VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI, KARNATAKA-590018

"AUTO WATER DISPENSER"

A Report

Submitted By

Ashwini C N	**4MN22CS008**
Bhanu Prakash R	**4MN22CS009**
Chandana M	**4MN22CS013**

Submitted in partial fulfillment of the requirement for the award of the degree of
Bachelor of Engineering in Computer Science and Engineering

Under the guidance of

Prof. ASHWINI G
Prof. MADHU B

Assistant Professor
Department of Computer Science & Engineering
MIT Thandavapura

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA
Just Off NH 766, Nanjanagudu Taluk, Mysore District – 571302
(Approved by AICTE, Accredited by NBA, New Delhi and Affiliated by VTU, Belagavi)

2024-2025

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

Just off NH 766, Nanjanagudu Taluk, Mysore District – 571302

(Approved by AICTE, Accredited by NBA, New Delhi and Affiliated by VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled **“Auto Water Dispenser**” carried out by **Ashwini C N**(4MN22CS008), **Bhanu Prakash R**(4MN22CS009), **Chandana M**(4MN22CS013) a students of **Maharaja Institute of Technology Thandavapura** in partial fulfilment for the award of Bachelor of Engineering in **Computer Science & Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2024-25. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signature of guide	Signature of the HOD	Signature of the Principal
Prof. Ashwini G	**Dr. Ranjit K N**	**Dr. Y. T. Krishne Gowda**
Prof. Madhu B		
Assistant Professor	Associate Professor	Principal
Dept. of CS&E,	HoD of CS&E	MIT Thandavapura
MIT Thandavapura	MIT Thandavapura	

Declaration

This is to declare that this report has been written by us. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. We aver that if any part of the report is found to be plagiarized, we will be solely responsible for it.

Ashwini C N

4MN22CS008

Bhanu Prakash R

4MN22CS009

Chandana M

4MN22CS013

Place: Mysuru

Date: 24-12-2024

ACKNOWLEDGEMENT

Wholeheartedly, I thank our MET Management for providing the necessary environment, infrastructure and encouragement for carrying out our project work at Maharaja Institute of Technology Thandavapura, Mysuru.

We would like to express our sincere thanks to our beloved Principal Dr. Y.T. Krishne Gowda for giving us moral support and continuous encouragement which has been the key for the successful completion of the mini project.

We are pleased to acknowledge Dr. Ranjit K N, HoD, Department of Computer Science and Engineering, for her encouragement and support throughout the project.

We would like to express our heart-felt gratitude to our Mini Project Co-ordinator Prof. Ashwini G and Prof. Madhu B, Assistant Professor, Department of Computer Science and Engineering for his valuable suggestions and excellent guidance rendered throughout this project.

Further, we extend our thanks to all the faculty members and technical staff of our department for their suggestions, support and providing resources at needed times.

Finally, we express our sincere gratitude to our parents and friends who have been the embodiment of love and affection which helped us to carry out the project in a smooth and successful way.

Ashwini C N[4MN22CS008]

Bhanu Prakash R[4MN22CS009]

Chandana M[4MN22CS013]

ABSTRACT

An **auto water dispenser** is a system designed to automatically provide water without manual intervention. It typically consists of a container or tank, a mechanism for dispensing the water, and sensors or controls to monitor the water level or detect when water is needed. These dispensers can be used in a variety of settings, including homes, offices, or public spaces, ensuring a consistent supply of water while conserving resources. The system may feature features such as automatic refilling, filtration, temperature control, and user-friendly interfaces for ease of use.

LIST OF CONTENTS

LIST OF FIGURES

Chapter 1
INTRODUCTION

1.1 Background

The background of **auto water dispensers** can be traced to the need for more efficient and convenient water dispensing systems, particularly in environments with high water consumption, such as homes, offices, and public spaces. Traditional water dispensers required manual filling or constant attention to ensure a steady supply of water, which led to the development of automatic systems.
Early versions of auto dispensers were simple mechanical systems that relied on gravity or basic float mechanisms to control water flow. With advancements in technology, modern auto water dispensers now incorporate electronic sensors, smart controls, and filtration systems to offer enhanced functionality, convenience, and hygiene.
These systems are particularly useful in reducing waste, maintaining water temperature, and ensuring a continuous, clean water supply without human intervention, making them popular in various sectors, from residential to commercial use.

Water dispensers have become a common household and office appliance, providing a convenient way to access drinking water. The idea of easily accessible, purified water dates back to the early 20th century, although the concept of water dispensing has existed in various forms for centuries.

1.2 Problem statement

In many households and workplaces, accessing clean and safe drinking water remains a challenge due to issues such as contamination, lack of proper filtration, and the inconvenience of manual water collection. Additionally, traditional water dispensing methods, such as using bottled water, can result in high costs, environmental waste, and limited availability of water temperature options. There is also a growing concern over the hygiene and efficiency of water dispensers, especially in shared spaces like offices, where frequent handling of dispensers can lead to contamination.

1.2Objective

The objective of an **auto water dispenser** is to provide a convenient, efficient, and hygienic way to dispense water automatically, reducing the need for manual intervention. Key objectives include:

1. **Convenience**: Ensuring users have easy access to water without the need for constant refilling or manual effort.
2. **Efficiency**: Optimizing water usage by controlling flow, refilling, and reducing waste.
3. **Hygiene**: Minimizing contamination by providing filtered and clean water through automatic mechanisms.
4. **Resource Conservation**: Ensuring water is dispensed only when needed, preventing overuse.
5. **User Experience**: Offering a user-friendly interface with features like temperature control, automatic refilling, and easy maintenance.

1.4 Scope

The **scope** of an **auto water dispenser** is broad, extending across multiple sectors where easy access to clean water is essential. As technology advances, the scope of these devices continues to expand, incorporating new features to meet diverse needs. The primary areas of scope for auto water dispensers include:

1. **Residential Use**:
   - Home installations providing clean, filtered, and temperature-controlled water.
   - Integration with smart home devices for easy operation and monitoring.
2. **Commercial Use**:
   - Offices and workplaces to ensure employees have quick access to hydration without manual refilling.
   - Businesses like restaurants, hotels, and cafes, where automatic dispensers streamline water access for both staff and customers.
3. **Public Spaces**:
   - In parks, gyms, airports, and shopping malls, auto dispensers offer accessible, clean water, contributing to sustainability by reducing bottled water consumption.
4. **Healthcare and Medical Facilities**:
   - In hospitals and clinics, ensuring sterile and filtered water is available, especially in high-traffic areas, while minimizing the risk of contamination through touchless dispensing.

- Use in medical research and laboratories where precise water quality is needed.

