

EGR 402 – Capstone Design & Presentation
Design History File – Final Report

Team Name: DiWheel

Team Members: Kari Dennis
Kevin McLaughlin
Christopher Vanjoff
Joshua Dean
Christopher Parisi

Client: Dr. Matthew Rickard

Technical Advisors: Dr. Keith Hekman and Dr. Mark Gordon

Date Submitted: Friday April 19, 2013

Table of Contents

| | |
|------------------------------------------------------------------|----|
| 0. Abstract..... | 5 |
| 1. Milestone 1: Problem Definition and Needs Identification..... | 6 |
| 1.1 Initial Problem Statement..... | 7 |
| 1.2 Client Interview..... | 8 |
| 1.3 Background Information and Relative Technology..... | 10 |
| 1.3.1 Background Information..... | 10 |
| 1.3.2 Theory..... | 10 |
| 1.3.3 Relative Technology..... | 11 |
| 1.4 Objectives Tree..... | 14 |
| 1.5 Pairwise Comparison Chart..... | 15 |
| 1.6 Problem Definition..... | 16 |
| 2. Milestone 2: Requirements Specification..... | 17 |
| 2.1 Constraints..... | 18 |
| 2.2 Standards..... | 19 |
| 2.3 Requirements Specifications..... | 20 |
| 3. Milestone 3: Concept Generation and Selection..... | 22 |
| 3.1 Concept Generation..... | 23 |
| 3.2 Concept Selection..... | 25 |
| 4. Milestone 4: Design Architecture and Detailed Design..... | 27 |
| 4.1 Design Architecture..... | 28 |
| 4.1.1 Product Schematic with Clusters..... | 29 |
| 4.1.2 Geometric Layout..... | 30 |
| 4.1.3 Incidental Interaction Graph..... | 31 |
| 4.2 Electrical Design Architecture..... | 32 |
| 4.2.1 Level 0..... | 32 |
| 4.2.2 Level 1..... | 33 |
| 4.3 Mechanical Detailed Design..... | 43 |
| 4.4 Electrical Detailed Design..... | 53 |
| 4.4.1 Hardware Design..... | 54 |

| | |
|-----------------------------------------------|-----|
| 4.4.2 Software Design | 60 |
| 4.4.3 Diwheel Controls..... | 62 |
| 5. Milestone 5: Prototyping | 63 |
| 5.1 Mechanical Prototyping..... | 64 |
| 5.2 Electrical Prototyping..... | 81 |
| 6. Milestone 6: Testing and Verification..... | 94 |
| 6.1 Mechanical Component Testing..... | 95 |
| 6.1.1 Guide Wheels | 95 |
| 6.1.2 Leveling Set-Up..... | 97 |
| 6.1.3 Drive Train | 100 |
| 6.1.4 Controller Testing..... | 103 |
| 6.2 Electrical Component Testing | 105 |
| 6.2.1 Wireless Receiver/Transmitter | 106 |
| 6.2.2 Drive Wheel Motor Drivers..... | 107 |
| 6.2.3 10A Motor Driver Arduino Shield | 110 |
| 6.2.4 IMU Sensor | 113 |
| 6.2.5 Arduino Uno Microcontroller..... | 115 |
| 6.3 Diwheel System Testing..... | 116 |
| 6.3.1 Systems Integration | 116 |
| 6.3.2 Drivability Testing..... | 119 |
| 6.3.3 Linear motor function testing. | 123 |
| 6.4 Specifications Testing | 125 |
| 6.5 Future Testing Plan..... | 126 |
| Conclusion..... | 127 |
| A. Appendix A: Project Management Plan | 128 |
| A.1 Work Breakdown Structure | 129 |
| A.2 Gantt Chart | 130 |
| A.3 Statement of Work..... | 133 |
| B. Appendix B: Detailed Design | 134 |
| B.1 Drawings..... | 134 |

| | |
|--------------------------------------------------|-----|
| B.2 Specifications | 145 |
| B.3 Parts List and Budget | 156 |
| C. Appendix C: Arduino Sketch | 158 |
| C.1 Initial Diwheel Program | 159 |
| C.2 Updated Diwheel Program | 166 |
| D. Appendix D: MatLab Controls Derivations | 175 |
| D.1 Dynamics Derivations | 175 |
| D.2 Sliding Weight Position Controller | 177 |

0. Abstract

The diwheel has been around for centuries. At its basics, it is two wheels that rotate on the same axis with a center chassis in between them. They have evolved from simple pedal power to now being electrically driven with controls for stability. Most designs for diwheels today are very simple without any stability correcting and as a result the chassis swings and rocks every time the vehicle accelerates or decelerates. This problem is especially pertinent when trying to mount a device onto the chassis. The team realized the problem of being able to keep the chassis level and sought a solution.

The team performed research on related technologies on the market already that could be used as well as the theory that would need to be known. They talked with the client to further define what the project itself would look like and began to formulate a list of objectives that the project would entail. The two main objectives were Marketability and Functionality. A Pairwise comparison chart was made to prioritize the different qualities that the device should possess. After all these steps were taken, a formal problem definition was developed. This definition stated that the team would design a two-wheeled, durable, mobile, radio controlled vehicle that can have different hardware, such as a camera or flashlight, mounted to it. The main chassis will have controls built into it so that the hardware will stay level.

After the formal problem definition was made, the team created the work breakdown structure, which outlined what steps need to be taken to finish the design process. The chart was then fit to a timeline in the Gantt chart.

The team then began to design the device by first defining the constraints, standards, and specifications. The only constraint was that the device would not hinder the functionality of the bike, and the standards had to do with shipping, outlet size, and water/dust proofing. These constraints and standards went into the specifications, which had to be quantifiable and testable.

1. Milestone 1: Problem Definition and Needs Identification

In this milestone, the design team will provide the following sections: Initial Problem Statement, Background Information, Client Interview, Objective Tree, Pairwise Comparison Chart, and Problem Definition. The initial problem statement says what problem the team is trying to solve. The background information is the information necessary to develop an answer. The client interview subsection is a report of how the interview with the client went in discussing the problem definition and the project as a whole. The objective tree is the graphical display of what objectives the team wants to accomplish and all the pieces necessary in completing such task. The pairwise comparison chart is a method used to determine the priority of different qualities that we want the product to have. Finally, the problem definition is the compilation of all of these pieces in a more elaborate version of the problem statement.

1.1 Initial Problem Statement

The design team was given this problem statement from the client with the motivation to improve on the previously attempted projects. The project is being built with military/police application in mind. The following problem statement stands as the original problem with the most basic guidelines to solving it. This statement will be the foundation of the project. It is later updated in section 1.6 based upon the information found in sections 1.2 through 1.5.

The client wants to have a two-wheeled, durable, mobile, radio controlled vehicle that can have different hardware, such as a camera or flashlight, mounted to it. The main chassis needs to have controls built in so that the hardware will stay level.

1.2 Client Interview

We met with our technical advisor in order to get a grip on what technical aspects we would need to get comfortable with in order to complete this project. This included different existing technologies we will encounter as well as the kinds of equations we would need to solve and understand in order for our device to work.

We also set up a meeting with our client to give us a chance to introduce ourselves to who we were working for. It was our goal to clarify the project with him and let him know that we were determined and willing to complete this project in an effective and timely manner. One of our goals related to the client meeting was to gain personal contact with the client and become comfortable with him as he becomes comfortable with us. Another main topic of this discussion was to clarify some concerns we had with the project and get a clear direction.

Our Client is Dr. Matthew Rickard, Mechanical Engineering Professor at California Baptist University. We will meet with him for information and conformation throughout the entire project.

Dr. Rickard: So what is the project you will be designing?

Team DiWheel: We will be designing a diwheel. We would like to build a full scale prototype that uses controls to operate it.

Rickard: How would it be controlled?

Team: We were thinking we would have a joystick for each wheel to allow us to go forward and backward and would also allow us to rotate.

Team: Based on our budget, do you suggest we design a full scale or scaled version of the diwheel?

Rickard: It will depend on the materials. It may be difficult to find a large wheel for a full scale. Research the cost of materials and also availability of parts, such as the wheel, to determine which design route to take.

View examples of diwheels on YouTube

Rickard: What will you implement that will make your design unique and different from diwheel projects that have already been done?

Team: Building a scaled model might be better if we want to add a unique feature. We could focus on designing a functional scaled version and then add new features to it. We could implement suspension and build a jump or track for the diwheel. Adding a safety feature could also be a unique aspect of our design.

1.3 Background Information and Relative Technology

This subsection provides the research that the team has done that is needed to start the project. It includes models and technologies already out in the market that provides similar functions. It also provides the physics and science behind the product itself. For this project in particular, that would include dynamic equations for the forces on the diwheel such as the rotation of the wheel and the center of mass of the chassis.

1.3.1 Background Information

The diwheel (also known as a dicycle) has been around for almost two centuries, this recreational vehicle is constructed of two large outer wheels that are located side by side. These wheels encompass the inner frame where the driver sits, and are usually powered by two electric motors. Sloshing and tumbling have been two major issues with the diwheel since its invention. Sloshing refers to the oscillating motion the inner frame experiences during transportation due to the offset center of gravity. Tumbling is when the inner frame completely flips over with extreme braking and accelerating. Both of these issues can be reduce with the use of damping feedback controls. There has not been an enormous market for these vehicles yet because they have not been legalized for road use and the feedback control system has not been refined to a marketable state.

1.3.2 Theory

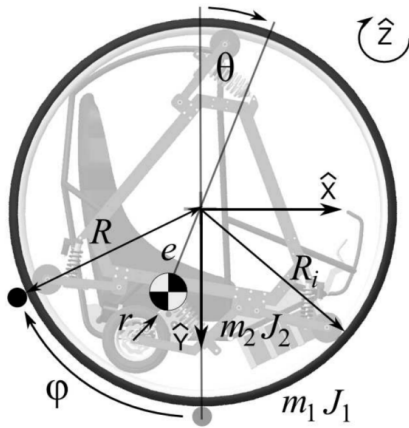
The first step to solving how a diwheel functions is to determine the dynamics behind its motion. The control system can only be started after the equation of motion is determined. There are many equations that can describe the motion of a diwheel and they have to be broken up into several sub equations for simplicity. The diwheel system is comprised of two degrees of freedom in the xy-plane. The system also has three coordinates:

θ is the rotation of the inner frame about the z-axis

$\phi_L = \phi_R = \phi$ which is the rotation of the outer wheels about the z-axis

x is the displacement of the whole diwheel as a system relative to the earth

This is a diagram that shows how the different variables related to the physical structure of the diwheel.



Dynamic equations need to be derived using the listed variables to solve for velocities, kinetic energies, potential energies and the Euler-Lagrange equations. All of these formulas comprise the motion of a diwheel and will be vital when programming the control systems.

Apart from the equations of motion are the equations that are relating to the input velocity and input torque of the motors that are going to power the diwheel. Permanent magnet DC motors with brushes are a common type of electric motor that will serve the purpose of powering the diwheel. The equations that are derived for the electric motor are used alongside the equations of motion to help create a dynamic control system.

1.3.3 Relative Technology

Although the diwheel has been around for a couple hundred years, there is still not an abundance of the actual creation roaming the streets. These vehicles have been limited to hobbyists and student design projects. There is one design that has gone above and beyond what anyone else has done with the diwheel. There was a project that was created by the University of Adelaide that constructed a full sized diwheel equipped with suspension and powered by DC electric motors. They have also shown their equations of motion and briefly discussed the control system they used. There are some other models that are full sized, but they are human

powered. Most of these designs have a through axle that connects both wheels and allows the frame to hang underneath the axle. The few designs that are available to look at will allow our team to gain knowledge and in the construction of our own design.

Simple electric motors are found in many different devices that are used every day. The size and functionality of electric motors range from tiny brushless DC motors used in artificial hearts to enormous AC motors that are used in hydroelectric dams. Electric motors are a useful piece of technology for the consumer, and also very practical for industries that need mechanical energy without the use of combustion engines.

Microprocessors are the future of technology because they are a very important component when constructing computers. Mechanical devices such as vehicles, robots and machines alike rely heavily on this type of technology because there are many physical movements that need to be fed through a microprocessor and then processed so that the right mechanical movement can be implemented.

Other diwheel designs have used various technologies to implement their controls. The diwheel engineered by the University of Adelaide used dSpace rapid prototyping hardware for their controls. Another diwheel team used a wireless router connected to a microcontroller for their control system. Microcontrollers are useful in bringing together all the inputs and outputs in order to gain control of the system. With a diwheel, the microcontroller will control the electric motors and the functionality of the device. It will also contain the algorithms for controlling the slosh of the device.

Sources:

Dicycle, Wikipedia, Mediawiki, page last modified October 3, 2012, site visited October 10, 2012 <<http://en.wikipedia.org/wiki/Dicycle>>

Control of an electric diwheel, B. Cazzolato, J. Harvey, C. Dyer, K. Fulton, E. Schumann, C. Zhu and Z. Prime. University of Adelaide.
<http://data.mecheng.adelaide.edu.au/robotics/projects/2009/EDWARD/DiwheelPaper_v3.pdf>

EDWARD - Electric Diwheel With Active Rotation Damping, University of Adelaide, pd_forms website system.

<http://sites.mecheng.adelaide.edu.au/robotics/robotics_projects.php?wpage_id=44&title=60&browsebytitle=1>

Cazzolato, Ben, Dr. "University of Adelaide Undergraduates Design, Build, and Control an Electric Diwheel Using Model-Based Design." *MathWorks.com*. N.p., 2012. Web. 08 Oct. 2012. <<http://www.mathworks.com/company/newsletters/articles/university-of-adelaide-undergraduates-design-build-and-control-an-electric-diwheel-using-model-based-design.html>>.

Cazzolato, Ben S., Dr., Chris Dyer, Kane Fulton, Jonathon Harvey, Evan Schumann, Charles Zhu, and Luke Francou. "Section Navigation." *Mecheng.adelaide.edu*. The University of Adelaide, 12 Dec. 2010. Web. 08 Oct. 2012.

<http://sites.mecheng.adelaide.edu.au/robotics/robotics_projects.php?browsebytitle=1>.

"Diwheel." *Sariel.pl*. N.p., n.d. Web. 08 Oct. 2012. <<http://sariel.pl/2009/01/diwheel/>>.

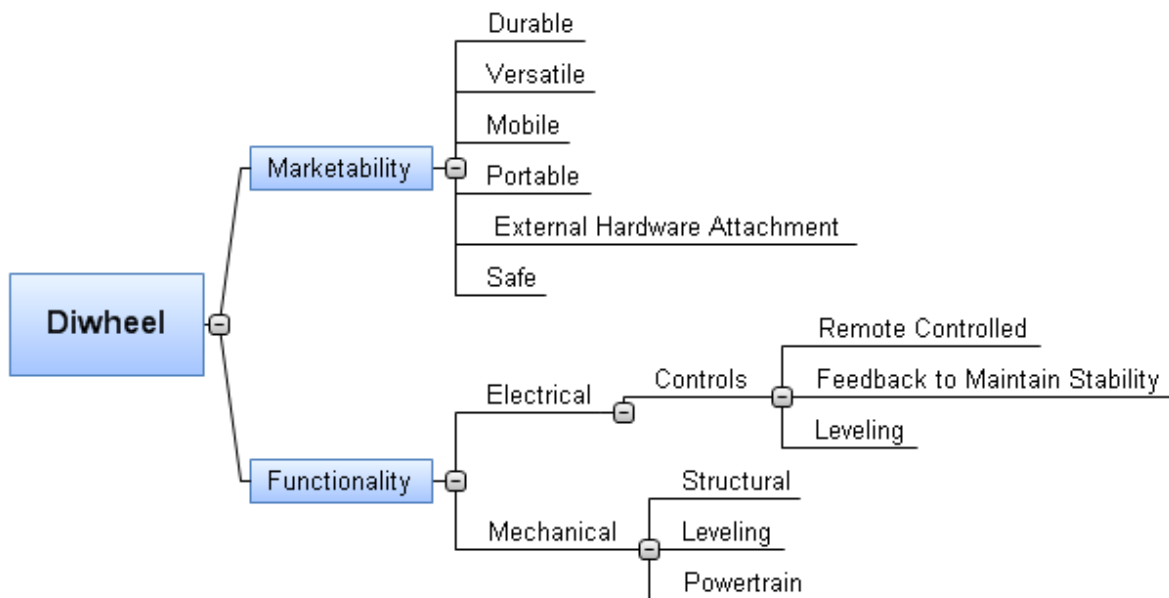
"Diwheel Exploror." *LEGO.com MINDSTORMS*. N.p., n.d. Web. 08 Oct. 2012.

<<http://us.mindstorms.lego.com/en-us/Community/NXTLog/DisplayProject.aspx?id=37b7cc3f-02e3-4258-a4fb-cce6ea91b91a>>.

Mitchdenda. "Rutgers MAE Senior Design 2012 Autonomous Di Wheel Robot." *YouTube*. N.p., 01 May 2012. Web. 08 Oct. 2012. <<http://www.youtube.com/watch?v=WGyEQHuc8ts>>.

1.4 Objectives Tree

An objectives tree is a flow chart of each objective that a certain project needs to accomplish. For each objective on the tree there are subsections that list what the objective must cover. For this specific project there are two main objectives, Functionality and Marketability. For each objective there are subcategories that the objective needs to address. An example of subcategories for this project is for Marketability; it needs to address Durability, Versatility, Mobility, Portability and External Hardware Attachment.



1.5 Pairwise Comparison Chart

As a team, we understand the importance of establishing the priority of different attributes of the device. This is another form of communication among team members by creating a standard that all members will be held to. If there is ever a situation where one attribute will have to be chosen over another, this chart will allow for a quick, decisive decision that will not be disputed. This chart works by comparing one attribute against another and deciding which one has greater priority. If the attribute on the left hand side has greater priority than the attribute in comparison above, a 1 is marked in the box. If it is not of greater priority, a 0 is marked in the box. After all the boxes are marked, the points are added from left to right. The attribute with the highest score has the highest priority.

| | Weight | Maintenance | Mobility | Safety | Leveling | Cost | TOTAL |
|-------------|--------|-------------|----------|--------|----------|------|-------|
| Weight | | 0 | 0 | 0 | 0 | 0 | 0 |
| Maintenance | 1 | | 0 | 0 | 0 | 0 | 1 |
| Mobility | 1 | 1 | | 0 | 0 | 0 | 2 |
| Safety | 1 | 1 | 1 | | 1 | 1 | 5 |
| Leveling | 1 | 1 | 1 | 0 | | 0 | 3 |
| Cost | 1 | 1 | 1 | 0 | 1 | | 4 |

Listed below are the attributes and their descriptions in order of greatest to least importance.

- 5) Safety- This attribute covers the safety for the user of the diwheel. In order to keep the user safe there must be an emergency shut off on the remote. Also, if the signal to the diwheel is lost it will automatically shut off.
- 4) Cost– This attribute entails the cost to build the final product which affects the retail price.
- 3) Leveling – This attribute is the ability of the chasis and attached hardware to stay level while the diwheel is in motion.
- 2) Mobility – This attribute covers the diwheel’s multidirectional specifications. The diwheel will need be able to travel forward, backward, and also turn in order to achieve optimal mobility.
- 1) Maintenance – This attribute covers two main things, durability and repair. The main concern is making the device durable enough to handle obstacles, jumps, and various terrains. This attribute also entails if and how repairs will happen.
- 0) Weight – This pertains to the weight of the diwheel as an entire vehicle. In order to maintain reasonable power consumption and speed, the diwheel will need to be an appropriate weight.

1.6 Problem Definition

The problem definition is a summation of what our product is going to look like and a basic overall representation of what the product should be able to do. These aspects are going to be the guidelines of the project and will give a good idea of how the final product should perform. It will merely be an outline and will not give exact specifications on how the product should perform.

After meeting with the client, we have developed the following problem definition:

The problem that we have been given to solve is to design a two-wheeled, mobile, durable vehicle on which different hardware can be mounted and kept level. Diwheel mobility should include the ability to move forward, backward, and in circles using some form of radio control. For vehicles with two wheels in parallel side by side, the problem comes in keeping the center chassis level. Our project should incorporate a unique way to keep the chassis level while in motion. Because the only connection between the chassis and the wheels is through the drive train, adding any torsional correction for stability will take away from the power being delivered to the wheels. Our team must design a way to incorporate stability without taking away from the power.

2. Milestone 2: Requirements Specification

This milestone includes the first subsections about specifications for the device. Specifications have to be qualities that can be tested and quantified in a repeatable manner. These specifications are driven by three things: Objectives, Constraints, and Standards. Objectives are found in the problem definition and help form what specifications can be made. Constraints are specifications that are driven by the client. Standards are specifications that are driven by government standards. Every specification will fall into at least one objective and will remain inside the bounds of constraints and standards.

2.1 Constraints

Constraints are client driven specifications. The following categories of constraints were emphasized by the client: weight of the components and attached hardware, velocity of the device, impact resistance of the device, economics of the project, acceleration of the device, leveling of the chassis, battery longevity, and safety of the device. Each category is coupled with an exact specification that the client is looking for in the device. Below are the exact specifications that the client is looking for in the end product.

The total weight of the device will not exceed 30kg and will have a target weight of 25kg so that it can be easily transported by no more than two people. Each component, mechanical and electrical, will have a target weight of 15kg.

The velocity of the device will exceed 7.3m/s, the maximum running speed of a human being. The device should be able to catch a human being running at full speed.

The device will be able to drive a minimum of 2 m/s on cement, asphalt, and short grass on a 5 degree incline. It will not be designed to traverse all types of terrain.

The device should be able to withstand a drop from 0.5 meters without experiencing any difficulties performing its functions.

The cost of remote controlled diwheel must be within the given budget of \$1000. The design will be based upon the cost of materials and components.

The device should be able to accelerate to 7.3 m/s, in no more than 3 seconds with a target of 1.5 seconds. This acceleration will be on asphalt, not grass where it may decrease.

The leveling of the chassis is part of the unique design goal. When starting the chassis should not exceed an angle of 15 degrees and when stopping from 7.3 m/s, the chassis should not exceed an angle of -15 degrees. This will keep the attached hardware level while in motion.

The batteries will be rechargeable and be able to last for at least 20 minutes from a fully charged state. This time will allow for short demonstrations of the device.

The safety of the device should include a failsafe unit in case of frequency disturbance or if the receiver becomes out of range; the device should shut down when an error occurs.

The device will be able to be controlled within a minimum range of 100 meters so the user can see the device and be able to safely operate it.

2.2 Standards

Standards are specifications that are driven by government regulations that the device has to follow. The product will utilize wireless communication through radio frequency. The design team will follow the IEEE and FCC standards for frequency range when designing the device.

The IEEE C95.1-2005 standard for “Safety Level with Respect to Human Exposure to Radio Frequency and Electromagnetic Fields” states that ranges should be between 3 kHz and 300 GHz. Our device will operate within this range in order to keep the user and bystanders in close proximity to the device safe from hazardous frequencies.

To protect the device against harmful water and dust damage the Ingress Protection code 54 was chosen. This code states that the device will be water and dust resistant.

2.3 Requirements Specifications

The requirement specifications section is designated to introduce specific design requirements that the product will strictly follow. There are boundaries that have been established in order to keep the design following a specific path. There is a table that illustrates the specification that the design must abide by, a justification or reason why this specification was put in place and finally the objective that the specification will support. A list of the product objectives will also be listed for a quick comparison.

Objectives

- 1 Durable
- 2 Manoeuvrability
- 3 Mobile
- 4 Portable
- 5 External Hardware Attachment
- 6 Remote Controlled
- 7 Electrical Leveling
- 8 Structural
- 9 Mechanical Leveling
- 10 Drive Train
- 11 Safety
- 12 Price

| Objective | Specification | Justification |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3,4 | The mechanical components of the device will not exceed a weight of 15kg and will have a target weight of 10kg. | The completed device will need to be easily transported by a maximum of two people. |
| 3,4 | The electrical components of the device will not exceed a weight of 15kg and will have a target weight of 10kg. | The completed device will need to be easily transported by a maximum of two people. |
| 1,11 | The vital components of the device need to be water and dust resistant and follow code IP54 (Ingress Protection Code). | The device is to be water and dust resistant so it can still properly function while driving in wet or dusty conditions. |
| 2,3,10 | The maximum velocity of the device will exceed the maximum running velocity of a human being, 7.3 m/s. | The device will be able to catch up to a human being running at full speed, approximately 20 mph. |
| 1,8 | The vehicle should be able to sustain a drop from 0.5 meters and not lose any function. | The vehicle should be designed to be able to traverse rough terrain. As a design goal we want it to be able to be dropped and fall in any orientation and still maintain functionality. |
| 12 | The target price for production of this device is approximately \$1000. | This project has been given a budget of \$1000. |
| 2,3,10 | The device should be able to accelerate to top speed, 7.3 m/s, in no more than 3 seconds with a target of 1.5 seconds. | The device should be able to reach full speed in an equivalent time that a human would reach top sprint speed. |
| 2,3,5,8,9 | The device will be able to hold external hardware with a maximum weight of 7.5 kg | The weight of the hardware should be no more than half of the weight of the device. |
| 2,3,10 | The device chassis should not exceed a maximum angle of 15 degrees when starting from a stop and accelerating straight forward. It also needs to not exceed a maximum angle of 15 degrees when stopping from 10 kph. | The device chassis should stay level throughout its use. The purpose is to create a platform that stays level so that hardware such as a camera can be attached to the chassis and kept level. |
| 6,7 | The batteries will be rechargeable and able to power the device for at least 10 minutes from a fully charged state. The target longevity of the batteries will be 20 minutes. | The device will have the battery capacity to put on short demonstration shows. |
| 6 | The device will be equipped with a failsafe unit in case of battery loss, frequency disturbance or if the receiver becomes out of range | The reason for adding a failsafe unit to the device is for safety. The device will shut down if an error occurs. |

3. Milestone 3: Concept Generation and Selection

This milestone consisted of two subcategories, the first being our concept generation and the second being our concept scoring matrix. The concept generation sub category is an essential step in considering a multitude of different design options that are realistic as well as unrealistic. It is here that creativity is allowed to flow. After obtaining a number of design alternatives from the morphological chart, there needs to be an organizational process that rates the different concepts that have been previously established. A concept scoring matrix will be used to do this. This chart will allow an outside perspective of how each concept compares to each other and will determine an ideal model to prototype. This milestone is critical so that no design option will be left out. Finally this process will determine what ideas will advance onto the next step in the design process and which designs will not.

3.1 Concept Generation

The concept generation is a key component in the design process because it gives the opportunity to have a multitude of different designs that capitalize on a variety of different shapes, materials, fasteners and devices. Two morphological charts were used to organize all of the different ideas. One chart was the hardware aspects of the design and the other being the materials aspect of design. Different design concepts can be created by mixing and matching the different design features. By creating these charts it was ensured that no design would be left unaccounted for and all possible designs would be analyzed.

Hardware Morphological Chart

| | | | | | | |
|------------------------|----------------------|----------|-------------------|--------------|------|---|
| Drive train | Chain | Gears | Wheel to Wheel | Direct | Belt | |
| Electric Motor | Brushless | DC | AC | --- | --- | 1 |
| Controls | Arduino | FPGA | PSoC | MultiWii Pro | --- | 2 |
| Dynamic Leveling | Counterweight Slider | Pendulum | Rotating Flywheel | --- | --- | 3 |
| Static Leveling | Counterweight Slider | Pendulum | --- | --- | --- | 4 |
| Leveling (Measurement) | Integrated | Laser | Separate | --- | --- | |

Our first prospective design is highlighted in the hardware morphological chart as the green design. This design will be driven by wheel to wheel contact powered by a DC motor. It will be controlled by the Arduino Uno Controller in order to handle the mechanical and electrical inputs and outputs of the control system. This design uses a counterweight slider and integrated leveling sensors to help the chassis stay level. This concept will be cost effective, lightweight, and capable of controlling the “sloshing” movement of the chassis.

The second design is highlighted in blue. This model will have a chain drive train and be powered by a brushless DC motor. The MultiWii Pro controller will be used for the controls along with an infrared laser to measure the leveling. These will be used in correspondence with a counterweight slider for static leveling and a rotating flywheel for dynamic leveling.

The third design is highlighted in purple. This design will have a belt drive train powered by an AC motor. The static and dynamic leveling will be controlled by an FPGA controller. Clips will be used to mount the hardware to the chassis. A high strength belt will be used in unison with an AC electric motor to apply mechanical energy for the device. The control system will be handled by a Field Programmable Gate Array (FPGA) and stand alone sensors which will interact with the control algorithm programmed on the FPGA.

The fourth design is highlighted in yellow. This design will be driven by direct drive train, meaning straight from the shaft of the DC motor to the wheel. The device will be driven from a direct connection between a DC motor and the axel. A Programmable System on Chip (PSoC) will be used with a laser sensor to control the leveling as well as control of the design. The leveling system will be a rotating flywheel for the dynamic leveling.

Materials Morphological Chart

| | | | | | | |
|----------------------------|-----------------|--------------------|--------------|------------------|-------|---|
| Material (Housing) | Aluminum | Plastic Mold | Plexiglas | Carbon Fiber | --- | A |
| Shape (Housing) | Cube | Triangular Prism | Cylinder | Sphere | | B |
| Material (Chassis) | Aluminum | Steel | Carbon Fiber | Plastic | Wood | C |
| Shape (Chassis) | Platform(Swing) | Rectanglular Prism | Cylinder | Triangular Prism | --- | D |
| Hardware Attachment | Suction Cup | Clips | Straps | Clasps | Bolts | |

Our first prospective design for the materials morphological chart is highlighted in orange. The housing of the diwheel will be made from a plastic mold that will be in the shape of a triangular prism. The chassis material will be aluminum and will also be in the shape of a triangular prism. For the hardware, it will be attached with bolts straight onto the chassis.

The second design is shown in purple. The housing will be built out of Plexiglas and put in a cube shape. The chassis will be a platform made of steel. The hardware will use clasps to attach and detach from the chassis.

The third design is highlighted in pink. This design will have a housing made of aluminum in the shape of a cylinder. The chassis of the diwheel will be a plastic rectangular prism that will use clips to attach external hardware to it.

The fourth prospective design is highlighted in blue. The housing of the diwheel will be made from carbon fiber that will be in the shape of a triangular prism. The chassis will also be made of carbon fiber in the shape of cylinder. The hardware will be attached using suction cups.

3.2 Concept Selection

The concept scoring matrix is a chart that is used to categorize and compare the different concept ideas that were created in the morph chart. Criteria were determined based on the project objectives. These criteria were then assigned a weight in order of their importance in regards to the problem definition. Each concept design was then rated in each of the different criteria categories and then given a score. This chart will allow us to determine how each design ranks compared to one another and whether or not the design should be developed.

Hardware Concept Scoring Matrix

| | | Concepts | | | | | | | |
|------------------|-----|-----------|-------|-----------|-------|-----------|-------|-----------|-------|
| | | Concept 1 | | Concept 2 | | Concept 3 | | Concept 4 | |
| Criteria | Wt | Rating | Score | Rating | Score | Rating | Score | Rating | Score |
| Weight | 5% | 4 | 0.2 | 4 | 0.2 | 5 | 0.25 | 5 | 0.25 |
| Maintenance | 10% | 5 | 0.5 | 3 | 0.3 | 3 | 0.3 | 3 | 0.3 |
| Mobility | 15% | 4.5 | 0.68 | 4.5 | 0.675 | 4.5 | 0.68 | 4.5 | 0.675 |
| Safety | 30% | 4 | 1.2 | 4 | 1.2 | 4 | 1.2 | 4 | 1.2 |
| Leveling | 17% | 5 | 0.85 | 5 | 0.85 | 3 | 0.51 | 3 | 0.51 |
| Cost | 23% | 5 | 1.15 | 1.5 | 0.345 | 2 | 0.46 | 3 | 0.69 |
| Total | | 4.575 | | 3.57 | | 3.395 | | 3.625 | |
| Rank | | 1 | | 2 | | 4 | | 3 | |
| Continue? | | Develop | | No | | No | | No | |

The first concept scoring matrix scored the hardware of the design. Concept 1 one was selected for development because it received the highest score in the scoring matrix. The top criteria for the matrix include safety, cost, and leveling. According to the above chart, concept 1 received high scores in each of these important categories. The device may be used for a military or police application so keeping the chassis level will be of high importance. Safety will receive a high score due to the fact that it will have a plastic housing. This design received a high score for cost because it is estimated to have a relatively low cost to prototype compared to the other conceptual designs. Using the Arduino Uno to control the leveling gave Concept 1 a high ranking because it is a powerful cost effective device. The mechanical aspect of leveling, achieved with a counterweight slider, will be very efficient and require low maintenance. This design received a high score for weight because the components used in the system are fairly light in weight.

Material Concept Scoring Matrix

| | | Concepts | | | | | | | |
|------------------|-----|-----------|-------|-----------|-------|-----------|-------|-----------|-------|
| | | Concept A | | Concept B | | Concept C | | Concept D | |
| Criteria | Wt | Rating | Score | Rating | Score | Rating | Score | Rating | Score |
| Weight | 5% | 4 | 0.2 | 2 | 0.1 | 5 | 0.25 | 5 | 0.25 |
| Maintenance | 10% | 5 | 0.5 | 5 | 0.5 | 5 | 0.5 | 5 | 0.5 |
| Mobility | 15% | 5 | 0.75 | 5 | 0.75 | 5 | 0.75 | 5 | 0.75 |
| Safety | 30% | 5 | 1.5 | 4 | 1.2 | 4 | 1.2 | 4 | 1.2 |
| Leveling | 17% | 5 | 0.85 | 5 | 0.85 | 5 | 0.85 | 5 | 0.85 |
| Cost | 23% | 5 | 1.15 | 4 | 0.92 | 4 | 0.92 | 1 | 0.23 |
| Total | | 4.95 | | 4.32 | | 4.47 | | 3.78 | |
| Rank | | 1 | | 2 | | 4 | | 3 | |
| Continue? | | Develop | | No | | No | | No | |

The second concept scoring matrix scored the material and shape of the design. Concept A was selected for development because it received the highest score in the matrix. Weight received a high score because the material chosen was lighter in weight. Maintenance scored high because the material and shape chosen will not cause many breaks and failures. Mobility scored high because the material is light making the device more mobile. The shape also yields more mobility. Safety scored high because the material chosen does not cause any safety issues like sharp metallic corners. Leveling scored high because the material and shape will make leveling easier. Finally, cost scored high because the material chosen is not very expensive. Careful analysis of this concept how it compares to the project objectives lead us to choose concept 1A for further development.

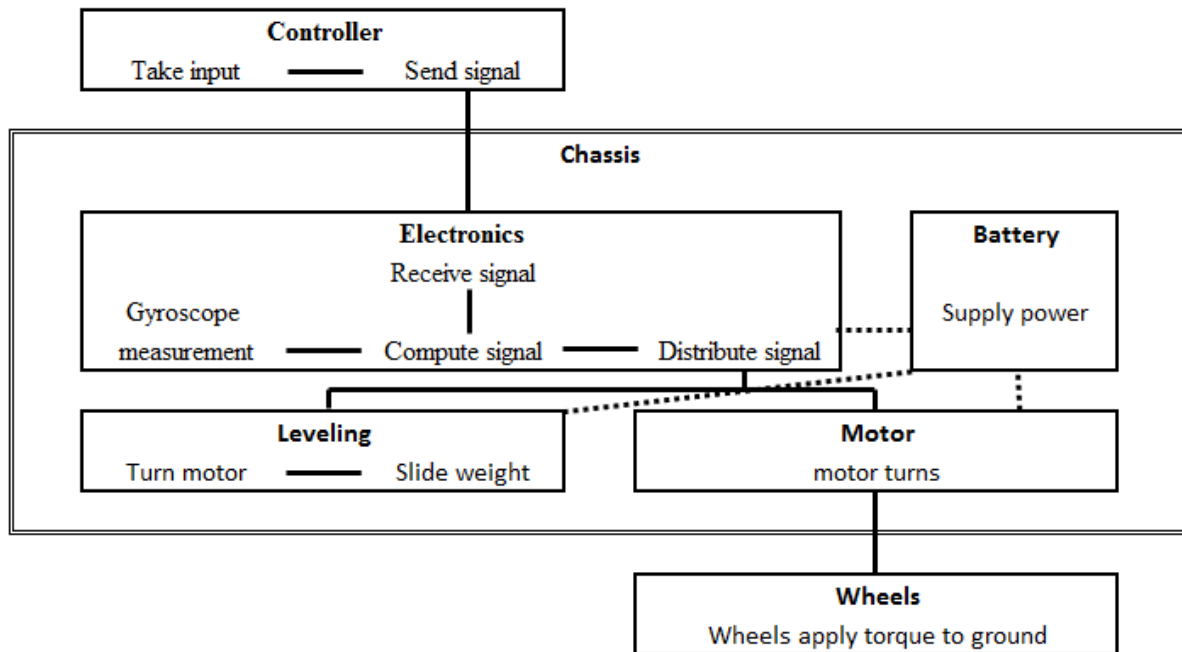
4. Milestone 4: Design Architecture and Detailed Design

This is the last milestone deliverable and contained most of what the actual design was going to look like. This milestone had the most detail and required the most amount of work because precise and detailed CAD drawings needed to be drawn. Again there were two subcategories, the first being design architecture and the second being detailed design. The design architecture was necessary in order to visually and technically see how the subcomponents would fit together as a final assembly. This step ensured that there would be no physical, mechanical or electrical conflict between components. After the all of the bugs were flushed out from the design architecture, inauguration of the detailed drawings could begin. This is where everything came together and the first prototype could be visually referenced. CAD parts and assemblies were created to exact dimensions; this was the first step in creating an actual prototype.

4.1 Design Architecture

The design architecture is an important step to help orientate the different components of the entire assembly. It helps in visually representing where each piece will be located and how each piece will interact with one another. A product schematic with clusters was created in order to break up the entire design into subcategories and document how they related to each other. An incident interaction graph was also a requirement for the design architecture. This was a graph that showed the different components of the product and what potential problems might occur due to their interaction with each other. Three main components include in the design are the Controller, the Chassis, and the Wheels. The relationship between these three components can be seen below in the following subsection.

4.1.1 Product Schematic with Clusters



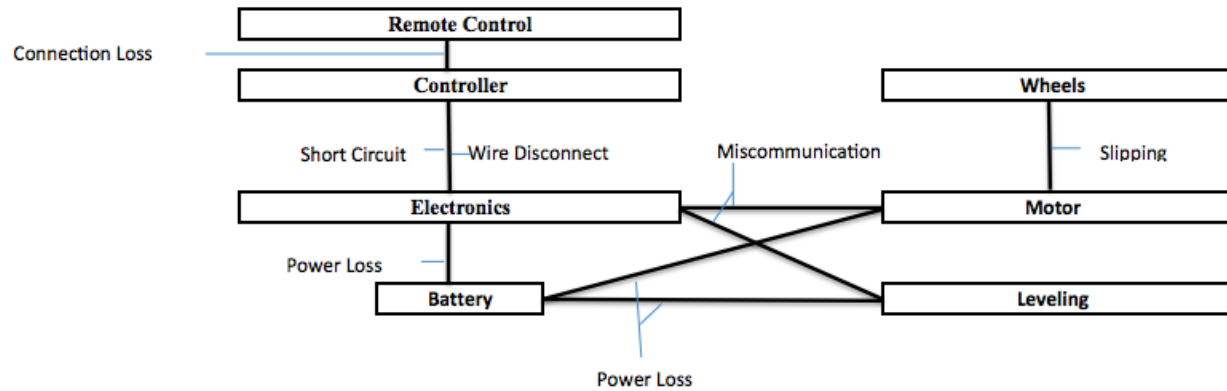
The Product Schematic with Clusters is a graphical representation of the flow of information and energy while also representing the physical interaction. The design is broken up into clusters that have their different function defined inside. The main clusters that we have are the controller, the chassis and the wheels. The chassis is also further broken down into electronics, battery, leveling, and motor. As seen below, the black lines denote the transfer of information, and the dotted lines represent the transfer of power.

4.1.2 Geometric Layout



The Geometric Layout is graphical layout of what we expect our design to resemble. Included in the Geometric Layout is a physical representation of the different clusters that are defined in the Product Schematic with Clusters. This representation is not any final layout or design, but just a way for the general layout and interaction to be visualized.

4.1.3 Incidental Interaction Graph



The Incidental Interaction Graph demonstrates what can go wrong for each interaction between each cluster. Some examples of things that can go wrong are loss of signal, slipping between the motor and wheels, and short circuiting. By brainstorming for what can go wrong, we can attempt to plan ahead to ensure that these things do not happen.

4.2 Electrical Design Architecture

The design architecture describes the modules and how they are connected together to form the complete system. This section describes the decomposition and functionality of each component. The Design Architecture is important because it shows a visual of theoretical connections. This sections describes and shows the components, how they are connected, and the interfaces between them. The main components for the electrical side of the design architecture are the microcontroller, wireless communication chips, Inertia Measurement Unit, and the H-bridge. These components are described in the following sections.

