Two Wheel Drive Electronic Differential

NC State Engineering at UNC Asheville
Senior Design 2014

Jennifer Cory	Nathan Lewis	Jason Shores
Nick Faught	Jason McCrary	Bryan Short
Robert Fussell	Cullen Reed	Rachael Stanfield
Jason Hopper	Hallie Sheaffer	Gus Tabaileh
Dakota Lazenby		Brandon Zschokke

Abstract—The 2013-14 UNC Asheville / NC State Mechatronics Senior Project team designed and built an electronically controlled, differentially driven, two wheel drive vehicle. The vehicle employs a student-designed and constructed drive system; integrating two electric motors, two controllers, and necessary sensors under the supervisory control of a microprocessor. The vehicle hardware allows for comparison of two rear axle configurations: 1) a single, solid axle, with both wheels rotating in unison; and 2) a pair of independent axles, each with its own motor, controlled to provide differential operation. Appropriate control schemes were implemented via a feedback loop through the microprocessor. Standard automotive industry tests were adapted to gauge the change in performance between the modes. One primary motivation for the project is to provide a "proof of concept" demonstrator. With further refinement beyond the work presented here, future student teams will be able to use the vehicle to showcase the UNC Asheville / NC State Mechatronics Program's capabilities to potential academic collaborators.

1 Introduction

1.1 Institutional Background

The Mechatronics Engineering program at UNC Asheville is a hybrid of the electrical, mechanical and computer engineering disciplines. EGM 484 and 485, the classes which constitute the senior design project, are designed to allow the students creative license on a project that uses all of the skills learned in the first three years of the program. The project also introduces project management techniques such as budgeting, goal setting and task management.

Formula SAE (FSAE), sanctioned by the Society of Automotive Engineers, offers a highly attractive venue for learning, applying, and demonstrating automotive engineering skills in a relevant, fun, hands-on competition. Unfortunately, successful FSAE efforts demand a level of resources beyond the present capabilities and budget of the NC State / UNC Asheville Mechatronics program; however, because Mechatronics specializes in the computer control of electromechanical systems, it is reasonable to think that UNC Asheville could serve as an electric drivetrain "subcontractor" to an existing FSAE team on another campus. Such a joint venture would provide a staged entry into FSAE for UNC Asheville, with the possibility of building toward a completely in-house effort over time.

The goal of this year's senior project is to design and build a vehicle employing an electronic differential. The underlying goal of the project, in demonstrating an improvement in performance via this differential control, is the pursuit of increasing the likelihood of a larger team adopting the UNC Asheville/NCSU program as a subcontractor for its expertise in control of integrated electronic and mechanical systems. Since the drive system's control is the most technologically challenging aspect of the project, the successful implementation of a low power, two-wheel drive system on a

simple vehicle should serve as an adequate advertisement for the Mechatronics Program's capabilities. As a result, success in this collaboration allows for a more competitive electric vehicle in teams that have previously been traditionally divided between strictly mechanical and electrical engineers. This initial step taken by the current senior design group also allows future undergraduate students to continue improving upon the existing control systems, as well formulating and developing partnerships with other universities. In addition, setting this framework increases general awareness of the Mechatronics program at UNC Asheville in attempts to create potential to attract new students.

1.2 Technical Background

In addition to increased energy efficiency and a reduction in harmful emissions, powering a vehicle electrically offers opportunities for significant improvements in stability and handling. This increased stability and traction is particularly useful in conditions where traction is inconsistent or compromised by conditions such as rain, snow, and off-road situations. Differential control of the driven wheels facilitates an improvement in cornering precision and reduces the net power demand from the drivetrain. Independently driven wheels also have a place in motorsports, allowing for increases in cornering speeds and increased handling. This increase in performance is due to better controlling the vehicle dynamics. When cornering, the wheel on the outside of the turn rotates faster than the one on the inside. Most mechanical differentials compensate for this through a system of gears that allows one wheel to spin slower than the other when the vehicle is in a turn. In a mechanical differential, the average of the angular speeds of the the drive wheels is equal to the rotational speed of the input shaft to the differential. The senior design team's experimental system mimics these mechanical systems on our split axle, electric system.

The design of the electronic differential keeps both wheels under power and is tuned to differ the throttle to each motor to allow variable wheel speeds while still providing the requested vehicle velocity around a turn. The difference in speeds depends upon the turning angle, measured from the steering shaft, and the throttle input. These user inputs are then used in calculations inside the microcontroller to decide the theoretical difference in wheel speeds and then to vary the throttle out of the controller accordingly. This is the functionality of the open loop control scheme. The closed loop control then takes the wheel speed, referenced from the encoders attached to the motors, and verifies the effect of differing the throttle to the two motors. If the difference is not enough, then our control law then increases the difference in throttle to increase the difference in wheel speed. Equivalently, should the wheel speeds be too far apart, then the control law reduces the difference in throttle applied to the motors to bring the wheel speeds closer together.

2 Project Overview

The project team chose the electronic differential project and defined its success as the demonstration of improved performance when utilizing the differentially controlled rear wheels in comparison with the solid axle. In pursuit of this success, the team had to decide what parts to acquire and how to assemble the parts into a coherent and effective design that would demonstrate the performance increase sought after.

The initial decision made was that the vehicle must be capable of at least two modes of operation:

1) The first mode must allow the separate drive axles to be coupled mechanically into a single solid axle. The control system would then allow the motors to exert the same total force available in the second mode by applying equal amounts of throttle, and thus equal amounts of drive force to the live axle.

2) The second mode must involve two independently driven axles, each with its own motor. This type of control would then allow the two sides to turn independently at speeds determined by the control system. The control system must include the necessary components

and code to discern the driver's intentions and provide the correct response.

The team, after having made the specifications listed above, then commenced to do research as to what motors and controllers could be acquired, what chassis to install the system on, and what sensors and controller to employ in creating the completed system. The major factor that guided decisions concerning parts acquisition was budgetary restrictions. After having obtained most key parts from donations, the team then purchased, from the budgeted fund, parts to complement the donations.

After acquiring most of the parts, construction ensued. This construction many problems. Fighting the bent chassis, installing a full electric system on a chassis meant for a small gasoline motor, and many other problems presented themselves as the team furiously attempted to construct the system. Construction, as a result of these problems, took far longer than expected and left little time for controller design and fine tuning. Nearing the end of the project deadlines, the team found that there was just insufficient time to devise a complicated and well oiled controller and thus resolved to implement a rudimentary controller and then provide sufficient notes to allow for improvement by later teams.

3 Preliminary Design

3.1 Objectives

The primary objectives of the team were as follows:

1) Acquire parts to construct a fully electric vehicle and implement a control system on the vehicle all within budgetary limitations of two-thousand dollars of expenditure.

2) Construct the electric vehicle to hold at least one person (driver).

3) Construct a drivetrain that allows for two independent motors driving two independent axles and wheels, while allowing the axles to be coupled for "live axle" driving.

4) Implement a control system to provide variant throttle to the motors, thus controlling the wheels speeds in a differential manner.

5) Retain a sufficient level of safety for the driver to prevent severe bodily harm in the event of system mayhem, and potential loss of user control.

3.2 Team Management

The project team was initially split into three main divisions with groups of team members assigned to sub-teams. Each team reported to a designated overseer. The three sections were:

1) Fabrication/Mechanical: The team working on this section was responsible for carrying out all of the mechanical operations necessary to create a working vehicle. The donated chassis required significant mechanical modification to satisfy the project requirements. These modifications mainly consisted of converting the vehicle from a single axle gas powered setup to an electronic two axle system.

2) Electrical: The vehicle drive system consists of several electrical components, including sensors, a microcontroller, motors, and programmable motor controllers. It was the responsibility of the electrical team to research, procure, and implement these items.

3) Programming/Controls: The team working on this section was responsible for implementing the algorithms necessary to achieve differential control. This requires programming code to take sensor input and produce a controlled output.

In addition to their specific responsibilities, each team was also responsible for documenting the processes used to achieve their goals. Operating in this manner assured that future teams could refer to the notes set forth by this team in attempts to continue and progress with this project, or to use parts of it to aid with other future projects. By the end of this project, the teams dissolved and formed a smaller group. The smaller group consisted of the most knowledgeable members of the prior group, these leaders then collaborated as one "interdisciplinary" team.

3.3 Models

3.3.1 Chassis

One of the first models created was a *SolidWorks* model of the donated chassis. The 3D

model allowed the team to visually inspect and determine proper locations of various components before permanently mounting them to the chassis. The chassis model can be seen in Figure 1.

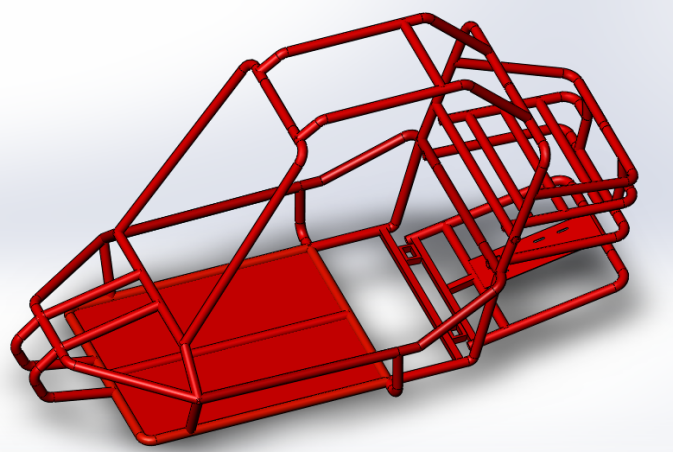


Fig. 1: Initial SolidWorks Model

3.3.2 Controls

In order to implement the control system necessary for this project, an initial top level block diagram was created. This block diagram would provide an outline that would guide the design of the control system. In order to properly control two separate electric motors, it was determined that inputs from a variety of sensors were needed. These sensors included various potentiometers to measure the input of the throttle and steering angle, as well as multiple encoders to measure the individual rear wheel speeds. The initial model was designed in *Simulink* and built on a pre-existing model provided by *MathWorks, Inc.* The source location for the initial model can be seen in the references section. The original model was adapted to incorporate the sensors needed as well as conform to a particular control scheme that would meet the needs of the project. The design of the initial top level block diagram can be seen in Figure 2.

3.4 Power Calculations

One of the first tasks was the calculation of required power to drive the kart. The first step in this process was to decide upon a satisfactory speed for testing the split versus solid

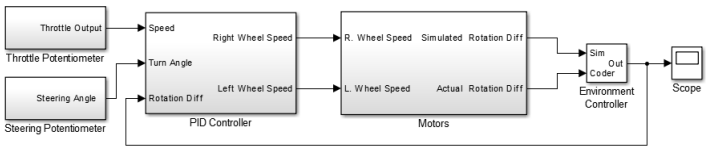


Fig. 2: top level block diagram

axle configuration. After some rudimentary research into standard vehicle performance tests, It was agreed that a maximum speed of twenty miles per hour would be adequate for testing.

The second step was to gather data about the karts rolling resistance. Simulating a fully loaded chassis was done by loading the cart with three people. A linear force gauge was attached to the front of the kart and used to measure the amount of force required to accelerate it to a constant speed. Throughout this process, readings of the force gauge were recorded by video. Several runs were performed to ensure consistent data. A graph from this set of data can be seen in Appendix B.1.

The rolling resistance data gathered were then used to calculate minimum power requirements. Several estimations were made including frontal area of the kart, final vehicle mass, and a drag coefficient. Calculations showed that a minimum of 1.25 horsepower was required to maintain a 20 mile per hour speed on flat ground, under the ideal assumptions made. For the same assumptions and a 10% grade, 3.35 horsepower was required. These calculations were utilized in finding a pair of motors that suited the project requirements.

A spreadsheet containing calculations for power requirements on both a 0% and a 10% grade can be seen in Appendix B.2.

3.5 Budget

In order to successfully complete the project, a variety of parts were needed. The parts needed by the senior design group consisted of parts to build or repair a chassis, items for safety implementation, the core drive-train equipment, various sensors and microcontrollers needed for the control system. The senior project was allocated a budget of two thousand dollars.

During the first phases of the project, it was decided that the budget would be partitioned

into sections including the motors, motor controllers, sensors, and chassis construction or repair. After some preliminary research and projections of total cost based on similar projects, it was apparent that the parts required to build the proposed system would not be easily obtained under the budget cap. This meant that many parts would have to be acquired through donations and discounts. After the successful acquisition of a chassis, motors, controllers, and a couple of batteries from donations, many of the remaining required components were obtainable given the budget constraints.

4 Design Implementation

4.1 Overview

First, the design had to undergo a significant amount of mechanical modification. The chassis that the team had donated was in a state of disrepair from the beginning. The frame was bent, and to retrofit the electric parts onto the kart was no small task.

The electrical systems had to be configured to include the three twelve-volt batteries in series and then use them to power the motors and controllers. A schematic provided with the motor controllers was used to aid in this process.

Lastly, the differential motor control scheme was implemented with an Arduino microcontroller. Using the *Arduino Integrated Development Environment*, code was written that takes inputs from the various sensors and sends outputs to the motor controllers. The control scheme uses potentiometer readings and encoder values to feed algorithms that were designed to differentiate the wheel speeds. The controller uses the computed values to send throttle input back to the motor controllers to control the wheel speeds.

4.2 Mechanical

4.2.1 Chassis

The steel chassis used for the electric vehicle was modified from a gasoline powered go-kart originally designed for children. A drawing of the frame is depicted in Figure 3. When it was donated, it was discovered that some of the components had been damaged and needed to be repaired or replaced. These components included the shocks, steering column, rear axle and much of the front suspension. In addition to these repairs, chassis modifications were required to accommodate an electrical control system.

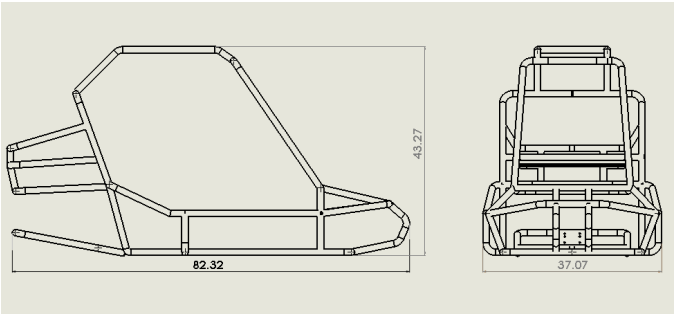


Fig. 3: Side and Front View of Chassis

The chassis' main modifications were driven by the need to repair broken parts and to provide places to retrofit parts. The first major repair that was made was altering the rear axle. The rear of the kart was bent and had to be straightened before splitting the solid rear axle and reattaching it with new bearings to support the cut ends. Also, the kart was subjected to the tightening of the suspension and steering. The damaged steering links were replaced and the column eyelets were tightened to eliminate wheel play. This resulted in increased physical control and maneuverability of the kart, as well as a better platform for the steering angle sensor. These steering modifications also worked to provide better accuracy in test repetitions, as it had less play and could hold more consistent turns.

After repairing and improving the suspension and steering, the kart was fitted with mounts for a battery box and for the controllers. The bench seat was removed and replaced with a single seat to make space for the additional hardware components, especially the battery box. Batteries were secured in the space to the right of the driver's seat. Structural members were welded to the floor to support the additional weight of the batteries. In addition, stronger shocks were installed to replace the broken ones and to support the extra weight associated with an electronic control

system.

Lastly, a shield of Plexiglas was installed between the driver's back and the high voltage components to increase the safety for the driver. Later, a wooden shield was fabricated to cover the top of the high voltage components to protect them from debris and to provide a platform on which laptops could be rested whilst programming and testing in the lab. Details on the modifications made can be found in their respective sections below.

4.2.2 Rear Axle

To prepare for the differential control of the wheels, the solid rear axle that came with the kart was split into two pieces. The method of securing wheels and sprockets on the axle employed a key-way and notches in the items being mounted on the axle. The existing key-way was extended across the entire length of both sections of the axle to prepare for mounting more items on it. Extending the key-way made it much easier to position and attach the drive train components. These components included two sprockets for a dual-motor chain drive system, pillow bearings and a coupling that allowed for a solid or split axle configuration. One of the sides had to be straightened with a press to repair pre-existing damage from bending of the axle. The chassis also required modification to support the newly split axle. To account for this, the frame was extended downward along the center platform above the axle by means of two plates spaced out by steel rod standoffs and then welded together. Bearings were then attached to the newly added plates and the cut ends of the axle were inserted into the new bearings.

Despite the initial attempts to fully support the split axle halves, they experienced shifting when subjected to hard cornering forces. The shifting was eliminated by adding screws that passed through bearing collar set screw holes and directly into the axle shafts. These screws acted as a shear pin to resist the axial loading that is applied to each axle while turning, thereby preventing lateral axle shifting inside the bearings.

4.2.3 Front Wheels & Suspension

The front of the kart came configured with an A-arm style suspension system. The system employed two arms pinned to the frame that could rotate in the x-z plane (assuming that the ground is the x-y plane with the y axis running from front to back of the kart). The front wheels were then mounted to these arms by way of a kingpin (oriented along the z-axis). Originally, the wheels sagged inward due to the mount holes being worn out by the kingpin that results from strenuous use. This was fixed by welding on machine bushings to both ends of the mounting tube and replacing the kingpins with steel carriage bolts.

The shocks that came with the kart needed to be replaced as well. One of them was bent and disallowed movement in any direction, rotational or compression. In addition to the one shock being broken, all of them were underrated for the projected weight of the modified vehicle. They were also repositioned in the front of the vehicle to help with the camber and toe angles caused by the A-arm suspension. The new shocks were chosen because they are rated to support 730 pounds of load compression each, and three inches of compression length.

4.2.4 Steering Column

The steering column needed repairs due to being sloppy and bored out at the mounting holes. The original tie rod was bent and had to be completely replaced, and the holes were tightened to clean up the some of the play. Machine bushings were welded to the mount holes here as well, to tighten them up.

After tightening up the steering as much as possible, the steering column was chosen as the most reliable place to take a reading of the steering angle. Because of this, a potentiometer shaft was inserted into the steering shaft and the body of the potentiometer was secured to the chassis. The potentiometer acts as the sensor and was chosen for its simplicity, ease of coding, and modest expense. Originally the column was a flat-ended, solid steel rod. Using a lathe, a hole was drilled into the end to accommodate the adjustable shaft of a potentiometer.

A mounting bracket was fabricated to hold the potentiometer in place as the steering column rotates, and to reduce the axial and shearing stresses acting on it. The potentiometer shaft was then centered and tied into the steering shaft with a set screw such that it rotated in response to the steering column.

4.2.5 Seat

The original seat in the kart was designed for two passengers. However, it was decided early in the design process that the second seat needed to be removed to create space for other components. In addition, with the original seat, drivers who were taller than about 5 feet and 7 inches could not operate the vehicle. Both of these issues were resolved by installing an alternative seat. The new seat was made from a plastic, cloth, and foam portion of a standard school chair without the stainless steel legs. The new design used the runners from the old seat to adjust the height so that taller people could operate the kart. Using the original runners also allowed the vehicle to still make use of the original seat belt.

4.2.6 Battery Box

The most significant addition to the kart, in terms of size and weight, are the three 80 lb batteries required to power the motors and controllers. Several accommodations had to be made to safely accommodate the batteries when installing them on the kart. A steel battery tray and wooden box were fabricated to secure the batteries from sliding around and to protect the driver from any potential malfunctions, meltdowns, etc. First the passenger side of the kart was modified to support the extra weight of the batteries by welding cross members to the tube steel which ran lengthwise down the kart. The battery tray was then constructed of L-channel steel and welded to the chassis on the cross bar supports. The box was constructed of plywood and had a hinged door on the top for battery access. The location of the battery box was chosen such that it would minimally affect the center of mass (the calculations for the center of mass are in Appendix B.4).

The weight of the batteries and battery box essentially replace, and possibly double, the weight of the second passenger in the old seating arrangement and places the center of mass three inches to the right of the true center. Placing the box beside the driver also allowed it to serve as a mounting platform for the control panel and arduino housing.

4.2.7 Motor Mounts

The motors obtained for this project were electric DC motors meant for a golf-cart. They were designed to bolt directly into a mechanical differential in the rear of the golf-cart. Because of this configuration, they required motor mounts to be designed and built that would eventually accommodate a sprocket mounted near the face of the motor. The size of the motors, position on the kart, and a mechanism for tightening the chain were all considered in the mount design. Because the motors were designed to bolt directly into a mechanical differential, the housing was used to provide support for the front of the motor and to cover the armature windings. Taking this into consideration, the motor mount design included a front motor plate which could both secure the motor, cover the windings and provide support for the output shaft on which the sprocket was mounted. The drawing depicting the design of the motor and its mount can be seen below in Figure 4.

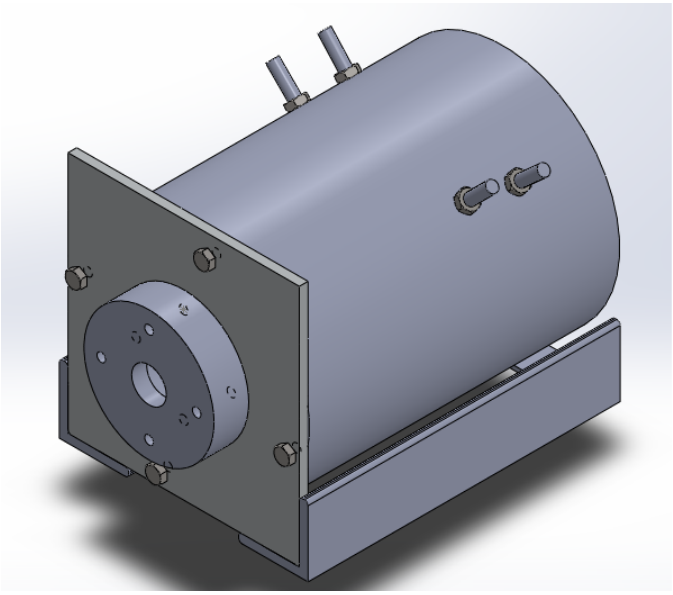


Fig. 4: SolidWorks Design of the Motors

The motor mounts were constructed from four separate pieces. A front motor plate was cut from 1/4 inch steel and drilled to accept the motor and bearing cap mounting bolts. The second and third pieces were a pair of 3/16 steel angle iron that were welded to the front plate and ran along the length of the motor. These pieces of angle iron served as the medium for the oblong mounting slots that allowed the whole bracket to slide, thus solving the chain tensioning problem. The final piece of the bracket was a small iron cross member welded across the back of the angle iron to provide more support to the whole setup. Once welded together, the motor was mounted to the faceplate using allen head bolts. In addition to the holes for the motor mount bolts, the face plate was also drilled and tapped to accept bolts that held an aluminum bearing cap. The bearing cap is described in more detail in the next section.

4.2.8 Output Shaft & Bearing Caps

Because the motors used in this project were designed to be attached directly to a mechanical differential it was necessary to find an alternative means to distribute the power. Normally, bolting directly to the differential meant plugging the motor onto a shaft built into the device; however, in this design?s particular condition, the motor needed to be fitted with a free standing shaft that could be made to accept a sprocket for a chain. The necessary component for this process was a nineteen tooth output shaft that normally connected the motor to the gear train of the differential unit. The original shaft had a toothed section that interfaced with those gears. The shaft also had a stepped end that fit into the motor socket and had bearings at both ends.

The bearing on the free end of the shaft and the toothed section that normally interfaced with the differential weren?t needed for this design. The teeth of the gear were turned down using a lathe and a keyway was milled into the turned down section to accommodate the sprocket necessary for a chain drive configuration. This solved most of the problems associated with the output shaft, but there was still the problem of how to keep the shaft held into the motor. This issue was addressed by designing and machining caps to fit over the bearing at the motor end of the shaft and hold the shaft into the motor. The bearing caps were also designed to prevent any driveshaft movement from axial forces (pushing or pulling on the shaft). The bearing caps were machined out of aluminum and were drilled to bolt to the faceplate of the motor mount.

Issues arose when the central openings in the bearing caps caused binding on the shaft because they were turned down with too low of a tolerance for non-rotational movements in the shaft. The restriction of deflection at the free end of the shaft, caused by low machining tolerances, in turn caused the shaft to severely rub against the aluminum caps when loaded with the moment caused by the chain tension. The resultant binding was relieved by boring out the bearing cap openings to higher tolerances. Also, it was found that the bearing area of the caps was bored out too deeply and with too large of a diameter. Bushings were added to the caps, taking up the excess depth and allowing the bearings to be pressed into the caps harder and preventing the bearing from moving along the shaft within the cap. In addition to the bushings, small shims were added between the outer shell of the bearing and the inner circumference of the cap to tighten up the bearings against the cap.

