ROBOTIC STEERING

Gaetan Foisy **Andrew Miller** **Hanna Saba** **Tianyu Zhou**

ABSTRACT

This team was tasked with designing and building a small robot that can be programed by computer science students at the University of Rochester in order to learn more about programming. To meet this challenge, the team designed a four-wheeled robot powered by two DC motors which, in turn, are controlled by a PIC32 microcontroller and a raspberry pi, which are accessible on the top of the design. The robot is steered using "skid-steer" which is where the wheels on one side of the robot are synchronized so that the left and right side move independently. This allows the robot to have a zero-degree turning radius. Most of the manufactured parts were 3D printed using ABS for ease of replication. The robot is housed in an 11" x 11" x 7 ¾" box of 8020 aluminum framing. This allows for all the moving parts, except for the wheels, to be completely enclosed for students' safety. By the end of the semester the team had designed and built a robot that was capable of moving with direct power to the motors but was not able to create a program that would run the robot independently. All other requirements and specifications were met, and the sponsor expressed his satisfaction with this team's work. Still, more testing and programming should be carried on in the future to fully develop the robot if possible.

PROBLEM DEFINITION

Robotics courses at the University of Rochester need inexpensive, reliable, and adaptable robots which can be used for hands-on learning by computer science students. These robots will be used to teach the fundamentals of robotic control, a field that is becoming increasingly integral across many disciplines and industries. This team's contribution to the problem will be to create an inexpensive and easy to replicate solution that fits the needs of the robotics classes being taught at the University of Rochester.

REQUIREMENTS, SPECIFICATIONS, DELIVERABLES

Deliverables:

- A functional prototype.
- A complete set of documentation for assembling and servicing the robots.
- A comprehensive bill of materials.
- CAD models with both NX and STEP files for replication and repairs.
- A wiring diagram.
- A video demonstration of an operational prototype.

Requirements:

- The final product must have the ability to make controlled movements in SE(2).
- The suspension system on the final product must keep all wheels on the ground while in operation.
- The PIC32 microcontroller and raspberry pi must be able to be reprogrammed without disassembly of the robot.
- An emergency stop, which will secure all power, must be provided on the top of the final design and remain accessible during operation.
- There must be at least two DC motors powering at least two wheels.
- The final product must not have any accessible points where a person's finger could become entangled and injured.
- The final product must incorporate a mobile power supply in the design for untethered operation.
- The final design must contain no sharp or otherwise dangerous edges.

Specifications:

- The DC motors must experience less that 1lbf of off-axis loading at any time.
  - Tested by static analysis.
- Each unit must cost less than $500 in raw material and purchased components.
  - Measured by cost analysis.

Copyright © 20xx by ASME

- The final product must be able to achieve within 5% of the target top speed of 0.5 m/s on a flat tile floor.
  - Tested with a stopwatch and tape measure.
- The final product must not topple when tilted to an angle of 20 degrees from horizontal.
  - Tested with CAE an analysis and physical tilt test.
- The battery in the final product must last for at least 15 minutes while operating at full speed and driving in a 15ft diameter circle.
  - Tested using initial power consumption calculations in addition to performing an endurance test.

CONCEPTS

The first concept featured a differential rocker suspension system driven by skid-steering. The rocker suspension is created using PVC pipes. The main axle is suspended under the top of the chassis and utilizes gears to create the two differential elements. The four wheels are belt driven and each side of the steering system is driven by a single motor. PVC pipes are used for ease of manufacturing. The pipes snap together, and the belts can run through them to the drive the wheels. The electronics and battery are placed under the axle on a 3D printed platform resting on the bottom of the chassis. This allows for easy access to most of the electrical parts, except for the motors which are on the PVC pipes.

The second concept utilized a skid-steer design with wheels connected to an independent suspension system, which is called pushrod suspension. The suspension system consists of trailing arms and a spring attached between the joints in the middle. This way if one side of the robot is going over rough terrain, the wheels on the other side will remain in contact with the ground.

The third concept had a simple belt driven design with direct drive from the motor. Spring-loaded suspension would be used to maintain all wheels on the ground over rough terrain. This would necessitate the need for a spring-loaded belt tensioner which would increase the off-axis loading on the motor shaft, especially over rough terrain. This design would utilize skid steering.

The final concept had a rocker system with two differential elements placed on the front and rear of the frame. Having these differential elements out of line with the main axis of rotation for the robots has two main advantages. The first, is the placement of the differential elements close to the wheels. The mechanical advantage gained from this makes the loading on the differential elements much lower than in a conventional set-up. The second main advantage of the off-axis differentials is that the main body of the robot isn't freely suspended. In traditional rocker machines, it is common for the main body to be hanging off the center axle. This allows the body to remain horizontal while traveling up steep inclines. However, this is less stable and robust, especially in a small formfactor. Fortunately, this problem is negated by putting the differential off-axis. It securely constrains the main body of the robot while maintaining the advantages of a rocker system.

The concepts were ranked based on a predefined set of criteria in a Pugh matrix (table 1). The safety criteria are based on the perceived ease of inclosing moving parts and other hazards. It may be subject to change as these concepts are fleshed out and new hazards are identified. All the models tied in the safety category because the moving parts could easily be enclosed in a clear acrylic box, which is a huge improvement in safety over the previous year's model in addition to being visually interesting. The "differential rocker" scored higher in the adaptability category than the "PVC rocker". While both enjoy the benefits of a rocker bogie system, the "differential rocker" is a more robust design.

The results of this preliminary mechanical analysis were also factored into the pugh decision matrix under the term "stability". For this analysis, mass data from the CAD models were used to estimate the center of gravity for all the models. Then, the center of gravity was used to calculate the maximum angle the robots could be tilted before toppling. The relative stability of these concepts is important as their functionality is tied to their ability to keep all wheels on the ground when maneuvering over rough terrain. A robot that topples over does not have all its wheels on the ground. From the analysis it is shown that the "skid-steering" design is the most stable, followed by the "differential rocker" and "belt drive".

In the end, the "differential rocker" was the design the team decided would best meet the sponsor's needs. It is a simple and robust design that has the capability to be efficiently reproduced and repaired as necessary. The team believes that the final product will help teach the fundamentals of robotic control for many years to come.

MECHANICAL ANALYSIS

The team performed a series of mechanical analyses in order to decide upon the best design features. These included FEA models and basic tolerance analysis. One tolerance issue that the team came across concerned the gears in the drivetrain. To get the contact point between the gears on the wheel and the gears on the drivetrain frame to be as close to their respective pitch angles as possible, the tolerances of the axel mounts, suspension paddle, and shoulder screws that connect the gear to the drivetrain frame must be very small. This then would have to contend with any bending experienced in the wheel axel. An RSS analysis (Equation 1) shows that the 3σ value is .074mm. The data for the analysis can be seen in table 2.

Copyright © 20xx by ASME

The rigidity of this design depends mostly on the suspension paddles. The suspension paddles house the drive train, wheels, differential suspension system, and motors. During the initial assembly it was clear that the paddles were under heavy load and needed to be redesigned. To ensure that the redesigned paddles are stiffer and won't fail, a fine element analysis was run. Nastran was used through NX CAD to calculate the stresses in the paddles while under load. FEA was run on both the initial and redesigned paddles.

The initial parameters for both the tests were the same (Figures 2 & 6). The paddles were 3D hybrid meshed with element CTETRA(10) and an element size of 1mm. The material properties were set to ABS for both paddles. A load of 250N was applied to each paddle (125N/paddle & 62.5N/wheel). This value was determined from the weight of the robot converted to newtons (54.2 N) and a factor of safety of 4. A fixed constraint was added to the center axle and the bevel gear connections. The axle constrains the parts in all directions but allows for a little bit of rotation in the Y and Z directions. This is due to wobble that can occur between the Oilite bearings and axle. The bevel gears only restrict movement in the Z direction. The forces and constraints were applied properly to their respective areas of contact, as shown by the load forces applied in figures 3 & 7.

Surprisingly, the maximum stress in the redesigned paddle was slightly larger than the initial design. The initial paddle's calculated maximum stress was 45.37 MPa (Figure 4) while the redesigned paddle had a maximum stress of 46.1 MPa (Figure 8). Despite the increase in stress, the maximum displacement in the redesigned paddle was 1.158 mm (Figure 5) which was smaller than the 1.214mm (Figure 9) displacement of the initial design. The differences between the paddles aren't very significant however, the redesigned paddle should have noticeably decreased the maximum stress and increased the stiffness. The similarities between the paddles were due to the redesign occurring before any FEA had been completed. It was clear that the initial design needed to be stronger and stiffer, but without knowing the location of maximum stress, the redesign did little to improve the paddle. Despite the little improvement, both paddles function properly and can barely withstand the 250N applied to the robot. The maximum yield strength of ABS is 48.26 MPa (7,000 psi) and the robot only needs to withstand its own weight (50N) to function. Now with the FEA being completed, the paddle can be properly redesigned.

