AUTONOMOUS PLATOONING GOLF CART FOR SHORT DISTANCE TRAVEL

A Research Paper submitted to the Department of Mechanical Engineering
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Mechanical Engineering

By

Charles Rushton

May 9th, 2022

ADVISOR
Tomonari Furukawa, Department of Mechanical Engineering

1 Introduction

Interest in automated vehicles has continued to grow as their ability to potentially reshape society has become more realized. Autonomous vehicles most notably have the potential to diminish the amount of road incidents as they can eliminate human decisions and error, which is the cause of at least 90% of vehicle incidents. (Smith, 2013). They also provide a significant increase in independence and accessibility to disabled people and the elderly. Implementations of automated systems in the automotive industry are expected to drive not only on their own, but in conjunction with other vehicles around them. At the Virginia Cooperative Autonomous Robotics (VICTOR) lab, the development of an autonomous golf cart platooning system will add to this ongoing research and provide the project sponsor, Club Car, with a model for developing their own autonomous vehicles.

Many large companies are developing autonomous vehicles and associated technologies for commercial use. Waymo, a subsidiary of Google's Alphabet Company, has been developing an autonomous ridesharing taxi service since 2018 and first starting in Phoenix (Waymo, 2022). Waymo has even expanded to San Francisco, and recently obtained a commercial license to monetize their ride sharing service (Bellan, 2022). Each vehicle uses both a 360° LiDAR and a long-range radar to obtain accurate information about the vehicle's surroundings from far away to allow for longer lead times. Although these sensors are good at judging distances at very large ranges and multiple conditions, they are still affected by noise and can’t easily differentiate between essential objects (Lampinen, 2020). The radar and lidar for example cannot easily identify the difference between a person, a road sign, and a trash bag. To help reduce these blind spots, Waymo uses a peripheral system with extra lidars and radars in addition to cameras to reliably identify objects (Jeyachandran, 2020). This increase in object detection certainty has allowed Waymo to breakthrough and operate in more urban environments.

Beyond the sensors used for mapping environments, the development of drive by wire systems and motion planning is essential to autonomous movement. Autonomous platooning systems are an example of this because the goal is one manually-driven truck leading one or more autonomously driven trucks. Platooning systems are more energy efficient because the autonomously-controlled vehicle is able to safely maintain a closer following distance than a manually-driven vehicle, which reduces the aerodynamic drag the following vehicles experience. This closer following distance would also allow for more cars to fit in the same length of lane, potentially reducing traffic jams, and with that the time and fuel spent idling in stop and go traffic (Fernandes & Nunes, 2010). The technologies involved in different truck platooning systems explained by Tsugawa, Jeschke et al. (2016) include a wide range of sensors from vision units (cameras and lasers) to LiDAR, vehicle-to-vehicle communications (V2VC), and GPS systems for localization. The quantities measured when the Energy ITS platooning system was tested were lateral and longitudinal control (staying within the bounds of the lane markings and within following distance of the preceding truck, respectively), speed, lane changing while following a target path, and platoon formation time (Tsugawa, Jeschke et al., 2016). The vision units placed on each of the trucks, which consisted of a CCD camera and laser scanner, were able to effectively detect the lane markings on the road under a variety of lighting and weather conditions and identify the trucks' distance from and orientation with respect to the lane markings (Tsugawa, 2014). In the event that the lane markings could not be detected, the yaw rate information from a leading truck could be sent to the following truck (Fukao et al., 2013). The combination of V2VC and radar and LiDAR sensors placed on each truck allowed for the trucks to mostly stay within the desired following distance behind its preceding truck within 10 cm (Tsugawa, 2014). However, platoon formation time and lane changing that followed the target path were not successful, and the reliability of the various devices and systems was not tested.

The main goal of this project is to develop an autonomous platooning system using a leader and a follower golf cart that can be used in many areas of a university campus such as Engineers Way, a section of the University of Virginia's campus. The follower cart will accurately follow the leader cart which can be either autonomously driven or operated by a human. In order to accomplish this, the leader cart must be able to create a map of its environment and localize in it as well as detect and avoid obstacles using a suite of sensors. It must also send accurate motion information to the follower cart which will be used in conjunction with a motion model and other sensors to follow the leader. Lastly, it must be able to interact with users to provide a safe experience.

This report will expand on essential knowledge about this project including the work done by previous teams including motion subsystems, environmental detection, and leader detection. Next, concept generation and subsequent design decisions will be explained and justified as to why they were chosen. The overall final design will be described and the progress made to reach the design in each of the specific subsystems will be highlighted. Results from the validation of the design will be explained and shown. Finally, conclusions will be made in order to guide future team's work to success with this project.

2 Essential Knowledge

Last year, UVA Mechanical Engineering undergraduate students Art Ken Fontelera, Jee Soo Shin, William Smith, Peter Wellman, and Jack Yocom worked towards the goal of developing a fully autonomous golf cart platooning system. During their progress, they were able to achieve autonomous braking, steering, and acceleration in one golf cart and develop a point cloud map using sensors in that same golf cart. In 2018-2019, Virginia Tech Mechanical Engineering undergraduate students Jeronimo Cox, Alina Voelker, Andrew Chen, Daniel Cox, Shen Chu, Alejandra Arriaga, David Lee, Monica Karas, Alias Ishanzai, and Webster Bray worked towards the same goal as well, but they focused more heavily on the platooning aspect of the goal. During their progress, a method to transfer data between carts for autonomous platooning was developed, but their motion subsystems did not work well enough to safely execute their model. The knowledge gained by those students during the development of their systems is valuable to have their design improved upon by future teams.

2.1 Braking

The braking system developed by last year's team was installed and functional on cart 788, which is the cart to be used as the Leader Cart. This braking system consisted of a rotary DC motor that, when engaged, pulled on an airline cable that directly pulled back the brake pedal. The DC motor is attached underneath the golf cart, as shown in Figure 1 below. The cable then passed through a series of pulleys before it eventually attached to the brake pedal directly, as shown in Figure 2 below.

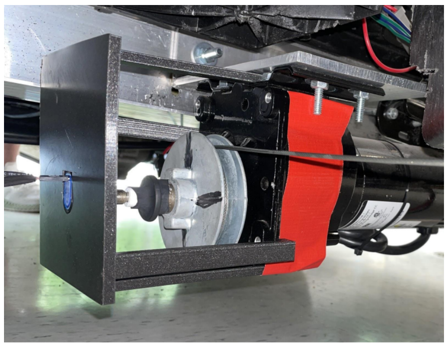


Figure 1: DC motor used on Cart 788 to pull the braking cable.

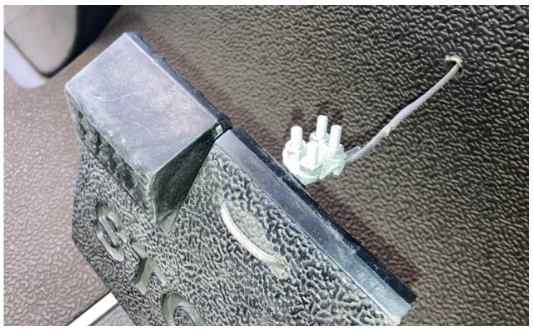


Figure 2: Attachment of the airline cable to the brake pedal of Cart 788 that allows for the pedal to be pulled when the brake motor is engaged.

2.2 Acceleration

Both the Virginia Tech and UVA teams have controlled acceleration through an electrical solution rather than a mechanical actuator. Their implementation consisted of an Arduino that is connected to a digital potentiometer, which varies the voltage supplied to the golf cart motor controller. Cart 788 has this subsystem in place, but Cart 789 does not. An adequate method of controlling acceleration on the following cart will be needed for successful automated following.

2.3 Steering

There have been two different steering systems developed by past teams. The first solution developed by the Virginia Tech team consisted of a Nexteer Electric Power Steering (EPS) system mounted onto cart 789. The EPS system as seen in Figure 3, allowed for CAN bus communication and the ability to program steering angles directly to the steering column manipulated by a geared DC motor. Last year's UVA team used a ClearPath servo motor mounted in parallel to the steering column on cart 788. The reason why the UVA team switched to the Clearpath servo motor was due to communication issues with the Nexteer EPS system. The motor as seen in Figure 3, was connected and controlled through an Arduino and moved using absolute positioning based on the built in encoder.

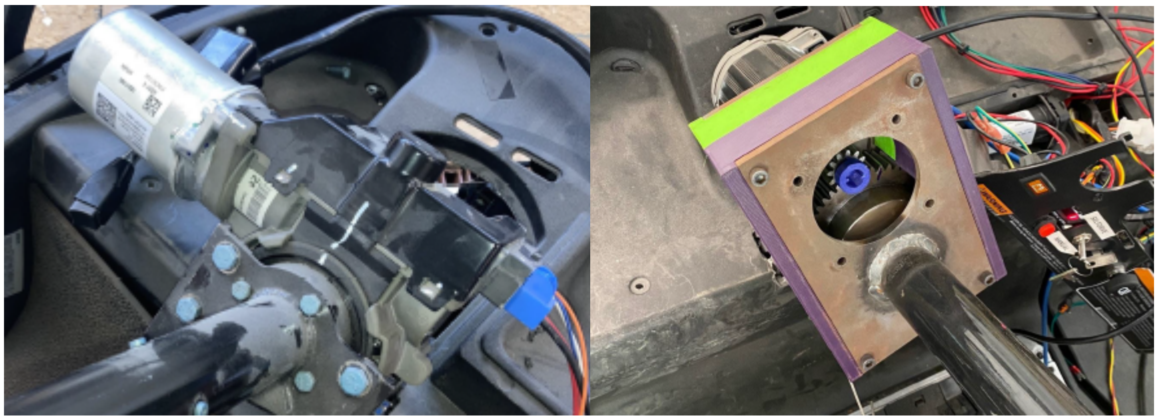


Figure 3 : Nexteer electric power steering system (left); Teknic ClearPath Servo Motor with gears housed in an enclosure (right).

3 Design Process

3.1 Customer needs

3.1.1 Gather Raw Data

To gather raw data, interviews were conducted in the markets for this autonomous golf project. The autonomous golf cart's goal is to track and follow a lead vehicle, with time permitting will also be autonomous, to pick up students. Therefore, the primary market of this project was Professor Tomonari Furukawa, a faculty member at the University of Virginia (UVA), and William Smith, a graduate student in Professor Furukawa's research laboratory. UVA students, faculty, and staff, as well visitors of the campus were deemed to be the secondary market.

Initially, questions were posed to both markets to determine information about how stakeholders move around UVA's campus and their interest in autonomous vehicles. To determine key information on transportation, the following questions were asked: "How do you get to class", "What's important in the way you get to class", "What's important in the way you get to class", "What do you like about your current method", and "What do you not like about your current method”. To determine various user's perspectives on autonomous vehicles, the following questions were asked: “How comfortable are you with your car indicating cars next to/behind you?”, “How comfortable are you with your car indicating cars next to/behind you?”, “How comfortable are you with your car changing lanes?”, “How comfortable are you with your car turning?”, “How comfortable are you with your car accelerating/decelerating/cruise control?”, and “What would make you more trustworthy of computer autonomy?”.

To determine more technical customer needs from the primary market, the team asked William Smith and Siddharth Singh, a Ph.D candidate in Professor Furukawa’s laboratory, the following questions: “Is there an initial budget?”, “What are the main capabilities expected?”, “Is there any technology that must be implemented?”, “Are there any safety features that you would like?”, “Are there any goals regarding aesthetics initially or is functionality the main concern?”, “Is there a specific timeline?”, “Should the driver be able to have emergency take over”, “How long should the golf cart be able to run compared to a non autonomous vehicle”, “How fast should the carts be able to go?”, “What distance would you like the carts to maintain?”, “Is there a minimum number of passengers required per cart?”, “What are the specific milestones?”, “What is the necessary information to be shown to the passengers on screen?”, and “Is there a preferred method of pick-up?”.

From the raw data, responses followed multiple trends. The team found that the market currently likes autonomous vehicles technology in the following circumstances: preventing accidents, aid in parking, aid in lane adjustment, and cruise control. However, the market generally dislikes autonomous vehicles because of a general lack of trust and the lack of versatility in different weather conditions. Additionally, the market would typically use the autonomous vehicle for shuttling around campus in a pick-up drop-off model, similar to bus stops. To meet the large magnitude of students shuttling, multiple vehicles platooning with sophisticated object detection seem necessary for the stakeholders. Lastly, the market suggested that autonomous vehicles should have improved awareness of surroundings and more reliability when following other vehicles.

3.1.2 Construction of Customer Needs as Primary and Secondary Needs

Using the raw data, the team interpreted the need from the information. Next, the interpreted need was given priority. High priority needs were considered primary needs, while less necessary needs were considered secondary needs. Figure 4 summarizes the customer statements to interpret need with priority.

