty" SMS alert (optional).
3. **Roof Tank Full**: Turns off roof pump (if running), sends "Roof Tank Full" SMS alert.
4. **Roof Tank Empty**: Checks if underground tank is full. If yes, turns on roof pump; otherwise, waits. Sends "Roof Tank Empty" SMS alert (optional).

CHAPTER 4

Building an Automated Water Level Control System with Arduino and Ultrasonic Sensors

1. Introduction

Water conservation is a growing concern. In this chapter, we'll embark on a journey to build an automated water level control system using an Arduino microcontroller and ultrasonic sensors. This system will manage two tanks, preventing overflows and ensuring they stay topped up, saving you time and water!

2. **Gathering the Parts**

Here's the arsenal we'll need:

- Arduino Uno microcontroller board - The brain of the operation.
- Ultrasonic sensors (x2) - HC-SR04 sensors are popular choices for this project.
- Breadboard - A platform for connecting components without soldering.
- Jumper wires - Colourful wires for easy connections.
- LEDs (x4) - Two different colours (e.g., green and red) to indicate water level.
- Resistors (220 ohms) - Four resistors to regulate current for the LEDs.
- 12V DC power supply - To power the system.
- Transformer- If your wall outlet voltage is much higher than 12V.
- Bridge rectifier- Converts AC to DC current (needed if using a transformer).
- Busbar - A metal strip for distributing 12V DC (handy for multiple devices).
- Water pumps (x2) - Make sure they work with 12V DC (if applicable).
- Buck converter - Steps down 12V DC to 5V DC for powering the Arduino and other components.
- Relay modules (x2) - One for each pump, acting as electronic switches.
- SIM module (GSM module - optional) - For sending SMS alerts (requires additional coding).

- Enclosure (optional) - A box to house your finished project for a clean look.

3. **Tools we use**

- Wire cutters and strippers - To prepare the jumper wires.
- Soldering iron and solder (optional) - For stronger connections, but not essential.
- Multimeter - To check voltage and continuity (helpful for troubleshooting).
- Computer - To run the Arduino software which you can download for free from https://www.arduino.cc/.

4.1 Building the System

Power Up

If our wall outlet voltage is much higher than 12V, we'll need to step it down using the transformer. Then, the bridge rectifier converts the AC current from the transformer into unregulated DC. An optional busbar can help distribute this 12V DC to the pumps and the buck converter. Finally, we'll connect the pumps directly to the 12V DC source (if we're using them).

