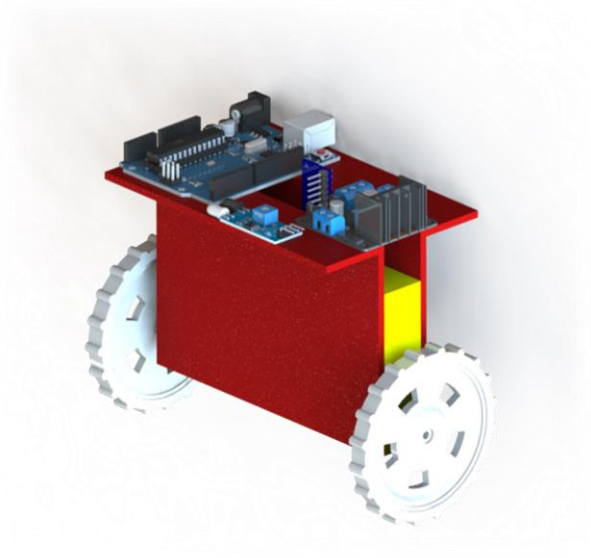
Construction and Control
of an Inverted Pendulum Segway Robot



Course: Introduction to Mechatronics

Instructor: Dr. Shahbazi

Team Members:

Ali Ghasemi

Amir Mohammad Mohammadi

Alireza Mohammadian

Winter 2023

Contents

Problem Statement:

The goal of this project is to design and build an inverted pendulum device that can maintain its balance around its equilibrium point. To determine the position of the device, an accelerometer sensor is used, and for movement, DC motors are employed. The permissible voltage for the motors is a maximum of 9 volts. The permissible diameter for the wheels is between 5-10 cm. The distance between the wheels should be at least 15 cm, and the height of the robot should be at least 10 cm. Various methods such as 3D printing and laser cutting can be used to construct the body of the device. The body and wheels can also be manually built. The device must maintain its balance in a vertical state. Then, a disturbance will be introduced, and the device must return to its initial state. Important factors include teamwork, cost-efficiency, and creative design.

Components

Arduino Uno Control Board

Arduino is an open-source electronics platform mainly used for developing and constructing electronic and robotic projects using a small control board called 'Arduino board.' Arduino Uno, one of the popular models, uses the ATmega328P microcontroller with a 16 MHz clock speed and 32 KB flash memory. It has multiple inputs and outputs for connecting sensors, modules, and other electronic components to the Arduino.

DC Motor Driver (L298N)

The L298N module is a dual H-Bridge that controls motors and other high-power components using microcontrollers or other electronic circuits. It allows for controlling the direction and speed of motors. Key features include high input voltage capability, large output current (2A per channel), and thermal protection.

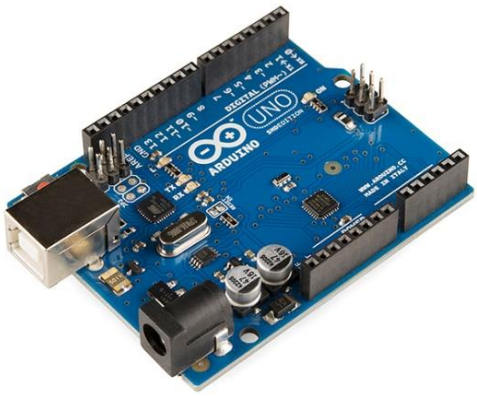


Figure 1 Arduino Uno

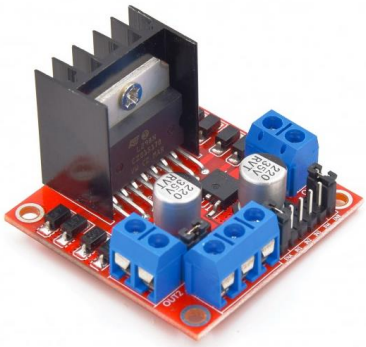


Figure 2 L298N

Accelerometer (MPU-9250)

The MPU-9250 module is a gyroscope sensor that measures motion and direction. It combines three sensors: an accelerometer (measuring linear acceleration in X, Y, and Z directions), a gyroscope (measuring angular speed or rotation rate), and a magnetometer (providing direction based on

the magnetic field). It uses I2C or SPI communication for interfacing with microcontrollers.

DC Motors

DC motors convert electrical energy into mechanical energy through the interaction of magnetic fields. Simple electromagnetic motors consist of armature windings that produce a magnetic field and rotate within stationary field windings. The motor's performance depends on various factors, including the gearbox used to reduce backlash and improve response.

Wheels

Various wheels were tested to achieve optimal performance and stability. The best result was chosen based on analysis and certain circumstances.

