

Design Summary

Our design team, Team 45, was really excited to construct an electronic motorized mountainboard for our project. Using a powerful, yet small, electronic motor we would create a mountainboard that would move without the need of propulsion from a human foot. The controls would have to be a part of a hand device connected to the board itself so speed could be measured and regulated.

An electric motor, which operates on a maximum current of 27.5 amps is attached to the rear of the board to provide forward thrust for movement. A chain connects the motor component to the sprocket, which is attached to the rear axle of the board. The rotational power operates on a gear ratio of about 4.5 to 1. We decided on this gear ratio based on the RPMs of the motor so as to reduce the speed while still maintaining enough torque to get up hills. Two twelve volt batteries, one which is twenty amp hours and the other being thirty-five amp hours, are connected in series make up the main power source, providing the motor with enough current to operate. These batteries are very large and heavy. We found the only safe place to contain them was between the feet of the rider of the board. This also brings the center of mass for the board down and decreases the likely-hood of tipping over while turning. The batteries are able to be recharged for longevity of life and reuse. A motor controller acting as a safeguard from the giant batteries and the motor assures only twenty amps maximum is delivered to the motor, so the electrical components inside do not get fried and release the smoke. Attached to the outside of the electrical encasement mounted to the back of the board is a photocell which detects the ambient light. When it gets dark enough outside it triggers reverse logic LEDS and at a high resistance turns on mounted headlights and taillights to let the rider see and be seen at night. All the electrical components are controlled by a two PICs, one on a breadboard in the electrical encasement on the board and the other in the hand device. The two devices are connected by a coil wire which gives enough slack hold in the riders hand and still communicate between the two breadboards. A variable trigger throttle on the hand device controls the amount of current supplied to the motor and can regulate the speed the board will travel.

Controlled by the hand device PIC is an LCD screen attached to the side and a buzzer attached to the nose of the device. Firstly, the LCD is powered by two nine volt batteries on the body of the skateboard to ensure it will have enough life to stay operational for a long time. A second photocell measuring ambient light will turn on an LED over the LCD display screen if it is too dark to see. Displayed on the screen are the speed and the distance traveled by the board. The distance is calculated on one of the PICs by information sent by a photo-interrupter mounted to the rear of the board. Knowing the circumference of the wheel, the PIC is able to calculate how many miles the board has traveled during the current battery life as well as the current speed. A button mounted on the opposite side of the hand device clears the odometer reading if the rider chooses to. Through testing, we were able to get the board to travel at a maximum speed of 13 miles per hour. This value decreased while traveling up hills but still had enough torque to conquer the hill with ease. Lastly, the buzzer on the nose acts as a horn to alert people or other vehicles of your presence if needed. A button located close to the handle of the hand device operates the buzzer and is activated by the rider.

Design Details

Basically the electronic mountainboard is a group of separate systems working in tandem to achieve a working product. First off is the battery level indicator. This works by using the internal analog to digital converter in a PIC16F88 in conjunction with a voltage divider in order to proportionally lower the voltage to a level the PIC can read. Basically two factors went into choosing the resistors for the voltage divider, power consumption and resistance value. We needed the larger branch to be just over four times as much as the smaller branch while controlling the current so that the power never surpassed a half of a watt. Then the analog to digital converter checks to see if the battery level is lower than the set point. If it is then it turns off the green LEDs and moves on to the next set point. This goes on until the battery drains to a critical level. The last led signal is actually being sent to a transistor which while high creates a complete circuit from a timer chip to ground, bypassing the 100 microfarad capacitor. Basically while the pin is high, the LED is on, and when it is low, the LED blinks.

The next subsystem was the headlight and taillight automation. A photocell is mounted on the board connected to the PIC. The POT command was used in order to find out the resistance of the photocell. When more light is applied the resistance lowers and vice versa. The PIC then decides whether or not it is dark enough to need headlights and taillights. We also did not want it to turn on if the rider is just passing through some shade, otherwise it will blink every time it comes out of the shade. This was controlled by having a much lower set point for turning on the lights and a higher one to turn them off. This merely means there would have to be a much larger difference between the light and shade before it changed. In order to increase the brightness of the headlights a 33 Ohm resistor was bridged in parallel with the 330 Ohm resistor.