The stopblocks for the suspension arms are an integral part of the system. If one of them breaks, the differential bevel gears can over-rotate and de-mesh. Fortunately, there are a few simple beam calculations that can provide a reasonable estimate for the internal stresses. A load of 250N is applied at the limiting screw, to simulate a suspension linkage pushing against it in a worse-case scenario. The whole system can be approximated by a cantilever beam to approximate the reaction forces on the mounting bolt (figure 10). These forces are then used in another hand calculation which takes the bolt's location and the geometry of the stopblock to find the stress in the plane where the bolt head sits. These calculations can be seen in figure 10 and show that the maximum stress in the part is 4.28MPa. This is far below the 40-60MPa yield strength of ABS plastic, so the stopblocks can reasonably be assumed safe, and could even be a source for future optimization.

MANUFACTURING

For the suspension subsystem, the main method of manufacturing was 3D printing. This is due to 3D printing being well suited for short production runs, and the client has access to inexpensive 3D printing services. Additionally, it allows the client to change and modify the designs with ease. The remainder of the parts in the suspension were purchased and then finished with simple tools like a hacksaw or wire cutters, depending on the component.

The drivetrain components were primarily manufactured using 3D printing. As stated above, this would allow for the customer to reprint replacement parts and make modifications as they saw fit. It would also allow for massive production of robots for students to use. The gears for the drivetrain were modified from a design of an existing plastic gear, instead of being purchased due to the ease of adding necessary mounting holes and shaft thicknesses. The 3D printed parts were assembled using basic hardware purchased through McMaster-Carr.

The gear rack and pinion for the differential suspension system were manufactured using 3D printing. Due to the complex design of the gear teeth, they would be difficult to machine. The parts can not be purchased online but can be made with purchased parts from McMaster. However, that would require machining and welding which would be difficult to do accurately for the four gears racks required for every robot. 3D printing allows for the parts to be manufactured in the easiest way possible. The suspension system is also constantly under load and if the parts were to break, the client can quickly print new ones. To 3D print the parts, the gear and pinion were designed in ways to reduce overhang. One side of the gear rack was made flat to print laying on its side. The holes through the gear would usually be difficult to print but are no issue with proper supports.

For electronics subsystem, the components are divided into three different manufacturing processes: purchased, laser cut, and 3D printed. Purchased parts include all the electronics devices, the fasteners that are needed for mounting, 4 acrylic sheets, and 1 high strength perforated plastic sheet. Two of the acrylic sheets are laser cut to mount the switch, E-stop, and screen with the PIC32. The other two acrylic sheets are laser cut into side panels which enclose the suspension system to prevent accessibility into the gears and other moving parts during operation. It is preferable for people to see the mechanical systems on inside the robot while it is functioning. This is the reason why clear acrylic sheets are used for the side panels instead of a less expensive opaque option. These acrylic sheets are also strong enough to hold electronic devices such as the PIC 32 and Raspberry PI screen. In order to put holes in these acrylic sheets

Copyright © 20xx by ASME

to mount the electronic devices, a laser cutting process is recommended as it is efficient and accurate. A perforated plastic sheet is used for mounting the batteries and motor driver. Batteries are attached to the sheet with zip ties while the motor drivers are attached using aluminum standoffs. One of the advantages of a perforated sheet here is that most electronic devices have a mounting diameter of 3mm which is a similar size as the holes on the perforated sheet. If, in the future, more devices are required in the system, they can easily be mounted on the perforated sheet. The sheet is needed to be cut into size using a router. In this subsystem the mounting brackets that hold the perforated sheet to the frame are 3D printed. Since the team could not find any brackets online with a suitable size, designing and 3D printing a mounting bracket saved time and money. Since the side panels also function as a support for the perforated sheet, the 3D printed bracket only needed to be strong enough to hold the perforated sheet with the weight of the batteries. Finally, a wiring diagram is included in the reference manual provided by the sponsor. The reference manual will be in the same folder that contains this final report. A general circuit diagram is appended as figure 11 in appendix. As shown on the diagram, pins on one side of the motor driver used in this project have a high input and output voltage range while the pins on the other side have a much lower voltage range. Pins with the high voltage connects the e-stop, the switch, the 12 volts battery and the two motors in series. Two motors are in parallel. The loop with the high voltage is formed with blue lines. The loop with low voltage is presented by green lines. The low-voltage end of the H-bridge connects to PIC32 and Raspberry Pi. The PIC32 and the Raspberry Pi is in parallel. They are each powered by a 5-volt portable charger (also known as a power bank). In this case, if the e-stop is being used, the low-voltage loop would still be functioning.

With the manufacturing process for all the components of this project being addressed above, a bill of materials is generated and listed as table 2 in appendix. The table is divided into sections based on different subsystems. Along with the purchased components, the required number, unit price, and the purchasing links are listed in the table. The price for 3D printing is also listed in the last row for each section. The printing price is calculated using the mass of the printing parts multiplied by the typical filament cost (reference [4]). Considering that parts should be printed with infill, the mass is multiplied by a factor of 0.8. As shown in the bill of materials, the estimated cost for purchasing is about $390.40. The table does not include the price for most of the electronics parts, such as motor driver, PIC32, Raspberry Pi screen, etc., since they are all provided by the sponsor. In order to assemble all the components together, it requires approximately 4 hours to build the robot. Combing the cost for building, which is charged at $100/hr, the final cost for this project is $790.40. Table 3 in the appendix represents the development time each team member spent on this project. The numbers come from the daily SCRUM each member filled out every day since the beginning of the project.

If the scope of this project was scaled to 1000 robots, several changes can be made to improve the cost and build time. For example, the 12 volts battery powering the motors is currently 2000mAh. This is sufficient to meet the specification of lasting 15 minutes while the robot is under operation. The original design was meant to supply the power for the motors, the Raspberry Pi screen, as well as the PIC32 microcontroller. However, since the screen and microcontroller operate under 5 volts, either a regulator or a separate 5-volt power supply is required. In order to incorporate a regulator, the wiring diagram would take time to be refined. The solution that was implemented was to add two 5-volt portable power-bank chargers directly connected to the screen and the microcontroller. With the extra battery power added on, the 2000mAh battery is overpowered and can be replaced with a lower capacity power supply. Usually, low-capacity power supplies cost less, which would tremendously reduce cost if the scale of this project was increased. Another modification that can be considered in the future is the manufacturing process of the perforated sheet. Even though the sheet is easy to cut, it is not convenient to drill holes on. To be held by the mounting bracket mentioned before, four holes with size of 5-millimeter diameter need to be drilled on. However, since the distance between the holes on the perforated board is less than 5 millimeters, mounting holes need to be drilled in between. The current solution is to widen one of the holes of the perf board until the mounting screw can fit. The problem with this method is that it is imprecise, and the mounting locations will likely be off. This can be fixed, however, it takes time. For a large-scale production, this manufacturing process is entirely too inefficient. Changing the material for the mounting board or changing the way of drilling would be a dependable alternative.

TEST PLAN AND RESULTS

- The DC motor must experience less than1 lbf of off-axis loading under any circumstances.

  Testing for the maximum amount of off-axis loading was computed using the situation of the robot going up a 20-degree incline. The final maximum off-axis loading was 0.7 lbf, below the requirement of 1lbf max. The data for this analysis can be found in table 5.

- Each unit must cost less than $500.

  Test for determining the cost of each unit was constructing a bill of material table. Each team member put all the parts that were needed to be purchased for their responded subsystem into one excel sheet. The excel sheet contained information of all

Copyright © 20xx by ASME

the components with their corresponding amounts, price per unit, as well as the accessible links. In the last row of the excel sheet, the total cost for one unit was calculated.

From table 3 attached in the appendix, the final cost for one unit is $390.40, which is less than $500. Thus, the project meets the cost specification.

- The final robot must be able to achieve within 5% of 0.5 m/s speed on flat tile floor.

  To test the speed of the robot, the robot was planned to be placed on a 20-meter-long flat tile track which would be measured ahead. The robot would run through the track with one of the team members standing at the end of the track and pressing the stopwatch as the robot passes through. The test would be repeated for at least 3 times in order to get a relatively accurate velocity.

  However, due to time limitation and coding difficulties, this velocity test is not able to be performed at current stage, which means the project has not meet this specification yet. Future testing on velocity should be conducted once the wiring and coding for the robot are completed.

- The final robot must not topple when tilted to an angle of 20 degrees.

