PROJECT REPORT

ON

"DIY Digital Level Indicator with PID Control!"

THIS PROJECT & RESEARCH REPORT HAS BEEN INDEPENDENTLY DEVELOPED BY ME AND IS NOT SUBMITTED TO ANY COLLEGE OR UNIVERSITY FOR ACADEMIC EVALUATION

Developed by

Name:-Suzen Kumar Mohanty (**Reg.No**.2421331063)

SKMP PROJECT MANAGEMENT

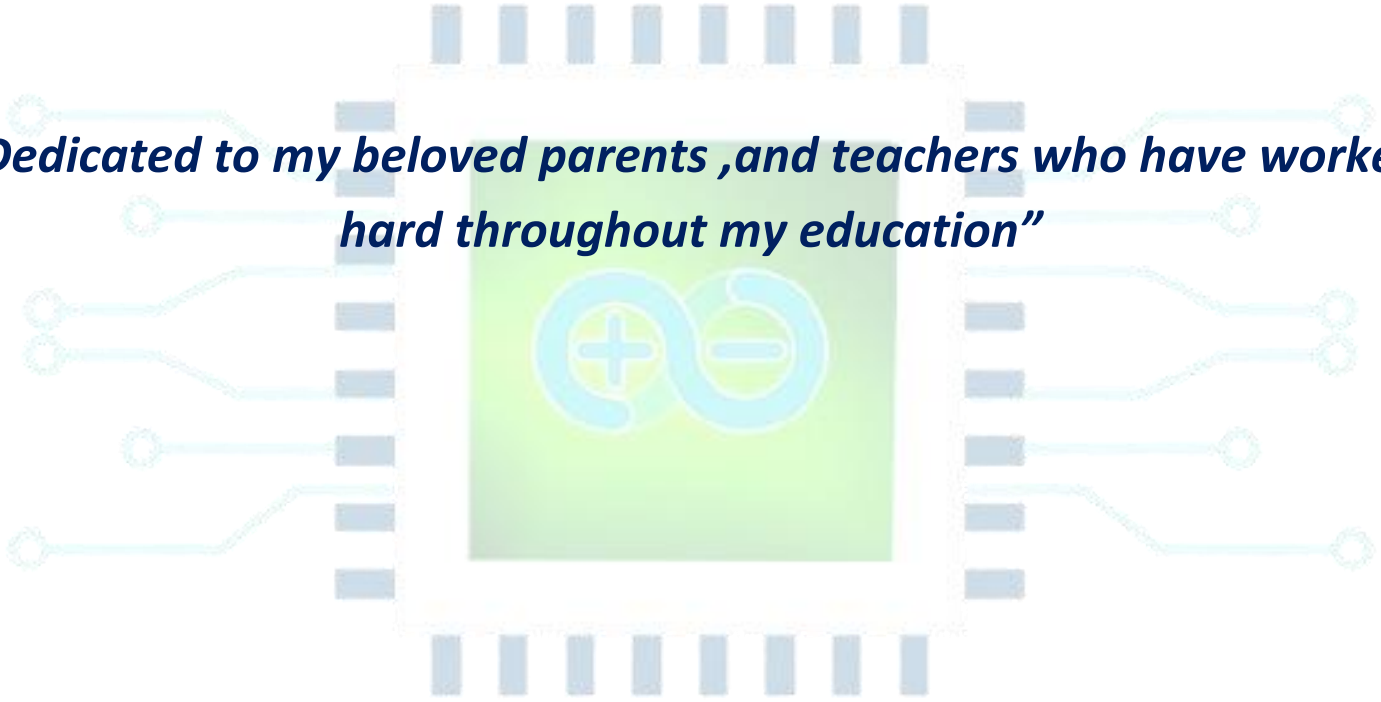
SKMP PROJECT MANAGEMENT

DEPARTMENT OF IOT

POLISHLINE.BALASORE, ODISHA, INDIA

JUNE-2026

“Dedicated to my beloved parents ,and teachers who have worked hard throughout my education”



SKMP PROJECT MANAGEMENT

CERTIFICATE

This is to certify that the project entitled" DIY Digital Level Indicator with PID Control"

Has been independently developed and completed by Suzen Kumar Mohanty for the purpose of enhancing technical knowledge, skills, and practical understanding.

This project is a self-initiated work carried out with person al interestand research. It has not been submitted to any college, university, or institution for the award of any degree or diploma.

The work presented in this project is genuine and has been completed by the developer himself.



Suzen Kumar Mohanty

(Project Developer)

Date:15/06/2026

Place:-Balasore,Odisha,India

SELF ATTESTATION

I here by declare that I have personally worked on the project entitled "**DIY Digital Level Indicator with PID Control ".**

All the data, analysis, and implementation presented in this project are the result of my own efforts and genuine work .Any external sources of information, if used, have been properly acknowledged.

This project is created solely for learning , skill development ,and professional improvement, and has not been submitted to any institution for academic purposes.



SKMP PROJECT MANAGEMENT

Suzen Kumar Mohanty

(Project Developer)

Date: 15/06/2026

Place:Balasore,Odisha,India

DECLARATION

I hereby declare that the project titled "**DIY Digital Level Indicator with PID Control**"is an original work carried out by me based on my own idea, research, and implementation.

This project that has not been submitted previously ,in part or in full, to any college, university, or institution for the award of any degree, diploma, or certification.

I confirm that all the information presented in this project is true to the best of my knowledge. Any references or sources used have been properly acknowledged.

I take full responsibility for the authenticity and originality of this work.

Name: Mr. Suzen Kumar Mohanty

Branch: Computer Science And Engineering

Institution:[Srinix Collage Of Engineering ,Independent]

Date:[15/06/2026]

My PortPolio Link:-https://suzen-kumar-mohanty-8fdv.vercel.app/

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to everyone who contributed, directly or indirectly, to the development of this project, "DIY Digital Level Indicator with PID Control."

This project is the result of my personal interest, dedication, continuous learning, and exploration of electronics, automation, sensor technology, and control systems.

I am thankful to the developers of Arduino, the creators of open-source hardware and software resources, and the technical communities whose documentation, tutorials, and knowledge-sharing platforms helped me understand and implement the concepts used in this project.

I would also like to acknowledge the importance of self-learning, practical experimentation, and problem-solving throughout the development process. These experiences helped me strengthen my technical knowledge and improve my understanding of real-world automation systems.

Finally, I am grateful to everyone who supports innovation, DIY engineering, and independent technical projects. This work reflects my enthusiasm for learning, creating, and exploring new possibilities in technology.

Developed By:

Suzen Kumar Mohanty

ABOUT MYSELF



I am Mr. Suzen Kumar Mohanty , a passionate and dedicated Web Developer and SOC Analyst (Level 1) from the field of Computer Science and Engineering, with strong expertise in HTML, CSS, JavaScript, and I specialize in building scalable, responsive, and high-performance web applications that deliver seamless user experiences across all devices and screen sizes.

With a keen eye for detail, I focus on writing clean, maintainable, and well-structured code. My development approach is centered around Creating intuitive user interfaces, ensuring both functionality and aesthetic quality. I follow modern front-end development practices such as component-based architecture, responsive design principles, and performance optimization techniques to build efficient and user-friendly applications.

Beyond web development, I actively work on IoT-based solutions, where I integrate hardware and software to create smart, real-world systems. I have hands-on experience with Arduino Uno and various sensors, enabling me to develop projects such as environmental monitoring systems, smart automation solutions, and real-time data tracking applications.

One of my key strengths lies in connecting IoT devices with web technologies. I build interactive dashboards that allow real-time data visualization, giving users the ability to monitor, analyze, and control devices remotely. My work focuses on bridging the gap between the physical and digital world by transforming raw sensor data into meaningful insights.

I am deeply interested in developing innovative, data-driven solutions that combine web technologies with embedded systems. My goal is to create impactful applications that solve real-world problems while improving efficiency, automation, and user experience.

With a strong passion for continuous learning and innovation, I consistently explore new tools, frameworks, and technologies to enhance my skills and deliver high-quality, future-ready solutions.

Name: Mr. Suzen Kumar Mohanty

Branch: Computer Science And Engineering

MyPortPolioLink:-https://suzen-kumar-mohanty-8fdv.vercel.app/

ABSTRACT

This project presents the design, development, and implementation of a **DIY Digital Level Indicator with PID Control** for automatic liquid-level monitoring and regulation in a tank. The system uses an ultrasonic distance sensor to measure the distance between the sensor and the liquid surface. The microcontroller converts this distance into the actual liquid level and displays it on an LCD or OLED display.

