Final Design Report

µLAREN – Small-Scale Intelligent Vehicle Design Platform

Jay Miley | jcmiley@calpoly.edu
Evan Phillips | evphilli@calpoly.edu
Chris Grant | chgrant@calpoly.edu

Dr. Charles Birdsong, Sponsor

Mechanical Engineering Department Senior Project
California Polytechnic State University, San Luis Obispo

2016 - 2017

Statement of Disclaimer

Since this project is a result of a class assignment, it has been graded and accepted as fulfillment of the course requirements. Acceptance does not imply technical accuracy or reliability. Any use of information in this report is done at the risk of the user. These risks may include catastrophic failure of the device or infringement of patent or copyright laws. California Polytechnic State University at San Luis Obispo and its staff cannot be held liable for any use or misuse of the project.

Abstract

Intelligent Vehicle Design is a growing field with the potential to save many lives by actively minimizing the impacts of human error. Though there are many ways to research intelligent vehicle control, full-scale implementations are expensive and dangerous and computer simulations have extremely steep learning curves. Researchers and students need an accessible, adaptable, and robust development platform to rapidly create and test autonomous control algorithms. While small-scale platforms are often designed from the ground up for specific projects, this requires analysis, design, and manufacture. The goal of this project is to develop a small-scale intelligent vehicle that can be configured with physical sensors and programmed with control algorithms designed in Simulink. We will strive to make our design adaptable and reproducible through intentional design and documentation. We have completed the design to adapt a $1/7^{th}$ scale remote control vehicle with a custom chassis, independently driven wheels, and a Raspberry Pi based control package. An inertial measurement unit, an ultrasonic rangefinder, and a camera will give the system real-time data about itself and its surroundings. This well-documented research platform will enable more students to get hands on experience in developing and testing intelligent vehicle systems. These students will become the next generation of vehicle safety engineers, developing the life-saving intelligent vehicle systems of the future.

Acknowledgements

We would like to thank our sponsors:

maxon motor

driven by precision

Mechanical Engineering Student Fee Allocation Committee (MESFAC) 2016-2017

CPConnect 2016-2017

We would also like to thank our Project Advisor, Technical Advisor, and CNC support:

Dr. Charles Birdsong, Project Advisor

Charlie Refvem, Mechanical Engineering Graduate Student and Technical Advisor

Max Selna, CNC Support

Contents

Figures

Tables

1. INTRODUCTION

Intelligent Vehicle Design is a growing field with the potential to save many lives. It enables vehicles to adapt and react to situations that the driver may not notice, resulting in a safer, more efficient traffic flow. Researchers and students need an accessible, adaptable, and robust development platform, which does not currently exist in an accessible format. Presently, research is completed in several ways. Modified full scale vehicles are used but are expensive and dangerous. Simulation can be used for preliminary research, but requires both mechanical and extensive software knowledge. While small scale platforms are often used, they require significant start-up design. A well-documented, programmable and modular computer driven small-scale vehicle will enable rapid potentially life-saving advancements.

The goal of this project is to develop a small scale intelligent vehicle that can be configured with physical sensors and programmed with control algorithms designed in Simulink. We will enter the final design to compete internationally in the International Technical Conference on the Enhanced Safety of Vehicles (ESV) Student Safety Technology Design Competition (SSTDC), hopefully bringing prestige to the university and department. Additionally, the design, build, and testing of this vehicle will support creation of kits for an intelligent vehicle controls course being designed by Dr. Charles Birdsong for the Mechanical Engineering Department at Cal Poly. The new course will allow students with limited programming knowledge to gain practical experience with developing control algorithms for autonomous vehicle functions.

Cal Poly will provide an open system that can be used by other research and educational institutions to easily participate in intelligent vehicle design research.

2. BACKGROUND

To develop this project well, we must understand the implications and impact of our project, similar technologies, and context of our industry. Having this context will allow us to create a more useful, pertinent, and thoughtful product.

With the increasing presence of autonomous technologies in the automotive industry, there is an inherent need to develop testing methods for the control systems that will improve vehicle safety. New systems are being generated at an impressive pace, which means that test methods need to be established. In Ann Arbor Michigan, the UMTRI Safety Pilot Program has invested $20 million into creating a test track for vehicle to vehicle connection experiments [1]. Projects such as this are extremely valuable in pushing the envelope regarding autonomous vehicle technologies, but are inaccessible to smaller institutions. Universities across the United States contain the country's next generation of intelligent vehicle researcher's and engineers. A cheaper and more practical alternative for autonomous vehicle research is through small scale models. Although small scale vehicles are largely thought as inferior in terms of similitude, a prototyping platform

would provide valuable experience to engineering students who can bring knowledge and new ideas to the automotive industry.

2.1 ESV COMPETITION

The SSTDC is a competition that challenges students to conceive, design, and test cutting edge vehicle safety technologies. Competitors converge at the ESV conference to present their findings to the automotive industry. The ESV conference is a technical conference that is intended to provide a collaborative space for the leading edge of vehicle safety research. It is a collaboration between the United States' Department of Transportation & National Highway Traffic Safety Administration and 14 participating ESV member countries and governmental organizations. The 2017 conference will take place from June 5-8, 2017 in Detroit, Michigan. The first stage of the competition is a call for 300 word abstracts, due November 11, 2016. Judges will select which teams participate in each of the three regional competitions (North America, Europe, Asia-Pacific) based on those abstracts. In March 2017, prototypes / progress will be evaluated and three finalists will be selected from each regional competition to proceed to the ESV conference & international competition. At the conference, teams will set up displays and offer demonstrations to conference attendees. Each team will give both a 15-minute technical oral presentation and a 10-minute functional model demonstration to judges and other teams. After these sessions, judges will deliberate and select the winner and runner up.

Abstract submissions will be scored out of 100 as follows:

- Potential impact on safety problem being addressed (30 points)
- Originality (25 points)
- Practicability of creating a functional scale model (25 points)
- Supporting details, quality, technical depth (20 points)

Abstract submissions should specify which 'safety category' the project falls under. One such category is 'Autonomous Vehicle Issues,' which this project falls under.

Both regional and international competitions consist of a six-page report and a functional prototype demonstration. They will be scored out of 100 as follows:

- Potential impact on safety problem being addressed (40 points)
    - Did the team address a safety problem?
    - How did the team test and evaluate its system?
    - What metrics did the team use?
    - What are the results of the testing?
    - Are conclusions presented clearly?
    - What potential or expected effects will the system have on traffic safety?
- Originality (20 points)
- Functional scale model, physical presentation (20 points)
- Oral presentation (10 points)
- Supporting details, quality, thoroughness, technical depth (10 points)

Additionally, students are encouraged to include in their report: estimated safety benefits in terms of lives saved or crashes presented and percentage of the fleet covered.

2.2 EXISTING TECHNOLOGIES

There are a wide variety of autonomous and semi-autonomous technologies that are being developed as well as ones that are already present in the automotive industry today. These technologies include, but are not limited to: pedestrian detection, terrain sensing, adaptive cruise control, collision detection, motion prediction, lane assist, and a wide variety of stability control systems. All of these and more aim to improve the driving experience and decrease the inherent risk that comes with traveling the roadways.

Pedestrians account for roughly 14% of the total fatalities per year in US traffic accidents [2]. In order to avoid pedestrian fatalities, extensive research is being devoted to human detection software that enables collision avoidance. This software requires in depth algorithms to assess and analyze very complicated and sometimes noisy data. One method of pedestrian detection is through computer vision. Lie Guo et al. uses a complicated camshaft algorithm to detect color probabilities at the users' torso, and processes data with a Kalman filter [3]. Another method of pedestrian detection is by using recognized shapes to analyze the contours through a camera. These shapes can allow a controller to accurately predict future movements that are common among pedestrians [4]. There are a significant number of already discovered methods for detecting pedestrians, predicting movement, and avoiding collisions, as well as many that are yet to be developed. A small scale vehicle platform could provide a means for developing pedestrian detection and avoidance algorithms.

Although terrain sensing abilities is not necessary for urban vehicles that travel on paved roadways, development in this area could lead to more adaptable and safe off-road vehicle. Typically, terrain sensing is an area of research for military or government operations, but could generate a significant amount of interest among students. Furthermore, vehicles that can accurately predict and adjust to a changing terrain would decrease the likelihood of dangerous rollover or loss of traction on dirt roads. An adapted off-road small scale vehicle could implement traction control systems that allow vehicle to better adapt to changing terrain. The fundamentals of traction control systems are discussed in more detail below.

A more applicable area for roadway safety research is adaptive cruise control. Adaptive cruise control is a highway vehicle feature that adapts an automobile's speed based upon the traffic environment. Typical systems use radar to determine the distance between two vehicles, and then changes the following vehicle velocity based upon a relative safe distance. Figure 1 depicts the basic principle behind adaptive cruise control systems [5]. These systems are an example of semi-autonomous vehicle technology that could drastically reduce the amount of accidents due to distracted driving or carelessness.

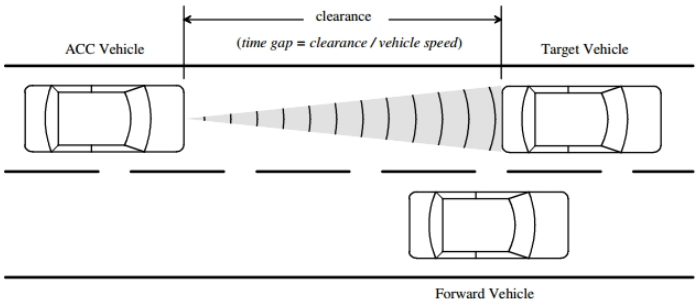


Figure 1: Schematic of adaptive cruise control

Collision detection and avoidance is one of the most heavily researched autonomous vehicle technology, but it is also very difficult to develop algorithms than can safely avoid obstacles. The safest option is to simply brake to avoid a collision, but more research is being done for alternatives. When traveling at high speeds, swerving to avoid an object can potentially be very dangerous, as this could cause rollover or potentially even collision with other undetected objects. As a result, a system is needed to detect dangerous obstacle in the path of the vehicle and monitor vehicle angular and linear accelerations. The system also needs to be careful to not interfere to much with a driver's naturals abilities. A system too invasive will take away a driver's confidence and potentially have adverse effects, including the car seizing control from the driver at an inopportune time [6].

A master's thesis at Cal Poly by Thomas Stevens investigated the effects that a collision avoidance system has on a driver. The key to developing an effective collision avoidance system is that the software should not be too overpowering as to make the driver feel like they are no longer in control. It should be a seamless integration that only interferes when absolutely necessary. Drivers need to be willing to embrace an active safety system. Fitzgerald's vehicle platform was designed to assess how to integrate a semi-autonomous driving system, and found that there needs to be a balance between lightly tuned interferences and more sensitive ones [7]. This balance largely depends on the driver and their tendencies. Developing a research platform can drastically increase the rate at which useful information can be gathered from the driving population.

When considering vehicle to vehicle collisions, extensive research must go into motion prediction models that govern the control algorithms. There are a variety of physics, maneuver, and interaction aware models [8]. The numerous research possibilities presented by the differing motion prediction models alone is enough to suggest the need of a small scale platform. Developing a model that accurately assesses risk and predicts future vehicle trajectories is the framework to a successful collision avoidance system.

Electronic Stability Control (ESC) is an intelligent vehicle feature that works to reduce the loss of traction when cornering or swerving. The Society of Automotive Engineers (SAE) defines an ESC system as a car that does the following: [9]

- Is computer controlled and the computer contains a closed-loop algorithm designed to limit understeer and oversteer of the vehicle
- Has a means to determine vehicle yaw velocity and side slip
- Has a means to monitor driver steering input

- Has a means of applying and adjusting the vehicle brakes to induce correcting yaw torques to the vehicle
- Is operational over the full speed range of the vehicle (except below a low-speed threshold where loss of control is unlikely)

The system works by engaging specific brakes to help direct the vehicle in the direction intended. There are a wide variety of active stability and traction systems operating in modern vehicles. For example, BMW developed a Dynamic Traction Control (DTC) system that actually allows slip in certain situations. DTC claims to allow a more 'sporty' drive by permitting some small levels of slip at the tires that increase the vehicles ability to corner at a faster rate. Although this is not necessarily a safety system, it works in conjunction with the Dynamic Stability Control (DSC) system to keep the vehicle on the roadway [10]. Electronic systems such as this all monitor several kinematic and kinetic aspects. The angular accelerations of the vehicle—pitch, roll and yaw are all determined by an IMU that sends signals back to the car's computer. The computer determines what the driver is intending to do, and adjusts the torque through the drivetrain. A stability control system must be able to work in tandem with collision avoidance systems in order to prevent potentially dangerous maneuvers that may send a vehicle off the roadway.

In order to determine the necessary requirements of a small scale vehicle platform used for testing a traction or stability control system, it is important to understand how these systems work at a fundamental level. Traction systems simply work by limiting the slippage at each tire/ground interface: Tire slippage, λ, can be defined as the ratio of the relative velocity between the tire road interface to the absolute velocity of the vehicle:

$$\lambda = \frac{(\omega_w * r_w - V_V)}{V_V}, \qquad (1)$$

where ω_w is the angular velocity of the wheel, r_w is the radius of the wheel, and V_V is the absolute velocity of the wheel [11]. On a scale vehicle platform, the angular velocity can be measured by a tachometer mounted at the inside of each wheel, and the absolute velocity can be obtained by integrating the IMU longitudinal acceleration data. A control system would work to minimize the slip ratio by adjusting the output torque from the motor shaft and the braking force applied to each wheel. Therefore, a more effective vehicle platform would allow for independent torque adjustment at each wheel.

Furthermore, torque adjustments for electronic stability control are dependent upon the type of motion the vehicle intends to make. Figure 2, from the Insurance Institute for Highway Safety [12] shows how individual braking systems can be used to correct oversteer and understeer in a vehicle. The basic principle is to control the yaw rate by applying a braking torque to one of the wheels. This load is transferred through the tire and into the ground to create a restoring moment about the vehicle's central axis.

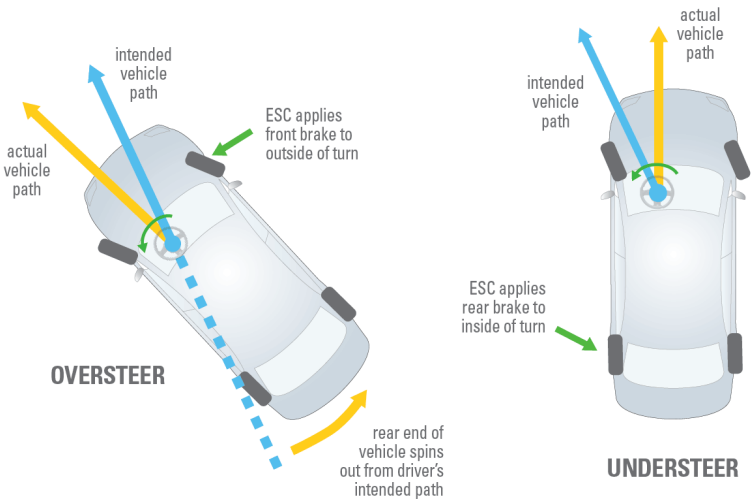


Figure 2: Braking procedures for oversteer and understeer corrections

2.3 SIMILAR PRODUCTS

The typical benefactors of a small scale vehicle platform are engineering instructors and students who are attempting to develop and test control algorithms associated with autonomous vehicle technologies. Very few courses at universities in the United States are offered that present students with the chance to experience autonomous vehicle technologies in a "hands on" environment. CSU Northridge currently offers a two semester, capstone style course for intelligent ground vehicles [13]. The course specifically focuses on the navigation of autonomous vehicles, where students design the vehicle from start to finish. Figure 3 shows students at CSU Northridge testing their vehicle's navigation skills.

Figure 3: CSU Northridge Autonomous vehicle and test course. Students develop and program the vehicle to navigate through a course. The product competes at the International Ground Vehicle Competition every year.

Testing control systems on full-scale models would be expensive and time consuming to implement. A smaller test platform would allow for rapid testing, as well as utilize the fresh and creative minds of undergraduate and graduate students.

Turkish researchers developed a small scale prototyping platform for vehicle dynamics. They modified a LOSI brand electric RC car by adapting it with an I/O board and IMU. The I/O board relays data from the IMU via Bluetooth to MATLAB-Simulink, where the computer operates as the controller. The researchers then tested and developed an anti-lock brake safety system, as well as a roll over prevention system. The ABS system on the model used a pulse brake operating at 10 Hz to simulate an actual ABS system. A weight was also placed on top of the vehicle to raise the center of gravity [14].

Advancements in technology suggest that fully automated vehicles are a very realistic possibility in the future. Research is already being done to model computer governed highways. Manh la et al. developed a small scale research platform to investigate potential fully autonomous transportation systems [15]. The platform consists of a driving arena, an indoor localization system, automated RC cars, and roadside monitoring facilities. The small scale remote controlled vehicles used in the model were programmed to follow a pre-determined trajectory, where they were tracked by the localization system. The algorithm used to control the vehicles trajectory was the primary parameter under investigation. Future research problems were also presented that could be investigated by this kind of research platform. The platform was found to be very effective at investigating potential intelligent transportation algorithms. Small scale research can be very useful when investigating and developing intelligent vehicles.

With the growing presence of autonomous technology in the automotive industry, more vehicles are becoming equipped with advanced safety features. These systems are governed by complex algorithms that cannot be verified until a test or simulation is conducted. The testing process can be very tedious, which slows down the development of potentially innovative control algorithms associated with autonomous vehicle safety. Small scale research platforms naturally increase the rate at which programs can be evaluated in a realistic environment.

An article on Hackaday.com [16] describes a project for a master's dissertation. The student took a $1/10^{th}$ scale RC car and completely removed the drivetrain, replacing it with a custom implementation of a brushless out-runner motor on each wheel. Combining the unique drivetrain with hall effect sensors for speed feedback, an IMU, and a Simulink programmable microcontroller, the student produced a rapidly adjustable controls system. At the time the article was written, the student had implemented front and rear power offsets as well as virtual differentials that could transition between locked, open, and overdriven. The student commented that they planned to implement torque vectoring – a technology useful in electronic stability control. Between the Simulink compatibility, IMU, and independently drive wheels, this project's capabilities are probably the closest to our needs, though it lacks open documentation and thus does not fulfill the project need.

In addition to physical prototyping platforms, the use of computer simulation for vehicular dynamics and control systems is widespread [7]. These simulation systems can be extremely robust, with the ability to monitor and export up to 800 variables. Unfortunately, the available systems are so versatile and full-featured at the expense of usability. The resultant learning curve is very steep, preventing rapid development of multiple algorithms. Computer simulation is often used as a verification method for potential algorithms before bringing the control system to a full scale vehicle test platform. Despite the accuracy and importance of simulation and full scale vehicle testing, small-scale testing remains extremely relevant as an avenue for rapid, accessible innovation.

2.4 RC PLATFORMS

For the purposes of this project, RC cars can be divided into two classes – hobby grade and toy grade. Hobby grade RC cars tend to be more fully featured – including powerful motors, suspension, spare parts, and more. Toy grade tend to be less expensive but less robust. Table 1 below contains various feature ranges that RC cars may have.

Table 1: RC Car common feature options

Vehicle Scale	$1/18^{th}$ to $1/7^{th}$
Power Source	Electric / Gas / Nitro
Motor Type	Brushed Electric / Brushless Electric / Gas Piston
Driven Wheels	4WD / 2WD
Drivetrain	Single shaft driven / Differential driven
Suspension	None / Full independent suspension / Independent front
Chassis Materials	Plastic / Metal / Mixed
Drivetrain Materials	Plastic / Metal / Mixed
Reparability	Kit style car / Spare parts available / No sourceable parts

Several things will narrow the scope of cars that we will look at. To maximize dynamic similitude between our platform and a real car, suspension and a differential are strongly desired. Selecting an electric car will serve the dual purpose of improving safety and providing a power source to our electronic system. Since a secondary goal of this platform is to be used in a course (for several years), the kit should be repairable with easily source-able parts. In the preliminary design phase, a decision matrix will be used to compare several different RC car options and support the final decision.

2.5 MICROCONTROLLER PLATFORMS

This project has several requirements that affect the selection of a microcontroller. A critical requirement is the usage of Simulink for programming to provide mechanical engineers with a familiar environment to explore intelligent vehicle control. On a deeper level, selection of a microcontroller with extensive documentation and user base will enable the possibility of more advanced projects. Since the goal is to integrate sensors and in turn control the car, the microcontroller must have reasonable input / output support. Finally, the microcontroller must have reasonable storage and processing power for extensive control algorithms. Table 2 below contains several popular microcontrollers and project critical specifications.

Table 2: Microcontrollers and critical specifications

Arduino Uno [17]	Simulink Compatibility	Yes
	I/O Ports	14 Digital (6 PWM), 6 Analog
	Clock Speed / Program Memory / RAM	16Mhz / 32KB / 2KB
	Operating System / Firmware	Arduino Bootloader, C variant
	Cost	$25
Arduino Mega [18]	Simulink Compatibility	Yes
	I/O Ports	54 Digital (15 PWM), 16 Analog
	Clock Speed / Program Memory / RAM	16Mhz / 256KB / 8KB
	Operating System / Firmware	Arduino Bootloader, C variant
	Cost	$46
Raspberry Pi 3B [19]	Simulink Compatibility	Yes, incl. computer vision
	I/O Ports	26 GPIO / 4 USB / Camera
	Clock Speed / Program Memory / RAM	1.2GHz / SD card / 1GB
	Operating System / Firmware	Linux Variants
	Cost	$35
Raspberry Pi 1 A+ [20]	Simulink Compatibility	Yes, incl. computer vision
	I/O Ports	26 GPIO / 1 USB / Camera
	Clock Speed / Program Memory / RAM	700MHz / SD card / 512MB
	Operating System / Firmware	Linux Variants
	Cost	$25
BeagleBone Black [21]	Simulink Compatibility	Yes, thru embedded coder, incl. computer vision
	I/O Ports	69 GPIO / 1 USB
	Clock Speed / Program Memory / RAM	1GHz / 4GB / 512MB
	Operating System / Firmware	Linux Variants
	Cost	$55

Technical specifications were retrieved from the referenced webpages, save for Simulink compatibility which was retrieved directly from MathWorks by searching the Hardware Support webpage [22].

2.6 TECHNICAL CHALLENGES

Although it is near impossible to plan for every problem a project will have, thinking through some possible challenges in the project provides a great start. The first challenges involve the implementation of the technology. This includes the compatibility of Simulink, ease of extra module integration, and response time of the vehicle to user and environmental inputs.

Compatibility with Simulink is essential to this project. Mechanical engineers at Cal Poly are exposed to Simulink in their controls course, and it is used across the industry for controls and automotive applications. To use this to our advantage we must know in advance that the microcontroller we choose can load code exported from Simulink. The next challenge is to allow the user to have confidence that any additional sensors will function properly on the vehicle platform. The platform must be powerful enough to run the designed algorithms, and it must be fast enough to respond to new inputs. A product that works perfectly but takes ten times longer to use will not be seen as an effective prototyping platform, so it must be powerful and adaptable.

The next area of challenges deals with the physical properties of the platform. There are many details that must be incorporated to create a small-scale vehicle with reasonable similitude to full scale vehicles. In our design for similitude we will consider factors including weight, center of gravity, friction, acceleration, braking, turning radius, and many more. Some factors that will challenge the physical properties of the system include the weight distribution of the vehicle and keeping the system rugged and resilient. Though similitude will be considered throughout the design process, it is not the primary goal – the critical path is creating an accessible, adaptable development platform.

3. DESIGN REQUIREMENTS AND SPECIFICATIONS

The primary goal of this project is to develop a 10^{th} scale research platform that will allow students to develop new and innovating intelligent vehicle safety systems. To accomplish this, the model must be adaptable, robust, easily maintained, and easily reproduced.

3.1 CUSTOMER REQUIREMENTS

Professor Birdsong is in the process of developing a controls course that will be focused around a platform similar to ours. We will focus on developing a project for the ESV competition but design it in such a way that it can be adapted for the course. After meeting with him and discussing the project goals and scope, the following requirements were developed. The vehicle must be:

- Physically robust so that it can absorb repeated impacts
- Easy to program for undergraduate and graduate level mechanical engineering students
- Adaptable to new sensor modules
- Compact for transportation and storage
- Low cost so that a number of models can be reproduced
- Display as close to realistic vehicle dynamics as possible
- Operate via autonomous and semi-autonomous control
- Easily maintained and reproduced

3.2 SPECIFICATIONS

Through Dr. Birdsong's input and researching the ESV design competition, a set of engineering specifications were developed. Though low-cost is a customer requirement, we did not include it as an engineering specification because while it is a consideration, we are making this model specifically for the ESV competition and hope to make an impressive 'floor model.' The quality function deployment model helped to produce a set of engineering specifications related to these requirements. The QFD, found in Appendix A, also helped to assess where we need to fall relative to other similar products.

Table 3: List of engineering specifications and tolerances

Spec. #	Parameter Description	Requirement or Target (units)	Tolerance	Risk	Compliance
1	Withstands impacts	---	---	H	T
2	Latency	50 ms	Max	M	I,A
3	Vehicle Size	1/10th Scale	---	L	A,I
4	Vehicle Acceleration	2.5 ft/s^2	Max	M	T,A
5	Turning Radius	3.8 ft	+/- 0.5 ft	L	T
6	Vehicle Speed	15 ft/s	Max	L	T
7	CG Height	2 inches	Min	L	A
8	Works with Simulink	Yes	---	H	I
9	Suspension	Yes	---	M	I
10	Digital I/O Ports	3 Modules	Min	M	I
11	Independently powered wheels	2	Min	M	I
12	Tetherless	Yes	---	M	I
13	Autonomous / Hybrid Control	Yes	---	H	I
15	Battery Life	3 hours	Min	L	A,T
16	Protected Electronics	Yes	---	M	I,T

Specific requirements were developed through brief calculations to maintain similitude between our tenth scale model and a full scale model. Latency was based upon an average human reaction time to visual stimuli of 200 ms [23]. We decided that our system should perform at least 4x faster, reacting to stimuli faster and thus being able to catch things that a human may not. Maximum acceleration was based upon a full scale zero to sixty MPH time of four seconds. The turning radius was based upon a standard vehicle turning radius. Top vehicle speed assumes a full scale speed of 100 MPH. The minimum height of the center of gravity assumed a 6-inch scale wheel base (reasonable for RC cars), where rollover occurred when the normal force on the outside tire was reduced to zero and the vehicle was turning its minimum radius at maximum speed (note: this is a bare minimum case). The battery must be able to last the duration of a 3-hour lab period, with moderate use. Risk assessment was defined as how critical meeting each specification is to the overall success of the project. Specifications that were assigned a high risk (Withstands impacts, Works with Simulink, Digital I/O ports, User controller) are deemed to be absolutely necessary to make the vehicle platform perform its intended function and adapt to future needs. Parameters pertaining to similitude (Size, acceleration, speed, independently powered wheels) were medium risk. These parameters would not strictly determine the effectiveness of the platform, but still need to be met as best as possible. Furthermore, low risk assessment was given to specifications that were simply desirable, but not critical.

Compliance methods primarily fell under inspection when the specification was simply a Yes/No answer. However, similitude could be analytically determined with the Buckingham-Pi theorem. The vehicle's durability could be simply tested by subjecting it to extreme cases (i.e. top speed into a cinderblock), and the exposure of fragile electronics during roll-over could be assessed through a similar test. Most other specifications depend on the type of vehicle platform that is purchased, so an extensive amount of research will need to be dedicated to choosing the correct RC car.

3.3 BENCHMARKING RESULTS

Benchmarking with similar products is a great way to analyze the field of competition for their weaknesses and strengths. To help us find what we need to focus on, we hypothesized a 'current product' that would meet all the specifications outlined in the QFD. Similar products were not designed to explicitly meet our expectations; therefore, it is expected that our vehicle will outperform the competition in this form of benchmarking. From our benchmarking section of the QFD, we realized that the biggest challenges were in the low cost, realistic, and manufacturability customer requirements. Interestingly enough, these three columns have a deep relation with each other. Due to the lack of acceptable vehicle platforms, a user must create his/her own vehicle to live up to their own specifications. If a user aims to have a quality and realistic machine, many modifications and enhancements must be made to bring the product up to par, leaving the user spending more money and time than they would like. In order to beat the competition, our project will need to be especially careful with these three customer requirements so it can maximize the score in these areas.

Customer requirements where we were strong included being portable, and having hybrid autonomous control capabilities. As far as being portable goes, this requirement is the easiest for an RC car to achieve given its scale and function. The hybrid autonomous control is a bit more of an interesting design challenge – the car needs to be able to adjust user input for minor corrections and fully override user inputs in critical situations. With a microcontroller to take input from the RC receiver, it is a manageable challenge to adjust or ignore this input.

Most of the benchmarked products seemed to rate in the middle ground for each category. In fact, only the Independent Wheel Drive project seemed to reach the maximum, as well as minimum, allotted point total in any category. The fact that most projects are found in the middle-ground seem to represent that those projects have different goals to meet. These findings reinforce the fact that there is no product that fills the role of our proposed product. Some projects barely cover all the requirements, and others find themselves in all-or-nothing scenarios, but our product should do well in all required areas.

Specifications developed in the QFD include: withstand impacts without damage, visible latency, vehicle size, top vehicle speed, works with Simulink, suspension, digital I/O ports, tetherless, independently controlled wheels, and user controller. A vehicle that can withstand impacts is considered to be able to survive crashes and rollovers at top vehicle speed. Visible latency is dependent upon the capabilities of the microcontroller and its ability to run basic algorithms without lag. The top vehicle speed is a parameter to maintain similitude and is calculated as a full scale vehicle traveling at 100 MPH. Based upon our sponsor discussions, it is imperative that the microcontroller is compatible with Simulink and the algorithms can be developed in this program. Digital I/O ports are necessary to enable adaptations to the platform. A tetherless vehicle is also necessary for ease of use and more portability. A vehicle platform capable of testing electronic stability and traction control must have at least two independently controlled wheels for braking and accelerating. Finally, a user controller is necessary for the hybrid manual/autonomous interface. The list of specifications was assessed, modified and expanded to create our final specifications table.

3.4 SCOPE

The main scope of this project is simple and straightforward: to create an intelligent vehicle platform to participate in and win the 2017 ESV competition. Most projects entered in this contest are solving a

straightforward and tangible safety problem that can be applied to bigger projects in a reasonable amount of time. Our finished product will instead build upon the promise of what is to come. This will require convincing the judges that the platform that we made is more important for future of human safety than the solution another team made to immediately help save lives.

The other goal we hope to reach is to inform the design and construction of eight more platforms for Dr. Birdsong's new course. We initially believed this goal to be the main focus but soon realized a platform for the ESV competition could easily be modified for this course.

3.5 BOUNDARY SKETCH

Located in Appendix B, the Boundary Sketch gives a visual image of what the project will be held responsible for. We included the purpose of our project in the sketch as well to clarify our objectives even further. Creating an exact model for the upcoming design class is not within our scope; however, we make a design from which a class model can be established while maintaining our focus on competing in the ESV competition. The aim of this project is not to design new, complicated control algorithms. Instead, we will only pick a few basic algorithms to show off the ability and the platform foundation for which the former idea is possible.

The scope also includes advanced features such as manual/remote capabilities, an adjustable center of gravity, and realistic vehicle dynamics. Dynamic components such as the manual/remote capability will allow future algorithms the opportunity to implement computer driven adjustments to each motor. The platform as a whole will be powered by a rechargeable battery. An adjustable center of gravity will allow for flexible vehicle dynamics when interested students need to experiment with how a particular algorithm interacts with the center of gravity. We expect to strive towards the best response time possible for the system by limiting the latency wherever possible for the components used in the platform.

Other key scope features include durability, accessibility, manufacturability, and the connection between the platform and a Simulink-capable computer. Durability corresponds to our need to make our platform robust while accessibility refers to the need to make components on the platform convenient for modifications. Manufacturability represents an extension of the purpose and refers to the idea of creating a plan for the class model that can be easily built. The connection to download Simulink code to the platform needs to be robust, fast, and as convenient as we can make it.

4. DESIGN DEVELOPMENT

Following a structured design process, we have identified customer requirements and specifications, developed various concepts, and completed initial analysis and design for the selected concept.

4.1 THE DESIGN PROCESS

To ensure an excellent final product, a structured design process will be used. Figure 4 shows the Cal Poly Mechanical Engineering senior project process, as outlined in the *Student Success Guide* [24].This flowchart indicates required processes and deliverables that make up the senior design project.

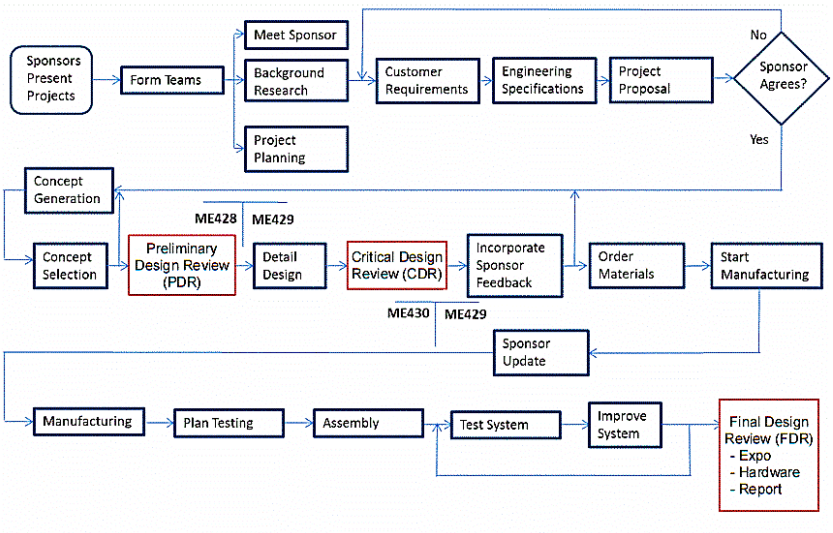


Figure 4: Cal Poly ME senior project flowchart.

We first met with the sponsor to discuss the details of the project. After that we began background research on both existing systems and potential components of our solution. We concurrently worked to identify customer requirements and expand those into engineering specifications. After bringing our research, ideas, and specifications to our sponsor, we entered the brainstorming and preliminary design phase. From there, we built upon our work with detail part design and systems architecture. We have completed the mechanical and systems design, and are ready to begin construction and implementation of our design.

Going forward, we have to construct a prototype and test it against our engineering specifications and customer requirements. We will continue to document our design philosophies, lessons learned, and retrospective changes we would have made to ensure that this project is useful for those who would adapt it. If time permits, we will modify our prototype design based upon our testing and knowledge gained throughout the course of the project. Since the ESV competition is a major interest of this project, key deliverables and milestones for the ESV competition are laid out in the Management Plan section. The regional evaluation is in the beginning of March; while we will not have a functional prototype at this point, we will have systems demonstrations and sensor interactions prepared to demonstrate. Fortunately, the international conference is not until June, providing us the time to continue developing and refining the prototype.

We have encountered and processed several design challenges, and we expect to encounter more throughout the course of this project. The first method that will be utilized on a regular basis is to confer with other group members. Each member has unique skills and can offer alternate perspectives. However, problem-solving is never this simple, and more extensive measures will likely need to be taken. Problem solving is a lot like the design process, just scaled down. The problem should be defined and causes determined. Next,

brainstorming solutions, preferably with a group, can generate a wide variety of ideas. Assessing each idea and choosing the best solution requires a bit more thinking and is a very important step in the problem-solving process. Once all of this has been completed, the solution should be implemented, and modified if the need arises.

4.2 CONCEPT DEVELOPMENT

In developing an idea of what we needed our prototype to be we needed to consider a few unique ideas. The first of which is that we are not developing this for ourselves but for other students and researchers to use. That means it needs to have a high level of usability and accessibility in the design. Along with building it for usability, we also needed to construct a very robust and enduring model. Since our group won't be there to fix any parts, we needed to make a model and will last through 2+ years of student use. This is no small feat if you've seen how students treat lab equipment.

Another consideration would be the true performance of the vehicle and how many detailed features we can install for in-depth development. This third performance aspect creates somewhat of a Venn-diagram triangle with usability and durability. We want all three but we won't be able to significantly sacrifice one aspect for the sake of another.

A structured concept development process ensures a variety of potential solutions can be generated, assessed, and validated. The first step to generating good design concepts was to brainstorm. The overall concept of our project was already defined as a small scale vehicle platform. This meant that the brainstorming was primarily focused on specific subsystems and functions that the vehicle needed to perform. The first brainstorming session focused on developing concept ideas for the different subsystems. The subsystems/functions were identified as follows:

- Protective System
- Electronics Layout
- Vehicle Protection
- Types of Sensors
- Center of Gravity Adjustment
- Algorithms
- Braking Systems

To generate the new concept ideas, we each individually came up with concept ideas that fell within each of the subsystems listed above. We transcribed our ideas on sticky notes and placed them on an empty wall in no particular order. Three sessions were conducted that lasted for two minutes each. The results from this brainstorming session can be seen as a morph chart in Appendix C**Error! Reference source not found.**.

Another brainstorming session focused on a functional decomposition of what makes a vehicle. This is what helped us to explicitly define the subsystems of our concept. To be a complete scale model, our vehicle would need a drivetrain, braking system, electrical system layout, autonomous/manual control system, sensor integration, chassis, protective frame, wireless remote control, and a Simulink template.

We then conducted a brain sketching activity to help to visually generate concepts of the subsystems in a variety of different ways. We each started off with a specific system and sketched a concept. The concepts

were rotated between the three of us until we had developed a concept for each subsystem. Our drawings were not professional by any means but this helped us to flush out ideas at a high pace. An example of a drawing can be seen in Figure 5 and the rest of the production can be seen in Appendix C.

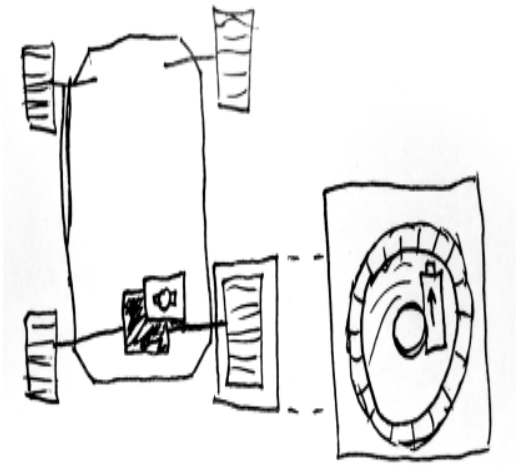
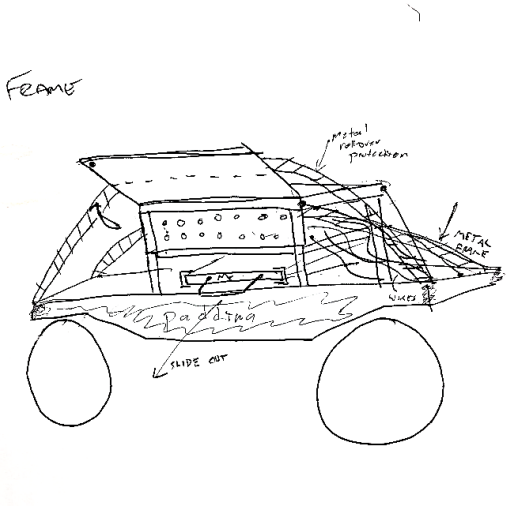
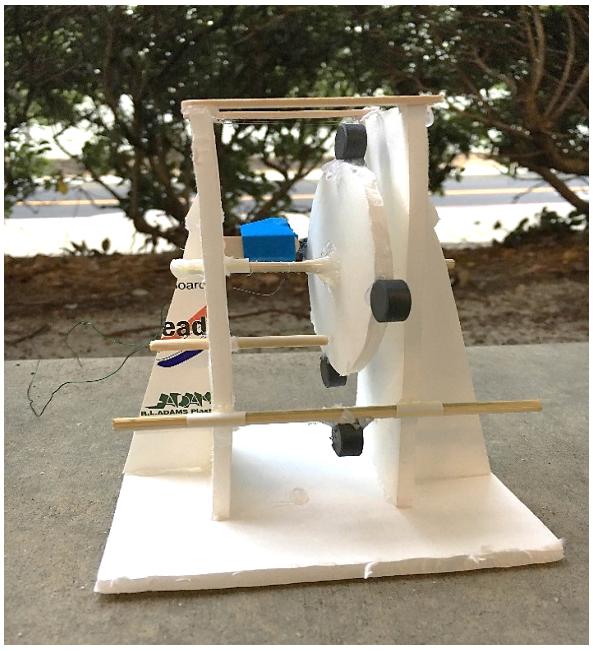


Figure 5: Braking Subsystem Example

Eventually when we had burnt ourselves out of subsystem drawings we decided to try to visualize different subsystems working together. We used the same brain sketching model as described above to try to draw a full system. Depicted in Figure 6a below is an example brainsketch of a full system design.



(a)



(b)

Figure 6: Brainstorming and prototyping examples for (a) brain sketching activity and (b) braking method test stand.

After generating concept ideas and sketching them, we next made physical prototypes. Our prototypes consisted of a braking method test stand (Figure 6b), an electrical layout, and several protective housing systems. The braking stand had a mounted disk that could be braked with an actuated rod that modeled the friction solenoid, a permanent magnet, and an electromagnet. Results from the braking test led us to believe that any magnetic braking system would not be feasible. The electrical layout was a piece of foam board cut in the shape of the vehicle chassis, where we could move components around and place them with pins. The protective housings consisted of a wire cage, a hinged box, and a drop out electronics board. Images from the prototyping session can be seen in Appendix C.

4.3 COMPARISON OF CONCEPTS

After the brainstorming sessions were complete, we began to eliminate ideas with Pugh matrices for the RC car selection, microcontroller selection, potential algorithms, braking methods, protective housing, CG modifications, and electrical layout. The complete set of matrices can be seen in 0, while a discussion of each matrix is found below.

