Differential Drive Wheeled Robot Trajectory Tracking

By

Yizhou Zhao

A Thesis

Submitted to the School of Graduate Studies

In Partial Fulfillment of the Requirements

For the Degree of

Master of Applied Science

McMaster University

Master of Applied Science (2023) McMaster University

(Mechanical Engineering) Hamilton, Ontario, Canada.

Title: Differential Drive Wheeled Robot Trajectory Tracking

AUTHOR Yizhou Zhao

SUPERVISOR Dr. Fengjun Yan

NUMBER OF PAGES 73

Abstract

This thesis summarizes an approach for building a trajectory-tracking framework for autonomous robots working in low-speed and controlled space. A modularized robot framework can provide easy access to hardware and software replacement, which can be a tool for validating trajectory-tracking algorithms in controlled laboratory conditions. An introduction to other existing methods for trajectory tracking is presented. These advanced trajectory control methods and studies aim to improve trajectory tracking control for better performance under different environments.

This research uses ROS as the middleware for connecting the actuators and computing units. A market-existing global position measurement tool, the UWB system, was selected as the primary localization sensor. A Raspberry Pi and an Arduino Uno are used for high-level and low-level control. The separation of the control units benefits the modularization design of the framework. A robust control approach has also been introduced to prevent the disturbance of uneven terrain to improve the framework's capability to drive arbitrary robot chassis in different testing grounds. During each stage of development, there are offline and online tests for live control tests.

The trajectory tracking controller requires a robot kinematic model and tracking control program for better results of controlled behaviour. A custom trajectory control program was made and implemented into the tests. A digital simulation and a physical robot are built to validate the algorithm and the designed framework for performance validation. This framework aims to suit the other scholar's developments and can be used as a testing platform to implement their autonomous driving algorithms or additional sensors. By replacing the control algorithm in the existing trajectory-tracking

robotic framework, this autonomous, universal platform may benefit the validation of these algorithms' performance in the field experiment.

Acknowledgement

It is an excellent opportunity to express sincere gratitude to the academic supervisor Dr. Fengjun Yan and the visiting student Dr. Ying Liu for guidance and mentorship for this project during the research and development work of the framework.

I specifically appreciate master students Su Liu and Anzhou(Rico) Xing working in the same team for robot system engineering design and development.

I also would express my thanks to Doctoral student Chang (Ash) Liu and undergraduate student Lin (Lydia) Yang for providing support with their knowledge of programming and my research work.

Table of Contents

List of Figures

List of Table

Abbreviations

CANbus	Controller Area Network bus
DC motor	Direct Current motor
ESC	Electronic Speed Controller
GPIO	General-purpose input/output
GPS	Global Positioning System
MPC	Model Predictive Controller
NP-Hard	Non-deterministic polynomial-time hard
PID Controller	Proportional Integral Derivative Controller
PD Controller	Proportional Derivative Controller
ROS	Robot Operating System
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UWB	Ultra-wideband

1. Introduction

1.1 Background

The research effort on autonomous vehicles' autonomous motion control area has been a few decades. Trajectory tracking is a typical motion control problem, forcing the vehicle to move and follow with the time-parameterized reference. [1]Trajectory control is often used in robotic control in the ground vehicle for autonomous driving, aerospace drone flying control and robotic arms motion control. [2] The trajectory control requires sensors to provide information for feedback algorithms. It is considered an NP-Hard problem and requires a long execution time for planning methods. [3] This control method is expected that when more factors are involved, such as the number of vehicles, the computation time will experience an exponential growth rate.

Robots' autonomous driving in the designated area requires a combined system to provide the autonomous driving capability. Autonomous driving can be achieved by combining trajectory/pathway planning and trajectory tracking control. [4] In this project, the trajectory tracking control is focused. The mathematical model is required for building the tracking control system, and the following items must be modelled: Kinematic Model, Kinematic Controller, and dynamics actuators.

1.1.1 Existing Trajectory tracking examples

1.1.1.1 Pure Pursuit Toolbox

The pure pursuit algorithm has been used massively in the tracking control method, calculating the arc around the center point for the robot to follow a trajectory pathway. The robot follows the intersection point from the pre-set angle and the

reference trajectory to the center point designated on the robot. The controller will require a value of "look-head" distance to indicate the arc radius for smaller arcs with more accurate tracking with reference trajectory. [5]

1.1.1.2 Nominal Model Predictive Controller

Model predictive controller (MPC) has been tested for unmanned vehicles and robots for aerial, ground, and navy operations. Under constraints, the MPC can process complex systems, including non-linear and linear problems. [6] The MPC is an optimal control method that generates actions for minimal cost function in a constrained scenario. The MPC usually is iteration based at each iteration or step, which will repeat. The MPC controller will estimate the state and generate the optimized control action under the constraint. [7]

In the MPC application implementation stage, the optimal control toolbox can be used as a black box to process the prediction model by setting parameters up and operating as an MPC controller for the controlled moves.

1.1.2 Middleware – ROS 1 Kinetic

Robot Operating System (ROS) is an open sources operating system designated for robot control. It is a service that provides software libraries, message passing for multiple processes and program package management for robot control applications. Based on the development purpose of this research, the ROS developer's goal matches the research expectation, which is not only providing the framework with features but supporting the developers to reuse their program/codes from their research and make it easy to apply to other frameworks. [8]

ROS 1 Galactic is primarily designated for the framework of Ubuntu 20.04(Focal Fossa). There is no significant difference between the different versions of ROS 2

packages [9]. The distributions of the version of ROS 2 help development work within a relatively stable codebase to prevent rapid changes in the version of the operating system environment. The ROS 2 Galactic has the most recent End of Life date, April 2021. [8] Considering the purpose of this development is to build a stable modularized framework that can be upgraded in the future, the ROS 1 Kinetic Kame for Ubuntu 16.04(Xenial) has been selected as the operating system environment.

1.1.3 Hardware, microcontrollers with low-level control and high-level control

Arduino UNO provides a 5V open-source electronics platform as a programmable board for hardware based on the Atmega328P with 14 digital input/output pins, six analog inputs and a 16 MHz ceramic resonator. [10] The microcontroller on the Arduino UNO board is being used to manage the General-Purpose Input/Output (GPIO) port for direct control of the actuators through Electrical Motor Speed Controllers. This setup aims to modularize the low-level control and separate the functionality from the high-level controller, the Raspberry Pi.

The communication protocol between the Raspberry Pi and Arduino UNO uses serial communication through a USB port. Using a USB port as the serial communication port is integrated with a mature connection protocol. It is easy and can reduce the effort of building communication channels between hardware. In the low-level control between Arduino UNO and Electronic Speed Controller (ESC), the PWM signal through the GPIO port will communicate the signal.

1.1.4 Simulation tools

Webots is an application that enables the simulation of robots and is available across multiple platforms. The software offers a comprehensive development environment to facilitate the modeling, programming and simulation of robots. This

simulation application can be used based on Linux system to provide realistic simulation for robot subframe development and trajectory tracking study. [11]

1.2 Thesis contributions

This thesis aims to develop and provide a modularized framework to provide the capability of testing trajectory tracking. A repeatable result and a reproducible testing platform are the desired result. As the work for the research and development work, this paper has the following contributions:

1. To design and build hardware for trajectory-tracking mobile robots while providing the capable setup for making the trajectory-tracking framework.
2. To provide mathematical and physics background for robot kinematics and dynamics in control system development work.
3. Implement the PID controller into the developed functional trajectory tracking framework to enable the robot pathway tracking capability.
4. Design and create a simulation study of the control system and provide a control group of results for field experiment preparation.
5. Design and create field experiments and collect results for analysis.
6. Validate the repeatability of this PID-controlled tracking platform under the controlled environment.

1.3 Thesis layout

This thesis contains five chapters.

Chapter 1 provided the information and background for this research topic regarding the key components, methods, and tools for creating trajectory tracking.

Chapter 2 focuses on the existing methods and deep study of tools, equipment and hardware setups for trajectory tracking in simulation and physical setups. The experiments referenced from other studies can benefit the research and development work for the current trajectory tracking development work. The review provides different kinematics models for robot layouts, which impacts the final design of the field experiment robot.

Chapter 3 presents the design work process for creating a controller based on the final field experiment robot. This chapter provides steps and considerations while building the control system for a trajectory-tracking robot from scratch.

Chapter 4 demonstrates the simulation results and field experiment results. A study of error analysis and repeatability justification can also be found in this chapter.

Chapter 5 summarizes the research and development contribution, primary findings, and concerns for identified problems.

2. Literature Review

2.1 Model of trajectory tracking for differential drive robot.

The differential drive system allow robot to turn and maneuverable in the workspace. There are many studies provide comprehensive and critical evaluation of building and improving the trajectory tracking. These studies identify the technical gaps, providing innovative finding and most importantly, providing a guideline of enabling the capability of trajectory tracking for robot implementation. The model of robot is the key for a controller to control.

2.1.1 Kinematic modelling introduction

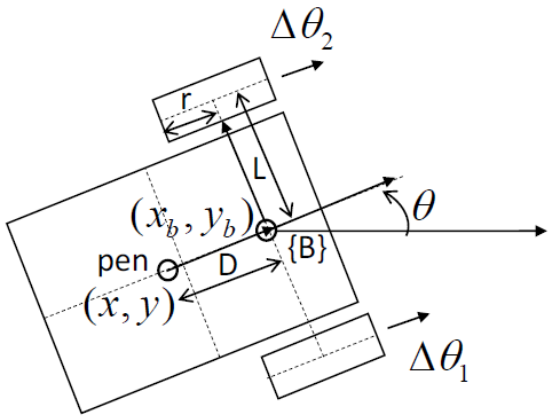

Figure 1 A normal Differential Drive Robot Configuration [24]

The following equations are obtained from the early basic robot configuration as the demonstration for the kinematic analysis of the differential drive robot. The factors that will be considered in this model will be the Radius of the drive wheels *r*, the distance between drive wheels *L*, and the robot orientation angle measured from the x-axis *Θ*.

$$v = r\omega \tag{1}$$
$$w = \frac{d\theta}{dt}$$

The linear velocity, *v*, is in m/s; the Angular velocity, *w*, is in rad/s.

The speed of the individual drive wheel will be $r * \omega$, which will lead to the translational speed of the robot at the average velocity of:

The rotational velocity for drive wheels is:

$$\omega = \frac{v_R - v_L}{L} \tag{2}$$

$$v_{mob} = \frac{r(\omega_R + \omega_L)}{2} \tag{3}$$

2.1.2 Dynamic modelling introduction

The inertia frame and the robot mapping will be considered in standard orthogonal transformation, so the velocity in the global kinematics is:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r(\omega_R + \omega_L)}{2} \\ 0 \\ \frac{r(\omega_R - \omega_L)}{2} \end{bmatrix} \tag{4}$$

This robot's dynamic modelling study will focus on forces and energies—the relationship between the control signal and mechanical system input.

I: Dynamics of the actuator: Open loop transfer function for DC motor.

$$\frac{\omega_m(s)}{V(s)} = \frac{k_t / n}{\left[(L_a J)s^2 + (R_a J + BL_a)s + (R_a B + k_t k_b)\right]} \tag{5}$$

II: Dynamics of the chassis: Lagrange equation of motion

$$\begin{aligned} \dot{v} &= \frac{(T_L + T_R)}{m} - k_v v \\ \dot{\omega} &= \frac{(T_L + T_R)l}{r} - k_\omega \omega \end{aligned} \tag{6}$$

2.2 Vehicle Modeling of Parameters, Hardware, Subframe and Low-level Control

2.2.1 Mobile-robot control-system example

Many trails of the physical robot with differential drive designs have been built in this study. This thesis will discuss two physical robots to consider their kinematics and other property. The second subframe will be used as an example for simulation and control system verification in the comparisons for experiment performance.

The objective for finding control laws for the linear and angular velocities for the robot system at the following condition:

$$\begin{aligned} &\lim t\to\infty |y(t)-yr(t)| = 0 \\ &\lim t\to\infty |x(t)-xr(t)| = 0 \\ &\lim t\to\infty |\phi(t)-\phi(t)| = 0 \end{aligned} \tag{7}$$

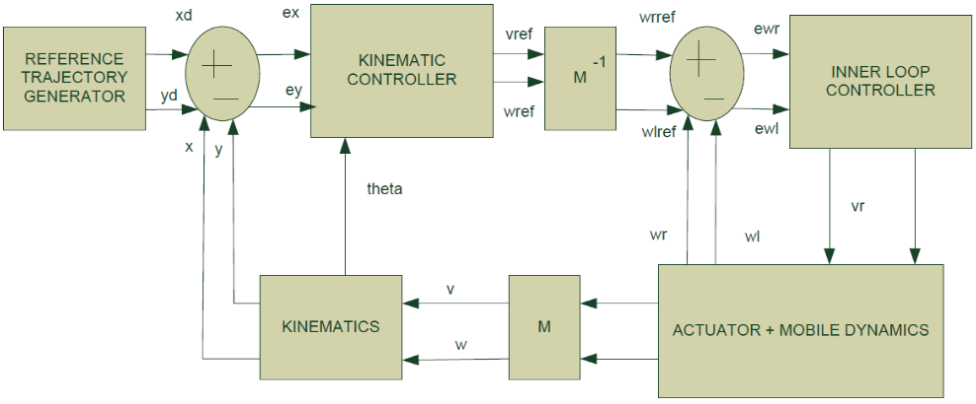

Figure 2 Control Block Diagram [11]

- The outer kinematic controller will read the errors in the x and y coordinates between the reference and the actual position obtained by sensors. The equation for the outer loop is listed below:

- The inner loop controller will be a simple P controller.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = [M] \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{8}$$

$$M = \begin{bmatrix} r/2 & r/2 \\ r/l & -r/l \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ \frac{-1*\sin\theta}{a} & \frac{-1*\cos\theta}{a} \end{bmatrix} \begin{bmatrix} \dot{x}_d + l_x \tanh \frac{k_x}{l_x}(x_d - x) \\ \dot{y}_d + l_y \tanh \frac{k_y}{l_y}(y_d - y) \end{bmatrix} \tag{10}$$

The linear velocity and angular velocities will be transferred into the individual wheel angular velocities. Transformation can be done by obtaining the motor information.

In the example, the control scheme for trajectory tracking is implemented in simulation software.