5. **Industrial and Manufacturing**:
   - Providing water for employee hydration in factories or large production areas, improving convenience and worker wellness.
   - Integration in processes that require water for cooling or other equipment needs.
6. **Agriculture and Farming**:
   - Auto dispensers used for livestock hydration, ensuring animals have continuous access to clean water.
   - Integration into irrigation systems to efficiently distribute water to crops.
7. **Sustainability and Environmental Impact**:
   - Reducing single-use plastic bottle consumption, as dispensers can be used with reusable containers.
   - Water conservation through sensors that monitor water levels and control water flow, reducing waste.
8. **Smart and IoT Integration**:
   - Connectivity with IoT devices, allowing for remote control, real-time monitoring, predictive maintenance, and analytics on water usage.
   - Smart dispensers that adjust water temperature, monitor hydration patterns, and optimize water use based on data.

1.5 Application

The **applications** of auto water dispensers are varied, meeting the needs of both individual consumers and large institutions. Some key applications include:

1. **Home Use**:
   - Providing instant access to filtered water in kitchens, living rooms, or home offices.
   - Ensuring water temperature preferences, such as chilled or hot water, are met without needing to manually adjust or wait.
2. **Office and Commercial Settings**:
   - Providing hydration to employees, reducing the need for bottled water, and promoting wellness in workplace environments.
   - Offices with large numbers of people use these dispensers to meet the demand for drinking water without constant maintenance.
3. **Public Spaces**:
   - In public areas like parks, airports, and shopping malls, auto water dispensers provide

easy access to water while promoting sustainability by reducing plastic bottle usage.

- Events such as festivals, conferences, and sports events often employ auto dispensers to cater to large crowds, reducing waste and ensuring a steady water supply.

4. **Healthcare Settings**:
    - Touchless dispensers in hospitals or clinics ensure that patients and healthcare professionals can easily access water while minimizing the spread of germs and maintaining hygiene.
    - Some dispensers offer water with specific purification levels for medical use or patient hydration.
5. **Schools and Educational Institutions**:
    - Auto water dispensers in school cafeterias, classrooms, and outdoor areas promote hydration for students and staff, often reducing reliance on bottled water.
    - Universities and colleges use these systems in dorms or common areas for convenience and sustainability.
6. **Food and Beverage Industry**:
    - Restaurants and cafes use auto dispensers to offer chilled or filtered water to customers, reducing the need for disposable plastic bottles and improving customer experience.
    - In hotels, dispensers may be placed in rooms or lobbies to provide easy access to water for guests.
7. **Industrial Applications**:
    - In large industrial or manufacturing facilities, dispensers provide water to employees on the floor, ensuring hydration during long shifts.
    - Auto dispensers can be integrated into industrial systems for water cooling or other operations where consistent water supply is needed.
8. **Agriculture**:
    - Auto dispensers in farms or ranches provide consistent water supply to livestock, ensuring proper hydration without constant human intervention.
    - In greenhouses, dispensers help automate irrigation, delivering water to crops based on sensors that monitor soil moisture.
9. **Emergency and Disaster Relief**:
    - In areas affected by natural disasters or crises, portable auto water dispensers can provide clean drinking water to displaced people, using filtration systems to ensure water safety.

- These dispensers can also be used in refugee camps or temporary shelters to ensure access to water when traditional water supply systems are unavailable.

In summary, the **scope** and **applications** of auto water dispensers cover a wide range of industries and sectors, from home use to commercial, healthcare, public spaces, and industrial settings. As they evolve with new technologies like IoT integration, water quality monitoring, and smart features, their role in improving convenience, hygiene, and sustainability continues to expand.

Chapter 2

SYSTEM REQUIREMENT AND SPECIFICATION

2.1 Functional Requirement

The **functional requirements** define what the system (in this case, the system controlling the **servo motor** using the **ultrasonic sensor**) should do. Below are the key functional requirements for the provided code:

1. **Ultrasonic Distance Measurement**:
   - The system should use the ultrasonic sensor to measure the distance between the sensor and the nearest object in front of it.
   - The sensor should send out a pulse and measure the time it takes for the pulse to bounce back from the object.
   - The system should calculate the distance in centimeters based on the time it takes for the pulse to return.
2. **Distance Threshold**:
   - The system must define a threshold distance (in this case, 6 cm).
   - If the measured distance is less than 6 cm, the system should activate the servo motor to move to the center position (130 degrees).
   - If the distance is greater than or equal to 6 cm, the system should move the servo motor to the 0-degree position.
3. **Servo Control**:
   - The servo motor should move to the specified angle (0 or 130 degrees) based on the distance measurement.
   - If the distance is less than 6 cm, the servo motor should remain at 130 degrees for 7 seconds.
   - If the distance is greater than 6 cm, the servo motor should move back to 0 degrees.
   - The servo movement should occur in response to the change in distance measurement.
4. **Continuous Operation**:
   - The system must operate continuously, measuring the distance and controlling the servo accordingly in a loop.
   - The ultrasonic sensor should continuously emit pulses to measure the distance and check against the threshold.
   - The servo should only move to the 90-degree position when the distance is less than 6

cm and should reset to the 0-degree position when the distance is greater than or equal to 6 cm.

5. **Time Delay for Servo in 130-Degree Position**:
   - When the servo is at the 130-degree position (when the distance is below 6 cm), it should remain in that position for exactly 7 seconds before moving back to the 0-degree position.
6. **Distance Conversion**:
   - The system should accurately convert the pulse duration into a distance measurement in centimeters using the formula:

$$cm = \frac{58.82}{duration}$$

   - The conversion should account for variations in sensor performance (e.g., in case of no object or an invalid measurement, the system should handle those cases appropriately).
7. **Sensor Pulse Control**:
   - The system must control the **trigger pin** to send out the pulse (HIGH for 10 microseconds) and then **receive** the echo signal on the **echo pin** to measure the time taken for the pulse to return.

These functional requirements outline the primary tasks and behaviors that the system (the code provided) must perform.

2.2 Non-Functional Requirement

Non-functional requirements describe the overall attributes or qualities of the system, such as performance, reliability, usability, and other characteristics. These are the factors that ensure the system operates smoothly and meets user expectations, even though they don't directly describe specific functionalities. Below are the **non-functional requirements** for the provided **servo control system** using the **ultrasonic sensor**:

1. **Performance**:
   - The system must be able to measure the distance and adjust the servo motor position in real-time with minimal delay.
   - Distance measurement should be completed in under 100 milliseconds, ensuring quick response time for servo movements.
   - The servo motor should respond immediately to distance changes, and the system should avoid excessive lag or waiting times between readings.
2. **Reliability**:
   - The system should reliably measure the distance and control the servo motor without

frequent errors or malfunctions.
  - The ultrasonic sensor should consistently return accurate distance readings within the operational range (usually up to 400 cm).
  - The servo motor should work without mechanical failure over extended periods of use.