4.2.1 Level 0

The diwheel will be remotely controlled by the user. As the user interacts with the controller, the diwheel will respond and move accordingly. This is the highest level of the design and shows the basic interaction between the user and the response of the diwheel. A diagram of this top level can be seen in Figure 4.1.1 and an overview of the functionality, inputs, and outputs can be found in Table 4.1.1.



Figure 4.2.1: Level 0 Diwheel Functionality

| | |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Module | Hardware |
| Inputs | -Directional data from user's controller (forward, backward, left, right) -Current inertial data from diwheel (leveling) |
| Outputs | A voltage to control direction of DC motors |
| Functionality | Using the data received from the user's controller and the current status of the diwheel's inertia, the control system will output a positive or negative voltage to each DC motor in order to make the diwheel move as the user instructs. |

Table 4.2.1: Diwheel Functionality

4.2.2 Level 1

Going further down into the design, level 1 describes the functionality of the hardware and its modules. As the user interacts with the controller, the data is wireless transmitted to the receiver on the diwheel. Once received, the data is passed to the microcontroller. The microcontroller accepts the user's commands as well as the Inertial Measurement Unit (IMU). The program in the microcontroller will then determine the needed output to the electric motors based on the current inertia of the diwheel and user's commands. The output from the microcontroller will be passed through an H-bridge and then to the DC motors. The diagram of this level can be found in Figure 4.1.2.

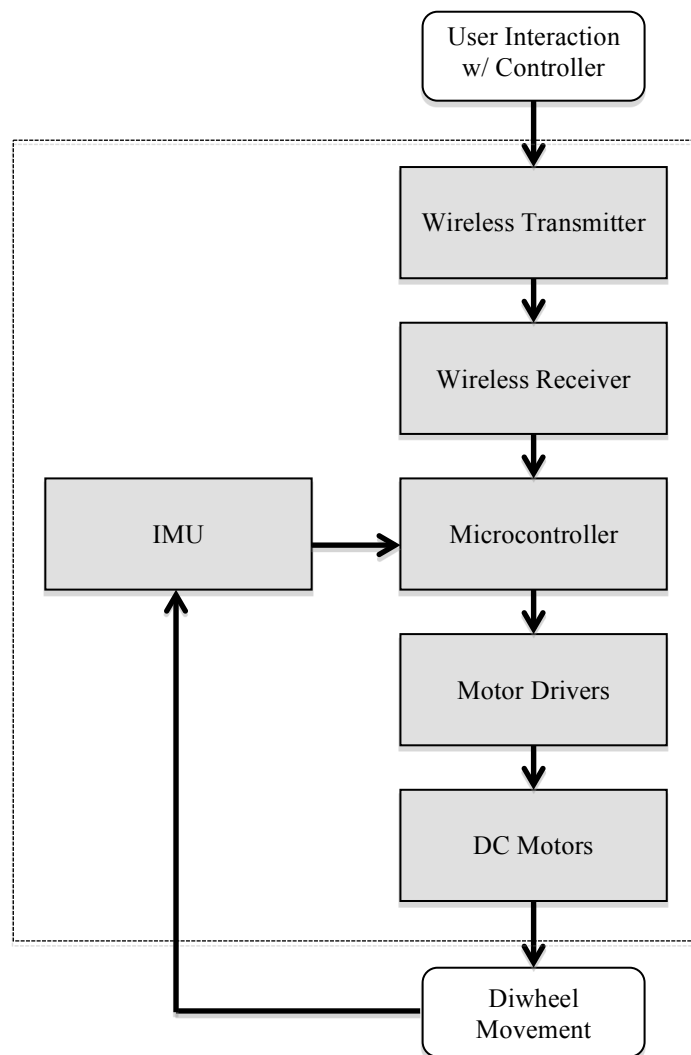


Figure 4.2.2: Level 1 Hardware Functionality and Modules

Wireless Receiver
Wireless Inventors Shield for Arduino
RFD21815

For the wireless receiver, we have chosen to use the Wireless Inventors Shield for Arduino. This device can be easily attached to the Arduino and allows for up to a 500ft range. This shield comes equipped to receive inputs from the transmitter without the need for initialization and setup. A photograph of this module can be found in Figure 4.1.3. This module is attached directly to the Arduino so the inputs are already configured to work. A detailed schematic of this module can be found in the Detailed Design portion of the report.

| | |
|----------------------|----------------------------------------------------------------------------------------------------------------|
| Module | Wireless Receiver |
| Inputs | Data Transmitted from the Wireless Transmitter |
| Outputs | Data which described the user's commands |
| Functionality | The wireless receiver will receive the data provided by the wireless transmitter which is the user's commands. |

Table 4.2.2: Wireless Receiver Functionality

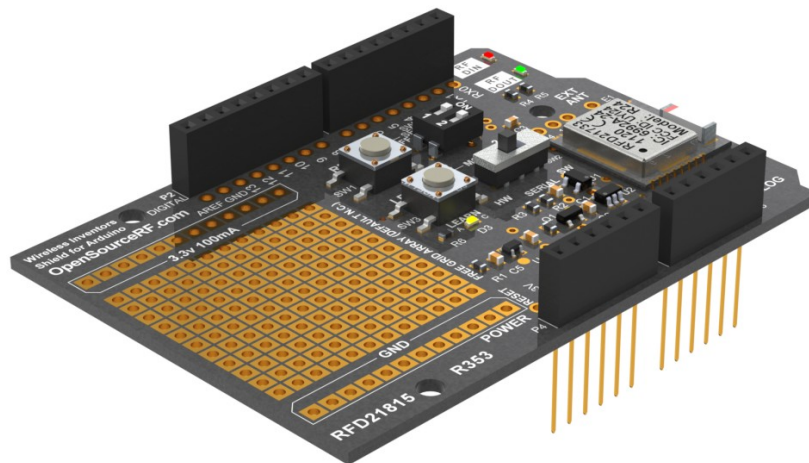


Figure 4.2.3: Wireless Inventors Shield for Arduino

<http://www.kickstarter.com/projects/1608192864/open-source-wireless-inventors-shield-for-arduino>
<http://www.OpensourceRF.com/rfd21815-wireless-inventors-shield-for-arduino.html>

Wireless Transmitter
Wireless RF USB Dongle
RFD21807

For the wireless transmitter, we have chosen to use the Wireless RF USB Dongle. This device is created by the same company as the Wireless Inventors Shield and will be easily interfaced. This USB dongle will accept user serial input data from the HyperTerminal on the PC. The device also has a range of 500ft. A photograph of this module can be found in Figure 4.1.4. A detailed schematic of the wireless transmitting chip can be found in the Detailed Design portion of the report.

| | |
|----------------------|------------------------------------------------------------------------------------------------------------------------------|
| Module | Wireless Transmitter |
| Inputs | Serial Data From the Controller (PC) |
| Outputs | Data Transmission |
| Functionality | The wireless transmitter will take the User's Control entered from the PC, and transmit the data wirelessly to the receiver. |

Table 4.2.3: Wireless Transmitter Functionality



Figure 4.2.4: Wireless RF USB Dongle

<http://www.kickstarter.com/projects/1608192864/open-source-wireless-inventors-shield-for-arduino>
<http://www.opensourcerf.com/rfd21815-wireless-inventors-shield-for-arduino.html>

Microcontroller
 Arduino Uno R3
 DEV-11021

For the microcontroller, we have selected the Arduino Uno as our component to handle all calculations and data flow. The Arduino Uno is an affordable yet powerful microcontroller that is easy to program and interface with the other electrical components. We will be using the Arduino to handle all leveling related calculations as well as control of the DC motors.

| | |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Module | Microcontroller |
| Inputs | User input from wireless receiver Inertia data from IMU sensor |
| Outputs | Voltages for controlling Motor Drivers |
| Functionality | The microcontroller will gather the input from the user as well as the feedback from the IMU sensor and determine what the needed voltage for the DC motors will be. |

Table 4.2.4: Microcontroller Functionality

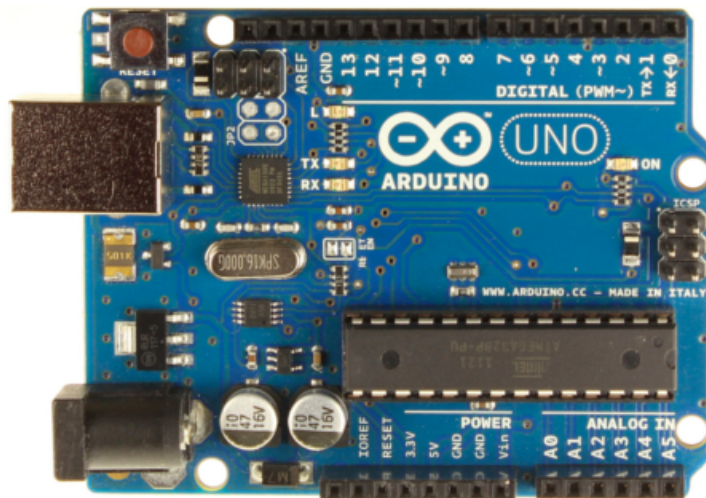


Figure 4.2.5: Arduino Uno R3 Front

<http://www.arduino.cc/en/Main/arduinoBoardUno>

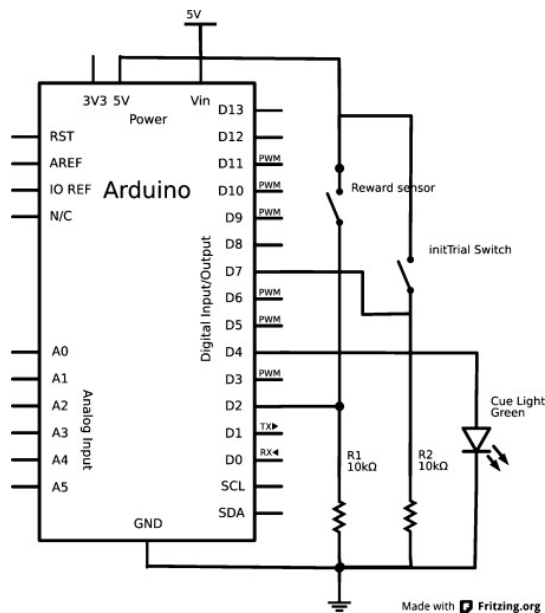


Figure 4.2.6: Arduino Uno Block Diagram

<http://www.sciencedirect.com/science/article/pii/S0165027012003846>

| Name | Direction | Description |
|-----------|-----------|--------------------------------------|
| Reset | Input | Resets Microcontroller When LOW |
| AREF | Output | Analog Input Reference Voltage |
| IOREF | Output | Input/Output Reference Voltage |
| 3.3V | Output | 3.3V Output |
| 5V | Output | 5V Output |
| GND | Input | Common Ground |
| GND | Input | Common Ground |
| GND | Input | Common Ground |
| A0 | Input | Analog Input 0 |
| A1 | Input | Analog Input 1 |
| A2 | Input | Analog Input 2 |
| A3 | Input | Analog Input 3 |
| A4 | Input | Analog Input 4 |
| A5 | Input | Analog Input 5 |
| D0 (RX) | Either | Digital Pin 0 (Receive serial data) |
| D1 (TX) | Either | Digital Pin 1 (Transmit serial data) |
| D2 | Either | Digital Pin 2 |
| D3 (PWM) | Either | Digital Pin 3 |
| D4 | Either | Digital Pin 4 |
| D5 (PWM) | Either | Digital Pin 5 |
| D6 (PWM) | Either | Digital Pin 6 |
| D7 | Either | Digital Pin 7 |
| D8 | Either | Digital Pin 8 |
| D9 | Either | Digital Pin 9 |
| D10 (PWM) | Either | Digital Pin 10 |
| D11 (PWM) | Either | Digital Pin 11 |
| D12 | Either | Digital Pin 12 |
| D13 | Either | Digital Pin 13 |

Table 4.2.5: Arduino Uno Input/Output

Motor Driver (For 250W Motors)
 Pololu High-Power Motor Driver 24v23 CS
 Pololu Item #1456

A motor driver is needed to control the speed and direction of the DC motors. We chose the Pololu High-Power Motor Driver because of its capability to handle high power. The driver also contains a PWM signal input which allows the control of the motor speed. The motor driver has a high power side and a lower power digital side. The digital side uses the output from the Arduino and the power side handles the high power from the batteries to the motors.

| | |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Module | Motor Driver (250W Motors) |
| Inputs | PWM signal from Microcontroller Direction signal from Microcontroller 24V From Batteries |
| Outputs | Voltage to control speed and direction of DC motors |
| Functionality | The motor driver will take in the output voltage of the microcontroller and output the correct polarity to the DC motors. Controls the DC motors to go forwards or backwards. The PWM signal controls the average voltage allowed to pass through to the motors which controls the speed. |

Table 4.2.6: Motor Driver (23A) Functionality

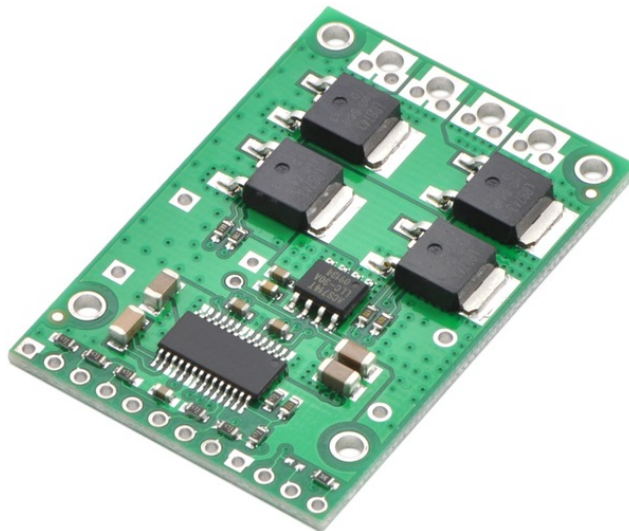


Figure 4.2.7: Pololu High-Power Motor Driver

<http://www.pololu.com/catalog/product/1456>

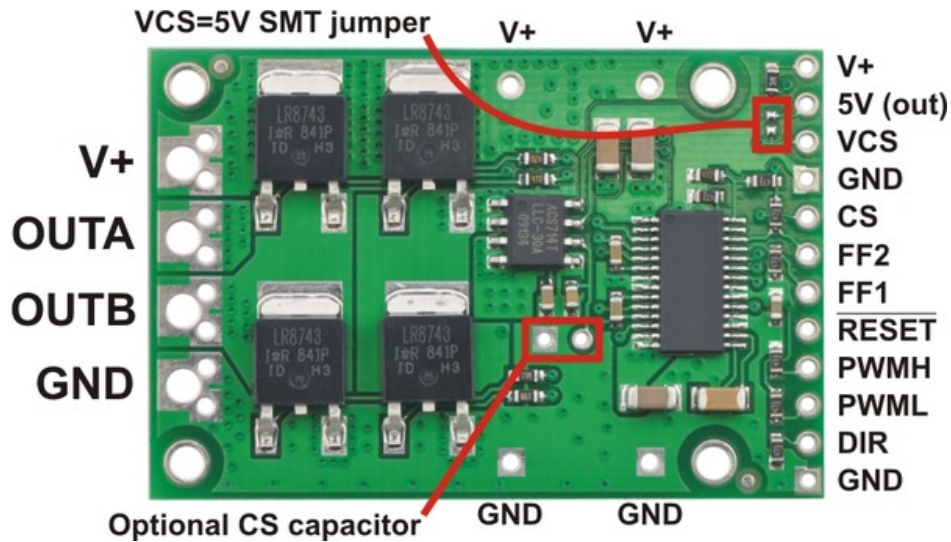


Figure 4.2.8: Pololu High-Power Motor Driver Pins

<http://www.pololu.com/catalog/product/1456>

| PIN | Default State | Description |
|----------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| V+ | | This is the main 5.5 – 40 V (absolute max) motor power supply connection, which should typically be made to the larger V+ pad. The smaller V+ pads along the long side of the board are intended for power supply capacitors, and the smaller V+ pad on the logic side of the board gives you access to monitor the motor's power supply (it should not be used for high current). |
| 5V (out) | | This regulated 5V output provides a few milliamps. It can be shorted to VCS to power the current sensor. This output should not be connected to other external power supply lines. Be careful not to accidentally short this pin to the neighboring V+ pin while power is being supplied as doing so will instantly destroy the board! |
| VCS | | Connect 5 V to this pin to power the current sensor. |
| GND | | Ground connection for logic and motor power supplies. |
| CS | | ACS714 current sensor output (66 mV/A centered at 2.5 V). |
| OUTA | | A motor output pin. |
| OUTB | | B motor output pin. |
| PWMH | LOW | Pulse width modulation input: a PWM signal on this pin corresponds to a PWM output on the motor outputs. |
| PWML | HIGH | Control input that enables coasting when both PWML and PWMH are low. See the "motor control options" section below for more information. |
| DIR | LOW | Direction input: when DIR is high current will flow from OUTA to OUTB, when it is low current will flow from OUTB to OUTA. |
| RESET | HIGH | The reset pin, when pulled low, puts the board into a low-power sleep mode and clears any latched fault flags. |
| FF1 | LOW | Fault flag 1 indicator: FF1 goes high when certain faults have occurred. See table below for details. |
| FF2 | LOW | Fault flag 2 indicator: FF2 goes high when certain faults have occurred. See table below for details. |

Table 4.2.7: Pololu High-Power Motor Driver Pin Descriptions

<http://www.pololu.com/catalog/product/1456>

Motor Driver (For Metal Gear Motor)
10A DC Motor Driver Arduino Shield
RB-Cyt-116

Another motor driver is needed to control the speed and direction of the small linear motor. We chose the 10A DC Motor Driver Arduino Shield because it operated in the range we require and fit into our limited space. The shield connect directly to the Arduino and also the Wireless Inventors shield. A couple of the pins are designated for controlling the speed and direction of the small metal gear DC motor.

| | |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Module | Motor Driver (W Motors) |
| Inputs | PWM signal from Microcontroller Direction signal from Microcontroller 24V From Batteries |
| Outputs | Voltage to control speed and direction of DC motors |
| Functionality | The motor driver will take in the output voltage of the microcontroller and output the correct polarity to the DC motors. Controls the DC motors to go forwards or backwards. The PWM signal controls the average voltage allowed to pass through to the motors which controls the speed. |

Table 4.2.8: Motor Driver (metal gear motor) Functionality



Figure 4.2.9: 10A DC Motor Driver Arduino Shield

<http://www.robotshop.com/10a-dc-motor-driver-arduino-shield-2.html>

Inertial Measurement Unit
 Triple Axis Accelerometer & Gyroscope Breakout Board
 MPU-6050

An Inertial Measurement Unit is a device that will gather data about the inertia of the IC using the on board accelerometers and gyroscopes. Our diwheel will be using the Triple Axis Accelerometer & Gyro in order to measure the inertia of the product while in motion. This component is key in keeping the device level as it will be the source of feedback to the microcontroller. Because the IC is very small, we will be using the breakout board in order to make accessing the ports easier.

| | |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Module | Inertial Measurement Unit |
| Inputs | Inertial forces acting on device |
| Outputs | Data describing current status of the diwheel’s inertia |
| Functionality | The accelerometers and gyroscopes inside the IMU will provide data describing the angle, pitch, and momentum of the device. |

Table 4.2.9: IMU Functionality

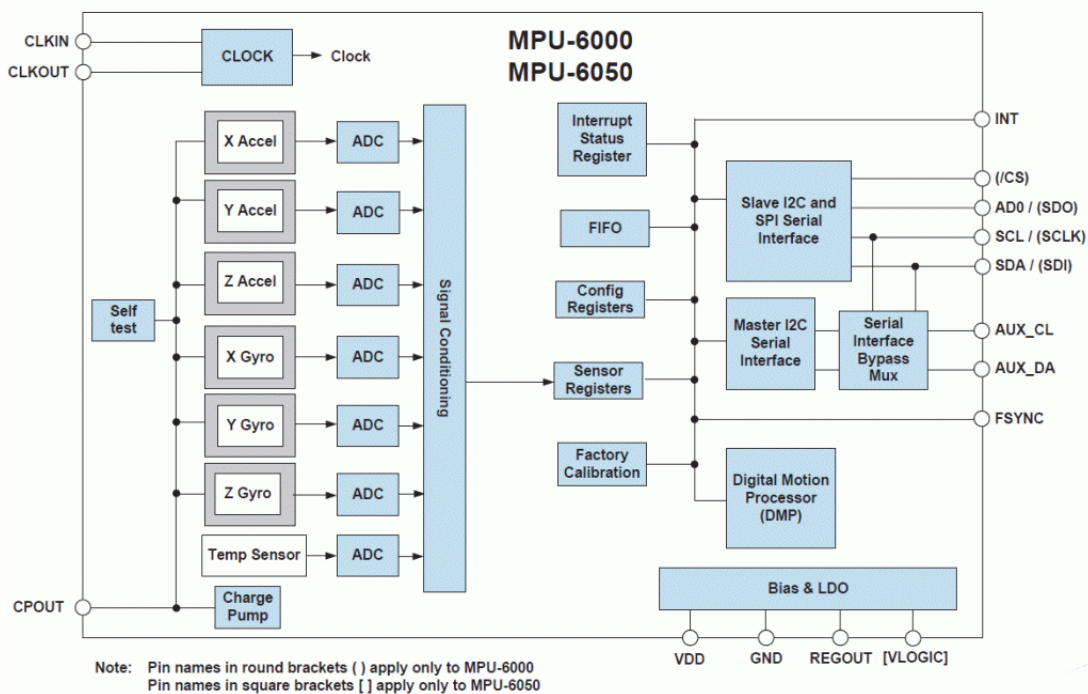


Figure 4.2.10: MPU-6050 IC Block Diagram

| Name | Direction | Description |
|---------|-----------|--------------------------------------------------------------------|
| VCC | Input | Power Supply Voltage and Digital I/O supply voltage |
| GND | Input | Common Ground |
| INT | Output | Interrupt digital output (totem pole or open-drain) |
| FSYNC | Input | Frame synchronization digital input. Connect to GND if unused. |
| SCI | Input | I2C serial clock (SCL); SPI serial clock (SCLK) |
| SDA | Output | I2C serial data |
| VIO | Input | SPI chip select (0=SPI mode) Digital I/O supply voltage |
| CLKIN | Input | Optional external reference clock input. Connect to GND if unused. |
| AUX_SCL | Input | I2C Master Serial Clock, for connecting to external sensors |
| AUX_SDA | Input | I2C Master Serial Data, for connecting to external sensors |

Table 4.2.10: IMU Breakout Board Inputs/Outputs

| Parameter | Rating |
|----------------------------------------------------------------|----------------------------------------------|
| Supply Voltage, VDD | -0.5V to +6V |
| VLOGIC Input Voltage Level (MPU-6050) | -0.5V to VDD + 0.5V |
| REGOUT | -0.5V to 2V |
| Input Voltage Level (CLKIN, AUX_DA, AD0, FSYNC, INT, SCL, SDA) | -0.5V to VDD + 0.5V |
| CPOUT (2.5V ≤ VDD ≤ 3.6V) | -0.5V to 30V |
| Acceleration (Any Axis, unpowered) | 10,000g for 0.2ms |
| Operating Temperature Range | -40°C to +105°C |
| Storage Temperature Range | -40°C to +125°C |
| Electrostatic Discharge (ESD) Protection | 2kV (HBM); 200V (MM) |
| Latch-up | JEDEC Class II (2), 125°C Level A, ±100mA |

Table 4.2.11: IMU Breakout Board Absolute Maximum Ratings

<https://www.sparkfun.com/products/11028>

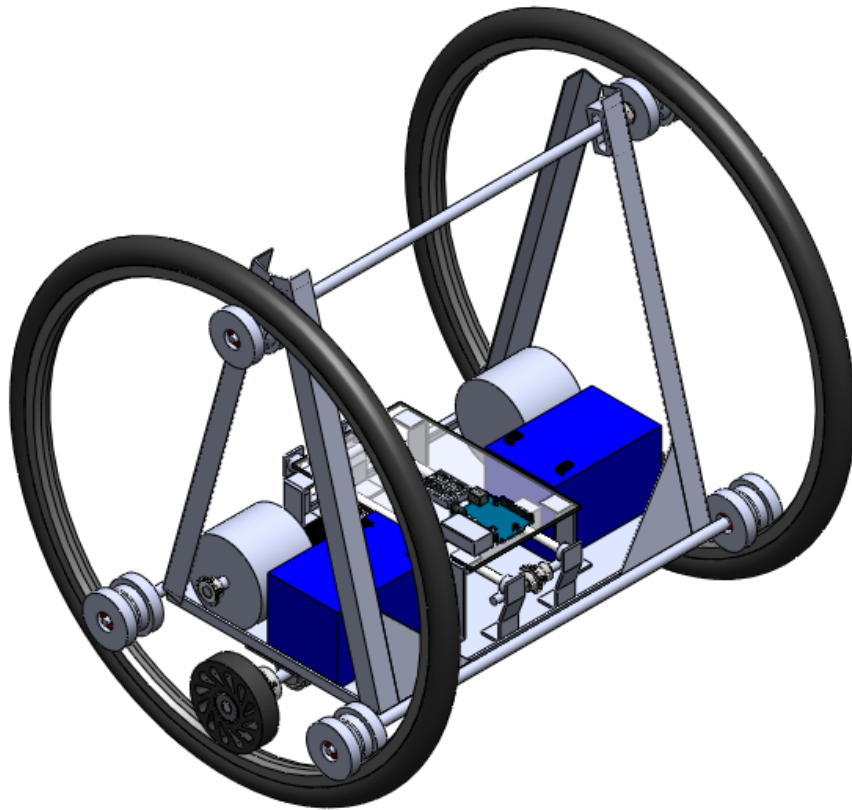
<http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General%20IC/PS-MPU-6000A.pdf>

<http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General%20IC/PS-MPU-6000A.pdf>

4.3 Mechanical Detailed Design

The detailed design portion of this project could be one of the most important pieces to the puzzle. This section takes everything that has been done so far and compiles everything that has been documented into a first prototype design. It will be a visual model off what the actual prototype is going to look like.

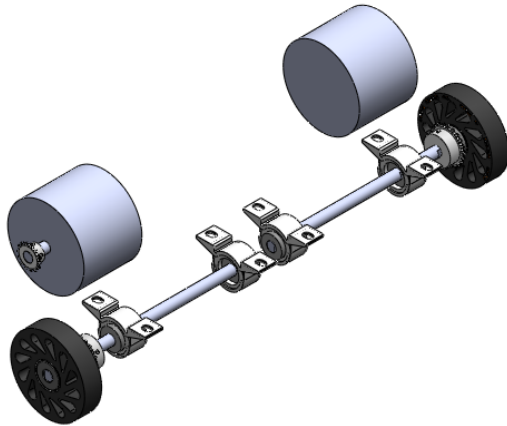
This model is the complete design for the DiWheel. It includes all aspects of the concept that was chosen from the morphological chart. That design was wheel to wheel contact, triangular chassis and housing, and a counterweight slider for dynamic and static leveling.



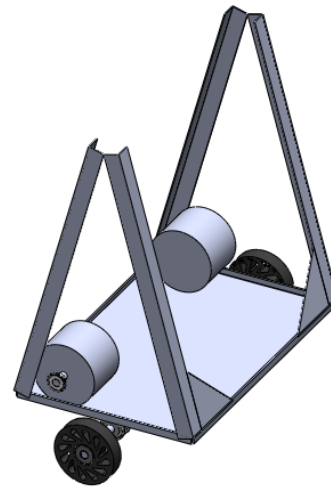
Full 3D Solidworks model.

Drive Train

This is a focused model on the drive train for the DiWheel. This is an essential subsystem that will provide the DiWheel with mechanical motion. The drive train includes the DC motor, drive gear, axel, bearings, and drive wheel.



Drive Train

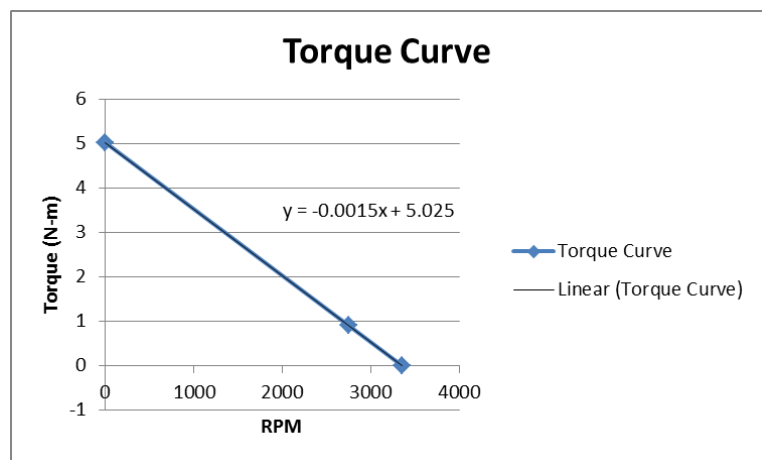


Drive Train on Chassis

The components of the drive train model shown above were selected after first finding what characteristics had to be met in order to meet our specifications as per Section 2.3. Below, we define the values determined in the specifications and calculated the power required from the motor. The power calculation values were derived from unit conversions, power output at different motor speeds, and ratios between wheel sizes and gear sizes. Our data and calculations are shown below.

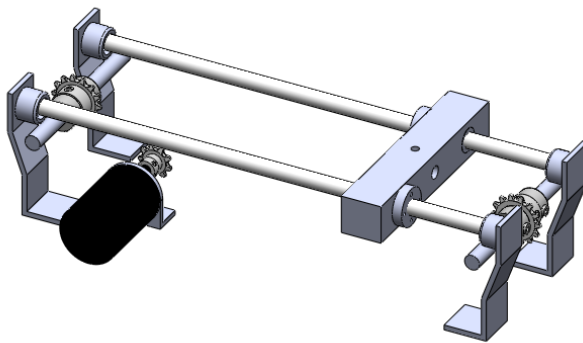
Power Calculations for the Drive Motors:

| | |
|-----------------------------------|-----------|
| top velocity (m/s) | 7.3000 |
| time to full speed (s) | 3.0000 |
| top acceleration (m/s^2) | 2.4333 |
| mass (kg) | 20.0000 |
| force (N) | 48.6667 |
| distance to top speed (m) | 10.9500 |
| power (W) | 177.6333 |
| power (hp) | 0.2382 |
| lin acc | 2.4333 |
| ang acc | 0.0618 |
| | |
| wheel diameter (m) | 0.6096 |
| drive wheel diameter (m) | 0.1016 |
| wheel circumfrance (m) | 1.9151 |
| drive wheel circumfrance (m) | 0.3192 |
| inside wheel diameter (m) | 0.5080 |
| inside wheel circumfrance (m) | 1.5959 |
| | |
| outter wheel to inner wheel | 1.2000 |
| | |
| wheel rpm | 228.7069 |
| | |
| inside wheel to drive wheel ratio | 5.0000 |
| | |
| drive wheel rpm | 1143.5345 |
| | |
| wheel torque (N*m) | 14.8336 |
| | |
| drive wheel torque (N*m) | 2.9667 |
| in*Ib | 26.2576 |
| | |
| motor drive ratio | 2.0000 |
| motor rpm | 2287.0691 |
| | |
| each drive wheel (N*m) | 1.4834 |

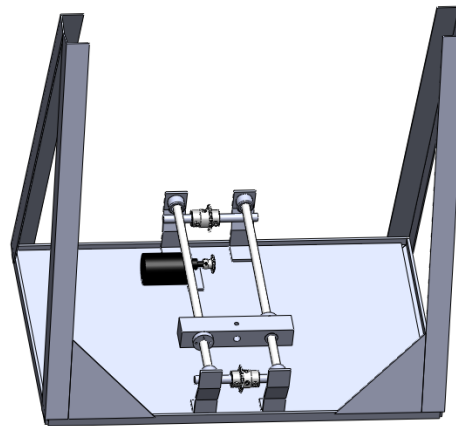


Dynamic and Static Leveling: Counter Weight

The model seen below is of the counter weight slider mechanism that will provide the Diwheel with static and dynamic leveling. The slider consists of two guide rails, the counterweight, and a drive motor. This motor is in charge of rapidly moving the weight along the rails. As the weight slides forward and back, it creates a torque on the chassis to counteract the torque created by the drive motor.



Leveling



Leveling on Chassis

The components of the counter weight slider model shown above were selected after first finding what characteristics had to be met in order to meet our specifications as per Section 2.3. Below, we define the values determined in the specifications and calculated the displacement needed by the slider to counter balance the torque of the motor. There are three dynamic equations defining our diwheel that were adapted from the Edwards Diwheel Project, Appendix C. The first equation is the sum of the torques on the chassis about the wheel axis. The second equation is the sum of the torques on the wheels about the wheel axis. The third equation is the sum of the torques about the vertical axis. The variables are defined in Appendix C.

Defining and solving Dynamic Equations:

Initial Equations

$$(1) \hat{J}_{b_{zz}} \ddot{\theta} - a_x \cos(\theta) \ddot{\phi} - \sin(\theta) \cos(\theta) Z \Omega^2 + a_g \sin(\theta) + 2b(\dot{\theta} - \dot{\phi}) - m_s g(l \cos(\theta) + r \sin(\theta)) + 2T_M = 0$$

$$(2) \left(2\hat{J}_{w_{zz}} + \frac{m_b R^2}{2}\right) \ddot{\phi} - a_x \cos(\theta) \ddot{\theta} + a_x \sin(\theta) \dot{\theta}^2 + 2b_0 \dot{\phi} + 2b(\dot{\phi} - \dot{\theta}) - 2T_M = 0$$

$$(3) \left(\frac{\hat{J}_{w_{zz}}}{2\alpha} + \hat{J}_{yy}\right) \dot{\Omega} + 2\alpha \sin(\theta) \cos(\theta) Z \dot{\theta} \Omega + \frac{b_0 \Omega}{2\alpha} + \frac{b \Omega}{2\alpha} - \left(\frac{R}{P_{w,z}}\right) \frac{2K_t N_m}{R_m} V_{m\Delta} + \frac{2K_t^2 N_m^2}{R_m} \Omega = 0$$

For our detailed design, we are only going to focus on the first equation, which has to do with the sum of the torques on the chassis about the wheel axis. We will ignore friction in the system and look at a case where there is not turning about the vertical axis. This will allow the equation to be simplified. Once simplified the displacement of the slider can be solved for.

Simplified

$$(1)$$

$$\hat{J}_{b_{zz}} \ddot{\theta} - a_x \cos(\theta) \ddot{\phi} - \sin(\theta) \cos(\theta) Z \Omega^2 + a_g \sin(\theta) + 2b(\dot{\theta} - \dot{\phi}) - m_s g(l \cos(\theta) + r \sin(\theta)) + 2T_M = 0$$

$$-m_b e R \cos(\theta) \ddot{\phi} + m_b e g \sin(\theta) - m_s g(l \cos(\theta) + r \sin(\theta)) + 2T_M = 0$$

$$m_s g l \cos(\theta) = 2T_M - m_b e R \cos(\theta) \ddot{\phi} - m_b e g \sin(\theta) - m_s g r \sin(\theta)$$

$$l = \frac{2T_M - m_b e R \cos(\theta) \ddot{\phi} - m_b e g \sin(\theta) - m_s g r \sin(\theta)}{m_s g \cos(\theta)}$$

After we solved for the displacement of the counter weight slider, we simplified again for two different cases. The first case is the displacement needed at constant maximum acceleration and the second is the displacement needed at constant maximum velocity. This is the displacement that the slider needs to move to keep the chassis at a given angle during maximum acceleration.

Displacement at Constant Max Acceleration

$$l_{acc} = \frac{2T_{Ma} - m_b e R \cos(\theta) \ddot{\phi} - m_b e g \sin(\theta) - m_s g r \sin(\theta)}{m_s g \cos(\theta)}$$

We then simplified the equation for the displacement needed at constant velocity. For this case, we set the angle of the chassis to zero simplifying to the following equation.

Displacement at Constant Max Velocity

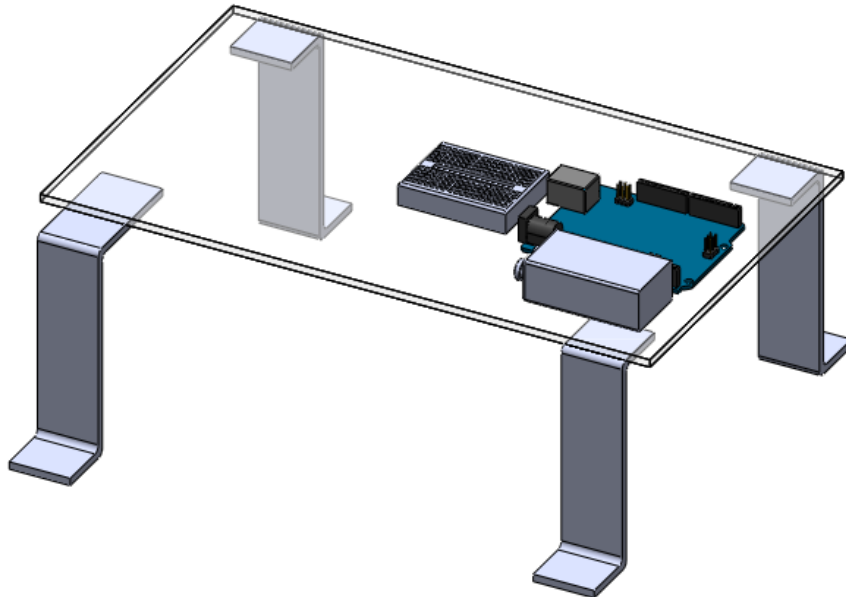
$$l_{vel} = \frac{2T_{Mv}}{m_s g}$$

Below we define the value of the variables used and show our calculated values for the displacement of the sliding counter weight.

Defining Variables and Solving for Displacement of Slider:

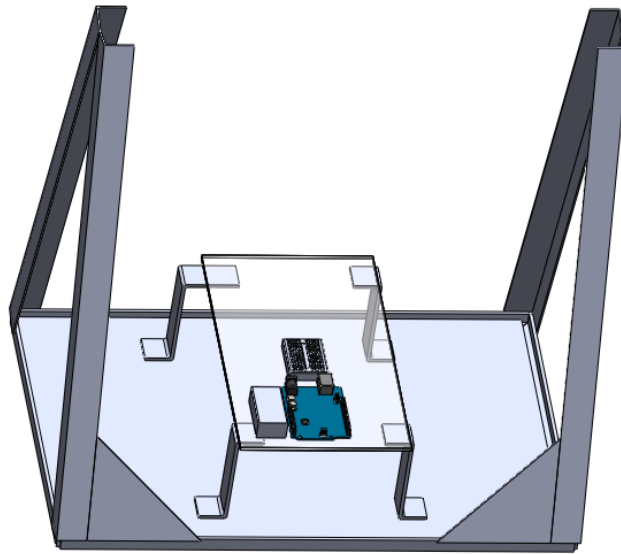
| Term | Variable | Value | Units |
|---------------------------------------------------------------------|------------------|--------------|--------------------|
| Mass of body | m_b | 30.000 | kg |
| Radius of Wheels | R | 0.305 | m |
| Radius of Mass | e | 0.127 | m |
| Max Pitch Allowed | Θ | 2.750 | deg |
| Max Pitch Allowed | Θ | 0.048 | rad |
| Max Torque | T_m | 7.480 | Nm |
| Max Displacement of Slider | l_max | 0.110 | m |
| Mass of Slider | m_s | 1.000 | kg |
| Max Linear Acceleration | a | 1.217 | m/s ² |
| Max Angular Acceleration | $\ddot{\phi}$ | 3.992 | rad/s ² |
| Radius of Slider | r | 0.095 | m |
| Intermediate Terms | Variable | Value | Units |
| Gravitational constant | a_g = m_b*e*g | 37.376 | |
| Cross Coupling term | a_x = m_b*e*R | 1.161 | |
| Final Calculations | Variable | Value | Units |
| Displacement of slider at max acceleration, 1.2166 m/s ² | l_acc | 0.103 | m |
| Displacement of slider at constant velocity, 7.3 m/s | l_vel | 0.022 | m |

Hardware Table/Electrical Components



Hardware Table with Electrical Components

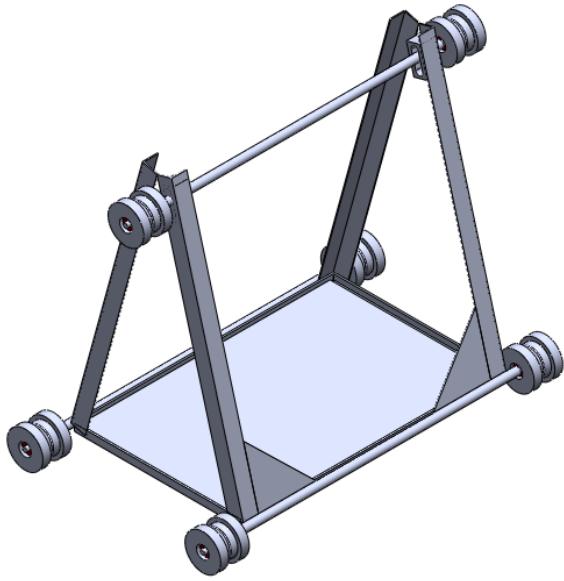
The hardware table was then modeled and will be able to hold all of the electrical components and the select user hardware. The electrical components will include the breadboard, Arduino Uno controller set up, and 9V battery. This part will be comprised of four legs and one platform that will all be printed on the 3D printer.



