University of Science and Technology in Zewail City

Communication and Information Engineering Program

CIE 318: Control Systems Project SP25

Design and Implementation of a PID Controller for an RC Circuit

Mohammed Ali 202200594

Abdulrahman Magdy 202200341

Ahmed Amgad 202200393

Alaa Sabry 202200331

Supervisor: Dr. Abd-Elshafaei

May 4, 2025

Abstract

This project presents a comprehensive analysis and control design for an electric circuit modeled using state-space and transfer function approaches. The objective is to design and implement a feedback control system that achieves specific performance metrics, including a settling time of approximately 6.5 seconds and a maximum overshoot of less than 8.

The system is initially analyzed by formulating its state-space representation, deriving its transfer function, and calculating key dynamic characteristics such as the damping ratio, natural frequency, and DC gain. A gain controller is investigated using MATLAB's sisotool to explore the limits of performance enhancement and steady-state error reduction.

A PID controller is then tuned using Simulink to meet the desired transient response specifications, and the resulting PID parameters are reported along with output and control signal plots.

Furthermore, the controller is implemented in hardware using both analog components (operational amplifiers) and an Arduino microcontroller to validate the simulation results. The project concludes with a comparative analysis of the simulation and hardware outcomes, highlighting the effectiveness and limitations of each implementation method.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Description

Various engineering systems widely use electrical circuits composed of resistors, capacitors, and inductors. Controlling the behavior of such circuits is essential to ensure stability, precision, and performance in applications like signal processing, power electronics, and automation. The circuit under investigation in this project consists of two capacitors and a resistor. The objective is to analyze its dynamics and design a controller that meets specific performance criteria.

1.2 Project Objective

The primary objective of this project is to design and implement a control system that regulates the voltage across one of the capacitors in the circuit. The control design should ensure a settling time of approximately 6.5 seconds and a maximum overshoot of less than 8%

1.3 Methodology

The approach followed in this project includes modeling the system using both state-space and transfer function representations. Dynamic properties like natural frequency, damping ratio, and DC gain are derived. MATLAB tools such as `sisotool` and Simulink PID Tuner are used to analyze and tune controllers. The final controller is then implemented in both analog hardware and digitally using an Arduino microcontroller.

1.4 Report Outline

The remainder of this report is organized as follows:

- Section 2 presents theoretical modeling and controller design, including state-space analysis, transfer function derivation, time response analysis using MATLAB software, and

PID tuning in Simulink.

- Section 3 details the hardware implementation using passive components and operational amplifiers, then covers the PID controller hardware implementation using an Arduino microcontroller.
- Section 4 provides results, comparisons, and critical discussion.
- Section 5 concludes the report with a summary of findings.
- The appendix includes relevant code listings, circuit schematics, and supporting data.

Chapter 2

Theoretical Modeling and Software Simulation

This chapter presents the mathematical derivation of the system models (state-space and transfer function) for the RC circuit under study, followed by analysis and controller design using software simulation tools. We assume the circuit diagram is presented elsewhere, for example in Figure **??**.

2.1 State Space Model

The first step is to derive the state-space representation for the circuit. The given component values are $R_1 = R_2 = 100\,\mathrm{k\Omega}$ and $C_1 = C_2 = 10\,\mathrm{\mu F}$.

Following the project requirements, we select the state variables as the voltages across the capacitors:

- $x_1(t) = v_{C1}(t)$: Voltage across capacitor C_1.
- $x_2(t) = v_{C2}(t)$: Voltage across capacitor C_2.

The input to the system is the source voltage $u(t) = V_{in}(t)$. The output is defined as the voltage across the second capacitor, $y(t) = v_{C2}(t)$, which is equal to the state variable $x_2(t)$.

We apply Kirchhoff's Current Law (KCL) at the essential nodes.

- **KCL at Node 1 (Voltage x_1):** The current from the source through R_1 equals the sum of the current through C_1 and the current through R_2.

$$\frac{u(t) - x_1(t)}{R_1} = i_{C1}(t) + i_{R2}(t) = C_1 \frac{dx_1(t)}{dt} + \frac{x_1(t) - x_2(t)}{R_2}$$

  Rearranging to isolate $\dot{x}_1(t)$:

$$\begin{aligned} C_1 \dot{x}_1(t) &= \frac{u(t)}{R_1} - \frac{x_1(t)}{R_1} - \frac{x_1(t)}{R_2} + \frac{x_2(t)}{R_2} \\ \dot{x}_1(t) &= -\left(\frac{1}{R_1 C_1} + \frac{1}{R_2 C_1}\right) x_1(t) + \frac{1}{R_2 C_1} x_2(t) + \frac{1}{R_1 C_1} u(t) \end{aligned}$$

- **KCL at Node 2 (Voltage x_2):** The current through R_2 equals the current through C_2.

$$i_{R2}(t) = i_{C2}(t) \implies \frac{x_1(t) - x_2(t)}{R_2} = C_2 \frac{dx_2(t)}{dt}$$

Rearranging to isolate $\dot{x}_2(t)$:

$$\dot{x}_2(t) = \frac{1}{R_2 C_2} x_1(t) - \frac{1}{R_2 C_2} x_2(t)$$

The system can now be written in the standard state-space form $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t)$ and $y(t) = C\mathbf{x}(t) + Du(t)$, where the state vector is $\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$. The system matrices A, B, C, and D are:

$$A = \begin{bmatrix} -\left(\frac{1}{R_1C_1} + \frac{1}{R_2C_1}\right) & \frac{1}{R_2C_1} \\ \frac{1}{R_2C_2} & -\frac{1}{R_2C_2} \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{R_1C_1} \\ 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D = [0]$$

