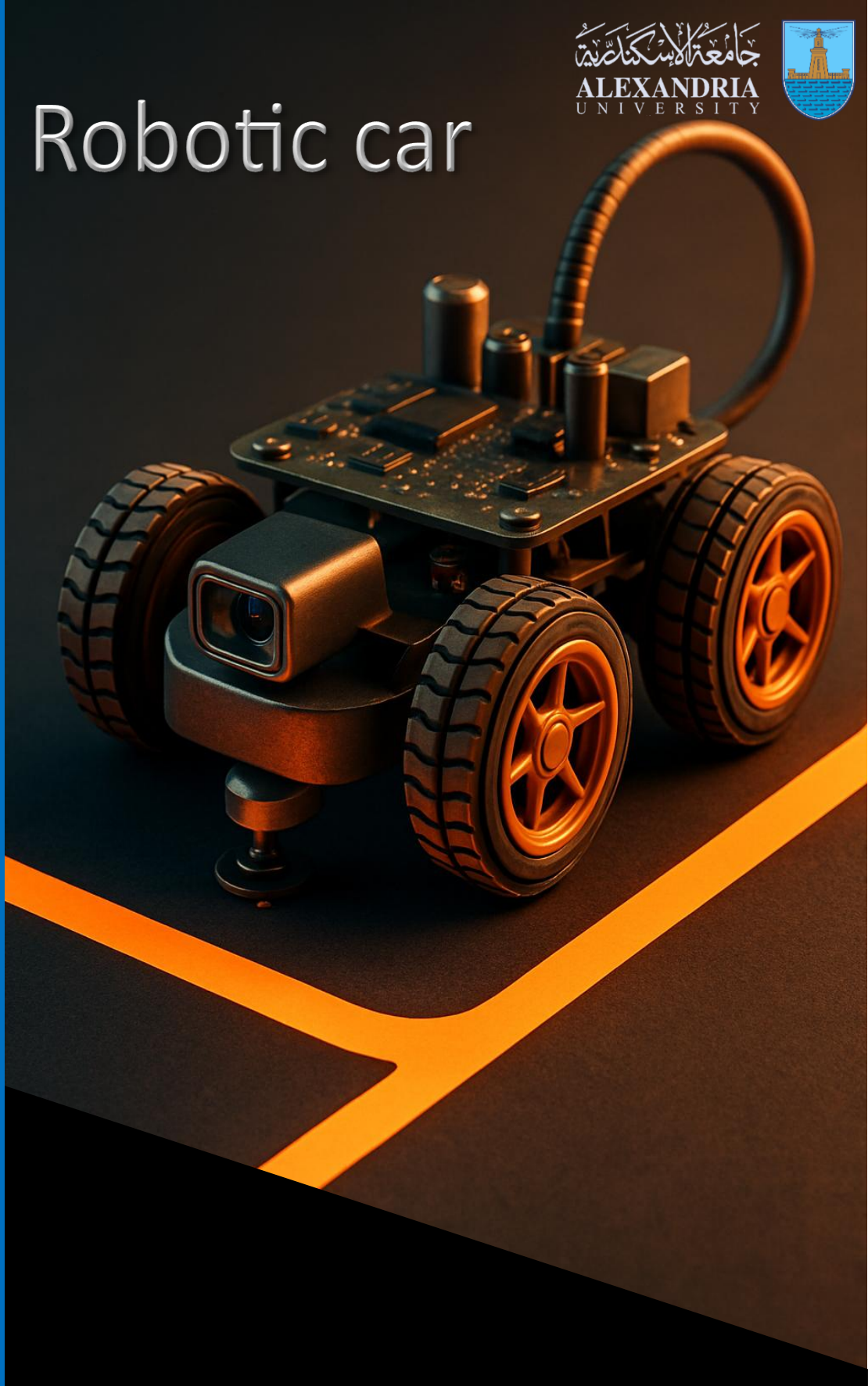
Scan me!

Automatic control

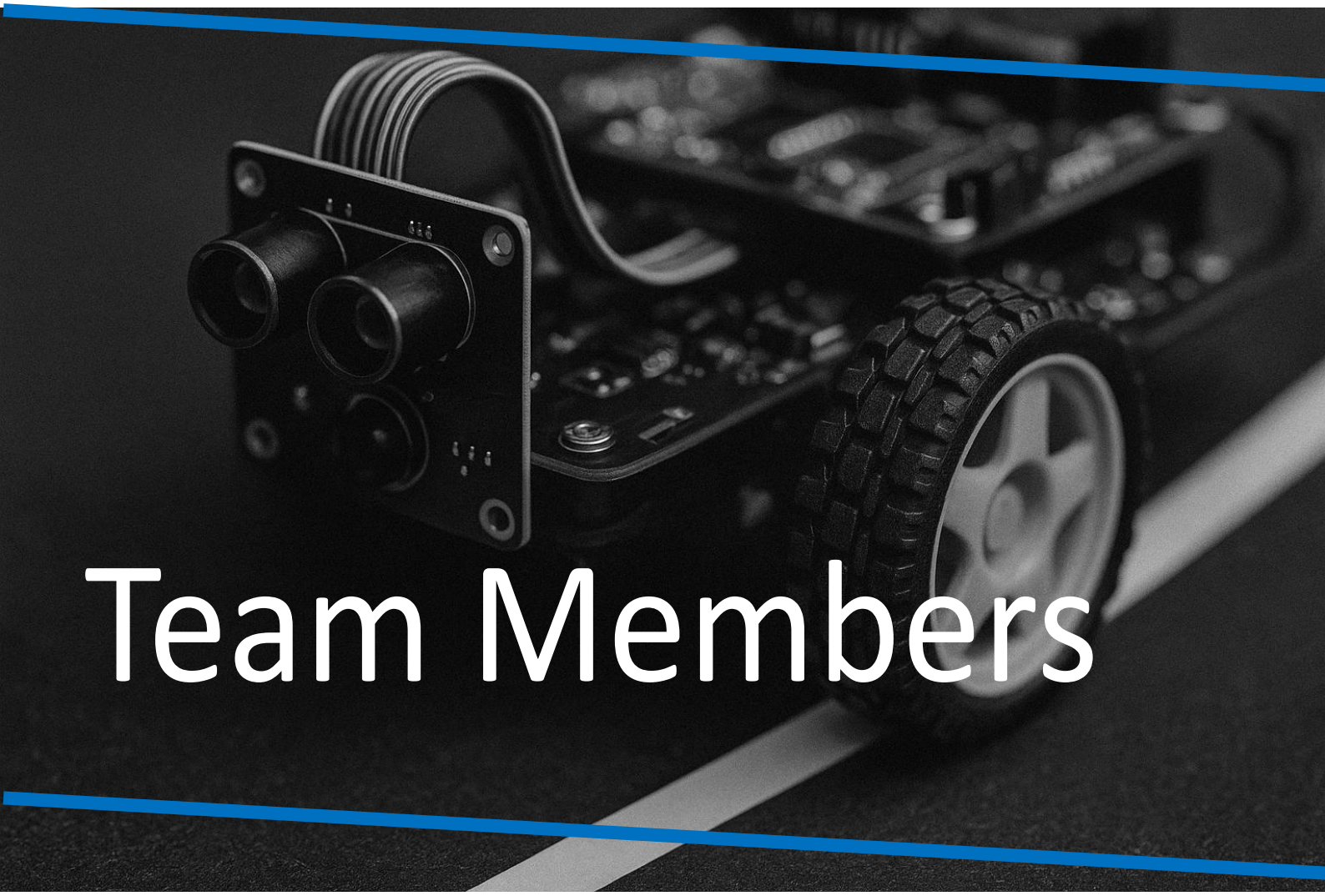
Robotic car



Presented to

Prof. Dr. Ahmed Saleh

Mechanical Department | Design



Team Members

Ahmed Samy Elsaiid Mohamed Ahmed Elshimy	21010085
Ahmed Saiid Abdelnaby Abdelrahman	21010090
Ahmed Mohamed Ahmed Saleh	21010144
Ahmed Mohamed Refaat Bassiouny	21010158
Ayman Muhammed Hussien El-sayed Muhammed	21010334
Ziad Ashraf Mohamed Ali	21010561
Mohamed Ashraf Ali Mohamed	21011086
Mohamed Alaa Hassan Mabrouk	21011192
Mahmoud Mohamed Abdallah Elnahas	21011285

Contents

Abstract

This comprehensive study presents the design, development, and implementation of an autonomous path-tracking robotic vehicle, integrating principles from mechanical engineering, electronics, and control systems. The project addresses the fundamental challenge of creating a reliable and efficient line-following robot capable of navigating predefined paths with precision, while maintaining robustness against real-world disturbances.

The mechanical subsystem features a meticulously designed two-tiered chassis fabricated from laser-cut acrylic, optimized through an iterative design process that balanced structural integrity, component integration, and spatial constraints. The fabrication process leveraged computer-aided design and laser cutting technologies to achieve precision and repeatability in manufacturing.

The electronic architecture centers on an Arduino Uno microcontroller interfaced with a three-sensor infrared array and L298N motor drivers. The sensor array provides real-time feedback for path correction, while the motor drivers enable precise control of the propulsion system.
A rigorous mathematical framework was developed to model the system's electromechanical dynamics, incorporating motor torque characteristics, inertial effects, and disturbance factors. These simulations provided critical insights into the system's behavior and informed the control strategy development.

The control system implementation compared open-loop and closed-loop architectures, with particular focus on their respective capabilities in trajectory tracking and disturbance rejection. Simulation results demonstrated significant performance enhancements through closed-loop control, with measured improvements in steady-state error reduction and substantial gains in transient response characteristics.

This project serves as a valuable case study in mechatronic system integration, highlighting the interdisciplinary nature of robotics development. The successful integration of mechanical design, electronic systems, and control theory principles offers a replicable framework for similar robotic applications.

Introduction

Autonomous mobile robots are increasingly deployed in industrial automation, logistics, and educational environments due to their ability to perform repetitive tasks with high precision. Among these, line-following robots represent a fundamental yet critical application, requiring seamless integration of mechanical design, sensor systems, and control algorithms.

This project focuses on the development of a robotic car capable of autonomously tracking a predefined path using infrared (IR) sensors. The primary objectives include:

1. **Mechanical Design**: Creating a lightweight, structurally sound chassis that accommodates propulsion systems, sensors, and control electronics while adhering to spatial constraints.
2. **Sensor and Actuation Systems**: Selecting and configuring sensors for reliable path detection and motors for precise motion control.
3. **System Modeling**: Deriving a mathematical representation of the robot's dynamics to facilitate simulation and control design.
4. **Control Strategy**: Implementing and validating a feedback-based control system to enhance trajectory tracking and disturbance rejection.

The project adopts a systematic methodology, beginning with conceptual design and progressing through prototyping, mathematical modeling, and experimental validation. The outcomes contribute to a deeper understanding of the interplay between mechanical design, embedded systems, and control theory in autonomous robotics.