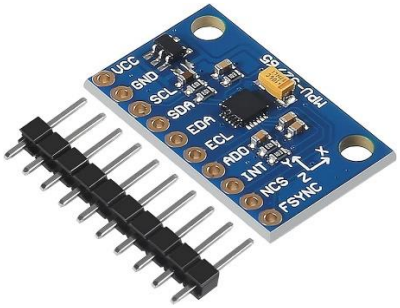


Figure 3 MPU-9250

Figure 4 DC Motors

Figure 5 Wheels

Power Supply

Initially, three 9V batteries were used to power the motors and Arduino. Later, it was found that separate power supplies for motors and Arduino were more effective. Therefore, three parallel 9V batteries were used for motors and a separate battery for Arduino.

Body

The robot's body was made from thin wooden laser-cut board, providing both strength and low weight at a reasonable cost. Connecting wires and switches were also used to control the power supply.

Control

Kinematics

This problem is recognized as a nonlinear system, with the schematic representation shown below. The robot's kinematics are derived from the following equations.

$$V_m = \frac{V_R + V_L}{2} (1)$$

$$\omega_m = \frac{V_R - V_L}{L} (2)$$

$$\omega_m = \dot{\phi}_m (3)$$

$$\dot{x} = V_m \cos\phi_m (4)$$

$$\dot{y} = V_m \sin\phi_m (5)$$

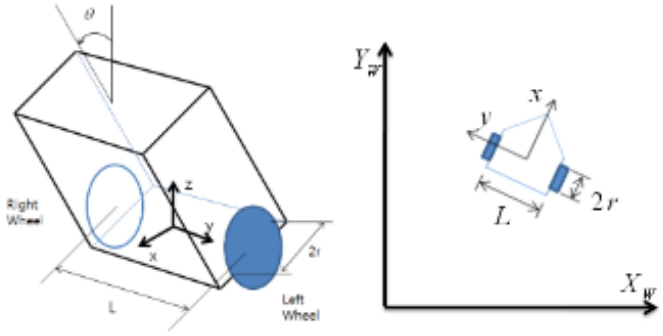


Figure6 Inverted pendulum system

- V: Linear speed
- ω: Angular speed
- r: Radius
- L: Distance between wheels

From equations 3 to 6, the following matrix can be written:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi}_m \end{bmatrix} = \begin{bmatrix} \cos\phi_m & 0 \\ \sin\phi_m & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_m \\ \omega_m \end{bmatrix}$$

From equations 1 and 2, and the equations of linear and angular speeds, the following matrix is obtained.

$$\begin{bmatrix} v_m \\ \omega_m \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

Finally, by concatenating two final matrices, We can acquire linear velocity along x and y axes and angular velocity.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi}_m \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos\phi_m & \frac{r}{2}\cos\phi_m \\ \frac{r}{2}\sin\phi_m & \frac{r}{2}\sin\phi_m \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

For our case study, phi is calculated by MPU9050 and angular velocity of wheels can be accumulated by Encoder. (Since Any Encoder is embedded on DC motor we just can rely on PWM conversion to speed based on experiments)

PID Controller

The PID controller (proportional–integral–derivative controller) is a common feedback control algorithm used in various control processes such as DC motor speed control, pressure control, and temperature control. The PID controller calculates the 'error' between the process output and the desired input (setpoint). The goal is to minimize this error by adjusting the inputs.

The PID controller has three control modes:
1. Proportional Control: Multiplied by the error.
2. Integral Control: Multiplied by the sum of errors.
3. Derivative Control: Multiplied by the difference in error.

Each control mode responds differently to the error. The response of each control mode can be adjusted by changing the controller settings. Due to the complexity of the system, we used trial and error to determine the control coefficients. The reasons for not using other methods are discussed in the challenges section.

Control Challenges

1- At primary point, we started to simulating robot in simulink-simscape in MATLAB to tune PID coefficients. Nonetheless, Lack of some information from the environment and circumstance made us to determine them by Error-and-Trial process.
2- Due to fulfil the avoiding buying pricey items, we chose the DC motors without encoder. If we had had the permission to spend more money, we would take the better options.

Hardware Setup and Code

Proteus Simulation

In the Proteus environment, we defined the final wiring and pin configuration for Arduino implementation. After confirming the wiring, all required modules and sensors were connected and embedded on the body. Here you can see Arduino circuit and wiring.

Arduino pins are chosen owing to the ability to convey the PWM pulses for L298N speed control ports. Both DC motors are connected to same driver.

An IR remote receiver is connected for the further job which is out of this project scheme.