The main feature of the project is the use of a **Proportional–Integral–Derivative (PID) controller** to maintain the liquid level at a user-defined setpoint. Instead of switching the pump fully ON or OFF, the PID controller continuously calculates the error between the desired level and measured level. It then adjusts the pump speed or valve opening using a PWM signal. This results in smoother control, reduced overshoot, better accuracy, and improved system stability.

The project can be applied to water tanks, irrigation systems, chemical storage tanks, laboratory equipment, food-processing tanks, and small industrial automation systems. The system also includes safety features such as dry-run protection, low-level alarm, high-level alarm, and sensor-fault detection.

The project demonstrates the practical integration of sensors, microcontrollers, actuators, embedded programming, closed-loop control theory, and human-machine interface design.

SKMP PROJECT MANAGEMENT

Keywords: Digital Level Indicator, Liquid-Level Control, PID Controller, Ultrasonic Sensor, Arduino, ESP32, PWM, Automation, Embedded System, Water Tank.

LIST OF FIGURES

Figure No.	Figure Title	Page No.
01	Digital level indicator with Pid Control fig	10
02	Arduino Uno Board	17
03	HC-SR04 Ultrasonic Sensor	17
04	Servo Motor	18
05	Jumper Wire	19
06	Usb Cable	20
07	Power Supply (9V Battery With Connector)	21
08	Circuit Diagram	22

SKMP PROJECT MANAGEMENT
SKMP PROJECT MANAGEMENT

INDEX

Sl. No.	Contents / Chapter Headings	Page No.
1	Literature Review	13
2	5.1 Liquid-Level Measurement Methods	13
3	5.2 Ultrasonic Level Measurement	14
4	5.3 PID Control	14
5	5.4 PID Implementation on Arduino	15
6	Hardware Requirements	16
7	Arduino UNO Board	16
8	HC-SR04 Ultrasonic Sensor	17
9	Servo Motor (SG90)	18
10	Jumper Wires	18
11	USB Cable	19
12	Power Supply (9V Battery with Connector)	20
13	10.4 Relay-Based Control	24
14	11. Working Principle	24
15	11.1 Level Measurement	24
16	11.2 Setpoint Comparison	25
17	11.3 PID Control Action	25
18	11.4 Control Behaviour	26
19	12. PID Control Theory	26
20	12.1 Closed-Loop Control	26
21	12.2 PID Equation	27
22	12.3 Effect of PID Parameters	27
23	12.4 PID Tuning	27

SKMP PROJECT MANAGEMENT

24	13. System Algorithm	28
25	13.1 Main Algorithm	28
26	13.2 Flowchart	29
27	14. Arduino Program Code	30
28	15.1 Firmware Limitations	32
29	16. Assembly and Commissioning Procedure	33
30	17. PID Tuning Procedure	34
31	17.1 Begin with Proportional and Derivative Control	34
32	17.2 PID Parameter Adjustment	35
33	17.3 Integral Control Adjustment	35
34	17.4 System Response Observation	35
35	17.5 Final PID Tuning	35
36	18. Testing and Validation Plan	36
37	18.1 Performance Metrics	36
38	19. Results Reporting Template	37
39	20. Troubleshooting	38
40	21. Advantages	39
41	22. Limitations	39
42	22.1 Sensor Limitations	39
43	22.2 Mechanical Limitations	39
44	22.3 Actuator Limitations	40
45	22.4 Model Limitations	40
46	22.5 Evidence Limitations	40
47	23. Applications	40
48	25. Future Improvements	41
49	26. Budget Planning	41
50	27. Safety Precautions	42
51	28. Publication Notes	42

SKMP PROJECT MANAGEMENT
SKMP PROJECT MANAGEMENT

52	Last Chapter – Conclusion & Future Scope	**44**
53	Future Scope	**45–46**
54	References	**47–48**
55	Project Link	**49**

SKMP PROJECT MANAGEMENT
SKMP PROJECT MANAGEMENT

1. Introduction

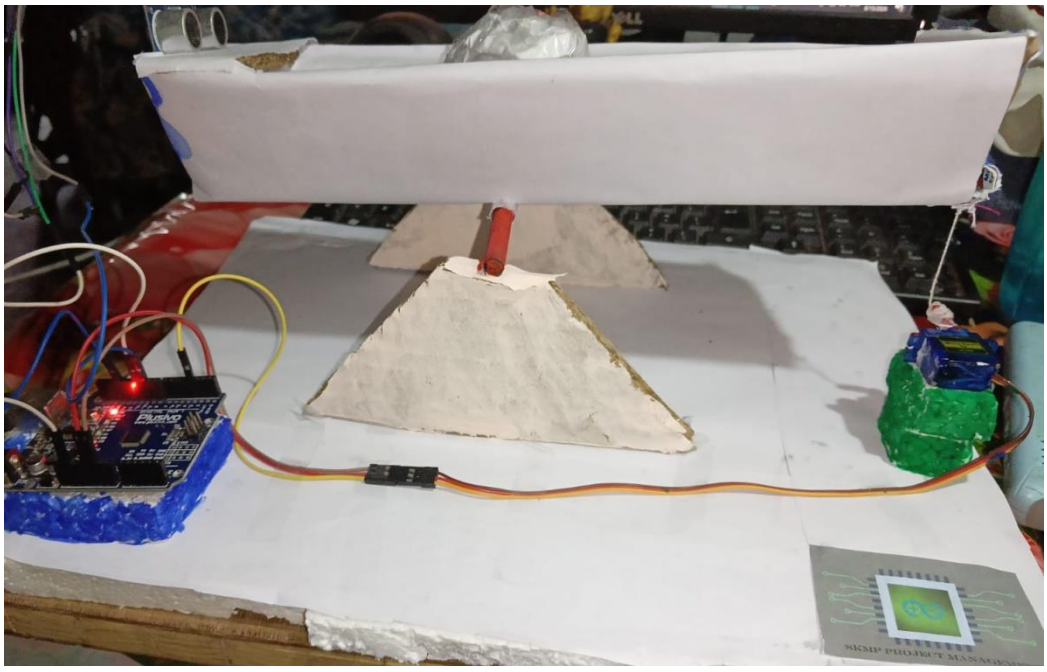


Fig 01

Liquid-level measurement and control are important in many domestic, agricultural, industrial, and laboratory applications. Water tanks, chemical tanks, fuel tanks, boilers, and storage containers require accurate monitoring of liquid level to avoid overflow, shortage, dry running of pumps, and process failure.

In a conventional water-level control system, a pump is usually switched ON when the tank level becomes low and switched OFF when the tank becomes full. This method is simple but has several disadvantages. It may cause frequent pump operation, inaccurate level control, water hammer, overflow, and excessive mechanical stress on the pump.

This project proposes a digital level indicator with automatic PID-based control. An ultrasonic sensor continuously measures the liquid level inside the tank. The measured level is compared with a desired setpoint entered by the

user. Based on the error, the microcontroller calculates the required control action using the PID algorithm.

The PID controller adjusts the output to the pump or valve gradually. For example, if the tank level is much lower than the setpoint, the pump runs at high speed. As the level approaches the setpoint, the pump speed reduces gradually. When the level reaches the setpoint, the pump output becomes stable. This prevents sudden switching and improves control accuracy.

The Arduino PID library is designed to keep an input variable close to a desired setpoint by adjusting an output, and its behaviour can be adjusted using the proportional, integral, and derivative parameters. The HC-SR04 ultrasonic sensor is suitable for non-contact level measurement, with a typical range of 2 cm to 400 cm and accuracy up to 3 mm

2. Problem Statement

In many existing liquid-level control systems, pumps operate using simple ON/OFF control. The pump starts when the liquid level falls below a lower threshold and stops when the level reaches an upper threshold.

This approach creates the following problems:

- Frequent starting and stopping of the pump.
- Overshoot of the desired liquid level.
- Delay in maintaining the required level.
- Poor control accuracy near the setpoint.
- Possible overflow due to delayed pump shutdown.
- Possible dry running of the pump when the tank is empty.
- Increased power consumption and mechanical wear.
- No real-time display of actual liquid level.
- No automatic adjustment based on changing demand.

