Bike Generator with Fitness Monitoring

ECE 445 Final Paper

David Zhang, Daniel Davidar, and Micheal Westfall
Team 21
TA: Kexin Hui
12/12/18

Abstract: This project is the bike generator with fitness monitoring. This project aims to allow users to charge their electronics while exercising. It also aims to track and display their exercise statistics such as speed and distance traveled on a relatively small detached display.

Contents

1 Introduction

1.1 Objective

Obesity is a severe threat to the well-being of people. It affects 93.3 million American adults and is often the cause of type 2 diabetes [1][2]. Exercise is the most critical action humans can take towards keeping a healthy body. Unfortunately, many Americans have cut exercise from their daily routine. We drive to work only to sit at work. There are standing desks but standing is similar to sitting regarding exercise. For many working Americans, the only time they exercise is during their leisure times. Many of them do not have the motivation to exercise because the consequences for inactivity occur far into the future for them. The objective of this project is to provide additional benefits including charging a battery and monitoring their fitness to give them more reasons to exercise. The impact of this project could be huge since over 100 million Americans have bikes but don't use them, and the average American watch over two hours of television every day [3][4].

The project has two main aspects. One aspect is creating a micro power generation station for bicycles. The other aspect is creating a subsystem which calculates and displays a user's exercise stats. All that is required from the user is to have a bike on which to use the device. It should not be an invasive process, so the user does not need to take apart the bike. The power generator of this project is meant to charge small devices. Larger machines would not be suitable for this project.

1.2 Background

Few companies tackle the same problem the same way and their products are $600 [5]. Exercise bike allows for cycling indoors but those can be $60 to over $1000 [6]. There are also smaller products that can charge smaller things like your phone or headlights but use dynamos instead of a typical generator. These dynamos can cost anywhere from $10 to $50 and do not generate much power [7]. Also, most people have a plug that they can charge their phone with. These phone chargers cost less than $10 [8]. We want our product to be relatively cheap, be used as an exercise machine, be easy to set up, and generate power that can charge a laptop. For example, if there is a blackout, our device can charge phones to be used in emergency communications.

1.3 High-Level Requirements

- Bike generator must be able to net at least 25 watts of power, which is the power generated by the alternator minus the power that is taken from the battery to energize the alternator.
- Our fitness monitor must be able to track the calories burned and energy generated to 3 significant digits. Calories burned must be estimated using an algorithm. Energy generated must be measured.
- Our setup must take less than 5 minutes and require zero outside tools besides the user's bike. The user can remove the bike from our setup in less than 5 minutes. This requirement would differentiate our project from other bike generator projects as invasive procedures in the setup would require more than 5 minutes and outside tools.

2 Design

2.1 Block Diagram

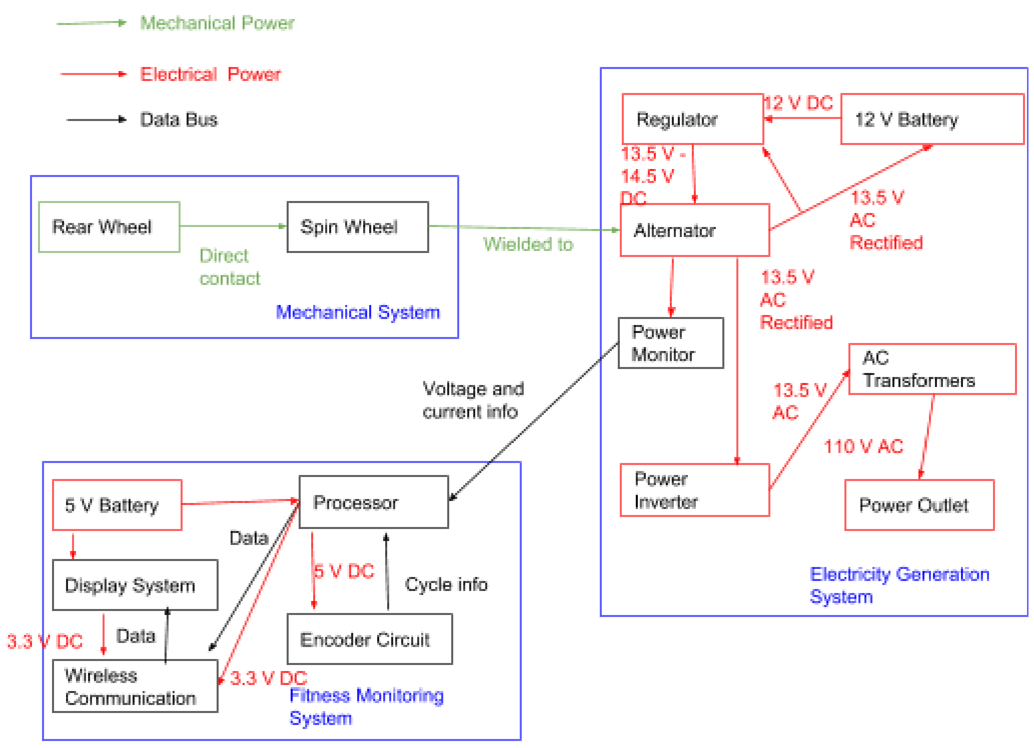

Figure 1: Block Diagram

Our system has three main blocks to it: mechanical, electrical power, and fitness monitoring. The bike wheel is meant to drive the spin wheel that is connected to the alternator. When current is supplied to the field winding of the alternator and is spun it produces voltage and current proportional to how fast the alternator and its field current. The regulator is meant to regulate how much current the alternator is receiving by responding to the output voltage. The alternator has a rectified output so as not to supply a negative voltage to the field winding. This means it has to be turned back into a full-wave ac, which is done by a filter on the power inverter which then takes dc voltage and turns it into ac voltage. The monitoring block is meant to check speed, power generation, and calories burned and display it to the user. The encoder counts the revolutions made by the bike and sends it to the processor; which calculates stats and sends it wireless communication that sends it to the display system.

