Power Regeneration for BLDC Bicycle Motor

University of Southern Indiana
Pott College of Science, Engineering, and Education
Engineering Department

8600 University Boulevard
Evansville, Indiana 47712

Power Regeneration for BLDC Bicycle Motor

Evan Fleming and Luis Lopez
ENGR 491 – Senior Design
Spring 2021

Approved by: ______________________________

Faculty Advisor: Mina Asghari Heidarlou, Ph.D. Date

Approved by: ______________________________

Department Chair: Paul Kuban, Ph.D. Date

ABSTRACT

An e-bike is a bicycle that that has an electric motor as an additional power source. There are two common types of e-bikes, pedal assist, and throttle only. The motor of a pedal assist e-bike outputs a power proportional to the output power of the rider. The second type of e-bike is throttle only. In this type, the user controls the speed of the motor via a throttle. The purpose of this project was to design a throttle only e-bike with power regeneration conversion system that will have a top speed and enough power to be used for daily commuting, while using a readily available power supply that is rechargeable without external electrical power. The system must also be easily integrated, affordable, and easy to use.

During the design phase of the project, the main components considered were the rectifier, boost converter, and the charging circuitry. A custom battery and rear wheel mount were also designed, which were modeled and then 3D printed/fabricated to be utilized to keep the battery in a stable position and allow the rider to reproduce power while stationary.

There were some issues with the project with the time constraints and deadlines, therefore the project's scope was reduced to focus on the recharging. During testing of the charger, it was shown that, while the charger was designed to charge at 2A, when tested the max charging current ever obtained was 85mA. As a result of the charger not being able to operate correctly, using just the boost converter to recharge the battery was tested. The boost converter had adjustable voltage and current output and was configured with the skillet as a load before implementing the battery as the load. The results with just the rectifier and boost converter showed what was expected with the multimeters indicating the battery being charged at a current of 2.08A and a voltage of 25.19V. The system was therefore successful in charging the battery

even without the IC charger. The disadvantage of not using the charging circuitry is a higher inaccuracy in the charging setpoints and less control over the charging operation. Therefore, it would still be desirable to implement the IC charger into the system in the future.

The final cost of the system was $542.59 although, if chosen, the cost of the recharging system as an add-on to existing e-bikes, is $58.81.

Some future considerations to increase the usability and efficiency of this system are creating a single board recharging circuitry consisting of the rectifier, boost converter, and charging circuitry. Allowing the charge current and float voltage used to control the recharging parameters to be adjustable. Integrating the power regeneration mode within the motor controller, and finally designing a custom buck-boost converter for a wider range of usable voltage.

Table of Content

S

List of Figures

List of Tables

Power Regeneration for BLDC Bicycle Motor

1 INTRODUCTION

The inspiration for this project originated with the idea of further empowering those in Third World countries who are not able to afford expensive means of transportation. In these regions, bicycles are a very common and affordable mode of transportation for the middle and lower classes. Those that are more fortunate can afford small combustion motor scooters. However, these motor scooters are a large source of pollution in these countries. Thus, it is desirable to provide a cleaner alternative to motor scooters, as well as an economic upgrade to bicycles.

The goal of this project is to design an affordable electric bicycle to fit these needs. Initially, the proposed plan was to design the entire e-bike system with riding mode along with recharging mode. However, due to issues with ordering the motor and time constraints, the project was limited to the design of the recharging portion of the system by purchasing a conversion kit, which came with completed component for riding mode, reducing the scope of the project. To be affordable to those most in need, the design would need to be considerably cheaper than existing electric bikes conversion kits. The electric bike should also be capable of use anywhere in the world. This would mean that there would not always be access to electrical power from a grid or generator. Therefore, the design must be capable of power regeneration and charging of the system's battery.

2 BACKGROUND

To begin the discussion of this project, some background knowledge about the subject is beneficial; the first being the definition of an electric bicycle. An electric bike or e-bike generally has a very similar construction and operation to a traditional bicycle. The main difference between the two is that an e-bike has an electric motor that can provide power to the bike in addition to or instead of the rider's power. There are a couple variations as to how that additional power is controlled. The first method is pedal assist. With a pedal assist e-bike, the rider can simply start pedaling and the motor will detect the pedaling and begin providing power to reduce the effort needed by the rider. The second method, called throttle only, uses a throttle, typically on the handlebar, to give the rider control over the motor. With throttle only control, it is not necessary for the rider to pedal .

Next, we will briefly discuss the two most common motor types used on e-bikes, the rear hub drive and the mid drive. The rear hub drive motor, shown in Figure 1, integrates into the rear wheel of the bike and acts as the rear axle. The torque of this motor is applied directly to the wheel.

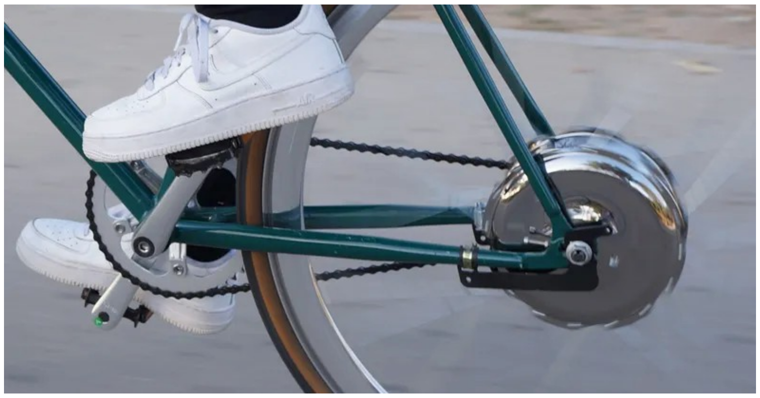


Figure 1: Rear Hub Drive Motor

The second motor type, the mid drive motor, is demonstrated in Figure 2. As shown, this motor is built into the crankset of the bike. In this design, the motor applies torque to the bike's chain drive, just like the rider. This makes most riders feel that the mid drive motor gives an experience closer to that of a traditional bike.

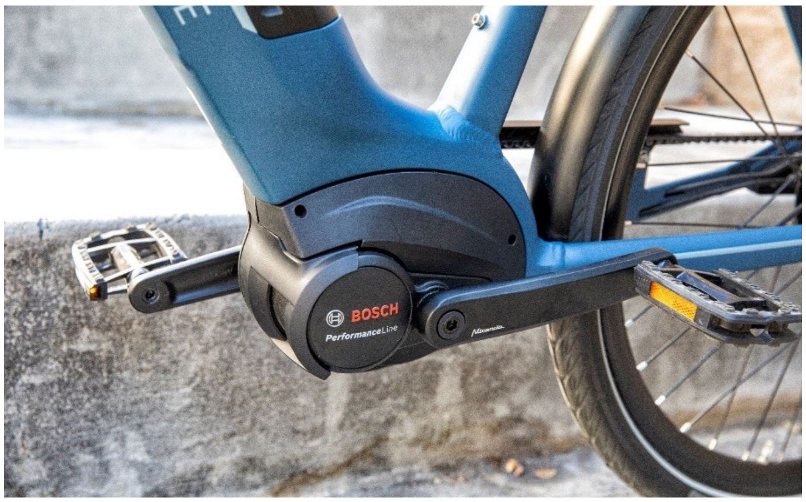


Figure 2: Mid Drive Motor

2.1 *Marketing Requirements and Objective*

The objective of the project is to design a brushless DC (BLDC) Ebike conversion kit focusing on the recharging control system. The motor must be powered from a readily available power source, the system must be easy to integrate with existing bikes, and affordable. The marketing requirements are listed below.

1. The bicycle's top speed and power must be reasonable for daily commuting.
2. The motor must use a readily available power supply.
3. Charging the power supply should be possible without external electrical power.
4. The system must easily integrate with a preexisting bicycle.
5. The system must be affordable.
6. System must be easy to use.

2.2 *Existing E-bike systems*

There are two main types of e-bike system on the market, an e-bike with the motor and controls already integrated within the bike itself, and a conversion kit to convert your bike from a traditional bike to electrically assisted.

Shown in Figure 3, is the RadCity Step-Thru 3 electric commuter bike with a 750W direct drive hub motor with regenerative braking, and a 48V 672Wh lithium-ion battery. Priced at $1599, this e-bike is not very affordable for the everyday consumer. Another example is shown in Figure 4, this is a conversion system that will convert your traditional bike to an e-bike without any power regeneration system. This system includes the front wheel, motor controller, controller bag, pedal assist crank Sensor, twist throttle, and one pair of brake handles, for a total cost of $269.99 without the battery. The battery that is compatible with this conversion system, Figure 5 , has a cost of $229, for a total cost of the system at $498.99.

Figure 3: RadCity Step-Thru 3 electric commuter bike

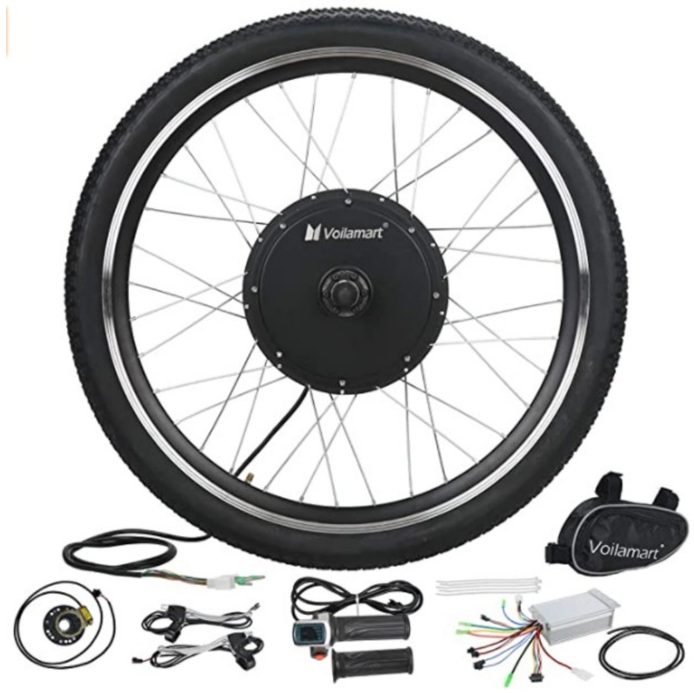


Figure 4: Voilamart Ebike Conversion Kit

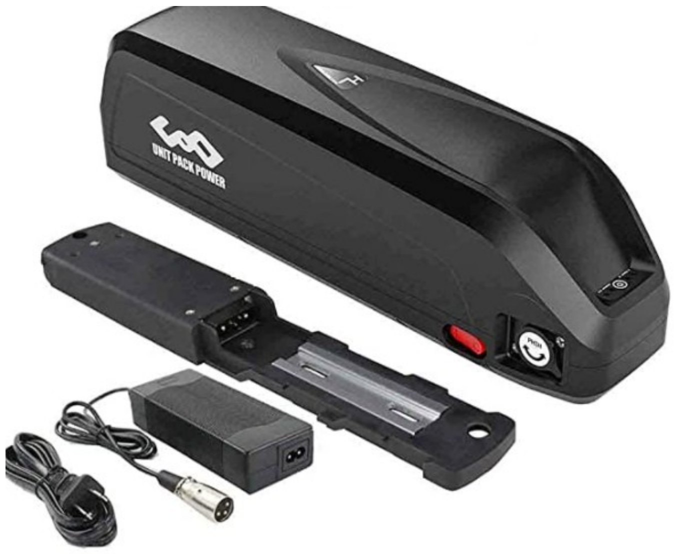


Figure 5: 36V Bike Battery for 200-1500W Bafang Voilamart

3 E-BIKE POWER CONSIDERATIONS

To get a concept of the motor required a power calculation will need to be solved first. The equations that were used are shown next in Table 1.

Table 1: Equations for Power Calculations

$Cd = coefficient\ of\ drag$ (1)
$p = density\ of\ air \left(\frac{kg}{m^3} \right)$ (2)
$A = frontal\ area\ (m^2)$ (3)
$v_w = wind\ speed \left(\frac{m}{s} \right)$ (4)
$v_g = ground\ speed \left(\frac{m}{s} \right)$ (5)
$M = total mass\ of\ bike \wedge rider\ (kg)$ (6)
$G = coefficeint\ of\ slope$ (7)
$C_r = coefficient\ of\ rolling\ resistance$ (8)
$F_{rollingresistance} = 9.81 * M * C_r * \cos \alpha$ (9)
$F_{sloperesistance} = 9.81 * M * C_r * \sin \alpha$ (10)

$F_{wind resistance}=\frac{Cd*p*A*(v_g+v_w)^2}{2}$ (11)
$F_g=9.81*M*sin\alpha$ (12)
$Max\ velocity=29.33\frac{m}{s}(20mph)$ (13)
$acceleration=\frac{Max\ velocity}{time}$ (14)
$F_{acceleration}=$¿¿ ¿ $+F_g+mass$¿$*acceleration$
$F_{velocity}=F_{rolling\ resistance}+F_{slope resistance}+F_{wind\ resistance}+F_g$ (16)
$Power\ for\ acceleration=F_{acceleration}*wheel radius*wheel\ speed$ (17)
$Power\ for\ constant\ velocity=F_{velocity}*wheel\ radius*wheel speed$ (18)

A Python script, which is included in Appendix C: Python Script for Power Calculations, was used to code these formulas to quickly calculate new power value based on changes in the inputs. Using a max speed of 20 MPH, total mass of 175lb, coefficient of rolling friction of 0.004, time to get to max speed of 15 s, grade level of 0, coefficient of drag of 0.7, density of air of 1.225 (kg/m^3), and an area of 0.3716 m^2; the equations 1-9 , produce a power required of 564.45 watts for acceleration and 141.67 watts for constant velocity.

4 SYSTEM ARCHITECTURE

The level 1, 2, & 3 system architecture for the project are presented in Figure 6, Figure 7, and Figure 8.

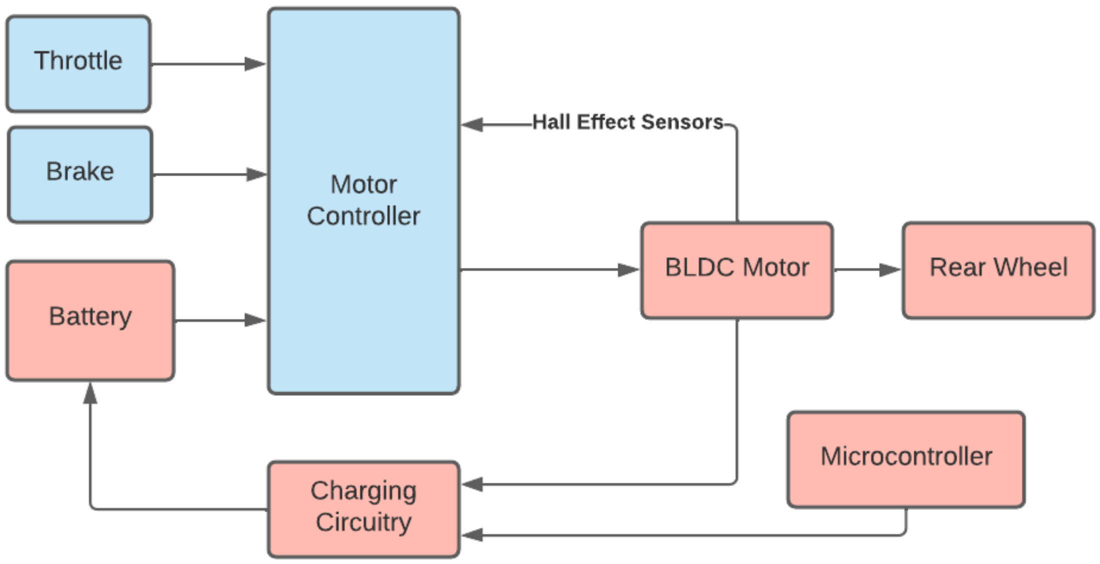


Figure 6: Level 1 Architecture

The system's level 1 architecture is color coded in blue and red diagram boxes. These boxes consist of the main components used for the riding mode and power regeneration mode of the e-bike. The riding mode, in blue, includes primarily the throttle, brakes, and motor controller. In riding mode using the throttle and brakes, the user has full control over the power delivered to the rear wheel. The blue diagram boxes components were not designed in this project but purchased and integrated with the power regeneration system. The red diagram boxes are all part of the power regeneration mode which consists of the rear wheel hub motor, microcontroller, the charging circuitry, and the battery. In power regeneration mode the motor is used as a generator. The motor sends power to the charging circuitry which is controlled by the microcontroller and finally the charging circuitry sends power to the battery.

