Final Report

ElecTrek

EE Senior Design

May 9, 2011

Katie Heinzen

Kathryn Lentini

Neal Venditto

Nicole Wehner

Table of Contents

1 Introduction

As renewable energy technologies continue to develop, new ways to generate electricity have been researched. The owner of Outpost Sports in Mishawaka would like to generate electricity from his bicycling training facility. The facility allows eight riders to bring in their bikes to train during the winter. The back wheel of the bike is held in a frame in contact with a braking system to provide resistance. During the 30-40 minute training session, the average power produced by each rider is 250 watts, with peaks of up to 400 watts. Currently, this is all lost to heat. This project is an opportunity to store and use energy that is already there. If all of the energy from the eight riders could be converted to electricity, it would be enough to power ten 100 W bulbs for an hour. ElecTrek's goal is to work with the current system to achieve this conversion by adding components that capture the energy as electricity and store it in a battery or use it during the session to power a device such as a fan or an iPod.

1.1 Problem Description and Proposed Solution

The customer for ElecTrek is the owner of Outpost Sports, J.V. Peacock. Mr. Peacock identified a way to improve his bicycle training facility and to save him money on electricity. The goal of this project is to route the mechanical energy generated by the riders into electrical energy that he can use elsewhere in the store. Of the eight set ups, only one will be altered for the project, but if successful, Mr. Peacock can implement the design on all eight systems. There is also the option of contacting the company that produces the systems, RacerMate, about adopting the design and improving it to market with their products.

The training facility uses a system called CompuTrainer. CompuTrainer controls the resistance seen by the rider. It allows individuals to program their own rides, via a controller on the handlebars, or to use a preprogrammed course that is controlled from an external computer. The software includes videos of international races that can be projected on a screen to enhance the riding experience. An Outpost store employee enters various parameters for the rider (weight, age, gender, etc) as well as the desired training course into the computer so the only control the rider will have is how fast he pedals during the 30-minute session.

The main goal of indoor cycling at Outpost Sports is for the rider to experience a realistic racecourse, and the CompuTrainer system provides this simulation. ElecTrek's design works with the system to keep the same training conditions. The solution is one that allows for variable resistance and one that does not make any modifications to the mechanics of the braking system or to the coding of RacerMate's CompuTrainer software.   See the high-level solution description below.

1.2 System Requirements

1.2.1 Overall System

The overall system should take the wasted mechanical energy of the stationary bike resistance system and turn it into usable electrical energy. The components added to the basic bike/CompuTrainer system should not interfere with the rider's training experience. Ideally, the rider would not be able to tell the difference between the original CompuTrainer-only system and the new ElecTrek system. The user interface with the ElecTrek system should not change from the existing user-handheld device or user-personal computer interfacing, i.e. the ElecTrek system should be non-invasive.

The size and weight of the ElecTrek system should be such that the additional components do not interfere with the rider's experience. Additionally, the components should be easily movable, so that the ElecTrek system can be implemented on any of the eight bike stands at Outpost Sports. As for safety, tripping hazards may exist due to all the wires involved in the project. All exposed wires should be insulated and taped to the floor. Other potential tripping hazards or sharp corners will be flagged to alert users to the danger. Due to the inherent dangers of lead-acid batteries, the battery voltage will be carefully regulated to avoid overcharging, which can lead to battery leakage or, in rare cases, explosions.

The power for the ElecTrek system will be provided by the battery. Depending on the user's preference, or if the battery voltage drops to its threshold level, the microcontroller can also be run off a 120 VAC wall outlet connection. The final overall system requirement is that the entire project cost must fall within the allotted $500 budget.

1.2.2 Subsystem and Interface Requirements

The CompuTrainer subsystem is a pre-existing component obtained from Outpost Sports owner J.V. Peacock. For the purposes of this project, the CompuTrainer is treated as a "black box." Other than removing the flywheel from the shaft and non-permanently coupling the shaft to the alternator's shaft, the CompuTrainer was untouched. The electromagnetic brakes provide resistance to the rider based on the signals received from the software either in an attached computer or, as with this project, from the handheld device. The CompuTrainer system includes a PC software package.

The alternator generates a peak output of roughly 14.4 VDC. The alternator increases the mechanical resistance presented to the rider when it is forced to provide electrical power, meaning the microcontroller must compensate by adjusting how much resistance the CompuTrainer provides. The alternator must be connected to the CompuTrainer shaft by a shaft coupler, which must be able to unite two shafts of different sizes, after the CompuTrainer's flywheel is removed. Ideally, this connection will be as smooth as possible. Ultimately, the alternator should rotate freely with the CompuTrainer. The alternator must vary its RPMs based on the resistance signal output by the microcontroller. The alternator must be able to handle at least 1000-3000 RPMs and peaks of 400 W, with occasional spikes up to 1 kW. The charging circuit for the battery and a built-in voltage regulator for a 12 Volt, lead-acid battery should be included in the alternator, as should a fan to provide necessary cooling. The alternator subsystem must include current sensors (obtained from DigiKey) and improvised voltage sensors at its output which report their measurements to the microcontroller to determine the power generated by the alternator.

Another critical component of this subsystem interface is the automatic disconnect for the battery. Once the sensors determine the battery is full, an automatic disconnect needs to prevent current from flowing into the battery so overcharging does not occur. The lead-acid battery itself operates at 12 VDC and has two modes in this ElecTrek setup. The battery is either charging or powering a load through the inverter. The battery must be able to handle several amps of direct current, deep cycling and intermittent charging.

A moisture-resistant plastic enclosure must secure the surface-mount electronic boards to protective the sensitive components from exposure to water or mechanical damage, and to protect users from potentially harmful currents.

The inverter attached to the battery converts 12 Volt DC voltage into 120 VAC to power a load. The load can vary, e.g. a fan, lamp, cell phone with charger, etc, as long as it can connect through a standard two-prong socket.

The LCD screen is displayed on the cover of the plastic enclosure and operates on a 5 VDC power supply and must be compatible with an SPI interface. The screen must have enough display space to show the instantaneous power, maximum power, speed, battery charge level, and operating mode.

The microcontroller software is a critical interfacing component. It runs multiple functions and interfaces, e.g. intercepting signals from the CompuTrainer software; sending signals to the LCD, CompuTrainer, and alternator; and polling the various voltage and current sensors in the system. It must have an analog-to-digital converter pin port so the analog values reported by the voltage and current sensors can be digitally displayed by the LCD. The microcontroller interfaces with the CompuTrainer software by intercepting the signals from the handheld device, determining required resistance and distributing this information to the CompuTrainer resistance device and the alternator.

*The LCD now operates with an SPI interface. Originally it was slated to be an I^2C interface.

*Instead of needing two ADCs on the microcontroller, the microcontroller came with 12 A-to-D pins, of which 3 were used.

1.3 High Level Description of Solution

The design uses an alternator to convert the mechanical energy of the braking system to electrical energy. This electrical energy is then either used to power a device such as a fan during the training session through an inverter, or is used to store in a battery as chemical energy for later use. A major goal was to make the system as non-invasive as possible to prevent voiding the warranty on the CompuTrainer system and altering any of the commercial software. The alternator is connected to the rotating shaft on the eddy current brake. We use an alternator because it operates at a speed compatible with the bike (roughly 40-60 rpms on the bike wheel, which corresponds to 1000-1500 rpms on the alternator shaft) and outputs at least 12 VDC.

The DC voltage is fed into a battery and is converted to AC voltage with an inverter, so that external devices are powered during the training session. Sensors monitor the current out of the alternator and the voltage across the battery terminals. This battery information is used to determine the power output of the alternator and the charge level of the battery, which is displayed on the LCD screen.

A microcontroller controls the LCD. It also interprets the information from the sensors to compute power production and battery charge level. The microcontroller dynamically controls the resistance alternator applies to the bike based on the resistance signal coming from the CompuTrainer, while the CompuTrainer eddy current brake just rotates freely (with the alternator shaft). When the battery is fully charged, the microcontroller shuts of current to the alternator field coil and generates resistance with only the eddy current brakes, while the alternator shaft spins freely.

1.4 Design vs. Expectations

The design met its high-level expectations. We were able to output electrical power through the CompuTrainer/alternator resistance system to charge the 12V battery, which in turn powered the inverter so external devices (e.g. a lamp) could be operated from our system. In addition, as we changed the resistance on the handheld CompuTrainer device, the resistance presented to the rider by the alternator varied.

More specific expectations, however, were not met. One stipulation for this project was that the rider’s training experience would not be affected by the implementation of our system. Unfortunately, the alternator was not an ideal power generation device. It does not respond quickly enough to current changes on the field coil to be able to smoothly adjust resistance levels. Furthermore, the alternator provided a large level of base resistance even when no power was being generated (i.e. the alternator output was disconnected from the battery). At high levels of resistance selected on the CompuTrainer, the bike was almost impossible to pedal.

The efficiency of the alternator was also poor. Modern alternators only achieve 50-60% efficiency at best, and ours is several decades old. Coupled with the mechanical losses of our system, such as an imperfectly aligned, welded shaft coupler between the alternator and eddy current brake, and the fact that we were operating the alternator outside of its optimal operating range of 2000-3000 rpms, our efficiency was quite a bit smaller. These inefficiencies limited electrical power generation by our system to 20-40 Watts.

2 Detailed Project Description

2.1 System Theory of Operation

The new addition of ElecTrek's system interfaces with the existing software and resistance system. It does not alter how the bike is anchored down. The direct contact between the wheel and the eddy current brake stays the same. The alternator is attached to the braking system on the rotating shaft on the CompuTrainer. All other interfaces in the system are electrical connections. Figure 1 shows the contacts between the bike, CompuTrainer, and alternator.

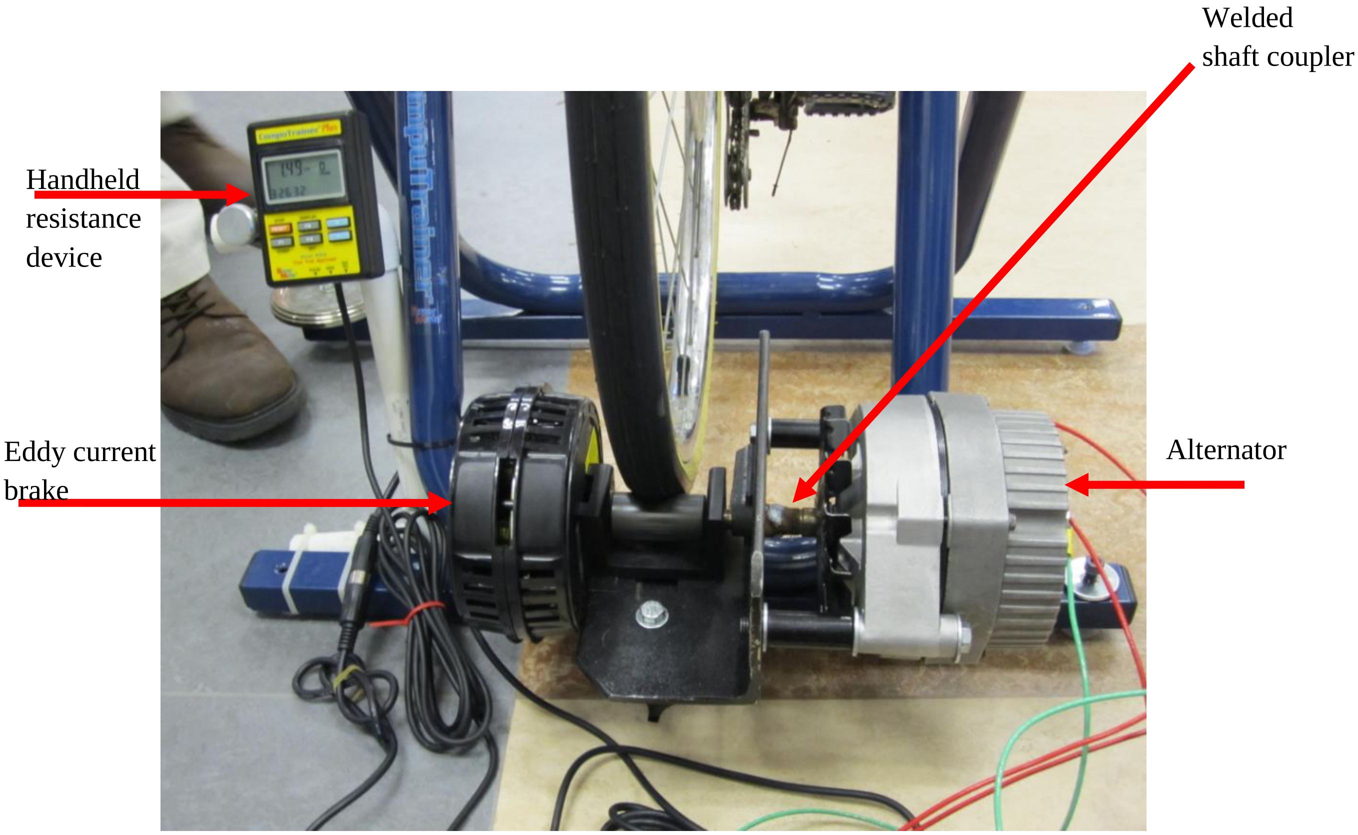

Figure 1. The interface between bike tire and the CompuTrainer system.