Therefore, there is a need for a low-cost, reliable, automatic, and accurate liquid-level control system that can continuously monitor the level and adjust the pump output smoothl

3. Objectives

The main objectives of this project are:

- To design a digital liquid-level indicator using an ultrasonic sensor.
- To measure the liquid level in centimetres and percentage.
- To display real-time level, setpoint, and pump status on an LCD or OLED display.
- To implement a PID-based closed-loop control system.
- To automatically control a DC pump, solenoid valve, or motor using PWM.
- To allow the user to set the desired liquid level.
- To reduce overshoot and oscillation in level control.
- To provide dry-run protection and sensor-fault detection.
- To activate alarms during low-level, high-level, or fault conditions.
- To develop a low-cost and scalable embedded-control system.

Scope of the Project

The project is designed for a small-scale water tank or liquid-storage tank. It can be used for educational demonstration, laboratory experiments, home automation, irrigation systems, and small industrial applications.

The system includes:

- 1 x Arduino UNO Board
- 1 x HC-SR04 Ultrasonic Sensor
- 1 x Servo Motor (SG90)

- Jumper Wires
- USB Cable
- Power Supply (9V Battery with connector)
- Cardboard and plastic ball for the demonstrative structure

The project can be expanded for larger tanks by replacing the ultrasonic sensor with a pressure sensor, radar-level sensor, capacitive sensor, or industrial level transmitter

Literature Review

5.1 Liquid-Level Measurement Methods

Liquid level can be measured using different techniques:

Method	Working Principle	Advantages	Limitations
Float switch	Mechanical float moves with liquid level	Simple and low cost	Mechanical wear, limited accuracy
Ultrasonic sensor	Measures echo time of ultrasonic pulse	Non-contact, easy to use	Affected by foam, vapour, and tank shape
Capacitive sensor	Measures change in capacitance	Suitable for non-metallic tanks	Calibration required
Pressure sensor	Measures hydrostatic pressure at tank bottom	Accurate for closed tanks	Requires calibration
Radar sensor	Uses electromagnetic waves	Highly accurate, industrial-grade	Costly
Load cell	Measures tank weight	Suitable for mass measurement	Requires mechanical installation

For this project, an ultrasonic sensor is selected because it is low-cost, non-contact, easy to interface with Arduino, and suitable for demonstration purposes.

5.2 Ultrasonic Level Measurement

The HC-SR04 ultrasonic sensor has an ultrasonic transmitter and receiver. When the trigger pin receives a 10-microsecond HIGH pulse, the module transmits eight cycles of 40 kHz ultrasound. The echo pin remains HIGH until the reflected signal returns. The time between transmission and reception is used to calculate distance.cdn.sparkfun+1

The distance is calculated as:

Distance=Time×Speed of Sound2Distance = \frac{Time \times Speed\ of\ Sound}{2}Distance=2Time×Speed of Sound

At approximately 20°C, the speed of sound is about 343 m/s. The HC-SR04 datasheet provides the practical formula:

Distance (cm)=Duration (µs)58Distance\ (cm) = \frac{Duration\ (\mu s)}{58}Distance (cm)=58Duration (µs)

The sensor is recommended to be used with a measurement cycle of more than 60 ms to prevent the trigger signal from interfering with the echo signal.cdn.sparkfun+1

5.3 PID Control

PID stands for Proportional–Integral–Derivative. It is one of the most widely used feedback-control techniques in industrial automation. A PID controller calculates the difference between a desired setpoint and a measured process variable, known as error, and adjusts the actuator output accordingly.

The PID controller output is:

u(t)=Kpe(t)+Ki∫0te(τ)dτ+Kdde(t)dtu(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}u(t)=Kpe(t)+Ki∫0te(τ)dτ+Kddtde(t)

Where:

- u(t)u(t)u(t) = controller output
- e(t)e(t)e(t) = error
- KpK_pKp = proportional gain
- KiK_iKi = integral gain
- KdK_dKd = derivative gain

The proportional term responds to the present error. The integral term corrects accumulated past error. The derivative term predicts future error based on the rate of change of error.

5.4 PID Implementation on Arduino

Arduino provides several PID libraries. The standard Arduino PID library is compatible with all Arduino architectures and adjusts an output to keep an input close to a desired setpoint. The ArduPID library provides improved accuracy and additional configuration options such as output limits, integral limits, anti-windup modes, sample time, and controller direction.

6. System Description

The proposed system consists of a microcontroller-based closed-loop level-control system. The ultrasonic sensor measures the distance to the liquid surface. The microcontroller converts the distance into liquid level and compares it with the desired setpoint.

The PID algorithm calculates the required pump output. The output is applied to a motor driver or MOSFET circuit, which controls the speed of a DC pump.

If the measured level is below the setpoint, the pump speed increases. If the measured level is above the setpoint, the pump speed decreases or the pump stops.

The system also includes a display module to show the current level, setpoint, and pump status. Push buttons or a potentiometer are used to change the desired level. LEDs and a buzzer provide visual and audible alerts.

8. Hardware Requirements

- 1 x Arduino UNO Board
- 1 x HC-SR04 Ultrasonic Sensor
- 1 x Servo Motor (SG90)
- Jumper Wires
- USB Cable
- Power Supply (9V Battery with connector)
- Cardboard and plastic ball for the demonstrative structure

➢ *Arduino UNO Board*

- The Arduino UNO is an open-source microcontroller development board based on the ATmega328P microcontroller. It is widely used in electronics, robotics, automation, and embedded system projects. It allows users to connect various electronic components, such as sensors, LEDs, motors, displays, and other devices, to develop interactive electronic applications.
- The Arduino UNO board has 14 digital input/output pins, of which 6 can be used as PWM outputs, 6 analog input pins, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
- It can be programmed using the Arduino IDE with a simple programming language based on C and C++. The board receives instructions from the program and controls connected devices according to the given commands.

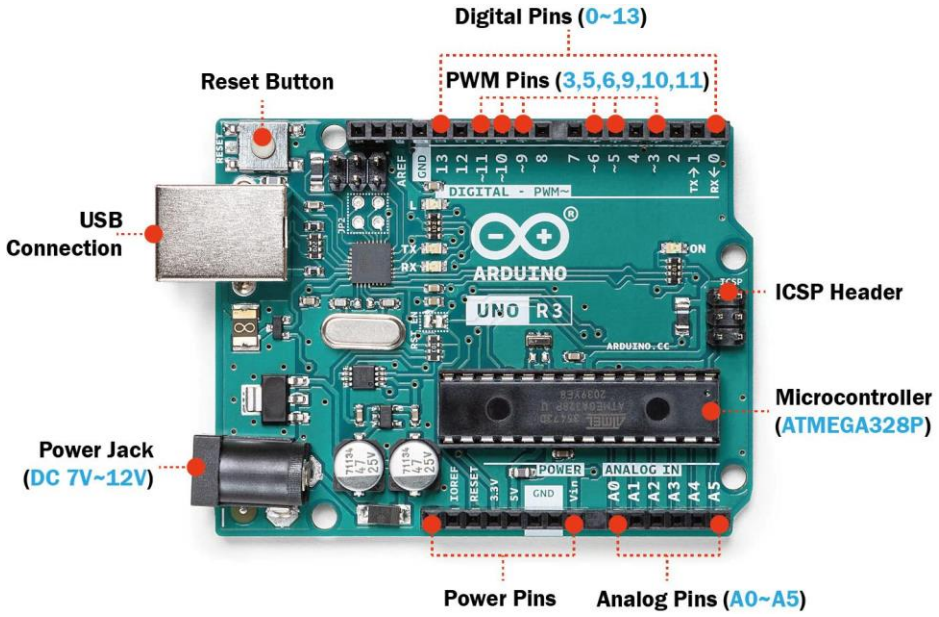


Fig -02

➢ *HC-SR04 Ultrasonic Sensor*

- The HC-SR04 Ultrasonic Sensor is an electronic distance-measuring sensor that uses ultrasonic sound waves to detect objects and measure the distance between the sensor and an object. It operates by transmitting high-frequency ultrasonic waves (approximately 40 kHz) and receiving the reflected waves (echo) from an object.
- The sensor consists of two main ultrasonic transducers: one acts as a transmitter (Trig), which sends ultrasonic waves, and the other acts as a receiver (Echo), which detects the reflected waves. The distance is calculated by measuring the time taken for the ultrasonic waves to travel to the object and return to the sensor.
- The HC-SR04 sensor is commonly used with microcontrollers such as Arduino UNO for distance measurement, obstacle detection, robotics, and automatic control systems.