MATLAB Simulink. With the following attached simulation diagram, the robot can move with different behaviours to reach a point, and the generation of the required trajectories checks to track lines.

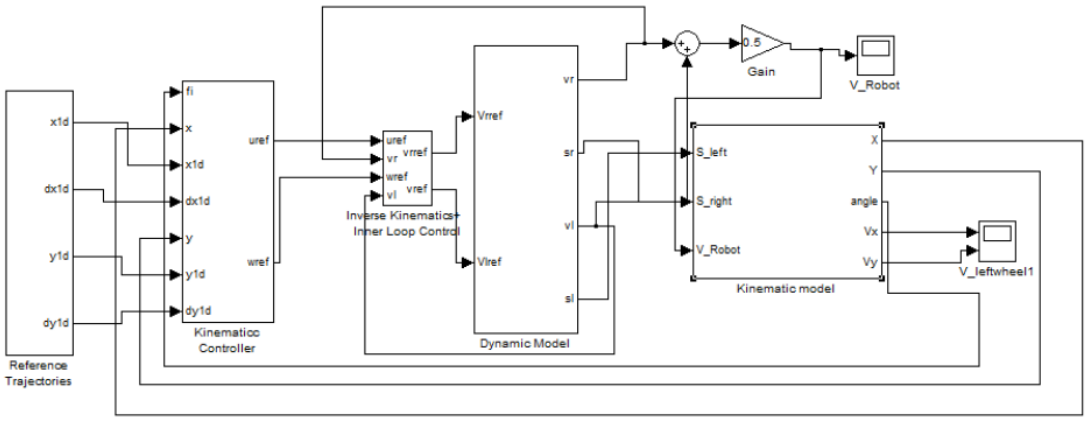

Figure 3 Simulink diagram for trajectory tracking [11]

The simulation result from the tracking representation is listed below. The result is generated based on different cases.

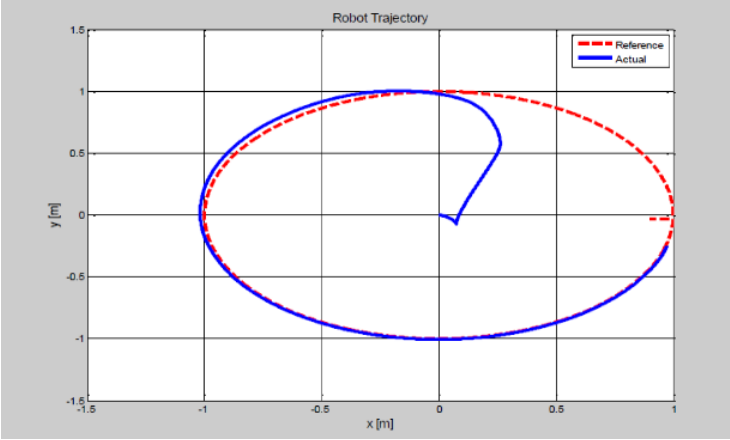

Figure 4 The trajectory tracking for the robot on the circular trajectory.

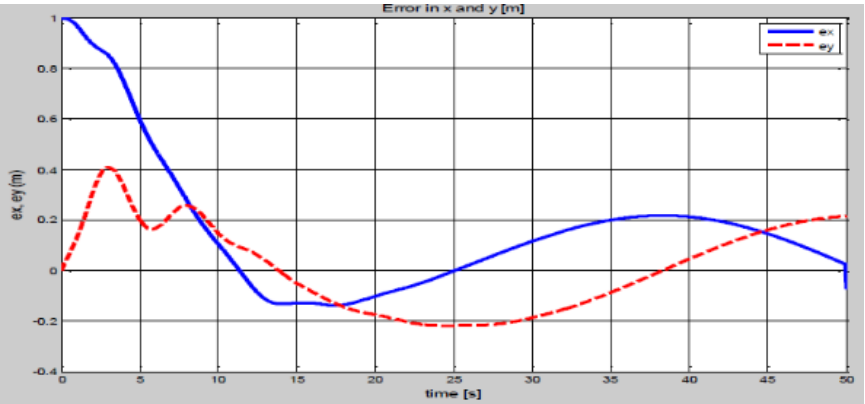

Figure 5 The error figure in the x and y-axis of the circular trajectory.

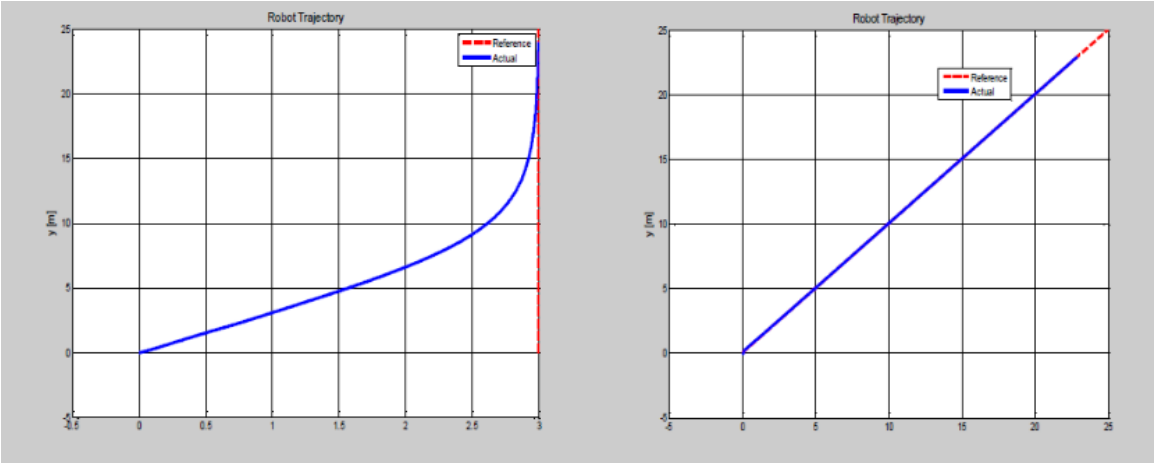

Figure 6 The trajectory tracking for the robot on the curved trajectory and straight line.

2.3 Robot subframe Kinematics model, limitation, and performance constraints

The kinematics model is required for the plant model, transfer function and future control system. In considering the kinematics model, the differential drive requires separate consideration of the wheels in case they are slipping or not slipping. Through the slipping concern, the differential drive robot subframe can be considered as the following scenarios:

Table 1 Slipping condition table.

Scenario	Right wheel	Left wheel
1	No Slipping	No Slipping

2	No Slipping	Slipping
3	Slipping	No Slipping
4	Slipping	Slipping

By following these scenarios, the kinematics model of each scenario can be generated to represent the real motion situation of the robotic subframe accurately. The following equation parameter can reference in Table 3 Symbol/Notation table. [12]

Identification of the slipping check:

$$\left|\ddot{\theta}_{wheel}\right| > \frac{g\mu_s}{4r} \tag{11}$$

Rolling:

$$F_{wheel} = \left(\frac{m_{robot}r}{2R_g}\right)\ddot{\theta}_{wheel} \tag{12}$$

Slipping:

$$F_{wheel} = \frac{1}{4}\mu_k m_{robot} g \operatorname{sign}\left(\frac{\dot{\theta}_m r}{R_g} - v_{wheel}\right) \tag{13}$$

Based on the previous kinematics equation, the slipping and none slipping can be applied to the scenarios while the control system determines if the motion matches the motion scenario characteristics. [12]

2.4 Basic chassis controller and Arduino-based GPIO inputs introduction

An Arduino UNO microcontroller will control the low-level chassis controller. The Arduino UNO will use the GPIO signal as input and output ports by receiving the signal from high-level control and sending the signal to ESC, respectively. The Arduino UNO is programable to control the power output and signal input scale. In future studies and modifications, the Arduino UNO controller can be replaced by other controllers to ensure the low-level control is modularized. It is recommended to use upgrades, including a Controller Area Network bus (CANbus) with other microcontrollers for more sophisticated control of various chassis in the robot system. However, in the current application, the GPIO communication protocol is more friendly to be implanted and develop.

3. Control System Design

3.1 Robot Subframe Design and Modeling

3.1.1 Design Scope

This robot's subframe's design scope aims for the original design requirement: a working robot that can handle multifunctional tooling for different task requirements. However, in the various design stages, the design scope is different.

The early design scope is to enable the capability of manual control to validate that the basic powertrain is working. Also, in the second robot design, the tooling is installed in the central area, which requires four wheels to support the entire robot. This design is not mature since the central location has limited scalability to replace another tooling due to the limitation of space and the way of operating.

The design scope changes in the later design stage since the robot's mechanical design needs to meet the following requirements.

Table 2 Design requirement for field experiment Robot

1	Simple to mathematical modelling
2	Friendly to differential driving control algorithm
3	Capable of installing arbitrary tooling with free space
4	Capable of replacing different wheels for different terrain
5	Powerful motors for heavy-duty work
6	A space for control hardware installation

Based on these requirements, the following robot has been designed. The front caster wheel will temporarily support the robot and simulate the low weight of the tooling payload on the robot.

3.1.2 Kinematic Modelling

In this plant model, we treat this mobile robot as a rigid, uniform cylinder moving. The plant itself has a mathematical relationship to the real-time position of the robot in the global fixed coordinate of *(X, Y)* between the robot moving direction *(X_m, Y_m)*. However, in this case, we consider the robot path is either straight or curved, so there is no sliding in the Y_m direction. The Φ is the rotating angle of the direction of the robot's central axis, which is always perpendicular to the shaft of the wheels. Based on this, we can have the mathematical relationship equation (13) to transfer the mobile robot speed into the global fixed coordinate.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos\,\varphi & 0 \\ sin\,\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{y}_m \\ \dot{\varphi} \end{bmatrix} \tag{14}$$

In this model, the linear velocity of the center of the wheels in the robot in the Y_m direction is as the following linear velocities:

$$v_{w1} = \dot{y}_m + L\dot{\phi} \text{ and } v_{w2} = \dot{y}_m - L\dot{\phi} \tag{15}$$

The net force for the robot in the Y_m direction is:

$$F_y = F_{w1} + F_{w2} \tag{16}$$

From the forces on the wheel through the length of shift, we can have the torque applied on the robot from wheels at:

$$\tau_z = L(F_{w1} - F_{w2}) \tag{17}$$

The uniform cylinder robot with a mass of m will have the result of the following linear and angular accelerations:

$$\ddot{y}_m = \frac{F_y}{m_r} \tag{18}$$

$$\ddot{\phi} = \frac{\tau_z}{\frac{1}{2} m_r {r_c}^2} \tag{19}$$

In the situation of no slipping of the wheels in the motion, the kinematic rotation model of wheels is:

$$F_{w1} = \left(\frac{m_r r}{2R_g}\right) \ddot{\theta}_{m1} \tag{20}$$

$$F_{w2} = \left(\frac{m_r r}{2R_g}\right)\ddot{\theta}_{m2} \tag{21}$$

From the equations we mentioned above, we can sort the equations to clear out the relationship from the initial variables; electric currents i_1 and i_2 will power the entire plant model. The acceleration of $\ddot{\phi}$ and the acceleration of $\ddot{y}_m$ can be formed in the following equations.

$$\ddot{y}_m = \frac{\left(\frac{m_r r}{2R_g}\right)(\ddot{\theta}_{m1} + \ddot{\theta}_{m2})}{m_r} \tag{22}$$

$$\ddot{\phi} = \frac{l \times (\left(\frac{m_r r}{2R_g}\right)(\ddot{\theta}_{m1} - \ddot{\theta}_{m2}))}{\frac{1}{2} m_r {r_c}^2} \tag{23}$$

The equation above contains the following terms:

Table 3 Symbol/Notation Table

Symbol/Notation	Description	Units
$\ddot{y}_m$	Linear acceleration on the perpendicular direction of robot motor shaft for uniform cylinder mobile robot	m/s^2
m_r	Mass of the entire robot	kg
r	Robot wheel radius	m
R_g	Transmission Gear Ratio	N/A

$\ddot{\phi}$	Angular acceleration of the angle changes on the perpendicular direction of the robot motor shaft for uniform cylinder mobile robot	$degree/s^2$
$\ddot{\theta}_{m1}\ and\ \ddot{\theta}_{m2}$	The wheel speed of the right and left wheels respectfully	$degree/s^2$
r_c	Cylinder robot radius	m

Hence, by controlling wheel acceleration, the global coordinate can be controlled by manipulating the wheels' acceleration through the robot plant.

3.2 Hardware

3.2.1 Implantation of the sensor reading and input of the control system

In the robot's control system, the only sensor that provides feedback information is the UWB position sensor. UWB position sensors will be sent back the real-time location of the robot through the publisher in the program into the ROS server for other hardware or program to read. However, the accuracy of the UWB position sensor may not be sufficient for the currently designated working robot and environment. In this case, the collected online waypoint data will act as a fixed and accurate feedback position data array. Fixed feedback position data is used to test the framework and controller's repeatability. Under the controlled environment and input, different approaches can be compared.

The UWB readings will communicate through the micro-USB port to transmit reading, publish the UWB tag reading to service topics in the ROS system, and wait for programs to retrieve and process.

3.2.2 UWB Localization

3.2.2.1 Experiment for UWB live testing and data reading

The UWB testing is the major test at an early stage. Understanding the sensor read data requires observation from recorded sensor reading and will play an important role in implementing the tracking control program. The UWB represents the low-cost sensors that can be easily acquire and install on the robot. Due to the low quality and low-cost, using the UWB data to precise track the robot location is becoming a challenge, however, if the precision of the low-cost sensor can be improvement through the control system, it will bring cost reduction benefit in robot applications. The tracking control program works based on how the position data is measured. Since the UWB sensors provide the local designated area's coordinate based on the anchor's location, there are 2 reference frames between the low-level control on the electric motor voltage reference frame and the global coordinate reference frame.

3.2.2.2 Error observation

The UWB localization sensor provides a 10Hz of the sampling rate. However, through the live testing, online/offline reading and recording of the sensor, empty readings of the location data occur frequently. According to the experiments of UWB reading records, measured individually on the reading records, the percentage of missing reading data is 4% based on the measurement from the laboratory team.

It is a common understanding that error occurs in the sensors. However, since the deep missing analysis of the sensor accuracy specification, the error should be well

maintained in the controlled environment to have a controllable error. This approach treats the error as consistent in the following study and development.