3. **Scalability**:
   - The system should be able to handle potential future integration with additional sensors or features (e.g., multiple sensors for different angles or a more complex servo system).
   - If the system is expanded to control more than one servo or sensor, it should be easily adaptable with minimal rework in the code.
4. **Usability**:
   - The system should be easy to install and operate, with minimal setup required by the user (e.g., connecting the ultrasonic sensor and servo motor to the correct pins).
   - The behavior of the servo (moving to 130 degrees when the object is close and 0 degrees otherwise) should be intuitive and straightforward to users, requiring no additional explanation or configuration.
5. **Energy Efficiency**:
   - The system should operate in a power-efficient manner. Since it's an embedded system, the sensor and servo should only be active when necessary, reducing overall power consumption.
   - The system should minimize power usage during idle times, especially the servo motor, which should only move when required (based on the distance).
6. **Maintenance**:
   - The system should be easy to maintain, requiring minimal intervention. This could include simple tasks such as ensuring the servo motor is functioning, the ultrasonic sensor is clean, and the wiring remains intact.
   - The code should allow for easy updates or modifications, such as adjusting the threshold distance or modifying the servo movement behavior.
7. **Error Handling**:
   - The system should be able to handle errors gracefully, such as invalid or out-of-range distance readings. If an invalid reading occurs (e.g., a very high or zero value), the system should discard it or give an appropriate error message.
   - If the ultrasonic sensor fails (e.g., disconnected or malfunctioning), the system should notify the user or handle the failure by performing a fallback action (like stopping the servo movement).

8. **Safety**:
    - The system should ensure that the servo motor does not overheat or get damaged due to constant movement. The servo's limits should be respected in the code.
    - The system should ensure that no harmful voltages are sent to the servo or the sensor, and proper safety measures should be followed in hardware wiring.
9. **Robustness**:
    - The system should be robust enough to handle small fluctuations in the environment, such as small variations in the power supply or interference from nearby objects that might affect the ultrasonic sensor's readings.
    - The code should handle minor hardware issues, such as short-lived sensor misreads, without crashing or hanging the system.
10. **Portability**:
    - The system should be portable and easily transportable to different environments (e.g., different Arduino boards or microcontrollers). The code should be able to run on different hardware platforms (like Arduino Uno, Mega, Nano) with minimal modification.
11. **Scalability**:
    - The system should support adding more sensors or servos without needing a complete rewrite. The modularity of the design should allow for future improvements (e.g., adding more distance sensors for broader area monitoring).
12. **Compatibility**:
    - The system should be compatible with various servo motors and ultrasonic sensors, allowing for easy replacements or upgrades of hardware components.
    - It should also be compatible with standard Arduino environments (e.g., Arduino IDE), allowing users to upload and modify the code easily.
13. **Aesthetic Design**:
    - Although this is more relevant for physical products, the hardware should be designed in a way that allows easy integration of the sensor and servo into existing environments, whether in a machine, a prototype, or a real-world application.

These non-functional requirements ensure that the system is effective, reliable, and maintainable in real-world applications, offering a good balance between performance and usability.

2.2.1 Performance Requirement

Performance requirements define how well the system should perform under various conditions. These ensure that the code operates efficiently and meets the required response times, speed, and resource usage. Below are the performance requirements specific to the provided servo control system using the ultrasonic sensor.

1. Response Time:

- Distance Measurement Response Time:
    - ➢ The system should measure the distance and calculate the result (in centimeters) within 100 milliseconds after the ultrasonic sensor sends a pulse. This ensures the servo motor reacts promptly.
- Servo Response Time:
    - ➢ Once the distance is measured, the servo motor should respond immediately. There should be no noticeable delay in servo movement when the distance threshold changes (e.g., when the object moves closer or farther).

2. Update Frequency:

- The system should measure the distance and adjust the servo position at least once every 100 milliseconds (ideally faster), ensuring continuous monitoring of the environment and timely adjustments of the servo motor position.
- Sensor Pulse Rate:
    - ➢ The ultrasonic sensor should send pulses at intervals of 1 second or less. This allows the system to continuously monitor the distance and control the servo without causing unnecessary delays.

3. Accuracy of Measurement:

- The distance measurements should be accurate to within ±1 cm for typical operating conditions (i.e., within the normal range of the ultrasonic sensor).
- If the distance exceeds the sensor’s maximum range (typically 400 cm), the system should handle the scenario gracefully and either return a default value or skip the erroneous reading.

4. Servo Control Precision:

- The servo motor should be able to move to the desired position (0° or 130°) with high precision and minimal jitter. The servo motor's position should match the required angles within ±2 degrees of the target position.

- The time for the servo to reach the target angle should be under 1 second (depending on the servo's mechanical speed), ensuring quick feedback and response.

5. Power Consumption:

- The system should be power-efficient, with the servo motor and ultrasonic sensor only consuming power when necessary.

- The overall power consumption should be kept to a minimum during idle times. For example:
    - The servo motor should only be powered when moving or holding a position (not unnecessarily powered during idle periods).
    - The ultrasonic sensor should only be triggered to measure distance once every 1 second, reducing its continuous power consumption.

6. System Resource Usage:

- The system should run efficiently on the hardware (e.g., Arduino Uno), using minimal memory and processing power.
- The RAM usage should be kept under 80% of available memory for smooth operation, allowing the system to perform its tasks without memory overflow or crashes.
- Processing Power: The code should run smoothly, using the CPU cycle efficiently to avoid overloading the microcontroller, especially during multiple servo movements or when recalculating distances in quick succession.

7. Stability and Reliability under Load:

- The system should be stable, with no system crashes or unexpected behaviors when running continuously for long periods. It should operate for several hours without failure.
- The system must also handle occasional sensor inaccuracies (e.g., incorrect readings due to noise or interference) without crashing or becoming unresponsive.

8. Environmental Tolerance:

- The ultrasonic sensor should perform reliably in a typical environment with moderate temperature and humidity variations. It should be able to function within a range of -20°C to 60°C without significant performance degradation.
- The system should be resilient to short-term environmental disturbances, such as changes in ambient light or minor electrical fluctuations, without causing major errors in distance measurement or servo control.

9. Response to Errors:

- If an invalid or out-of-range sensor reading occurs (e.g., no object detected or reading exceeds the sensor range), the system should process the error and recover without crashing.
    - The system should handle errors by either using the last valid measurement or skipping the erroneous reading and proceeding with the normal flow.

10. Error Recovery Time:

- The system should be able to recover from any minor errors (e.g., temporary sensor malfunction or disturbance) within 1 second and continue its normal operation without user intervention.

These performance requirements ensure the system is responsive, reliable, and efficient in handling its tasks.