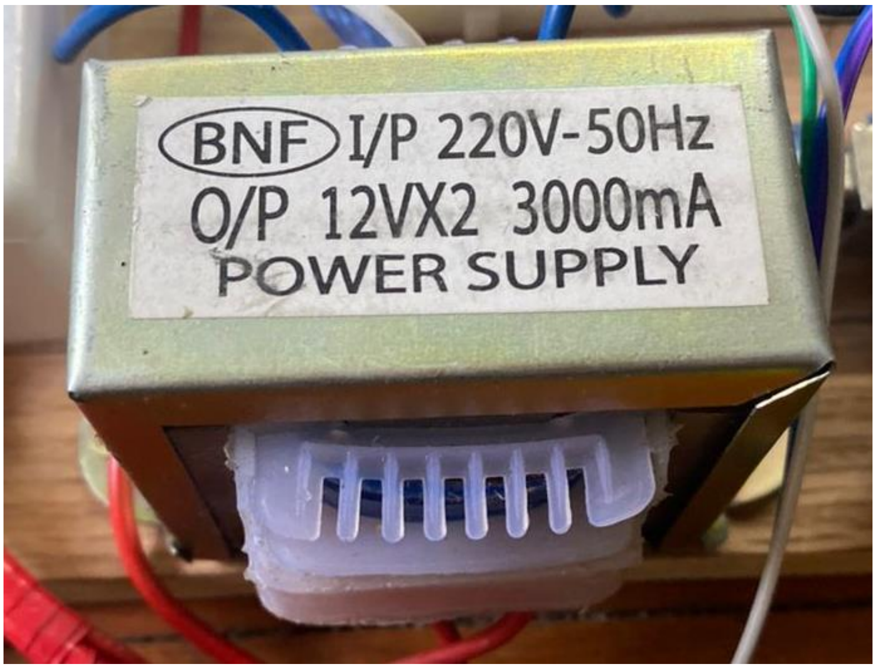


Fig 4.1 transformer

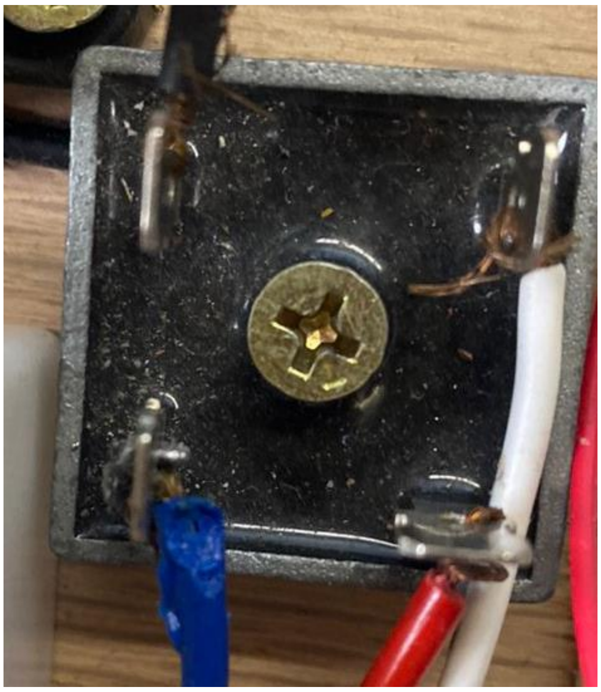


Fig 4.2 bridge rectifier

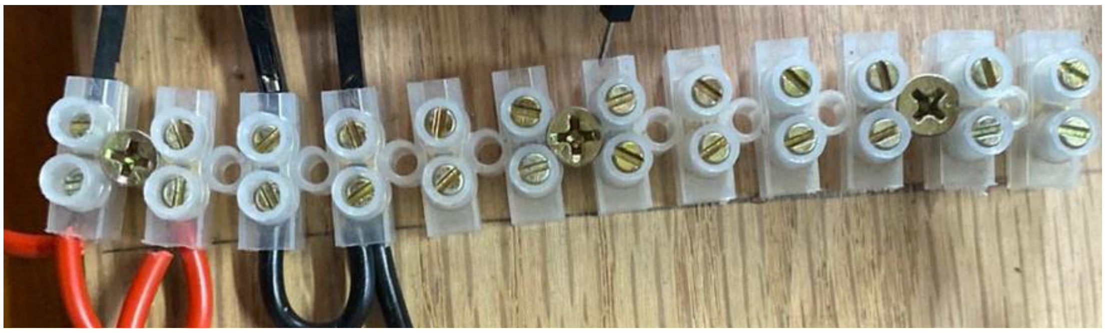


Fig 4.3 busbar

4.1.1. Building the Brains (Breadboard):

1. The buck converter takes the 12V DC (or from the busbar) and steps it down to 5V DC, which is what the Arduino and other components prefer. We'll connect the buck converter's output to the positive rail of the breadboard.

