Final Report

Team HEV

William Carson, Siyuan Dai, Anne-Marie Krishnan, Matthew Zenz

Table of Contents

Introduction

The goal of our project was to hybridize an electric truck for the Metropolitan Water Reclamation District of Greater Chicago. This final report will include a description of the problem, a high level description of our solution, and a discussion of how well our design met our expectations. It will then cover a detailed project description of each subsystem and our system integration testing methods, followed by a user's manual.

Problem Statement

The Metropolitan Water Reclamation District of Greater Chicago currently owns a fleet of purely electric trucks. These trucks are used for projects requiring driving ranges less than 30 to 40 miles per charge. However, they would like these trucks to have the capabilities to perform jobs that require driving distances further than 40 miles. In order to provide this capability, they have asked us to transform this purely electric vehicle into a hybrid electric vehicle. The goal is to reach a city mpg rating between 60 – 80 mpg (50 mpg minimum). They would also like to see an increase in speed capabilities—from 25 mph to at least 35 mph. Our goal is to reach a maximum speed of 40 mph. An additional complication to this situation is that they require the option to undo any changes we make during the hybridization process if they choose to revert to the purely electric version.

System Requirements

In order to solve the problem, we determined the following system requirements:

1. Power source capable of providing between 55-80 VDC to the motors.
    - Solution: Ultracapacitors
2. Method to regulate the voltage on each ultracapacitor in order to keep them each within the safe operating range.
    - Solution: Three diode balancing circuit
3. Power generator to keep the above power source within that operating voltage.
    - Solution: Diesel Generator
4. Method to increase the current feeding into the ultracapacitor stack.
    - Solution: Transformer
    - *Over the course of the design process, we determined that the 2:1 Transformer we were given did not provide a high enough current to the ultracapacitor stack. Thus, we determined that a 3:2 autotransformer would allow higher currents to flow in parallel through the transformer coils into the ultracapacitor stack.*
5. Method to convert the AC current from the generator to a DC current into the ultracapacitor stack.
    - Full wave bridge rectifiers.

- *In the late stages of the design process, we discovered that the bridge rectifiers would overheat after extended periods of use, causing the system to fail. We thus had to modify our design to include heat sinks for each rectifier to prevent this from occurring.*

6. Circuitry to determine when the generator would need to be turned on and off.
    - Solution: Microcontroller interfaced to the ultracapacitor voltage reading by a resistor divider circuit
7. Circuitry to turn the generator on.
    - Solution: Microcontroller interfaced to MOSFET circuitry, current sensor, and temperature sensor
    - *In the beginning of the design process, we had decided that we would leave the starter on for a set amount of time in order to start the generator. However, we later determined that it was ideal to design a circuit for which the starter signal was only on for the necessary amount of time. Thus, we utilized a current sensor in order to determine when the generator had started, allowing us then to turn the starter signal off. Additionally, we discovered that the diesel generator cannot be turned on right away in the event of cold weather conditions. By attaching a temperature sensor to the glow plugs, we are able to monitor this, allowing the system to warm up before turning on the generator.*
8. Circuitry to turn the generator off.
    - Solution: Microcontroller interfaced to MOSFET circuitry
9. Method to collect and store data for research purposes.
    - Solution: GPS device, SD card, 2 current sensors reading from ultracapacitor bank (into the motor controller) and transformer
10. Method to have some sort of interface with the end user.
    - Solution: LCD

High Level Description of Solution

Our solution involves the use of the following components:

- Electric truck
- Diesel generator, rated at 7 kVA
- 5 kVA transformer
- Bridge rectifiers
- Ultracapacitor bank, consisting of 34 ultracapacitors
- Balancing circuitry
- MOSFET circuitry
- Current sensors
- Temperature sensor
- LCD
- GPS
- SD card

Our truck is designed as a series hybrid. This means that the engine is used to charge the ultracapacitors, which in turn power the driving motors. In a parallel hybrid design, both the engine and the battery source power the driver motors.

Before the current from the diesel generator can flow to the ultracapacitor bank, we needed to increase the current and convert it to a DC signal. To increase the current, we utilized a 3:2 autotransformer. Three 10-gauge wires are used to carry current in parallel to three bridge rectifiers. These rectifiers convert the AC signal to the DC signal required to charge the ultracapacitor bank.

Within the ultracapacitor bank, we designed a balancing circuitry to minimize the variation of the voltages on each individual ultracapacitor. This consisted of three standard diodes, which dissipated more current at higher voltages than at lower voltages.

We designed MOSFET circuitry that will be described in detail later to control the generator along with the microcontroller. After determining a safe operating voltage range for the ultracapacitor bank, we programmed the microcontroller to send a 4-5 V signal to this circuitry to send the on/off and starter signals to the generator. We use a temperature sensor to determine whether the glow plugs are warm enough to start the diesel engine. If it isn't, we warm the glow plugs before starting the generator. We also use a current sensor to monitor the starter current; when this drops, we know that the starter signal can be switched low (0 V). This has two primary advantages over holding the starter signal high for a fixed amount of time (which would be hardcoded):

1. This ensures that we do not unnecessarily damage the starter by holding the signal high for longer than needed to start the generator.
2. This ensures that we have the starter on long enough to actually start the generator. This will likely take longer in the winter, and we would risk damage due to the preceding reason if we hardcoded the starter signal to stay high long enough to start the generator in the winter every time we start the generator.

We have two more current sensors monitoring the current going into the driving motors (out of the ultracapacitor bank) and out of the transformer for data gathering purposes. We also have a GPS capable of reading the latitude, longitude, velocity, and altitude of the vehicle. An SD card connector has been mounted to the circuit board to store this information. Lastly, our truck is LCD capable to provide an interface to the end user. Right now, we have code to display the voltage level on the ultracapacitor bank, but we do foresee other uses for the LCD to provide useful information to the end user.!

Design Expectations

Overall, we feel as though our project met most of our expectations. Our truck meets the fundamental functionality objectives of fuel efficiency, driving range, and increase speed capabilities. The biggest challenge in this project was getting the

truck to drive. The generator current successfully runs through the transformer and the full wave bridge rectifiers to charge the ultracapacitor bank, and the control circuitry successfully controls when the generator turns on and off based upon the voltage level on the ultracapacitor stack. Additionally, the current sensor monitoring the starter current allows us to correctly determine how long to send the starter signal to the microcontroller. We were also able to ensure using the temperature sensor that the glow plugs are warm before we send the starter signal.

We were unable to get the LCD, GPS, and SD card completely functioning, however. The code has been written for each of these components, but we could not get it debugged in time due to issues with getting the truck to drive in the late stages of the design process. We felt that getting the truck to drive was a priority, and so we decided to divert our resources to making that aspect of the project function correctly, which unfortunately left too few resources to fix these programming issues.