3.2.2.3 Discussion of equipment limitation.

As previously discussed, the available localization sensor is the DWM10001 localization sensor. According to the experiment tested, the best performance of UWB is under the 6.5 meters x 6.5 meters square range based on the testing on the performance of current UWB sensors.

Since the current lack of sensors to provide enough real-time pose, the real-time location is the only source of feedback to serve the tracking. In this trajectory tracking study, the design of the experiment will assume the location position data are accurate and acceptable. This research scope is for trajectory tracking only. Hence, the accurate location data reading and the final testing area will be controlled within $5 * 5\ meters^2$ to ensure accuracy.

3.2.3 Approaches of accuracy increment

Several ways have been conducted during the development to increase the accuracy of data collection and control.

A fixed, accurately measured and marked working space was created at the experimental level. The measurement accuracy is based on the measurement tools, which are 0.01 meters. The angles are built based on a laser tool. After the measurement, semi-permanent duct tape was applied on the ground to mark the reference frame and the robot's desired pathway.

At the simulation level, an identical robot module was built in the simulation software, which provides the identical weight and dimension of the testing model of the

robot. The physical experiment workspace and a mimicked desired tracking pathway are also duplicated in the simulation world.

3.3 Network middleware

3.3.1 ROS Implementation

ROS provides server services to the control system. Once the UWB sensor acquires the readings, the program will upload/publish the information on the server, which can be read through the ROS commands by reading topics. Once the tracking control program requires a new ready-of-the-location coordinate, it will read the most recent data from the topic and process it within the program. The drive command generated from the control process will also be uploaded through the ROS server into the specific topic. [8]

3.4 Control system design

The control system development in this topic uses an open-looped subframe with closed-looed position data to provide feedback on the desired waypoints tracking. The control system only controls the high-level position with the required acceleration output to send a signal to the subframe to react to the acceleration through a digital signal.

While considering the previous studies on the pure pursuit and MPC methods, both methods show flaws in implementing this robot framework.

The pure pursuit in this application requires the information of entire trajectory functions. Since this robot trajectory tracking framework may only receive the pathway information for the following action point, the whole trajectory equation may not be explicit. The pure pursuit method in this application may can and only can reach the next waypoint at a specific time.

The MPC method requires high computing power for a higher number of optimized result generation. Based on the simulation result while using the toolbox of MPC on MATLAB, a 51 iteration of controlled action requires 282.446 seconds to finish. The high computing time is not acceptable for a real-time tracking robot. In this simulation, the N_p value is 5. However, in the Linux-based simulation and field experiment, a new toolbox from other recourses is needed since MATLAB does not support the Python application.

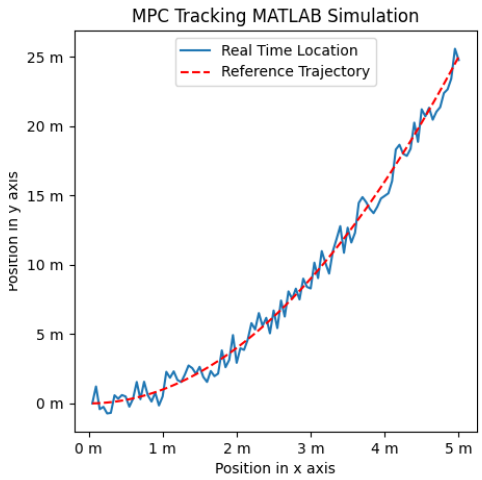

Figure 7 MPC Simulation in MATLAB by using the Yalmip toolbox.

Since the only feedback loop is the position data, the concept of the dead-reconning with fast reacting PID controller will be used to track the designated trajectory. Dead reconning is commonly used in naval or aerial navigation to determine the position based on the record of courses without the aid of celestial navigation. [13]The only known information is the starting point and the drift. In this case, the real-time position UWB data matches this information requirement. Dead reconning can be a good concept in the controller design reference.

Based on the previous consideration, a trajectory tracking controller should be designed as easy to develop, require less computing power and not need an external toolbox. The PID controller combines with following the following tracking method to meet the desired outcome. A tunning PID value coefficient has been developed during the experiment as the improved estimation of the final position poses status from the mathematical perspective to increase the approximation of the driving direction based on the experiment and simulation results. These actions can act like manual calibration/compensation work for the platform to carry the control system.

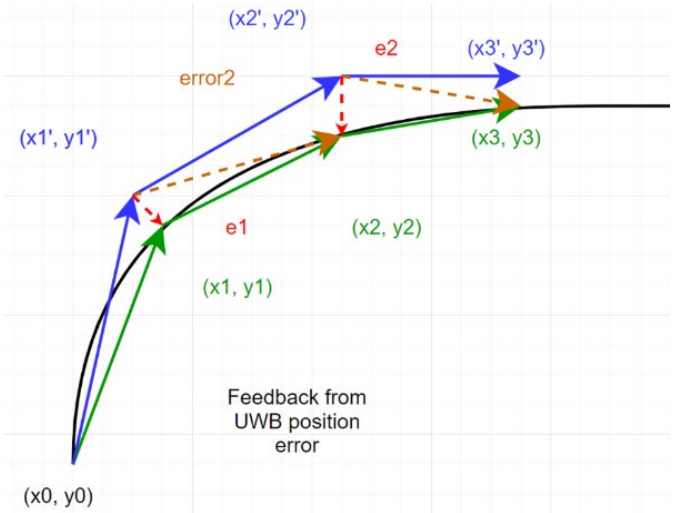

Figure 8 A simple demonstration of control tracking theory

The control system design requires a clear goal of what it will achieve. As the base parameter development, the time interval will be a constant to divide the entire tracking motion universally.

1. The control system shall output u_1 and u_2 as the control variable for a differential drive.

2. The control system requires the sensor reading measure location and target location to generate the necessary robot angular displacement and linear motion requirement.

On the other hand, the level of variables in the dynamics of this control system design scope can be listed below:

1. Measured location P_x, P_y.
2. Targeted location T_x, T_y.
3. Required velocity V to react for motion.
4. Required wheel accelerations to respond to needed V.
5. Required control variables of u_1 and u_2 to cause wheel acceleration.

In these five layers of variables, cumulative errors will be introduced in each layer. The only variable that does not contain error is the targeted location T_x, T_y since they are pre-planned. The measured location P_x and P_y has a gaussian error, and the following calculations are based on the measured location. The time interval is a constant, but the discrete system transfer will also introduce error. The control system supposes to compensate for these errors. [14]

The practical problem of the control system is:

1. The battery voltage fluctuation may cause the *u1* and *u2* to impact uncontrollable acceleration control.
2. The system time is not the perfect time to match the desired frequency and motion time. However, this can be solved through the dynamic waiting time in the algorithm design.

3.5 Tracking Control Design

3.5.1 Tracking Control Method of Error Correction for better performance

In the tracking control design, a control method must be determined to plan the architecture of the control algorithm. Considering the feedback information from the sensors, the only sensor input in the system is providing the localization data to the control system. The PID controller will control and correct the error by developing algorithms based on the error elimination concept. [15]

The integral components accumulate past errors to minimize the steady-state and tracking error. The derivative part will reduce the overshoot. In the dead reckoning method of trajectory tracking, the driving direction angle is the most critical variable that needs to be controlled in this tracking control since the driving direction angle is an open loop having no sensor to transmit back to the controller. The tunning for the PID controller has to be done through manual testing of the K_p, K_i, and K_d. However, in the current system, the controlled variables are:

Proportional: velocity

Integral: displacement

Derivative: acceleration

In this system, controlling the displacement has no physical meaning in the current system since only velocity and acceleration can be controlled. The integral will be deducted from the PID controller that the PD controller will be used in the controller design. The PD controllers reduce the rise time, overshoot, and oscillations in the output variables. The PD controller reduces the entire system's rise time and steady-state error. In this case, a PD controller is enough to control the system's behaviour. [16] In

this case, two PD controllers will be controlling the robot heading direction and robot global main body linear velocity represented in the variable of *Phi* and $Y_{m,}$ respectively.

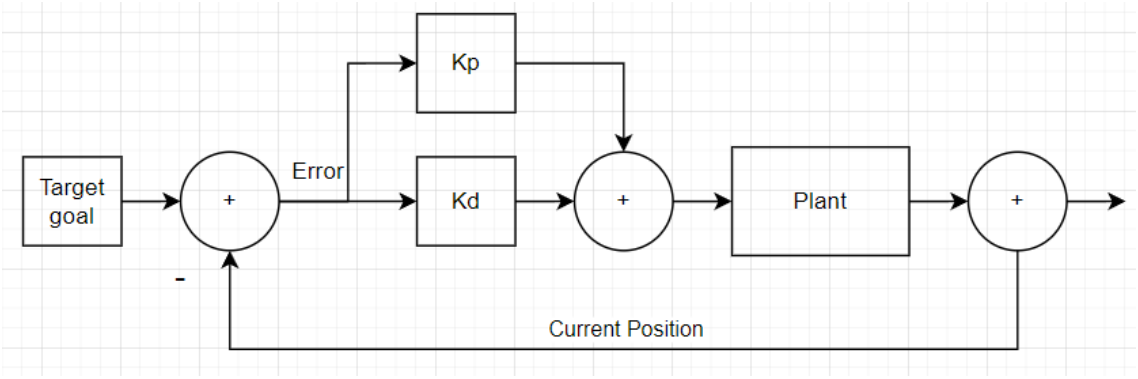

Figure 9 PID Control Diagram

3.5.2 Implementation of the Algorithm in the Operating System.

The development platform is on the Linux environment Ubuntu 16.04. The ROS2 Galactic has been used as the middleware to connect the actuators, main control program and sensors.

Since the main compatible language of ROS2 is C++ and Python, Python is more familiar based on the previous study. Hence the selection of Python would be optimal for the development language of the system. The ROS2 and the control program are constructed in Python language. However, in the setting and launch program of the simulation program, WeBot, the knowledge of C++ is required, which needs effort for the unfamiliar programmer. The C++ program for setting up the WeBot simulation program is being thoroughly tested. With the comments and program documentation, any other researcher shall have no problem editing or tuning other models on this simulation.

3.5.3 Offline data reading and testing with the tracking program

The offline reading data bags were read before the previous work with actively running UWB on online data reading. The offline reading data tasks rely on the ROS-provided offline data bag functions to recall and rerun the recorded offline data identically as when the reading was acquired during online reading. The offline data bag is valuable for testing the program's repeatability and stability. After trying the offline data bag with the control program, the resulting output must be monitored to see if the result matches the desired output. If the result is in the selected range of the output, the next step will be running on the simulation software. In the simulation software WeBot, a particular sensor module called the "GPS" node is being installed. The GPS node will simulate the behaviour of the UWB localization sensor. The tracking control program will read the GPS data at the same sampling rate as the reading in the UWB localization sensor to ensure the similarity of controlled behaviours.

4. Simulation, Experiments and System Verification

4.1 Simulation

4.1.1 Simple robot simulation for pathway tracking

In a high-level design of this simulation, the scope is to build a very close parameter of the physical robot in the simulation to study and improve the control system.

In the first simulation trial, this robotic system's control unit uses the Library of MATLAB Robotics Systems Toolbox and Navigation Toolbox. The Pure Pursuit Block is a tool to compute the linear and angular velocity to track a path using a set of waypoints with the current pose of the differential drive robot. Pure pursuit uses the Max angular

velocity and Desired Linear Velocity as the main parameters to update the velocities as the factor to alter the robot's performance. [17]

The Inputs of that Pure Pursuit function are the pose, the current vehicle's pose, and the waypoints. The parameters of the functions are linear velocity, angular velocity, look-ahead distance, and target direction indicators.

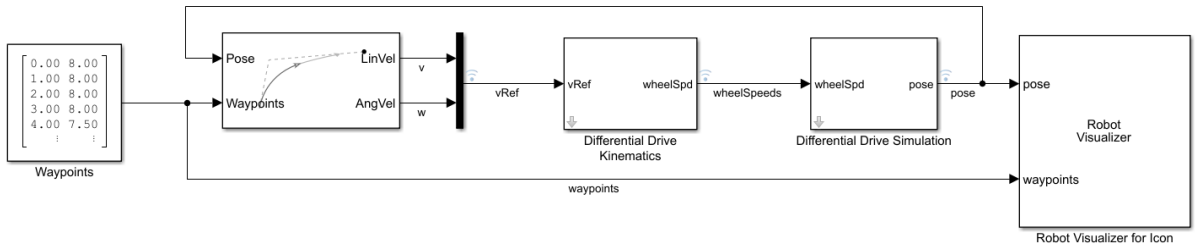

Figure 10 Simulink of Path Tracking with Pure Pursuit Function [18]

While using the Pure Pursuit Function, the look-ahead distance parameter significantly impacts the algorithm's performance. This parameter is the main tuning property for the controller. By regaining the direction and maintaining the path, the look-ahead distance will cause different waypoints following characteristics. The following simulation will show the difference.

The following tracking pathway is having look-ahead value of 0.5.

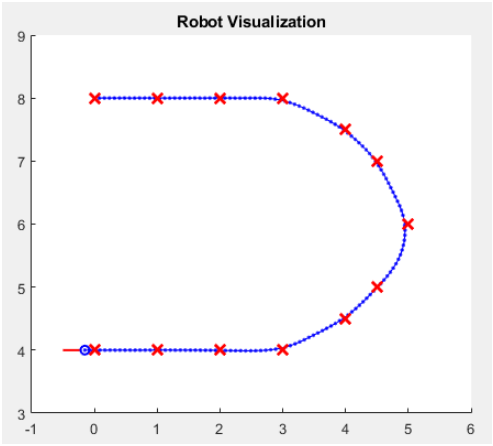

Figure 11 Robot Visualization Simulation, look-ahead value is 0.5.

The following tracking pathway has look-ahead value of 1.

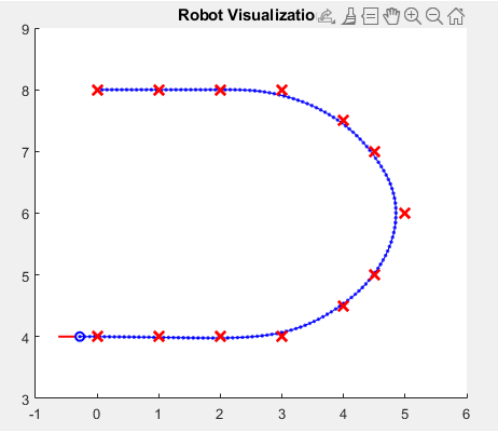

Figure 12 Robot Visualization Simulation, look-ahead value is 1.