4.2.9 Chain Drive

In order to transfer torque from the motors to the wheels, a chain drive system was used. Number forty roller chain with a single strand working load of 810 lbs was chosen due to the relatively large amount of torque (30 ft-lbs each) output from each of the motors. To reduce the overall stress on the system and to allow for the differential control of the wheels, one chain was used for each motor. A pair of sprockets keyed to the motor output shaft and axle shaft, respectively, were implemented to engage the chain system. As mentioned previously, chain tensioning was implemented through the use of sliding motor brackets. Once moved to create the desired tension, they were secured to the frame with a set of bolts.

Choosing a gear ratio was a process that required some experimentation. It was first decided that a ratio of $10:1$ would be adequate for testing. This ratio was chosen because it would allow to the motors to turn at maximum speed when the kart was traveling twenty miles per hour. After searching for parts, it was discovered that the sprockets needed to create this ratio were not available for the axle. There was also concern that this high of a ratio might create too much torque. It was then decided that a 2.66 ratio would be easier to implement due to the smaller sizes of the sprockets, and the availability of the parts. This setup; however, caused the motors to draw too much amperage at startup, failed to provide enough torque to provide adequate acceleration or hill climbing ability and allowed the kart to achieve borderline dangerous speeds. Finally, a $5:1$ ratio was settled upon. This ratio gave the best balance between top speed and acceleration that was feasibly implemented with available parts.

4.2.10 New Pedal (Throttle)

The original throttle system for the kart was not suitable for the project. Its original purpose was to pull a rod that ran from the pedal to the rear of the kart where the throttle cord was attached to the motor. In addition to being an unsuitable configuration to accept a throttle sensor, the amount of freedom concerning side to side motion of the pedal was unacceptable for the potentiometer sensor that was going to be installed. In response to these problems, the rod running to the rear of the kart was scrapped, the pedal was removed from the kart and a new configuration was designed to clean up the side to side motion while also accepting easily a potentiometer as a sensor. A drawing of the proposed configuration is shown below in Figure 5.

The new configuration consisted of a pedal mount that was fabricated from a piece of steel square bar and angle iron. The pedal was detached from the old setup and then secured to the square bar with a copper pin, acting as the axis of rotation, was secured into the slot that the pedal rotated about. Finally, the copper pin was coupled to a potentiometer shaft. When

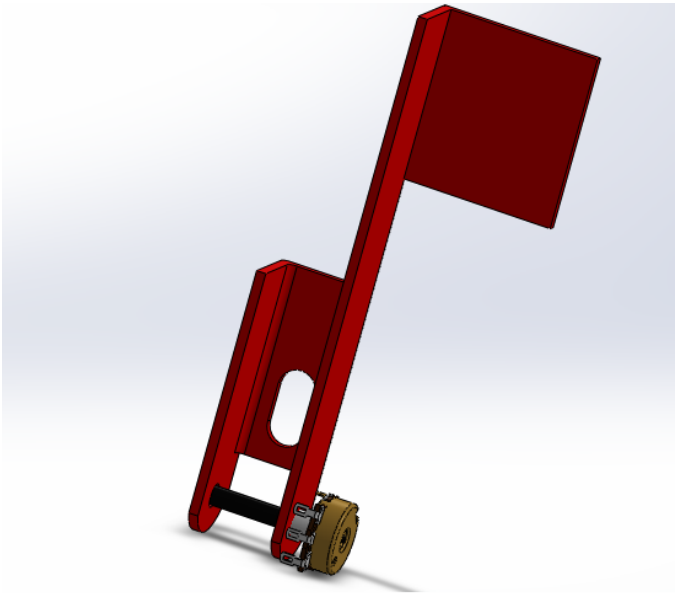


Fig. 5: Throttle Pedal

the pedal was pressed, it rotated the copper pin, which in turn rotated the potentiometer. This acted as a throttle sensor. Also included in the new configuration was a more powerful tension spring as opposed to the compression spring originally on the kart. The new spring provided more force to return the pedal to its original position. The new configuration was mounted on the floor pan next to where the original setup was mounted.

4.2.11 Brakes

Originally, a single brake drum with a cable actuated band was installed on the kart. Due to the splitting of the live axle, this setup only provided braking on one side of the kart. To solve this, another brake drum and band were purchased and installed on the opposite side of the kart for additional safety and stability whilst braking. Immediately after installing the complimentary setup, when comparing the two brakes, it was noted that the original brake drum wasn?t as effective as the new one. The original brake drum was replaced to create uniformity.

A couple of issues arose during the implementation of the new braking system. The issues needed to be addressed to let the brakes act in unison. While the original brake cable worked fine for the old brake, there were no

parts available from the manufacturer to actuate the new one. In place of the OEM brake actuation cable, a motorcycle brake cable was used. It had the sufficient length and strength required for the system, and was relatively easily obtainable. A custom bracket was fabricated to connect both brake cables to the rod attached to the brake pedal, allowing both brakes to be actuated simultaneously. The original rod was replaced because it was not long enough to allow sufficient movement to fully release the brakes.

4.2.12 Control Panel

A control panel was designed that featured switches and connections according to the motor controller schematic. The panel was required to route wiring to its necessary locations and to house the prescribed operations such as the emergency stop, vehicle on/off key switch, rotary forward/neutral/reverse switch, pedal interlock, and mode selection. Most of the controls were put in place as per the schematic provided in the motor controller manual; however, the interlock switch was left out and an emergency stop was added. The emergency stop button enabled the driver to quickly break the connection of the control power to the motor controller, thus disengaging the solenoid and killing power to the motors. The pedal interlock switch was fed into the appropriate pins of the motor controllers to shut the field current to the motors off when the switch was disengaged, thus adding a secondary layer of safety to the system. The mode selection was a combination of two switches that allowed for four modes of programmed motor controller operation. A drawing of the control panel can be seen in Figure 6, and its specific dimensions have been provided in Figure 7.

The panel was made from plexiglass and was supported with medium sized angle brackets. It was positioned on top of the battery box at an angle that is efficient for the driver to press the emergency stop button with little effort. The panel is powered by the 36 Volt battery pack and is hardwired to each controller. While all main control signals run through this panel, all of the actual signal processing and throttle control is left to the microprocessor.

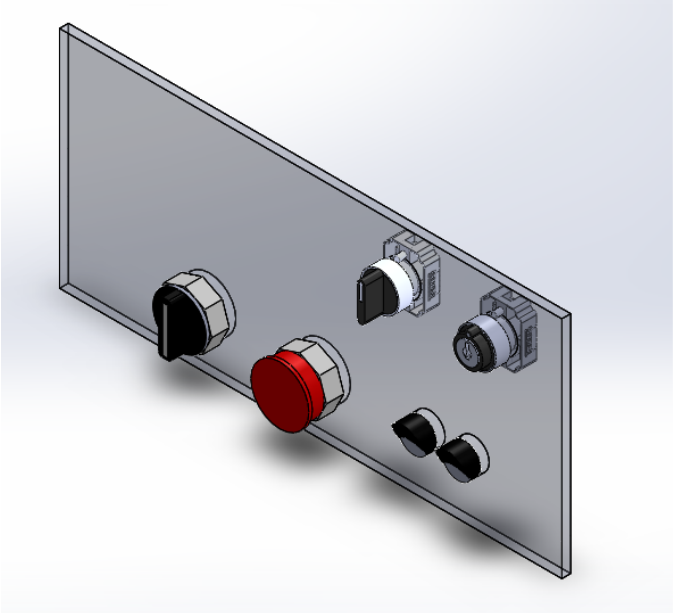


Fig. 6: Control Panel

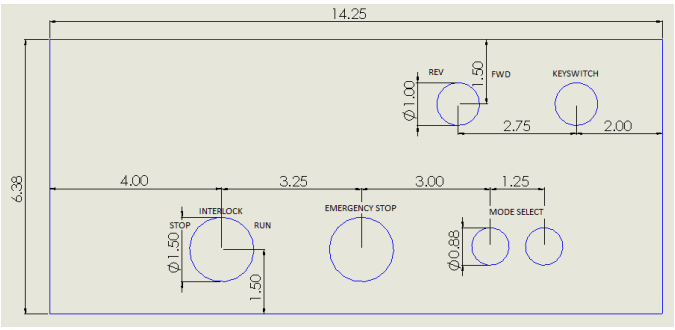


Fig. 7: Control Panel Diagram

4.3 Electrical

4.3.1 Motors

Two 36 Volt EZ-Go golf cart motors were donated and then taken into Carolina Cars and Clubs to be tested very early in the design process. Once confirmed that they were both in working condition, attention was turned to finding appropriate motor controllers and batteries to operate them. A picture of one of the motors is provided in Figure 8.

The motors produce a fairly flat torque curve with a peak of 30 foot pounds at startup. Each motor is capable of drawing nearly 400 amps. Because they are very powerful relative to what was calculated as sufficient horsepower to propel the kart, safeguards were imposed to limit their maximum output. The controllers obtained for the motors allow current limiting.

A torque curve for these motors can be seen in Appendix C.1.

Other than the mechanical caveats stated above, one motor proved to be quite problematic. When picking up the motors at the beginning of the project, it was noted that one of the armature windings seemed to be bent slightly out of place, but the team was assured that this would not cause problems. Late into the project timeline, the motor was acting poorly and the motor controller complained of "field winding faults." Eventually it was discovered that the bent armature windings had rubbed against the field windings during rotation sufficiently enough to tear through the fiber tape holding the field windings and break the winding. In response to this problem in conjunction with the closing deadline and lack of budget, the team decided to perform surgery on the motor. The motor was dismounted from the kart and dismantled. The armature windings that were bent and uncoated were pushed back into place, and an epoxy coating was applied to them to prevent shorting. The broken field winding was pulled from beneath the tape, ran to the bottom of the windings and soldered back together. Any exposed field windings were also covered with epoxy and the motor was reassembled. The motor, upon being reinstalled onto the kart, proceeded to perform appropriately.

Pictures of the deformed and broken windings can be seen in Appendix C.3.

Fig. 8: Motor used in Drivetrain

4.3.2 Motor Controllers

The Curtis motor controllers used on the kart were donated from E-Z-GO golf carts based out of Atlanta, GA. The controllers were donated from a shelf of parts that were no longer getting used at E-Z-GO, and thus were easily repurposed for academic use. E-Z-GO also donated a manual that included a wiring diagram for the controllers and some general guidelines for programming them. In addition to the manual, the controllers came with a programming unit that was used to display an impressive amount of data about the controller and motor including throttle percentage, motor currents, voltages, faults in the system and much more.

The Curtis motor controllers have a wide range of functionality. They can output information such as the voltage and current being supplied to the motors in real time. This was immensely useful when testing some of the control code. They can also be programmed to set parameters such as a limit on the throttle response or top speed and much more. The programmable parameters were carefully gone through and values were established to make the motors perform optimally for our proposed system. The throttle type was changed to accept a "sensor" input, which was provided by the Arduino microcontroller. The acceleration, speed and other performance variables were chosen to run the motors in an acceptable range of current while attempting to get the performance characteristics desired for the differential control. Additional safeguards were set via the programming unit to account for potential malfunctions, and several system issues were caught using the fault output of the programmer that would have been nearly impossible to diagnose otherwise.

The motor controllers were mounted in the rear of the kart on the rack behind the driver's head. The mount was fabricated from a large piece of aluminum. Aluminum was chosen due to its light weight and ability to dissipate heat quickly. The location was chosen such that the controllers would be close to the motors, yet somewhat separated from the driver. In addition to being optimally placed in relation

to the motors, the controllers were installed on the top, left rear section of the kart for ease of access for programming. Being set on the rack provided a great platform to make the controllers accessible, but it also allowed air to move over and under the controllers to increase heat dissipation through convection. A final perk to the choice of mounting the controllers behind the driver was that their placement helped to offset the right to left influence of the battery weight on the passenger side of the kart.

4.3.3 Solenoids

Two Trombetta Bear P/N 114-3611-010 solenoids were obtained to act as relays that open and close the high current motor circuitry using a low current input. The low current input is engaged by rotating the key switch into the on position. This provides control power, low current 36V power, to the motor controllers. The motor controllers then drop a programmed amount of voltage across the low current side of the solenoids, thus activating the coils. These bring the metal contacts into place with a magnetic field to close the loop of the high power circuit. The circuit connecting the batteries to the motor controllers and the motors is initiated by the keyswitch via the solenoids as shown in Figure 9. These particular solenoids are capable of sustaining a resistive or inductive load carry or interrupt of 225 amps with a maximum inductive inrush capacity of 600 amps.

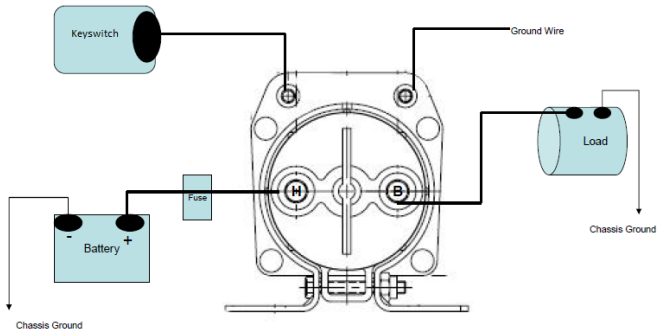


Fig. 9: Schematic showing connections with the solenoids

4.3.4 Chassis Ground

A chassis ground was used in an attempt to reduce the amount of high amperage wiring required in the vehicle. To implement the chassis ground, studs were welded to various points on the chassis to serve as grounding points. This allows the ground side connections from each speed controller to be terminated at the closest chassis point, rather than running wire back to the ground battery post. This also allows the ground side of the battery pack to require only a single point of connection (a short distance from the battery) to complete the circuit.

Initially, both the high amperage and signal circuits were connected using the common chassis ground. After suffering several issues with noise and interference, it was decided to isolate the signal circuits from high amperage circuits by using separate grounds. This was accomplished by using a separate power source and ground for the microcontroller and control circuitry which completely isolated the low voltage systems now from the high capacity lead-acid batteries. The issue of noisy and randomized control circuit outputs was mostly solved by separating the grounds. It was later discovered that because the motor controllers can accept multiple inputs for a "type 2 throttle," they had a feature to feed voltage on the line that we had connected to "sensor ground" between the motor controller and the control circuitry. This back-feed of voltage into our system was causing throttle issues when the control circuitry was either turned off completely, or turned on after the motor controllers were powered up. This was solved by turning the control power on before engaging the motor controller power. This ensured that a "sensor input" was provided to the motor controllers on startup to prevent confusion as to the intended input for "type 2 throttle."

4.3.5 Arduino

Of the many microprocessors considered for this project, the Arduino Dué was chosen as the best fit. Arduino was chosen as the brand of microcontroller because of its high level of support in the engineering and hobbyist

community and also due to the availability of pre-developed code and ease of obtaining the hardware. The Dué board specifically was chosen over other Arduino products based on the extensive research and comparisons made between similar products.

There were two major advantages that the Dué has over many of the other Arduino microprocessors. First, the clock speed is set at 83 MHz which provided one of the highest processing speeds of the initially proposed microprocessors. Secondly, the Dué contains two integrated digital-to-analog output pins, which eliminated the need for an extra component to make the transition between the digital signals of the microprocessor and the analog input that motor controllers require.

There are also two significant disadvantages to using the Dué that had to be overcome. The first is that this particular board was still in the "beta" phase. This meant that Arduino had yet to work out all of the bugs in this current model and its IDE. The second caveat was that the Dué's maximum allowable I/O pin voltage is 3.3 Volts. This I/O restriction was problematic because many of the commercially available sensors operate with 5 Volt I/O, which could potentially damage the processor on the Dué. In order to accommodate this problem, the team put extra care into choosing as many configurations as possible that were easily adapted to 3.3 Volts. The throttle and steering sensors were simple potentiometers that could be fed by the Arduino and thus restricting them to 3.3 Volts maximum. The encoders posed a challenge, however. To adapt the encoders to the Dué's limitations, the team hooked the encoder input line to the 5 Volt output pin on the Arduino Dué and then employed two-channel voltage level converters to step down the output voltage of the encoder sensors to 3.3 Volts.

Several more problems presented themselves during testing. One Arduino failed during initial bench testing of some op-amp circuitry designed to modify the output range of the digital-to-analog outputs. Although an exact cause was never explicitly proven for the failure, it was presumed that a high voltage was encountered while testing the output from the op-amp circuit. The circuit was later reconfigured to prevent high voltage from being applied directly to the I/O pins of the Dué. In order to prevent damage to other components, a system was then implemented to ensure that circuitry and code were double checked by peers before any serious and potentially dangerous testing was performed.

Another issue was encountered during turn radius testing. A spiking voltage response was received at the Curtis controllers without any user input. It was determined that this spiking was caused by noise in the system and from varying resistances in the physical potentiometers which fed a mapping function in the throttle-response code. Due to these causes, the throttle input was falling below zero and negative values were assumed by the microcontroller to represent 100% throttle.

Lastly, it was assumed that because of the placement of the Arduino and its housing, that the high current wires running under it could have been causing electromagnetic interference with the sensors and microcontroller. Originally, the housing for the Arduino Due was milled out of a piece of wood and enclosed with a clear acrylic top so that the inside components were visible. During testing of the accelerator and steering inputs, it was determined that excessive noise from the potential electromagnetic fields were likely causing the Arduino to send errant outputs to the motor controllers. Shielded cat-5 cable was utilized to protect the inputs to the Arduino from interference. In addition, a new box was fabricated out of aluminum to shield the Arduino, 8 Volt-voltage regulator, and op-amp circuitry from other possible electromagnetic fields. The box was reattached on top of the battery box next to the control panel for ease of access and to keep the majority of the wires running along the right side of the vehicle. For an additional level of protection, the ground node for the Arduino circuit was separated from the ground node of the high-current circuit, as stated before.

When finally implemented, the Arduino Due was powered by a 11.3 Volt LiPo battery that was run through an 8 Volt regulator. The 8 Volt regulator supplied the Arduino directly with 7.88 Volts at less than 800 milliAmps. The Ar-

duino was wired to accept two analog inputs, one for each potentiometer. It was wired to also accept six encoder signals, three per encoder and per encoder, two for quadrature rotational pulses and one for a single absolute pulse per revolution. The outputs, DAC pins 0 and 1 were fed to the op-amp circuit, and the 3.3 Volt and 5 Volt pins were wired to busses to provide power to the sensors. Other devices were also connected to the Arduino including two LEDs, and SD card breakout board and a pull-down switch . A table of pin connections can be seen below.

TABLE 1: Arduino Pin Connections

Pin	Use
A10	Steering Potentiometer
A11	Throttle Potentiometer
DAC0	Throttle output to op-amp for left motor
DAC1	Throttle output to op-amp for right motor
GND	To ground bus for various applications
Vin	Power from the voltage regulator
3.3V	To 3.3V power bus to power sensors and the SD card
5V	To 5V power bus to power encoders
8	LED for general purpose use
9	Pulldown Switch
10	SD Card signal
11	LED for power indication
23	Encoder 1 Pin A
25	Encoder 0 Pin A
27	Encoder 1 Pin B
29	Encoder 0 Pin B
31	Encoder 1 Pin Z
33	Encoder 0 Pin Z

A schematic of the Arduino Dués pin layout can be seen in Appendix C.2.

4.3.6 Op Amp Circuitry

During preliminary research into the Curtis motor controllers, it was noted that the required input for a "type 2" sensor throttle did not match the stock output for the Arduino Dué. While testing for throttle response, it was noted that the digital-to-analog output voltage range from the Arduino Dué was between 0.55 Volts and 2.75 Volts, not the 0 Volts to 3.3 Volts that the team had anticipated. This voltage range needed modification before communicating with the Curtis motor controllers. The voltage range recommended for the motor controllers to perceive a usable input was discovered to be between 0 Volts to 5 Volts, with a possible deadband of nearly 0.2 Volts and max input of as little as 3.5 Volts based on the programmed parameters. This meant that without adjusting the output from the Arduino, the motor controllers would receive throttle input with no actual input from the throttle sensor. This was due to the minimum output of the Dué being higher than the minimum input allowed for the controllers. In addition, expanding the upper value of the DAC pins gave better resolution to more appropriately operate the vehicle and provide greater control of the wheel speeds given sensor inputs. To remedy this communication error and regulate the voltage range for maximum resolution, a circuit using an op-amp was designed and implemented.

The following equations and diagrams were obtained from the NCSU ECE-455 class notes and were used to calculate the new resistance values discussed above:

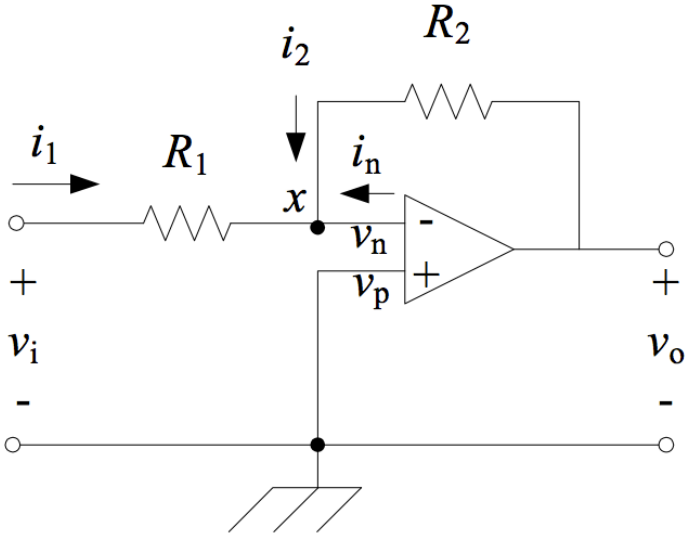


Fig. 10: Inverting Amplifier

$$\frac{v_o}{v_i} = -\frac{R_2}{R_1} \tag{1}$$

$$V_o = -(\frac{R_2}{R_A}V_A + \frac{R_2}{R_B}V_B + \frac{R_2}{R_C}V_C) \tag{2}$$

$$v_L = \frac{R_2}{R_1 + R + 2}v_s \tag{3}$$

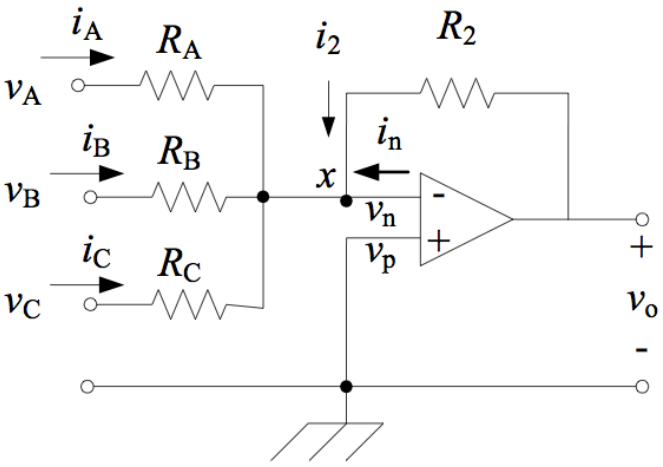


Fig. 11: Summing Amplifier

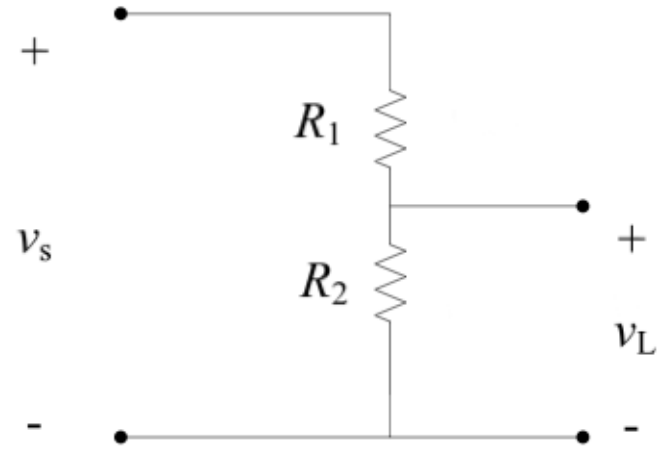


Fig. 12: Voltage Divider

The op-amp circuit shown in Figure 13 utilizes two sub-circuits in series to attain the appropriate expansion of the range while also offsetting the minimum voltage outputted. Shown on the left of the diagram is an inverting amplifier circuit (Figure 10) utilizing resistors $R1$ and $R2$. Its purpose is to amplify the maximum output voltage range and, based on this particular configuration, change its polarity. Shown on the right of the diagram is a summing amplifier circuit (Figure 11), utilizing resistors $R1$ and $R3$. Its purpose is to subtract a constant voltage from the previously expanded range in order to decrease the minimum output down to the required low end of the range. The voltage being subtracted was obtained from the voltage divider circuit utilizing resistors $R3$ and $R4$. Note that the source value for the voltage divider is negative, this was done because the summing amplifier circuit can only add voltages, thus it was necessary to obtain and add a negative value to simulate subtraction. Together, the pieces of this circuit create a wider voltage range with a closer-to-desired minimum and maximum output voltages.

