Work on overall flow. Feel free to talk big picture and then go through design + implementation + test for each subsystem

Final Project Report
Description of the Project

Pacific's campus is just the right size to walk from one end to the other in about 15 minutes, and the same trip can be made quicker with a longboard. However, for students with injuries or disabilities this 15-minute trip is magnified, and it can make a trip from one class to another impossible if the time in-between is too small. Team Budget Tesla proposes a solution to this issue: an autonomous system that would enable the transfer of people around campus, specifically targeted towards those for whom travel is more difficult and time-consuming.

passive voice

To fill this niche, we plan on refurbishing a golf cart. Sensors and data processing would be added onboard to provide obstacle avoidance and self driving capabilities. Stripes, the current method to receive rides on campus, currently only runs during the night and must have a driver in order for students to be picked up and dropped off. During the day, there are no methods of getting around campus for students with reduced mobility. Additionally, there is cost and training required for Stripes drivers, which would be eliminated in an autonomous system. While there may be an initial cost of building the system, over the long term it would prove to be a less expensive and reliable option. Our client has also offered to provide a golf cart we will modify, and there is some other pre-existing inventory that can help make this form of transportation affordable. If we successfully implement our project, we will be able to provide a faster method of transportation for people to get around campus during the day. This will significantly improve the experience of impaired-mobility students on campus, potentially saving hours of travel time per student.

that

Project Requirements and Specifications

work on flow
needs like diagram
sentences

This transportation system is not limited to ~~just~~ doing tours for visitors or pickup for disabled students but should be a general form of transportation accessible by all. There should be a variability in the pickup and destination locations implemented by the system, but these locations can be limited to the general Stockton campus of University of the Pacific. All locations must be connected by a paved road that our vehicle can safely drive on. It should handle multiple location pickup and drop-off requests with a remote calling system and a queuing system, to allow all passengers to arrive as promptly and happily as possible. The accuracy of the cart will drop off and pick up at the paved road closest to the main entrance of the location, within 10 meters from the front entrance of all specified locations. Each named building should have at least one such point, with some (such as the University Center) having multiple for convenience. The cart should be able to aptly adapt to changes in campus geography. For example: removing or adding points, avoiding detours or closed roads, and other such occasions. Table 1 highlights waypoints planned for the first prototype.

↳ not a complete sentance

Location	Alternate pick up spot
CTC Brookside Entrance, park in front of the stairs.	CTC: Koury side entrance, park at the end of the yellow brick road? Too hard to navigate, in front of stairs leading up to Human resources
UC: Book store side, will park on UOP North Srv Rd in front of Lathy University Bookstore	Frat Row side: Will park on the Presidents Dr. adjacent to Alpha Phi house
Baun Fitness Center: Park on Baxter way closest to the only entrance	
Mail Room: Park on North Srv Rd in front of the Amazon Lockers	
Classroom Building Street Entrance: Park in front of stairs adjacent to N Kensington Way	Classroom Building front entrance: Park at the end of brick road facing sliding doors
William Knox Holt Memorial Library fountain side: Park in front of the water fountain	William Knox Holt Memorial Library Starbux side: Park on Dave Brubeck Way in front of Starbucks
Olson Hall: Park in parking lot in front of Olson Hall	

Table 1: Campus Pickup/Drop-off Points

The system should be able to handle all of the terrain around campus and be able to resist the standard weather and climate of Stockton. Both moving and stationary objects should be detected and safely avoided. The system should be able to run for at least 12 hours, but can be idle in between calls for transportation. Nighttime operation is not a requirement for this system. The batteries should be fully charged after a night of charging. Speed of the cart will be variable but it will have maximum limits of 15 mph and will follow all traffic signs and laws. The system should be able to prioritize the safety of its surroundings and have safety mechanisms implemented to prevent collisions with possible obstacles. This includes an emergency switch that can be activated by the passenger at any time. The system will also have user convenience mechanisms, such as a non-emergency stop switch.

some issue
connect like ideas togther or tell more in a user experiance story style?

Details of All Solutions Considered

Computer Vision

In terms of sensors, there are many different ways we could have solved our design problems. With our budget being our biggest constraint, we considered LiDAR sensors, RADAR sensors, and cameras for our vision.

LiDAR sensors come in at the top, being the most expensive sensors we can buy, but also give lots of good information with a high resolution. A LiDAR camera sends out a laser and measures the time it takes to bounce back to get an accurate reading of the distance in a point cloud. This point cloud can then be used to recreate an accurate 3-D Model of the area. This is great for detecting objects and making sure they are not too close to the vehicle. The downsides of LiDAR are ~~in~~ the cost and the sensors are without color. Due to high cost, we are only able to afford a LiDAR sensor with an angular resolution of 1°. This means we will only be able to detect things on a plane around our cart, which does not give us all the information we need to drive the vehicle.

RADAR sensors will not be as accurate as a LiDAR sensor, but are significantly more affordable and still provide distances from objects around but with a lower resolution. This means RADAR will not be able to detect smaller objects or fast moving ones. RADAR can detect objects in bad weather like snow and rain, ~~which~~ LiDAR does not work as well. RADAR emits electromagnetic waves and reads the rebounded waves. This also allows for measuring relative speed of all detected objects. A single short-range RADAR sensor with a wide resolution or multiple RADAR sensors would give our autonomous driving system plenty of information to use to avoid collisions.

Both sensors return distance to objects, but do not actually see. Only cameras can see color, but lack depth perception. Cameras are our cheapest option for computer vision, but also require the most computing power for image recognition. Cameras see the same way humans do, but without considerable calculations, we cannot determine distances or create models of our surrounding areas. Cameras also do not work well in bad weather or in low-light conditions.

Since there is a considerable amount of overlap between each type of computer vision, we should be able to pull enough information to create an autonomous driving system without the use of LiDAR cameras. Using cameras for color and image recognition and RADAR for object detection and emergency braking, our system should provide enough information to properly navigate campus safely. This would ease the amount of stress on our budget using cheaper sensors, and also give us a higher resolution to work with as opposed to the range we get with a LiDAR sensor with such a low angular resolution. This also covers our bases with weatherproofing our cart because RADAR can see in all weather conditions.

