A

LAB REPORT

submitted in partial fulfillment of the requirement of course for

EE 555

submitted

by

Yogesh Kumar Chauhan

Roll No.234102508

&

Isha Rani Das

Roll No.234102504

Under the instruction and guidance of

Dr Parijat Bhowmick

Department of Electronics and Electrical Engineering
Indian Institute of Technology, Guwahati
May 2024

Contents

1 Introduction:

This report focuses on Two-wheeled differential drive robots (specifically using the TurtleBot3 platform developed by ROBOTIS), which represent a class of versatile and agile robotic platforms characterized by their simplicity, maneuverability, and wide-ranging applications. These robots leverage the Robot Operating System (ROS), a powerful framework for building robotic software, to enable seamless integration of sensors, actuators, and algorithms for perception, control, and navigation.

Our main focus here is on robots consisting of two motorized wheels typically mounted on a common axis, with each wheel independently driven and controlled, and a passive caster wheel. The distinctive differential drive mechanism enables differential wheel speeds, allowing the robot to achieve various motion behaviors such as forward/backward translation and rotation, while the caster wheel provides stability.

These mobile robots have found extensive use in diverse fields such as robotics research, industrial automation, logistics, surveillance, exploration, SLAM, and education. Their compact size, lightweight design, and movement capabilities make them ideal for navigating confined spaces, rough terrain, and dynamic environments with ease. Moreover, their affordability and accessibility have made them popular platforms for educational projects, experimentation, and prototyping in robotics laboratories and academic institutions worldwide.

Despite their simple mechanical structure, differential drive robots pose unique challenges in terms of control, navigation, and motion planning, particularly when tasked with executing complex trajectories or interacting with uncertain environments. Achieving precise trajectory control, which involves guiding the robot along specified paths with high accuracy and stability while attenuating disturbances, is essential for enabling effective robot operation in real-world scenarios.

In this report, we will explore the trajectory control of a differential drive two-wheeled mobile robot using the TurtleBot3 platform and ROS, focusing on the design, simulation, analysis, and implementation of control strategies to achieve optimal motion performance and trajectory tracking accuracy. The simulations were conducted using Simulink, and the robot's behavior was tested in Gazebo, a ROS-based robot simulation environment, before actual experimental validation. By investigating the underlying principles of differential drive kinematic, control theory, and motion dynamic within the ROS ecosystem, we aim to uncover the intricacies of trajectory control and provide insights into the key considerations, challenges, and advancements in this critical aspect of mobile robotics. The report will include experimental results that validate the fundamental concepts and practical techniques employed in trajectory control, paving the way for the development of robust, agile, and intelligent robotic systems capable of navigating complex environments and performing sophisticated tasks autonomously.

2 Turtlebot Illustration

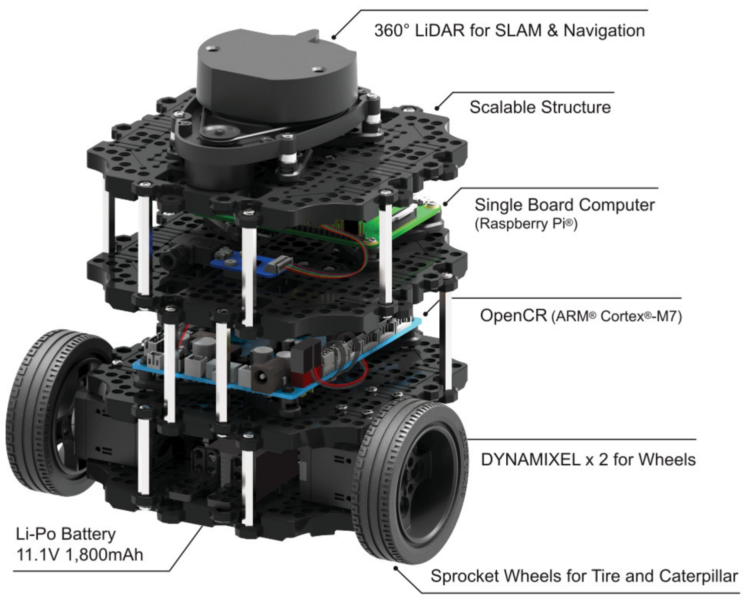

360° LiDAR for SLAM & Navigation
Scalable Structure
Single Board Computer (Raspberry Pi®)
OpenCR (ARM® Cortex®-M7)
DYNAMIXEL x 2 for Wheels
Li-Po Battery 11.1V 1,800mAh
Sprocket Wheels for Tire and Caterpillar

Figure 1: Turtlebot Illustration

3 Brief Theory

The differential drive two-wheeled mobile robot (WMR) allows independent control of each wheel. In a differential drive system, if both wheels turn at the same angular velocity and direction, the robot moves straight forward or backward. If the wheels turn at different angular velocities, the robot turns. By adjusting the angular velocities of the left and right wheels, you can control the turning behavior of the robot.

Important component of Turtlebot Raspberry Pi

The TurtleBot3 is powered by a Raspberry Pi, which serves as the main processing unit. Ubuntu, a popular Linux distribution, is installed on the Raspberry Pi, providing a stable and flexible operating system environment. Ubuntu enables seamless integration with ROS, allowing for the development and execution of robotic applications.

OpenCR ARM Cortex-M7

The OpenCR (Open-source Control Module for ROS) is a microcontroller board based on the ARM Cortex-M7 architecture. It is specifically designed for ROS-based robotic applications and serves as the low-level controller for the TurtleBot3 platform. The OpenCR board handles real-time control tasks, such as motor control, sensor interfacing, and communication with external devices.

Dynamixel Motors

The TurtleBot3 utilizes Dynamixel motors for driving its wheels. These high-performance servo motors offer precision, durability, and programmability, allowing for precise control of the robot's movement. Dynamixel motors enable the TurtleBot3 to achieve differential drive motion, facilitating agile navigation and maneuverability.

Basic Specifications of TurtleBot3:

1. Dimensions: 138mm x 178mm x 192mm (W x L x H)

2. Weight: 1.28 kg (including battery)
3. Payload Capacity: 1.0 kg
4. Maximum Speed: 0.22 m/s
5. Communication: USB, Wi-Fi, Bluetooth
6. Sensors: IMU, LiDAR (optional)

Utilizing the \cmdvel node, Linear and Angular Velocity commands can be published to the Turtle-Bot3, either via Simulink or by scripting specific commands. Additionally, the \odom node can be employed to receive feedback from the robot, facilitating closed-loop control and navigation tasks.

Moreover, the ROS (Robot Operating System) toolbox in Simulink offers a powerful platform for designing and simulating robotic control systems. Simulink provides blocks to interface with ROS nodes directly within the Simulink environment, enabling seamless integration of ROS functionalities into the control system design process. This integration allows for the development, testing, and deployment of complex robotic control strategies using Simulink's intuitive graphical interface.