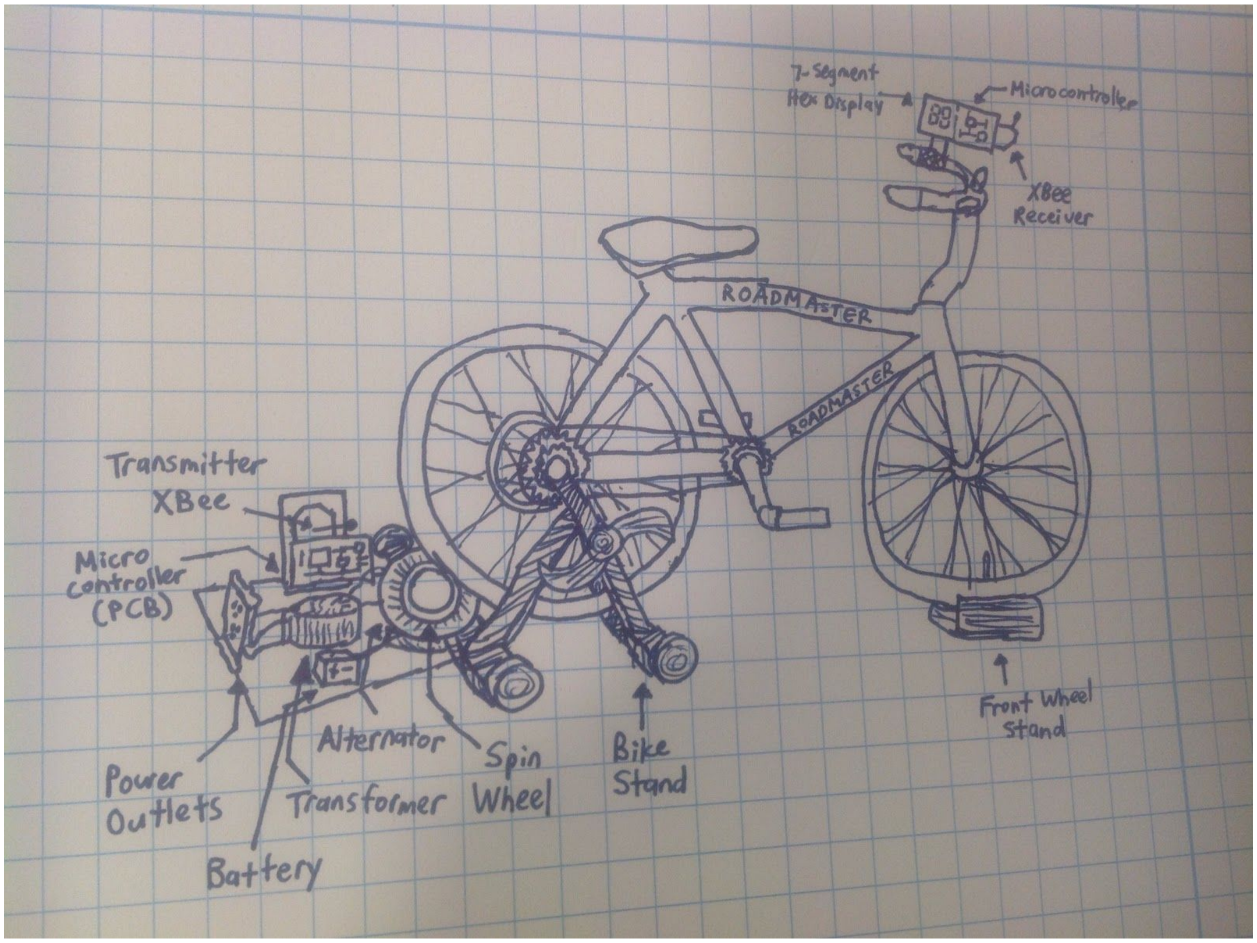

Figure 2: Physical Design Sketch

2.2 Mechanical Systems

The ECE machine shop has created the mechanical system. There are two aspects of the mechanical system. The mechanical system will need to support the bike, and keep it stable, as well as spin the alternator at a fast enough speed for it to generate power. The alternator has a turn on speed of 1415 revolutions per minute [9]. The torque constant at the turn-on rate is 1.28-Newton meters which comes from knowing that the diameter of the alternator wheel is 1.75 inches. As the alternator moves faster, it will generate more current, but will also require a higher torque. The mechanical system will have to spin the alternator at the turn-on rate to generate any power. This is a concern since it requires 190 Watts of mechanical power to run the alternator at that speed and torque. Additionally, there could be frictional losses which will require more mechanical power to overcome. To prevent slipping between the bike and alternator 40 N needs to be applied between them; since the max force produced is 24 N and the coefficient of friction between rubber and steel is .64.

2.3 Alternator

The main part of the mechanical to electrical power system is the alternator. There are two reasons why we picked alternator over its competition, induction motor and dynamo. An alternator has a relatively high power efficiency, in contrast to the induction motor, which loses efficiency to higher leakage reactance [1]. An alternator has a rather reliable output given an input, in contrast to the dynamo, which loses reliability as transmitting DC power is less efficient than transmitting AC power [2]. It should also be noted that most of the dynamos on the market suited for a bike are not capable of generating 25 watts needed for this project as most produce about 6W [3]. The car alternator we bought from Advance Auto Parts has a maximum power generation of 976 watts [4].

Although the alternator provided us the best avenue to produce electrical power, there are two main issues we needed to solve. The first was to design a support system to mount the alternator. Safety of the user was of great concern, so we needed a structure that would be able to withstand the vibrations of a user as he or she is pedaling the bike. We decided to buy a bike stand to hold the bike in a stationary position but still allow movement of its back wheel. We also needed a part that would connect the bike wheel to the alternator. This part ultimately was a spin wheel. The texture on the spin wheel will grip with the texture of bike tires so that there would be less mechanical power lost to slip. The spin wheel was also small enough to provide enough revolutions per minute (RPM) to the alternator but not too small so that the torque required to turn the alternator is very large. In the tolerance analysis, torque is calculated using

$$P = \tau \cdot 2\pi \cdot v \tag{1}$$

To produce 50 watts of mechanical power, the torque required is 7.11 Nm for a biker traveling at 14.2 mph on a 26 inch bike wheel attached to a 1.75 inch diameter spin wheel. We deduce that the gear ratio would be about 15.

$$Gear\ Ratio = \frac{D_{bike}}{D_{spin\ wheel}} \tag{2}$$

We submitted the design to the machine shop. They mounted the alternator on the bike stand but the spin wheel attached is 2.5 inch in diameter. Later, as we tested the power coming from the alternator, this discrepancy proves to not be an issue.

The other issue is to check if the alternator is even compatible with the project. The alternator is originally designed to be connected to car engines. Car engines can run at 2000 - 7500 RPM, which is far larger than the RPM of our bike [5]. This issue became a learning process as we changed the design and experiments while testing the alternator in various ways, all with little success. For example, we initially connected the alternator directly to a small resistor. We did not measure any power across the resistor as we fundamentally failed at understanding the conditions the alternator requires in order to produce electrical power. As we did more research, we realized that the alternator needs a current injected into it and bought a lawn mower battery that would provide enough current to energize the alternator. The results for the learning process will be covered in detail in the Testing and Verification section of this report. Ultimately, with the help of a graduate student working in the power lab, Joe Zatarski, we learned that the alternator produces a rectified AC current as represented in the circuit below. R is the reference voltage and is not used in the design as it is not important.