Now, we substitute the numerical values of the components. First, calculate the time constants:

- $R_1C_1 = (100 \times 10^3\,\Omega) \times (10 \times 10^{-6}\,\text{F}) = 1\,\text{s}$
- $R_2C_1 = (100 \times 10^3\,\Omega) \times (10 \times 10^{-6}\,\text{F}) = 1\,\text{s}$
- $R_2C_2 = (100 \times 10^3\,\Omega) \times (10 \times 10^{-6}\,\text{F}) = 1\,\text{s}$

Substituting these into the matrices yields:

$$A = \begin{bmatrix} -(1+1) & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.1}$$
$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2.2}$$
$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{2.3}$$
$$D = [0] \tag{2.4}$$

Therefore, the final state-space model for the given RC circuit is:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \tag{2.5}$$
$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + [0]u(t) \tag{2.6}$$

Section Summary: This section derived the state-space representation for the RC circuit using capacitor voltages as state variables. The resulting system matrices, based on the given component values, are defined in Equations 2.1 through 2.4.

2.2 Transfer Function of the System

The transfer function $G(s) = Y(s)/U(s)$ relates the Laplace transform of the output $y(t)$ to the Laplace transform of the input $u(t)$, assuming zero initial conditions. It can be derived from the state-space model (A, B, C, D matrices found in Equations 2.1-2.4) using the formula:

$$G(s) = C(sI - A)^{-1}B + D \tag{2.7}$$

where I is the identity matrix.

First, we compute the matrix $(sI - A)$:

$$sI - A = s\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}-2 & 1\\ 1 & -1\end{bmatrix} = \begin{bmatrix}s+2 & -1\\ -1 & s+1\end{bmatrix}$$

Next, we find the inverse of $(sI - A)$. The determinant is:

$$\det(sI - A) = (s+2)(s+1) - (-1)(-1) = s^2 + 3s + 2 - 1 = s^2 + 3s + 1$$

The inverse is then:

$$(sI - A)^{-1} = \frac{1}{\det(sI - A)}\begin{bmatrix}s+1 & 1\\ 1 & s+2\end{bmatrix} = \frac{1}{s^2 + 3s + 1}\begin{bmatrix}s+1 & 1\\ 1 & s+2\end{bmatrix}$$

Now, substitute into the formula (2.7), noting that $D = [0]$:

$$\begin{aligned}
G(s) &= C(sI - A)^{-1}B \\
&= \begin{bmatrix} 0 & 1 \end{bmatrix} \left(\frac{1}{s^2+3s+1} \begin{bmatrix} s+1 & 1 \\ 1 & s+2 \end{bmatrix} \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= \frac{1}{s^2+3s+1} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s+1 & 1 \\ 1 & s+2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= \frac{1}{s^2+3s+1} \left(\begin{bmatrix} 0\cdot(s+1)+1\cdot 1 & 0\cdot 1 + 1\cdot(s+2) \end{bmatrix} \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= \frac{1}{s^2+3s+1} \begin{bmatrix} 1 & s+2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= \frac{1}{s^2+3s+1} (1\cdot 1 + (s+2)\cdot 0) \\
G(s) &= \frac{1}{s^2+3s+1}
\end{aligned}$$

This is the transfer function of the system from input voltage V_{in} to output voltage V_{C2}.

2.2.1 System Parameters: Natural Frequency, Damping Ratio, DC Gain

We compare the derived transfer function (??) to the standard form of a second-order system:

$$G_{std}(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.8}$$

where K is the DC gain (or steady-state gain), ω_n is the natural frequency, and ζ is the damping ratio.

DC Gain (K): The DC gain is the transfer function value at $s = 0$.

$$\boxed{K = G(0) = \frac{1}{0^2 + 3(0) + 1} = 1} \tag{2.9}$$

Natural Frequency (ω_n): Comparing denominators, $\omega_n^2 = 1$.

$$\boxed{\omega_n = \sqrt{1} = 1\,\text{rad}\cdot\text{s}^{-1}}$$

Damping Ratio (ζ): Comparing denominators, $2\zeta\omega_n = 3$. With $\omega_n = 1$:

$$\boxed{\zeta = \frac{3}{2} = 1.5}$$

Since $\zeta > 1$, the open-loop system is **overdamped**.

Section Summary: This section derived the system's transfer function, resulting in $G(s) = \frac{1}{s^2+3s+1}$. Key open-loop system parameters were calculated:

- **DC Gain:** $K = 1$
- **Natural Frequency:** $\omega_n = 1\,\mathrm{rad}\cdot\mathrm{s}^{-1}$
- **Damping Ratio:** $\zeta = 1.5$ **(indicating an overdamped system).**

2.3 Time Response Analysis in MATLAB

From the MATLAB `sisotool` analysis (Figure 2.1):

