Electronic CVT - Controls
Final Design Report

Sponsored by Cal Poly Baja SAE

Prepared by Controls Crew Baja

Authors:

Alec Hardy	awhardy@calpoly.edu
Jessalyn Bernick	jbernick@calpoly.edu
Nick Capdevila	ncapdevi@calpoly.edu
Tristan Perry	tcperry@calpoly.edu

Advisement by:

Professor John Fabijanic	jfabijan@calpoly.edu

July 2019

Table of Contents

Table of Tables

Table of Figures

Abstract

The following document outlines the design process, manufacturing, and testing of the control system for an electronically controlled continuously variable transmission (ECVT). This control system was integrated into the custom designed and manufactured mechanical transmission system created in parallel by another senior project group. The transmission was designed for use in the Cal Poly Baja SAE vehicle. Through researching customer needs, competition requirements, previous and alternate CVT designs, and vehicle characteristics, we were able to determine the requirements and specifications for our unique system. Input, output, speed, and durability requirements guided our hardware selection. The primary components which comprised our system include an alternator and regulator, a custom circuit board, rotary encoders and hall effect sensors, brushed DC motors, lead screws, and a custom system enclosure; further details are included in the Final Design section of this report. With the knowledge of our vehicle characteristics, actuation mode, and inputs, a system model determined that a standard proportional + integral action (PI) controller would be sufficient to obtain the speed and accuracy demanded by our customer needs. Electrical components were assembled, tested, and programmed on a prototyping breadboard, and a custom printed circuit board (PCB) was outsourced for manufacture following qualification of our prototype. The final production board was bench tested with the mechanical CVT system to ensure it met all customer and design requirements. Furthermore, the enclosure was tested to ensure the safety and durability of the electrical systems. Planning and timing mismanagement between our team, the mechanical design team, and Cal Poly SAE Baja team, in conjunction with controls specific setbacks, resulted in the final combined system remaining untested on the Baja vehicle. This project is being continued by a new senior project group which will continue to test and improve upon the current system during the 2019-2020 academic year.

1 Introduction

Our purpose for designing and constructing an electronically controlled continuously variable transmission (ECVT) is to improve the performance of the Cal Poly Baja SAE vehicle. The currently existing transmission is a bottleneck in the vehicle's performance because of its inability to be reliably tuned for different race events. An electronically controlled CVT will offer optimum vehicle performance by maintaining the maximum power throughput from the car's engine to the wheels. Electronic in nature, the control mechanism is easily tunable and adjustable for the Baja team, and each tune will be entirely repeatable to ensure reliability.

Due to the large scope of work associated with designing, manufacturing, controlling, and testing a complete eCVT, the project has been divided into two sections to be completed by two different groups. The design and manufacturing of the transmission is to be completed by the Electronic CVT - Mechanical Design group while the focus of our group, Electronic CVT - Controls, is to design, implement, and test the controls system.

This final design report (FDR) document covers our background research, objectives, concept design development, final design, manufacturing plan, design verification plan, and project management. Background information consists of summaries from our initial team and sponsor meetings, a table of existing transmission designs and pertinent patents, relevant industry standards and regulations, and a summary of technical literature research. Our objectives set the scope of work of our project and states specifically what we will be producing, including problem statement, list of customer needs and wants, quality function deployment, and updated engineering specifications with discussion. Our concept design development section discusses concept development and selection, preliminary analyses, concept modeling, concept functionality, and a discussion of design challenged and risks. The final design portion outlines the overall design with wiring schematics and pseudo-code, evidence that we will meet our design requirements, and a discussion of safety, maintenance, and repair. Our manufacturing plan gives information on how we planned to purchase materials, assemble our protoboard and PCB, and our plan for building our base control algorithm. Additionally, it covers how manufacturing occurred in real life over the course of the project. Our design verification plan gives information about how we planned to ensure that we've met our specifications, and the testing we will perform to improve to tune and validate our controls, along with details on the testing that was actually completed and is upcoming. Lastly, project management covers our theoretical overall design process; key deliverables and project timeline; techniques to be used for prototyping, analysis, and testing; and a discussion of the real timelines and results produced.

2 Background

The following section contains background research on already existing transmission systems, including automatic transmissions, work done by other SAE Baja teams, mechanical continuously variable transmissions, electronically controlled continuously variable transmissions.

2.1 Customer Needs: Interviews and Observations

For this project, we first determined our customers and found that there are many members of the Baja SAE team that will interact with our CVT, including:

- Our Project Teams (both the controls and design group)
- CVT leads
- Electronics Leads
- Manufacturing Leads
- Competition Drivers

Through discussion and interviews with our customers, we found that our project was to create a method of electronically controlling a CVT that could meet the performance needs of a Baja SAE off-road vehicle in a reliable manner, while ensuring that it was easy to use and pass down through the team over the generations. In addition, we plan to create a model of our CVT for intelligent tuning, and to allow future generations to make informed design changes. In order to tackle this problem, we had to first understand the mechanics of how a CVT works, so that we could begin to determine how this shifting could be electronically controlled.

Interviews were conducted, shown in their entirety in Appendix A - Customer Interviews .
These interviews consisted of questions composed to be open-ended to allow for honest feedback from the customers about the problems they face with the current CVT. These interview results are further analyzed in section 3.1 of this document where the problem statement is discussed.

2.2 Technical Information and Existing Products

Through researching several sources we were able to build up a technical background of knowledge of CVT design, modeling, and control schemes. It was most important to get a fundamental understand of the shifting dynamics and tuning basic of a mechanical CVT. “Aaen’s Clutch tuning Handbook” was the primary source used to build this knowledge. The CVT shifts to keep the engine in its power band so it can produce the highest amount of torque and speed. Figure 1 shows a speed diagram of a vehicle accelerating to top speed, the CVT is responsible for keeping the engine in the power band despite change road load. The solid lines show the maximum and minimum speed ratios of the CVT (Aeen).

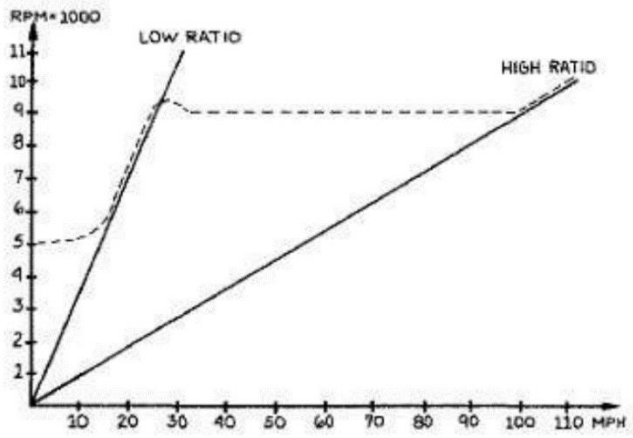


Figure 1: Speed Diagram showing High and Low Speed Ratios.

As described by this resource, the CVT has two main assemblies, the driver clutch (primary) and the driven clutch (secondary). Both clutches utilize two conical sheaves which move in and out to produce an infinite number of ratios within a range set by the maximum and minimum size of the sheaves. To change ratio, the driver and driven clutch expand and contract based on different engine and road conditions. The driver clutch, which is fully expanded when the engine is off to create a minimum pulley size, uses a system of weights that roll of ramps to contract the sheaves and increase its effective diameter. To control the movement of the weights, a spring is used so that the shift does not happen all at once. Adjustments made to the driver clutch greatly influence engine speed. The driven clutch, which with the engine off begins fully contracted or maximum pulley size, operates by the balancing back torque from the road and belt friction. Adjusting the driven clutch most dramatically changes the efficiency and back shifting, reducing the ratio (Aeen).

Efficiency of the CVT was simply defined as power out over power in. The factors that make up efficiency are more complex and includes pulley radius, belt speed, sheave angle, and clamping force. Decreasing pulley radius decreases efficiency because it increases the amount of force needed to bend the belt around the pulley. Increasing belt speed decreases efficiency because it increases frictional loss between the sheave and the belt. Clamping force is the only factor that can be controlled in tuning and is the most difficult to get right. Too much clamping force and the belt gets pinched increasing exit friction, too little and the belt begins to slip excessively and limits max torque. All these factors combined would look like the graph shown in Figure 2. In Figure 2, line A shows a CVT with too much clamping force, and line B Shows a properly tuned CVT. The beginning of the curve starts low because the driver pulley is at its smallest, the driver is smaller than the driven because a bigger speed reduction is needed for the transmission, thus requires higher bending forces. In the middle of the ratio the efficiency is the highest because the belt speed is not high enough and pulley size is not small enough to dramatically efficiency. The last part of the curve shows belt speed losses taking over and reducing efficiency. Finally the graph shows two lines B with the correct amount of clamping force and C with too much clamping force.

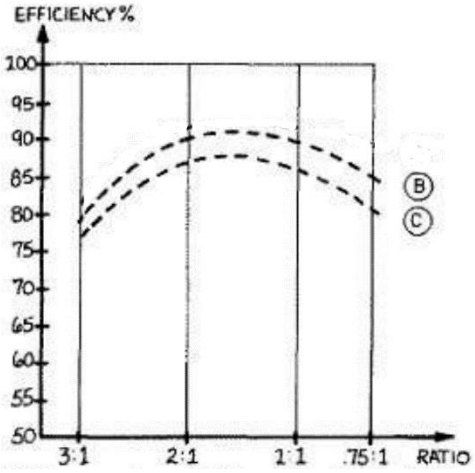


Figure 2: Hypothetical Graph of CVT Ratio Over Ratio Range.

After developing a solid understanding of the dynamics and mechanical design of a CVT, the control schemes and methodologies are identified. Transmission control on modern vehicles is governed by a "Transmission Control Module" (TCM) which uses input from various sensors and the engine control module to determine which gear to shift to. The TCM is "designed to optimize vehicle performance, shift quality, and fuel efficiency" (Clemson University Vehicular Electronics Laboratory). An understanding of the sensors used in modern day transmissions, regardless of transmission type (ie. automatic vs continuously variable), is an integral part in determining the sensors that will need to be implemented to electronically control a CVT. Figure 3 below shows the TCM sensor inputs (annotated in yellow) for a standard automatic transmission used in commercial vehicles today (Subaru).

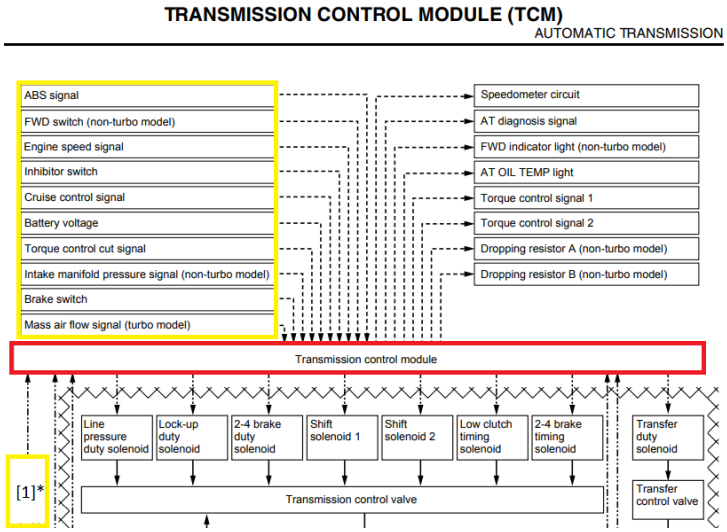


Figure 3: Transmission Control Module sensor inputs for a modern passenger vehicle.

The transmission outlined above operates mechanically on much different principles compared to an electronically actuated CVT. While the vehicle's sensor inputs remain the same regardless of the mechanical system inside the transmission, internal sensors must be specific to the unique transmission design. It was necessary to identify which specific mechanical components inside our implementation of a CVT will need to be tracked, and accordingly the correct sensors will need to be identified to do so. The process of identifying and selecting sensors is covered in more detail in Section 4 of this document. A preliminary list of sensors is included as Appendix E - List of Potentially Required Sensors, and our final list is within Appendix L – Bill of Materials. Research on sensor types and usage was performed by analyzing US patents. Specifically, the absolute position encoder patented in 1959 by S. Reiner was analyzed (Reiner) along with the hall effect quadrature encoder patented in 1965 by A. G. Lautzenhiser that provides highly accurate relative position (Lautzenhiser). This research helped narrow down types of position sensors that can be implemented in our CVT controller.

While designing and implementing our own version of a transmission control module, it is very important to select a controller that meets the demands of the system being controlled. Because it is necessary to be monitoring and reacting to data from multiple sensors simultaneously, a fast enough clock speed and software execution time for our controlling module is an integral part in maintaining a software structure that can run tasks seemingly simultaneously. This requirement coincides with the sample frequencies that sensor data will need to be obtained. The selection of proper microcontroller hardware guarantees that we can meet our control requirements.

Currently there exists a wide selection of microcontrollers on the market, each with different processors, uses, and features. The majority of microcontrollers are programmed in Embedded C/C++, while some of the newer experimental boards can run interpreted languages such as MicroPython. An understanding of the implications of using a compiled language such as Embedded C/C++ rather than an interpreted language such as MicroPython is very important when considering the design, development, and operation of the completed system. In Table 1and

Table 2 below the benefits and drawbacks of coding in Embedded C/C++ and MicroPython are addressed, respectively.

Table 1: Pros and Cons of using an embedded C/C++ language.

Embedded C/C++	
Pros	Cons
Very fast/efficient. Up to 2 orders of magnitudes faster than interpreted languages.	No real-time debugger.
Compiled language.	More code required for same functionality.
Typed language.	Very complex memory management.
	Difficult to read and understand.

Table 2: Pros and Cons of using interpreted MicroPython

MicroPython	
Pros	Cons
Very easy to read and write.	Interpreted language – Up to 2 orders of magnitudes slower than a compiled language.
Real-time debugger.	Untyped language.
REPL (Read-eval-print loop).	

Said and done, the benefit to coding in MicroPython is the drastic simplification in code writing and readability. However, the performance of the interpreted language is orders of magnitude poorer compared to the more robust, compiled C/C++ language.

Controller selection can now be based on preferred coding language and processor performance. The "official MicroPython microcontroller board" is powered by the STM32F405RG microcontroller based on the ARM Cortex-M4 32-bit RISC processing unit. The processing unit operates at 168MHz and contains a hardware floating point unit, while the board features one megabyte of flash read-only memory and 192KiB of random access memory (George). The most common type of embedded C/C++ boards are Arduino boards – a combination of hardware and software that runs a single-board microcontroller. The most prevalent Arduino board – the Arduino Uno – is powered by the ATmega328P microchip, which features an 8-bit AVR RISC processing unit operating at clock speed of 16MHz, and includes 32 kilobytes of flash random access memory, 2 kilobytes of electronically erasable read-only memory, and 1 kilobyte of static random access memory. Arduino's most powerful board is the Arduino Due, powered by the 32-bit Atmel SAM3X8E ARM Coretex-M3 RISC processing unit operating at a clock frequency of 84MHz. This board contains half a megabyte of flash memory and 96 kilobytes of static random access memory.

One of the most important considerations of choosing a microcontroller for a project involving input from multiple quadrature encoders is the ability to easily decode each of the encoders quickly and accurately. As per the Atmel SAM3X8E SAM3X8C SAM3X4E SAM3X4C SAM3A8C SAM3A8C Datasheet, the Arduino Due discussed above contains a single embedded hardware quadrature decoder module (QDEC) driven by timer counter modules on two different ports, ports A and B. Then QDEC is driven by three different pins, TIOA0, TIOB0, and TIOB1; the first two respective pins connected to two quadrature channels and the third pin connected to an optional index channel used for angular speed calculation (Atmel). This would be problematic if choosing this MCU for implementation in a system where multiple quadrature encoders are necessary without purchasing additional quadrature encoder hardware modules. These modules exist, however, they will require additional circuitry and configuration, as they

communicate to the MCU through I2C (Inter-Integrated circuit) or SPI (Serial peripheral interface) protocols (Refvem).

Paul J Stoffregen and Robin C Coon of PJRC have developed a USB development board using a 32-bit 120 MHz ARM Cortex-M4 processor with floating point unit. Version 3.5 of this board, named "Teensy", has the added benefit of 5V tolerance on all pins. This board is open-source, inexpensive, heavily documented, and includes an MCU that is much more powerful compared to the MCU's of most official Arduino boards. The open source schematic is shown below in Figure 4 (PJRC). The MK64FX512 MCU that is featured by the Teensy 3.5 has two embedded 2-channel Flex-Timer modules that can be configured to run in Quadrature Decoder Mode. This can be achieved by setting FTMEN=1 and QUADEN=1 on the appropriate module. The encoder module operates by reading raising and falling edges from two encoder phase inputs, as shown below in Figure 5 (Freescale Semiconductor, Inc.). This will benefit real-time hardware quadrature decoding by alleviating the concern of needing to use interrupts to increment or decrement the encoder counter each time a pulse is received from the encoder and will not require any additional hardware.

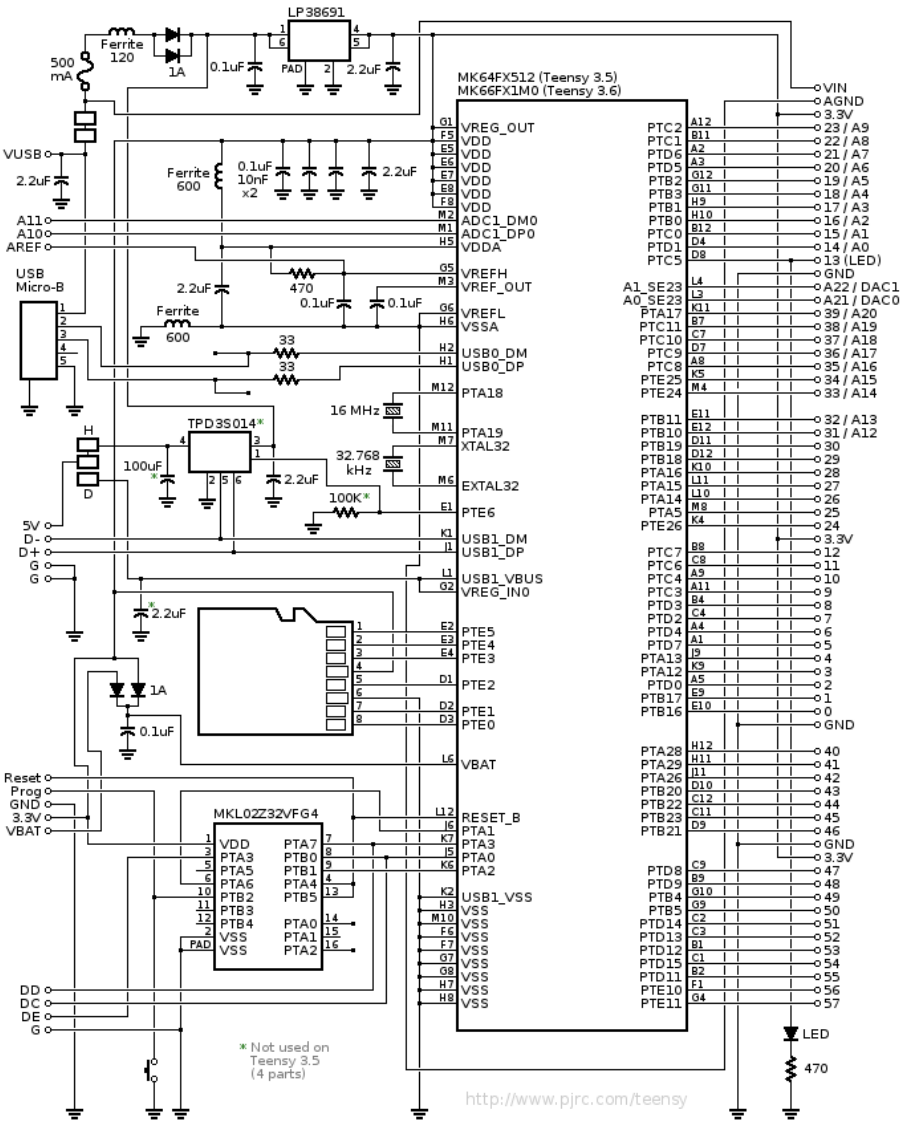


Figure 4: Schematic of Teensy 3.5 USB development board.

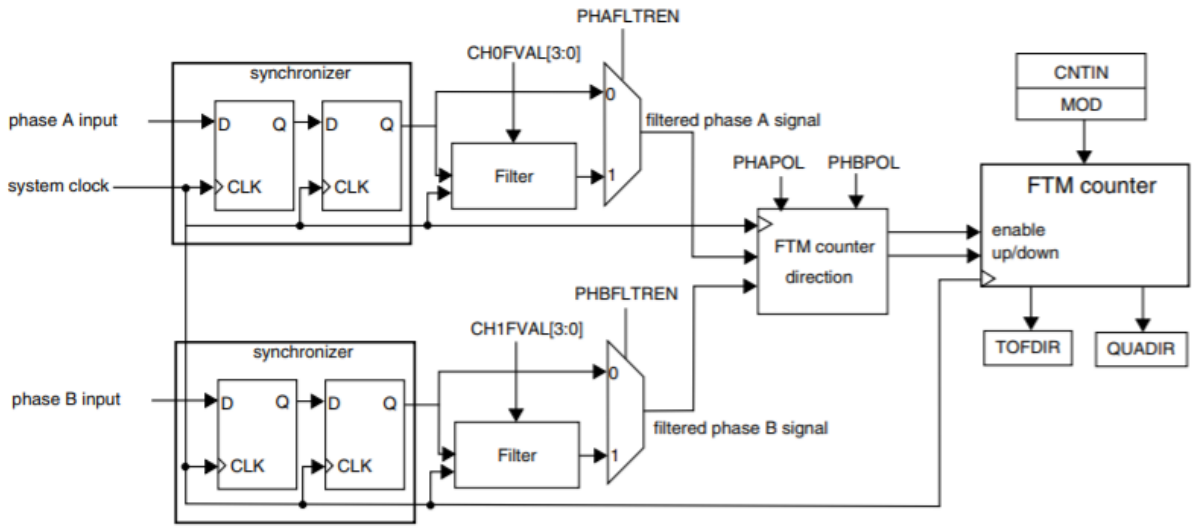


Figure 5: FlexTimer Module Quadrature Decoder block diagram.

As discussed above, programming the ARM or AVR MCU in C/C++ will yield a significantly higher amount of performance from the same hardware, but at a cost of complexity and development time. The ARM processor used in the Teensy board requires the GNU Embedded Toolchain for Arm to compile C/C++ code for the ARM architecture. The specific toolchain required, *gcc-arm-none-eabi* (Arm Limited), is much more complex and convoluted in comparison to the AVR toolchain, *gcc-avr*, and the AVR C Library, *avr-libc*, which can quickly and simply be implemented to program AVR systems through an in-system programmer (ISP) with the AVRDUDE software (Free Software Foundation, Inc.). However, if implementing an ARM MCU in the same configuration as the Teensy development board, the Teensyduino program (PJRC) can be used to port Arduino code to ARM compatible code, saving a huge amount of time during programming. For example, the Encoder Arduino library has been tested to work fully with any version of Teensy. This specific library uses interrupts to track encoder position and does not implement the FlexTimer modules of the MK64FX512, however, it's use will spare tens of hours of development and debugging and may be acceptable to use in production because of the high clock speed of the MCU. If necessary, a custom header can be written to implement the FlexTimer module.

When considering our method of actuation, it is important to consider the size and speed requirement of this actuation. During actuation, we will want to have control over the clamping force applied to the pulley(s), which can be most easily controlled through linear actuation. It was important to make an educated decision about the type of motor used to actuate our pulley(s), and the benefits and downfalls of each can be seen in Table 1 below. The correct combination of actuator motor and mechanical actuation method depend on the results of our modeling, which give us a better idea of the force input, speed, and precision required to obtain acceptable results.

Table 3: Background of Actuator Drivers

Motor Type	Advantages	Disadvantages
Brushed DC	Simple Speed Control Low Initial Cost	Low Lifespan
Brushless DC	Long Lifespan Low Maintenance High Efficiency	High Initial Cost Closed Loop Controller
Stepper	Precise Positioning High Holding Torque Open Loop Controller	Finite Number of Positions Non-Seamless Motion
Magnetic Coil Linear Actuator	No Mechanical Counterpart Low Cost Small Profile High Speed	High Power Draw Low Positional Accuracy Can Interfere w/ Sensors

As the controls group, one of the largest obstacles we will face in providing reliability is ensuring that our wiring and electronics remain protected both from human factors and the environment. Through research on International Protection (IP) ratings, we have decided to design to IP67 standards, meaning that our electronics will be protected from total dust ingress, as well as protected from immersion up to 1 meter (Rainford Solutions). This decision is based on our testing and competition environment, which includes high levels of dust, and frequently includes water features, which result in limited submersion of our vehicle.

In lieu of additional patent research, papers from other SAE Baja teams were referenced. As a start, University of Michigan Baja team designed their own mechanically actuated CVT (Justin Lopas), and their design considerations were looked at. Much of the paper focuses on the mechanical design however, it was a good starting point as it lead us to other more useful sources. The sensors used in their "Testing CVT" could be good options to look into as they have been used by in the exact same situation as our CVT will be used in, however, they do not have information relating to electronic actuation or supporting equations, it is merely a good paper for qualitative understanding.

A paper from the University of Akron (Gibbs) that researched several different CVT actuation methods in an effort to conclude if a computer controlled CVT could result in performance gains in small vehicle performance. This paper provided equations that define the speed of actuation in relation to the change of ratio with respect to time, which can be used to determine our necessary actuation speed. Equations documented in this paper also provide a simple way for calculating necessary clamping force using rudimentary methods of finding torque, which can be applied to our beginning model, before our final product is created and tested. It is important to note that these calculations neglect slip and other efficiency losses, and must be adjusted later to produce the best results. Later in the paper, testing data, concludes that an electromechanically controlled could result in efficiency gains.