  For this specification, the testing plan contains a CAE analysis using NX as well as a physical test. For the CAE analysis, the location of center of mass was found after assigning all the materials and for the final CAD model. Using equation 2, a simulated tippling angle can be obtained. The physical test plan was to set the final robot onto a piece of plank. One end of the plank would be pushed against the wall and the other end would be held and gradually pushed up until one of the wheels of the robot lose contact with the plank. Then the distance between the tilted end of the plank and the ground would be measured with a flexible ruler. The length of the plank would also be measured. By using equation 3, the actual tilted angle can be calculated. The orientation of the robot on the plank should be facing the same direction as the CAE analysis performed in NX so that the results are comparable. Figure 12 and 13 in appendix represents the setups for the CAE analysis and physical tilting test described above. The progress of the physical test will be repeated for three times. In this way, the error generated during testing would be reduced.

  With the CAE analysis and physical test being conducted, it is found that the simulated tilted angle is 40.9 while the physical tilted angle being 33.43 degrees based on three testing trails (see table 4 in appendix). Both results are above 20 degrees which meets the specification on toppling here. Still, certain amount of error might be generated during testing due to the setups and hand-on measurements. If future test would focus more on the accuracy, attaching a tilting sensor would be a better approach to measure the toppling angle.

- The battery for the final robot must be able to last for at least 15 minutes while operating at full speed and driving in a 15ft diameter circle.

  Steps for this test are simply charging the 12 volts battery to maximum state first and then run the robot on an approximately 15-foot diameter circular track. This circular track should be measured and marked ahead of the testing. The robot would be controlled by a wireless gamepad and at full speed until the 12 volts battery runs out. One of the team members would start the timer as the test begins and stop the timer after the time hits 15 minutes or longer.

  However, due to time limitation of the semester and coding difficulties, the team has not established a connection between the controller and the microchip installed on the robot. Thus, the project has not yet passed this specification. Future testing on endurance should be accomplished as soon as the wiring and coding for the robot are completed.

INTELLECTUAL PROPERTY

The idea of using differential gears at the ends of the suspension arms is not present in any patents online, suggesting that it is a patentable idea. There are many ways to incorporate suspension in a robot platform, and there are a few notable examples that are similar to the one that was used in this project. The first [1] is from a Chinese group working with omni-wheel robots. Their solution was to use independent suspension arms with springs. However, without any mechanical or electrical systems linking the wheels together, there is no guarantee that all four wheels are firmly planted on the ground. [2] Is a patent for another omnidirectional robot with a suspension system similar to the design presented in this project. It uses a rocker bar to constrain the motion of the suspension arms, which is the exact same function as the bevel gear method presented in this project. A rocker bar is better suited for traditional manufacturing as it is made up of simple shapes and linkages, while the bevel gear design is ideal for additive manufacturing. It is interesting to note that the designers for these projects have patents in a wide range of robotics applications, and don't seem to specialize in suspension design, rather the patented designs are a byproduct of other projects. In the end, using a bevel gear located at the ends of the suspension linkages to exactly constrain the suspension system of a robotic platform is a novel approach and could be considered for a patent.

Copyright © 20xx by ASME

SOCIETAL AND ENVIRONMENTAL IMPLICATIONS

The robots that are designed in this project will help hundreds of computer science students to learn more about programming and robotics. This field has seen tremendous growth over the past decade and continues to solve many problems that are found in the world today. Robots can fight fires, disarm bombs, provide transportation, and deliver goods. By training the engineers that will program these robots, the world can gain more access to them and lower the overall cost.

The cost of one of these robots in maintained under $500, allowing access to less privileged classrooms. That being said, the cost may be prohibitive for some of the more impoverished areas of the world where access to a computer can be extremely limited, even without the ability to purchase our robot.

These robots are designed with the environment in mind. The frame is made of highly recyclable aluminum and all the 3D printed parts are also recyclable. There are minimal electronic components that would contribute to e-waste and other hardware is made of steel. One component that does pose a potential issue is the lithium-ion battery which, if not recycled properly, could be harmful to the environment.

RECOMMENDATIONS FOR FUTURE WORK

Given more time to work with the design, there are a few areas that would be of interest. Reducing part count and standardizing the fasteners would be a worthy goal. Additionally, more testing would be quite useful. This project would benefit hugely from a few months of endurance testing and iterating the designs. After these iterations, the models could be recreated from the ground up. In this version, the assemblies and subcomponents were created before the model was finished. This means that the assembly tree was based on educated guesses as to how the project would progress. With the project finished, the file structure could be recreated with more robust structure and assembly constraints, which would result in a more cohesive model. Finally, with extra time, the drawing package could be expanded into a more comprehensive manual.

FORMULAS

$$RSS = \sigma_{\Sigma} = \sqrt{{\sigma_1}^2 + {\sigma_2}^2 + {\sigma_3}^2 + \cdots} \quad (1)$$

$$Tipping\ Angle = 90° - tan^{-1}\left(\frac{C.O.M.(z)}{\frac{wheel\ base}{2} - C.O.M.(x)}\right) \quad (2)$$

$$Tipping\ Angle\ from\ Physical\ Testing = asin\left(\frac{height}{length}\right) \quad (3)$$

TABLES

TABLE 1
Design Concept Pugh Matrix

Criteria	Current Design (2020)	Belt Drive	PVC Rocker	Skid Steering	Differential Rocker Bogie
Simplicity	0	3	2	1	3
Turning radius	0	0	0	0	0
Safety	0	5	5	5	5
Cost	0	2	2	2	3
Adaptability (ability to conform to rough terrain)	0	3	3	3	4
Stability	0	2	1	4	1
Total	0	15	13	15	16

Pugh matrix of design concepts

Copyright © 20xx by ASME

TABLE 2
Tolerance Analysis Data

Component	Tolerance (mm)
Bearing	0.0075
Bearing Housing	0.01
Bearing Base Plate	0.005
Suspension Paddle	0.01
Drivetrain Frame	0.005

This table shows the ± tolerances all components in line with drivetrain gears

TABLE 3
Bill of Materials

Subsystem	Component	Amount	units	Price per unit	Info and links	Total for each component
Drivetrain	m5x0.8mm, 65mm bolt	8	count	$0.31	https://www.mcmaster.com/91290A270/	$2.48
	4-40, 5/8" Bolt	8	count	$0.10	https://www.mcmaster.com/91251A112/	$0.80
	m4x0.7mm, 35mm Bolt	20	count	$0.78	https://www.mcmaster.com/91290A173/	$15.60
	m4x0.7mm, class 8 Nut	20	count	$0.01	https://www.mcmaster.com/90592A090/	$0.20
	Alloy Steel Shoulder Screws	4	count	$1.45	https://www.mcmaster.com/92981A100/	$5.80
	m5 nut	4	count	$0.02	https://www.mcmaster.com/90592A095/	$0.08
	3D printing	0.425726	kilogram	$100.00	https://www.fusion3design.com/how-much-does-3d-printing-filament-cost/#:~:text=PLA%20%26%20ABS%20(Generic%20Formulations),per%20kilogram%20from%20quality%20suppliers.	$42.57
Drivetrain/axle retainer	m3 nut	12	count	$0.01	https://www.mcmaster.com/90592A085	$0.12
	m3x22	4	count	$0.17	https://www.mcmaster.com/91290A124	$0.68
	m3x14	12	count	$0.12	https://www.mcmaster.com/91290A119	$1.44
	Nail 97801A509	4	count	$0.03	https://www.mcmaster.com/97801A509	$0.12
	12.8mm roller bearing	8	count	$1.06	https://www.amazon.com/uxcell-R188ZZ-Groove-Bearings-Shielded/dp/B082PQ9FSY/ref=sr_1_1?dchild=1&keywords=uxcell+R188ZZ&qid=1619734265&sr=8-1	$8.48

Copyright © 20xx by ASME

	3D printing		kilogram	$100.00	https://www.fusion3design.com/how-much-does-3d-printing-filament-cost/#:~:text=PLA%20%26%20ABS%20(Generic%20Formulations),per%20kilogram%20from%20quality%20suppliers.	
Frame	M5x8 rounded head screw	30	count	$0.13	https://www.mcmaster.com/94500A230	$3.90
	Open builds corner cube	8	count	$3.48	https://openbuildspartstore.com/cube-corner-connector/	$27.84
	3/8'' steel rod	1	feet	$1.93	https://www.mcmaster.com/8920K135/	$1.93
	Aluminum Extrusion HFS5-2020-156-TPW	4	count	$6.78	https://us.misumi-ec.com/vona2/detail/110302683830/?ProductCode=HFS5-2020-156	$27.12
	Aluminum Extrusion HFS5-2020-239-TPW	5	count	$6.78	https://us.misumi-ec.com/vona2/detail/110302683830/?ProductCode=HFS5-2020-239-TPW	$33.90
	acrylic sheets (3/16'')	2	sheet	$12.98	https://www.amazon.com/Source-Premium-Acrylic-PlexiGlass-S1-12x12-187/dp/B006WPXZZO/ref=sr_1_4?crid=29FTDBJLSR8TS&dchild=1&keywords=clear+acrylic+sheet+3%2F16&qid=1618785052&sprefix=clear+acrylic+sheet+3%2F%2Caps%2C256&sr=8-4	$25.96
	m5x20 SHCS	2	count	$0.14	https://www.mcmaster.com/91290A242	$0.28
	M5 nut	2	count	$0.02	https://www.mcmaster.com/90592A095/	$0.04
	M5_T_nuts	2	count	$0.37	https://www.mcmaster.com/5537T651/	$0.74
Suspension	3/8'' steel rod	2	feet	$1.93	https://www.mcmaster.com/8920K135/	$3.86
	6338K414 oil bearing	4	count	$0.88	https://www.mcmaster.com/6338K414/	$3.52
	M5x8 rounded head screw	4	count	$0.13	https://www.mcmaster.com/94500A230	$0.52
	M5_T_nuts	4	count	$0.37	https://www.mcmaster.com/5537T651/	$1.48
	3D printing	0.184543	kilogram	$100.00	https://www.fusion3design.com/how-much-does-3d-printing-filament-cost/#:~:text=PLA%20%26%20ABS%2	$18.45