4.3.1 Initial RC Car Comparison

Six initial RC cars were identified from leading manufactures. The main criteria for the cars that were selected was that they have suspension and a source for purchasing individual replacement parts. The six selected vehicle were then compared against each other, with a $1/16^{th}$ scale Traxxas E-Revo as the baseline. Despite being suggested to build a $1/10^{th}$ scale prototype, a $1/16^{th}$ scale car would be a cheaper alternative and would allow us to easily outfit an independent motor system with a custom chassis. A $1/16^{th}$ scale 4WD vehicle would be cheaper than its $1/10^{th}$ scale 2WD counterpart and the 4WD would already include constant velocity shafts for the front two wheels. This could save us a considerable amount of work integrating a custom drivetrain. Compared to HPI racing and Axial, Traxxas vehicles have a much wider array of replacement parts, which is a very important parameter in choosing which RC car to purchase. Results from the Pugh Matrix allowed us to eliminate most of the vehicles, where we narrowed it down to the $1/10^{th}$ scale Traxxas Slash and $1/16^{th}$ scale 4WD Traxxas slash. These vehicles were included in the final decision matrix as different system level concepts. We also decided to entertain the idea of a custom chassis. This would allow us to design a shape for the chassis that can explicitly fulfill our space requirements. If designed correctly, a custom built chassis would also result in a sleeker, more professional and high quality build. It would also allow a more precise and consistent result, as we would not have to drill mounting holes into the stock chassis by hand.

4.3.2 Microcontroller Selection

In our microcontroller analysis, we compared the Arduino Mega 2560, BeagleBone Black, and the Raspberry Pi 3B. We quickly determined that the Arduino Mega 2560 did not have the clock speed or RAM quantity that we were looking for. It also lacked compatibility with the Simulink computer vision toolbox. While computer vision is not directly in the scope of our project, it is a system that many find interesting and our platform may see computer vision research in the future. This left the BeagleBone and the Raspberry Pi. These two systems are very comparable and both had some beneficial attributes that the other

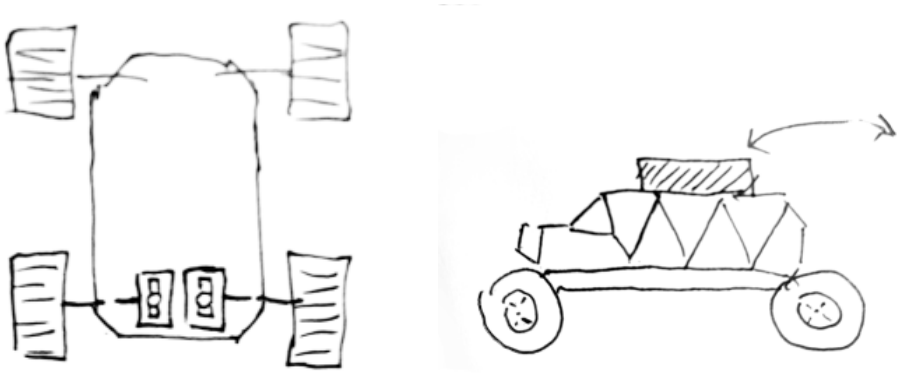
didn't. While the Raspberry Pi is slightly less expensive than the BeagleBone, the difference is a small portion of our overall project; we decided to focus on assessing which would work better. The BeagleBone had nearly twice as many ports for future sensor connections than the Raspberry Pi. However, The Raspberry Pi had a faster CPU, more RAM available, and had multi-processing functionality. The multi-processing feature for the Raspberry Pi is what ultimately tipped the scales. In a complicated and intense control algorithm, many tasks will need to be computed as fast as possible and the multi-processing feature means that we can do more than one task at a time. This attribute could greatly speed up an intense control algorithm and had a substantial influence in our Microcontroller selection.

4.3.3 Baseline Algorithms

The Pugh Matrix for potential algorithms helped us to determine what we wanted our platform to be able to run without any adjustments or add-ons. Essentially, these are the simplest, most practical, and interesting systems that we want to be able to test. These systems were determined by the matrix to be adaptive cruise control, electronic stability control, multi-vehicle management, and collision detection. Reassessing these results led us to believe that a lane following system would be useful on the baseline model. Multi-vehicle management and collision detection systems would require more complex systems that would be interesting to incorporate by adding different modules, but is not something that should be required by the simplest model. As a result, our baseline model will simply have adaptive cruise control capabilities through a 1-D distance sensor, lane following through a light sensing module, and electronic stability control through the IMU and independently powered wheel systems.

4.3.4 Braking Method

The braking method was seen as one of the most important brainstorming topics. It is imperative the vehicle can brake at least two wheels independently in order to test ESC algorithms. Our prototyping session essentially allowed us to discard of any magnetic system, however, we still included these in the initial Pugh matrix. As expected, the results from the matrix helped us to narrow down braking concepts to a friction solenoid and regenerative braking through independently powered motors. A two motor system (Figure 7a) would be much simpler than a four motor system, but would not provide as much control when compared to a four motor system. These concepts were next incorporated into our final set of concepts, which was analyzed in a decision matrix.



(a) (b)

Figure 7: Design concept sketches for (a) 2 motor power system and (b) rack with weights mounted on wire cage.

4.3.5 Protective Housing

Electronic components need to be protected in order to prevent irreparable damage or unnecessary replacements. During brainstorming we came up with several methods for protecting the MCU and associated central components. A wire cage, hinged shell, drop-out assembly, and cushioned bumpers were each considered in the Pugh matrix. Results determined that a drop-out assembly and boxed hinge system were potentially the least effective and most impractical options. We predict the wire cage to be easy to implement and adjust, as roll cages are available for many RC cars from third-party manufacturers and a demo we recently looked at enhanced these suspicions. A bumper system also usually comes equipped on off the shelf products. These two systems may be used in conjunction with each other and not require a lot of modification, which would be ideal.

4.3.6 Center of Gravity Modification

Raising the center of gravity is necessary to allow the vehicle to rollover and mimic standard vehicle dynamics. We considered using an adjustable center of gravity through some sort of an angled boom, but this idea was unnecessarily complex. Simply using a rack mounted to the wire cage (Figure 7b) would be a very simple, and even adjustable method of approach. If a custom chassis is used, we may also implement an adjustable chassis height through spacers. The size of the spacers could be left up to the user and would provide valuable customization through the height of the chassis.

4.3.7 Electronics Layout

Electronics layout is an interesting design challenge – we must interface with sensors across the car, motor(s) for the drivetrain, and the central processing unit, all while being adaptable to other sensors and components. For adaptability, connecting the microcontroller to a breadboard may be ideal as it provides the ability to plug in and remove components and wires at the user's discretion. Unfortunately, this leads to inconsistent connections, loose wires that complicate cable management, and an inconsistent user experience. Another option is to create a 'stack' of plates and attach individual components to that. While this enables better cable management, it is space intensive and not as adaptable. Instead of a stack, the components could be affixed to the chassis and permanently interconnected, but this again loses adaptability and complicates manufacturing. The most attractive choice for us was to use a motherboard / daughterboard system. The microcontroller will act as a daughterboard, and a motherboard will be made of single PCB or perfboard will be attached to the microcontroller's pins. We will attach the IMU and other sensors to motherboard directly, and break out multi-pin connectors for connecting motors, baseline sensors, and user-selected sensors. This layout gives us the most compact system and provides excellent durability and consistency while maintaining adaptability. The use of multi-pin connectors also supports effective cable management.

4.4 SELECTED CONCEPT

After narrowing down our subsystem concepts and deciding on a microcontroller, compatible algorithms, protective housing, and an electrical layout, a collection of system concepts was generated. The next task was to decide the final RC car and chassis type, braking method, and type of CG modification. Table 4

defines the eight distinct concepts that were compared in our decision matrix, which can be found in Appendix D. It should be noted that the weight factors generated from the QFD were redistributed for our version of the decision matrix. This is because at this point in the decision process, subsystem designs were already determined, meaning that all concepts would have the same score in certain sections.

Table 4: System level concept descriptions.

	Function						
System	RC Platform	Microcontroller	Compatible Algorithms with Baseline Design	Braking Method	Protective Housing	CG Modification	Electrical Layout
1	Traxxas Slash	Raspberry Pi 3	ACC, ESC, Lane Following	Friction Solenoid	Wire Cage	Rack with weights	Mother/Daughterboard
2	Traxxas Slash w/ custom Chassis	Raspberry Pi 3	ACC, ESC, Lane Following	Regenerative (2 x Motor)	Wire Cage	Rack with weights	Mother/Daughterboard
3	1/16th Slash w/ custom chassis	Raspberry Pi 3	ACC, ESC, Lane Following	Regenerative (4x Motor)	Wire Cage	Rack with weights	Mother/Daughterboard
4	Traxxas Slash w/ custom Chassis	Raspberry Pi 3	ACC, ESC, Lane Following	Regenerative (4x Motor)	Wire Cage	Ajustable Chassis	Mother/Daughterboard
5	1/16th Slash w/ custom chassis	Raspberry Pi 3	ACC, ESC, Lane Following	Friction Solenoid	Wire Cage	Adjustable Chassis	Mother/Daughterboard
6	Traxxas Slash	Raspberry Pi 3	ACC, ESC, Lane Following	Regenerative (2 x Motor)	Wire Cage	Rack with weights	Mother/Daughterboard
7	Traxxas Slash w/ custom Chassis	Raspberry Pi 3	ACC, ESC, Lane Following	Regenerative (4x Motor)	Wire Cage	Rack with weights	Mother/Daughterboard
8	1/16th Slash w/ custom chassis	Raspberry Pi 3	ACC, ESC, Lane Following	Friction Solenoid	Wire Cage	Rack with weights	Mother/Daughterboard

The results from the decision matrix indicated that concept 2 was the best system level design for meeting our specifications. Upon inspection, this seems like a very reasonable result. We would expect a $1/10^{th}$ scale vehicle with a custom chassis to be physically robust, and easily adaptable. Incorporating a two motor system would also be much easier than a four motor system and even the friction solenoid method. However, we have decided that a four motor system would be much more beneficial in the long run, allowing more in depth development of stability and traction control algorithms. A sketch of the selected concept can be seen in Figure 8.

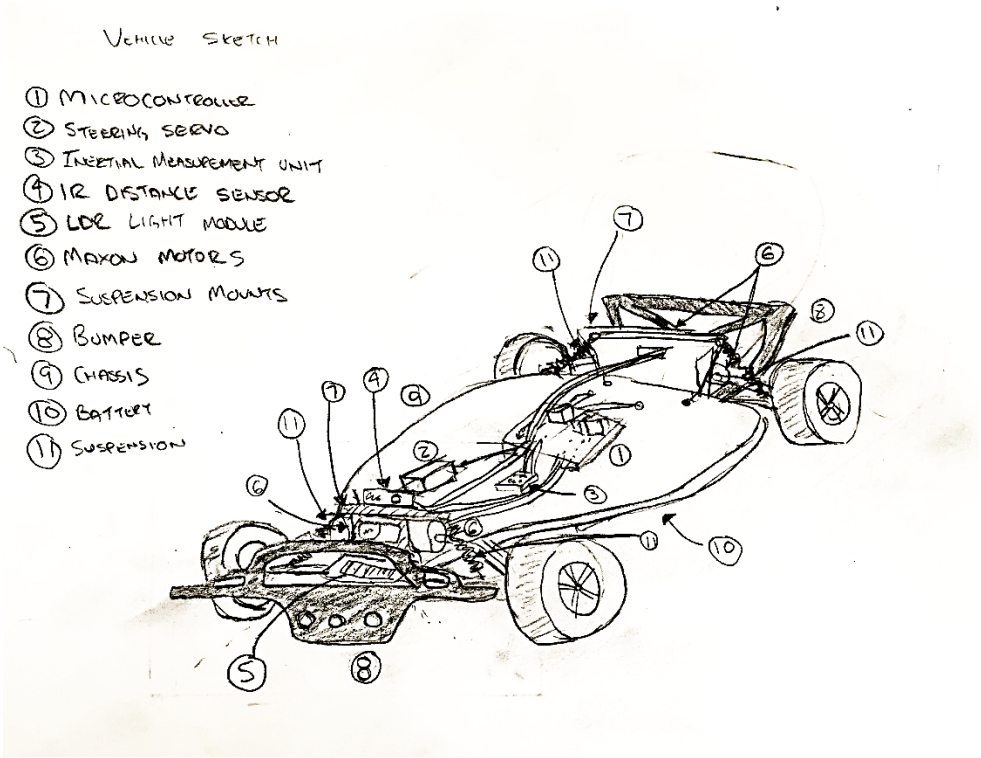


Figure 8: Final Concept Sketch of Modified RC Car. Not Shown: Wire cage for component protection and weight rack for CG modification

In our selected design, a custom chassis would be easily repairable, and if designed correctly easily manufactured. When considering cost concept 2 is relatively average, but cost is not a driving factor. In conclusion, our final design will be adapted from a $1/10^{th}$ scale Traxxas Slash with a custom chassis and four independent motors. Though our initial thought was to use the 2WD Traxxas Slash, the inclusion of four independent motors makes the 4WD Slash a more attractive chassis. While more expensive, it contains all of the drive-shafts and steering geometry required. The microcontroller will be a Raspberry Pi 3 and be able to link with sensors for ACC, ESC, and lane following control systems. The microcontroller will easily be able to intermediate control algorithms without any visible latency. ACC systems will be made possible using an ultrasonic linear distance sensor. Yaw rate and axle angular velocities necessary for stability control will be determined by the IMU and halls sensors, respectively. A wire cage will protect the fragile components with a rack on top for a weighted CG modification. A mother/daughterboard layout will also be utilized. A secondary microcontroller will be present on the motherboard, allowing for additional input / output pins and handling of time sensitive tasks. Parts will be easily sourced for any repairs from the Traxxas website.

We will work to compartmentalize our design by having our drivetrain, mechanical system, and electrical system all work on their own. While they will work together in our final design, this compartmentalization will allow easier adaptation of our design work for future researchers. While our design does well to meet our customer requirements and specifications, it does come with challenges, including space constraints, custom PCBs, manufacturability concerns, and more. Figure 9 presents the challenges of specific subsystems and how they play in to our compartmentalized design.

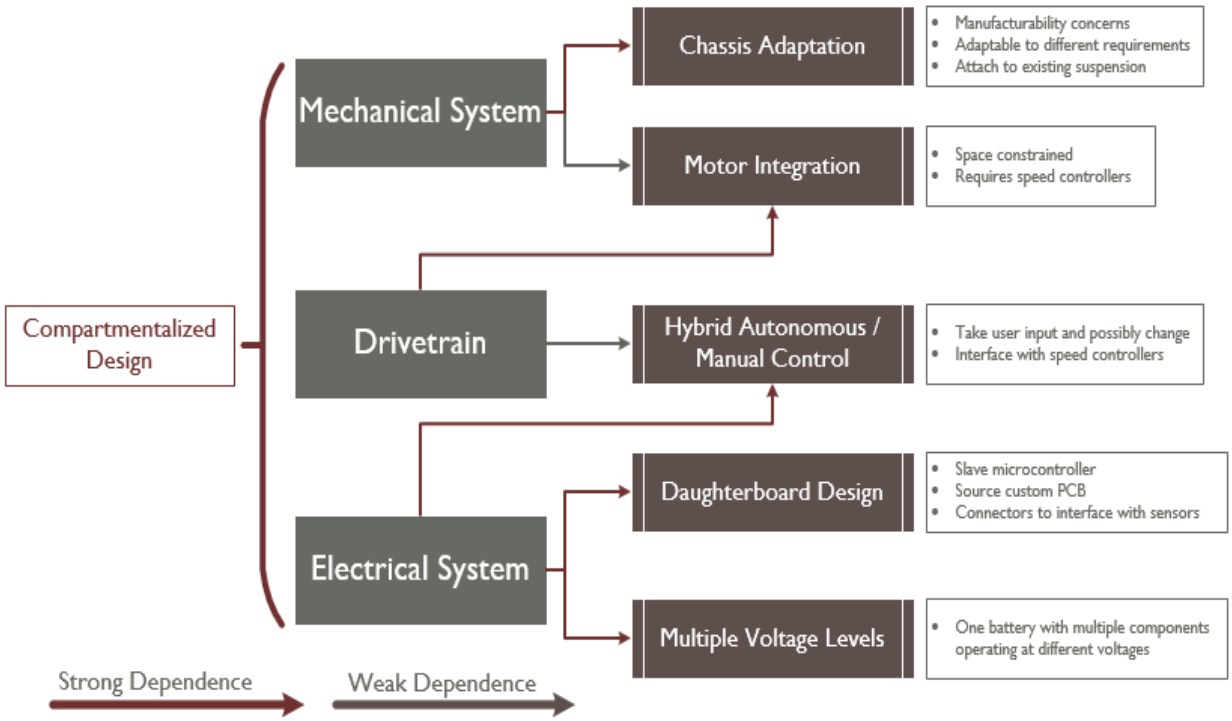


Figure 9: Design challenges with the specific subsystems that they stem from

4.5 PRELIMINARY DESIGN ANALYSIS

Now that we have selected a concept, analysis must be completed to ensure that the design meets our requirements and is feasible to manufacture within the time and budget constraints. In addition to identifying if our selected components will work, our preliminary analysis will include similitude calculations for dimensions and forces to show the strengths and weaknesses of our design as an advanced rather than preliminary research platform.

4.5.1 Dimensional Similitude

When designing a scale model, it is very important to maintain relative similitude between the full scale model under investigation and the small-scale model used to gather information. As a result, a similitude analysis should be completed. For this specific case, similitude is not necessarily a critical issue, as the main purpose of the platform is for experience and testing the feasibility of vehicle algorithms, not for exact modeling of a full scale vehicle. Nonetheless, we want to design a platform as realistic to a full scale vehicle as possible. When creating a scale model, it is possible to scale different base quantities, primarily being mass and length. Scaling by length puts the mass at a different scale, and vice versa. To be thorough, we analyzed the dimensional requirements of a vehicle scaled on both mass and length, just mass, and just length. These values were compared to that of a standard vehicle, for our purposes, a Ford Edge. The analysis can be found in Appendix E, and the results are tabulated below:

Table 5: Results of dimensional similitude analysis, with RC car for comparison

Parameter	Wheel Diameter (in)	Wheelbase (in)	Track Width (in)	CG height (in)	Power (hp)	Weight/Power (lb/hp)
Typical Vehicle (Ford Edge)	**20**	**112**	**65**	**36**	**250**	**16**
All Parameters Scaled	2.0	11.2	6.5	3.6	25	16
Length Scaled	2.0	11.2	6.5	4	0.25	16
Weight Scaled	11.1	88.2	33.4	18.6	25.0	16
RC Car (Slash 2WD) [25]	2.20	13.20	11.65	-	0.60	1.97

Following the similitude analysis, it becomes immediately apparent that scaling a vehicle by weight would result in large dimensions that are impractical for the purpose of this project. The most reasonable method is to scale the vehicle by its dimensions, and simply purchase and off the shelf $1/10^{th}$ scale RC car. When scaling by length, mass scales in a cubic fashion – our $1/10^{th}$ length scale vehicle will have $1/1000^{th}$ mass scale. The smaller mass scale has implications on both dynamic similitude and power. The Traxxas vehicle is significantly more powerful per pound than a full scale car, which is something to be aware of when attempting to address similitude. We can maintain a more similar power by using the lower portion of the motor's performance curve and not letting the motor reach its full potential. Operating the motor at this point reduces the motor efficiency, but also reduces the overall power consumption. Since efficiency is not a primary concern for our project, artificially limiting the motor power will not detract from our final design.

Table 6: Comparison between ideal dimensionally scaled model and Traxxas Slash

Parameter	Traxxas Slash	Dimensions Scaled	Percent Difference
Weight (lb)	4.76	4	19
Wheel Diameter (in)	2.2	2.0	10
Wheelbase (in)	13.2	11.2	19
Track (in)	11.65	6.5	80
CG Height (in)	-	4	-
Power (hp)	0.6	0.25	140
W/P (lb/hp)	1.97	16	88

Table 6 provides insight to what parameters are going to be important to pay attention to when modifying an off the shelf RC car. It becomes obvious that the vehicle will be overpowered, but another important aspect will be the center of gravity. The ratio of the track length to the wheelbase is much greater for the Traxxas vehicle than a scaled model. This means the Slash will be more difficult to roll-over than a standard vehicle. Normally this would not be an issue. In fact, the Slash is probably designed not to roll-over. However, for our experimental purposes, we want a car that can flip and rollover in ways similar to many standard cars on the market. This can be overcome by simply modifying the center of gravity of our $1/10^{th}$ scale vehicle.

4.5.2 Braking Forces

One of the main requirements of the vehicle platform is that it has independently controlled wheels. Standard RC cars do not come equipped with disc brakes, or independently powered wheels, meaning that an off the shelf vehicle would have to be equipped with some sort of a mechanism to brake at least two wheels. During brainstorming, we developed several methods for braking an RC car, one of which was a friction solenoid. In this concept a solenoid equipped with a rubber tip would actuate toward a disk mounted on the wheel shaft, causing a reversing torque that would be transmitted to the ground as a force. We were interested in the feasibility of using a solenoid as a brake, so brief calculations were conducted and can be seen in Appendix E. Through these calculations, it was found that a solenoid would need to generate about 6 lbf to successfully stop an RC car traveling at 15 ft/s. This is a reasonable force for a solenoid to generate, meaning we could continue pursuing it as a possible braking solution.

4.5.3 Motor Selection

Collaborating with Charlie Refvem, a prospective graduate student interested in completing work consistent with our goals, we have selected motors to use on each wheel. We plan to use Maxon flat brushless motors, Part Number 397172 [26]. These are 24V, 70W brushless motors that have built in hall effect sensors (for low resolution position + speed) and the option for a built in high-resolution encoder. These specific motors were selected due to their unique design for high torque, relatively low speed, profile, and low starting threshold. Standard brushless motors have extremely low starting torques and extremely high nominal speeds, reducing their applicability for ground vehicle design. While a gear reduction would often be used in an RC car to either increase torque or speed, these motors were selected specifically to not require a gearbox. A gearbox at each wheel would add complexity to the drivetrain, reduce the amount of space we have to work with, and the high torque of the motor may wear down the gears, creating another potential point of failure. Additionally, gearboxes would add backlash and inertia – two things that make electronic stability control more difficult. We confirmed that these motors would be appropriate for our platform by analyzing their torque and speed outputs, calculations for which are found in Appendix E. We found that we could produce a nominal speed of 43.3 mph and an acceleration of 9.5 ft/s^2. While this exceeds our specifications, we can control the motors to a lower speed and torque.

4.6 CONCEPT REFINEMENT

As we exited the preliminary design phase, we began refining the specifics of the motherboard, drivetrain system, and more. At this stage, we were beginning to think about manufacturing and documentation. Figure 10 presents our initial plan for work from PDR through April with Critical Design Review and ESV Prototype Evaluation noted as important deliverables.

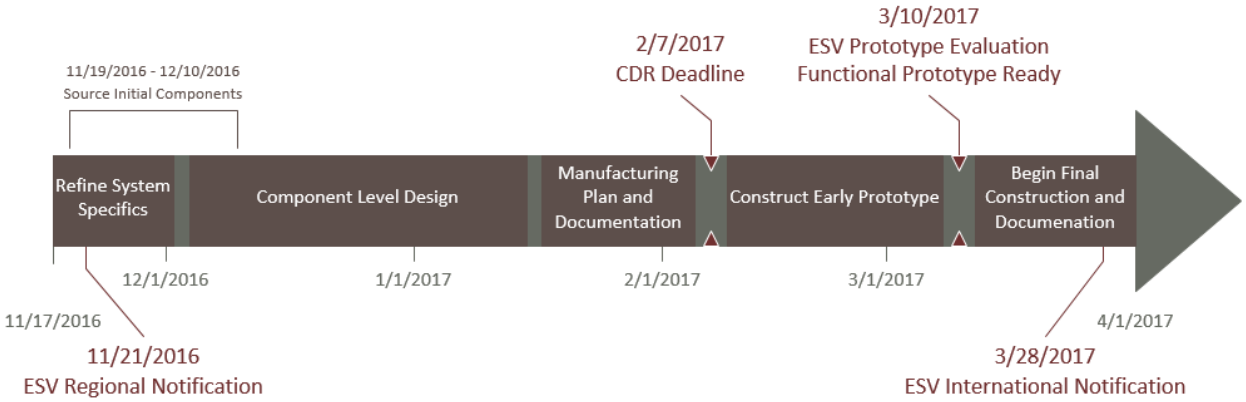
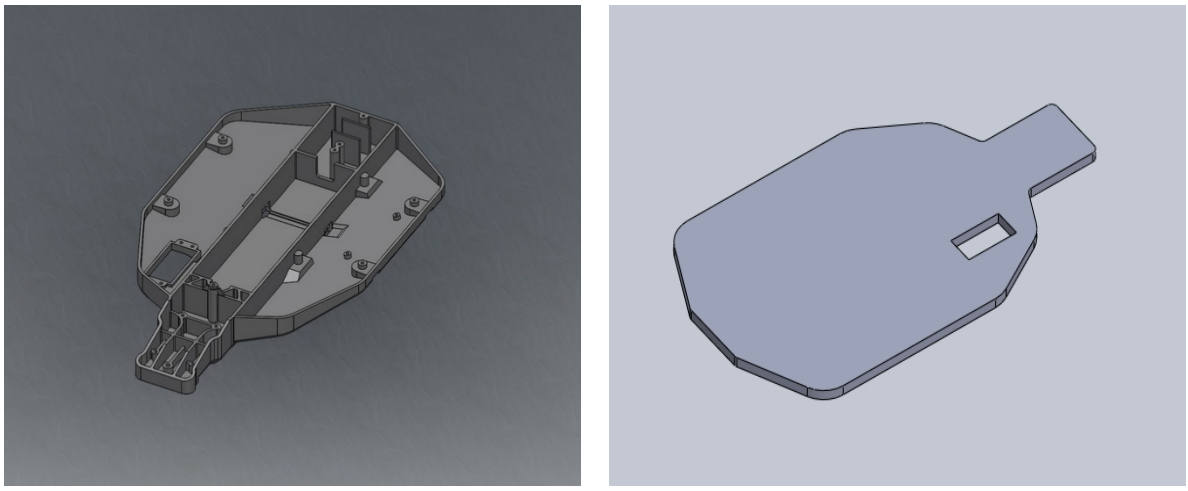


Figure 10: Early planning timeline of milestones through ESV International Notification

4.6.1 Design Planning

A preliminary solid model for our custom chassis has been created based upon the dimensions of the Slash and can be seen below in Figure 11. More detail will be added once the Traxxas Slash is purchased and inspected. We will create geometry and attachment points that alight to our specific system's needs. This will support adaptation and recreation of the design, support consistent wire management, and enable us to create space for sensors that the user may add.



(a) (b)

Figure 11: Solid models of (a) Stock Traxxas Slash (2WD) chassis and (b) early mockup of custom replacement chassis.

In addition to the custom chassis, we must design motor and component mounts, shaft collars for the motors, and our motherboard.

For the motor mounts, we need to integrate the mounting locations to incorporate the Traxxas suspension. Our motor selection does result in us widening the drivetrain due to the possible locations for u joint / shaft

couplers. To complete this in a way that fully supports the stock Traxxas suspension / drive system, we will review the geometry of the Traxxas Slash and create a full drivetrain solid model. Splitting the geometry down the middle, we will expand the hole layout and u-joint position to accommodate the additional space requirements. Using calipers, we reviewed, documented, and recreated the Slash's front and rear drivetrain geometry in Solidworks. The solid model of the Slash's front drivetrain is presented below in Figure 12.

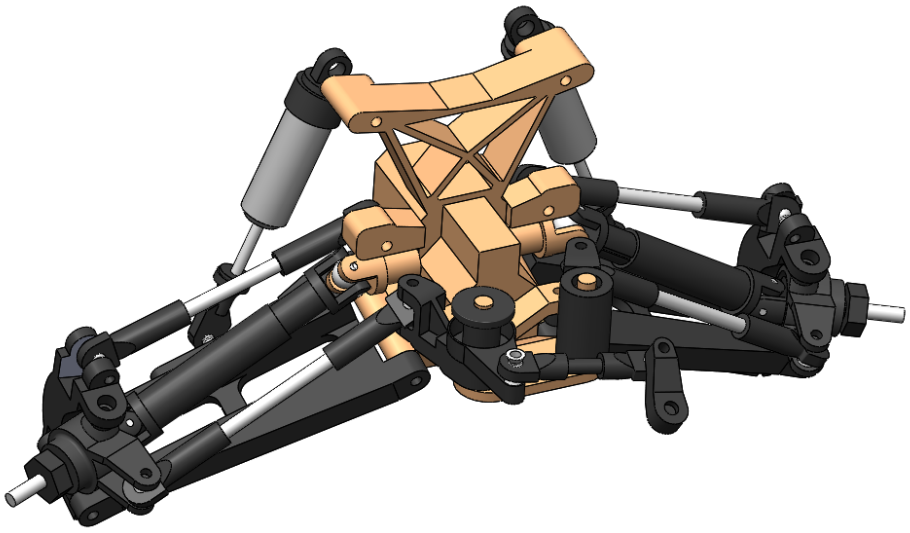


Figure 12: Stock front drivetrain of Traxxas Slash – tan components will be replaced with custom parts that accommodate the Maxon motors.

Our adaptation maintains the relative location of each side's mounting points to other mounting points on the same side, but increases the dimension from the mounting points to the centerline. This provides more space to integrate the motors.

As we developed the geometry to integrate the Maxon motors in to the drivetrain, we found that we were having to increase the track width more than anticipated. Figure 13 compares the stock and custom drivetrain mounts and shows the increase in track width.

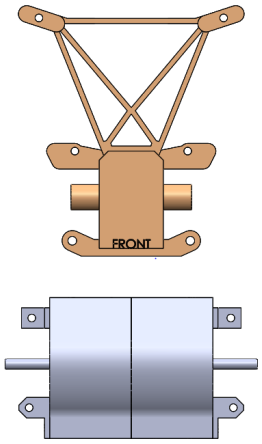


Figure 13:Comparison of stock (top) and custom rear drivetrain mounts. Note that the rear suspension uprights are not pictured on the custom mount.

Since the Traxxas Slash already had an aspect ratio more square than real cars, we elected to also increase the wheelbase to refine the dynamic properties that stem from aspect ratio. We found that our new geometry reflected a $1/7^{th}$ scale vehicle. RC cars often have oversized suspension and wheels, so the increased size improves the drivetrain similitude.

As a result of the increase in vehicle size, we are no longer able to easily implement an aftermarket protection system for the Traxxas Slash. Through the prototyping process we will continue to search for and develop ideas for an effective protection system, but we have withdrawn the protection system from our critical deliverables. This is appropriate because our system is intended to demonstrate the capabilities of a small scale vehicle platform and not to be a final design. While we continue to search for an effective protective system, we will complete thorough stationary testing on the controls system and low speed dynamic tests. This methodology will also us to test and refine the system without putting it in jeopardy.

5. FINAL DESIGN

Our final design implements a custom drivetrain and control scheme while retaining the Traxxas suspension, drive-shafts, steering configuration, and radio receiver. We will provide a Simulink template that includes custom blocks receive data from the sensors and output control data to the steering servo and motor on each wheel. The user can upload designed Simulink algorithms to the on-board Raspberry Pi, which interprets sensor data to produce the programmed response. Figure 14 below pictures the car and component placement – cabling and sensors are omitted.

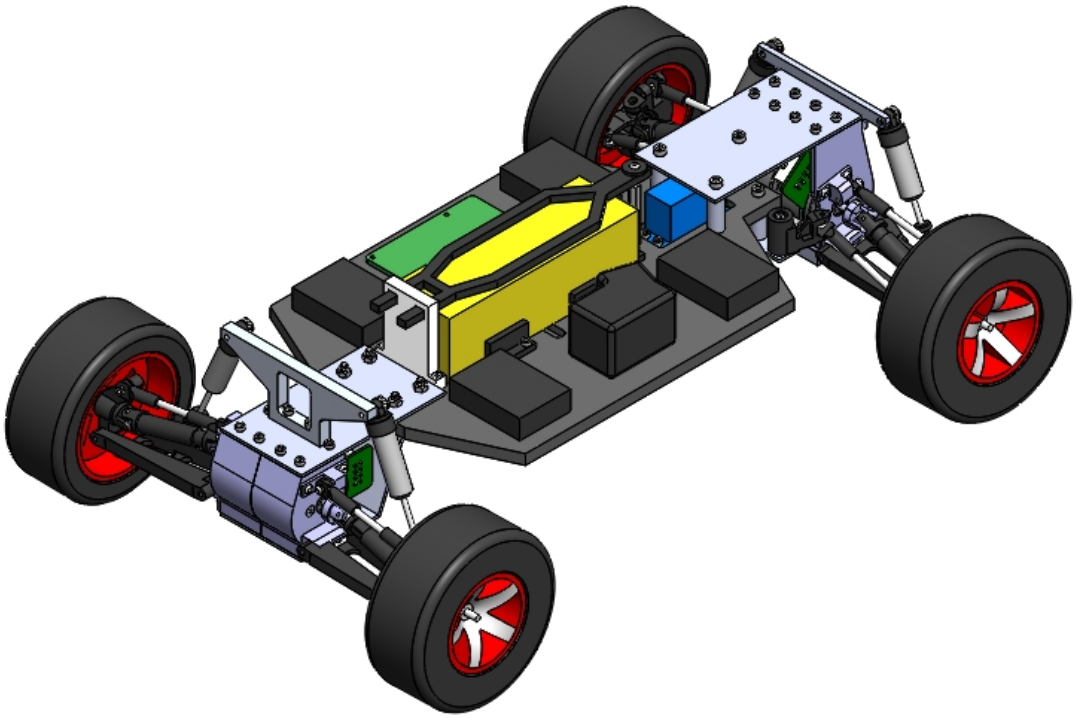


Figure 14: Isometric view of final design with approximate shapes for electronic components and cabling omitted.

This design accomplishes our primary objective – provide a simple accessible platform for developing intelligent vehicle control. It provides an Inertial Measurement Unit (IMU), Ultrasonic Rangefinder (URF), camera, and motor encoders to provide the spatial data required to implement many intelligent vehicle control regimens. Through the designed control algorithms, the user will be able to adjust the remote control inputs and control the steering angle and manipulate the position / speed of each wheel. Our custom motherboard attaches directly to the Raspberry pi and offers connections to the battery, motor drivers, and sensors. Additionally, it can offer data-logging support and has an RGB indicator led to visually notify the user of fault states. For additional safety, we have included a remote cutoff switch that will disable the motors.

5.1 MECHANICAL ARCHITECTURE

5.1.1 Structural Design

The new components that we designed were typically static members, which means they would need to be able to support impulses and static forces but would not be required to translate or rotate. The two exceptions are the shaft couplers and the new steering linkage. We tended to take a conservative approach with our design, with similar geometry to the Traxxas parts we were replacing. The majority of the parts will be manufactured from aluminum, while the chassis and non-critical components will be made out of Nylon and PLA, respectively.

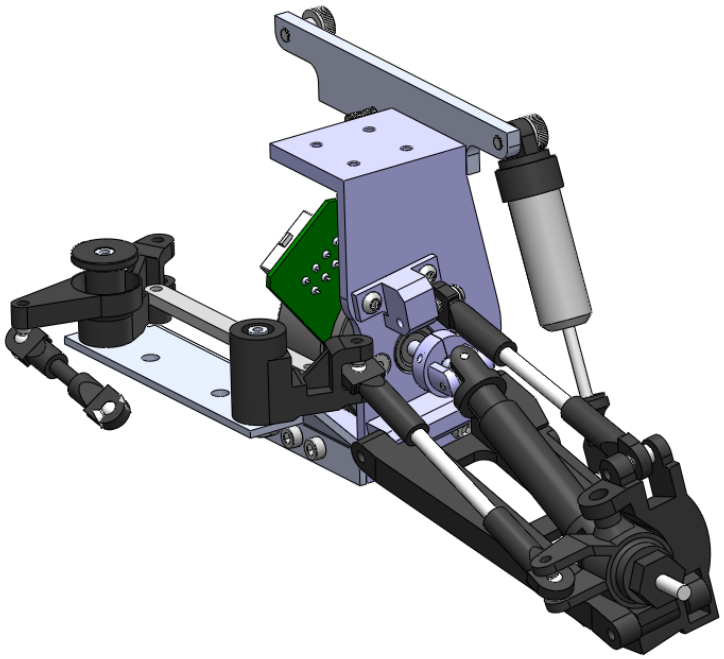


Figure 15: Existing Traxxas steering and suspension system integration with designed components.

The mechanical layout is simple, and will maintain the vehicle like properties of the Traxxas Slash. The front and rear motor blocks consist of the original Traxxas suspension linkages, where the front is integrated with the original steering system. Widening the track width requires that a new steering linkage be designed that connects the left and right steering bell cranks. Furthermore, using the Maxon motors requires a coupler that is compatible with the existing Traxxas constant velocity shaft.

5.1.2 Dynamic and Similitude Properties

Developing a realistic model in Solidworks is key to estimating the mass and geometric properties of our design. The estimated parameters, along with their sources, can be seen in Table 7. A significant portion of the design phase was spent creating solid models of the existing Traxxas Slash parts so that we could accurately incorporate them into our final design. This allowed us to design our new parts so that the steering and suspension geometry would not change. However, incorporating the Maxon motors into the design meant that we would need to increase the track width.

Table 7. Estimated final design vehicle parameters

Symbol	Description	Value	Units	Source
m	Vehicle mass	3.8	kg	Measured
a	Distance from CG to rear axle	204	mm	Measured
b	Distance from CG to front axle	197	mm	Measured
H	Height of CG	75	mm	Measured
I_{zz}	Moment of inertia about vertical axis	0.136	kg-m^2	Solidworks Model
c_α	Tire cornering stiffness	63.2	N/rad	T. Stevens
L	Wheelbase	402	mm	Measured
T	Track	275	mm	Measured
D_w	Wheel diameter	112	mm	Provided
AR	Aspect ratio (L/T)	1.45	-	Measured

Our scale model is sized to be similar to a standard SUV, such as the Ford Edge. A side by side comparison of the two can be seen below:

Parameter	$1/7^{th}$ Scale Ford Edge [27]	Our Vehicle
Wheelbase (mm)	406	402
Track (mm)	236	275
Aspect Ratio	1.72	1.45
Weight (kg)	5.03	3.8
Wheel Diameter (mm)	90	112
Turning Radius (mm)	1600	1300

We can see from this comparison that widening the track width has made the vehicle more of a $1/7^{th}$ scale vehicle, but dimensionally it is very similar. The tires appear to be oversized, but this is a factor out of our control. Another interesting parameter is the weight. As mentioned earlier in the report, scaling $1/10^{th}$ by dimension scales the weight by $1/1000^{th}$. As a $1/7^{th}$ scale dimensionally, the weight should scale by $1/343^{rd}$. This makes a $1/7^{th}$ scale model of the Ford Edge slightly heavier than our model. However, our model is not complete, or completely accurate and we anticipate the final weight to rise to 4.5 to 5 kg.

5.2 CONTROLS ARCHITECTURE

The mechatronics systems were designed to maximize functionality while retaining a high level of usability and accessibility. Upon successful firmware implementation, the user will be able to access the full functionality of the sensors and motor drivers from Simulink. The first implementation will have hardware that allows for the addition of new sensors and outputs. While the user will have to modify the firmware depending on the functionality they hope to add, we will provide an example of what a user would have to do to add a separate module. Figure 16 shows the designed modularity, with native Raspberry Pi code (Simulink, C++, Python, or otherwise) talking through the motherboard in a standardized way. The motherboard holds the Teensy, which will have firmware specifically adapted to talk to the hardware

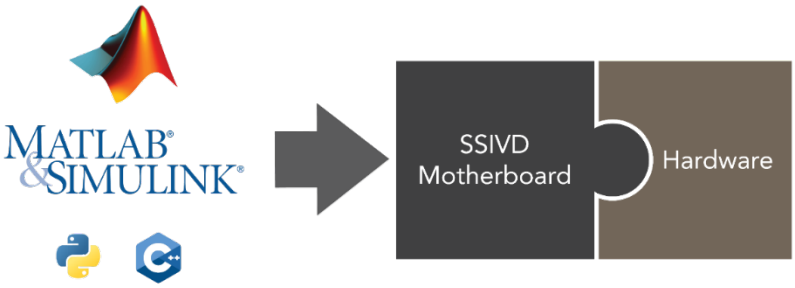


Figure 16: Hardware abstraction design of the SSIVD platform.

5.2.1 Steering and Drive Motors

The car is moved by the drive motors and the steering servo. For steering, we are still using the servo from the Traxxas Slash for compatibility with the steering hardware that is part of the original suspension and drive system. On the Slash, this servo is driven at 6V and the angular position controller takes a pulse-width modulation (PWM) input. The drive motors are Maxon EC 45 Flat 70W brushless motors with a 1024 count per revolution encoder. Additionally, they have a 3 count hall sensor that is used to assist the motor driver in properly timing signal at lower speeds. We selected the 24V configuration of this motor, and will be driving them using the Maxon EPOS4 Compact 50/5 positioning controller, pictured in Figure 17.

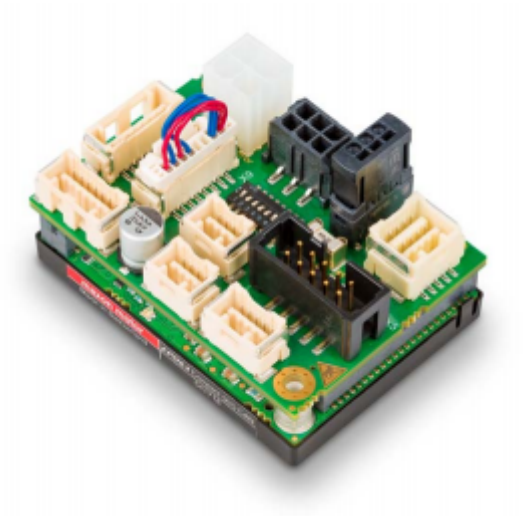


Figure 17: Maxon EPOS4 Compact 50/5 Can Positioning controller with manufacturer supplied connector board.

