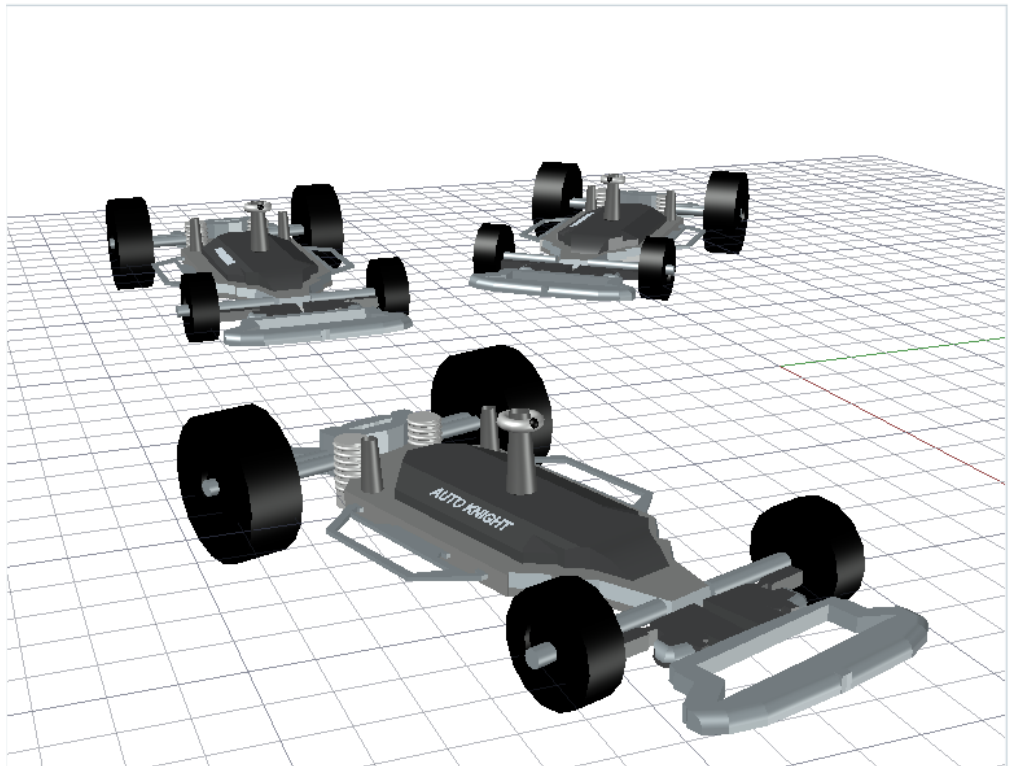




Senior Design I Fall 2017 Auto-Knight: The LIDAR Guided Autonomous Car

Eduardo Linares - EE
Christian Theriot - CE
Tyler Thompson - EE
Bruce Hardy - EE




Other Contributors

Dr. Yaser Fallah – Principal Investigator & Sponsor
Nitish Gupta – M.Sc. Candidate
Behrad Toghi – Ph.D. Research Fellow

Table of Contents

1.	Executive Summary.....	1
2.	Project Description	2
2.1.	Project Goals	2
2.2.	Project Motivation	3
2.2.1.	Market Analysis.....	3
2.2.1.1.	Ridesharing Projects and Partnerships.....	3
2.2.1.2.	Chip Manufacturers Projects/ Partnerships	4
2.3.	Improvements to Public Safety.....	4
2.4.	Requirement Specifications	5
2.4.1.	Communication Requirements.....	6
2.4.2.	Physical Requirements	6
2.4.3.	Sensor Requirements	7
2.4.4.	Power Requirements	7
2.4.5.	Camera Requirements.....	7
2.4.6.	Hardware Requirements	8
2.4.7.	PCB Requirements	8
2.4.8.	Software Requirements	9
2.5.	House of Quality	9
3.	Research Related to Project Definition.....	12
3.1.	Similar Projects	12
3.1.1.	MIT RACECAR	12
3.1.2.	Zheng Wang's OPEN CV RC Car.....	13
3.2.	Relevant Technology.....	13
3.2.1.	LiDAR Sensors	13
3.2.2.	Ultrasonic Sensors.....	15
3.2.3.	Infrared Sensors	16
3.2.4.	Inertial Measurement Unit (IMU)	17
3.2.5.	Differentials	17
3.2.6.	PID Controller	18
3.3.	Parts Selection	19
3.3.1.	RC Car Selection	20

3.3.2.	CPU Selection.....	23
3.3.3.	Microcontroller Selection.....	26
3.3.4.	Sensor Selection.....	32
3.3.5.	Stereo Image Sensor	35
3.3.6.	LiDAR Sensor	37
3.3.7.	Auxiliary Battery Pack and USB Hub	40
3.3.8.	Wireless Router and USB Network Adapter	41
3.3.9.	Summary	43
4.	Project Constraints and Standards.....	46
4.1.	Project Standards.....	47
4.1.1.	Power Supply Standards	47
4.1.2.	Wi-Fi DSRC Standard (802.11b).....	48
4.1.3.	Frequency Allocation	49
4.1.4.	Network Security.....	49
4.1.5.	Road Safety Standards.....	50
4.1.6.	IPC PCB Standards	51
4.1.7.	Inertial Measurement Unit (IMU) Standard	52
4.1.8.	AI and Self-Driving Car Standards	52
4.1.9.	Programming Standards	53
4.2.	Constraints.....	53
4.2.1.	Cost Constraints	54
4.2.2.	Environmental Constraints	54
4.2.3.	Social Constraints	54
4.2.4.	Political Constraints	54
4.2.5.	Ethical Constraints.....	55
4.2.6.	Health and Safety Constraints	55
4.2.7.	Manufacturability Constraints.....	56
4.2.8.	Testing Constraints	56
4.2.9.	Time Constraints.....	57
5.	Project Design Details	58
5.1.	Hardware Design.....	58
5.1.1.	Sensors and Calibration.....	61
5.1.2.	Infrared Sensor Calibration	64

5.2.	Software Design	67
5.2.1.	Localization Algorithms	69
5.2.2.	ROS	78
5.2.3.	Mapping	83
5.2.4.	Collision Detection and Avoidance.....	86
5.2.5.	Motion Tracking with Optical Flow	87
5.2.6.	Object Detection through Histogram of Oriented Gradients	95
5.3.	Vehicle Dynamics and Modeling.....	103
5.3.1.	Vehicle Disassembly	104
5.3.2.	Measuring Motor Speed.....	111
5.3.3.	Three-Dimensional Modeling – Initial Proposition	112
5.4.	PCB Details.....	128
5.4.1.	PCB Manufacturing.....	128
5.4.2.	PCB Design Software Selection.....	132
5.4.3.	PCB Design	137
6.	Administration.....	139
6.1.	Budgeting and Finance.....	141
6.2.	Milestones and Timeline.....	142
6.2.1.	Senior Design Timeline.....	142
6.2.2.	Sponsor Schedule.....	144
7.	Appendices.....	145
HYPERLINK "bookmark://_Toc500025891"		
146135		

List of Figures

Figure 1- House of Quality	10
Figure 2- MIT RACECAR from 2016 (Permission Pending on Photo).....	13
Figure 3- Generic LIDAR Sensor Block Diagram	15
Figure 4- Sound Spectrum	16
Figure 5- Ultrasonic Sensor Block Diagram	16
Figure 6- IMU Block Diagram	17
Figure 7- Type of Differentials	18
Figure 8- Exceed Sunfire Pro	20

Figure 9- Iron Track Shootout E8XBL	21
Figure 10- Traxxas Rally Racer.....	22
Figure 11- Traxxas Slash 4x4 Platinum	22
Figure 12- NVIDIA Performance Comparison	25
Figure 13- Purchased USB Hub, Router, Battery Pack, Stereo Camera, NVIDIA CPU, Traxxas RC Car and LiDAR.....	43
Figure 14- Purchased Arduino Uno, LCD for testing, Ultrasonic Sensor, IR Sensor, IMU and GPS Shield.....	43
Figure 15- PCB Standards	52
Figure 16- Hardware and Vehicle Interfacing.....	59
Figure 17- Wiring Diagram	61
Figure 18- Sensor 1 Test Results.....	64
Figure 19- Sensor 2 Results.....	64
Figure 20- JetsonTX2 J21 Header Pinout	66
Figure 21- Software Flow Diagram.....	68
Figure 22- Software Class Diagram	68
Figure 23- Updated Software Diagram.....	69
Figure 24- Steps during a Kalman filter's measurements.....	70
Figure 25- Example Gaussian Distributions	71
Figure 26- Gaussian Created by other Gaussians	72
Figure 27- 2D Gaussian Distribution	73
Figure 28- Before (Top) and After (Bottom) Filtering Without Motion	75
Figure 29- ROS Publish-Subscribe Framework	79
Figure 30- Diagram of Overall Software Architecture and Flow	81
Figure 31- ROS Message Definitions.....	83
Figure 32- Gazebo Simulated Environment	84
Figure 33- Gmapping Visualized in RVIZ	84
Figure 34- Octomap Visualized in RVIZ.....	86
Figure 35- Still Shot of Dense Optical Flow Output.....	91
Figure 36- Still Shot of Dense Optical Flow Output in Motion.....	91
Figure 37- Dense Optical Flow Program Flowchart.....	92
Figure 38- Still Shot of Dense Optical Flow Output Bounding Box.....	95
Figure 39- Flow Diagram of HOG Pedestrian Detection Program.....	99
Figure 40- Simple HOG Pedestrian Detection Test 1.....	100
Figure 41- Complex HOG Pedestrian Detection Test 1	101
Figure 42- Complex HOG Pedestrian Detection Test 2	102
Figure 43- Simple HOG Pedestrian Detection Test 2.....	103
Figure 44- Center Differential (left) and Clutch (right)	105
Figure 45- Two Springs in Parallel	105
Figure 46- Two Springs in Parallel on the Shock	106
Figure 47- Test Setup for PWM Measurement.....	107
Figure 48- Steering Angle Test Setup	108
Figure 49- Pulse from Function Generator (blue) vs. Pulse from the Arduino UNO (yellow).....	109

Figure 50- Duty Cycle vs. Steering Angle	109
Figure 51- Smooth Steering Logic.....	110
Figure 52- Smoothing Function	110
Figure 53- Hall Effect Sensor Testing and Coding	111
Figure 54- Installed Hall Effect Sensor.....	111
Figure 55: RPM counter flowchart.....	112
Figure 56- Component View with C.O.M.....	116
Figure 57- Profile View and Measurements (inches).....	117
Figure 58- Design Revision, Current Layout	119
Figure 59- AutoCAD Platform Design	122
Figure 60- LiDAR Mount Placement.....	123
Figure 61- LiDAR Mount.....	123
Figure 62- Stereo Camera Mount.....	124
Figure 63- Antenna Mount.....	125
Figure 64- AutoCAD Designed Body Mount.....	126
Figure 65- USB Hub Bracket Mount.....	127
Figure 66- Final fabricated mounts and platform installed to the car.....	127
Figure 67- PCB Circuit Schematic.....	138

List of Tables

Table 1: Communication Requirements	6
Table 2: Physical Requirements.....	6
Table 3: Sensor Requirements.....	7
Table 4: Power Requirements	7
Table 5: Camera Requirements	8
Table 6: Hardware Requirements	8
Table 7: PCB Requirements.....	9
Table 8: Software Requirements.....	9
Table 9: Effect of PID parameters on system output.....	19
Table 10: RC Car Specification Analysis.....	23
Table 11: NVIDIA Development Board Analysis	26
Table 12: Texas Instruments Development Board Analysis	Error!
Bookmark not defined.	
Table 13: Arduino Development Board Analysis.....	Error! Bookmark not defined.
Table 14: Arduino Sensor Specifications	35
Table 15: Stereo Image Sensor Analysis	37
Table 16: LiDAR Sensor Analysis	40
Table 17: Final Product Selection	45
Table 18: Circuit Power Specification.....	47
Table 19: Spectrum Allocations.....	49
Table 20: ISO/IEC Network Security Standards.....	50
Table 21: Pre-Calibration Data for Sensor 1	62

Table 22: Dijkstra’s Algorithm Steps	77
Table 23: List of Message Types	82
Table 24: HOG Test Results	100
Table 25: Test 2 Parameters.....	101
Table 26: Turning Parameters.....	109
Table 26: Important Values of Components	113
Table 27: Locations of Components in Figure 46.....	116
Table 28: Center of Mass Calculations	117
Table 29: Cost Analysis Table.....	142
Table 30: Senior Design Schedule.....	143
Table 31: Senior Design Schedule.....	144

List of Equations

Equation 1. Distance recorded by LiDAR Sensor.....	14
Equation 2. Ultrasonic Sensor Distance Calculation	16
Equation 3. Transfer function of PID controller.....	19
Equation 4. Gaussian Distribution	71
Equation 5. New Gaussian Distribution	72
Equation 6. Gaussian Movement Update.....	73
Equation 7. SLAM Location Equation.....	78
Equation 8. Revolutions per Second	112

1. Executive Summary

Every year thousands of people die in preventable car accidents, and millions are injured. The Auto-Knight team saw the opportunity to work with Dr. Yaser Fallah and the Networked Systems Laboratory at UCF to make strides in consumer safety by designing a small scale preliminary model of a self-driving vehicle that can communicate with other vehicles. This small scale model will work as a proof of concept to help raise money to create a full sized vehicle and also simultaneously learn the hardware and software necessary to create an autonomous vehicle.

The vehicle uses an RC car as a base, a LiDAR Sensor, a depth camera, and 4 ultrasonic sensors to gather data about its surroundings. The data is processed using an NVIDIA Jetson TX2 board, and that data is then used in multiple algorithms to localize the vehicle and create a map of its surroundings. If the sensors detect that an object is too close, the vehicle stops or turns to avoid said obstacles. By the end of Senior Design II, at least 4 of these vehicles will be created and will be able to route around one another efficiently. The implications of this achievement are not just increased consumer safety, but also more efficient routing of vehicles, leading to less time spent on the road for the average driver.

This report outlines the specific motivations for our team and the requirements laid out before us by our sponsor. This is immediately followed by a basic overview of some relevant technology and a review of all the considered hardware and the final choices for what hardware was used. After that, standards and constraints, such as political, economic, and ethical concerns, are discussed. After that, the calibration and testing of relevant hardware, a description of all the localization algorithms used, the OS used for the vehicle, modifications made to the vehicle, and circuit design for our PCB are described in detail with results for relevant tests. The final section contains our budget and our project timelines, which are separated based off our sponsors required deadlines for our project and the design courses relevant due dates for the paper, meetings with the faculty, and other relevant documentation.

2. Project Description

With input from our principal sponsor, Dr. Fallah, one of his MS.D students, Nitish Gupta, and one of his Ph.D research students, Behrad Toghi, we plan on creating two to five smaller scale vehicles that utilize wireless communication to navigate around one another safely.

These cars will then be used by Dr. Fallah in his laboratory to improve upon the technology. The cars will be small so they can be tested in Dr. Fallah's lab, low cost, send highly accurate data to one another in real time, and have communication ranges and maximum speeds relative to the final size of the product.

2.1. Project Goals

The goal of our project is to create a vehicle network that will model concepts of “deep learning” in machine intelligence; that is, to form a neural network between units in order to react cooperatively to each other's motion. By utilizing sensor input, the vehicles may project a data signal to all receiving units, simultaneously. Thus, each unit will predict the collective path(s) to avoid a variety of real world collision scenarios. Examples include collisions created by one car cutting off another, multiple cars reaching a 4 way stop and deciding the order they will move in, and a car parking itself.

LiDAR, Ultrasonic Sonar, and camera sensor inputs paired with probability algorithms are the operative means of achieving computer vision. A LiDAR scanner introduces laser range sensing capabilities to each unit, allowing a multi-planar representation of the vehicle's surroundings at each rotation of the LiDAR mount. By utilizing an infrared camera calibrated to LiDAR sensitivity, we can receive the signal reflected off of the surroundings and model the real-time position and orientation of nearby objects. Ultrasonic sensors may be utilized in addition to LiDAR for accurate short-range applications. The camera's purpose is to recognize pedestrians, stoplights, and other road signs so the vehicle can follow driving rules.

Methods of communication must be considered to form a viable

DSRC (Dedicated Short Range Communication) network. Outfitting the vehicles with modules capable of broadcasting a DSRC high-frequency signal 5.9 GHz optimizes the transmission of signals due to the “unlicensed” nature of this spectrum. In 1999, the Federal Communications Commission designated the 5.9 GHz band to be used exclusively by intelligent transportation systems; the standard of which is denoted within IEEE 802.11p.

2.2. Project Motivation

Our motivation behind our project has two aspects to it, personal gain and societal gain. The opportunity to help create a working prototype of an autonomous vehicle for Dr. Fallah’s research will provide us with the skills necessary to work on the development of a commercial model for a traditional automaker or one of their partners. Furthermore, we could end up pursuing research at a university for a graduate degree using our knowledge gained from this project. There is a very clear personal economic gain to this project. Our project also has the potential to help decrease the amount of deaths caused by reckless driving every year.

2.2.1. Market Analysis

According to a study released by Intel in June of 2017, the self-driving vehicle market has the potential to become an 800-billion-dollar market by 2035 and a 7 trillion dollar market by 2050[58]. Currently a large number of established companies in the auto-industry are engaged in a rat race to create a fully autonomous self-driving vehicle. Numerous partnerships have created between ride sharing companies, chip manufacturers, and auto-manufacturers to try to create strategic advantages over their competition.

2.2.1.1. Ridesharing Projects and Partnerships

The utilization of self-driving cars will provide a massive boon to the ridesharing industry by allowing companies to replace drivers with their own autonomous vehicles. The potential for this long term cost savings has led to companies like Uber, Lyft, and Mobileye to deploy self-driving car prototypes to Arizona for testing on their roads. Uber has also created a partnership with

Volvo and Daimler, and Lyft has generated multiple partnerships with Drive.AI, Ford, General Motors, Jaguar, nuTonomy, and Waymo to speed up the creation of a self-driving car fleet [58].

2.2.1.2. Chip Manufacturers Projects/ Partnerships

Recently, companies that traditionally only produced products for personal computers have entered the self-driving car race. In 2016, NVIDIA, who are known primarily for their popular GPU's for gaming, developed a car that could read the steering angles necessary to drive a car using the NVIDIA Drive™ PX 2 and a neural network based system called PilotNet. NVIDIA has also worked with Tesla to develop its auto-pilot features on the Model 3, Model S, and Model X vehicles. NVIDIA is also partnering with Toyota, Mercedes-Benz, Honda, BMW, and other manufacturers to provide the hardware for their self-driving vehicles. On October 10th of 2017, NVIDIA also announced a new chip called the NVIDIA Drive PX Pegasus, which is capable of performing 320 trillion operations per second using four processors. NVIDIA's main competition is coming from AMD and Intel, who have secured their own partnerships with Tesla and Waymo respectively [59].

2.3. Improvements to Public Safety

In the United States, over 37,000 people die in car accidents every year. 1,600 of these people are below the age of 15, 2.35 million people are injured, and the overall cost to the U.S is over 230 billion USD every year. Around the world, road crashes are the leading cause of death between the ages of 15-29. 1.3 million people are killed every year in car accidents, an average of 3,287 people every day, and 20 to 50 million are injured as a result. This makes road crashes responsible for 2.2% of all deaths, making it the ninth leading cause of death around the world. The overall cost is estimated to exceed 518 billion USD every year[1].

This grim description of the world's auto accidents shows a great need for increasingly automated cars capable of preventing crashes. According to an article published by the Atlantic, self-driving cars can prevent 90% of accident related fatalities by half by the middle of the century. This equates to about 30,000 lives saved using the number of 2013 traffic fatalities in America.

Globally, this equates to about 50 million lives saved in half a century. In comparison, modern vaccines are estimated to save 42,000 lives in America according to the Centers for Disease Control (CDC)[60]. This project is a step towards creating that reality.

In general, car manufacturers are adding more and more safety features that either give the driver more information, like blind spot detection, or take away control from the driver, like automatic braking. One feature that has yet to be implemented in commercial models is short range communication networking. To increase safety, cars would send each other data about their relative positions and velocities to ensure they all remain a safe distance from one another. In the event of an accident, the cars could communicate with one another and circumnavigate any debris efficiently. The effects of this feature have incredible potential to create a better and safer world.

Car related deaths would plummet due to the relayed data preventing accidents that could have occurred otherwise, less people would be injured or disabled in accidents, and people would spend less on repairing their vehicles. Less accidents would inevitably lead to lower insurance premiums for everyone as well, making vehicle ownership less of a strain on working families.

A short range communication network could also decrease travel times and allow passengers to spend more time at their destinations and less time getting there and back home.

2.4. Requirement Specifications

The requirements for our project, developed in conjunction with our sponsors and contributors, are detailed below. The requirements cover the minimum communication, physical, sensor, power, printed circuit board (PCB), and software performance metrics to build a functioning autonomous vehicle that meets Dr. Fallah's standards.

2.4.1. Communication Requirements

After careful and prolonged discussion with all our sponsors and contributors, we agreed to the requirements stated in Table 1 for our vehicle's communication standards. This is the most important standard to adhere to because this parameter is what separates other self-driving car projects from our own and has the greatest potential to change the standards for self-driving vehicles.

Table 1: Communication Requirements

Deliverable	Specifications
Vehicle to Vehicle Communication Range	The vehicle to vehicle communication range must be 30 feet in any direction.
Communication Frequency	Signals will be transmitted at a frequency of 5.9GHz

2.4.2. Physical Requirements

After exploring a variety of options for our vehicle's physical build, we agreed to the following requirements, with slight leeway given to the exact size of our base model due to the unknown sizes of various parts we are adding. A minimum speed of 15 miles an hour was decided on because anything slower would remove any concern for latency and not be an accurate model of a real vehicle. A brushless DC motor was a requirement due to the increased efficiency and reduced friction, which could potentially be a fire hazard. A summary of these decisions is shown in Table 2.

Table 2: Physical Requirements

Deliverable	Specifications
Chassis Size	Chassis will be as close 1/16 scale as possible. May be changed if necessary to fit more components.
Speed Requirements	The car must be able to move at least 15 miles per hour
Motor	The motor must be a brushless DC electric motor
Motion	The car must be capable of steering forward and turning left or right

2.4.3. Sensor Requirements

For our sensor requirements, we were given 3 primary requirements from our sponsor. We are only required to use two kinds of sensors, but were provided requirements for infrared sensors should we choose to use them. This is detailed in Table 3 below.

Table 3: Sensor Requirements

Deliverable	Specifications
LiDAR	LiDAR must have 360 degree vision and be able to detect anything within the minimum communication range of 10 meters
SONAR	Sonar sensors must be able to accurately detect anything within 30 centimeters of the sensor range.
Infrared (optional)	If any are used for collision avoidance, Infrared Sensors must be able to detect anything within 20 cm of the sensor range.

2.4.4. Power Requirements

For battery requirements, our sponsor wanted to be able to run experiments for at least 15 minutes. After some discussion, 20 minutes became the new minimum run time to account for adjusting or fine tuning experimental conditions. The number of batteries was also limited to 2, with one powering the vehicle and the other providing power to all sensor related functions. This is detailed in Table 4 below.

Table 4: Power Requirements

Deliverable	Specifications
Battery Life	Batteries must be capable of running the system for at least 20 minutes before requiring recharging
Number of Batteries	No more than 2 batteries can be used to power the system and motor of the vehicle. One battery must supply power to all modules of the system.

2.4.5. Camera Requirements

The requirements for our camera were agreed upon after discussing the role the camera would play in our project. It will be our second most important form of data collection because of it's use in pedestrian detection and road sign detection, so we

agreed to hold it to the same standards as our LiDAR sensor. This is detailed in Table 5 below.

Table 5: Camera Requirements

Deliverable	Specifications
Camera Range	The Camera will have a minimum range of 10 meters and will face the front of the vehicle

2.4.6. Hardware Requirements

For our hardware requirements we agreed that we create a vehicle with an onboard transceiver for wireless communications, that the vehicle would have the ability to be driven autonomously or with a controller, and that there would be at least 1 microcontroller or processor to process the data, and we have the option of adding more for vehicle controls or PCB requirements. This is shown in Table 6.

Table 6: Hardware Requirements

Deliverable	Specifications
Modes of Operation	The vehicle will be driven autonomously without use input.
Processors/Microcontrollers	At least 1 microcontroller or processor will process all sensor data. We may add one more microcontroller to control the vehicle and another to implement PCB functions

2.4.7. PCB Requirements

To meet our Senior Design requirements, the team proposed the following functions for a PCB that will be added to the vehicle. The details for our PCB are provided in Table 7.

Table 7: PCB Requirements

Deliverable	Specifications
LED Turn Signals	At least 4 LEDs will be powered by the PCB and blink according to the state of the vehicle. States are defined as: stopped, forward, turning left, and turning right.
LCD	THE LCD will display the current battery life, the current steering angle, and the speed of the vehicle in real time.
Temperature Management	The PCB will have a temperature sensor and fan for regulating heat.

2.4.8. Software Requirements

The final requirements for our project are our software requirements. The designed software will take input from all sensors for localization, be able to communicate the processed data wirelessly, be able to coordinate with other vehicles, and will be able to avoid collisions with obstacles on its path. This is summarized in Table 8.

Table 8: Software Requirements

Deliverable	Specifications
Processing Data	The software will process data from all sensors and data
Size	The software will not consume more than 4 GB of memory
Data Transfer	Data will be transferred wirelessly between different vehicles
Routing	Software will implement basic route planning.

2.5. House of Quality

The house of quality, as seen in Figure 1, is a design tool used to create and organize the marketing and engineering design requirements and their respective tradeoffs relative to one another. The 4 rows contain our house of quality's marketing requirements. These are the descriptive qualities that potential customers are drawn to. The columns contain our engineering requirements, which detail the specific performance metrics of

the system. These are created from a developer standpoint to compliment the marketing requirements.

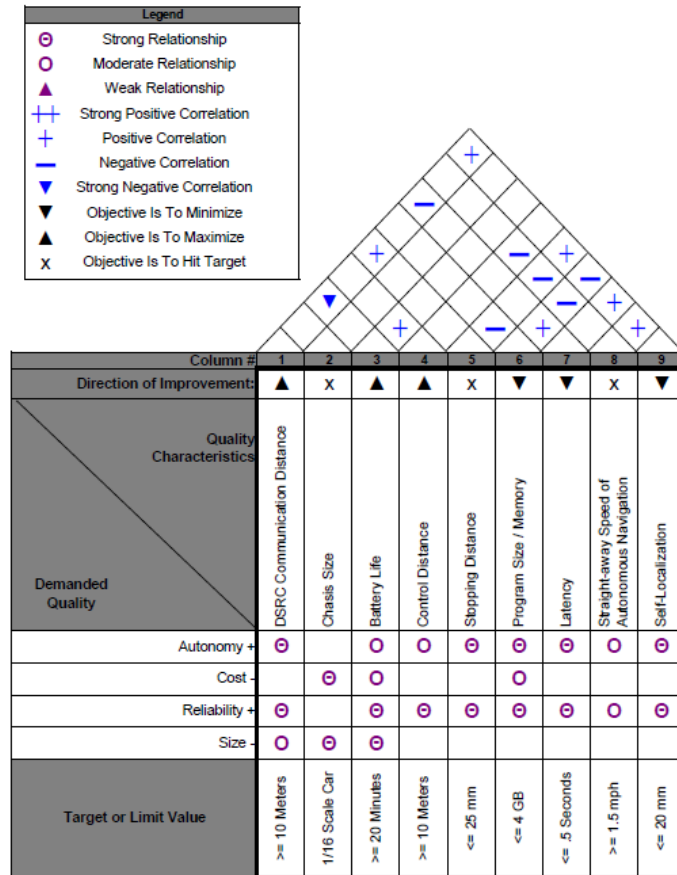


Figure 1- House of Quality

The marketing requirements we chose for our project are the cost, reliability, size, and autonomy. Lowering cost will save Dr. Fallah money that could be used towards other research projects in his lab. A smaller car will be easier to test inside of his research lab and generally will reduce costs as well. Should the car be tested in a different environment, a smaller car will be more versatile if the new environment is smaller than what was anticipated. Reliability must be maximized to provide the research lab with consistent data that can be used for analysis and development of a consumer sized self-driving car prototype. Autonomy must also be maximized because this is what our sponsor's research is primarily focused on.

The engineering requirements we chose are the Digital Short-Range Communication (DSRC) system's operating distance, Chassis Size, Battery Life, Control Distance, Stopping Distance, Program Size, Latency, Straight Away Speed of the Autonomous Vehicle, and Self Localization. The DSRC system's communication distance was chosen as ten meters because any of the vehicles we design would be able to communicate with each other from any 2 points in the research lab. The chassis size was determined to be 1/10 scale so we could have enough room to mount all of our equipment and sensors, reduce cost, and have a relatively small vehicle. The battery life was chosen to be at least 20 minutes because this provides enough time to test the cars and gather data from them.

Unfortunately, a tradeoff is present because a larger battery will most likely be necessary to increase the length of time the vehicle can operate between recharges, meaning our chassis would need to be larger or we would be more limited for space when placing our sensors. A larger battery would also distribute heat differently across the vehicle, meaning more heat protection might be necessary. The vehicle will be able to be remote controlled from a minimum of 10 meters because this will encompass the minimum communication range for the cars. The stopping distance was chosen to be 25 mm because this is a stringent condition that would force us to create a more reliable vehicle. We chose to keep our total program size below 4 gigabytes because we assume latency will increase with program size, which we want to reduce at all costs because latency will compromise reliability. The straightaway speed was chosen to be 30 miles per hour because a larger speed could lead to significant damage to the vehicle if a crash occurs, and the project is very expensive. The self-localization error was decided to be less than 20 mm to increase reliability. This is important for full size vehicles because they need to stay within the lanes of the road to avoid collisions with other cars.

3. Research Related to Project Definition

To construct our project, research into similar projects and relevant technology was conducted to assist with choosing our specific components and maximizing our performance within the budget.

3.1. Similar Projects

In this section we examine similar projects that were the inspiration for our project. The choice of software algorithms and hardware was heavily influenced by these projects.

3.1.1. MIT RACECAR

One of the main inspirations for our project was the MIT Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot (RACECAR) course videos from 2015. In this course, students learn to implement a self-driving car that can navigate a tunnel system it does not have a map of. The students are divided into 4 teams and then must compete against each other for the fastest time. The students were given an already assembled RC Car with an NVIDIA Jetson TK1 board and various sensors [85]; their only task was programming their vehicle. This differs greatly from our project, where we will modify the car ourselves and then program it to work autonomously. We are also using newer and more powerful hardware for our main CPU, and our vehicle is also going to be distinctively faster than the MIT vehicles. The maximum speed of our vehicle will be more than double that of the MIT cars. A picture of the MIT racecar is shown in Figure 2.



Figure 2- MIT RACECAR from 2016 (Permission Pending on Photo)

3.1.2. Zheng Wang's OPEN CV RC Car

Zheng Wang's project incorporated OPEN CV software and a Raspberry PI B+ model to create a car with computer vision that could perform in traffic scenarios e.g. stopping at a stop sign and recognizing the differences between red and green lights [86]. This inspired us to use the OpenCV tech for to simulate traffic scenarios with our car, but also go a step forward and incorporate pedestrian detection.

3.2. Relevant Technology

In this section we provide a brief overview of the sensors and important vehicle components to provide insight into how our vehicle will collect data and how the vehicle functions mechanically. Specific choices regarding performance, brand, and cost will be covered in the next section. These sections may be referenced in future discussions in the component selection and the project design

3.2.1. LiDAR Sensors

A LiDAR sensor records when the beam of light was fired and when the reflected light returns to the sensor. This measurement is double the time it takes for the light beam to travel from the object to the sensor [61]. Using this time and the speed of light, the distance of the measured object can be calculated using Equation 1.

$$\text{Distance Recorded} = (\text{time}) * \text{Speed of Light}/2$$

Equation 1. Distance recorded by LiDAR Sensor

LiDAR systems are typically composed of a laser, scanners/optics, photodetectors/receivers, and a positioning system. The laser is usually within the 600 to 1000 nanometer range at lower power levels or at the 1550 nanometer range. The 600 to 1000 nanometer range laser has the benefit of higher accuracy, but must maintain the lower power level to be eye safe. The 1550 nanometer laser has the advantage of longer reach and being invisible to night vision, but also has lower accuracy. The scanners and optics determine how fast and at what resolution and range the data can be entered into the system. Photodetectors are responsible for reading and recording the data that the laser returns to the LiDAR system.

Photodetectors are usually solid-state devices (e.g. a silicon photodiode) or photo multipliers. The final component is the navigation system, which helps the laser determine its orientation, velocity, and the position of the system when it fires the laser pulse [62]. Without knowing the angle the laser is fired at and where it is being fired, the data becomes unreliable for creating a point cloud. The navigation system is typically composed of a GPS and an Inertial Measurement Unit (IMU) to measure position and velocity. A generic LIDAR sensor is shown in Figure 3.

Generally speaking, LiDAR has two detection methods, coherent and incoherent detection. Incoherent detection only measures changes in the received signal's power. Incoherent detection can measure changes in phase and frequency, making it useful in applications where the Doppler Effect can take place. LiDAR also has two main pulse models called micropulse and high energy systems. Micropulse models use low energy lasers to gather data and are safe to the human eye. High energy systems are commonly used for atmospheric research for measuring cloud data (e.g. density, height, and pressure).

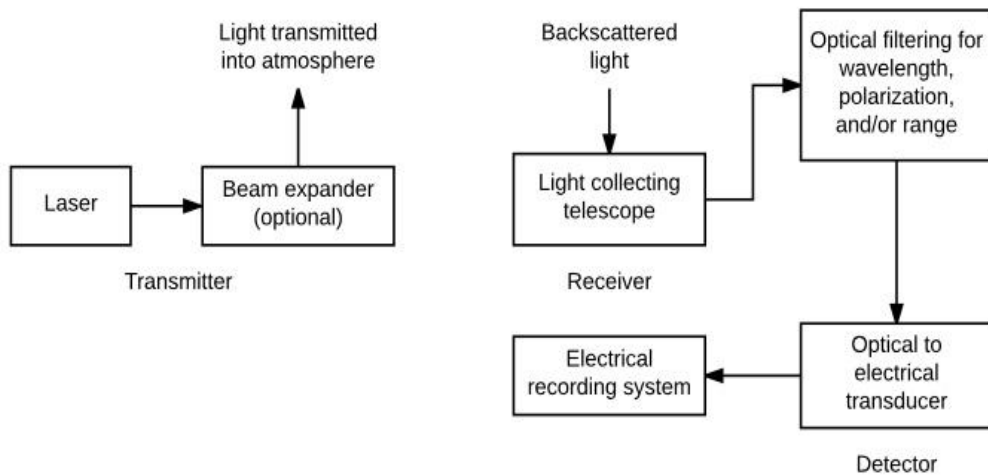


Figure 3- Generic LIDAR Sensor Block Diagram

3.2.2. Ultrasonic Sensors

Ultrasonic sensors are sonar sensors that use sound waves that are at a frequency above 20,000 hertz (see Figure 4). The reason this frequency range is used is because the human ear cannot process these sounds. The speed of sound can vary with temperature and the medium it travels through. Higher temperatures increase the speed of sound through a medium because hotter particles have more energy. Sound travels faster through liquids than gases due to the relative distance in particles being much smaller at the expense of needing more energy to propagate a noticeable sound wave. The same logic applies to solids and liquids, but even greater energy is necessary to propagate the sound due to the more powerful chemical bonds present. Ultrasonic sensors are used to measure distances within 3 meters of the sensors, and generally have an error of about 3 centimeters.