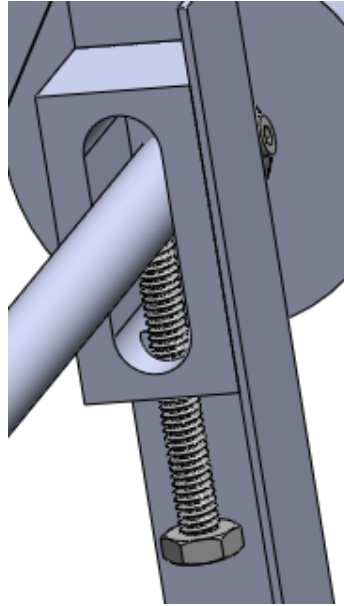
Hardware Table with Electrical Components on Chassis

Chassis and Guide Wheel

The chassis is shown modeled below. It is comprised of aluminum angle iron, plate, and bar and will form a triangular prism shape. At each of the three corners there will be a guide wheel to keep the bike wheel in line and in contact with the drive wheel. As the diwheel is used the guide wheels will begin to wear causing vibrations in the system so a tensioner will be added to the top guide wheel to account for this wearing.



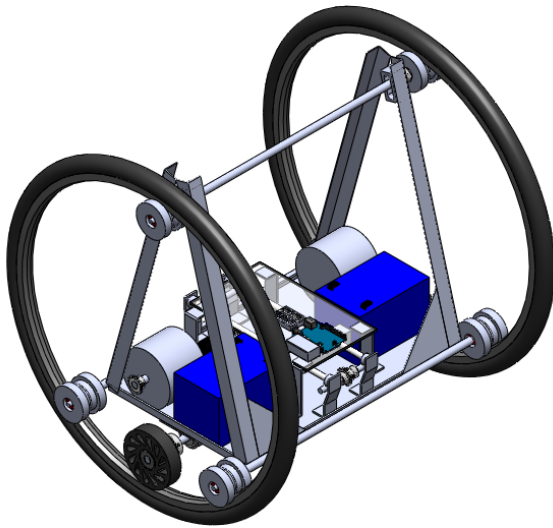
Chassis



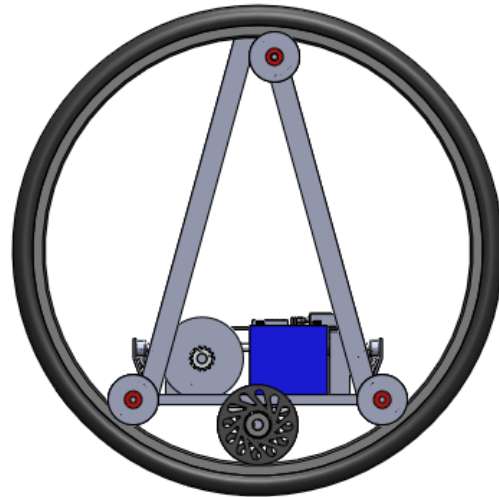
Tensioner

Full Exploded View

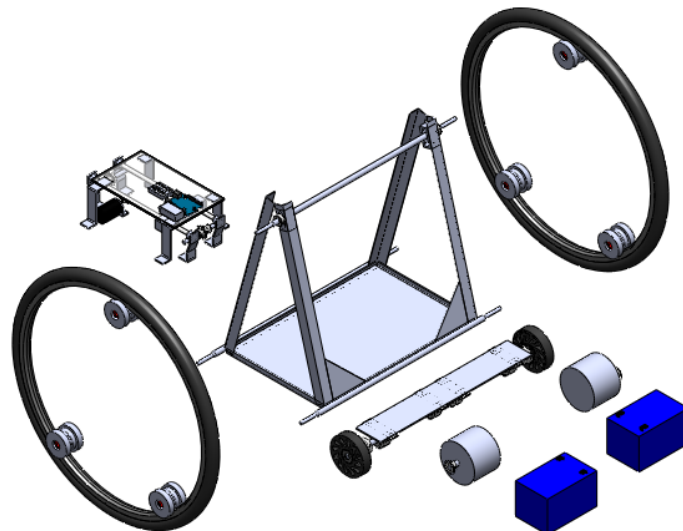
Shown below is the full Solidworks model of the DiWheel from different views. You can see how each subsystem interacts with each other.



Isometric View



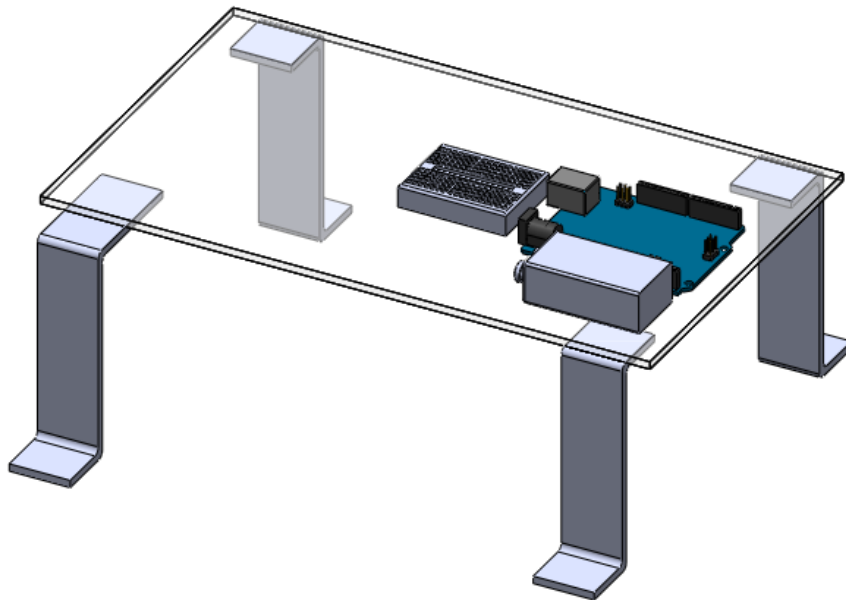
Side View



Exploded View

4.4 Electrical Detailed Design

Detailed design is one of the most important aspects of the design process as it describes how the design will be created. This section is divided up into two sections; one for hardware design, and the other for software design. The hardware design section describes some of the physical connections that will be made between components. The hardware section will also give an overview of the electrical components and how each fits into the design. The software portion describes the logic flow of the programming and gives detail to what algorithms will be implemented.



4.4.1 Hardware Design

This project will not require extensive designing of hardware, but will utilize various components together. The hardware will revolve around the Arduino Microcontroller as it will be handling all inputs and outputs. The various components will be brought together to create the functionality the diwheel requires. This section will describe some of the connections that will be made between components and how they will interact with each other.

The Wireless Inventors Shield for Arduino is directly attached to the Arduino Uno's I/O ports. The shield will utilize some of the Arduino's pins while leaving the others unattached. Because the shield sits on top of the Arduino itself, headers are used to connect and extend the I/O pins of the Arduino. The circuit schematic of this shield in relation to the Arduino's I/O pins can be found in Figure 4.4.1.

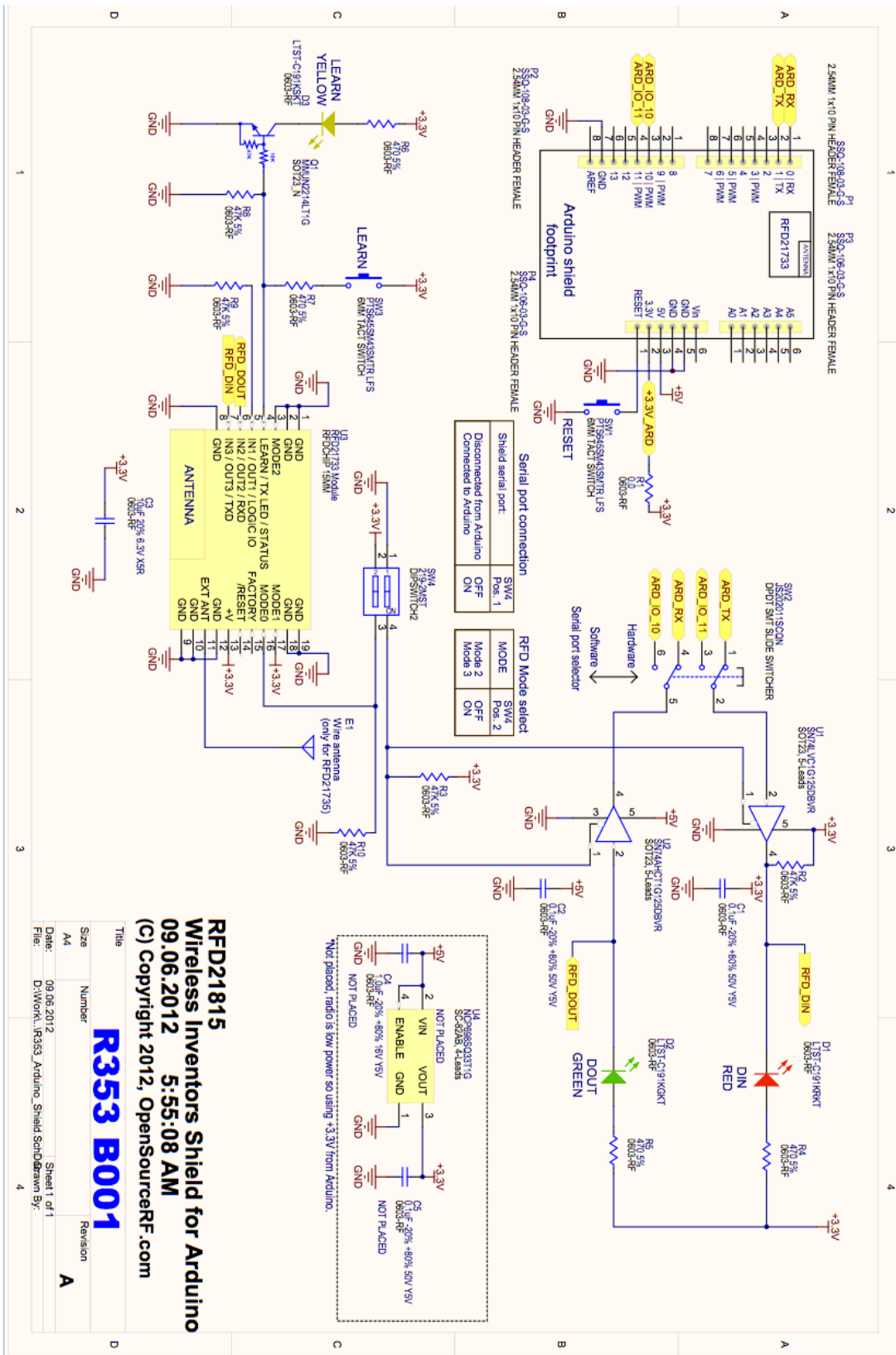


Figure 4.4.1: Wireless Inventors Shield for Arduino Circuit Schematic

The Wireless RF USB Dongle utilizes the RFD21733 transmitter within the hardware. The circuit schematic for the RFD21733 transmitter can be found in Figure 4.4.2.

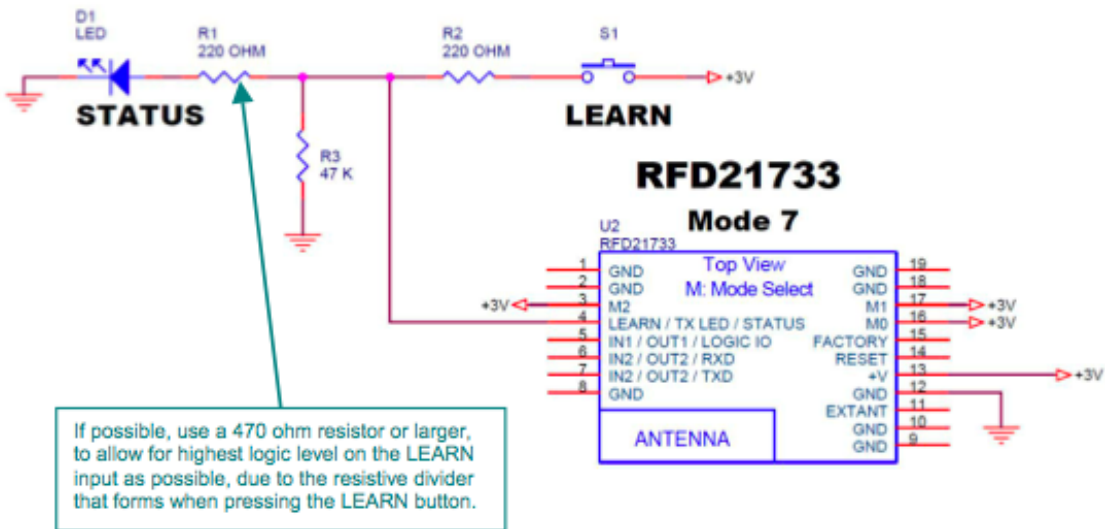


Figure 4.4.2: RFD21733 Transmitter Circuit Schematic

<http://www.opensourcerf.com/common/RFD8.RF.Modules.Manual.pdf>

Our design will utilize the breakout board for the IMU sensor. This breakout board will allow us to easily access the needed terminals in our design. The circuit diagram for this interface can be found below in Figure 4.4.3.

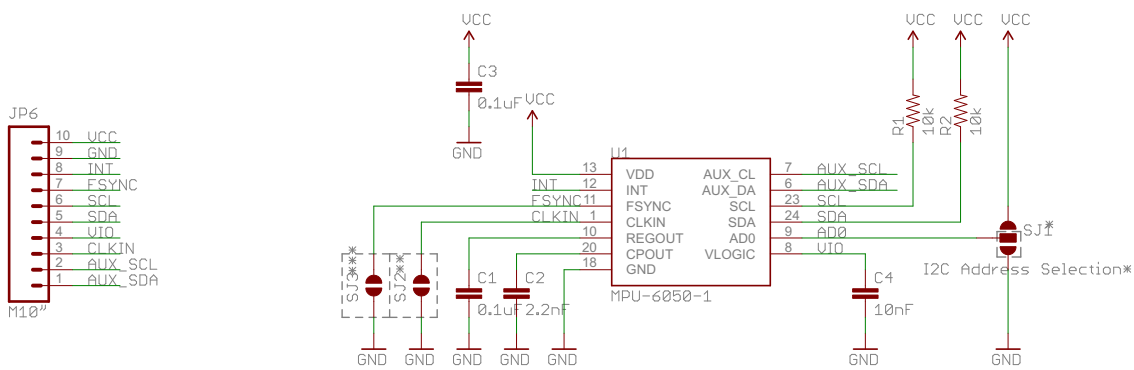


Figure 4.4.3: MPU-6050 Breakout Board Circuit Diagram

http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/IMU/MPU-6050_Breakout%20V11.pdf

An overall picture of the hardware used and connections can be found below in Figure 4.4.4. The completed design will also contain certain safety precautions such as fuses and switched to turn the system on and off easily. The complete circuit diagram for the hardware can be found in Figure 4.4.5. Table 4.4.1 lists the pins and connections of the digital side of the design.

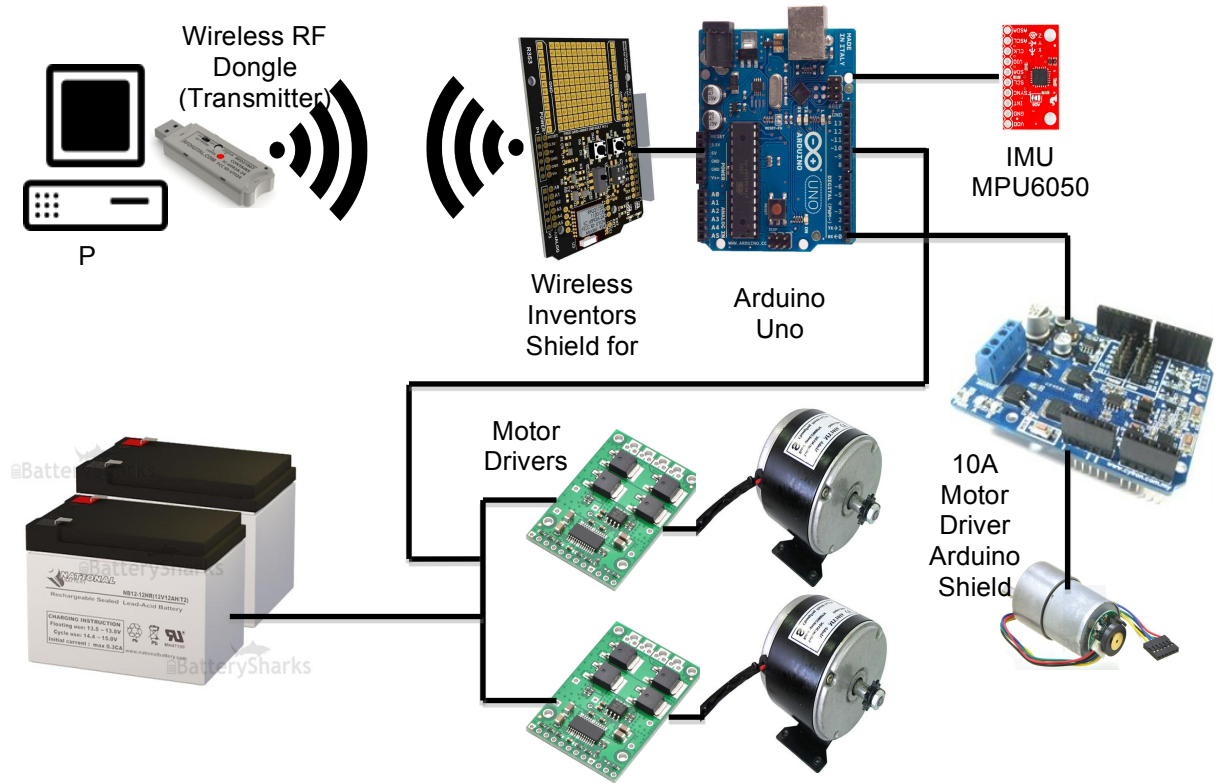


Figure 4.4.4: Abstract Electrical Connections

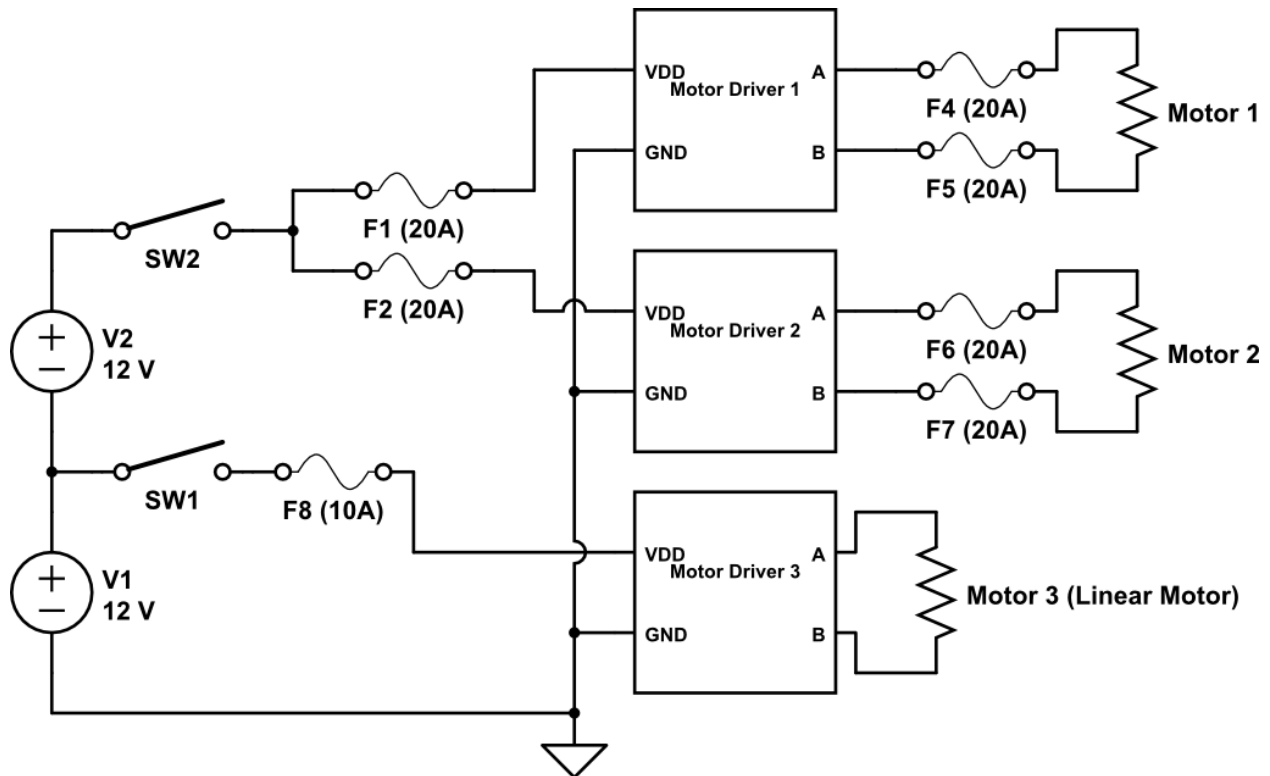


Figure 4.4.5: Diwheel Circuit Diagram

| Arduino Pin | Connection | Wire Color | Function |
|-------------|-----------------|----------------------|-----------------------------|
| 0 | Wireless Shield | No Wire (Header) | RX (Receive Serial Data) |
| 1 | Wireless Shield | No Wire (Header) | TX (Transmit Serial Data) |
| 2 | LM EncA | Yellow | Encoder output from LM |
| ~3 | LM PWM | No Wire (Header/JD3) | PWM Signal of LM |
| 4 | - | - | - |
| ~5 | LM EncB | White | Encoder Output From LM |
| ~6 | M1 Direction | Green | Controls direction of motor |
| 7 | M2 Direction | Green | Controls direction of motor |
| 8 | M1 RESET | White | Resets motor driver 1 |
| 9 | M2 RESET | White | Resets motor driver 2 |
| ~10 | M1 PWMH | Yellow | PWM Signal of M1 |
| ~11 | M2 PWMH | Yellow | PWM Signal of M2 |
| 12 | LM Direction | No Wire(Header/JD12) | Controls direction of motor |
| 13 | - | - | - |

Table 4.4.1: Diwheel System Digital Side Pin Mapping

- : No Connection
 ~ : Indicates a PWM capable pin
 LM: Linear Motor
 M1: Motor 1 (Left Motor)
 M2: Motor 2 (Right Motor)

4.4.2 Software Design

The diwheel's chassis is required to be level in our design. In order to keep the chassis level, a control algorithm is needed. This algorithm will monitor the current movement and adjust the chassis accordingly. The microcontroller will house a program that will implement a PID controller to accomplish this. The basic structure of this algorithm can be found below in Figure 4.4.6. When the device is powered on, the system will first initialize all inputs to the default/starting values. The control loop will then start and read the data from the IMU sensor as well as the user's commands. Once the data is read, it will be processed and the output voltage for the DC motors will be adjusted to correct any sloshing that will occur. This will keep the chassis level.

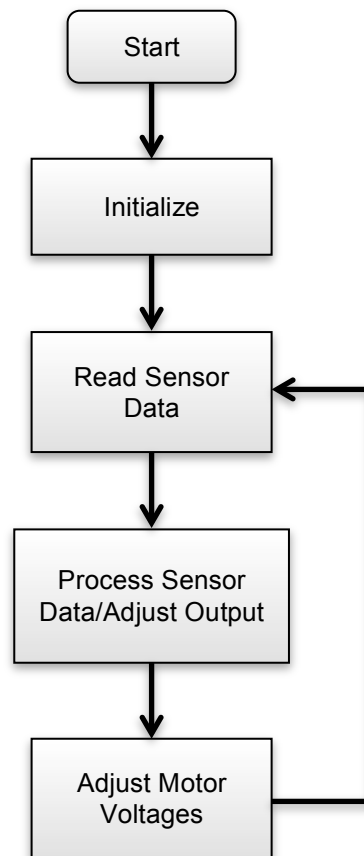


Figure 4.4.6: Control Loop

A PID Controller is the most common feedback controller and will be the most effective for our diwheel's leveling design. A PID controller feeds back information to be taken away from the desired state. When the diwheel accelerates or decelerates, the chassis will want to rotate forward and backward. The PID controller will take the data from the IMU sensor and use it to create the error signal. When the diwheel goes past the desired degrees of freedom, feedback will be send to the input to correct the “slosh” of the device. A typical feedback loop can be found in Figure 4.4.7. Our PID controller will be implemented in the software on the Arduino Microcontroller.

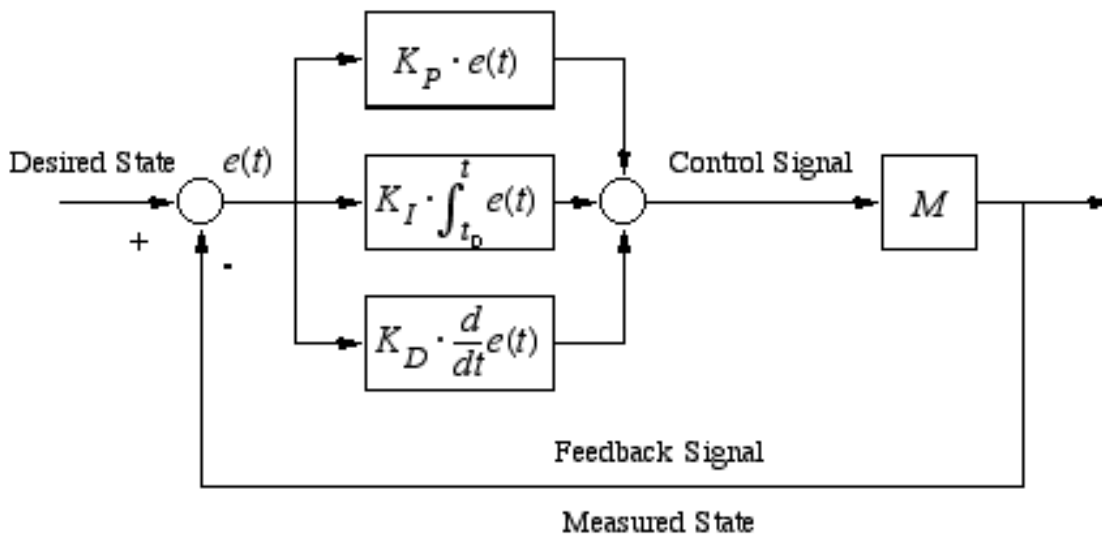


Figure 4.4.7: PID Controller

4.4.3 Diwheel Controls

The diwheel will be controlled by the user through serial communication. This can be done in either HyperTerminal on Windows, or the Arduino's Serial Monitor tool on all operating systems. The device is designed to operate at a baud rate of 9600. The program has been written with specific keys mapped for specific functions. The commands/controls can be found in Table 4.4.2.

| Key | Command |
|------------------------------|-------------------------------|
| w | Forward |
| a | Left |
| s | Backward |
| d | Right |
| q | Spin counter-clockwise |
| e | Spin clockwise |
| x | Stop (Coast to a stop safely) |
| v | Stop (Emergency Stop – Brake) |
| g | Turn on system (On) |
| r | Reset system (Off) |
| 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 | Speed Control (Low to High) |

Table 4.4.2: Diwheel Controls

The diwheel is controlled via the onboard program within the Arduino. This code is written in the Arduino language which is C based programming language. The program has been written with several safety precautions in mind such as protection against changing a motor's direction too suddenly and ramping the speed down before completely stopping. The program ("sketch") can be found in Appendix C.

5. Milestone 5: Prototyping

Before the complete system could be assembled and tested, prototyping had to be completed. Prototyping involved establishing a basis for which the final product would be built upon. Each component was prototyped and then the system was prototyped. Components were assembled, constructed, wired, soldered, and built in order to meet the design specifications.

5.1 Mechanical Prototyping

Mechanical prototyping consisted of the manufacturing process for all the mechanical components. In this section, each component is listed and a description of how they were manufactured, why that design was chosen, and a picture of the finished product.

Component: Chassis

Functionality: Component Mount

The chassis is where the motors, batteries, leveling components and electrical components are mounted and where the axels to the guide wheels and drive train are attached. Angle iron was cut and welded at the corners in order to build the frame of chassis platform. Once the frame was completed, the aluminum plate was cut to fit.



Figure 1: Cutting Angle Iron



Figure 2: Welding Steel Frame



Figure 3: Corner Weld



Figure 4: Finished Frame

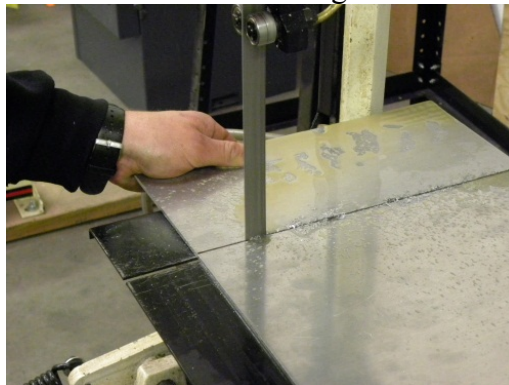


Figure 5: Cutting Aluminum Plate

Component: Tensioner**Functionality: Keeping Top Guide Wheel Axel in Tension**

The tensioner was made out of 1in x 1in steel stock to be 3 inches long. It was cut to length on the band saw and then a $\frac{1}{2}$ in slot was milled out using the manual mill.



Figure 6: Milled Tensioner

Component: A-Frame**Functionality: Providing Structure For Top Guide Axel**

Four lengths of 1in x 1in x 1/16in angle iron were cut to create the four arms of the A-frame. Two 4in x 4in right triangles were cut from 1/16in thick sheet to later be welded to support the A-frame from leaning.

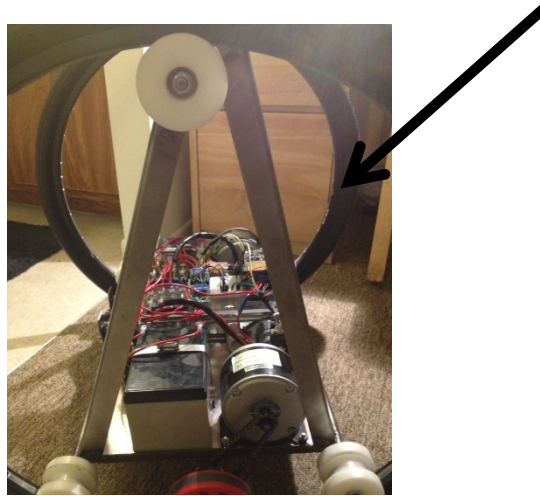


Figure 7: A-Frame

Component: Guide Wheels

Functionality: Wheel Attachment

There are a total of six guide wheels, three on each wheel. Grooves in the middle of hard plastic cylinders were milled using the lathe. With all six wheels milled, the three axels for them were made. Using $\frac{1}{2}$ in diameter steel rod, three rods were cut to 24 in each. Each end of the rod was taken down using the lathe to be $\frac{5}{8}$ in diameter to create a shoulder for the guide wheels to sit upon. The ends of the rods were threaded and a nut was used to secure the guide wheels to the rod.



Figure 8: Lathe Groove in Guide Wheel



Figure 9: Axle Shoulder and Thread



Figure 10: Lathing Guide Wheel Axels



Figure 11: Guide Wheel

Component: Drive Plate**Functionality: Plate to Attach Drive Axel System to**

A plate was cut from 1/8in x 3in sheet to the same width as the frame. This plate is where the drive axel assembly will be mounted to and this plate will be welded to the bottom of the frame. This plate will allow the drive axel to be mounted to the frame and not the aluminum plate.

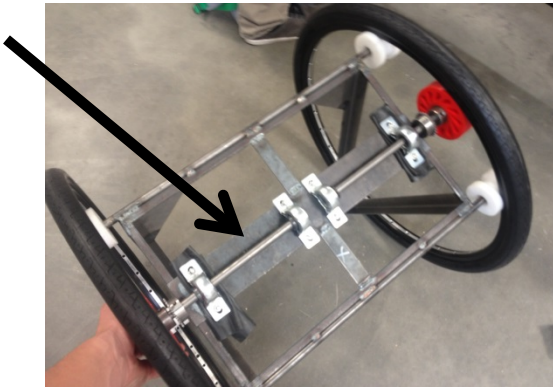


Figure 12 & 13: Drive Plate

Component: Supporting Strap**Functionality: Provide Support for the Middle of the Drive Plate**

The supporting strap was cut from a 1 in x 1/8in strap at a length that fit tight between the front and back of the frame. This strap was welded on to the frame and then welded to the drive plate to support the middle of drive plate. This support was necessary so that the drive plate did not deflect too much as the drive wheel axel was tensioned.



Figure 14 & 15: Supporting Strap

Component: Leveling Weight

Functionality: Leveling

The sliding counter weight was created from steel. It was cut five inches in length from a 1in x 1in stock and two 5/8 in holes were drilled; holes were drilled beginning with a 1/4 bit, moving to a 1/2 bit, and then finishing with a 5/8 bit. Attaching the sliding weight to two bars was done by press fitting two brass sleeve bearings into the drilled holes. These bearings were slightly too large so to make sure they would fit properly, the outside was sanded down before being press fitted in. Lastly, the motor mount for the leveling motor was machined.



Figure 16: Cutting 5" Weight

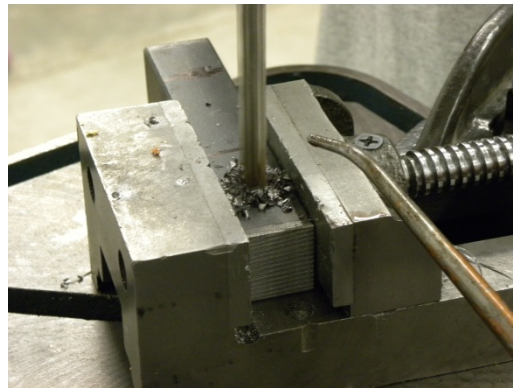


Figure 17: Drilling with 1/4 Bit



Figure 18: Drilling with 5/8 Bit



Figure 19: Finished Weight with Bearings



Figure 20: Weight on Rails



Figure 21: Leveling Motor Mount

Component: Leveling Rod Mounts**Functionality: Hold Leveling Rods in Place**

The Leveling Rod Mounts are made from 1 in x 1/8in strap. They have a ninety degree bend at the bottom for a bolt to secure it to the aluminum plate. The vertical portion then has a slight deviation to make them slightly wider to fit the leveling rods. At the top a small steel cylinder was welded to hold the leveling rods in place.



Figure 22: Leveling System Rod Mounts

Component: Leveling Idle Axel**Functionality: Hold Idle Gear to Leveling Rod Mounts**

This axel was made from a 3/8 steel rod. The process of making the part included drilling and machining.

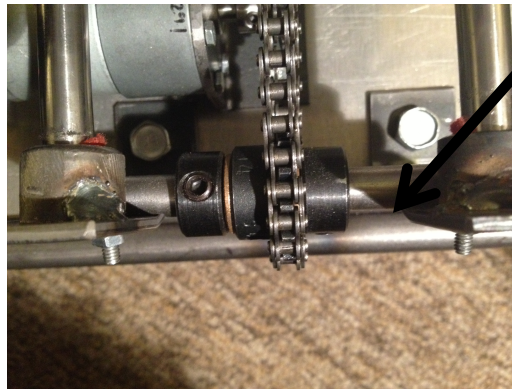


Figure 23: Leveling Idle Axel

Component: Leveling Chain Tensioner**Functionality: Tension Chain and Maintain Contact with Leveling Motor**

The leveling chain tensioner was made from 3/4in PVC pipe held down by two 3/4in conduit clamps. This tensioner was placed close to the leveling motor so that the chain would maintain a high angle of contact on the gear on the motor. The tension can be adjusted by tightening the bolts.



Figure 24: Leveling System Chain Tensioner

Component: Hardware Table**Functionality: Table to Mount the Electrical and External Hardware**

A tabling using plexiglass and steel was built. The legs were made by bending steel straps at 90 degree angles on either end. Holes were drilled through the plexiglass and legs in order to bolt the two together. Also, holes in the plate and bottom of the table legs were drilled to be bolted.

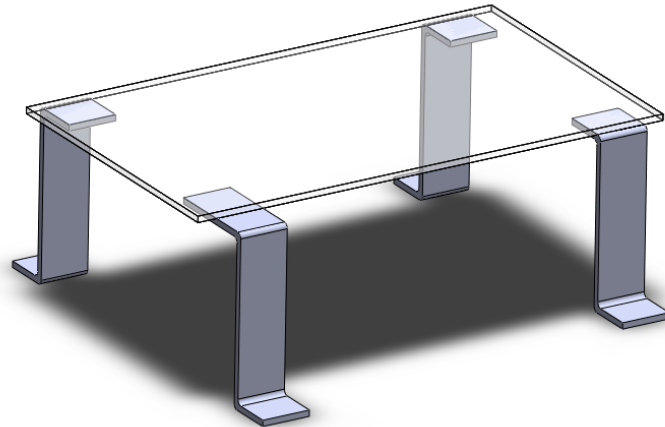


Figure 25: CAD of Hardware Table

Systems Prototyping: Frame and Guide Wheels

The guide wheel axels were welded to the side of the frame of the chassis. This design will help keep everything aligned as well as provide structural support for the chassis. The next step was to weld each tensioner to an arm of the A-Frame. The arms of the A-frame were then placed on the frame and tack welded on. They were checked with a right angle to ensure that they were vertical. Next the support triangles were welded on the A-frame arms and the frame to keep the arms vertical. The drive axel assembly was then placed on the drive plate and the holes were drilled to place the bolts. The plate was then in the middle below the left and right angles. The support strap was then placed between the front and back angles of the frame and welded to the frame and drive plate.



Figure 26: Guide wheels axels welded to frame

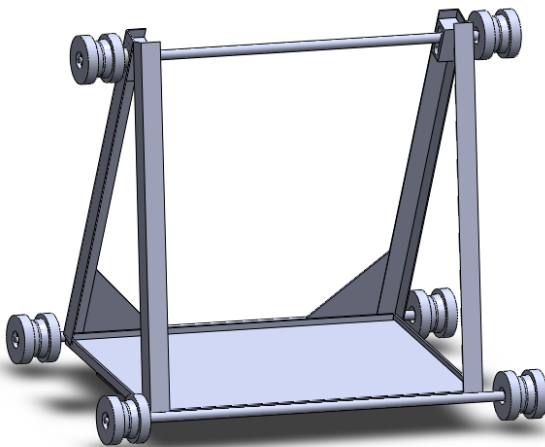


Figure 27: CAD of Chassis



Figure 28: Support Straps

Systems Prototyping: Drive Train Assembly

The location of the motors on the frame was determined and they were bolted down. Next, the location of the drive axel was determined and the bearings were bolted to the frame. Once these two key components were in place, the chains were attached via gears. In order to properly tension the chains, adjustments to the bearings were made by tightening/loosening bolts and adding rubber bike tubes as “spacers”. Securing the two batteries was done by strapping them to the plate.

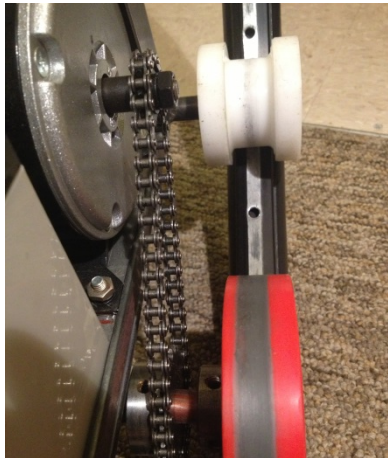


Figure 29: Chain and Gears



Figure 30: Battery Strap



Figure 31: Bearings with Adjustments

Systems Prototyping: Linear Leveling Assembly

The linear leveling system required that the four rod mounts line up properly so that the mass would be able to slide affectively. All four mounts were bolted to the middle of the frame of the chassis. Next, the leveling drive train, consisting of the motor, gear, chain, and chain tensioner, was assembled.