Detailed Project Description

This section details how the entire system works together—via the system block diagram—as well as a description of each subsystem. This description will include a schematic or flow chart, the subsystem function, how it interfaces with other subsystems, and how the subsystem was tested. Furthermore, it will explain how we came to make the engineering decision required to design the subsystem.

System Theory of Operation

We used a diesel generator, rated at 7 kVA, to serve as our power generator. This then fed into the transformer, which we converted into two 3:2 autotransformer in order to generate the required current level into our ultracapacitor bank. The current coming out of this transformer is routed into three bridge rectifiers to convert the current from AC to DC. This is then fed into the ultracapacitor stack, and is referred to as the “charging circuitry.”

Before we could design how to control the generator, we needed to determine how we could read the voltage of the ultracapacitor stack, as controlling of the generator is directly related to this voltage level. We take the 60-80 V signal from the ultracapacitor bank using a voltage divider.

In order to control the generator, we utilized MOSFET circuitry, temperature and current sensors, and a microcontroller. The MOSFET circuitry was designed to send 12 V signals to the on/off circuit, the starter circuit, and the glow plugs on the generator. When the microcontroller reads that the ultracapacitor stack has reached either the low or high threshold, it sends a 4-5 V signal to the MOSFET circuitry to turn on and off the generator.

The data collection circuitry consisted of a GPS device, two current sensors, and SD card. The GPS unit is designed to capture the latitude, longitude, velocity, and

altitude of the vehicle. The current sensors monitor the current coming out of the ultracapacitor bank and the transformer. This information can be stored on the SD card for research purposes.

We implemented an LCD to allow for user interface with the system. This LCD is currently programmed to display the voltage level on the ultracapacitor stack. At this stage of the design, we found it important for the user to have this information to ensure that the control circuitry was working properly—i.e. that the generator was turning on and off at the correct times.

System Block Diagram

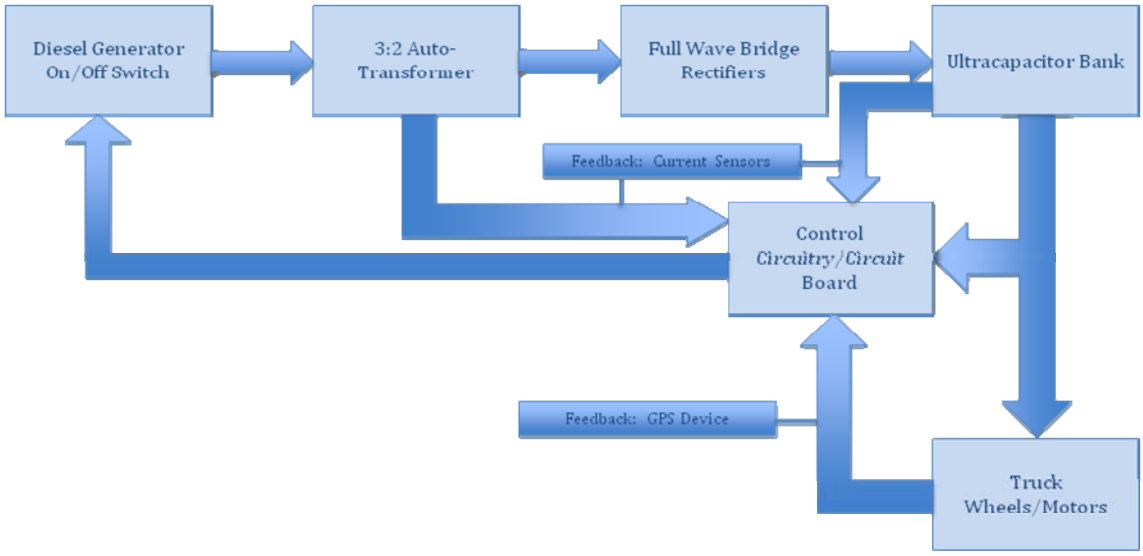

Figure 1

Detailed Operation

This section will detail the design of each subsystem.

Diesel Generator On/Off Switch

Schematics:

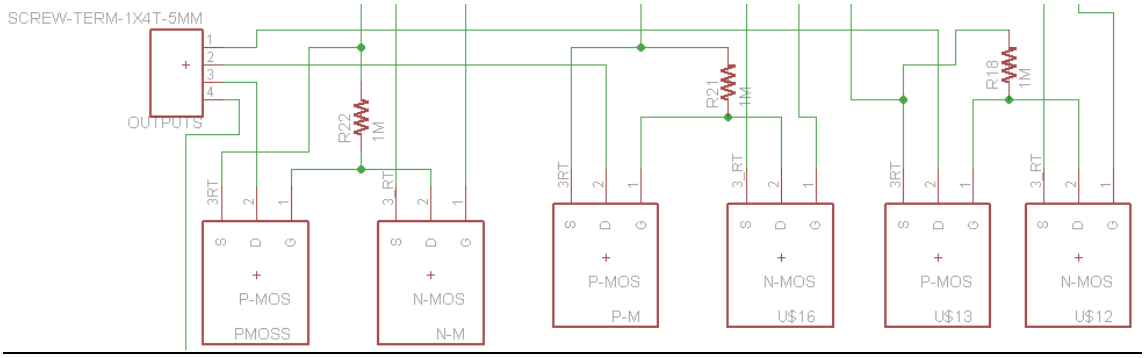

Figure 2: This is the schematic of the MOSFET circuitry used to control the generator.

Description:

The 7 kVA diesel generator serves as the power generator for the entire system. Because we have designed a series hybrid, the generator is used to charge the ultracapacitor bank, and not also to provide power to the driving wheels.

We used MOSFET circuitry to control the electric signal to the generator's fuel valve. When the microcontroller sends a 3 V signal to the P-type MOSFET, the output from the N-type MOSFET is high, which is the 12 V passed from the battery of the generator. A 12 V signal opens the fuel valve, allowing diesel to flow and run the generator. When the microcontroller sends a 0 V signal to the P-type MOSFET, the output from the N-type MOSFET is low, which is 0 V. A 0 V signal from this circuitry closes the fuel valve of the generator, thus turning it off.

As we will explain in a later section, we determined that the generator should be turned on when the voltage level on the ultracapacitor bank reaches 62 V and should be turned off when the voltage level reaches 79 V.

In the event of cold temperatures, we have designed a circuit to ensure that the glow plugs were warm before the generator starts. A temperature sensor connected to the glow plugs acts as a variable resistor, which we have calibrated to determine when the temperature is too low and high enough to start the generator. The resistance of the sensor decreases 0.1 ohm for each degree Celsius the temperature increases. The glow plugs turn on when the temperature is below freezing, and the generator will not start until they have reached above freezing temperatures.