The PIC18F4620 microcontroller is the heart of our design. The microcontroller intercepts the signals from the CompuTrainer software that indicate how much resistance to apply to the bike wheel. The microcontroller interfaces with the handheld resistance device and the eddy current brake through DIN-8 connectors. The microcontroller directs the LCD to display power information, gathered by voltage and current sensors, through an SPI interface.

The auxiliary board is the interface between the alternator and the battery, and between the battery and the inverter. The information from the current sensor is communicated to the microcontroller via a wired connection. There are two voltage-sensing circuits, one located at the alternator output, and the other located after the current-limiting power MOSFET between the alternator output and positive battery terminal. The microcontroller disconnects the battery from the alternator when the battery is full. When use of the inverter depletes the battery again, the microcontroller switches the alternator subsystem back on. The complete system, with interfaces, is illustrated in Figure 2.

2.2 System Block Diagram

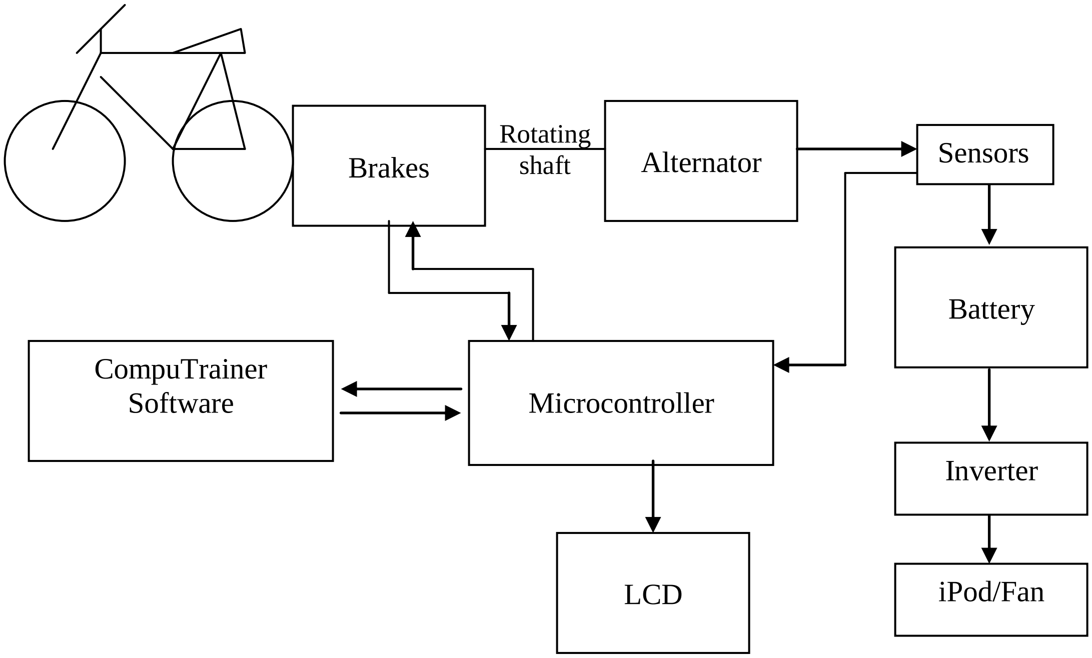

Figure 2. System block diagram

2.3 Detailed Operation of Subsystems

2.3.1 Alternator Subsystem

The alternator being used is a Delco-Remy alternator, part number V119 (no longer being manufactured). The shaft of the alternator connects through a coupler to the shaft of the eddy current brake. (This required removing the flywheel from the brake.) The shaft coupler connects the smooth 0.75 inch CompuTrainer shaft to the threaded 0.5 inch alternator shaft. In this case, the coupler is made from metal, and one end covers the shaft of the eddy current brake, and two set screws hold it in place. Meanwhile, the other half of the rod is threaded. It fits inside the threaded shaft of the alternator, and the two sets of threading lock in place. This connection is also welded in place. This means the alternator shaft rotates with the CompuTrainer at the same rpm. As the alternator rotates, it generates electricity, which is fed through the alternator's built-in voltage rectifier (converts AC voltage to DC) and charging circuit to the 12V, deep-cycle, lead acid battery. This is a straightforward connection very similar to the electrical setup in a car. The primary difference is that the setup for this system with the eddy current brake will not use an ignition switch, the Sensing #2 terminal, or the Test Hole (Figure 3) connections. Instead, Bat terminal (Figure 3), which is the output terminal of the alternator, connects to a screw terminal on the auxiliary board and feeds into the current-limiting MOSFET. Field #1 terminal is connected to the positive terminal of the battery. This connection uses the battery to energize the field coils of the alternator.

The current into the field coil is controlled by the microcontroller through another MOSFET and provided by the battery. When the MOSFET is turned on, current flows from the battery and energizes the field coils, providing the alternator's magnetic field. In order to generate electricity, the alternator's field coils must be energized. When power is provided to the field coil the alternator provides a significant resistance, even if no electrical power is being produced. The resistance provided to the user is thus manipulated by pulsing the field coil on and off through a PWM signal to the controlling MOSFET. This pulsing signal is the same one received from the handheld resistance device.

This setup will use the ground and battery terminals on the alternator (Figure 3). The battery output terminal on the alternator connects to the auxiliary board, which eventually feeds into the positive battery terminal. The metal casing of the alternator acts as ground.

The testing plan for the alternator consisted of connecting a 1 horsepower motor to the pulley of the Delco-Remy alternator by means of a belt. The alternator was also wired to the battery. A tachometer was used to ensure the motor was spinning at least 1000 rpm. Once the alternator was running at the same rpm as the motor (determined again by the tachometer), the voltage across the battery terminals and the current going into the battery were measured with a digital multimeter (DMM) in order to get baseline readings. Readings were taken at a variety of rpms. In addition, an inverter was attached to the battery so that loads with large power draws (e.g. a soldering iron) could be connected to test their effect on alternator operation and fluctuations in voltage and current in the alternator/battery setup.

For example, a test conducted on February 27 demonstrated that the alternator – driven by a motor/pulley configuration – was capable of charging the 12V battery. The alternator spun at roughly 1030 rpms, outputting a current of 2.2 Amps and a voltage of 12.14 Volts. The voltage 14.14 Volts can only be achieved at higher rpms (in the 1500-3000 range), which it was determined a bike rider can provide to the eddy current brake and the alternator. The testing motor's maximum speed was just under 1100 rpm. The critical aspect of this test was that, at these voltage and current output values, the alternator was able to charge the battery from 11.46 to 11.89 V.

Despite extensive searching, we were unable to find a dynamometer to measure the torque the motor needed to provide to the alternator. Had we done so, we would have realized that the alternator required more torque to maintain a constant power output than was feasible with a human bike rider.

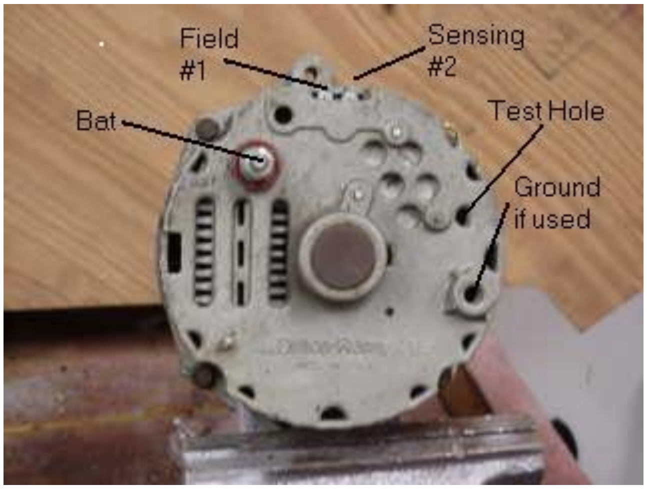

Figure 3. Alternator electrical pin diagram.

2.3.2 Battery/Inverter Subsystem

The main hardware components of this subsystem are the 12V deep-cycle, lead acid battery and the inverter. This subsystem also includes the MOSFET control circuit and the current sensor on an auxiliary printed circuit board. The two main signals of this subsystem between the battery and inverter are positive and ground. The battery and the inverter are connected in parallel in the final system. See Figure 4 for a block diagram of this subsystem.

The inverter that matches our specifications is the Cobra CPI-480. The inverter is rated for 12VDC input and 120VAC output. It is also rated for 800W for power surges and 400W for continuous operation. The inverter has two types of output connectors: two standard three-prong plugs and a USB port. Not much testing beyond basic functionality checks was required on the inverter. That is, we hooked up the inverter to a power source and plugged a soldering iron into one of its output ports. When the soldering iron heated up, we knew the inverter was functioning properly. The inverter essentially operates as a black box in this system: it takes input of 12VDC from the alternator and outputs a constant 120VAC.

The battery that matches our specifications is the Sportsman Marine Battery DC27. This 12V deep-cycle, lead acid battery is rated for 75 Amp-hours. We monitor the battery status by its terminal voltage. Table 1 can be used to equate terminal voltage with battery charge. If the battery is fully charged, the alternator's output is disconnected so no more current flows into the battery, and the eddy current brake takes over providing resistance to the rider. However, if the battery is discharged by use of the inverter, the microcontroller switches the alternator output on again, and the alternator resumes providing resistance to the rider so that the battery can charge.

Table 1. Relation between Terminal Voltage and Battery Charge.

State of Charge	12 Volt battery	Volts per Cell
100%	12.7	2.12
90%	12.5	2.08
80%	12.42	2.07
70%	12.32	2.05
60%	12.20	2.03
50%	12.06	2.01
40%	11.9	1.98
30%	11.75	1.96
20%	11.58	1.93
10%	**11.31**	**1.89**
0	**10.5**	**1.75**

Source: *http://www.solarnavigator.net/battery_charging.htm*

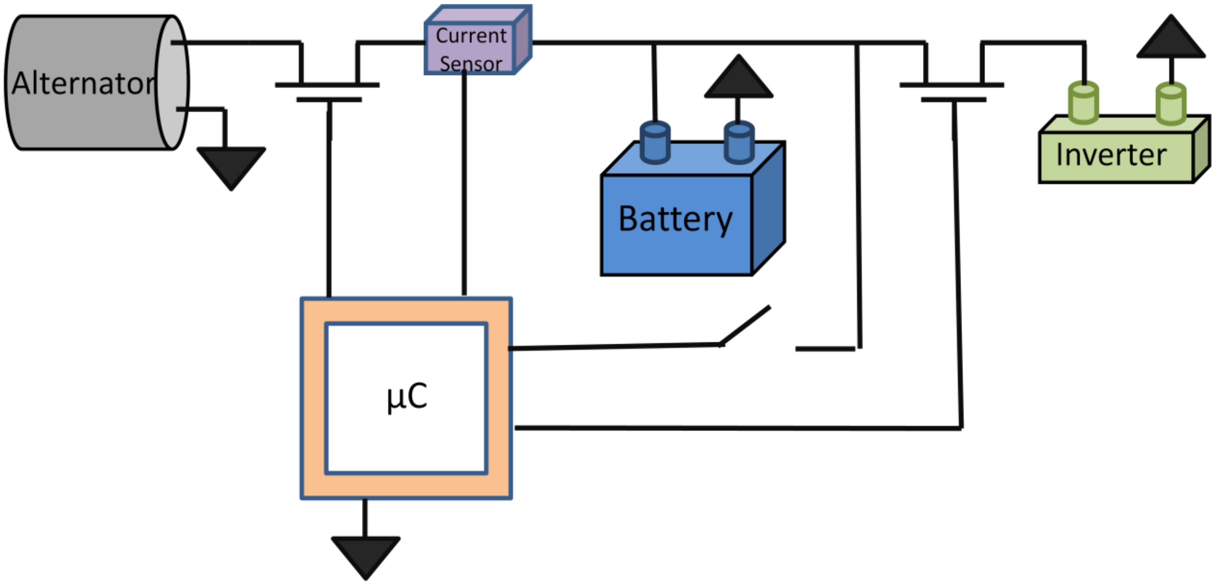

Figure 4. Subsystem Block Diagram.

