Ministry of Higher Education
Pyramids Higher Institute (PHI) for Engineering and Technology, 6th of October
Electronics and Communications Engineering Department

وزارة التعليم العالي
معهد الأهرامات العالي للهندسة والتكنولوجيا بالسادس من أكتوبر
قسم هندسة الالكترونيات والاتصالات

Robot Modeling and Simulation

Report 1

Level 1 Autonomous Car - Modeling Implementation and Control

Group Name:

Yellow Team

Supervisors:

Dr. Mohamed Hamdy El-Saify

Eng. Ahmed Abdel-Gawad

Objective:

To model, implement and control level 1 - autonomous car depending on range measurements around the car.

Introduction:

In the world of self-driving cars, innovation knows no bounds. Our project uses three ultrasonic sensors and an Arduino microcontroller, along with a PID (Proportional-Integral-Derivative) controller, to provide a small but powerful platform for autonomous automobile development.

The use of ultrasonic sensors provides a comprehensive solution for real-time environmental perception, allowing our self-driving car to navigate its surroundings with precision and agility. By strategically arranging these sensors, our system can detect barriers, estimate distances, and make intelligent decisions, like how humans perceive the road.

The Arduino microcontroller is at the heart of our project, acting as the central nervous system that controls the autonomous driving capabilities. Using Arduino's versatility and accessibility, we enable enthusiasts and developers to investigate the complexities of autonomous car technology, promoting a culture of experimentation and creativity.

Furthermore, the use of a PID controller adds sophistication to our autonomous vehicle's control system, allowing for smooth and responsive navigation in demanding surroundings. By fine-tuning the PID settings, we can improve the car's trajectory, speed, and stability, ensuring efficient and dependable performance in a variety of scenarios.

As we embark on this path, we acknowledge the inherent difficulties and complexities of autonomous vehicle development. However, through collaboration, testing, and persistent iteration, we want to overcome these challenges and pave the way for a future in which autonomous mobility is more than just a concept, but an actual reality.

Join us on this exciting journey as we push the boundaries of innovation and reshape the future of transportation, one ultrasonic pulse at a time. We want to reinvent mobility with self-driving cars, changing how we travel, explore, and interact with our surroundings.

Problem statement and modeling:

In the next analysis, we simplify the robot as a solid object with wheels moving on a flat surface. We describe its movement in three dimensions: two for its position on the surface and one for its vertical orientation. Although the robot has extra moving parts, like wheel axles and joints, we focus only on its main body, ignoring these internal components.

To pinpoint the robot's position on the surface, we establish a connection between a global reference frame of the surface and a local reference frame of the robot. We use coordinates to specify where the robot is on the surface and how it's oriented. This involves choosing a point on the robot as a reference and using it to define the robot's local coordinate system. The position of this reference point on the surface is described by coordinates (x, y), and the angle between the global and local reference frames indicates the robot's orientation (θ). We represent this information as a vector with three elements, clarifying that it's based on the global reference frame.

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

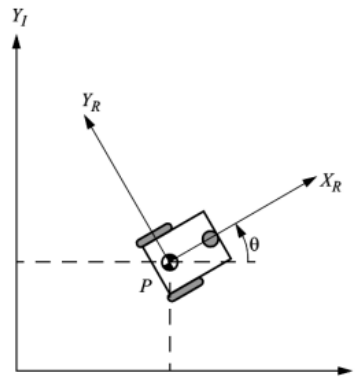


Fig. 5.1 The global reference frame and the robot local reference frame.

To describe robot motion in terms of component motions, it will be necessary to map motion along the axes of the global reference frame to motion along the axes of the robot's local reference frame. Of course, mapping is a function of the current pose of the robot. This mapping is accomplished using the **orthogonal rotation matrix**.