2.3 External Interface Requirement

External interface requirements define how the system interacts with external components, devices, or other systems. These interactions are critical to ensure that the system communicates effectively with sensors, actuators, and possibly other subsystems. Below are the **external interface requirements** for the provided **auto water dispenser system** using the **ultrasonic sensor** and **servo motor**.

1. Hardware Interfaces

Ultrasonic Sensor (HC-SR04)

- **Signal Pins**:
  - ➢ **Trigger Pin**: The ultrasonic sensor has a **Trigger Pin** (connected to Arduino pin 10 in the code). The pin will be set to **HIGH** for 10 microseconds to initiate a measurement pulse.
  - ➢ **Echo Pin**: The **Echo Pin** (connected to Arduino pin 11 in the code) will receive the returned signal after the pulse is sent out, providing the time taken for the pulse to travel.
- **Operating Voltage**:
  - ➢ The sensor operates on a voltage of **5V** (or 3.3V depending on the microcontroller used). The system must supply the appropriate voltage to ensure accurate measurements.
- **Communication Protocol**:
  - ➢ The ultrasonic sensor uses simple **digital signals** (HIGH and LOW) for triggering and measuring the echo. It does **not use serial communication** and relies on **pulse width measurement** to determine distance.
- **Distance Measurement**:
  - ➢ The sensor sends out a pulse and waits for the echo to return. The duration of this pulse determines the distance to the object. The system needs to handle the timing and compute the distance based on this pulse.

Servo Motor

- **Signal Pin**:
  - ➢ The servo is connected to **pin 9** of the Arduino. The servo motor will receive a **PWM signal** (pulse-width modulation) from the Arduino to adjust its angle.
- **Power Requirements**:
  - ➢ The servo motor typically requires a supply voltage of **5V**, which is provided by the

Arduino or an external power source.

- It should also handle the current requirements, typically ranging between **100mA - 500mA**, depending on the servo's size and load.

- **Control Interface**:
  - The servo motor is controlled using PWM signals. The servo will move to specific angles (e.g., 0°, 130°) based on the pulse width received. The system must ensure that the pulse width is correctly calculated and transmitted to the servo to control its position.

2. Software Interfaces

Arduino IDE (Integrated Development Environment)

- **Software Platform**: The system is developed in the **Arduino IDE**, where the code is written, compiled, and uploaded to an Arduino microcontroller.
- **Programming Language**: The code is written in **C/C++** for Arduino. The system must interface with the Arduino runtime environment for compiling and uploading the code.
- **Libraries**:
  - The **Servo library** is used to control the servo motor. It simplifies the task of controlling the servo's position by sending the correct PWM signal.
  - The **PulseIn function** is used to measure the time taken for the echo pulse from the ultrasonic sensor, which is then used to compute the distance.
- **Serial Monitor**:
  - The system may use the **Arduino serial monitor** to display debug messages, such as the measured distance, to assist in troubleshooting or monitoring the sensor's readings. This can be done by sending the distance data over the **USB serial interface**.
- **Serial Communication**:
  - If there is a need to integrate the system with external devices (such as a computer or another embedded system), the system can use **UART (Universal Asynchronous Receiver-Transmitter)** over the USB interface to communicate data, such as status or error messages.

3. Power Supply Interface

Power Supply to Arduino:

- **Voltage**: The Arduino requires a power supply of **5V** (or 9-12V if using an external adapter). The system should ensure that the Arduino receives a stable power supply, either from USB or an external power source (e.g., a battery or power adapter).

Servo and Sensor Power:

- **Power to Servo**: The servo motor should also be powered from the same **5V** supply as the Arduino (or via a dedicated power supply if the servo draws significant current).
- **Power to Ultrasonic Sensor**: The ultrasonic sensor operates on **5V** and typically draws very little current (around **15mA** during operation).

4. Environmental Interface

Temperature and Humidity Tolerance:

- The **ultrasonic sensor** typically operates well within a temperature range of **-20°C to 60°C** and humidity levels of **95%** (non-condensing). The system should ensure that the components, including the sensor and servo, remain within this range for accurate performance.

Physical Placement:

- The ultrasonic sensor must be placed in a location where it has an unobstructed view of the area to measure distances. Any obstruction or irregular surface might cause inaccurate measurements.
- The servo should be installed in such a way that it can move freely to its desired angles (0° and 130°) without hitting physical barriers or constraints.

These external interface requirements ensure that the system can communicate and interact with its physical components (sensor, servo) and other external systems (like the Arduino IDE for programming).

Chapter 3

SYSTEM ANALYSIS AND DESIGNS

3.1 Existing System

Here’s a normal description of the three types of water dispensers, elaborating on their features and limitations:

1. Manual Dispensers

Manual water dispensers are the most basic type of system. Users operate them by manually pressing a lever, turning a knob, or using another form of direct interaction to release water.

- **Operation**: The user has to physically operate the dispenser to release water. It’s simple and doesn’t require any sensors or technology.
- **Contamination Risk**: Since the user directly interacts with the dispenser, there's a higher chance of transferring germs or bacteria to the water dispensing area. This can be a concern, especially in public places.
- **Water Wastage**: There’s no automatic control of water flow, so users may accidentally waste water by overfilling a container or leaving water running unnecessarily.

2. Traditional Automatic Dispensers

Traditional automatic dispensers are commonly used in public areas, offering sensor-based, hands-free operation for water dispensing.

- **Operation**: These dispensers detect the presence of a container or hand under the nozzle using a sensor, dispensing water automatically. Users don’t need to touch the dispenser, making it more hygienic than manual dispensers.
- **No Flow Control**: Traditional automatic dispensers often don’t have a way to control the amount of water dispensed. Once the sensor is triggered, water continues to flow for a fixed duration, which can lead to water wastage if the container is already full.
- **Basic Functionality**: These dispensers are designed for simple water dispensing, making them ideal for high-traffic public spaces, but they might lack advanced features like volume control or water level monitoring.

3. Bottle-Based Dispensers

Bottle-based dispensers use large water bottles that need to be replaced when empty. These are commonly used in homes, offices, or locations without direct plumbing access.

- **Operation**: The water is stored in large bottles, typically 5 to 20 liters. The bottle is either manually or automatically placed into the dispenser, and the water is dispensed from the bottle.

- **Refill Requirement**: Since these dispensers rely on bottled water, they require regular refills once the water runs out. This can lead to interruptions in service, especially in high-traffic environments.
- **Contamination Risk**: If the bottles aren’t sealed or stored properly, there’s a risk of contamination. Additionally, improper handling during bottle replacement can lead to bacteria or dirt getting into the water.
- **Interruptions in Service**: When the bottle runs out, users must replace it with a new one. This can cause delays or service interruptions, especially in busy locations.

