Rapid prototyping and simulation for EV powertrains

University of Pennsylvania
ESE 350: Embedded Systems/Microcontroller Laboratory
Andrew Botelho & William Price
May 9, 2012
Table of Contents
Abstract ......................................................................................................................... 2
Introduction .................................................................................................................. 2
Vehicle Modeling and control. ..................................................................................... 4
  Motor Modeling .......................................................................................................... 6
  Scaling ......................................................................................................................... 8
  Control ......................................................................................................................... 9
System Architecture .................................................................................................... 11
  Hardware ...................................................................................................................... 11
  Software ....................................................................................................................... 13
Design .......................................................................................................................... 15
  Mechanical .................................................................................................................. 15
    Modular Sections ....................................................................................................... 16
    Power flow indicator LEDs ...................................................................................... 17
    Safety and charging switches .................................................................................. 19
  Electrical (Hardware) .................................................................................................. 19
  Software ....................................................................................................................... 21
Testing and Evaluation ............................................................................................... 22
Conclusion ................................................................................................................... 23
Links ............................................................................................................................. 24
Abstract

Protodrive is a rapid prototyping and simulation platform for electric vehicle powertrains. The powertrain is modeled at the small-scale, making it low-cost and compact enough to fit on a desk. It consists of a physical model of an electric vehicle powertrain coupled to an active dynamometer, making it possible to run the powertrain through its full speed and torque range. The fact that this system has been constructed in hardware allows it to capture intricacies in vehicle operation that may be missed by simulation in software alone. The system is designed to be highly modular and configurable, with all of its major components being isolated allowing for easy replacement. Its small scale and modularity allow for rapid and easy construction of new architectures and subsequent design iteration. Protodrive has many applications, some of which include generating approximations of the range of full-scale electric vehicles, testing the performance of novel powertrain designs, and predicting the miles per gallon equivalent (MPGe) rating of a vehicle.

Introduction

Electric Vehicles (EVs) have had a recent resurgence in popularity and are showing promise as a future mainstream means of transportation. However, the low energy density, high cost and long recharging time of batteries are formidable obstacles to mass consumer acceptance. There are a number of things that can be done to increase the viability of EVs, such as:

- Powertrain system optimization to extract the maximum range
- Development of better tools to predict range and reduce “range anxiety”
- Optimal fuel control and driver behavior influence to increase range

An electric vehicle powertrain consists of all the components necessary to deliver power to the wheels. This typically consists of a battery, motor controller, motor, gearbox and differential. Currently, EV powertrains are modeled and simulated in software and then prototyped and tested in a full scale vehicle. While software can provide decently accurate predictions of performance, it may fail to miss some of the detailed intricacies of a real system. Full scale models obviously demonstrate all real problems, however, iterating on a full scale vehicle is time consuming and expensive.

Protodrive is a small scale electric vehicle prototyping platform that attempts to find a middle ground between simulating purely in software, and prototyping at full scale. It is a real hardware system, representative of a real powertrain, however, it is implemented at a scale small enough to fit on a desk top. The hope is that it will allow the quick and cost effective
characteristic of simulating in software, while still being able to capture the intricacies of real hardware performance.

Protodrive has a number of interesting applications that will further EV development. These include:

- Enabling rapid prototyping and evaluation of novel powertrain architectures
- Simulating federal drive cycles to determine a vehicle’s fuel consumption and MPGe rating (Miles Per Gallon equivalent)
- Predicting range, when coupled with elevation data from Google maps and a driver control strategy

Protodrive is an ongoing research project that will be carried out by the mlab at the University of Pennsylvania. The scope of this project for ESE 350 is to develop the initial hardware and software platform.

The requirements are:

- Develop a platform that models an electric vehicle powertrain at the small scale.
- Have the ability to access the full speed and torque range of the powertrain. This is necessary to calculate energy consumption.
- Be able to run the powertrain in both the drive mode and regenerative braking mode.
- Use an embedded microcontroller to control the system.
- Be able to measure system outputs such as motor voltage and current, to allow for closed loop control.