Question/Prompt	Customer Statement	Interpreted Need	P or S
Typical Uses	Shuttling Students	● Detection for students ● Dynamically determining environment ● Recognition of stops	S
	Series of Vehicles	● Detection of lead vehicle ● Communication between different machines	P
	Determination of Obstacles	● Ability to detect objects in the surrounding area	P
Likes-Current technology	Accident Prevention	● Recognition of situations that likely lead to imminent collisions ● Analyze the surrounding environment in relation to car speed	P
	Parking/Lane Assistance	● Constant use of sensors to detect surroundings and white lines for lanes/parking spots	S
	Cruise Control	● Maintain a set distance between carts by controlling speed	P
Dislikes-current technology	Lack of Weather Reliability	● Multiple visual sensors and radar ● Ability to perform under varying circumstances	S

	Lack of Trust	● Rigorous testing ● Need for better communication between innovators and clients	P
Suggested Improvements	Awareness of surroundings	● Fast and accurate processing of sensor data	P
	Reliably follows other vehicles	● Vehicle to vehicle communication	P

Figure 4: Customer Needs with Priority. The P represents "primary" needs, while S represents "secondary needs".

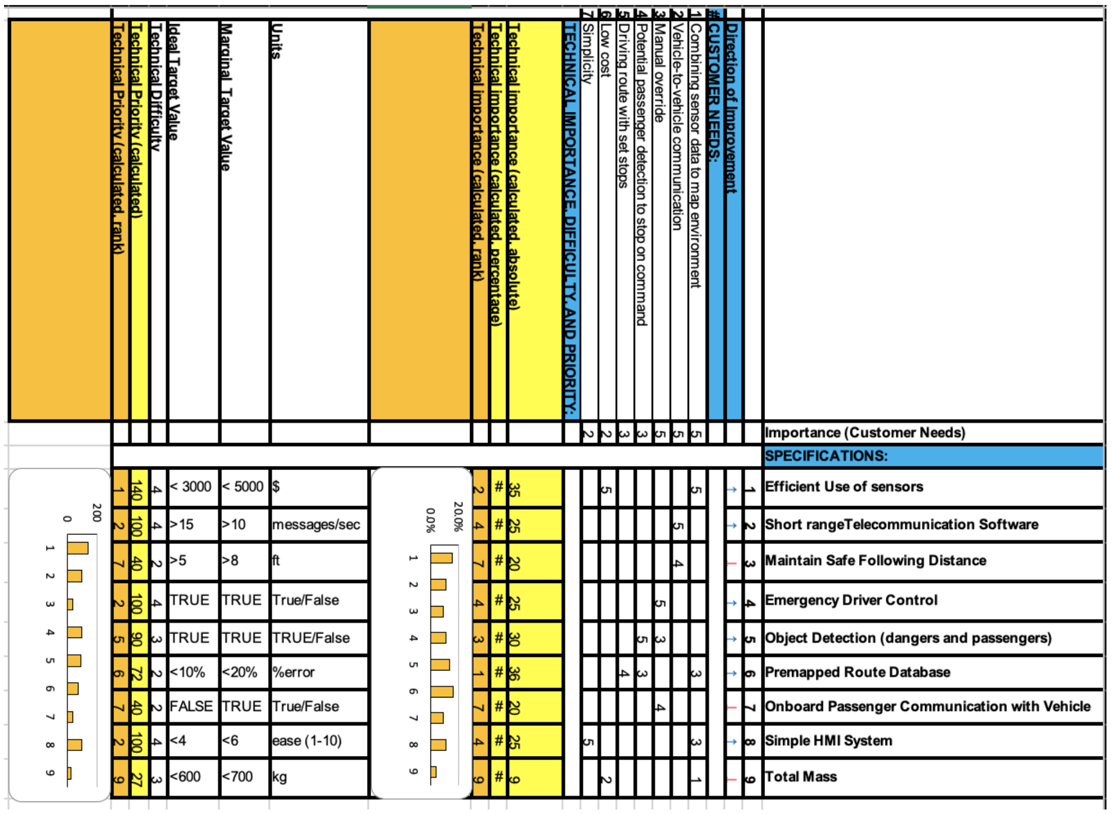
3.1.3 Rank Order the Customer Needs

Lastly, customer needs were ordered in relative importance. Rank ordering the customer needs allows for the team to focus on the most important tasks. Figure 5 summarizes the ordering of the five tasks that may have significant technical trade-offs in the final design.

Autonomous Golf Cart Importance Safety			
Scale: 1. Feature is undesirable. I would not consider a product with this feature. 2. Feature is not important, but I would not mind having it. 3. Feature would be nice to have, but is not necessary. 4. Feature is highly desirable, but I would consider a product without it. 5. Feature is critical, I would not consider a product without this feature.			
#	1-5	Autonomous Golf Cart Need	Unique, Exciting, and or Unexpected
1	5	Accurate detection and navigation of environmental surroundings	
2	5	Follower cart reliably follows leader cart	
3	5	Manual override and emergency stops	
4	3	Potential passenger detection to stop on command	✓
5	3	Driving route with set stops	

Figure 5: Summary Table of the Customer Needs

3.2 Target specifications



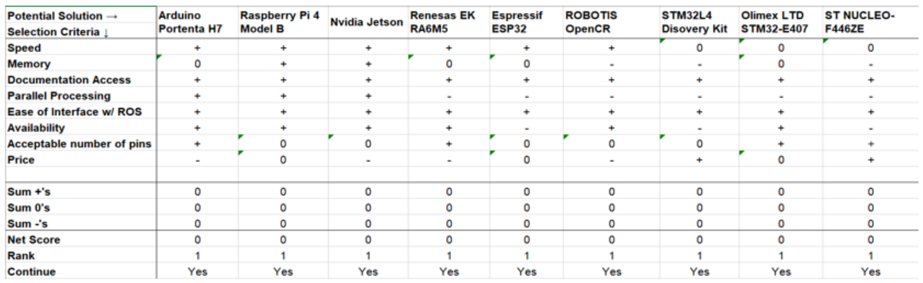
#	SPECIFICATIONS:	Direction of Improvement	1 Combining sensor data to map environment	2 Vehicle-to-vehicle communication	3 Manual override	4 Potential passenger detection to stop on command	5 Driving route with set stops	6 Low cost	7 Simplicity	Technical importance (calculated, absolute)	Technical importance (calculated, percentage)	Technical importance (calculated, rank)	Units	Marginal Target Value	Ideal Target Value	Technical Difficulty	Technical Priority (calculated)	Technical Priority (calculated, rank)
	Importance (Customer Needs)		5	5	5	3	3	2	2									
1	Efficient Use of sensors	→	5					5		35	#	2	$	< 5000	< 3000	4	140	1
2	Short rangeTelecommunication Software	→		5						25	#	4	messages/sec	>10	>15	4	100	2
3	Maintain Safe Following Distance	—		4						20	#	7	ft	>8	>5	2	40	7
4	Emergency Driver Control	→			5					25	#	4	True/False	TRUE	TRUE	4	100	2
5	Object Detection (dangers and passengers)	→			3	5				30	#	3	TRUE/False	TRUE	TRUE	3	90	5
6	Premapped Route Database	→	3			3	4			36	#	1	%error	<20%	<10%	2	72	6
7	Onboard Passenger Communication with Vehicle	—			4					20	#	7	True/False	TRUE	FALSE	2	40	7
8	Simple HMI System	→	3						5	25	#	4	ease (1-10)	<6	<4	4	100	2
9	Total Mass	—	1					2		9	#	9	kg	<700	<600	3	27	9

Figure 6: Quality Function Diagram from the Customer Needs

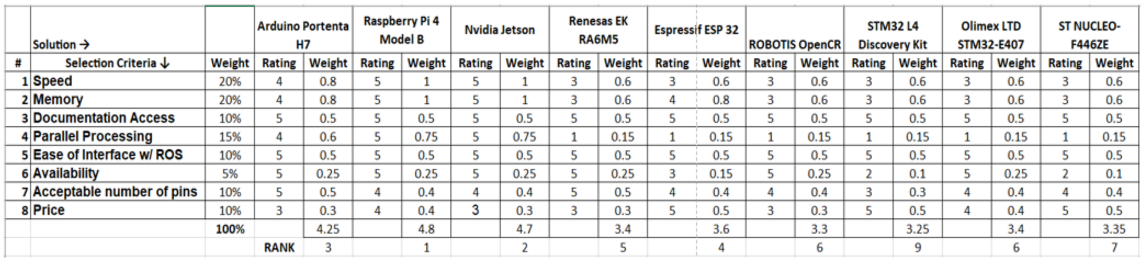
The correlation between customer needs and technical specifications is determined through the Quality Function Deployment (QFD) diagram shown in Figure 6. A score between 1 and 5 is given, with 1 indicating minimum correlation and 5 indicating maximum correlation between factors. Based on the correlation, the technical importance is then calculated and ranked. Lastly, the technical difficulty for each of the factors is taken into consideration to determine a score and rank for technical priority. The efficient use of sensor specification ranked the highest, with short range telecommunication software, emergency driver control, and simple HMI system specifications ranking second. Ranking and graphing technical importance and technical priority allows the team to develop an understanding of how much time and resources need to be allocated to meet specifications.

3.3 Concept Generation and Concept Selection

The braking system, microcontroller, and leader cart detection were the three areas the team had to determine concepts for. A braking actuation system was needed for Cart 789 and a more powerful microcontroller was needed to control the motion subsystems simultaneously on both golf carts. To implement a leader following system, a leader detection method needed to be determined. After generating concepts for each of the areas, a concept screening and scoring was completed for each system (Figure 7, 8, and 9). Concept screening involved rating a potential solution "+" for excellent performance, "-" for poor performance, and "0" for acceptable level of performance for each of the selection criteria. Concept scoring phase consisted of giving each selection criteria a weight based on its importance. Then, a score between 1 and 5 was given based on their expected performance, with 5 representing the highest performance and 0 representing poor performance. Based on the scores and weight, a rank was computed for all potential solutions.

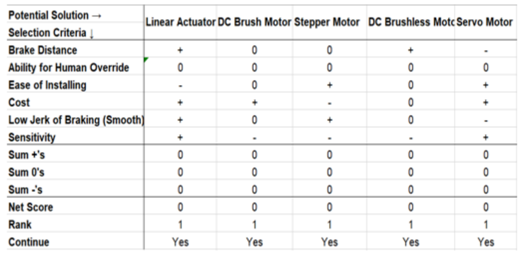


Potential Solution → Selection Criteria ↓	Arduino Portenta H7	Raspberry Pi 4 Model B	Nvidia Jetson	Renesas EK RA6M5	Espressif ESP32	ROBOTIS OpenCR	STM32L4 Disovery Kit	Olimex LTD STM32-E407	ST NUCLEO-F446ZE
Speed	+	+	+	+	+	+	0	0	0
Memory	0	+	+	0	0	-	-	0	-
Documentation Access	+	+	+	+	+	+	+	+	+
Parallel Processing	+	+	+	-	-	-	-	-	-
Ease of Interface w/ ROS	+	+	+	+	+	+	+	+	+
Availability	+	+	+	+	-	+	-	+	-
Acceptable number of pins	+	0	0	+	0	0	0	+	+
Price	-	0	-	-	0	-	+	0	+
Sum +'s	0	0	0	0	0	0	0	0	0
Sum 0's	0	0	0	0	0	0	0	0	0
Sum -'s	0	0	0	0	0	0	0	0	0
Net Score	0	0	0	0	0	0	0	0	0
Rank	1	1	1	1	1	1	1	1	1
Continue	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

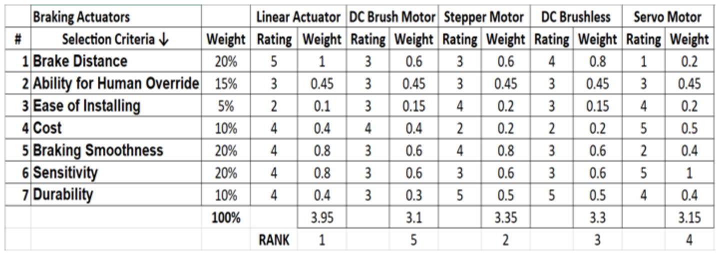


	Solution →		Arduino Portenta H7		Raspberry Pi 4 Model B		Nvidia Jetson		Renesas EK RA6M5		Espressif ESP 32		ROBOTIS OpenCR		STM32 L4 Discovery Kit		Olimex LTD STM32-E407		ST NUCLEO-F446ZE	
#	Selection Criteria ↓	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight
1	Speed	20%	4	0.8	5	1	5	1	3	0.6	3	0.6	3	0.6	3	0.6	3	0.6	3	0.6
2	Memory	20%	4	0.8	5	1	5	1	3	0.6	4	0.8	3	0.6	3	0.6	3	0.6	3	0.6
3	Documentation Access	10%	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5
4	Parallel Processing	15%	4	0.6	5	0.75	5	0.75	1	0.15	1	0.15	1	0.15	1	0.15	1	0.15	1	0.15
5	Ease of Interface w/ ROS	10%	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5	5	0.5
6	Availability	5%	5	0.25	5	0.25	5	0.25	5	0.25	3	0.15	5	0.25	2	0.1	5	0.25	2	0.1
7	Acceptable number of pins	10%	5	0.5	4	0.4	4	0.4	5	0.5	4	0.4	4	0.4	3	0.3	4	0.4	4	0.4
8	Price	10%	3	0.3	4	0.4	3	0.3	3	0.3	5	0.5	3	0.3	5	0.5	4	0.4	5	0.5
		100%		4.25		4.8		4.7		3.4		3.6		3.3		3.25		3.4		3.35
			RANK	3		1		2		5		4		6		9		6		7

