
Oran 1 University, Ahmed Ben Bella
Institute of Applied Sciences and Technologies
Department of Applied Physics and Optics
Final year project
Submitted for the attainment of the Master's Degree
Specialization: Instrumentation and Metrology
The ministerial decree n°1275 of 27 September 2022 on the mechanism
one diploma / one startup

Theme

***Design, implementation and optimization of an
embedded PID controller for efficient water
pump regulation***

Presented by:

Mr. Mati Redouane Abd El Madjid.

Mr. Benarbia Othmane

Defended on June 24, 2026 before the jury composed of:

Dr. CHAIB Mohamed.

Dr. CHERIET Amine

Dr. MAHIDA Hanane

Dr. GOURBI Abdelkader.

President.

Examiner

Examiner (Startup)

Supervisor

Academic year 2025/2026

Acknowledgements

We wish to express our deepest gratitude to all those who contributed, directly or indirectly, to the completion of this thesis.

Our sincere thanks go first and foremost to our supervisor Mr Gourbi Abdelkader for his rigorous guidance, wise advice, and unwavering support throughout this project. His availability and expertise were crucial to its successful completion.

We also thank all the professors in the department for the quality of their teaching and for the scientific and technical knowledge they imparted to us during our studies.

Finally, we warmly thank our classmates especially Mehidi Djamel and family for their moral support, encouragement, and invaluable assistance.

Dedication

We dedicate this work:

To our dear parents, for their love, sacrifices, and unwavering support throughout our academic journey.

To our family, for their encouragement and constant presence.

To our friends and colleagues, for their moral support and the shared moments that have enriched this experience.

To all those who contributed, directly or indirectly, to the completion of this work.

Abstract

This project presents the design and implementation of an ESP32-based water level control system using a Proportional–Integral–Derivative (PID) controller. Water level is measured by an ultrasonic sensor and regulated through a PWM-controlled DC pump to maintain a desired setpoint. The system supports both automatic tuning, based on the Åström–Hägglund relay feedback and Ziegler–Nichols methods, and manual PID parameter adjustment.

A local Human–Machine Interface (HMI) consisting of a 16×2 LCD and a 4×4 keypad enables user interaction, while a Python-based SCADA application provides wireless real-time monitoring and visualization via Wi-Fi.

Résumé

Ce projet présente la conception et la mise en œuvre d'un système de contrôle du niveau d'eau basé sur l'ESP32 utilisant un contrôleur Proportionnel–Intégral–Dérivé (PID). Le niveau d'eau est mesuré par un capteur à ultrasons et régulé via une pompe à courant continu (CC) contrôlée par MLI (PWM) afin de maintenir une consigne souhaitée. Le système prend en charge le réglage automatique, basé sur la méthode de rétroaction par relais d'Åström–Hägglund et les méthodes de Ziegler–Nichols, ainsi que l'ajustement manuel des paramètres du PID. Une interface homme-machine (IHM) locale, composée d'un écran LCD 16×2 et d'un clavier matriciel 4×4, permet l'interaction de l'utilisateur, tandis qu'une application SCADA développée en Python assure la surveillance en temps réel et la visualisation sans fil via Wi-Fi.

الملخص

يعرض هذا المشروع تصميم وتنفيذ نظام للتحكم في مستوى الماء باستخدام معالج ESP32 ووحدة تحكم تناسبية تكاملية تفاضلية (PID). يُقاس مستوى الماء بواسطة مستشعر فوق صوتي، ويُضبط عبر مضخة تيار مستمر تعمل بتقنية تعديل عرض النبضة (PWM) للحفاظ على القيمة المطلوبة. يدعم النظام الضبط التلقائي، بالاعتماد على طريقتي التغذية الراجعة المرحلية لـ Åström–Hägglund و Ziegler–Nichols، بالإضافة إلى الضبط اليدوي لمعاملات PID.

تتيح واجهة المستخدم (HMI) المحلية، المكونة من شاشة LCD بحجم 16×2 ولوحة مفاتيح بحجم 4×4، تفاعل المستخدم، بينما يوفر تطبيق SCADA المكتوب بلغة بايثون مراقبة وعرضًا لاسلكيًا فوريًا عبر شبكة Wi-Fi.

Table of contents

Acknowledgements.....	2
Dedication	3
Abstract	4
Résumé	4
المخلص	4
Table of contents.....	5
List of Abbreviations and Symbols.....	8
Acronyms	8
Mathematical Symbols.....	8
List of Figures	10
List of Tables.....	11
List of Equations.....	12
General Introduction	13
Chapter 1: Literature review	14
1. Introduction	15
2. History of Liquid-Level Control	16
3. Pumping Systems.....	16
4. The Fundamental Principles of Moving Fluid.....	17
5. Pumping System Hydraulic Characteristics.....	17
6. Pump Types	17
7. Level measurement types.....	18
8. Control Systems.....	19
8.1. Control System Definition	20
8.2. System Configurations.....	20
9. PID water level control	21
10. Conclusion	24
Chapter 2: Simulation and Control Performance Analysis	25
1 Introduction	26
2 Principle of Operation	26
3 System Mathematical Modeling and Transfer Functions	27
3.1 Nomenclature	27

3.2 Outlet & Torricelli's Law	28
3.3 Linearization.....	28
3.4 Closed-Loop Representation and Actuation Chain	29
4. Control Methodology: Tuning and Adaptation.....	31
4.1 Ziegler-Nichols (ZN) Tuning Principle.....	31
4.2 Gain Scheduling Principle.....	31
5. Simulation and Performance Analysis	32
5.1 Simulation Implementation Strategy	32
5.2 Simulation results	32
6. Global Discussion	37
7. Conclusion.....	38
Chapter 3: system realization and practical implementation	39
1. Introduction	40
2. Engineering Design and System Architecture	40
2.1 Functional Architecture	40
2.2 Overall System Block Diagram.....	41
3. Operating Principle.....	42
3.1 Water Level Measurement	42
3.2 Error Computation.....	42
3.3 PID Control Law	43
4. Materials and Hardware Components	43
4.1 Component Selection Criteria	43
4.2 Hardware Component Catalogue	44
4.3 Component Descriptions	45
5. Tinkercad Wiring Diagram.....	48
6. Electrical Design	48
6.1 Power Distribution	48
6.2 Protection Circuits.....	49
6.3 Electrical Schematic	49
6.4 Pin Connection Map.....	50
7. Mechanical Enclosure Design	51
7.1 Main Box.....	51
7.2 Lid	51

8. Software Architecture	52
8.1 Development Environment	52
8.2 Firmware Module Overview	53
9. Program Organigram	53
9.1 System Boot Sequence	55
9.2 Operating Modes	55
9.3 Autotuning Algorithm (Åström–Hägglund Relay Method)	56
10. Human-Machine Interface (HMI)	56
10.1 LCD Display (16×2 I2C)	56
10.2 Keypad Interface (4×4)	57
10.3 Python SCADA Monitoring Application	57
11. Software Protections	58
12. Prototype Realization	58
13. Experimental Results and Performance Analysis	60
13.1 Auto-Tuning Trial 1	61
13.2 Auto-Tuning Trial 2 — Same Setpoint	62
13.3 Manual / Auto Mode — Validation with Auto-Tuning Parameters	63
13.4 Manual Mode (Improved Ki)	65
13.5 Analysis and Interpretation	66
14. conclusion	69
General conclusion	70
References	72
Annexes	74
MATLAB Libraries and Functions Used	74
Python HMI code	75
Arduino code	77

List of Abbreviations and Symbols

Acronyms

PID Proportional–Integral–Derivative

PWM Pulse Width Modulation

ESP32 Espressif 32-bit microcontroller (dual Xtensa LX6, 240 MHz)

HC-SR04 Ultrasonic distance sensor (40 kHz)

MOSFET Metal–Oxide–Semiconductor Field–Effect Transistor

LCD Liquid Crystal Display

I2C Inter–Integrated Circuit serial communication bus

HMI Human–Machine Interface

SCADA Supervisory Control and Data Acquisition

ZN Ziegler–Nichols tuning method

GPIO General Purpose Input/Output

TCP Transmission Control Protocol

Mathematical Symbols

A Cross-sectional area of the tank (m^2)

a Cross-sectional area of the discharge orifice (m^2)

h(t) Instantaneous liquid height (m)

Q_{in}, Q_{out} Volumetric inflow / outflow rate (m^3/s)

g Gravitational acceleration (9.81 m/s^2)

R Hydraulic resistance

K_s Plant static gain

τ Plant time constant (s)

K_p, K_i, K_d PID proportional, integral, derivative gains

K_u, T_u Ultimate gain and oscillation period (Ziegler–Nichols)

T_s Sampling period (100 ms)

List of Figures

Figure 1 pump types.....	18
Figure 2: level sensors types	19
Figure 3 open-loop system	20
Figure 4 closed-loop system.....	21
Figure 5: Structure PID controller.....	22
Figure 6; Principle of Operation: Passive Drain Water Level Control System	27
Figure 7: closed loop PID regulation diagram	30
Figure 8 Step Transition (0.15 m $\rightarrow$ 0.20 m): Water Level and PWM Duty Cycle Response graphs	33
Figure 9: Direct Start-Up (0.18 m Setpoint): Response graphs.....	35
Figure 10 Downward Transition (0.15 m $\rightarrow$ 0.10 m): Water Level and PWM Duty Cycle Response graphs	36
Figure 13: Functional Architecture of the system	41
Figure 14: ESP32 microcontroller.....	45
Figure 15: The HC-SR04 Ultrasonic Sensor	45
Figure 16: IRLZ44N Logic-Level MOSFET	46
Figure 17: 12V DC submersible pump	46
Figure 18: 4 $\times$ 4 Matrix Keypad.....	47
Figure 19: I2C LCD 16 $\times$ 2 Display	47
Figure 20 Tinkercad wiring diagram of the prototype	48
Figure 21 Electrical schematic of the PID water level control system	49
Figure 22 FreeCAD 3D model of the enclosure main box.....	51
Figure 23 FreeCAD 3D model of the enclosure lid	52
Figure 24: System flowchart	54
Figure 25 Keypad Interface	57
Figure 26 Python SCADA Monitoring Application (v04.py).....	58
Figure 27 real word main box controller.....	59
Figure 28 water source tank & inlet pump	59
Figure 29 main box controller interior circuit	60
Figure 30: Level response — Auto-Tuning Trial 1 (Auto mode, setpoint 10 cm).....	61
Figure 31: Level response — Auto-Tuning Trial 2 (Auto mode, setpoint 10 cm).....	62
Figure 32: Level response — Auto-Tuning Trial 3 (Auto mode, setpoint 5 cm)	63
Figure 33: Level response — Manual mode with auto-tuned parameters K_{p1} , K_{i1} , K_{d1}	64
Figure 34: Level response — Manual mode with increased K_i (enhanced integral action)	65

List of Tables

Table 1: PID terms roles.....	24
Table 2 Ziegler-Nichols method calculation	31
Table 3 Step Transition (0.15 m $\rightarrow$ 0.20 m): Water Level and PWM Duty Cycle values	34
Table 4 Direct Start-Up (0.18 m Setpoint): Filling from Empty Tank.....	35
Table 5 Downward Transition (0.15 m $\rightarrow$ 0.10 m): Energy-Conscious Level Reduction.....	37
Table 6 subsystem functional architecture	41
Table 7 Hardware Component Catalogue	44
Table 8 Pin Connection Map.....	50
Table 9 Software tools and development environment	52
Table 10 Firmware and software module summary	53

List of Equations

Equation 1: Proportional term equation	22
Equation 2: Integral term equation.....	23
Equation 3: Derivative term equation	23
Equation 4: PI controller equation	23
Equation 5: PD controller equation	23
Equation 6: PID controller equation.....	23
Equation 7: Volumetric Mass Balance.....	27
Equation 8: Torricelli Outlet Velocity	28
Equation 9: Volumetric Outflow Rate	28
Equation 10 level equation.....	28
Equation 11: First-Order Taylor Linearization of Outflow Around h_0	28
Equation 12 Partial Derivative of Q_{out} with Respect to h at h_0	28
Equation 13: Hydraulic Resistance	29
Equation 14: Plant Transfer Function	29
Equation 15 numerical transfer function	29
Equation 16 controller transfer function	29
Equation 17: error equation.....	29
Equation 18: output PID equation	29
Equation 19: closed loop transfer function	29
Equation 20 distance equation.....	42
Equation 21 level equation.....	42
Equation 22 practical error equation	42
Equation 23 practical output equation.....	43
Equation 26 ultimate gain equation.....	56
Equation 27 optimal k_p gain equation.....	56

General Introduction

The rapid development of automation and embedded systems has significantly transformed industrial processes by improving efficiency, reliability, and operational safety. Among the various applications of automation, water level regulation remains an important challenge in many industrial, agricultural, and domestic systems. Maintaining a desired liquid level is essential for ensuring process stability, preventing overflow or dry-running conditions, reducing energy consumption, and guaranteeing optimal system performance.

Traditionally, water level regulation was achieved using manual supervision or simple on/off control techniques. However, such approaches often suffer from poor accuracy, excessive oscillations, and limited adaptability to changing operating conditions. To overcome these limitations, modern control systems increasingly employ feedback control strategies, particularly the Proportional–Integral–Derivative (PID) controller. Due to its simplicity, robustness, and widespread industrial acceptance, PID control remains one of the most effective methods for regulating dynamic processes.

The objective of this project is to design and implement an intelligent water level control system based on an ESP32 microcontroller. The proposed system continuously measures the water level using an ultrasonic sensor and regulates a DC pump through Pulse Width Modulation (PWM) to maintain a user-defined setpoint. To enhance usability and performance, the system incorporates both automatic PID tuning using the Åström–Hägglund relay feedback method combined with the Ziegler–Nichols tuning approach, and manual tuning capabilities for experimental analysis. In addition, a local Human–Machine Interface (HMI) based on an LCD display and keypad, together with a wireless SCADA monitoring application developed in Python, provides real-time visualization and supervision of the control process.

This dissertation is organized into three chapters. Chapter 1 presents the theoretical background related to water level measurement techniques, control system fundamentals, PID controllers, and tuning methods. Chapter 2 focuses on the mathematical modeling and simulation of the water level system, including transfer function derivation, controller design, and performance evaluation through simulation. Chapter 3 describes the practical realization, hardware architecture, and software development of the prototype, concluding with experimental results and a performance analysis of the autotuning control system under various operating conditions.

Chapter 1: Literature review

1. Introduction

Water-level regulation is a cornerstone of hydraulic systems used in industry, agriculture, municipal supply and domestic appliances. Precise control prevents overflow, avoids pump cavitation, reduces energy consumption and guarantees product quality in processes such as chemical treatment, irrigation and cooling towers.

The complexity and dynamic nature of modern hydraulic systems necessitate the deployment of advanced automatic control techniques. Manual control methods are often inefficient, prone to human error, and incapable of responding rapidly to sudden disturbances or changes in demand. Automatic control systems provide continuous monitoring, accurate adjustment, and optimized performance, leading to significant improvements in reliability, energy efficiency, and overall system stability. These systems integrate various sensors, such as level sensors for water depth and flowmeters for flow rate, with actuators like pumps and valves, and intelligent controllers to maintain desired operational setpoints. The integration of modern microcontrollers, such as the ESP32, facilitates the deployment of these sophisticated control algorithms in real-world applications, offering cost-effective and powerful solutions.

Among the many control strategies, the Proportional-Integral-Derivative (PID) controller remains the most widely adopted because it combines three corrective actions in a single, simple algorithm. The proportional term reduces the instantaneous error, the integral term eliminates steady-state offset, and the derivative term damps oscillations, delivering fast response and high stability.

This chapter aims to lay the foundational understanding necessary for comprehending water level control systems and the principles of Proportional-Integral-Derivative (PID) regulation. It will trace the historical evolution of water management and control, introduce fundamental concepts of automatic control systems, and then delve into the specific application of these principles to water level regulation. The objective is to provide a comprehensive overview that contextualizes the subsequent development and implementation of advanced control strategies.

2. History of Liquid-Level Control

The Greeks began engineering feedback systems around 300 B.C. A water clock invented by Ktesibios operated by having water trickle into a measuring container at a constant rate. The level of water in the measuring container could be used to tell time. For water to trickle at a constant rate, the supply tank had to be kept at a constant level. This was accomplished using a float valve similar to the water-level control in today's flush toilets.

Soon after Ktesibios, the idea of liquid-level control was applied to an oil lamp by Philon of Byzantium. The lamp consisted of two oil containers configured vertically. The lower pan was open at the top and was the fuel supply for the flame. The closed upper bowl was the fuel reservoir for the pan below. The containers were interconnected by two capillary tubes and another tube, called a vertical riser, which was inserted into the oil in the lower pan just below the surface. As the oil burned, the base of the vertical riser was exposed to air, which forced oil in the reservoir above to flow through the capillary tubes and into the pan [1]. The transfer of fuel from the upper reservoir to the pan stopped when the previous oil level in the pan was reestablished, thus blocking the air from entering the vertical riser. Hence, the system kept the liquid level in the lower container constant [2].