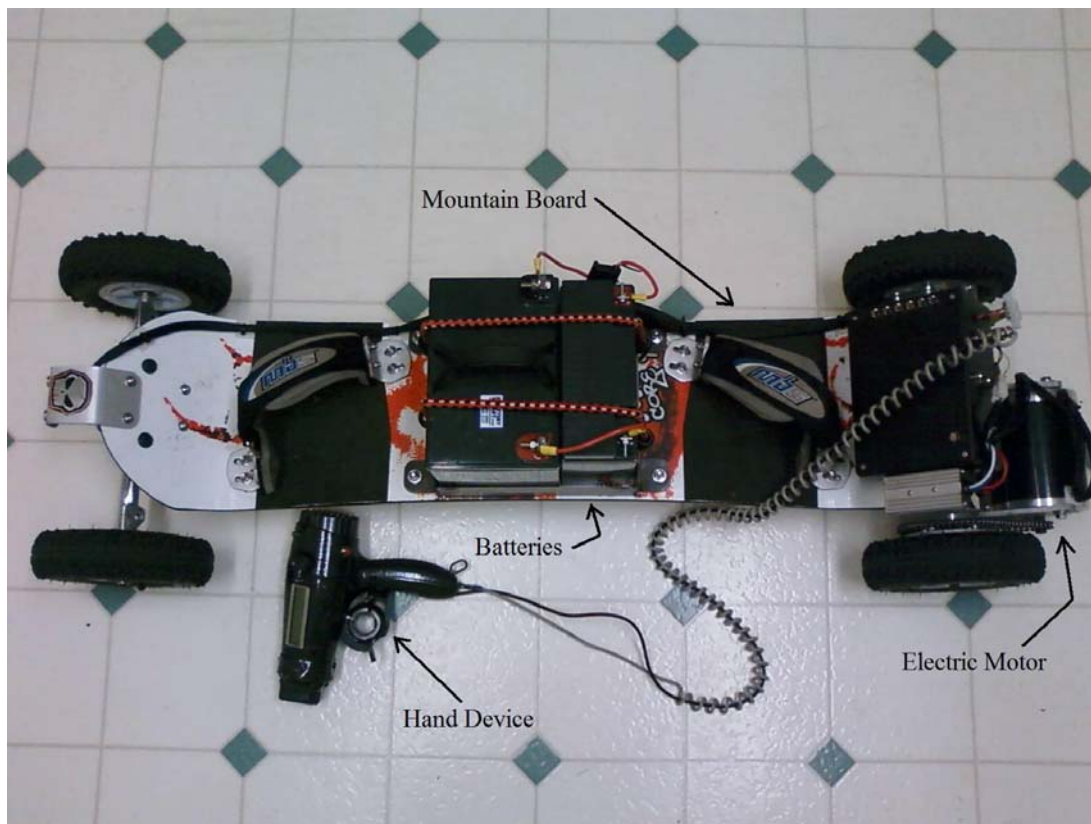
Another subsystem is the motor control. This is controlled by a hall-effect variable throttle. A hall-effect's output is similar to a large potentiometer except instead of varying the resistance it uses magnetic fields inside the throttle to control how high the output signal will be. This then goes into our motor controller which has built in fly back diode protection and utilizes a pulse width modulation output to control the speed of the motor. It has a few other inconsequential features that we decided not to use, but the biggest draw it had was that it did everything we needed while protecting the motor and our circuits as well.

The last subsystem that has components on the board section is the speed and distance computer. A photo-interrupter is mounted near the rear non-drive wheel. It has a disc that runs through it while the wheel spins. One small notch is cut out so that every time one revolution has been made a pulse is sent to the PIC on the hand device. The PIC counts how many revolutions there are in a two second time frame and then calculates speed. We decided on three variables for speed and distance. The first variable is in small units so that rounding does not end up making our calculations erroneous. The second two are the displayed numbers converted from the first variable. One acts as a decimal place, and the other one acts as the whole number. The same is done with the distance taking just the revolutions times the circumference. The smaller unit number is in feet so unfortunately the max distance for one trip is 12.4 miles before it will automatically reset to zero. The rider can also manually reset the odometer by

pressing the button on the back of the hand device. All the data is continually output onto the LCD mounted on the hand device.

The next subsystem is the most simple but very effective. The horn is a very necessary component as a safety upgrade from a regular board. Not only do we need to warn pedestrians that we are coming but since we will be in traffic, it may be necessary to warn cars of our presence as well. For that reason we chose a piercing 120 decibels buzzer. It sounds more like a siren than a horn at times, but it achieves its goal of making our presence known and keeping the rider safe. It is hooked up straight from the two nine volts through a button and then to the buzzer.

The last subsystem is the LCD backlight LED. Unfortunately the three LCDs we had donated, the two that had their own back lights were broken overall. The third one either had a broken backlight or never had one to begin with. This is why we decided to make a circuit involving a photocell so that when it was bright outside less energy was drawn by the LED, and vice versa. It also utilizes a 10k potentiometer and a transistor to make it just about 0.7 volts at the base of the transistor in decent light. When it got darker, a greater voltage drop occurs over the photo cell making the transistor fully saturated. When it got lighter, more of the flow of is allowed to flow through the photo cell making the transistor partially saturated.



