Tesla Who?
Developing an Autonomous Vehicle using PID Control to Race through WashU

January 2022 to May 2022

ESE 498/499 Capstone Design Project Final Report

Submitted to Professor Wang and the Department of Electrical and Systems Engineering

Advisor:

Dr. Dorothy Wang (dorothyw@wustl.edu)

Engineers:

Tom Goon (tgoon@wustl.edu), B.S. Systems Science and Engineering Candidate

Allison Todd (allisontodd@wustl.edu), B.S. Electrical Engineering Candidate

Caitlind Walker (caitlindwalker@wustl.edu), B.S. Electrical Engineering Candidate

Submission Date: May 4, 2022

Abstract

Our project revolves around the Electrical and System's Engineering department's semester common goal of navigating a track located in Brauer Arcade with a Pi Car. Equipped with LiDAR, Inertial Measurement Unit (IMU), and an encoder, our main focus was to implement PID control to maintain a consistent direction on straightaways and constant speed despite terrain. The LiDAR sensor is responsible for detecting a barrier signaling our PiCar to turn around a corner. For this track, there were three left turns. The IMU sensor is equipped with a gyroscope which returns angular velocity and is used to maintain a zero-angle on straightaways and turn. Finally, the encoder measures speed and will be used to control the speed of the PiCar. Integration via software is handled by Python and ROS.

1 Introduction

Autonomous cars can be found around almost every corner in technologically innovated areas. The streets of Silicone Valley are full of next generation technologies, but so are the sidewalks of Washington University in St. Louis. In the Electrical and Systems Engineering Capstone common project for Spring 2022, we will be building upon previous work to design, develop, and implement an autonomous vehicle that can navigate and race around a predetermined track through campus.

1.1 Background Information

Autonomous cars are the way of the future and everyone from Tesla to Google are investing in these capabilities. It is easy to imagine a safer work where drunk, distracted, and dangerous driving are a thing of the past by relying more and more heavily on autonomous systems. Simply put, autonomous systems are able to make decisions for themselves given inputted commands and sensing capabilities. Similarly to how a human behind the wheel of a car can take information such as where they want to go, how fast they are driving, and where they currently are to make decisions about the steering wheel placement and accelerator pedal placement, an autonomous control system can make these same decisions. Our group's previous experience in the Autonomous Aerial Vehicle Control Laboratory sets us up for success as we develop a control system that can do just that.

One of the most important factors in building a successful control system is accurate and reliable sensor data. Many autonomous vehicles in commercial development utilize LiDAR technologies to visualize a 3-Dimensional view of the vehicles surroundings. The spinning black box on the top of self-driving or autonomous cars are LiDAR sensors. Additionally, ultra-sonic sensors are a relatively reliable and easy to use sensor used to measure distance. Previously we have used ultra-sonic sensors to measure distance above the ground in drone control. Line-tracking sensors have been proven to work within the context of previous iterations of the ESE Capstone common project and will be a reliable source of navigation sensing. Additionally, tangential and angular velocities are important measurements that will need to be reliably collected. An encoder is a mechanical sensor that is used to count the number of gear rotations the car's wheels are experiencing and can be used to measure forward velocity. Lastly, an IMU containing both an accelerometer and a gyroscope can be used to measure acceleration in the x, y, and z directions as well as angular velocity around the x, y, and z axis. In order to utilize each of these sensors in tandem, the Robot Operating System, referred to as ROS, was implemented. ROS is an open-source software framework that connects nodes containing independent sensor code with each other allowing simultaneous use of sensor data in a primary node.

1.2 Problem Statement

The actual confines of the ESE Capstone common project are quite straightforward. We must design and build an autonomous vehicle that is capable of navigating quickly around a predetermined track (Figure 1). The exact solution to this problem statement was up to each group's discretion.

