Developing an Autonomous Campus Vehicle

A Technical Report submitted to the Department of Mechanical Engineering

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Jack Yocom
Spring, 2021
Technical Project Team Members
Art Ken Fontelera
JeeSoo Shin
William Smith
Peter Wellman

Professor Tomonari Furukawa, Department of Mechanical Engineering

1. Introduction

Research in the autonomous vehicle sector has been increasing exponentially in recent years. This new emerging technology has the potential to drastically change the world from where it is today. For example, autonomous vehicles will diminish the amount of car accidents across the country, and some autonomous cars take 15% less space to park, which could cause real estate to change in cities across the globe (CB Insights, 2018). A plethora of universities and private companies are trying to further develop sensors, actuators, and machine learning code in order to make driverless cars commonplace. Here, the development of an autonomous golf cart is to function as an exhibit for the University of Virginia Mechanical and Aerospace Engineering Department and to give Club Car a model for developing autonomous vehicles of their own.

Several large public companies are researching and developing autonomous vehicle technology for widespread use. Tesla vehicles now ship with all of the hardware necessary to achieve full vehicle autonomy, limited in capability only by software (Ingle, 2016). The system includes 8 cameras that provide 360-degree camera coverage, a radar sensor, and several ultrasonic sensors. All cameras are equipped with heaters to melt any snow and ice that would obscure their view (Lambert, 2016). Much like its namesake in airplanes, Tesla's "Autopilot" does not currently provide full autonomy, as the driver is fully expected to take control of the vehicle in dangerous situations. Some drivers do not understand this and are unprepared when the vehicle fails to make a lane change or unexpectedly changes speed (Dikmen, 2016). Nevertheless, Autopilot is expected to improve greatly over time with software updates.

Under Google's parent company Alphabet, Waymo has been independently developing a fleet of autonomous taxis for use in urban areas since 2017 (Waymo, 2021). In their current fleet, each vehicle uses radar and lidar sensors to map surroundings. These sensors are not susceptible to glare and can also detect obstacles at much larger distances than cameras (Tabora, 2020). However, they are bulkier, more expensive, and lack the imaging capabilities that make cameras useful for recognizing signs and pedestrian gestures. Waymo therefore uses long-range and perimeter cameras to supplement the radar and lidar data (Jeyachandran, 2020). By increasing certainty of object detection through sensor fusion, Waymo has begun operating its vehicles without a human backup driver in Phoenix (White, 2020).

In addition to sensors that perceive the vehicle's surroundings, vehicle autonomy requires drive-by-wire systems that control steering, braking, and acceleration. Hydraulic power steering systems are increasingly becoming electronic, allowing a computer to control the motion of the steering mechanism. In such systems, the mechanical link between the steering wheel and the road wheels is completely eliminated (Amberkar, 2004). In manual operation, the driver's steering commands are measured by an encoder, and a power steering motor receives position data from a central controller. Similarly, modern braking systems use electromechanical actuators to apply a braking force according to input from a human driver or a computer (Xiang, 2008). Electronic throttle control also replaces mechanical linkages with sensors, actuators, and controllers to change engine speed (Garrick, 2006). In electric vehicles, the accelerator pedal controls the speed of the car’s electric motor. All sensors and motion systems work together to form a vehicle that can sense and react to its surroundings, much like a human driver would.

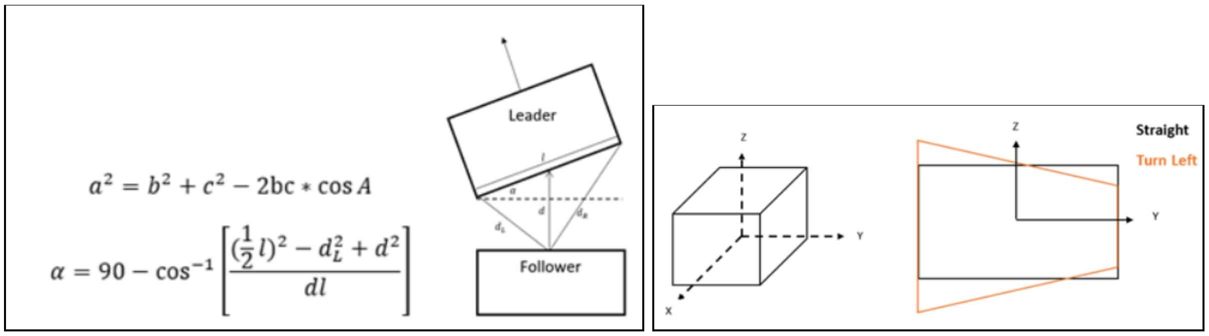
The main objective of this project is to develop a safe autonomous golf cart that acts as a campus vehicle for patrons of the University. Each cart should drive a loop between Observatory Mountain Engineering Research Facility (OMERF) and the High Energy Physics Laboratory, stopping to allow people to get into or out of the vehicle. In order to do this, the vehicle must detect and avoid obstacles, yet also recognize passengers trying to get into the cart. It must be able to interact with its surroundings using sensors and interact with passengers through a user interface. Lastly, the carts should not be overly complex or expensive.

This report will first expand on the essential knowledge at play in the project. Next, an examination of the mechatronic systems that were already implemented on the carts before this semester will be done. Design decisions that made changes to the carts will be justified and the progress made over both semesters will be illuminated. Tests completed to ensure cart safety will be discussed, followed by an operation manual for anyone trying to run the cart. Lastly, conclusions will be made in order to provide stepping stones for future team’s success with this cart.

2. Essential Knowledge

Past Work

Despite not being two independently self-driving carts, principles of autonomy are present in the carts and serve as a convenient starting point. Upon the completion of the 2018-2019 academic year, a “leader-follower” system was implemented. The two-cart system works on the basis of the “leader” cart sending sensor data to the “follower” cart via Bluetooth, and the “follower” cart would act accordingly (hence, autonomously). Data such as kinematics, steering and braking actuation were continuously sent. How the right data was converted to and from signals/code was through a series of trigonometric calculations. This is because the “follower” relies on its camera to make kinematic adjustments staying on the follower trajectory – it “sees” in 2D rather than 3D. Example calculations are shown below in Figure 3.



$a^2 = b^2 + c^2 - 2bc * \cos A$
$\alpha = 90 - \cos^{-1}\left[\frac{(\frac{1}{2}l)^2 - d_L^2 + d^2}{dl}\right]$
Leader
l
α
d_L
d
d_R
Follower
Z
Y
X
Z
Straight
Turn Left
Y

Figure 1: Left) Law of Cosines used to calculate turn angle of leader vehicle; Right) Field of view of sensors used to calculate angle of rotation.

To understand the work done by previous teams, each of the two carts were thoroughly examined to produce system diagrams. Cart 788 was the follower cart for the 2018-2019 team and is the most complete autonomous system. It includes additional actuators, sensors, switches, and processors alongside the stock components highlighted in green.

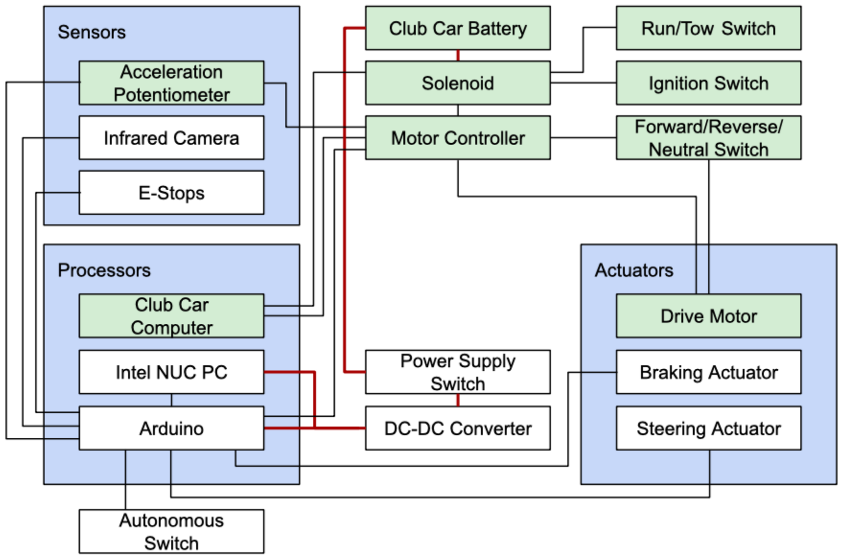


Figure 2: Cart 788 system diagram; red lines represent power, black lines represent signals

As shown, the batteries are directly connected to a solenoid that receives input from run/tow and ignition switches. The run/tow switch serves as a master lockout for the cart's power supply. The batteries also connect to a separate power supply switch and DC-DC converter that powers a separate autonomous system. A switch fitted to the dashboard is used to enable or disable this system through an Arduino microcontroller. In actuality, the Arduino in the system diagram represents several different microcontrollers, each assigned to various autonomy tasks. The connected Intel NUC PC facilitates communication between sensors and actuators over ROS serial nodes.

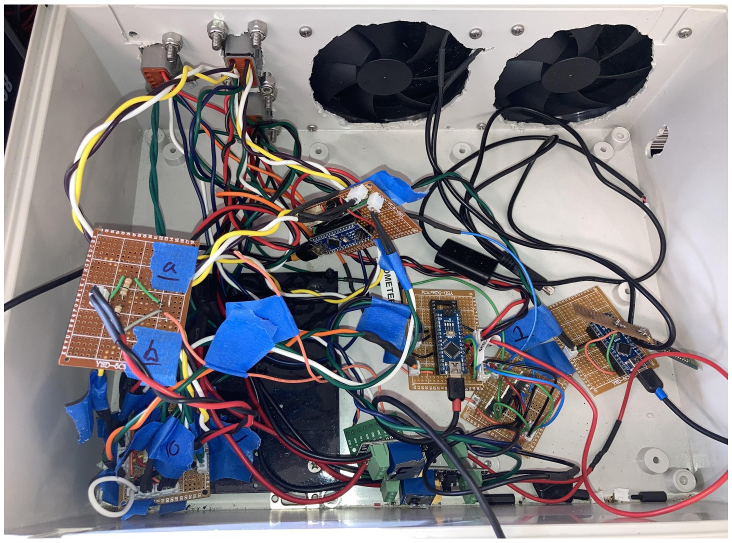


Figure 3: Cart 788 controls with Arduino Nano boards connected to various components

The powered braking system in this cart consists of a geared DC motor that connects to the brake pedal via cabling wrapped around three pulleys. To actuate the brake pedal, this cable is wound tighter by the motor, pulling the pedal closer to the floor. The cart's steering is also directly manipulated by a geared DC motor as part of a Nexteer power steering system. An infrared camera mounted near the roof was originally used to identify the position and spacing of beacons on the leader cart through blob detection in OpenCV.

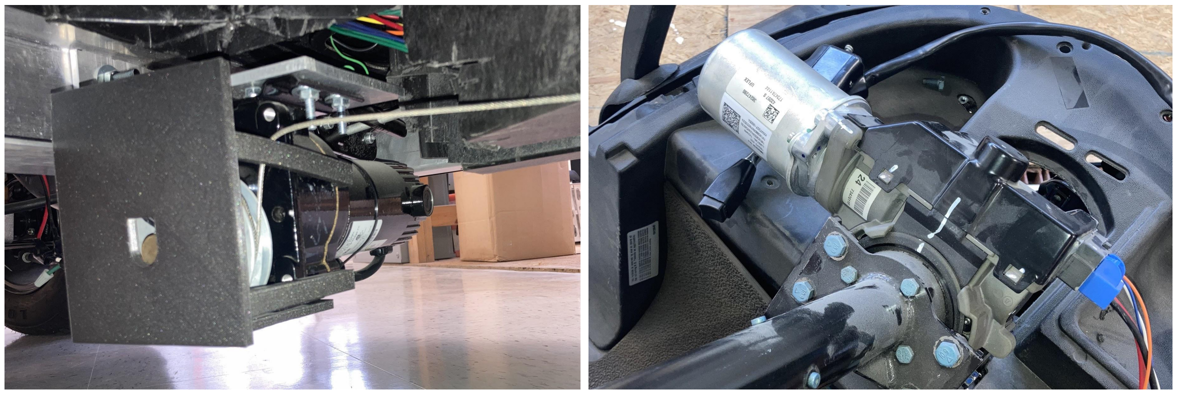


Figure 4: Geared DC motor with attached cable for braking; Nexteer electric power steering system and steering column

Cart 789 was the leader cart for the previous team and thus does not possess all components necessary for autonomy. Most notably, there is no connected braking actuator and no sensors are present. The cart was intended to be driven manually and communicate steering angle to the follower cart via Bluetooth.

