Group 12

AUTONOMOUS STEEL-COLUMN CLIMBING ROBOTS

Senior Design Project – Winter 2023

April 10th, 2023

Final Report Draft

Prepared for:

ME 4999 - Dr. Stephen Bazinski

ECE 4999 - Dr. Jun Chen

Submitted by:

Name	Major	Leader	Section
Jacob Baehr	CE	ECE	Computer Subsystem Overview, Computer Subsystem (ROS, Detection Solutions, PID), Detailed Code Listing (Python, Motor Control)
Jacob Mattern	ME	ME	Design Overview (Overall, Mechanical Subsection), Discussion (Safety, Ethical Considerations, Economic Factors, Reliability, Aesthetic, Potential Customers, Societal and Global Impact)
Mikela Marku	E/CE		Description of the project, EE Requirements, Power Delivery (Buck Power Diagram), Detailed Circuit Schematics of Each Component, Discussion
Jonah Moore	EE		Electrical Subsystem Overview, Gantt Chart creation(electrical, computer, and mechanical subsystem), power simulation diagram/graphs
Salvatore Narusch	EE		Computer Subsystem (initial design concepts, image classification, edge/contour detection, Gyro Sensor), Discussion (challenges), Conclusion & Recommendation (Comment on Project)
Colin Steed	EE		Electrical Overview, Battery Standards, EE requirements, Electrical Subsystem (Power Delivery, Motors, Sensors, Microcontroller)
Katty Tran	ME		Introduction, Design Specifications (ME Specification), Mechanical Subsystem (entire), Electrical & Computer Subsystem (BOM), Discussion (Technical Discussion, Total BOM, Gear Standards), Conclusion & Recommendation (Overall Solution), References, Appendices

ME 4999 – ECE 4999 – Senior Project

OAKLAND UNIVERSITY

School of Engineering and Computer Science

Table of Contents

I. INTRODUCTION

1. Description of the Project

To replace human labor in dangerous environments or unfavorable terrain, autonomous climbing robots are a satisfactory and practical solution. Not only effective in terms of safety for workers, climbing robots also bring benefits in terms of cost and help to save time. More importantly, by being programmed, the climbing robot can not only collect data and provide accurate information, but also perform tasks quickly and accurately.

The goal of this project is to design and build an automatic climbing robot that can perform the following operations:

- Task 1: starting at a starting line on the floor of the lab.
- Task 2: finding the steel column.
- Task 3: climbing on the steel pole.
- Task 4: ringing the bell hanging on the top of the steel column.
- Task 5: climb back to the ground.
- Task 6: return to starting position.

The above 6 tasks are the customer's needs and function, summed up in a task that is to design an automatic robot that can climb the I-beam, ring the bell hanging on it and return to the starting position. In terms of engineering requirements, speed and safety are the most important

factors for this project. Although there are no specific specifications given, a wireless video camera is required for machine vision and to collect the robot's journey data if possible. As an automated product, the bottom line is that the robot must operate on its own without human interaction.

2. Description of the Design Solution

The ideas offered revolve around applying the attraction of magnets to the steel column. In terms of mechanical operation, a device with a magnetic tank track as in Figure 1, or with magnet wheels as in Figure 2 is suggested. A design for a mechanical structure system that combines body/frame, gear, axles, etc. is needed.

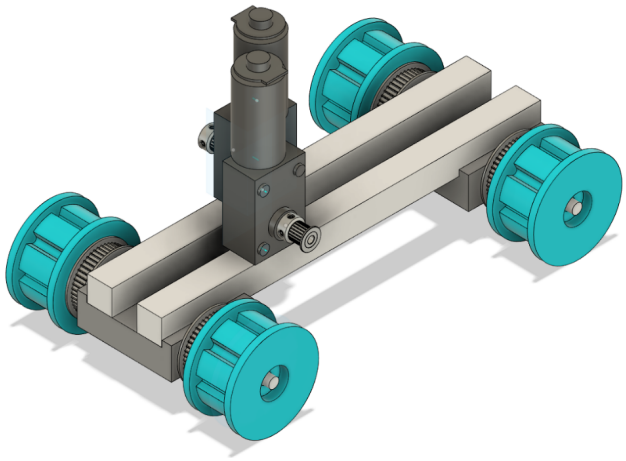


Figure 1. Rough Design of Magnetic Tank Track

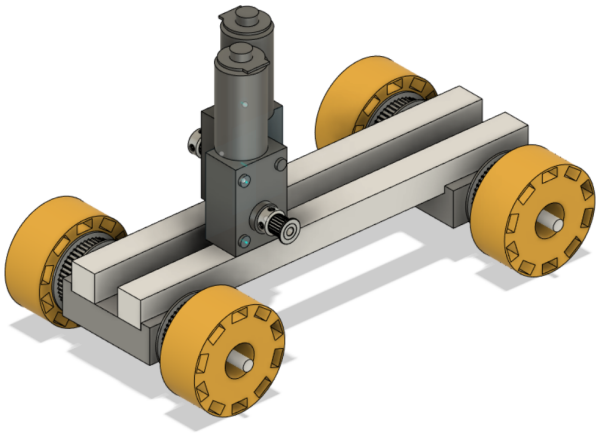


Figure 2. Rough Design of Magnetic Wheels

In terms of computers and programming, there are a number of electronic devices required for the control system and the robot-environment interaction system such as motors, batteries, sensors, microcontrollers, vision hardware, vision control, and wires.

The entire project must be carefully prepared by doing market research, finding information & standards, sorting and comparing the pros and cons of ideas, and choosing the

most feasible solution. Product ideas must be approved by a panel of experts (professors) and approved by the customer before being assembled and operated.

II. DESIGN SPECIFICATION

Table 1. Engineering Design Requirements and Validation

	Engineering Requirement	Importance	Target	Method
ME	Weight	High	≤ 5lbm	CAD analysis, inspection
ME	Volume	Moderate	≤ $1ft^3$	CAD analysis, inspection
EE	Input Power	Moderate	< 22V	Pspice analysis, demonstration
EE	On/Off Switch	High	on handle	CAD analysis, inspection
EE	Battery Life	High	> 20 min	Estimate through analysis
Team	Development/Tooling Cost	Low	< $22	Estimate through analysis
Team	Design Project Cost	High	< $500	Budget analysis
EE/CE	Object Detection Software	High	x% error rate	Inspection, testing
EE/ME	Ring Bell (Robot's Arm)	High	Single joint	Testing

1. ME Requirements

The weight of the robot and the magnetic force on the steel beam are the key factors in finding the minimum torque required for the motor. The smaller the weight, the more energy is saved for the battery. The ideal condition to determine the total minimum torque needed for the motors is when the robot climbs on the beam because the motor torque has to overcome both the weight of the robot and the magnetic force. Therefore, limiting the robot's weight as much as possible is one of the key focuses of the project. A weight of 5lbm or less will be pursued if possible.

Compact design is one of the mechanical engineering specifications of the project. The smaller the volume, the easier it is for the robot to move on all terrains and tight spaces. This increases the robot's level of efficiency, convenience, and aesthetics. Furthermore, the limited volume as much as possible also means savings in terms of weight. A volume of one cubic foot

is based on the fact that the steel beam the robot will be climbing is slightly larger than one foot in width. Given this, the width of the robot should be less than one foot and the design will have a larger length and smaller height compared to the width. The ideal volume will be equal or less than $1ft^3$.

Ringing the bell is one of the high priorities due to it being integral to the main objective of the robot. With this in mind, a simple design incorporating a single joint would be most effective.

2. EE Requirements

The electrical design requirements of a mobile robot are particularly important, as a tethered robot is not as flexible in nature. The two most important features of a mobile robot are the battery life and the input power.

The input power refers to the input voltage the robot will receive. This characteristic largely depends on the goals and components of the robot, however in a case like ours the majority of our components operate within the range of 5V - 12V. This is a common range to be in and there is a plethora of batteries that can achieve this. Though a common issue with running on a battery is the voltage drops off as the battery is drained of power. For example, a 12V battery is actually at about 12.5V when fully charged, though by the time it is 50% depleted, it may only be at 11V. This is a clear issue if something in the robot runs off of 12V. To mitigate this, a common practice is to start with a high input power, let's say 14.8V, and step down the voltage to fit the needs of all the components. This will ensure adequate power throughout the life of a single charge.

Battery life is another factor that can make or break the functionality of a robot. When thinking about the battery life that is required, a good start is to understand how long your robot needs to be on in order to effectively complete its task. In our case at hand, a 15 minute window should be more than enough time to do what we need to do. Now that a common range is found, gathering the current consumption of each component that will be using power will determine how much amperage it will consume. This robot is calculated to be at about 3.89A assuming max power drawn on each component. To calculate how long a battery size will last, based on a known amperage, the following equation can be used:

$$(Battery\ Size\ mAH\ /\ Amperage\ Used\ (A)\ )\ =\ Battery\ Life\ Minutes$$

In our case:

$$(1300\ \ mAH\ /\ 3.89\ A)\ =\ 15\ minutes$$

3. CE Requirements

The computer engineer requirements of the robot include its ability to be repeatable and flexible for other solutions. With the incorporation of ROS and Object detection, the robot maintains these requirements.

The Robot Operating System gives the ability of many devices to simultaneously work in conjunction with one another. This gives the robot a great deal of flexibility for future interactions and upgrades. Allowing for the robot to be built on a platform to be used in many different scenarios.

Object detection allows the robot to stay repeatable. As a form of machine learning, the robot could in theory continue training based on successes or failures. Having an open ended solution that gets better over time, could prove to become very repeatable after many iterations.

4. Team Requirements

With the initial idea of applying magnetic wheels, color or distance sensors, and object detection techniques, we did some research on the cost of the devices that support those techniques. The total cost is estimated to be around $500. Therefore, the goal we set for the total budget is less than $500. Since we decided to use the 3D-printing technique for the designed parts, the total tooling cost will basically be the material cost which is estimated to be less than $20.

III. DESIGN OVERVIEW

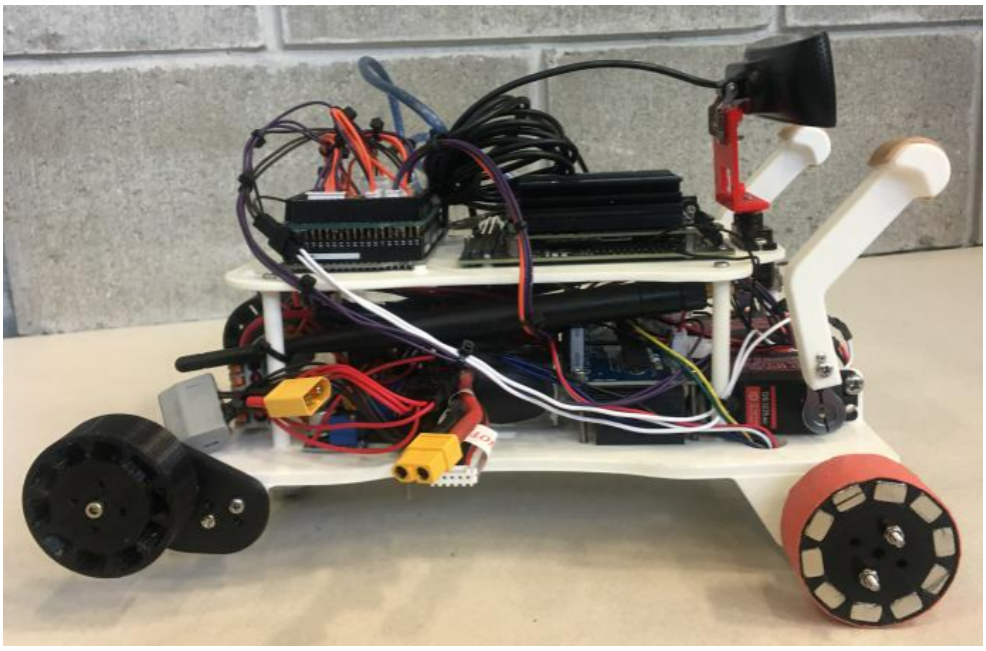


Figure 3. Side View of the Actual Robot.

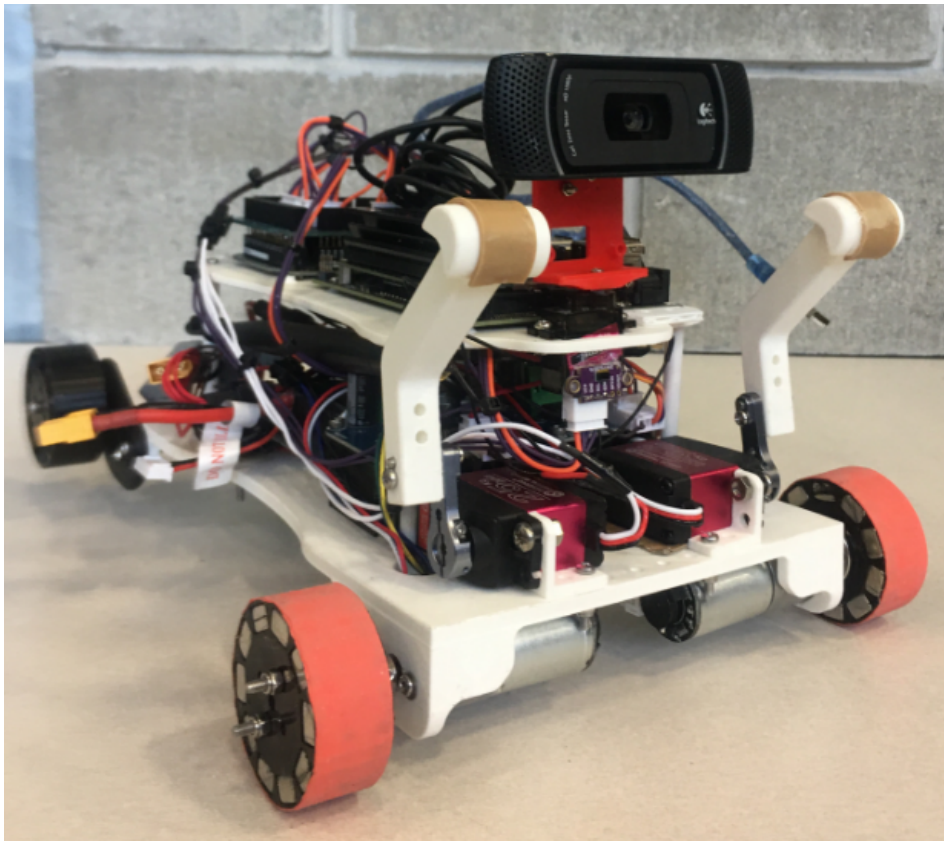


Figure 4. Front View of the Actual Robot.

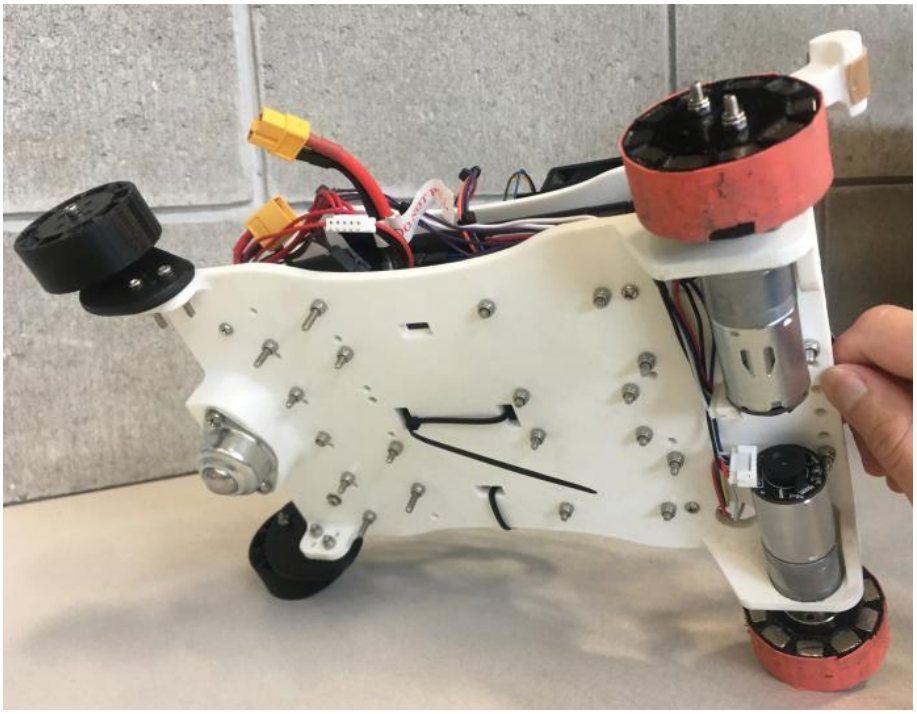


Figure 5. Bottom View of the Actual Robot.

The design solution utilizes a pair of wheels in the front and one ball-universal wheel in the rear to ensure the basic movement of the robot as shown in Figure 5. Using object detection techniques to locate the beam, the robot then can stick to the beam and climb up thanks to the attraction of the magnets attached to the front wheels as shown in Figure 3. After ringing the bell by bumping the frame into it, the robot travels down the beam in a similar manner as it ascended. The pushing arms attached at the front can rotate and push against the beam to create the leverage needed to detach the magnets from the beam so that the robot can return to its starting position, completing its task.

1. Mechanical Subsystem

The 3D-printed frame of the robot is separated into two layers to provide more room for the electrical components while still being able to fit onto the beam. Both layers have several holes that second the electrical components and allow wires to travel between the layers safely and securely. The bottom frame is the main skeleton to connect all the wheels, electrical, and computer components together, ensuring enough hardness to endure any movement of the robot. The top frame is designed to support the "brain" components of the robot such as the microcontroller, jetson, and a micro servo to mount with the camera, allowing it to turn and search for the beam from a variety of angles. There are four 3D-printed support pillars that are used to separate and connect the two frames. The 3D-printed pushing arms positioned at the front of the robot serve to produce a force that assists in removing the magnetic wheels from the beam.

The movement of the robot is provided by a set of motors attached to a pair of 3D-printed wheels in the front and a steel ball-universal wheel at the rear of the robot. The magnets attached along the rim of the front wheels ensure the movement of the robot on the beam by the attraction between the magnets and the steel beam. The front wheels are rigidly mounted to 2 DC motors so that they can rotate based on the torque adjusted by the electrical subsystem and controlled by the computer subsystem. Another two 3D-printed wheels are set to each side of the robot's rear, providing a pivot, and preventing the back end of the bottom frame from scraping on the ground when it transitions onto or off of the beam.

2. Electrical Subsystem

The way our robot is powered is through a lithium polymer battery. The battery is then routed to five DC-DC converters which converts the battery power into an appropriate level of power for each electrical component. The power is then routed to each respective component through harnesses, using connectors to ensure integrity of the wire connections.

To control and interface with our electronics, we used an Arduino Mega. Every sensor is routed to the Mega, connected with connectors that have been soldered onto the prototype PCB. This prototype PCB is seated in the microcontroller, ensuring that proper contact has been made between the microcontroller and the respective component.

Sensors, such as distance and gyroscope/accelerometer sensors are also implemented throughout the system. These will assist in the location of the beam and the robot’s required position before it is able to perform the required task of climbing.

3. Computer Subsystem

The software of the robot was built with future development in mind. It uses common software standards in the realm of robotics and implements similar solutions found often. Such standards include a Linux based operating system known as Ubuntu, along with a common meta-operating system for robot development known as Robot Operating System. Ubuntu is a lightweight Linux kernel that is often used on single board computers, while paired with a framework like ROS, provides great flexibility for future development.

On top of the systems used for vision, C++ scripts will be used on the microcontroller, from which will be handling the motor control and robot states. The communication between these two will be initiated through a ROS node, which is a script that operates as a Publisher or Subscriber of information.

IV. MECHANICAL SUBSYSTEM

1. Detailed Description of the Mechanical Subsystem

The components purchased for the mechanical subsystem include magnets, rigid flange couplings, ball-universal wheel, screws, and nuts. The rest of the mechanical components are drawn in CATIA and manufactured using the 3D printing method such as the upper and lower frames, front and rear wheels, mounting pad, and brackets.

a. Initial Concept and Final Solution

i. Robot's Movement Method

There are 2 solutions for the subsystem responsible for the movement of the robot: magnetic tank track and magnetic wheels as shown in Figures 6 and 7.

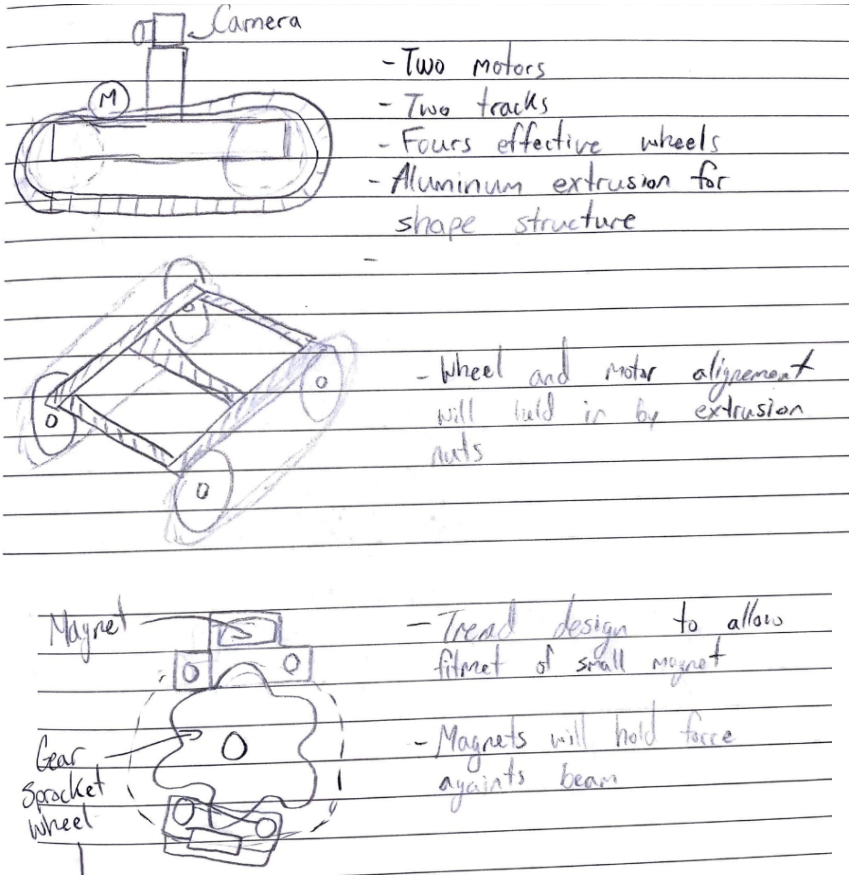


Figure 6 . Initial Tank Tread Design

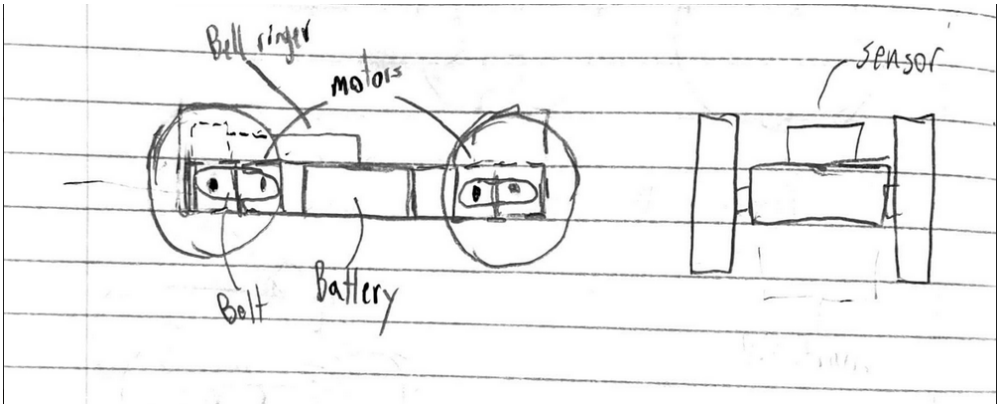


Figure 7 . Magnet Wheel Initial Sketch

Both the Tank and Wheel design are going to have similar approaches when it comes to designing, prototyping, testing, and integrating. For one, each solution will share the same robot frame design, this is the case due to the use of aluminum extrusions and custom plastic parts. The major difference between these two is how the drive system is designed and especially in regard to magnet placement.

The Tank design will have magnets all throughout the treads on each side of the robot. Due to this, magnet strength is not much of a concern as there will likely be about 20 points of contact at any given moment. The team will have to prototype and test which methodology is best for encasing the magnets in the treads.

The Wheel design on the other hand will have a more simplistic view, as each wheel will simply have slots around the circumference of the wheel. One major concern with this, as prototyping will answer, is the strength of the magnetization, as there are effectively only four points of contact with this design.

Other hurdles that occur in both of these include whether there will be an rolling effect when the robot is vertical, if there will be any sliding effect when the robot is vertical, if the robot will be able to dismount effectively from the column, and if the weight of the magnets will push the limit of our chosen motors.

Alternative approaches that the team had briefly discussed, included having a mechanical approach that involved motors gaining leverage on the column to crawl the robot upward. This would be similar to a monkey or spider, grabbing the outside edges of the beam. This approach was quickly dismissed as the motors would experience long periods of hold current, which would drain the battery at unacceptable rates.

Table 2 shows the Pugh Matrix to compare these solutions.

Table 2. Pugh Matrix of Robot's Movement Methods.

Criteria	Baseline	Magnetic Tank Tread	Magnetic Wheels
Speed	4	0	3
Weight	3	0	2
Cost	3	3	3
Maneuverability	2	0	2
Mobility	3	2	1
Materials	2	0	1
Simplicity	2	-1	2
Power Efficiency	4	1	0
Traction	3	3	0
Aesthetics	0	2	1
Ground Impact	0	2	1
Weight Growth Potential	2	2	0
Total		14	16

The wheels need a lower amount of torque to move on from stationery or to speed up and down. However, due to more friction and a complex mechanical system, the tank track will have lower speed compared to wheels [2]. Since speed is one of the important goals of this project, the slowness of the tank system gets a minus point. In terms of maneuverability, the wheel system operating at a higher speed will provide greater maneuverability than the tank track. However, the all-terrain adaptability of the wheels cannot be compared with that of a tank track, so the mobility of the wheel system is deducted 1 point.