The original values for this were calculated to be:

$$R1 = 510\Omega \quad R2 = 1.15k\Omega$$
$$R3 = 2.67k\Omega \quad R4 = 8.2k\Omega$$

$$V_{cc} = 5V$$

The 5 volt input was supplied from the Arduino's 5 Volt output pin. V_{sig} was the Digital to Analog output from the Arduino. After testing this circuit however, it was found that this particular setup was passing too much current and not providing adequate amplification of range. To address this, the resistance values were re-calculated to keep roughly the same ratios, but to include higher values in order to limit current more. The calculations, the slight changes to circuit layout, and the updated values are addressed below.

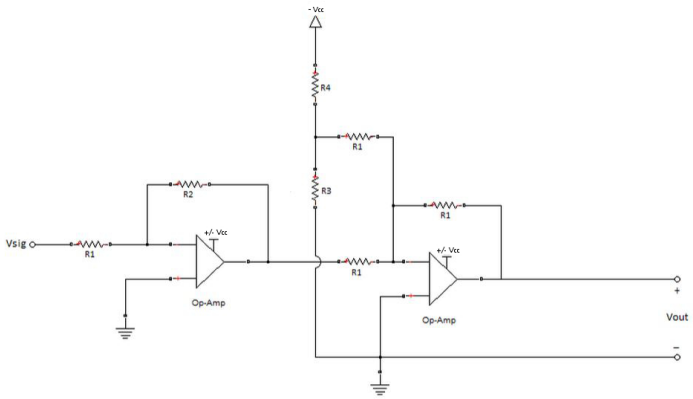


Fig. 13: Op-amp Circuit

Equations (1), (2), and (3) from above were manipulated to arrive at the ending values for the resistors. These equations were combined and simplified to obtain the following:

$$V_{out} = -\frac{R_2}{R_1}\left(-\left(\frac{R_1}{R_1}\left(-V_{cc}\left(\frac{R_3}{R_3+R_4}\right)\right)+\frac{R_1}{R_1}\left(V_{in}\right)\right)\right) \tag{4a}$$

$$V_{out} = \frac{R_2}{R_1}\left(\left(-V_{cc}\left(\frac{R_3}{R_3+R_4}\right)+V_{in}\right)\right) \tag{4b}$$

$$0V = \frac{R_2}{R_1}\left(-8V\left(\frac{R_3}{R_3+R_1}\right)+0.55V\right) \tag{4c}$$

$$0V = -8V\left(\frac{R_3}{R_3+R_1}\right)+0.55V \tag{4d}$$

$$5V = \frac{R_2}{R_1}\left(-8V\left(\frac{R_3}{R_3+R_1}\right)+2.75V\right) \tag{4e}$$

$$V_{out} = \frac{22k\Omega}{10k\Omega}\left(-V_{cc}\left(\frac{820\Omega}{820\Omega+10k\Omega}\right)+V_{in}\right) \tag{4f}$$

$$V_{out} = 2.2\left(-0.0759V_{cc}+V_{in}\right) \tag{4g}$$

$$V_{out} = 2.2\left(V_{in}\right)-1.334V \tag{4h}$$

Equation 4a represents the entire circuit performance from input to output. To solve for the variables and the resistances, the equation was simplified again. The simplification of equation 4a yielded equation 4b.

This equation was then given a few boundary conditions including the lower range bounds of the input and output, the upper bounds of input and output, and the projected V_{cc} voltage to be applied to the circuit. Previously, the V_{cc} value was chosen to be 5V and resistances were found in a similar manner, however, the values chosen previously did not yield optimal results. Thus a new V_{cc} value of 8V was chosen here to power the op-amps, feed the voltage divider, and power the Arduino. This 8 Volt source was used as a condition in solving equation 4b for the resistances. Lastly, to eliminate a variable, the values of $R4$ and $R1$ were chosen to be the same. The system of equations below was then solved for $R1$, $R2$, and $R3$.

This equation used the lower output of the Arduino and the desired low output of the circuit along with the chosen V_{cc} value to get equation 1. Because the left side of the equation was zero, the gain obtained by $R2/R1$ could be divided over to the left side and gotten rid of. This left what can be seen in equation 4c.

This equation could then be solved if a semi-arbitrary value for $R1$ was chosen. $R1$ was chosen to be $10k\Omega$ because of two reasons. Firstly, a $10k\Omega$ resistor is extremely common and secondly, $10k\Omega$ puts the amperage flowing through the circuit in the mA range. If all values revolved around a $10k\Omega$ base resistance, then the circuit was almost assured to have acceptable amperage ratings so as to not damage any components in the Arduino or sensors. After choosing a value for $R1$, equation 4d was solved for $R3$ to produce a value of 820Ω. This completed the voltage divider circuit. After $R3$ had been solved for, the values for $R1$, $R3$ and the boundary conditions were placed back into equation 4c and $R2$ was determined to be $22k\Omega$. This gave a amplifying gain of 2.2. Finally, the upper bounds were placed into equation 4b and the resistance values were verified to work for the upper bound as well. The the derivation of the final equation can be seen below in sequence.

After verifying that the resistance values worked theoretically, the theoretical outputs of the system were found to be -0.124 Volts on the low end and 4.716 on the upper end using equation 4h. These were acceptable theoretical values knowing that actual components would have variation. From here, an op-amp chip and resistors were acquired and placed on a breadboard for testing. Testing of the actual circuit proved to have a range between 0.13 Volts on the low end and 5.01 Volts on the high end. This difference in theoretical to actual values was attributed to the variance in components' real values (within their respective tolerance ranges). Once fully tested, the circuit was transferred to a prototyping board and soldered together.

The physical circuit was implemented with two small IC chips. A LM324 quad op-amp IC chip was chosen to create the summing circuit and the inverting circuit. Two op-amps on each end of the chip were wired to configure the right and left amplification channels. The two channels worked in unison to achieve the desired result of dual outputs from the control circuitry to the motor controllers, one output to each controller. This particular chip was chosen because it was readily available and was suitable for the requirements. The circuit diagram for the LM324 chip is shown in Figure 14.

The circuit is implemented with two smaller circuits. Using a LM324 quad op-amp IC chip to create the summing circuit and the inverting circuit; the two worked in unison to achieve the desired result. This particular chip was chosen because it was readily available and was suitable for the requirements. The circuit diagram for the LM324 chip is shown in Figure 14.

One issue that was encountered in addition to the initial resistance values not providing appropriate outputs was the issue of obtaining the negative Vcc value. It was discovered in the initial batch of testing that the low voltage pin of the op amp chip needed an inverted voltage equivalent to that of the power supply as opposed to ground as was originally planned. After some rudimentary research, an IC chip LMC7660 DC/DC voltage converter was found and implemented to correct this issue. It was chosen for its ability to take a positive voltage input and invert it with minimal error. The circuit diagram utilizing the LMC7660 is shown in Figure 15.

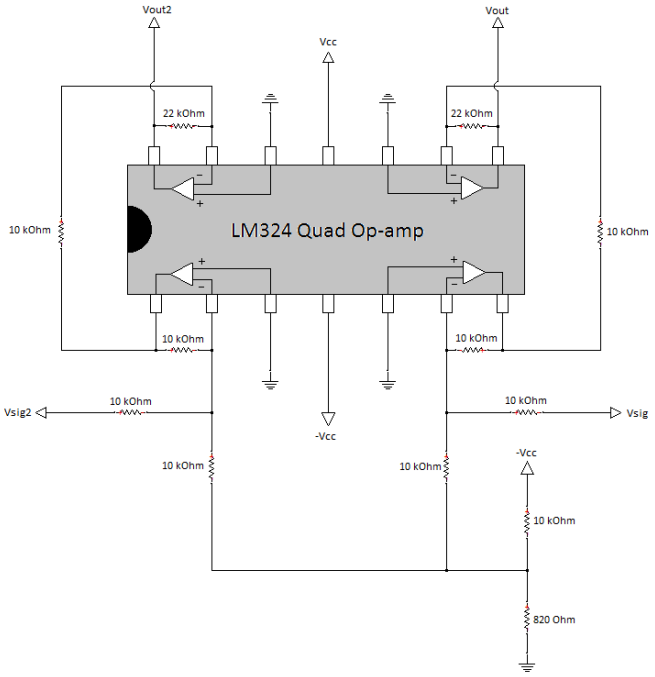


Fig. 14: Circuit diagram for the LM324

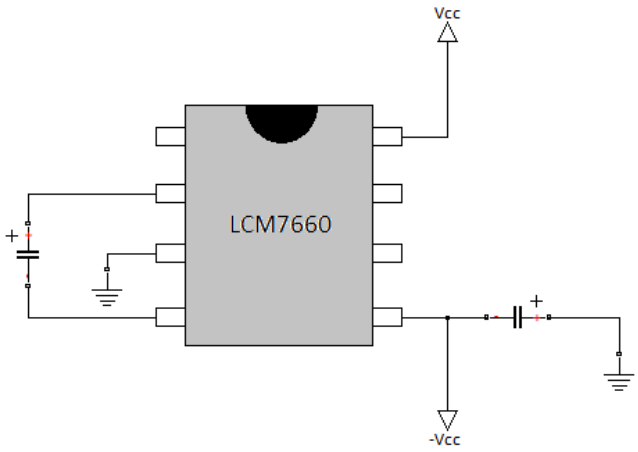


Fig. 15: Circuit diagram for the LM7660

4.4 Control

An electronic differential requires two separate motors being controlled by an algorithm that takes into account throttle position, turning radius, and wheel speed. The first control system flowchart was created October 2013 (Figure 21). It gave the team an overall direction to plan. First, the Arduino checks the angle reading from the steering potentiometer and determines whether the vehicle should be turning right, left, or going straight. The angular velocity of the wheels is compared to the turning radius to determine if one of the wheels has broken contact with the ground or is slipping.

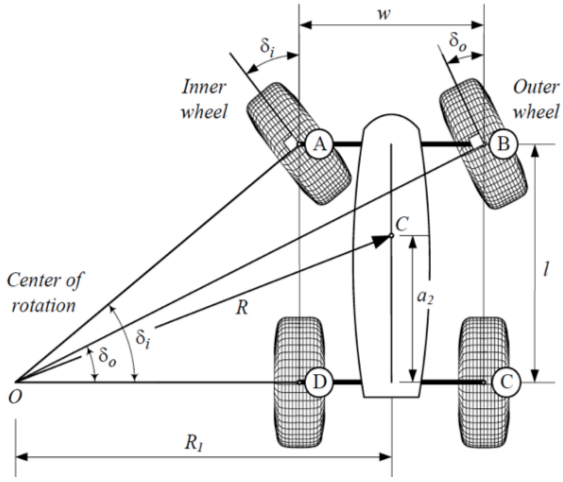


Fig. 16: Ackerman Steering Geometry

The kart's steering is in an Ackerman configuration (also referred to as Ackerman steering, Ackerman mechanism, or Ackerman geometry). Ackerman steering is a geometric arrangement of the linkages that accommodates the inner and outer wheels needing to travel

tangent to circles of different radii. Figure 16 is a diagram from page 387 of *Vehicle Dynamics - Theory and Application* that shows Ackerman steering.

This configuration gives us the following equations:

$$\cot(\delta_{out}) - \cot(\delta_{in}) = \frac{w}{l} \tag{5}$$

where w is the width between the centers of the front wheels, l is the length of the vehicle between the front and back axle, and δ_{in} and δ_{out} are the angles between the straight direction and the inner and outer wheel direction respectively.

$$R = \sqrt{a^2 + l^2 \cot \delta^2} \tag{6}$$

$$\cot \theta = \frac{\cot(\delta_{in}) + \cot(\delta_{out})}{2} \tag{7}$$

where R is the turning radius and a is the distance between the kart's center of gravity and the rear axle

4.5 Sensors

Four sensors were chosen to provide feedback to the microcontroller. First, a potentiometer was incorporated into the accelerator pedal assembly. The angle of the pedal is directly related to the rotation of the potentiometer shaft. The analog voltage output from the potentiometer is directly related to the rotation of the shaft and thus was used to create an electronic throttle system. The variable resistance changed the voltage which was then relayed to the controller and in turn, related throttle response to driver input.

A second potentiometer was also used for detecting the steering wheel angle. It was mounted at the end of the steering column and turned with the shaft in unison. Using the same principles mentioned above, this device was used to provide a voltage to the controller for steering and to keep track of the rotary position of the steering shaft.

The information concerning steering angle and throttle percent was a factor in how the controller determined which wheel required less throttle and how much overall throttle to apply. Both the steering and throttle potentiometers were 5k devices. One notable difference between the steering angle potentiometer and the accelerator potentiometer was that the steering potentiometer receives its voltage from the 3.3 Volt power out of the Arduino, rather than the 7.88 Volts provided by the voltage regulator. The reason for putting the steering potentiometer on the 3.3 Volt pin was to make sure the voltage didnt fluctuate as the batteries for the kart became depleted. The reason for powering the throttle potentiometer with the 7.88 Volts was to increase the maximum voltage out of the potentiometer. It was reduced due to the limitation in rotational capability caused by restricting the throttle pedal to about seventy degrees of rotation (to keep the pedal in a comfortable range of motion for the driver). Potentiometers with resistance values above $1k\Omega$ were chosen to prevent the Arduino from getting too much current into the analog pins. Resistance values below $15k\Omega$ were chosen in order to provide sufficient current to reduce the time needed to charge capacitors built into the Arduinos analog input circuitry.

Upon thoroughly researching optical encoders and affiliated mounting hardware, a pair of Koyo medium-duty optical encoders were purchased, along with mounting brackets and special couplings that minimize radial and axial loading to the encoders via their input shaft. These encoders were mounted to the drive shafts of each motor. They utilized 3 channels of digital signals: A, B, and Z. Channels A and B came offset by one-half of a pulse cycle and each channel pulsed 400 times per revolution. Channel Z changed value once per revolution, for a pulse $\frac{1}{400}$ revolution(s) in length. Having three channels allowed for exceptional variance in motor speed calculation. At low speeds, the quadrature pulses on channels A and B could be used to determine rotation direction and speed and at high speeds, where direction is less important, the Z channel could be counted to get the exact number of rotations over a sample of time.

The data sheets for these encoders can be seen in Appendix 34 and 35.

4.6 Software

4.6.1 Overview

To maintain consistency and keep everyone working on the same version of the code, the team formed a *Github* repository. The advantage of using a repository was that everyone would have access and the ability to edit current code, as well as revert back to older versions if necessary.

The code is divided into two main sections.

The first section is Control Algorithm code. It contains code necessary for driving the wheels and implementing the differential control. This includes feedback processing, steering and throttle angle sensing, and calculating the proper outputs to send to the Curtis motor controllers depending on the mode (differential or solid axle).

The second section consists of data acquisition code. It is used to to test hardware and software against their design specifications. The crux of this method is to use a Secure Digital Memory (SD) card accompanied by a breakout board. It is used with the corresponding SD card library to save data into comma separated value (CSV) files.

The code described in the project can be found via the weblink provided in the "references section".

4.6.2 Throttle Code

Due to the physical location of the accelerator pedal, the available range of the throttle potentiometer is about $60\,^{\circ}$. The Arduino uses the AnalogRead function to determine the input voltage associated with the throttle position. At the lowest position, the Arduino reads $.14$ Volts. At the highest position, the input voltage is $4.93V$. Data acquisition testing showed that the voltage input to the Dué increases almost linearly with respect to the accelerator pedal position. This relationship can be seen in Appendix B.3.

The range of the input voltage was programmed into the Dué and mapped to a 10-bit resolution, resulting in a value between 0 and 1023. If the kart is going straight or the operator selects the solid axle configuration, the same output is sent to each motor controller. In a similar way to the input mapping, the Arduino DAC output was calculated to fall between 0 and 1023.

4.6.3 Steering Code

The steering potentiometer works in a similar way to the accelerator pedal. It feeds a voltage to an analog input pin and is then mapped to a 10-bit integer, which is fused to calculate the turning radius that the operator desired. The steering potentiometer is attached to the steering column and displays about $50\,^{\circ}$ of range from hard left to hard right.

While turning, the vehicle travels two circles of different radii simultaneously and thus with no differential action, the outer tire will drag or the inner tire will break traction with the ground. This is due to the excessive force created by traveling a smaller radius circle at the same speed as the larger circle. Taking this into consideration, the steering code needed to be as accurate as possible to increase the effectiveness of the differential simulation. As a result, this greatly reduces the force acting on the drive wheels.

Furthermore, because of human error and a small amount of play in the column, the steering position associated with "straight" fluctuates by about $15\,^{\circ}$. Because the turn radius is rather large in this range and approaches infinity at perfectly straight, the difference in output between the two motors can be minimized inside of this region.

4.6.4 Encoder Code

When the motors are rotating under 500 revolutions per minute (RPM), the Arduino reads channels A and B. Channels A and B change from digital HIGH to digital LOW every time the shaft rotates $\frac{1}{400}$ of a revolution, or every $0.9\,^{\circ}$. After one channel changes logic value, the other channel changes and the position of the encoder is recorded. Based on the order in which this happens, the position goes up or down. Below zero indicates counter-clockwise and above zero indicates clockwise. After $\frac{1}{8}$ of a second, the angular velocity in RPM is calculated by the following equation:

$$\omega = \frac{position \cdot fractionsOfOneSecond \cdot}{400} \quad (8)$$

One of the flaws with this code is that it takes $\frac{1}{8}$ of a second to complete. This is necessary however, as the data wouldnt be accurate enough to be of any use if it were sped up. In order to speed up the process of computing angular velocity, a second method is used if the motors are rotating above 500 RPM. The Arduino ignores channels A and B and reads channel Z. The Arduino takes a timestamp and then checks channel Z to see if it is logical TRUE. Assuming it is, it then increments the value Z_{rev} for 10 total revolutions. Assuming Z has remained TRUE for 10 revolutions, the Arduino takes a second timestamp to determine $\Delta t = t_2 - t_1$. Angular velocity is then computed with the following equation:

$$\omega = \frac{pulseCounts}{time} \quad (9)$$

Encoder validation and calibration tests were performed using a stopwatch and a ramp-up function. The Curtis controllers were fed a gradual increase in voltage to approximately 13% of of the maximum. For one minute, the wheel rotations were counted manually and the calculated angular velocity was displayed on a monitor. The total number of visible rotations was multiplied by 5 (from the gear ratio) and compared to the calculated velocity. Velocities over 1000 RPM are difficult to count manually, and therefore it is hard to conclude that the computed angular velocity is the actual velocity.

4.6.5 Differential Code

The differential code is currently functional, however, it could still be improved upon. To read the inputs provided by the various sensors and drive the wheels in a similar manner. As of right now the Curtis controllers are set to Type 0 which institutes amp based Torque control. If it detects a difference in the amperage in one of the wheels, it will compensate, which can provide a very basic form of limited slip differential control.

In order to better refine this process and provide a platform for future design teams to continue the work the 2014 senior design team has started, an algorithm has been developed that will refine the differential control, and allow for future teams to implement various extra controls (traction control, electronic stability control, 4 wheel torque vectoring, etc.).

The algorithm that is currently being employed measures the angle provided by the steering potentiometer and verifies the rate of turn that the vehicle is trying to accomplish with the wheel encoders. Factoring out the noise by taking several samples and averaging, the code then adjusts the outputted signal, according to our control equations, and outputs a signal to allow for differential control of the motors.

4.6.6 Data Logging

As mentioned previously, a SD card with a breakout board was purchased to assist with testing and data acquisition. A mini-SD card fits into the slot on the breakout board. The breakout board connects to the Arduino via Serial Peripheral Interface (SPI) communication header pins. When in use, the data logging code requires the interaction of the tester via either the Arduino IDE or a pull-down switch. Typically, a tester uses the computer to enter data via the serial connection to the Arduino to specify what is being tested.

To test the steering position, the user enters the steering position angle in degrees. The Arduino then takes ten analog input samples and saves them in a CSV file. This process is repeated as many times as specified by the user. The user tells the Arduino to quit when they no longer want to run the test and the file is closed. For throttle position, additional data is acquired and saved to the CSV file. After the tester enters the position of the throttle pedal (in degrees), the DAC output is enabled allowing the tester to measure the voltage at the DAC output. Then the tester enters the voltage reading and it is saved, along with the 10 samples of analog input data.

The purpose of acquiring this data is threefold. The first reason was to find out how linear was the relationship between throttle position,

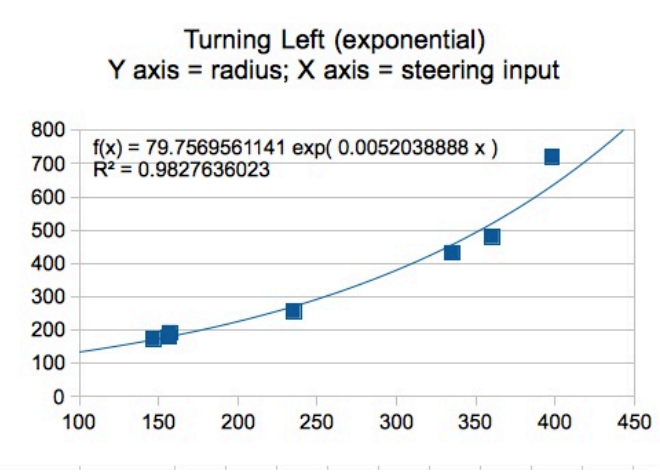


Fig. 17

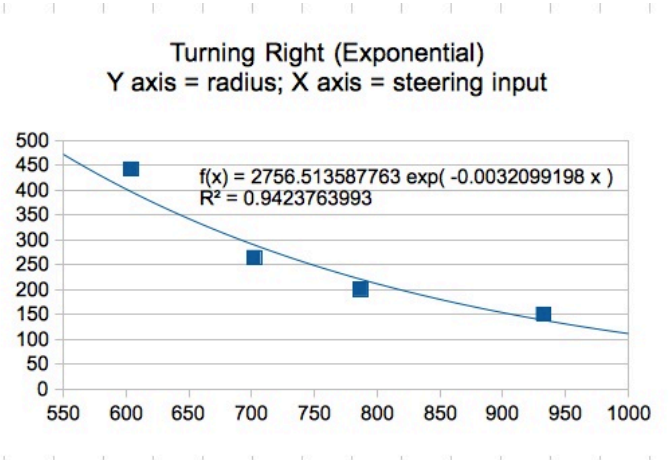


Fig. 18

analog input, and DAC output. Furthermore, the saved steering position and input data could be used to compute turn radius. Lastly, it was used to verify that the hardware was set up correctly by testing input/output before testing the system as a whole.

The data logging code was adapted to allow data acquisition of turn radius and analog steering input readings simultaneously. This process involves the user inputting the steering angle while people pushed the unpowered cart around a circle. The center of the the circled that the vehicle traveled around was marked with chalk. Then the radius of the circle was calculated and saved to the SD card along with 25 analog reads. Figure 17 and 18 show the steering input with respect to the turn radius. There are 6 samples of left turns and 4 samples of right turns. It is worth noting the exponential nature of the curves. As the vehicle has a slighter turn , the radius of curvature approaches infinity. This is evident in the increasingly upward trend as the analog input is in the middle of its range, from approximately 400 to 600. In this range, the vehicle will not require a differential input to each motor.