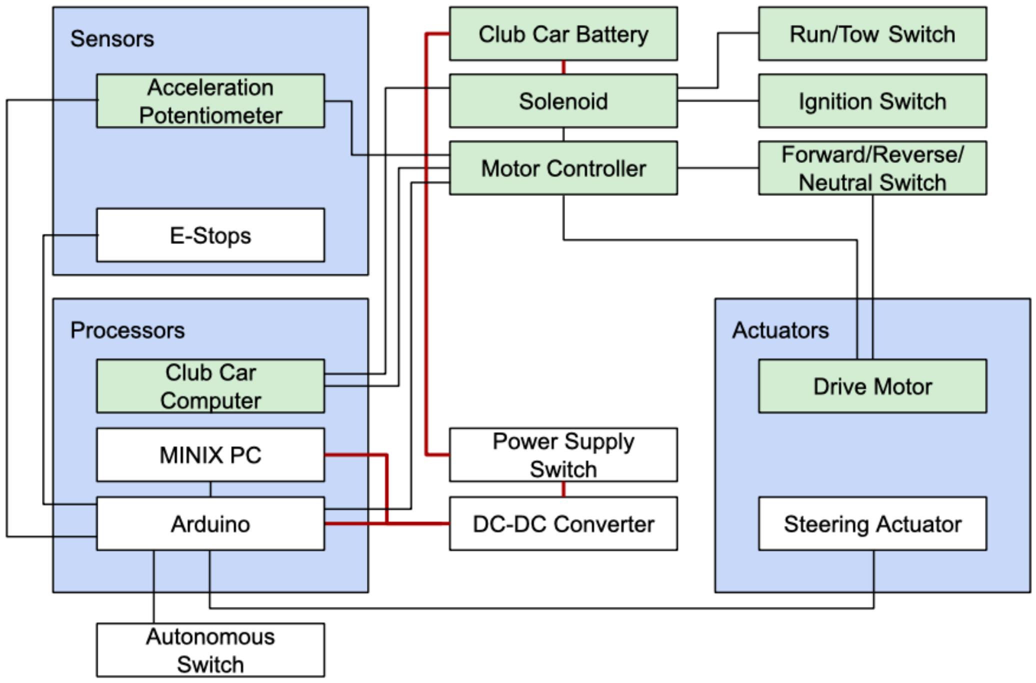


Figure 5: Cart 789 system diagram; missing brake actuator and sensors

When the team began work on this cart in September, two of the four 8-volt batteries were not able to be charged. The previous team had left the DC-DC converter connected for an extended period of time and the batteries were completely drained. The affected batteries were promptly replaced, allowing the cart to be driven around the lab. As in cart 788, a geared DC motor fitted to the steering column allows for electronic control of steering wheel angle. An attached potentiometer serves as an encoder, measuring the angle. The controls at the rear of the cart are simplified, with only a single Arduino Nano, Bluetooth module, and Intel MINIX PC.

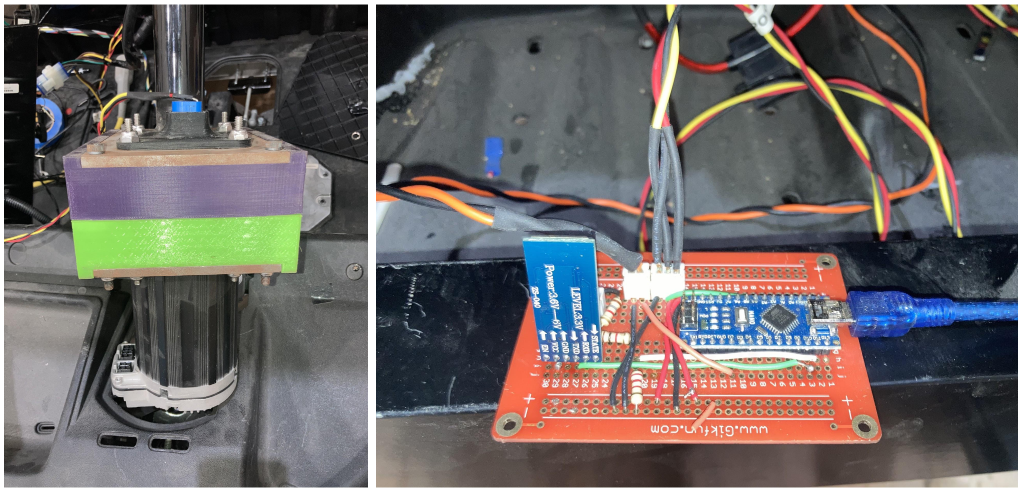


Figure 6: Geared DC motor and potentiometer for steering with 3D-printed casing; communications module with Arduino and Bluetooth boards

Upon further examination, a disconnected Misumi RSD306 linear actuator was found on the bottom of the cart. Previous teams had attempted to use this actuator to control braking, but found difficulty in achieving the precision required to gradually stop the vehicle. Also visible was a system of linkages used to connect the actuator to the braking cables. This design did not allow for simultaneous manual operation of the brake pedal, as the existing linkage needed to be disconnected to use the linear actuator.

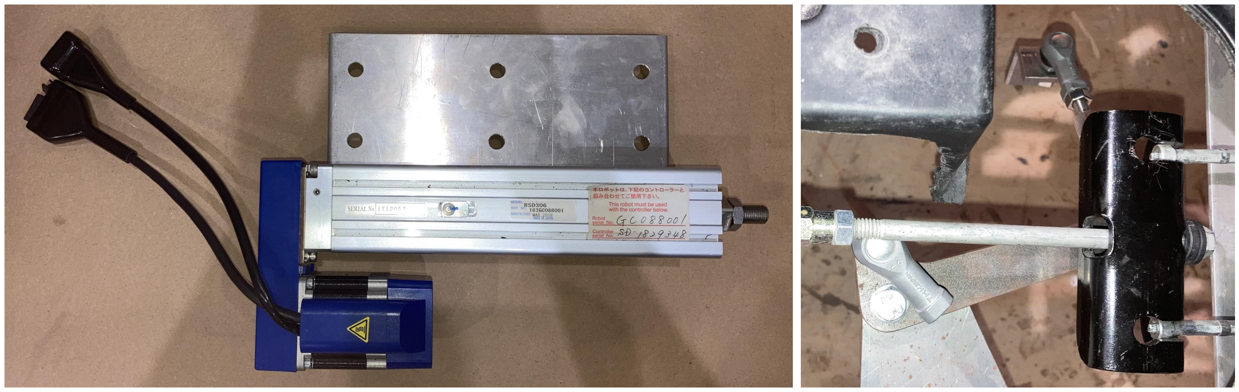
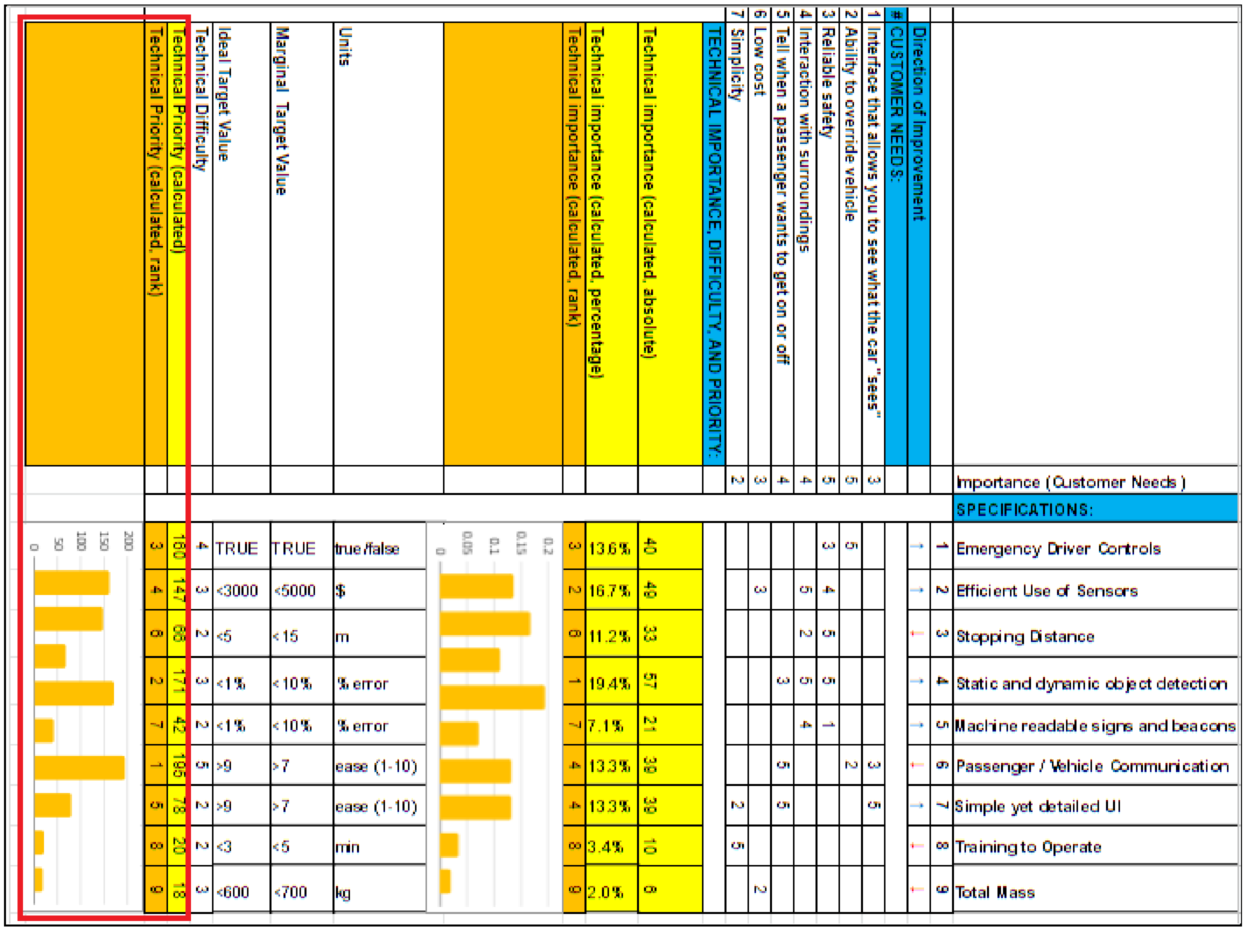


Figure 7: Misumi RSD306 linear actuator and associated braking linkages

3. Design Process

Customer Needs Identification

Information on customer needs were gathered through “interviews” with the “customers.” This information was then generated into a Quality Function Deployment (QFD) and applied to the Concept Scoring Chart. In the QFD (shown in Figure 1), customer needs were listed in its respective section, and were rated on a scale of 1 to 5 based on importance, 1 being the least important and 5 being the most important. Precise technical features/specifications were then derived from the customer’s needs and listed (under the SPECIFICATIONS section). Any correlation between the customer needs and their respective specification were identified by placing a number wherever is needed. That number, again on a scale from 1 to 5, was used to rate how much correlation exists. Technical importance of each specification was then calculated, taking into account how much of the customer’s needs were attributable/applicable to a specification. Finally, technical priority of each specification was then calculated, which takes into account difficulty of achieving a certain specification. This information allowed the team to determine which engineering characteristics required more time and were deemed more important than the other. Passenger/vehicle communication resulted in having the most priority, with static/dynamic object detection closely following.