Braking

In terms of braking three options were considered. One was to create an H bridge using relays. The second option was to create an H bridge using mosfets. The last option was to simply buy an IC controller that had the integrated circuits inside.

Disadvantages of the relay circuit is that it is difficult to implement PWM for the motor speed control. The disadvantage of buying a gate driver is that the ones available were expensive due to the high current in our project. The final decision was to implement these circuits with mosfets.

The next design decision was whether to use GPIO output pins, a PWM signal, or a PID

controller to drive our mosfets. Final decision was to use PWM since we can implement motor speed and have a much smoother drive. A PID controller would require feedback to compare with for its adjustments. A GPIO output would be simpler but we would be at full duty cycle (always on) or off.

System Design Overview

put before details

The block diagram ~~below~~ seen in Figure 1, and the functionality table seen ~~below that~~ in Table 2 depict the Level 0 Functional Decomposition of our overall system. At the most broad viewpoint of our project, we will have a user send in pick-up location and a destination location, and with supply of power to our system, our golf cart should be able to provide transportation for that person while using sensors to take in data of ~~what~~ the golf cart's surroundings ~~are~~.

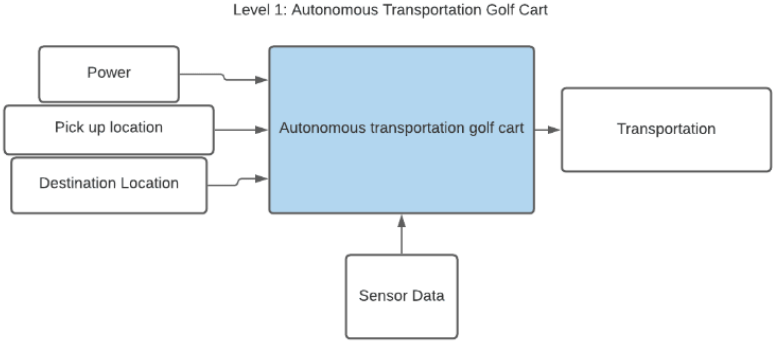


Figure 1: Level 0 Functional Decomposition (Overall System)

Overall System	Description
Inputs	Power: Rechargeable system for the 6V batteries Locations for pick up and drop off Real time sensor data
Outputs	Movement
Functionality	Pick up passenger and take them to a destination on campus Completely autonomous behavior

Table 2: Overall System Functionality

The block diagram below seen in Figure 2, and functionality table below it seen in Table 3, depict what the level 1 Functional Decomposition of our system is. It looks more into what elements are going to be required in our system to achieve the goals set in the Level 0 Functional Decomposition. Our project will be broken up into 3 different subsystems: a module for sensors, a module for navigation decisions and a module for controlling the golf carts motor functions.

write out numbers less than 10

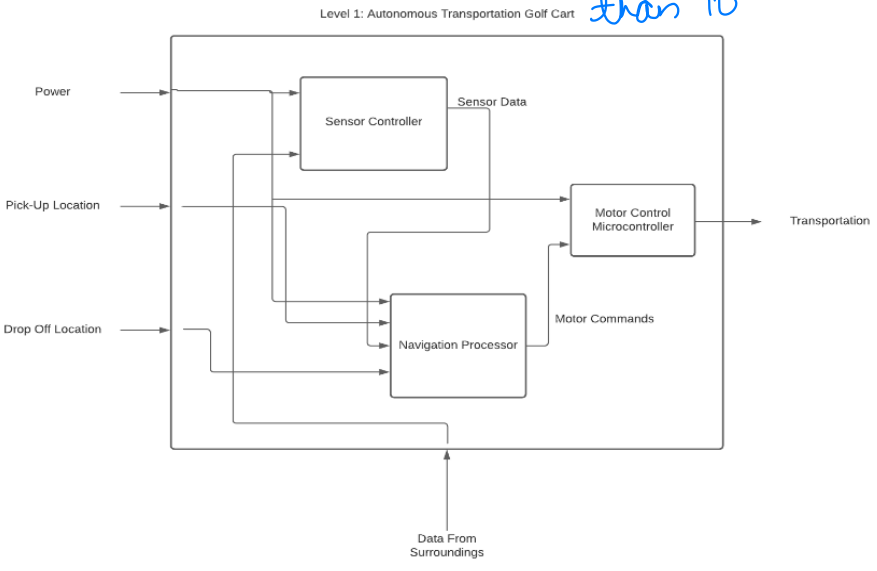


Figure 2: Level 1 Functional Decomposition

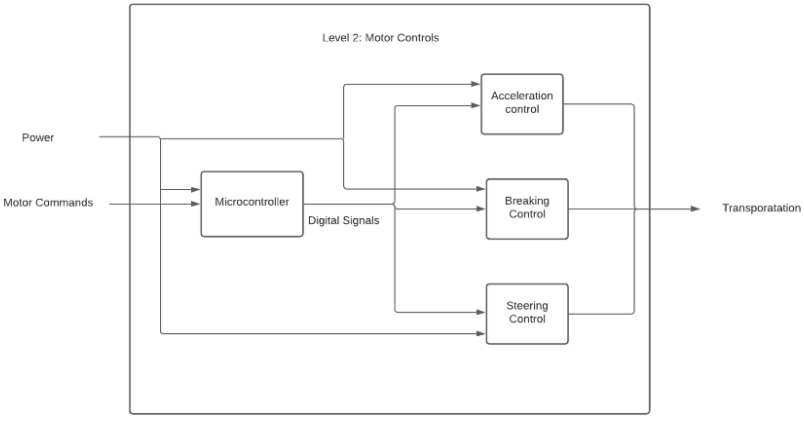


Figure 3: Level 2 Functional Decomposition - Subsystem 1 Motor Controls

make sure to discuss

Motor Controls	Description
Inputs	Power: supplied to microcontroller, acceleration control, breaking control, and steering control Motor commands: input from navigation processor
Outputs	Digital signals: from microcontroller that processes the motor controls Movement of vehicle motor functions
Functionality	Control the motor functions of the golf cart using motor commands from the navigation processor