Fig -03

➢ *Servo Motor (SG90)*

- The SG90 Servo Motor is a small, lightweight, and compact rotary actuator used to control angular position with high precision. It is commonly used in robotics, automation, and Arduino-based electronic projects. The SG90 is a micro servo motor that can rotate approximately 180 degrees, allowing accurate control of its shaft position.
- The SG90 servo motor consists of a DC motor, a gear mechanism, a position feedback system, and an electronic control circuit. It operates based on Pulse Width Modulation (PWM) signals received from a microcontroller such as the Arduino UNO. By varying the width of the control pulse, the motor shaft can be positioned at a desired angle.
- The SG90 servo motor is widely used in projects requiring controlled movement, such as robotic arms, automatic doors, obstacle-avoiding robots, and digital measurement systems.



Fig -04

• *Jumper Wires*

Jumper Wires are electrical connecting wires used to establish temporary connections between different electronic components, sensors, and microcontroller boards. They are commonly used in electronics prototyping, Arduino projects, and embedded system development.

Jumper wires are used to connect components such as Arduino UNO, ultrasonic sensors, servo motors, breadboards, LEDs, and other electronic modules without requiring soldering. They provide an easy and flexible method of transferring electrical power and signals between different components.

Jumper wires are generally available in three types: Male-to-Male, Male-to-Female, and Female-to-Female. They are commonly insulated with different colours to help identify power supply, ground, and signal connections.



Fig -05

• *USB CABLE*

A USB (Universal Serial Bus) Cable is an electrical communication cable used to establish a connection between electronic devices for data transmission and power supply. In Arduino-based projects, a USB cable is commonly used to connect the Arduino UNO board to a computer or laptop for programming, uploading code, and supplying electrical power.

The USB cable allows communication between the computer and the Arduino UNO microcontroller. It transfers program instructions from the Arduino IDE to the board and also supports serial communication for monitoring sensor readings and debugging the project.

The Arduino UNO R3 commonly uses a USB Type-B cable for connecting to a computer. One end of the cable connects to the Arduino board, while the other end connects to the computer's USB port.



Fig -06

• *Power Supply (9V Battery with connector)*

A 9V Battery with Connector is a portable DC power source used to provide electrical energy to electronic circuits and microcontroller-based projects. It consists of a 9V battery and a battery connector that allows easy connection to the power input terminals of an electronic circuit or Arduino UNO board.

In Arduino-based projects, the 9V battery is commonly connected to the Arduino UNO through its DC barrel jack using a 9V battery snap connector with a suitable DC plug. It supplies electrical power to the microcontroller and connected electronic components when a computer or external power adapter is not available.

The Arduino UNO has an onboard voltage regulator that converts the input voltage into a suitable operating voltage for the board. The 9V battery provides portability and makes the project suitable for standalone operation.

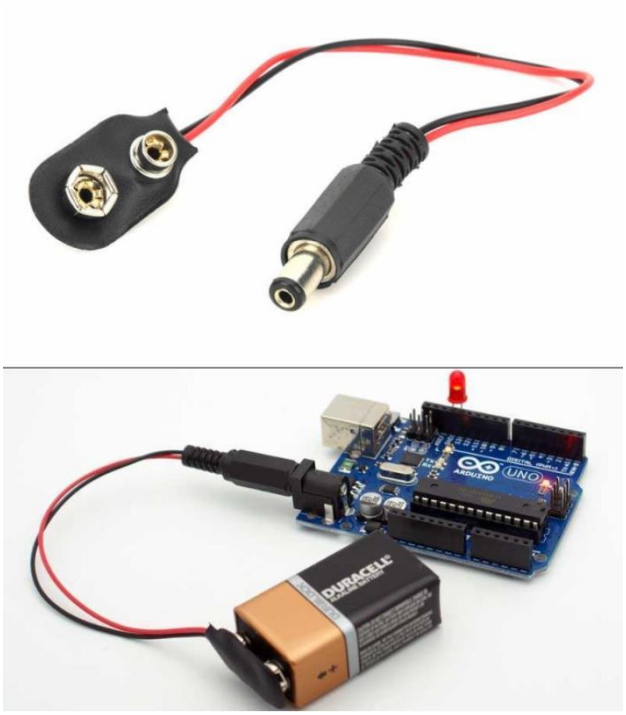


Fig -07

9. Software Requirements

- Arduino IDE
- Embedded C / C++ programming language
- Arduino PID library
- LiquidCrystal_I2C library
- Adafruit SSD1306 library
- Adafruit GFX library
- Ultrasonic sensor library or direct `pulseIn()` method
- Serial Monitor for debugging
- Optional: Blynk, ThingSpeak, Firebase, MQTT, or Node-RED for IoT monitoring
- Window 10 or higher versiom

The Arduino PID library is compatible with all Arduino boards and allows tuning using P, I, and D parameters. ArduPID also supports output limits, anti-windup, sample-time configuration, and direction control.docs.arduino

10. Circuit Design and Connections

Circuit Diagram

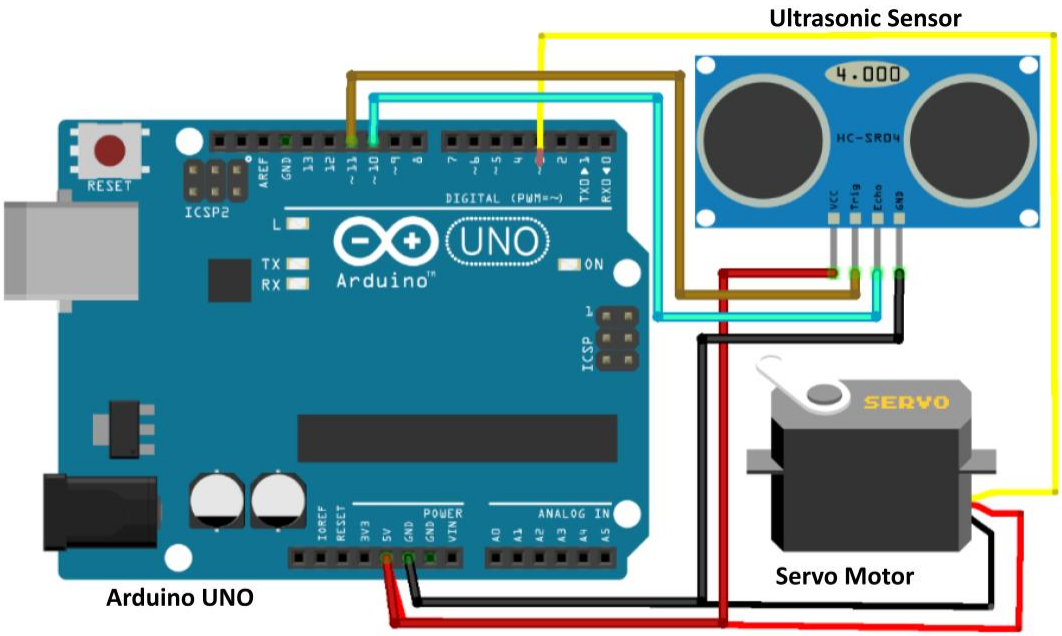


Fig 08

10.1 Ultrasonic Sensor Connection

The HC-SR04 sensor is mounted at the top of the tank, facing downward toward the liquid surface.

HC-SR04 Pin	Arduino Uno Pin
VCC	5V
GND	GND
TRIG	D9

HC-SR04 Pin	Arduino Uno Pin
ECHO	D10

The sensor requires a 10-microsecond trigger pulse. It then transmits an ultrasonic burst and measures the time taken for the echo to return.cdn.sparkfun+1

10.2 LCD Display Connection

LCD I2C Pin	Arduino Uno Pin
VCC	5V
GND	GND
SDA	A4
SCL	A5

For ESP32:

LCD I2C Pin	ESP32 Pin
VCC	3.3V or 5V
GND	GND
SDA	GPIO21
SCL	GPIO22

10.3 Pump Control Using MOSFET

For variable-speed control of a DC pump, a MOSFET-based driver is recommended.

Arduino PWM Pin --> 1 kΩ Resistor --> MOSFET Gate
Arduino GND --> MOSFET Source
DC Power Supply Negative --> MOSFET Source
DC Power Supply Positive --> Pump Positive Terminal

Pump Negative Terminal --> MOSFET Drain
Flyback Diode across Pump Terminals

A flyback diode should be connected across the pump terminals to protect the circuit from voltage spikes generated when the pump switches OFF.