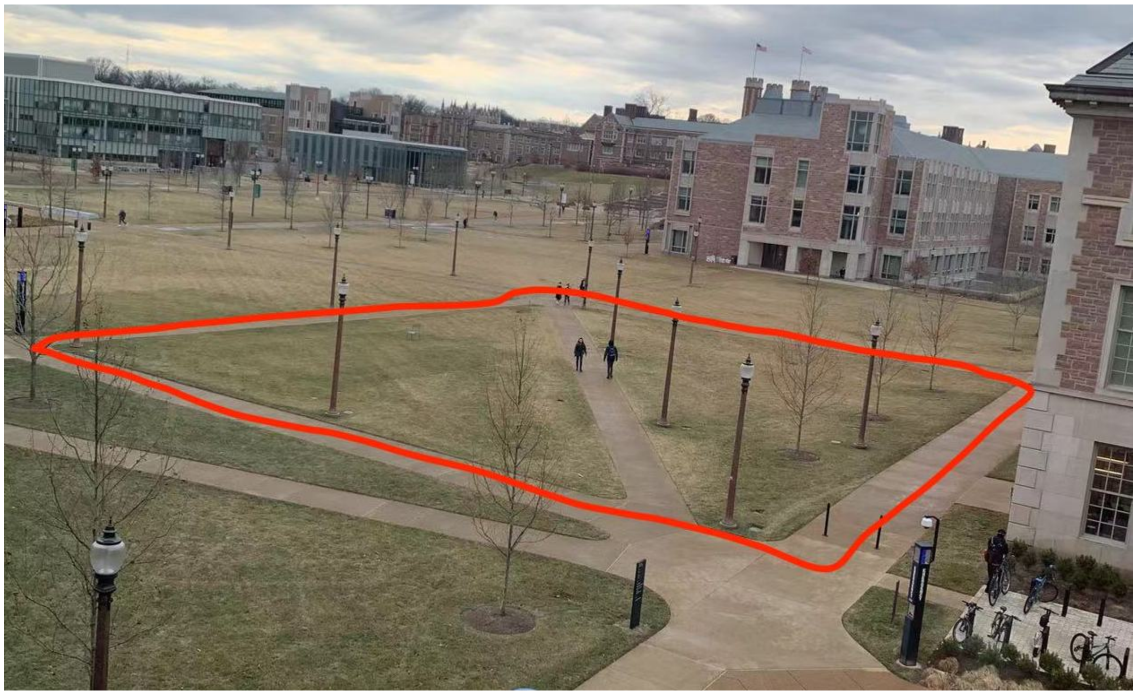


Figure 1: Autonomous Vehicle Race Track

1.3 Project Objectives

Ultimately, the objective of this capstone project is the successfully develop an autonomous vehicle that is capable of completing the prescribed track as defined in the problem statement. This will be done by first testing each of the desired sensor independently. Once tested, we will combine each of the sensors with ROS is order to make informed control decisions from these data points. These decisions will be developed using PID control. Our PID control will be used to command the forward velocity to a constant value, command the straight line angular velocity to zero, and minimize the angular error after each turn on the track. Because only left turns are necessary on this track, we will trigger a left turn using a distance sensor, so that the car begins its turn at an appropriate location. Once combined, we will be able to tune our system to allow for the most accurate racing possible.

2 Methods

$$e(t) = r(t) - y(t) \tag{1}$$

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{2}$$

Our approach to the problem is an Proportional, Integral, Derivative (PID) control system. This is control method operates on the error between the desired value and the actual value to determine the input to the plant (the car). The error equation is shown in Equation 1. The input then is found via a proportional term, a integral term, and a derivative term that each act on the error (Equation 2). In the proportional term, the control algorithm simply multiplies the error by the proportional gain value. The proportional path is the simplest form of control, but it fails to look at differences over time, which can lead to the system failing to follow the commanded value exactly. Thus, the integral control becomes necessary. In the integral term, the control algorithm integrates error over time and then multiplies this value by the integral gain value. While the integral path allows the control algorithm to act on error over time, it is a fairly slow process, so it does not act on the system quickly. To speed up this process, we add in the derivative term. In the derivative term, the control algorithm derives the error over time and then multiplies it by the derivative gain value. We determined the gain values via tuning. To tune the gains, we started with the proportional term and slowly increased this gain until the movement of the car became periodic. Then, we chose the gain as approximately 50% of the period gain value. We then repeated this process for integral and derivative control.