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix can be used to map motion in the global reference frame to motion in terms of the local reference frame {XR, YR}. This operation is denoted by $R\ \theta\ \xi\ I$ because the computation of this operation depends on the value of : $\dot{\xi}_R = R(\frac{\pi}{2})\dot{\xi}_I$

Given some velocity (x , y , θ) in the global reference frame we can compute the components of motion along this robot's local axes XR and YR. In this case, due to the specific angle of the robot, motion along XR is equal to y , and motion along YR is $-x$:

$$\dot{\xi}_R = R(\frac{\pi}{2})\dot{\xi}_I = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{y} \\ -\dot{x} \\ \dot{\theta} \end{bmatrix}$$

In the simplest cases, the mapping described by The Previous equations is sufficient to generate a formula that captures the forward kinematics of the mobile robot: how does the robot move, given its geometry and the speeds of its wheels? More formally, consider the example shown in **Fig. 5.3**. This differential drive robot has two wheels, each with diameter r. Given a point P centered between the two drive wheels, each wheel is a distance l from P. Given r, l, θ, and the spinning speed of each wheel, $\varphi 1$ and $\varphi 2$, a forward kinematic model would predict the robot's overall speed in the global reference frame.

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \dot{\varphi}_1, \dot{\varphi}_2)$$

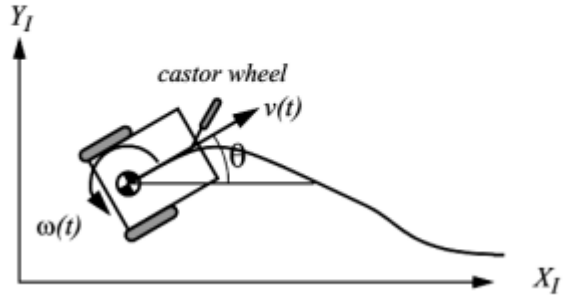


Fig. 5.3 A differential-drive robot in its global reference frame.

From Previous equations we know that we can compute the robot's motion in the global reference frame from motion in its local reference frame: $\xi\ I = R(\theta)\ -1\ \xi\ R$ Therefore, the strategy will be to first compute the contribution of each of the two wheels in the local reference frame, $\xi\ R$. For this example of a differential-drive chassis, this problem is particularly straightforward. Suppose that the robot's local reference frame is aligned such that the robot moves forward along +XR, as shown in Fig. 5.1.

First, consider the contribution of each wheel's spinning speed to the translation speed at P in the direction of +XR. If one wheel spins while the other wheel contributes nothing and is stationary, since P is halfway between the two wheels, it will move instantaneously with half the speed. In a differential-drive robot, these two contributions can simply be added to calculate the $x\ R$ component of $\xi\ R$. Consider, for example, a differential robot in which each wheel spins with equal speed but in opposite directions. The result is a stationary, spinning robot. As expected, $x\ R$ will be zero in this case. The value of $y\ R$ is even simpler to calculate. Neither wheel can contribute to sideways motion in the robot's reference frame, and so $y\ R$ is always zero. Finally, we must compute the rotational component $\theta\ R$ of $\xi\ R$. Once again, the contributions of each wheel can be computed independently and just added. Consider the right wheel (we will call this wheel 1). Forward spin of this wheel results in counterclockwise rotation at point P. Recall that if wheel 1 spins alone, the robot pivots around wheel 2. The rotation velocity at can be computed because the wheel is instantaneously moving along the arc of a circle of radius 2l:

$$\omega_1 = \frac{r\dot{\varphi}_1}{2l}$$

The same calculation applies to the left wheel, with the exception that forward spin results in clockwise rotation at point P:

$$\omega_2 = \frac{-r\dot{\varphi}_2}{2l}$$

Combining these individual formulas yields a kinematic model for the differential-drive example robot:

$$\dot{\xi}_I = R(\theta)^{-1} \begin{bmatrix} \frac{r\dot{\varphi}_1}{2} + \frac{r\dot{\varphi}_2}{2} \\ 0 \\ \frac{r\dot{\varphi}_1}{2l} + \frac{-r\dot{\varphi}_2}{2l} \end{bmatrix}$$

We can now use this kinematic model in an example. However, we must first compute R(θ) −1 . In general, calculating the inverse of a matrix may be challenging. In this case, however, it is easy because it is simply a transform from ξ_R to ξ_I rather than vice versa:

$$R(\theta)^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Suppose that the robot is positioned such that θ = π/2, r = 0.06, and l = 0.067. If the robot engages its wheels unevenly, with speeds φ 1 = 255 and φ 2 = 255, we can compute its velocity in the global reference frame or we can use MATLAB to simulate the kinematics of the robot car in the next steps.