Programmable Parameters for Curtis Controllers

Parameter	M1	M2	M3	M4	ALL Modes
Battery Voltage	-	-	-	-	3
Drive Current Limit	280	325	325	375	-
Braking Current Limit	200	200	250	275	-
Throttle Braking Parameters	80	50	50	50	-
Acceleration Rate	2.5	2	2	1	-
Deceleration Rate	-	-	-	-	2.5
Braking Rate	2	2	2	2.5	-
Quick Start	-	-	-	-	2
Taper Rate	-	-	-	-	20
Maximum Speed	60	75	80	100	-
Creep Speed	0	0	0	0	-
Regen Speed	-	-	-	-	40
Control Mode	-	-	-	-	0
Throttle Type	-	-	-	-	2
Throttle Deadband	-	-	-	-	8
Throttle Maximum	-	-	-	-	98
Throttle Map	50	60	60	80	-
Min Field Current Limit	-	-	-	-	6
Max Field Current Limit	-	-	-	-	11
Field Map Start	-	-	-	-	5
Field Map	-	-	-	-	50
Current Ratio	-	-	-	-	2
Restraint	-	-	-	-	3
High Pedal Disable	-	-	-	-	Type 0
Static Return to Off	-	-	-	-	Type 0
Sequencing Delay	-	-	-	-	0
Main Contactor Driver Interlock	-	-	-	-	On
Main Contactor Dropout Delay	-	-	-	-	0
Main Contactor Weld Check	-	-	-	-	On
Main Coil Open Check	-	-	-	-	On
Auxiliary Enable	-	-	-	-	Off
EM Brake	-	-	-	-	Off
Auxiliary Driver Dropout Delay	-	-	-	-	0
Auxiliary Coil Open Check	-	-	-	-	Off
Electromagnetic Brake Delay	-	-	-	-	0

4.6.7 Curtis Parameters

4.7 Debugging & Troubleshooting

The first choice for measuring wheel speed was *VEX* encoders. They were picked because they were relatively inexpensive and readily available. These encoders, however, failed during testing because of their non-robust, plastic construction and lack of shaft bearings. Once the motors spooled up to over 1000 RPM, the plastic on plastic rotating parts melted due to friction and fused together, thus completely ruining the encoders. To resolve this issue,

Electromagnetic Brake Open Check	-	-	-	-	Off
Reverse Signal Open Check	-	-	-	-	Off
Contactor Pull-In Voltage	-	-	-	-	100
Contactor Holding Voltage	-	-	-	-	70
Emergency Reverse Enable	-	-	-	-	Off
Emergency Reverse Current Limit	-	-	-	-	200
Emergency Reverse Check	-	-	-	-	Off
Anti-Tiedown	-	-	-	-	Off
Fault Code	-	-	-	-	On
Pedal Interlock	-	-	-	-	Off
Precharge	-	-	-	-	On
Node Address	-	-	-	-	N/A
Load Compensation	-	-	-	-	5

Fig. 19

new TRD-N(H) incremental encoders were installed that were rated to withstand both radial and axial forces of 50 Newtons, far surpassing the ratings of the *VEX* encoders. The new encoders were also rated to 3000 continuous rpm, whereas the VEX encoders did not have a posted rpm rating because of their purpose being mainly in small robotics running low torque, low speed motors.

During the testing of the kart it was discovered that the two motors were not performing similarly. It was noted during several tests that one motor was getting much warmer that the other and it was postulated that the warmer motor was having to work harder to compensate for some fault of the other motor. When the underpowered motor was disassembled, it was discovered that windings inside the motor had been rubbing quite a bit, to the point that one of the wires in the field winding had been broken. Portions of the armature windings were bent in a manner that they were outside of their specified circumference and came into contact with the field windings, thus scraping, and eventually, severing them. To fix this, a small section of wire was pulled out of the coil and re-routed so it could be soldered back together. The armature pieces were also repositioned to their proper positions, and all of the exposed wire was recoated with an epoxy covering.

Another issue that arose during the karts debut drive when the brakes were quick to lock up and difficult to manage. The cables were adjusted in the bracket for increased travel and a spring was added behind the pedal for improved feel.

5 Testing

5.1 Methods

To quantify performance, budget-feasible tests were conducted in accordance with scaled versions of industry standard vehicular performance tests. These tests included a simple turn radius test, a skid-pad test and a slalom test which are all are dependent on wheel and axle performance. These are also standard performance tests for auto manufacturers and journalists. The measure of success for these tests are shortest diameter, speed/g's, and speed/time respectively. Out of the different tests considered, this selection displayed the greatest potential to expose the most significant differences between solid axle and controlled differential vehicle dynamics.

5.2 Objectives

The vehicle must be tested in appropriate, documented, replicable operating conditions. The system must show measurable improvement over a solid axle system and an forced couple (an independently driven split axle without differential control) system in the skidpad and slalom tests.

6 Results

When testing the kart in the solid axle configuration, it was concluded that testing could only be conducted while traveling straight. As expected, the tires skipped if the kart was turned. Testing was halted after 20 feet because the strain to the drivetrain was very large and would have resulted in damage.

Initial results have turning radii of the control system as equal to the tests of an uncoupled system. While the results versus the solid axle system are and will remain undetermined, the control system is superior simply based on hardware considerations.

7 Conclusion

The team solicited and received many donated parts in order to remain in the budget constraints. These donations helped make this project possible, however they also had certain

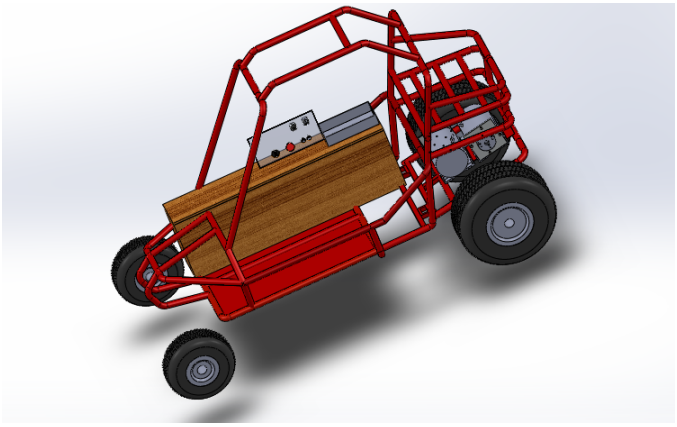


Fig. 20: Final SolidWorks Model

limitations and anomalies that wouldnt have been present in new and/or higher quality equipment. Much of the work and time was spent making many of these donated parts work as a single cohesive unit. Some of these issues were due to frame and axle damage that could never fully be repaired (e.g. bends in the tube structure and axles). These issues also required a reduction in the scope and goals of the project. Traction control and regenerative braking were goals that had to be omitted as the time grew short.

Proving the advantages over the solid axle was easier than anticipated. It was known that the physics of the situation would favor the split differential, however the severity of the karts reaction to the extra stress of the solid axle setup was unexpected.

APPENDIX A
DONATED PARTS, REMOTE KILL SWITCH, BUDGET, SCHEDULE,

A.1 Donated Parts

The Mechatronics Senior Design group was able to procure numerous parts for the project through the generosity of various businesses and individuals. Due to the budget constraints, these donations were necessary in order to stay within the allocated budget. The parts donated, as well as their donors are listed below.

Donated Parts

Donor	Items	Value
Chris Finlayson, Existential Motorcycles, Alexander, NC	Motorcycle brake cables (2)	$100 ($50 each)
Mountaintop Golf Cars	36V Advanced Concepts EZ-GO Motors (2)	$ 800($ 400 ea.)
	Battery connection cables	$ 15
	3 hours time	$ 45
EZ-GO (Textron)	Curtis Programmable Motor Controllers (3)	$ 1800 ($ 600 ea.)
	Curtis Handheld Programmer	$ 200
	Solenoids (2)	$ 50 ($ 25 ea.)
Joe and Ellen Reece	Go-Kart chassis	$ 400
ProMatic Automation	Emergency stop button	$ 50
	Key switch & Interlock switch	$ 75
	Dinrail and Other Material	$ 25
Skyland Automotive	12v golf cart battery	$ 215
	Total	$ 3775

A.2 Remote Kill Switch

Although not originally in the scope of the project, it became clear that a remote kill switch was necessary to safely operate the finished vehicle. It allows a bystander to disable the vehicle in an emergency situation if the driver of the kart is unable to do so. This process is implemented using an Arduino to communicate with the vehicle via a bluetooth connection. The microcontroller on board intercepts the transmitted signal and shuts down the output voltage to the speed controllers. The range of the bluetooth transmitter is one hundred meters.

Safety algorithms mainly concern the remote kill-switch. Other safety devices such as the dash-mounted E-stop switch and brake interlock switch do not require software to implement. The safety switch is a device that acts as a remote kill switch in case the operator of the vehicle is unable to properly shut down the vehicle in an emergency. This code happens first in the main loop and searches for a specific bluetooth signal. If the signal isn't available (meaning the kill switch is left open) the loop carries on to the next task. If the signal is received, the code immediately shuts down output voltage to the speed controllers.

A.3 Budget

Budget

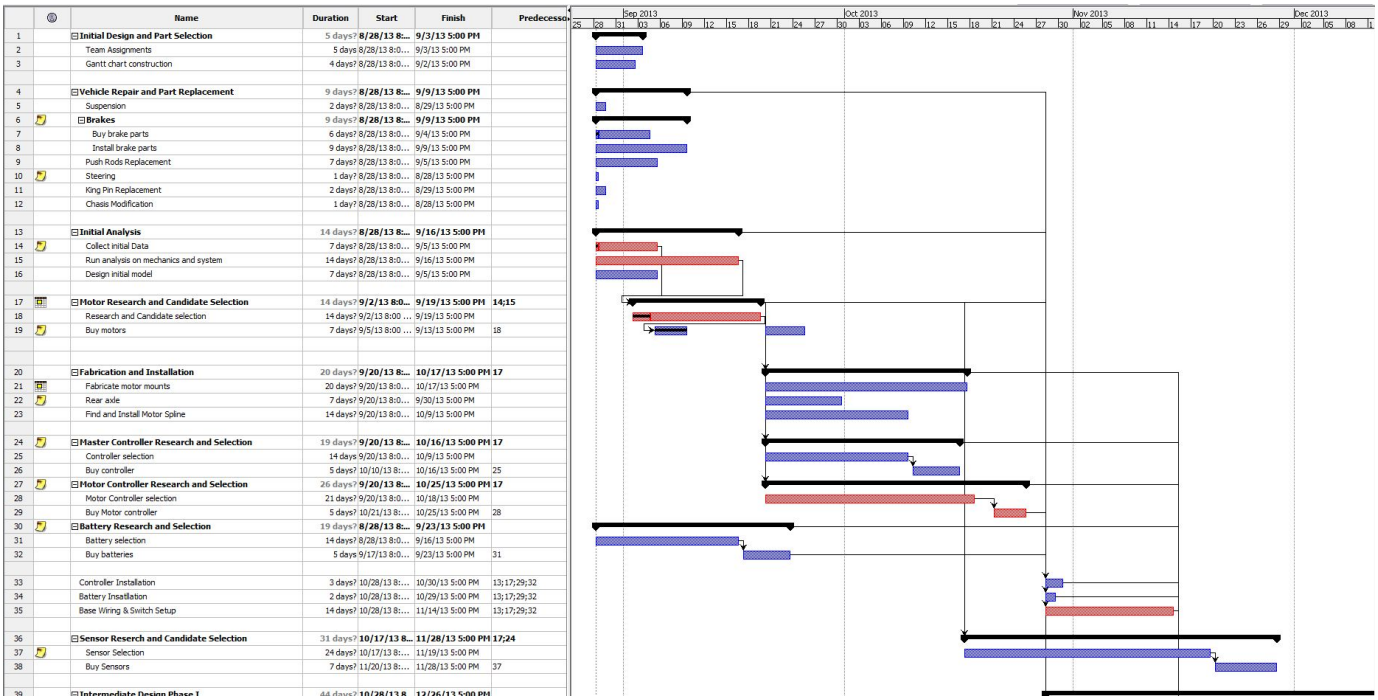
Vendor	Items	Cost
VEX Robotics	Vex motor encoders (4)	$ 52.98
BMI Karts	Tie rod, brake drum, brake band	$ 67.82
Mouser	Arduino Mega 2560	$ 52.11
Sparkfun	Bluetooth Mate Gold	$ 64.95
Interstate Batteries	Battery	$ 460.10
BMI Karts	Brake drum, brake band	$ 76.97
Tomtop	Mini digital LCD scale	$ 27.78
Nextwarehouse	Wireless bluetooth adapter	$ 20.48
Digikey	Arduino Dué	$ 59.40
MFG Supply	4 Hydraulic Shocks	$ 158.95
Golf Car Discounters	Input shaft kit	$ 304.00
Sparkfun	Breakout Board	$ 21.33
Mouser	Micro SD Card	$ 20.64
Tractor Supply	Bushings, coupler, pillow blocks	$ 21.84
Grainger	Solenoid (2)	$ 131.48
Grainger	400A fuses (2)	$ 26.71
Newark	Voltage converter	$ 13.93
Tractor Supply	Axle Sprocket	$ 59.66
Automation Direct	Encoders (2)	$ 270.00
Automation Direct	Coupling (2)	$ 55.00
Automation Direct	Mounting bracket (2)	$ 21.00
Amazon.com	Arduino Dué	$ 39.70
Sparkfun	Logic Level Converter(4)	$ 11.73
	Total	$ 2138.60

A.4 Schedule

To budget for the limited time, the team implemented a Gantt Chart to keep track of the progress of the project. The Gantt Chart provided an effective means to forecast potential bottlenecks in the design. This allowed the team to compensate and adjust for these problems in order to maintain the necessary pace to complete the project.

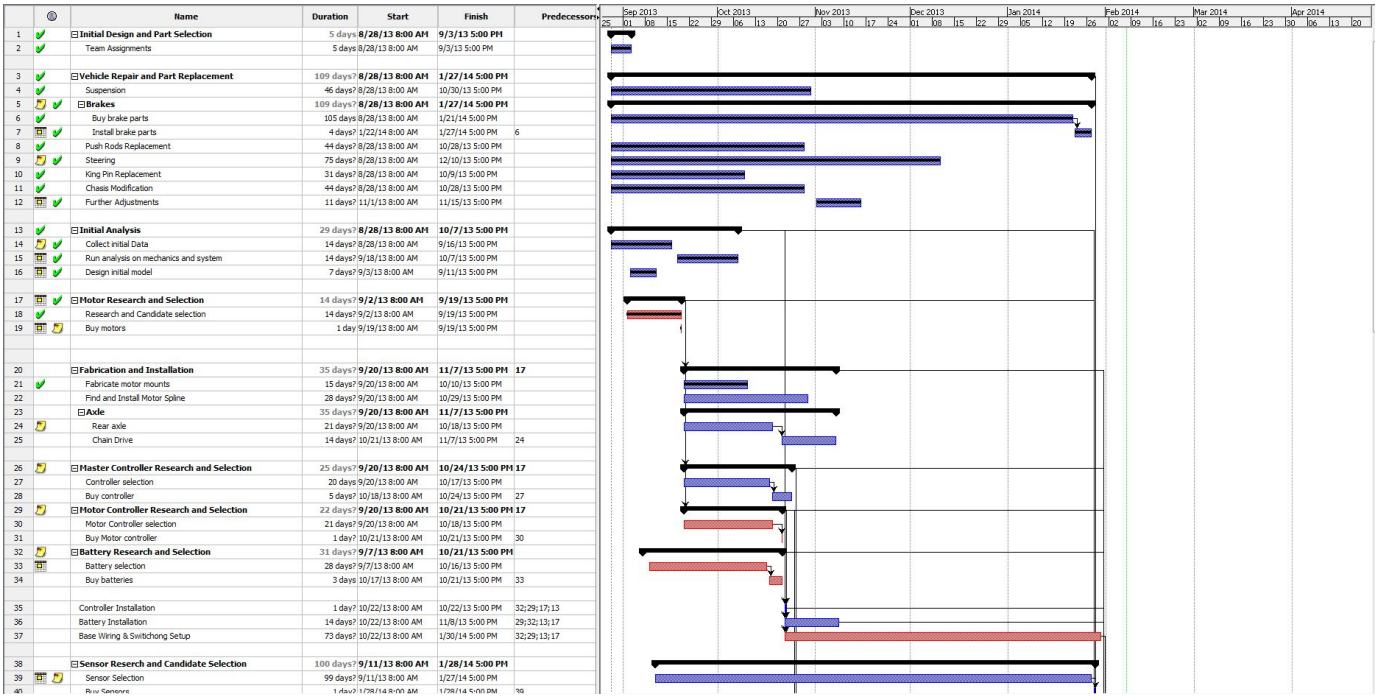
Below are two Gantt charts that illustrate our anticipated timeline and the projects actual timeline. One of the biggest deviation between the two was due to vehicle repair and part replacement. The section of our timeline was stretched because of part procurement for our

modified chassis (this includes significant vehicular weight and increased drive power) and part fitting (items needing to be fabricated or modified).



	Name	Duration	Start	Finish	Predecessors
1	Initial Design and Part Selection	5 days?	8/28/13 8:...	9/3/13 5:00 PM	
2	Team Assignments	5 days	8/28/13 8:0...	9/3/13 5:00 PM	
3	Gantt chart construction	4 days?	8/28/13 8:0...	9/2/13 5:00 PM	
4	Vehicle Repair and Part Replacement	9 days?	8/28/13 8:...	9/9/13 5:00 PM	
5	Suspension	2 days?	8/28/13 8:0...	8/29/13 5:00 PM	
6	Brakes	9 days?	8/28/13 8:...	9/9/13 5:00 PM	
7	Buy brake parts	6 days?	8/28/13 8:0...	9/4/13 5:00 PM	
8	Install brake parts	9 days?	8/28/13 8:0...	9/9/13 5:00 PM	
9	Push Rods Replacement	7 days?	8/28/13 8:0...	9/5/13 5:00 PM	
10	Steering	1 day?	8/28/13 8:0...	8/28/13 5:00 PM	
11	King Pin Replacement	2 days?	8/28/13 8:0...	8/29/13 5:00 PM	
12	Chasis Modification	1 day?	8/28/13 8:0...	8/28/13 5:00 PM	
13	Initial Analysis	14 days?	8/28/13 8:...	9/16/13 5:00 PM	
14	Collect initial Data	7 days?	8/28/13 8:0...	9/5/13 5:00 PM	
15	Run analysis on mechanics and system	14 days?	8/28/13 8:0...	9/16/13 5:00 PM	
16	Design initial model	7 days?	8/28/13 8:0...	9/5/13 5:00 PM	
17	Motor Research and Candidate Selection	14 days?	9/2/13 8:0...	9/19/13 5:00 PM	14;15
18	Research and Candidate selection	14 days?	9/2/13 8:00 ...	9/19/13 5:00 PM	
19	Buy motors	7 days?	9/5/13 8:00 ...	9/13/13 5:00 PM	18
20	Fabrication and Installation	20 days?	9/20/13 8:...	10/17/13 5:00 PM	17
21	Fabricate motor mounts	20 days?	9/20/13 8:0...	10/17/13 5:00 PM	
22	Rear axle	7 days?	9/20/13 8:0...	9/30/13 5:00 PM	
23	Find and Install Motor Spline	14 days?	9/20/13 8:0...	10/9/13 5:00 PM	
24	Master Controller Research and Selection	19 days?	9/20/13 8:...	10/16/13 5:00 PM	17
25	Controller selection	14 days	9/20/13 8:0...	10/9/13 5:00 PM	
26	Buy controller	5 days?	10/10/13 8:...	10/16/13 5:00 PM	25
27	Motor Controller Research and Selection	26 days?	9/20/13 8:...	10/25/13 5:00 PM	17
28	Motor Controller selection	21 days?	9/20/13 8:0...	10/18/13 5:00 PM	
29	Buy Motor controller	5 days?	10/21/13 8:...	10/25/13 5:00 PM	28
30	Battery Research and Selection	19 days?	8/28/13 8:...	9/23/13 5:00 PM	
31	Battery selection	14 days?	8/28/13 8:0...	9/16/13 5:00 PM	
32	Buy batteries	5 days	9/17/13 8:0...	9/23/13 5:00 PM	31
33	Controller Installation	3 days?	10/28/13 8:...	10/30/13 5:00 PM	13;17;29;32
34	Battery Insatllation	2 days?	10/28/13 8:...	10/29/13 5:00 PM	13;17;29;32
35	Base Wiring & Switch Setup	14 days?	10/28/13 8:...	11/14/13 5:00 PM	13;17;29;32
36	Sensor Reserch and Candidate Selection	31 days?	10/17/13 8...	11/28/13 5:00 PM	17;24
37	Sensor Selection	24 days?	10/17/13 8:...	11/19/13 5:00 PM	
38	Buy Sensors	7 days?	11/20/13 8:...	11/28/13 5:00 PM	37

Anticipated Schedule



	Name	Duration	Start	Finish	Predecessors
1	Initial Design and Part Selection	5 days	8/28/13 8:00 AM	9/3/13 5:00 PM	
2	Team Assignments	5 days	8/28/13 8:00 AM	9/3/13 5:00 PM	
3	Vehicle Repair and Part Replacement	109 days?	8/28/13 8:00 AM	1/27/14 5:00 PM	
4	Suspension	46 days?	8/28/13 8:00 AM	10/30/13 5:00 PM	
5	Brakes	109 days?	8/28/13 8:00 AM	1/27/14 5:00 PM	
6	Buy brake parts	105 days	8/28/13 8:00 AM	1/21/14 5:00 PM	
7	Install brake parts	4 days?	1/22/14 8:00 AM	1/27/14 5:00 PM	6
8	Push Rods Replacement	44 days?	8/28/13 8:00 AM	10/28/13 5:00 PM	
9	Steering	75 days?	8/28/13 8:00 AM	12/10/13 5:00 PM	
10	King Pin Replacement	31 days?	8/28/13 8:00 AM	10/9/13 5:00 PM	
11	Chasis Modification	44 days?	8/28/13 8:00 AM	10/28/13 5:00 PM	
12	Further Adjustments	11 days?	11/1/13 8:00 AM	11/15/13 5:00 PM	
13	Initial Analysis	29 days?	8/28/13 8:00 AM	10/7/13 5:00 PM	
14	Collect initial Data	14 days?	8/28/13 8:00 AM	9/16/13 5:00 PM	
15	Run analysis on mechanics and system	14 days?	9/18/13 8:00 AM	10/7/13 5:00 PM	
16	Design initial model	7 days?	9/3/13 8:00 AM	9/11/13 5:00 PM	
17	Motor Research and Selection	14 days?	9/2/13 8:00 AM	9/19/13 5:00 PM	
18	Research and Candidate selection	14 days?	9/2/13 8:00 AM	9/19/13 5:00 PM	
19	Buy motors	1 day	9/19/13 8:00 AM	9/19/13 5:00 PM	
20	Fabrication and Installation	35 days?	9/20/13 8:00 AM	11/7/13 5:00 PM	17
21	Fabricate motor mounts	15 days?	9/20/13 8:00 AM	10/10/13 5:00 PM	
22	Find and Install Motor Spline	28 days?	9/20/13 8:00 AM	10/29/13 5:00 PM	
23	Axle	35 days?	9/20/13 8:00 AM	11/7/13 5:00 PM	
24	Rear axle	21 days?	9/20/13 8:00 AM	10/18/13 5:00 PM	
25	Chain Drive	14 days?	10/21/13 8:00 AM	11/7/13 5:00 PM	24
26	Master Controller Research and Selection	25 days?	9/20/13 8:00 AM	10/24/13 5:00 PM	17
27	Controller selection	20 days	9/20/13 8:00 AM	10/17/13 5:00 PM	
28	Buy controller	5 days?	10/18/13 8:00 AM	10/24/13 5:00 PM	27
29	Motor Controller Research and Selection	22 days?	9/20/13 8:00 AM	10/21/13 5:00 PM	17
30	Motor Controller selection	21 days?	9/20/13 8:00 AM	10/18/13 5:00 PM	
31	Buy Motor controller	1 day?	10/21/13 8:00 AM	10/21/13 5:00 PM	30
32	Battery Research and Selection	31 days?	9/7/13 8:00 AM	10/21/13 5:00 PM	
33	Battery selection	28 days?	9/7/13 8:00 AM	10/16/13 5:00 PM	
34	Buy batteries	3 days	10/17/13 8:00 AM	10/21/13 5:00 PM	33
35	Controller Installation	1 day?	10/22/13 8:00 AM	10/22/13 5:00 PM	32;29;17;13
36	Battery Installation	14 days?	10/22/13 8:00 AM	11/8/13 5:00 PM	29;32;13;17
37	Base Wiring & Switchong Setup	73 days?	10/22/13 8:00 AM	1/30/14 5:00 PM	32;29;13;17
38	Sensor Reserch and Candidate Selection	100 days?	9/11/13 8:00 AM	1/28/14 5:00 PM	
39	Sensor Selection	99 days?	9/11/13 8:00 AM	1/27/14 5:00 PM	

Actual Schedule

A.5 Traction Control

When designing the traction control, the flow chart in Figure 21 helped in overall direction in the planning phase. In the diamond decision blocks above, ω represents wheel speed and θ represents steering angle. Through the combination of tests points, the control scheme falls to one of 5 cases. Once the "state" of the vehicle has been determined, the controller will calculate the necessary adjustments in power to return the kart to a non-slipping state. As the project moved forward, this control scheme became obsolete.