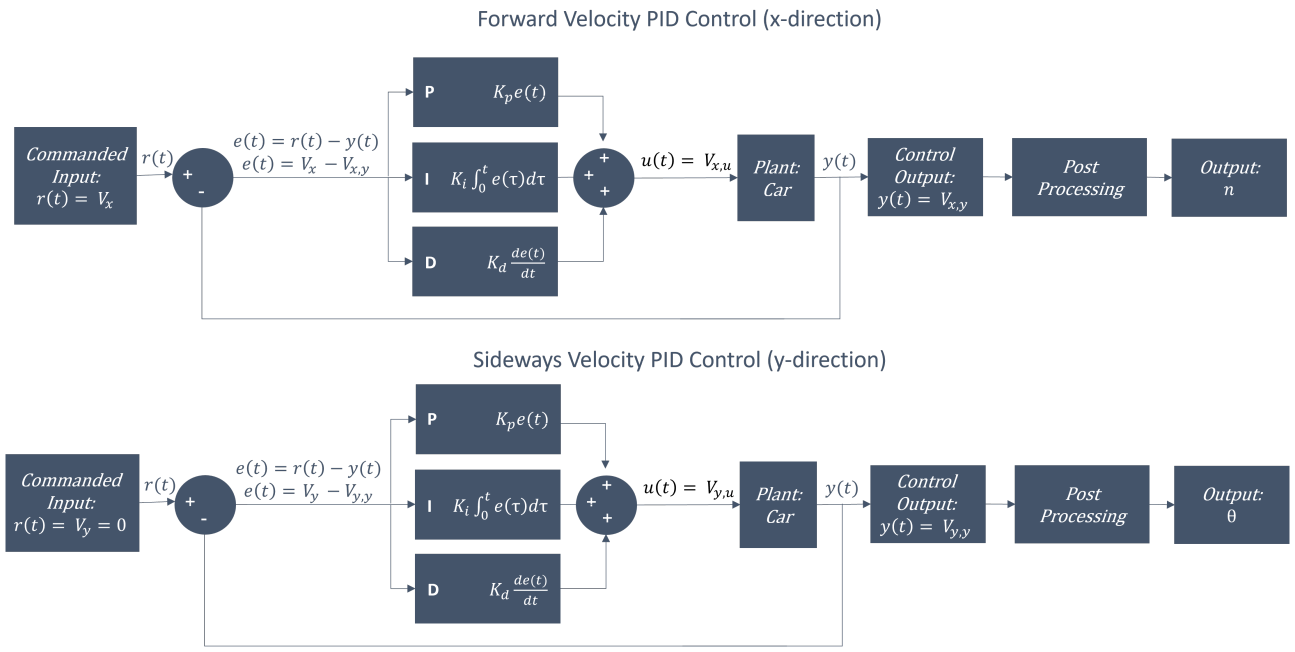


Figure 2: PID Control Methods for Forward Movement and Sideways Movement

Figure 2 shows the PID control methods for forward velocity (x-direction) and sideways velocity (y-direction). To ensure the car goes straight when moving on the course, we command the forward velocity to a constant value in m/s while holding the sideways velocity as zero. This allows the car to move forward at a fairly constant velocity, while preventing the car from drifting to the left and right.

To turn the car, the team decided on a hard-coded turn based on our data collection. By hard coding the turn, we were able to focus only on the PID controls shown above. Additionally, the focus on the forward movement and sideways movement PID controls allows the car to re-orient itself should a turn ever be imperfect. The car knows to turn based on LiDAR measurements. When the car gets within a meter of a barrier, as indicated by the LiDAR measurements, the hard-coded turn is initiated.

3 Data Collection

To test the feasibility of our approach, we sought to verify the sensors were properly functioning and accurate. We downloaded all software necessary and wrote test scripts for every sensor. Since our approach is dependent on PID control, we focused on testing 3 sensors: the ultrasonic as an indicator for when we should initiate a turn, the IMU for angular velocity measurements for avoiding drift and turning, and the encoder for forward movement.

3.1 Ultrasonic Sensor & LiDAR: Indication for Turning

3.1.1 Ultrasonic Sensor

We started with testing the ultrasonic sensor. A test script from Professor Feher's ESE205 class was utilized. From here, we used a tape measure to measure out set distances of 10 cm and 25 cm. Then we placed a box at those set points and ran our script 10 times each. While the time it for the sound wave to return may be important, we mainly focused on an accurate distance. The averages were respectively 11.715 cm and 26.011 cm with a standard deviation of 0.606 cm and 0.391 cm. Raw results shown in Table 1.

10 cm	11.07	11.40	11.40	12.28	12.85	12.05	11.08	11.06	11.98	11.98
25 cm	25.98	26.18	25.89	25.89	25.87	25.90	27.05	25.86	25.60	25.89

Table 1: Ultrasonic Sensor Test Data

Overall, the ultrasonic sensor was fairly accurate in its readings (always within 2 cm of the actual distance). However, these tests were only done with close distances. The accuracy of the ultrasonic sensor greatly degrades after 2.5 meters. We discovered this when we attempted full-scale testing of the car on the track. Due to the inaccuracies of the ultrasonic sensor, we chose to move to a LiDAR sensor.

3.1.2 LiDAR

The inaccuracies of the ultrasonic sensor at large distances led to the team pivoting to the LiDAR sensor. We performed tests on the accuracy of the LiDAR and found it be much more accurate with a significantly smaller standard deviation of measured values for a given distance. The LiDAR was initially tested in the lab prior to mounting it to the car and testing it outside on the track. While accuracy decreased when placed on the track, this was overcome by implementing a moving-average filter on the LiDAR values. After implementing both the LiDAR and the moving average filter, the accuracy of the system was high enough to function well as an indication for turning.