Each of these water dispenser types has its pros and cons. **Manual dispensers** are simple and cost-effective but have hygiene concerns and potential inefficiency. **Traditional automatic dispensers** offer hands-free convenience but lack precise control over the water flow. **Bottle-based dispensers** provide portability but require frequent refills and introduce contamination risks.

3.2 Proposed System

Here’s an elaboration of the features Touchless Operation and Smart Water Flow Control for water dispensers:

1. Touchless Operation

Touchless operation refers to the use of sensors that allow the user to dispense water without needing to physically touch the dispenser. This enhances convenience and hygiene by eliminating the need for manual interaction.

- Sensors for Hands-Free Water Dispensing:
  - ➢ Description: Touchless water dispensers are equipped with sensors (such as infrared or ultrasonic sensors) that detect the presence of a hand or container near the spout. When the sensor is activated, water is dispensed automatically without the user needing to press a button or handle any part of the dispenser. This process is seamless and requires no physical contact.
  - ➢ Benefits: This technology ensures that the user does not come into contact with potentially contaminated surfaces. It's particularly useful in high-traffic areas such as public restrooms, offices, hospitals, or schools, where minimizing contact with shared surfaces is a priority to prevent the spread of germs and bacteria.
- Reduces Contact, Ensuring Hygiene:
  - ➢ Description: By eliminating physical interaction with the dispenser, touchless operation significantly reduces the chances of transferring germs, bacteria, or viruses

from one person to another. Users simply need to place their container under the spout, and the system dispenses water automatically.

  - ➢ Benefits: This results in improved hygiene, making it especially suitable in environments where cleanliness and public health are important, such as in medical facilities, foodservice areas, and other public spaces. Reducing the need for contact also makes the system more user-friendly and accessible for people of all ages, including children and the elderly.

2. Smart Water Flow Control

Smart water flow control is an advanced feature that allows the dispenser to control the rate of water dispensed, offering more precise and efficient usage of water.

- Adjustable Flow Rates to Minimize Waste:
  - ➢ Description: Water dispensers with smart flow control systems can be programmed to adjust the flow rate based on the user's needs or the container size. For example, the dispenser can automatically adjust the flow to fill a small cup quickly or dispense a controlled amount into a large bottle. This system can prevent excessive water from being dispensed when less is needed.
  - ➢ Benefits: This functionality helps to minimize water wastage, making it more eco-friendly and cost-efficient. It is especially important in areas where water conservation is critical. The ability to adjust the flow ensures that the system is versatile and can accommodate different container sizes or user preferences.
- Automated Shut-Off to Prevent Overflow:
  - ➢ Description: An automated shut-off feature ensures that once the desired amount of water is dispensed, the system will stop automatically. This is achieved through sensors that detect when a container is full, preventing any overflow. The water dispenser can monitor the container's water level in real time, and as soon as the container is full, the flow will stop.
  - ➢ Benefits: This reduces the risk of water wastage due to overflow, improving the overall efficiency of the system. It also saves users from having to monitor the process manually and ensures that the water dispensing is accurate and precise. This feature is especially beneficial in environments where large volumes of water are dispensed frequently, as it helps avoid spills and mess.
- Touchless operation and smart water flow control are two advanced features that make water dispensers more efficient, hygienic, and user-friendly.
  - ➢ Touchless operation reduces the need for physical contact, which is crucial for

maintaining cleanliness and hygiene, especially in public spaces.

- Smart water flow control helps minimize waste by offering adjustable flow rates and an automated shut-off feature, ensuring water is dispensed efficiently and accurately without unnecessary spillage.

These features not only improve user experience but also contribute to better environmental sustainability by reducing water wastage and improving hygiene standards.

3.3 Use Case

A **use case** describes a specific scenario in which a user interacts with a system or product to achieve a particular goal. In the context of the **Arduino Uno**, a use case can describe how an end user might program and interact with the board to achieve a specific objective.

Example Use Case: Water Dispenser Control System using Arduino Uno

Use Case Title: Automated Water Dispenser Control System

Primary Actor: User (or System Administrator)

Goal: To control and monitor the dispensing of water through the use of sensors and actuators connected to the Arduino Uno.

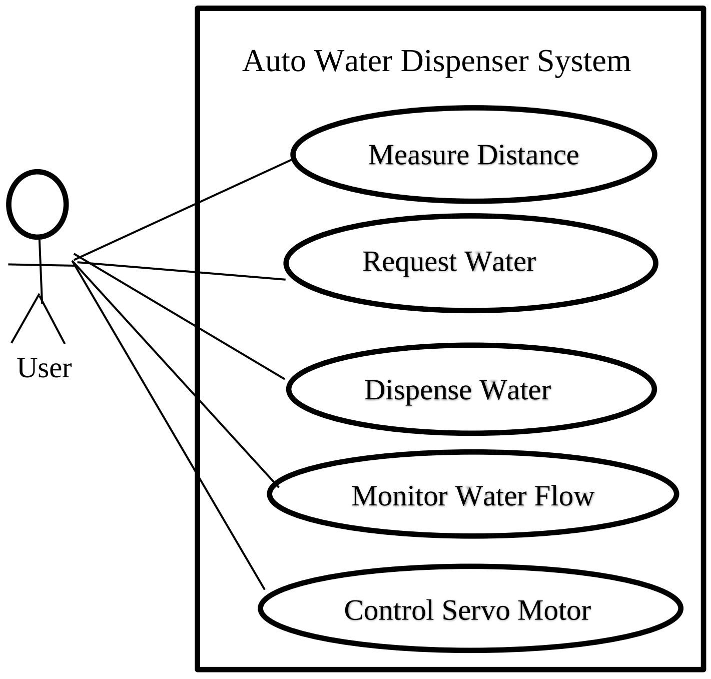

Fig.3.3: Use Case

3.4 Data Flow Diagram

A **Data Flow Diagram (DFD)** is a visual representation of how data flows through a system. It shows the various processes, data stores, and data flows between entities. In the context of a **Water Dispenser System**, the DFD can illustrate how data is processed when the dispenser operates.
Here's a simplified **Data Flow Diagram** for a water dispenser:

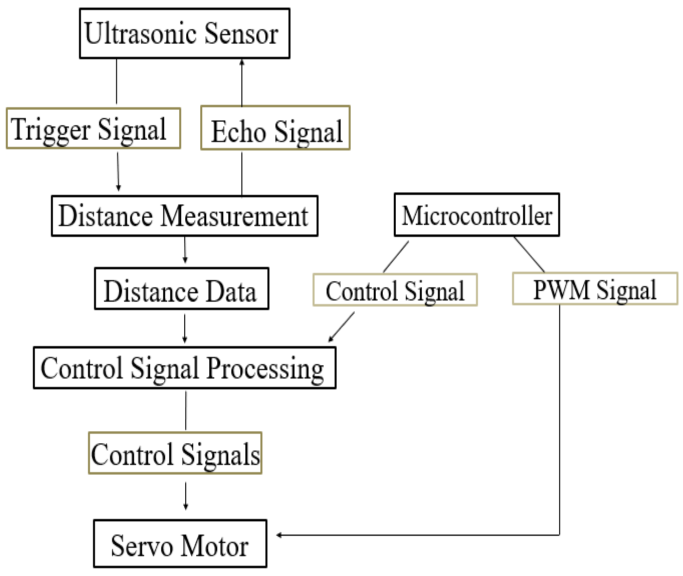

Fig.3.4: Data Flow Diagram

3.5 Architectural Design

3.5.1 System Architecture Design

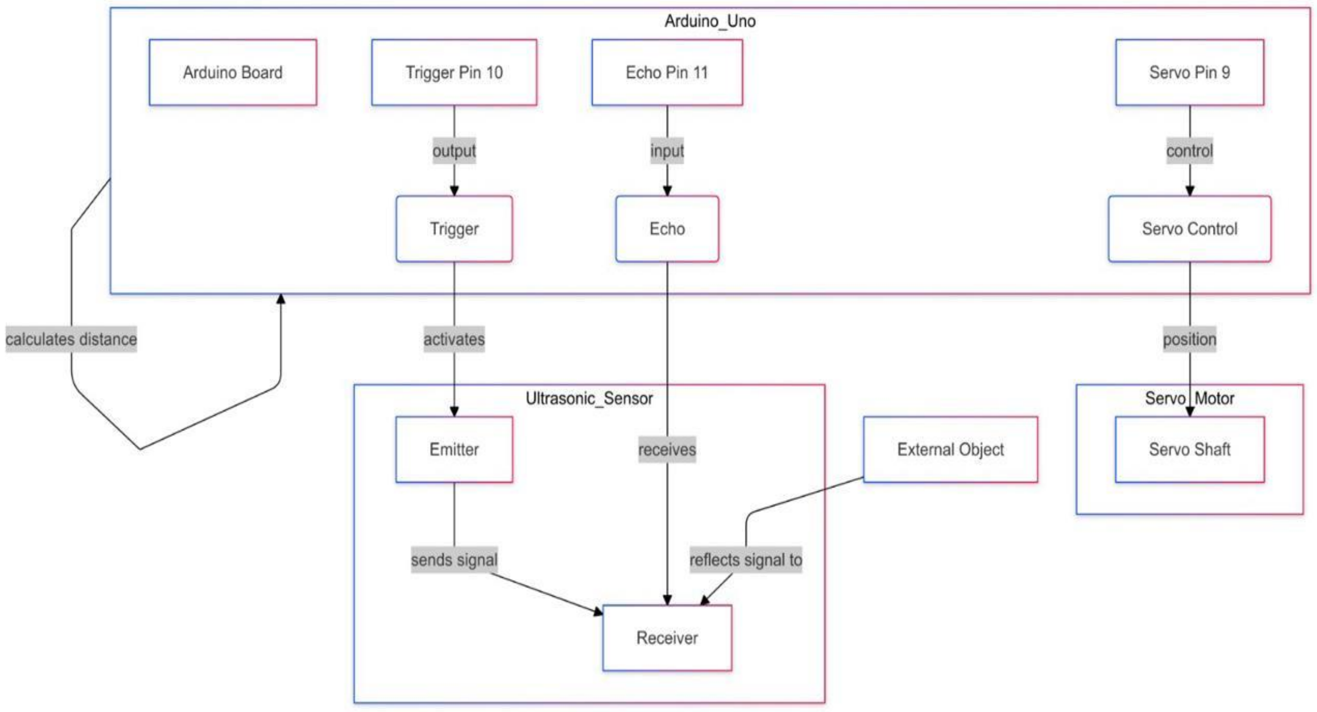

Fig.3.5.1: System architectural design

3.5.2 Description of Architecture Design

1. System Components

The system consists of the following major components:

- Arduino (Microcontroller)
- Ultrasonic Sensor (HC-SR04)
- Servo Motor
- Power Supply

2. Functional Description of Each Component

1. Arduino (Microcontroller):

- The Arduino board serves as the central controller of the system.
- It processes the input from the ultrasonic sensor and determines how to control the servo motor.
- The Arduino runs the code to:

- Trigger the ultrasonic sensor.
- Calculate the distance based on the time of flight of the ultrasonic pulse.
- Send control signals to the servo motor to adjust its position based on the measured distance.

2. Ultrasonic Sensor (HC-SR04):

- The HC-SR04 sensor consists of two main components: the Trigger Pin and the Echo Pin.
- The Trigger Pin is used to send a signal (pulse) to initiate the emission of ultrasonic waves.
- The Echo Pin detects the reflected ultrasonic waves and calculates the time taken for the waves to return.
- The sensor measures the time it takes for the signal to travel to the object and back, which is then converted into a distance by the Arduino.

3. Servo Motor:

- The Servo Motor is used to perform physical motion based on the commands from the Arduino.
- The servo can rotate between 0° and 180°, controlled via a PWM signal sent by the Arduino.
- Based on the distance measured by the ultrasonic sensor, the Arduino sends a PWM signal to adjust the servo's position.

4. Power Supply:

- A power source is required to power both the Arduino and the servo motor. This can be a USB connection (for the Arduino) or an external battery (for both Arduino and servo).
- The servo may require more current than the Arduino can supply through its pins, so an external power supply might be needed for the servo to ensure proper functioning.

3. Data Flow and Control Flow

The architecture follows a control flow where the Arduino receives inputs, processes the information, and then controls the output devices (servo). Here's a detailed breakdown of the process:

Step 1: Ultrasonic Sensor Interactio

- Triggering the Ultrasonic Pulse:
  - The Arduino sends a short pulse (10 microseconds) to the Trigger Pin of the ultrasonic sensor (pin 10).
  - This pulse triggers the sensor to emit ultrasonic waves.
- Echo Reception:
  - The ultrasonic sensor listens for the reflected waves using the Echo Pin (pin 11).
  - The Arduino measures the time (pulseIn()) it takes for the echo to return from the object.

- Distance Calculation:
    - ➢ The Arduino uses the formula distance = (duration / 58.82) to convert the duration (in microseconds) to a distance in centimeters.

Step 2: Decision Making Based on Distance

- Once the distance is calculated, the Arduino decides the servo motor's position based on a threshold:
    - ➢ If distance < 6 cm, the Arduino commands the servo to move to 130 degrees (center position).
    - ➢ If distance ≥ 6 cm, the servo is commanded to move to 0 degrees (initial position).