The auxiliary board supports connections between the alternator, the battery, and the inverter. Three MOSFETs are on the auxiliary PCB and are connected to the microcontroller on the main board. The auxiliary board also contains a 20A-rated current sensor, which allows the microcontroller to monitor the current feeding into the battery. Ribbon cables will be used for the signals from the auxiliary board to the microcontroller.

The MOSFET that matches our specifications is the Infineon Technologies BSZ067N06LS3G. This part is available from DigiKey under the part number BSZ067N06LS3GINCT-ND. It is rated for a maximum drain to source voltage of 60V and a maximum power of 69W.

In order to measure voltages, we employ a voltage divider circuit at the alternator output terminal and at the battery terminals, since the microcontroller can only handle inputs of 5 V. Using resistor values of 120kΩ and 360kΩ, this circuit divides the incoming voltage by 4, so that it is at 5V or below. A unity gain buffer isolates this circuit from the microcontroller, so the current into the microcontroller is limited.

2.3.3 Microcontroller and Software

The microcontroller we used is the PIC18F4620. It was chosen because of its Pulse Width Modulation (PWM) modules which can be used for convenient reading and generation of the PWM signals necessary for controlling the CompuTrainer's resistance. The first PWM module is used to read the resistance signal from the CompuTrainer software, and the second module is used to generate a PWM signal, when needed, to the eddy current brake. The microcontroller also has a 10-bit analog-to-digital converter with up to 12 input pins. Pin A0 is the alternator voltage reading, pin E2 is the battery voltage reading, and pin E0 is the current sensor reading. The voltage of the current sensor is converted to current via the formula $I = (V_{sen}-2.5)/16$;

The LCD screen is connected through an SPI interface on pins C3 (serial clock), C4 (data in), C5 (data out), and A5 (slave select). The built in SPI mode of the Master Synchronous Serial Port (MSSP) Module simplified the programming needed to control the LCD. A library of functions for simple LCD control was written in files *spiLib.h* and *spiLib.h*. Timer1 was used to control the frequency of LCD updates.

Memory was not a major issue with the microcontroller as our application was not very memory-intensive. The microcontroller and board runs on a 5 V supply from the voltage regulator, which can be powered through either a wall outlet or the 12V battery. The microcontroller is shown in Figure 5.

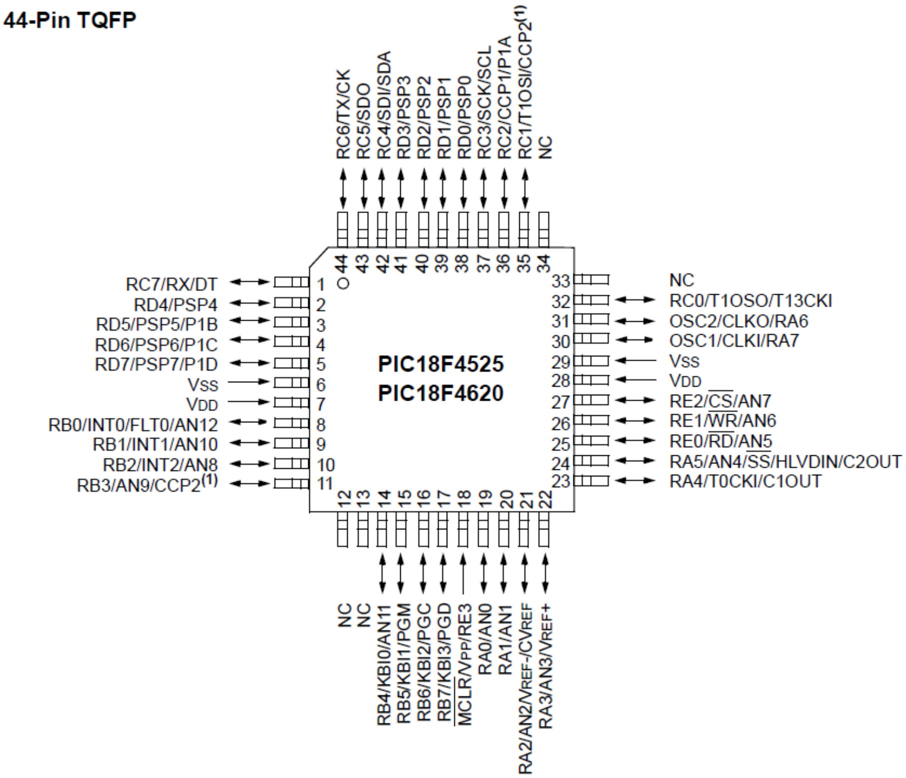

Figure 5. Microcontroller schematic.

The microcontroller takes regular readings of the voltage and current produced by the alternator and calculated the electric power being generated. This information is displayed on the LCD along with the peak power generated since the system has been on. Additionally, the voltage readings of the battery are used to determine its charge level. The charge level is displayed on the LCD via a battery icon and monitored to ensure that the battery is neither overcharged nor depleted. If the battery is being overcharged, the microcontroller will electrically disconnect the battery from the alternator via pin A2, which connects to the gate of the current limiting MOSFET. In this mode of operation, the microcontroller then sends a PWM signal to the eddy current brake to generate the needed resistance.

The RPM and resistance signals from the CompuTrainer are constantly read. The resistance signal is a PWM signal at 3.58 kHz that is normally high (5V). The speed signal is also normally high with occasional low pulses. The time between pulses is related to the speed of the CompuTrainer shaft. The RPM of the bike wheel is then calculated via the formula: $\mathrm{RPM}_{Bike} = \frac{39062}{16(t_{pulse})}$ Where t_{pulse} is the time between pulses.

The software was initially written on and tested on the kit board, which has a microcontroller of the same PIC family with the same hardware modules we required. Signal reading and generation were tested through the use of a signal generator and oscilloscope. LCD software could not be written and tested until our main board had been printed, however, due to conflicting pin use on the kit board. Sensor readings were verified by comparing microcontroller values to DMM readings made at the sensors. The RPM signal reading was tested by comparing our microcontroller's value to the CompuTrainer software's value.

2.3.4 Main Board Design

The board is powered by the battery or the wall socket and is regulated to 5 V by the LT1129CS8-5#PBF-ND voltage regulator made by Linear Technology. It takes an input voltage of up to 20 V and outputs a fixed 5 Volts at 500 mA. The voltage regulator is shown in Figure 6. A switch selects whether the power supply for the board comes from the wall socket or the battery. This can be used for testing, and to power the system if the battery charge level drops too low. The programmer, illustrated in Figure 6 (created by Mike Schafer), has a reset button for any issues that arise during use.

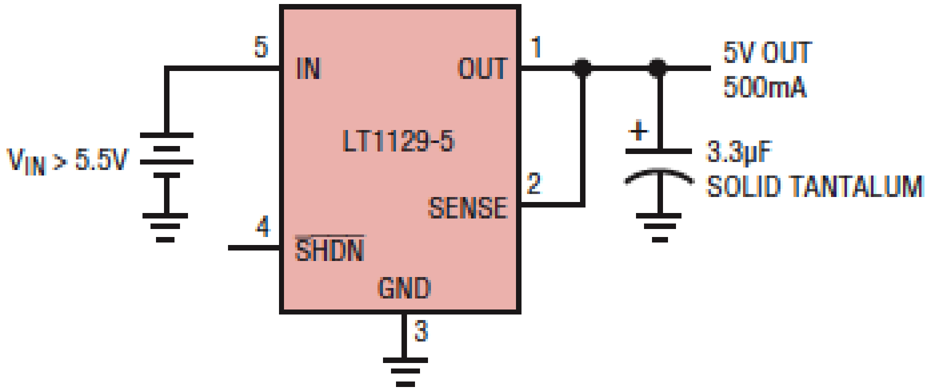

Figure 6. Voltage regulator schematic.

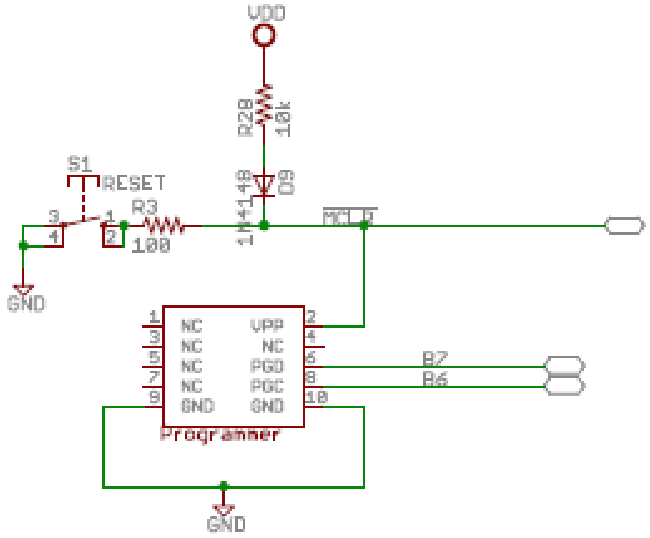

Figure 7. Programmer Schematics.

The main board also includes connections to the auxiliary board, the CompuTrainer, and power switches. Two female din8 connectors are used to take the input of the CompuTrainer software and output the resistance to the brake. A Molex connector is used to connect between the two power switches and the board. There is one switch to choose how the board is powered, either from the battery or the wall and one switch is used to turn our system on or off. Standard pin headers and a ribbon cable are used to connect the programmer and the auxiliary board. A unity gain buffer was added between the drive input signal from the CompuTrainer software and the microcontroller input pins.

The testing of the main board included testing of each part as it was soldered on and detailed voltage readings of each pin and connection at various vias.

The complete board schematic is included in Appendix 7.1.

2.3.5 Auxiliary Board Design

The auxiliary board was designed to keep all the high currents and voltages on a separate, isolated board away from the microcontroller. The auxiliary board has wide traces for high-current signals. Examples of wide traces are the traces from the sources and drains of the MOSFETs. In general, all other traces and vias are a standard size.

The auxiliary board is connected to the main board using a ribbon cable. We used an 8 pin port on the auxiliary board to attach the ribbon cable. See Table 2 for a listing of the board connections.

Table 2. Table of connections between main and auxiliary board.

Port Pin	Signal
1	Voltage measurement at alternator output
2	Voltage measurement at battery terminals
3	Reference voltage from µC to limit current into gate of first MOSFET
4	On/Off control signal to driver to second MOSFET
5	NC
6	Analog voltage output from current sensor to µC
7	GND
8	V_{dd} = 5V

The main components of the auxiliary board are the 2 MOSFETs, a DC-to-DC converter, a quad op amp chip, a current sensor, and a MOSFET driver. The DC to DC converter, which can output up to 32V, outputs a voltage of 21V. This is used as the supply voltage for the quad op amp chip. We needed a large supply voltage because op amps require a supply voltage that is higher than the voltages on its terminals. The largest alternator voltage we saw was 15.5V, and the maximum battery terminal voltage is 12.7V. Three of the op amps on the quad chip are used as unity gain buffers throughout the board. The remaining op amp is used to amplify the reference voltage from the microcontroller to the gate of the first MOSFET.

To control the gate voltage of the first MOSFET between the alternator and battery, we use the reference voltage feature of the microcontroller. We output 3.7V from the microcontroller, which goes through an op amp circuit on the auxiliary board. The gain of this op amp is 5.3 in order for the gate to be at a high enough voltage to turn the FET on.

We originally designed our auxiliary board with a MOSFET driver to control the gate voltage of the second FET between the battery and the inverter. The driver takes a 5V input from the microcontroller and outputs a large enough voltage to control the gate of the FET. We made a mistake in our auxiliary board design, and pins 5-8 were connected backwards on our board. We cut those traces and rerouted them by hand on our board. Then, we started to troubleshoot the driver circuit because it was not working properly. After rerouting the traces, however, we realized we had a simpler option available to us. The output of the DC to DC converter was a large enough voltage to control the gate of the second MOSFET. We originally used the DC to DC converter to provide a supply voltage to our quad op amp chip. In our final board, we added a wire between the output of the DC to DC converter to the gate of the second MOSFET and did not use the MOSFET driver circuit at all.

We used the current sensor to measure the current coming out of the alternator and going into the battery terminals. It was important to keep track of this current in order to control the charging conditions of the 12V lead acid battery. The current sensor outputted an analog signal, which went to an A-to-D pin on the microcontroller via the ribbon cable. The microcontroller converted the analog signal into a digital one, and translated the digital value into a usable current value in Amps.

We used many unity gain buffers in the auxiliary board. We used one for the current sensor, one for the voltage measurement circuit on the battery terminals, and one for the voltage measurement circuit on the alternator output voltage. The unity gain buffers helped us to isolate the current and voltage measurements from the microcontroller. We wanted to ensure that the microcontroller didn’t draw any current from those connections and we wanted to protect the microcontroller from any over-voltage or high current mistakes.