4 System modelling

Kinematic Equations:

$$\dot{x} = \frac{R}{2}(\dot{\theta}_R + \dot{\theta}_L)\cos\phi \tag{1}$$

$$\dot{y} = \frac{R}{2}(\dot{\theta}_R + \dot{\theta}_L)\sin\phi \tag{2}$$

$$\dot{\phi} = \frac{R}{2a}(\dot{\theta}_R - \dot{\theta}_L) \tag{3}$$

Joint Space:

Joint variables:	θ_L, θ_R
Description:	Space of all possible joint configurations of the robot, where each joint represents a degree of freedom

Task Space:

Task variables:	x, y, ϕ
Description:	Space specifying the robot's position and orientation in the environment.

where:

$$\begin{aligned}
\dot{x} &= \text{linear velocity along the } x \text{ axis} \\
\dot{y} &= \text{linear velocity along the } y \text{ axis} \\
\dot{\phi} &= \text{angular velocity} \\
\dot{\theta}_R &= \text{angular velocity of the right wheel} \\
\dot{\theta}_L &= \text{angular velocity of the left wheel} \\
R &= \text{radius of the wheels} \\
a &= \text{half of the distance between the wheels (wheelbase)}
\end{aligned}$$

The linear and anguar velocities can be resolved into their component as below:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Dynamic Equations:

$$M_R \ddot{x}_R = F_{\text{ext}_R} - F_{\text{friction}_R} \tag{4}$$

$$M_L \ddot{x}_L = F_{\text{ext}_L} - F_{\text{friction}_L} \tag{5}$$

$$M\ddot{x} = [(F_{\text{ext}_R} - F_{\text{friction}_R}) + (F_{\text{ext}_L} - F_{\text{friction}_L})]/2 \tag{6}$$

$$I\ddot{\theta} = [(F_{\text{ext}_R} - F_{\text{friction}_R}) - (F_{\text{ext}_L} - F_{\text{friction}_L})]\frac{R}{2} \tag{7}$$

where:

$$\begin{aligned}
M_R &= \text{point mass considered of the right wheel} \\
M_L &= \text{point mass considered of the left wheel} \\
I &= \text{moment of inertia of the robot} \\
\ddot{x}_R &= \text{acceleration of the right wheel} \\
\ddot{x}L &= \text{acceleration of the left wheel} \\
F\text{ext}R &= \text{external force applied on the right wheel} \\
F\text{ext}L &= \text{external force applied on the left wheel} \\
F\text{friction}R &= \text{frictional force on the right wheel} \\
F\text{friction}_L &= \text{frictional force on the left wheel}
\end{aligned}$$

But for the scope of this report we will be neglecting frictional forces.

$$v = \frac{1}{mr}(\tau_r + \tau_l) = \frac{1}{m}\tau_a \tag{8}$$

$$\omega = \frac{2a}{I_r}(\tau_r - \tau_l) = \frac{1}{I}\tau_b \tag{9}$$

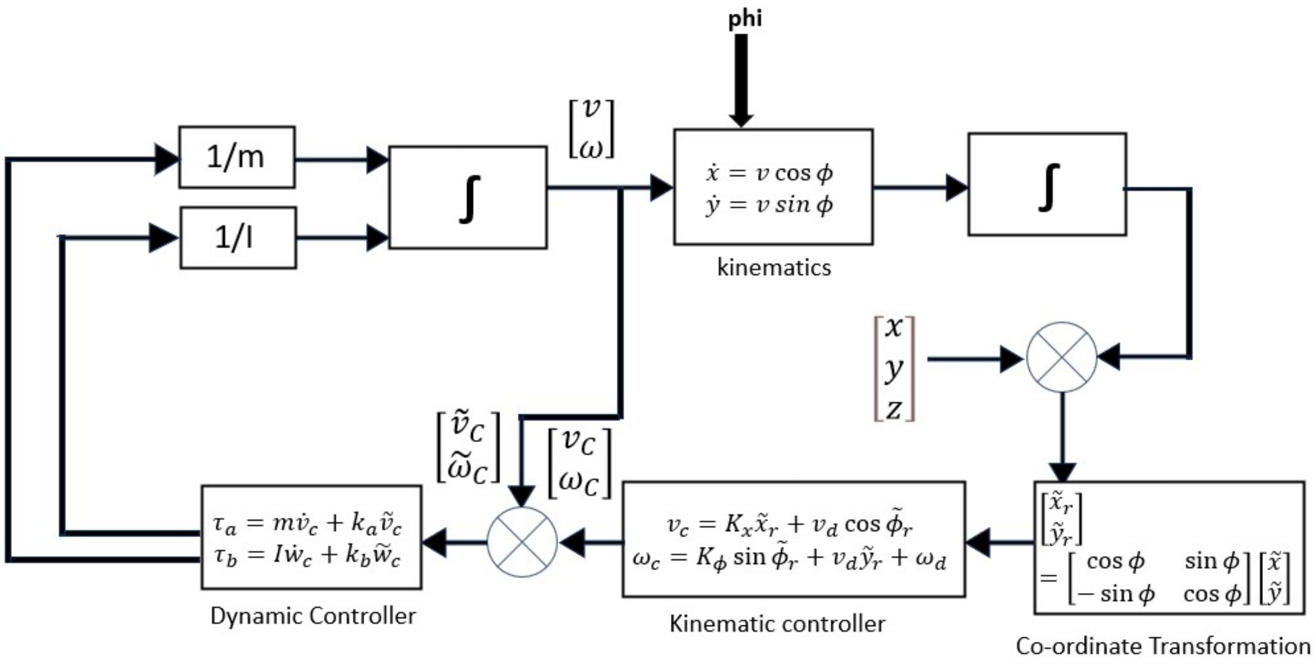
$$\dot{x} = v\cos\phi \tag{10}$$

$$\dot{y} = v\sin\phi \tag{11}$$

5 Controller Design

5.1 Lyapunov controller

For the trajectory control of Two-wheeled differential drive mobile robot (here Turtlebot3) for which we designed a Lyapunov-based controller since we have found that Lyapunov based controller is easy to implement and mathematically guaranties the asymptotic stability so more useful in nonlinear control.



phi
1/m
1/I
$\int$
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
$\dot{x} = v\cos\phi$
$\dot{y} = v\sin\phi$
kinematics
$\int$
$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$
$\begin{bmatrix} \tilde{v}_C \\ \tilde{\omega}_C \end{bmatrix}$
$\begin{bmatrix} v_C \\ \omega_C \end{bmatrix}$
$\tau_a = m\dot{v}_c + k_a\tilde{v}_c$
$\tau_b = I\dot{w}_c + k_b\tilde{w}_c$
Dynamic Controller
$v_c = K_x\tilde{x}_r + v_d\cos\tilde{\phi}_r$
$\omega_c = K_\phi\sin\tilde{\phi}_r + v_d\tilde{y}_r + \omega_d$
Kinematic controller
$\begin{bmatrix} \tilde{x}_r \\ \tilde{y}_r \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}$
Co-ordinate Transformation

Figure 2: Complete Trajectory Tracking Feedback control system based on Lyapunov Technique

5.1.1 Kinematic Tracking Control

For Kinematic Tracking Control we have chosen a energy like function as Lyapunov function candidate

$$V = \frac{1}{2}(x^2 + y^2) + (1 - \cos\phi_r),$$

which is a positive definite function.