10.4 Relay-Based Control

For simple ON/OFF control:

Arduino Digital Pin --> Relay IN
Relay VCC --> 5V
Relay GND --> GND
Relay COM --> Power Supply
Relay NO --> Pump / Valve

A relay is suitable for a solenoid valve or AC pump. However, for smooth PID-based speed control, PWM with a MOSFET driver is preferred.

11. Working Principle

11.1 Level Measurement

The ultrasonic sensor is installed at the top of the tank. It sends an ultrasonic pulse downward. The pulse reflects from the liquid surface and returns to the sensor. The microcontroller measures the time taken by the pulse to travel to the liquid surface and return.

The measured distance is:

Distance=Time×Speed of Sound2Distance = \frac{Time \times Speed\ of\ Sound}{2}Distance=2Time×Speed of Sound

If the total height of the tank is known, the liquid level is calculated as:

Liquid Level=Tank Height−Measured DistanceLiquid\ Level = Tank\ Height - Measured\ DistanceLiquid Level=Tank Height−Measured Distance

The liquid level can also be displayed as a percentage:

Level (%)=Liquid LevelMaximum Tank Height×100Level\ (\%) = \frac{Liquid\ Level}{Maximum\ Tank\ Height} \times 100Level (%)=Maximum Tank HeightLiquid Level×100

11.2 Setpoint Comparison

The user sets the desired liquid level, for example 60%. The microcontroller compares this setpoint with the measured level.

Error=Setpoint−Measured LevelError = Setpoint - Measured\ LevelError=Setpoint−Measured Level

If the measured level is below the setpoint, the error is positive. The controller increases the pump output. If the measured level is above the setpoint, the error is negative. The controller reduces or stops the pump output.

11.3 PID Control Action

The PID controller calculates the output using three terms:

- **Proportional term:** Responds to the current error.
- **Integral term:** Removes steady-state error by accumulating past error.
- **Derivative term:** Reduces overshoot by responding to the rate of change of error.

The PID output is converted into a PWM signal. The PWM duty cycle controls the average voltage applied to the DC pump, thereby controlling its speed.

11.4 Control Behaviour

Condition	Measured Level	Error	Controller Action
Tank is empty	Very low	Large positive	Pump runs at high speed
Level is below setpoint	Low	Positive	Pump speed increases
Level is near setpoint	Close to target	Small	Pump speed reduces gradually
Level is at setpoint	Equal to target	Zero	Pump output remains stable
Level is above setpoint	High	Negative	Pump slows down or stops
Tank is full	Maximum level	Negative	Pump stops and alarm may activate
Sensor fault	Invalid reading	Undefined	Pump stops and fault alarm activates

12. PID Control Theory

12.1 Closed-Loop Control

In a closed-loop control system, the output is continuously measured and compared with the desired value. The difference between the desired value and measured value is called error. The controller uses this error to adjust the actuator.

For this project:

- **Process variable:** Actual liquid level
- **Setpoint:** Desired liquid level
- **Controller:** PID controller
- **Actuator:** DC pump or solenoid valve
- **Sensor:** Ultrasonic level sensor
- **Controlled output:** Liquid level in tank

12.2 PID Equation

The continuous-time PID equation is:

u(t)=Kpe(t)+Ki∫0te(τ)dτ+Kdde(t)dtu(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}u(t)=Kpe(t)+Ki∫0te(τ)dτ+Kddtde(t)

In digital form:

un=Kpen+Ki∑i=0nei∆t+Kden−en−1∆tu_n = K_p e_n + K_i \sum_{i=0}^{n} e_i \Delta t + K_d \frac{e_n-e_{n-1}}{\Delta t}un=Kpen+Kii=0∑nei∆t+Kd ∆ten−en−1

Where:

- unu_nun = current PID output
- ene_nen = current error
- en−1e_{n-1}en−1 = previous error
- ∆t\Delta t∆t = sampling time

12.3 Effect of PID Parameters

Parameter	Effect on System
Increase in KpK_pKp	Faster response, but may cause oscillation

Parameter	Effect on System
Decrease in KpK_pKp	Slower response, more stable
Increase in KiK_iKi	Removes steady-state error, but may cause overshoot
Decrease in KiK_iKi	Less overshoot, but steady-state error may remain
Increase in KdK_dKd	Reduces overshoot and improves damping
Decrease in KdK_dKd	Faster response but more overshoot

12.4 PID Tuning

PID tuning is the process of selecting suitable values of KpK_pKp, KiK_iKi, and KdK_dKd. The values depend on tank size, pump capacity, pipe diameter, liquid viscosity, sensor noise, and required response speed.

A practical tuning procedure is:

1. Set Ki=0K_i = 0Ki=0 and Kd=0K_d = 0Kd=0.
2. Increase KpK_pKp gradually until the system responds quickly.
3. If oscillation occurs, reduce KpK_pKp.
4. Increase KiK_iKi slowly to remove steady-state error.
5. Add a small value of KdK_dKd to reduce overshoot.
6. Test the system at different setpoints.
7. Fine-tune the values until the response is stable.

The Ziegler–Nichols method is a well-known PID tuning approach that can be applied to systems with clear oscillatory behaviour. It provides heuristic formulas for selecting PID gain parameters based on the desired loop response.

13. System Algorithm

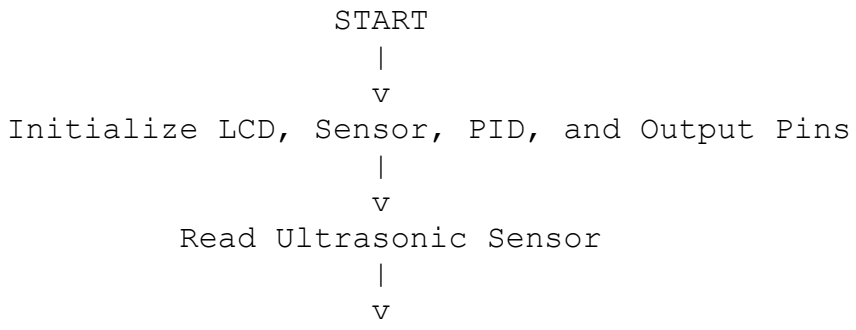
13.1 Main Algorithm

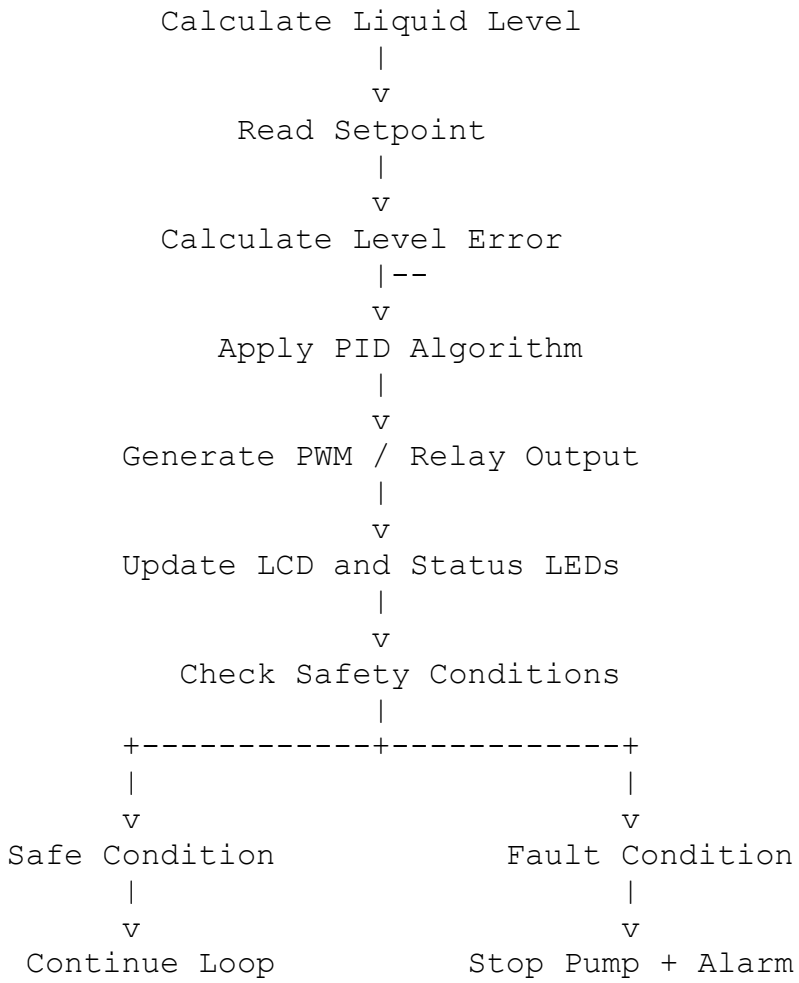
1. Start the system.
2. Initialize Arduino, LCD display, ultrasonic sensor, PID controller, and output pins.
3. Read the ultrasonic sensor.
4. Calculate the distance to the liquid surface.
5. Convert distance into liquid level.
6. Display the current liquid level.
7. Read the desired level setpoint.
8. Calculate the error:

   Error=Setpoint−Measured LevelError = Setpoint - Measured\
   LevelError=Setpoint−Measured Level

9. Apply the PID algorithm.
10. Generate PWM output for the pump.
11. Update the display.
12. Check safety conditions.
13. If the tank is empty, stop the pump and activate the dry-run alarm.
14. If the tank is full, stop the pump and activate the high-level alarm.
15. If the sensor fails, stop the pump and display an error message.
16. Repeat the process continuously

13.2 Flowchart



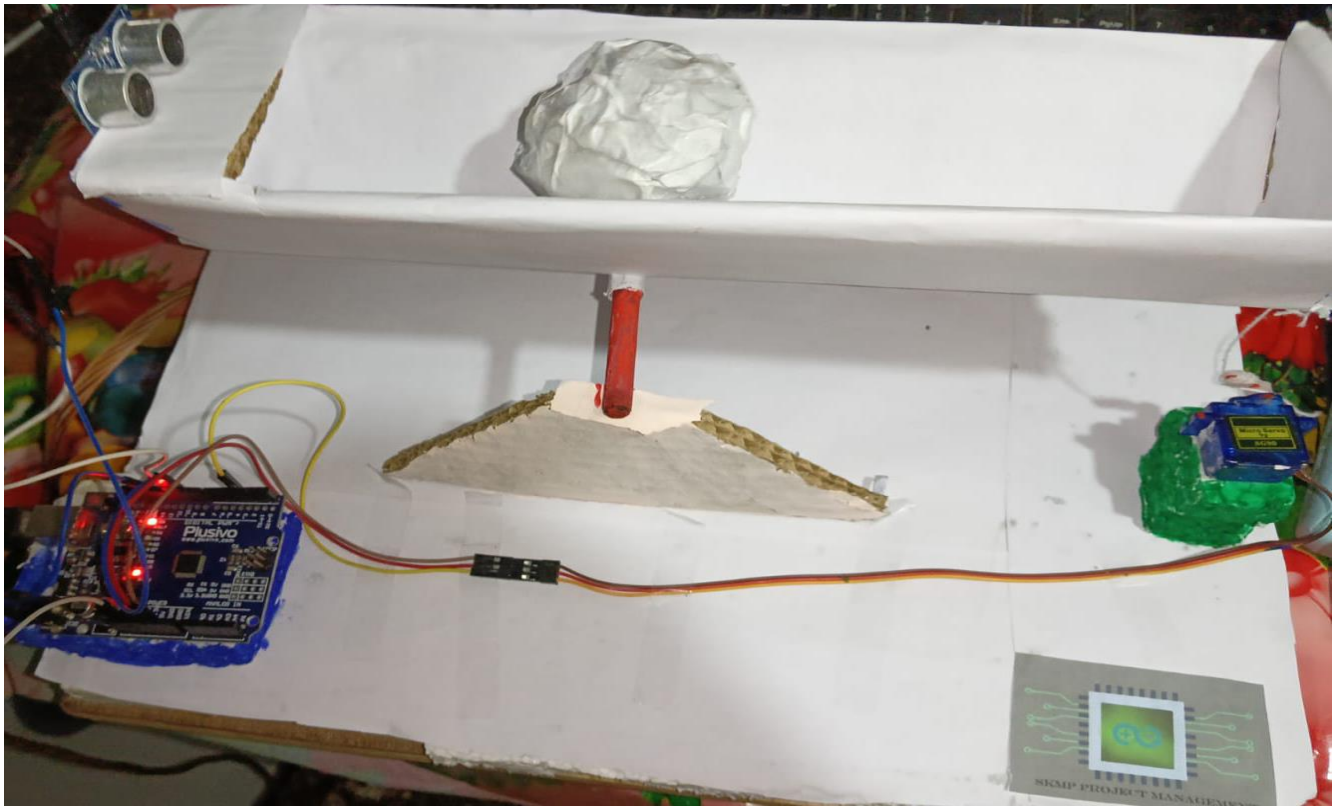


14. Arduino Program Code

```
#include <Servo.h>
Servo servo;


#define trig 11
#define echo 10


#define kp 0//15
#define ki 0//0.02
#define kd 20
```

```
double priError = 0;
double toError = 0;

void setup() {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  servo.attach(3);
  Serial.begin(9600);
  servo.write(50);

}
void loop() {
  PID();
  //  int a = distance();
  //  Serial.println(a);
}

long distance () {
  digitalWrite(trig, LOW);
  delayMicroseconds(4);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  long t = pulseIn(echo, HIGH);
  long cm = t / 29 / 2;
  return cm;
}

void PID() {
  int dis = distance ();


  int setP = 15;
  double error = setP - dis;


  double Pvalue = error * kp;
  double Ivalue = toError * ki;
  double Dvalue = (error - priError) * kd;

  double PIDvalue = Pvalue + Ivalue + Dvalue;
  priError = error;
  toError += error;
  Serial.println(PIDvalue);
  int Fvalue = (int)PIDvalue;


  Fvalue = map(Fvalue, -135, 135, 135, 0);
```

```
  if (Fvalue < 0) {
    Fvalue = 0;
  }
  if (Fvalue > 135) {
    Fvalue = 135;
  }

  servo.write(Fvalue);
}
```



15.1 Firmware limitations

This example does not include:

- Automatic PID tuning.
- Independent beam-angle sensing.
- Servo stall detection.

- An emergency-stop input.
- Persistent calibration storage.
- An LCD interface.
- Protection against every possible false ultrasonic echo.

A plausible but incorrect echo can pass simple range validation. Sensor placement and bench testing remain essential.

16. Assembly and Commissioning Procedure

Step 1: Build and inspect the mechanism

Construct the base, pivot, and channel. Confirm that the beam rotates freely and the ball remains contained.

Step 2: Check the servo independently

With the linkage disconnected, command small movements around its centre. Identify a safe travel range.

Step 3: Attach the linkage

Connect the servo horn to the beam without forcing either component. Check the full intended movement manually with power disconnected.

Step 4: Calibrate neutral

Adjust the linkage and neutral command so that the beam is approximately horizontal.

Step 5: Test sensor visibility

Move the ball through the usable channel and record distance readings. Check for reflections from side walls, support structures, or the opposite end.

Step 6: Verify direction

Apply a small servo correction while restraining the ball. Determine whether that correction would move the ball toward or away from the desired position.

Reverse `CONTROL_DIRECTION` if necessary.

Step 7: Enable feedback

Start with a central setpoint, conservative travel limits, zero integral gain, and a person ready to disconnect actuator power.

Step 8: Record behaviour

Log position and command data before changing gains.

17. PID Tuning Procedure

17.1 Begin with proportional and derivative control

Set:

Ki=0K_i=0Ki=0

Use a small proportional gain and add derivative damping cautiously. The goal is a controlled return toward the target without large repeated oscillations.

17.2 Adjust proportional gain

Increase KpK_pKp gradually if the response is too weak.

Reduce it if the beam repeatedly reaches its command limits or the ball accelerates aggressively through the target.

17.3 Adjust derivative gain

Increase KdK_dKd gradually if the ball repeatedly overshoots despite reasonable proportional response.

If the servo jitters in response to sensor noise, review filtering, mechanics, sampling, and derivative gain.

17.4 Add integral only when justified

After obtaining a stable response, add a small integral term only if a persistent offset remains.

First check whether that offset is caused by an incorrect neutral position or a mechanical problem.

17.5 Change one setting at a time

For every trial, record:

- Gain values.

- Set point.
- Initial ball position.
- Sampling interval.
- Filter settings.
- Observed oscillation.
- Actuator saturation.
- Recovery time.