3.2 Inertial Measurement Unit (IMU): Angular Velocity

The angular velocity in the z-direction measurements were necessary for two portions of this project: avoiding drift while moving forward and hard-coding a turn. Prior to either of these uses, though, we tested the accuracy of the z-direction angular velocity.

To test the angular velocity measurements, we programmed the car to a constant turn angle and a constant speed. This resulted in the car moving in a circle for about 60 seconds, which allowed the car to gain a constant angular velocity over a long period of time. The complete data set, including the car ramping up for the first 4 seconds prior to constant angular velocity being achieved are shown in Figure 3.

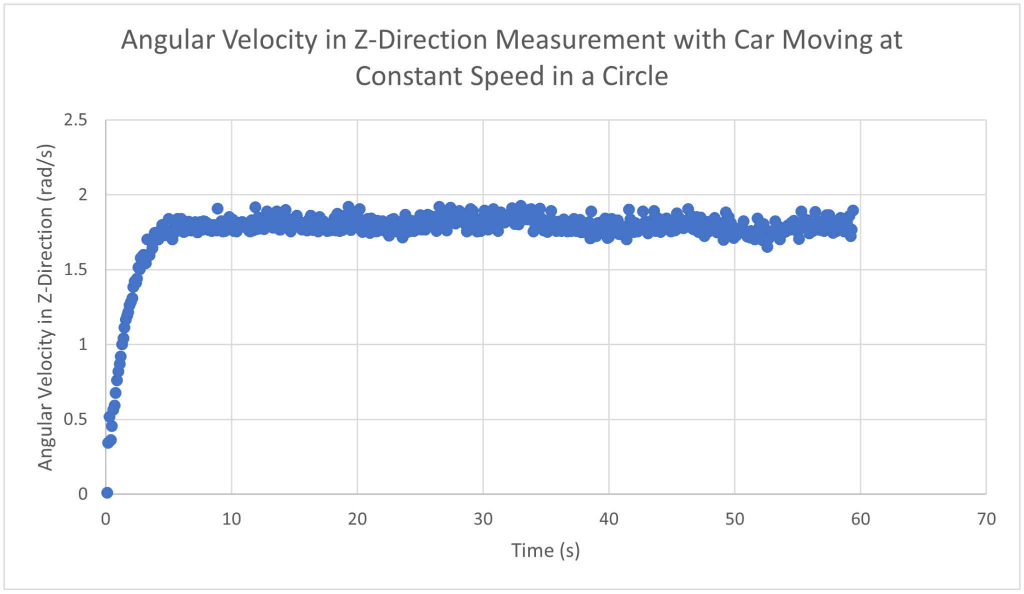


Figure 3: Angular Velocity in Z-Direction Measurement with Car Moving at Constant Speed in a Circle

Once the complete data set was graphed, we cut off the initial ramping-up measurements to find the average constant angular velocity in the z-direction. This data is shown in Figure 3. As shown in this Figure, the trendline indicates that the constant angular velocity is 1.8085 rad/s = 103.619 deg/s. Thus, the car should take about 3.47 s to complete a circle. This matches up with our timing of the car, which found the average time to complete a circle was about 3.5 s.