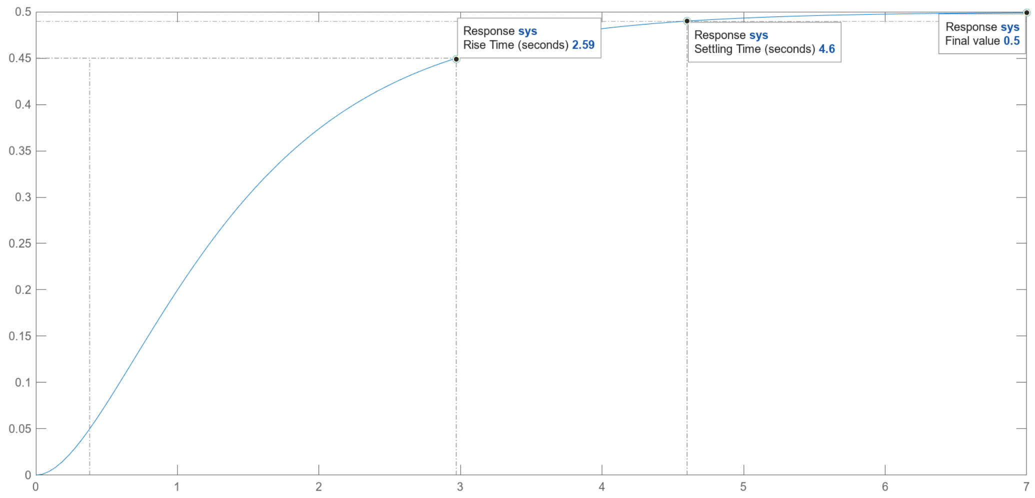


Figure 2.1: System response with proportional gain controller

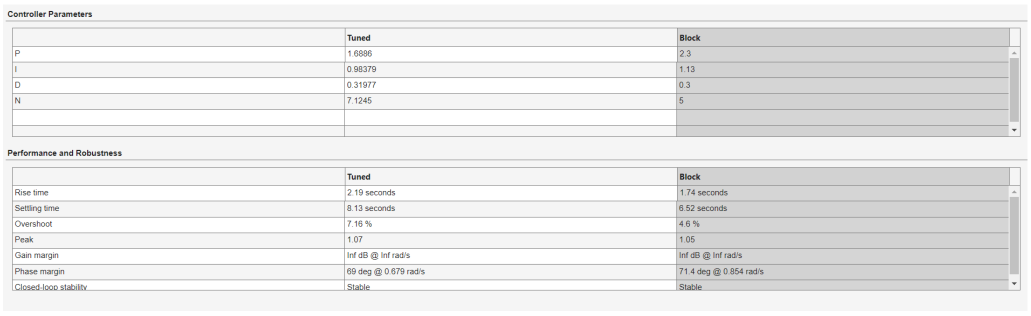
2.3.1 Performance Metrics

- **Shortest achievable settling time:** 4.6 seconds
- **Steady-state error:** $e_{ss} = 1 - 0.5 = 0.5$ (for unit step input)
- **Rise time:** 2.59 seconds

2.3.2 Key Observations

- Zero steady-state error is **not achievable** with proportional control
- Reason: The steady-state error formula $e_{ss} = \frac{1}{1+K}$ requires infinite gain to reach zero error
- System remains Type 0 (no integral action to eliminate steady-state error)

2.4 PID Controller Tuning Using Simulink



Controller Parameters

	Tuned	Block
P	1.6886	2.3
I	0.98379	1.13
D	0.31977	0.3
N	7.1245	5

Performance and Robustness

	Tuned	Block
Rise time	2.19 seconds	1.74 seconds
Settling time	8.13 seconds	6.52 seconds
Overshoot	7.16 %	4.6 %
Peak	1.07	1.05
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	69 deg @ 0.679 rad/s	71.4 deg @ 0.854 rad/s
Closed-loop stability	Stable	Stable

Figure 2.2: Manually tuned PID controller parameters

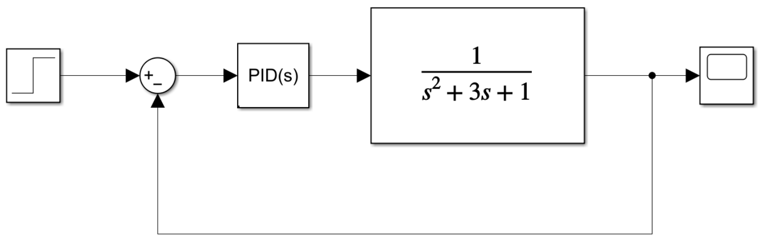


Figure 2.3: Closed-loop control scheme with $G(s) = \frac{1}{s^2+3s+1}$ plant

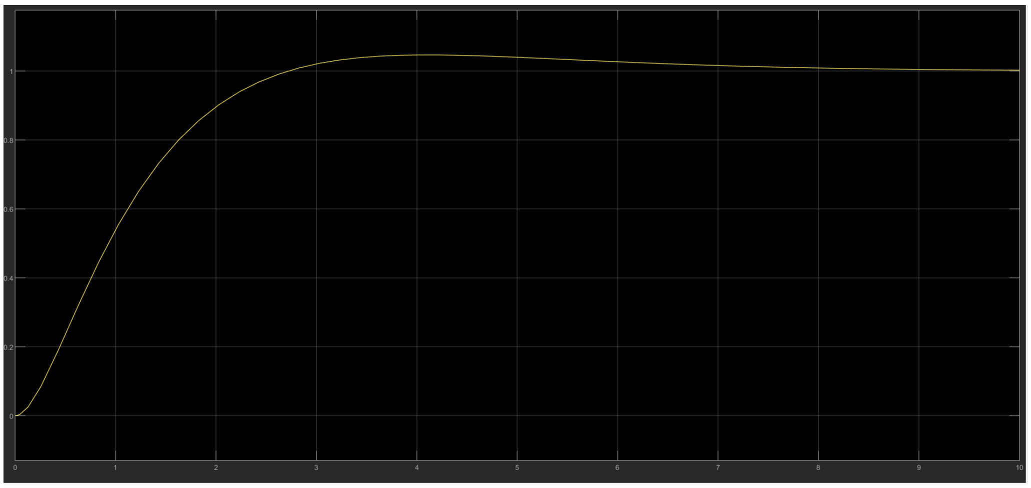


Figure 2.4: System step response with manual PID tuning

2.4.1 Controller Design

The PID controller was manually tuned with the following implementation:

- Time-domain equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

- Laplace-domain transfer function:

$$U(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) E(s)$$

- Final parameters:
  - Proportional gain $(K_p) = 2.3$
  - Integral gain $(K_i) = 0.98$
  - Derivative gain $(K_d) = 0.3$
  - Filter coefficient (N) = 5

2.4.2 System Performance

The manually tuned controller achieved:

- Settling time: 6.52 seconds
- Overshoot: 4.6%
- Phase margin: 71.4° @ 0.854 rad/s
- Stable closed-loop operation (Fig. 2.3)

Chapter 3

Hardware Implementation

3.1 Controller Implementation using Op Amps

3.1.1 Circuit Design Using LTSpice Software

To simulate the controller using analog components, we designed a circuit based on operational amplifiers (Op Amps) using LTSpice. The goal was to replicate the behavior of the PID controller using only analog components.

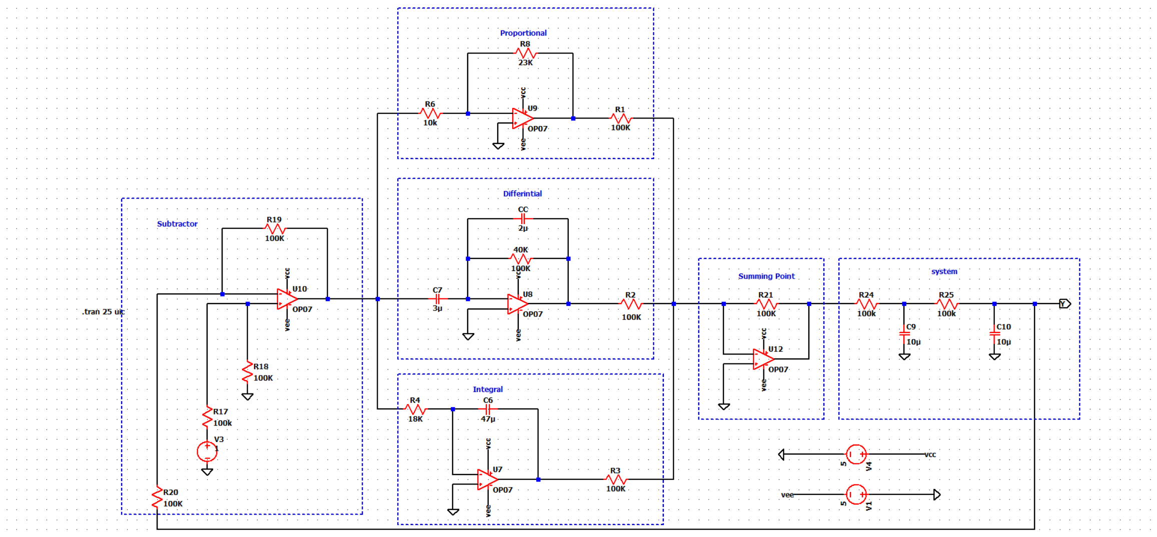
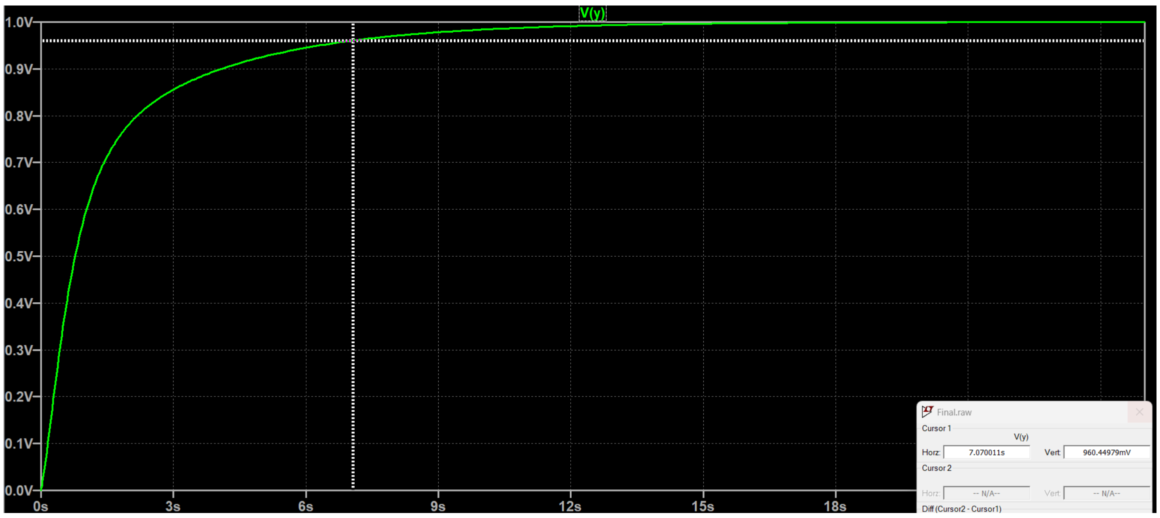


Figure 3.1: LTSpice simulation scheme of the analog controller.