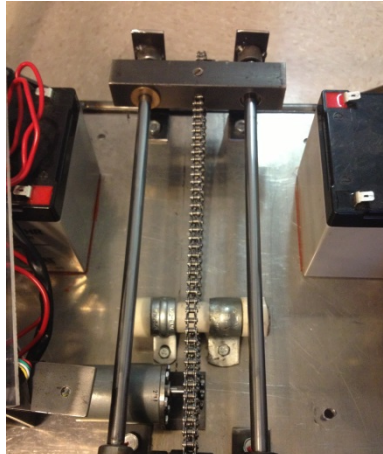


Figure 32: Linear Leveling Assembly

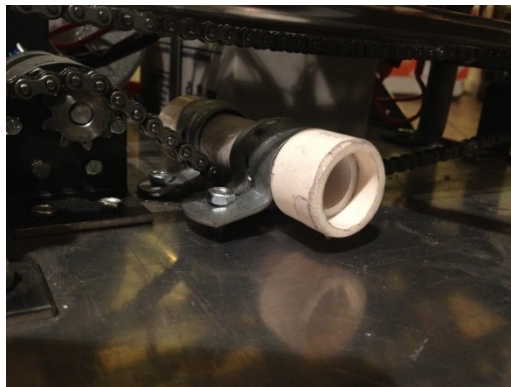


Figure 33: Linear Motor and Chain Tensioner

Full Mechanical Prototype

Once the sub systems of prototyping were complete, the full mechanical prototype was assembled. Everything was bolted down prepared so that the electrical portion could be easily integrated.

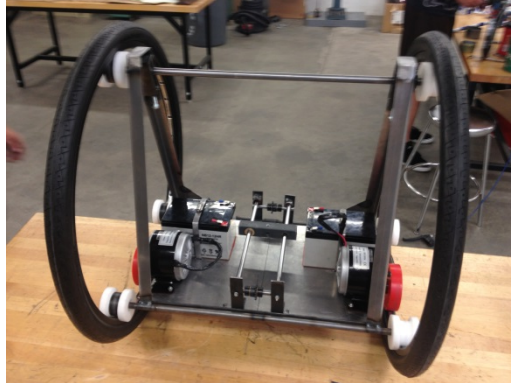


Figure 34: Full Mechanical Prototype

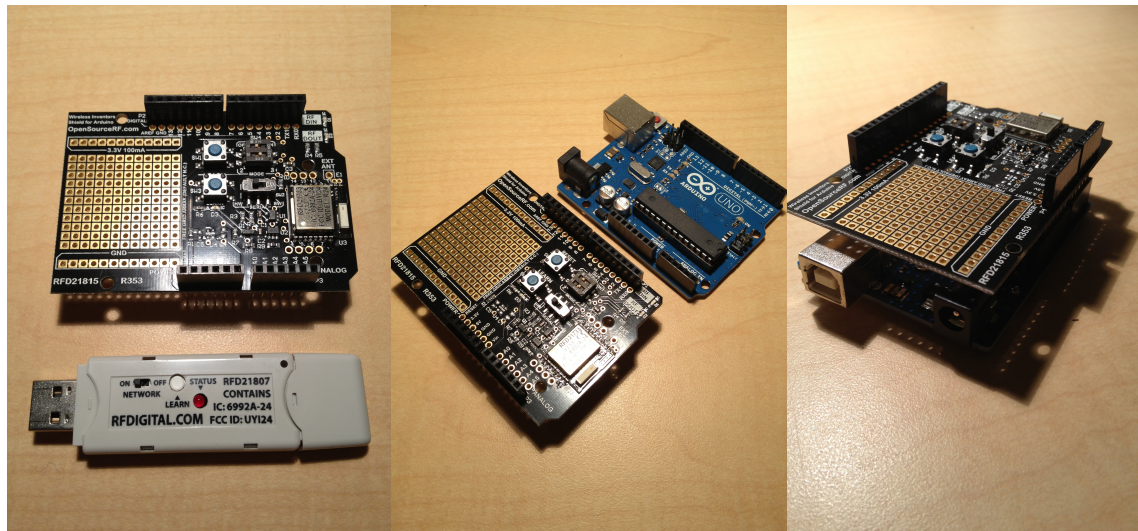
5.2 Electrical Prototyping

Electrical prototyping involved setting up the electrical components. This involved syncing the wireless communications, soldering wires, wiring the circuits, and mapping the digital pins. The Arduino code also had to be written during this time in order to set up certain components.

Component: Wireless Inventors Shield/Dongle

Functionality: Wireless Communication

The first step in the electrical portion of prototyping was to establish the wireless communication of the device. In order to do this, the Wireless Inventors Shield needed to be synced with the Wireless USB Dongle. The Wireless Inventors Shield was attached to the Arduino for power, and the Dongle was plugged into a laptop. Each device has a “learn” button which allows the two wireless chips to be able to communication with each as well as ignore all other signals. Once paired, the communication had to be tested. Using HyperTerminal on the laptop, serial data was sent over wirelessly to the Arduino. The test was successful and was shown by the RX LED on the Arduino lighting up when a character was received.





```
software_serial_loop | Arduino 1.0.1
software_serial_loop
//Wireless Inventors Shield Test (software)

#include <SoftwareSerial.h>

SoftwareSerial mySerial(10,11);

int incomingByte = 0;

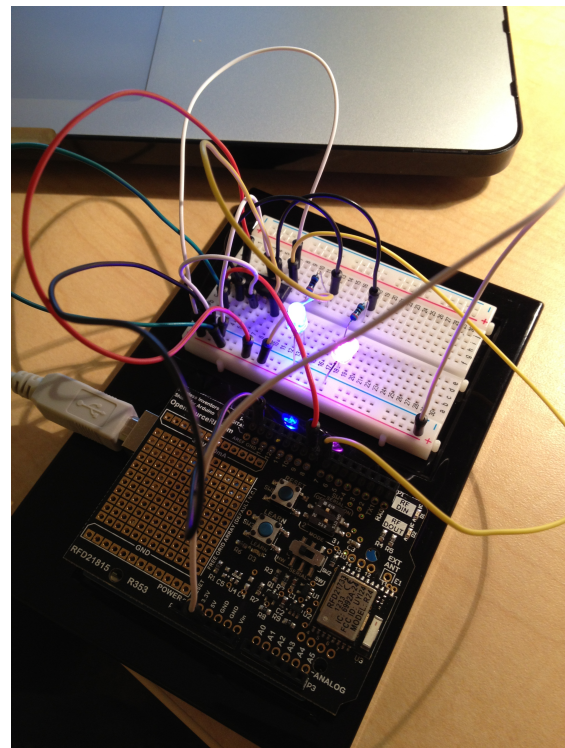
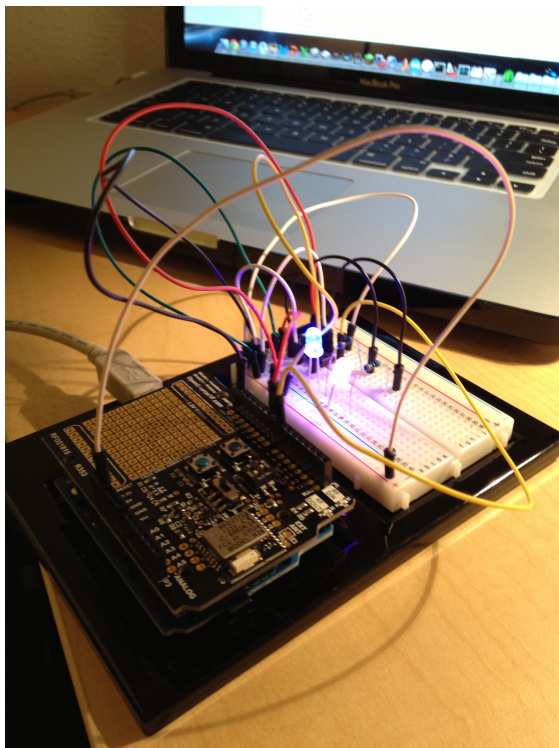
void setup()
{
  mySerial.begin(9600);
}

void loop()
{
  if (mySerial.available() > 0) |
  {
    incomingByte = mySerial.read();
    mySerial.print("I received: ");
    mySerial.write(incomingByte);
    //mySerial.print("/r/n");
  }
}

16 Arduino Uno on /dev/cu.usbmodem621
```

Component: H-Bridge Motor Driver 1A**Functionality: Motor Control**

The next step in the electrical prototyping was to learn and set up the H-Bridge. The H-bridge is what governs the motors as well as the power source. The H-Bridge IC chip was placed on a breadboard and wired according based on the datasheet. Because the motors had not yet arrived when this prototyping began, LEDs were used to simulate the direction of the motors and test the logic of the program. Two different colored LEDs were used and connected to the H-Bridge in order to simulate how the motors would be set up.





```
hardware_serial_loop | Arduino 1.0.1
hardware_serial_loop $

int incomingByte = 0;
int brightLED = 0;
int outputValue = 0;
const int analogOutPin = 2;

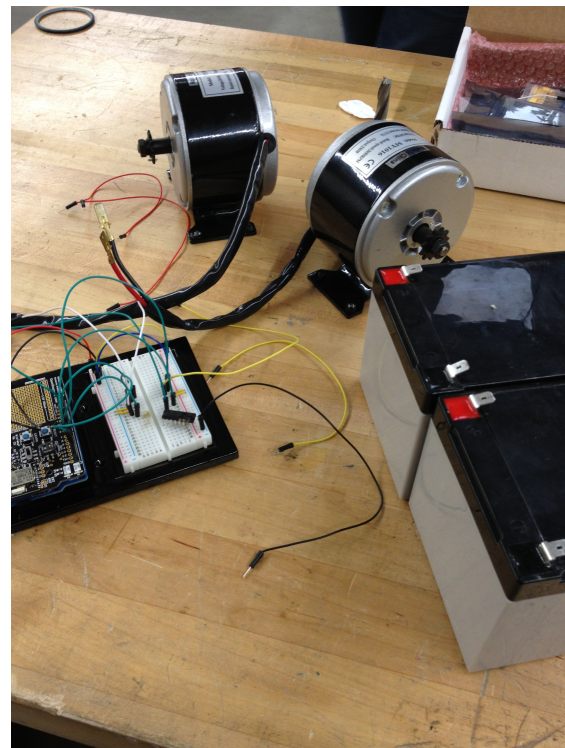
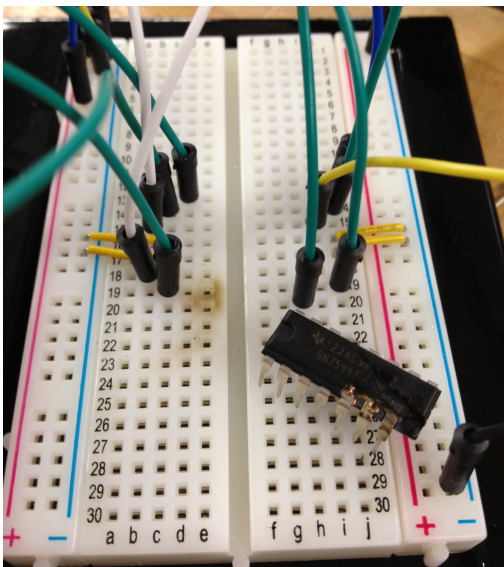
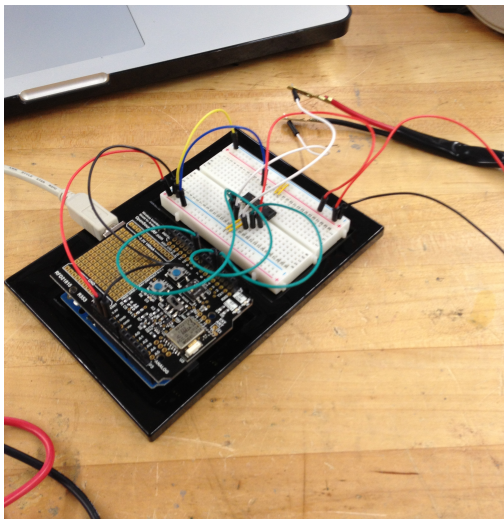
void setup()
{
  Serial.begin(9600);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
}

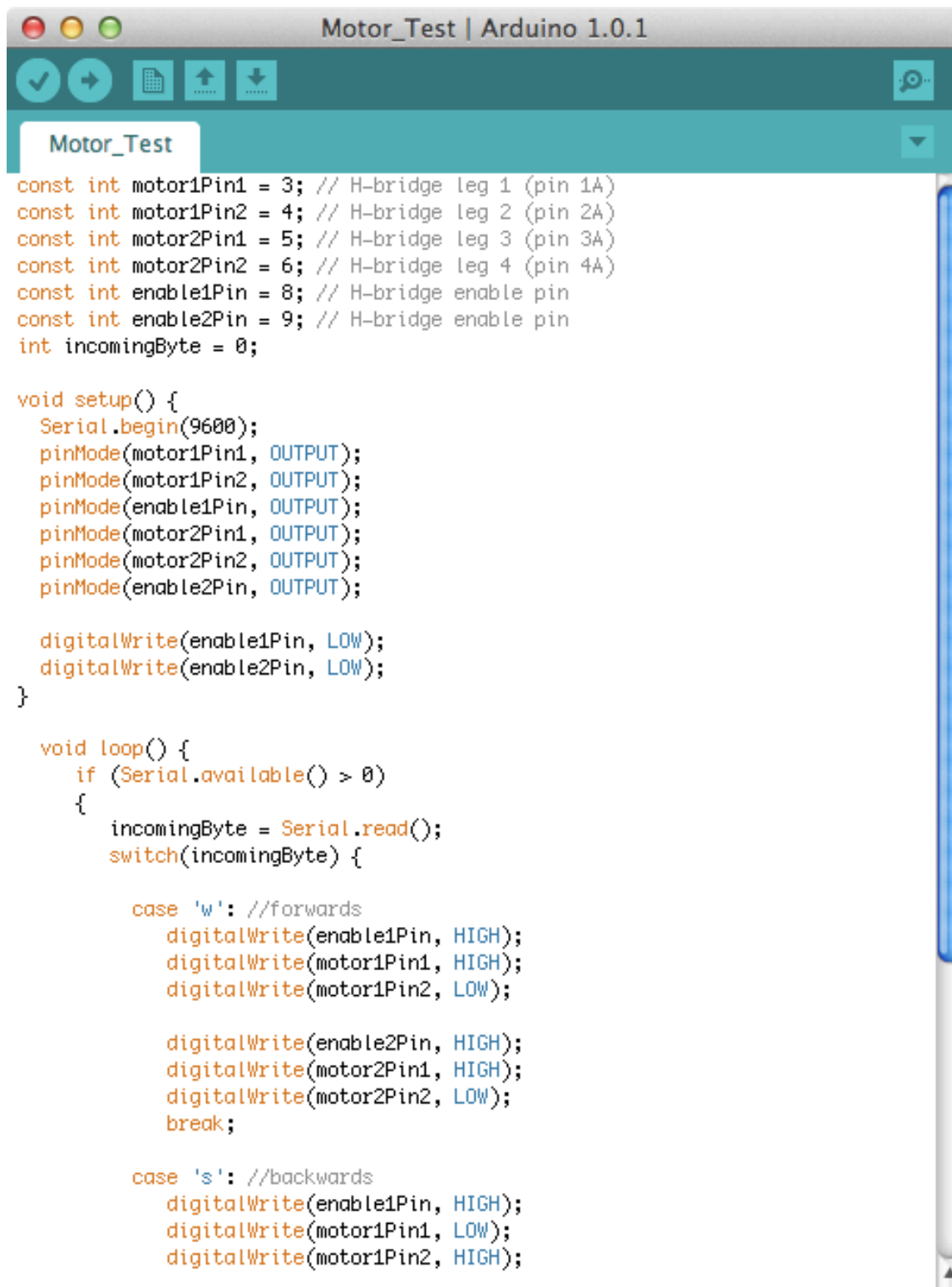
void loop()
{
  if (Serial.available() > 0)
  {
    incomingByte = Serial.read();
    switch(incomingByte) {
      case 'w':
        digitalWrite(7, HIGH);
        digitalWrite(8, HIGH);
        break;
      case 's':
        digitalWrite(7, HIGH);
        digitalWrite(8, HIGH);
        break;
      case 'a':
        digitalWrite(7, LOW);
        digitalWrite(8, HIGH);
        break;
      case 'd':
        digitalWrite(7, HIGH);
        digitalWrite(8, LOW);
        break;
      case 'x':
        digitalWrite(7, LOW);
        digitalWrite(8, LOW);
        break;
      default:
        break;
    }
  }
}
```

19 Arduino Uno on /dev/cu.usbmodem621

Component: 24V DC Motor**Functionality: Drives the Drive Wheels for Movement**

Once the motors and batteries were received, the next part in the prototyping process was to swap out the LEDs in the preliminary testing circuit, with the actual DC motors. Prototyping was first done on one motor with a single 12V battery. Once this was successful, the second motor was added. However, when the second motor was added, there was not enough power to drive both motors. In order to add power, a second 12V battery was added in series to obtain 24V. This was too much power for what the H-Bridge was designed for and the IC chip failed. As a result, more research was done on obtaining a motor driver suited for high power applications.





```
Motor_Test | Arduino 1.0.1
Motor_Test
const int motor1Pin1 = 3; // H-bridge leg 1 (pin 1A)
const int motor1Pin2 = 4; // H-bridge leg 2 (pin 2A)
const int motor2Pin1 = 5; // H-bridge leg 3 (pin 3A)
const int motor2Pin2 = 6; // H-bridge leg 4 (pin 4A)
const int enable1Pin = 8; // H-bridge enable pin
const int enable2Pin = 9; // H-bridge enable pin
int incomingByte = 0;

void setup() {
  Serial.begin(9600);
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);
  pinMode(motor2Pin1, OUTPUT);
  pinMode(motor2Pin2, OUTPUT);
  pinMode(enable2Pin, OUTPUT);

  digitalWrite(enable1Pin, LOW);
  digitalWrite(enable2Pin, LOW);
}

void loop() {
  if (Serial.available() > 0)
  {
    incomingByte = Serial.read();
    switch(incomingByte) {

      case 'w': //forwards
        digitalWrite(enable1Pin, HIGH);
        digitalWrite(motor1Pin1, HIGH);
        digitalWrite(motor1Pin2, LOW);

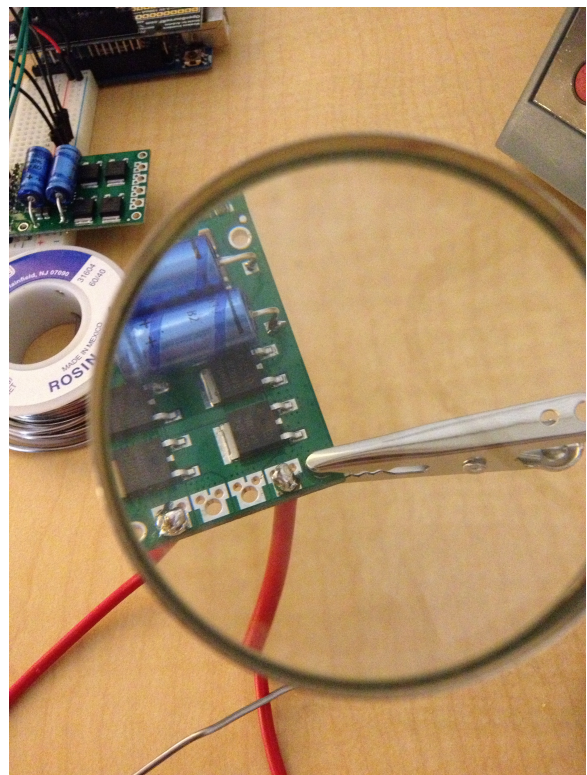
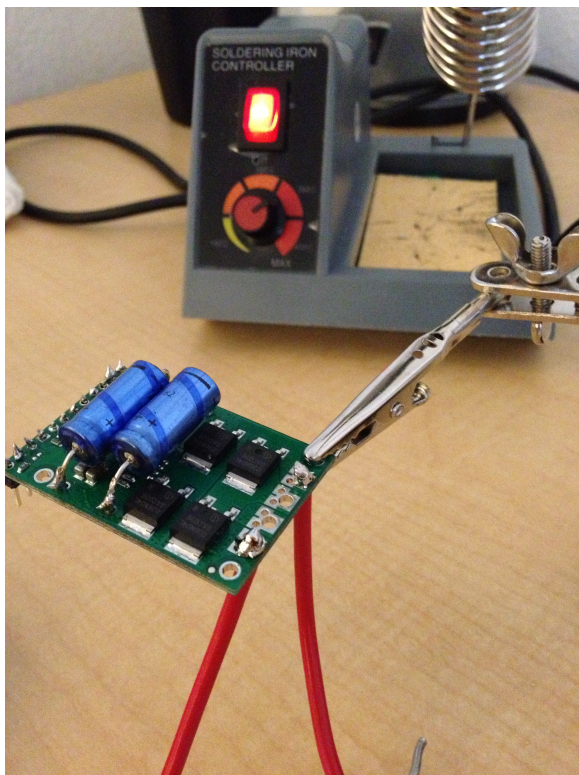
        digitalWrite(enable2Pin, HIGH);
        digitalWrite(motor2Pin1, HIGH);
        digitalWrite(motor2Pin2, LOW);
        break;

      case 's': //backwards
        digitalWrite(enable1Pin, HIGH);
        digitalWrite(motor1Pin1, LOW);
        digitalWrite(motor1Pin2, HIGH);
```

```
digitalWrite(enable2Pin, HIGH);  
digitalWrite(motor2Pin1, LOW);  
digitalWrite(motor2Pin2, HIGH);  
break;  
  
case 'q': //quit  
digitalWrite(enable1Pin, LOW);  
digitalWrite(enable2Pin, LOW);  
digitalWrite(motor1Pin1, LOW);  
digitalWrite(motor1Pin2, LOW);  
digitalWrite(motor2Pin1, LOW);  
digitalWrite(motor2Pin2, LOW);  
break;  
}  
}  
}
```

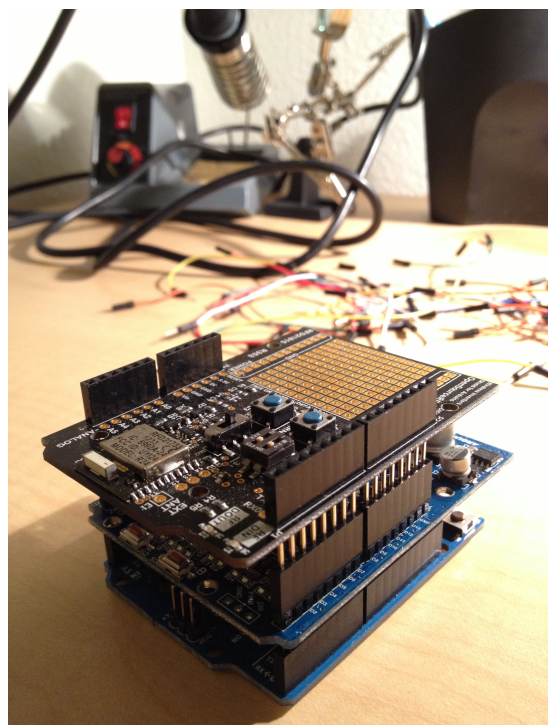
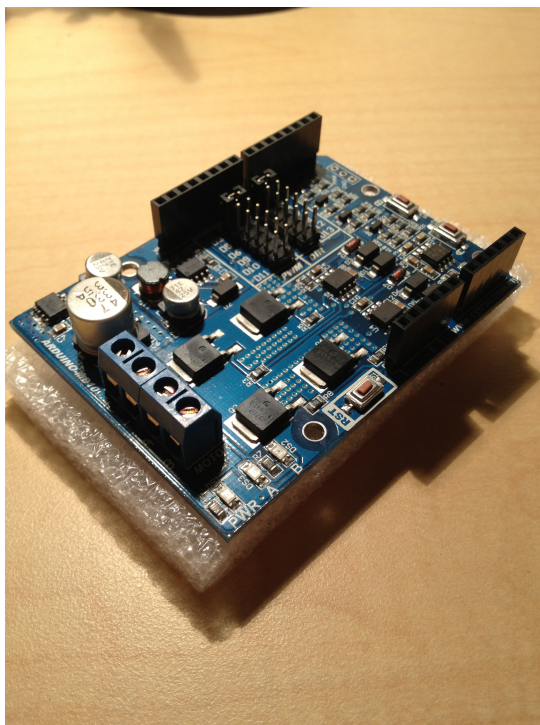
Component: Pololu High-Power Motor Driver 24v23 CS**Functionality: Motor Control**

After the failure of the IC H-Bridge, this motor driver was selected to take its place. Two drivers are required because each can support only one motor. Prototyping began with soldering on the capacitors and headers to the board. Wires were then soldered onto the power side of the board for the connection to the batteries and motors. Before soldering was completed, the motor driver was first tested in the lab. Once the drivers had been successfully tested, the final soldering was done and the component was brought into the completed system.



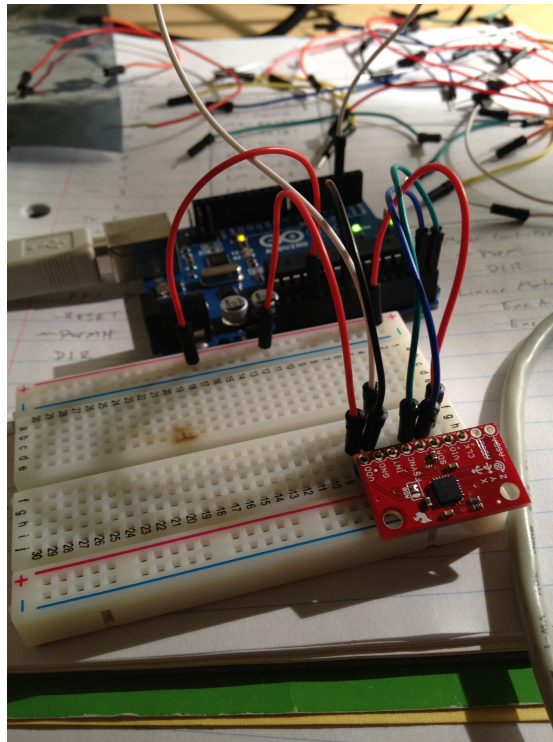
Component: 10A DC Motor Driver Arduino Shield**Functionality: Linear Motor Control**

After the main motor drivers were ready, the next step in prototyping was to bring the linear motor driver online. This motor driver is another shield for the Arduino and is connected between the Wireless Inventors Shield and the Arduino Uno. The driver had jumper pins which make connections to specified I/O pins for use by the Arduino. The pins were chosen based on which Arduino pins were available and fit into the overall design. Once the pin assignments had been established, the motor driver was placed into the system for testing.



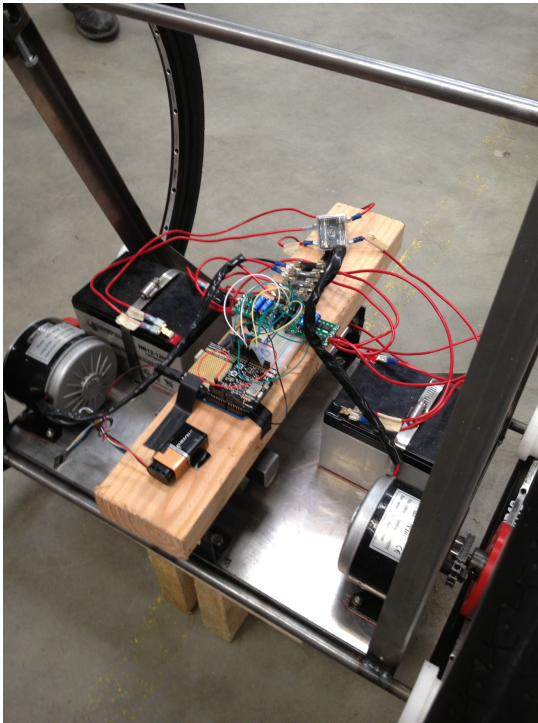
Component: Triple Axis Accelerometer & Gyro Breakout-MPU-6050**Functionality: Leveling Feedback**

Lastly, the accelerometer and gyroscope which is used in providing feedback to the system was worked on. The component sends values corresponding to the chip's inertial data back to the Arduino for use in computations. Prototyping for this component simply involved soldering on the pins and wiring it to the Arduino.

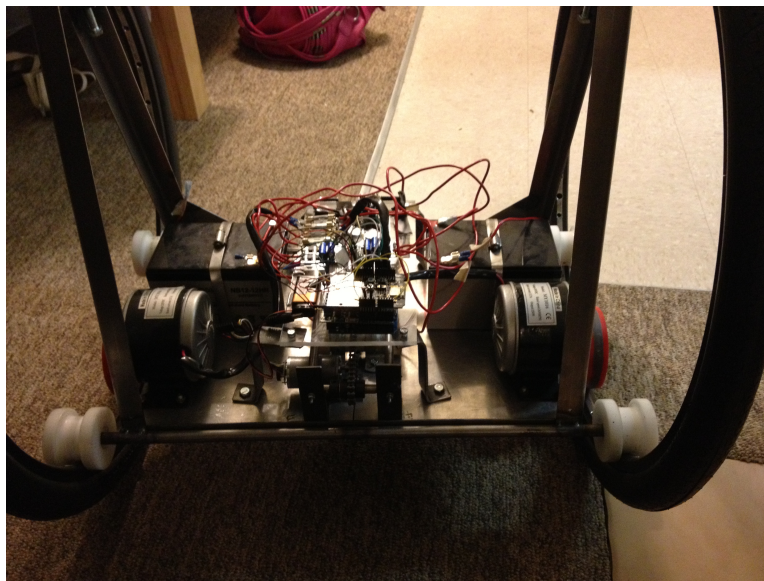
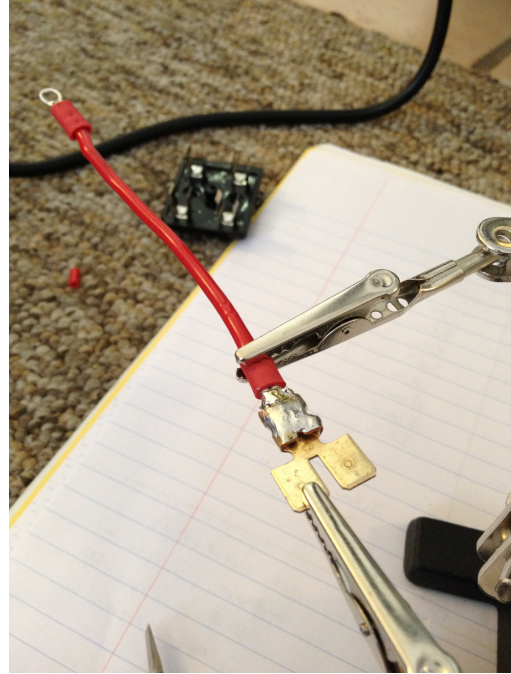
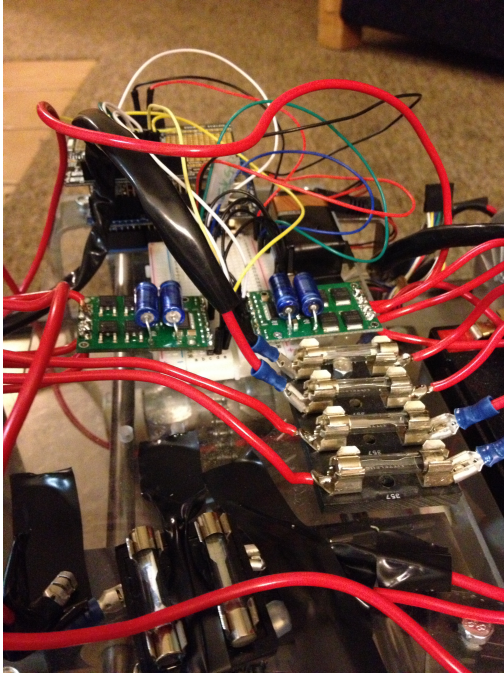


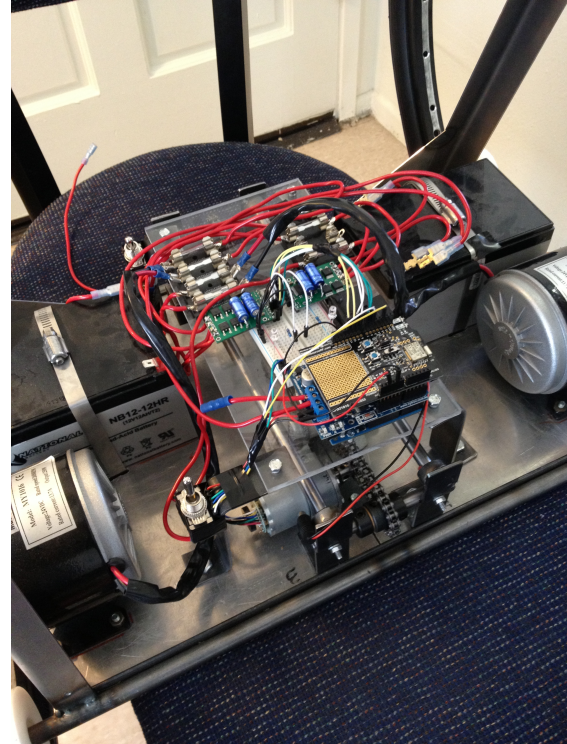
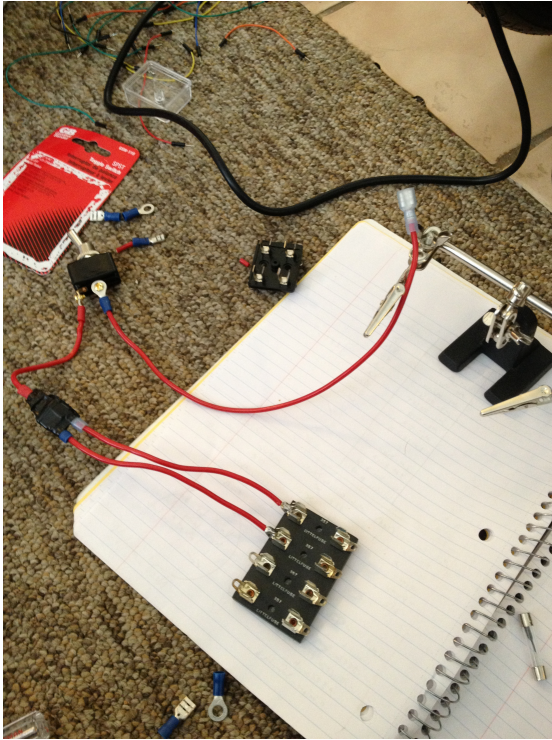
Electrical System Integration Prototyping

After each component had been through prototyping, system integration began. System prototyping involved bringing all of the components together to form one complete system. This was the most lengthy prototyping process and was started after the main parts of the diwheel had been assembled. First, system integration prototyping began with making the connections between each component. The Arduino and its two shields were wired to the breadboard. The Motor drivers were wired to the breadboard as well as to the DC motors and batteries. Fuses and switches were also implemented in the final design which were wired in during system integration prototyping.



After some initial testing, the final system needed to be finalized. Once the geometries of the locations of the batteries and motors could be observed, wire could be cut down to an appropriate length to clean up the system. During this time, the switches were added and the connections were soldered in to keep everything secure and stable. Green LEDs were also added to the design to increase the aesthetic appeal and user friendliness. Once everything was secured and finalized, more system testing began.





Software prototyping was done in parallel with the hardware prototyping. As certain hardware connections were made, the appropriate changes were implemented into the software. Software prototyping continued through into testing as well. When there was an error or undesired functionality, the software had to be updated. One issue we ran into during system prototyping involved the Arduino Uno. Because of the many components, we quickly ran out of I/O pins on the Arduino. In order to solve this issue, the current sensors and fault flags of the motor drivers were removed.

6. Milestone 6: Testing and Verification

Testing is the step in the design process that comes after the prototype has been completely built. Testing for the diwheel has been split up into five different categories, which include mechanical, electrical, system integration, design specifications, and future testing. Each section illustrates each and every test that was performed, how it was performed, the results from the test, and the solutions that would be implemented to correct any failed tests.

6.1 Mechanical Component Testing

The mechanical component testing consists of all the tests that pertained to the physical workings of the diwheel. Each of the mechanical components was tested to ensure they interacted correctly with one another. The tests that were carried out involved dynamic components such as the wheels and the leveling system.

6.1.1 Guide Wheels



Figure 6.1.1: Original Guide Wheel Design

| Test No. | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|----------|---------------------------------------------------|-----------------------------------------------------------------------------------------|---------------------------------------------------------------------------|-----------|----------------------------------|
| 1 | Spinning bike wheels to simulate a forward motion | A reasonable amount of friction between the guide wheels and the rim of the bike wheels | Too much friction between the guide wheels and the rim of the bike wheels | Fail | Material was too soft and sticky |

Table 6.1.1: Testing of Original Guide Wheels

Plan for Correcting:

The material used, soft rubber, ended up creating too much friction and not easily sliding along the bike wheel rims. To fix this, hard plastic was bought and machined into the same shape as the original guide wheels. In doing this, the guide wheels were made into the needed shape. Being made from hard plastic, they do not create as much friction with the bike wheels; instead, they allow the bike wheels to spin freely and are truly guide wheels just keeping everything in place.



Figure 6.1.2: Redesigned Guide Wheels

| Test No. | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|----------|------------------------------------------------------------------------|--------------------------------------------------------------------------|-------------------------------------------------|-----------|----------|
| 2 | Spinning bike wheels to simulate a forward motion (after modification) | Very little friction between the guide wheel and rims of the bike wheels | Very little friction. Allows for free spinning. | Pass | |

Table 6.1.2: Testing of Modified Guide Wheels

6.1.2 Leveling Set-Up

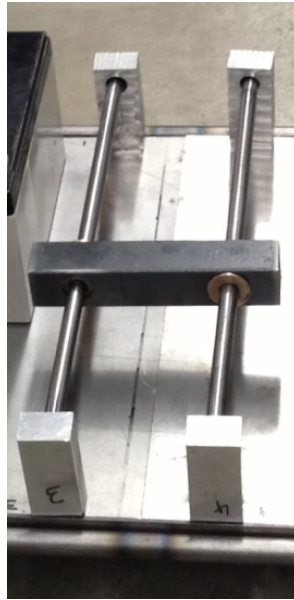


Figure 6.1.3: Original Linear Leveling Set Up

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|--------------------------------------------------------------------------------------------|-------------------------------------------------|-----------------------------------------------------------------|-----------|----------|
| 1 | Sliding mass along the rails | Very little friction | Too much friction and binding | Fail | |
| 2 | With mounts, placing the motor in the proper location and sliding the mass along the rails | Everything fits nicely and has enough clearance | Mounts were too short causing the sliding mass to hit the motor | Fail | |
| 3 | While sliding the mass, observing the stability of the rail mounts | Sturdy and able to take the forces placed on it | Slightly unstable | Fail | |

Table 6.1.3: Linear Leveling Set-Up Tests

Plan for Correcting:

The rail mounts were too short and not strong enough, so new mounts were designed and machined out of steel instead of aluminum. Within the redesign, steel, instead of aluminum, was used for more strength and stability. Also, the redesign simply used sheets of steel that were bent at a 90° angle with cylinders welded for where the rails are set. It was determined that in order for the mass to slide with little friction, the force needed to be applied at the center of the mass.



Figure 6.1.4: Modified Linear Leveling Rail Mounts



Figure 6.1.5: Modified Linear Leveling System Set-Up

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|--------------------------------------------------------------------------------------------|-------------------------------------------------|------------------------------------------------------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------|
| 1 | Sliding mass along the rails | Very little friction | Too much friction | Fail | The reason there is added friction is that the holes were not drilled perfectly straight and therefore the rails bow. |
| 2 | With mounts, placing the motor in the proper location and sliding the mass along the rails | Everything fits nicely | There is enough clearance for all components | Pass | |
| 3 | While sliding the mass, observing the stability of the rail mounts | Sturdy and able to take the forces placed on it | With applied force, the mounts do not move and are able to handle them | Pass | |

Table 6.1.4: Linear Leveling Set-Up Tests

Plan for Correcting:

Ensuring that there is enough lube on the railings while the diwheel is being used will help eliminate some of the unwanted friction. A Teflon lubricant was purchased which aided in reducing the friction.

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|------------------------------|----------------------|----------------------|-----------|----------|
| 4 | Sliding mass along the rails | Very little friction | Very little friction | Pass | |

Table 6.1.5: Linear Leveling Set-Up Tests

6.1.3 Drive Train

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|----------------------|-----------|----------|
| 1 | Drive axel assembly was held in place under the plate in order to check the distance between the drive wheel and the frame of the bike wheel | Drive wheel is in contact with the rim of the bike wheel | There was a 0.5" gap | Fail | |

Table 6.1.6: Drive Train Location Testing

Plan for Correcting:

The drive axel assembly was attached to a separate plate which was in turn mounted to the bottom of the plate. Also, to close the gap even more, the assembly was moved to a new location, forward on the plate instead of in the middle, to bring the drive wheel up higher on the rims of the bike wheels.