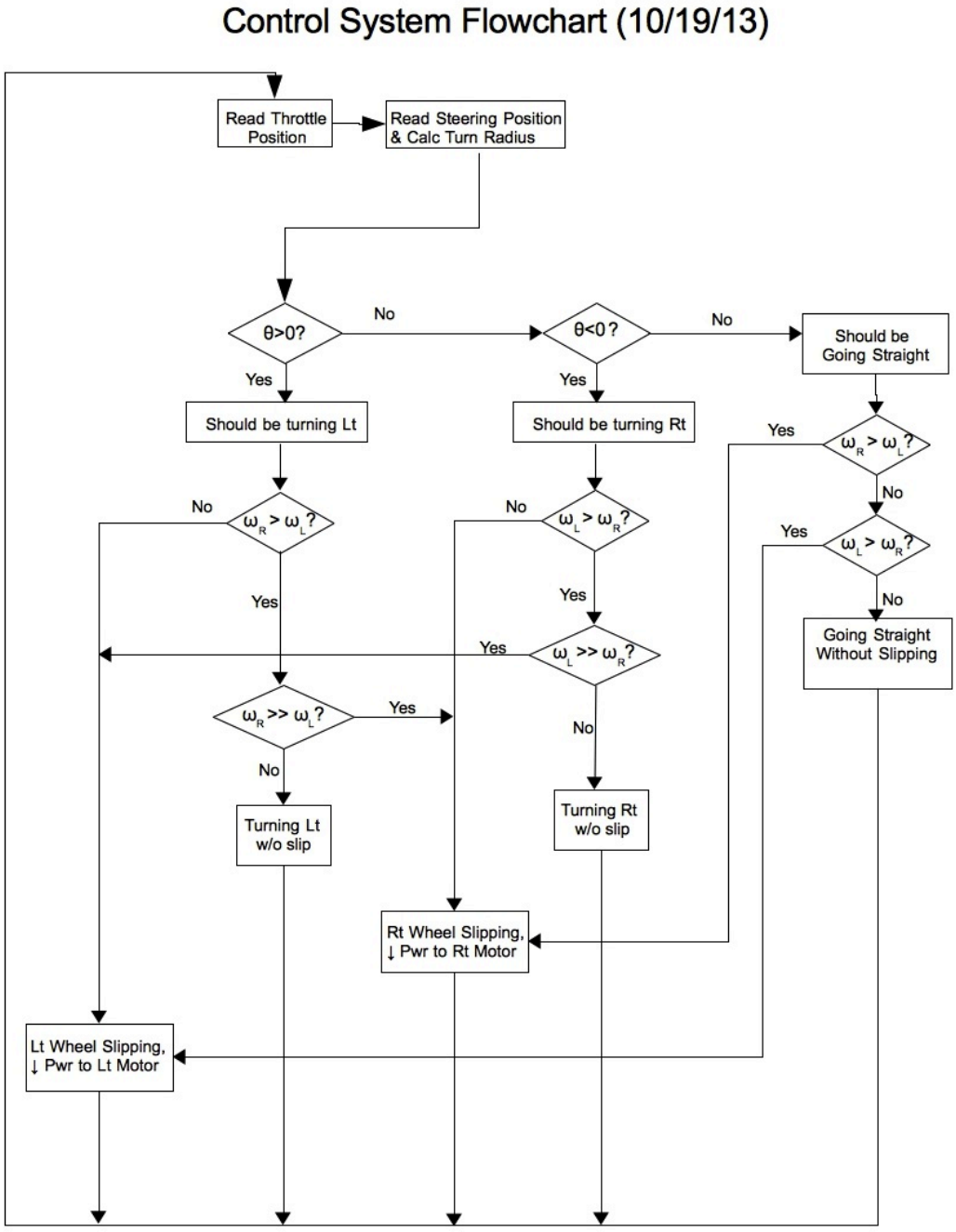
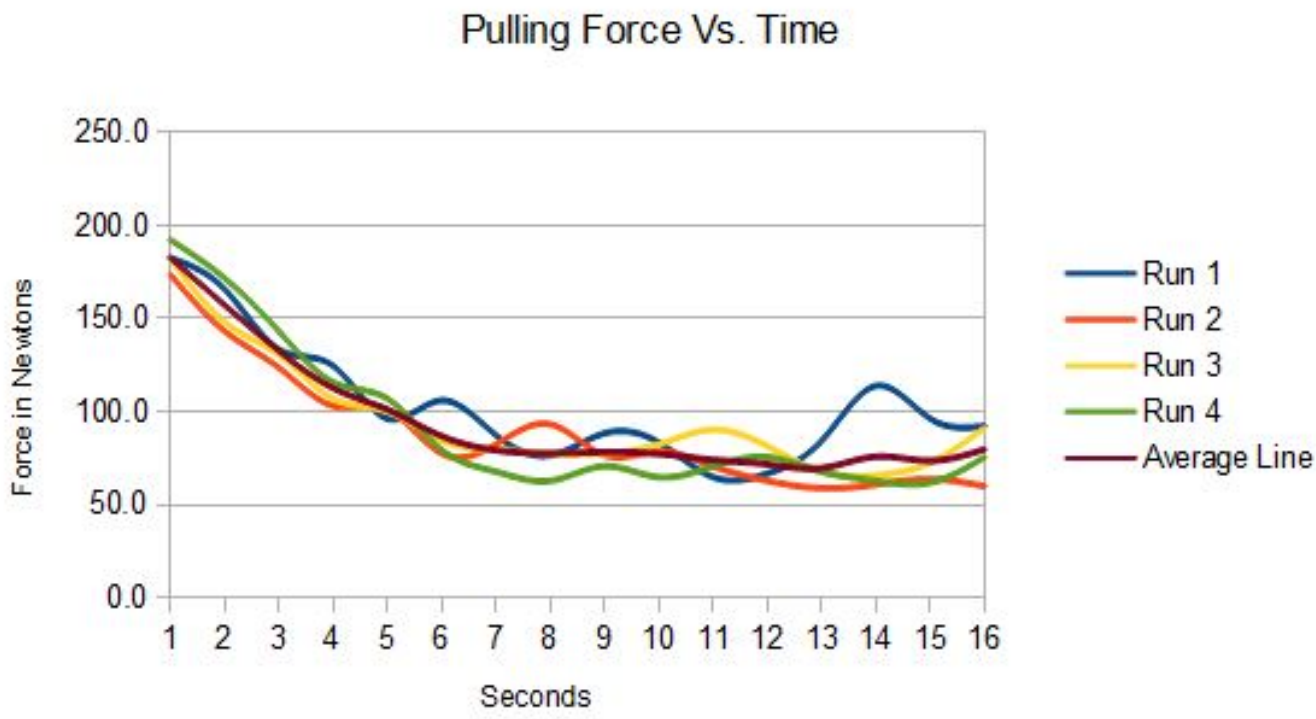


Fig. 21: Control Flow Chart

APPENDIX B
DATA TABLES AND GRAPHS

This test reflects the rolling resistance of the kart before any actual components were added. Three people were riding the cart to simulate the additional weight of said components. The distance was 40 feet, enough to reach a steady state.

B.1 Initial Horse Power Calculation

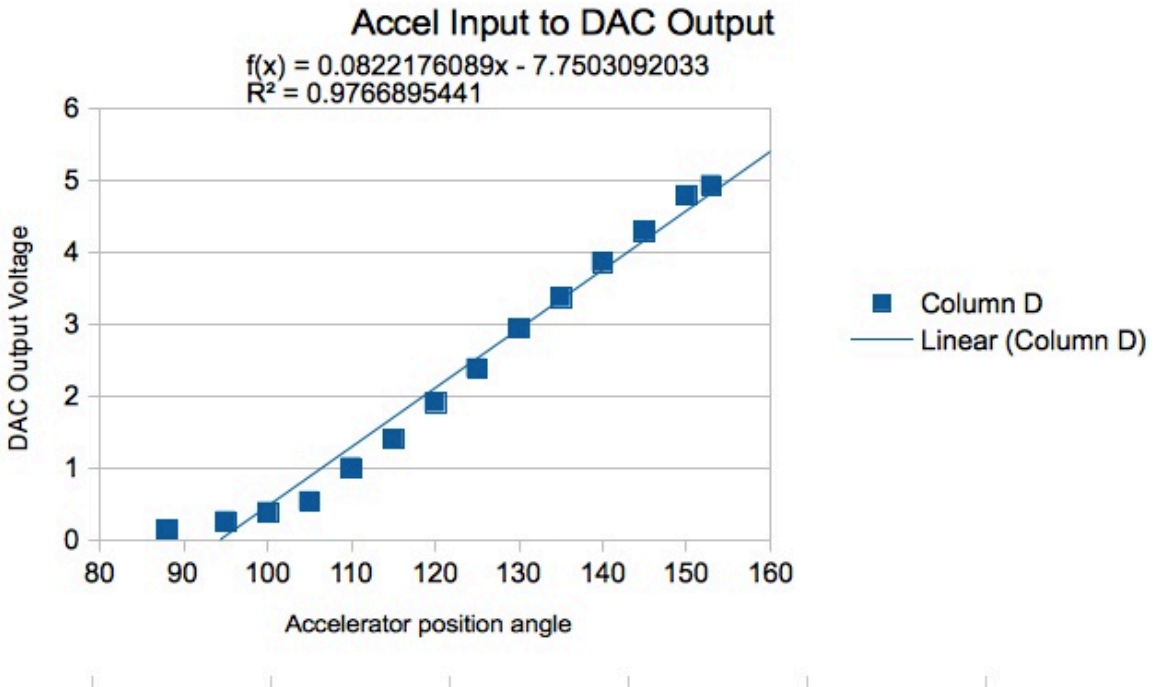


B.2 Power Requirements for Rolling Resistance

Speed (in MPH)	Speed (m/s)	0% Grade Total Drag	10% Grade Total Drag10	0% Grade Wattage Requirement	10% Grade Wattage Requirement	0% Grade HP Requirement	10% Grade HP Requirement10
1	0.447	76.77	253.11	34.32	113.14	0.05	0.15
2	0.894	76.97	253.31	68.81	226.46	0.09	0.30
3	1.341	77.31	253.65	103.67	340.15	0.14	0.46
4	1.788	77.79	254.13	139.08	454.38	0.19	0.61
5	2.235	78.40	254.74	175.22	569.34	0.23	0.76
6	2.682	79.14	255.49	212.26	685.21	0.28	0.92
7	3.129	80.03	256.37	250.40	802.18	0.34	1.07
8	3.576	81.05	257.39	289.82	920.42	0.39	1.23
9	4.023	82.20	258.54	330.69	1040.11	0.44	1.39
10	4.47	83.49	259.83	373.20	1161.44	0.50	1.55
11	4.917	84.92	261.26	417.53	1284.60	0.56	1.72
12	5.364	86.48	262.82	463.86	1409.76	0.62	1.89
13	5.811	88.17	264.52	512.38	1537.10	0.69	2.06
14	6.258	90.01	266.35	563.26	1666.81	0.75	2.23
15	6.705	91.98	268.32	616.70	1799.07	0.83	2.41
16	7.152	94.08	270.42	672.86	1934.06	0.90	2.59
17	7.599	96.32	272.66	731.94	2071.96	0.98	2.77
18	8.046	98.70	275.04	794.12	2212.96	1.06	2.96
19	8.493	101.21	277.55	859.57	2357.24	1.15	3.16
20	8.94	103.86	280.20	928.48	2504.98	1.24	3.35

B.3 Throttle Response Graph

In order to determine if the accelerator input was linear and verify the output voltage going to the Curtis speed controllers, an SD card was purchased and code written to acquire data including accelerator position, input to the Arduino, and measured output voltage to the controllers. The information was saved to the SD card and converted to a spreadsheet. The graph below indicates the response is close to linear.

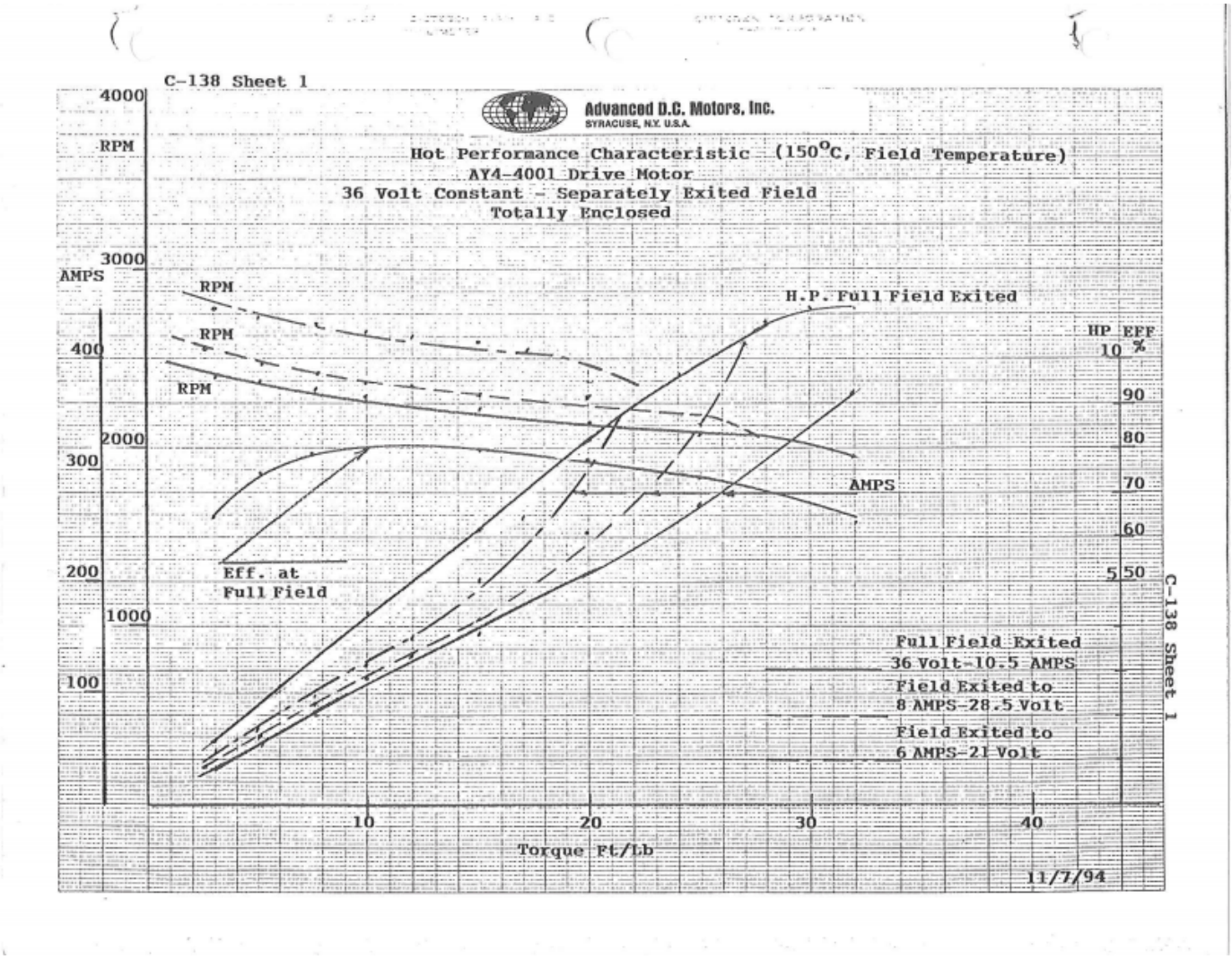


Time (millis)	Input	Angle	DAC Voltage	10 samples of Analog data									
5000	Accelerator	88	0.14	0	1	1	0	0	0	0	0	0	0
40000	Accelerator	95	0.25	15	16	17	16	14	16	14	17	16	15
124000	Accelerator	100	0.38	33	33	34	33	35	33	34	34	34	35
149000	Accelerator	105	0.53	56	55	54	55	54	56	57	55	55	54
168000	Accelerator	110	1	117	118	116	118	117	118	118	117	118	118
188000	Accelerator	115	1.4	174	176	175	175	176	175	175	175	176	175
200000	Accelerator	120	1.91	249	249	248	248	249	248	249	249	248	248
214000	Accelerator	125	2.38	311	311	311	310	311	311	311	323	310	311
227000	Accelerator	130	2.94	388	390	389	387	389	388	389	389	389	389
241000	Accelerator	135	3.37	448	448	447	447	447	447	446	447	447	448
254000	Accelerator	140	3.86	517	517	516	517	516	517	517	518	517	517
266000	Accelerator	145	4.29	631	630	630	631	630	631	631	631	631	630
278000	Accelerator	150	4.79	675	674	673	674	674	674	674	674	673	674
289000	Accelerator	153	4.93	680	676	675	675	675	651	676	676	676	675

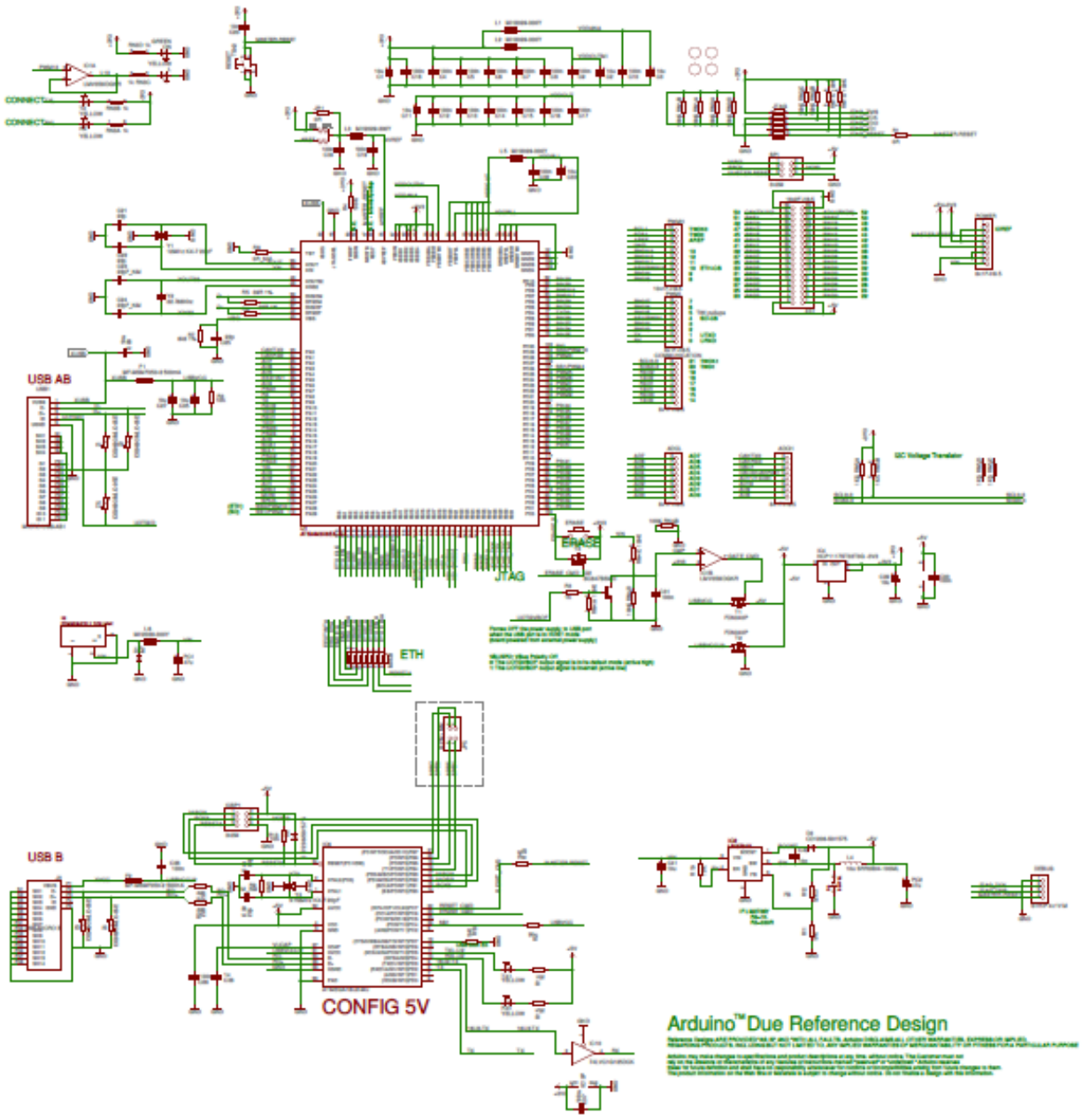
B.4 Center of Mass Calculations

Center of Mass Calculations		Reference origin is the right rear of the cart			
	x (in)	y (in)	weight (lb)	xm (in-lb)	ym (in-lb)
Wheel 1 (large)	5	11	12	60	132
Wheel 2 (large)	40	11	12	480	132
Controller 1	30.5	10.5	9	274.5	94.5
Controller 2	22	10.5	9	198	94.5
Batteries	14.5	48	240	3480	1152
Chair (with driver, variable)	29.5	38	165	4867.5	6270
Wheel 1 (small)	2.75	77.5	6	160.5	465
Wheel 2 (small)	41.25	77.5	6	246	465
Motor 1	23	5.75	50	1150	287.5
Motor 2	22	13.5	50	1100	675
Solenoids	12	10.5	4	48	42
Frame	22	26	180	3960	4680
Total			743	16024.5	14489.5
Center of Mass	21.373 (from right)	19.5 (from rear)			
Actual length and width	37.07	62.82			

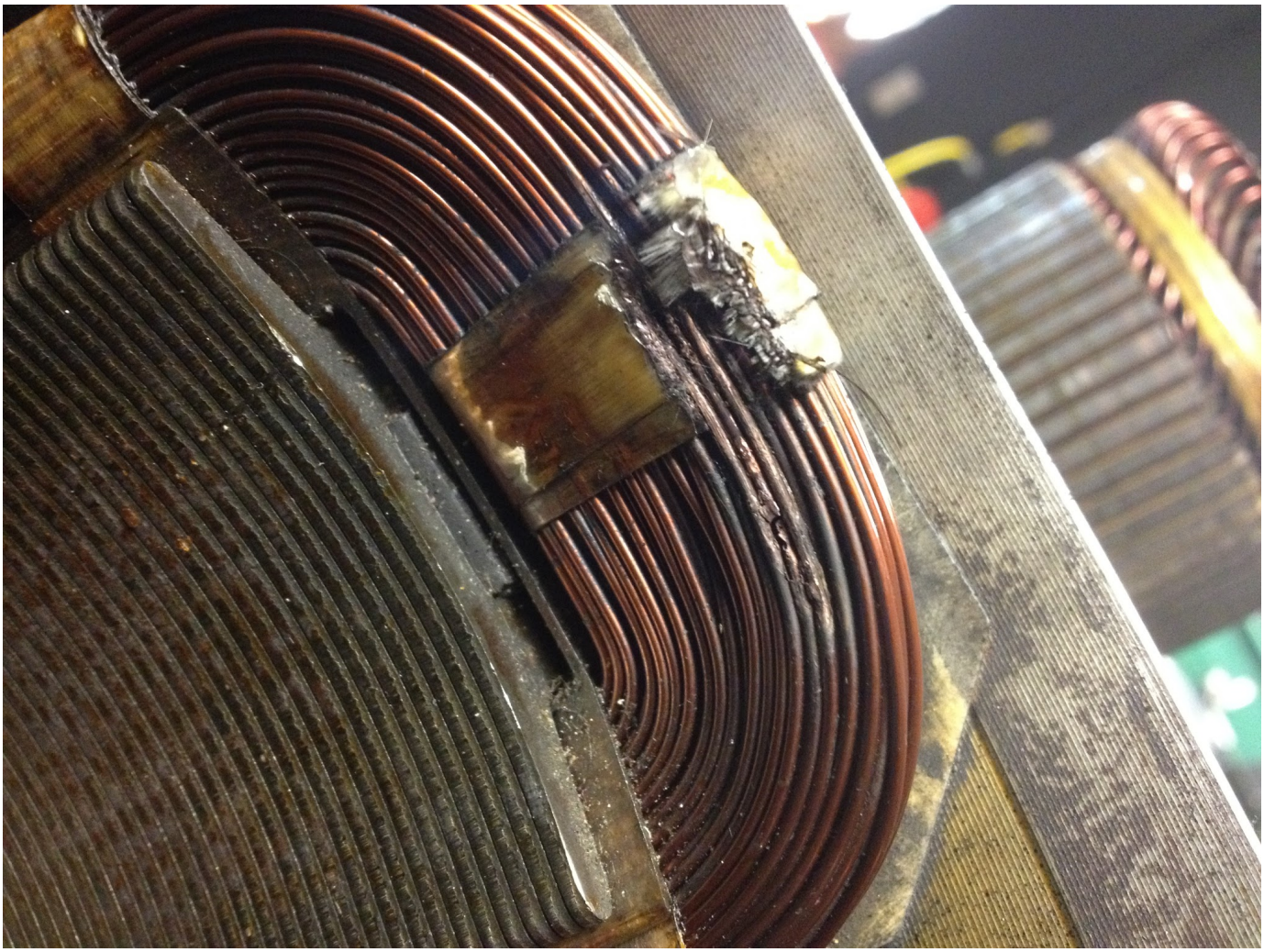
APPENDIX C
DATA SHEETS
C.1 Advanced D.C. Motors AY4-4001 Motors