No universal gain values can be inferred from the photograph.

18. Testing and Validation Plan

No experimental dataset has been supplied. The following table is a test plan, not a record of completed tests.

Test	Method	Evidence to record
Sensor calibration	Place ball at measured positions	Reference position and sensor readings
Neutral calibration	Command neutral with mechanism assembled	Beam orientation
Direction verification	Apply a small correction	Resulting tilt and expected ball motion
Position regulation	Release ball near a central target	Position-versus-time trace
Setpoint change	Change target within safe travel	Overshoot and settling time
Disturbance recovery	Gently displace the ball	Recovery response
Sensor loss	Remove ball or obstruct echo path	Fault status and servo response
Power integrity	Observe supply during movement	Resets or voltage dips
Repeatability	Repeat identical trials	Variation across runs
Mechanical endurance	Run controlled repeated trials	Loosening, heating, or wear

18.1 Performance metrics

➢ Overshoot

For a setpoint-change test, report how far the position passes beyond the new target. State whether overshoot is expressed in centimetres or normalized to the setpoint step.

➢ Sensor validity

Valid-reading rate=valid readingstotal readings×100%\text{Valid-reading rate}= \frac{\text{valid readings}}{\text{total readings}}\times100\%Valid-reading rate=total readingsvalid readings×100

19. Results Reporting Template

Complete this table only after physical testing.

Parameter	Measured result
Usable beam travel	To be measured
Ball diameter and mass	To be measured
Sensor operating range	To be verified
Neutral servo command	To be calibrated
Safe servo limits	To be calibrated
Final KpK_pKp	To be tuned
Final KiK_iKi	To be tuned
Final KdK_dKd	To be tuned
Sampling interval	To be recorded
Mean steady-state error	To be measured
RMSE	To be calculated
Maximum overshoot	To be measured
Settling time	To be measured

Parameter	Measured result
Invalid-reading rate	To be measured
Number of repeated trials	To be recorded

Do not publish labels such as “successful,” “high accuracy,” or “stable within ±1 cm” unless the recorded tests support them.

20. Troubleshooting

Symptom	Possible cause	Investigation or correction
Ball moves away from target	Incorrect feedback direction	Check linkage response and controller sign
Large repeated oscillations	Excessive gain, delay, or insufficient damping	Reduce aggressiveness and inspect response logs
Servo jitters with a stationary ball	Sensor noise or derivative amplification	Inspect raw readings and filtering
Board resets during movement	Supply limitations or wiring problems	Review servo power and common ground
Distance remains fixed	Echo from channel or support	Reposition sensor and check line of sight
Ball cannot be detected consistently	Poor echo geometry	Test another ball or sensing method
Ball stays offset from target	Neutral bias, friction, or insufficient compensation	Recalibrate mechanics before adding integral
Servo reaches limits repeatedly	Wrong direction or excessive correction	Stop and review sign, gains, and setpoint
Linkage clicks or bends	Binding or insufficient mechanical strength	Disconnect power and repair the mechanism

External servo power and appropriate supply selection are particularly important when movement causes controller resets.

21. Advantages

The proposed platform offers:

- A visible demonstration of closed-loop control.
- Direct comparison between controller settings.
- A practical introduction to sensor filtering.
- Experience with mechanical-electrical integration.
- Serial data suitable for plotting and analysis.
- A reusable laboratory platform for PID experiments.

Its educational value comes from making the consequences of control decisions physically observable.

22. Limitations

22.1 Sensor limitations

Ultrasonic measurements depend on target geometry and alignment. Ball detection must be verified over the complete operating region.

22.2 Mechanical limitations

Flexible supports, rough surfaces, backlash, and pivot friction can prevent repeatable behaviour even with well-written software.

22.3 Actuator limitations

A hobby servo has finite speed, torque, and positioning resolution. These characteristics restrict the range of ball motion that can be corrected.

22.4 Model limitations

The simplified rolling model omits several real effects. It should explain behaviour, not substitute for experimental identification.

22.5 Evidence limitations

The supplied photograph establishes construction features, not measured balancing performance.

23. Applications

The project is primarily suitable for:

- Control-systems laboratory demonstrations.
- Arduino and embedded-programming education.
- Robotics workshops.
- Student exhibitions.
- Studies of filtering and sampling.
- Comparison of P, PD, PI, and PID control.
- Introductory system-identification experiments.

Comparable Arduino ball-and-beam projects use the same general sensor-controller-servo feedback structure.projecthub.arduino+1

The pictured prototype should not be described as a water-level, boiler-feed, or chemical-tank controller.

25. Future Improvements

Potential improvements include:

1. A stiffer beam and low-friction pivot.
2. An adjustable linkage with reduced backlash.
3. A position sensor selected specifically for reliable ball detection.
4. An IMU or encoder to measure actual beam inclination.
5. An LCD showing position, error, and status.
6. A potentiometer or encoder for local setpoint adjustment.
7. Data logging for repeatable comparisons.
8. An emergency-stop input.
9. Servo-current monitoring.
10. Model-based or state-feedback control experiments.

If an angle sensor is added, the report can distinguish actual beam-level indication from ball-position regulation more directly.

26. Budget Planning

A defensible publication budget should use actual purchase prices or dated supplier quotations. The photograph alone does not establish component costs.

Budget category	Items
Control electronics	Arduino-compatible board and USB cable
Sensing	Ultrasonic sensor
Actuation	Servo, horn, and linkage
Power	Regulated supply and switch
Structure	Beam, pivot, supports, and base
Assembly	Connectors, wires, fasteners, and adhesive
Optional monitoring	Display, controls, and logging hardware

Calculate:

Ctotal=∑iqipiC_{\text{total}}=\sum_i q_i p_iCtotal=i∑qipi

Record shipping, taxes, and reused components separately so readers can reproduce the estimate.

27. Safety Precautions

- Use a low-voltage regulated supply.
- Disconnect power before changing linkage connections.
- Keep fingers away from servo horns and pinch points.
- Fit end stops to contain the ball.
- Do not force the servo against mechanical limits.
- Secure the base before operating.
- Provide an accessible power-disconnect method.
- Stop testing if the servo stalls, overheats, or behaves unexpectedly.
- Do not leave the experimental apparatus operating unattended.
- Supervise demonstrations involving children.

A neutral servo command cannot guarantee safe ball containment; mechanical protection is still required.

28. Publication Notes

➢ *Suggested web headline*

DIY Arduino Ball-and-Beam Balancer: Digital Position Monitoring with PID Control

➢ *Suggested short description*

An Arduino-based balancing demonstrator that uses ultrasonic position feedback and a servo-driven beam to explore PID control, filtering, calibration, and automatic stabilization.

➢ *Suggested photograph caption*

Prototype showing an Arduino-compatible controller, ultrasonic sensing module, pivoted channel, and hobby servo. The photograph documents the construction; operating performance requires separate testing.

➢ *Recommended supporting material*

Before publishing a completed-build claim, include:

- Final circuit diagram.
- Actual component models.
- Beam dimensions.
- Linkage close-up.
- Calibrated servo limits.
- Versioned firmware.
- Measured response plots.
- A video showing both the mechanism and monitoring output.
- Clear identification of proposed versus implemented features.

Internet publication of the report does not mean the project itself has an internet connection. Wi-Fi monitoring would be a separate addition.

Last Chapter

Conclusion & Future scope

The pictured apparatus is consistent with an Arduino-based ball-and-beam control demonstrator: a position sensor supplies feedback, a controller computes corrective action, and a servo-driven mechanism changes the beam inclination. The actual linkage and operation still require verification.

The project provides a coherent way to study feedback control because its performance depends on the interaction of software, sensor behavior, actuator response, and mechanical construction. A credible public report should preserve that distinction: describe the visible prototype accurately, document the proposed control method, and reserve numerical performance claims for measured experiments.

FUTURE SCOPE

The proposed **PID-controlled** ball-and-beam system can be developed into a more capable platform for automatic balancing, digital monitoring, and control-system experiments. The following improvements are proposed extensions, not confirmed features of the photographed prototype.

1. Improved Mechanical Construction

The beam, pivot, and supporting structure could be redesigned using more rigid materials and adjustable mechanical components. A smoother rolling surface, reduced linkage backlash, and a low-friction pivot would provide a more consistent platform for evaluating controller performance.