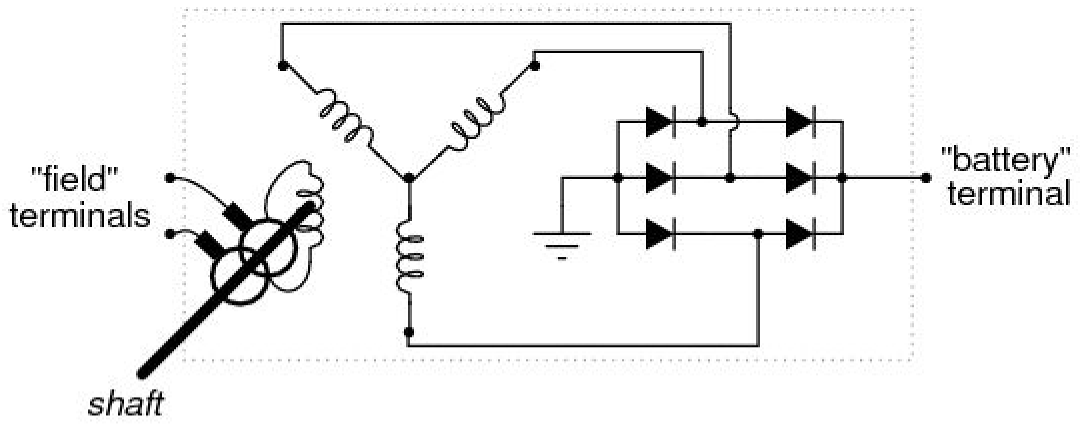

Figure 3: Circuit within the alternator

To power the alternator, we needed to wire it correctly and to the correct components, as shown below:

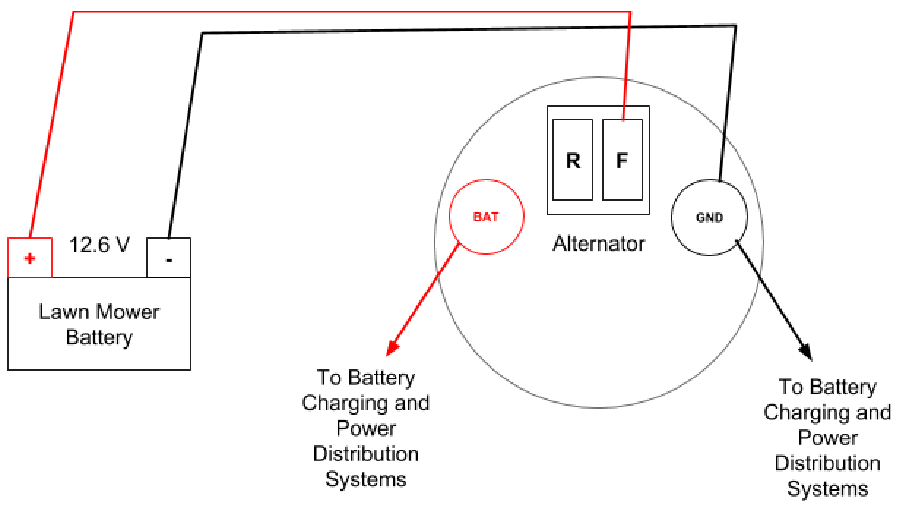
+
12.6 V
-
Lawn Mower Battery
R
F
BAT
GND
Alternator
To Battery Charging and Power Distribution Systems
To Battery Charging and Power Distribution Systems

Figure 4: Circuit connecting the alternator to other parts of the project

We tested the alternator to make sure it has the capabilities to generate 25 watts for the circuit. For the setup, we asked the machine shop to build a suitable shaft for the motor in the Machinery Lab on the fourth floor that will turn the alternator. Before testing, we fixed some loose diodes in the alternator and proceeded to connect the alternator to the motor provided in the Machinery Lab. The Everitt Power Supply is providing the power to energize the alternator, and the current and voltage drawn from the power supply is shown to us. For the test, the alternator ran at different RPM, and the load that was connected to the alternator are low resistors.

The results are listed below on the following table. Various resistors and RPM were used in this test to check if the alternator matches the performance of its datasheet. We calculated the power produced with each load and recorded the torque required.

Table 1: The results of the alternator

RPM	Resistance (ohms)	Current (A)	Voltage (V)	Power (W)	Torque (Nm)
500	3	1.333	4	5.333	0.45
750	3	2.167	6.5	14.1	0.62
1000	3	2.967	8.9	26.7	0.765
1100	3	10	3.333	33.3	0.825
1200	3	3.633	10.9	39.6	0.875
1300	3	4	12	48	0.93
1400	3	4.25	12.75	54.2	0.965
1500	3	4.58	13.75	63	1.03
1600	3	4.92	14.75	72.6	1.09
1000	100	0.1	10	1	0.435
1000	10	0.92	9.2	8.5	0.525
1000	1	6.7	6.7	45	1.14
1400	1000	0.02	20	0.4	0.515
1400	100	0.14	14.14	2	0.52
1400	10	1.32	13.2	17.5	0.655
1400	1	10	10	100	1.525
2500	3	4.67	14	65	0.67

To summarize, the alternator can produce 25 watts with a reasonable amount of torque. We went ahead and replace the motor from the Machinery Lab with the bike.

2.4 Power Inverter

A power inverter a circuit that takes DC voltage and transform it into AC. It was designed using a RC timer and npn and pnp mosfet. The timer would produce a square wave with frequency based on the resistances and capacitor used; and voltage equal to input voltage. The square wave is inputted into the bases of the mosfet, so when one of the mosfet is off, the other is on. The emitters of both mosfets are inputted into a capacitor to filter out harmonics and to allow for reverse current flow. The problem with the design is that it needs a pure DC voltage that is stable. The reason we needed this was because we found out that the alternator gave out half-wave rectified AC, which is DC but with a huge ripple. So a filter needed to be made to filter out the ripple. We ran out of time, and it’s harder to make than a filter for a full-wave rectified AC. The RC timer worked, but the mosfet did not as then voltage supply had a current limiter on it and had reached its maxed, and it cut down on voltage.

2.5 Voltage Regulator

We originally intended to use a buck-regulator as we found that to have maximum efficiency. It was going to use pulse width modulation on a transistor and have a high frequency of turning on and off the transistor. This did not work out since a buck-regulator only converts DC to DC. As well as it misses the point of how the regulator is actually suppose to work. Our previous design had the field current having 12 volts on it which meant that we were always getting high voltages from the output when we were spinning really fast. The voltage regulator is meant to cut current to the field windings when the output voltage gets too high; so as to allow for lower voltages from the alternator when it is being spun really fast. We built the buck-regulator and it worked but then we realized it was going to work. We didn’t have time to build a new regulator. We should have just bought a regulator as we didn’t decide to make this worth points, and we’re not innovating by building our own.

2.6 AC Transformers

Different voltage levels are required by the electrical system. Due to this fact, AC transformers will be required to step up and down the voltage. AC transformers will be purchased. AC transformers will be used to step the voltage level to 120 volts. An AC transformers may also be used to step the voltage down to charge the battery.

$$Eff = \frac{P_{out}}{P_{in}} \tag{1}$$

Efficiency of transformer = 93.75%. According to the transformer, the input is 120 volts and 0.52 amps , the output is 11.7 volts and 5 amps [12]. The input power is 62.4 watts and the output power is 58.5 watts. The efficiency is assumed to be similar when it is operated in reverse [12]. We were originally going to have two transformer so the user could switch between America and Europe's standard, but transformers are expensive so we decided only to have 120 volts.