C.2 Arduino Pin Layout



C.3 Broken Field Windings



APPENDIX D
CODE REFERENCE

```
/*********************************************************************************************
 * Senior Design Project: Electronic "Soft Differential"
 * Credit: UNCA-NCSU Senior Class 2014
 * Purpose: The purpose of this program is to read in various inputs from two wheel speed
 *          encoders, one steering potentiometer, and one throttle potentiometer.  The
 *          code will determine if the car is straight or turning, and by how much it is
 *          turning.  An algorithm will process this and send outputs to two electric
 *          motors which will run at different speeds in a turn to simulate what is
 *          commonly seen in a regular mechanical differential.
 *
 * Versions & Version Notes:
 *   v0.1.0- Code written for analog input of potentiometer inputs for steering and throttle
 *      Credit: Dakota Lazenby
 *   v0.1.1- Added support for motor encoders and modified structure
 *      Credit: Dakota Lazenby
 *   v1.0.0- Ported code over for Arduino Due and added throttle response functionality
 *      Credit: Dakota Lazenby
 *   v1.0.1- Imported simplified working motor encoder code
 *      Credit: Brandon Zschokke, Jennifer Cory, and Hallie Sheaffer
 *   v2.0.0- Combined and created confirmed working throttle code and encoder code
 *      Credit: Brandon Zschokke
 *   v2.1.0- Re-Located the encoder code to a function, cleaned up some variables and added conditional prints
 *      Credit: Dakota Lazenby
 *   v2.2.0- Added in a throttleFilter function.
 *      Credit: Dakota Lazenby
 *   v2.3.0- Added functionality for power LED, checking pack voltage from LiPo battery, new encoder Z pin reads
 *           at high RPMs,
 *      Credit: Dakota Lazenby, Hallie Shaeffer
 *
 * Future Versions / Roadmap:
 *   v2.x.x- Implement Hallie's SD Card code for data logging. Implement Steering Code
 *   v3.0.0- Implement controller code
 *   v3.x.x- Debug and refine control code
 *   v4.0.0- Remove excess global variables and modularize code into various functions
 *********************************************************************************************/

#define DEBUG      1
#define USER_INPUT 1
#define ENCODERS   1
#define SPEED      1
#define ADC_DELAY 10
// Global Declaration Section

int controlScheme = 1;
int loopCount = 0;

// Throttle Declarations
int throttlePin = A11; // Throttle Pot Signal
int leftMotor = DAC0;  // To Protoboard (5)
int rightMotor = DAC1;  // To Protoboard (6)
int minThrottle = 7;
int maxThrottle = 235; // This value needs to correspond to the maximum voltage reading off of the potentiometer
                       // plugged into the formula : maxThrottle = measuredVoltage * (3.3/(2^Res))
int throttle_in_left = 0;
int throttle_in_right = 0;

int Res = 10;  // Resolution
int noiseDelay = 10; //Delay to ensure that analogReads are performed correctly

```

```
// Steering Declarations
int steeringPin = A10; // Steering Pot Signal
int steeringMax = 600;   //THESE VALUES NEED TO BE MEASURED AND ENTERED CORRECTLY
int steeringMidpt = 400;  //IDEALLY WE WRITE CALIBRATION CODE FOR THESE VALUES
int steeringMin = 200;
int Lr = 35;                    //35 inches from center to center on rear wheels
int minTurnRadiusRight = 148;  //Minimum turn radius in inches
int minTurnRadiusLeft = 244;
int leftSteerBuffer = -75;
int rightSteerBuffer = 75;

// Encoder Declarations
 //CHECK THESE PINOUTS!!!!!
 int encoder0PinA = 25; // Encoder A2
 int encoder0PinB = 29; // Encoder A1
 int encoder0PinZ = 33;
 int encoder1PinA = 23; // Encoder B2
 int encoder1PinB = 27; // Encoder B1
 int encoder1PinZ = 31;
 int maxRPM = 20;
 int RPM_0_Last = 0;
 int RPM_1_Last = 0;

 //PID control Variables
 int Kp = 10;
 int Ki = 1;
 int Kd = 0;
 long loopStart = 0;
 long loopEnd = 5000; //Need to tune this value for first run
 long dT = 0;

 double control = 0;
 double error = 0;
 double controlPrev = 0;
 double errorPrev = 0;

 //Encoder read variables
 int encoderPos = 0;
 int encoderPinALast = LOW;
 int encoderSampleTime = 125;  //Sample time for the encoders (in milliSeconds)
 int RPM_0 = 0;
 int RPM_1 = 0;

 double steerLast = 0;

 //Pack voltage variables
 int minPackVoltage = 720;  //Integer value corresponding to roughly 9.65 volts (safe operating LiPo voltage)

 // Extra Pins
 int PwrIn = A9;  //Pin to read the battery pack nominal voltage
 int LEDPwr = 11; //Pin to control the "power to arduino" LED
 int SDPin = 10;
 int PullDownPin = 9;          // Press Sensor pin
 int LEDPin = 8;          //Red LED in digital pin 8 with resistor in series.
//--------------------------------------------------------------------------------------------
```

```
/*****************************************************************************************
 *****************************************************************************************
 * Functions are defined below:
 *****************************************************************************************
 *****************************************************************************************/

int readEncoderAB(int encoderPinA, int encoderPinB){
  int n = 0;
  int RPM = 0;
    uint32_t time = millis();
   while(millis() < (time + encoderSampleTime)){

   n = digitalRead(encoderPinA);
   if ((encoderPinALast == LOW) && (n == HIGH)) {
     if (digitalRead(encoderPinB) == LOW) {
       encoderPos--;
     } else {
       encoderPos++;
     }
   }
   encoderPinALast = n;
 }
  encoderPos = 0;
  RPM = (((encoderPos*(1000/encoderSampleTime)*60)/400)/5);  //In terms of wheel RPM
  return(RM);
}

int readEncoderZ(int encoderPinZ){
  int RPM = 0;
  int Z_Rev = 0;
    long time = micros();
    long no_time = 0;
    //Serial.print("Begin Read");
    while((digitalRead(encoderPinZ) == LOW)&&((no_time = ((micros() - time)/1000)) < 125)) {}
    while((digitalRead(encoderPinZ) == HIGH)&&((no_time = ((micros() - time)/1000)) < 125)) {}
    long time1 = micros();     //Take a timestamp
     while ((Z_Rev >= 0)&&(Z_Rev < 20)&&((no_time = ((micros() - time)/1000)) < 125)) {     // Turn 10 Revolutions
        //Serial.print("Reading");
        if ((digitalRead(encoderPinZ)) == LOW) {  //Test for high pulse
          while (((digitalRead(encoderPinZ)) == LOW)&&((no_time = ((micros() - time)/1000)) < 125)) {}  //Wait for a pulse
            Z_Rev++;
         }
      }
      long time2 = micros() - time1;
      RPM = ((Z_Rev + 1)*1000*1000*60/time2)/5;
      if(RPM > 750){
       RPM = 0;
      }
      if(no_time > 150){
        Serial.println("NO_TIME_NO_TIME_NO_TIME_NO_TIME_NO_TIME_NO_TIME_NO_TIME_NO_TIME_NO_TIME_NO_TIME");
        RPM = 0;
      }
 return(RPM);
}

//*************************************************************************************************
//*************************************************************************************************
```

```
//*************************************************************************************************************
//*************************************************************************************************************

int packVoltage(){
    int Voltage = analogRead(PwrIn);
    delay(ADC_DELAY);
    if(Voltage < minPackVoltage) {
       while(1){
         //Infinite Loop Halting all other processes other than indicating low voltage (LED Blinks)
         digitalWrite(LEDPwr, HIGH);
         delay(500);
         digitalWrite(LEDPwr, LOW);
         delay(500);
       }
    }
}

//*************************************************************************************************************
//*************************************************************************************************************

void differential(int Steering, int throttle){
  //UNFINISHED, NEED TO WORK OUT EQUATIONS FOR Ri AND THEN WORK OUT GOOD CODE FOR DIFFERENTIAL
  //THESE EQUATIONS ARE EXTREMELY RUDIMENTARY AND NEED TUNING BUT THEY SHOULD "WORK"...
    double IV, OV, Ri;
    double idealDiff;
    double steerDifference = Steering - steeringMidpt;
    double controlApplied;
    double rpmDifference;

    if(((steerDifference >= 0) && (steerLast <= 0)) || ((steerDifference <= 0) && (steerLast >= 0))){
       control = 0;  //This is the "passed through zero" check and resets control to accomodate that
    }

    if(steerDifference < leftSteerBuffer){
      Ri = minTurnRadiusLeft*((steeringMidpt - steeringMin)/(steeringMidpt - Steering));
      if((RPM_1 == 0)){
        rpmDifference = 0;
      } else {
        rpmDifference = ((RPM_1) - (RPM_0))/(RPM_1);  //Outer wheel minus inner wheel
      }
    } else if(steerDifference > rightSteerBuffer){
      Ri = minTurnRadiusRight*((steeringMax - steeringMidpt)/(steeringMax - Steering));
      rpmDifference = ((RPM_0) - (RPM_1))/(RPM_0);  //Outer wheel minus inner wheel
      if((RPM_0 == 0)){
        rpmDifference = 0;
      } else {
        rpmDifference = ((RPM_0) - (RPM_1))/(RPM_0);  //Outer wheel minus inner wheel
      }
    } else {
      Ri = 35000; // This value is "infinity" because the radius of curvature for a straight line is infinity
    }

    #if DEBUG
        Serial.print("rpmDifference = ");
        Serial.println(rpmDifference);
        Serial.print("Ri = ");
        Serial.println(Ri);
    #endif
```

```
232         OV = throttle;      //currently throttle referenced, need to get in speed referenced??
233         IV = (Ri/(Ri+Lr))*OV;
234
235         #if DEBUG
236           Serial.print("OV = ");
237           Serial.print(OV);
238           Serial.print(".....IV = ");
239           Serial.println(IV);
240         #endif
241
242         idealDiff = ((OV - IV)/OV);
243
244         #if DEBUG
245           Serial.print("Weighted difference of velocities = ");
246           Serial.println(idealDiff);
247         #endif
248
249         // INSERT PI CONTROL
250         error = idealDiff - rpmDifference;
251
252         #if DEBUG
253           Serial.print("Error (calculated)");
254           Serial.println(error);
255         #endif
256
257         dT = loopEnd - loopStart;
258         dT = dT/(1000*1000);
259         //dT = 1;
260         control = (Kp*error)+(Ki*(((error+errorPrev)/2)*dT+controlPrev))+(Kd*((error-errorPrev)/dT));
261
262         #if DEBUG
263           Serial.print("deltaTime = ");
264           Serial.println(dT);
265           Serial.print("Control (calculated) = ");
266           Serial.println(control);
267         #endif
268
269         controlApplied = control;
270
271         if(steerDifference < leftSteerBuffer){
272           throttle_in_right = throttle;
273           throttle_in_left = throttle - control; // plus or minus control???
274         } else if(steerDifference > rightSteerBuffer){
275           throttle_in_right = throttle - control; // plus or minus control???
276           throttle_in_left = throttle;
277         } else {
278           throttle_in_right = throttle;
279           throttle_in_left = throttle;
280         }
281
282         #if DEBUG
283           Serial.print("Throttle (right calculated) = ");
284           Serial.println(throttle_in_right);
285           Serial.print("Throttle (left calculated) = ");
286           Serial.println(throttle_in_left);
287         #endif
288
289         errorPrev = error;
290         controlPrev = control;
291         steerLast = steerDifference;
```

```
292
293   }
294
295   //*************************************************************************************************************
296   //*************************************************************************************************************
297
298   void calibrateThrottle () {
299      //This function establishes min and max throttle values_____________________________
300     pinMode (PullDownPin, INPUT);
301     digitalWrite(PullDownPin, HIGH);
302     pinMode(LEDPin, OUTPUT);
303     while (digitalRead(PullDownPin) == HIGH)  {
304      digitalWrite(LEDPin, LOW);
305      delay(500);
306      digitalWrite(LEDPin, HIGH);
307      delay(500);
308     }
309     analogReadResolution(Res);
310     minThrottle = analogRead(throttlePin);
311     delay(ADC_DELAY);
312     minThrottle = analogRead(throttlePin);
313     delay(ADC_DELAY);
314     #if DEBUG
315       Serial.print("minThrottle = ");
316       Serial.print(minThrottle);
317     #endif
318     digitalWrite(LEDPin, LOW);
319     delay(1000);
320     while (digitalRead(PullDownPin) == LOW)  {
321      digitalWrite(LEDPin, LOW);
322      delay(500);
323      digitalWrite(LEDPin, HIGH);
324      delay(500);
325     }
326     analogReadResolution(Res);
327     maxThrottle = analogRead(throttlePin);
328     delay(ADC_DELAY);
329     maxThrottle = analogRead(throttlePin);
330     delay(ADC_DELAY);
331     maxThrottle = (maxThrottle + 10);
332     #if DEBUG
333       Serial.print(".......maxThrottle = ");
334       Serial.println(maxThrottle);
335     #endif
336     digitalWrite(LEDPin, LOW);
337     delay(500);
338   }
339
340   //*************************************************************************************************************
341   //*************************************************************************************************************
342
343   void calibrateSteering () {
344      //This function establishes min and max throttle values_____________________________
345     pinMode (PullDownPin, INPUT);
346     digitalWrite(PullDownPin, HIGH);
347     pinMode(LEDPin, OUTPUT);
348     while (digitalRead(PullDownPin) == HIGH)  {
349      digitalWrite(LEDPin, LOW);
350      delay(200);
351      digitalWrite(LEDPin, HIGH);
```

```
//*************************************************************************************************
//*************************************************************************************************


void setup() {
  // Setup encoder inputs
  pinMode (encoder0PinA,INPUT);
  pinMode (encoder0PinB,INPUT);
  pinMode (encoder0PinZ,INPUT);
  pinMode (encoder1PinA,INPUT);
  pinMode (encoder1PinB,INPUT);
  pinMode (encoder1PinZ,INPUT);
  pinMode (LEDPwr, OUTPUT);
  digitalWrite(LEDPwr, HIGH);
  Serial.begin (9600); // Start serial output for debugging
  loopCount = 0;
  calibrateThrottle();
  calibrateSteering();
}

/*************************************************************************************************
 * \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
 * /\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
 *!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! MAIN LOOP STARTS HERE!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 * /\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
 * \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
 *************************************************************************************************/


void loop() {

  loopStart = millis();
  //packVoltage();  //Check the battery pack to see if it is within safe operating voltage
 // This section of the code uses the throttle code by Dakota to drive the cart from the throttle potentiometer

  analogReadResolution(Res);
  int throttle_in = analogRead(throttlePin);
  delay(ADC_DELAY);   // Delays after analogRead to allow capacitors to charge and cancels cross talk between reads
  throttle_in = analogRead(throttlePin);

  analogReadResolution(Res);
  int Steering = analogRead(steeringPin);
  delay(ADC_DELAY);
  Steering = analogRead(steeringPin);

    #if DEBUG
      #if USER_INPUT
      Serial.print("Gas: ");                          //Debug Statements ~ visual data
      Serial.println(throttle_in);
      Serial.print("Steering");
      Serial.println(Steering);
      #endif
    #endif

  /*if ((RPM_0_Last < maxRPM) || (RPM_1_Last < maxRPM)){
    RPM_0 = readEncoderAB(encoder0PinA, encoder0PinB);  //This function has no output because it changes global variables within the function.
                                                        // Read Encoder pins A and B if RPM is in "low range" (Reads for both encoders)
    RPM_1 = readEncoderAB(encoder1PinA, encoder1PinB);  //This function has no output because it changes global variables within the function.
                                                        // Read Encoder pins A and B if RPM is in "low range" (Reads for both encoders)
  } else {
```

```
    RPM_0 = readEncoderZ(encoder0PinZ);   // Read Encoder pin Z if RPM is in "high range" (Reads for both encoders)
    RPM_1 = readEncoderZ(encoder1PinZ);   // Read Encoder pin Z if RPM is in "high range" (Reads for both encoders)
  }*/

  RPM_0 = readEncoderZ(encoder0PinZ);   // Read Encoder pin Z if RPM is in "high range" (Reads for both encoders)
  RPM_1 = readEncoderZ(encoder1PinZ);   // Read Encoder pin Z if RPM is in "high range" (Reads for both encoders)
  #if DEBUG
    Serial.print("RPM 0 =");
    Serial.print(RPM_0);
    Serial.print("......RPM 1 =");
    Serial.println(RPM_1);
  #endif
  //RPM_0_Last = RPM_0;
  //RPM_1_Last = RPM_1;

  //(insert diff code here)

  throttle_in_left = throttle_in;
  throttle_in_right = throttle_in;

  if((controlScheme == 1) && (loopCount > 0)){
      differential(Steering, throttle_in);
  }

  int leftSpeed  = map(throttle_in_left, minThrottle, maxThrottle, 0, 1023);
  int rightSpeed = map(throttle_in_right, minThrottle, maxThrottle, 0, 1023);

  if( leftSpeed < 5 ){  // Error checking code to ensure we do not have throttle spikes at the extremities
    leftSpeed = 0;
  }
  if ( leftSpeed > 1023 ){
    leftSpeed = 1023;
  }
  if ( rightSpeed < 5 ){
    rightSpeed = 0;
  }
  if( rightSpeed > 1023 ){
    rightSpeed = 1023;
  }

  analogWriteResolution(Res);
  analogWrite(leftMotor, leftSpeed);
  delay(ADC_DELAY/5);
  analogWriteResolution(Res);
  analogWrite(rightMotor, rightSpeed);
  delay(ADC_DELAY/5);

      #if DEBUG
        #if SPEED
        Serial.print("Left Motor Voltage ");                //Debug Statements ~ visual data
        Serial.println(leftSpeed);
        Serial.print("Right Motor Voltage");
        Serial.println(rightSpeed);
        #endif
      #endif

  loopEnd = millis();
  loopCount = loopCount + 1;
  delayMicroseconds(5);
}
```

APPENDIX E
ALTERNATE IDEAS

E.1 PWM Circuitry

Pulse Width Modulation is a way to map an analog signal to a digital signal. The digital signal is a collection of a square waves. The control of the duty cycle of these waves, will produce

an average voltage. The average voltage for the signal is viewed as analog signal for the other devices such as the controllers and motors.

In the case of using two different voltages, an auxiliary device needed to control this difference. This case required amplification of the signal from the microprocessor to the motor controllers. PWM can work greatly to control a solid state relay or to control a transistor. Based on the duty cycle, ON/OFF duration, the transistor is opened and closed to control the voltage seen across the controlled device. The Arduino contain a PWM I/O. Using the analogWrite() function in the microprocessor this functionality is implemented with help of small circuitry. Figure 22 shows the inputs and output of this simple device. The VCC voltage is the maximum voltage needed. The transistor is controlled through one of the Arduino pins. A $1\,\mu F$ capacitor was added to eliminate noise and chatter in the signal.