The errors $\tilde{x} = (x_d - x)$, $\tilde{y} = (y_d - y)$, and $\tilde{\phi} = (\phi_d - \phi)$, expressed in the wheeled mobile robots' (WMR's) local (moving) coordinate frame $Q_{x_r y_r}$, are given by:

$$\begin{pmatrix} \tilde{x}_r \\ \tilde{y}_r \end{pmatrix} = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

and

$$\tilde{\phi}_r = \tilde{\phi}$$

Differentiating the lyapunov function candidate with respect to time we get:

$$\dot{V} = \dot{\tilde{x}}_r \tilde{x}_r + \dot{\tilde{y}}_r \tilde{y}_r + \dot{\tilde{\phi}}_r \sin\tilde{\phi}_r \tag{12}$$

$$\dot{V} = (-v + v_d \cos\tilde{\phi}_r)\tilde{x}_r + (-\omega + v_d \tilde{y}_r + \omega_d)\sin\tilde{\phi}_r \tag{13}$$

To make $\dot{V} \leq 0$, the control inputs v and ω are selected as:

$$v = v_c = K_x \tilde{x}_r + v_d \cos\tilde{\phi}_r \tag{14}$$

$$\omega = \omega_c = K_j \sin\tilde{\phi}_r + v_d \tilde{y}_r + \omega_d \tag{15}$$

which makes the $\dot{V}$ negative semi-definite:

$$\dot{V} = -(K_x \tilde{x}_r{}^2 + K_j \sin^2\tilde{\phi}_r) \tag{16}$$

Clearly, for $K_x > 0$ and $K_j > 0$, we have $V_{\dot{p}_r} \leq 0$. Thus, the controller (14) and (15) guarantees total asymptotic tracking to the desired trajectory.

5.1.2 Dynamic Tracking Control

After selecting v and ω, we select the control inputs (torques) τ_a and τ_b as:

$$\tau_a = m\dot{v}_c + K_a \tilde{v}_c \tag{17}$$

$$\tau_b = I\dot{\omega}_c + K_b \tilde{\omega}_c \tag{18}$$

where:

$$\tilde{v}_c = v_c - v \tag{19}$$

$$\tilde{\omega}_c = \omega_c - \omega \tag{20}$$

Subtituting (17) and (18) into (8) and (9) we get the velocities' error equations:

$$\dot{v}_c + \left(\frac{K_a}{m}\right) v_c = 0 \tag{21}$$

$$\dot{\omega}_c + \left(\frac{K_b}{I}\right) \omega_c = 0 \tag{22}$$

which for $K_a > 0$ and $K_b > 0$ are stable first order dynamic and v_c, ω_c converge to zero asymptotically.

5.2 PD controller

For the trajectory control of the two-wheeled differential drive mobile robot (Turtlebot3), we also implemented a Proportional-Derivative (PD) controller. Unlike the Lyapunov-based controller, the PD controller provides a simpler approach to trajectory following, relying on proportional and derivative feedback to adjust the robot's velocity and orientation error.

The block diagram of the Turtlebot kinematic model for trajectory-following using the PD-based controller is illustrated in Figure **??**. This controller calculates the control inputs based on the error between the desired trajectory and the current robot position and orientation.

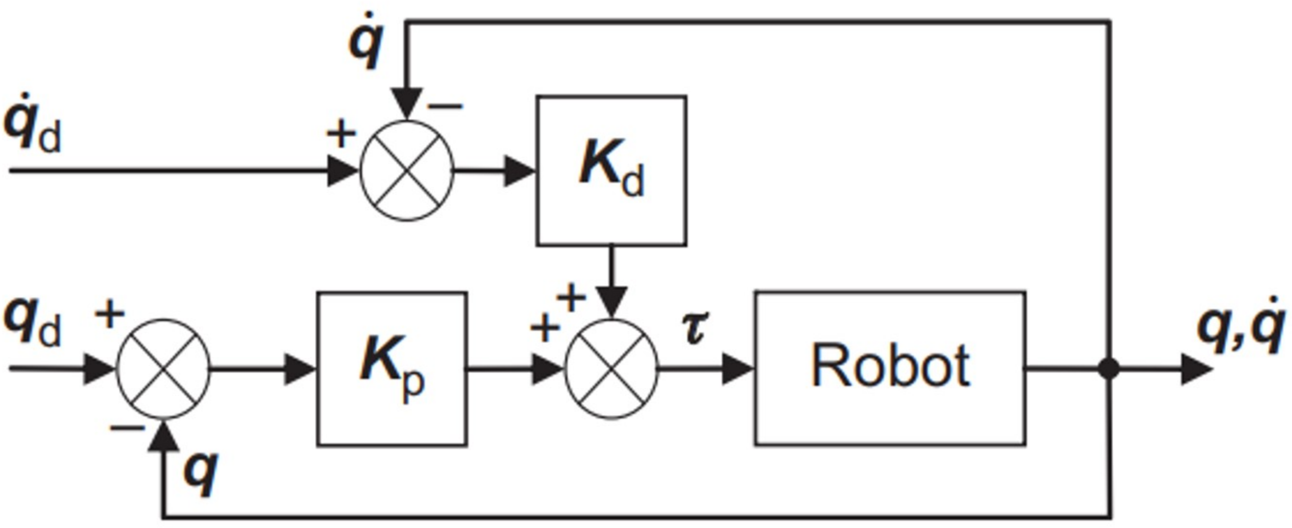


Figure 3: Block Diagram Turtlebot Kinematic model for trajectory-following using PD based controller

6 Simulation model and results

a. Open loop Kinematic model

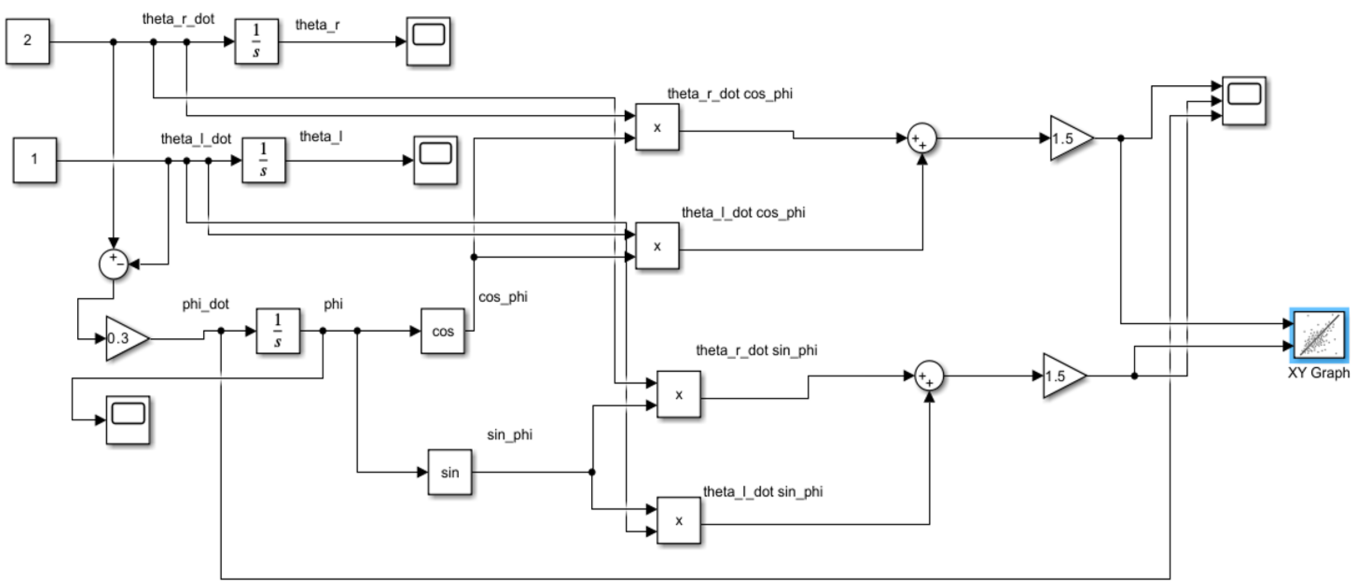


Figure 4: Turtlebot Open loop Kinematic Model

XY graph of Open loop Kinematic model (for constant $\dot{\theta}_r$ and $\dot{\theta}_l$)