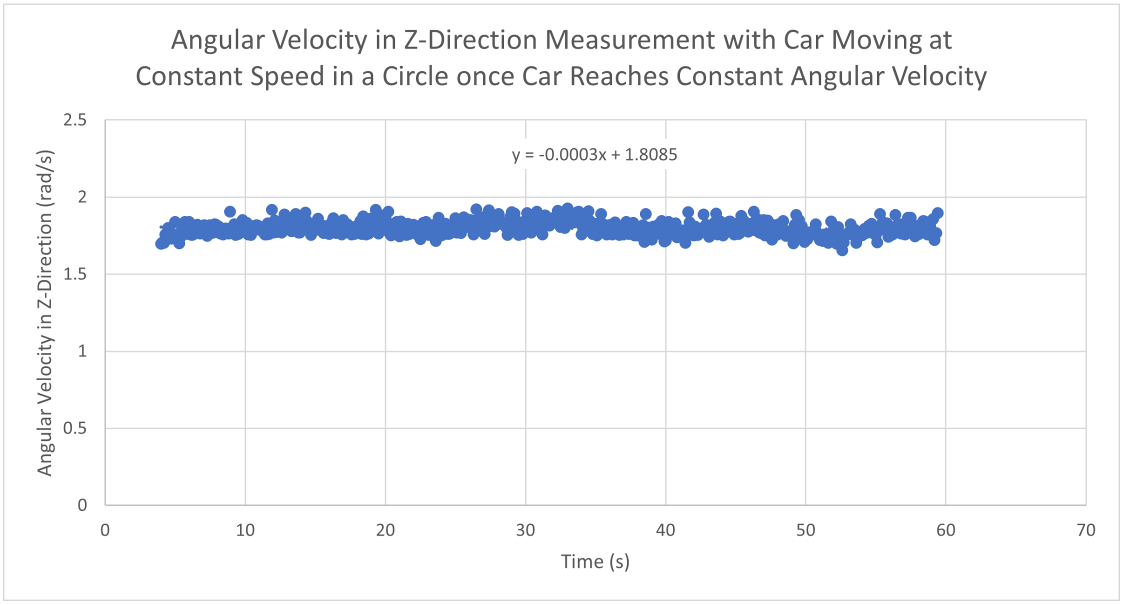


Figure 4: Angular Velocity in Z-Direction Measurement with Car Moving at Constant Speed in a Circle Once Car Reaches Constant Angular Velocity

This data was used to confirm that accuracy of the angular velocity measurements. This data proved that the IMU received relatively accurate angular velocity measurements, which allowed us to move ahead with use of the IMU in our control algorithms.

3.2.1 Hard-Coded Turning

Once it was determined that angular velocity in the z-direction was precise enough to use in PID control, we moved on to testing for a 90 degree turn. To test this, we held a constant speed of 0.160 and a constant time of 1 second and changed commanded steering angle. We tested commanded steering angles of 650 to 730 in increments of 10 (where 595 is going straight, 810 is a hard left turn). Then, we found the actual angle by integrating the angular velocity measurements during the turn.

An example data set is given to demonstrate this process. The data for a commanded steering angle of 700 is shown in Figure 5. For the commanded steering angle of 700, we integrated by taking the measurements during the turn (4.8 s to 5.9 s) and multiplied them by dt (0.1 s).

The results of varying commanded steering angle from 650 to 730 and the resulting angle are shown in Figure 6.

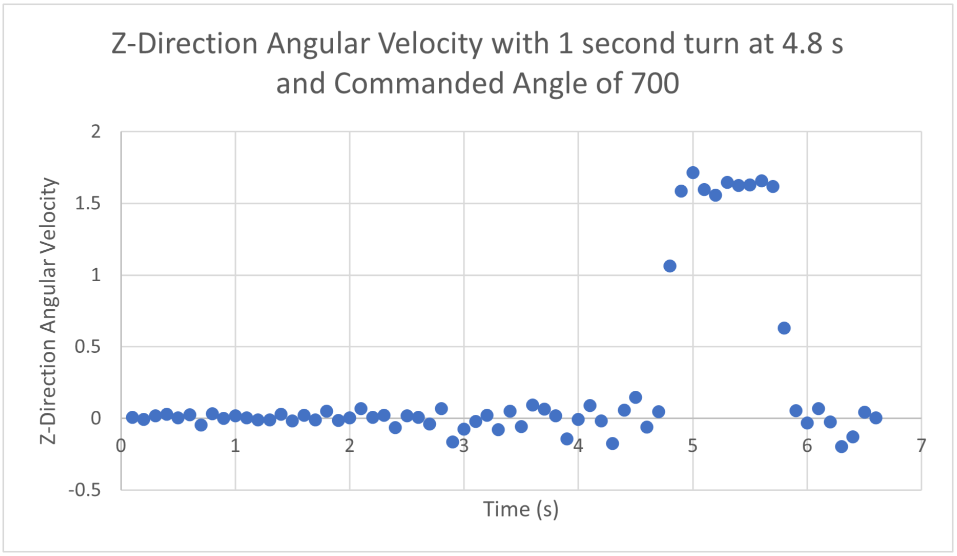


Figure 5: Z-Direction Angular Velocity with 1 second turn at 4.8 s and Commanded Angle of 700