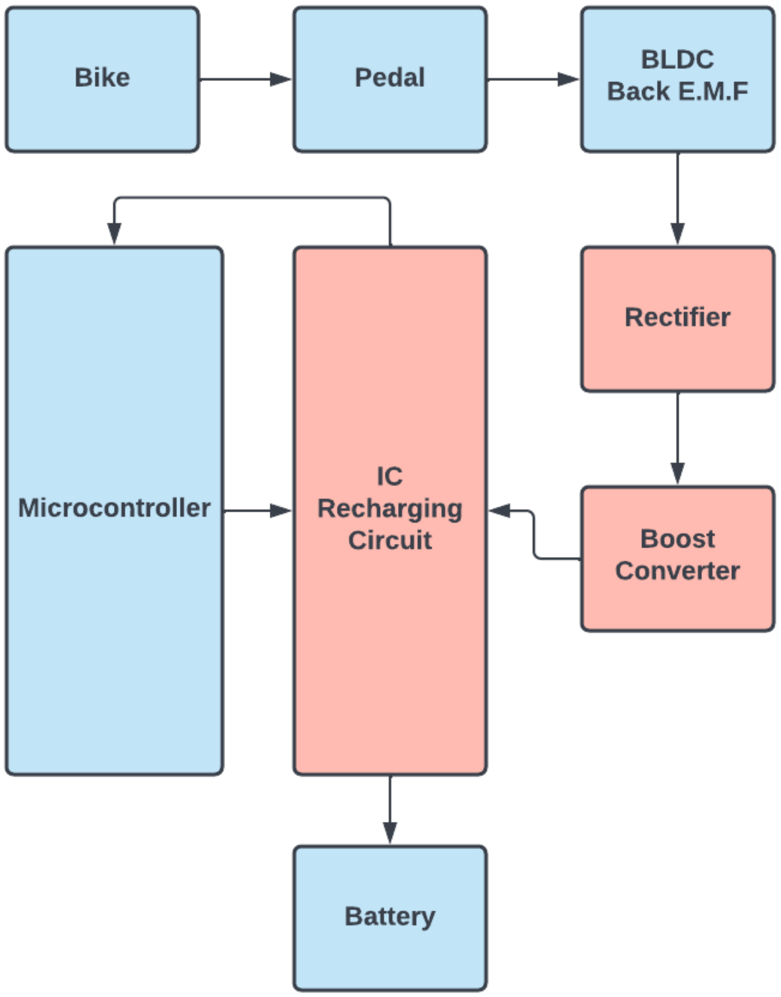


Figure 7: Level 2 Architecture

Figure 7 shows a step-by-step process of the recharging system, system's level 2 architecture. In red consists of all the circuitry needed to recharge the battery. The first step is applying the system on a bike, then pedaling or rotating the rear wheel, a back emf AC voltage is generated which then moves through a rectifier that converts AC Voltage to DC Voltage. After being rectified the voltage is boosted to a constant fixed value required by the IC recharging circuit. The IC circuit then sends a signal to the microcontroller which controls the recharging parameters. Finally, the IC circuit sends the final current and voltage to the battery for recharging.

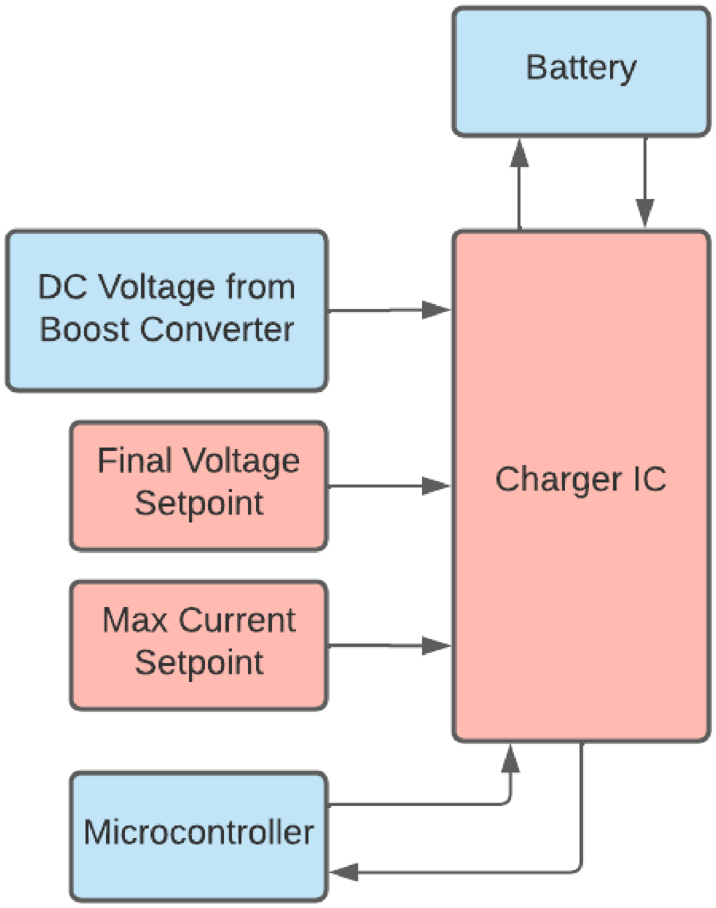


Figure 8: Level 3 Architecture

In the diagram above, the system level 3 diagram shows the inputs and outputs of the charger IC. In the red diagram boxes, are the setpoints that the charger IC uses to limit the power sent to the battery. The input consists of the DC voltage from the boost converter, signals from the microcontroller, the final voltage setpoint, and max current setpoint. These setpoints are used to switch from constant current to constant voltage charging. The output is the feedback to the microcontroller and power to the battery.

5 MOTOR SELECTION AND INTEGRATION

5.1 Initial Motor Selection

The selection of the motor was done via an analytical hierarchy process (AHP). The weightings chosen for the AHP are shown in Error: Reference source not found.

Table 2: Weights for Motor AHP

	Max power	Max torque	Cost	Physical size	rpm at max torque	weight	efficiency	GM	Norm
Max power	1	3	1/5	1/3	5	3	3	1.369	0.150
Max torque	1/3	1	1/5	1/3	1	3	3	0.795	0.087
Cost	5	5	1	7	5	5	3	3.875	0.426
Physical size	3	3	1/7	1	1/7	3	5	1.156	0.127
rpm at max torque	1/5	1	1/5	7	1	7	1	1.101	0.121
weight	1/3	1/3	1/5	1/3	1/7	1	1/7	0.285	0.031
efficiency	1/3	1/3	1/3	1/5	1/5	7	1	0.521	0.057
								9.101	1

Cost was a very important criteria for the selection of the motor and this is importance is reflected in the cost row of the table. Once the weightings were normalized, they were used to compare five different alternatives for the motor selection. Error: Reference source not found lists these five alternatives.

Table 3: Alternatives for Motor Selection

Alternatives	Company	Model	Type
1	Unite Motor	MY1020	brushed
2	Banggood USA	BM1418ZXF	brushless
3	AMG Power solution	BM1109	brushless
4	RobotDigg	60BLS-400W	brushless
5	ATO	ATO-80WDM02420	brushless

These alternatives were selected based on several factors. The first factor was the type of motor. A brushless direct current (BLDC) motor was preferred due to its long life, low maintenance, and efficiency. The second factor was the rated power of the motor. Based on the power calculations, a motor around 500W would be needed; thus, three of the alternatives were 500W motors. A third factor was speed. To minimize the amount of gearing needed in the design, a motor with a low rated speed was desired.

Once the alternatives were entered into the AHP, the scores were computed for each alternative. Table 4 shows that alternative 4 received the largest score and won the AHP.

Table 4: Alternatives' Scores for AHP

	Weights	**Alt 1**		**Alt 2**		**Alt 3**		**Alt4**		**Alt 5**	
Max power	0.150	0.147	0.022	0.147	0.022	0.147	0.022	0.118	0.018	0.441	0.066
Max torque	0.087	0.135	0.012	0.121	0.011	0.121	0.011	0.091	0.008	0.533	0.047
Cost	0.426	0.274	0.117	0.144	0.061	0.206	0.088	0.318	0.135	0.058	0.025
Physical size	0.127	0.065	0.008	0.624	0.079	0.065	0.008	0.154	0.020	0.094	0.012
rpm at max torque	0.121	0.191	0.023	0.214	0.026	0.214	0.026	0.229	0.028	0.153	0.018
weight	0.031	0.116	0.004	0.116	0.004	0.116	0.004	0.386	0.012	0.264	0.008
efficiency	0.057	0.258	0.015	0.258	0.015	0.258	0.015	0.226	0.013	0.824	0.047
Score			**0.185**		**0.203**		**0.158**		**0.220**		**0.176**

According to Error: Reference source not found, alternative 4 corresponded to the brushless motor, 60BLS-400W. Thus, the 60BLS-400W was selected to be utilized for the project.

5.2 MOTOR INTEGRATION

To drive the bike forward the motor will need to transmit the power from the motor to the wheel of the bicycle. To achieve this, a sprocket will be added to the shaft of the motor and connect by a chain to the biggest gear on the bicycle existing cassette. The problem that can arise from this method is that the torque created by the motor can cause bending based on the material the carrier is made from. Some simulated testing will need to be done after the max torque produced from the motor is acquired. Figure 9 show a cassette on the rear wheel of the bike, in which the biggest gear closest to the wheel spokes will be used to transmit the power from the motor to the wheel. A safety device will be designed to prevent the chain from the pedal from interfering or overlapping with the motor's chain.

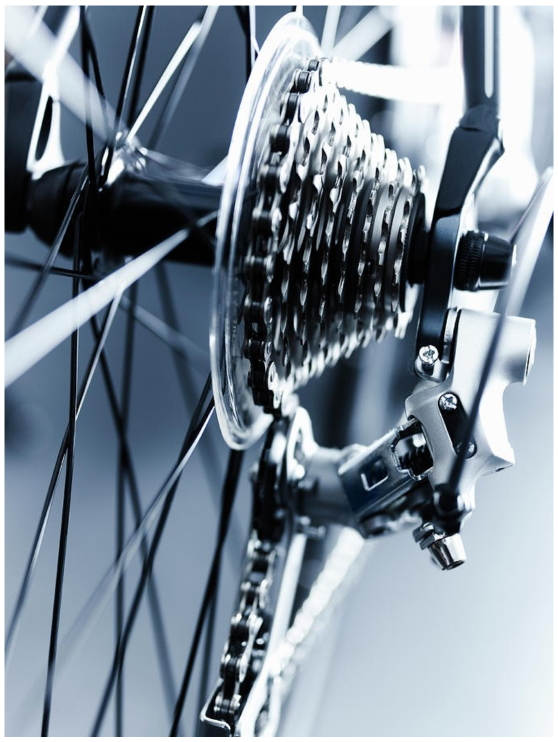


Figure 9: Close up of bicycle's gear

5.3 Motor Control and Drive Considerations

The control of the e-bike's motor is crucial to its operation. The current concept for this control is a throttle that the rider will have access to on the handlebar of the bike. This throttle will send a signal to the microcontroller corresponding to the speed that the user desires. The microcontroller will then translate that speed into the needed armature voltage of the motor and the corresponding duty cycle for the pulse-width modulated (PWM) signal being sent to the motor drive. This PWM signal controls the MOSFETs that will be discussed next.

Motor drives for BLDC motors are typically composed of a power stage and a gate driver. Most commonly, the power stage of the driver uses a half H-bridge circuit for each phase of the motor. Most BLDC motors, including the motor selected for this project, have three phases. Thus, the power stage typically has three half H-bridge circuits. Motor drives come in several topologies, including external FET (MOSFET) and integrated FET. The external FET topology uses a gate driver and MOSFETs that each come in separate packages of housings. Whereas, in the integrated FET topology, these components are housed in one package or can even be on the same die or semiconductor chip .

Since the rated current of the selected motor is 12A, a motor driver that uses a topology of external power stage MOSFETs, similar to that shown in Figure 10, would be ideal. Some of the benefits of this design are the ability to use MOSFETs with lower drain-to-source resistance and better heat dissipation due to the larger area of the board.

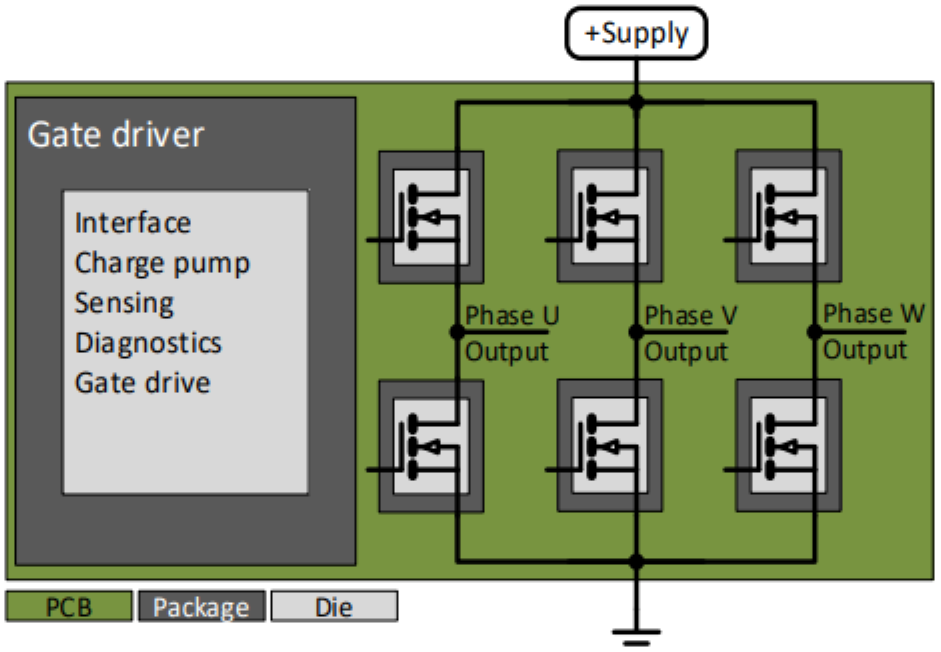


Figure 10: External FET Topology

5.4 Secondary Motor Selection

The motor selected was to be purchased directly from RobotDigg website. There were issues ordering this motor as the shipping date kept being delayed. The motor selected was not found on any other retailer's website and as the deadline for this project was quickly approaching, we decided to look into ordering an alternative. The four other alternatives listed in Error: Reference source not found were searched up and they all had shipping lead times of 1-2 months. A 1-2 month waiting period would put us behind 3 months on this project as the remaining component needed to be purchased depended on the tests that would be done on the motor. Given this predicament we decided to search for an e-bike conversion system and reduce the scope of the project.

Our main focus of the project changed to focus on the recharging mode design and implementation. Taking this into consideration we chose a e-bike conversion system with the same criterion as Error: Reference source not found. The conversion kit chosen is shown in Figure 11, the kit contained a 1000W 48V rear wheel hub motor, throttle, brakes, Pedal Assist System (PAS), motor controller, and some wiring and storage bags. Note that batteries were not included in this conversion kit. The total cost of this kit was $259.00.

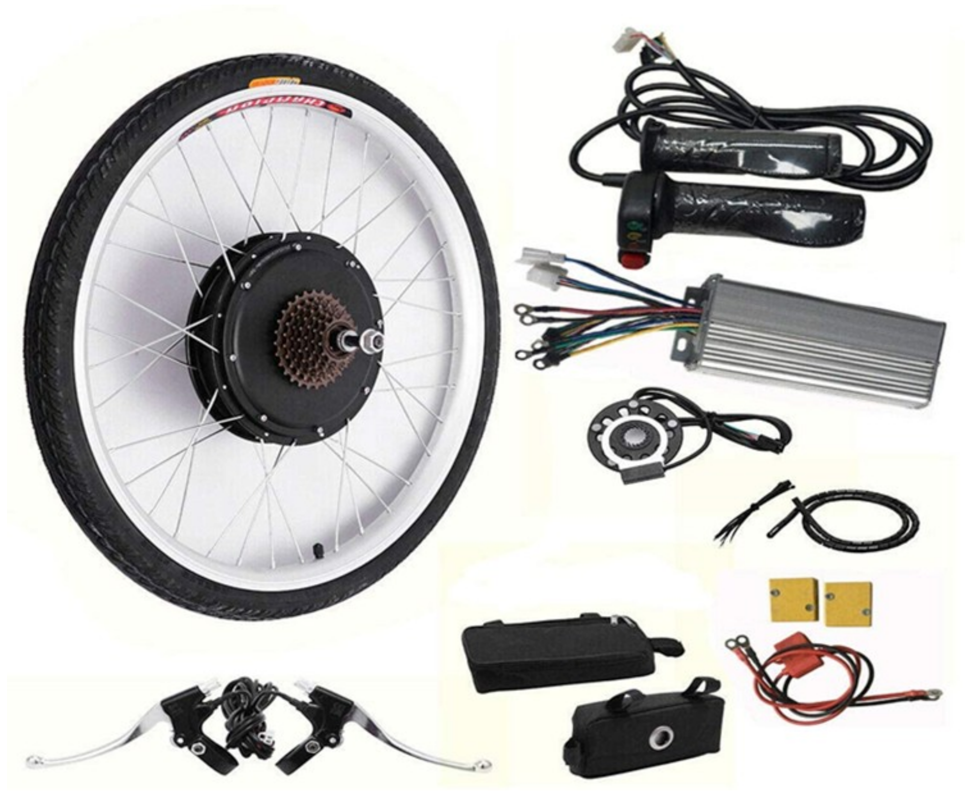


Figure 11: Electric Bicycle Rear Wheel Ebike Conversion Kit - 48V 1000W

6 BATTERY CONSIDERATIONS AND SELECTION

After the selection of a motor, an appropriate power source for the motor needed to be selected. To meet the requirements for the system, this power source had to be a portable power source. Since the system must be both portable and usable in remote locations, external power supplies such that draw power from an electrical outlet were immediately eliminated from consideration. Thus, the motor's power source needed to be a type of electrical battery. Furthermore, this battery needed to meet several criteria including power requirements for the selected motor, size constraints for portability, and energy capacity to ensure the motor can be used for a practical length of time. From these constraints, the first step in the selection was choosing the appropriate battery chemistry.

6.1 Battery Types

In selecting a chemistry type for the battery, all chemistry types that are non-rechargeable were first eliminated as the needs of the system required a rechargeable battery. From the chemistry types of rechargeable batteries, those considered were Lead-acid, Nickel-Cadmium, Nickel-metal Hydride, and Lithium-ion. Lead-acid batteries can offer a high-power output at low cost but at a low voltage and a large physical size. Nickel-Cadmium batteries can provide high current, but they are rarely found in high voltage packs such as needed for an ebike motor. The third chemistry type, Nickel-metal Hydride, has higher energy capacities than Nickel-Cadmium and is capable of high currents similar to Nickel-Cadmium. However, Nickel-metal Hydride has an individual cell voltage of 1.2V, meaning many cells would be required to obtain the needed voltage range for the motor. The final chemistry type was Lithium-ion. This chemistry type is also capable of high currents (up to 30A per cell in some), has a high energy density, and has a high cell voltage (approximately 3.6V per cell), making high voltage battery packs common. For these reasons, Lithium-ion was chosen for the chemistry type of the system's battery.

6.2 POWER CONSIDERATIONS

After selecting a chemistry type for the battery, specifications like voltage, energy capacity, and max current needed to be considered for the selection of a battery. From the specifications of the selected motor, a battery of 48V was required. This limited the selection to either a battery pack that was listed at 48V or multiple battery packs that would be wired in series to obtain the needed 48V.