Figure 7: Microcontroller Concept Scoring and Screening

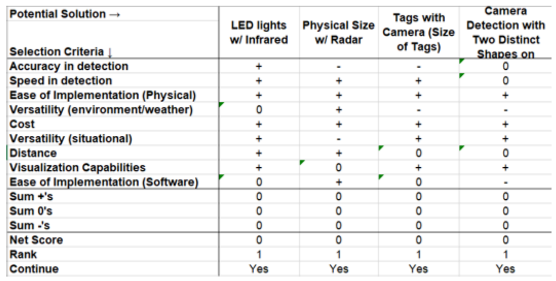


Potential Solution → / Selection Criteria ↓	Linear Actuator	DC Brush Motor	Stepper Motor	DC Brushless Motc	Servo Motor
Brake Distance	+	0	0	+	-
Ability for Human Override	0	0	0	0	0
Ease of Installing	-	0	+	0	+
Cost	+	+	-	0	+
Low Jerk of Braking (Smooth)	+	0	+	0	-
Sensitivity	+	-	-	-	+
Sum +'s	0	0	0	0	0
Sum 0's	0	0	0	0	0
Sum -'s	0	0	0	0	0
Net Score	0	0	0	0	0
Rank	1	1	1	1	1
Continue	Yes	Yes	Yes	Yes	Yes

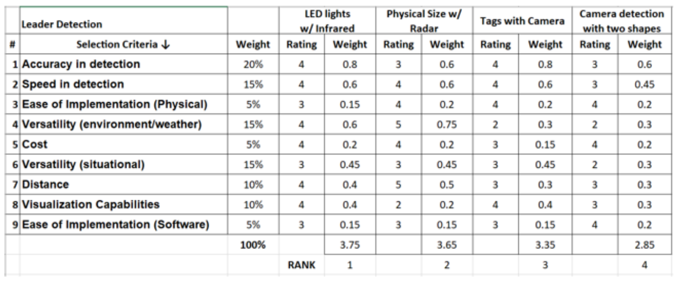


	Braking Actuators		Linear Actuator		DC Brush Motor		Stepper Motor		DC Brushless		Servo Motor	
#	Selection Criteria ↓	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight
1	Brake Distance	20%	5	1	3	0.6	3	0.6	4	0.8	1	0.2
2	Ability for Human Override	15%	3	0.45	3	0.45	3	0.45	3	0.45	3	0.45
3	Ease of Installing	5%	2	0.1	3	0.15	4	0.2	3	0.15	4	0.2
4	Cost	10%	4	0.4	4	0.4	2	0.2	2	0.2	5	0.5
5	Braking Smoothness	20%	4	0.8	3	0.6	4	0.8	3	0.6	2	0.4
6	Sensitivity	20%	4	0.8	3	0.6	3	0.6	3	0.6	5	1
7	Durability	10%	4	0.4	3	0.3	5	0.5	5	0.5	4	0.4
		100%		3.95		3.1		3.35		3.3		3.15
			RANK	1		5		2		3		4

Figure 8: Braking Actuation Concept Scoring and Screening



Potential Solution → / Selection Criteria ↓	LED lights w/ Infrared	Physical Size w/ Radar	Tags with Camera (Size of Tags)	Camera Detection with Two Distinct Shapes on
Accuracy in detection	+	-	-	0
Speed in detection	+	+	+	0
Ease of Implementation (Physical)	+	+	+	+
Versatility (environment/weather)	0	+	-	-
Cost	+	+	+	+
Versatility (situational)	+	-	+	+
Distance	+	+	0	0
Visualization Capabilities	+	0	+	+
Ease of Implementation (Software)	0	+	0	-
Sum +'s	0	0	0	0
Sum 0's	0	0	0	0
Sum -'s	0	0	0	0
Net Score	0	0	0	0
Rank	1	1	1	1
Continue	Yes	Yes	Yes	Yes



	Leader Detection		LED lights w/ Infrared		Physical Size w/ Radar		Tags with Camera		Camera detection with two shapes	
#	Selection Criteria ↓	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight
1	Accuracy in detection	20%	4	0.8	3	0.6	4	0.8	3	0.6
2	Speed in detection	15%	4	0.6	4	0.6	4	0.6	3	0.45
3	Ease of Implementation (Physical)	5%	3	0.15	4	0.2	4	0.2	4	0.2
4	Versatility (environment/weather)	15%	4	0.6	5	0.75	2	0.3	2	0.3
5	Cost	5%	4	0.2	4	0.2	3	0.15	4	0.2
6	Versatility (situational)	15%	3	0.45	3	0.45	3	0.45	2	0.3
7	Distance	10%	4	0.4	5	0.5	3	0.3	3	0.3
8	Visualization Capabilities	10%	4	0.4	2	0.2	4	0.4	3	0.3
9	Ease of Implementation (Software)	5%	3	0.15	3	0.15	3	0.15	4	0.2
		100%		3.75		3.65		3.35		2.85
			RANK	1		2		3		4

Figure 9: Leader Detection Concept Scoring and Screening

The Raspberry Pi 4, linear actuator, and infrared camera with IR LEDs were the highest-ranking systems in the three categories. The selected solution for leader detection would have IR LEDs placed on the leader cart and an IR sensor on the following cart. While the linear actuator was the highest ranked for braking, the team decided to use the clear path servo motor that was used for steering in the leader cart. The decision was made based on the information that a team previously had encountered difficulty using linear actuator for smooth braking.

4 Final Design

The final design of our system consists of a leader cart that is able to map and localize itself within the surrounding environment and a follower cart that is able to track the leader cart and autonomously follow its path. In order to achieve this autonomous platooning, the leader cart must send position, orientation, steering, and acceleration data to the follower cart. The follower cart uses sensors to detect the distance between the two carts. Recursive Bayesian Estimation (RBE) is used to combine this sensor data with the motion model of the follower golf cart in order to determine the position of the follower cart in the global frame. Using this information, the follower cart is able to execute autonomous platooning with controls for acceleration and steering in order to follow the path of the leader cart. A diagram of the information transfer between the two carts is shown in Figure 10 below.

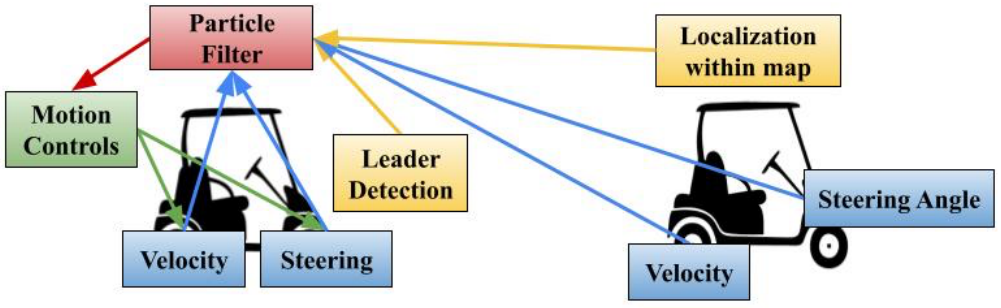


Figure 10: Depiction of the information transferred between the two carts.

As shown in Figure 10, the leader cart receives position and orientation data from the map localization and receives steering angle and velocity from the encoders on those respective sub-systems. The leader detection system uses sensors to determine the distance between the two golf carts. All this information is used in the particle filter to localize the follower cart's position. Using the positions of both carts in the global frame, the follower cart is able to execute motion controls to autonomously follow the leader cart. The controls used, along with the descriptions of each subsystem will be discussed further below.

4.1 Platooning

In order to achieve successful platooning, a particle filter is used to accompany the leader detection sensor readings in order to accurately determine the follower cart's position in the global frame. Motion controls are then able to control the follower cart to autonomously follow the leader cart's path.

4.1.1 Particle Filter

Particle filter is an example of RBE technique that we have implemented to estimate the position of an object, in this case the follower cart. RBE uses motion data to make a prediction of where an object will be, then uses sensor data to update its prediction in order to develop an accurate estimation of the object's position. The particle filter was chosen due to its ability to handle non-linear and non-gaussian systems.

The motion of the golf carts follows an Ackerman steering motion model. Figure 11 below depicts the equations of motion for the states of the vehicle using this motion model.

$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \frac{v}{L}\tan\gamma$$

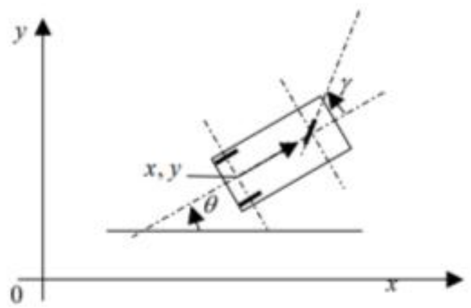


Figure 11: Equations of motion for Ackerman steering vehicles. x, y, and theta represent the vehicle's position and orientation in the global frame and v and gamma represent the control inputs of speed and steering angle, respectively. L is the distance between the front and back wheels.

The particle filter iteratively goes through a four-step process in order to accurately estimate the position of the follower cart. First, an initial sample of particles is spread evenly through the global frame. Each of these particles acts as an individual cart with its own position and orientation. Then, the motion model is used to predict the next state of the golf cart according to the speed and steering control inputs. In this step, each of the particles is moved to a new position according to the motion model. Following the prediction, the sensor reading of the distance between the two carts is used to determine the leader cart's estimated position. This is done using Eqns (1-6) below. The following equations estimate the leader cart position (X_L,Y_L)

according to follower cart state (X_F, Y_F, θ) and sensor reading of the position of the leader cart with respect to the follower cart (dx_f, dy_f).

$$X_L = X_F + x \tag{1}$$
$$Y_L = Y_F + y \tag{2}$$
$$x = dcos(d\theta) \tag{3}$$
$$y = dsin(d\theta) \tag{4}$$
$$d\theta = \theta - \tan^{-1}(\frac{dy_f}{dx_f}) \tag{5}$$
$$d = \sqrt{dx_f^2 + dy_f^2} \tag{6}$$

After the estimation of the leader cart position is made for each particle, it is compared to the actual position of the leader cart in order to assign that particle a weight (a closer estimation results in a higher weight). This weight is then used for the next resampling of particles. Particles with higher weight are resampled more than particles with lower weight. By iteratively going through this four-step process of predicting, updating, assigning a weight, and resampling the particles, an accurate estimation of the follower cart's position is determined by averaging the positions of all the particles.

4.1.2 Controls

Once both positions are known for the golf carts, the follower cart then must control the three motion subsystems in order to follow the leader cart. The following distance between the two carts will be set to a finite distance, so in order to control the acceleration and steering, a PID controller will be used to maintain minimal deviation from the desired following distance and a feed-forward PID controller using the steering angle of the leader cart will be used to maintain minimal deviation from the path set by the leader cart.

To compare with the feed-forward PID controller, a Stanley Controller will be used. A Stanley Controller steering control is shown in Figure 12 and Equation 7.

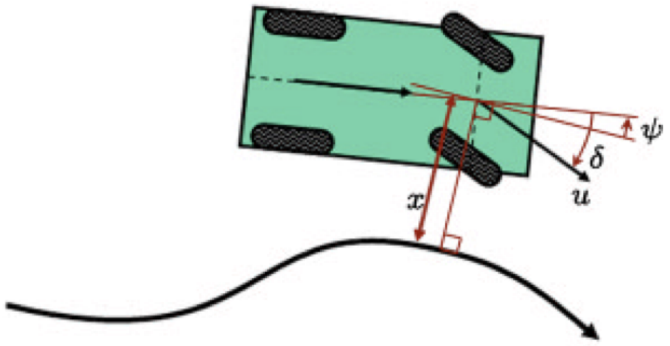


Figure 12: A visual representing the key variables in a stanley controller. (Thrun et al., 2007). Sigma is the turning angle, x is the cross track error, phi is orientation of the nearest path segment, u is the the speed of the vehicle.

$$\delta(t) = \psi(t) + arctan(\frac{k*x(t)}{u(t)}) \tag{7}$$

For the Stanley Controller, the Leader Cart will transmit the location it traveled. Next, a cubic spline will be fitted to the series of points to interpolate between points in case of a loss of communication between the vehicles. Afterwards, a point is selected and Equation 7 is used to determine the ideal turning angle of the vehicle.