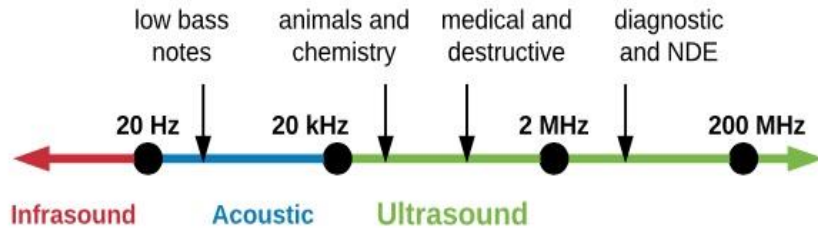


Figure 4- Sound Spectrum

Ultrasonic sensors have a small electro-acoustic amplifier that transmits a pulse called a ping. When the ping hits another object, it is reflected towards the sensor and then hits a tiny microphone that acts as a receiver. A generalized block diagram is shown in Figure 5. The distance is then calculated using Equation 2.

$$\text{Distance} = (\text{Recorded Time}) * \text{Speed of Sound}/2$$

Equation 2. Ultrasonic Sensor Distance Calculation

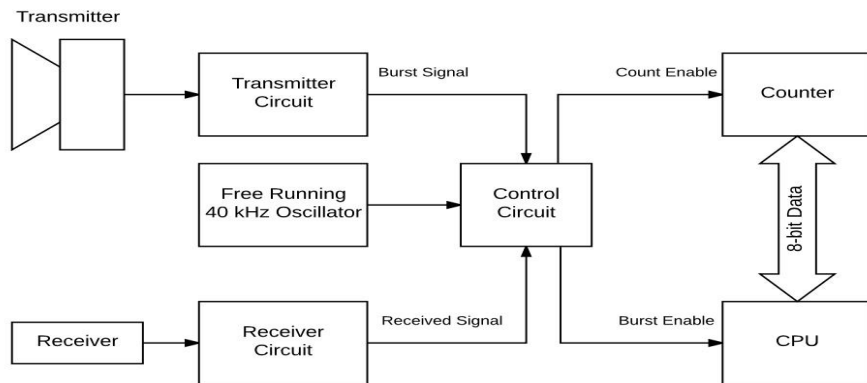


Figure 5- Ultrasonic Sensor Block Diagram

3.2.3. Infrared Sensors

Infrared sensors function by measuring radiation in the 300 GHz to 430 THz frequency range. Infrared sensors can exist as either passive or active sensors. Active sensors transmit infrared light and then detect the reflected light using a photodiode for processing [66]. Passive sensors detect the thermal radiation off of an object. Not all thermal radiation is in the infrared range, but

objects near room temperature and on the earth's surface mainly emit thermal radiation in this range. These waves are then used to construct the point cloud based on temperature at different points, with hotter objects emitting different frequencies in this range than colder ones. IR sensors are accurate within one meter.

3.2.4. Inertial Measurement Unit (IMU)

An IMU is an electronic component that has 2 sensors to measure the angular and linear velocity of whatever it is attached to. The 2 sensors are a triad of accelerometers and gyroscopes respectively [64]. The signals produced from these components are analog signals that are then put through an ADC converter for processing with a microcontroller using a Kalman Filter algorithm. A block diagram is shown in Figure 6.

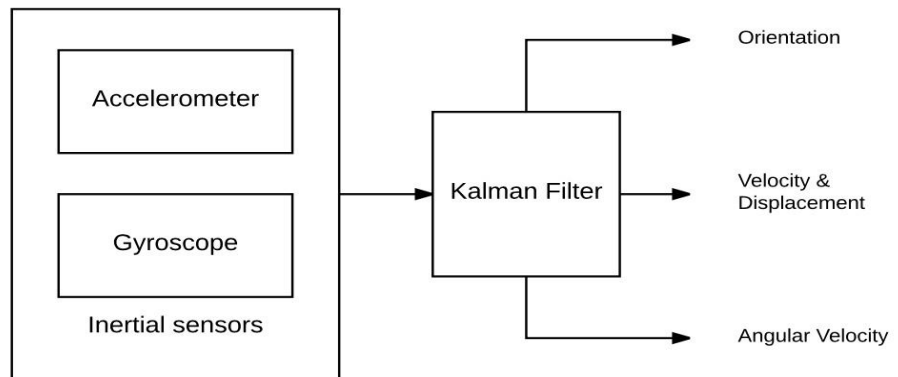


Figure 6- IMU Block Diagram

3.2.5. Differentials

Differentials are a mechanical device located at the bottom of a car that redistributes the torque from the engine to the wheels on an axle. This is necessary because when a car is turning, the inner and outer wheels on an axle travel different distances around the curve. For the car to make the turn, the outer wheel must be spinning faster so when the turn is finished, the wheels are parallel again. Differentials come in different types as well. Open differentials split the torque between the two wheels, but encounter issues if one wheel loses traction because the other wheel will experience a loss of torque to compensate. Any vehicle

that is not intended to go off-road uses open differentials (e.g. sedans, economy cars, and minivans).

A locking differential has a mechanism that makes the angular speed of the wheels remain equal until it is released. This is useful for terrains where traction is more difficult to maintain (e.g. snow or mud). These differentials experience difficulties if they lock on high traction surfaces like pavement because turning with wheels spinning at equal speeds can lead to stuttering or skidding. These differentials are present in off-road vehicles and some full-size trucks. Limited slip differentials combine the best of both worlds, and only have their locking mechanism activate when one wheel begins to slip [65]. These differentials are present in a small number of sports cars. Figure 7 shows these three differentials.

In four-wheel drive cars, a center differential is present as well to distribute torque to the forward and rear differentials. The center differential will always distribute more torque to the back differential.



Figure 7- Type of Differentials

3.2.6. PID Controller

Controllers are electronic systems that are designed to improve the response characteristics of a plant whose transfer function cannot be changed (e.g. a motor). In control theory, 5 basic types of controllers exist: Proportional (P), Integral (I), Derivative (D), Lead, and Lag. In most control systems today, the first 3 kinds of controllers are combined to create the PID controller. The integral and derivative controllers are tuned using the values T_i and T_d to create the desired response. Sometimes the constants in the PID

transfer function are represented using K_i and K_d . The transfer function of a PID controller is shown in equation 3 for both forms.

$$G(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s \right)$$

$$G(s) = K_p \left(1 + \frac{K_i}{s} + K_d s \right)$$

$$K_i = \frac{1}{K_p T_i} \quad K_d = \frac{T_d}{K_p}$$

Equation 3. Transfer function of PID controller

K_P refers to the gain of the proportional controller's gain. A higher K_P creates a larger overshoot, a smaller rise time, and reduces steady state error. K_i increases the settling time and overshoot of the system, but greatly decreases the steady state error and decreases the rise time of the system. K_d reduces the overshoot and the settling time. K_d has no effect on the steady state error and minimal effect on the rise time. Table 9 summarizes the effects of the components.

Table 9: Effect of PID parameters on system output

<i>PID Parameter</i>	<i>Rise Time</i>	<i>Overshoot</i>	<i>Settling Time</i>	<i>Steady State Error</i>
K_p	Decreases	Increase	Minimal effect	Decreases
K_i	Decreases	Increases	Increases	Decreases
K_D	Minimal Effect	Decreases	Decreases	No Effect

3.3. Parts Selection

In this section, a variety of potential components and their specifications are listed and compared along with their costs to provide a full cost benefit analysis. Our list of components includes microcontrollers, CPUs, a variety of different sensors, RC cars, batteries, and routers. Minor components used for testing will not be listed, and this list may be changed as the project progresses due to component failure or need for better hardware.

3.3.1. RC Car Selection

In creating an accurate model of a road vehicle for the purposes of crash-avoidance testing, one may begin with modifying a pre-built design to cater the specific project requirements. For the purposes of this project, we aim demonstrate the motion of a standard vehicle by use of an RC car. By scaling down the size a real-world application, one may fine-tune the various aspects of the system in a more workable manner. The specifications requested by the project sponsors are: a scale size between 1/10 and 1/8, an above average steering ability, replaceable or adjustable coil over suspension, rear-end differential to minimize slipping of tires, an aluminum or plexiglass chassis, and the availability of spare parts. The selection was narrowed to 4 vehicles. Table 10 displays the full specification analysis of the three cars discussed below.

Exceed RC makes the Sun fire Brushless Pro Off Road Buggy (See Figure 8) which comes manufactured with a lightweight aluminum alloy chassis, high capacity 3000mAh battery and an above average stock shock system. The long travel oil-filled shocks are made to handle high impact jumps and rough terrain by delivering a quick response. The car comes with quick access to the front and rear differentials for fast changes and customization. Powered by a brushless 3300KV electric motor the car reaches a top speed of almost 50 mph which exceeds our attempted straight-away speed. The Sun fire pro not only has the aluminum chassis but durable aluminum shock system and driveshaft. All motors and gear are fully sealed for guaranteed protection on all terrains.



Figure 8- Exceed Sunfire Pro

The Iron Track Shootout E8XBL (See Figure 9) has an impressive three-part differential drivetrain for maximum efficiency and customization. All differentials and suspension are made from hardened ionized materials and filled with silicon oils

to increase performance even under excessive use and heat. Its stock 2700mAh battery would provide an adequate run time to meet our specifications. While the E8BXL has a 2075KV motor that is the weakest of the three choices it would still be fast enough for all purposes of this project. It has a large adjustable big bore shock absorber system that makes for an above average handling capability. While both the Exceed Sun Fire and Iron Track E8XBL certainly satisfy most of our requirements there was a lacking of excess parts and manufacturer support which would be needed in an emergency scenario. This realization led us to our next choice.



Figure 9- Iron Track Shootout E8XBL

The Traxxas Rally racer was proven to be the best choice when compared to its competition and is shown in Figure 10. With a single motor powering all 4 wheels and 3-part differential system distributing its power Traxxas products have the most similar structure to an actual car. The Traxxas is superior in speed, versatility, battery life and has a higher availability in spare parts and support from the manufacturer. Its superior control system supports customizable throttle response, three different control modes, a near-zero latency control response, and optional wireless control that can be added and integrated with either Android or Apple operating systems. Internally this vehicle has easy access to components and motors making it an ideal candidate for total customization and manipulation of the electronics for the purposes of this project. The front and rear differentials will provide the least amount of slippage to keep all the sensors accurate for data measurement, localization and autonomous decision making.



Figure 10- Traxxas Rally Racer

The Traxxas Slash 4X4 Platinum addition (shown in Figure 11) is well worth the additional investment. It comes stock with aluminum upgrades, front and rear sway bars, high volume GTR shock system, and a performance-optimized low center of gravity chassis. The low center of gravity enables this vehicle to have superior traction compared to its rival models. Its silicon-filled center differential will more accurately model a real-world car for the purposes of this project. The battery option for this car enables up to a 5800mAh battery to be added for optimum run time performance. This car would be the most reliable option and would require minimal customization even though the cost is higher than the other options. This car does not come stock with a battery and gives several options. In an effort to meet our requirement of a 20-minute run time the 5800mAh battery selection was chosen.

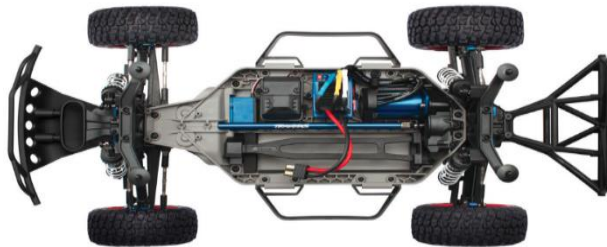


Figure 11- Traxxas Slash 4x4 Platinum

The Traxxas may not have the exact chassis material requested, but it can be easily modified to suit the project's specifications. A customized aluminum or Plexiglas chassis upgrade may be added to the car to increase its resiliency and reliability along with aluminum or Plexiglass mounts for the LiDAR, depth camera, and other sensors. A new center differential to replace the clutch and possibly a reduction gear may also be added to minimize the speed of the vehicle to increase its maximum torque so the car

will still be able to move fast and accelerate quickly, despite the car being loaded down with the relatively heavy equipment additions. The addition of a center differential will also allow for smoother turning and prevents the wheel from slipping if that becomes an issue. Other customizations to the drive train, axles, shocks and steering servo may be implemented with the help of our contributors to increase the maximum steering angle and increase stability due to the additional, unevenly distributed weight from the battery, CPU, any additional mounts that created, and the camera. Table 10 shows a summation of all the RC cars considered with the relevant specification for the project.

Table 10: RC Car Specification Analysis

Specification	Sunfire Pro	Iron Track E8XBL	Traxxas Rally Racer	Traxxas Slash Platinum
Scale	1/10	1/8	1/10	1/10
Cost	\$192	\$245	\$300	\$429
Size	L: 15.7 in W: 9.8 in	L: 19.2 in W: 11.42 in	L: 21.7 in W: 11.7	L: 22.36 in W: 11.65 in
Motor	Brushless 3300KV	2075KV	Brushless 3500KV	Brushless 3500KV
Suspension	Aluminum Shocks	Independent and Adjustable	Adjustable Oil-filled	Aluminum Shocks
Differential	Metal Gears	Gear Ratio: 11.3	Hardened Steel Bevel, LSD	Hardened Steel Bevel, LSD
Chassis	Not Specified	Plastic Nylon	Nylon Composite	Nylon Composite
Spare Parts	Yes	Yes	Yes	Yes
Battery	3000mAh	3 Cell Li-Po	7-cell NiMH	Optional
Wheelbase	10.8 in	12.8 in	12.8 in	12.75 in
Drive	4 Wheel Drive	4 Wheel Drive	4 Wheel Drive	4 Wheel Drive

3.3.2. CPU Selection

The selection of the processing unit is an integral part of this project. The central processing unit that is selected will be responsible for not only controlling the 4 motors and steering servo but for all of the image and sensor processing. The

Raspberry Pi or an Arduino board were among the initial choices of the group. Since none of the microcontrollers in these categories are going to be capable of the intensive amount of image processing from our LiDAR and camera, the solution we thought was optimal at the time was to have the board communicate the sensor data to a computer for processing wirelessly. The processed data could then be sent back to the vehicle for it to make decisions. This solution, however, would only increase the amount of delay between instructions and making the autonomous judgements of the vehicle slower and less reliable. Furthermore, it would make it impossible to test the cars in outdoor environments without a computer. Further research was done into different options for development boards and microcontrollers.

The discovery of NVIDIA development products was effortless since the market for self-driving vehicles and processors strong enough to support the amount of data generated are still extremely new. NVIDIA is a leading manufacturer of Artificial Intelligence Embedded software and development boards, their Jetson TX1 and TX2 boards excel in both the fields of robotics and image processing [71]. Figure 12 below shows NVIDIA Jetson TX2 performance versus an Intel Xeon, the data clearly shows the superiority of the NVIDIA product. The NVIDIA Jetson series boards were an obvious choice because they operate on a Linux platform which is necessary for the ROS robotics programming that will be used in the vehicles.

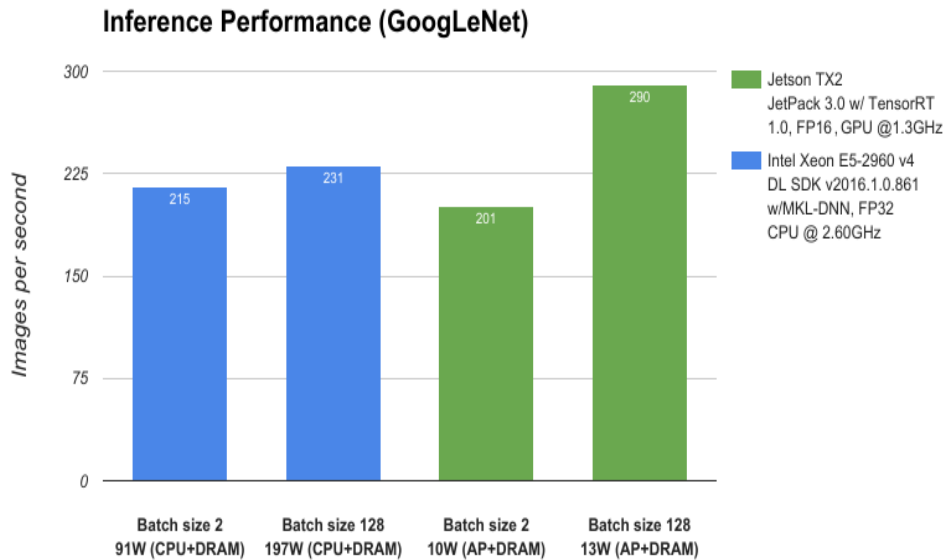


Figure 12- NVIDIA Performance Comparison

The clear advantages of the TX2 for this specific application became apparent upon further inspection of the board specifications. Both boards have header extensions for cameras and display, and have interfacing for UART flow control, I2C and SPI serial communications. USB 3.0 and 2.0 ports support recovery and host modes for programming and deep learning to suit the project's needs. The TX1 and TX2 models use an ARM 32-bit Quad core processor that has a frequency of 2.3 GHz. However, the TX2 has an additional Dual Denver 64-bit CPU that has a higher frequency of 2.5 GHz that boasts an internal memory cache of 128MB and can convert the ARM instructions into its own internal ISA. The dual Denver CPU can perform up to seven operations per clock cycle and has an optimized code sequencing. Its innovative dynamic code optimization compacts and stores frequently used software routines by converting them into equivalent micro-routines that can be reused and recalled from the cache memory.

The addition of the extra processor with its optimized code sequencing improves the power efficiency of the Jetson TX2 board when compared to the TX1 and its other competition. The Jetson TX2 outperforms its predecessor in High Efficiency Video Coding, memory, data storage, and its Camera Serial Interface. The TX2 is not only more efficient at processing camera data,

and instructions but it is more power efficient which will increase battery life and help meet our specification of a 20-minute run time on a single battery charge. The Table 11 below displays a more detailed comparison of these specifications.

Table 11: NVIDIA Development Board Analysis

<i>Specification</i>	<i>NVIDIA Jetson TX2</i>	<i>NVIDIA Jetson TX1</i>
CPU	Quad ARM A57 and a HMP Dual Denver	Quad ARM A57
Video Processing	Encoding: HEVC 4K x 2K at 60Hz Decoding: 4K x 2K at 60Hz with 12-bit Support	Encoding: HEVC 4K x 2K at 30Hz Decoding: 4K x 2K at 60Hz with 10-bit Support
Memory	8 GB / 128-Bit / 59.7 GB per sec	4 GB / 64-Bit / 25.6 GB per sec
Display	2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4	2x DSI, 1x eDP 1.4 / DP 1.2 / HDMI
Camera Serial Interface	6 Cameras in 2 Lanes 2.5 Gbps per Lane	6 Cameras in 2 Lanes 1.5 Gbps per Lane
Data Storage	32 GB	16GB
Serial Communication	CAN, UART, SPI, I2C, I2S, GPIOs	UART, SPI, I2C, I2S, GPIOs

3.3.3. Microcontroller Selection

With the NVIDIA development board controlling the autonomous movements and processing the data from our stereo camera and LiDAR we decided that the inclusion of an additional microcontroller to process data from Ultrasonic and Infrared distance sensors and an IMU would increase the accuracy of movement, object avoidance and the overall reliability of the project. For this purpose, we observed microcontrollers from Texas Instruments, Raspberry Pi and Arduino, a full analysis of the boards discussed in this selected is presented below.

3.3.3.1. Raspberry Pi Model 3

The Raspberry Pi microcontrollers were the first company to come to mind during the brainstorming sessions for product selection. These computers are a hobbyist favorite because of their incredible versatility and virtually limitless applications. They are used for applications ranging from home automation to complete online media centers and retro-gaming system

emulators. Since there is currently only one line of boards manufactured by this company we only discuss the most recent release.

The new model 3, shown is latest model and naturally the first choice from this product line. It has a 64-bit quad core ARM processor, however it effective available compatibility is 32-bit ARMv7. The board also has built in WIFI which would be an added bonus for this project since our vehicles will be communicating with one another over a WIFI frequency. While this board is indeed powerful it is bulky compared to its competition because of its multiple USB and HDMI output ports. It also requires 2.5 A power supply and would decrease the runtime of the cars significantly. This brings us to our examination of TI and Arduino products which as much lighter and more power efficient.

3.3.3.2. Texas Instruments (TI) Microcontrollers

Texas Instruments is another company that first comes to mind for many electronic and hobby enthusiasts. These boards were considered by the group to be a smart option for our microcontroller because every member in the group has past experience in our courses with this MSP430 development board. TI is also a great choice because of their user friendly Integrated Development Environment (IDE) that comes equipped with built in functions, variety of example codes and has an entire online community for any implementation problems we may encounter in the near future. A full comparison of all the TI boards presented in this section is displayed below in Table 12

Table 12: Texas Instruments Development Board Analysis

Specification	MSP432P401R	exp432F5529LP	CC3200	CC3220S
RAM	64 KB	66 KB	256 KB	256 KB
Memory	256 KB	512 KB	1 MB	1 MB
Power	1.6-3.7 VDC	1.8-3.7 VDC	1.8-3.7 VDC	1.8-3.7 VDC
Serial Communication	UART I2C SPI	UART I2C SPI	UART I2C SPI	UART I2C SPI
I/O Pins	48	80	27	27
Size	14 mm x 14mm	14mm x 14mm	58mm x 94mm	Not Found
Weight	3.5 oz	3.5 oz	3.5 oz	3.5 oz
Cost	\$25	\$27	\$32	\$39

The Texas Instruments (TI) MSP432P401R Launch Pad Development Kit would be a good fit for the tasks outlined above. It has a non-volatile memory of 256 KB, up to 64 KB SRAM with a 32-bit ARM Cortex processor with DSP acceleration. This controller has an extremely low power-active mode of 80 μ A/MHz and 660 nA standby mode and voltage operation of 1.6-3.7 VDC. Along with built in DC-DC converters and multiple 16-bit timers this board supports I2C, SPI and UART serial communication, allowing it to easily communicate data with our NVIDIA controller. This board has up to 48 pins for input and output and all pins come equipped with an interrupt enable. The applications of this board range from home automation, consumer electronics and health and fitness products.

The next board in the TI Line that was examined was the MSP-EXP430F5529LP Developer Board. Its memory is double that of the previous series with 512 KB of non-volatile flash memory and 66 KB of RAM. This unit has a standby power consumption of 2.1 μ A at an operating voltage of 3 V with a fast wake up time of 3.5 μ s . The highest of the four 16-bit timers in this unit has up to seven capture/compare registers and operates on a 32 MHz watch crystal for pristine accuracy. Other product features include: 12-bit Analog-To-Digital converter, full speed universal serial bus, and an enhanced auto baud rate detector for UART communication protocol. The main applications of this board is

data logging for analog or digital sensing systems which would be the exact purpose of a microcontroller for this project. Another advantage of the MSP-430F5529LP is the familiarity the design team has with the product. Each member knows how to program interrupts, use the onboard temperature sensor and LEDs, how to use an LCD with microcontroller, and how to change the clock speed due to previous classes involving this controller. Each member also possesses source code to accomplish all these things and more in both the C and assembly languages.

The CC3200 Launch XL was the next candidate from the TI microcontroller family. With an ARM Cortex-M4 that runs at 80 MHz and 256 KB RAM and 1MB of flash memory this board is much faster than the two previously examined. The CC3200 is more versatile. It supports multiple IDE platforms for programming through USB to a computer. Out of the box this board boasts an on board WIFI chip, 27 GPIO pins, 4 timers with pulse width modulation modes, 8-bit camera interface, on-board accelerometer and temperature sensor and advanced low-power modes. This is a great controller, but for a small price difference there is an upgraded version of this board with better specifications and additional features.

The final TI development board that was researched was the CC3220S-LAUNCHXL microcontroller. This board is extremely like the CC3200 but has integrated WIFI compatibility, which will be useful in sending data between our vehicles. This development board has two low-power modes, hibernate that draws 4.5 μA and Deep-Sleep which draws 135 μA . However, the problem with Texas Instrument products is the logic level and operating voltage of their products. The standard for TI is 1.8-3.7 V while the operating voltage of most sensors researched and the average peripheral components and LCD screens run on 5 V. This would require either a logic level converter or power regulation creating a larger circuit which should be avoided if possible.

3.3.3.3. Arduino Microcontrollers

Arguably the most popular microcontroller manufacturer currently on the market, Arduino provides customers with a wide range of boards and microcontroller chips to accomplish a variety of goals.

These boards are even more versatile than the Texas Instruments being that their logic and operational voltages are more in the range of most of the sensors and other peripheral devices that were researched. Like TI, Arduino also has an excellent IDE platform for easy programming and a vast community for troubleshooting purposes. A full comparison of the Arduino boards examined can be seen in Table 13 below.

Table 13: Arduino Development Board Analysis

Specification	Uno R3	101	Due	Mega 2560
RAM	2 KB	24 KB	96 KB	8 KB
Memory	32 KB	196 KB	512 KB	256 KB
Power	1.8-5V	3.3V	3.3V	5V
Serial Communication	UART SPI I2C	UART SPI I2C	UART SPI I2C	(4) UART SPI I2C
I/O Pins	Digital: 14 Analog: 6 PWM: 6	Digital: 14 Analog: 6 PWM: 4	Digital: 54 Analog: 12 PWM: 12	Digital: 54 Analog: 16 PWM: 15
Size	68.6mm x 53.4mm	68.6mm x 53.4mm	101.52mm x 53.3mm	101.5mm x 53.3mm
Weight	25 g	45 g	36 g	37 g
Cost	\$22	\$30	\$37	\$40

The Arduino Uno is a great board for hobbyists of all levels. Part of the Arduino 8-bit controller family this board runs at 20 MHz and can reach 20 MIPS at that frequency. The Arduino library is extensive and has functions for all kinds of sensors and peripheral devices, making programming their devices incredibly easy for a wide range of applications. The Uno has 6 low-power sleep modes, a power-save mode current of 0.75 μ A, and an active mode current of 0.1mA. The Uno also includes the following features: two 8-bit timers, a single 16-bit timer; Digital, analog and PWM specific GPIO pins; ADC converter, temperature sensor, and on-chip oscillator.

The Arduino 101 is a dual core microcontroller containing a x86 and 32-bit Arc cores that both clock in at 32 MHz. It uses an intel

toolchain that enables parallel processing to accomplish complex goals. This board has a 6-axis accelerometer to recognize gestures and enables blue tooth connectivity for wireless control from blue tooth enabled devices. This board has more GPIO pins compared to the Uno and was made in collaboration with Intel. Intel created an open source real-time operating system for the 101 that compiles Arduino code using static registers to execute a list of commands. The Arduino 101 can have current consumption as low as 250 μ A and has an operational voltage of 3.3 VDC.

The Arduino Due was made for larger scale projects and is based on a 32-bit ARM core with 54 Digital GPIO pins and 12 Analog pins. The board operates at 3.3 V and its pins are constrained at that voltage which could pose limitations with integrating sensors. However, the board has high flash memory, two separate SRAM memory banks, relatively low power consumption (800 mA at 3.3 V) and a clock speed of 84 MHz. The Due has four hardware UART ports for serial communication which enable parallel processing of different sensor data to the NVIDIA CPU. This board can be powered through either a dedicated micro USB cable or a barrel power connector.

The Arduino Mega 2560 was designed for more complex projects and advanced programming with its superior memory capacity. This board was made for robotics applications using a high number of sensors with its 54 GPIO pin count. It has 4 UART communication ports so a wide range of data can be sent to different locations and a 16 MHz crystal oscillator for accurate timing. The 101 board achieves a max throughput reaching 1 MIPS per MHz and its active-mode uses 500 μ A making it perfect for low-power application. This board would far exceed the needs of our application since the NVIDIA board will be doing the bulk of the data processing. For that reason, we will most likely not be choosing to use this for either our PCB or any other sensor based application.

3.3.4. Sensor Selection

The Microcontroller will process data from platform compatible Ultrasonic, Infrared sensors and an Inertial Measurement Unit. The sonar and IR sensors will be used to monitor certain ranges of distance and set on an interrupt so car can achieve safe stopping distances in emergency situations such a collision, short stops, or to avoid pedestrian traffic. The IMU will be used to obtain additional data on speed to be cross referenced with the NVIDIA readings to reduce and possibly eliminate errors in sensor data. ROS will subscribe to these sensor topics to help create an even more detailed map that will aid in localization, mapping, and autonomous decision making. Several sensors for each platform are presented and examined below. Availability, range, resolution, and price of these sensors will be the main determining factors of what board and platform will be used for this purpose. The specifications for the sensors chosen are displayed below in Table 14.

3.3.4.1. Infrared Sensors (IR)

Infrared sensors have a much lower range than Ultrasonic and will be used to detect closer range obstacles that could appear in emergency situations. DAOKI makes an Arduino compatible IR module specifically made for object avoidance in autonomous car applications. Its range is for 1mm to 25mm and uses 15mA current draw. Its binary output will trigger an interrupt and be able to help stop the car faster than if the object was detected through the camera or LiDAR. These sensors are available for purchase in a multi-pack from Amazon for an affordable price.

Adafruit's IR sensor is sleeker and has a higher range of close to 1m. This sensor outputs an analog voltage that corresponds to a specific distance (3V for 10cm, 0.4V for 80cm). This sensor would be compatible with every platform examined so far because it outputs a voltage that could be read by any pin. While having an actual value for the distance would be more ideal this sensor would consume more power. Another downside is the cost of this sensor is equivalent to the cost of 5 of the previous. Most other IR sensors have a much higher cost compared to the DAOKI sensor multi-pack with not much improvement in detection range or resolution.

3.3.4.2. Ultrasonic Sensors (Sonar)

For mid-range object detection, we will be using Ultrasonic sensors to monitor multiple directions. Since each vehicle will require several of these sensors keeping the cost low is crucial to stay within our budget. The majority of these sensors use only four pins (VCC, GND, Trigger, and Echo) and operate at similar ranges and voltages. The Trigger pin releases the sonic pulse at intervals and the Echo pin then receives the pulse and using Equation 2.

SparkFun makes an Ultrasonic sensor with a range of 2cm to 400cm and with an operating voltage of 5V. The best price found was a pack of 10 of these sensors for \$14.99 on Amazon. This price best fits our budget and was the main factor in the purchasing decision.

3.3.4.3. Inertial Measurement Unit (IMU)

The IMU is an integral part of the robot as it will measure data that will aid the self-localization of the car within its environment. The NVIDIA could also measure velocity using the stereo camera and LiDAR but these values could have some discrepancy due to the latency of image processing. Adding an additional sensor will be useful to find errors and improve accuracy.

Adafruit makes a 3-axis accelerometer that can measure to 16g with a 57mV/g sensitivity. The sensor uses a supply voltage of 3-5VDC which would work for a variety of platforms. The price of this sensor was well within our budget but we found another that will include angular velocity to our data with a minimal cost increase.

SparkFun also makes an affordable Arduino compatible IMU Breakout board. The 9DoF breakout module contains a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer for a total of 9 degrees of freedom. Its serial communication supports SPI and I2C that will have the ability to communicate with the NVIDIA board running our ROS. This board can measure angular velocity for turning speeds and linear acceleration for speed. The additional data will aid the car in its autonomous decision making.

3.3.4.4. GPS Sensor Module

The GPS module is a valuable component because ROS will be using it to localize the robot and map its environment as accurately as possible. The GPS is important to integrate into this project because it is found in most new models of cars on the market today, and creating the most accurate model of a full-size autonomous vehicle that would be sold in the future is the overall objective of our design.

GPS standards ensure that most modules work very similarly with little deviation. The main differences from companies like Adafruit and SparkFun are the logic voltages, and communication protocol. Since compatibility is of the utmost importance, the SparkFun GPS module was an obvious choice. It is compatible with logic voltages ranging from 3.3 to 5VDC and would work with either Texas Instruments or Arduino microcontrollers. The module uses the UART serial communication protocol and would be able to communicate with both our microcontroller and NVIDIA computer system. Table 14 shows a summary of the sensors we have covered up until this point.

3.3.4.5. Temperature Sensor and Fan

SparkFun manufactures a great temperature sensor called a TMP36. This sensor requires a 2.7 to 5.5VDC for power which will make it compatible with any of the microcontrollers outlined above. The sensors sensitivity is 1°C to 125°C which will include the operational temperature constraint of 80 degrees for the NVIDIA board we are trying to monitor. This temperature range will also suit the acceptable range of temperatures for our Auxiliary battery. The sensor will be placed between the battery and NVIDIA board to monitor the temperature between the two and within the outer shell of the car as a system.

The sensors output voltage is 10mV per degree Celsius, this signal will be read by the ATmega IC chip and within the critical temperature range trigger a 2x2 DC powered fan for cooling. The sensor only requires 3 GPIO pins, one for power supply, one for grounding and one data line. These sensors are extremely

affordable and because of its versatility in voltage requirements was chosen for our design. The sensor was purchased in a 3 pack on the amazon website for less than \$10 and the 2x2 fan was purchased from a local RadioShack for \$5.

Table 14: Arduino Sensor Specifications

Specification	Ultrasonic Sensor	DAOKI IR Module	SparkFun IMU	SparkFun GPS-14030
Detection Range	2-400 centimeters	2-30 centimeters	± 2, 4, 8, or 16 g ± 245, 500, and 2000 °/s	Not Found
Resolution	1 centimeter	N/A (Binary Output)	± 2g - 0.061 mg ± 245 °/s - 8.75 mdps	Not Found
Sensitivity	0.3 centimeter	Not Found	Not Found	Not Found
Power	5 VDC	3-5 VDC	1.9-3 VDC	3.3-5 VDC
Cost	\$15 for 10	\$9 for 5	\$25	\$13

3.3.5. Stereo Image Sensor

Among the system of sonic and light sensors, each unit will possess a stereo camera for the purposes of capturing images and interpreting data in a more familiar way than that of LiDAR or SONAR. By using a camera with two lenses working in stereo, this module will enable the system to complement the sensors with a three-dimensional view of the surrounding area. In creating machine vision, the goal is to achieve a model that is both resolute and able to be processed digitally. Typically, digital optics record and process images in two dimensions; however, for the purposes of this product, the ability to capture depth is nearly a necessity for the intelligence in the system's ability to localize and predict motion of other objects.

A stereo optical scanner is simply a device that makes use of the same principles which enable our brains to process the distance to an object - depth. In a two-dimensional image, one may attempt to classify objects as either near or far; yet, without a second reference, these assumptions would not be sufficient for this project. Using images taken simultaneously from each lens, a stereo image sensor employs trigonometric methods to calculate the distances of all points in the field of vision. While the

ability to calculate distance from the aforementioned methods is timeless, the digital technology of stereo optics has existed for just nearly a decade. More recently, even, is the functionality of processing the images internally - providing the sensory output as a full-color representation of a three-dimensional environment.

There exist, on the market, multiple devices that achieve three-dimensional machine vision. Various products were researched and compared for optimal functionality. Several factors that were considered of importance are as follows: resolution, range, frame rate, field of vision, and illumination method. Resolution is clearly a significant aspect to sensor quality. The amount of precision and clarity of data output per input greatly influences the quality of the three-dimensional model; this is of great importance in an action scenario for properly determining the spatial coordinates of the vehicle's surroundings. Range, secondly, is integral in the efficacy of the system, as a whole. The stereo camera, in itself, serves as a supplemental means of sensory input to that of the LiDAR scanner; therefore, the range has to be of comparable proportions. This range must be large enough to observe objects that the two-dimensional LiDAR detects, in order to cohesively create the three-dimensional model in parallel with the other sensory inputs. Frame rate will ultimately contribute to the data refresh rate - a high frame rate ensures that the motion and velocities of objects are calculated in a seamless manner. Field of vision and illumination method are qualities that vary greatly between models of digital stereo vision sensors. The field of view must be such that no object of considerable "danger" is frequently in a blind spot. For this project, it was determined that horizontal field of view is more relevant than vertical.

The sensor that was selected was the ZED stereo camera. For comparison purposes, a similar contender, the Structure.IO model specifications will be observed. In the following Table 15 one may note the differences in the products and observe why the decision lay with the ZED sensor. The ZED camera possessed multiple modes of video recording, one of which boasts a fine resolution of 1344x376 megapixels. This will be sufficient for object detection to a precise degree. The range of 20 meters aligned with both the LiDAR range (the range greatly exceeds the LiDAR) and the desired environment scale, indoors. 20 meters would easily capture the span of a typical room or lab

for testing capabilities. The frame rate of 100 fps was among the top in the market, providing the system with a high frequency of image refreshing for motion detection. While other models could detect multiple spectrums of light, visible light was deemed acceptable for the needs of this project. The wide field of vision of 110 degrees (max) arises from the lens quality of the ZED camera. Five Volts DC was optimal for the USB hub connection; and while other models were able to be processed on multiple operating systems, the ZED camera operates on the only operating systems in use for this project.