Interfaces:

The diesel generator interfaces with the microcontroller, temperature sensor, and transformer. We connect to the microcontroller using 20-gauge wires, capable of handling the current generated by the control circuitry signals. The temperature sensor is connected to one of the globe plugs using a ring connector and to the microcontroller using a 22-gauge wire. The generator connects to the transformer using two AC wall plugs on the front panel of the generator. We had originally decided to tap directly into the hot wire on the generator to control the on/off switching. However after we determined that the benefit of the more secure connection did not outweigh the potential cost of cutting into the generator wiring, we decided to utilize these AC wall plugs on the front panel.

Software:

The code listed below is used to control the fuel valve on/off signal, the starter signal, and glow plug warming (when necessary). We chose to program in C because our primary coder, Siyuan, felt most comfortable with coding this language and it is compatible with the microcontroller we decided to use in our design.

```
void main()
{
```

```
	// initialize the microcontroller and SD card
	initialize_microcontroller();

	// check temperature
	temp_check();

	while(1)
	{
		// calls functions to do all A/D conversion routines
		ucap_v_AD_conv();
		starter_motor_current_AD_conv();
		ucap_stack_current_AD_conv();
		driving_motor_current_AD_conv();
		temp_sensor_AD_conv();

		// a ucap_v of greater than 154 (corresponding to 62 V) should
send nothing
		// 49 corresponds to a voltage of 20
		if (ucap_v > 154)
		{
			// leave the generator alone
			starter = 0;
			on_off = 0;
		}

		else if (ucap_v <= 154)
		{
			// start the starter motor to initialize generator
			starter = 1;
			on_off = 1;
			delay_ms(50);
			starter_motor_current_AD_conv();

			// this while loop waits for starter motor current to go down
			while (starter_motor_current > 150)
			{
				// keep polling for starter_motor_current
				// once the current drops, will jump out of loop
				// call all A/D conversion functions
				ucap_v_AD_conv();
				starter_motor_current_AD_conv();
				ucap_stack_current_AD_conv();
				driving_motor_current_AD_conv();
				temp_sensor_AD_conv();
			}
			starter = 0; // turn off starter once current goes down
```

```
                        // 198 corresponds to a stack voltage of 80 V
                        // we would like about 86 V across the stack before switching
off though
                        // 212 corresponds to a voltage of 86 V
                        // jumps out of loop once the ultra-cap voltage gets too high
                        // 62 corresponds to 25 V
                        while (ucap_v < 212)
                        {
                                // call all A/D conversion functions
                                ucap_v_AD_conv();
                                starter_motor_current_AD_conv();
                                ucap_stack_current_AD_conv();
                                driving_motor_current_AD_conv();
                                temp_sensor_AD_conv();
                        }
                        on_off = 0;
                }
                else
                {
                        // do nothing
                }
        }
}
```

The following code is the function programmed to check the temperature of the glow plugs:

```
/*checks temperature to determine whether or not to turn on the glow plug*/

void temp_check(void)
{
        temp_sensor_AD_conv(); // first get temp_sensor value
        while (temp_sensor <= 22)
        {
                on_off = 0;
                starter = 0;
                glow_plug = 1;
                temp_sensor_AD_conv();
                ucap_v_AD_conv();
                starter_motor_current_AD_conv();
                ucap_stack_current_AD_conv();
                driving_motor_current_AD_conv();
                temp_sensor_AD_conv();
        }
        glow_plug = 0;
        return;
```

}

Subsystem Testing:

In order to test that the generator turned on and off at the right times, we began with testing the voltage levels coming out of the control circuitry. Simulating the signal that would go to the microcontroller when the ultracapacitor bank voltage level was too low, we determined that the signals from the microcontroller to the starter and the fuel valve were the required 12 V signals to turn on the generator. Similarly, we simulated the signal sent to the microcontroller when the voltage level on the ultracapacitor bank was too high and monitored the microcontroller's output signal to the generator's fuel valve. The signal was the correct 0 V.

Once we determined that these signals were correct, we connected the generator through the autotransformer and bridge rectifiers to the ultracapacitor bank. We started with the ultracapacitor bank voltage level just above 62 V. After resetting the microcontroller, we caused a spark across the ultracapacitor stack in order to discharge some of the voltage. This caused the voltage to reach a level lower than the low threshold, and the generator successfully turned on. We allowed the generator run while we monitored the voltage level on the ultracapacitor stack. This also allowed us to ensure that the generator was able to charge the ultracapacitors. When the voltage level rose to 79 V, the generator turned off.

3:2 Auto-Transformer

Description:

The function of this subsystem is to increase the current from the generator to a level capable of charging the ultracapacitor bank at a desirable rate. As described earlier, we wired an originally 2:1 transformer into two 3:2 autotransformers. The 2:1 transformer was only able to provide 25 A at 50 V, corresponding to just over 1kW of power.

We applied the idea of an autotransformer because an autotransformer can be wired so that a variety of voltage ratios are possible. We chose to wire it as a 3:2 autotransformer as a test in order to see if we could get enough current through the ultra-capacitors. Since the transformer has 4 coils, we were able to take 2 coils each for 2 autotransformers. This allows our high current levels to flow in parallel. Through the test that we ran with this configuration, we found that the ultracapacitor stack can be charged with 80 A of current at 50 V, which corresponds to 5 kVA. This power level is sufficient to drive the wheel motors.

Interfaces:

The transformer interfaces with the generator and the full wave bridge rectifiers. The transformer connects the generator via two AC wall plugs on the front panel of

the generator. It connects to the full wave bridge rectifiers through three 10-gauge wires running in parallel carrying up to 30 A each.

Subsystem Testing:

Testing of the transformer was relatively easy. We plugged the transformer into an AC outlet and verified that the output was around 80 V, corresponding to a 3:2 ratio using a 120 V input.

Full Wave Bridge Rectifiers

Description:

The function of this subsystem is to convert the AC current from the generator to the DC signal required to charge the ultracapacitor bank. It consists of three 600 V/50 A full bridge rectifiers connected in parallel. They take an AC input voltage from the low end of the transformer and output a fully rectified DC signal. Initially, we picked 25 A rectifiers; however, after realizing that a total of 75 A would not be enough for our application, we changed the design to incorporate 50 A rectifiers. This allows us to have a total current of 150 A, which is more than enough for our application. We also used heat sinks with each rectifier to prevent overheating.