3. Pumping Systems

A pumping system is a mechanical device designed to move fluids—liquids or gases—from one location to another. This transfer involves increasing the fluid's pressure to overcome resistance from friction or gravity, allowing the substance to flow through complex piping networks or uphill. Pumps operate by converting external energy, typically electrical or mechanical, into fluid energy to maintain continuous movement. The ability of pumps to manipulate fluid dynamics makes them foundational components in nearly every major piece of global infrastructure and industrial process [3].

4. The Fundamental Principles of Moving Fluid

Pumping systems achieve fluid movement through two distinct engineering methodologies. The first relies on the transfer of kinetic energy, where a rotating component imparts speed to the fluid mass. This acceleration converts velocity into pressure as the fluid exits the pump casing. The second method involves mechanical trapping, operating on the principle of volume displacement. This process physically encloses a fixed quantity of fluid within a chamber and mechanically forces that volume out of the discharge point. This mechanism ensures a consistent volume of fluid is delivered regardless of the pressure resisting the flow [4].

5. Pumping System Hydraulic Characteristics

In a pumping system, the objective, in most cases, is either to transfer a liquid from a source to a required destination, e.g., filling a high-level reservoir, or to circulate liquid around a system, e.g., as a means of heat transfer. Pressure is needed to make the liquid flow at the required rate and this must overcome losses in the system. Losses are of two types: static and friction head [5].

Static head, in its most simple form, is the difference in height of the supply and destination of the liquid being moved, or the pressure in a vessel into which the pump is discharging, if it is independent of flow rate.

Friction head (sometimes called dynamic head loss), is the friction loss on the liquid being moved, in pipes, valves, and other equipment in the system. This loss is proportional to the square of the flow rate. A closed-loop circulating system, without a surface open to atmospheric pressure, would exhibit only friction losses [6].

6. Pump Types

Proper selection of pumps, motors, and controls to meet the process requirements is essential to ensure that a pumping system operates effectively, reliably, and efficiently. All pumps are divided into the two major categories of positive displacement (PD) and rotodynamic as shown in the figure 1 [7] .



Figure 1 Pump types

7. Level measurement types

Level sensors are devices used to measure the level or height of a liquid or fluid in a container. They can be used to measure the level of water in a tank, oil or fuels in a storage container, or liquid in a chemical process vessel. There are several types of liquid level sensors as shown in the figure 2, which can be used to measure the level of a liquid in different ways. These sensors, or level switches, can be used to control and monitor the liquid level in a system and can also be used to trigger alarms or shut-off valves in the event of a high or low liquid level [8].

The main difference as shown in the figure 1 between level sensor technologies lies in their physical contact with the measured material. They can also be divided into two other categories [9]:

Contact sensors touch the substance they measure. These include:

- Float sensors that rise and fall with liquid levels
- Capacitive probes that measure electrical properties through direct contact
- Conductive sensors that measure electrical conductivity between electrodes

Non-contact sensors measure from a distance. Common examples include:

- Ultrasonic sensors that emit sound waves
- Radar sensors that use electromagnetic waves
- Optical sensors that detect levels using light beams
- Laser-based systems that provide precise measurements

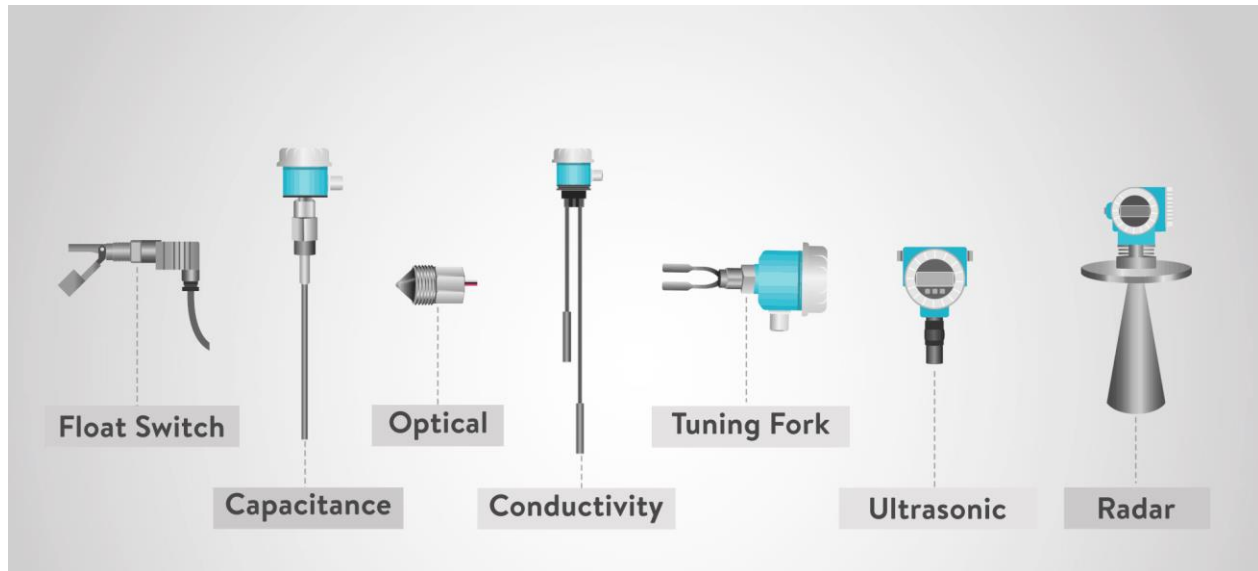


Figure 2: Level sensors types [10]

Contact sensors provide affordable solutions for standard applications but face challenges with corrosive substances or high-pressure environments. Non-contact measurement provides faster readings and prevents contamination. These sensors work well in hazardous or sterile environments, though they cost more than contact options.

8. Control Systems

Control systems are an integral part of modern society. Numerous applications are all around us: The rockets fire, and the space shuttle lifts off to earth orbit; in splashing cooling water, a metallic part is automatically machined; a self-guided vehicle delivering material to workstations in an aerospace assembly plant glides along the floor seeking its destination. These are just a few examples of the automatically controlled systems that we can create [11].

8.1. Control System Definition

A control system consists of subsystems and processes (or plants) assembled for the purpose of obtaining a desired output with desired performance, given a specified input.

8.2. System Configurations

There are two major configurations of control systems: open loop and closed loop.

8.2.1. Open-Loop Systems

The figure 3 shows the Open-Loop Systems Output transducer which converts the form of the input to that used by the controller. The controller drives a process or a plant. The input is sometimes called the reference, while the output can be called the controlled variable. Other signals, such as disturbances, are shown added to the controller and process outputs via summing junctions, which yield the algebraic sum of their input signals using associated signs [12].

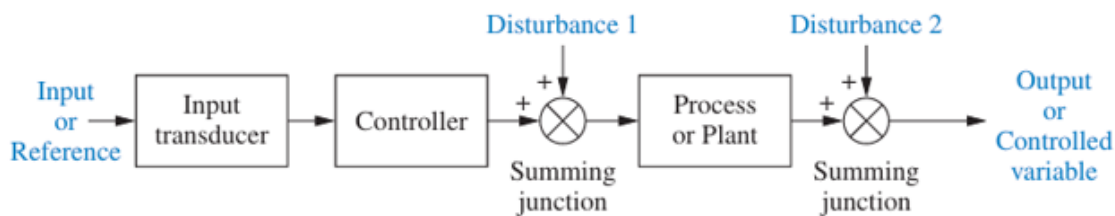


Figure 3 open-loop system

The distinguishing characteristic of an open-loop system is that it cannot compensate for any disturbances that add to the controller's driving signal

Open-loop systems, then, do not correct for disturbances and are simply commanded by the input

8.2.2 Closed-Loop (Feedback Control) Systems

The disadvantages of open-loop systems, namely sensitivity to disturbances and inability to correct for these disturbances, may be overcome in closed-loop systems (figure 4).

The input transducer converts the form of the input to the form used by the controller. An output transducer, or sensor, measures the output response and converts it into the form used by the controller.

The closed-loop system compensates for disturbances by measuring the output response, feeding that measurement back through a feedback path, and comparing that response to the input at the summing junction.

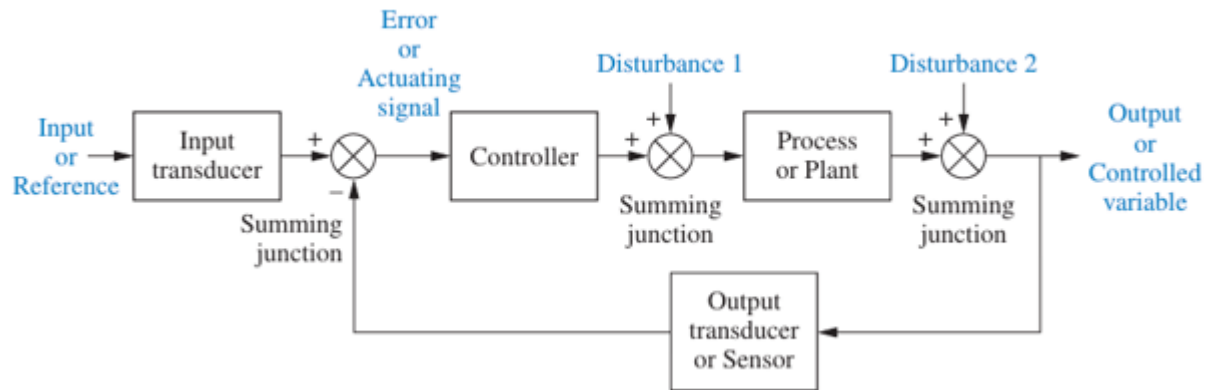


Figure 4 closed-loop system

Closed-loop systems, then, have the obvious advantage of greater accuracy than open-loop systems. They are less sensitive to noise, disturbances, and changes in the environment. Transient response and steady-state error can be controlled more conveniently and with greater flexibility in closed-loop systems, often by a simple adjustment of gain (amplification) in the loop and sometimes by redesigning the controller. We refer to the redesign as compensating the system and to the resulting hardware as a compensator [12].

9. PID water level control

Water level control systems are integral to various sectors, including agriculture, water treatment plants, industrial processes, and residential water management. These systems aim to maintain desired water levels within reservoirs, tanks, or other storage units to ensure optimal performance and prevent overflow or depletion. Efficient water level control is crucial for resource conservation, process optimization, and environmental protection. Proportional-Integral-Derivative (PID) controllers are widely used in these applications due to their simplicity, reliability, and effectiveness in maintaining desired set points [13].

PID controllers are a type of feedback controller that combines proportional, integral, and derivative actions to control a process variable, such as water level. This combination allows PID controllers to address both transient and steady-state behaviors, ensuring that water levels remain within desired limits as shown in figure 5 [14] .

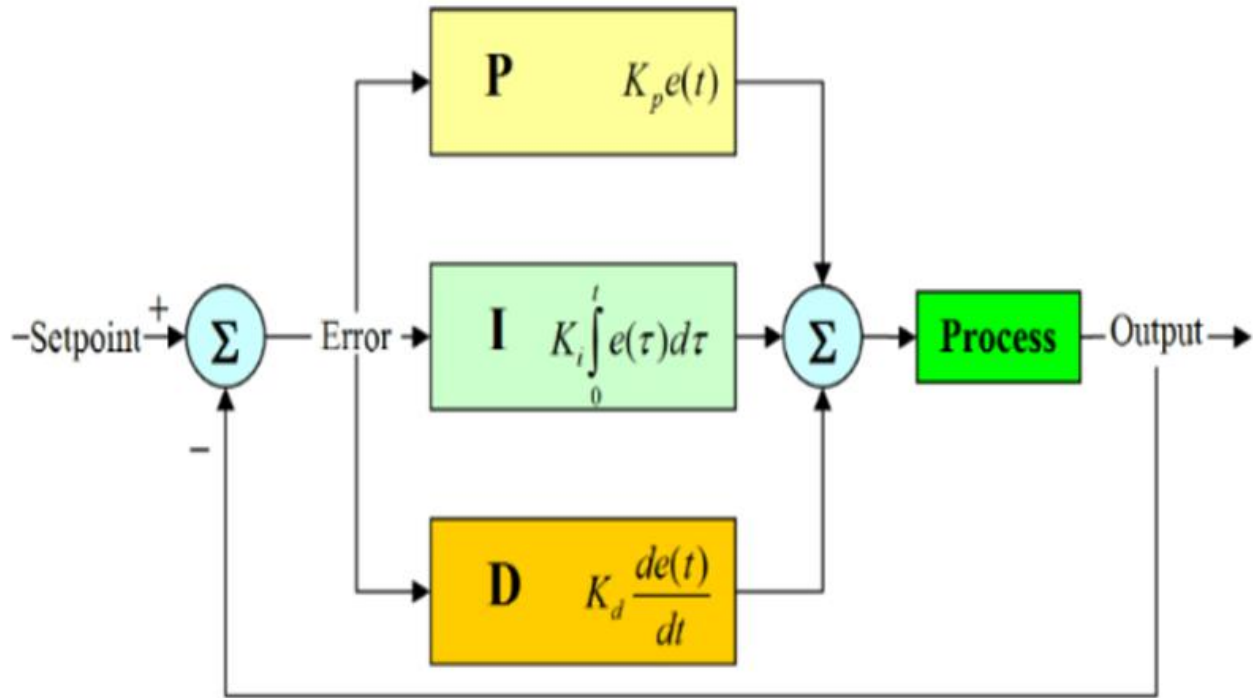


Figure 5: Structure PID controller

The provided content outlines the three core components of a PID controller—Proportional (P), Integral (I), and Derivative (D)—which work together to regulate a system like water level. The proportional term reduces current error by responding to the difference between the desired setpoint and the actual measurement, the integral term eliminates steady-state error by accumulating past errors, and the derivative term stabilizes the system and dampens oscillations by predicting future errors based on their current rate of change. While P, I, and D function as individual terms or controls on their own, combining them transitions them into distinct controllers, such as PI, PD, or full PID controllers.

$$\mathbf{P(t)} = k_p \cdot e(t)$$

Equation 1: Proportional term equation

where K_p is the proportional gain and $e(t)$ is the error at time t .

$$\mathbf{I}(t) = k_i \cdot \int_0^t e(\tau) d\tau$$

Equation 2: Integral term equation

Where K_i is the integral gain.

$$\mathbf{D}(t) = k_d \cdot \frac{de(t)}{dt}$$

Equation 3: Derivative term equation

Where K_d is the derivative gain.

$$\mathbf{PI} = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau) d\tau$$

Equation 4: PI controller equation

$$\mathbf{PD} = k_p \cdot e(t) + k_d \cdot \frac{de(t)}{dt}$$

Equation 5: PD controller equation

$$\mathbf{PID}(t) = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau) d\tau + k_d \cdot \frac{de(t)}{dt}$$

Equation 6: PID controller equation

These components' outputs are combined in another summation block to form the control signal, which is then applied to the process being controlled. The output of the process is feedback and compared with the setpoint, maintaining a closed-loop system to continuously adjust and correct the process variable (table 1) [14].

Term	Math Function	Effect on control system
Proportional (P)	$P(t) = k_p \cdot e(t)$	Typically, the main drive in a control loop, k_p reduces a large part of the overall error.
Integral (I)	$I(t) = k_i \cdot \int_0^t e(\tau) d\tau$	Reduces the final error in a system. Summing even a small error over time produces a drive signal large enough to move the system toward a smaller error.
Derivative (D)	$D(t) = k_d \cdot \frac{de(t)}{dt}$	Counteracts the k_p and k_i terms when the output changes quickly. This helps reduce overshoot and ringing. It has no effect on final error.

Table 1: PID terms roles

10. Conclusion

This chapter presented the theoretical background necessary for understanding and developing the proposed water level regulation system. The study began with an overview of pumping systems, their operating principles, and their importance in fluid transport and hydraulic applications. Different pump categories and their hydraulic characteristics were also introduced in order to better understand the actuator used in this project. Then the main techniques used for level and flow measurement were discussed, highlighting the role of sensors in providing accurate process information for monitoring and control purposes. Since reliable measurement is essential in any automated system, the selection of suitable sensing methods represents a key aspect in the design of water regulation systems. In addition, water level control strategies were discussed, focusing particularly on the Proportional-Integral-Derivative (PID) method as the primary approach for achieving precise and robust system regulation.

These theoretical foundations establish the scientific basis for the next phase of the project, which involves modeling and simulating the water level control system before practical implementation.

Chapter 2: Simulation and Control Performance Analysis

1 Introduction

The design and implementation of an embedded control system require a robust mathematical understanding of the physical plant. Before deploying control algorithms onto a microcontroller, the dynamic behaviors of the system must be modeled and simulated to verify performance and safeguard against instability. This chapter establishes the mathematical foundations of the water pumping system, deriving the continuous-time transfer functions for two structural configurations: a passive gravity-drained outlet system and an active pump-driven outlet system.