Table 15: Stereo Image Sensor Analysis

Specification	Sense 3D Sensor	Sturctured.IO Stereo Camera
Resolution	1344x376 megapixels (max)	640x480 megapixels
Range	20 meters	3.5 meters
Frame Rate	100 fps	30/60 fps
Field of Vision	110 degrees Horizontal and vertical	58 degrees horizontal 45 degrees vertical
Illumination Method	Visible light	Visible light and Infared
Power	5 VDC	5 VDC
Hardware Requirements	Windows, Linux, ROS	Windows, IOS, Linux, Android Operating Systems
Cost	\$449	\$449

3.3.6. LiDAR Sensor

The primary means of gathering data around each vehicle is by use of a sweeping LiDAR module. In machine vision, a LiDAR sensor proves extremely practical in gathering precise data on the positions of surrounding obstacles and objects. The primary advantages to using a sweeping LiDAR sensor compared to a camera are the following: creating a 360 degree 3-D map upon each revolution, emission of light provides sensor input independent of the ambient room lighting, and much less prone to interference than that of SONAR or RADAR [27]. Due to the usage of lasers, a LiDAR module provides a more reliable and precise definition of the world around the machine.

This precision is a necessity for the system to visualize in every obstacle, regardless of the interference. Much like a human driver, influences such as a glare, blind spot, or lack of light often result in vehicle collision; through the employment of LiDAR, these scenarios can be avoided. Glare and/or interference is cancelled through the very spectrum of the light emitted and being sensed. The laser LED in the Scanse Sweep module is an infrared laser of 905 nm [28]; this falls into the infrared spectrum. While an infrared laser and sensor would obviously be sensitive to heat, the advantage is that these heat “shadows” are stationary, and do not travel with a moving object at ambient temperature [27]. Hence, the LiDAR will create a monochromatic representation of standard objects while simultaneously being very sensitive to warm bodies, clearly a target to avoid. A blind spot is another issue we drivers face on the road as our field of vision may only be directed in a certain window at one time, leaving the opportunity for a myriad of unfortunate circumstances. With machine vision using a sweeping LiDAR sensor, the system receives a 360-degree input at a set number of revolutions per second. This directional range of sensory data is unparalleled with any conceivable method of standard vehicle operation. Lack of ambient light is yet another hindrance to drivers. While there are alternatives to sensing in darkness aside from LiDAR, LiDAR proves extremely effective, if not more effective, in sensing objects in complete darkness. Again, due to the pulsating laser output, LiDAR is able to emit a frequency independent of ambient light, or lack thereof, and sense where these emissions reflect.

It is noteworthy to discuss why LiDAR is not the sole sensor in project design. While LiDAR is a revolutionary way in which we can bypass difficulties in visual sensory data, there are always tradeoffs in any system. The aim, here, is to discuss where LiDAR falls short and why it should not be the only sensor in intelligent vehicle design. LiDAR can be deceived in several ways, such as interference from an alternate LiDAR transmitter, or the calibration between receiver and transmitter [29]. In the Scanse Sweep SEN 14117, fortunately, one may program settings to an individual pattern of laser pulses so that these are unique to that LiDAR’s receiver. This is not the case in every module, and may present a problem for other designs. Similarly,

the receiver must be very attuned to the precise rate and pattern of emissions, or the data could be greatly skewed. Additionally, LiDAR does not function well with non-rigid obstacles such as water vapor or any fluid state [28]. The laser may reflect off of the individual particles in a fog, thereby confusing the receiver. Furthermore, a sweeping LiDAR has a latency when it comes to short range detection, necessary in collision avoidance scenarios. For these reasons, a SONAR sensor shall be outfitted to the vehicle to complement the regions where the LiDAR and stereo camera are lacking.

The Scanse Sweep SEN 14117 was chosen to outfit the design specifications in terms of an efficient LiDAR module. This product exhibited traits that deliver the aforementioned advantages of a sweeping LiDAR over other sensor types. Several metrics to consider when choosing an appropriate LiDAR sensor for use in robotic vehicles are: range of vision (distance), horizontal degree of vision, scan frequency, resolution, and cost (cost included here due to the expensive nature of LiDAR modules). Range of vision is clearly a desirable trait in any visual sensor. While LiDAR boasts an increased range compared to a visible light or SONAR sensor, it is best to maximize this value as machine intelligence takes time to process an object on course for a collision. Horizontal degree of vision is certainly a characteristic that separates the utility of one LiDAR sensor compared to another; a 360-degree field greatly increases the efficacy of any sensor in a spatial environment. While not all LiDAR modules are 360-degree, in the situation of vehicle intelligence, it proved to be a feature worth the cost. Rotation frequency is directly related to the update rate of data, much like frames per second of a camera sensor. A high rotational frequency leads to a sensor transmitting a moving model of its surroundings with great accuracy. Resolution is, again, a factor which tends to be apparent across all sensors. High resolution is sought to provide a clear input at minimal means of output. Finally, cost is considerably variant among LiDAR scanners; this was sought to be kept minimal while a 360-degree rotation was kept as a must-have feature for this model. The Scanse Sweep SEN 14117 was compared to various other LiDAR sensors. While it is not the premium device on the market, it met the needs of this project at a reasonable cost. Below, in Table 16, the comparison between the Scanse Sweep

SEN 14117 and a similar product, the RPLiDAR A1M8 to see why the former was preferred:

Table 16: LiDAR Sensor Analysis

Specification	RPLiDAR A1M8	Scanse Sweep SEN 14117
Resolution	0.019 inches	0.4 inches
Range	6 meters	40 meters
Field of Vision	360 degrees horizontal	360 degrees horizontal
Rotation Frequency	Up to 10 Hz	Up to 1075 Hz
Power	4.9-5.5 VDC	5 VDC
Hardware Requirements	Intel core i5 or equivalent	Windows, IOS, Linux, Android Operating Systems
Cost	\$199	\$349

3.3.7. Auxiliary Battery Pack and USB Hub

To power the NVIDIA and Arduino boards along with our other components an auxiliary battery will be need to be added as to not interfere with the battery capacity of the RC car decreasing the car’s run time. The NVIDIA board will need a 19V power supply so portable laptop battery chargers were researched. The power bank must have enough capacity to meet our 20-minute runtime specification.

The first power bank that was researched was the Lizone Extra Pro. It has a battery capacity of 40,000mAh. It has extremely high customer ratings and doesn’t seem to have current draw limitations like other power banks. The bank has two USB ports that support 2.1A charging and one 19V port for laptop charging. However, the storage capacity of this power bank may not be entirely meet our runtime objectives so other products were observed.

MAXOAK sells a 50,000mAh battery pack that should much better suit our requirement of a 20-minute runtime. The battery pack has two 2.1A 5V USB ports and two 1A 5V USB ports, with the addition of these two extra ports it could support a few of our sensors as well as our Arduino Board. Included in the outputs is a 20V and a 12V output for laptops and notebooks that support up to a 4.5A current draw. This device not only has more outputs

than the Lizone battery but it rated equally as high and has a lower price point.

In order to not only deliver power but data to all components from the portable battery and CPU respectively, a USB hub must be used. This product must have compatible 3.0 ports to work with all devices in this project that will deliver sufficient amperage required by our components. The chosen product was an AUKEY powered USB hub. This device has 3 charging ports that output 2.4 A each and 7 USB 3.0 data ports so our microcontroller, LiDAR, and Stereo camera can all communicate with our main CPU.

3.3.8. Wireless Router and USB Network Adapter

The vehicle-to-vehicle communication will be achieved through DSCR communication using a wireless frequency of 5.9 GHz. In order to do this the cars will be outfitted with a USB wireless adapter that will all communicate with a central wireless router. The decrease in the latency of the wireless communication is a parameter that cannot be overlooked since the timely transmission and receiving of important data will improve the reliability of these robots.

Linksys is probably the most widely known and frequently used wireless routers. The router from this company that was examined was the dual-band E2500. This device has a maximum data transfer rate of 300Mbps and works with 2.4 and 5GHz frequencies. It is compatible with Windows or Mac operating systems and comes equipped with 4 Ethernet ports for connectivity. While Linksys is a more notable and preferred brand for many users its price is much higher and would negatively offset our budget.

TP-Link is a notable manufacturer of wireless internet products with affordable price points. They make routers with transmission speeds of 300Mbps to 5334Mbps. However, the higher data rate routers have an extremely high cost and far exceed the needs of this project's communication specifications. The TP-Link TL-WR940N would be an excellent candidate because it has above average data transfer rates and a lower cost compared to routers of the same caliber from other manufacturers. The 3 antennas

help to increase the robustness of its data transfer signal. Compared to the Linksys it is not superior in speeds but it should meet the project specifications and help us keep our budget on track.

Speed tests from technology sites like *tomshardware* and *smallnetbuilder* were extensively reviewed for multiple top brands of routers within our price range (Linksys, Net Gear, and TP-Link). The tests revealed that most wireless routers generally do not meet their full advertised wireless transfer speeds. Reaching the highest speeds possible requires the tinkering of settings. With overlapping networks common in buildings and neighborhoods changing the wireless channel can improve wireless signal strength, bandwidth, and range. Speed tests will need to be ran and settings modified to be able to reach the highest possible range and data transfer rates defined in our requirement specifications.

For the USB network adapter, the first candidate was the TP-Link N300. This product would be great for the purposes of our project since it has a data transfer rate of up to 300Mbps that is suitable for even online gaming purposes. This product is extremely compact and lightweight which will help to reduce the overall mass of our vehicle and save much needed space for other components in our design. It is compatible with Linux Kernel 2.6.18-3.10.10 which is important because it will be utilized by our NVIDIA CPU which operates on a Linux Operating system.

The Diza100 adapter has 802.11ac dual band operation with a transfer rate of 433Mbps at 5.8GHz frequency. This definitely suits our needs since wireless communication for this project will utilize 5.9GHz and our TP-Link Router has made speeds of 300Mbps. This device is only slightly out of the cost constraint but it is higher rated than the previous devices with more reviewers. This device is also Linux operating system compatible across multiple versions. This device has a simple set up with a WPS button to automatically search for and connect to a Network and a detachable antenna option.

3.3.9. Summary

The final selection of parts was a group effort between our senior design team and our project sponsors and contributors. Decisions were made according to market availability, cost and performance. All items purchased had to be approved by our primary sponsor Dr. Fallah before they were bought. The item list is displayed below in Table 17, and photos of the physical components in-hand are shown in the Figures 13 and 14 below.

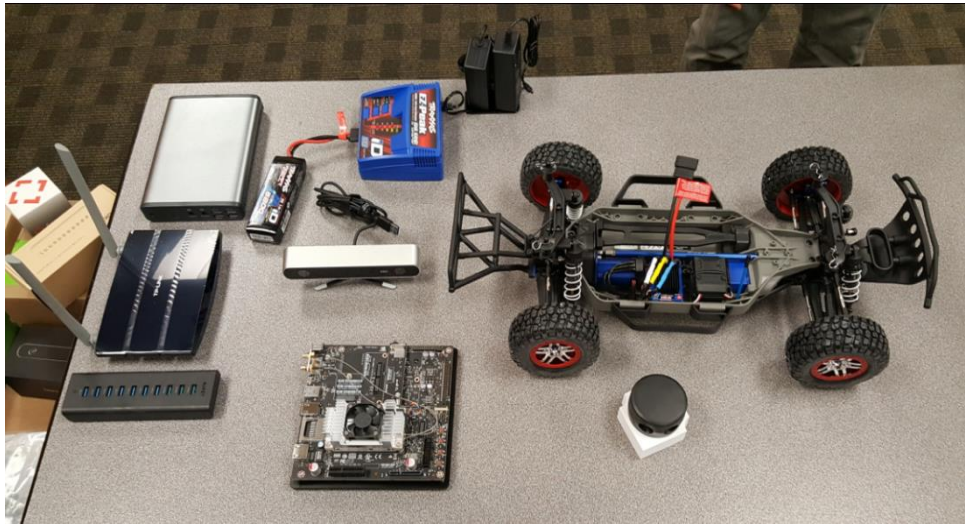


Figure 13- Purchased USB Hub, Router, Battery Pack, Stereo Camera, NVIDIA CPU, Traxxas RC Car and LiDAR

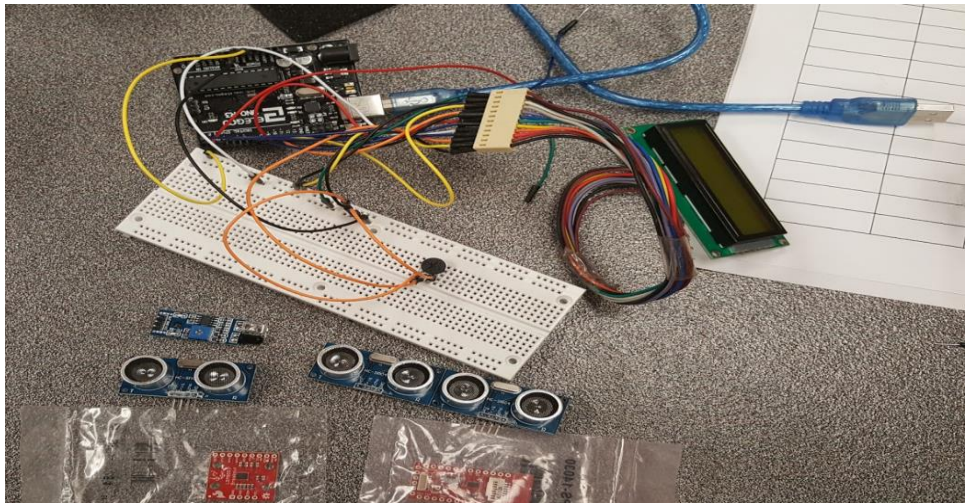


Figure 14- Purchased Arduino Uno, LCD for testing, Ultrasonic Sensor, IR Sensor, IMU and GPS Shield

The major investments for this project were the Traxxas Slash 4x4 Platinum, the Zed Stereo camera and the NVIDIA Jetson TX2. The NVIDIA's high-class performance in image processing made it possibly the most necessary component of this project. With data coming from both a LiDAR and a stereo camera, and a dense optical flow coding sequence we needed a computer system that would handle all of this data with minimal latency so our vehicle can make faster and more decisive autonomous movements. NVIDIA also offers an education discount which made the selection of this vital piece even easier and helped keep the project within the budget.

The next component that was heavily discussed and debated was the vehicle. The goal of our project is to model an actual autonomous car that would be sold in future markets and acquire accurate data that could be applied in the design of a full-size car. Traxxas was undoubtedly the best manufacturer researched and the Slash platinum is well worth the investment. This RC car is the closest to real-world vehicle dynamics that we could find on the market. It has superior materials, access to upgrades and spare parts, and excellent customer service available to our team. The Slash Platinum was over our initial budget constraint given by our project sponsors, however, after much discussion and with the savings from the NVIDIA student discount it was decided to be the best choice to meet the project goals.

The ZED stereo camera will allow the optical flow program in sensing pedestrians, signs, traffic signals and other cars while also providing a First-Person View (FPV) on the vehicle for manual control from significant distance away. On top of the benefits previously mentioned this camera will provide the car with an additional depth sensing capability to record distances from all objects seen within its range. This camera was chosen because of its accuracy and its compact size, which is an important characteristic is given all of the components we will be adding to this vehicle.

Our initial design for the LiDAR sensor was to use a 3D sensor and an additional mount to obtain a true 360-degree view (in all directions) of the car's surroundings. The mount however, was an expensive component and due to time constraints designing and manufacturing our own mount was out of the question. The

sacrifice of this mount was made so other higher priority components could be invested in for the overall benefit of the project. Without the mount our LiDAR sensor will still provide a sufficient 3-Dimensional model, on a single plane of view, for our data.

Our microcontroller research was extensive, but quite revealing. Without it, we would have encountered multiple implementation issues in the near future. After sensor research it appeared that the majority of them available on the market require 5VDC operational and logic high voltages. As discussed above in the Microcontroller Selection section this would cause issues with Texas Instrument devices and would require additional converters which would complicate our design. With a very limited budget due to our major investments above the selected controller was the Arduino UNO.

Arduino has significant resources available and the proper logic and supply voltages for sensors. In fact, there are many more sensors made specifically for Arduino Microcontrollers which will reduce possible errors we could encounter. Instead of using the full microcontroller on the vehicle we will be incorporating a preprogrammed Arduino IC into our Printed Circuit Board (PCB) Design. The UNO Rev3 chosen because it has a removable IC chip for these purposes as well as a low cost to fit within our budget. All sensors were with Arduino compatibility and moderate cost were also chosen for exact models.

Once the Arduino microcontroller was selected the appropriate and compatible sensors were chosen and purchased. Additional sensors requested by project sponsors were Hall sensors to measure motor speed to aid localization and autonomous decision making. Temperature sensors for the PCB design were researched and results yielded similar values for sensitivity and dynamic range for many of these sensors available on the market. The purchasing decision for these two sensors were easy. For the Hall sensor we purchased a model from Traxxas to ensure its compatibility and a popular temperature sensor was chosen according to online reviews.

Table 17: Final Product Selection

Component	Selection
Radio Controlled Car	Traxxas Slash 4x4 Platinum
RC Car Battery	5600mAh
Main Processing Unit	NVIDIA Jetson TX2
Microcontroller	Arduino UNO Rev3
IR Sensor	DAOKI IR Sensor Pack
Ultrasonic Sensor	SparkFun Ultrasonic Sensor Pack
Inertial Measurement Unit	SparkFun 9DoF Breakout IMU
GPS Module	SparkFun GPS-14030
Stereo Image Sensor	Zed Stereo Camera
LiDAR Sensor	Scanse SEN14117
Auxiliary Battery Pack	MAXOAK 50,000mAH
USB Hub	AUKEY Powered USB Hub
Wireless Router	TP-Link TL-WR940N
Wireless USB Network Adapters	TP-Link N-300 Adapter
Temperature Sensor	TMP36
Hall Sensor	Traxxas RPM Telemetry Sensor
LCD Screen	HD44780 Size: 16x2
Additional Arduino ICS for PCB	ATmega328

4. Project Constraints and Standards

Alongside the engineering requirements, we also must consider relevant industry standards for design to provide a quality product. Constraints unrelated to our engineering requirements, such as ethics, environment, and government policy, will also be explored.

4.1. Project Standards

There are many standards to consider while developing the NSL Self Driving Car. Wireless and wired communication, sensors making use of the electromagnetic spectrum, and powered electronics all have regulations to ensure proper use and standardized operation. There's also the Department of Transportation (DoT) guidelines to take into account in pathfinding algorithms such as lane width, speed limits, and many more road rules.

4.1.1. Power Supply Standards

CUI defines standards for power supply safety, detailing which components can be used to develop a power supply, what classification the power system falls under, and which insulation to use for the system [37]. Table 18 outlines the circuit definitions according to CUI; according to this table, the Self Driving Car system would fall under Extra-Low Voltage. It is critical to consider this standard while designing the PCB and other electrical components.

Table 18: Circuit Power Specification

<i>Circuit Type</i>	<i>Circuit Definition</i>
Hazardous Voltage	Any voltage exceeding 42.2 V _{AC} peak or 60 V _{DC} without a limited current circuit
Extra-Low Voltage (ELV)	A voltage in a secondary circuit not exceeding 42.2 V _{AC} peak or 60 V _{DC} , the circuit being separated from hazardous voltage by at least basic insulation.
Safety Extra-Low Voltage (SELV) Circuit	A secondary circuit that cannot reach a hazardous voltage between any two accessible parts under normal operations or a single fault. Under fault, ELV limits are satisfied. Limits of 71 V _{AC} and 120 V _{DC} must not be exceeded. Must be double-insulated from hazardous voltage. Considered safe for operator access.
Limited Current Circuits	Circuits may be accessible even though voltages > SELV requirements. A limited current circuit is designed to ensure non-hazardous current under fault. For frequencies < 1 kHz, steady state current ≤ 0.7 mA peak AC or 2 mA DC. For frequencies ≥ 1 kHz, the limit is 0.7 mA * frequency (kHz) but shall not exceed 70 mA. For accessible parts not exceeding 450 V _{AC} peak or 450 V _{DC} , the max circuit capacitance is 0.1 μF. For accessible parts not exceeding 1500 V _{AC} peak or 1500 V _{DC} the max stored charge is 45 μC and available energy shall not be above 350 mJ. Must have the same segregation rules as SELV circuits.

In the short-term, the system must be safe and operational for the Senior Design showcase and satisfy basic industry

standards. In the long-term, if the product is to be generalized for market and research, the system must conform to international standards. There are several such international product conformance marks, or regional safety marks, such as CE (European), UL (USA), CSA (US and Canada), GOST-R (Russia), PSE mark (Japan), and CCC (China).

4.1.2. Wi-Fi DSRC Standard (802.11b)

The topic of wireless communication, which will be the primary method of connecting the network of vehicles, is very involved. The main organizations to consider are Institute of Electrical and Electronics Engineers (IEEE), International Organization for Standardization (ISO), European Telecommunications Standards Institute (ETSI), Federal Communications Commission (FCC), and the Department of Transportation (DoT).

The two methods of wireless communication currently under consideration are standard WI-FI as described by 802.11(b) and Dedicated Short-Range Communication (DSRC), which is a modification to 802.11.

IEEE defines 802.11b as having a data rate of 11 Mbit/s using the same media as the original standard. Interference can be experienced in the 2.4 GHz band, which is used by this standard, from such products as microwave ovens, Bluetooth devices, and phones.

This disadvantage leads us to consider DSRC as a medium of wireless communication. In October 1999, FCC dedicated 75 MHz of the 5.9 GHz band to be used by intelligent transportation systems (ITS, which includes but is not limited to self-driving technologies).

The bandwidth allows for 1 control channel and 6 service channels [38]. In August 2008, ETSI also allocated 30 MHz of the 5.9 GHz band. The two are incompatible and used in different ways, however the allocation is present for use by ITS. DSRC is currently promoted by the DoT as the method of wireless communication. ISO and the European Committee for Standardization (CEN) have several standards related to DSRC as listed in Table 18.

Each standard in the table addresses a different layer in the OSI model of the network implementation of DSRC. Considering the physical layer require DSRC be transmitted at 5.9 GHz, a natural implementation would be a modified 802.11b system to transmit and receive at 5.9 GHz instead of 2.4 GHz [39].

Another method of short-range wireless communication is Bluetooth, however the distance of communication over Bluetooth is much shorter as compared to Wi-Fi. The highest range (100 m) is only possible with a class 1 device [40], which requires much more power than Wi-Fi would. DSRC over Wi-Fi would allow a communication range of 1000m while satisfying the low power requirement of an embedded system [41].

4.1.3. Frequency Allocation

While a US-specific implementation could use the 9.02-9.28 GHz range, or a Japan-specific implementation could use a 7.15-7.25 GHz range, Table 19 suggests 5.8-5.9 GHz would provide the most interoperability internationally. This strengthens the decision to use WI-FI as our DSRC medium, because Bluetooth only operates at 2.4 GHz.

Table 19: Spectrum Allocations

<i>Region</i>	<i>Frequency (GHz)</i>	<i>Reference Documents</i>
<i>ITU-R (ISM band)</i>	5.725-5.875	Article 5 of Radio Regulations
<i>Europe</i>	5.795-5.815 5.855-5.905 5.905-5.925	ETS 202-663, ETSI; EN 302-571, ETSI; EN 301-893
<i>North America</i>	9.02-9.28, 5.85-5.925	FCC 47 CFR
<i>Japan</i>	7.15-7.25, 5.77-5.85	MIC EO Article 49

4.1.4. Network Security

According to the ISO/IEC standard, “the purpose of ISO/IEC 27033 is to provide detailed guidance on...security aspects of system networks and their inter-connections” [42]. ISO outlines several sections on maintaining a secure network over various architectures and scenarios. Since our project involves the use

of a wireless network to allow cars to communicate over a distance, we must make sure to follow these standards to ensure connection security.

The secure transfer of data across our network will be imperative for a safe, reliable, and effective self-driving system. The dangers of leaving our network open are far greater than an ordinary application over a home or standard public network. Table 20 outlines the 5 network security standards that will directly affect our network.

Table 20: ISO/IEC Network Security Standards

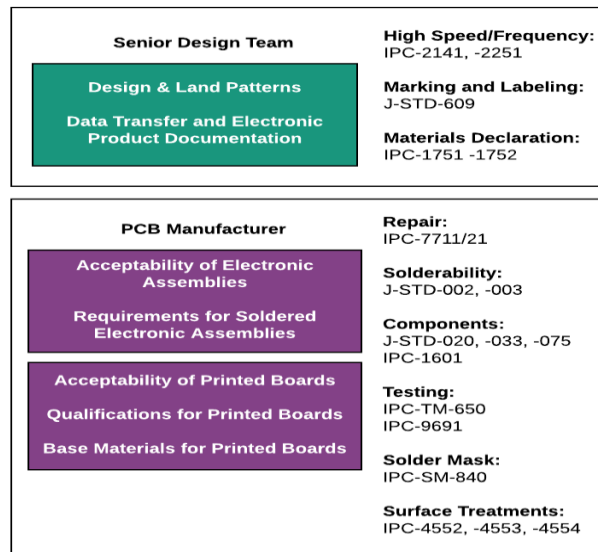
<i>ISO Code</i>	<i>Scope of Standard</i>
<i>ISO/IEC 27033-1:2015</i>	Network Security Overview and concepts: Provides a roadmap and overview of the concepts and management guidance for network security.
<i>ISO/IEC 27033-2:2012</i>	Guidelines for the design and implementation of network security: defines how organizations should...plan, design, implement, and document network security.
<i>ISO/IEC 27033-3:2010</i>	Reference networking scenarios – threats, design techniques, and control issues: discusses specific threats associated with typical network scenarios.
<i>ISO/IEC 27033-5:2013</i>	Securing communications across networks using Virtual Private Networks (VPNs): provides guidelines on selection, implementation, and monitoring of network security using VPN connections.
<i>ISO/IEC 27033-6:2016</i>	Securing wireless IP network access: define specific risks, design techniques, and control issues, providing basic advice for Wi-Fi, Bluetooth, 3G, and other wireless networks.

4.1.5. Road Safety Standards

Road safety can be divided into active and passive features. Passive safety features are only applied in response to a collision. The purpose of these are to ensure the safety of drivers and passengers in the case of collisions. Active safety features are deployed to avoid collisions, and continuously operate during the voyage to ensure the reduction in probability of an accident. As such, the fundamental components of the Self-Driving Car: Lidar, mapping, routing, classify this project as an Active Safety product [43].

Passive features include seat belts, air bags, head rests, laminated glass, correctly positioned fuel tanks, fuel pump kill switches, and a passenger safety cell. Air bags are currently only

regulated by NHTSA to be included for front passengers. Smart airbags further increase safety by measuring the passenger’s weight and deploying in a way specific to those measurements. Vehicle crashworthiness is a regulation whose inception traces to the late 1960s; presently all vehicles must pass crash tests before being marketed to the public. The law surrounding passive protection has a long and conflicted past, including several eras of enacting and repealing laws mandating seat belts in future car models. By 1998, all car models were to have passive safety features in them. However it wasn’t until 2006 that these features were further defined to be child-safe [44].



Active features provide a layer of protection that cannot be granted with passive features alone. Classical examples of such systems include Antilock Braking System (ABS), Tire Pressure Monitoring System (TPMS), Electronic Stability Control (ESC), traction control, collision-detection-and-avoidance, cruise control, and more recently, various levels of autonomy and auxiliary sensors. IPC PCB Standards Considering the most important feature of this project in terms of electrical design is the PCB, our board must conform to the relevant standards. IPC provides a list of standards for

Figure 15- PCB Standards

the whole process of manufacturing, testing, and distributing printed boards. Considering we will be designing our own board and utilizing a producer to build our board for us, the relevant IPC standards are IPC-2220 series + 7351 (Design & Land Patterns), and potentially IPC-A-600 (Acceptability of Printed Boards) [45].

The utility of conforming to these IPC standards are increased control over the quality and reliability of the PCB. The benefit of ordering the PCB from a manufacturer who conforms to the IPC manufacturing standards is ensured reliability, as well as an affordable board. Figure 15 indicates which IPC standards are to be fulfilled, and by whom.

Figure 15- PCB Standards

4.1.6. Inertial Measurement Unit (IMU) Standard

IEEE P1780 – Standard for the Specification of Inertial Measurement Units (IMU) provides specifications, units, format, and terminology for manufacturers and users for IMUs. This allows us to select an IMU which fits the specification of our project, namely, cost-effective, accurate, and reliable data [46].

4.1.7. AI and Self-Driving Car Standards

NHTSA (U.S. National Highway Traffic Safety Administration) lists five “eras of safety” on their automated vehicles website, consisting of safety and convenience features, advanced safety features, and advanced driver assistance features [47]. According to NHTSA, we are currently in the partially automated safety era, and by 2025 on, we are projected to implement fully automated safety features such as “highway autopilot.” This is one of many predictions, however this is a period in history where no such definite regulations or standards exist yet. NHTSA advises the states to let the DOT alone to regulate these technologies, however many (source) states are implementing policies on their own.

A recent law in Germany legalized autonomous cars such that the auto industry in Germany can adapt to the changing market [48]. This makes apparent the fact that this field is still developing and thus, a clear set of standards to follow has yet to be developed internationally.

4.1.8. Programming Standards

Adopting a standard programming style allows a team to create code that is modular, adaptable to the project, and interoperable with other parts of the project. Thus, the subsequent programming style guides outline recommended practices that will be employed in a collective “Programming Standard” for our team. Using this, implementing the software architecture will be efficient and effective.

4.1.8.1. C/C++ Standard

The C standard and the C++99 [49] standard together provide the definitive syntax and behavior of C and C++ code, which will be used to integrate sensor information, simulation and visualization of sensor data, and create real-time route decisions and networked behavior.

In addition to the C/C++ standards, which define how the languages work, the ROS Cpp style guide [50] will be employed due to the reliance on ROS for inter-process communication. This style guide defines such nuances as formatting, variable and function names, and other language-specific details that will further streamline development of the software design.

4.1.8.2. Python Style Guide

Python has a standard defined as PEP 8 [51], and ROS Py [52] defines a style guide which will both be followed where Python code is necessary. Since ROS allows C++ and Python code to work with the same resources, it is preferable that code be designed in such a way that neither Python nor C/C++ style guides nor standards be violated.

4.2. Constraints

In this section we discuss general constraints that can apply to any engineering project. The constraints will be described in reference to our specific project, the small-scale model for a self-driving vehicle and/or the eventual goal of creating a full scale self-driving car wherever applicable.

4.2.1. Cost Constraints

Our biggest constraint in our project is arguably the cost constraint imposed by our sponsor. While our project is sponsored and has received funding, numerous pieces of our project are incredibly expensive (e.g. One NVIDIA board alone is approximately twelve percent of our total budget for the project). This has led to us having to purchase cheaper sensors with less dynamic features and lower accuracy, especially in the case of our LiDAR sensor. Our sponsor also wants multiple of these cars constructed, so any design changes that are necessary to one are very likely to be needed on the others. This makes our project's budget extremely sensitive to change.

4.2.2. Environmental Constraints

While our project might help self-driving cars achieve more efficient routing and reduce pollution from associated greenhouse emissions, there are no environmental constraints on our specific project. In the future, the research lab may integrate solar cells to charge the batteries for our vehicle, but this is unlikely due to the additional cost because priority of upgrading sensors is much higher.

4.2.3. Social Constraints

While our project does not have social constraints, self-driving vehicles in the commercial market do. In a May 2016 survey done by AAA, it was found that 75% of respondents feared using self-driving vehicles if they hadn't used cars with semi-autonomous features like adaptive cruise control. Getting the public to release these fears will likely be a long-term side effect of our project, but in the interim, it remains a massive constraint to profitability for any manufacturer or major player in the industry. One of the primary reasons for these fears is covered in the Ethical Constraints section.

4.2.4. Political Constraints

Once again, our project is not specifically affected by public policy, but the self-driving vehicles that our project wishes to

emulate are. Only about 35 cities worldwide actually have self-driving cars being tested on their roads, and less than 20 more are considering allowing them to drive on their roads for testing. The main reasons for this are lack of regulatory oversight in regards to self-driving vehicles, lack of human and financial resources to appropriately manage a project, and interference from state or federal governments. However, Congress is currently considering passing a new set of rules regarding self-driving cars to provide a consistent set of standards for testing.

Another political constraint that might be overlooked is lobbying and the effects of that self-driving vehicles can have on certain industries. Right now, the U.S employs more than 3.5 million truckers. Eventually, these trucks will also become self-driving, and this could lead to massive dis-employment of a fairly large staple of the working middle class. This creates an incentive for truckers and their unions to want to limit this technology to protect themselves. Depending on how many lose their jobs, unemployment could also spike to dangerous levels and create political and economic instability in a worst case scenario. Public policy is going to need to play a very big role in stabilizing and re-tooling these displaced workers.

4.2.5. Ethical Constraints

While the primary purpose of a self-driving vehicle is to reduce accidents, there may be a time where environments may create a situation where an accident is unavoidable. The biggest ethical concern with self-driving cars is determining what action the car should take in that scenario. Should the car prioritize the life and safety of the driver, possible passengers, other drivers, property, or pedestrians in an emergency situation? This is an area where lawmakers may want to get involved to help provide a consistent ethical standard and also prevent lawsuits from victims or their family in accidents involving cars making these sort of decisions.

4.2.6. Health and Safety Constraints

The primary purpose of our project is to improve public safety, thus making this one of, if not the most important design constraint. Cars moving on a highway are moving likely moving close to 70 miles an hour, making latency a huge safety issue. A

latency of 1 second has the potential to cause catastrophes at these speeds. If our project is to model a car as realistically as possible we must minimize our latency.

Another concern related to sensors are the conditions a self-driving vehicle is present in. Weather conditions such as rain, temperature, snow, or humidity can affect sensor range and accuracy. Our project will mainly be tested indoors, but eventually our sponsor will want to collect more real world data in real conditions. If our vehicle design is used, additional sensors for measuring weather conditions and temperature and the incorporation of these measurements into our software will likely be required to maintain the accuracy of all the original sensors.

The strength of our network security. The possibility of someone being able to hack a car by accessing its controls via Wi-Fi is a massive concern. Malicious hacking could lead to forced crashes to hurt people, or routing vehicles into dangerous places where the passengers or cargo could be taken hostage. It's very unlikely that this iteration of the project will focus heavily on security, but improvements upon security will definitely need to be taken to create a better prototype for a full-sized vehicle.

4.2.7. Manufacturability Constraints

Our manufacturing constraints are mainly related to both accounting and time costs. The need to modify our vehicles to improve the specifications increases our time cost by forcing us to calculate what parts need to be changed, how or what should replace them, waiting on the arrival or construction of the new parts, and then physically deconstructing our vehicle and making the modifications. Accounting costs are accrued when we order said new parts or manufacture them via 3D printing. Both these costs will apply mainly to obtaining and modifying Plexiglas or aluminum for our mounts, unless the UCF innovation lab can provide us with free laser cutting with a very short turnaround time.

4.2.8. Testing Constraints

The main testing constraints will be related to how early we can modify the car and calibrate our sensors. We will have until the

senior design showcase to do this, but calibrating multiple IR, Ultrasonic, and LiDAR along with successfully changing out our vehicles gears and differentials will prove to be a time-consuming task.

Another disadvantage we have is that we cannot test our wireless communication network until we have at least 2 of our cars fully modified with the appropriate sensors and vehicle parts. We will very likely not be ordering more parts to create more vehicles until we can confirm our prototype works. The wait on the parts can vary from a few days to a few weeks, which creates some artificial deadline date weeks before the showcase so we can properly modify our cars and test the wireless communication protocol in the desired testing environment and make tweaks as necessary.

While not as significant as the other constraints, the testing environment provides another constraint. The floor of our sponsor's lab is carpet which possesses a much lower coefficient of friction compared to paved roads. This will require additional modifications to our car to ensure the wheels do not slip as a result of the lack of traction. While slippage might not cause physical damage to the car in most scenarios, it can distort the accuracy of the car's localization algorithm. Testing in an outside environment can be done, but the conditions are much less controlled and outsiders can possibly interfere. The high price of a single vehicle makes this very unlikely due to the risk involved.

4.2.9. Time Constraints

The next most important constraint next to cost is time. At numerous points in the other constraints we have mentioned that time is lost due to having to deal with other constraints on our design or to meet certain requirements. Unlike money, we cannot obtain more time to work on this project and still complete the project due to the design showcase having a fixed date. This makes time our most important resource. To help maximize the projects efficiency, 2 other undergraduate engineers, Yannick Roberts and Billy Blanchard, to help with the implementation of our project's software. Furthermore we have two graduate students, Nitish Gupta and Behrad Tohgi, providing us with guidance in relation to the localization, routing, sensor

calibration, vehicle dynamics, Robot Operating System (ROS), and overall design.

5. Project Design Details

In this section, we provide the details behind the design of our software and hardware and how we integrated different pieces of hardware and software together to create a working prototype.