Methodology

1. Mechanical Design & Fabrication

- Developed a two-tiered acrylic chassis via iterative prototyping, transitioning from mecanum to standard wheels for compactness
- Fabricated components using laser-cutting for precision, with dimensions optimized for motor/sensor integration

2. Electronic System Implementation

- Employed **Arduino Uno** with **L298N motor drivers** to control **12V DC motors**
- Optimized sensor configuration from 4 → 3 IR sensors for balanced line-detection accuracy and computational efficiency

3. Mathematical Modeling & Simulation

- Derived electromechanical equations (motor torque, back-EMF, load dynamics)
- Simulated in **Simulink** under two conditions:
    - *Ideal case*: No external disturbances
    - *Realistic case*: Inertial/frictional loads included

Mechanical Design

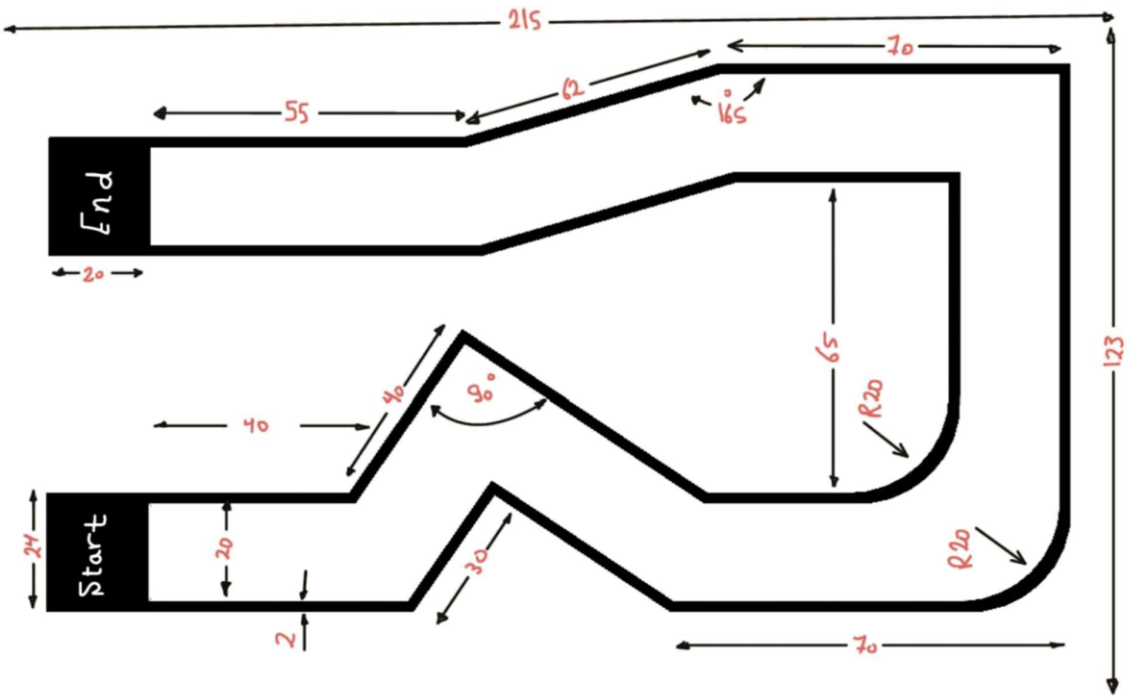
In this project, we used a structured design process to guide our work. This method helped to move step by step from understanding the problem to developing and refining the solution. By following this approach:

Definition of the Problem

The initial stage of the project involves understanding the core problem and identifying the key requirements for the system. The objective is to design a mobile platform capable of following a predefined path. To achieve this functionality, the system must integrate several essential components, including motors for movement, a control system to manage operations, infrared sensors (IR) for line detection, wheels for mobility, and a chassis to support all hardware elements.

A clear understanding of each component's role is essential for designing a functional and reliable path-following mobile. This phase lays the foundation for the design and ensures that all necessary elements are considered from the outset.

The path



Brain storming

During the initial stages of the design process, the concept was developed around the use of mecanum wheels driven by two 12V DC motors. This configuration was chosen for its ability to provide omnidirectional movement, which offered high maneuverability and flexibility. However, as the design progressed, it became apparent that the overall dimensions of the platform exceeded the allowable spatial constraints, making this configuration unfeasible.

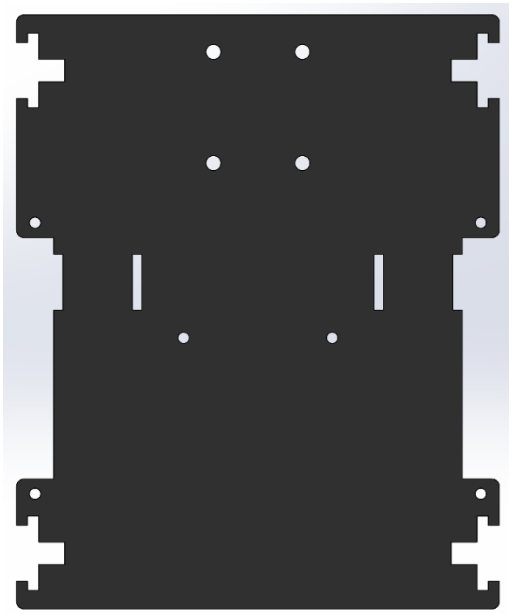
To address this issue, a redesign was undertaken with the aim of reducing the platform's dimensions. The mecanum wheels were replaced with standard wheels in an effort to minimize the overall footprint while retaining a dual 12V motor drive system. Unfortunately, this approach also failed to meet the dimensional requirements, as the physical spacing required to accommodate the two 12V motors still exceeded the prescribed limits.

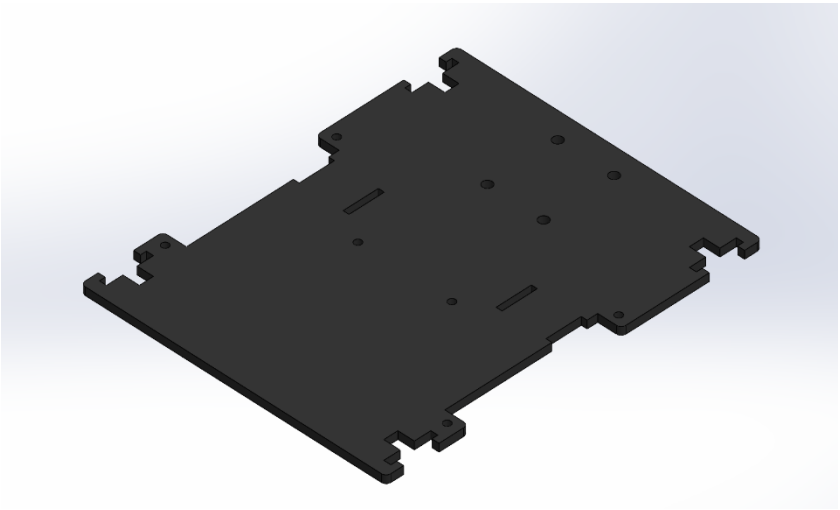
After further evaluation and iterative design modifications, a viable solution was identified. The final design employed regular wheels in combination with two 6V DC motors, which offered a more compact form factor while still meeting the basic mobility requirements of the system. Due to the integration of various functional components and the need to maintain structural and operational efficiency within the constrained space, the design evolved into a two-stage mobile platform. This modular approach allowed for better spatial distribution of components, improved balance, and facilitated maintenance and scalability of the system.

Prototype design

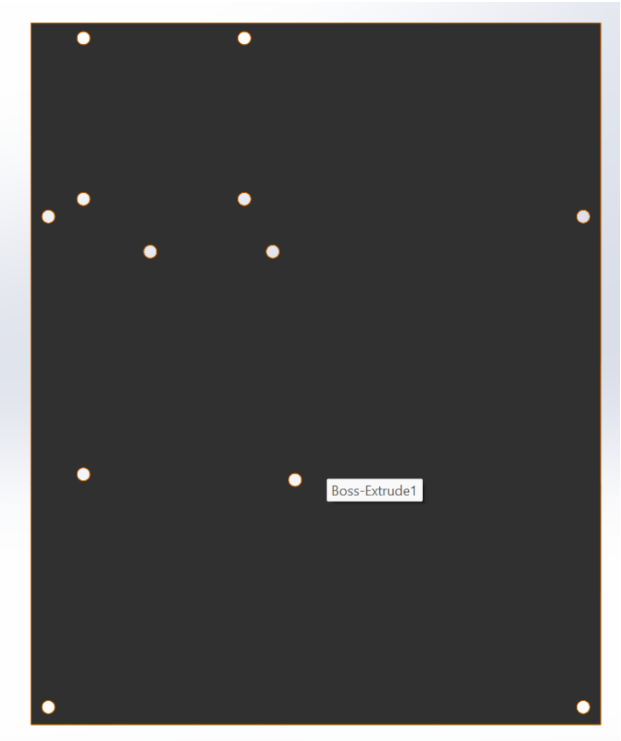
a) Bottom Platform :

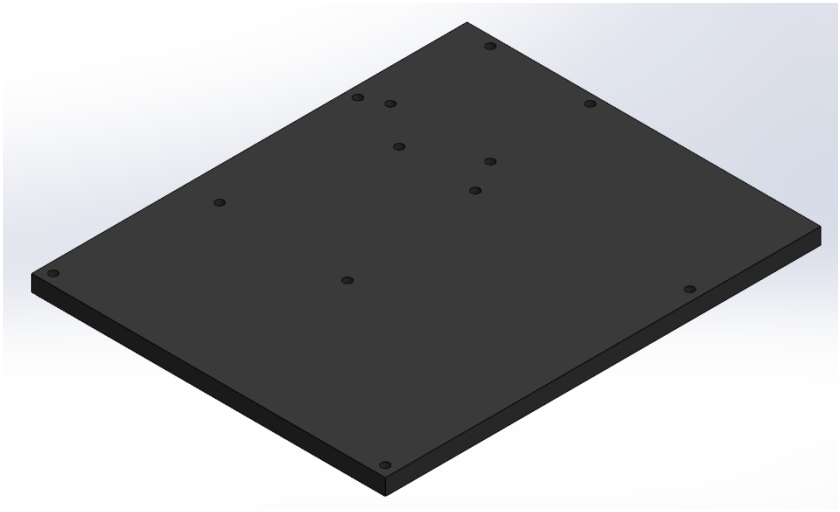
- Using **SOLIDWORKS** to draw the requried parts 3D to visualize the actual mates and dimensions for every component





b) Top Platform





- While designing the mobile platform, the layout of essential parts—including sensors, wheels, and motors—was carefully planned to maintain both efficiency and compactness. Each component was positioned to fit within the limited available space without affecting the platform's overall function.

- Sensors were mounted in open areas to allow clear detection of the surroundings, ensuring accurate data collection. The wheels and motors were strategically placed to maintain balance and allow smooth movement, while also fitting within the physical limits of the chassis.

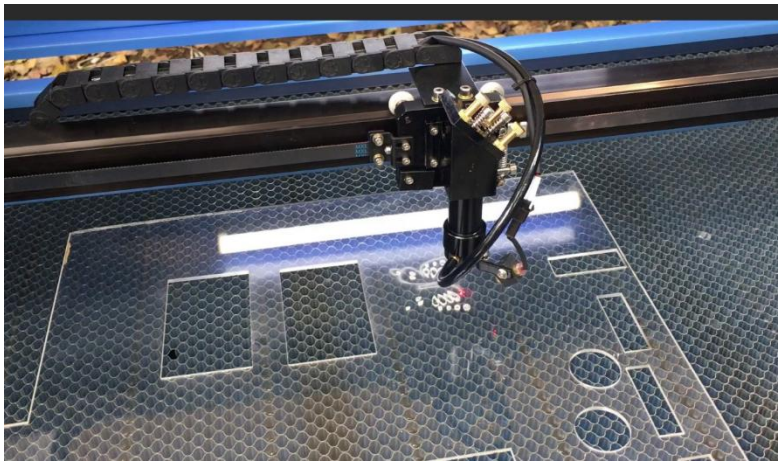
- This organized setup not only improved the system's functionality but also made it easier to assemble and modify in future stages. The thoughtful placement of each part played a key role in achieving a compact and reliable design.

Fabrication

The chassis of the mobile platform was fabricated using acrylic, a lightweight yet durable thermoplastic known for its excellent strength-to-weight ratio, transparency, and ease of machining. Acrylic was selected as the primary material due to its favorable mechanical properties and its compatibility with precision fabrication techniques. Its rigidity and relatively low cost make it an ideal choice for prototyping and small-scale robotic applications where structural stability and weight minimization are critical factors.