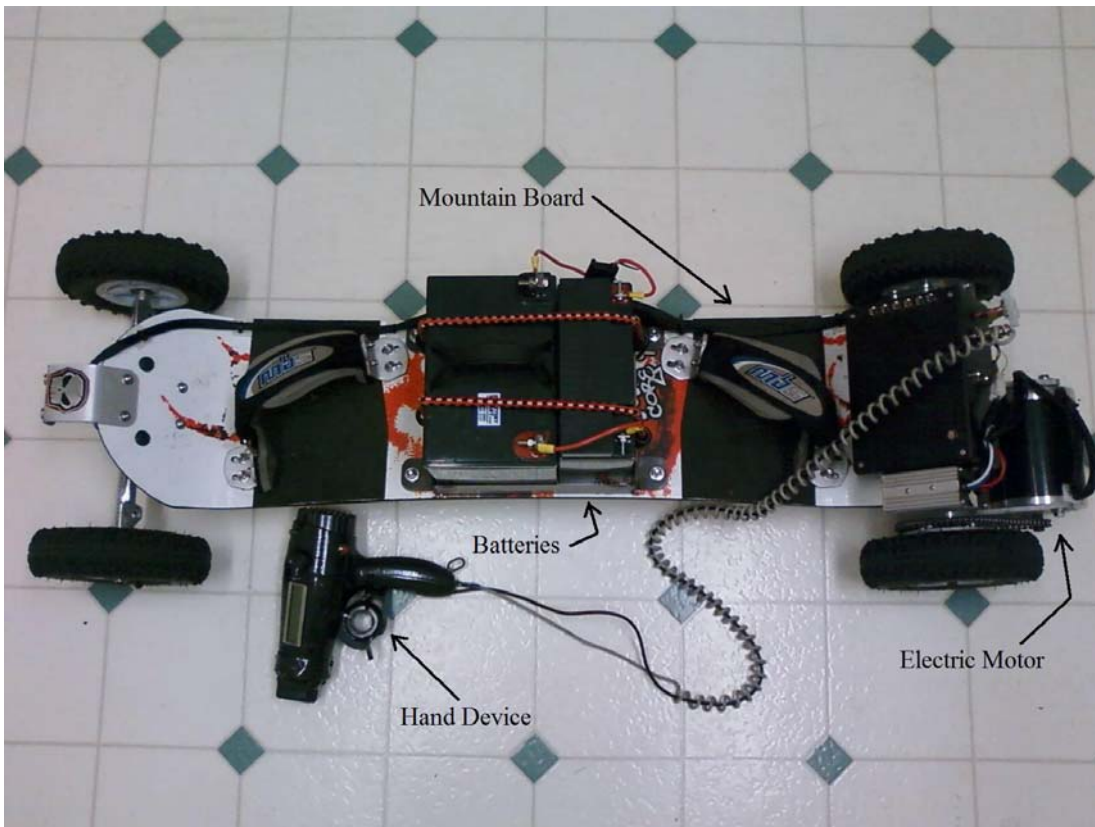


Figure 1: Top View



Figure 2: Front View

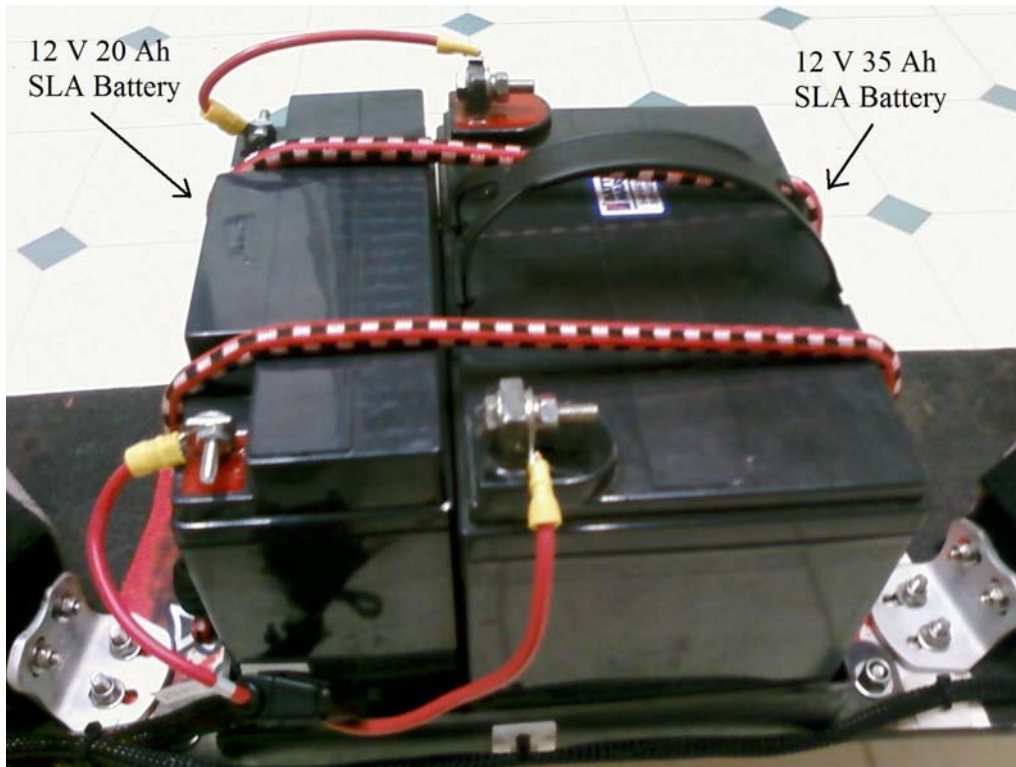


Figure 3: Motor Batteries

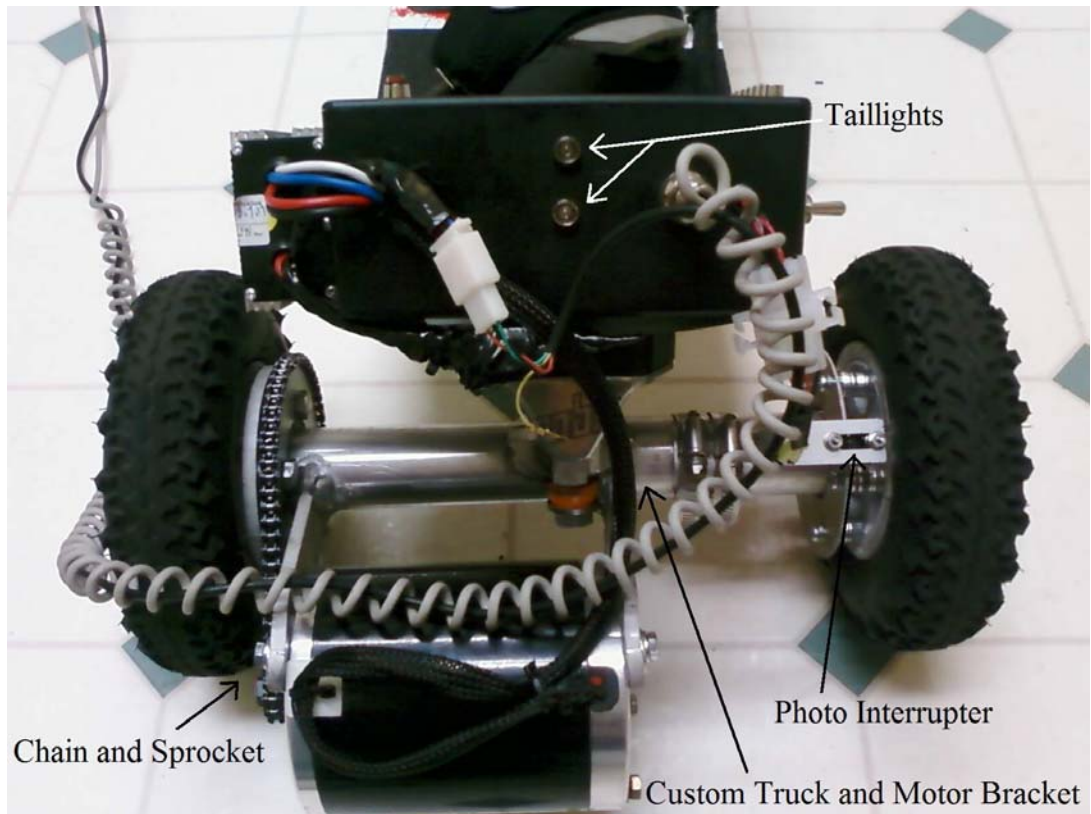


Figure 4: Rear View

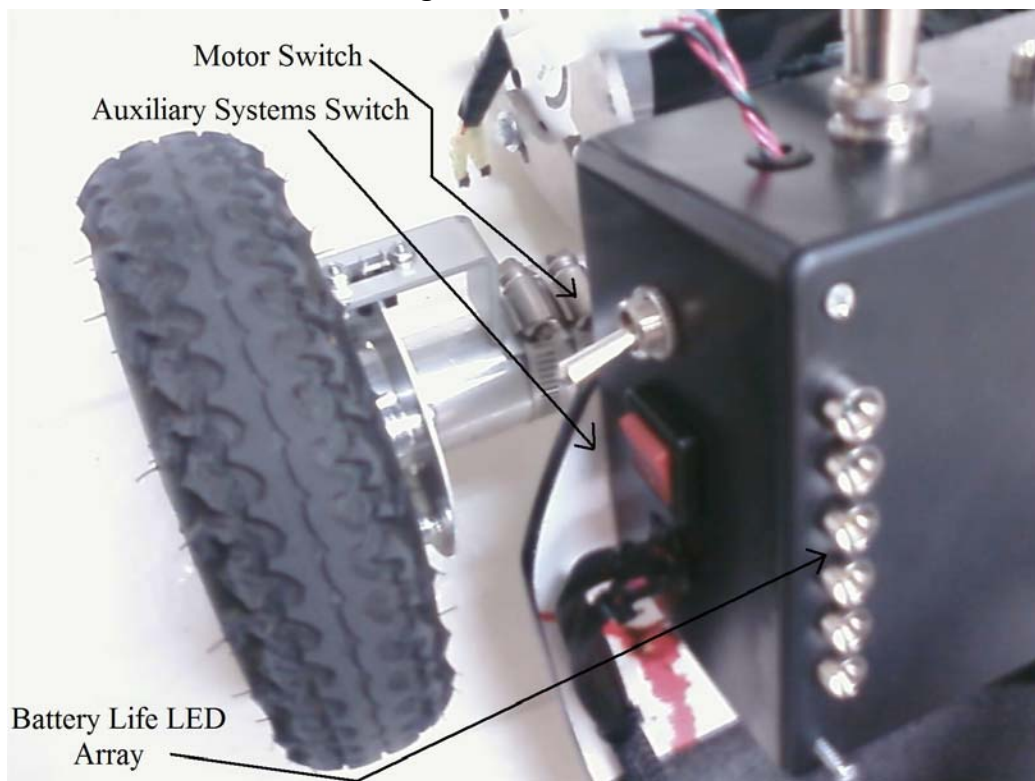


Figure 5: Switches

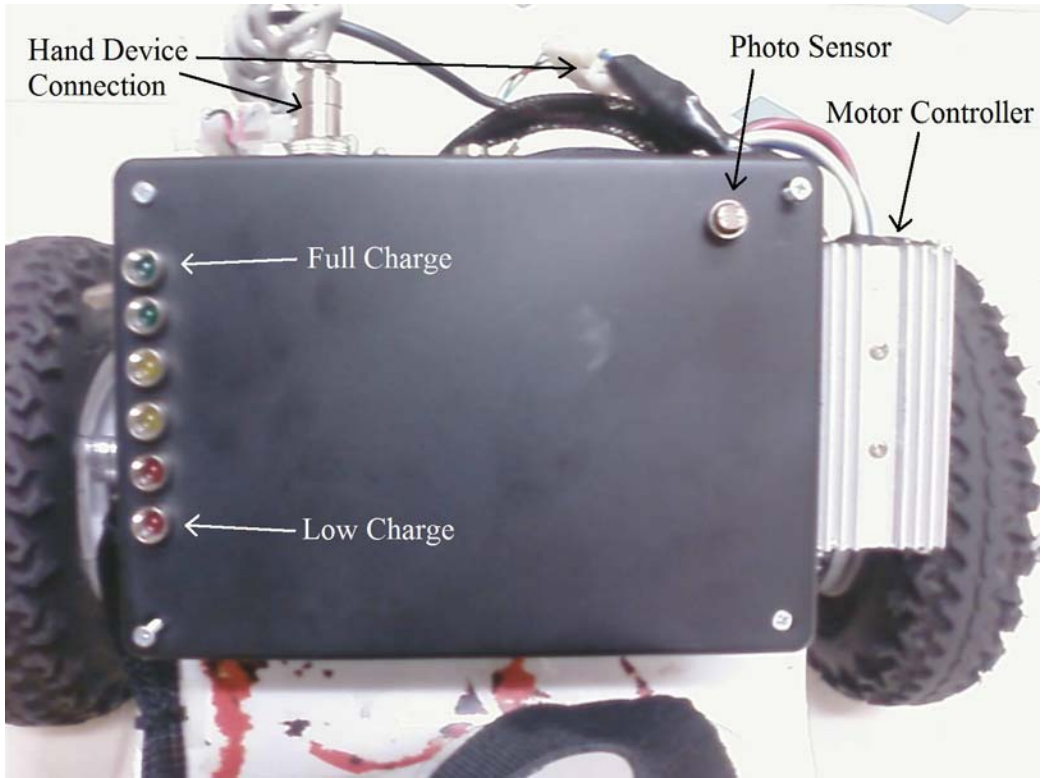


Figure 6: Top of Circuit Box



Figure 7: Front View of Hand Device



Figure 8: Rear View of Hand Device

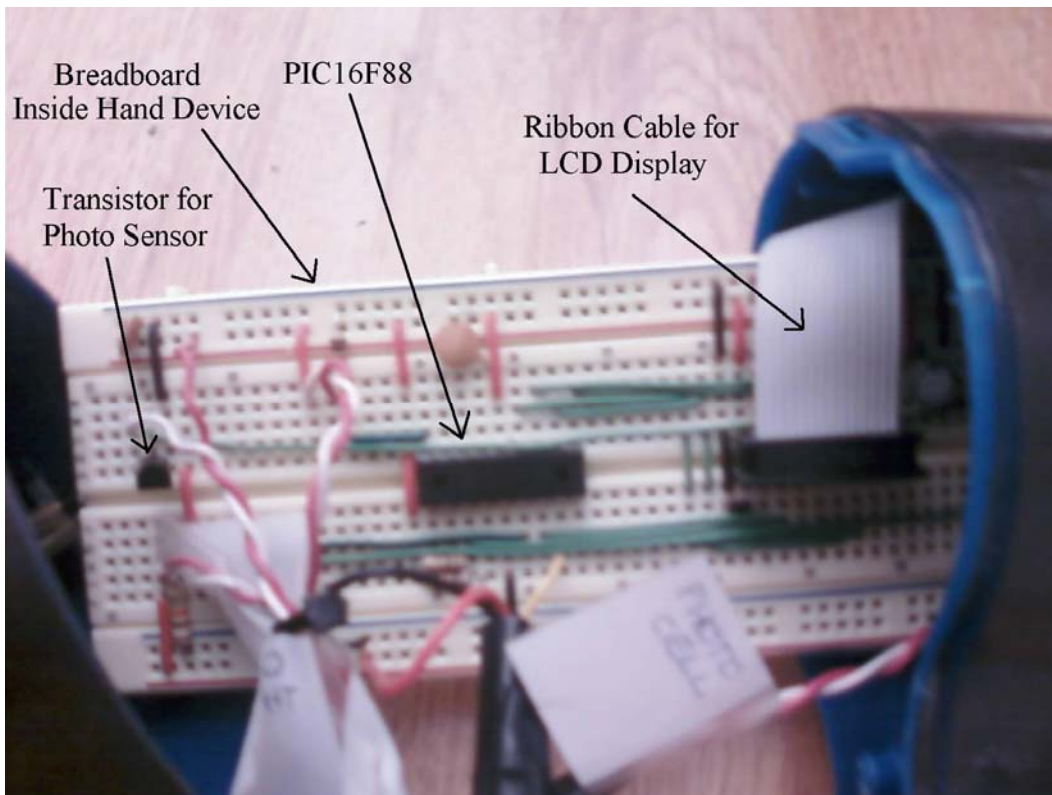


Figure 9: Breadboard Inside Hand Device



Figure 10: Inside View of Circuit Box

Design Evaluation

A. Output Display

For output display we fully meet the requirements and beyond. We have multiple LED arrays including six for the battery level indicator, two for the tail lights, two superbright for the headlights, and one for the LCD backlight. We also have an LCD display which gives data on current speed and distance traveled to the rider. The headlight LED's in particular involved extra research into what resistance values would maximize the light output without releasing the smoke. Also the LCD required a lot of extra research since we were unaware that the LCDs that were supplied were fried and also the differing voltage requirements for contrast. Also since it was more convenient for us to use different pins than the defaults we needed to find out what the functionality of each pin was and how to change the defaults in programming.

B. Audio Output Device

This is the one category we are weak in, as we only have one very simple sound device. Although it is simple, the buzzer does serve a vital purpose to the project and that is the safety of the rider at all times. We really have no need for any other sounds in the board either, the board is fine as is.