5.1. Hardware Design

This section's focus specifically focuses on our hardware connections of our design. Responsibilities for the preliminary hardware interface is described in Figure 16 below. Each engineer on the team was assigned to research and configure the necessary hardware and learn the relevant software, if any,

to configure it. The Figure 17 below shows the electrical connections of the project and is explained in detail.

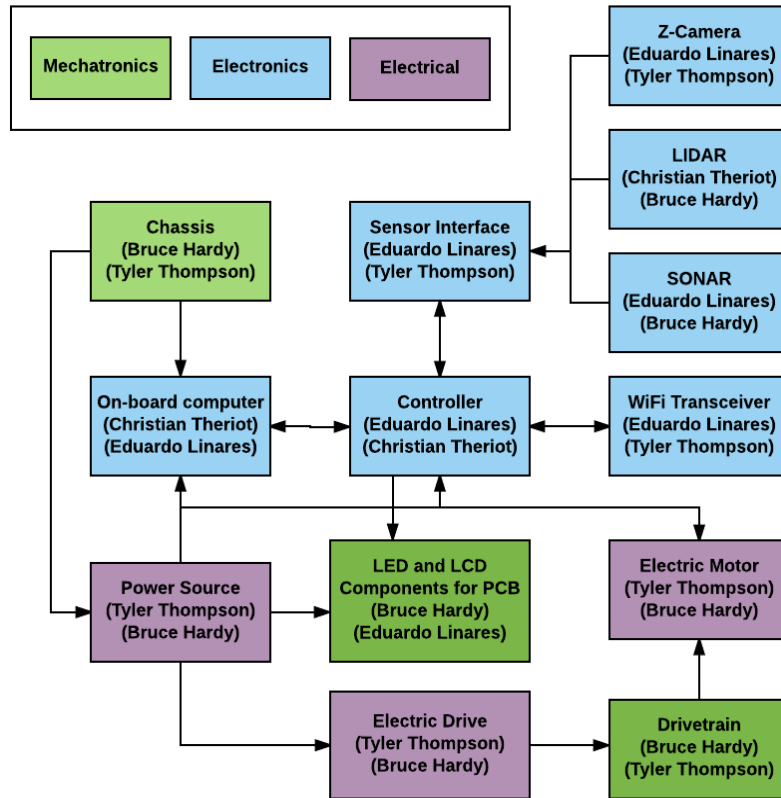


Figure 16- Hardware and Vehicle Interfacing

The MakOak battery bank supplies the USB Hub and the NVIDIA board with necessary voltage and current requirements specified by the manufacturers. The battery bank has a 20V 3A output for laptops that will connect to our NVIDIA CPU and another 12V 2.5A power output that will be used by our USB Hub. The Traxxas battery pack will only power the driving and servo motors. The NVIDIA board comes with a 19V 4.74A power supply for a total of 90W however, the acceptable power range is +9 to +15 V at around 60W when connecting USB devices. The USB Hub comes with a 12V 4A power adapter, the USB charging ports output 5V and 2.4A, and the USB 2.0 data ports are intended for data transfer to the host however if a component has low current draw they will be able to use these ports for power and data transfer. The Traxxas battery has voltage of 7.2VDC and a capacity of 5500mAh to supply the motor, these values will be

important for measuring the charge on the battery on our PCB design.

The Scanse LiDAR sensor requires 5VDC and draws a maximum current of 650mA and the Zed Stereo Camera also uses 5VDC and has a current draw of 380mA. From these values outlined by the component's technical specifications both of these sensors are compatible with the USB Hub's data ports for both power and data sharing to the NVIDIA board. The Arduino board requires a customized USB to barrel power connector to supply the controller with suitable power that will be connected to the USB Hub for power. An additional micro USB to USB-A cable will be used to transmit data between the Arduino and NVIDIA boards. The PCB was designed to use a 5VDC USB power cable and a current draw of less than 1A.

The USB Hub has multiple USB 3.0 data ports. These will communicate to the NVIDIA board data from our LiDAR, and Stereo Camera for visualization and localization purposes. Ultrasonic and Hall sensors will utilize a fraction of the 40 GPIO on the NVIDIA to be stored and used for localization data. The USB Hub relays data from the NVIDIA computer to the PCB board to determine drive states and other localization data for display. The display will be achieved through LED lights to simulate front rear and direction status in an effort to simulate standards for road vehicles.

The Traxxas battery will be connected through a conversion circuit to the PCB board for battery level management. The hopes are to manage battery levels and eventually have the robot making driving decisions in regards to the battery level data. The LCD will display the battery level as well as data supplied from the NVIDIA for heading, average speed, and errors in the programming. The USB Network adapter will connect directly to the NVIDIA board to deliver the strongest Internet connectivity.

The Arduino Uno board will connect to the servo motor for steering and the driving motor control unit through 3-pin DuPont connectors that contain data, power and ground wires. The Hall sensor, used to determine the rpm of the motor for localization data will be read by the Arduino board to adjust speed and heading through a similar connector. Data and directions will be

transmitted to the other vehicles through the Internet connection and DSRC control that utilizes one of the USB ports on the NVIDIA CPU. The car will also have a second control mode using a host computer that is connected to the same wireless network from a distance by writing codes to the NVIDIA control unit.

The flowchart below in Figure 17 shows the hardwired connections between components in our design for power and data transmission. Customized cables will be made using DuPont, USB 2.0 and USB 3.0 solder-type connectors and cables. All cables had to be purchased carefully to adhere to voltage and current requirements of the attached components. After the testing is complete all wires will be covered by heat shrink tubing for protection and to improve the overall aesthetics of the vehicle for the design showcase.

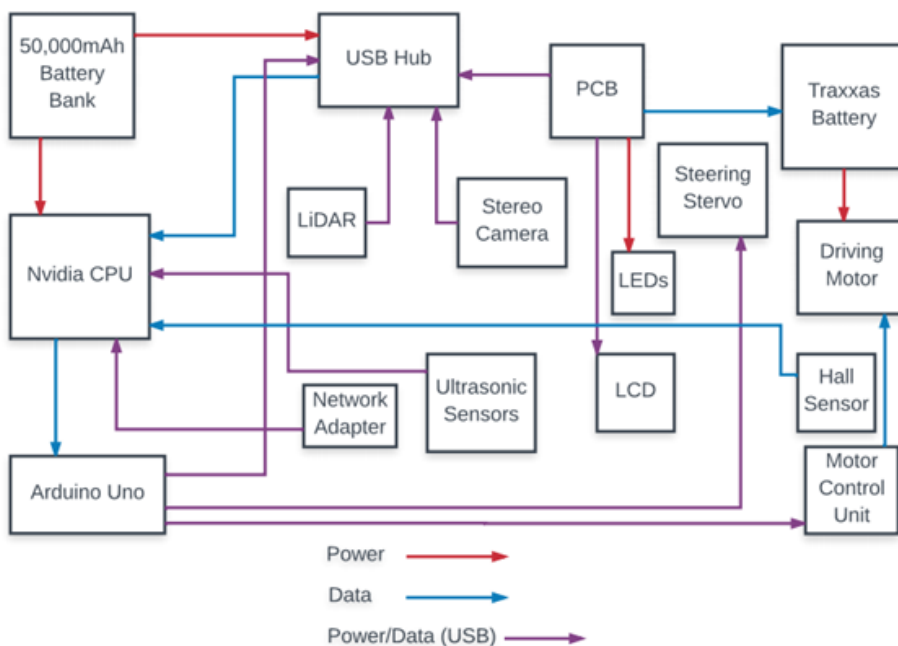


Figure 17- Wiring Diagram

5.1.1. Sensors and Calibration

Due to the nature of our project, sensor accuracy is a priority for optimizing the accuracy of our model. To counter the error initially present in our sensors, we tested the initial error and devised a testing scheme for each type of sensor.

5.1.1.1. Ultrasonic Sensor

Our tests were conducted on a small using an Arduino Uno, a dictionary, the HC-SR04, and a ruler. The ruler was placed in a straight line away from the center of sensor the transmitter. The book was shifted back by one centimeter every measurement, and the data would be sent to the Arduino and ROS. From there we could determine what the error between the position that the sensor measured and actual position.

This setup is extremely simplistic, and also has faults as a result. Slightly pivoting the book could provide massive error because of the angle that the sound waves bounce back towards the receiver. Slight shifts of the ruler would put the surface off center and introduce small amounts of error. We could also only measure a maximum of 30 centimeters accurately due to the length of the ruler, but in order to prevent the ruler from moving, we placed objects behind it to fasten it into place. Pre-calibration results can be seen in Table 21.

Table 21: Pre-Calibration Data for Sensor 1

Distance (cm)	Mean (cm)	% Error	Absolute Error (cm)
3	4.15	38.33	1.15
4	5.364	34.1	1.364
5	6.506	30.12	1.506
6	7.857	30.95	1.857
7	8.452	20.74	1.452
8	10.41	30.12	2.41
9	12.09	34.33	3.09
10	12.84	28.4	2.84
11	14.74	34	3.74
12	16	33.33	4
13	17.2	32.31	4.2
14	20.39	45.64	6.39
15	21.609	44.06	6.609
16	23.45	46.56	7.45
17	24.22	42.47	7.22
18	49.4	174.44	31.4
19	50.69	166.79	31.69
20	28.3	41.5	8.3

We decided to only calibrate for the first sensor, and then use that calibration for sensor 2 to check if it would create consistent results across sensors. If it did, we would test our calibration on the next two sensors when we received the opportunity and

continue moving forward. Otherwise, we would repeat this test again for the other 3 sensors.

5.1.1.2. Calibration and Code

ROS provides a pub-sub (Publisher and Subscriber) framework to send the ultrasonic data recorded by the sensor and capture that data via a USB connection to the UNO board. The standard Arduino ultrasonic code was modified to publish the distance data as a float over the "/ultrasound" topic.

From here, the ideal approach would be to create a subscriber which would receive the published values and would be able to process those values accordingly – finding average mean, maxima, and minima. However, there were issues creating such a program, thus a simple data scrubber was written in C++. ROS allows you to display the data being transferred on a topic, thus the scrubber would take this data and filter out the "range: [float]" value semi-automatically. Thus, the operator would have to have two terminals, one running "rostopic echo /ultrasound/range" and another running the scrubber. They would then have to copy the data from the ROSTopic terminal into the scrubber, which would then display the minimum, average mean, and maximum of the data.

It was noted at this point that temperature affects the speed of sound, which would significantly affect recordings. Thus, a calculation correction was applied to the code, and all subsequent calculations included this correction. This data was then recorded in a spreadsheet containing the actual distance, the mean recorded distance, the difference between these two values, and the average error. Distance was varied between 3 cm and 23 cm, taking intermediate recordings (for example 8.5cm and 10.5cm) when the change in error increased significantly after increasing distance. Average error ranged between 20.7% (at 7 cm) up to 174% (at 18 cm) and back to 43% (at 23 cm).

From this data, lookup tables were created as float arrays in the sensor code. The distance was corrected by dividing it by $(1 + \text{error})$, error being one of the values of average error previously mentioned. It was originally intended to apply the appropriate

error according to the lookup table, however it was noticed that the code effectively only used the first or second recording (i.e. for 3 cm or 4cm, respectively) at every distance.

For the second sensor, the code was modified to present the distance before the error adjustment in order to find its error measurements for 3cm. This error was already slightly lower than the previous sensor, however applying the error adjustment to the code allowed the second sensor to be as accurate as the first sensor.

5.1.1.3. Testing Post-Calibration

After recalibrating using our code, we plotted the absolute and percent error for each sensor. Absolute error peaks less than 1.5 centimeters for each sensor, and percent error below 10% for each sensor. The results are displayed in Figures 18 and 19.

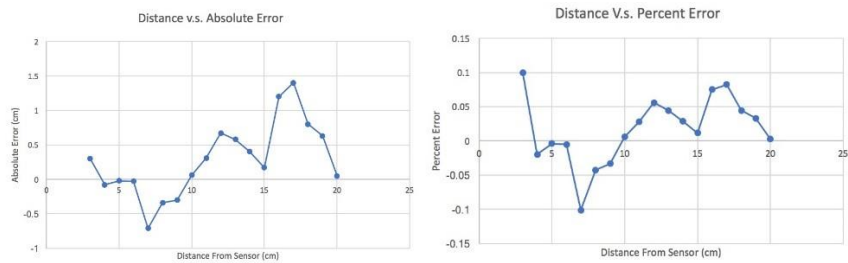


Figure 18- Sensor 1 Test Results

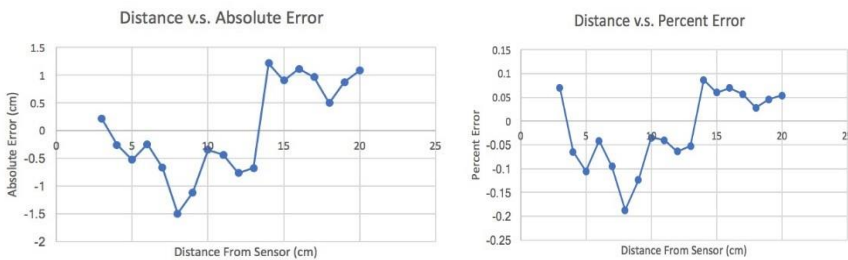


Figure 19- Sensor 2 Results

5.1.2. Infrared Sensor Calibration

Unlike the HC-SR04 Ultrasonic Sonar, the DOAKI infrared sensors are digital and can only tell the use if they detect an object or if they don't detect an object. No distance data can be

drawn from the sensor. The sensor, can however, have its range adjusted by using a screwdriver to adjust the settings on the sensor's potentiometer. Unfortunately, the sensor is also prone to becoming oversensitive and declaring that there is always an object in the path even when there is nothing in the sensor's maximum theoretical range.

After carefully tuning the potentiometer for one of our sensors, we found it had a maximum range of 5 centimeters before it became oversensitive and always return a high signal even if there was no object in its range. We repeated this same process on another sensor and received a maximum range of 9 centimeters. Due to the unsatisfactory performance relative to what was advertised, we will not be incorporating these into our design.

5.1.3. GPIO Configuration

Our original plan was to use the J21 [88] GPIO pins on the Jetson TX2 for serial communication with the arduino, the ultrasonic sensors, and the hall RPM sensors. However, it was determined that many of these pins are used elsewhere in the Jetson for processes such as video and audio interrupts. Thus, preliminarily we decided to connect the arduino to the Jetson via the USB serial hub and free up the remaining five usable GPIO pins for ultrasonic and hall sensors.

Pictured below in Figure 20, four available pins are 29, 31, 33, and 37. Two of these will be used for all ultrasonic sensors, limiting the ability to detect which sensor is detecting collision.

	SPI_CLK <i>SPI #1 Shift Clock</i>	23	24	SPI1_CS0# <i>SPI #1 Chip Select #0</i>
	GND	25	26	SPI1_CS1# <i>SPI #1 Chip Select #1</i>
	I2C_GP1_DAT <i>General I2C #1 Data (3.3V)</i>	27	28	I2C_GP1_CLK <i>General I2C #1 Clock (3.3V)</i>
gpio398	GPIO19_AUD_RST <i>Audio Reset (1.8/3.3V)</i>	29	30	GND
gpio298	GPIO9_MOTION_INT <i>Motion Interrupt (3.3V)</i>	31	32	AO_DMIC_IN_CLK <i>Unused</i>
gpio389	GPIO11_AP_WAKE_BT <i>AP Wake Bt GPIO</i>	33	34	GND
	I2S0_LRCLK <i>AUDIO I2S #0 Left/Right Clock</i>	35	36	UART0_CTS# <i>UART #0 Clear to Send</i>
gpio388	GPIO8_ALS_PROX_INT <i>(3.3V)</i>	37	38	I2S0_SDIN <i>Audio I2S #0 Data in</i>
	GND	39	40	I2S0_SDOUT <i>Audio I2S #0 Data in</i>

Figure 20- JetsonTX2 J21 Header Pinout

In the future, it would prove helpful to be able to disable these pins for their original purposes in order to use them for our needs. We are unable to do this at this time because the documentation for the TX2 is still very sparse and not as detailed as that for the TX1 or the TK1. Also, the TXx series has less available pins in general than the TK1, thus it might be useful to downgrade to the TK1 in the future.

5.1.4. Serial Communication

The Jetson communicates to the Arduino via the serial-USB connection. In order to test this connection, a C++ program was written on the Jetson to send bytes to the Arduino via the `/dev/ttyACM0` device in Linux. The Arduino was then set to apply speed to the motor to show the successful connection and reception of data from the Jetson.

This demonstrated that moving the serial communication from the GPIO pins was a viable option and thus we decided to establish a protocol for controlling the Arduino via the Jetson. Two bytes would be used to determine the motor speed and servo angle. These bytes would be converted to their integer values and then their respective servos would be set to these integer values. The Arduino would then await the next two bytes from the Jetson.

Thus, the Arduino becomes a mere actuator, which applies the values desired by the Jetson. Therefore all the planning, routing, speed, and angle calculations would be handled by the Jetson, as opposed to leaving speed and angle calculations to the Arduino. This protocol would allow faster communication to the Arduino, free up any latency on the USB hub, and

5.2. Software Design

Figure 21 outlines the general software flow from initialization until power off. The logic is simple and generic at this stage, partly due to being the first software drawing, and partly as a way to show the overall structure of the software. The green boxes represent end and terminal states, the blue boxes represent a process or method, and the orange parallelograms represent transitions between states. The specifics of localization, object detection, and software architecture will be detailed below.

Figure 22 gives insight into how the hardware is modelled in software. The actual software relationships will be displayed in Figure 23. The actual class diagram as a result of using ROS is much simpler due to the specifics of multiprocessing and networking being taken care of by open-source algorithms.

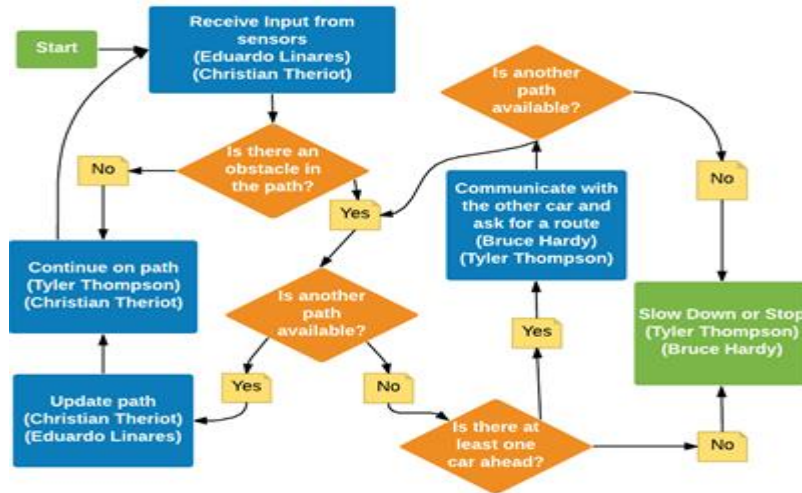


Figure 21- Software Flow Diagram

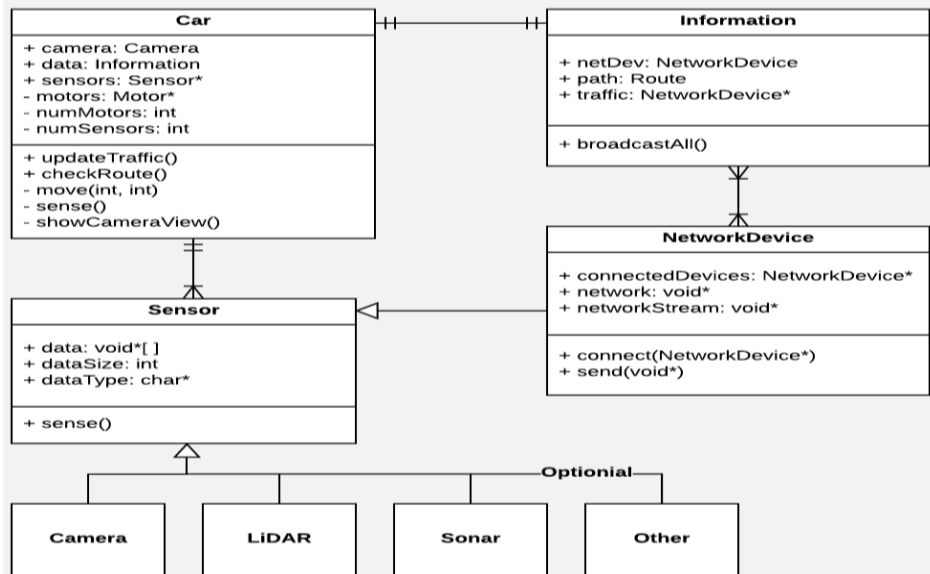


Figure 22- Software Class Diagram

- A double-bar indicates only one instance of the connected class is present.
- The triangle with one line over it indicates at least one instance is present.
- The white triangle points to the base class for inheritance.

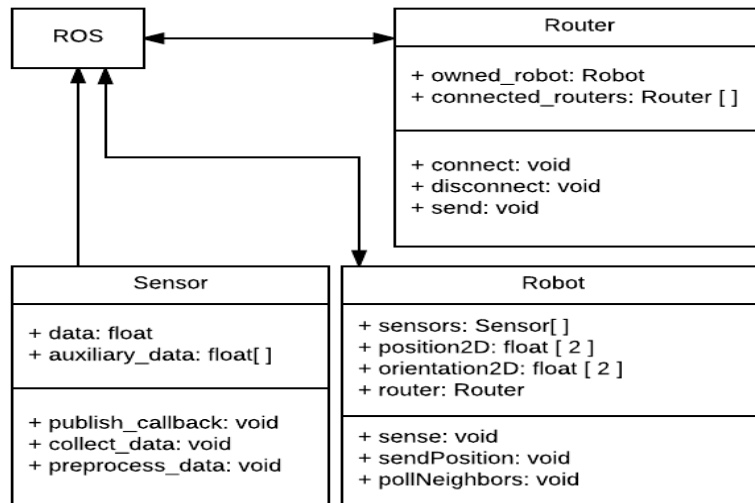


Figure 23- Updated Software Diagram

Since ROS simplifies the transmission of data within a robot, the process of manipulating the Wi-Fi connection between robots and the process of transmitting data between robots is also simplified. Therefore, the "Information" class is redundant considering each class or program can store its own data variables and transmit them appropriately.

Each class in Figure 22 includes its own version of "publish" and/or "subscribe," which is critical to linking it to ROS and by extension, to other classes. Data is simplified from a class down to the relevant data types for the class. For example, the Sensor only requires a float value (for ultrasonic and distance recordings), while the Router requires the robot it belongs to, and other visible Routers.

The starkest update from integrating the software with ROS is that the relationships between classes aren't handled explicitly by linking the two classes, rather the relationships are resolved within the operating system itself. For example, even though Router includes the robot it belongs to, this would be implemented by using some unique identifier in the Robot class, which is passed to its own router via a pub-sub (shorthand for publish/subscribe).

5.2.1. Localization Algorithms

In order to produce a self-driving vehicle with mapping capabilities, the use of localization algorithms became a necessity. Numerous algorithms exist, and this section will detail which algorithms we will implement in and why. Considerations for each algorithm were complexity, simplicity to program, exact function of the program, and if different algorithms performing the same function can compensate for inaccuracies in another under different scenarios.

5.2.1.1. Kalman Filter

The Kalman filter is a set of equations that provides an efficient recursive estimate to what state a process is in while providing a minimal mean squared error even with sensor error and external influences affecting the system. The filter is also capable of processing future states even when the modeled system's functions are not necessarily known. Kalman Filters are also used in a variety of fields unrelated to vehicle localization, like statistical analysis and econometrics.

The Kalman filter in a self-driving car will measure the position and the velocity of a car. Updates to the equations, which will we go into detail later, will be made during each step shown in Figure 24.

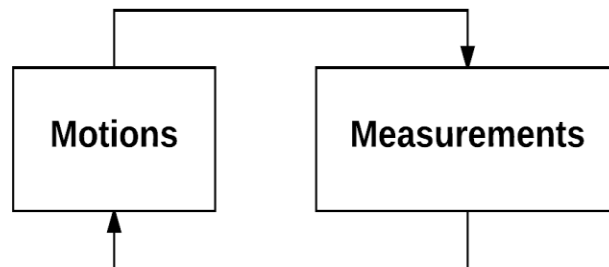


Figure 24- Steps during a Kalman filter's measurements

During the measurement steps, the Kalman filter only cares about 2 things: The new measurements and previous belief of where the vehicle is located. During the motions step (sometimes called predictions step because velocity is measured through changes in distance), only the velocity inferred from previous measurements and the previous beliefs are necessary. This

keeps the computational power of a Kalman filter extremely low relative to other algorithms, making it extremely advantageous for localization.

5.2.1.1.1. Probabilistic Origins of Kalman Filter

The Kalman Filter's origins lie in probabilistic theory. It is assumed that the variables that comprise the system's state are Gaussian distributions. A Gaussian distribution is a probabilistic distribution that is unimodal, centered on a mean value that represents the most certain point, and have a variance that represents how uncertain the mean value is. Gaussian distributions are defined by the equation shown below. The shape of the Gaussian distribution is determined by the exponential and the term outside the exponential normalizes the values produced by the equation.

$$F(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Equation 4. Gaussian Distribution

Two examples of Gaussian distributions centered at zero with different variances are shown in Figure 25. The red Gaussian distribution has a higher variance, and has more uncertainty of where the real value is as a result. An ideal Gaussian distribution has a variance as low as possible around the mean value.

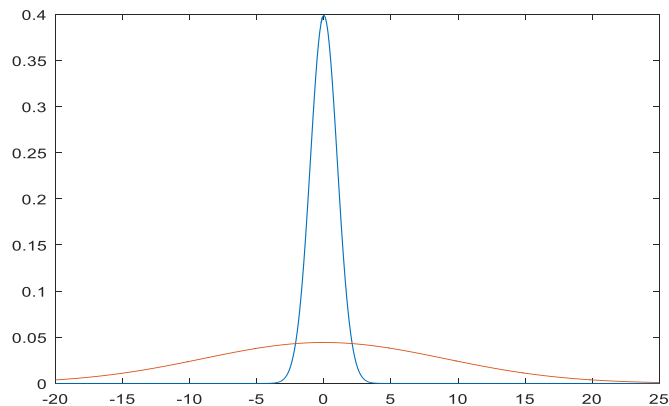


Figure 25- Example Gaussian Distributions

When a Kalman Filter receives new measurements, it simulates them as Gaussian distributions as well. The ensuing Gaussian will always have a lower variance than either the prior distribution or the new measurement. The new distribution will also have a different mean value. The calculations for the new variance and new mean are shown below in Equation 6. An example of the new Gaussian is shown in Figure 26.

$$\mu_{new} = \frac{\sigma_1^2 \mu_2 + \sigma_2^2 \mu_1}{\sigma_1^2 + \sigma_2^2}$$

$$\sigma_{new}^2 = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}$$

Equation 5. New Gaussian Distribution

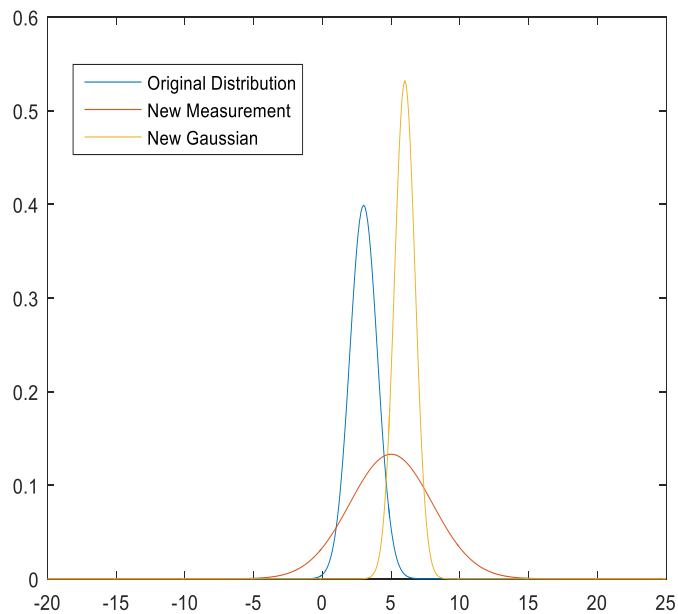


Figure 26- Gaussian Created by other Gaussians

By taking these different Gaussian measurements, the Gaussian becomes less uncertain. Note that this only applies to measurements and that the examples given are for a one dimensional system. A multidimensional Gaussian will have a D

x 1 matrix of means and a D x D matrix of variances will be defined by a contour, as shown in Figure 27.

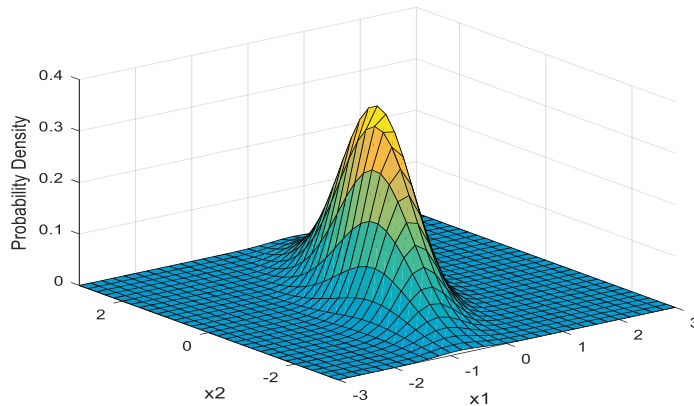


Figure 27- 2D Gaussian Distribution

In the case of a self-driving vehicle, motions will create uncertainty in position and increase the mean value of the Gaussian distribution. Motions are also modeled as Gaussian distributions, and the resulting Gaussian distribution after a movement can be summarized by Equation 6.

$$\mu_{new} = \mu_{old} + \mu_{motion}$$

$$\sigma_{new} = \sigma_{old} + \sigma_{motion}$$

Equation 6. Gaussian Movement Update

5.2.1.1.2. Implementation of Kalman Filter

In the real world, two forms of the Kalman Filter can be created for localization: the Unscented Kalman Filter (UKF) and the Extended Kalman Filter (EKF). The reason the linear transformation cannot be used is because the algorithm's will create small errors that will slowly accumulate and eventually become significant if the calculated covariance becomes too small, indicating a large certainty that the location is known, then these errors will become massive and lead to improper state estimation.

Among the two mentioned variants, the Unscented Kalman Filter has proven to be the best one because it maintains the same complexity as its counterpart, works for non-linear systems, and

approximates the true mean and covariance to the 3rd order Taylor Series approximation as opposed to the first order with the EKF. It does so by taking numerous sigma points

The Unscented Kalman Filter is based around the idea that it is easier to approximate a probability distribution than it is for a random non-linear function or transformation of a function.

5.2.1.2. Particle Filter

Another localization algorithm we will be using will be the particle filter. Particle filters use a set of guesses for position and orientation (“particles”) alongside measurements from sensors and odometer data to implicitly localize itself given a map of the location.

“Particles” are re-distributed across the environment after every motion. Sensor data is used to determine the location of landmarks. “Particles” that are more likely to represent the current state of the vehicle are then given a weight, with higher weights being more accurate states. The weights are then normalized so the sum of the weights is equal to 1, and the particles are resampled so the highest weighted ones have the most likely chance of recurring. Eventually, only the particles that represent the state closest to the true state of the vehicle.

The Particle Filter excels in areas the Unscented Kalman Filter does not. It works better than the Unscented Kalman Filter with non-Gaussian noise. It is a multimodal distribution that can keep track of where multiple objects are. Occlusions, or an object blocking or obstructing another, do not affect the Particle Filter, unlike the Unscented Kalman Filter. The Unscented Kalman Filter would have a corrupted entry for its measurement step, and in the absence of a way to detect an occlusion, would lead to a massive error.

The Particle Filter does have its disadvantages: mainly related to computing power and filter degeneracy. The amount of particles needed to measure a system increases exponentially with every dimension added, so computing power and memory need to increase to match these as well. If an observation model is too accurate e.g. high probability, but very low variance for a

measurement, the weights of the accurate particles can be reduced to near zero and then be removed during resampling. If the particles are resampled without new measurements, the particle filter can assume that vehicle is in a false location. An example of this is that two rooms can be identical to one another, but a particle filter can incorrectly make an assumption that the vehicle is in one room that it actually is not in. See Figure 28 for a visualization of this scenario.

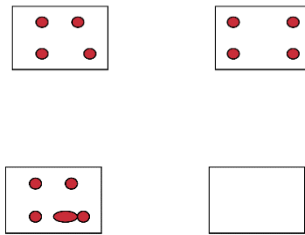


Figure 28- Before (Top) and After (Bottom) Filtering Without Motion

Despite these issues, we've decided to use the Particle Filter alongside the UKF due to its strengths. The Particle Filter is immune to errors that can occur in the Kalman Filter, and should one filter fail, the other could still be used for localization.

5.2.1.3. A* Search Algorithm

One of the most famous search algorithms is Dijkstra's shortest path algorithm, which was then expanded to create A*, another algorithm which will be employed in this project. Table 22 describes the steps of the Dijkstra's algorithm, which is used to find the shortest path between two nodes.

The purpose of using infinity as a tentative distance is to signify unvisited nodes, such that the algorithm will not visit them if there is another node with a better (smaller) distance between the initial and current node. Each subsequent node is updated with the sum of all distances of previous nodes. Also, tentative shortest paths might also be updated with smaller values, if a shorter path is found. Once the destination node is reached, the path can be traced back usually via backtracking algorithms or an array in a dynamic programming implementation.

The advantage of this algorithm is that it will *always* find the shortest path between two nodes, assuming such a path exists. The disadvantage of this algorithm is that it is inefficient given a sufficiently large topology. Since this project requires pathfinding in a real context, the topology can become large and nearly infinite, given the structure of roads and the fact that new roads can be added at any moment.

One way to make the algorithm more efficient would be to implement a priority queue such that the minimum-distance nodes are selected with less processing time. Also, the neighboring nodes would be added concurrently to checking which one is closest to the current node. Thus, instead of working with all of the possible nodes, only the most relevant ones are in the priority queue. This would be relatively more efficient than the standard algorithm, however it would still be wholly inefficient with an infinite graph.

For infinite graphs, a modified “uniform-cost” search algorithm is created when, instead of adding all nodes to the graph, nodes are only added when discovered to be in the shortest path. Thus, a path from the initial location to a set of target locations can be achieved on an infinite graph, otherwise it would be inconceivable with a limited memory constraint.

A* searches all possible paths for the one which incurs the smallest “cost,” which can be defined uniquely for the application, and considers the one that appear to most quickly lead to the solution. It selects which of its paths to expand towards the goal using a heuristic function which estimates the cost remaining to reach the goal. Thus the path which it will select will minimize this function, as that will lead to the shortest remaining path. The heuristic function takes into account the cost of the path from the beginning to the current node, added to the estimation of the remaining cost of the path to reach the goal.

The requirement of A* is that the heuristic is “admissible,” or that the cost to the nearest goal is never overestimated. A more efficient version requires the heuristic also be “monotone.” The function is monotone when the heuristic of one of the vertices is less than or equal to the distance of the edge plus the heuristic

of the other vertex. Thus, no negative costs are allowed and a vertex will never be considered more than once. However, setting the heuristic of all nodes to 0 would be a special case of A* which can also be viewed as Dijkstra's algorithm. Another special case would be a heuristic in which earlier nodes have higher values than later nodes; this would be a depth-first search.

A* will always optimally return the shortest path given if and only if it uses an admissible heuristic, and only if it considers one search problem (point A to point B rather than Point A to point C via point B). While an admissible heuristic always returns the shortest path, A* must consider all equally viable paths up to the terminal node. There is a method called "bounded relaxation" in which a less optimal variant of the algorithm is used to conserve

Table 22: Dijkstra's Algorithm Steps

<i>Step</i>	<i>Explanation</i>
1. Define Initial Node	Assign a tentative distance to every node in the graph (zero for the initial location, infinity everywhere else)
2. Define Unvisited Nodes	Set the initial node as current; mark all other nodes unvisited. Create a set of all unvisited nodes.
3. Update Neighboring Nodes	Consider all the neighbors of the current node. If the distance to the neighboring node is less than its tentative value, update it to the smaller distance.
4. Mark Current Node as Visited	Mark the current node as visited and remove it from the unvisited set.
5. Terminate for Final or Unreachable Node	If the destination node has been visited or the smallest distance between the two nodes is infinity (i.e. unreachable path), stop the algorithm.
6. Update the Current Node and Continue	Select the unvisited node with the smallest tentative distance as the current node, go back to step 3.