2.7 Power Monitoring System

The two key components of the power monitoring system are a microprocessor, and a current to voltage transducer. The transducer is used to change the current produced by the alternator into a voltage signal. The voltage divider circuit is used to scale the voltage down and then have the voltage read by the analog input of a microcontroller. The voltage divider circuit will use a very high resistance. This is because a high resistance voltage divider will divert less power from the circuit than a lower resistance voltage divider circuit. To calculate the power produce by the alternator, the instantaneous voltage and current must be measured by the circuit, then instantaneous power is determined by multiplying by the absolute value of the two values. This is not useful by itself since the average power is what is wanted; since instantaneous power cycles between 0 and max value. So the energy of the cycle must be calculated using the riemann sum of the values and divided by the time of a complete cycle. This circuit was never completed as we were trying to get the processor, display, and communication working; and the transducers were ordered too late for them to be in the demo.

2.8 Encoder Circuit

The core of the fitness monitoring system is a 3 phase rotary encoder. An encoder is a device which can be used to determine the position of a mechanical system with predefined precision. A 3 phase rotary encoder has a VCC pin, GND pin, Shield cable and phase A,B, and Z phases. The a and b phases alternate between high and low to create 4 different permutations. This is used for precise control. The encoder used in this project has a count number of 1024 which means that there are 1024 different cycles of A and B phases in one revolution. The encoder used is an absolute encoder, which means that the z phase will drive high when the encoder is in a specific position. This is used to give the interface a reference when a revolution has happened. Due to the size ratios between the bike wheel and the encoder wheel, counting the a and b phases is not necessary to get an accurate measurement of the velocity. This is because it is assumed that the encoder will make around 20 revolutions per time step.

input_number=Q0+2*Q1+4*Q2+8*Q3+16*Q4+32*Q5+64*Q6+128*Q7
Where Q0 through Q7 are the parallel inputs to the parallel to serial converter.
Velocity=input_number*encoder diameter*pi/delta_t
Where encoder diameter=1.75 inches, and delta_t=0.5 seconds

The z phase will drive high when the encoder is in a specific position. However, this results in certain problems. The z phase can not be directly be counted to determine the speed which the encoder spins; because the time duration of the z phase depends on the speed of the bike wheel. If the bike wheel is stationary, it will cause the z phase to drive high for an undefined time. Additionally, the z phase of the encoder is rather noisy, and needs to be processed in order to be useable. To address this, the interface must be able to detect the edges of the z phase. The interface that was used for this project involved TTL chips and a perfboard. The edge detector used was created by using a configurable shift register and a quad xor chip. The register was configured in right shift mode and clocked at 4 MHz. This was accomplished using a crystal oscillator. This frequency was chosen because it was determined that 4 MHz would allow this circuit to reliably detect the encoder edges. The first two flip flops have their outputs connected to an a and b input port on the quad xor chip. The output from that xor chip will then be counted. This edge detector circuit removes the dependence on the speed of the bike, and instead replaces it with a reliance on clock speed.

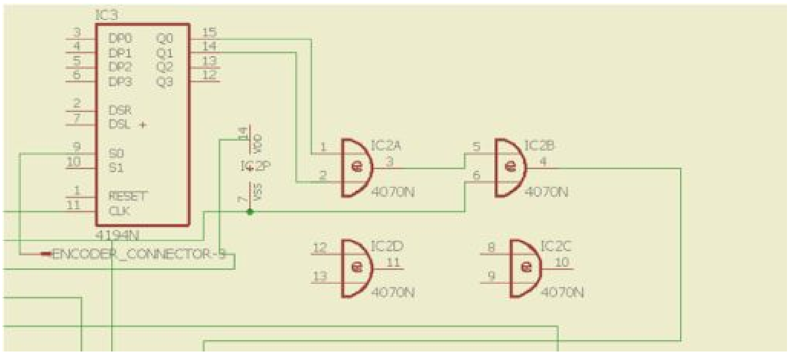

Figure 5: Schematic for Encoder edge detector circuit

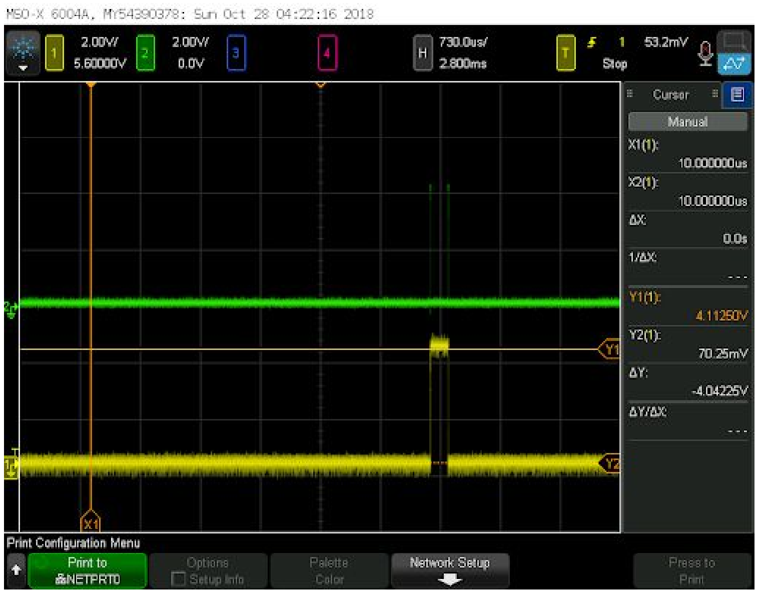

Figure 6: z-phase (yellow), edge detection (green)

The encoder edge detector circuit functioned correctly. It could accurately determine the positive and negative edges of the z phase of the encoder. It also significantly decreased the noise in z phase, and had the additional effect of making sure that the voltage level remained within the thresholds to be counted by the rest of the ttl chips. Additionally, the ripple counter can accurately count the output from the edge detector. The failure in the encoder interface occurred with interfacing the parallel to serial converter with the microcontroller.

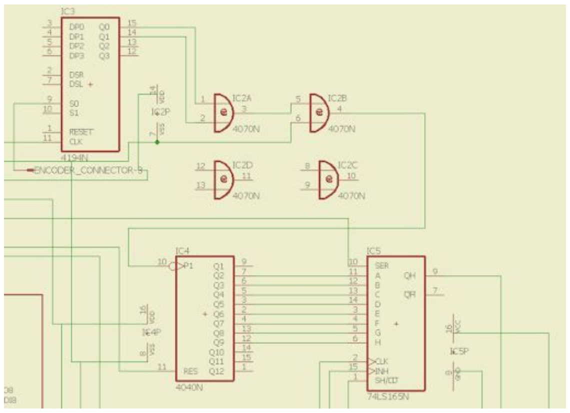

Figure 7: Encoder edge detector with interface to microcontroller

2.9 Processor

The counter that was used is a 12 bit asynchronous counter. This counter was used because it was important to make sure that a high level of precision was achievable. It was determined that

this level of precision would be satisfactory. The counter will have its outputs be sent to a parallel to serial converter which will send 8 bits to the microcontroller.
The Parallel to serial converter will turn its parallel data into serial data. It need asynchronous signals in order to function as expected. These will be provided by the microcontroller. This allowed the velocity calculation to be accomplished with sufficient accuracy.