Copyright © 20xx by ASME

					0(Generic%20Formulations),per%20kilogram%20from%20quality%20suppliers.	
Differential	6338K414 oil bearing	4	count	$0.88	https://www.mcmaster.com/6338K414/	$3.52
	Shaft Clamp	2	count	$2.40	https://www.mcmaster.com/6435K13/	$4.80
	m3X22 screw	8	count	$0.17	https://www.mcmaster.com/91290A124/	$1.36
	m3 nut	8	count	$0.01	https://www.mcmaster.com/90592A085	$0.08
	3D printing	0.050001	kilogram	$100.00	https://www.fusion3design.com/how-much-does-3d-printing-filament-cost/#:~:text=PLA%20%26%20ABS%20(Generic%20Formulations),per%20kilogram%20from%20quality%20suppliers.	$5.00
Electronics	E-stop	1	count	$14.88	https://www.amazon.com/mxuteuk-Stainless-Emergency-Connection-MXU-DT-CT/dp/B07WTL3KPB/ref=sr_1_47?dchild=1&qid=1616985322&s=industrial&sr=1-47	$14.88
	5V charger	2	count	$6.00	https://www.amazon.com/Miady-5000mAh-Portable-Charger-Android/dp/B08T8TDS8S/ref=sr_1_5?dchild=1&keywords=Bar+Portable+Charger+5v&qid=1619726942&sr=8-5	$12.00
	NiHM Battery with the charger	1	count	$42.99	https://www.amazon.com/Tenergy-Battery-Projects-Equipments-Portable/dp/B08J4H39JV	$42.99
	acrylic sheets (3/16'')	2	sheet	$12.98	https://www.amazon.com/Source-Premium-Acrylic-PlexiGlass-S1-12x12-187/dp/B006WPXZZO/ref=sr_1_4?crid=29FTDBJLSR8TS&dchild=1&keywords=clear+acrylic+sheet+3%2F16&qid=1618785052&sprefix=clear+acrylic+sheet+3%2F%2Caps%2C256&sr=8-4	$25.96
	perforated plastic sheets (24'' x 24'')	1	sheet	$26.30	perforated plastic sheets | McMaster-Carr	$26.30
	rocker switch	1	count	$0.40	https://www.amazon.com/ZUPAYIPA-Solder-Rocker-Switch-	$0.40

Copyright © 20xx by ASME

					Toggle/dp/B01N2U8PK0/ref=sr_1_9?dchild=1&keywords=rocker+switch&qid=1617335663&sr=8-9	
	M5_T_nut	19	count	$0.37	https://www.mcmaster.com/5537T651/	$7.03
	M5X0.8_10mm screw	19	count	$0.96	https://www.mcmaster.com/91292A124/	$18.28
	M5X0.8_16mm rounded head screw	4	count	0.1754	https://www.mcmaster.com/94500A232/	$0.70
	M3X0.5_6mm screw	4	count	$0.19	https://www.mcmaster.com/90657A104/	$0.76
	M3X0.5_16mm screw	4	count	$0.23	https://www.mcmaster.com/90657A108/	$0.90
	M3X0.5_1.8mm_height nuts	4	count	$0.04	https://www.mcmaster.com/90695A033/	$0.15
	M5X0.8 nuts	4	count	$0.03	https://www.mcmaster.com/90695A037/	$0.13
	2-56''_3/8'length standoff	2	count	$0.46	https://www.mcmaster.com/91780A028/	$0.92
	2-56'' screw	4	count	$0.0574	https://www.mcmaster.com/91772A076/	$0.23
	3D printing	0.0496	kilogram	$100.00	https://www.fusion3design.com/how-much-does-3d-printing-filament-cost/#:~:text=PLA%20%26%20ABS%20(Generic%20Formulations),per%20kilogram%20from%20quality%20suppliers.	$4.96
Total cost for the project	$390.40					

This table represents the total cost for this project. The table is divided into sections based on different subsystems.

Copyright © 20xx by ASME

TABLE 4
Development Time

Member	time(hour)
Andrew	92
Gaetan	111.8
Hanna	87.5
Tianyu	98

This table represents the development times for each member based on the data from daily SCRUM.

TABLE 5
Physical Tilting Test

Trail	Plank length (inch)	height (inch)	angle (radian)	tilting angle (degrees)
1	39.75	21.6	0.574477486	32.91514712
2	39.75	21.9	0.583494252	33.43176994
3	39.75	22.2	0.592565015	33.95148661
Average	39.75	21.9	0.583512251	33.43280123

This table represents the measured parameters and results of the physical tilting tests.

ACKNOWLEDGMENTS

A special thanks to Jim Alkins for printing out our prototypes and being a great reference, Professor Muir for helping to guide our project towards success and asking the tough questions we needed to hear, Professor Howard for providing advice on every aspect of this project and providing this opportunity for us to learn more about robotic steering, and to Molly Over for being super helpful, especially with the laser cutter.

REFERENCES

[1] *Omnidirectional moving transfer robot with Mecanum wheels,* 唐炜刘勇李忠国顾金凤刘操于香志 (2016), CN105479433A, https://patentimages.storage.googleapis.com/c0/96/5e/11f4f23244c4ac/CN105479433A.pdf

[2] *Omnidirectional mobile robot-design and implementation, Doroftei I, Grosu V, Spinu V (2007),* B62D57/032, https://www.researchgate.net/profile/V_Spinu/publication/221786657_Omnidirectional_Mobile_Robot_-_Design_and_Implementation/links/5617985808ae90469c629690.pdf

[3] *Conformal suspension for unmanned ground vehicle (2015), Matther D Summer, Paul M Bosscher, Nicholas Murphy-DuBay,* US10065690B2, https://patentimages.storage.googleapis.com/a3/1d/cf/bac0473b22612a/US10065690.pdf

[4] *FUSION3 BLOG HOW MUCH DOES 3D PRINTING FILAMENT COST? Fusion3. (2021, April 27).* https://www.fusion3design.com/how-much-does-3d-printing-filament-cost/#:~:text=PLA%20%26%20ABS%20(Generic%20Formulations),per%20kilogram%20from%20quality%20suppliers.