5.2.1.4. Simultaneous Localization and Mapping

Simultaneous Localization and Mapping, also known as SLAM, is a necessary component for the project. The Particle Filter only works if the vehicle has a map of the area it is travelling in. SLAM uses the measurements taken by sensors of landmarks and exact motions to create a map in real time. These distances are saved in an N x N matrix called Omega (ω), where N is the

number of motions and landmarks, and an $N \times 1$ vector called ξ that is updated with each movement. Using the inverse of the Omega matrix and the Xi matrix, the vehicle's heading and location alongside the location of the landmarks will be stored in a new matrix called μ . This is shown in Equation 7.

$$[\mu] = [\omega]^{-1}[\xi]$$

Equation 7. Graph SLAM Location Equation

The advantage of Graph SLAM is that even in cases of incorrect measurements of sensors, Equation 7 will produce the most accurate answer based on these measurements due to the constraints for the algorithm including movement as well.

5.2.2. ROS

ROS (Robot Operating System) provides "libraries and tools to [developers] create robot applications" [56]. ROS is a framework that runs on the Linux OS, providing several distributions developed specifically for each version of Linux. In our development system, we use Ubuntu Linux 14.04 LTS, and thus the appropriate ROS distribution is nicknamed "Indigo."

5.2.2.1. General Architecture

Included in ROS are pre-developed libraries to simulate robot configurations, environments, sensor data, as well as integrating real-world sensor data to visualization software to allow for development of a robot system. ROS implements a publisher-subscriber framework, meaning that there is a "master" thread which ensures that programs can publish data and subscribe to receive said data from "topics." As long as the ROS master program is running, programs can publish and subscribe as many topics as they need to accomplish their specific tasks.

Custom topics can also be created to send user-specified data to and from applications. This will prove useful for our purposes because instrumentation and data collection will be able to send and receive data on their own respective topics without interference from the other. This will ensure a streamlined flow of information from sensors, to a set of processing threads to capture the data, to another set of processing threads to localize,

plan, and route the vehicles, and even another thread to facilitate networked communication of this data.

As can be seen in Figure 29, the publish-subscribe concept is very simple yet powerful. The publish-subscribe architecture as well as the topic implementation allows for the modularization of the processing tasks that need to be created. Thus if an updated graphics card is to be purchased, the processing task is the only one which will have to be updated as a result.

The data collection, routing, and localizing tasks will be otherwise unaffected and will be able to continue running while the processing task is updated. The same is true if more sensors are to be added, or the type of sensor is to be modified. In this case, only the sensor task will have to be changed. This will reduce the amount of maintenance required for the project, and thus save time and energy which can be directed elsewhere.

Another result of the modularization of ROS is that heavy processing tasks can run in parallel with other heavy processing tasks (robot vision processing in parallel with a complex routing algorithm, for example) as well as networking algorithms. Data can be transferred between the cars while they continue to sense their surroundings. This is critical because a non-multiprocessing paradigm would mean that the car could potentially stall or otherwise be blinded in the middle of a networking event. Thus, the publish-subscribe framework ensures the safety and time-sensitivity constraints inherent in this project.

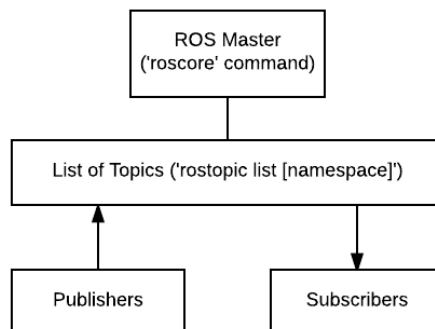


Figure 29- ROS Publish-Subscribe Framework

Publishers and subscribers are both implemented using callback functions which are referenced in the ROS-provided "publish"

and "subscribe" functions which are run in the main function (C++). This allows for user-defined code to be run every time data is able to be sent or received. ROS also allows for a stack of data to be accumulated between calls.

Two programs included in ROS that will be used extensively throughout this project in accordance with other ROS-defined and user-defined classes are 'Gazebo' and 'RVIZ.' Gazebo is a 3D robot simulator which includes physics simulation in a Linux environment. RVIZ provides a 3D environment in which to visualize sensor data that is being sent throughout several data topics. Therefore, a 3D model of the robot will be created in Gazebo, including sensors in their respective locations. Then, this will be sent through topics specific to gazebo and RVIZ, which will be visualized in RVIZ.

Simulating the robot(s) in gazebo and RVIZ allow us to develop the localizing, routing, mapping, and intercommunication algorithms with the full consequences of the algorithm, including collisions, losing the robot at a far distance, or any other anomaly. Once the features are fully developed and the anomalies aren't damaging the simulated robots, it can be ported to the physical hardware and tested in the lab. This saves money and time considering the hardware is preserved as much as possible, and the number of tests that can be run in a simulation are orders of magnitude greater than what can be run on a physical model.

Figure 30 shows the overall software architecture design for this project. It takes into account every aspect of the project, from the fact that the algorithms should work similarly for both simulated and real environments, as well as the different mapping techniques employed.

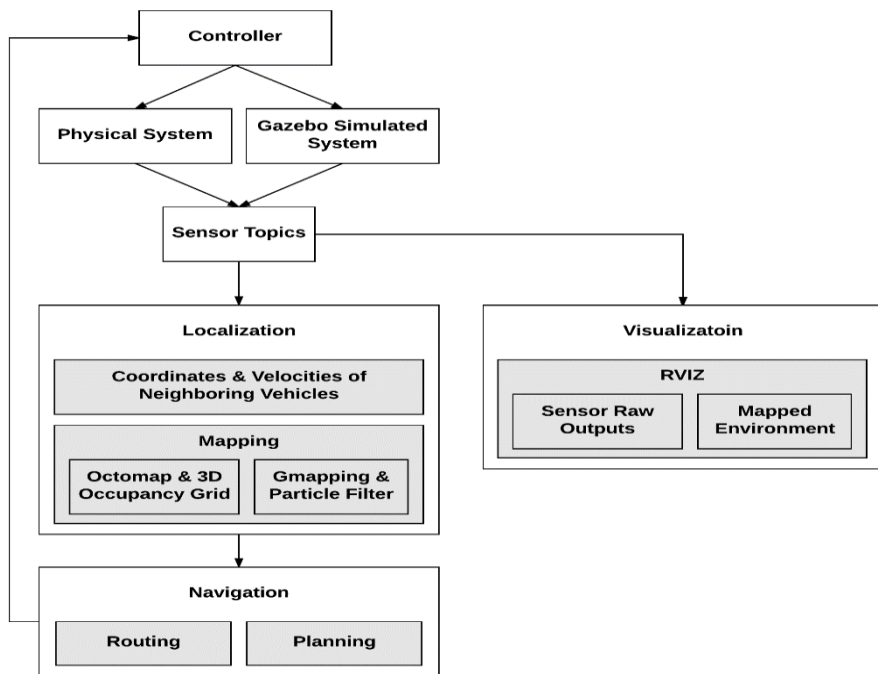


Figure 30- Diagram of Overall Software Architecture and Flow

Since ROS handles the intercommunication of these parts, it is helpful to conceptualize them as communicating directly with one another. As such, the sensor data is conceptualized into their respective sensor topics and sent simultaneously to the localization and visualization sections. There, processing is done to map and synchronize the network of vehicles together, as well as visualize this process on a universal access point, such as a computer on which RVIZ runs.

A navigation process then takes the localization information and computes the next steps to be taken by the vehicle, and sends it back to the controller to move, accelerate or decelerate, stop, or turn.

5.2.2.2. ROS Topics

ROS allows threads to talk to each other along what is called a “topic.” This includes the data that is sent as well as information about the types of data that is sent. There are topics for every part of the project including sensors, navigation, and visualization. Table 23 outlines some relevant message types

that will be used as well as the purpose and functionality of those messages.

Table 23: List of Message Types

Message Name	Relevant Component
sensor_msgs/Range	Ultrasonic, IR sensors; used for the range value which is the distance in cm between the sensor and the point detected.
sensor_msgs/LaserScan	LIDAR, Kinect sensors; used for an array of ranges between a minimum and maximum azimuth (min azimuth of 0 and max azimuth of 360 degrees for LIDAR).
sensor_msgs/Image	Z-Camera; used to store images, both for depth and a stream of actual camera data.
nav_msgs/Odometry	Position and Velocity in free space; used to keep track of the location and speed of the vehicle, both for itself and other vehicles in the network.
visualization_msgs/MarkerArray	Map visualization; used to display the 3D occupancy grid in RVIZ.

These topics are defined below in Figure 31, including the relevant variable definitions for each message. For Range, the radiation type is an enumerator which determines whether it is an Ultrasound or Infrared sensor. Any range that is outside of the bounds of min and max range are to be discarded. This is used to calibrate the ultrasonic sensor and for detecting distances using ultrasonic.

The Image message includes data specific to the camera which sends it, thus a normal RGB camera would provide pixels representing the image. However a depth camera's data field includes the depth of the image.

LaserScan uses the same basic algorithm of discarding any ranges outside of the bounds of min and max range. This is used for LIDAR scans, where angle_min and angle_max can be set to ignore any scans outside of the desired ranges. This can be used to only process scans in front of the vehicle, or coordinating other directed scans.

Odometry will be the main message that shares the position and velocity of the vehicle over the network. It consists of a "pose" and "twist" along with a covariance coefficient to be used with probabilistic algorithms. Pose consists of a builtin-type for position and rotation, while Twist contains velocity and angular velocity.

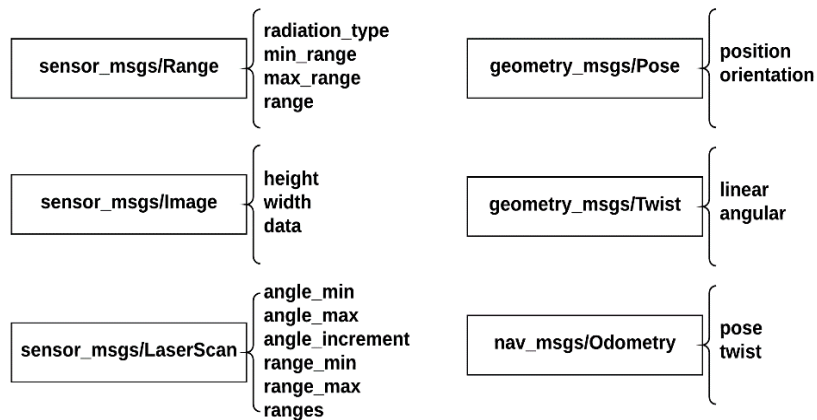


Figure 31- ROS Message Definitions

5.2.3. Mapping

Mapping is a technique used to keep track of the world around the vehicle. This map keeps track of information about the environment that can be used to localize the vehicle, keep track of obstacles and walls, or find unique landmarks. Maps can be used in conjunction with Particle Filters, Kalman Filters, or other techniques in order to accomplish these greater tasks with higher accuracy.

Two types of maps that are implemented in ROS are gmapping and octomaps. Figure 32 shows a test setup in gazebo using the built-in simulation named Turtlebot. This will be used later to compare the fidelity of the mapped environment with the real environment.

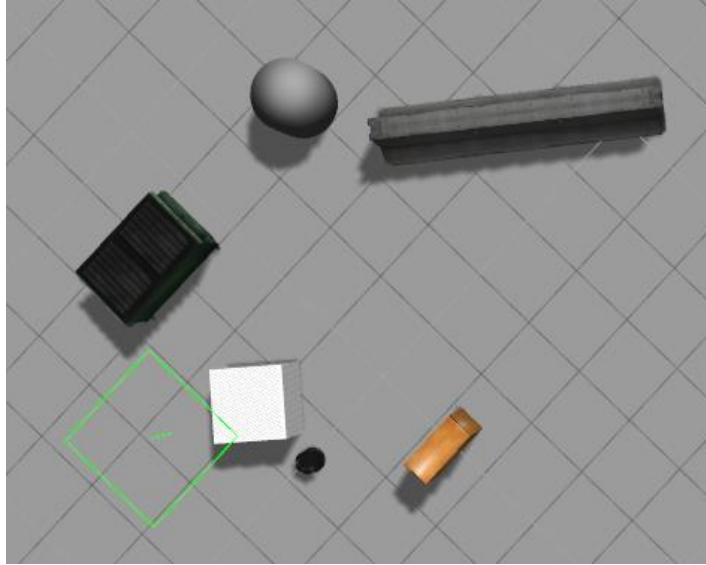


Figure 32- Gazebo Simulated Environment

5.2.3.1. GMap

Gmapping implements a Rao-Blackwellized particle filter to learn grid maps from laser range data [72]. This can be used in ROS in conjunction with either simulated or physical hardware to navigate the environment. Figure 33 shows the result of gmapping the gazebo environment. It can be seen that the resulting map is two-dimensional, with the borders of the objects depicted as black outlines, unmapped areas in grey, and the floor mapped as white.



Figure 33- Gmapping Visualized in RVIZ

While populating the gmap, the robot was navigated around the map using teleoperation. The speed had to be slow due to the settings of the application, as well as the load of visualizing the map in realtime. However the underlying particle filter behavior was apparent as the robot's position would move as the map was updating rather than as a direct result of teleoperating. It can be seen that the outlines of the objects aren't completely discreet, and in fact do update as the robot observes more features of the environment. The outline of the cylinder, for example, can be seen to not be perfectly circular, as a result of seeing it in the scan while moving. It can also be seen that the trashcan (just to the left of the cylinder) also has intermittent outlines, as a result of updating its border while moving and rotating.

While the result of this map is crude, it can be used with relatively high probability to detect and avoid the general vicinity of objects. The latency of the system however, might pose a challenge. Tests will have to be made using gmapping using the Jetson in order to determine the satisfaction of the functional requirement of latency.

One major advantage of this type of mapping is that the whole map has the same probability of position so any algorithm ran using this map is as accurate as the map itself. Given the designers of this package ensure it is very efficient and with reasonable certainty, pathfinding and routing can be done very efficiently using this technique.

5.2.3.2. Octomap

Octomap implements a 3D occupancy grid based on octree [73]. Octomap is able to map completely arbitrary 3D environments as well as free space. The map is also updatable using different sensors, employing a probabilistic approach to filter out sensor noise or dynamic environment changes, e.g. dynamic objects. One key feature that can be exploited by this project is that multiple cars can contribute to the same map at any time. The map is dynamically expanded as needed, and as such could be expanded indefinitely. There are also different resolutions such that there can be one controlling server hosting the high-level

map, which can then send smaller sections with higher resolution to individual cars.

This has obvious application to the networked aspect of this project, and can also be applied to simulated or physical hardware using ROS. Figure 34 shows the result of mapping the same gazebo environment depicted in Figure 33 using octomap. It can be seen that the octomap is three-dimensional, with the borders of objects colorized to show height. It's also apparent that the general shape of objects are preserved via small squares to approximate all sides of objects, including the cylinder and boxes at different angles. It's also shown that the map can only depict what is seen by the sensors of the vehicle, so the interior of the cylinder and garbage can appear to be hollow only because the sensor have no information of that region. However because it cannot be accessed due to the collisions of the borders, that region can be left empty to preserve processing power.

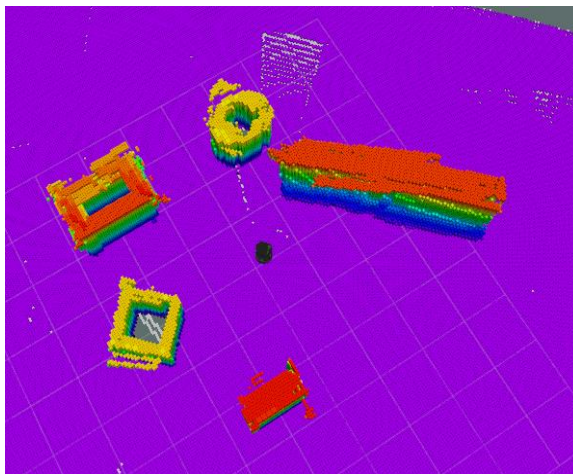


Figure 34- Octomap Visualized in RVIZ

The latency of this mapping technique was notably faster than gmapping, and thus would be a better decision for quicker collision detection information. The 3D aspect as well as the collaborative aspect of this technique make it yet a stronger candidate to be a dedicated collision detection map.

5.2.4. Collision Detection and Avoidance

Either used in conjunction or separately, the two mapping techniques can be used to generate maps of objects in the

vehicle's surroundings. These maps can be used to detect and avoid collisions based on the vehicle's current position in the map. Since multiple vehicles are able to update the same map, they will be able to model any surrounding vehicles and use that along with synced positions of neighboring vehicles to help each other localize themselves.

5.2.5. Motion Tracking with Optical Flow

In the ongoing developments of robot vision and machine learning, the process of tracking motion remains to be an extremely influential attribute to the success of the system. For autonomous vehicles, the ability to track another object's motion is essential for localization purposes and predictions of path and velocity of surrounding objects in motion. One method of implementing this concept is by using optical flow programming to a visual sensor such as a camera.

While there exist many different schemes and strategies to implement motion tracking, in the following sections, we will explore the OpenCV platform and subsequent libraries used. At this stage of testing, the requirements have yet to be refined to fit the final product; this program will serve as a basis on which to build towards autonomous vehicle purposes. The Optical Flow Initial Test Specification Requirements are as follows:

- Dense optical flow live-tracking via camera input
- Output window of red-blue-green representation of motion
- Stationary objects shall be omitted from output
- Bounding box shall target the object in motion
- Bounding box shall be displayed in foreground
- Code written in Python language

5.2.5.1. OpenCV Platform and Theory

Optical flow is the procedure in which a program compares the pixels between two consecutive images and makes various assertions relative to the change in position of each pixel [29]. By forming a bounded grid of pixels, the program should be able to both recognize particular objects by use of a database and track them by performing various calculations of the pixel differential.

For this particular project, OpenCV was the primary resource of the logic needed for our robot vision requirements.

OpenCV (Open Source Computer Vision Library) [30] stands as a major contributor to computer vision and machine learning, boasting grand database function libraries and algorithms. Fortunately, there exist OpenCV Python libraries for nearly every functional task that a machine vision system needs to perform. In this project, the initial installation of Python and OpenCV was required for testing. After doing so, importing the desired libraries to obtain the algorithmic functions was necessary to test the effectiveness of the OpenCV library. A laptop with an in-house webcam served as the initial test setup for experiments with optical flow, motion tracking, and object recognition. The following sections will denote specific OpenCV functions used for these purposes, as these are powerful means to produce a functional visual output stream. All of the code used in this experiment was written in Python 3, but should be functional in Python 2 if hardware cannot support the newest version.

5.2.5.2. Dense Optical Flow Test

The testing phase began with a specific category of optical flow called Dense Optical Flow. Dense Optical Flow differs from other forms of optical flow in that the output target is solely the result of motion; this means that all objects not-in-motion will be omitted from the output similar to a “don’t care” condition, visualized as a black backdrop. Other variations of optical flow are designed to deliver outputs of vision in parallel with motion differential data, this was undesirable as the aim is to isolate moving objects. OpenCV Dense Optical Flow uses the Lewis-Kanade algorithm set to provide the user with a color-coded visualization of motion where a specific color indicates the characteristic of the motion; hue (shade) represents direction, while value (brightness) represents magnitude/velocity [31]. Creating a Dense Optical Flow test program required installing “*cv2*”, “*numpy*”, and “*matplotlib*” libraries. The program then accessed the laptop’s camera to obtain a standard image of the room using the *cv2.VideoCapture()* and *.read()* Python functions in OpenCV 3.3.0. This provides a continuous data flow to arrays within the function, one may dedicate a variable to store this information for further calling, here named “cap”.

Once the program is granted access to the built-in camera, it begins to store the data in the format suited for the Lewis-Kinade method of motion tracking. An initial variable, "frame1" is obtained from the previously mentioned `.read()` function; this is where the program stores one of the frames for future comparison purposes. A variable named "prvs" is defined as the grayscale image of the capture by using the function `cv2.cvtColor(..., cv2.COLOR_BGR2GRAY)`. "prvs" serves as the first of two images being compared. It also is necessary to have a grayscale version of the image for implementation of the algorithms, the argument `COLOR_BGR2GRAY` indicates that the variable array is being transformed from a blue-green-red image to a grayscale. The program then creates an array of zeros, here named "hsv" identical to that of frame1. This is done through the "numpy" function `np.zeros_like()`. The variable "hsv" stands for hue-saturation-value. The array is sliced and the second (1th) element set to 255 for maximum saturation. The program then enters an infinite "while loop" for continuous operation with the dynamics of motion tracking all contained within the loop. Entering the loop, a second variable "frame2" is created which reads the data from the camera, again. Because of the latency between the few lines of capture commands, these two frames are not identical, but differ by an extremely small amount of time. A variable "next" is defined as the conversion of the second capture to grayscale, much alike the variable "prvs".

In this stage, the program begins using statistical processes. Another variable named "flow" is defined as the output from a "cv2" function `calcOpticalFlowFarneback()`. Various integer values are necessary to be placed within the function - these are not mentioned here due to their precise and standard nature in the probabilistic determination of an object in the frame. The inner-workings of this function are very complicated, and the output is not an image but an array of numerical data; therefore, these values must be converted back to a usable format to view the output image. To do so, we define two more variable "mag" and "ang" to transform the data into polar vector coordinates. The array "hsv" is loaded with these polar values into sliced elements of hue and value. The array "hsv" is fed back into the `cv2.cvtColor(..., cv2.COLOR_HSV2BGR)` function to transform the image back to standard color for viewing ease. To display an

image window of the output from the algorithm, the `cv2.imshow()` function was used. This function creates a window with a label, takes in one variable array, here called "bgr". The `cv2.waitKey()` function is used to complement the output image window; it is necessary to set a certain, yet arbitrary value, otherwise the program may freeze at. At the end of the while loop, we set "prvs" equal to "next"; this updates the second frame to the first frame, allowing the program to obtain another fresh image to compare to the ever-changing "prvs". Finally, the while loop is exited and two more functions: `cap.release()` and `cv2.destroyAllWindows()` are employed to terminate the program without error. The result, if done correctly, should be similar to the still shot show below in Figure 34:

In Fig. 35, one can observe the various color-coded aspects of motion. The contours of the image are highlighted due to the backdrop being a significantly different color (a white wall). Due to the pixelated differential of my outline to the backdrop, this is considered to be a change. One may notice that elements of hair and skin are blacked-out, for there is little relative change between pixel colors. While the backdrop possessed various items such as room decorations, these are a constant black due to the fact that although they are different coloration values, there is no change between them in the two frames being compared. To better visualize how magnitude and direction of motion are captured, refer below to the image in Figure 36. Here, the image underwent more drastic motion unlike that of Figure 35 where it was relatively still. As the object turns, certain elements of the contour are highlighted while others remain a constant color. This demonstrates how the output of the Lewis-Kinade method and the `calcOpticalFlowFarneback()` function provide the user with an image that reflects not only motion in general, but the differences between motion. Certain areas of the image were changing faster than others, or in a different direction. Turning, the green/yellow areas are highlighted to show a horizontal position shift. The white/light blue area of hair is highlighted in that way to show a vertical change in position. The intensity of each region indicates how fast it is moving relative to black (no velocity). It is simple to see why this particular process is very useful in dealing with computer vision scenarios - the differential output allows for predictions to be made with the vector components of position

change i.e. directional velocity. The entire process is shown as a block diagram in Figure 37.



Figure 35- Still Shot of Dense Optical Flow Output

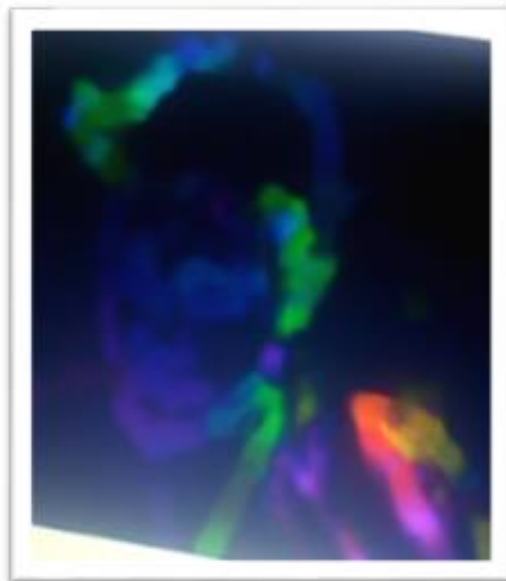


Figure 36- Still Shot of Dense Optical Flow Output in Motion

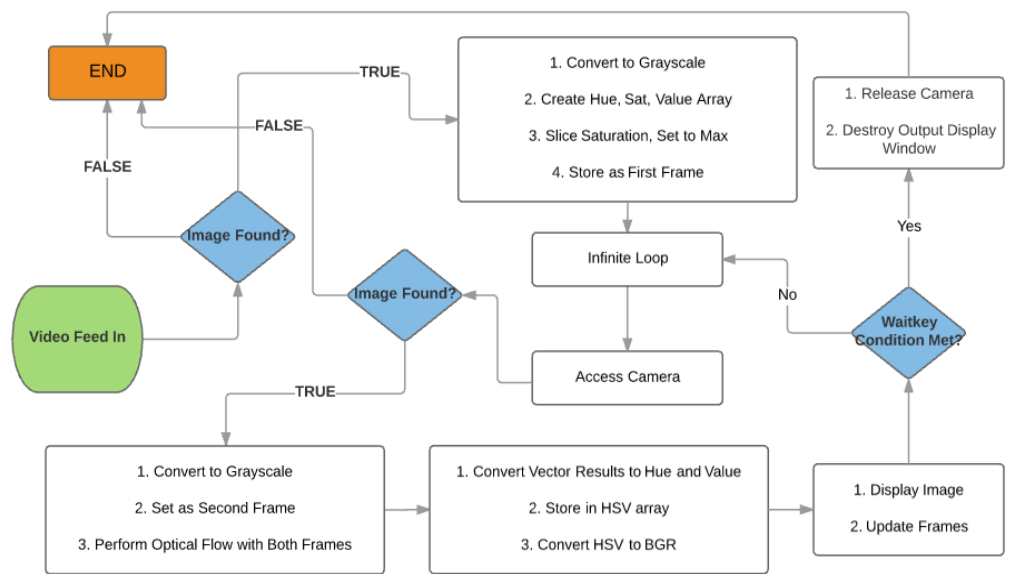


Figure 37- Dense Optical Flow Program Flowchart

5.2.5.3. Bounding Box Test Using Image Centroid

As mentioned in the previous section, it was desired for the program to output a “bounding box” around the object in motion. A bounding box is a rectangle or square in the foreground of the image that follows a desired target. This bounding box should be unaffected by the processes and functions of optical flow output configuration; the rectangle should exist as an overlay without undergoing any manipulation. For this requirement, the optical flow data was transformed and fed into a separate code block to process the information in parallel with the standard dense optical flow procedures.

There exist many possibilities and metrics to targeting a specific object. In the initial testing, this was done with the whole image. In doing so, it is considered there be only one object in motion in the field of vision that is of concern, as the densest region of colored pixels will be considered the target. While this may prove advantageous in certain scenarios, the final autonomous system shall require multiple bounding boxes if one is to consider real world conditions. However, for the purposes of primary development, the centroid/moment method of threshold comparison was utilized.

To create a bounding box of static proportions (unchanging width and height), one must manipulate the information within the main while loop of the program. The `.calcOpticalFlowFarneback()` function must already have been called, as the goal is to target the object in motion and not a specific object in the image. Logically, here, the bounding box will take the image data from the function after it has been transformed into a usable image by transforming the data into polar vectors. Ergo, the variable “bgr” will be used as the primary means of input, as it is defined originally as the usable image data after optical flow procedures. At the point of use, this image data is the blue-red-green representation of motion that is seen in Figures 33 and 34. This variable is sufficient to use as it subtracts all pixels that are not in motion.

The methods of “drawing” a bounding box upon an existing image are developed by first finding the contours of that image. Not only does contouring assist in this task, but may prove useful in a variety of detection requirements such as “drawing” the outline of motion, or path. The OpenCV contouring process is most efficiently computed in grayscale; therefore, the first objective for a bounding box is to convert the image as such, by using the `cv2.cvtColor(...,cv2.COLOR_BGR2GRAY)` function.

A new variable “gray” was defined as the output of this function, the other input argument being “bgr”. Secondly, a variable called “thresh” is defined; this will be the threshold of the image. A threshold image is an output that further subtracts various elements from the input. For example, thresholding an image of an apple on a table by isolating the red color would result in an output of solely the apple as the threshold, or limit to the output. The variable “thresh” was defined as the output from the OpenCV function `cv2.threshold()`.

Arguments to the function included the input, “gray”, certain grayscale color values, as well as the method of thresholding `cv2.THRESH_TRUNC`. Three variables are then defined to be the simultaneous outputs of the function `cv2.findContours()`, which takes input “thresh” as well as two methods, here, `cv2.RETR_EXTERNAL`, and `cv2.CHAIN_APPROX_SIMPLE`. The former argument is used in order to subtract the possibility

of contouring more than one object, and the latter argument is used for the purposes of limiting the contour to four points. By using these methods, we will compute only the necessary contours of the "brg" image data [32]. While this function requires writing to three variables, the first of which "im2" shall be called upon later; the other two variables are of arbitrary definition and represent the contour data and the hierarchy data. "im2" now serves as the image of the contours found from the "brg" dense optical flow image.

The program will now utilize methods of centroid approximation to find the center of the contours defined by the aforementioned processes. We define a variable array "M" to take outputs from the OpenCV function *cv2.moments()*, feeding "im2" as input. This moments function will create an array of pixel value density. Once obtained, the centroid is located by dividing the array "M" by its first 0th elements. These are denoted as "cx" and "cy". Finally, use of the function *cv2.rectangle()* will draw the box onto the input argument, "bgr", around "cx" and "cy" found from the contour centroid. Further arguments of the *cv2.rectangle()* function include setting the box's width and height to be of 150 pixels from each centroid, to possess a green color (0,255,0), as well as to be a thickness of 3 pixels. The whole code block should exist before the *imshow()* function, as the bounding box will be included in the final output. The final test output combining the dense optical flow and bounding box overlay is as shown in Figure 37 below:

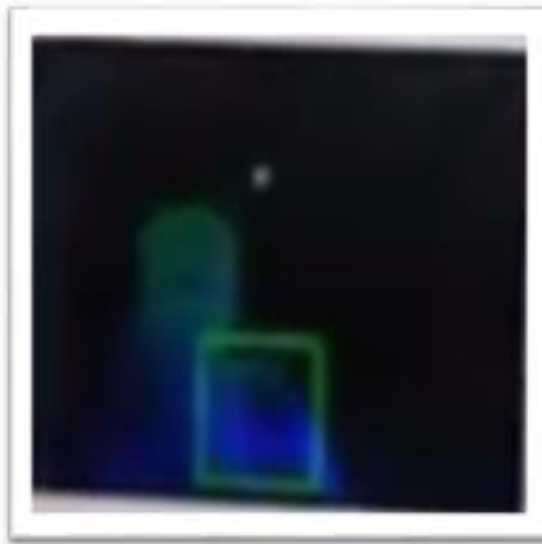


Figure 38- Still Shot of Dense Optical Flow Output Bounding Box

While the image in Figure 38 is unable to be seen with a still shot, the box continually followed the illuminated moving object. Further considerations of improvement include a bounding box that disappears with at minimal movement as well as bounding more than one object simultaneously.

5.2.6. Object Detection through Histogram of Oriented Gradients

A key function in machine vision intelligence is the ability to detect and identify objects. If a robotic system is able to distinguish, for example, a pedestrian from a collective image – it introduces a grand opportunity for the system to react specifically to various scenarios. This feature is popularly achieved by the use of a “histogram of oriented gradients”. This concept involves separating an image into parts and analyzing each “block” piece-by-piece in order to detect a specific object through a probabilistic nature. It is easy to envision why this plays a very important role for autonomous vehicle design. Modern autonomous vehicles must be able to distinguish a multitude of objects and have the ability to process a response to the behavior and location of each object. For the purposes of this project, histogram of oriented gradients or “HOG” will be integrated into the software for simple object detection. While detecting motion through dense optical flow is crucial, it is also highly beneficial to the overall intelligence

of the system to process object detection in parallel with optical flow.

5.2.6.1. Histograms of Oriented Gradients Processes

The method behind object detection in HOG is a clever means to separate a desired grouping of pixels out of an image and classifying it as an object. The full image is considered as an input; it is then processed so that pertinent information is extracted and the rest is considered extraneous. This is done by applying a horizontal and vertical gradient to the image in overlapping blocks. The image is divided into subsections with a specified cell width of x and y pixel dimensions; the blocks will overlap such that the difference between sections may be detected as a gradient [83]. The number of blocks used to form the gradient is coincident with the size of the image; therefore, it is common practice to resize the image in accordance with the type of object one is detecting. Additional divisions do not necessarily imply a more accurate result, the objective is to apply the appropriate amount of divisions to where a difference in pixels is apparent between adjacent blocks.

Before any computational processes on the input image are performed, there must be a database to obtain a reference for what object the program is attempting to identify. This is done by creating a reference library of sample images containing the desired object and samples lacking the object. In practice, it is more beneficial for the number of negative samples to outweigh the positive for more reliable detection [84].

Returning to the image processing, a “sliding window” technique is applied where the program will test each block and record the RGB values of the pixels. This value is represented as a normalized vector. The result will be a vectorially-represented visual difference between each cell. This process is implemented over the sample library as well, and the resulting vector fields of each reference image are compared to the test image. Comparing the vector fields of the test image to the reference images allows the program to normalize the histogram, (probabilistic weight within a region), and determine that a specific object is present within the image. By this cellular method, it also allows the program to determine where the object

is located within the full image, making it a simple process to then bound the object's contour with a bounding box.

5.2.6.2. Open CV Pedestrian Detection

As discussed in the “Motion Tracking with Dense Optical Flow” section, OpenCV is a veritable resource in image processing. Python language was used to apply the HOG functionality to detect objects in a stream of video. Unlike motion tracking with dense optical flow, the image was not altered – instead, an RGB image stream must be kept intact to properly visualize the object as it is. The “cv2” library contains a histogram descriptor for pedestrian detection, used here as the initial test for object tracking. The Python program is relatively simple for pedestrian detection, as the libraries and functions have already been formed. For the purposes of testing, there existed certain specifications this program was designed to achieve. These are bulleted below:

- Utilization of histogram of oriented gradient
- Detection of an upright humanoid object
- Capture of multiple objects in a single frame
- Bounding of each object
- Visual verification
- Program implemented in Python language

The program was tested on laptop with an in-house camera. The program begins by importing the appropriate libraries, (numpy, imutils, and cv2), and defining the histogram of oriented gradients. The variable “*hog*” is created and set to the function *cv2.HOGDescriptor()*. It is then fed to a function in which the pedestrian detection library and methods are specific to the desired pedestrian object, *hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())*.

Following the initialization of the HOG, the program enters an infinite “while” loop to run indefinitely. Once the loop is entered, the program captures an individual image with the *.read()* function. The image is resized both for proper gradient cell application and increased processing speed; the image is resized to a square with side length of 500 pixels. This new resized image is defined as the variable “*img*”. Next, the variable “*hog*” is fed

into the function `.detectMultiScale(...)`; the contents of the function within the parentheses include the resized image as well as several parameters for proper cell division and window sliding. The output of this function is assigned to two variables named “*rects*” and “*weights*”, where “*rects*” contains the location data of each object detected and “*weights*” contains the probabilistic data of how close the match is.

Next, the program will enter a phase in which it will draw a bounding box around the “*rects*” localization data. An array is created with the `np.array()` function which houses the dimensions of “*rects*” in x-y coordinate plane. The variable “*pick*” is assigned to the output of the function `non_max_suppression(...)`; the parameters within this function include the input, “*rects*” as well as probabilistic and threshold variables which will allow overlapping of bounding boxes. This step will enable the program to output more than one bounding box at a single instant with the ability of the boxes to overlap; a larger overlap increases the likelihood that a box will appear around a certain gradient contour, or “*pick*” an object. The program then enters another “for” loop, initializing secondary x-y coordinates for the actual bounding box. The `cv2.rectangle()` function is used, here, to cycle through the variable “*pick*” and draw a green bounding box with a line thickness of 2 pixels. For more information on rectangle drawing parameters in Python, please refer to the “motion tracking with dense optical flow” section. Once the bounding box has been computed and drawn, the program displays the output image with the `cv2.imshow()` function. A “waitkey” is used for proper execution, as well as the `.release()` and `.destroyAllWindows()` functions for proper termination of the program. A summary of this process is shown in Figure 39.