C. Manual Data Input

This category we have quite a bit, including one that is not mentioned. Our hall-effect throttle acts as a much more sophisticated potentiometer in that it's voltage output is linear depending on how much the trigger is pulled. We also have multiple buttons for the horn and the odometer reset. We also have two potentiometers, one for the contrast on the LCD and one for the LCD backlight circuit.

D. Automatic Sensor Input

This category is met easily by the mountain board's photo interrupter for clocking the RPMs. The photo interrupter runs through a disc with a small notch in it so as to send out a pulse every time the wheel comes around once. We also have two potentiometers as I have said before for contrast of the LCD and brightness of an LED. The two photo cells are also obvious contributors to this system as they control almost a third of the board's functionality. Last but not least is the circuit utilizing a timer chip to make an LED blink without bogging down the PIC with useless code. This category was easily met and surpassed since our group tried to use these as much as possible to simplify the programming and thus less debugging.

E. Actuators

This category is also easily met by the main component of our board, the pulse width modulated electric motor. The motor controller we have does have the fly back protection needed to keep the motor from being fried or releasing a large amount of electromagnetic interference to our circuits. We did try to do it ourselves with a PIC and h-bridges, but it turned out to be more costly with less protection to do it ourselves. The PWM controlled motor being the only real actuator we need for our project we meet the requirements for this category.