Table 3: Motor Controls Functionality

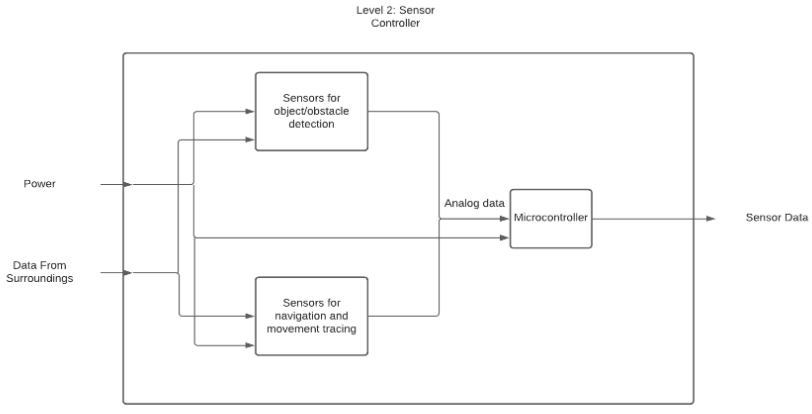


Figure 4: Level 2 Functional Decomposition - Subsystem 2 Sensor Controller

Sensor Controller	Description
Inputs	Power: Supplied to microcontroller and all the sensors Data from surroundings: Collected by sensors
Outputs	Analog data from the sensors sent to the microcontroller Microcontroller takes analog data processes it and then outputs sensor data to be used by navigation processor
Functionality	Take in real time data of gold cart surroundings and feed that data to be used by navigation processor

Table 4: Sensor Controller Functionality

The block diagram ~~below~~ seen in Figure 5, and the functionality table ~~below that~~ seen as Table 5, depict the navigation subsystem. This Level 2 Functional Decomposition shows ~~what~~ the systems ~~will be~~ inside of our navigation subsystem ~~in order to be~~ that provide the logic and decision making behind our golf cart's movements. The first module in this subsystem will be a data processing unit that can take in the raw sensor data and drop off location, and then apply the necessary algorithms to make sense of what the data means. Once that information is processed and the current situation is understood, there will be another module that decides what kind of motor controls are required to achieve safe and accurate transportation of our passengers. The latter module should be able to let the winch know to apply the brakes if there is an obstacle in the golf cart's path, or give steering and throttle instructions to get the golf cart to its desired destination.

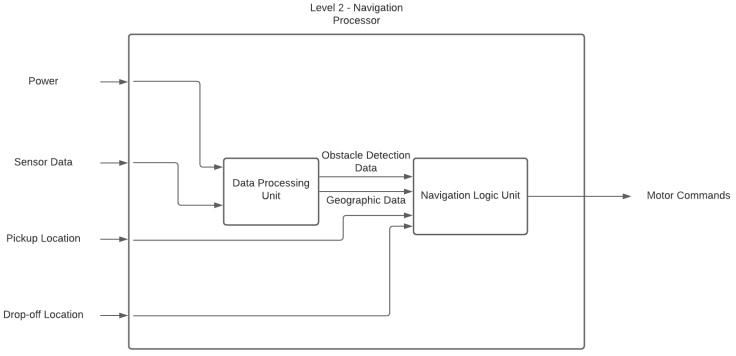


Figure 5: Level 2 Functional Decomposition - Subsystem 3 Navigation Processor

Navigation Processor	Description
Inputs	Power: Takes in power to run any processor/computer/microcontroller in the module Pick-up Location: Takes in user input of pick-up Drop-off Location: Takes in user input of drop-off Sensor Data: Takes in digital sensor data from the sensor microcontroller to be processed
Outputs	Motor Commands: Sends out commands to motor control such as stop, turn left/right, accelerate forward
Functionality	Takes in sensor data and drop off location as inputs, to then compute what motor controls are necessary in order to achieve transportation to the desired destination in a safe and efficient manner.

Table 5: Navigation Processor Functionality

*Functionality of all level 2 subsystems are discussed further in the following section and will need to be implemented in ECPE 196.

connect to pros sections

Design Details

Subsystem 1: Motor Controller

The motor controller will take in commands from a microcontroller which will power the braking, acceleration, and steering. Sensors will provide information to the microcontroller regarding when to power these subsystems. Since different speeds are required at different locations of campus, we will be using a digital potentiometer to alter our resistance and in turn the speed of the golf cart. The braking is operated with a winch which will have different braking speeds based on how near an object is.

Once the sensor recognizes an object, the information will be sent to the microcontroller. Right now the braking of the system is done with two transistors for switching. In the previous design the output buffer transistor blew as it couldn't handle the load. To fix this problem we could substitute for a higher load transistor or a relay. The relay would have a slower response time which we do not want. To solve this issue I researched freewheeling diodes, and optical couplers. A freewheeling diode reduces the harmonics and it also reduces sparking and arching across the mechanical switch so that it reduces the voltage spike seen in the motor. An optical coupler transfers electrical signals by using light waves to provide isolation between circuits systems. This will help to protect our microcontroller from our Mosfet by isolating the two. The microcontroller will send pwm signals based on the sensor data processed. These signals will drive our mosfets high with the desired duty cycle for the speed wanted. Since our motor operates at 12 volts with a stall current of 100 amps we will be using a SQJ123ELP_T1_GE3 PMOS. THis has a gate to source turn on voltage of 1.5v which will be good since our pwm operates at 5 volts. It also has a drain to source voltage of 12 volts and a continuous drain current of 137A. We needed at least 25 A to ensure stall current wouldn't be a problem.

The steering controller utilizes an SLA7076 driver in conjunction with a 34KM-K221 stepper motor in order to turn the wheels. Seems like no work needs to be done for the steering except for additional features if wanted.