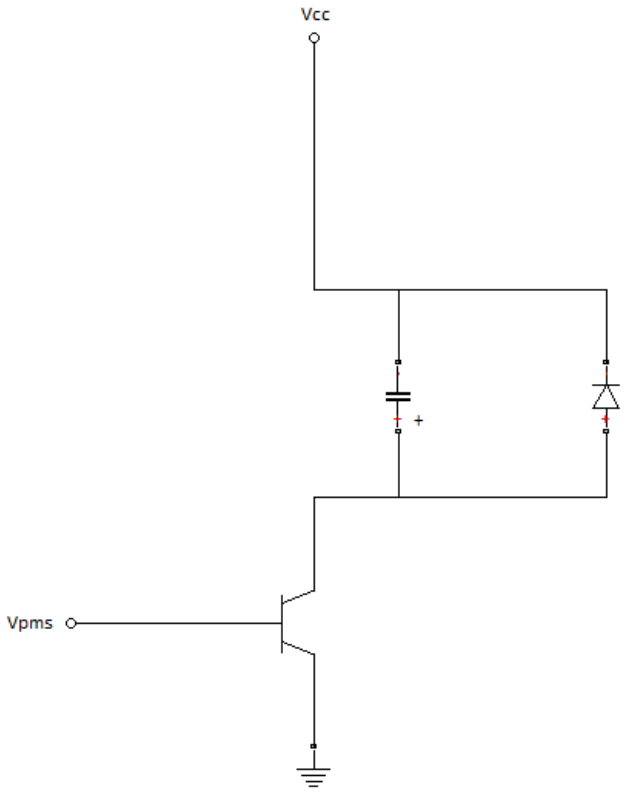


Fig. 22: PWM Circuit

APPENDIX F
PARTS & DATA SHEETS

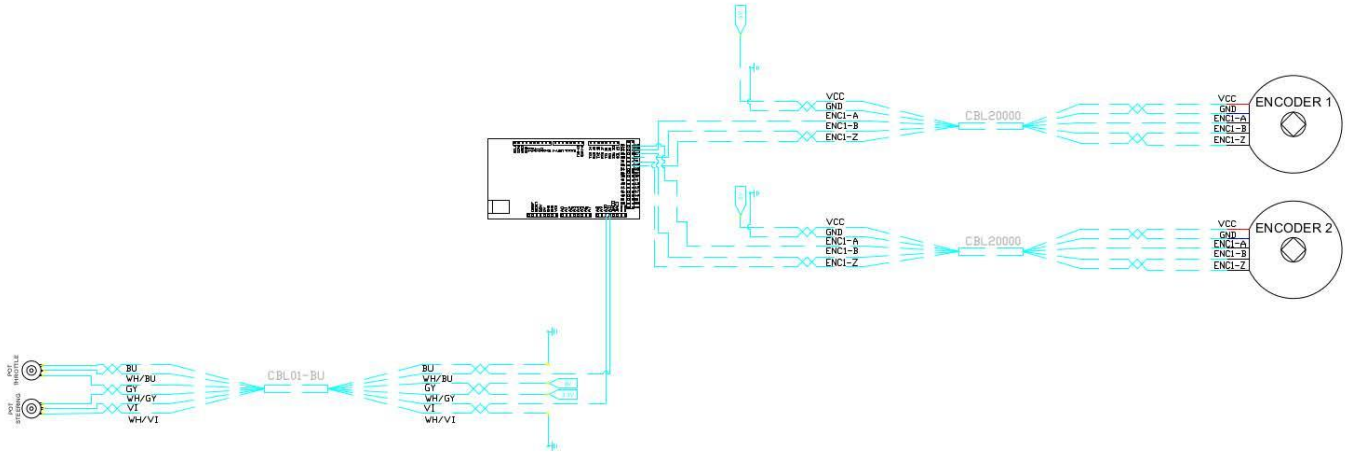


Fig. 23: Sensor Connections to Arduino

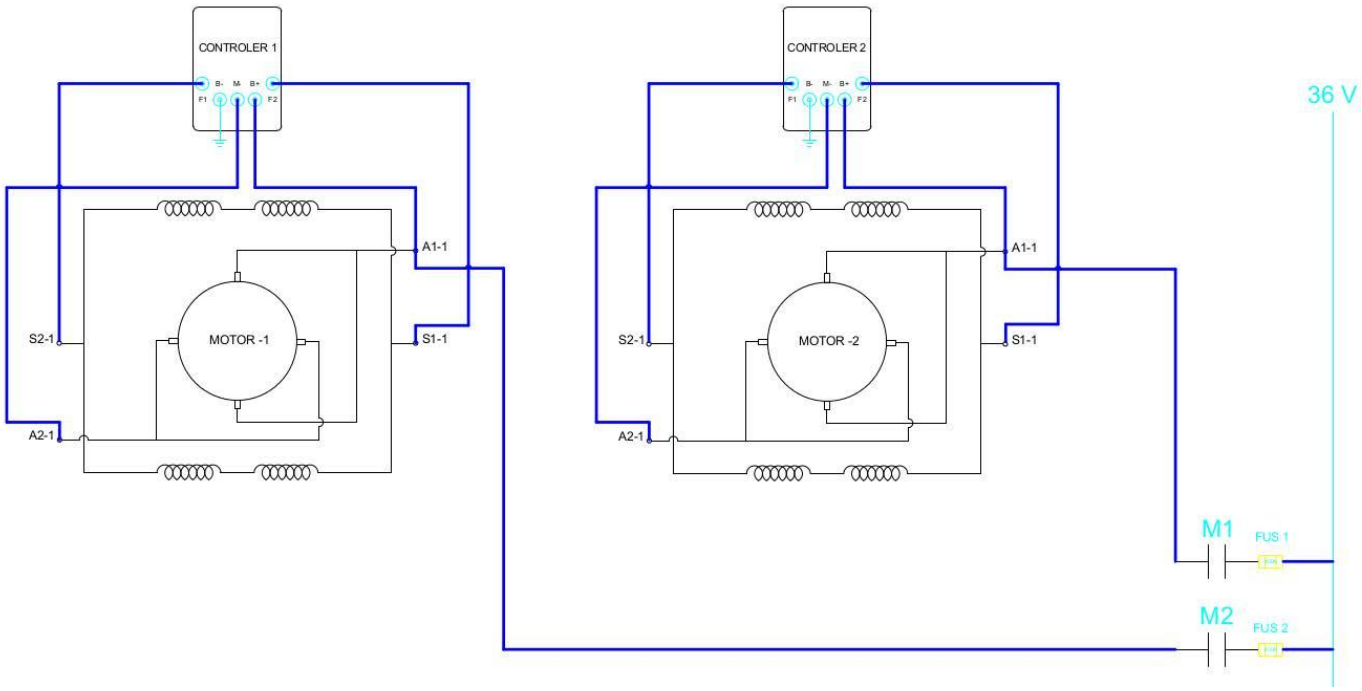


Fig. 24: Battery Connection to Controllers and Motors

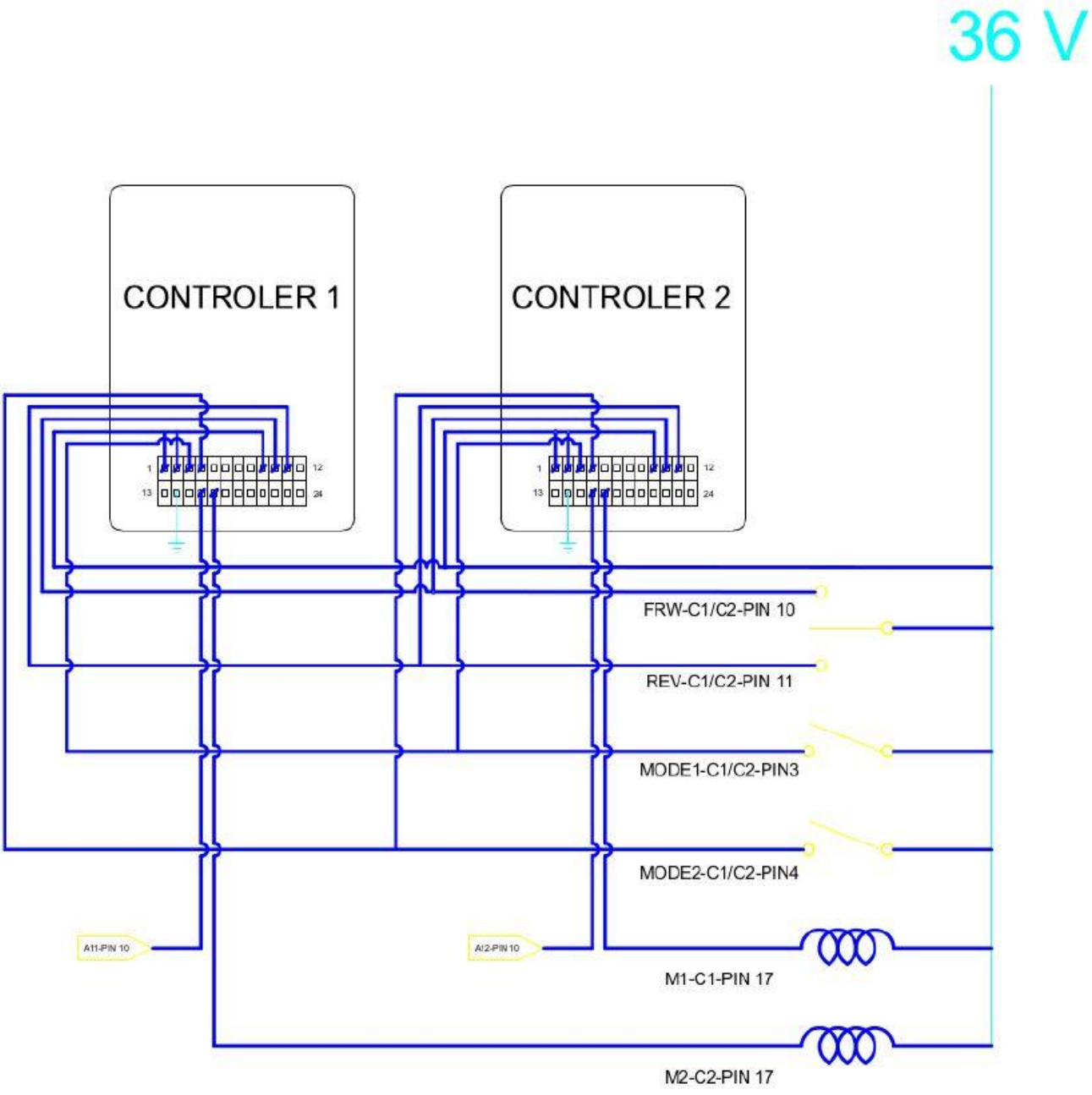


Fig. 25: Control Panel Connections

MC78M00, MC78M00A Series

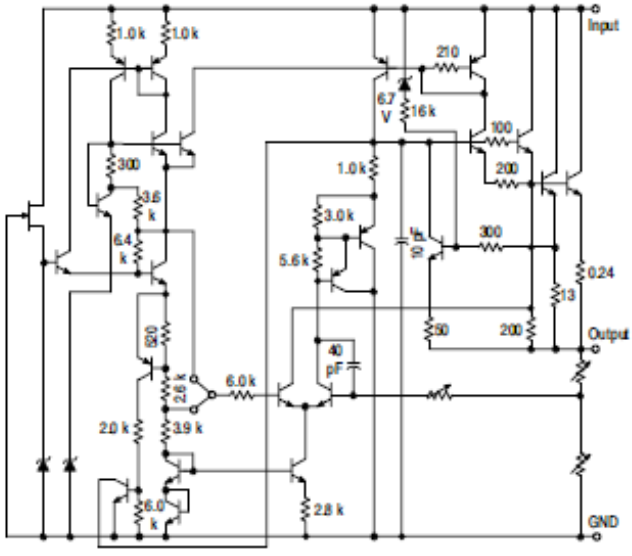
500 mA Positive Voltage Regulators

The MC78M00/MC78M00A Series positive voltage regulators are identical to the popular MC7800 Series devices, except that they are specified for only half the output current. Like the MC7800 devices, the MC78M00 three−terminal regulators are intended for local, on−card voltage regulation.

Internal current limiting, thermal shutdown circuitry and safe−area compensation for the internal pass transistor combine to make these devices remarkably rugged under most operating conditions. Maximum output current, with adequate heatsinking is 500 mA.

Features

- No External Components Required
- Internal Thermal Overload Protection
- Internal Short Circuit Current Limiting
- Output Transistor Safe−Area Compensation
- MC78M00A High Accuracy (±2%)
  Available for 5.0 V, 8.0 V, 12 V and 15 V
- Pb−Free Packages are Available*



This device contains 28 active transistors.

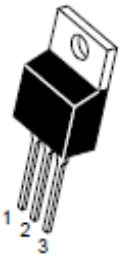
Figure 1. Representative Schematic Diagram

*For additional information on our Pb−Free strategy and soldering details, please download the ON Semiconductor Soldering and Mounting Techniques Reference Manual, SOLDERRM/D.

ON Semiconductor®

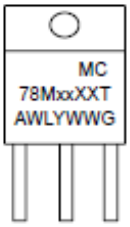
http://onsemi.com

MARKING DIAGRAMS



TO−220
T SUFFIX
CASE 221A

Heatsink surface connected to Pin 2.



xx = Voltage Option
XX = Appropriate Suffix Options
A = Assembly Location
WL = Wafer Lot
Y = Year
WW = Work Week
G = Pb−Free Package

DPAK−3
DT SUFFIX
CASE 369C

Heatsink surface (shown as terminal 4 in case outline drawing) is connected to Pin 2.

xxxxx = Device Type and Voltage Option Code
A = Assembly Location
L = Wafer Lot
Y = Year
WW = Work Week
G = Pb−Free Package

Pin 1. Input
2. Ground
3. Output

ORDERING INFORMATION

See detailed ordering and shipping information in the package dimensions section on page 10−14 of this data sheet.

DEVICE MARKING INFORMATION

See general marking information in the device marking section on page 10 of this data sheet.

Fig. 26: LM7808 Voltage Regulator

MC78M08C/AC/B/AB ELECTRICAL CHARACTERISTICS (V_I = 14 V, I_O = 350 mA, T_J = T_{low} to T_{high}, $P_D \le$ 5.0 W, unless otherwise noted) (Note 3)

Characteristics	Symbol	Min	Typ	Max	Unit
Output Voltage (T_J = 25°C) MC78M08C MC78M08AC	V_O	 7.70 7.84	 8.0 8.0	 8.30 8.16	Vdc
Output Voltage Variation (10.5 Vdc ≤ V_I ≤ 23 Vdc, 5.0 mA ≤ I_O ≤ 350 mA) MC78M08C MC78M08AC	V_O	 7.6 7.7	 – –	 8.4 8.3	Vdc
Line Regulation (T_J = 25°C, 10.5 Vdc ≤ V_I ≤ 25 Vdc, I_O = 200 mA)	Reg_{line}	–	6.0	50	mV
Load Regulation (T_J = 25°C, 5.0 mA ≤ I_O ≤ 500 mA) (T_J = 25°C, 5.0 mA ≤ I_O ≤ 200 mA)	Reg_{load}	 – –	 25 10	 160 80	mV
Input Bias Current (T_J = 25°C)	I_{IB}	–	3.2	6.0	mA
Quiescent Current Change (10.5 Vdc ≤ V_I ≤ 25 Vdc, I_O = 200 mA) (5.0 mA ≤ I_O ≤ 350 mA)	ΔI_{IB}	 – –	 – –	 0.8 0.5	mA
Output Noise Voltage (T_A = 25°C, 10 Hz ≤ f ≤ 100 kHz)	V_n	–	52	–	μV
Ripple Rejection (I_O = 100 mA, f = 120 Hz, 11.5 V ≤V_I ≤ 21.5 V) (I_O = 300 mA, f = 120 Hz, 11.5 V ≤V_I ≤ 21.5 V, T_J = 25°C)	RR	 56 56	 – 80	 – –	dB
Dropout Voltage (T_J = 25°C)	V_I–V_O	–	2.0	–	Vdc
Short Circuit Current Limit (T_J = 25°C, V_I = 35 V)	I_{OS}	–	50	–	mA
Average Temperature Coefficient of Output Voltage (I_O = 5.0 mA)	$\Delta V_O/\Delta T$	–	±0.2	–	mV/°C
Peak Output Current (T_J = 25°C)	I_O	–	700	–	mA

3. T_{low} = 0°C for MC78MxxAC, C; −40°C for MC78MxxAB, B T_{high} = +125°C for MC78MxxAB, AC, B, C

Fig. 27: LM7808 Voltage Regulator

National Semiconductor

April 1997

LMC7660
Switched Capacitor Voltage Converter

General Description

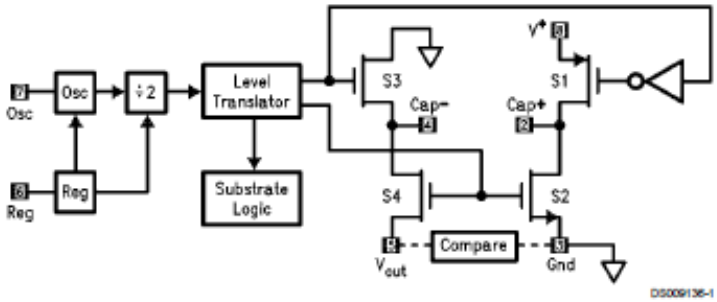
The LMC7660 is a CMOS voltage converter capable of converting a positive voltage in the range of +1.5V to +10V to the corresponding negative voltage of −1.5V to −10V. The LMC7660 is a pin-for-pin replacement for the industry-standard 7660. The converter features: operation over full temperature and voltage range without need for an external diode, low quiescent current, and high power efficiency.

The LMC7660 uses its built-in oscillator to switch 4 power MOS switches and charge two inexpensive electrolytic capacitors.

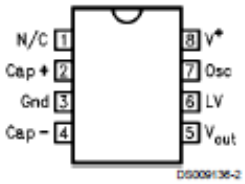
Features

- Operation over full temperature and voltage range without an external diode
- Low supply current, 200 µA max
- Pin-for-pin replacement for the 7660
- Wide operating range 1.5V to 10V
- 97% Voltage Conversion Efficiency
- 95% Power Conversion Efficiency
- Easy to use, only 2 external components
- Extended temperature range
- Narrow SO-8 Package

Block Diagram



Pin Configuration



Ordering Information

Package	Temperature Range Industrial −40°C to +85°C	NSC Drawing
8-Lead Molded DIP	LMC7660IN	N08E
8-Lead Molded Small Outline	LMC7660IM	M08A

Fig. 28: LMC7660 Chip

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	10.5V
Input Voltage on Pin 6, 7 (Note 2)	−0.3V to (V^+ + 0.3V) for $V^+ < 5.5V$ (V^+ − 5.5V) to (V^+ + 0.3V) for $V^+ > 5.5V$
Current into Pin 6 (Note 2)	20 µA
Output Short Circuit Duration ($V^+ \leq 5.5V$)	Continuous
Power Dissipation (Note 3)	
Dual-In-Line Package	1.4W
Surface-Mount Package	0.6W
T_J Max (Note 3)	150°C
θ_{JA} (Note 3)	
Dual-In-Line Package	90°C/W
Surface-Mount Package	160°C/W
Storage Temp. Range	−65°C ≤ T ≤ 150°C
Lead Temperature (Soldering, 5 sec.)	260°C
ESD Tolerance (Note 7)	± 2000V

Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	Typ	LMC7660IN/ LMC7660IM Limit (Note 5)	Units Limits
I_s	Supply Current	$R_L = \infty$	120	200 **400**	µA max
V^+H	Supply Voltage Range High (Note 6)	R_L = 10 kΩ, Pin 6 Open Voltage Efficiency ≥ 90%	3 to 10	3 to 10 **3 to 10**	V
V^+L	Supply Voltage Range Low	R_L = 10 kΩ, Pin 6 to Gnd. Voltage Efficiency ≥ 90%	1.5 to 3.5	1.5 to 3.5 **1.5 to 3.5**	V
R_{out}	Output Source Resistance	I_L = 20 mA	55	100 **120**	Ω max
		V = 2V, I_L = 3 mA Pin 6 Short to Gnd.	110	200 **300**	Ω max
F_{osc}	Oscillator Frequency		10		kHz
P_{eff}	Power Efficiency	R_L = 5 kΩ	97	95 **90**	% min
$V_{o\ eff}$	Voltage Conversion Efficiency	$R_L = \infty$	99.9	97 **95**	% min
I_{osc}	Oscillator Sink or Source Current	Pin 7 = Gnd. or V^+	3		µA

Note 1: Absolute Maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 4 for conditions.

Note 2: Connecting any input terminal to voltages greater than V^+ or less than ground may cause destructive latchup. It is recommended that no inputs from sources operating from external supplies be applied prior to "power-up" of the LMC7660.

Note 3: For operation at elevated temperature, these devices must be derated based on a thermal resistance of θ_{ja} and T_j max, $T_j = T_A + \theta_{ja} P_D$.

Note 4: Boldface numbers apply at temperature extremes. All other numbers apply at T_A = 25°C, V^+ = 5V, C_{osc} = 0, and apply for the LMC7660 unless otherwise specified. Test circuit is shown in *Figure 1*.

Note 5: Limits at room temperature are guaranteed and 100% production tested. Limits in **boldface** are guaranteed over the operating temperature range (but not 100% tested), and are not used to calculate outgoing quality levels.

Note 6: The LMC7660 can operate without an external diode over the full temperature and voltage range. The LMC7660 can also be used with the external diode Dx, when replacing previous 7660 designs.

Note 7: The test circuit consists of the human body model of 100 pF in series with 1500Ω.

Electrical Characteristics (Note 4) (Continued)

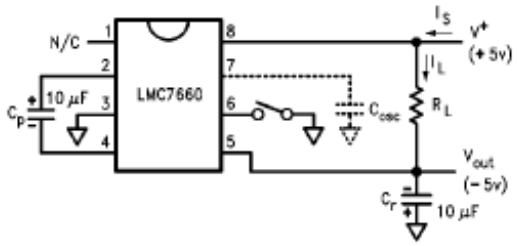
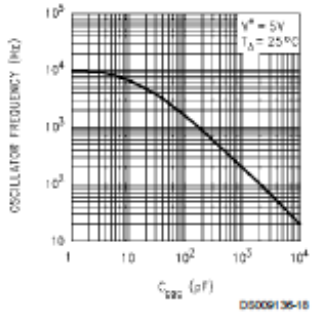


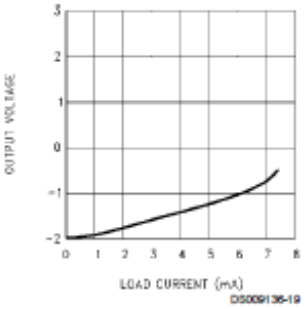
FIGURE 1. LMC7660 Test Circuit

Typical Performance Characteristics

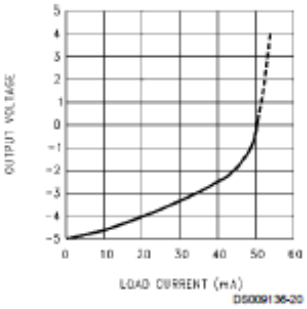
OSC Freq. vs OSC Capacitance



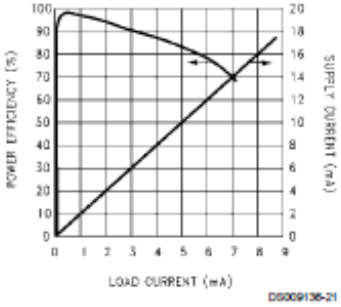
V_{out} vs I_{out} @ V^+ = 2V



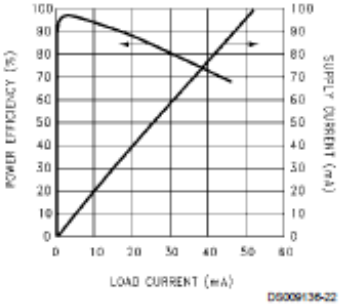
V_{out} vs I_{out} @ V^+ = 5V



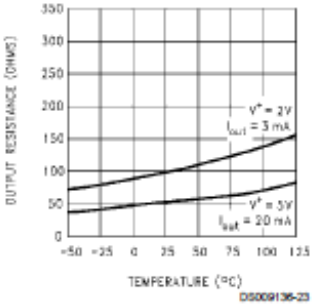
Supply Current & Power Efficiency vs Load Current (V^+ = 2V)



Supply Current & Power Efficiency vs Load Current (V^+ = 5V)



Output Source Resistiance as a Function of Temperature



LM324, LM324A, LM224, LM2902, LM2902V, NCV2902

Single Supply Quad Operational Amplifiers

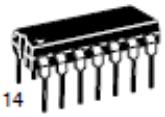
The LM324 series are low−cost, quad operational amplifiers with true differential inputs. They have several distinct advantages over standard operational amplifier types in single supply applications. The quad amplifier can operate at supply voltages as low as 3.0 V or as high as 32 V with quiescent currents about one−fifth of those associated with the MC1741 (on a per amplifier basis). The common mode input range includes the negative supply, thereby eliminating the necessity for external biasing components in many applications. The output voltage range also includes the negative power supply voltage.

Features

- Short Circuited Protected Outputs
- True Differential Input Stage
- Single Supply Operation: 3.0 V to 32 V
- Low Input Bias Currents: 100 nA Maximum (LM324A)
- Four Amplifiers Per Package
- Internally Compensated
- Common Mode Range Extends to Negative Supply
- Industry Standard Pinouts
- ESD Clamps on the Inputs Increase Ruggedness without Affecting Device Operation
- NCV Prefix for Automotive and Other Applications Requiring Unique Site and Control Change Requirements; AEC−Q100 Qualified and PPAP Capable
- These Devices are Pb−Free, Halogen Free/BFR Free and are RoHS Compliant

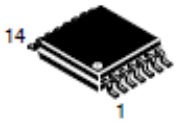
ON Semiconductor®

http://onsemi.com



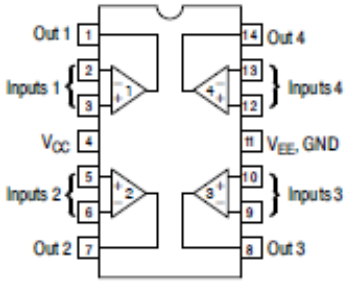
PDIP−14
N SUFFIX
CASE 646

SOIC−14
D SUFFIX
CASE 751A



TSSOP−14
DTB SUFFIX
CASE 948G

PIN CONNECTIONS



(Top View)

ORDERING INFORMATION

See detailed ordering and shipping information in the package dimensions section on page 10 of this data sheet.

DEVICE MARKING INFORMATION

See general marking information in the device marking section on page 11 of this data sheet.

Fig. 29: LM324 Op Amp

LM324, LM324A, LM224, LM2902, LM2902V, NCV2902

MAXIMUM RATINGS (T_A = +25°C, unless otherwise noted.)

Rating		Symbol	Value	Unit
Power Supply Voltages				Vdc
Single Supply		V_{CC}	32	
Split Supplies		V_{CC}, V_{EE}	±16	
Input Differential Voltage Range (Note 1)		V_{IDR}	±32	Vdc
Input Common Mode Voltage Range (Note 2)		V_{ICR}	−0.3 to 32	Vdc
Output Short Circuit Duration		t_{SC}	Continuous	
Junction Temperature		T_J	150	°C
Thermal Resistance, Junction−to−Air (Note 3)	Case 646	$R_{\theta JA}$	118	°C/W
	Case 751A		156	
	Case 948G		190	
Storage Temperature Range		T_{stg}	−65 to +150	°C
ESD Protection at any Pin		V_{esd}		V
Human Body Model			2000	
Machine Model			200	
Operating Ambient Temperature Range		T_A		°C
	LM224		−25 to +85	
	LM324, 324A		0 to +70	
	LM2902		−40 to +105	
	LM2902V, NCV2902 (Note 4)		−40 to +125	

Stresses exceeding Maximum Ratings may damage the device. Maximum Ratings are stress ratings only. Functional operation above the Recommended Operating Conditions is not implied. Extended exposure to stresses above the Recommended Operating Conditions may affect device reliability.

1. Split Power Supplies.
2. For supply voltages less than 32 V, the absolute maximum input voltage is equal to the supply voltage.
3. All $R_{\theta JA}$ measurements made on evaluation board with 1 oz. copper traces of minimum pad size. All device outputs were active.
4. *NCV2902 is qualified for automitive use.*

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V, V_{EE} = GND, T_A = 25°C, unless otherwise noted.)

Characteristics	Symbol	LM224			LM324A			LM324			LM2902			LM2902V/NCV2902			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage V_{CC} = 5.0 V to 30 V V_{ICR} = 0 V to V_{CC} −1.7 V, V_O = 1.4 V, R_S = 0 Ω	V_{IO}																mV
T_A = 25°C		–	2.0	5.0	–	2.0	3.0	–	2.0	7.0	–	2.0	7.0	–	2.0	7.0	
T_A = T_{high} (Note 5)		–	–	7.0	–	–	5.0	–	–	9.0	–	–	10	–	–	13	
T_A = T_{low} (Note 5)		–	–	7.0	–	–	5.0	–	–	9.0	–	–	10	–	–	10	
Average Temperature Coefficient of Input Offset Voltage T_A = T_{high} to T_{low} (Notes 5 and 7)	$\Delta V_{IO}/\Delta T$	–	7.0	–	–	7.0	30	–	7.0	–	–	7.0	–	–	7.0	–	μV/°C
Input Offset Current T_A = T_{high} to T_{low} (Note 5)	I_{IO}	– –	3.0 –	30 100	– –	5.0 –	30 75	– –	5.0 –	50 150	– –	5.0 –	50 200	– –	5.0 –	50 200	nA
Average Temperature Coefficient of Input Offset Current T_A = T_{high} to T_{low} (Notes 5 and 7)	$\Delta I_{IO}/\Delta T$	–	10	–	–	10	300	–	10	–	–	10	–	–	10	–	pA/°C
Input Bias Current T_A = T_{high} to T_{low} (Note 5)	I_{IB}	– –	−90 –	−150 −300	– –	−45 –	−100 −200	– –	−90 –	−250 −500	– –	−90 –	−250 −500	– –	−90 –	−250 −500	nA
Input Common Mode Voltage Range (Note 6) V_{CC} = 30 V	V_{ICR}																V
T_A = +25°C		0	–	28.3	0	–	28.3	0	–	28.3	0	–	28.3	0	–	28.3	
T_A = T_{high} to T_{low} (Note 5)		0	–	28	0	–	28	0	–	28	0	–	28	0	–	28	
Differential Input Voltage Range	V_{IDR}	–	–	V_{CC}	–	–	V_{CC}	–	–	V_{CC}	–	–	V_{CC}	–	–	V_{CC}	V
Large Signal Open Loop Voltage Gain R_L = 2.0 kΩ, V_{CC} = 15 V, for Large V_O Swing	A_{VOL}																V/mV
		50	100	–	25	100	–	25	100	–	25	100	–	25	100	–	
T_A = T_{high} to T_{low} (Note 5)		25	–	–	15	–	–	15	–	–	15	–	–	15	–	–	
Channel Separation 10 kHz ≤ f ≤ 20 kHz, Input Referenced	CS	–	−120	–	–	−120	–	–	−120	–	–	−120	–	–	−120	–	dB
Common Mode Rejection, R_S ≤ 10 kΩ	CMR	70	85	–	65	70	–	65	70	–	50	70	–	50	70	–	dB
Power Supply Rejection	PSR	65	100	–	65	100	–	65	100	–	50	100	–	50	100	–	dB

5. LM224: T_{low} = −25°C, T_{high} = +85°C
LM324/LM324A: T_{low} = 0°C, T_{high} = +70°C
LM2902: T_{low} = −40°C, T_{high} = +105°C
LM2902V & NCV2902: T_{low} = −40°C, T_{high} = +125°C
NCV2902 is qualified for automotive use.
6. The input common mode voltage or either input signal voltage should not be allowed to go negative by more than 0.3 V. The upper end of the common mode voltage range is V_{CC} −1.7 V, but either or both inputs can go to +32 V without damage, independent of the magnitude of V_{CC}.
7. Guaranteed by design.