4.2 Steering

For the steering system, two iterations have been previously used: a Teknic ClearPath servo motor mounted in parallel with the steering column and a Nexteer Electric Power Steering system. As mentioned earlier the ClearPath servo motor was used in cart 788 due to communication errors involved with the Nexteer EPS. The issues with communicating between a CPU and the Nexteer EPS system have been solved, and the EPS will be installed onto the steering columns of both carts 788 and 789 as seen in Figure 13.

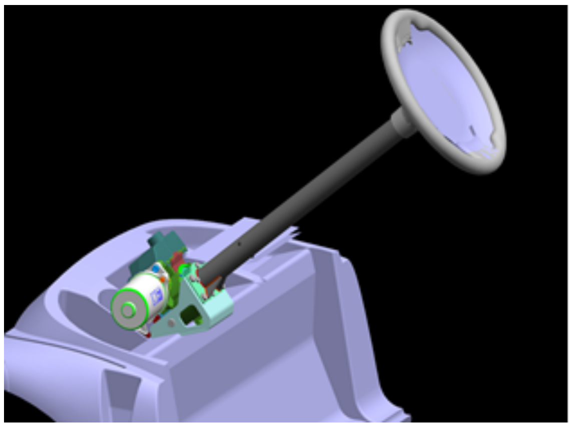


Figure 13: Nexteer Electric Power Steering System

The Nexteer power steering system has five wires coming out of it, one being power, one being ground, one being ignition, and two being CAN HIGH and CAN LOW. The EPS also has its own encoder allowing it to measure the current angle of the shaft, and convert it into an electrical signal read by a computer. The system is driven by a motor and runs on Controller Area Network (CAN) communication protocols to allow a connection between the steering motor and the CPU. To establish communication protocols the two CAN communication wires were connected to pins on the CAN USB Adapter as seen in Figure 14. The system was wired in accordance with the PCAN-USB user manual and is soldered onto a PCB board mounted next to the EPS system.

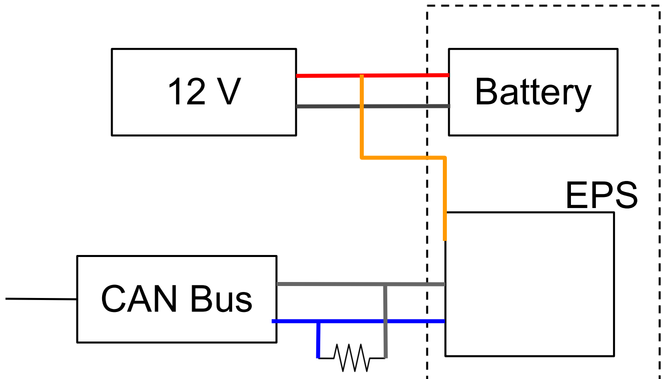
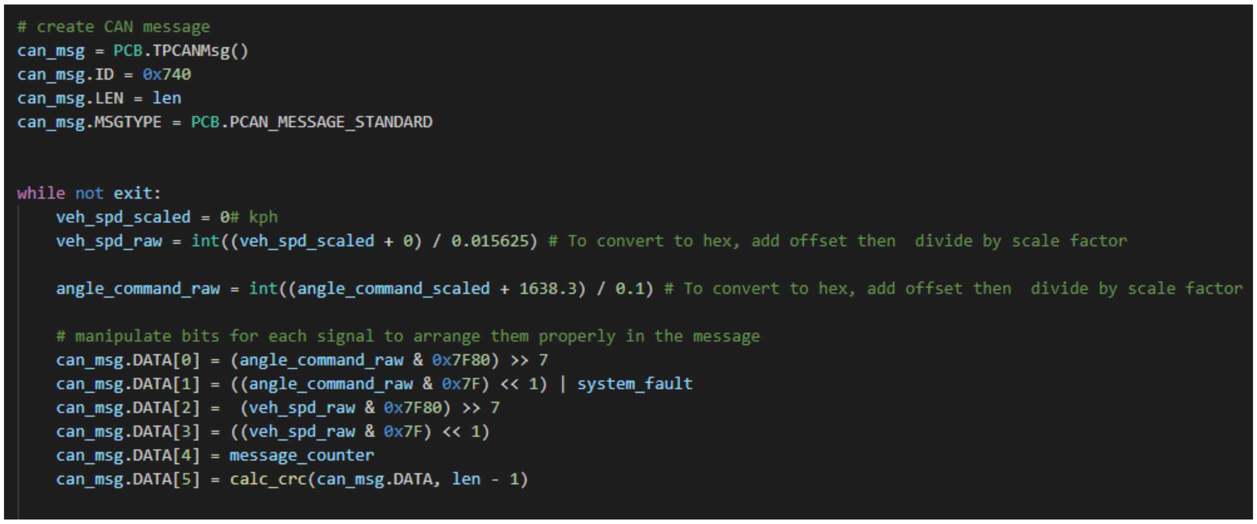


Figure 14: Circuit diagram for CAN communication

CAN communication allows for messages to be written directly to the EPS system by manipulating bits in a signal and sending them as messages to the CPU as seen in Figure 15. These bits can also be read directly from the EPS by the CPU and can serve as a loop validation on the angle of the column. By writing the messages directly to the steering column through ROS topics such as /cmd_vel, it is possible to avoid using a microprocessor and control the steering system directly from a CPU.



```
# create CAN message
can_msg = PCB.TPCANMsg()
can_msg.ID = 0x740
can_msg.LEN = len
can_msg.MSGTYPE = PCB.PCAN_MESSAGE_STANDARD


while not exit:
    veh_spd_scaled = 0# kph
    veh_spd_raw = int((veh_spd_scaled + 0) / 0.015625) # To convert to hex, add offset then  divide by scale factor

    angle_command_raw = int((angle_command_scaled + 1638.3) / 0.1) # To convert to hex, add offset then  divide by scale factor

    # manipulate bits for each signal to arrange them properly in the message
    can_msg.DATA[0] = (angle_command_raw & 0x7F80) >> 7
    can_msg.DATA[1] = ((angle_command_raw & 0x7F) << 1) | system_fault
    can_msg.DATA[2] =  (veh_spd_raw & 0x7F80) >> 7
    can_msg.DATA[3] = ((veh_spd_raw & 0x7F) << 1)
    can_msg.DATA[4] = message_counter
    can_msg.DATA[5] = calc_crc(can_msg.DATA, len - 1)
```

Figure 15: Writing Message to EPS System

4.3 Acceleration

A similar acceleration system to Cart 788 will be implemented on Cart 789 due to its ease of implementation and effectiveness at precisely controlling the acceleration. A digital potentiometer is used as a voltage divider to supply a variable voltage between 0 and 5V to the golf cart’s motor controller. The difference is that the Raspberry Pi 4 microcontroller will be used on Cart 789 instead of an Arduino. The digital potentiometer chip MCP 4151 is connected to the Raspberry Pi 4 microcontroller as shown in Figure 16 and it communicates with the golf cart’s microcontroller using Serial Peripheral Interface (SPI) communication protocol. Pins 1-3 of the potentiometer are connected to the SPI pins on the Raspberry Pi. There are three wires that connect the accelerator pedal with the golf cart motor controller. These wires were disconnected and a new Deutsch three pin connector was installed to connect with pins 5-7 on the digital potentiometer circuit.

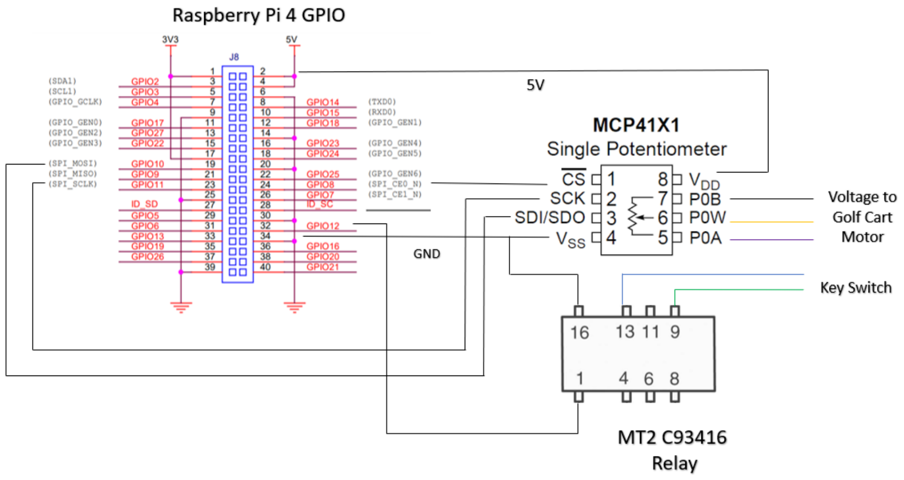


Figure 16: Circuit diagram for acceleration control on cart 789

For SPI to function properly, SPI interface had to be enabled on the Raspberry Pi in configurations and the spidev library was also installed. A value between 0 and 255 is written to the chip to change the wiper position. This outputs a variable voltage to the golf cart motor controller to accelerate the golf cart. The acceleration pedal is connected to a switch and a clicking sound is made when the pedal is pressed down. To replicate this behavior, a relay was connected to the input switch wires to turn the acceleration on when the digital potentiometer is sending varying voltage and off otherwise.

4.4 Braking

The braking system consists of a cable that is attached to a rotary motor at one end, then travels through a series of pulleys and connects to the brake pedal at the other end. When the motor rotates, it pulls the cable and thus pulls the brake pedal down.

The follower cart (Cart 789) uses a ClearPath Servo motor to pull the brake cable. This motor is depicted in Figure 17 below. The motor is attached to the golf cart with a mount that was custom-made from steel sheets, which can also be seen in Figure 17.

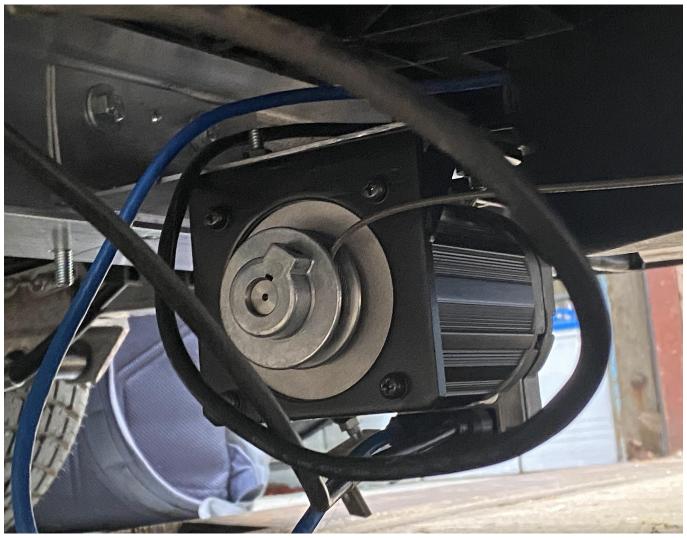


Figure 17: ClearPath Servo motor used for braking system on Cart 789 with motor mount attaching it to the steel bracket underneath the cart.

The leader cart does not currently have a braking system set up because only the follower cart needs to have autonomous capabilities for our platooning system, but if future teams would like to make Cart 788 autonomous as well, then the braking design can be replicated on that cart. Any rotary motor can be used in this design, but the ClearPath servo motor is recommended due to its encoding capabilities.

4.5 Mapping & Leader Detection

There are two components to the follower cart correctly following the driving path of the leader cart. The first component is sending pertinent data from the leader cart to the follower, and the second component is the follower cart physically detecting the leader cart. The former is feedforward, while the latter will act as feedback. These two systems will be used to create the follower cart's belief of how far away the leader cart is and whether it is turning. To achieve feedforward, the leader cart will use three sensors, two zed cameras and a LiDAR, to localize within the map. One zed camera is mounted on the front, approximately 1.448 meters in front of the LiDAR and 1.255 meters below. The second zed is mounted on the back, approximately 1.038 meters in behind the LiDAR and 0.349 meters below. Figure 18 depicts a digital layout of the sensors.

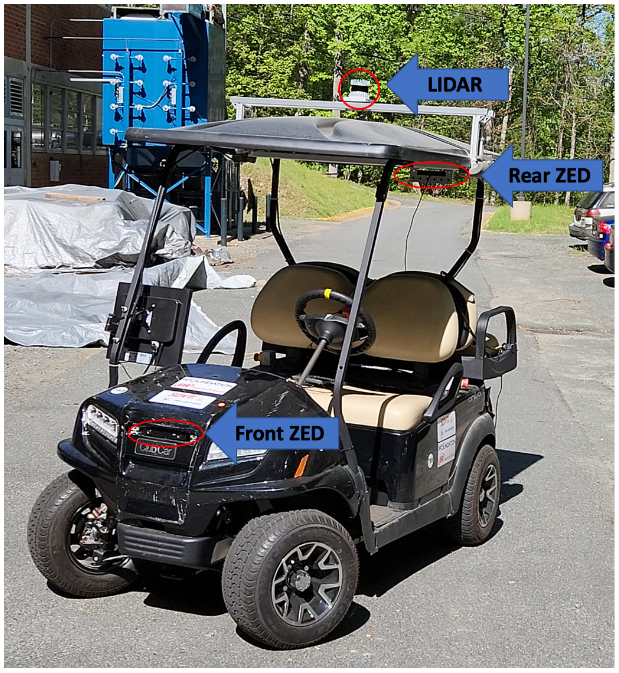


Figure 18: A depiction of the layout of the sensors.