Copyright © 20xx by ASME

APPENDIX

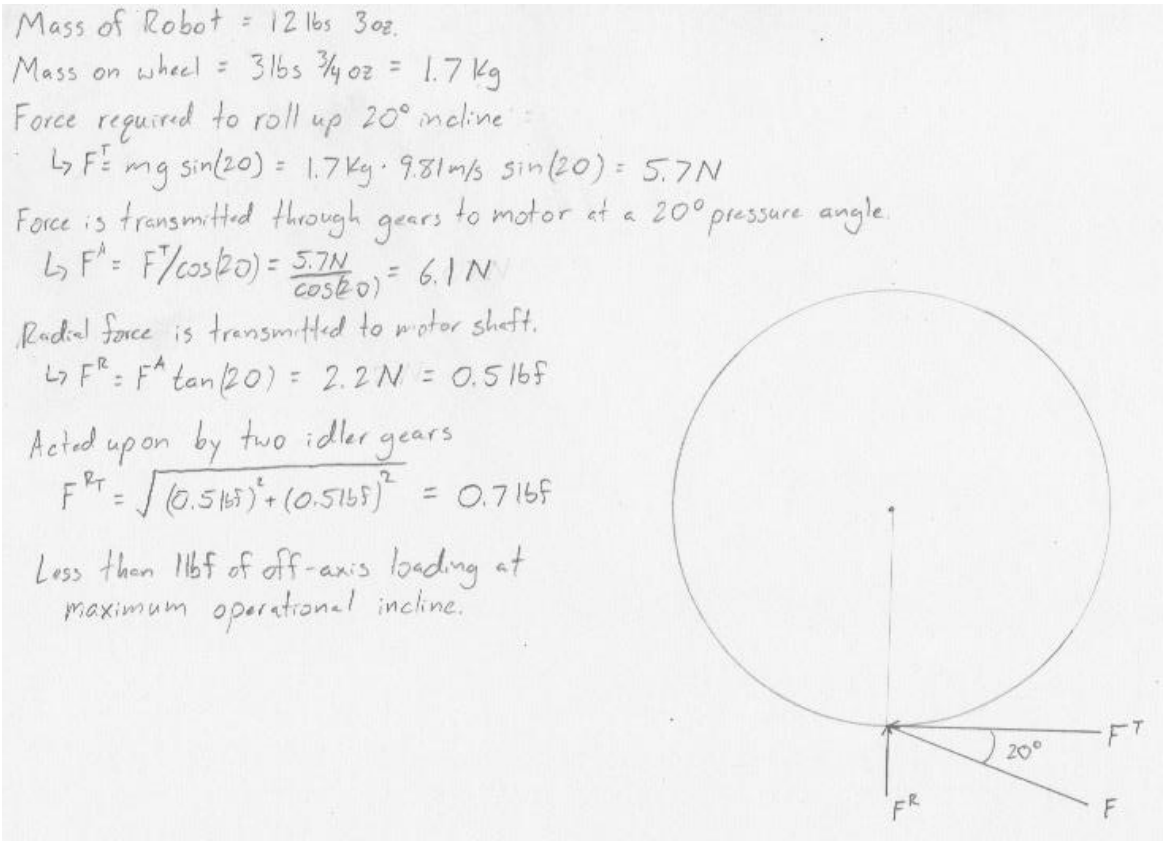

Figure 1. Off-Axis Load Calculation

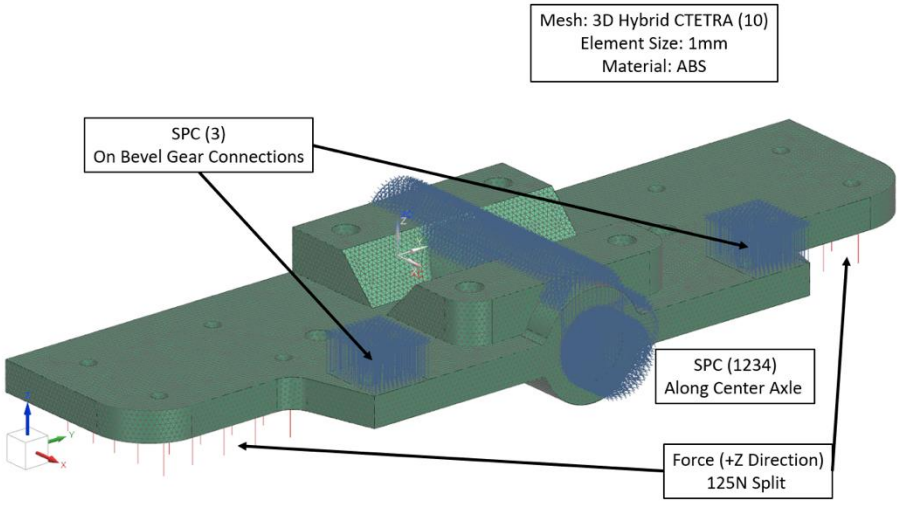

Figure 2. Initial Paddle Design Parameters (Top)

Copyright © 20xx by ASME

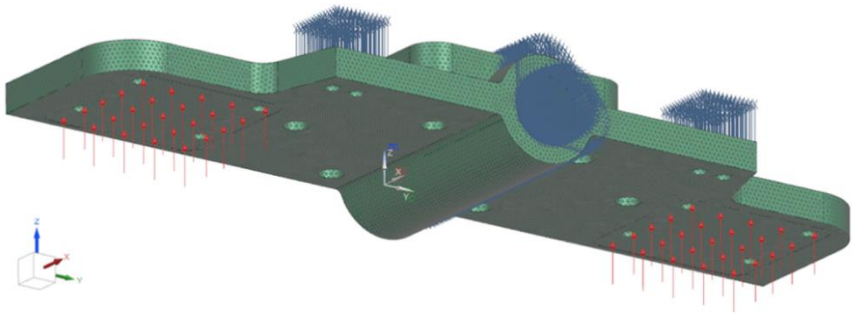

Figure 3. Initial Paddle Design Parameters (Bottom)

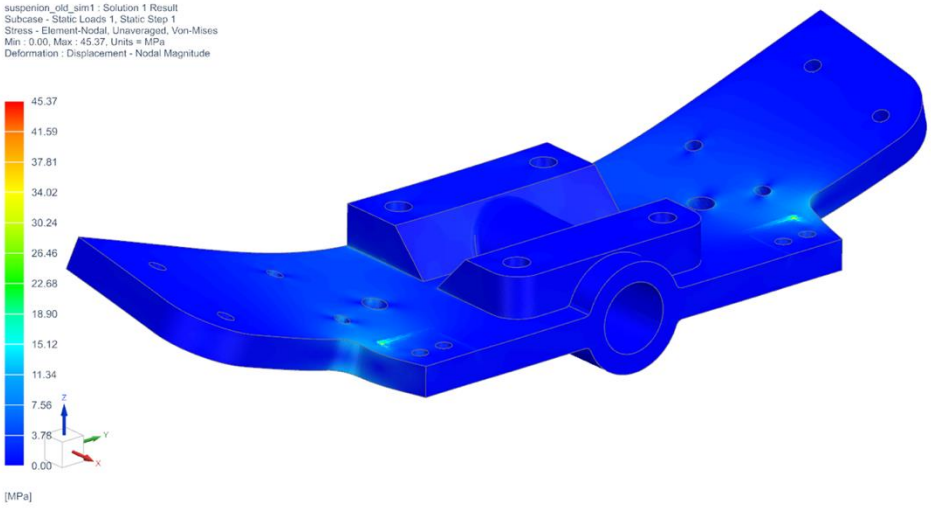

Figure 4. Stress Elemental-Nodal of Initial Paddle Design

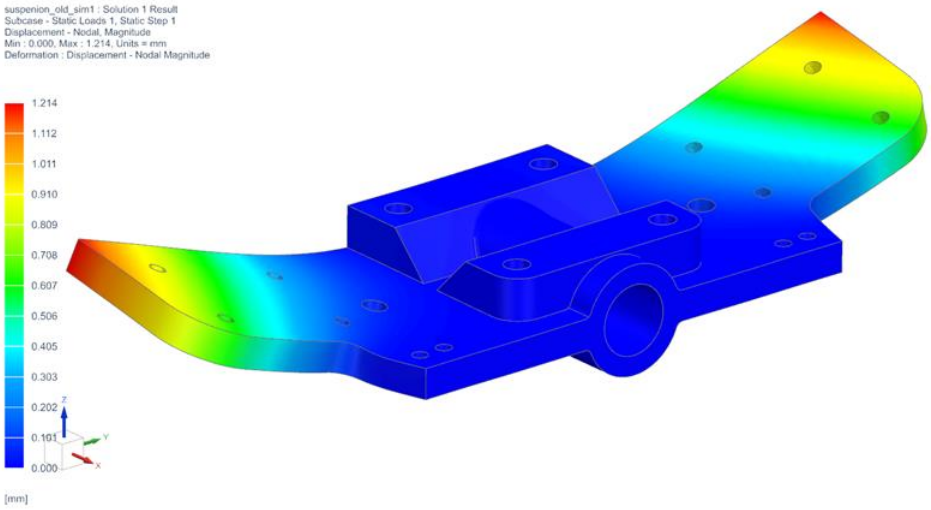

Figure 5. Displacement of Redesigned Paddle Design

Copyright © 20xx by ASME

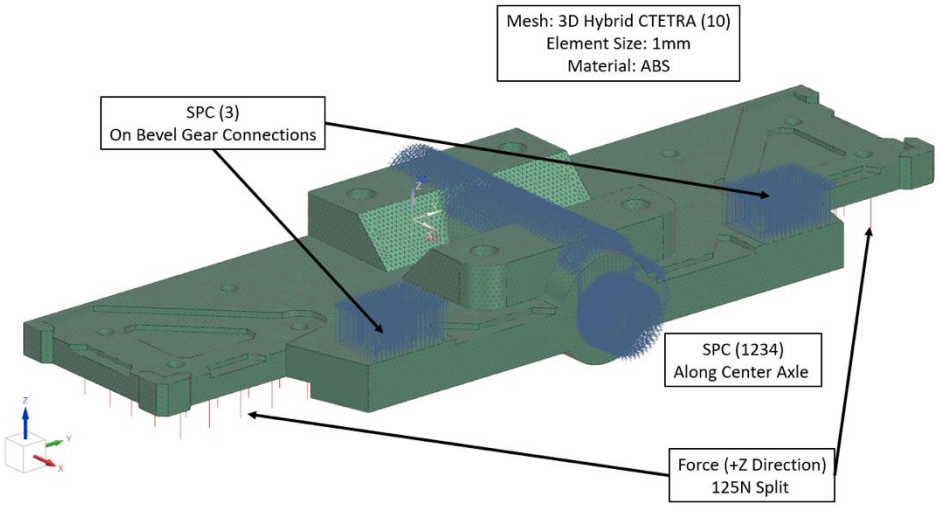

Figure 6. Redesigned Paddle Design Parameters (Top)