Step 3: Servo Motor Control

- The Arduino sends the appropriate PWM signal to the servo motor:
    - ➢ Servo.write(90) moves the servo to the center position (130 degrees).
    - ➢ Servo.write(0) moves the servo to the initial position (0 degrees).

Step 4: Loop and Repeat

- After each servo movement, the Arduino pauses briefly using a delay() to prevent overloading the servo motor and provides time for the system to stabilize.
- The system then repeats the process by continuously measuring the distance and adjusting the servo position accordingly.

4. Block Diagram

Here’s a simplified block diagram of the system architecture:

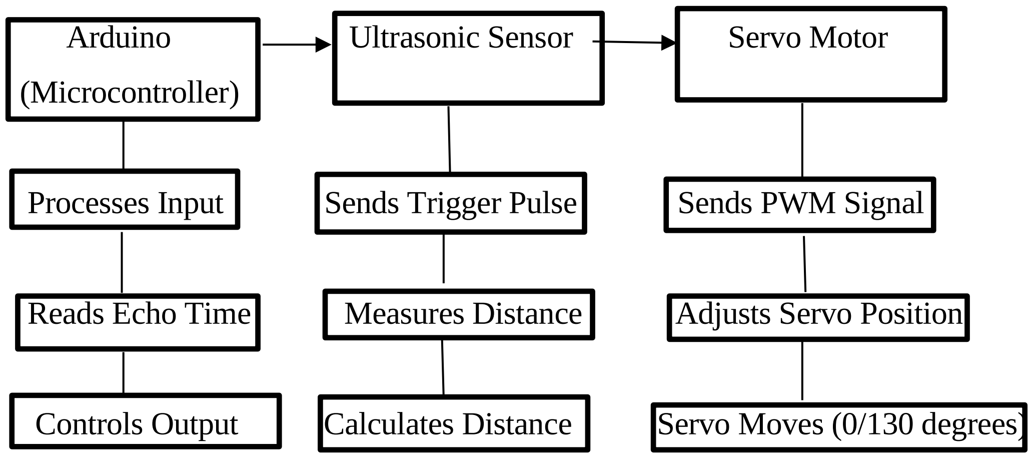

Fig.3.5.2: Block Diagram

5. Flowchart for System Operation

1. Start: Initialize the system.
2. Trigger Ultrasonic Pulse: Send a pulse to the ultrasonic sensor to emit waves.
3. Measure Echo Time: Capture the time it takes for the waves to return after bouncing off an object.
4. Calculate Distance: Use the duration to calculate the distance using the formula (duration / 58.82).
5. Decision:
    - If the distance is less than 6 cm, move the servo to 130 degrees.
    - If the distance is greater than or equal to 6 cm, move the servo to 0 degrees.
6. Repeat: Wait briefly (e.g., 50 milliseconds) before starting the next iteration.

6. Timing Diagram

- Ultrasonic Pulse Emission: A short pulse (10 µs) is sent to the Trigger Pin.
- Echo Time: The duration it takes for the echo to return is measured by the Arduino and used to calculate the distance.
- Servo Action: Based on the distance, the servo motor moves to either 0 degrees or 130 degrees.

7. Advantages of the Architecture

- Modularity: The design is modular, with distinct components (sensor, microcontroller, servo) performing individual tasks, making the system easy to modify or expand.
- Low-Cost: The system uses common, affordable components such as the Arduino, ultrasonic sensor, and servo motor.
- Real-time Control: The system provides real-time feedback by measuring distance and responding immediately by adjusting the servo.
- Scalability: This architecture can be scaled up to control multiple sensors or motors, with slight modifications to the code and wiring.

8. Limitations and Future Enhancements

- Distance Range: The system is limited by the sensor’s range and accuracy (typically 2 cm to 400 cm). It could be enhanced by using more accurate or long-range sensors for specific applications.
- Servo Response Time: The servo’s reaction time can be slow due to the delay() function. Implementing non-blocking code, like millis(), could improve responsiveness.
- Power Supply: The servo might draw more power than the Arduino can supply, requiring an external power source.

- Multiple Sensors: The system could be expanded to control multiple servo motors or use multiple ultrasonic sensors for more complex control.

The architecture for this system is simple yet effective for a variety of basic robotics or automation applications. It is designed for ease of use and can be easily extended by adding additional sensors or control mechanisms. The Arduino microcontroller plays a crucial role as the "brain" of the system, processing sensor input and controlling outputs based on pre-defined conditions.

Chapter 4
IMPLEMENTATION

4.1 Algorithm

Here is the algorithm for the **auto water dispenser** with **ultrasonic sensor** and **servo motor**:

1. **Initialize Components**:
   - Attach the **servo motor** to pin **9**.
   - Set **trigpin** to pin **10** and **echopin** to pin **11**.
   - Set **trigpin** as **OUTPUT** and **echopin** as **INPUT**.
2. **Send Ultrasonic Pulse**:
   - Set **trigpin** LOW briefly (2 milliseconds) to clear any previous signals.
   - Set **trigpin** HIGH for **10 microseconds** to trigger the ultrasonic sensor.
   - Set **trigpin** LOW to stop the pulse.
3. **Measure Echo Duration**:
   - Use the pulseIn() function to measure the duration for the echo signal to return to **echopin**.
4. **Calculate Distance**:
   - Convert the measured **duration** to **distance** in centimeters using the formula:

$$\text{Distance} = \frac{58.82}{\text{Duration}}$$

5. **Check Distance**:
   - If **distance** is less than **6 cm**, proceed to the next step for dispensing.
   - If **distance** is greater than **6 cm**, go to step 7.
6. **Activate Servo for Dispensing**:
   - Move the **servo motor** to **130 degrees** (center position) to start dispensing water.
   - **Delay** for **7 seconds** to allow enough time for water dispensing.
7. **Return Servo to Initial Position**:
   - Move the **servo motor** back to **0 degrees** (initial position).
   - Repeat the loop to continue checking the distance and controlling the servo accordingly.

Explanation:

- **Servo Motor**: The servo motor controls the water flow, moving to a 130-degree position when an object (such as a cup or bottle) is detected within 6 cm of the ultrasonic sensor.

After dispensing water for a set time (7 seconds), it returns to the initial 0-degree position.

- **Ultrasonic Sensor**: The sensor measures the distance to the object. When the distance is less than 6 cm, it triggers the servo to dispense water.
- **Delay**: The 7-second delay ensures the servo stays in the dispensing position long enough to pour the water.

This simple algorithm allows for a basic **automatic water dispensing** system that uses an ultrasonic sensor to detect the presence of a container and a servo motor to control the flow of water.