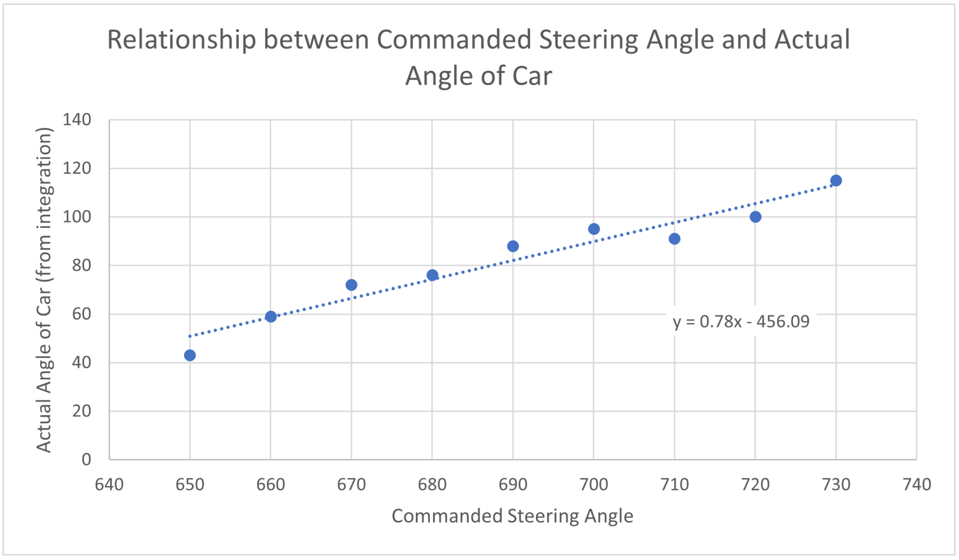


Figure 6: Relationship between Commanded Steering Angle and Actual Angle of Car

Using the data in Figure 6, the relationship between commanded angle and actual angle was found via a best-fit line shown in Equation 3.

$$angle_{actual} = 0.78 * (angle_{cmd}) - 456.09 \tag{3}$$

Thus, changing the commanded steering angle by 10 results in an actual degree change of 7.8 degrees. To achieve a 90 degree turn under these constraints (0.160 speed, 1 second time), we should command the car to a steering angle of 700.

3.3 Encoder

To understand the relationship between encoder readings and coded motor percentage, we ran the car at motor percentages of 0.158, 0.159, 0.160, 0.161, 0.162, 0.163, 0.164. Then, we collected data on the encoder measurements and averaged the measurements for each speed. Then, we plotted the data to find the trendline between average encoder reading and commanded motor percentage. The results are shown in Figure 7.

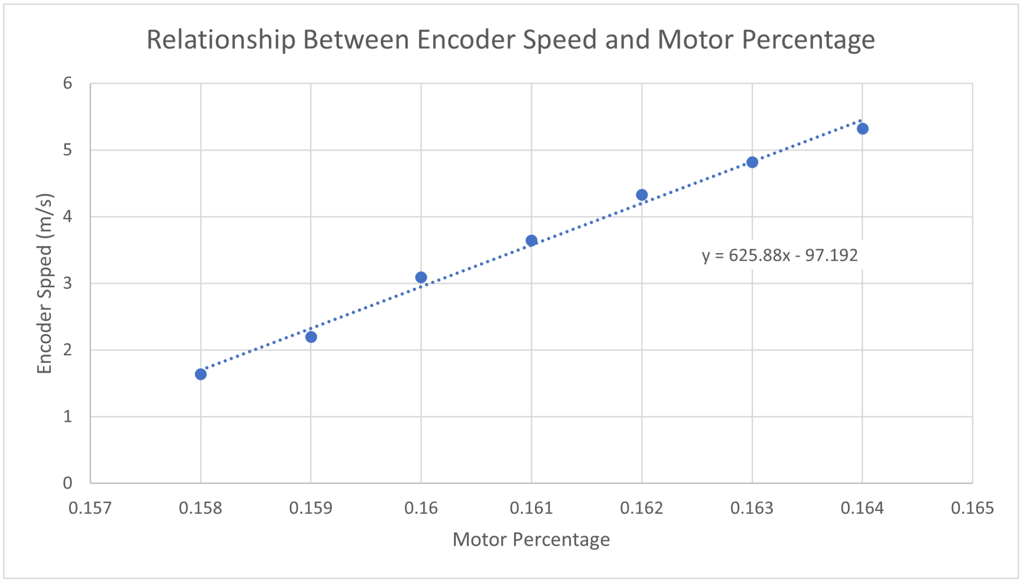


Figure 7: Relationship Between Encoder Speed and Motor Percentage

As shown in Figure 7, the relationship between encoder measurements and commanded motor speed/percentage is written in Equation 4.

$$speed_{encoder} = 625.88 * (speed_{motor}) - 97.192 \tag{4}$$

This data was then used as the conversion between encoder data that the car returns and the PID system that determines commanded car speed/percentage.

4 Results

In the end, our PiCar performed admirably given the challenges and made it through two-thirds of the track. A video can be seen on our website. We were able to complete all of our main goals with turning falling slightly short. Firstly, our PID control system of correcting drift via the IMU sensor performed well with proper gain tuning. It can be seen especially at the beginning that drift is corrected and the car can steered back in the correct direction. Secondly, our other PID system for controlling speed works well for maintaining a constant speed. At the start of the track, we go slightly uphill but the car speeds up slightly after overcoming it. More noticeable is what happens after the first turn, where the car loses a lot of speed and momentum and thus speeds up tremendously after finishing the turn to get back up to speed before throttling down once it reaches its desired speed.