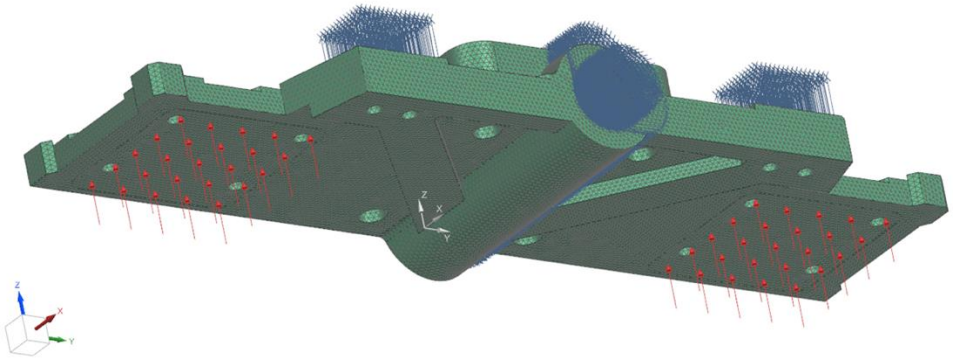

Figure 7. Redesigned Paddle Design Parameters (Bottom)

Copyright © 20xx by ASME

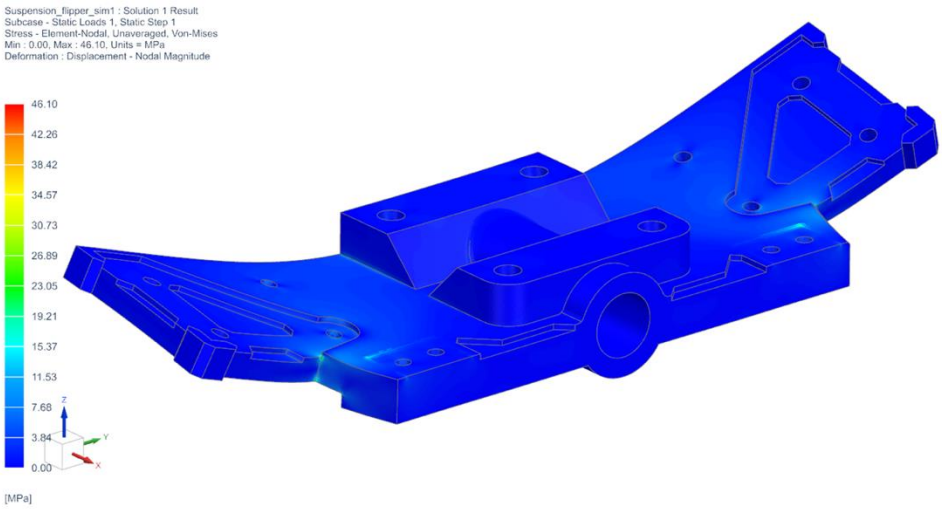

Figure 8. Stress Elemental-Nodal of Redesigned Paddle Design

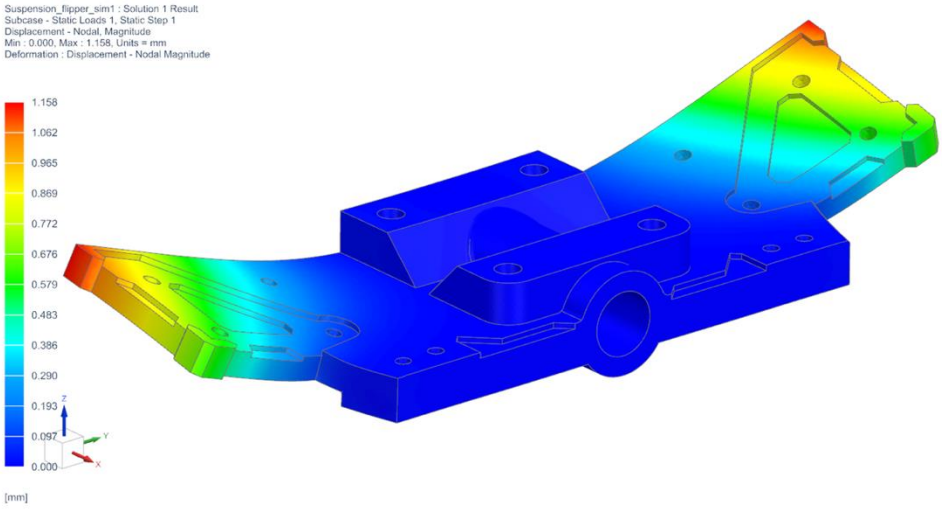

Figure 9. Displacement of Redesigned Paddle Design

Copyright © 20xx by ASME

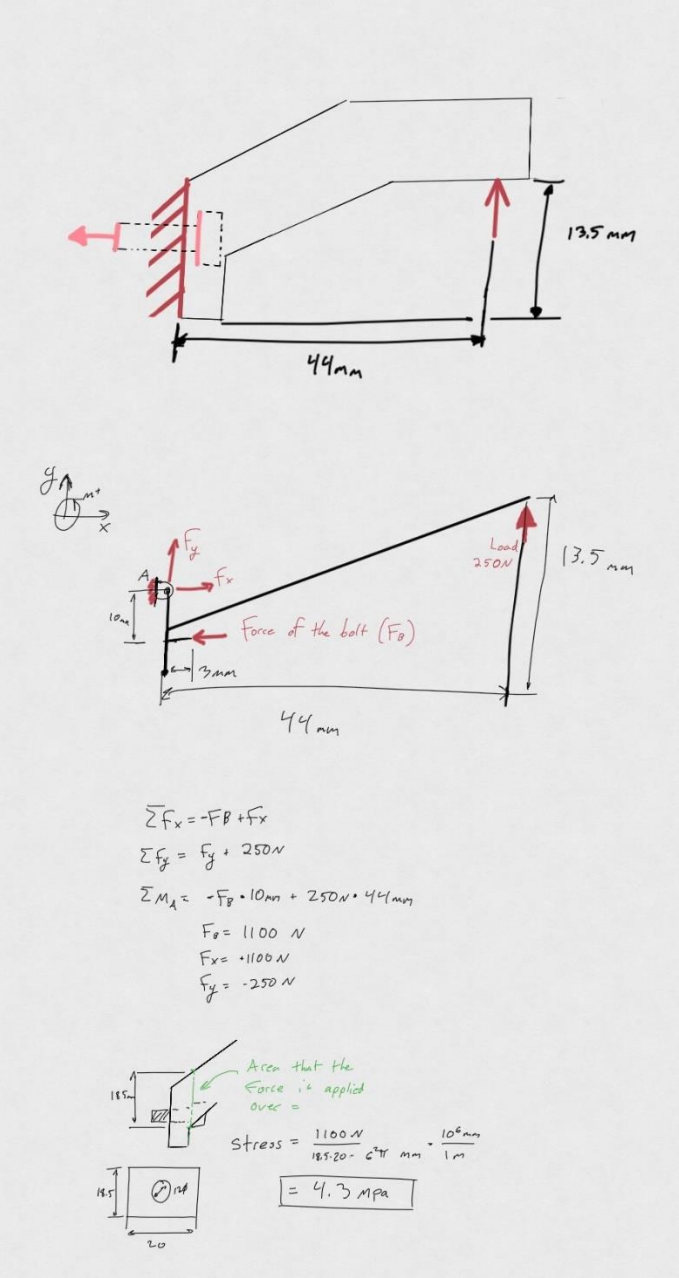

Figure 10. Stopblock Stress Calculations.