A final point of research is the Baja SAE Competition Rules, which must be met in order to make this product usable. A list of relevant rules are listed in Appendix C -Baja SAE Relevant Rules, and can be summarized into limitations on actuator selection, powertrain guarding, and data acquisition.

3 Objectives

This section of the document contains information about the problem statement delivered to our sponsor and our team's proposed objective solution. Information in this section includes a boundary diagram, customer needs and wants determined through interview and observation, quality function deployment (QFD), and a list and discussion of engineering design specifications.

3.1 Problem Statement

Our team has taken on the challenge of electronically controlling a custom CVT for the Cal Poly Baja team. The purpose of this project is to improve upon the current CVT, which is purchased from Gaged, and causes many issues, including difficult tuning and unsatisfactory reliability. We will be applying the same engineering concepts involved in the Gaged CVT, but using a controls system to change ratio in order to allow for easier testing and tuning, and less off-the-car time for the CVT. One of our main problems with the Gaged CVT is that tuning it requires a variety of springs, weights, and cams, which all must be adjusted off the car, resulting in a lost time. Additionally, properly tuning requires trial and error and cannot be perfectly recreated every time. With an electronically-controlled CVT most of our tunes can be made externally (without removing the CVT from the car), and our tunes will perform the same each time they are applied, greatly improving our CVT reliability at competition.

Through interviews of our customers (listed in Background), in conjunction with our own requirements, we have compiled the following list of desires for CVT improvement:

- Reliable Performance, including Robust Electronic Connections
- Faster Backshift
- Lighter Weight
- Increased Torque Output
- Better Overall Event Performance
- Easy External Access to Tuning for Users
- Output a Variety of Data (Engine Speed, Vehicle Speed, Rear Wheel Speed, Temperature)
- Mounting Locations for Sensors
- Board for Data Acquistion
- Quick Boot Time
- Compliant with Baja SAE Competition Rules

Our interviews, shown in their entirety in Appendix A - Customer Interviews , were open-ended to allow for honest feedback from the customers about the problems they face with the current CVT. We found that their comments overlapped with one another, guiding a clear path to our requirements for best performance.

3.2 Quality Function Deployment and Engineering Specifications

From established requirements, we performed Quality Function Deployment (QFD), to establish the relationship of the customer requirements to our engineering deliverables and compare our solutions against our competitors. It can be seen in our QFD, shown in Appendix B - QFD House of Quality, that the main engineering objectives we must focus on are our time to top speed, the hill grade that our car can handle, and ability to output maximum torque. Meeting these requirements, along with our other goals, will make our product meet the specific needs of the current Cal Poly Baja car as best as possible.

Through these considerations, we have created the following table of engineering requirements. Table 4 lists the overall CVT requirements derived from the customer requests. The ability to meet these requirements is dependent on both the mechanical and control systems.

Table 4: Specification Table for CVT System Performance

Description	Target	Tolerance	Risk	Compliance
Maximum Shift Velocity	0.8 in/s (from limiting acceleration case)	+/- 0.1 in/s	H	T, A
Maximum Clamping Force	650 lbf	FOS. 1.25	H	T, A
Max Hill Grade	120% (from tipping limit)	+/-5%	H	T, A
Belt Slip	3%	+/-2%	H	T
Steps to CVT Tuning	6	+/-2	L	T
Cost	$1000	+/- $150	L	---
Precision of Ratio	0.075	+/- .05	L	T, I, A
Maximum Deviation from Desired Engine Rpm	50 rpm	+/- 25 rpm	H	T, A, I
Maximum Total Current Draw	15A	+/-1A	L	A
Actuator Driver Voltage	12V	+/-2V	H	A
ESD Protection	Protection above 30V	---	H	A
Controller Protection	IP67	---	H	I

The shift velocity target comes from our simulation of an acceleration run, which will be discussed later in this report. The max hill grade requirement directly comes from the customer, and affects the road load the CVT will experience, which is factored into the system model. The

customer's request was to minimize belt slip and according to our research, a CVT operates best with roughly 3% slip. Also requested by the customer was a simple tuning. To set a benchmark for this, the team step a goal of 6 based on reducing the current number of steps to tune the Gaged CVT.

The cost target was based on the availability of funds from our sponsor. The deviation of the engine rpm is a target for the control loop to be able to hold the engine rpm and the target and was set based on the sensitivity of the dyno curve. The current draw and voltage limits were set, with a margin for safety for amperage, based on the maximum output from the biggest alternator Briggs sells. Controller protections were selected very conservatively to ensure the reliability of the system.

As a collaborative senior project with both the Baja SAE team, and our sister senior project, Baja Electronic CVT- Mechanical Design group, it is important that we understand which responsibilities lie under this project, as opposed to other groups. We have created the following boundary diagram to represent our portion of the system.

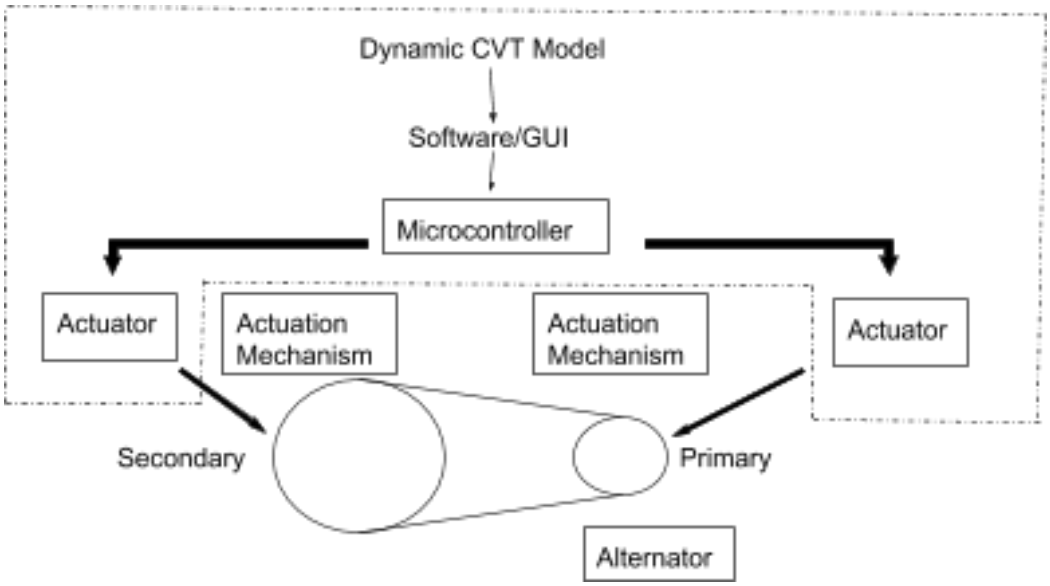


Figure 6: Boundary Diagram of the Baja eCVT- Control Project

4 Concept Design Development

4.1 Concept Development Process & Results

Control System design

Several control algorithms were researched, with a few simulated, to figure out which controller design would work best for this system.

PI controller

PI controllers are the most common type of control algorithm that is taught in an introductory course to controls and are an industry standard. The basic control architecture for a PI controller is shown in Figure 7. PI controllers are commonly used for systems that have a single actuator controlling a single state in the system. However, PI controllers can also be used to control multiple states with a single actuator or multiple states with several actuators. PI controllers are commonly used for systems that have a single actuator controlling a single state in the system. However, PI controllers can also be used to control multiple states with a single actuator or multiple states with several actuators.

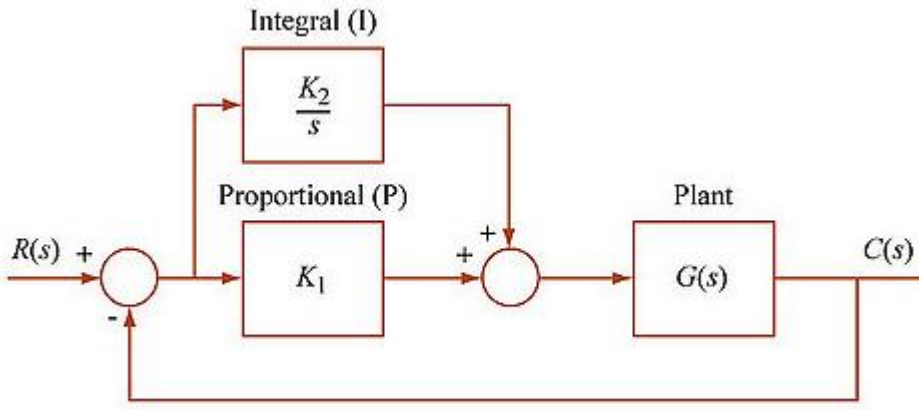


Figure 7: Basic Proportional-Integral Controller Architecture

Pros: PI controllers are easy to design, intuitive control action, easy to implement in hardware and easy to tune.

Cons: PI controllers are only guaranteed to work for linear systems, not robust.

Fuzzy Logic

Fuzzy logic removes the math from a standard control algorithm and controls the system solely based upon user-programmed logic. The basic control architecture for a fuzzy logic controller is shown in Figure 8. A fuzzy logic controller is designed from the user's deductive reasoning of how the actuator should control the system's state based on the current sensor inputs. Because fuzzy logic control is designed based on user intuition, a fuzzy logic controller is not designed

based upon a mathematical model of the system. It is easy to make a bad fuzzy logic control. However because of a fuzzy logic controller's degrees of freedom, it is possible to build a functional controller.

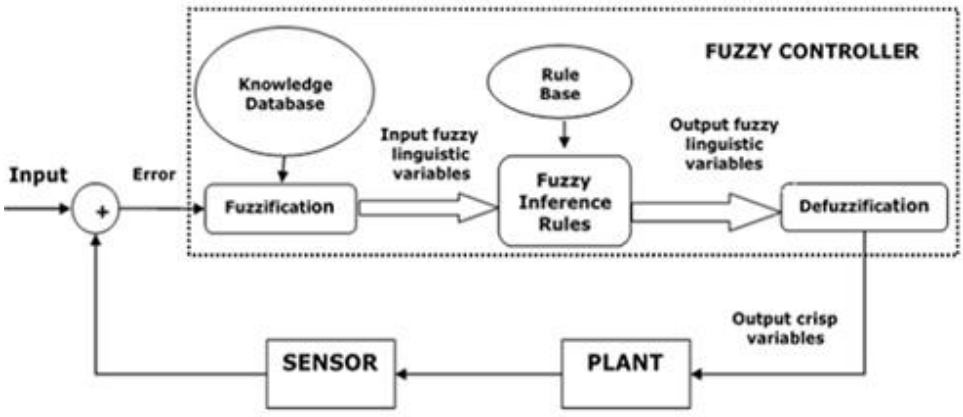


Figure 8: Fuzzy Logic Controller Architecture

Pros: A model of the plant is not required to design the controller, lots of design freedom to make a robust control.

Cons: Very difficult to tune, not designed to be optimal with respect with the system state or input, difficult to program.

Optimal Control/ FSFB

Optimal control and full state feedback (FSFB) are lumped in the same control algorithm category because optimal control uses the same feedback principle as FSFB, but optimal control is used to design optimal FSFB gains based upon minimizing some cost function relating to the plant. The basic control architecture for FSFB is shown in Figure 9. FSFB is used primarily to control multi-input-multi-output systems. FSFB feedbacks the state of the system that when multiplied by the feedback gain, k, the system's eigenvalues become negative real parts. Optimal control uses mathematical analysis to pick an 'optimal' K gain that minimizes some desired variable of the system. FSFB is used primarily to regulate the system (drive the states to zero), but can also be used to track some input into the system.

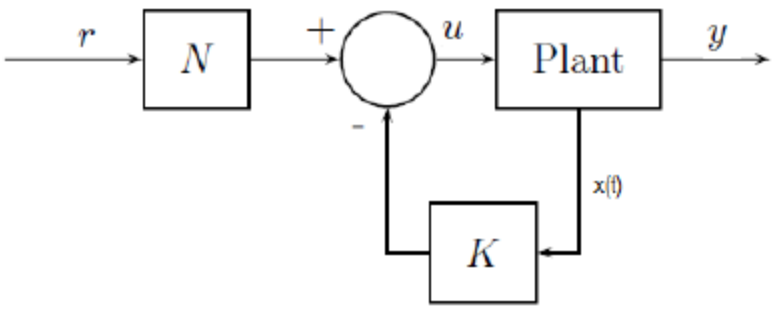


Figure 9: Optimal Control and Full State Feedback

Pros: FSFB is the best when trying to control multiple states, easy to implement in hardware, easy to tune.

Cons: Abstracts the intuition from the controller, not easy to design for performance requirements.

Look up Table

A look up table is not a control algorithm; however, it could be a viable solution. Since an IC engine runs at peak power for a certain motor speed it is the case that a unique gear ratio of the CVT exists for every wheel speed. A plot of the Baja IC engine's power vs. engine rpm is shown in Figure 10.

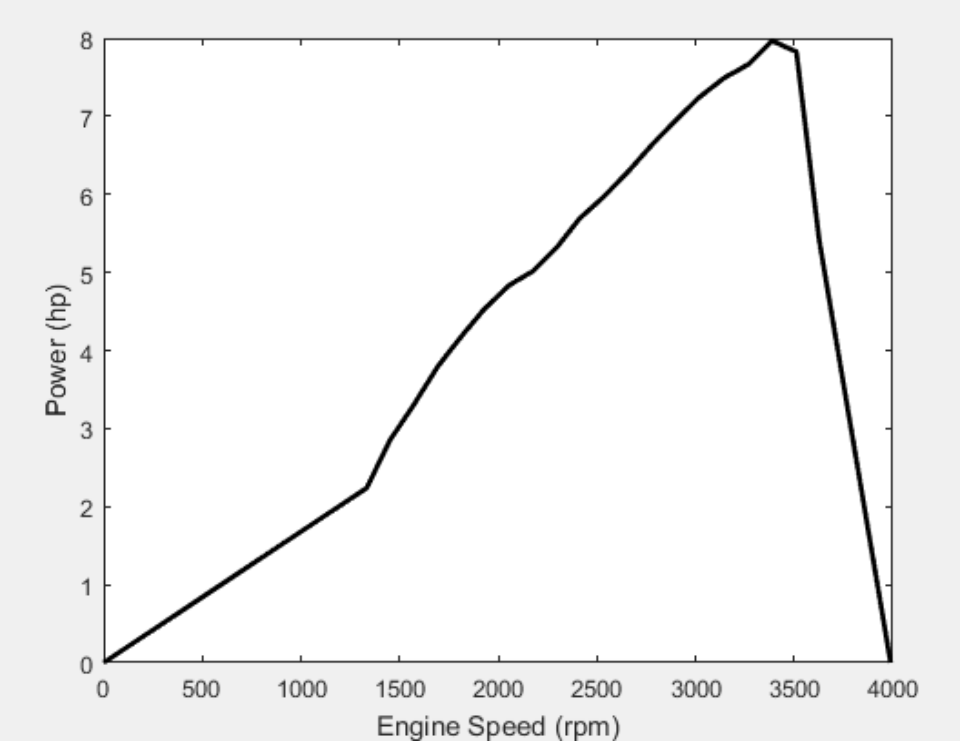


Figure 10: Baja Engine Power Curve

It is seen from Figure 10 that the peak power output of Baja's IC engine when the engine's speed is at about 3400 rpm. Figure (ideal shift curve) shows the CVT's gear ratio vs. the wheel's rotational speed.

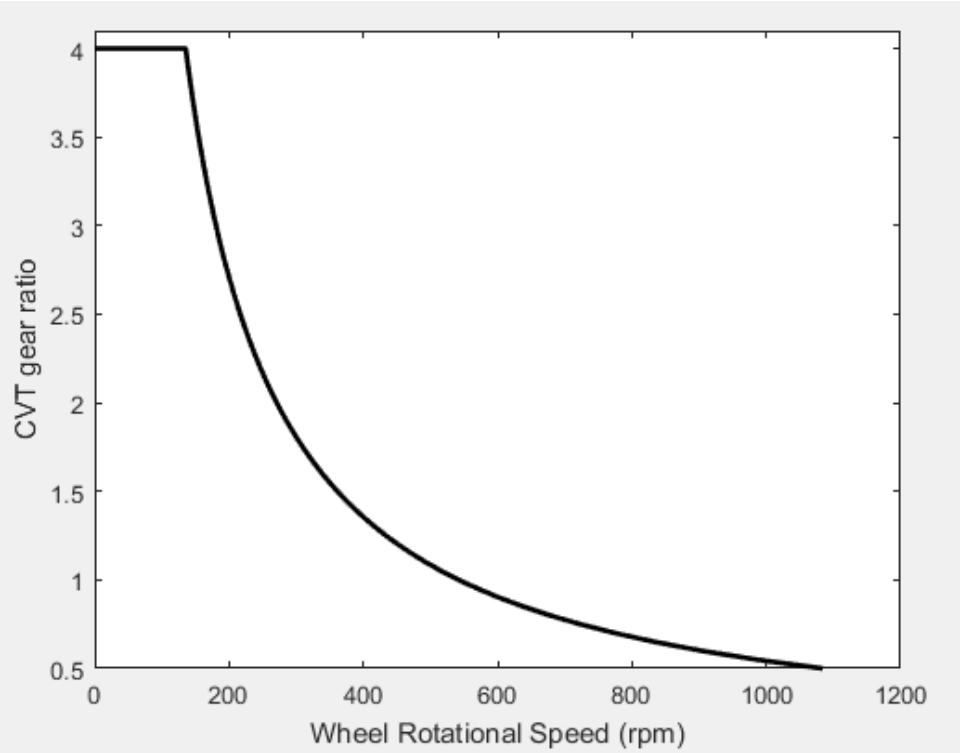


Figure 11: Ideal CVT Shift Curve

Figure 11 shows possibility of a look up table containing the ideal CVT gear ratio for every wheel rotational speed. The lookup table would be implemented in the hardware that controls the CVT.

Pros: The CVT gear ratio would theoretically be actuated to the ideal gear ratio for a given wheel rotational speed. Very simple to employ and understand.

Cons: The lookup table doesn't account for the time it takes to actuate the CVT to a desired gear ratio. The lookup table removes all 'intelligence' that a regular controller provides; the lookup table is only a function of wheel rotational speed and would not act differently if slip occurred between the pulleys and belts.

Neural Network

Neural Networks are a branch of controllers that are based upon artificial intelligence principles. Neural Networks control a given system by 'learning' what control inputs give a desired control output and which control inputs don't give a desired control output and adjusts the control algorithm accordingly. Figure 12 shows what a standard neural network control architecture looks like.

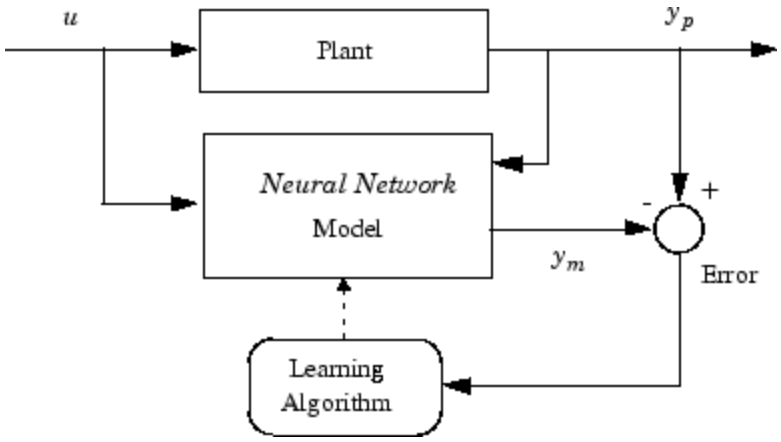


Figure 12: Neural Network Controller Model

Pros: Control system can adopt to new environments; which could be helpful for an off roading environment.

Cons: When mechatronics system turns on it takes time for the neural network to adjust to its environment. Neural networks do not guarantee convergence to a stable controller. Also we would never truly understand how are CVT shifts.

Motor Selection

Stepper Motor

A stepper motor is a type of DC brushless motor that has motion divided to small angle steps. The motor steps around by pulsing alternating electromagnetic coils incrementing a gear with teeth attracted to the magnetic coils. Applying a voltage, the motor applies constant holding torque however the strength of this magnetic field is not very controllable. The motor rotates very slowly but very high torque meaning no reduction will likely be needed in our application. Closed loop control is not needed to get reasonability repeatable position control however, it can lose count if a step is missed. The form of the motor is very flat and does not stick out very much past the CVT case.

Pros: Steppers have small profiles making packing inside the chassis very simple. The high torque produced by a stepper with no reduction need

Cons: Only open loop control, and no torque control. Can lose steps easily if max load exceeded. Constant power draw.

Brushless DC Motor

A brushless DC motor, BLDC, is powered by DC electricity via an inverter or switching power supply which produces an AC electric current to drive each phase of the motor via a closed loop controller. The controller provides pulses of current to the motor windings that control the speed and torque of the motor. This motor typically operates at speed in the thousands of rpm meaning a reduction will be needed to operate at the speeds needed for our application. Because of the

Pros: Very high torque, strong brake force. Smaller in size and packaging. Small time constant.

Cons: Expensive, Needs a designated driver circuit.

Brushed DC Motor

The brushed DC motor is most simple and widely used motor. Closed loop control can be easily implemented to control position, velocity, or torque. The average operating speed of roughly in the thousands of rpm so a reduction will be required. The brushes in the motor are a wear component and must be replaced. However, the short lifetime of the Baja Car that this may not be a concern. These motors are very long and could lead to issues with fitting in with the current rear packaging of the Baja car.

Pros: Simple, inexpensive, versatile. Only power and ground required. Responds to PWM.

Cons: Large Form Factor and a gear reduction needed. Brushes cause sparks and wear.

Linear Magnetic Actuator

An electromagnetic actuator takes electricity and converts it into magnetic force. Magnetic force is used to move the spool or poppet which in turn controls the direction of flow. The actuator is very long and will not fit well in the packaging. Due to the complex coil designs needed, the actuators are expensive.

Pros: Motion already linear. Open loop control.

Cons: Only precise force control. Expensive to purchase. Very long and bad for packaging.

4.2 Analysis

The selection of choosing a microcontroller board running an interpreted language such as MicroPython or a compiled language such as Embedded C/C++ is easy to make after comparing the performance of different boards running different languages. A simple stress test was performed on two different microcontroller boards and a microcomputer board: Arduino Uno Rev3, STM32 Nucleo-64 MB1136revC, and a Raspberry Pi 3 Model B. The Arduino Uno runs an 8-bit processor at 16MHz and runs compiled C++ code. The STM32 Nucleo-64 runs a 32-bit processor at 84MHz and runs MicroPython code. The raspberry Pi 3 Model B runs a stripped version of Debian Linux, and features a 64-bit quad-code ARM Cortex A53 processor running at 1200MHz, and runs native Python 3 code. The stress test code written in Embedded C++ is attached to this document in Appendix I. The code in Python and MicroPython was ported directly from the C++ code. The stress test generates an array of 850 random integers and sorts them using a selection sort algorithm. This algorithm has a time complexity of $O(n^2)$ and performs 360,825 comparisons during the test. The results of the time taken for the stress test of each platform is tabulated below in Table 5.

Table 5: Comparison of processing time to run stress test on different platforms

	Arduino Uno	STM32 Nucleo-64	Raspberry Pi 3 Model B
Time to Complete	3.89s	15.51s	0.67s
Comparisons/Second	92757cmp/s	23264cmp/s	538545cmp/s
Comparisons/Second/Clock	5797cmp/MHz-s	277cmp/MHz-s	449cmp/MHz-s

From analyzing the stress-test data from the three boards, it is obvious that an embedded C type controller will provide far-superior performance compared to a board running an interpreted language.

The overall design process for our team is to derive a mathematical model of the eCVT, design a control system around the eCVT model and implement the controller using a mechatronic system on the physical eCVT.

A mathematical model of the vehicle's longitudinal dynamics was developed and used to evaluate the eCVT's dynamic performance when it is simulated over a variety of road loads, mass and geometric properties of the eCVT. Derivation of vehicle longitudinal dynamics can be found in Appendix G. The primary objective of the eCVT model is to make the model robust; with very little altercation, to simulate the effects of changes to mass, geometric properties, and change the road load. We also used the model to determine an ideal shift curve to stay within the maximum power band and test and compare control algorithms. The mathematical model will be thoroughly documented so future teams can use the model with reliability and ease.

From a mechatronics standpoint, the purpose of a vehicle dynamics model is to design a control system that produces nominal control gains that will be used to tune the physical system. Since the primary purpose of the model is to produce nominal control gains, the benefit to produce a sophisticated model is not worth the time and effort it would take to build the model. As a result, assumptions are employed to the vehicle dynamics model that simplify the model while still capturing the vehicle's primary dynamics. Specifically, longitudinal dynamics of the Baja vehicle are analyzed to better understand how the CVT affects performance characteristics like time-to-top speed, uphill drive, steady state drive and to design a control system that changes the CVT's gear ratio based upon the current state of the system. The vehicle dynamics model will be thoroughly documented so future teams can use the model with reliability and ease to further develop the controller.

As seen in Appendix F, the longitudinal dynamics of the Baja car are simulated in the MATLAB/Simulink environment. Different cases were simulated, an acceleration run and a sudden hill, to see not only the controller's response to both up and down shifting. Also gathered from our sister senior project group, was the maximum clamping force. Finally, the efficiency of the mechanism was considered to develop requirements for the motors. There is a difference in the upshift and down shift power requirements due to the lead screw. In upshift, the primary is

compressing the belt causing a larger power draw. On the other hand, the secondary is unclamping the belt, so it helps drive it reducing the power draw. For downshifting, the opposite happens and the secondary draws more power. The result of all this was speed and torque requirements of the gearbox of the motor. The plot for the acceleration case is shown in Figure 13: Primary motor requirments in upshift and Figure 14: Secondary motor requirements in back shift were used to select the final motor. Calculations were done in metric because the motor company provided technical details in metric.