The following tracking pathway has a look-ahead value of 0.1.

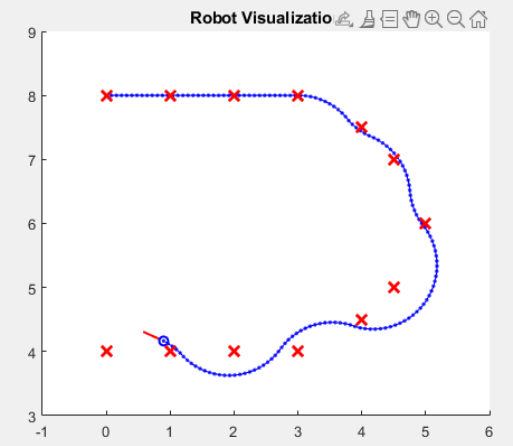

Figure 13 Robot Visualization Simulation, look-ahead value is 0.1.

Comparing these different values of the Pure Pursuit tracking shows the various motions of tracking the waypoints. This parameter requires tunning to face different environments for tracking purposes. However, this may be influenced by the sensor's sampling rate and the processor's preference. It may require specific tuning to fit different dynamic models and environments.

4.1.2 Hardware, modelling, Robot subframe model and control architecture

There is a total of 4 robots have been developed in this study.

Robot 1 is purchased from an online store with a built-up chassis for easy implementation of control. The dimension of this robot 1 is *126cm x 40cm*. The weight is 23.2 kg. Arduino Uno microcontrollers and four DC motors are installed on the subframe. However, there is no encoder to provide wheel speed as feedback. A dual 12V power supply powers the entire system.

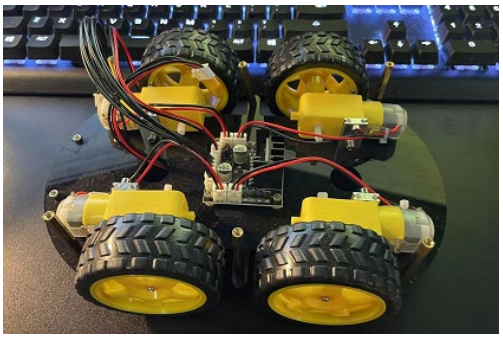

Figure 14 Robot 1 picture

In this robot 1 four-wheel design, the wheel slip will be critical for the differential speed steering behavior. The longitudinal slip of wheels on both sides would stay different on two sides while turning. The proportional relationship in the equation of speed for wheels is satisfied into a new equation which is: [19]

$$\frac{\Delta x}{\Delta y} = \frac{\omega_R}{\omega_L} \tag{23}$$

The kinematical model for differential speed steering will be shown in the graph below: [19]

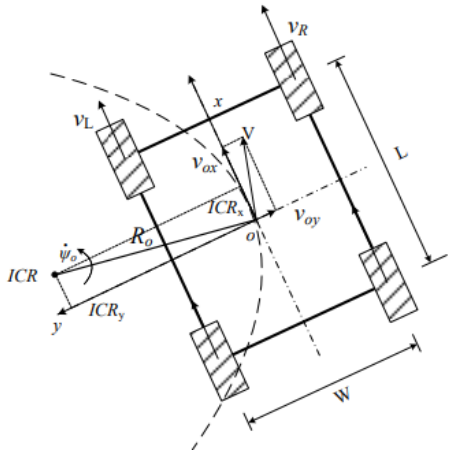

Figure 15 ICR diagram for 4 wheels differential drive [19]

Robot 2 is built by the research team. In the design scope, this robot application is set to achieve a lawn mower structure with rear-wheel drive. The robot's central area contains enough space to install a rotatory blade-cutting system. The front wheel initially chose the caster wheels to prevent the influence of additional inertia and friction for the kinematics model, but this is a faulty choice since the rotation of the casters is unpredictable from a control perspective. The caster has no suspension capability to provide sufficient grip with the group and may free-rotate if the caster is dangling due to uneven terrain. The fixed front unpowered wheels can provide predictable fixed angle and friction input to be considered in the controller. However, due to the poor workmanship, robot 2 is unsymmetric, very fragile and experiencing vibration problems. Robot 2 was destroyed during one of the manual control tests. The team built an improved version based on the robot 2 design.

Figure 16 Robot 2 picture

The creation of robot 3 was drafted from the Computer-Aided Design (CAD) process and manufactured by the local metal workshop. It was used to perform tests the manual control and early trajectory tracking control.

The picture of robot 3 is attached below:

Figure 17 Robot 3 picture

The final robot 4 design aims to minimize caster wheels' uncontrollable behaviour and enables the modular loadout carrying capability. However, the four-wheel kinematics modelling is different from two wheels model. [20]

This robot is lighter and more agile since the power wheel is big and shares a low turning inertia in the horizontal direction. Since the trajectory tracking control system will be implemented on this version of the robot, the robot design should provide suitable parameters for a simplified dynamic model for the control system to prevent the control system from reaching constraints. The parameter of robot 4 is attached below:

Table 4 Robot 4 Parameter

Items	Value
Shaft Distance	0.61m
Wheel size(radius)	0.15 m
Weight	5.78 kg
ESC signal recipient range	-255 to 255
Voltage of battery	20V

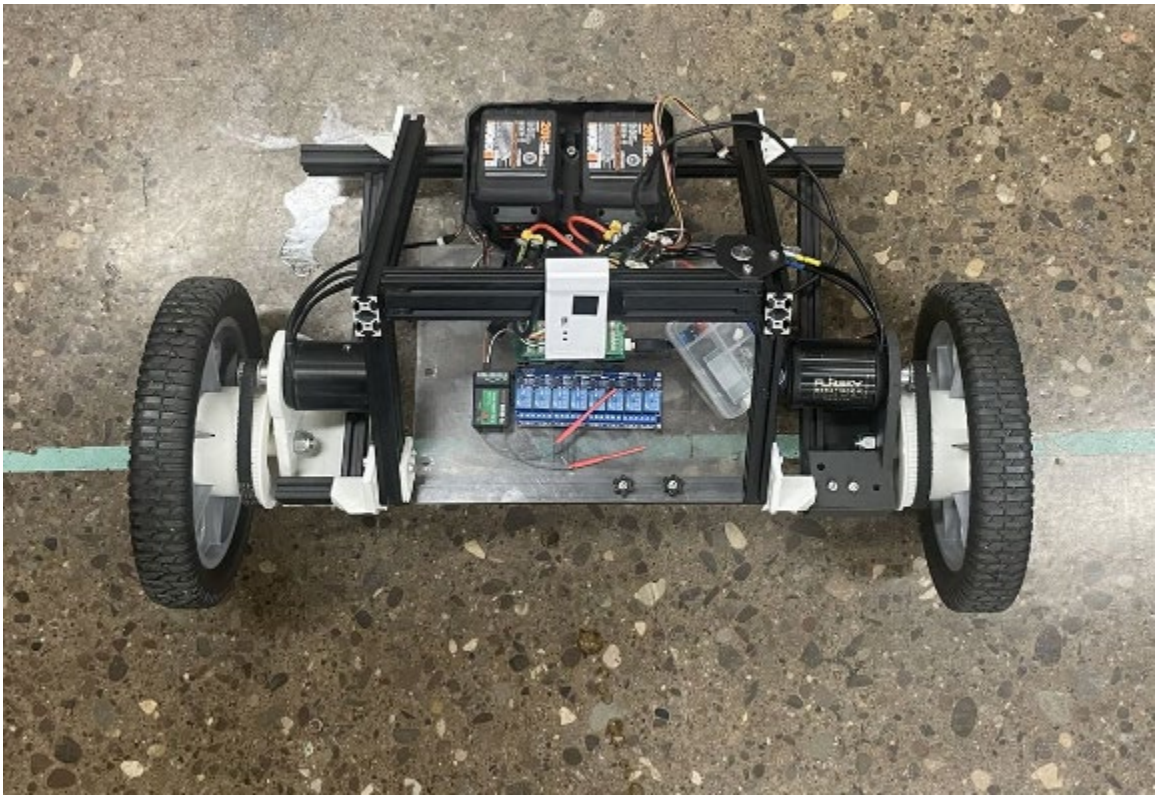

Table 5 Robot 4 picture

The assumption of no-slip motion is considered in the initial study to ensure the experiment and robot motion control model is accurate under the controlled environment and motion constraint. The maximum critical acceleration for the existing subframes may require an experimental value. In this case, the robot acceleration is controlled in a safe zone to prevent uncontrollable slipping. The slipping management and mathematical model will be discussed later in the content.

The hardware setup used in the simulation is trying to be identified as the physical robot, including the most important parameters: shaft distance, wheel radius and robot weight.

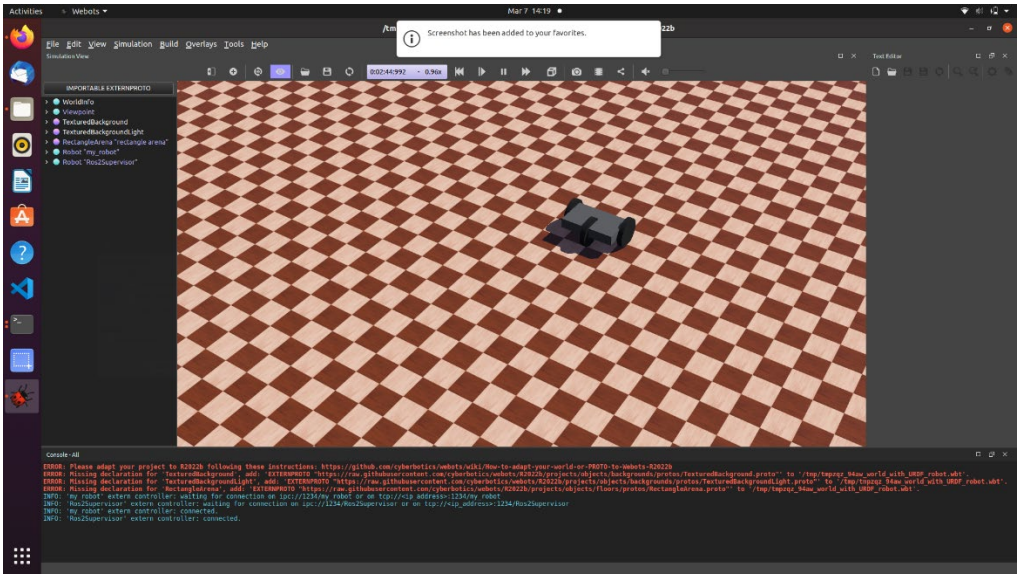

Figure 18 Simulation Picture

The simulation robot is shown in the following screenshot from the simulation software. It should be clear that the difference between the simulation and the physical robot is different. In the simulation, the motor parameter's-controlled driver is the motor's velocity. However, in the physical robot, the controlled parameter on the motor is

acceleration, which is powered by the Pulse Width Modulation (PWM) signal for the electrical speed controller. This transfer relationship can be solved by applying the time frame to convert the required correct value for these parameters.

4.1.3 Embedded design

The embedded design in the ROS system is separated into 3 layers: the application software layer, the system software layer and the hardware layer. In the simulation stage, the program, hardware commend, and program structure differ from the physical robot since the simulator programs stay in the operating system and require no communication.

The first layer, the application software layer, is the ROS server initiation and path tracking. In this layer, the program will call sensors and collect data from sensors to publish in the ROS server and process the data to provide a wheel power signal as output to the next layer. Raspberry Pi manages this program since Pi has higher computing power and can carry programs under the Linux operating system.

The second layer is the system software layer. In this layer, the actuator driver/controller is managed by the Arduino UNO.

The third layer is the low-level motor control. This layer will inflict the result caused by the simulated motor output and demonstrate virtually in a 3D-rendered simulation. In the simulation stage, the third layer uses the built-in driver for the motor control, so no effort is needed in the simulation. In the live testing stage, the Arduino UNO and VESC communication uses a PWM signal through General-purpose input/output (GPIO) port.

The first and second layers would be similar in the experiment stage of running a physical robot. The third level layer requires building a new network to enable the control of the motors.

4.1.4 Kinematic design

The kinematic model in the simulation contains the slipping condition by adding the ground surface roughness and wheel surface friction value. The condition is based on the experiment workspace environment: concrete ground surface and rubbered wheel. The slipping condition is considered based on the following parameters:

Table 6 Surface Roughness Coefficient

Surface	Absolute Roughness Coefficient
Ordinary Concrete	$(0.3 - 1)\text{x}10^{-3}m$ Using an average of $0.65\text{x}10^{-3}m$ as value [18]
Rubber, Smoothed	$1\text{x}10^{-2}m$ Using an average of $1\text{x}10^{-2}m$ as value [21]

All other kinematics conditions applied are based on studying the differential drive-controlled robot to ensure the motion is fully understood.

4.1.5 Simulation Concept

It requires the creation of nodes for simulation in the Operating System Linux, using the Linux-based ROS-compatible simulation software Webot to achieve the

simulation result. All the simulation sensor readings will add gaussian error to simulate practical situations based on the same accuracy of UWB sensors.

Creating nodes in the simulation:

Node 1: robot model, publish P_x and P_y, the subscriber of u_1 and u_2.

Node 2: controller, the subscriber of P_x and P_y, publish u_1 and u_2.

The subscriber updates the u_1 and u_2. The node 1 publisher has 1000 Hz to ensure the accuracy of resolution. It will be updating the P_{xtrue} and P_{ytrue} for 1000Hz. Timer 2 would simulate the UWB localization sensor of 10 Hz and publish the real-time reading of P_x, P_y.

$$P_x = P_{xt} + E_x$$

$$P_y = P_{yt} + E_y$$

It adds a position sensor to represent the UWB sensors in the simulation. The sensor used is the GPS sensor node, a standard add-on in the simulation software. The GPS sensor node can set up the accuracy to present simulated results from UWB reading on the same route.