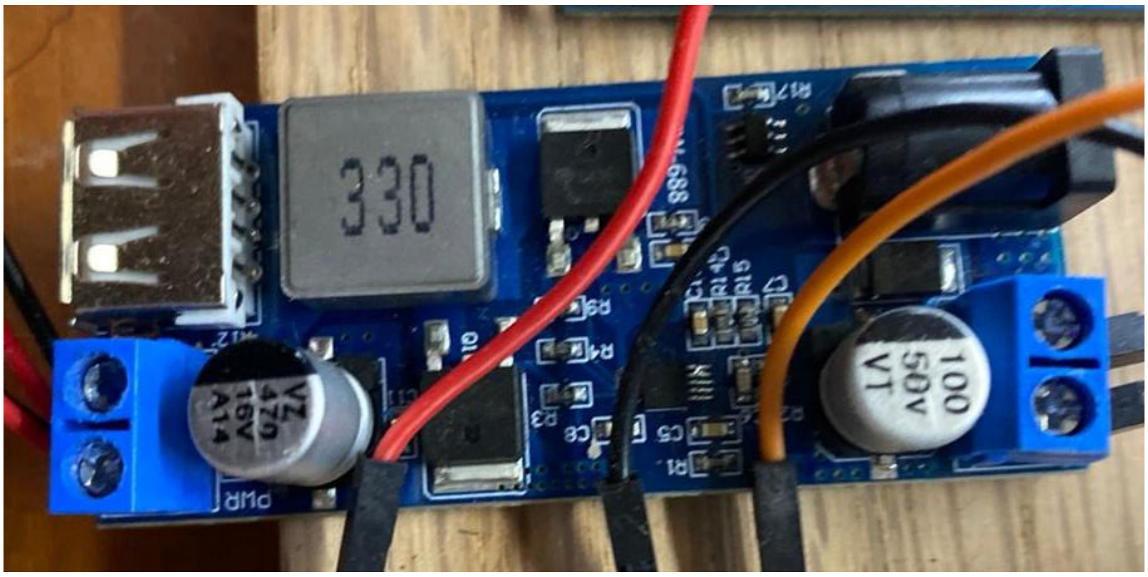


Fig 4.5 buck converter

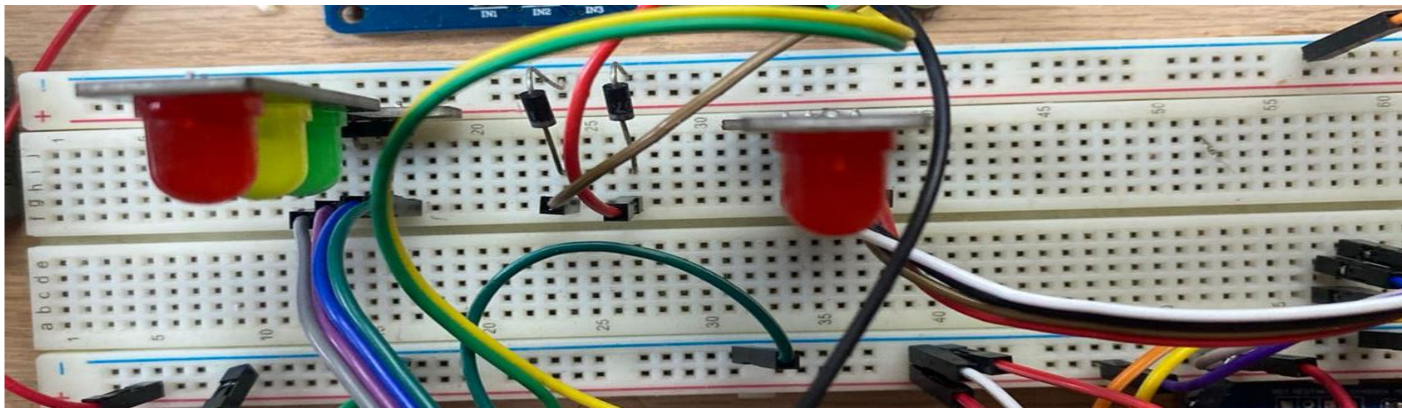


Fig 4.6 bread board

2. Plug the Arduino Uno into the breadboard, making sure the power and ground pins are lined up correctly.

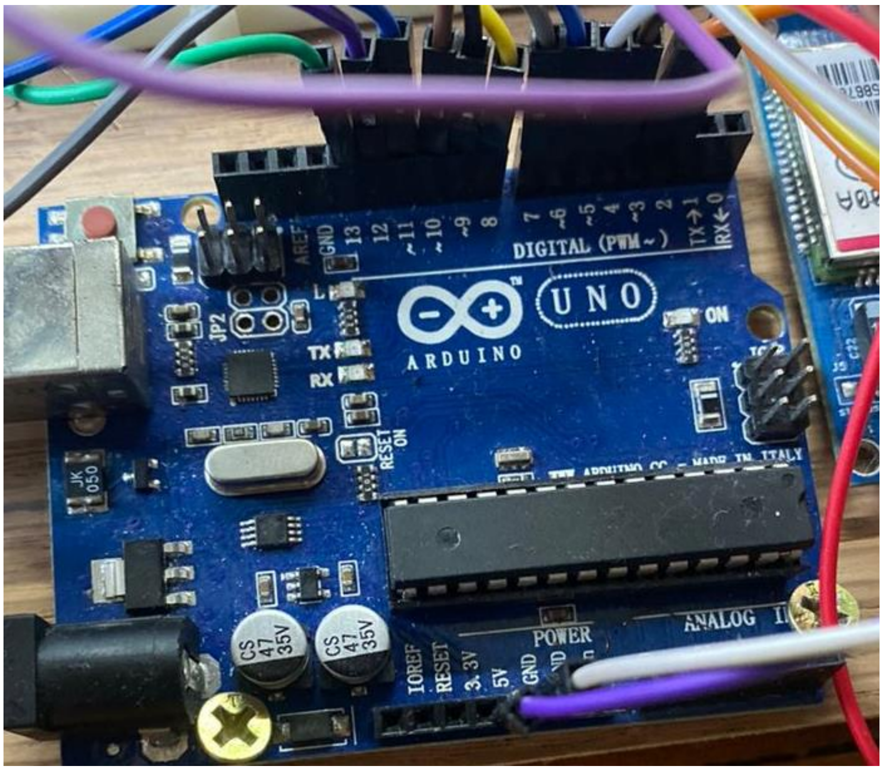


Fig 4.7 arduino

3. Time for the ultrasonic sensors! Let's connect them to the breadboard according to their specific pin configuration (check the sensor's datasheet for details).
4. Each LED. We connected LED to the breadboard to absorb the 5V for the indication, of the water tank.
5. We'll use relays to control the pumps electronically. Let's connect the relay modules (two of them) to the breadboard following their datasheet instructions.
6. We'll connect the control pins of each relay module to digital output pins on the Arduino. These pins will tell the relays when to turn on the pumps.
7. The relay modules have coils that need power. We'll connect them to the 12V DC source (or busbar if used).
8. Each relay module has three important contacts: common, normally open (NO), and normally closed (NC). We'll use the common and normally open contacts. Let's connect them to the respective water pumps, making sure the polarities are correct (positive to positive, negative to negative).
9. Thinking about adding SMS alerts? Great idea! If we have a SIM module, we'll connect it to the breadboard following its datasheet instructions.

4.1.1.1. Programming the Magic (Arduino Code):

1. We've downloaded and installed the Arduino software. Now, we need some code to tell the Arduino what to do. This code snippet is a good starting point, but you might need to adjust it based on your specific components and desired functionalities. Let's break it down:

- o Pin Definitions: The code defines which Arduino pins are connected to the ultrasonic sensors (trig and echo pins), LEDs (green and red for each tank), pumps, and (if used) the SIM module.
- o Variable Declarations: The code declares variables to store sensor readings (distance), LED statuses (on/off), pump states (on/off), and state variables for tracking tank fill levels.