From an embedded systems perspective, the modeling process accounts for the operational constraints of the ESP32 microcontroller. Unlike continuous-time theoretical models, the microcontroller operates in the discrete-time domain, executing code through periodic timer interrupts. Consequently, this chapter bridges the gap between analog physical processes and digital execution by modeling the system in the Laplace domain and subsequently transforming it into the discrete-time domain. This approach utilizes a fixed sampling time ($T_s = 0.1$ seconds) and Zero-Order Hold (ZOH) discretization to accurately reflect the data acquisition cycles and pulse-width modulation (PWM) update behavior inherent in the embedded control loop.

The subsequent sections detail the plant modeling, the theoretical design of the PID control methodologies, and the results of the gain-scheduled simulation, providing a comprehensive verification of the system's control authority across its full operational range.

2 Principle of Operation

Figure 6 illustrates the overall system architecture. The objective is to maintain a precise target liquid level within a process tank, ensuring the water column remains stable despite continuous fluid loss. This stability is achieved through a dynamic balance between a variable inflow and a constant, externally determined outflow.

When the system reaches a steady state, the inlet flow rate perfectly offsets the discharge rate. By adjusting the inflow to align with the constant drainage, the system maintains the liquid at the desired height. Effectively, the controller treats the constant leakage as a predictable disturbance, modulating the supply to ensure that the equilibrium between inflow and outflow consistently holds the water at the required setpoint.

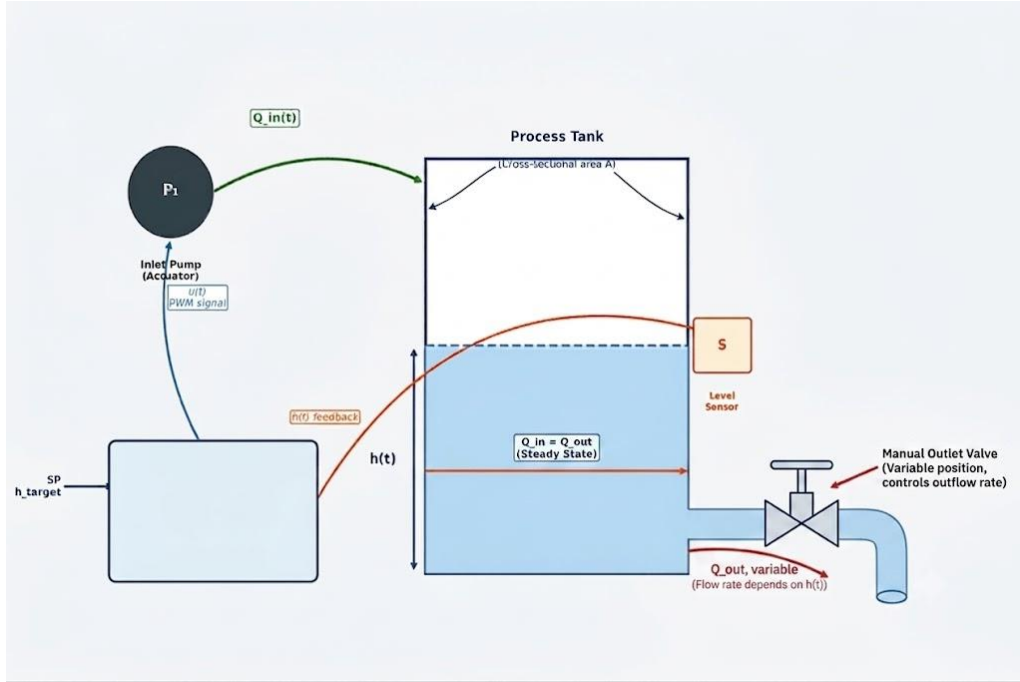


Figure 6 Principle of Operation: Passive Drain Water Level Control System

3 System Mathematical Modeling and Transfer Functions

The fluid system consists of a vertical, cylindrical process tank with a uniform cross-sectional area A . The fundamental state variable is the water height $h(t)$. The system is governed by the principle of conservation of mass, yielding the linear volumetric mass balance equation:

$$A \frac{dh}{dt} = Q_{in}(t) - Q_{out}(t)$$

Equation 7: Volumetric Mass Balance

3.1 Nomenclature

The system variables are defined as follows:

- A : Cross-sectional area of the tank (0.0625 m^2 , corresponding to a 20 cm radius).
- $h(t)$: Instantaneous liquid height as a function of time (m).
- dh/dt : Rate of change of water level over time (m/s).
- $Q_{in}(t)$: Dynamic volumetric inflow rate entering the top of the tank (m^3/s).

- $Q_{out}(t)$: Dynamic volumetric outflow rate leaving the bottom of the tank (m^3/s).

3.2 Outlet & Torricelli's Law

In the passive drainage layout, water exits the tank via a fixed orifice at the base. The flow rate is governed by Torricelli's Law [15], derived from Bernoulli's principle. The fluid velocity $v(t)$ at an exit orifice under a liquid column of height $h(t)$ is:

$$v(t) = \sqrt{2g \cdot h(t)}$$

Equation 8: Torricelli Outlet Velocity

Multiplying by the discharge orifice cross-sectional area a yields the volumetric outflow rate:

$$Q_{out}(t) = a \cdot \sqrt{2g \cdot h(t)}$$

Equation 9: Volumetric Outflow Rate

- a : Cross-sectional area of the passive discharge orifice (0.000019 m^2).
- g : Acceleration due to gravity (9.81 m/s^2).

$$h = \int (Q_{in}(t) - Q_{out}(t)) / A$$

Equation 10 level equation

Note: As indicated in (Eq. 9), the square-root relationship renders this system non-linear.

Therefore, linearization via a first-order Taylor series expansion around an equilibrium height h_0 is required for classical controller tuning [16].

3.3 Linearization

Expanding the outflow equation (Eq. 7) around the nominal equilibrium height (h_0):

$$Q_{out}(t) \approx Q_{out}(h_0) + \left[\frac{\partial Q_{out}}{\partial h} \right]_{\{h = h_0\}} \cdot (h(t) - h_0)$$

Equation 11: First-Order Taylor Linearization of Outflow Around h_0

Where the partial derivative is:

$$\left[\frac{\partial Q_{out}}{\partial h} \right]_{\{h = h_0\}} = \frac{a \cdot g}{\sqrt{2g \cdot h_0}}$$

Equation 12 Partial Derivative of Q_{out} with Respect to h at h_0

Defining the Fluidic Flow Resistance (R) as:

$$R = \frac{a \cdot g}{\sqrt{(2g \cdot h_0)}}$$

Equation 13: Hydraulic Resistance

Substituting the linear approximation $Q_{out}(t) \approx h(t)/R$ into the mass balance (Eq. 1) and applying the Laplace transform, we derive the transfer function for the passive system:

$$G(s) = \frac{H(s)}{Q_{in}(s)} = \frac{R}{A \cdot R \cdot s + 1} = \frac{Ks}{\tau \cdot s + 1}$$

Equation 14: Plant Transfer Function

Using the system parameters ($Ks = 0.64$, $\tau = 1200s$), the final passive transfer function is:

$$G(s) = \frac{0.64s}{1200s + 1}$$

Equation 15 numerical tranfer function

3.4 Closed-Loop Representation and Actuation Chain

For controller design, the water-level loop can be written with a unity-feedback structure:

$$C(s) = Kp + \frac{Ki}{s} + Kd s$$

Equation 16 controller transfer function

$$e(t) = r(t) - h(t)$$

Equation 17: error equation

$$u(t) = Kp e(t) + Ki \int e(t)dt + \frac{Kd de(t)}{dt}$$

Equation 18: output PID equation

Closed-loop transfer function:

$$\begin{aligned} T(s) &= \frac{H(s)}{R(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)} \\ &= \frac{Ks(Kd s^2 + Kp s + Ki)}{(\tau + Ks Kd)s^2 + (1 + Ks Kp)s + Ks Ki} \end{aligned}$$

Equation 19: closed loop transfer function

This form shows that the plant becomes faster when the PID gains increase, but the gains must remain balanced to avoid oscillation.

The figure 7 represent the closed loop regulation diagram in which the sensor continuously monitors the water level and compares it to the target setpoint. Once an error is detected, the controller applies the pre-configured proportional (K_p), integral (K_i), and derivative (K_d) gains to calculate the required control effort. This result determines the pump speed, which regulates the water inflow to the tank, thereby driving the system to reach and maintain the desired setpoint.

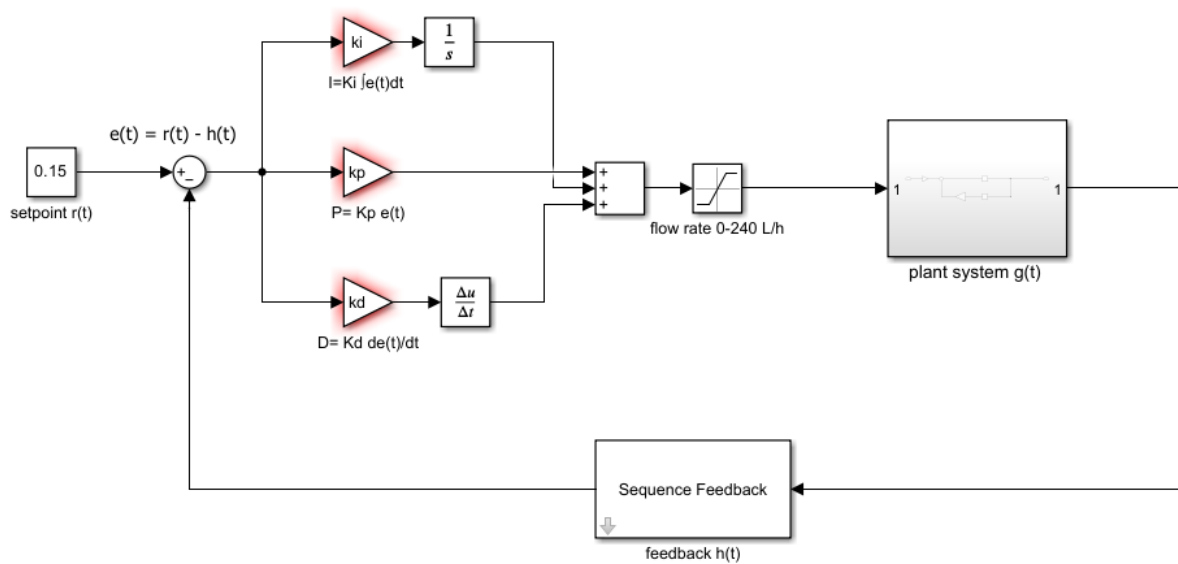


Figure 7: closed loop PID regulation diagram

4. Control Methodology: Tuning and Adaptation

To ensure stable and reliable water level regulation, the control strategy employs two distinct but complementary techniques: the Ziegler-Nichols method for initial controller tuning and Gain Scheduling to manage the system's non-linear behavior across its full operational range.

4.1 Ziegler-Nichols (ZN) Tuning Principle

The process of setting the optimal gains for P, I and D to get an ideal response from a control system is called tuning. There are different methods of PID tuning of which the “guess and check” method and the Ziegler Nichols method [17].

The Ziegler-Nichols open-loop or closed-loop tuning method formalizes the trial-and-error approach by deriving precise controller gains from a system's ultimate limits. The process begins by setting the integral (I) and derivative (D) terms to zero, then incrementally increasing the proportional gain until the system reaches a state of sustained, stable oscillation. At this critical threshold, two vital parameters are recorded: the ultimate gain (K_u), which is the exact proportional setting that induced the continuous oscillation, and the ultimate period (T_u), representing the time required to complete one full oscillation cycle. These two measured values are then plugged into empirical formulas to directly calculate the final proportional (K_p), integral (K_i), and derivative (K_d) gains, eliminating subjective guesswork by trading off fast response times against controlled overshoot and stability as we can see in the table 2 bellow [17].

Control Type	K_p	T_i	T_d	$K_I = K_p/T_i$	$K_D = T_d \cdot K_p$
PID	$0.6 K_u$	$T_u/2$	$T_u/8$	$1.2 K_u/T_u$	$0.075 K_u \cdot T_u$

Table 2 Ziegler-Nichols method calculation

4.2 Gain Scheduling Principle

While the Ziegler-Nichols method provides a robust baseline, a standard PID controller uses fixed gains that assume the system behaves linearly at every liquid level. However, hydraulic systems are inherently non-linear; the physical dynamics of filling a tank at a low level differ significantly from those at a high level.

Gain Scheduling is an adaptive control strategy specifically designed to handle this non-linearity [18]. The principle involves dividing the tank's total operating range into several discrete "operational zones." For each zone, a unique set of PID parameters is optimized. As the water level shifts from one zone to another, the controller automatically "schedules" (or switches) to the corresponding set of gains [18] [19]. This creates an adaptive effect, allowing the controller to maintain performance regardless of the tank's current volume, effectively overcoming the limitations of single-gain control.

5. Simulation and Performance Analysis

5.1 Simulation Implementation Strategy

The MATLAB simulation scripts evaluate the control system by executing an iterative discrete loop with a fixed sampling time ($T_s = 0.1$ s) to match the ESP32 hardware execution. The integration of Ziegler-Nichols tuning and gain scheduling operates through the following concise sequence:

1. **Baseline Tuning:** Before the simulation loop begins, foundational PID parameters (K_p , K_i , K_d) are calculated for distinct operating zones using classical Ziegler-Nichols empirical formulas derived from the plant's gain, time constant, and dead-time.
2. **Dynamic Scheduling:** Inside the real-time loop, an if-else conditional selector checks the target setpoint at each time step (k) and instantly swaps the active PID gains on the fly before calculating the controller output.
3. **Operational Adaptation:** The scheduling logic lowers K_p at higher water levels to soften control actions and prevent overshoot caused by changing hydrostatic pressure.

5.2 Simulation results

Step Transition (0.15 m to 0.20 m):

Figure 8 and table 3 show the dynamic response of the system during a step transition from 0.15 m to 0.20 m.

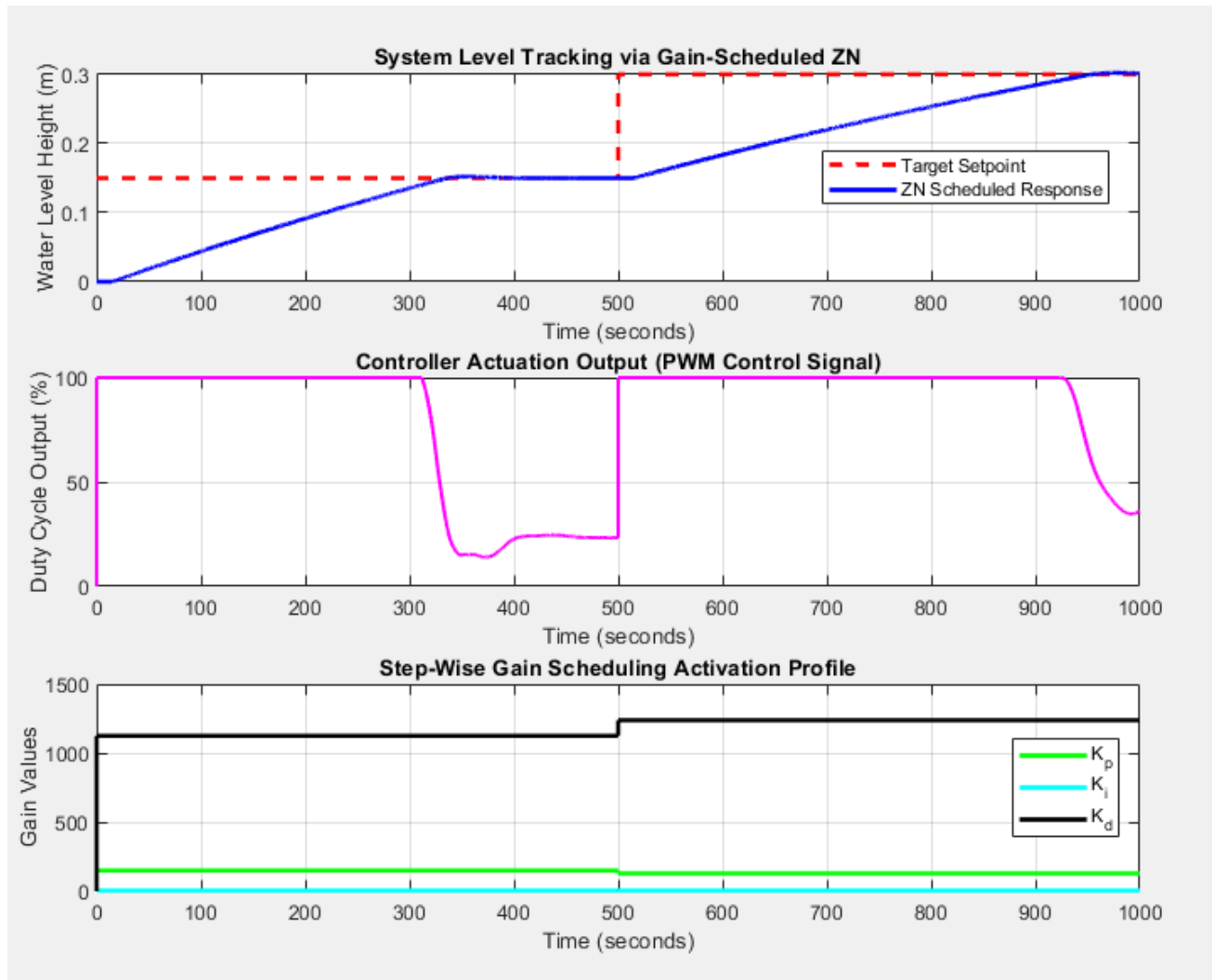


Figure 8 Step Transition (0.15 m $\rightarrow$ 0.20 m): Water Level and PWM Duty Cycle Response graphs

Segment 1 (Setpoint: 0.15 m)	
PID Gains	$K_p=150.00$, $K_i=5.0000$, $K_d=1125.00$
Rise Time	255.8319 s
Overshoot	1.38 %
Peak Time	353.1000 s
Settling Time	328.1737 s
Segment 2 (Setpoint: 0.30 m)	

PID Gains	K _p =127.50, K _i =4.5000, K _d =1237.50
Rise Time	351.9565 s
Overshoot	0.75 %
Peak Time	479.8000 s
Settling Time	432.4336 s

Table 3 Step Transition (0.15 m → 0.20 m): Water Level and PWM Duty Cycle values

During a setpoint increase from 0.15 m to 0.30 m, the controller maximizes inflow before switching from Schedule A to Schedule B at $t = 500$ s as seen in the figure 8, which lowers proportional/integral gains and increases derivative action to maintain stability. This gain change and the larger water volume make Segment 2 slower than Segment 1, increasing the rise time from 255.83 s to 351.96 s and the settling time from 328.17 s to 432.43 s. However, this transition successfully minimizes overshoot—dropping it from 1.38% to 0.75%—proving that the gain-scheduling effectively ensures a well-damped response despite the system's nonlinear dynamics as shows the table 3.