The reach goals are:

- Make the platform user friendly, so that using the platform is intuitive, allows for rapid prototyping and allows other students in mlab to easily continue the project.
- Develop control systems for the powertrain.
- Incorporate a novel powertrain design.
Vehicle Modeling and control

Protodrive is a scaled down version of an electric vehicle powertrain coupled to a load that represents the forces on the vehicle. At any point in time, the total force exerted by the wheels of a vehicle can be broken up into the following component forces:

\[ F_t = F_a + F_g + F_d + F_r \]

- **Total tractive force at wheels**
- **Force due to a slope**
- **Aerodynamic drag**
- **Rolling resistance of the tires**

\[ F_t = m_a + m_v g \sin \alpha + \frac{1}{2} A_f C_d v^2 + C_r m_v g \cos \alpha \]

- \( m_v \) - vehicle mass
- \( \alpha \) - acceleration
- \( g \) - gravity
- \( \alpha \) - slope angle
- \( A_f \) - projected frontal area
- \( C_d \) - coefficient of drag
- \( v \) - velocity
- \( C_r \) - coefficient of rolling resistance

Additionally, the mass, \( m \), in \( F_t \) can be broken up into the real vehicle mass \( m_v \) and a fictitious mass, \( m_r \), which is added due to the inertia of the rotating components of the vehicle eg. wheels, gearbox and rotor. The conversion of inertia to equivalent mass is shown below.
Vehicles travel with highly varying speeds, particularly in an urban area where there are frequent stops. As an example, the graph below shows the E.P.A. UDDS drive cycle, commonly known as the City Test. This is a standard test used to approximate how a vehicle’s speed will vary while driving in an urban environment.
The equations above also demonstrate that the force required by a vehicle is highly variable and partly dependent on the instantaneous speed. Therefore, a key element in the design of Protodrive was figuring out how to design a system such that both the speed and torque on the vehicle motor could be varied and precisely controlled. Additionally, it is worth noting that the forces to produce acceleration and the force used to climb a slope are conservative forces and will add energy to the vehicle in the form of kinetic energy and potential energy. The force to overcome drag and rolling resistance are non-conservative, causing energy losses to the environment. In an electric vehicle, the conservative forces can be partially recovered due to regenerative braking (regen), which converts potential and kinetic energy into electrical energy using the motor as a generator. This electrical energy can then be used to recharge the battery. To summarize, the two primary requirements for our system were:

- The ability to precisely control the speed and torque on the vehicle motor.
- The ability to use regenerative braking to recover energy and recharge the battery.

We explored multiple concepts that would allow us to fulfill these two requirements. The final architecture has one motor that acts as the vehicle motor, which is rigidly coupled to second identical motor which acts as a load and represents all the forces on the vehicle. By controlling both motors at the same time, it is possible to precisely control the speed and torque of the vehicle motor.

![Diagram of the selected vehicle and load configuration]

**Motor Modeling**

Brushed DC motors were selected due to their low cost and simplicity of control. Additionally, it was only possible to find a motor controller with a regenerative braking function for brushed DC motors (controllers with regen braking are readily available for full sized brushless vehicle motors, but seem to be unavailable for small brushless motors). The equivalent electrical circuit of the vehicle motor and the load motor are given below. It is important to note that the motors are rigidly coupled; therefore, the shaft speed and back EMF for both motors will be the
same. Note that the “D” subscript on the vehicle powertrain stands for “Device Under Test” and is used to distinguish the Protodrive vehicle powertrain from a real vehicle power train.

\[ V_D = R I_0 + L \frac{dI_0}{dt} + V_{\text{emf}} \]
\[ V_L = R I_L + L \frac{dI_L}{dt} + V_{\text{emf}} \]

\[ V_{\text{emf}} = k_e \omega \quad T = k_t I \]

Symbols:
- \( V \) - supply voltage [V]
- \( V_{\text{emf}} \) - back emf [V]
- \( I \) - current [A]
- \( L \) - inductance [H]
- \( R \) - resistance [Ohm]
- \( k_e \) - speed constant \([V/(\text{rad/s})]\) 
- \( k_t \) - torque constant \([\text{Nm/A}]\)
- \( \omega \) - angular velocity [rad/s]
- \( \dot{\omega} \) - angular acceleration [rad/s]

The mechanical coupling equation is as follows.