Based on the experiments, on the same coordination reference frame and pathway, the UWB real-time reading has a 4% of error, which is close to 0.2 meters error. This error was measured based on the UWB set-up in the 6 square meters fixed working space. The GPS sensor nodes were set to 0.2 meters in accuracy in the simulation.

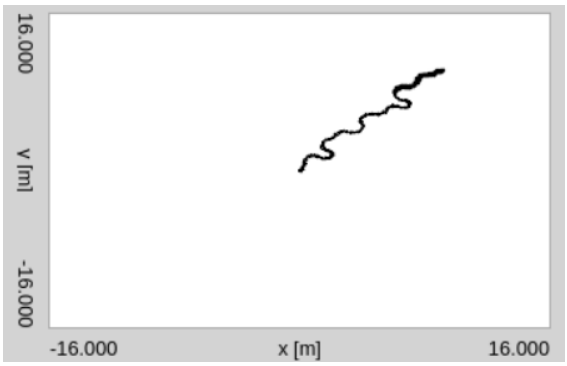

Figure 19 Early simulation testing for robot motion

Understanding the K value control will help the tunning work progress in the PD tuning process. In this situation, finding the K value through calculation is impractical since there is no method to determine the system response. Manually tunning work is required. The critical coefficient value tunning also brings the limitation of the PD-controlled mobile robot to the table: The tested PD value can only fit the controlled environment where the PD value was manually tested and selected. The tuned parameter fits a robot that works in a controlled environment only. If the robot is going to navigate into a new environment that brings new elements that will impact the motion, further manual testing will be required. The early simulation result is shown below:

In this simulation, the following interference is not part of the simulation:

- Wind blowing interference.
- The slop of the ground interference

4.1.6 Linux-Based Simulation Result

In the Linux-based Simulation, the parameters aim to stay at the same value as the field experiment. All significant digits will be kept as 4 digits in the simulation recording. The PD value was *phi_P=1. 5*, *phi_D=0.4* and Y_m*_P=1.5*, Y_m*_D=0.2*.

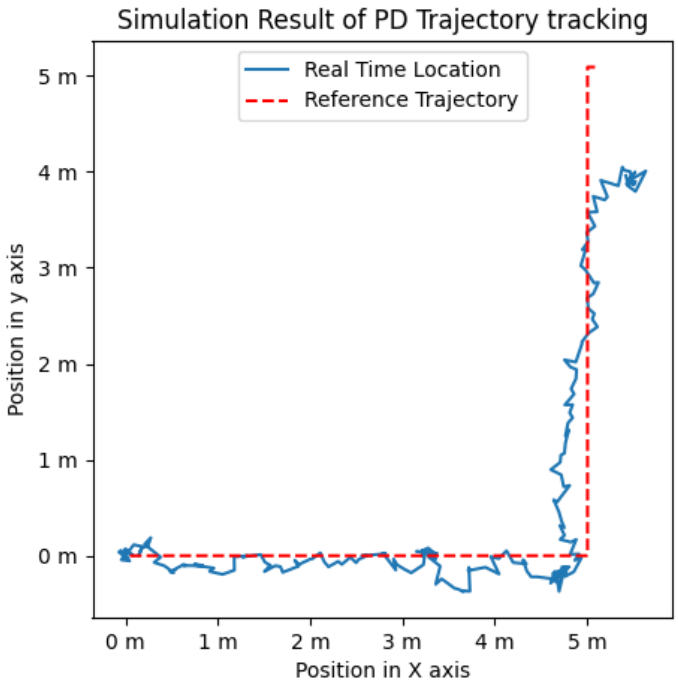

Figure 20 Simulation result of PD trajectory tracking

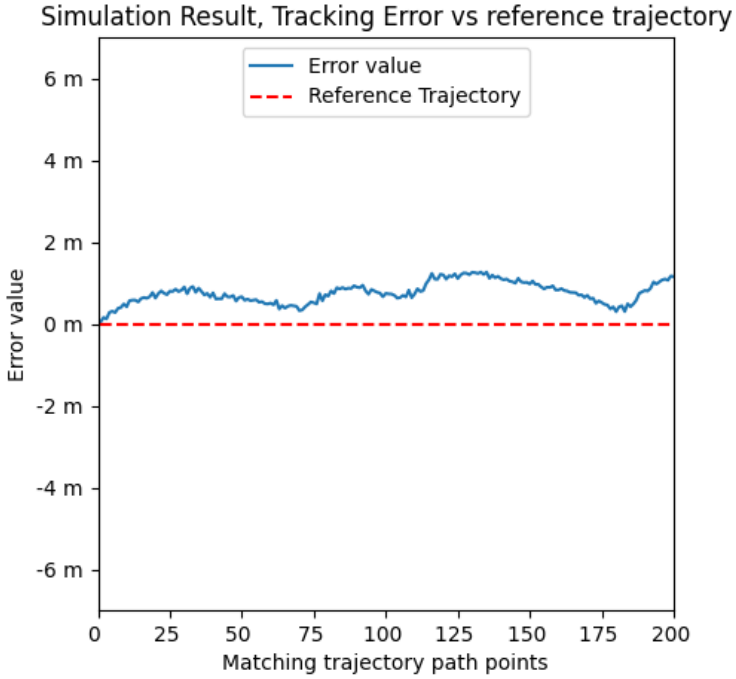

Figure 21 Simulation error result of PD trajectory tracking

The simulation result shows that the Euclidean distance maximum error value is 1.262 meters, and the average error value is 0.755 meters.

4.2 Field Experiment

4.2.1 Mechanical and hardware development

The mechanical setup can be referenced to the previous robot 4 for field experiment purposes.

The hardware setup on the robot is shown as follows:

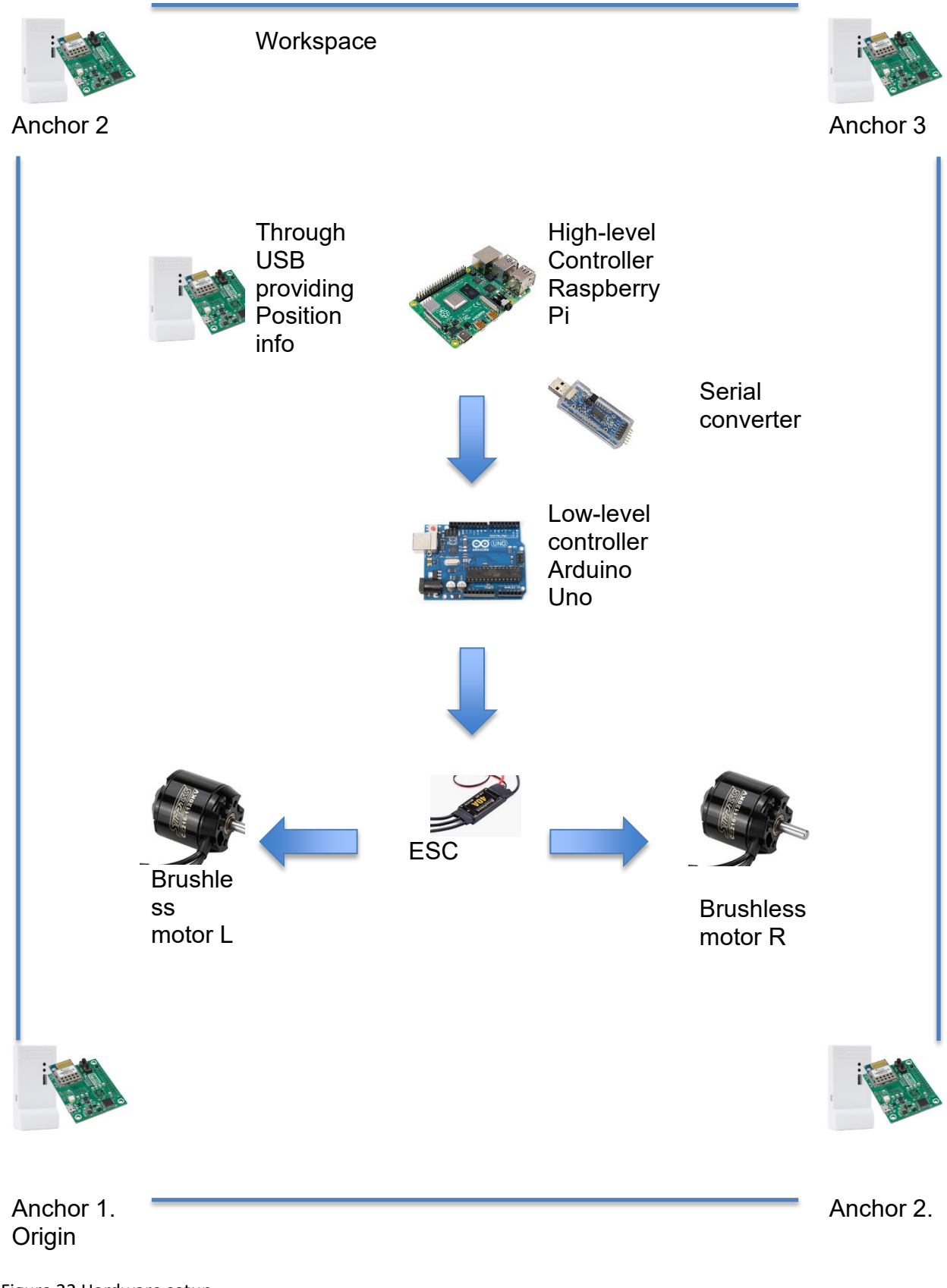

Figure 22 Hardware setup

The following hardware electronics component list is what is needed for running the robot in the live testing at a minimum:

- Raspberry Pi (need 1, High-level control)
- Arduino (need 1, Low-level control)
- Motor ESC module (need 2, motor power control)
- Serial communication USB plug (need 1, install on Raspberry Pi) [22]
- UWB tag (need 1, install on the robot)
- UWB anchors (need 4, install on working area corners)
- Jump wires and other misalliances.

4.2.2 Serial communication and State-machine

In the communication process between serial ports, while transmitting the serial information regarding the motor control data, the state machine must identify the desired value based on the characteristics of serial communication.

In serial communication, data is sent sequentially transmitted through the channel. Sequential serial data is a long serial bit that contains information that needs to be identified by the receiving program to match the correct bits for information reading. In this case, a state machine is required to read the section of the desired ordered data bits by identifying the start and stop elements. An example can be found in the following reference figure as a segment of serial communication. [22]

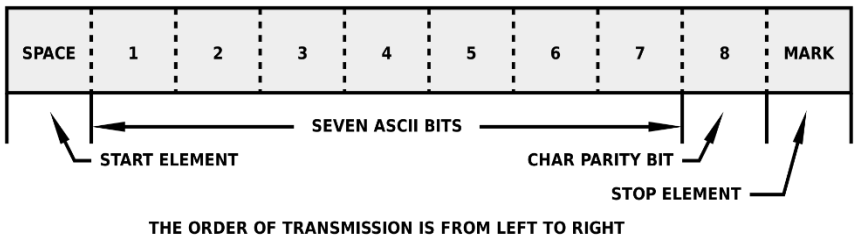

Figure 23 Serial communication layout

The state-machine logic diagram is attached below to demonstrate the Arduino program's serial communication reading flow process while receiving motor commands from Raspberry Pi.

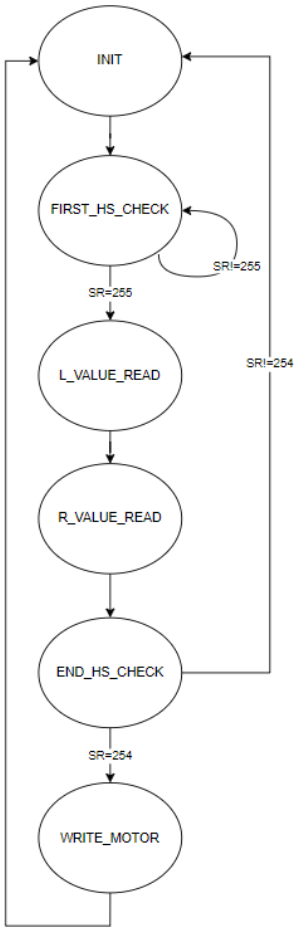

Figure 24 State machine logic layout

The state machine will follow the logic to identify the reading and process the motor command in low-level control.

4.2.3 Control System changes on physical experiment

In the physical experiment stage, the control system also needs to adapt to the changes from the simulation. The main differences are in the following aspects.

1. Time

In the live testing, the latency concern should be considered in the control program development because the operating system of Linux is a soft real-time system. The soft real-time system will experience latency during the program executions. A real-time difference calculation program is required to prevent the influence of unpredictable latency, which may cause the overwrite of the motor control signal while the desired motor control signal period has not yet. The time difference calculation program will use the hardware clock as the time anchor, ensuring each control iteration is executed without interruption. Each iteration of trajectory tracking action in the program will update a counter and the time anchor. [23]

2. Device to Device communication

The data and programs are running under the same system in the simulation. The server of the ROS system will receive published data from the location tracker sensor node and make it available for a control program to read within the single operating system.

In the live testing situation, the sensors are out of the main device running the control system. It requires the UART communication between UWB sensors and the low-level motor control Arduino. Considering the device's compatibility in future development, using the common USB port is easy as it can be established through the

UART communication port. However, this requires additional hardware, the USB port adaptor with UART communication to convert the USB port to the UART port.

4.2.4 Field Experiment Result

4.2.4.1 Dry Field Experiment Result

The field experiment has been performed in a designated testing area for trajectory tracking control. Eight models are being used as the representation of control results. In these models, six experiments are performed on solid asphalt ground and two on wet solid asphalt ground. In the collected experiment results, the comparison study focused on the real-time locations with reference trajectory and location error analysis.