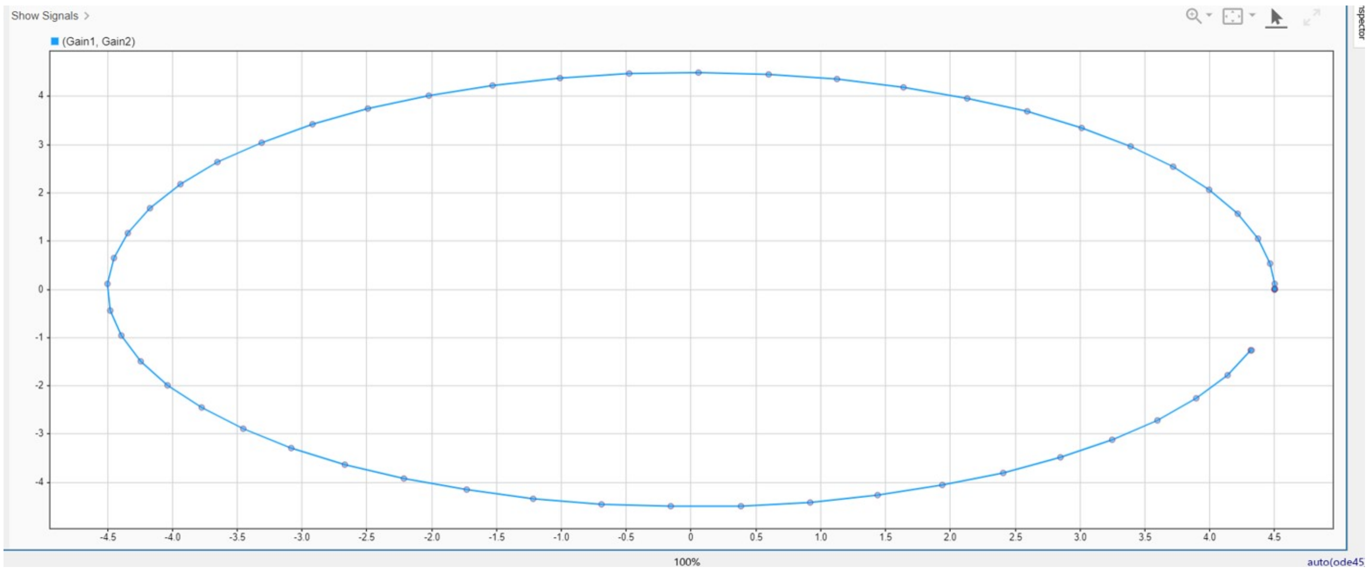


Figure 5: XY graph of Open loop Kinematic model

b. Open loop Dynamic model

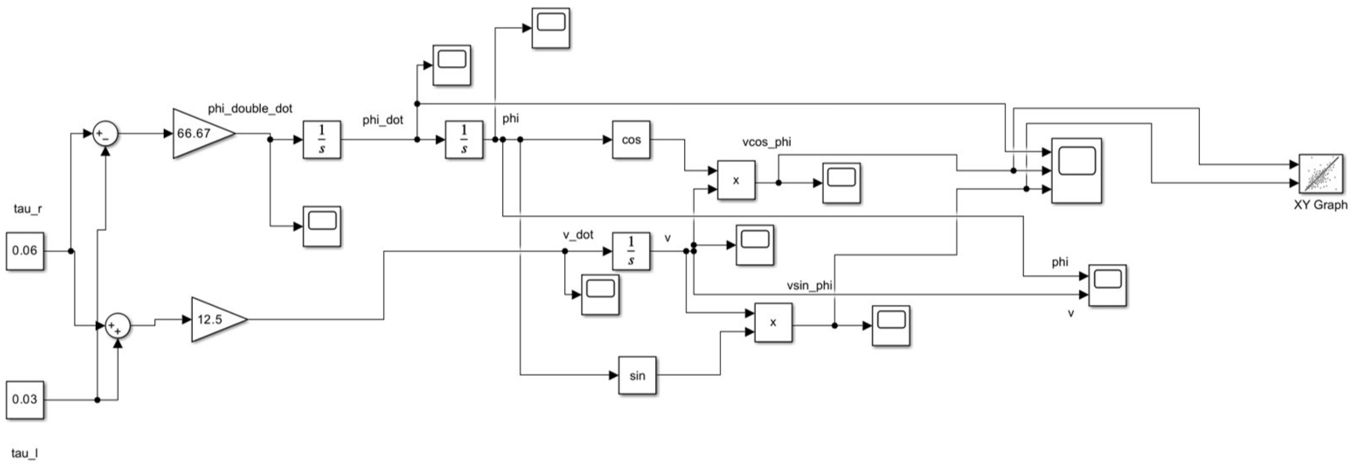


Figure 6: Turtlebot Open loop Dynamic Model

XY graph of Open loop Dynamic model for contant τ_r and τ_l (spiralling out of origin for the shown value of τ_r and τ_l)

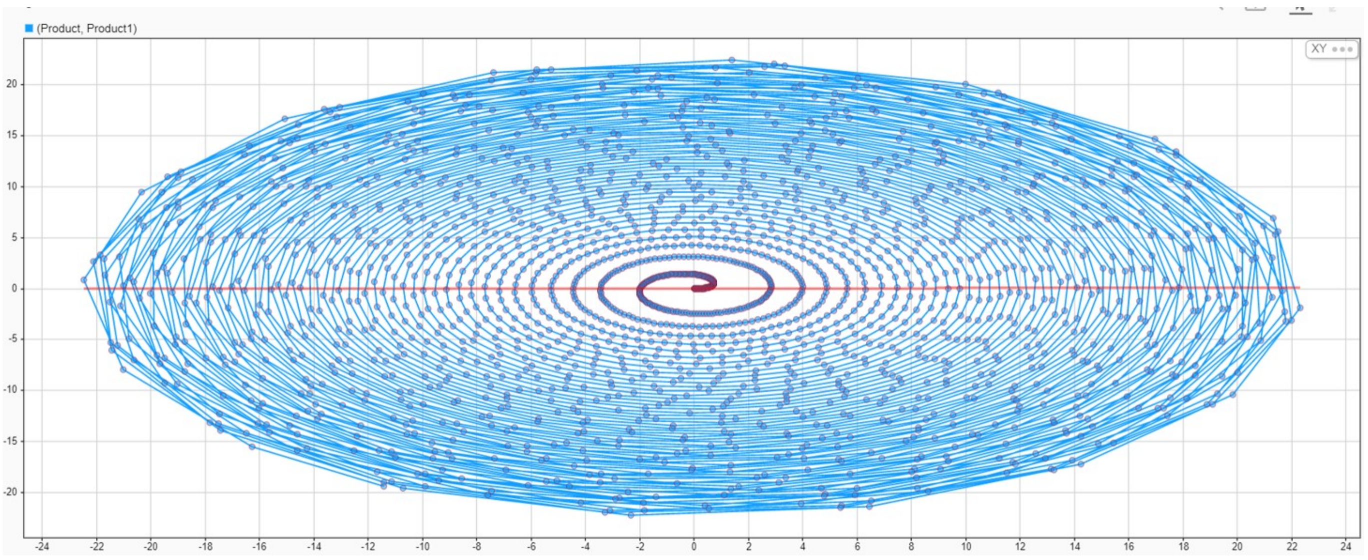


Figure 7: XY graph of Open loop Dynamic model

c. Kinematic model of differential drive Two wheeled mobile robot using Lyapunov controller for trajectory control:

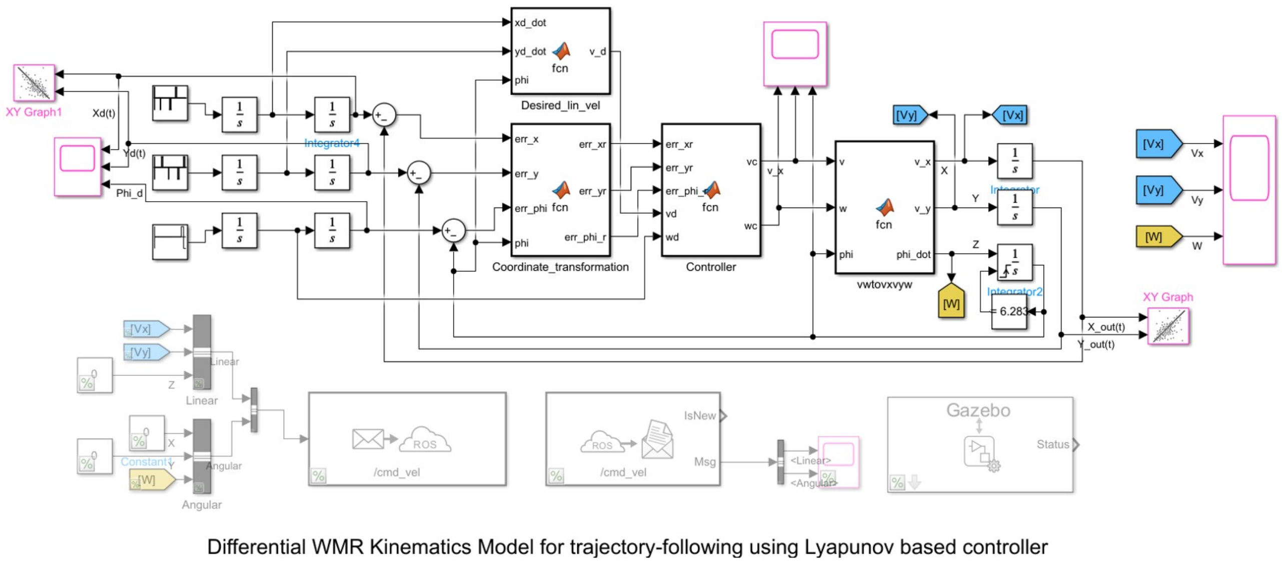


Figure 8: Turtlebot kinematic model for trajectory-following using Lyapunov based controller

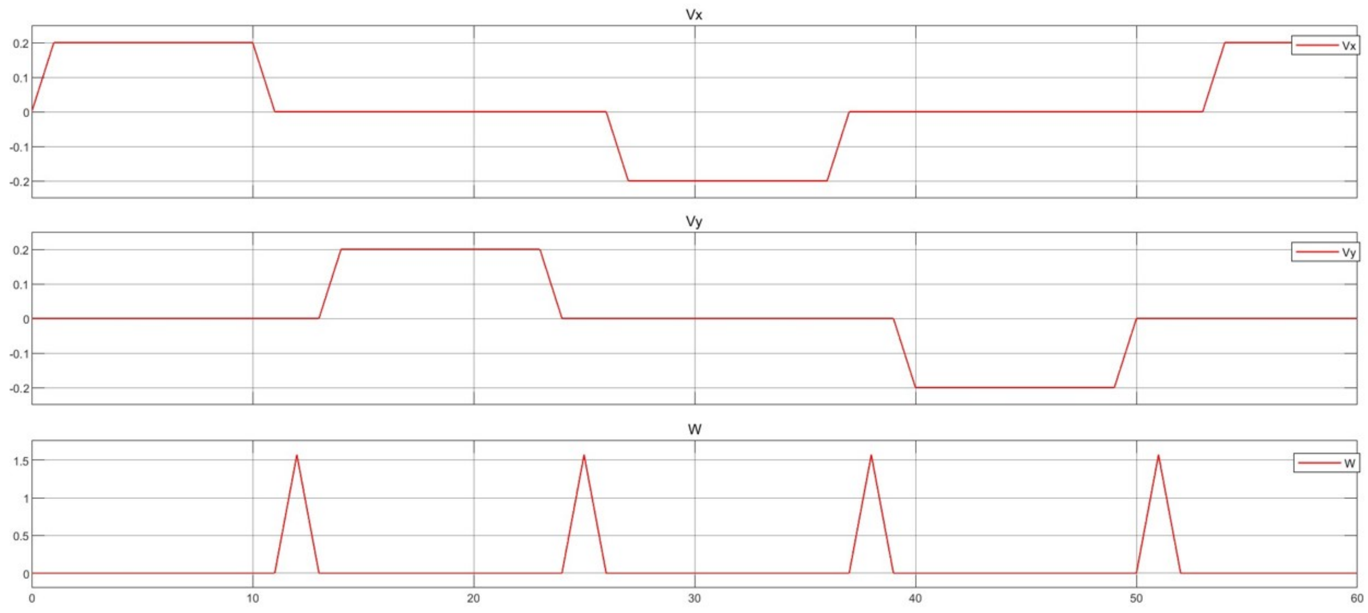


Figure 9: Response of kinematic model for trajectory-following using Lyapunov based controller