\[ J \dot{\omega} = T_D + T_L + T_f \quad \text{Note: } T_f \text{ always } < 0 \]

Symbols:
- \( J \) - moment of inertia of 2 motors and a coupler \([\text{kgm}^2]\)
- \( T_D \) - vehicle motor torque \([\text{Nm}]\)
- \( T_L \) - load motor torque \([\text{Nm}]\)
- \( T_f \) - Torque to overcome friction \([\text{Nm}]\)
By combining both the electrical and the mechanical model it is possible to determine the necessary motor terminal voltages to give a desired \( T_{\text{ref}} \) and \( \omega_{\text{ref}} \). In the equations below, \( L \) is assumed to be zero for simplicity.

\[
V_D = \frac{R T_{\text{ref}}}{k_t} + k_e \omega_{\text{ref}} \quad V_L = \frac{R (J \omega_{\text{ref}} - T_{\text{ref}} - T_f)}{k_t} + k_e \omega_{\text{ref}}
\]

\( T_{\text{ref}} \) and \( \omega_{\text{ref}} \) come from the vehicle force model and some velocity profile. For example, if a drive cycle were to be simulated, \( \omega_{\text{ref}} \) would be derived from the velocity profile of the drive cycle and then \( T_{\text{ref}} \) would be calculated from the vehicle force model based on the instantaneous velocity and acceleration. In order to translate the real vehicle forces to the Protodrive vehicle powertrain, a scaling model is needed.

**Scaling**

The goal of the Protodrive system is to use the same amount of energy as a real vehicle, but reduced by some constant.

\[
W_{\text{vehicle}} = k W_{\text{protodrive}}
\]

\[
\int_0^{t_v} \omega_v \, dt = k \int_0^{t_0} \omega_0 \, dt
\]

if \( \omega_v = \omega_0 \) and \( t_v = t_0 \), then \( T_v = k T_0 \)

The above equations show that by keeping the time scale and the angular velocity equal to that of the real vehicle, the Protodrive torque can just be scaled down by a constant. This will work in reality, since many small brushed DC motors are available with the same speed ranges as larger motors used in electric vehicles.
Control
A large part of the future research on Protodrive will be developing control models and writing the software for these models. There are 3 very useful types of control that can be used with Protodrive:

1. Given a velocity profile, and gradient, calculate the fuel consumption. This will be useful for verifying the accuracy of the Protodrive simulation. For example, a vehicle could be driven around the city on a particular route, and the velocity could be recorded along the route. Knowing the path taken, the elevation data could be obtained for the route. At the end of the journey, the real vehicle energy consumption can be measured. The route can then be simulated with Protodrive, and the final energy consumption should match in both cases (once scaling has been taken into account).

2. Running a federal drive cycle to determine a vehicle’s MPGe. In this model, the velocity profile is supplied. It must be tracked and the correct force must be applied for each instantaneous velocity.

3. Given starting and ending points, and a control strategy, calculate vehicle range. This would be used to determine the range of the vehicle based on a given route and Google maps data. Because the velocity profile is not known, some driver control strategy is necessary eg. quickest time between points, or optimal fuel control. This has a number of applications, for example, letting drivers know how many times they need to stop and charge or replace their battery on a long trip, or developing optimal control strategies for autonomous vehicles.

A high level control flow chart schematic is shown for the three control types below. At the time of this paper’s submission, we have been working on a control model for a federal City Cycle test using Simulink and Simscape. This is not a trivial task as the system is not a Single Input Single Output system that is commonly encountered in classes and textbooks. The control model will use a supplied drive cycle velocity profile as a reference and use this to calculate vehicle forces. The gearbox ratio and scaling factor will be taken into account to determine a final reference torque and angular velocity. This will then be fed into a closed loop control scheme that uses the motor speed and current measurements in the feedback loop. The controls project is beyond the requirements of this project, but will be available on our blog when it is ready.
Values

Transfer function

Common to all paths

1. Given $v(t)$, $\alpha(t)$, distance

2. Federal Test Procedure

3. Control strategy eg. Fuel optimal control or quickest time to destination.
System Architecture

Hardware

The hardware for this system can be broken into three main groups: the powertrain representing the vehicle, the powertrain representing the load on the vehicle, and the computing system. A high level schematic is shown above with high power lines shown in red, low power signal lines in blue and mechanical links in black.