F. Logic, Counting, Integration, and Control

This category is met and surpassed once again. Our photo interrupter circuit counts our revolutions per minute allowing us to derive speed and distance. We utilized an analog to digital converter internally in the PIC in order to show battery level. Our board also has multiple PIC microcontrollers in order to control all of the functions and the calculations that need to be made. The board is controlled by both, though they are not interconnected. All of these subsystems are accumulated into the project to make it controlled mostly by simple logic statements and counters.

Grading adjustments

Our proposal got a positive score meaning it was well written, concise and clear. We also have gotten good scores on our weekly deliverables in lab and we met and surpassed most if not all of the categories

for grading. We have kept up with everything we have needed to turn in, and we have done them all well. The tasks set out each week have helped us to fix our initial timeline as it was too ambitious and have kept us on track working towards the end of the project.

We expect to do well for this project since it is just a prototype of an electronic mountain board and with a bit more time, the cost could easily become marketable. We were frugal for this project even though we had many large expenses, the project was a very large scope and required more expense than most. We also had many things donated to us which we tried to use instead of getting new parts even if they were not the best part for the job. We worked very hard making the wiring clean and basically unnoticeable, not only for aesthetics but also to reduce the chance of noise in our circuits.

We expect to do well in proving that the project is reliable and each component is an integral part of the whole function of the device. We have done extensive testing and had minimal problems with mechanical issues, not electrical ones. Also we have spent considerable time and effort making the mountain board safe to ride for us, however if marketed to younger generations more work will have to be done in order to keep it safe for them as well.

Parts List

Description	Part #	Source	Price	Qty	Total
24V 500W 2500RPM Motor	MOT-24500X2500	Electric Scooter Parts	\$84.95	1	\$84.95
Hall- Effect Thumb Throttle	THR-65	Electric Scooter Parts	\$19.95	1	\$19.95
24V 500W Motor Controller	SPD-2404	Electric Scooter Parts	\$39.95	1	\$39.95
12V 20AH Sealed Lead Acid Batt	LA-12V20-NB	Battery Space	\$48.95	1	\$48.95
In Line Mini Blade Fuse Holder	2700015	Radioshack	\$2.69	1	\$2.69
30A Mini Blade Fuse 3 Pack	2700014	Radioshack	\$1.99	1	\$1.99
10mm LED Holder	LED-10H-CB	Moddersmart	\$2.25	2	\$4.50
10mm White LED 15000mcd	LED-10-W	Moddersmart	\$0.25	2	\$0.50
12V 36AH Sealed Lead Acid Batt		Donated	\$0.00	1	\$0.00
Photointerrupter		Donated	\$0.00	1	\$0.00
Photo Cells		Donated	\$0.00	1	\$0.00
					\$203.48

Table 1: Parts List


```

'low voltage program                disabled
'data EE read protect              off
'flash program write                write protect off
'ccp1 mux                          RBO
'code protect                       off
'Fail-safe clock monitor enable     enabled
'internal external switch over mode disabled
.....