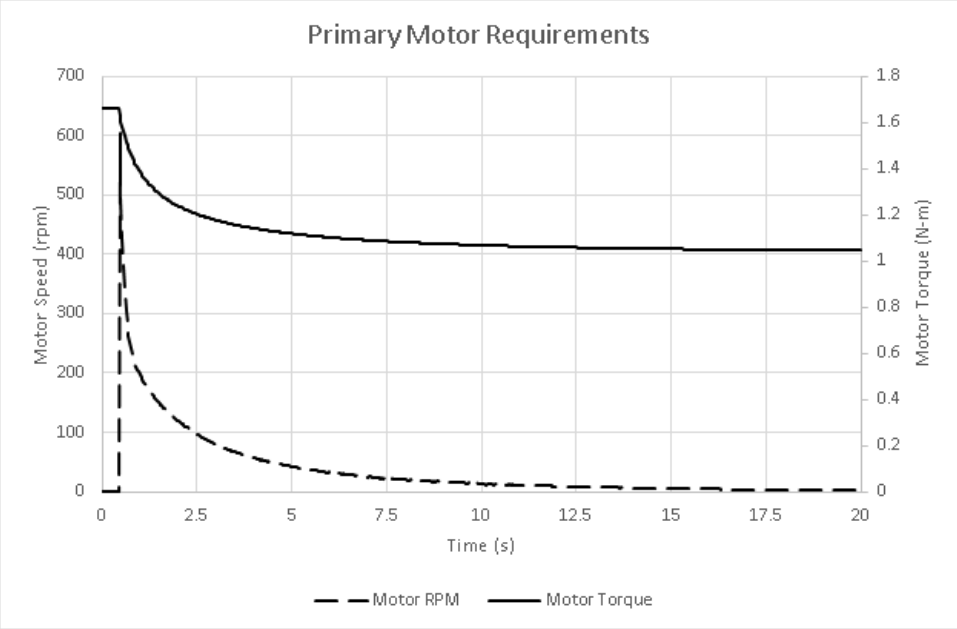


Figure 13: Primary motor requirments in upshift

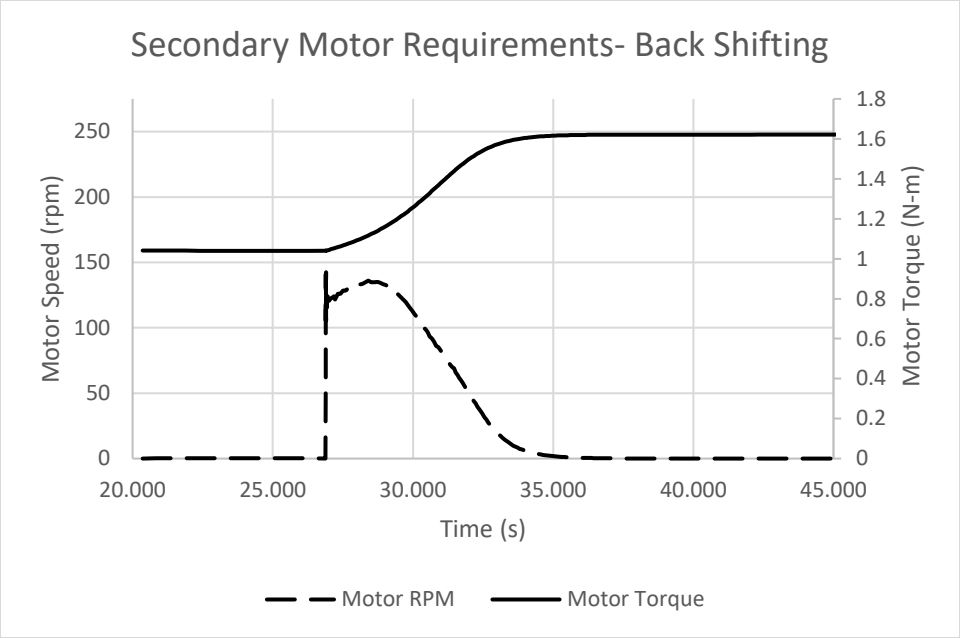


Figure 14: Secondary motor requirements in back shift

4.3 Concept Selection Development & Results

After running an in-depth analysis of different programming languages and different microcontrollers, a decision matrix was used to determine the optimal setup. This is shown below in Table 6.

Table 6: Decision Matrix for Microcontroller Selection

Controller	Ease of Use (0-5)	Documen-tation (0-5)	Language (0-2)	Performance (0-5)	Auxilary Features (0-5)	Total Score (0-18)
Arduino Uno (AtMega 328)	5	5	1	3	2	16
Arduino Due (Atmel SAM3X8E)	5	3	1	5	4	18
PyBoard	5	4	1	3	4	17
Raspberry Pi	3	4	2	2	5	16
Teensy 3.5 (ARM Cortex M4)	**5**	**4**	**1**	**5**	**5**	**20**

When considering the five listed control algorithms for the eCVT FSFB and neural networks can be ruled out without rigorous analysis because of their low score in the decision matrix as seen in Table 7. FSFB is not ideal for the eCVT because the system is a first-order single-input single-output system. FSFB dominates when several states are needed to be controlled simultaneously. A neural network is also not ideal for the eCVT primarily because of the competition environment the control algorithm would be acting in. For a Baja competition all components of the vehicle, including the control system, need to be finely tuned and ready to perform their best immediately. It would not be ideal for the control system to be learning its environment during events such as acceleration run. It needs to be made sure that the control algorithm is deterministic during any event.

Table 7: Decision Matrix for Control Algorithm that decides when to change the gear ratio for the eCVT. The five listed controllers use different methods to determine when to change the gear ratio of the eCVT, based on engine rpm

Controller	Ability to tune (0-5)	Ability to design (0-3)	System Fit (0-5)	Robustness (0-3)	Ability to program (0-2)	Total Score (0-18)
PI controller	5	3	3	1	2	14
Fuzzy Logic	2	1	5	2	2	12
FSFB	4	2	0	1	2	9
Look Up Table	5	3	4	0	2	14
Neural Network	2	0	4	1	0	7

With the remaining three control algorithms, a basic longitudinal acceleration run was simulated in the MATLAB/Simulink environment to compare each controller's performance. The PI controller and fuzzy logic controller are both compared to the control produced from the ideal shift curve look up table. As seen in Figure 15, PI control and the look up table have almost identical performance characteristics.

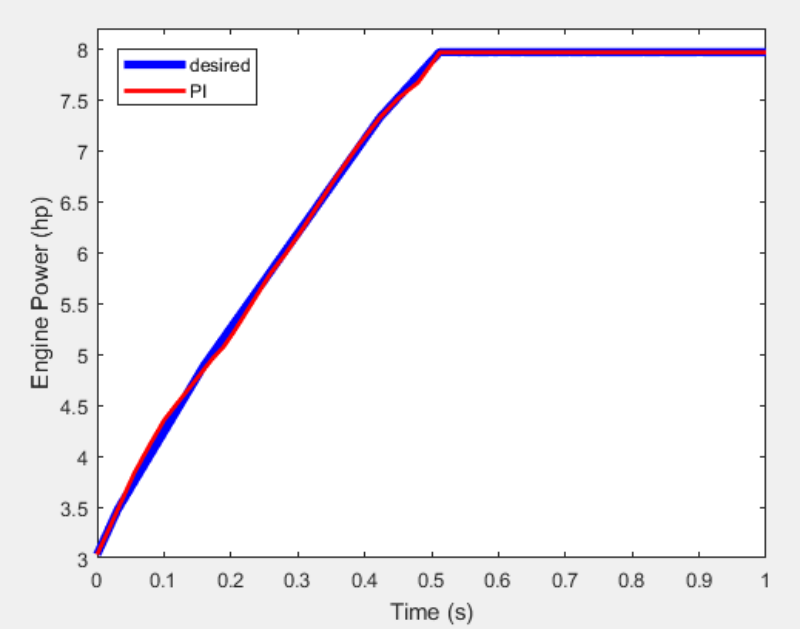


Figure 15: Proportional + Integral controller vs. Look-up Table Performance

The Fuzzy Logic controller was designed based upon the membership functions and fuzzy rules defined in Appendix H. As seen in Figure 16, the fuzzy logic controller does not perform well relative to the look up table.

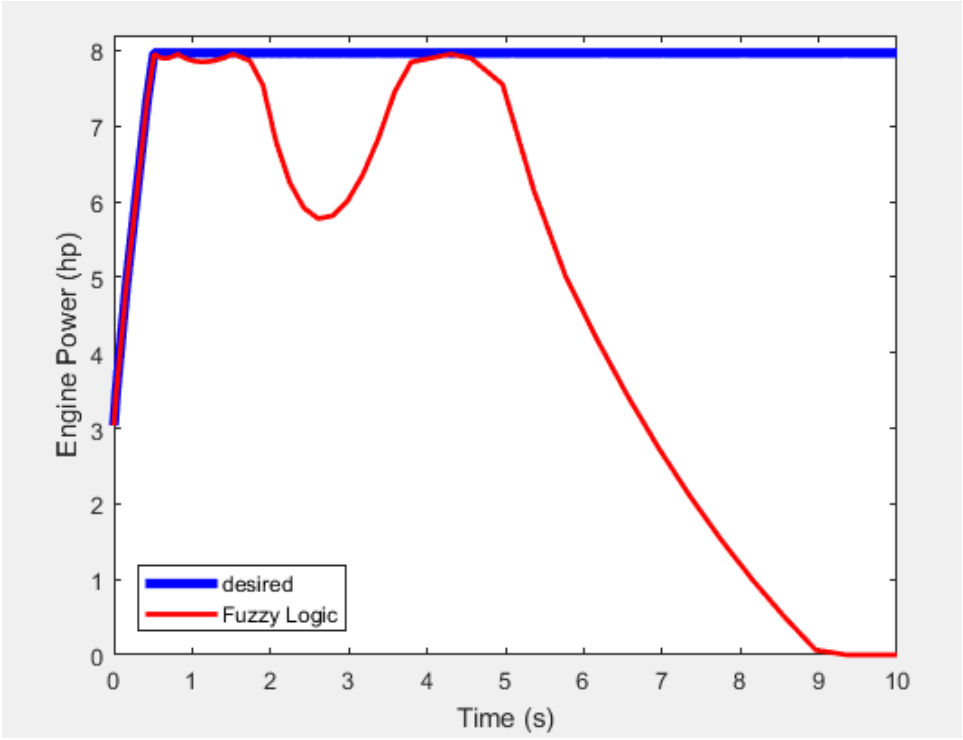


Figure 16: Fuzzy Controller Performance vs. Look-up Table Performance.

Although the fuzzy logic controller can be further tuned to produce a better response it would not be worth the time because of the better response the PI controller gives.

Based from the research, decision matrix and simulation results it is seen that PI control is the best control algorithm to use for the eCVT. The reason PI control is the best choice for the eCVT is that PI control has almost identical performance to the look up table and is more robust than the look up table. The main problem with the look up table is that there is no 'intelligence' associated with its design; the look up table is precomputed assuming no slip between the belt and pulleys. As a result of the look up table's inherit design, it is clear that the look up table would not perform well in a dynamic environment where there is slip between the belt and pulley. PI control on the other hand could be designed to adjust when there is slip present in the eCVT.

To select an actuator, a decision matrix was used shown in table 9. The rankings were based our previous knowledge, research and recommendations from Professors. Our results show that the actuator we will proceed to spec will be a Brushed DC with a reduction. This will be the easiest to implement given the time constraints while giving us reasonable performance.

Table 8: Decision Matrix for Motor Selection

Motor	Packaging (0-3)	Position Control (0-5)	Torque Control (0-5)	Controller Cost (0-2)	Additional Reduction (0-3)	Total Score (0-18)
Brushed DC Motor	1	5	5	2	0	13
Brushless DC Motor	2	5	5	0	0	12
Stepper Motor	3	4	0	2	1	10
Magnetic Actuator	0	2	4	0	3	9

4.4 Detailed Description of Selected Concept

The schematic for the electronic control system is dependent upon the microcontroller used and the number of and placement of the sensors that we use. A microcontroller with sufficient general purpose input and output pins for each sensor will allow each sensor to be wired (and therefore accessed) in parallel. With this configuration, each encoder, if configured in quadrature configuration, will correspond to two general input pins. If we do not implement the quadrature configuration, then each encoder will only require one general input pin. Each actuator will accordingly have some position/velocity sensor, and therefore will each need an additional one or two input pins. Each actuator will also need a limit switch to serve as the zero datum if we are not using an absolute position encoder. This yields a total of ten independent input pins required for the four encoders in quadrature configuration with two contact limit switches. The ideal actuator driver will only require two pins per actuator – one to act as the enable pin and the other to send a

pulse-width modulated signal to control the speed of the actuator. Since our final design has two different actuators, then we will need two general output pins and two pulse-width modulation (analog) pins. Lastly, a thermocouple sensor will require one analog input pin to record the temperature inside the transmission, and the SDA and SLC pins can be used to control the heads-up display in the driver's cockpit using the inter-integrated circuit protocol. Any additional sensors or controller input/output can be wired to any remaining pins or can be connected in parallel to the inter-integrated circuit bus. A schematic showing our initial electronic control system is shown below in Figure 17. For an updated circuit diagram, since PDR, check Appendix O – Circuit Schematic Blow-up.

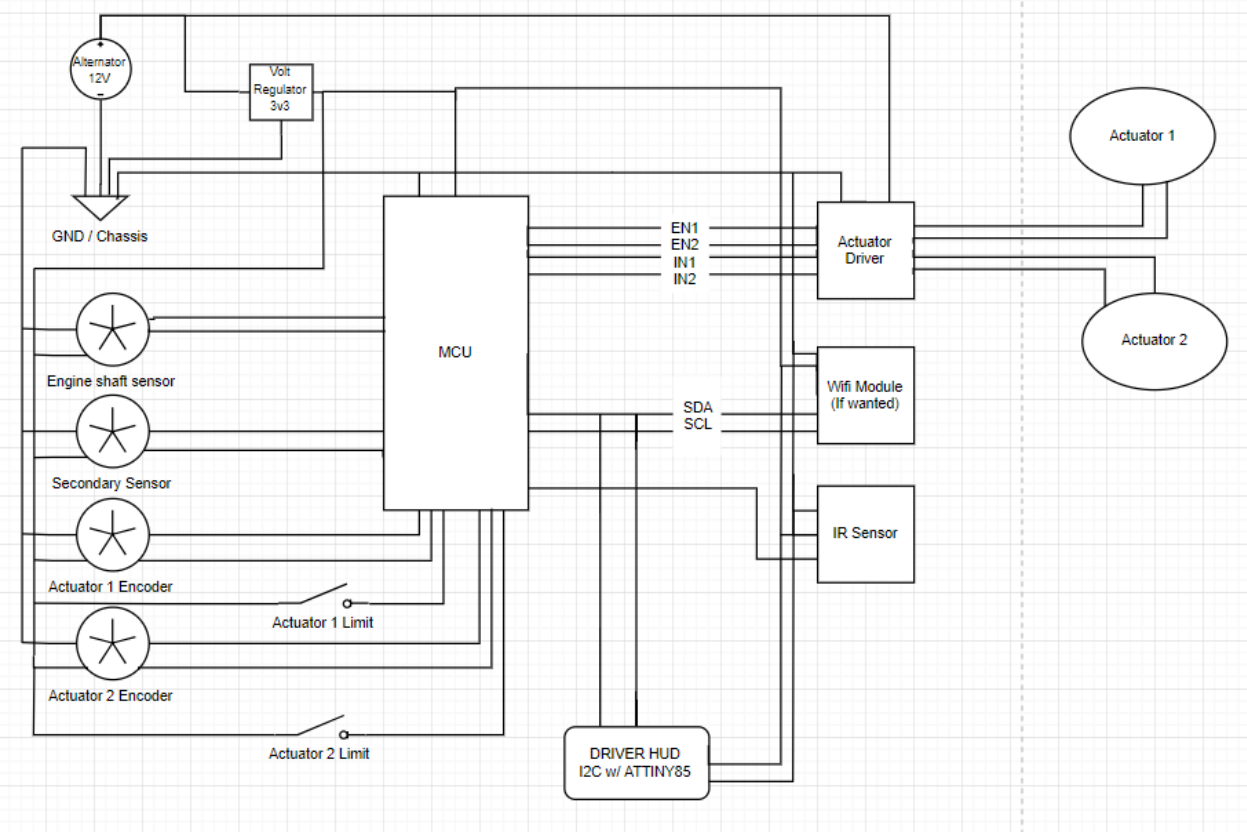


Figure 17: Circuit Schematic using direct digital GPIO pins.

The software for the control circuity will be implemented using cooperative multitasking to collect and interpret sensor data and to drive the actuators. All collected data will be logged for debugging purposes and for performance data for the rest of the Baja team. For PDR our task structure for the controller was broken into eight distinct tasks: one task for each of the actuators and the actuator encoder, a temperature monitoring task, a throttle position monitoring task, a sequencer "Mastermind" task, and a data logger task. The actuator tasks are responsible for maintaining the position of the actuator as specified by the sequencing "Mastermind" task. These tasks have a priority of twice that of the encoder task, but have a timing of ten times that of the encoder tasks, allowing the encoders to collect ten points worth of data before the actuator driver task can react

– preventing instability. The task diagram is shown below in Figure 18. An updated task diagram can be found in Section 5.4

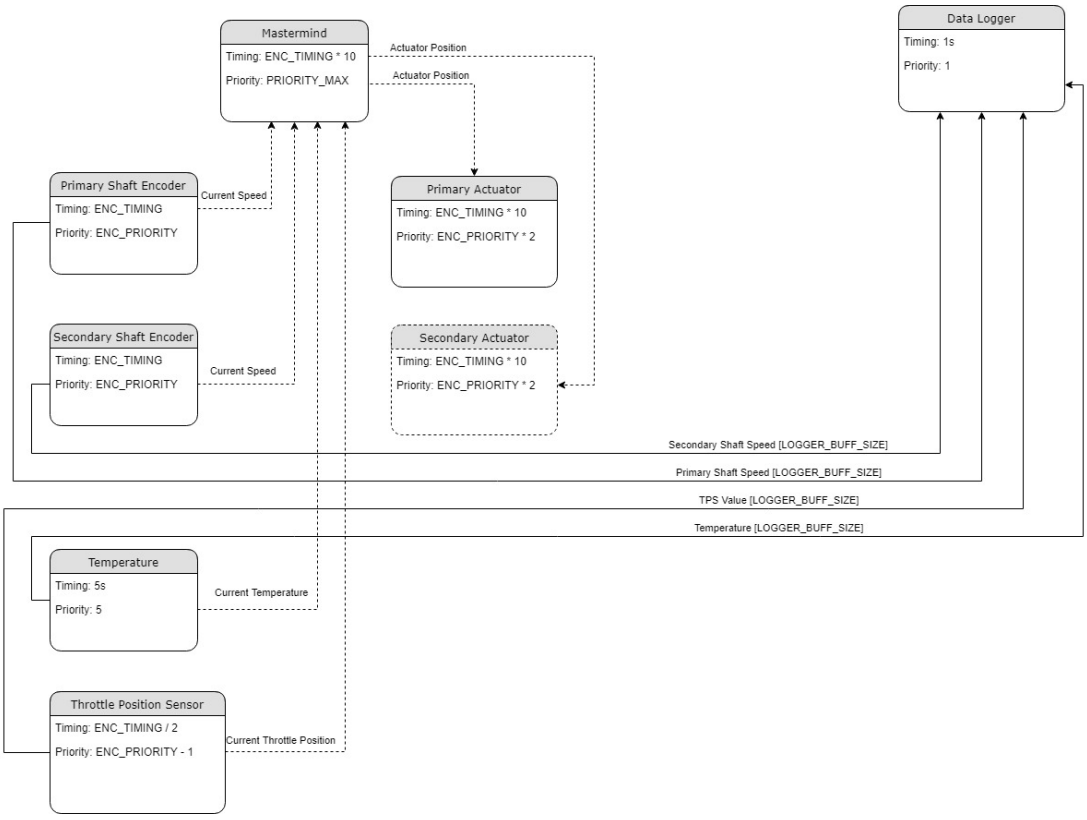


Figure 18: Task Diagram for Controller Software

The values of ENC_TIMING will be determined by measuring the open-loop time constant of the actuator moving to a predetermined position given as a step, τ. The timing for the actuator will be one-tenth this time constant, and accordingly, the numeric value for ENC_TIMING as shown in the diagram above will be equal to one-hundred times the actuator time constant τ. Priority will be determined by setting the sequencer "Mastermind" task to an arbitrary large priority and assessing the priorities of the lesser tasks as follows such that the priority order specified in the task diagram is maintained. The cooperative multitasking system will order priorities in accordance with assigned priorities in the task diagram above in descending order, where the highest priority number takes precedence over a lower priority number.

The actuators will each need their own encoder or position feedback device, and each pulley will need a hall effect or other position sensor in near proximity to the outer radii of the pulley. A computer aided design mockup of the sensor locations is shown below in Figure 19.

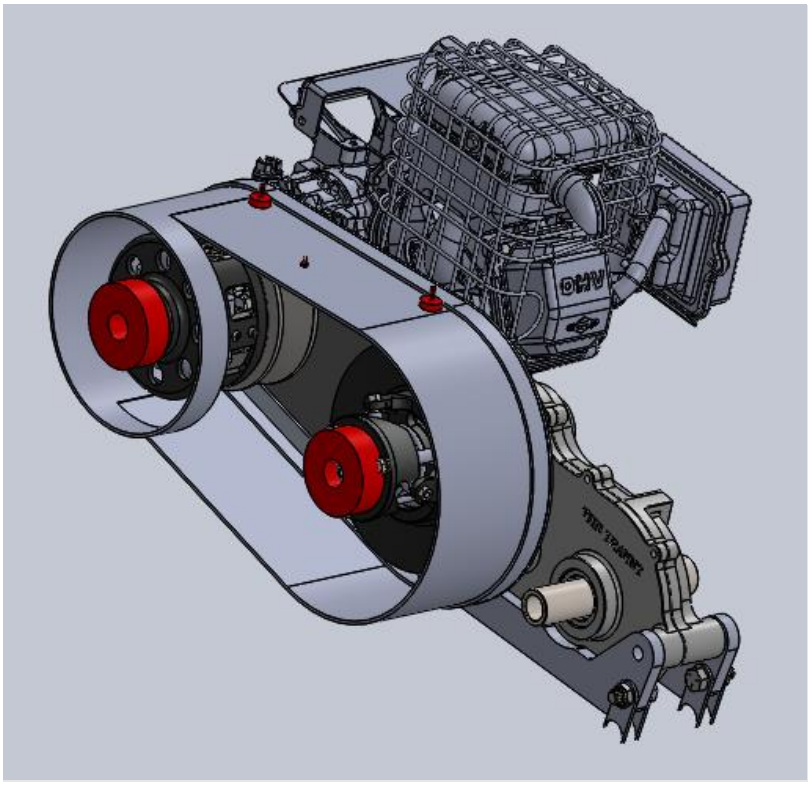


Figure 19: CAD Mockup of Sensor Placement

4.5 Concept Functionality

To tune and control the functionality of the transmission, a graphical user interface application is to be developed that will allow quick and easy access to the recorded data and a quick and easy method to change performance variables. Such performance variables that can be tweaked include actuator gains, i.e. proportional, integral, and derivative gains for actuator drivers. GPIO and I2C pins and protocols can be directly changed and addressed without the need to modify controller source code. Furthermore, the graphical user interface can contain a serial read-eval-print loop to be used during controller development, debugging, and tuning. The purpose of the graphical user interface is to allow Baja SAE members the ability to tune the transmission in the least amount of time possible. A mockup of the graphical user interface is shown below in Figure 7. The graphical user interface software will be written in the Python programming language to allow for cross-platform compatibility and will communicate with the microcontroller through a serial interface or through a wireless fidelity module attached through the I2C protocols.

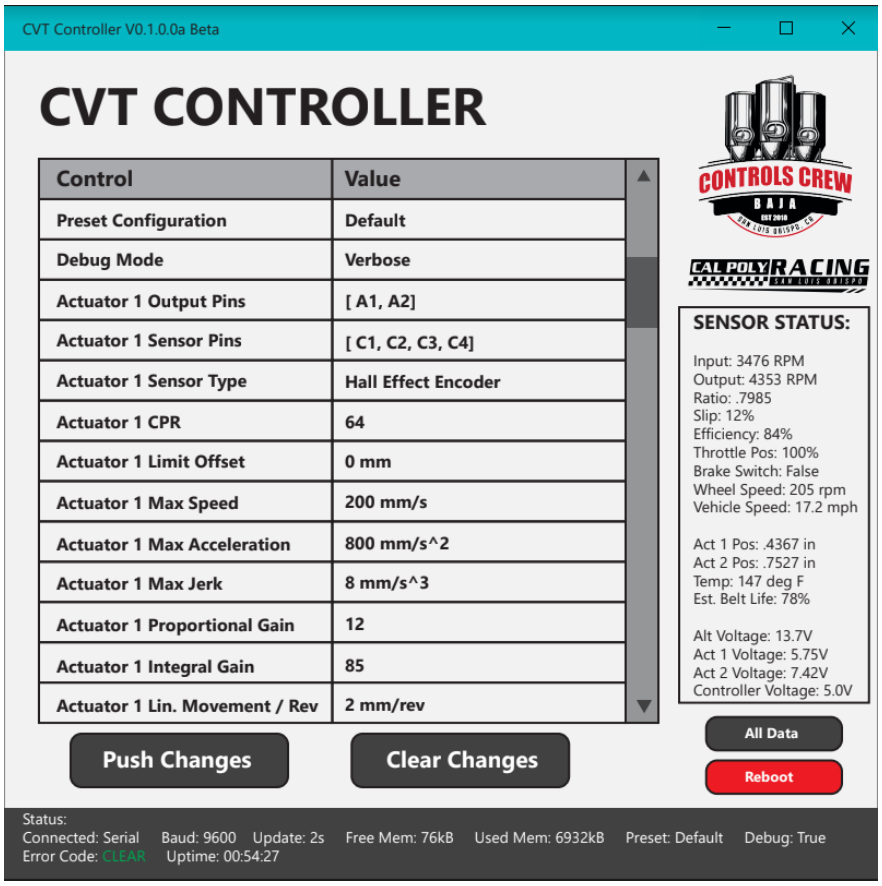


Figure 20: Graphical User Interface Mockup for CVT Tuning

4.6 Challenges, Unknowns, and Risks

The challenges associated with deriving a control algorithm for the CVT is to understand the non-linearities of belt slip. Because there may not be a perfect mathematical model for belt slip, the system will need to compensate and behave in real-time to account for slip. While this could be accounted for using a Fuzzy controller, as analyzed previously, the performance of the system would be deteriorated compared to the performance of a system using a proportional + integral controller. The rest of the control algorithm is straight-forward to model and develop. The performance of the controller will be dependent upon the accuracy of the model, however, through tuning small errors can be eliminated. Another issue to contend with could be belt wear. Our current concept assumes a certain belt length to be able to predict the correct locations of the sheaves. However if the belt wears too much, it could change this relationship unpredictably. To combat this, we were planning on having a using ratio of engine to wheel speed to recalibrate. Finally, how this system will deal with the wet conditions, which will happen in off-road conditions, could seriously impact the performance.