4.2 Source code

```
#include <Servo.h>
Servo servoMain;
int trigpin = 10;
int echopin = 11;
int distance;
float duration;
float cm;

void setup()
{
   servoMain.attach(9);
    pinMode(trigpin, OUTPUT);
    pinMode(echopin, INPUT);
}

void loop()
{
  digitalWrite(trigpin, LOW);
  delay(2);
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
  duration = pulseIn(echopin, HIGH);
  cm = (duration/58.82);
  distance = cm;

  if(distance<6)

 {
    servoMain.write(130);  // Turn Servo back to center position (130 degrees)
   delay(7000);
  }
else
{
    servoMain.write(0);
    delay(50);
  }

}
```

Chapter 6

RESULTS

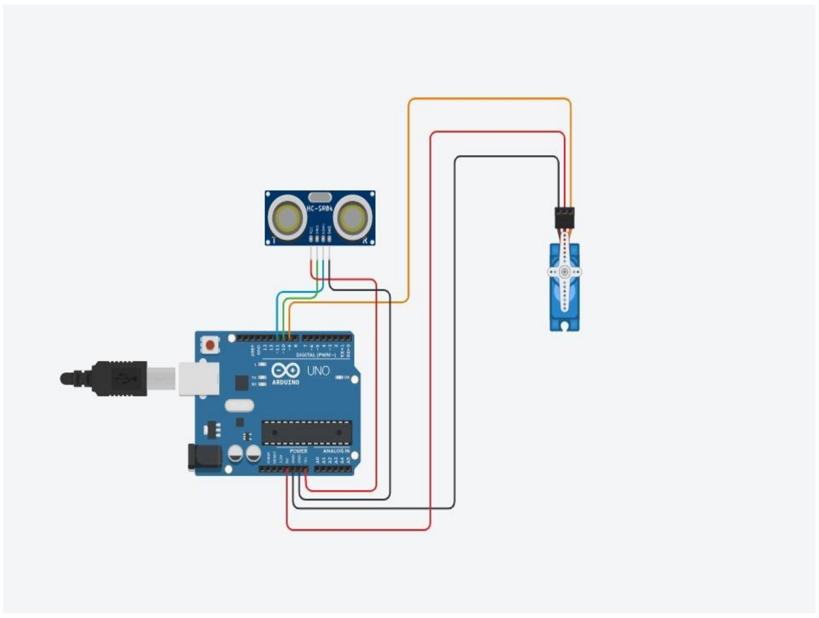

Fig.6.1: Circuit Connection

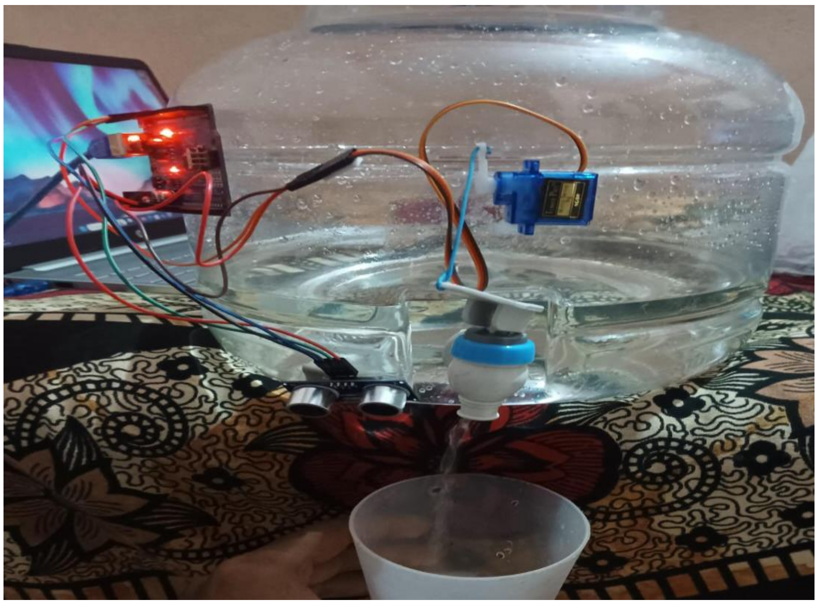

Fig.6.2: Working

CONCLUSION

The **Auto Water Dispenser System** utilizing an **ultrasonic sensor** and a **servo motor** provides an efficient, hands-free solution for dispensing water. By detecting the presence of a container using the ultrasonic sensor, the system automatically activates the servo to release water. The key benefits and conclusions from the system are:

1. **Hygienic Operation**: The touchless nature of the system minimizes physical contact, reducing the risk of contamination and ensuring a higher level of hygiene. This is particularly important in public places or environments where cleanliness is a priority.
2. **Automatic Water Dispensing**: The system detects the presence of a container and dispenses water only when needed, ensuring that water is not wasted. The smart functionality of the servo motor allows for controlled water release, improving water conservation.
3. **Cost-Efficiency and Environmental Impact**: By preventing overflow and over-dispensing, the system helps save water, making it more environmentally friendly and cost-effective in the long run.
4. **User Convenience**: The automatic and hands-free operation makes the system easy to use for people of all ages, eliminating the need for manual intervention and offering convenience in various settings like offices, schools, and public spaces.
5. **System Reliability**: With the use of an ultrasonic sensor and a servo motor, the system provides a reliable and straightforward solution to automatic water dispensing, reducing human error and making the process more efficient.

In conclusion, the **Auto Water Dispenser System** is a practical, efficient, and hygienic solution that promotes sustainability and user convenience, making it ideal for both private and public use.

REFERENCES

[1] Arduino Servo Library. (n.d.). Retrieved from https://www.arduino.cc/en/Reference/Servo

[2] Bracken, D. (2015). Ultrasonic Distance Measurement with Arduino. Retrieved from https://www.digikey.com/en/maker/projects/ultrasonic-distance-measurement-with-arduino/74668

[3] Bang, A. (2016). Using Ultrasonic Sensors with Arduino. In *Make: Electronics* (pp. 68-72). Maker Media, Inc.

[4] Arduino Documentation. (n.d.). digitalWrite. Retrieved from https://www.arduino.cc/en/Reference/DigitalWrite

[5] Arduino Documentation. (n.d.). pulseIn. Retrieved from https://www.arduino.cc/en/Reference/PulseIn

[6] Bueche, P., & Finneran, M. (2012). *Arduino For Dummies*. Wiley.

[7] Monk, S. (2014). *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education.

[8] Arduino Documentation. (n.d.). Structure of an Arduino Sketch. Retrieved from https://www.arduino.cc/en/Tutorial/Foundations/Structure

[9] Banzi, M., & Shiloh, M. (2014). *Getting Started with Arduino*. Maker Media, Inc.

[10] Wirth, S. (2013). Understanding Ultrasonic Sensors. *Adafruit Learning System*. Retrieved from https://learn.adafruit.com/ultrasonic-sonar-distance-sensors