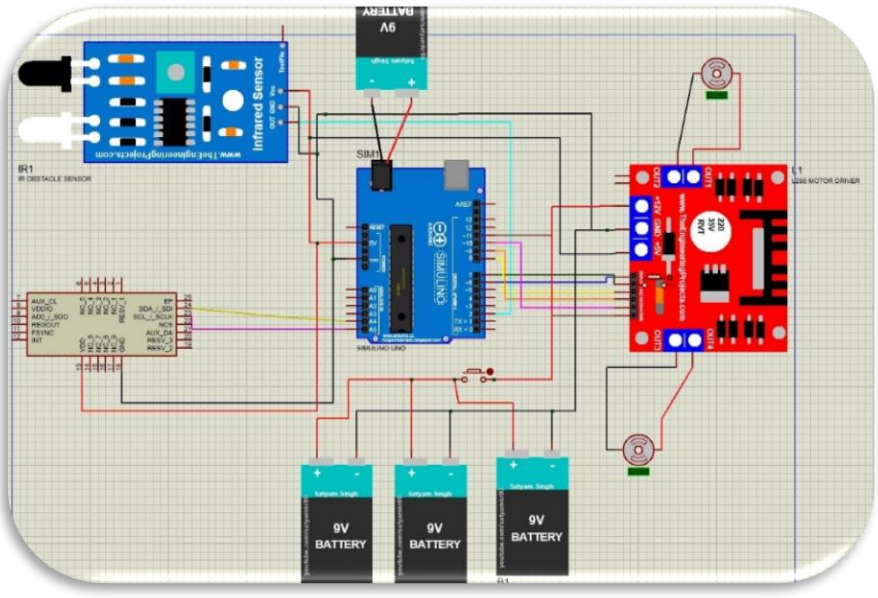


Figure 7 Proteus Environment

Arduino Code

Self balancing Robot PID controller Pseudo Code

```
1. // Call Required Libraries
2. // Determine Variables
3. // Setup function (Executes once)
   a. Initialize Serial and Wire communication
   b. Verify MPU functionality
   c. Set up pin modes

4. // Main Loop
   a. Read MPU data
   b. Extract Roll and apply Band-pass filter
   c. Calculate error-dependent variables
   d. Implement PID control
   e. Send PWM signals to the motors
```

The proper Arduino code provided in the GitHub address Givin in the last section of report.

Construction and Experiment

Initial Design

The initial design, encapsulated in a first chassis, represents the first attempt at constructing a functional self-balancing robot. This prototype was conceptualized with simplicity and rapid development in mind, with an emphasis on achieving basic operational functionality.

The first chassis is characterized by a vertically elongated box-like structure, which serves as the housing for the robot's electronic components, including the Arduino controller, motor driver, and battery packs. The design prioritizes a high center of gravity, which, while simplifying certain aspects of the design process, introduces challenges in maintaining balance, particularly in dynamic scenarios.

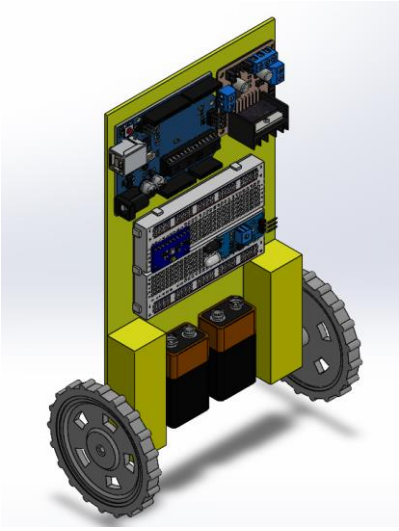


Figure 8 Incomplete design

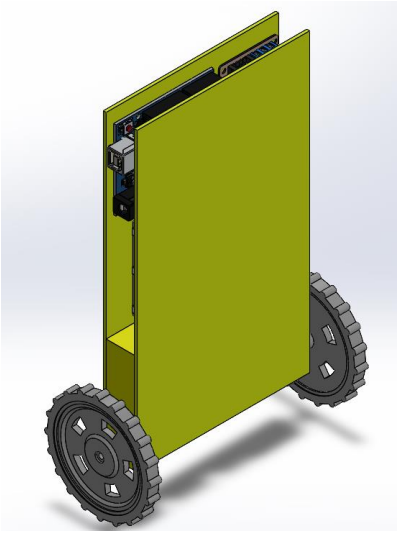


Figure 9 Last design

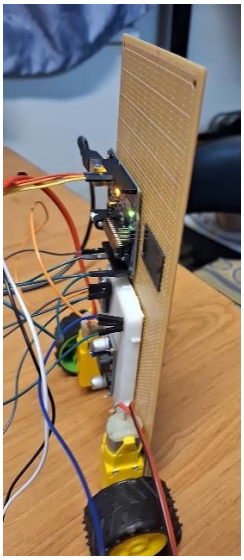


Figure 10 Constructed system

Stability Issues: The elevated center of gravity, coupled with the enclosed design, presents significant challenges in maintaining dynamic balance, especially when the robot encounters unexpected perturbations.