2.4 Interfaces and Sensors

The battery and the inverter are connected by wires. The terminals of both the battery and inverter have a screw and nut. A wire tab can be placed over the screw and the nut can be used to tighten the wire against the screw. This provides a secure mechanical and electrical connection. Precautions have been taken to prevent someone from touching exposed terminals. Plastic caps cover the battery terminals, all of the board connections are covered by the plastic enclosure, and a rubber DIN-8 cable terminal sheath fits over the exposed alternator output terminal. The input terminals of the battery are connected to the auxiliary board battery screw terminals. The inverter is wired directly to the inverter screw terminals on the auxiliary board. The output interfaces of the inverter are standard. There are two standard 3-prong 120 VAC plugs and one 5V USB port. These can be easily used to connect external devices, such as fans or iPods.

The power MOSFET circuit has more options for testing. The output current and voltage from the alternator should be tested to make sure that they are within the ratings of the MOSFETs and the other components on the board. The voltage reading and the current sensor output can be verified using a multimeter.

Another important situation to test occurs when the rider experience low-resistance courses when the battery voltage is low. For example, when a rider is coasting down a steep hill, there is little to no resistance felt by the rider when he is pedaling. When the battery voltage is low, the current coming from the alternator into the battery will be high. This means that the resistance provided by the alternator will also be high. In this situation, the resistance provided by the alternator is larger than the resistance requested by the CompuTrainer software. Therefore, it is necessary to have control over the amount of current flowing into the battery. This will ensure the correct resistance is seen by the rider and will also help protect the battery from dangerous charging conditions. The microcontroller should recognize this situation and direct the first MOSFET to limit the current going into the battery. The auxiliary board will reduce the current going into the battery to 1-2 Amps. This will reduce the resistance of the alternator and, in turn, the resistance seen by the rider. The current going into the battery will only be limited when the battery voltage is lower than a certain threshold, or when the current going into the battery is too high. In other situations, the charging circuit built into the alternator will adequately control the current going into the battery. This behavior can be tested by connecting a discharged battery, selecting a low-resistance riding course, monitoring the microcontroller software signals, and verifying the expected behavior of the MOSFET.

3 System Integration Testing

3.1 Integrated Subsystem Tests

The first two subsystems we connected were the CompuTrainer eddy current brake and the alternator. We tested this subsystem by pedaling the bike and ensuring that the eddy current brake and alternator shafts spun together. Mr. Dee Davis was responsible for connecting the two shafts. The initial method of connecting the two – by having one end of the Allen stock rod cover the eddy current brake shaft and be screwed down, and having the other end inserted into the alternator shaft – quickly broke down. The end of the rod did not extend far enough into the alternator shaft, so it wore down the metal holding it in place until the two shafts were able to spin independently of each other, meaning no power could be produced. The solution was to weld the rod in place at the alternator connection. Two 10-32 set screws are used to secure the Allen rod to the eddy current brake shaft, which allows the two systems to be separated (a key element of the non-invasiveness of the design). Once this was fixed, the shafts did spin together. However, the shafts had significant vibration due to inexact alignment. Attaining exact shaft alignment is very difficult to attain without special machines.

The main board and auxiliary board were connected through a ribbon cable. Throughout the course of this testing, we discovered a problem and corrected it. Pin A1 on the microcontroller was drawing an abnormal amount of current when connected. This caused voltage levels on the auxiliary board to drop. This was rectified by instead connecting the battery voltage sensor to pin E2 on the microcontroller.

Then we started testing the interface between the main board and the CompuTrainer. We noticed that the microcontroller was drawing too much current and this was causing voltages to drop. We inserted a unity gain buffer between the CompuTrainer resistance pin and the microcontroller. This isolated the signal and we observed the correct voltages. When we enabled Timer1, we were unable to use our original PWM output pin. We corrected this problem by switching to a new PWM output pin, D7.

We also tested the interface between the auxiliary board and the alternator. We discovered that the field coil of the alternator needed to be connected to the battery in order to energize the coils and start the alternator. We made this connection to the positive terminal of the battery and solved this problem.

We did not have any problems with connecting the inverter or battery to the auxiliary board.

3.2 System Meets Design Requirements

One of our main design requirements was that the rider should experience a range of resistances when riding the bike. Our system meets this requirement by varying the resistance seen by the rider as the resistance value of the handheld device changes. As discussed before, the range of resistances seen by the rider is not optimal. This is due to using an alternator as the generating device. The high base resistance provided by the alternator was not known until the system integration tests. This behavior was not apparent in our subsystem tests because the DC drive we used was able to provide more torque to spin the alternator than is possible for a human to provide while riding. In this way, it does not meet design specifications because the rider is aware of the extra resistance when our system is connected. However, we were still able to demonstrate a variable range of resistances on demonstration day.

Another design requirement was to generate power from our system. We fulfilled this requirement because the current output from the alternator effectively charged the battery. The inverter was able to run off the battery, and we could power devices using the 12V output from the inverter.

Another major design requirement was that our system maintains the warranty on the CompuTrainer system. This was achieved because our system is able to be fully disassembled from the Eddy current brake. The shaft coupler, which is welded to the alternator, is attached to the CompuTrainer using two set screws. These screws can be easily unscrewed in order to have the original CompuTrainer system back to its original state.

4 User’s Manual

Note: This user’s manual assumes the user is familiar with the RacerMate CompuTrainer stationary bike system, including how to install it.

Safety Considerations

- Make sure both terminals on the 12V battery are not exposed at the same time (keep rubber tips at least one at all times).

- Never touch both terminals of the battery at the same time.

Never touch the alternator output screw, unless nothing else is connected (including the connection to the ground screw)

- Before adjusting any of the wires or connections, make sure all switches are off and all wall plugs are unplugged

- Ensure that all connection cables that run on the floor are taped down with to reduce tripping hazards.

4.1 Installation

The addition of ElecTrek's system will require about two to three feet behind the bike and braking system. ElecTrek's system works in conjunction with the CompuTrainer system. Power the CompuTrainer as normal with the power cord connected to the braking system.

If the alternator is not already connected, insert the shaft of the CT into the shaft of the alternator, then align the two screw terminals of the alternator to the holes on the stand, the spacers, and washers. Screw together using the two screws and washers included. Be sure to connect the smaller side with the smaller screw to the hole that is closer to the bike stand to ensure clearance for adjustments of the bike wheel. See Figure 8.

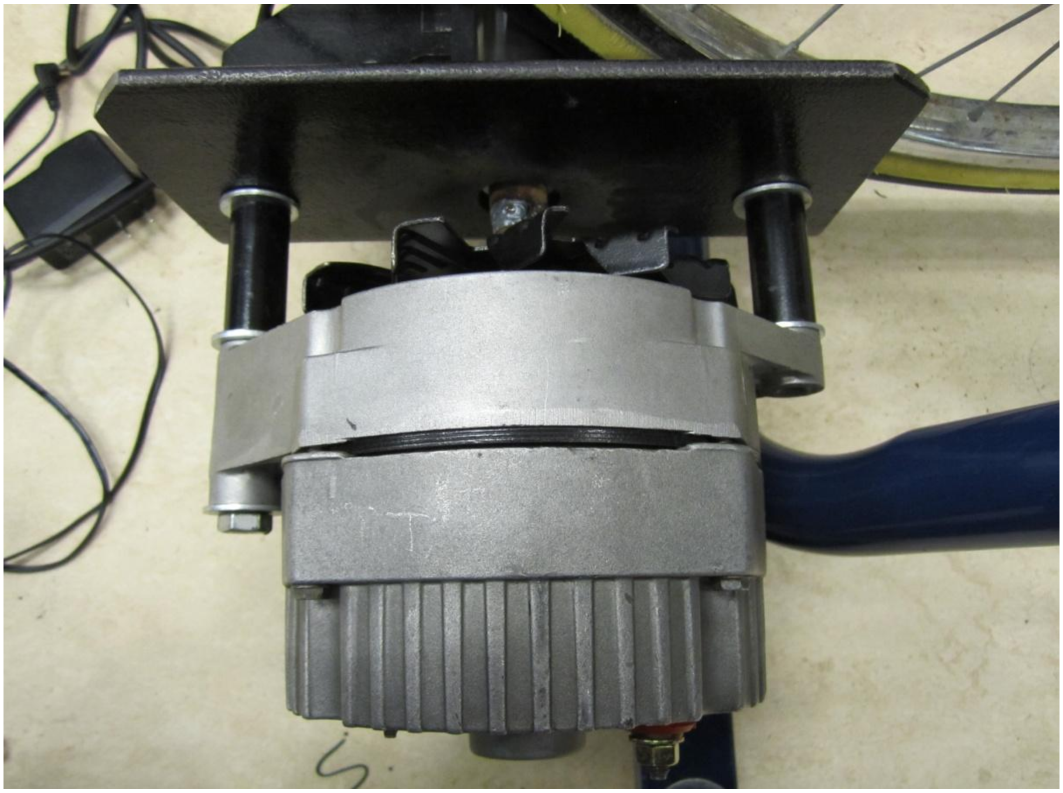

Figure 8. Alternator and CompuTrainer connection.

See Figures 9, 10, and 11 and refer to Table 3 for the electrical connections. All of the necessary electronic components are contained within the plastic enclosure.

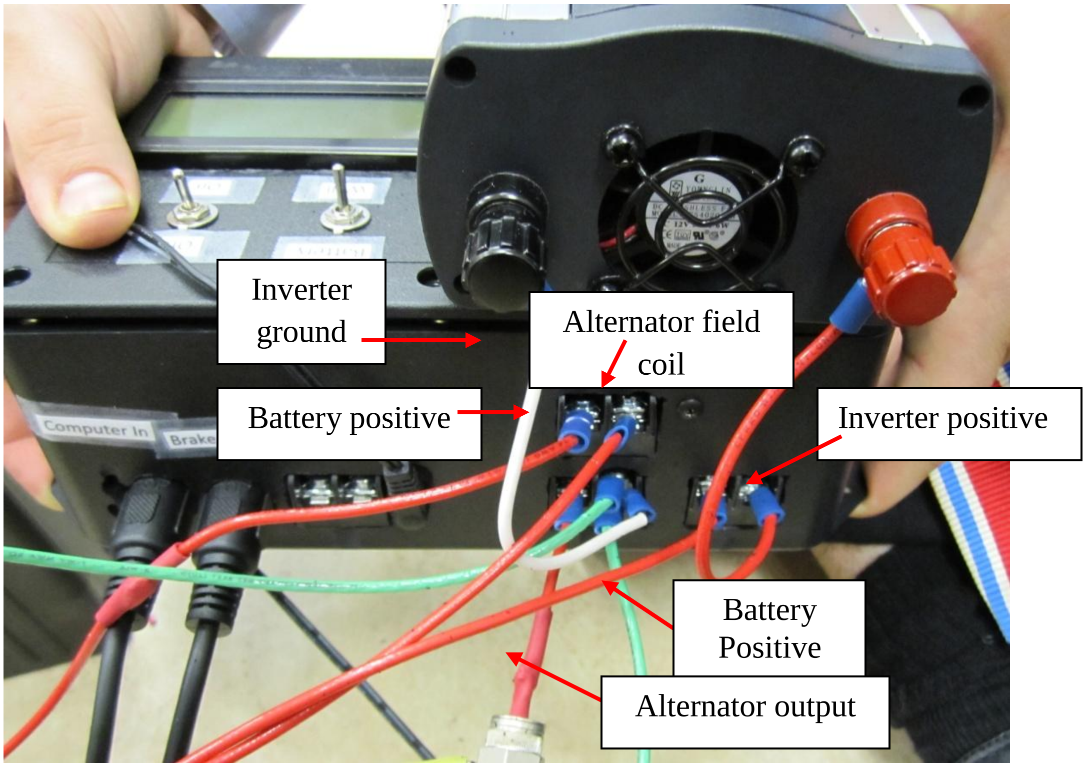

Figure 9. Wire connections from enclosed ElecTrek system.