In electric vehicles, potentiometers are attached to the gas pedal. The signal based on how much throttle is applied is sent to the controller. Electric vehicles have 2 potentiometers that deliver the same signal to the controller for safety. The controller reads both signals to ensure they are the same. If they're not the same the system applies no power. Since we have no driver we will have to emulate the throttle being applied by tapping into the potentiometers. We will have our microcontroller send out analog voltage. The first idea was to have this be connected to an op-amp to scale the voltage to match the potentiometer's voltage. This voltage will then be sent to the ECU of the golf cart. The problem is we would have to have an automated variable resistor to affect the gain. Instead we will be using a digital potentiometer which has a wiper resistance. This automatically changes the internal resistance which will change our output voltage sent to the ECU. Since there is a check to ensure both potentiometers are functional a resistor is needed in between two of the inputs of the ECU to bypass this check. Figure ## is a circuit to show this.

fix flow of ideas here. it talks about things multiple times.

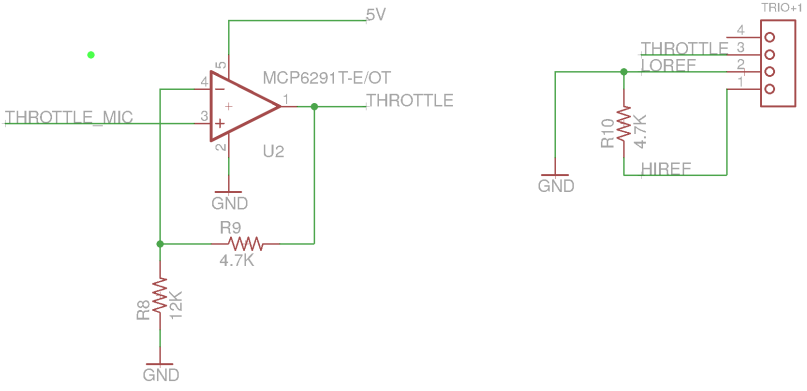


Figure _: Potentiometer Bypass

Subsystem 2: Sensor Controller

The sensor controller will take in raw data via multiple sensors attached to the golf cart. The data will be processed by a microcontroller and sent to the navigation to process to carry out instructions. The data that will be sent out will allow the navigation to determine location through recognition of the surrounding area. The microcontroller for the sensor controller system will power and configure all the sensors. There will not be any processing done in this subsystem 2 so we may find that microcontrollers can be replaced with something more elementary. The function is to configure the sensor then pass the data in a usable form to the navigation system where the data will be processed. The sensores will interpret the surroundings as analog data which will be sent into the microcontroller, and then digital data will be sent out of senor, controller subsystem 2 to the navigation controller.

Name	Voltage	Current	Hardware Interfaces	Price	Range	Distance Resolution	Angle	Dimension	Weight
Slamtec RPLIDAR A1 - 360 Laser Range Scanner	5V-9V	300-350mA	Rasberry pi, UART/ USB	$99.95	12-meter range	0.2cm distance resolution	360 degrees	Overall width: 98.5mm; Height: 60mm	170g

Table _: Lidar Sensor

reference & discuss

Name	Current	Voltage	Interfaces	Price	Dimensi	Field of	Frames	Resoluti

	Supply	Supply	Available		ons	View	per sec	on
Arducam CMOS MT9J003 1/2.3-Inch 10MP Color Camera Module	Still Mode at 7.5 fps: 338mW	3.3V	Two-Wire Interface (Can be used with I2C compatible devices)	$59.99	40 x 40mm	140 degrees	7.5fps at full resoluti on	10 M pixels

Table_: Camera

Figure _ shows a closed loop buck converter simulated in matlab, simulink. An open loop DC to DC buck converter was designed with the parameters to take in 16V and output 3.3V. At a switching Frequency of 25kHz. The allowed ripple in the capacitor and inductor is 20mV and 0.8V respectively. After the open loop converter successfully simulated, a controller was added. A controller is needed because the Voltage from the battery will vary as the cart accelerates and the battery power drains. The PI controller will keep the output voltage constant even with a varying input voltage. The next step is to simulate in SPICE when the school's license returns.

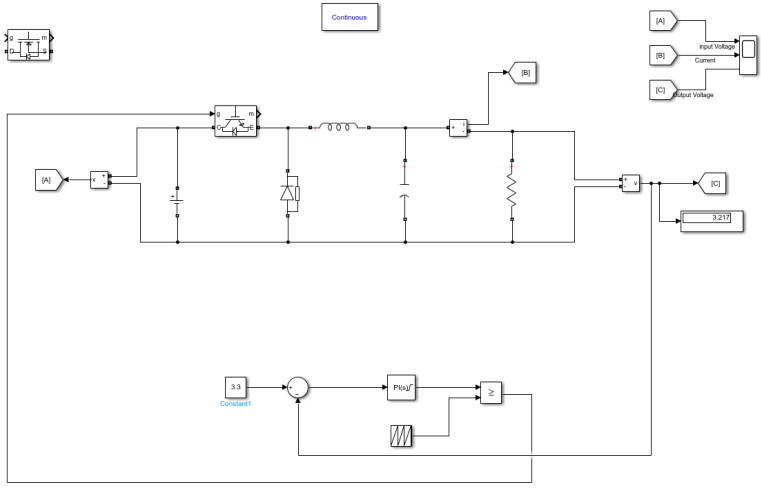


Figure _: Closed Loop Buck Converter

Subsystem 3: Navigation Processor

The navigation processor module, seen above in Figure 5, will consist of two main subsystems to generate the desired outputs from our planned inputs. While taking in power, sensor data, pickup location, and drop-off location, the output goal of the module will be to send out motor commands equivalent to: apply specified amount of throttle, apply brake, and adjust steering a specified turning radius. This module will take the input of sensor data, processing it to find both approaching obstacles and find geographic data to locate the golf cart. This data will be placed into a logic unit that combines the inputs of pickup and output location to determine what sort of path is required to get to the destination point safely, keeping in mind the safety of the riders and of pedestrians first. The logic unit should also translate this navigational path and

directions, deciding what sort of motor commands will be required to steer the golf cart in the desired motion. These motor commands are sent to subsystem 1 to receive and execute.

After discussing various options for the navigation system the following hardware and software will be initially implemented to create this subsystem: microcontroller TM4C123GH6PMI7, mini computer NVIDIA Jetson Nano 2GB, image processing library TensorFlow/Keras in Python using Anaconda environment, (insert radar data processing software here).