Component Accessibility: While the enclosed chassis offers protection, it also limits the ease of access to critical components. This complicates the process of adjusting and conducting troubleshooting during the iterative design process.

Final Design

The final design, represented by the laser-cut and then assembled chassis, embodies significant advancements over the initial prototype. This design addresses many of the deficiencies identified in the first prototype, particularly in terms of stability, component accessibility, and overall performance.

The chassis presents a more compact and streamlined design, focusing on lowering the center of gravity and improving the robot's stability. The structure is designed to be more open, facilitating easier access to the robot's internal components and improving its overall responsiveness to control inputs.

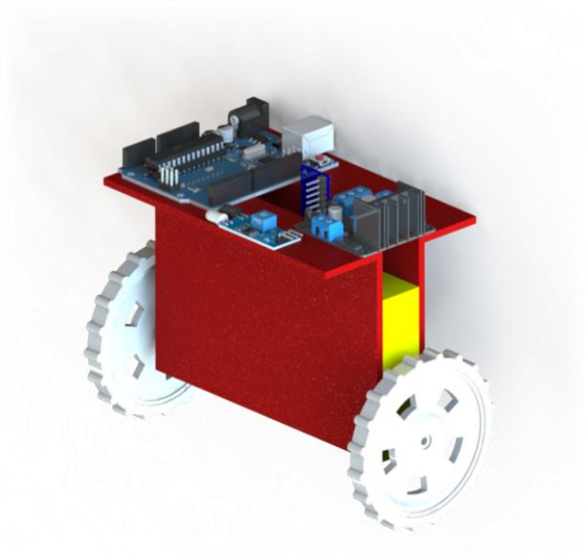


Figure 11 SolidWorks design

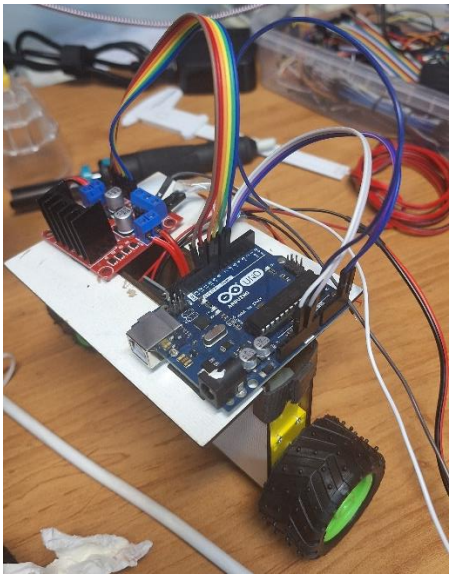


Figure 12 Final system

Stability Enhancements: The most significant improvement in the final design is its enhanced stability. The lowered center of gravity, combined with the refined chassis design, allows the robot to maintain balance more effectively, even when subjected to external disturbances.

Improved Accessibility: By relocating critical components to the top of the chassis, the final design allows for easier access during testing and adjustments. This configuration also facilitates better heat dissipation, which is essential for maintaining the optimal performance of electronic components.

The transition from the initial design to the final prototype demonstrates a clear evolution in both conceptual understanding and practical application of self-balancing robot principles. The initial design, while functional, highlighted several critical areas for improvement, particularly in terms of stability and power management. These lessons informed the development of the final design, which incorporates a more thoughtful approach to component placement, chassis design, and overall system integration.

This comparative analysis underscores the importance of iterative design in the development of self-balancing robots. The initial prototype provided a valuable learning platform, revealing critical design flaws that were systematically addressed in the final prototype. The result is a more stable, accessible, and dynamically capable robot that better meets the demands of self-balancing operation. Future work will build on these findings, with a focus on further optimizing the control systems and exploring advanced materials to enhance the robot's performance and reliability.

Conclusion and References

By completing this project, we gained invaluable knowledge about mechanisms and how to Build and Control A mobile robot.

We confronted with linear control challenges like tuning PID controller coefficient and acquired a pricey wisdom to make us able to overcome more conveniently in the future of our journey in control and Mechatronic world.

A huge shoutout to our course instructor and teaching assistants for their guidance and support.

Supplementary materials, including videos of the working robots, code, and pictures, are uploaded to the following GitHub repository.

https://github.com/AlivGH/Inverted-Pendulum-Segway-Robot

Reference:

Hyung-Jik Lee and S. Jung, "Control of a mobile inverted pendulum robot system," *2008 International Conference on Control, Automation and Systems*, Seoul, Korea (South), 2008, pp. 217-222, doi: 10.1109/ICCAS.2008.4694552