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|----------------------------------------|-----------|----------|
| 2 | Drive axel assembly was held in new location in order to check the distance between the drive wheel and the bike wheel | Drive wheel is in contact with the rim of the bike wheel | There was no gap | Pass | |
| 3 | Drive axel assembly was held in new location under the plate in order to check the functionality of the new location | New location would be easily adjustable | New location was not easily adjustable | Fail | |

Table 6.1.7: Redesigned Drive Train Location Testing

Plan for Correcting:

The drive wheel was brought back to the center of the plate, which again increased the gap to the rim of the bike wheels. To close the gap, rubber was secured between the bearings of the drive wheel assembly and the aluminum plate. Also, in order to avoid any bowing along the drive wheel axel plate, a steel support plate was welded perpendicular to the drive wheel axel plate and in the center of the frame.



Figure 6.1.6: Drive Train Assembly Drive Wheel on Bike Wheel Rim



Figure 6.1.7: Drive Axel Plate and Support Assembly

| | | | | | |
|---|----------------------------------------------------------------------------------------------------------------------|----------------------------------------------|--------------------------------------------|------|--|
| 4 | Drive axel assembly was held in new location under the plate in order to check the functionality of the new location | Location would not require precise alignment | Location did not require precise alignment | Pass | |
| 5 | Applied tensioner and checked to see if there was any bowing along the drive axle plate | No bowing will occur | No bowing occurred | Pass | |

Table 6.1.8: Final Design Drive Train Location and Support Testing

6.1.4 Controller Testing



Figure 6.1.9: First Three Successive Peaks for Damping Test

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|----------------------------------------------------------------------------|-----------|-----------------------------------------------------|
| 1 | The body of the Diwheel was lifted until the bottom was perpendicular to the ground. It was then let go and the successive peaks were counted. | The damping of the Diwheel will be calculated | The damping of the diwheel was calculated using the peak times and angles. | Pass | Damping was used to create accurate control models. |

Table 6.1.10: Damping Test Table

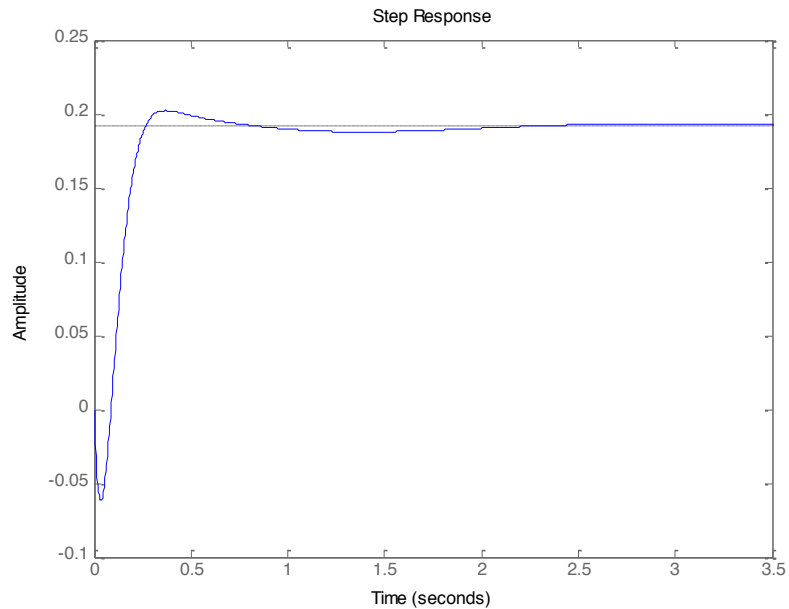


Figure 6.1.10: Response of System to Step Input

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------|
| 1 | Use Sisotool in MatLab to determine controller for linear sliding system. | Controller allows system to attain steady state quickly when step is applied. | Damping was not obtained from actual sliding weight system, so effective controller could not be determined. | Fail | Once the encoder is working we can obtain an accurate controller. |
| 2 | Use Sisotool in MatLab to determine controller for determining the position of sliding weight. | Controller allows system to attain steady state quickly when step is applied. | Controller allows system to attain steady state quickly when step is applied. | Pass | Controller was converted from continuous to discrete and implemented into programming. MatLab code can be found in Appendix D.3. |

Table6.1.11: Controls Testing

6.2 Electrical Component Testing

The electrical component testing involved testing the various parts that make up the electrical system. The motor drivers, wireless communications, DC motors, and sensors all needed to be tested before system integration. Once individual components were successfully tested, they moved on to system integration.

6.2.1 Wireless Receiver/Transmitter

| Test No | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|---------|------------------------------------------------------------------------------------------------|------------------------------------------------------------------------|--------------------------------------------------------------------|-----------|--------------------------------------------------------------------------------|
| 1 | Send an arbitrary value to the Wireless Inventors Shield | RF DIN (Received) LED indicator should light up | RF LED lit up showing success | Pass | Wireless transmitter and receiver are in sync and ready for data |
| 2 | Send a specified value to Arduino to light up an LED | Red LED turns on when "2" is entered and turns off when "1" is entered | LED did not turn on | Fail | RF LED is lit indication received data, but value is not being received |
| 3 | Send a specified value to Arduino to light up an LED. This time using the Hardware Serial Pins | Red LED turns on when "2" is entered and turns off when "1" is entered | LED turned on when 2 was pressed and turned off when 1 was pressed | Pass | Need to use Hardware Serial Pins unless specified otherwise in Arduino program |

Table 6.2.1: Wireless Communication Components Testing Procedures and Results

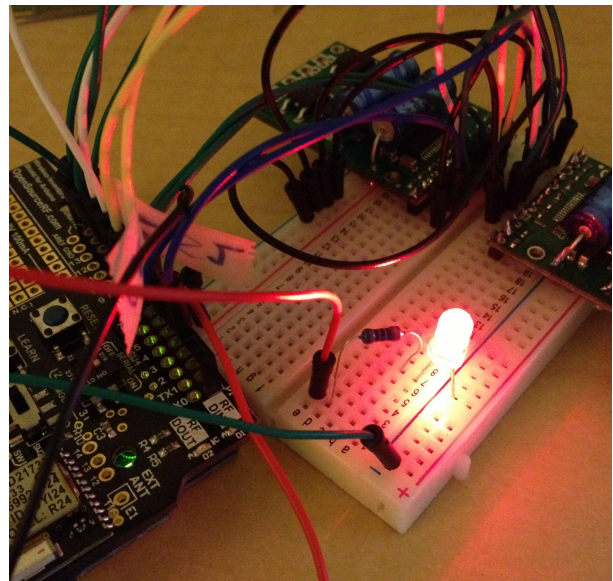
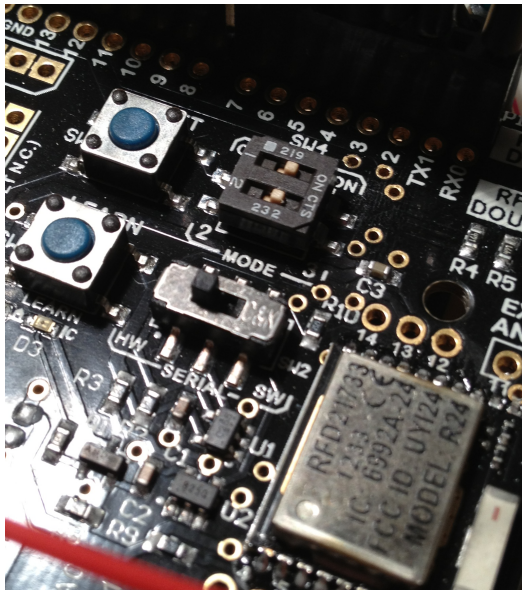


Figure 6.2.1: Wireless Communication Components Test

6.2.2 Drive Wheel Motor Drivers

| Test No | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|---------|--------------------------------------------------------------------------------|----------------------------------------------------------------------------|----------------------------------------------------------------------------|-----------|------------------------------------------------------------|
| 1 | Use LEDs to simulate motor direction with H-Bridge Motor Driver 1A | LEDs turns on when in forward direction and turns off in reverse direction | LEDs turns on when in forward direction and turns off in reverse direction | Pass | Ready for motor integration |
| 2 | Send an enable signal to motors to turn them on with 1 12V battery | Motors turn on in the desired direction when forward command is entered | Motor turned on in the correct direction but only ran for a few seconds | Fail | Possibly not enough power to drive motor? |
| 3 | Send an enable signal to motors to turn them on with 2 12V batteries in series | Motors turn on in the desired direction when forward command is entered | Motor driver exploded, computer turned off, and Arduino shorted out | Fail | Motor driver not rated for the high current draw of motors |

Table 6.2.2: H-Bridge Motor Driver 1A Test Procedures and Results

Plan for Correcting:

The H-Bridge Motor Driver 1A is not rated for the 16A current draw of the DC motors. A new motor driver will be selected that can handle 24V as well as at least 20A of current.

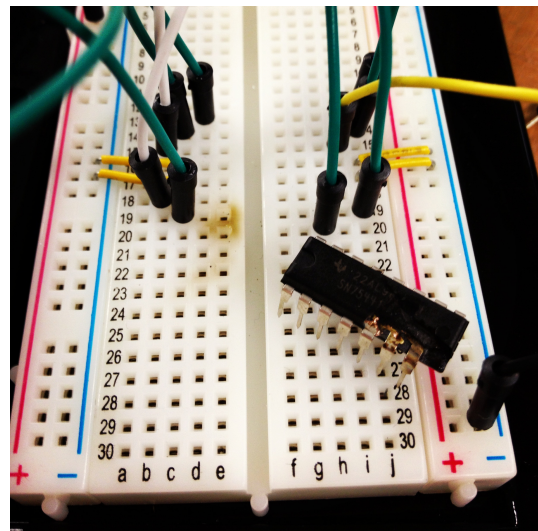
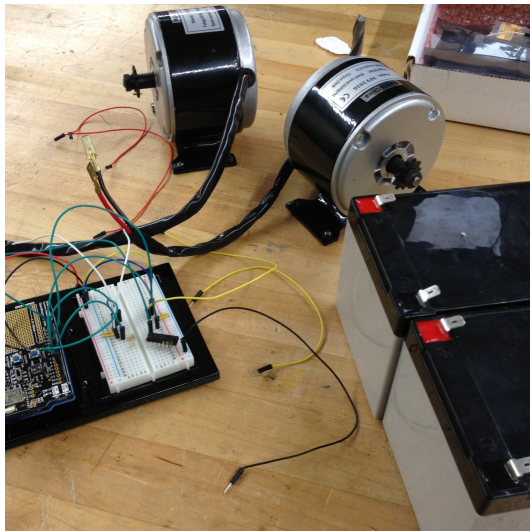


Figure 6.2.2: H-Bridge Motor Driver 1A Testing

Corrected testing with new Pololu High-Power Motor Drivers

| Test No | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|---------|--------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|-----------|------------------------------------------------------------------------------------------|
| 1 | Test both motor drivers simultaneously with DC power supply and Oscilloscope (No Load Test) | Square wave should become wider with a higher PWM value and flip polarities when direction is changed | No Signal was appearing on oscilloscope | Fail | Check reset in programming |
| 2 | Test both motor drivers simultaneously with DC power supply and Oscilloscope (No Load Test) | Square wave should become wider with a higher PWM value and flip polarities when direction is changed | Signal was changing but was very inaccurate and noisy | Fail | Check soldered connections of wires |
| 3 | Test both motor drivers simultaneously with DC power supply and Oscilloscope (No Load Test) | Square wave should become wider with a higher PWM value and flip polarities when direction is changed | Square wave became wider with a higher PWM value and flipped polarities when the direction was changed | Pass | Ready for system integration |
| 4 | Test motor driver speed control functionality with 2 DC motors and 2 12V batteries in series | Motors increase and decrease speed based on PWM signal received | Motors increased and decreased speed based on PWM signal given | Pass | Adjust levels of speed in program for enhanced control |
| 5 | Test motor driver direction control functionality with 2 DC motor sand 2 12V batteries in series | Motors turn in the desired direction based on command | Motors changed direction based on the command given | Pass | Make sure motors do not change directions instantaneously. Add safety measure in program |

Table 6.2.3: Pololu High-Power Motor Driver 24v23 CS Test Procedures and Results

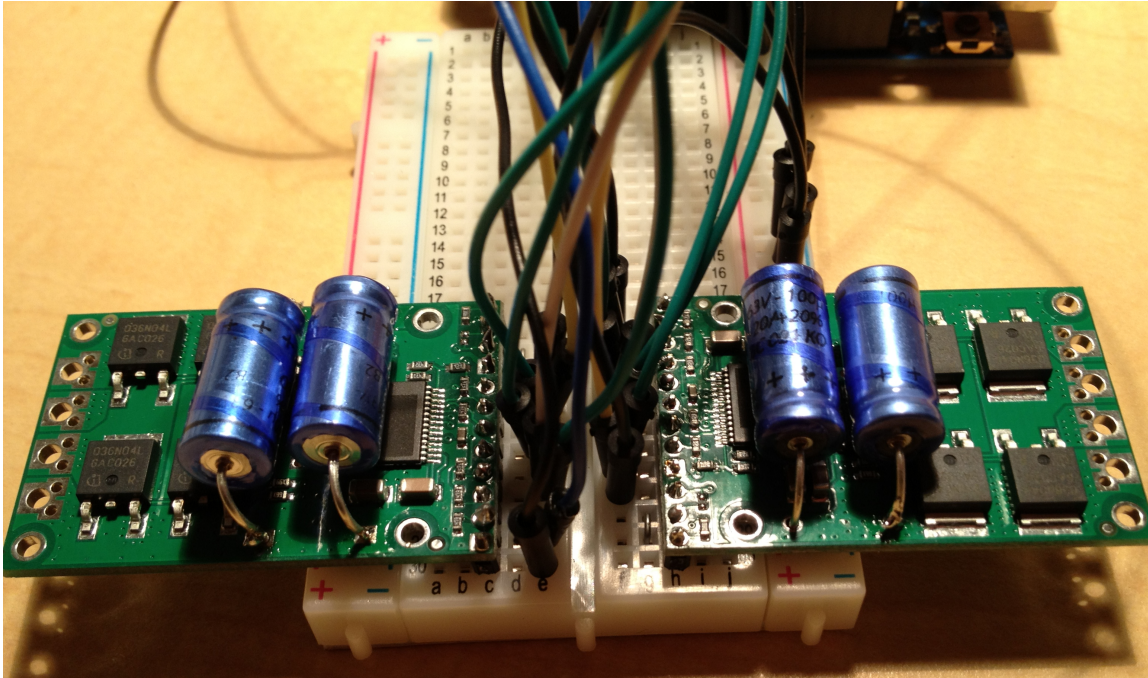


Figure 6.2.3: 23A Motor Driver Arrangement and Setup

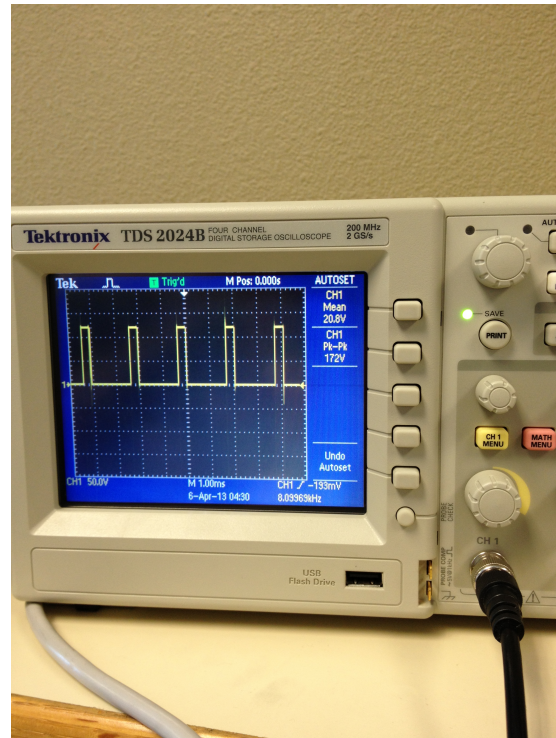
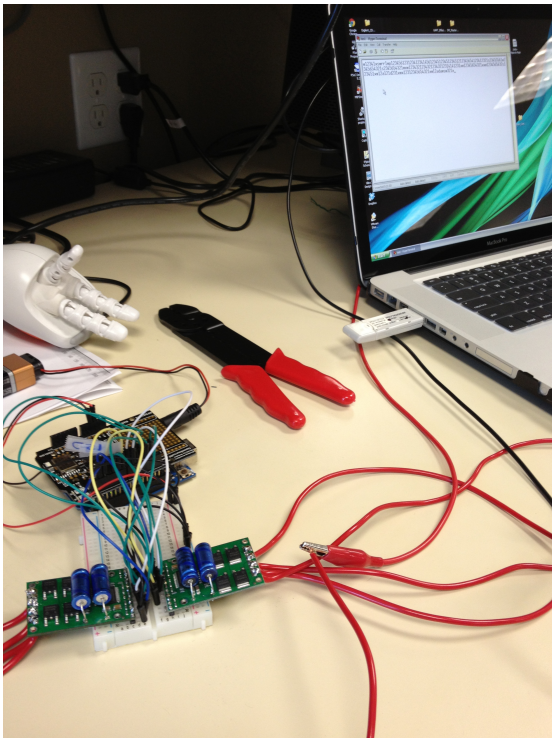


Figure 6.2.4: 23A Motor Driver Testing

6.2.3 10A Motor Driver Arduino Shield

| Test No | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|---------|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|-----------------------------------------------------------------------|-----------|------------------------------------------------------------|
| 1 | Connect shield to Arduino and batteries. Use on board test buttons to test connectivity. | Motor turns forward or backward depending on which button is pressed. | Motor turned the correct direction when buttons were pressed | Pass | Ready to test connected to chain and load |
| 2 | Connect motor to chain and weight to test its ability to move weight | Motor spins and pulls weight along track | Motor moves weight along track nicely | Pass | Ready to measure distance and time taken to move weight |
| 3 | Set motor to full speed and time how long it takes to get weight from one side to the other | Motor moves weight at a constant time each test | Takes about 2 seconds to move weight across each time | Pass | Implement into program |
| 4 | Manually control motor to see what speed is necessary | A certain speed is better than another for moving the weight | High speed seems to be the only one capable of moving weight smoothly | Pass | Test the encoder |
| 5 | Measure the motor turns form the encoder signals | HyperTerminal reads back motor position | System stops responding after motor starts | Fail | Check the interrupt |
| 6 | Measure the motor turns form the encoder signals after interrupt in program has been modified | HyperTerminal reads back motor position | System stops responding after motor starts | Fail | Interrupt is not allowing any other commands to be entered |

Table 6.2.2: 10A Motor Driver Shield for Arduino Testing and Results

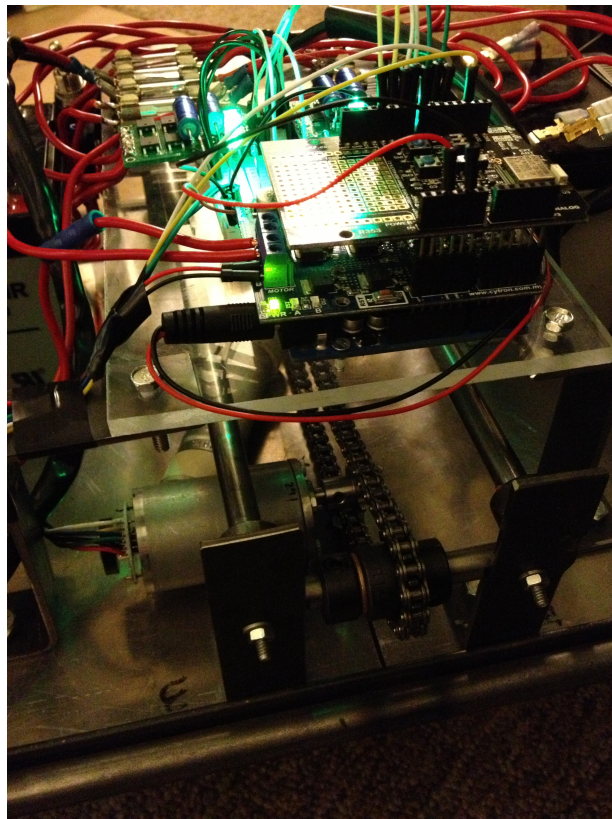


Figure 6.2.4: 10A Motor Driver Arduino Shield Testing

6.2.4 IMU Sensor

| Test No | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|---------|-----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|-----------|---------------------------------------------------------------------------------------|
| 1 | Setup IMU sensor with basic program downloaded from internet and get raw values | HyperTerminal will display values corresponding to yaw, pitch, and roll angles | HyperTerminal displayed values corresponding to yaw, pitch, and roll angles | Pass | Now need to translate raw data into angle value that can be used in control algorithm |
| 2 | Setup IMU sensor with basic program downloaded from internet and get angle values | HyperTerminal will display angle values corresponding to yaw, pitch, and roll angles | HyperTerminal displayed error saying could not connect | Fail | Check raw values again |
| 3 | Setup IMU sensor with basic program downloaded from internet and get raw values | HyperTerminal will display values corresponding to yaw, pitch, and roll angles | HyperTerminal displayed 0s and was not reading any data | Fail | Try another program |
| 4 | Setup IMU sensor with a program downloaded from internet and get raw values | HyperTerminal will display values corresponding to yaw, pitch, and roll angles | HyperTerminal displayed 0s and was not reading any data | Fail | Chip may be broken? |

Table 6.2.4 Triple Axis Accelerometer and Gyroscope Breakout Board Testing and Results

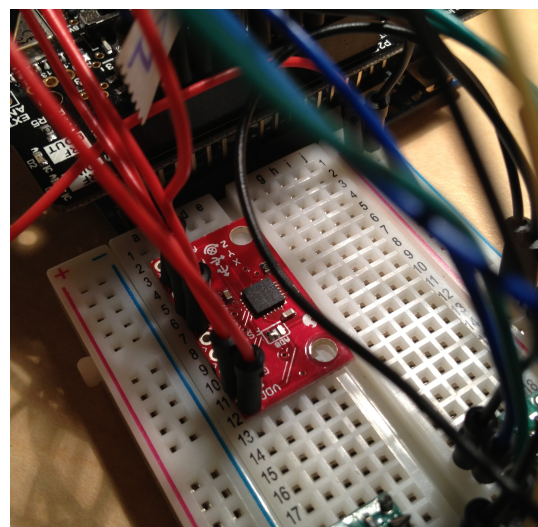
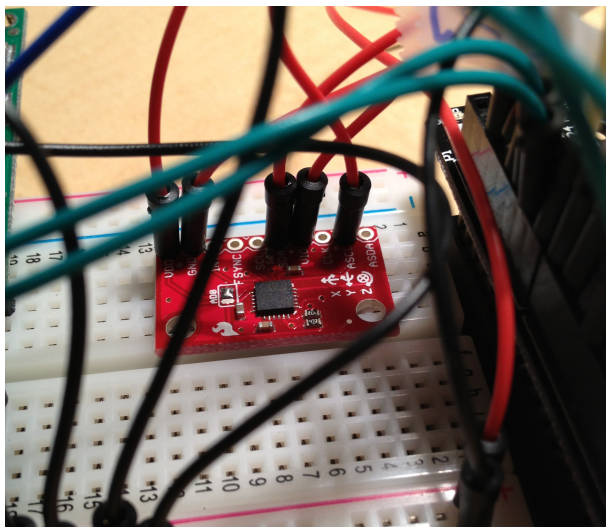


Figure 6.2.5: IMU Sensor Testing

6.2.5 Arduino Uno Microcontroller

| Test No | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|---------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------|-------------------------------------------------------|-----------|----------------------------------------------------------------------|
| 1 | Write a simple program to verify Arduino functionality | A character is entered and the Arduino replies with “received” character sent | Arduino replied with character is was sent | Pass | Arduino is functional and ready for system integration |
| 2 | Attach the Arduino shields and test to see if it can be programmed | Programs the Arduino with the new program | Upload did not go through | Fail | Can it be programmed without the shields? |
| 3 | Upload same program but without shields attached | Programs the Arduino with the new program | Upload was successful | Pass | Shields must be removed to be reprogrammed |
| 4 | Assign pins for all inputs/outputs of motor drivers and motor encoder | Pins are available for each component | Arduino Uno does not have the required number of pins | Fail | Take out optional operations such as current sensors and fault flags |
| 5 | Assign pins after removing certain functions of the motor drivers | Pins are available for each component | There were enough pins for operation of all systems | Pass | Use fuses to avoid over current without current sensors |

Table 6.2.5 Arduino Uno Microcontroller Testing and Results

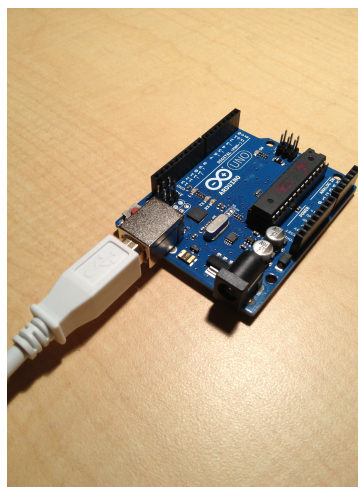


Figure 6.2.6: Arduino Uno Testing

6.3 Diwheel System Testing

After the mechanical and electrical components were integrated together, diwheel system testing could begin. This section of testing included all of the functionality tests. The diwheel was tested to make sure that it would respond correctly to given commands and to ensure the mechanical and electrical systems would run smoothly in unison.

6.3.1 Systems Integration

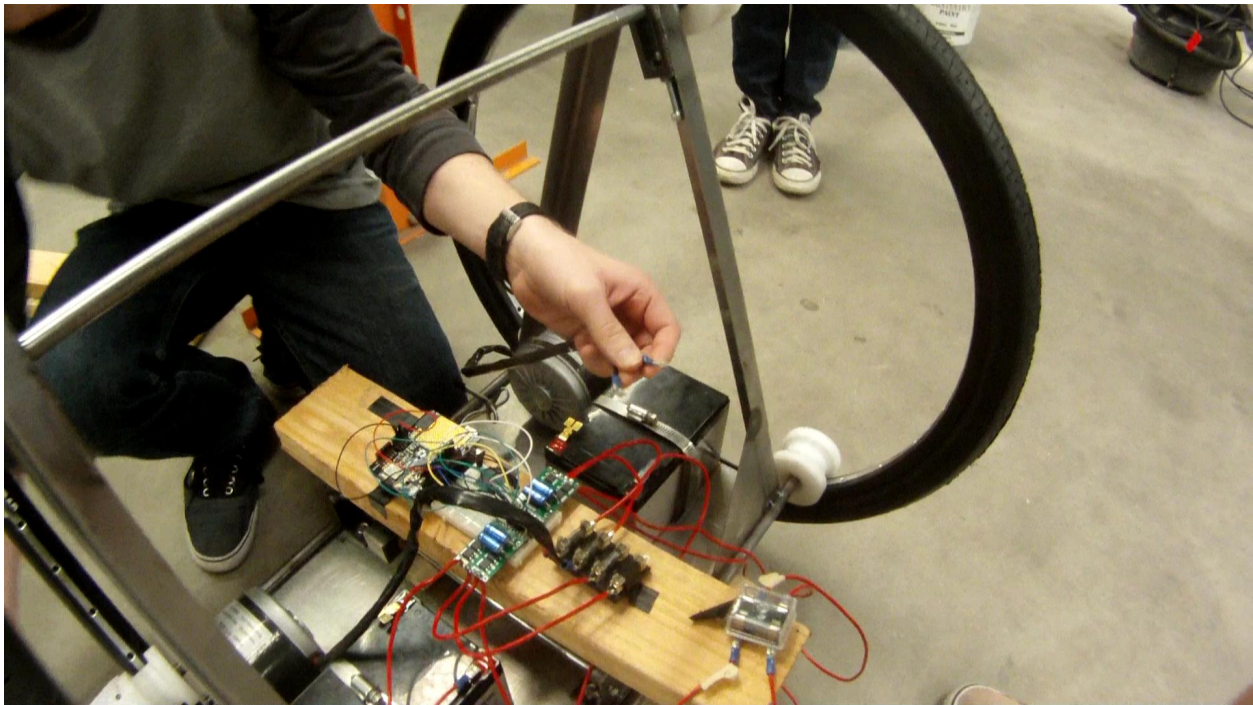


Figure 6.3.1: Initial Integration of the Electrical and Mechanical Systems

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|----------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|---------------------------------------------------------------------------------|-----------|----------|
| 1 | Prop up the diwheel so that the wheels are free to rotate and give the command for forward low speed | Both wheels should spin freely in the forward direction | One wheel could not keep a consistent speed | Fail | |
| 2 | Place the diwheel on the ground and give the command for forward low speed | The diwheel should drive forward at a low speed in a straight line | Diwheel began circling in a counter clockwise direction | Fail | |
| 3 | Prop up the diwheel so that the wheels are free to rotate and give the command for each direction at each speed greater than one | Each wheel should keep a consistent speed when freely spinning | Each wheel kept a consistent speed when each of the various commands were given | Pass | |

Table 6.3.1: Initial system tests

Plan for Correcting:

The left wheel was experiencing more than normal friction due to the guide wheels while it lacked the necessary friction between the diwheel and bike rim in order to transfer smooth power. Both the guide wheel tensioner and the drive wheel axel assembly required adjustments. The guide wheel tensioner needed to be slightly loosened while the drive wheel required more force to be exerted against the bike rim.

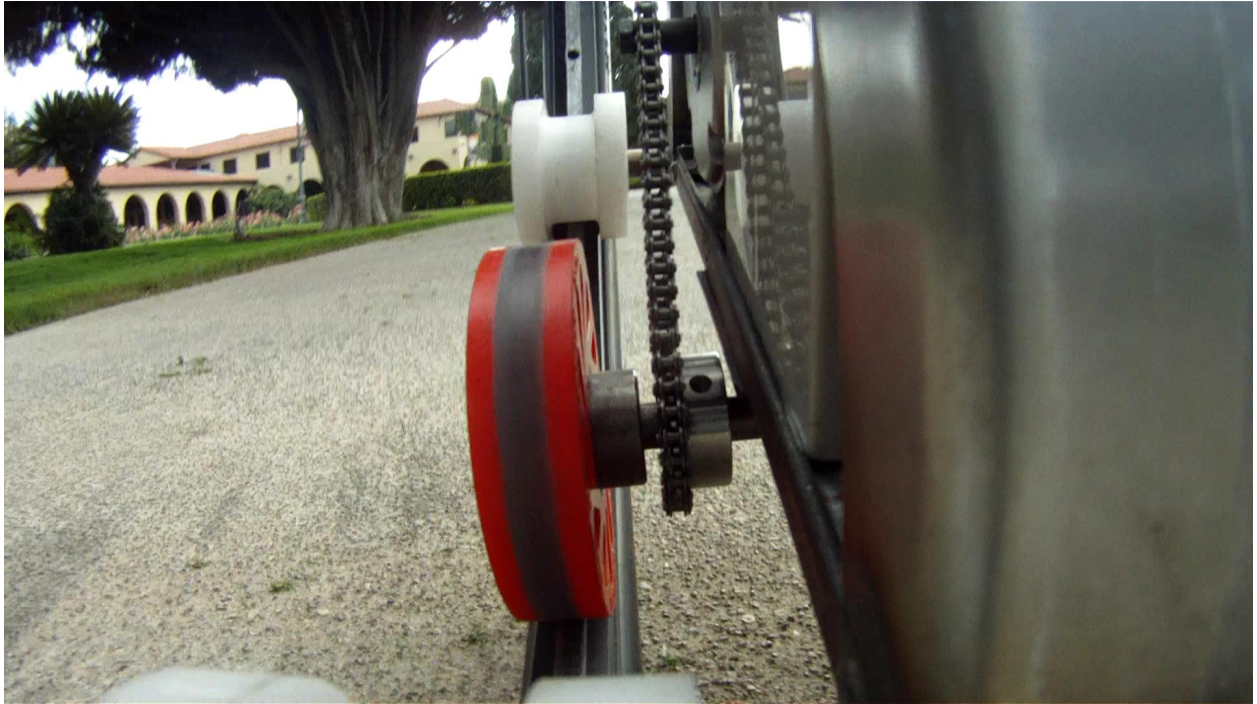


Figure 6.3.2: Adjustable drive wheel as it connects with the inner surface of the bike rim



Figure 6.3.3: Guide Axel Tensioner

6.3.2 Drivability Testing

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|------------------------------------------------------------------------|--------------------------------------------------------------------------|---------------------------------------------------------------------|-----------|----------|
| 4 | Place the diwheel on the ground and give the command for forward speed | The diwheel should drive forward at the desired speed in a straight line | The vehicle drove relatively straight and only slightly veered left | Fail | |

Table 6.3.2: Initial drivability test

Plan for Correcting:

The diwheel still did not drive straight when the forward command at various speeds was given. The mechanical system could not be adjusted to improve this problem, so the decision was made to alter the programming of the controller. The new program would implement active turning so that when the diwheel was cruising and wanted to turn, or adjust for the error, it would drop the either the left or the right voltage by a certain percentage. For the forward command the right wheel was permanently reduced by 5% in order to counteract the diwheel from veering.

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|------------------------------------------------------------------------|--------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|-----------|----------|
| 5 | Place the diwheel on the ground and give the command for forward speed | The diwheel should drive forward at the desired speed in a straight line | The vehicle traveled along a much straighter path. It also could now correct itself and initiate a turn | Pass/Fail | |

Table 6.3.3: Secondary drivability test



Figure 6.3.4: Testing the updated control system

Plan for Correcting:

The diwheel was becoming more controllable with the new programming and became even better with operator practice. The programming still produced a controllability that was rough around the edges. The new program will be refined with acute changes that will result in a more stable system. A greater variety of speeds will be implemented, along with different levels of turning. For the lower speeds the turning will drop either wheel 30% power, the medium speeds will drop either wheel 20% power and the top speeds will only drop either wheel by 10% power.

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|--------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-----------|-----------------|
| 6 | Give the command for forward and adjust the speed and turning accordingly | The diwheel should drive at the desired speed and direction of the operator | The vehicle is now moderately controllable by the operator, but still is not superb | Pass/Fail | |
| 7 | Continue testing diwheel movement and check the specifications for its functionality | Diwheel moves and keeps chassis swing below 15 degrees | While turning, diwheel stopped working | Fail | Check circuitry |

Table 6.3.4: Final drivability test.

The circuitry was checked and no faults could be found. The motors were connected directly to the batteries to check motor/battery functionality. Both motors and batteries were operational. The digital side of the components was checked and everything was function. The motor drivers are the only components not responding and may have shorted out. The reason is unknown as a simple spin had been performed many times previously.



Figure 6.3.5: Final controllability testing

6.3.3 Linear motor function testing.

| Test No | Test Procedure | Expected Result | Actual Result | Pass/Fail | Comments |
|---------|---------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|-----------------------------------------------------------------------|-----------|---------------------------------------------------------|
| 1 | Connect shield to Arduino and batteries. Use on board test buttons to test connectivity. | Motor turns forward or backward depending on which button is pressed. | Motor turned the correct direction when buttons were pressed | Pass | Ready to test connected to chain and load |
| 2 | Connect motor to chain and weight to test its ability to move weight | Motor spins and pulls weight along track | Motor moves weight along track nicely | Pass | Ready to measure distance and time taken to move weight |
| 3 | Set motor to full speed and time how long it takes to get weight from one side to the other | Motor moves weight at a constant time each test | Takes about 2 seconds to move weight across each time | Pass | Implement into program |
| 4 | Manually control motor to see what speed is necessary | A certain speed is better than another for moving the weight | High speed seems to be the only one capable of moving weight smoothly | Pass | Test the encoder |

Table 6.3.4: Specifications tests.

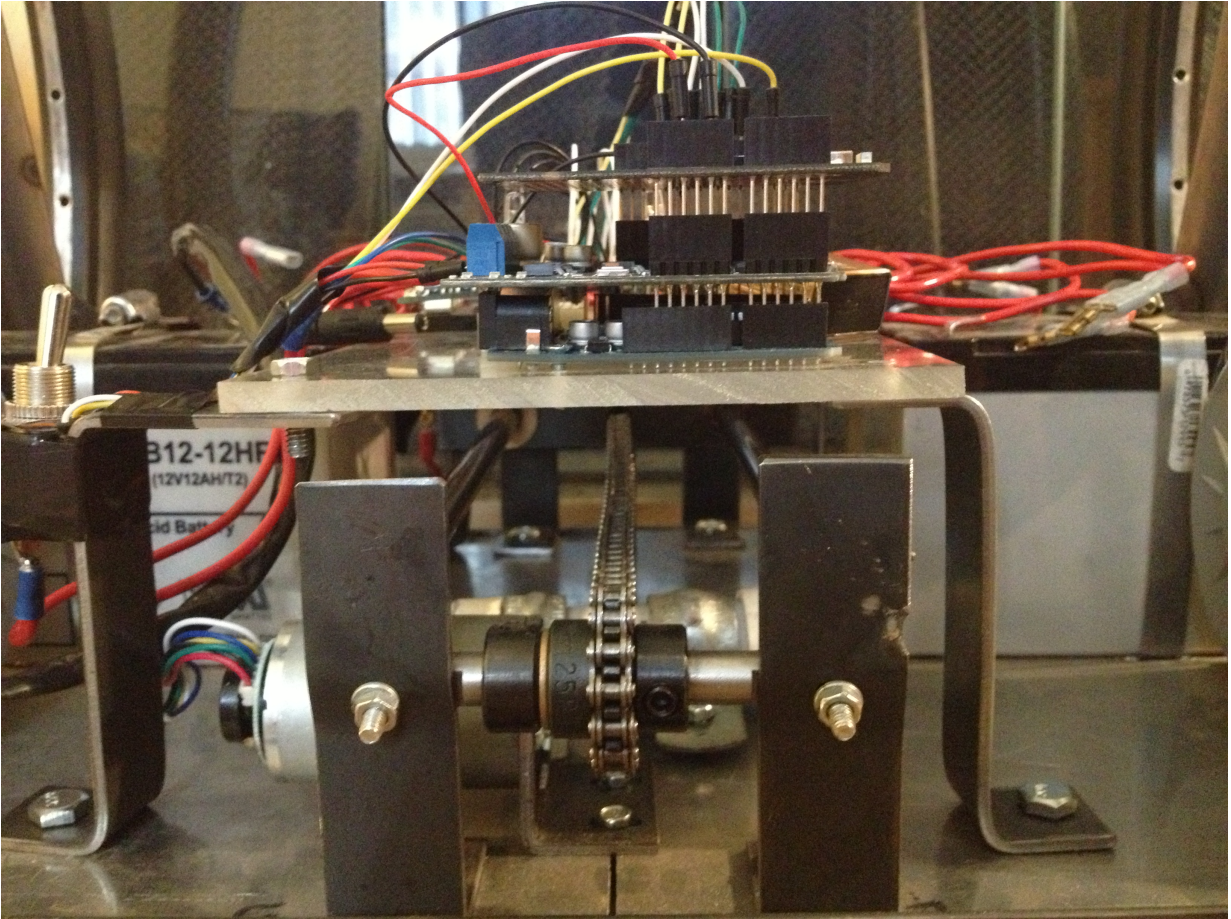


Figure 6.3.6: Leveling system

6.4 Specifications Testing

Specification testing was an important section of testing because it would prove that the diwheel was constructed to perform as initially desired. The design specification list was used to determine the test that would be carried out. These tests would assess the performance of the diwheel and ensure the proper functionality as an entire system.

| Test No. | Test Procedure | Expected Results | Actual Results | Pass/Fail | Comments |
|----------|----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|-----------|----------|
| 1 | Record the weight of the entire diwheel on a scale | The diwheel is expected to weigh less than 30kg | 26.2kg | Pass | |
| 2 | Record the amount of time the batteries last for operation | The diwheel batteries last at least 30 minutes | Batteries lasted past 30 minutes and spanned several days of testing | Pass | |
| 3 | The diwheel will be dropped from a height of 0.5 meters | Diwheel sustainability of a 0.5 meter drop without loss of functionality | The diwheel remained functional after the impact of the drop test | Pass | |
| 4 | Weigh the hardware, attach it and run the diwheel to ensure that the hardware remains secure | The attached hardware will remain secure and will not inhibit standard functionality of the diwheel | The hardware remained secure and did not inhibit standard functionality of the diwheel | Pass | |
| 5 | Diwheel testing by using the gyroscope and linear leveling system and using the desired controller | The control system will stabilize the diwheel and keep the overall angle below 15 degrees when accelerating | Gyroscope and encoder were not functioning so test could not be performed. | Fail | |

Table 6.4.1: Specifications tests.

6.5 Future Testing Plan

It has been determined that while controlling the diwheel from a computer and only having the input of buttons to be pushed and released, the diwheel has reached its level of controllability. In order for the vehicle to become superbly controllable there would have to be a change in the user interface. This means a joystick or a gaming controller would have to be implemented into the program. These controllers have the ability to administer a smooth signal that can gradually increase, decrease and turn mechanical system.



Figure 6.5.1: Xbox controller with joysticks.