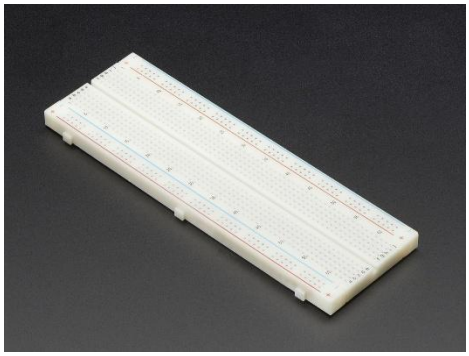
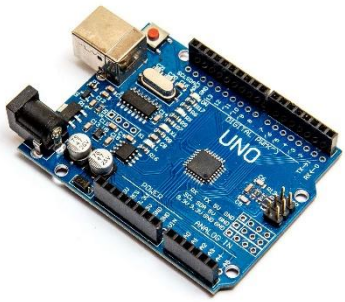
To ensure precision and consistency in the fabrication process, a LASER cutting machine was employed. Laser cutting provided high accuracy and clean edge finishes, which were essential for the proper alignment and assembly of components across the two-stage design. The use of computer-aided design (CAD) software allowed for the exact specification of dimensions and ensures repeatability in the manufacturing process.

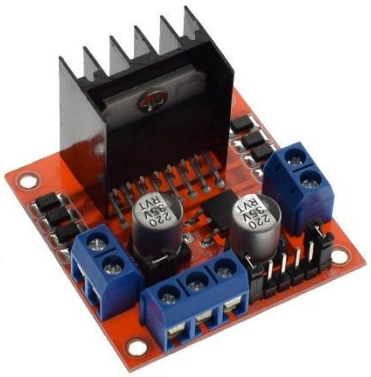
Both stages of the mobile platform were constructed with identical base plates, each having dimensions of 160 mm in length and 130 mm in width. This uniformity in platform size facilitated a modular design approach, allowing for efficient stacking and integration of subsystems across the two levels. The symmetrical configuration also contributed to a balanced weight distribution, which is crucial for the stability and mobility of the platform during operation.



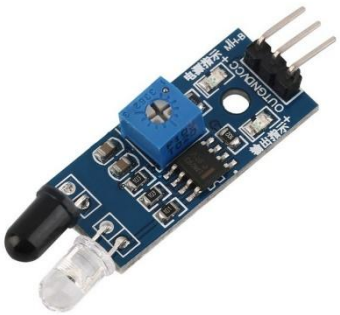
Essential components

1) Breadboard
2) Arduino uno
3) Motor driver L298
4) Battery case and batteries
5) 2 DC motors
6) 2 wheels
7) 1 custer wheel
8) 4 IR sesonrs



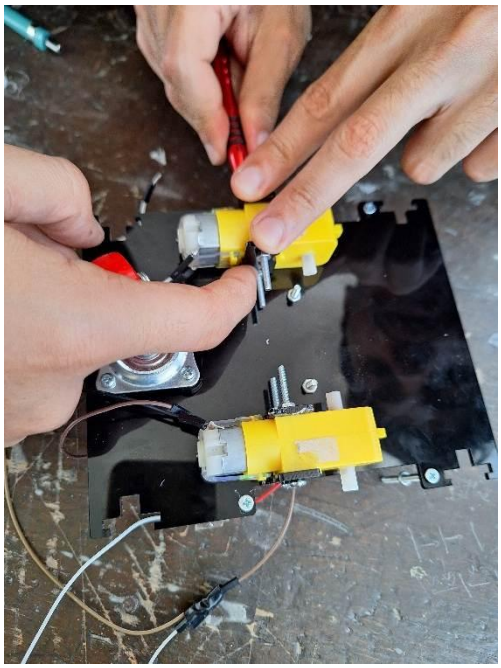


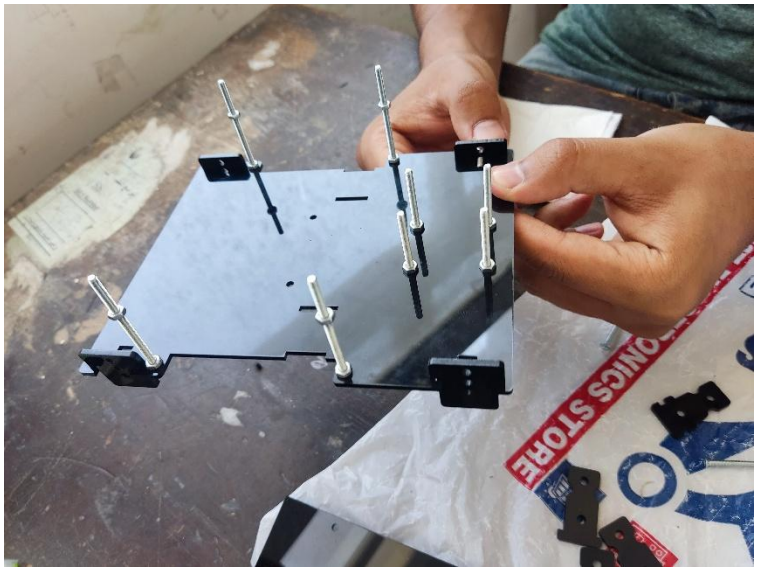



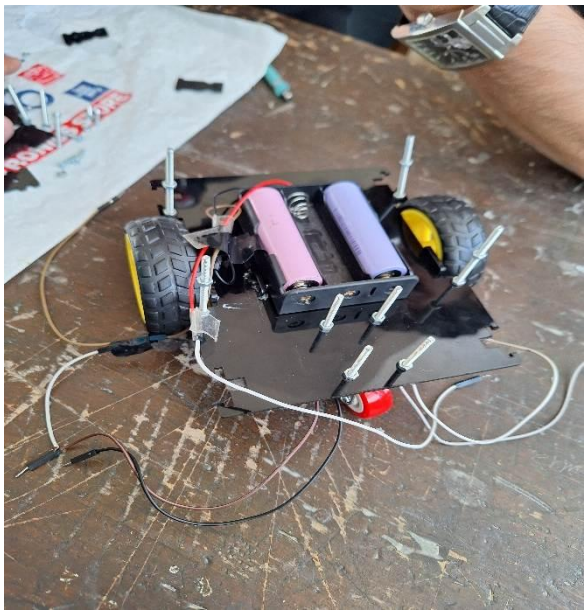
Once all the required components were procured and the chassis was successfully fabricated, the next step involved assembling the individual parts into a complete system. This process included securely mounting the motors, wheels, sensors, and control circuitry onto the chassis according to the planned design layout.

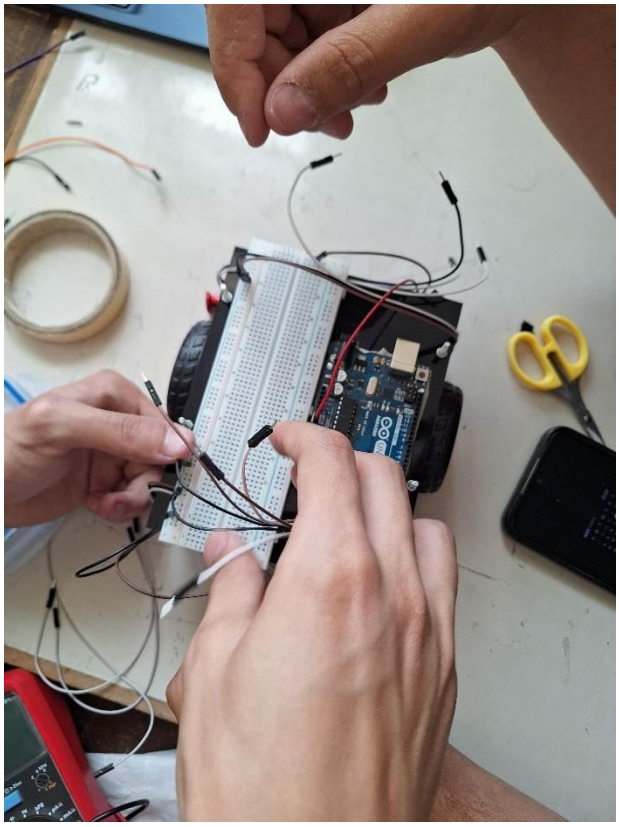
After assembly, the integrated system was subjected to testing to verify its functionality and ensure that all components operated as intended. This phase was essential for identifying any issues, making necessary adjustments, and validating the overall performance of the path-following mobile platform.