Thanks to the optimized traction system with high power efficiency, the tank track system achieved a plus point in terms of traction and load capacity. However, these criteria do not play a central role in the project. In terms of aesthetics, there are many opinions that tank tracks give power beauty and look more aggressive than the wheels. Again, this advantage of the tank is not the top criterion of the customer's need. On the other hand, what customers focus on is the cost, so the wheel with low production costs is a plus point. In terms of mechanical design, the wheel has fewer moving parts, which means that there are fewer components, and is simpler than the tank track.

After all the discussion and careful consideration from the Pugh matrix, the team had come to the decision to choose the magnetic wheel as the solution for the robot's movement.

ii. Front Wheels

There are 2 magnetic front wheels needed for the robot that can be designed in CAD and manufactured using the 3D-printing method. The original wheel is designed with holes to mount the rigid flange coupling and square holes along the rim for the magnets as shown in Figure 8.

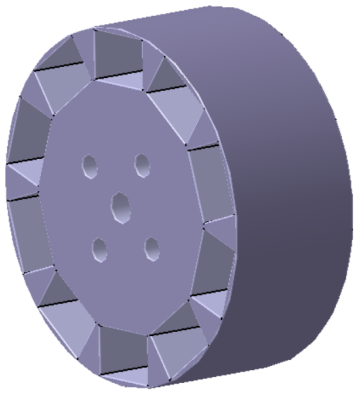


Figure 8. Non Side-hole Wheel

Even though the specifications of the magnets are provided, in our design, the magnetic force to the steel column will be rapidly reduced by the presence of the plastic layer of the wheel rim. Therefore, the design of the front wheels is changed with a configuration that limits the plastic layer between the magnets and the steel surface as much as possible. A square side hole is added along the rim for each magnet as shown in Figure 9.

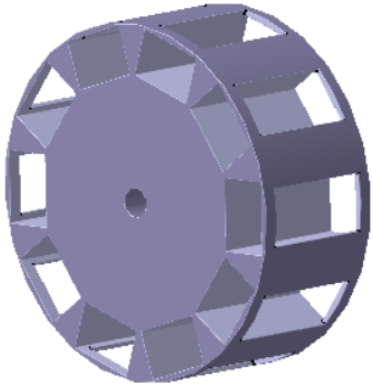


Figure 9. Square Side-hole Wheel

Then, the design of the side holes is upgraded to oval shapes in order to hold the magnets firmer as shown in Figure 10.

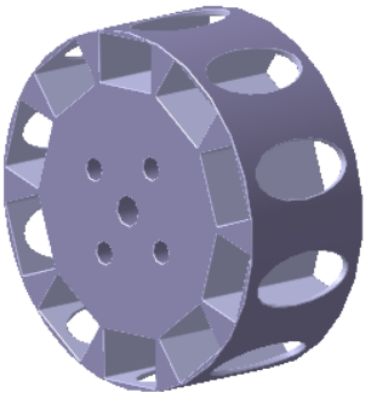


Figure 10. Oval Side-hole Wheel.

The testing shows that the magnetic force on the side doesn't really affect the attraction of the magnetic wheel to the steel beam. On the other hand, because we planned to cover a layer of rubber along the rim to increase friction, the surface of the rim needs to be smooth to limit the rolling resistance. Therefore, the team had come to the decision to choose the original design (non-side-hole wheel) as the solution for the robot's front wheel.

iii. Rear Wheels

There are 3 solutions considered for the rear movement of the robot: ball-universal wheel, 3D-printing wheels, and non-swivel caster wheel.

The flexible rolling motion of the ball makes the movement of the robot smoother, minimizing friction when the robot changes direction. In addition, a small ball-universal wheel mounted in the middle of the rear part of the robot saves more weight and cost than using two rear wheels.

Figure 11. Ball-Universal Wheel [1].

One downside of the ball universal wheel is that it doesn't solve the problem of getting off the beam of the robot. Because the robot will reverse when it climbs down the beam, the ball-universal wheel will be the front wheel during the climb-down movement. However, it is

attached underneath the bottom frame, so the bottom frame will reach the ground first and the robot will get stuck. Therefore, the plastic 3D-printed wheels mounted sticking out of the frame a bit as shown in Figure 9 will touch the ground first instead of getting stuck by the frame. A long bolt secured with nuts and washers will be the shaft for this wheel.

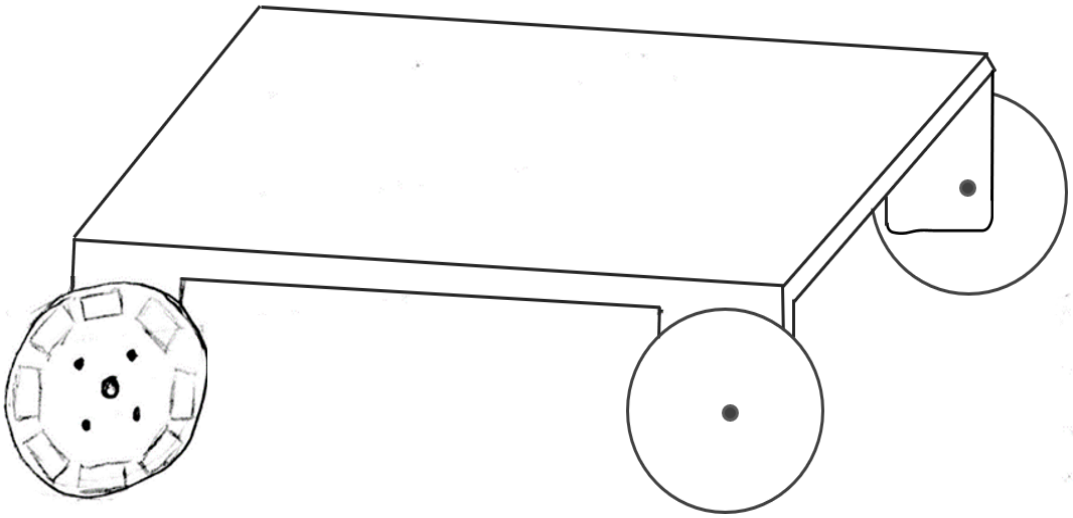


Figure 12. Oval Side-hole Wheel.

However, based on the force analysis, the 3D-printed wheels rotating around the shaft can create friction since the surfaces of the shaft and the inner surface of the wheel's hole are not really smooth as we expected. That will take more power from the motor. The caster wheels can save the power of the motor so the robot can move smoother and faster.

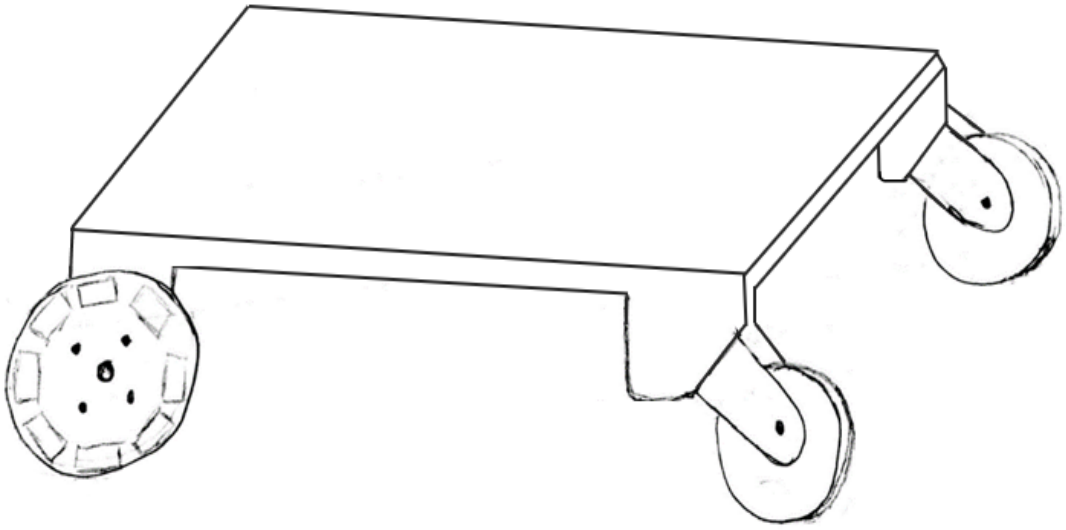


Figure 13. Oval Side-hole Wheel.

Table 3 shows the Pugh Matrix to compare the three solutions for rear part of the robot.

Table 3. Pugh Matrix of Robot's Rear Wheels.

Criteria	Baseline	Ball Universal Wheel	3D-printing wheel	Non-Swivel Caster Wheel
Speed	4	4	2	3
Weight	4	3	4	2
Cost	3	2	3	-1
Mobility	4	3	1	2
Simplicity	2	2	1	1
Power Efficiency	3	3	1	2
Traction	3	1	2	3
Aesthetics	1	1	0	1
Getting off the beam	4	-1	2	4
Total		18	16	17

Since the caster wheel is made from steel and plastic, its weight is a minus point compared to the others. Even though the non-swivel caster and the 3D-printed wheels can solve the problem of getting off the beam, their mobility is not good during the robot changes directions due to the friction they create. Therefore, the ball-universal wheel has the highest mobility and power efficiency compared to the others. In terms of speed, the ball-universal wheel and the caster wheel outperform 3D-printed wheels. In terms of price, the caster wheel is not a good choice to save costs.

After all the discussion and careful consideration from the Pugh matrix, the team had come to the decision to combine the ball universal wheel and the 3D-printed wheels as the solution for the robot's rear movement. The special feature of this design is that the 3D-printed wheel will be mounted at a certain height so as not to touch the ground when the robot moves on the plane. That means the 3D-printed wheels only help the robot slide down to the ground and then, the rolling motion will be smoothly transitioned from the 3D-printing wheel to the universal wheel without concern about friction.

iv. Upper and Lower Frames

The initial bottom frame is designed to fit basic electronic components such as motors, driver modules, batteries, and buck converters as shown in Figure 14.

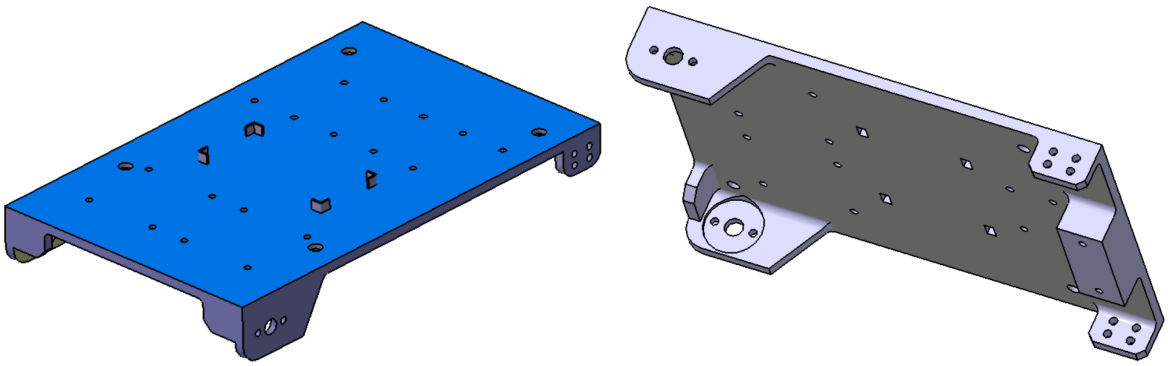


Figure 14. Bottom Frame Initial Version

During the design and testing process, the team decided to use servo motors and pushing arms to solve the problem of releasing from the beams. As a result, the bottom frame is extended the length to fit the servos in addition and enhanced in shape to enhance aesthetics as shown in Figure 15.

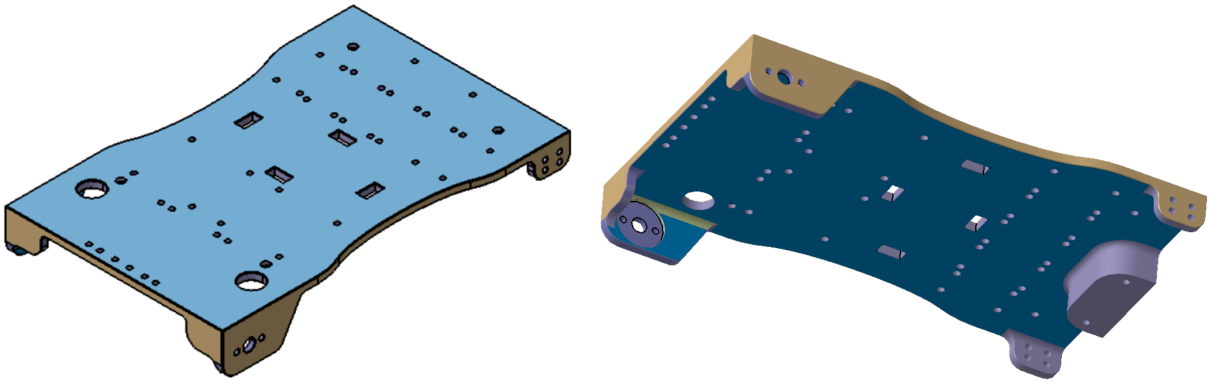


Figure 15. Bottom Frame Final Version

Originally, the top frame was designed to fit the PCB board and the Jetson Nano as shown in Figure 16. The little rectangular pad acts as the camera bracket which is used to mount the camera.

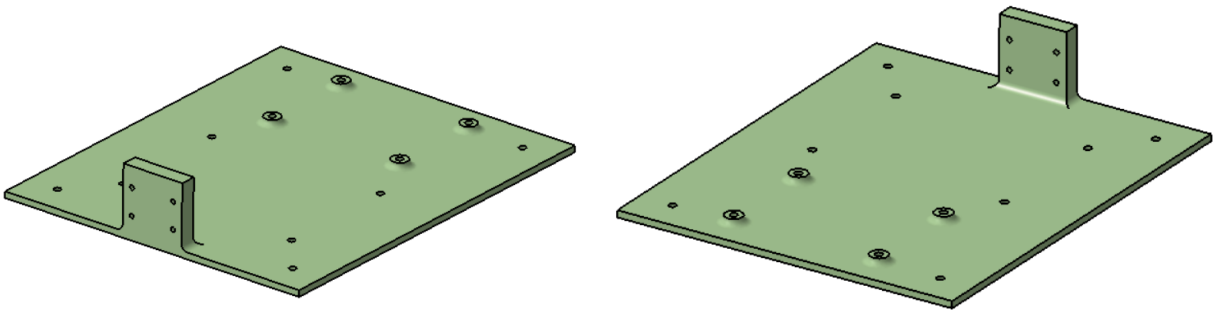


Figure 16. Top Frame Initial Version

The design of the top frame was then subjected to multiple revisions due to changes during testing. Since the camera range is limited, a micro servo motor is needed so that the camera can rotate and scan for the beam and the bell. Therefore, the camera bracket is removed from the top frame and replaced with a hole to mount the micro servo. The final result is presented in Figure 17.

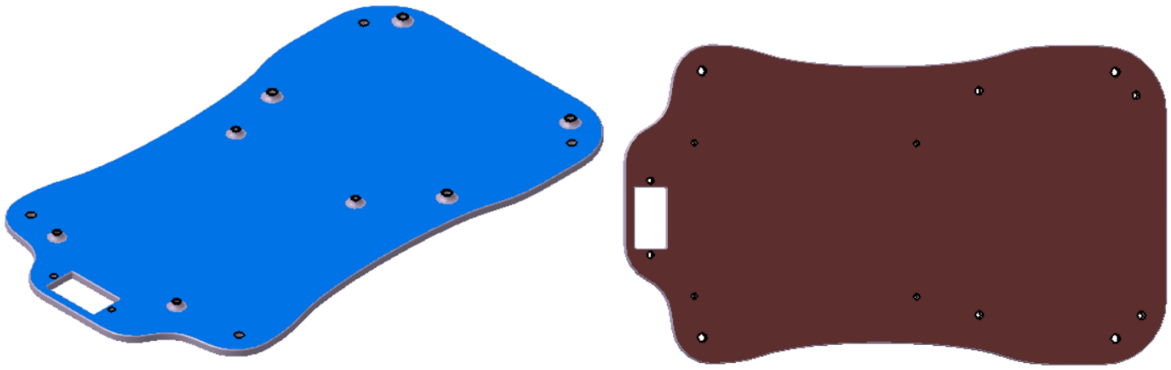


Figure 17. Top Frame Final Version

v. Pushing Stick and Alignment Arm

The initial design of the pushing stick is modeled after a lollipop with a spherical block on one end and a hole at the other end to attach the servo motor as shown in Figure 18. The spherical surface reduces friction in contact with the beam surface and requires less torque from the servos.

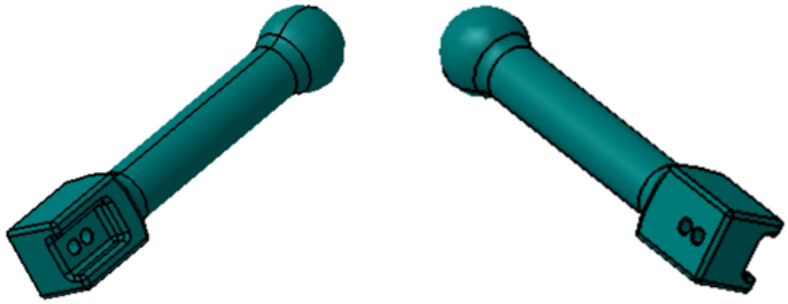


Figure 18. Pushing Stick Initial Version

The team then came up with the idea of using alignment arms to guide the robot into the center of the beam surface. The best solution is to attach the alignment arm to the pushing stick so that it can rotate with the desired movements of the robot. On the other hand, the solution to slow down the robot when it climbs down the beam is to take advantage of the friction between the pushing stick and the beam. Therefore, the pushing stick is modified to fit with the alignment arm and increase the contact area with the beam as shown in Figure 19. The design of the alignment arm is shown in Figure 20.

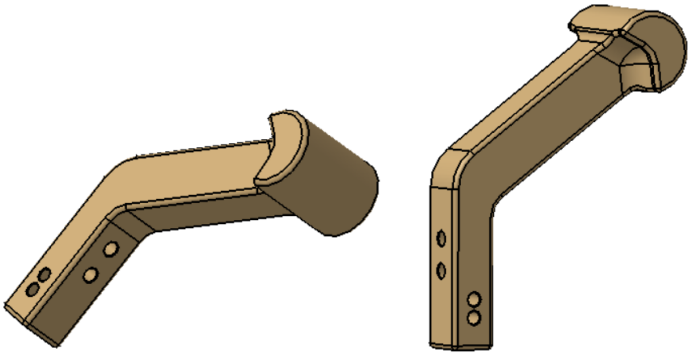


Figure 19. Pushing Stick Final Version

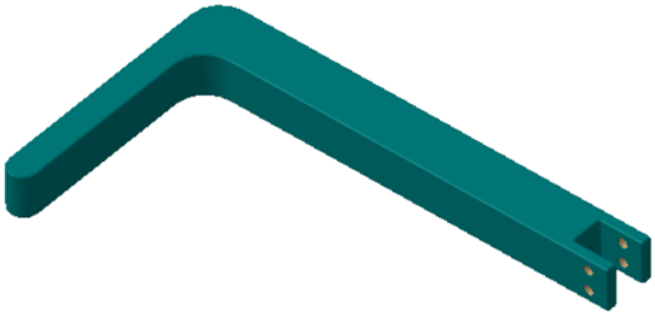


Figure 20. Alignment Arm.

b. CAD Assembly and Simulation

i. CAD Assembly

To ease the process of assembling all the parts, the model is divided into the sub-assemblies to easily organize and manage the constraints. The first sub-assembly is the components that are attached rigidly to the bottom frame such as buck converters, battery, driver modules, distance sensor and its brackets, servo motors and its brackets, sliding pads, and universal wheel as shown in Figure 21.

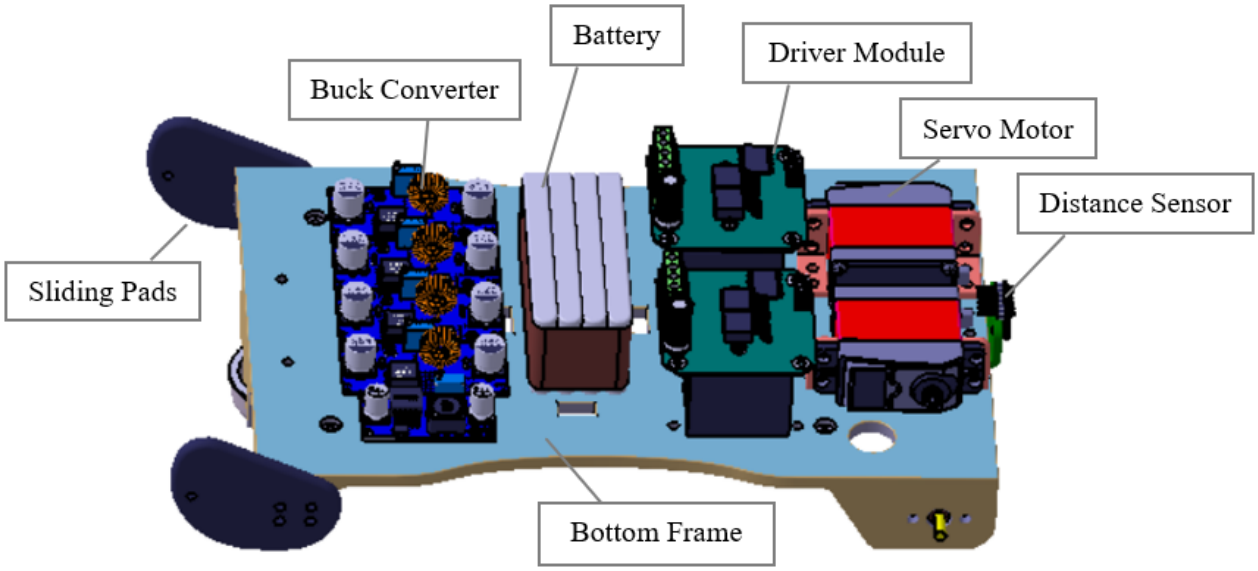


Figure 21. Bottom Frame Sub-assembly.

The sub-assembly of the front wheel includes the wheel with magnets, rubber band, and rigid coupling as shown in Figure 22.

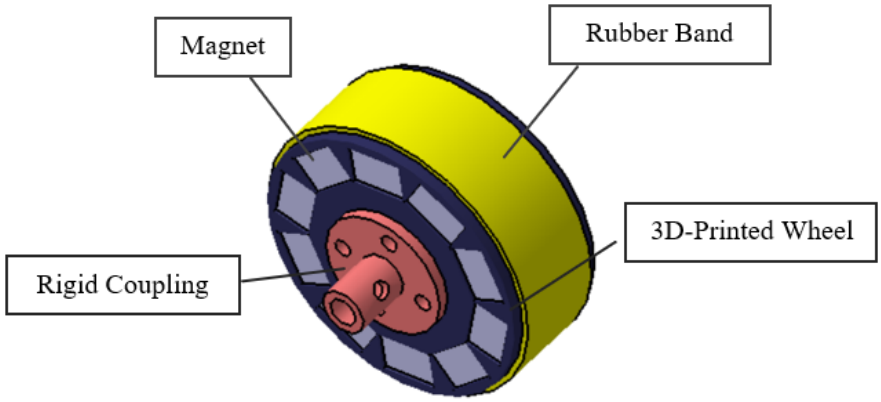


Figure 22. Front Wheel Sub-assembly.

During testing, we realized that the alignment arms weren't really necessary and could even interfere with the robot's climbing beam, so we decided to remove them from the final design. The sub-assembly of the pushing stick includes pushing stick and servo horn as shown in Figure 23.

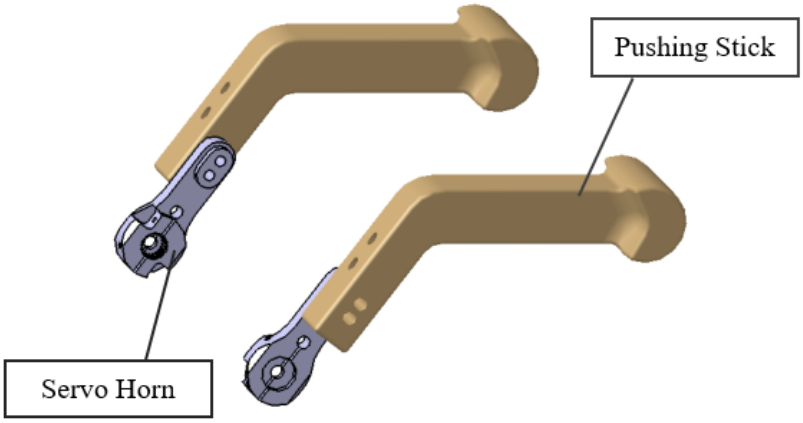


Figure 23. Pushing-Beam Sub-assembly.

The sub-assembly of the top frame includes the support pillars, jetson nano, shield board, microcontroller, camera, micro servo motor, and camera mounting bracket as shown in Figure 24. The other heads of the support pillars will be attached to the bottom frame.