Interfaces:

The rectifiers interface with the transformer and ultracapacitor bank. Because we needed the rectifiers to handle at least 100 A, we chose to use 3 10-gauge wires in parallel, each capable of handling 35 A. When making the connections to the wires, we decided to use screw-on connectors instead of crimp-on connectors. We did not choose crimp-on connectors because it is a permanent connection, which makes future modifications to the system almost impossible. We chose screw-on connectors because they are robust and well-insulated.

Subsystem Testing:

In order to test this subsystem, we first connected the full wave bridge rectifiers to the transformer, leaving the circuit open. Upon plugging the transformer into an AC outlet, we measured the output of the rectifiers. We saw 87 VDC coming from this measurement, verifying that the rectifiers successfully converted the AC signal from the transformer to a DC signal capable of charging the ultracapacitor bank.

Ultracapacitor Bank

Charging Diagram:

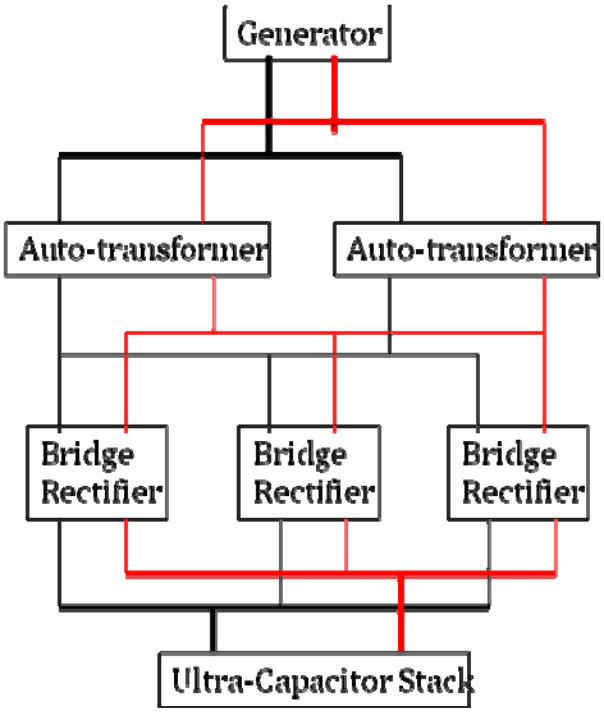

Figure 3: The black wires are neutral lines, and the red wires are hot lines. The thick wires are 3-gauge, and the thin wires are 10-gauge.

Description:

In our series hybrid design, the ultracapacitor bank is the sole driver of the truck motors. In order to operate the truck, we needed the bank to hold a consistent charge of at least 50 V and provide 80 A in order to create 4kW of power. Each ultracapacitor can hold a maximum of 2.7 V. Based upon our design requirements, we determined we needed 34 of the 3000 F ultracapacitors connected in series.

In order to maintain a somewhat consistent voltage level on each individual capacitor, we designed a balancing circuit. In between each pair of ultracapacitors, we connected three generic 1N4001 diodes in series. This circuitry dissipates large amounts of current—about 0.7 A—when the voltage of an ultracapacitor reaches a certain high level (like 2.7 V). In contrast, these diodes dissipate very low current—just under 50 mA—at lower voltages (like 2.5 V). The 1N4001 diodes provide this non-linear characteristic we desired, unlike the power diodes we originally tested. Balancing the ultracapacitor stack minimizes the potential for overcharging and possible explosions of the ultracapacitors.

Interfaces:

The ultracapacitor bank interfaces with the full wave bridge rectifiers(3 10 G merge into 1 4 G each for plus and minus), the driving motors (3 G wires), and the microcontroller (via voltage divider using 22 G wires). Connection info here.

Subsystem Testing:

As mentioned above, we tested that the ultracapacitors could be charged by the generator through the transformer and the bridge rectifiers during the testing of the generator and control circuitry. We tested the balancing circuitry by monitoring the voltage levels on each ultracapacitor during multiple phases of the design process. The ultimate test—ensuring that the ultracapacitors could drive the truck motors—occurred in the late stages of the design process when we were able to drive the truck using the entire system.

Control Circuitry

Schematic:

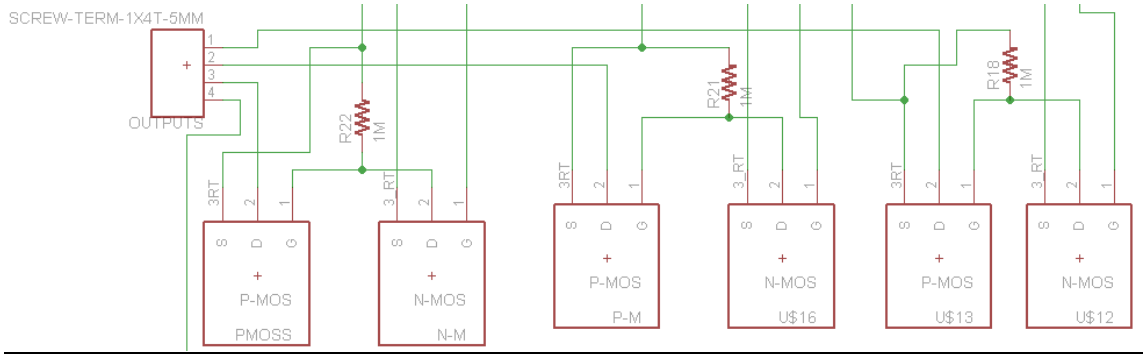

Figure 4: This schematic is the same as the above On/Off circuitry. The controls for this truck use identical MOSFET switching circuits.

Programming Flowchart:

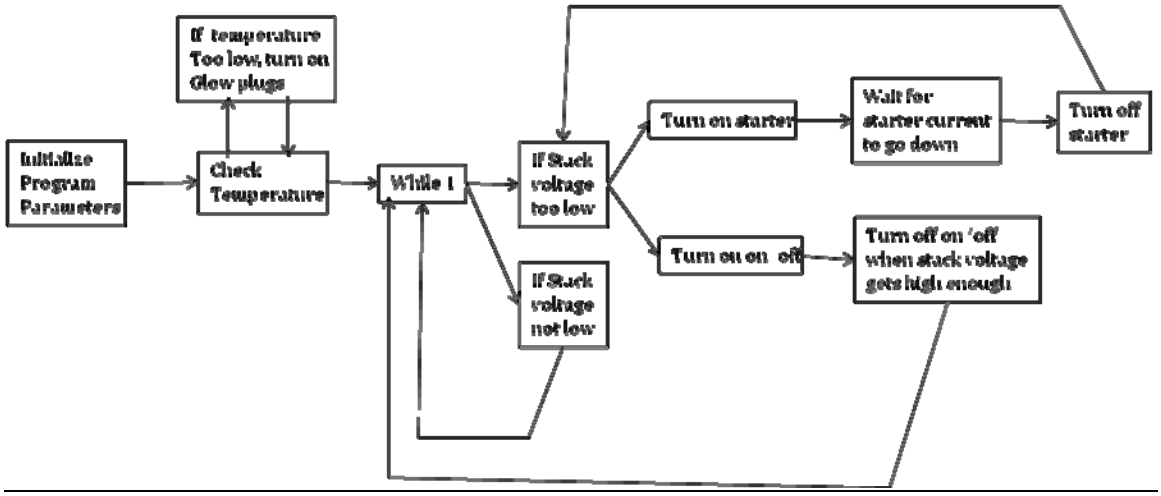

Figure 5: This is the central part of the program; the voltage level on the ultracapacitor stack controls when the generator turns on and off. The temperature sensor determines whether that can happen immediately or if the glow plugs need to be warmed first.