- o Library Inclusion: The #include statement includes necessary libraries for the ultrasonic sensors and (if used) the SIM module. You might need to find and install these libraries separately.
- o SoftwareSerial: These lines (commented out) are for the SIM module communication. You'll need to uncomment and configure them based on your SIM module's specifications.
- o Setup Function: This function runs once when the Arduino starts up. Here, we:
- ▪ Set pin modes for all connected components (output for trig pins, input for echo pins, output for LEDs and pumps).
- ▪ Start serial communication with the computer for debugging purposes (optional).
- ▪ Initialize the pumps to be off (optional, depending on your wiring).
- ▪ (Optional) Initialize the SIM module communication (refer to its datasheet).
- o Loop Function: This function runs repeatedly. Here's the core logic:

  Ultrasonic Sensor Readings:
- ▪ The code clears the trig pin.
- ▪ It sends a short pulse to the trig pin to initiate a sound wave.
- ▪ It reads the echo pin to measure the time it takes for the sound wave to travel to the water surface and back.
- ▪ Based on the travel time, the code calculates the distance to the water surface in each tank.

LED and Pump Control:

- The code checks the distance readings and state variables to control the LEDs and pumps for each tank:
- If the distance is less than or equal to 5 cm (full tank) and the state variable is 0 (not already full), the green LED turns on, the red LED turns off, the pump turns off (optional, depending on wiring), and an SMS alert is sent (if using a SIM module). The state variable is then set to 1 (full).
- If the distance is between 15 cm and 25 cm (empty tank) and the state variable is 1 (previously full), the red LED turns on, the green LED turns off, the pump turns on (optional, depending on wiring), and an SMS alert is sent (if using a SIM module). The state variable is then set to 0 (empty).