The next consideration was the battery's energy capacity. This specification is typically measured in amp-hours (Ah) for a given battery, and it tells how many amps the battery can supply for 1 hour. For instance, a 2800 mAh battery would be capable of supplying 2.8A for 1 hour or 1.4A for 2 hours. For this project, the high energy capacity was desired as this would allow the user to utilize the motor for a longer duration before requiring a recharge of the battery. To obtain an idea of the needed energy capacity, some calculations were performed. For these calculations, it was assumed that the rider averaged riding at 50% of max power, which for the selected motor was 500W. The need energy capacity for a ride time of 30 minutes was then determined as shown in Equation 10.

$$energy\,capacity = \frac{power}{voltage} \times time = \frac{500\,W}{48\,V} \times 0.5\,hours = 5.2\,Ah$$

$$(19)$$

From this calculation, a battery rated for 5200mAh would be required for riding the ebike for 30 minutes at 50% of max power.

The last power consideration was the max current required by the motor. From the selected motor’s specifications, the maximum current that the motor was designed to draw was 20A. Therefore, the selected battery must be capable of supplying at least 20A. Typically, this rating is not listed for a battery pack. However, it can be determined if the model of the Lithium-ion cells is known for the battery pack.

6.3 BATTERY MANAGEMENT SYSTEM

One other consideration that was evaluated when selecting a battery was that of an included battery management system (BMS). The BMS of a Lithium-ion battery is essential to the safe operation of the battery. The BMS is responsible for ensuring that the battery is never overcharged or overdischarged. These operations are critical as overdischarging the battery can cause permanent damage to it, and overcharging the battery can cause overheating, possibly even leading to explosion . Additionally, the BMS monitors both the current drawn from the battery and the temperature of the battery pack. If either property becomes too high, the BMS will disable the battery until operation can return to normal. Finally, the BMS also ensures the voltage of each cell remains balanced. Maintaining balanced cells are important; otherwise, the higher voltage cells will have more current drawn from them, possibly harming the cell.

6.4 BATTERY SELECTION

With the previous criteria in mind, three batteries were selected as options for the final selection. These three batteries were a 48V Lithium-ion battery made by SuperHandy, a 2-pack of 24V Lithium-ion batteries made by Jialipok, and a 2-pack Lithium-ion of 24V batteries made by Kobalt. These batteries, along with some of their specifications are listed in Table 5.

Table 5: Battery Selection Alternatives

Quantity	Brand	Voltage	Capacity	Total Price
1	SuperHandy	48V	2Ah	$119.99
2	Jialipok	24V	3Ah	$89.98
2	Kobalt	24V	4Ah	$134.00

One benefit of the battery from SuperHandy is that only one battery pack would be required since the pack has a voltage of 48V. This battery was not selected, however, due to its low capacity at only 2Ah and relatively high price for the capacity. This narrowed the selection to the batteries from Jialipok and Kobalt, both of which would require two battery packs to obtain the needed 48V. Ultimately, the selection was made in favor of the battery from Kobalt because of its higher capacity, which was greatly needed to make riding the ebike practical. Specifications of the battery showed that the Lithium-ion cells within the battery were the INR21700-40T cells made by Samsung. The datasheet for these cells stated that the maximum continuous discharge current was 35A, well above the required maximum current of 20A. Sources such as YouTube teardown videos of the Kobalt battery were also consulted to determine if the battery included a BMS. While the videos showed that the Kobalt battery did include a BMS, once the batteries were obtained for the project, the cover of one of the batteries was removed as shown in Figure 12 to ensure the presence of a BMS.

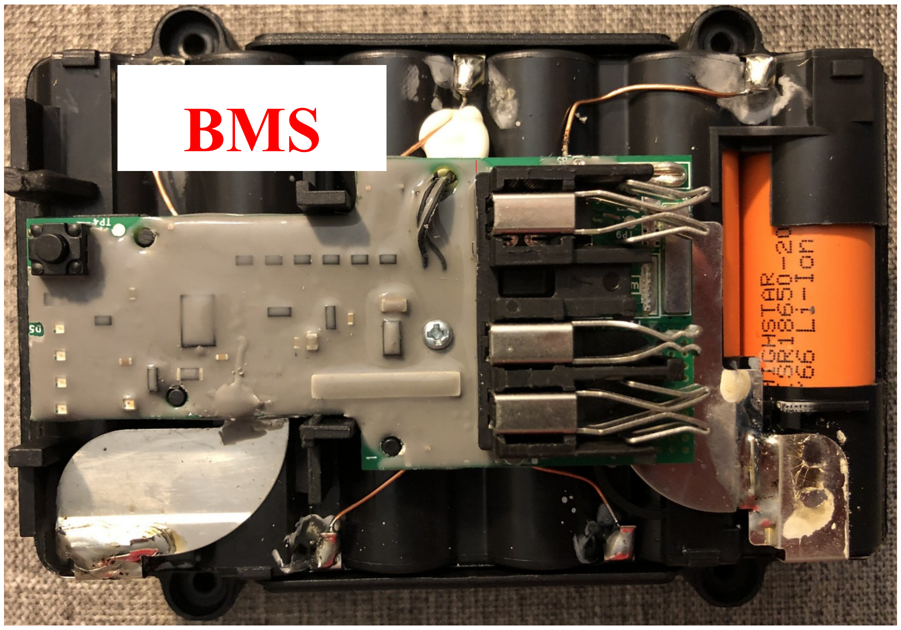


Figure 12: Battery Management System inside Kobalt Battery

Once the battery was confirmed to have a BMS, the batteries could be safely implemented into the system.

6.5 BATTERY MOUNTING

With two 24V Kobalt batteries were used as the power source we needed a way to keep them in series and in a stable position while being in use and while recharging. The first step was to design a mount with positive and negative leads. We achieved this by using Autodesk Inventor, Figure 13 shows the model of the mount and the Original Prusa i3 MK3S+ 3D printer.

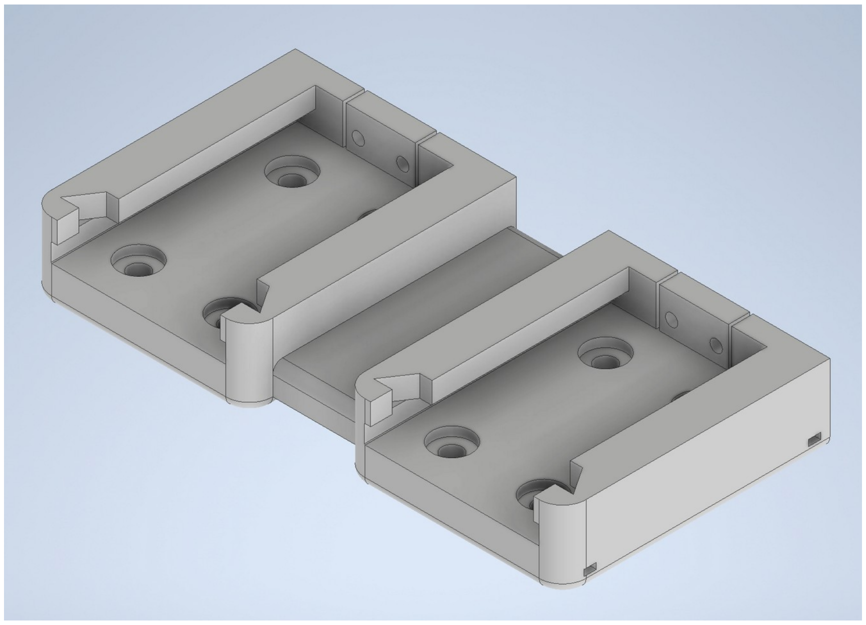


Figure 13: CAD model of battery mount

For the positive and negative leads slots and holes were added to the mount to allow for copper pins to be mounted on the mount and wires to be spliced on. After testing with a single model with a battery to ensure of proper fitment a new model was created with two mounts connected by a middle sleeve, Figure 14 shows the printed model with two battery slots.

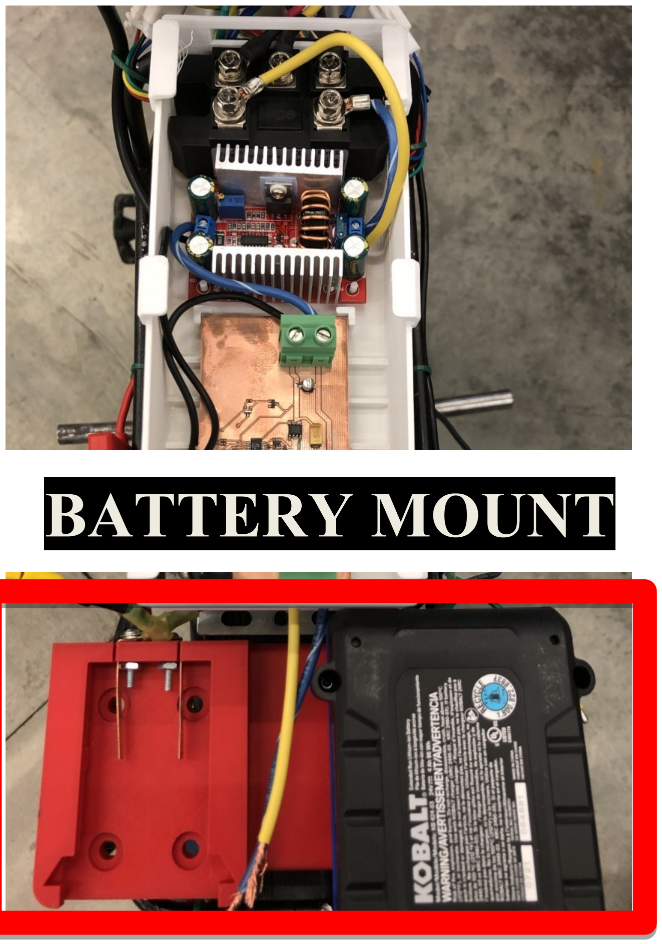


Figure 14: Shown in the red box is the battery mount with 1 battery removed

7 ENERGY REGENERATION

7.1 CONSIDERATION AND SELECTION

There are a few different ways to generate energy. The ones that were considered in this project where regenerative braking and solar charging. The downfall of solar charging is the cost associated with design and implementation along with cost of the solar panels. Heat is a major waste of energy and is most often created when braking or friction occurs to slow the movement of a moving object. One way we can capture this energy is by regenerative braking. This method takes the energy and stores it back into the power supply or a capacitor to be ready to use. Figure 15 below shows an example of a system operational diagram for a regenerative braking system.

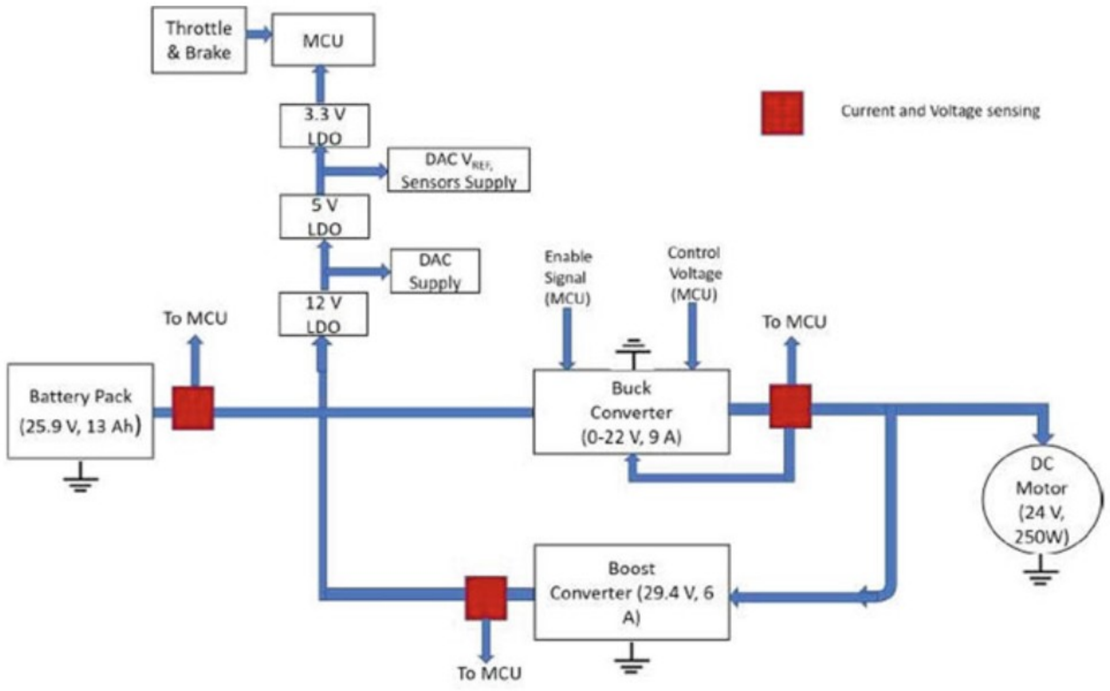

Throttle & Brake
MCU
3.3 V LDO
DAC V_{REF}, Sensors Supply
5 V LDO
DAC Supply
12 V LDO
Current and Voltage sensing
Enable Signal (MCU)
Control Voltage (MCU)
To MCU
To MCU
Battery Pack (25.9 V, 13 Ah)
Buck Converter (0-22 V, 9 A)
DC Motor (24 V, 250W)
Boost Converter (29.4 V, 6 A)
To MCU

Figure 15: Regenerative braking system operational diagram

Taking into consideration of the additional cost associated with solar charging, a form of regenerative braking was chosen.

7.2 REAR WHEEL STAND

Since we would be able to regenerate the energy while pedaling, an option to regenerate energy while stationary will also be added to this system. To be achieve this the rear wheel needed to be elevated to allow rider to pedal while stationary. The max payload, stability of the structure, and cost was taken into consideration. The chainstay member of the bike was chosen as the contact points for the mount since it provided the most stability and would not allow the mount to interfere with the pedaling. The Figures presented below are the inspiration for our design of a mount. Figure 16, was the main inspiration where we based the contact point onto the chainstay of the bike as wheel as have the contact points adjustable on the mount, the main downside of this stand was the integrity as this particular stand has a max payload of 12 lbs. Figure 17 is a mount made of wood and had the contact point on the rear derailer of the bike, the main downside of this mount is that it was bulky. Lastly, Figure 18 was the sturdiest because it was meant for a motorcycle and as well easily to use as a lever arm mechanism was used to lift up the bike.

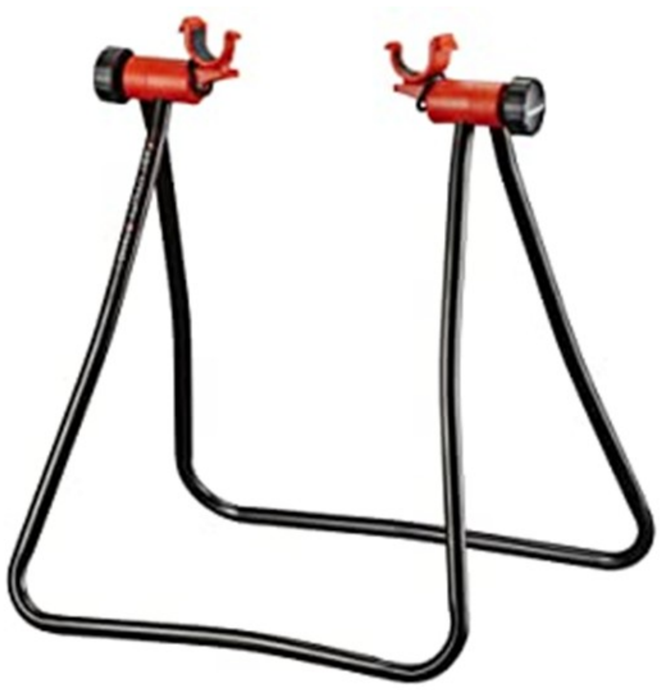


Figure 16: Triangle Vertical Foldable Bike Stand

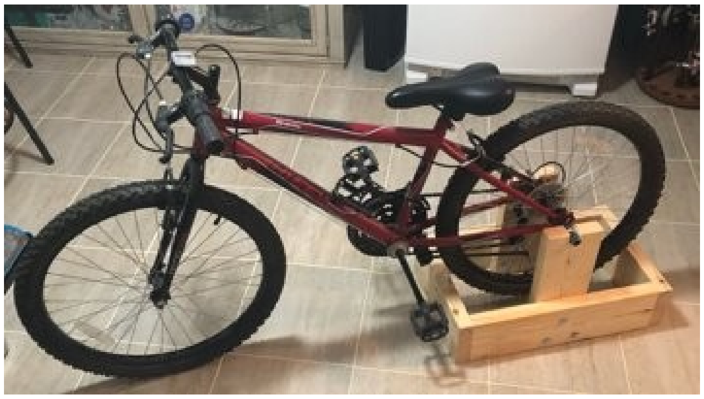


Figure 17: Custom made wood rear wheel mount stand

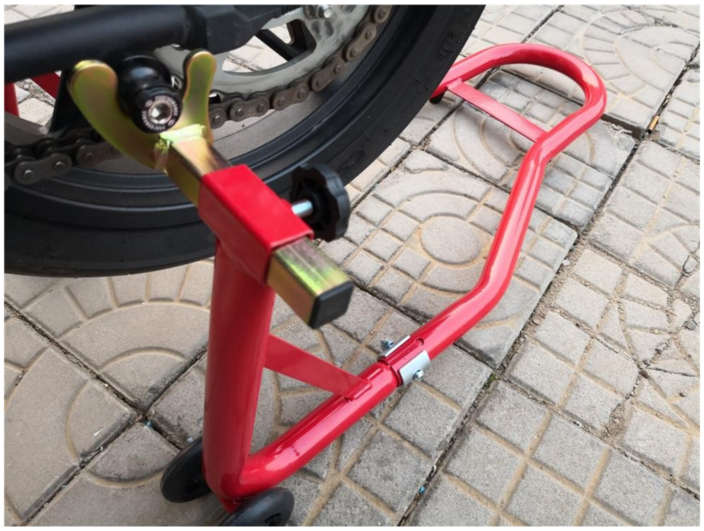


Figure 18: Motorcycle rear wheel stand

Using these previous mounts as inspiration, a custom rear wheel stand was designed using Autodesk Inventor. The modeled version is shown in Figure 19, with a triangle portfolio and adjustable contact point. The fabricated stand is shown in Figure 20, with some changes made to the contact point, mainly the shape composition. Instead of a screw adjustable contact point a round tube is pulled in or out depending on the separation length on the chainstay members.