5 Final Design

5.1 Abstract Overview and Functionality

This project can be broken down, like many mechatronic systems, into hardware and software. The software consists of two PI position controllers for the controlling the location of the sheaves with a master PI controller dictating the ratio needed based on engine speed. The hardware need for the control of the CVT can be broken down into six categories: power supply, controller circuit board, user-interface, sensors, motors, and packaging.

The power supply system will consist of an alternator and regulator that complies with the SAE Baja rules. These components are embedded into the engine and should be plug-and-play without requiring any configuration or setup after installation. We have selected the largest alternator allowed, with a max output of 12V at 20 A, to give us comfortable breathing room with our predicted 6.78 A of motor draw.

The controller circuit board system consists of the MCU board, a Teensy 3.5, an H-Bridge module, and a dedicated BUZ10 metal-oxide-semiconductor field-effect transistor for each engine and wheel hall-effect sensor on the vehicle. Each individual component on the Teensy 3.5 and dual H-bridge module will need to be soldered onto a PCB after prototyping the control board, it will be determined if the protoboard is unsuitable to use in production.

The user-interface system consists of an indicator light to alert the driver of an overheated CVT, and a variable resistor or potentiometer to allow for manual selection of gear ratio during testing and debugging (and possibly in production if driver desired manual control option).

The sensor system consists of rotary encoders which will be configured and installed onto the pulley drive motors. These encoders will work in conjunction with limit (or proximity) switches to provide absolute positioning after zeroing. Hall Effect sensors on the vehicles engine shaft, CVT output shaft, and front and rear wheels will provide angular velocity measurements. An IR sensor will be placed inside the CVT case to monitor temperature.

The motor system consists of two 12V, 70 watt brushed DC motor pre-configured with a 28:1 gear reduction and 32 count quadrature encoder. These motors are supplied by Maxon Motors, a very high quality supplier.

The mounting system consists of a board enclosure, race quality connectors, and direct mounting to the mechanical system. The board enclosure is designed to be mostly 3D printed, with a laser cut rubber seal, and laser cut plastic sheet top. The body will contain threaded inserts to screw the top on. This simple case will be easy to manufacture, allowing for more time to be allocated to controls tuning. The sensors will be mechanically fastened to the mechanical system's case. The hall effect sensors within the case will be attached to the backing plate, and the IR sensor will be on the band between the sheaves. The engine speed and rear wheel speed sensors already exist on the Baja 2018 car we will be using for testing.

5.2 Motors

The motors were chosen based on the speed and torque requirements for back shifting and acceleration detailed earlier. Using Maxon Motors supplied dyno curves as seen in Figure 21, the size of the motor and reduction of the gearbox were determined. Motor and gear box detailed data sheets are included in Appendix P – Motor Data Sheet and Appendix Q – Gear Box Data Sheet

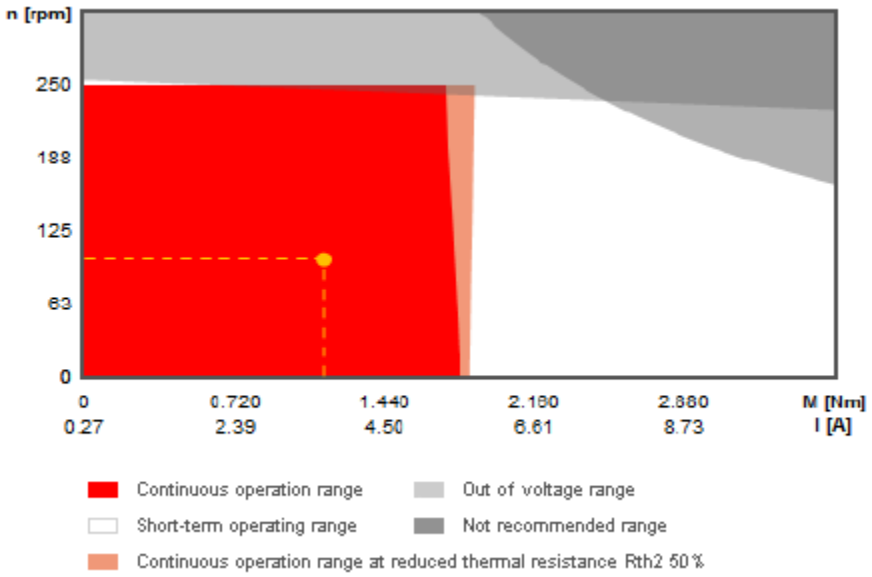


Figure 21: Maxon DX-32L with 28:1 gearbox Performance Plot

Using all of the simulation requirements plots in Figure 22 and Figure 23 were generated showing that the motor would be operating within its continuous operation range. There is a 0.2 second period of time when the motor will be out of its RPM range. However, it was deemed by the team that this was acceptable as there was a slight issue with the simulation stability at that moment.

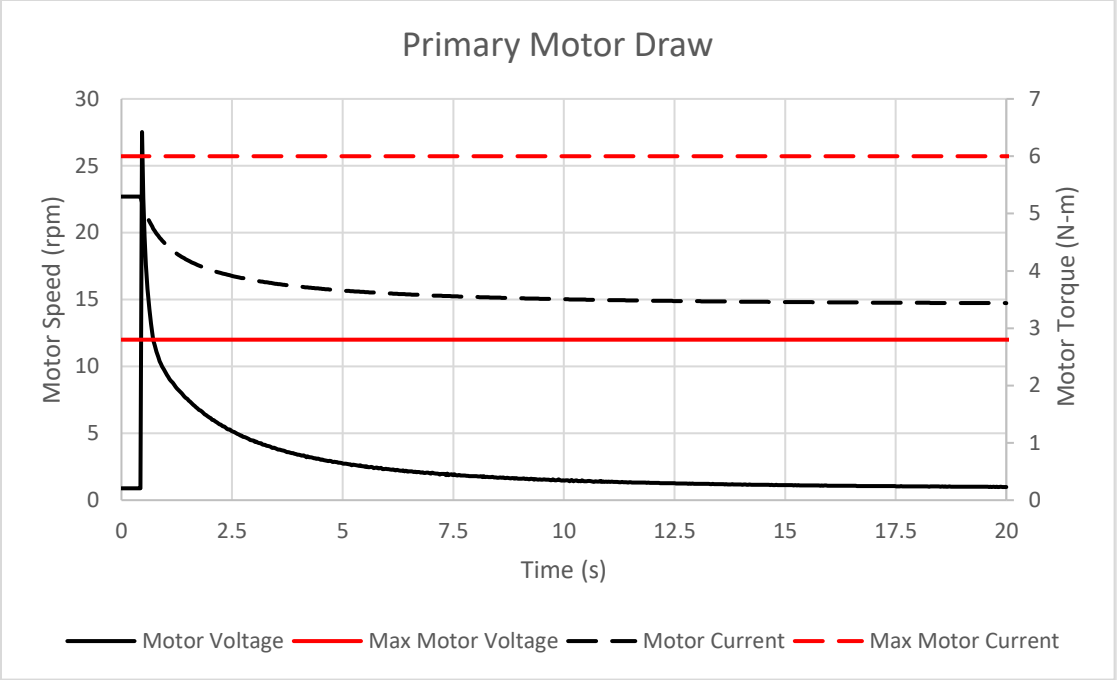


Figure 22: Primary Motor Draw for Acceleration

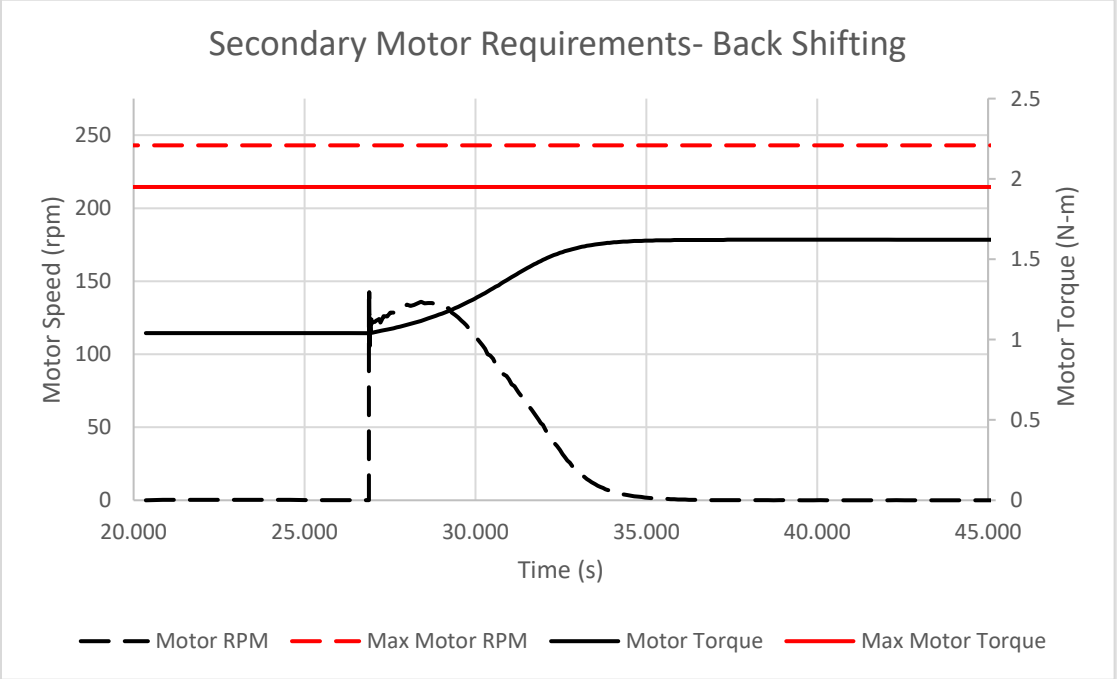


Figure 23: Secondary Motor Draw for Backshifting

To summarize the predicted performance and requirements of the motors throughout these cases, was develop. As can be seen the motor meets all of the requirements for performance and is below our limits for current draw and requirement for power draw. There is likely to be a difference between the actual starting draw because friction will be higher at starting but might be lower at steady state because of the holding the acme screw provides.

Table 9: Sumamary of Motor Draw

Case	Acceleration (20s)	Backshifting (25s)
Max Primary Voltage (V)	27.5	9.0
Max Secondary Voltage (V)	12.2	7.2
Max Primary Current (A)	5.29	1.58
Max Secondary Current (A)	0.51	5.18
Max Total Current (A)	5.52	6.76
Avg Primary Power (w)	32.2	2.0
Avg Secondary Power (w)	1.0	12.3
Avg Power Loss (%)	0.5	0.2

5.3 Electrical System

The electrical system for the final design will follow the circuit schematic in Figure 24. It is also available in a larger blow-up at the end of his document in Appendix O – Circuit Schematic Blow-up. This schematic was generated in EAGLE and can easily be implemented on a custom PCB in the future if necessary. The circuit schematic for the Teensy controller board was included above under the background research section in Figure 4.

Each component of the electronic system has been sized and selected to be well within their specifications with respect to voltage, signal quality and process ability, and heat dissipation. For instance, the MCU was chosen specifically for its 5V pin tolerance despite its 3.3V logic level. The datasheets of each component were carefully read and understood to ensure that all inputs and outputs that we expect are in compliance with the allowable input and output range.

Software development will follow test-driven development (TDD) procedures to improve the probability of success and to guarantee that during each feature addition, update, or patch that the software remains in compliance with our test specifications. The C standard library function "assert()" can be used along with the custom made, cross-platform compatible, C89 unit test library

as documented in Appendix N – Unit Test Framework. This framework can easily be implemented to debug and test ARM code, and will remain the standard to guarantee that code-patches and updates comply with existing code infrastructure. As additional control algorithms are implemented, through the unit test framework and TDD we can guarantee that existing algorithms and system stability remain unbroken. Additional tools such as kcachegrind and valgrind will be used to detect memory leaks and errors after gcc compilation to guarantee that our control algorithms are capable of operating for extended periods of time. Valgrind will be used during simulations exceeding the typical lifetime of a car operating cycle.

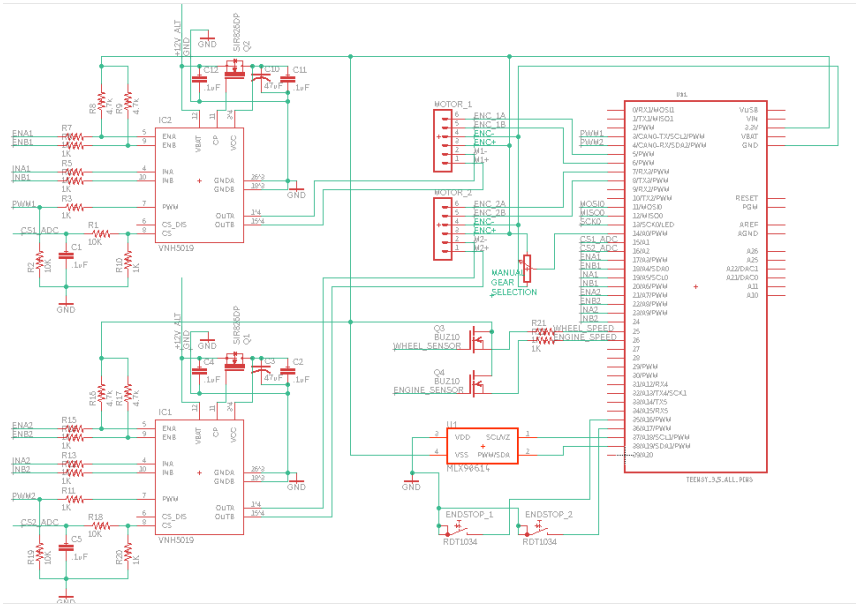


Figure 24: Circuit Diagram

5.4 Software Algorithms

After making the decision to use PI control to actuate both motors, a detailed control system design was made which would account for both CVT and actuator dynamics, as seen in Figures 25, 26. As seen in Figure 25, the lookup table maps a given gear ratio to how much each motor has to move to obtain that given gear ratio. Although the design currently uses a lookup table, we have yet to decide whether we will be using the lookup table or hard code the equations that provide gear ratio to motor position mapping. Deciding which method is more computationally efficient will get hatched out during on-car testing, which has not been completed yet due to conflicts with the mechanical design and Baja teams. The desired motor positions will become the set point for both motors' control algorithms, as seen in Figure 26.

A primary concern with controlling both motors is the possibility that the position controller would go unstable, causing the motor to continuously output maximum effort, breaking the mechanical system. To avoid this issue a saturator is included in both the master and motor control so that each motors' travel is bounded through software.

Another concern we have with the motor controller is that position control will not output enough torque to maintain contact between the belt and sheaves. If this concern becomes an issue then the

motor controller can be switched to Full-State-Feedback (FSFB). FSFB will allow for the motor torque and position to be simultaneously controlled. The reason why we did not default to FSFB is because it is more complicated to tune the control algorithm. Therefore, FSFB will only be implemented if testing deems it necessary.

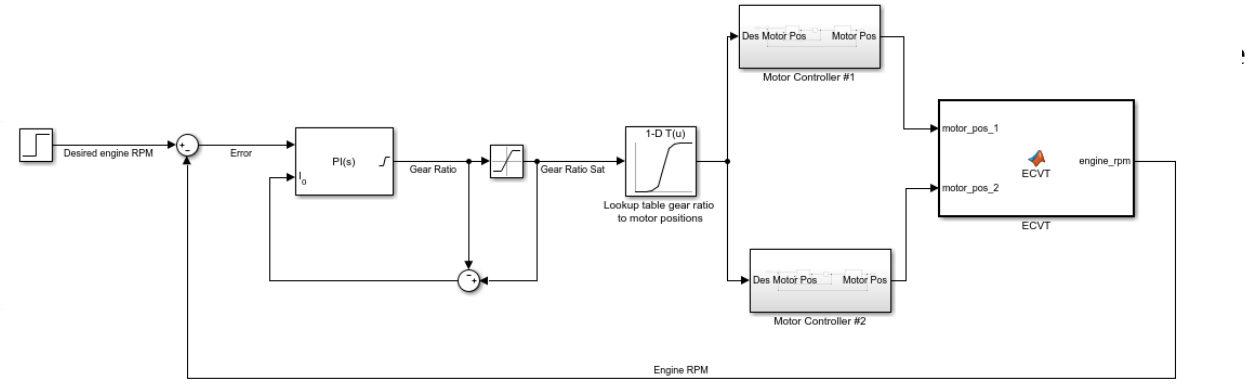


Figure 25: Master Control Algorithm

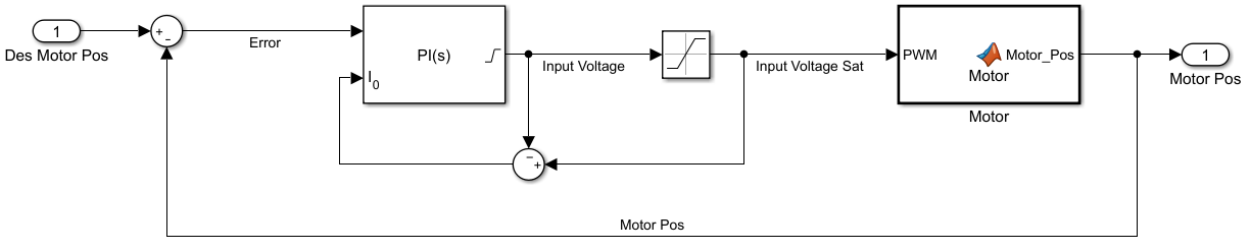


Figure 25: Motor Control Algorithm

After designing a more detailed control algorithm, updating the software design, based upon changes since PDR, was the next necessary step to executing a proper control system. As seen in Figure 26, the task diagram outlines how the control system will be designed in software. Since PDR we have changed the number of tasks from 8 tasks to 4 tasks. The reason for this change is because we decided that we will write classes for each encoder and motor and use those classes inside a generic control task. Using classes instead of tasks will not affect timing performances because our microcontroller has internal hardware that will keep track of encoder ticks. 1000 Hz was chosen for the timing in each motor task because each motor controller should be running 5-10 times faster than the mechanical time constant of the system. These tasks need to run 5-10 times faster than the mechanical time constant of the CVT in order to close the loop fast enough for ideal control. The timing requirements for the master control and user interface were estimated based upon the motor control tasks. The idea is that the master control, emulating **Error! Reference source not found.** needs to run slower than the motor control tasks because the motors will not be able to move to their desired positions in time if the master control loop is running faster than the motor control loop. In the case, during testing, when the timing requirements for any of these tasks is not sufficient, we can easily alter the timing without any serious impact on performance.

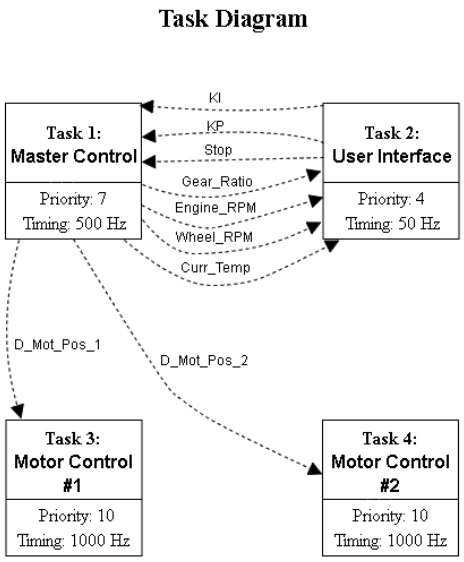


Figure 26: Task Diagram Used to Set Priorities

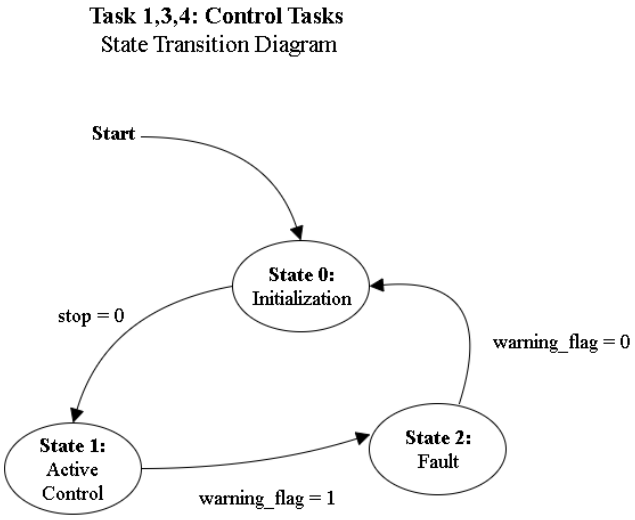


Figure 27: Controls State Transition Diagram

5.5 Safety, Maintenance, and Repair

With respect to the mechatronic system of the final product, many layers of redundancy and safeguards will be put in place to ensure the safety and longevity of the electrical and electro-mechanical systems. Electrical components are sized to withstand voltages and currents higher than their rated values. The MCU runs with at a 3.3V logic level, yet all pins are tolerant up to 5V. As per the schematic shown in background research under Figure 4, the MCU voltage sources are cleaned with various decoupling capacitors and ferrite beads. A 500mA fuse protects against current spikes and possible danger caused by a short. The vehicle's alternator is sized to provide

up to 20 amps of current, yet the maximum theoretical current draw of our control system is 6A per motor, for a total of 12A (plus a few hundred milliamps of current for the control system and H-bridge losses.) An 18 amp fuse will be used to protect the alternator. The H-bridge module which will run the motors is driven by two ST VNH5019 ICs and feature an operating range between 5V and 24V. Each IC allows a continuous current draw of 12A, with a peak allowance of 30A. These H-bridge modules also contain pull-up and protection resistors and a reverse-battery protection metal-oxide-semiconductor field-effect transistor. Motor voltage inputs are each cleaned with two series 47uF electrolytic capacitors to filter low-frequency noise and a single 0.1uF ceramic capacitor to filter high-frequency noise.

Software algorithms will be used to determine the state of the mechatronic system in real-time. Input from engine speed, front and rear wheel speed, throttle position, pulley positions and speeds, and transmission temperature will be implemented. In the case of overheating, the driver will be alerted through a visible alarm light in the cockpit. Current sensing capabilities delivered through the VNH5019's allow for software current-limiting. In the event of a sensor failure, routines can be added to the control algorithm to account for missing data, and may be able to provide a "limp home" fallback mode.

Since the components we have selected have been so heavily over-sized, there is very little risk at damaging the components electrically, provided the components have sufficient cooling. All microelectronics are protected in an IP68 rated case with vibration dampers to protect against physical damage. In the case of damage, electronics can be easily replaced from a PCB by de-soldering the damaged component and re-soldering a replacement. Motors can be easily mounted and unmounted using hex fasteners as per the mechanical design specifications. No regular maintenance should need to be performed on the mechatronic systems.

The safety risks associated with controlling an electronic continually variable transmission are rather small and rely upon the mechanical integrity of the system being controlled. Appendix K includes a safety hazard checklist that notes the risks associated with the transmission. The control hardware will be enclosed and protected pursuant to IP67 or better protection to protect the system from environmental factors. All electronic components and solder joints that may contain hazardous materials, especially materials known to cause cancer in the state of California, will be enclosed and protected from accidental touch. See the table at the end of Appendix K for FMEA.

5.6 Post-CDR Changes

Following CDR, the changes to our project were mainly a limit in scope due to compressed timeline following the completion of the mechanical system. This limit to our scope included limiting our tuning to an acceleration run and removing the GUI deliverable due to the impeding change in controls by the new senior project group. As far as hardware selection, our initial selections remain the same, and as far as the software, our initial algorithm methodology also goes unchanged. Minor updates to the BOM can be seen in Appendix L – Bill of Materials.

6 Manufacturing

Our manufacturing includes the manufacturing of a prototype control board, a printed circuit board, the enclosure for the circuit board, and the writing of a preliminary control algorithm with embedded redundancies for the Baja team's use.

The manufacturing of our prototype board began with a breadboard, Teensy 3.5 MCU, and Pololu H-Bridge being purchased. The details of these items can be seen in Appendix L – Bill of Materials. The pinout of the Teensy 3.5, shown in Figure 28, was studied to determine the pin configuration for the motor driver, encoders, and hall effect sensors. Because each of the two hall effect sensors have different steady-state operating frequencies, different libraries will need to be used in order to sample the hall effect sensors. The Teensy FreqCount library is capable of sampling between 1kHz and 8 MHz using a hardware implementation, while the FreqMeasure library is capable of sampling frequencies between 0.1Hz and 1kHz using an interrupt-based implementation. FreqCount can only measure using pin 13, but its implementation does not block any other pins such as on lower-performance Teensy boards (PRJC).