In order to control the image, radar, and geographical sensors, we have decided to go with the 32-bit Texas Instrument TM4C microcontroller used in ECPE 172. We chose this option because it has a wide variety of interfaces on it to be able control a wide range of sensors, it is already in the inventory at school and so will not have the added cost or unreliability of having to buy new microcontrollers/development boards, and as the entire team has taken the ECPE 172 course, everyone is familiar with the system and should be able to help out if necessary. We will likely need at least two separate microcontrollers: one to control the image sensor and the other to control the radar, but this decision can change in the future if there are alternative needs in terms of power requirements or if there is no capacity to have multiple large microcontrollers in our system.

For the mini-computer that will provide all the computing power we need as well as run the overall program for navigation decisions, our design will be moving forward with the NVIDIA Jetson Nano 2GB. From speaking with previous senior project groups that have used the NVIDIA Jetson Nano, it has been proven in the past to be more than capable of image processing programs, and should be cost efficient for our own project. An alternative to this particular mini-computer would be to get a different version of the same computer, with more RAM or more computing power. We will be starting with the base version, and can look into upgrading if the need comes up in the future.

For the image processing algorithms used in this navigation module, it was decided to utilize transfer learning of pre-existing neural networks to bring in pre-trained models that have been proven effective at image classification. We will likely be using the tools from the TensorFlow and Keras Libraries, imported into an Anaconda Python environment as these libraries provide various different pre-trained models as well as functionality to facilitate transfer learning. The viability of this method is proven down below in the critical questions section, where we were able to port in the Xception Convolutional Neural Network and apply it to the image classification between cats and dogs. The exact details of this system are explained further below, but we were able to successfully bring in the NN, retrain it, and apply it to a specific problem.

insert radar data processing software here

External Subsystem: User Interface

To provide an effective service, the autonomous vehicle must be able to communicate with passengers. Our current vision for the User Interface is an interactive map of campus with the rendezvous points highlighted. Passengers should be able to choose a pickup and dropoff locations, as well as a pickup time and a dropoff location. They'll be given a time estimate before confirmation. This could ideally be implemented via a phone app or a web page. To control and track access, passengers would have to log in with a Pacific ID to use the app, and possibly also swipe their school ID card when they board the vehicle. In the case that a website or app proves obtuse or costly, a text line or onboard interface could be considered. This subsystem is of the lowest completion priority for the design process. If the cart itself proves to be one semester's worth of work, then this can be presented as a project that future CS students can undertake. In

could find one in senior project in Spring

the meantime, the cart would be controlled via a terminal, either directly or over ssh.

Table ##: User Interface

	Cost	Usability	Comments
Standalone App	~ 25 + 99/yr	High. Download the app, log in once, and use it whenever needed.	Cost is $0 if school is already part of the Apple & Android Developer programs. Can do both pickup and drop-off scheduling.
InsidePacific Page	0	Medium. Logging into insidepacific takes time and effort, website would have to run well on both desktop and mobile.	A web page would likely be the easiest thing to develop.
Text Line	unknown	Low. students would have to know what number to text and how to select a pickup/dropoff point (i.e. text "BAUN1 UC2" to 84437).	Would need to ask about existing text services for an estimate on price, complexity, etc. Backup option.
Onboard Controls	Cost of hardware	Depends on UI usability. Ideally high. Display the rendezvous points and allow passengers to select where they need to go.	Only to select drop-off points. At minimum, a card-swipe might be good for access control and usage tracking. At most, a display with a method of selecting the drop-off point.

Other Impacts of Your Design

Potential Impacts on Health and Safety

Safety is the top priority for our senior project because the cart can pose a threat to students on campus. Working with a vehicle that can drive 15 mph and weighs more than 700 pounds puts a large responsibility on our team. Especially since we are implementing an autonomous driving system meaning we will not be in front of the wheel. Not only are pedestrians at risk, but other drivers on the road. ~~So~~ our golf cart must be able to follow traffic laws to ensure the safety of other vehicles on the road. The safety of pedestrians and passengers is the top priority and we weighted it the highest in our weight table. After all, the purpose of our project is to serve the students and faculty at the university. Our senior project will make UOP and more friendly campus handicapped students and faculty. The purpose of our project is to provide safe and efficient travel for students with disabilities. Our cart can also act as a safer

form of transportation around campus as opposed to walking alone.

Cultural and Environmental

A working autonomous vehicle could pave the way for a fleet of similar vehicles. The transportation ecosystem within UOP could be more efficient and safer. Students would spend less time walking to classes and use that extra time to be more productive. The potential for night time vehicles that could patrol the campus or transport students that feel unsafe to walk at night. A patrolling golf cart, even unmanned, would act as a deterrent to any thieves looking for bicycle wheels. Access to more autonomous vehicles would make living in residential buildings such as calaveras, town houses, and grace hall more appealing. The amount of skateboard users would decrease, in turn the number of skateboard accidents and thefts would decrease.

Costs

The immediate cost of the cart is outlined in the Dollar Budget section, summing to **<whatever number>**. This could be accomplished with **(what do we need here? Just one budget, maybe one plus a request for more funds, maybe multiple budgets: depends on final cost)**. However, running the vehicle on campus would come with future and recurring costs. A location for the cart to charge would have to be built. The charging process could be human-operated or automatic, depending on future designs. Building a charging station entails an initial flat cost and ongoing operating costs. The electricity bill would increase for charging the cart everyday. 2.5kWH with the gold cart running for about 8 hours a day. The cart would consume 20000W if it were driving the entire day. The school's insurance may increase with an autonomous vehicle designed by students driving around. The vehicle would require regular maintenance, both visual and functional. Cleaning the cart surfaces would probably be a daily job, either before deployment or after-hours. Mechanical maintenance would be performed similar to any other cart in Pacific's fleet. In addition, the cart would have to receive regular system checks, to make sure that no part of the automation is failing.