Perhaps one of the most challenging difficulties was turning, a common challenge for every group. Although not our focus, it proved to be essential to a reliable product. Turning proved to be unreliable even with empirical analysis to try to perfect the perfect turn. Too many variables including slight variation in placement, accumulated buildup of error on sensors, and uncontrollable terrain variables such as bumps, wind, and the environment all combined to make it so turning was never the same. PID did its best to control a turn, for example if the turn undershot or overshot PID would account for this. However, the slightest difference in angle (even as little as a degree) where the PiCar believed it was at a true zero degree angle when it was slightly off can build up overtime in terms of drift. Over several turns the error is only furthermore exacerbated. Our end result is a definitive proof of concept. The engineering principles of our design were proven and improvements are discussed next.

5 Discussion

A fundamental drawback of the IMU sensor is because we record angular velocity, we need to integrate in order to get absolute angle. As mentioned, this exacerbates our error significantly. For example, a slight error in angular velocity (0.1 rad/s for instance) is fine for a tenth of a second, but over the course of navigating a straightaway this could prove problematic. Solutions also have their drawbacks. Putting a filter such as a Kalman filter or moving average filter inherently slows our sampling rate of measurements to use, which is crucial for something as fast as a turn which is typically less than a second. Increasing our speed so there is less time for error to accumulate is better for less accumulation of error. However, a higher speed gives us a shorter window of time for the LiDAR to detect obstacles or a turn. This should be common sense - there is a reason why speed limits exist! Ultimately, we think a grand improvement could be using a 9-DOF IMU or magnetometer to read in absolute angular orientation. This would remove integral windup and error and gives us a better means to correct angle even past the turn.

In truth, probably the biggest limitation of all is time. In the last the week, after switching to LiDAR from ultrasonic for range issues, we killed three Raspberry Pis. After confirming it was probably a power supply issue, we split power to the motors and Raspberry Pi using an additional external portable charger. Last minute changes like these alter the weight distribution of the car significantly. These changes invalidation a lot of the tuning we did for turning, speed control, and more. In a perfect world, the turn would be perfected to perform approximately a 90-degree turn with a couple degrees of variation. Our gains for PID for both systems would be meticulously tuned. Yet, probably more importantly, a more robust layout of sensors and additional components for the car is needed to keep weight distribution consistent and reliable. The PiHat being established by a previous group will be crucial in organizing sensors, and 3D mounts will be fundamental for keeping loose components in check.

6 Conclusions

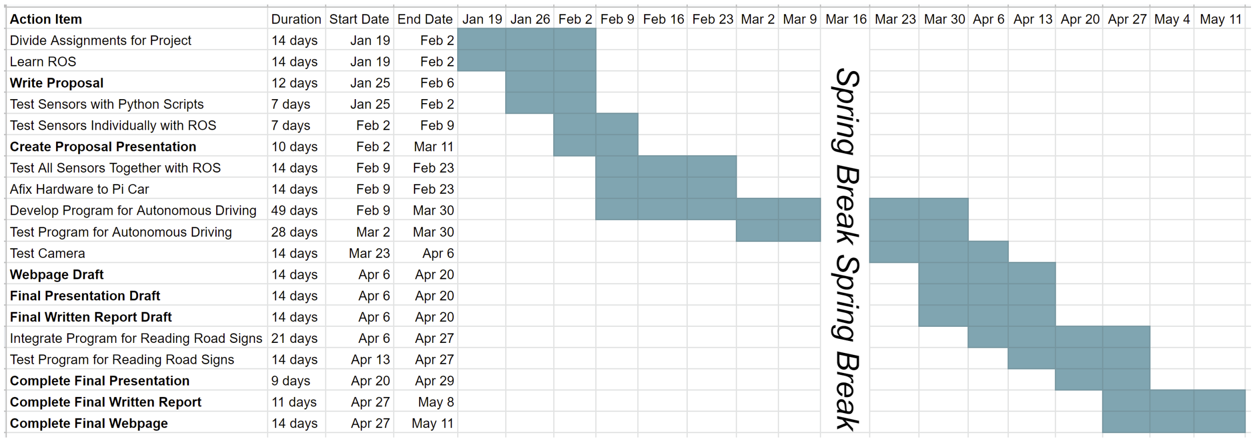
In conclusion, our development of a PID control system to navigate the the track proved relatively successful. With additional tuning and time, our system would have completed the track. While the ultrasonic sensor was too unreliable to utalize, the LiDAR sensor was a very reliable way to trigger the turn.