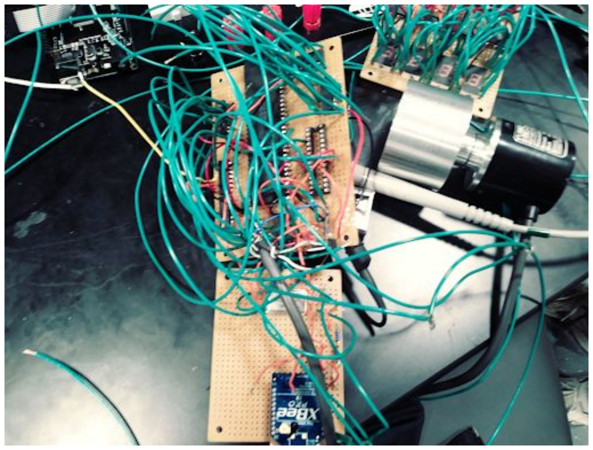

Figure 8: Processor on a Perfboard

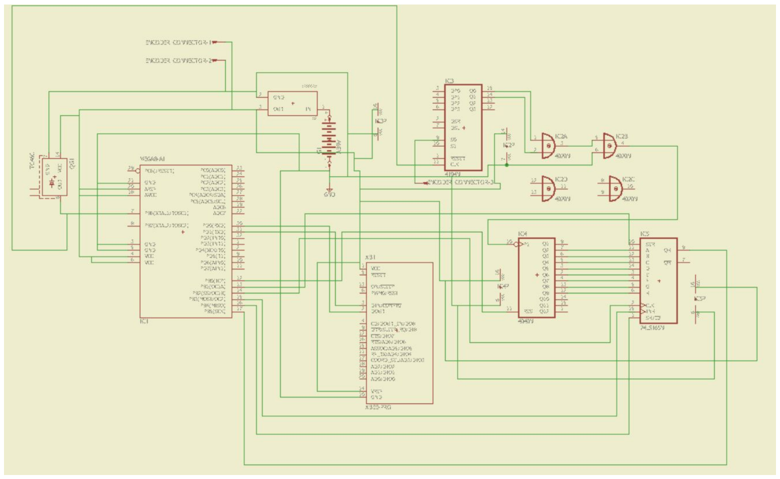

Figure 9: Schematic of the Processor

2.10 Wireless Communication

XBees are wireless chips that can act like serial ports except without a wire. This design is useful as it spares a wire to connect between a display, which is shown to the user at the front of the bike, and the processor, which is located at the back of the bike with all the other parts. The XBee communication modules did not function, causing integration problems.

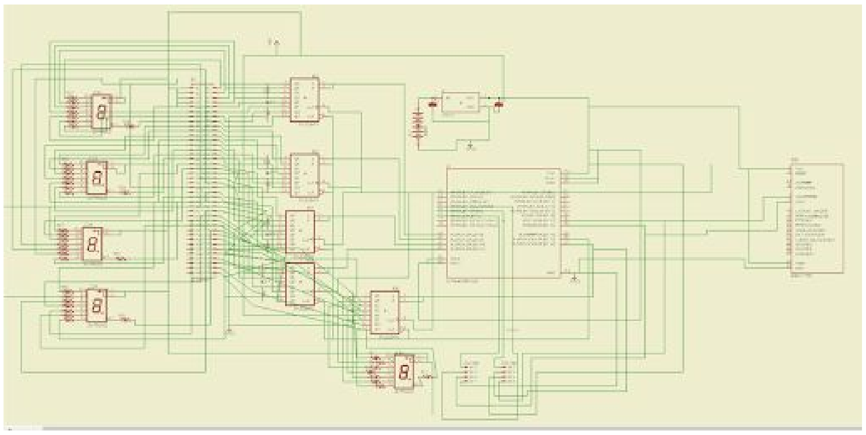

Figure 10: Schematic for Display system

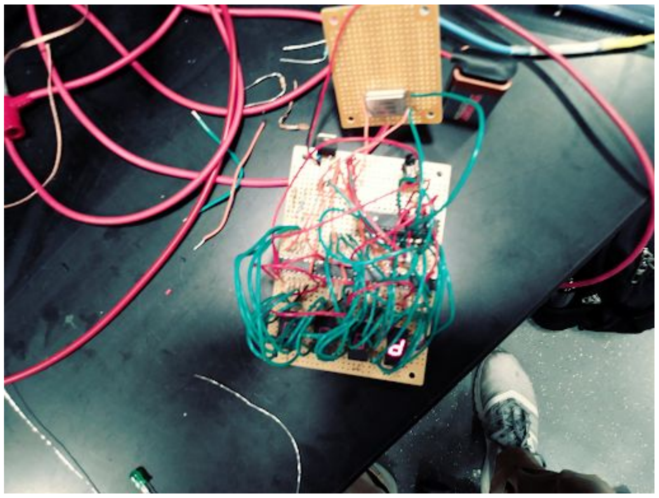

Figure 11: Display System on Perfboard

2.11 Display System

Power will be supplied by a battery to provide a voltage to the display system, which is composed of a microprocessor and a four digit 7-segment display. The microprocessor is responsible for deserializing the input from the XBee into an input that the 7-segment can recognize. The 7-segment display will display the digits of the number as well as a letter to distinguish what stat is being shown. The display system was going to cycle through calories burned, current speed, and energy generated, but it was changed to have a button to change what stat is being shown

The serial to parallel register holds 8 bits of data. This data is shifted into the each register using a data port. Additionally, a clock input is required to shift data. This input is manipulated by a digital output port. The clock input port of every chip is connected to the same digital output port on the microcontroller which allows it to write all 4 values at once.

The protocol first writes the 4 values are written to the correct output ports. Once these values have been written, the clock value will be toggled to give the registers both a positive edge clock signal. This will allow the new data to be written into the registers at one time.

The seven segment hex display contains 7 led segments which are active low. Each segment of the display is connected to a resistor which is connected to an output pin on the register. This allows any number to be arbitrarily written to the four 7-segment displays. The hex displays are common cathode displays which means that the value stored in the register will be inverted when it is displayed on the 7-segment display.

3 Cost Analysis

The complete list of parts and their costs can be found in Appendix D.

The total cost of the parts used in building this project is $205.19. This is rather expensive, but it is still less expensive than the competition. However, if this project was created without the use of the electronic services shop, the cost of parts will be $304.01. This is because many chips were sourced for free from the services shop, and the XBees were borrowed from the ece 445 lab. Additionally, the ece machine shop was very useful in attaching the alternator to the bike stand.

Name	Hourly Rate	Estimate of Hours worked	Multiplier	Labor Cost
Daniel Davidar	$30	180	2.5	$13,500
David Zhang	$30	150	2.5	$11,250
Michael Westfall	$30	90	2.5	$6,750
Machine Shop	$50	40	0.6	$1,200
Totals				$32,700

4 Conclusion

At the end of our semester, for our project, we managed to prove that it is possible to generate 25 watts needed from simply pedaling on a bicycle. We were also able to prove that this energy generation can be done so without a complicated set-up. We even made the 7-segment display to display any 4-digit number and mode we wanted. However, we encountered a few roadblocks that prevented us from transmitting the data from the encoder to the display and we simply ran out of time to transfer the electrical power to the outlet or back to the battery. We took measures to make our project safe, and if we finished, we would not have hazards like overcharging a battery.