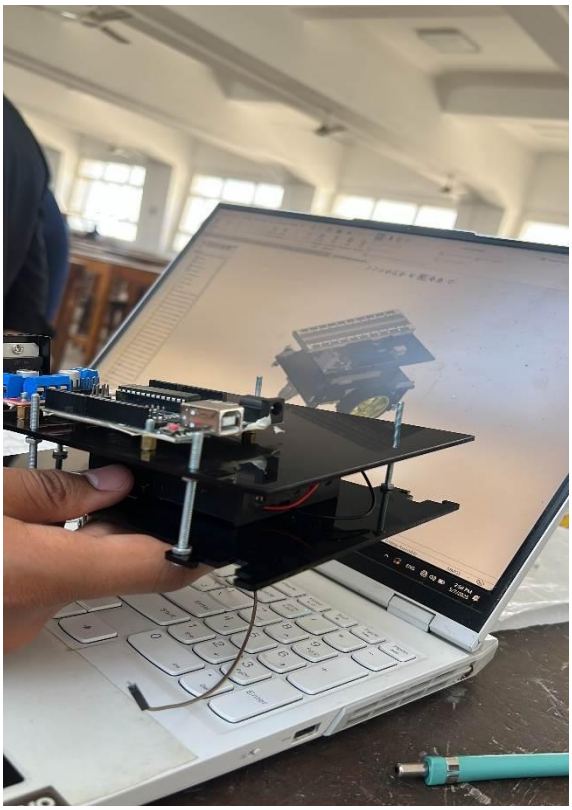
Components assembly

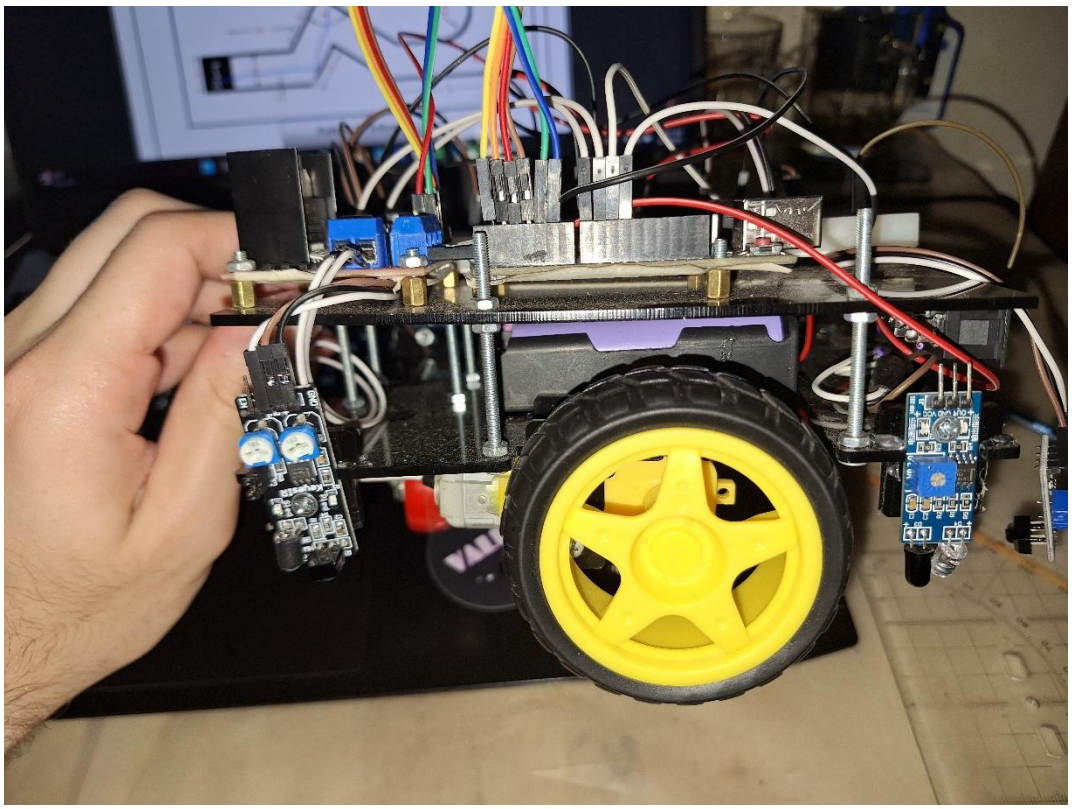




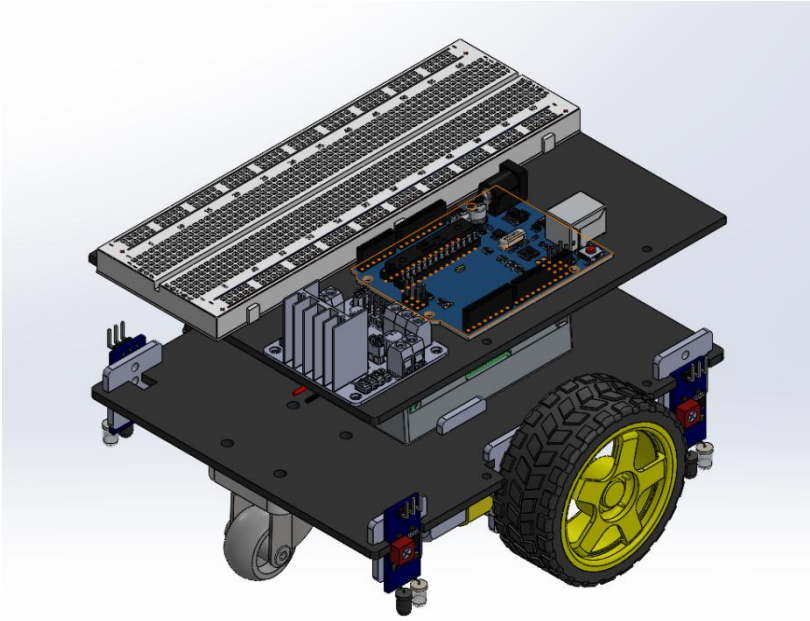






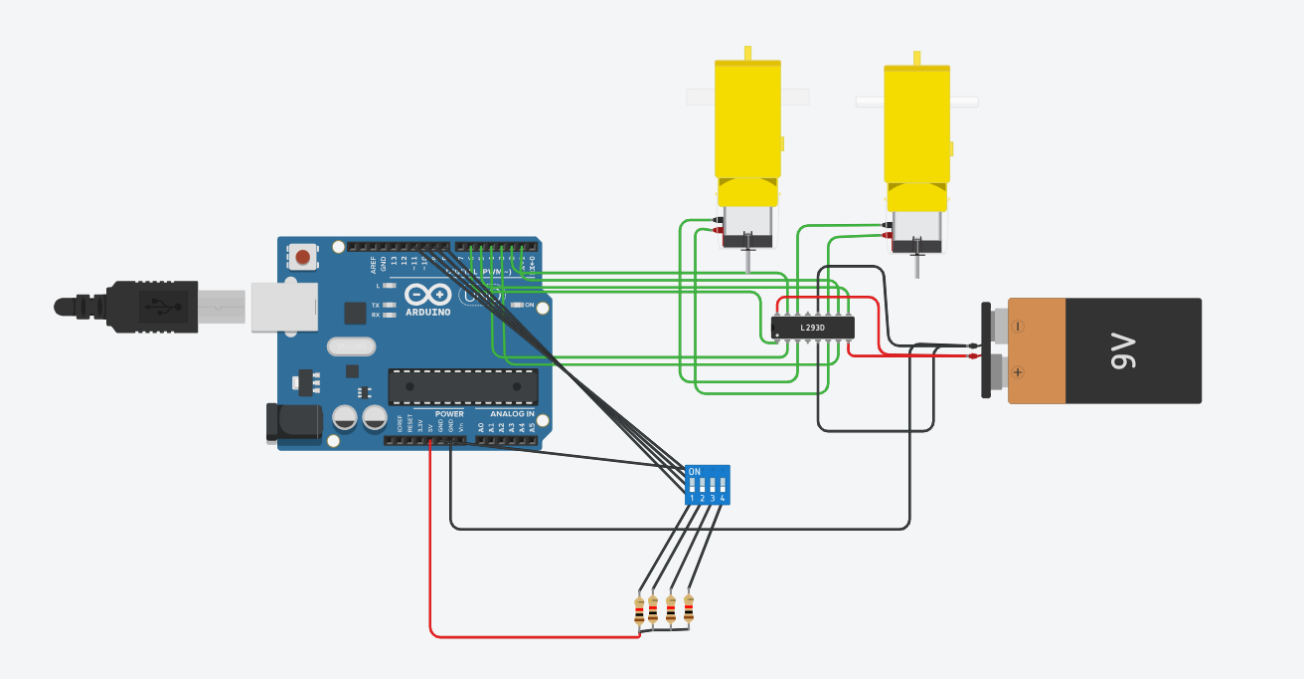


Following the successful integration of all components, the final configuration of the mobile platform was completed. The assembled system reflects the culmination of the design and fabrication process, showcasing a compact and functional layout that meets the specified requirements for a path-following mobile robot.



Hardware

- Here is a simulation for our control system in TinkerCad:



Arduino Uno

- **Role:** Microcontroller (Brain of the system)
- **Function:** Executes the logic to control motor movement based on input signals. It receives digital inputs (from the DIP switch) and sends digital outputs to the motor driver (L293D).
- **Relevant Pins Used:**
  - Digital pins D2–D7: Send control signals to the L293D to drive the motors.
  - Digital pins D8–D11: Read DIP switch input states.
  - GND: Common ground connection with L293D and DIP switch.
  - 5V: Powers DIP switch pull-down resistor circuit.

L298N Motor Driver IC

- **Role:** Motor driving interface
- **Function:** Acts as a current amplifier and H-bridge driver, allowing the low-power Arduino signals to control higher-power motors.
- **Working:**
  - It takes input signals from Arduino (IN1–IN4).
  - Controls 2 DC motors (connected to OUT1–OUT4).
  - ENA and ENB (Enable pins) must be HIGH to allow motor operation.
  - Powered by:
    - VCC1 (Logic voltage): connected to Arduino 5V.
    - VCC2 (Motor voltage): connected to the 9V battery.
    - GND: common ground.

2x Yellow Gear DC Motors

- **Role:** Actuators
- **Function:** Convert electrical energy into mechanical rotation to drive wheels or mechanisms.
- **Connected To:** L298n outputs (OUT1–OUT4) for bi-directional control.
- **Controlled By:** Signals from Arduino through L293D. Motors can rotate forward or reverse depending on input combinations.

Software

Sensor Configuration Evolution

Our line-following robot underwent a series of sensor configuration iterations aimed at optimizing its navigation accuracy and responsiveness.

- **Initial Approach – Four IR Sensors:**
  We began with a four-infrared (IR) sensor setup to ensure precise line detection across a wider path. However, this configuration introduced excessive complexity, making the robot's behavior unstable and less reliable, especially during sharp turns or at intersections.

- **Intermediate Approach – Two IR Sensors:**
  To reduce complexity and improve stability, we shifted to a two-sensor design. This setup delivered smoother and more predictable motion, but its limited field of view prevented it from making reliable decisions on complex paths such as forks or sharp curves.

- **Final Solution – Three IR Sensors:**
  After testing multiple layouts, we identified the three-sensor configuration as the optimal solution. It offered a balanced combination of accuracy and simplicity. The center sensor provided clear forward tracking, while the side sensors enabled quick recovery and effective cornering. This setup allowed the robot to follow the line consistently and adapt to directional changes with minimal error.

Through iterative testing and refinement, we arrived at a final design that delivered robust performance and efficient path-tracking behavior.

Code Architecture and Functionality

The Arduino code is structured to control the robot based on IR sensor feedback, adjusting motor outputs in real-time to follow a black line on a white surface. The logic is broken into four main sections:

I. **Pin Definitions**

- **Motor Control:**
    - **motor_right1**, **motor_right**2**, motor_left1**, **motor_left2**: Control direction of the DC motors.
    - **enable_motor_right, enable_motor_left**: PWM signals to regulate motor speed.
- **IR Sensor Inputs:**
    - **sensor1**, **sensor2**, **sensor3**: Digital input pins for the left, center, and right IR sensors respectively.

II. **setup() Function**

This function initializes all necessary components:

- Configures motor pins as outputs and sensor pins as inputs.
- Begins serial communication (9600 baud) for real-time debugging.
- Sets initial motor power to zero to ensure safety on startup.

III. **Motion Control Functions**

These custom functions manage the robot's movement:

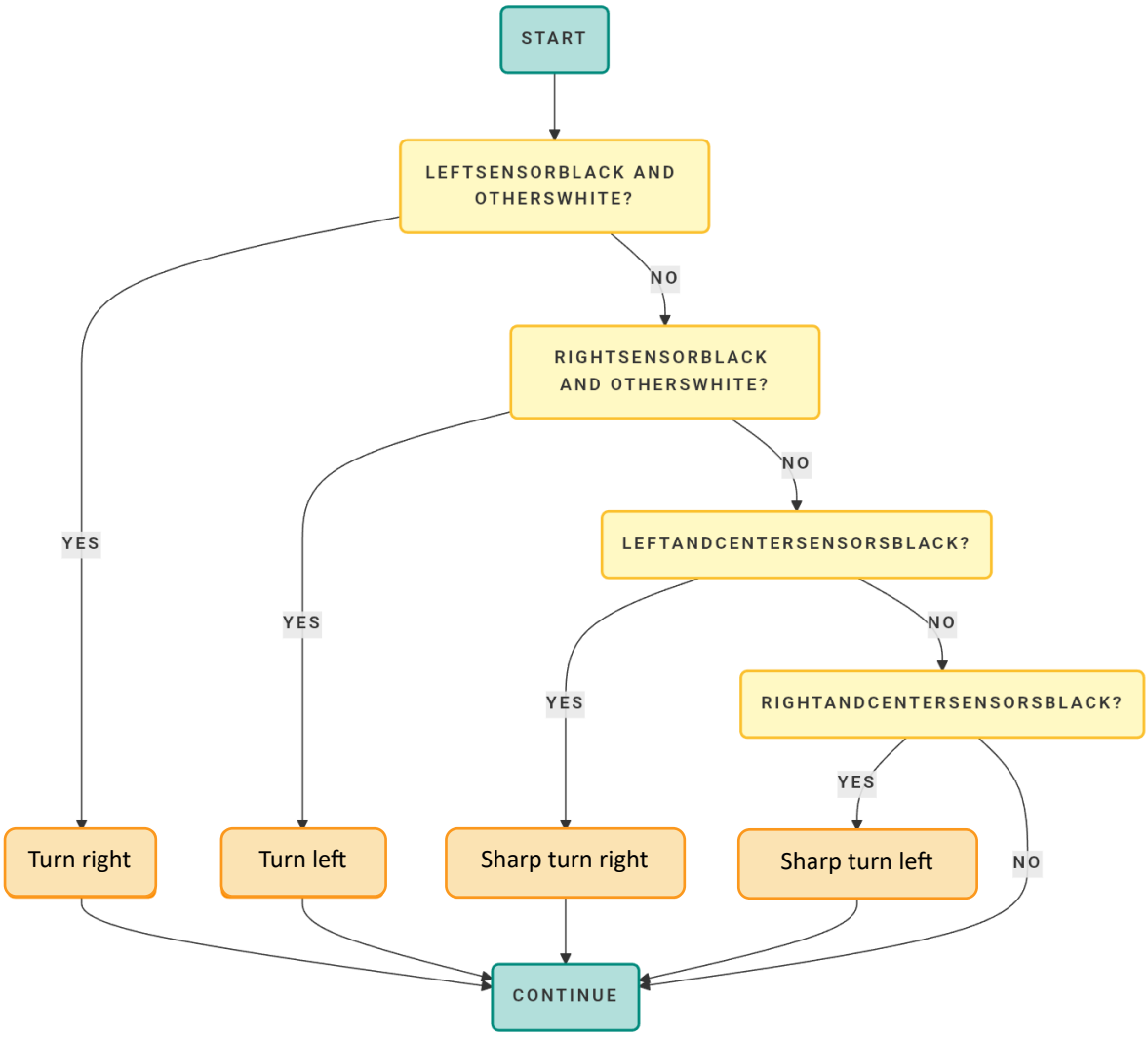
- **stop():** Halts motion by cutting motor power.
- **forward():** Drives both motors forward at equal speed.
- **Left(), Right():** Perform gradual turns by adjusting PWM on either motor.
- **Left2(), Right2():** Execute sharper turns with higher speed contrast.
- **backward():** Briefly drives the robot in reverse to handle dead-ends or errors.