The vehicle powertrain is comprised of a motor, a motor controller, and an energy storage system which contains batteries and a supercapacitor (supercap) which one can switch between by controlling the relay. The energy storage system is connected to the motor controller, which is then connected to the drive motor.

The loading system is a slightly simpler mirror of the drive system: the energy storage system is simply batteries, which are then connected to the motor controller, which is in turn connected to the load motor. The load motor and the drive motor are then mechanically coupled so that they will spin with the same angular velocity.
The computing system consists of an mbed microcontroller connected to a PC with MATLAB which was used for input and output. Various other hardware components were connected to the mbed for measurement: the battery and supercapacitor voltages were measured using the analog measurement pins on the mbed, as was the output voltage of the current sensor. The speed of the motors was measured by manipulating the sinusoidal signal output by the encoder in hardware to produce a digital signal, and then measuring the frequency of this signal using input capture. The relay in the vehicle powertrain energy storage system was controlled using a digital output pin, and both motor controllers were controlled using pulse-width modulated signals.

There were also bi-color power flow indicator LEDs in our system which showed the direction of the flow of energy between components, and these were controlled by digital signals from the mbed.

The detailed schematic of the system shown above can be viewed on the Upverter website (see Links section at the end).
Software

The software for the mbed consists of several files: main.cpp, mbed.h, measurement.h, motor_control.h, and power_sources.h. All of the header files simply contain function definitions, declarations of constants and variables, and pin assignments which are then used in the main file. Once declarations and definitions are completed, the PC interface is set up. Then the program enters its main function. Here, it enters a loop where each time it checks the state of the kill switch: if the switch is open it disables the motors and enters the LED demonstration mode, which simply alternates the color of the power flow indicator LEDs. Otherwise, it will run the motors once given a command. There are then two loops within the main loop: one which is checked every second, and one which is checked every 1/100 of a second. In the slower loop, voltages are checked. If the serial interface is set up (as opposed to the MATLAB interface), then certain values will also be sent to the terminal; this was largely used for debugging. In the faster loop the current is measured, the duty cycle of each motor is set, the speed is checked, and the indicator lights are updated to reflect the current state of the motors.

For the MATLAB interface, the initial code for the GUI was auto-generated using GUIDE, and then the desired functionality was added. The main file for this interface is protodrive_io.m, and it drew from function declarations in read_states.m. This file
has some initializations in the beginning, all of which come from the auto-generated GUIDE code. On the GUI, there is a drop-down menu which contains options for sample, fictitious drive cycles. The user would then select one of these cycles, and then hit the play button. This calls a special type of function, called a callback function, for this element, which then runs the large portion of the code. When this function is called, the serial connection with the mbed is set up, and the variables that will be controlled by MATLAB are linked to this program. Variables created on the mbed, such as the PWM duty cycle, are tied directly to variables in Matlab using the mbed’s RPCInterface library. In this way, when a variable is changed in Matlab, it is changed on the mbed as well. Then, the program determines which drive cycle has been selected via a switch statement. The PWM duty cycle variables are changed in Matlab (and also on the mbed) as dictated by the selected drive cycle, and both input variables and output measurements are plotted the PC’s display.
Design

Mechanical
Protodrive is a prototyping and simulation platform designed to be used in labs and educational environments. The mechanical design incorporates the following features which cater to the end user:

- Modular
- Small scale
- Rapid prototyping friendly
- Portable

The entire system is neatly laid out on a 14”x22” acrylic board, with components placed in logical positions to reduce the footprint of the board and the length of connection lines. The positioning of components also allows easy connectivity between components, and makes the difference between the two powertrains visually obvious. The “vehicle” powertrain is located on the left and is clearly separated from the “load” powertrain on the right. A breadboard is located in the center to allow the integration of a microprocessor and allow for easy rapid prototyping. Placing the breadboard between the two powertrains allows wires from either powertrain to be plugged into the breadboard, and creates a physical separation between the powertrains. The motors are placed at the top of the board, the furthest point from the user, to prevent any accidental contact with the spinning components. The “vehicle” and “load” labels on the board clearly show which powertrain is which, but also indicate to the user the correct positioning of the board. In the correct position, all components lie within easy reach and the board will easily fit on most desk tops.