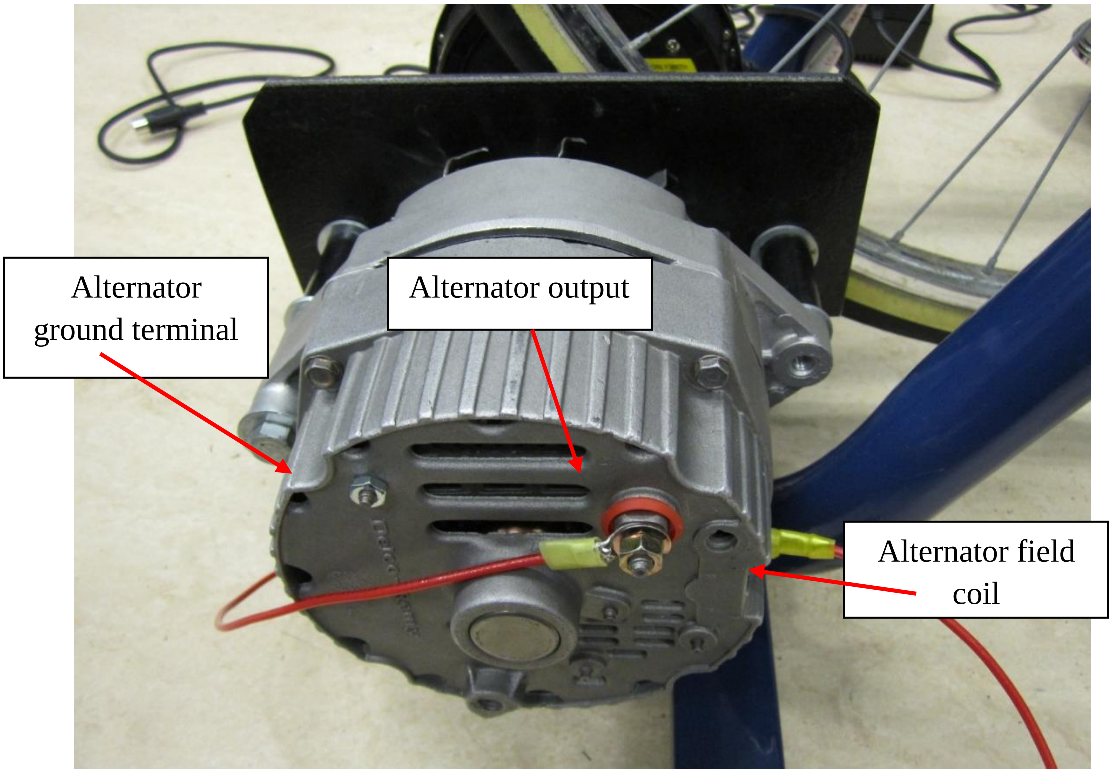

Figure 10. Alternator electrical connections.

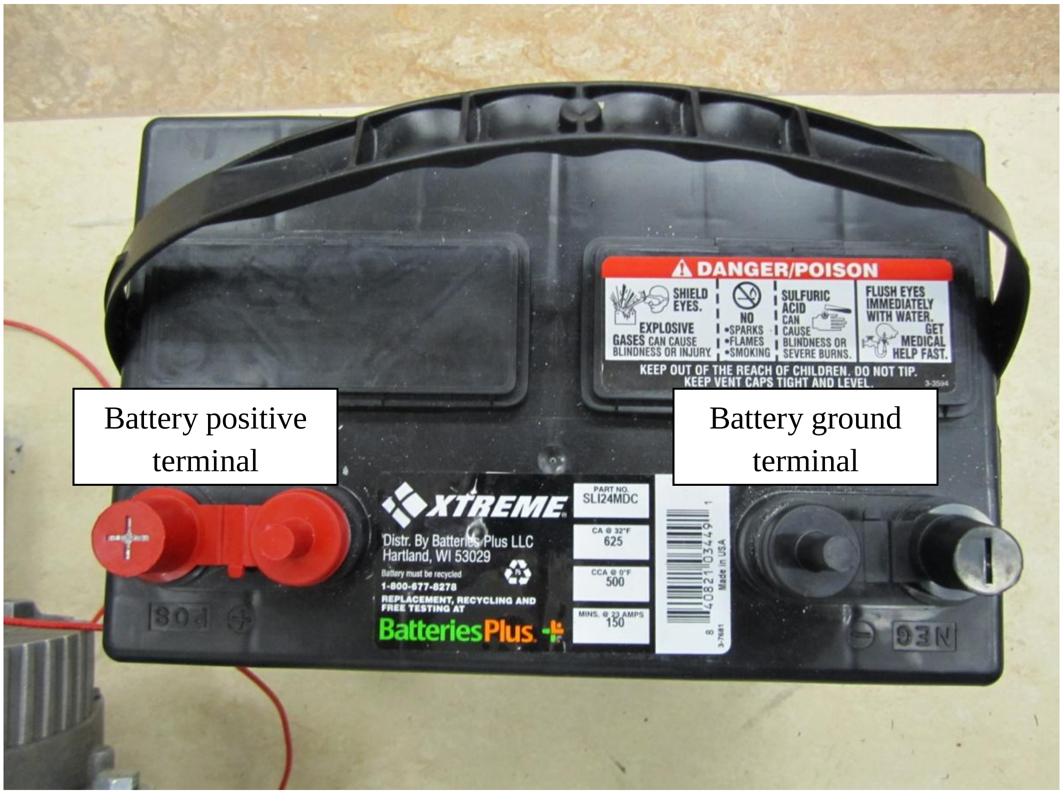

Figure 11. Battery electrical connections.

Table 3. Connections from the enclosure.

Number/Ordering	Connection To	Wire Color	Terminal type
1	Alternator output	Red	Red screw
2	Battery positive terminal	Red	Red screw
3	Battery positive terminal	Red	Red screw
4	Alternator field coil	Red	Yellow tab
5	Inverter positive	Red	Blue screw
6	Inverter ground	White	Blue screw
7	Battery ground	Green	Black screw
8	Alternator ground	Green	Blue screw

4.2 Setup

Ensure that the bike frame has plenty of room for you or observers to walk around freely, without tripping hazards. Situate the enclosure with the electronic components so that the wires connecting it to the bike stand setup are not stretched. Plug the cable from the handheld device, or external computer into the terminal labeled “CompuTrainer In” and plug the cable from the CompuTrainer brake into to the terminal labeled “Brake Out.” Either plug the wall adapter into the wall and into the round terminal next to the two CompuTrainer cables or switch the metal switch on the top of the enclosure to “Battery.”

4.2.1 Normal CompuTrainer Operation

If you want to use the CompuTrainer braking device alone, without the ElecTrek system, make sure the switch on the enclosure are switched to “Off.”

4.2.2 ElecTrek system with Battery Charging and Inverter Option

Set the main switch to “On,” with the system either plugged into the wall or the top switch set to “Battery.” In this mode, the alternator will provide all of the resistance that the rider feels. Adjust the resistance using the handheld CompuTrainer device or software.

This mode allows the battery to charge while the rider is pedaling. In addition, the rider may plug in devices to the inverter. This includes any item that can operate on a 120V AC connection, including fans, iPod chargers, or mobile phone chargers.

When the battery is completely charged and the inverter is not drawing any power, the ElecTrek system will automatically disconnect and the resistance will be provided by the CompuTrainer software as normal to prevent the battery from overcharging.

4.2.3 Battery/Inverter Stand-Alone Option

If desired, you may have the CompuTrainer system off and still use the battery and inverter to power devices. The ElecTrek system needs to be on to keep the connection between the battery and the inverter. Care should be taken to ensure the battery is not completely discharged; otherwise the inverter will lose power without warning.

4.3 How to Tell if the Product is Working

You will know if the ElecTrek system is working if the LCD is on. When pedaling, it should show the RPMs of the bike wheel. When outputting power, the “Power” should be flashing numbers and the battery symbol should show it is charging. If the ElecTrek system is providing resistance, the LCD should say “Green Mode.” If the CompuTrainer brake is providing resistance the LCD should say “Standard Mode.”

In the various modes, you will be able to tell if the system is functioning correctly if the resistance changes when the speed of the rider changes and when adjusted with the software.

If there is charge in the battery, and the ElecTrek system is on, when the inverter is turned on the LED will light up.

4.4 Troubleshooting

If the ElecTrek system is not turning on, it is most likely that it is being powered off the battery and the battery is completely discharged. If this is the case, plug the wall adapter in and turn the switch on top of the enclosure to "Wall." If the system turns on and the inverter is still not turning on, there is not enough battery charge to work the inverter and you will have to pedal to charge the battery more.

If the ElecTrek system is on and not providing any resistance, check that the CompuTrainer is on. The connections to and from the enclosure should also be checked.

If the LCD is not outputting the values it should be, first turn the system completely off and then completely on again. If that does not work, unscrew the six screws on the black box and press the black button (hidden slightly by yellow wires) shown on Figure 12. The screw terminals on the inside should also be checked to make sure they are tight as well as to make sure that there are no loose wires.

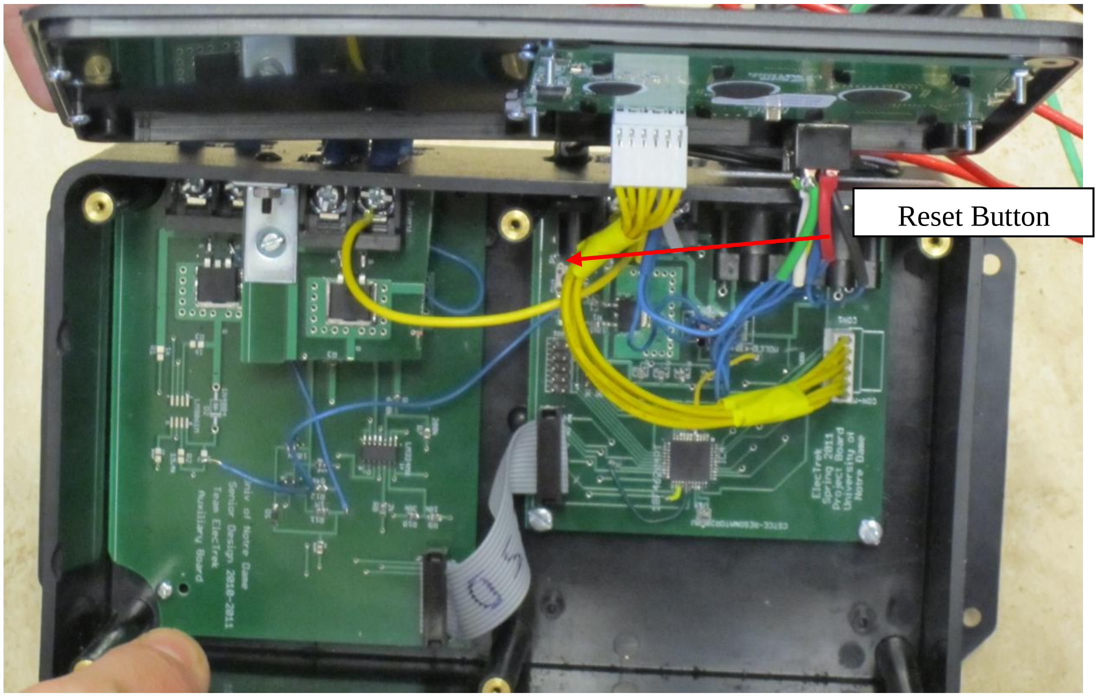

Figure 12. Inside of enclosure.

5 To-Market Design Changes

Our prototype proved the viability of our project; however it did not function ideally. Before bringing the system to market, a much better generator should be used to provide resistance instead of the alternator. We have shown that the generator is sufficient to provide all the necessary resistance, so it can replace the eddy current brake completely. Switching to a permanent magnet DC generator should provide a greater range of physical resistances to the user so that our system does not interrupt the rider's experience. Additionally, if properly picked, the generator would be more efficient than the alternator. This would improve our electrical power generation significantly.

The system can be made to generate power all the time if a grid-tie inverter was included as well. This would allow the system to provide power into the power grid, which would make using its power much easier and even provide an opportunity for the user to sell electricity to the power company. The grid-tie inverter would also allow the complete removal of the eddy current brake because the generator could be used to provide resistance constantly and without concern for energy storage capacity.

An alternative to the grid-tie inverter includes a resistive element that dissipates outside the building the head generated by the system, rather than releasing all the heat outside the building. Additionally, an ideal system would charge multiple batteries in parallel, or be able to support multiple inverters, therefore increasing the efficiency of the product in converting wasted mechanical energy to useful electrical energy.

There is also room for improvement in the mechanical connection of the generator and eddy current brake. The current shaft coupler is not exactly aligned and resulted in a great deal of vibration. A spider coupler would reduce this vibration substantially. Alternatively, the generator could connect to the eddy current brake via a gear system, which could be used to adjust the RPM of the generator as needed. Note, however, that if an appropriate generator and grid-tie inverter are used, the eddy current brake would be unnecessary and a custom wheel contact could be designed for the generator. The market-ready design of this product would be one device that both provided resistance and output a voltage and current able to charge the battery.

Another area for improvement involves the scenario when the battery charge level drops below 30%. This situation can be tested by discharging the battery using a high power consumption device, such as a soldering iron. Then, the battery can be connected to the system. If the entire system is functioning correctly, the microcontroller should read the terminal voltage of the battery and recognize that it is below the desired threshold. The microcontroller should send a signal to the auxiliary board, and the second MOSFET should completely disconnect the inverter and the battery. This means that the devices connected to the battery will not be allowed to discharge the battery below a certain level to extend the lifetime of the battery. This feature cannot be implemented in the current design because the control of the second MOSFET, between the battery and the inverter, is tied directly to the output of the DC to DC converter. Since the DC to DC converter is powered by Vdd from the microcontroller, its output cannot be easily interrupted. An automatic shut-off feature would be an important improvement upon the existing system. See Table 4 for a summary of the To-Market design requirements.