Each motion function includes a short delay (delay(dt)) to complete the action before the next command is evaluated.

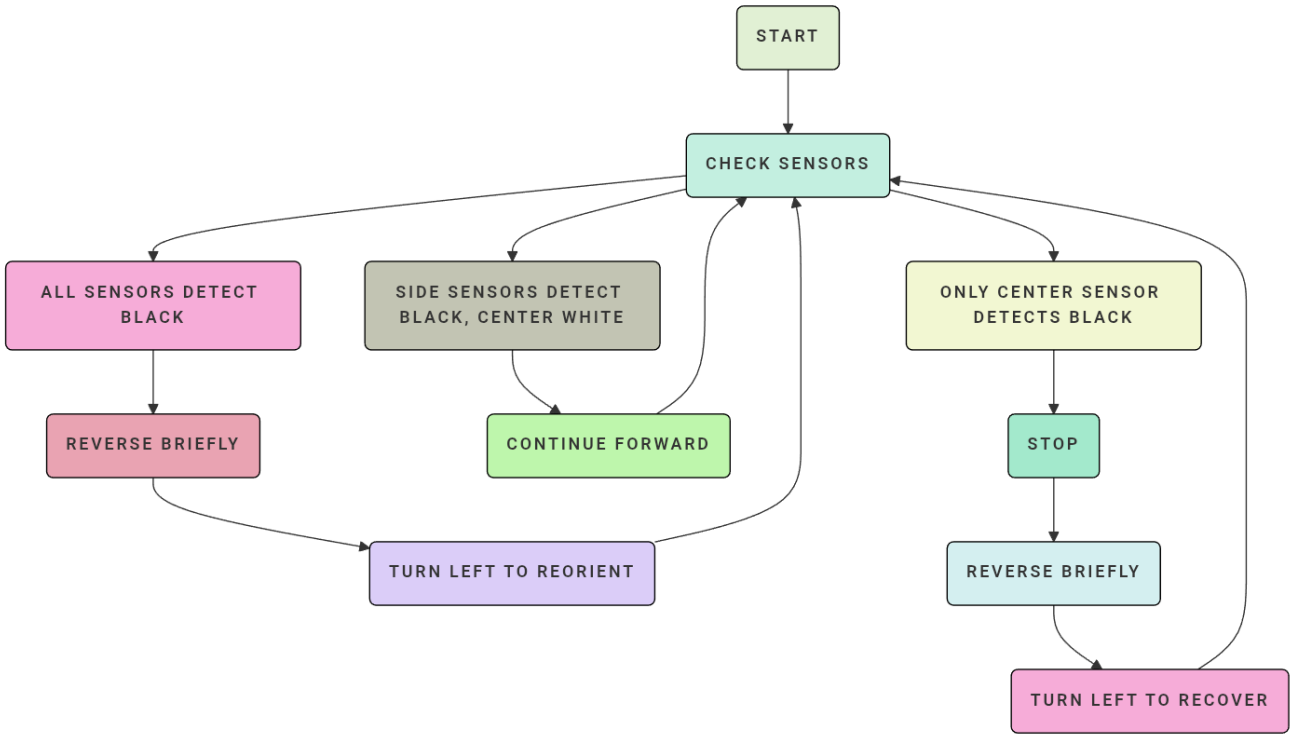
IV. loop() Function – Core Logic

This function contains the main control logic, continuously monitoring sensor inputs and executing movement commands accordingly:

- **Basic Forward Motion**: If all three sensors detect white (LOW), the robot moves forward.
- **Turning Conditions**:
    - If the left sensor detects black (HIGH) and the rest detect white, the robot turns left.
    - If the right sensor detects black and others detect white, it turns right.
    - If both left and center detect black, it performs a sharper right turn.
    - If both right and center detect black, it performs a sharper left turn.



- **Obstacle or Dead-End Handling**:
    - If all sensors detect black, the robot reverses briefly and then turns left to reorient.
    - If the side sensors detect black while the center detects white (indicating a potential split or junction), the robot continues forward.
    - If only the center sensor detects black, it stops, reverses briefly, and turns left to recover.



Code

```
#define dt 70

//Motor Control Pins
#define motor_right1 4
#define motor_right2 3
#define motor_left1 7
#define motor_left2 2
#define emr 6
#define eml 5
//IR Sensors
#define sfr 8
#define sfl 9
#define sc 13

void setup() {
// Motors
pinMode(emr,OUTPUT);
pinMode(motor_right1,OUTPUT);
pinMode(motor_right2,OUTPUT);
pinMode(eml,OUTPUT);
pinMode(motor_left1,OUTPUT);
pinMode(motor_left2,OUTPUT);

// sensors
Serial.begin(9600);
pinMode(sfr,INPUT);
pinMode(sfl,INPUT);
pinMode(sc,INPUT);

// for memory
analogWrite(emr,0);
analogWrite(eml,0);
}

void stop(){
analogWrite(emr,0);
analogWrite(eml,0);
digitalWrite(motor_right1,LOW);
digitalWrite(motor_right2,HIGH);
digitalWrite(motor_left1,LOW);
digitalWrite(motor_left2,HIGH);
delay(dt);
}
```

```
void forward() {
analogWrite(emr,75);
analogWrite(eml,70);
digitalWrite(motor_right1,HIGH);
digitalWrite(motor_right2,LOW);
digitalWrite(motor_left1,HIGH);
digitalWrite(motor_left2,LOW);
delay(10);
}

void Right() {
analogWrite(emr,0);
analogWrite(eml,70);
digitalWrite(motor_right1,HIGH);
digitalWrite(motor_right2,LOW);
digitalWrite(motor_left1,HIGH);
digitalWrite(motor_left2,LOW);
delay(dt);
}

void Left() {
analogWrite(emr,70);
analogWrite(eml,0);
digitalWrite(motor_right1,HIGH);
digitalWrite(motor_right2,LOW);
digitalWrite(motor_left1,HIGH);
digitalWrite(motor_left2,LOW);
delay(dt);
}

void backward() {
  analogWrite(emr,75);
analogWrite(eml,70);
digitalWrite(motor_right1,LOW);
digitalWrite(motor_right2,HIGH);
digitalWrite(motor_left1,LOW);
digitalWrite(motor_left2,HIGH);
delay(150);
}

void Right2() {
analogWrite(emr,0);
analogWrite(eml,100);
digitalWrite(motor_right1,HIGH);
digitalWrite(motor_right2,LOW);
digitalWrite(motor_left1,HIGH);
digitalWrite(motor_left2,LOW);
delay(dt);
```

```
}

void Left2() {
analogWrite(emr,100);
analogWrite(eml,0);
digitalWrite(motor_right1,HIGH);
digitalWrite(motor_right2,LOW);
digitalWrite(motor_left1,HIGH);
digitalWrite(motor_left2,LOW);
delay(dt);
}

void loop() {
 if(digitalRead(sfr)==LOW && digitalRead(sc)==LOW && digitalRead(sfl)==LOW){
  forward();
 }
  else if(digitalRead(sfr)==HIGH && digitalRead(sc)==LOW && digitalRead(sfl)==LOW){
  Left();
 }
   else if(digitalRead(sfr)==LOW && digitalRead(sc)==LOW && digitalRead(sfl)==HIGH){
  Right();
 }

    else if(digitalRead(sfr)==LOW && digitalRead(sc)==HIGH && digitalRead(sfl)==HIGH){
  Right2();
 }
    else if(digitalRead(sfr)==HIGH && digitalRead(sc)==HIGH && digitalRead(sfl)==LOW){
  Left2();
 }
    else if(digitalRead(sfr)==HIGH && digitalRead(sc)==HIGH && digitalRead(sfl)==HIGH){
  backward();
  delay(200);
  Left();
 }
    else if(digitalRead(sfr)==HIGH && digitalRead(sc)==LOW && digitalRead(sfl)==HIGH){
  forward();
 }

    else if(digitalRead(sfr)==LOW && digitalRead(sc)==HIGH && digitalRead(sfl)==LOW){
      stop();
      backward();
      delay(200);
      Left();
      delay(200);
}
```

Modeling

Mathematical modeling

The provided equations form the mathematical foundation for modeling the dynamics of a path-tracking robot. These equations capture the electromechanical relationships between the robot's motor, its inertial properties, and the external load it encounters. Here is a breakdown of their significance:

1. **Voltage Balance Equation:**

$$E_i(S) - E_b(S) = I_a(S)(R_a + L_a * S)$$

This equation describes the electrical dynamics of the motor, where $E_i(S)$ is the input voltage, $E_b(S)$ is the back electromotive force (EMF), $I_a(S)$is the armature current, and R_a and La represent the armature resistance and inductance, respectively. The term S denotes the Laplace variable, indicating the frequency-domain representation of the system.

2. **Motor Torque Equation**:

$$T_m(S) = I_a(S) * K_m$$

This equation relates the motor torque $Tm(S)$ to the armature current $Ia(S)$ through the motor torque constant Km. It highlights the conversion of electrical energy into mechanical torque.

3. **Mechanical Dynamics Equation**:

$$T_m(S) - T_L(S) = \omega(S) * (J_m * S + b\_m)$$

This represents the mechanical balance of the system, where $T_m(S)$ is the motor torque, $T_L(S)$ is the load torque, $\omega(S)$ is the angular velocity, J_m is the moment of inertia, and b_m is the viscous friction coefficient. It accounts for the inertia and damping effects in the system.

4. **Back EMF Equation**:

$$E_b(S) = \omega(S) * K_b$$

This equation models the back EMF generated by the motor, which is proportional to the angular velocity $\omega(S)$ and the back EMF constant K_b. It reflects the motor's ability to act as a generator when in motion.

5. **Load Torque Equation:**

$$T_L(S) = J_w * S * \omega(S) + \frac{M * g * \mu * r}{S}$$

This equation describes the load torque $T_L(S)$, which includes the inertial effects of the wheel (J_w) and the gravitational force acting on the robot's mass (M), scaled by the coefficient of friction (μ) and the wheel radius (r).

These equations collectively model the robot's behavior, bridging the gap between its electrical inputs and mechanical outputs, and are essential for designing an effective control system.