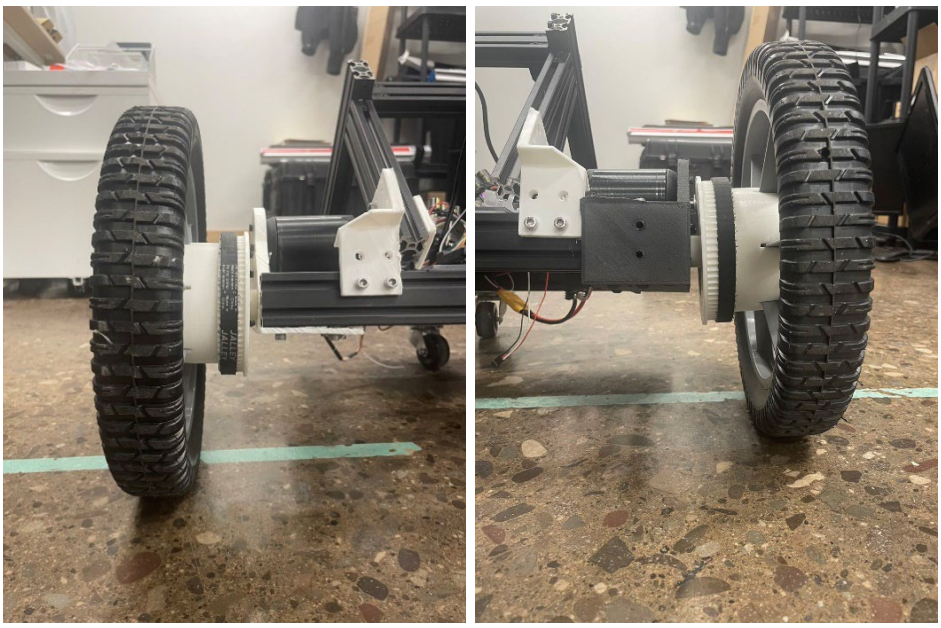

Figure 25 Left and right wheel detailed picture

It should be noted that the left and right wheels will not provide the same power since the 3D printed gear train will sustain different friction and experience unexpected behaviour, as reflected on the field experiment result graphs. However, the control system will provide correction tracking to follow the reference pathway. The

unpredictable behaviour of the subframe will provide increased difficulty and error-building speed.

Figure 26 Field experiment picture

All significant digits will be kept as 4 digits in the field experiment recording. The field experiments result is attached as follows:

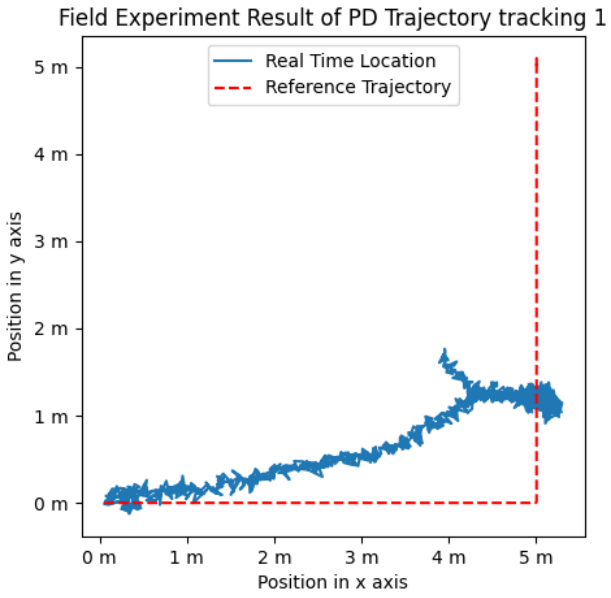

Figure 27 Field experiment Result of track 1

This figure is the field experiment 1 result with the default value (which is 1, 1) of the PD controller.

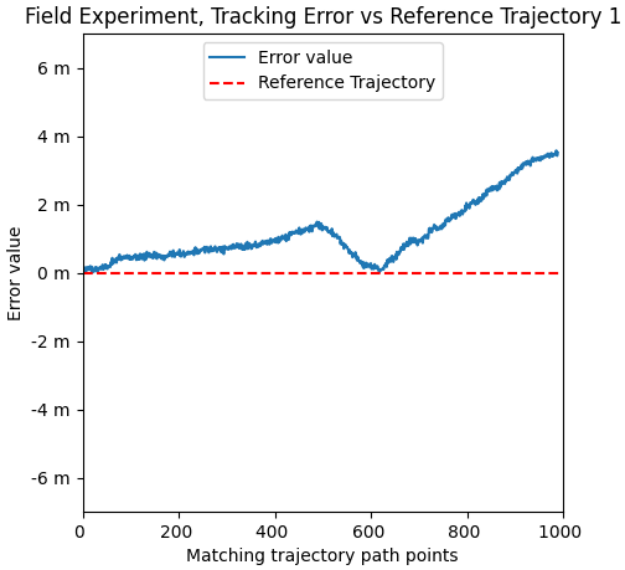

Figure 28 Field experiment error result of tracking 1

This figure is the field experiment 1 result of tracking error comparison of the reference trajectory. This result is the Euclidean distance to collect the error distance. It

shows that the maximum error value is 3.57 meters, and the average is 1.15 meters. The error value can be observed in the motion and divided into two sections. In the first section, the continuous error increases, indicating that the error is not converging. The error increases faster in the second tracking section than in the first section and is not converging. The error value between the two sections is caused by the 90-degree left turn angle that moves approaches the robot's real location, which had a direction error that moves left. The error graph would differ if the robot moved to the right offset instead of the left offset.

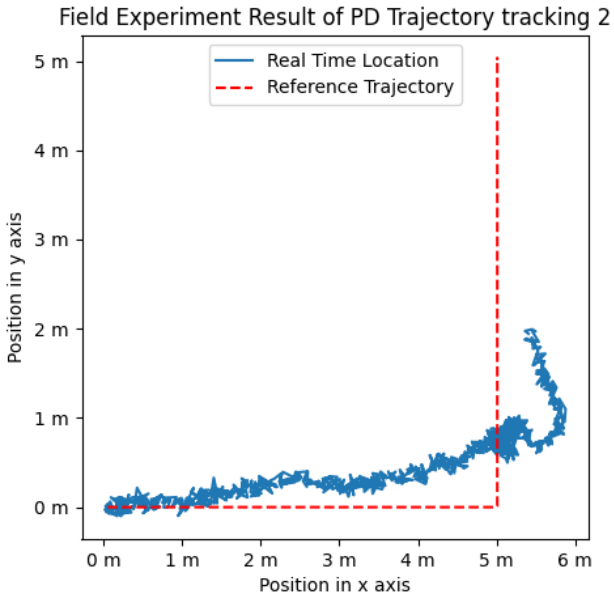

Figure 29 Field experiment Result of track 2

This figure is the experiment result with the value of the PD controller as the same value in the previous simulation. The PD value was *phi_P=1.5*, *phi_D=0.4* and, Y_m*_P=1.5*, Y_m*_D=0.2*.

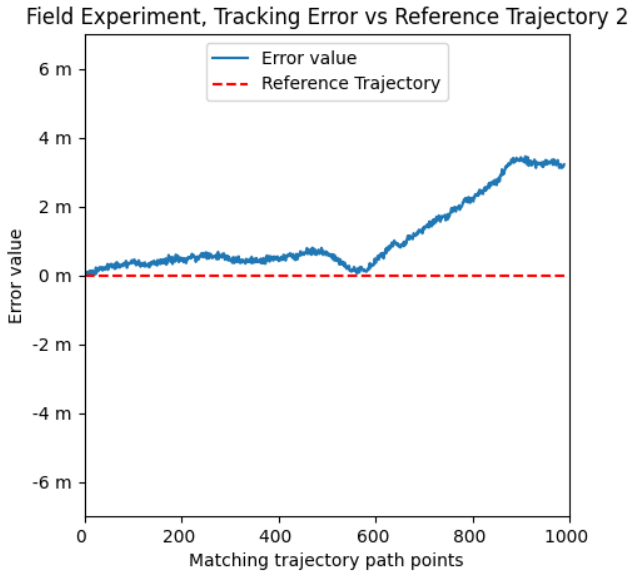

Figure 30 Field experiment error result of tracking 2

This figure is the field experiment 1 result of tracking error comparison of the reference trajectory. It shows that the maximum error value is 3.44 meters, and the average error value is 1.12 meters. The error value can still be divided into two sections. The first section controls the error better than the experiment 1 result. The error increment in the second tracking section is lower than in the last comparison. The error result figure shows the improved result by applying the new PD value in the control system.

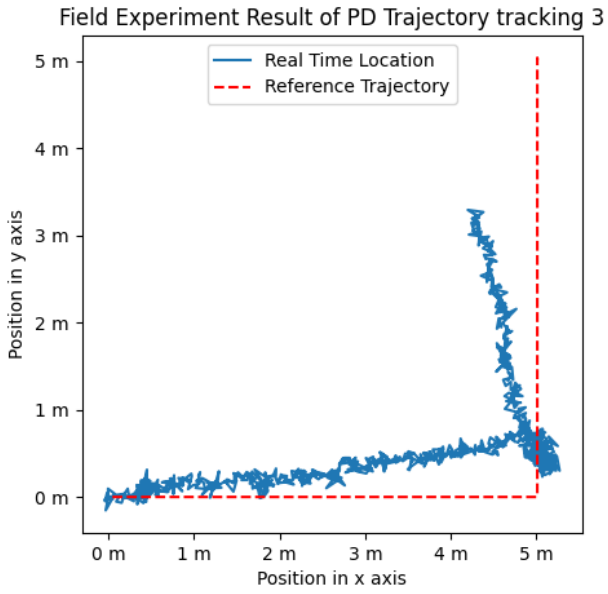

Figure 31 Field experiment Result of tracking 3

This figure is the experiment result with the initial tested PD value with better performance of the result. The PD value was *phi_P=2.25*, *phi_D=0.35* and Y_m*_P=3.75*, Y_m*_D=0.2*.

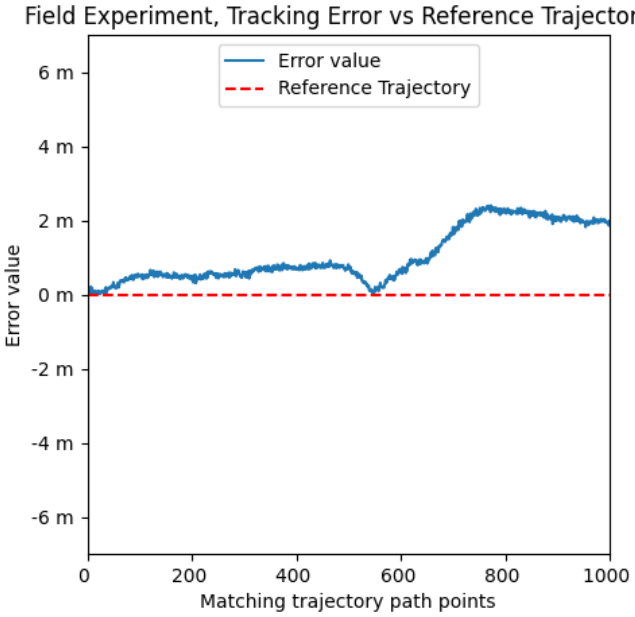

Figure 32 Field experiment error result of tracking 3

This figure is the field experiment 3 result of tracking error comparison of the reference trajectory. It shows that the maximum error value is 2.39 meters, and the average error value is 1.04 meters. The first section controls the error better than the experiment 2 result. In the second tracking section, the error increment is lower than in the last comparison, and the error was also reduced. The error result figure shows the improved result by applying the revised PD value in the control system, and the error shows a controllable trend.

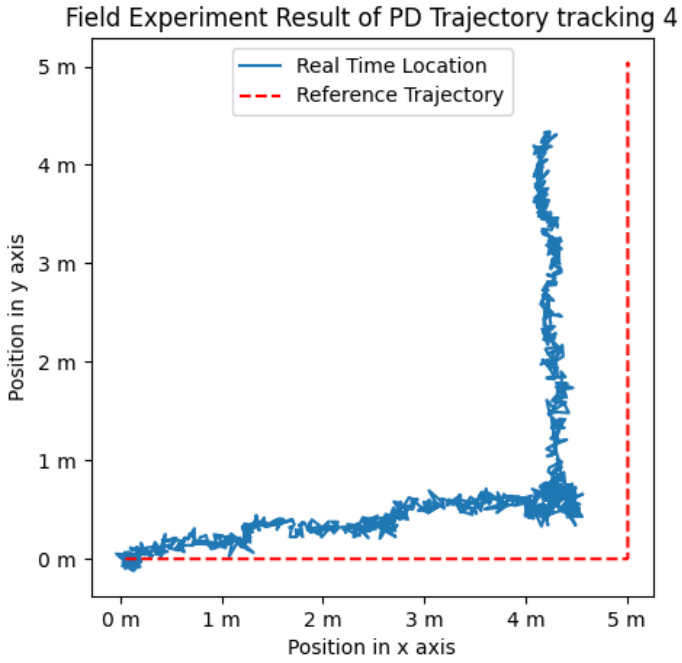

Figure 33 Field experiment Result of tracking 4

This figure is the experiment result with the tested PD value with better performance. The PD value was *phi_P=2.25*, *phi_D=0.325* and Y_m*_P=3.55*, Y_m*_D=0.175*.

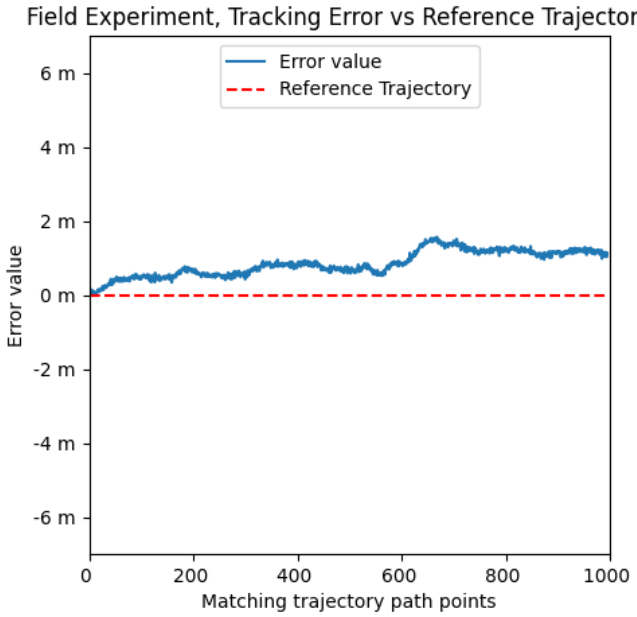

Figure 34 Field experiment error result of tracking 4

The field experiment 4 error result of tracking error comparison of the reference trajectory. It shows that the maximum error value is 1.56 meters, and the average error value is 0.84 meters. The maximum error value has been decreased, and the overall error trend converges into a similar value. The first section controls the error better than the experiment 3 result. In the second section of the tracking record, the error tends to stay like the first section of the error. The error result figure shows the improved result by applying the revised PD value in the control system, and the error indicates a better-controlled trend.