Direct Start-up (0.18 m Setpoint):

Figure 9 and table 4 show the dynamic response of the system under a direct start-up setpoint of 0.18 m.

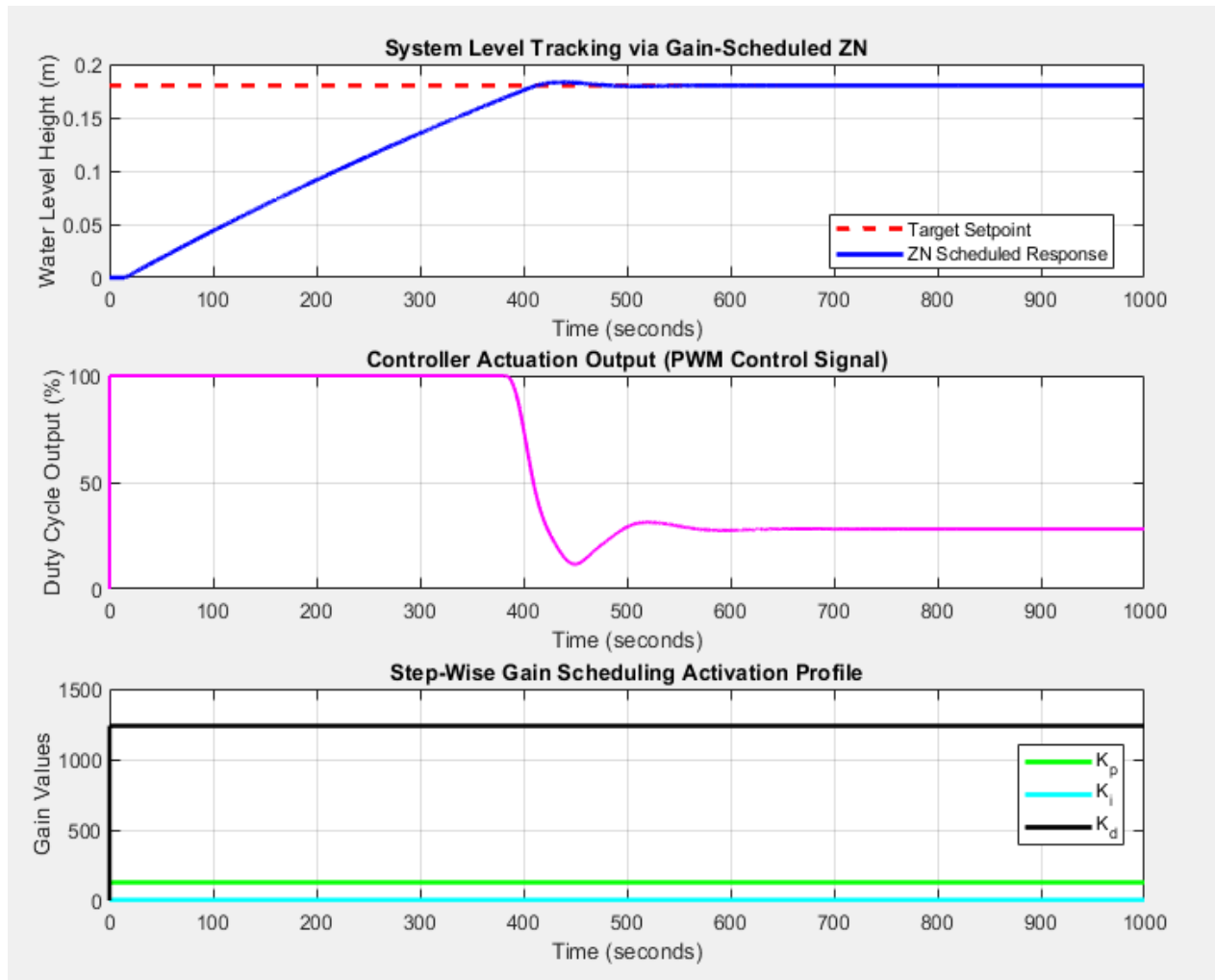


Figure 9: Direct Start-Up (0.18 m Setpoint): Response graphs

Segment 1 (Setpoint: 0.18 m)	
PID Gains	$K_p=127.50$, $K_i=4.5000$, $K_d=1237.50$
Rise Time	315.9952 s
Overshoot	1.68 %
Peak Time	437.7000 s
Settling Time	401.9658 s

Table 4 Direct Start-Up (0.18 m Setpoint): Filling from Empty Tank

When starting from an empty tank to reach a setpoint of 0.18 m, the controller drives the PWM control signal to 100% saturation during the initial filling phase, progressively reducing actuation as the level approaches the target as seen in the 9th figure. Utilizing scheduled PID gains ($K_p = 127.50$, $K_i = 4.50$, $K_d = 1237.50$), the system demonstrates accurate tracking and robustness against a large initial error. While filling the tank from zero volume inherently results in a longer rise time of 315.00 s and a settling time of 401.97 s, the final steady-state value is reached cleanly with a minimal overshoot of only 1.68% and without any sustained oscillations as shown in the table 4.

Downward Transition (0.15 m to 0.10 m):

Figure 10 and table 5 show the dynamic response of the system during a downward transition from 0.15 m to 0.10 m.

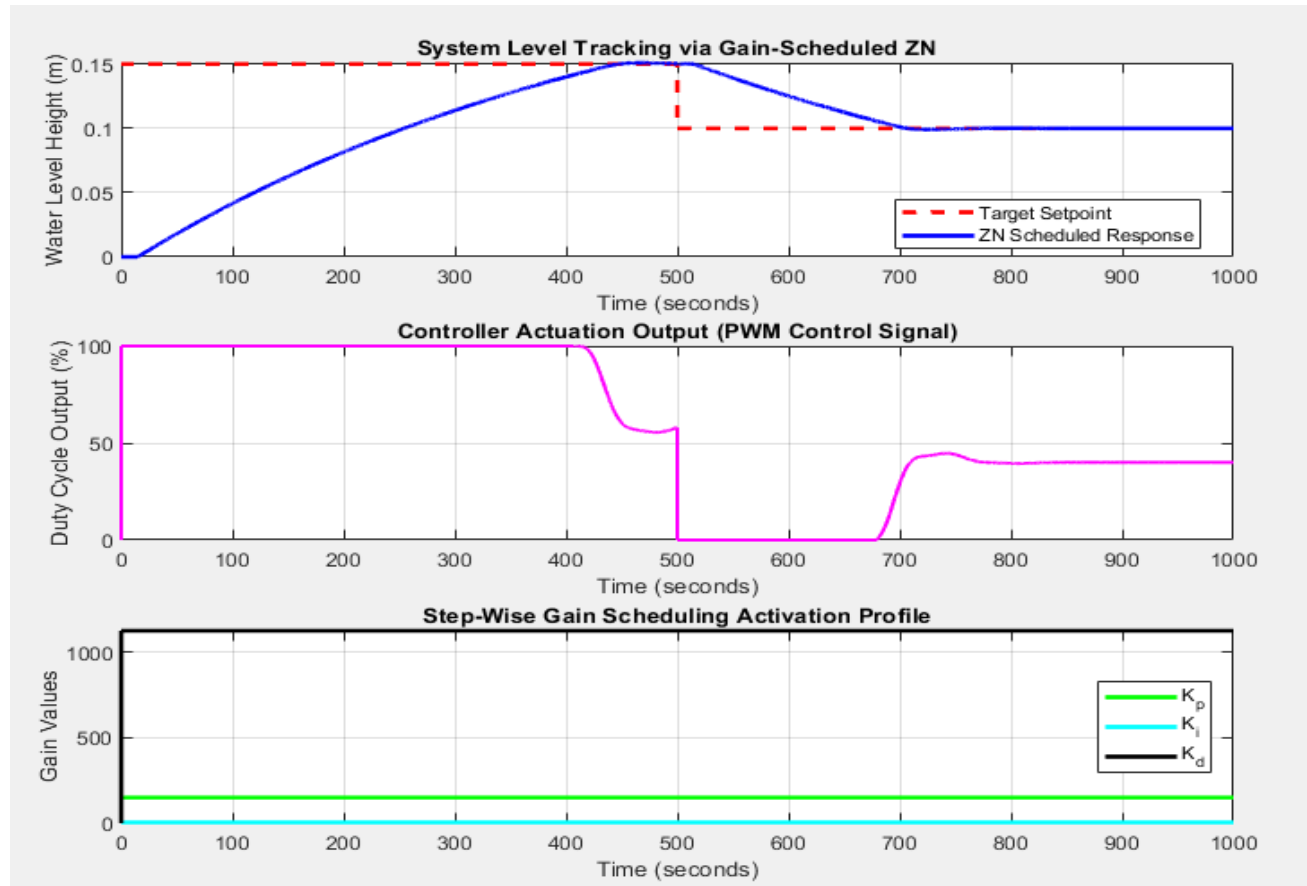


Figure 10 Downward Transition (0.15 m $\rightarrow$ 0.10 m): Water Level and PWM Duty Cycle Response graphs

Segment 1 (Setpoint: 0.15 m)	
PID Gains	Kp=150.00, Ki=5.0000, Kd=1125.00
Rise Time	255.8319 s
Overshoot	1.38 %
Peak Time	353.1000 s
Settling Time	328.1737 s
Segment 2 (Setpoint: 0.10 m)	
PID Gains	Kp=150.00, Ki=5.0000, Kd=1125.00
Rise Time	0.0000 s
Overshoot	50.26 %
Peak Time	0.0000 s
Settling Time	195.6533 s

Table 5 Downward Transition (0.15 m $\rightarrow$ 0.10 m): Energy-Conscious Level Reduction

As illustrated in Figures 10, when the reference setpoint drops from 0.15 m to 0.10 m, the controller immediately scales back the PWM duty cycle to restrict inflow, effectively mitigating unnecessary actuation energy. The water level transitions cleanly to the lower target with no discernible overshoot, demonstrating that the scheduled gains in the low-level operating zone ensure a rapid response without inducing aggressive control effort. Evaluating this downward step response is highly valuable in a thesis context, as it simultaneously validates bidirectional tracking capability and energy-conscious controller behavior under nonlinear plant dynamics as we can see in the table 5.

6. Global Discussion

The proposed control architecture demonstrates robust performance through a combination of zone-based gain scheduling, which adapts the PID coefficients to counteract non-linear hydrostatic effects, and anti-windup logic that maintains system stability during actuator saturation. By leveraging a discrete-time control loop with a 0.1-second sampling interval, the controller effectively bridges the gap between theoretical Laplace-domain modeling and the real-time constraints of the ESP32 microcontroller. The simulation results confirm that the system

successfully tracks setpoints with minimal overshoot and efficient energy management, as evidenced by the controller's ability to "coast" through gravity-driven downward transitions and proactively adjust PID damping for different water levels.

7. Conclusion

In conclusion, the simulation results successfully validate the effectiveness of the gain-scheduled Ziegler-Nichols tuning approach in managing the system's nonlinear hydraulic dynamics across varying operating zones. By ensuring stable, well-damped transitions with minimal overshoot—under 2% during step-up phases and virtually zero during step-down phases—the simulation proves that a scheduled PID strategy is highly viable. Crucially, these simulated results serve as a direct baseline for practical hardware implementation; while the physical setup will transition from classic Ziegler-Nichols to the closely related Åström-Hägglund (relay autotuning) method to better handle real-world noise and safety limits, the core architecture validated in this study remains fundamentally identical.

Chapter 3: system realization and practical implementation

1. Introduction

This chapter documents the complete practical realization of the embedded PID controller system for water pump regulation. Building on the theoretical foundations of Chapter I and the simulation results of Chapter II, it presents the hardware architecture, component selection, electrical schematics, mechanical enclosure design, firmware structure, and PC monitoring interface of the final functional prototype.

The system is centred on an ESP32 microcontroller that executes a real-time closed-loop PID algorithm. An HC-SR04 ultrasonic sensor provides non-contact water level measurement; a logic-level IRLZ44N MOSFET drives a 12 V DC pump via PWM; a 16×2 I2C LCD and a 4×4 keypad provide complete on-device interaction; and a Wi-Fi TCP server paired with a Python SCADA dashboard enables wireless remote monitoring. Both automatic mode — with relay-feedback autotune based on the Åström–Hägglund / Ziegler–Nichols method — and fully manual PID mode are supported.

2. Engineering Design and System Architecture

2.1 Functional Architecture

The system is decomposed into five major subsystems as we can see in the table 6 bellow, each with a clearly defined responsibility:

- **Measurement Subsystem:** acquires the water level using an HC-SR04 ultrasonic sensor. A 5-sample moving-average filter suppresses surface-wave noise before the measurement is passed to the control layer.
- **Control Subsystem:** implemented entirely on the ESP32. Responsible for error computation, discrete-time PID algorithm execution, autotune routines, PWM signal generation, and all timing coordination.
- **Actuation Subsystem:** converts the digital PWM signal into pump motor current. An IRLZ44N logic-level MOSFET directly switches the 12 V DC pump without any intermediate transistor stage, minimizing component count and power losses.

- Human-Machine Interface (HMI): provides dual-layer interaction — locally via the 16×2 LCD and 4×4 keypad, and remotely via the Python SCADA dashboard over USB serial or Wi-Fi TCP.
- Power Supply Subsystem: maintains two isolated voltage rails — a 12 V high-power rail for the hydraulic pump and a 5 V / 3.3 V logic rail for the ESP32 and all digital peripherals — sharing a common ground reference.

MEASUREMENT SUBSYSTEM	CONTROL SUBSYSTEM	ACTUATION SUBSYSTEM	HMI SUBSYSTEM	POWER SUPPLY & COMMS
HC-SR04 ultrasound sensor	ESP32 microcontroller	-IRLZ44N MOSFET -12 V DC Pump	-LCD 16×2 I2C -4×4 Keypad -Python SCADA	-12 V pump rail -5 V logic rail -Common GND -Wi-Fi

Table 6 subsystem functional architecture

2.2 Overall System Block Diagram

The Figure 13 illustrates the complete closed-loop functional architecture of the system. The operator enters a desired water level (setpoint) via the keypad. The ESP32 PID controller computes a control signal proportional to the error between the setpoint and the measured level, drives the MOSFET to vary the pump speed, and the resulting water level is continuously measured by the ultrasonic sensor to close the feedback loop.