LM324, LM324A, LM224, LM2902, LM2902V, NCV2902

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V, V_{EE} = GND, T_A = 25°C, unless otherwise noted.)

Characteristics	Symbol	LM224			LM324A			LM324			LM2902			LM2902V/NCV2902			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Output Voltage – High Limit	V_{OH}																V
V_{CC} = 5.0 V, R_L = 2.0 kΩ, T_A = 25°C		3.3	3.5	–	3.3	3.5	–	3.3	3.5	–	3.3	3.5	–	3.3	3.5	–	
V_{CC} = 30 V R_L = 2.0 kΩ (T_A = T_{high} to T_{low}) (Note 8)		26	–	–	26	–	–	26	–	–	26	–	–	26	–	–	
V_{CC} = 30 V R_L = 10 kΩ (T_A = T_{high} to T_{low}) (Note 8)		27	28	–	27	28	–	27	28	–	27	28	–	27	28	–	
Output Voltage – Low Limit, V_{CC} = 5.0 V, R_L = 10 kΩ, T_A = T_{high} to T_{low} (Note 8)	V_{OL}	–	5.0	20	–	5.0	20	–	5.0	20	–	5.0	100	–	5.0	100	mV
Output Source Current (V_{ID} = +1.0 V, V_{CC} = 15 V)	I_{O+}																mA
T_A = 25°C		20	40	–	20	40	–	20	40	–	20	40	–	20	40	–	
T_A = T_{high} to T_{low} (Note 8)		10	20	–	10	20	–	10	20	–	10	20	–	10	20	–	
Output Sink Current	I_{O-}																mA
(V_{ID} = −1.0 V, V_{CC} = 15 V) T_A = 25°C		10	20	–	10	20	–	10	20	–	10	20	–	10	20	–	
T_A = T_{high} to T_{low} (Note 8)		5.0	8.0	–	5.0	8.0	–	5.0	8.0	–	5.0	8.0	–	5.0	8.0	–	
(V_{ID} = −1.0 V, V_O = 200 mV, T_A = 25°C)		12	50	–	12	50	–	12	50	–	–	–	–	–	–	–	µA
Output Short Circuit to Ground (Note 9)	I_{SC}	–	40	60	–	40	60	–	40	60	–	40	60	–	40	60	mA
Power Supply Current (T_A = T_{high} to T_{low}) (Note 8)	I_{CC}																mA
V_{CC} = 30 V V_O = 0 V, R_L = ∞		–	–	3.0	–	1.4	3.0	–	–	3.0	–	–	3.0	–	–	3.0	
V_{CC} = 5.0 V, V_O = 0 V, R_L = ∞		–	–	1.2	–	0.7	1.2	–	–	1.2	–	–	1.2	–	–	1.2	

8. LM224: T_{low} = −25°C, T_{high} = +85°C
LM324/LM324A: T_{low} = 0°C, T_{high} = +70°C
LM2902: T_{low} = −40°C, T_{high} = +105°C
LM2902V & NCV2902: T_{low} = −40°C, T_{high} = +125°C
NCV2902 is qualified for automotive use.
9. The input common mode voltage or either input signal voltage should not be allowed to go negative by more than 0.3 V. The upper end of the common mode voltage range is V_{CC} −1.7 V, but either or both inputs can go to +32 V without damage, independent of the magnitude of V_{CC}.

F.1 Curtis Controllers

APPENDIX D

SPECIFICATIONS

Table D-1 SPECIFICATIONS: 1244 CONTROLLER

Nominal input voltage	24–36 V, 36–48 V, and 48–80V
PWM operating frequency	16 kHz
Electrical isolation to heatsink	500 V ac (minimum)
KSI input voltage (minimum)	16.8 V for 24–36 V systems
KSI input current (no contactors engaged)	160 mA without programmer; 200 mA with programmer
Logic input voltage	>7.5 V High; <1 V Low
Logic input current	10 mA
Operating ambient temperature range	-40°C to 50°C (-40°F to 122°F)
Heatsink overtemperature cutback	85°C (185°F)
Heatsink undertemperature cutback	-25°C (-13°F)
Package environmental rating	IP64 / IP67
Weight	3.9 kg (8.5 lb)
Dimensions (LxWxH)	229 x 178 x 81 mm (9.0" x 7.0" x 3.2")

MODEL NUMBER	NOMINAL BATTERY VOLTAGE (volts)	ARMATURE CURRENT LIMIT (amps)	ARMATURE 2 MIN RATING (amps)	ARMATURE 1 HOUR RATING * (amps)	FIELD 2 MIN RATING (amps)
1244-44XX	24–36	400	400	160	60
1244-45XX	24–36	500	500	175	60
1244-46XX	24–36	600	600	190	60
1244-47XX	24–36	700	700 †	190	60
1244-54XX	36–48	400	400	140	50
1244-55XX	36–48	500	500	160	50
1244-56XX	36–48	600	600	160	50
1244-64XX	48–80	400	400	125	50
1244-65XX	48–80	500	500	140	50
1244-66XX	48–80	600	600 †	140	50

* at 25°C ambient temperature
† 1-minute rating

Fig. 30

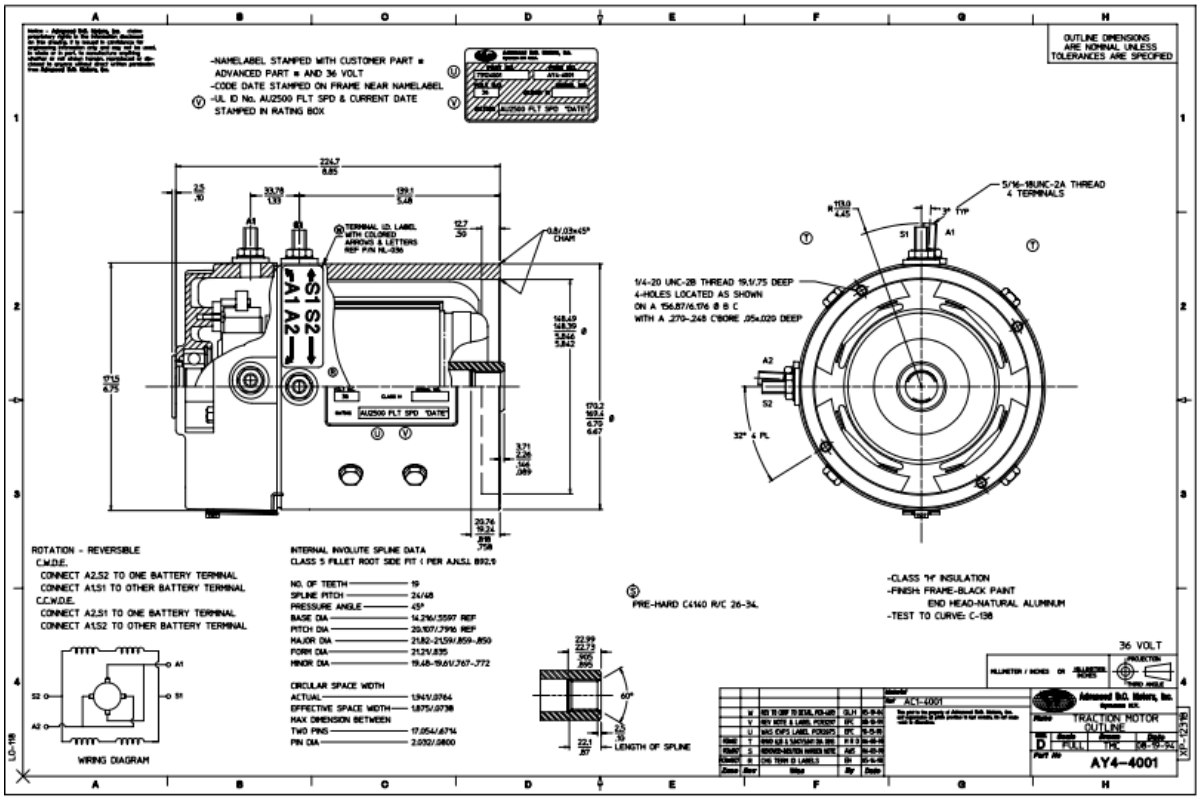


Fig. 31

F.2 Batteries

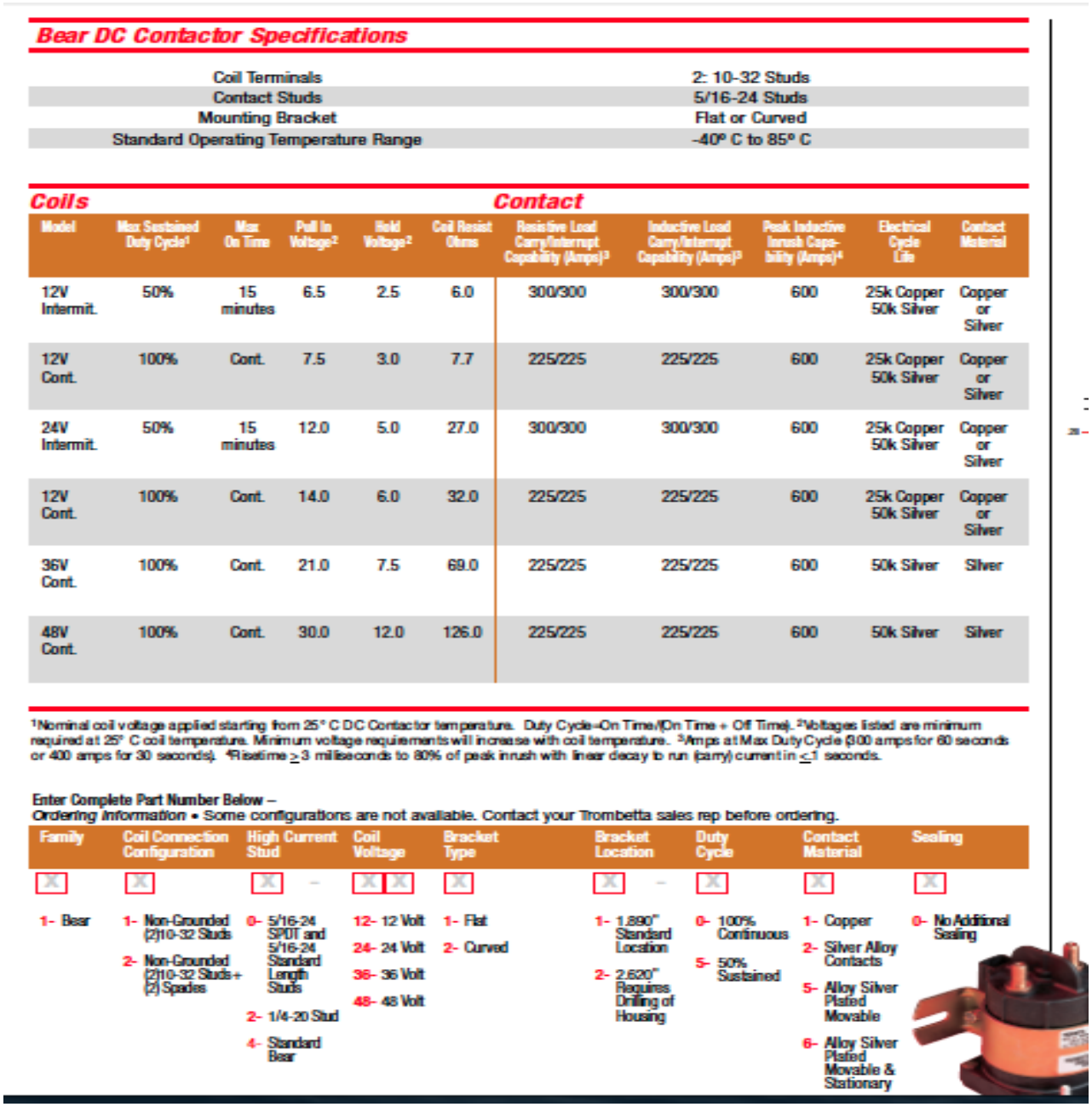
12VGCUTL :

SPECIFICATIONS

Product ID:	12VGCUTL
Cranking Amps:	155
Voltage:	12
Capacity:	6
Weight:	86
Width:	7.0625
Length:	13.12
Height:	11.38
CAP-20:	155
Plates:	99
ReserveCapacity 25:	292
ReserveCapacity 75:	77
WET/DRY:	W

Fig. 32: Battery Specifications

F.3 Solenoids 114-3611-020



Bear DC Contactor Specifications

Coil Terminals	2: 10-32 Studs
Contact Studs	5/16-24 Studs
Mounting Bracket	Flat or Curved
Standard Operating Temperature Range	-40° C to 85° C

Coils / **Contact**

Model	Max Sustained Duty Cycle[1]	Max On Time	Pull In Voltage[2]	Hold Voltage[2]	Coil Resist Ohms	Resistive Load Carry/Interrupt Capability (Amps)[3]	Inductive Load Carry/Interrupt Capability (Amps)[3]	Peak Inductive Inrush Capability (Amps)[4]	Electrical Cycle Life	Contact Material
12V Intermit.	50%	15 minutes	6.5	2.5	6.0	300/300	300/300	600	25k Copper 50k Silver	Copper or Silver
12V Cont.	100%	Cont.	7.5	3.0	7.7	225/225	225/225	600	25k Copper 50k Silver	Copper or Silver
24V Intermit.	50%	15 minutes	12.0	5.0	27.0	300/300	300/300	600	25k Copper 50k Silver	Copper or Silver
12V Cont.	100%	Cont.	14.0	6.0	32.0	225/225	225/225	600	25k Copper 50k Silver	Copper or Silver
36V Cont.	100%	Cont.	21.0	7.5	69.0	225/225	225/225	600	50k Silver	Silver
48V Cont.	100%	Cont.	30.0	12.0	126.0	225/225	225/225	600	50k Silver	Silver

[1]Nominal coil voltage applied starting from 25° C DC Contactor temperature. Duty Cycle=On Time/(On Time + Off Time). [2]Voltages listed are minimum required at 25° C coil temperature. Minimum voltage requirements will increase with coil temperature. [3]Amps at Max Duty Cycle (300 amps for 60 seconds or 400 amps for 30 seconds). [4]Risetime ≥ 3 milliseconds to 80% of peak inrush with linear decay to run (carry) current in ≤ 1 seconds.

Enter Complete Part Number Below –
Ordering Information • Some configurations are not available. Contact your Trombetta sales rep before ordering.

Family	Coil Connection Configuration	High Current Stud	Coil Voltage	Bracket Type	Bracket Location	Duty Cycle	Contact Material	Sealing
1- Bear	1- Non-Grounded (2)10-32 Studs 2- Non-Grounded (2)10-32 Studs + (2) Spades	0- 5/16-24 SPDT and 5/16-24 Standard Length Studs 2- 1/4-20 Stud 4- Standard Bear	12- 12 Volt 24- 24 Volt 36- 36 Volt 48- 48 Volt	1- Flat 2- Curved	1- 1.890" Standard Location 2- 2.620" Requires Drilling of Housing	0- 100% Continuous 5- 50% Sustained	1- Copper 2- Silver Alloy Contacts 5- Alloy Silver Plated Movable 6- Alloy Silver Plated Movable & Stationary	0- No Additional Sealing

Fig. 33: Solenoid Specifications

F.4 Upgraded Shocks

Hydraulic shock with 3″ compression. 730 lbs total load compression. 12″ eye to eye. 10mm (.39″) Hole ID.

F.5 Koyo Optical Encoders

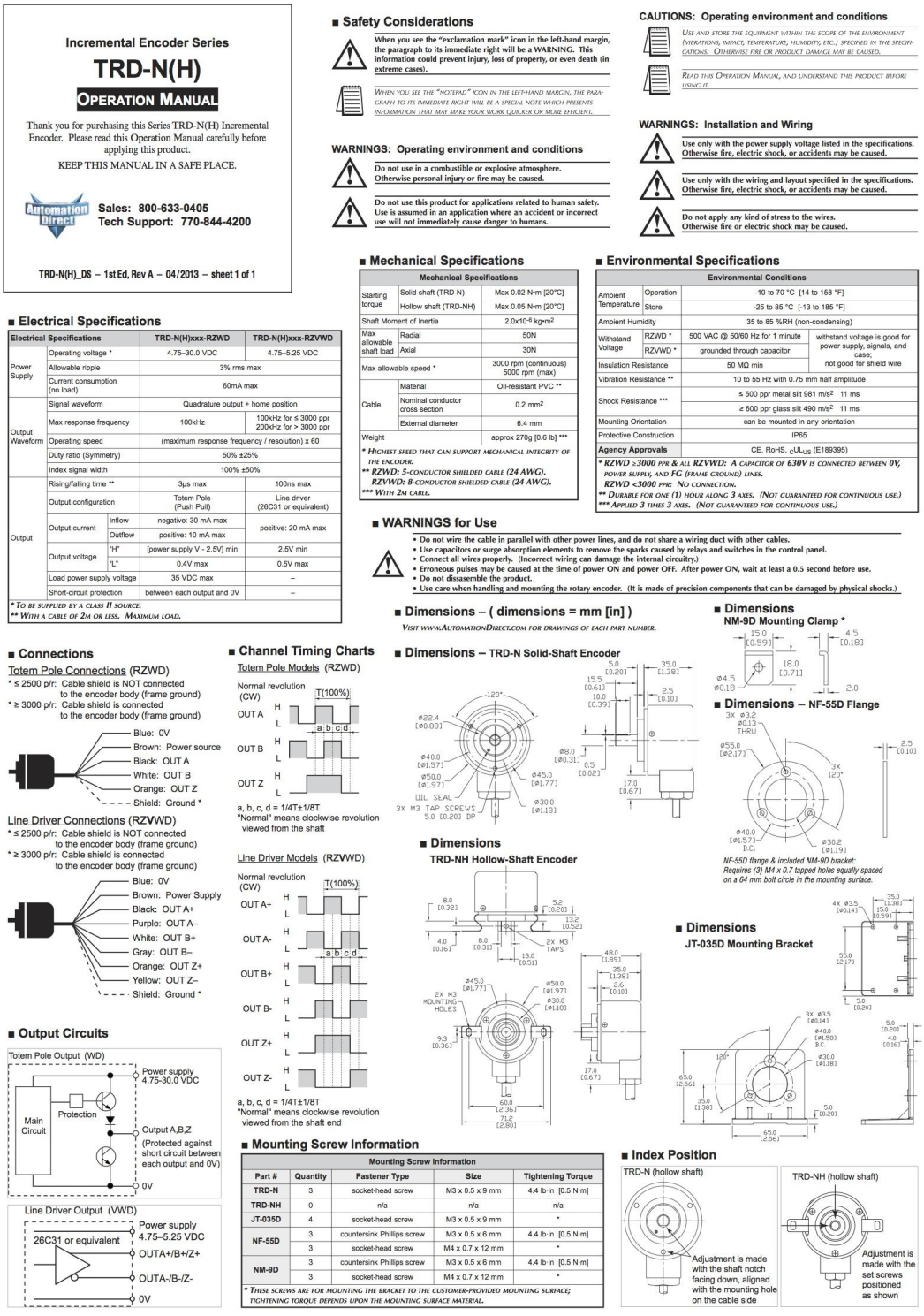


Fig. 34: Koyo Optical Encoders Specifications

F.6 VEX Motor Encoders

The optical shaft encoder is used to measure both relative position of and rotational distance traveled by a shaft. It works by shining light onto the edge of a disk outfitted with evenly spaced slits around the circumference. As the disk spins, light passes through the slits and is blocked by the opaque spaces between the slits. The encoder then detects how many slits have had light shine through, and in which direction the disk is spinning.

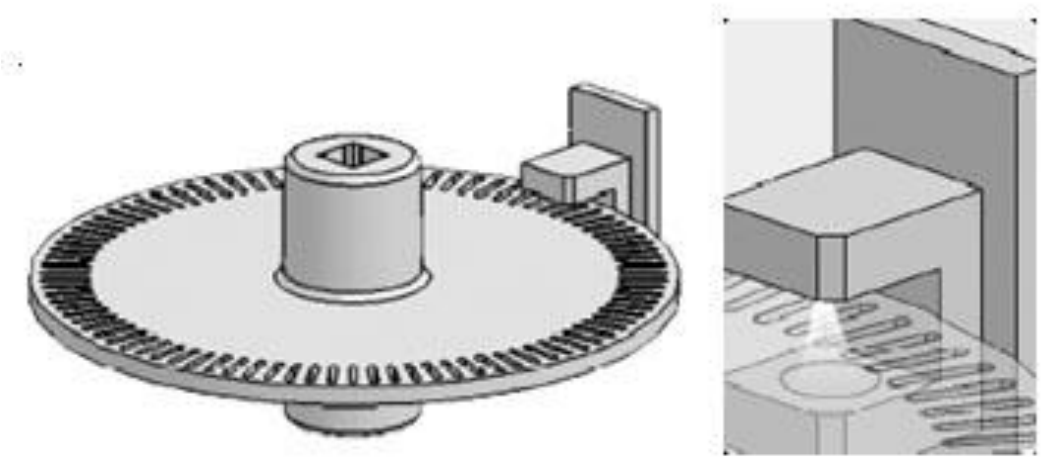


Fig. 35: VEX Motor Encoders Operation

Technical Information:

- Sensor Type: Infrared light sensor and infrared LED
- Resolution: 90 pulses per revolution
- Range: No limit, full 360° continuous rotation
- Size: 2 $\frac{5}{8}$in x 2in
- Weight: 0.08lb per sensor
- Shaft Size: $\frac{1}{8}$in square
- Wiring: Black - ground; Red - (+) power; White - control signal

ACKNOWLEDGMENTS

The authors would like to thank...

Carolina Cars and Clubs	Advanced Concepts	Chris Finlayson of Existential Motorcycles
Mountaintop Golf Cars	EZ-GO (Textron)	Joe and Ellen Reece
ProMatic Automation	Dave Erb	Skyland Automotive

All University of North Carolina Asheville and
North Carolina State University Faculty and Staff

REFERENCES

[1] L. Brown, *Improving Performance Through Torque Vectoring on an Electric All-Wheel Drive Formula SAE Race Car*. Diss. The University of Western Australia, 2013. Web.

[2] A. EmranHasan, M. M., M. Motamed Ektesabi, and A. Kapoor. *An Investigation into Differential Torque Based Strategies for Electronic Stability Control in an In-Wheel Electric Vehicle*.

[3] J. Reza N. *Vehicle Dynamics - Theory and Application*. Springer New York, 2014. 387-495. eBook. <http://link.springer.com/book/10.1007/978-1-4614-8544-5>.

[4] Z. Salam. *DC Motor Drive*. *Power Electronics and Drives*. Version 3. (2003): 1-34. Web. 9 Feb. 2014. <http://www.docstoc.com/docs/21935976/DC-Motor-Drive>.

[5] 2014 UNCA/NCSU Senior Design Team. *2wd Electronic Differential*. (2014). GitHub Repository. <https://github.com/Ultrashock/EGM484>.

[6] J. Reynold. *ANSI Standard Roller Chain*. Web. <http://www.renoldjeffrey.com/nmsruntime/saveasdialog.asp?lID=1006&sID=2750>.

[7] *Drive with PID Control on an Arduino Mega 2560*. MakerZone. 13 Sept. 2013. *Mathworks, Inc*. 16 Sept. 2013. Web. <http://makerzone.mathworks.com/uncategorized/drive-with-pid-control-on-an-arduino-mega-2560/>.