```

```

DEFINE OSC 8

```

```

OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz

```

```

OSCCON.5 = 1

```

```

OSCCON.6 = 1

```

```

ansel.2 = 1          'activates adc for an2

```

```

adcon1.1 = 1        'makes the adc 10 bits and right justified

```

```

adcon1.7 = 1

```

```

Define ADC_BITS 10 'ensures the adc is 10 bits

```

```

batt_word   Var   WORD      'stores the battery level as a word from the adc

```

```

batt_byte   Var   BYTE      'creates variable for making the adc word a byte and more
easily used

```

```

batt_led_1  Var   PORTA.7    'sets the first battery LED to RA.7

```

```

batt_led_2  Var   PORTA.0    'sets the second battery LED to RA.0

```

```

batt_led_3  Var   PORTA.1    'sets the third battery LED to RA.1

```

```

batt_led_4  Var   PORTB.0    'sets the fourth battery LED to RB.0

```

```

batt_led_5  Var   PORTB.1    'sets the fifth battery LED to RB.1

```

```

batt_led_6  Var   PORTB.2    'sets the signal to the timer for the last battery LED
to RB.2

```

```

Full_charge Var   Byte      'shows the full charged level note though when a load is
applied a temporary loss

```

```

'is seen in the voltage so the first step is

```

```

larger to account for that

```

```

No_charge   Var   Byte      'shows the level we dont wish to go below for fear of
losing the batteries

```

Charge1	Var	Byte	'the first partition of battery level
Charge2	Var	Byte	'the second partition of battery level
Charge3	Var	Byte	'the third partition of battery level
Charge4	Var	Byte	'the forth partition of battery level
Full_charge = 253			'shows the full charged level note though when a load is applied a temporary loss
			'is seen in the voltage so the first step is larger to account for that
No_charge = 232			'shows the level we dont wish to go below for fear of losing the batteries
Charge1 = 247			'the first partition of battery level
Charge2 = 242			'the second partition of battery level
Charge3 = 237			'the third partition of battery level
Charge4 = 232			'the forth partition of battery level
head_led_1	Var	PORTB.4	'sets the first Headlight LED to RB.4
head_led_2	Var	PORTB.5	'sets the second headlight LED to RB.5
tail_led_1	Var	PORTB.6	'sets the first taillight LED to RB.6
tail_led_2	Var	PORTB.7	'sets the second taillight LED to RB.7
Photo_Cell	Var	PORTB.3	'sets te photo cell input as RB.3
Scale	Con	255	'sets the scale for the pot command
Brightness	Var	Byte	'sets the brightness pot output as a variable
Light_pot	Con	150	'the level of light when the headlights should turn off
Dark_pot	Con	100	'the level of light when the headlights should turn on
TRISA = %00000100			'sets all the pins in PORTA as outputs except for RA.2
TRISB = %00001000			'sets all the pins in PORTB as outputs except for RB.3
LOW Head_led_1			'ensures headlights and taillights are off
LOW Head_led_2			
LOW Tail_led_1			
LOW Tail_led_2			
High Batt_led_1			'boot up lights

High Batt_led_4
High Batt_led_5
High Batt_led_6

Pause 1000

Low Batt_led_1
Low Batt_led_4
Low Batt_led_5

Pause 100
High Batt_led_5
Pause 100
High Batt_led_4
Pause 100
High Batt_led_1
pause 1000

led1: 'first battery LED loop

ADCIN 2, batt_word 'takes the reading from the adc and outputs as a word
batt_byte = batt_word / 4 'shortens the word to a byte
if (batt_byte < Charge1) Then 'tests if the first LED should go off
 Low Batt_Led_1
 Goto led2
endif
gosub ambient 'goto subroutine for sensing ambient light
goto led1 'do it until the LED is low

led2:

ADCIN 2, batt_word 'takes the reading from the adc and outputs as a word
batt_byte = batt_word / 4 'shortens the word to a byte
if (batt_byte < Charge2) Then 'tests if the second LED should go off
 Low Batt_Led_4
 Goto led3
endif

```
gosub ambient      'goto subroutine for sensing ambient light
goto led2          'do it until the LED is low
```

led3:

```
ADCIN 2, batt_word 'takes the reading from the adc and outputs as a word
batt_byte = batt_word / 4 'shortens the word to a byte
if (batt_byte < Charge3) Then 'tests if third LED should go off
    Low Batt_Led_5
    Goto led4
endif
gosub ambient      'goto subroutine for sensing ambient light
goto led3          'do it until the LED is low
```

led4:

```
ADCIN 2, batt_word 'takes the reading from the adc and outputs as a word
batt_byte = batt_word / 4 'shortens the word to a byte
if (batt_byte < Charge4) Then 'tests if the forth LED should go off
    Low Batt_Led_6
endif
gosub ambient      'goto subroutine for sensing ambient light
goto led4          'do it forever
```

```
ambient:          'subroutine for sensing ambient light
POT Photo_cell, Scale, Brightness 'take the pot value on the photo cell
if (brightness < dark_pot) then      'test if the headlights should be on
    High Head_led_1
    High Head_led_2
    High Tail_led_1
    High Tail_led_2
Endif
if (brightness > Light_pot) then      'test if the headlights should be off
    Low Head_led_1
    Low Head_led_2
    Low Tail_led_1
    Low Tail_led_2
```

```
Endif
Return

end
```