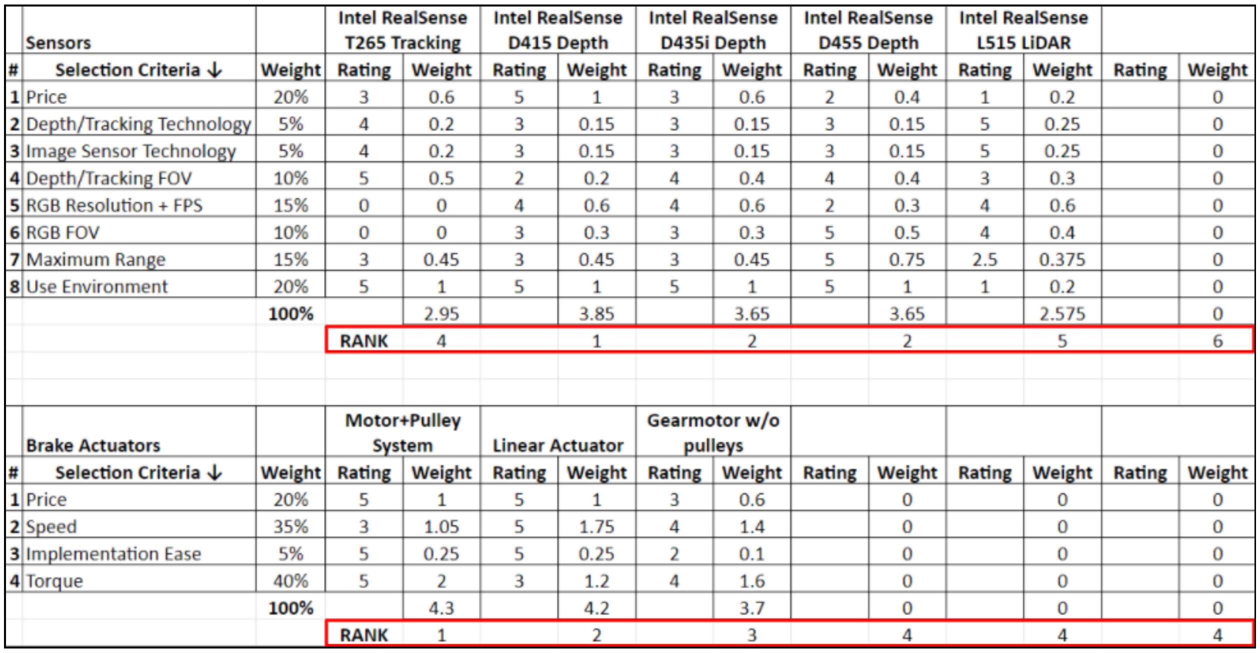
#	SPECIFICATIONS:	Direction of Improvement	1 Interface that allows you to see what the car "sees"	2 Ability to override vehicle	3 Reliable safety	4 Interaction with surroundings	5 Tell when a passenger wants to get on or off	6 Low cost	7 Simplicity	Technical importance (calculated, absolute)	Technical 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)		3	5	5	4	4	3	2									
1	Emergency Driver Controls	↑		5	3					40	13.6%	3	true/false	TRUE	TRUE	4	160	3
2	Efficient Use of Sensors	↑			4	5		3		49	16.7%	2	$	<5000	<3000	3	147	4
3	Stopping Distance	↓			5	2				33	11.2%	6	m	<15	<5	2	66	6
4	Static and dynamic object detection	↑			5	5	3			57	19.4%	1	% error	<10%	<1%	3	171	2
5	Machine readable signs and beacons	↑			1	4				21	7.1%	7	% error	<10%	<1%	2	42	7
6	Passenger / Vehicle Communication	↓	3	2			5			39	13.3%	4	ease (1-10)	>7	>9	5	195	1
7	Simple yet detailed UI	↑	5				5		2	39	13.3%	4	ease (1-10)	>7	>9	2	78	5
8	Training to Operate	↓							5	10	3.4%	8	min	<5	<3	2	20	8
9	Total Mass	↓						2		6	2.0%	9	kg	<700	<600	3	18	9

Figure 1: Quality Function Deployment Chart; customer needs listed vertically in top row; technical priority and respective rankings of specifications boxed in red

Concept Generation and Selection

The team needed to generate concepts for two features – a camera and a braking actuation. For determining which camera would be ideal, five candidates were listed. Selection criteria, along with their weight, were then listed as well. Selection criteria for each candidate was then rated from 1 to 5, 1 being the least desirable and 5 being the most desirable. The Intel RealSense D415 Depth camera resulted in being the most ideal camera (despite not being used). The same process was used for determining which braking system would be ideal, with its respective candidates and selection criteria. The motor + pulley system resulted in being the ideal braking system. Scoring methods can be found in Figure 2.

After careful consideration, it was decided that another camera sensor be added to aid in the overall visibility of the system, assisting the 360° camera by covering its blind spots. Stereo cameras, having two lenses, allows users to create images that will appear three-dimensional, giving the perception of depth, unlike 360° cameras. Depth perception would enable us to pinpoint how far nearby objects are, and that information can be used when programming the cart's object detection and reaction accordingly (i.e. brake).



	Sensors		Intel RealSense T265 Tracking		Intel RealSense D415 Depth		Intel RealSense D435i Depth		Intel RealSense D455 Depth		Intel RealSense L515 LiDAR			
#	Selection Criteria ↓	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight
1	Price	20%	3	0.6	5	1	3	0.6	2	0.4	1	0.2		0
2	Depth/Tracking Technology	5%	4	0.2	3	0.15	3	0.15	3	0.15	5	0.25		0
3	Image Sensor Technology	5%	4	0.2	3	0.15	3	0.15	3	0.15	5	0.25		0
4	Depth/Tracking FOV	10%	5	0.5	2	0.2	4	0.4	4	0.4	3	0.3		0
5	RGB Resolution + FPS	15%	0	0	4	0.6	4	0.6	2	0.3	4	0.6		0
6	RGB FOV	10%	0	0	3	0.3	3	0.3	5	0.5	4	0.4		0
7	Maximum Range	15%	3	0.45	3	0.45	3	0.45	5	0.75	2.5	0.375		0
8	Use Environment	20%	5	1	5	1	5	1	5	1	1	0.2		0
		100%		2.95		3.85		3.65		3.65		2.575		0
			RANK	4		1		2		2		5		6

	Brake Actuators		Motor+Pulley System		Linear Actuator		Gearmotor w/o pulleys							
#	Selection Criteria ↓	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight
1	Price	20%	5	1	5	1	3	0.6		0		0		0
2	Speed	35%	3	1.05	5	1.75	4	1.4		0		0		0
3	Implementation Ease	5%	5	0.25	5	0.25	2	0.1		0		0		0
4	Torque	40%	5	2	3	1.2	4	1.6		0		0		0
		100%		4.3		4.2		3.7		0		0		0
			RANK	1		2		3		4		4		4

Figure 2: Concept Scoring Spreadsheet for Cameras; final ranks boxed in red

Camera Mounts

After deciding the types of cameras needed, their placements needed to be finalized. To begin, the ideal location of a camera that provides a 360^o view would be above the canopy of the cart. In addition, the camera would have to be raised to compensate for the canopy covering a majority of the camera's vertical field of view. The canopy prohibits the cart from seeing what is in close range with respect to its vertical field of view. It was decided that the mount be raised 4 inches, resulting in a 7.6^o field of view below the horizontal (shown in Figure 3).

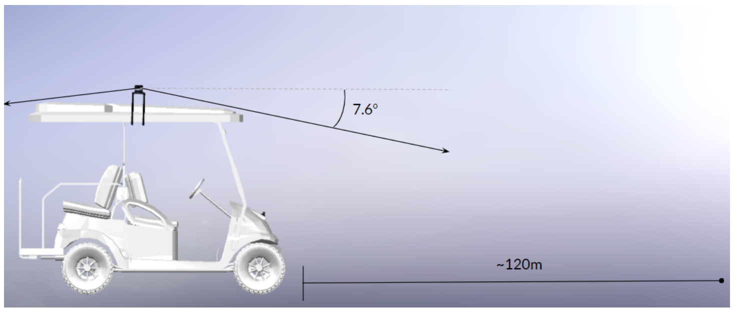


Figure 3: Approximate field of view for the 360^o camera sensor.

With the amount of height used to raise the sensor, a reliable mount design was needed for the camera. To ideate and provide a starting point of what the camera mount would look like, a simple design was created (illustrated in Figure 4).

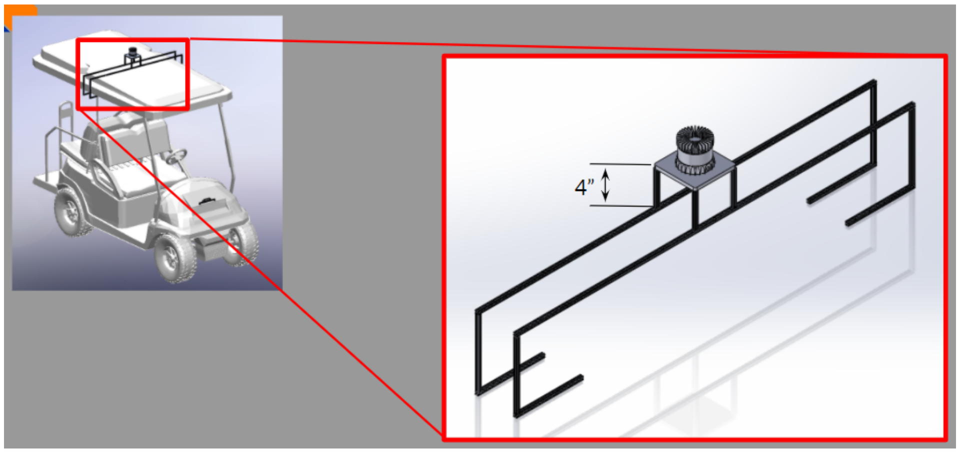


Figure 4: Draft of 360^{o} camera mount.

It was observed that the initial 360^{o} camera’s mount was not a structurally integrous design. It was predicted that this design would promote vibration, since the canopy of the cart vibrates itself. Since the canopy serves as the base of the mount, the vibration of the canopy would cause a chain reaction up the mount, where the amplitude would be greatest at the apex, which is where the camera is located. This setup would then heavily affect the data and image information gathered by the sensor, containing inaccuracies and large margins of error. To reduce the possibility of the mount being affected by the elements and various forces, sturdier hardware and fastening pieces were needed. Instead of having the 360^{o} camera being held up by four thin vertical beams, the decision was made to have it be lying on two thicker horizontal beams. Another important design decision was to sandwich vibration dampeners between the camera and the two horizontal beams. The main convincing factor for this design is that it takes advantage of the extrusion beams already present under the canopy of the cart, so it was convenient and feasible to extend linkages around the canopy. The final design is presented below in Figure 4.

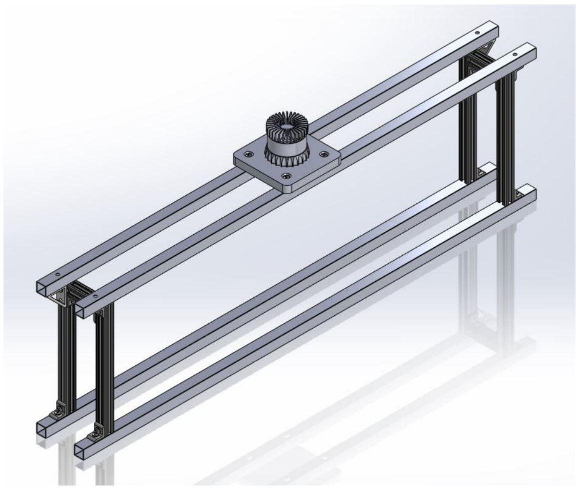


Figure 4: Final design of 360^{o} camera mount.

For the front-facing camera, the first ideation is presented in Figure 5. The main idea of the first prototype is to have the camera sit on top of the golf cart hood. However, it was realized that placing the camera on top of the hood would block a substantial amount of the camera's field of view below.

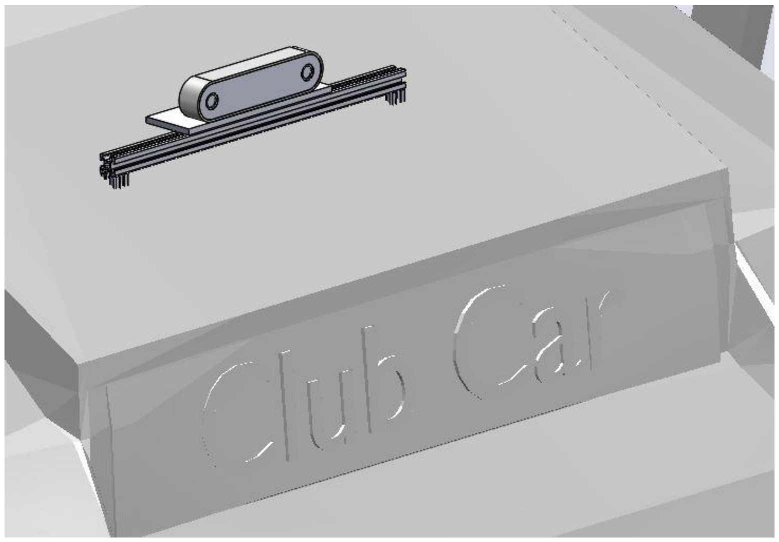


Figure 5: Draft of front-facing camera mount.

As a result, the front-facing camera mount was moved to the front of the hood. Like the 360^o camera mount, a series of fasteners, brackets, and extrusions were needed to reduce vibration as much as possible. Another strategy implemented, to reduce the amount of avoidable hardware protruding from the cart, was to place the mount inside the golf cart hood. Material needed to be removed from the hood to give access to the space needed for installation. The final design of the front-facing camera mount is displayed in Figure 6.

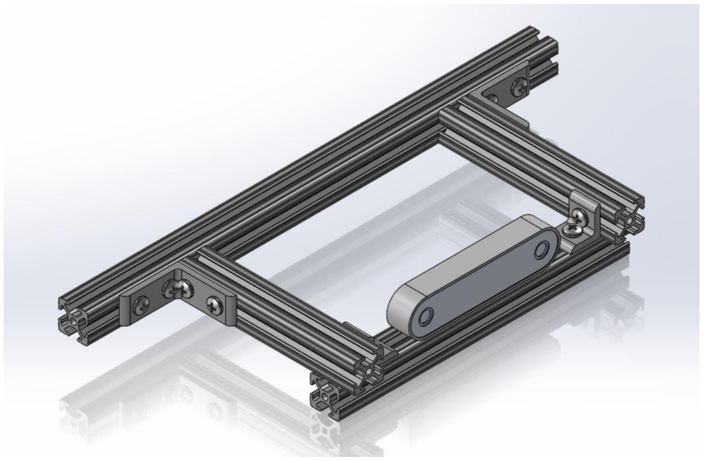


Figure 6: Final design of front-facing camera mount.