Critical Questions

An important question that needs to be addressed is will our sensors have enough data to properly and safely navigate our campus. This is a critical question because if our cart does not have enough data from the sensors to detect obstacles, there could be collisions and potential harm to pedestrians and passengers alike. In order to answer this question, we need to simulate our sensors in an environment similar to how it will be in real life. For this, we decided that Matlab was the most effective way to simulate and test our different sensors. Using a map of the UOP campus found on OpenStreetMap, I imported the data for roads to the driving scenario designer in Matlab.

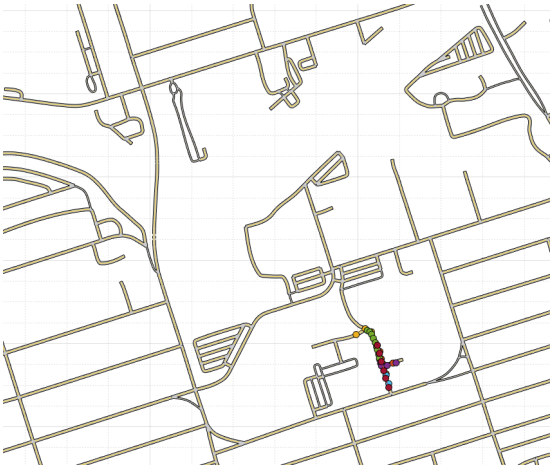


Figure _ : UOP Campus Simulation

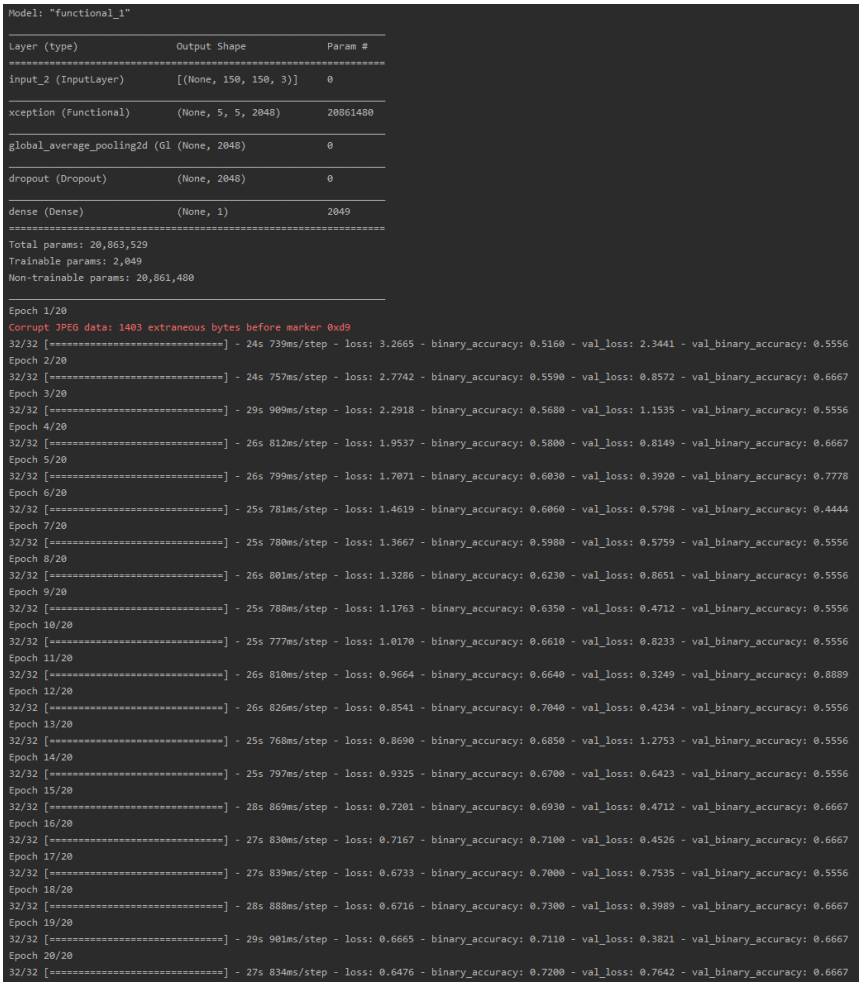
Using this map of campus, I added in things we would see around campus into our simulation. Our main actor car was mounted with 2 sensors, a camera and LiDAR sensor with an angular resolution of 1° for the first set of trials. Then, with a second set of trials, I used the same camera but with a radar sensor instead. (unfinished sorry basha) still time, thanks !

Another critical question we wanted to address was what sort of image processing algorithms would be able to develop in order to utilize potential cameras for the system, and how would they be able to help us in our navigational systems. After some thought discussion, it was decided that image processing would be used as a rudimentary form of obstacle avoidance, as well as assisting with traffic signs and geographical information. In order to achieve these sorts of functions, the image processing system will need to be able to detect common objects such as people, traffic signs, and buildings on campus. In trying to avoid building and training a neural network from scratch, transfer learning was used in a python conda environment to import the Xception convolutional neural network. The base neural network and the data set used for training and testing came from the Tensorflow and Keras libraries. To prove the viability of this method for future implementation into our project, I began with the basic problem of having the neural network classify between dogs and cats in images. I was able to successfully adapt and implement a transfer learning model into my own environment and was able to train the model to classify dogs versus cats with a 88% accuracy.

The validation data set of what the final classification test was evaluated against can be seen down below in Figure 3, as the cluster of 9 different cat and dog images. These were randomly selected from a larger data set provided by the TensorFlow_dataset library, as well as 1000 images for training and 18 images for a testing set. Then after importing in the Xception convolutional neural network, additional trainable layers were added on top and trained with the base model frozen to keep its integrity. The accuracy results after training the additional layers can be seen below in Figure 4, where 72% accuracy was achieved after 20 epochs of training. Then to improve the accuracy further, the base model layers were unfrozen and allowed to