First, we define the parameters of your robot, such as the wheel radius (r), the distance between the wheels and the center (l), and the velocities of the right and left wheels φ 1 and φ 2 .

```
close all
clear
clf
r = 0.06; % wheel radious
l = 0.067; % distance between whel and center
fhiRDot = 255;  %right wheel velocity
fhiLDot = 255;  %Left wheel velocity
```

Then, we set the initial conditions for the position (X0,Y0) and the orientation (θ0) of the robot, as well as the time span for simulation.

```
x0 = 0;
y0 = 0;
theta0 = pi/2;
tspan = [0 10];
```

Then we calculate the linear and angular velocities (x, y, theta) of the robot based on the wheel velocities.

```
xrdot = (r/2) * (fhiRDot + fhiLDot);
yrdot = 0;
omegadot =  (r/(2 * l))* (fhiRDot - fhiLDot);
```

Next, we define the system of ordinary differential equations (ODEs) that describe the robot's motion in terms of its position (x(t),y(t)) and orientation (θ(t)) with respect to time and specify the initial conditions for the ODEs.

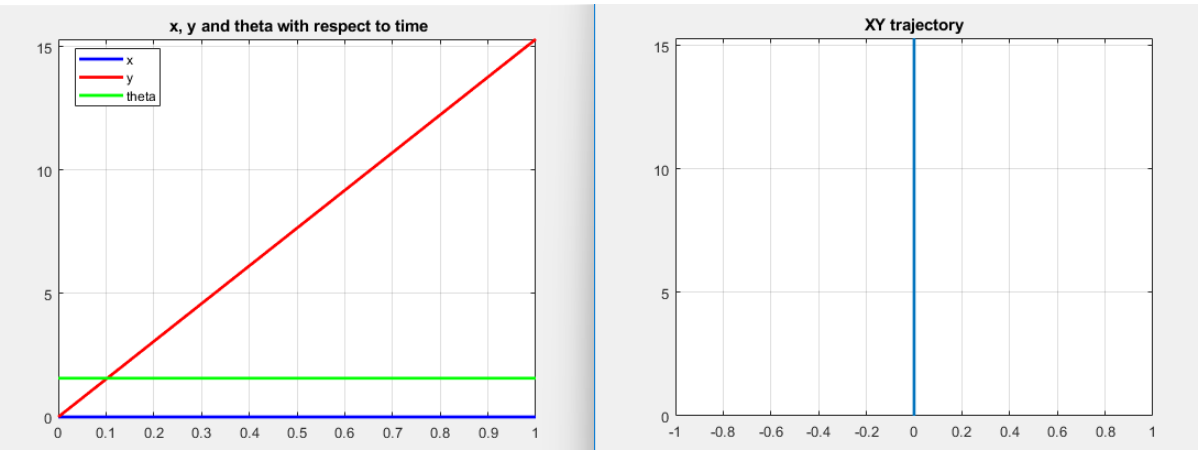
```
xrdot = (r/2) * (fhiRDot + fhiLDot);
yrdot = 0;
omegadot =  (r/(2 * l))* (fhiRDot - fhiLDot);

syms x(t) y(t) theta(t)
ode1 = diff(x) == cos(theta) * xrdot;
ode2 = diff(y) == sin(theta) * xrdot;
ode3 = diff(theta) == omegadot;
odes = [ode1; ode2; ode3];
cond1 = x(0) == x0;
cond2 = y(0) == y0;
cond3 = theta(0) == theta0;
conds = [cond1; cond2; cond3];
```

Using the 'dsolve' function, you solve the system of ODEs with the initial conditions to obtain symbolic expressions for x(t),y(t), and θ(t).