4. Final Design

Sensors

There was a clear initial impression that the cart was lacking sensors. Previously, there was only a singular camera on the front of the cart, which was loosely mounted, and no documentation exists in order to understand how to use it. Thus, the decision was made to drastically expand and update the sensor systems. First, the type of sensors had to be decided upon. In studying past research and observing sensor implementation practices, it was clear that just one camera or LiDAR would not be sufficient for safely operating the cart autonomously. Thus, a stereo camera, the ZED 2, and an Ouster LiDAR are used in tandem in order to detect objects and map the areas which the cart can drive. The ZED 2 has a range of 20m, with a vertical and horizontal field of view of 70° and 110°, respectively. It has been placed at the front of the cart, within the hood in order to diminish vibrations, allowing it to detect objects in front of and close to the cart. The Ouster LiDAR has a 360° horizontal and 45° vertical field of view. It has been mounted to the canopy of the cart. The canopy blocks the LiDAR from seeing 13.4 meters in front of or behind the cart and 7.5 meters to either side. These restrictions are illustrated in Figure 3. Using trigonometry, it was determined that the LiDAR will be able to detect objects one meter from the side of the cart that are at a height of 1.55 meters, and one meter in front or behind the cart at a height of 1.66 meters.

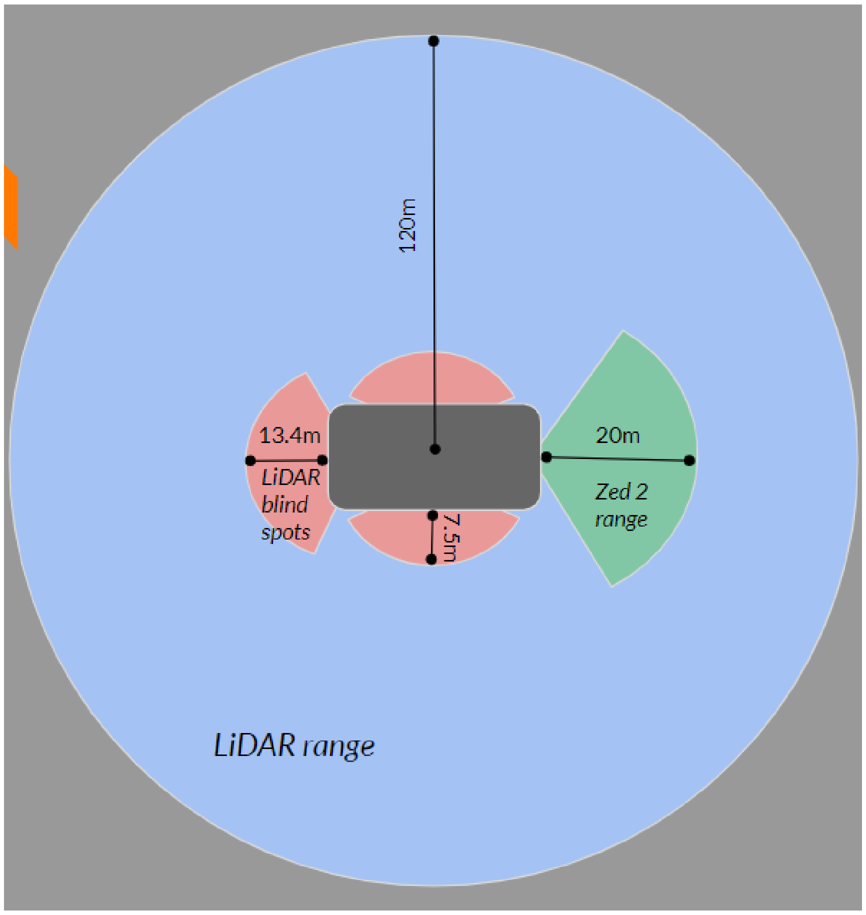


Figure 1: Field of views for the LiDAR and ZED cameras.

Point clouds are formed in rviz along with mapping data to track where the cart has driven. Currently, these two sensors output their own data, but the team is working to combine the two data sets to create a more complete map. Both sensors are connected to the cart via aluminum extrusions built by the team. In addition, another mount has been placed on the back of the cart in case future teams decide they need to integrate another camera.

Braking System

There are three main systems to control when operating the golf cart: braking, acceleration, and steering. In order to electronically control these systems, they were analyZED to determine how they operate with a human driver. Human behavior and control were also taken into consideration when designing the electrically controlled systems so that the golf cart behaved similarly regardless of the operation mode.

The first system tackled is arguably the most important since it is key for safety, the braking system. Cart 788 already had a large DC gear motor installed and mechanically implemented with the stock braking system. The motor was connected to the brake pedal with high strength airline cable through a series of pulleys, as seen in Figure 2. The airline cable was connected directly to the pedal to utilize the mechanical advantage provided by the lever. It also allows the user to see the brake being applied and could increase the comfort by visually knowing it is working.

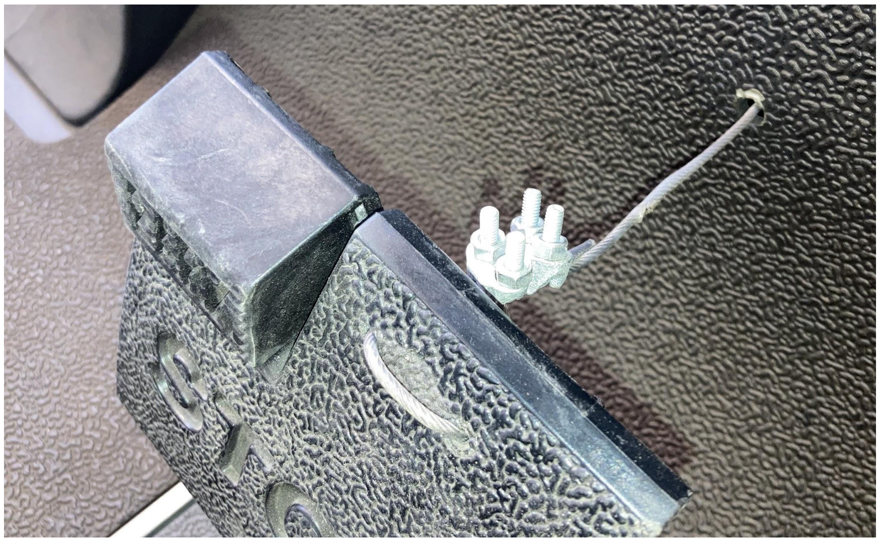


Figure 2. Airline cable secured to the brake pedal

The motor was not electrically integrated with the golf cart or microcontrollers, so a multi turn potentiometer was added to the motor shaft, as seen in Figure 3. The potentiometer was used to act as an absolute encoder so that the exact position of the motor could be known and with proper calibration, the position of the brake pedal.

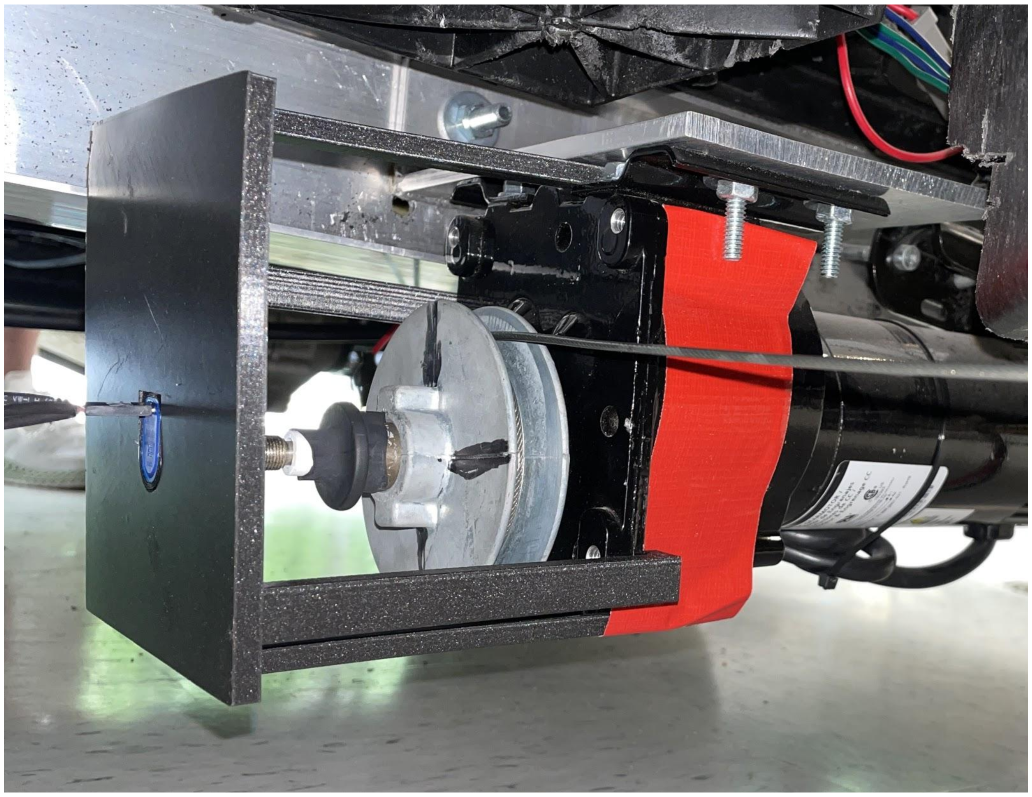


Figure 3. DC gear motor mounted on the underside of the golf cart with the potentiometer mounted on the left and held stationary with a 3D printed support

The motor had to be controlled electronically, so a high voltage, high current motor shield was used to control the motor through an Arduino, as seen in Figure 4. The motor is powered by 24 volts which is sourced from a DC to DC converter which drops the battery voltage down. The Arduino controls the speed and direction of the motor, through a pulse width modulation (PWM) pin and a standard digital pin respectively. The potentiometer was set up as a voltage divider so that the position of the wiper could be read through one of the Arduino's analog pins.

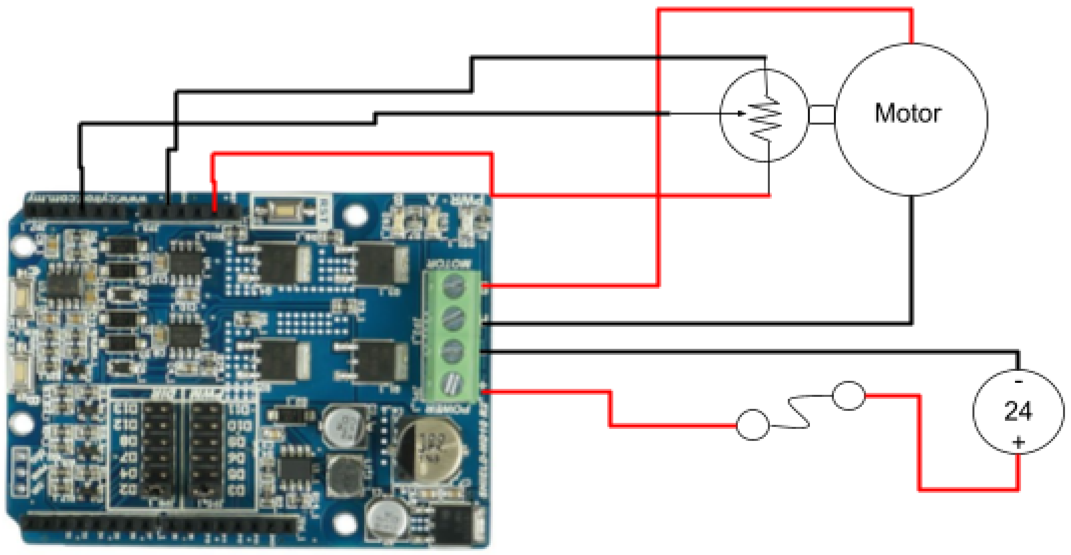
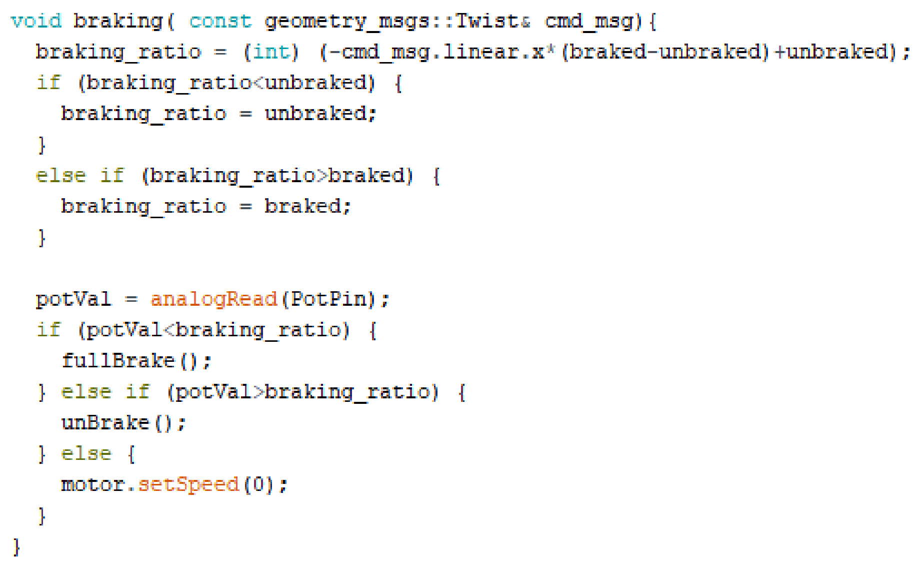