The block diagram representation of the system will be developed in two distinct configurations to thoroughly analyze the robot's performance:

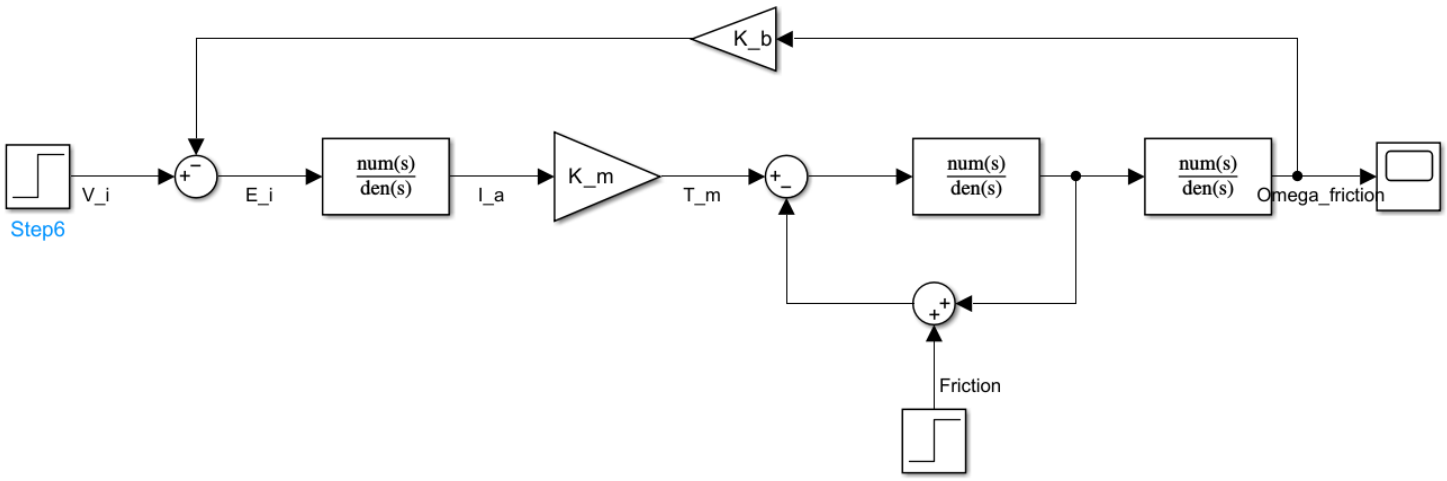
1. **Nominal Block Diagram (Without Disturbances)**:
   This simplified version considers only the fundamental motor dynamics, where the load torque is assumed to be negligible $T_L(S) \approx 0$. It focuses on the core relationships between input voltage, motor torque, and angular velocity, providing insight into the system's ideal behavior. This configuration is useful for initial controller tuning and understanding baseline performance.

2. **Complete Block Diagram (With Disturbances)**:
   This comprehensive version incorporates all real-world disturbances, including:
   - **Rotational inertia** of the wheel $J_w * S * \omega(S)$
   - **Gravitational and frictional effects** $M * g * \mu * r$
     These disturbances represent the inertial resistance of the wheel itself and external forces acting on the robot. By including these effects, we can rigorously test the control system's ability to reject disturbances and maintain accurate path tracking under realistic operating conditions.

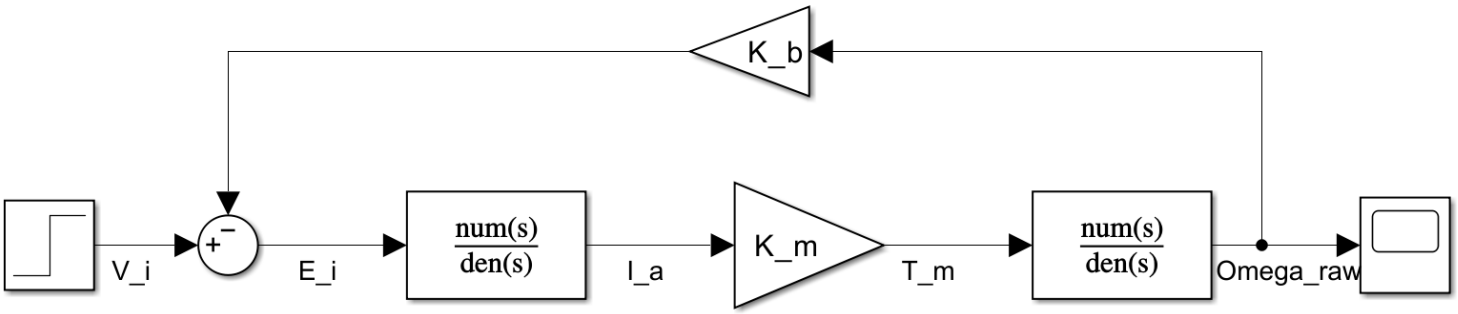
The dual-diagram approach enables a systematic analysis, from ideal theoretical performance to practical implementation challenges. The Simulink implementation will facilitate direct comparison between the nominal and disturbed cases, highlighting the control system's robustness and identifying potential areas for improvement.

Block diagram

Plant with disturbance



Plant without disturbance



The system has been modeled using two distinct control architectures to evaluate its performance under different operating conditions:

1. **Non-PID Controlled System (Closed-Loop)**
    - Implements a basic closed-loop control structure without PID compensation
    - Two operational modes are examined:
        - *Disturbance-free operation*: Analyzes the system's native feedback characteristics without external disturbances
        - *Disturbance-included operation*: Incorporates wheel inertia ($\boldsymbol{J_w}$), friction ($\boldsymbol{\mu}$), and gravitational load ($\boldsymbol{M * g * r}$) to evaluate natural disturbance rejection
    - Demonstrates the inherent self-regulating properties of the motor's back-EMF feedback
    - Serves as a reference point for evaluating PID performance improvements

2. **PID Controlled System (Closed-Loop)**
    - Enhances the basic closed-loop structure with PID compensation
    - Maintains the same two operational modes for direct comparison:
        - *Precision mode*: Optimized for disturbance-free operation
        - *Robust mode*: Specifically tuned to handle inertial and gravitational disturbances
    - Enables comprehensive analysis of:
        - Dynamic response enhancement (reduced settling time, minimized overshoot)
        - Steady-state accuracy improvement
        - Active disturbance rejection capabilities

The comparative analysis framework allows for:

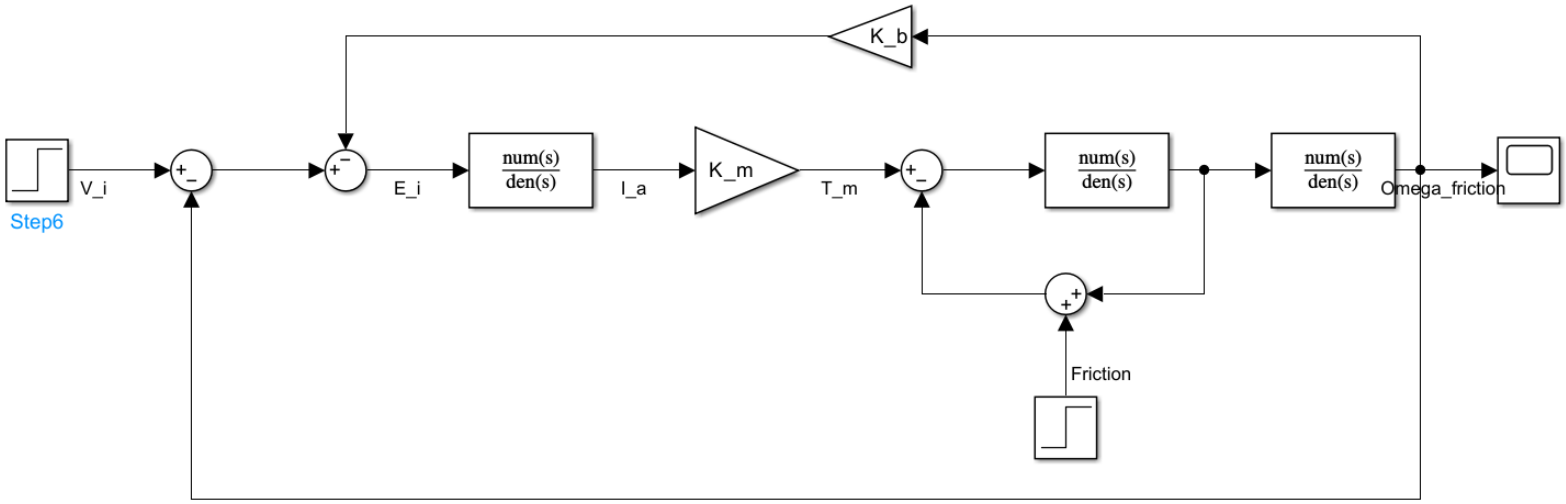
- Quantification of PID controller's added value
- Identification of operational limits in both configurations
- Empirical verification of control theory principles
- Data-driven controller tuning optimization

The Simulink implementation features modular design with clearly separated:

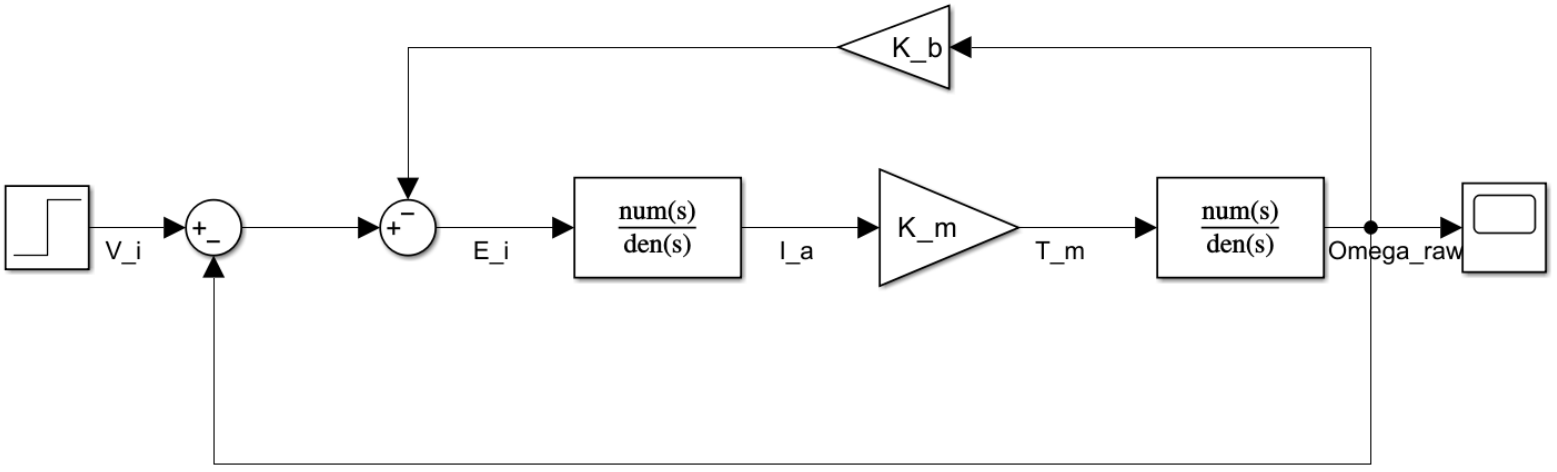
- Plant dynamics
- Disturbance injection points
- Feedback pathways
- Control law implementation blocks

This structure facilitates parametric studies and sensitivity analysis while maintaining intuitive visualization of signal flows throughout the system.