Code: The code for this function is listed above within the diesel generator subsystem section.

Description:

The control circuitry is the automation portion of our design. We have five control inputs to a PIC18F4620 microcontroller: three current sensors, a temperature sensor, and the ultracapacitor voltage. Each sensor requires a 5 V power supply. However, our only voltage source is the 12 V battery on the generator. To step down the voltage from 12 to 5 V, we used a 5 V voltage regulator. To protect the voltage regulator, we added a capacitor to both the input and output. Next, we passed the signal through a unity gain buffer, which protects the sensors from the microcontroller sending a signal to the sensors. Additionally, the unity gain buffer helps by providing impedance protection, reducing the chance of a short circuit. The signal entered the microcontroller after passing through the unity gain buffer.

The microcontroller uses A/D conversion on the input voltage signals and outputs a 0 V or 3.3 V signal. The 3.3 V signal to the starter is time limited, only staying high at 3.3 V until the starter current sensor detects a current of under 25 A. This value is fed back to the microcontroller, which in turn set the signal low to 0 V.

For the glow plug signal, the value from the temperature sensor is fed back to the microcontroller. If the temperature sensor's voltage is over a programming threshold (corresponding to a temperature warm enough to turn on the diesel generator), it sets the glow plug signal low, from 3.3 V to 0 V, to turn the glow plugs off.

The 12 V signals were sent to the on/off circuit, the starter circuit, and the glow plugs. We used this same MOSFET circuitry to produce each of the three 12 V signals.

Interfaces:

The control circuitry interfaces with the ultracapacitor bank, the generator, and the microcontroller, as described in their respective sections earlier.

Subsystem Testing:

The MOSFET circuitry was first laid out on a breadboard. We input a 3.3 V signal to the circuit to simulate receiving a signal from the microcontroller and we used a 12 V signal from the power supply to simulate the voltage from the battery on the generator. We used a voltmeter to measure the output signal and verified that we had a 0 V output for a 0 V input and a 12 V output for a 3.3 V input. As we varied the input voltage from 0 V to 3.3 V, the circuitry correctly changed the output from 0 V to 12 V when we passed a threshold of V_{in} = 1.2-1.5 V.

After the tests were verified on the breadboard, we soldered our components onto a perf board and repeated our tests. Once this was complete, we were able to test our circuit using a signal from the microcontroller and the battery on the generator. We simulated the voltage across the ultracapacitor stack by using a variable resistor with an A/D converter for the microcontroller. With this microcontroller input and the 12 V battery source, we verified our prior results.

Following this test, we connected our current sensors and temperature sensors to a breadboard. We put a varying current of 0-2 A through the current sensor. As the current increased, the voltage reading at the output of the current sensor increased by 12 mV / 1 A. We verified the functionality of the temperature sensor by placing a heat source next to the sensor and measuring the output voltage. Using a soldering iron to heat up the sensor, we observed an increase of 11 mV / °C.

The next test we conducted was measuring the voltage across the ultracapacitors and verifying that the microcontroller's A/D converter could process the voltage and send a 0 V signal to turn off the microcontroller upon passing the upper voltage threshold of 80 V.

Upon soldering the circuit components to the final circuit board, we completed the above tests to verify that the connections were made correctly.

Data Collection Circuitry

Schematic:

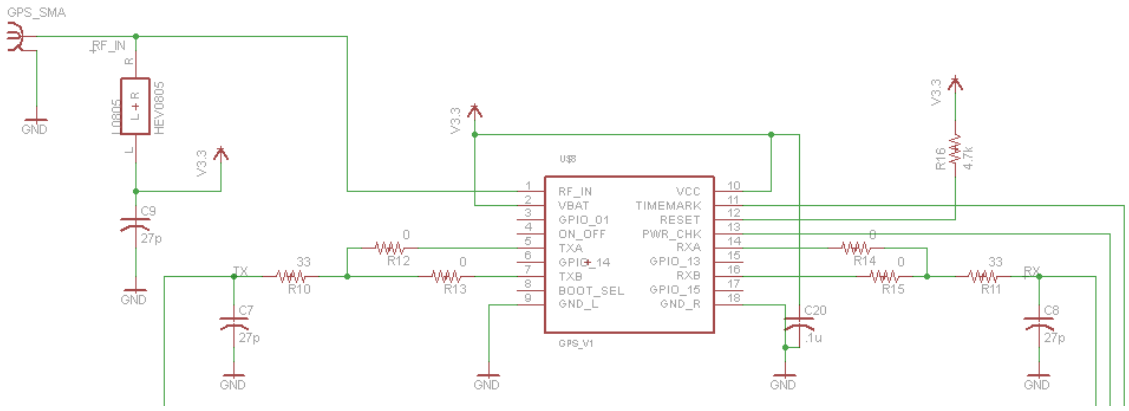

Figure 6: GPS

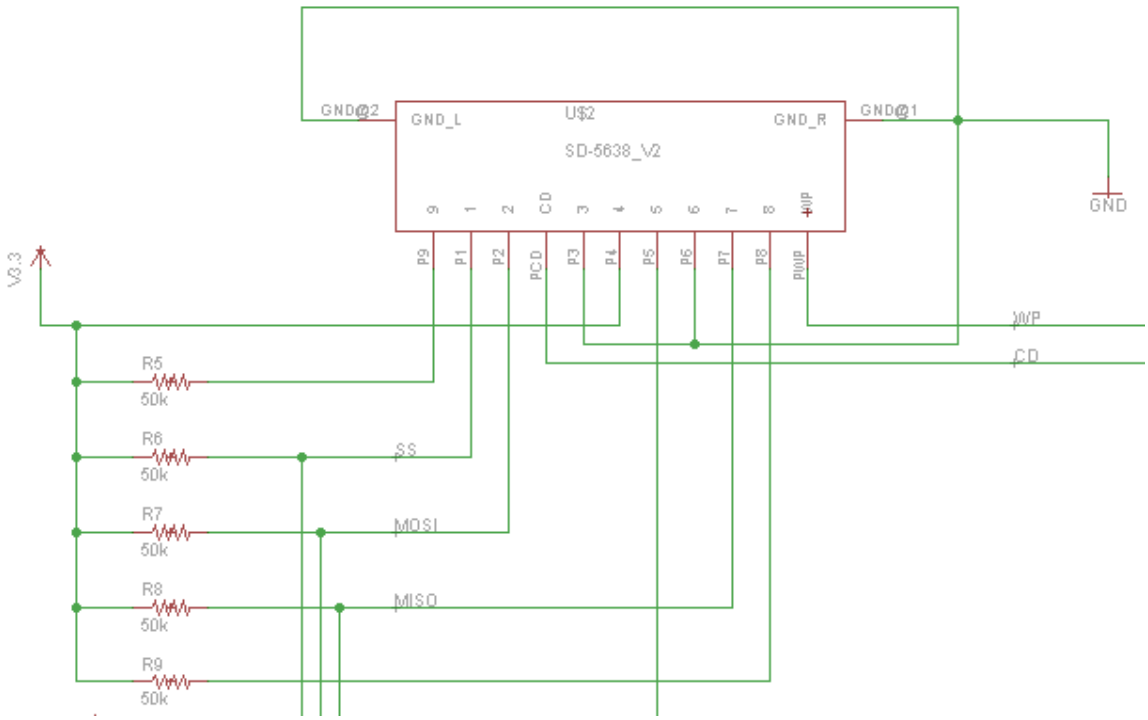

Figure 7: SD card

Description:

This circuitry is designed to provide data from the truck for research purposes. We have soldered a GPS device and an SD card holder to the final circuit board. The GPS device is designed to capture the latitude, longitude, altitude, and velocity of the vehicle and store this data on the SD card.

The current sensors read the current coming out of the ultracapacitor bank and transformer.

Interfaces:

The GPS interfaces with the SD card, both of which are soldered on the circuit board. Each current sensor interfaces with the SD card. A complete microcontroller pin listing can be found in Appendix E.

GPS Code Listing:

```
/*initialize the clock and UART*/

void GPS_init(void)
{
        // set Fosc as 8 MHz
        osctune = 0b10000000; // use 31.25 kHz device clock
        osccon = 0b11111111;
```

```
	// set UART registers
	txsta = 0b00100100;
	rcsta = 0b10110000;
	trisc.7 = 1;
	trisc.6 = 1;
	baudcon |= 0b00001000;

	// SPBRGH:SPBRG = 416 corresponds to baud rate of 4.8 k
	spbrg = 416 & 0xFF;
	spbrgh = (416 >> 8) & 0xFF;
	return;
}

/*transmit a command to the GPS*/
void GPS_transmit(unsigned char gps_command)
{
	while(!txif);
	txreg = gps_command;
	return;
}

/*send command to GPS to tell speed of vehicle*/
/*only need to call once, then sends data once a second*/
void tell_speed(void)
{
	unsigned char cmd[22];

	// want cmd to be ASCII '$PSRF103,05,00,01,01*25'
	cmd[22] = '$';
	cmd[21] = 'P';
	cmd[20] = 'S';
	cmd[19] = 'R';
	cmd[18] = 'F';
	cmd[17] = '1';
	cmd[16] = '0';
	cmd[15] = '3';
	cmd[14] = ',';
	cmd[13] = '0';
	cmd[12] = '5';
	cmd[11] = ',';
	cmd[10] = '0';
	cmd[9] = '0';
	cmd[8] = ',';
	cmd[7] = '0';
	cmd[6] = '1';
	cmd[5] = ',';
```

```
	cmd[4] = '0';
	cmd[3] = '1';
	cmd[2] = '*';
	cmd[1] = '2';
	cmd[0] = '5';
	int i;
	for (i=22;i>=0;i--)
	{
		GPS_transmit(cmd[i]);
	}
	return;
}

/*use this interrupt to receive data*/
void interrupt(void)
{
	// interrupt routine for UART receive
	char dat;
	dat = rcreg;
	// if first command char is received
	if (dat != '$' && k == 34)
	{
		// if first char isn't $, set k = 34
		k = 34;
	}
	else if (dat == '$' && k == 34)
	{
		// if first char is $, subtract one from k
		// and see the second char
		GPS_receive[k] = dat;
		k = k - 1;
	}
	else if (dat != 'G' && k == 33)
	{
		// if second char isn't G, reset k to 34
		k = 34;
	}
	else if (dat == 'G' && k == 33)
	{
		// if second char is G, subtract one from k
		// and see the third char
		GPS_receive[k] = dat;
		k = k - 1;
	}
	else if (dat != 'P' && k == 32)
	{
```

```
            // if third char isn't P, reset k to 34
            k = 34;
    }
    else if (dat == 'P' && k == 32)
    {
            // if third char isn P, subtract one from k
            // and see the fourth char
            GPS_receive[k] = dat;
            k = k - 1;
    }
    else if (dat != 'V' && k == 31)
    {
            // if fourth char isn't V, reset k to 34
            k = 34;
    }
    else if (dat == 'V' && k == 31)
    {
            // if fourth char is V, subtract one from k
            // and see the fifth char
            GPS_receive[k] = dat;
            k = k - 1;
    }
    else if (dat != 'T' && k == 30)
    {
            // if fifth char isn't T, reset k to 34
            k = 34;
    }
    else if (dat == 'T' && k == 30)
    {
            // if fifth char is T, subtract one from k
            // and see the sixth char
            GPS_receive[k] = dat;
            k = k - 1;
    }
    else if (dat != 'G' && k == 29)
    {
            // if sixth char isn't G, reset k to 34
            k = 34;
    }
    else if (dat == 'G' && k == 29)
    {
            // if sixth char is G, subtract one from k
            // this indicates that $GPVTG is received as the first six chars
            // so that means we've received the velocity message
            GPS_receive[k] = dat;
            k = k - 1;
```

```
        }
        // wait for k to reach 9, that's when speed starts
        // between k is 9 to 7, the three ASCII chars shows the speed
        else if (k == 9)
        {
                GPS_receive[k] = dat;
                k = k - 1;
        }
        else if (k == 8)
        {
                GPS_receive[k] = dat;
                k = k - 1;
        }
        else if (k == 7)
        {
                GPS_receive[k] = dat;
                k = k - 1;
        }
        else if (k == 0)
        {
                // once we've reached k = 0, reset k to 34
                // and start over in looking for $
                k = 34;
        }
        else
        {
                k = k - 1;
        }
}
```