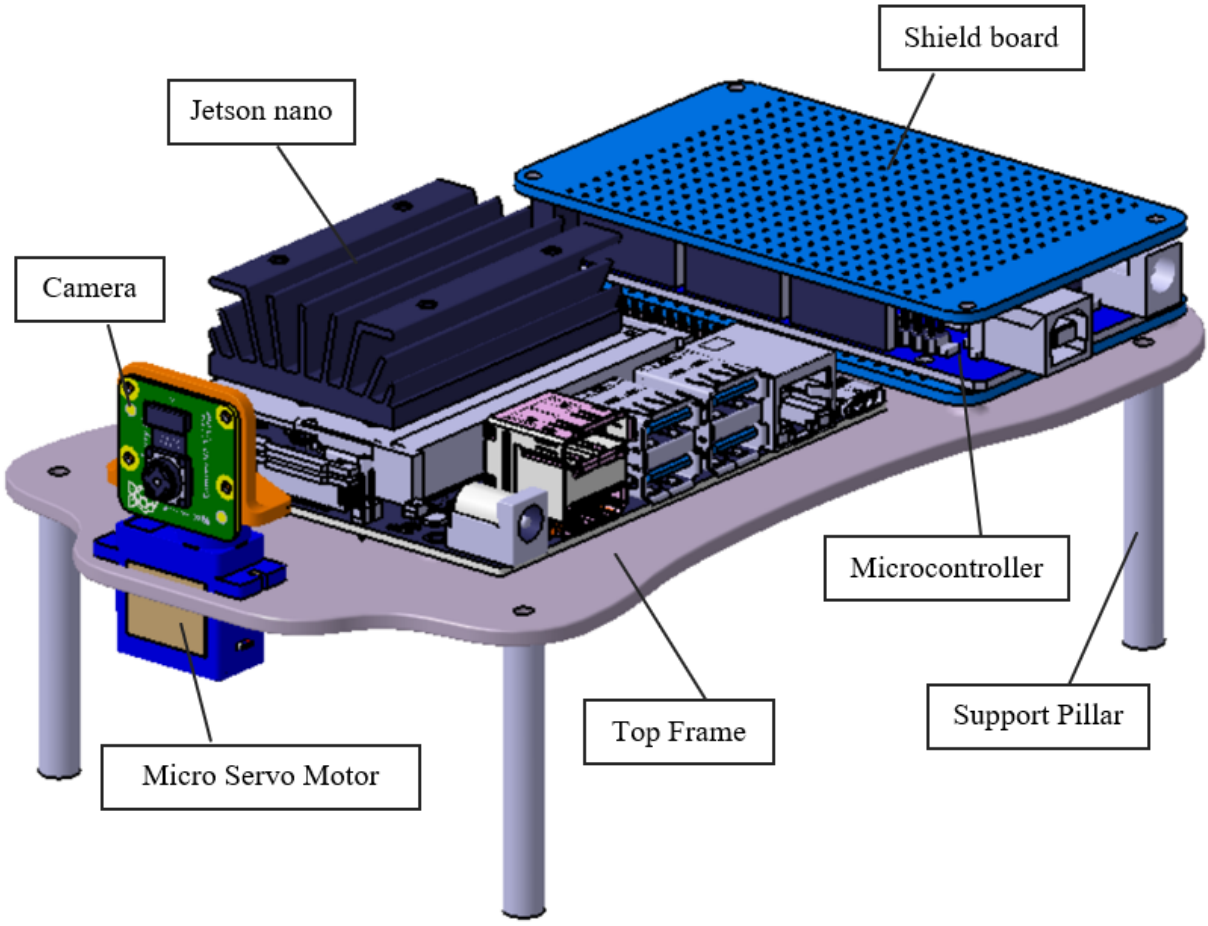


Figure 24. Top Frame Sub-assembly.

The full CAD assembly was created step by step with the sub-assemblies. The 3D-printed rear wheels which are the same size as the front wheels are attached to the sliding pads. Since we need more friction for the pushing surface, a rubber layer is attached on the head of the pushing sticks as shown in Figure 25.

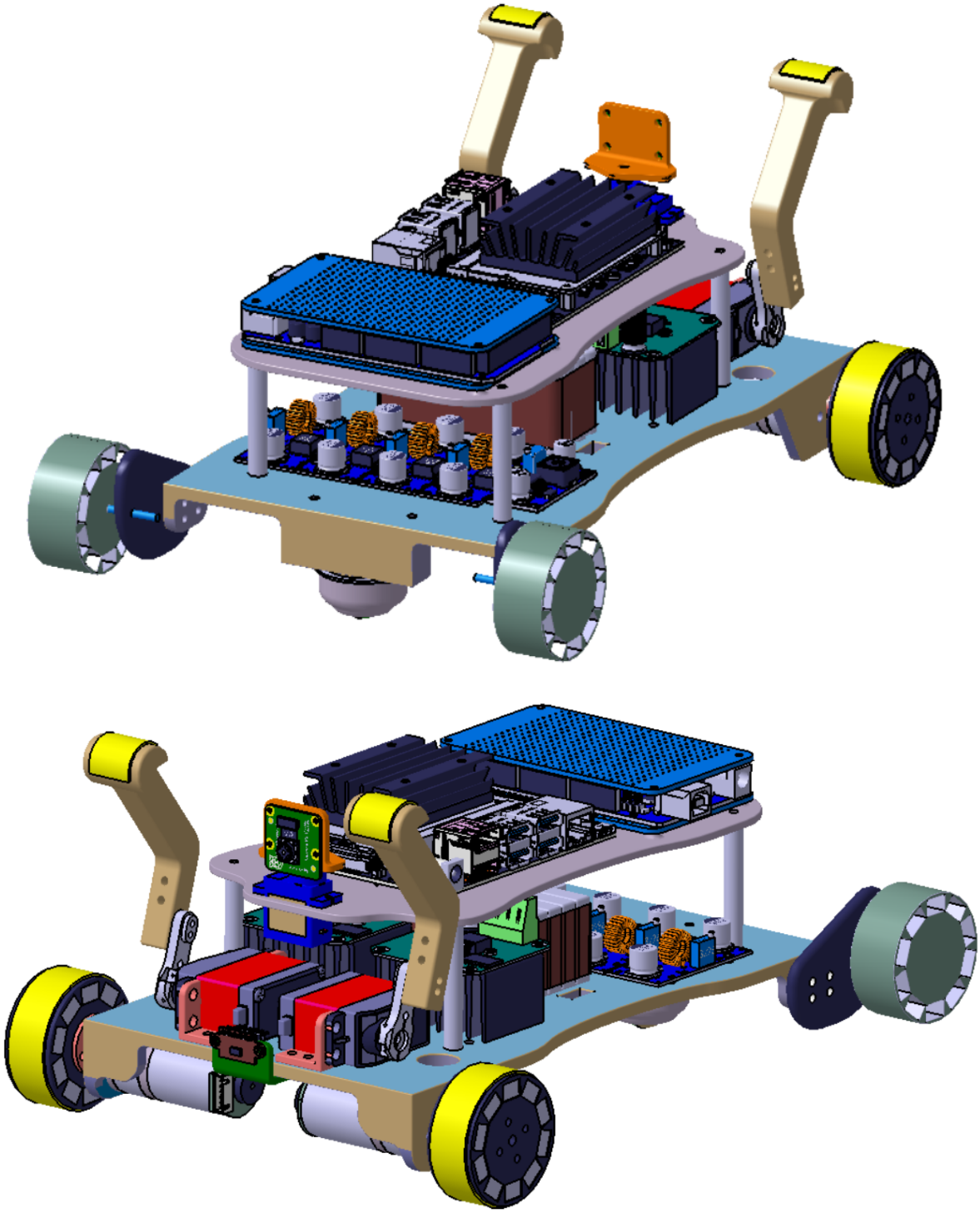


Figure 25. Full CAD Assembly.

ii. CAD Simulation

The model of the ground and the beam is created in order to make the simulation. Small grooves are added on the surface of the ground and the wall to make the constraints between the wheels and the sliding surface and guide the path for the robot. In reality, the robot's wheel will roll on the ground due to the torque of the motor and will continue to roll on the steel beam due to the magnetic force from the front wheels acting on the beam. However, the CAD model cannot improvise so flexibly. The wheel will only be able to roll on a surface that has a direct and continuous bond with the wheel surface. That means the wheel will not be able to continue rolling when it reaches the right angle between the floor and the beam because the right angle creates a disconnect between the wheel surface and the rolling surface of the environment. The only solution is to turn the right angle into a curved surface with a radius equal to or greater than the radius of the wheel as shown in Figure 26.

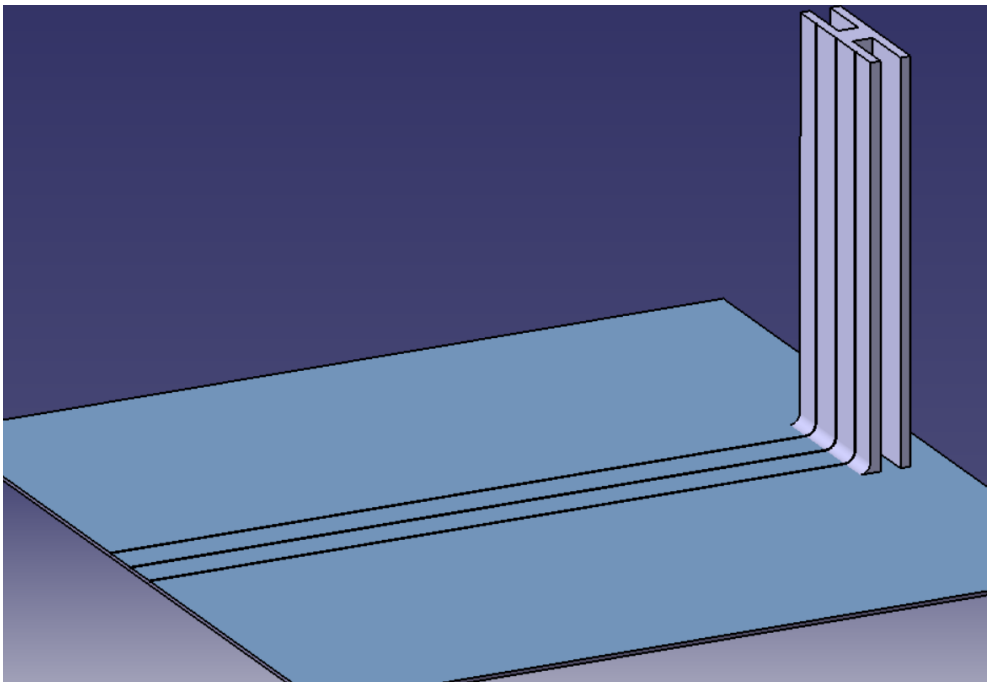


Figure 26. Curved Surface Connecting Wall and Floor.

Another important point is that in reality, when climbing down the beam, the plastic rear wheels will touch the ground first and continue to roll. At this point, the universal wheel will be disconnected from the beam surface until the rolling motion is smoothly transitioned from the plastic wheel to the universal wheel when the robot is completely on the ground. Then, the rear part of the robot will move based on the universal wheel as before. This sliding-down process can not be done in the CAD simulation. To make the simulation, we assumed that the motion of the rear part in the whole process is based on the universal wheel. Therefore, the sliding pads and rear wheels are hidden in the CAD simulation. Then, the CAD simulation is created with the following movements:

- Approaching the beam from the starting point.
- Climbing the beam (pushing sticks rotate to not touch the beam).
- Before climbing down the beam, pushing sticks rotate to push against the beam with a pushing force smaller than the magnetic force.
- Climbing down with a lower speed
- When reaching the ground, pushing sticks continues to push against the beam with a pushing force larger than the magnetic force to release from the beam.
- Returning to the starting point.

Those motions are shown in Figures 27-32.

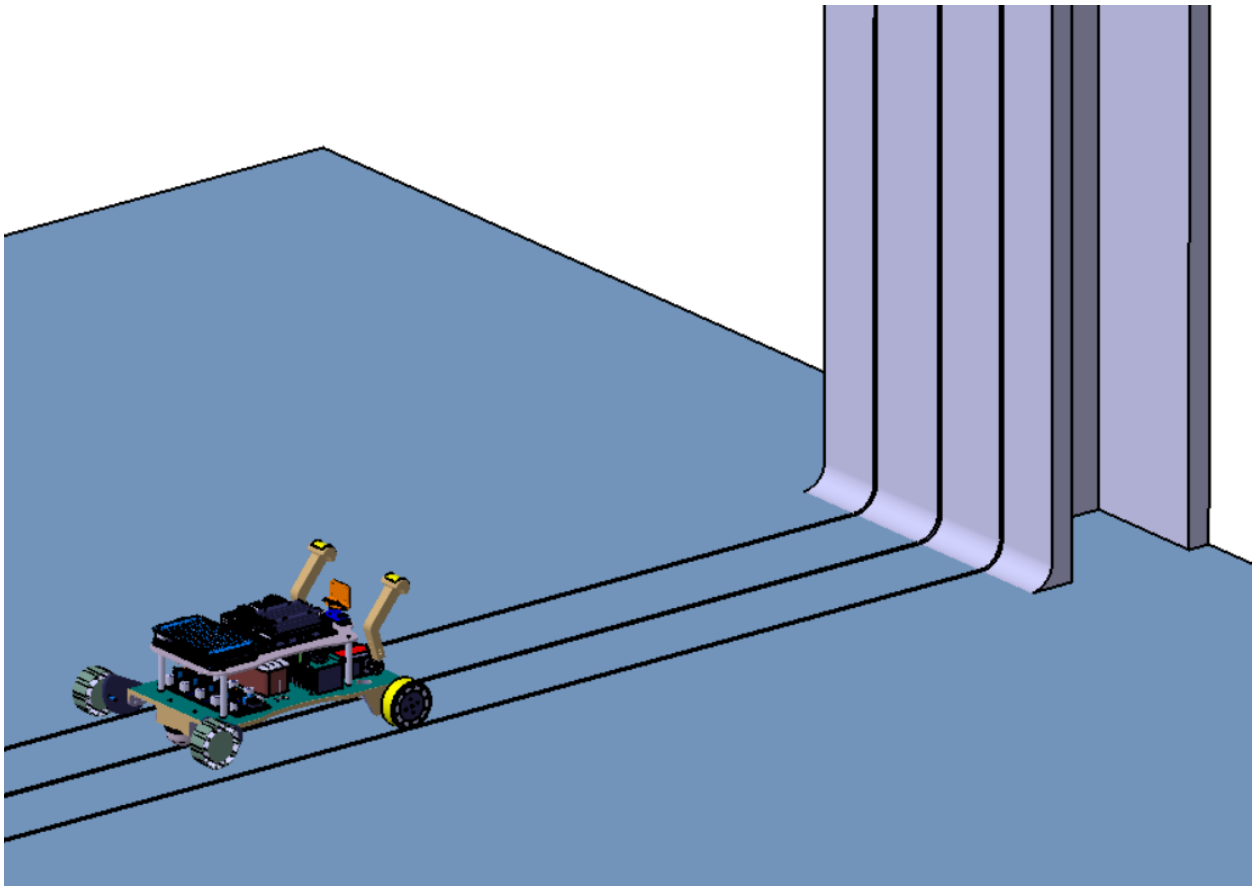


Figure 27. Approaching the Beam.

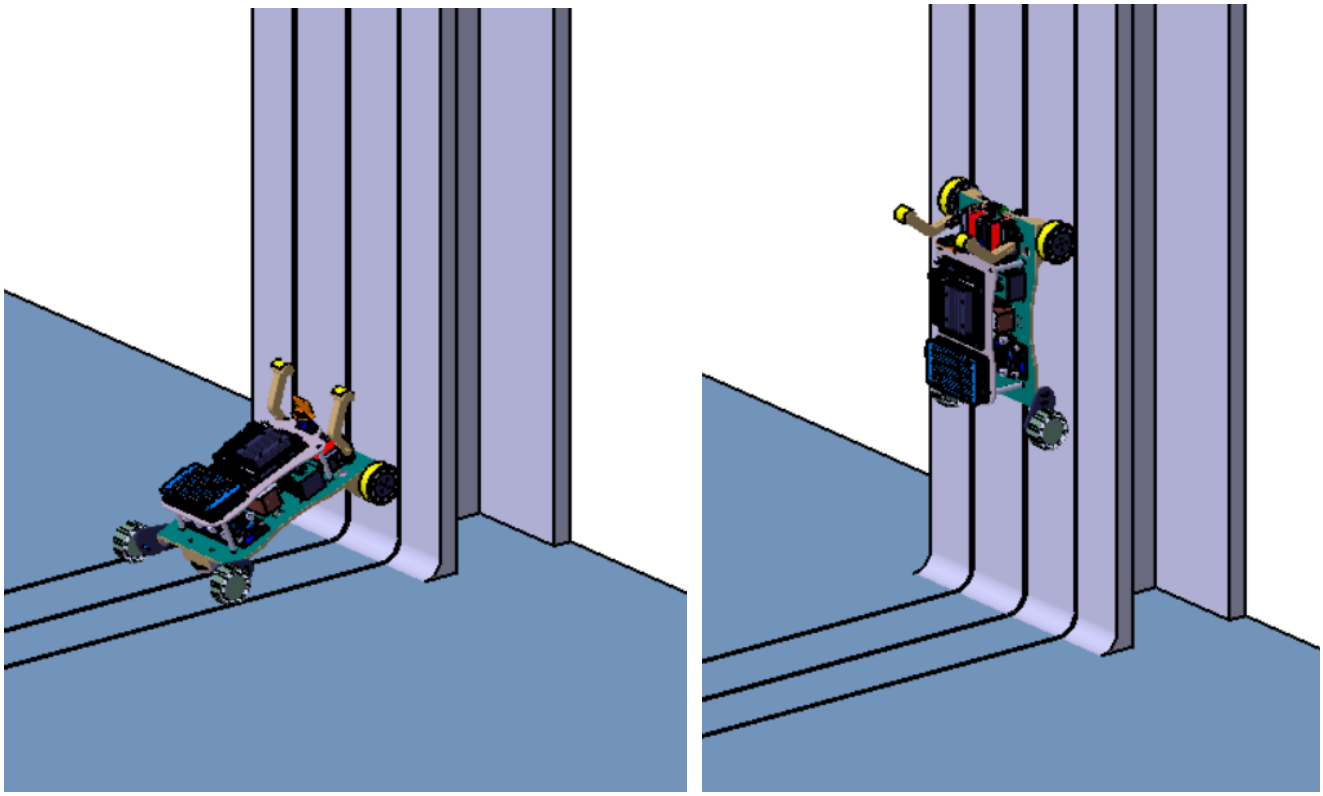


Figure 28. Climbing the Beam.

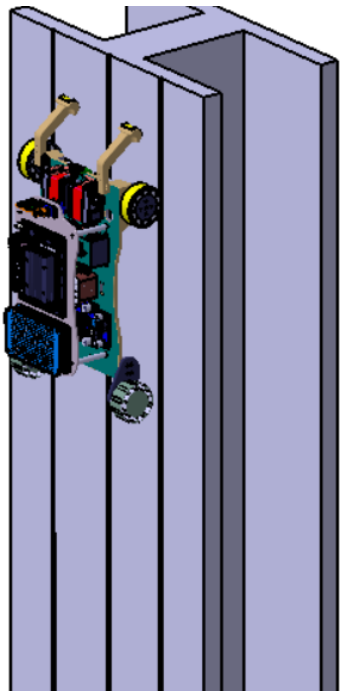


Figure 29. Pushing Stick Pushes Against the Beam and Climbing Down.

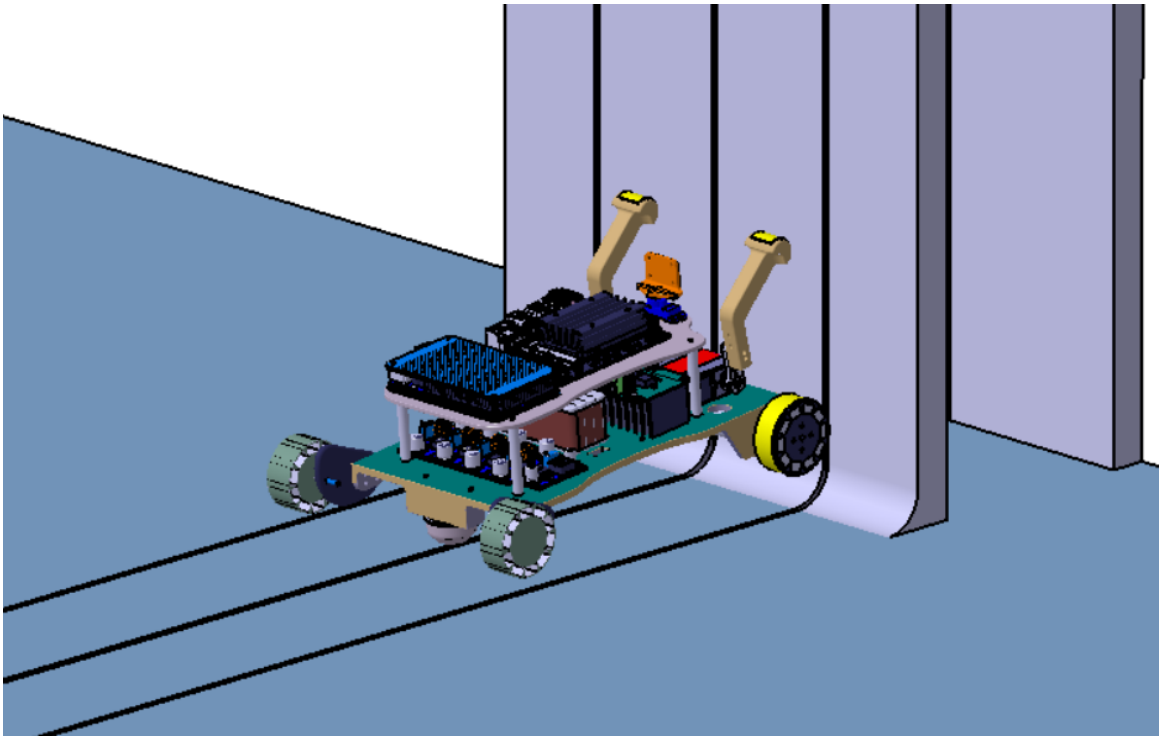


Figure 30. Sliding down to the Ground.

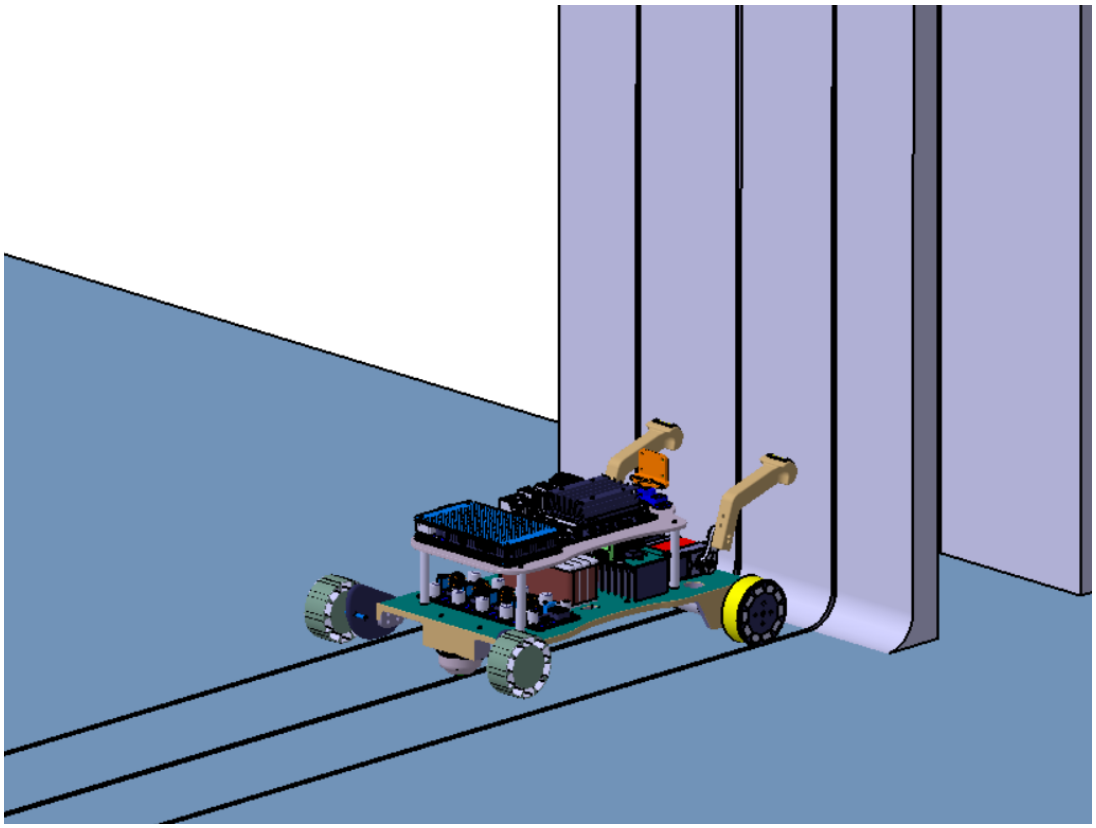


Figure 31. Increasing Pushing Force to Release from the Beam.