2. Enhanced Position Measurement

Future work could compare the existing ultrasonic sensing approach with alternative position-measurement methods, such as optical distance sensing or camera-based tracking. Each method should be evaluated for measurement accuracy, response time, reliability, and suitability for the selected ball and beam dimensions.

3. Direct Beam-Angle Measurement

An additional angle sensor, such as an inertial measurement unit or rotary encoder, could be incorporated to measure the actual beam inclination. This would allow the system to display beam angle separately from ball position and servo command, making the “digital level indicator” function more explicit.

4. Automatic PID Tuning

An automatic tuning feature could be developed to help select suitable proportional, integral, and derivative gains. Its performance should be compared with manual tuning using measured settling time, overshoot, steady-state error, and repeatability.

5. Adjustable Setpoint and Local Display

A potentiometer, rotary encoder, or push-button interface could allow users to change the desired ball position without modifying the program. An LCD or OLED could display:

- Desired and measured ball position.
- Position error.
- Servo command.
- Measured beam angle, if available.
- Controller status and sensor faults.

6. Wireless Monitoring and Control

A future version could incorporate a wireless-capable controller and a web or mobile interface. Users could monitor system variables, view response graphs, and adjust permitted settings remotely. Mechanical limits and local shutdown controls should remain active independently of the wireless connection.

7. Data Logging and Performance Analysis

The system could record time-stamped position, error, and control-output data for later analysis. These records would support comparisons between tuning settings and provide evidence for publication claims about accuracy, stability, and disturbance recovery.

8. Advanced Control Techniques

The platform could be extended to investigate control approaches beyond conventional PID, including state-feedback, fuzzy-logic, adaptive, or model-predictive control. Each approach should be assessed under the same experimental conditions rather than assumed to outperform PID.

9. Improved Safety and Fault Handling

Future versions could include an emergency-stop switch, improved end stops, sensor-loss detection, and actuator monitoring. Fault responses should be tested specifically, including what happens when the ball is no longer detected or the mechanism becomes obstructed.

10. Two-Axis Ball-and-Plate Extension

A more advanced project could replace the single-axis beam with a two-axis tilting plate. This would extend the task from controlling position along one line to controlling position across a surface, requiring additional actuation, position sensing, and coordinated control.

11. Educational Experiment Platform

The prototype could be developed into a reusable laboratory kit with documented experiments for proportional, PD, PI, and PID control. Supporting worksheets, circuit diagrams, firmware, and recorded datasets would make the project easier to reproduce and useful for teaching feedback-control concepts.

REFERENCES

- K. Ogata, *Modern Control Engineering*, 5th ed. Pearson Education, 2010.quanser
- K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.onlinelibrary.wiley
- K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed. Instrument Society of America, 1995.onlinelibrary.wiley+1
- Arduino, "PID," *Arduino Documentation*. Online documentation for implementing and tuning PID control on Arduino-compatible boards.docs.arduino
- Arduino, "Servo," *Arduino Documentation*. Reference documentation for controlling hobby servo motors using the Arduino Servo library.docs.arduino
- Arduino, "Servo Motor Basics with Arduino," *Arduino Documentation*, April 3, 2024. Guide to servo operation, wiring, power requirements, and programming.docs.arduino
- SparkFun Electronics, "Ultrasonic Ranging Module HC-SR04," datasheet. Technical information on ultrasonic distance measurement, triggering, and echo timing.cdn.sparkfun
- Nihal Suri, "Self Balancing Ball and Beam using PID control," *Arduino Project Hub*, April 6, 2021. Practical reference for an Arduino-based ball-and-beam balancing system.projecthub.arduino
- Wolfram Research, "Balancing a Ball on a Beam," *Wolfram Language & System Documentation Center*. Tutorial covering system modelling, sensor calibration, servo actuation, and controller design.reference.wolfram
- PowerBroker2, "ArduPID: PID Library for Arduinos," *GitHub repository*. Software reference for configurable PID implementation, including output limits and anti-windup features.
    - .

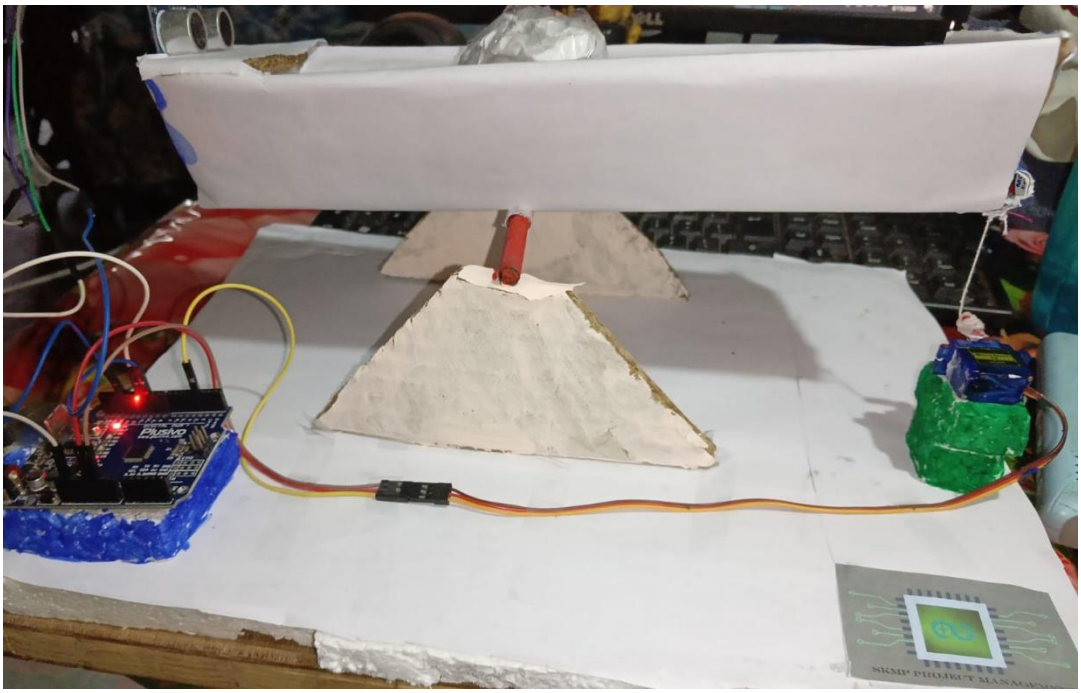
- University of Michigan, “Ball & Beam: System Modeling,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Covers the physical arrangement, governing equations, transfer function, and state-space representation.ctms.engin.umich
- University of Michigan, “Ball & Beam: System Analysis,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Examines the system model, pole locations, and open-loop response.ctms.engin.umich
- University of Michigan, “Ball & Beam: PID Controller Design,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Demonstrates proportional and proportional–derivative control for ball-position regulation.ctms.engin.umich
- University of Michigan, “Ball & Beam: Digital Controller Design,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Explains discretization and the development of a digital controller for the ball-and-beam system.ctms.engin.umich
- University of Michigan, “Ball & Beam: Simulink Modeling,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Describes the construction of a simulation model using the system’s governing equations.ctms.engin.umich
- University of Michigan, “Ball & Beam: Simulink Controller Design,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Demonstrates closed-loop simulation and implementation of a lead compensator.ctms.engin.umich
- University of Michigan, “Ball & Beam: Root Locus Controller Design,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Explains controller design using root-locus analysis and a lead compensator.ctms.engin.umich
- University of Michigan, “Ball & Beam: Frequency Domain Methods for Controller Design,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Discusses Bode plots, phase margin, and frequency-domain compensation.ctms.engin.umich
- University of Michigan, “Introduction: PID Controller Design,” *Control Tutorials for MATLAB and Simulink*. Online technical tutorial. Explains proportional, integral, and derivative actions and their effects on transient response and steady-state error.ctms.engin.umich
- Arduino, “HC-SR04,” *Arduino Documentation*. Online library documentation. Provides a software reference for measuring distance using an HC-SR04 ultrasonic sensor.

PROJECTLINK

Website lnk:-https://skmp-project-management.vercel.app/

Google Drive Link:- https://drive.google.com/drive/folders/12-CyjMh9ov8V9Vt9pT2FDhGmy57lwHWG?usp=sharing

VideoLink:- https://youtu.be/lR3lD4CF_lc?si=8rsIbsFlYHibPj5K





SKMP PROJECT MANAGEMENT

==========

SKMPPROJECTMANAGEMENT

EPARTMENTOFIOT