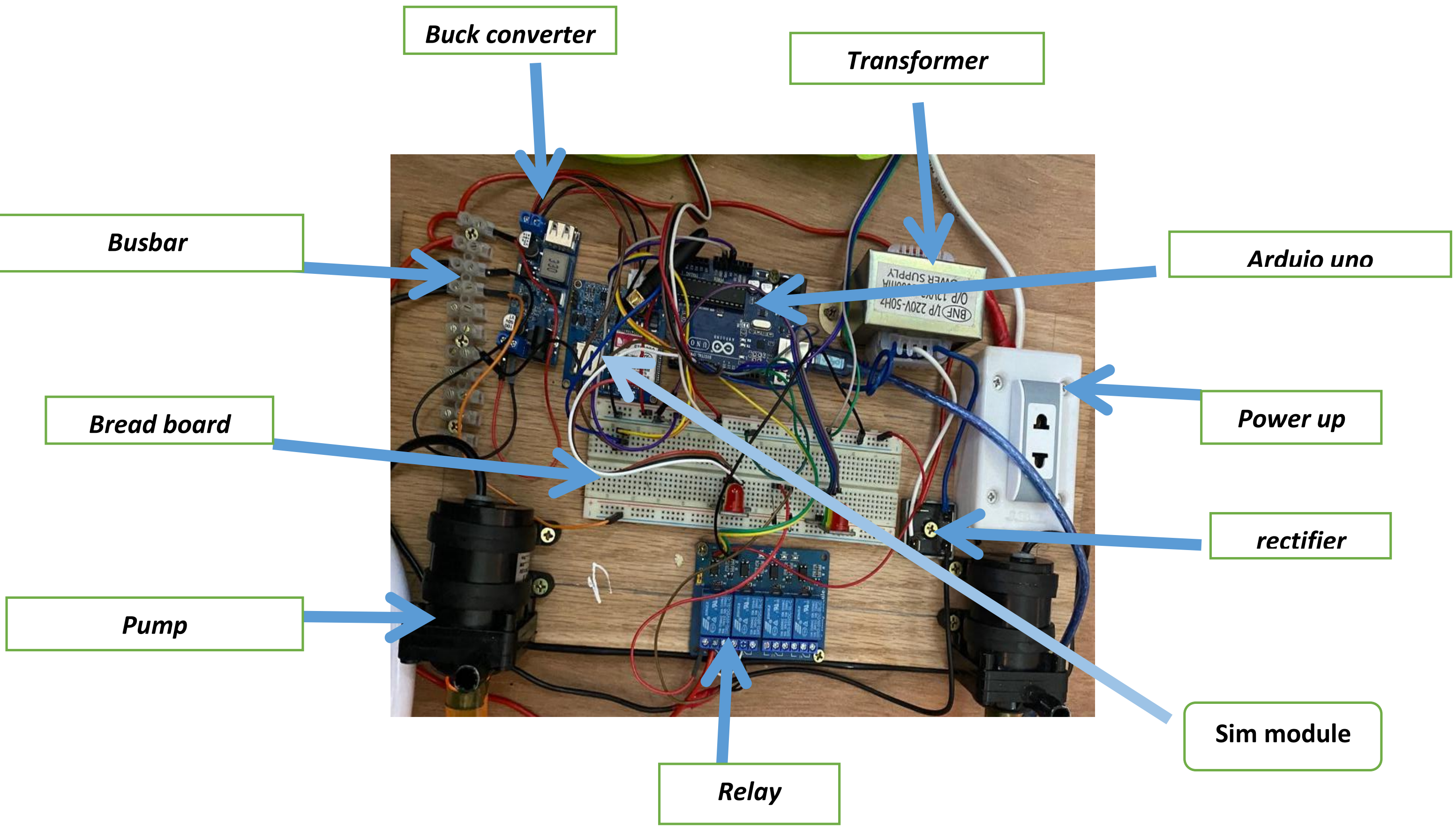
4.1.1.1.1. Testing and Calibration:

1. Connect the complete system to a power source.
2. Verify the functionality of the LEDs by uploading a simple test sketch to the Arduino (not included in this code snippet).
3. Fill one of the tanks with water. Observe the ultrasonic sensor readings using the Arduino serial monitor (opened in the Arduino software).
4. Adjust the water level thresholds in the code (e.g., 5 cm for full) based on the observed sensor readings for full and empty tank conditions.
5. Test the pump activation by manually triggering the code or using a test sketch. Ensure pumps turn on and off at the desired water levels.
6. (Optional) If using a SIM module, test the SMS sending functionality by triggering the code or using a test sketch. You might need to replace the phone number placeholder with your actual number.
7. Final Stage and Deployment

   Once everything is tested and calibrated, you can:

- Enclose the project in a box for a clean look (optional).