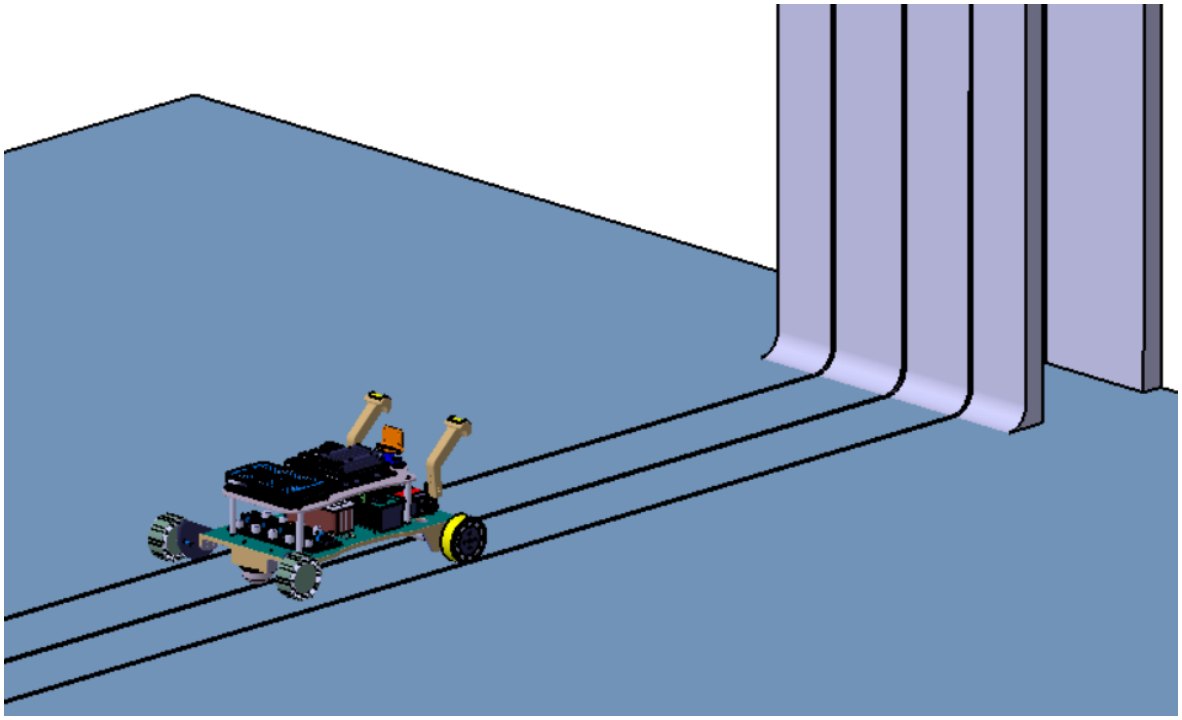


Figure 32. Returning to the Starting Point.

c. Kinematic and FEA analysis

i. Weight and Volume Calculation

The volume is obtained from the CAD model as shown in Figure 33. The total volume of the model is calculated to be 0.322ft^3.

$$Volume = 0.483ft \times 0.958ft \times 0.696ft = 0.322\ ft^3$$

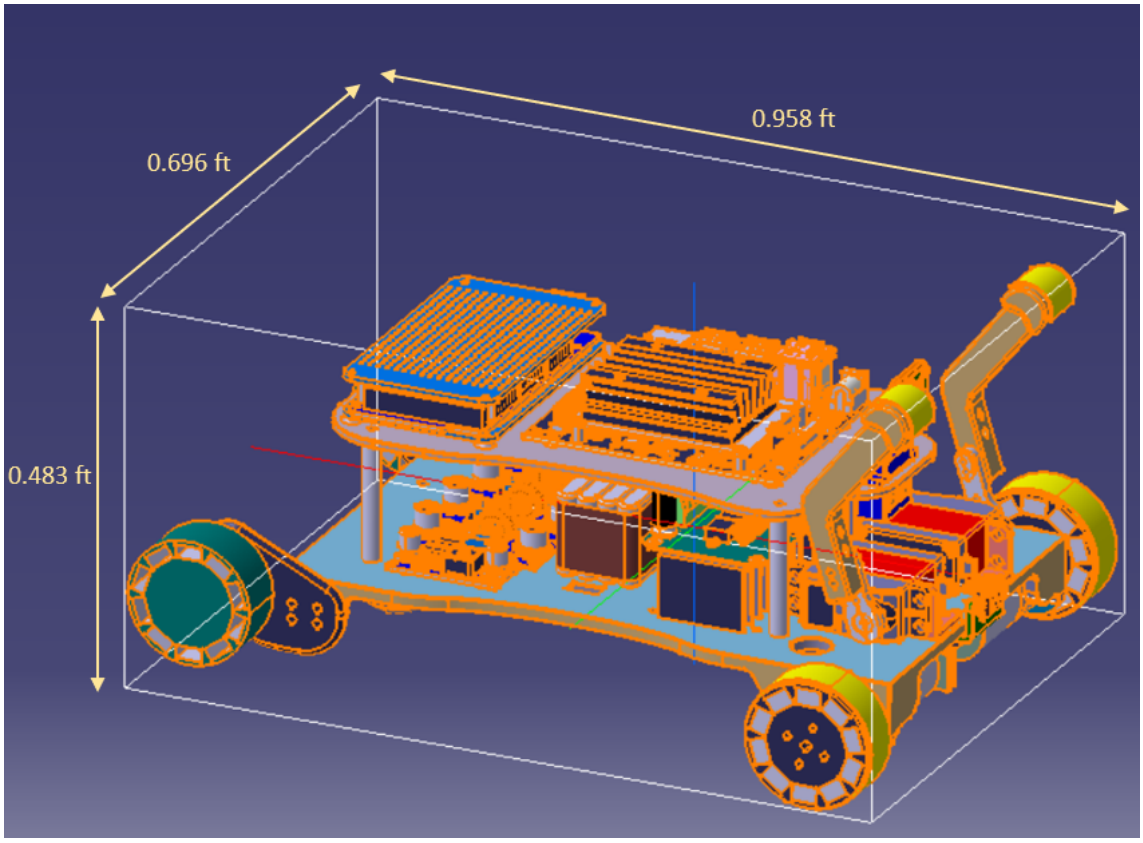


Figure 33. Volume of the CAD Assembly.

The weight for all components is measured and shown in Table 4 with the total weight is 1.560kg = 3.439lbm.

Table 4. Weight Calculation.

Amount	Components		Weight		
			Each [g]	Total	
				[kg]	[lbm]
1	Electrical	Battery	120	0.120	0.265
2		Motor	96	0.192	0.423
2		Driver	50	0.100	0.220
1		Jetson	250	0.250	0.551
5		Buck	11	0.055	0.121
1		Shield & Microcontroller	40	0.040	0.088
1		Camera	8	0.008	0.018
1		Distance Sensor	10	0.010	0.022
2		Servo Motor	47	0.094	0.207
1		Micro Servo Motor	16	0.016	0.035
2	Mechanical	10-hole wheel with magnet attached and rubber band	110	0.220	0.485
1		Ball Universal wheel	48	0.048	0.106

1		1st level frame	120	0.120	0.265
1		2nd level frame	87	0.087	0.192
4		Column	10	0.040	0.088
2		Sliding Pad	10	0.020	0.044
2		Rear Wheel	15	0.030	0.066
2		Pushing Stick & Rubber Cover	10	0.020	0.044
Many		Bracket (for Sensors and Servo Motors)	10	0.010	0.022
Many		Other (screws, wires, cable ties)	80	0.080	0.176
Total				**1.560**	**3.439**

ii. Rolling Resistance

The material for the fabrication of wheels is ABS plastic which is a combination of the strength and stiffness of acrylonitrile and styrene with the toughness of polybutadiene rubber to become a thermoplastic amorphous polymer that is transparent in its natural state [3]. The rolling resistance of the magnetic wheels when the robot climbs on the steel beam is influenced by many factors including walk resistance, magnetic force, friction, and air resistance. Some of these factors can be obtained from research such as friction, but some cannot be accurately defined.

Walk Resistance (energy distribution due to flattening).

Even though the front wheels are covered by rubber band, the surface contact with the ground, as well as the steel column, is very small because the rubber layer is very thin and stiff due to tension. That leads to energy dispassion due to the contact surface between the wheel and the ground or the column is pretty small. Therefore, we can estimate that the walk resistance since it is very small.

Magnetic Force

The pull force of the magnet to the steel beam creates rolling resistance for the wheels. The force between the magnets and the steel column will be strongest when the magnets are in contact with the steel, and it decreases rapidly as they are separated.

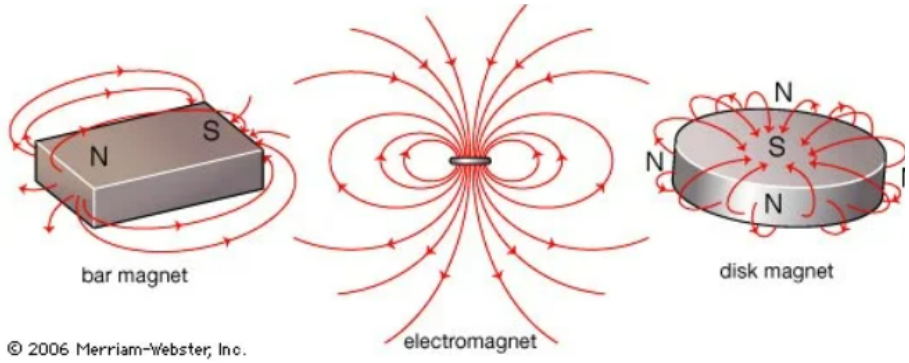


Figure 34. Magnetic Field [4].

Figure 4 shows the magnetic field for the different types of magnets. Using the spring scale, the total magnetic force of 2 front wheels is measured to be 5.66lbf.

Friction Resistance

The friction in this criterion comes from the roughness of the two contact surfaces. The friction force is the multiplication of the normal force F_N and the friction coefficient μ_f.

$$F_f = F_N \mu_f$$

When the robot moves on the floor, normal force is the force that the floor exerts on the front and rear wheels to balance the weight of the robot. When the robot climbs on the beam, normal force will balance with the magnetic force. In reality, friction is very important for the rolling movement of any circular object because it acts as the lever to amplify the torque and resist the slip condition. The more flexible the material, the higher the coefficient of friction. The fact that the front wheels are covered with rubber bands helps the wheels roll easier and faster.

For the front wheel, the friction is between the covering rubber band and the concrete floor surface as well as the steel beam surface. For the rear part, the friction comes from the contact between the steel ball universal wheel and the concrete floor surface as well as the steel beam surface.

Air Resistance

We all know that air is able to slow down the motion of an object. By definition, the collision of an object's leading surface with air molecules results in air resistance [5]. It describes the forces opposing the relative motion of an object as it passes through the air. These drag forces act against the incoming flow velocity, thereby slowing the object. The speed of the object and the cross-section of the object are the two most common factors affecting the air resistance which can be calculated as follows.

$$F_D = \frac{1}{2} \rho v^2 C_D A$$

where F_D = drag force, ρ = density of the air, v = speed of the robot, C_D = drag coefficient, and A = cross sectional area.

In this case, the air resistance can be ignored since the speed of the robot is not very fast and the air density inside the SD lab is stable, especially, with no wind.

iii. Torque Analysis for DC Motors

To do the kinematic analysis, we need to find the position of the center of gravity where the weight acts on. Figure 35 displays the distance from the center of gravity obtained for the CAD model.

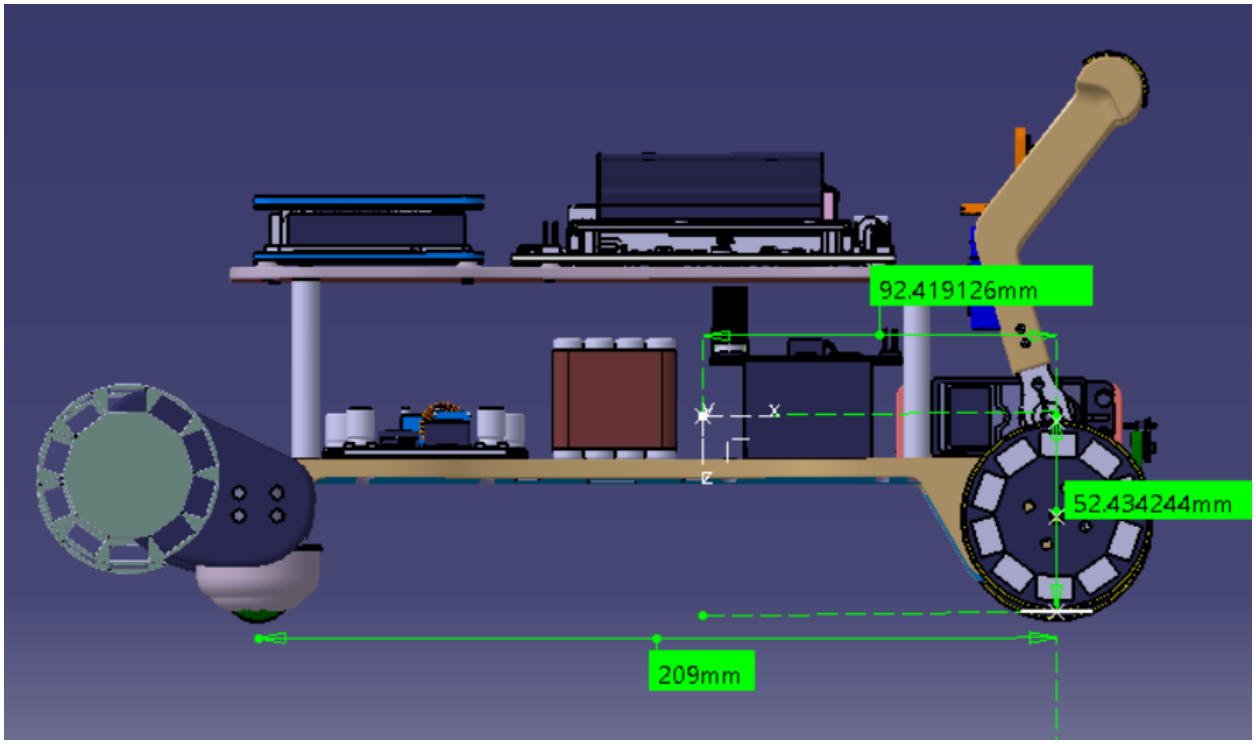


Figure 35. Position of Center of Gravity.

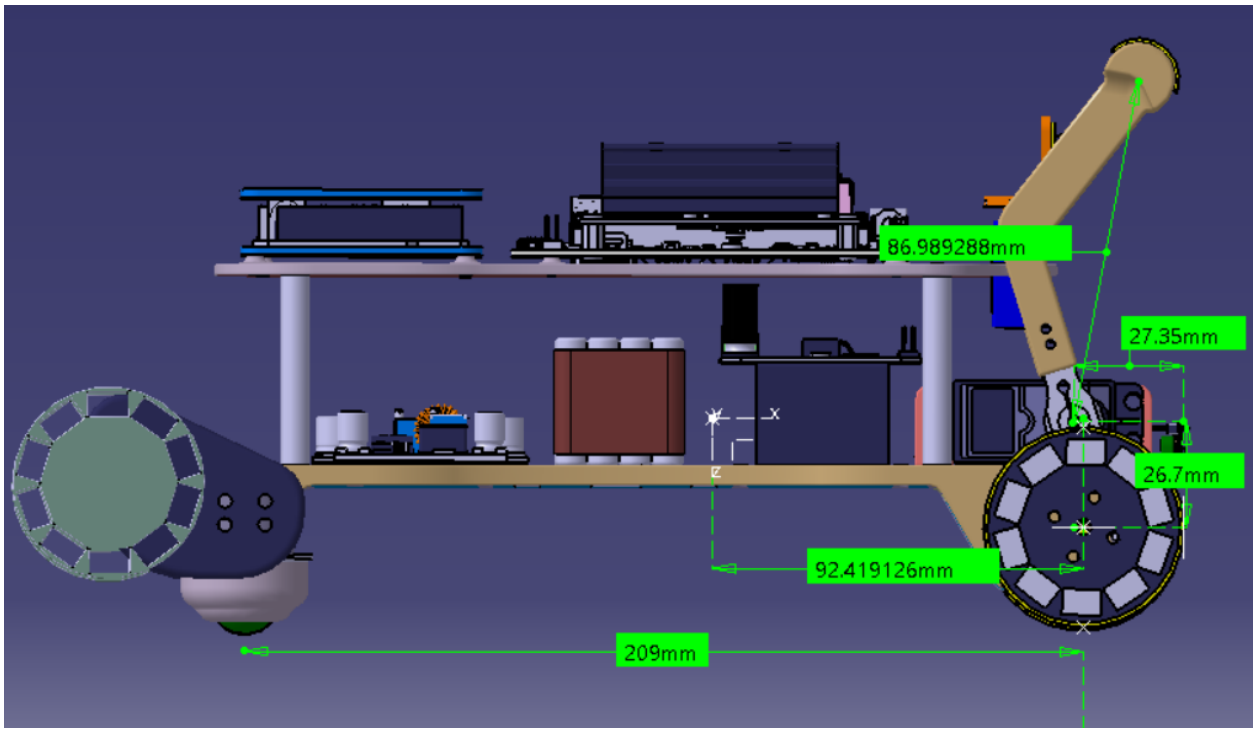


Figure 36. CAD Measurement.

To determine the minimum value of the total torque needed for the motors, we need to analysis the system when the robot is climbing the beam as shown in Figure 36.

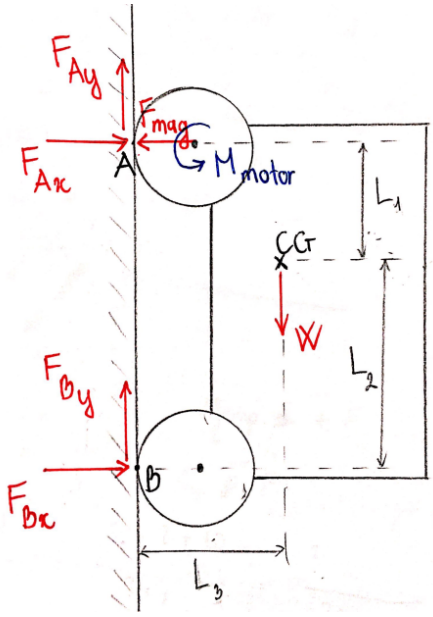


Figure 37. Free Body Diagram of the Robot Climbing the Beam.

In Figure 36,

- $F_{Ax} = total\ normal\ force\ exerted\ on\ the\ front\ wheels$
- $F_{Ay} = F_{Ax}\mu_1 = total\ friction\ force\ between\ the\ front\ wheels\ and\ the\ steel\ beam$
- $F_{Bx} = total\ normal\ force\ exerted\ on\ the\ rear\ wheels$
- $F_{By} = F_{Bx}\mu_2 = total\ friction\ force\ between\ the\ rear\ wheels\ and\ the\ steel\ beam$
- $F_{mag} = total\ magnetic\ force\ exerting\ on\ the\ beam$
- $W = mg = total\ weight\ of\ the\ robot$
- $M_{motor} = total\ DC\ motor\ torque$
- $\mu_1 = 0.64 = friction\ coeficient\ of\ rubber\ on\ steel$ [6]
- $\mu_2 = 0.8 = friction\ coeficient\ of\ steel\ on\ steel$ [6]

Taking the force analysis from figure 36, we get

$$\sum F_x = 0$$

$$\rightarrow F_{Ax} + F_{Bx} = F_{mag}$$

$$\rightarrow F_{Ax} = F_{mag} - F_{Bx}$$

$$\sum F_y = 0$$

$$\rightarrow W = F_{Ay} + F_{By} = F_{Ax}\mu_1 + F_{Bx}\mu_2 = \left(F_{mag} - F_{Bx}\right)\mu_1 + F_{Bx}\mu_2$$

$$\rightarrow F_{Bx} = \frac{W - \mu_1 F_{mag}}{\mu_2 - \mu_1}$$

$$\sum M_A = 0$$

$$\rightarrow M_{motor} = WL_3 - F_{Bx}\left(L_1 + L_2\right) = WL_3 - \left(\frac{W - \mu_1 F_{mag}}{\mu_2 - \mu_1}\right)\left(L_1 + L_2\right)$$

The lengths $L_1 + L_2$, L_3 can be found from Figure 35. Using the weight found from Table 4, the calculated value for total minimum torque required for 2 DC motors is displayed in Table 5.

Table 5. Minimum Motor Torque Calculation [6].

Measured Data		**Calculated Data**
$L_1 + L_2$ = 209mm	Mass m = 1.560 kg	Weight $W = mg$ = 15.304N
L_3 = 52.43mm	μ_1 = 0.64	$F_{Bx} = -5.06\ N$
F_{mag} = 5.66lbf = 25.177N	μ_2 = 0.8	$M_{motor} = 1.86N.m = 18.968kgf.cm$

iv. Torque Analysis for Servo Motors

To determine the servo torque, Figure 37 shows the FBD of the system with all the lengths and exserting forces on the model.

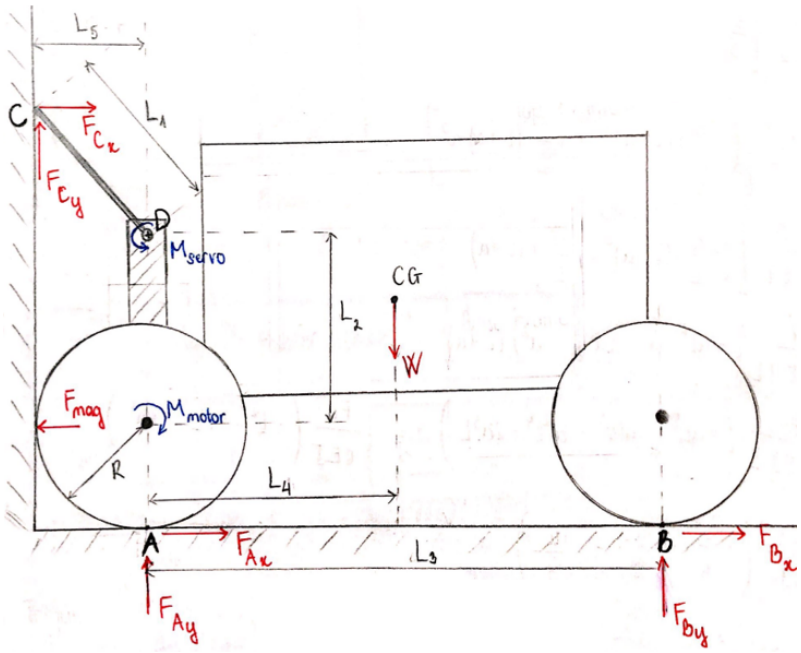


Figure 38. Free Body Diagram When the Robot on the Ground.

In Figure 23,

- $F_{Ay} = total\ normal\ force\ exerted\ on\ the\ front\ wheels$
- $F_{Ax} = F_{Ay}\mu_1 = total\ friction\ force\ between\ the\ front\ wheels\ and\ the\ floor$
- $F_{By} = total\ normal\ force\ exerted\ on\ the\ rear\ wheels$
- $F_{Bx} = F_{By}\mu_2 = total\ friction\ force\ between\ the\ rear\ wheels\ and\ the\ floor$
- $F_{Cx} = total\ normal\ force\ exerted\ on\ the\ pushing\ sticks$
- $F_{Cy} = F_{Cx}\mu_3 = total\ friction\ force\ between\ the\ pushing\ stick\ and\ the\ beam$
- $F_{mag} = total\ magnetic\ force\ exerting\ on\ the\ beam$
- $W = mg = total\ weight\ of\ the\ robot$
- $M_{motor} = total\ DC\ motor\ torque$
- $M_{servo} = total\ servo\ motor\ torque$
- $\mu_1 = friction\ coeficient\ of\ rubber\ on\ concrete$ [6]
- $\mu_2 = friction\ coeficient\ of\ steel\ on\ concrete$ [7]
- $\mu_3 = friction\ coeficient\ of\ rubber\ on\ steel$ [6]

Then, the minimum torque needed for the servo is

$$\sum M_D = 0$$

$$\rightarrow M_{servo} = M_{motor} + F_{Cx}\sqrt{L_1^2 - L_5^2} + F_{Cy}L_5 + F_{mag}L_2 + F_{Ay}\left(L_5 - R\right) - F_{Ax}\left(L_2 + R\right) - F_{By}L_3 - F_{Bx}\left(L_2 +$$

$$\rightarrow M_{servo} = M_{motor} + F_{Cx}\sqrt{L_1^2 - L_5^2} + F_{Cx}\mu_3 L_5 + F_{mag}L_2 + F_{Ay}\left(L_5 - R\right) - F_{Ay}\mu_1\left(L_2 + R\right) - F_{By}L_3 - F_{By}\mu$$

$$\rightarrow M_{servo} = M_{motor} + F_{Cx}\left(\sqrt{L_1^2 - L_5^2} + \mu_3 L_5\right) + F_{mag}L_2 + F_{Ay}\left[\left(L_5 - R\right) - \mu_1\left(L_2 + R\right)\right] - F_{By}\left[L_3 + \mu_2\left(L_2 +\right.\right.$$

Now, we need to determine the values of F_{Ay}, F_{By}, and F_{Cx} by analyzing the horizontal force as follows.

$$\sum F_x = 0$$

$$\rightarrow F_{Cx} + F_{Ax} + F_{Bx} - F_{mag} = 0$$

$$\rightarrow F_{Cx} = F_{mag} - F_{Ax} - F_{Bx} = F_{mag} - F_{Ay}\mu_1 - F_{By}\mu_2$$

To get the minimum value needed for the servo-motor torque, consider a dynamic system when the robot is moving on the horizontal plane as shown in Figure 38.

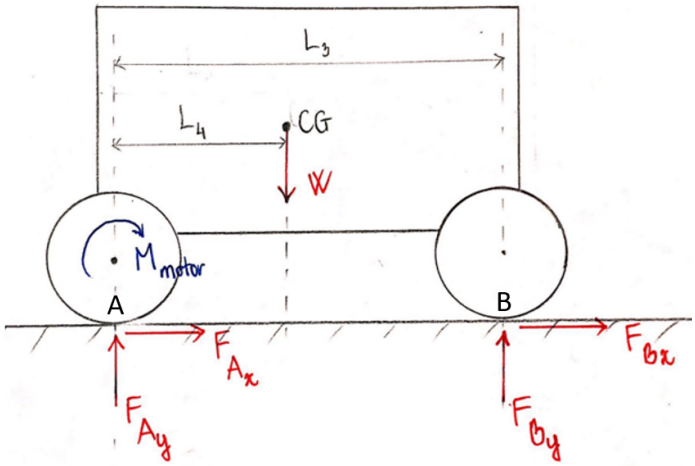


Figure 39. Free Body Diagram When the Robot Moving on the Horizontal Plane.

From that, we can get the minimum values of F_{Ay} and F_{By} as follows.

$$\sum M_A = 0$$

$$\rightarrow -M_{motor} - WL_4 + F_{By}L_3 = 0$$

$$\rightarrow F_{By} = \frac{M_{motor}+WL_4}{L_3}$$

$$\sum F_y = 0$$

$$\rightarrow F_{Ay} + F_{By} - W = 0$$

$$\rightarrow F_{Ay} = W - F_{By} = W - \frac{M_{motor}+WL_4}{L_3}$$

Table 6. Kinetic Calculation for the Pushing Arm [6], [7].