Due to unforeseen complications with the motor drivers and the gyroscope, there were some specifications testing that could not be completed. The tests that are listed below will be left for the future mechanical engineering students.

| Specifications | Proposed Testing Procedure |
|------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Diwheel maximum velocity of 7.3 m/s | Putting the diwheel into motion and then timing how long it takes to travel a set distance between two points |
| Diwheel maximum acceleration time of 3 seconds | Accelerate diwheel and calculate its actual acceleration |
| Diwheel is equipped with a functioning failsafe unit | Simulate loosing battery power, frequency disturbance, and receiver being out of range and ensure the failsafe is operating properly under each circumstance |

Table 6.5.1: Future Testing Plans and Procedures

Conclusion

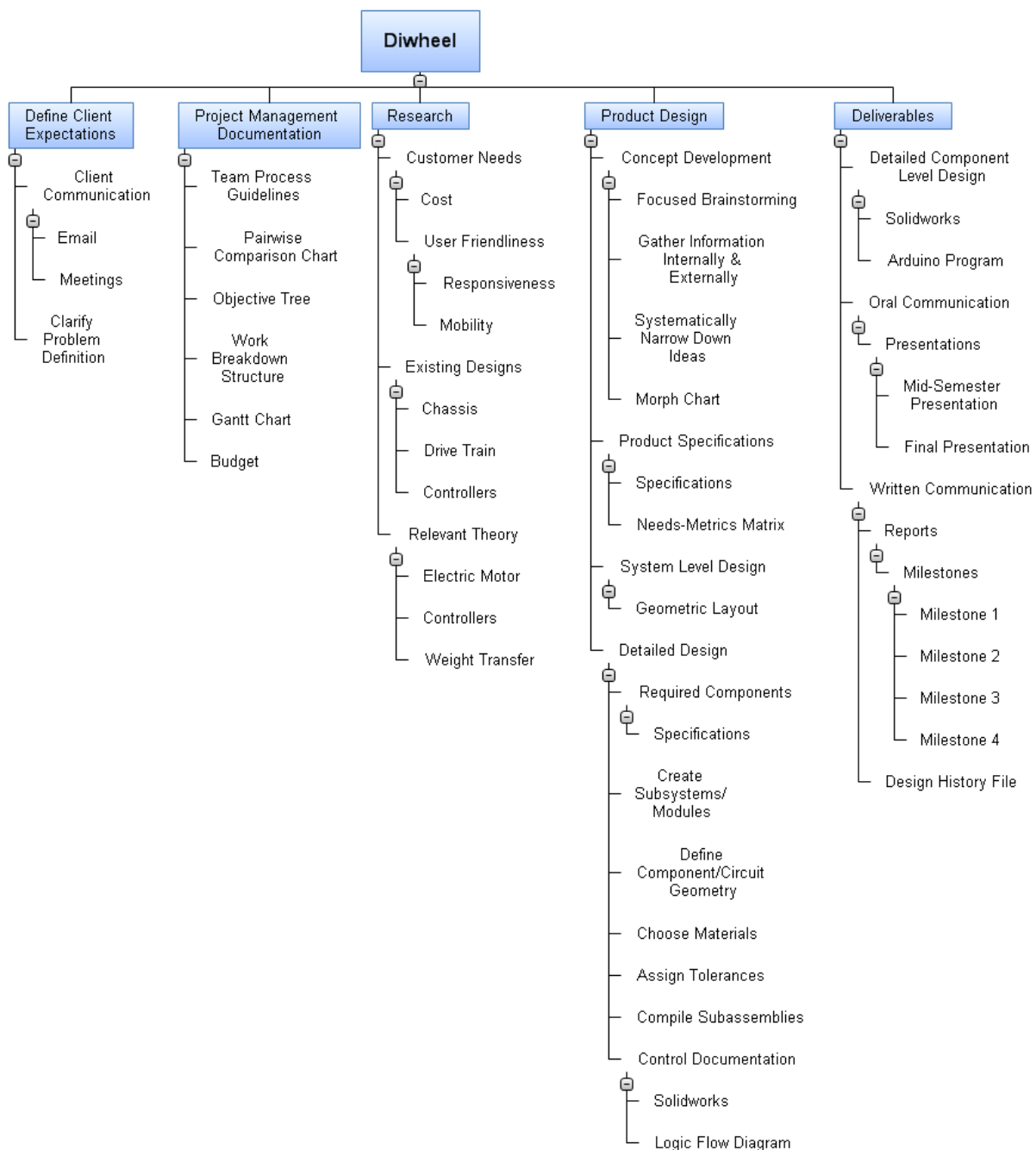
In order to solve to problem of making a two-wheeled, mobile, durable vehicle on which different hardware can be mounted and kept level, the team researched and designed a suitable product for manufacturing. With the help of computer-aided design and computer programming, we aim to construct our proposed model and test the functionality of it. The diwheel will be built within budget and within the allotted time given by the client.

A. Appendix A: Project Management Plan

The Project Management Plan includes Work Breakdown Structure and Gantt chart. The Project Management Plan shows what steps need to be taken as a team to achieve the final product. The Work Breakdown Structure is a graphical representation of how the project is broken down. It is broken down into sections such as the milestones and deliverables and then each of those sections are broken down further into subsections. The Gantt chart is like a timeline version of the Work Breakdown Structure. It shows the start and end date of each portion as well as completion percentage and who is in charge of each section. The Project Management Plan will make sure that the team stays on track with the deadlines.

A.1 Work Breakdown Structure

A work breakdown structure is a flow chart or tree representation of the Gantt Chart. It shows the tasks that need to be finished in order for a project to be completed. The tasks listed then have subsections that list the topics that the task must cover in order to be completed. For this project there are the tasks of Define Client Expectations, Research, Product Design, Project Management Documentation and Deliverables.



A.2 Gantt Chart

A Gantt Chart is a timeline representation of the Work Breakdown Structure. It provides deadlines and durations for each task that was listed in the Work Breakdown Structure. For this project the tasks that are listed in the Gantt Chart are Define Client Expectations, Research, Product Design, Project Management Documentation and Deliverables. Each task has a deadline in order to meet the final deadline of December 7, 2012.

| Task Name | Duration | Start | End | Predecessors | Completion | Priority | Resources | Timeline (27 Aug '12 - 10 Dec '12) | | | | | | | | | | | |
|----------------------------------------------------------------------------------|----------|-----------|------------|--------------|------------|----------|------------|--------------------------------------|--|--|--|--|--|--|--|--|--|--|--|
| <input checked="" type="checkbox"/> [Milestone] Define Client Expectations | 70 days | 9/3/2012 | 12/7/2012 | | 26% | 0 | Dennis | [Gantt bar from 9/3/12 to 12/7/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Client Communication | 70 days | 9/3/2012 | 12/7/2012 | | 0% | 0 | Dennis | [Gantt bar from 9/3/12 to 12/7/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Email | 70 days | 9/3/2012 | 12/7/2012 | | 0% | 0 | Dennis | [Gantt bar from 9/3/12 to 12/7/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Meetings | 70 days | 9/3/2012 | 12/7/2012 | | 0% | 0 | Dennis | [Gantt bar from 9/3/12 to 12/7/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Clarify Problem Definition | 11 days | 9/17/2012 | 10/1/2012 | | 83% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Project Management Documentation | 41 days | 9/17/2012 | 11/12/2012 | | 100% | 0 | Dean | [Gantt bar from 9/17/12 to 11/12/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Team Process Guidelines | 6 days | 9/24/2012 | 9/24/2012 | | 100% | 0 | Dean | [Gantt bar at 9/24/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Pairwise Comparison Chart | 6 days | 10/1/2012 | 10/1/2012 | | 100% | 0 | Paris | [Gantt bar at 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Objective Tree | 6 days | 9/21/2012 | 10/1/2012 | | 100% | 0 | Paris | [Gantt bar from 9/21/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Work Breakdown Structure | 6 days | 10/1/2012 | 10/8/2012 | | 100% | 0 | Paris | [Gantt bar from 10/1/12 to 10/8/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Gantt Chart | 6 days | 10/1/2012 | 10/8/2012 | | 100% | 0 | McLaughlin | [Gantt bar from 10/1/12 to 10/8/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Budget | 6 days | 11/5/2012 | 11/11/2012 | | 0% | 0 | McLaughlin | [Gantt bar from 11/5/12 to 11/11/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Research | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Customer Needs | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Customer Needs | 6 days | 9/24/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/24/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] User Friendliness | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Cost | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Responsiveness | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Mobility | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Existing Designs | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Chassis | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Drive Train | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Controllers | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Paris | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Relevant Theory | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Electric Motor | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Controllers | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Paris | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |
| <input checked="" type="checkbox"/> [Milestone] Weight Transfer | 11 days | 9/17/2012 | 10/1/2012 | | 100% | 0 | Vaporf | [Gantt bar from 9/17/12 to 10/1/12] | | | | | | | | | | | |

| ID | Task Name | Duration | Start Date | End Date | Progress % | Count | Assignee |
|----|--------------------------------------------|----------|------------|------------|------------|-------|------------------|
| 28 | Product Design | 31 days | 10/15/2012 | 11/26/2012 | 0% | 0 | |
| 29 | Concept Development | 11 days | 10/15/2012 | 10/29/2012 | 0% | 0 | Dean |
| 30 | Focus/Broadening | 11 days | 10/15/2012 | 10/29/2012 | 0% | 0 | Vaporf |
| 31 | Gather Information Internally & Externally | 11 days | 10/15/2012 | 10/29/2012 | 0% | 0 | Dennis |
| 32 | Systemically Narrow Down Ideas | 11 days | 10/15/2012 | 10/29/2012 | 0% | 0 | McLaughlin |
| 33 | Morph Chart | 11 days | 10/15/2012 | 10/29/2012 | 0% | 0 | |
| 34 | Product Specifications | 5 days | 10/30/2012 | 11/5/2012 | 0% | 0 | Vaporf |
| 35 | Specifications | 2 days | 10/30/2012 | 10/31/2012 | 0% | 0 | Parisi |
| 36 | Needs-Merics Matrix | 5 days | 10/30/2012 | 11/5/2012 | 0% | 0 | |
| 37 | System Level Design | 1 day | 11/8/2012 | 11/8/2012 | 0% | 0 | Dean |
| 38 | Geometric Layout | 1 day | 11/8/2012 | 11/8/2012 | 0% | 0 | |
| 39 | Detailed Design | 14 days | 11/7/2012 | 11/26/2012 | 0% | 0 | |
| 40 | Required Components | 8 days | 11/7/2012 | 11/16/2012 | 0% | 0 | Vaporf |
| 41 | Specifications | 8 days | 11/7/2012 | 11/16/2012 | 0% | 0 | McLaughlin |
| 42 | Create Subsystems/Modules | 8 days | 11/7/2012 | 11/16/2012 | 0% | 0 | Vaporf |
| 43 | Define Component/Critical Geometry | 8 days | 11/7/2012 | 11/16/2012 | 0% | 0 | Parisi |
| 44 | Choose Materials | 8 days | 11/7/2012 | 11/16/2012 | 0% | 0 | Vaporf |
| 45 | Assign Tolerances | 8 days | 11/7/2012 | 11/16/2012 | 0% | 0 | Dennis |
| 46 | Compile Subassemblies | 8 days | 11/7/2012 | 11/16/2012 | 0% | 0 | McLaughlin |
| 47 | Control Documentation | 6 days | 11/19/2012 | 11/26/2012 | 0% | 0 | |
| 48 | Solidworks | 8 days | 11/19/2012 | 11/28/2012 | 0% | 0 | Dennis |
| 49 | Logic Flow Diagram | 6 days | 11/19/2012 | 11/26/2012 | 0% | 0 | Parisi |
| 50 | Deliverables | 6 days | 9/17/2012 | 12/7/2012 | 5% | 0 | |
| 51 | Detailed Component Level Design | 16 days | 11/5/2012 | 11/26/2012 | 0% | 0 | Dennis |
| 52 | Solidworks | 16 days | 11/5/2012 | 11/28/2012 | 0% | 0 | Parisi |
| 53 | Audience Program | 16 days | 11/5/2012 | 11/28/2012 | 0% | 0 | |
| 54 | Oral Communication | 35 days | 10/15/2012 | 12/3/2012 | 0% | 0 | |
| 55 | Presentations | 35 days | 10/15/2012 | 12/3/2012 | 0% | 0 | |
| 56 | Mid-Semester Presentation | 6 days | 10/15/2012 | 10/22/2012 | 0% | 0 | Vaporf, Den... |
| 57 | Final Presentation | 6 days | 11/23/2012 | 12/9/2012 | 0% | 0 | Parisi, Dean ... |
| 58 | Written Communication | 6 days | 9/17/2012 | 12/7/2012 | 9% | 0 | |
| 59 | Reports | 14 days | 9/25/2012 | 11/26/2012 | 22% | 0 | |
| 60 | Milestones | 5 days | 9/25/2012 | 10/1/2012 | 100% | 0 | Dennis |
| 61 | Milestone 1 | 5 days | 10/6/2012 | 10/15/2012 | 0% | 0 | Dennis |
| 62 | Milestone 2 | 6 days | 10/26/2012 | 11/1/2012 | 0% | 0 | Dennis |
| 63 | Milestone 3 | 6 days | 11/16/2012 | 11/26/2012 | 0% | 0 | Dennis |
| 64 | Milestone 4 | 6 days | 11/16/2012 | 11/26/2012 | 0% | 0 | Dennis |
| 65 | Design History File | 60 days | 9/17/2012 | 12/7/2012 | 0% | 0 | Dennis |

A.3 Statement of Work

A Statement of Work is a bulleted list derived from the Work Breakdown Structure. It shows the tasks that need to be finished in order for a project to be completed. The tasks listed then have subsections that list the topics that the task must cover in order to be completed. The Statement of Work clearly shows who is in charge of ensuring completion for each of the given tasks.

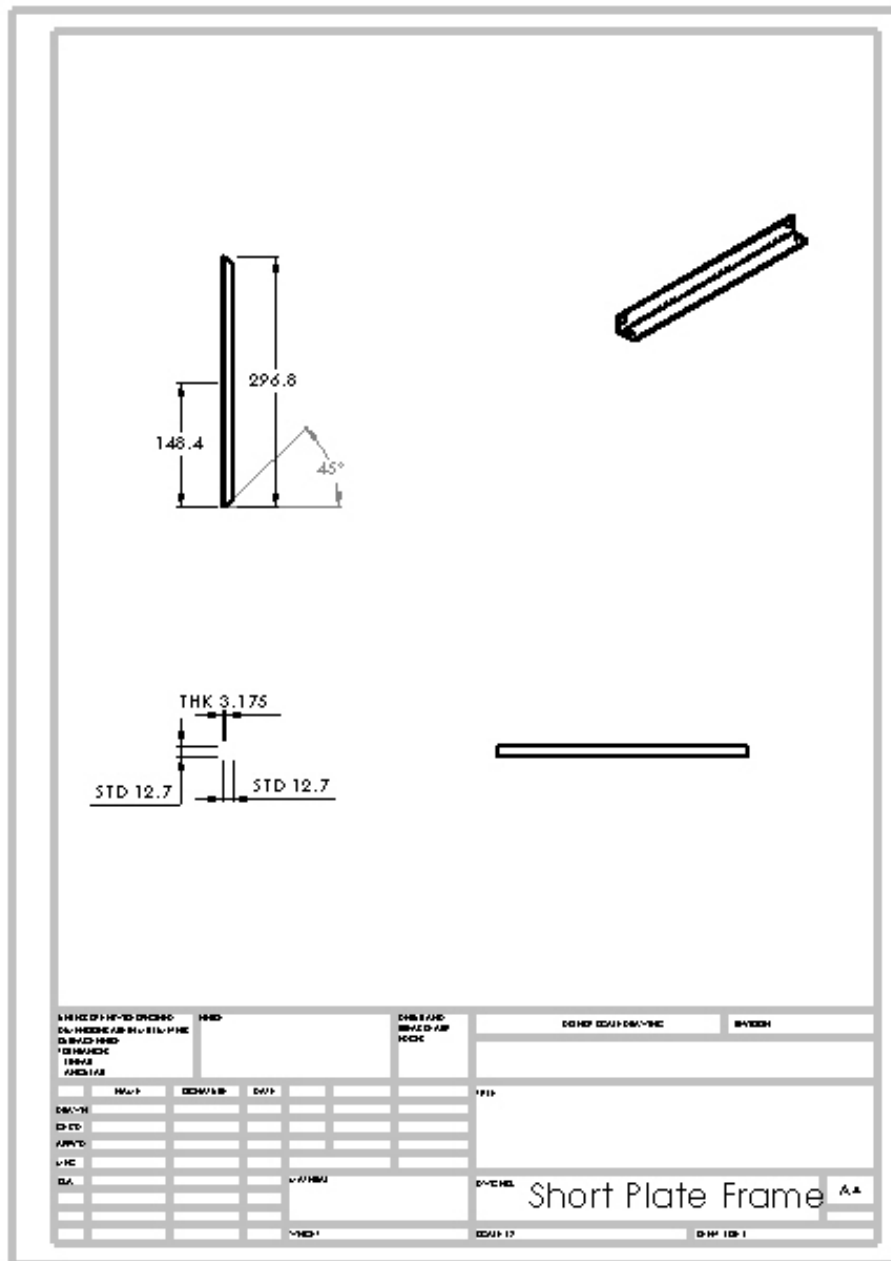
Construction of Design

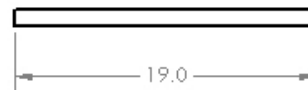
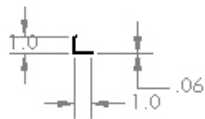
- Construct Mechanical Components (Dean, Dennis, Vanjoff, McLaughlin)
- Construct Electrical Components (Parisi)

B. Appendix B: Detailed Design

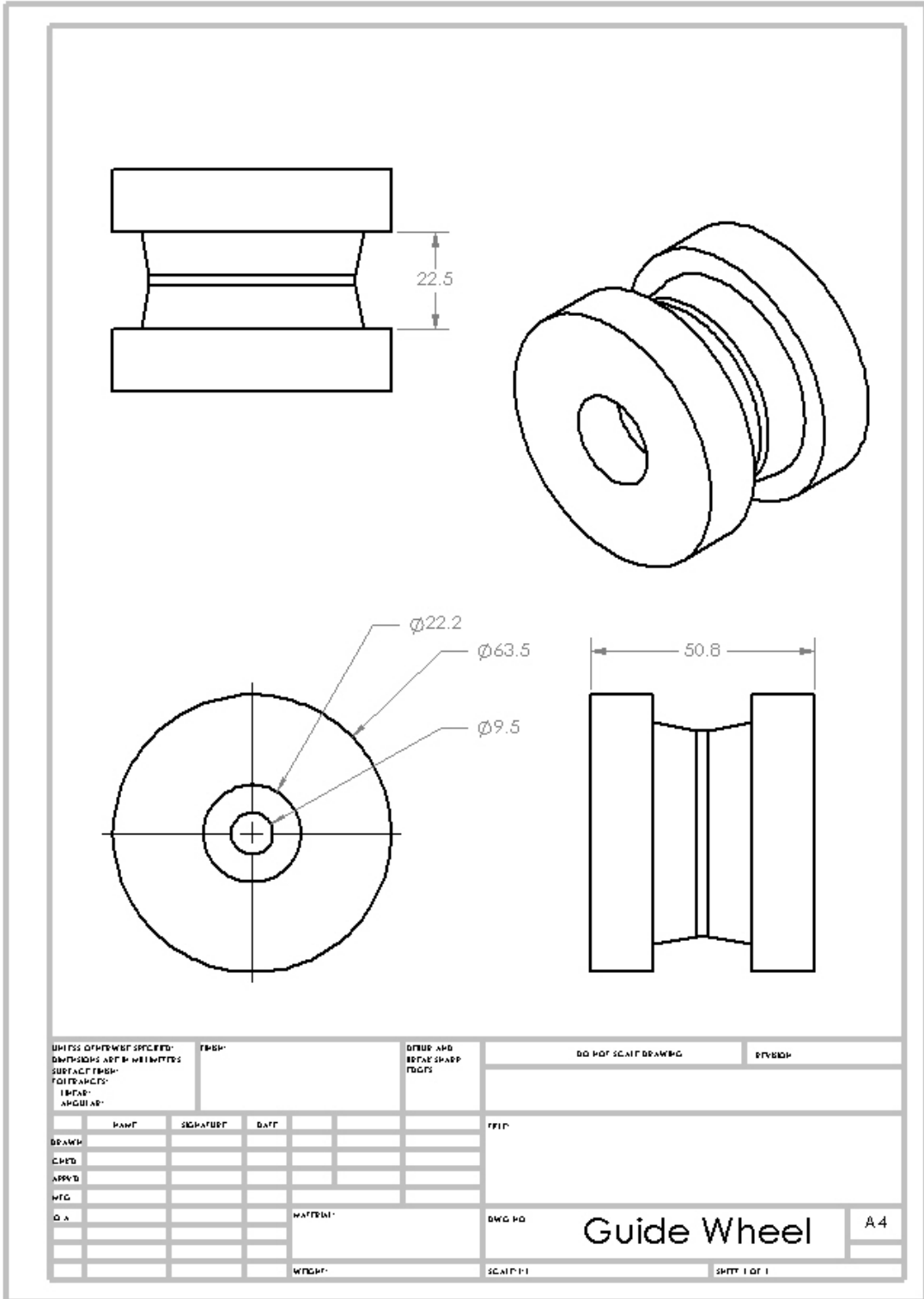
B.1 Drawings

The detailed design drawings are crucial elements to the design of the product. They are 2-D representations of the 3-D model. These drawings include the detailed dimensions and tolerances needed for the manufacturing of the product.



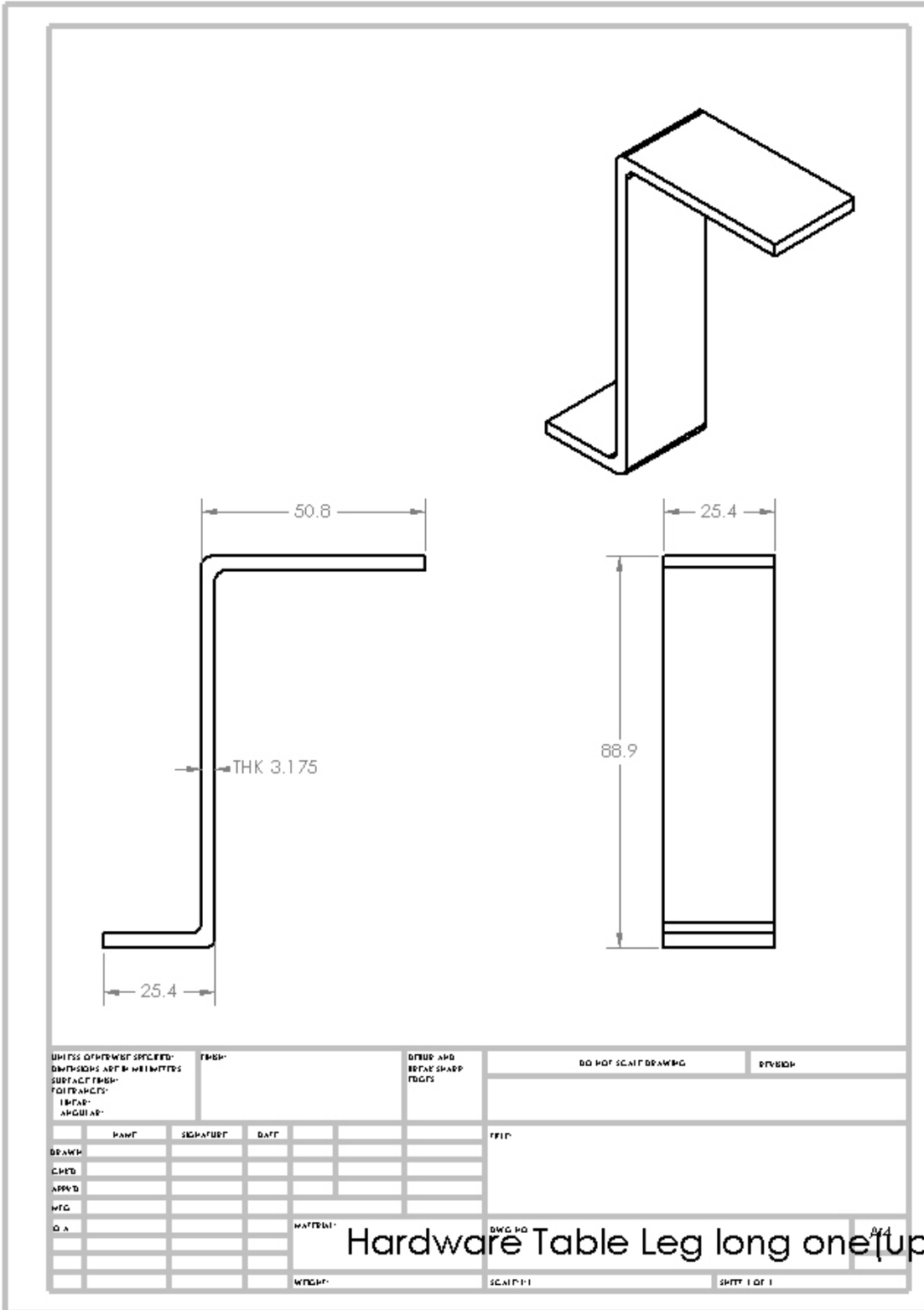


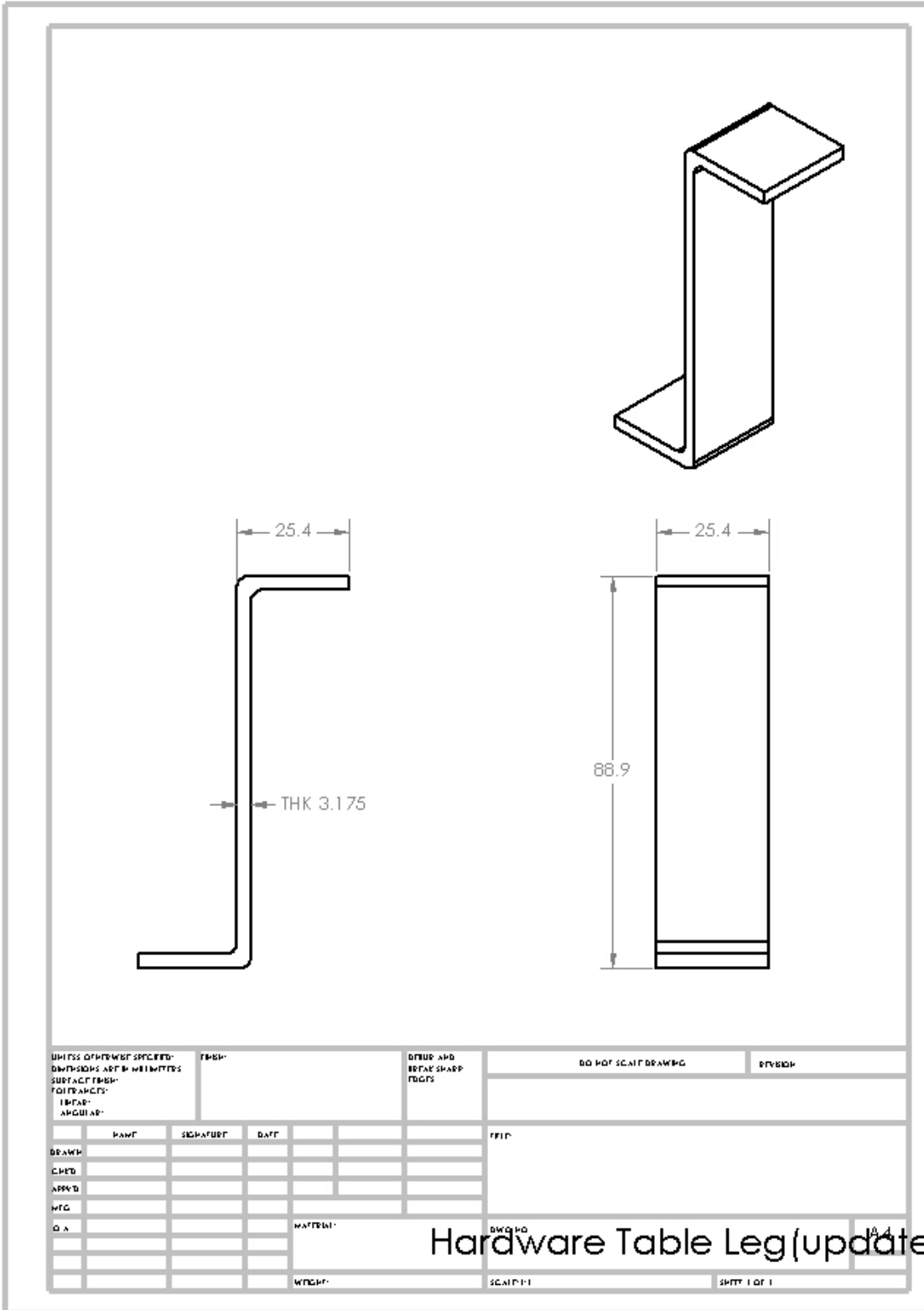
| | | | | | | | | | |
|-----------------------------------------------------------------------------------------------------------------------|--|------------|--|------------------------------------|--|----------------------|--|--------------|--|
| UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: FINISH: ANGULAR: | | FINISH: | | DITUE AND BEFORE SHARP EDGES | | DO NOT SCALE DRAWING | | STANDARD | |
| DRAWN: | | SIGNATURE: | | DATE: | | TITLE: | | | |
| CHECKED: | | | | | | | | | |
| APPROVED: | | | | | | | | | |
| MFG: | | | | | | | | | |
| D.A. | | | | MATERIAL: | | DWG NO. | | New Arm | |
| | | | | | | | | A4 | |
| | | | | WEIGHT: | | SCALE: | | SHEET 1 OF 1 | |

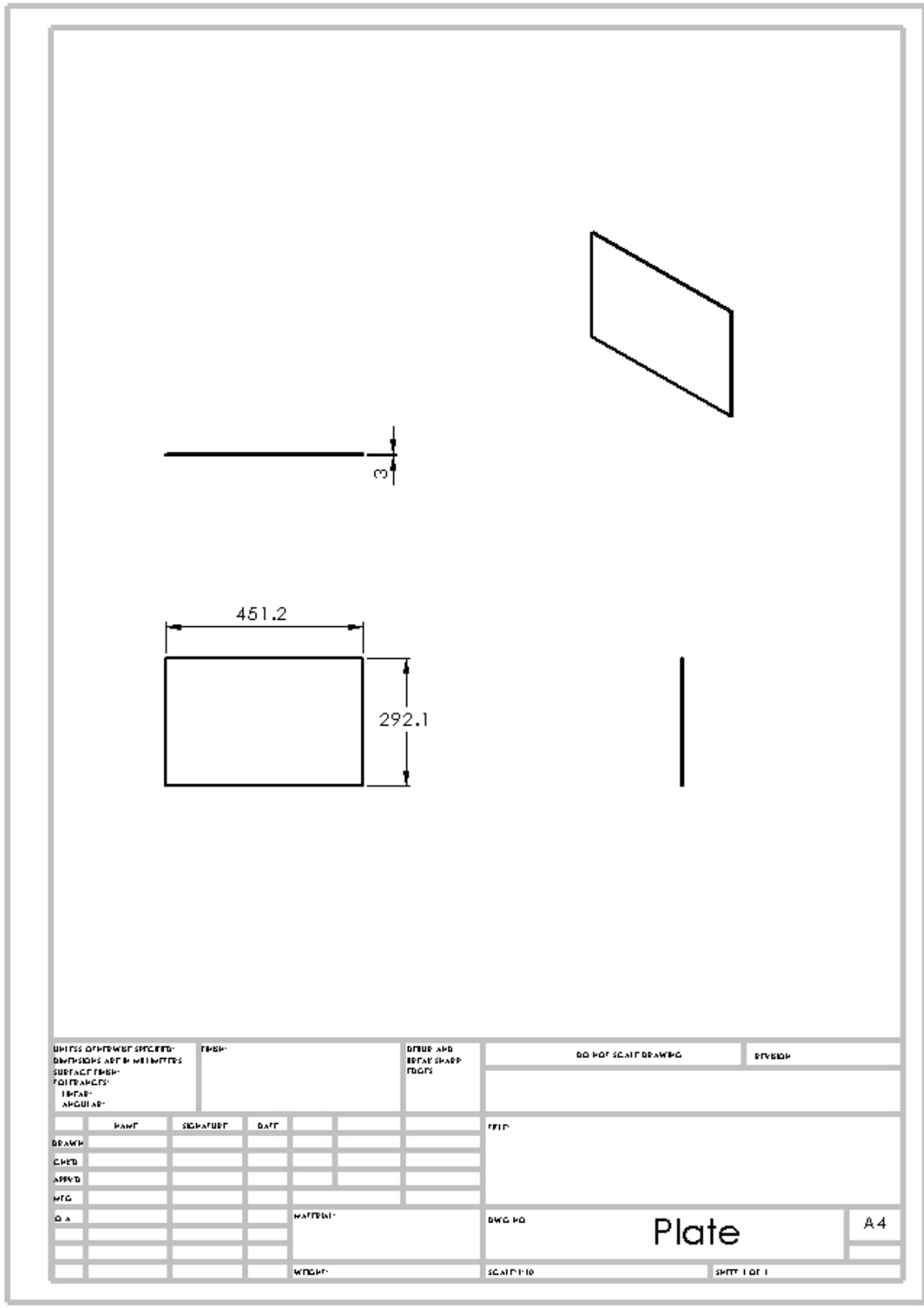


| | | | | | | | | | | |
|--------------------------------------------------------------------------------------------------------------------------|--|--|-----------|--|-----------------------------------|--|----------------------|--|--------------|--|
| UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACES UNLESS NOTED OTHERWISE: FINISH: ANGLE: | | | TOLERANCE | | OTHER AND REFER SHARP EDGES | | DO NOT SCALE DRAWING | | REVISION | |
| DRAWN | | | SIGNATURE | | DATE | | PLOT | | | |
| CHECKED | | | | | | | | | | |
| APPROVED | | | | | | | | | | |
| MFC | | | | | | | | | | |
| D.A. | | | | | MATERIAL | | DWG NO | | A4 | |
| | | | | | | | SCAFF | | SHEET 1 OF 1 | |
| | | | | | | | | | | |

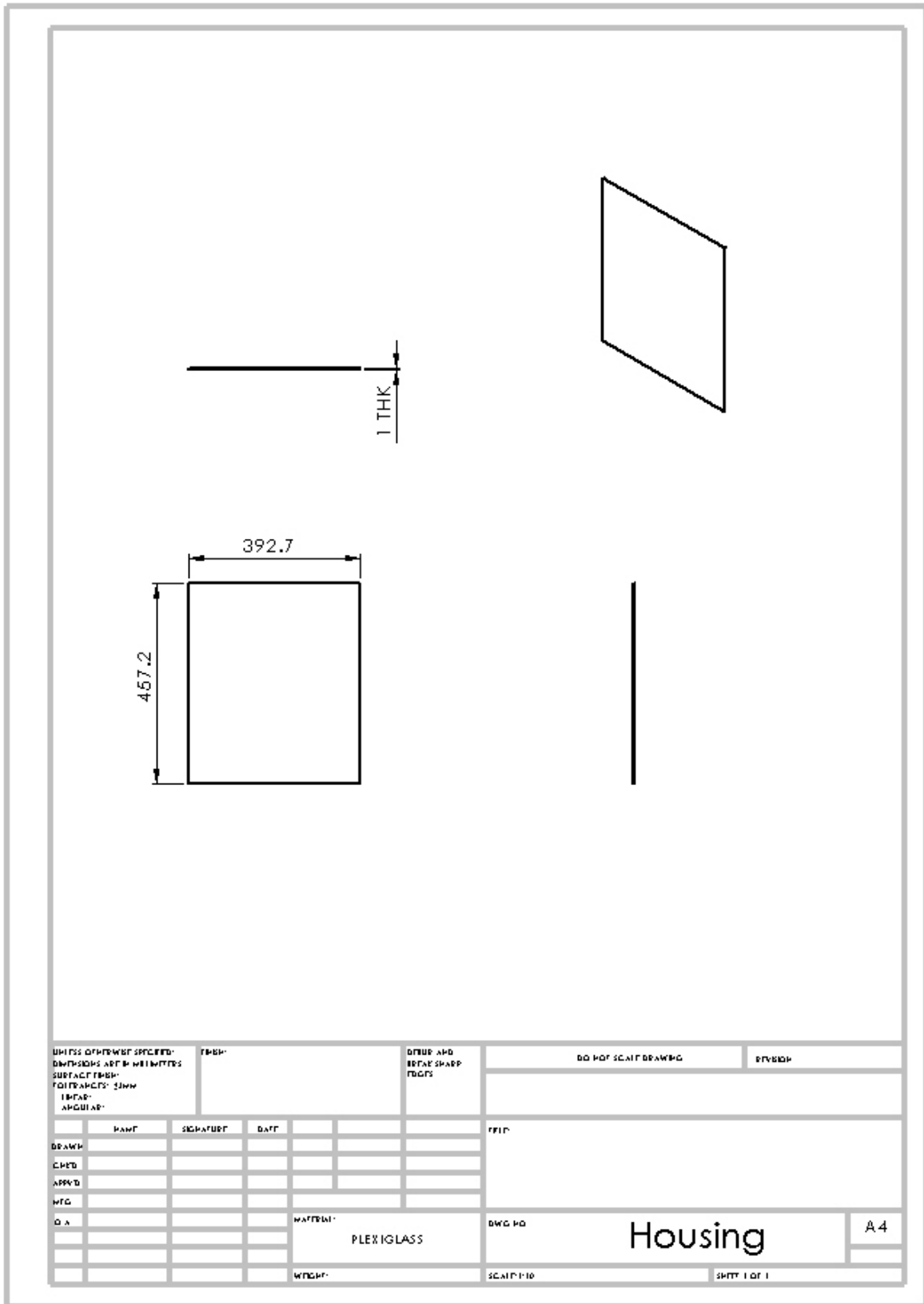
Guide Wheel



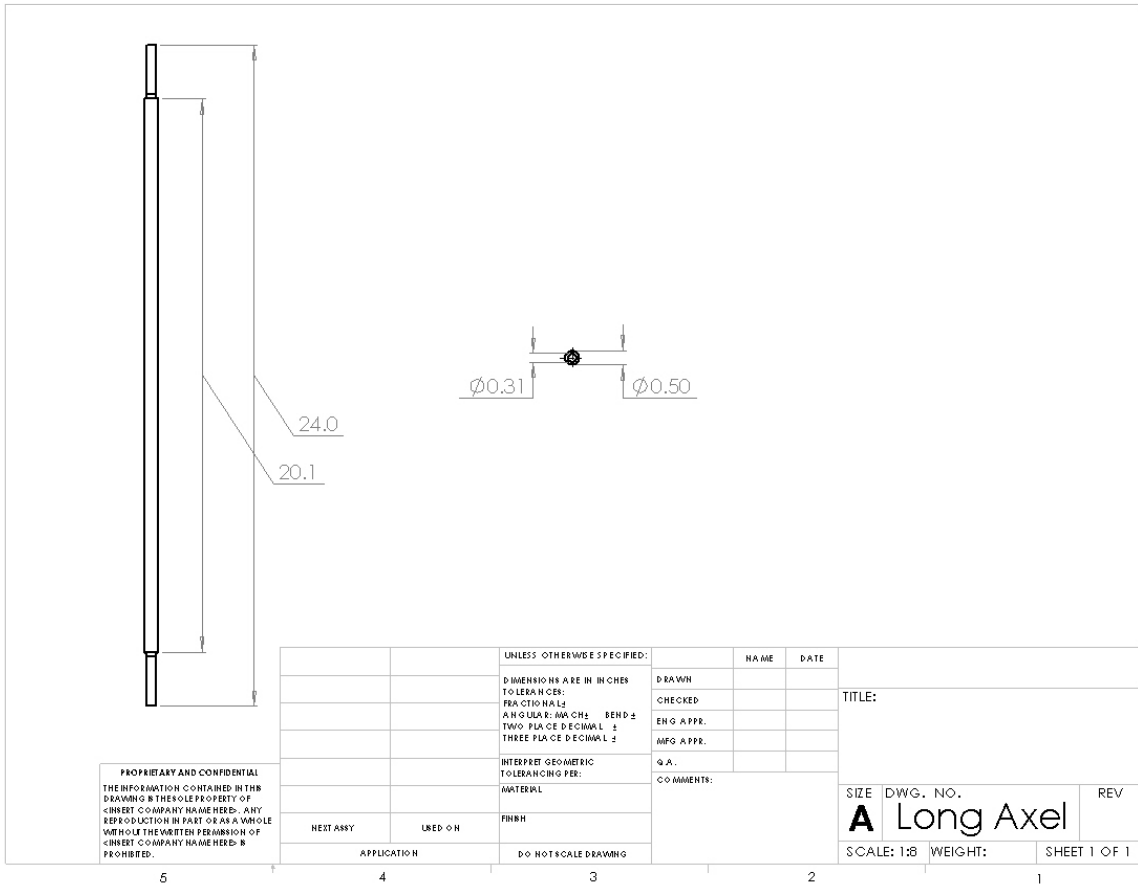




| | | | | | | | | | |
|--------------------------------|--|------------|--|---------------------------------|--|----------------------|--|--------------|--|
| UNITS OF DIMENSIONS SPECIFIED: | | TOLERANCE: | | OTHER AND SPECIAL REQUIREMENTS: | | DO NOT SCALE DRAWING | | STANDARD | |
| DIMENSIONS ARE IN MILLIMETERS | | | | | | | | | |
| SURFACE FINISH: | | | | | | | | | |
| CORNER RADIUS: | | | | | | | | | |
| TYPICAL: | | | | | | | | | |
| ANGULAR: | | | | | | | | | |
| | | | | | | | | | |
| DRAWN | | SIGNATURE | | DATE | | | | TITLE | |
| CHECKED | | | | | | | | | |
| APPROVED | | | | | | | | | |
| MFG | | | | | | | | | |
| D.A. | | | | MATERIAL | | DWG NO | | Plate | |
| | | | | | | | | A4 | |
| | | | | WEIGHT | | SCALE 1:10 | | SHEET 1 OF 1 | |



| | | | | | | | | | |
|---------------------------------------------------------------------------------------------------------------------------|--|-----------|--|----------------------------------|--|----------------------|--|--------------|--|
| UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: FOLIARACTS: 32MM TOLER: ANGULAR: | | FINISH: | | DITRE AND EDGE SHARP EDGES | | DO NOT SCALE DRAWING | | REVISION | |
| NAME | | SIGNATURE | | DATE | | FILED | | | |
| DRAWN | | | | | | | | | |
| CHECKED | | | | | | | | | |
| APPROVED | | | | | | | | | |
| MFG | | | | | | | | | |
| D.A. | | | | MATERIAL: | | DWG NO | | A 4 | |
| | | | | PLEXIGLASS | | Housing | | | |
| | | | | WEIGHT: | | SCALE: 1:10 | | SHEET 1 OF 1 | |