- Mount the system near the tanks, ensuring the ultrasonic sensors have a clear view of the water surface.
- Now you have an automated water level control system that will keep your tanks topped up and prevent overflows!



CHAPTER 5

FUTURE PLAN AND CONCLUSION

5.1. Future Plan

In this chapter, we embark on a visionary journey towards sustainable water management by exploring future strategies and implementation methods. We delve into innovative approaches such as rainwater harvesting, greywater recycling, and leak detection and repair automation to promote environmental stewardship and resource conservation.

1. Rainwater Harvesting: Maximizing Rainwater Usage

Implementation Steps:

Collection Infrastructure:

Install gutters and downspouts to channel rainwater from rooftops.

Direct rainwater into storage tanks or barrels using a network of pipes and filters.

Advanced Filtration:

Utilize mesh screens and first flush diverters to remove debris and contaminants.

Employ advanced filtration systems like sediment filters and UV sterilizers for improved water quality.

Smart Distribution:

Implement an automated distribution system for efficient routing of rainwater to usage points.

Use gravity-fed or pump-assisted methods for irrigation, landscaping, and household uses.

2. Greywater Recycling: Reusing Water Responsibly

Implementation Steps:

Dual Plumbing Systems:

Separate greywater from black water with dedicated plumbing systems.

Direct greywater to treatment and recycling facilities.

Treatment Innovations:

Employ filtration and biological treatment processes for greywater purification.

Implement advanced technologies like membrane bioreactors and reverse osmosis systems.

Efficient Distribution:

Design a distribution network for recycled greywater delivery.

Incorporate automated controls for optimized usage and conservation.

3. Leak Detection and Repair Automation: Preventing Water Waste

Implementation Steps:

Sensor Integration:

Deploy water leak detection sensors strategically throughout the plumbing infrastructure.

Connect sensors to a centralized monitoring system for real-time detection.

Automated Response:

Integrate automated shutoff valves to isolate leaks and prevent water loss.

Implement alert mechanisms for prompt notification and action.

Continuous Monitoring and Maintenance:

Establish regular inspection schedules for monitoring system effectiveness.

Conduct maintenance to ensure sensors, valves, and monitoring equipment are operational.

By implementing these forward-thinking strategies, we can create a sustainable water management system that conserves resources, minimizes waste, and preserves the environment for future generations. Through innovation and collaboration, we can achieve our vision of a greener, more resilient world.

5.1.1 Conclusion

Our project, "Building an Automated Water Level Control System with Arduino and Ultrasonic Sensors," has successfully demonstrated a practical solution for managing water levels in tanks. By utilizing Arduino microcontrollers and ultrasonic sensors, we have developed a system that prevents overflows and ensures efficient water usage, addressing a critical real-life problem of water conservation.

Key Stages of the Project:

Gathering Parts and Tools:

Assembled necessary components, including Arduino boards, ultrasonic sensors, LEDs, water pumps, and power supplies.

Utilized tools such as wire cutters, soldering irons, and multimeters.

System Construction:

Built the system in stages, starting with power management using transformers, bridge rectifiers, and buck converters to step down voltage.

Set up the Arduino and connected the sensors, LEDs, and pumps using a breadboard.

Programming:

Wrote and uploaded code to the Arduino, enabling it to read sensor data and control the pumps and LEDs.

Ensured that pumps activated when the water level was low and deactivated when it was high, with visual indicators provided by the LEDs.

Testing and Calibration:

Rigorously tested the system to ensure accurate sensor readings and proper pump activation.

Made adjustments to fine-tune the water level thresholds and ensure reliable performance.

Final Deployment:

Housed the system in an enclosure and installed it near the water tanks, with ultrasonic sensors positioned to monitor water levels accurately.

Ensured ongoing water level management, preventing overflows and maintaining optimal levels.

Overall, the project not only addressed the technical challenges of building an automated control system but also emphasized the importance of practical problem-solving and efficient resource management. The result is a reliable and effective solution that conserves water and reduces the need for manual monitoring, showcasing the potential of modern technology in everyday applications.

5.1.1.1. Outcome-Based Education (OBE) Principles

The culmination of our project, "Building an Automated Water Level Control System with Arduino and Ultrasonic Sensors," showcases the practical application of Outcome-Based Education (OBE) principles. Through the meticulous design, development, and implementation of this project, we have effectively addressed multiple Course Outcomes (COs) and Program Outcomes (POs), ensuring a holistic and rigorous engineering education experience.