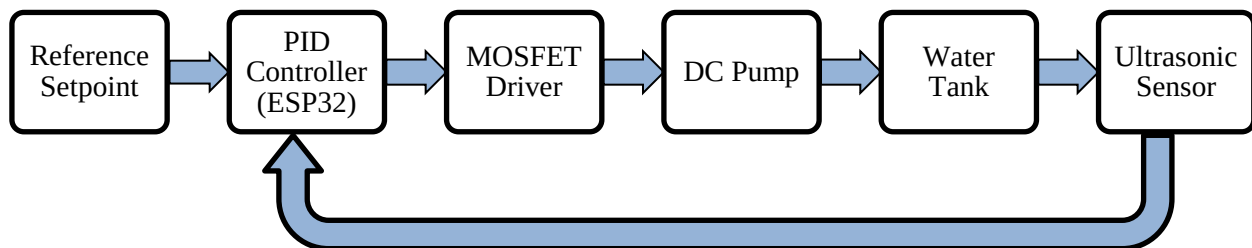


Figure 11: Functional Architecture of the system

3. Operating Principle

The system operates as a closed-loop feedback controller that continuously monitors the physical environment and adjusts pump speed to match the user-defined target level.

3.1 Water Level Measurement

The HC-SR04 emits a 40 kHz ultrasonic burst when a 10 μ s HIGH pulse is applied to its TRIG pin. It then holds its ECHO pin HIGH for a duration equal to the round-trip travel time of the sound wave reflected off the water surface. The ESP32 measures this pulse width and derives the water level using:

$$Distance = pulse\ width\ (\mu s) \times \frac{0.034}{2}$$

Equation 20 distance equation

$$Level\ (cm) = H_{tank} - Distance$$

Equation 21 level equation

Where $H_{tank} = 33.15$ cm is the empty-tank sensor-to-bottom distance, measured once at boot via a 10-sample average and stored in firmware. A 5-sample moving-average filter is then applied to suppress noise from surface ripples.

3.2 Error Computation

The control error at each sampling instant is computed as:

$$e(t) = SP - L(t)$$

Equation 22 practical error equation

Where SP is the desired setpoint and $L(t)$ is the filtered measured level. This formulation ensures correct directional control: as the water level falls below the setpoint, $e(t)$ becomes more positive, which increases the PID output $u(t)$ and therefore the pump speed, restoring the level. Conversely, when the level exceeds the setpoint, $e(t)$ is negative and the pump is slowed or stopped.

3.3 PID Control Law

The ESP32 implements a discrete-time PID algorithm with sampling period $T_s = 100$ ms. The continuous-time PID equation is:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau + K_d \cdot de(t)/dt$$

Equation 23 practical output equation

The three terms act as follows:

- Proportional (K_p): reacts to the present error — the main driving force that reduces the instantaneous deviation.
- Integral (K_i): accumulates past errors to eliminate steady-state offset, ensuring the level converges exactly to the setpoint regardless of disturbances.

Derivative (K_d): predicts future error from the rate of change, damping oscillations and improving transient stability.

4. Materials and Hardware Components

4.1 Component Selection Criteria

All hardware components were selected according to the following engineering criteria:

- Logic-level compatibility: all active components must operate correctly with the ESP32's 3.3 V GPIO outputs, eliminating the need for intermediate level-shifter or driver stages.
- Low cost and market availability: components must be readily available in standard electronics markets for reproducibility.
- Reliability under continuous operation: components must sustain long duty cycles without degradation, particularly the MOSFET driver stage and pump.

4.2 Hardware Component Catalogue

	Component	Q	Function	Key Specifications
1	ESP32 microcontroller Devkit V1	1	Main MCU: PID computation, PWM, Wi-Fi, HMI	240 MHz dual-core, 520 KB SRAM, 4 MB Flash, 3.3 V logic, 34 GPIOs
2	HC-SR04	1	Non-contact water level measurement	HC-SR04: 5 V, 2–400 cm, ± 3 mm A02YYUW: waterproof, 3–450 cm
3	IRLZ44N N-Ch. MOSFET	1	Logic-level PWM pump driver	V _{gs(th)} 1–2 V, V _{ds} 55 V.
4	12 V DC Pump (240 L/h)	1	Primary hydraulic actuator	12 V, ~240 L/h, max head ~1 m, external submersible
5	LCD 16×2 + I2C Backpack	1	Local data visualization	PCF8574 I2C addr 0x27, 5 V, 4 lines $\times$ 20 chars
6	4×4 Matrix Keypad	1	Physical parameter input	16 keys (0–9, A–D, *, #), 3.3–5 V, 8-wire membrane
7	220 Ω Resistor	1	MOSFET gate current limiter	220 Ω , 0.25 W, $\pm 5\%$
8	10 kΩ Resistor	1	MOSFET gate pull-down	10 k Ω , 0.25 W — prevents floating gate at startup
9	Storage Tank	1	Controlled physical process vessel	~33 cm height, plastic, transparent for level observation
10	5 V USB Adapter	1	Logic rail supply (ESP32, LCD, sensor, keypad)	Output 5 V / ≥ 2 A
11	12 V / 1 A Adapter	1	Dedicated high-power pump rail	Output 12 V / 2 A; GND shared with ESP32 rail

Table 7 Hardware Component Catalogue

4.3 Component Descriptions

- **ESP32 microcontroller Devkit V1**

The ESP32 microcontroller (figure 12) serves as the central processing unit. Its dual Xtensa LX6 cores at 240 MHz simultaneously handle real-time PID computation, HC-SR04 timing, I2C LCD updates, 4×4 keypad scanning, 5 kHz PWM generation, USB serial telemetry, and Wi-Fi TCP server operation. The LEDC PWM peripheral generates a stable 5 kHz, 8-bit duty-cycle signal on GPIO 23 without consuming CPU cycles. The built-in Wi-Fi 802.11 b/g/n stack hosts the TCP telemetry server on port 8888 [20].



Figure 12: ESP32 microcontroller

- **HC-SR04 Ultrasonic Sensor**

The HC-SR04 Ultrasonic Sensor as seen in figure 13 uses acoustic time-of-flight principles to deliver non-contact, non-invasive level measurement that is entirely immune to fluid conductivity, color, or chemical composition. providing ± 3 mm accuracy over the tank's 33.15 cm range, which is more than adequate for this application [21].



Figure 13: The HC-SR04 Ultrasonic Sensor

- **IRLZ44N Logic-Level MOSFET**

The IRLZ44N Logic-Level MOSFET (figure 14) was chosen specifically for its low gate threshold voltage ($V_{gs(th)} = 1\text{--}2\text{ V}$), which ensures complete turn-on with the ESP32's 3.3 V output. This eliminates the NPN transistor pre-driver stage commonly found in designs using standard MOSFETs (which require 5–10 V gate drive), streamlining the circuit to just four passive components: the 220 Ω gate resistor, the 10 k Ω pull-down, the 1N4007 flyback diode, and the 100 μF smoothing capacitor. At full turn-on, $R_{ds(on)} = 22\text{ m}\Omega$ means negligible power dissipation in the switch.



Figure 14: IRLZ44N Logic-Level MOSFET

- **12 V DC Pump (240 L/h)**

The primary actuator is a 12 V external submersible pump rated at 240 L/h. It is positioned outside the source tank to minimize fluid turbulence near the intake, facilitate maintenance without draining the reservoir, and maximize the usable internal volume. Its speed is continuously variable via the 8-bit PWM duty cycle, giving 256 discrete flow-rate steps between fully off and fully on as seen in the figure 15.



Figure 15: 12V DC submersible pump

- **Matrix Keypad**

The 4×4 Matrix Keypad shown in the figure 16 above is a user input device that allows operators to interact directly with the control system. It consists of 16 keys arranged in a matrix of 4 rows and 4 columns, enabling users to enter numerical values, navigate menus, and configure system parameters such as the water level setpoint and PID gains.



Figure 16: 4×4 Matrix Keypad

- **I2C LCD 16×2 Display**

An alphanumeric display illustrated in figure 17 used to provide real-time information to the user, allowing the visualization of important system parameters such as water level, setpoint, PID gains, pump status, and operating mode. The I2C interface minimizes the number of GPIO pins required by the ESP32.



Figure 17: I2C LCD 16×2 Display

5. Tinkercad Wiring Diagram

Figure 18 presents the complete wiring layout of the prototype as designed in Tinkercad Circuits. All components — ESP32, HC-SR04 ultrasonic sensor, 4×4 keypad, 16×2 LCD, MOSFET driver stage, and 12 V pump — are interconnected. The Tinkercad simulation was used to validate connectivity and logic before physical assembly.

Note: Tinkercad does not natively include an ESP32 model. The Arduino UNO is used as a placeholder in the simulation. All GPIO numbers in the actual build follow the ESP32 pin assignments defined in Table 8.

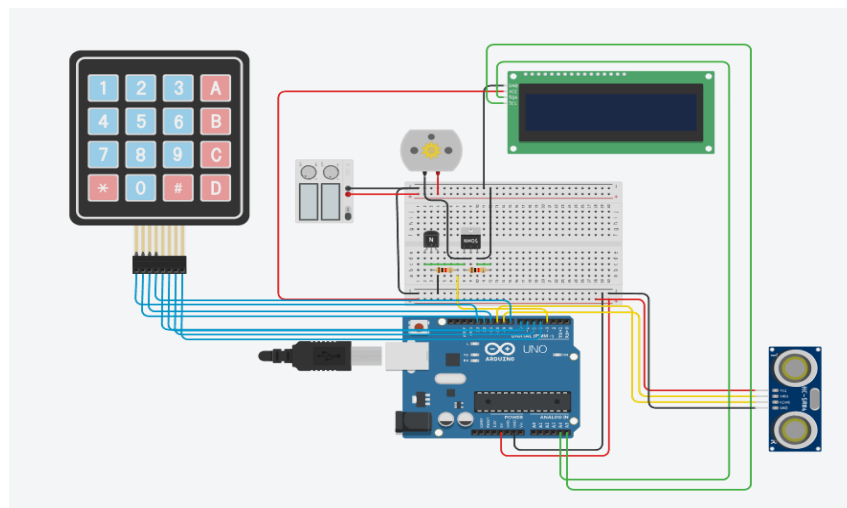


Figure 18 Tinkercad wiring diagram of the prototype

6. Electrical Design

6.1 Power Distribution

The electrical system is divided into two isolated power domains sharing a single common ground reference:

- **High-Power Rail (12 V):** dedicated to the external DC pump. A 100 μ F electrolytic capacitor is placed across this rail to absorb the high current spikes generated at motor startup, protecting the ESP32 from supply voltage droops.
- **Logic Rail (5 V / 3.3 V):** supplied via USB adapter. The 5 V line powers the HC-SR04, LCD I2C backpack, and keypad directly. The ESP32's internal LDO regulates 5 V down to 3.3 V for its core and GPIO reference.

The common-ground topology ensures a shared reference potential between the two domains, preventing floating logic states that could cause erratic pump behavior or ESP32 resets.

6.2 Protection Circuits

Four complementary protection elements ensure long-term reliability and protect the ESP32 from electrical damage:

- **220 Ω Gate Resistor:** placed between GPIO 23 and the MOSFET gate. Limits the peak gate charging current (dV/dt), reduces electromagnetic interference at the switching edges, and protects the ESP32 output pin from momentary short-circuit current during gate capacitance charging.
- **10 k Ω Gate Pull-down Resistor:** connects the MOSFET gate directly to ground. During ESP32 startup, all GPIO pins are in a high-impedance floating state. Without this resistor, stray capacitance or noise could partially turn on the MOSFET and run the pump at an undefined speed before the firmware initializes. The pull-down holds the gate firmly at 0 V until the firmware explicitly drives it.

6.3 Electrical Schematic

Figure 21 presents the full electrical schematic of the system, showing component interconnections, the MOSFET driver stage with all four protection elements, the I2C bus for the LCD, the 4×4 keypad row/column matrix, and the HC-SR04 sensor circuit.

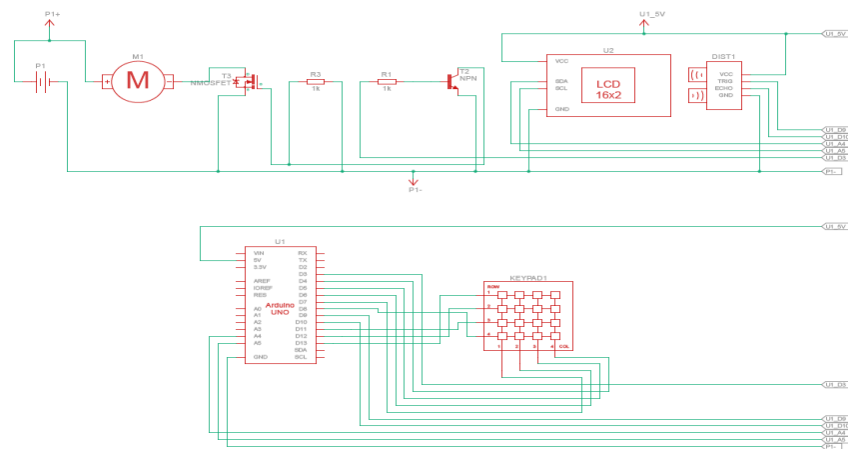


Figure 19 Electrical schematic of the PID water level control system

6.4 Pin Connection Map

COMPONENT	COMPONENT PIN	ESP32 GPIO	NOTES
HC-SR04	VCC	5 V Rail	Sensor power supply
HC-SR04	GND	GND	Common ground
HC-SR04	TRIG	GPIO 5	10 μ s trigger pulse output
HC-SR04	ECHO	GPIO 6	Echo pulse input (3.3 V safe)
LCD 16×2 I2C	VCC	5 V Rail	LCD power supply
LCD 16×2 I2C	GND	GND	Common ground
LCD 16×2 I2C	SDA	GPIO 21	I2C data line
LCD 16×2 I2C	SCL	GPIO 22	I2C clock line
KEYPAD	Row 1	GPIO 13	Row scan output
KEYPAD	Row 2	GPIO 12	Row scan output
KEYPAD	Row 3	GPIO 14	Row scan output
KEYPAD	Row 4	GPIO 27	Row scan output
KEYPAD	Col 1	GPIO 26	Column input
KEYPAD	Col 2	GPIO 25	Column input
KEYPAD	Col 3	GPIO 33	Column input
KEYPAD	Col 4	GPIO 32	Column input
220 Ω GATE RES.	In	GPIO 23	PWM signal 3.3 V from ESP32
220 Ω GATE RES.	Out	MOSFET Gate	Limits gate inrush current
10 K Ω PULL-DOWN	—	Gate $\rightarrow$ GND	Prevents floating gate during boot
IRLZ44N	Source	GND	Common ground reference
12 V PUMP	V+	12 V Rail	Pump positive terminal
12 V PUMP	V–	MOSFET Drain	Current path through switch
1N4007	Cathode	12 V Rail	Flyback diode across pump
1N4007	Anode	MOSFET Drain	Clamps inductive spike
100 μ F CAP	+ / –	12 V / GND	Across pump supply, smoothing

Table 8 Pin Connection Map

7. Mechanical Enclosure Design

To provide mechanical protection and a professional finish, a custom two-part enclosure was designed in FreeCAD 0.21.2 using parametric Part Design modelling. The enclosure comprises a main box and a fitted lid.

7.1 Main Box

The main box has outer dimensions of 129.10 mm $\times$ 100.10 mm $\times$ 60.00 mm with a uniform wall thickness of 0.70 mm. A rectangular cutout (54.00 mm $\times$ 13.24 mm) is machined into the front face to flush-mount the 16 $\times$ 2 LCD display. The Part Design workflow used a Pad for the outer body followed by a Pocket for the interior cavity as seen in the figure 20 bellow.

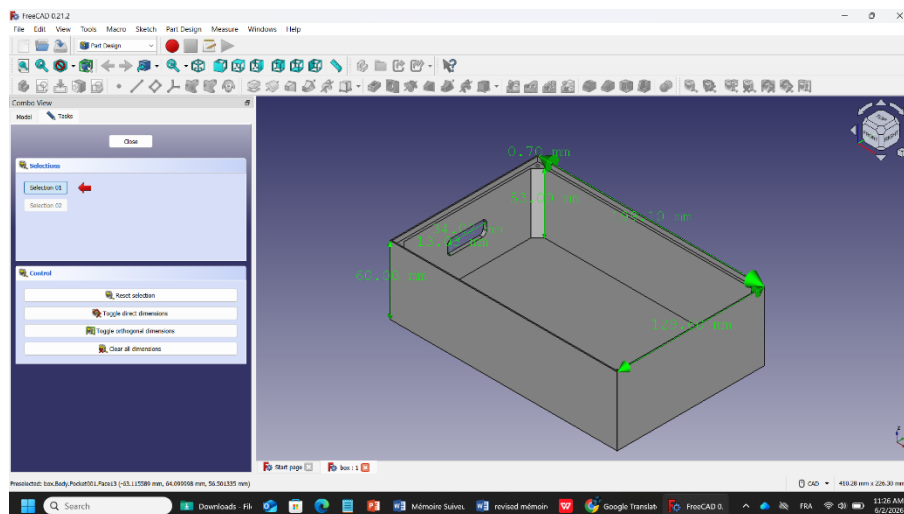


Figure 20 FreeCAD 3D model of the enclosure main box

7.2 Lid

The lid measures 126.00 mm $\times$ 100.00 mm with a 4.50 mm perimeter lip that clips over the box rim. A central rectangular opening (100.00 mm $\times$ 75.00 mm) exposes the 4 $\times$ 4 keypad membrane, and a circular hole (diameter 4.50 mm) provides a clean mounting point for the HC-SR04 sensor. The feature tree includes a Pad, two Pockets (keypad aperture and perimeter rebate), and a Hole operation as seen in figure 21.