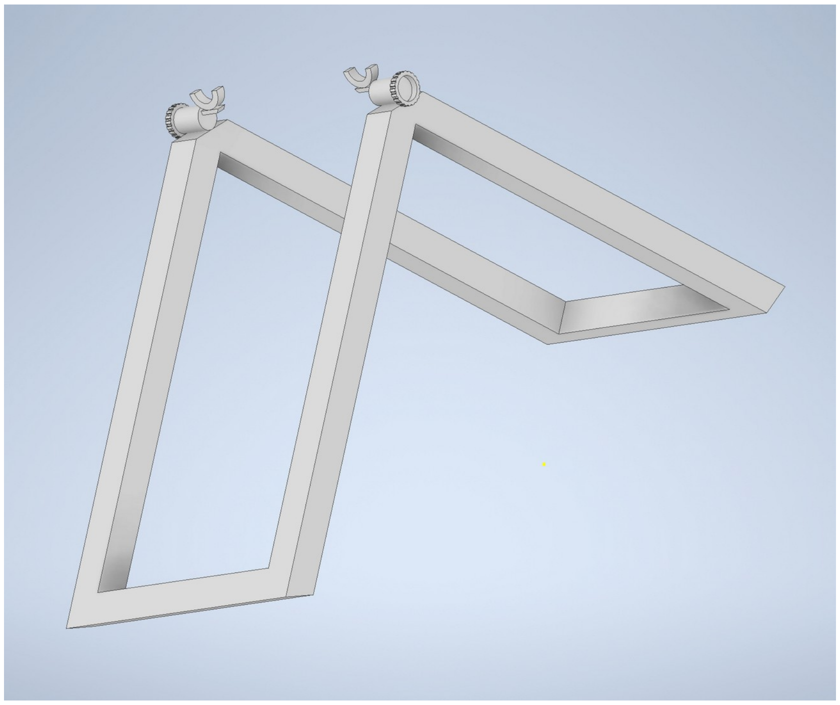


Figure 19: CAD model of rear wheel stand

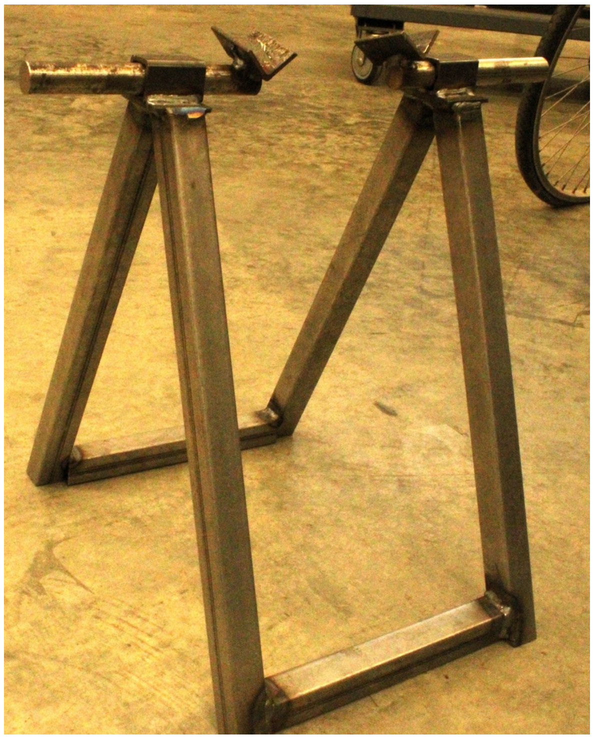


Figure 20: Fabricated Rear Wheel Stand

8 MOUNTING OF THE SYSTEM

The mounting of the recharging system was straightforward since the bike used had a rear rack preinstalled. The controller was bolted down, while the battery mount was zip tied down. A custom enclosure was model, and 3D printed to secure the rectifier, boost converter, and IC recharging circuit. A side view is shown in Figure 21 while a top view is shown in Figure 22. For the ebike conversion kit installation was problematic because of the handlebars on the bike. Because of this, the throttle was zip tied to the right handlebar. The motor was an easy installation by just removing the rear wheel and installing the new rear hub motor integrated within a wheel. The pedal assist system was not used, nor the brakes since the installation would have become a safety hazard. The original front brakes were used for stopping while the rear had to be removed to allow clearance for the new rear wheel.

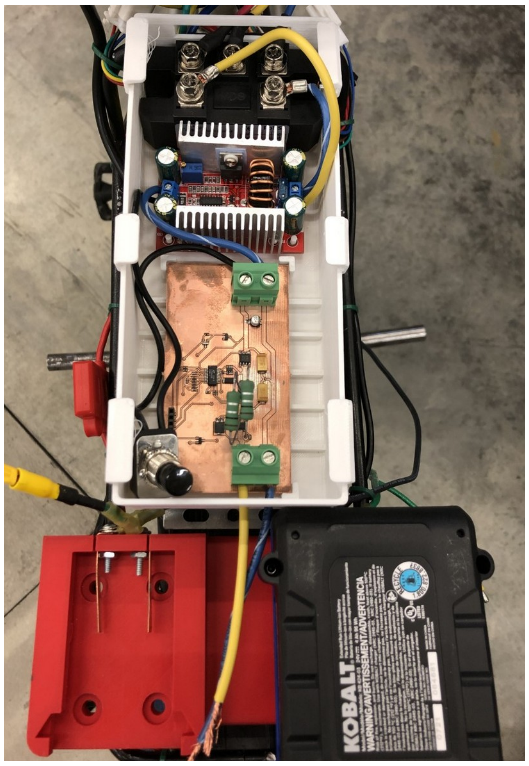


Figure 22: Final Design Top View

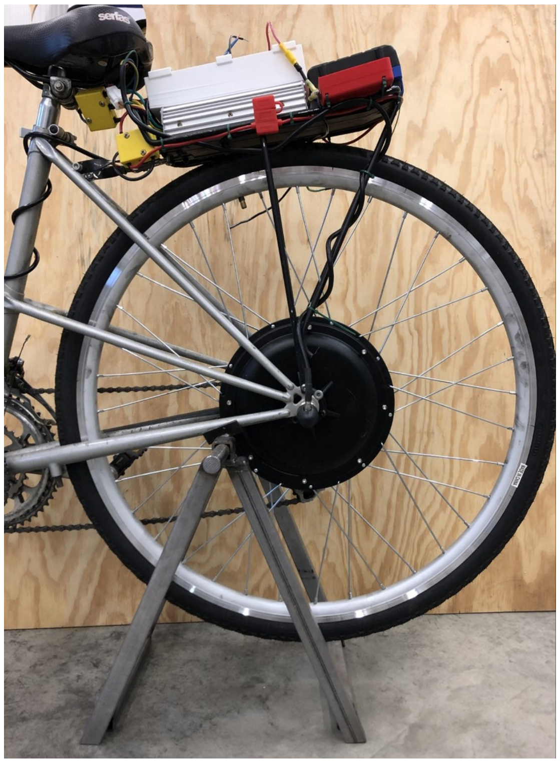


Figure 21: Final Design Side View

9 CHARGER SELECTION AND DESIGN

9.1 Charger IC Selection

Based on the Lithium-ion battery that had been selected for the system, the Kobalt 24V Max 2-pack, a charger that would meet the required battery specifications needed to be chosen. Initially, an idea was to use the charger produced by Kobalt for the battery. However, that charger was designed specifically for electrical outlets, requiring an input of 120 V_{rms} at 60 Hz. Since the speed of the user's pedaling determines the electrical frequency and the user's speed cannot be strictly regulated, the Kobalt charger was named an inviable option.

Other options for a charger included third-party Lithium-ion chargers. But when these options were looked into, chargers were only found for single Lithium-ion cells and batteries with a voltage less than 24 V. The decision was then made to investigate custom Lithium-ion chargers. This required researching the charging techniques used for Lithium-ion batteries.

Some of the most common charging methods include constant current (CC), constant voltage (CV), and constant current constant voltage (CC-CV). Both CC and CV charging can be simple to implement, but they also have their downsides. CC charging often leads to bad utilization of the battery's entire capacity. While CV charging can introduce high charging currents that stress the battery or even cause permanent internal damage. Meanwhile, CC-CV charging, a combination of the previous two techniques, uses constant current to charge during its first phase and constant voltage during its final charging phase . The transition point between phases is typically called the battery's final float voltage, and it is defined by the chemistry of and number of cells within the battery. Through the combination of the two charging methods, CC-CV charging obtains a higher efficiency than either CC or CV charging. For this reason, CC-CV charging was chosen as the charging method to be utilized in the project. The operation of CC-CV charging can be further understood looking at Figure 23.

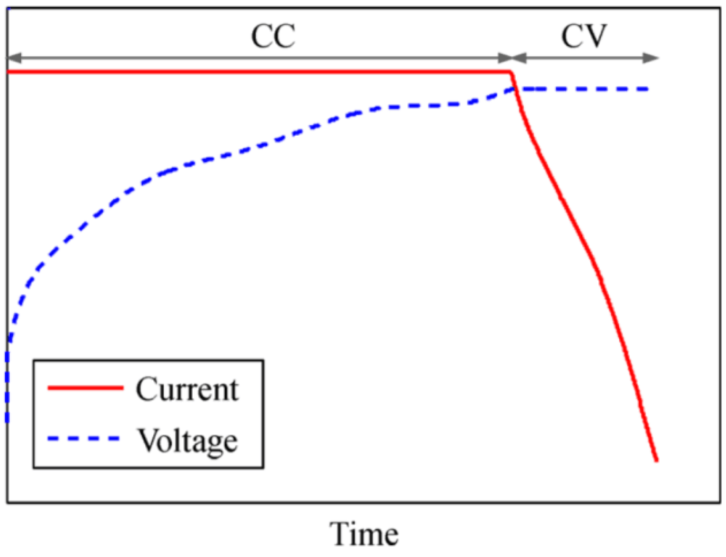


Figure 23: Graph of Constant Current Constant Voltage Charging

During the first phase of charging the current being supplied to the battery remains constant while the battery's voltage steadily increases. Eventually that voltage reaches a preset value, previously mentioned as the battery's final float voltage. At this point, the charger switches into constant voltage charging. During this phase, the charging current continuously decreases as the battery continues charging. The charger then typically cuts off when the charging current reaches 10% of the constant current value.

To implement the constant current constant voltage charger, a charger integrated circuit (IC) with that ability needed to be chosen. Before making the selection, a few other specifications were needed, the charging current and charging voltage. Both of these specifications could be found in the datasheet for the specific Lithium-ion cells, which for the selected battery was the INR21700-40T from Samsung. According to item 3.4 of the datasheet, the standard charge for a cell consists of constant current phase of 2 A, a constant voltage phase of 4.2 V, and a cutoff current of 200 mA. Additionally, since the battery consists of six cells connected in series, the constant voltage spec must be multiplied by six to obtain the proper constant voltage value. Thus, the charger requirements consisted of an output voltage of 25.2 V, a charging current of 2 A, and a cutoff current of 200 mA.

For the selection of the charger IC, Digi-Key and Mouser, two popular electronic component distributors, were used. Using the previously determined charger specifications, the options for a charger IC were quickly limited to the line of LTC4008 charger ICs from Analog Devices. Furthermore, the only charger that was in stock at the time was the LTC4008EGN-1. Therefore, it was chosen for the design.

9.2 PRINTED CIRCUIT BOARD DESIGN

After selecting the LTC4008EGN-1, a custom printed circuit board (PCB) needed to be designed to support the charger IC. The design for this PCB was largely based on an example application within the datasheet for the LTC4008. The schematic of this example application is shown in Figure 24.

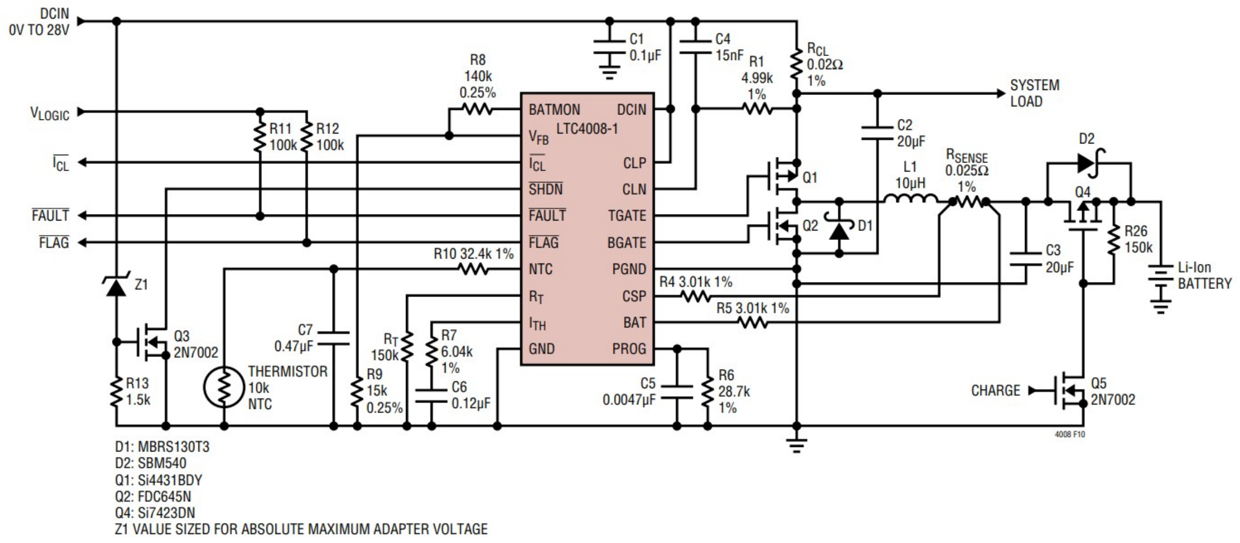


Figure 24: Typical LTC4008-1 Application (12.3V/4A)

Due to the similarities of the application, many of the component values could remain unchanged from the example. These included C1, C2, C3, C5, C6, C7, Q1, Q2, Q3, Q4, Q5, D1, D2, R4, R5, R7, R11, R12, R26, and R_T. Some components could also be eliminated because of some design simplifications. For instance, the thermistor function was not needed in the design. For disabling the thermistor function the LTC4008 datasheet provided this instruction: "If the thermistor is not needed, connecting a resistor between DCIN and NTC will disable it. The resistor should be sized to provide at least 10µA with the minimum voltage applied to DCIN and 10V at NTC. Do not exceed 30µA" . Based on this, a 750kΩ resistor was chosen as it would provide approximately 20µA with the expected DCIN voltage range of 25-28V.

The second simplification performed was the elimination of the adapter limiting. This feature is used to prevent the charger from drawing more current than circuitry feeding the charger is capable of handling. Since the charger is the only load to be powered and it would be designed to pull a max of 2A, the adapter current limiting was not needed. This allowed R1, R_{CL}, and C4 to be removed and CLN to be directly connected to CLP.

Next, the values of R_{SENSE}, R6, R8, R9, and L1 had to be determined. Based on suggested values from the datasheet, first two resistors, R_{SENSE} and R6, were chosen to be 0.050Ω and 28.7kΩ, respectively. R8 and R9, which set the final float voltage of the charger, were calculated next. R9 was chosen to be an 11.5kΩ resistor. This then forced R8 to be 232kΩ based on Equation #.

$$V_{FLOAT} = 1.19V \left(1 + \frac{R8}{R9}\right) \quad (20)$$

Lastly L1, the inductor for the charger, was calculated by Equation #.

$$\Delta I_L = 0.4(I_{MAX}) = \frac{1}{(f)(L)} V_{OUT}\left(1 - \frac{V_{OUT}}{V_{IN}}\right) \quad (21)$$

Rearranging the equation and plugging in V_{OUT} = 25V, V_{IN} = 28V, f = 300kHz, and I_{MAX} = 2A, the inductance (L) can be calculated.

$$L = \frac{1}{0.4(2\,A)(300\,kHz)}(25\,V)\left(1 - \frac{25\,V}{28\,V}\right) = 11\,\mu H \quad (22)$$

Following this, a 10μH inductor was selected for L1.

After selecting all the needed component values, Digi-Key and Mouser websites were used to find components meeting the needed values. A table in the appendix lists all selected components. Most of the required PCB footprints for these components were acquired from Ultra Librarian and Component Search Engine, two free online PCB symbol and footprint libraries. These component models were input in EAGLE, a software for designing the layouts of PCBs. The first step in creating the board within EAGLE is connecting all the components in an electrical schematic. This final schematic is illustrated in Figure 25.