Our project successfully identified a contemporary real-life problem, emphasizing the importance of water conservation, which is increasingly critical in today’s context. By leveraging modern engineering tools and methodologies, we proposed and implemented a viable solution that not only prevents tank overflows but also ensures efficient water usage. This aligns with PO2 and PO4, demonstrating our ability to analyze and synthesize information to solve complex engineering problems.

Throughout the project, we adhered to professional ethics, standards, and codes, highlighting the importance of ethical considerations in engineering solutions (PO8). The use of modern engineering resources and tools (PO5) facilitated the development of an efficient and effective solution, further underscoring the importance of technological proficiency in contemporary engineering practice.

The project's impact on health, safety, society, and the environment was thoroughly analysed and considered, aligning with PO6 and PO7. This comprehensive approach ensures that our solution is not only technically sound but also socially responsible and sustainable.

Our teamwork and individual contributions were crucial in achieving the project goals (PO9). Effective communication (PO10) was maintained throughout, ensuring clear documentation and presentation of our findings. The project's management aspects, including budgetary planning and resource allocation, were meticulously handled, addressing PO11.

Recognizing the need for continuous learning and professional development (PO12), this project has equipped us with the skills and knowledge necessary for lifelong learning and adaptation in the rapidly evolving field of electrical and electronic engineering.

In conclusion, our project not only demonstrates the integration of OBE principles but also provides a practical solution to a real-world problem. It exemplifies the application of theoretical knowledge to practical scenarios, preparing us for future professional challenges. The project’s success is a testament to the effectiveness of OBE in fostering comprehensive educational development and professional readiness in engineering students.

REFERENCE

1. Journal Articles

   Gleick, P. H. (2003). "Water Use," Annual Review of Environment and Resources, vol. 28, no. 1, pp. 275-314.

   Gude, V. G. (2015). "Desalination and sustainability – An appraisal and current perspective," Water Research, vol. 89, pp. 87-106.

2. Conference Proceedings

   Bayindir, R., Sefa, I., & Colak, I. (2007). "The development of a smart home energy management system," Scientific Research and Essays, vol. 2, no. 10, pp. 481-488.

3. Dissertation/Project

   Knight, K. A. (2011). "Media epidemics: Viral structures in literature and new media," A Doctoral Dissertation presented at University of California, Santa Barbara.

4. Internship Report

   Alejandro Maria (2009). "Development of a GUI for a SAXS data analysis modeling program for modeling proteins in solution," An internship report presented at Joseph Fourier University, France.

5. **Books**

   Jennings, D., & Flint, A. (2010). Introduction to Medical Electronics Applications, 4th Edition, John Wiley and Sons.

6. **Electronic Resources (websites)**

   UN Water. (2021). The United Nations World Water Development Report 2021: Valuing Water. Accessed at http://www.unwater.org/publications/un-world-water-development-report-2021-valuing-water/ on 23 October, 2022.

7. **Company/Industry/Organization Report**

   EPA (2021). Water Efficiency Management Guide: Home Water Use. Environmental Protection Agency.