Figure 4. Circuit diagram of the braking motor setup

The Arduino was coded to be controlled using ROS so that future upgrades and implementations would require minimal adjustments to the Arduino code. A ROS node was started on the Arduino and subscribed to the topic cmd_vel of type geometry_msgs::Twist. The subscribed topic is a common message type so that various control methods can be used with the Arduino such as a joystick or navigation stack. The subscriber method of the Arduino code processes and determines the action for the braking motor, as seen in Figure 5. First the data is extracted from the message and scaled to match the potentiometer data. The data is then processed to ensure that the requested braking position is within a calibrated range based on physical limits. Finally the motor is commanded to either wind or unwind the brake depending on the current and desired position. This loop moves the motor a small step closer to the desired position so that if a new position is sent, the motor can quickly respond to the new command.



```
void braking( const geometry_msgs::Twist& cmd_msg){
  braking_ratio = (int) (-cmd_msg.linear.x*(braked-unbraked)+unbraked);
  if (braking_ratio<unbraked) {
    braking_ratio = unbraked;
  }
  else if (braking_ratio>braked) {
    braking_ratio = braked;
  }

  potVal = analogRead(PotPin);
  if (potVal<braking_ratio) {
    fullBrake();
  } else if (potVal>braking_ratio) {
    unBrake();
  } else {
    motor.setSpeed(0);
  }
}
```

Figure 5. ROS subscriber method for the Arduino which allows control of the braking motor

Acceleration System

The next system worked on was the acceleration system, since the golf cart needs to be able to move for the other systems to work properly. First the existing system was analyzed to determine how pressing the pedal instructed the drive motor to move. The acceleration system is made up of four major components: accelerator pedal, forward/reverse switch, motor controller, and drive motor. The accelerator pedal has two sets of signal wires going to the motor control box. The first set of signal wires represent a safety switch in the pedal so that the cart does not wander when the pedal is not pressed. It appears that there is little to no acceleration when the pedal is pressed less than one fourth of the way after overriding the safety line. The switch can be audibly heard when the pedal is pressed about one fourth of the way down. The second set of signal wires is a potentiometer which tells the motor controller how fast to drive. In analyzing the system, it was determined that emulating human behavior would be the most efficient method for controlling the acceleration.

The pedal was emulated using a digital potentiometer that was controlled by an Arduino, as seen in Figure 6. Pins 1, 2, and 3 of the potentiometer were wired up to the Arduino so that a digital pin could send messages to the digital potentiometer using serial peripheral interface (SPI). Pins 4 and 8 of the potentiometer were used to power the chip off of the ground and five volt pins of the Arudino respectively. Pins 5-7 of the potentiometer made up the voltage divider component of the circuit which were connected to the signal wires connected to the motor controller.

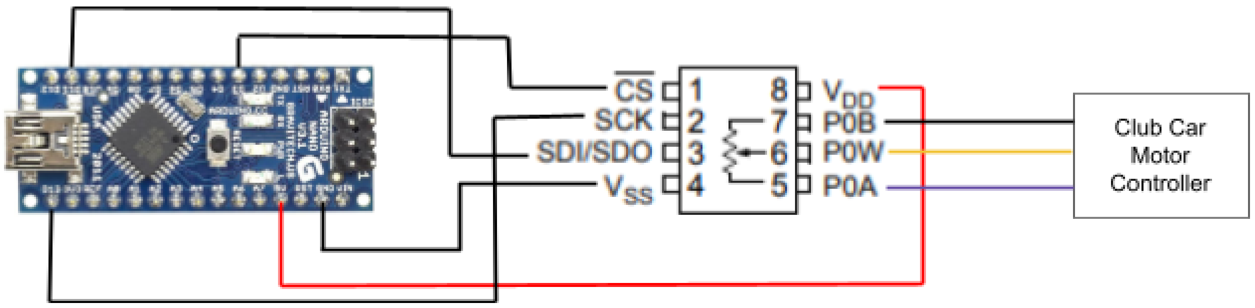
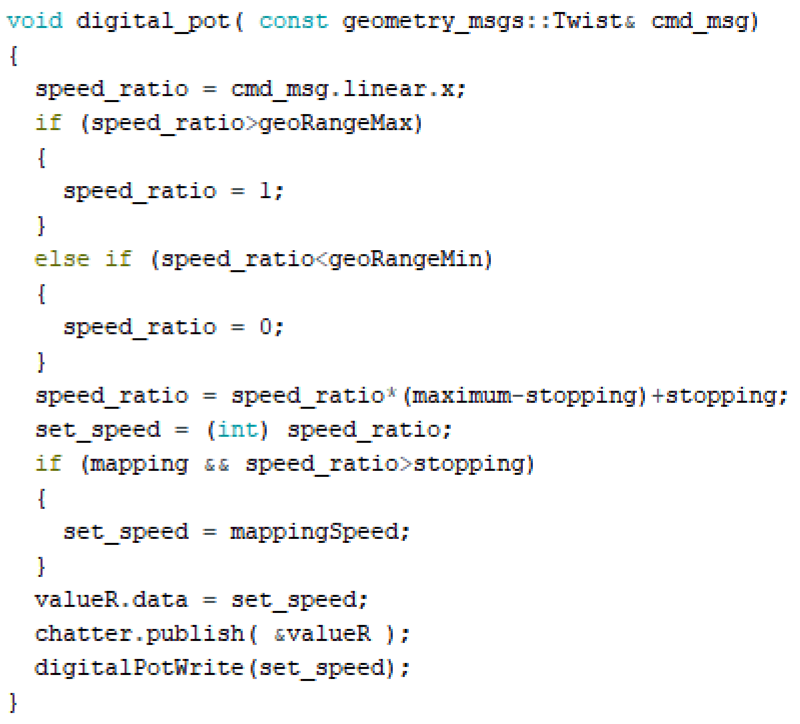


Figure 6. Circuit diagram of the acceleration system

Using ROS messages as the sole means of communication with the Arduino, the desired speed or acceleration of the golf cart can be achieved. The code subscribes to messages published by the joystick and extracts the pertinent information from the message, as seen in Figure 7. The speed is then compared to a range to ensure the messages are on a scale of 0 to 1. Next the speed is scaled experimentally calibrated values so that 0 corresponds to no acceleration and 1 corresponds to the maximum desired speed. These calibrated values were determined to provide an optimal range for operation since this cart stops on the lower end and does not go too fast on the upper end. The speed is now in the range of 0 to 255 since the digital potentiometer has an eight bit resolution, with each integer value representing a different step of the potentiometer. The speed is sent to the digital potentiometer through SPI which causes the wiper to digitally turn and thus create an analog voltage across the wiper line. The analog voltage is then read by the motor controller and the cart begins to accelerate.



```
void digital_pot( const geometry_msgs::Twist& cmd_msg)
{
  speed_ratio = cmd_msg.linear.x;
  if (speed_ratio>geoRangeMax)
  {
    speed_ratio = 1;
  }
  else if (speed_ratio<geoRangeMin)
  {
    speed_ratio = 0;
  }
  speed_ratio = speed_ratio*(maximum-stopping)+stopping;
  set_speed = (int) speed_ratio;
  if (mapping && speed_ratio>stopping)
  {
    set_speed = mappingSpeed;
  }
  valueR.data = set_speed;
  chatter.publish( &valueR );
  digitalPotWrite(set_speed);
}
```

Figure 7. ROS subscriber method for the Arduino to control acceleration

Steering System

The last system to control was the steering system and it was the most difficult to get working properly. The original motor used was a part of the Nexteer power steering system but due to communication problems with the motor it was swapped for a Teknic ClearPath servo motor. The ClearPath motor was mounted parallel to the steering column, as seen in Figure 8, and connected via a pair of gears to allow for more precise movement.

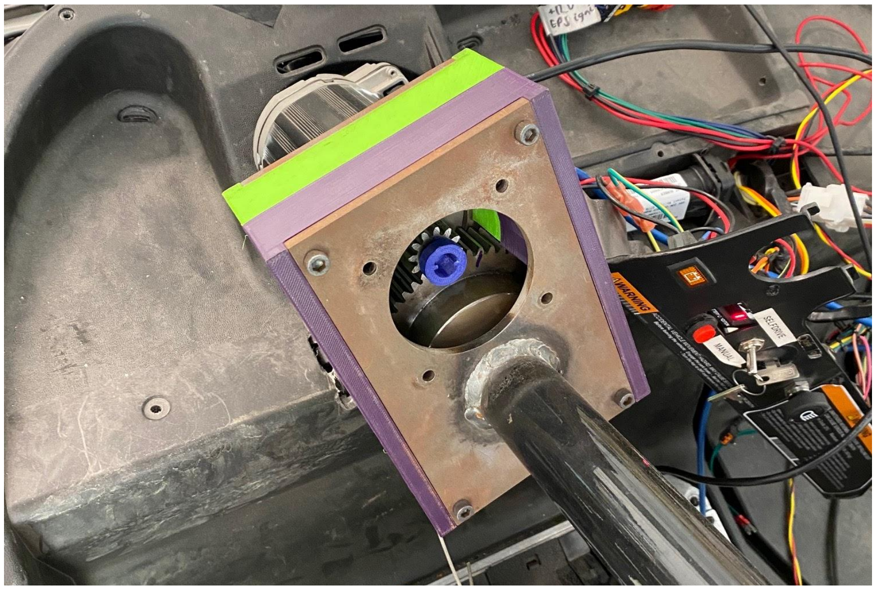


Figure 8. The Teknic ClearPath servo motor steering system. The gears are housed in a steel and 3D printed enclosure.

The motor has two cables connected to it, one being power and the other being communication. The power cable was connected to the 48 volt (battery voltage) rails in the rear of the cart. The communication cable was spliced on the Arduino side to reveal the multi-stranded wires underneath. The wires were then stripped so that they could be connected to the Arduino properly. The communication wires were connected as guided by the motor user manual with all of the negative wires being grounded to the Arduino and the positive wires being connected to digital pins, as seen in Figure 9. Since pin 4 was connected to the high level feedback (HLFB) wire from the motor, it was set to be an input while the other active pins were set to be outputs. According to guidance from the user manual, the HLFB + wire was also wired with a pull up resistor to ensure proper readings from the feedback sensor.