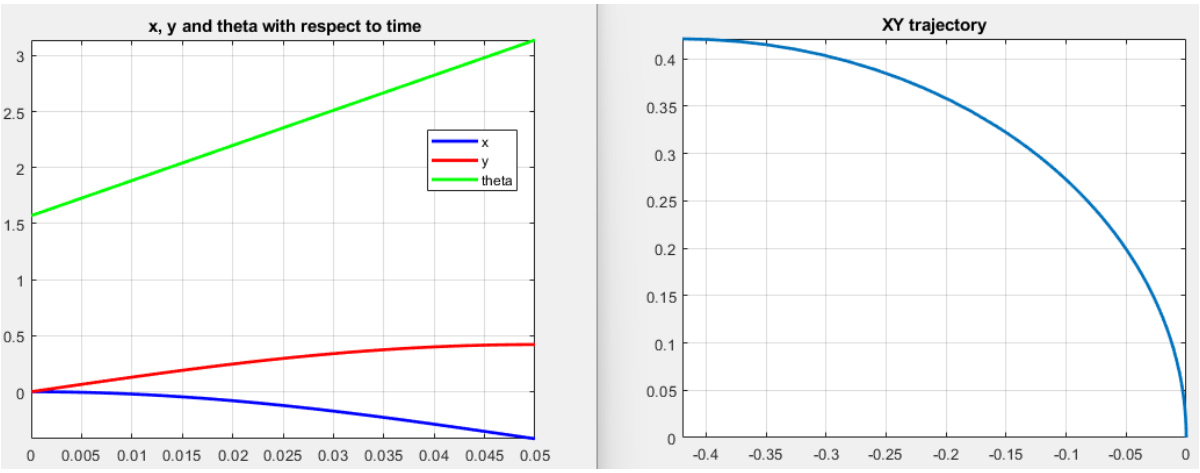
Finally, you plot the functions x(t),y(t), and θ(t) over the specified time span and also plot the trajectory of the robot in the xy-plane.

```
S = dsolve(odes,conds);
disp('x(t)=')
disp(S.x)
disp('y(t)=')
disp(S.y)
disp('theta(t)=')
disp(S.theta)

figure
fplot(S.x, tspan, 'b','linewidth',2),hold on
fplot(S.y,tspan, 'r','linewidth',2)
fplot(S.theta,tspan, 'g','linewidth',2),grid on
legend('x','y','theta','Location','best')
title(' x, y and theta with respect to time')

figure
fplot(S.x,S.y,tspan,'linewidth',2),grid
title('XY trajectory')
```

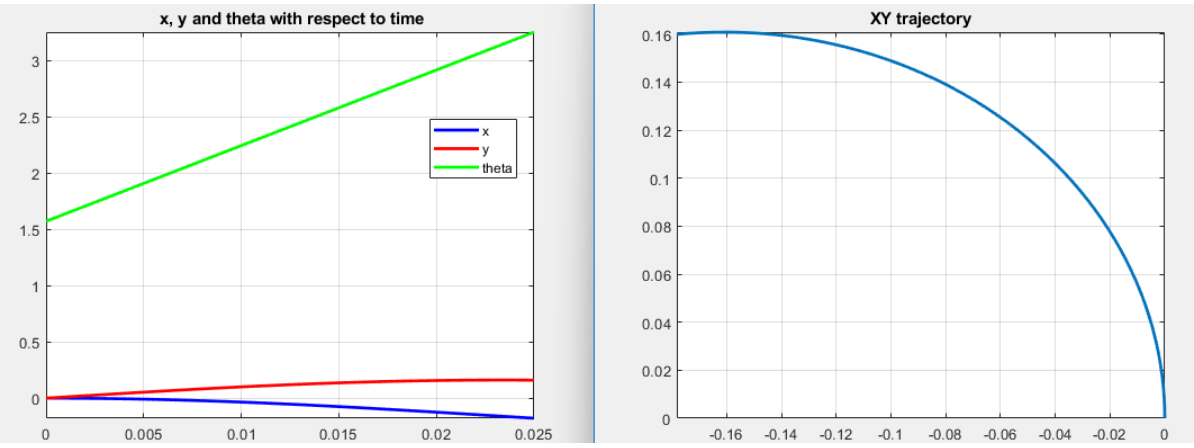
Next Figure is the result of this MATLAB simulation for same wheels speeds :



Next Figure is the result of this MATLAB simulation for variable wheels speeds in the track :



Next Figure is the result of this MATLAB simulation for variable wheels speeds in the turns :



Control law:

- **Sensor Configuration**

  Two ultrasonic sensors are mounted on the autonomous car, one on the left side and one on the right side. These sensors emit ultrasonic waves and measure the time it takes for the waves to return after hitting an obstacle. The distance to nearby objects is calculated based on this time measurement.