The board includes a number of convenient features. A handle is integrated at the top of the board, which makes it easy to transport. All components are designed to be secure when the board is in the vertical position. The board layout and additional features are shown and discussed below.
Modular Sections

The board contains three quickly removable acrylic mounting plates. There are two “power supply” plates which each contain a battery, motor controller and perfboard, and the third plate contains the coupled motors. By isolating these three groups of components, it makes it easy to replace powertrain components by simply laser cutting a new mounting plate and swapping it into the Protodrive board. This is faster and wastes less material than having to cut a whole new board each time a component is replaced. This modularity actually proved very useful during testing - we burnt out a set of motors, and the motors available for replacement required a completely different mounting design. It was very convenient to be able to simply replace the motor mounting plate, as opposed to cutting an entire new board for the whole system, then having to disconnect and reconnect all the other components.
Clockwise from top left: The 3 removable mounting plates, quick-release cams, the board with all three plates removed (the old motors are shown in the top left), new motors securely locked into the board on a new mounting plate.

**Power flow indicator LEDs**

To allow observers to quickly visualize the flow of power within the powertrains, LEDs indicating the direction of power flow are included. The LEDs are bi-color and can either display orange or green. In front of the motors, LEDs are arranged to form two arrows. If the vehicle powertrain is in drive mode, then power will be flowing out of its batteries and will be used to charge the load batteries. In this case, an orange arrow is displayed pointing to the right, indicating that power is flowing to the load. If the vehicle is in regen mode, power flows from the load batteries and will charge the vehicle batteries. In this case, a green arrow is shown pointing to the left.
Each battery and the supercap also have an LED indicating the direction of current flow to or from that particular device. When the LED is orange, it means that current is draining from the device, and when it is green, it means the device is being charged. So, for example, in Regen mode, the vehicle battery or supercap LED will be green, while the load battery will be orange.
**Safety and charging switches**

The board contains a switch in the center of the board, at the closest point to the user. This switch can be configured for any function, by plugging it into the breadboard. We used the switch as a whole circuit kill switch, which would disable all devices if it was turned off. This created a layer of safety and allowed us to quickly kill power to the components if anything behaved in an unexpected manner.

Each power supply plate (consisting of the battery, motor controller and perfboard) also contains a switch that is connected to the batteries. The switch performs three functions. When the batteries are in use, they are connected in series however, when they are charging, they need to be connected in parallel. When the switch is in the on position, it connects the batteries in series so that they can be used to drive the motor. When the switch is in the off position, it disconnects the batteries from each other, making sure that no current can flow through the battery pack. Additionally, by attaching leads to the terminals of the switch, the batteries can then safely be connected in parallel for charging (when the switch is off).

**Electrical (Hardware)**

All the powertrain components and electronics are designed to operate at a low voltage (less than 12V). This makes it possible to use common low voltage, low current lab equipment with the system like oscilloscopes, power supplies, and multimeters. Although the final implementation of a powertrain on Protodrive should not require this equipment, during the prototyping stage it may be useful, for example, to verify a current measurement, power a motor from a DC supply, etc. It also keeps the cost of components down, and prevents having large components that occupy an unnecessary amount of space.

The vehicle section of Protodrive was designed to represent current commercial EV powertrain architectures, with a Lithium-Ion battery pack powering an electric motor. Additionally, a supercapacitor has been incorporated in the hardware design, giving the option to experiment with a battery-supercapacitor energy storage architecture that is showing promise for future EVs. In this architecture, either the battery or the supercapacitor can provide or collect energy, and it is possible to rapidly switch between the two. This architecture is advantageous because it combines the high energy density of a battery with the high power density and rapid charge/discharge characteristics of a supercapacitor. In order to connect one of these at a time to the motor controller, a single pole double throw relay was used. When no current was flowing through the coil of the relay, it would connect the positive terminal of the battery to the positive input of the motor controller, and when there was current through the coil the positive terminal of the supercapacitor would be connected to the positive input of the motor controller. The supercapacitor and batteries shared a common negative terminal, which was
connected to the negative input of the motor controller. The current sensor for the drive section was connected between a terminal of the motor controller and the appropriate terminal of the motor, and in this way directly measured the current through the motor. This sensor gave an analog output voltage, which was then read directly by the mbed.