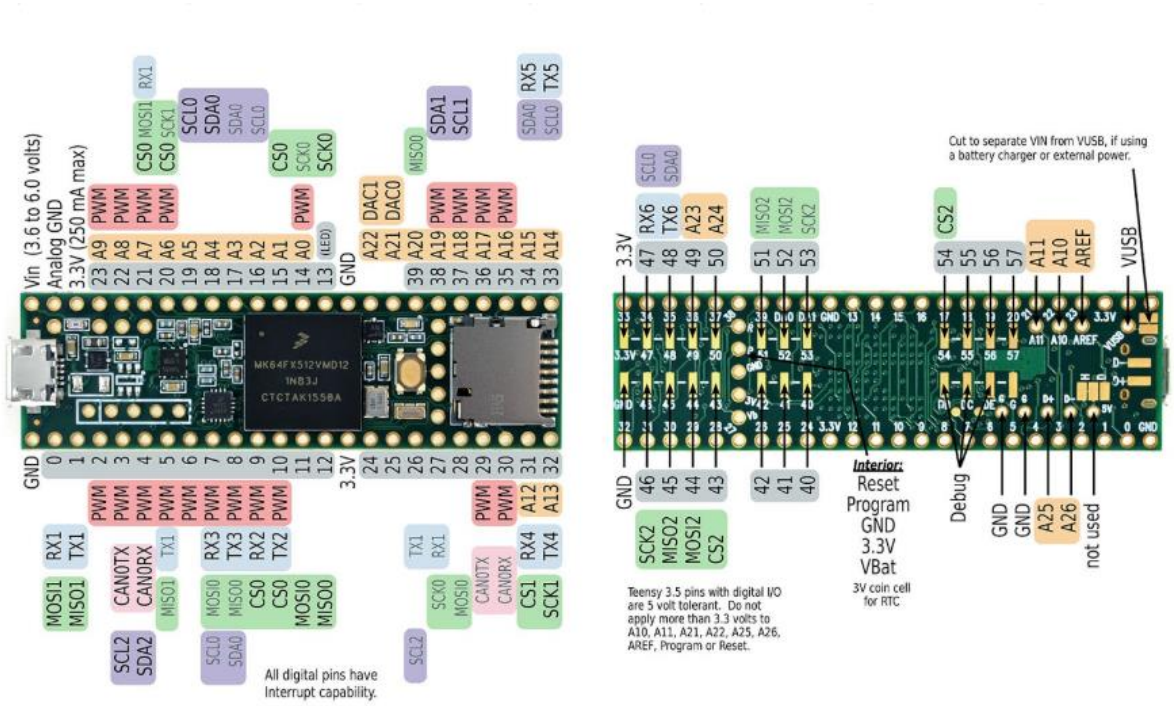


Figure 28. Pinout of Teensy 3.5.

UART 1 and 2 were reserved for any communication needs which may arise in future implementations. UART 1 is by default used in programming the Teensy board, and when reserved with UART 2 pins 5, 21, 26, and 27 become unusable. Each required pin, required software library, and pin constraints such as disabled pins and peripherals were tabulated as shown in

Table 10.

Table 10: Pin Configuration of eCVT Controller

Device	Pin #	Disables Pin #	Note	Library
Engine RPM (CVT Primary Reluctor)	13		LED Pin	FreqCount
Wheel Speed	3	4 (Analog Write)		FreqMeasure
M1 INA	8			
M1 INB	9			
M1 PWM	6			
M1 ENCA	29			
M1 ENCB	30			
M1 STOP	33			
M1 SENSE	14		A0	
M2 INA	11			
M2 INB	12			
M2 PWM	7			
M2 ENCA	31			
M2 ENCB	32			
M2 STOP	34			
M2 SENSE	15		A1	
UART1 and 2	0, 1, 9, 10	5, 21, 26, 27	TX/RX 1, 2	Serial

After deciding on a pin configuration, the boards and peripherals were connected on a breadboard and powered by a PSU. See Figure 29.

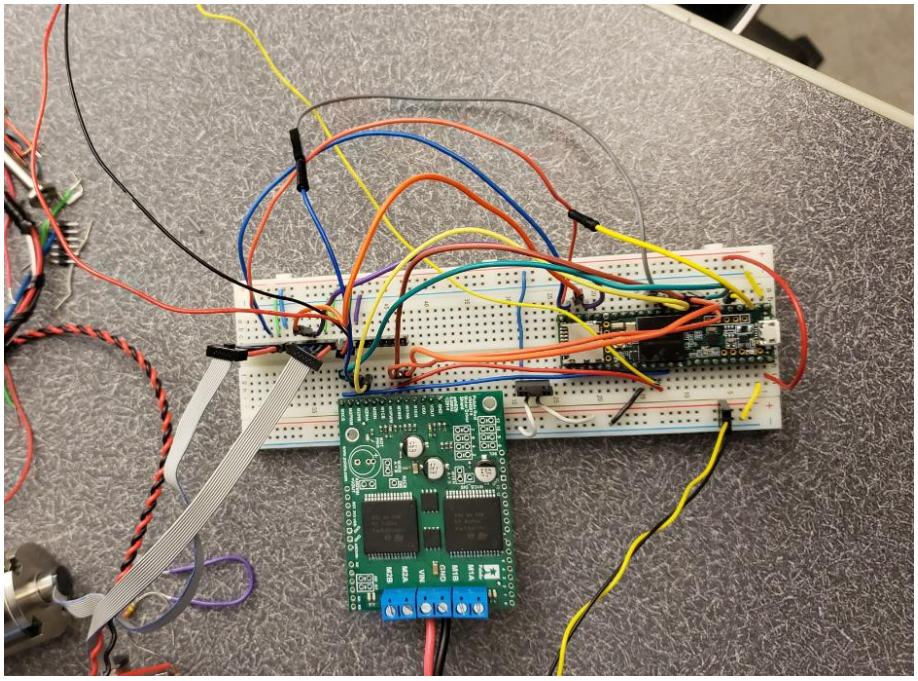


Figure 29. Breadboard tests of controller circuit.

Each motor, encoder, and hall-effect sensor were tested in parallel. The code to test each peripheral is documented in Appendix S. The correct closed-loop control of both motors simultaneously, while simultaneously reading quadrature encoder data and hall-effect sensor frequencies was successful after tweaking a few pins and software modifications.

After verification of the hardware and software, a final board design was started in EAGLE. The final board features removable shields for the motor driver and microcontroller. If the motor driver burns out or the controller is damaged (by voltage or physical damage), the shields can quickly be replaced and software flashed to the Teensy board. During bench testing, it was discovered that the IC's on the motor driver board would get very hot if a heavy load was applied to the motors, so heatsinks will be added if the steady-state load is discovered to be high enough to warrant heat management.

Custom library parts were created for the Pololu motor driver board and for the Teensy 3.5 board. Because the Pololu motor driver board is Arduino compatible, the Arduino shield pin configuration was implemented in the final design. This decision led to a greater mechanical bond and vibration damping of the motor driver board. The schematic of the final board is shown below in Figure 30. For now, the 5V rail is generated using an LM7805 linear voltage regulator. In a future iteration, this should be replaced by a switching voltage regulator to save energy and keep components cool. Note that the VDD and 3V3 pins on the motor driver board needed to be bridge externally from the PCB. The board design generated from the above schematic is shown in Figure 31. The ground plane is not shown in the figure.

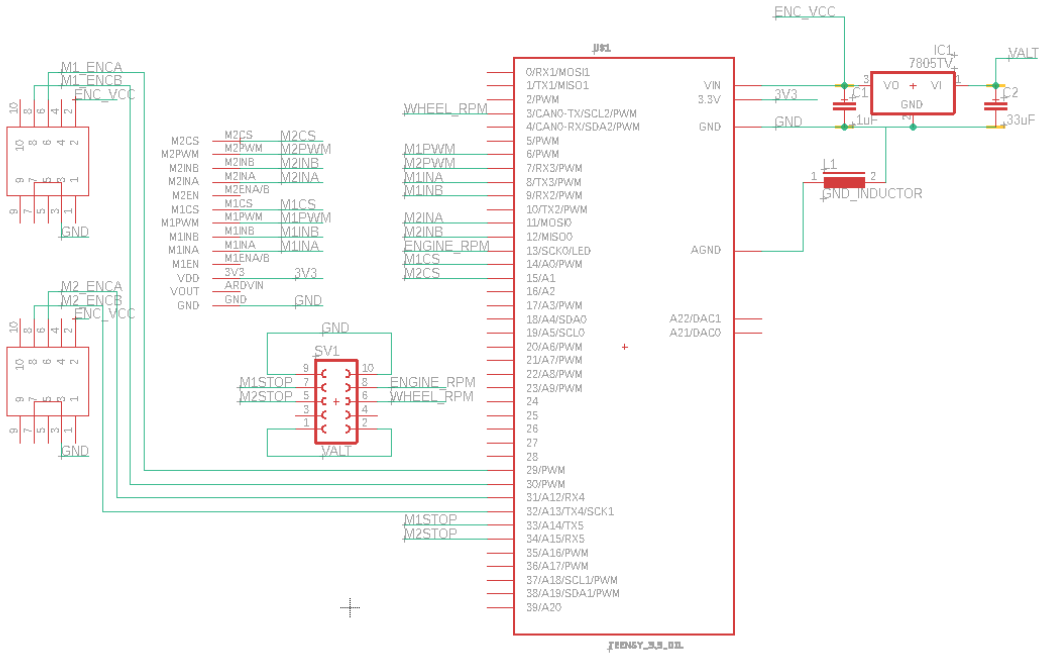


Figure 30. Schematic of final control board.

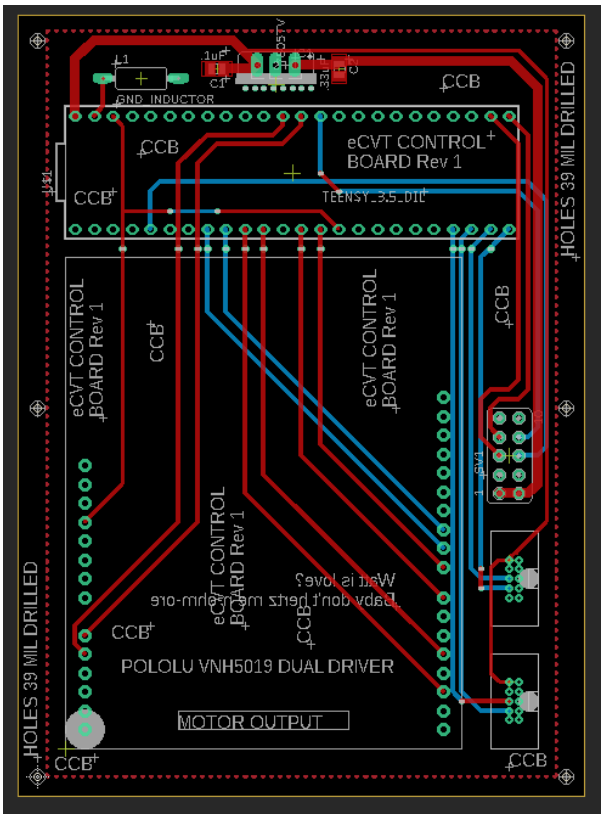


Figure 31. PCB Board design of controller circuit.

The circuit board was soldered and assembled. The completed board is shown in Figure 32.

Figure 32. Completed custom PCB.

The final board was tested using the same code and procedure that was used to verify the hardware and software configuration during breadboard testing, as shown in Figure 33. No modifications were needed to the PCB, however, as noted above, the 3V3 pin and VDD pin on the motor controller board needed to be bridged by solder. These pins are adjacent to each other. The final PCB performed as expected.

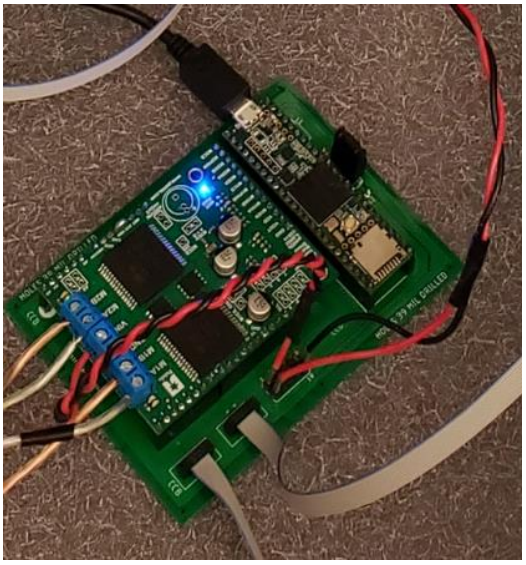


Figure 33. PCB under bench testing

The Simulink model of our ECVT is complete, and represents the logic of our control loop, with our desired engine RPM as an input, which goes through a PI controller to find the necessary gear ratio to reach this desired RPM based on current engine speed. This necessary ratio then goes into a look up table and the motor locations are determined. This position goes through a secondary PI controller to find the voltage that must be provided to the motor, with a saturation point implemented to protect our motors. Through a bench testing procedure, the desired control algorithm was tested. The software used to perform the bench testing can be found in Appendix S. A generic PI controller function was written which would perform PI control on both the outer loop and motor loops, as discussed in section 5.4. Another function was created, *myinterp1*, to take data points from a provided look up table and use linear interpolation to 'fill in' the gaps of the remaining data.

For our bench testing, we provided a lookup table to our software which mapped desired gear ratios to encoder ticks on each motor. Through the bench testing we ran our code in an infinite loop where we constantly 'asked' the ECVT to change a gear ratio of .1 at each iteration so we could prove the motors' full range of motion. As a result, the ECVT constantly shifted between a gear ratio of 4 and a gear ratio of .5, proving the validity of the PI control and interpolation software. Following further testing with the ECVT on the car, this loop may grow to include torque control as needed. Torque control would be useful to ensure that there is enough torque to maintain contact between the sheaves and the belt, as position control may not guarantee this.

As for our budget, there were minor updates after CDR, which include the addition of the custom PCB components, which have been added to Appendix L – Bill of Materials. Despite the addition of these components, we remained under our initial budget goal of $1000.

7 Design Verification

To verify that our design met the criteria laid out in the early section we intended to test our hardware to ensure that it can withstand the requirements of the competition. For the summary of the critical tests, refer to Appendix M – Design Verification Plan. However, many of these tests were not conducted as a result of the combination of the mechanical system being delayed in completion, the 3D printed model failing because of design miscommunication, and delay of critical electrical components. The testing we did do is detailed in the section below.

We began by bench testing the prototype board as mentioned in manufacturing. We ensured the board communicated with and accurately read all the different components: hall effect sensors, encoders, motors, limit switches, and current sensing. Table 11 shows the results of this testing.

Table 11: Bench Test Results

Requirement	Target	Measured
Ratio Precision	+/-.075	+/- .05
Engine Speed Precision	+/- 50 rpm	+/- 7 rpm

During this testing the temperatures of the board, the motor H-bridges and board power dissipation, were found to be quite hot and in need cooling. The sheave accuracy was tested by moving to the same encoder position 5 times (from both directions) and for 5 different sheave positions while recording the actual sheave positions with dial calipers. This precision was then propagated through the relationship between axial position and ratio. This test was done with no load and might not be exactly representative of dynamic precision on the functioning system. This precision does not account for the deflection of the mechanical system. Also, in this testing a large source of the uncertainty came from the backlash in the motor gearbox.

Engine rpm accuracy was measured by recording data off the setup on the Baja car. While the triggers are different lengths on the actual system it was considered negligible. To analyze this data the average rpm was calculated over .25 seconds, and it was found to have an uncertainty of 7 rpm. However, looking at the data, there was quite a lot of noise. Each individual data point varied from the average by nearly +/-100 rpm. Averaging over .25 sec was used to smooth the data, but in the future a running average would be a better way to reduce the bumpiness of the data while maintaining similar logging speed.

To further ensure the protection of our board, performed a submersion testing on the 2019 Baja Team's DAQ box, manufactured very similarly. The box failed to meet the IP67 requirement that the Baja team has asked for. However, the box did manage to keep out dust during a 4-hour endurance race in competition setting and on the bench, resisted powerful jets meeting the IP66 standard. To hit this higher standard, a gasket or O-ring should be used for sealing, in the mean time we will use tape.

Up to this point in the report the ECVT has not been tested on the Baja Car. This will be detailed more in project management section but ultimately issues with manufacturing pushed ECVT testing schedule back into the competition season of the Baja team making getting on the car impossible. On-car testing of the ECVT will be performed by the next senior project group in order to identify mechanical issues and improve the sophistication and tuning of the controls system.

8 Project Management

Our original design process and deliverables are summarized into five main categories, which are system modeling, sensor selection and placement, prototype control board, control algorithm creation and tuning, and creation of our graphic user interface (GUI). These deliverables can be seen below in

Table 12, with their corresponding goal completion dates and adjusted completion dates. You can see that the 3D printed mechanical model was not completed, due to a low feasibility of success based on continuous iterative design by the mechanical design team, and the GUI is also to remain incomplete due to the existence of a second senior project group that is preparing to take over and adjust the controls effective immediately.

Table 12: Summary of Deliverables

Deliverable	Goal Completion	Adjusted Completion
Prototype Control Board	11/27/2018	3/1/2019
Printed Circuit Board	4/2/2019	5/12/2019
Implemented Sensors	3/14/2018	3/14/2018
3D Printed Mechanical Model	11/27/2018	N/A
Configured Motors	11/2/2018	4/16/2019
Preliminary Control Algorithm	11/25/2018	
Controls Tuning	3/14/2018	6/14/2019
GUI	11/26/2018	N/A

To start our design process, we began with background research. This research included customer interviews, CVT research and comparisons, research on existing Baja EVTs, different actuation methods, and control schemes.

Our first project benchmark was a scope of work, which outlines our goals and necessary features, along with our plan to reach these goals. In order to better understand our system and the performance needs we began a model of our eCVT using an ideal shift curve look up table, PI controller and fuzzy logic controller. This model allowed us to narrow possible courses of action, using inputs for engine RPM, gearbox reduction and range of ratios. Through communication with drivers, CVT leads, electronics leads, and our design group we were able to determine our necessary sensor input and data outputs. These parameters allowed us to determine which sensors

we will need and begin to work with the design group on placement of these sensors. We also used this information to prototype our GUI, one of our main deliverables.

Through the selection of an actuation method by the design group, success of our rough system model, and additional research including discussion of eCVTs utilized by other Baja teams, we decided that a PI controller with a lookup table would be the best option for our eCVT. PI control is the best option due to its ease of implementation, tuning, and simulation performance. It was found in section 4.1 that PI control had the best performance via simulation. In addition, it only takes a few lines of code to program a PI controller, and the tuning for PI control has physical intuition associated with it. Furthermore, when passing the project to the next group, PI control is ideal because it is the only control scheme that most undergraduates learn, which will give good continuity to the project. It is important to note that PI controllers can be less robust than other control systems, which we will account for using redundancy in our system. The main reason this will work is because we have chosen to control both the primary and secondary faces, which will allow us to change our control system to use the information from one of these pulleys in the event of a failure in any components of the other actuated face. This allows us to continue design on another large deliverable, our control system.

The combination of this information allowed us to complete our second benchmark, which was to determine a preliminary design, and set forth with a plan of action for the rest of our design, manufacturing, and implementation.

Moving forward in this project, Tristan continued to refine our system model and test different shifting situations. This modeling has allowed him to decide on the design for our base control system. Additionally, Alec and Nick completed component and sensor selection. The specific components we have selected can be seen in Appendix L – Bill of Materials. With this knowledge materials were selected and purchased for the fabrication of a prototype board which was completed and tested for any unforeseen issues during Winter quarter before ordering a printed circuit board (PCB). Based on the success of this test the manufacturing plan for the final product was set, and this plan along with the actual process followed are discussed in more detail in the

Manufacturing section of this report. As a summary, manufacturing of the board and a 3D printed mechanical system were set to be completed by the end of Winter quarter for use in preliminary testing, however, setbacks altered this timeline. A prototype board was completed during the Fall quarter and tested to determine any issues, and upon successful testing a PCB was ordered during Spring quarter. The original hope was to test the prototype board on the mechanical system before ordering the PCB to avoid ordering a board without full testing with the mechanical system, but this was unable to be completed due to setbacks with the mechanical team during manufacturing, so the PCB was ordered during Spring when this determination was made. The board has since been assembled with the motors and lead screws and applied to the mechanical system for bench testing. Fortunately, the hardware and controls appeared to behave properly during this testing and no large adjustments were needed. A summary of our design verification plans along with the results of our bench tests can be seen in the Design Verification section of this report. The design verification process so far has involved testing the prototype board in Fall, testing the PCB for functionality in Spring, testing the controls algorithm through a bench test in Spring, and design verification will conclude with on-car acceleration testing for use of the future ECVT project team, which will be summarized into a separate testing report and submitted as an addendum to this Final Design Review.

Overall, project management was poorly implemented for this project as a whole. GroupMe was the primary avenue of communication for this project, with separate group chats for the mechanical team, controls team, and overall team. This method of communication was poor because not all relevant information was communicated and/or accessible between groups and GroupMe proved to be inconsistent about delivering notifications, which led to large delays in communication. In addition, the timeline of the project was heavily drawn out due to issues in project management. To begin, the concept of developing both a mechanical system and controls system proved to be a poor decision due to the heavy emphasis and necessity of testing controls in order to complete and tune a well-functioning algorithm. The lack of accessible time to test the system due to the project setbacks and Baja competitions caused the controls team to have to limit scope to things that could be easily programmed without on-car testing. That is, the team was unable to properly program and tune the system to handle any driving conditions that could not be predicted through modeling (which is many off-road driving scenarios), along with being unable to take data to create a verified look-up table and tune gains. In addition, timelines for both teams were chosen separately, not well communicated, and delays on either side often led to setbacks for the other team. In the future, I would choose to develop the mechanical system and controls in offset timing, along with improving communication through a more accessible and well-documented form of communication such as Slack, and communicate deadlines through something more accessible than a Gantt chart, such as a shared project-only Google calendar, along with a project checklist for short-term goals that notes the impact of delays. For the future group, it is recommended that the focus be on testing and tuning the controls portion of the CVT and that the mechanical changes be limited to only those that are necessary to operate, if any are required. Developing a properly tuned control algorithm is a project in itself, and the interdependence of the controls and mechanical system make it impossible to develop a well-performing controls system concurrently with large mechanical improvements. Since there is a completed mechanical system, that lends the

largest improvements to be made in the controls system for the upcoming year. It is best in the future to alternate large improvements to the mechanical system and controls system to occur in different years to make these changes manageable within the timelines the team operates. This will still require updates in the opposite system but will make the work-load and predictability more manageable.

For more details on the precise timeline of this project following CDR, you may look to our Gantt chart, located in Appendix D.

9 Conclusion

Over the course of the Spring 2018 quarter, we formed an understanding of our scope of work, researched necessary background information, and compared different design options along with the mechanical design group to determine our course of action for design. We confirmed these choices with system modeling, and chose to move forward with dual actuation (that is, of both the driving and driven pulleys), utilizing a PI controller with position controlled loops. The setpoint that dictates the position of the motors is dictated by an outer PI controller that takes engine rpm, outputs a desired gear ratio, and uses a look-up table to map the gear ratio to encoder ticks on the motor.

Upon the review and approval of our critical design plans from our noble sponsor, Junior Gonzalez, and our faculty advisor, John Fabijanic, we begun finalizing our software/hardware, and testing on the physical system. What this project primarily achieved is the first big steps in the development of the Baja eCVT. The hardware/software was constructed, and our control algorithm was bench-tested on the eCVT. We found that our control logic was proven to be valid in a 'no load' scenario.

A goal we did not achieve is performing an acceleration test, with the eCVT, on the actual car. There were many reasons, we did not get to this stage, both involving delays in expected deliverable dates on both the mechanical and mechatronic teams' parts. One thing we would have done differently was have a quarter delay between the mechanical team's project and the mechatronics team's project. Often in the last quarter, we found that our push forward to the next-phase in the mechatronics testing was often delayed due to unexpected manufacturing times with the mechanical design team. Although the mechanical team's components took longer to make than expected, our group, next time, would make sure that we could start testing on the eCVT, the second they were finished with manufacturing.

Aside from a change in project goals, a great deal was learned in this project. Along with the technical knowledge our team gained through the design and testing process, we learned many logistics of working on a full-scale, industry-like, project. Clear communication between design choices for both the mechanical and mechatronics team are invaluable for success in this multidisciplinary project. Often, we found that what the mechanical team wanted to do and what the mechatronics team wanted to do, did not intersect. Taking this knowledge to industry our team now knows the value in clear and constant status updates. Furthermore, another thing we learned is the value in accurately estimating deadlines. It was found throughout the year that our deadlines were more ambitious than originally thought. Being consistent with projected deadlines makes you a more reliable engineer.

Moving forward in this project, our system model, electrical system, control algorithm, and test results will be passed down to the next team for proper project continuation. We plan for the next team to take what worked, and what didn't work, to further improve the functionality for the eCVT. The next group will perform more sophisticated hardware and software tests (start-up, turn-off sequences etc) that will allow for the eCVT to be fully functional at the next Baja competition.

References

Aeen, Olav. *Aaens Clutch Tuning Handbook: For Serious Racers and Anyone who Wants More Performance From Their Variable Ratio Belt-Transmission*. Racine: Aaen Performance, 2015. Book.

Arm Limited. *GNU Arm Embedded Toolchain*. 27 June 2018. 2 October 2018.

Atmel. "SAM3X / SAM3A Series Datasheet." Datasheet. 2015. Datasheet.

Clemson University Vehicular Electronics Laboratory. *Transmission Control*. n.d. Website . April 2018.

Free Software Foundation, Inc. . *AVRDUDE - AVR Downloader/UploaDEr*. 8 Janurary 2009. 2 October 2018.

Free Software Foundation, Inc. *AVR Libc Home Page*. 9 February 2016. 2 October 2018.

Freescale Semiconductor, Inc. "K64 Sub-Family Reference Manual." Datasheet. 2014. Datasheet.

George, Damien. *MicroPython - Python for microcontrollers*. 2018. Website. 22 May 2018.

Gibbs, John H. *Actuated Continuously Variable Transmission for Small Vehicles*. PhD Thesis. Akron, 2009. Document .

Justin Lopas, Vincent Peng, Travis McIntyre, Jennifer Uang. *"Testing CVT" Final Report*. Class Report. Ann Arbor, 2015. PDF.