- **Error Calculation**

  The distances measured by the left and right ultrasonic sensors serve as reference points. The difference between these distances indicates the error in the car's position relative to the center of the track. This error is used as input for the PID controller.

- **PID Controller Implementation**

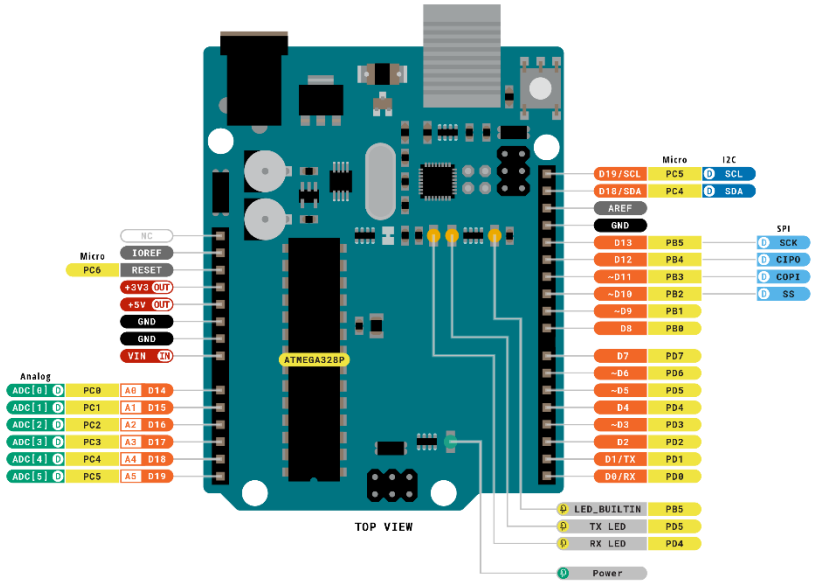
  A PID controller is implemented on the Arduino microcontroller. It takes the error calculated from the ultrasonic sensors as input and computes an output value to minimize the difference between the desired setpoint (center of the track) and the actual position of the car.

- **Steering Adjustment**

  The output from the PID controller is utilized to adjust the steering mechanism of the car. If the car veers towards the right side of the track, the PID controller generates a corrective signal to steer the car towards the left, and vice versa. This continuous adjustment ensures that the car remains centered on the track autonomously.

Hardware and software:

The Arduino Uno is a popular microcontroller board based on the ATmega328P chip. It's the most widely used Arduino board due to its simplicity and versatility.



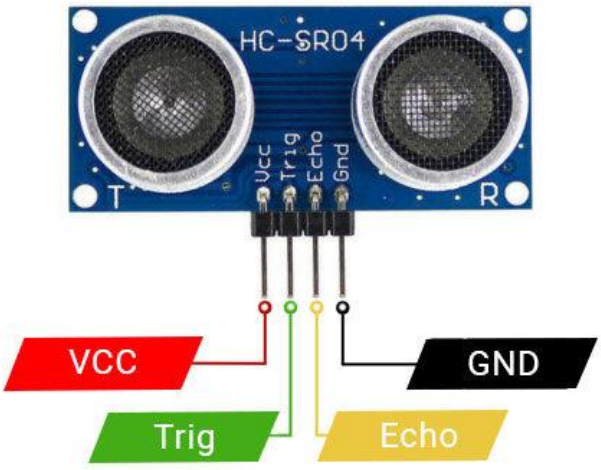
Here's a concise overview:

1. **Microcontroller**: The heart of the Arduino Uno is the ATmega328P microcontroller, which runs at 16 MHz and has 32 KB of flash memory for storing code and 2 KB of RAM for data storage.
2. **Digital and Analog I/O:** The Uno features 14 digital input/output pins, of which 6 can be used as PWM (Pulse Width Modulation) outputs, and 6 analog input pins. These pins allow interfacing with various sensors, actuators, and other electronic components.
3. **USB Interface**: The Uno can be easily connected to a computer via USB, allowing for programming and serial communication. It uses a standard USB Type-B connector.
4. **Power Options**: The board can be powered via USB connection or an external power supply (7-12V DC). It also has a built-in voltage regulator, allowing it to be powered directly from a 5V source.
5. **Integrated Development Environment (IDE)**: Arduino Uno is programmed using the Arduino IDE, a user-friendly software environment that simplifies coding and uploading sketches (programs) to the board.
6. **Open-Source Platform**: Arduino Uno is part of the Arduino ecosystem, which is open-source. This means the hardware designs, software, and documentation are freely available, encouraging collaboration and innovation among the community.

Overall, the Arduino Uno is a versatile and beginner-friendly platform for electronics projects, ranging from simple blinking LED experiments to complex automation and robotics projects. Its ease of use, extensive community support, and vast array of compatible sensors and shields make it a favorite among hobbyists, educators, and professionals alike.

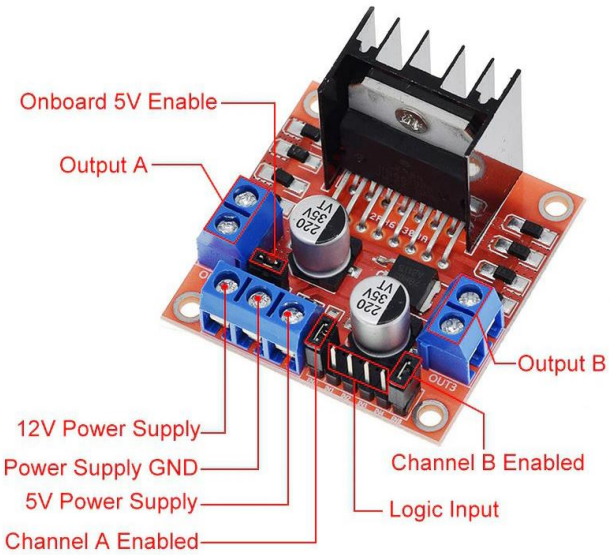
The 18650 lithium-ion battery is a small, cylindrical rechargeable battery known for its high energy density and reliability. It measures 18mm in diameter and 65mm in length, hence the name "18650." These batteries typically provide a nominal voltage of 3.7 volts and are widely used in electronic devices such as laptops, flashlights, and electric vehicles due to their compact size and long-lasting power.

Ultrasonic sensors utilize high-frequency sound waves (above 20,000 Hz) to detect objects and measure distances. They typically consist of a transmitter that emits ultrasonic waves and a receiver that detects the waves after they bounce off objects. By measuring the time, it takes for the waves to return, the sensor calculates the distance to the object. Ultrasonic sensors find applications in various fields such as robotics, automotive, industrial automation, and security systems, offering non-contact, reliable, and accurate object detection capabilities.



DC geared motors are a type of electric motor coupled with a gearbox, designed to provide high torque at low speeds. They operate on direct current (DC) power and are widely used in robotics, automation, and various electromechanical systems. The gearbox reduces the motor's speed while increasing its torque output, making it suitable for applications requiring precise control and high starting torque. DC geared motors are available in different configurations, including brushed and brushless varieties, offering options for different performance requirements and operating conditions. They are popular for their simplicity, reliability, and versatility in a wide range of applications.

The L298 is a popular integrated circuit (IC) used as a motor driver, especially for controlling DC motors and stepper motors. It's designed to handle high-current applications and provides bidirectional control for two motors simultaneously. The L298 IC typically comes in a multiwatt package with built-in H-bridges, allowing it to control the direction and speed of motors by adjusting the voltage and polarity of the applied signals.