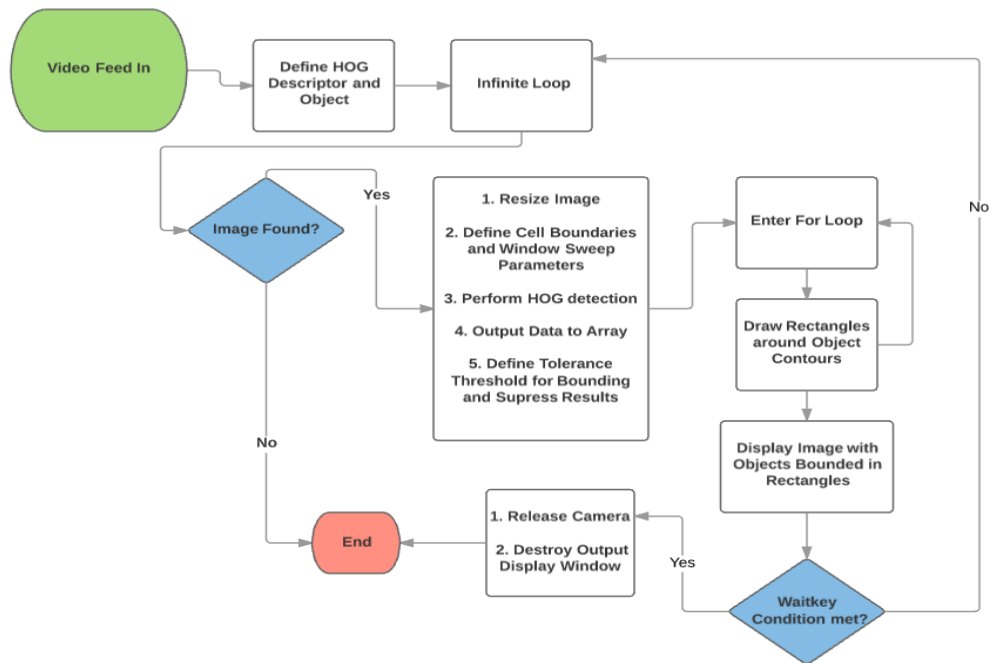


Figure 39- Flow Diagram of HOG Pedestrian Detection Program

5.2.6.3. Pedestrian Detection Testing and Output

As the program was ran, it became clear that this complicated procedure requires certain alignment with the exact type of environment one is operating in. For example, the sensitivity of detection is adjusted by altering both the number of cell divisions and image size; furthermore, the overlapping threshold in bounding the objects can be manipulated to allow or deny a bounding box to be applied at a certain likelihood of an object. It was also found that the speed of the program was sub-standard for the employment in an actual moving vehicle scenario. This is likely due to the visual verification process – that is, in the testing phase where one must create a visual aid to accurately display the program’s behavior. In a real-world scenario, this data would not need to be rendered visually, but simply fed to the processor responsible for object localization.

5.2.6.4. Pedestrian Detection Test 1

The following figures display the output of the program ran at the following parameters found in Table 24.

Table 24: HOG Test Results

Parameter	Value	Description
WinStride	4,4	Sliding window area of 4x4 cells
Padding	32,32	Image divided into 32x32 cells
Scale	1.05	Perform sliding window method to multiple image scales up to 5% of original
Width	500	Original image side length resized to 500 pixels (square)
overlapThresh	0.85	85% overlap in object bounding

As one can observe in Figure 40, the program was able to detect two, upright, humanoid shapes while ignoring the triangle. This is the simplest application of the HOG concept as it is very easy for the program to distinguish the gradient difference of a dark silhouette on a white background. The program was then tested with a more complex image:

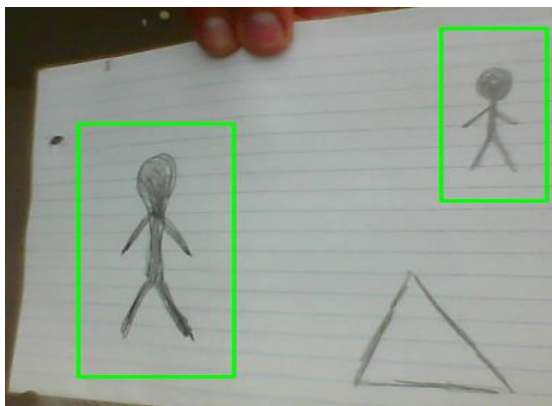


Figure 40- Simple HOG Pedestrian Detection Test 1

In Figure 41, (faces omitted for privacy), the image contained many different opportunities for error. The increased number of pedestrian objects added to the complexity of the image. Furthermore, the pedestrians were not fully visible, they were overlapping, and the multitude of color in the image makes for an uneven gradient. However, one can still observe that the program performed relatively well. To the left and middle, the pedestrians are ignored which should have been detected; however, the

pedestrians on the right and lower-middle were detected with higher accuracy due to a better-defined outline and color differential.



Figure 41- Complex HOG Pedestrian Detection Test 1

5.2.6.5. Pedestrian Test 2

The following figures display the output of the program ran at the following parameters found in Table 25. Various metrics were altered to test the effects on accuracy and tolerance. While the type of image being detected influences the results greatly, tuning these metrics individually allowed for an observation of how each could benefit or harm the desired response.

Table 25: Test 2 Parameters

Parameter	Value	Description
WinStride	4,4	Sliding window area of 4x4cells
Padding	64,64	Image divided into 64x64 cells
Scale	1.05	Perform sliding window method to multiple image scales up to 5% of original
Width	400	Original image side length resized to 400 pixels (square)
OverlapThresh	.95	95% overlap in object bounding

In Figure 42, an increase in the number of bounding boxes from test 1 can be seen. This is due to the image being separated into 64x64 cells instead of 32x32. The overlap threshold is

responsible for the larger boxes, as it now has a higher tolerance and aims to remove redundant boxes. Image rendering was faster, seeing as the image was scaled smaller. This can be seen as an improvement from test 1 in complex imagery. However, results were not as positive for simplified cases.

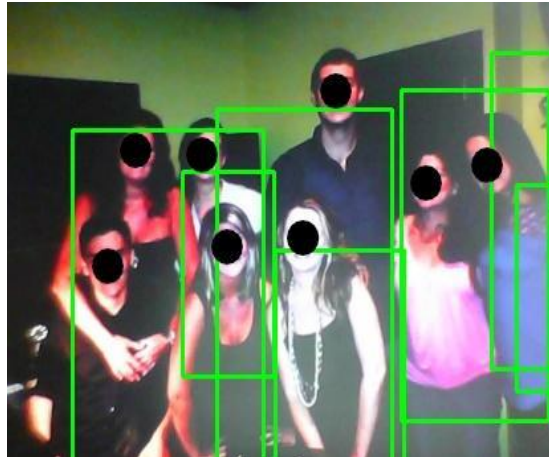


Figure 42- Complex HOG Pedestrian Detection Test 2

In Figure 43, the program is now too sensitive for simple fields of vision. Where it had ignored the triangle in test 1, the triangle is now erroneously considered a pedestrian due to the threshold tolerance increase. Additionally, the left humanoid was double counted – not necessarily a decrease in effectiveness, but inaccurate. It is clear from these results that the parameters should be balanced in such a way to provide consistent results depending on the environment. It should also be noted that the program latency was much too high when the sliding window was set to 2x2 or lower. Increasing scale drastically increased the latency between operations with decreased sensitivity. Image resizing to 1000 pixels resulted in approximately twice the latency, but improved accuracy. It appears that the ratio of cellular padding determines the sensitivity, while image resizing and overlap threshold should be tuned for redundancy and accuracy.

Further considerations for improving this system are to “retrain” the HOG detection for a specific environment. This is commonly done by updating the reference library with positive and negative data from past test results, allowing the process to adapt to a certain environment [84]. One could also consider a dynamic library for reference images e.g. urban vs rural. This would

involve a more complicated means of image processing to determine which environment the reference point exists in.

Additional applications of HOG detection in autonomous vehicle operation may involve the detection of common objects on the road such as: other vehicles, road signs, turn signals, lane boundaries, and many others. If one is to supply the HOG descriptor with a large sample set, the program should be able to detect nearly anything - within the constraint of processing speed.

It can be surmised from the test output data that there exist clear limitations to HOG pedestrian detection. If the program is to operate effectively, there must a clear distinction between object and background. The objects should not have a significant amount of overlap and should not be crowded into one area. While these ideal conditions are obviously not the case from a moving vehicle reference point, it stands to assume HOG detection is a viable method of recognizing and isolating a pedestrian or local object from the field of vision.

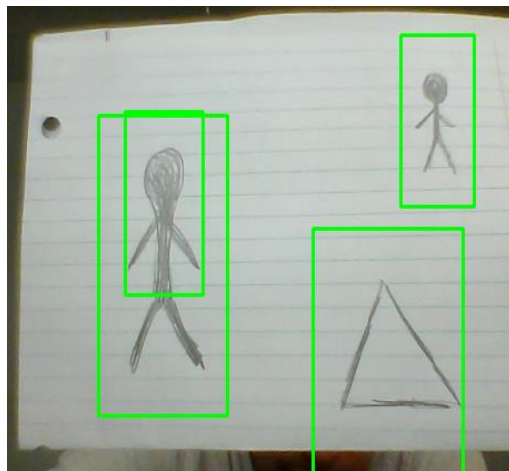


Figure 43- Simple HOG Pedestrian Detection Test 2

5.3. Vehicle Dynamics and Modeling

In this section we cover testing done on our vehicles components and any modifications made to the vehicle base, such as differential and spring replacement, and any information obtained from the vehicle operations, such as steering angle range and motor speed. Not every modification may be implemented by our

team due to time constraints, but will be implemented by our contributors in Dr. Fallah's research lab.

5.3.1. Vehicle Disassembly

To improve the performance of the vehicle, the Traxxas Slash 4X4 Platinum was disassembled to determine what mechanical pieces would needed to be replaced. Tests on numerous components, such as hall sensors, the brushless motor, and the servo, were conducted to find the signals necessary to control the vehicle and improve performance.

5.3.1.1. Differential Replacement

After disassembling the vehicle, the contributors came to the decision to remove the clutch on the limited slip differential that attached to the front and rear axles. Referring to the section on differentials, a limited slip differential would cause the wheels to lock if they tried to move at different speeds. Since we cannot us the measurement of the RPM's on a slipping wheel to calculate turning distance, this could lead to issues in localization. The best way to avoid this issue was to replace the clutch with a center differential so the wheels can move at different angular velocities. An additional benefit to this change is that it will make turning smoother in the process. A photo of the clutch and center differential is shown in Figure 44.



Figure 44- Center Differential (left) and Clutch (right)

5.3.1.2. Springs

After loading the weight of the battery, NVIDIA Jetson-TX2, and the ZED camera, the springs on the vehicle's front shocks were not stiff enough to prevent bouncing. Research was done to find springs from Traxxas or other companies that cater to hobbyists, but the price range and shipping times were absurd for what we were asking for. Instead, it was decided that a second spring would be placed in parallel with the original. Two springs in parallel function the way that two capacitors in parallel do (see Figure 45).

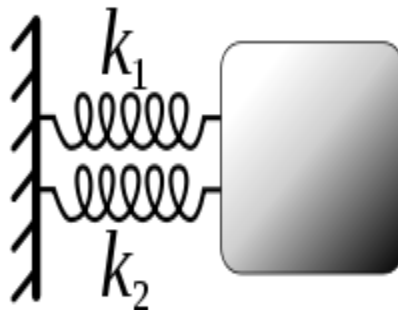


Figure 45- Two Springs in Parallel

In the case of the capacitors, the overall capacitance increases to the value of the two capacitances added together. In the case of the springs, the spring stiffness increases to the value of the two spring stiffness's added together. A picture of the two springs in parallel on the vehicle is shown in Figure 46. The new springs were obtained in an Ace Hardware and then were cut in half, and each half was placed on the inside of one of the front shocks. This removed the bouncing issues entirely and will be done to every vehicle constructed by the senior design team.



Figure 46- Two Springs in Parallel on the Shock

5.3.1.3. Servo Testing

To determine our steering angles, we performed tests using a function generator and the Arduino UNO to find the range of pulse width's that controlled the servo angle without damaging the servo. After we determined our steering angle range, we programmed the Arduino UNO to implement changes in wheel speed. The goals of our tests were as follows:

- What frequency should we run our pulses so we get the largest range of control via duty cycle?
- What is the range of our steering angles?
- How can we get our UNO board to create the necessary pulses and pulse widths?
- How can we change the speed the wheels change its angles at?

- How do we make the change in wheel speed independent of distance between 2 different steering angles?

5.3.1.3.1. Determining Pulse Widths

To determine our necessary pulse-width range, we began by attaching our servo's control wire to a function generator running a pulse at 50 Hz, the positive voltage wire to a 5 V DC source, and the ground wire to the ground of the servo. The grounds are connected through a breadboard. This configuration is shown in Figure 47 below.

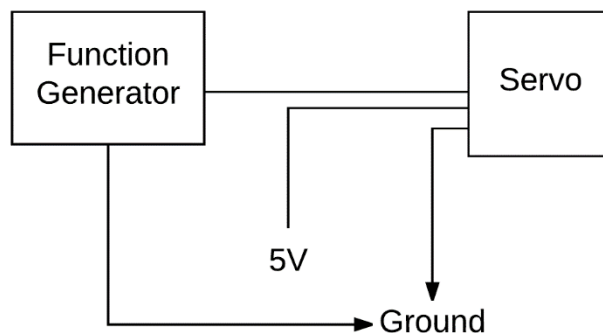


Figure 47- Test Setup for PWM Measurement

We began by placing the duty cycle at 50%, and then turned on the servo. We saw the servo try to extend past its maximum range and immediately powered off the generator. We then lowered our duty cycle consistently until we found ourselves in the middle of the working range. We then increased and decreased our duty cycles slowly to find the range of our servo. Using this setup, we determined that pulses ranging between 1100 microseconds and 1900 microseconds provided the full range of motion for the servo.

5.3.1.3.2. Determining Steering Angle

After determining our PWM range, we decided to attach a ruler to the outer front wheel of the vehicle. We adjusted the PWM through the entire range of motion in equal intervals of 0.01% at a frequency of 50 Hz and took photos after each change to capture the entire range of motion. We then used Photoshop's ruler tool to find the steering angle of related to each pulse width. The test setup is show in Figure 48. Our final Steering angle

range was determined to be approximately 22 degrees to the left and 24 degrees to the right. We are expecting about 1 degree of error due to our method of measurement, so we assume we have 23 degrees on the right and left side.



Figure 48- Steering Angle Test Setup

5.3.1.3.3. Determining PWM Frequency

After determining our steering angle and range, we tried numerous ways to change the PWM frequency of the Arduino output pins. We discovered methods involving using delays, but due to the extra memory consumed by the code, the inability to use dividers, and the amount of variables we needed to pass to a function to implement a variable pulse width, we decided to use the standard 50 Hz signal that is output when using the *Servo.write* command in Arduino's Servo library.

While the difference in pulse quality was minimal (see Figure 49), we encountered issues when trying to run the servo using the servo library. We received a much smaller range of total points than expected and a lower quality pulse. Arduino's servo library is supposed to be able to adjust for different pulse widths, but adjusting our range to match the range we had calculated led to a smaller range than the default one. Instead of having 180 points like we originally thought we would, we were left with a total of 53. These 53 points, however, gave a mostly linear response for our steering angle. A graph of steering angle and duty cycles created by the *Servo.write* command is shown in Figure 50. Table 26 shows the final results of our testing to this point.

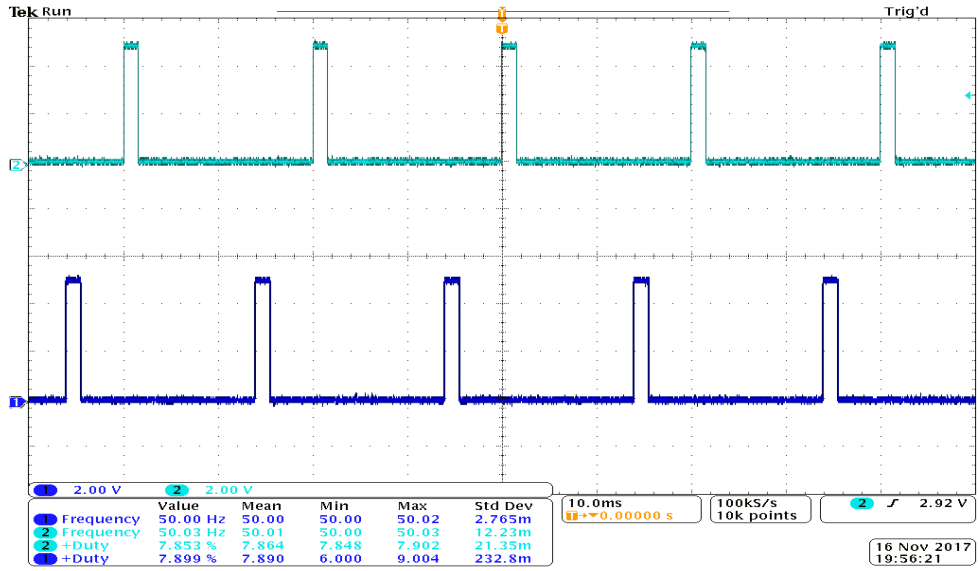


Figure 49- Pulse from Function Generator (blue) vs. Pulse from the Arduino UNO (yellow)

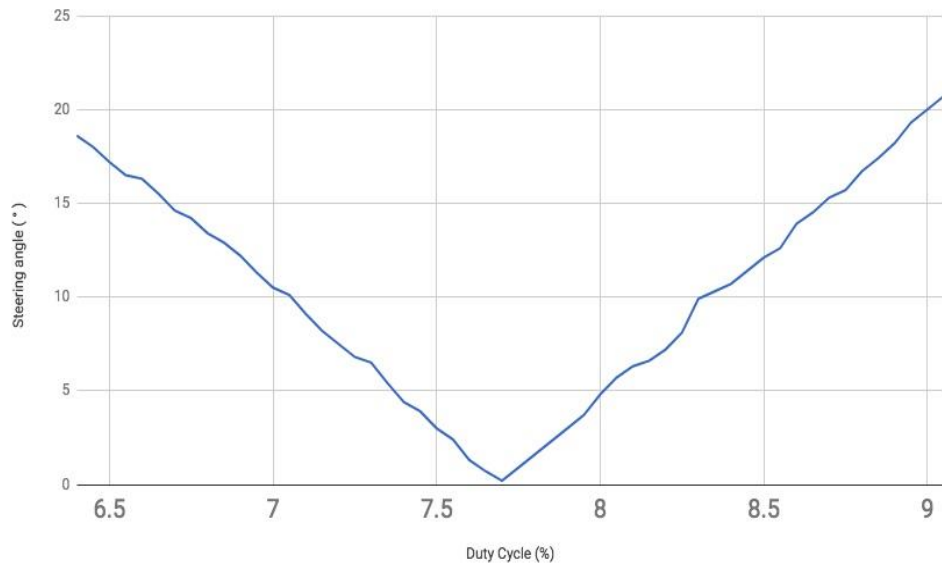


Figure 50- Duty Cycle vs. Steering Angle

Table 26: Turning Parameters

Parameter	Measurements
Frequency of Control Signal	50 Hz
Duty Cycle Range	5.5%-9.5%
Steering Angle Range	46 degrees

5.3.1.3.4. Changing Steering Angle Speed

After obtaining our range of motion and the input range for our vehicle, the next step became to implement code to allow for variable changes in steering angle e.g. changing 10 degrees in 1 millisecond or 10 degrees in 10 milliseconds. Figure 51 below shows the logic which was developed to achieve this. Figure 52 shows the logic implemented in the smoothing function which is used to prevent jerking motions resulting from linear servo movements.

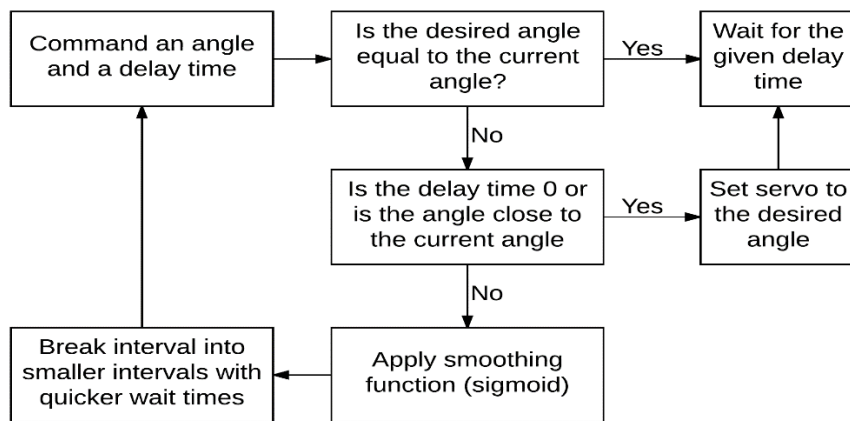


Figure 51- Smooth Steering Logic

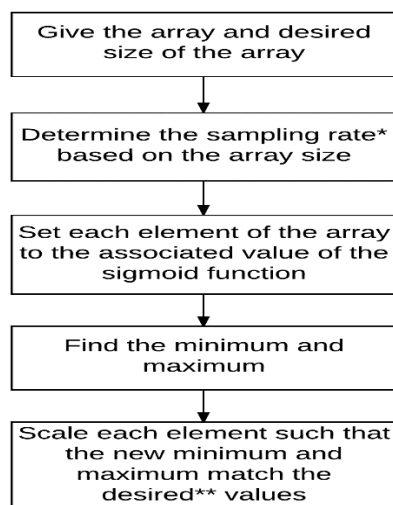


Figure 52- Smoothing Function

5.3.2. Measuring Motor Speed

The next step in setting up our vehicle was to determine the speed of our motor. Using this data, we approximated our wheel speeds using the gear ratios between the spur gear, center differentials, and front and back differentials to obtain speed assuming the wheels do not slip. In order to measure motor speed, we decided to use Traxxas' Hall Effect sensor, which was specifically designed for the Traxxas Slash 4X4. The sensor testing consisted of measurement of voltages and writing a code to detect the movement of a magnet attached to the spinning motor gears. The testing setup is shown in Figure 53. The sensor is placed in a small compartment near the spur-gear. It can be seen surrounded by a red rectangle in Figure 54.

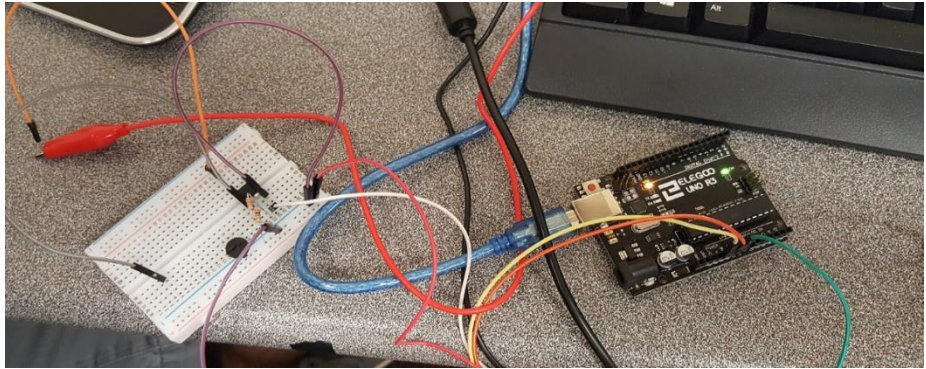


Figure 53- Hall Effect Sensor Testing and Coding



Figure 54- Installed Hall Effect Sensor

The magnet was then placed in front of the sensor to find the output voltage. The output voltage measured was about 0.004 V,

which was too low to be usable. To circumvent this issue, a 1 kilo-ohm pull-up resistor was connected between the magnet and the sensor to raise the output voltage to boost the voltage above 3.3 V so it could be detected as “on” by the Arduino UNO or the Jetson board. When the magnet is introduced, the voltage goes back down to 0.04 mV. The output of the sensor was then attached to digital pin 2 of the Arduino, and whenever a “0” is detected, the Arduino updates a counter called revolutions. After 100 revolutions are detected, the code then calculates revolutions per minute using Equation 8, where the time is measured in milliseconds using the millis() function in the Arduino library. The program flow is described in Figure 55.

$$RPM = \frac{100}{\text{millis}()}$$

Equation 8. Revolutions per Second

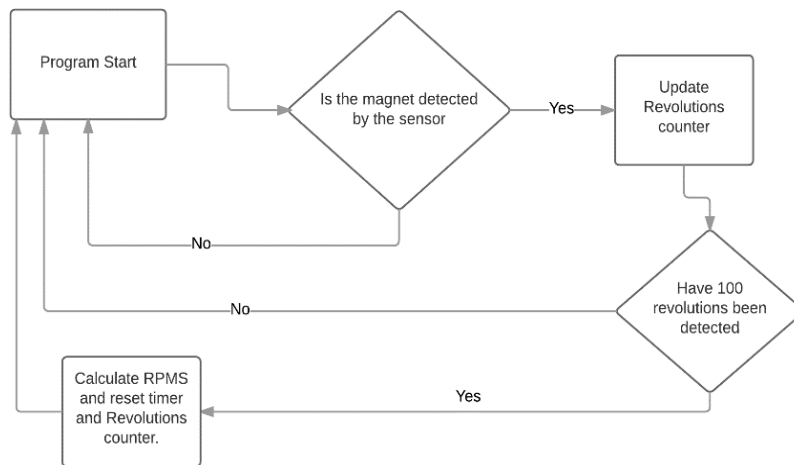


Figure 55: RPM counter flowchart

5.3.3. Three-Dimensional Modeling – Initial Proposition

Prior to assembling any product, one must achieve a visualization of any and all components and how they are to be fit together for the most efficient design. In terms of autonomous vehicle design, each outfitted sensory or computational component must be modeled to-scale to adhere to the project’s

special requirements. In addition to location, the relative weight of each object must be considered as to not offset the vehicle's center of mass. The following document contains the first proposed model of all components adhering to the basic RC vehicle chassis.

To produce a visual three-dimensional model of the, AutoCAD 2016 software was utilized. The vehicle chassis was traced from an image and rasterized as accurately as possible for the current visualization requirements. Once the chassis was converted to a three-dimensional scale, each component was created one-by-one and added to the chassis. Multiple levels were introduced above the chassis, as to consider the range of vision of sensor inputs as well as special crowding. The requirements for a successful representation are as follows:

- Three-dimensional visual rendering of the vehicle depicting all major components
- To-scale representation of components added to chassis
- Relative weight considered of each component to center of mass of vehicle
- Three-dimensional representation of new center of mass

5.3.3.1. Data Collection

The initial step taken in this task was to take to required measurements of all components. In Table 26, one may observe all dimensions to be modeled. Weights were also considered for center of mass computation. In terms of processing and sensory units, The NVIDIA Jetson Launchpad's dimensions and weight were considered; however, the UNO microcontroller, GPS-14030, SparkFun 9DoF IMU, DAOKI 5 PCS IR sensor, WYPH Ultrasonic sensors, and the printed circuit board were all approximated. These dimensions and weight were considered negligible compared to that of components that occupied much more space or were too difficult to model with exact specifications. They were, however, still modeled from assumption.

Table 26: Important Values of Components

Parts / Components	Dimensions (inches)	Weight
Traxxas Chassis	22.0 x 8.0 x 3.30	2.28 kg
Aux Battery	8.0 x 5.52 x 1.25	1.26 kg
USB Hub	6.60 x 2.0 x 0.75	0.5 kg
NVIDIA Jetson	8.75 x 8.25 x 4.75	1.54 kg
WIFI Router	7.0 x 4.50 x 1.50	0.3 kg
Stereo Camera	1.25 x 6.75 x 1.20	0.16 kg
Scanse Lidar	2.50 x 2.50 x 2.40	0.2 kg

5.3.3.2. Model Design

Multiple factors had to be considered with the initial design: spatial occupancy, center of mass, sensor field of vision, and reserved space for interconnectivity between components. All components considered, the model was rearranged multiple times before coming to the initial proposal. Two additional tiers in the vertical z direction were introduced, the material and exact dimensions of which are unknown at this phase. Support beams for these tiers are envisioned to be mounted on the lateral bumpers of the chassis as well as towards the anterior chassis point. Cylinders and suspension mechanisms were best modeled from tracing the image of the chassis, but should not be taken as absolute at this phase.

The vehicle engine and several other stock components were best approximated from this image as well. Center of mass was approximated using the “*massprop*” command in AutoCAD. This command is designed to approximate the center of mass relevant to the origin. In this case, the origin was placed at the center of the anterior bumper, and at the vertical z-axis of the center of the wheel. The center of mass was aimed to be lower on the z-axis and as close to the original x-y axis as the original chassis. Center of mass is a key factor to consider for the prevention of vehicle slipping or becoming too top heavy and rotating on its side. Therefore, the heaviest objects were placed in such a way to counteract the overall displacement. The NVIDIA Jetson board was first placed directly on the chassis surface, as it is a very heavy and spatially cumbersome object. The NVIDIA board occupied nearly every bit of space on the chassis surface, leading to all other large components to be mounted above it. The second heaviest object, the auxiliary battery pack, was considered next; it was placed in such a way to not introduce positive (z-axis) pitch to the anterior. This positive pitch would

increase the likelihood of slipping as the anterior wheels would not have as much weight on the surface. The USB hub had to be placed in a manner to allow a good amount of access from all directions, as multiple components would be connecting to it. Working upward, it was of importance to allow all sensors to have an unobstructed view. This involved the stereo camera mounted in the anterior direction, and the LiDAR mounted at the top-most region of the design. Infrared was placed at the top anterior and SONAR was placed lower to the ground in both anterior and posterior regions. Finally, the additional boards such as the PCB, IMU, and the GPS module were placed in a way to allow connectivity and to avoid crowding / overheating.

Connectivity and subsequent wiring between components and modules had to be considered. By utilizing multiple levels, one may envision the connections adhering the surfaces of the tiers both on top and underneath. Because the NVIDIA board's grand special occupancy prevents wiring on the top of the chassis, the support rods to these multiple tiers were designed hollow to allow protected wiring between levels. Space was reserved around the edges of each tier and between modules to allow convenient wiring paths.

Future alterations to this concept design may arise from several factors. For example, upon constructing the physical model, it may be apparent that the center of mass needs to be altered for a desired motion characteristic, such as being able to turn at faster speeds or sharper angles. Sensors may need to be added for proper functionality, or existing sensors may require a different location. Additionally, it may be the case that components are altered entirely due to an insufficient provision to the system, or to a better option as far as special and weight metrics are concerned. Most likely, wiring and connectivity will become the culprit to repositioning components to allow a secure connection to its power source and data destination. In the following section, one may consider several figures to better visualize this design.

5.3.3.3. Initial Model Visualization

In Figure 56, the model of what has been described in the previous section can be seen. The "concept" view in AutoCAD

was used to visually render each component as a solid while displaying the outline. Table 27 details the different components. Note that the SONAR sensors are not visible here. Center of Mass is depicted by the cross of two, thin cylinders. The WIFI router was chosen to be placed upside-down to avoid the antennae obstructing the view of the LiDAR scanner. It was considered that this configuration would have no consequences of significant proportions in data transmission.

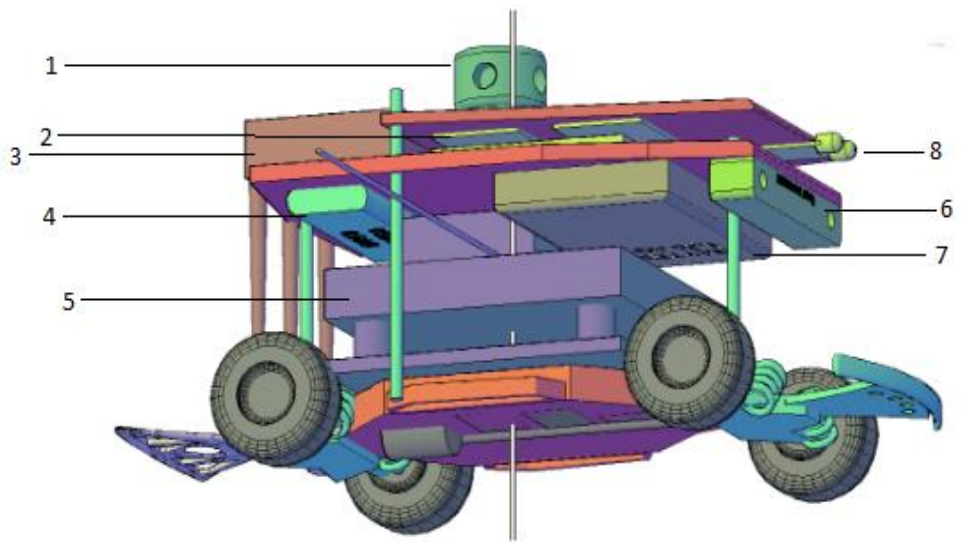


Figure 56- Component View with C.O.M.

Table 27: Locations of Components in Figure 46

Point	Component
1	LiDAR
2	GPS / IMU / PCB / UNO
3	WIFI Router
4	USB Hub
5	NVIDIA CPU
6	Stereo Camera
7	Aux Battery
8	IR Sensor

In Figure 57, the “X-RAY” view was rendered in AutoCAD. This allows one to better visualize the placement of components. One may also observe various measurements in the y-z axis. The center of mass in the x axis was 0.8 inches away from axis, and considered to be nearly negligible. With the addition of the tiers and components, however, the y and z centroid were greatly altered. However, because of inaccuracy of the NVIDIA Jetson

board component model as well as the chassis, one may imagine that the true z-axis centroid is lower to the ground.

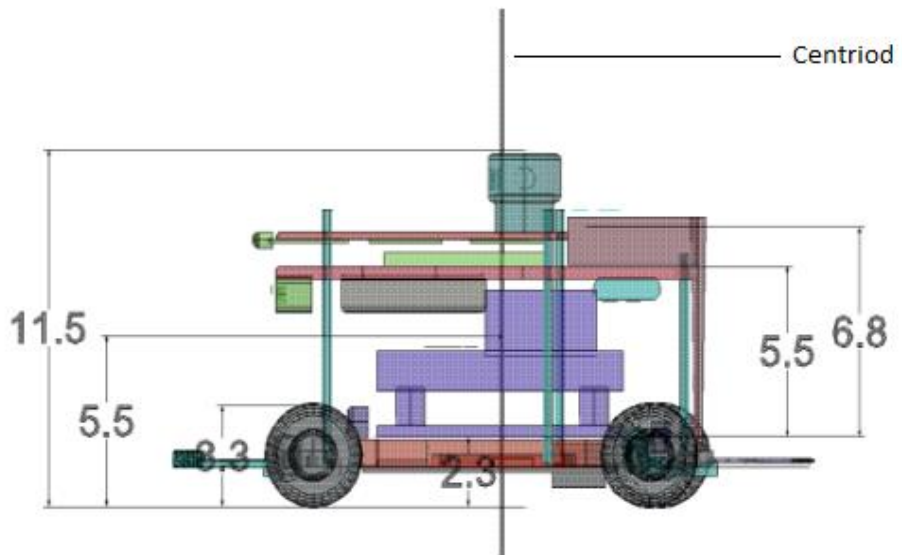


Figure 57- Profile View and Measurements (inches)

It can clearly be seen that the total height of the system greatly increased in order to house all components. This must be taken in consideration as a negative tradeoff, and may require alterations in the future to avoid tipping scenarios. Height of the lower tier was modeled to be approximated a quarter inch and the second tier to be an eighth of an inch; however, these may prove unsuitable in practice. The following Table 28 describes each distance for clarification purposes:

Table 28: Center of Mass Calculations

Measurements	Dimension
11.5	Total height from ground
5.5	Height of z-axis centroid
3.3	Wheel height from ground
2.3	Top of chassis from ground
5.5	Top of first tier from chassis
6.8	Top of second tier from chassis

5.3.3.4. Model Design Revision – Current Layout

Upon physically interacting with the Traxxas 4x4 Slash vehicle chassis, it became clear that the initial design would not be the optimal solution. Several factors contributed to the necessity of changing the schematic almost entirely. First, the Traxxas kit included a thin, see-through, plastic body in the shape of an actual vehicle. This was not initially considered as an option, and was kept in the design for aesthetic and protective purposes. Because of this body inclusion, a multitude of spatial constraints were introduced, mainly height. Holes were cut into the body to allow the stereo camera and LiDAR to have an unobstructed view. Furthermore, the vehicle's actual shape and contours were not accurately represented from the images used to create the initial proposal. Shock placement, bumper shape, wheels and suspension were able to be modeled more accurately and as a result, changed the layout significantly. Lastly, it became clear that the rear bumper was sturdy enough to support multiple items.