Copyright © 20xx by ASME

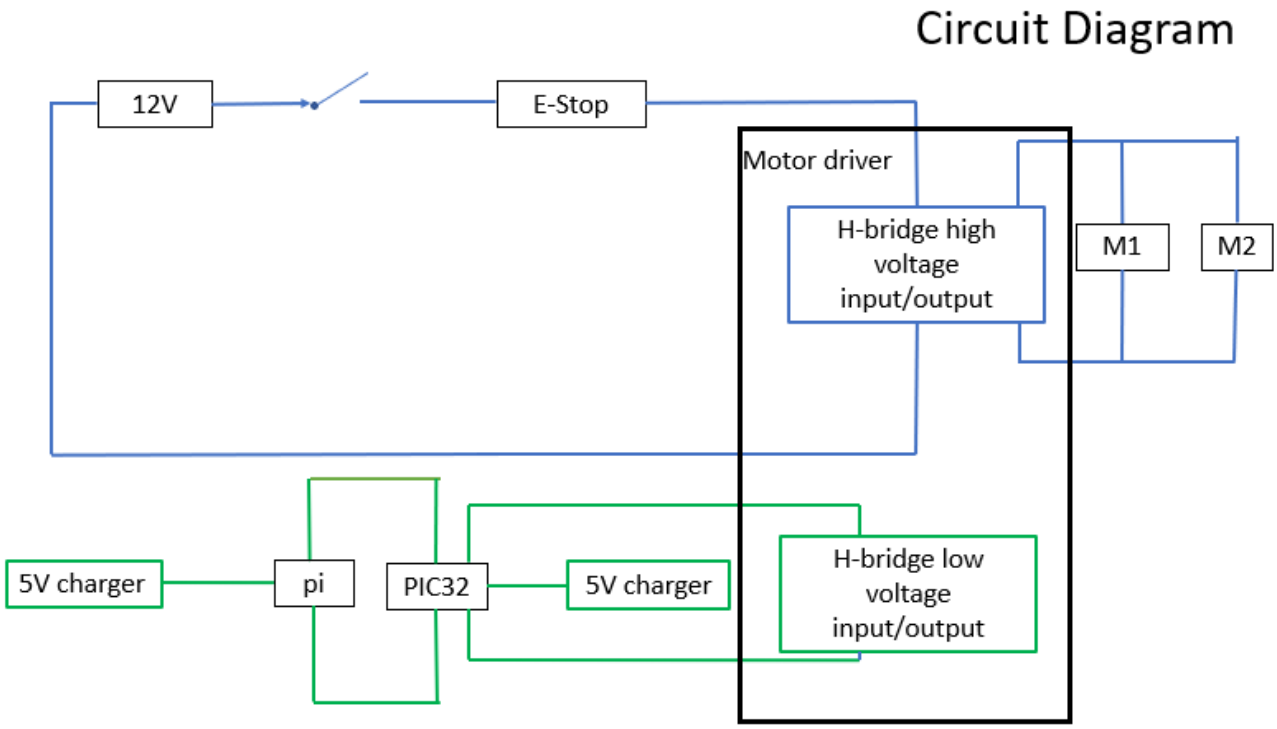

Figure 11. This figure shows a general logic of how the electronics devices are connected.

Copyright © 20xx by ASME

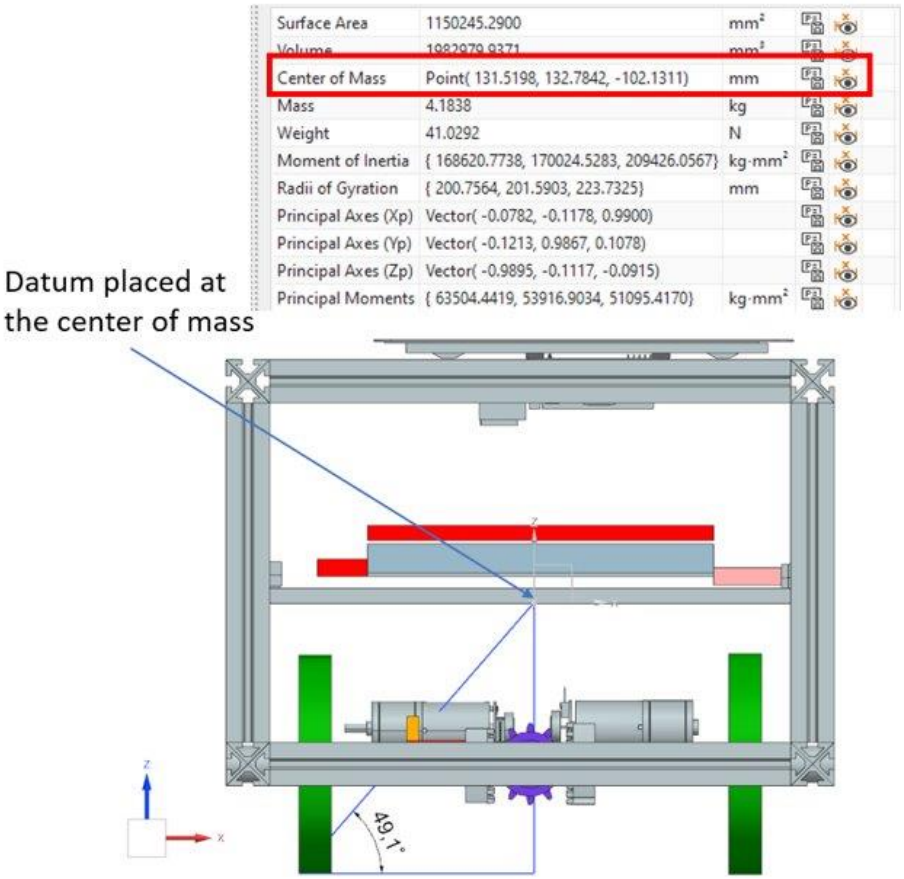

Figure 12. This figure represents the CAE analysis on tilting angle.

Figure 13. This figure shows the setups for physical tilting test.

Copyright © 20xx by ASME

CODE:

Main.c

```
/* Robotic Steering
 * File: main.c
 * Author: Hanna Saba
 * Date: 4/28/21
 * Description: main c file
 */

#include <xc.h>
#include <sys/attribs.h>
#include <string.h>
#include <stdio.h>
#include "display.h" // includes the functions defined in the header file


#pragma config FPLLMUL = MUL_20, FPLLIDIV = DIV_2, FPLLODIV = DIV_1, FWDTEN = OFF // sets up the frequency of the cpu and peripheral
#pragma config POSCMOD = HS, FNOSC = PRIPLL, FPBDIV = DIV_1 // Sets Periheral Bus Clock to 80MHz

#define VOLTS_PER_COUNT (3.3/1024)
#define CORE_TICK_TIME 25 // nanoseconds between core ticks
#define SAMPLE_TIME 10 // 10 core timer ticks = 250 ns

// creating static volatile integers to use throughout the program
static volatile int gamepad_x = 0, gamepad_y = 0;

int main(void){ // main function

    TRISFbits.TRISF0 = 0; // sets RF0 to input (pin87)
    TRISFbits.TRISF1 = 0; // sets RF1 to input (pin88) 78
    TRISEbits.TRISE0 = 0; // sets RE0 to input (pin93)
    TRISEbits.TRISE1 = 0; // sets RE1 to input (pin94) 77
    TRISDbits.TRISD4 = 0; // sets RD4 to input
    TRISDbits.TRISD5 = 0; // sets RD5 to input
    TRISBbits.TRISB15 = 0; // sets RB15 to input
    TRISA = 0xFF00;
    TRISE = 0xFF00; // sets the first 8 bits of register E to output
    DDPCON = 0x0000; // communicates with the explorer board


    lcd_display_driver_initialize(); // initialize the display

    INTCONbits.MVEC = 0x1; // enable multi-vector mode
    __builtin_disable_interrupts(); // disable interrupts

    // Setting up UART2
    U2MODEbits.BRGH = 0; // setting M to 16
    U2BRG = ((80000000 / 9600) / 16) - 1; // set the baud rate
    U2MODEbits.PDSEL = 0b00; // 8 bit no parity
    U2MODEbits.STSEL = 0; // 1 stop bit
    U2STAbits.UTXEN = 1; // configure TX & RX pins
    U2STAbits.URXEN = 1;
    U2MODEbits.UEN = 0;
```