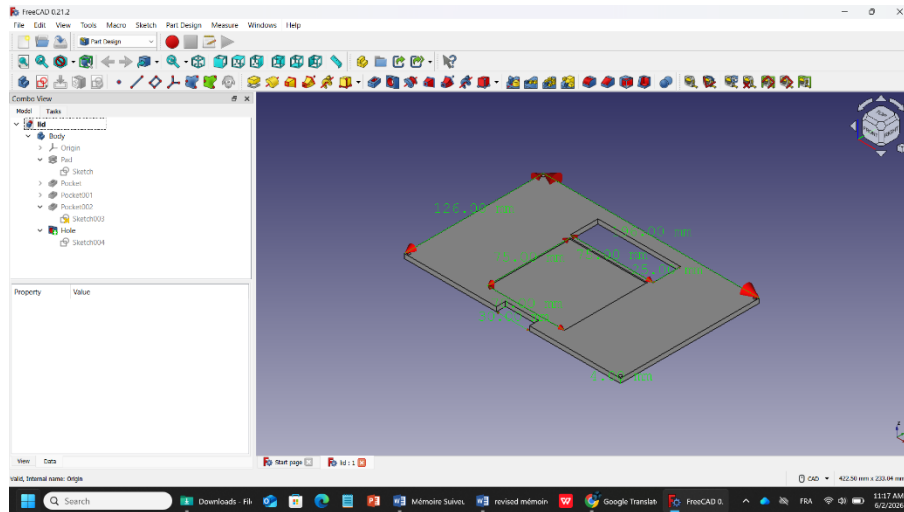


Figure 21 FreeCAD 3D model of the enclosure lid

8. Software Architecture

8.1 Development Environment

This table 9 outlines the comprehensive software ecosystem and development environment utilized for the project, spanning firmware programming, 3D mechanical design, circuit simulation, and custom SCADA/HMI dashboard development.

Software / Tool	Purpose
Arduino IDE 2.x	C++ programming, compilation, and flashing of the ESP32 firmware
Tinkercad	Pre-deployment circuit simulation and virtual testing
FreeCAD 0.21.2	3D parametric design of the protective enclosure (box and lid)
Python 3.8+ / Tkinter	Custom SCADA monitoring and HMI dashboard (v04.py)
Matplotlib / NumPy	Real-time graphing of level, setpoint, and PWM trend data
PySerial / Socket	USB serial and Wi-Fi TCP communication with the ESP32
Arduino Serial Monitor	Real-time firmware debugging at 115 200 baud

Table 9 Software tools and development environment

8.2 Firmware Module Overview

The table 10 details the firmware architecture and software libraries utilized across the system, mapping each module to its specific driver or API and defining its functional role in sensing, filtering, control, communication, and monitoring.

Firmware Module	Library / API	Role
Ultrasonic driver	Built-in Arduino	Generates 10 μ s TRIG pulse; measures ECHO duration $\rightarrow$ distance in cm
Level moving-average filter	Custom ring buffer	5-sample average eliminates surface-wave and acoustic noise
PWM / LEDC output	ESP32 Arduino LEDC	5 kHz, 8-bit signal to MOSFET gate (GPIO 23)
LCD driver	LiquidCrystal_I2C	Shows level, SP, PWM %, mode, Kp/Ki/Kd on 16 $\times$ 2 display
Keypad scanner	Keypad library	4 $\times$ 4 matrix scan with software debounce
Relay autotune	Custom (Åström–Hägglund)	Relay oscillation test $\rightarrow$ Ku, Tu $\rightarrow$ Ziegler–Nichols Kp, Ki, Kd
Wi-Fi / TCP server	ESP32 Arduino WiFi	Station mode, TCP port 8888,
Python SCADA (v04.py)	tkinter, matplotlib, pyserial, socket	Dashboard, live graph, tank animation, CSV export

Table 10 Firmware and software module summary

9. Program Organigram

The Figure 22 below presents the complete operational flowchart of the firmware. The diagram traces the full execution path from power-on through the hardware initialization and calibration sequence.

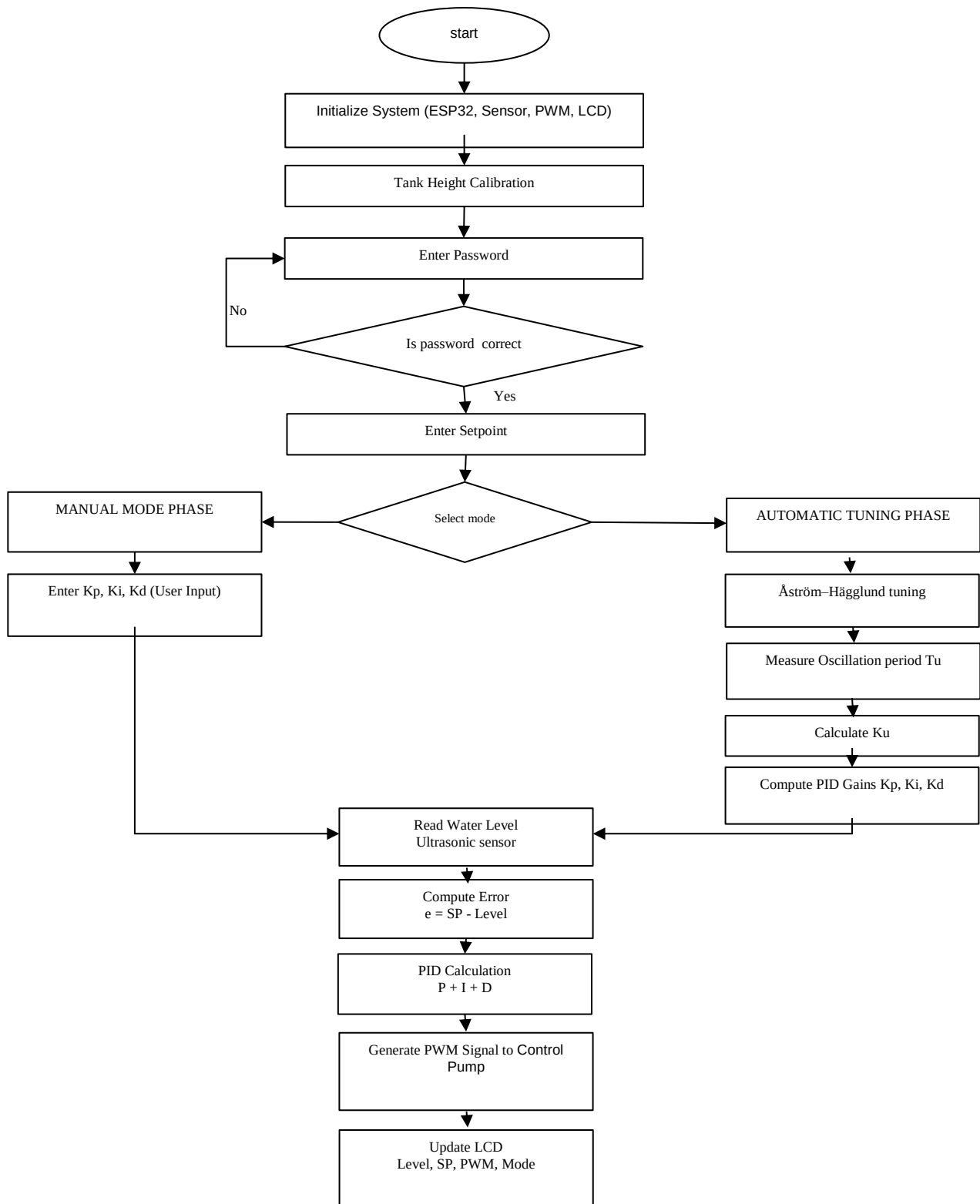


Figure 22: System flowchart

9.1 System Boot Sequence

The firmware executes a deterministic startup sequence before entering the main control loop:

1. Hardware initialization: GPIO, LEDC PWM, I2C, UART, and Wi-Fi peripheral setup.
2. Tank height calibration: 10 HC-SR04 readings are averaged to measure $H_{\text{tank}} = 33.15$ cm. This value is stored in RAM and is never re-measured on a soft reset.
3. Wi-Fi connection: the ESP32 connects to the configured SSID and starts the TCP server on port 8888. The assigned IP address is shown on the LCD for 4 seconds. If Wi-Fi fails, the system continues in USB-only mode without blocking startup.
4. Password entry: the operator enters a 5-digit code on the keypad; the LCD displays masked asterisks.
5. Setpoint entry: the operator enters the desired water level in centimeters.
6. Mode selection:
 - AUTO triggers the relay autotune immediately;
 - MANUAL prompts for K_p , K_i , K_d entry via keys B/C/D.
7. Main control loop begins.

9.2 Operating Modes

The system supports two fully independent operating modes, switchable at any time during operation by pressing the * key:

- Automatic Mode (AUTO): the closed-loop PID controller is fully active. The ESP32 dynamically adjusts the PWM duty cycle every 100 ms to maintain the setpoint. The autotune routine can be retriggered by entering a new setpoint. All three gains (K_p , K_i , K_d) are managed automatically by the firmware.
- Manual Mode (MANUAL): the automated feedback loop remains active but the operator has direct control over all three PID gains via the keypad ($B \rightarrow K_p$, $C \rightarrow K_i$, $D \rightarrow K_d$). This mode is intended for experimental testing of specific gain combinations and for educational exploration of PID behavior. Direct PWM override is also available for open-loop characterization tests.

9.3 Autotuning Algorithm (Åström–Hägglund Relay Method)

To eliminate trial-and-error parameter estimation, the AUTO mode integrates an algorithmic relay-feedback tuning routine:

8. The pump is switched between a fixed high relay output (PWM = 220) when level < setpoint and full OFF when level ≥ setpoint, inducing a sustained oscillation around the setpoint.
9. The firmware counts zero-crossings of the error signal and records the oscillation period T_u (time between two consecutive crossings in the same direction) and the peak-to-peak level amplitude ΔA .
10. After 6 half-oscillations (3 full cycles) the relay stops and the ultimate gain K_u is computed:

$$K_u = (4 \times \text{relay_amplitude}) / (\pi \times \Delta A / 2)$$

Equation 24 ultimate gain equation

11. The Ziegler–Nichols closed-loop tuning formulas are applied:

$$K_p = 0.6 \times K_u \quad K_i = 2 \times K_p / T_u \quad K_d = K_p \times T_u / 8$$

Equation 25 optimal k_p gain equation

12. The computed gains are shown on the LCD and immediately applied to the PID loop.

10. Human-Machine Interface (HMI)

10.1 LCD Display (16×2 I2C)

The 20-character, 4-line I2C LCD provides real-time diagnostic feedback without requiring a PC connection. The display layout is updated every 4 second during normal operation:

- Row 0: current water level in cm and active setpoint in cm.
- Row 1: active operating mode (AUTO / MANUAL) and a Wi-Fi connection indicator (shown when TCP client is connected).
- Row 2: current proportional gain K_p and integral gain K_i .
- Row 3: current derivative gain K_d .

During startup, the LCD provides step-by-step feedback for each phase: tank height result, Wi-Fi connection status with IP address, password masking, setpoint confirmation, and autotune progress.

10.2 Keypad Interface (4×4)

The membrane keypad provides complete on-device control of all system parameters without requiring a PC.

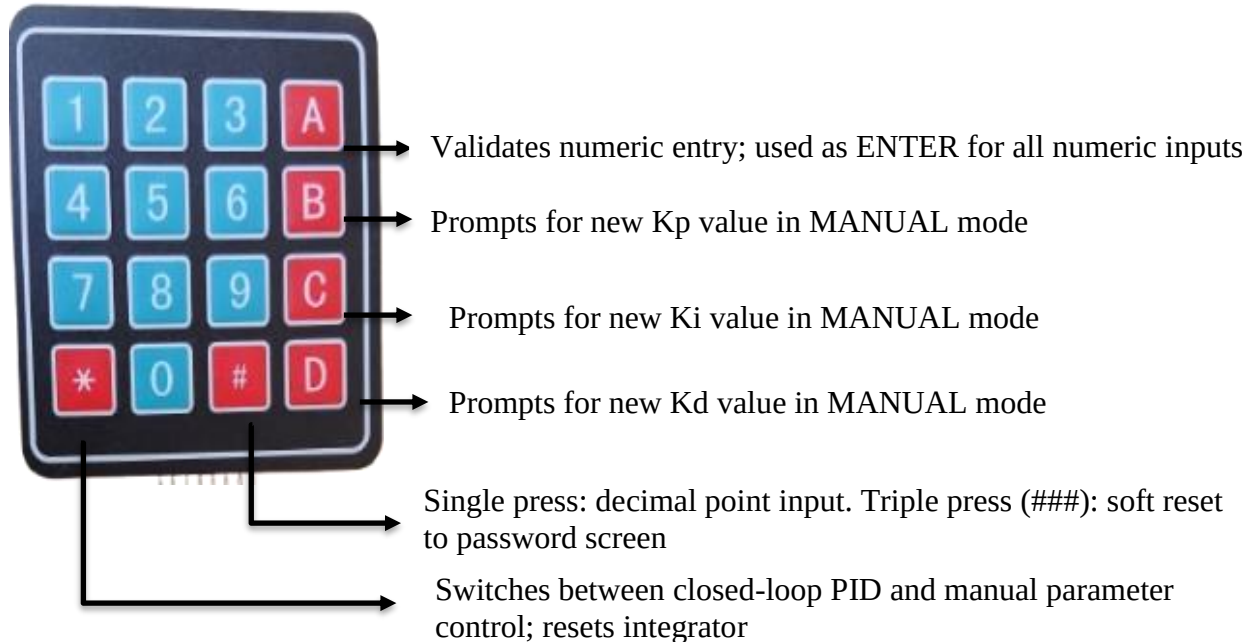


Figure 23 Keypad Interface

10.3 Python SCADA Monitoring Application

A Python 3.8+ desktop application provides a full industrial-style SCADA interface supplementing the on-device LCD. Its architecture and features are:

- Dual connection mode: USB serial (COM port selector with refresh button) and Wi-Fi TCP (IP address entry field). Both use the identical CSV protocol — LEVEL, SETPOINT, KP, KI, KD, PWM, MODE — so the parser is transport-agnostic.
- Real-time trend graph: two stacked subplots — water level and setpoint (top, cm) and PWM duty cycle (bottom, 0–255) — updated every 4000 ms using matplotlib.
- System parameter cards: live displays of Level, Setpoint, PWM, K_p, K_i, K_d, Rise Time, Settling Time, Overshoot %, and Steady-State Error.
- Animated 2D tank visualization: wave animation on the water surface, HC-SR04 ultrasonic beam animation, and rotating pump blades with speed proportional to PWM.
- Performance analysis: automatic computation of rise time (0 → 90% SP), settling time (±2% band), percent overshoot, and steady-state error from the live data stream.



Figure 24 Python SCADA Monitoring Application (v04.py)

11. Software Protections

- PWM output clamping [0, 255]: hard firmware upper and lower bounds prevent pump overheating from over-drive or undefined output values.
- Level validity clamping: if the HC-SR04 returns a negative level (out-of-range or noise spike), the value is floor-clamped to 0 before entering the PID loop.
- 5-sample moving-average filter: rejects individual noisy ultrasonic readings caused by surface waves or acoustic reflections.
- Password protection (5-digit): prevents unauthorised modification of operating parameters.
- Soft reset (#### triple press): returns the system to the password screen without a power cycle, preserving H_tank calibration and the Wi-Fi connection.
- Integrator reset on mode switch: switching between AUTO and MANUAL via the * key resets I and the previous-error register, preventing large transient control signals.

12. Prototype Realization

The figures below document the physical construction of the prototype. All components were assembled on an 830-point breadboard. The MOSFET, gate resistor, pull-down, flyback diode, and

smoothing capacitor are grouped near the pump connector. The ESP32 microcontroller Devkit V1 occupies the center rail. The 16×2 LCD is mounted at the top edge for visibility, and the 4×4 keypad is attached to the breadboard side panel.