5 References

[1] Desai, B. (2014). *What is the difference between induction generator and alternator*. [online] ResearchGate. Available at: https://www.researchgate.net/post/What_is_the_difference_between_induction_generator_and_alternator2 [Accessed 3 Nov. 2018].

[2]Joan, B. (2018). *Difference Between Dynamo and Alternator | Difference Between*. [online] Differencebetween.net. Available at: http://www.differencebetween.net/business/structure-systems/difference-between-dynamo-and-alternator/ [Accessed 3 Nov. 2018].

[3]Walmart.com. (2018). *Dynamo Generator 12V 6W. Bike light part, bicycle lightpart, lowrider,*
beach cruiser, stretch bike, bmx, track, fixie. [online] Available at: https://www.walmart.com/ip/Dynamo-Generator-12V-6W-Bike-light-part-bicycle-lightpart-lowrider-beach-cruiser-stretch-bike-bmx-track-fixie/102496753?wmlspartner=wlpa&selectedSellerId=3792&adid=22222222227121125657&wl0=&wl1=s&wl2=c&wl3=233287467057&wl4=pla-383021674016&wl5=9022196&wl6=&wl7=&wl8=&wl9=pla&wl10=113548444&wl11=online&wl12=102496753&wl13=&veh=sem&gclid=CjwKCAjwsfreBRB9EiwAikSUHfKIbrk5fRNVvyS_1Cxkbwsrkp1qDHwJT2e3Zl34h-vm1kflN3gbphoCH2AQAvD_BwE [Accessed 3 Nov. 2018].

[4] Delco, "ALT-98 Computerized Test Report: A7122." 05-Oct-2016.

[5] En.wikipedia.org. (2018). *Redline*. [online] Available at: https://en.wikipedia.org/wiki/Redline [Accessed 3 Nov. 2018].

[6]"SN54F86, SN74F86 QUADRUPLE 2-INPUT EXCLUSIVE-OR GATES." [Online].

Available: http://www.ti.com/lit/ds/symlink/sn74f86.pdf.

[7]*datasheet MC74HC86N*. [Online]. Available:

https://www.digchip.com/datasheets/parts/datasheet/311/MC74HC86N-pdf.php. [Accessed:

06-Nov-2018].

[8]"Semiconductor electrical data." [Online]. Available:

https://assets.nexperia.com/documents/data-sheet/74HC_HCT86.pdf.

[9]"SN74HC164 (ACTIVE)," *SN74HC164 8-Bit Parallel-Out Serial Shift Registers | TI.com*.

[Online]. Available: http://www.ti.com/product/SN74HC164. [Accessed: 05-Nov-2018].

[10]“74HC(T)194,” *74HC(T)194 - 4-bit bidirectional universal shift register*. [Online]. Available: https://www.nexperia.com/products/logic/i-o-expansion-logic/shift-registers/series/74HC-T-194.html. [Accessed: 05-Nov-2018].

[11]*(Datasheet) MM74C162 pdf - w w w .d e e h s a t a . u t4 m o c (MM74C160 - MM74C163) Decade Counter with Asynchronous Clear*. [Online]. Available: https://datasheetspdf.com/pdf-file/520780/NationalSemiconductor/MM74C162/1. [Accessed: 05-Nov-2018].

[12] “ Getting Started with Atmel Studio 7,” *Atmel Studio 7*. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Getting-Started-with-Atmel-Studio7.pdf.

[13] “Switching Regulator Fundamentals.” [Online]. Available: http://www.ti.com/lit/an/snva559a/snva559a.pdf.

[14] tutorialspoint.com, “Arduino Tutorial,” *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/arduino/. [Accessed: 05-Nov-2018].

[15] “Datasheet.” [Online]. Available: https://www.robotshop.com/media/files/pdf/datasheet-com-11102.pdf.

Appendix A: Main-Side Code

```
#include<avr/io.h>
#include<util/delay.h>
#include<SoftwareSerial.h>
int main()
{
 DDRB = 0xFF; //set all pins of B as input
 //The microcontroller board is transmitting the data.
 DDRD = 0x03; //set pin 0 and pin 1 of D as output (digital pin 21 and 20)
 PORTD = 0x03;
 /*
 //setting up XBee
 SoftwareSerial XBee(2, 3);
 pinMode(2, INPUT);
 pinMode(3, OUTPUT);
 Serial.begin(1200);
 XBee.begin(1200);

 pinMode(4, OUTPUT); //controls the clock of the parallel to serial registers
 pinMode(5, INPUT); //reads from the registers
 */

 //Send the initial values over to the display MC
 int distance_d = 0;
 double distance = 0.0;
 int velocity_d = 0;

  //...XBee serialization and transmission code here
  //XBee.write("7d13pe"); //write a random number
  //_delay_ms(50);
  int mode = 0;
  int output = 0;
 while(1)
 {

  int value = 0;
  for(int i = 0; i < 8; i++)
  {
```

```
  //start by reading from the serial register
  PORTD = 0x03; //set clock high
  value = value + PIND / 4;
  PORTD = 0x02; //set clock low
  value = value * 2;
}

output = value / 2; //canceling the final multiplier by 10...*/
/*
//Reset Counter is controlled by MC pin 4
PORTD = 0x00; //Pin 4 (PD2) [active low, if high, change to 0x04]
_delay_ms(50); //to make sure that the counter resets
PORTD = 0x04; //Pin 4 (PD2) [active low, if high, change to 0x00]

//wait half a second
_delay_ms(100);

//now take the value and compute
int spins = PINB; //Pin 14 - 19 (PB0 - PB5) [6 bits, currently connecting to counter bit 1 to 5]

double gear = 15.0; //Encoder gear diameter is 15 mm [please update this value ***]
double counter_sig = 1.0; //bit 2 - 7 of counter is considered [doubles per left-shift of counter]
double velocity = spins * gear * counter_sig * 3.14159 * 2.5 * 3.6 / 1609.0; //miles per hour
velocity_d = (int) velocity;

distance = distance + spins * gear * counter_sig * 3.14159 * 1.25 / 1000; //in meters
distance_d = (int) distance;*/
/*
  //...XBee serialization and transmission code here using Pin 3 (TX, PD1)
  XBee.write("7d13pe"); //write a random number
  _delay_ms(25);
  */
 /* if(mode == 0)
  {
    int value = 9999 / 256;
    PORTD = value;
    _delay_ms(65);
    value = 9999 % 256;
    PORTD = value;
```

```
        _delay_ms(65);
      }
      else if (mode == 1)
      {
        int value = 9998 / 256;
        PORTD = value;
        _delay_ms(65);
        value = 9998 % 256;
        PORTD = value;
        _delay_ms(65);
      }*/

        unsigned char val = output % 256;
        val = ~val;
        PORTB = val;
        _delay_ms(500);
        PORTD = 0x01; //reset
        PORTD = 0x03;


  }
  //Results show that pin 6 and 7 of Port B are not usable. They are crystal pins.
  //Also pin 5 of Port D is tied high unless forced low by pin B. Everything else appears to be tied
low.
}
```