APPENDICES

Code

```
// defines arduino pins numbers
const int trigPin = 10; const int
echoPin = 9; const int trigPin1 =
12; const int echoPin1 = 11; // de-
fines variables long duration; int
distance; int green = 7; int red = 6;


long     duration1;
int distance1; int
green1  =  5;  int
red1 = 4;


int pump = 13; int
pump1 = 8;


int state=0; int
state1=0;


#include <SoftwareSerial.h>



//SoftwareSerial sgps(12,13);

SoftwareSerial sgsm(2, 3);


void setup()
```

```
{

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output pinMode(echoPin,
INPUT); // Sets the echoPin as an Input


pinMode(green, OUTPUT);


pinMode(red, OUTPUT);


pinMode(trigPin1, OUTPUT); // Sets the trigPin as an Output pinMode(echoPin1,
INPUT); // Sets the echoPin as an Input


pinMode(green1, OUTPUT);



pinMode(red1, OUTPUT);


pinMode(pump,  OUTPUT);   pinMode(pump1,
OUTPUT);

Serial.begin(9600); // Starts the serial communication
digitalWrite(pump, HIGH); digitalWrite(pump1, HIGH);


sgsm.begin(9600);

Serial.println("Setup Complete.... ");

}
```

```
void loop() { // Clears the
trigPin digitalWrite(trigPin,
LOW);        delayMicrosec-
onds(10);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin,   HIGH);   delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH); // Calculating the distance distance= du-
ration*0.034/2;

// Prints the distance on the Serial Monitor

//Serial.print("Distance = ");

//Serial.print(distance);

//Serial.println(" cm");

//delay(100);


digitalWrite(trigPin1, LOW); delayMicroseconds(10);

// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin1,   HIGH);   delayMicroseconds(10);
digitalWrite(trigPin1, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration1 = pulseIn(echoPin1, HIGH); // Calculating the distance distance1=
duration1*0.034/2;

// Prints the distance on the Serial Monitor

//Serial.print("Distance1 = ");

//Serial.print(distance1);

//Serial.println(" cm");
```

```
//delay(100);


//............UNDER GROUND TANK...... if (distance<= 5 && state == 0){
//.....UNDERGROUND TANK FULL.....

 digitalWrite(green, HIGH);  digitalWrite(red,
LOW);  digitalWrite(pump, HIGH);



  Serial.println("Sending.....");
sgsm.write("\r");       delay(1000);
sgsm.write("AT+CMGF=1\r");
delay(1000);

     /*Replace XXXXXXXXXX to 10 digit mobile number &        ZZ to 2
digit                               country                               code*/
sgsm.write("AT+CMGS=\"+8801641160290\"\r");              //  seri-
alSIM800.write("AT+CMGS=\"+8801987141781\"\r\n");            de-
lay(1000);      sgsm.write("Undergroung Tank Full");




     delay(1000);
sgsm.write(0x1A);      delay(1000);

    Serial.println("MESSAGE SENT Undergroung Tank Full");



  state = 1;

}
if ((distance> 15 && distance<= 25)&& state == 1 ){//......UNDERGROUND TANK EMPTY......
```

```
 digitalWrite(red, HIGH);  digitalWrite(green,
LOW);   digitalWrite(pump, LOW);



  Serial.println("Sending.....");
sgsm.write("\r");       delay(1000);
sgsm.write("AT+CMGF=1\r");
delay(1000);

     /*Replace  XXXXXXXXXX  to  10  digit  mobile  number  &
ZZ          to          2          digit          country          code*/
sgsm.write("AT+CMGS=\"+8801641160290\"\r");

      //   serialSIM800.write("AT+CMGS=\"+8801987141781\"\r\n");
delay(1000);      sgsm.write("Undergroung Tank Empty");




     delay(1000);
sgsm.write(0x1A);      delay(1000);

    Serial.println("MESSAGE SENT Undergroung Tank Empty");



 state = 0;



}
//..................ABOVE SURFACE TANK.....................



if (((distance1<= 5) && state1 == 0)){ //...........ABOVE TANK FULL...........

 digitalWrite(green1, HIGH);   digitalWrite(red1,
LOW);  digitalWrite(pump1, HIGH);
```

```
Serial.println("Sending.....");
sgsm.print("\r");       delay(1000);
sgsm.print("AT+CMGF=1\r");
delay(1000);

     /*Replace XXXXXXXXXX to 10 digit mobile number &        ZZ to 2
digit                              country                              code*/
sgsm.print("AT+CMGS=\"+8801641160290\"\r");                // seri-
alSIM800.write("AT+CMGS=\"+8801987141781\"\r\n");             de-
lay(1000);      sgsm.print("Above Tank Full");




     delay(1000);
sgsm.write(0x1A);      delay(1000);

    Serial.println("MESSAGE Above Tank Full");


 state1 = 1;


}


if (((distance1> 15 && distance1<= 25)&& state1 == 1)&& state == 1){//......ABOVE TANK EMPTY.........

 digitalWrite(red1, HIGH);   digitalWrite(green1,
LOW);   digitalWrite(pump1, LOW);


Serial.println("Sending.....");
sgsm.print("\r");       delay(1000);
sgsm.print("AT+CMGF=1\r");
delay(1000);

    /*Replace XXXXXXXXXX to 10 digit mobile number &
ZZ to 2 digit country code*/
sgsm.print("AT+CMGS=\"+8801641160290\"\r");

     //   serialSIM800.write("AT+CMGS=\"+8801987141781\"\r\n");
delay(1000);      sgsm.print("Above Tank Empty");




    delay(1000);
sgsm.write(0x1A);      delay(1000);

   Serial.println("MESSAGE SENT Above Tank Empty");



 state1 = 0;



}
}
```