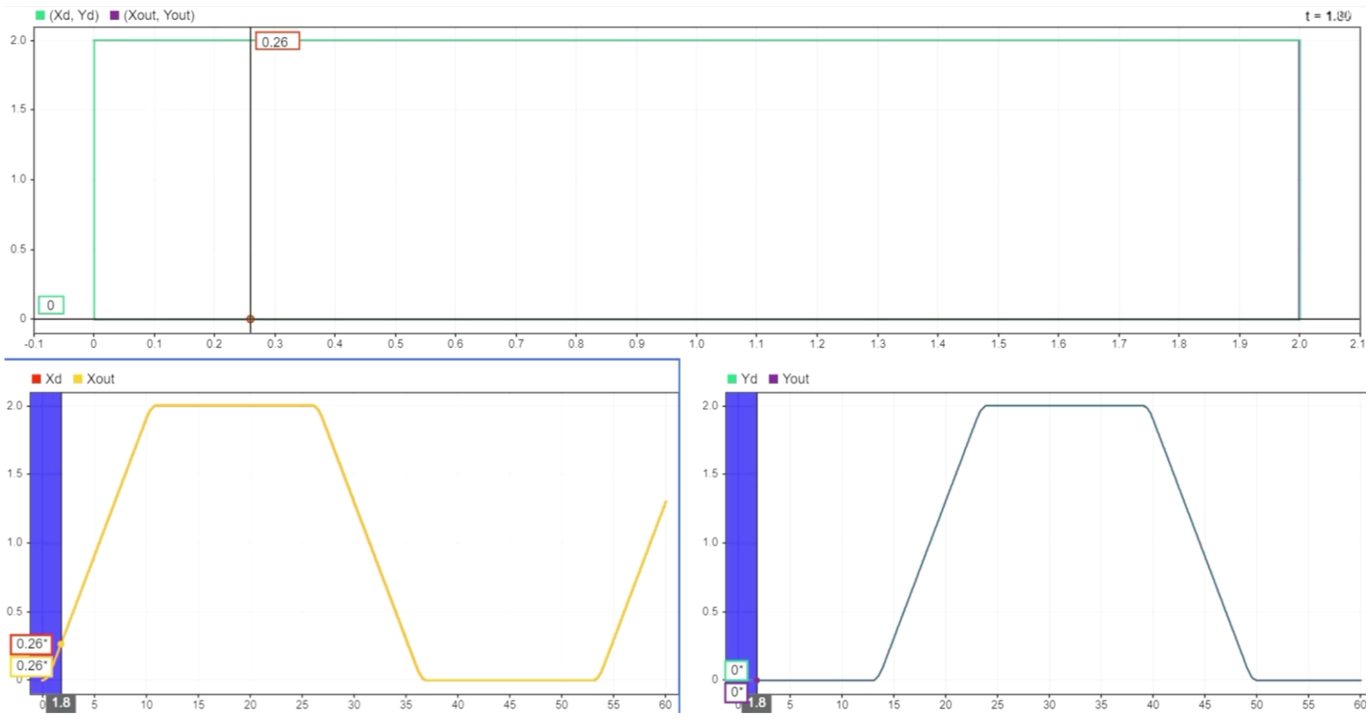
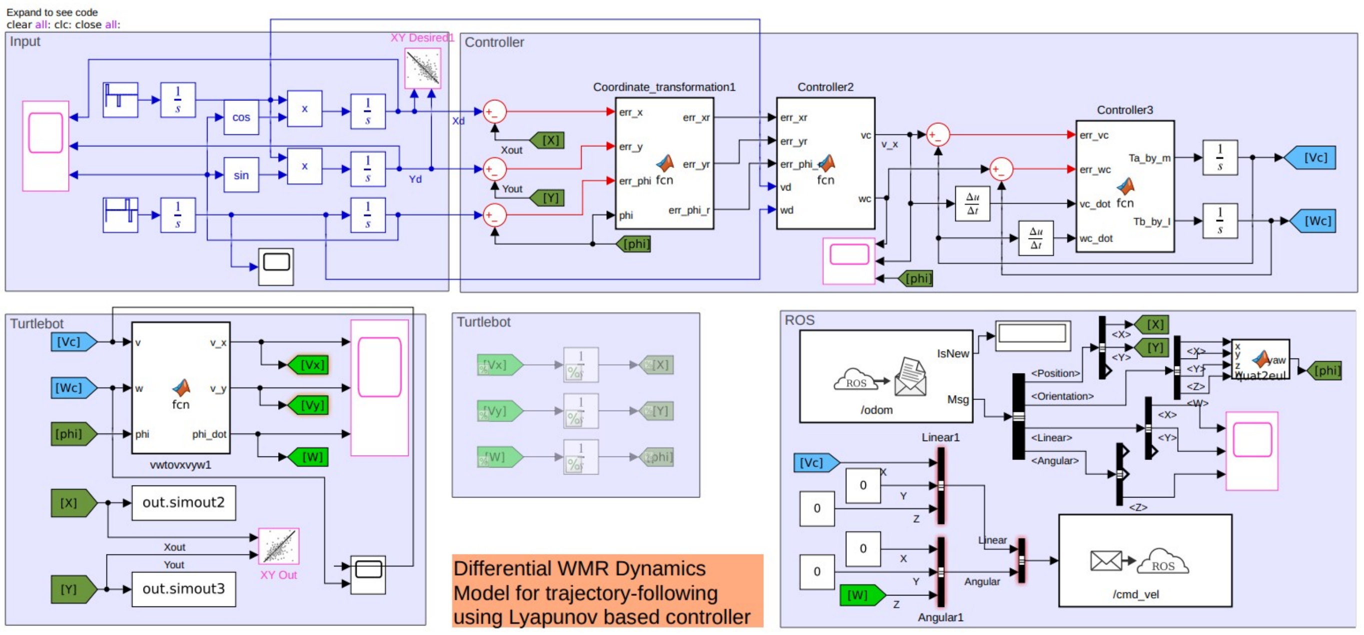


Figure 10: Actual vs Desired Trajectory of kinematic model for trajectory-following using Lyapunov based controller

d. Dynamic model of differential drive Two wheeled mobile robot using Lyapunov controller for trajectory control:



Expand to see code
clear all; clc; close all;
Input
Controller
Coordinate_transformation1
Controller2
Controller3
Turtlebot
vwtovxvyw1
out.simout2
out.simout3
XY Out
ROS
/odom
/cmd_vel
Differential WMR Dynamics Model for trajectory-following using Lyapunov based controller

Figure 11: Turtlebot Dynamic model for trajectory-following using Lyapunov based controller

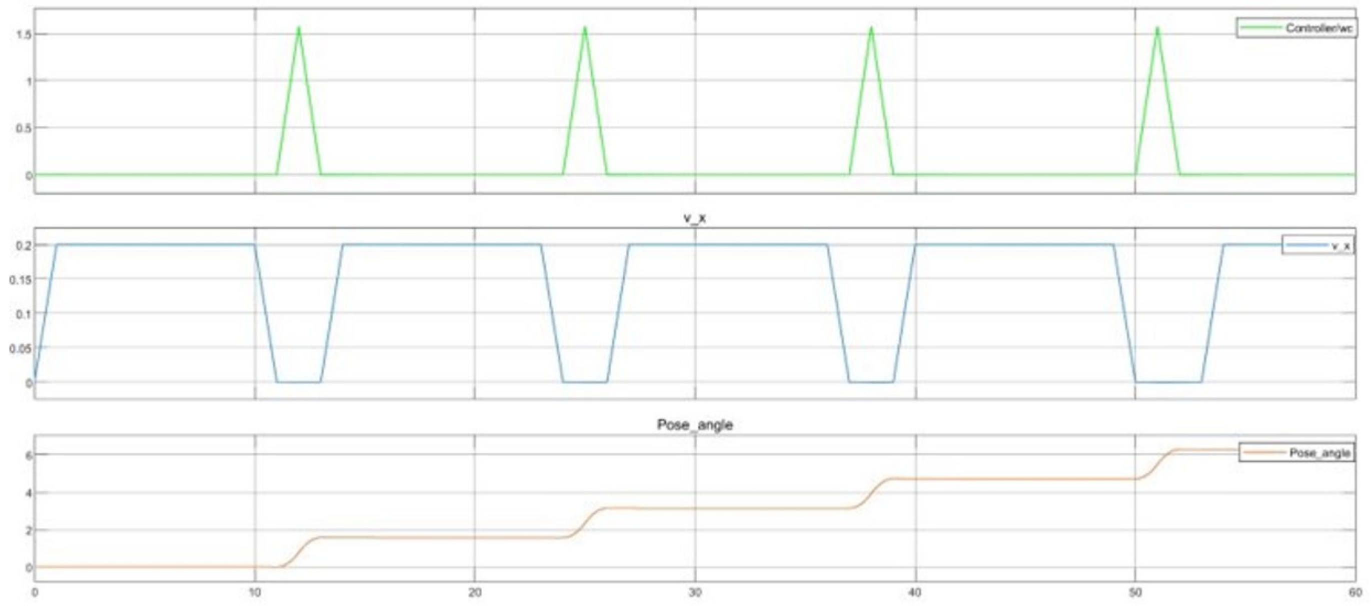


Figure 12: Response of Turtlebot Dynamic model for trajectory-following using Lyapunov based controller