Lautzenhiser, A. G. magnetic Shaft Encoder with Relatively Moving Toothed Members. United States of America: Patent 3226711A. 28 December 1965. Patent.

PJRC. *Teensy USB Development Board*. n.d. 27 October 2018.

PRJC. *FreqCount Library*. n.d. Website. 27 March 2019.

Rainford Solutions. *IP Enclosure Ratings & Standards Explained*. 14 May 2018. Website. May 2018.

Refvem, Charlie. *Mechatronics Support and Feedback* Alec Hardy and Tristan Perry. October 2018.

Reiner, R. Code Wheel Shift Encoder. United States of America: Patent 2899673A. 8 August 1959. Website.

Subaru. "Transmission Control Module (TCM)." Subaru. *Repair Shop Manual- Subaru Impreza WRX- STi 2004*. 2013. 4AT82-4AT107. Online Book.

Appendix A - Customer Interviews

Appendices. Your FDR report should include at least the following appendices. Changed items are in bold: o QFD House of Quality o Decision Matrices o Preliminary analyses and/or testing details o Drawing Package, including a Bill of Materials (BOM), Assembly Drawing, exploded view Assembly Drawing, and detailed part drawings for all manufactured parts. This should reflect your final design, with any changes incorporated after CDR. o Electrical schematics or wiring diagrams, if your design includes electrical components. o Flowcharts and/or pseudocode, if your design includes programming. o Final code for any software you developed. o Links to product literature for all purchased parts. o A project Budget showing all actual material and part purchases, with part numbers and vendors identified for each. Indicate which component(s) in your BOM is supported by each purchase. o Legible analyses and/or test results to support all design decisions, with explanations. o Failure Modes & Effects Analysis o Design Hazard Checklist o Risk Assessment o Operators' Manual o Design Verification Plan & Report (all columns completed) o Gantt Chart (updated to what actually happened in the project)

Driver:

1. *What do you like and dislike about current CVT?*
2. *How can you tell if the CVT is responding well?*
3. *How much do you feel acceleration and top speed?*
4. *Which characteristic do you feel most limited on?*
    a. *Acceleration*
    b. *Torque*
    c. *Top Speed*
    d. *Other*
5. *Would you like to manually control the ratio, have another party control the ratio, or both?*
6. *Is there any information you'd like to access while driving?*
7. *Do you want to have the option to change out of pre-set mode while driving? (paddle-shift)*

1). I really don't like how the CVT can have "on" or "off" days. Even when we have the same settings, sometimes the CVT performs poorly; it's finicky. We haven't figured out how to make it backshift quickly, it's heavy, and doesn't provide us enough torque.
2). You can feel the shift speed and top speed. The quicker the shift speed, the better the performance. I can tell it isn't performing well when the car has trouble making it up hills, the entire car shudders, or loses power all together.
3). Not sure how to answer this… I feel like I have a pretty good understanding of the car's full potential, so I can feel when the car accelerates well or is nearing top speed. Also you can hear the engine overrun at top speed.
4). Torque mostly, acceleration next. Reliability is limited too.
5). Only in very particular situations. I believe it is too much to ask the driver to control the ratio when you have no feedback (live engine speed, wheel speed, etc.) There are too many

combinations and more often than not, I think the effort going into shifting the car will distract the driver and it will often be slower than having a decent CVT that does the work for the driver. It would be nice to have like a manual override button to have high speed or high torque though… Like to hold it in a certain ratio for a long straightaway or hill climb.
6). CVT temp. It would greatly affect how hard the driver pushes the car. Don't need temperature, just like a green, yellow, red. Maybe a warning light would be useful for all other issues, so the driver can diagnose the issue and radio to pits.
7). Yes, but I am not a fan of the paddles. For events like maneuverability on off-road terrain, it would be pretty easy for a driver to accidentally click the paddles. It would be better to have something less likely to be accidentally switched.

CVT Leads:

1. *What are the main problems you face?*
2. *What are your limitations caused by an "off-the-shelf" CVT?*
3. *Do you want to directly alter the code? What will make this easiest for you?*
4. *How do you want your GUI setup?*
5. *What data output would you like to see?*
6. *What do you look for in a good tune?*
7. *What data could you provide us?*
8. *Are there any considerations we aren't keeping in mind?*

1. After developing an ideal shift curve, we would still need to guess-and-check to find the tune that most closely resembles it. Additionally, a mechanical CVT might not have a tune that fits the ideal shift curve. For example, let's assume that a mechanical CVT behaves linearly to a certain input; we might want it to behave differently – with an eCVT that's possible.
2. Having a design specific to our car is better and more efficient than adapting a generic design for all cars. This includes both system integration and component choice.
3. Yes, it grants complete control over the fundamentals behind the eCVT.
4. A GUI is probably better and more reliable external from the car. I'm not sure what input variables you plan to use, so I can't say which GUI options would be most useful.
5. Vehicle speed, engine speed, rear wheel speed, theoretical CVT ratio (that is, assuming no belt slip). Comparing the theoretical ratio to the actual ratio, determined by engine speed / (rear wheel speed * gearbox reduction), will help us analyze belt behavior.
6. A good tune provides the best acceleration possible. This tune is partially a function of road load – some ways to find road load with sensor data include comparing the rear wheel speed to the vehicle speed and calculating the pitch of the car due to a slope (perhaps by using an accelerometer and/or gyroscope). Additionally, we always change the tune before each event, but with a perfect tune you wouldn't need to.
7. We can measure vehicle speed from the rotational speed of the front wheels because they rarely slip (this mostly only fails when the car is airborne and sometimes fails when the car turns). We can measure engine speed and rear wheel speed. We have the capability to measure the pitch and roll of the car, but we currently don't do so with high accuracy.
8. I think it's important to assess the speed of the vehicle as an input variable for determining the ideal instantaneous CVT ratio. Also, a higher resolution for both the front and rear wheel speeds (if needed) is attainable but would require being better designed into the car. Lastly, the car's behavior can change drastically when it's airborne (front wheels don't determine

vehicle speed, road load drops to almost 0, the engine temporarily increases RPM if not already at full); these situations might need to be determined and handled by the code.

Electronics Lead:

1. *What current sensors can we access data from?*
2. *What interface is used to access sensor data? For hardware, ie. I2C bus, soldered wire connectors, molex connections? For software, what microcontroller is used? Any API or frameworks?*
3. *Should we interface off of our own board, or integrate our CVT control system into the existing controller?*
4. *What power sources are currently available on the vehicle? Maximum voltage and max power draw?*
5. *Can you help with electronic manufacturing?*
6. *What serviceability requirements?*
7. *Any environmental requirements / international standard ratings should we follow? IP-68?*
8. *Are there any other considerations we aren't keeping in mind?*

1. We use hall effect sensors to collect data on the front left wheel, front right wheel, rear wheels, and engine speeds. We have a GPS which provides GPS coordinates and satellite time (among some other things). We have the capability to read data from accelerometers and/or gyroscopes, but not a great mounting location (for example, the acceleration/orientation of the center of mass of the car).
2. We use simple analog and digital connections for most sensors. We have used I2C connections with an accelerometer and gyroscope connected to an Arduino for temporary testing in the past. Our permanent DAQ is proprietary with limited control over the software. For more complicated applications, we have resorted to an Arduino due to its high level of online documentation, full control over code, and because we do not have the knowledge (yet) to manufacture our own boards. No APIs or frameworks for the hall effect sensors (they are digital on/off), we have used libraries (provided by the manufacturers) for the accelerometers and gyroscopes.
3. Use your own board. Our DAQ sucks ass at output control. If you decide to use an Arduino (which we sometimes use depending on the application of the collected data), you will probably need one with a higher clock speed than the UNO R3 (for example, a MEGA).
4. By rules, any electrical power used for powertrain components must be produced by an OEM alternator. If I remember correctly, there are three choices to choose from. I'm planning to research and test them in a couple weeks.
5. We have some sponsors that can manufacture circuit boards for us (but we must buy our own components). We can help with wiring.
6. If you use an Arduino (or similar off-the-shelf board), keep in mind that many of these are meant for prototyping purposes. Most of their connections aren't very strong/reliable. In the past, we have used hot glue to avoid soldering connections directly onto the board, but that would be way too janky for something as critical as an eCVT.
7. Use racing sensors where possible; all our hall effect sensors are dustproof and waterproof. For overall electronics, at minimum you probably want at least IP55 on Baja, but if better is possible then shoot for it (IP67 or IP68 is a good talking point for design). The CVT on

the Baja car is rarely submerged underwater, but even when it is, the CVT is underwater for 5 seconds at most.

8. Keep in mind that your system may take time to "boot" after the engine turns on (the eCVT must be powered by the engine) and that the eCVT should be able to handle that interim.

CVT Testing/Tuning:

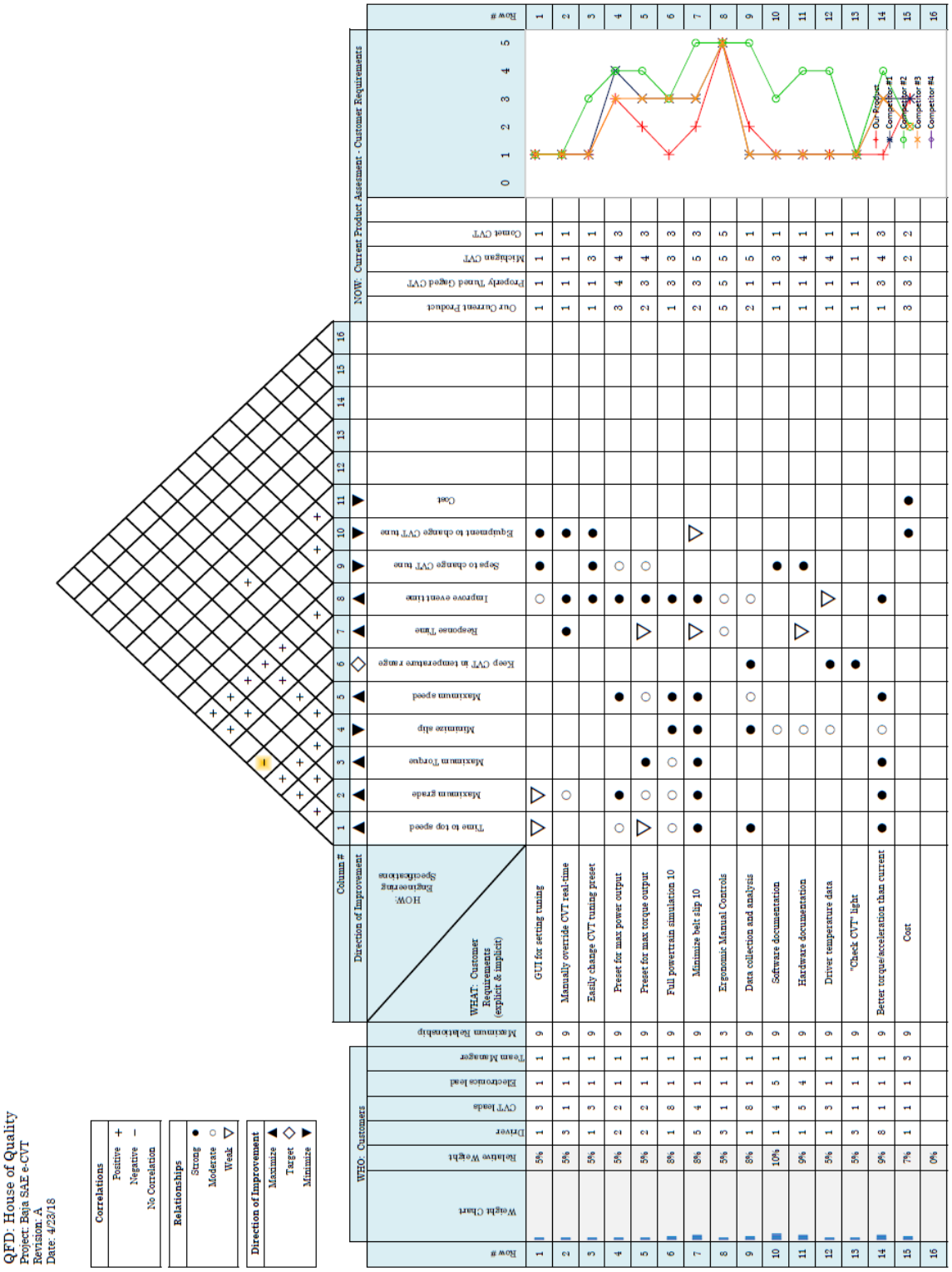
1. *How do you want to be able to tune the car? How do you currently tune? What do you dislike about this method?*
2. *What will make it easier for you to tune during testing and competition?*
3. *What equipment do you have to interface with the electronics? Are you willing to use a new interface system?*

1. Onboard tuning for an eCVT would likely be clumsy and/or unreliable for the first year of its implementation; it would likely be easier to tune with an external device or computer, especially if it is easily handled with code. On the other hand, this would require a computer every time an eCVT tune needs to be changed/updated.

2. Improving the precise control over CVT tuning would likely provide the largest performance gains. Additionally, bettering the time-efficiency of the tuning process is always helpful.

3. We have a laptop, and sure.

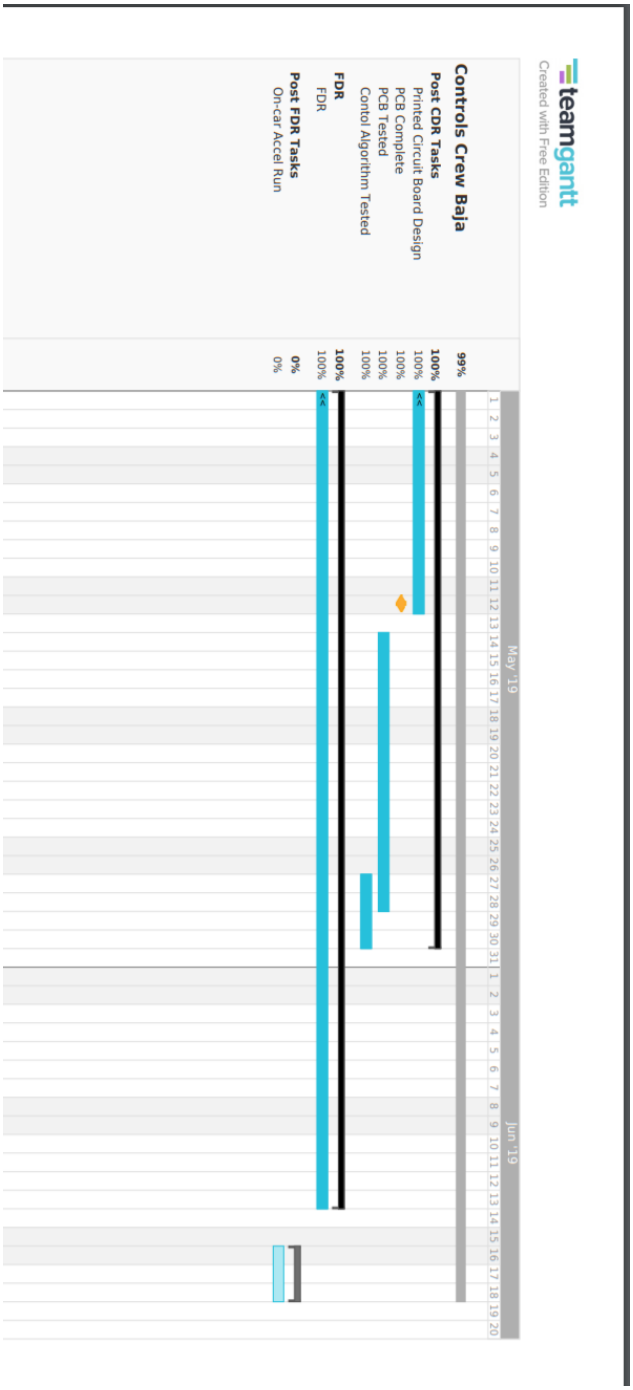
Appendix B - QFD House of Quality



Appendix C -Baja SAE Relevant Rules

B.2.7.15	“The engine may be fitted with an approved alternator to generate electrical power. The only alternators which are permitted are those which Briggs & Stratton specifies for the engine model. Available alternators are sized in 3, 10, and 20 Ampere versions.”
B.9.1	“All rotating powertrain components (CVTs, Gears, Sprockets, Belts and Chains) shall be shielded to prevent injury to the driver, track workers, or bystanders. Guards shall protect against hazardous release of energy should rotating components fail. Guards shall also protect against fingers, loose clothing, or other items from being entangled in the rotating components (pinch points). Universal joints, CV joints, hubs, rotors, wheels and bare sections of shafts are exempt from the requirements of B.9.1 and B.9.2.”
B.9.2	“Powertrain guards and shields protecting against hazardous release of energy shall extend around the periphery of the rotating components (chains, gears, sprockets, belts, and CVT’s) and have a width wider than the rotating part the guard is protecting. Note: This means the entire periphery of the primary CVT pulley, not just the belt width. All powertrain guards shall be constructed of one or both of the following required materials: -Steel, at least 1.5 mm (0.06 in.) thick, meeting or exceeding the strength of AISI 1010 steel. Page 68, Revision D – 2018/05/01 -Aluminum, at least 3.0 mm (0.12 in.) thick, meeting or exceeding the strength of 6061-T6 aluminum. Holes and/or vents in the portion of the powertrain guard surrounding the rotating components are acceptable provided that in the event of a powertrain failure, no parts can escape. No direct path shall exist tangent to any rotating components. Powertrain guards shall be mounted and secured with sound engineering practices in order to resist vibration and shock.
B.9.3	“Rotating parts in the powertrain system rotating faster than the final drive shall be guarded on all sides, in addition to the guard around the periphery. Guarding for pinch points shall prevent small, searching fingers from getting entrained in any rotating part. Flexible, non-rigid, fabric coverings such as "Frogskin", Ceconite, and neoprene are unacceptable for use as finger guards. Powertrain covers fastened with adhesive, ratcheting tie-downs, and other temporary methods are explicitly prohibited. All powertrain covers shall have resilient and durable mountings with easily accessed and actuated fastening devices. A complete cover around the engine and drivetrain is an acceptable shield for pinch points, but does not relieve the requirement for release of hazardous energy.”
B.10.2	“All vehicle wiring and connectors shall be cleanly and neatly installed. Wiring shall be routed away from sources of excessive heat, abrasion, chafing, and possible short circuit. Wiring shall be installed and routed such that it does not become a hazard to cockpit egress.”
B.10.6	“Vehicles may be equipped with data acquisition (data logging) systems. Data acquisition systems providing live feedback to the driver or telemetry data to the team must be included in the cost report. Data acquisition systems not providing live data to the driver and/or telemetry data to the team may be excluded from the cost report.”

Appendix D - Gantt Chart



Appendix E - List of Potentially Required Sensors

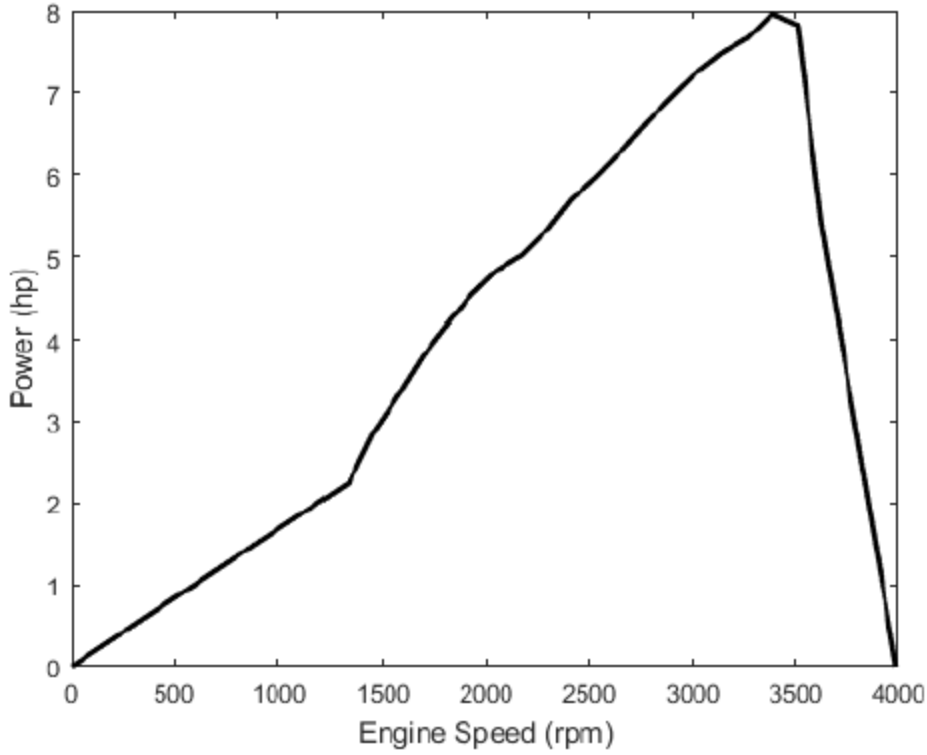
Sensor	Types	Reason / Clarification
Primary shaft encoder	Optical Encoder or Hall Sensors (3)	Position of input shaft
Secondary shaft encoder	Optical Encoder or Hall Sensors (3)	Position of output shaft
Belt position sensor	Optical	Optical sensor to track position/speed of belt. Compare this to primary/secondary position to determine slip. Note: We will need to draw marks on the belt for the encoder to read
Temperature Sensor	Thermocouple	Temperature inside CVT
Throttle Position Sensor / Brake Switch	Potentiometer / Contact switch	Probably already on vehicle, can be used to tune controller to react to throttle/brake input (ex. disenguage when throttle pos = 0)
Voltage Sensor	[Built into microcontroller]	Alternator performance check. Built into uController, use voltage divider circuit
Program Enable	slide switch / DIP switch	Hardware enable for programming/modifying controller
Manual Override	Latching switch (slide or push)	In cockpit, manual override
Manual Gear Position (option 1)	Potentiometer	Allows driver to control gear ratio (infinite)
Manual Gear Position (option 2)	Push (non-latching) switch	Allows driver to control fixed number of manual gear ratios (paddles?)

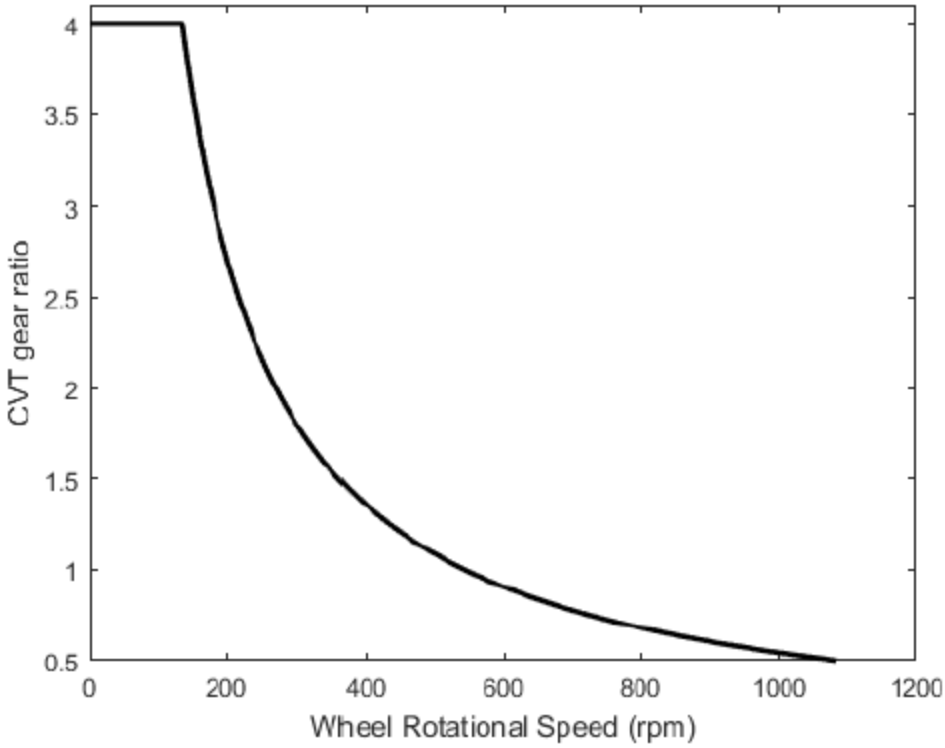
For each actuator - If using stepper motors in OL control		
Sensor	**Types**	**Reason / Clarification**
Limit switch	Contact switch	End stop. Only need 1 for a known amount of travel

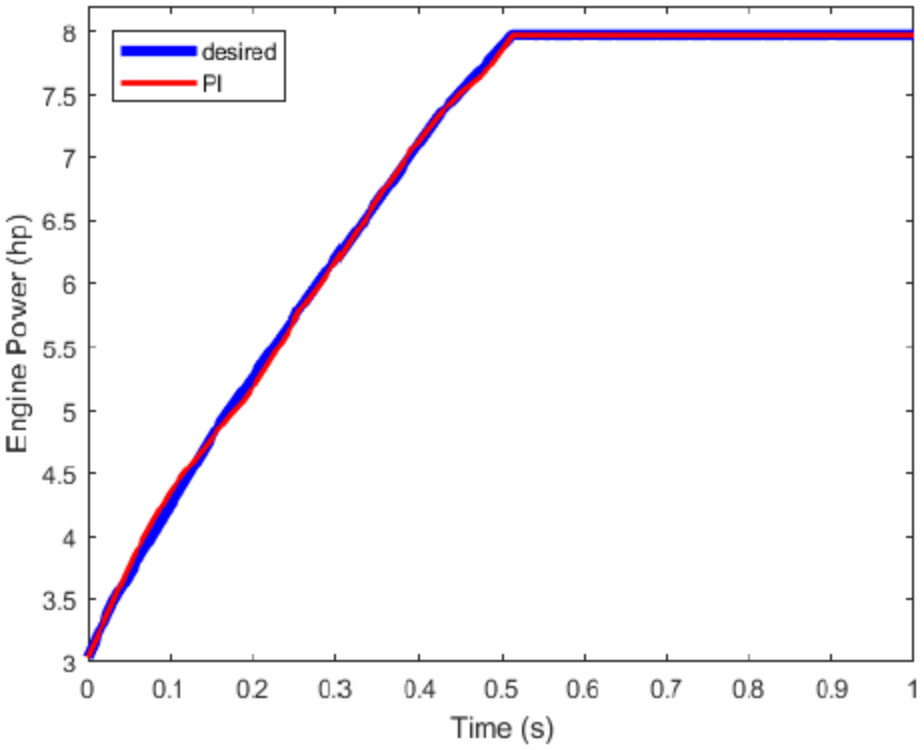
For each actuator - if using DC or BLDC motors in CL control		
Sensor	**Types**	**Reason / Clarification**
Linear position sensor (option 1)	Optical Encoder or Hall Sensors (3) AND limit (contact) switch	Position/Ratio of pulley. This method uses an encoder on the output shaft of the actuator with a limit switch as the endstop
Linear position sensor (option 2)	Linear Potentiometer, LVRT, or Linear Optical Encoder	Position/Ratio of pulley. This method involves either moving a slider attached to the pulley or optically determining the position of the pulley.