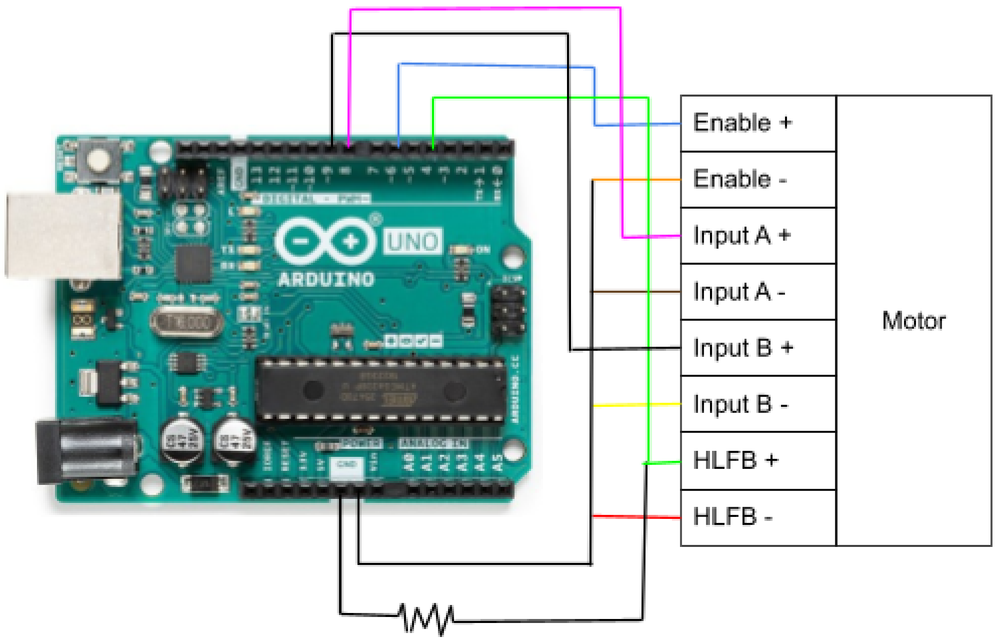
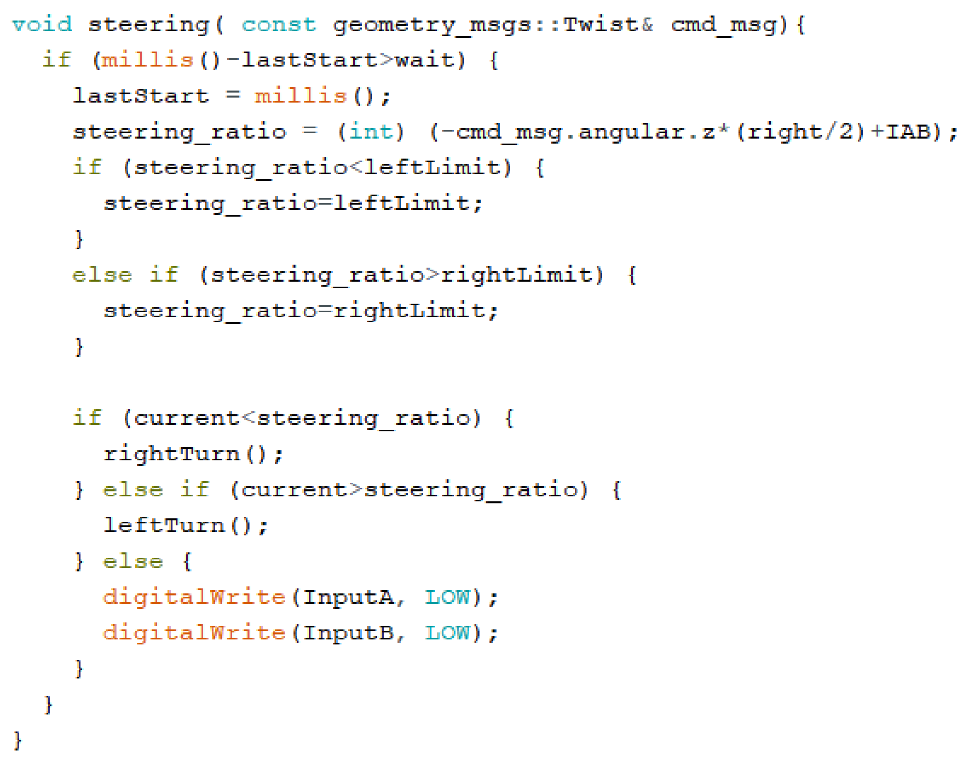


Figure 9. Circuit diagram of the steering system

The motor has numerous modes and tuning parameters that can be programmed through the ClearPath MSP app. Each mode requires the enable to be HIGH for the motor to be engaged and commands to be accepted. The mode selected was “Move to Absolute Position” which allows the motor to move a set number of steps, based on the relative encoder readings. The motor is set to HOME on the first enable after power up by turning to the left until a torque threshold is reached. This homing sets the zero position for the relative encoder values to be used for estimating the absolute position. This mode allows for four different movements based on Inputs A and B being HIGH or LOW. To send a command to the motor, then Enable pin was triggered to LOW for 30 ms and then the inputs were interpreted to execute the command. The motor has a sixteen command buffer, with one of those commands being the current command, if this buffer is exceeded the motor throws a “Move Buffer Overrun” error and the motor shuts down. To avoid this error, the Arduino checks to make sure commands are not sent too fast to ensure continuous commands are sent without shutting down the motor. The position of the motor is tracked on the Arduino using the same step sizes programmed in the MSP app and assumes no error, so after long uses there is a possibility of error compounding.



```
void steering( const geometry_msgs::Twist& cmd_msg){
  if (millis()-lastStart>wait) {
    lastStart = millis();
    steering_ratio = (int) (-cmd_msg.angular.z*(right/2)+IAB);
    if (steering_ratio<leftLimit) {
      steering_ratio=leftLimit;
    }
    else if (steering_ratio>rightLimit) {
      steering_ratio=rightLimit;
    }

    if (current<steering_ratio) {
      rightTurn();
    } else if (current>steering_ratio) {
      leftTurn();
    } else {
      digitalWrite(InputA, LOW);
      digitalWrite(InputB, LOW);
    }
  }
}
```

Figure 10. Arduino method that controls the commands sent to the motor

Emergency System

Emergency stop buttons (EStops) were added to the golf cart to ensure the safety of the rider and pedestrians around the cart, as seen in Figure 11. These buttons are normally closed switches, so the circuit is complete when the button is not pressed. The buttons are latching so that when the button is pressed, it stays pressed until it is specifically undone. The EStops are wired in series so that if any of the buttons are pressed, then the circuit is broken. The Estop circuit is connected to the Arduino and powered from a 5 volt pin, grounded with a pull down resistor, and connected to a digital pin as an input. The Arduino is checking for a digital HIGH on every loop and then proceeding with normal operations. However, if a digital LOW is read then the loop switches to an emergency mode. The emergency mode is then locked in and can only be exited after the EStops and Arduino have been reset. Once in emergency mode, the brake is fully applied and acceleration is turned to zero, and this repeats until the system is reset. Another emergency feature was added to the Arduino which utilizes the emergency mode which monitors the connection to ROS. When the Arduino is not connected to ROS, the Arduino will enter emergency mode but can exit to normal operation mode when connected to ROS. Both of these emergency systems are put in place to ensure abnormal reactions are mitigated.

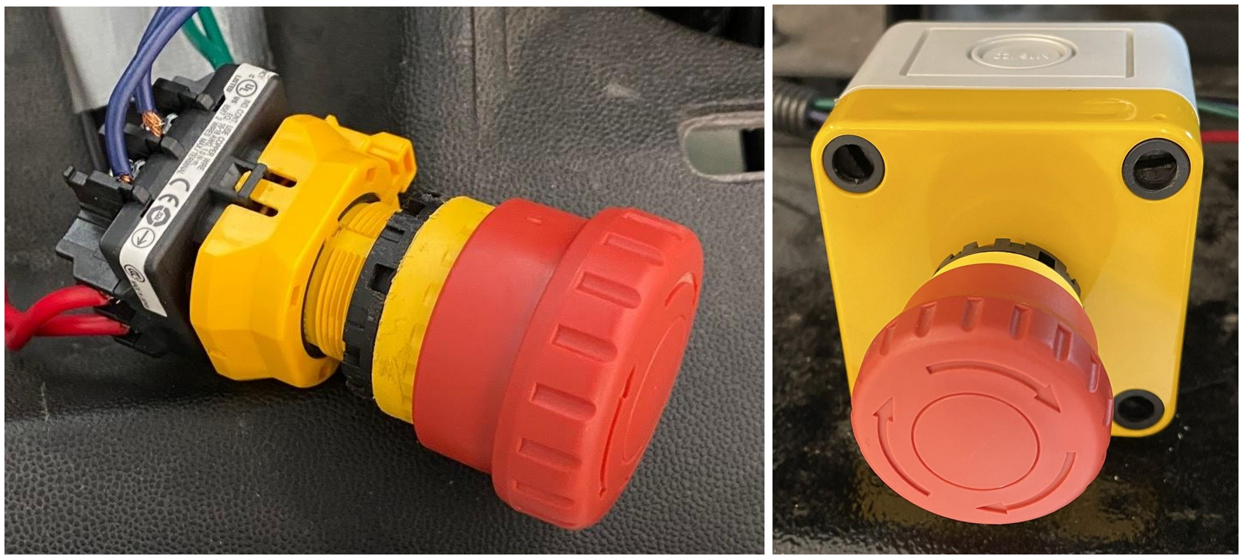


Figure 11. Emergency stop buttons present around the cart to safely shut down the cart when pressed

Software System

In order to control the mechatronic systems and begin introducing autonomy into the system, a solid software system had to be set up. Many of the desired software features needed for autonomy required significant processing power and there is limited space on the golf cart, so a custom computer was built to meet these requirements. The computer was the main source of computing, with it sending messages to the Arduinos for the mechatronic system control. Most of the software is for visual based tasks, such as processing sensor data. Ubuntu 18.04, Bionic Beaver, was installed on the computer as the operating system. Many software packages were installed to accomplish specific tasks, so when combined the system functions well as a whole.

The main software install was Robot Operating System (ROS), a middleware used by the robotics community to aid in the development of robotics projects. ROS provides access to countless software packages that were designed specifically for robotic applications. All of the data is transferred through topics as messages by ROS. This makes it easy to pass data from one package to the next since the messaging system is consistent. Nodes are executable programs that can perform endless tasks, and subscribe and publish topics. Topics are the path by which messages are sent and it is similar to radio frequencies since a node can only get messages from the topic to which it is subscribed. Messages are sent along topics and represent the data that is being sent from one node to the next. By utilizing ROS, it eases the development of robotic systems by providing an easy framework to integrate.

The ZED 2 stereo camera was the main sensor used since it provided an RGB image and a depth image. Fused visual and inertial odometry was also published by the ZED 2, making it the optimal choice for outdoor use. This required the installation of the ZED SDK, which provides the means for the raw sensor data to be converted into more useful data. After the installation, all of the data was available for use including point clouds and odometry. The ZED SDK, required the CUDA platform, by Nvidia, which allows the graphics processing unit (GPU) to be used for general processing and computing. CUDA allows the ZED SDK to process the data faster by utilizing the GPU instead of the central processing unit (CPU). The ZED ROS wrapper had to be installed into the catkin workspace to allow the ZED 2 to be used with ROS. The wrapper converted the generic data into ROS messages and published them on specific topics for other nodes to use.

Navigation requires an accurate and precise map so that the golf cart can avoid obstacles and travel towards its destination. Real-Time Appearance-Based Mapping (RTAB-Map) was installed from source as described on the rtabmap_ros github page. The following dependencies were installed to access the full capabilities of RTAB-Map: OpenCV, g2o, GTSAM, and libpointmatcher (for LiDAR use). RTAB-Map standalone was installed from source into the Home directory and RTAB-Map ROS was installed into the catkin workspace. RTABMAP ROS was built using:

```
$ catkin_make -DRTABMAP_SYNC_MULTI_RGBD=ON -DRTABMAP_SYNC_USER_DATA=ON
```

Building with these two settings allows multiple RGB depth (RGBD) cameras to be used at once and user data synchronized topics. Both the ZED 2 and

Many other ROS packages were installed as dependencies for the main packages but some smaller packages were installed that were used regularly. ROS has the ability to communicate and send messages to the Arduinos by using the rosserial_python package. Running the serial_node.py sets up a serial communication between ROS and the Arduino allowing all topics to be exchanged. Since the golf cart is being controlled by a joystick, the ds4drv package was required to use a wireless Playstation 4 DualShock Controller. Then teleop_twist_joy was used to convert the joystick messages into geometry_msgs Twist which is easier for the Arduinos to process. The ouster_example package was installed to interface with the Ouster LiDAR through ROS. The ouster package comes with a bunch of nodes and parameters that allow for specific tuning for the LiDAR based on the environment and goals. Most of the other package are install because they are dependencies for the main packages but there are some packages not need that are needed for operation

5. Validation

Brake Testing

The first actuator to be tested was the braking motor. To do this, a starting line was marked and the cart was moved uphill 58.5 feet from the line. The gas pedal was pressed to the maximum acceleration and released 19.5 feet from the line. The cart coasted for those 19.5 feet, then the brake was applied. The braking motor being used has 3 parameters, speed, power, and the amount of time for which they are applied. Data was collected for different speeds, powers, and times in order to determine the minimum stopping distance the cart could achieve. These were compared to trials where a driver pressed the brake pedal himself to ensure that there would not be a major difference between the two, which there was not. It was found that increasing the speed and power of the motor decreased the resulting stopping distance, so those have been adjusted to be close to their maximum. Applying the brakes for a longer period of time reduced the stopping distance as well, so the code was adjusted to maximize the amount of time they are applied. At the end of the trials, a stopping distance of 12’ 3” was found when the cart was moving at its operating speed. This is considered a success since the ZED camera that is being used in the front of the cart has a detection range of 20 meters and the manual brakes at operational speed had a stopping distance of 15’ 4”.

Run	Cart Speed	Motor Speed	Motor Power	Brake Type	Brake Distance	Time Brakes are Applied
1	Maximum	140	465	Motor	72’	Short
2	Maximum	N/A	N/A	Manual - Normal	35’ 2”	N/A
3	Maximum	N/A	N/A	Manual - Lock up	13’ 4”	N/A
4	Maximum	140	470	Motor	60’ 4”	Short
5	Maximum	220	475	Motor	43’ 9”	Long
6	Maximum	250	475	Motor	40’ 9”	Long
7	Operational	N/A	N/A	Manual - Normal	15’ 4”	N/A
8	Operational	250	475	Motor	12’ 3”	Long

Table 1: Testing braking distance with different settings on the braking motor.