Table 4. To-Market Design Requirements.

Component	Requirements
CompuTrainer/Alternator Interface	- Will be one system instead of two integrated components. The system will use purely electrical resistance from an applied load to make the alternator harder to spin, thereby generating mechanical resistance. This would make for a more efficient combined training and energy capture system.
Energy Capture Subsystem (Alternator-Battery-Load)	- Will include a resistive element that dissipates heat generated by the system to the outside rather than in the building - Must be used with the single CompuTrainer/alternator system mentioned above - Charge multiple batteries in parallel - Alternative: Uses a grid-tie inverter to dump power directly into power grid, instead of charging batteries

6 Major Component Costs

Tables 5, 6, and 7 below give an estimated cost breakdown of our project. Our most expensive items are the alternator, battery, and printed circuit board. Other components will be required to build the charging circuit and to populate the boards, but these costs are minimal.

Table 5. Power Generation Subsystem Costs.

Component	Cost	Source
CompuTrainer system with software	$0.00	Outpost Sports
Car Alternator	$0.00	Supply closet
Alternator-CompuTrainer shaft coupler	$0.00	Courtesy of Dee Davis

Din-8 Cable	$10.63	RacerMate
Din-8 Male Connector	3 @ $1.71 each total: $5.13	DigiKey
Din-8 Female Connector	3 @ $1.67 total: $5.01	DigiKey
Current sensor on generator output (620-1226-1-ND)	2 @ $4.66 each total: $9.32	DigiKey (Allegro Microsystems)
AC/DC Adapter	$5.93	DigiKey (CUI Inc.)
Subsystem Total:	**$36.02**	

Table 6. Battery Subsystem Costs.

Component	**Cost**	**Source**
12V Deep Cycle Lead Acid Battery	$0.00	Supply Closet
Inverter	$27.25	www.amazon.com
Power MOSFETs (2)	4 @ $1.65 total: $6.60	DigiKey (International Rectifier)
Subsystem Total:	**$33.85**	

Table 7. Circuit Board Subsystem Costs.

Component	**Cost**	**Source**
Printed Circuit Board	$50.00	Advanced Circuits
Microcontroller (PIC18F4620)	3 - $0.00	Microchip samples
LCD, NHD-0420D3Z-FL-GBW	$24.90	DigiKey
Voltage Regulator, LT1763CS8-5#PBF	3 @ $1.12 total: $3.36	DigiKey
DC-DC Converter, ZXLD161	2 @ $1.01 total: $2.02	DigiKey
Quad Op-Amp, LM324MX	3 @ $0.60 total: $1.80	DigiKey
Ribbon Cable	$0.00	Supply closet
Board Enclosure	2 @ $11.93 each total: $23.86	Polycase
Power MOSFET Driver (eventually taken out of design)	4 @ $2.08 total: $8.32	DigiKey

Screw Terminals	4 @ $1.07 total: $4.28	
Subsystem Total:	**$118.54**	
Project Total:	**$188.41**	

6 Conclusions

ElecTrek’s design for Outpost Sports’ bicycle training facility allows the owner to capture the energy created by the riders into a useful form. The design generates electricity without altering the braking system or the existing software. The primary components are an alternator, battery, inverter, sensors, and a microcontroller.

During the course of the project, we discovered that instead controlling the output of the alternator, the current input to the field coil should be altered. We also discovered that a car alternator is not the best option for this use. With continued work, this project could become a viable addition to the green energy portfolio.

7 Appendices

7.1 Hardware Schematics

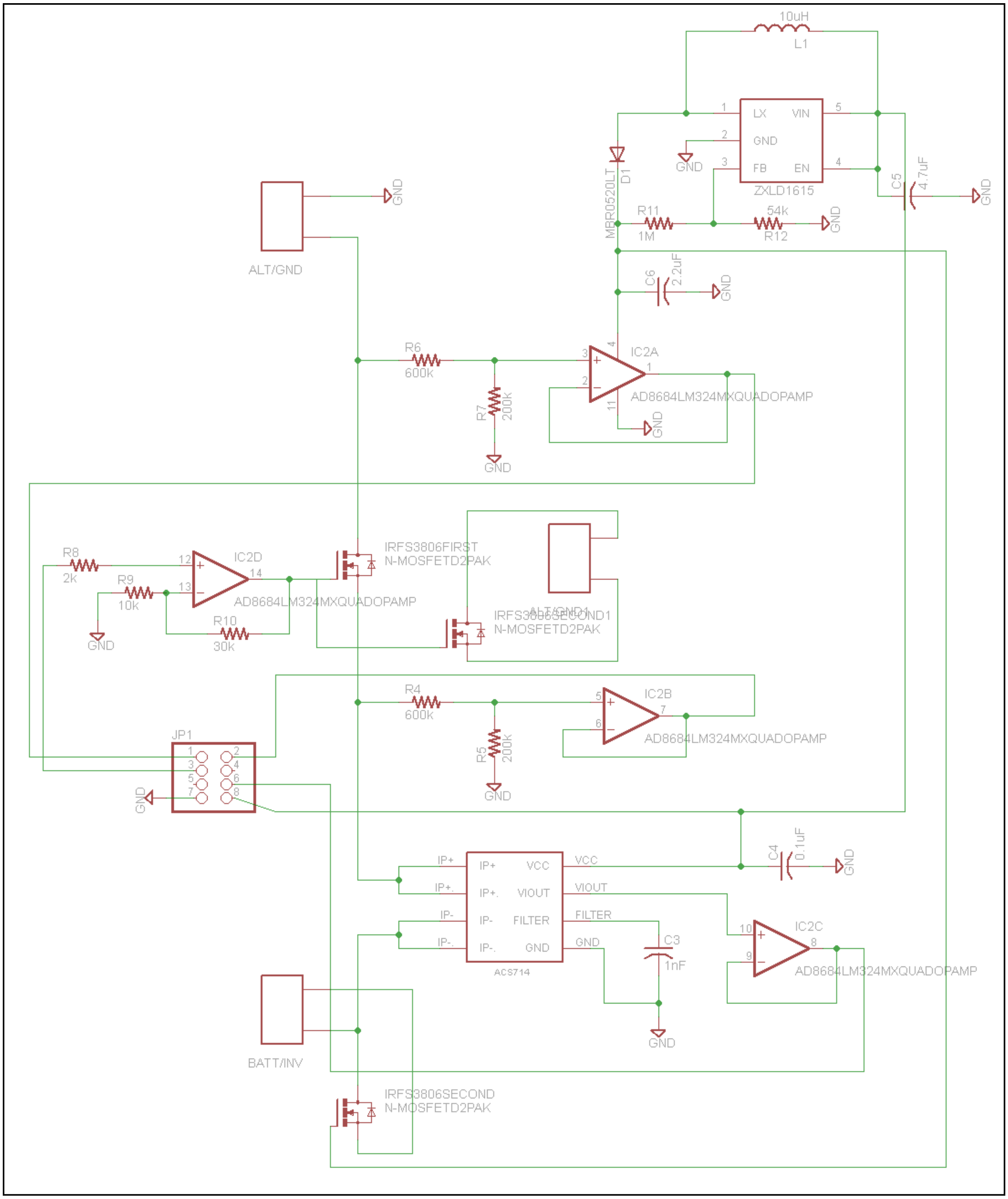

Figure 13. Final auxiliary board schematic.

Note: This schematic contains all the changes made to the auxiliary board after receiving it from the board house.

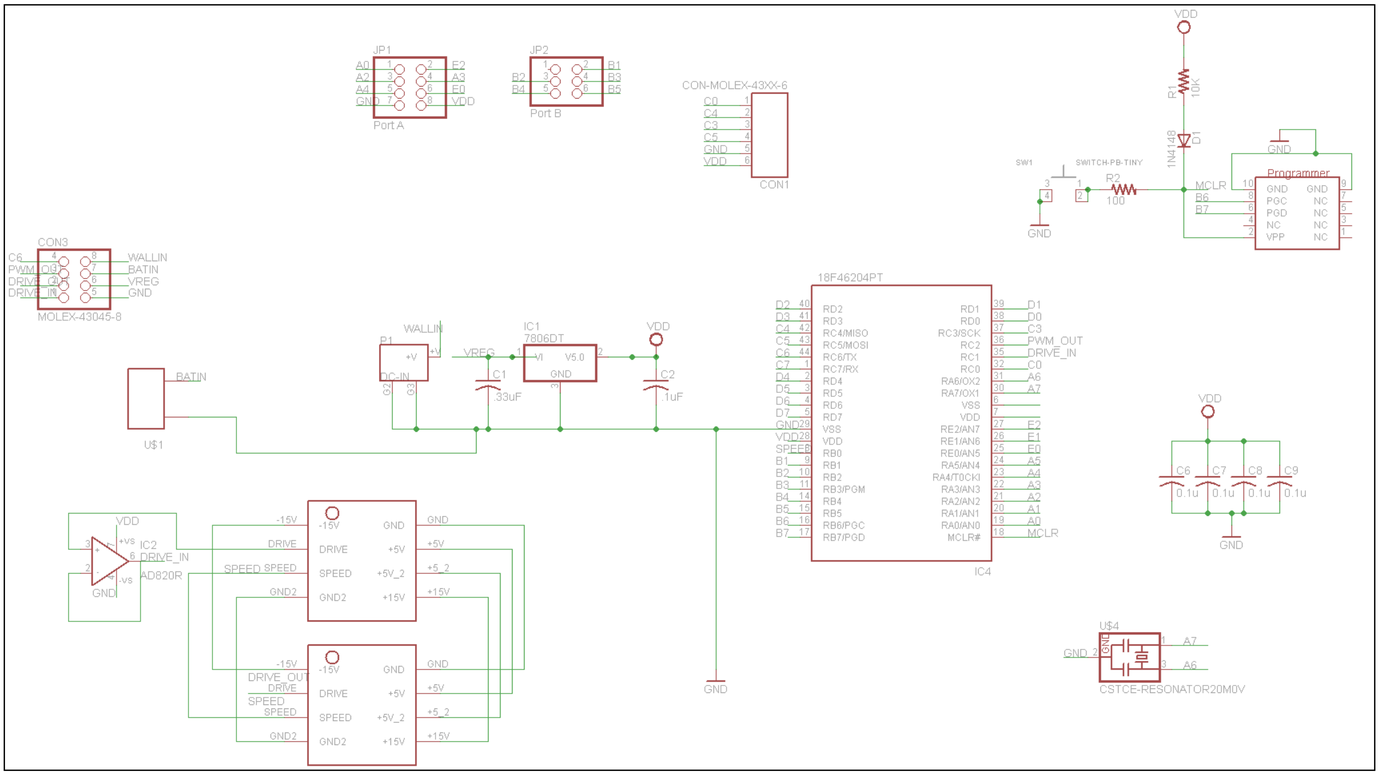

Figure 14. Final main board schematic.

Note: This schematic contains all the changes made to the main board after receiving it from the board house.