Each controller can drive up to 50V and 5A, and have a voltage drop of approximately 10% the supplied voltage. A 7s LiPo battery will supply the controllers with 25.9V, which will output just under the desired 24V. The drivers receive commands and transmit position and speed data on a Communication Area

Network (CAN) line compliant with the CiA 301 V4.2 Communication Protocol for Industrial Systems. The drivers run the CAN transceiver at 5V [28].

5.2.2 Radio Control

Using the Traxxas radio receiver, we can take the throttle and steering input from the handheld controller. It comes in as a 5V 100Hz PWM signal, which can be read, processed by the Raspberry Pi, and re-output with modifications defined by the control algorithm. Additionally, we are using a 315Mhz single button radio toggle as a remote kill switch.

5.2.3 Sensors

An intelligent vehicle requires awareness of itself and awareness of its surroundings. For awareness about itself, we are using a BNO055 IMU to provide orientation and acceleration data at 100Hz. The BNO055 module has a built-in processing unit that can output three-dimensional vectors for absolute orientation, angular velocity, acceleration, magnetic field strength, gravity, and temperature. It can be operated from 3.3-5V and communicates via the I2C protocol. To give a powerful but accessible awareness of the surroundings, we are providing a URF for distance measurements and a camera intended for computer vision applications. The Maxbotix HRLV-EZ0 URF operates at 2.7-5V, detecting objects from 1mm to 5m and returning a range for objects from 30cm to 5m with up to 1mm accuracy. The specific detection characteristics for varying shaped objects is shown below in Figure 18.

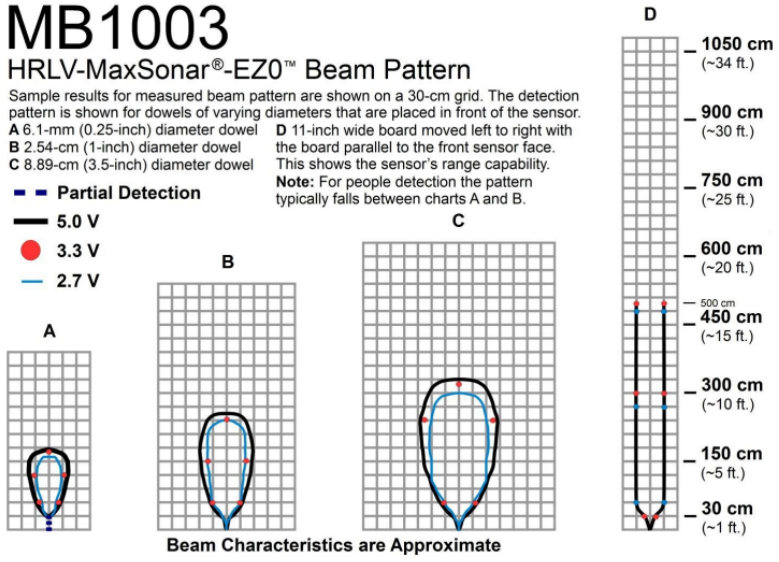


Figure 18: Maxbotix URF Detection characteristics for various object types.

The included Raspberry Pi Camera Board v2 is an 8 megapixel still and video camera than connects directly to the Raspberry Pi. It can be used with the Simulink computer vision toolbox for intelligent vehicle control algorithms or it can record and be used to review the performance of algorithms.

Since the sensor position configuration will depend on the user's application, we have not created fixed mounting locations. For our application, we 3D printed brackets for the IMU at the center of gravity and for the other sensors at the front of the car.

5.2.4 Motherboard

The motherboard will have an attached wire that connects directly to the LiPo battery. A commercial module will isolate and regulate 5V to power the Raspberry Pi, which will then output 3.3V. Our custom motherboard is designed to plug directly in to the top of the on-board Simulink programmable Raspberry Pi. We use a secondary microcontroller to interface with the peripheral sensors and the motors. On the motherboard is an attachment for any of the pin compatible Teensy 3.2, 3.5 or 3.6 microcontrollers. Any of those three microcontrollers work with the default sensors and configuration we have created. For maximum processing power and pin capability, we will be using the Teensy 3.6 microcontroller, the pinout for which is shown below in Figure 19.

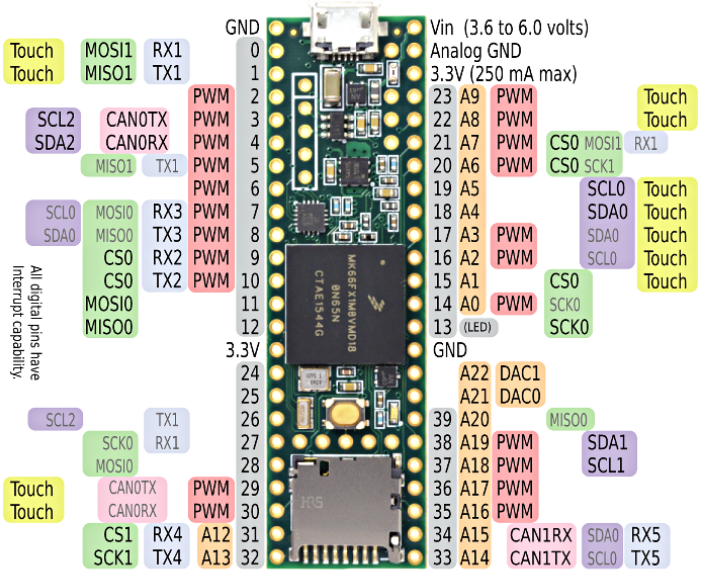


Figure 19: Teensy 3.6 microcontroller pinout and pin capabilities. The Teensy 3.5 is physically similar, and the Teensy 3.2 is truncated to pins 0 through 23.

The motherboard breaks out connectors for each of the components. Screw Terminals are used for the Motor Driver power and CAN lines. Male pin header is used for the controller receiver and servo output. Female header is used for the IMU, URF, and remote cutoff receiver. The microcontroller runs off of 3.3V and operates the IMU and the URF at that voltage level. The servo, remote cutoff receiver, and Traxxas receiver are powered by 5V, and an on board logic shifter is used to translate the 5V logic to 3.3V and vice versa. The CAN line on the motherboard is 3.3V, however the 5V CAN on the motor drivers is fully compatible with 3.3V CAN [29]. Figure 20 shows the 3.3V CAN Transceiver from the Teensy sending a message to the bus while the 5V CAN Transceiver on the Motor Controllers acknowledges the message with a spike in voltage levels.

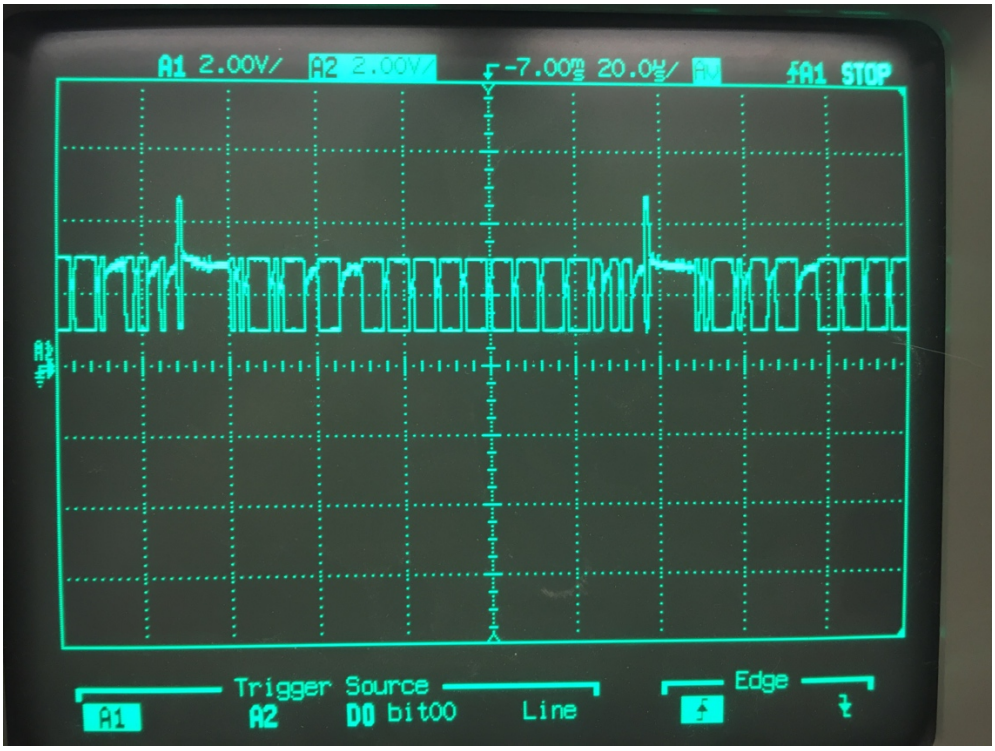


Figure 20: Oscilliscope Capture of the 3.3V and 5V Compatible CAN Bus

5.2.5 Communication Scheme

The Raspberry Pi receives data from and instructs the Teensy microcontroller via a serial connection. We had originally tried to use SPI as a connection protocol but due to trouble implementing this we switched over to strict serial communication. A CAN line is produced by a CAN transceiver chip connected to the Teensy's onboard CAN port which enables communication to the Maxon 50/5 Motor Controllers. PWM modules are used to communicate from the radio receiver to the Teensy and from the Teensy to the servo. I2C is used to communicate with the IMU and Laser Distance Sensor. Communication with the URF Distance Sensor uses analog input. Further discussion on the implementation of Pi / Teensy communication and sensor interaction can be found in the Firmware Architecture section below.

5.3 FIRMWARE ARCHITECTURE

5.3.1 Raspberry Pi

The Raspberry Pi will be programmed through Simulink and its control modules. The main functionality of the Raspberry Pi will be a loop format, continuously getting sensor and receiver data and outputting the calculated values to the Teensy microcontroller. Due to the limitations of the Raspberry Pi, it will need to be setup as the master in the SPI protocol when communicating with the Teensy. This isn't much of a problem though as the communication procedures will be planned to give enough clock cycles for the Teensy to respond back with all the sensor and receiver data. The Simulink code provided will then act as

the control algorithm and repeat the SPI communication. This loop will repeat for as long as the vehicle is turned on.

5.3.2 Teensy

The Teensy will be pre-programmed in C and will act as a fixed firmware. The Teensy, like the Raspberry Pi, will be also be programmed in a loop format. We had a goal of using hardware capture timers but instead decided to use interrupts due to time constraints.

With the steering servo that uses a PWM signal, the Teensy will be using a hardware timer that turns the pin on or off. The timer registers will be updated according, and will be discussed further shortly. The motor controller uses the CAN protocol which will be sent out as messages as soon as we receive the info. This leads us to talk about the SPI protocol with the Raspberry Pi. An interrupt will be programmed to read the input from the SPI signal and update the hardware timer for the servo and send a CAN message every time the SPI transfer is initiated.

The remaining modules are the URF sensor, receiver, and the IMU sensor. The IMU will be the only current module that will need to be polled for data in the main loop. The I2C protocol will need to query the sensor for its info periodically, which will be implemented in the main loop. The URF sensor will be read from a hardware analog module that updates a register with the new information. The register data will then be copied to a known variable in the main loop. The last module, the receiver, will be read using the hardware input capture which uses hardware to read the incoming PWM signal. The data received will then be copied to a known variable in our loop.

In all, we will have one interrupt from the raspberry pi driving the motor controller and steering servo's data. We will be capturing data using two different hardware configurations and one polling mechanism.

5.3.3 Fault States

Within both the Raspberry Pi and Teensy's firmware we will be fixing a series of fault states. These states will be more fully defined as a part of the firmware programming, but will prevent motion and illuminate the indicator led in a manner specific to the fault state. We anticipate designing fault states for user errors, communication failures, loss of radio control, and remote cutoff being triggered. As we encounter faults in the electrical prototyping phase we will continue to define and document fault states.

5.4 SAFETY HAZARDS

Using the Design Hazard Checklist from student success guide, we have proactively identified the following safety hazards in our device and have created mitigation plans for each.

- Design contains revolving action at wheels and potential pinch / shear points in structure
- Design can undergo high acceleration due to powerful motors
- Design will be able to go very fast presenting impact hazard
- There will be a large Lithium Polymer battery in the system, running at 25.9V
- It will be possible to use the system in an unsafe manner by driving into people or walls

The full design hazard checklist including mitigation plans is presented in Appendix E.

5.5 MAINTENANCE & REPAIR

The driving factor behind our decision to use Traxxas parts is that they are easily sourced and can be replaced if need be. Therefore, any Traxxas component that fails can simply be purchased from their website. All other designed components are intended to last for the entirety of the vehicle's life. However, some parts may need to be replaced simply due to wear and tear. This may include any electrical connections that become frayed or disconnected, bolts that strip from being over-torqued, or some of the 3-D printed parts. In each case, we anticipate that repair will be simple and not time-consuming or expensive. Overtime bearings used in the steering bell cranks may need to be lubricated, as well as the motor shafts.

6. DETAILED DESIGN & SUPPORTING ANALYSIS

6.1 MATERIAL SELECTION

We have been working with a materials engineering consultant group to help determine the best material to use for our purposes. The first design we discussed with them was the chassis. Our primary objectives with the chassis was that it would be lightweight, durable, and stiff. They generated a set of Ashby charts that helped to compare different material classes against each other. These charts can be seen in Appendix E – Materials Selection. The results showed that compared to other plastics, polyamides, such as Nylon, would maximize fracture toughness and fatigue strength, while still remaining stiff and light. Nylon is also easily machined and will be easy to tap for mounting purposes. Regarding other parts, we have sent the materials consultants all relevant information pertaining to the rest of our design. We are waiting for verification that aluminum is a strong choice for use in the motor housing and suspension mounting components.

6.2 BASICS OF VEHICLE DYNAMICS

Developing control algorithms for a small scale vehicle implies the inherent need to understand how the vehicle will move. A simple way of modeling a vehicle is through a single-track, bicycle model. This method combines the two tracks into one, eliminating a great degree of complexity while still maintaining an accurate representation of the vehicle dynamics. The basics of vehicle dynamics are outlined effectively in Rajesh Rajamani's book "Vehicle Dynamics and Control". It should be noted that the majority of the governing equations utilized in this section are referenced from Rajamani's derivations.

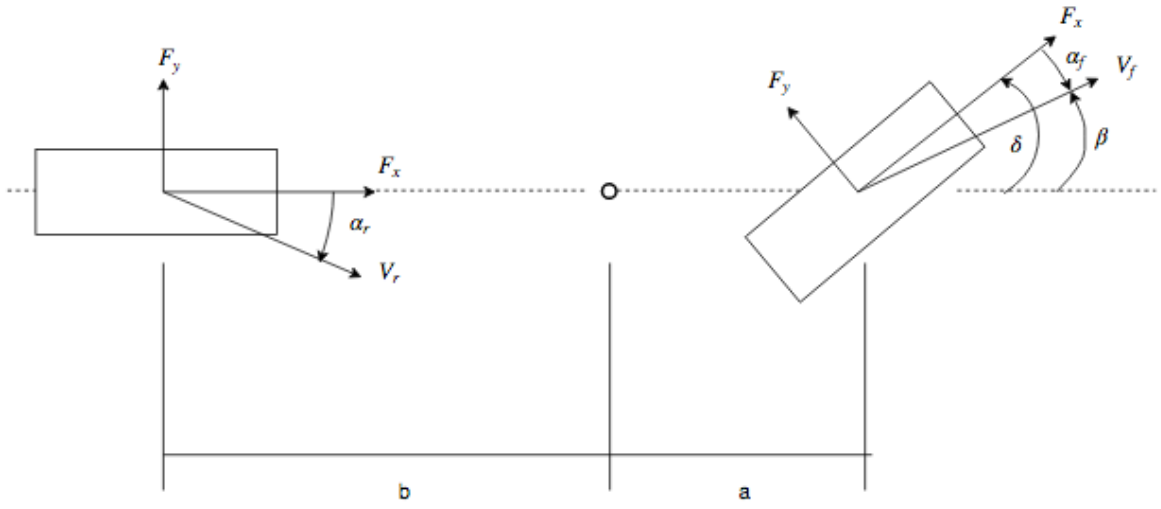


Figure 21: Schematic of Bicycle Model

6.2.1 Equations of Motion

The governing equations for the bicycle model can be derived using Newton's Second Law, resulting in a system of coupled, first order differential equations. In order to fully describe the motion of the vehicle, it is necessary to know the details of the forces that are acting at each tire-road interface. This is a widely studied area in vehicle dynamics, so a wide variety of literature is available on the subject. Note that the equations below omit the influence of rolling resistance and drag. These forces can be accounted for by simply including them in the longitudinal component.

$$\ddot{x} = \frac{F_{xf}}{m} cos\delta + \frac{F_{xr}}{m} - \frac{F_{yf}}{m} sin\delta + \dot{\varphi}\dot{y}$$

$$\ddot{y} = \frac{F_{yf}}{m} cos\delta + \frac{F_{yr}}{m} + \frac{F_{xf}}{m} sin\delta - \dot{\varphi}\dot{x}$$

$$\ddot{\varphi} = \frac{a * F_{yf}}{I_z} cos\delta + \frac{a * F_{xf}}{I_z} sin\delta - \frac{b}{I_z} F_{yr}$$

6.2.2 Lateral Forces

Understanding tire mechanics is fundamental to predicting how a vehicle will react to a steering angle change or throttle input. Most tire models rely on empirical data gathered in a controlled test set up. One such model is Pajecka's magic formula, where the lateral tire force is calculated as a function of the slip angle:

$$F_{yf} = Dsin\{Ctan^{-1}[B\alpha_f - E(B\alpha_f - tan^{-1}(B\alpha_f)]\}.$$

Fortunately, Thomas Stevens, who worked on a similar project involving a Traxxas Slash, developed a tire test stand specifically for calculating the coefficients for the magic formula. These coefficients were slightly modified, and are listed below in Table 8.

Table 8. Pajecka's Magic formula coefficients for Traxxas RC car tires.

B	Pajecka stiffness factor	10.0	-
C	Pajecka shape factor	0.09	-
D	Pajecka peak factor	70.0	N
E	Pajecka curvature factor	0.65	-

Figure 22 shows the lateral force plotted as a function of slip angle using Pajecka's equation and the parameters collected by Stevens. Investigation of this plot shows that at small slip angles (about -0.1 to 0.1), the lateral force developed by the tire is linearly proportional to the slip angle. This is described by the cornering coefficient, c_α, which is approximately 63.2 N/rad. This value will potentially be very useful in the development of vehicle control algorithms that need to manage the motion of the vehicle.

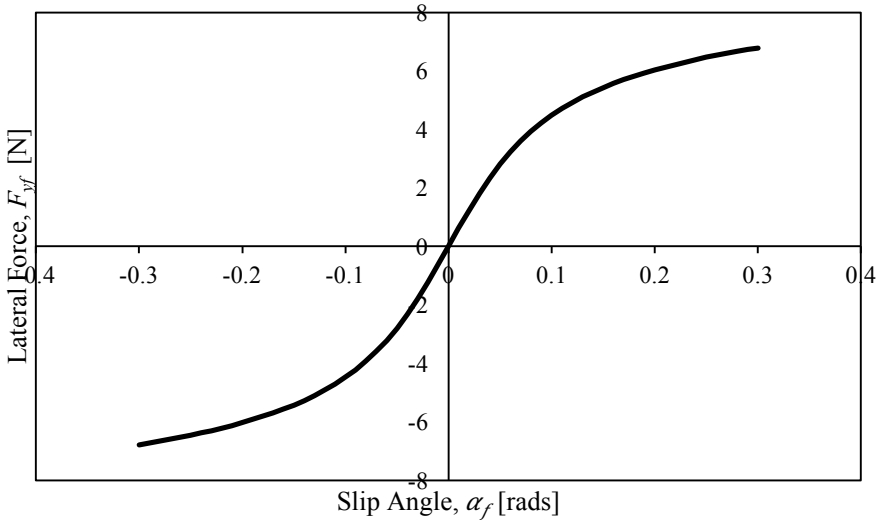


Figure 22: Lateral tire force as a function of slip angle for Traxxas slash tires. The curve is generated using Pajecka's magic formula and data referenced from Thomas Fitzgerald.

6.2.3 Longitudinal Forces

The longitudinal forces at each of the tires is what drives a vehicle forward. The drivetrain is responsible for transferring the mechanical energy generated by the motor to the wheels, which in turn is transmitted to the ground. Typically, vehicles will have some sort of gear reduction from the output driveshaft, that reduces the speed and increases the torque at each wheel. Our vehicle platform will not have a gear reduction from the motors, which means that the output speed of the motor shaft is equal to the speed of the wheel it is driving. Used in conjunction with data processed from the IMU, a slip ratio can be calculated with the following two equations:

$$\sigma_x = \frac{r_{tire}\omega_{wheel} - V_x}{r_{tire}\omega_{wheel}} \qquad \text{(Acceleration)}$$

$$\sigma_x = \frac{r_{tire}\omega_{wheel} - V_x}{V_x} \qquad \text{(Braking)}$$

Similar to the lateral tire mechanics, a force is generated at small slip angles that is linearly proportional to the slip ratio. If the longitudinal stiffness, c_σ, is known, the longitudinal tire force can be calculated.

$$F_x = c_\sigma \sigma_x$$

Typically, the only provided data is the speed of the output shaft of the motor, rather than the torque. This means that the generated tire force cannot be directly calculated and this relationship will need to be utilized. The slip ratio would be a readily calculated variable on our vehicle platform, as the IMU will be able to measure the absolute velocity of the vehicle and the tachometers in the motors can quantify the wheel speeds. However, the longitudinal stiffness still needs to be quantified in order for this relationship to be useful. Estimating this value for the Traxxas Slash would require extensive testing and falls outside the scope of our project.

6.2.4 The Understeer Gradient

At low speeds, a vehicle travels in the direction of the steering angle. At high speeds, a slip angle is generated that produces a force proportional to the cornering stiffness of the tire. An important parameter in vehicle design is the understeer gradient. This value helps to define the type of handling a vehicle will have. The understeer gradient is defined as:

$$K_V = \left(\frac{m_f}{2c_{\alpha f}} - \frac{m_r}{2c_{\alpha r}}\right)$$

where $m_f$and m_l are the mass transfers to the front and rear axle, respectively. Typical consumer vehicles have a positive understeer gradient, meaning that the driver has to turn the wheel more in order to travel the intended path. This makes complete sense from a safety standpoint, because it would minimize the risk of sharp, accidental turns that could potentially be dangerous. This is also something to take into account when modifying the center of gravity on the Traxxas Slash. Since all four tires have the same cornering coefficient, the only way to ensure the understeer gradient is negative is by positioning the weight further back from the center of the vehicle. This shifts the weight to the rear axle, resulting in an understeer governed vehicle. Our vehicle would be classified as having a positive understeer gradient, as the CG is positioned slightly forward of center.

6.2.5 Preliminary Vehicle Simulation

The equations of motion are very useful for developing vehicle models that can predict the motion of the vehicle. Even though the bicycle model greatly simplifies the overall system equations, further simplifications can be made develop a more basic simulation. Assuming a constant longitudinal velocity removes the longitudinal parameters, resulting in two first order differential equations. Substituting the lateral tire forces for functions of the slip angles and rearranging the equations to solve for the highest order derivative (See Appendix E for details) allows us to develop a state-space model [30]:

$$\frac{d}{dt}\begin{Bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2(C_{\alpha f}+C_{\alpha r})}{mV_x} & 0 & -V_x - \frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{I_z V_x} & 0 & -\frac{2(C_{\alpha f}l_f^2 + C_{\alpha r}l_r^2)}{I_z V_x} \end{bmatrix} \begin{Bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{Bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2l_f C_{\alpha f}}{I_z} \end{bmatrix} \delta$$

The output from the state-space model gives positions and velocities in the vehicles coordinate system. These outputs are similar to values we would see coming directly from the IMU, as it measures the vehicle's acceleration and can be integrated to get the position and velocity. Figure 23 shows the vehicle response to a given steering input using the state space model. For more details and other results from the simulation, see Appendix E. Results from this simulation can be compared to tests with our completed vehicle to tune the vehicle parameters.

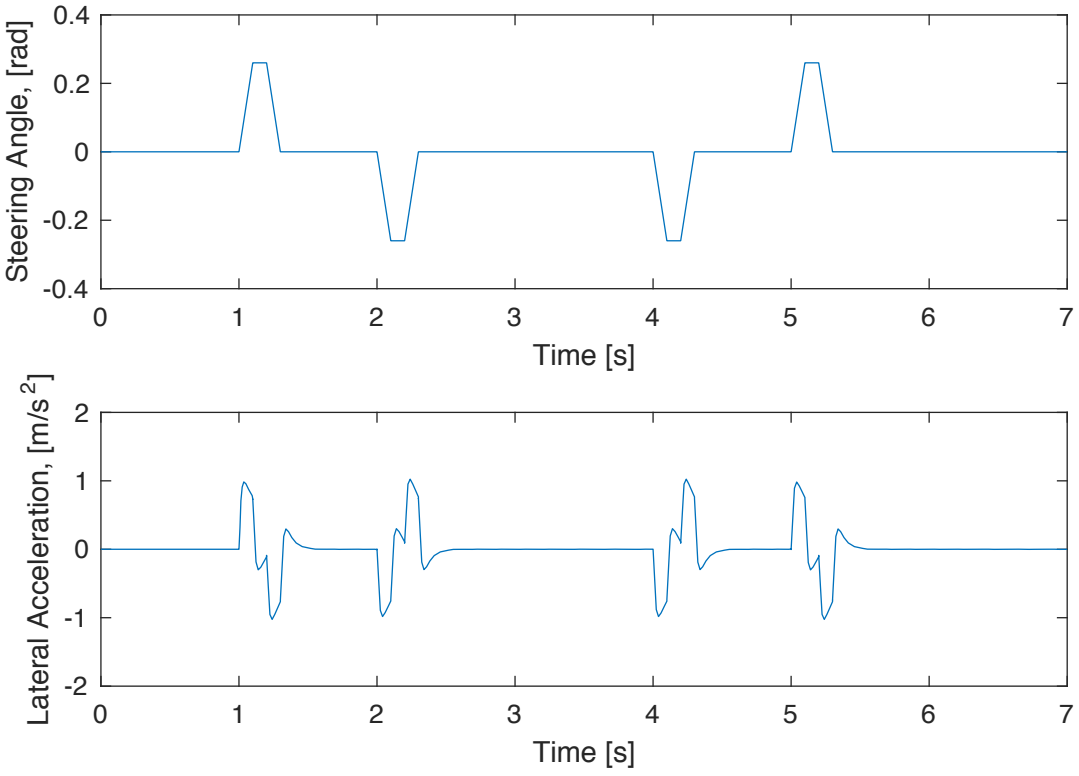


Figure 23: Steering response for vehicle simulation. Top plot shows the input steering angle and the bottom plot shows the lateral acceleration.

If path planning is important, a coordinate transformation can be used to change to the global coordinate system:

$$X = xcos(-\varphi) + ysin(-\varphi)$$

$$Y = -xsin(-\varphi) + ycos(-\varphi)$$

The coordinate transformation is useful for visualizing the anticipated path based upon the applied steering angle. For the plotted results from a sample vehicle simulation, refer to Appendix E.

6.2.6 Turning Radius

Another important vehicle parameter to estimate based upon our design is the turning radius. Increasing the wheelbase and track naturally increases the turning radius, which is determined by:

$$R = \frac{T}{2} + \frac{L}{\sin(\delta)}$$

Our final design limited the maximum steering angle due to the inclusion of the motors. We estimated that the maximum angle decreased from 25 degrees to 20 degrees. Furthermore, our overall vehicle dimensions increased. As a result, the turning radius increased from about 900 mm to 1300 mm. This seems like a substantial change, but our vehicle is also a 1/7th scale dimensionally rather than 1/10th. The Ford Edge has a curb to curb turning radius of 38.6 ft [31] – or in 1/7th scale 1675 mm. This means that our model actually has better turning capabilities than necessary, even despite the steering angle reduction.

6.3 CHASSIS DESIGN

6.3.1 Similitude

The first design decision with the chassis was deciding upon a wheelbase that would be realistic compared to our new track width. Overall, our track width increased 33.6mm compared to the original Traxxas Slash. This required a brief similitude analysis. A standard vehicle SUV has a wheelbase to track width ratio of between 1.7 and 2.0. Increasing the track width meant we would need to lengthen the wheelbase to maintain similitude. Our updated track width is approximately 250 mm, which corresponds to a wheelbase of at least 425 mm. Furthermore, typical nylon sheets come in lengths of 12 in (~305mm), which would result in a wheelbase of 420mm with the chassis mounts. For simplicity, a 300 mm long chassis would suffice. This would result in an aspect ratio of 1.7 for a 1/7th scaled vehicle to an SUV, such as the Ford Edge.

6.3.2 Design for Stiffness

The overall stiffness of a vehicle is dependent upon the tires, suspension and suspension members, and the chassis. The interplay between these elements can be simplified into a stick model as shown in Figure 24 [32]. The central chassis is modeled as a torsional spring and the suspension members are simple longitudinal springs. Treating the springs as acting in series, meaning that the total deflection of the chassis is the sum of the deflection of each element, the following equation is used to determine the total torsional stiffness of the vehicle:

$$\frac{1}{K_{Total}} = \frac{1}{K_{Chassis}} + \frac{Track^2}{K_{Suspension}} + \frac{Track^2}{K_{Tire}}$$

Note that this equation converts the longitudinal stiffness of the springs to a torsional stiffness in order to maintain consistency between the spring constants. Furthermore, including the tires as another term will account for their contribution to the overall stiffness of the vehicle. This can also be done for the suspension members, however, it could be difficult to approximate values for these parameters, so they are assumed to be rigid.

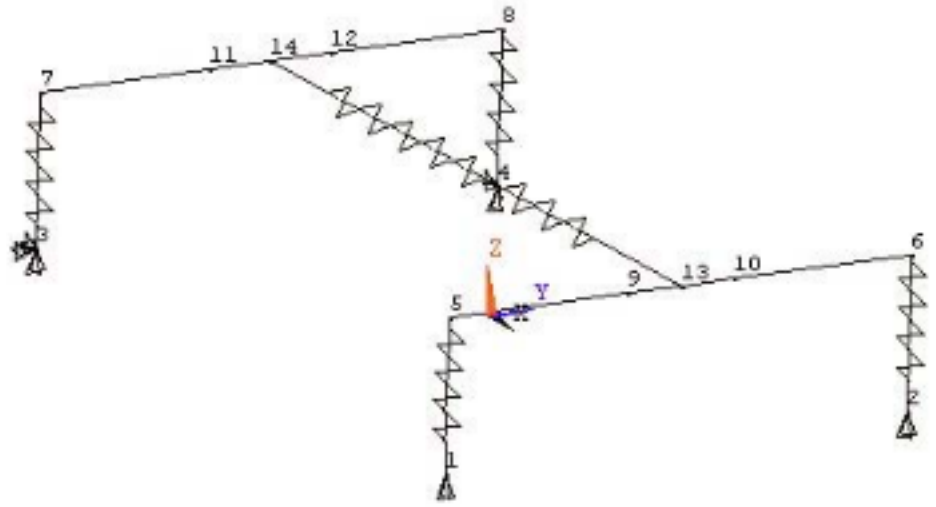


Figure 24: Vehicle stick model used to predict overall vehicle compliance

Ideally, the chassis is designed to be as stiff as possible in order to allow the suspension to be the primary contributor to shock absorption. However, weight considerations typically mean that compromises are made that increase the overall compliance of the vehicle. Weight is not a primary concern for the purpose of our project. In fact, a heavier vehicle would represent an actual vehicle more accurately. The only concern regarding weight is that if the chassis is designed too thick and additional components add a sufficient amount of weight, the suspension would bottom out too easily. Once again, keeping in mind the end goal for this project, the vehicle will not likely be used in scenarios where the full travel from the suspension is necessary.

An important simplifying assumption for the chassis design is that it is a flat plate in the shape of a rectangle. A complex geometry would mean that a computer software running a finite element code would need to be utilized. In reality, chassis design is not a rectangular shape, but for the purposes of this design, we can assume it is a rectangle. Torsional rigidity is a shape's ability to resist torsion. It is similar to the moment of inertia, in fact, it even has the same units. Unfortunately, there is not a closed form solution to solve for the torsional rigidity of a rectangle. The exact solution for torsional rigidity of a rectangle is given as [33]:

$$K_{Rect} = \frac{a^3 b}{3}\left[1 - \frac{192}{\pi^5}\frac{a}{b}\sum_{n=1}^{\infty}\frac{1}{(2n-1)^5}\tanh\left(\frac{\pi(2n-1)b}{2a}\right)\right]$$

where we assume a is the average width of the chassis and b is the thickness, which we would like to solve for. This equation can be used to predict the effective torsional stiffness of the chassis through the relationship:

$$K_{Torsion} = \frac{K_{rect}G}{L}$$

where G is the shear modulus of the material used and L is the total length of the chassis.

The measured spring constant for the suspension is 0.821 N/mm. This was calculated by measuring the unsprang length and then placing a mass (1.675 kg) on the spring and measuring the deflection. This methodology assumes the relationship between deflection and force is linear. Calculations for this section of the report were completed with a 305 mm track width and 250 mm chassis width, which is not the same as the final design parameters, but provides an accurate estimate for design purposes.

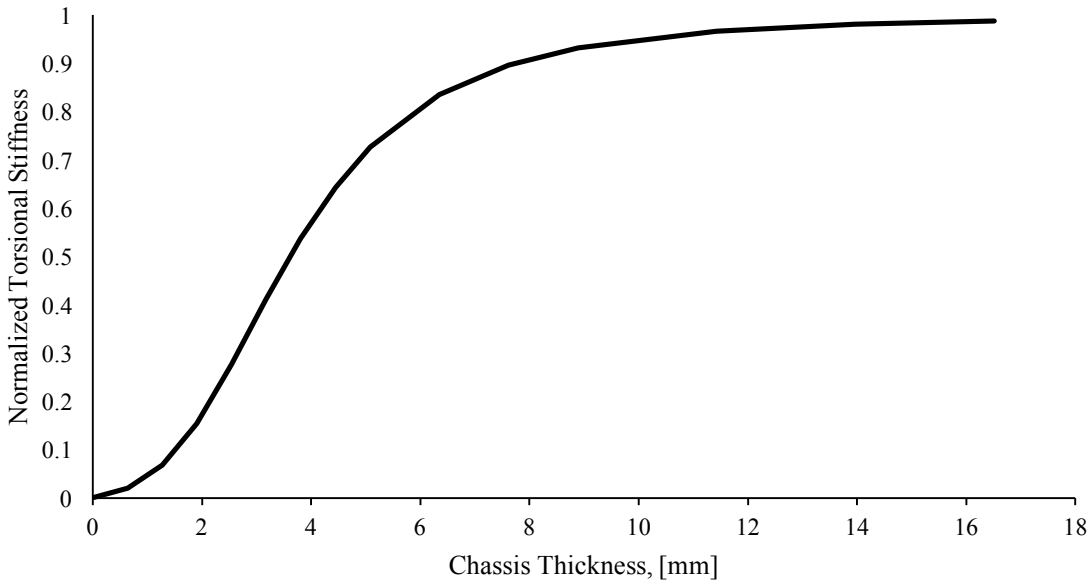


Figure 25: Normalized torsional stiffness of the entire vehicle as a function of chassis thickness.

For our purposes, the spring constants are a set value that need to be designed around. This means that any additional compliance added to the system by the chassis should not significantly affect the overall compliance of the vehicle. A simple way to ensure this was to plot the total torsional stiffness of the vehicle as a function of the chassis stiffness—which was determined by the thickness. This can be seen in Figure 25. It should be noted that the stiffness was normalized to where "1" represents a completely rigid chassis. A completely rigid chassis would allow for the suspension, tires and suspension members to dictate how the vehicle responds to ground disturbances, but may not be feasible from a design standpoint. A normalized stiffness value of about 0.85 is a more realistic value, as it results in a chassis that is about 8 mm thick.

6.3.3 Chassis Mount Design

Mounting the chassis directly to the top of the motors would have put the chassis height at approximately 125 mm. This would have been a very high position to mount the chassis and would have lifted the steering servo above the driveline plane. As a result, we chose to step the chassis down using aluminum sheet metal. However, we needed to ensure that the gage of sheet metal would be thick enough as to not compromise

the structural integrity of the chassis itself. From the previous section, we concluded that 8mm thick nylon would only reduce the overall stiffness of the vehicle by 15%. Aluminum is much stiffer than nylon, however we need to use a reasonable gage of sheet metal so that it can still be bent with a sheet metal bending brake.

Comparing the flexural rigidity of the aluminum sheet metal cross section with the nylon chassis cross section would give us a good idea of how the two components would bend relative to each other. If the two rigidities are similar, we would expect the two materials to behave similar to each other. That is, the overall compliance of the chassis would not be reduced by adding the sheet metal mounting brackets. The flexural rigidity of a rectangular cross section is defined as:

$$Flexural\ Rigidity\ = \frac{1}{12} bh^3 * E_f$$

where E_f is the flexural modulus (tensile modulus of isotropic materials). The flexural rigidity of the 8mm x 330 mm nylon cross section is about 56 N-m^2. The necessary thickness of an aluminum cross section 64mm wide to match this value is about 5 mm thick. We chose to use two sheets of 2mm thick sheet aluminum to clamp the chassis. Although this is would essentially be just 4mm the difference is still small enough as to not play a large role in decreasing the overall stiffness of the chassis. Furthermore, the geometry of the bent sheet metal and the clamping structure only furthers the bending stiffness, as more material is distributed further away from the neutral axis, effectively increasing the moment of inertia.

6.4 SHAFT COUPLER DESIGN

6.4.1 Concept Generation

We have two primary requirements for the shaft couplers – they must act as a u-joint compatible with the Traxxas drive-shafts and they should occupy as little depth along the motor shaft as possible. This allows us to minimize further widening of the track width. We initially looked at set screw couplers and split couplers, but found that those would not be feasible. Since there must be a way to get the u-joint on to the fixed ball of the Traxxas shaft, we had to opt for a split design as shown in Figure 26.

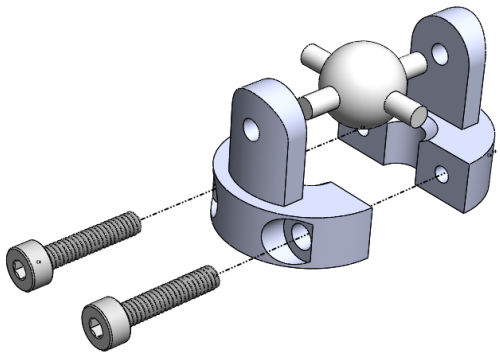


Figure 26: Exploded view of shaft coupler converging on the u-joint ball.

The clamping force generated by the tightening the screws will be sufficient for the transfer of torque between the shaft and coupler. This will nullify the need to modify the motor shafts, saving significant manufacturing time as well as eliminating the possibility of damaging any of the expensive motors.

6.4.2 Failure Analysis

The driveline yokes that the original Traxxas Slash came assembled with were made out of a heavy duty molded plastic. The shaft couplers that we are implementing will be made out of aluminum. Using aluminum will simplify both the analysis and manufacturing process for the shaft couplers. Aluminum is a stronger and stiffer material than the plastic used by Traxxas, which means we can assume the coupler design will not fail under normal, or even more extreme operating conditions. We anticipate that the most failure prone piece of the assembly will be the clamping screws. These will need to be able to support the torque in the shaft, while still be pre-stressed to effectively transfer torque between the motor shaft and yoke.

The chosen fastener to generate the clamping force on the coupler is an M2x0.5 10 mm screw. We chose this screw because it fits the compact profile of the coupler and it is the appropriate length to maximize the thread surface area that will be clamping the two coupler pieces together. However, calculations were needed to ensure that the resultant stress in each screw would not exceed the proof stress at maximum torque. The first assumption to make for the analysis was that the clamping screws would be tightened to 75% of their proof strength. This is a common loading condition for non-permanent fasteners [34]. This generates a clamping force of 1000 N in each screw, plenty sufficient for effective power transmission between the coupled shafts.

The clamped material also carries a portion of the load. The portion of the load carried by the fastener is:

$$C = \frac{K_b}{K_b + K_m}$$

where K_b and K_m are the bolt and member stiffnesses, respectively. The applied load, P (different from the preload), was based upon the maximum torque output from the Maxon motor. The resultant stress in the motor with this loading condition was found with the following equation, which sums the preload and applied external load from the motor:

$$\sigma_b = \frac{CP + F_i}{A_t}$$

The proof stress of the M2 screw is about 650 MPa. The loading condition above approximated the load to be 495 MPa, where the majority of the load comes from the clamping force. This fact means that a fatigue analysis was not necessary, as the compressive resultant on one side of the rotation would not make a significant difference in the loading conditions.

A supplementary analysis was conducted to determine the feasibility of creating shaft couplers out of a 3D printed plastic. Figure 27 shows the deformed finite element results using ABAQUS to determine the maximum normal stress that may occur in the coupler arm. The PolyJet printer uses FullCure 720 plastic with a yield strength of approximately 60.5 MPa. Even in the unmodified design, the highest normal stress is about 46 MPa, which is reduced to 38 MPa for the improved design. The improved design increases the width of the coupler arm and fillets the corners to reduce the severity of the stress concentration. It should also be noted that the couplers will be in a torsional stress state. The reported von Mises stress for the unmodified coupler is about 58 MPa, which is still below the yield stress of the material.