Measured Data		Calculated Data
L_1 = 86.989mm	F_{mag} = 5.66lbf = 25.177N	$W = mg = 15.304N$
L_2 = 26.7mm	μ_1 = 0.6	$F_{Ay} = -0.845N$
L_3 = 209mm	μ_2 = 0.42	$F_{By} = 16.149N$
L_4 = 92.42mm	μ_3 = 0.64	$F_{Cx} = 18.942N$
L_5 = 27.35mm	$M_{motor} = 20\ kgf.cm = 1.96N$	$M_{servo} = 2.195N = 22.39kgf.cm$
R = 25mm	Mass $m = 1.560kg$	

v. FEA Analysis

Since the materials for making the wheels and the frames are mainly ABS plastic which has a yield strength of about 40-60 MPa [8], the 5-lb weight of the robot will not be a problem. Therefore, FEA analysis will focus on the design of the push arm to see how it bears the magnetic force to push the robot off the beam.

The force applied on the pushing arm is shown in Figure 39.

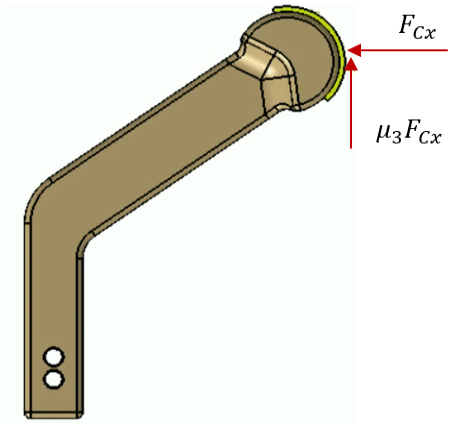
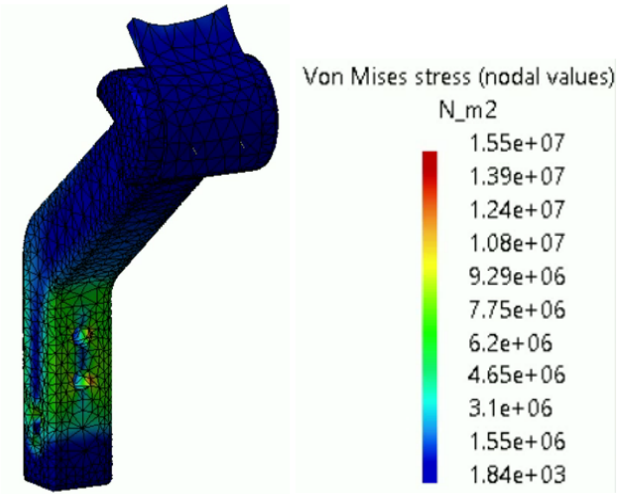


Figure 40. Force Inserting on the Pushing Arm.

Applying the value of F_{Cx} obtained from Table 6, the stress analysis was done using Catia V5 as shown in Figure 40 with the Von Mises Stress and the Displacement.



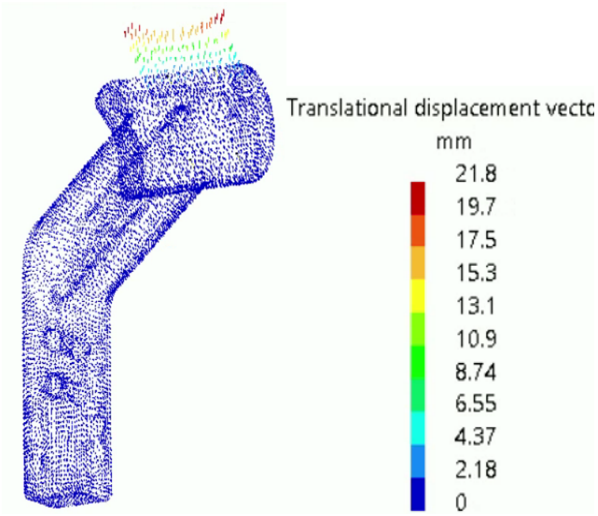


Figure 41. Von Mises Stress and Displacement Analysis.

d. Bill of Mechanical Components

Table 6. Bill of Mechanical Components.

Quantity	Part	Model	Price	Amount used	Cost of item used	Notes
20	Magnets	BC64	$51.75	20	$40.32	Purchased Item
2	Rigid Flange Coupling	Motor Shaft Flange	$8.47	2	$7.99	Purchased Item
300	Bolts		$4.24	60	$0.80	Purchased Item
100	Nylon Nuts		$6.35	20	$1.20	Purchased Item
300	Zipties		$11.00	15	$0.55	Purchased Item
1	Steel Ball Universal Wheel		$2.00	1	$2.00	Purchased Item
2	Wheel	5 magnet-hole		7.03	$0.14	TPU (3D-Printing)
1		9 magnet-hole		13.35	$0.33	PLA (3D-Printing)
1		9 magnet-hole		13.35	$0.27	TPU (3D-Printing)
2		9 magnet-hole v2		11.95	$0.24	TPU (3D-Printing)
2		10 magnet-hole		14.71	$0.29	TPU (3D-Printing)
2		10 magnet-hole v2		14.67	$0.29	TPU (3D-Printing)
1	Frame	Top Frame v1		119.82	$2.99	PLA (3D-Printing)
1		Top Frame v2		114.73	$2.29	PLA (3D-Printing)
1		Top Frame v3		115.32	$2.30	PLA (3D-Printing)
1		Bottom Frame v1		112.76	$2.24	PLA (3D-Printing)
1		Bottom Frame v2		113.23	$2.25	PLA (3D-Printing)
Total			**$97.44**	**Total on Robot**	**$66**	

2. Detailed Engineering Drawings of Each Component

a. Mechanical Parts Responding for Frame

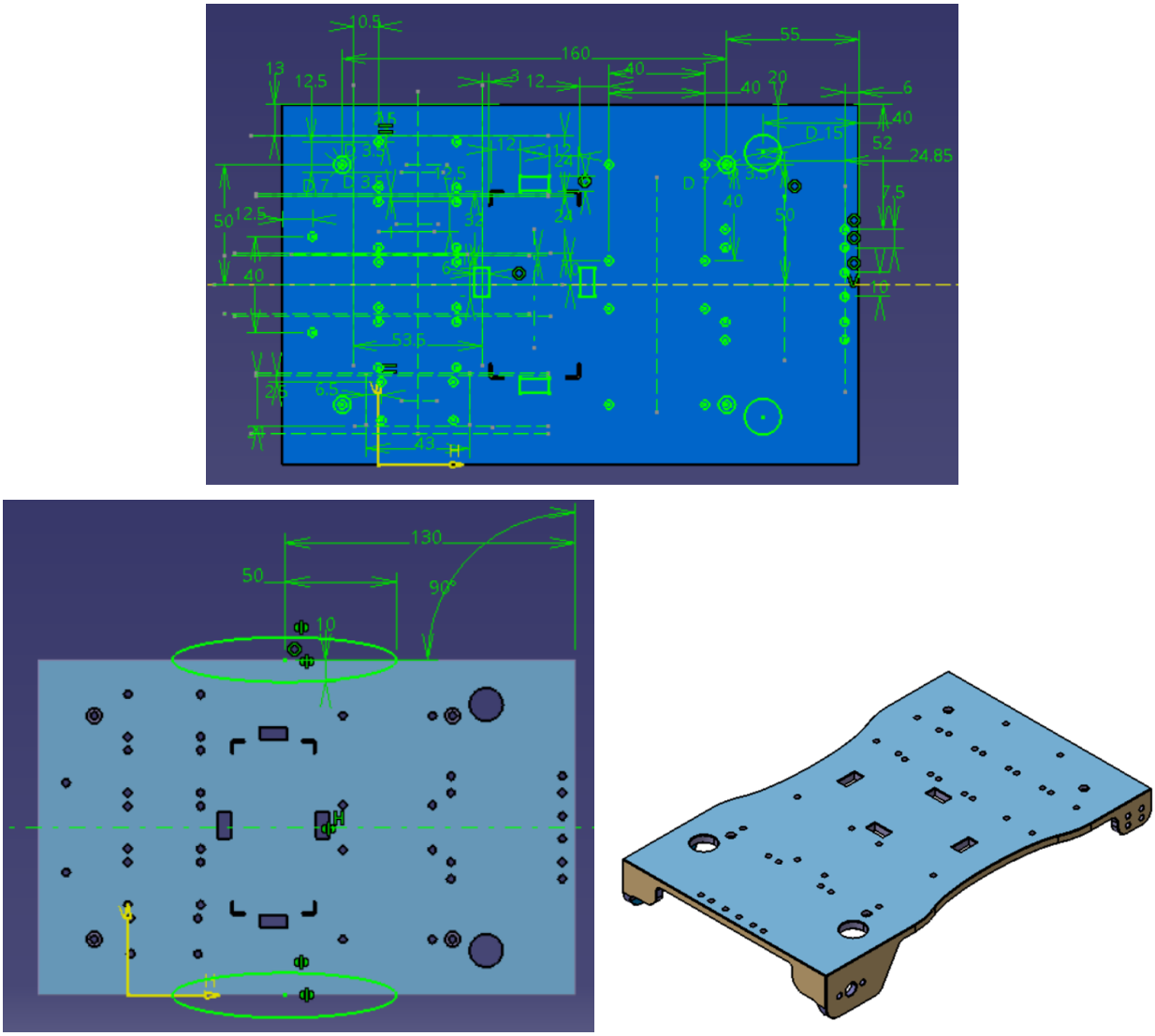


Figure 42. Engineering Drawing for Bottom Frame (3D-Printing).

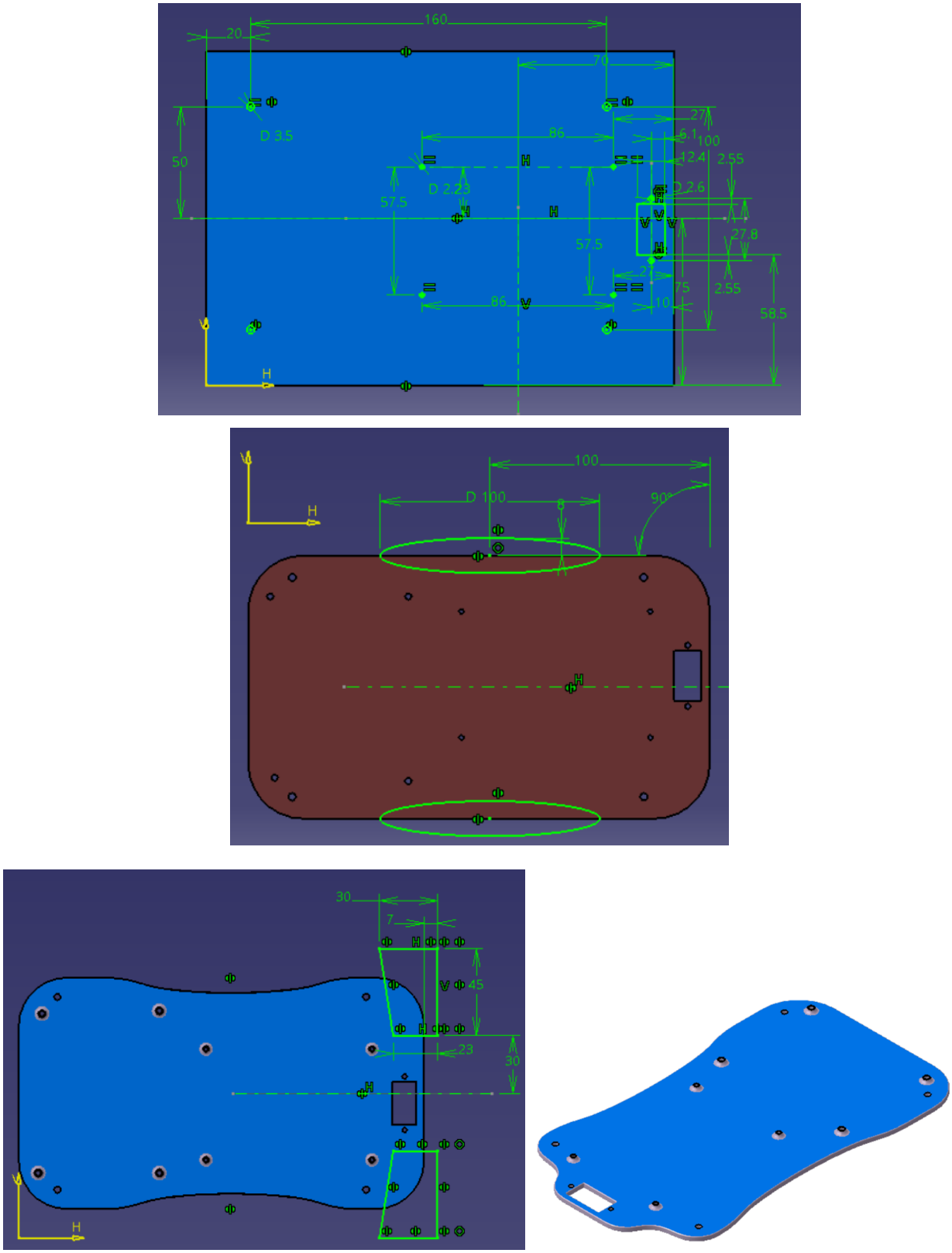


Figure 43. Engineering Drawing for Top Frame (3D-Printing).

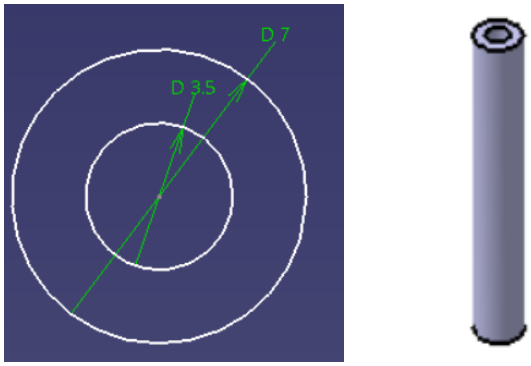
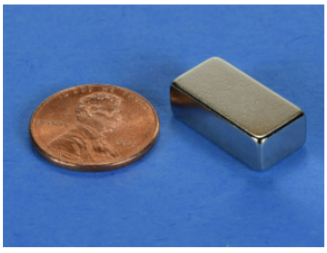
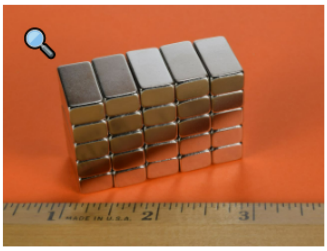


Figure 44. Engineering Drawing for Support Pillar (3D-Printing).

b. Mechanical Parts Responding for Robot's Movement



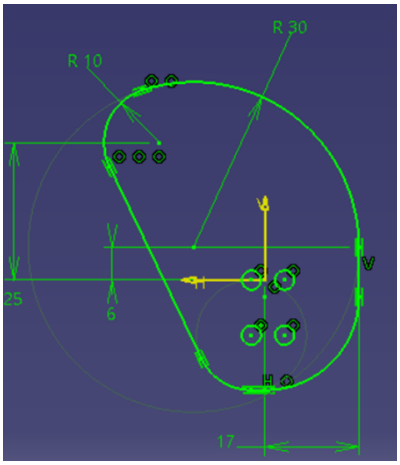


Description | Technical | Downloads

Dimensions: 3/4" x 3/8" x 1/4" thick
Tolerances: ±0.004" x ±0.004" x ±0.004"
Material: NdFeB, Grade N42
Plating/Coating: Ni-Cu-Ni (Nickel)
Magnetization Direction: Thru Thickness
Weight: 0.305 oz. (8.64 g)
Pull Force, Case 1: 14.36 lbs
Pull Force, Case 2: 24.23 lbs
Surface Field: 4404 Gauss
Max Operating Temp: 176°F (80°C)
Brmax: 13,200 Gauss
BHmax: 42 MGOe

The BC64 is a powerful block in the very popular 1/4" thickness. This block is very strong and quite durable.

Figure 45. Engineering Specifications for BC64 Magnets [9] (purchased).



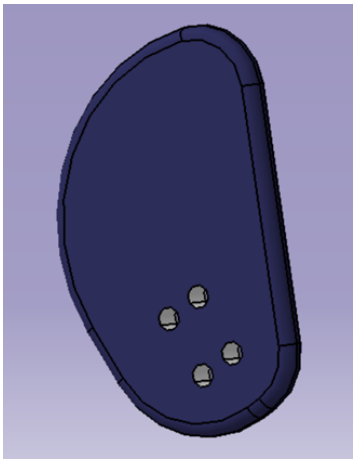


Figure 46. Engineering Drawing for Sliding Pad.

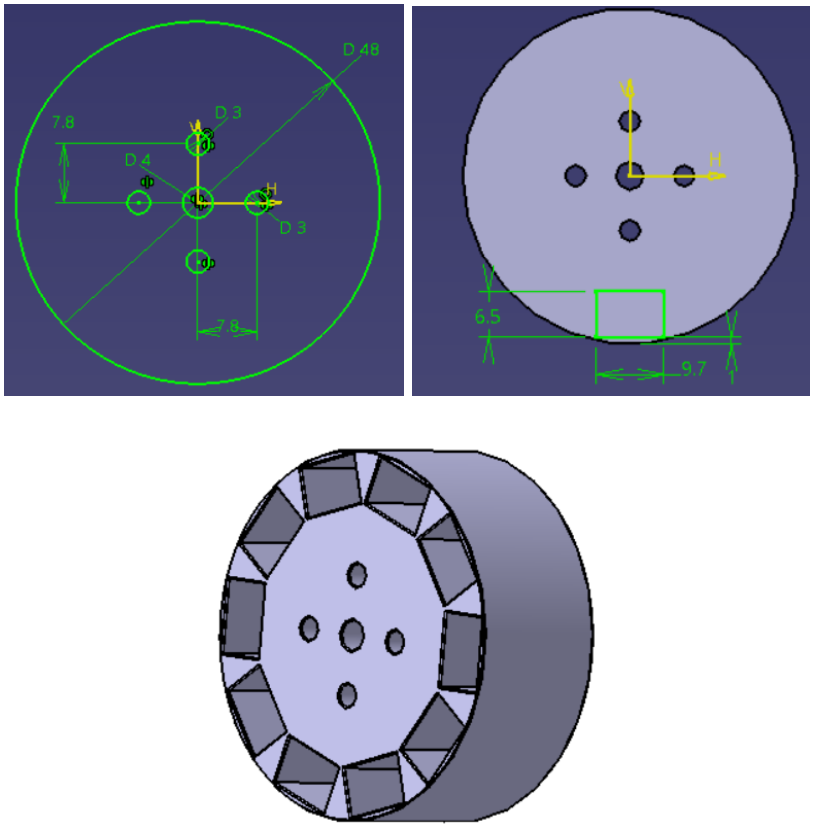


Figure 47. Engineering Specifications for 10-holes Magnetic Wheel (3D-Printing).

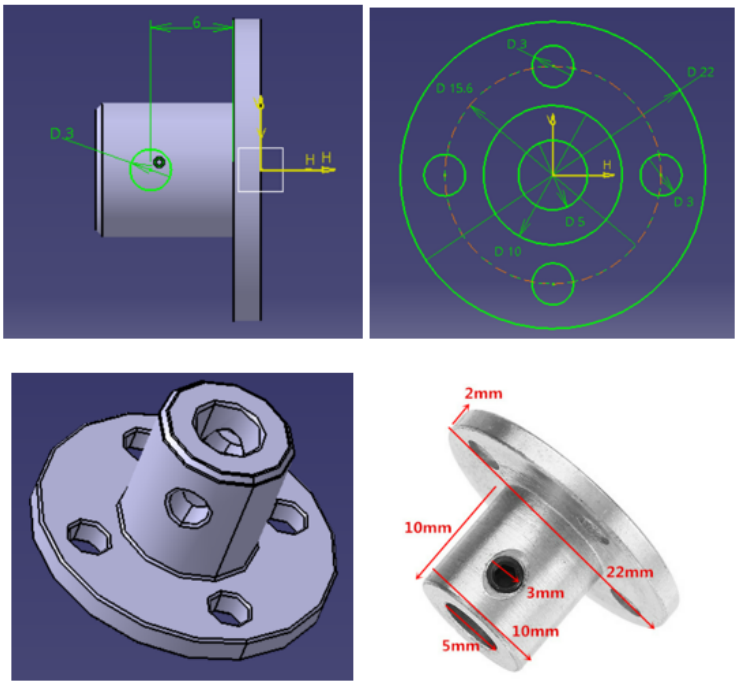


Figure 48. Engineering Drawing for Rigid Flange Coupling [10] (purchased).

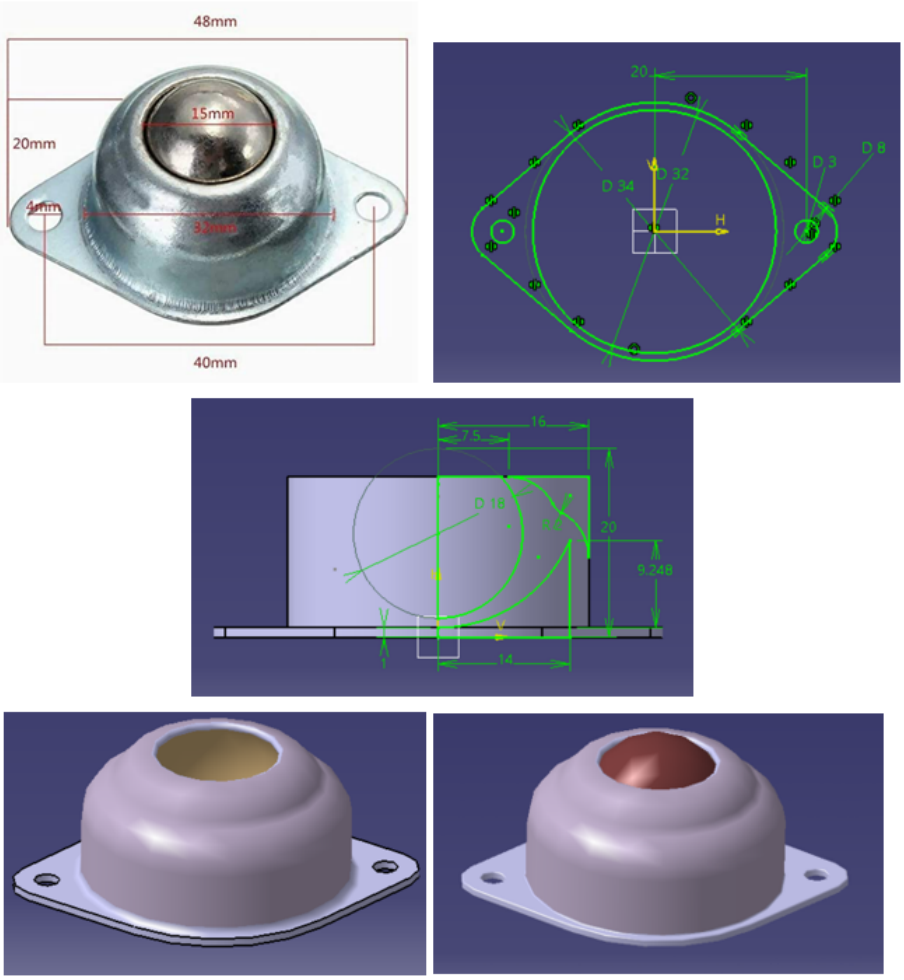
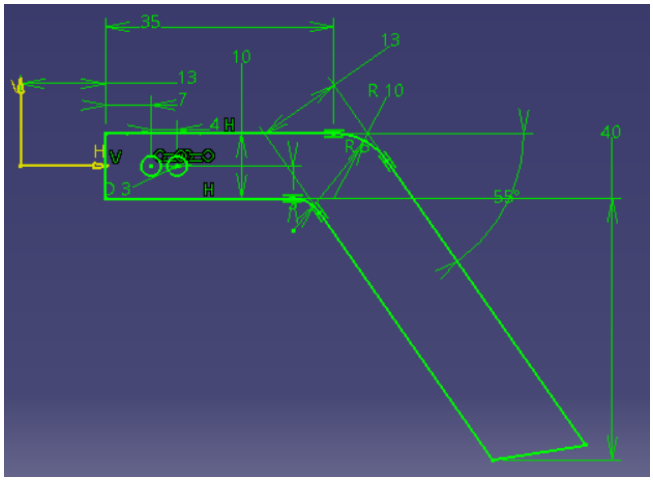


Figure 49. Engineering Drawing for Steel Ball Universal Wheel [11] (purchased).

c. Mechanical Parts Responding for Pushing the Beam



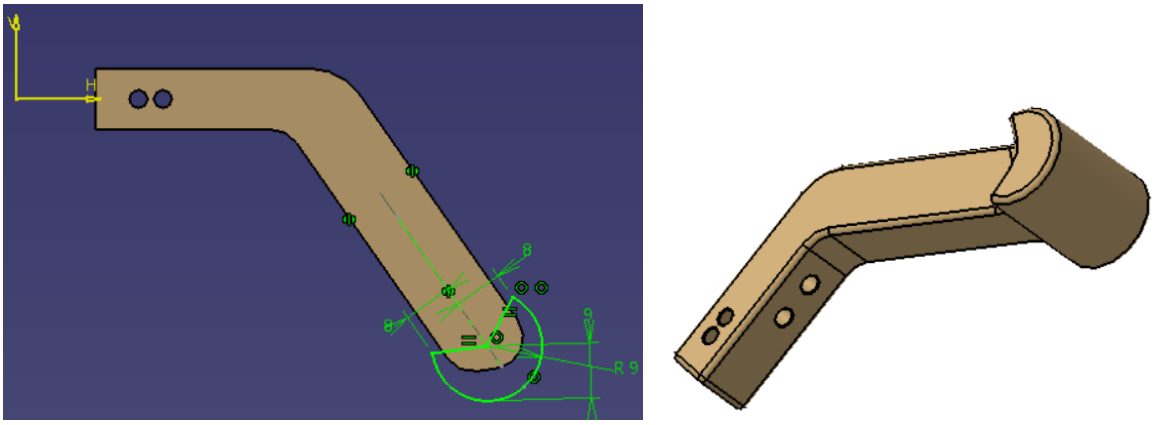


Figure 50. Engineering Drawing for Pushing Stick (3D-Printing).

d. Mechanical Parts Responding for Mounting Electrical Components

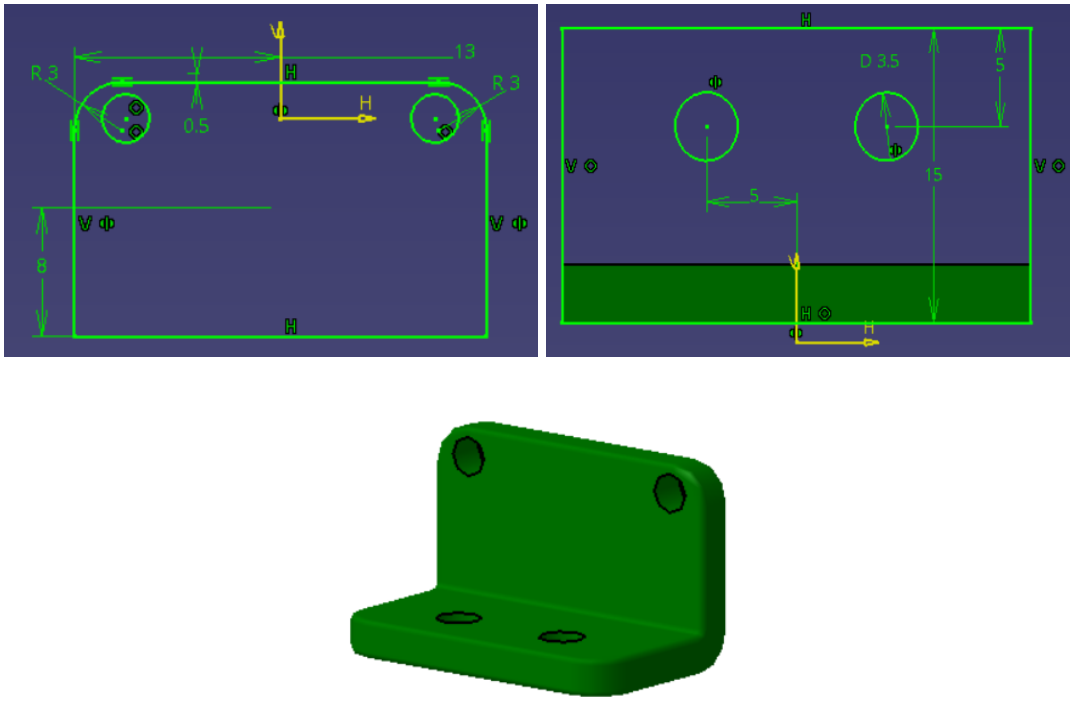


Figure 51. Engineering Drawing for Distance Sensor Mounting Bracket (3D-Printing).