After localization, the leader cart will communicate its location, speed, and steering commands to the follower cart. The follower cart will interpret these commands to determine proper movement.

To achieve feedback, a TeraRanger Evo Thermal 33 IR camera is placed at the front of the follower cart as shown in Figure 19.

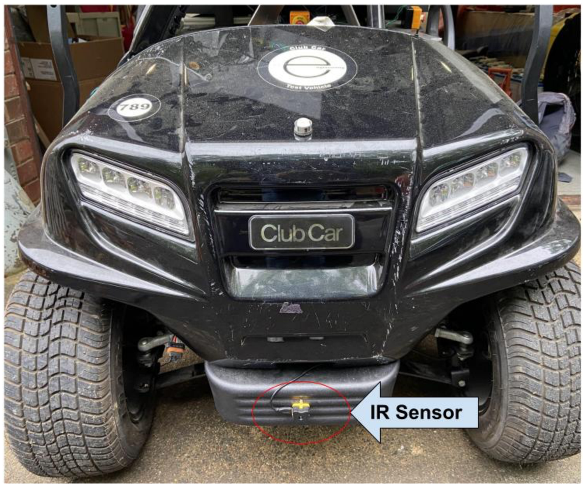


Figure 19: Location of IR sensor at the front of the follower cart.

The camera is used to detect four separate heat lamps placed on the back of the leader cart. The heat lamps are placed in a square formation as shown in Figure 20, with two at the top and two further down the sides.

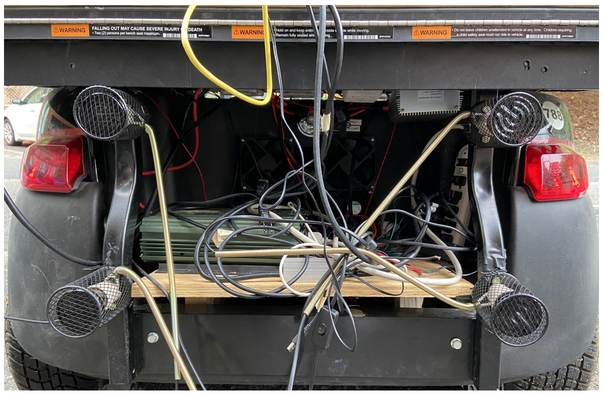


Figure 20: Four heat lamps are secured to metal bars at the back of the leader cart.

This formation was chosen to be able to more accurately differentiate between detecting changes in distance or orientation of the leader cart compared to the follower. The IR sensor maps the temperature data onto a 32 x 32 pixel image to display the location of distinct heat sources. A diagram of the heat lamp and sensor system as seen from a birds-eye view of the golf carts is shown in Figure 21.

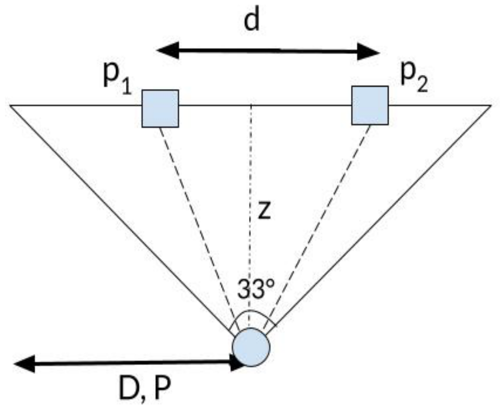


Figure 21: Birds eye view of two heat lamps (squares) and the IR sensor (circle).

The pixel locations of the heat lamps, p_1 and p_2, are identified in the pixel frame. The horizontal distance between the two heat lamps, d, is known, and so is half of the pixel viewing dimension, P. Half of the sensor's observable region range, D, is calculated using a distance-per-pixel ratio. The formula for D is shown in Equation 8.

$$D = P * \frac{d}{|p_2 - p_1|} \tag{8}$$

The distance between the two carts, z, is derived using D and half of the field of view of the sensor which is 16.5°. The calculation for z is shown in Equation 9.

$$z = \frac{D}{tan(16.5^o)} \tag{9}$$

The process of calculating D and z is repeated for every combination of two out of four lamps. These values are then averaged and any outliers are removed to obtain an average distance measurement between the two carts. If the horizontal and vertical distances between the lamps in the pixel frame are changing, this means that the distance between the two carts is changing. If only the horizontal distances between the heat lamps in the pixel frame is changing, it is known that the orientation of the leader cart is changing.

5 Mathematical/Numerical Analysis

5.1 Path Following Algorithm

To test the accuracy of the path following algorithm, the path of the leader cart needs to be compared to the path of the follower cart. First, the two paths will be discretized into a set of points. At each x coordinate, there is an associated y coordinate for the leader path and the follower path. Figure 22 is the visual representation of this idea.

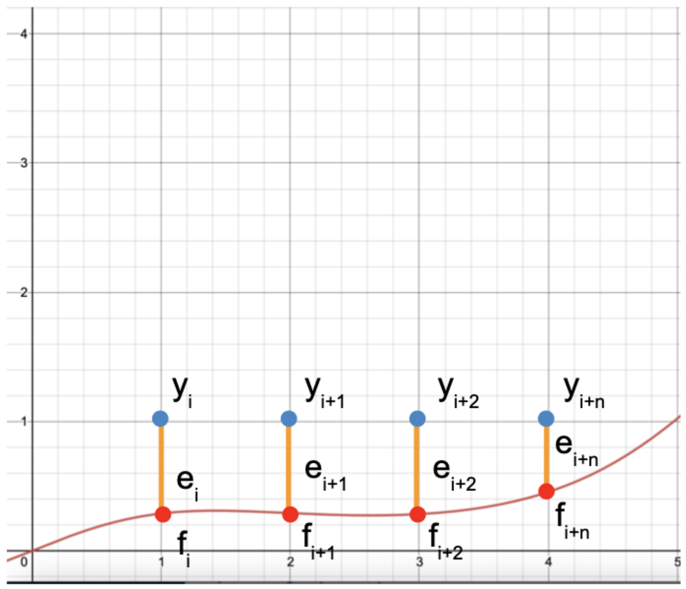


Figure 22: Visual for Residual Sum of Squares. The blue points represent where the actual point of the cart is. The red points represent a point on the given path. The orange bars represent a residual.

For each x coordinate, the residual is calculated with Equation 10.

$$e = y - f \tag{10}$$

The residual represents the error between the two paths. To find the total error, the residual sum of squares must be calculated using Equation 11.

$$SS_{res} = \Sigma\, e_i^2 = \Sigma(y_i - f_i)^2 \tag{11}$$

The residual sum of squares acts as a usual metric. However, as the path increases, the residual sum of squares will increase. Therefore, a better metric would be the residual sum of squares per unit length. Therefore, Equation 12 gives us the metric that should be minimized for optimal path following.

$$(metric) = \frac{SS_{res}}{l} = \frac{\Sigma(y_i - f_i)^2}{\Sigma\sqrt{(x_{j+1}-x_j)^2+(y_{j+1}-y_j)^2}} \tag{12}$$

6 Experimental Validation

6.1 Leader Detection

For the leader detection, the accuracy was tested by recording the measured value at known distance values of 5ft, 6ft, 7ft, 9ft, and 11ft. The percent errors from closest to farthest were 33.19, 6.34, 7.32, 2.72, and 0.096. There was a relatively large error at 5ft, but that was expected due the limited field of view of the IR camera causing the heat lamps to be partially out of frame.

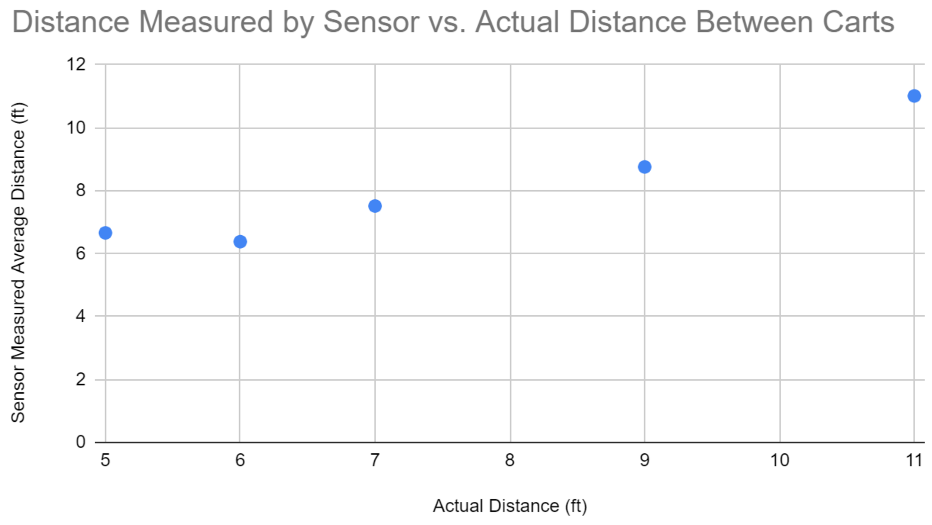


Figure 23: Test of Leader Detection Accuracy

6.2 Path Following Algorithms

To test the path following algorithm, a computer visualization software simulated the ability of the follower cart to platoon. Instead of receiving signals from the leader cart, a fictitious path was given to the follower cart to interpret. Three different paths were used to verify the platooning ability. The three paths were a slow turn to the left, a fast turn to the right, and a semi-circle. After the paths were given to the simulated follower cart, the locations of the cart were recorded to compare to the original path. Figures 24-26 shows the results.

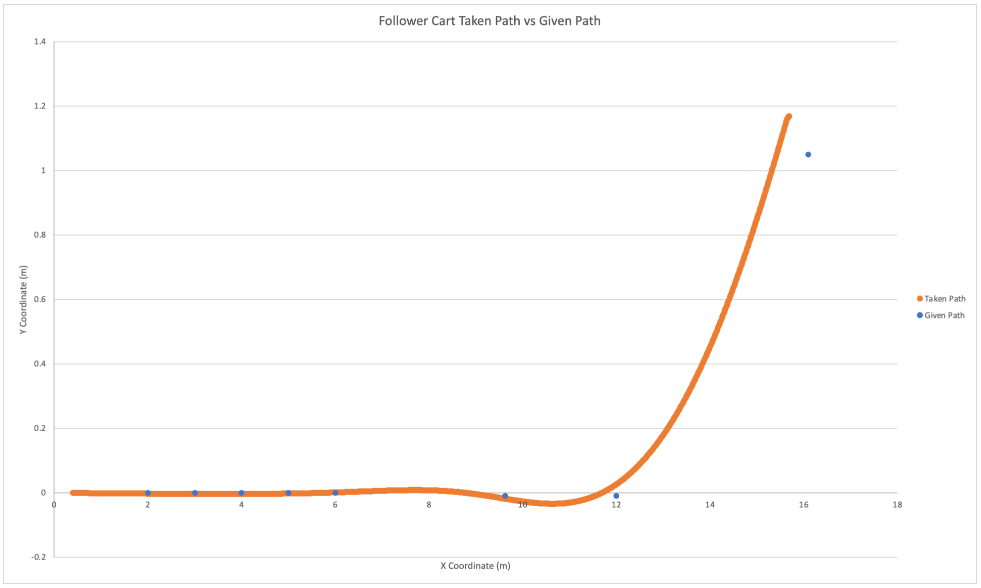


Figure 24: Test 1 of the path following algorithm. The given path is a straight line, followed by a slow turn to the left.