PROPRIETARY AND CONFIDENTIAL
 THE INFORMATION CONTAINED IN THIS
 DRAWING IS THE SOLE PROPERTY OF
 [COMPANY NAME HERE]. ANY
 REPRODUCTION IN PART OR AS A WHOLE
 WITHOUT THE WRITTEN PERMISSION OF
 [COMPANY NAME HERE] IS
 PROHIBITED.

| | | | | | | |
|--|--|-----------------------------------------|-----------|------|------|---------------------------------|
| | | UNLESS OTHERWISE SPECIFIED: | | NAME | DATE | |
| | | DIMENSIONS ARE IN INCHES TOLERANCES: | DRAWN | | | TITLE: |
| | | FRACTIONALS | CHECKED | | | |
| | | ANGLES: MAXIMUM BEND ± | ENG APPR. | | | |
| | | TWO PLACE DECIMAL ± | MFG APPR. | | | |
| | | THREE PLACE DECIMAL ± | Q.A. | | | SIZE DWG. NO. REV |
| | | INTERPRET GEOMETRIC TOLERANCING PER: | COMMENTS: | | | |
| | | MATERIAL | | | | SCALE: 1:8 WEIGHT: SHEET 1 OF 1 |
| | | FINISH | | | | |
| | | APPLICATION | | | | |
| | | USED ON | | | | |

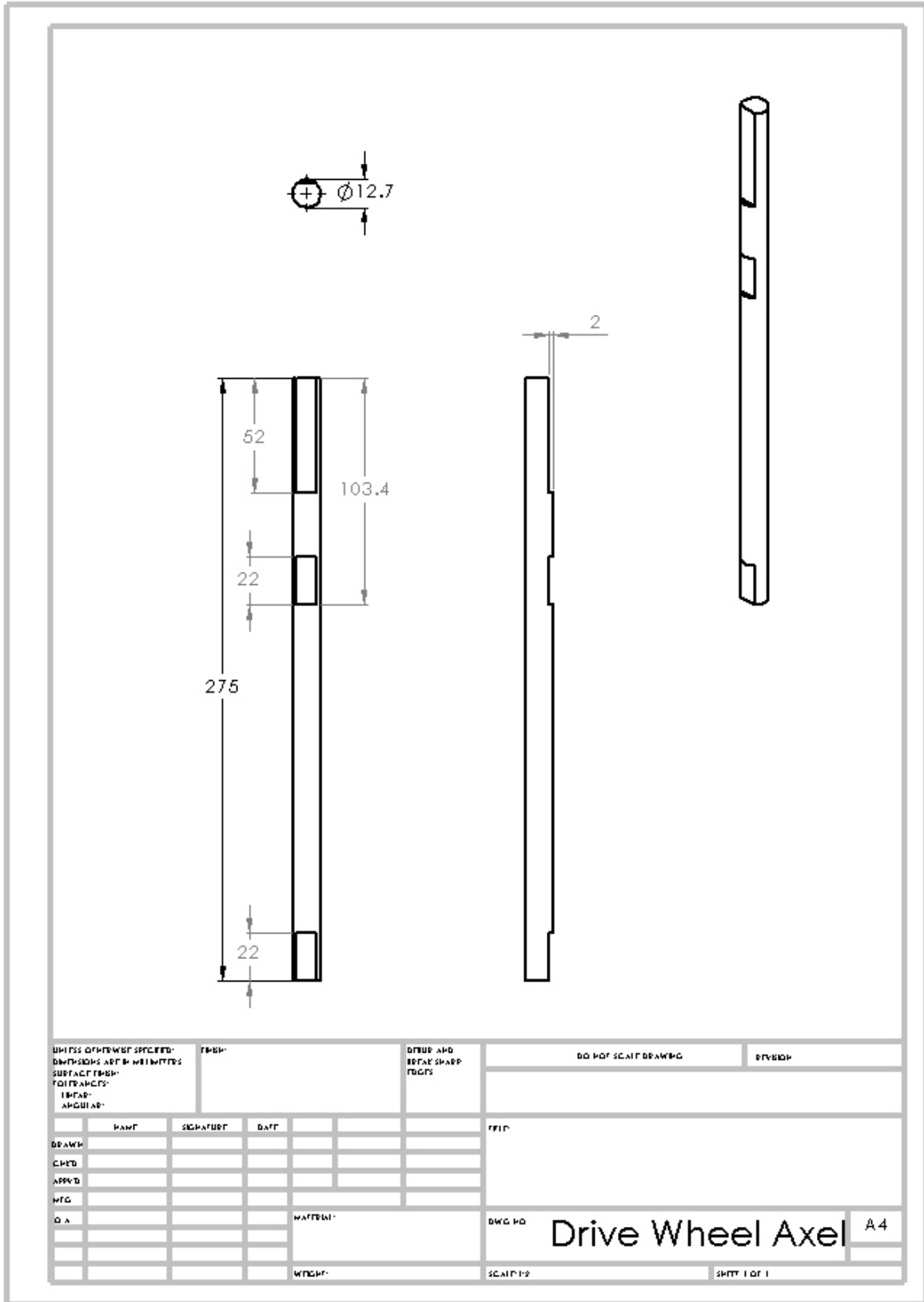
5

4

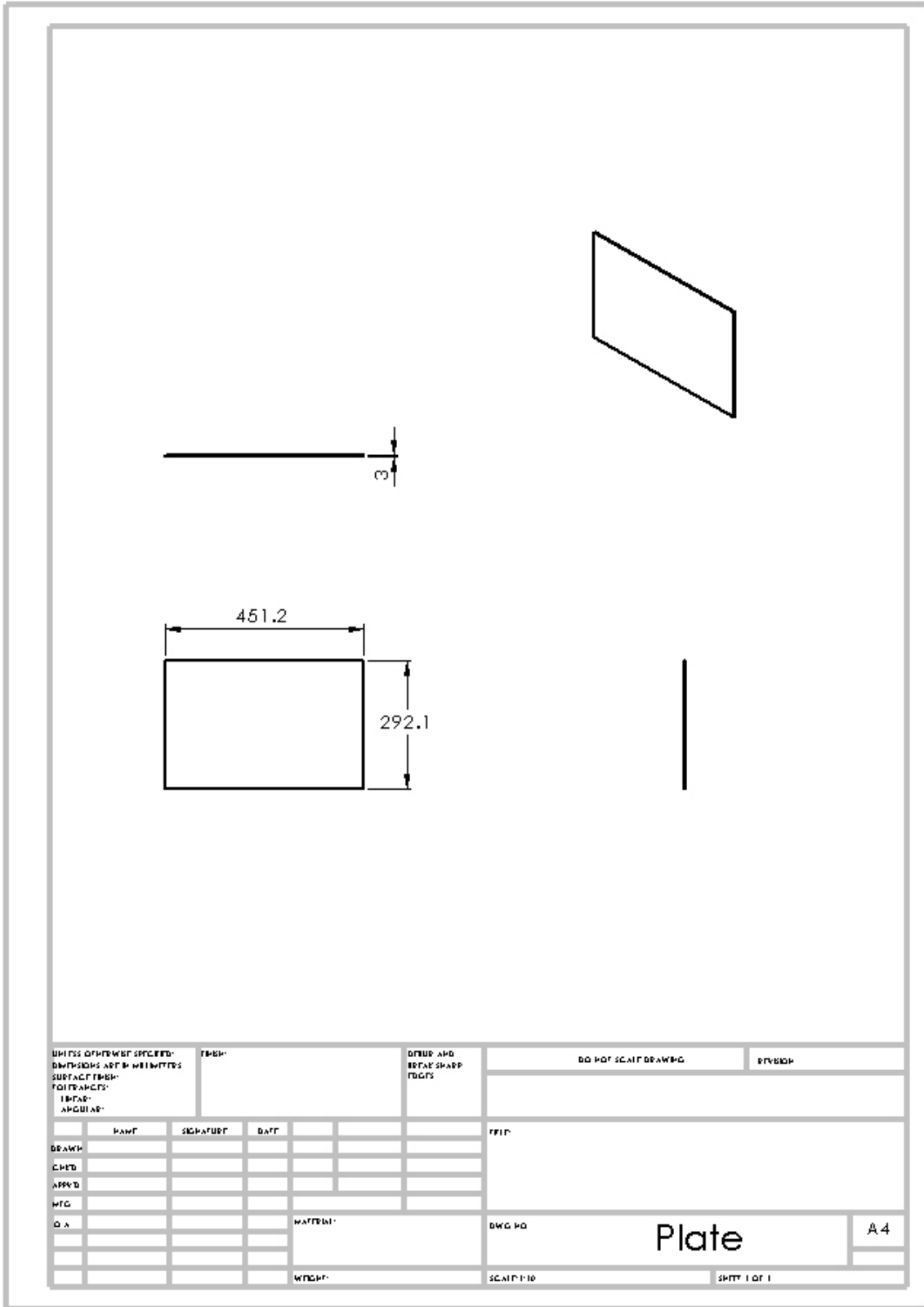
3

2

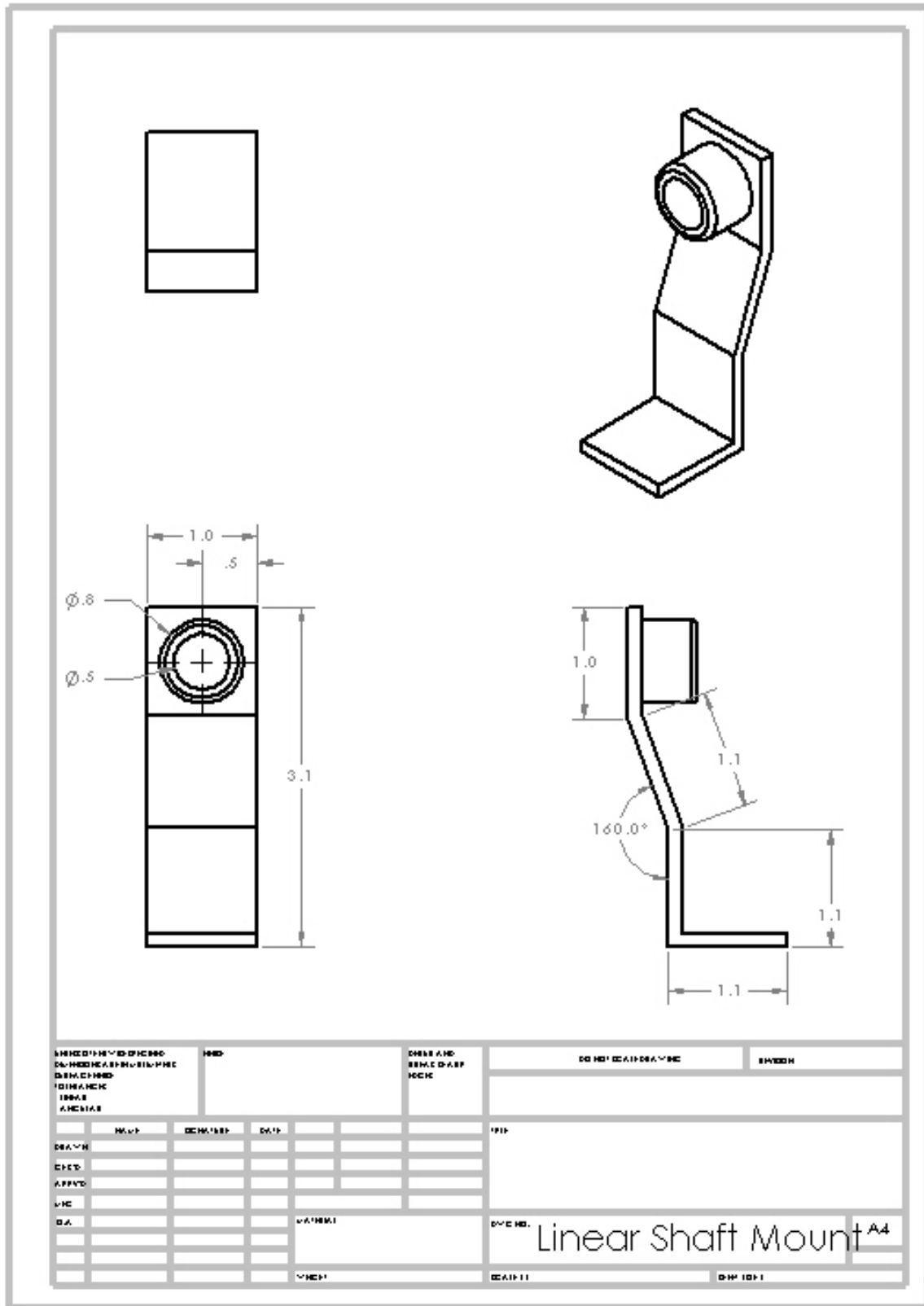
1



| | | | | | | | | | |
|------------------------------------------------------------------------------------------------------------------------------|--|-------------|--|---------------------------------|--|----------------------|--|----------------------|--|
| UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: F01 FINISH: F12 FINISH: F04 FINISH: | | TOLERANCES: | | DIMS AND EDGE SHARP EDGES | | DO NOT SCALE DRAWING | | REVISED | |
| NAME | | SIGNATURE | | DATE | | FILED | | | |
| DRAWN | | | | | | | | | |
| CHECKED | | | | | | | | | |
| APPROVED | | | | | | | | | |
| MFG | | | | | | | | | |
| D.A. | | | | MATERIAL | | DWG NO | | Drive Wheel Axle A 4 | |
| | | | | WEIGHT | | SCALE 1:2 | | SHEET 1 OF 1 | |



| | | | | | | | | | |
|--------------------------------------------------------------------------------------------------------------------|--|-----------|--|------------------------------------|--|----------------------|--|--------------|--|
| UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: FIT: ANGULAR: | | FINISH: | | OTHER AND REFER SHARP EDGES: | | DO NOT SCALE DRAWING | | STANDARD | |
| DRAWN | | SIGNATURE | | DATE | | TITLE | | | |
| CHECKED | | | | | | | | | |
| APPROVED | | | | | | | | | |
| MFG | | | | | | | | | |
| D.A. | | | | MATERIAL | | DWG NO | | Plate | |
| | | | | | | | | A4 | |
| | | | | WEIGHT | | SCALE 1:10 | | SHEET 1 OF 1 | |



| | | | | | |
|-----------------------------------------------------------------------------------------------|----------|----------|------------------------------|--------------------------------------|------------|
| SHREEDHAR ENGINEERING 24, HINDI ABBAY, SRI RANGI DISTRICT TIRUPATI ANDHRA PRADESH | | HNO. | DATE AND TIME OF ISSUE | DO NOT WRITE OVER THIS | SHEET NO. |
| DRAWN | DESIGNED | DATE | TIME | Dwg No. Linear Shaft Mount A4 | |
| CHECKED | | | | | |
| APPROVED | | | | | |
| V.P.C. | | | | | |
| D.A. | | MATERIAL | | | |
| | | WEIGHT | SCALE | | DIMENSIONS |

B.2 Specifications

Chassis Assembly Parts

Figure B.2.1: Long Axle Collar

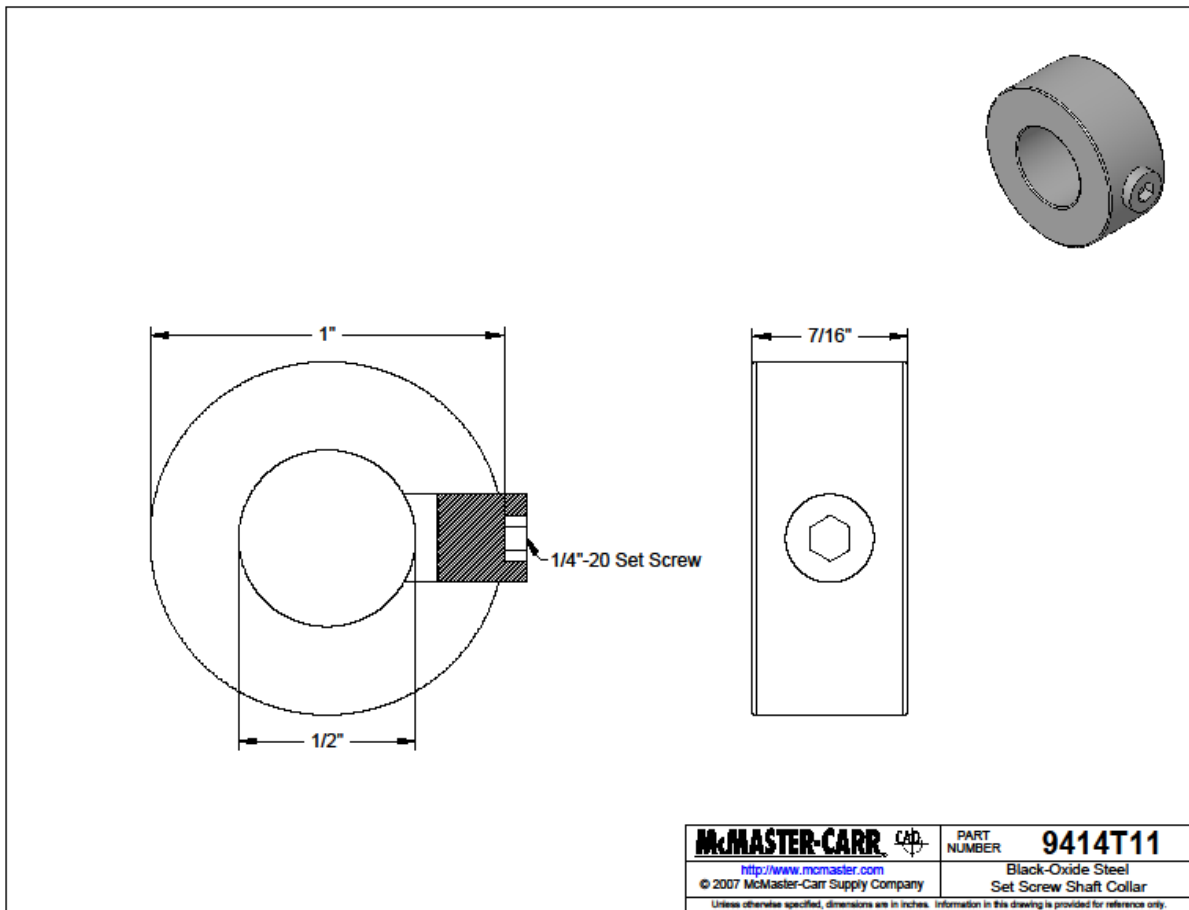
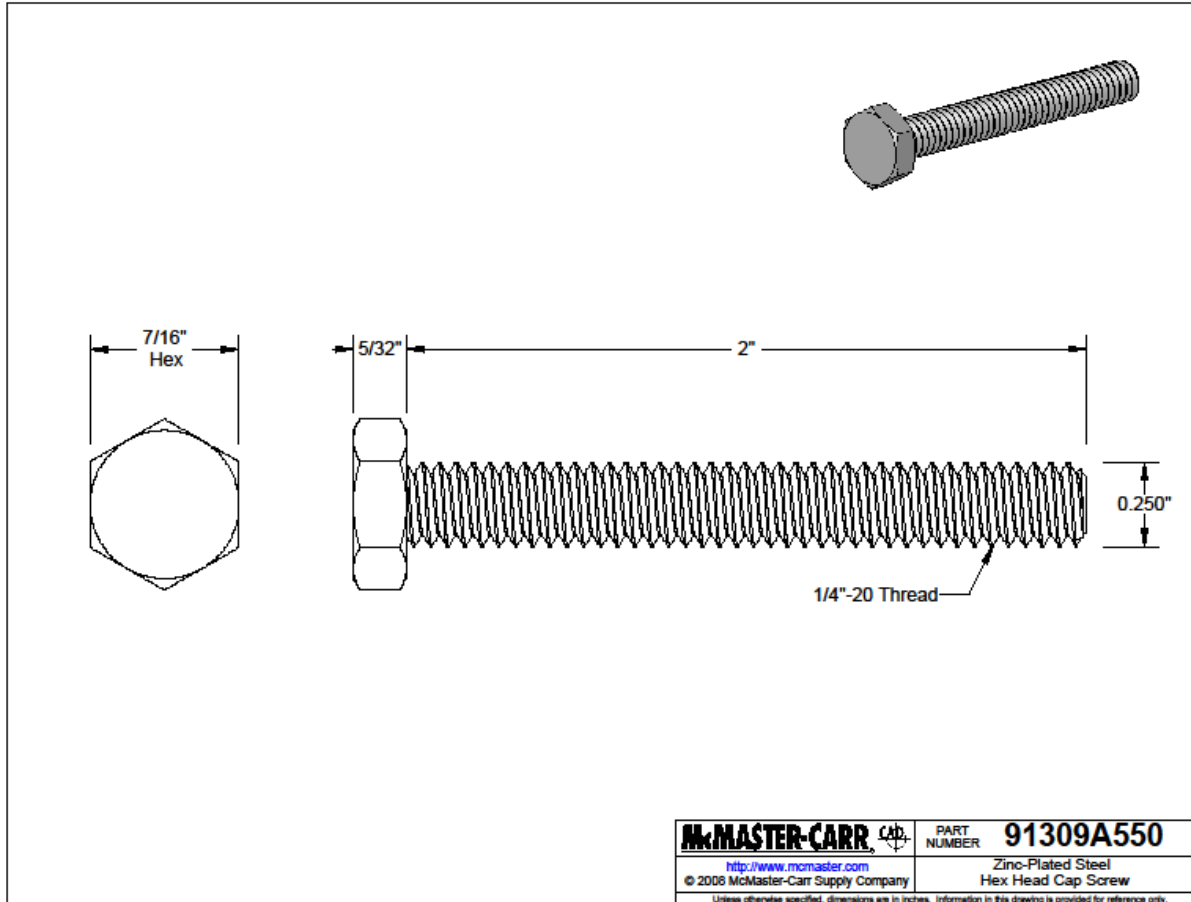


Figure B.2.2: Bolt



Leveling Assembly Parts

Figure B.2.3: Motor

29:1 Metal Gearmotor 37Dx52L mm

Pololu item #: 1103 23 in stock

| Price break | Unit price (US\$) |
|-------------|-------------------|
| 1 | 24.95 |
| 10 | 21.20 |
| 50 | 18.66 |

Quantity: Add to cart
backorders allowed Add to wish list

www.pololu.com

This 2.05" x 1.45" x 1.45" gearmotor is a powerful brushed DC motor with **29:1** metal gearbox intended for operation at 12 V. These units have a 0.61"-long, 6 mm-diameter D-shaped output shaft. This gearmotor is also available [with an integrated encoder](#).

Key specs at **12 V**: 350 RPM and 300 mA free-run, 110 oz-in (8 kg-cm) and 5 A stall.

Select options: [any gear ratio] Go

[Compare all products in 37D mm Gearmotors](#)

Description **Specifications (12)** **Pictures (6)** Resources (0) FAQs (0)

Gearmotor Options

This powerful brushed DC gearmotor is available in six different gear ratios. A version with an integrated encoder is also available.

| Gear Ratio | Speed @ 12V | Stall Torque @ 12V | Stall Current @ 12V | www.pololu.com | |
|------------|-------------|--------------------|---------------------|----------------------------|----------------------------|
| | | | | With Encoder | Without Encoder |
| 19:1 | 500 RPM | 84 oz-in | 5 A | 37Dx52L mm | 37Dx52L mm |
| 29:1 | 350 RPM | 110 oz-in | 5 A | 37Dx52L mm | 37Dx52L mm |
| 50:1 | 200 RPM | 170 oz-in | 5 A | 37Dx54L mm | 37Dx54L mm |
| 67:1 | 150 RPM | 200 oz-in | 5 A | 37Dx54L mm | 37Dx54L mm |
| 100:1 | 100 RPM | 220 oz-in | 5 A | 37Dx57L mm | 37Dx57L mm |
| 131:1 | 80 RPM | 250 oz-in | 5 A | 37Dx57L mm | 37Dx57L mm |

Figure B.2.4: Motor Dimensions

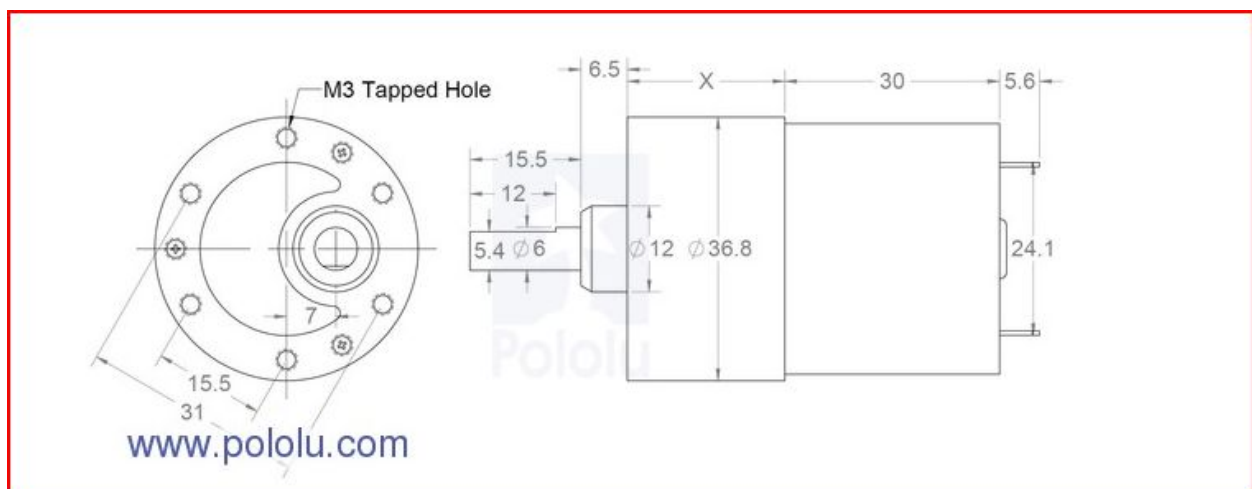


Figure B.2.6: Gear

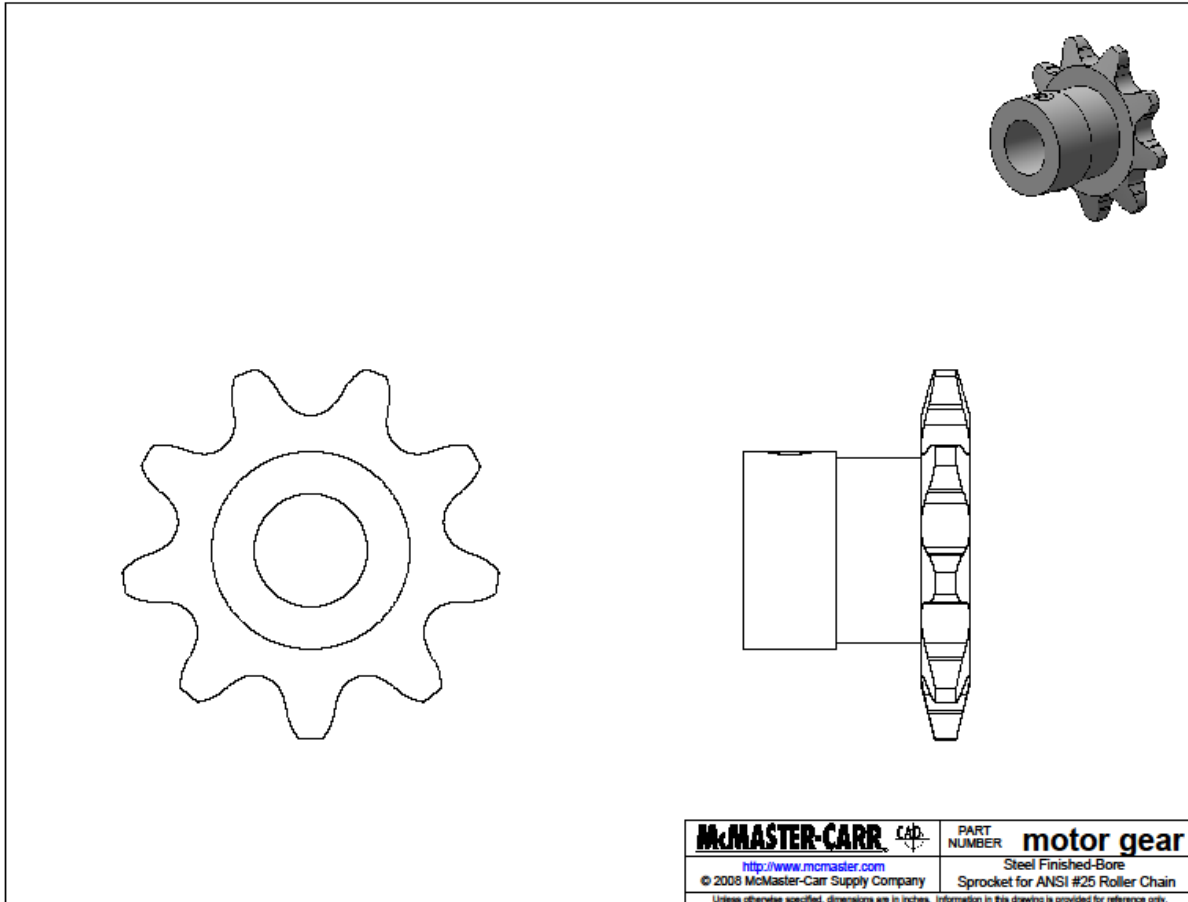


Figure B.2.7: Sprocket

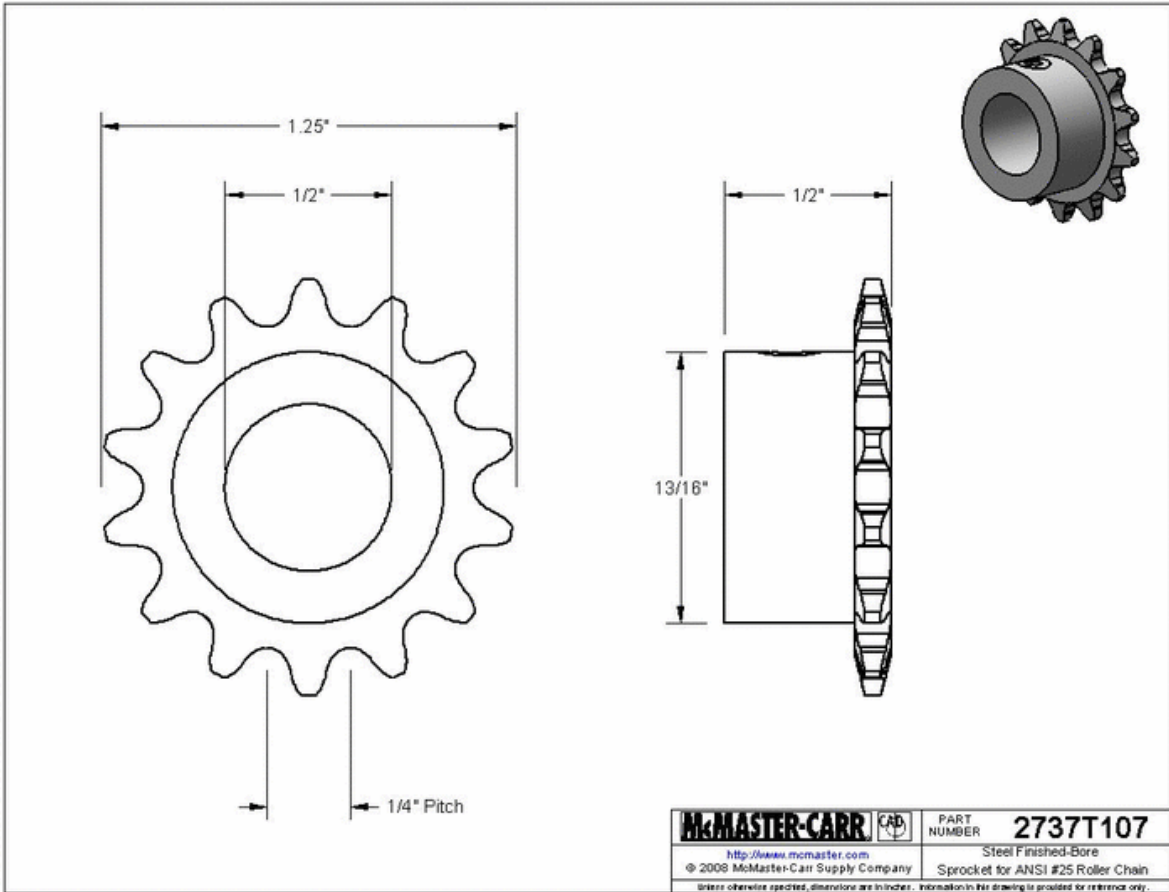


Figure B.2.8: Rail Mount

Wheels and Drivetrain Assembly Parts

Figure B.2.9: 250W Motor

250W Motor - 24 Volts (Style: MY1016)



Price: \$32.00
Availability: In Stock
Model: MOT-106100
Average Rating: Not Rated

Qty: [Add to Cart](#)

[Click to enlarge](#)

| Description | Additional Images (0) | Reviews (0) | Related Products (2) |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|-------------|----------------------|
| MY1016 24 Volt, 250 Watt, 2650 RPM, 13.7 Amp, permanent-magnet motor. <ul style="list-style-type: none">• 11 tooth sprocket for #25 chain.• 12 gauge power leads with standard 1/4" push-in connectors. Mounting bracket measures 4-1/2" x 2-1/8" with 4 threaded mounting holes. | | | |

Figure B.2.10: 250W Motor Specs.

| Basic Info. | | | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|--------------|--------------|--------------|
| Model NO.: MY1016 | | | | |
| Export Markets: Global | | | | |
| Additional Info. | | | | |
| Packing: Carton | | | | |
| Origin: China Mainland | | | | |
| Production Capacity: 100000PCS/Month | | | | |
| Product Description | | | | |
| UNITE Motor ElectricBike Motor Electric Scooter Motor Brush Motor | | | | |
| MODEL MY1016 | | | | |
| Permanent Magnetism Direct-Current (DC) Motor | | | | |
| Suitable for 24V250W Power | | | | |
| We provide all kinds of ATV Quad Bike Parts, Dirt Bike Parts, Electric Bike Parts, Electric Scooter Parts and etc. | | | | |
| Electric Bike Part including Electric Motor, Electric Controller, Electric Charger, Battery and etc. | | | | |
| ATV And Dirt Bike part Including Complete Motorcycle Engines (Lifan Engines Yingxiang Engines Zongshen Engines Loncin Engines, Motorcycle Plastic Kits, Motorcycle Carburetors, Motorcycle Tyres, Motorcycle Rims, Plastics and Oil tank, Motor Bike Seats, Electric Bike Chains, Motorcycle Chains, Motorcycle Sticker Decal, Absorb Spring, Brake Disc, Handlebars, Motorcycle CNC part, Air filters, Bearings, Assembly Switches, Headlight, Tail light, Clothes and Apparences, Motorcycle Electric Part (CDI, Ignition Coil, Regulator, Relay)and etc. | | | | |
| We appreciate quantity order and we also accept small orders, fast delivery by air and other ways shipping. | | | | |
| Accept OEM orders | | | | |
| Any other parts you after please contact | | | | |
| Model | MY 1016 | | | |
| Standard | 200W 24V/36V | 250W 24V/36V | 300W 24V/36V | 350W 24V/36V |
| No-load current/A | ≤1.5/1.0 | ≤1.6/1.2 | ≤1.8/1.4 | ≤2.0/1.4 |
| No-load rate speed /rpm | 3300 | 3350 | 3400 | 3450 |
| Rating Torque/N·m | 0.70 | 0.90 | 1.04 | 1.22 |
| Rating speed /rpm | 2700 | 2750 | 2750 | 2750 |
| Rating current/A | ≤10.6/7.1 | ≤13.4/8.9 | ≤16.0/10.7 | ≤18.7/12.5 |
| Efficiency/% | ≥76 | ≥78 | ≥78 | ≥78 |
| Use | Electric Scooter / Small Scooter | | | |

Figure B.2.11: Drive Wheel

> Material Handling > Rollers > Shaft Drive > DuraSoft

1/2 in ID

5/8 in ID

3/4 in ID

1 in ID

DuraSoft® Shaft Drive Rollers - Steel Insert - 1/2 in I.D. - Inch

To activate, select the **CAD** button for a part number in the table below

Confirm your selection with the part number & description directly below the line drawing

To download or view select: 'CAD Download' or '3D ON'

DR-9754-20-EX500:DURASOFT® ROLLER - 1/2 IN SHAFT DRIVE - 20 DUR NEOPRENE - 4.00 IN DIA X 0.92 IN WIDE

[CAD DOWNLOAD](#)

[PRINT PAGE](#)

PAN ZOOM ROTATE

HYPERVIEW 3D OFF 3D ON

PRESS ANY KEY TO EXIT HYPERVIEW

| | | | | | | | | |
|-----------------------------------------------------------|------|------|-----------|------|-----|----------|----|---------------------|
| Buy DR-502-35UR-EX500 | 2.50 | 1.94 | .501/.505 | 1.25 | .50 | Urethane | 35 | CAD |
| Buy DR-502-60UR-EX500 | 2.50 | 1.94 | .501/.505 | 1.25 | .50 | Urethane | 60 | CAD |
| Buy DR-502-80UR-EX500 | 2.50 | 1.94 | .501/.505 | 1.25 | .50 | Urethane | 80 | CAD |
| Buy DR-502-95UR-EX500 | 2.50 | 1.94 | .501/.505 | 1.25 | .50 | Urethane | 95 | CAD |
| Buy DR-9754-20-EX500 | 4.00 | .92 | .501/.505 | 1.25 | .50 | Neoprene | 20 | CAD |
| Buy DR-9754-35-EX500 | 4.00 | .92 | .501/.505 | 1.25 | .50 | Neoprene | 35 | CAD |
| Buy DR-9754-20W-EX500 | 4.00 | .92 | .501/.505 | 1.25 | .50 | Nitrile | 20 | CAD |
| Buy DR-9754-35W-EX500 | 4.00 | .92 | .501/.505 | 1.25 | .50 | Nitrile | 35 | CAD |
| Buy DR-9754-35UR-EX500 | 4.00 | .92 | .501/.505 | 1.25 | .50 | Urethane | 35 | CAD |
| Buy DR-9754-60UR-EX500 | 4.00 | .92 | .501/.505 | 1.25 | .50 | Urethane | 60 | CAD |

Figure B.2.12: Drive Wheel Gea

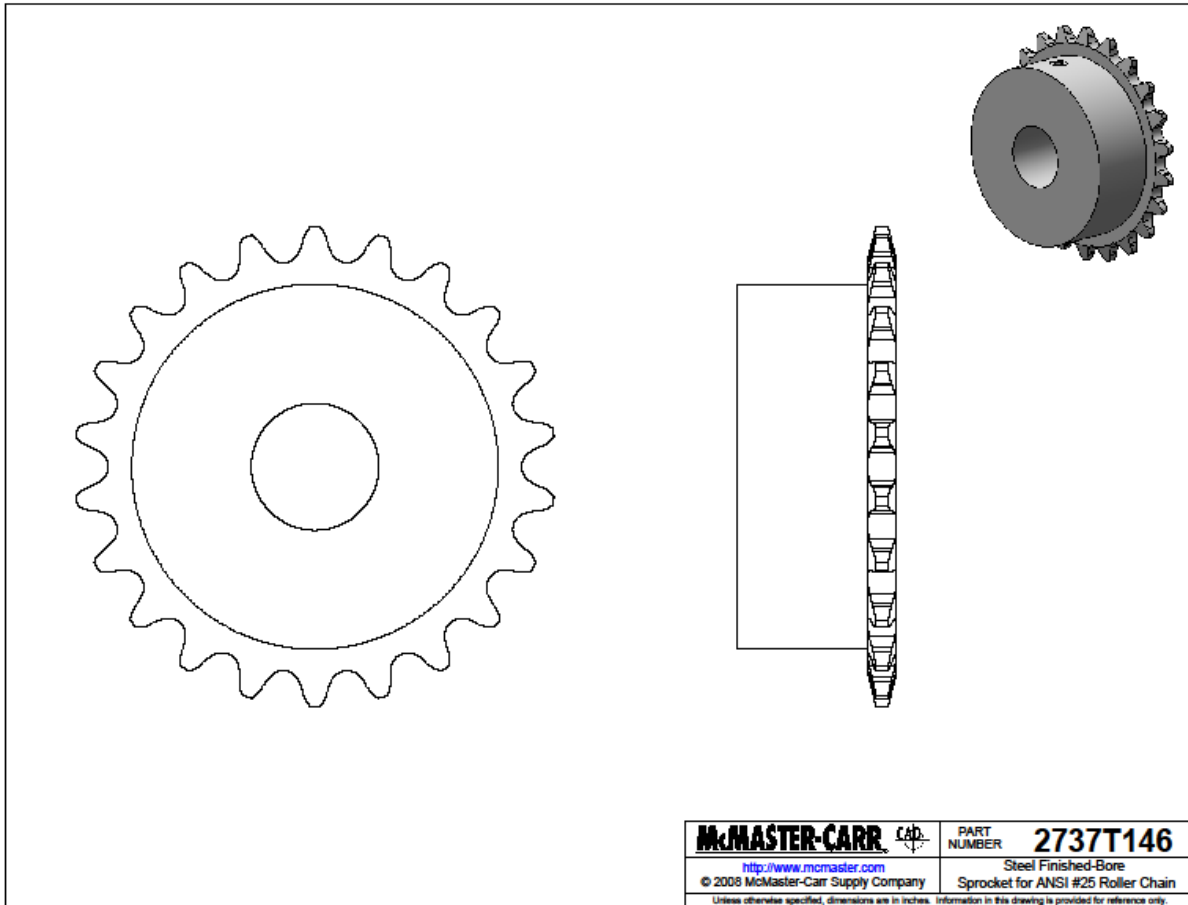


Figure B.2.13: Battery

E-Scooter 24V System Battery Set



Our Price: \$43.80

Qty:

1

ADD TO C

Availability: Ships in 24 Hours

f Like 0

Description

| Model | Terminal | Volts | AH | Length | Width | Height | Qty | Weight (Lbs) |
|------------|----------|-------|-------|--------|-------|--------|-----|--------------|
| 24V System | F2 | 12.00 | 12.00 | 5.95 | 3.86 | 3.70 | 2 | 15.84 |

The brand new replacement battery set for the E-Scooter 24V System includes 2 batteries - NB12-12 (12 Volts 12.0 AH). You must use the existing cables and hardware to connect your replacement electric scooter batteries.