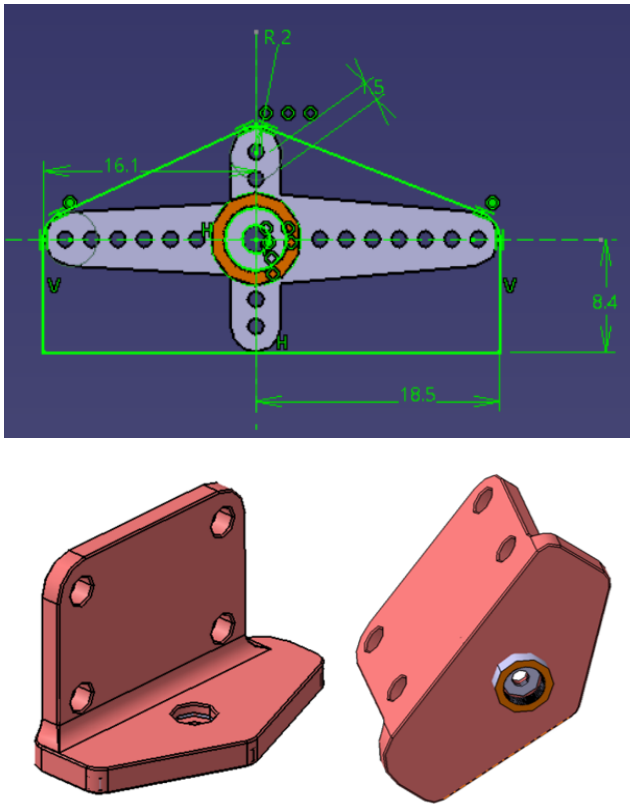


Figure 52. Engineering Drawing for Camera Mounting Bracket (3D-Printing).

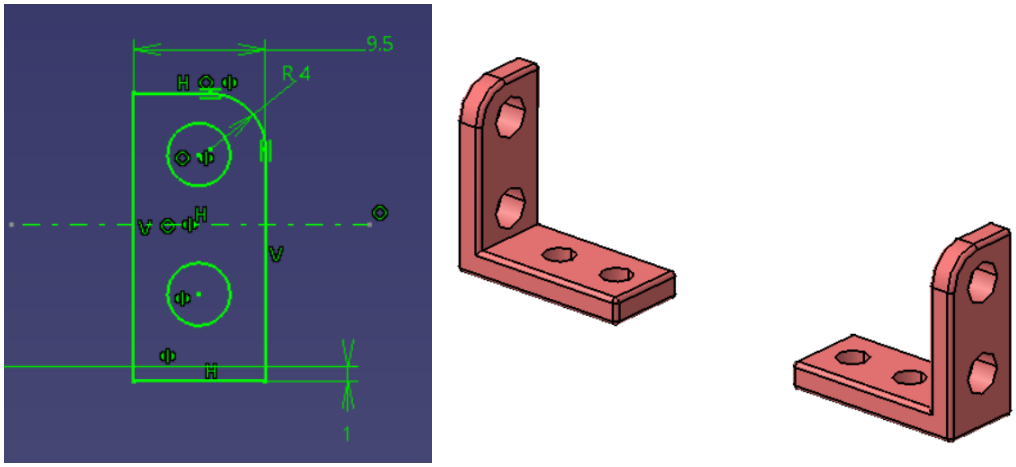


Figure 53. Engineering Drawing for Servo Bracket (3D-Printing).

V. ELECTRICAL/ELECTRONIC SUBSYSTEM

1. Detailed Description of the Electrical/Electronic Subsystem

a. Initial Concept and Final Solution

i. Power Delivery

Initially we had designed our power delivery of our robot using a LiPo battery for our main power source, dropping the voltage through buck converters. That is still the case. The only significant change to our power system was adding an additional buck converter due to the servo motors added.

The first factor we needed to look at for our power system We went with a 4S LiPo battery, rated at 14.8v. Although none of our electronics required a voltage that high, we went with a higher voltage for better power performance.

With a higher voltage we were able to reduce the capacity needed on our battery. After simulating our power requirement.

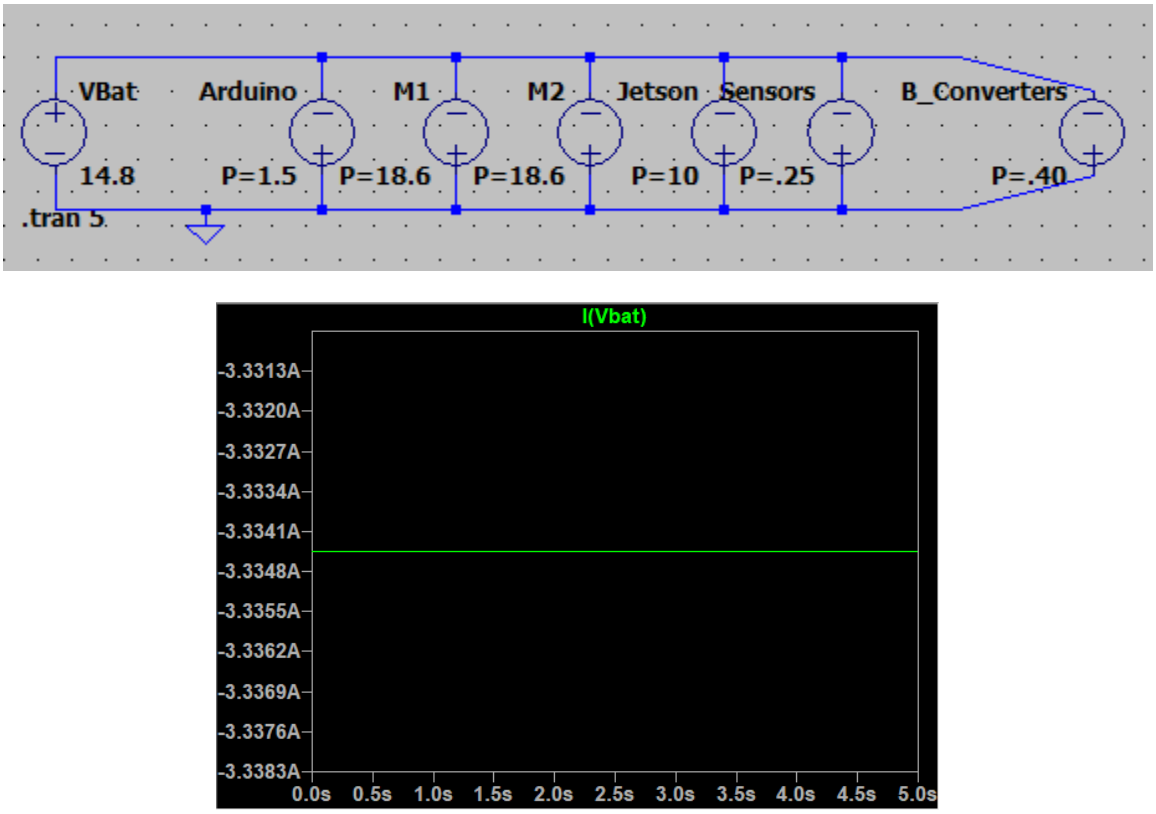


Figure 54. Power Simulation Diagram

We ended up choosing a battery with a 1300maH capacity. To deliver power to all of the electronics into our system, we fed the battery into five 75w buck converters.

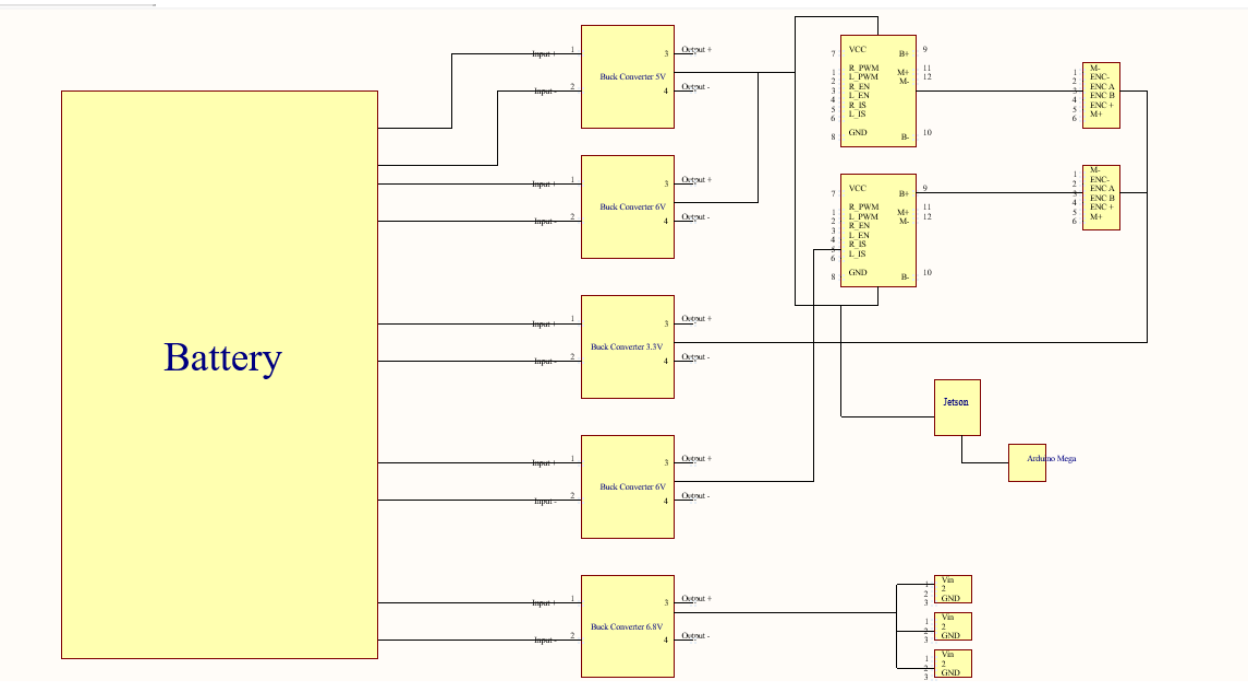


Figure 55. Buck Power Diagram

Since all of the bucks are rated for 5A, every electronic is able to get the power needed with some overhead. When accounting for buck efficiency we are still looking at a ~15 Minute runtime on our robot.

To physically route power to all of our electronics, we wanted to ensure that all of our connections stayed secure while our robot was moving around. To do this we adopted JST connectors into our harnesses, allowing us to connect our electronics to different modules easily.

ii. Motors

Initially for our design the only motors we planned on including into our robot were the two main drive robots. The FIT-052s proved to be enough to mount onto the beam and drive up the beam. When it came off the beam however, they did not have enough power/friction on the wheels to pull off the beam with the magnets. We needed a way for our robot to get off the beam.

We decided the best way to do this was with servo motors. The idea was that we would use two servo motors to assist the robot to push off the beam. After being given a force requirement, we looked for servo motors that matched the spec as shown in Table 7.

Table 7. Pugh Matrix of Servo Motor for Pushing Stick

Criteria	Baseline	DS3225	SG90 9g
Controllability	2	1	1
Speed	1	1	1
Power Draw	1	2	1
Price	1	2	1
Size	1	2	1
Total		**8**	**5**

We ended up choosing the based on the price, voltage requirement, and power requirement.

We also wanted to add another servo motor to our robot to give our camera the option to look perpendicular to the direction of movement. This would allow our robot to locate the beam a lot easier and allow turning our robot to be a lot less complex. To decide on what servo to use for this application we used these categories as shown in Table 8.

Table 8. Pugh Matrix of Servo Motor for Camera Rotation

Criteria	Baseline	MG90S	SG90 9g
Difficulty	2	1	1
Accuracy	1	1	1
Speed	1	2	1
Functionality	1	2	1
Cost	1	2	1
Total		**8**	**5**

We chose the MG90 due to its reliability, precision, and form factor. Due to the metal gearing, the MG90 is much sturdier and is able to handle the light load of the camera with little wobble.

iii. Sensors

Originally, we only planned on using machine vision for our robot, so we only considered a Raspberry Pi Cam 2 as our sensor, which plugged into our NVIDIA Jetson. After doing some initial testing with our design we learned two things about our model; our robot didn't know when to back off and go down the beam, and it struggled to stay straight on the beam.

To fix the first problem we added our first additional sensor, a distance sensor. To decide on the type of sensor we wanted to use for a distance we decided on these factors as shown in Table 9.

Table 9. Pugh Matrix of Distance Sensor

Criteria	Baseline	IR Sensor	Ultrasonic Sensor
Object Detection	1	1	-1
Size	1	1	-1
Distance	1	1	-1
Reliability	1	1	1
Total		**4**	**-2**

We chose this sensor due to the low cost, the range on it, and the precision that comes with it. For $5 the precision is really impressive for the sensor, it has a tolerance of ± 2 mm. The other big reason for selecting this sensor is that it uses the I2C protocol, saving GPIO.

For our second sensor we wanted to use an IMU, to detect if our robot wasn't going straight. To decide on an IMU we looked at these factors as shown in Table 10.

Table 10. Pugh Matrix of Detection Solution

Criteria	Baseline	GY-521	HC-SR04
Difficulty	2	1	1
Accuracy	1	1	1
Speed	1	2	1
Functionality	1	2	1
Cost	1	2	1
Total		**8**	**5**

We went with the GY-521, which utilizes the MPU6050 chip. The GY-521 is cheap ($3), includes a gyroscope and accelerometer, and is well documented regarding Arduino compatibility. It also utilizes I2C communication, making it easy to integrate with the rest of the electronics. We wanted to avoid IMUs with magnetometers, as we were worried our magnetic wheels would mess with the reading.

iv. Microcontroller

The most significant change in our electronics had to be the switch from the ESP-32 to the Arduino Mega. When initially considering our controllers, we never included the Arduino Mega due to its price and form factor. This was a mistake, as we ultimately ended up going with the Arduino Mega.

Something we should have thought about and included in our initial microcontroller Pugh matrix when discussing the ESP-32, is the complexity of programming and documentation of the system. We saw the ESP-32 had “more” GPIO, was cheaper than the Arduino boards, and was faster.

The initial problem with the ESP-32 we first ran into was the fact that a lot of the talking point GPIOs were limited. Some GPIO could only be used as inputs, some only as outputs, others not at all. This made designing a proto-PCB for our harnesses significantly harder.

The final straw for changing the microcontroller was when we were not able to get readings from our IMU with the ESP-32. The ESP-32 is compatible with the Arduino library and has support. The problem is that libraries developed for Arduino aren’t always compatible for that specific chip, and with the limited time we had for programming we decided that continuing with an Arduino supported board was the way to go.

Table 11. Pugh Matrix of Microcontroller Comparison

Criteria	Baseline	ESP-32	Arduino Mega 5060
Difficulty	2	-1	1
Accuracy	1	1	1
Speed	1	1	1
Functionality	1	1	1
Cost	1	1	2
Total		**3**	**6**

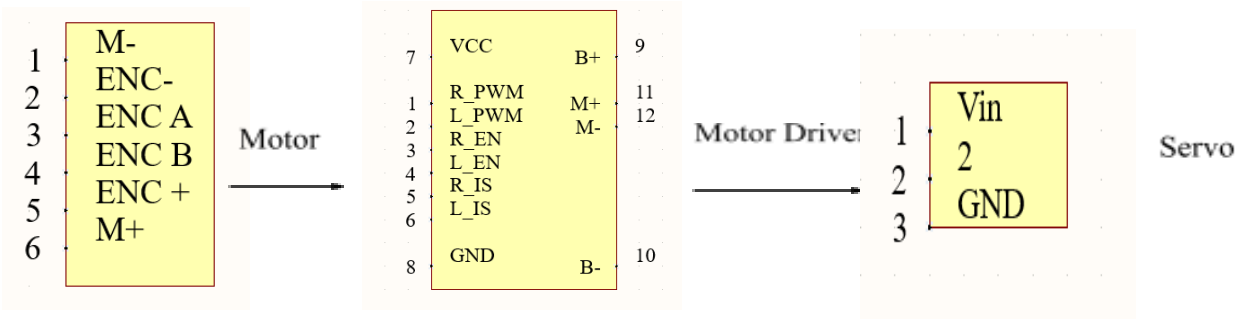
Based on the Pugh Matrix in Table 11 The Mega was the one we went with. After looking at the prices of Mega “clones” we saw that the Mega was close in price with the ESP-32 ($11 vs $15) and included enough interrupt pins for our encoders and IMU unit. The prototype shield that was compatible with the Mega was considerably easier to design connections to also.

b. Bill of Electrical Components

Table 12. Bill of Electrical Components

Quantity	Part	Model	Price	Amount used	Cost of item used
1	Battery	14V 4 Cell	$100.66	1	$19.99
1	Charger	HB6 RC		1	$33.99
4	Buck Converter			4	$14.99
5	Driver	BTS7960		2	$10.40
2	Motor	Geared DC	$52.78	2	$39.80
2	Servo Motor	DS3225 25KG	$46.62	2	$33.99
2	Distance Sensor	VL53L0X		1	$5.00
1	Gyro Sensor		$3.00	1	$3.00
2	Micro Servo Motor	SG90 9g	$9.32	1	$4.40
4	Connectors	Wires	$11.23	1	$2.65
100	Fuses		$10.59	2	$0.20
3	Proto Board		$17.07	1	$5.66
26	JSTs		$48.76	1	$1.77
Total			**$300.03**	**Total on Robot**	**$175.83**

2. Detailed Circuit Schematics of Each Component



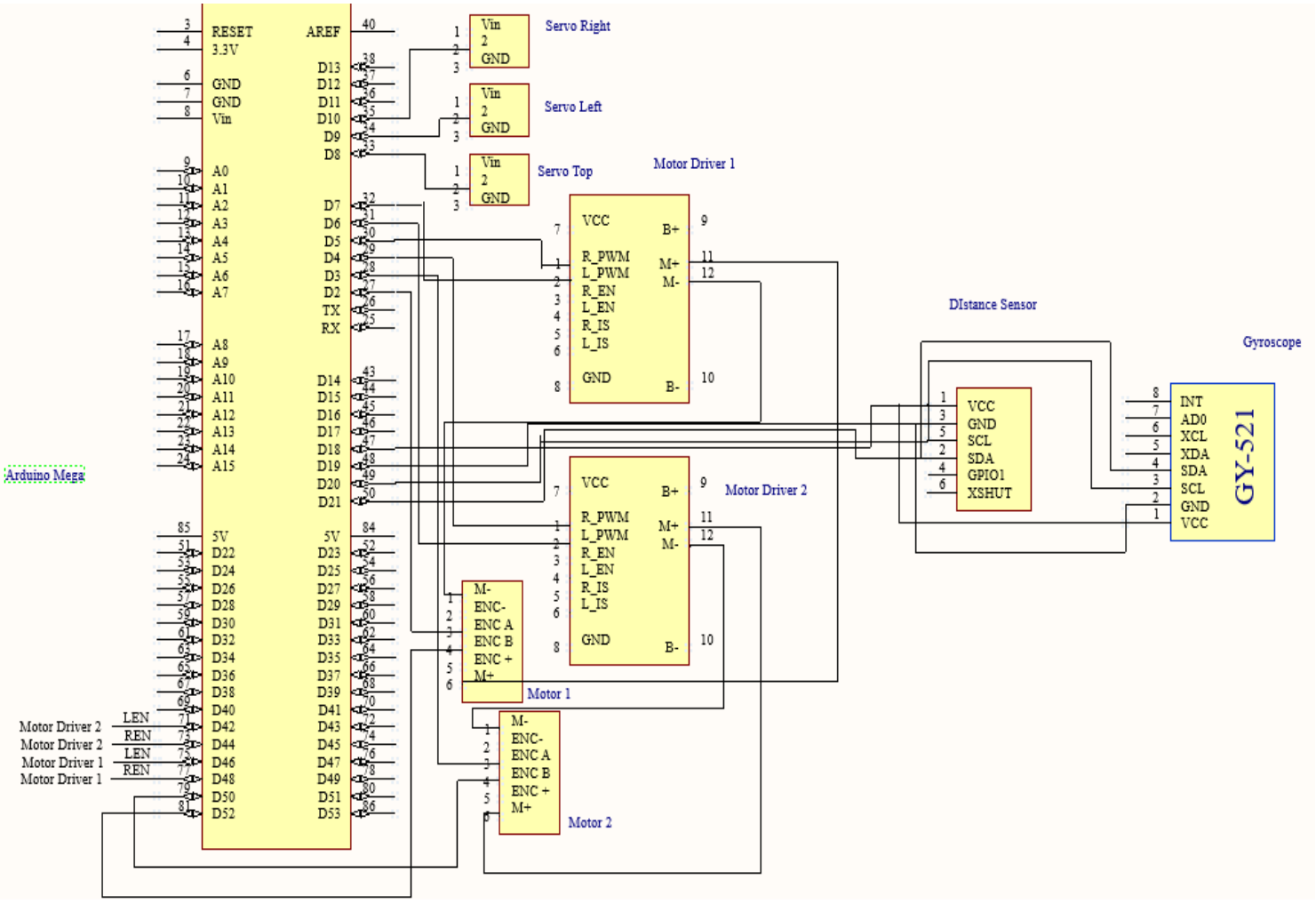


Figure 56. Wiring Diagram

VI. COMPUTER/SOFTWARE SUBSYSTEM

1. Detailed Description of the Computer/Software Subsystem

a. Initial Concept and Final Solution

When building an autonomous robot, organization is important as there may be many systems running at any given moment. For instance, there will be vision systems running on the Jetson Nano and then motor control systems running on the microcontroller. Robot Operating System, ROS, is a meta-operating system that can be implemented on the Jetson Nano to achieve

an organized framework for such systems. ROS also includes many features that could prove themselves useful for future development. Features such as ROSserial and Teleop twist, which will prove themselves useful during development.

i. ROS Nodes

The main feature that ROS provides is the idea of nodes. Nodes can be viewed as scripts of code that can be built in a package with other nodes or launched independently to perform specific tasks. Nodes can easily communicate to each other by either Publishing or Subscribing to Topics of data.

ii. ROSserial

ROSserial is a node that can be downloaded and built in a ROS framework. The objective of this node is to connect external devices with serial communication to other nodes running in ROS. It is very useful and easily implemented.

iii. Teleop Twist

Similar to ROSserial, Teleop Twist is a node that can be downloaded and built in a ROS framework. This node's objective is to read inputs from external devices like keyboards or gamepads and turn the readings into useful data. Essentially, this allows us to test out the robot manually.

To achieve certain goals within our robot, such as beam detection, alignment and climbing, we pursued a few different methods. These methods that will be described revolve around the use of a vision system, which given enough development time, can be very flexible and robust with a variety of environments. In order to make a decision on how to achieve our overall goals, the team developed a Pugh Matrix as shown in Table 13 that represents our research.