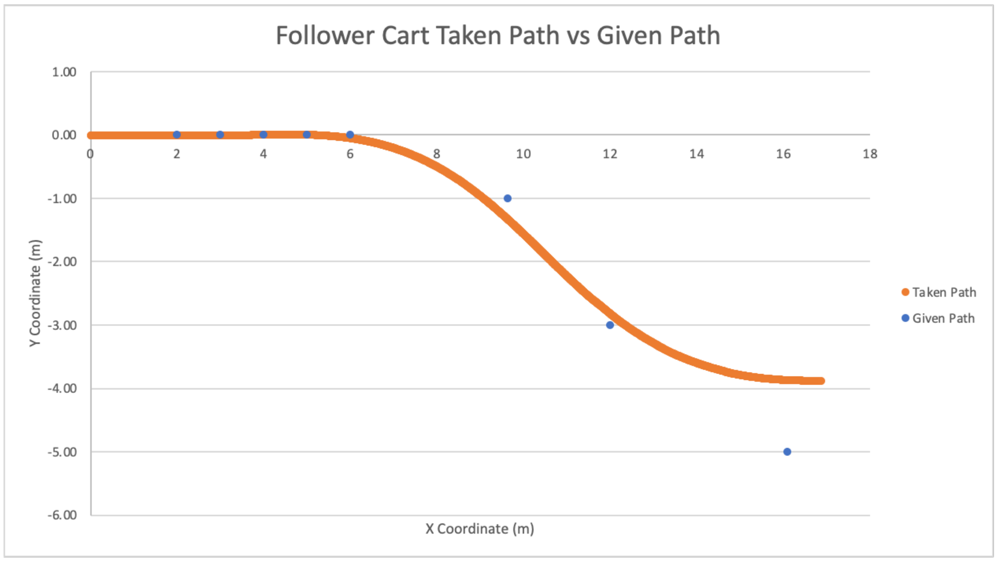


Figure 25: Test 2 of the path following algorithm. The given path is a straight line, followed by a quicker turn to the right compared to Test 1.

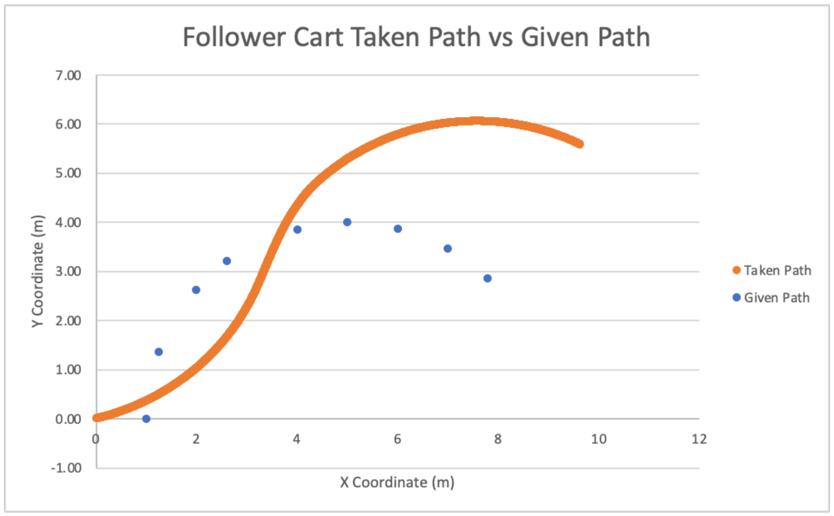


Figure 26: Test 3 of the path following algorithm. The given path is a semicircle.

Using Equations (10-12), Figure 27 shows the results. The cart performed best from Test 1 and the worst on Test 3. Therefore, the current algorithm can support straight line motion as well as a small turning angle. However, as the turning angle increases, the cart is unable to follow the path. The issues arise from steering tuning. The Stanley Controller performs properly, but the low-level control of translating the message into specific instructions for the cart needs to be tuned properly.

	Test 1	Test 2	Test 3
Residual Sum of Squares	8.62	100.23	2373.39
Total Given Path Length	13.82	15.19	6.88
Residual Squared per Unit Length	0.62	6.60	344.83

Figure 27: Summary of Results

6.3 Acceleration

The first implementation of acceleration was designed to strictly take in the cmd_vel topic values and did not correspond to actual movement in the real world. To translate these values to the real world an experiment was conducted where cmd_vel topic values were sent to the golf cart, and the time it took to reach 4 meters was measured. Knowing the distance and time it took to reach 4 meters provides the velocity of the cart in the real world. Figure 28 shows the results of the tests and a fit equation was generated.

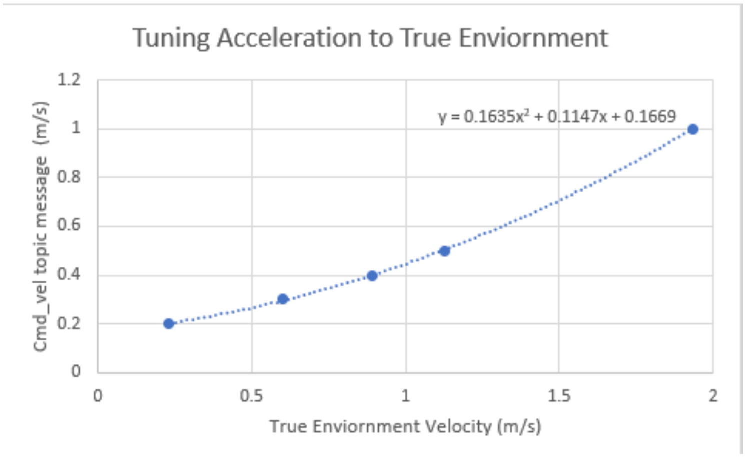


Figure 28: Tuning Acceleration to True Environment

This fit equation was then implemented into our code to ensure that cmd_vel values sent to the cart had real world implications as well. The same test as above was implemented again to test our new code and validate our work (Figure 29).

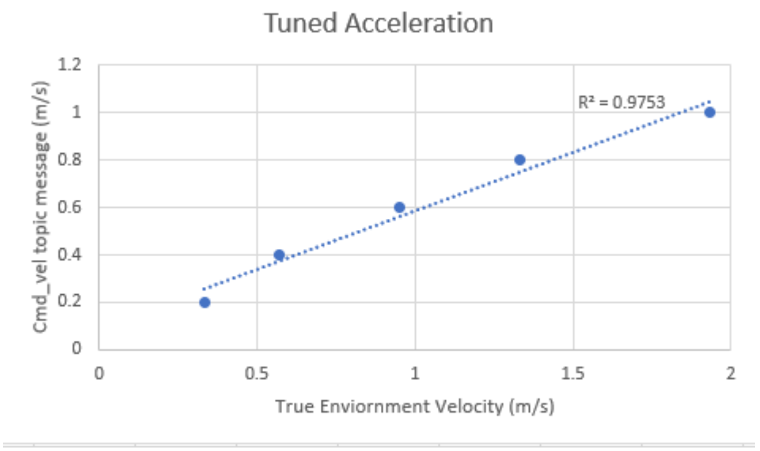


Figure 29: Tuned Acceleration

As seen in Figure 12 the new tuned acceleration code is fit with a linear trendline and a high R^2value helps validate the fact that the cmd_vel topic corresponds with a velocity in the real world.

6.4 Particle Filter

To determine the effectiveness of the particle filter, an experiment was conducted in a simulated environment to determine the effect of the golf cart's velocity on the accuracy of the estimated position. In order to conduct this simulation, a script was created with mock publishers that the particle filter node subscribes to. These mock publishers send information about the position, orientation, velocity, and steering angle of each golf cart to the particle filter. The simulation started with a distance of 3 meters between each cart for each trial and the speed of the follower cart matched the speed of the leader cart. The leader cart was then given varied inputs for velocity to determine the correlation between those velocity inputs and the accuracy of the estimated position. To measure accuracy, the actual x and y values for the follower cart were compared to the weighted average x and y values of the particle filter estimation. Equations (13-15) were then used to calculate the distance (D) between the follower cart's actual position (X_{actual}, Y_{actual}) and the particle filter's estimation of the follower cart's position $(X_{estimated}, Y_{estimated})$.

$$D = \sqrt{dx^2 + dy^2} \tag{13}$$

$$dx = X_{actual} - X_{estimated} \tag{14}$$

$$dy = Y_{actual} - Y_{estimated} \tag{15}$$

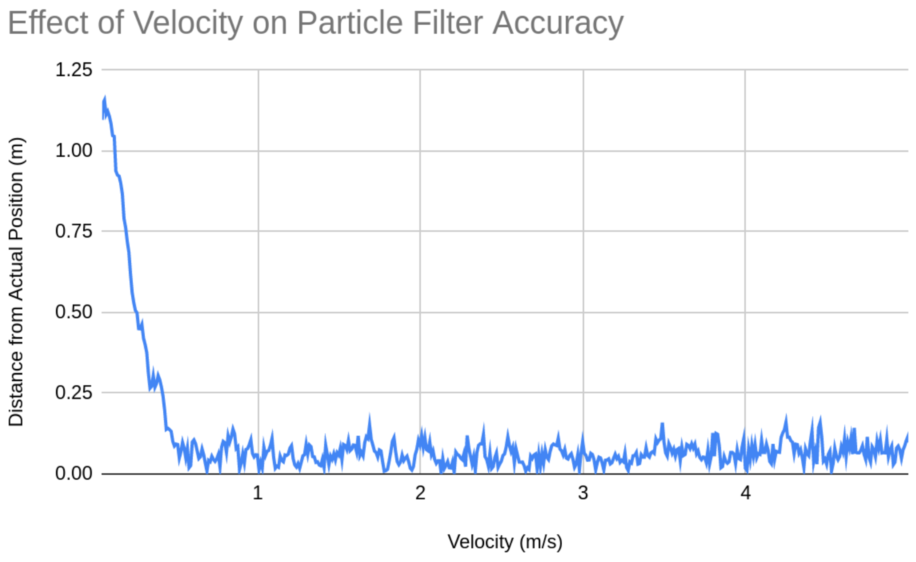


Figure 30: Effect of the golf carts' velocity on particle filter accuracy.

As shown in Figure 30 above, the particle filter has larger error when moving at velocities lower than 0.5 m/s, but at velocities of 0.5 m/s or larger, the accuracy of the particle filter is relatively stable and unaffected by velocity. These results could be due to the fact that the particle filter relies on sensor data to update the belief of the golf carts position. If the golf cart is moving too slowly, then the error of the sensor readings is larger than the difference in the golf cart's true position between readings. For example, if the golf cart's leader detection accuracy is

0.3 m, but the golf cart's change in position is only 0.2 m, then it is difficult for the particle filter to determine whether the change in position is due to the actual movement of the golf cart or due to the inaccuracy of the sensor.

This information can be used to determine the optimal velocity for platooning. When the golf cart is traveling at speeds greater than 0.5 m/s, then the average error of the estimated position is 0.0633 m. This is an acceptable accuracy for successful platooning.

6.5 Platooning

For validation of our PID controller platooning method, we set our follower cart to have a desired distance of 3 meters away from the back of the leader cart. The follower cart starts exactly at this 3-meter distance for our testing. The leader and follower carts were then started at the same time with the leader cart being driven at a constant speed in a straight line. The ZED camera on the back of the leader cart recorded the distance that the follower cart was behind for the duration of the test. Figure 31 below shows the results of our testing.

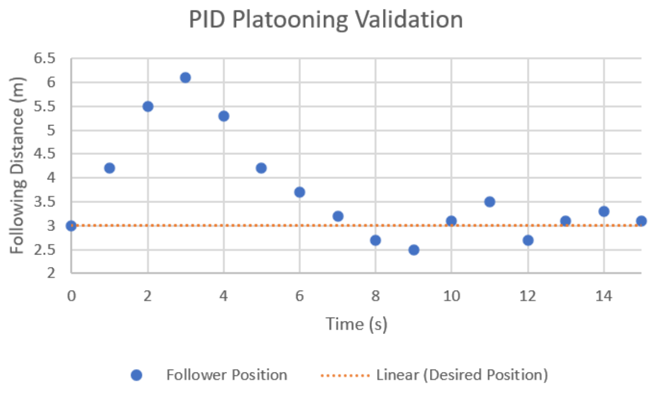


Figure 31: PID Platooning Validation

The initial peak in following distance from the start to around 3 seconds is primarily due to the acceleration relay initially receiving the signal to start up. It is also an indication that proportional gain (K_p) was too small to shorten this longer initial rise time. From 6 - 15 seconds the distance starts to hover around the desired 3 meters. When it dips below the desired 3 meter mark an acceleration is no longer applied and only is reapplied when it is farther than 3 meters away. This is also an indication that our integral gain (K_I) did its job of helping our system reach its steady state in a relatively quick manner.

6.6 Accuracy of Map

To test the map generation of the leader cart, the cart drove around the VICTOR lab three times. An aerial view of the VICTOR lab is shown in Figure 32.