Key features of the **L298 motor driver** include:

1. **Dual H-Bridge Configuration**: The L298 IC consists of two H-bridge circuits, allowing it to control two motors independently. This configuration enables both forward and reverse motion control of the motors.
2. **High Current Capability**: The L298 can handle high currents, making it suitable for driving motors with higher power requirements, such as those used in robotics, RC vehicles, and industrial automation.
3. **Built-in Diodes**: The IC includes built-in flyback diodes, providing protection against voltage spikes generated by the motor when it is turned off, thereby increasing reliability, and preventing damage to the circuit.
4. **Logic Inputs**: The L298 accepts standard logic-level inputs (TTL or CMOS), allowing easy interfacing with microcontrollers, Arduino boards, or other control systems for motor control applications.

Overall, the L298 motor driver IC is a versatile and widely used component for controlling DC motors and stepper motors in various electronic projects and applications. Its robust design, high current handling capability, and bidirectional control make it a popular choice among hobbyists and professionals alike.

Flow Chart :

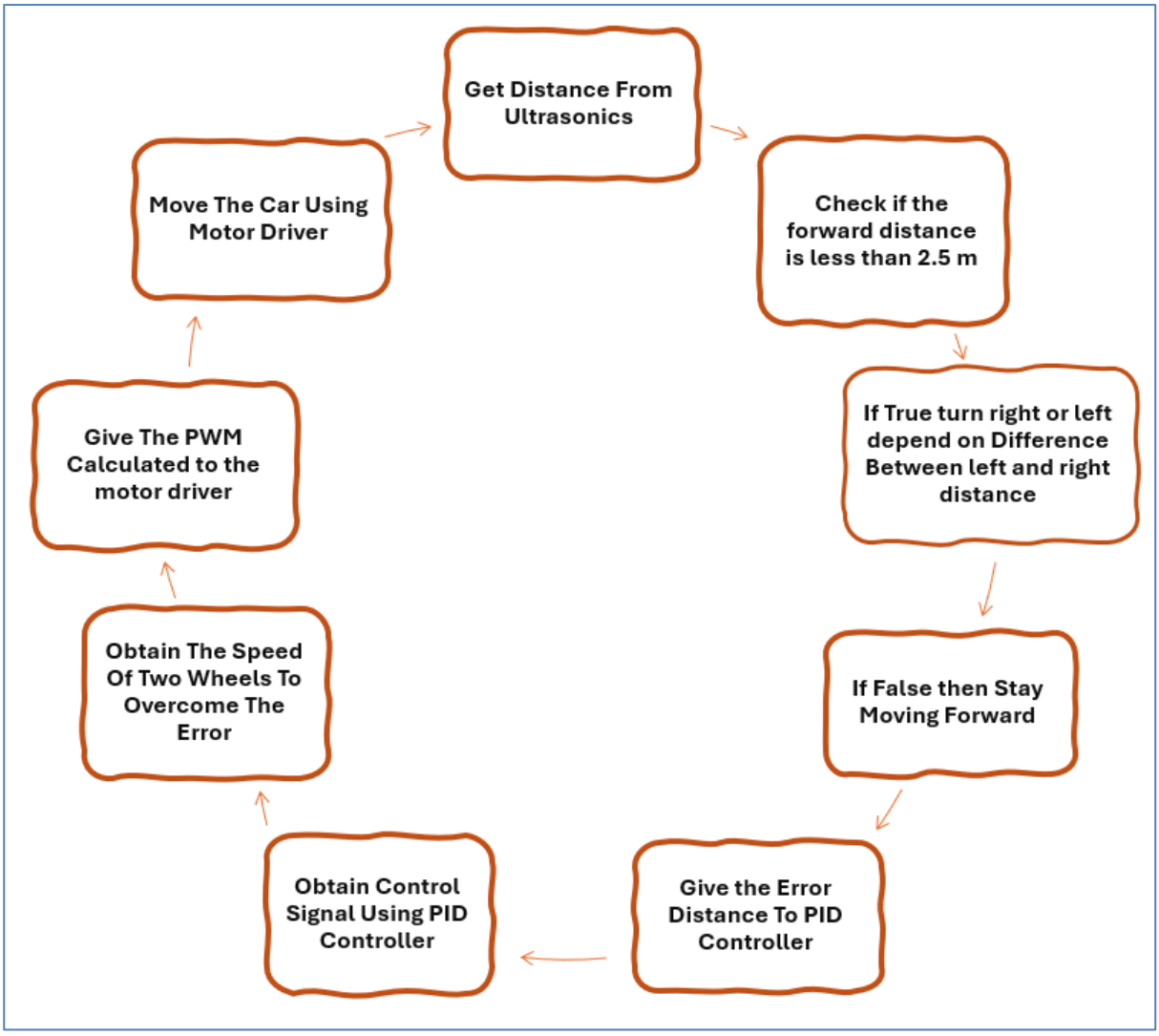


Figure 1 Flow Chart Of The Project

Results and analysis:

We thoroughly tested our autonomous vehicle on a twisting route with a variety of obstacles. The testing determined how effectively the car handled steep turns, small routes, and numerous obstructions.

Throughout the tests, the automobile effectively plotted its route, spotted obstructions, and avoided crashes. It used modern sensors, such as ultrasonic to properly analyze its surroundings.