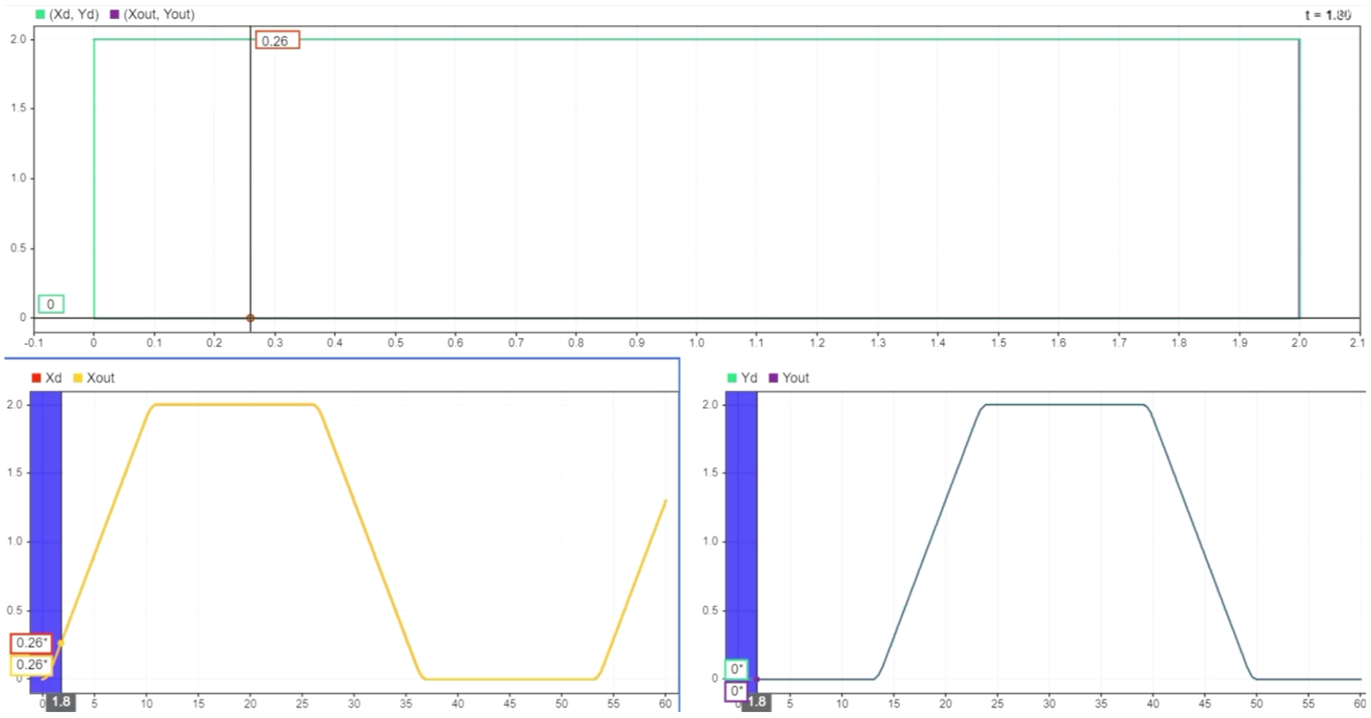


Figure 13: Actual vs Desired Trajectory of Dynamic model for trajectory-following using Lyapunov based controller