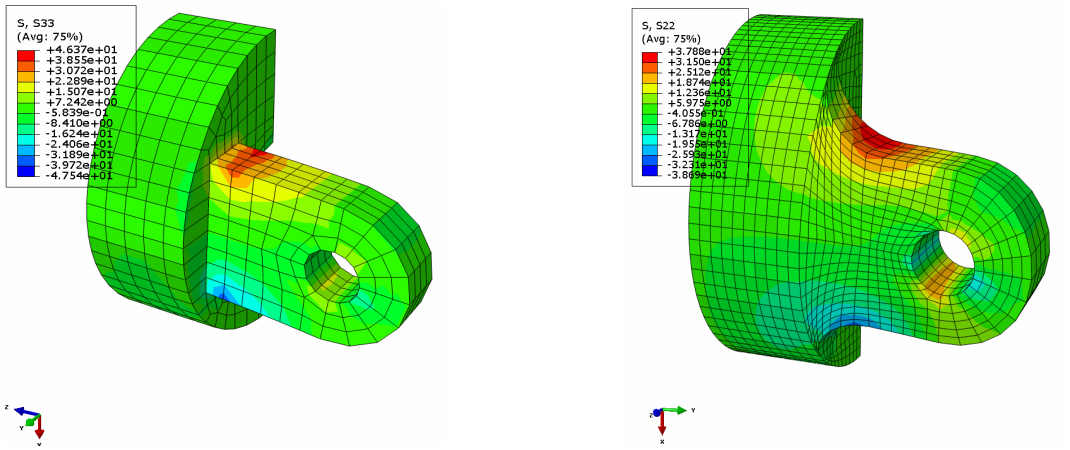


Figure 27. Shaft coupler FEA results for unmodified (left) and modified coupler designs (right).

6.5 DESIGN ANALYSIS OF LOAD BEARING MEMBERS

Modifying the mounting components for the suspension, motor, and steering members requires an analysis that takes into account high impact loads. Large cycles are not expected for members other than the shaft coupler, meaning we could forego a fatigue analysis and simply ensure that the new components could support non-cyclic forces. The basis for our design was built around the motor housing blocks, where brackets for mounting the suspension turnbuckles, a-arm, and ultra-shock are attached. Working in a very small space meant that the brackets would are small and there is a limited amount of material to prevent the mounting bolts from tearing out. As a result, our main form of analysis was hole tear out in the direction of minimum material thickness.

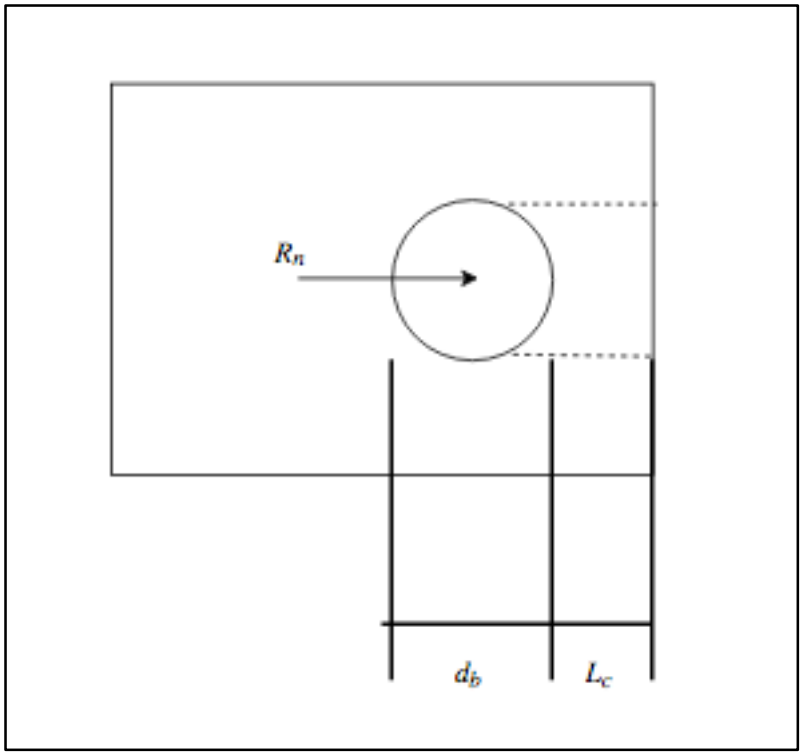


Figure 28: Hole tear-out path and critical dimensions.

A typical rule to follow when designing joints that are exposed to a shear force is $L_c = 1.5d_b$. If this rule is followed the primary mode of failure is the bolt shearing. As mentioned earlier, we were not able to design to this specification, meaning that we need to ensure the joints will not fail through shear of the aluminum mounts. R_n, the maximum resistive force before the joint failed, is applied in the direction of minimum material between the hole and parts edge. The equation of the limiting state for bolt bearing where the hole deformation is not a concern is provided by Bartlett Quimby [35]:

$$R_N = 1.5L_c t F_u$$

where t is the material thickness and F_u is the ultimate shear strength of the member. Results for the maximum force before hole tear out in each member can be seen Table 9. Also listed are the estimated loads and a resulting factor of safety. The loads were estimated based upon a large vertical deceleration and the vehicle mass. It is impossible to know exactly what magnitude of forces will actually occur in the joints, so seeing such large factors of safety is good. This will result in the robust and sturdy design we are looking for.

Table 9. Results from hole tear out calculations for custom components.

	Member	L_C [mm]	t [mm]	F_{Max} [N]	F_{approx} [N]	FS [-]
Rear	Turnbuckle Mount - A	2.3	2	1100	87	12.6
	Turnbuckle Mount - B	1.0	9.0	2150	174	12.3
	Ultra-shock Mount	3.4	8.0	6400	150	42.8
	A-Arm Mount	2.3	1.0	549	87	6.3
Front	Suspension Turnbuckle	2.1	2.0	980	87	11.3
	Ultra-shock Mount - A	2.5	4.0	2390	146	16.4
	Ultra-shock Mount - B	2.3	4.0	2200	146	15.1
	A-Arm Mount	2.3	1.0	550	87	6.3

After this analysis it becomes clear that majority of the members would not fail despite the relatively low hole spacing from the edges. We could increase the safety factors by simply increasing the thickness of the mounting surface or lengthening the distance of the edge from the hole. Nonetheless, the lowest F.S. is 6.3, which is still a very large safety factor even based off the ultimate strength.

6.6 MAXON MOTOR CURVES AND HEAT CONSIDERATIONS

Maxon provides data for the motor specifications on their website. This data needs to be compared to the vehicle system and parameters to ensure that the motors can produce the required amount of power for our purposes. Charlie Refvem generated a set of system curves for the Traxxas slash, as well as the Maxon performance curves based upon the provided data from the specifications sheet. These curves can be seen plotted in Figure 29. The vehicle system curves account for losses due to kinetic friction, drag, an incline angle of degrees, driveline inefficiencies and electrical inefficiencies. The assumptions made in this model are conservative, meaning we would expect to require less power than what is calculated. It should be noted that our desired maximum vehicle speed of 15 MPH corresponds to a motor speed of approximately 2000 RPM. The motors will allow us to continually operate at this speed without overheating the windings in the motor.

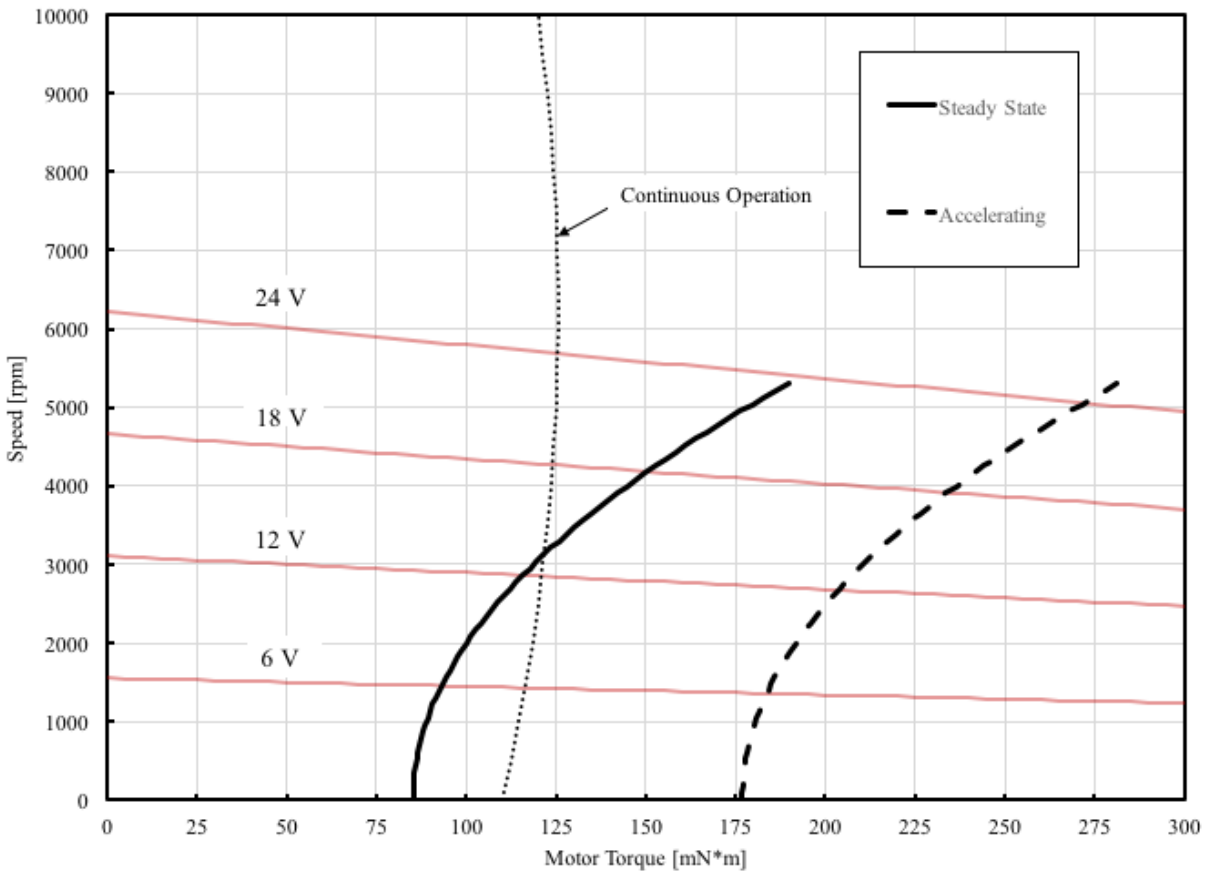


Figure 29: Maxon performance curves, steady state and acceleration system curves.

Our motor housing design mounts the motors back-to-back, with essentially eliminates one of the surfaces for heat transfer to the surrounding environment. The curves above to do not account for this for continuous operation. A more accurate model would push the continuous operation curve to the left, limiting the power output that the motors can operate at before overheating. This fact alone may warrant the need to include some sort of a heat sink between the motors to better disperse the heat. We noted earlier that the assumptions made to develop the system curves were conservative, which would also mean that a more accurate model would be pushed further to left. For this reason and the fact that a heat sink would widen the track width even more, we do not believe heat dissipation will be much cause for concern. The Maxon motors are being operated well below their limits, which means overheating will be unlikely.

6.7 UNDERSTANDING THE EXISTING SYSTEM FOR FUTURE DESIGN

Everything up until this point has been focused on what functionality the electrical system will have and specifying the general requirements needed to provide that functionality. This is the part of the project where we can get into detailed selection and design of the components, like wiring diagrams and types of digital communication, and actually specify every connection and component in the system.

Before we can get detailed, we need a scope and a big picture of the project to work with. While we were on break we took a lot of time thinking of the system as a whole and constructing a mental image of the

components we might need. After creating this mental picture, we researched the system of the Traxxas Slash vehicle we had chosen to work with. The Traxxas control system is as shown in Figure 30.

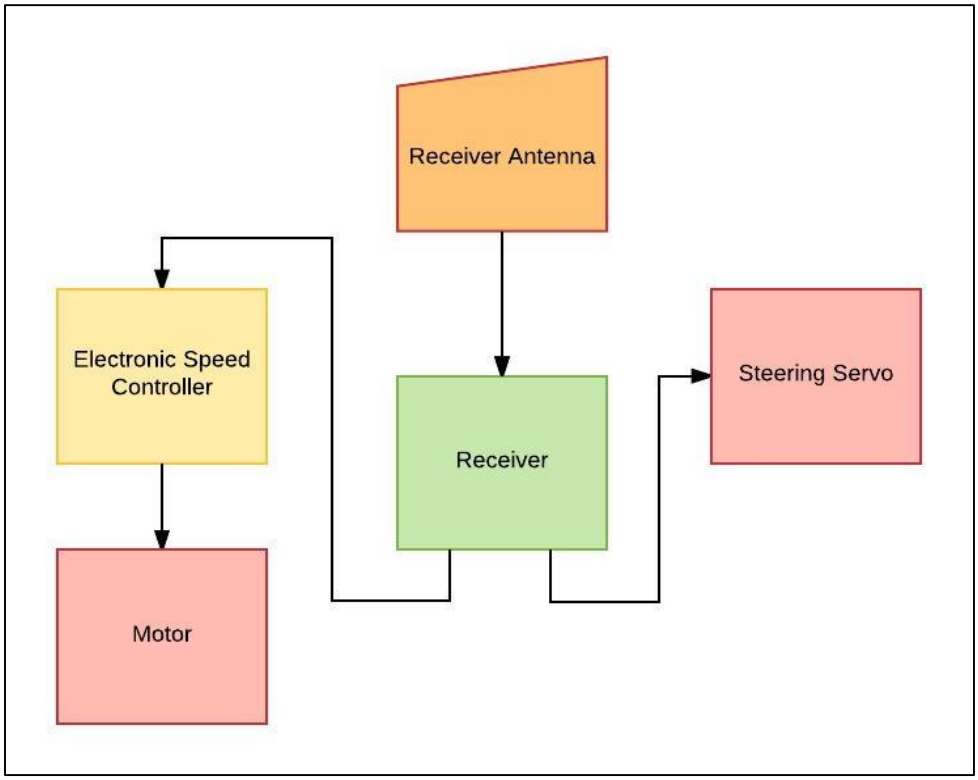
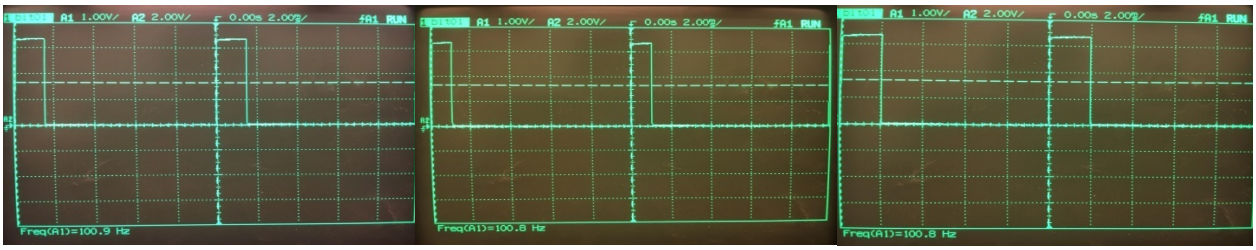


Figure 30: Existing Traxxas Slash Flow of Control

Luckily, not much time ended up going by until we could get our hands on the physical vehicle we ordered and we could spend time looking into the actual signals being sent. The first logical part to look at is the receiver as this is the component that starts all internal signals. We figured out which wires were which and hooked the output signal and ground signal up to an oscilloscope. As expected, we saw a Pulse Width Modulation (PWM) signal in the control wires that were associated with the motor/speed controller and the steering servo. The oscilloscope outputs from the signals are presented below in Figure 31.



(a) (b) (c)

Figure 31: Signal from receiver to ESC at neutral throttle (a), 15% duty cycle, reverse (b), 10% duty cycle, and forward (c), 20% duty cycle.

As indicated, these signals operate on a 10ms period ranging from a 10% duty cycle to a 20% duty cycle. These signals are also the same ones sent to the steering servo with the 10% duty cycle indicting a hard left turn and a 20% duty cycle indicating a hard right turn. After analyzing the signals from the Traxxas Slash, we can interpret this data for our own control regimen.

6.8 CREATING THE NEW SYSTEM

When reinventing a system on a machine, we've learned through previous endeavors it is best to use as much of the old system as possible. For our project, we've already listed the necessity of using new motors and their motor controllers to power the vehicle. Other items already required in our new system is the Raspberry Pi 3, a motherboard, a camera sensor, IMU, and ultrasonic range finder. So all in all we have new motors selected, a new "brain trust" of the Raspberry Pi and its motherboard, and new sensors. Other criteria crucial to the system but haven't had a need to be replaced yet include the steering servo, the receiver, and its antenna. Since we've seen no reason to replace these parts and we have verified they will work with the new system, we will go ahead and plan to include them in the new system. If you're reading this and wondering why he haven't chosen the motherboard yet, this is because we are waiting to plan out all the communication protocols and the specific pin arrangements we will need the motherboard to work with before we choose one that may or may not have the characteristics we are looking for. We have included Figure 26 to depict the power and data connections in our system.

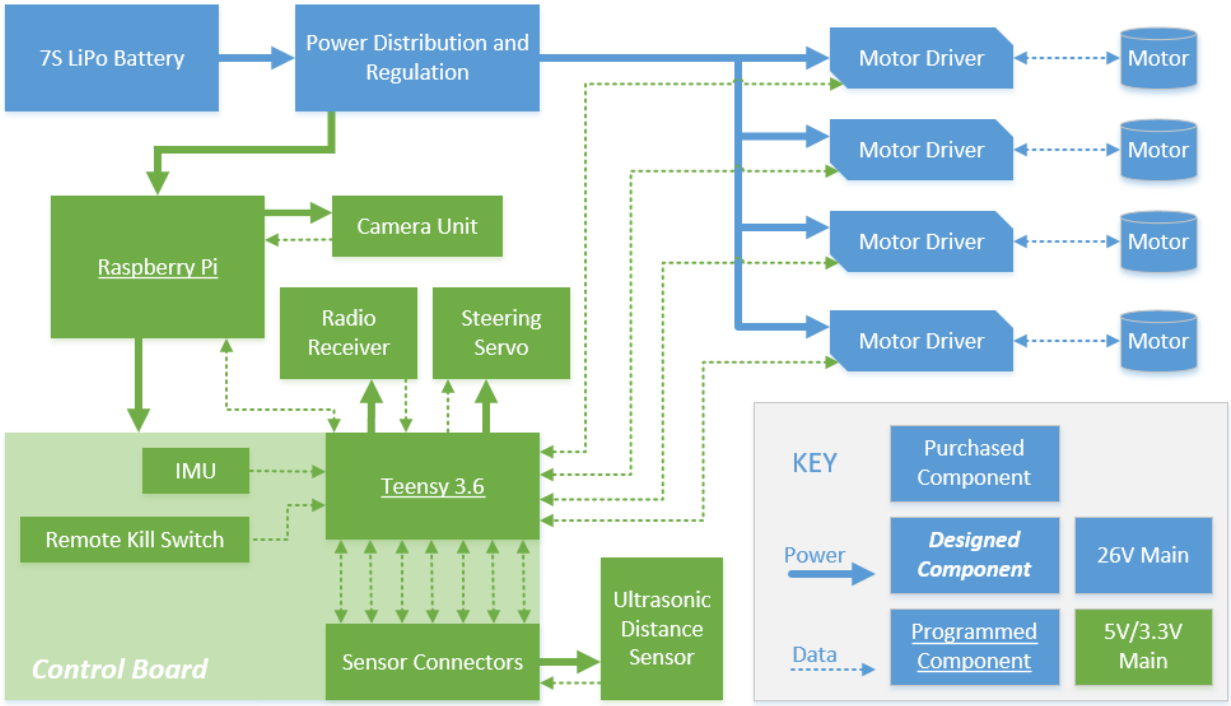


Figure 32: Completed Component Selection and electrical layout, with voltages and component information.

Now that we have chosen all the parts we will need we can proceed with planning the communication system between the electrical components. The existing servo will need to use PWM protocols as we discovered in the previous section. The receiver will obviously output a PWM signal for both the servo and motor that will need to be read by the motherboard. The IMU we have selected requires the use of the I^2C communication protocol. The ultrasonic range sensor we picked out uses Analog protocols to communicate with the motherboard. The camera will attach directly to the Raspberry Pi using a specialized Raspberry Pi connector. The last communication we need to scope out before we pick which motherboard we need is the motor controllers to the motherboard. We talked about using SPI, PWM, or Analog (in a worst case scenario) for the motor controller. However, after a meeting with Maxon who will be sponsoring the motors and motor controller, we were able to upgrade motor controllers to one that is able to use CAN protocols. CAN is widely used in the automotive industry when they construct their full-vehicle communication systems and we thought that using the CAN protocol in our system would improve our platform for both learning purposes and for making a more reliable and flexible communication system.

Our motherboard microcontroller will now need to be able to use CAN, SPI, Analog, I^2C, and PWM protocols. It will also need to accommodate our requirement of 15 pins available for data usage. We identified a few microcontrollers which seemed appropriate for our project including the Arduino Mega and the Teensy 3.2. We decided to go with the Teensy. We found that the Teensy had good documentation, ran on 3.3V (which matches the voltage level of the Raspberry Pi for a simpler circuit), is a very small part, and ran at a decently high clock rate to go along with the fact that it had enough pins for our system and can run all the communication protocols we listed above. We also decided to upgrade to the Teensy 3.6 device for the higher clock speed and having more pins for future sensor connections. The Teensy 3.6 and 3.2 are very similar in design and the Teensy 3.6 is almost an exact extension of the 3.2 version board.

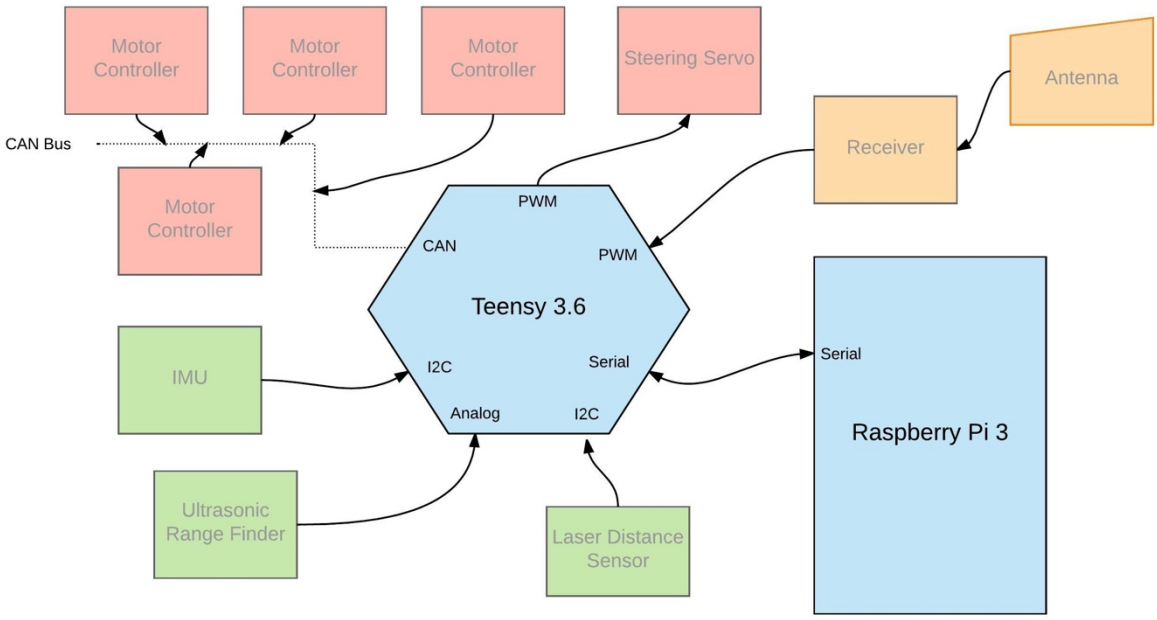


Figure 33: Design of the new system from a communication standpoint

6.9 SOFTWARE DESIGN

Software planning is the next logical step in the design process. We will have two main components to program in our project: the Teensy and the Raspberry Pi. The Teensy will mostly act as a signal handler for the Raspberry Pi but has some important tasks. The Raspberry Pi on the other hand will take the data available to it (through its connection with the PiCam and the sensor data that is passed along by the Teensy), calculate the appropriate actions to take, if any need to be taken, and output the result back to the Teensy for redistribution. The Teensy will then give the steering signal to the steering servo through PWM communication and the motor signal to the motor controllers through the CAN Bus protocols. Through looking at the specific needs of each input/output signal we developed a list of tasks the Teensy will need to do.

There are a couple options of how to implement the software in the Teensy. The first is a basic while loop running in main that continuously performs our tasks by calling functions and updating its data as is appropriate. The second idea is to implement a real time operating system in the Teensy and incorporate a scheduler that splits the different functionality needed into tasks which each gets their own amount of time to run. To implement a real-time operating system and scheduler is no easy task. It also didn't seem very appropriate for the Teensy's tasks.

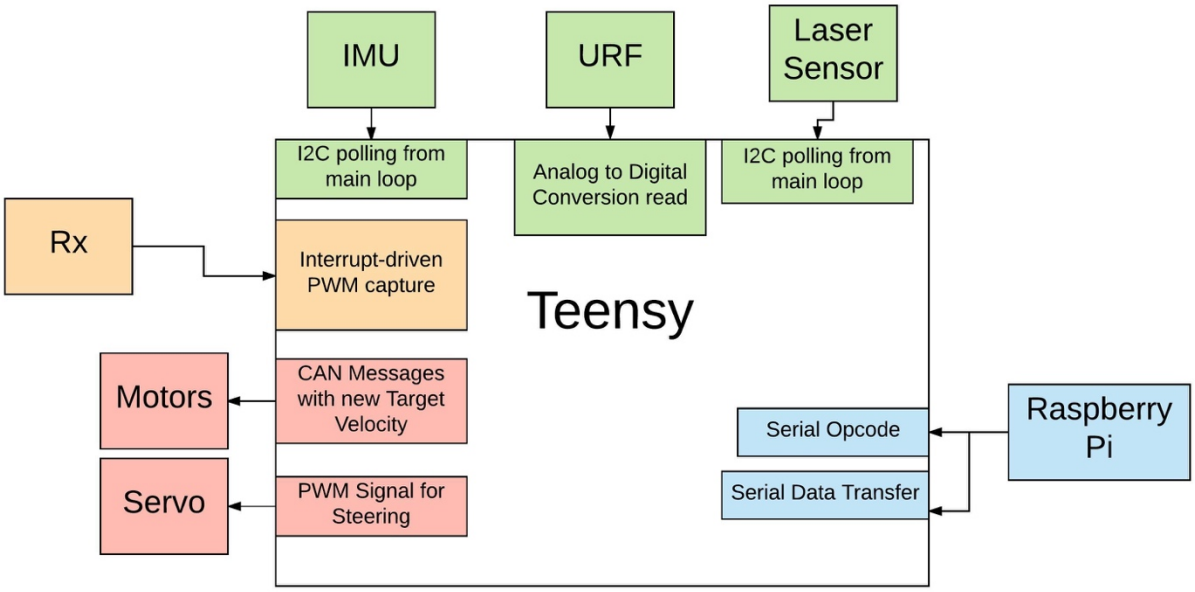


Figure 34: Teensy Task Decomposition

In Figure 28 we depicted a scenario of using the loop configuration for our software. All the tasks were possible and seemed fairly straight-forward in this manner. With a loop configuration in mind, the teensy will keep busy but won't be overloaded. We ultimately chose to incorporate the loop model because it helps simplify the chances for error while also creating a more straightforward approach. In the loop we communicate with the Raspberry Pi, motor controllers, IMU, and URF or laser sensor. It's a bit to do but we are sure that the Teensy can handle at least this much operating at 180 Mhz.

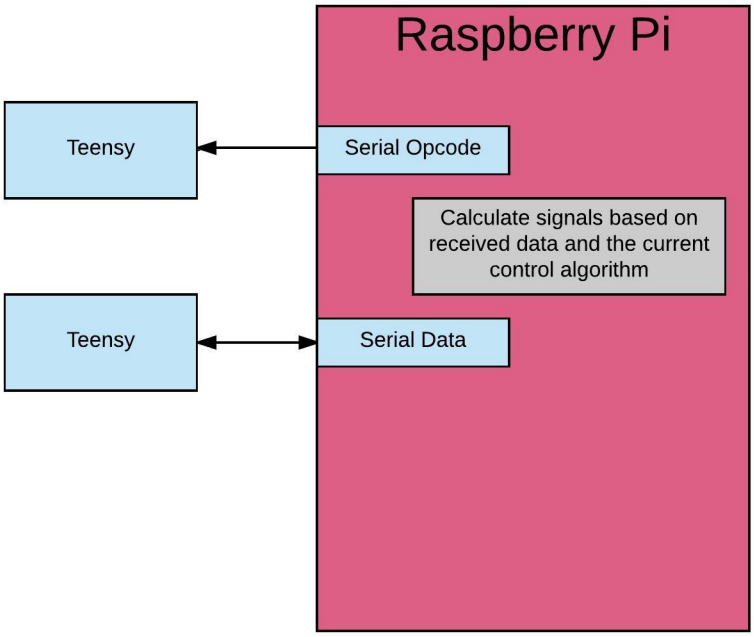


Figure 35: Raspberry Pi Software

The Raspberry Pi is more straightforward. Our responsibility is to create a SPI protocol which happens every loop cycle and to update the data we receive to specific variables. Most of the hard programming will be left up to the user to design and implement their own control algorithms.

7. MANUFACTURING

Over the winter and spring quarters, we worked to concurrently complete manufacture and assembly and fulfill the ESV competition expectations. From CDR, we had approximately a month to prepare for the ESV evaluation. We continued broad project work while creating the demo, and the demo supported implementation of our final design. Figure 36 below presents our planned timeline from CDR to completion.

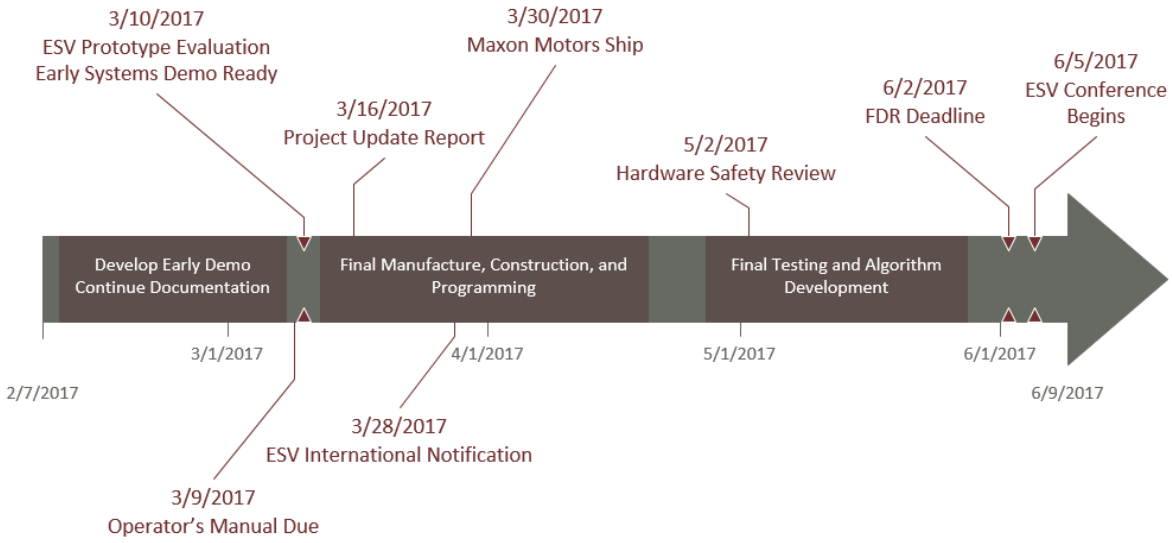


Figure 36: Planning timeline from CDR, including critical dates, milestones and deliverables.

7.1 ESV SYSTEMS DEMO

It would have been prohibitively difficult for us to manufacture the entire prototype in a month, and the Maxon motors were backordered until late March. Because of this, we did not have a full system demonstration for the ESV regional review. We intended to instead demonstrate that we will have a fully functional prototype by the ESV conference. We showed the judges that our system could communicate with hardware via the Matlab environment, and demonstrated responsive steering and throttle based on the orientation of the car.

7.2 SOURCING

We ordered the Traxxas Slash, Raspberry Pi, Sensors, and electronic prototyping equipment at the end of winter quarter and have been working with those parts to inform our design. The Maxon motors and drivers are ordered but are not scheduled to ship until the end of March. Through the partnership that Charlie has formed with Maxon, we ordered the motors and drivers at a significantly discounted cost. By February 20th, we anticipate having the rest of our parts and materials ordered. We ordered the motherboard from OSH Park, and received three unpopulated copies of the board for five dollars per square inch. The components for the motherboard were ordered from Digikey, and the raw material for manufacturing was ordered from McMaster Carr. The full bill of materials describes the exact source location for each item.

7.3 BUDGET AND BILL OF MATERIALS

Calculating the expenses of specific component costs and providing estimates for components not currently specified totals the project cost at $1502.72. We factored in tax rates and estimated shipping costs along with all material costs into our total. We used the standard California tax rate of approximately eight percent

for budget calculations. For shipping costs, we used an estimate of twelve percent of the material cost. As a non-profit project, we have been awarded $350.00 from the Mechanical Engineering Student Fee Allocation Committee (MESFAC) of Cal Poly. We were also awarded a grant from CPConnect in the form of $500.00. Our sponsor Dr. Birdsong also has $1000.00 set aside for this project in the event we need extra funding. Figure 37 below shows the allocated funds for our product and the final project cost. The full Bill of Materials can be found in Appendix I.

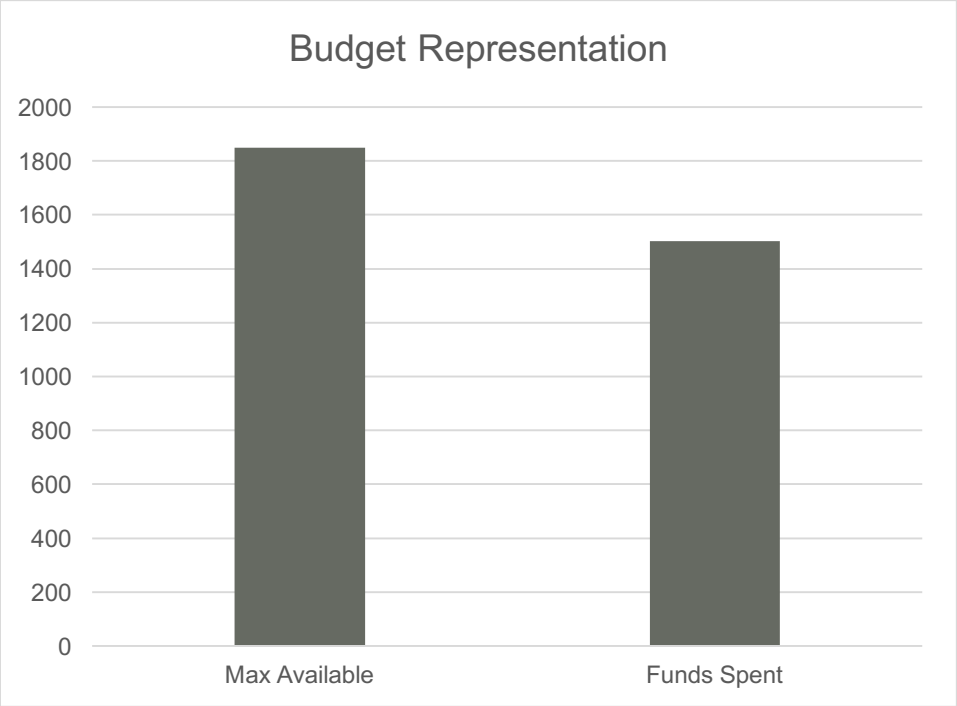


Figure 37: Current budget compared to PDR expectations and total available budget.

7.4 MANUFACTURING

The mechanical parts were designed in such a way that as few parts as possible must be manufactured with specialized tooling and knowledge. Save for the motor housing and shaft couplers which must be produced using a CNC mill, everything can be manufactured using manual machine tools. During the manufacturing phase, Jay will be working on tooling and manufacturing the CNC parts while Chris works on the remaining simplified parts.

While the Cal Poly machine shops have limited tooling available and it is often worn out, it is recommended that projects acquire their own drills. For our purposes, we will have to acquire or borrow the tooling presented in Table 10.

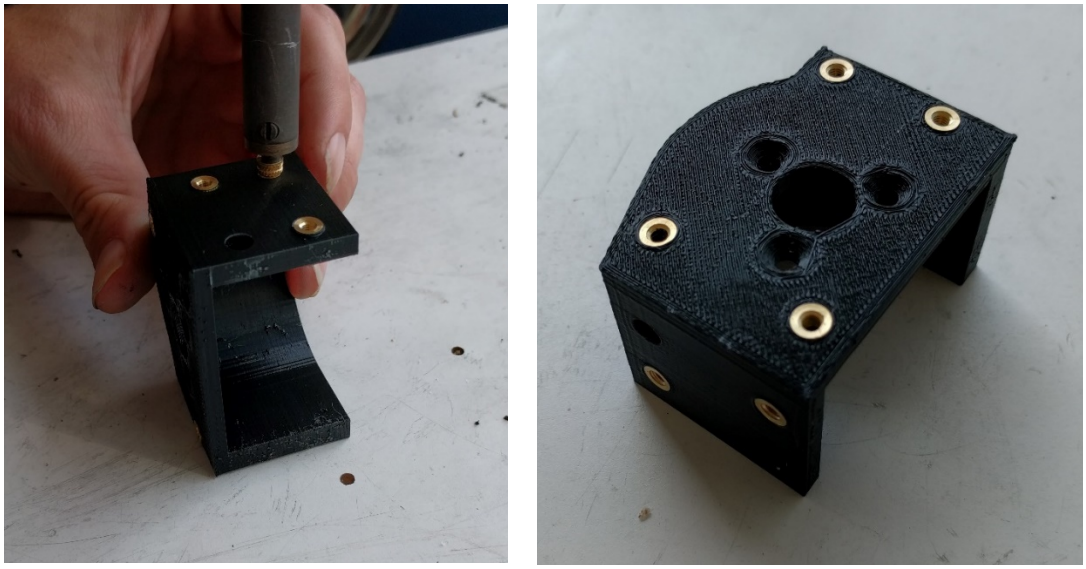
Table 10: Tooling required to manufacture the custom components.

Tooling Type
M2 pre-tap drill
M2 tap
M3 pre-tap drill
M3 tap
M3 clearance drill

The full set of manufacturing drawings can be found in Appendix J.

7.4.1 Motor Housing

The motor housings were 3-D printed out of PLA. The designs were updated to incorporate room for heat set inserts to be bonded in the plastic. The inserts are M3 brass inserts that are installed via a soldering iron with a special tip for pressing them into the plastic. The soldering iron heats the conductive metal and melts the plastic around the insert. This provides a bond that would be stronger than simply tapping the plastic and threading a screw. Originally, our plan was to CNC motor housings, but 3D printing greatly simplifies the manufacturing process while not degrading the overall build quality. As a part of the shift to 3D printing, we also changed the designs slightly. The updated drawings are present in Appendix J.



(a) (b)

Figure 38: Heat setting threaded inserts into the motor housings using the soldering iron tip (a). A motor housing with most of the inserts in place (b).

The installation of the heat set inserts worked very well, with the inserts providing more than adequate strength and quality of attachment. While the PLA is not extremely strong, it is more than adequate for

the small loads that it must bear. If there are issues with durability, the same models can be adapted for CNC machine easily, as they were initially designed with machining in mind.

7.4.2 Shaft Couplers

The shaft couplers are made of two similar parts differing only in hole pattern, and will be manufactured on the CNC using three operations. Starting with rectangular stock, the net half circle and u joint upright is shaped while a portion of the original stock is held in the vice. The part is flipped over into a set of custom soft jaws with a mating half circle cutout on one side of the vice, and the rest of the original stock is removed. The part is turned semicircle up and clamped in another portion of the soft jaws that locates the semicircle and u joint post, and the necessary holes are drilled to complete the part. With a properly designed set of soft jaws, multiple parts can be in the CNC at once, increasing the turnover speed and reducing setup overhead. We will be using identical shaft couplers on all four motors. The assembled shaft coupler can be found in drawing 225, and the component halves in 225A and 225B.

Since we 3D printed the motor housings, we were able to repurpose the stock intended for that purpose for the soft-jaws, which are shown in the CNC in Figure 39.

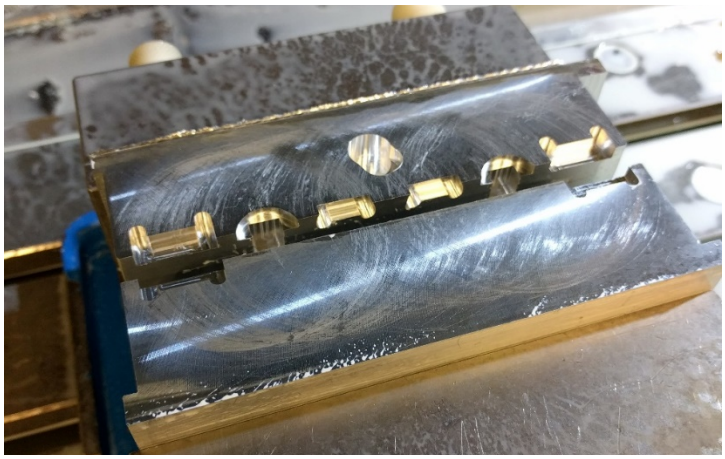
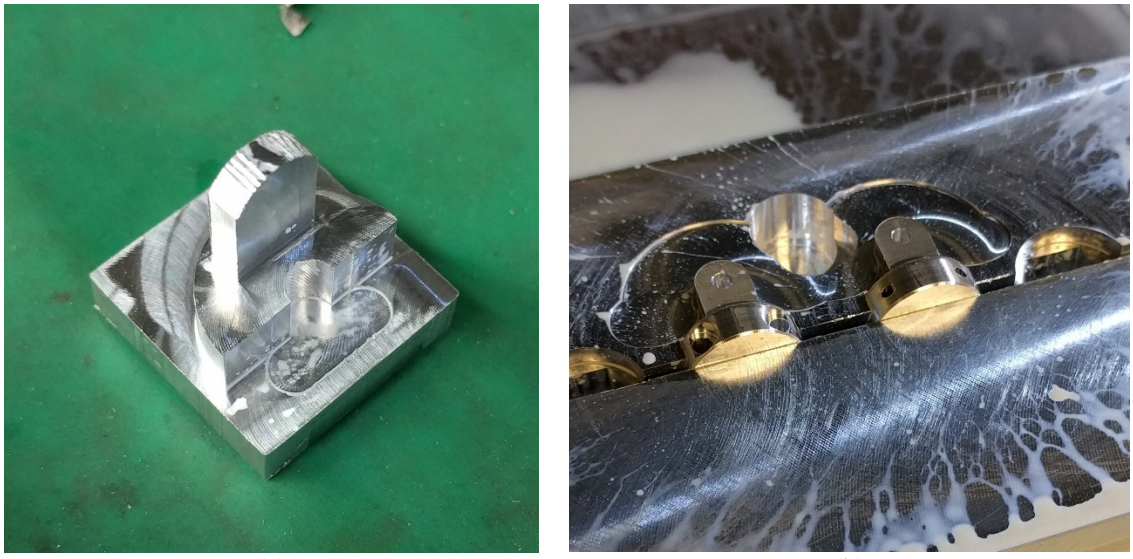


Figure 39: Finished soft-jaws for manufacturing the custom shaft couplers.