SD Card Code Listing:

```
/*SD card command function*/

void sd_command(unsigned char command[], unsigned char response[])

{
  int i, limit;
  if (command[0] == 0x40 || command[0] == 0x58 || command[0] == 0x51)
                limit = 1;
  if (command[0] == 0x48 || command[0] == 0x7A)
                limit = 5;

        sspbuf=0xFF;
        while(!sspif);
        sspif=0;
```

```
        for (i=0;i<6;i++)
        {
                sspbuf=command[i];
                while(!sspif){};
                sspif=0;
        }

        do
        {
                sspbuf=0xFF;
                while(!sspif&&!bf);
                sspif=0;
                response[0] = sspbuf;
        }

        while (response[0]==0xFF);

        if (response[0] & 0x04) // if illegal command
        {
                limit=1;
        }

        for (i=1;i<limit;i++)
        {
                while(!bf&&!sspif) {};
    response[i] = sspbuf;
    sspif=0;
        }
  return;
}



/*initialize SD card*/

int sd_init(void)

{
        SD_cs = 0;              // select SD


        // wcol=0, sspov=0, sspen=1, ckp=0 for idle clock is low

        // spi master mode with clock=Fosc/64
        sspcon1 = 0b00100010;
```

```
// sample at middle of input, transition on rising clock

sspstat &= 0b00111111;
sspie = 1;

unsigned char command[]= {0x40,0x0,0x0,0x0,0x0,0x0,0x95}; // CMD0
unsigned char response[5] = {0x00};

// waiting for SD card to reach operating point

int i;

for (i=0;i<20;i++)

{
        sspbuf=0xFF;
        while(!sspif){};
        sspif=0;
}

sd_command(command,response); //CMD0
command[0]=0x48;
command[1]=0x0;
command[2]= 0x0;
command[3]= 0x1;
command[4]= 0xAA;
command[5]=0x95; // CMD8
sd_command(command,response);
do
{
        command[0]=0x77;
        command[1]=0x0;
        command[2]= 0x0;
        command[3]= 0x0;
        command[4]= 0x0;
        command[5]=0x95; // CMD55
        sd_command(command,response);
        command[0]=0x69;
        command[1]=0x0;
        command[2]= 0x0;
        command[3]= 0x0;
        command[4]= 0x0;
        command[5]=0x95; // ACMD41
        sd_command(command,response);
}
```

```
        while (response[0]!= 0x00);

        command[0]=0x50;
        command[1]=0x0;
        command[2]= 0x0;
        command[3]= 0x02;
        command[4]= 0x0;
        command[5]=0x95; // CMD16 setting 512b block length
        sd_command(command,response);

        SD_cs = 1;

        return 1;
}
```

Subsystem Testing:

The current sensors were tested by placing the wires coming out of the transformer and ultracapacitor bank, respectively, through each of the current sensors. The current sensors were connected to the microcontroller, allowing us to verify that they were reading the current flowing through the wires. Because we could not successfully debug the SD card programming, we were unable to test that the system could write data to the SD card.

Because we were unable to debug the GPS and SD card programming, we were unable to test its functionality.

Interfaces and Sensors

Because our interfaces and sensors are integral to our subsystems, we have already described all but one of them in detail in the above sections.

We have also designed an LCD interface for the end user. It is currently programmed to display the voltage level on the ultracapacitor bank. Although the code has not been completely debugged, it is listed below.

LCD Code Listing:

```
/*initialize SPI LCD*/

void LCD_init(void)
{
        sspen = 0;              // first disable SPI,

        // then configure SPI
        trisc.0 = 0;
        trisc.3 = 0;
```

```
        trisc.5 = 0;
        sspcon1 = 0x12;
        sspstat = 0;

        // then re-enable SPI
        sspen = 1;
        return;
}

/*send 'x' byte of the tx_packet[] to the serial port*/
void send_packet(unsigned int x)
{
        unsigned int ix;
        LCD_cs = 0;                                     // select LCD
        for (ix = 0; ix < x; ix++)
        {
                sspbuf = tx_packet[ix];
                while(!bf);
                delay_us(5);                    // reduce effective clock rate
        }
        LCD_cs = 1;                                     // unselect LCD
        return;
}

/*set LCD cursor (position)*/
/*@param x: send 8 bit number that tells the position of LCD display*/

void LCD_cursor(unsigned int x)
{
        tx_packet[0] = 0xFE;
        tx_packet[1] = 0x45;
        tx_packet[2] = x;
        send_packet(3);
        delay_ms(10);
        return;
}

/*clear one line of display*/
/*@param x: the line in which we want to clear*/

void clear_line(unsigned int x)

{
        unsigned int ij;
        for (ij = 0; ij < x; ij++)
        {
```

```
                tx_packet[ij] = 0x20;
        }
        send_packet(x);
        return;
}

/*clear the LCD display*/
void LCD_clear(void)
{
        tx_packet[0] = 0xFE;
        tx_packet[1] = 0x51;
        send_packet(2);
        delay_ms(10);
        return;
}

/*outputs ASCII character val to the LCD*/
void LCD_char(unsigned char val)

{
        tx_packet[0] = val;
        send_packet(1);
        delay_ms(10);
        return;
}

/*outputs decimal data to the LCD*/
void LCD_dec(char data)

{
        LCD_char(((data /100) & 255) + 0x30);
        data = data % 100;
        LCD_char(((data / 10) & 255) + 0x30);
        LCD_char(((data % 10) & 255) + 0x30);
        delay_ms(10);

        return;
}

void LCD_ucap(char data)

{
        unsigned short data_conv1;
        data_conv1 = (data*2)/5;
        data_conv1 = data_conv1 + 1;
        unsigned char data_conv2 = data_conv1;
```

```
        LCD_char(((data_conv2 / 10) & 255) + 0x30);     // display value in 10
        LCD_char(((data_conv2 % 10) & 255) + 0x30);     // display value in 1
        LCD_char(0x20);                                 // display a space
        LCD_char(0x56);                                 // display a V
        delay_ms(10);
        return;
}
```

System Integration Testing

In order to test our entire system, we first loaded generator, transformer, and ultracapacitor bank onto the bed of the truck. We bolted the transformer and generator onto a wood platform to minimize movement while the truck is in motion and/or the generator is on.

After making all the necessary connections as mentioned in the interface section of each subsystem, we tested the entire system by driving the truck. Intially, everything worked, and we were able to reach one of our overall goals by driving 35 mph. However, after some time the control circuitry stopped working. Upon realization that a current spike of about 35-40 A was running through the MOSFET circuitry, we realized that we had fried our MOSFETs. Thus, we replaces these MOSFETs and added a relay capable of handling this current.