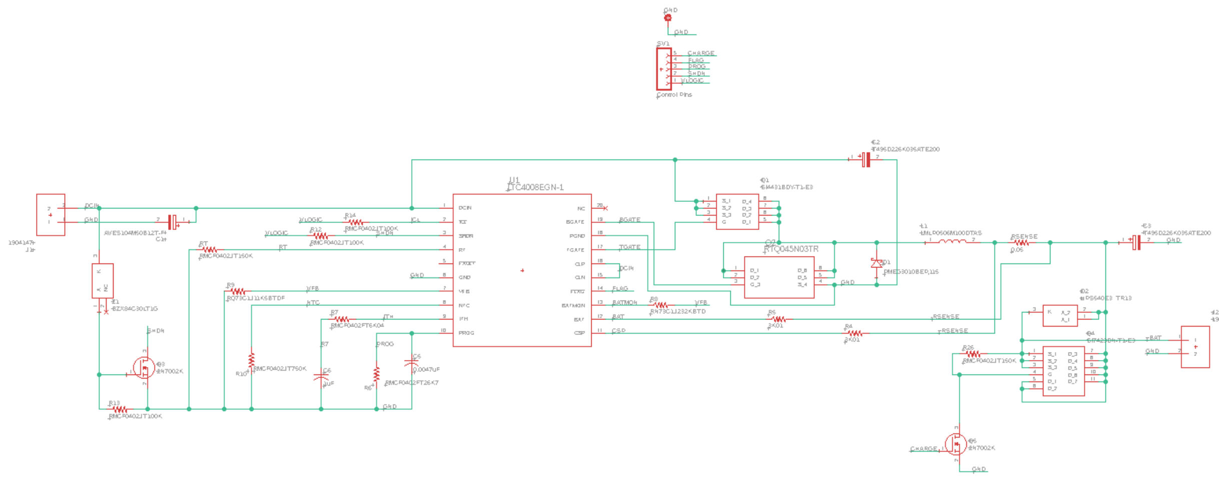


Figure 25: EAGLE Schematic for Charger

Following the PCB layout considerations given on page 21 of the LTC4008 datasheet, the traces for the PCB were designed. For each trace, the expected current was considered to size them appropriately. An online trace width calculator from Digi-Key was used to calculate the needed trace widths. The final board layout is shown in Figure 26.

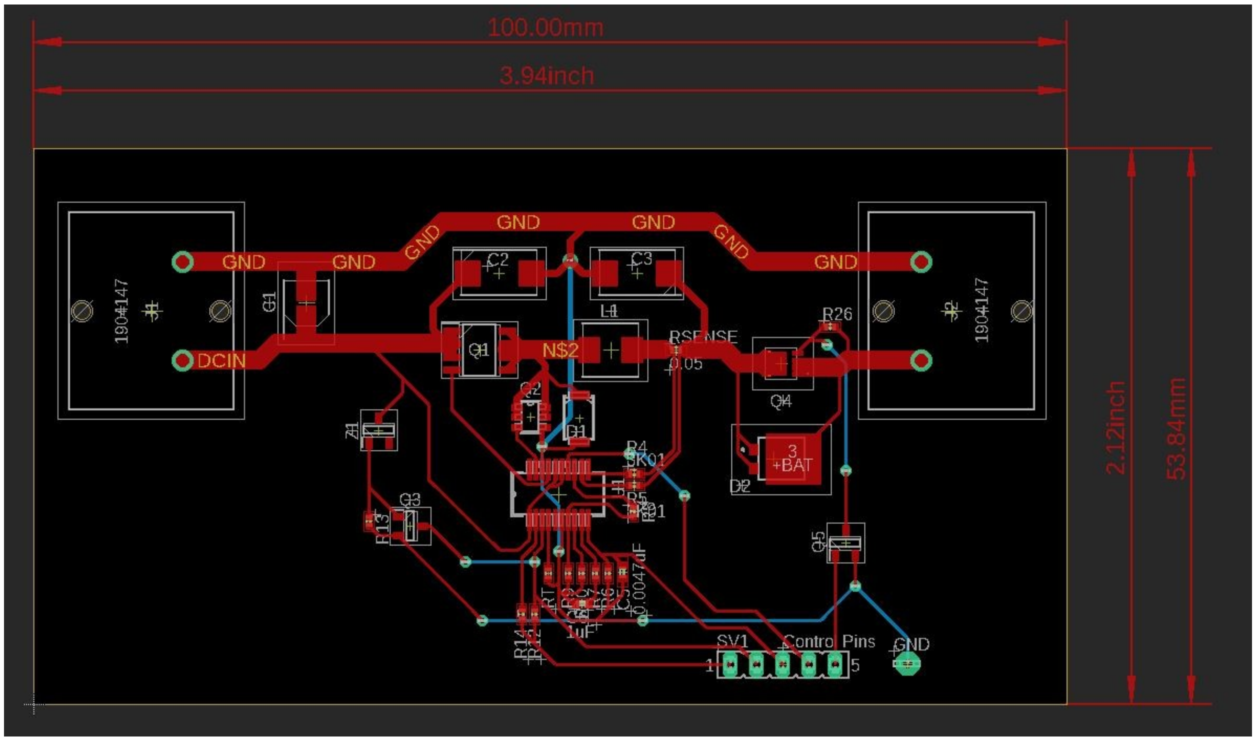


Figure 26: EAGLE Board Layout for Charger

Terminal blocks were added to either end of the board to provide easy input and output connections. A five-pin header was also included in the design to allow for easy interfacing with a microcontroller.

The board was then printed using the LPKF PCB milling machine at the University of Southern Indiana's Applied Engineering Center. After printing, the board was examined and tested to ensure that all connections and traces were correct. Each component was then manually placed on the board using solder paste with a pneumatic dispenser for the surface-mount devices and solder wire with a soldering iron for the terminal blocks and pin header. After all components were on the board, all component-to-copper connections were tested to ensure proper continuity. The final construction of the board is illustrated in Figure 27.

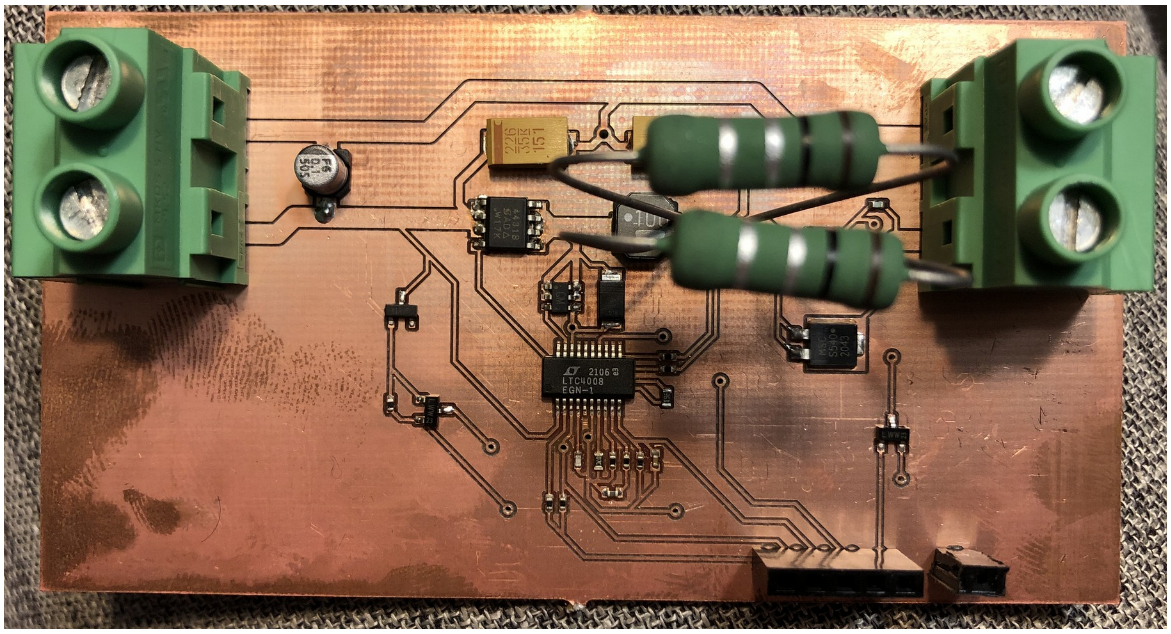


Figure 27: Final Construction of PCB Charger

The board then needed to be tested for proper operation. To accomplish this, a 25V power supply was used for the power source, and one of the 24V Lithium-ion batteries was connected to be charged. While the charger was designed to charge at 2A, the power supply could only provide a maximum of 1A at 25V. Thus, it was expected to max out the power supply at 1A. When tested though, the max charging current ever obtained was 85mA, as shown in Figure 28.

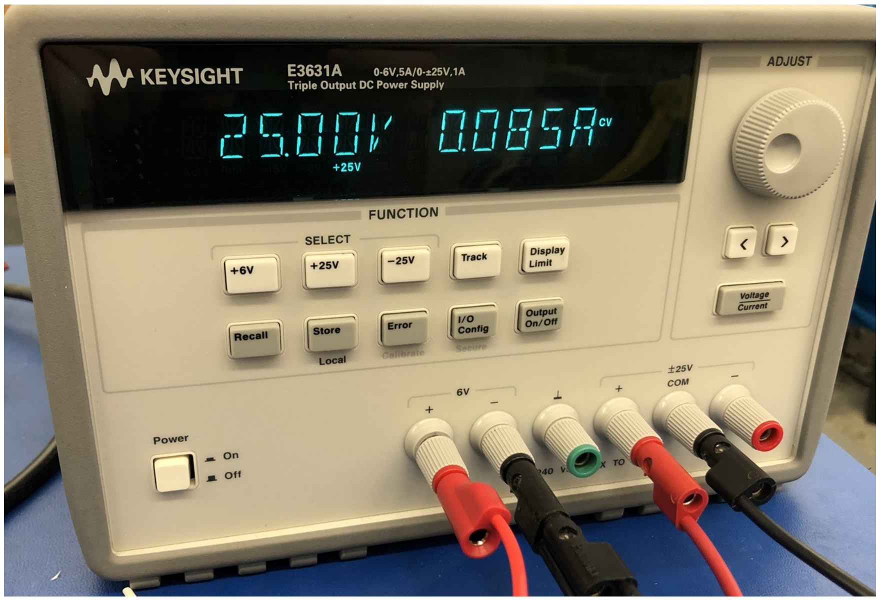


Figure 28: Power Supply Output during Charging

This current is far below designed specifications of the charger. Immediately, component connections were once again checked, but no issue was found. Next, bad components were considered, including resistors, capacitors, and even the main IC of the board. Still, no solution was found. In the future, further test should be performed to find the reason for the charger providing an insufficient charging current.

10 POWER REGULATION FOR CHARGER

10.1 Back-EMF Motor Testing

The first test done with the motor is testing the back emf voltage produced. This was done by connecting one of the 3 leads of the motor to an oscilloscope. Performing multiple test at different pedaling rates led us to conclude that the back emf voltage produced was sinusoidal full wave ac voltage shown in Figure 29 & Figure 30. This allowed us to proceed to the next step, being able to utilize this power to recharge by converting the voltage from ac to dc.

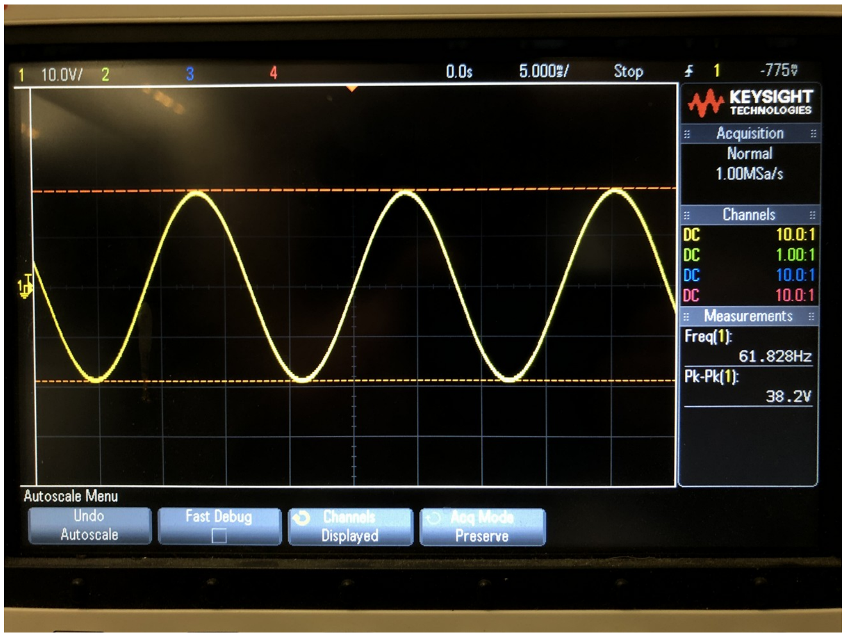


Figure 29: Hub Motor Back EMF Voltage at 60 RPM

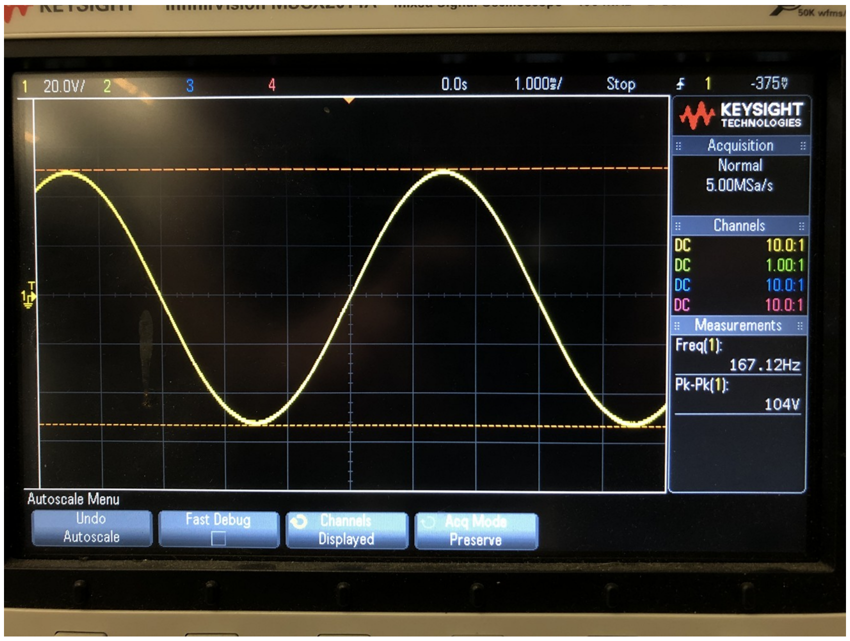


Figure 30: Hub Motor Back EMF Voltage at 180 RPM

10.2 Rectifier Selection

The first requirement was to figure out which rectifier type was needed. The majority of BLDC motors produce a trapezoidal back emf. To confirm this, tests were run on the hub motor with an oscilloscope. The test results showed that the back emf of this hub motor was sinusoidal instead of trapezoidal. This allowed for the selection of a 3-phase full wave bridge rectifier as it was more efficient by using the full wave rather than only half and was more readily available than a center tap rectifier. The next specification to consider was the max rectifiable voltage. Pedaling at max effort on the bike, which was around 180 RPM, the max peak to peak voltage produced was around 100V shown in Figure 31.

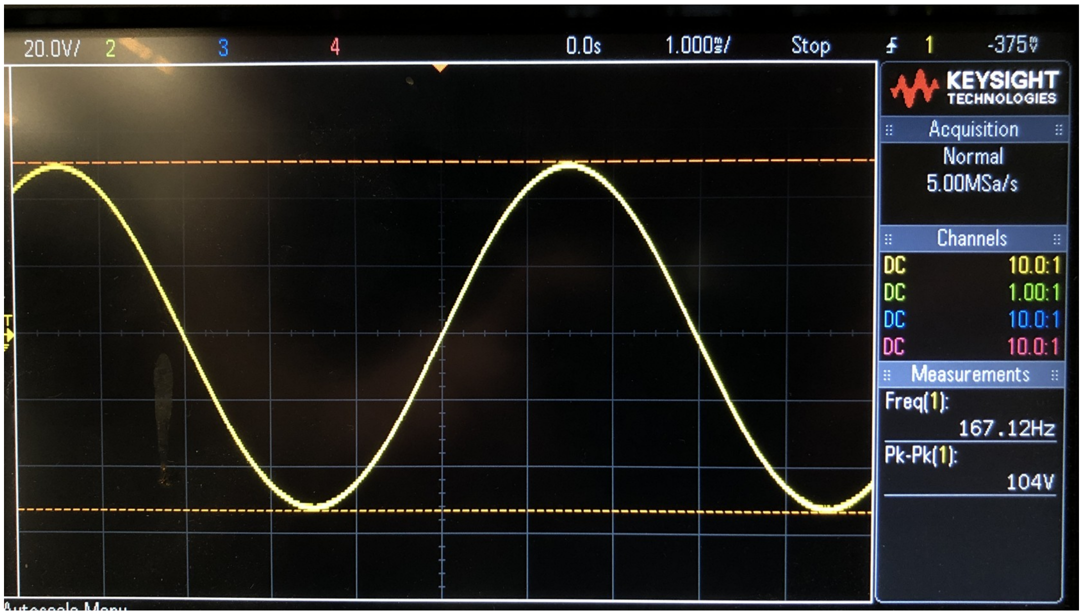


Figure 31: Motor Back-EMF when pedaling at max effort

The last consideration was the max current output. This was found from the max power needed for the IC. The max power needed was 50W, estimating some voltage loss from the rectifier and ensuring enough current was rectified the max current needed was 5A.

The selected rectifier was Taiss Bridge Rectifier shown in Figure 32. This rectifier is a 3-phase full wave bridge rectifier with a max average forward rectified current of 100A and repetitive peak reverse voltage of 1600V. This rectifier specification was over the required specification requirements, but because of the low cost and fast delivery date it was chosen for this project.

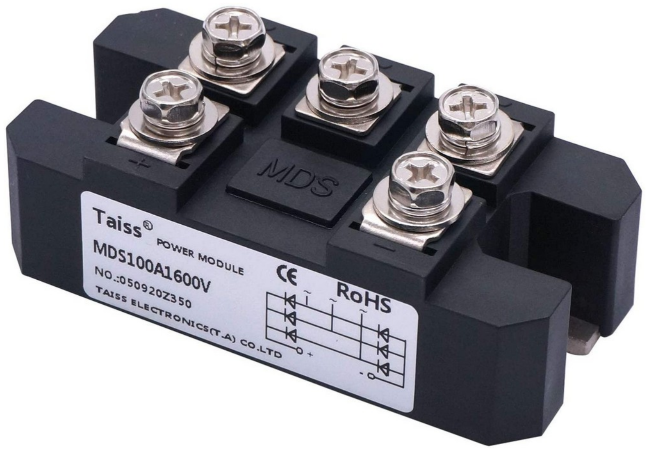


Figure 32: MDS-100A Taiss Bridge

10.3 Load Testing

To ensure that the motor could sufficiently power the required load when being used as a generator, a couple load tests were performed. The first test was performed at a low speed – one that would be comfortable for the rider to endure for extended periods of time. This speed was found to be approximately 60 RPM or pedaling approximately once per second. The second test was performed at the rider's max effort. This speed was found to be approximately 180 RPM or pedaling approximately three times per second.