Both halves of a coupler can be produced at once, with the first operation being on the furthest outside, the next being the middle, and the final operation being the closest to the middle. This organization serves two purposes. First, ensuring that there are parts symmetrical across the jaw centerline during each operation is important. The most material removal happens in the first operation, so that operation should take place at the edges of the softjaws, where the material can support the most moment induced by the cutting tool. The shaft couplers can be seen in two different stages of the manufacturing process in Figure 40.



(a) (b)

Figure 40: Shaft coupler half after the first operation (a). Both halves in the soft-jaws after the final operation (b).

During the first pair of halves, we were focused on verifying the toolpath to prevent crashes. We found that the softjaw position shown in Figure 40b was too tight of a fit, so we reduced the outer diameter of the next coupler pairs by a few thousandths of an inch. After finishing the first pair, we assembled it on the motor with the Traxxas slip-yoke to confirm that the geometry and tolerances were properly set. This successful test fit is shown in Figure 41.

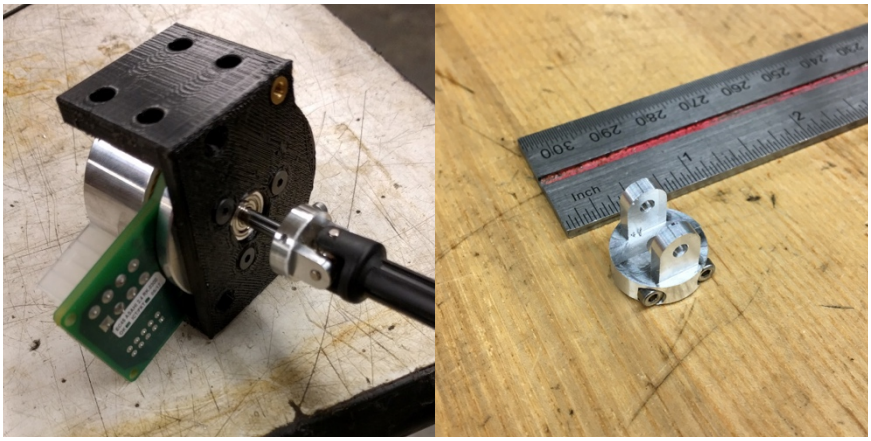


Figure 41: Test fits of the first shaft coupler, completed before manufacturing the rest.

We confirmed that the geometry and tolerances were up to par, and were ready to proceed with the other three pairs of coupler halves. Since we had already confirmed the toolpath, we were able to proceed with manufacturing the rest of the couplers extremely quickly. Each set took approximately 10 minutes to produce after installing tooling and verifying toolpaths. For future manufacturing runs of shaft couplers with an experienced CNC tech, one should expect about an hour of tooling set-up and approximately 10 minutes per coupler pair.

7.4.3 Steering Posts

The steering posts were the only part that required a lathe to manufacture. We started with a 12” long 7mm round stock and cut it to about 0.25” longer than the required length using the vertical band saw. Next, each post was faced to the necessary length. The outer diameter was achieved by turing just over half the length down to 5mm, and then turning the other side down to the same size. This operation could be omitted by simply purchasing 5mm tight tolerance rod and only facing it to length. The lathe would still need to be utilized for drilling through the center of the post, as well as tapping each side.

7.4.4 Suspension System Mounts

The A-Arm and turnbuckle mounts have very simple geometry that only required a few different operations. First, ¾” x ¾” aluminum stock was cut to about 200 thousandths of an inch larger than the width. The Bridgeport mill (~1800 RPM) was then used to cut the stock to the appropriate length, width and height. Excess material was then removed to reach the final shape of each respective mount. Next, clearance and pre-tap holes were made on the drill press using the purchased drill bits. For the turnbuckle mounts, the necessary holes were tapped with M3 thread and tap magic cutting fluid. The A-arm mount dowel slot was drilled using the M3 clearance drill bit (~3.20 mm OD, #30). This provided a slight clearance between the dowel and inner diameter of the hole. Chamfered edges were made by simply grinding the edges down with a belt sander and holding the part with a pair of vice-grips.

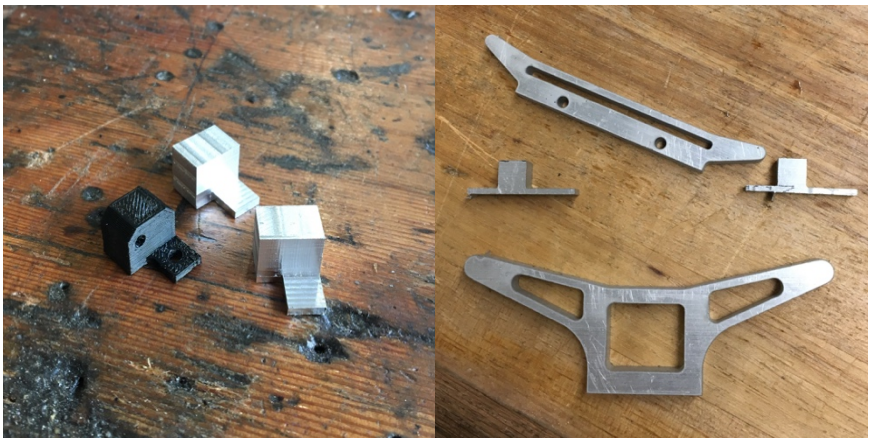


Figure 42. Suspension mounts manufactures on the Bridgeport mill (top left) and cut with the water jet (right). Manufacturing of front suspension mounts on Bridgeport mill (bottom).

The front and rear suspension mounts have a two-dimensional profile that allows them to be easily manufactured by the waterjet (Figure 42 – Top Right). The water jet uses a .dxf file created from the CAD model to cut the 5/16" thick aluminum stock. Pre-tap holes were made with a #40 drill bit and then tapped with M3 thread and tap magic. Clearance holes for mounting to the top of the chassis mounts were drilled with the M4 clearance drill.

7.4.5 Chassis

The base shape of the chassis was cut using the waterjet. Features included in this first operation was the outer profile, mounting holes for the motor housings, and the slot for the servo. The mill with a 3/16" end mill was used to cut a flat surface into the chassis where the servo could mount, allowing the servo horn not to interfere with the chassis. The mounting holes were made on the drill press and tapped with M3 thread. Slots also needed to be made on the underside of the chassis to allow space for the steering bellcranks. These were made with a ¾" forstner bit, and then ¼" forstner but drilled deeper to enclose the mounting screw.

As we were prototyping the arrangement of the components on the chassis, we realized that it would be more beneficial to mount the battery on the bottom of the car. This gives us the space to run cabling, and developing a bottom motor housing was a simple task.

The final modification of the chassis was drilling the mounting holes for the motor controllers, motherboard and IMU mount. To reduce the likelihood of error and potentially ruining the chassis, a template was cut out of acrylic with the laser cutter. This template allowed us to easily and accurately drill holes for all the components.

Figure 43. Chassis template made with the laser cutter and mounted to the chassis for hole locating (left) and 2D profiles cut on water jet for chassis and chassis mounts (right).

7.4.6 Chassis Mounts

The chassis mounts were made out of 0.08” sheet 3003 aluminum. The outer profiles and hole patterns were made using the flat pattern in Solidworks and exporting the face as a .dxf file. The profile was then cut with the water jet. The sheets were bent using the sheet metal bend brake. The flat top mounts are in drawings 204 and 304, while the bent bottom mounts are in drawings 203 and 303. Figure 43(right) shows the chassis mounts after they were cut with the water jet.

7.4.7 Sensor Mounts

Our design is extremely adaptable with plenty of space to grow. Currently, we have designed two sensor mounts. An IMU mount is located in the center of the chassis and is compatible with the BN0055 IMU. A sensor array is located at the front of the vehicle and contains a laser rangefinder, Raspberry Pi camera, and ultrasonic rangefinder Figure . The mounting holes on the top chassis mount can be used to incorporate a wide variety of custom designed sensor mounts in the future. New parts can be quickly designed and prototyped using rapid prototyping technology such as 3D printing.

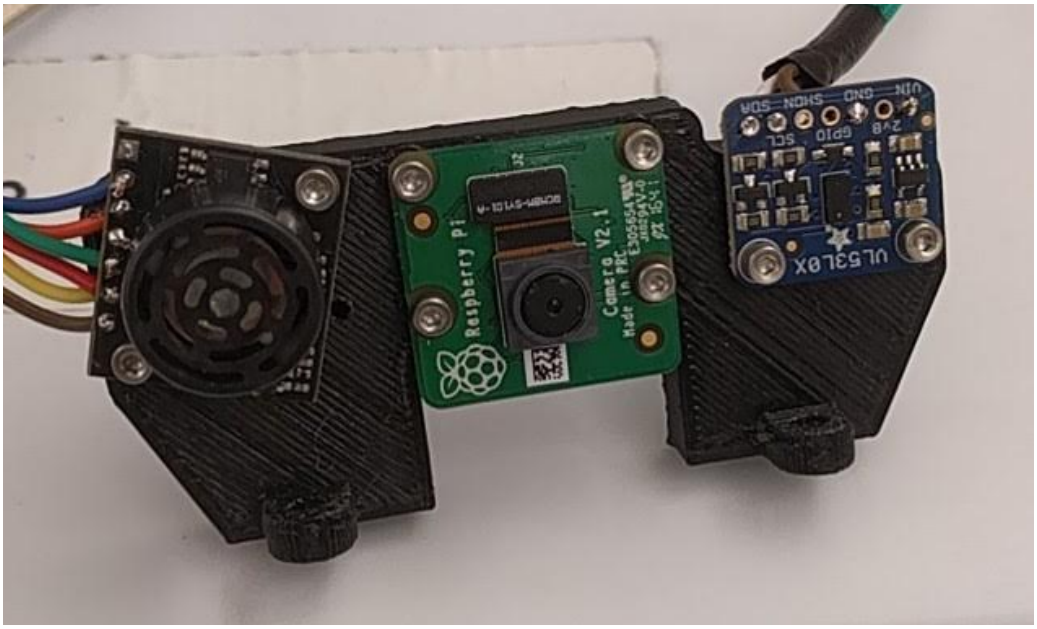
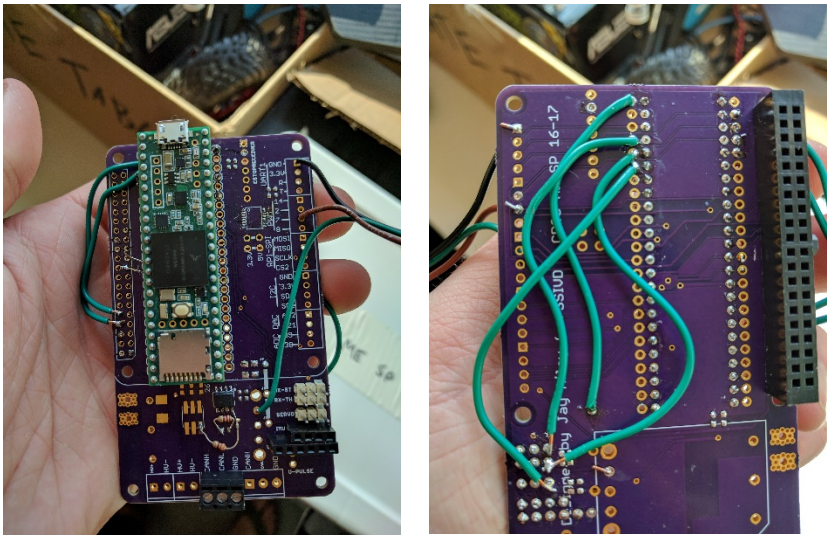


Figure 44: 3D printed sensor array featuring (from left to right) an ultrasonic distance sensor, raspberry pi camera, and laser range finder.

7.4.8 Motherboard

After designing the motherboard with the chips and connections we thought necessary to run the platform, we ordered a small prototype batch from OSH park, a US based community PCB batch ordering service. OSH Park accepts user designs and ships boards within 12 days of ordering. Pricing is $5 per square in, and you get three copies of the board. Our board is 7.5 in^2, which costs $37.50 plus tax. We sourced our components from Digikey. Components were be placed and soldered by hand in the mechatronics lab in building 192. The PCB has silkscreened labeled indicating orientation for all symmetric polarized components. The motherboard schematic is found in drawing 420, and the board layout is found in drawing 421. The components used on the motherboard are specified in drawings 422-431.

To get the first version of the motherboard working with all of our peripherals, we had to make several wire and resistor modifications to route signals differently than designed. These wire modifications are shown in Figure 44. The isolating-switching voltage regulator was designed to be mounted upside down, but the board footprint was improperly designed. For this version, we felt it was most important to get functionality like the CAN bus and serial interface working, which we successfully did.



(a) (b)

Figure 44: Wire modifications on top (a) and bottom (b) faces made to the first revision of the motherboard.

Based on the modifications we had to make to the motherboard, we redesigned certain aspects and ordered another batch from OSH park. In addition to resolving minor design and connection problems, we changed the RGB led to have a lower current so the Teensy pins could adequately sink/source the required current. We also added a power indicator LED, shown in Figure 45, which lights up when there is 3.3V coming from the Teensy. We also made the design safer by ensuring that the high voltage contacts were not easily shorted against each other, which was a problem in the first revision. We added a diode to prevent back-powering the switching regulator, and added through-holes under the input fuse, as we had torn the pads off one of the boards from the first revision. Before populating the active components, we thoroughly tested continuity across the power lines to ensure that no components would experience over-voltage or shorts.

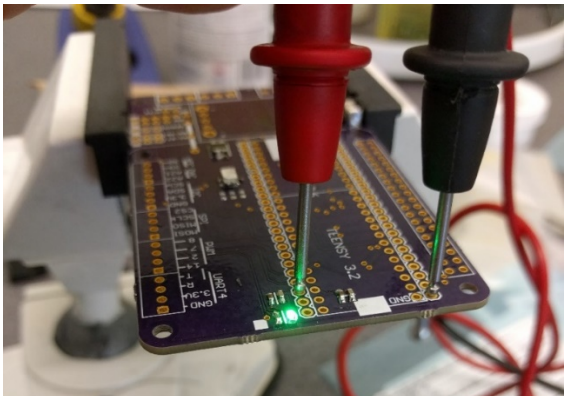


Figure 45: Probing the second iteration of the motherboard to ensure it is safe for integration of components and microcontrollers.

After soldering the components to the motherboard, we realized that the footprint had been designed improperly for the RGB indicator LED. In addition to the orientation being wrong, the footprint had been designed for a common-anode LED, while it was actually a common-cathode LED. By cutting some traces

and using small-gage wire to route the correct voltage, we were able to get the indicator successfully integrated on the board.

During testing of the second revision of the motherboard, we had both the Teensy and Raspberry Pi connected to USB ports – somehow this faulted, and it appears that we overvolted the Teensy, destroying that microcontroller. We believe that one of the USB supplies browned out, creating a voltage difference on a the connected 5V lines from each USB cable. In addition to the 5V lines being connected, both processors have their own 3.3V regulators, which are then connected with a shared 3.3V line. To help prevent this from happening in the future, we severed the shared 3.3V line so that each processor had its own. We also are ensuring that only one USB is plugged in at a given time. Fortunately, we ordered a new Teensy to replace the fried one.

When testing the motherboard's capabilities to run the motors and microcontroller off of battery power alone, we noticed that the Raspberry Pi appeared to be browning out. With a multimeter, we found that the diode was causing a voltage drop, supplying only 4.76V nominally to the 5V line. The Raspberry Pi has a minimum input voltage of 4.75V, and any ripples in our output voltage were developing the brown-out. Fortunately, the motherboard had been designed with footprints to adjust the output of the regulator – by putting a 33kΩ resistor between the output voltage and the sense pin, we were able to tune the output voltage after the diode to 5.00V. After implementing this resistor, we no longer encountered issues with the Raspberry Pi browning out.

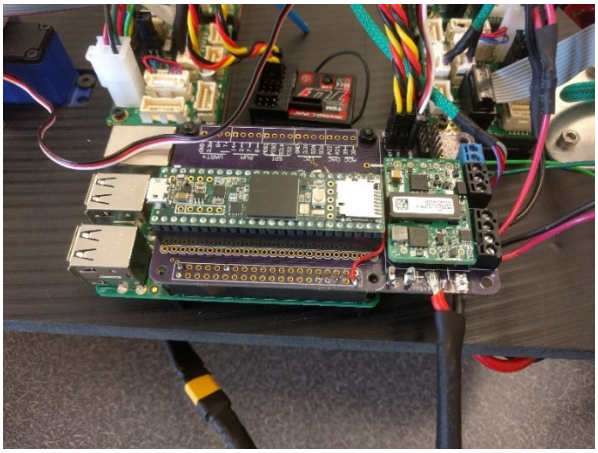


Figure 46: Final implementation of the motherboard, with Raspberry Pi and Teensy.

The motherboard performs excellently, and acts as a solid base to structure an intelligent vehicle research platform around. There are minor suggestions for further improvement, which are fully discussed in section -.

7.4.9 Wiring

To interface with our sensors and the motor drivers, we had to build a series of custom cables for our system. These cables are built with the green expandable sleeving to protect the wires and create visual organization on our final system since there are many cables that will have to be routed all over the car. At both ends of the expandable sleeving we incorporated heat-shrink to add strain relief and prevent fraying. We first build the cables to interface with the motor drivers. With the motor drivers, Maxon includes a spring-loaded

connector for power & ground. We created a spliced y-cable to connect two motor drivers to each of the high-voltage output screw terminals. The wires were first soldered together, then wrapped tightly in high gage bus wire and soldered again – this allows for better strain relief on a linear splice, which can be a common point of failure if improperly done. We had to create four sets of cables for the motors, which connect to the drivers with two cables each – a 10pin ribbon IDC cable and an asymmetrical cable to connect the motor coils and hall sensors. The asymmetrical cable has an 8pos Molex Mini-fit jr cable connecting to the motor, 4pos Molex Mini-fit jr cable connecting the motor coils to the driver, and a 6pos Molex Microfit 3.0 cable connecting the hall sensors to the driver. The connectors and crimp pins for the motor cable are shown below in Figure 47.

Figure 47: Connectors and crimp pins for the motor to driver cable harness.

While unintuitive to have an asymmetrical cable, it allows the motor driver to support DC and BLDC motors in both sensored and sensorless configurations. The completed motor to driver cable is shown in Figure 48.

Figure 48: Fully assembled motor to driver cable harness.

To connect the CAN bus of the drivers and the motherboard, we had to develop another set of cables. 3 symmetric cables with 4pos Molex Clik-Mate 1.5mm connect the drivers to one CAN bus, and a Clik-Mate to exposed stranded wire connects the driver bus to the motherboard. An unfinished CAN cable is shown in Figure 49, breaking out CAN high / low via a twisted pair and breaking out a ground line.

Figure 49: Open ended CAN connector, with twisted wire pair for CAN high and low.

For each of our sensors, we soldered wires to the sensor boards, and broke them out to a female header row. These female header rows plug directly in to male header on the motherboard, so the order of the cables must be correct. The wire arrangement for all of the motor driver cables is shown in Drawing 450.

7.5 ASSEMBLY

Part of the idea behind our design is that the motor blocks can be removed from the chassis. This separates our design into three main sections: the electronics, the chassis, and the motor blocks. Everything is attached with non-permanent mechanical fasteners. The suspension system is first connected to the motor blocks to create four different subassemblies. The motor housing can be joined together through the chassis and suspension mounts, where the chassis mounts then connect the chassis to the motor blocks. It is anticipated that all electrical components will be wired on the chassis before mounting to the rest of the assembly. However, they can also be assembled after the motor blocks are mounted. The overall design allows for easy access to the electrical components on the vehicle. Figure 50 shows the compartmentalized design that allows for easy modifications for future designs.

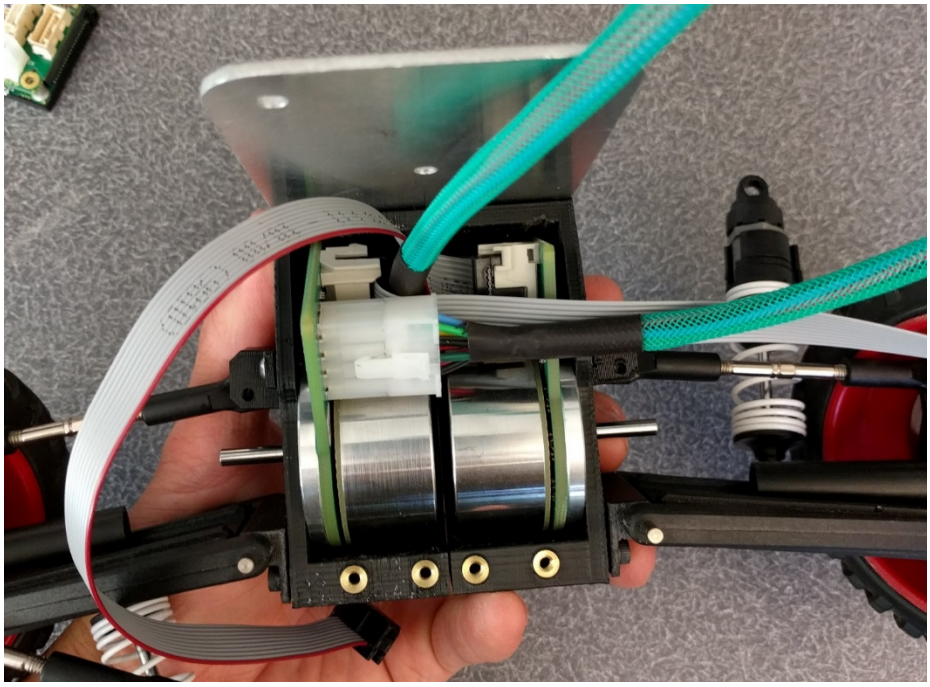


Figure 50: Motor integration on the front drivetrain.

The final assembly, shown in Figure 51 is sturdy and durable.

Figure 51: Fully assembled vehicle

7.6 HARDWARE SAFETY CONSIDERATIONS

In developing our system, we have identified that the primary safety concerns lie with the motion of the platform, the battery, and the electrical system. The motion of the system is the intended use, so it is difficult to ease the safety concerns with that, but we have limited the speed and acceleration in the motor drivers to that of our engineering specifications (2.5ft/s^2 and 10mph). For the battery, we use a good balancing charger that monitors individual cell voltage and can indicate poor battery health. For charging and storage,

we have a fire-resistant Li-Po bag. We also configured the firmware to enter a fault state if the battery voltage drops too low. Electrically, we have several fast-acting fuses throughout the system. We have a 30A fuse directly after the battery, meaning any short in the system will result in a cutoff of the battery power. On the motherboard, we have a 6A fuse in-line with our switching power supply, preventing damage and overdraw on our low voltage system. Each motor driver comes with a 10A fuse to handle any possible shorts or hardware failures. The multiple layers of fuses ensure that if something goes wrong anywhere along the system that we can handle it without damaging other parts of the system. It also allows us to be confident that we will not damage our hardware our hurt the end user through electrical shorts.

7.7 SOFTWARE BREAKDOWN

7.7.1 Teensy Software File Layout

There are many files attached to the main Arduino loop, Teensy_Firmware. In this section we will go over which files you need to be aware of and which you can ignore.

IMU: (From a documented Library)

The BNO055 takes up quite a few files. Adafruit_BNO055.cpp, Adafruit_BNO055.h, Adafruit_Sensor.h, imumaths.h, matrix.h, structs.h, quarternion.h, vector.h, BNO.cpp, and BNO.h all have to do with the IMU. If you need to look into how the IMU is handled start with BNO.cpp and BNO.h as these are the files that incorporate all the rest.

Indicator LED: (User made files)

Fault_handler.cpp and fault_handler.h hold the functions we use to initialize and set the on-board LED. There's not too much here, although if you're looking to change the LED color check out our pre-made selection in fault_handler.h.

Throttle/Steering Input and URF Distance Sensor : (User made files)

All handling of the radio receiver's PWM input is handled through input_handler.cpp and input_handler.h. Here you will find the interrupt that catches the starting timestamp of the PWM and the handling of the calculated difference between that and the ending. We limit the throttle input from –500 to 500 and the servo input from –400 to 400. The URF sensor's initializing and polling functions are also handled here.

Servo Output: (User made files)

Sending a PWM pulse to the servo happens through output_handler.cpp and output_handler.h. The useful functions here are initServo and writeServo.

CAN Library: (From a documented Library)

FlexCAN.cpp, FlexCAN.h, and kinetis_flexcan.h provide a CAN library that allowed us to simply call functions to start the Teesny's CAN bus and to read and send the CAN messages. All CAN library support comes from these two files.

Motor Controller Interfacing: (User made files)

Building from the CAN library in FlexCAN.cpp, uLaren_CAN_Driver.cpp and uLaren_CAN_Driver.h have the heavy task of creating all the specific CAN functionality to use the motor controllers. Setting up their network, initializing them, setting the target velocity and much more is found in these files.

Laser Sensor (From a documented Library)

Adafruit_VL53L0X.cpp, Adafruit_VL53L0X.h, and the twenty other files that start with "vl53l0x" all provide support for the VL53L0X Laser Sensor. The only notable files are the Adafruit_VL53L0X.cpp and the Adafruit_VL53L0X.h. The other files support these two.

Main Loop Support (User made files)

As the system is right now, we have one file that supports the main loop, loop.h. This file contains the states the main loop is allowed to cycle through.

7.7.2 Teensy Main Loop Brief

There are two critical things to know that pertain to the main loop file Teensy_Firmware: the state transitions and the global variables used. We'll start with explaining the state transitions.

INITIALIZE_PERIPHERALS

In this state we initialize any sensors we wish to use. The indicator led gets set to white here. Once we're done here we move on to INITIALIZE_CONTROLLERS.

INITIALIZE_CONTROLLERS

This state is used specifically for the Maxon motor controllers. We start by resetting the nodes in case they were previously in a fault state. We continue by initializing the CAN network for every node (motor controller) on the network and going around one by one and initializing them to various modes and settings. The notable settings are changing to Profile Velocity Mode and turning the controller into the Switched-On state. The code in uLaren_CAN_Driver.cpp is well documented for all settings. Once initialized, the code moves on to the WAIT_FOR_ARM state.

WAIT_FOR_ARM

As the name implies, we wait for the user to arm the system and motor controllers by turning and holding the steering as far right as possible. This is indicated on the LED by transitioning to a yellow color. Once armed, we then transition to the LINK_COMMUNICATION state.

LINK_COMMUNICATION

Here we set each motor controller to the Operation Enabled state which is the final state and enables the motors to be "running". In this state the indicator LED becomes red. The initial velocity here is set to '0' which gives the motors holding power. This stage is rather complex and

sometimes we need to try to rearm the controller. The code here handles this case and is quite impressive in its robustness. Once all motor controllers are armed, we go to either the RUNNING_NOMINALLY state or RUNNING_SIMULINK state depending on whether the defined variable SIMULINK is set to a '0' or '1' respectively.

RUNNING_NOMINALLY

This stage is where the loop will stay at until we encounter an error. This state starts by checking to see if any CAN messages can be processed. It then attempts to write a new value to the motor controllers if 20 milliseconds have gone by; if it hasn't then it moves on. We then check the voltage level of the motor controllers to regulate the battery level. We decided to set a minimum voltage level of 22V and if this level isn't met the LED indicator changes to a purple color and shuts down all the motor controllers. In most cases we continue through the state and attempt to write to the servo if 10 milliseconds have gone by. In this state the indicator LED is green.

RUNNING_SIMULINK

This state is very similar to RUNNING_NOMINALLY. The difference is that instead of taking input directly from the radio receiver we send it to Simulink first and use the values that Simulink sends back. To implement this, we had to use the Raspberry Pi/Simulink as the SPI master while the Teensy was the receiver. We used a one-byte opcode to indicate to the Teensy which functionality the Raspberry Pi/Simulink is trying to use. To achieve this setting we incorporated a switch case using the one-byte prefixed opcode. Other than the switch case this state also incorporates all functionality that the RUNNING_NOMINALLY state has. Here the indicator LED is supposed to be cyan but it looks more white in reality.

INDICATE_AND_LOG_ERROR

WAIT_FOR_CLEAR

These states have yet to be implemented. They were developed in the prototype phase and serve as a base for future projects to use.

7.7.3 Teensy's Pertinent Variables

SIMULINK: The SIMULINK defined variable is used to switch between the RUNNING_NOMINALLY functionality and the RUNNING SIMULINK functionality by writing a '0' or '1' respectively.

MC_VOLTAGE_THRESHOLD: This variable is used to set a minimum limit that the perceived voltage by the motor controllers must not dip below. This value is counted in 0.1V and we recommend not changing its value.

CANbus: The CANbus global variable is the key to all CAN communication. Key functions regarding this variable are specified in FlexCAN.h.

Next_state: This crucial global variable is how we interpret which state we are currently in.

All data and output variables are self-explanatory and are meant to be apparent in their meaning.

Timing variables are used to control how often we exercise certain functionality. As an example, we use the motor's timing variables to write approximately 50 times per second.

PRINT: Located in uLaren_CAN_Driver.h, if set to a '1' many items will be printed in the serial port and is quite helpful for debugging purposes.

SCALE_FACTOR: Be careful with this defined variable as it controls the factor we scale the throttle by. For normal use we advise not going above 2 to maintain similitude. The motors however are capable of going much higher (up to 8 is theoretically possible but is very strongly advised against and could destroy the motors).

7.7.4 CAN User Guide

Introduction

To communicate with our Maxon motor drivers, we had to create a partial implementation of the CANopen CAN in Automation (CiA) protocol. We encountered many difficulties while doing this, as it was our first time working with CAN, but we have successfully pieced together an implementation that works nominally. CAN stands for Communication Area Network, and is a popular communication protocol, particularly in Automotive applications. CANopen is an 'Application Layer' on the CAN protocol, meaning that it is a pre-defined way of controlling hardware using a CAN bus. This brief will overview and consolidate our discoveries and documentation, acting as a springboard for someone who wishes to create their own CANopen partial implementation.

CAN Brief

CAN uses a differential voltage signal, meaning that the hardware level of communication is interference and noise resistant as well as compatible across slightly different voltages (3.3V CAN and 5V CAN are fully cross-compatible). The protocol has built in collision avoidance, preventing one node on the bus from communicating while another is in progress. The protocol can be considered to be made up of data 'frames.' Each frame contains several different fields. The critical fields to be aware of are message ID, data length, and data. Message ID indicates which node on the bus should interpret the message, and how it should interpret it. Data length specifies how many bytes of data will be transmitted, and data contains the actual message to be passed from one node to another. Other parts of the CAN frame, including the CRC field, are automatically calculated and used for data validation and collision avoidance. A typical CAN frame is presented below in Figure 1.

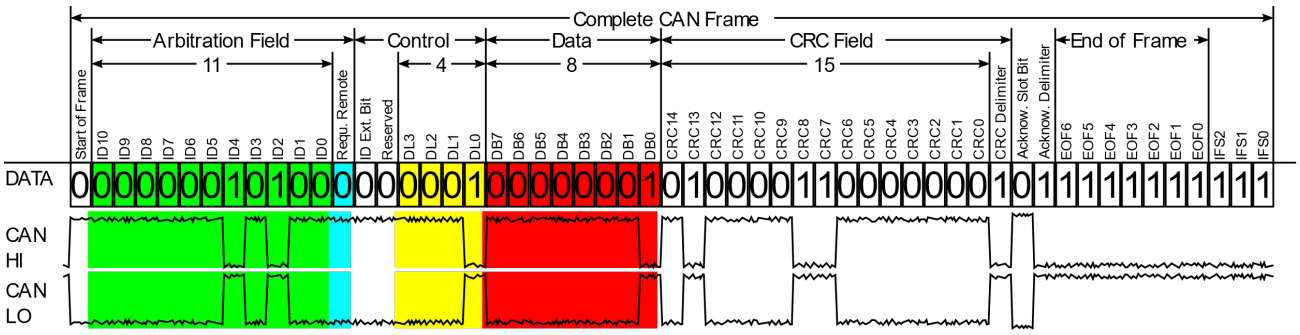


Figure 52: A Breakdown of a Complete CAN Frame

CANopen Brief

CANopen CiA is a standardized way of using CAN to control industrial actuators, allowing hardware to be modular on both the controller and actuator sides of the system. It acts as an 'Application Layer,' enabling consistent data passing and parsing on a CAN bus. CiA defines objects and commands. The 'Object Dictionary' defines standardized properties such as 'Target Velocity' and 'Actual Velocity.' These are just two examples of the hundreds of objects defined by the CiA standard. On the command side, there are ways to control or read the objects and manage the behavior of the CiA-compliant device.

There are four main parts of CiA that are important to understand and implement in the scope of our project. These are *Network Management* (NMT)*, Heartbeating, Service Data Objects* (SDO), and *Process Data Objects* (PDO). The next section overviews the basics and nuances of CANopen message types.

CANopen Methods Overview

Network Management is used in CiA compliant devices to control the communication behavior of individual nodes. This allows for more control and modularity in a given system. The NMT utilizes a master/slave architecture, with a controller (master) commanding the communication status of the nodes (slaves). On startup, the master commands all nodes or individual nodes to join the bus, allowing for full communication. Similarly, the master can command all nodes or individual nodes to leave the bus, limiting those nodes' functionalities.

Heartbeating is a method to ensure that the master is still communicating on the bus even if the individual slave node is not receiving commands. This particularly important for our project – if the SSIVD motherboard fails, we want the motor controllers to shut off and not continue driving at the same speed indefinitely. The master sends out a heartbeat message on regular intervals, and every node 'consumes' this message. If the node goes an interval (usually >20ms longer than the master interval) without seeing a hearbeat message, the controller can be configured to enter 'Safe Torque Off' mode or 'Quick Stop' mode.

Service Data Objects are a way to write to or read from individual objects on a node. An SDO is made up of two CAN frames, a request and response. There are two types of SDOs, write requests and read requests. For a write request, the request frame indicates which object to write to and what data to write to it. The response frame then returns with either an acknowledgement of the write or an error indicator. In a read

request, the request frame indicates which object to pass the data from. The response frame contains the object that was read from as well as the data that was read.

Process Data Objects are a way to write to or read from multiple objects on a node. These are often used periodically or responsively to pass data in between slave nodes. For our purposes, PDOs can be used to read or write pre-defined sets of data with less back and forth required. We can use a rxPDO (receive from the perspective of the slave node) to pass data into multiple predefined objects using one frame, and no return frame is passed. A txPDO can be used in multiple ways, including synchronously, asynchronously, and asynchronously requested. For synchronous use, the bus master can command all nodes with the configured synchronous PDO to record data simultaneously, and have them all pass this data after the next sync command. For standard asynchronous use, the slave nodes will transmit the PDO whenever one of the contained objects changes (only as frequently as configured). The other method of asynchronous use is to inhibit the automatic transmission and only respond with the data from configured objects when a request frame is sent.

CANopen Message Structure

CiA compliant messages are interpreted with information from two parts, the message ID (called COB-ID or CAN Object ID) and the data part of the frame. The following section outlines the basic format for NMT, Heartbeating, SDO, and PDO communication. Discussion regarding specific roadblocks and discoveries we had during development will then close out this portion of the SSIVD documentation. This documentation is from the perspective of a partial CiA master, and it is not meant to support development of a CiA node partial implementation. The information here may still be helpful as it consolidates pieces of information that are not readily available in a single location.

SDO's

The first piece of information to start with is the COB-ID. For CANopen, a COB-ID consists of one field of 4 bits called a function code and another field of 7 bits for a device id. The seven bits of the device id is how you differentiate between devices on the network, allowing for a maximum of 127 devices. The 4 bits of the COB-ID for the function code is used to describe what type of a message the node is sending. For example, sending an SDO to a node would require a COB-ID of 0x600 for the function code + the node id of the device you are communicating with. When the node responds to this, it will respond with a function code of 0x580 + its own node id. Note that when you look up the function code it pertains to how the node recognizes it, so using our previous example we'd know that 0x600 is an SDO RX and a 0x580 is an SDO TX. That's it for the COB-ID.

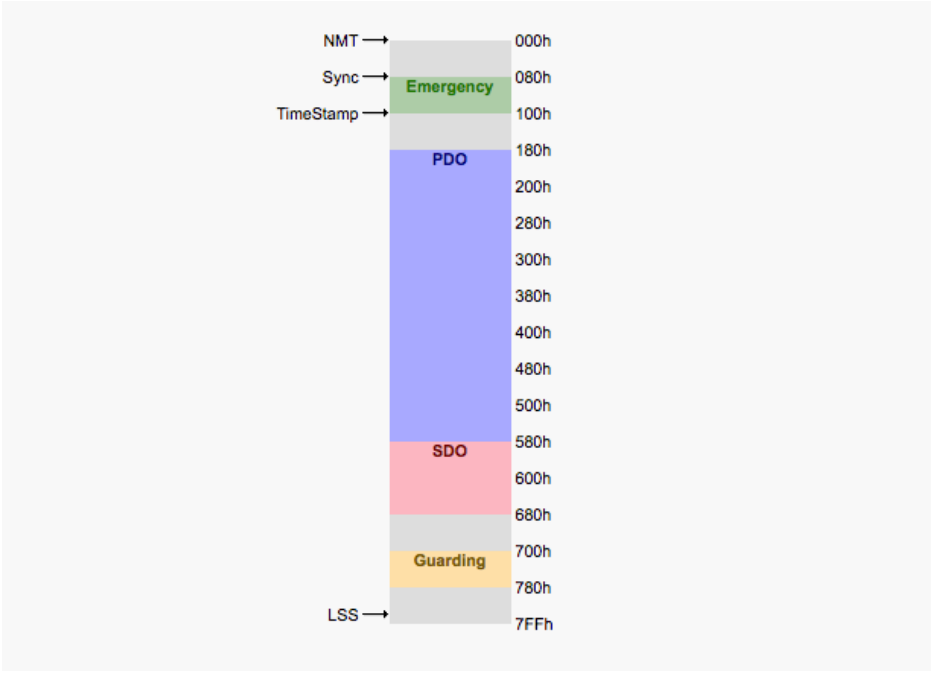


Figure 53: COB-ID Values Broken Up by Function Codes

The next place to start is the payload data. There are 8 bytes in this region of a CAN message. In CANopen, the first 4 bytes are used to describe the message being sent leaving the last 4 bytes as the place to store the actual data. Figure 60 is included below as a visual reference.

COB-ID	Command	Index		Subindex	Service Data (Parameters)			
11 Bit	Byte 0	Byte 1 (LSB)	Byte 2 (MSB)	Byte 3	Byte 4 (LSB)	Byte 5	Byte 6	Byte 7 (MSB)

Service data longer than 4 bytes (Objects 1008h, 1009h, 100Ah) are transferred by the segmented protocol.

Figure 54: Notable CANopen Fields

As you can see from Figure 60, the first four bytes are used to hold a Command Specifier, Index, and a Subindex. The Index and Subindex refer to the object in the Object Dictionary that you are attempting to write to or read from. You can find these specific objects for our motor controllers in the Maxon Firmware Specification for that specific device. The tricky part is the Command Specifier. This took us a long time to figure out. We read somewhere the wrong values to use for this and were quite frustrated when they weren't working. There are different values to use depending on whether it's a read or a write. 0x40 will work for any read request. For Write Requests, the CS value changes depending of the length of the data you wish to write; but remember, this length must match the length of the object as specified in its Object Dictionary. Figure 4 below describes the correct values to use as a write Command Specifier.