Table 13. Image Processing Pugh Matrix

Criteria	Baseline	Object Detection	Color Detection	Image Classification
Difficulty	1	-1	0	1
Accuracy	1	1	-1	0
Speed	1	-1	0	1
Functionality	0	2	1	-2
Precision	-1	1	-1	0
Repeatable	1	1	-1	1
Total	0	3	-2	1

iv. Color Detection

One approach we researched to incorporate with our vision system is the use of color detection. Color detection is the method of finding given ranges of color or perhaps hues of color, to distinguish objects from one another. This method is generally easy to implement, especially with libraries such as OpenCV with Python. To better understand where color detection works best and where it struggles, Figure 57 shows an example picture that would be excellent for color detection. What makes this a good example is the distinct hues of colors, the separations of these colors, a non-noisy background and the clearness of the objects in the photo. Though, when dealing with the real world as shown in Figure 58, it becomes more complicated. For instance, looking in the background of the image shows there are potential conflicts that could cause the color detection to give faulty data. With this approach, it would be easy but potentially challenging to fight through faulty data that the camera sees.

Figure 57. Color Detection Illustration Example

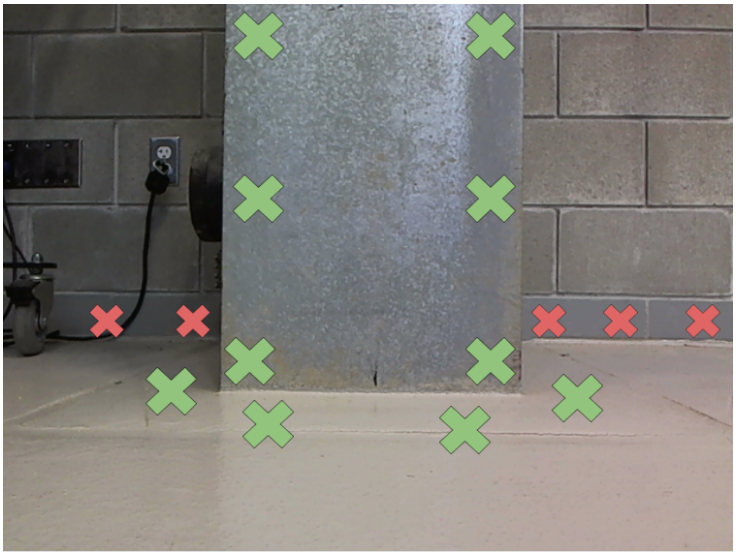


Figure 58. Color Detection Illustration of Beam

v. Image Classification

The initial method of vision detection that was being considered for this robot was image classification. Image classification involves defining a set of classes, or the objects that are desired to be recognized in the images presented to the trained model. Under each of these classes, a data set of images that include that object or class in the frame are fed into the deep learning model to train it on recognizing that specific class. Image classification function is essentially to return true or false and a confidence percentage on the class it declares as "true" based on each frame that is passed through. An example of the output can be seen in Figure 59.

Figure 59. Vision Output

For this application, the plan was to train four different classes. The first class is the “right side” which indicates to the robot that it still needs to move forward to reach the desired position of being right in front of the beam to approach. Next was the “front side” class which indicated that the robot was in the proper position to approach the beam and begin climbing. The third class. “left side” indicated to the robot that it overshot the desired position and drove past the beam. This class was used as a safety measure so that the robot would not get lost if it failed to recognize the front of the beam on first pass. For most applications, a background class is defined and taught for instances where no objects of interest are in frame so that no false positives occur i.e. the model detects that a certain class is present when it’s not because the model doesn’t know that there are possible instances of no objects in the frame. This approach was also used in this robot's training model.

The advantages of using this vision method were very enticing upon initial research. Google’s Teachable Machine was used to create the image classification model which provided a very easy and user friendly interface to create and edit models so that anyone on the team, regardless of background, could work with it. thousands of images could be trained to classes in 2-3 minutes and datasets could be edited in the same amount of time. The models produced could be exported to Jupyter Notebook and tested on the user’s pc at any time. Unfortunately, there were a number of drawbacks and limitations to this model that ultimately made this technique inferior to other detection methods. The first con of using image classification is the lack of data that can be accessed from the images that are passed through the model. As discussed previously, the model returns a binary value indicating whether or not the object is in frame or not. It does not provide any detail about the object’s location in the frame or how much of the object is in the frame. This can lead to inaccuracies and inconsistencies in the output of the model. Object detection models can provide a bounding box and centroid on the target object whereas this model cannot. Another consequence of the lack of object data is that the robot will not have enough data to determine if it's in an appropriate position to begin mounting the beam. The robot can determine if it is in front of the beam but there may be an offset where the robot will approach the beam too close to one of the edges and not be able to climb properly. To fix this issue, more subclasses would need to be made to teach the beam centered in the frame and would be very inefficient. Lastly, Google's Teachable Machine exports the model as a tensor flow model. This robot utilizes the Jetson Nano for image processing which operates on Linux. The Jetson Nano comes with a specific version of TensorFlow pre-installed on it but this version is not compatible with the version the Teachable Machine model utilizes. Unfortunately, trying to install and implement the appropriate version of TensorFlow on the Jetson Nano was unsuccessful due to incompatibility with other packages required for other processes performed on the Jetson Nano. Due to these drawbacks, Google’s Teachable Machine model was abandoned.

vi. Object Detection

The most prominent approach that was researched was object detection. In a way, all these methods are a form of object detection, however true object detection thoroughly learns an item from data. It learns not only the color and shape of an item, but also its orientation and location in the picture. The research concluded that it would help significantly in goals such as alignment and beam recognition. To build a better understanding on this approach, this can be broken down in three topics; Annotation, Training, and Model selection.

Annotation is an important and necessary step to achieve such a system. It is what a user specifies as important in many photos across a dataset of photos. Essentially, when a photo is annotated, what is happening is a user creates an area of interest that captures the item that is needed. With this, an .XML file is produced that ties the coordinates of the area to the photo it was developed from. These coordinates tell a program where the item is in the image, then is fed into the process of machine learning. Figure 60 shows an example of what annotating photos looks like. Figure 61 is the details of the .XML file created of that photo previously discussed.

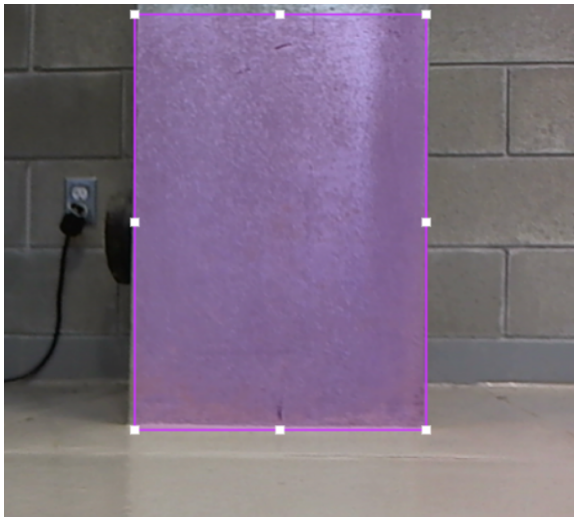


Figure 60. Illustration of Annotating a photo

```
<?xml version="1.0"?>
- <annotation>
    <folder>my-project-name</folder>
    <filename>image_211.png</filename>
    <path>/my-project-name/image_211.png</path>
  - <source>
      <database>Unspecified</database>
    </source>
  - <size>
      <width>640</width>
      <height>480</height>
      <depth>3</depth>
    </size>
  - <object>
      <name>beam</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>193</xmin>
        <ymin>9</ymin>
        <xmax>399</xmax>
        <ymax>320</ymax>
      </bndbox>
    </object>
</annotation>
```

Figure 61. XML Details of photo

Other considerations include training and model choice, which go hand in hand as each model may require different forms of training. For single board computers, there are a plethora of options that all work well in their own respects. The team decided on using a SSD Mobile Net model that is in a VOC training format, purely based on its ease of use and friendliness to the OS used.

vii. Additional Features

A feature that was considered to be used alongside the three main approaches as discussed earlier, was the potential use of a sort of edge detection and an addition of a gyroscopic module. OpenCV, a widely used computer vision library, was utilized to test the additional detection feature in order to isolate the beam and its landmarks. With this, this could potentially give information to assist with beam alignment when climbing. More specifically, keeping the robot in the middle of the edges detected. This feature can be accomplished with what is known as Canny/Gaussian Detection or Contour Detection. On top of these software additions, a physical gyroscope could prove useful while mounted on the beam as information on robot tilt can be monitored.

viii. Canny/Gaussian Detection

Canny detection creates binary images and filters out everything in the image except the edges of cleared defined objects as seen in Figure 62.

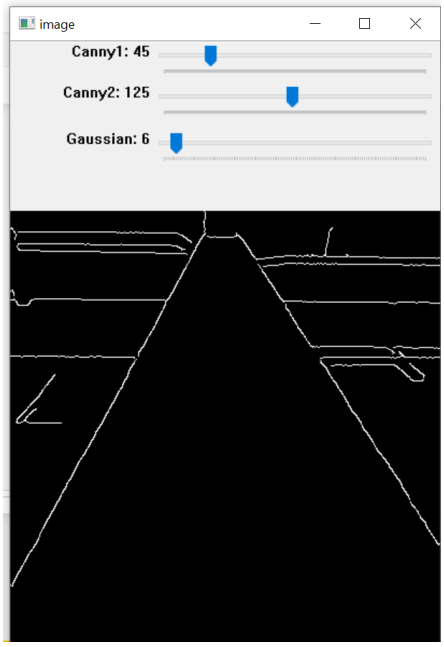


Figure 62. Canny Detection

To accomplish this output image, canny and gaussian filters were applied. The gaussian filter is used to slightly blur the image to reduce noise while the canny filter amplifies the contrast in the edges of separate objects in the frame. The problem with this technique was the variability in lighting conditions. As the light changes, the canny and gaussian filters need to be adjusted as well to filter out noise and these variables need to be adjusted manually which will not be possible for this robot's desired implementation. It is also very difficult to isolate what edges to monitor and stay between.

ix. Contour Detection

Contour detection was another method that was researched. Contour detection works in a similar way to Cany/Gaussian detection except it isolates shapes or “blobs” instead of just the edges as seen in Figure 63. The centroid or center point of these blobs can then be extracted to use a landmark for the robot to stay aligned with.

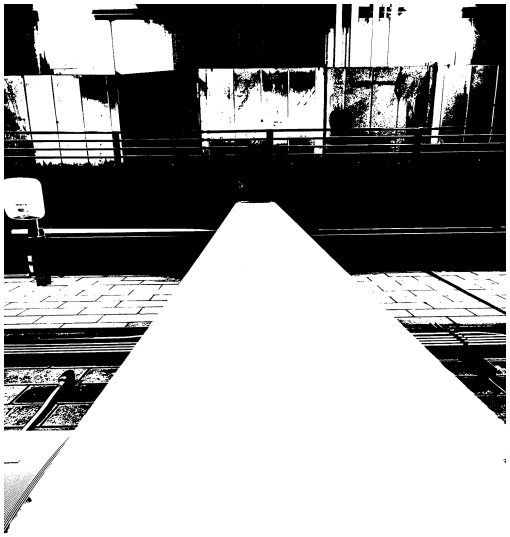


Figure 63. Contour Detection

While this method is a step better than canny/gaussian detection, it also has similar complications. Like the canny/gaussian detection filters, the filters used for contour detection need to be manually tweaked in order to produce a clear image as seen in Figure 62 but different lighting conditions affect the proper filtering values needed to produce the clean image. The method also detects every shape it finds in the image and sometimes separates shapes when it is not appropriate, which can lead to errors when finding a centroid value to center with.

x. Gyroscope

Once it was concluded that basic image processing may not suffice for ensuring alignment on the beam, a gyroscopic sensor was investigated to track the robot's alignment while it was climbing the beam. The sensor that was chosen was the MPU 6050 seen in Figure 64.

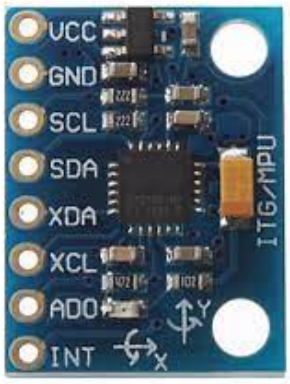


Figure 64. MPU 6050

This sensor communicates via I2C to the Arduino Mega board and provides yaw, pitch, and roll values based on its orientation with respect to gravity. For this robot’s implementation,

the pitch value was monitored and sent an interrupt to the program whenever the set pitch threshold was breached either high or low. When the MPU sensor read that it was outside this threshold, that meant the robot was veering left or right on the beam and it would soon drive off. Depending on which way the sensor read outside the threshold, it was all a function that would overdrive one of the motors to steer it back into alignment by monitoring the pitch value readings. Through bench testing and live trial runs, it was observed that this technique was very effective in monitoring the robot's orientation on the beam and making appropriate adjustments so it was ultimately included in the final design paired with object detection.

xi. Motor PID

An important aspect of the robot is the ability to maneuver in a consistent manner. Given that the project uses DC motors, there can be some loss in control if not properly addressed. However, taking this into account early on, the DC motors that were decided upon have encoders on the back of them. Encoders are important as it allows the use of control methodologies such as PID, which tune the speed of the motors to move at nearly the same rate. By having properly tuned motors, the robot can reliably travel in straight paths.

b. Bill of Computer/Software Components

Table 14. Bill of Computer Software Components.

Quantity	Part	Model	Price	Amount used	Cost of item used
1	Camera	Raspberry Pi	$32.33	1	$30.50
1	Jetson Nano	NVIDIA	$157.94	1	$149.00
1	Wi-Fi Module		$24.32	1	$22.00
1	Microcontroller	Arduino Mega2560	$15.89	1	$15.89
Total			**$230.48**	**Total on Robot**	**$217.39**

2. Detailed Code Listing

a. Flow Chart

Before detailing all of the code that is present in this project, a look at a general overview of how the robot will be operating is shown in Figure 65.

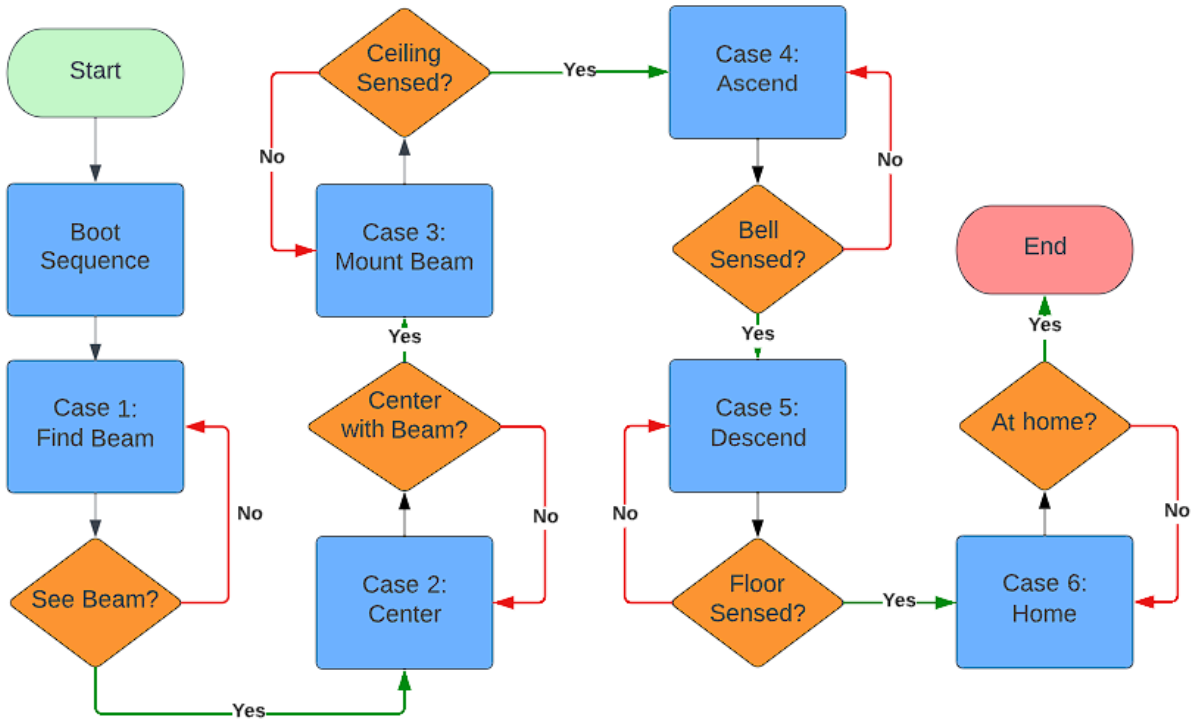


Figure 65. Robot Sequence Flow Chart

a. Arduino Motor Control and Case Functionality

In order to have success with this project, developing a flow of decisions makes troubleshooting easier. By using a switch case, we can achieve just that. Figure 66, 67, 68, and 69 illustrates some of the first few sequences the robot goes through.

```
switch (Robot_Sequence) {
  case 0: //Wait for bootup to occur
    servoT.write(10);
    valueDetect = 0;
    approachValue = 0;
    visionState.data = valueDetect;
    approach.data = approachValue;
    if (s >= 0) {
      pub.publish(&visionState);
      pub2.publish(&approach);
      Robot_Sequence = 1;
    }
    break;
```

Figure 66. Case 0: Wait for bootup

```
355     case 1:   //Find Beam
356       servoT.write(10);
357       if (s == 2) { // Move forward with PID if not center
358         digitalWrite(RMotorEN_1, HIGH);
359         digitalWrite(RMotorEN_2, HIGH);
360         digitalWrite(LMotorEN_1, HIGH);
361         digitalWrite(LMotorEN_2, HIGH);
362 
363         analogWrite(RMotorFWD, Rforward_SPEED);
364         analogWrite(LMotorFWD, Lforward_SPEED);
365       }
366       if (s == 0) { // Move forward with PID if not center
367         digitalWrite(RMotorEN_1, HIGH);
368         digitalWrite(RMotorEN_2, HIGH);
369         digitalWrite(LMotorEN_1, HIGH);
370         digitalWrite(LMotorEN_2, HIGH);
371 
372         analogWrite(RMotorFWD, Rforward_SPEED);
373         analogWrite(LMotorFWD, Lforward_SPEED);
374       }
375       if (s == 1) { // Stop if center
376         Robot_Sequence = 2;
377         digitalWrite(RMotorEN_1, LOW);
378         digitalWrite(RMotorEN_2, LOW);
379         digitalWrite(LMotorEN_1, LOW);
380         digitalWrite(LMotorEN_2, LOW);
381 
382         analogWrite(RMotorFWD, 0);
383         analogWrite(LMotorFWD, 0);
384         statepreviousMillis = currentMillis;
385         LENCAx = LENCA; // Record distance traveled
386         RENCAx = RENCA;
387       }
388       break;
```

Figure 67. Case 1: Find Beam

```
389     case 2:   //Wait 2 Seconds
390       if (currentMillis - statepreviousMillis > 2000) {
391         Robot_Sequence = 3;
392         turnLENCA = LENCA;
393         turnRENCA = RENCA;
394       }
395       break;
```

Figure 68. Case 2: Pause for 2 seconds

```
396     case 3: //Backup to center
397       if ((turnLENCA - LENCA > 400) && (turnLENCA - RENCA > 400)) {
398         digitalWrite(RMotorEN_1, LOW);
399         digitalWrite(RMotorEN_2, LOW);//If rotation distance is 400, stop
400         digitalWrite(LMotorEN_1, LOW);
401         digitalWrite(LMotorEN_2, LOW);
402 
403         analogWrite(RMotorFWD, 0);
404         analogWrite(LMotorFWD, 0);
405         analogWrite(RMotorBWD, 0);
406         analogWrite(LMotorBWD, 0);
407         statepreviousMillis = currentMillis;
408         Robot_Sequence = 4;
409       }
410       else if ((turnLENCA - LENCA < 400) && (turnLENCA - RENCA < 400)) {
411         digitalWrite(RMotorEN_1, HIGH);
412         digitalWrite(RMotorEN_2, HIGH);//If rotation distance is not 400, continue
413         digitalWrite(LMotorEN_1, HIGH);
414         digitalWrite(LMotorEN_2, HIGH);
415 
416         analogWrite(RMotorBWD, 100);
417         analogWrite(LMotorBWD, 100);
418       }
419 
420       break;
```

Figure 69. Case 3: Backup robot before centering

Other features within the robot control code include encoder reading and rpm calculation. By using the encoders on the motors, as shown in Figure 70, we can perform calculations like RPM in Figure 71. This is important because it allows PID control of motors.

```
828   void leftENCA() {
829 
830     if (digitalRead(LEncoderB) == HIGH) {
831 
832       LENCA++; //If B side goes high, going forward so increment
833     }
834 
835 
836     else {
837       LENCA--; //Else if B is LOW, going backward so decrement
838     }
839   }
840 
841 
842   void rightENCA() {
843 
844 
845     if (digitalRead(REncoderB) == HIGH) {
846 
847       RENCA--;//If B is LOW, going backward so decrement
848     }
849 
850     else {
851       RENCA++;//Else B side goes high, going forward so increment
852     }
853   }
```

Figure 70. Encoder Reading

```
currentMillis = millis();
if (currentMillis - previousMillis >= interval) { //takes RPM reading every 150 ms
  previousMillis = currentMillis;
  double LENCA_meas = LENCA - prevLENCA;
  double RENCA_meas = RENCA - prevRENCA;
  prevLENCA = LENCA;
  prevRENCA = RENCA;
  rpm_L = (double)((LENCA_meas / 374) * 400);
  rpm_R = (double)((RENCA_meas / 374) * 400);
  get_MPU_data(); //pulls pitch yaw roll values from gy 521
}
```

Figure 71. RPM Calculation for PID

i. Python

Object Detection

To perform object detection on both the bell and beam, a programmer could either train a single model to detect both objects or train two separate models with each object allocated apart. The team decided to have two different models as we felt it was easier to focus on one object at a time during development.

Training a model for object detection can be done in many ways, depending on the model and hardware type. In this project, we heavily followed the Jetson Inference guide for object detection. As annotated as already been discussed, this next portion will illustrate how the models are implemented in code.

In order to achieve object detection and ROS integration, Figures 72 and 73 show how they are implemented. Figure 72 shows all the types of libraries that are being used as well as define the camera, display, and two separate detection models. While 73 shows the setup of flags, functions and the ROS publisher and subscribers.

```
#!/usr/bin/env python3
import jetson.inference
import jetson.utils
import rospy
from std_msgs.msg import Int8
from std_msgs.msg import Int32


cam = jetson.utils.videoSource("/dev/video0")
disp = jetson.utils.videoOutput("display://0")
net = jetson.inference.detectNet(argv=['--model=/home/jb/catkin_ws/src/jetson-inference/python/training/detection/ssd/models/beam-v2/ssd-mobilenet.onnx',
                                       '--labels=/home/jb/catkin_ws/src/jetson-inference/python/training/detection/ssd/models/beam-v2/labels.txt',
                                       '--input-blob=input_0','--output-cvg=scores', '--output-bbox=boxes'],
                                        threshold=0.75)
net2 = jetson.inference.detectNet(argv=['--model=/home/jb/catkin_ws/src/jetson-inference/python/training/detection/ssd/models/bell/ssd-mobilenet.onnx',
                                        '--labels=/home/jb/catkin_ws/src/jetson-inference/python/training/detection/ssd/models/bell/labels.txt',
                                        '--input-blob=input_0','--output-cvg=scores', '--output-bbox=boxes'],
                                         threshold=0.75)
```

Figure 72. Libraries and Two Different Detection Models Implemented

```
19
20  #cap=cv2.VideoCapture("csi://0")
21  detect = 0
22  approach = 0
23  def outputCallback(data):
24          global detect
25          detect = data.data
26  def outputCallback2(data2):
27          global detect
28          approach = data2.data
29
30  while not rospy.is_shutdown():
31      #while disp.IsStreaming():
32      pub = rospy.Publisher('data',Int8,queue_size=10)
33      sub = rospy.Subscriber('detect', Int32, outputCallback)
34      sub2 = rospy.Subscriber('approach', Int32, outputCallback2)
35      rospy.init_node('jetson', anonymous=True)
36
37      rate=rospy.Rate(8)
```

Figure 73. ROS Publisher and Subscriber Setup

Once the setup of the system is complete, the phases of detection can be established. As shown in Figures 74, 75, and 76, the code uses a few different flags in order to change the behavior.

In Figure 74, if detect and approach are both at 0, the beam detect model is initialized. With this, the robot outputs data to let the Arduino know whether to continue on its path or stop when it sees the robot centered with the beam. Stopping at a point center on the beam is important for the approach.

```
    if detect == 0 and approach == 0:
        img=cam.Capture()
        detections = net.Detect(img)
        disp.Render(img)
        disp.SetStatus("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
        flag = 0

        #Find object
        for det in detections:
            center_x = int(det.Left + det.Right) // 2
            center_y = int(det.Top + det.Bottom) // 2
            #jetson.utils.cudaDrawCircle(img, (center_x, center_y,), 5,(0,255,0,200))
            flag = 1

        if(flag == 1):
            if center_x < img.width // 2:
                #print("Move Left")
                #print("Detect: %s", detect)
                rospy.loginfo(0)
                pub.publish(0)
                rate.sleep()
            if abs(center_x - img.width // 2) <= 35:
                #print("Center")
                #print("Detect: %s", detect)
                rospy.loginfo(1)
                pub.publish(1)
                rate.sleep()
            if center_x > img.width // 2:
                #print("Move Right")
                #print("Detect: %s", detect)
                rospy.loginfo(2)
                pub.publish(2)
                rate.sleep()
```

Figure 74. First Detection Phase

In Figure 75, if detect is at 0 and approach is at 1, this continues to use the beam detect model. From which it publishes its position according to how it sees the detected beam. This is what gives the robot its ability to navigate itself closer and square to the beam.

```
    if detect == 0 and approach == 1:
        img=cam.Capture()
        detections = net.Detect(img)
        disp.Render(img)
        disp.SetStatus("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
        flag = 0
        print("Approach: %s", approach)
    #Find object
    for det in detections:
        center_x = int(det.Left + det.Right) // 2
        center_y = int(det.Top + det.Bottom) // 2
        #jetson.utils.cudaDrawCircle(img, (center_x, center_y,), 5,(0,255,0,200))
        flag = 1

    if(flag == 1):
        if center_x < img.width // 2:
            #print("Move Left")
            #print("Detect: %s", detect)
            rospy.loginfo(0)
            pub.publish(0)
            rate.sleep()
        if abs(center_x - img.width // 2) <= 10:
            #print("Center")
            #print("Detect: %s", detect)
            rospy.loginfo(1)
            pub.publish(1)
            rate.sleep()
        if center_x > img.width // 2:
            #print("Move Right")
            #print("Detect: %s", detect)
            rospy.loginfo(2)
            pub.publish(2)
            rate.sleep()

        if(flag == 0):
            #print("No object")  #No object
            #print("Detect: %s", detect)
            rospy.loginfo(2)
            pub.publish(2)
            rate.sleep()

```

Figure 75. Second Detection Phase

In Figure 76, if detect is 1 then the final phase has occurred. Within this stage, the bell detection model is being used rather than the beam. On top of this, the data is also being sent to the Arduino to help with aligning with the bell and therefore staying on the beam.