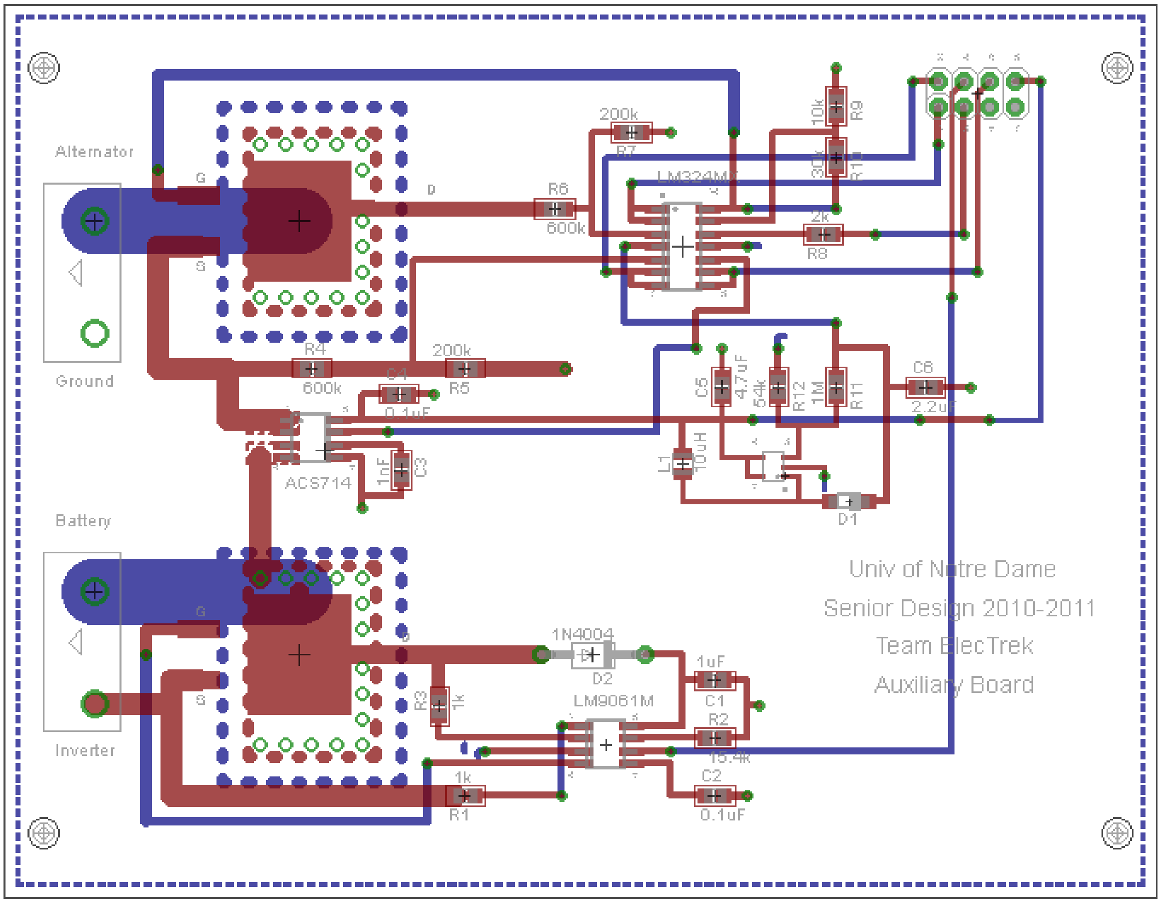

Figure 15. Main Board Layout

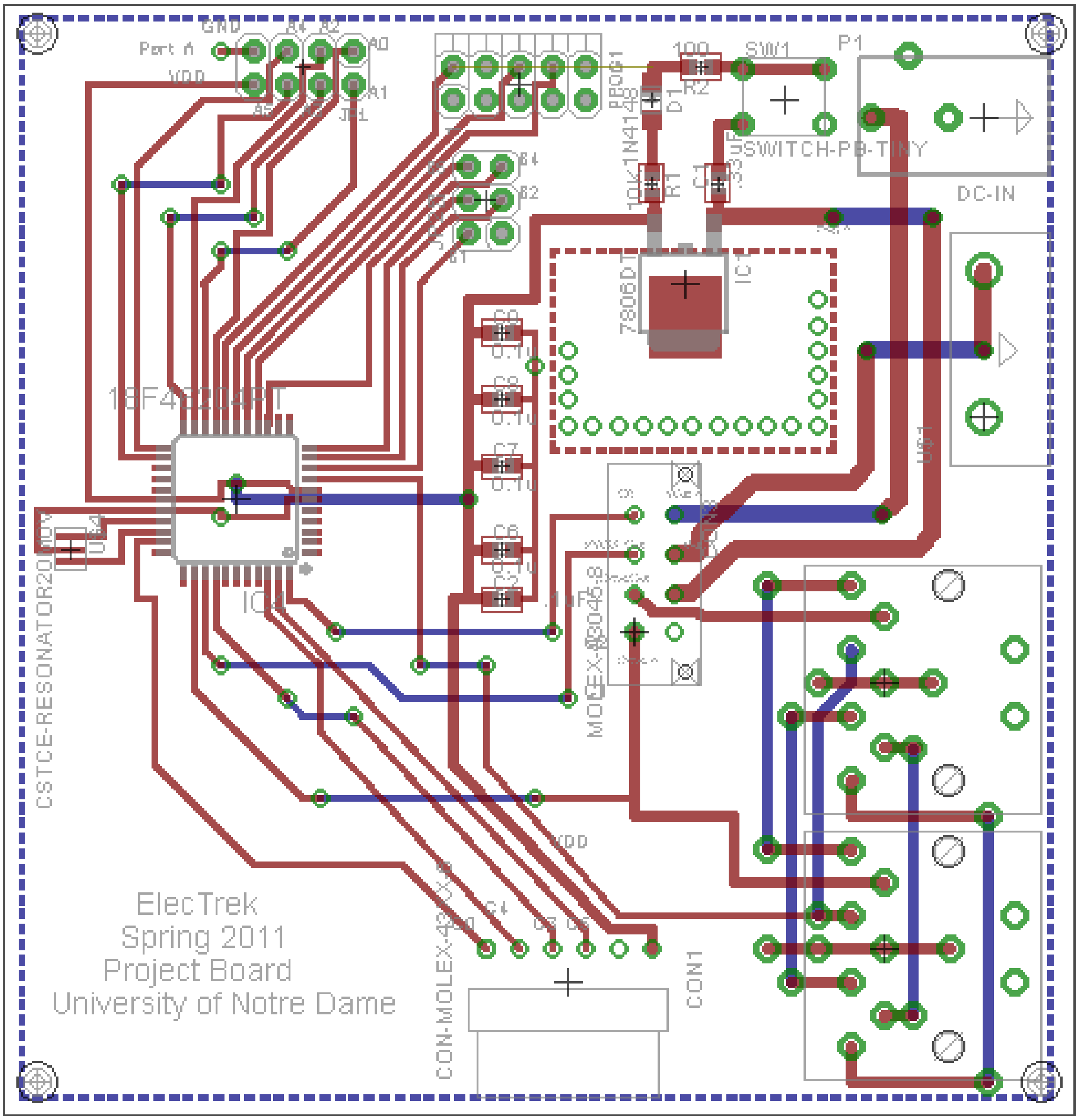

Figure 16. Main Board Layout

7.2 Software Listings

spiLib.h

```
#ifndef _SPILIB_
#define _SPILIB_
#include<system.h>


volatile bit spiWCOL@SSPCON1.7;
void spiCmd(char val);  // Send LCD command
void spiLoadBatt(); // Loads Battery Indicator Characters
void spiChar(char data); // sends ascii character to display
void spiDec(char data); // displays byte as decimal number
void spiDec(unsigned short data); //displays large decimal number
void spiFloat(unsigned short data); //displays fake float
void spiHex(char data);  // displays byte as hex number
void spiHex(unsigned short data);  // displays two bytes as hex number
void spiBin(char data);  // displays byte as binary number
void spiBin(unsigned short data);  //displays 2 bytes as binary number
void spiInit();     //initializes LDC display
void spiPrintf( const char* text ); //sends ascii string to display
void spiSetPos(char line, char pos); //sets postition on display
void spiClear(void);    // clear LCD Screen
#endif
```

spiLib.c

```
#include <system.h>
#include "spiLib.h"


void spiInit()
{
  sspcon1 = 0b00110010; //SPI1 Control Register
  sspstat.7 = 1;
  sspstat.6 = 0;
  // Configure SPI pins
  trisc.5=0;
  trisc.4=1;
  trisc.3=0;
  trisc.0=0;
  // Give the LCD a chance to catch up...
  delay_ms(500);
  spiClear();
}

void spiLoadBatt()
{
  spiCmd(0x54);
  spiChar(0x00);  // Character Address 0
  spiChar(0x00);        // Byte 1
  spiChar(0b00001111);  // Byte 2
  spiChar(0b00001000);  // Byte 3
  spiChar(0b00011011);  // Byte 4
  spiChar(0b00011011);  // Byte 5
  spiChar(0b00001000);  // Byte 6
  spiChar(0b00001111);  // Byte 7
  spiChar(0x00);        // Byte 8
  spiCmd(0x54);
  spiChar(0x01);  // Character Address 1
  spiChar(0x00);        // Byte 1
  spiChar(0b00011111);  // Byte 2
  spiChar(0b00000000);  // Byte 3
  spiChar(0b00011011);  // Byte 4
  spiChar(0b00011011);  // Byte 5
  spiChar(0b00000000);  // Byte 6
  spiChar(0b00011111);  // Byte 7
  spiChar(0x00);        // Byte 8
  spiCmd(0x54);
  spiChar(0x02);  // Character Address 2
  spiChar(0x00);        // Byte 1
  spiChar(0b00011110);  // Byte 2
  spiChar(0b00000010);  // Byte 3
  spiChar(0b00011010);  // Byte 4
  spiChar(0b00011010);  // Byte 5
  spiChar(0b00000010);  // Byte 6
  spiChar(0b00011110);  // Byte 7
  spiChar(0x00);        // Byte 8
  spiCmd(0x54);
  spiChar(0x03);  // Character Address 3
  spiChar(0x00);        // Byte 1
  spiChar(0b00001111);  // Byte 2
  spiChar(0b00001000);  // Byte 3
  spiChar(0b00011000);  // Byte 4
  spiChar(0b00011000);  // Byte 5
  spiChar(0b00001000);  // Byte 6
  spiChar(0b00001111);  // Byte 7
  spiChar(0x00);        // Byte 8
  spiCmd(0x54);
  spiChar(0x04);  // Character Address 4
  spiChar(0x00);        // Byte 1
  spiChar(0b00011111);  // Byte 2
  spiChar(0b00000000);  // Byte 3
  spiChar(0b00000011);  // Byte 4
  spiChar(0b00000011);  // Byte 5
  spiChar(0b00000000);  // Byte 6
  spiChar(0b00011111);  // Byte 7
  spiChar(0x00);        // Byte 8
  spiCmd(0x54);
  spiChar(0x05);  // Character Address 5
  spiChar(0x00);        // Byte 1
  spiChar(0b00011111);  // Byte 2
  spiChar(0b00000000);  // Byte 3
  spiChar(0b00000000);  // Byte 4
  spiChar(0b00000000);  // Byte 5
  spiChar(0b00000000);  // Byte 6
  spiChar(0b00011111);  // Byte 7
  spiChar(0x00);        // Byte 8
  spiCmd(0x54);
  spiChar(0x06);  // Character Address 6
  spiChar(0x00);        // Byte 1
  spiChar(0b00011110);  // Byte 2
  spiChar(0b00000010);  // Byte 3
  spiChar(0b00000010);  // Byte 4
  spiChar(0b00000010);  // Byte 5
  spiChar(0b00000010);  // Byte 6
  spiChar(0b00011110);  // Byte 7
  spiChar(0x00);        // Byte 8
}
void spiCmd(char cmd)
{
  spiChar(0xFE);
  spiChar(cmd);
}
```

```
void spiChar(char data)
{
  latc.0=0;
  char throw;
  throw = sspbuf;
  sspbuf = data;
  delay_us(500);
  latc.0=1;
}
void spiDec(char data)
{
      spiChar(((data/100) & 255)+0x30);
      data = data%100;
      spiChar(((data/10) & 255)+0x30);
      spiChar(((data%10) & 255)+0x30);
      return;
}
void spiDec(unsigned short dat)
{
      unsigned short val;   // ascii results
      unsigned short div;
      unsigned short data;
      char i;
      char digit;
      data = dat;  // make it unsigned
      div = 10000;
      for(i=0; i <= 4; ++i)   // get all 5 digits
      {
            val = data/div;       // get most signif. digit
            spiChar(val + '0');    // print digit
            data -= val * div;   // what we’ve printed
            div=div/10;         // adjust divisor
      }
      return;
}
void spiFloat(unsigned short data)
{
  unsigned short val;
  unsigned short div;
  char i;
  div = 10000;
  for(i=0;i<5;i++)
  {
    val = data/div;
    spiChar(val + '0');
    data -= val*div;
    div=div/10;
    if(i==2){spiChar('.');}
  }
  return;
}
void spiHex(char data)
{
      char n;
      n = ((data >> 4) & 0x0F) + 0x30;
      if (n > 0x39) n = n+7;
      spiChar(n);
      n = (data & 0x0F) + 0x30;
      if (n > 0x39) n = n+7;
      spiChar(n);
      return;
}
void spiHex(unsigned short data)
{
    char n;
      n = ((data >> 12) & 0x0F) + 0x30;
      if (n > 0x39) n = n+7;
      spiChar(n);
      n = ((data >> 8) & 0x0F) + 0x30;
      if (n > 0x39) n = n+7;
      spiChar(n);
      n = ((data >> 4) & 0x0F) + 0x30;
      if (n > 0x39) n = n+7;
      spiChar(n);
      n = (data & 0x0F) + 0x30;
      if (n > 0x39) n = n+7;
      spiChar(n);
      return;
}
void spiBin(char data)
{
      char n;
      char temp;
      temp =0x80;
      for( n=1; n<=8; ++n)
      {
            if(temp & data)
            spiChar(0x31);
            else
            spiChar(0x30);
            temp = temp >> 1;
      }
      return;
}
void spiBin(unsigned short data)
{
      char n;
      unsigned short temp;
      temp =0x8000;
      for( n=1; n<=16; ++n)
      {
            if(temp & data)
            spiChar(0x31);
            else
            spiChar(0x30);
            temp = temp >> 1;
      }
}
void spiPrintf( const char* text )
{
      while(*text!=0)
      {spiChar(*text++);}
      return;
}
void spiSetPos(char line, char pos)
{
  if(line==1) {
    pos = 0x40+pos;
  } else if(line==2) {
    pos = 0x14+pos;
  } else if(line==3) {
    pos = 0x54+pos;
  }
  spiCmd(0x45);
  spiChar(pos);
}
void spiClear(void)
{
    spiCmd(0x51);
    spiCmd(0x46);
}
```