Command Code	Meaning
0x23	Write Dictionary Object reply, expedited, 4 bytes sent
0x27	Write Dictionary Object reply, expedited, 3 bytes sent
0x2B	Write Dictionary Object reply, expedited, 2 bytes sent
0x2F	Write Dictionary Object reply, expedited, 1 bytes sent

Figure 55: Command Codes for a Write Request Frame

This about sums up a SDO CANopen message frame. We talked about the COB-ID and how SDO frames have function codes of 0x580 and 0x600 and the need to add the node id to these base values. We went over the payload format and what values are expected to reside in each byte. Below you will find an example of a SDO Read Request followed by a SDO Write Request. One final note of advice is to recognize the format of the Index and Parameter fields. Both of these fields expect the Least Significant Byte (LSB) to be written first, followed by the corresponding bytes of increasing significance ending with the Most Significant Byte (MSB).

```
//statusword request
msg.id = 0x600 + node_id;
msg.ext = 0;
msg.len = 4;
msg.timeout = 0;
//
msg.buf[0] = 0x40;
msg.buf[2] = 0x60;
msg.buf[1] = 0x41;
msg.buf[3] = 0x00;
//
```

Figure 56: Example of a Read Request SDO

```
//tell MC's to go to shutdown state
msg.id = 0x600 + node_id;
msg.ext = 0;
msg.len = 6;
msg.timeout = 0;
//
msg.buf[0] = 0x2B;  //68
msg.buf[2] = 0x60;  //60
msg.buf[1] = 0x40;
msg.buf[3] = 0x00;
//
msg.buf[5] = 0x00;
msg.buf[4] = 0b00000110;
msg.buf[6] = 0;
msg.buf[7] = 0;
```

Figure 57: Example of a Write Request SDO

PDO's

Since we've gone over the process of using SDO's we will now explain how to use Process Data Objects (PDO). PDO's can be thought of almost like a custom SDO. As previously mentioned PDO's can be configured in many different ways. For our use we configured each node to accept the Target Velocity and the accompanying Controlword that enables the controller to start to reach the new Target Velocity (we will go over how to use the Maxon controllers, Target Velocity, and Controlwords later on). Once we set this up we simply use the function code of PDO4 (0x500) + the id of the node we are directing as the COB-ID. We then set bytes 0 and 1 to be the new Controlword and bytes 2-5 as the value of the new Target Velocity. Most of the work on PDO's come from the EPOS Software as we configure the motor control to accept the PDO we envision. To configure a PDO, you must first set the number of objects to 0. You can then modify the assigned objects, and the datalength for each. After completing this to satisfaction, you can set the number of mapped objects to your desired number.

```
//initiate target velocity
//tell MC's to go to operation enabled state w/target velocity enabled
msg.id = 0x500 + node_id;
msg.ext = 0;
msg.len = 6;
msg.timeout = 0;
//
msg.buf[0] = 0x0F;
msg.buf[1] = 0x00;
memcpy(&(msg.buf[2]), (void *)(&throttle), 1);
memcpy(&(msg.buf[3]), ((char *)(&throttle) + 1), 1);
memcpy(&(msg.buf[4]), ((char *)(&throttle) + 2), 1);
memcpy(&(msg.buf[5]), ((char *)(&throttle) + 3), 1);
msg.buf[6] = 0;
msg.buf[7] = 0;
```

Figure 58: Example of a Configured PDO

Using CANopen with the Maxon 50/5 Controller

Starting up the CAN Network

The network initialization is very straightforward. It takes one command to turn on the network of every node (there is also the option to select which nodes to turn on). Figure 65 below is what we used to do this. The message id must be '0' for this to work but data byte 1 is where you select the node id of the specific node you wish to enable (a '0' will turn on every node in the network).

```
//"Start Remote Node"
msg.id = 0;
msg.ext = 0;
msg.len = 2;
msg.timeout = 0;
msg.buf[0] = 0x01;
msg.buf[1] = 0;
```

Figure 59: Start CAN Network Example

Initializing the Controller

There are a few things to be done for initializing the motor controller. As written in the firmware specification, there is a state diagram to be followed to get the controller into Operation Enabled mode. For good practice we start it up through the Shutdown State and bring it to Switched On, where we then wait for the user to arm the system. To do this we must write new values to the controlword. The controlword is the key to controlling the state of the device. Below you will find examples as to the messages we sent for Shutdown and Switched On. We also need to set the mode of the controller into Profile Velocity Mode as this is the mode we chose to run in our project.

```
//tell MC's to go to shutdown state
msg.id = 0x600 + node_id;
msg.ext = 0;
msg.len = 6;
msg.timeout = 0;
//
msg.buf[0] = 0x2B;
msg.buf[2] = 0x60;
msg.buf[1] = 0x40;
msg.buf[3] = 0x00;
//
msg.buf[5] = 0x00;
msg.buf[4] = 0b00000110;
msg.buf[6] = 0;
msg.buf[7] = 0;
```

Figure 60: Controlword Write of Shutdown State

```
//initialize MC's to profile velocity mode
msg.id = 0x600 + node_id;
msg.len = 5;
msg.timeout = 0;
msg.buf[0] = 0x2F;
msg.buf[2] = 0x60;
msg.buf[1] = 0x60;
msg.buf[3] = 0;
msg.buf[4] = PROFILE_VELOCITY_MODE;
```

Figure 61: Profile Velocity Mode Selection

```
//tell MC's to go to switch-on state
msg.id = 0x600 + node_id;
msg.ext = 0;
msg.len = 6;
msg.timeout = 0;
//
msg.buf[0] = 0x2B;
msg.buf[2] = 0x60;
msg.buf[1] = 0x40;
msg.buf[3] = 0x00;
//
msg.buf[5] = 0x00;
msg.buf[4] = 0b00000111;
```

Figure 62: Controword Write of Switched-On State

Arming the Controller and Motor

Arming the controller only takes one step and again it is a change to the controlword. This will bring the controller finally to the Operation Enabled state. It is advised that you wait 500-1000ms to allow the controller to switch into this armed state.

```
//tell MC's to go to operation enabled state
msg.id = 0x600 + node_id;
msg.ext = 0;
msg.len = 6;
msg.timeout = 0;
//
msg.buf[0] = 0x2B;
msg.buf[2] = 0x60;
msg.buf[1] = 0x40;
msg.buf[3] = 0x00;
//
msg.buf[5] = 0x01;
msg.buf[4] = 0b00001111;
```

Figure 63: Controlword Write of Operation Enabled State

Nominal Use

At this point in time the motors will have a holding torque, as the current written Target Velocity is '0'. To change this value we simply write a 4 byte integer into the Target Velocity Index in the Object Dictionary. An example is shown below. That last thing we need to do is enable the controller to achieve this Target Velocity. Again we will change the controlword, finally writing a '0' into bit 8 and watching as the motor finally spins!

```
//write to Target Velocity
msg.id = 0x600 + node_id;
msg.ext = 0;
msg.len = 8;
msg.timeout = 0;
msg.buf[0] = 0x23;
msg.buf[2] = 0x60;
msg.buf[1] = 0xFF;
msg.buf[3] = 0;
//
memcpy(&(msg.buf[4]), (void *)(&throttle), 1);
memcpy(&(msg.buf[5]), ((char *)(&throttle) + 1), 1);
memcpy(&(msg.buf[6]), ((char *)(&throttle) + 2), 1);
memcpy(&(msg.buf[7]), ((char *)(&throttle) + 3), 1);
```

Figure 64: Writing to Index 0x60FF (Target Velocity)

```
//initiate target velocity
//tell MC's to go to operation enabled state w/target velocity enabled
msg.id = 0x600 + node_id;
msg.ext = 0;
msg.len = 6;
msg.timeout = 0;
//
msg.buf[0] = 0x2B;
msg.buf[2] = 0x60;
msg.buf[1] = 0x40;
msg.buf[3] = 0x00;
//
msg.buf[5] = 0b00000000;
msg.buf[4] = 0b00001111;
msg.buf[6] = 0;
msg.buf[7] = 0;
```

Figure 65: Controlword Write with Target Velocity Enabled

7.8 RECOMMENDATIONS FOR CONTINUED SOFTWARE DEVELOPMENT

7.8.1 Simulink

Currently, our model is made primarily using MATLAB code that we run through Simulink using Interpreted MATLAB Function Blocks. These blocks, quite unfortunately, are unable to be run on a target hardware, which means that the computer must be made to run Simulink on its own CPU while the MATLAB code calls the Raspberry Pi to do specific instructions over Ethernet. We ended up running into this issue late in our development while trying to integrate all of our subsystems together.

The goal for this project was to let Simulink manipulate an small-scale vehicle in real-time. While we achieved this, having a cord connected to the Raspberry Pi at all times was not what we had in mind. To allow Simulink to run without a tether there are a few options. It appears to be easiest to use S-Function blocks to write a device driver that will do the same functionality as we implemented in MATLAB. We are unsure how long this task might take and someone more familiar with Simulink would be more suited to this task.

Notable features in the MATLAB code include establishing a connection to the raspberry pi, creating a serial data object, and using this serial object to send data to the Teensy microcontroller. The first two features are easy enough to do in MATLAB and the first would not be necessary when configured to run on the raspberry pi itself. What someone would need to do first is to initialize a serial path using the default tx and rx pins on the raspberry pi. The next necessity is to use the serial port to transmit and receive data. We established an arbitrary connection method that allows the Teensy and Raspberry Pi microcontrollers to ask for and receive specific information. Table 1 below shows the mapping we use for various requests.

Table 11: Serial Connection Sequences

Initial Byte from Raspberry Pi	Corresponding Functionality	Byte Length the Teensy Expects
11	Write the Steering Value (-400 to 400)	2
12	Write the Right Front Motor (-1000 to 1000)	2
13	Write the Left Front Motor (-1000 to 1000)	2
14	Write the Left Rear Motor (-1000 to 1000)	2
15	Write the Right Rear Motor (-1000 to 1000)	2
21	Read the Steering Input Value (-400 to 400)	2
22	Read URF Distance Sensor	2
23	Read GYRO_X position data	2
24	Read GYRO_Y position data	2
25	Read GYRO_Z position data	2
26	Read Throttle Input Value (-500 to 500)	2

As you can see from the table above, the method to connecting to the Teensy requires a sequential method. The Teensy expects an instruction byte to map the proceeding data to a specific variable. For example, if we wanted to tell the right rear motor to operate at 200 rpm, we would send one byte with an opcode of '15' followed by two bytes in 'int16' form to specify a value between 1000 and –1000.

7.8.2 Adding a Sensor

All the sensors communicate directly to the Teensy microcontroller. If you need to hook up another sensor there are two distinct steps you need to go through. The first is to demonstrate reliable connection between the Teensy and the new sensor. For almost every sensor imaginable there is already someone who was created a reliable demo for you and has put it on the web for free. Search around and see if you can find this; if not, you'll have to build one the old fashioned way through documentation.

Once you have a demo, open up the Arduino IDE (if it's not already installed visit our Software Installation document) and plug in the source code for the demo. Remember to include any accompanying files the source demo might need. From here, upload it to Teensy and watch it work (if it doesn't, debug and use the internet to correct any mistakes). Once you have a working demo you are done with the first step.

The next step is to insert your main sensor variable(s) into the top of the page next to all the other variables. For good coding practices and to stick with our design, you should develop one function to initialize your sensor. Once you have this, call that function from the INITIALIZE_PERIPHERALS case in our main loop and refer to Figure 1. Also, take this time to copy the needed files from your demo into our Teensy_Firmware folder.

```
void loop() {
  CAN_message_t msg;

   switch(next_state)
   {
      case(INITIALIZE_PERIPHERALS):
         //initialize other things
         if (PRINT)
         {
          Serial.println("Initializing Peripherals");
         }
         initPWMin();
         initServo();
         next_state = INITIALIZE_CONTROLLERS;
         break;
```

Figure 66: Insert Initialize Function in the INITIALIZE_PERIPHERALS Case

For normal operation, the car will either run in the RUNNING_NOMINALLY case or the RUNNING_SIMULINK case, depending on whether you are running Simulink (this can be turned on or off from the SIMULINK defined variable at the top of Teensy_Firmware.cpp).

Teensy_Firmware | Adafruit_BNO055.

```
#include "uLaren_CAN_Driver.h"
#include "FlexCAN.h"
#include "kinetis_flexcan.h"
#include "input_handler.h"
#include "output_handler.h"
#include "loop.h"
#include "fault_handler.h"
#include "structs.h"

#define NODE_1 1
#define NODE_2 2
#define NODE_3 3
#define NODE_4 4

#define MC_VOLTAGE_THRESHOLD 220

#define SIMULINK 1

//main globals
FlexCAN CANbus(1000000);
State next_state;
```

Figure 67: Where to Find the SIMULINK Variable ('1' means on, '0' means off)

In the correct RUNNING_"____" loop, insert whatever code you want to continually run. If you are wanting to incorporate this sensor into Simulink, read on.

The first thing you are going to want to do is to locate the switch case in the RUNNING_SIMULINK case. Here is where the magic happens and the transferring of variables to the Raspberry Pi/Simulink takes place. Pick a number (any number from 27 – 127 will work) and setup a case block for your variable as we have done in numbers 21-26. You will want to do a Serial1.write() with the first parameter being your variable and the second number being the number of bytes your sensor variable takes up (all of ours ended up being two bytes long).

```
case 22:  //URF write to pi
  Serial1.write((const uint8_t*)&URF_dist, 2);
  break;
case 23:  //gyroX write to pi
  Serial1.write((const uint8_t*)&g2, 2);
  break;
```

Figure 68: Two Examples of Sending a Sensor Variable to Simulink

The second and last step is to incorporate the serial connection on the Raspberry Pi/ Simulink side. This is surprisingly easy to do since our model is mostly running through MATLAB code. Figure 4 provides an example function we have that incorporates the same URF data sequence that we see coming from the Teensy in Figure 3.

```
function y = readURFInput(~)

    global myserialdevice;
    global myconnection;

    if (myconnection > 0)
        write(myserialdevice, 22);
        x = typecast(read(myserialdevice,2), 'int16');

        y = double(x);
    else
        y = 0;
    end
end
```

Figure 69: Example of Sending Serial Opcode and Receiving the Sensor Data

To change this function to incorporate your own sensor data, all you need to change is the values in the write and read functions. First change the '22' opcode to the same value you chose to use in the Teensy switch block. Then modify the read function to match the number of bytes you set the Teensy to send (if your variable was two bytes you don't need to change anything on the read function). Also, if your variable is different than two bytes a change to the parameter 'int16' is needed to be appropriately changed ('int8' and 'int32' are usable parameters).

The last thing to do is to create an "Interpreted MATLAB Function" block in Simulink and add your function as a parameter. Create the necessary wiring from the "Create Connection Variable" block we made and design your own algorithm!

7.8.3 Teensy Loop

We wanted to implement error handling from the RUNNING_NOMINAL and RUNNING_SIMULINK states but we will leave this up to future contributors. When we encountered an error we expected to first transition to the INDICATE_AND_LOG_ERRORS case. Once done there, we would transition to the WAIT_FOR_CLEAR state and wait for the user to rearm the system using some predefined action (possibly turning the steering completely to the left). At this point there are many different transitions you could make. One possible solutions would be to go back to the INITIALIZE_PERIPHERALS state to give the system a reboot of some kind. Another could be to just transition back to a running state. Ultimately the decision is up to you and creating your own state(s) could end up being the best solution.

8. DESIGN VERIFICATION

8.1 TESTING PLAN

We performed a series of tests with and on the vehicle to ensure that it meets all of our design specifications and customer requirements. We used quantitative methods– mass properties and CG position, as well as compared real results to that of our steering model developed earlier. To test the more qualitative parameters, we examined the system for any defects and provided recommendations for future fixes or improvements. Simulink modeling will help us to tune the model vehicle parameters to allow for more effective control systems.

See Appendix H for the full description of our design verification plan.

8.2 BUILD QUALITY EVALUATION

In lieu of a destructive test that would potentially render our hard work useless, a qualitative evaluation was conducted to establish potential areas of structural weakness. Where possible, modifications were made to strengthen any weak areas. However, there are a few areas where the mechanical system could be improved in the future to create a more durable and dynamically similar vehicle to that of a full scale vehicle. These will be discussed in a later section.

8.2.1 Modifications

Bump steer occurred due to the position of the steering turnbuckles on the steering blocks. As a result, we raised the mounting position and threaded a larger screw through ball joint as shown:

Figure 70. Steering linkage modification to mitigate bump steer.

This adjustment effectively reduced the bump steer, as well as the toe angle. We also noticed that the servo did not sit evenly its slot due to the structural ribs on the mounting flanges. As a result, the chassis slot was filed down to incorporate a slot, as shown below:

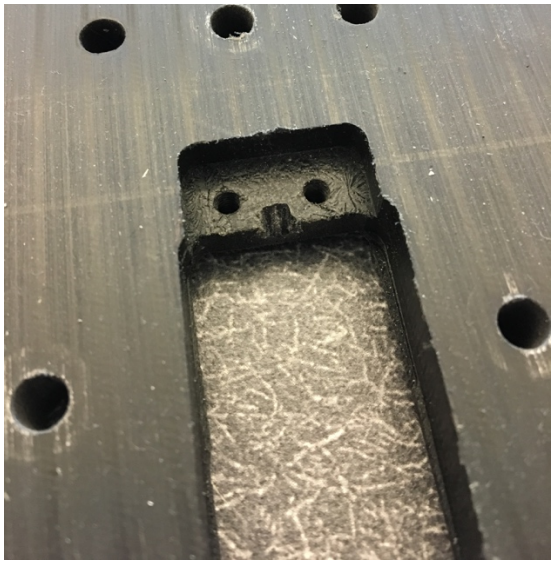


Figure 71. Grooves filed in servo slot to incorporate the structural ribs on the servo.

This change should be added as an extra manufacturing step for future chassis using the same servo. Spacers were also added between the hex standoffs and chassis mount on the front end of the chassis. This was due to a slight difference in height between the top of the standoff and top of the motor housing that was initially thought to be negligible.

Changes were also made to stiffen the suspension by adding Traxxas supplied spacers between the top of the spring and ultrashock endcap. The size and amount of spacers used set the sag to about 30% in both the front and rear. The original progressive springs in the rear were moved to the front, and a 10% stiffer set of Integy springs were installed in the rear. Clips and zip ties were also added for cable management.

Another important note is that there were no necessary modifications to the motor shaft when aluminum shaft couplers were used. This greatly simplified the manufacturing process and eliminated any chances of unnecessary damage to the motors.

Finally, during expo we noticed that the a-arm mounts became loose over time. This was due to the eccentric vibration of the wheels when ran on the display stand. The loosening may not be an issue when not on the displaying stand, but it will be worth monitoring in the future. Future modifications of the 3D print could be made that incorporate the A-arm mounts into the motor housing deisgn.

8.3 QUANTITATIVE TESTING

8.3.1 Center of Gravity Position

The center of gravity(CG) can be determined based upon the static weight distribution of the vehicle. Four standard kitchen scales (11 lb max) were purchased to give an accurate reading of the reaction forces at each tire-ground interface (Figure 72). First, we determined the longitudinal position of the CG through the relationships:

$$b = \frac{R_R L}{W} \quad , \quad a = L - b$$

As a result, we needed to measure the wheelbase, and reaction force at both the front and rear axles.

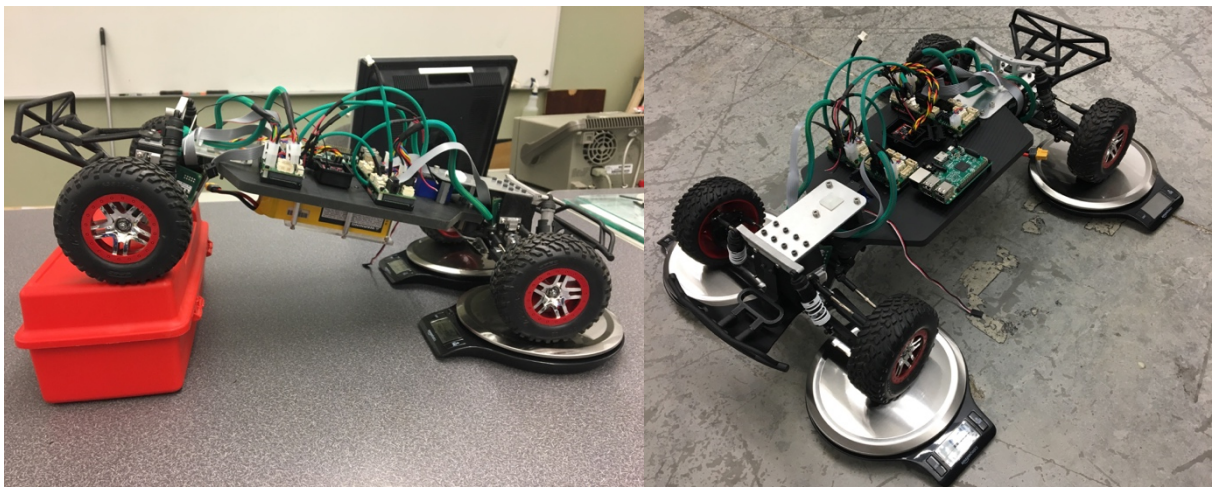


Figure 72. Lifted rear end for measuring CG height(left) and static weight distribution (right).

This meant that we would have uncertainty in the calculated value due to measurement error. It should be noted that the reading on the scale fluctuated based upon the position of the tire on the scale, but this variation was ignored for purposes of the uncertainty analysis. Any reported uncertainty is due to resolution uncertainty only. Through statics relationships, we were able to determine the approximate CG height by lifting the rear end of the vehicle approximately 120 mm. This causes a weight shift towards the front of the vehicle, which can be used to determine the position of the CG through the relationship:

$$h = \left[a - L\left(\frac{R_f}{W}\right)\right]\cot(\theta) + r$$

$$where,\ \theta = sin^{-1}(H/_L)$$

The results from this testing can be seen in Table 12, while the data collection and uncertainty analysis is tabulated in Appendix K.

Table 12. Position of center of gravity with propagated resolution uncertainty.

Variable	Value	Unc.	Units	Description
W	3.85	+/-0.001	[kg]	Total weight of vehicle
b	196.8	+/-2.5	[mm]	Distance from front to CG
a	204.2	+/-2.5	[mm]	Distance from rear to CG
h	75.1	+/-1.2	[mm]	Height of CG from ground

8.3.2 Steering Model Verification & Repeatability

In our firmware, we control the steering servo by controlling the pulse-width of a PWM signal, which the servo interprets as an angular position setpoint. While we know what values we are passing the servo, it is important to understand what steering angle these values actually produce so we can develop a proper correlation to the bicycle dynamic model. Using a protractor, we measured the angle of the inside wheel relative to forward that was produced in response to a given input to our steering function. Our system has some backlash induced hysteresis, which we attempted to correct for to find the 'center' output steering angle. Our control scheme does not account for the hysteresis, and it is something that can likely be improved by tightening mechanical tolerances. The steering data is presented in Figure 73, with the resulting input to angle correlation shown.

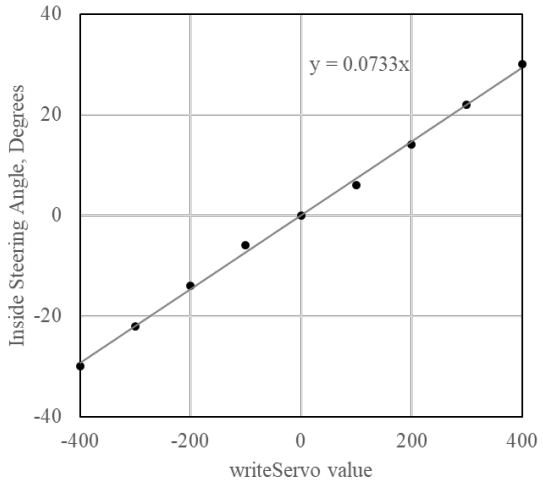


Figure 73: Inside steering angle versus the input value to our servo function.

For testing purposes, we developed a steering profile to be applied to both the model and the platform. The input function, shown in Figure 74, should produce an 'obstacle avoidance' profile – a lane change like maneuver in one direction, and a lane change line maneuver to return to the original path.

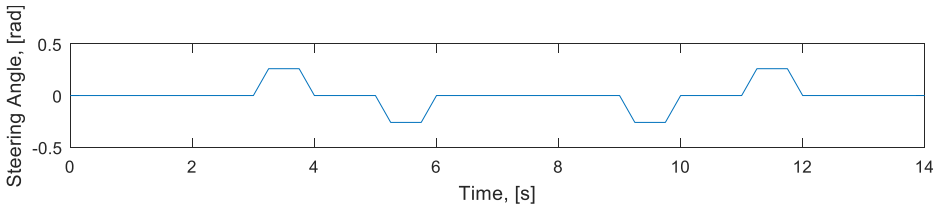


Figure 74: Steering angle profile for the repeated steering profile test.

We developed a simple function to allow for the repeated testing and data collection for an input profile. The steering profile, including steering angle, throttle, and times, were pre-defined in arrays, and the loop would interpolate between points to create a motion ramp. For this test, the user is required to hold down the throttle. If they let go, the car comes to a stop quickly. This allows the user to abort the test if it is nearing a wall or getting out of control. The abort functionality was thoroughly tested on a stand before placing the vehicle on the ground. The 14s profile took the car about 10 meters. The car is at starting position and Chris is standing at the finish location in Figure 75a, and the finish location spread of 3 discreet tests of the same profile is shown in Figure 75b.

(a) (b)

Figure 75: Starting point of the repeated steering profile test (a) with Chris standing at the endpoint. Marked end locations of the repeated steering profile test (b) demonstrating high repeatability across large distances.

During these tests, we were collecting raw linear acceleration data and filtered Euler vector orientation data from the BNO055 imu and logging it for comparison to the model. We used the Euler vector to develop the output yaw relative to the starting orientation. Figure 76 shows the comparison of model and real yaw resposnes.

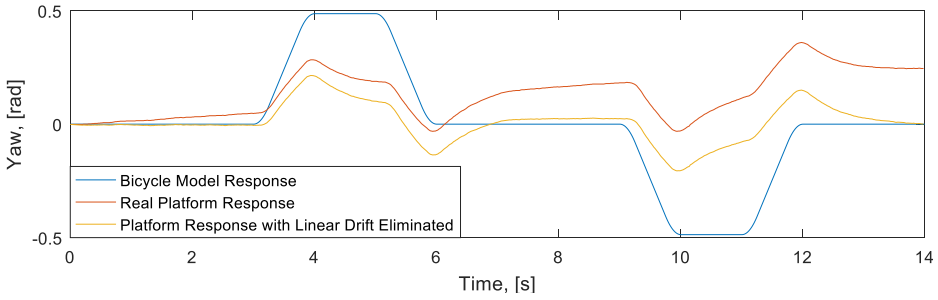


Figure 76: Comparison of steering profile yaw from the bicycle model, the platform response, and filtered platform response.

Note that Euler vector angles are from 0 to 360 degrees, and the output had to be conditioned to be a signed output. Knowing that our expected yaw angles were to be less than 180 degrees (as the car was not making a u-turn), we were able to condition the response with a simple if statement applied to each data point, shown in pseudocode below:

```
if (angle_in > 180): angle_out = - (360 - angle_in)
else: angle_out = angle_in
```

We know that the drift is due to a misalignment in our steering center, which we would expect to produce a linear drift. The drift does appear to be very close to linear, and we can filter it out using an incremental function applied to the whole array. We also know that the ending yaw should be the same as the starting yaw, because this is meant as an obstacle avoidance algorithm. This incremental function is shown in pseudocode below:

```
for each yaw in yaw_array:
    yaw = yaw - yaw@t_max * (t / t_max)
```

Implementing this filter allows us to view what the response would look like if the steering was tuned more accurately, which can be done in the future.

We also felt it was important to look at the acceleration response of the vehicle during the steering profile. Since the profile is to be operating at a constant velocity, and is operating on flat ground, we are most interested in the lateral acceleration, or acceleration to the left or right from the frame of reference of the vehicle. The bicycle model response and platform response measured by the IMU are shown in Figure 77.

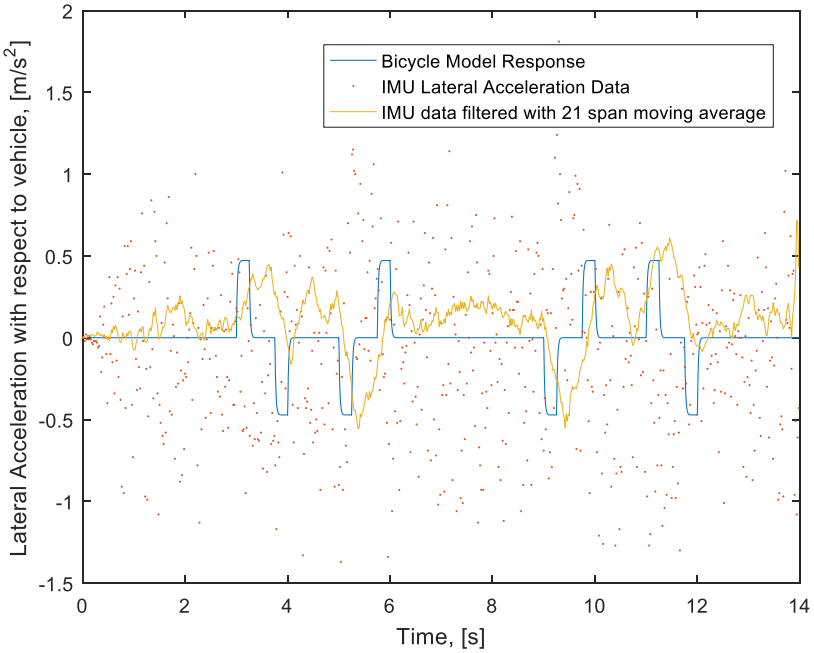


Figure 77: Comparison of steering profile lateral acceleration from the bicycle model, the platform response, and filtered platform response.

The raw data has an extremely high noise floor, and it makes it incredibly hard to decipher meaningful information from the unfiltered IMU data. To attempt to decipher the actual response, we can smooth the data with minimal overhead by using a moving average. We found that a span of 21 was a good compromise between the quality of output data and the phase delay induced by using a moving average. The data still has noise, but the magnitude of peaks and valleys more closely matches what we expected from the bicycle model. The number and order of peaks and valleys does not perfectly match the bicycle model, but they do align with the differences we saw in the real and model yaw response. The peaks are shifted about a quarter

second behind where they should be, which makes sense with the moving average span of 21. The sample frequency of the IMU is 50Hz in this data, meaning that the span covers 0.42 seconds. In hopes of better understanding the noise floor, we can take the Fast-Fourier Transform (FFT) of the raw IMU data, shown in Figure 78.

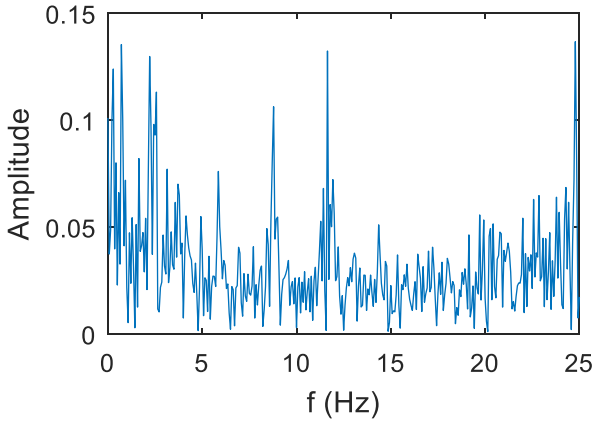


Figure 78: FFT of the raw IMU lateral acceleration data.

The IMU can only sample at 100Hz, and we were only sampling at 50Hz, so we are unable to characterize higher frequency vibrational modes. Within the range of frequencies that we can analyze, we can see that there are vibrational modes at approximately 3, 6, 9, and 12Hz, but none of those modes have overwhelming amplitudes. These modes may also be a result of aliasing, where we are sampling higher-frequency vibration modes at a lower rate, showing false modes. The unfiltered data will be useless for trying to complete stability control. A moving average filter could be applied on the Teensy, but the output data still has non-negligible noise and inherently has a phase shift. Further discussion on how to get more appropriate acceleration data is present in section 8.4, Possible Improvements.

8.4 POSSIBLE IMPROVEMENTS

8.4.1 Geometry & Structure

The build quality evaluation provided us with insight into what areas would require modifications for future adaptations of the vehicle. Although these will be discussed in more detail in later sections a few to note are:

- Installing 1/7th scale wheels that mimic realistic tire dynamics
- Increase the wheelbase for an aspect ratio that compares to a standard SUV (1.7 for Ford Edge)
- Perform tire characterization tests for cornering coefficients
- Install end caps on A-Arm dowels to prevent them from sliding out
- Lock/clamp the battery in place
- Mount IMU closer to CG (below chassis)
- Manufacture or purchase a protective cage for the electronics
- Test clamping capabilities of 3D printed shaft couplers

8.4.2 Circuit Design

With minor modifications, this revision of the circuit board worked nominally. Specifically, the LED footprint had to be modified and augmented, and the 3.3V line between the Pi and Teensy was severed. The Teensy to Pi GPIO pin was also modified, as the pin connected on the Pi is not a valid GPIO pin.

Design wise, we would have liked to have selected dimmer LEDs for both power indication and state indication, or increased the size of the current limiting resistors. We found that we only needed one screw terminal breaking out the CAN bus, but that we would have liked to have 4 screw terminals breaking out the high voltage power and ground for the motor drivers. The screw terminals only easily accept one wire, and trying to route four power cables into two ports was unfruitful. To remedy this in our implementation, we created y-splice cables joining two power cables into one wire to connect to the screw terminal.

8.4.3 Sensors & System Architecture

The serial communication link between the Raspberry Pi and the Teensy microcontroller was the right choice for our application – it helps create a 'black box' that handles the hardware interface. It does, however come with some limitations. Our serial port operates at 115200 bits per second, which means that we can only pass 14.4 kilobytes per second. With the control loop operating at 50 Hz, we can only pass 288 bytes per loop. In our simple demonstration implementation, we are passing 22 bytes per control loop, meaning that we have already consumed 7.5% of the maximum load. The serial communication also does consume clock cycles on both the Pi and the Teensy, introducing further delays on both ends of the communication scheme. While this is manageable at the bus load we are at, implementation of more data-heavy sensors such as LIDAR may require development of Raspberry Pi driver blocks so that the sensors are driven directly off of the Raspberry Pi GPIO. In the future, it may be desirable to develop an SPI interface as it can operate significantly faster than the standard serial interface, but we had significant difficulty implementing it and had to proceed, so serial was the right choice for us. have In the scope of what we were trying to accomplish, the Serial communication allows for easier, faster development and integration of new sensors. It does, however come with limitations when hoping to implement more data throughput.

The interpreted function blocks cannot deploy to the Raspberry Pi hardware, but these can be recreated using S-Function blocks or Device Driver blocks which can deploy to the hardware. It was quite an unfortunate scenario when we tried to embed the Simulink model and have our whole platform running remotely only to have Simulink be unable to do so. Earlier in our development process we had gotten Simulink running great with the Raspberry Pi's hardware and had been writing and reading from the Serial pins. There was no reason to believe a model that could use the Raspberry Pi as effectively as it had been doing could not just embed the same code onto the Raspberry Pi itself.

When we went to select the *External* option and hit *Run* it wouldn't build. We weren't sure what had happened. We looked online for similar errors only to find out it wasn't possible to use the Interpreted MATLAB Functions in an embedded situation. It turns out that the MATLAB functions that are so easy and nice to use don't work when embedded. The only solution is to use S-Function blocks or Device Driver blocks which require a whole process of writing C and C++ code with specific MATLAB files that are used in their specialized process of embedding code. Needless to say there is not much documentation on it, and the ones that we have found have problems we don't have time to fix. If we had, say, two more weeks to

work on this project I am confident that we could implement a solution but time is not something we have anymore.

Additionally, the IMU acceleration data should be processed to filter the noise out of the system as best as possible. This can be implemented in multiple ways, including some configurations of the BNO055 IMU itself. This includes calibrating the IMU, pulling the calibration register data, and pushing the calibration data back to the IMU upon every power-up. For the highest possible accuracy, this should be done for each sensor on the BNO055. The IMU doesn't have any non-volatile memory, so it is critical to pull this data, store it, and push it every time the IMU is powered on. More information on how to calibrate the sensor can be found in the BNO055 datasheet, the first page of which is present in the Drawings section. On the Teensy, several different data filtering methods can be used, the simplest of which would be a low-pass filter. This can be done with a moving average, where the Teensy averages the last *n* readings from the IMU. As *n* increases, the low-pass filter can filter out lower and lower frequency noise. A more accurate way to filter noise would be to develop a system specific Kalman filter, but this is often a project in and of itself. With calibration and a moving average low-pass filter, the lateral acceleration data can be conditioned to be usable.

Currently, our system does not have built-in data logging outside of the repeated profile testing firmware. This can be implemented on the Teensy using the micro-sd card slot, or it can be implemented on the Raspberry Pi. It was not critical for the scope of our project to develop this data-logging, and we ran out of time to implement it as a 'nice to have.' It will be straightforward for someone to implement this in parallel to the development that we have already done. Removeable media is the preferred location for data-logging, but care should be taken that the data-logging does not slow down the response time of the system.

9. REPRODUCTION

Throughout the design and manufacturing phases it was important to keep in mind the reproducibility of individual parts for future SSIVD platforms. Because the vehicle we manufactured consists of parts that are time consuming to reproduce manually, we have identified alternative methods and part designs that would be sufficient for a robust and durable SSIVD.

9.1 MOTHERBOARD

The motherboard was designed in EAGLE cad, which is free for educational and non-commercial use. The board and schematic files are available on the GitHub, along with a full bill of materials. We used a prototype version of the motherboard for our system, but have put up version 1.0 on the GitHub. This version resolves the specific design errors that we had to correct with wire modifications, but does not address the 'nice-to-have' design points mentioned in the Possible Improvements section.

9.2 MECHANICAL SYSTEM

Future vehicle platforms will utilize automated processes to decrease the overall production time. As mentioned in the manufacturing section, many parts were produced via water jet, which is a simple process that only requires two dimensional .dxf files to operate. The majority of the manufacturing time would be spent drilling holes with the drill press and tapping any necessary holes. If made through CNC, the couplers will require a significant amount of time, approximately 4 pairs per hour after set up. However, the results of our FEM indicate that 3D printing the couplers is a feasible solution if they can generate the required clamping force. The other manual operation that would need to be completed is tapping and drilling the steering posts. This would be a quick operation that can be done on the lathe once the stock is cut to length. Tight tolerance rod would eliminate the need to turn the rod down to the 5mm diameter.

Table 13. Custom parts and suggested manufacturing methods.

Part/Operation	Mfg Method
Chassis Net Shape Profile Chassis Mounts Front Suspension Mount Profile Rear Suspension Mount Profile Battery Plate	Water Jet
Chassis Mounting Holes Front Suspension Mount Holes Rear Suspension Mount Holes	Drill Press
Steering Posts	Lathe
Motor Housings Steering Linkage A-Arm Mounts Rear Turnbuckle Mounts Front Turnbuckle Mounts Bumper Mounts IMU Mount URF + Camera Mount	3D Print
Shaft Coupler	3D Print/CNC

9.3 OVERALL SYSTEM COST

We estimate that any future systems will cost approximately $1,050. Depending on manufacturing methods and costs, as well as product sourcing, this value could fluctuate significantly. The Traxxas estimates are based upon retail costs, so we highly suggest looking for third party sources that sell parts at a discounted process. The rough breakdown for future SSIVD costs can be seen below in Table 14. This cost does not include the cost of motors, and uses only an estimate for the electronics.

Table 14. Reproduction costs for future SSIVD platforms.

Total Cost	
Traxxas Parts	$475.75
Electronics	$450.00
Raw Materials and Hardware	$125.84
Total	$1,051.59

10. REFERENCES

[1] D. Walsh, "Connected cars: AA test has effort shifting to higher gear," Crain's Detroit Business, 30 May 2014. [Online]. [Accessed 22 October 2016].

[2] National Center for Statistics and Analysis. , "Pedestrians: 2013 Data," National Highway Traffic Safety Adminsistration, Washington DC, 2015, February.

[3] L. Guo, L. Li, Y. Zhao and Z. Zhao, "Pedestrian Tracking based on Camshift with Kalman Prediction for Autonomous Vehicles," *International Journal of Advanced Robotic Systems,* vol. I, no. 13, pp. 1-9, 2016.

[4] S. Munder, C. Schnorr and D. Gavrila, "Pedestrian Detection and Tracking Using a Mixture of View-Based Shape-Texture Models," CVGPR Group, Amsterdam, The Netherlands, 2007.

[5] US Software System Safety Working Group, "Adaptive Cruise Control System Overview," Anaheim, CA, 2005.

[6] J. Nilsson, A. C. Odblow and J. Fredriksson, "Worst-Case Analysis of Automotive Collision Avoidance Systems," *IEEE Transactions on Vehicular Technology,* vol. 65, no. 4, pp. 1899-1911, 2016.

[7] T. Stevens, "A LIDAR BASED SEMI-AUTONOMOUS COLLISION AVOIDANCE SYSTEM," California Polytechnic University, San Luis Obispo, CA, 2015.

[8] S. Lefevre, D. Vasquez and C. Laugier, "A survey on motion prediction and riskassessment for intelligent vehicles," *ROBOMECh Journal,* vol. 1, no. 1, pp. 1-14, 2014.

[9] SAE International, "Automotive Stability Enhancement Systems," *Surface Vehicle Information Report,* vol. J2564, 2004.

[10] Bavarian Motor Works, "Dynamic Traction Control," BMW, 2016. [Online]. Available: http://www.bmw.com/en/technology/insights/dynamic_traction_control. [Accessed 21 October 2016].

[11] H. Lee and M. Tomizuka, "Adaptive Vehicle Traction Force Control for Intelligent Vehicle Highway Systems (IVHSs)," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS,* vol. 50, no. 1, pp. 37-47, February 2003.

[12] Insurance Institute for Highway Safety, "ESC and side airbag availability by make and model," Highway Loss Institute, 2016. [Online]. Available: http://www.iihs.org/iihs/ratings/safety-features. [Accessed 6 November 2016].

[13] California State University, Northridge, "2016-2017 Catalog," Mechanical Engineerign Department, CSU Northridge, 2016. [Online]. [Accessed 20 October 2016].

[14] S. Solmaz and T. Coskun, "An automotive vehicle dynamics prototyping platform based on a remote control," *Turkish Journal of Electrical Engineering & Computer Sciences,* vol. I, no. 21, pp. 439-451, 2013.

[15] H. Manh La, R. Lim, J. Du, S. Zhang, G. Yan and W. Sheng, "Development of a Small-Scale Research Platformfor Intelligent Transportation Systems," *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, ,* vol. 13, no. 4, pp. 1753-1762, 2012.

[16] A. Fabio, "Independent Wheel Drive R/C Car," Hackaday, 2014. [Online]. Available: http://hackaday.com/2014/07/12/independent-wheel-drive-rc-car/. [Accessed 15 11 2016].

[17] "Arduino Uno," Arduino, [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed 23 10 2016].

[18] "Arduino MEGA 2560," Ardunio, [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardMega2560. [Accessed 23 10 2016].

[19] J. Man, "Raspberry Pi 3 Model B Technical Specifications," Element14, [Online]. Available: https://www.element14.com/community/docs/DOC-80899/l/raspberry-pi-3-model-b-technical-specifications. [Accessed 23 10 2016].

[20] "Raspberry Pi Model A+," Adafruit, [Online]. Available: https://www.adafruit.com/products/2266. [Accessed 23 10 2016].

[21] "BeagleBone Black," BeagleBone, [Online]. Available: https://beagleboard.org/black. [Accessed 23 10 2016].

[22] "Hardware Support Index," Mathworks, [Online]. Available: https://www.mathworks.com/hardware-support/index.html. [Accessed 23 10 2016].