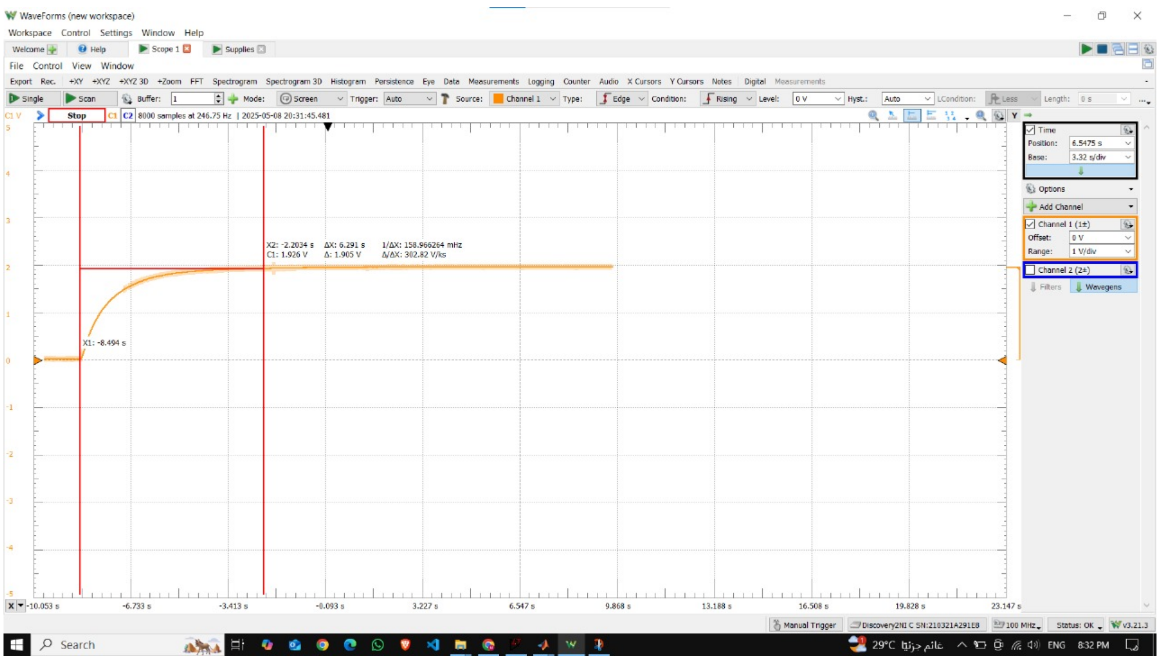


Cursor 1 V(y)
Horz: 7.070011s Vert: 960.44979mV
Cursor 2
Horz: --N/A-- Vert: --N/A--
Diff (Cursor2 - Cursor1)

Figure 3.2: Simulated response of the analog controller circuit in LTSpice.

3.1.2 Hardware Implementation and Response of the Circuit

When building the circuit in real life, we found that many of the exact resistor and capacitor values used in LTSpice were not available. To solve this, we used approximate values by combining available components in series or parallel to get as close as possible to the needed values.



C2 8000 samples at 246.75 Hz | 2025-05-08 20:31:45.481
X1: -8.494 s
X2: -2.2034 s ΔX: 6.291 s 1/ΔX: 158.966264 mHz
C1: 1.926 V Δ: 1.905 V Δ/ΔX: 302.82 V/ks
Time Position: 6.5475 s Base: 3.32 s/div
Channel 1 (1±) Offset: 0 V Range: 1 V/div
Channel 2 (2±)
Mode: Screen Trigger: Auto Source: Channel 1 Type: Edge Condition: Rising Level: 0 V Hyst.: Auto Length: 0 s
Manual Trigger Discovery2NI C SN:210321A291EB 100 MHz Status: OK v3.21.3

Figure 3.3: Measured response of the hardware circuit.