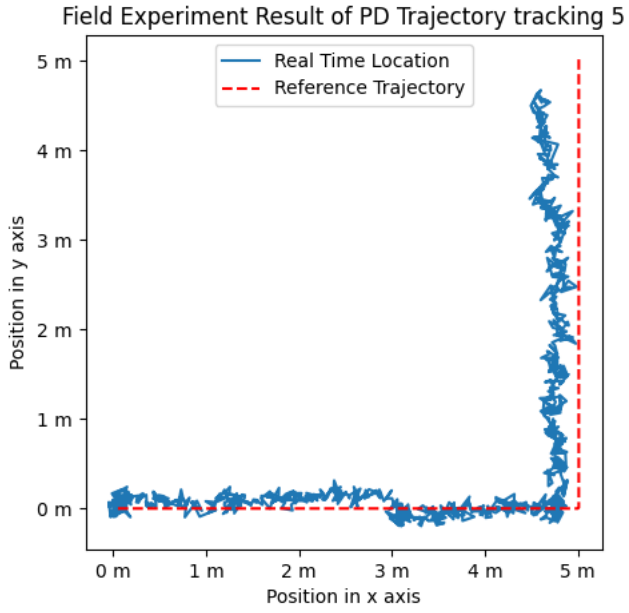

Figure 35 Field experiment Result of tracking 5

This figure is the experiment result with the final tested PD value with the best performance of the result. The PD value was *phi_P=2.55*, *phi_D=0.325* and Y_m*_P=4.75*, Y_m*_D=0.180.*

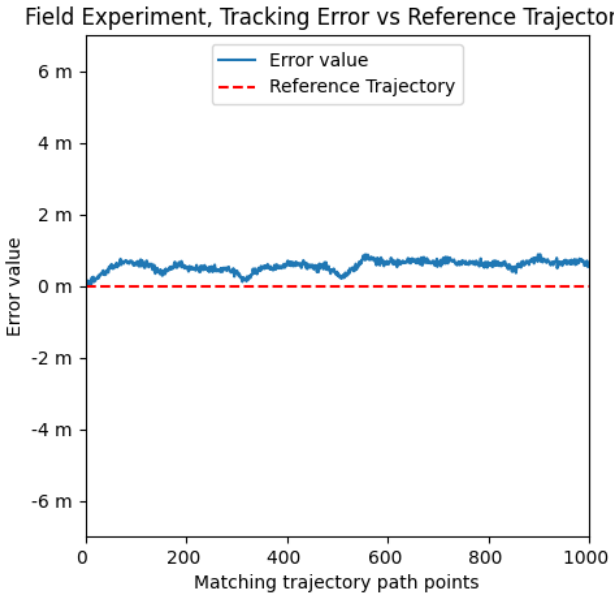

Figure 36 Field experiment error result of tracking 5

This figure shows the experiment results with the new PD value with the best performance. It shows that the maximum error value is 0.87 meters, and the average error value is 0.55 meters. The maximum error value has been decreased under 1 meter, and the overall error trend converges and stays stable. The first section controls the error better than the experiment 4 result. In the second tracking section, the error shares the same error behaviour as the first. The error result figure shows the improved result by applying the revised PD value in the control system, and the error shows a steady result.

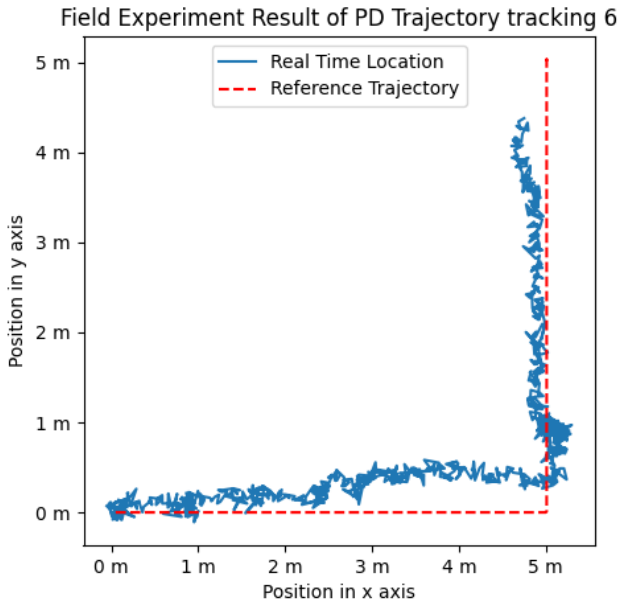

Figure 37 Field experiment Result of tracking 6

This result shows a promising steady trajectory tracking robot platform that can navigate and follow the designated tracking path under the same control system.

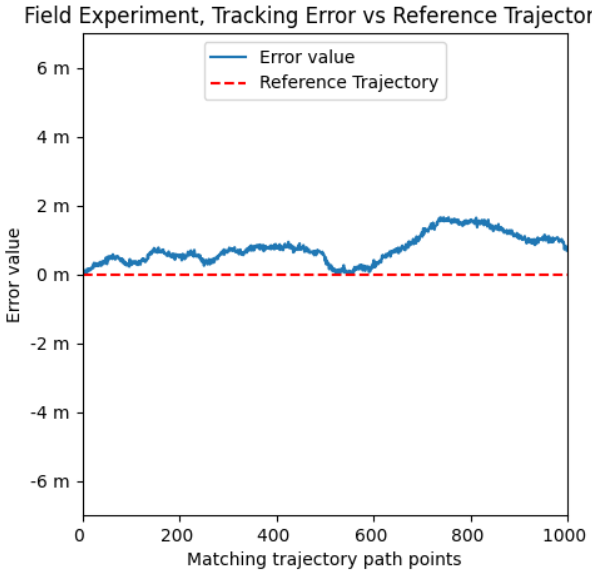

Figure 38 Field experiment error result of tracking 6

This figure is the field experiment 6 result of tracking error comparison of the reference trajectory. It shows that the maximum error value is 1.65 meters, and the average error value is 0.74 meters. The maximum error value is larger than experiment 5, but the average error value remains like that of experiment 5. This maximum error value may cause by unreliable subframes or slipping of the gear train during the tracking motion. Through the observation of the tracking error, in experiment 6, the tracking is doing worse than in experiment 5 with the larger average and maximum error value.

4.2.4.2 Robustness testing, conditions changed in the wet field experiment.

In this wet field experiments, there is no testing nor measurement of the level of ground wetness. These wet experiments were performed on a random rain day. The level of ground wetness will be assumed to meet the previous wet pavement coefficient. Which is 0.4, in this study.

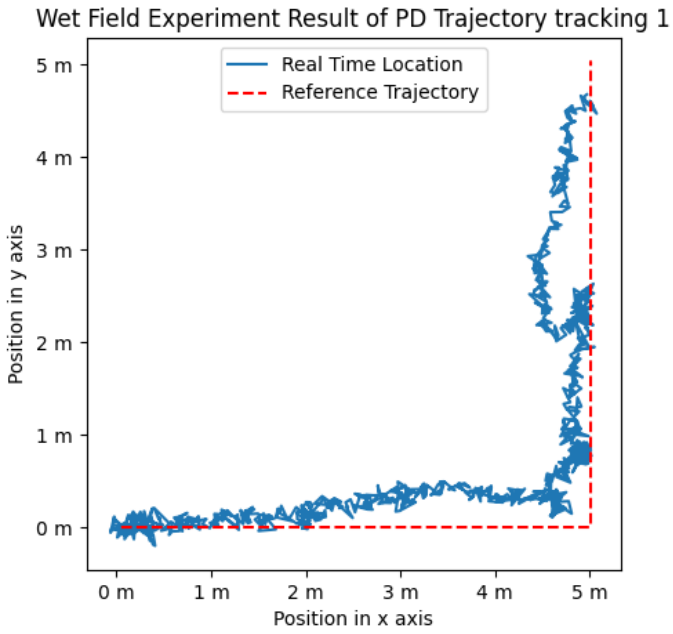

Figure 39 Wet field experiment Result of track 1

This figure is the experiment result under the same PD value of the controller but on the after-rain wet ground environment. There is one interesting spot around during the robot heading north near the end of the tracking. The turning around/turning back behaviour indicates that the robot's motion was losing control due to slipping or speeding and that the robot's actual location is beyond the designated tracking path point. This issue will cause the robot to turn around and find the past path point. Since there is no reverse logic in the control system, the robot will turn around. However, the average frictional coefficient changes for rubber on dry concrete is 0.6 to 0.85, and rubber on wet concrete is 0.45 to 0.75. The difference in the frictional coefficient may not be big enough that the control system is still applicable and robust under the manageable difference.

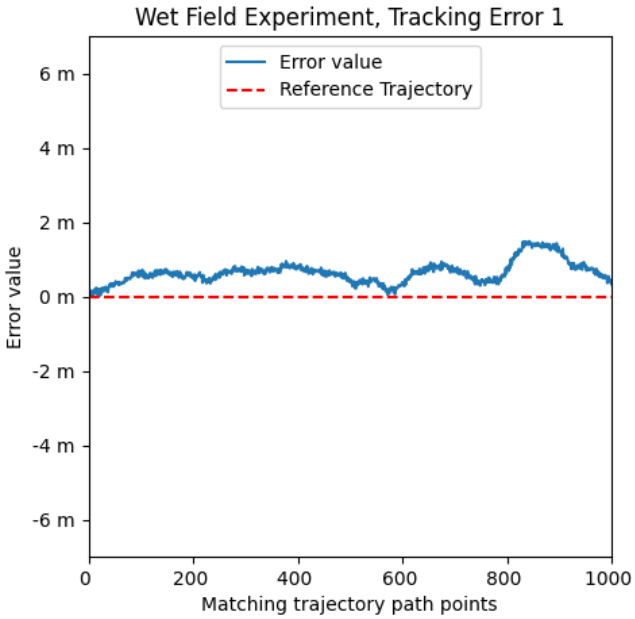

Figure 40 Wet field experiment error result of tracking 1

This figure is the wet field experiment 1 result of tracking error comparison of the reference trajectory. It shows that the maximum error value is 1.48 meters, and the average error value is 0.64 meters. The error level remains similar level as in the field experiment 6. The wet field experiment shows that the control system performs similarly if the ground roughness or friction coefficient condition is changed.

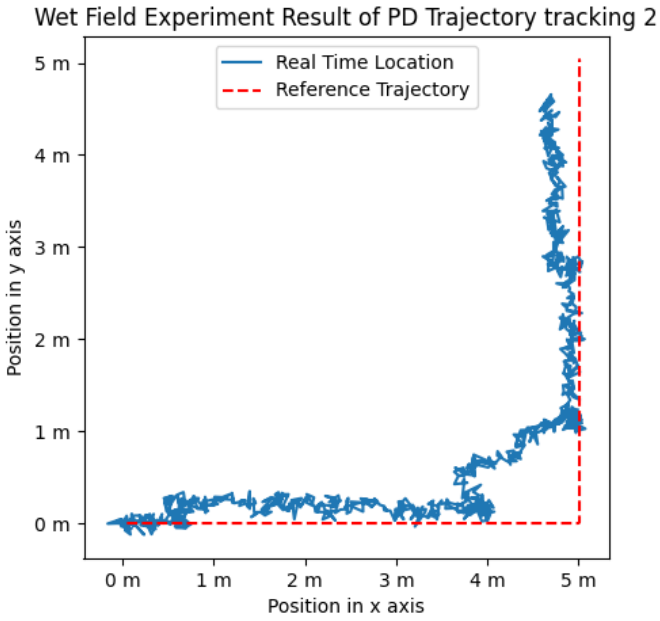

Figure 41 Wet field experiment Result of track 2

This figure shows the wet field experiment 2 result under the same controller but in a different environment. This result provides the sense of reproducibility to justify the control system performing similarly in the after-rain wet ground in a different controlled environment.

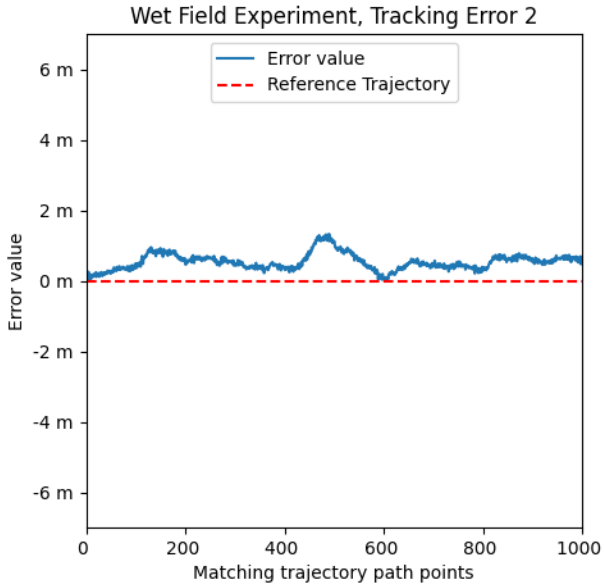

Figure 42 Wet field experiment error result of tracking 2

This figure is the wet field experiment 2 result of tracking error comparison of the reference trajectory. It shows that the maximum error value is 1.33 meters, and the average error value is 0.53 meters. The error level remains similar to the other controlled results using the same value of the PD controller.

4.3 Performance and repeatability analysis

This section will analyze the result of the repeatability and error study.

The first analysis will be the controller improvement for the mobile robot's overall trajectory tracking performance. From figure 42, it is observable that the error was reduced, and the error remains at a stable level.