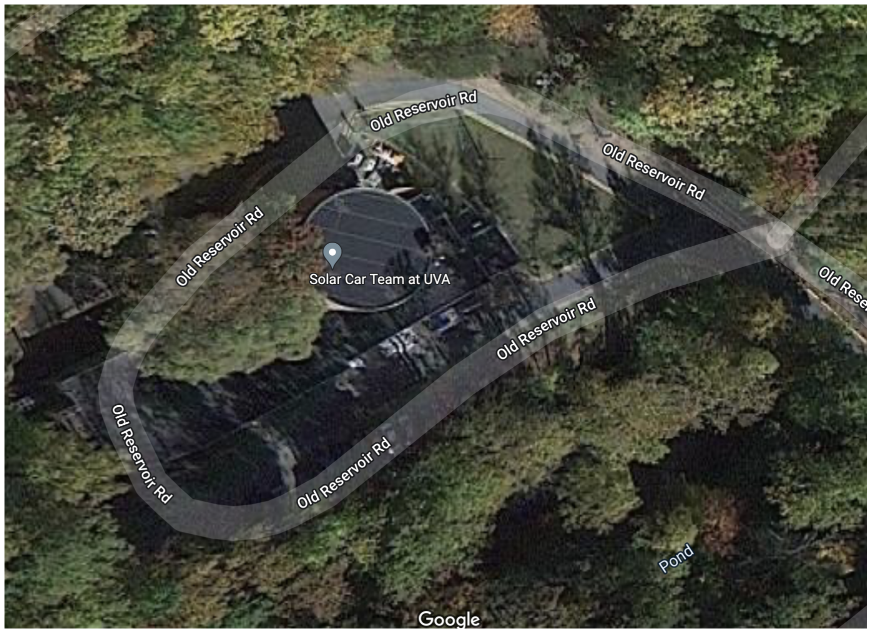


Figure 32: A Google Map image of the mapped area.

As the cart moved around the lab, the LIDAR and two ZEDs all generated a point cloud at a discretized series of locations. Synchronizing messages creates a map, shown in Figure 33.

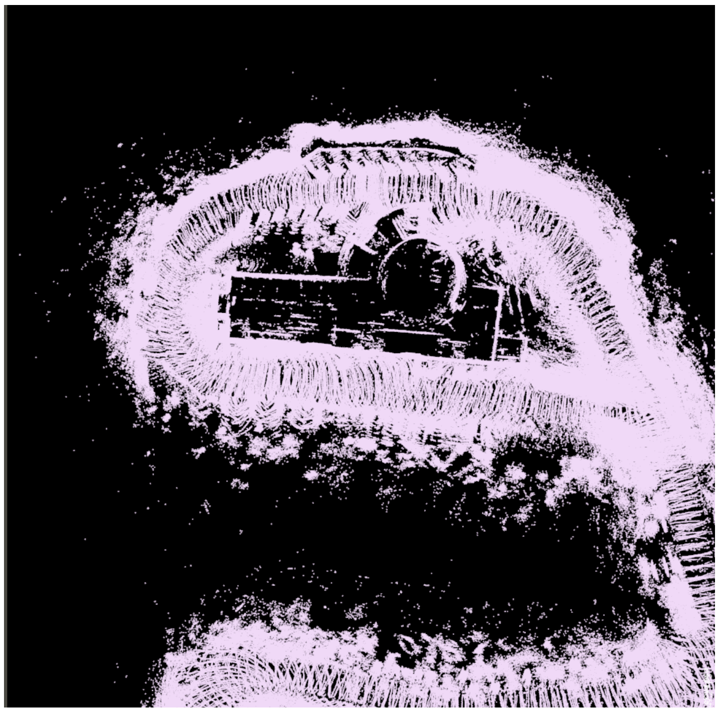


Figure 33: A point cloud visualization of the generated map from the leader cart.

Figure 34 shows the generated map overlaid on the satellite map of the VICTOR lab. The two map line up well. The outlines of the building is crisp and accurate to the real world. Most visual discrepancies from the two maps comes from the manual overlay rather than the accuracy of the map itself.

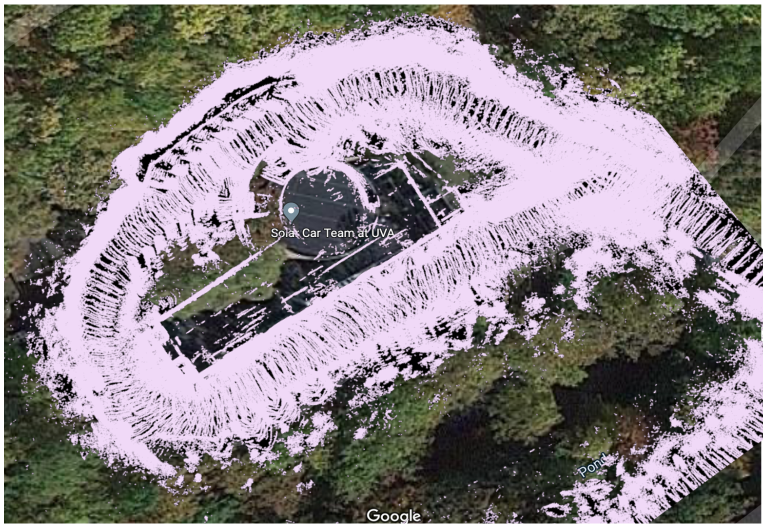


Figure 34: The generated map overlaid on Google maps.

7 Operation Manual

7.1 Setting up Leader Vehicle (Hardware)

1) Lift up the front row seats (within the cab of the golf cart)
2) There will be rows of batteries. On the wall nearest to the rear of the vehicle, there is a switch located in the center of that wall. The switch should be pointed down. Flip the switch up to “Run” to give the vehicle power.
3) On the dash of the vehicle (located to the right of the battery level), there will be a key. Turn the key to the clockwise to allow the vehicle to run.
4) Approach the rear of the vehicle. Look between the rear row and the wheels, there will be electronics in the storage area. In the back, there is a silver turning switch. Turn the switch clockwise to allow electricity to flow in the back.
5) There will be a large inverter straight in front (large green and black box). On the right side of the inverter, there should be a small switch to turn it on. Flip the switch to turn on the inverter.
6) With all the power turned on, find the computer sitting on the rear row of the vehicle. There should be a button on the right side of the computer. Press it to turn the computer on.

Common troubleshooting:

A. If the computer doesn’t power on:
   a. **The cart may be dead**: plug the cart in using the port in the cab of the vehicle.
   b. **The power strip might be off**: find the power strip in the electronics portion of the vehicle (look at step 4 to locate). The power strip will be on the right side of the vehicle. Ensure the switch is on “reset” and not “off”

c. **The computer didn't receive a signal**: wait 10-15 seconds to allow the computer to get the proper power. Press the on button again. Now the computer should be on.

d. **The computer appears off, but it is on:** When the computer is on, the fan on the computer will be spinning. Check to make sure it is on. If so, check the monitor in the cab of the vehicle. Press the power button on the monitor to turn it on.

7.2 Map Generation with the Leader Cart

1. Follow 7.1 to power the computer and sensors.
2. Find the keyboard and mouse associated with the rear computer. Use them to login to the computer.
3. Open the terminal. Type "cd ~/all_sensors" and press enter. The terminal will now be in the proper workspace.
4. In that terminal, type "source devel/setup.bash" and press enter. This terminal now correctly sourced the workspace.
5. Now, type "roslaunch zed_rtabmap_example demo_mapping.launch". This will launch all sensors and rtabmap. Generally, rtabmap usually takes ~30 seconds to launch properly. Once rtabmap launches, the vehicle is ready to drive around and map the surroundings.

Common troubleshooting:

A. ZEDs don't launch properly:

a. **The front ZED isn't plugged in**: find the ZED at the front of the vehicle. Follow the cord at the back of the ZED. It should be plugged into a USB extender near the pedal. The USB extender might be hidden below the mat. Next, check the computer in the rear to see if the USB extender is properly connected.

b. **The rear ZED isn't plugged in**: find the ZED at the back of the vehicle. Follow the cord at the back of the ZED. It should be plugged into the computer in the rear of the vehicle.

c. **Other ZED nodes are running:** Other ZED nodes might be running. First, make sure that no other programs are running. Second, make sure that there is no other "roscore" running in another terminal. Either of these can create issues with launching the ZEDs.

B. LiDAR doesn't launch properly:

a. **The LiDAR isn't powered**: Find the LiDAR box on the left side of the rear of the vehicle. It is the only thing connected to the LiDAR sensor on the top of the vehicle. Make sure the black cord is plugged into the box.

b. **The LiDAR isn't connected to the router:** Located the yellow ethernet cord from the LiDAR. Make sure that it is both plugged into the LiDAR sensor and router. Make sure the router is connected to the computer with another yellow ethernet cord. Both of the cords are connected to ports 1-4.

c. **The computer changed IP addresses:** Check the IP address of the computer. There are many ways to do this. One is to find the Angry IPScanner application. Running the application and find the IP address for golf-desktop. Next, open the file directory and go to the directory: "Home/all_sensors/src/zed-ros-examples/examples/zed_rtabmap_example/launch". Open file "lidar_ouster_2.launch". Look at the udp_dest parameter. Make sure that the string of the IP address matches what you found earlier.

7.3 Map Localization with the Leader Cart

1. Follow 7.1 to power the computer and sensors.
2. Find the keyboard and mouse associated with the rear computer. Use them to login to the computer.
3. Open the terminal. Type “cd ~/all_sensors” and press enter. The terminal will now be in the proper workspace.
4. In that terminal, type “source devel/setup.bash” and press enter. This terminal now correctly sourced the workspace.
5. Now, type “roslaunch zed_rtabmap_example demo_localization.launch”. This will launch all sensors and rtabmap. Generally, rtabmap usually takes ~30 seconds to launch properly. Once rtabmap launches, the program will attempt to localize. Drive around until you get a green box on the left of the rtabmap program. Once that occurs, the vehicle has been localized.

Common troubleshooting:

C. Rtabmap doesn’t launch properly:
    a. **There is no map:** To localize, there needs to be an existing map. Go to the file directory and click the icon with three horizontal bars and then show hidden files. Next, locate the directory “Home/.ros”. See if there is a file called “rtabmap.db”. If not, follow 7.2 to generate a map.

D. ZEDs don’t launch properly:
    a. **The front ZED isn’t plugged in**: find the ZED at the front of the vehicle. Follow the cord at the back of the ZED. It should be plugged into a USB extender near the pedal. The USB extender might be hidden below the mat. Next, check the computer in the rear to see if the USB extender is properly connected.

b. **The rear ZED isn't plugged in**: find the ZED at the back of the vehicle. Follow the cord at the back of the ZED. It should be plugged into the computer in the rear of the vehicle.

c. **Other ZED nodes are running:** Other ZED nodes might be running. First, make sure that no other programs are running. Second, make sure that there is no other "roscore" running in another terminal. Either of these can create issues with launching the ZEDs.

E. LIDAR doesn't launch properly:

a. **The LIDAR isn't powered**: Find the LIDAR box on the left side of the rear of the vehicle. It is the only thing connected to the LIDAR sensor on the top of the vehicle. Make sure the black cord is plugged into the box.

b. **The LIDAR isn't connected to the router:** Located the yellow ethernet cord from the LIDAR. Make sure that it is both plugged into the LIDAR sensor and router. Make sure the router is connected to the computer with another yellow ethernet cord. Both of the cords are connected to ports 1-4.

c. **The computer changed IP addresses:** Check the IP address of the computer. There are many ways to do this. One is to find the Angry IPScanner application. Running the application and find the IP address for "golf-desktop". Next, open the file directory and go to the directory: "Home/all_sensors/src/zed-ros-examples/examples/zed_rtabmap_example/launch". Open file "lidar_ouster_2.launch". Look at the udp_dest parameter. Make sure that the string of the IP address matches what you found earlier.

7.4 Object Detection with Leader Cart

1. Follow 7.1 to power the computer and sensors.
2. Find the keyboard and mouse associated with the rear computer. Use them to login to the computer.
3. Open the terminal. Type “cd ~/zed_ws” and press enter. The terminal will now be in the proper workspace.
4. In that terminal, type “source devel/setup.bash” and press enter. This terminal now correctly sourced the workspace.
5. Type “roslaunch zed_wrapper zed2.launch” and press enter.
6. In another terminal, check to make sure you’re in the zed_ws. If so, type “source devel/setup.bash”.
7. Lastly, type “rviz” and press enter.
8. In rviz, click the add button in the lower left corner of the program. Click the “By Topic” tab on the top. Scroll down and find “zed2”. Within “zed2”, find “/point_cloud” and click on it. It should drop down to show “/cloud_registered” and click on the “PointCloud2”. After this you should see the point cloud from the zed.
9. Next, click the add button in the lower left corner of the program. Click the “By Topic” tab on the top. Scroll down and find “zed2”. Within “zed2”, find “/obj_det” and click on it. It should drop down to show “/objects” and click on the “ZedOdDisplay”. After this you should see the point cloud from the zed.

Common troubleshooting:

A. **Bad parameters:** In “zed_ws/src/zed-ros-wrapper/zed_wrapper/params/”, there is a file called zed2.yaml. In this, make sure that “od_enable” is “true”. Select the mc_[object] that you wish to detect. Additionally, change the “confidence_threshold” if you are over/under selecting objects.

B. **The ZED isn’t plugged in**: find the ZED. Follow the cord at the back of the ZED. It should be plugged into the computer in the rear of the vehicle.