The car's control system, which was powered by advanced algorithms and a PID controller, provided smooth navigation around the track. It could accurately alter its steering to remain on course.

The testing demonstrated that the automobile could respond to changing track conditions, such as tight curves or unexpected obstructions, by modifying its speed and direction.

Overall, our tests demonstrated the effectiveness and reliability of our autonomous vehicle's technologies. They demonstrated the feasibility of deploying such vehicles in real-world scenarios, where they can manage complicated environments and unpredictable conditions with ease.

Figure 2 Turn 1

Figure 3 Turn 2

Figure 4 Turn 3

Figure 5 Car In Turns

Figure 6 Track 1

Figure 7 Track 2

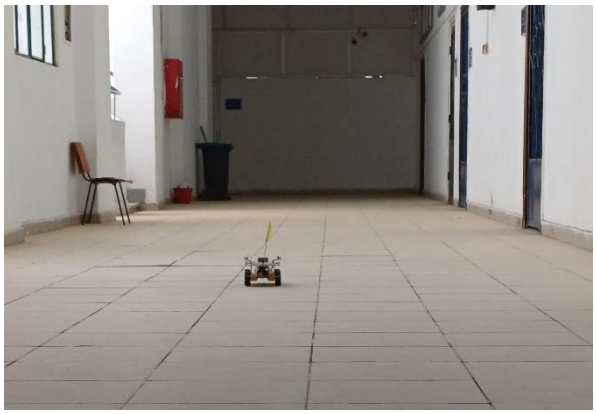


Figure 8 Car In Tracks

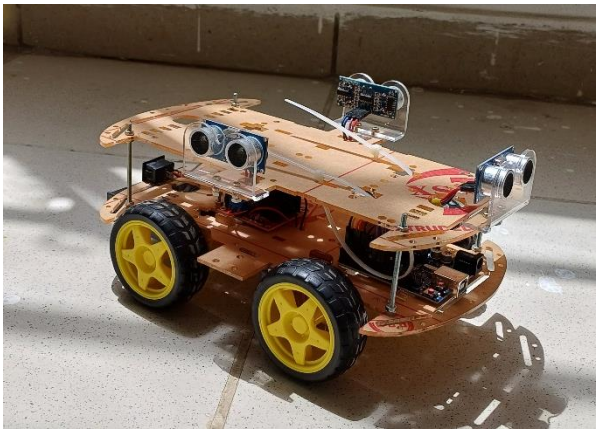


Figure 9 Car Photo

Conclusion:

In summary, the goal of our project was to create a differential robot car that could use three ultrasonic sensors combined with a PID controller to navigate turns and center itself automatically on a track.

In addition, we developed a MATLAB model to mimic how the car would move in various wheel speed scenarios. With careful planning and execution, we were able to accomplish our goals.

The differential robot car proved the efficacy of our control algorithm and sensor integration by demonstrating strong performance in tracking within the track and maneuvering through turns.

The MATLAB simulation demonstrated how wheel speed differentials affect the car's trajectory and offered insightful information about the dynamics of differential drive systems.