Appendix B: Display-Side Code

```
#include<avr/io.h>
#include<util/delay.h>
#include<SoftwareSerial.h>

//**NOTE Clock is 4x as slow
int main()
{
 DDRB = 0x3F; //set all pins of B as output but pin 7 and 6
 DDRD = 0x00; //ignore the button
 PORTB = 0x2F; //make MR low.

 /*//setting up XBee equivalent to pin 4, 5 or Port D bit 2 and 3.
 SoftwareSerial XBee(2, 3);
 pinMode(2, INPUT);
 pinMode(3, OUTPUT);
 Serial.begin(300);
 XBee.begin(300);*/

 //Initialize the following values
 int distance_d = 0;
 int velocity_d = 9999;
 int mode = 0; //0 is distance, 1 is velocity
 int prev_button_state = 0; //0 means currently not pressed, 1 means currently pressed [active
high button]
 int value = 0;
 int output = 0; //holds the value to be displayed

  //...XBee deserialization from pin 2 (PD0) [variables are set here]
 /* while(XBee.available())
  {
   char c = XBee.read();
   value = 0; //reset the value upon new variable
   while(c != 'e')
   {
    //read the values and then the mode
    if(c == '0')
    {
```

```
  value = value;
}
else if(c == '1')
{
  value = value + 1;
}
else if(c == '2')
{
  value = value + 2;
}
else if(c == '3')
{
  value = value + 3;
}
else if(c == '4')
{
  value = value + 4;
}
else if(c == '5')
{
  value = value + 5;
}
else if(c == '6')
{
  value = value + 6;
}
else if(c == '7')
{
  value = value + 7;
}
else if(c == '8')
{
  value = value + 8;
}
else if(c == '9')
{
  value = value + 9;
}
else if(c == 'd')
```

```
      {
        distance_d = value;
        value = 0;
      }
      else if(c == 'p')
      {
        velocity_d = value;
        value = 0;
      }
      value = value * 10;
      c = XBee.read();
    }
  }*/

//read the button input
/*int cur_button_state = PIND / 4; //divide by 4 = right-shift of 2
if(prev_button_state == 1 && cur_button_state == 0) //on a release [active high button]
{
  mode = mode + 1; //next mode
  mode = mode % 2; //2 = two different modes, distance, velocity
}
prev_button_state = cur_button_state;*/

while(1)
{
  /*our 7-segment display are active low
   *connections:
   *Q0 = d, Q1 = c, Q2 = g, Q3 = b,
   *Q4 = e, Q5 = ldp, Q6 = f, Q7 = a.
   *      -a-
   *    f|   |b
   *      -g-
   *    e|   |c
   *      -d-
   *  .ldp
   *
   *  However, the way the shift register works, Q7 must be input first and Q0 must be input last
   */
  /*
```

```
//change the mode every cycle
mode++;
mode = mode % 2;

//display mode for a second, we can ignore main-side MC at this time
for(int i = 0; i < 12; i++)
{
  //display takes 80 milliseconds. Only necessary if mode is changed.
  if (mode == 0) //distance, display 'd' on most significant digit, clear all other digits
  {
    printd();
    _delay_ms(20);
  }
  else if (mode == 1) //speed, display 'p' on most significant digit, clear all other digits
  {
    printp();
    _delay_ms(20);
  }

  //read the button input
  cur_button_state = PIND / 4; //divide by 4 = right-shift of 2
  if(prev_button_state == 1 && cur_button_state == 0) //on a release [active high button]
  {
    mode = mode + 1; //next mode
    mode = mode % 2; //2 = two different modes, distance, velocity
    update_display = 1;
  }
  prev_button_state = cur_button_state;
}*/

//display stat of chosen mode for 4 seconds
// for(int i = 0; i < 20; i++)
//{
  _delay_ms(50); //wait

  //read the button input
  /*cur_button_state = PIND / 4; //divide by 4 = right-shift of 2
  if(prev_button_state == 1 && cur_button_state == 0) //on a release [active high button]
  {
```

```
 mode = mode + 1; //next mode
 mode = mode % 2; //2 = two different modes, distance, velocity
}

prev_button_state = cur_button_state;*/
//...XBee deserialization from pin 2 (PD0) [variables are updated here]
/*while(XBee.available())
{
 char c = XBee.read();
 value = 0; //reset the value upon new variable
 while(c != 'e')
 {
  //read the values and then the mode
  if(c == '0')
  {
   value = value;
  }
  else if(c == '1')
  {
   value = value + 1;
  }
  else if(c == '2')
  {
   value = value + 2;
  }
  else if(c == '3')
  {
   value = value + 3;
  }
  else if(c == '4')
  {
   value = value + 4;
  }
  else if(c == '5')
  {
   value = value + 5;
  }
  else if(c == '6')
  {
      value = value + 6;
    }
    else if(c == '7')
    {
      value = value + 7;
    }
    else if(c == '8')
    {
      value = value + 8;
    }
    else if(c == '9')
    {
      value = value + 9;
    }
    else if(c == 'd')
    {
      distance_d = value;
      value = 0;
    }
    else if(c == 'p')
    {
      velocity_d = value;
      value = 0;
    }
    value = value * 10;
    c = XBee.read();
  }
}*/
value = 0;
value = value + PIND;
value = value * 256;
_delay_ms(65);
value = value + PIND;
_delay_ms(65);

/*
if(mode == 0) //distance mode
{
  output = distance_d;
```

```
    }
    else if(mode == 1) //velocity mode
    {
      output = velocity_d;
    }*/

    //do display code here
    if(value == 9999)
    {
      printd();
    }
    else if(value == 9998)
    {
      printp();
    }
    else
    {
      printNum(value);
    }
   //}

  } //end of while loop
  //Results show that pin 6 and 7 of Port B are not usable. They are crystal pins.
  //Also pin 5 of Port D is tied high unless forced low by pin B. Everything else appears to be tied
low.
}

void printd() //takes 80 ms
{
  //first round: a < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2F; //no a's
  PORTB = 0x3F; //activate the clock
  PORTB = 0x2F; //finish the clock cycle

  //second round: f < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2F; //no f's
  PORTB = 0x3F; //activate the clock
  PORTB = 0x2F; //finish the clock cycle
  //third round: ldp < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2F; //no ldp's
  PORTB = 0x3F; //activate the clock
  PORTB = 0x2F; //finish the clock cycle

  //fourth round: e < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //fifth round: b < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //sixth round: g < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //seventh round: c < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //eighth round: d < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle
}

void printp() //takes 80 ms
{
  //first round: a < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //second round: f < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //third round: ldp < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2F; //no ldp's
  PORTB = 0x3F; //activate the clock
  PORTB = 0x2F; //finish the clock cycle

  //fourth round: e < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //fifth round: b < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //sixth round: g < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2E; //digit 3
  PORTB = 0x3E; //activate the clock
  PORTB = 0x2E; //finish the clock cycle

  //seventh round: c < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2F; //no c's
  PORTB = 0x3F; //activate the clock
  PORTB = 0x2F; //finish the clock cycle

  //eighth round: d < 0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
  PORTB = 0x2F; //no d's
  PORTB = 0x3F; //activate the clock
  PORTB = 0x2F; //finish the clock cycle
}

void printNum(int num) //prints a number
{
  int digits[] = {10, 10, 10, 10}; //start out empty
  int number = num;
  //automatically take digit zero
  int digit = number % 10;
  number = number / 10;
  digits[0] = digit;

  for(int i = 1; i < 4; i++) //take the next 3 digits, if possible
  {
   if(number == 0) //less than 4 significant digits
   {
    break;
   }
   digit = number % 10;
   number = number / 10;
   digits[i] = digit;
  }
  int key[11][8] = {{0,0,1,0,0,1,0,0}, {1,1,1,1,0,1,0,1},
                   {0,1,1,0,0,0,1,0}, {0,1,1,1,0,0,0,0},
                   {1,0,1,1,0,0,0,1}, {0,0,1,1,1,0,0,0},
                   {0,0,1,0,1,0,0,0}, {0,1,1,1,0,1,0,1},
                   {0,0,1,0,0,0,0,0}, {0,0,1,1,0,0,0,0},
                   {1,1,1,1,1,1,1,1}};
  //for each round
  for(int i = 0; i < 8; i++)
  {
   //0-0-mr-clk-dg0-dg1-dg2-dg3 in binary
   PORTB = 0x20 + key[digits[0]][i] * 8 + key[digits[1]][i] * 4 + key[digits[2]][i] * 2 +
key[digits[3]][i];
   PORTB = PORTB + 0x10; //set the clock
   PORTB = PORTB - 0x10; //reset the clock
  }
}
```