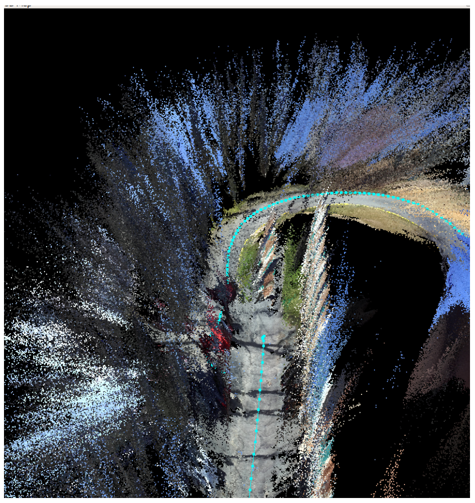
The DC motor braking was also tested. This braking occurs in two scenarios: the first happens when the digital potentiometer is set to 0, effectively setting the speed of the acceleration system to 0; the second happens when the car moves downhill. This downhill braking feature is integrated directly into the original Club Car system. The DC motor braking was tested in two locations: just outside the garage door on the ground floor of OMERF and on the downhill portion of the road up to the High Energy Physics Laboratory. In both situations, the motor braking was initialized by using the joystick to set the digital potentiometer to 0. At OMERF, a starting line was marked and the cart was moved uphill 58.5 feet from the line. After three trials, the average stopping distance was 110 feet. At High Energy Physics, another starting line was marked but the cart was moved uphill to two different locations; one at 20 feet and another at 40 feet uphill. At 20 feet, the average stopping distance of three trials was 102 feet while at 40 feet, the average stopping distance was 196 feet. Despite the OMERF trials leading with a longer windup distance, the High Energy Physics trials led to a longer stopping distance. This is likely due to the greater elevation decrease at High Energy Physics.

Acceleration Test

Acceleration is controlled digitally via a joystick. In order to be sure that the acceleration system and circuit would not fault, many loops of the proposed pathway were driven using the joystick. The system was validated in the fact that it could move the cart where it needed to go. Measurements were taken in order to find the average speed of the cart while using digital acceleration via cell phones in the cart. On flat ground, the cart averaged 13 miles per hour. Moving uphill, the cart averaged 6 miles per hour. Lastly, moving downhill was extremely variable. This variability was due to the motor braking itself. Whenever acceleration is not on, the cart's motor is braking slowly. Thus, it was difficult to measure the average speed when it was moving downhill. The proposed OMERF to High Energy Physics Laboratory route takes the golf cart one minute and 30 seconds to complete without stopping, and about 2 minutes when it does stop. Given the cart's long battery life, it will easily be able to complete this loop enough times to be useful.

Mapping Test

In order to test the mapping capacity of the cart, it was manually driven while the ZED 2 stereo camera was running. The data was shown in real time on the cart's monitor in order to know what the camera is mapping on each route. As the cart was driven, the camera produced both a point cloud and 2-D map of the route. Since the IMU in the ZED is not the best, it would say that the second loop of a route was slightly off from the first. Despite this, the camera data would recognize that it had seen that place before and the map data would "snap" over top of the data from the last loop. This way, a more accurate point cloud and map would be formed. Different team members drove the loop between OMERF and the High Energy Physics Laboratory many times at multiple different speeds so that the map data was optimized. This driving and data collection was an ongoing process throughout the semester. Despite the fact that an accurate map was created, more problems arose as the semester progressed. When the computer data was wiped to try to get the LiDAR to work, the map data was lost. This would not have been a big issue, but parameters were changed in Rtabmap and the ZED would no longer run on our PC. After two weeks of changing parameters, the ZED was finally up and running again. The mapping capability is the same as before, and the team learned how to detect people with it. Now, when a person walks in front of the camera, they are highlighted within a green box that passengers will be able to see. This should quell any nerves the passengers may have about autonomous driving.



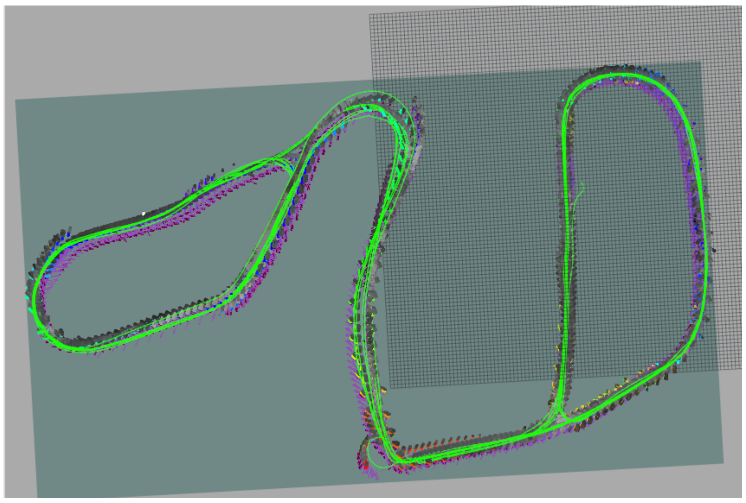


Figure 1: (Left) Point cloud data received from the ZED 2. (Right) Map formed from IMU data received from ZED 2, OMERF is inside the loop on the right side and the High Energy Physics building is inside the loop on the left. Both are displayed using Rviz.

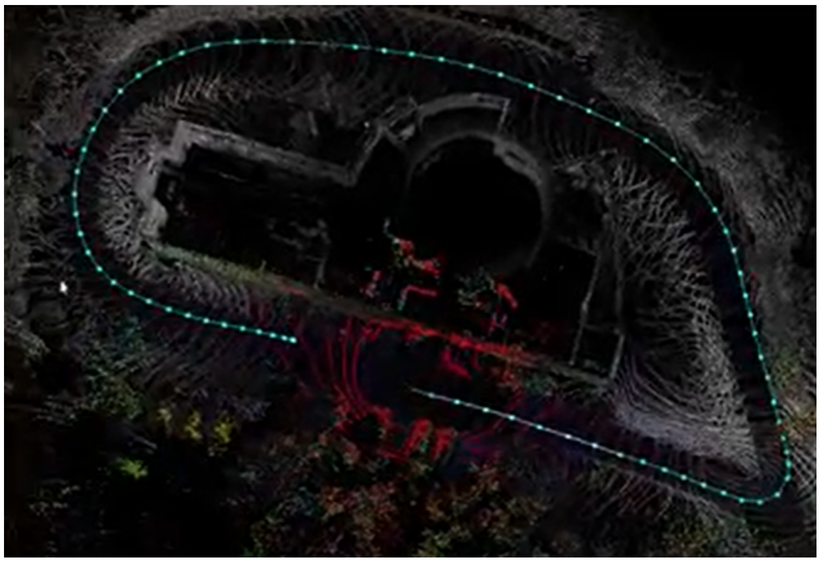
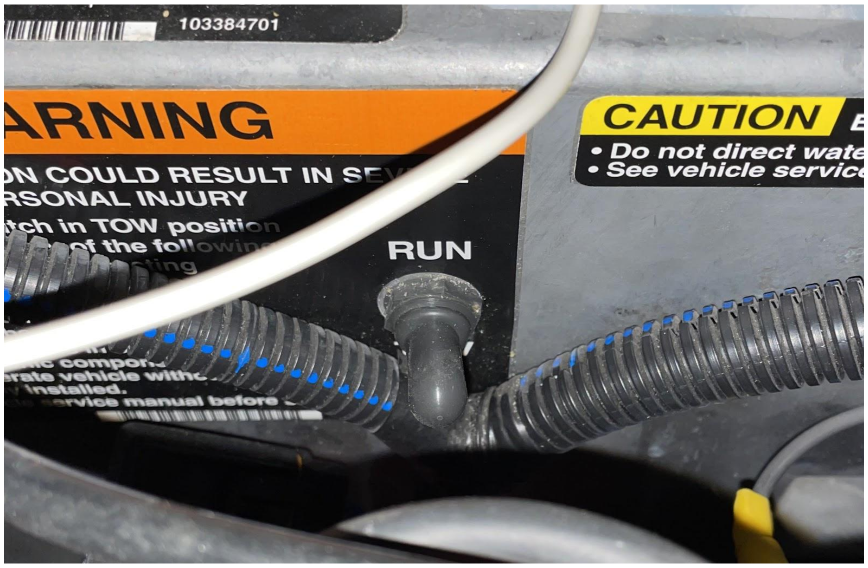


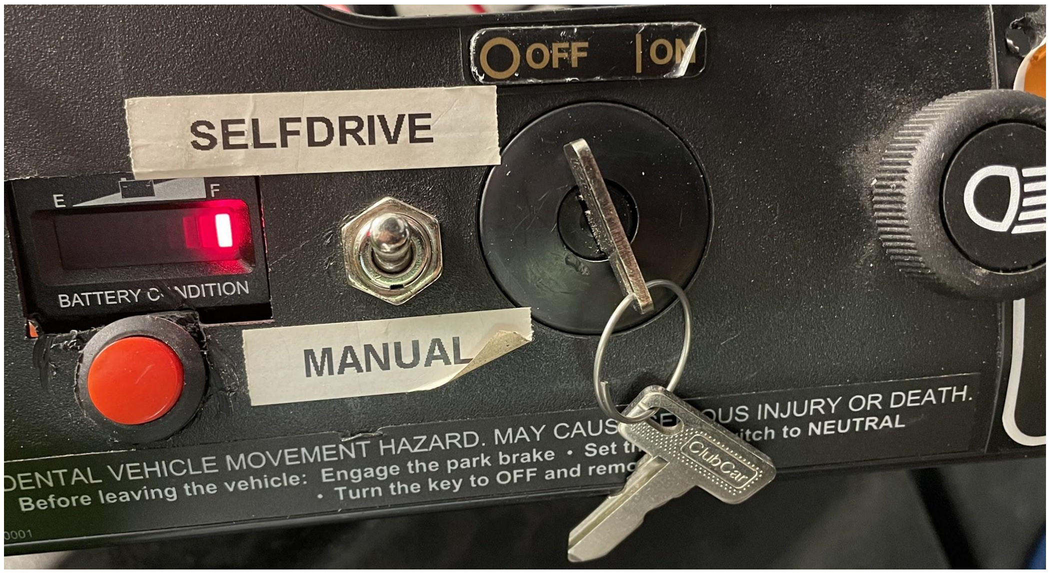
Figure 2: Map formed from LiDAR data displayed using Rviz.

6. Operations Manual

To allow the golf cart's electric motor to operate, the run-tow switch underneath the front seat of the cart must be switched upwards in the "RUN" direction.



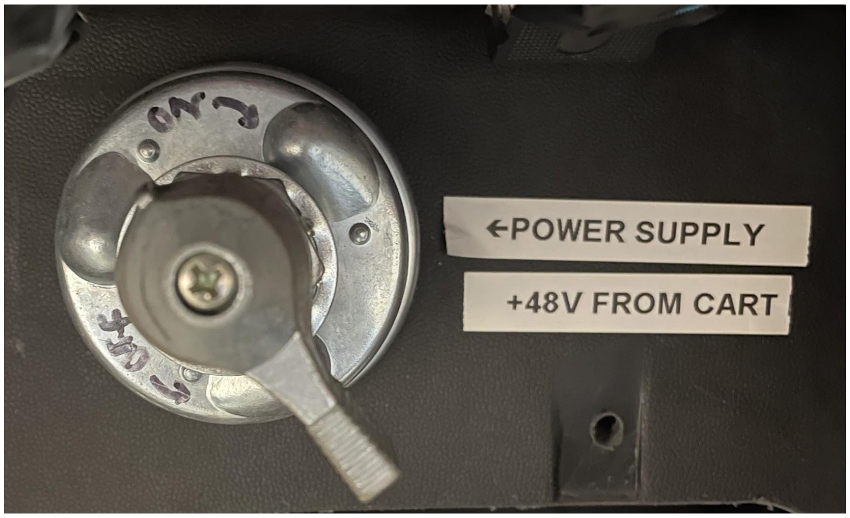
Next, to turn on the golf cart, the ignition key switch must be turned upward towards the "ON" position.



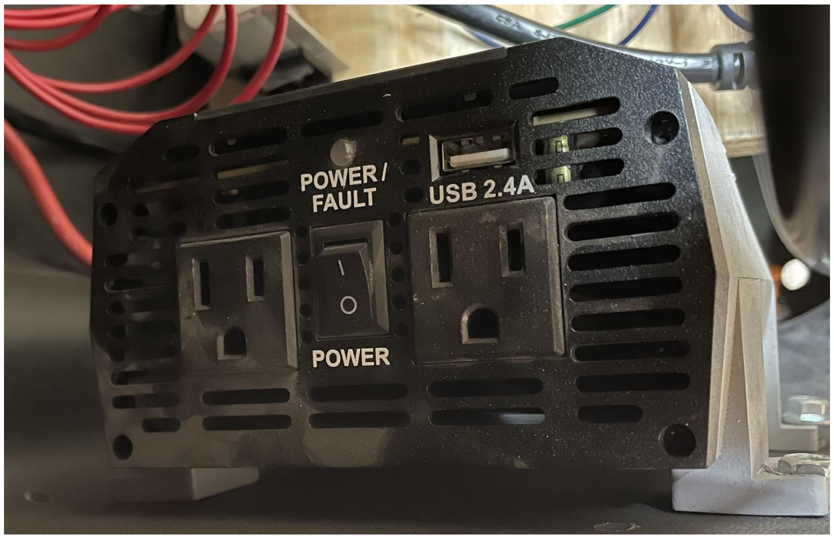
The golf cart drivetrain can be put into neutral, forward, or reverse using the switch shown below.