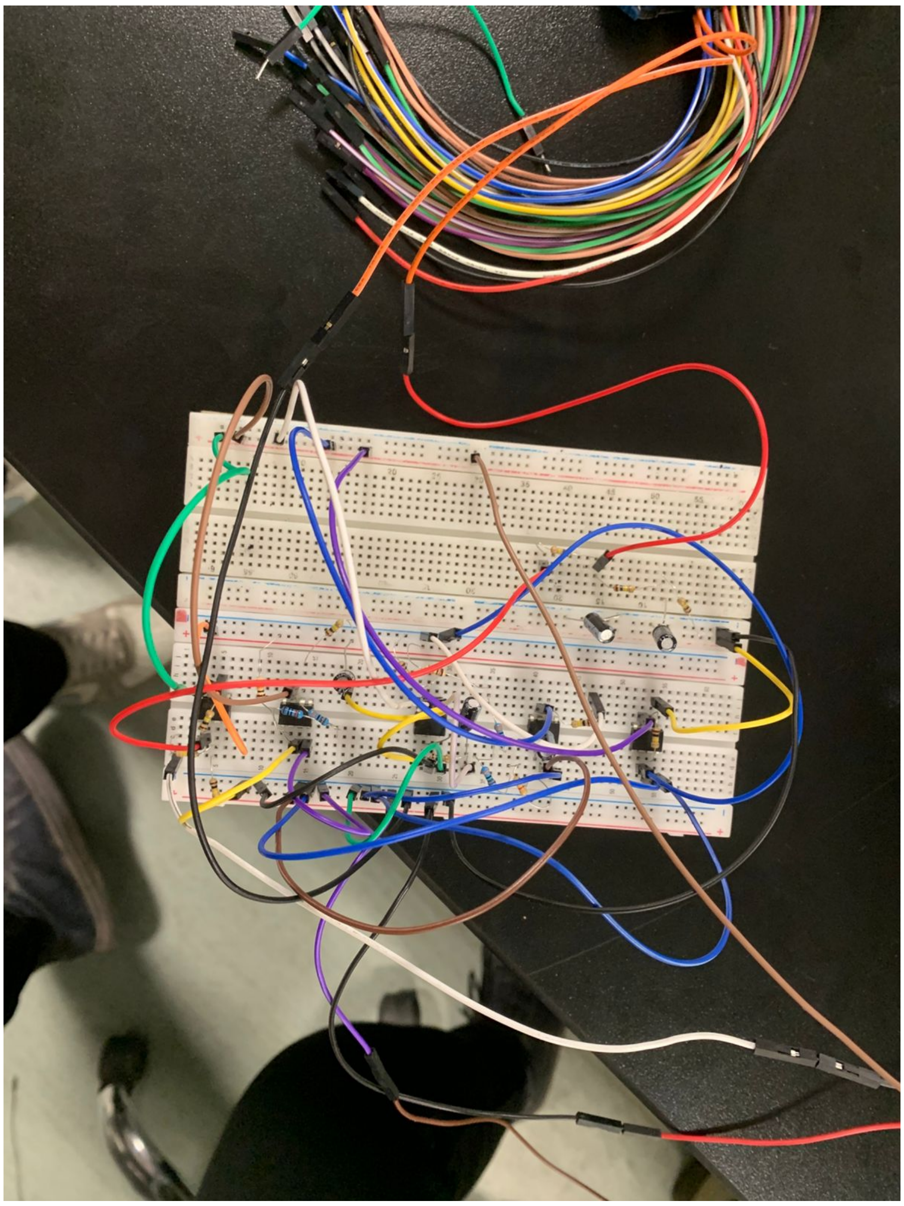


Figure 3.4: Actual hardware circuit implementation using Op Amps.

3.1.3 Comment

We noticed that the analog implementation using Op Amps did not fully match the expected behavior. The differences are likely due to the use of approximate component values, tolerance variations in resistors and capacitors, and possible noise or errors in the measurement tools, power supply, or temperature conditions.

3.2 Controller Implementation using Arduino Microcontroller

The controller was also implemented using an Arduino Uno board. Instead of writing C++ code, we used a Simulink model to control the Arduino. This made it easier to test and update the controller by directly sending the control logic from Simulink to the hardware.

3.2.1 Simulink Model

Figure 3.5 shows the actual Arduino-based implementation, and Figure 3.6 shows the Simulink model used to program the Arduino.