Appendix C: Requirement and Verification Table

Table : System Requirements and Verification

Component	Requirements	Verification
Power Monitoring System	1. The power monitoring system will need to accurately calculate the power generated during operation. The expected amperage produced will be between 15 and 20 amps at 12 volts dc +/- 5%. The power monitoring system will be calibrated to be very accurate in this range. The goal will be at least 90% accuracy. [5 points] 2. The processor should be processing rate of at least 10 kHz. It can be higher.[5 points]	1. A. Using the oscilloscope, measure the voltage and current using the two probes and calculate the power being produced. B. Have the monitoring system calculate power being produced. Compare the two different output power and make sure they do not have a 10% difference. 2. A. Set the clock rate of the processor to highest possible setting. B.Connect the clock signal to an oscilloscope and check to make sure it is above 10 kHz.
Encoder Circuit	1. The encoder circuit will accurately detect edges in phase z, to make sure the processor accurately counts turns of the wheels.[5 points] 2. The encoder circuit	1. A. Oscilloscope traces of the encoder circuit will be included,the circuit must count once per turn regardless of the speed at which the encoder is turning.

	will be created on a pcb, which interfaces with the processor.[5 points]	B. The encoder will need to accurately resolve the speed and distance the wheel has travelled. 2. The PCB must have the components required for the encoder to count, and must interface reliabally with the processor, this will be verified using oscilloscope traces.
Processor	1. The processor must calculate distance travelled in meters to within 10% based on encoder input. [4 points] 2. The processor must calculate energy produced from the generator to within 10% in kilojoules. [6 points] 3. The processor must calculate speed in meters per second in tenths of meters to within 10% based on encoder input. [4 points]	1. A. Connect the bike to the spin wheel of the alternator. B. Using hands on the rear wheel, spin it for a ten cycles. C. Check the bits output to the XBee. The number should be between 19 and 22. Calculations above. 2. A. Connect the bike to the spin wheel of the alternator. B. Connect a power analyzer to the outlet. Alter the resistance connected in series with the alternator until the load is matched. C. One person should ride the bike. The other person checks the bits output to the XBee. D. Multiply the current squared with the resistance to get the wattage. Divide by 1000 and check that the output of the XBee is within 10%. 3. A. Connect the bike to the spin wheel of the alternator.

		B. One person use his hands on the rear wheel and spin it steadily for ten cycles. Another person records the input and time to complete ten cycles using a stopwatch. (1 cycle is equal to 2.07 meters with calculations above.) C. Count the number of cycles in the recording, and compute speed.
Wireless Communication	1. XBee must transmit 80 bits to the other XBee correctly. [5 points] 2. Takes less than 1 second to transmit between the two Xbees. [1 point]	1. A. 80 bits of low and high voltages will be sent into transmitting pin of a XBee one at a time. B. A voltmeter will measure the voltage of the receiving pin of the other XBee and see if all 80 bits are displayed correctly. 2. A. Use a stopwatch to measure the time it takes to get a update on pins.
Display System	A binary representation of a 4 digit number will be passed into the microprocessor via XBee. The 7-segment display must show that specific 4 digit number. [10 points]	A. Check to make sure XBee is operational. B. Send a binary representation of a 4 digit number into one of the XBee. C. Check that the 7-segment hex display reveals the correct 4 digit number. D. Check to see that display changes what current value is being shown (between the three values), every 5 seconds.

Appendix D: Parts

Table : Parts Cost

Part Description	Part number	Quantity	Unit price $, and reference	Supplier for this project
Microcontroller	Atmel Atmega328p-pu	2	2.14	Digikey
2 X 2 3/8 SOLDERABLE PERF BOARDS	590103350	2	$1.43	ECE supply Store
1 1/2 X 1 3/4 SOLDERABLE PERF BOARDS	590103300	2	$0.92	ECE supply store
3 Phase rotary encoder	YUMO E6B2-CW3ZE	1	39.95	Robot Shop
4 MHZ Crystal oscillator	NA	2	1.95(Jameco Electronics)	Electronic services shop(free)
5 Volt Linear regulator	L7805 CV	2	0.52(Digikey)	Electronic services shop(free)
Parallel to serial converter	SN74LS165N	4	0.27(Digipart)	Electronic services shop(free)
4 bit left shift-right shift register	M74C95N	1	Obsolete	Electronic services shop(free)
Quad XOR gate	SN74LS85n	1	1.43(Mouser)	Electronic services shop(free)
Serial input parallel output register	SN74LS164N	5	0.74(Digikey)	Electronic services shop(free)
LED HEX display	LSD 3211-1	4	2.81(Digikey)	Electronic services shop(free)

12-bit ripple counter	MC14040BCL	1	Obsolete part, 0.53(Alternate product)	Electronic services shop(free)
32-bit Multiplier (Not integrated)	CY7C510-75PC	1	Obsolete	Electronic services shop(free)
Wireless communication module	XBee Pro	2	37.95 (AdaFruit)	Borrowed from ECE 445 lab (free)
Through hole resistor ¼ watt	NA	16	NA	ECE 445 Lab(free)
Automotive Alternator	NA	1	79.95	Advance Auto Parts
Bike Trainer stand	NA	1	46.99	WalMart
Sparkfun pocket programmer	1568-1080-ND	1	16.95	Digikey
9 volt battery two pack	Energizer Max	1	6.59	Target
9 volt battery clip	X000Y10Y43	2	2.89	ECE Supply center