After making these modifications, we tried running the truck again. Similar to the first test run, the entire system worked for a while and then stopped. We first thought that the generator was no longer providing power under a load. This was because the control circuitry was still capable of turning the generator on and off, but the voltage level on the ultracapacitor bank was not increasing as the generator ran. Luckily, we were able to determine that this was not the case by connecting the generator to multiple loads (space heaters) to the generator and observing that it was still capable of outputting power under a load. Our next option was to disconnect everything and test each of the subsystems very much like described in the subsystem descriptions. It was this testing that led us to discover that the full wave bridge rectifiers had overheated. We went back to the lab to re-solder a new rectifier circuit, being sure to include heat sinks to prevent overheating.

Upon this final modification, we were able to connect everything to drive the truck successfully. We estimated the fuel efficiency by driving the truck around the parking lot in Innovation Park. After driving 1.5 km, we used 60 mL of diesel gas. This corresponds to driving 0.932 miles using less than 0.001 gallons. Using these numbers, we estimated that the fuel efficiency of our design is about 117 mpg. We acknowledge that this is not the most realistic testing scenario, as our driving schedule was much slower than a city driving schedule (5 mph) and the typical stop-and-go traffic of city driving has a dampening effect on fuel efficiency. Even so, we believe our fuel efficiency meets our goals of 60-80 mpg even after accounting for the reductions in fuel efficiency just discussed.

User's Manual

This user's manual will detail how to make the necessary connections to drive the hybrid truck.

First, the 12 V battery on the generator must be disconnected after each use of the truck. Our design utilizes this battery to power the microcontroller, and, if this is not disconnected, the microcontroller will continue to drain the battery.

The following picture will be useful for making the necessary connections:

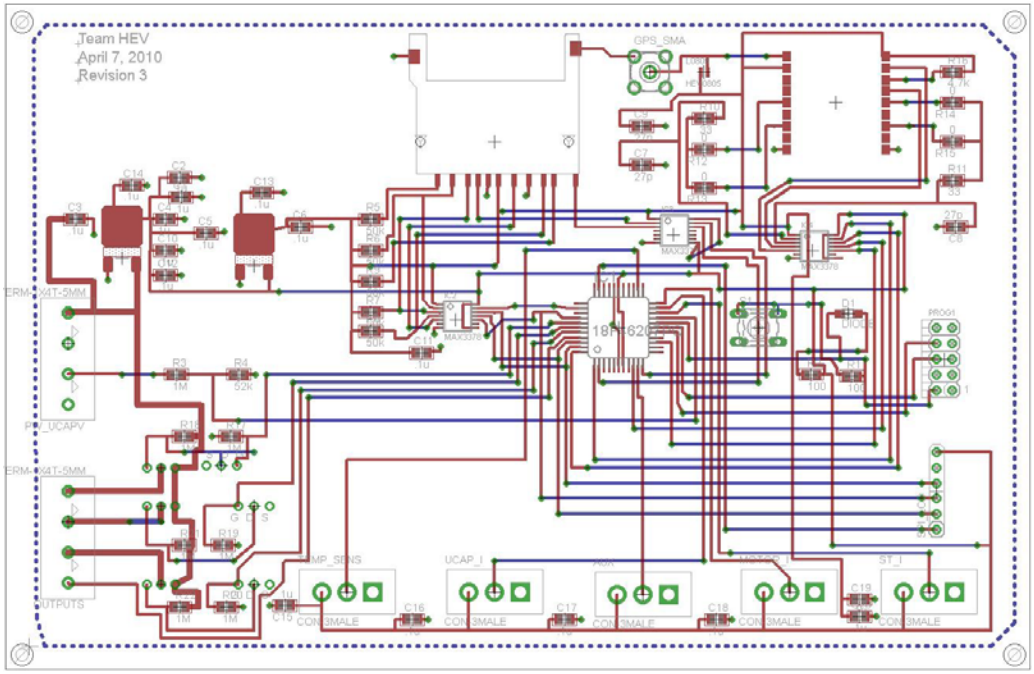

Figure 8: Board Layout

1. Positive terminal of generator's 12 V battery.
2. Negative terminal of generator's 12 V battery.
3. Positive terminal of ultracapacitor stack.
4. Negative terminal of ultracapacitor stack.
5. On/Off switch for generator's fuel valve.
    - Located in the middle section of the panel side of the generator.
6. Switch for generator's glow plug.
    - Located on the top side of the generator opposite the diesel tank.
7. Switch for generator's starter motor.

- Located on the battery side of the generator in the middle near the battery.

8. Auxiliary digital output; not necessary to run truck.

a. Temperature sensor.
   - Located on top of the generator attached to the glow plugs.
b. Ultracapacitor current sensor.
   - Blue device surrounding cable connecting the bridge rectifiers to the ultracapacitor bank.
c. Auxiliary analog or digital input; not necessary to run truck.
d. Driving motor current sensor.
   - Blue device surrounding cable connecting the ultracapacitor stack to the driving wheels (which connects out of the bottom of the truck bed).
e. Starter motor current sensor.
   - Blue device surround cable connecting the starter motor to the engine of the generator.

Conclusions

The purpose of this project was to hybridize an electric truck using a diesel generator, while implementing an alternative technology than batteries to power the motors. Using ultracapacitors to drive the motors, we were able to design a way to use a diesel generator to charge these ultracapacitors using a series hybrid design. We met our overall goals:

- 60 – 80 mpg fuel efficiency
- driving range greater than 30 – 40 miles per charge
- increase speed capabilities from 25 mph to 35-40 mph

We also designed capabilities for GPS, SD card, and an LCD interface. While we were unable to get these capabilities fully functioning, we do have code for each.

Appendices

APPENDIX A: *HARDWARE SCHEMATICS*

Control Board: attached

Current Sensor Board: attached

APPENDIX B: *COMPLETE SOFTWARE LISTING*

APPENDIX C: *COMPLETE BILL OF MATERIALS:* attached

APPENDIX D: *DATA SHEETS*

Microntroller:
http://seniordesign.ee.nd.edu/2010/Design%20Teams/HET/PIC18F4620.pdf

GPS
http://seniordesign.ee.nd.edu/2010/Design%20Teams/HET/ISM300F2_Functional_Spec.pdf

SD Card connector
http://seniordesign.ee.nd.edu/2010/Design%20Teams/HET/Simplified_Physical_Layer_SD_Spec.pdf

LCD
http://seniordesign.ee.nd.edu/2010/Design%20Teams/HET/SPI_LCD_specs.pdf

Temperature sensor: attached

Current sensor: attached

APPENDIX E: *MICROCONTROLLER PIN CONNECTION SHEET:* attached.