For the other voltage measurements in the system, the voltage could not be fed directly to the mbed’s analog inputs because the voltages that were being read were greater than 3.3 V, the input rating for these pins. So, these voltages were scaled down to \( \frac{1}{3} \) of their original voltage using a voltage divider, and then put through an operational amplifier in a follower configuration which buffered the input to the microcontroller. This scaling was then compensated for in software.

As previously mentioned, the mbed interfaced with the motor controllers via a pulse-width modulated 3.3 V signal. This implementation was chosen because it allows for the use of the analog input mode on the motor controller, the simplest of the available modes. This simplicity was prioritized because of the time constraints on this project. The disadvantage to this mode is that the motor controller reads in the filtered PWM signal as a percentage of five volts and applies this percentage of the battery or supercap voltage the terminals of the motor. Since the maximum voltage the mbed can supply is 3.3 V, one cannot access the full voltage of the batteries. However, it should be noted that our original motors were rated to 6V, while the battery was 7.4V and the supercap 9V. This meant that we would have to scale down the input voltage demand to the motor controller anyway, so having the 3.3V limit on the PWM signal helped to make sure that we stayed below the rated 6V of the motors. Given more time, some other input method such as serial would have offered more precise control.

In order to incorporate the indicator LEDs into this system, we had to find a way to power them with an external source while controlling them with the mbed. This is due to the limited current that the mbed can source; since all of the LEDs in each group were connected in parallel, a relatively large amount of current was required. To do this, each group of LEDs was controlled with a discrete pnp bipolar junction transistor. The LED was connected to the emitter of a transistor in series with a current-limiting resistor. The anode of each LED was connected to the resistor, and all of the cathodes were shorted and connected to ground. The collector of the transistor was connected to a 5 V rail which could provide sufficient current and voltage to properly illuminate the LEDs. Once properly biased, the LEDs would light up when the output pin from the mbed connected to the gate of the transistor was pulled low, and the LEDs would turn off when that pin was pulled high.
The encoder on the motors that were used in this system output a sinusoidal waveform, the frequency of which was 8 times the mechanical frequency of the motor. In order to measure this frequency, this sinusoidal signal was put through an optoisolator. This is an integrated circuit which contains both an LED and a photodiode. So, when a sufficient positive voltage is applied across the terminals of the LED, it will light up, which will allow current to flow through the photodiode. This component was used to convert the sinusoidal waveform output by the encoder into a digital signal. The sinusoidal signal was applied across the terminals of the LED, which would light up only when the signal was above a certain positive threshold voltage. This not only rectified the signal, but turned it into a digital signal. A 3.3 V rail was applied to the positive end of the photodiode, and a resistor was connected between the other end and ground. So, when current would flow through the photodiode (when the LED would light up), there would be very nearly 3.3 V across this resistor. A digital pin of the mbed was then connected to the node where the resistor and diode were connected. The end result was a digital “1” at the input pin whenever the voltage in the sinusoid was above a certain threshold and a digital “0” otherwise. In turn, the time between the rising edges of the “1”s was the period of the signal, and this could be measured using input capture. From this value one could calculate the frequency at which the motor was spinning.

Software

One aspect of the software design for Protodrive which proved to be extremely helpful was the combined use of `#define` and `#ifdef` statements. In order to allow for various modes of operation and rapid switching between them, many portions of our code were enclosed in `#ifdef` statements. These portions of code would only compile if the appropriate variables were defined. In this way we were able to avoid the deletion or cutting and pasting of large chunks of code while still preventing it from taking up memory and processing power on the mbed. Such statements were included to switch between the MATLAB and serial interfaces, only one of which could be used at a time; to switch into a debug mode where the motors were simply given constant loads; and to switch on sections of the code which enabled the supercapacitor, which were written but ultimately not used in our demonstration.