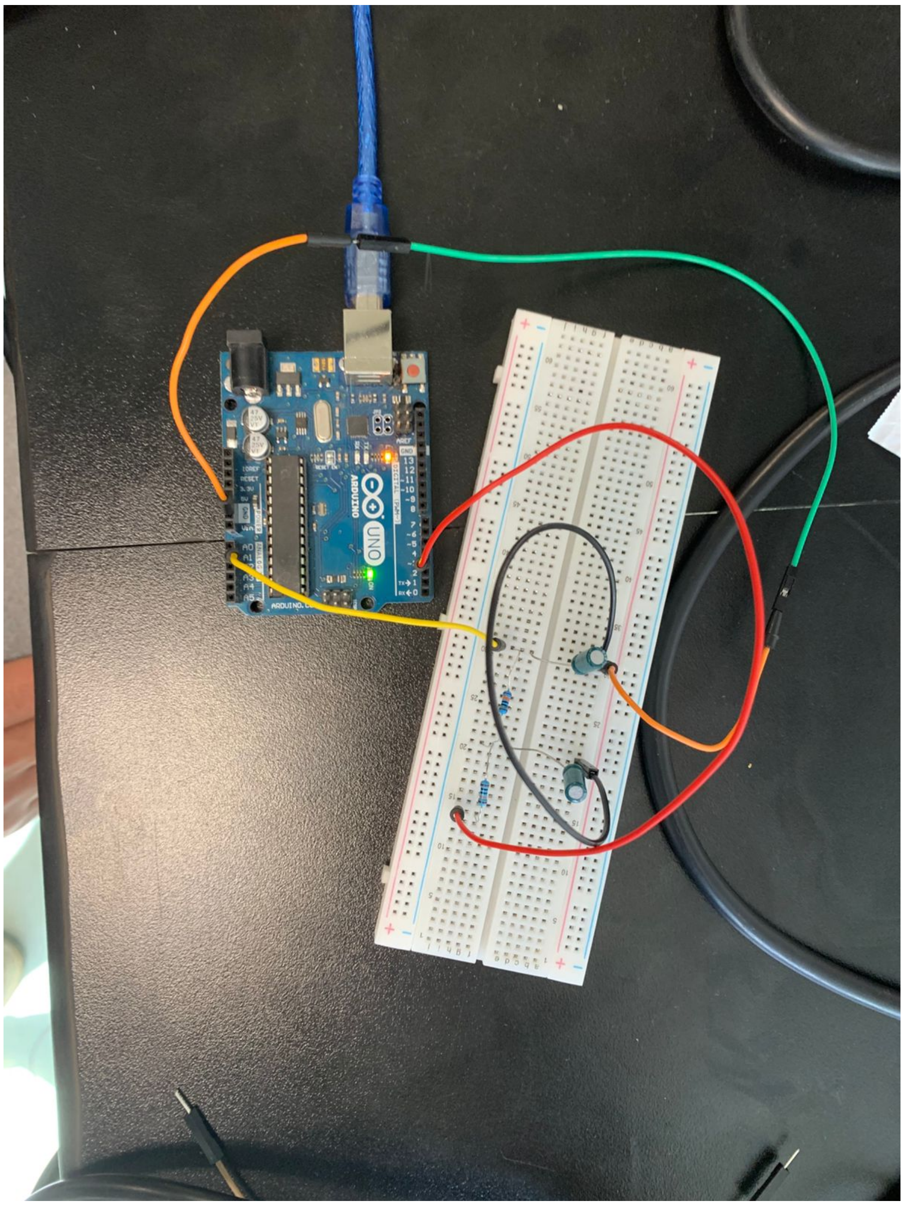


Figure 3.5: Actual Arduino-based control setup.

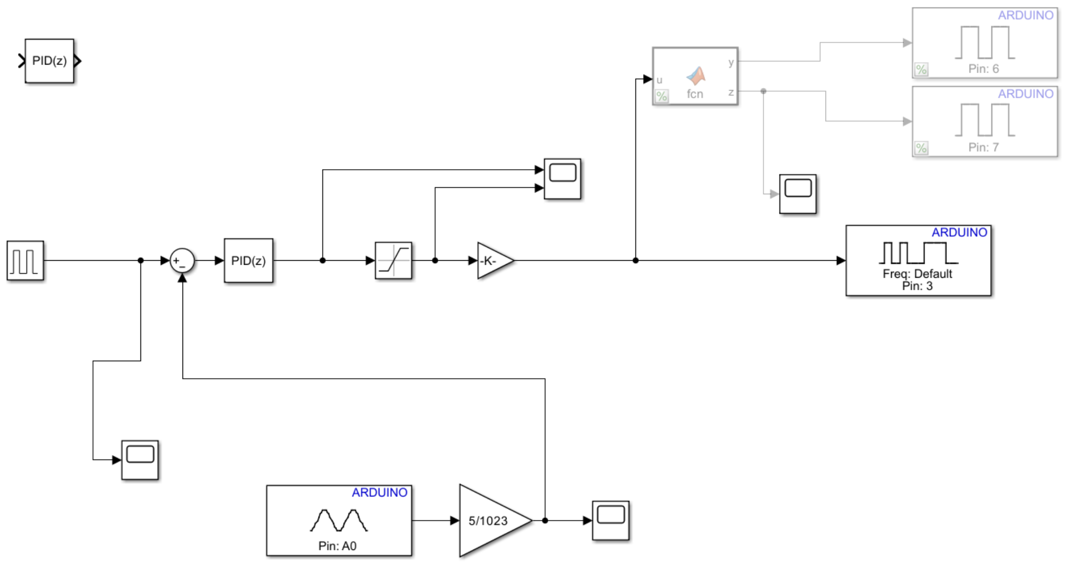
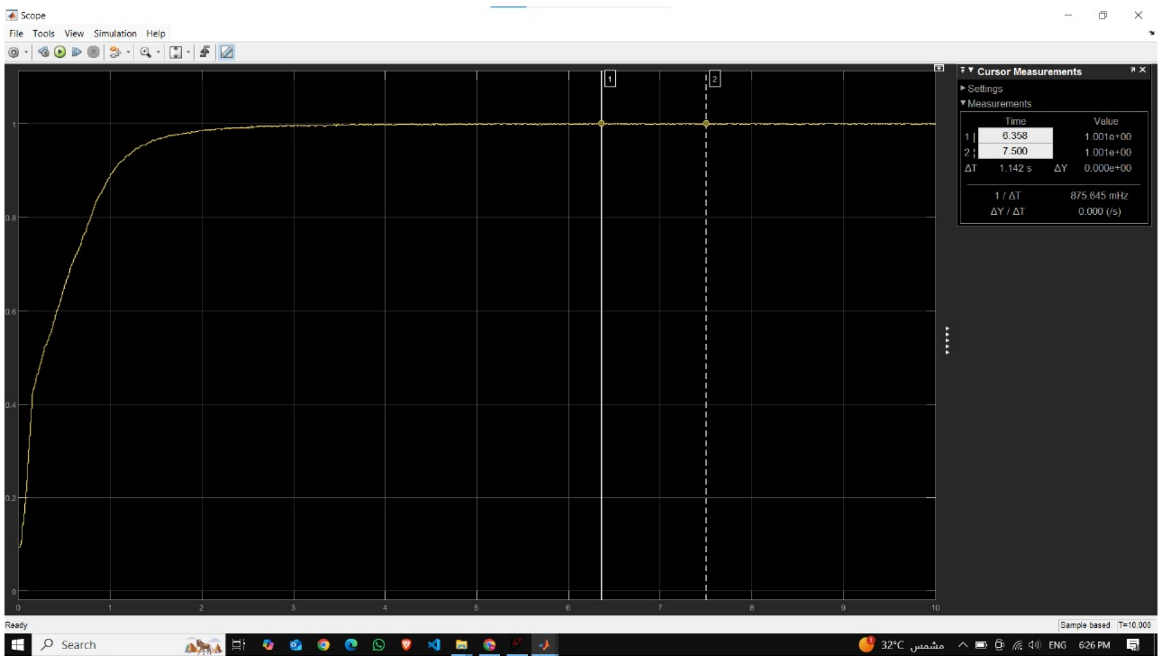