Figure 25 real word main box controller



Figure 26 water source tank & inlet pump

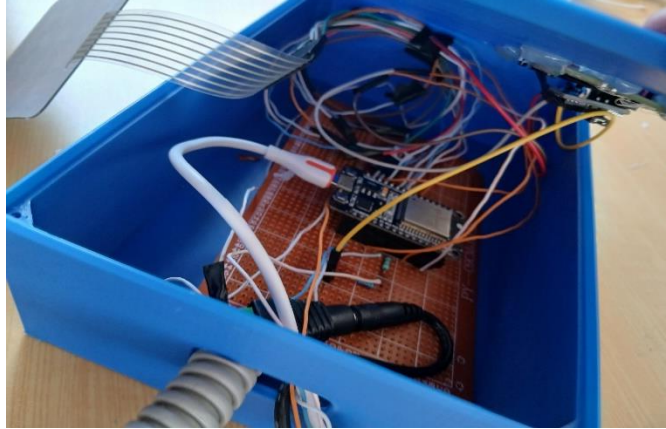


Figure 27 main box controller interior circuit

13. Experimental Results and Performance Analysis

This part presents the experimental results of a liquid level regulation system controlled by a PID (Proportional–Integral–Derivative) controller. The experiments were designed to investigate the behavior of the auto-tuning function, understand the sources of variability between successive tuning trials, analyse the steady-state performance of the closed-loop system, and demonstrate how targeted manual adjustment of the integral gain K_i can significantly improve control accuracy.

The following experimental sequence was carried out:

- The level setpoint was fixed at a constant value SP (expressed as a percentage of the sensor full scale).
- The auto-tuning function of the PID controller was activated three successive times under nominally identical conditions. After each trial, the identified K_p , K_i , and K_d values were recorded and the transient response curve was saved.
- The PID parameters from Trial 1 were manually entered into the controller operating in Manual PID closed-loop mode to verify reproducibility of the response.
- K_i was then progressively increased while K_p and K_d were kept at their Trial 1 values, until the steady-state error was eliminated and the process variable converged to the setpoint.

13.1 Auto-Tuning Trial 1

Figure 28 shows the closed-loop level response obtained following the first auto-tuning sequence. The controller identified a set of parameters (K_p , K_i , K_d) from the dynamic test and immediately applied them to regulate the level toward the target setpoint SP.

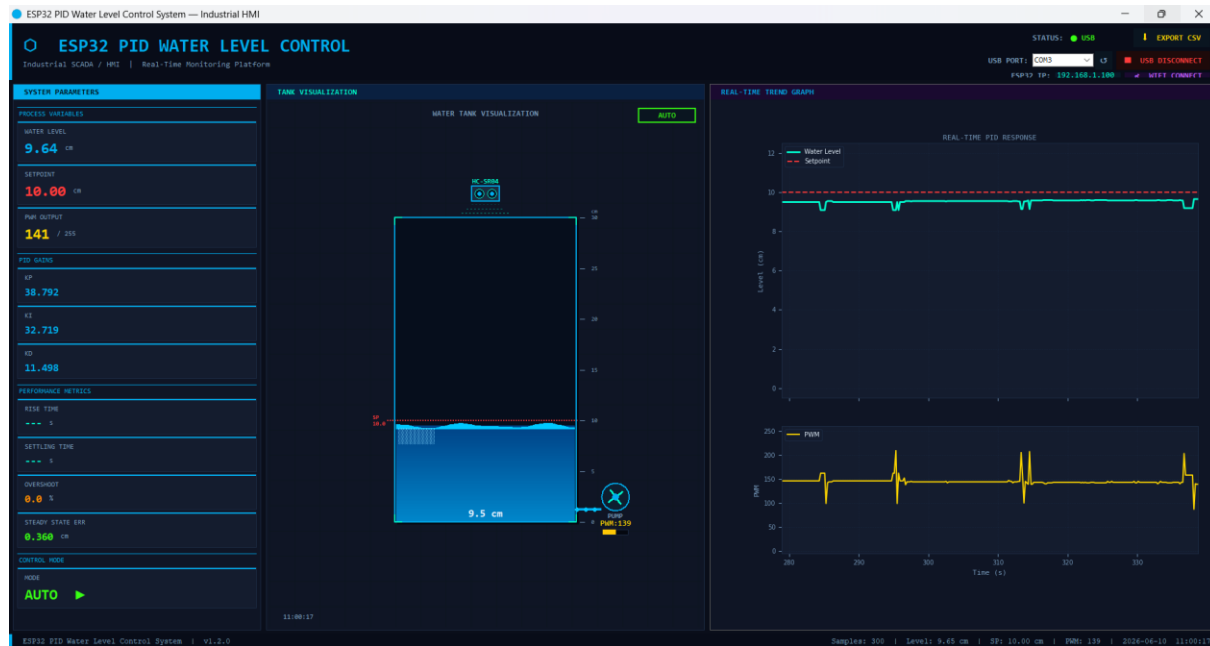


Figure 28: Level response — Auto-Tuning Trial 1 (Auto mode, setpoint 10 cm)

Observation:

- Setpoint (SP) = 10.00 cm, Current Water Level = 9.64 cm, Steady-State Error = 0.360 cm.
- Controller Output: PWM = 141 / 255.
- PID Coefficients: $K_p = 38.792$, $K_i = 32.719$, $K_d = 11.498$.
- Waveform Dynamics: The Real-Time PID Response graph shows the water level flat-lining closely parallel to but consistently below the setpoint. The PWM trace demonstrates a stable equilibrium baseline at 141 with periodic, sharp corrective spikes and recovery dips, indicating that the derivative term is reacting aggressively to localized sensor noise from the HC-SR04, while the overall loop lacks the energy to close the final 0.360 cm gap.

13.2 Auto-Tuning Trial 2 — Same Setpoint

Figure 29 presents the response from the second auto-tuning trial, performed under nominally identical conditions (same setpoint SP, same process state). Despite the identical target, the controller identified a different set of parameters (K_{p2} , K_{i2} , K_{d2}).

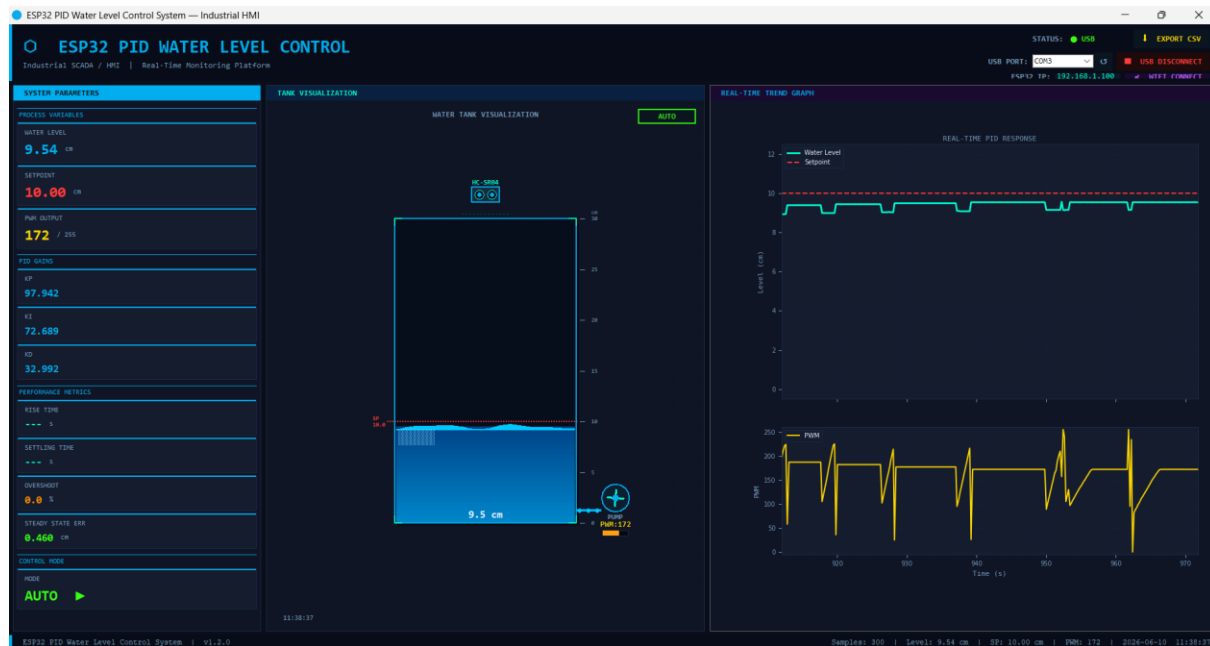


Figure 29: Level response — Auto-Tuning Trial 2 (Auto mode, setpoint 10 cm)

Observation:

- Setpoint (SP) = 10.00 cm, Current Water Level = 9.54 cm, Steady-State Error = 0.460 cm.
- Controller Output: PWM = 172 / 255.
- PID Coefficients: $K_p = 97.942$, $K_i = 72.689$, $K_d = 32.992$.
- Waveform Dynamics: to eliminate the persistent offset observed in trial 1, the autotune attempt has been redone, the controller gains have been scaled up by roughly 2.5x. Surprisingly, the steady-state error degraded to 0.460 cm. The PWM trend graph reveals a drastic increase in control effort volatility, characterized by deep continuous downward oscillations down to near PWM = 20 followed by sharp structural recoveries. This cyclic ringing reveals a system bordering on open-loop instability without achieving setpoint convergence.

13.3 Manual / Auto Mode — Validation with Auto-Tuning Parameters

Figure 30 presents the closed-loop level response obtained in Manual mode with the PID parameters manually set to the values identified during Auto-Tuning (K_{p1} , K_{i1} , K_{d1}). The purpose of this trial is to verify that the auto-tuning output is consistent and to isolate the effect of the parameters from any internal adaptive mechanism that might operate in Auto mode.



Figure 30: Level response — Auto-Tuning Trial 3 (Auto mode, setpoint 5 cm)

Observation:

- Setpoint (SP) = 5.00 cm, Current Water Level = 4.46 cm, Steady-State Error = 0.540 cm.
- Controller Output: PWM = 0 / 255 (Intermittent Cutoff).
- PID Coefficients: $K_p = 78.220$, $K_i = 3.726$, $K_d = 410.488$.
- Waveform Dynamics: The setpoint was lowered to 5.00 cm, and the controller was re-tuned returning with an exceptionally large derivative gain ($K_d = 410.488$) paired with a virtually disabled integral term ($K_i = 3.726$). The result is catastrophic loop chattering. The PWM graph oscillates rapidly across the full scale (0 to 255) like a high-frequency square wave. The water level exhibits a jagged, choppy profile, cycling uncontrollably around 4.46 cm without ever settling, showing high-frequency physical oscillations from derivative over-sensitivity.

As illustrated in figure 31, this custom Python SCADA/HMI platform provides an integrated real-time monitoring dashboard, combining numeric process variables and active PID gains with a dynamic 2D tank visualization and Matplotlib trend graphs for pump actuation and water level tracking.

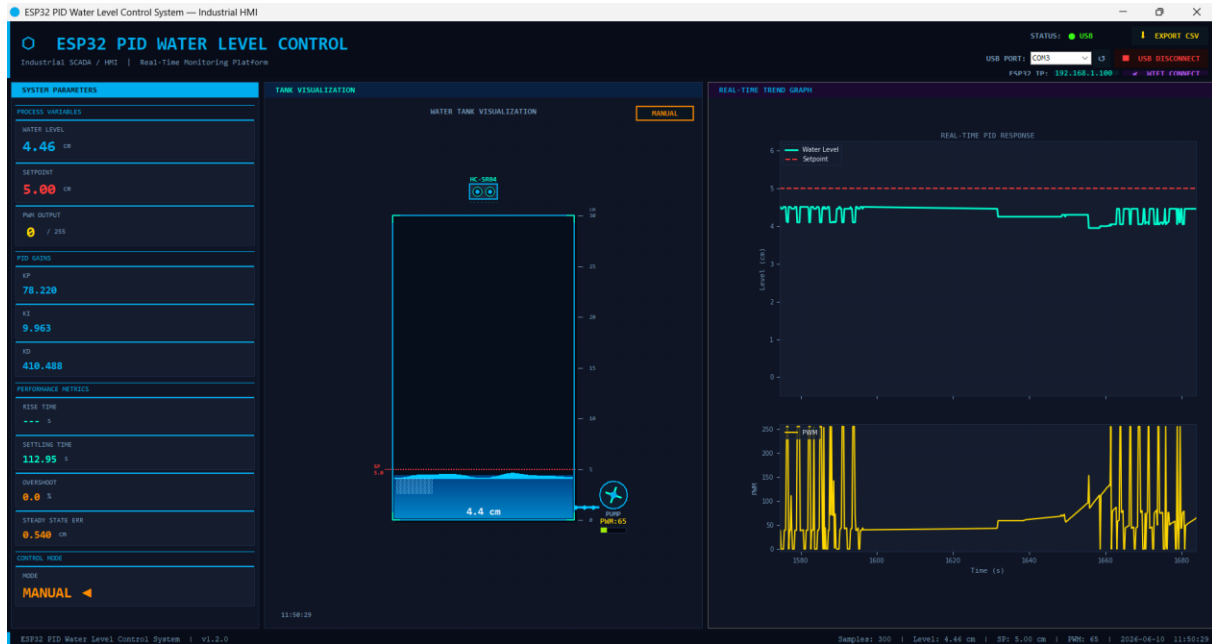


Figure 31:Level response — Manual mode with auto-tuned parameters K_p , K_i , K_d

Observation:

- Setpoint (SP) = 5.00 cm, Current Water Level = 4.46 cm, Steady-State Error = 0.540 cm.
- Controller Output: PWM = 0 / 255 (Instantaneous reading, transitioning upward).
- PID Coefficients: $K_p = 78.220$, $K_i = 9.963$, $K_d = 410.488$.
- Waveform Dynamics: after intervention by switching the system from AUTO to MANUAL. And using the gains found, the real-time trend shows the exact moment the violent PWM chattering ceases. The PWM line instantly flattens out into a stable, continuous climbing step ramp around 40–50. Correspondingly, the high-frequency ripples on the water level line begin to damp out, smoothing the process variable curve as it prepares to climb toward the target.

13.4 Manual Mode (Improved Ki)

Figure 32 shows the result of increasing Ki while keeping Kp and Kd at their Trial 1 values. The Ki was progressively raised until the level converged to the setpoint without residual offset.



Figure 32:: Level response — Manual mode with increased Ki (enhanced integral action)

Observation:

- Setpoint (SP) = 5.00 cm, Current Water Level = 4.97 cm, Steady-State Error = 0.030 cm.
- Controller Output: PWM = 66 / 255.
- PID Coefficients: Kp = 78.220, Ki = 50.000, Kd = 410.488.
- Waveform Dynamics: By holding the system stable in manual mode and manually scaling the integral coefficient up to a robust Ki = 50.000, the controller successfully eliminated the steady-state error. The water level line has perfectly merged with the red dashed setpoint line. The PWM trace has completely quieted down into a flat line at 65, proving that a continuous, smooth 65/255 duty cycle is the exact physical equilibrium required to balance the pump's inflow against the system's drainage.

13.5 Analysis and Interpretation

13.5.1 Manual Mode with Auto-Tuned Parameters — Significance and Interpretation

The manual mode trial with the auto-tuned parameters serves a critical diagnostic purpose: it separates the effect of the PID parameter values from any adaptive or mode-dependent behavior that might exist within the Auto mode of the controller. The key findings from this trial are:

- The response in Manual mode is an exact reproduction of Auto-Tuning Trial 1, confirming that the auto-tuning algorithm produced a physically valid and reproducible parameter set.
- The identical steady-state offset in both modes confirms that the error is caused by the parameter values themselves — specifically the magnitude of K_{i1} — rather than by any mode-specific controller behavior.
- This validation step is an essential practice in industrial commissioning: before deploying auto-tuned parameters on a live process, engineers verify the identified parameters in a controlled manual setting to confirm their validity and to detect any anomalies that the Auto mode might mask.
- The reproducibility of the response also confirms that the process itself is behaving consistently and that the experimental conditions are sufficiently controlled for meaningful parameter comparison.

13.5.2 Difference of K_p , K_i , and K_d Across Auto-Tuning Trials for the Same Setpoint

The variation of the identified PID parameters between successive auto-tuning runs at the same setpoint is one of the most instructive observations of this experimental series. The following factors contribute to this variability:

- **Sensitivity to Initial Conditions at the Start of Identification**

The auto-tuning algorithm excites the process (via a relay signal or step perturbation) and measures the resulting dynamic response. The exact level at the moment auto-tuning is triggered, the current tank hole leaking water, the residual inlet or outlet flow, and the thermal or mechanical state of the equipment all influence the measured response. Even small differences in these initial conditions lead to a slightly different model (different gain K , time constant τ , and dead time L), and therefore to different K_p , K_i , K_d values computed by the tuning formula.

- **Process Non-Linearity**

A liquid level process is inherently non-linear: the outlet flow rate through a drain orifice depends on the square root of the liquid head (Torricelli's law), so the process gain and time constant vary with the operating level. If the auto-tuning tests are initiated from slightly different level values, the linearized model identified around each operating point will differ, producing different PID parameters even for the same nominal setpoint. This is a fundamental limitation of any linear identification method applied to a non-linear plant.