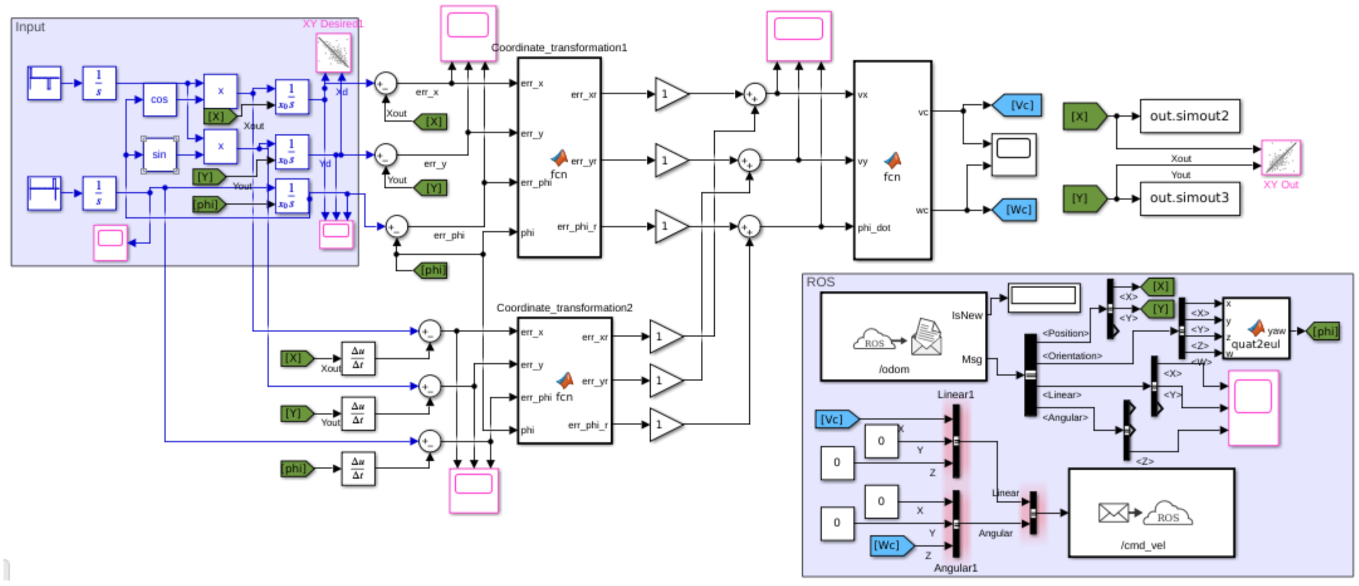


Figure 14: PD controller based Trajectory following scheme using feedback

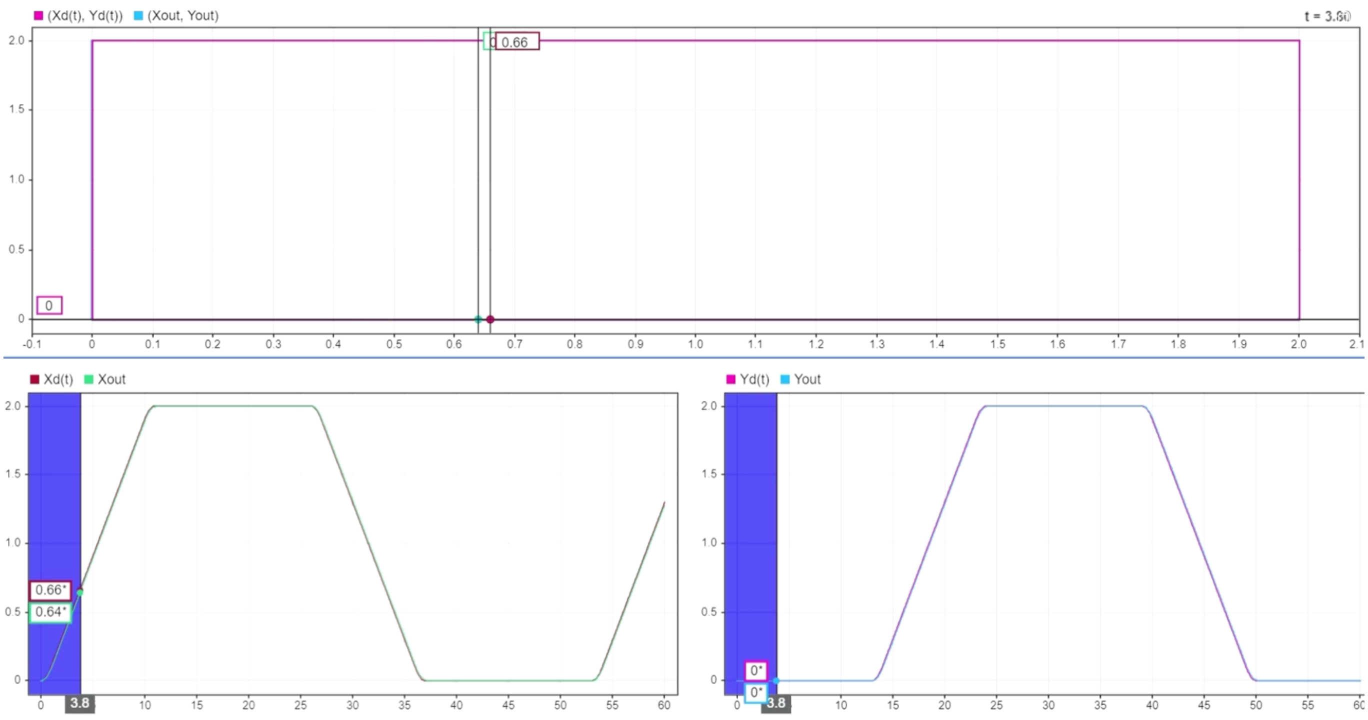


Figure 15: PD controller actual vs desired trajectory

The below is a prototype of Multi-bot Linear-Platooning control scheme. Although this gives satisfactory result for linear case. But some more changes are required to adapt it for 2-D case.

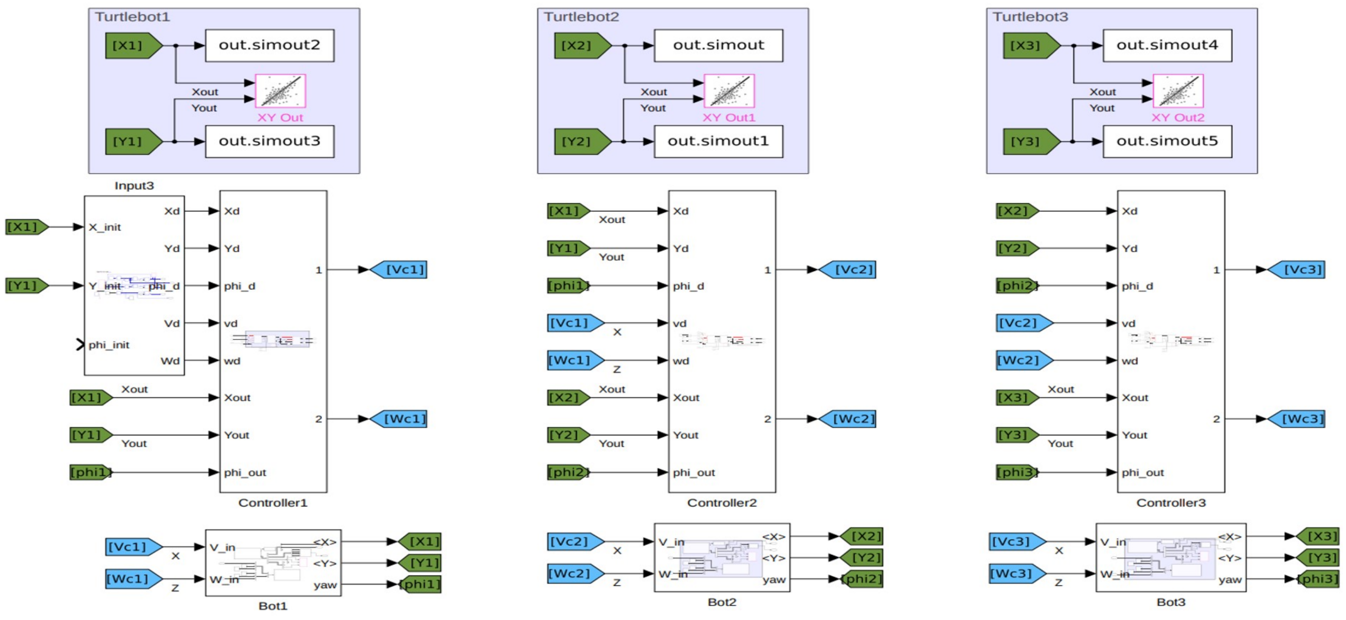


Figure 16: Multi-bot Linear-Platooning prototype-1

7 Analysis and Discussion

7.1 For Lyapunov-based controller

7.1.1 Kinematic model

In the analysis of the kinematic model with the Lyapunov-based controller, we observed that the robot's trajectory tracking performance was highly satisfactory. The Lyapunov controller demonstrated robustness against disturbances and uncertainties in the environment. The controller effectively regulated the robot's velocity and orientation, ensuring accurate trajectory following even under disturbance.

7.1.2 Dynamic model

Similarly, the dynamic model analysis revealed the effectiveness of the Lyapunov-based controller in handling dynamic motion scenarios. Despite the presence of external forces and torques, the controller maintained stable and precise trajectory tracking, showcasing its robustness and adaptability to varying environmental conditions. The control scheme implemented was tested in the Gazebo simulator and then on the actual hardware. In simulator although it was showing overshoot due to negligible friction condition but the response was found out to be better in the hardware. Different types of trajectories were tested under disturbances and we achieved satisfactorily result in them.

7.2 For PD-based controller

Even the simpler PD-based controller showed commendable performance in trajectory following tasks. Just like the non-linear Lyapunov based control scheme, PD controller was also able to attenuate the disturbances under test conditions. This too was tested in Gazebo simulator and satisfactory performance was obtained.

8 Conclusion and Future Scope

In conclusion, the experiments conducted with the TurtleBot3 platform and ROS-based controllers showcased the effectiveness of both Lyapunov-based and PD-based controllers in achieving accurate trajectory tracking for differential drive mobile robots. The control schemes implemented were tested both in the Gazebo simulator and on the actual hardware. While the simulator showed some overshoot due to negligible friction conditions, the hardware implementation demonstrated better performance, particularly in handling disturbances and dynamic scenarios. Various types of trajectories were tested under disturbances, and satisfactory results were achieved.

For future scope, several avenues for research and development are identified. Firstly, exploring advanced control techniques such as Model Predictive Control (MPC) and Reinforcement Learning (RL) could enhance the trajectory tracking performance of the mobile robot. Additionally, integrating obstacle avoidance algorithms, Simultaneous Localization and Mapping (SLAM) techniques, and cooperative control strategies for multi-robot systems could further extend the capabilities of the platform. Furthermore, energy-efficient motion planning algorithms can be investigated for fuel critical applications.

Furthermore, the laboratory experience also provided us with an opportunity to learn and apply Robot Operating System (ROS), a widely used framework for developing robotic applications.

Overall, the laboratory experience provided us with invaluable hands-on learning opportunities in the field of mobile robotics, enabling us to gain practical insights. By actively engaging with the TurtleBot3 platform and experimenting with various control strategies, we deepened our understanding of robotic principles and navigation, control, and trajectory tracking.

Appendix:

8.1 MATLAB code common to all the simulations

```
m=0.94;  % mass of WMR
a=0.08;  % half the distance between two wheels
Kx=1;
K_phi=1;
Ka=1;

% For smooth file with acceleration input
sideLength=10;
Z=zeros(1, sideLength-1);
D=zeros(1, 2);
A_t=[0.2 Z -0.2 D 0 0 D];
w_dot= [0 Z 0 D pi/2 -pi/2 D];

% Infinity pattern
% sideLength=11;
% Z=zeros(1, sideLength-1);
% A_t=[0.2 Z 0 0 Z 0  -0.2]; % infinity  pattern
% w_dot= [pi/6 Z -pi/6 -pi/6 Z pi/6  0];
```

Thank you