The communication with MATLAB was established by using a library for the mbed called RPCInterface. This library provides functions which allowed for the direct linking of variables between the mbed and MATLAB; in other words, when a variable was changed in MATLAB, a signal was automatically sent to the mbed to change this variable. So, when a drive cycle was selected in the MATLAB interface, the appropriate values were calculated in MATLAB and sent to the mbed. The MATLAB program would then read the values of certain variables and plot them. However, if this project is to be taken further, the calculation and setting of these variables should be done on the mbed, and MATLAB should only send a start signal, read
the values, and plot them. This would make the whole system more efficient, as it would reduce the amount of time spent passing variables back and forth over USB. It would also make the MATLAB program run faster and make better use of the mbed’s fast clock.

**Testing and Evaluation**

As this system was constructed, individual subsystems were tested by isolating them and regulating inputs to those systems. The best of example of this was the testing which was done to verify that the analog inputs to the mbed were working properly. In order to do this, we used `#ifdef` statements to add sections to our code that could easily be prevented from compiling once the desired functionality was verified. In these sections of our code, we printed to a terminal the values being read. Then, we applied a voltage from a regulated source to these pins to make sure that we were reading the right values. In this way we verified individual subsystems which made it possible to isolate and correct issues in other subsystems, like the current sensor and the batteries.

This stepwise verification of subsystems led to the verification of the complete system. We can say definitively that our system successfully controlled both motors separately, in two directions, and with regenerative braking. More meaningfully, we have created a system where the full speed and torque range of the motor can be accesses by controlling the terminal voltages of two coupled motors. However, we have failed to achieve sufficient accuracy in our measurements to put this into a control loop as we had originally intended. Unfortunately, the current measurement subsystem failed to function properly due to hardware limitations. The motors we had initially started using took much more current, and so the current sensor was rated for a different range and had very noisy signal in the range that we were attempting to measure with our final motors. We also were unable to fully integrate the supercapacitor and use it to absorb and supply energy.
Conclusion

This project taught us a great deal about interfacing components at a system level. Our system involved a large amount of both mechanical and electrical equipment, and the electrical systems were a mix of analog and digital components, at both high and low power as well as varying voltages. This made interfacing a challenge and we gained significant experience overcoming the many issues we faced. In this case, our mixed backgrounds were very helpful because we were able to help one another fill in some gaps in our individual knowledge. However, even with our combined knowledge, we consistently ran into parts of our system which did not act as we expected. From biasing BJTs to reading in analog signals, we regularly had to revisit our approach to a problem because of some detail that had not been originally considered. One important lesson we learned from this is that things will usually take longer than expected because all of these unexpected details. We also learned that is sometimes necessary to make approximations, but is still important to do this carefully. Many of the areas we had to revisit were due to oversimplifications in our assumptions about our problem.

In the end we were able to create a small scale model of an electric vehicle powertrain in which the motor speed and torque can be controlled. The motor speed and current, and battery voltages can also be accurately measured, and the direction of the flow of energy between the vehicle powertrain and the load powertrain can also be detected. We met all of our stated requirements and partially met many of our reach goals:

- The Protodrive board is modular and rapid prototyping friendly, and the power flow indicator lights make it intuitive to understand how energy is transferred through the system.
- A control system to follow a federal drive cycle is well under development and will be completed within a few weeks after this report is submitted.
- The hardware for a novel powertrain featuring a supercapacitor was designed and integrated into the board. However, we were unable to get this system to behave has planned by demo day.

From here, this project can be taken many different directions. It can be used to rapidly model new vehicle architectures cheaply, efficiently, and safely; it can be mapped to real vehicle parameters so that the efficiency of larger vehicles can be more realistically and more efficiently estimated; and can be used in conjunction with real map data to calculate the energy required to travel real-life routes. While not all of our goals for this project were fully realized,
we view it as a success because we have successfully provided a highly useable platform for further research.

Demo day: Dr. Rahul Mangharam, William Price, Dean Eduardo Glandt and Andrew Botelho

Links

Blog: protodrive.blogspot.com
Code (on mbed Website): http://mbed.org/users/protodriveev/programs/
Schematic: http://upverter.com/Wprice/df7c3a41d0992cc6/Protodrive-schematic/