[23] K. Amano, N. Goda, S. Nishida, Y. Ejima, T. Takeda and Y. Ohtani, "Estimation of the Timing of Human Visual Perception," *The Journal of Neuroscience,* p. 3981–3991, 2006.

[24] Department of Mechanical Engineering, Cal Poly, "ME 428/429/430 Senior Design Project Reference Book and Success Guide," 2016.

[25] Traxxas, "Traxxas Slash: 1/10 scale short-course truck," Traxxas, 2016. [Online]. [Accessed 9 November 2016].

[26] Maxon Motor, "EC 45 flat Ø42.8 mm, brushless, 70 Watt, with Hall sensors," 2016. [Online]. Available: http://www.maxonmotor.com/maxon/view/product/397172.

[27] Ford Motor Company, "2015 Ford Edge Technical Specifications," 2015. [Online]. Available: https://media.ford.com/content/dam/fordmedia/North%20America/US/2015_Specs/2015_Edge_Specs.pdf.

[28] Maxon Motor, "EPOS4 Compact 50/5 CAN Hardware Reference," November 2016. [Online]. Available: http://www.maxonmotorusa.com/medias/sys_master/root/8823917740062/534130-Hardware-Reference-En.pdf. [Accessed 12 January 2017].

[29] Texas Instruments, "Overview of 3.3V CAN (Controller Area Network) Transceivers," January 2013. [Online]. Available: http://www.ti.com/lit/an/slla337/slla337.pdf. [Accessed 27 January 2017].

[30] R. Rajamani, Vehcile Dynamics and Control, New York, NY: Springer Science and Business Media, 2006.

[31] CarSort, "Ford Edge SEL vs. Ford Explorer," 2015. [Online]. Available: http://carsort.com/compare/Ford-Edge-vs-Ford-Explorer.

[32] W. Riley and A. George, "Design, Analysis and Testing of a Formula SAE Car Chassis," *Proceedings of the 2002 SAE Motorsports Engineering Conference and Exhibition,* vol. 01, p. 382, January 2002.

[33] E. W. Weisstein, "Torsional Rigidity," From MathWorld--A Wolfram Web Resource, 2010. [Online]. Available: http://mathworld.wolfram.com/TorsionalRigidity.html. [Accessed 16 January 2016].

[34] R. Budynas and K. Nisbett, Shigley's Mechanical Engineering Design, New York, NY: McGraw Hill, 2016.

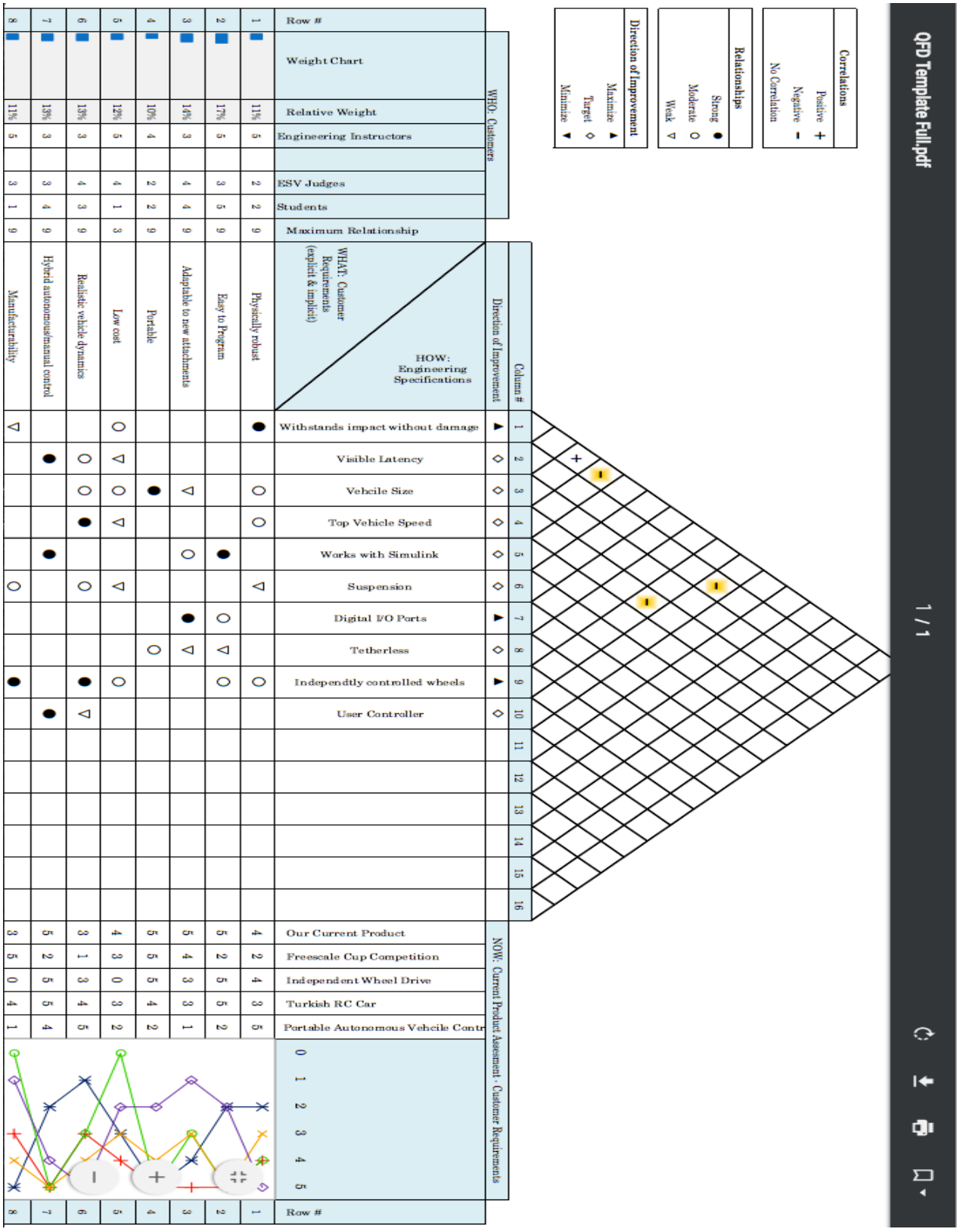
[35] B. Quimby, "Chapter 3 - Tension Members - Bearing on Holes," BG Structural Engineering, 27 July 2011. [Online]. Available:

http://www.bgstructuralengineering.com/BGSCM13/BGSCM003/BGSCM00306.htm. [Accessed 29 January 2017].

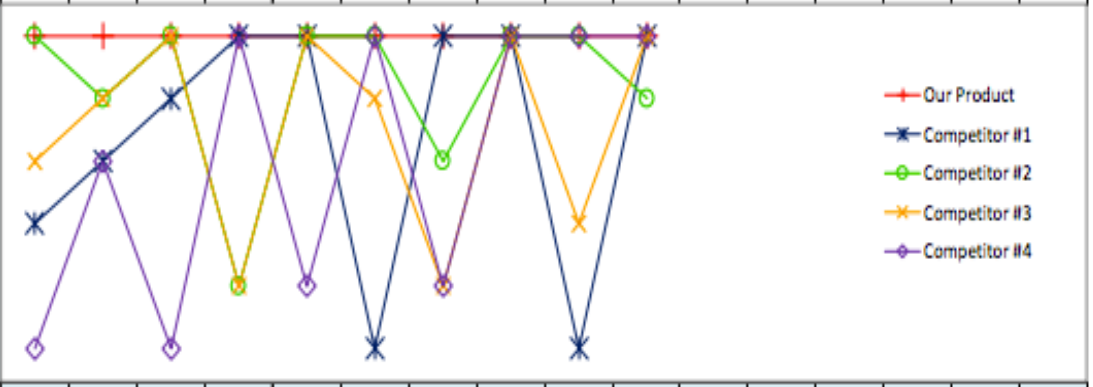
[36] NHTSA, "ESV 2017," 2016. [Online]. Available: http://www-esv.nhtsa.dot.gov/. [Accessed 22 10 2016].

APPENDICES

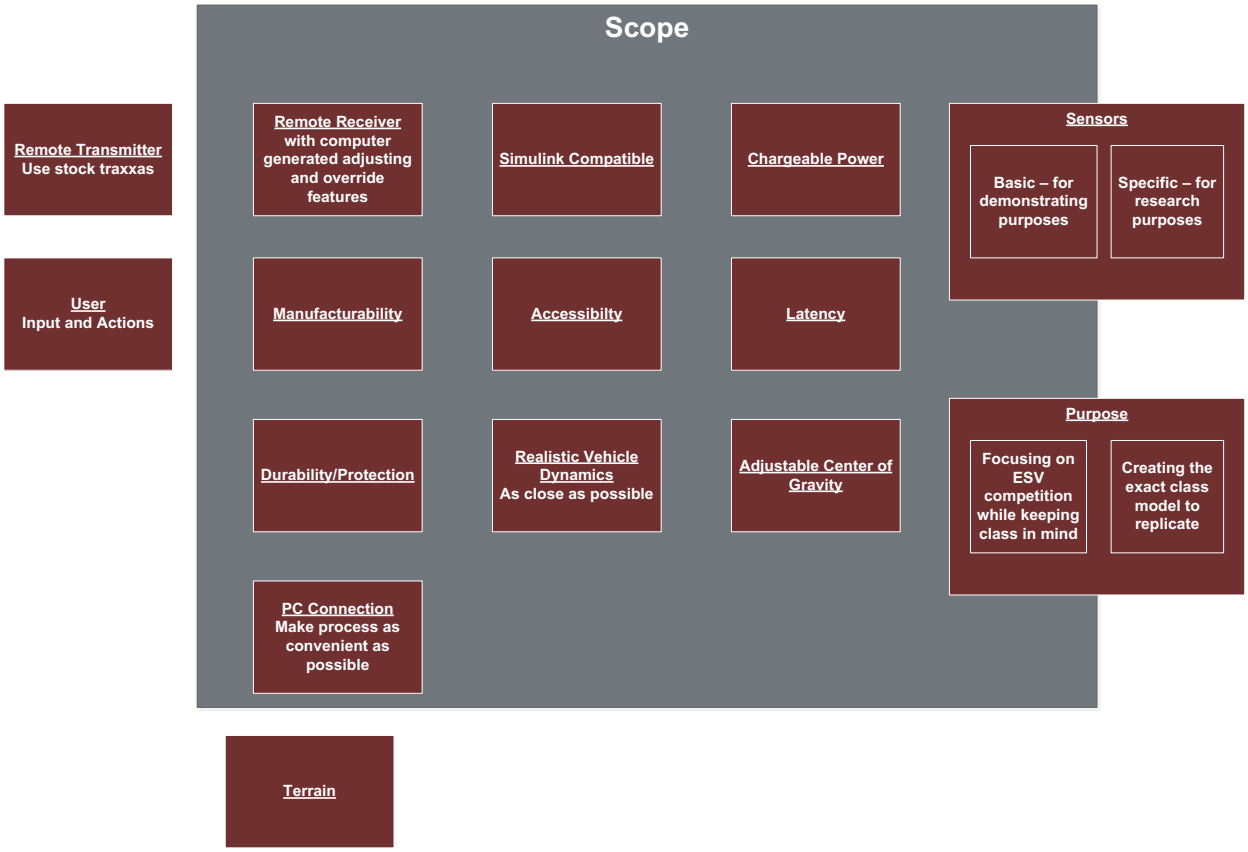
APPENDIX A QUALITY FUNCTION DEPLOYMENT



	HOW MUCH: Target	20 MPH impact into cinder block	None	1/10th Scale	15 MPH	Yes	Yes	>3 modules	Yes	>2	Yes						
	Max Relationship	9	9	9	9	9	3	9	3	9	9						
	Technical Importance Rating	142.99	168.44	208.73	160.32	310.73	93.071	179.7	60.263	329.09	130.73						
	Relative Weight	8%	9%	12%	9%	17%	5%	10%	3%	18%	7%						
	Weight Chart																
Current Product Assessment - Engineering Specifications	Our Product	5	5	5	5	5	5	5	5	5	5						
	Freescale cup competition	2	3	4	5	5	0	5	5	0	5						
	Independent Wheel Drive	5	4	5	1	5	5	3	5	5	4						
	Turkish RC Car	3	4	5	1	5	4	1	5	2	5						
	Portable Autonomous Vehicle	0	3	0	5	1	5	1	5	5	5						
		0 1 2 3 4 5															
	Column #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



APPENDIX B BOUNDARY SKETCH

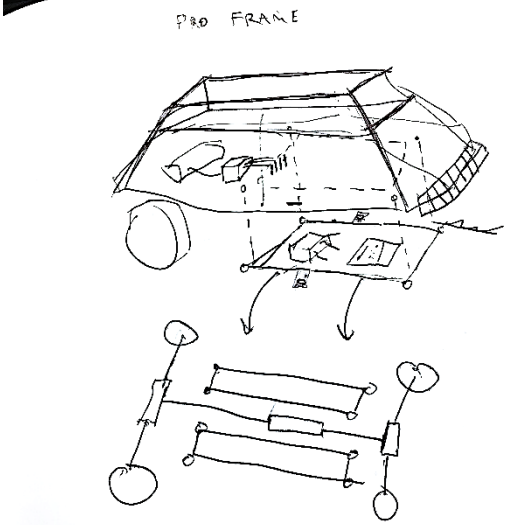


APPENDIX C BRAINSTORMNG TABLES AND FIGURES

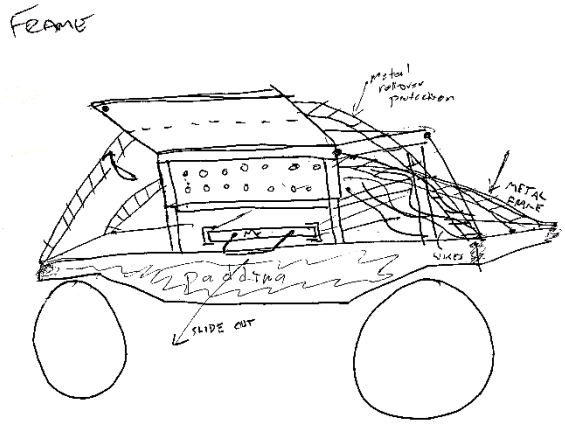
Morph Chart

	Function					
Potential Options	**Electronics Protection**	**Protective Housing**	**Types of Sensors**	**Dynamic CG**	**Algortithms**	**Braking systems**
	Lid system	Cushion Bumper System	Microphone / sounds	Adjustable Mast	Adaptive cruise control	Friction clutch
	Wire Cage	Protective sides(go-kart)	IMU	Roll Bar w/ weights	Stop light detection	Hydraulic brakes
	Plexiglass / acrylic housing	Roll cage	Radar	Dynamic Angle Boom	Line-following	Shaft braking
	Custom PCB	Mechanical fuse	Wheel Encoder	Tub of water on top	Following	Tiny Disc Brakes
	Cast MCU in epoxy	Crumple zone	LIDAR	Battery in middle	Obstacle avoidance	Split diff, two motors
		Inflatable bumber	Pressure transducer tires	Placeable weights	Lane tracking	4 wheel steering
		Airbag	Road condition(friction)	weight rack	Dynamic stability control	Friction solenoid
		Deployable bumpers	Tachometer		Road sign deteection/reading	
		Foam Bumpers	Hall Sensor		Pedestrian Detection	
		Vaccum Formed Shell	Digital Camera		Outside Camera system integration	
		Zip tie mounting points			Parking auto-pilot	
		Detachable middle component			Controller feedback	
		Central breadbox design			Light detection	
		Box with ports			Vehicle to vehicle coms	
					Collision avoidance	
					Stop-light cam	
					Adaptive speed limit	

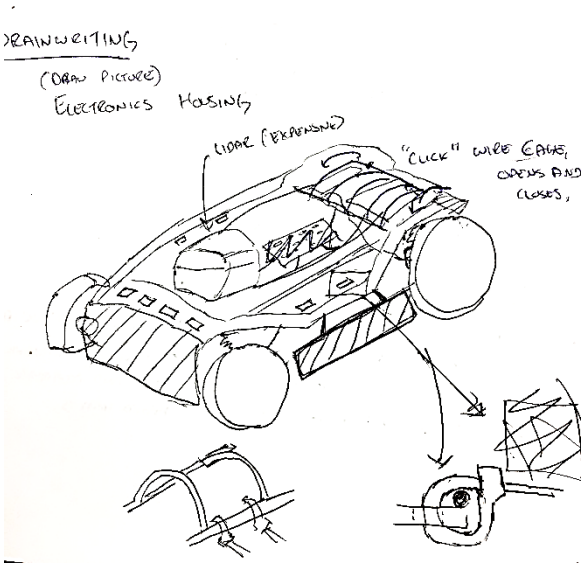
Brainstorming Examples



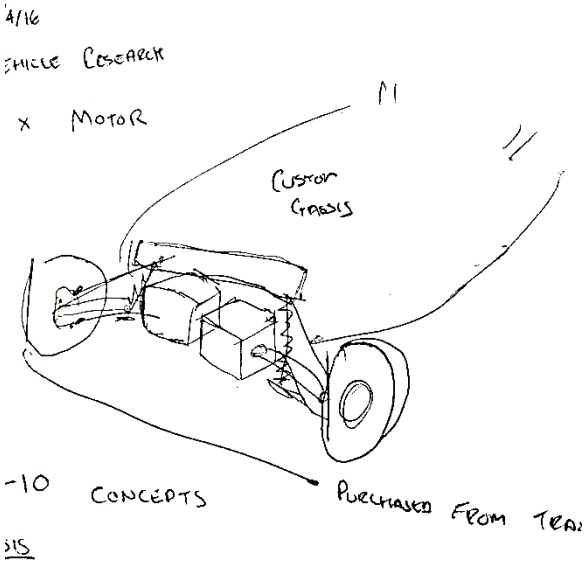
Drop out electronics board



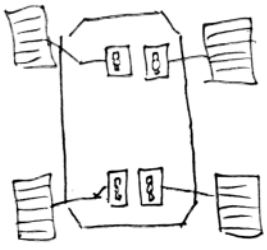
Frame and easy access sensor plug-ins



Overall concept ideation

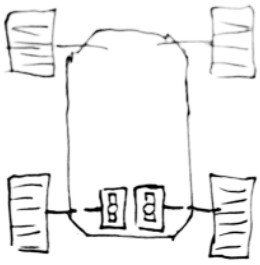


Dual Motor System w/ custom chassis



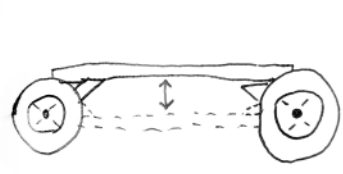
4 motor power system

Friction solenoid



2 motor power system

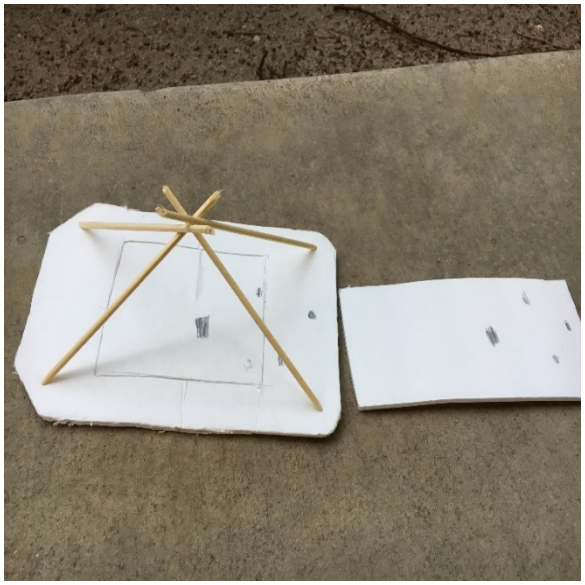
Rack with weights



Adjustable chassis height

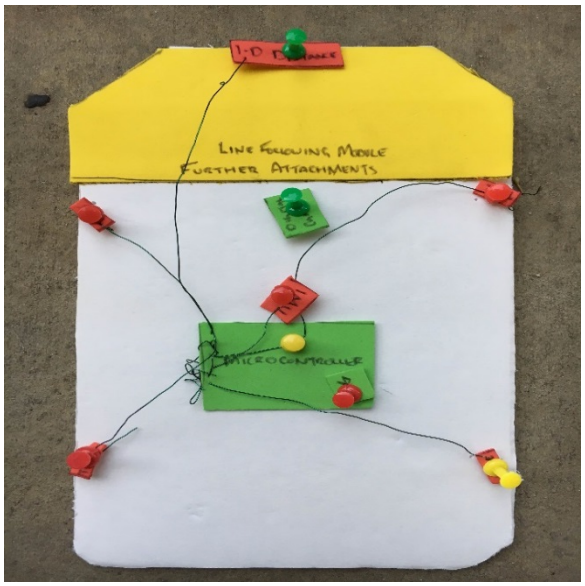
\

Concept Prototypes

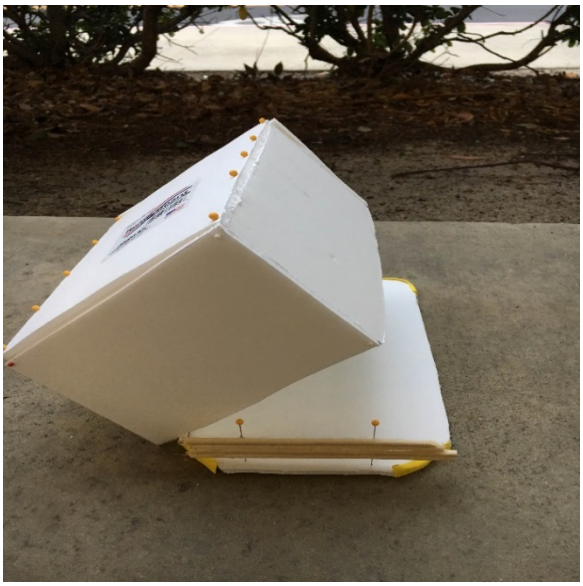


Electrical Layout – Slide in-out board with electrical connections on the bottom of the attachment (slide out piece shown upside down).

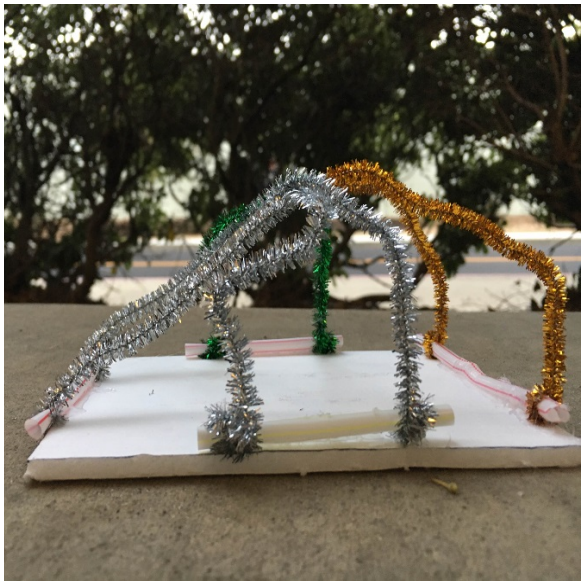
Electrical Layout – Flip board which allows the electrical components to be accessible from the bottom of the vehicle.



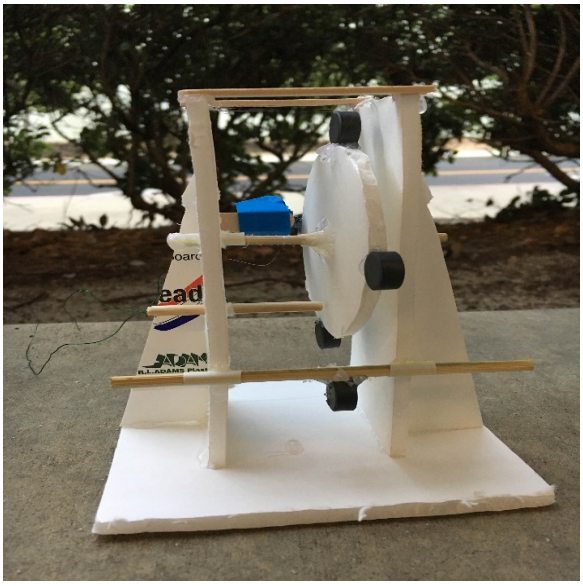
Electrical Layout – An example of the possible components we could use and the arrangement they could have. Included is a Microcontroller, multiple distance sensors, a Bluetooth sensor, an IMU, and prepared space for more sensors.



Protective Housing – This features a hinged cage design which allows the components to be covered and protective while also being readily available for human modification.



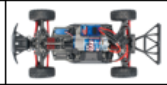
Protective Housing – A possible solution using a wired cage design where the pipe cleaners represent metal bars.



Braking System – This prototype features three different possible methods. The blue tape in the back of the picture holds an experimental eddy current system with thin copper wire wrapped approximately 100 turns that connected to a 9V battery. The next experimental design was magnetic braking which we glued a magnet to the stick in the front of the design and used the attractive magnetic forces to slow down the wheel. The last trial method is a solenoid method which is an applied-friction design and is shown here as the bottom stick in the middle of the design being applied to the disc.

APPENDIX D PUGH AND DECISION MATRICES

Function 1: RC Platform

	Concept					
	Traxxis Slash	Axial Yeti	Traxxis E-Revo 1/16th	Axial SCX10	HPI Trophy Flux	1/16th Slash 4WD
						
Criteria	**1**	**2**	**3**	**4**	**5**	**6**
1/10th scale	+	+		+	+	0
Durable Components	0	-	D	-	+	0
Parts Database	0	0		0	-	0
Cost	+	-	A	-	-	+
Space for Components	+	0		+	+	0
Suspension	0	0	T	0	0	0
Rubber Tires	0	0		0	0	0
Weight (Heavy=+)	+	+	U	+	+	0
∑+	4	2		3	4	1
∑-	0	-2	M	-2	-2	0
∑s	4	0		1	2	1

Function 2: Microcontroller

	Concepts		
Criteria	**1**	**2**	**3**
CPU Speed	*+	*+	D
RAM	*+	*+	A
ROM	*+	*+	T
Ports	*-	*0	U
Cost	*+	*-	M
∑+	4	3	
∑-	1	1	
∑s	3	2	

Function 3: Algorithms

	Concept							
	Adaptive Cruise	Dynamic Stability	Lane Following	Pedestrian Detection	Road Sign / Light detection	Auto Parking	Multi Vehicle Management	Obstacle Detection
Criteria	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Additional Sensors Reqd (more is bad, baseline IMU / Tach)		+	-	=	=	-	-	=
Practicality		-	=	-	-	-	-	-
Intrigue	D	+	=	+	+	=	+	+
Student Challenge	A	+	=	+	+	=	+	+
Physical Simplicity	T	-	=	=	=	+	+	+
Wiring Simplicity	U	=	-	=	=	-	=	=
∑+	M	3	0	2	2	1	3	3
∑-		-2	-2	1	-1	-3	-2	-1
∑s		1	-2	-1	-1	-2	1	2

Function 4: Braking Method

	Friction solenoid	Eddy Current	Disc Brakes	4x Motors	Electromagnets	Permanent Magnets
Criteria	**1**	**2**	**3**	**4**	**5**	**6**
Braking Force	-	-		+	-	-
Feasability	+	0	D	+	0	0
Cost of implementation	+	0	A	0	0	0
Complexity of redesign	+	0	T	0	0	0
∑+	3	0	U	2	0	0
∑-	-1	-1	M	0	-1	-1
∑s	2	-1		2	-1	-1

Function 5: Protective Housing

	Concept			
	Wire Cage	Foam Bumpers	Box-Hinge	Drop Out Assembly
Criteria	**1**	**2**	**3**	**4**
Ease of Implentation	0		-	-
Structural Rigidity	+	D	+	+
Feasability	0	A	0	-
Cost of implementation	0	T	0	0
Complexity	0	U	0	-
∑+	1	M	1	1
∑-	0		-1	-3
∑s	1		0	-2

Function 6: CG Modification

	Concept	
	Rack System (Static)	Adjustable Column(Dynamic)
Criteria	**1**	**2**
Effectiveness		+
Feasability	D	-
Cost of implementation	A	0
Complexity	T	-
∑+	U	1
∑-	M	-2
∑s		-1

Function 7: Electrical Layout

	Concept			
	Breadboard	Mother / Daughterboard	Discrete Components	Component Stack
Criteria	**1**	**2**	**3**	**4**
Adaptability to new senso	+	+		-
Cost	-	-		-
Ease of manufacture	+	-	D	-
Durability	-	+	A	+
Accessibility	+	+	T	-
Self Contained	-	+	U	+
∑+	3	4	M	2
∑-	-3	-2		-4
∑s	0	2		-2

	Factor 1		Factor 3		Factor 5		Factor 6		Factor 8		Factor 8			
	Physically Robust		Adaptable		Cost		Realistic Dynamics		Manufacturability		Repairability			
	Weight	20%	Weight	15%	Weight	15%	Weight	10%	Weight	20%	Weight	20%	Sum wt.	100%
System	Rank	Wtd. Rank	Rank	Wtd. Rank	Rank	Wtd. Rank	Rank	Wtd. Rank	Rank	Wtd. Rank	Rank	Wtd. Rank	Total	Wtd Total
1	3	0.6	3	0.45	5	0.75	3	0.3	3	0.6	2	0.4	19	3.1
2	5	1	5	1	2.5	0.5	4	0.8	5	1	5	1	26.5	5.3
3	5	1	4	0.8	1	0.2	4	0.8	3	0.6	4	0.8	21	4.2
4	5	1	5	1	1	0.2	5	1	3.5	0.7	4	0.8	23.5	4.7
5	4	0.8	4	0.8	4	0.8	2	0.4	2	0.4	2.5	0.5	18.5	3.7
6	4	0.8	3	0.6	2.5	0.5	4	0.8	3.5	0.7	3.5	0.7	20.5	4.1
7	5	1	5	1	1	0.2	5	1	4	0.8	4	0.8	24	4.8
8	4	0.8	4	0.8	4	0.8	2	0.4	2	0.4	2.5	0.5	18.5	3.7
Notes:	**Physically Robust** refers to how well protected the electonics are and how well the vehicle can withstand impacts. Rank out of 5.		**Adaptable** means that new sensors and modifications can be easily added to the vehicle. Rank out of 5.		The **cost** refers to the overall price to develop a single unit. Rank out of 5.		A vehicle with **realistic dynamics** has systems and properties similar to a full scale vehicle.		**Manufacturability** refers to how easily the prototype can be reproduced.		**Repairabilty** refers to how easily broken components can be fixed or replaced.			

Decision Matrix used to aid final system design decision.

APPENDIX E DESIGN SAFETY CHECKLIST

ME 428/429/430 Senior Design Project 2016-2017

DESIGN HAZARD CHECKLIST

Team: 46 - uLaren Advisor: Dr. Birdsong

Y	N	
☑	☐	1. Will any part of the design create hazardous revolving, reciprocating, running, shearing, punching, pressing, squeezing, drawing, cutting, rolling, mixing or similar action, including pinch points and sheer points?
☑	☐	2. Can any part of the design undergo high accelerations/decelerations?
☑	☐	3. Will the system have any large moving masses or large forces?
☐	☑	4. Will the system produce a projectile?
☐	☑	5. Would it be possible for the system to fall under gravity creating injury?
☐	☑	6. Will a user be exposed to overhanging weights as part of the design?
☐	☑	7. Will the system have any sharp edges?
☐	☑	8. Will any part of the electrical systems not be grounded?
☑	☐	9. Will there be any large batteries or electrical voltage in the system above 40 V?
☑	☐	10. Will there be any stored energy in the system such as batteries, flywheels, hanging weights or pressurized fluids?
☐	☑	11. Will there be any explosive or flammable liquids, gases, or dust fuel as part of the system?
☐	☑	12. Will the user of the design be required to exert any abnormal effort or physical posture during the use of the design?
☐	☑	13. Will there be any materials known to be hazardous to humans involved in either the design or the manufacturing of the design?
☐	☑	14. Can the system generate high levels of noise?
☐	☑	15. Will the device/system be exposed to extreme environmental conditions such as fog, humidity, cold, high temperatures, etc?
☑	☐	16. Is it possible for the system to be used in an unsafe manner?
☐	☑	17. Will there be any other potential hazards not listed above? If yes, please explain on reverse.

For any "Y" responses, add (1) a complete description, (2) a list of corrective actions to be taken, and (3) date to be completed on the reverse side.

Figure 4: Design Hazard Checklist, Page 1

Description of Hazard	Planned Corrective Action	Planned Date	Actual Date
Design contains revolving action at wheels and potential pinch / shear points in structure	o Affix warning to car to disconnect battery power while programming or maintaining, as forced turning or movement or turning only occurs under battery power	5/10/17	5/10
Design can undergo high acceleration due to powerful motors	o This is part of intended operation, but we will ensure the software cannot prompt excessive acceleration without user input	4/10/17	5/20
Design will be able to go very fast presenting impact hazard	o Can install default software limitation and lock maximum speed to controller input levels	4/10/17	5/20
There will be a large Lithium Polymer battery in the system, running at 25.9V	o We will purchase a fire resistant bag for storing and charging o We will use a polarized connector to prevent shorting or applying reverse voltage o We will design several possible protective systems to prevent impact or puncture	3/10/17	3/20
It will be possible to use the system in an unsafe manner by driving into people or walls	o While this is out of our control, we will write a user manual with an introduction that introduces specific safety hazards that the user should be aware of o We will consider including a training mode that limits acceleration and speed	4/10/17	5/10

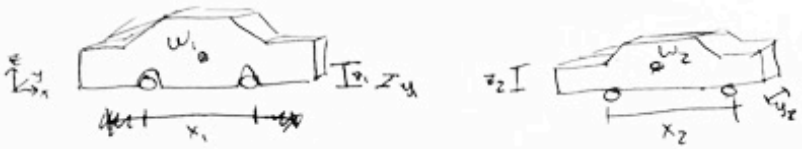
Figure 5: Design Hazard Checklist, Page 2

APPENDIX F SUPPORTING ANALYSIS

Similitude Analysis

SIMILITUDE ANALYSIS

BY DIMENSIONS:



$W_1 = x_1 y_1 z_1 \rho$

$W_2 = x_2 y_2 z_2 \rho$ WHERE $x_2 = \frac{x_1}{10}$, $y_2 = \frac{y_1}{10}$, $z_2 = \frac{z_3}{10}$

$W_2 = \frac{x_1 y_1 z_1}{10 \times 10 \times 10} \rho$

$= \frac{W_1 \rho}{1000}$ SO, WEIGHT SCALES BY A FACTOR OF $\frac{1}{1000}$

BY WEIGHT

$W_2 = \frac{W_1}{10}$

$W_1 = x_1 y_1 z_1 \rho$

$W_2 = x_2 y_2 z_2 \rho$

$x_2 y_2 z_2 \rho = \frac{x_1 y_1 z_1 \rho}{10}$

ASSUMING EACH DIMENSION SCALES BY THE SAME WEIGHT,

$x_2 = x_1 \sqrt[3]{1/10} = x_1 (0.464)$

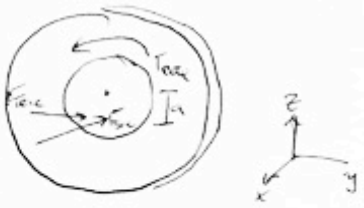
$y_2 = y_1 \sqrt[3]{1/10} = y_1 (0.464)$

$z_2 = z_1 \sqrt[3]{1/10} = z_1 (0.464)$

SO, DIMENSIONS EACH SCALE BY ABOUT 0.464.

Braking Force Analysis

BRAKING FORCE - FRICTION SOLENOID, CONTINUED



$F_{Fric} = \mu F_{Sol}$, ASSUME $\mu = 0.7$

$T_{Req} = F_{Fric} a$ WHERE a IS DISTANCE FROM WHEEL AXEL TO SOLENOID FRICTION PT. $a \approx 0.2''$

$T_{Req} = \mu F_{Sol} a$

$F_{Sol} = \frac{T_{Req}}{\mu a}$

$= \frac{0.076 \text{ lb}_f \cdot \text{ft}}{(0.7)(0.2''/12 \text{ in/ft})}$

$= 6.5 \text{ lb}_f$

* THIS A REASONABLE ACTUATING FORCE FROM SOLENOID

Motor Analysis

Maxon Motor Specifications		
Max Continuous Torque	128	*mNm*
Max Continuous Speed	4860	*rpm*

RC Platform Specifications		
Traxxas Slash 4X4 Mass	2.64	*kg*
Rough Estimate of Added Mass	1.5	*kg*
Estimated Total Mass	4.65	*kg*
Maximum Wheel Diameter	76	*mm*

With the assumed operation being on flat ground, we can calculate the estimated continuous speed and acceleration of our system with the selected motors. Velocity can be calculated based on angular velocity and radius.

$$V = \omega * r$$

Since we have angular velocity in rotations per minute and radius in millimeters, we must use unit conversions to get the ft/s needed to compare to our engineering specifications.

$$V = \omega * \frac{2\pi}{60} \frac{\frac{rad}{s}}{rpm} * r * \frac{1}{10 * 2.54 * 12} \frac{ft}{mm}$$

For 4860 rpm and a wheel diameter of 76mm, we can calculate the max sustained speed to be 63 ft/s or 43.3 mph. This is far above our specified speed, but the motors can be controlled at a lower voltage for slower speeds.

Acceleration can be calculated based on mass and force, which we can calculate as a function of torque.

$$A = \frac{F}{m} = \frac{\frac{T}{r}}{m}$$

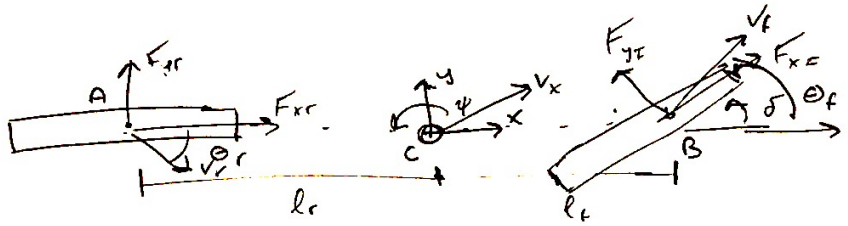
We must convert the final acceleration to ft/s^2 to compare directly with our engineering specifications.

$$A = \frac{\frac{T * \frac{1}{1000} \frac{Nm}{mNm}}{r}}{m} * 3.28 \frac{ft}{m}$$

For four wheels with 128 mNm of torque, a 76mm wheel diameter, and a mass of 4.65kg, we estimate that we will be capable of an acceleration of 9.5 ft/s^2. While this exceeds our specification, we can control the motors at a lower voltage to produce lower torque.