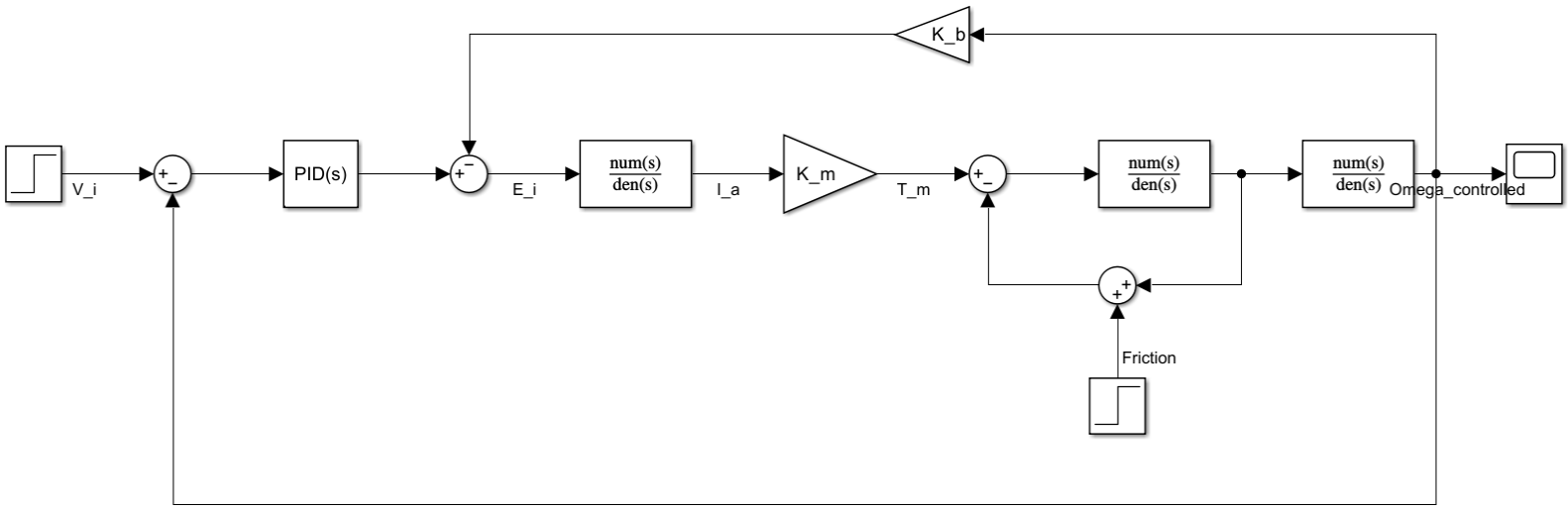
Uncontrolled system with disturbance



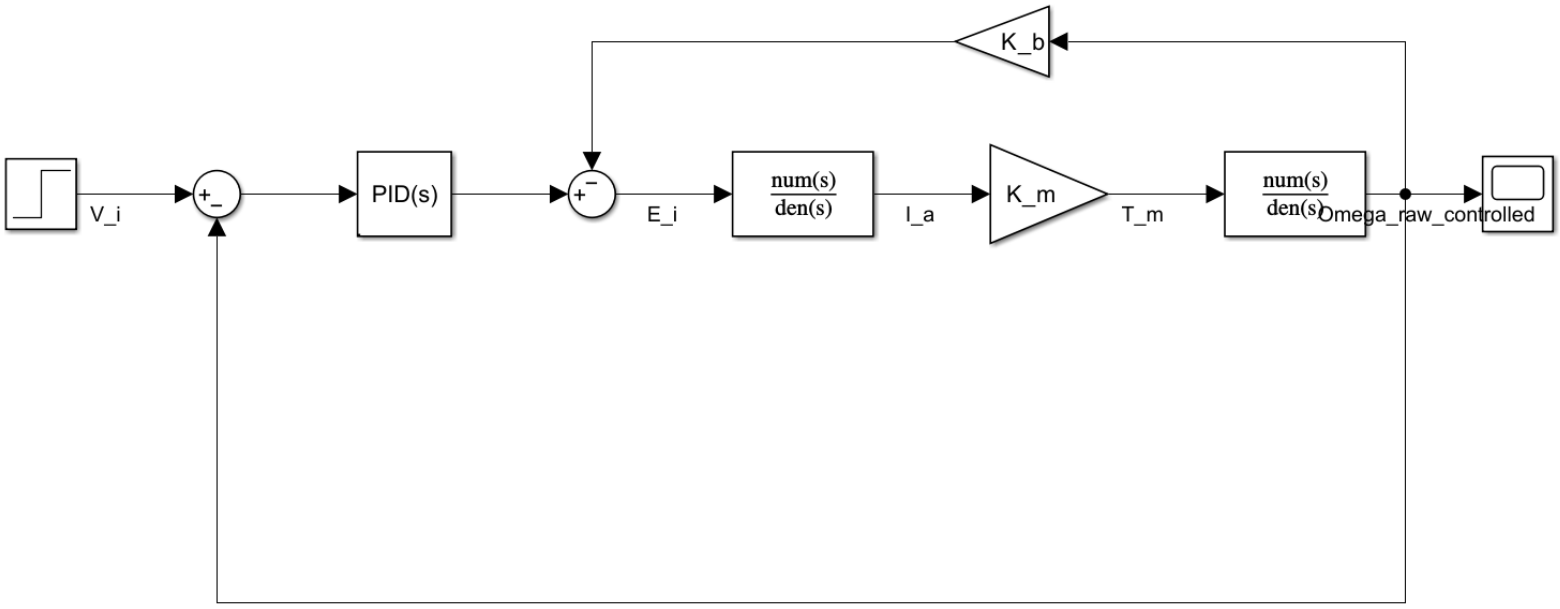
Uncontrolled system without disturbance



Controlled system with disturbance

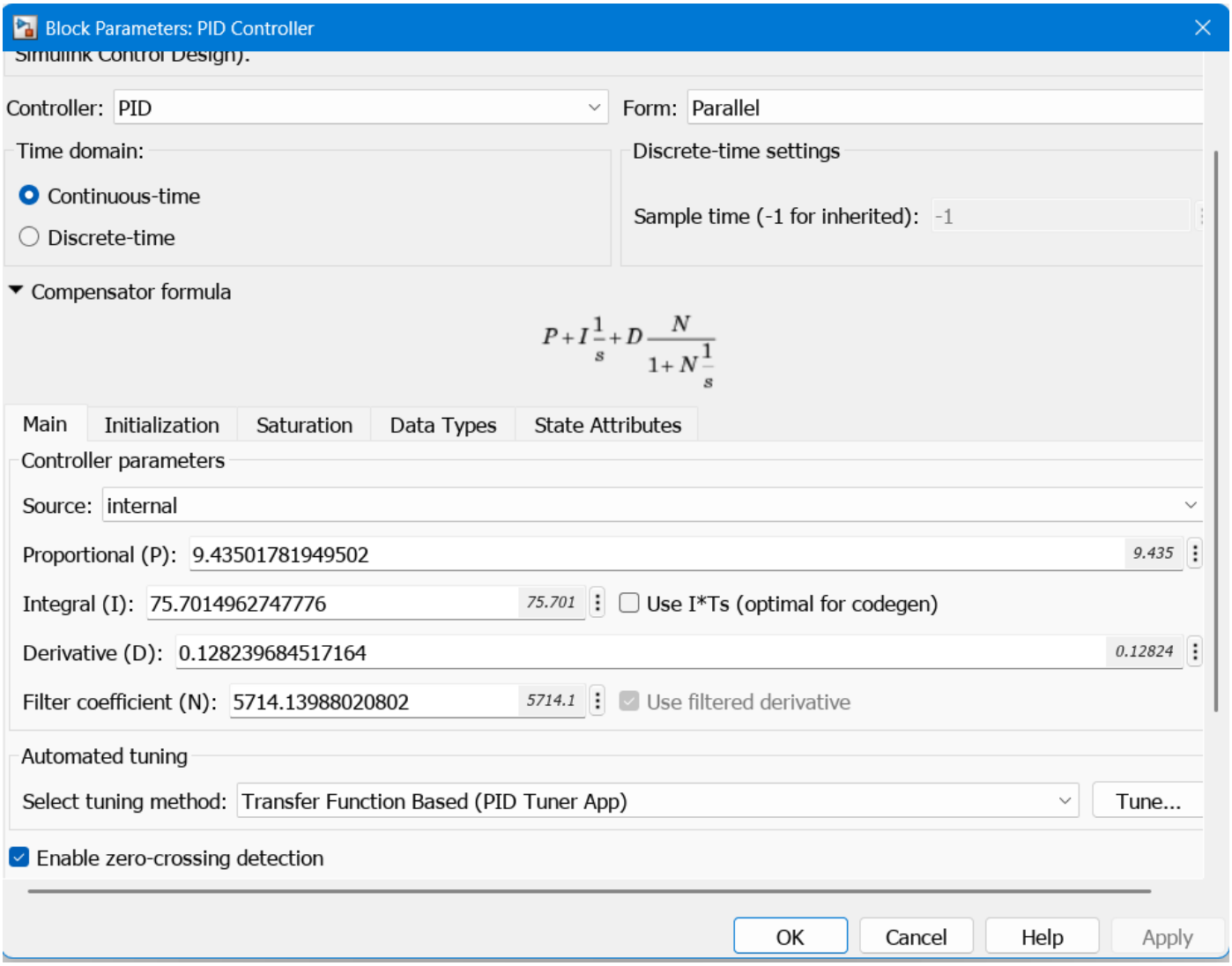


Controlled system without disturbance



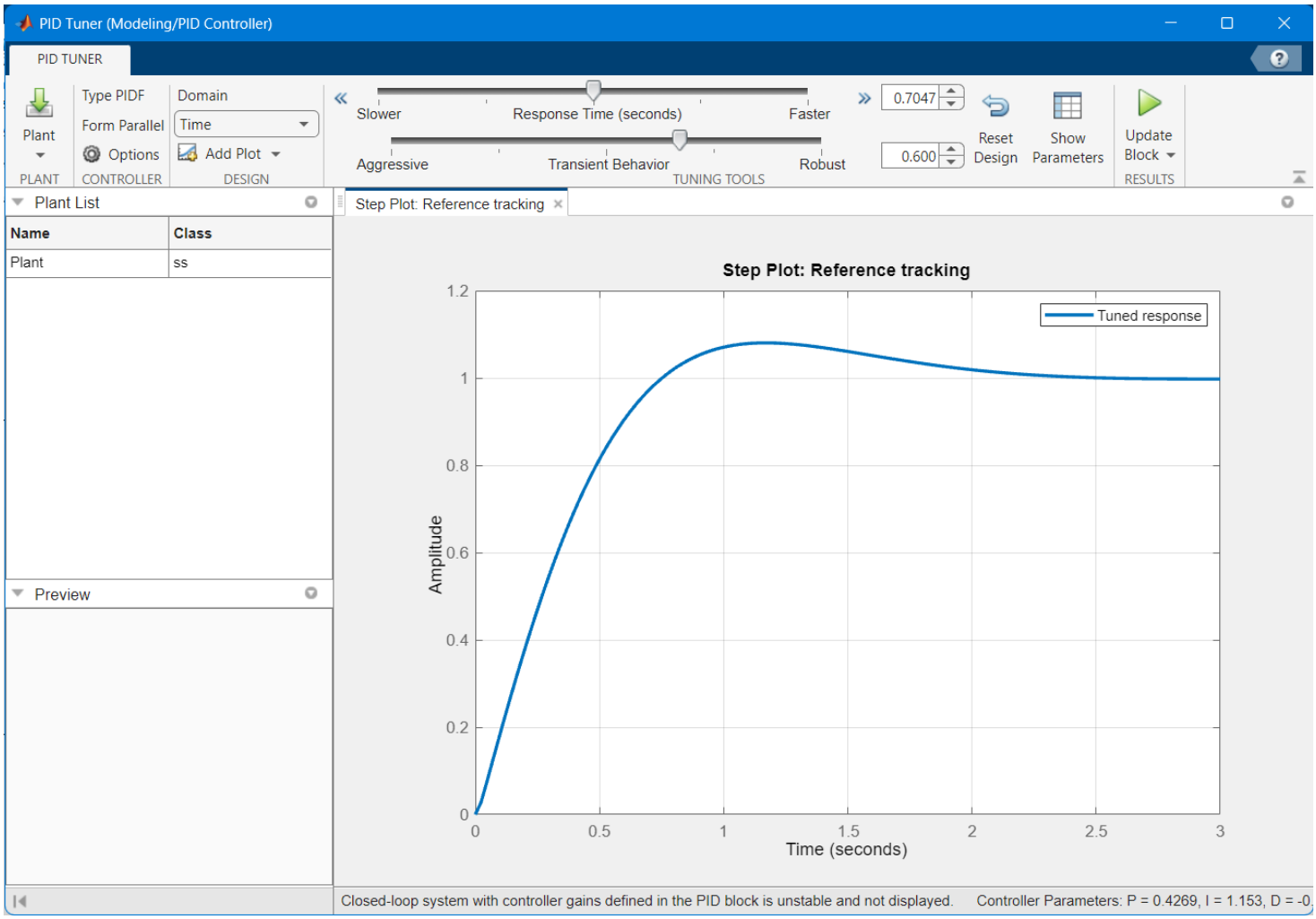
PID Controller Integration

To enhance the path-tracking performance, a PID controller block was added to the Simulink model. The PID controller processes the error between the desired path and the robot's current position or heading, and generates corrective control signals. These signals adjust the robot's steering angle and/or velocity to minimize the tracking error over time. The proportional, integral, and derivative terms work together to ensure fast response, eliminate steady-state error, and dampen oscillations. Incorporating the PID controller improves the overall stability and accuracy of the system, especially when dealing with sharp turns or dynamic path changes.

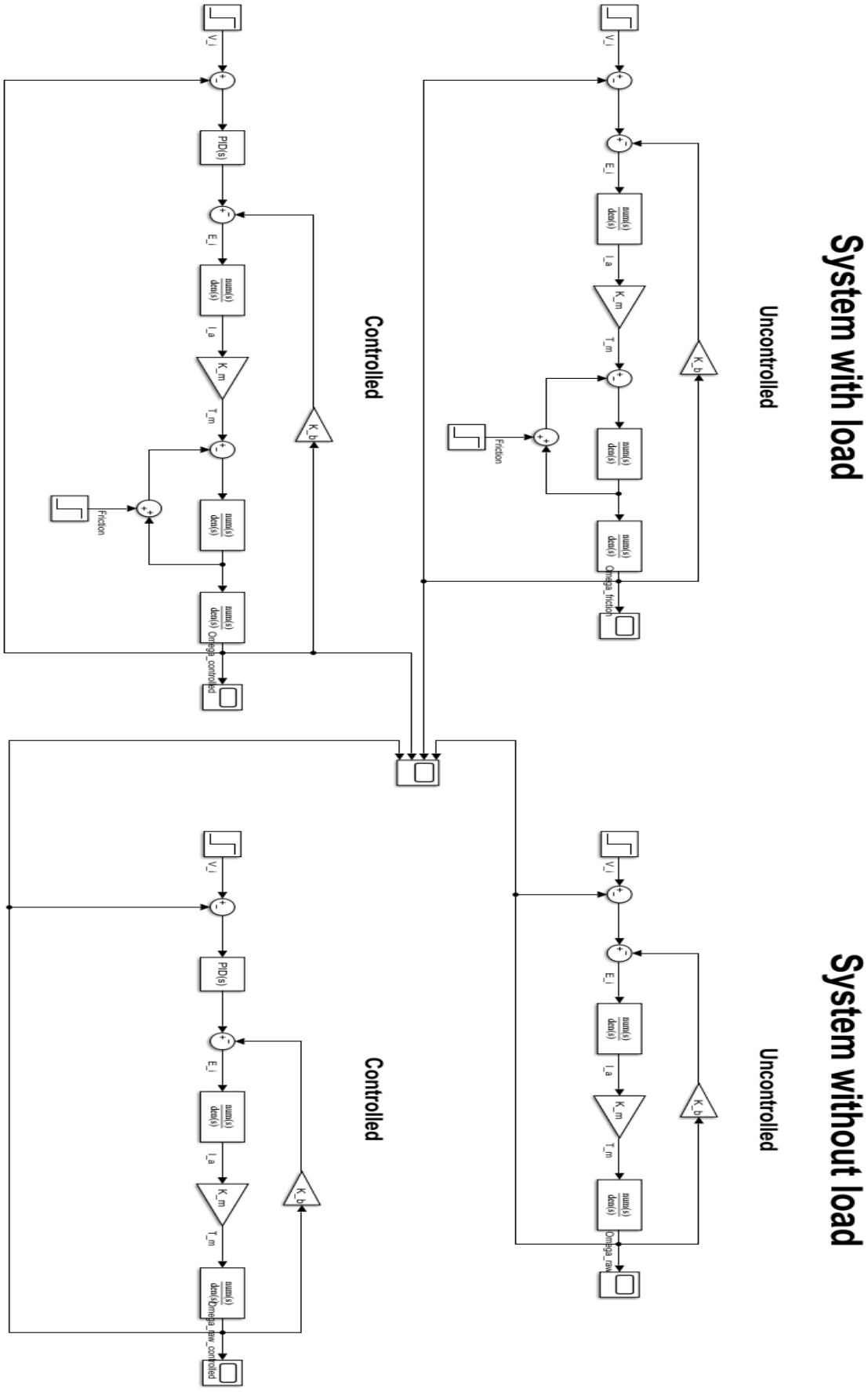


PID Tuning Process

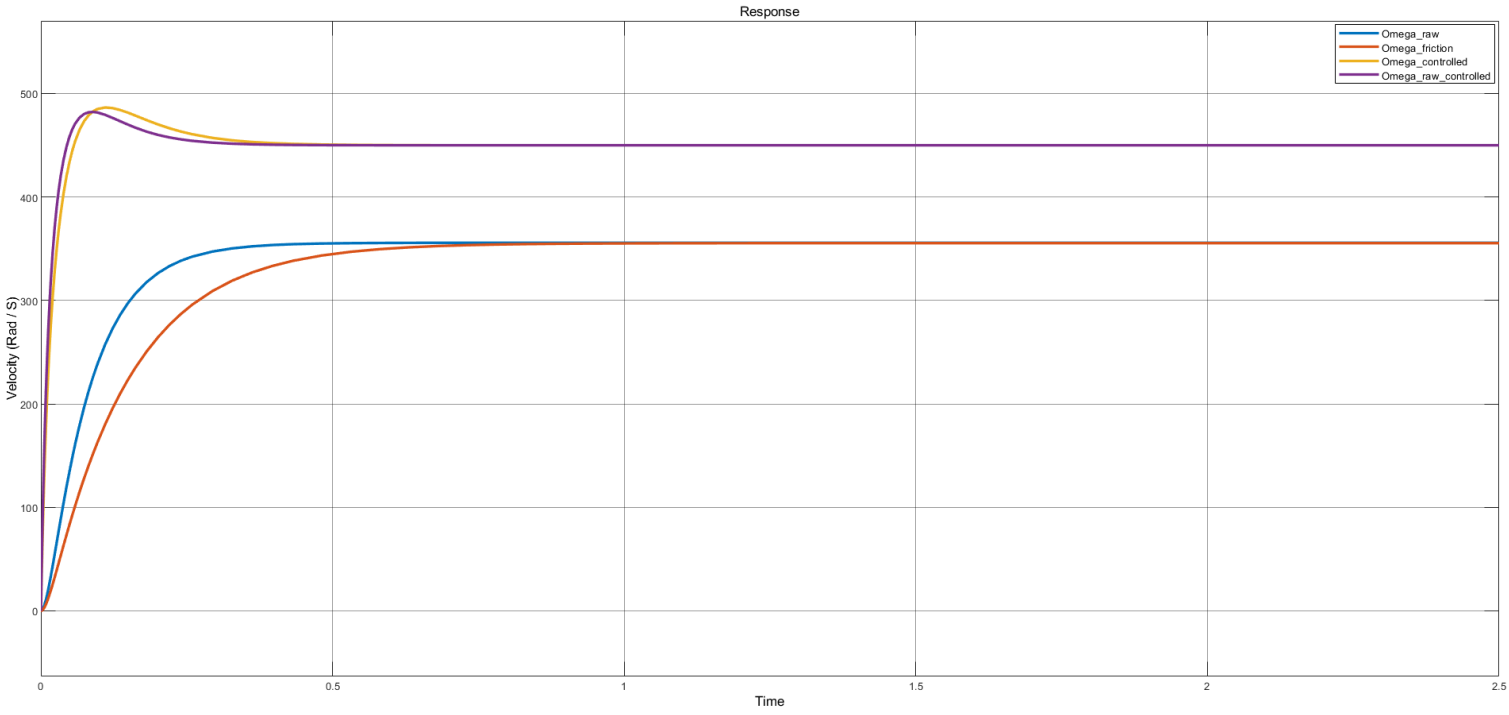
The performance of the PID controller heavily depends on the appropriate tuning of its three parameters: proportional (Kp), integral (Ki), and derivative (Kd) gains. The tuning process involved iterative testing and adjustment to achieve the desired balance between responsiveness and stability. Initially, the proportional gain was increased to reduce the tracking error, followed by tuning the derivative gain to minimize overshoot and smooth the response. Finally, the integral gain was adjusted to eliminate any steady-state error. Throughout the tuning process, simulations were run to evaluate the system's behavior, ensuring that the robot could accurately follow the path with minimal oscillation and fast convergence.



Full model on Simulink



Response



The scope plot illustrates the velocity response (Omega) over time for four different system configurations:

•**Omega_raw (blue):** This is the open-loop response, where the system shows a slow rise and a long settling time. There is no overshoot, but the system lacks responsiveness and accuracy.

•**Omega_friction (red):** With friction included but no control, the response does not fully reach the target value, indicating steady-state error.