The E-Scooter 24V System replacement batteries have the following characteristics:

- High Rate Discharge Design
- Performance Guaranteed
- Brand New and Factory Fresh
- In Stock and Same Day Shipping
- Professional Customer Support
- 1 Year Warranty

If you need more help with purchasing E-Scooter electric scooter replacement batteries, please call our customer service department at 1 (800) 657-1303 or [email us](#).

Our staff consists of experienced and knowledgeable battery specialists. We have extensive resources to provide you with any electric scooter replacement battery.

If you want to set up a corporate or government account, feel free to [email us](#). For larger quotes please fill out a [Request for Quote \(RFQ\) form](#). Since we have access to different suppliers it is particularly useful to provide us with the type of application to better match your performance requirements.

B.3 Parts List and Budget

| Part Description | Part Name | Price (\$) | Weight (kg) |
|--------------------------------|------------------------------------------------------------------------------------------|------------|-------------|
| Drive Motor (X2) | 250W Motor - 24 Volts (Style: MY1016) | \$100.15 | 3 |
| Drive Wheel Gear (X2) | Steel Finished-Bore Roller Chain Sprocket for #25 Chain, 1/4" Pitch, 22 Teeth, 1/2" Bore | \$15.14 | 0.239 |
| Multipurpose Chain (6ft) | Standard ANSI Roller Chain #25, Single Strand, 1/4" Pitch, .13" Dia | \$22.38 | 0.2617 |
| Drive Wheel | Solid Roller - 1/2 in Shaft Drive - 60 Dur Neoprene - 4.00 in Dia X 0.92 in Wide | \$103.16 | 0.7385 |
| Battery | Ezip Scooter eZip 900 Battery Set | \$61.68 | 7.7 |
| Battery Charger | 12 Volt 2 Amp Battery Charger | \$20.87 | |
| Drive Wheel Axel | 1/2 inch Dia. 1018 Cold Finish Steel Round | \$0.00 | |
| Drive Wheel Axel Bearings (X4) | Stamped-Steel Mounted Ball Bearing--ABEC-1 2-Bolt Base Mount, for 1/2" Shaft Diameter | \$43.80 | 0.7252 |
| | | | |
| Tensioner Bolt (X2) | 1/4-20 Bolt - Tensioner | \$8.00 | 0.0655 |
| Top Long Axel Collars | Black-Oxide Steel Set Screw Shaft Collar 1/2" Bore, 1" Outside Diameter, 7/16" Width | \$1.68 | 0.0632 |
| Guide Wheel (X6) | 1 ft Delron Stock 2.5" Diameter | \$25.00 | 1 |
| Rim (X2) | sun cr18 | \$50.00 | 2 |
| Tire (X2) | 26 x 1.5 Amerityre Solid Rubber | \$100.20 | 1 |
| | | | |
| Idle Axel | 3/8 inch Dia. 1018 Cold Finish Steel Round, 2ft length | \$2.24 | 0.25 |
| Idle Axel Collar (X4) | Black-Oxide Steel Set Screw Shaft Collar 3/8" Bore, 3/4" Outside Diameter, 3/8" Width | \$2.88 | 0.0592 |
| Idler (X2) | Steel Finished-Bore Roller Chain Sprocket for #25 Chain, 1/4" Pitch, 14 Teeth, 1/2" Bore | \$12.12 | 0.0476 |
| Idle Bearings (X4) | SAE 841 Bronze Flanged-Sleeve Bearing for 3/8" Shaft Diameter, 1/2" OD, 1/4" Length | \$2.80 | 0.0142 |
| Rails (X2) | Hardened Precision Steel Shaft 3/8" Diameter, 12" Length | \$12.12 | 0.3424 |
| Linear Bearing (X2) | SAE 841 Bronze Flanged-Sleeve Bearing for 3/8" Shaft Diameter, 5/8" OD, 1" Length | \$3.88 | 0.0486 |
| Leveling Motor | 29:1 Metal Gearmotor 37Dx52L mm | \$24.95 | 1 |
| Leveling Motor | 19:1 Metal Gearmotor 37Dx52L mm with encoder | \$40.91 | 1 |
| Leveling Motor Mount (1ft) | 1/8 X 1-1/2 Hot Rolled Steel Flat, 2ft length | \$3.06 | |
| Mass Plate | 1 x 1 Square Bar Hot Rolled A-36 Steel Square, 2ft length | \$8.84 | 1 |
| Gear motor sprocket | Steel Finished-Bore Roller Chain Sprocket for #25 Chain, 1/4" Pitch, 9 Teeth, 1/4" Bore | \$7.95 | 0.0098 |

| | | | |
|------------------------------------------------------|-----------------------------------------------------------|----------|------|
| Plate | .190 (3/16) thick 3003-H14 Aluminum Plate, 1ftx2ft | \$30.06 | 2 |
| Arms | 1/8 X 1 Hot Rolled Steel Flat, 6ft | \$8.80 | 1 |
| Plate Frame | 1/2 X 1/2 X 1/8 Steel Angle A-36 Steel Angle | \$10.92 | 1.5 |
| Multi Purpose Axel | Steel Drive Shaft 1/2" OD, 8ft length | \$16.08 | 1 |
| Housing (X2) | Optically Clear Cast Acrylic Sheet 1/16" Thick, 24" X 24" | \$30.06 | 1.49 |
| Hardware Table Legs | Machined | \$0.00 | 0.1 |
| Hardware Table Legs | Machined | \$0.00 | 0.1 |
| Hardware Table Legs | Machined | \$0.00 | 0.1 |
| Hardware Table Legs | Machined | \$0.00 | 0.1 |
| Hardware plate | Machined | \$0.00 | 0.1 |
| Arduino/Breadboard | NA | \$31.99 | |
| Wireless RF USB Dongle | RFD21807 | \$75.00 | |
| Wireless Inventor Shield | RFD21815 | \$30.00 | |
| Pololu High-Power Motor Driver 24v23 CS | 1456 | \$125.90 | |
| 10A DC Motor Driver Arduino Shield | RB-Cyt-116 | \$19.06 | |
| Gardner Bender 20 Amp Single-Pole Toggle Switch | GSW-11 | \$7.94 | |
| Fuse Holder | | \$5.00 | |
| 60 Piece AGC Glass Automotive Fuse Set | 67962 | \$5.99 | |
| H-Bridge Motor Driver 1A | COM-00315 | \$2.35 | |
| Double BTS Motor Driver | BTS75960B | \$40.00 | |
| Triple Axis Accelerometer & Gyro Breakout - MPU-6050 | SEN-11028 | \$39.99 | |

| | | |
|---------------|-------------------|--------------|
| TOTAL: | \$1,152.95 | 28.05 |
|---------------|-------------------|--------------|

C. Appendix C: Arduino Sketch

The Arduino Sketches are what control the entire Diwheel and its movement. Several revisions were made and the two majorly changed codes are located in this Appendix. The first was tested and functioned as desired. The second had revisions made to it that have not been tested due to the system failure during testing.

C.1 Initial Diwheel Program

```
//*****Senior Capstone Design Diwheel Program 2.0*****//
/*
```

Author: Christopher Parisi
 Organization: California Baptist University College of Engineering

Code has been tested and works for controlling the diwheel

Baud Rate: 9600 (Arduino Standard)
 Use either HyperTerminal or Arduino Serial Monitor for Control

CAUTION! When controlling Diwheel, be sure not to change motor direction instantaneously.
 Back EMF can harm the motor drivers and permanently break them.

Controls for Diwheel

```
w: forward
s: backward
a: left
d: right
q: spin counter-clock wise
e: spin clock wise
x: stop
g: start system
r: reset system
1: 10% Speed
2: 20% Speed
3: 30% Speed
4: 40% Speed
5: 50% Speed
6: 60% Speed
7: 70% Speed
8: 80% Speed
9: 90% Speed
0: 100% Speed
*/
```

//Arduino Pin Assignments

```
const int LM_DIR = 12;           //Direction control for the linear motor
const int LM_PWM = 3;           //PWM Signal for speed control of linear motor
const int LM_EncA = 2;          //Linear motor encoder A
const int LM_EncB = 5;          //Linear motor encoder A
const int M1_DIR = 6;           //Direction control for Motor 1
const int M2_DIR = 7;           //Direction control for Motor 2
const int M1_RESET = 8;         //Reset signal for Motor Driver 1
const int M2_RESET = 9;         //Reset signal for Motor Driver 2
const int M1_PWMH = 10;         //PWM signal for speed control of Motor 1
const int M2_PWMH = 11;        //PWM signal for speed control of Motor 2
```

//Other Variable Assignments

```
//volatile unsigned int encoderPos = 0;
int command = 0;                //User input from Serial Terminal
char dir = 'f';
int spd = 0;
char lm_dir = 'f';
```

```

char lm_pos = 'b';

void setup()
{
  //code section from from http://playground.arduino.cc/Main/RotaryEncoders#Example1
  pinMode(LM_EncA, INPUT);
  digitalWrite(LM_EncA, HIGH); // turn on pullup resistor
  pinMode(LM_EncB, INPUT);
  digitalWrite(LM_EncB, HIGH); // turn on pullup resistor
  //attachInterrupt(0, doEncoder, CHANGE); // encoder pin on interrupt 0 - pin 2

  Serial.begin(9600); //9600 Baud Rate

  Serial.println("Starting Up...");
  pinMode(LM_DIR, OUTPUT);
  pinMode(LM_PWM, OUTPUT);
  //pinMode(LM_EncA, INPUT);
  //pinMode(LM_EncB, INPUT);
  pinMode(M1_DIR, OUTPUT);
  pinMode(M2_DIR, OUTPUT);
  pinMode(M1_RESET, OUTPUT); //default HIGH
  pinMode(M2_RESET, OUTPUT); //default HIGH
  pinMode(M1_PWMH, OUTPUT);
  pinMode(M2_PWMH, OUTPUT);

  stopmotors();
  Serial.println("System Ready.");
}

void loop()
{
  /* Fault Flags removed due to lack of I/O Pins on Arduino
  if ((FF1Motor1 == LOW && FF2Motor1 == HIGH) || (FF1Motor2 == LOW &&
    FF2Motor2 == HIGH)) //Short Circuit
  {
    stopmotors();
    Serial.println("Short Circuit Detected. Stopping Diwheel");
  }
  else if((FF1Motor1 == HIGH && FF2Motor1 == LOW) || (FF1Motor2 == HIGH && FF2Motor2 == LOW))
  //Over Temperature
  {
    stopmotors();
    Serial.println("Over Temperature. Stopping Diwheel");
  }
  else if ((FF1Motor1 == HIGH && FF2Motor1 == HIGH) || (FF1Motor2 == HIGH && FF2Motor2 ==
  HIGH)) //Under Voltage
  {
    stopmotors();
    Serial.println("Under Voltage. Stopping Diwheel");
  }
  */

  if (Serial.available() > 0)
  {
    command = Serial.read();
  }
}

```



```

switch(command) {

case 'w': //forward
  if (dir == 'b' || dir == 's')
  {
    stopmotors();
    delay(1000);
  }
  digitalWrite(M1_DIR, HIGH); //Current flows from OUTA(+) to OUTB(-)
  digitalWrite(M2_DIR, HIGH); //Current flows from OUTA(+) to OUTB(-)
  dir = 'f';
  speedControl(dir, spd);
  break;

case 'a': //left
  dir = 'l';
  speedControl(dir, spd);
  break;

case 's': //backward
  if (dir == 'f' || dir == 's')
  {
    stopmotors();
    delay(1000);
  }
  digitalWrite(M1_DIR, LOW); //Current flows from OUTB(-) to OUTA(+)
  digitalWrite(M2_DIR, LOW); //Current flows from OUTB(-) to OUTA(+)
  dir = 'b';
  speedControl(dir, spd);
  break;

case 'd': //right
  dir = 'r';
  speedControl(dir, spd);
  break;

case 'e': //spin clockwise
  stopmotors();
  digitalWrite(M1_DIR, HIGH);
  digitalWrite(M2_DIR, LOW);
  dir = 's';
  break;
case 'q': //spin counter-clockwise
  stopmotors();
  digitalWrite(M1_DIR, LOW);
  digitalWrite(M2_DIR, HIGH);
  dir = 's';
  break;
case 'r': //Reset Motor Driver Circuits
  digitalWrite(M1_RESET, LOW); //Resets Motor 1 (Clears Fault Flags)
  digitalWrite(M2_RESET, LOW); //Resets Motor 2 (Clears Fault Flags)
  break;
case 'g': //Releases hold from RESET signal. Must use for initial start up
  digitalWrite(M1_RESET, HIGH);
  digitalWrite(M2_RESET, HIGH);
  break;

```

```
case 'x': //stop(coast)
  analogWrite(M1_PWMH, 0); //Turn the motor off by shorting it to GND
  analogWrite(M2_PWMH, 0); //Turn the motor off by shorting it to GND
  analogWrite(LM_PWM, 0); //Turn the motor off by shorting it to GND
  spd = 0;
  break;

case 'k': //Linear motor forward

  if (lm_dir == 'b')
  {
    stopmotors();
    delay(1000);
  }
  digitalWrite(LM_DIR, LOW);
  lm_dir = 'f';
  speedControl(dir, spd);

  break;

case 'l': //Linear motor backward

  if (lm_dir == 'f')
  {
    stopmotors();
    delay(1000);
  }
  digitalWrite(LM_DIR, HIGH);
  lm_dir = 'b';
  speedControl(dir, spd);

  break;

case '1':
  spd = 25; //10% power
  speedControl(dir,spd);
  break;
case '2':
  spd = 50; //20% power
  speedControl(dir,spd);
  break;
case '3':
  spd = 75; //30% power
  speedControl(dir,spd);
  break;
case '4':
  spd = 100; //40% power
  speedControl(dir,spd);
  break;
case '5':
  spd = 125; //50% power
  speedControl(dir,spd);
  break;
case '6':
  spd = 150; //60% power
```

```

    speedControl(dir,spd);
    break;
case '7':
    spd = 175;    //70% power
    speedControl(dir,spd);
    break;
case '8':
    spd = 200;    //80% power
    speedControl(dir,spd);
    break;
case '9':
    spd = 225;    //90% power
    speedControl(dir,spd);
    break;
case '0':
    spd = 255;    //100% power
    speedControl(dir,spd);
    break;
}
}
}

void speedControl(char dir, int x)
{
if (dir == 'f' || dir == 'b' || dir == 's')
{
    analogWrite(M1_PWMH, (x-0.1*x));    //10% reduced to counter drift
    analogWrite(M2_PWMH, x);
    if (dir == 'f' && lm_pos != 'b')
    {
        digitalWrite(LM_DIR, HIGH);
        analogWrite(LM_PWM, 255);
        delay(1600);
        analogWrite(LM_PWM, 0);
        lm_pos = 'b';
    }
    else if (dir = 'b' && lm_pos != 'f')
    {
        digitalWrite(LM_DIR, LOW);
        analogWrite(LM_PWM, 255);
        delay(1700);
        analogWrite(LM_PWM, 0);
        lm_pos = 'f';
    }
}
}
else if (dir == 'l')
{
    if (spd == 50 || spd == 75 || spd == 100)    //30% drop in speed in left wheel
    {
        analogWrite(M1_PWMH, (x-0.3*x));
        analogWrite(M2_PWMH, x);
    }
    else if (spd == 125 || spd == 150 || spd == 175) //20% drop in speed in left wheel
    {

```

```

    analogWrite(M1_PWMH, (x-0.2*x));
    analogWrite(M2_PWMH, x);
  }
  else if (spd == 200 || spd == 225 || spd == 255) //10% drop in speed in left wheel
  {
    analogWrite(M1_PWMH, (x-0.1*x));
    analogWrite(M2_PWMH, x);
  }
  else
  {
    analogWrite(M1_PWMH, x/2);
    analogWrite(M2_PWMH, x);
  }
}
else if (dir == 'r')
{
  if (spd == 50 || spd == 75 || spd == 100) //30% drop in speed in right wheel
  {
    analogWrite(M1_PWMH, x);
    analogWrite(M2_PWMH, (x-0.3*x));
  }
  else if (spd == 125 || spd == 150 || spd == 175) //20% drop in speed in right wheel
  {
    analogWrite(M1_PWMH, x);
    analogWrite(M2_PWMH, (x-0.2*x));
  }
  else if (spd == 200 || spd == 225 || spd == 255) //10% drop in speed in right wheel
  {
    analogWrite(M1_PWMH, x);
    analogWrite(M2_PWMH, (x-0.1*x));
  }
  else
  {
    analogWrite(M1_PWMH, x);
    analogWrite(M2_PWMH, x/2);
  }
}
}

void stopmotors()
{
  analogWrite(M1_PWMH, 0); //Turn the motor off by shorting it to GND
  analogWrite(M2_PWMH, 0); //Turn the motor off by shorting it to GND
  analogWrite(LM_PWM, 0); //Turn the motor off by shorting it to GND
}

/*
void doEncoder()
{
  /* If pinA and pinB are both high or both low, it is spinning
  * forward. If they're different, it's going backward.
  *
  * For more information on speeding up this process, see
  * [Reference/PortManipulation], specifically the PIND register.
  */
}

```

```

if (digitalRead(LM_EncA) == digitalRead(LM_EncB)) {
  encoderPos++;
} else {
  encoderPos--;
}

Serial.println (encoderPos, DEC);
}*/

/* Alternate function to show detail in encoder
void doEncoder_Expanded(){
  if (digitalRead(encoder0PinA) == HIGH) { // found a low-to-high on channel A
    if (digitalRead(encoder0PinB) == LOW) { // check channel B to see which way
      // encoder is turning
      encoder0Pos = encoder0Pos - 1;    // CCW
    }
    else {
      encoder0Pos = encoder0Pos + 1;    // CW
    }
  }
  else // found a high-to-low on channel A
  {
    if (digitalRead(encoder0PinB) == LOW) { // check channel B to see which way
      // encoder is turning
      encoder0Pos = encoder0Pos + 1;    // CW
    }
    else {
      encoder0Pos = encoder0Pos - 1;    // CCW
    }
  }
  }
  Serial.println (encoder0Pos, DEC); // debug - remember to comment out
  // before final program run
  // you don't want serial slowing down your program if not needed
}
*/

```

C.2 Updated Diwheel Program

```

//*****Senior Capstone Design Diwheel Program 3.0*****//
/*

```

CODE MODIFIED AFTER DIWHEEL FAILURE: PROGRAM HAS NOT BEEN TESTED

Author: Christopher Parisi
 Organization: California Baptist University College of Engineering

Baud Rate: 9600 (Arduino Standard)
 Use either HyperTerminal or Arduino Serial Monitor for Control

CAUTION! When controlling Diwheel, be sure not to change motor direction instantaneously.
 Back EMF can harm the motor drivers and permanently break them.

Discrete Transfer Function For Sensor Feedback:

If the gyro and accelerometer IMU sensor was functional, this transfer function would have been used for controlling the angle of the chassis.

The transfer function takes in a theta value and provides a distance that the linear weight needs to be moved.

This number would be used along with the linear motor encoder to control how far the weight is moved at what time.

Z represents the pitch angle given from the IMU sensor and then give a distance.

This movement will be tied to Arduino's Interrupt and will constantly be moving to compensate for various accelerations.

$$(5.519*z^2 - 10.95*z + 5.43) / (z^3 - 2.573*z^2 + 2.204*z - 0.6278) = \text{distance}$$

Controls for Diwheel

```

w: forward
s: backward
a: left
d: right
x: stop(safety stop-slows down before stopping)
v: emergency stop
g: start system
r: reset system
1: 10% Speed
2: 20% Speed
3: 30% Speed
4: 40% Speed
5: 50% Speed
6: 60% Speed
7: 70% Speed
8: 80% Speed
9: 90% Speed
0: 100% Speed
*/

```

//Arduino Pin Assignments

```

const int LM_DIR = 12; //Direction control for the linear motor
const int LM_PWM = 3; //PWM Signal for speed control of linear motor --CAN I ACTUALL USE THIS PIN?
const int LM_EncA = 2; //Linear motor encoder A

```

```

const int LM_EncB = 5;    //Linear motor encoder A
const int M1_DIR = 6;    //Direction control for Motor 1
const int M2_DIR = 7;    //Direction control for Motor 2
const int M1_RESET = 8;  //Reset signal for Motor Driver 1
const int M2_RESET = 9;  //Reset signal for Motor Driver 2
const int M1_PWMH = 10;  //PWM signal for speed control of Motor 1
const int M2_PWMH = 11;  //PWM signal for speed control of Motor 2

//Other Variable Assignments
//volatile unsigned int encoderPos = 0;
int command = 0;         //User input from Serial Terminal
int sub_command = 0;     //Command used to update information during ramping function
char dir = 'f';         //keeps track of diwheel direction
int spd = 0;            //Updates speed which is used to control diwheel speed
int current_spd = 0;     //keeps track of current speed for ramping function
int desired_spd = 0;    //keeps track of desired speed for ramping function
char lm_dir = 'f';      //keeps track of linear motor direction
char lm_pos = 'b';      //keeps track of which side the weight is located

void setup()
{
  //code section from from http://playground.arduino.cc/Main/RotaryEncoders#Example1
  pinMode(LM_EncA, INPUT);
  digitalWrite(LM_EncA, HIGH);    // turn on pullup resistor
  pinMode(LM_EncB, INPUT);
  digitalWrite(LM_EncB, HIGH);    // turn on pullup resistor
  //attachInterrupt(0, doEncoder, CHANGE); // encoder pin on interrupt 0 - pin 2

  Serial.begin(9600);    //9600 Baud Rate

  Serial.println("Starting Up...");
  pinMode(LM_DIR, OUTPUT);
  pinMode(LM_PWM, OUTPUT);
  //pinMode(LM_EncA, INPUT);
  //pinMode(LM_EncB, INPUT);
  pinMode(M1_DIR, OUTPUT);
  pinMode(M2_DIR, OUTPUT);
  pinMode(M1_RESET, OUTPUT); //default HIGH
  pinMode(M2_RESET, OUTPUT); //default HIGH
  pinMode(M1_PWMH, OUTPUT);
  pinMode(M2_PWMH, OUTPUT);

  brakeMotors();
  Serial.println("System Ready.");
}

void loop()
{
  /* Fault Flags removed due to lack of I/O Pins on Arduino
  if ((FF1Motor1 == LOW && FF2Motor1 == HIGH) || (FF1Motor2 == LOW && FF2Motor2 == HIGH))
  //Short Circuit
  {
    stopmotors();
    Serial.println("Short Circuit Detected. Stopping Diwheel");
  }
  */
}

```

```

}
else if((FF1Motor1 == HIGH && FF2Motor1 == LOW) || (FF1Motor2 == HIGH && FF2Motor2 == LOW))
//Over Temperature
{
  stopmotors();
  Serial.println("Over Temperature. Stopping Diwheel");
}
else if ((FF1Motor1 == HIGH && FF2Motor1 == HIGH) || (FF1Motor2 == HIGH && FF2Motor2 ==
HIGH)) //Under Voltage
{
  stopmotors();
  Serial.println("Under Voltage. Stopping Diwheel");
}
*/

if (Serial.available() > 0)
{
  command = Serial.read();
  switch(command) {

    case 'w': //forward
      if (dir == 'b' || dir == 's')
      {
        brakeMotors();          //Stop Motors to avoid back EMF
        delay(1000);
      }
      digitalWrite(M1_DIR, HIGH); //Current flows from OUTA(+) to OUTB(-)
      digitalWrite(M2_DIR, HIGH); //Current flows from OUTA(+) to OUTB(-)
      dir = 'f';
      speedControl(dir, spd);    //adjust direction while moving
      break;

    case 'a': //left
      dir = 'l';
      speedControl(dir, spd);    //adjust direction while moving
      break;

    case 's': //backward
      if (dir == 'f' || dir == 's')
      {
        brakeMotors();          //Stop Motors to avoid back EMF
        delay(1000);
      }
      digitalWrite(M1_DIR, LOW); //Current flows from OUTB(-) to OUTA(+)
      digitalWrite(M2_DIR, LOW); //Current flows from OUTB(-) to OUTA(+)
      dir = 'b';
      speedControl(dir, spd);    //adjust direction while moving
      break;

    case 'd': //right
      dir = 'r';
      speedControl(dir, spd);
      break;

    case 'e': //spin clockwise
      brakeMotors();

```



```

delay(1000);
digitalWrite(M1_DIR, HIGH);
digitalWrite(M2_DIR, LOW);
dir = 's';
break;
case 'q': //spin counter-clockwise
  brakeMotors();
  delay(1000);
  digitalWrite(M1_DIR, LOW);
  digitalWrite(M2_DIR, HIGH);
  dir = 's';
  break;

case 'r': //Reset Motor Driver Circuits
  digitalWrite(M1_RESET, LOW); //Resets Motor 1 (Clears Fault Flags)
  digitalWrite(M2_RESET, LOW); //Resets Motor 2 (Clears Fault Flags)
  break;
case 'g': //Releases hold from RESET signal. Must use for initial start up
  digitalWrite(M1_RESET, HIGH);
  digitalWrite(M2_RESET, HIGH);
  break;

case 'x': //stop(coast)
  slowMotors();
  break;

case 'v': //emergency stop (break)
  brakeMotors();
  break;

case 'k': //Linear motor forward (USE ONLY FOR TESTING!)
  if (lm_dir == 'b')
  {
    brakeMotors();
    delay(1000);
  }
  digitalWrite(LM_DIR, LOW);
  lm_dir = 'f';
  speedControl(dir, spd);
  break;

case 'l': //Linear motor backward (USE ONLY FOR TESTING!)
  if (lm_dir == 'f')
  {
    brakeMotors();
    delay(1000);
  }
  digitalWrite(LM_DIR, HIGH);
  lm_dir = 'b';
  speedControl(dir, spd);
  break;

case '1':
  spd = 25; //10% power
  desired_spd = 1;
  speedRamp();

```

```

    break;
case '2':
    spd = 50;      //20% power
    desired_spd = 2;
    speedRamp();
    break;
case '3':
    spd = 75;      //30% power
    desired_spd = 3;
    speedRamp();
    break;
case '4':
    spd = 100;     //40% power
    desired_spd = 4;
    speedRamp();
    break;
case '5':
    spd = 125;     //50% power
    desired_spd = 5;
    speedRamp();
    break;
case '6':
    spd = 150;     //60% power
    desired_spd = 6;
    speedRamp();
    break;
case '7':
    spd = 175;     //70% power
    desired_spd = 7;
    speedRamp();
    break;
case '8':
    spd = 200;     //80% power
    desired_spd = 8;
    speedRamp();
    break;
case '9':
    spd = 225;     //90% power
    desired_spd = 9;
    speedRamp();
    break;
case '0':
    spd = 255;     //100% power
    desired_spd = 0;
    speedRamp();
    break;
}
}
}

void speedRamp()
{
    if (desired_spd > current_spd)
    {
        while ((current_spd != desired_spd) && (Serial.read() != 'x') && (Serial.read() != 'v'))
        {

```

```

sub_command = Serial.read(); //check if a new command has been updated
if (sub_command == 'l')
{
  dir = 'l';
}
else if (sub_command == 'r')
{
  dir = 'r';
}
current_spd++; //ramp of speed 10% at a time
spd = current_spd*25; //Pass PWM number corresponding to speed
speedControl(dir, spd); //Update speed and direction
delay(500); //Wait 1/2 a second until ramping up again
}
if (Serial.read() == 'v')
{
  brakeMotors();
}
else if (Serial.read() == 'x')
{
  slowMotors();
}
}
else if (desired_spd < current_spd)
{
  while ((current_spd != desired_spd) && (Serial.read() != 'x') && (Serial.read() != 'v'))
  {
    sub_command = Serial.read();
    if (sub_command == 'l')
    {
      dir = 'l';
    }
    else if (sub_command == 'r')
    {
      dir = 'r';
    }
    current_spd--;
    spd = current_spd*25;
    speedControl(dir, spd);
    delay(500);
  }
  if (Serial.read() == 'v')
  {
    brakeMotors();
  }
  else if (Serial.read() == 'x')
  {
    slowMotors();
  }
}
}
else
  speedControl(dir, spd);
}

void speedControl(char dir, int x)
{

```

```

if (dir == 'f' || dir == 'b' || dir == 's')
{
    analogWrite(M1_PWMH, (x-0.1*x));           //10% reduced to counter drift
    analogWrite(M2_PWMH, x);
    if (dir == 'f' && lm_pos != 'b')
    {
        digitalWrite(LM_DIR, HIGH);
        analogWrite(LM_PWM, 255);
        delay(1000);
        analogWrite(LM_PWM, 0);
        lm_pos = 'b';
    }
    else if (dir == 'b' && lm_pos != 'f')
    {
        digitalWrite(LM_DIR, LOW);
        analogWrite(LM_PWM, 255);
        delay(1600);
        analogWrite(LM_PWM, 0);
        lm_pos = 'f';
    }
}
else if (dir == 'l')
{
    if (spd == 50 || spd == 75 || spd == 100) //30% drop in speed in left wheel
    {
        analogWrite(M1_PWMH, (x-0.3*x));
        analogWrite(M2_PWMH, x);
    }
    else if (spd == 125 || spd == 150 || spd == 175) //20% drop in speed in left wheel
    {
        analogWrite(M1_PWMH, (x-0.2*x));
        analogWrite(M2_PWMH, x);
    }
    else if (spd == 200 || spd == 225 || spd == 255) //10% drop in speed in left wheel
    {
        analogWrite(M1_PWMH, (x-0.1*x));
        analogWrite(M2_PWMH, x);
    }
    else
    {
        analogWrite(M1_PWMH, x/2);
        analogWrite(M2_PWMH, x);
    }
}
else if (dir == 'r')
{
    if (spd == 50 || spd == 75 || spd == 100) //30% drop in speed in right wheel
    {
        analogWrite(M1_PWMH, x);
        analogWrite(M2_PWMH, (x-0.3*x));
    }
    else if (spd == 125 || spd == 150 || spd == 175) //20% drop in speed in right wheel
    {
        analogWrite(M1_PWMH, x);
        analogWrite(M2_PWMH, (x-0.2*x));
    }
}

```

```

    else if (spd == 200 || spd == 225 || spd == 255) //10% drop in speed in right wheel
    {
        analogWrite(M1_PWMH, x);
        analogWrite(M2_PWMH, (x-0.1*x));
    }
    else
    {
        analogWrite(M1_PWMH, x);
        analogWrite(M2_PWMH, x/2);
    }
}

}

void brakeMotors()
{
    analogWrite(M1_PWMH, 0); //Turn the motor off by shorting it to GND
    analogWrite(M2_PWMH, 0); //Turn the motor off by shorting it to GND
    analogWrite(LM_PWM, 0); //Turn the motor off by shorting it to GND
    desired_spd = 0;
    current_spd = 0;
    spd = 0;
}

void slowMotors()
{
    desired_spd = 0;
    while (current_spd != desired_spd)
    {
        current_spd--;
        spd = current_spd*25;
        speedControl(dir, spd);
        delay(300);
    }
    analogWrite(M1_PWMH, 0); //Turn the motor off by shorting it to GND
    analogWrite(M2_PWMH, 0); //Turn the motor off by shorting it to GND
    analogWrite(LM_PWM, 0); //Turn the motor off by shorting it to GND
    spd = 0;
    current_spd = 0;
}

/*
void doEncoder()
{
    /* If pinA and pinB are both high or both low, it is spinning
    * forward. If they're different, it's going backward.
    *
    * For more information on speeding up this process, see
    * [Reference/PortManipulation], specifically the PIND register.
    */
    if (digitalRead(LM_EncA) == digitalRead(LM_EncB)) {
        encoderPos++;
    } else {
        encoderPos--;
    }
}

```

```
Serial.println (encoderPos, DEC);
}*/

/* Alternate function to show detail in encoder
void doEncoder_Expanded(){
  if (digitalRead(encoder0PinA) == HIGH) { // found a low-to-high on channel A
    if (digitalRead(encoder0PinB) == LOW) { // check channel B to see which way
      // encoder is turning
      encoder0Pos = encoder0Pos - 1;    // CCW
    }
    else {
      encoder0Pos = encoder0Pos + 1;    // CW
    }
  }
  else // found a high-to-low on channel A
  {
    if (digitalRead(encoder0PinB) == LOW) { // check channel B to see which way
      // encoder is turning
      encoder0Pos = encoder0Pos + 1;    // CW
    }
    else {
      encoder0Pos = encoder0Pos - 1;    // CCW
    }
  }
  Serial.println (encoder0Pos, DEC); // debug - remember to comment out
  // before final program run
  // you don't want serial slowing down your program if not needed
}
*/
```

D. Appendix D: MatLab Controls Derivations

D.1 Dynamics Derivations

The following MatLab code was used to make the transfer functions used in creating the controller from the dynamic equations.

```
%Diwheel derivation 1
syms theta phi l T F s Mb Mw m e Jb R r Jw g
Mb=30; %Mass of body
Mw=2; %Mass of Wheels
m=1; %Mass of sliding weight
e=.136; %Center of mass pendulum length
Jb=7.5; %moment of inertia of body
R=.310; %Wheel Radius
r=0.122; %sliding weight pendulum length
Jw=2.5; %moment of inertia of wheels
g=9.81; %gravity

E1=(Jb+Mb*e^2+m*(r^2))*theta*s^2-Mb*g*e*theta+m*g*(1-r*theta)-
(Mb*R*e+m*R*r)*phi*s^2+T+m*r*1*s^2;
E2=(Mw*R^2+Jw-Mb*(R-e)*R-m*(R-r)*R)*phi*s^2+(Jb+Mb*(R-e)*R+m*(R-
r)*R)*theta*s^2-Mb*g*e*theta+m*g*(1-r*theta)-m*(R-r)*1*s^2;
E3=-F+m*g*theta+m*(-r*theta*s^2+R*phi*s^2+1*s^2);
phi_sol=solve(E1,phi);
E2s=subs(E2,phi,phi_sol);
E2s=simplify(E2s);
E2s=collect(E2s,theta);

%%%%%%%%%%%%%%
E2s_theta=subs(E2s,T,0)
E2s_theta=subs(E2s_theta,1,0)
E2s_theta=collect(E2s_theta,s)
E2s_theta=collect(E2s_theta,theta)
%%%%%%%%%%%%%%
E2s_l=subs(E2s,T,0)
E2s_l=subs(E2s_l,theta,0)
E2s_l=collect(E2s_l,s)
E2s_l=collect(E2s_l,1)
%%%%%%%%%%%%%%
E2s_T=subs(E2s,1,0)
E2s_T=subs(E2s_T,theta,0)
E2s_T=collect(E2s_T,s)
E2s_T=collect(E2s_T,T)

E2s_theta
pretty(E2s_theta)
E2s_l
pretty(E2s_l)
E2s_T
pretty(E2s_T)
```

```

%Diwheel Derivation 2
syms theta phi l T F s Mb Mw m e Jb R r Jw g
Mb=30; %Mass of body
Mw=2; %Mass of Wheels
m=1; %Mass of sliding weight
e=.136; %Center of mass pendulum length
Jb=7.5; %moment of inertia of body
R=.310; %Wheel Radius
r=0.122; %sliding weight pendulum length
Jw=2.5; %moment of inertia of wheels
g=9.81; %gravity

E1=(Jb+Mb*e^2+m*(r^2))*theta*s^2-Mb*g*e*theta+m*g*(l-r*theta)-
(Mb*R*e+m*R*r)*phi*s^2+T+m*r*l*s^2;
E2=(Mw*R^2+Jw-Mb*(R-e)*R-m*(R-r)*R)*phi*s^2+(Jb+Mb*(R-e)*R+m*(R-
r)*R)*theta*s^2-Mb*g*e*theta+m*g*(l-r*theta)-m*(R-r)*l*s^2;
E3=-F+m*g*theta+m*(-r*theta*s^2+R*phi*s^2+l*s^2);

l_sol=solve(E3,l)
E1s=subs(E1,l,l_sol)
E2s=subs(E2,l,l_sol)

phi_sol=solve(E1s,phi);

E2s=subs(E2s,phi,phi_sol);
E2s=simplify(E2s);
E2s=collect(E2s,phi)

E2s_theta=subs(E2s,T,0);
E2s_theta=subs(E2s_theta,F,0);
E2s_theta=collect(E2s_theta,s);
E2s_theta=collect(E2s_theta,theta);

E2s_T=subs(E2s,theta,0);
E2s_T=subs(E2s_T,F,0);
E2s_T=collect(E2s_T,s);
E2s_T=collect(E2s_T,T);

E2s_F=subs(E2s,T,0);
E2s_F=subs(E2s_F,theta,0);
E2s_F=collect(E2s_F,s);
E2s_F=collect(E2s_F,F);

E2s_theta
pretty(E2s_theta)
E2s_T
pretty(E2s_T)
E2s_F
pretty(E2s_F)

```


D.2 Sliding Weight Position Controller

Theta Controller =

$$0.2304 s^2 + 0.38 s + 1$$

$$0.000333 s^3 + 0.0155 s^2 + 0.223 s + 1$$

DiWheel Model =

$$-0.006003 s^2 + 1.129$$

$$s^2 + 1.764 s + 4.742$$

Controlled Diwheel Model =

$$-0.001383 s^4 - 0.002281 s^3 + 0.254 s^2 + 0.4289 s + 1.129$$

$$0.000333 s^5 + 0.0147 s^4 + 0.2496 s^3 + 1.721 s^2 + 3.25 s + 5.871$$