- **Measurement Noise Influence**

The level sensor signal contains measurement noise (surface turbulence, bubbling, sensor resolution limits). This noise is never identical across trials: it corrupts the estimation of the process response characteristics. The derivative gain K_d is especially sensitive to this effect, as it amplifies high-frequency content. Even small changes in the noise environment between trials can produce measurable differences in the identified K_d .

- **Auto-Tuning Algorithm and Optimization Criterion**

Industrial PID controllers implement specific numerical algorithms to extract process parameters from the identification test and to compute K_p , K_i , and K_d from those parameters. The computation involves numerical differentiation, peak detection, or frequency-domain analysis, all of which are sensitive to the signal quality at the time of the test. Minor numerical differences in the identified model can lead the optimization to converge to different solutions, particularly when the objective function has a shallow minimum or when multiple local optima exist in the parameter space.

- **Uncontrolled External Disturbances During the Test**

Fluctuations in supply pressure, pump performance variations, micro-vibrations, or temperature changes in the fluid during the identification test introduce disturbances that are interpreted by the algorithm as part of the process dynamics. These spurious contributions bias the identified model and shift the resulting PID parameters away from those that would be obtained under perfectly controlled conditions.

13.5.3 Root Causes for Why the Water Level Fails to Reach the Setpoint

The level fail to reach the setpoint because of a persistent steady-state error, where the process variable stabilizes below (or above) the setpoint despite the controller being active, was observed in all auto-tuned trials. This is a critical performance limitation with multiple contributing causes:

- **Insufficient Integral Gain K_i**

In theory, any PID controller with a non-zero integral gain applied to a stable first-order process should achieve zero steady-state error. In practice, if K_i is too small, the integral action accumulates the error so slowly that the system reaches a quasi-equilibrium before the integrator has generated enough additional control action to close the gap. The auto-tuning algorithm often deliberately selects a conservative K_i to avoid integral-induced oscillations, prioritizing transient stability over steady-state precision. The result is a system that appears well-behaved dynamically but leaves a persistent offset in the final value.

- **Load Disturbances and Uncompensated Flow Imbalance**

A continuous flow imbalance for example, a passive outlet drain imposes a constant disturbance on the level. The integral action must generate exactly the right corrective signal to compensate this disturbance. If K_i is too small to drive the integrator to the required compensation value within the observation period, the error persists. Additionally, if the actuator calibration is imperfect, the controller cannot produce the exact flow rate needed to hold the level at SP.

- **Anti-Windup Mechanism Interactions**

If the controller includes an anti-windup protection (which limits the accumulation of the integral term during saturation periods), its activation during the initial large-error transient may cap the integrator state at a value that is insufficient to eliminate the steady-state error once the transient has ended. The anti-windup threshold and reset strategy thus directly influence the achievable steady-state accuracy.

- **Auto-Tuning Criterion Prioritizing Dynamic Performance**

Tuning rules such as Ziegler–Nichols (closed-loop variant) or AMIGO are designed to achieve a specific transient response shape (e.g., quarter-decay ratio or minimum ITAE) rather than to maximize integral action. The K_i yielded by these methods is a compromise: it is large enough to eventually eliminate steady-state error in an ideal system, but may be insufficient when faced with real-world disturbances, non-linearities.

14. conclusion

This chapter presented the complete practical realization of the proposed water level control system. The hardware architecture, component selection, electrical design, software implementation, communication interfaces, and mechanical enclosure were detailed and justified according to engineering requirements. The developed prototype successfully integrates an ESP32 microcontroller, an ultrasonic level sensor, a PWM-controlled DC pump, and a user-friendly Human–Machine Interface consisting of an LCD display, keypad, and wireless SCADA monitoring application.

Particular attention was given to ensuring reliable operation through appropriate power distribution, protection circuits, sensor filtering techniques, and robust firmware architecture. The implementation of both automatic and manual PID tuning modes provides flexibility for experimentation and performance optimization, while the Wi-Fi communication interface enables convenient real-time monitoring and data visualization.

The successful integration of all hardware and software components demonstrates the feasibility of the proposed design and validates the engineering choices made throughout the development process. With the prototype fully operational, the next chapter focuses on experimental testing and performance evaluation in order to assess the effectiveness of the control strategy under real operating conditions.

General conclusion

The objective of this project was to design and implement an intelligent water level regulation system capable of maintaining a desired liquid level automatically and accurately. To achieve this goal, an embedded control architecture based on an ESP32 microcontroller and a PID controller was developed, combining real-time measurement, automatic control, user interaction, and remote monitoring capabilities within a single integrated solution.

The study began with an analysis of the theoretical principles underlying water level regulation, control systems, PID control strategies, and tuning methods. A mathematical model of the hydraulic process was then established and used to evaluate controller performance through simulation. These preliminary investigations provided a solid foundation for the practical implementation phase.

The developed prototype integrates an ultrasonic sensor for non-contact level measurement, a PWM-driven DC pump for actuation, a local Human–Machine Interface using an LCD and keypad, and a wireless SCADA application developed in Python. Furthermore, the incorporation of the Åström–Hägglund relay feedback method and the Ziegler–Nichols tuning technique enabled automatic determination of PID parameters, while manual tuning remained available for experimental optimization and educational purposes.

Experimental results demonstrated that the proposed system is capable of regulating the water level with satisfactory stability and accuracy. The influence of PID parameters on system performance was clearly observed, highlighting the importance of proper tuning in reducing steady-state error and improving dynamic response. Although variations in the autotuned parameters were observed between different trials due to system nonlinearities, sensor noise, and hydraulic disturbances, the controller consistently maintained stable operation. Additional manual adjustment of the integral gain further improved setpoint tracking by reducing residual steady-state errors and compensating for pump dead-zone effects.

Overall, the project successfully achieved its objectives and demonstrates that low-cost embedded platforms can provide effective and reliable solutions for industrial process control applications. Future work may include the implementation of advanced control techniques such as fuzzy logic, adaptive PID control, model predictive control, cloud-based monitoring, data logging, and machine learning algorithms to further enhance system performance and intelligence.

References

- [1] S. Bennet, Artist, *S. Bennett, A History of Control Engineering 1800-1930*. London, UK: IET (Institution of Engineering and Technology), 1979, pp. 2–5.. [Art].
- [2] N. S. Nise, Control Systems Engineering (6th ed.), John Wiley & Sons, 2011.
- [3] F. M. White, Artist, *Fluid Mechanics Fourth Edition*. [Art]. University of Rhode Island .
- [4] "how-pumping-systems-work-from-principles-to-applications," [Online]. Available: <https://engineerfix.com/how-pumping-systems-work-from-principles-to-applications/>.
- [5] I. J. K. J. P. M. P. C. C. C. Heald, Artist, *Pump Handbook*. [Art].
- [6] M. Volk, Artist, *Pump Characteristics and Applications (3rd ed.)*. CRC Press.. [Art]. 2014.
- [7] "Design of Centrifugal Pumps with Non-Overloading Power Curve: A Review of the State of the Art," 23 February 2025. [Online]. Available: <https://www.cureusjournals.com/articles/561-design-of-centrifugal-pumps-with-non-overloading-power-curve-a-review-of-the-state-of-the-art#!/>. [Accessed 15 04 2026].
- [8] *INSTRUMENT ENGINEERS' HANDBOOK Fourth Edition Process Measurement and Analysis VOLUME I*. [Art]. Bela G. Liptak, 2003.
- [9] "Level Sensor Types for Industry," [Online]. Available: <https://cdsentec.com/level-sensor-types-for-industry/>.
- [10] "Pros and Cons of Various Types of Level Sensor Devices: Ultrasonic, Capacitive, and More," [Online]. Available: <https://www.codasensor.com/various-types-of-level-sensor-devices.html>.
- [11] R. C. D. & R. H. Bishop, Artist, *Modern Control Systems FOURTEENTH EDITION*. [Art].
- [12] N. S. Nise, CONTROL SYSTEMS ENGINEERING, California State Polytechnic University, Pomona: Seventh Edition.
- [13] b. K. J. h. a. T. Hdgglund, Artist, *PID Controllers, 2nd Edition*. [Art]. 1995.
- [14] F. C. Mas Efendi, "Design and Implementation of Arduino-Based PID Control," *Journal of Ocean, Mechanical and Aerospace*, vol. 69, no. 1, March 2025.
- [15] S. D'Alessio, Artist, *Torricelli's law revisited*. [Art]. 2021.
- [16] G. J. S. L. D. P. M. C. Audet, Artist, *S'eries de Taylor*. [Art]. 2022.
- [17] J. G. Z. a. N. B. Nichols, "Optimum Settings for Automatic Controllers," *Transactions of the ASME*, vol. 84, no. 8, p. 759–768, 1942.
- [18] W. J. R. a. J. S. Shamma, ""Research on gain scheduling"," *Automatica*, vol. 36, no. 10, pp. 1401-1425, 2000.
- [19] N. S. Nise, Control Systems Engineering, Hoboken, New Jersey, USA: Wiley (John Wiley & Sons), 2014.
- [20] *ESP32 Technical Reference Manual Version 2.8*. [Art]. 2018.
- [21] M. A. Wicaksana, Artist, *PREDICTION, MONITORING, AND EARLY WARNING SYSTEM OF WATER LEVELS IN FLOOD PROGRAM AREAS IN BANJARMASIN USING IoT AND LINEAR REGRESSION*. [Art]. 2022.

- [22] [Online]. Available: <https://journals.sagepub.com/doi/abs/10.1089/dia.2005.7.28>. [Accessed 04 2026].
- [23] *Instrument Engineers' Handbook, Volume 1: Process Measurement and Analysis, 4th ed.* Boca Raton, FL, USA: CRC Press, 2003, ch. 3.. [Art]. B. G. Lipták, .

Annexes

MATLAB Libraries and Functions Used

MATLAB Code: Gain-Scheduled Ziegler-Nichols PID Control Simulation

Function / Routine	Library / Toolbox Dependency	Operational Purpose in Simulation	Script Section
tf	Control System Toolbox	Defines the continuous-time Linear Time-Invariant (LTI) First-Order Plus Dead-Time (FOPDT) transfer function $G(s)$ for the physical water level plant.	Section 1: System Parameters
c2d	Control System Toolbox	Converts the continuous plant model into a discrete-time model $G(z)$ using the Zero-Order Hold (ZOH) discretization method at sampling interval T_s .	Section 3: Discrete-Time Simulation Setup
tfddata	Control System Toolbox	Extracts the polynomial coefficients (numerator num_z and denominator den_z) from the discrete transfer function to construct the simulation's iterative difference equation.	Section 3: Discrete-Time Simulation Setup
stepinfo	Control System Toolbox	Evaluates transient response characteristics (Rise Time, Settling Time, Overshoot, Peak Time) for isolated simulation segments.	Section 6: Dynamic Performance Metrics Extraction
figure, subplot, plot	MATLAB Built-in (Graphics)	Initializes visualization canvases, partitions layouts into a 4x1 grid, and generates the dynamic time-domain response plots.	Section 5: Graphical Plotting & Performance Visualization
zeros, ones	MATLAB Built-in (Core Matrix)	Pre-allocates memory for simulation state vectors (error, control efforts, and tracking	Section 3: Discrete-Time Simulation Setup

Function / Routine	Library / Toolbox Dependency	Operational Purpose in Simulation	Script Section
		profiles) to optimize execution speed and establish the setpoint matrix.	
diff, find	MATLAB Built-in (Data Analysis)	Performs vector differentiation and conditional scanning to automatically pinpoint indices where setpoint transitions occur.	Section 6: Dynamic Performance Metrics Extraction
fprintf	MATLAB Built-in (I/O)	Formats and outputs scheduled gain matrices and calculated transient metrics directly to the MATLAB Command Window.	Sections 2 & 6: Gains & Performance Metrics

Python HMI code

Module / Package	Source / Dependency Type	Operational Purpose in HMI Application	Structural Implementation
tkinter / ttk	Python Standard Library	Constructs the high-performance graphical user interface framework, managing the dark industrial theme grid system, forms, and canvas rendering.	Core GUI Layout, Widgets, and TankCanvas
matplotlib	Third-Party Package (pip install matplotlib)	Renders the real-time responsive trend graphs via a dual subplot layout (Level and PWM signals) and bridges them into the UI layout via FigureCanvasTkAgg.	RealtimeGraph Class
serial	Third-Party Package (pip install pyserial)	Handshakes with the physical USB layer, auto-detects available serial devices, and continuously reads incoming telemetry streams at 115,200 baud.	SerialReader Thread

Module / Package	Source / Dependency Type	Operational Purpose in HMI Application	Structural Implementation
socket	Python Standard Library	Spawns low-level network communication streams using non-blocking TCP/IP architectures to link bidirectionally with the target ESP32 local server IP.	WiFiReader Thread
threading	Python Standard Library	Decouples the user interface from high-frequency blocking network/serial read loops using asynchronous workers (Thread), resource lock structures (Lock), and signaling flag switches (Event).	Concurrency & Data Stream Management
numpy	Third-Party Package (pip install numpy)	Vectorizes active tracking lists into mathematical multidimensional data arrays to perform sub-millisecond maximum boundary scans for plot scaling.	Real-Time Dynamic Plot Auto-Scaling
collections.deque	Python Standard Library	Deploys memory-capped circular buffers initialized to a rigid timeline constraint (MAX_SAMPLES = 500) to guarantee deterministic memory usage over extended runtime tracking.	Thread-Safe Data Log Arrays
csv	Python Standard Library	Leverages structured dictionary parsers (DictWriter) to safely serialize execution history buffers directly into structural database-compatible logs.	export_csv Subroutine
math	Python Standard Library	Processes geometric mathematical calculations (2π cyclical ranges, sinusoids, and vector offsets) to	Dynamic Tank Wave Animation

Module / Package	Source / Dependency Type	Operational Purpose in HMI Application	Structural Implementation
		synthesize fluid dynamics waveforms on the canvas.	
time	Python Standard Library	Computes execution elapsed reference timelines, handles driver thread cycle pacing (sleep), and pulls local network calendar stamps for display strings.	Performance Metrics & Clock Synchronization

Arduino code

Library / Core Module	Source / Dependency Type	Operational Purpose in Firmware	Hardware / Structural Implementation
LiquidCrystal_I2C.h	Third-Party Library	Drives the 20×2 character LCD screen using an 8-bit I2C I/O expander to display dual-page real-time telemetry, menus, and debugging metrics.	Polled sequentially at 10 Hz (LCD_MS = 600) during autotuning and 0.25 Hz (LCD_MS = 4000) during runtime.
Keypad.h	Third-Party Library	Scans and decodes the 4×4 matrix keypad state machine with non-blocking debouncing logic to capture parameter inputs and character overrides.	Maps 4 row and 4 column GPIOs; utilizes the # key as a custom decimal delimiter and a triple-escape system reset sequence.
WiFi.h	ESP32 Arduino Core Standard	Initializes the network hardware stack in Station (STA) mode, establishes local	Connects to standard local 2.4 GHz Access Points via hardcoded credentials with a 15-

Library / Core Module	Source / Dependency Type	Operational Purpose in Firmware	Hardware / Structural Implementation
		network handshakes, and provides automatic background reconnection flags.	second hardware timeout ceiling.
WiFiServer / WiFiClient	ESP32 Arduino Core Standard	Spawns a dedicated, non-blocking TCP socket stream layer over port 8888 to process incoming text frames and broadcast telemetry streams.	Interrogated natively inside the primary loop to parse asynchronous SCPI-style configurations (e.g., SP=, MODE=).
Wire.h	Arduino Core Native	Interfaces directly with the ESP32's internal hardware I2C master peripheral controller, handling clock lines and data arbitration.	Provides the low-level hardware communication backend required by the I2C LCD wrapper on address 0x27.
HardwareSerial (Serial)	Arduino Core Native	Configures the primary Universal Asynchronous Receiver-Transmitter (UART0) peripheral for concurrent USB-to-PC monitoring.	Handles upstream data parsing and serial serialization streams synchronized at a high transmission speed of 115,200 baud.
esp32-hal-ledc (Core PWM API)	ESP32 Core Abstraction Layer	Allocates and controls the internal LEDC peripheral timer to synthesize high-resolution output frequencies for the pump driver.	Upgraded to Core 3.x specifications via ledcAttach and ledcWrite to generate a 5 kHz, 8-bit PWM profile on Pin 13.
Arduino.h Utilities (pulseIn, GPIO registers)	Arduino Core Native	Orchestrates precise time-of-flight measurements for the ultrasonic sensor and	Triggers the HC-SR04 transducer via a 10 μs pulse and tracks echo reflections using a 30

Library / Core Module	Source / Dependency Type	Operational Purpose in Firmware	Hardware / Structural Implementation
		configures digital register pin directions.	ms blocking window cap.
Standard C Library (memcpy, math.h macros)	Built-in GCC Compiler	Executes low-level memory block replication routines and bounded mathematical data constraints.	Uses memcpy to copy the level buffer for an insertion-sort algorithm, identifying the true mathematical median of 7 samples.