The load placed on the generator needed closely simulate the same load that would be required when charging one of the batteries. Charging the battery would require a voltage output around 25V with a charging current of 2A. Performing Ohm's Law with this information, an effective resistance of around 12.5Ω is found. This load would also need the capability of dissipating nearly 100W. A low-valued resistor with a high-power rating was searched for, but one was not found. Another solution was to order the needed resistor. However, both the price and the waiting time that would be required were undesirable, so an alternative solution was searched for. Eventually, the idea of possibly using an electric skillet as a high-power resistor was considered. To test the feasibility of this solution the resistance of an electric skillet needed to be measured. One of the team’s members had an electric skillet at home which was measured to have a resistance of 14.2Ω. This was very close to the needed 12.5Ω. Furthermore, the skillet had a power rating of 1000W, so it would be able to easily handle the required power.

To perform the testing, the e-bike was mounted on the rear wheel stand, and the rectifier was connected to the three phases of the motor. The electric skillet was then connected to the DC output of the rectifier as shown in Figure 33, with two digital multimeters monitoring the voltage and current of the load.

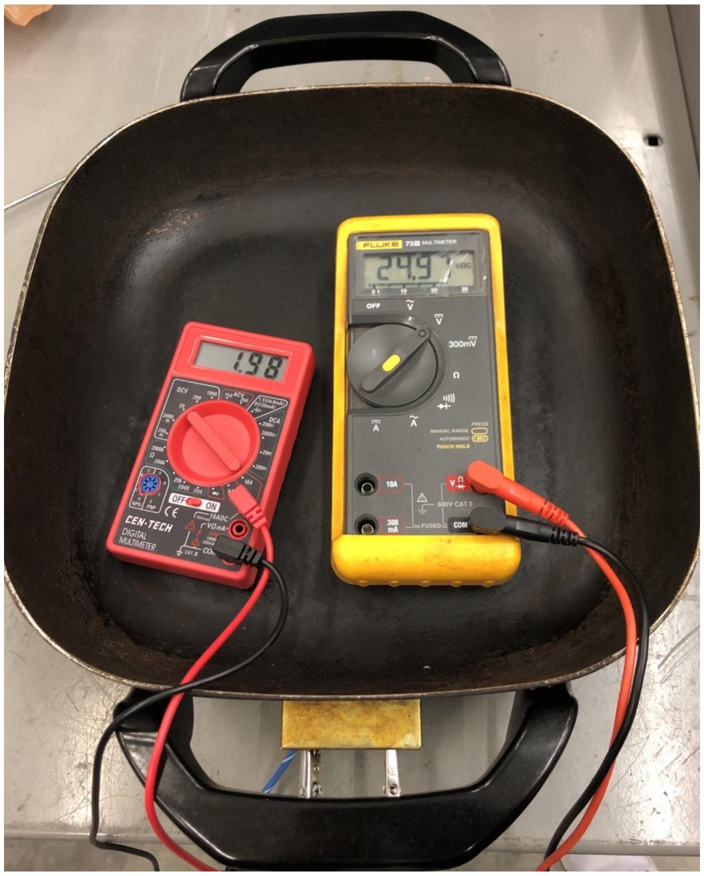


Figure 33: Electric Skillet Load Test Setup

During the low-speed test, had a voltage of 18V and a current of 1.6A. Meanwhile, the high-speed test showed a voltage of 52V and a current of 4.5A. A no-load voltage, or open-circuit voltage, of the generator was also measured at both test speeds. At low speed, the generator had a no-load voltage of 22V, and at high speed, it had a no-load voltage of 62V. The voltage regulation of the generator was then calculated from Equation #.

$$Voltage\ Regulation = \frac{V_{no\ load} - V_{full\ load}}{V_{full\ load}} \times 100\% \quad (23)$$

At low and high speed, the resulting voltage regulations were 22.2% and 19.2%, respectively. This showed that while not having a very good regulation, it was fairly consistent over a wide range of loads. Lastly and most importantly, the test also showed that the generator could easily provide sufficient power to charge one of the Lithium-ion batteries.

10.4 VOLTAGE REGULATOR SELECTION

From the load test, at a comfortable pedaling rate of 60 RPM, 18V and 1.6A was produced and at max an effort pedaling rate of 180 RPM, 52V was produced with 4.5A. A more comfortable pedaling rate was desired over higher speed. The IC recharging circuit had a max input of 28V, and the battery needs at least 25.2V to charge fully. A boost converter was chosen based on the require voltage of the IC charger. To charge the battery at the max rate, 2A was required therefore the boost converter must be able to provide a constant current of 2A. The max power produced happened at 180 RPM at 234W so the boost converter should be able to handle at least 234W.

The boost converter selected had the following specifications:

- Input Voltage 8.5V – 50V DC
- Output Voltage 10V – 60V DC
- Max. Input Current: 15A
- Constant Current Output: 0.2A to 12A

- Max Power 400W

This boost converter shown in Figure 34, met all the specification requirements needed to provide the IC recharger with the correct amount of voltage and current.

Figure 34: 400W DC Constant Current Boost Converter

11 CHARGING CONTROL

11.1 MICROCONTROLLER

A microcontroller was needed to interface with the charger on the PCB to give control and feedback to the user. This control consisted of having control over when the charger began charging and receiving feedback on the charging current being supplied to the battery, as well as keeping track of the total time of continuous charging. Many different microcontrollers could have been used for this implementation. However, for this project, the nRF51 Development Kit board, shown in Figure 30, was used.

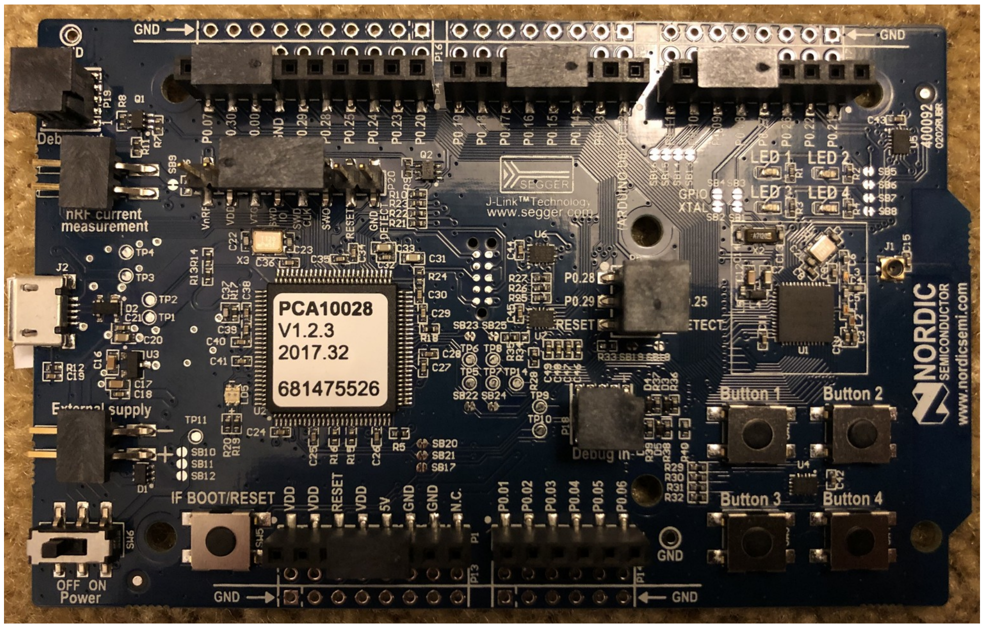


Figure 35: nRF51 Development Kit board

This microcontroller was selected for a couple reasons.

1) The nRF51 was already on hand and did not need to be purchased.
2) Both team members were familiar with the board and its associated integrated development environment CrossStudio, as it was being used in another course.

11.2 Code Structure

The code for the nRF51 was written in C language and was based primarily on interrupts. By using interrupts, the microcontroller is able to operate more efficiently, and the code can be more simplified. Before actually writing the code, a flowchart was developed to define how microcontroller needed to operate. This flowchart is shown in Figure 36.

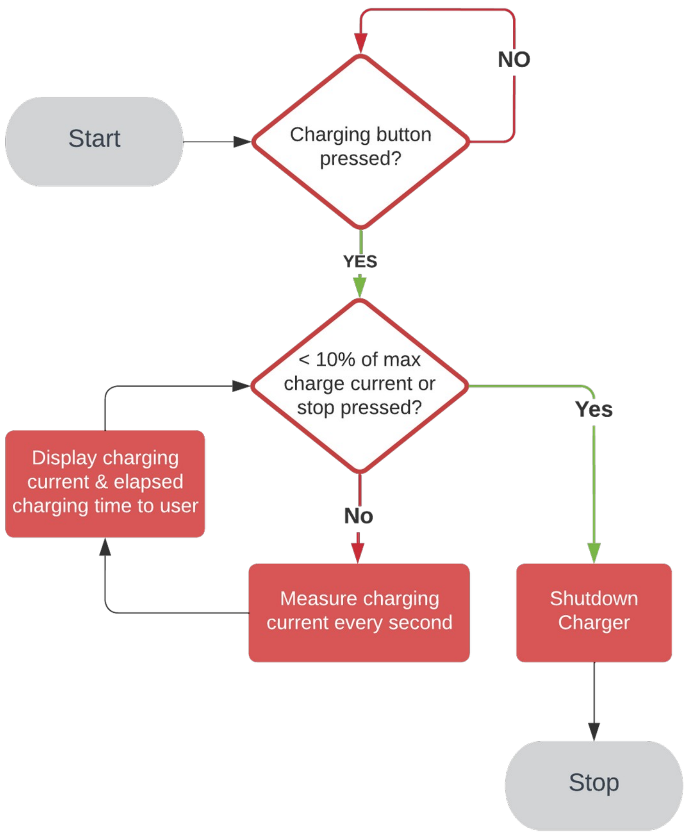

NO
Start
Charging button pressed?
YES
< 10% of max charge current or stop pressed?
Yes
Display charging current & elapsed charging time to user
No
Measure charging current every second
Shutdown Charger
Stop

Figure 36: Charging Controller Flowchart

The flowchart shows that initially the controller is just waiting for the user to begin charging by pressing the charge button. Once the user decides to start charging, the controller goes into a charging state where the controller takes readings each second of PROG pin on the LTC4008. This measurement is performed by an analog to digital converter (ADC) on the controller. According to the datasheet for the LTC4008, the voltage value of the PROG pin is always between 0.309V and 1.19V, with 0.309V corresponding to no charging current and 1.19V corresponding to max charging current. Knowing this, the reading of the PROG pin could be converted into the actual value of the charging current in amps.

To consistently take current measurements every second, the Timer peripheral on the nRF51 was utilized. By setting up the Timer's interrupt to trigger every second, the timing of the measurements could be easily regulated. Furthermore, by having an additional variable just increment each time the interrupt was triggered, the total elapsed time of charging could be tracked. Both the elapsed time and the current measurement would then be displayed to the user using the Universal Asynchronous Receiver/Transmitter (UART) of the nRF51. The UART transfers data via serial communication over a connected USB cable. To receive the serial communication, a laptop application called PuTTY was used. If time had allowed, a better method of displaying the information to the user could have been included. This could involve using a small LCD display that accept serial communication, allowing the design to be more mobile and user friendly.

Finally, the controller breaks out of its charging state if the current falls to 10% of the max current or if the user decides to stop the charging. At this point, the controller would shut down the charger and once again wait for the user to begin charging. The full code used in implementing the charging controller can be found in the Appendix.

12 TOTAL SYSTEM EVALUATION

As discussed previously, the custom charger PCB did not provide a sufficient charging current when undergoing testing. Thus, it could not be included in the evaluation of the total system. In an attempt to overcome this issue, the possibility of solely using the boost converter to charge one of the Lithium-ion batteries was considered. Since the selected boost converter came with both an adjustable output voltage and current limiter, the boost converter could be set to 25.2V with a max current of 2A. To effectively adjust these settings, a load needed to be attached to the boost converter, so the electric skillet was again elected for the purpose.

After setting the voltage and current of the boost converter, a battery was connected to attempt charging. This connection required the positive terminals of boost converter and battery to be connected. Likewise, the negative terminals of each were also connected. With two multimeters connected to monitor the voltage and current, charging was ready to begin. When the rider began pedaling, the battery began to successfully charge. The two multimeters set up to monitor the charging are shown in Figure 37, with the current measurement displayed on the left multimeter and the voltage measurement on the right.

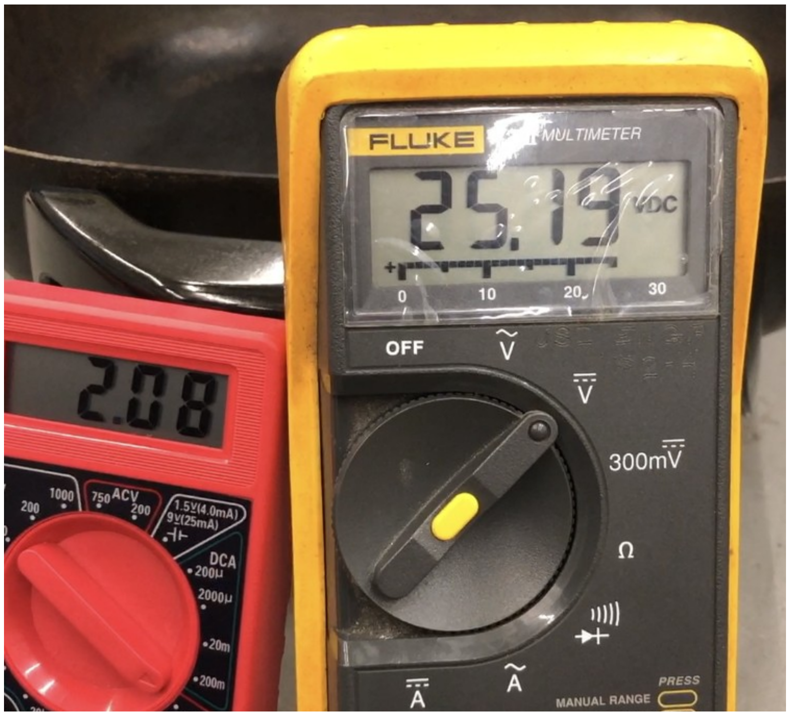


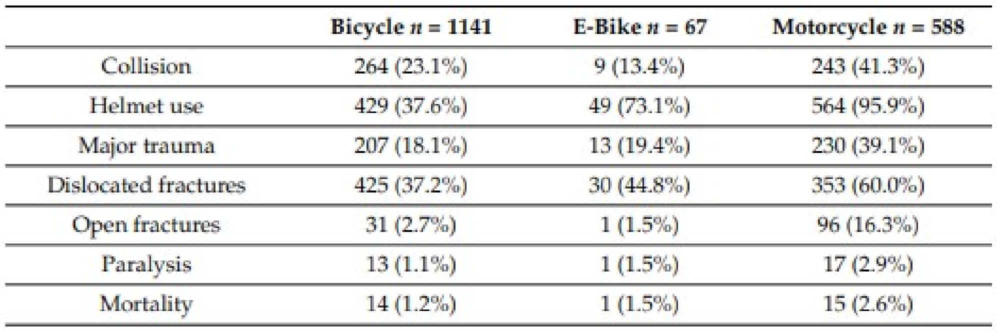
Figure 37: Monitoring Charging with Multimeters

The multimeters showed the battery being charged at a current of 2.08A and a voltage of 25.19V. Both of these values were almost exactly the same as the desired parameters for charging the battery. The system was therefore successful in charging the battery even without the charger PCB. The downside to only using the boost converter for charging is a higher inaccuracy in the charging setpoints and less control over the charging operation. Hence, it would still be desirable to implement the charger PCB into the system in the future.

13 PUBLIC HEALTH, SAFETY, AND WELFARE

Safety is a major concern with any product. When safety is a concern while designing a product, it reduces liability and possible recalls. During the design process, human error, maintenance, and foreseeable misuse were considered. With this system the most obvious safety concern is the speed and operability of the bike itself. A comparison of injury patterns between e-bike, bicycle, and motorcycle accidents was researched and analyzed. The conclusion of this study revealed that the overall injury pattern was most similar to that of the cyclist. Additionally, e-bikers who sustained injuries were older than bicycle or motorcycle riders and head protection is recommended when operating the e-bike at any age .

Table 6: Injury type analysis for bicycles, e-bikes, and motorcycles



	Bicycle *n* = 1141	E-Bike *n* = 67	Motorcycle *n* = 588
Collision	264 (23.1%)	9 (13.4%)	243 (41.3%)
Helmet use	429 (37.6%)	49 (73.1%)	564 (95.9%)
Major trauma	207 (18.1%)	13 (19.4%)	230 (39.1%)
Dislocated fractures	425 (37.2%)	30 (44.8%)	353 (60.0%)
Open fractures	31 (2.7%)	1 (1.5%)	96 (16.3%)
Paralysis	13 (1.1%)	1 (1.5%)	17 (2.9%)
Mortality	14 (1.2%)	1 (1.5%)	15 (2.6%)

Table 6 shows the percentage of injury types for different transportation modes. As shown in the table, the e-bike has a very low collision percentage showing that users are more aware of their surroundings when operating the e-bike. For major trauma, dislocated and open fractures, paralysis, and mortality injury type, the e-bike resembled the bicycle's percentage more than that of the motorcycle.