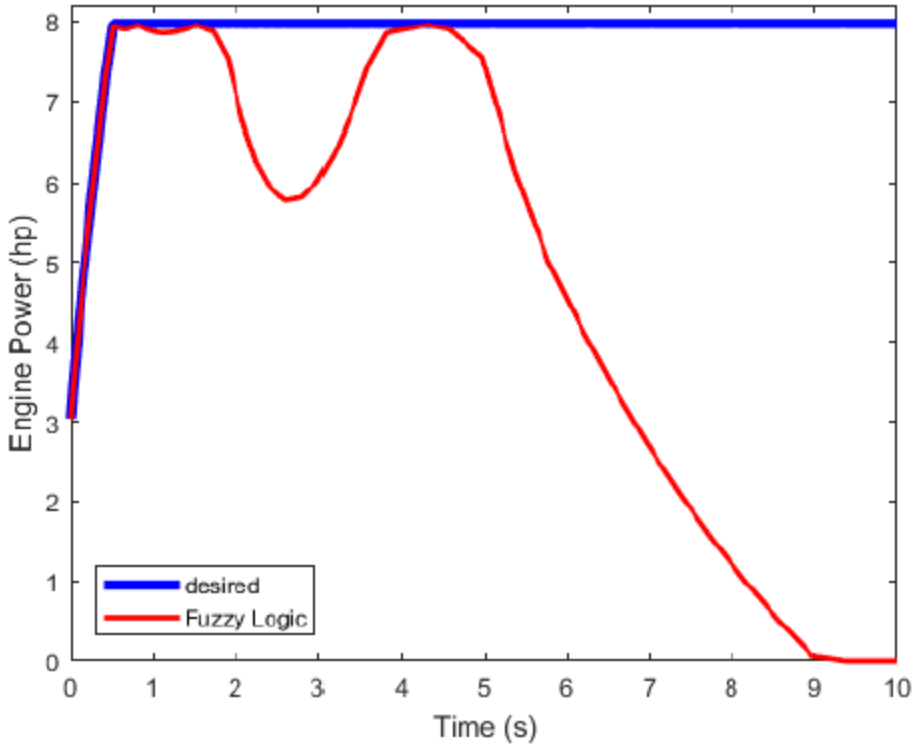
Appendix F – Longitudinal Dynamics of the Baja Vehicle

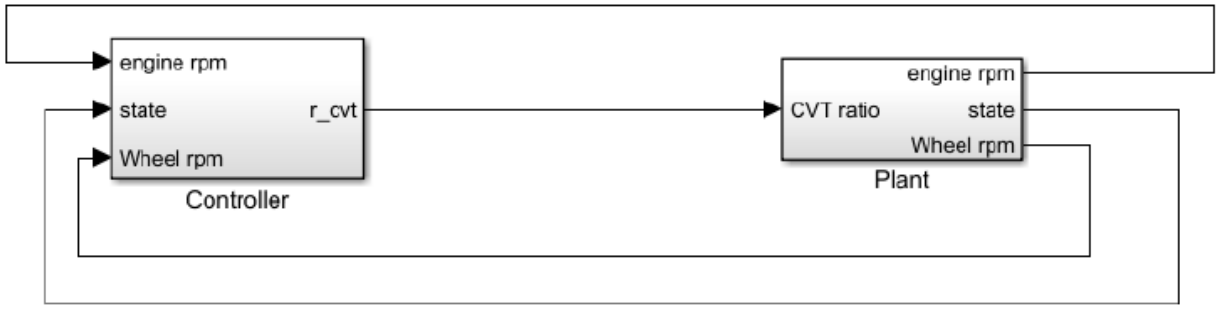
```
clc
clear all
close all
load('CVT_engine_tables.mat');
hp_v_speed = [[0 0];hp_v_speed;[1.1*max(hp_v_speed(:,1)) 0]];
figure(1)
plot(hp_v_speed(:,1),hp_v_speed(:,2),'Color','black','linewidth',2)
ylabel('Power (hp)')
xlabel('Engine Speed (rpm)')
torque_v_speed = [[0 0];torque_v_speed;[1.1*max(torque_v_speed(:,1))
 0]];
r_des = [.5:.05:4]';
wheel_rpm = 3388./(r_des*6.25);
des_curve = [flip([wheel_rpm;0]),flip([r_des;4])];
figure(2)
plot(des_curve(:,1),des_curve(:,2),'Color','black','linewidth',2)
ylabel('CVT gear ratio')
xlabel('Wheel Rotational Speed (rpm)')
ylim([.5,4.1])
KP = 10;
KI = 1.5;
PI = 0;
LT = 1;
res(1) = sim('CVT_dynamics.slx');
figure(3)
plot(res(1).tout,res(1).power_out,'Color','blue','linewidth',4)
hold on
PI = 1;
res(2) = sim('CVT_dynamics.slx');
plot(res(2).tout,res(2).power_out,'Color','red','linewidth',2)
legend('desired','PI','location','northwest');
ylabel('Engine Power (hp)');
xlabel('Time (s)');
axis([0 1 3 8.2])
hold off
LT = 0;
PI = 0;
res(3) = sim('CVT_dynamics.slx');
figure(4)
plot(res(1).tout,res(1).power_out,'Color','blue','linewidth',4)
hold on
plot(res(3).tout,res(3).power_out,'Color','red','linewidth',2)
legend('desired','Fuzzy Logic','location','southwest');
ylabel('Engine Power (hp)');
xlabel('Time (s)');
axis([0 10 0 8.2])
hold off
```

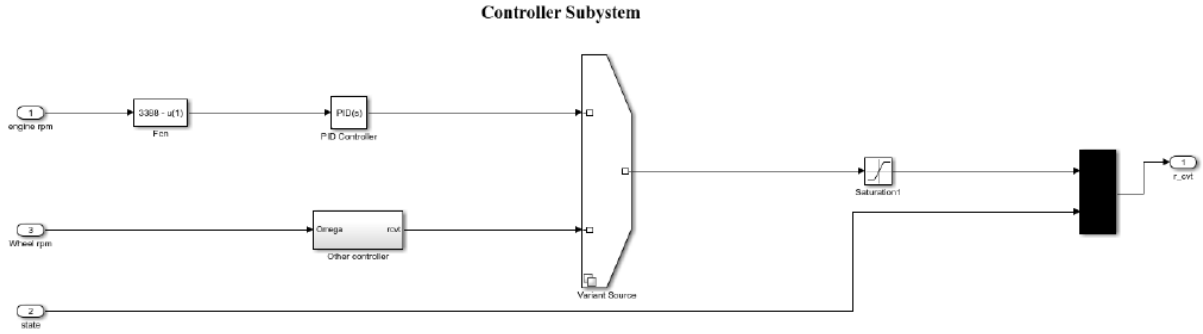

Power (hp)
Engine Speed (rpm)
0
1
2
3
4
5
6
7
8
0
500
1000
1500
2000
2500
3000
3500
4000

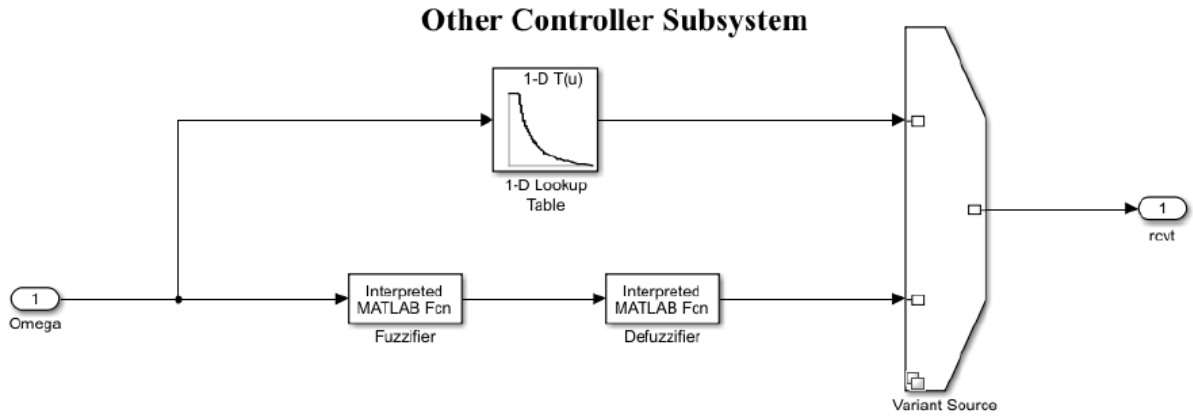

CVT gear ratio
Wheel Rotational Speed (rpm)
0.5
1
1.5
2
2.5
3
3.5
4
0
200
400
600
800
1000
1200

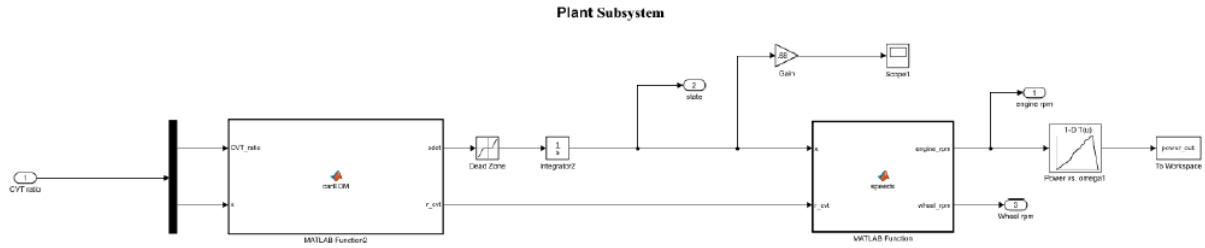

desired
PI
Engine Power (hp)
Time (s)
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


desired
Fuzzy Logic
Engine Power (hp)
Time (s)
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
8
9
10


engine rpm
state
Wheel rpm
r_cvt
Controller
engine rpm
CVT ratio
state
Wheel rpm
Plant


Controller Subsystem
engine rpm
3360 - u(1)
Fcn
PID(s)
PID Controller
Wheel rpm
Omega
rcvt
Other controller
state
Variant Source
Saturation1
r_cvt


Other Controller Subsystem
1-D T(u)
1-D Lookup
Table
Omega
Interpreted
MATLAB Fcn
Fuzzifier
Interpreted
MATLAB Fcn
Defuzzifier
Variant Source
rcvt




```
function y = Tristans_fuzzifier(x)

m_1 = -1/(174.08 - 136);      b_1 = 1;
m_2 = -m_1;                   b_2 = 0;
m_3 = -1/(241.8 - 174.08);    b_3 = -(241.8 - 136)*m_3;
m_4 = -m_3;                   b_4 = -(174.08 - 136)*m_4;
m_5 = -1/(395.6 - 241.8);     b_5 = -(395.6 - 136)*m_5;
m_6 = -m_5;                   b_6 = -(241.8 - 136)*m_6;
m_7 = -1/(1088 - 395.6);      b_7 = -(1088 - 136)*m_7;
m_8 = -m_7;                   b_8 = -(395.6 - 136)*m_8;

y = zeros(5,1);

if x >= 136 && x < 174.08
    y(1,1) = m_1*(x - 136) + b_1;
    y(2,1) = m_2*(x - 136) + b_2;
end

if x >= 174.08 && x < 241.8
    y(2,1) = m_3*(x - 136) + b_3;
    y(3,1) = m_4*(x - 136) + b_4;
end

if x >= 241.8 && x < 395.6
    y(3,1) = m_5*(x - 136) + b_5;
    y(4,1) = m_6*(x - 136) + b_6;
end

if x >= 395.6 && x <= 1088
    y(4,1) = m_7*(x - 136) + b_7;
    y(5,1) = m_8*(x - 136) + b_8;
end


end
```

Published with MATLAB® R2017a

```
function r_cvt    = Tristans_defuzzifier(y)

phi_low = 4;
phi_semi_low = 3.125;
phi_medium = 2.25;
phi_semi_high = 1.375;
phi_high = .5;

if y == zeros(5,1)
    r_cvt = 4;
else
    r_cvt = (y(1,1)*phi_low + y(2,1)*phi_semi_low +
 y(3,1)*phi_medium ...
     + y(4,1)*phi_semi_high + y(5,1)*phi_high)/(sum(y));
end



end
```

Table of Contents


```
function [xdot,r_cvt] = carEOM(CVT_ratio,x)
```

Constants and Parameters

```
persistent tables
if isempty(tables)
    tables = load('CVT_engine_tables.mat');
end
torque_v_speed = tables.torque_v_speed;
mew_s = .72;                            % Static friction coefficient for
 car tire to asphalt
g = 32.2;                               % Gravitational constant [ft/s^2]
h =  16/12;                             % center of gravity height [feet]
L =  59/12;                             % wheelbase [feet]
L1 = 31.9/12;                           % Front half length of car [feet]
L2 = 27.1/12;                           % Back half length of car [feet]
%ha = 24/12;                            % height that drag force acts at
 [feet]
W_tot = 503;                            % Total weight of car (lbf)
m = W_tot/g;                            % Total mass of car (slugs)
rho = 0.074887/32.2;                    % Air density at stp  [lb/ft3]
Afront = 19.50;                         % frontal area of vehicle
 [feet^2]
CD = 0.66;                              % Drag coefficient
Fdrag = (1/2)*rho*Afront*CD*x(1)^2;     % Drag Force (lbf)
tire_diameter = 21/12;                  % Tire diameter [feet]
tire_radius = tire_diameter/2;          % Tire radius [feet]
gear_box_ratio = 6.25;                  % input/output
motor_speed = (x(1,1)/tire_radius)*(60/
(2*pi))*((gear_box_ratio*CVT_ratio));

if motor_speed < torque_v_speed(:,1)
    Tm = interp1(torque_v_speed(:,1),torque_v_speed(:,2),motor_speed);
else
    Tm = torque_v_speed(end,2);
end
W_rear = W_tot/2 ;                      % rear vehicle weight [lbs] Will
 account for dynamic weight later
W_front = W_tot/2 ;                     % front vehicle weight [lbs] Will
 account for dynamic weight later
W_tire = 12.4;                          % Weight of a single tire [lbs]
m_tire = W_tire/g;                      % Mass of a single tire [slugs]
```

```
I_wheel = (m_tire*tire_radius^2)/2;  % Polar Moment of Inertia of a
 single tire modeled as a solid cylinder
grade = 0.07;                        % Cuesta grade
theta = atan(grade);                 % grade angle [radians]
theta = 0;
% if x(1) >= 50
%     theta = 0;
% end                                % Flat road angle [radians]
fo = .02;                            % rolling resistance coefficient
fs = .01;                            % rolling resistance coefficient
mtot = (1/tire_radius^2)*(4*I_wheel) + m;
                                     % Total mass [slugs]
gear_box_ratio = 6.25; % input/output
```

Initialization of ODE

```
Fgrav = m*g*sin(theta);                          % Gravitational
 force [lbf]
% fr = fo + 3.24*fs*(((x(1)*.6818)/100)^2.50);   % rolling resistance
 coefficient
% FRtot = fr*m*g*cos(theta);                     % Total rolling
 resistance [lbf]
```

ODE

ode describing time rate of change of velocity if isnan(Tm) xdot = 0; r_cvt = CVT_ratio; else if x(1)>=25 Tm = 0; end

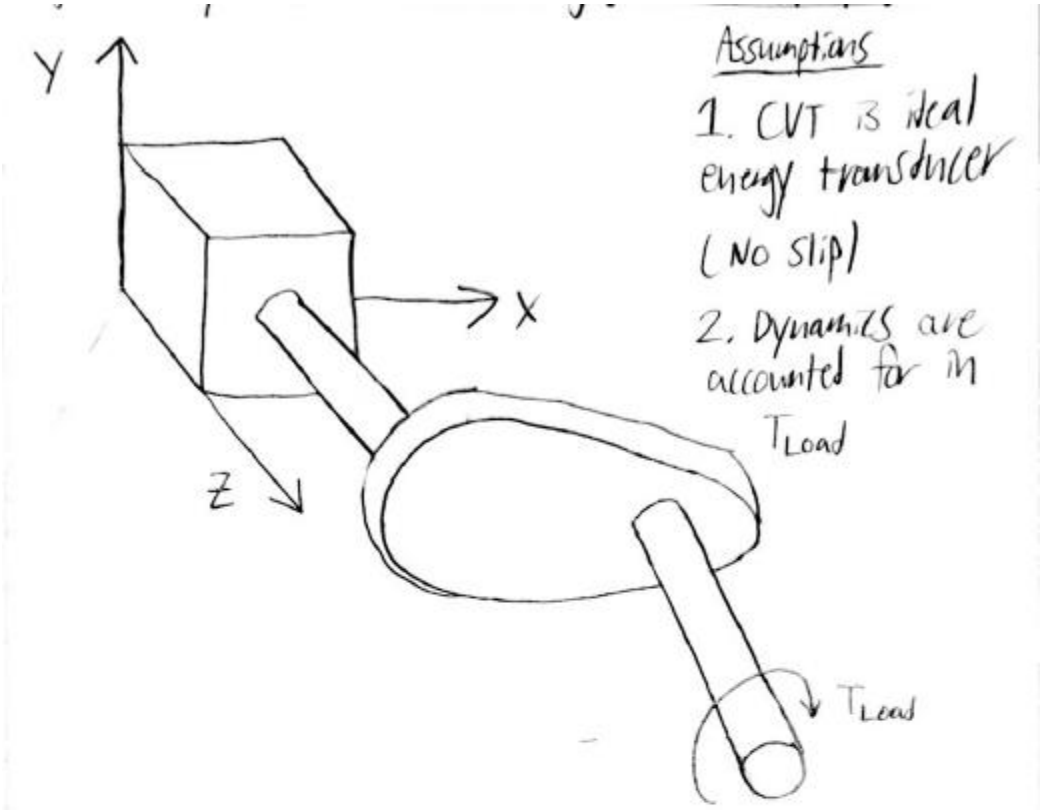
```
xdot = (((CVT_ratio*gear_box_ratio)*(Tm/tire_radius) - Fgrav - Fdrag)/
mtot);

r_cvt = CVT_ratio;

end
```

Published with MATLAB® R2017a

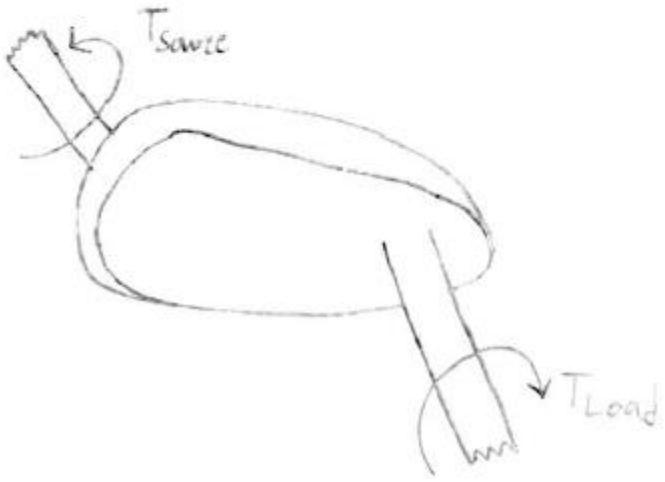
```
function [engine_rpm,wheel_rpm] = speeds(x,r_cvt)

engine_rpm = (1/(21/24))*(60/(2*pi))*x*((6.25*r_cvt));
wheel_rpm = (1/(21/24))*(60/(2*pi))*x;
```

Published with MATLAB® R2017a

Appendix G – Longitudinal Dynamics



FBD



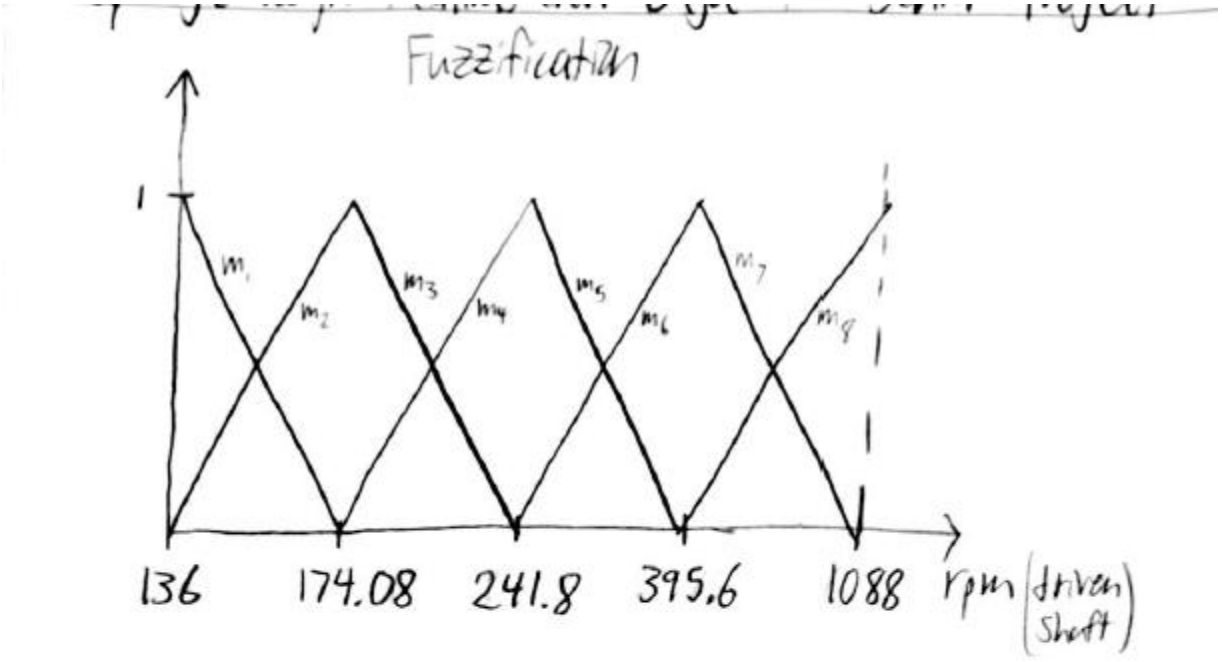
$$\Sigma \vec{M}_g = 0$$

$$r_{CVT} T_{source} - T_{Load} = 0$$

$$r_{CVT} r_{\substack{gear \\ box}} T_{source} = I_{eff} \ddot{\theta}_{wheel} + F_{drag} + F_{grav} + F_{\substack{rolling \\ resistance}}$$

$$\ddot{\theta}_{wheel} = \left(r_{CVT} r_{\substack{gear \\ box}} T_{source} - F_{drag} - F_{grav} - F_{\substack{rolling \\ resistance}} \right) / I_{eff}$$

Appendix H – Fuzzy Logic Design



136 rpm – slow

174.08 rpm – semi-slow

241.8 rpm – Medium

395.6 rpm – Semi-medium

1088 rpm – Fast

Rule base

If slow then low

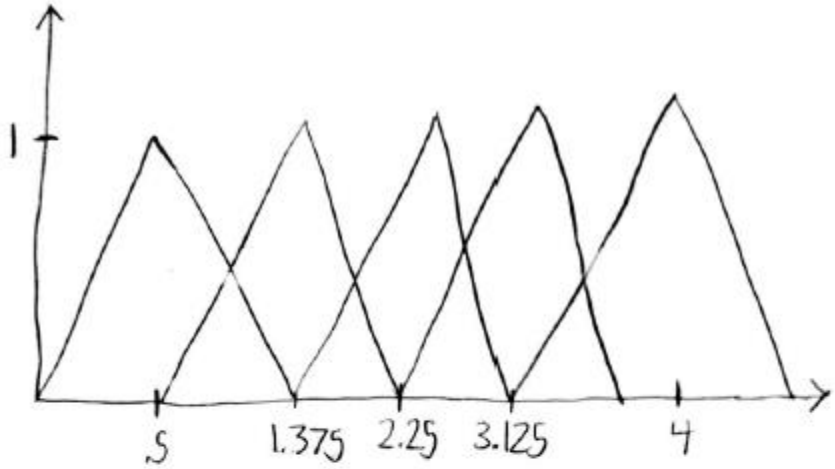
If semi-slow then semi-low

If medium then medium

If semi-medium then semi-high

If fast then high

Defuzzification



4 - low gear ratio

3.125 - Semi-low gear ratio

2.25 - Medium gear ratio

1.375 - Semi-high gear ratio

.5 - high gear ratio

$\Phi_{low} = 4$ $\Phi_{high} = .5$

$\Phi_{semi-low} = 3.125$

$\Phi_{medium} = 2.25$

$\Phi_{semi-high} = 1.375$

Appendix I – Stress Test Code for Controller Selection

```
import random
import time


def selection_sort(list):
    """ Selection Sort Algorithm """

    # Empty List Case
    if len(list) == 0:
        return 0

    num_comparisons = 0
    lowest_index = 0
    current_index = 1
    index_to_swap = 0

    while index_to_swap < len(list) - 1:

        lowest_item = list[index_to_swap]
        lowest_index = index_to_swap

        while current_index < len(list):

            num_comparisons += 1
            if list[current_index] < lowest_item:
                lowest_item = list[current_index]
                lowest_index = current_index
            current_index += 1

        # Swap lowest item with index_to_swap
        old_index_to_swap = list[index_to_swap]
        list[index_to_swap] = list[lowest_index]
        list[lowest_index] = old_index_to_swap

        # Reset current index, repeat
        index_to_swap += 1
        current_index = index_to_swap + 1

    return num_comparisons


def insertion_sort(list):
    """ Insertion Sort Algorithm """

    # Empty list case
    if len(list) == 0:
        return 0
```

```
    num_comparisons = 0

    sort_stop = 0
    while sort_stop < len(list):
        curr_index = sort_stop - 1 if sort_stop > 0 else 0
        num_comparisons += 1
        if list[sort_stop] < list[curr_index]:
            # Swap
            sort_swap = sort_stop
            num_comparisons += 1
            do_loop = list[curr_index] > list[sort_swap]
            while do_loop:
                old_sort_stop = list[sort_swap]
                list[sort_swap] = list[curr_index]
                list[curr_index] = old_sort_stop
                if curr_index == 0:
                    break
                curr_index -= 1
                sort_swap -= 1
                do_loop = list[curr_index] > list[sort_swap]
                num_comparisons += 1
        sort_stop += 1
    return num_comparisons

def main():
    # Give the random number generator a seed, so the same sequence of
    # random numbers is generated at each run
    random.seed(1234)

    # Generate n random numbers from 0 to 999,999
    n = 850
    randoms = random.sample(range(999), n)
    print(randoms)
    start_time = time.time()
    comps = selection_sort(randoms)
    stop_time = time.time()
    print(comps, stop_time - start_time)
    print(randoms)


if __name__ == '__main__':
    main()
```

Appendix J – Safety Hazard Checklist

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.