LCD PIC

```
'16F88 register settings for ME307 labs
'The following configuration bits and register settings
'enable the internal oscillator, set it to 8MHz,
'disables master clear, and turns off A/D conversion
```

```
.....
```

```
'
                                CONFIGURATION BITS
'oscillator                      INTRC-OSC2 as RA6
'watchdog timer                  off
'power up timer                  off
'RA5/MCLR pin function select    MCLR
'brown out detect                on
'low voltage program             disabled
'data EE read protect            off
'flash program write            write protect off
```

```
'ccp1 mux                                RBO
'code protect                             off
'Fail-safe clock monitor enable           enabled
'internal external switch over mode      disabled
.....
```

```
DEFINE OSC 8
```

```
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
```

```
OSCCON.5 = 1
```

```
OSCCON.6 = 1
```

```
ansel = 0      'Turns off analog to digital conversion. Refer to
                'Threaded Design Example A.4 p.296-299 of the textbook
                'for an example of how to configure and use A/D conversion
```

```
Distance1    Var    Word  'stores the distance variable as a word in feet (preserved so that
the rounding errors are minimal)
Speed1       Var    Word  'stores the speed variable as a word in inches/sec (preserved so
that the rounding errors are minimal)
Distance2    Var    Word  'stores the total distance variable as a word in miles
Speed2       Var    Word  'stores the speed variable as a word in miles/hr
Distance_dec Var    Word  'stores the decimal of the distance as a word in miles
Speed_dec    Var    Word  'stores the decimal place of the speed as a word in miles per hour
Revol        Var    Byte  'stores the revolutions of the wheel as a byte in the progress of the
movement
Time         Var    Byte  'Stores the time variable between pulses as a byte
Reset        Var    PORTA.1  'Sets the reset button to pin RB.0
Photo        Var    PORTB.2  'Sets the photointerupter to port RB.2
```

```
Distance1 = 0      'makes the starting distance 0
Speed1 = 0         'makes the initial speed 0
Distance2 = 0     'makes the starting distance 0
Distance_dec = 0  'makes the decimal place for the distance 0 initially
Speed2 = 0        'makes the initial speed 0
Speed_dec = 0     'makes the initial speed decimal 0
Revol = 0         'makes the initial revolutions 0
```

Time = 2 'makes the time between recycles 2 second

TRISA = %00000000 'sets all of the ports in port A as outputs except for RA.1

TRISB = %00000100 'sets all of the ports in port B as outputs Except for RB.2

DEFINE LCD_DREG PORTB 'defines the output pins to LCD on port B

DEFINE LCD_DBIT 4 'defines output pins to LCD start on pin RB.4

DEFINE LCD_RSREG PORTA 'defines the register select pin on the open drain pin in port A

DEFINE LCD_RSBIT 4 'defines the register select pin on the open drain pin RA4

DEFINE LCD_EREG PORTB 'defines the enable pin on port B

DEFINE LCD_EBIT 1 'defines the enable pin on pin RB1

Define LCD_BITS 4 '4-bit interface

Define LCD_LINES 2 '2 line LCD

Define LCD_COMMANDUS 2000 'Adjust for slower LCD modules

Define LCD_DATAUS 50

pause 1000

LCDOUT \$FE, \$80, "Welcome Sir" 'Welcome Screen

LCDOUT \$FE, \$C0, "Initializing..."

Pause 3000

LCDOUT \$FE, 1, "SPEED= MPH"

LCDOUT \$FE, \$C0, "ODO= MILES"

Loop: 'Main Loop

IF (RESET == 1) Then 'Resets the ODO

 Distance1 = 0

Endif

Count PORTB.2, 2000, Revol 'Counts the number of revolutions in two seconds

Math:

Speed1 = Revol * 24 / Time 'rev/s * in/rev

```
Distance1 = Revol * 2 + Distance1    'rev * ft/rev
Speed2 = Speed1 * 5 / 88    'in/s * s/hr * miles/in
Distance2 = Distance1 / 5280    'ft * miles/ft
Speed_dec = (Speed1 - (Speed2 * 88 / 5)) * 25 / 44
Distance_Dec = (Distance1 - Distance2 * 5280) / 528
```

LCDupdate:

```
Lcdout $FE, $80 + 7, " ", DEC Speed2, ".", DEC Speed_dec, " " 'Displays Speed in MPH
Lcdout $FE, $C0 + 5, " ", DEC Distance2, ".", DEC Distance_Dec, " " 'Displays Total Distance
Traveled in MILES
```

Goto Loop

End