With the aforementioned considerations, the design changed drastically. The multiple tiers were reduced to one platform and a rear "fin". This platform was envisioned to be bolted to the chassis suspension while the fin would be attached via a hinge and lay at an angle, supported by the rear bumper. The hardware was reduced as much as possible: the casing for the USB hub was removed, the NVIDIA Jetson board was taken off of its mount while leaving the vertical spacers, the front cover of the auxiliary battery was removed, and the rear body mount rods were removed from the chassis to provide spacing for the auxiliary battery. Lastly, the WIFI router was removed entirely; it shall be locally present in the area to pick up the signal, however two antennae and a small transceiver shall be the only devices mounted directly to the vehicle. The infrared sensor, unfortunately, did not operate to the desired efficacy and shall remain out of the design, tentatively. The printed circuit board shall be fixed onto the auxiliary battery along with the GPS module.

This new configuration proved advantageous in multiple ways. First, the center of gravity could now be kept lower to avoid tipping and slipping. The streamlined fashion of the new model not only looked more professional, but also served better in the dynamics of the moving system. The body now served as a

housing to all components, this removes the likelihood of damage to the valuable sensors in a collision scenario. Clearly, tradeoffs are always present between designs; by using a minimalistic, one-tier approach, a variety of individual mounts and platforms needed to be precisely designed in order to properly integrate and support all components. These designs are discussed in the following section. The current model can be seen in Figure 58, please note that not every component is visible.

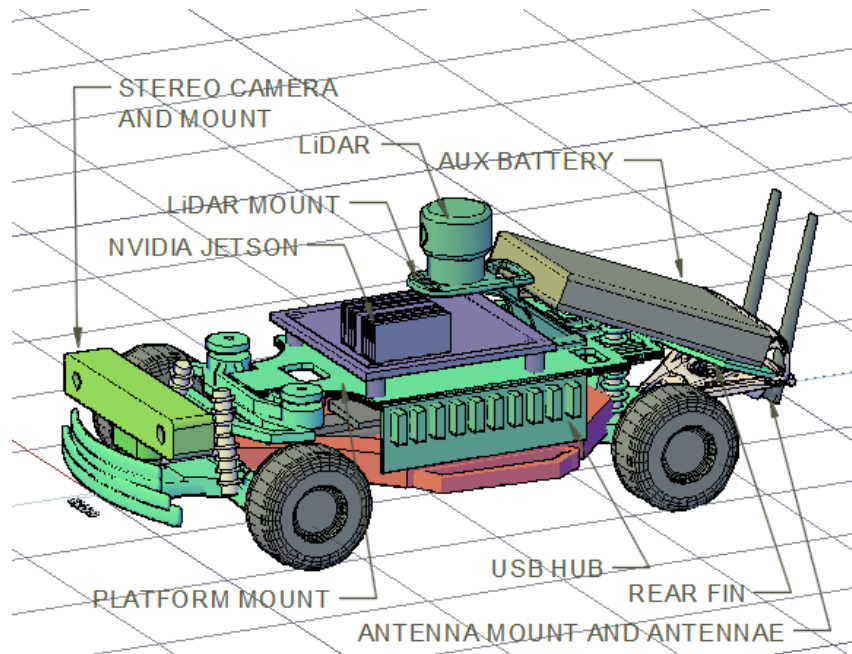


Figure 58- Design Revision, Current Layout

5.3.3.4.1. Supportive Structure Design

To design the current (revised) model, it was necessary to outfit the chassis with supportive structures for all sensory and hardware components. This is done in order to both protect and to allow all methods of sensory and computational power to operate independently, while abiding by spatial and temperature constraints. To accomplish the desired configuration, many additional designs would be required.

Upon initial observation of the chassis, it became clear that many structures would be required in order to properly brace all additional components. AutoCAD software was found to be the most applicable method of design in which to cater to specific

requirements. Objects were either laser cut or 3D printed in order to adhere to the very specific dimensions and to avoid the necessity of adhesives. All constructs were created in the UCF T.I. Innovation Laboratory and designed to uniquely fit the needs of this project.

The methods of the AutoCAD technique shall not be discussed, as they of preference to the designer. However, it is worth noting that the process of design involved precise measurements and modeling to-scale. Grid snapping and mirror functions in AutoCAD were utilized to ensure a symmetric and accurate product. Two-dimensional designs were used for objects that were laser cut, while three-dimensional models were used for all others. It was importance to consider the orientation of the object when 3D printing, as the structural integrity could be compromised if printed from an arbitrary start point. For the more complex designs, such as the LiDAR mount, a mold was printed simultaneously around the design to ensure that the structure did not collapse during the fabrication process. Resolution of the 3D printer had to be taken into consideration, as well as object shape. Below is a list of our requirements for our mounts:

- Platform mount must be bolted to chassis
- All cables and wiring must not be obstructed by mounts
- All volumes kept under five cubic inches in 3D printing
- No adhesives to keep any mounts together

5.3.3.4.2. Platform and Mount Design

The initial, and possibly the most important, design was that of the platform. This platform would serve the purpose of mounting the main computational hardware such as the NVIDIA Jetson board. It also served sufficient space to allow mounting of the USB hub and the Arduino UNO board. Several revisions of this mount were necessary to obtain the proper spacing. The platform was the only mount to adhere directly to the chassis. For this reason, the design specifics needed to fit with the bolt placements already in place with the Traxxas RC chassis product. Multiple factors also came into play when designing the platform: spacing availability for pulling cable, grills for heat convection, and avoiding the wheels at maximum displacement in all three axial directions.

The initial templates were cut into wood of 3/16" thickness. While this material served the structural requirements, it was surmised that better options were present. The final material was chosen to be Plexiglas for a balance of rigidity, flexibility, and heat conduction. In addition to these factors, Plexiglas was deemed a proper fit for the simple reason of the ability to see through the substrate. This visibility factor provides the designer with a great convenience of being able to see what is happening throughout the system's levels. The thickness of the Plexiglas was chosen to be 5mm to provide stability and ease of laser cutting. Two-dimensional laser cutting was chosen as the most accurate method for this design.

Through a sequence of adjustments, the final product was accepted to adhere to the requirements. Radial anterior edges were introduced as to not collide with the tires if the vehicle were to experience maximum compression of the suspensions system. A ventilation grill was placed between the top-mounted Jetson board and the Arduino board, mounted underneath. The rear fin was eventually mounted with a hinge to the rear of the platform.

A rear "fin" was designed in order to support the bottom of the auxiliary battery, as it lay at an angle from the rear bumper. This fin generally followed the shape of the rear bumper to keep from protruding too much, laterally. A hinge was used to fasten the fin to the rear of the platform. Ventilation grills were placed down the fin to allow heat convection from the auxiliary battery. Perhaps the primary purpose of the fin, however, is to allow the battery to be secured via straps. Without fastening the battery, it would have the ability to slide which is extremely undesirable in a collision or turning scenario – considering the heavy weight of the object. By designing "buckles", one may envision that a strap be fed through the buckle on each end of the fin and tightened around the battery to prevent any slipping. The fin was mounted directly to the rear bumper by drilling holes in the material.

Holes were placed in platform for the shocks have ample room through critical moments (6) and the back of the platform extended for a front strap to secure the battery-pack. The rear fin (7) will be secured to the back bumper and will also have a strap

for the battery. Additional openings, in the front and rear, for the cables to run through (4), holes for the ventilation of heat (5), and holes for battery straps to loop through (8) were added to the platform design. When placing the ventilation holes, we had to avoid the areas where the mounting of hardware would take place. The NVIDIA CPU (2) will be mounted to the top of the platform while the Arduino UNO (1) and USB HUB (3) will be secured to the underside. The current design is shown below in Figure 59.

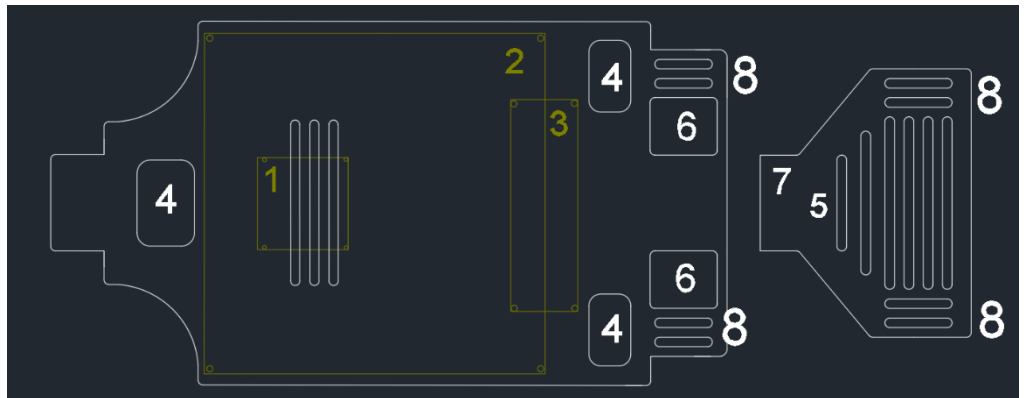


Figure 59- AutoCAD Platform Design

The LiDAR camera needed to be positioned in such a way that it would have an unobstructed field of view while being immune to vibration. While all vibration is unavoidable, the mount needed to possess a minimum torque moment around the axes of rotation. In addition to these factors, the mount's dimensions were designed very precisely to fit underneath the auxiliary battery while jutting forward, toward the anterior of the vehicle and the hole cut for the LiDAR to protrude through the roof of the body. The placement of this mount is shown in Figure 60.

To minimize the torque moment as vibrations travelled through the mount, arches were put in place in three positions. The first arch will support the neck of the mount. The top surface of the mount was created with a rounded edge to better fit the circular nature of the LiDAR camera. By fastening the neck to the center of the top surface at an angle, two more arches were included to prevent the top surface from bending. Further considerations may include creating arms to fasten the top surface to any existing bolts on the NVIDIA Jetson board or to the platform itself.

Wiring was also taken into consideration, as the LiDAR camera's wires needed to feed through the mount itself. The neck of the mount was designed to have a hole as well to feed cables through to the auxiliary battery. The base was designed to have a width and length to support bolts to the platform mount. Volume considerations were kept less than five cubic inches to save on materials in 3D printing. One may observe the LiDAR mount in the following Figure 61.

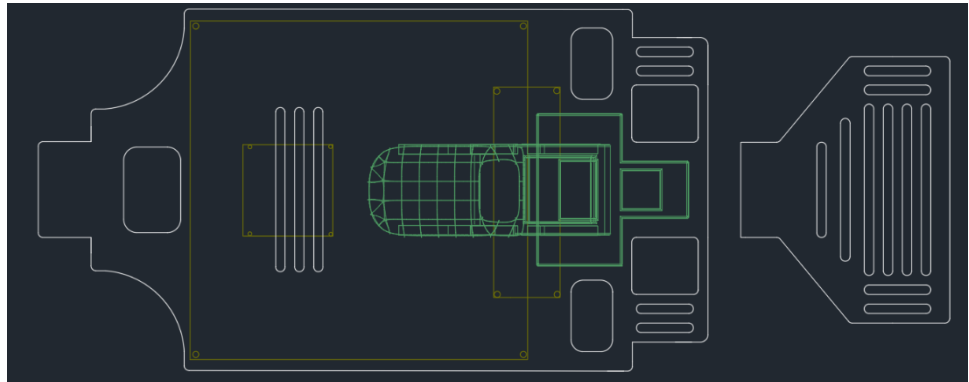


Figure 60- LiDAR Mount Placement

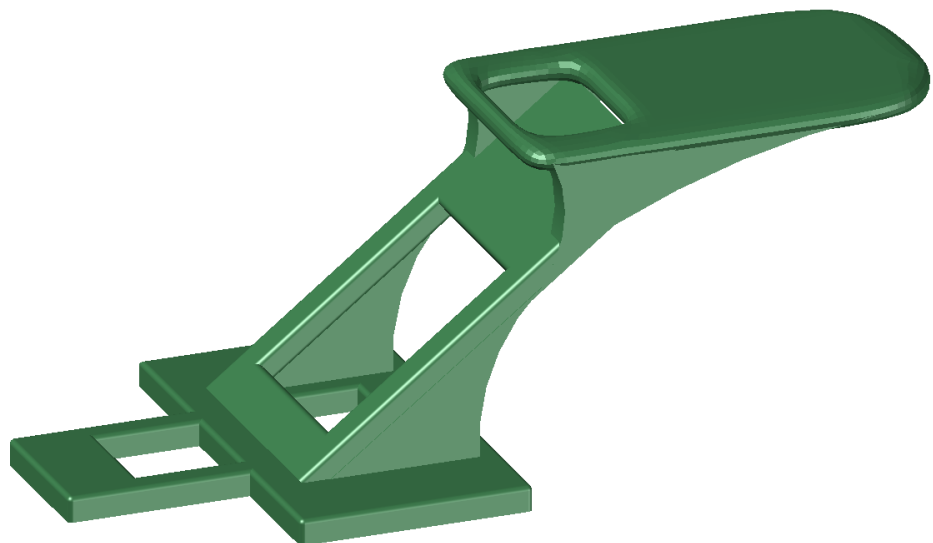


Figure 61- LiDAR Mount

The stereo camera was chosen to be placed at the front-most region of the vehicle to avoid interference with LiDAR vision. By cutting a hole in the anterior of the body, the camera would fit easily through. Originally, it was intended that an angular

connection be fastened to the platform mount; however, it was then discovered that by utilizing a convenient “hole” in the Traxxas bumper, one could fit a mount vertically from the bumper. This particular piece of the bumper is an intermediary piece from the chassis, and therefore does not exhibit bending in a collision scenario. Because of this trait, the camera could be safely mounted from this point.

To fit in this long, rounded hole, the mount needed to be designed with great precision in the x and y plane. To avoid sanding or an inaccurate measurement, it was decided that tapering the neck would allow the mount to fit tightly within the hole. The “head” of the mount draped over the hole to define the maximum boundary and to avoid sliding through the hole. Once the mount was fit into the hole, bolts were drilled into the sides to ensure a permanent placement. Further considerations may include an additional piece extending laterally to accommodate any tilt that the long stereo camera might experience. Figure 62, shown below, depicts this camera mount:

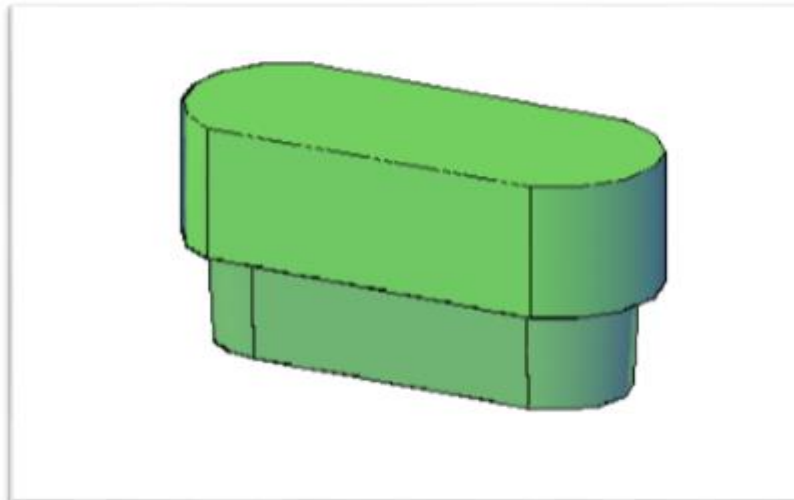


Figure 62- Stereo Camera Mount

For the placement of the antennae for the devices to communicate with one another, it was determined that they best be positioned toward the rear of the vehicle. This serves various purposes in both collision scenarios and the avoidance of obstructing camera or LiDAR fields of vision. This mount was to

be designed at an angle to adhere to the rear bumper, which tends to an angle of about forty degrees.

Two cylinder-shaped supports were designed to protrude from the base in order to house each antenna as it was screwed in. Holes were measured with precision and consideration taken to allow the thin cables from each antenna to pass through the base of this mount. The mount itself was to be bolted directly to the rear bumper by use of drilling through the material. For aesthetic purposes, the UCF Knight's logo was traced and etched into the base of the mount. It was mirrored on the mount in the hopes of an LED light being placed behind the mount itself to shine the "Pegasus" onto the floor as the car drives. This mount is depicted in Figure 63:



Figure 63- Antenna Mount

In placing the battery as an overlay above the NVIDIA Jetson board, it required that the stock body mounts in the rear be removed. These mounts are intended to support the clear, plastic body and secure it to the chassis. A new body mount had to be constructed to properly fasten the body in both the front and the rear. These designs mimicked closely Traxxas' design; simply, a thin cylinder with a small hole for a metal clip. This cylinder will protrude through the top of the body until the body meets the conical brace. The protruding portion has a 2mm hole for the stock metal clip to prevent the body from sliding, but also allow it to be removed. A base was designed for this rod mount, as it would be bolted directly to the platform. This design is shown in Figure 64 below.



Figure 64- AutoCAD Designed Body Mount

As the product was assembled with the aforementioned structural support mounts, it became evident that the USB hub would no longer be able to remain in the initial position. This issue was mainly due to the connectivity factor of nearly all other modules to this destination. USB cables are inherently bulky, and therefore needed ample room to avoid damaging the cable. Furthermore, rearranging these cables at any point was an arduous task due to the spatial constraint. The optimal location for the USB hub was found to be on the lateral edge of the Plexiglas platform; thus, a mount was needed. While this particular design is not very complex, the dimensions were very sensitive to the USB hub's surroundings. An "L" bracket was considered for purchase, but for the goal of accuracy and professionalism, the bracket was designed to be a 3D print, and is shown in the Figure 65.

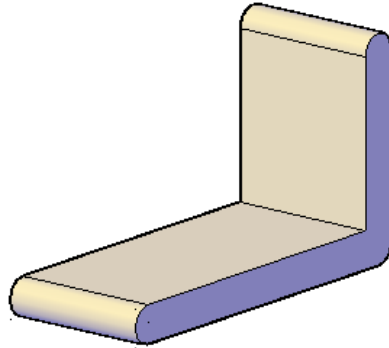


Figure 65- USB Hub Bracket Mount

All mounts were 3D printed, attached to the laser-cut platform and the car's front and rear bumpers. The implementation of these mounts was seamless due to the attention to detail and drawing to scale during the design process on AutoCAD software. The primary mounts can be seen in Figure 66 below (USB mount and body mount not shown). The antenna mount can be seen on the far left and above it the rear fin. The platform can be seen and above it the LiDAR mount. On the right hand side the stereo camera mount can be observed attached to the front bumper.

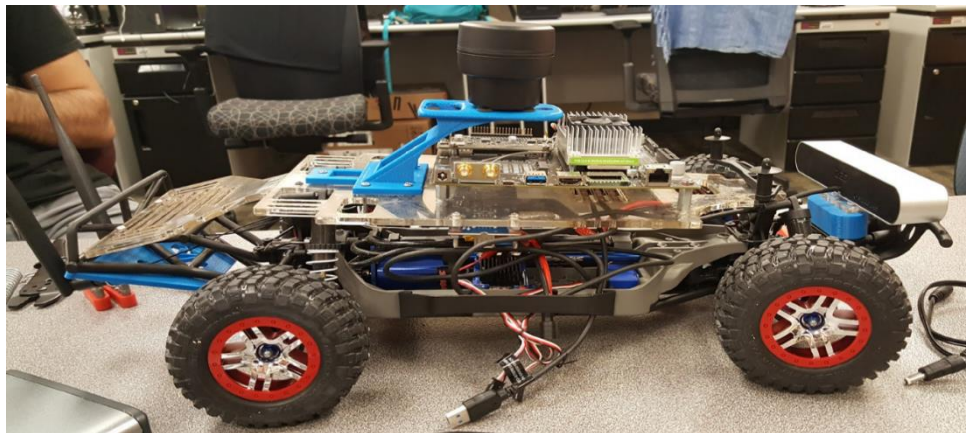


Figure 66- Final fabricated mounts and platform installed to the car

5.3.4. SSH Connection

In order to progress to the self-driving car, the Jetson must be removed from the ethernet connection to the wall. To achieve this, SSH was setup to connect to the Jetson via a terminal

through an ad-hoc network. This would allow the car to be controlled from a remote computer which could act as a central computer to connect several cars together.

This was achieved by installing openssh server on the Jetson and host computer, then setting up the ad-hoc router to forward on port 22 (ssh) in order to connect to the Jetson via its IP address.

5.3.5. Remote Connection

While SSH would be sufficient for controlling the Jetson via the terminal, it would also prove useful to be able to have full control of the Jetson using a remote connection. The built-in program “Remmina” will be used to connect to the Jetson.

Doing this gives full control over the car by a remote desktop, which will be useful for running tests and updating the software without having to connect it to a screen via an HDMI cable.

5.4. PCB Details

PCB design is a vital tool for Electrical Engineering students to master and the one of the purposes of senior design is to help students develop this skill. In an effort to build these skills, meet our Senior Design requirements, and further improve this project for our sponsors, we will design and assemble a Printed Circuit Board to perform a variety of beneficial advantages. This section will cover software, the basics of the manufacturing process, and specifications.

5.4.1. PCB Manufacturing Process

With the assigned task of designing and ordering a printed circuit board one can see that the process is not a simple one. Previous sections have covered the software that goes into the design but how exactly are these PCBs made? This section will provide insight into the chemical Etching process by which printed circuit boards are manufactured.

Etching is the terminology used for the purposeful subtraction of undesired copper from a prefabricated laminate material. The copper material is used as the electrical traces on the board by

which the electrical signals are able to travel between components. The copper is first applied to a nonconductive laminate material, phenolic cotton paper, epoxy and many other materials can be used for the base laminate. The copper then is covered by a mask that obstructs the etchant's ability to remove or etch away the copper material. Common acids that are used in this process are ammonium persulfate and ferric chloride. The mask is comes from a specific design file and exported to machines for application. The files can be made using software available from a variety of companies on the market as long as they are compatible with the manufacturer of your PCB. There are also many options for the application process of the masks.

One way is by screening the copper covered laminate base material with tin and lead which does not react with the acid leaving bare copper to chemically react and disappear. Another process available is to use an advanced printer called a plotter. The plotter printers administer photoreactive ink on two different layers. The first layer is for the etching process and uses black ink to denote the circuit traces and keep them from being etched away. The second surface layer is a solder mask that will be discussed in detail in the following paragraphs. These two layers work harmoniously to perfectly map out a printed circuit board design for a more streamlined manufacturing process.

The mask is then removed by using a wire brush to scratch away the thin layers of tin and lead. After the mask is removed the drilling process can begin. Drilling holes in the PCB must be done using a drill capable of high RPMs and typically uses tungsten carbide coated drill-bits or laser cutting as to not completely destroy the copper traces. With the size of these holes and the high possibility of error automated drills are used to increase accuracy. The types of through-holes are just one way to mount components to the circuit board, in recent years surface mount integrated circuits and transistors have been developed. These components are much smaller and attached to small contact pads through soldering instead of punching through the board material.

After pads or through holes for components are applied measures must also be taken to keep the copper from oxidizing and to increase its conductivity for the mounting of components.

To accomplish this solder, silver, nickel or gold can be coated on top of the copper traces, however any anti-corrosive metal will work just as fine. To keep areas on the board clear that are not meant for solder and to avoid the shorting of components by excessive application of solder resistant coating can often be applied. This resistive material is called a solder mask and is achieved through photo-sensitive coating. The layer is applied and put under light to become reactive. Once it is dry it is rinsed and the areas where the coating is undesired are carefully scrubbed and rinsed again.

The final processing steps include the addition of text, insignia or important instructions to the board through printing or silk screening. After this step the board is coated and cured and ready for testing. There are many ways to electrically test these boards but by far one of the best is called a "Flying Probe test", which applies incredibly fast mechanical probes to examine the performance of all net and bus systems. Another method is bare board testing which employs resistive or capacitive theory to measure inaccuracies. The capacitive method is done by charging net traces and validating using probes. The resistive approach to bare board testing follows the principal that metal elements have low resistivity, and to confirm this meters are used to check expected resistance versus the measured values. The resistivity of a metal is dependent on: its length, thickness and chemical properties; all of these values are known from the design files and using software can be calculated for each branch, net or bus of circuit traces.

5.4.2. PCB Manufacturer Selection

The selection of a PCB manufacturer will determine the overall effectiveness of our board and could impact our timeline. Selection will be based on price and turnaround time for the manufacturer to build and ship us our PCB. All manufacturers will be based in the United States to decrease shipping costs, and more importantly, save time. Overseas manufacturers will take approximately 2 months to ship via boat from Asia, which is dangerous if an error occurs in our design and we need to purchase a new one. The manufacturers considered were recommended by various other ECE and EE students in senior design 2 who had already received their PCB's.

5.4.2.1. 4PCB

The first recommended manufacturer we looked into was 4PCB. This company is based in the United States and by choosing them we would decrease our shipping cost and time. They also provide a wide variety of advanced circuit board design methods such as laser or bit drilled micro-vias, cavity boards, addition of heavy copper (up to 20oz of copper), boards with up to 40 layers, microwave and RF boards. The reliability of this company can be seen by its list of clientele which includes companies within the medical, military and commercial sectors of business. The company also provides PCB Artist, a free software for designing printed circuit boards, which will ensure compatible files types with all their machines.

A standard PCB order from 4PCB allow for up to 10 layers, FR-4 laminate, Lead free HAL layering, up to 2oz outer copper weight, and a size of 5 x 5 mils which can be finished and shipped within 5 days. However a custom PCB package includes access to more materials in laminate, and solder layers, higher dimensions of design, up to 40 layers and many other benefits but depending on the level of complexity these design can have a 4 week design and shipping process. This company uses the bare board and flying probe electrical testing mentioned above for PCB orders to ensure functionality. Among these electrical tests is a massive list of other electrical tests available for checking current flow, impedance, continuity, electrical leakage, field-effects, and shorts. 4PCB also provides customers with a great online order-tracking system so expected shipping dates can be counted upon and monitored.

5.4.2.2. OSH Park

The second recommended manufacturer was Osh Park. One of OSH Park's biggest selling points is free shipping anywhere in the world from the United States, and a clear set of design rules they use for printing and producing PCBs. They provide 2 and 4 layer PCB's at varying costs based on shipping times and layers. All OSH Park PCB's come with a purple mask over the bare copper and an ENIG finish.

The most expedited turnaround time for a 2 layer board has a cost of \$10 per square inch and takes only 5 business days to ship. The board maintains a 63 millimeter thickness and has a copper weight of 1 oz. The minimum design standards for a 2-layer PCB are a 6 mil trace clearance, a 6 mil trace width, a 10 mil drill size, and a 5 mil annular ring.

The boards can also be made with 4 layers and have easier design constraints concerning annular ring size, trace width, and trace clearance at \$10 per square inch. OSH Park 4 layer boards also have a FR408 Substrate. Unfortunately, the PCB's have a minimum turnaround time of 9 calendar days, and can take up to a month depending on the chosen service.

5.4.2.3. PCB Manufacturer Summary

In consideration of our budget and time constraints a decision for the circuit board manufacturer was discussed by between group members. It was decided that heightened cost for decreased development time would be a more ideal trade off due to possible redesigns, damage or errors that could occur in the next phase of this project.

The group decided that 4PCB would be the best option for this part of the project. Their attention to detail and wide range of available services sets them apart from other services. Their extensive electrical testing procedures will ensure the reliability and functionality of our design and will decrease the likelihood of a future redesign. 4PCB has options for 24 hour design which will suit our needs, and because our PCB will not require advanced manufacturing techniques we will be able to choose this expedited option. The comparative cost of 4PCB versus other options was not higher, especially because we will not require complex design methods. The PCB design will be discussed and an initial prototype will be shown in a later section.

5.4.3. PCB Design Software Selection

In an effort to model the PCB accurately and make design process go as smooth as possible multiple programs available were researched. Important parameters that were examined and

considered were costs of manufacturing and software, customer reviews, and software features.

5.4.3.1. EAGLE

EAGLE is an acronym that stands for Easily Applicable Graphical Layout Editor that was developed by CADsoft computer GmbH that was bought by Autodesk in 2016. EAGLE is an electronic design automation software useful for designing circuit schematics and printed circuit board design.

EAGLE has some great features that make it easy to use for beginners and even professionals. One such tool that we would find useful is called auto-routing, this feature automatically connects traces on a PCB design for connections specified by the circuit schematic created by the user. This program multiple files types for layout files, and drill files that are used by most PCB fabrication companies. Even if a company does not specifically use EAGLE (.BRD) file types this program is so widely used that companies who do not have conversion software that accepts this file type.

This program uses multiple windows and a menu system for the creation and editing of files and can be controlled through mouse, keyboard or the use of an embedded coding window. EAGLE makes their software available to students for free, which is a huge positive and will play into the decision. The free student version allows a PCB of 4 square-meters to be made with up to 6 layers, this is a value of up to \$700 a year.

Other features include checks and failsafe programming to make sure there are no fatal errors in your design, and real-time changes, so if a change is made to a schematic the PCB layout will reflect changes as soon as they are made. The PCB design software has intuitive alignment tools and obstacle avoidance trace routing so moving components about is made easy and will never interfere or cross paths. Also included within this software is the ability to design 3-Dimensional renderings of your finished PCB with components on board.

The online community for EAGLE supplies video and other tutorial materials for beginners, this support would be extremely

valuable to our team. Other valuable resources provided by this program are the libraries of the companies that use EAGLE to design their products. For this purposes of this project the Arduino, SparkFun and Adafruit libraries will provide valuable schematics and insight in the integration of these products into our PCB design. However, John Teel a circuit design software reviewer describes eagle as difficult to use and recommended days to weeks of learning before one can become proficient enough to design. Other reviewers online do agree that EAGLE can be quite difficult and frustrating however through use of online tools like youtube tutorials this program can be learned decently fast.

5.4.3.2. Altium

The first version of Altium was first created in 1985 and continues to be a user favorite in PCB design software. Altium extends to many different market places from automotive, aerospace and defense, and life sciences, to consumer electronics. Perhaps the top rated software in the field with hopes becoming an industry leader by 2020, this top of the line software does not come without a steep price of near seven-thousand dollars. Some of the tools included in this package are Altium Designers, CircuitStudio, CircuitMaker, and NEXUS among other programs for embedded software, data management and integration services. The programs relevant to this project will be discussed in detail below.

Altium Designer is a complete schematic and layout program packed with features the company hopes will bring back innovation and make Engineer's jobs feel less mundane. One such feature is a Unified environment that connects schematics, PCB layout and documentation that promotes a seamless exchange of information between all the various components of the design process. Intelligent auto-routing features help to reduce error and assistance programs to help your design be the best that it can be. Other features include: integrated tasking pin mapper, streamlined design rule editor, component placement system, among many other useful tools.

CircuitStudio is Altium's professional PCB design software made for entry level users. This program from Altium has a much lower

price point of almost \$700. User reviews say that is extremely easy to use and intuitive. This program will import EAGLE files to enable users to easily switch over to a newer software. CircuitStudio comes equipped with tutorials in the program to help beginners quickly become proficient circuit designers. Features included in this program are: Component creation and finding equivalent replacement components, XSPICE simulation, Multi-Sheet Design structures, Pin Swapping, Creation of unique board shapes, and generation of outputs.

Altium does have a free, community ran and open source program called CircuitMaker with no limitations. CircuitMaker allows for full PCB design with 16 signals and 16 layers with no restrictions on PCB size that can be made. This program is community based so users can get help from hobbyists and professionals with their design concerns. With expansive component libraries this program searching by company and model number for desired parts or even the creation of your own is incredibly easy.

Other features of this program are Push-N-Shove Routing, Topological AutoRouter, DRC/DFM Validated Outputs, and the ability to import designs from other tools. User reviews seem to reflect highly on this free Altium program, from tinkerers to Engineers it is agreed that this program is fresh and much needed in the EDA community. This program seems to be an excellent fit for our budget and with community sourced data and resources available it would help reduce possible errors and improve our overall design and efficiency.

The sponsors of the project requested that we use a well-known and reliable software for the design of our board so for this part of the project the final decision is to use the EAGLE software. The availability of tutorials and resources from other online communities will aid our team in maneuvering this often-complex software. The versatility of EAGLE output files will also enable us to shop around for the most cost-effective producer to manufacture our design.

5.4.2.3. KiCad EDA

KiCad is a free, cross-platform open source Electronics Design Automation Suite that was originally developed in 1992 by Jean-Pierre Charras. KiCad now utilizes help from developers at the European Organization for Nuclear Research, otherwise known as CERN to create more advanced tools for its software design. Other contributors to the success of KiCad include Raspberry Pi, Arduino, and Digi-Key.

KiCad has many component libraries available and for a project where something cannot be found the software can import files from more popular programs like EAGLE. KiCad's schematic capture tool is called Eeschema, and boasts that it is limitless and easy to use by breaking up large schematics in hierarchical sub-sheets. The schematic software has Electronic Rules Check (ERC), and component symbols that are coupled with footprints for the PCB design layout tools. Eeschema allows for the export of schematics from popular programs like Pspice, and Cadstar. KiCad brags that its libraries are extensive and constantly updated to account for the newest trends and products.

The PCB design of KiCad is managed by something that they called PcbNew. Similar to Eagle this software automatically avoids obstacles and pushes other traces out of the way when you are drawing new traces between components. Length tuning in PcbNew allows users to trim the length of traces and shrink the circuits to increase overall speed of the design. PcbNew supports 32 copper and technical layers of PCB design and a maximum size of 2.14 square-meters all with nanometers of precision. User reviews say that KiCad is easier to use than EAGLE but still has some negative attributes. Recurring user complaints are that the component list is small, and when making a new component in the program it must be done twice which can be time consuming.

Other problems reviewers remarked on was the outdated user interface that feels outdated and menu options that are out of place or not organized intuitively. The downside to free software is that there is little or slow maintenance done, users complain that the stable version is extremely old and when downloading updates code can be dysfunctional for weeks sometimes if it is being modified by KiCad software technicians. This all taken into account we move to another program.

5.4.4. PCB Design

Ideas for the functionality of our design came from discussions with our Senior Design advisors, project sponsors and brainstorming sessions within our group itself. Our PCB Design as stated in the Requirement Specifications will include the following features:

- LCD Display for data display (Speed, Direction, Heading, and Car battery level)
- Battery Level Sensing Circuit
- LEDs for headlight, taillight and turning signal simulation
- Additional LEDs for car state management
- Temperature Sensor and Fan to monitor and regulate heat

The PCB will utilize an ATmega328P Integrated Circuits (IC) that will be pre-programmed using the Arduino UNO Rev3 development microcontroller. A 16 x 2 LCD array will be used to display the data listed above in an effort to show imperative data during the testing phases. The LCD will require a voltage regulator to supply a dedicated 5VDC to utilize the full brightness of the backlight and also to display all of the data. In initial testing of the LCD data display would only happen with a dedicated 5V supply. To avoid any possible issues from maximum current draining, voltage drops or power surging in a failure situation the regulator will be a necessary component to maintain integrity of the supply voltage.

IC pins will be used to drive the LEDs, control the temperature through a sensor and turn on a 2 x 2 fan. The PCB board will receive the cars autonomous driving state (forward, back, left turn and right turn) from the NVIDIA controller and reflect these states on the headlights and taillights. When the car is in a forward driving state the front LEDs will go high, when the car is stopped all the LEDs will light up, when the car is turning in either direction the LEDs on that direction will blink similar to turning signals seen on the road. This will help to indicate errors in the drive state of the car for maintenance, will make our vehicle function aesthetically to a real-world car on the road and the programming could be applied to a full-size car project in the future.