Copyright © 20xx by ASME

```
52.    U2MODEbits.ON = 1;// turn on UART2
53.
54.
55.    // OC3, OC4, and TMR3
56.    PR3 = 1022; // given value of PR3
57.    TMR3 = 0; // initialize count to 0
58.    T3CONbits.TCKPS = 3; // set pre-scaler to 8
59.    OC3CONbits.OCTSEL = 1; // to set up timing
60.    OC3CONbits.OCM = 0b110; // PWM mode without fault pin; other OC1CON bits are defaults
61.    OC3RS = 0; // duty cycle
62.    OC3R = 0; // initialize before turning OC3 on; afterward it is read-only
63.    OC3CONbits.ON = 1; // turn on OC3
64.    OC4CONbits.OCTSEL = 1; // to set up timing
65.    OC4CONbits.OCM = 0b110; // PWM mode without fault pin; other OC1CON bits are defaults
66.    OC4RS = 0; // duty cycle
67.    OC4R = 0; // initialize before turning OC4 on; afterward it is read-only
68.    OC4CONbits.ON = 1; // turn on OC4
69.    T3CONbits.ON = 1; // turn on Timer3
70.
71.    __builtin_enable_interrupts(); //disables the interrupts
72.
73.
74.    while(1){ // loops until the change notification or timer goes off
75.
76.       LATA = 0b0000001;
77.       char xyval[10]; char message[20];
78.       ReadUART2(message,20);
79.
80.       lcd_display_driver_clear();
81.       lcd_display_driver_write(message);
82.
83.       LATA = 0b0000111;
84.       // splittng the string read from the raspberry pi
85.       char* direct = strtok(message, ":"); // setting GX or GY equal to direct
86.       char* num = strtok(NULL, ":");  // setting the number after equal to num
87.
88.       if(strcmp(direct,"GX") == 0){ // if direct is equal to GX
89.
90.          gamepad_x = atoi(num); // set gamepad_x to num
91.
92.       }else if (strcmp(direct, "GY") == 0){ // if direct is equal to GY
93.
94.          gamepad_y = -1*atoi(num); // set gamepad_y to num
95.
96.       }else{ // if direct is equal to anything else
97.
98.          lcd_display_driver_clear();// clears the display
99.          lcd_display_driver_write("Error");// print the string Error
100.      }
101.
102.      double xcon = gamepad_x/32767.0; // scale the value gamepad_x to be in between -1 and 1
103.      double ycon = gamepad_y/32767.0; // scale the value gamepad_y to be in between -1 and 1
104.
105.
106.
107.      double omega = (178.0*xcon);
```


Copyright © 20xx by ASME

```
108.      double VR = (ycon*178.0 + omega)/100.0/(1.8);
109.      double VL = (ycon*178.0 - omega)/100.0/(1.8);
110.      double speedL, speedR;
111.
112.      sprintf(xyval, "X:%5.2f Y:%5.2f", VR, VL); // create a string with the x and y values
113.      lcd_display_driver_clear();// clears the display
114.      lcd_display_driver_write(xyval); // write the string
115.      LATA = 0b00001111;
116.
117.      if(VR < -0.1){ // if VR is less than -0.1
118.         LATFbits.LATF1 = 0; // rotate in positive direction
119.         LATFbits.LATF0 = 1;
120.      }
121.      else if(VR > 0.1){ // if VR is greater than 0.1
122.         LATFbits.LATF1 = 1; // rotate in negative direction
123.         LATFbits.LATF0 = 0;
124.      }
125.      else{ // if VR is in between -0.1 and 0.1
126.         LATFbits.LATF1 = 0; // do not rotate
127.         LATFbits.LATF0 = 0;
128.      }
129.
130.      if(VL < -0.1){ // if VL is less than -0.1
131.         LATEbits.LATE1 = 1; // rotate in negative direction
132.         LATEbits.LATE0 = 0;
133.      }
134.      else if(VL > 0.1){ // if VL is greater than 0.1
135.         LATEbits.LATE1 = 0; // rotate in positive direction
136.         LATEbits.LATE0 = 1;
137.      }
138.      else{ // if VL is in between -0.1 and 0.1
139.         LATEbits.LATE1 = 0; // do not rotate
140.         LATEbits.LATE0 = 0;
141.      }
142.
143.   // code to try to lower values of VL and VR that exceed 1.0
144.
145.      //if (abs(VL) > 1){
146.      //    speedL = abs(VL)/abs(VL);
147.      //    speedR = abs(VR)/abs(VL);
148.
149.      //    OC3RS = speedL*1023.0; // set OC3RS
150.      //    OC4RS = speedR*1023.0; // set OC4RS
151.      //}
152.      //else if(abs(VR) > 1){
153.      //    speedL = abs(VL)/abs(VR);
154.      //    speedR = abs(VR)/abs(VR);
155.
156.      //    OC3RS = speedL*1023.0; // set OC3RS
157.      //    OC4RS = speedR*1023.0; // set OC4RS
158.      //}
159.      //else{
160.      //    OC3RS = abs(VL)*1023.0; // set OC3RS
161.      //    OC4RS = abs(VR)*1023.0; // set OC4RS
162.      //}
163.
```


Copyright © 20xx by ASME

```
            OC3RS = fabs(VL)*1023.0; // set OC3RS
            OC4RS = fabs(VR)*1023.0; // set OC4RS

        IFS1bits.U2RXIF = 0; // clear the RX interrupt flag
    }

    return (1);

}
```

display.c

```
/* Robotic Steering
 * File: display.c
 * Author: Hanna Saba
 * Date: 4/28/21
 * Description: Implements the functions
 */


#include "display.h"
#include <xc.h>
#include <string.h>

void ReadUART2(char * message, int maxLength){

    char data = 0;
    int complete = 0, num_bytes = 0;

    // loop until you get a ?\r? or ?\n?
    while (!complete){

        if (U2STAbits.URXDA){ // if data is available
        data = U2RXREG; // read the data
        LATA = 0b0000011;
            if ((data == '\n') || (data == '\r')){
                complete = 1;
            }
            else{
                message[num_bytes] = data;
                ++num_bytes;
                // roll over if the array is too small
                if (num_bytes >= maxLength) {
                    num_bytes = 0;
                }
            }
        }
    }

    // end the string
    message[num_bytes] = '\0';
}

void delay(int x){ // delay function
    int a = 0; // creates integer a and sets it to 0
```


Copyright © 20xx by ASME

```
    while(a < x){ // creating a delay that repeats 100 times
        a = a+1; // increase integer a by 1
    }
}

void lcd_display_driver_enable(){ // implementing function to enable display

    LATDbits.LATD4 = 1; // enable the display by setting RD4 to 1
    delay(1000);
    LATDbits.LATD4 = 0; // disable the display by setting RD4 to 0
    delay(1000);

};


void lcd_display_driver_initialize(){ // implementing function to initialize display

    // function Set found in Novateks Single-Chip 16C X 2L Dot-Matrix LCD Controller / Driver (NT7603)
    LATDbits.LATD5 = 0; // sets RD5 to 0
    LATBbits.LATB15 = 0; // sets RB15 to 0 (display input)
    LATE = 0b00111000; // sets display to 2-line mode and 5x7 dots
    lcd_display_driver_enable(); // calls on the enable function

    // display on/off
    LATDbits.LATD5 = 0; // sets RD5 to 0
    LATBbits.LATB15 = 0; // sets RB15 to 0 (display input)
    LATE = 0b00001100; // turns on display, turns off cursor, turns blinking off
    lcd_display_driver_enable(); // calls on the enable function

    // clear display
    LATDbits.LATD5 = 0; // sets RD5 to 0
    LATBbits.LATB15 = 0; // sets RB15 to 0 (display input)
    LATE = 0b00000001; // clears the display
    lcd_display_driver_enable(); // calls on the enable function

    // entry mode
    LATDbits.LATD5 = 0; // sets RD5 to 0
    LATBbits.LATB15 = 0; // sets RB15 to 0 (display input)
    LATE = 0b00000100; // sets to decrement mode and turns shift off
    lcd_display_driver_enable(); // calls on the enable function


};


void lcd_display_driver_clear(){ // implements the clear function

    // clear display
    LATDbits.LATD5 = 0; // sets RD5 to 0
    LATBbits.LATB15 = 0; // sets RB15 to 0 (display input)
    LATE = 0b00000001; // clears the display
    lcd_display_driver_enable(); // calls on the enable function

};


```


Copyright © 20xx by ASME

```
100.
101.void lcd_display_driver_write(char* data){ // function that writes char data on display
102.    int length = strlen(data);// creates int and sets value to length of char data
103.    int j = 0;// creates int j and sets value to 0
104.    while ( j < length ){// while loop for when j is less than length
105.        LATDbits.LATD5 = 0; // sets RD5 to 0
106.        LATBbits.LATB15 = 1; // sets RB15 to 0 (display input)
107.        LATE = data[j];// sets LATE to character
108.        lcd_display_driver_enable(); // calls on the enable function
109.        j = j+1;
110.    }
111.}
```

display.h

```
1.  /* Robotic Steering
2.   * File: display.h
3.   * Author: Hanna Saba
4.   * Date: 4/28/21
5.   * Description: Defines the functions
6.   */
7.
8.
9.  #ifndef LCD_DISPLAY_HEADER_H
10. #define  LCD_DISPLAY_HEADER_H
11.
12. void ReadUART2(char * message, int maxLength);
13.
14. void delay(int x); // delay function
15.
16. void lcd_display_driver_enable(); // function to enable display
17.
18. void lcd_display_driver_initialize(); // function to initialize display
19.
20. void lcd_display_driver_clear(); // function to clear display
21.
22. void lcd_display_driver_write(char* data); // function to write a character
23.
24. #endif
```


Copyright © 20xx by ASME