7.5 Setting up Follower Cart (Hardware)

1. Lift up the front row (within the cab of the golf cart)
2. There will be rows of batteries. On the wall nearest to the rear of the vehicle, there is a switch located in the center of that wall. The switch should be pointed down. Flip the switch up to “Run” to give the vehicle power.
3. On the dash of the vehicle (located to the right of the battery level), there will be a key. Turn the key to the clockwise to allow the vehicle to run.
4. Lift up the mat by the pedal. You will see a cluster of wires. There will be a deutsch connector with yellow, purple, and white wires coming out of the male end. Plug in purple, yellow, white end labeled “manual” to the male for the ability for a person to drive. Plug in purple, yellow, white end labeled “autonomous” to the male for the ability for autonomous driving.
5. Approach the rear of the vehicle. Look between the rear row and the wheels, there will be electronics in the storage area. There will be a rectangular gray with red dial and a battery logo centered on the dial. To power on, turn the dial to the green section. When the power is turned on, you should hear the brake motor engage briefly.
6. With all the power turned on, find the NUC on the leftahnd side of the electronics box. There should be a button on the front face of the NUC. Press it to turn the NUC on.

Common troubleshooting:

A. If the computer doesn't power on:

   a. **The cart may be dead**: plug the cart in using the port in the cab of the vehicle.
   b. **The power strip might be off**: find the power strip in the electronics portion of the vehicle (look at step 4 to locate). The power strip will be on the right side of the vehicle. Ensure the switch is on "reset" and not "off"
   c. **The computer didn't receive a signal**: wait 10-15 seconds to allow the computer to get the proper power. Press the on button again. Now the computer should be on.
   d. **The computer appears off, but it is on:** When the computer is on, the fan on the computer will be spinning. Check to make sure it is on. If so, check the monitor in the cab of the vehicle. Press the power button on the monitor to turn it on.

B. Raspberry PI doesn't turn on:

   a. **Raspberry PI isn't connected properly**: Look in the electronics box (look at step 5 to locate). Find the Raspberry PI in the box. Make sure that the red light is on the Raspberry PI. If not, unplug it and plug it back in.

7.6 Joystick Control

7.6.1 NUC Setup

1. Follow 7.5 to power the follower cart in autonomous mode.
2. Find the keyboard and mouse and plug into the NUC and login.
3. Open the terminal. Type "cd ~/steering_ws" and press enter. The terminal will now be in the proper workspace.
4. In that terminal, type "source devel/setup.bash" and press enter. This terminal now correctly sourced the workspace.

5. Type “cd /src/steering_files/src”. This folder contains all iterations of the steering code.
6. Type “python3 ModifiedSteeringCMDVEL_scaled.py”. The steering column will now be engaged and will be harder to turn
7. Run roslaunch teleop_twist_joy teleop.launch. The joystick is now on. Hold the middle logitech button and move the left stick to control.

7.6.2 Raspberry PI Setup

1. Disconnect the keyboard and mouse from the NUC and plug into the Raspberry PI and login. The password is “GolfCart”.
2. Open the terminal. Type “cd ~/testing_ws” and press enter. The terminal will now be in the proper workspace.
3. In that terminal, type “source devel/setup.bash” and press enter. This terminal now correctly sourced the workspace.
4. Type “rosrun accel_brake braking_cmd.py” and run. The brake will engage and then disengage in a second.
5. In another tab type “source devel/setup.bash” and hit enter. Run each of the following commands.
    a. sudo chmod 666 /dev/spidev0.0
    b. sudo chmod 666 /dev/spidev0.1
    c. sudo chmod 666 /dev/gpiomem
    d. rosrun accel_brake accel_cmd.py
6. The follower cart is now fully joystick controllable.

Common Troubleshooting

A. If commands are not allowed to be run. “Can not connect to localhost error”

a. Make sure that the Leader Cart's router is on.

b. The NUC and PI should also be connected to the follower carts router by ethernet.

c. Turn off WIFI on the NUC and PI, but allow wired connections.

d. Ensure a roscore is running on the leader cart.

B. Steering.

a. Ensure white CAN Bus cable is connected to NUC and power light is on.

b. Follow Figure 12 to ensure that all wires are correct on the PCB.

c. If the steering wheel is locked after code is stopped, unplug CAN Bus cable from NUC.

C. Braking and Acceleration

a. Ensure PCB shield is securely connected onto Raspberry PI pins.

b. Follow Figure 35 to ensure wires are connected correctly.

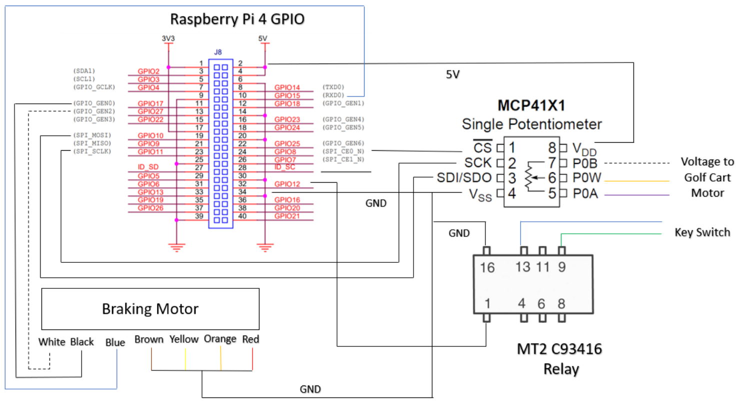


Figure 35: Raspberry Pi Acceleration and Braking Wiring

c. Ensure the car is put into forward/reverse when running acceleration.

7.7 Platooning

1. On the leader cart to set up sensor and follower detection.

a. Follow 7.1 to power the computer and sensors.

b. Open the terminal. Type “cd ~/zed_ws” and press enter. The terminal will now be in the proper workspace.

c. In that terminal, type “source devel/setup.bash” and press enter. This terminal now correctly sourced the workspace.

d. Type “roslaunch zed_ar_track_alvar _example zed_ar_track_alvar.launch”

e. Type “rosrun zed_rtabmap_example PID_w_steering.py”

2. On the follower cart to set up PID motion listening.

   a. Follow 7.5 to set up Raspberry PI.

   b. Open the terminal. Type “cd ~/testing_ws” and press enter. The terminal will now be in the proper workspace.

   c. In that tab type “source devel/setup.bash” and hit enter. Run each of the following commands.

      i. sudo chmod 666 /dev/spidev0.0

      ii. sudo chmod 666 /dev/spidev0.1

      iii. sudo chmod 666 /dev/gpiomem

      iv. rosrun accel_brake rosrun accel_brake accel_pid.py

   d. Follow 7.6.1 to set up steering.

Common Troubleshooting

A. **Orange on the follower cart is not detected by the leader cart.:** Check the Image window containing the mask. If the white outline does not appear the orange is not being detected. Different lighting conditions change the required HSV values in the BallDetect_skeleton.py code.

B. **If the follower gets too close to the leader:** Tune the Ki, Kp, Kd values in the PID.py file.

C. **Communication between both Carts:** Make sure that both routers are on and the front PC, follower NUC, and follower Raspberry Pi have connection. Check the bashrc files of each machine to make sure that they are exporting the correct IP and hostname.

7.8 Gazebo Simulation

1. Logon to the "Team 2" NUC with associated mouse and keyboard
2. Open terminal and type "roslaunch catvehicle catvehicle_neighborhood.launch". This creates the gazebo environment with the vehicle (Ford Escape).
3. Open a second terminal and type "gzclient". The gazebo program should launch and a Ford Escape should be in a small neighborhood.
4. Open a third terminal and type "rosrun catvehicle cmdvel2gazebo.py". This translates the cmd_vel topic to gazebo messages to move the vehicle.
5. Open a fourth terminal and type "rosrun catvehicle path_planner2.py". This program has a pre-programmed path and chooses the ideal point and publishes it.
6. Open a fifth terminal and type "rosrun catvehicle stanley_controller2.py". This uses the ideal published point and determines how the vehicle can reach that point. Then, the program publishes to cmd_vel.

Common Troubleshooting

A. **The terminal isn't properly sourced:** Go to the file directory and click the icon with three horizontal bars and then show hidden files. Find the .bashrc and open it. Make sure that the sourced workspace is the catvehicle_ws.

B. **Gazebo Program doesn't launch:** If you launch gzclient too quickly after the catvehicle_neighborhood, then the visual won't launch. Quit both programs and restart, allowing more time between the launches.

7.9 Leader Detection

1. Follow 7.1 and 7.5 to power on both the leader and follow cart
2. On the leader cart, plug in all four of the heat lamps; two of them are plugged into the white surge protector while the other two need to be plugged into an extension cord which is then plugged into the inverter.
3. Each lamp has a switch with a small line on one end which, when flipped down, turns on the lamp. Flip all of the switches to turn them on, ensuring that they have enough time to heat up before running leader detection code
4. On the NUC on the follower vehicle, open a terminal and type "sudo chmod 666 /dev/ttyACM0"
5. Open a new terminal and type "cd ~/golfcart_ws/src" to navigate to the folder containing the leader detection code.
6. In the same terminal, run the python file by typing "python3 leaderdetection_v3.py". You should see a 32 x 32 pixel grid pop up with four black blobs in a rectangular formation, indicating that all of the heat lamps on the leader cart are in the field of view of the IR sensor. Distance values should also be printed out in the terminal.

Common Troubleshooting

A. **Code outputs "ZeroDivisionError" instead of a distance:** If the code continues to run long enough to open the GUI, check if all four heat lamps are in the camera's view, and that the heat lamps have warmed up enough to be distinct from the environment

B. **One of the lamps is not heating up:** Check the lamp is plugged in, turned on, and receiving power. Then change the bulb in the malfunctioning lamp

8 Conclusions and Future Work

As stated above, the main goal of this project is to create an autonomous platooning system using two golf carts. The follower was able to autonomously maintain a consistent distance from the leader cart along a straight path. The steering was able to be remotely or computer controlled, but the control system needs to be refined to allow for it to be consistently used in platooning. The problems with the Nexteer EPS connection from prior years has been fixed, and the braking system and acceleration system have successfully mimicked the systems that were developed for cart 788 by prior teams. A novel concept was developed for the leader detection system that involved using a thermal sensor on cart 789 to detect four heat sources on cart 788, and an algorithm to calculate distance between the two carts was implemented.

The next steps are to coordinate the platooning controller with the results of the particle filter and leader detection systems to improve the reliability and accuracy of the follower's motion. Then, the system could be improved by fixing the platooning controls to more accurately be able to perform hard turns, and potentially utilizing the leader detection system to track the relative horizontal distance between the centers of the carts and their orientation to provide feedback on the actual motion of the versus the predicted motion of the cart. Another important feature to consider is improving and consolidating the information produced by the various subsystems into a single informative user interface to provide the average user with understandable updates and details on the motion of the carts.

References

Bellan, R. (2022, March 1). *Waymo to begin charging for robotaxi rides in San Francisco*. TechCrunch. Retrieved March 28, 2022, from https://techcrunch.com/2022/02/28/waymo-to-begin-charging-for-robotaxi-rides-in-san-francisco/

Fernandes, P., & Nunes, U. (2010). Platooning of autonomous vehicles with intervehicle communications in SUMO traffic simulator. Annual Conference on Intelligent Transportation Systems, 1313-1318. doi:10.1109/ITSC.2010.5625277

Fukao, T., Aoki, T., Sugimachi, T., Yamada, Y., Kawashima, H. (2013). Preceding Vehicle Following Based on Path Following Control for Platooning. *IFAC Proc. Vol., 46* (21), pp. 47-51

Jeyachandran, S. (2020, March 4). *Waypoint - the official waymo blog: Introducing the 5th-generation Waymo Driver: Informed by experience, designed for scale, engineered to tackle more environments*. Waymo Blog. Retrieved March 28, 2022, from https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html

Lampinen, M. (2020, August 3). *Lidars for self-driving vehicles: A technological arms race*. Automotive World. Retrieved March 28, 2022, from https://www.automotiveworld.com/articles/lidars-for-self-driving-vehicles-a-technological-arms-race/#:~:text=It%20is%20used%20by%20a,up%20to%2060%20metres%20away.

Smith, B. (2013, December 18). *Human error as a cause of vehicle crashes*. Center for Internet and Society. Retrieved March 20, 2022, from http://cyberlaw.stanford.edu/blog/2013/12/human-error-cause-vehicle-crashes

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., … Mahoney, P. (2007). Stanley: The robot that won the DARPA Grand Challenge. Journal of Field Robotics, 23(9), 661–692. https://doi.org/10.1002/rob.20147

Tsugawa, S. (2014). Results and issues of an automated truck platoon within the energy ITS project, Proc. 25th IEEE Intell. Veh. Symp., pp. 642-647.

Tsugawa, S., Jeschke, S., & Shladover, S. E. (2016). A review of truck platooning projects for energy savings. *IEEE Transactions on Intelligent Vehicles, 1*(1), 66-77. doi:10.1109/TIV.2016.2577499

Waymo. (2022). *Waymo Company History*. Waymo. Retrieved March 28, 2022, from https://waymo.com/company/#story