The temperature sensor will be located on the opposite side of where the NVIDIA CPU built-in fan is located. Regulating heat will be vital to the protection and functionality of our components. If the temperature sensor is reading outside the acceptable operating range of the NVIDIA or the portable battery pack (for the NVIDIA over 80 degrees Celsius) then a fan will be triggered on. An additional temperature sensor will be placed near the car's battery pack to ensure that it is functioning correctly. The PCB will be placed on the ar end of the car, to shield the PCB from heat a heat sink will be installed. To power our PCB, we will be using the 5V supplied by our USB hub via a customized power cable that will connect to the female barrel power connector soldered to the board. This type of power connector will be easier solder than a micro or mini USB female power connector.

For the car battery voltage monitor, the output voltage of the car's battery will be divided using a resistor network to a more manageable level. This small voltage will be read by the Arduino Board and through a code calculate the car battery voltage and send that data to the LCD display. If the battery falls below necessary levels and needs to be charged the LCD will display a prompt to the user through the display. The schematic for this circuit is shown in Figure 67.

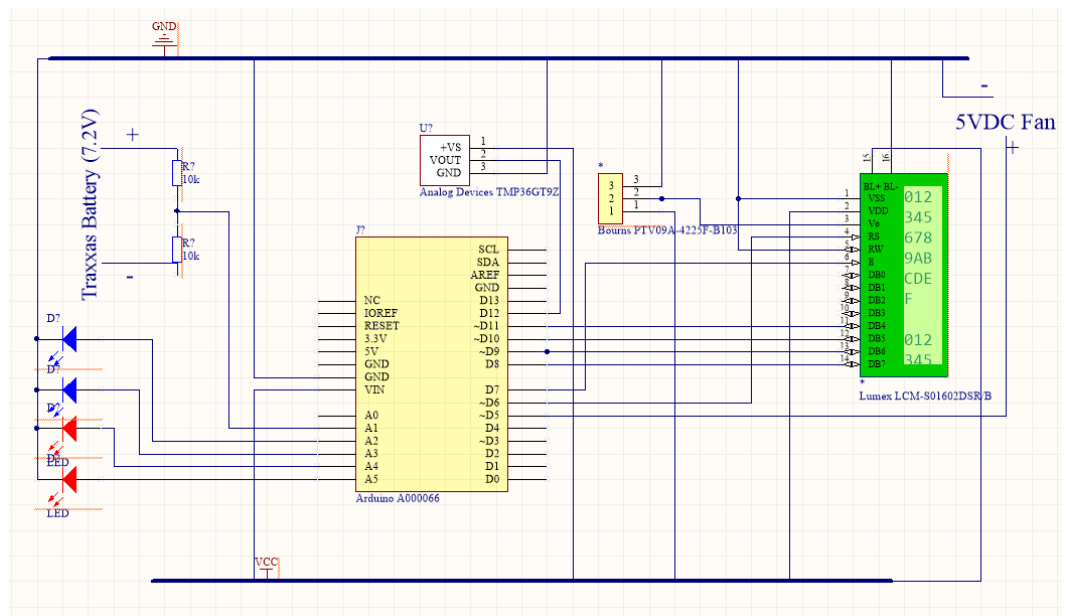


Figure 67- PCB Circuit Schematic

6. Administration

While meeting the engineering requirements set by our sponsor is the primary goal of this project, the steps necessary to achieve this must be laid out and then systematically completed as fast as possible. In this section, we discuss the schedules we created, generalized specialties and responsibilities that were developed as the project went on, and the schedule for the senior design course to track documentation, and one for tasks our sponsor has laid out for us. We also cover the budget we have to perform under to achieve our end goal.

6.1. Generalized Responsibilities

As the project progressed and the team learned more about the little intricacies and nuances of the project, each team member decided to take on a specialized role based on prior experience and the needs of our sponsor. In this section we detail the responsibilities of each design team member

6.1.1. Bruce Hardy

Bruce's main area of experience is in circuit design and because of this took on the responsibility of the Printed Circuit Board design and hardware design. In an effort to learn more about the various components being connected Bruce also became responsible for product research and ordering. The selection of the products enabled Bruce to ensure the cross compatibility of these components for a unified and working model.

Constraints that had to be considered were, logic voltages, supply voltages, power consumption, current draw, size, data and image processing statistics, reliability, dynamic ranges, among many other specifications. The ordering of products had to be done before the end of the semester so the time line for research and purchasing of products had to fit within that window.

Other products that needed to be ordered was heat shrink tubing, custom power connectors, customized DuPont connectors for organized and correct data connections between components.

In addition to product research and ordering Bruce assisted Tyler in the design process of the mounting hardware, offering ideas and taking measurements. Bruce was the person who ensured the prompt CNC laser cutting and 3-Dimensional printing of the platform and sensor mounts respectively. This was and will continue to be important due to the high volume of students from all areas in the school utilizing the Texas Instruments Innovation lab within the Engineering building.

6.1.2. Tyler Thompson

Having experience in AutoCAD design and electronics, Tyler's contribution was aligned with the electromechanical aspect of the project. As the needs for hardware mounts of very specific dimensions arose while building the car, 3D printed models and laser-cut platforms were deemed a better design than store-bought parts.

In addition to drafting and fabrication, OpenCV computer vision programs were Tyler's responsibility. Various programs were written for object-oriented detection software using Histogram of Oriented Gradients as well as Dense Optical Flow for motion detection.

6.1.3. Christian Theriot

Christian's main responsibilities involved software design and integrating software with hardware. Nitish directed him to learn ROS, which proved useful in designing a simple overall software architecture. When the Jetson TX2 arrived, he was also assigned to learn how to use it and integrate ROS on its Linux environment, considering he had prior experience with Linux and ROS.

When it was discovered that the GPIO interface on the Jetson would be involved, Christian advised our sponsors to use the USB-serial connection in the meantime. Considering the lack of documentation for the TX2, this decision saved time for the whole

team to focus on designing the PCB and integrating ROS to communicate on the serial port.

6.1.4. Eduardo Linares

Eduardo's main responsibilities in this project were a combination of administrative, mechanics, localization, and sensor testing. Localization responsibilities developed as a result of his knowledge of localization. When Dr. Fallah and Nitish first spoke with the team, they gave avenues for learning localization algorithms like the Kalman Filter and Particle Filter through numerous websites, like Udacity. Eduardo took on this responsibility because other programming responsibilities concerning ROS and OpenCV were being given to Tyler and Christian, respectively.

In terms of mechanical design and sensor testing, Eduardo led the testing of the hall sensors and steering controls alongside Bruce and Christian. He developed the preliminary Arduino code for measuring motor RPMs and changing the steering angle. All testing setups pictured in the document were also designed by him concerning steering, ultrasonic sensors, and hall sensor implementation. All spring modifications were researched and implemented by him as well.

Administrative responsibilities included the formatting and final editing of the document and making sure the schedule was adhered to by all members. Coordinating research topics and designating tasks was also facilitated by him alongside the sponsor.

6.2. Budgeting and Finance

The primary goal is to design the vehicle at a cost of no greater than \$2500 per car and to build at least two vehicles before the project showcase. The cost analysis is displayed below in Table 29.

Our budget only contains the preliminary major components we identified to build the final product. Other sensors may be purchased for testing purposes, but not necessarily included in the final design. If the additional cost of these sensors becomes

significant, a second table will be added specifying what was used as test equipment, and the sensors that are used in the final design will be added to the first table.

Table 29: Cost Analysis Table

Item	Cost
Traxxas Slash 4x4 Platinum	\$429
Traxxas Battery Pack and Charger	\$99
NVIDIA Jetson TX2 Development Board	\$299
Elegoo UNO R3 ATmega328p Board	\$11
ZED 2K Stereo Camera	\$449
(10) WYPH Ultrasonic Module	\$15
(5) DAOKI IR Module	\$5
GPS Module	\$13
LIDAR Scanse Sweep Sensor	\$349
SparkFun 9DoF IMU Breakout	\$25
TP-Link N450 Wireless Router	\$30
(10) Ethernet Cable	\$13
7 Port USB Hub	\$27
50,000mAh Auxiliary Battery Pack	\$136
0.22in 18 inx24 in Plexiglass sheet	\$25

6.3. Milestones and Timeline

Due to the large amount of responsibilities that we have been given by both our sponsors and the requirements for the senior design course, multiple separate schedules were created for ease of viewing and to separate the requirements of our course and our sponsors. The senior design timeline focuses on paper submissions and required meetings with Dr. Richie and Dr. Wei. The sponsor timeline focuses on tasks to make the vehicle functional.

6.3.1. Senior Design Timeline

The following schedule in Table 30 lists the dates for the Senior Design timeline and the team’s ideal approach to meeting these deadlines efficiently. The schedule details the paper deadlines for the senior design course, internal deadlines set by the group, and the required meetings with Dr. Lei Wei and Dr. Samuel Richie. Other meetings may occur but not necessarily appear on the schedule because they may occur as a result of issues with paper formatting of the paper.

Each paper deadline at least 2 days in advance to provide proper time to compile, edit, and format the work done by the senior

design time. The team also established internal deadlines between due dates to keep pace and prevent the project from falling off schedule.

Part orders are slated to occur before the end of the semester, but due to the sponsor deadlines, are very likely to be finished before the end of October or very early November. The final PCB design is slated for the end of the semester tentatively and covers the design of the circuit schematic, not the physical PCB itself. The physical PCB will be designed between semesters to allow for a more focused approach and to prevent other school-related stresses from interfering.

Table 30: Senior Design Schedule

Deliverable	Due Date	Estimated Time to Completion	Steps to Complete
Divide and Conquer 7-10 Pages	9/22/2017	4 hours (meetings) 10 hours (writing)	-Write preliminary document -Review preliminary draft with Dr. Fallah and Nitish -Re-write document to incorporate criticisms from Dr. Fallah, Behrad, and Nitish

Updated Divide and Conquer	10/6/2017	2 hours (meetings) 2 hours(writing)	-Review Standards with Dr. Fallah, Behrad, and Nitish based on our meeting with Dr. Wei and Dr. Richie -Finalize division of labor for the project
20 Page Internal Report	10/13/2017	20 hours per person	-Each group member produces 5 pages of new research on hardware or software implementation of an autonomous vehicle.
40 Page Internal Report	10/20/2017	20 hours per person	-Each person provides 5 pages of new research
60 Page Internal Report	11/1/2017	20 hours per person	-Each person produces an additional 5 pages of research. -Eduardo edits all research into the senior design format.
60 Page Report	11/3/2017	20 hours per person	-Each person provides 5 pages of new research
60 Page Meeting with Dr. Richie	11/7/2017	Approximately half an hour total	-Meet with Dr. Richie to discuss any errors in formatting, unnecessary documentation, provide an update on the project progress, and other issues.
100 Page Report	11/17/2017	40 hours per person	-Each person provides 10 pages of new research
Final 120 Page Report	12/4/2017	20 hours per person	-Each person provides 5 pages of new research
Part Orders	End of Fall	1 hour	-Verify that parts are ordered and in the lab before the end of Fall.
PCB design	End of Fall	10 hours	-Design PCB based on sponsor input and order at start of spring semester

6.3.2. Sponsor Schedule

Table 31 details the schedule created by Dr. Fallah's lab team, the senior design team, and the volunteers for project implementation. Deliverables are placed in the order they were assigned, but their due dates vary due to complexity and need for teamwork among group members. Every deliverable was also accompanied with a report whose data was incorporated into this document.

Table 31: Senior Design Schedule

Deliverable	Due Date	Estimated Time to Completion	Steps to Complete
-------------	----------	------------------------------	-------------------

Research RC Cars for Base	9/30/2017	10 hours (Bruce and Tyler)	-Find 4 models with a large enough chassis to hold all the parts.
Learn Localization Algorithms	11/1/2017	40 hours (Eduardo)	-Complete all 6 modules for "AI for Self Driving Cars" on Udacity.com
Configure Ultrasonic Sensors using Arduino and visualize in ROS	10/24/2017	10 hours (Christian and Eduardo)	-Create test setup for sensors -Create calibration algorithm based on initial tests. -Adjust the code so the data is sent to ROS
Complete AutoCAD model of the vehicle	10/13/2017	10 hours (Tyler)	-Find vehicle dimensions and dimensions of all the parts on the car. -Construct car in AutoCAD so the sensors do not interfere with one another. -Calculate Center of Mass of vehicle
Optical Flow	11/6//2017	20 hours (Tyler)	-Produce demo for optical flow.
Calculate Reduction Gear Ratio	11/1/2017	3 hours (Eduardo)	-Find gear ratio of differential and transmission -Calculate gear ratio to achieve desired top speed
Servo Testing	11/15/2017	10 hours (Eduardo and Christian)	-Find range of Pulse Widths -Find range of Steering Angle -Determine how to slow change in steering angle
Mounts	11/17/2017	10 hours (Bruce and Tyler)	-Create Mounts in autoCAD based on vehicle size -Laser cut wood to verify the pieces fit -Once pieces have been adjusted and approved by sponsors, cut mounts in plexiglass and aluminum
Wires and Cable Connectors	11/17/2017	2 hours (Bruce and Eduardo)	-Determine necessary wire connectors needed to interface sensors with microcontroller or NVIDIA board. -Get heat shrink for wires
Basic Movement	11/20/2017	10 hours (Christian)	-Determine a method of communicating between the Jetson and Arduino -Control the motor and servo via programming the Jetson
Running Car Demo for Sponsor	12/10/2017	N/A	-Summation of all other tasks before it.

7. Appendices

This section contains all citations referenced in the text, all approved and pending permissions given for images, and all code/datasheet references.

7.1. References

- [1]"Road Crash Statistics", *Asirt.org*, 2017. [Online]. Available: <http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics>. [Accessed: 01- Oct- 2017].
- [2] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., . . . Zieba, K. (2016, April 25). *End to End Learning for Self Driving Cars*[Scholarly project]. In *NVIDIA.com*. Retrieved September 20, 2017, from <http://images.NVIDIA.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf>
- [3] Kontzer, T. (2016, April 13). Volvo 'Drive Me' Project to Make Self-Driving Cars Synonymous with Safety | NVIDIA Blog. Retrieved September 20, 2017, from <https://blogs.NVIDIA.com/blog/2016/04/06/volvo-safety-self-driving/>
- [4] Novet, J. (2017, September 22). Tesla and AMD are working on an A.I. chip for self-driving cars, source says. Retrieved September 22, 2017, from <https://www.cnbc.com/2017/09/20/tesla-building-an-ai-chip-for-its-cars-with-amd.html>
- [5] Marshall, A. (2017, September 21). With Intel's Chips, Google Could At Last Deliver Self-Driving Cars. Retrieved September 22, 2017, from <https://www.wired.com/story/waymo-and-intel-self-driving/>
- [6]"NVIDIA Jetson Modules and Developer Kits for Embedded Systems Development", *NVIDIA.com*, 2017. [Online]. Available: <http://www.NVIDIA.com/object/embedded-systems-dev-kits-modules.html#section5>. [Accessed: 30- Sep- 2017].
- [7]H. Mujtaba, "NVIDIA's 64-Bit Denver CPU Architecture Details Unveiled - Dual Custom ARMv8 Cores Clocked at 2.50 GHz", *Wccftech*, 2017. [Online]. Available: <http://wccftech.com/NVIDIAs-64bit-denver-cpu-architecture-details-unveiled-dual-custom-armv8-cores-clocked-250-ghz/>. [Accessed: 31- Sep- 2017].
- [8]"Rally VXL: 1/10 Scale Brushless Rally Racer with TQi Traxxas Link Enabled 2.4GHz Radio System | Traxxas", *Traxxas.com*, 2017. [Online]. Available: <https://traxxas.com/products/models/electric/74076-1rally?t=gallery>. [Accessed: 15- Sep- 2017].
- [9]"Slash 4X4 Platinum: 1/10 Scale 4WD Electric Short Course Truck with Low CG chassis | Traxxas", *Traxxas.com*, 2017. [Online]. Available: <https://traxxas.com/products/models/electric/6804Rslash4x4platinum>. [Accessed: 19- Sep- 2017].
- [10]"Exceed RC SunFire Car", *Exceedrc.com*, 2017. [Online]. Available: <http://www.exceedrc.com/exrcsucar.html>. [Accessed: 14- Sep- 2017].
- [11]"Iron Track Shootout E8XBL 1:8 Scale ARTR 4WD Brushless Buggy (Red) RC Remote Control Radio Car", *Nitrorcx.com*, 2017. [Online]. Available: <http://www.nitrorcx.com/16c222-red-artr.html>. [Accessed: 15- Sep- 2017].
- [12]"Arduino Infrared Collision Avoidance", *rhydolabz.com*, 2017. [Online]. Available:

http://www.rhydolabz.com/documents/26/IR_line_obstacle_detection.pdf.

[Accessed: 20- Oct- 2017].

[13]"iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer", *SparkFun.com*, 2017. [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/3/7/3/LSM9DS1_Datasheet.pdf. [Accessed: 21- Oct- 2017].

[14]A. Industries, "Raspberry Pi 3 - Model B - ARMv8 with 1G RAM ID: 3055 - \$35.00 : Adafruit Industries, Unique & fun DIY electronics and kits", *Adafruit.com*, 2017. [Online]. Available: <https://www.adafruit.com/product/3055>. [Accessed: 17- Oct- 2017].

[15]"MSP432P401R, MSP432P401M", *ti.com*, 2017. [Online]. Available: <http://www.ti.com/lit/ds/slas826g/slas826g.pdf>. [Accessed: 10- Oct- 2017].

[16]"MSP430F552x, MSP430F551x Mixed-Signal Microcontrollers", *ti.com*, 2017. [Online]. Available: <http://www.ti.com/lit/ds/symlink/msp430f5529.pdf>. [Accessed: 15- Oct- 2017].

[17]"CC3220 SimpleLink™ Wi-Fi® Wireless and Internet-of-Things Solution, a Single-Chip Wireless MCU", *ti.com*, 2017. [Online]. Available: <http://www.ti.com/lit/ds/swas035a/swas035a.pdf>. [Accessed: 19- Oct- 2017].

[18]"ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH DATASHEET", *Atmel.com*, 2017. [Online]. Available: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf. [Accessed: 27- Oct- 2017].

[19]"8-bit Microcontroller with 16/32K bytes of ISP Flash and USB Controller DATASHEET", *microchip.com*, 2017. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf. [Accessed: 20- Oct- 2017].

[20]"Arduino Leonardo with Headers", *Store.arduino.cc*, 2017. [Online]. Available: <https://store.arduino.cc/usa/arduino-leonardo-with-headers>. [Accessed: 19- Oct- 2017].

[21]"Arduino Mega 2560 Rev3", *Store.arduino.cc*, 2017. [Online]. Available: <https://store.arduino.cc/usa/arduino-mega-2560-rev3>. [Accessed: 15- Oct- 2017].

[22]"Slash 4X4 Platinum: 1/10 Scale 4WD Electric Short Course Truck with Low CG chassis | Traxxas", *Traxxas.com*, 2017. [Online]. Available: <https://traxxas.com/products/models/electric/6804Rslash4x4platinum>. [Accessed: 19- Sep- 2017].

[23]"Structure Sensor Support Center | What are the Structure Sensor's technical specifications?", *Structure.io*, 2017. [Online]. Available: <https://structure.io/support/what-are-the-structure-sensors-technical-specifications>. [Accessed: 22- Oct- 2017].

[24]"Detailed Specs", *Content.ertilize.com*, 2017. [Online]. Available: <http://content.ertilize.com/Detailed-Specs/EN/1033980952.html>. [Accessed: 27- Oct- 2017].

[25]B. Templton, "Cameras or Lasers?", *Templetons.com*, 2017. [Online]. Available: <http://www.templetons.com/brad/robocars/cameras-lasers.html>. [Accessed: 14- Oct- 2017].

[26]"Scanse User Manual And Technical Specifications", *scanse.io*, 2017. [Online]. Available: https://s3.amazonaws.com/scanse/Sweep_user_manual.pdf. [Accessed: 20- Oct- 2017].

- [27]"RPLidar A1M8 - 360 Degree Laser Scanner Development Kit", *Robotshop.com*, 2017. [Online]. Available: http://www.robotshop.com/en/rplidar-a1m8-360-degree-laser-scanner-development-kit.html?gclid=CjwKCAjwgvfOBRB7EiwAeP7ehnnCKpjsZxnq3qsmpdY93fyz_ws0_yWQSRPF2DZms8RvKxjqo1w60xoCUE8QAvD_BwE. [Accessed: 25-Oct- 2017].
- [28]"LIDAR vs RADAR Comparison. Which System is Better for Automotive?", *Archer-soft.com*, 2017. [Online]. Available: <http://www.archer-soft.com/en/blog/lidar-vs-radar-comparison-which-system-better-automotive>. [Accessed: 16- Oct- 2017].
- [29]J. Hariyono, V. Hoang and K. Jo, "Moving Object Localization Using Optical Flow for Pedestrian Detection from a Moving Vehicle", <https://www.ncbi.nlm.nih.gov/>, 2014. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4121190/>. [Accessed: 29-Oct- 2017].
- [30]"OpenCV library", *Opencv.org*, 2017. [Online]. Available: <https://opencv.org/>. [Accessed: 20- Oct- 2017].
- [31]"OpenCV: Optical Flow", *Docs.opencv.org*, 2016. [Online]. Available: https://docs.opencv.org/3.2.0/d7/d8b/tutorial_py_lucas_kanade.html. [Accessed: 21- Oct- 2017].
- [32]"OpenCV: Contours : Getting Started", *Docs.opencv.org*, 2017. [Online]. Available: https://docs.opencv.org/3.3.0/d4/d73/tutorial_py_contours_begin.html. [Accessed: 21- Oct- 2017].
- [33]L. Router, "Linksys E2500 N600 Dual-Band Wi-Fi Router", *Linksys*, 2017. [Online]. Available: https://www.linksys.com/us/p/E2500-NP/?gclid=EAAlQobChMIqtXLx8OM1wIVxbjACh0D_AmJEAYYAIABEgLMFvD_BwE#product-features. [Accessed: 25- Oct- 2017].
- [34]"TL-WR940N | 450Mbps Wireless N Router | TP-Link", *Tp-link.com*, 2017. [Online]. Available: http://www.tp-link.com/us/products/details/cat-9_TL-WR940N.html. [Accessed: 25- Oct- 2017].
- [35]"Lizone Extra Pro External Battery Charger with Aluminum Unibody for Laptop and Smartphones – 4000mAh Black", *Amazon.com*, 2017. [Online]. Available: https://www.amazon.com/Lizone-External-Battery-Aluminum-Smartphones/dp/B00HLDSNKI/ref=sr_1_17?s=electronics&ie=UTF8&qid=1508961307&sr=1-17&keywords=portable+laptop+charger&refinements=p_36%3A10000-999999999#HLCXComparisonWidget_feature_div. [Accessed: 25- Oct- 2017].
- [36]"MAXOAK 50000mAh 6 Port(5/12/20v) Portable Charger External Battery Power Bank for Laptop & Notebook", *Amazon.com*, 2017. [Online]. Available: https://www.amazon.com/MAXOAK-50000mAh-Portable-External-Notebook-Most/dp/B00YP823NA/ref=sr_1_1_sspa?ie=UTF8&qid=1508961683&sr=8-1-spons&keywords=maxoak+50000mah+6+port+5+12+20v+portable+charger&psc=1. [Accessed: 25- Oct- 2017].
- [37]"Power Supply Safety Standards, Agencies and Marks." CUI INC. <http://www.cui.com/catalog/resource/power-supply-safety-standards-agencies-and-marks.pdf>

- [38]Li, Yunxin (Jeff). "An Overview of the DSRC/WAVE Technology." http://www.v2x.ir/Admin%5CFiles%5CeventAttachments%5CAn%20Overvie%20of%20the%20DSRCWAVE%20Technology-Yunxin%20Li_172.pdf
- [39] *Dedicated Short-Range Communications (DSRC) Standards in the United States - IEEE Journals & Magazine*, <http://ieeexplore.ieee.org/document/5888501/>.
- [40]"Security Laboratory." *Dispelling Common Bluetooth Misconceptions*, www.sans.edu/cyber-research/security-laboratory/article/bluetooth.
- [41]Ma, Xiaomin, et al. "Performance and Reliability of DSRC Vehicular Safety Communication: A Formal Analysis." *EURASIP Journal on Wireless Communications and Networking*, Springer International Publishing, 18 Jan. 2009, <https://jwcn-urasipjournals.springeropen.com/articles/10.1155/2009/969164>.
- [42]IsecT Ltd. "ISO/IEC 27033:2010+ Information Technology — Security Techniques — Network Security." *ISO/IEC 27033 IT Network Security Standard*, www.iso27001security.com/html/27033.html.
- [43]Liščák, Štefan, et. al. "Safety Requirements for Road Vehicles." http://pernerscontacts.upce.cz/33_2013/Liscak.pdf
- [44]Hamid, Haroon H. "The NHTSA's Evaluation of Automobile Safety Systems: Active or Passive?" <http://lawecommons.luc.edu/cgi/viewcontent.cgi?article=1161&context=iclr>
- [45]"IPC Standards." IPC. http://www.ipc.org/4.0_Knowledge/4.1_Standards/OEM-Stds-A4-English-1111-ONLINE.pdf
- [46]"1780 - Standard for the Specification of Inertial Measurement Units (IMU)." *IEEE SA - 1780 - Standard for the Specification of Inertial Measurement Units (IMU)*, <https://standards.ieee.org/develop/project/1780.html>.
- [47]"Automated Vehicles for Safety." NHTSA, 18 Oct. 2017, www.nhtsa.gov/technology-innovation/automated-vehicles.
- [48]Christiaan Hetzner Automotive News Europe May 15, 2017 11:11 CET. "German Industry Welcomes Self-Driving Vehicles Law." *Automotive News*, 16 May 2017, <http://europe.autonews.com/article/20170515/ANE/170519866/german-industry-welcomes-self-driving-vehicles-law>.
- [49]ISO/IEC 9899:1999, C99 Standard <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf>
- [50]"Wiki." Ros.org, <http://wiki.ros.org/CppStyleGuide>.
- [51]Python.org. (2017). *PEP 8 – Style Guide for Python Code*. <https://www.Python.org/dev/peps/pep-0008/>
- [52]"Wiki." Ros.org, <http://wiki.ros.org/PyStyleGuide>.
- [53]Etherington, Darrell. "Department of Transportation Releases New Self-Driving Vehicle Guidelines." *TechCrunch*, TechCrunch, 12 Sept. 2017, <http://techcrunch.com/2017/09/12/department-of-transportation-release-new-self-driving-vehicle-guidelines/>.
- [54]"Automated Driving." SAE International. https://www.sae.org/misc/pdfs/automated_driving.pdf
- [55]Specification of the Bluetooth System. https://www.inf.ethz.ch/personal/hvogt/proj/btmp3/Datasheets/Bluetooth_11_Specifications_Book.pdf
- [56]"Wiki." Ros.org, <http://wiki.ros.org>

- [57] The Verge. (2017). *Intel predicts a \$7 trillion self-driving future*. [online] Available at: <https://www.theverge.com/2017/6/1/15725516/intel-7-trillion-dollar-self-driving-autonomous-cars> [Accessed 10 Oct. 2017].
- [58] Recode. (2017). *Ford's partnership with Lyft finally gives it a clear plan for self-driving cars*. [online] Available at: <https://www.recode.net/2017/9/27/16374060/lyft-ford-self-driving-cars-partnership> [Accessed 8 Oct. 2017].
- [59]"Waymo and Intel Collaborate on Self-Driving Car Technology | Intel Newsroom", *Intel Newsroom*, 2017. [Online]. Available: <https://newsroom.intel.com/editorials/waymo-intel-announce-collaboration-driverless-car-technology/>. [Accessed: 01- Nov- 2017].
- [60] A. LaFrance, "Driverless Cars Could Save Tens of Millions of Lives This Century", *The Atlantic*, 2017. [Online]. Available: https://www.theatlantic.com/technology/archive/2015/09/self-driving-cars-could-save-300000-lives-per-decade-in-america/407956/?utm_source=SFTwitter. [Accessed: 01- Nov- 2017].
- [61]*Lidar-uk.com*, 2017. [Online]. Available: <http://www.lidar-uk.com/how-lidar-works/>. [Accessed: 01- Nov- 2017].
- [62]"Merits of Coherent Detection Optical Transmission", *100G Optical Components, Coherent, PIC, DWDM*, 2017. [Online]. Available: <https://www.neophotonics.com/merits-coherent-detection-optical-transmission/>. [Accessed: 01- Nov- 2017].
- [63] "Basic Sonar System (Active)", *Fas.org*, 2017. [Online]. Available: https://fas.org/man/dod-101/navy/docs/es310/asw_sys/asw_sys.htm. [Accessed: 01- Nov- 2017].
- [64]"Inertial Measurement Unit (IMU)", *Ssl.umd.edu*, 2017. [Online]. Available: <http://www.ssl.umd.edu/projects/RangerNBV/thesis/2-4-1.htm>. [Accessed: 01- Nov- 2017].
- [65]M. Martin, "How to Care for Your Car's Differential", *Popular Mechanics*, 2017. [Online]. Available: <http://www.popularmechanics.com/cars/how-to/g1140/how-to-care-for-your-cars-differential/>. [Accessed: 01- Nov- 2017].
- [66]"IR Sensor Circuit and Working with Applications", *EIProCus - Electronic Projects for Engineering Students*, 2017. [Online]. Available: <https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/>. [Accessed: 01- Nov- 2017].
- [67]Amazon.com. (2017). *Diza100 Wireless Network Adapter*. [online] Available at: https://www.amazon.com/Wireless-High-gain-Complies-Standard-Supports/dp/B075M5NQ8F/ref=sr_1_10?ie=UTF8&qid=1510159657&sr=8-10&keywords=wireless%2Bnetwork%2Badapter%2Blinux&th=1 [Accessed 8 Nov. 2017].
- [68]Amazon.com. (2017). *TP-Link N300*. [online] Available at: https://www.amazon.com/gp/product/B0088TKTY2/ref=s9u_simh_gw_i2?ie=UTF8&fpl=fresh&pd_rd_i=B008IFXQFU&pd_rd_r=df2286db-c4a1-11e7-9821-4b13dd402765&pd_rd_w=dxEKZ&pd_rd_wg=DSyA6&pf_rd_m=ATVPDKIKX0DER&pf_rd_s=&pf_rd_r=VMJEPJRMNKMKEK6ZS2EZ&pf_rd_t=36701&pf_rd_p=1cf9d009-399c-49e1-901a-7b8786e59436&pf_rd_i=desktop&th=1 [Accessed 8 Nov. 2017].
- [69] Store.arduino.cc. (2017). *Arduino Due*. [online] Available at: <https://store.arduino.cc/usa/arduino-due> [Accessed 1 Nov. 2017].

- [70] Ti.com. (2017). *CC3220 SimpleLink™ Wi-Fi® Wireless and Internet-of-Things Solution, a Single-Chip Wireless MCU*. [online] Available at: <http://www.ti.com/lit/ds/swas035a/swas035a.pdf> [Accessed 1 Nov. 2017].
- [71] Franklin, D. and →, V. (2017). *NVIDIA Jetson TX2 Delivers Twice the Intelligence to the Edge*. [online] Parallel Forall. Available at: <https://devblogs.NVIDIA.com/paralleforall/jetson-tx2-delivers-twice-intelligence-edge/> [Accessed 11 Nov. 2017].
- [72]"OpenSLAM.org", Openslam.org, 2017. [Online]. Available: <http://openslam.org/gmapping.html>. [Accessed: 11- Nov- 2017].
- [73] A. Kai M. Wurm, "OctoMap - 3D occupancy mapping", Octomap.github.io, 2017. [Online]. Available: <https://octomap.github.io/>. [Accessed: 11- Nov- 2017].
- [74] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees" in *Autonomous Robots*, 2013; DOI: 10.1007/s10514-012-9321-0
- [75] Autodesk.com. (2017). *PCB Design & Schematic Software | EAGLE | Autodesk*. [online] Available at: <https://www.autodesk.com/products/eagle/overview> [Accessed 11 Nov. 2017].
- [76] Kicad-pcb.org. (2017). *KiCad EDA*. [online] Available at: <http://kicad-pcb.org/> [Accessed 11 Nov. 2017].
- [77] "KiCad Reviews", *Pcbshopper.com*, 2017. [Online]. Available: <https://pcbshopper.com/kicad-reviews/>. [Accessed: 12- Nov- 2017].
- [78] "PCB Design Software | Innovation For PCB Design | Altium", *Altium.com*, 2017. [Online]. Available: <http://www.altium.com/>. [Accessed: 12- Nov- 2017].
- [79] "Explore | CircuitStudio", *Circuitstudio.com*, 2017. [Online]. Available: <http://www.circuitstudio.com/explore>. [Accessed: 12- Nov- 2017].
- [80] Teel, J. (2017). *PCB Design Software – Which One is Best?*. [online] PREDICTABLE DESIGNS. Available at: <http://predictabledesigns.com/pcb-design-software-which-one-is-best/> [Accessed 11 Nov. 2017].
- [81] "Free PCB Design Software | CircuitMaker", *Circuitmaker.com*, 2017. [Online]. Available: https://circuitmaker.com/#why_circuitmaker. [Accessed: 12- Nov- 2017].
- [82] "DipTrace - Schematic and PCB Design Software", *Diptrace.com*, 2017. [Online]. Available: <https://diptrace.com/>. [Accessed: 12- Nov- 2017].
- [83] Mallick, S. (2017). *Histogram of Oriented Gradients | Learn OpenCV*. [online] Learnopencv.com. Available at: <https://www.learnopencv.com/histogram-of-oriented-gradients/> [Accessed 13 Nov. 2017].
- [84] Rosebrock, A. (2017). *Histogram of Oriented Gradients and Object Detection - PyImageSearch*. [online] PyImageSearch. Available at: <https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/> [Accessed 13 Nov. 2017].
- [85] The Official NVIDIA Blog. (2017). *MIT Students Build Robotic Racecars with Jetson TK1 | NVIDIA Blog*. [online] Available at: <https://blogs.NVIDIA.com/blog/2015/10/07/robot-racecars-jetson/> [Accessed 17 Nov. 2017].
- [86]"Self Driving RC Car", *Zheng Wang*, 2017. [Online]. Available: <https://zhengludwig.wordpress.com/projects/self-driving-rc-car/>. [Accessed: 17- Nov- 2017].

[87] JetsonHacks. (2017). *MIT RACECAR Walkthrough - NVIDIA Jetson TK1* - JetsonHacks. [online] Available at: <http://www.jetsonhacks.com/2015/10/06/mit-racecar-walkthrough-NVIDIA-jetson-tk1/> [Accessed 17 Nov. 2017].

[88] "NVIDIA Jetson TX2 J21 Header Pinout - JetsonHacks", *JetsonHacks*, 2017. [Online]. Available: <http://www.jetsonhacks.com/nvidia-jetson-tx2-j21-header-pinout/>. [Accessed: 03- Dec- 2017].

7.2. Permissions

 Eduardo Linares
Today, 1:21 PM
pmwebmaster@hearst.com

Hello Ezra,


My name is Eduardo Linares and I'm a senior majoring in Electrical Engineering at UCF. I'd like to ask for permission to use the image shown below that was published in a PM article about differentials (<http://www.popularmechanics.com/cars/how-to/g1140/how-to-care-for-your-cars-differential/>) for my senior design paper. The PM article will be cited properly if we're given permission. If you're not authorized to provide permission for the image, can you point me toward who I should ask permission from? It would be greatly appreciated.

Have a great day,

Eduardo Linares



Permission to use data graphs

 info <info@nvidia.com>
Wed 11/8, 4:06 PM

Good afternoon Mr. Hardy,

Could you please cite the original publication in your paper, then all should be okay.

Thank you and have a great day!

Contact Us

Your Name (required)

Christian Theriot

Your Email (required)

theriotcet96@knights.ucf.edu

Subject

Jetson TX2 J21 Header Pinout

Your Message

Hello,
I'm a senior majoring in Computer Engineering at UCF, and I'd like to ask for permission to use the image shown in the [JetsonTX2 Header Pinout](http://www.jetsonhacks.com/nvidia-jetson-tx2-j21-header-pinout/) located (<http://www.jetsonhacks.com/nvidia-jetson-tx2-j21-header-pinout/>) for my senior design paper. This page is already cited properly elsewhere, however an image of the header will only be used with permission.

Thank you, have a great day,
Christian Theriot

Please enter the text below

G MZ 3

SEND

Jetson TX2 J21 Header Pinout



JetsonHacks <jim@jetsonhacks.com>



Reply all | v

Today, 1:45 PM

Christian Theriot v

Hi Christian,

Permission granted.

Your Knights looked good yesterday, it was a great game to watch!

Best,

Jim Benson