Figure 3.6: Simulink model used to control the Arduino.

3.2.2 System Response

Figure 3.7 shows the system response when controlled using the Arduino and Simulink. It is compared with the LTSpice analog controller and the Simulink-only PID controller.



Cursor Measurements
Settings
Measurements

	Time		Value
1	6.358		1.001e+00
2	7.500		1.001e+00
ΔT	1.142 s	ΔY	0.000e+00
	1 / ΔT		875.645 mHz
	ΔY / ΔT		0.000 (/s)

Figure 3.7: System response using Arduino compared to LTSpice and Simulink-only PID.

3.2.3 Comparison and Discussion

The response from the Arduino is very close to that of the analog circuit simulated in LTSpice. This confirms that the controller works well on real hardware.

However, it does not fully match the response of the Simulink-only PID controller. This difference may be due to:

- Small delays in the Arduino processing.
- Limited precision in digital signals.
- Noise or disturbances in the real system.

This shows that while simulations are helpful, testing on real hardware provides a more realistic and practical result — and the differences are still within acceptable limits.

Chapter 4

Results, Comparisons, and Discussion

4.1 LT-Spice VS PID Simulink Responses

A good point to start with would be Simulink's utilization of the time domain through numerical integration; this is then used to solve the system's differential equations. At the same time, LTSpice, typically working in the frequency domain, applies methods like modified nodal analysis. This major change in the method could result in the separation of the results.

Also, the models of the components being used in the two programs may be the reason for the differences. The models of the components used in LTSpice are generally more detailed because they include the non-ideal component's behavior, while in Simulink, the models are idealized and simplified. Differences in the solvers' settings and the methods of numerical computation in the two tools can also lead to variations.

Lastly, LTSpice can sometimes capture parasitic effects, for example, small capacitances and inductances that are not included in the Simulink model, which can further differentiate the results. In addition, ensuring that the initial conditions are kept constant between the two simulations is very important to make a valid comparison.

From these inherent deviations, it can be seen that understanding the advantages and the limitations of each tool is essential especially when the comparison of the outcomes in their simulations. Although there might be contradistinctions, both Simulink and LTSpice will be proficient for exploring the system extensively.

4.2 LT-Spice VS Arduino Response

In Simulink, e.g. component tolerances, noise, quantization due to the analog-to-digital converter (ADC), and also unmodeled parasitic capacitive and inductive effects. The hardware results typically coincide with LTSpice more closely mainly because of these real-world aspects. LTSpice, even when using idealized components, is capable of some real-life behavior and parasitic effects that are inherent in its simulation engine. Most of these effects are of the same nature as the physical Arduino implementation. Contrarily to this, Simulink contains mostly ideal components in its models and therefore, it does not directly simulate real-world restrictions.

Consequently, LTSpice is a better tool for the elements of reality and so it renders the most correct representation of the system, thus, it gives better matching with the physical Arduino system, especially for analog circuits. Despite the vary good initial designs and easy understanding of the system's dynamics, LTSpice is beyond doubt more reliable in foreseeing real-world hardware performance. The almost perfect correspondence between LTSpice anduino demonstrates the fact that one should always think about the limilimitations in real lifeduring the design and realization of the control ssystem.

Chapter 5

Conclusions

This project successfully designed and implemented a PID controller for an RC circuit, comparing simulations in Simulink and LTSpice with a real-world Arduino implementation. We found that LTSpice simulations more accurately predicted hardware performance than Simulink, primarily due to LTSpice's inherent inclusion of non-ideal component behavior and parasitic effects, even in "ideal" simulations. The Arduino implementation, subject to real-world limitations like component tolerances, noise, and quantization, validated the LTSpice results. While Simulink provided a valuable platform for initial design and understanding system dynamics, LTSpice proved essential for realistic performance prediction, especially with analog circuits. This underscores the importance of considering real-world imperfections when designing and implementing control systems, and highlights the strengths of LTSpice for hardware-focused control applications. The project demonstrated the effectiveness of PID control for the target RC circuit and emphasized the need for comprehensive simulation and hardware validation for robust control system design.

Chapter 6

References

References

[1] The MathWorks, Inc., *Control System Toolbox User's Guide*, Natick, MA, USA, 2024. [Software]. Available: https://www.mathworks.com/help/control/ref/sisotool.html

[2] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed., Research Triangle Park, NC: ISA, 1995.

[3] Texas Instruments, *LM741 Operational Amplifier Datasheet*, Rev. J, Dallas, TX, USA, Sept. 2015. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm741.pdf

Appendix A

Appendix

A.1 MATLAB codes used

```
G = tf(1, [1 3 1]);
sisotool(G)
```