Description of Hazard	Planned Corrective Action	Planned Date	Actual Date
Large Moving Masses and Forces	The system will be covered with a protective cover to prevent unintentional injury from moving masses.	MFG Date	X
Stored Energy in the System	The system will be covered with a protective cover to prevent unintentional injury for energy stored in battery.	MFG Date	X
Materials Known to the State of California to be Hazardous	The electronic components will contain traces of elements known to be hazardous. Solder used to connect nodes contains lead. These components are packaged to prevent direct contact with hazardous materials.	Electronics MFG Date	X
Exposure to Extreme Environmental Conditions	The transmission will operate in off-road environments. Control system will be designed with IP67 or better standards.	MFG Date	X

Appendix K – FMEA

Part	Potential Failure	Counter Measure
Alternator	Overdrawing, because motor stalls	Have 18 amp fuse inline
		Have saturation limits implement through code
Motor	Drawing too many amps continuously	Have saturation limits implement through code
	Overheating	Implement active cooling
	Submerging in water	Create separate enclosure for motor
	One motor damaged through comp	Implement shifting stop if no feedback seen from motor
Micro Controller	Reverse polarity plugged in	Voltage protection implemented in board
		Wiring schematic/ color coded wiring
	Plugging things into the wrong spots	Wiring schematic/ color coded wiring
		Preflight check list
Wiring Connections	Pulling out of connections	Strain Relief
		Using automotive grade connections
		Only crimped connectors
Sensors	Encoders getting bad data (muddy)	Enclosure from water and dust
		Use proximity sensor as back up
		Use engine and wheel speed to calculate ratio
	Wheel Speed Sensor getting bad data	Use engine speed and calculated ratio to calculate wheel speed
	Engine Speed Sensor getting bad data	Use wheel speed and calculated ratio to calculate engine speed
	More than 2 critical sensors down	implement shifting stop if no feedback seen from sensors

Appendix L – Bill of Materials

Item Number	Part/Material	Purpose	Part Number	Vendor	Unit Cost	Quanitity	Subtototal
1	Teensy 3.5 MCU	Control Board	3267	Amazon	$ 27.89	1	$ 27.89
2	H-Bridge	Control Board	2507	Pololu	$ 49.95	1	$ 49.95
3	Protoboard	Control Board	a14060400ux0668	Amazon	$ 6.84	1	$ 6.84
4	Alternator	Power Supply	790320	Briggs and Stratton	$ 65.95	1	$ 65.95
5	Regulator	Power Supply	790325	Briggs and Stratton	$ 62.61	1	$ 62.61
6	Configured Motor	Actuation Method	B78B98B35398	Maxon Motors	$ 83.33	6	$ 500.00
7	Limit Switches	Failure Prevention	SS-5GL13T	Sager Electronics	$ 1.19	2	$ 2.38
8	Variable Resistor	Failure Prevention	PTA6043-2015DPA103	Digikey	$ 1.72	1	$ 1.72
9	Hall Effect Sensor	Sensor	555050002B	Verical	$ 16.68	2	$ 33.36
11	Plastic 1/4" Sheet	Mounting	HD-B	Amazon	$ 11.47	1	$ 11.47
12	Rubber 1/16"	Mounting	RSBLK6x6x116-04-03	Amazon	$ 8.86	1	$ 8.86
13	Weather Pack Connectors	Mounting	Size TBD	Summit Racing	$ -	0	$ -
14	IR Sensor	Data	13461	Waveshare	$ 16.99	1	$ 16.99
15	Threaded Inserts	Mounting	93415A021	McMaster-Carr	$ 10.21	1	$ 10.21
16	Screws	Mounting	92196A106	McMaster-Carr	$ 3.92	1	$ 3.92
17	Rectangular Connector	PCB	1175-1664-ND	Digi-Key	$ 2.35	8	$ 18.80
18	Flat Ribbon Cable	PCB	3M156019-10-ND	Digi-Key	$ 6.20	2	$ 12.40
19	Rectangular Connector-Header Male	PCB	1175-1627-ND	Digi-Key	$ 0.58	4	$ 2.32
20	Ceramic Capacitor	PCB	445-9194-1-ND	Digi-Key	$ 0.78	4	$ 3.12
21	Ceramic Capacitor	PCB	445-8883-1-ND	Digi-Key	$ 0.38	4	$ 1.52
22	Rectagular Connector-Header Female	PCB	S7022-ND	Digi-Key	$ 1.37	2	$ 2.74
23	Rectagular Connector-Header Female	PCB	S7006-ND	Digi-Key	$ 0.65	4	$ 2.60
24	Rectagular Connector-Header Female	PCB	S7004-ND	Digi-Key	$ 0.52	2	$ 1.04
25	Rectagular Connector-Header Female	PCB	S7008-ND	Digi-Key	$ 0.65	2	$ 1.30
26	PCB	PCB	Cusom	JLCPCB	$ 10.70	1	$ 10.70
						Total:	$ 858.69

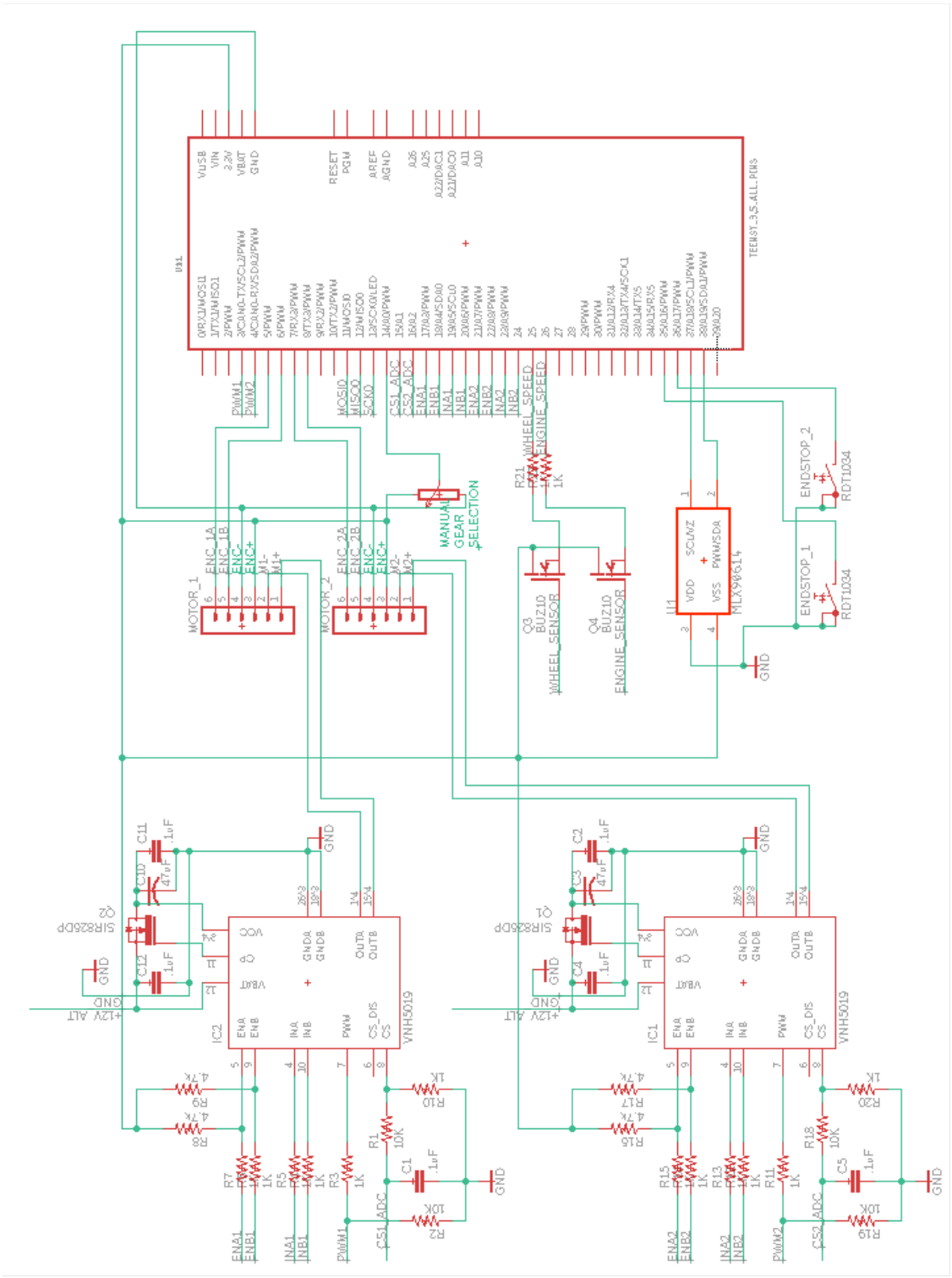
Appendix M – Design Verification Plan

TEST ID	Test Name	Description	Acceptance Criteria	Test Stage	Deadline
1	Motor Position Test	A test to see if the whole mechanical system will meet the required precision.	0.075 of Target Ratio	DV	12/7/18
2	Noisy Power Supply Test	Hook up electronics to dirty power and ensure there are no issues.	Data not Affected (+/-5%)	DV	12/7/18
3	Sensor Test	Testing the individual sensor to ensure they will output the information we need with the predicted precision.	5% of expected	DV	12/7/18
4	Accuracy of Slip Measurement	Compare visual measurements of slip.	Agreement with +/-2%	DV	12/7/18
5	Slip Test	Run the 3D print protype with only position control and ensure that the slip values stay in acceptable margins with varying coefficients of friction.	Belt Slip within 1-7%	DV	12/7/18
6	Submerse Test	Place enclosures in water and check if there is ingression	No water ingression	DV	2/23/19
7	Motor Acceleration Test	Run the sheaves in a similar way as they would in an acceleration run on the 3D printed prototype. Final testing on the car in a true acceleration run.	Comparison to mechanical CVT time	DV	3/8/19
8	Motor Back Shifting Test	Run the sheaves in a similar way as they would in aback shifting scenario on the 3D printed prototype. Final testing on the car.	Comparison to mechanical CVT time	DV	3/8/19
9	Elevated Temperature Test	Running the Final system in elevated temperatures and ensure there is no impact on performance.	Performance Not Affected	PV	3/8/19
10	Hill Climb Test	Run the sheaves in a similar way as they would in an acceleration run on the 3D printed prototype. Final testing on the car in a true acceleration run.	Comparison to mechanical CVT time	DV	3/8/19
11	Brake Check Test	Run the sheaves in a similar way as they would in brake check on the 3D printed prototype. Final testing on the car in a brake check.	Pass Brake Check	DV	3/8/19
12	Endurance Test	Running the Final system for a total of 20 hours to ensure the final product fulfills the lifetime requirements.	20 hours of Life	PV	3/8/19

Appendix N – Unit Test Framework

```
#ifndef UNITTEST_H
#define UNITTEST_H

#include <stdio.h>
#include <string.h>

#define TEST_SIGNED(_ACTUAL,_EXPECT)\
{\
   long _actual = _ACTUAL, _expect = _EXPECT;\
   if (_actual != _expect) {\
      fprintf(stderr, "Failed test in %s at line %d:\n", __FILE__, __LINE__);\
      fprintf(stderr, "   Found substitution %s, value %ld, expected %ld\n",\
         #_ACTUAL, _actual, _expect);\
   }\
}

#define TEST_UNSIGNED(_ACTUAL,_EXPECT)\
{\
   unsigned long _actual = _ACTUAL, _expect = _EXPECT;\
   if (_actual != _expect) {\
      fprintf(stderr, "Failed test in %s at line %d:\n", __FILE__, __LINE__);\
      fprintf(stderr, "   Found substitution %s, value %lu, expected %lu\n",\
         #_ACTUAL, _actual, _expect);\
   }\
}

#define TEST_BOOLEAN(_ACTUAL,_EXPECT)\
{\
   long _actual = _ACTUAL, _expect = _EXPECT;\
   char *actual_verbose = _actual==0 ? "false" : "true";\
   char *expect_verbose = _expect==0 ? "false" : "true";\
   if (strcmp(actual_verbose, expect_verbose)) {\
      fprintf(stderr, "Failed test in %s at line %d:\n", __FILE__, __LINE__);\
      fprintf(stderr, "   Found substitution %s, value %s, expected %s\n",\
         #_ACTUAL, actual_verbose, expect_verbose);\
   }\
}

#define TEST_CHAR(_ACTUAL,_EXPECT)\
{\
   char _actual = _ACTUAL, _expect = _EXPECT;\
   if (_actual != _expect) {\
      fprintf(stderr, "Failed test in %s at line %d:\n", __FILE__, __LINE__);\
      fprintf(stderr, "   Found substitution %s, value '%c', expected '%c'\n",\
         #_ACTUAL, _actual, _expect);\
   }\
}

#define TEST_REAL(_ACTUAL,_EXPECT,_EPSILON)\
{\
   double _actual = _ACTUAL, _expect = _EXPECT;\
   if (_actual - _expect > _EPSILON || _expect - _actual > _EPSILON) {\
      fprintf(stderr, "Failed test in %s at line %d:\n", __FILE__, __LINE__);\
      fprintf(stderr, "   Found substitution %s, value %g, expected %g +/-%g\n",\
         #_ACTUAL, _actual, _expect, _EPSILON);\
   }\
}

#define TEST_STRING(_ACTUAL,_EXPECT)\
{\
   const char *_actual = _ACTUAL, *_expect = _EXPECT;\
   if (strcmp(_actual, _expect)) {\
      fprintf(stderr, "Failed test in %s at line %d:\n", __FILE__, __LINE__);\
      fprintf(stderr, "   Found substitution %s, value %s, expected %s\n",\
         #_ACTUAL, _actual, _expect);\
   }\
}

#define TEST_ERROR(_FUNCTION_CALL)\
{\
   _FUNCTION_CALL;\
   fprintf(stderr, "Failed test in %s at line %d:\n", __FILE__, __LINE__);\
   fprintf(stderr, "   Expected error detection did not occur\n");\
}

#endif
```

Appendix O – Circuit Schematic Blow-up



Appendix P – Motor Data Sheet

maxon motor
driven by precision

DCX32L GB KL 12V

Product specification	
Values at nominal voltage	
Nominal voltage	12 V
No load speed	7120 min^{-1}
No load current	274mA
Nominal speed	6560 min^{-1}
Nominal torque (max. continuous torque)	89.4 mNm
Nominal current (max. continuous current)	6 A
Stall torque	1730 mNm
Stall current	111 A
Max. efficiency	85.5 %
Characteristics	
Max. output power continuous	90.2 W
Terminal resistance	0.108 Ω
Terminal inductance	0.0336 mH
Torque constant	15.6 mNm A^{-1}
Speed constant	612 min^{-1} V^{-1}
Speed/torque gradient	4.24 min^{-1} mNm^{-1}
Mechanical time constant	3.44ms
Rotor inertia	77.6 gcm^2
Thermal data	
Thermal resistance housing-ambient	7.28 KW^{-1}
Thermal resistance winding-housing	2.3 KW^{-1}
Thermal time constant of the winding	45 s
Thermal time constant of the motor	837 s
Ambient temperature	-40...100 °C
Max. winding temperature	155 °C
Mechanical data	
Max. permissible speed	11300 min^{-1}
Axial play	0...0.1 mm
Preload	7 N
Radial backlash	0.02 mm
Max. axial load (dynamic)	7 N
Max. force for press fits (static)	22.6 N

Appendix Q – Gear Box Data Sheet

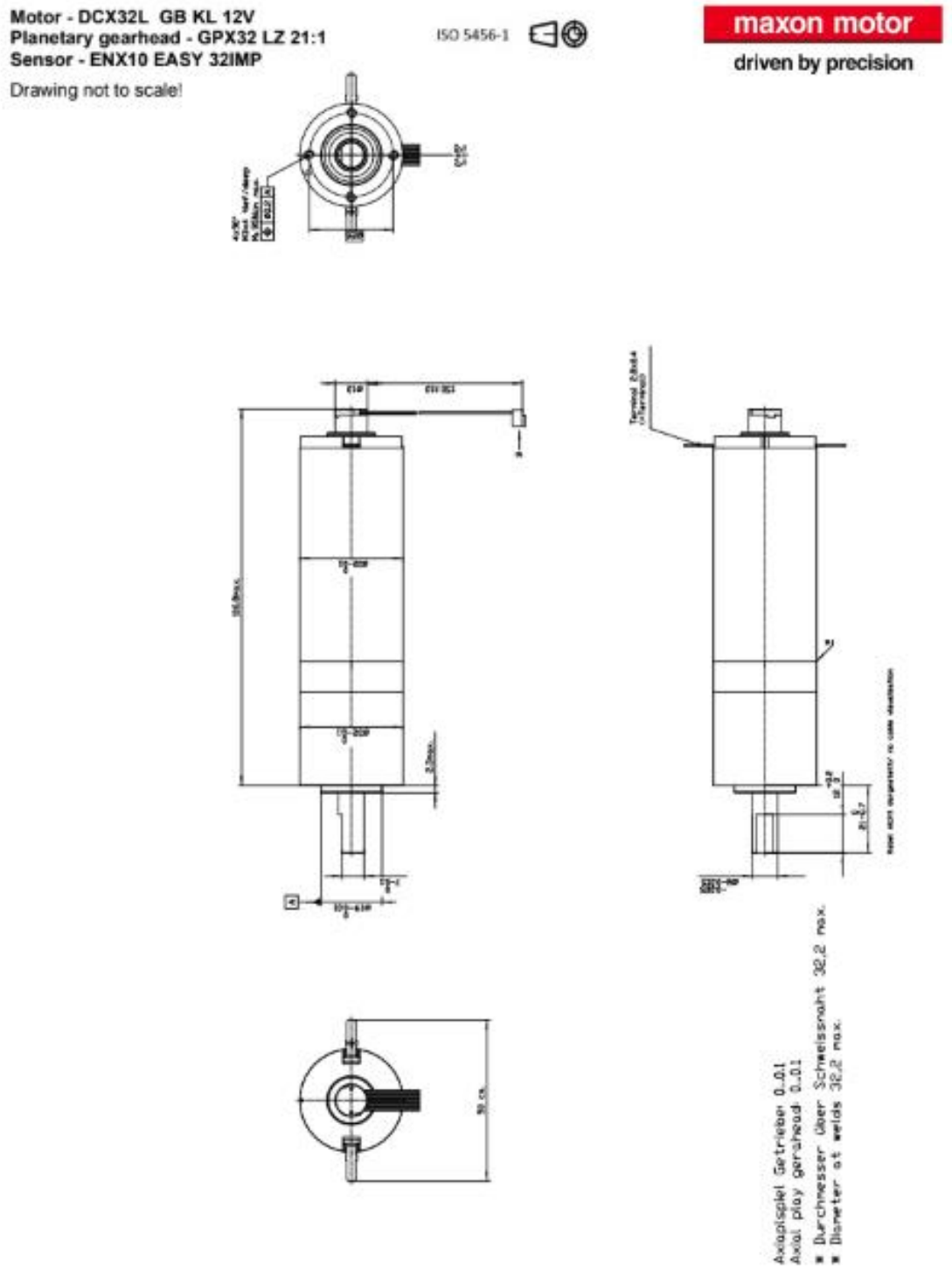
maxon motor
driven by precision

GPX32 LZ 21:1

	Product specification	
Gearhead data		
	Reduction	21:1
	Absolute reduction	5175/247
	Number of stages	2
	Max. continuous torque	2.9 Nm
	Max. intermittent torque	3.6 Nm
	Direction of rotation, drive to output	=
	Max. efficiency	78 %
	Average backlash no-load	0.6 °
	Mass inertia	7.921 gmc^2
	Max. transmittable power (continuous)	50 W
	Max. short-time transferable output	62 W
Technical data		
	Output shaft bearing	KL
	Max. radial play, 5 mm from flange	max. 0.14 mm
	Axial play	0...0.1 mm
	Max. permissible radial load, 10 mm from flange	180 N
	Max. permissible axial load	110N
	Max. permissible force for press fits	120 N
	Max. continuous input speed	7000 min^{-1}
	Max. intermittent input speed	8750 min^{-1}
	Recommended temperature range	-40...100 °C

Information about gearhead data: http://www.maxonmotor.com/medias/CMS_Downloads/DIVERSES/12_203_EN.pdf

Appendix R – Motor-Gear Box Drawing



Appendix S – Teensy Pin Configuration Tests

```
// This optional setting causes Encoder to use more optimized code,
// It must be defined before Encoder.h is included.
#define ENCODER_OPTIMIZE_INTERRUPTS
#include "Encoder.h"
#include <FreqCount.h>
#include <FreqMeasure.h>

#define ENC_A_PIN1 29
#define ENC_A_PIN2 30
#define M_PWM 6
#define M_DIR1 8
#define M_DIR2 9

#define ENC_B_PIN1 31
#define ENC_B_PIN2 32
#define M2_PWM 7
#define M2_DIR1 11
#define M2_DIR2 12

#define HALL_ENGINE 13

Encoder enc_a(ENC_A_PIN1, ENC_A_PIN2);
Encoder enc_b(ENC_B_PIN1, ENC_B_PIN2);

int e, esum;
int setpoint;
double Ki = .5;
double Kp = 1;
double push = 0;

double sat(int push){
   if (push > 255){
      return 255;
   }
   else if (push < -255){
      return -255;
   }
   return push;
}
```

```
void spinMotor(int DIR1_PIN, int DIR2_PIN, int PWM_PIN, double duty){
   if (duty > 0){
      //Spin foreward
      digitalWrite(DIR1_PIN, HIGH);
      digitalWrite(DIR2_PIN, LOW);
   } else{
      digitalWrite(DIR2_PIN, HIGH);
      digitalWrite(DIR1_PIN, LOW);
   }
   analogWrite(PWM_PIN, abs(duty));
}

double PIcontrol(double setpoint, double curr_point, double Kp, double KI){
   double error;
   error = setpoint - curr_point;
   static double error_sum;
   error_sum += error;
   return sat(KI*error_sum + Kp*error);

}

double myinterp1(double *sample_points, double *corr_points, double que_value){
  int i = 0;
  double x_1;
  double x_2;
  double y_1;
  double y_2;
  while (que_value < *(sample_points + i))
  {
    x_1 = *(sample_points + i);
    x_2 = *(sample_points + i + 1);
    y_1 = *(corr_points + i);
    y_2 = *(corr_points + i + 1);
    i++;
  }
  return y_1 + ((y_2 - y_1)/(x_2 - x_1))*(que_value - x_1);
}

void setup() {
  Serial.begin(38400);
  Serial.println("Encoder Test:\n");
  pinMode(M_PWM, OUTPUT);
  pinMode(M_DIR1, OUTPUT);
  pinMode(M_DIR2, OUTPUT);
  pinMode(M2_PWM, OUTPUT);
  pinMode(M2_DIR1, OUTPUT);
```

```
  pinMode(M2_DIR2, OUTPUT);

  analogWriteFrequency(M2_PWM, 375000);

  FreqCount.begin(1000);
  FreqMeasure.begin();
}

long prevPos = -1;
long prevPos2 = -1;

long newPos;
long newPos2;

double sum=0;
int count=0;

int i = 0;
bool goUp = true;
void loop() {
  newPos = enc_a.read();
  newPos2 = enc_b.read();
  if (newPos != prevPos){
    prevPos = newPos;
    Serial.println("A");
    Serial.println(newPos);
  }
  if (newPos2 != prevPos2){
    prevPos2 = newPos2;
    Serial.print("B ");
    Serial.println(newPos2);
  }

    // Generate a in/out motion
    if (goUp){
        i++;
        if (i >= 255){
            goUp = false;
        }
    } else{
        i--;
        if (i <= -255){
            goUp = true;
        }
    }
    delay(10);
```

```
  // End motion

 spinMotor(M_DIR1, M_DIR2, M_PWM, 50);
 spinMotor(M2_DIR1, M2_DIR2, M2_PWM, -50);
 //Serial.print("Position: ");
 //Serial.println(i);

  if (FreqMeasure.available()) {
     // average several reading together
     sum = sum + FreqMeasure.read();
     count = count + 1;
     if (count > 30) {
       float frequency = FreqMeasure.countToFrequency(sum / count);
       Serial.println(frequency);

     }
    }


}
```