14 ENVIRONMENTAL AND GLOBAL

In Third World countries, bicycles are a very common and affordable mode of transportation for the middle and lower classes. However, these motor scooters are a large source of pollution in these countries. Thus, it is desirable to provide a cleaner alternative to motor scooters. In May 2020, eBikeGo, an Indian company, launched a subscription based eBikeGo Environ electric cargo bicycle. It took 3 hours to charge completely and can run up to 70 km, with a loading capacity of 200 kg . E-bikes with this type of range can significantly reduce the pollution caused by transporting goods.

The main environmental consideration with these e-bikes lies in the battery production. With an increase in demand for battery packs the mining for lithium, graphite, cobalt, and manganese, the raw materials required for battery production, increases. Also depleted batteries that are not recyclable end up being left in landfills where they leak toxic chemicals into the air, and with an increase in battery demand this damage is further increased.

The global market for e-bike battery pack is moderately consolidated with major players holding more than 70% of the total market share. These companies include Bosch, Panasonic, and Samsung SDI . These companies already cause a great deal of pollution in their production of manufactured goods.

15 ECONOMICS

One of the goals of the project is to make a more affordable electric bike. Therefore, the cost of reproducing the end product should be less than the common price range for commercial e-bikes. A quick Google search shows that similar e-bikes are priced at $500 or more for a basic conversion system and \$1500 or more with power regeneration. Thus, the goal reproduction cost of the project should be under \$500.

16 SOCIAL AND CULTURAL

Bicycles are already widely accepted worldwide. For instance, a majority of the world, over 50%, already knows how to ride a bike . The skill of riding a bike is the largest and adding a throttle with a motor to the bike would be an easy adjustment for most, especially those who have experience driving a motor scooter, one of the main targeted audiences.

17 TEAMWORK

Since the team only consists of two members, communication has been fairly straightforward. The primary means of communication has been text messaging as it is quick and efficient. To assign tasks to each member, tasks have typically been categorized by whether they deal with the electrical or mechanical parts of the system. Most of the tasks associated with the electrical side of the system are assigned to the member majoring in Electrical Engineering, Evan Fleming, while the tasks dealing with mechanical portions of the system are assigned to the Mechatronics major, Luis Lopez. Organization of the team's research materials has also been well organized. A Zotero account was created to store all research materials and to automatically create citations for any documents. For hands on task both members were present for installation and testing of the system.

18 CONCLUSIONS AND RECOMMENDATIONS

18.1 CONCLUSION

In conclusion the system successfully recharged the battery with the motor acting as a generator using only the rectifier and boost converter. The custom Lithium-ion battery charger was designed, fabricated, and populated. The IC charger was able to recharge the battery, but at a low current. Furthermore, as shown in Table 7, the total cost of the system was $523.81, which was just over the projects goal of $500.

Table 7: Summary of Costs

Item Description	Price	Quantity
Rectifier	$16.99	1
Boost Converter	$14.16	1
Battery	$134.00	2
Conversion Kit	$259.00	1
Charger Components	$27.66	1 Board
Mount metal	$72.00	2

However, if considering just adding the components needed to enable recharging onto a preexisting e-bike, it would cost just under $60.

18.2 Future Recommendations

With the time constraints involved with this project there were additional modification that were left out. Some future considerations to increase the usability and efficiency of this system would include, creating a single board recharging circuitry consisting of the three main components used to regulate the voltage, the rectifier, boost converter, and charger IC. This would reduce the size and space used by the system.

Additionally, instead of a boost converter, which limit us to the voltage that it is set to and any voltage higher is unusable; a buck-boost converter would be designed where it would allow the system to use a wider range of usable voltage by boosting the voltage or reducing it to the voltage required by the charger IC.

The charger IC charge current and float voltage is not adjustable in this iteration. An improvement would be using potentiometers in place of a constant valued resistors to allow the charge current and float voltage to be adjustable which will allow the recharging system to recharge other batteries with different parameters compared to the 24V Kobalt batteries.

The main improvement that would make the system more user friendly is adding a switching mode directly to the motor controller. Adding a switching mode will cut off power from the controller and only allow the power induced by the motor to run through the recharging system without interference from the motor controller.

19 REFERENCES

Appendix A: Total Project Costs

Table 8: Bill of Materials

Part Description	**Qty**	**Price**	**Subtotal**
Electrical			
Ebike Conversion Kit	1	$259.00	$259.00
3-phase rectifier	1	$16.99	$16.99
Boost converter	1	$14.16	$14.16
Kobalt 24-Volt Max Lithium-ion Battery (2-pack)	1	$134.00	$134.00
3/16 in. Heat Shrink Tubing, Black (5-Pack)	1	$2.18	$2.18
12-10 AWG 8-10 Stud Ring Spade Terminal, Yellow (5-Pack)	1	$5.78	$5.78
12-10 AWG Snap Connector 5 Female 5 Male, Yellow (10-Pack)	1	$2.98	$2.98
#12 AWG Red Stranded CU THHN Wire (1ft)	2	$0.50	$1.00
Cambridge Resources 5-Count Butt Splice Wire Connectors	1	$5.29	$5.29
Charging Circuit Board			
Charger IC - LTC4008EGN-1	1	$10.02	$10.02
MOSFET - SI4431BDY-T1-E3	1	$1.16	$1.16
MOSFET - RTQ045N03TR	1	$0.76	$0.76
MOSFET - 2N7002K	2	$0.37	$0.74
MOSFET - SI7423DN-T1-E3	1	$1.70	$1.70
0.1uF capacitor -AVES104M50B12T-F	1	$0.40	$0.40
20uF tantulum cap - T495D226K035ATE200	2	$2.19	$4.38
0.0047uF cap - C0402C472K9RACTU	1	$0.38	$0.38
1uF capacitor - C0402C105K9PACTU	1	$0.10	$0.10
Schottky diode - PMEG3010EP,115	1	$0.42	$0.42
Schottky diode - UPS540E3/TR13	1	$0.50	$0.50
Zener diode - BZX84C30LT1G	1	$0.16	$0.16
10uH inductor - LMLP0506M100DTAS	1	$0.53	$0.53
0.05Ω sense resistor - ERJ-2BWFR050X	1	$0.42	$0.42

232kΩ resistor - RN73C1J232KBTD	1	$0.78	$0.78
11.5kΩ resistor - RQ73C1J11K5BTDF	1	$1.05	$1.05
26.7kΩ resistor - RMCF0402FT26K7	1	$0.10	$0.10
3.01kΩ resistor - RMCF0402FT3K01	2	$0.10	$0.20
6.04kΩ resistor - RMCF0402FT6K04	1	$0.10	$0.10
100kΩ resistor - RMCF0402JT100K	2	$0.10	$0.20
150kΩ resistor - RMCF0402JT150K	2	$0.10	$0.20
750kΩ resistor - RMCF0402JT750K	1	$0.10	$0.10
Terminal Block - 1904147	1	$1.88	$1.88
5-Pin Header - PPPC051LFBN-RC	1	$0.49	$0.49
Other Parts			
Hillman 1-in x 6-ft Plain Hot Rolled Steel Weldable Square Tube	2	$31.90	$63.80
#6-32 x 3/4 in. Combo Round Head Stainless Steel Machine Screw (6-Pack)	1	$1.28	$1.28
Hillman 4-in x 5-in Copper Sheet Metal	1	$9.36	$9.36
Project Total			
Total for all items purchased for project	N/A	N/A	**$542.59**

Appendix B: Charging Controller Code

```
/*
Authors: Evan Fleming and Luis Lopez
Assignment: Final Project
*/

#include <__cross_studio_io.h>
#include "nrf.h"
#include "nrf51.h"

// definition
#define high 1
#define low 0

// ADC Definitions
#define ADC_Res_8bit 0
#define ADC_Res_9bit 1
#define ADC_Res_10bit 2
#define ADC_Prescale_None 0
#define ADC_Prescale_Two_Thirds 1
#define ADC_Prescale_One_Third 2
#define ADC_Prescale_VDD_Two_Thirds 5
#define ADC_Prescale_VDD_One_Third 6
#define ADC_Ref_VBG 0
#define ADC_Ref_External 1
#define ADC_Ref_One_Half_VDD 2
#define ADC_Ref_One_Third_VDD 3
#define ADC_Input_select_0 1
#define ADC_Input_select_1 2
#define ADC_Input_select_2 4
#define ADC_Input_select_3 8
#define ADC_Input_select_4 16
#define ADC_Input_select_5 32
#define ADC_Input_select_6 64
#define ADC_Input_select_7 128

//GPIO DIRECTION
#define GPIO_DirIn 0
#define GPIO_DirOut 1
//GPIO INPUT BUFFER
#define GPIO_InputBufConn 0
#define GPIO_InputBufDisconn 1
//GPIO PULL
#define GPIO_PullNo 0
#define GPIO_PullLow 1
```

```
#define GPIO_PullHigh 3
//GPIO DRIVE
//S=Standard;H=High;D=disconnect
//0=low;1=high
#define GPIO_DriveS0S1 0
#define GPIO_DriveH0S1 1
#define GPIO_DriveS0H1 2
#define GPIO_DriveH0H1 3
#define GPIO_DriveD0S1 4
#define GPIO_DriveD0H1 5
#define GPIO_DriveS0SD 6
#define GPIO_DriveH0SD 7
//GPIO Sense
#define GPIO_SenseDisabled 0
#define GPIO_SenseHigh 2
#define GPIO_SenseLow 3

//DK Board Specific Pins
//Button GPIO PINS
#define BUT1 17
#define BUT2 18
#define BUT3 19
#define BUT4 20
//LED GPIO PINS
#define LED1 21
#define LED2 22
#define LED3 23
#define LED4 24


// decaring variable for pin
float Total_time=0;

// declaring interrupt
void GPIOTE_IRQHandler(void);
void TIMER0_IRQHandler(void);

// declaring functions
uint32_t ADC_GPIO_Config(uint32_t Res, uint32_t Prescale, uint32_t Ref, uint32_t
Input_select);
uint32_t GPIO_Config(uint32_t PIN,uint32_t PIN_Direction,uint32_t InputBuffer,uint32_t
Pull,uint32_t Drive,uint32_t Sense);
uint32_t GPIO_Dir(uint32_t PIN,uint32_t PIN_Direction);
uint32_t GPIO_toggle_pin(uint32_t PIN);
uint32_t GPIO_Write(uint32_t PIN,uint32_t Data);
void reverse(char* str, int len);
```

```
int intToStr(int x, char str[], int d);
void ftoa(float n, char* res, int afterpoint, int beforepoint);
uint32_t ADC_Current_Measure(float *Current_Percentage);
uint32_t UART_transmit(char data);

void main(void)
{
  //configure Timer0 for when to take current measurements
  NRF_TIMER0->MODE&=~(0x1<<0);
  NRF_TIMER0->MODE|=(0x0<<0); // set to timer mode
  NRF_TIMER0->BITMODE|=(0x2<<0); //24 bit timer bit width
  NRF_TIMER0->PRESCALER|=(0x9<<0); // prescale set to 9
  NRF_TIMER0->CC[0]|=31250; // counts for 1 second

  // configuring inputs/adc
   ADC_GPIO_Config(ADC_Res_8bit,ADC_Prescale_None,ADC_Ref_VBG,ADC_Input_select
_2); // PROG (pin 1)

  // Configuring outputs
  GPIO_Config(5,GPIO_DirOut,GPIO_InputBufDisconn,GPIO_PullHigh,GPIO_DriveS0S1,GPI
O_SenseDisabled); // charge (pin 5)
   GPIO_Config(6,GPIO_DirOut,GPIO_InputBufDisconn,GPIO_PullNo,GPIO_DriveS0S1,GPIO
_SenseDisabled); // stdwn (pin 6)

  // led configurations
    GPIO_Config(LED1,GPIO_DirOut,GPIO_InputBufDisconn,GPIO_PullHigh,GPIO_DriveS0S
1,GPIO_SenseDisabled); // led 1
    GPIO_Config(LED2,GPIO_DirOut,GPIO_InputBufDisconn,GPIO_PullHigh,GPIO_DriveS0S
1,GPIO_SenseDisabled); // led 2
    GPIO_Config(LED3,GPIO_DirOut,GPIO_InputBufDisconn,GPIO_PullHigh,GPIO_DriveS0S
1,GPIO_SenseDisabled); // led 3
    GPIO_Config(LED4,GPIO_DirOut,GPIO_InputBufDisconn,GPIO_PullHigh,GPIO_DriveS0S
1,GPIO_SenseDisabled); // led 4

  // buttons configuration
   GPIO_Config(BUT1,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullHigh,GPIO_DriveS0S1,GP
IO_SenseDisabled); // but 1
   GPIO_Config(BUT2,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullHigh,GPIO_DriveS0S1,GP
IO_SenseDisabled); // but 1
   GPIO_Config(BUT3,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullHigh,GPIO_DriveS0S1,GP
IO_SenseDisabled); // but 1
   GPIO_Config(BUT4,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullHigh,GPIO_DriveS0S1,GP
IO_SenseDisabled); // but 1

  /*Setting up UART*/
  //PSEL registers
  //program TXD,RXD,CTS,RTS before enabling
  // PIN TXD 9
  GPIO_Dir(9,1);

  // all others can be disabled 0xFFFFFFFF
  NRF_UART0->PSELCTS=0xFFFFFFFF;
  NRF_UART0->PSELRTS=0xFFFFFFFF;
  NRF_UART0->PSELRXD=0xFFFFFFFF;
  NRF_UART0->PSELTXD=9;

  // Speed of communication
  NRF_UART0->BAUDRATE=0x00275000;
  // Config register
  NRF_UART0->CONFIG&=~(0xF<<0); // disable all hardware control
  //enable UART
  NRF_UART0->ENABLE&=~(0x3<<0);
  NRF_UART0->ENABLE|=(0x4<<0);


  // setting up main interrupt for GPIOTE pin 17
  NRF_GPIOTE->INTENSET=(0x1<<0); // setting on intrpt for in[0]
   NRF_GPIOTE->CONFIG[0]&=~((0x1<<20)|(0x3<<16)|(0xF<<8)|(0x3<<0));  //  resetting  all
config
  NRF_GPIOTE->CONFIG[0]|=((0x2<<16)|(17<<8)|(0x1<<0)); // setting button as event

  // secondry interupt for flag
  NRF_GPIOTE->INTENSET=(0x1<<1); // setting on intrpt for in[1]
   NRF_GPIOTE->CONFIG[1]&=~((0x1<<20)|(0x3<<16)|(0xF<<8)|(0x3<<0));  //  resetting  all
config
  NRF_GPIOTE->CONFIG[1]|=((0x2<<16)|(20<<8)|(0x1<<0)); // setting button 4 as event

  // enabling interrupt for TIMER0
  NRF_TIMER0->INTENSET=(0x1<<16); // turn on interrupt for COMPARE[0] event

  NVIC_EnableIRQ(GPIOTE_IRQn);


  while(1){

  __WFI(); // wait for interrupt

  } // end while
} // end main
```

```
void GPIOTE_IRQHandler(void)
{
 // turn off interrupt
 NVIC_DisableIRQ(GPIOTE_IRQn);

 // interrupt for startup
 if (NRF_GPIOTE->EVENTS_IN[0]==1)
 {
  GPIO_toggle_pin(LED1);
  //clear and start Timer0
  NRF_TIMER0->TASKS_CLEAR=1;
  NRF_TIMER0->TASKS_START=1;
  NRF_TIMER0->EVENTS_COMPARE[0]=0;
  //enable interrupt for TIMER0
  NVIC_EnableIRQ(TIMER0_IRQn);
  // setting pins for charging
  GPIO_Write(5,high);
  GPIO_Write(6,high);
 }//end if


  // interrupt for flag
 else if (NRF_GPIOTE->EVENTS_IN[1]==1)
 {
  GPIO_toggle_pin(LED2);

  //disable interrupt for TIMER0
  NVIC_DisableIRQ(TIMER0_IRQn);
  //reset compare event and clear timer
  NRF_TIMER0->TASKS_STOP=1;
  NRF_TIMER0->TASKS_CLEAR=1;
  NRF_TIMER0->EVENTS_COMPARE[0]=0;
  // setting pins off (stop charging)
  GPIO_Write(5,low);
  GPIO_Write(6,low);
  // reset elapsed charging time
  Total_time=0;
 }//end else if

 // re-enable interrupt and reset events
 NRF_GPIOTE->EVENTS_IN[0]=0;
 NRF_GPIOTE->EVENTS_IN[1]=0;
 NVIC_EnableIRQ(GPIOTE_IRQn);

}// end of gpiote handler
```

```
void TIMER0_IRQHandler(void)
{
 // turn off interrupt
 NVIC_DisableIRQ(TIMER0_IRQn);

 //reset compare event and clear timer
 NRF_TIMER0->EVENTS_COMPARE[0]=0;
 NRF_TIMER0->TASKS_CLEAR=1;

 GPIO_toggle_pin(LED3);

 //local variables
 float Time_min, current_percent, current_amps;
 float max_current = 2;
 float minutes;
 float sec;
 char minutes_string[3];
 char sec_string[2];
 char current[5];
 char message[]={"Elapsed charging time = 000:00; Charging current = 0.000 A"};
 int i=0;

 //calculating elapsed time and current and converting to ASCII
 Total_time++; //add 1 second to total time
 Time_min=Total_time/(float)60; //calculate total time in minutes

 minutes=(int)Time_min; //grab only whole number value of minutes
 sec=(Time_min-minutes)*60; //grab fractional part of minutes and convert to seconds

 ftoa(minutes, minutes_string,0,3); //convert minutes to string
 ftoa(sec, sec_string, 0, 2); //convert seconds to string

 ADC_Current_Measure(&current_percent); //take measurement of PROG pin using ADC
 if(current_percent<0)
 {
  current_percent=0;
 }

 current_amps=current_percent*max_current; //calculate current in amps
 ftoa(current_amps, current, 3, 1); //convert current measurement to string variable

 for(i=0; i<3; i++) //update minutes value in message
 {
  message[24+i]=minutes_string[i];
 }
```