Main.c

```
/****************************************************
*      Main Code                                    *
*   Reads PWM signal from CompuTrainer              *
*   Reads Speed Signal from CompuTrainer            *
*   Outputs new PWM signal to CompuTrainer          *
*****************************************************/
#include <system.h>
#include "spiLib.h"
#pragma DATA _CONFIG1H, _OSC_HS_1H
#pragma DATA _CONFIG2H, _WDT_OFF_2H
#pragma DATA _CONFIG4L, _LVP_OFF_4L & _XINST_OFF_4L
#pragma DATA _CONFIG3H, _MCLRE_ON_3H & _PBADEN_OFF_3H
#pragma CLOCK_FREQ 20000000

// Here Be Semaphores
bool PWMTICK = false;
bool SPDTICK = false;
bool LCDUPDATE = false;

bool GREENPOWER = true;

// Interrupt Flags

volatile bit ccp2if@PIR2.0;

volatile bit tmr3if@PIR2.1;

volatile bit tmr0if@INTCON.2;

volatile bit tmr1if@PIR1.0;

volatile bit int0if@INTCON.1;

volatile bit int1if@INTCON3.0;

// Speed and PWM duty cycle variables

unsigned short spd = 0;

unsigned short pwm = 348;

void interrupt(void)

{

    if(int1if) // Falling Edge

    {

      int1if = 0;

      tmr3h=0;

      tmr3l=0;

      if(GREENPOWER)

      { lata.2 = 1; }

    }

    if(ccp2if) // Rising Edge

    {

      ccp2if = 0;

      pwm = ccpr2h<<8;

        pwm = ccpr2l|pwm;

      PWMTICK = true;

      if(GREENPOWER){lata.2=0;}

    }
    if(tmr3if) // Timer 3 expired
    {
      tmr3if=0;
      pwm = 0;
    }
    if(tmr0if) //Speed timer expired
    {
      tmr0if=0;
      spd = 39062;
      SPDTICK = true;
    }
    if(int0if) // Speed Tick
    {
      SPDTICK = true;
      spd = tmr0h<<8;
      spd = tmr0l|spd;
      tmr0h=0;
      tmr0l=0;
      int0if=0;
    }
    if(tmr1if) // UPDATE THE LCD
    {
      tmr1if = 0;
      LCDUPDATE = true;
    }
}
void main(void)
{
    unsigned short spdi = 0;
    unsigned short pwmo = 0;
    unsigned long aval=0;

    unsigned long bval=0;

    unsigned long cval=0;

    unsigned long pow = aval*cval;

    unsigned long ppow = pow;

    char ads = 0;

    char pc = 0;

    unsigned long p = 0;

    /////////////////////////////////////

    //   A/D Converter                 //

    /////////////////////////////////////

    //

    // A/D Converter Setup

    // A0 - Alternator Voltage

    // A1 - Battery Voltage (AN7) -> E2

    // E0 - Current Sensor (AN5)

    trise.2 = 1;

    trisa = 1; // Make Port A an input

    // Configure A/D

    adcon0 = 0b00000011;

    adcon1 = 0b00001001;

    adcon2 = 0b10000111;

    volatile bit GoDone@ADCON0.1;  // Flag for complete A/D read

    GoDone = 1;

    ////////////////////////////////////////

    //         LCD INITIALIZATION         //

    ////////////////////////////////////////

    //

    t1con = 0b11110001;

    pie1.0 = 1; // Enable Timer0
```

```
    char i=0;
    short y=0; // Keep track of battery animation
    spiInit();
    spiLoadBatt();
    spiClear();
    ///////////////////////////////////////////
    //    PWM OUTPUT                         //
    ///////////////////////////////////////////
    //
      // Signal P1A = pin RC2
      // Signal P1D = pin D7
    //
      //----PWM Period----
      // PWM period = 279us = [(PR2) + 1] * 4 * (1/20MHz) * TMR2 Prescalar
      // when PR2 = 0x56 and TMR2 Prescalar = 16
      // Frequency = 3.597kHz
      // >>> Note that this gives us a period of 278us.
      // >>> Meh. Close enough.
      pr2 = 0x56;
      //----Set PWM ports as outputs----
      trisc.2 = 0;
      trisd.7 = 0;
      //----PWM Duty Cycle----
      // PWM Duty Cycle = CCPR1L:CCP1CON<5:4> * (1/20MHz) * TMR2 Prescalar
      // 0 => 0% Duty Cycle
      // 348 => 100% Duty Cycle
      //ccpr1l = 0b01000000; // top 8 bits (9-2) of 10bit duty cycle number
      volatile bit dc1b1@CCP1CON.5 = 0;   // Bit1 of duty cycle
      volatile bit dc1b0@CCP1CON.4 = 0;   // Bit2 of duty cycle
      ccpr1l = pwmo >> 2; // top 8 bits (9-2) of 10bit duty cycle number
```

```
  dc1b1 = pwmo;          // Bit0 of duty cycle

  dc1b0 = pwmo>>1;

  ccp1con = 0b01001101;               // Configure in PWM mode

  //----Config and Enable Timer2----

  volatile bit t2ckps1@T2CON.1 = 1;   // 10 = prescale of 16

  volatile bit t2ckps0@T2CON.0 = 0;

  volatile bit tmr2on@T2CON.2 = 1;

///////////////////////////////////////

//       PWM  INPUT                 //

///////////////////////////////////////

//

// Signal CCP2 = pin C1

// Signal INTCON1 = pin B1

volatile bit gie@INTCON.7 = 1; //Globally enable interrupts

volatile bit peie@INTCON.6 = 1; //Enable peripheral interrupts

ccp2con = 0b00000101; // rising edge

intcon2.5 = 0; // falling edge

intcon3.3=1; //enable intcon1 interrupt

//---- Configure PWM ports as inputs----

trisc.1 = 1;

trisb.1 = 1;

//----Config and Enable Timer3----

t3con = 0b10111001;

volatile bit tmr3ie@PIE2.1 = 1; // Enable Timer 3 Interrupt

volatile bit ccp2ie@PIE2.0 = 1; //Enable interrupts for ccp2

ccp2if = 0;

tmr3h=0;

tmr3l=0;

/////////////////////////////////////

///         SPEED READ             //
```

```
/////////////////////////////////////
//
// INTCON0 -> Pin B0
trisb.0 = 1;
volatile bit int0ie@INTCON.4 = 1;  // Enable INTCON0
int0if=0;
volatile bit intedg0@INTCON2.6=0;  // Trigger on Falling Edge
t0con = 0b10010110; // Timer0
intcon.5=1; // enable timer0
tmr0h=0;
tmr0l=0;

////////////////////////////////////////
//   Comparator Reference Volt Module //
////////////////////////////////////////
//
//cvrcon = 0b11001111;
//cvrcon = 0b00000000;
trisa.2 = 0;
if(GREENPOWER)
  {
    spiPrintf("Green Mode");
  } else {
    spiPrintf("Standard Mode");
  }
  spiSetPos(0,17);
// Battery Characters :D
spiChar(0x00);
spiChar(0x01);
spiChar(0x02);
```

```
    spiSetPos(1,0);
    spiPrintf("RPM: ");
    spiDec(spd);
    spiSetPos(2,0);
    spiPrintf("Power:       W");
    spiSetPos(3,0);
    spiPrintf("Peak Power:       W");
    while(1)
      {
        if(PWMTICK)
        {
          PWMTICK=false;
        if(pwm>694)
        {
          pwmo = 348;
        } else {
          pwmo = pwm<<1;
          }
          if(!GREENPOWER)
          {
            ccpr1l = pwmo >> 2; // top 8 bits (9-2) of 10bit duty cycle
            dc1b1 = pwmo;          // Bit0 of duty cycle
            dc1b0 = pwmo>>1;    // Bit1 of duty cycle
          }
        }
        if(SPDTICK)
        {
          SPDTICK=false;
          spdi = (39062/spd)>>4;
        }
```

```
  // A/D reading
  if(GoDone == 0 && ads == 0)
{
  aval = adresh<<8;
  aval = aval + adresl;
  aval = ((aval*500)>>8);
  adcon0 = 0b00011111; // Now Measure Battery Voltage
  ads = 1;
}
if(GoDone==0 && ads == 1)
{
  bval = adresh<<8;
  bval = bval + adresl;
  bval = ((bval*500)>>8);
  adcon0 = 0b00010111; // Now Measure Current
  ads = 2;
  if(bval >= 1270)
  {
    GREENPOWER = false;
    lata.2 = 0;
  } else {
    GREENPOWER = true;
  }
}
if(GoDone==0 && ads == 2)
{
  cval = adresh<<8;
  cval = cval + adresl;
  cval = ((cval*500)>>10);
  if(cval>250)
```

```
      { cval = (cval-250)<<4;}
      else {cval = 0;}
      adcon0 = 0b00000011; // Now Measure Alternator
      ads = 0;
      p = (aval*cval)/100;
      if(p>0 || pc > 20)
      {
        pow = p;
        pc = 0;
      } else {
        pc = pc+1;
      }
      if(pow>ppow){ppow=pow;}
    }
    if(LCDUPDATE)
    {
      spiSetPos(0,0);
      if(GREENPOWER)
      {
        spiPrintf("Green Mode   ");
      } else {
        spiPrintf("Standard Mode");
      }
      // Speed stuff
        spiSetPos(1,5);
        spiDec(spdi);
        // Current Power
      spiSetPos(2,7);
      spiFloat(pow);
      // Peak Power
```

```
        spiSetPos(3,12);

        spiFloat(ppow);

              //////////////////////////////////////////////////

              //        CHARGING BATTERY ANIMATION            //

              //////////////////////////////////////////////////

            if(spdi==0)

            {

              if(bval <= 1130) {y=0;}

              else if(bval <= 1170) {y=1;}

              else if(bval <= 1206) {y=2;}

              else if(bval <= 1240) {y=3;}

              else {y=4;}

            } else {

              y=i%5;

            }

            spiSetPos(0,17);

              if(y==0)

              {

                  // EMPTY

                  spiChar(0x03);

                  spiChar(0x05);

                  spiChar(0x06);

              } else if(y==1)

              {

                  // 1/4 FULL

                  spiChar(0x03);

                  spiChar(0x05);

                  spiChar(0x02);

              } else if(y==2)

              {
                  // HALF FULL
                  spiChar(0x03);
                  spiChar(0x04);
                  spiChar(0x02);
              } else if(y==3)
              {
                  // 3/4 FULL
                  spiChar(0x03);
                  spiChar(0x01);
                  spiChar(0x02);
              } else {
                  // FULL
                  spiChar(0x00);
                  spiChar(0x01);
                  spiChar(0x02);
              }
              i = i+1;
              LCDUPDATE = false;
      }
      }
}
```

7.3 Relevant parts/component datasheets

Auxiliary Board:

MOSFET driver datasheet:
http://www.national.com/ds/LM/LM9061.pdf

Current sensor datasheet:
http://www.allegromicro.com/en/Products/Part_Numbers/0714/0714.pdf

Power MOSFET datasheet:
http://www.irf.com/product-info/datasheets/data/irfs3806pbf.pdf

DC-to-DC Converter datasheet:
http://www.diodes.com/datasheets/ZXLD1615.pdf

Quad Op-Amp datasheet:
http://www.fairchildsemi.com/ds/LM%2FLM324.pdf

10 µH Inductor datasheet:
http://www.tdk.co.jp/tefe02/e533_mlz2012.pdf

High Current Connector Screw Terminal:
ENG_CD_6-1437657-3_K4 High Current Connector.pdf

Main Board:

Voltage Regulator datasheet:
http://cds.linear.com/docs/Datasheet/1763fg.pdf

LCD datasheet:
http://www.newhavendisplay.com/specs/NHD-0420D3Z-FL-GBW.pdf

High Current Connector Screw Terminal:
ENG_CD_6-1437657-3_K4 High Current Connector.pdf

Microcontroller datasheet:
http://ww1.microchip.com/downloads/en/devicedoc/39626b.pdf

DIN-8 Connector datasheet:
http://products.cui.com/getPDF.aspx?fileID=2475