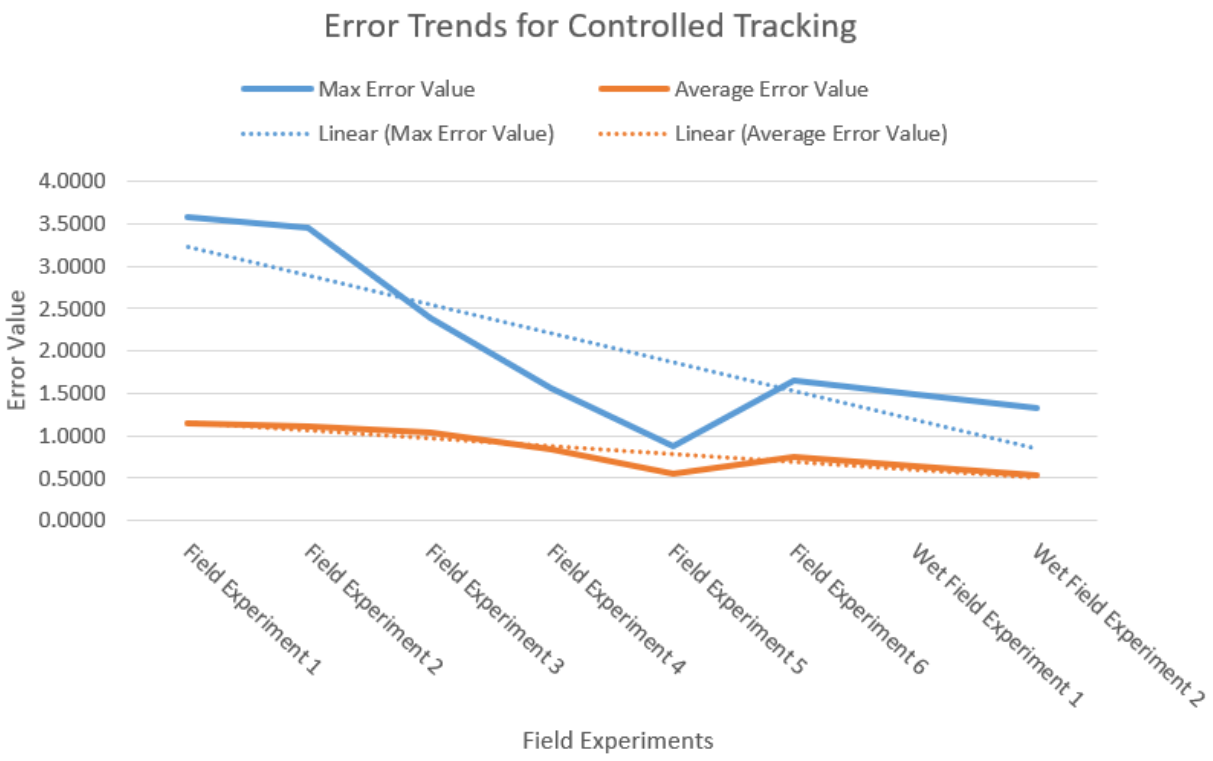

Figure 43 Error trends for controlled tracking overview

The error trends for the controlled tracking figure show that the max and average error values for Euclidean distance error in these 8 field experiments are reduced to low values with the improvement of the PD controller. However, in experiment 5, the result may be accidentally too promising that the error value is small. This result is not representative. Experiment 6, wet experiment 1 and wet experiment 2 share the regular deviation of error value under the same controller. In addition, the following table shows the error range based on the previous average error:

Table 7 Experiment and average error value

Number of Experiment	Average error value
1	1.15
2	1.12
3	1.04
4	0.84
5	0.55
6	0.74
7 (Wet 1)	0.64
8 (Wet 2)	0.53

The PD controller remains the same for experiments 5, 6, 7 and 8. Interestingly, experiments 7 and 8 were tested under wet ground conditions, but the average error value still fell into the range of 0.2 m which was the error of the locational measurement in the previous error observation.

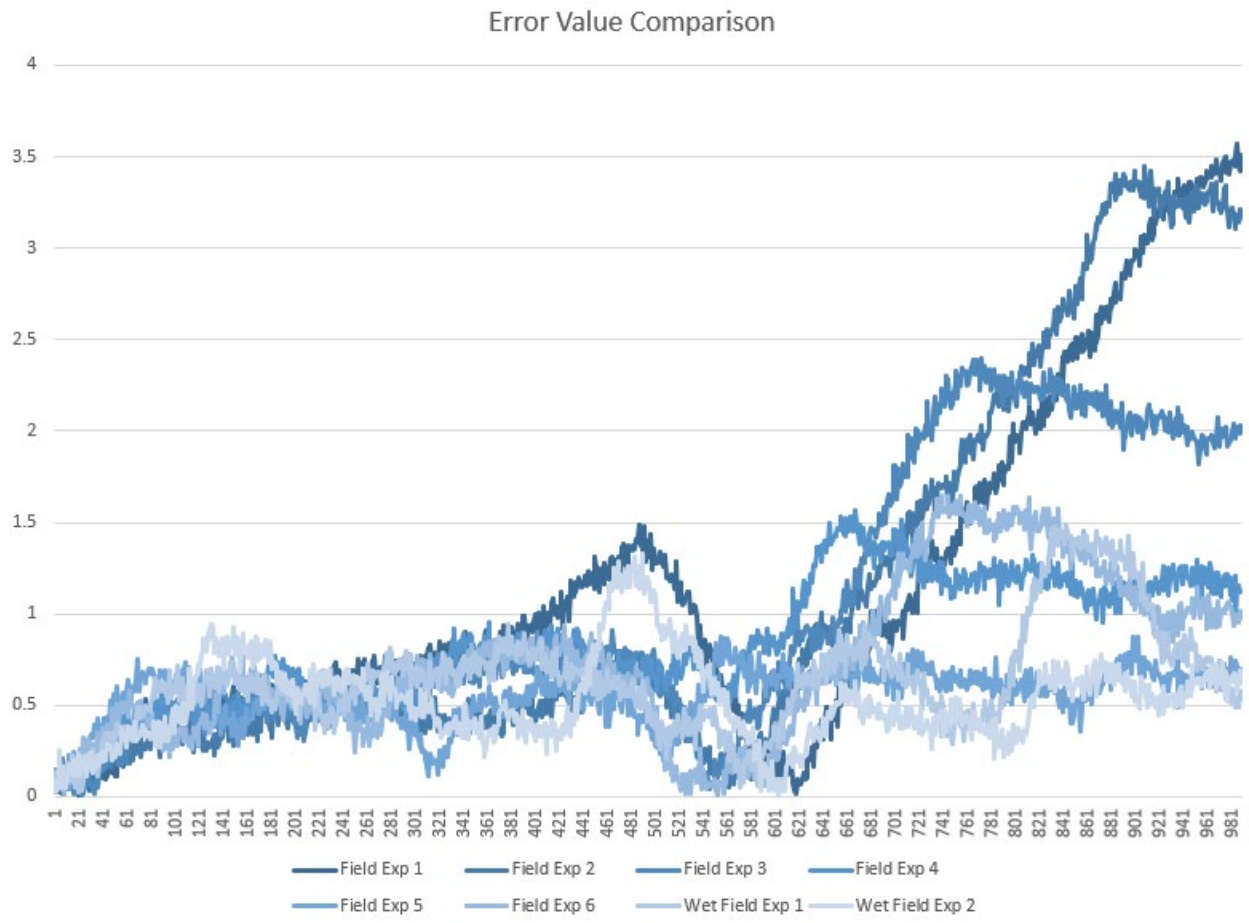

Figure 44 Error value comparison for 8 results.

From the error value comparison figure, the experiment error maps are distributed from deep colour to light colour to indicate the tuning of the PD controller. It clearly shows that the tuned PD controllers are smaller and more stable than the untuned PD controllers.

In the validation of the repeatability, the experiment result should be similar under the same conditions: the same controller and working environment. Under the same experimental area and all other parameters, a similar effect should be observed using the same PD value for the control system. Repeatability and reproducibility are important in this development since the design scope is to create a platform in any testing space with a different control system that other engineering teams can test.

Hence, another team can assemble a robot with the same hardware and control system to build this robot framework for testing results.

5. Conclusion

The focus of developing the differential drive wheeled robot trajectory tracking platform is to show stable and repeatable results for control system validation. The platform's development work requires research and development work combining the effort on building an embedded system, practical robot mechanical design and application of control systems. The study of finding a way to link hardware with the control system shows that middleware software is excellent for controlling and servicing the platform.

The robot dynamic models are being studied and represented in the control system. In the different generations of robot platforms, the robot platform's design trend is heading to the scope of simple control mechanics, lighter structures and modularized designs. The robot's movement conditions on the working area surfaces are also discussed to ensure all motion conditions are considered in the control program.

The early-stage simulation plays an important role to validate the capability of system design and the algorithm of the PID control system. The simulation successfully predicted the system behaviours and provided easy access to the simulated result for analysis and the modification capability for control system improvement and optimization. The transition of the operational system for robot control from the

simulation results requires an understanding and knowledge of the embedded system, device-to-device communication, and direct motor control.

During the live field experiment, it is notable that the current control system requires manual PID controller tuning to fit the new environment, including changing the robot infrastructure and operating environment conditions. The tuning work will benefit the control system that fits the new environment and provide a repeatable result based on the same trackable designated trajectory pathway. The experiment results showed that the control system successfully achieve the design objective.

This development successfully combined the one control theory into the robot system, enabling the capability of tracking arbitrary trajectories in the designated working area with limited sensor feedback. The repeatability also verified that the robot could operate and perform regularly in the testing field. By switching the modular control program, the platform can be driven by different control programs and navigate the designated working space for control theory validation.

6. Reference

[1] J. P. Hespanha. A. Pedro Aguiar, "Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles with parametric Modeling Uncertainty," *IEEE TRANSACTIONS ON AUTOMATIC CONTROL,* vol. 52, no. 8, 2007 August.

[2] S. Dixit. S.Fallah U.Montanaro. M. Dianati. A. Stevens F. Mccullough. A. Mouzakitis "Trajectory Planning and Tracking for Autonomous Overtaking: State-of-the-Art and Future Prospects," *Annual Reviews in Control,* vol. 45, pp. 76-86, 13 March 2018.

[3] A. De Luca &. B. Sciliano, "Trajectory control of a non-linear one-link flexible arm," *International Journal of Control,* vol. 50, no. 5, pp. 1699-1715, 1989.

[4] S.Vera. F.Petric. G.Heredia. A.Ollero. Z. Kovacic, "Trajectory Planning Based on Collocation Methods for Adaptive Motion Control of Multiple Aerial and Ground Autonomous Vehicles," *Control of Complet System,* pp. 585-634, 2016.

[5] R. C. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm," The Robotic Institute Carnegie Mellon University, Pittsburgh, 1992.

[6] F. Ke. Z. Li. C. Yang, "Robust Tube-Based Predictive Control for Visual Servoing of Constrained Differential-Drive Mobile Robots," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS,* vol. 65, no. 4, p. 3437, 2018.

[7] MathWorks, "What is Model Predictive Control?," MathWorks, 2022. [Online]. Available: https://www.mathworks.com/help/mpc/gs/what-is-mpc.html. [Accessed 15 12 2022].

[8] Creative Commons Attribution 3.0, "ROS Introduction," Creative Commons Attribution 3.0, 08 08 2018. [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Accessed 06 03 2022].

[9] Creative Commons Attribtuion 3.0, "Distribution," 15 05 2021. [Online]. Available: http://wiki.ros.org/Distributions. [Accessed 06 03 2022].

[10] Arduino, "Arduino UNO R3," Arduino, 2022. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3. [Accessed 16 3 2022].

[11] Cyberbotics Ltd., "Cyberbotics," Cyberbotics Ltd., 2023. [Online]. Available: https://cyberbotics.com/. [Accessed 05 12 2022].

[12] K. S. N. Leena, "Modelling and trajectory tracking of wheeled mobile robots.," *Procedia Technology,* vol. 24, pp. 538-545, 2016.

[13] D. G. M. Bone, "Differential-drive Mobile Robot Plant Information ver 2," Dr. Gary M. Bone, Hamilton, 2023.

[14] Britannica, The Editors of Encyclopaedia. , "dead reckoning," Encyclopaedia Britannica, 5 Aug 2019. [Online]. Available: https://www.britannica.com/technology/dead-reckoning-navigation. [Accessed 12 2 2023].

[15] I. ˇ. Gregor Klanˇcar, "Tracking-error model-based predictive control for mobile robots in real time," *ScienceDirect,* vol. 55, no. (2007)460-469, 2007.

[16] P. Pieter, "PID Controllers," 16 July 2021. [Online]. Available: https://tttapa.github.io/Pages/Arduino/Control-Theory/Motor-Fader/PID-Controllers.html. [Accessed 16 08 2022].

[17] ‡. S. B. M. S. Z. A. M. S. A. M. H. S. H. Umar Zangina†, "Non-linear PID Controller for Trajectory Tracking of a Differential Drive Mobile Robot," *Journal of Mechanical Engineering Research and Developments,* vol. 43, no. 7, pp. 255-270, 2020.

[18] MathWroks, "Pure Pursuit-Linear and angular velocity control commands," MathWorks, 2019. [Online]. Available: https://www.mathworks.com/help/robotics/ref/purepursuit.html. [Accessed 11 12 2020].

[19] MathWorks Student Competitions Team, "Mobile Robotics Simulation Toolbox," 15 Oct 2019. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/66586-mobile-robotics-simulation-toolbox. [Accessed 29 11 2020].

[20] M. X. Wu, "Differential Speed Steering Control for Four-Wheel Independent Driving Electric Vehicle," *International Journal of Materials, Mechanical and Manufacturing,* vol. 1, no. 4, 2013.

[21] Rahul Sharma K. D. Hone F.Duesk, "PREDICTIVE CONTROL OF DIFFERENTIAL DRIVE MOBILE ROBOT CONSIDERING DYNAMICS AND KINEMATICS," in *Proceedings 30th European Conference on Modelling and Simulation ECMS*, Regensburg, Germany, 2016.

[22] Engineers Edge, LLC, "Pipe Roughness Coefficients Table Charts | Hazen-Williams Coefficient | Manning Factor," Engineers Edge, LLC, 2023. [Online]. Available: https://www.engineersedge.com/fluid_flow/pipe-roughness.htm. [Accessed 1 12 2022].

[23] U. S. D. o. Defense, Military Standard: Common Long Haul and Tactical Communication System Technical Standards, Department of Defense, 1972.

[24] M. M. Madden*, "Challenges Using Linux as a Real-Time Operating System," NASA Langley Research Center, Hampton, VA, 23681, 2019.

[25] L.-C. L. Ching-Long Shih, "Trajectory Planning and Tracking Control of a Differential-Drive Mobile Robot in a Picture Drawing Application," *Robotics,* vol. 6(3), no. 2017, p. 17, 2017.