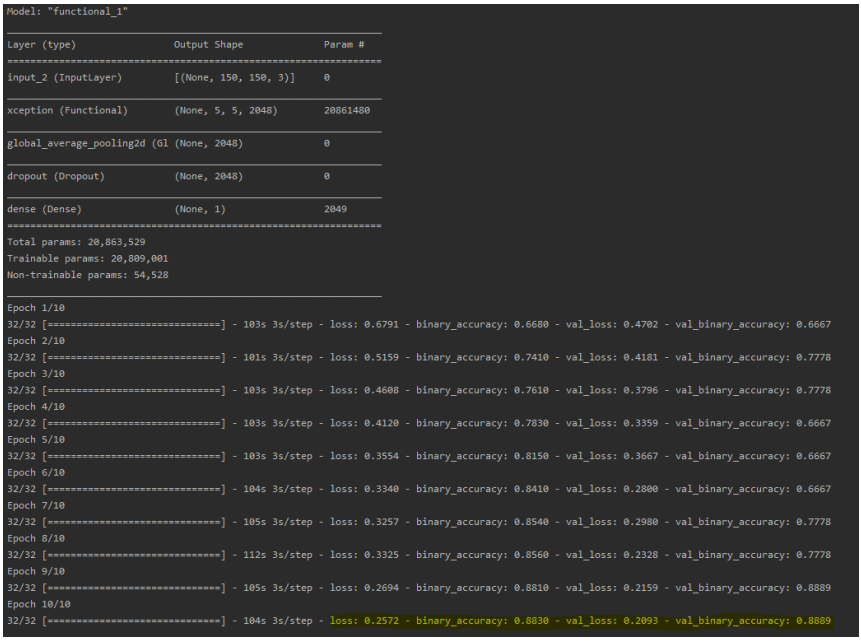
change weights at a very slow rate, trying to keep the integrity of the model while still fine-tuning it for the problem at hand. After 10 epochs of training, the transfer learning model was able to achieve a 88% accuracy of classifying the test data for the presence of cats versus dogs, seen down below in Figure 5.

Figure 3: Sample of Dogs vs Cats for Classification



```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
xception (Functional)        (None, 5, 5, 2048)        20861480
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 2049
=================================================================
Total params: 20,863,529
Trainable params: 2,049
Non-trainable params: 20,861,480
_________________________________________________________________
Epoch 1/20
Corrupt JPEG data: 1403 extraneous bytes before marker 0xd9
32/32 [==============================] - 24s 739ms/step - loss: 3.2665 - binary_accuracy: 0.5160 - val_loss: 2.3441 - val_binary_accuracy: 0.5556
Epoch 2/20
32/32 [==============================] - 24s 757ms/step - loss: 2.7742 - binary_accuracy: 0.5590 - val_loss: 0.8572 - val_binary_accuracy: 0.6667
Epoch 3/20
32/32 [==============================] - 29s 909ms/step - loss: 2.2918 - binary_accuracy: 0.5680 - val_loss: 1.1535 - val_binary_accuracy: 0.5556
Epoch 4/20
32/32 [==============================] - 26s 812ms/step - loss: 1.9537 - binary_accuracy: 0.5800 - val_loss: 0.8149 - val_binary_accuracy: 0.6667
Epoch 5/20
32/32 [==============================] - 26s 799ms/step - loss: 1.7071 - binary_accuracy: 0.6030 - val_loss: 0.3920 - val_binary_accuracy: 0.7778
Epoch 6/20
32/32 [==============================] - 25s 781ms/step - loss: 1.4619 - binary_accuracy: 0.6060 - val_loss: 0.5798 - val_binary_accuracy: 0.4444
Epoch 7/20
32/32 [==============================] - 25s 780ms/step - loss: 1.3667 - binary_accuracy: 0.5980 - val_loss: 0.5759 - val_binary_accuracy: 0.5556
Epoch 8/20
32/32 [==============================] - 26s 801ms/step - loss: 1.3286 - binary_accuracy: 0.6230 - val_loss: 0.8651 - val_binary_accuracy: 0.5556
Epoch 9/20
32/32 [==============================] - 25s 788ms/step - loss: 1.1763 - binary_accuracy: 0.6350 - val_loss: 0.4712 - val_binary_accuracy: 0.5556
Epoch 10/20
32/32 [==============================] - 25s 777ms/step - loss: 1.0170 - binary_accuracy: 0.6610 - val_loss: 0.8233 - val_binary_accuracy: 0.5556
Epoch 11/20
32/32 [==============================] - 26s 810ms/step - loss: 0.9664 - binary_accuracy: 0.6640 - val_loss: 0.3249 - val_binary_accuracy: 0.8889
Epoch 12/20
32/32 [==============================] - 26s 826ms/step - loss: 0.8541 - binary_accuracy: 0.7040 - val_loss: 0.4234 - val_binary_accuracy: 0.5556
Epoch 13/20
32/32 [==============================] - 25s 768ms/step - loss: 0.8690 - binary_accuracy: 0.6850 - val_loss: 1.2753 - val_binary_accuracy: 0.5556
Epoch 14/20
32/32 [==============================] - 25s 797ms/step - loss: 0.9325 - binary_accuracy: 0.6700 - val_loss: 0.6423 - val_binary_accuracy: 0.5556
Epoch 15/20
32/32 [==============================] - 28s 869ms/step - loss: 0.7201 - binary_accuracy: 0.6930 - val_loss: 0.4712 - val_binary_accuracy: 0.6667
Epoch 16/20
32/32 [==============================] - 27s 830ms/step - loss: 0.7167 - binary_accuracy: 0.7100 - val_loss: 0.4526 - val_binary_accuracy: 0.6667
Epoch 17/20
32/32 [==============================] - 27s 839ms/step - loss: 0.6733 - binary_accuracy: 0.7000 - val_loss: 0.7535 - val_binary_accuracy: 0.5556
Epoch 18/20
32/32 [==============================] - 28s 888ms/step - loss: 0.6716 - binary_accuracy: 0.7300 - val_loss: 0.3989 - val_binary_accuracy: 0.6667
Epoch 19/20
32/32 [==============================] - 29s 901ms/step - loss: 0.6665 - binary_accuracy: 0.7110 - val_loss: 0.3821 - val_binary_accuracy: 0.6667
Epoch 20/20
32/32 [==============================] - 27s 834ms/step - loss: 0.6476 - binary_accuracy: 0.7200 - val_loss: 0.7642 - val_binary_accuracy: 0.6667
```