7 Deliverables

The final deliverables of this capstone project include the design and implementation of our autonomous vehicle. This includes a completed prototype of the physical Pi Car with the necessary software. The Pi Car was completed successfully in terms of implementing PID control on speed and maintaining forward velocity. The final code is provided on our SD card and in our Github. The final project presentation deliverable was completed on April 29, 2022 during the lab demonstration for ESE Day. This is the final project report and our web-page can be found here. The presentation, this report, and the website all detail the design, results, and discussion of our project.

8 Project Timeline

The project was completed during the Spring semester of 2022 (January 2022 to May 2022). Each group member has distinct responsibilities that correspond to the tasks shown in the project Gantt chart in Figure 8. The deliverables are in bold.



Action Item	Duration	Start Date	End Date	Jan 19	Jan 26	Feb 2	Feb 9	Feb 16	Feb 23	Mar 2	Mar 9	Mar 16	Mar 23	Mar 30	Apr 6	Apr 13	Apr 20	Apr 27	May 4	May 11
Divide Assignments for Project	14 days	Jan 19	Feb 2	▨	▨	▨						Spring Break Spring Break								
Learn ROS	14 days	Jan 19	Feb 2	▨	▨	▨														
Write Proposal	12 days	Jan 25	Feb 6		▨	▨														
Test Sensors with Python Scripts	7 days	Jan 25	Feb 2		▨	▨														
Test Sensors Individually with ROS	7 days	Feb 2	Feb 9			▨	▨													
Create Proposal Presentation	10 days	Feb 2	Mar 11			▨	▨													
Test All Sensors Together with ROS	14 days	Feb 9	Feb 23				▨	▨	▨											
Afix Hardware to Pi Car	14 days	Feb 9	Feb 23				▨	▨	▨											
Develop Program for Autonomous Driving	49 days	Feb 9	Mar 30				▨	▨	▨	▨	▨		▨	▨						
Test Program for Autonomous Driving	28 days	Mar 2	Mar 30							▨	▨		▨	▨						
Test Camera	14 days	Mar 23	Apr 6										▨	▨	▨					
Webpage Draft	14 days	Apr 6	Apr 20											▨	▨	▨				
Final Presentation Draft	14 days	Apr 6	Apr 20											▨	▨	▨				
Final Written Report Draft	14 days	Apr 6	Apr 20											▨	▨	▨				
Integrate Program for Reading Road Signs	21 days	Apr 6	Apr 27												▨	▨	▨	▨		
Test Program for Reading Road Signs	14 days	Apr 13	Apr 27													▨	▨	▨		
Complete Final Presentation	9 days	Apr 20	Apr 29														▨	▨		
Complete Final Written Report	11 days	Apr 27	May 8															▨	▨	▨
Complete Final Webpage	14 days	Apr 27	May 11															▨	▨	▨

Figure 8: Project Gantt Chart

Our group adhered well to the schedule through Spring Break. However, after Spring Break, we started to encounter unanticipated issues due to not having a Pi Hat and issues with our sensors. This pushed our finalization of the PID control program back to mid-April. We also decided to no longer use road signs to indicate turning, but simply use distance detected by the LiDAR to commence a turn. This gave us additional time to work out issues with sensor readings, implementing the PID, and doing testing on the actual track. While the design may have gotten behind our anticipated schedule, the report, presentation, and webpage stayed on track with our schedule.

8.1 Responsibilities of Each Member

Tom Goon was responsible for the main software development of the project. His responsibilities included python scripts to test the sensors, integrating the sensors with ROS, and developing the main control program via implementation in Python. He also ensured integration of the Reading Road Signs program with the main control algorithm. As a computer science double major, Tom has years of experience with developing programs and algorithms, which prepared him for the software development role.

Allison Todd was responsible for integrating the hardware and software. Her responsibilities included analyzing results from sensor tests, creating data visualization for all results, and ensuring the validity of the main control program. She also developed the control system design. As a computer science minor with experience in hardware through classes and an internship, Allison was prepared to take on this role.

Caitlind Walker was responsible for the hardware of the project. Her responsibilities included all electrical and mechanical connections between the Raspberry Pi and the sensors, creation of the block diagram, and aiding in the development of the control system design with Allison. Caitlind's experience with classes in electrical engineering and mechanical engineering, along with an internship in each field, made her prepared to take on this role.

9 References

[1] Astels, Dave. “Using the SLAMTEC RPLIDAR on a Raspberry Pi.” *Adafruit Learning System*, Adafruit, 19 Mar. 2019, https://learn.adafruit.com/slamtec-rplidar-on-pi/overview.

[2] ESE205 Materials

[3] ESE441 Materials (PID Control)