At this stage, the golf cart can be fully operated manually.

In order to continue with digital operation, the main 48-volt battery must be connected to the external electronics with the dial switch in the rear of the cart.



Also in the rear of the cart, on the left hand panel, is an inverter. After the inverter is switched on, the computer and monitor may be turned on.



The computer will boot up into Ubuntu, a Linux operating system, and no password should be required upon initial login. The username and password for the computer are ‘av788’ and ‘password’, respectively. Before continuing, make sure that all connections into the computer are secure.

In order to control the golf cart digitally, the Arduino’s need to be plugged into the computer, but only one at a time. Once the Arduino has started up, rosserial can be started with the appropriate port to set up communication with the Arduino. Any joystick can then be connected to the computer and using teleop_twist_joy (make sure the buttons are mapped in a known fashion), it can convert its messages into ROS topics for the Arduinos to subscribe to.

- To control the acceleration digitally, the signal wires from the motor controller have to be switched from the pedal to the Arduino, which can be accessed under the pedal, and the switch on the dash, in between the key and battery indicator, must be flipped to “SELF DRIVE”. Always flip the switch back to “MANUAL” when not in operation to ensure the cart does not wander since this overrides a safety feature for the golf cart acceleration system. The steering motor has a tendency to blow fuses, an easy way to tell is to listen to the motor when the power is turned on. When the power is first turned on, you should hear the motor move slightly and if not the fuse is likely blown.
- When the steering Arduino is plugged in, the steering motor should HOME and then return to approximately straight before connecting to ROS, if this does not happen restart the process, including turning the motor power off, to reset the motor.
- The braking motor should be functional with minimal issues and requires no extra steps. When the motor exceeds the travel limits set by the potentiometer, it has the possibility to blow a fuse so check the lights on the motor shield to ensure there is power. If the lights are off, check the fuse to see if it needs to be replaced.

After the joystick and Arduinos are connected the system should be operational, if the desired output is not achieved then try: ensuring all ESTOPS are not pressed, restarting the process, trying with another computer, checking to make sure all the wiring is tight, checking all fuses, and checking the button mapping on the joystick by using

```
$ rostopic echo cmd_vel
```

to view the output of the joystick (linear.x controls acceleration and braking, and angular.z controls steering).

Mapping can be achieved through either the Ouster LiDAR sensor or the ZED 2 stereo camera. To utilize the LiDAR system, a few preliminary steps must be taken. First, in a terminal window, the following command must be run while in the Desktop directory: sudo bash ouster_setup.txt. Next, the LiDAR sensor can be plugged into the computer via Ethernet. There are two Ethernet ports on the computer's I/O shield. Please use the black Ethernet port. Once the LiDAR sensor is connected to the PC, the following command must be run: sudo bash ouster_setup_2.txt. These two commands initialize the LiDAR. To map using RTAB-Map in conjunction with the LiDAR sensor, the following command must be run:

```
$ roslaunch rtabmap_ros test_ouster_gen2.launch
```

This will launch RTAB-Map with point cloud data from the LiDAR sensor. To utilize the ZED system, all that is needed is the ZED to be plugged into the computer via USB3.0. RTAB-Map can then be launched by one of two methods. First, a version of RTAB-Map built from binaries can be launched with the following command:

```
$ roslaunch zed_rtabmap_example zed_rtabmap.launch
```

This will bring up an RVIZ window that displays the path and image feed, among other things. Additional topics such as IMU data can also be added in RVIZ. The other option would be to launch RTAB-Map built from source using the following command:

```
$ rtabmap
```

This will bring up an RTAB-Map window. In the toolbar, under detection, the ZED SDK must be selected. When mapping, the RTAB-Map window will display loop closures.

7. Conclusion and Future Work

To summarize, the development of an autonomous campus vehicle yielded many positive results. First, all three major control systems (acceleration, braking and steering) were able to be controlled digitally with a controller, thus, with code adjustments, one could control these systems with solely a computer. Furthermore, two detection sensors, a stereo camera and a LiDAR sensor, were incorporated into the vehicle. These two sensors feed depth and image information to the main computer, which was displayed on the onboard display monitor. People could be detected in front of the cart via the ZED 2. In addition, these two sensors, when used in conjunction with RTAB-Map, created a map of the golf cart's surroundings. The mounting systems for the two sensors were also designed to be modular, allowing for seamless retrofitting to additional carts. Emergency stops were integrated into the mechatronic system such that all systems would disable upon being pressed. Lastly, the mechatronic and vision systems were validated by stopping distance, speed, and mapping tests. The team is confident that the groundwork is laid in order to achieve a fully autonomous system in the future.

In order to take the step to full autonomous operation, there are a few more major steps. Though the three main control systems were able to be manipulated digitally, the systems were not able to work in tandem with each other, and could not be initialized using one launch file. In terms of the vision system, future work would include fusing the Ouster and ZED data together. This would create an ideal map of the cart's environment. To accomplish this, a transform is required to know where one sensor is relative to the other. Once this is accomplished, localization of the golf cart in a pre-built map would be the next goal. Operation through waypoint navigation, possibly using an Ackermann steering package due to the Ackermann steering system the golf cart operates on, would be an ideal next step. Finally, full autonomy would be tackled.

8. References

Amberkar, S., Bolourchi, F., Demerly, J., & Millsap, S. (2004). A control system methodology for steer by wire systems. *SAE Technical Paper Series*. doi:10.4271/2004-01-1106

Club Car DS Installation Notes. (n.d.). Retrieved from https://www.hpevs.com/Site/images/jpeg/instructions/club-car-ds-installation-notes-6-21-12.pdf

Dikmen, M., & Burns, C. M. (2016, October 01). Autonomous Driving in the Real World: Experiences with Tesla Autopilot and Summon. Retrieved April 29, 2021, from https://dl.acm.org/doi/pdf/10.1145/3003715.3005465

Garrick, R. D. (2006). Sensitivity of CONTACT electronic throttle Control sensor to control System Variation. *SAE Technical Paper Series*. doi:10.4271/2006-01-0763

Ingle, S., Phute, M. (2016). Tesla Autopilot : Semi Autonomous Driving, an Uptick for Future Autonomy. *International Research Journal of Engineering and Technology, 3*(9), 369-372.

Jeyachandran, S. (2020, March 04). Waypoint - the official waymo BLOG: Introducing THE 5th-generation Waymo DRIVER: Informed by experience, designed for scale, engineered to tackle more environments. Retrieved May 06, 2021, from https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html

Lambert, F. (2016, October 21). A look at Tesla's new autopilot Hardware suite: 8 cameras, 1 radar, ultrasonics & new supercomputer. Retrieved April 29, 2021, from https://electrek.co/2016/10/20/tesla-new-autopilot-hardware-suite-camera-nvidia-tesla-vision/

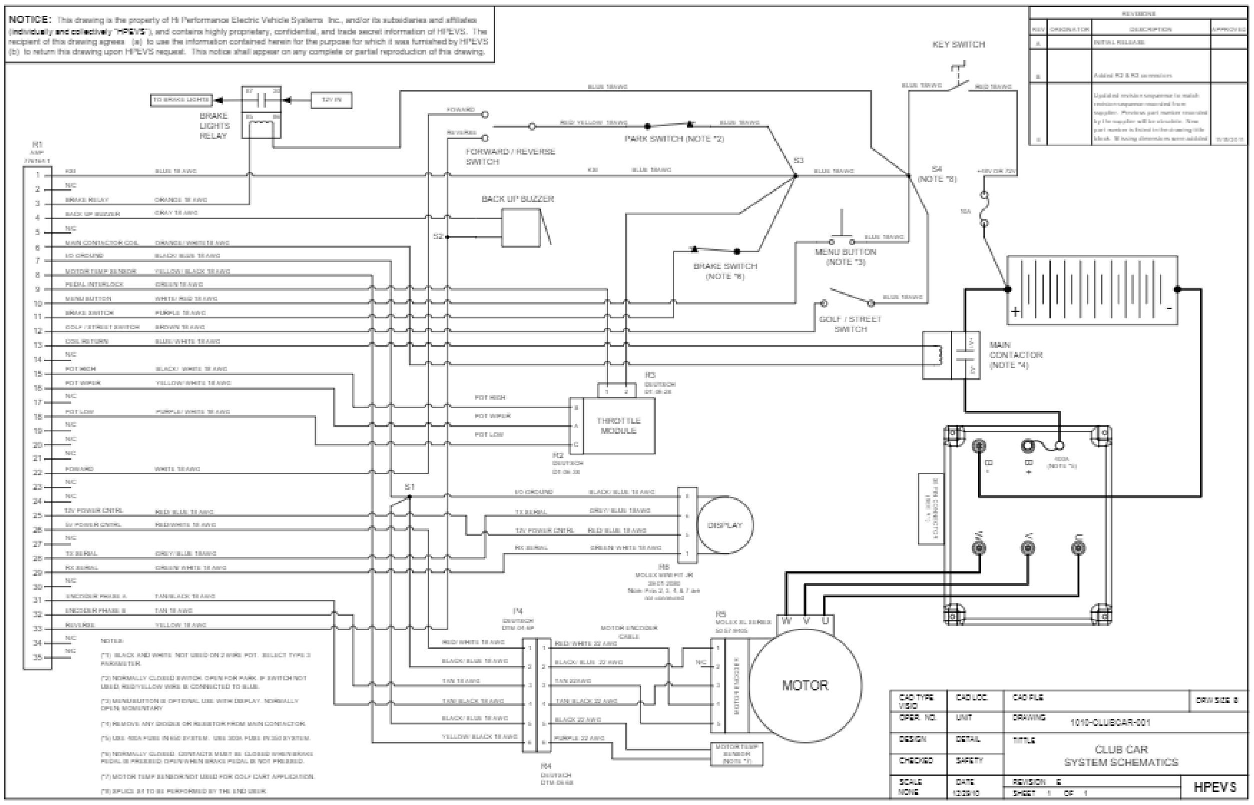
Tabora, V. (2020, September 15). LIDAR vs. camera - which is the best for self-driving cars? Retrieved May 04, 2021, from https://medium.com/0xmachina/lidar-vs-camera-which-is-the-best-for-self-driving-cars-9335b684f8d

Waymo - Company. (n.d.). Retrieved May 04, 2021, from https://waymo.com/company/

White, J. (2020, October 08). Waymo opens driverless Robo-taxi service to the public in Phoenix. Retrieved May 06, 2021, from https://www.reuters.com/article/us-waymo-autonomous-phoenix-idUSKBN26T2Y3

Xiang, W., Richardson, P. C., Zhao, C., & Mohammad, S. (2008). Automobile brake-by-wire control system design and analysis. *IEEE Transactions on Vehicular Technology, 57*(1), 138-145. doi:10.1109/tvt.2007.901895

9. Appendix



NOTICE: This drawing is the property of Hi Performance Electric Vehicle Systems Inc., and/or its subsidiaries and affiliates (individually and collectively "HPEVS"), and contains highly proprietary, confidential, and trade secret information of HPEVS. The recipient of this drawing agrees (a) to use the information contained herein for the purpose for which it was furnished by HPEVS (b) to return this drawing upon HPEVS request. This notice shall appear on any complete or partial reproduction of this drawing.

NOTES:

(*1) BLACK AND WHITE NOT USED ON 2 WIRE POT. SELECT TYPE 2 PARAMETER.

(*2) NORMALLY CLOSED SWITCH. OPEN FOR PARK. IF SWITCH NOT USED, RED/YELLOW WIRE IS CONNECTED TO BLUE.

(*3) MENU BUTTON IS OPTIONAL. USE WITH DISPLAY. NORMALLY OPEN, MOMENTARY

(*4) REMOVE ANY DIODES OR RESISTOR FROM MAIN CONTACTOR.

(*5) USE 400A FUSE IN 650 SYSTEM. USE 300A FUSE IN 350 SYSTEM.

(*6) NORMALLY CLOSED. CONTACTS MUST BE CLOSED WHEN BRAKE PEDAL IS PRESSED, OPEN WHEN BRAKE PEDAL IS NOT PRESSED.

(*7) MOTOR TEMP SENSOR NOT USED FOR GOLF CART APPLICATION.

(*8) SPLICE S4 TO BE PERFORMED BY THE END USER.

CAD TYPE VISIO	CAD LOC.	CAD FILE	DRW SIZE B
OPER. NO.	UNIT	DRAWING 1010-CLUBCAR-001	
DESIGN	DETAIL	TITLE CLUB CAR SYSTEM SCHEMATICS	
CHECKED	SAFETY		
SCALE NONE	DATE 12/29/10	REVISION E / SHEET 1 OF 1	HPEVS