Simulink Steering Model Derivation

STATE SPACE MODEL - DERIVATION



$$\Sigma F_y: \quad m(\dot{y} + \dot{\psi} V_x) = F_{yF} + F_{yR}$$

$$\Sigma M_C: \quad I_z \ddot{\psi} = l_f F_{yF} - l_r F_{yr}$$

FOR SMALL SLIP ANGLES, $F_y \propto C_\alpha$

$$F_{yF} = 2C_{\alpha f}(\delta - \theta_{Vf}) \quad \text{WHERE} \quad \theta_{Vf} = \frac{\dot{y} + l_f \dot{\psi}}{V_x}$$

$$F_{yR} = 2C_{\alpha r}(-\theta_{Vr}) \quad \text{WHERE} \quad \theta_{Vr} = \frac{\dot{y} - l_r \dot{\psi}}{V_x}$$

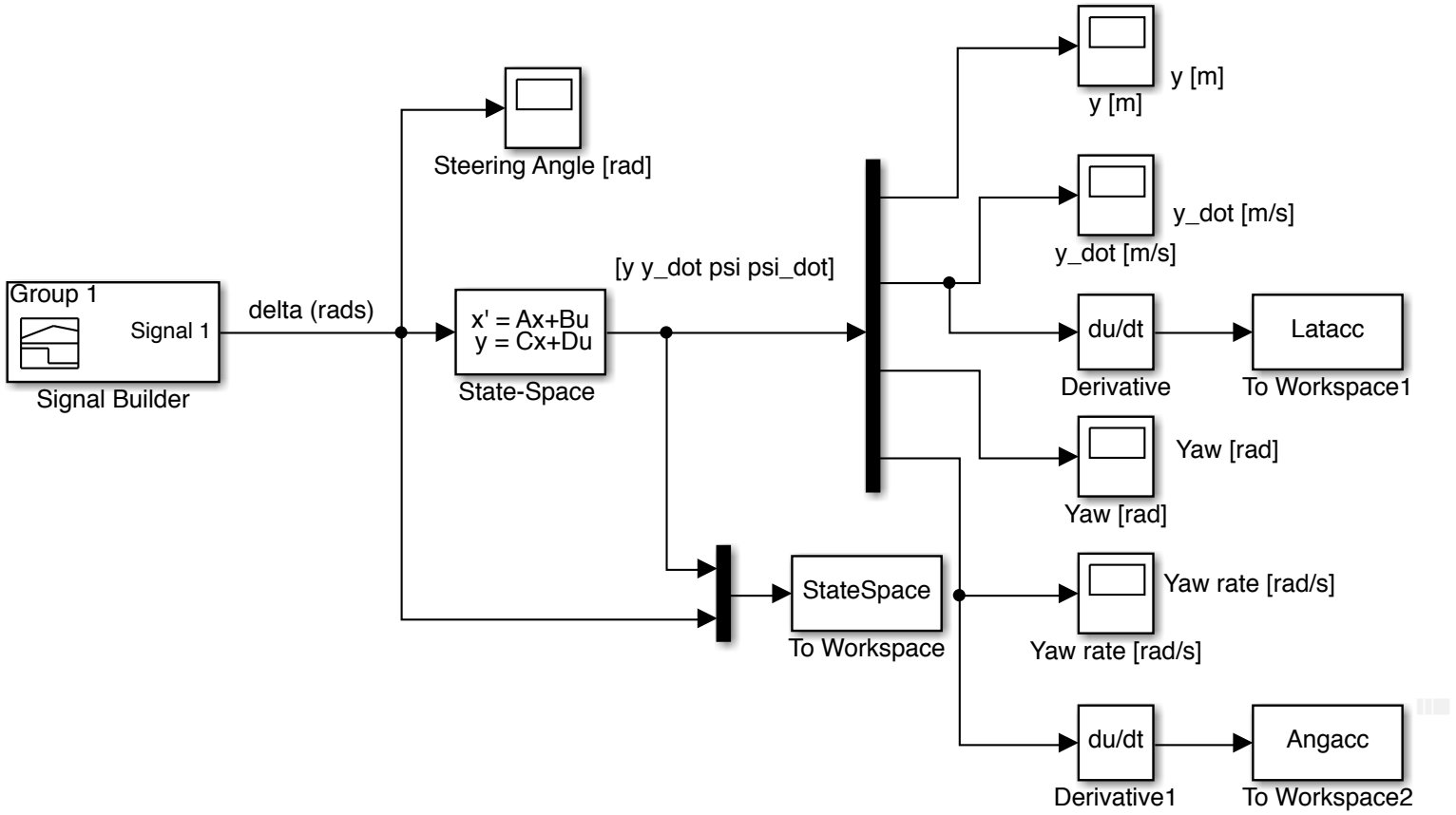
REARRANGING AND SOLVING FOR $\ddot{y}$, $\ddot{\psi}$, RESPECTIVELY

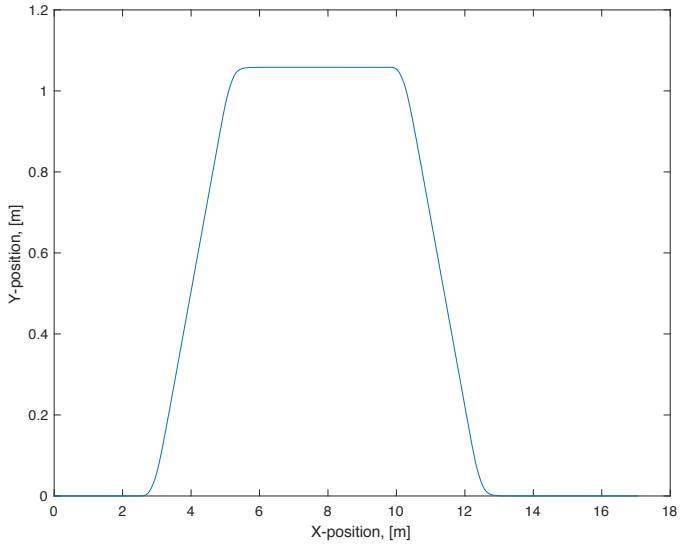
$$\ddot{y} = -\frac{2C_{\alpha f} + 2C_{\alpha r}}{mV_x}\dot{y} - \dot{\psi}\left[V_x + \frac{2C_{\alpha f} l_f - 2C_{\alpha r} l_r}{mV_x}\right] + \frac{2C_{\alpha f}}{m}\delta$$

$$\ddot{\psi} = \frac{-2l_f C_{\alpha f} - 2l_r C_{\alpha r}}{I_z V_x}\dot{y} - \dot{\psi}\frac{2l_f^2 C_{\alpha f} + 2l_r^2 C_{\alpha r}}{I_z V_x} + \frac{2l_f C_{\alpha f}}{I_z}\delta$$

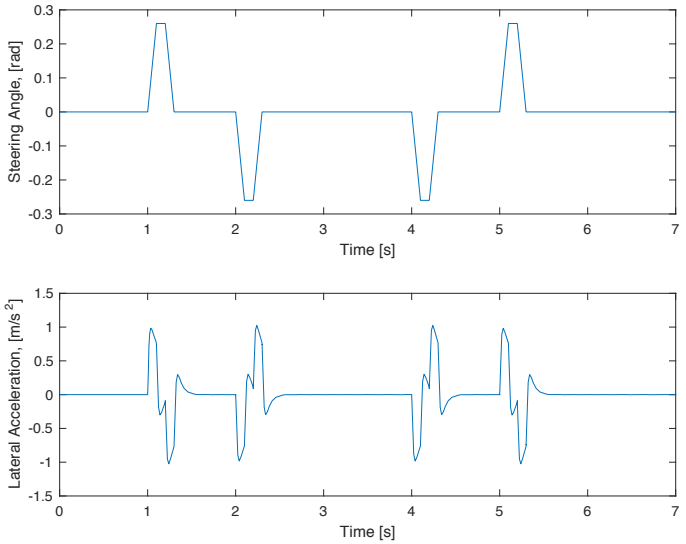
WHICH CAN BE SOLVED FOR AS A SYSTEM OF COUPLED, 1ST ORDER DIFFERENTIAL EQUATIONS

Simulink Steering Model

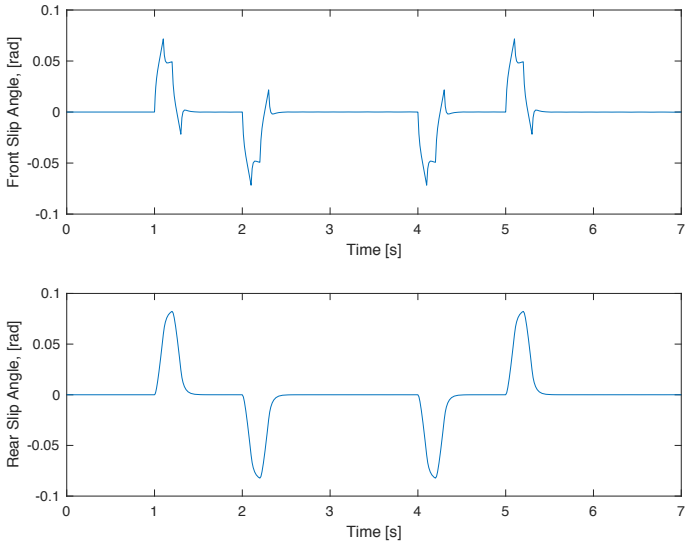




Global path of vehicle

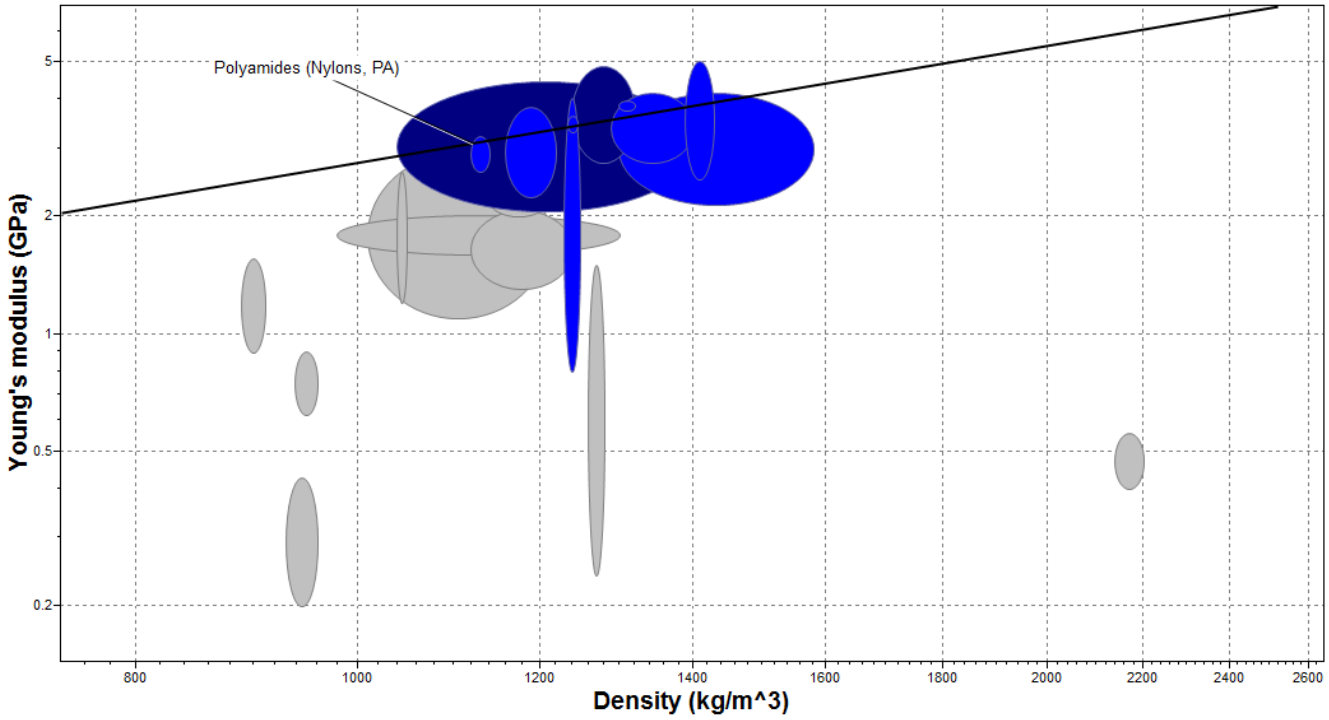


Top: Steering angle input, Bottom: Lateral acceleration

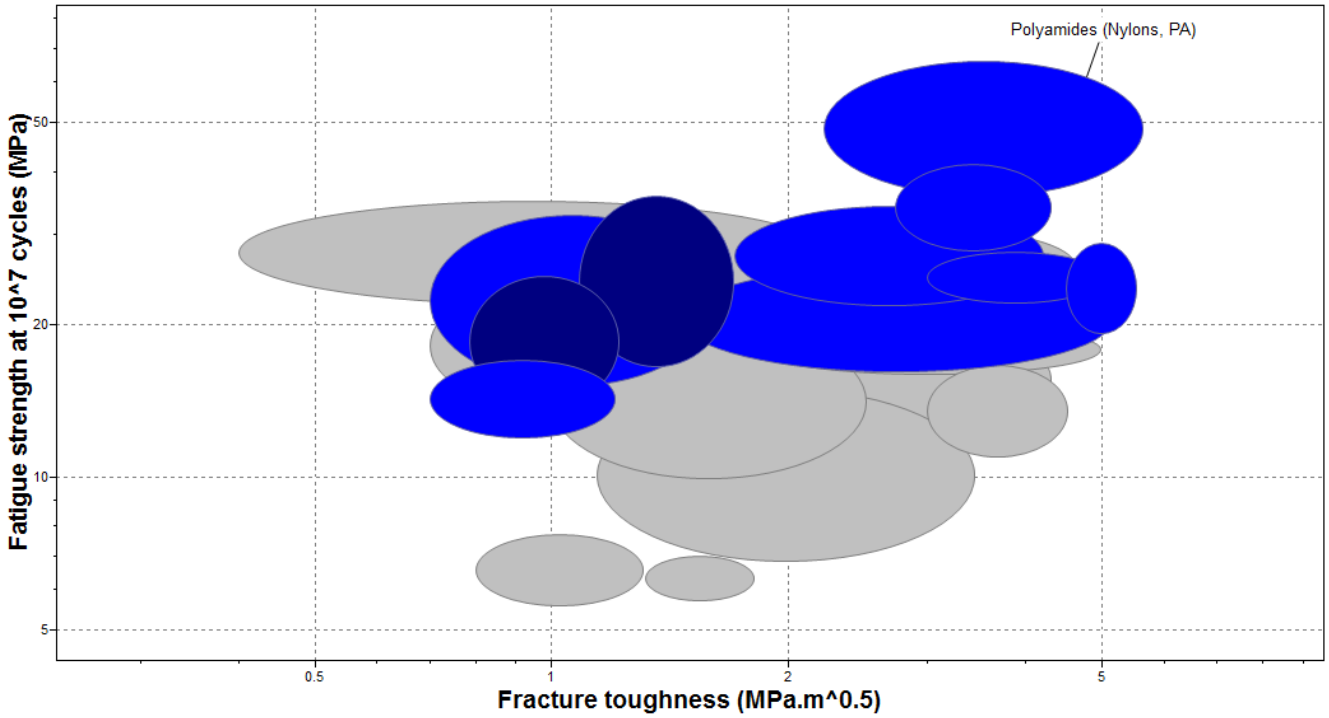


Top: Front slip angle, Bottom: Rear slip angle

Materials Selection – Ashby Charts



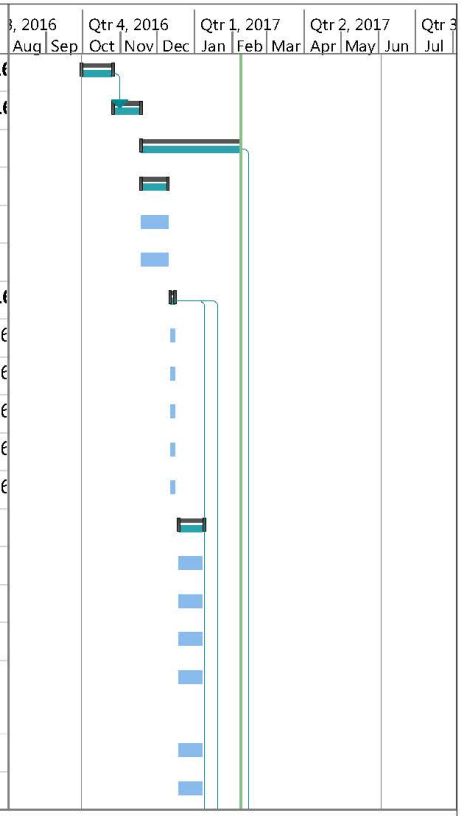
Young's Moduous and density. Ideally top left.



Fatigue strength and fracture toughness. Ideally top right.

APPENDIX G GANTT CHART

ID	Task Mode	Task Name	Duration	Start	Finish
1		**Project Proposal**	**18 days**	**Fri 9/30/16**	**Tue 10/25/16**
7		**Preliminary Design Review**	**17 days**	**Wed 10/26/1**	**Thu 11/17/16**
20		**Critical Design Review**	**58 days**	**Fri 11/18/16**	**Tue 2/7/17**
21		**Safety and Testing Documentation**	**16 days**	**Fri 11/18/16**	**Fri 12/9/16**
22		FMEA	16 days	Fri 11/18/16	Fri 12/9/16
23		DVP&R	16 days	Fri 11/18/16	Fri 12/9/16
24		**End of Fall Quarter Tasks**	**4 days**	**Mon 12/12/1**	**Thu 12/15/16**
25		Basic Solid Modeling	4 days	Mon 12/12/1	Thu 12/15/16
26		Electrical Layout	4 days	Mon 12/12/1	Thu 12/15/16
27		Order Traxxas Slash	4 days	Mon 12/12/1	Thu 12/15/16
28		Order Electronic Components	4 days	Mon 12/12/1	Thu 12/15/16
29		Update Gantt Chart for Next quarter	4 days	Mon 12/12/1	Thu 12/15/16
34		**Winter Break Tasks**	**16 days**	**Mon 12/19/1**	**Sun 1/8/17**
35		Solid Modeling	15 days	Mon 12/19/1	Fri 1/6/17
36		Update Brainstorming	15 days	Mon 12/19/1	Fri 1/6/17
37		Firmware Architecture Design	15 days	Mon 12/19/1	Fri 1/6/17
38		Vehicle Dynamics Research and Analysis	15 days	Mon 12/19/16	Fri 1/6/17
39		Specifications Checklist	15 days	Mon 12/19/1	Fri 1/6/17
40		Shaft Coupler Design	15 days	Mon 12/19/1	Fri 1/6/17

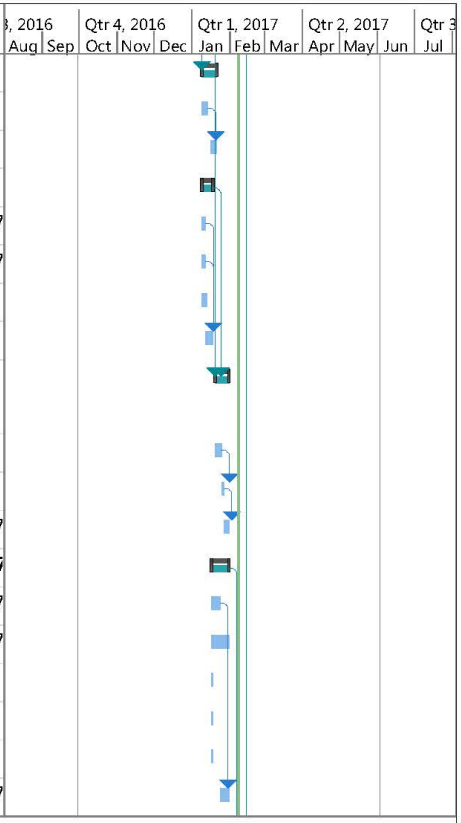


Project: Updated Gantt Chart.m
Date: Wed 2/8/17

Task	Inactive Summary	External Tasks
Split	Manual Task	External Milestone
Milestone	Duration-only	Deadline
Summary	Manual Summary Rollup	Progress
Project Summary	Manual Summary	Manual Progress
Inactive Task	Start-only	
Inactive Milestone	Finish-only	

Page 1

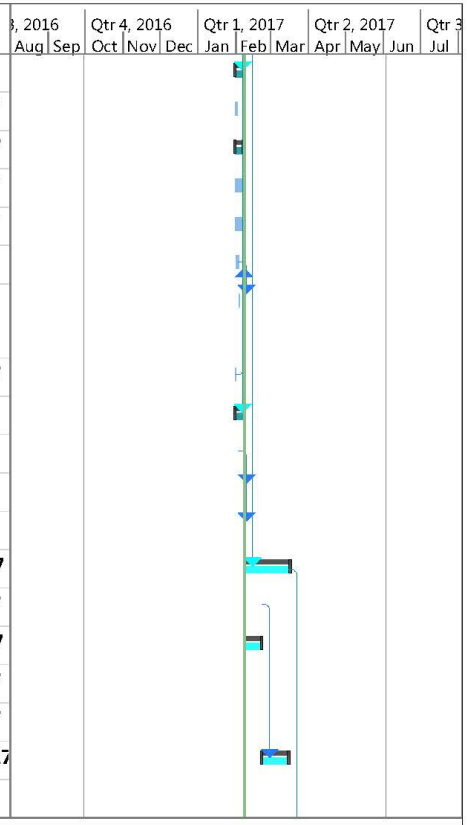
ID	Task Mode	Task Name	Duration	Start	Finish
41		**Mid Winter Quarter Tasks**	**10 days**	**Mon 1/9/17**	**Fri 1/20/17**
42		Collect product literature/data sheets	5 days	Mon 1/9/17	Fri 1/13/17
43		Order Final Components (if necessary)	5 days	Mon 1/16/17	Fri 1/20/17
44		**Subsystem layout**	**7 days**	**Mon 1/9/17**	**Tue 1/17/17**
45		Motor Couplers	3 days	Mon 1/9/17	Wed 1/11/17
46		Chassis	3 days	Mon 1/9/17	Wed 1/11/17
47		Electrical Layout	4 days	Mon 1/9/17	Thu 1/12/17
48		Motor Integration	4 days	Thu 1/12/17	Tue 1/17/17
49		**Preliminary Prototyping/Modifying of RC Car**	**7 days**	**Fri 1/20/17**	**Mon 1/30/17**
50		Test Motor Integration	3 days	Fri 1/20/17	Tue 1/24/17
51		Develop Elelctrical connections	2 days	Wed 1/25/17	Thu 1/26/17
52		Preliminanry Electrical System Testing	2 days	Fri 1/27/17	Mon 1/30/17
53		**Analysis to meet specifications**	**10 days**	**Tue 1/17/17**	**Mon 1/30/17**
54		Motor Capabilties	5 days	Tue 1/17/17	Mon 1/23/17
55		CG Calculations	10 days	Tue 1/17/17	Mon 1/30/17
56		Steering Model	1 day?	Tue 1/17/17	Tue 1/17/17
57		Chassis Design	1 day?	Tue 1/17/17	Tue 1/17/17
58		Hole Tear Calculations	1 day?	Tue 1/17/17	Tue 1/17/17
59		Coupler Fatigue Calculations	5 days	Tue 1/24/17	Mon 1/30/17



Project: Updated Gantt Chart.m
Date: Wed 2/8/17

Task
Split
Milestone
Summary
Project Summary
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
External Tasks
External Milestone
Deadline
Progress
Manual Progress

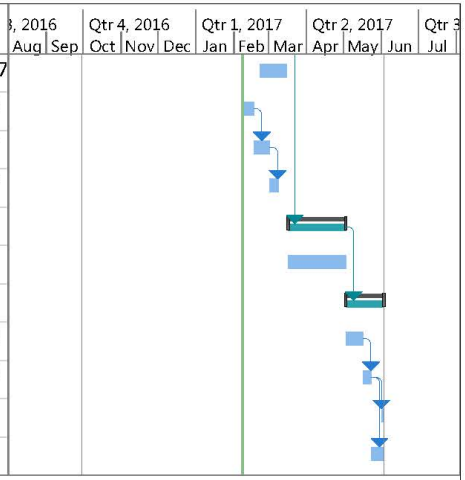
ID	Task Mode	Task Name	Duration	Start	Finish
60		**Report Documentation**	**6 days**	**Tue 1/31/17**	**Tue 2/7/17**
61		Discuss Safety Topics from FMEA	2 days	Tue 1/31/17	Wed 2/1/17
62		**Final Material Selection**	**5 days**	**Tue 1/31/17**	**Mon 2/6/17**
63		Couplers and Components	5 days	Tue 1/31/17	Mon 2/6/17
64		Chassis	5 days	Tue 1/31/17	Mon 2/6/17
65		Full CAD drawing	2 days	Wed 2/1/17	Thu 2/2/17
67		Develop Clear Manufacturing Instructions	1 day	Fri 2/3/17	Fri 2/3/17
68		Maintence and repair considerations	1 day	Tue 1/31/17	Tue 1/31/17
69		**Report Updates**	**6 days**	**Tue 1/31/17**	**Tue 2/7/17**
70		Update Budget and Examine Costs	3 days	Tue 1/31/17	Thu 2/2/17
71		UpdateManagement plan	3 days	Fri 2/3/17	Tue 2/7/17
72		Safety hazard checklist	3 days	Fri 2/3/17	Tue 2/7/17
73		**Project Update Report**	**27 days**	**Wed 2/8/17**	**Thu 3/16/17**
74		Order Raw Materials	10 days	Wed 2/8/17	Tue 2/21/17
75		**3-D Print Parts for checks**	**10 days**	**Wed 2/8/17**	**Tue 2/21/17**
76		Mounts and Housing	10 days	Wed 2/8/17	Tue 2/21/17
77		Shaft Couplers	5 days	Wed 2/8/17	Tue 2/14/17
78		**Begin Manufacturing of Simple Parts**	**16 days**	**Wed 2/22/17**	**Wed 3/15/17**
79		Chassis Shape	10 days	Wed 2/22/17	Tue 3/7/17



Project: Updated Gantt Chart.m
Date: Wed 2/8/17

Task
Split
Milestone
Summary
Project Summary
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
External Tasks
External Milestone
Deadline
Progress
Manual Progress

ID	Task Mode	Task Name	Duration	Start	Finish
80		Suspension Mounts	16 days	Wed 2/22/17	Wed 3/15/17
81		Experimental Design for Testing	7 days	Wed 2/8/17	Thu 2/16/17
82		Finish Project Update Report	9 days	Fri 2/17/17	Wed 3/1/17
83		Operator's Manual	5 days	Thu 3/2/17	Wed 3/8/17
87		**Project Hardware - Protoyping**	**33 days**	**Fri 3/17/17**	**Tue 5/2/17**
88		Final Prototype Construction	33 days	Fri 3/17/17	Tue 5/2/17
89		**Final Design Review**	**23 days**	**Wed 5/3/17**	**Fri 6/2/17**
90		Project Testing	10 days	Wed 5/3/17	Tue 5/16/17
91		Testing Results Documentation	5 days	Wed 5/17/17	Tue 5/23/17
92		Project Expo	2 days	Thu 6/1/17	Fri 6/2/17
93		Complete Final Report	8 days	Wed 5/24/17	Fri 6/2/17



Project: Updated Gantt Chart.m
Date: Wed 2/8/17

Task	Inactive Summary	External Tasks
Split	Manual Task	External Milestone
Milestone	Duration-only	Deadline
Summary	Manual Summary Rollup	Progress
Project Summary	Manual Summary	Manual Progress
Inactive Task	Start-only	
Inactive Milestone	Finish-only	

APPENDIX H DVP&R AND DFMEA

ME428 DVP&R Format												
Report Date		12/1/16	Sponsor	*Dr. Birdsong*				System	*Intelligent Vehicle Platform*		REPORTING ENGINEER:	
		TEST PLAN							**TEST REPORT**			
Item No	Specification or Clause Reference	Test Description	Acceptance Criteria	Test Responsibility	Test Stage	SAMPLES TESTED		TIMING		TEST RESULTS		NOTES
						Quantity	Type	Start date	Finish date	Test Result	Quantity Pass	Quantity Fail	
1	Trackwidth	Center of wheel to center of wheel measurement	275 mm +/- 15%	Chris	DV	1	B	5/22/17	5/23/17	276 mm - Pass	1	0	Uncertainty predicted +/- 1 mm
2	Wheelbase	Center of wheel to center of wheel measurement	406 mm +/- 15%	Chris	CV/DV	1	B	5/22/17	5/23/17	401 mm - Pass	1	0	Uncertainty predicted +/- 1 mm
3	Weight	Measure on scale	5.8 kg Max	Chris	CV	1	B	5/22/17	5/23/17	3.8 kg - Pass	1	0	Uncertainty analysis predicted +/- 50 g
4	Longitudinal CG Position	Position from center of rear wheel	203 mm +/- 25%	Chris	DV	1	B	5/22/17	5/23/17	204 mm - Pass	1	0	Uncertainty analysis predicted +/- 2 mm
5	CG Height	Lift front axle, measure reaction force on scale	50 mm Min	Chris	DV	1	B	5/22/17	5/23/17	75 mm - Pass	1	0	Uncertainty analysis predicted +/- 2 mm
6	Top Vehicle Speed	Test max speed on concrete for 10 yards	6.7 +/- 1 m/s	Chris	DV	1	B	5/25/17	5/25/17	Set Electronically in motor controller software	1	1	
7	Maximum Acceleration	Test max acceleration on concrete	2 m/s^2 +/- 50%	Chris	CV/DV	1	B	5/25/17	5/25/17	Set Electronically in motor controller software	1	1	
8	Turning Radius	Maxiumum steering angle, drive vehicle in circle	1600 mm +/- 25%	Chris	DV	1	B	N/A	N/A	Did not test			Theoretically calcualtion is a pass - 1300 mm
9	Rollover at top speed	Visually see inside wheel lose traction	Must Meet Standard	Chris	DV	1	B	N/A	N/A	Did not test			
10	Physically Robust	Visual inspection	Full Evaluation	Chris & Jay	DV	1	A	5/22/17	5/23/17	Pass	1	0	See Areas for improvement section in FDR.
11	Visual latency	Run sample collision avoidance algorithm	Must Meet Standard	Evan	DV		B	N/A	N/A	Did not test			
12	Works with Simulink	Development of algorithms	Must Meet Standard	Evan & Jay	CV	1	A	6/1/17	6/1/17	Pass	1	1	
13	Suspension	N/A (Yes or no)	Must Meet Standard	Jay	CV	1	B	6/1/17	6/1/17	Pass	1	1	
14	Digital I/O Ports	N/A (Yes or no)	NO ERROR (Must be able to hold 3	Evan	CV	1	B	6/1/17	6/1/17	Pass	1	1	
15	Independently Powered Wheels	N/A (Yes or no)	Must Meet Standard	Jay	CV/DV	1	B	6/1/17	6/1/17	Pass	1	1	
16	Tetherless	N/A (Yes or no)	Must Meet Standard	Everyone	CV	1	B	6/1/17	6/1/17	Fail - Not tetherless	1	1	Future modification necessary
17	Autonomous/Hybrid Control	N/A (Yes or no)	Must Meet Standard	Evan	CV/DV	1	A	6/1/17	6/1/17	Pass	1	1	
18	Battery Life	6 cycles of 10 min continued use with 15 min "reprogramming"	Must Meet Standard	Jay	CV/DV		B	N/A	N/A	Did not test			

Item / Function	Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) / Mechanism(s) of Failure	Occurence	Criticality	Recommended Action(s)	Responsibility & Target Completion Date	Actions Taken	Severity	Occurence	Criticality
Tests Simulink Algorithms	Loss of control	Car moves to fast	3	Bad Algorithm	7	21	Remote killswitch, sensors check/agree	Evan, 5/17/2017	Remote killswitch, maxiumum motor speed	3	7	21
				Component	3	9					3	9
				Faulty Sensor Data	2	6					2	6
		Wrong vehicle path	6	Bad Algorithm	7	42				6	7	21
				Component	3	18					3	9
				Not enough output info from sensors	4	24					4	12
				Faulty Sensor Data	2	12					2	6
Battery	Overheats	Loss of power	2	Excess current drainage	3	6	Position battery in safe location, limit operating time	Chris, 5/17/2017	LiPo safe bag for storage, voltage regulator to protect fragile electronics	2	3	9
				Overuse	3	6					3	9
	Over drained	Loss of power	2	Excess current drainage	3	6				2	3	9
				Overuse	3	6					3	9
	Punctured Casing	Exposure to toxic chemicals	8	Sharp object comes into contact with battery at high speeds	3	24				8	3	9
		Catches fire	9	Sharp object comes into contact with battery at high speeds	3	27				9	3	9
Protective Housing	Broken housing	Pinch points	6	High speed impacts from loss of control	4	24	Durable cage, killswitch to prevent loss of control, speed control	Jay, 5/17/2017	None taken, needs adaptation	6	4	12
				Fatigue failure from use	3	18					3	9
		Cage cannot protect components	5	High speed impacts from loss of control	4	20				5	4	12
				Fatigue failure from use	3	15					3	9
	Housing falls off	Damage to electronics	6	High speed impacts from loss of control	4	24				6	4	12
				Fatigue failure from	3	18					3	9
		Replacement of expensive parts	3	High speed impacts from loss of control	4	12				3	4	12
Motors	Burnout	No power to shafts	3	Motor lifetime	1	3	Design couplers for infinite life, detailed analysis	Jay, 5/17/2017	CNC couplers	3	1	3
	Coupler attachment breaks	Vehicle spins out of control	6	Fatigue failure in	3	18				6	3	9
				High speed impacts	4	24					4	12
Performs Like a Real Vehicle	Too stable	Roll over not possible	3	Center of gravity too low	4	12	Add wegiht/vary CG, limit motor supply current	Chris, 5/17/2017	Used existing Slash suspension and steering geometry, battery below chassis to lower CG	3	4	12
	Large power/weight ratio	Burn-out	2	Motors too powerful	5	10				2	5	15
		Can't test stbalilty	4	Motors too powerful	5	20				4	5	15
	Loss of traction	Can't test stabililty control	4	Excessive speeds	3	12				4	3	9
				Car too light	5	20					5	15
				Tires don't provide enough grip	3	12					3	9
		Vehicle crashes	5	Excessive speeds	3	15				5	3	9
				Car too light	5	25					5	15
				Tires don't provide enough grip	3	15					3	9

APPENDIX I BILL OF MATERIALS

CATEGORY	ITEM DESCRIPTION	QTY	COST PER UNIT	TOTAL COST	SOURCE
Shipping, etc.	Maxon Order + ?			16.00	
Traxxas Slash	Traxxas Slash 4x4 4WD	1	449.95	483.70	Amazon: https://www.amazon.com/Traxxas-68086-3-Electric-Vehicle-Colors/dp/B0115ERJOO/ref=sr_1_1?s=toys-and-games&ie=UTF8&qid=1486152828&sr=1-1&keywords=traxxas+slash+4x4&refinements=p_36%3A1253564011
Electronics	9-DOF Absolute Orientation IMU	1	34.95	34.95	Adafruit: https://www.adafruit.com/products/2472
	Raspberry Pi 3B	2	39.95	79.90	Adafruit: https://www.adafruit.com/products/3055
	Pi Cobbler Plus	1	6.95	6.95	Adafruit: https://www.adafruit.com/products/2029
	Maxbotix Ultrasonic Rangefinder	1	33.95	33.95	Adafruit: https://www.adafruit.com/products/983
	Raspberry Pi Cam	1	29.95	29.95	Adafruit: https://www.adafruit.com/products/3099
	Pi Cam Cable	1	1.95	1.95	Adafruit: https://www.adafruit.com/products/1648
	Keyfob Remote Control Button	1	6.95	6.95	Adafruit: https://www.adafruit.com/products/1648
	Keyfob Remote Receiver	1	4.95	4.95	Adafruit: https://www.adafruit.com/products/1097
	4GB SD Card	2	9.95	19.90	Adafruit: https://www.adafruit.com/products/1121
	Teensy 3.6 Microcontroller	1	37.50	37.50	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432068&uq=636220001070721314

Circuit Board and Parts	DC Level Shifter	1	18.73	18.73	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432067&uq=636220001070721314
	CAN Transceiver	2	2.19	4.38	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432071&uq=636220001070721314
	RGB Led Indicator	2	0.57	1.14	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432070&uq=636220001070721314
	Voltage Level Translator	2	0.85	1.70	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432069&uq=636220001070721314
	24 Pin Female/Male Header Connector	2	1.64	3.28	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432066&uq=636220001070721314
	4 Pin Female/Male Header Connector	2	0.60	1.20	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432065&uq=636220001070721314
	7 Pin Female/Male Header Connector	1	0.78	0.78	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432064&uq=636220001070721314
	6 Position Header Connector Through Hole Tin	2	0.64	1.28	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432062&uq=636220001070721314
	40 Pin Female/Male Header Connector	2	2.31	2.31	Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?Detail&itemSeq=218432061&uq=636220001070721314
	Custom Circuit Board	1	37.50	37.50	Osh Park:
	Battery	1	38.70	38.70	HobbyKing: https://hobbyking.com/en_us/zippy-compact-4000mah-7s-25c-lipo-pack.html

	0.1 µF Capacitor	25	Pack	0.33	http://www.digikey.com/products/en?newproducts=0&keywords=311-1343-1-ND&pkeyword=311-1343-1-ND&v=
	0.01 µF Capacitor	25	Pack	0.33	http://www.digikey.com/products/en?newproducts=0&keywords=311-1085-1-ND&pkeyword=311-1085-1-ND&v=
	1 µF Capacitor	3	Pack	0.99	http://www.digikey.com/products/en?keywords=478-8234-1-ND
	10 µF Capacitor	3	Pack	1.14	http://www.digikey.com/products/en?keywords=399-3684-1-ND
	5 A Fuse	3	Pack	7.65	http://www.digikey.com/products/en?keywords=F2969CT-ND
	26.7 Ω Resistor	10	Pack	0.13	http://www.digikey.com/products/en?keywords=311-26.7HRCT-ND
	3.3 Ω Resistor	10	Pack	0.10	http://www.digikey.com/products/en?keywords=311-3.3GRCT-ND
	Fine Pitch Screw Terminals 4pos	5	Pack	5.25	http://www.digikey.com/products/en?keywords=ED10563-ND
	6pos Female header	1	0.68	0.68	http://www.digikey.com/products/en/connectors-interconnects/rectangular-connectors-headers-receptacles-female-sockets/315?k=S7004-ND&pkeyword=S7004-ND
	Screw Terminal 3 Pos	2	0.50	1.00	http://www.digikey.com/products/en?keywords=ED2741-ND
	Screw Terminal 2 Pos	2	0.38	0.76	http://www.digikey.com/products/en?keywords=ED2740-ND
Motor	EC45 Motor (PN:397172)	5	20.00 (Sponsorship Price)	100.00	Maxon: http://www.maxonmotorusa.com/maxon/view/product/motor/ecmotor/ecflat/ecflat45/397172
	EPOS4 Compact 50/5 CAN Controller (PN: 541718)	5	20.00 (Sponsorship Price)	100.00	Maxon: http://www.maxonmotorusa.com/maxon/view/product/control/Positionierung/EPOS-4/541718

Raw Materials	Aluminum Round Stock 1 ft - 5 mm Tight Tol. Rod	1	7.99	8.63	McMaster – Carr 6940T12
	Aluminum Round Stock 1 ft - 10 mm Unpolished	1	1.80	1.94	McMaster – Carr 4634T36
	Aluminum Bar Stock 6” x 1” x ¾”	1	4.27	4.61	McMaster – Carr 8975K14
	Aluminum Bar Stock 12” x 1.625” x 1.625”	1	19.12	20.6	McMaster – Carr 9008K48
	Aluminum Bar Stock 12” x 1” x 3/4”	1	6.71	7.25	McMaster – Carr 8975K14
	Aluminum Sheet Stock 24” x 4” x 0.08”	1	13.50	14.58	McMaster – Carr 89015K192
	Nylon Sheet Stock 12” x 12” x 3/8”	1	38.52	41.60	McMaster – Carr 8540K117
Fasteners and Spacers	Miscellaneous	1	30.00	30.00	Local or Amazon

APPENDIX J PART AND ASSEMBLY DRAWINGS

APPENDIX K TESTING RESULTS AND DATA COLLECTION

Weight Data

Weight Data				Vehicle Measurements		
Rear Left [g]	Rear Right [g]	Front Left [g]	Front Right [g]	U_scale [g]	Length [mm]	U_length [mm]
867	1023	1016	946	0.5	401	5

Longitudinal CG position and uncertainty

Calcualtions			
R_r	18540.9	[N]	Rear Weight
W	37788.1	[N]	Total Weight
b	196.8	[mm]	Distance from front to CG
a	204.2	[mm]	Distance from rear to CG
Wr_unc	1890.7	g	Rear weight + .707
Wf_unc	1962.7	g	Front Weight + .707
L_unc	406.0	mm	Wheelbase + Unc
s1_a	0.0	mm	Sensitivity to rear weight
s2_a	0.0	mm	Sensitivity to front weight
s3_a	2.5	mm	Sensitivity to wheelbase
U_total	2.5	mm	Total uncertainty in "a"

Modified weight data

Measurement	Value	Unc.	Units	Description
FWl	1937	0.71	g	Front Weight Level
FWr	1996	0.71	g	Front Weight Raised
W	3832	1.00	g	Total Weight
L	401	5.00	mm	Wheelbase
H	118	1.00	mm	Rear Lift Height
r	55	1.00	mm	Effective wheel radius

Vertical CG position and uncertainty analysis.

Calculate			
CG Height	75.1	mm	
s_FWl	-0.2	g	Sensitivity to front weight
s_FWr	0.2	g	Sensitivity to rear weight
s_W	0.0	g	Sensitivity to total weight
s_L	0.5	mm	Sensitivity to wheelbase
s_H	-0.2	mm	Sensitivity to lift height
s_r	1.0	mm	Sensitivity to wheel radius
Uncertainty	1.2	mm	Total Uncertainty

APPENDIX L REPRODUCTION COSTS

Traxxas Parts

System	Description	Part #	Quantity	Cost (Each)	Total
Front Suspension and Steering	Front Driveshaft Assembly	6851X	2	$10.00	$20.00
	Caster Blocks	6832	1	$5.00	$5.00
	Steering Blocks	6837	1	$5.00	$5.00
	Steering Turnbuckle (58 mm)	5539	1	$7.50	$7.50
	Suspension Turnbuckle (49 mm)	3643	1	$7.50	$7.50
	Bellcranks	6845	1	$12.00	$12.00
	Spring Progressive +10%	6863	1	$5.00	$5.00
	Steering Linkage	6846	1	$3.00	$3.00
	Bumper	6853	1	$10.00	$10.00
Rear Suspension	Rear Driveshaft Assembly	6852X	2	$10.00	$20.00
	Carriers	1952	1	$3.00	$3.00
	Suspension Turnbuckle (39 mm)	3644	1	$7.00	$7.00
	Bumper	6838	1	$10.00	$10.00
	Spring Progressive +10%	6867	1	$6.50	$6.50
Both (Front and Rear)	A-Arm (Both)	3655X	2	$10.00	$20.00
	Ultrashocks	2662	1	$42.00	$42.00
	Wheel Nuts	3647	1	$3.25	$3.25
	Wheel Hubs	1654	2	$2.00	$4.00
	Ball Bearing	5119	1	$7.00	$7.00
	Ball Bearing	5116	3	$3.50	$10.50
	Suspension Pins	6834	1	$4.00	$4.00
	Wheels	5873	2	$50.00	$100.00
Chassis/Electronics	Receiver	6533	1	$50.00	$50.00
	Transmitter	6517	1	$60.00	$60.00
	Servo	2075	1	$40.00	$40.00
Hardware	Shoulder Screw	3642X	2	$3.00	$6.00
	M3 x 15 mm rounded (6)	2579	2	$2.50	$5.00
	M3 x 6mm flat (6)	3932	1	$2.50	$2.50
			Grand Total		$475.75

Mechanical Hardware and Raw Materials

Description	Quantity	Price (Each)	Total
Nylon Sheet Stock (12” x 12” x 3/8”)	1	$38.52	$38.52
Aluminum Round Stock (1 ft - 5 mm Tight Tol. Rod)	1	$7.99	$7.99
Aluminum Bar Stock (1' x 2" x 5/16”)	1	$5.48	$5.48
Aluminum Bar Stock (2' x 3/4” x 3/4”)	1	$8.78	$8.78
Aluminum 3003 Sheet Stock (2' x 6” x 0.08”)	1	$8.96	$8.96
Acrylic sheet (6" x 4" x 7/64")	1	$6.42	$6.42
Heat-Set Inserts M3, 3.8 mm (100 pack)	1	$12.30	$12.30
Aluminum 45 mm threaded hex standoff, M3 thread	6	$1.57	$9.42
Aluminum 10 mm threaded hex standoff, M4 thread	3	$1.27	$3.81
Aluminum 20 mm threaded hex standoff, M4 thread	3	$1.57	$4.71
M3 x 10 mm socket (50 pack)	1	$5.43	$5.43
M3 x 6mm rounded (100 pack)	1	$6.88	$6.88
M3 x 4 mm flat (50 pack)	1	$7.14	$7.14
M4 x 14 mm socket (50 pack)	1	$9.84	$9.84
M2.5 x 20 mm socket (25 pack)	1	$7.10	$7.10
M2 x 12 mm socket (50 pack)	1	$5.00	$5.00
	Grand Total		125.84

APPENDIX M CODE FILES