Figure 4: Initial Training Results of CNN Added Layers



```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
xception (Functional)        (None, 5, 5, 2048)        20861480
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 2049
=================================================================
Total params: 20,863,529
Trainable params: 20,809,001
Non-trainable params: 54,528
_________________________________________________________________
Epoch 1/10
32/32 [==============================] - 103s 3s/step - loss: 0.6791 - binary_accuracy: 0.6680 - val_loss: 0.4702 - val_binary_accuracy: 0.6667
Epoch 2/10
32/32 [==============================] - 101s 3s/step - loss: 0.5159 - binary_accuracy: 0.7410 - val_loss: 0.4181 - val_binary_accuracy: 0.7778
Epoch 3/10
32/32 [==============================] - 103s 3s/step - loss: 0.4608 - binary_accuracy: 0.7610 - val_loss: 0.3796 - val_binary_accuracy: 0.7778
Epoch 4/10
32/32 [==============================] - 103s 3s/step - loss: 0.4120 - binary_accuracy: 0.7830 - val_loss: 0.3359 - val_binary_accuracy: 0.6667
Epoch 5/10
32/32 [==============================] - 103s 3s/step - loss: 0.3554 - binary_accuracy: 0.8150 - val_loss: 0.3667 - val_binary_accuracy: 0.6667
Epoch 6/10
32/32 [==============================] - 104s 3s/step - loss: 0.3340 - binary_accuracy: 0.8410 - val_loss: 0.2800 - val_binary_accuracy: 0.6667
Epoch 7/10
32/32 [==============================] - 105s 3s/step - loss: 0.3257 - binary_accuracy: 0.8540 - val_loss: 0.2980 - val_binary_accuracy: 0.7778
Epoch 8/10
32/32 [==============================] - 112s 3s/step - loss: 0.3325 - binary_accuracy: 0.8560 - val_loss: 0.2328 - val_binary_accuracy: 0.7778
Epoch 9/10
32/32 [==============================] - 105s 3s/step - loss: 0.2694 - binary_accuracy: 0.8810 - val_loss: 0.2159 - val_binary_accuracy: 0.8889
Epoch 10/10
32/32 [==============================] - 104s 3s/step - loss: 0.2572 - binary_accuracy: 0.8830 - val_loss: 0.2093 - val_binary_accuracy: 0.8889
```

Figure 5: Results After Fine-Tune Training of Full Model

Another critical question we had to answer is how we could ensure our circuits would not blow under these high voltage and high current systems. For our H-Bridge circuit we have to use preventative measures to help our circuit from not blowing up. This includes the use of a freewheeling diode which will prevent voltage spikes. Also, we must include a delay in between switching to ensure all transistors turn on and off at the same time. Lastly, we have to purchase Mosfets capable of handling this high current. To see what current levels we were dealing with I created a PSPICE simulation to see what the output load is. Below in figure ## is the model.

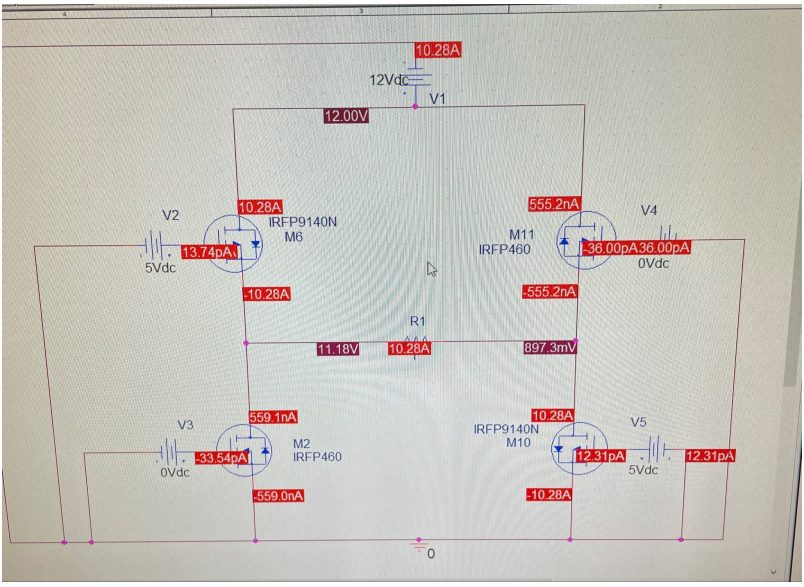


Figure #: H Bridge Circuit

As we can see we experience 10.28A on the motor when we have a 100% duty cycle. To ensure our transistors don't burn we will buy ones with a maximum continuous drain current of over 20 Amps. The SQJ123ELP_T1_GE3 PMOS has a continuous drain current of 137A which is well over what our load will experience.

Power Budget

Compute the total power consumption of your system using datasheets/measurement, and calculate expected battery life.

Dollar Budget

List cost of parts and components - including manufacturing and software costs as applicable - and compute total project cost.

Items	Cost
NVIDIA Jetson Nano 2GB	$59.00
SQJ123ELP_T1_GE3 PMOS (10)	$11.00

Total:	

Plan for Project Completion in ECPE 196

Team Member	Tasks to Complete	Due Date
	Find ECU?	12/9
All	Purchase All Parts	
All	Start of Next Semester	1/18
	Implement Microcontrollers for Sensors	2/15
Davis	Implementation for Mini-Computer Setup	2/22
Davis	Image Processing Algorithms Completed	3/22
All	System Put Together with All Components	4/4
All	Full System Testing Complete	4/18
Oscar	Build Closed loop Buck Converter	3/22

References

Material Type	**Words Cited**
Website	[1] P. Pieruigi, "Autonomous Golf Cart Testbed," *CUICAR*
Website	[2] U. Voelzke, "Three Sensor Types Drive Autonomous Vehicles," EVNOIA
Website	[3] S. Khvoynitskaya, "3 types of autonomous vehicle sensors in self-driving cars," itransition
Website	[4] "The Ultimate Sensor Battle: Lidar vs Radar," Intellias Automotive
Manual	[5] "YAMAHA G22 A/E SERVICE MANUAL Pdf," Yamaha Motorsports
Website	[6] "4 geolocation technologies compared" sensolus