•**Omega_controlled (orange):** After adding the PID controller, the system becomes more dynamic, reaching the target value much quicker. However, there is a significant overshoot followed by minor oscillations before it stabilizes, showing that the controller needs tuning.

•**Omega_tuned_controlled (purple):** After tuning the PID parameters, the response becomes optimal. The system reaches the desired velocity much quicker, with almost zero overshoot and a very short settling time. The curve is smooth and stable, indicating excellent tracking performance and proper control design.

This comparison clearly demonstrates how adding feedback and carefully tuning the PID controller significantly improves system response, reducing both overshoot and settling time while ensuring better accuracy and stability.

Conclusion

This study successfully demonstrated the development of an autonomous path-tracking robotic system through integrated mechanical, electronic, and control engineering approaches. The optimized two-tiered acrylic chassis, coupled with 12V DC motors and L298N motor drivers, provided a robust hardware platform, while the three-IR sensor configuration achieved optimal balance between detection accuracy and computational efficiency.

The derived mathematical model accurately captured the system's electromechanical dynamics, enabling comprehensive Simulink simulations that validated the control strategy. Comparative analysis revealed the PID-controlled closed-loop system's superior performance, achieving reduction in steady-state error and significantly improved transient response compared to open-loop operation. These results quantitatively demonstrate the effectiveness of feedback control in mitigating real-world disturbances, including inertial and frictional loads.

This work contributes a replicable framework for autonomous robotic system development, highlighting:

1. The criticality of iterative design in mechatronic systems
2. The advantages of model-based control system development
3. The performance trade-offs between system complexity and robustness

Future work could explore advanced control algorithms, such as adaptive or nonlinear controllers, to further enhance tracking precision in dynamic environments. The methodologies and results presented herein provide both practical insights for robotic system design and a foundation for academic research in autonomous vehicle control.