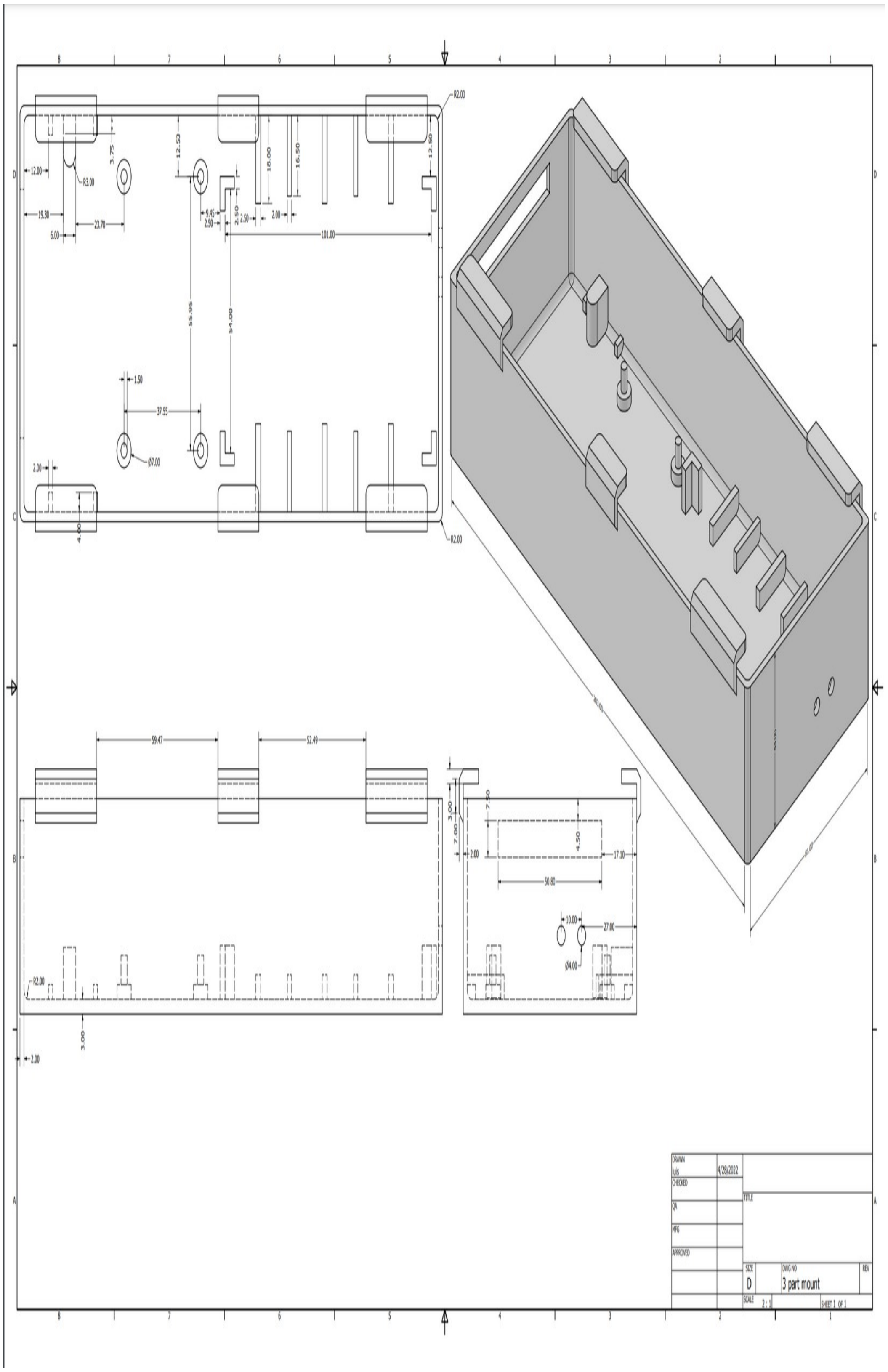
```
  for(i=0; i<2; i++) //update seconds value in message
  {
   message[28+i]=sec_string[i];
  }
  for(i=0; i<5; i++) //update current measurement in message
  {
   message[51+i]=current[i];
  }


  /* Send Data */
  //Start Task
  NRF_UART0->TASKS_STARTTX=1;

  for(i=0;i<57;i++) // transmit message
  {
   UART_transmit(message[i]);
  }

  //value line feed
  UART_transmit(10);

  //value carriage return
  UART_transmit(13);

  //re-enable timer interrupt
  NVIC_EnableIRQ(TIMER0_IRQn);
}// end TIMER0_IRQHandler


/*
name: ADC_GPIO_Config
description: Performs the configuration of an ADC pin and its respective GPIO pin and enables
the ADC
inputs: Res - resolution of ADC; Prescale - prescaling value for ADC; Ref - voltage reference for
ADC;
        Input_select - pin to use as ADC input
output: Successful(1)
*/
uint32_t ADC_GPIO_Config(uint32_t Res, uint32_t Prescale, uint32_t Ref, uint32_t
Input_select)
{

  // configure the GPIO pin as an input
  if(Input_select==ADC_Input_select_0)
  {
    GPIO_Config(26,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullNo,GPIO_DriveS0S1,GPIO_
SenseDisabled);
  }
  else if (Input_select==ADC_Input_select_1)
  {
    GPIO_Config(27,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullNo,GPIO_DriveS0S1,GPIO_
SenseDisabled);
  }
  else if (Input_select==ADC_Input_select_2)
  {
    GPIO_Config(1,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullNo,GPIO_DriveS0S1,GPIO_S
enseDisabled);
  }
  else if (Input_select==ADC_Input_select_3)
  {
    GPIO_Config(2,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullNo,GPIO_DriveS0S1,GPIO_S
enseDisabled);
  }
  else if (Input_select==ADC_Input_select_4)
  {
    GPIO_Config(3,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullNo,GPIO_DriveS0S1,GPIO_S
enseDisabled);
  }
  else if (Input_select==ADC_Input_select_5)
  {
    GPIO_Config(4,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullNo,GPIO_DriveS0S1,GPIO_S
enseDisabled);
  }
  else if (Input_select==ADC_Input_select_6)
  {
    GPIO_Config(5,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullNo,GPIO_DriveS0S1,GPIO_S
enseDisabled);
  }
  else if (Input_select==ADC_Input_select_7)
  {
    GPIO_Config(6,GPIO_DirIn,GPIO_InputBufConn,GPIO_PullNo,GPIO_DriveS0S1,GPIO_S
enseDisabled);
  }
  else
  {
   return 0; // return unsuccessful if GPIO did not configure properly
  }

  // configuring the ADC peripheral configuration
  NRF_ADC->CONFIG&=~((0x3<<16)|(0xFF<<8)|(0x3<<5)|(0x7<<2)|(0x3<<0));// resetting
  NRF_ADC->CONFIG|=((0x0<<16)|(Input_select<<8)|(Ref<<5)|(Prescale<<2)|(Res<<0));
  // enable the adc
  NRF_ADC->ENABLE|=(0x1<<0);

  return 1;
}// end ADC_GPIO_Config function


/*
name: GPIO_Config
description: Configure GPIO Pin
inputs: PIN - pin number ; PIN_Direction - configure as output or input pin ; InputBuffer - Connect or disconnect input buffer ;
        Pull - pull configuration (down, up, or no pull) ; Drive - drive configuration ;
        Sense - pin sensing mechanism (disabled, high, or low level)
output: Successful(1)
*/
uint32_t GPIO_Config(uint32_t PIN,uint32_t PIN_Direction,uint32_t InputBuffer,uint32_t Pull,uint32_t Drive,uint32_t Sense){

// reset the register  -  bits of interest to 0
NRF_GPIO->PIN_CNF[PIN]&=~((0x3<<16)|(0x7<<8)|(0x3<<2)|(0x1<<1)|(0x1<<0));
// program the register
NRF_GPIO->PIN_CNF[PIN]|=((Sense<<16)|(Drive<<8)|(Pull<<2)|(InputBuffer<<1)|
(PIN_Direction<<0));

//function ran properly
return 1;
}//end of GPIO_Config


/*
name: GPIO_Dir
description: Change the direction of a pin
inputs: PIN - pin number ; PIN_Direction - GPIO_DirIn(0) or GPIO_DirOut(1) for pin direction
output: Successful(1)
*/
uint32_t GPIO_Dir(uint32_t PIN,uint32_t PIN_Direction){
// reset the register  -  bit of interest to 0
NRF_GPIO->PIN_CNF[PIN]&=~(0x1<<0);
// program the register
NRF_GPIO->PIN_CNF[PIN]|=(PIN_Direction<<0);

//function ran properly
return 1;
}//end of GPIO_Dir
```

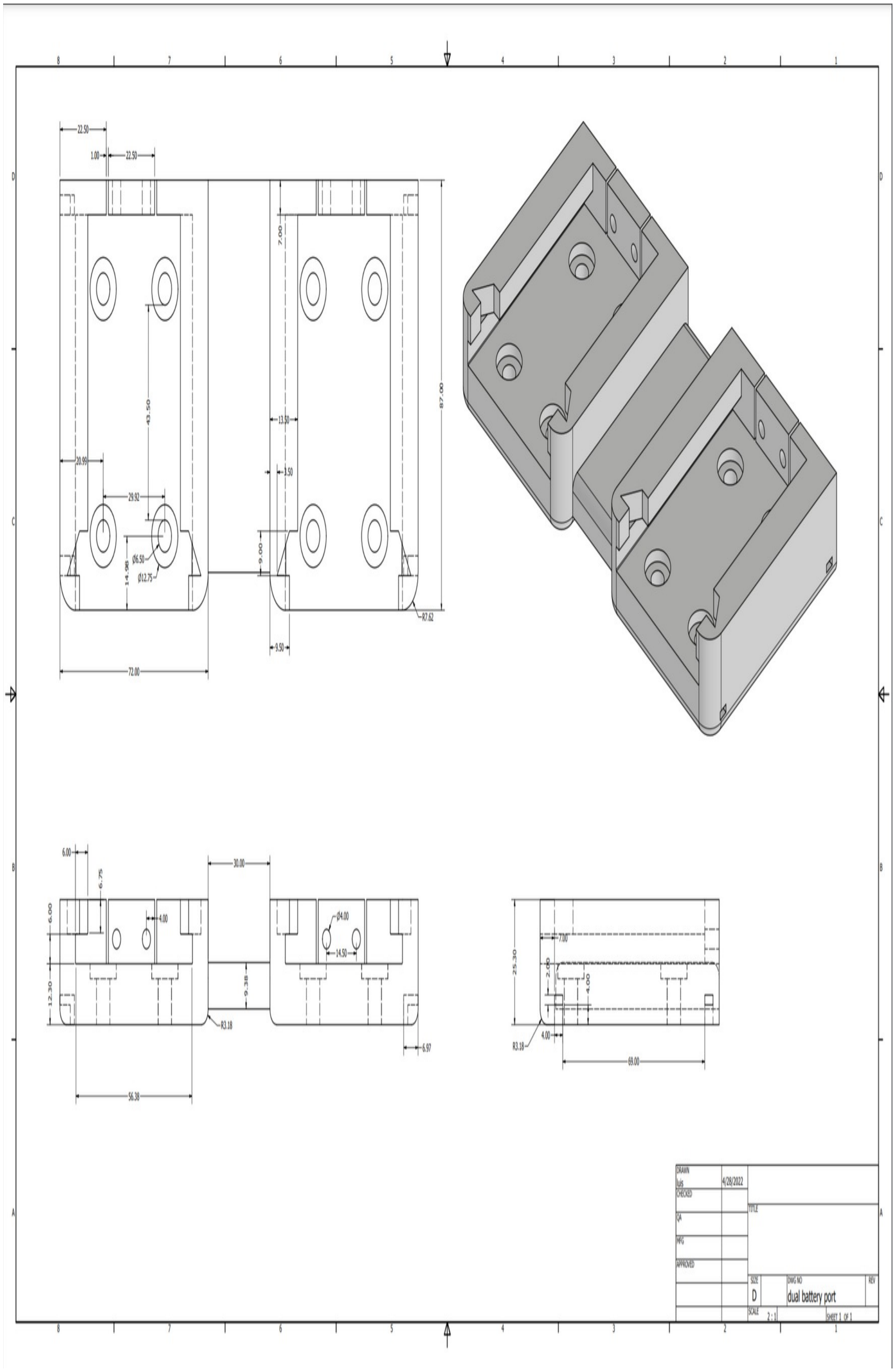
```
/*
name: GPIO_toggle_pin
description: This function toggles a selected pin
inputs: pin number
output: Successful(1)
*/
uint32_t GPIO_toggle_pin(uint32_t PIN){

      NRF_GPIO->OUT^=(0x1<<PIN);

  return 1; //function was successful
} // end of GPIO_toggle_pin function


/*
name: GPIO_Write
description: Write a 1(ON) or 0(OFF) to a pin in the OUT register
inputs: PIN - pin number ; Data - 1(ON) or 0(OFF) to write to the pin
output: Successful(1)
*/
uint32_t GPIO_Write(uint32_t PIN,uint32_t Data){
// program the register
if(Data)
{
  NRF_GPIO->OUTSET=(0x1<<PIN);
}
else
{
  NRF_GPIO->OUTCLR=(0x1<<PIN);
}

//function ran properly
return 1;
}//end of GPIO_Write


// Reverses a string 'str' of length 'len'
void reverse(char* str, int len)
{
    int i = 0, j = len - 1, temp;
    while (i < j) {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
        i++;
        j--;
    }
}



// Converts a given integer x to string str[].
// d is the number of digits required in the output.
// If d is more than the number of digits in x,
// then 0s are added at the beginning.
int intToStr(int x, char str[], int d)
{
    int i = 0;
    while (x) {
        str[i++] = (x % 10) + '0';
        x = x / 10;
    }

    // If number of digits required is more, then
    // add 0s at the beginning
    while (i < d)
        str[i++] = '0';

    reverse(str, i);
    str[i] = '\0';
    return i;
}



// Converts a floating-point/double number to a string.
void ftoa(float n, char* res, int afterpoint, int beforepoint)
{
    // Extract integer part
    int ipart = (int)n;

    // Extract floating part
    float fpart = n - (float)ipart;

    // convert integer part to string
    int i = intToStr(ipart, res, beforepoint);

    // check for display option after point
    if (afterpoint != 0) {
        res[i] = '.'; // add dot
        // Get the value of fraction part upto given no.
        // of points after dot. The third parameter
        // is needed to handle cases like 233.007
        fpart = fpart * pow(10, afterpoint);

        intToStr((int)fpart, res + i + 1, afterpoint);
    }
}


/*
name: ADC_Current_Measure
description: This function completes a reading of the ADC and converts the measurement into the corresponding charging current
inputs: *Current_Percentage - pointer to where the calculated current percentage should be stored
output: Successful(1)
*/
uint32_t ADC_Current_Measure(float *Current_Percentage)
{
  uint32_t ADC_data;

  ADC_Read(&ADC_data); //get reading of PROG pin from ADC
   *Current_Percentage=((ADC_data*1.2/(float)255)-0.309)/(1.19-0.309); //calculate percentage of max current from ADC data

  return 1; //function was successful

} // end of ADC_Current_Measure function


/*
name: UART_transmit
description: transmits one byte using the Universal Asynchronous Receiver/Transmitter
inputs: data - data to be transmitted
output: Successful(1)
*/
uint32_t UART_transmit(char data)
{
  NRF_UART0->TXD=data;//load data into TXD
  while(!NRF_UART0->EVENTS_TXDRDY)
  {//wait for txdready
  }
  NRF_UART0->EVENTS_TXDRDY=0; //reset the TXDRDY event
}// end Transmit function
```

Appendix C: Python Script for Power Calculations

```
import numpy as np

lin_speed=20 #linear speed of bike in mph
radius=13 #radius of bike wheel in inches
w=lin_speed*5280/3600/(radius/12) #angular speed
print('w =',w,'rad/sec')
nm=w*30/np.pi
print('nm =',nm, 'RPM')

theta=0.05993 #angle of slope in radians 0.09967 0.05993
weight_lb=175 #weight in lbf
speed_f=lin_speed #final speed in mph
t=15 #time to final speed in seconds
R_w=radius #radius of wheel in inches

a_imp=speed_f*5280/3600/t #acceleration in ft/s^2


#Constants
g=9.81 #gravity
Cr=0.004
Cd=0.7
p=1.225 #density of air at sea level (kg/m^3)
A=0.3716 #frontal area (m^2)
v_w=0 #wind speed (m/s)
v_g=speed_f*5280/3600*0.3048 #convert ground speed to m/s

#Unit conversions
weight=weight_lb*4.44822 #convert weight to newtons
a=a_imp*0.3048 #convert acceleration to m/s^2
a=0
M=weight/g #calculate mass
Rw=R_w*0.0254 #convert wheel radius to meters


Fr=M*g*Cr*np.cos(theta)
Mg=M*g
Fs=M*g*Cr*np.sin(theta)
Fg=M*g*np.sin(theta)
Fw=(Cd*p*A*(v_w+v_g)**2)/2
Ma=M*a
print(Fr)
print(Fs)
print(Fg)
print(Fw)

Fp=Fr+Fs+Fg+Fw+Ma #propulsion force
print('Propulsion force =', Fp, 'N')

Tw=Fp*Rw
print('Wheel torque =', Tw, 'N*m')

P=Tw*w
print('Power =', P, 'W')
```

Appendix D: CAD Drawings




22.50
1.00
22.50
7.00
43.50
87.00
13.50
3.50
20.99
29.92
Ø6.50
Ø12.75
14.98
9.00
R7.62
9.50
72.00
6.00
6.75
6.00
4.00
30.00
Ø4.00
14.50
12.30
9.38
R3.18
6.97
56.38
25.30
7.00
2.00
4.00
R3.18
4.00
69.00
DRAWN
luis
4/28/2022
CHECKED
QA
MFG
APPROVED
TITLE
SIZE
D
DWG NO
dual battery port
REV
SCALE
2 : 1
SHEET 1 OF 1