With this, the vision is concluded with all of the operations and the robot can restart to begin another run.

```
112
113     if detect == 1:
114         img2=cam.Capture()
115         detections2 = net2.Detect(img2)
116         disp.Render(img)
117         disp.SetStatus("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
118         flag = 0
119
120         #Find object
121         for det2 in detections2:
122             center_x = int(det2.Left + det2.Right) // 2
123             center_y = int(det2.Top + det2.Bottom) // 2
124             #jetson.utils.cudaDrawCircle(img2, (center_x, center_y,), 5,(0,255,0,200))
125             flag = 1
126
127         if(flag == 1):
128             if center_x < img2.width // 2:
129                 #print("Move Left")
130                 #print("Detect: %s", detect)
131                 rospy.loginfo(5)
132                 pub.publish(5)
133                 rate.sleep()
134             if center_x > img2.width // 2:
135                 #print("Move Right")
136                 #print("Detect: %s", detect)
137                 rospy.loginfo(4)
138                 pub.publish(4)
139                 rate.sleep()
140             if abs(center_x - img2.width // 2) <= 5:
141                 #print("Center")
142                 #print("Detect: %s", detect)
143                 rospy.loginfo(6)
144                 pub.publish(6)
145                 rate.sleep()
146         if(flag == 0):
147             #print("No object")  #No object
148             #print("Detect: %s", detect)
149             rospy.loginfo(2)
150             pub.publish(2)
151             rate.sleep()
152
153
154
```

Figure 76. Final Detection Phase

Start Up

Another important aspect that the team had to keep in mind when developing on a single board computer, namely the Jetson Nano, is how a start up sequence will look. Unlike a microcontroller such as an Arduino, the Jetson has an operating system. This proves to be a challenge for robotics in cases where you may want to begin operation at a robot startup.

By developing custom bash scripts and storing them as system services files, we can

achieve a startup sequence. A system service file is a file that runs upon system startup. Effectively, this allows us to integrate a startup sequence in our robot.

As shown in Figure 77, the first system that needs to be initialized is our ROScore. This is important because without it, our Publisher and Subscribers in the Jetson will not be able to begin operation. What is important here, is the ExecStart, which is unique to starting ROScore in the specific kernel.

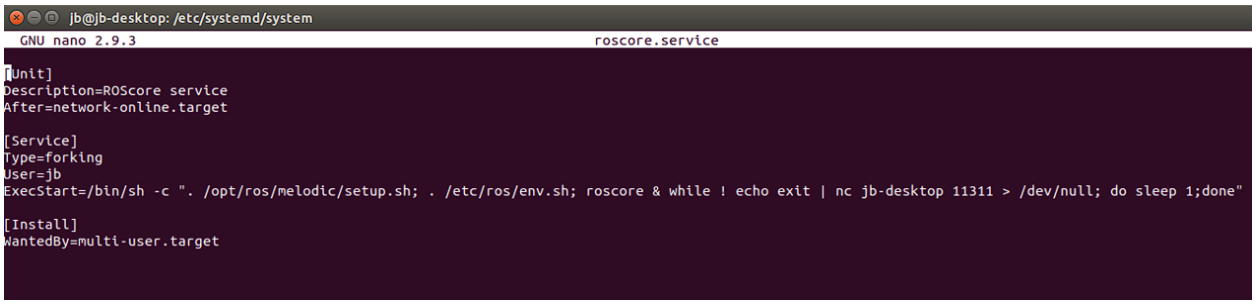


Figure 77. ROScore Startup

Since the robot uses ROS in order to achieve certain features, we can run all of the required nodes based on a launch file. As shown in Figure 78, the nodes that need to be run are the ROSserial node for serial communication and our custom vision node. The format is xml so it is very easy to integrate and build upon.

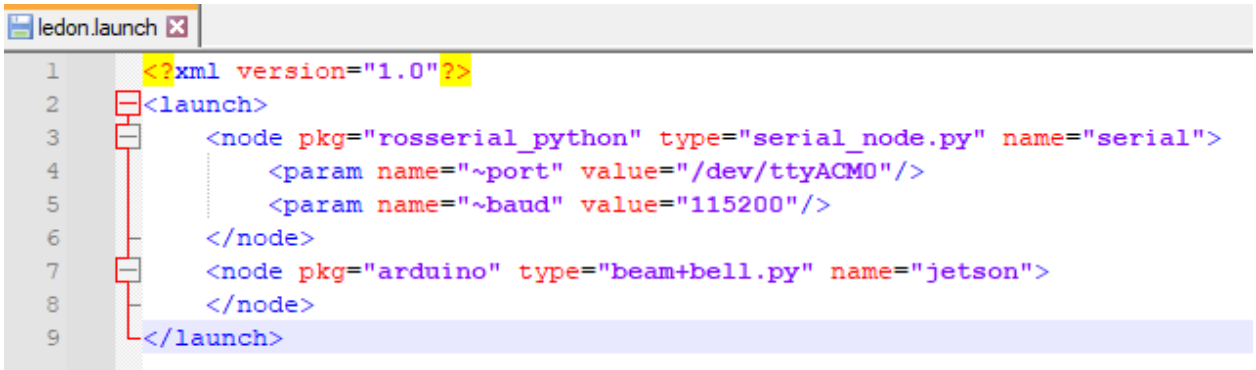


Figure 78. Nodes Launch File

With a launch file created, in this case named ledon.launch, another bash file can be created to do a few things. First, ROS needs to be sourced so that these nodes can run using roslaunch. Next, permissions need to be given to the Arduino so that it is allowed to communicate through the device's USB. In this case, /dev/ttyACM0 is the name of the port being used and sudo chmod 666 gives the allotted permissions.

Lastly, roslaunch is called upon and launches the previously created launch file. This will go through and start up all nodes that were written in there.

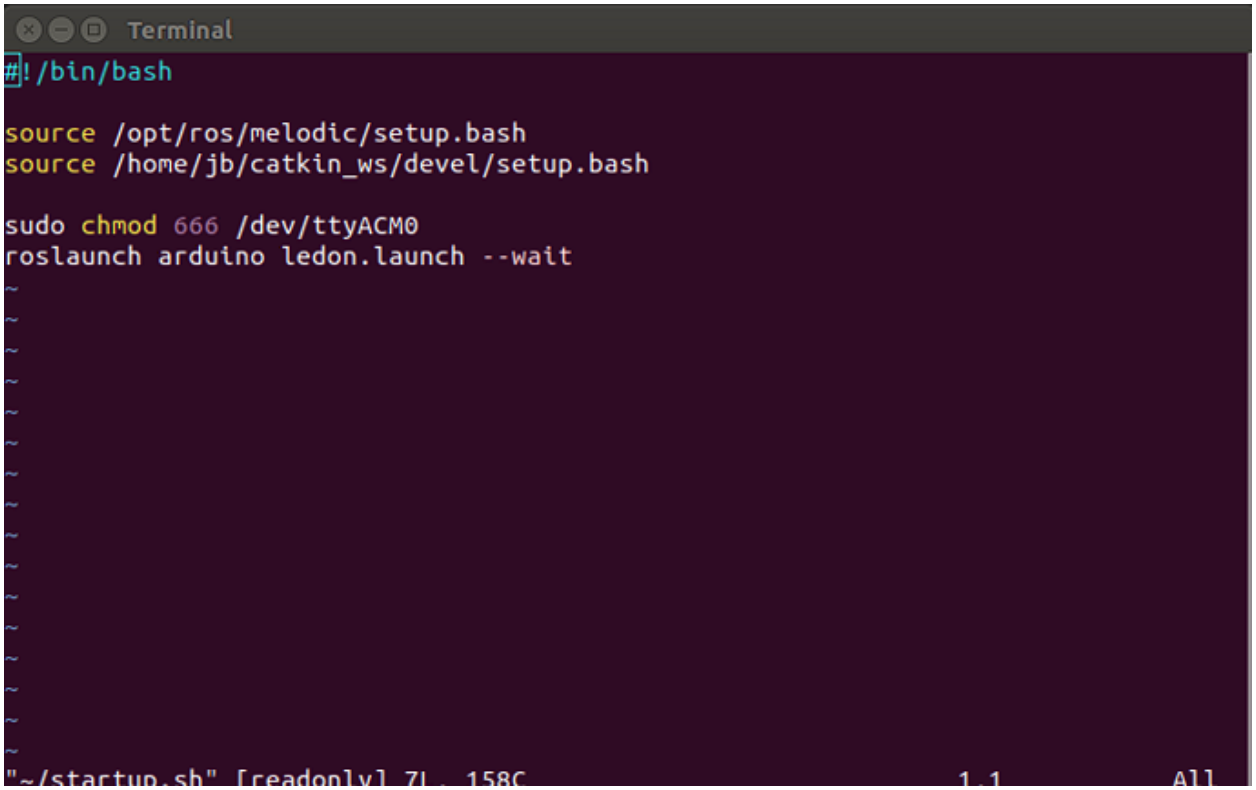


Figure 79. Give permissions and Start Nodes

VII. DISCUSSION

1. Technical Discussion

All the initial engineering specifications were set to focus on the two most important criteria, which are speed and cost as shown in Table 1, Section II. The final design is not much different from the original idea. Some component models are changed to suit the purpose and design context but still ensure the necessary functionality. The design solution is that each modification and addition is based on the test results of each phase. The obtained results in terms of engineering specification are presented in Table 15.

Table 15. Engineering Design Requirements and Validation

	Engineering Requirement	Importance	Target	Method	Status (date when complete)	Updated Result
ME	Weight	High	≤ 5lbm	CAD analysis, inspection	Done (04/01/23)	3.439lbm
ME	Volume	Moderate	< 1ft^3	CAD analysis, inspection	Done (04/01/23)	0.322ft^3
EE	Input Power	Moderate	< 22V	Pspice analysis, demonstration	Done (02/20/23)	14.8V
EE	Battery Life	High	> 20 min	Estimate through analysis	Done (02/16/23)	>5min
Team	Development/Tooling Cost	Low	< $22	Estimate through analysis	Done (04/03/23)	$13.63
Team	Design Project Cost	High	< $500	Budget analysis	Done (04/03/23)	$459.75
EE/CE	Object Detection Software	High	x% error rate	Inspection, testing	Completed	< 5% error
EE/ME	Ring Bell	High	Single joint	Testing	Completed	

In terms of Mechanical Engineering, we met the requirement of the weight and volume. Specifically, the total weight of the robot is 4,255lbm, which is smaller than meeting the criteria of less than 5lbm. The total volume of the robot is 0.922 ft^3, which is smaller than meeting the criteria of less than 1 ft^3. These results could be further improved as we are in the process of weighing the need for alignment arms. If the robot can work well based on the coding setup for the sensor and camera, the alignment arms can be removed. That will make a huge difference in the total volume and save a little weight for the robot. Because the robot's height exceeds the distance between the bell and the beam, the robot can automatically reach the bell when it reaches the top of the beam, the ringing problem is completely solved without the need for a separate arm. It's a small change from the original design.

In terms of Electrical Engineering, the sensors being used for identifying the beam should be sufficient. The distance sensor should be able to measure the distance for the robot to the beam as well with the bell. The gyroscope and accelerometer will give the robot updates on the orientation of the robot on the beam.

Even though throughout the semester changes were made regarding the electrical components, we made sure to maintain the same engineering specifications. After changing the microcontroller, adding sensors and other components the wiring of the robot changed. The input power (14.8V) met our initial target (22V). The battery life was another challenge for the electrical side. After researching and seeing that the battery if not handled properly can be dangerous, we made sure to closely calculate and simulate the battery before testing. From prototyping the battery life lasts more than 5 minutes.

In terms of Computer/Software engineering, we meet the requirements and can achieve the goals of the project in a few ways. First, the vision system is the backbone of the robot and gives it many possibilities for future development. The strong focus of the project was to give the robot as much information as possible to create some level of autonomy. By incorporating a vision system with the ability to learn objects, the robot achieves a high level of flexibility and adaptability.

The total cost of meeting the requirements of the cost specification ($459.75) compared to an initial estimate of less than $500. The tooling cost of $13.63 also met the requirement of less than $20 (see section 3, Appendix B for details).

Table 16. Total Budget of Materials

	Quantity	Part	Model	Price	Total cost including tax and shipping fee	Amount used	Cost of item used
Electrical Sub-System	1	Battery	14V 4 Cell	$19.99	100.66	1	$19.99
	1	Charger	HB6 RC	$33.99		1	$33.99
	4	Buck Converter		$14.99		4	$14.99
	5	Driver	BTS7960	$26.00		2	$10.40
	2	Motor	Geared DC	$39.80	$52.78	2	$39.80
	2	Servo Motor	DS3225 25KG	$33.99	$46.62	2	$33.99
	2	Distance Sensor	VL53L0X	$9.99		1	$5.00
	1	Gyro Sensor		$3.00	$3.00	1	$3.00
	2	Micro Servo Motor	SG90 9g	$8.79	$9.32	1	$4.40
	4	Connectors	Wires	$10.59	$11.23	1	$2.65
	100	Fuses		$9.99	$10.59	2	$0.20
	3	Proto Board		$16.99	$17.07	1	$5.66
	26	JSTs		$45.99	$48.76	1	$1.77
Computer/ Software Sub-System	1	Camera	Raspberry Pi	$30.50	$32.33	1	$30.50
	1	Jetson Nano	NVIDIA	$149.00	$157.94	1	$149.00
	1	Wifi Module		$22.94	$24.32	1	$22.94
	1	Microcontroller	Arduino Mega2560	$14.99	$15.89	1	$14.99

Mechanical Sub-System	20	Magnets	BC64	$40.32	$51.75	20	$40.32
	2	Rigid Flange Coupling	Motor Shaft Flange	$7.99	$8.47	2	$7.99
	300	Bolts		$3.99	$4.22	60	$0.80
	100	Nylon Nuts		$5.99	$6.35	20	$1.20
	300	Zipties		$11.00	$11.00	15	$0.55
	1	Steel Ball Universal Wheel		$2.00	$2.00	1	$2.00
	2	Wheel	5 magnet-hole		$13.63	7.03	$0.14
	1		9 magnet-hole			13.35	$0.33
	1		9 magnet-hole			13.35	$0.27
	2		9 magnet-hole v2			11.95	$0.24
	2		10 magnet-hole			14.71	$0.29
	2		10 magnet-hole v2			14.67	$0.29
	1	Frame	Top Frame v1			119.82	$2.99
	1		Top Frame v2			115.32	$2.30
	1		Top Frame v3			113.23	$2.25
	1		Bottom Frame v1			114.73	$2.29
	1		Bottom Frame v2			112.76	$2.24
Total					**$627.94**	**Total on Robot**	**$459.75**

One issue that was encountered early on in the building and testing of the first prototype for this robot was traction on the wheels. It was initially thought that the TPU the wheels are made out of would provide enough grip to allow the robot to climb up the beam without any issues with slipping but through testing this turned out not to be the case. Once the robot would mount the beam, the wheels would begin to slip and turn in place due to the lack of traction. To counter this issue, tear-resistance rubber was glued onto the treads of the wheels which provided much more grip and increased the friction coefficient so that the wheels would no longer slip.

Another problem that was encountered was the ability of the robot to dismount the beam at the end of its descent. The original design implemented fins to slide on the ground and guide the base of the robot to the ground. The issue that was found with this mechanism was that the fins would get stuck on the ground and could not overcome the force of friction with the ground to adequately slide on the ground. Wheels were implemented at the end of these fins in order to avoid the issue of friction and allowed the robot to smoothly roll along the ground in order to dismount the beam.

Wire management was also a difficult task when integrating the various electrical subsystems including in this robot. The wiring for all electronic components on the robot had to be stored between the first and second level of the robot, which is a very confined space. This densely packed area of wiring made it very difficult to manage connections and troubleshoot various electrical issues that were encountered throughout the development process.

2. Professional and Societal Context

a. Engineering Standards

Robotics, some may say, is made up of three primary disciplines. These three being software, electrical, and mechanical. So, when designing a robot, understanding the standards and best common practices of all three will assist in the robot's success and user safety.

i. Software Standards

Autonomous robots are becoming increasingly prevalent in a wide range of industries, including manufacturing, transportation, and healthcare. These robots rely on sophisticated software to control their movements and interactions with the environment. Unfortunately, as there is a variety of software that can achieve this, there isn't any software standardization. For instance, robotics companies such as FANUC, KUKA, and ABB all use proprietary software that is used strictly in-house [17]. The standards that will be discussed are all open source, which includes all documentation and resources available.

ROS (Robot Operating System)

ROS is an open-source, meta-operating system used for robots. It provides a set of tools and libraries for building robot applications, including a distributed communication framework, a library of common robot functionality, and tools for simulation and visualization. ROS is widely used in the robotics community and has a large user base, which makes it easy to find support and resources for development. [18]

OROCOS

OROCOS is an open-source software framework for real-time control systems. It provides a set of libraries and tools for building control systems for robots and other types of machines. OROCOS is designed for use in real-time systems and provides support for a wide range of platforms, including Linux and Windows. It also provides a set of extensions to be used alongside other robotic frameworks like ROS, Rock, and Yarp. [19]

YARP (Yet another Robot Platform)

YARP is an open-source software library for robotics that provides a set of libraries and tools for robot control and communication. It is designed to be lightweight and flexible, making it well-suited for use in applications such as service robots and mobile robots.[20]

OPC UA (OPC Unified Architecture)

OPC UA is a platform-independent, open-source communication protocol for industrial automation. It is used for exchanging data between devices and systems, including robots and other industrial equipment. OPC UA is designed for use in industrial environments and provides a high level of security, making it well-suited for use in applications such as manufacturing, transportation, and healthcare. [21]

ii. Electrical Standards

When dealing with systems that use electricity for operation, it is important to have a set of standards and guidelines to ensure safety. Depending on the type of circuit and its location, these standards may vary. In the case of a battery operated robot, the circuit will be DC and the standards that will be used will come from the IEC. The IEC stands for the International Electrotechnical Commission.

DC Color Codes

According to the IEC, DC circuits that are ungrounded will require a color that consists of a brown positive and a gray negative. Ungrounded systems are systems such as battery operated robots, also referred to as floating systems [22]. Due to this floating nature, it is not always required to ground the system and in some cases grounding a floating system may create other issues when there are shorts in the system. See Figure 65 for an illustrated guideline [23].

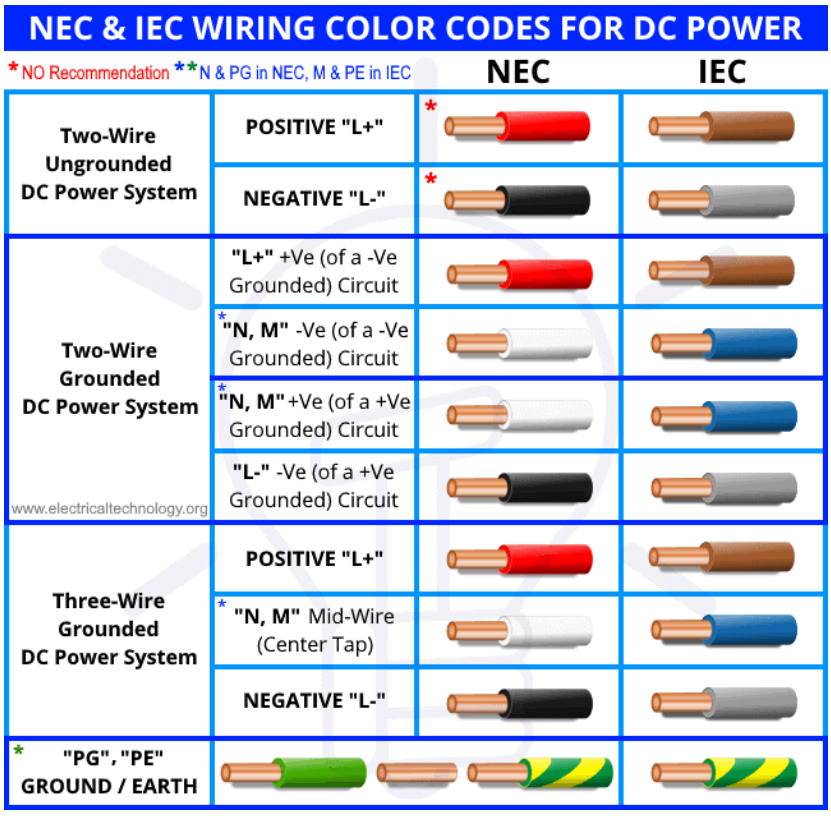


Figure 65. NEC & IEC Color Codes for DC Power

Gauge Codes

Electrical systems can operate within a very large range of power. Due to this, properly sizing wires for the application is a necessity. Improperly sized wires can cause damage to components, injury to users, or even poor performance of a system. Wires can be sized based on

pre-existing standards that correlate wire thickness and current applied. In this case, wire gauging will be based on IEC [24].

Due to the nature of mobile robot systems, the power used is generally low. However, it will still be good practice to properly size wires for safety reasons. According to Figure 66, with an application consisting of a max draw of 8A, an acceptable mm2 would be 1 or higher. In relation to the AWG size, this will place it at about 18 AWG.

American Wire Gauge -- IEC 60228 Wire Size

AWG / kcmil	Area Circ Mils	mm2	Amps TW/UF (60C)	Amps THHW (75C)	Amps THHN (90C)
18	1,620	1	--	--	
16	2,580	1	--	--	
14*	4,110	2	20 (15)	20 (15)	25 (20)
12*	6,530	3	25 (20)	25 (20)	30 (20)
10*	10,380	5	30	35 (30)	40 (30)
8	16,510	8	40	50	55
6	26,240	13	55	65	75
4	41,740	21	70	85	95
3	52,620	27	85	100	110
2	66,360	34	95	115	130
1	83,690	42	110	130	150
1/0	105,600	54	125	150	170
2/0	133,100	67	145	175	195
3/0	167,800	85	165	200	225
4/0	211,600	107	195	230	260
250	250,000	127	215	255	290
300	300,000	152	240	285	320
350	350,000	177	260	310	350
400	400,000	203	280	335	380
500	500,000	253	320	380	430
600	600,000	304	355	420	475
700	700,000	355	385	460	520
750	750,000	380	400	475	535
800	800,000	405	410	490	555
900	900,000	456	435	520	585
1000	1,000,000	507	455	545	615

IEC mm2	Area Circ Mils	Amps Free Air, 70C PVC	Amps in wall, 70C PVC
0.5	987	--	--
0.75	1,481	--	--
1.5	2,961	17.5	13
2.5	4,935	24	17.5
4	7,896	32	23
6	11,844	41	29
10	19,740	57	39
16	31,584	76	52
25	49,350	96	68
35	69,090	119	83
50	98,700	144	99
70	138,180	184	125
95	187,530	223	150
120	236,880	259	172
150	296,100	299	196
185	365,190	341	223
240	473,760	403	261
300	592,200	464	298

(table A.52.4 of IEC 60364-5-52)
Fig. G20, http://tinyurl.com/2u3pzqd

* Maximum overcurrent protection for # 14 = 15A, # 12 = 20A, # 10 = 30A

Figure 66. IEC 60228 Wire Size Charts

Low Voltage Fuses

Another consideration that electrical systems must have involved fail safes built in place to prevent injury or damage. One effective solution is to place inline fuses for certain systems, cutting off power when power is exceeded [25].

$$\text{Catalog Fuse Rating} = \frac{\text{Normal Operating Current}}{0.75}$$

- or -

$$\frac{2.25 \text{ Amperes}}{0.75} = 3 \text{ Amp Fuse (at 25°C)}$$

Figure 67. Common Fuse Selection Calculation.

iii. Battery Standards

We live in times where we use many electronic devices that require battery use. Diverse technologies are utilized to power electric and electronic devices. Given the wide range of actuators and electronics which go into a robot, choosing the right battery may not be an easy task. There are a lot of battery types in the market, which if not handled properly and used upon their specifications can cause damage. The most commonly used batteries in robots are lithium-ion (LI-Ion), lithium polymer (LiPo) and Nickel Metal Hydride (NiMh).

Li-ion battery

A lithium-ion or Li-ion battery is a type of rechargeable battery which uses the reversible reduction of lithium ions to store energy. It is the predominant battery type used in portable consumer electronics and electric vehicles. It also sees significant use for grid-scale energy storage and military and aerospace applications. Compared to other rechargeable battery technologies, Li-ion batteries have high energy densities, low self-discharge, and no memory effect (although a small memory effect reported in LFP cells has been traced to poorly made cells).

Nickel-Metal Hydride Batteries (Ni-MH)

Many battery applications are well suited to be powered by NiMH rechargeable batteries. In general, devices that require large amounts of energy and are used frequently are well matched to the performance characteristics of NiMH batteries. Examples of these devices would include digital cameras, GPS units, and MP3 players.

Lithium Polymer Batteries (Li-Po)

These are becoming the most popular type of batteries for use in robotics because of their lightweight, high discharge rates and relatively good capacity, except the voltage ratings are available in increments of 3.7 V.

LIPO Design Parameters

When picking out a LiPo battery it is important to understand the different rating codes on a LiPo battery.

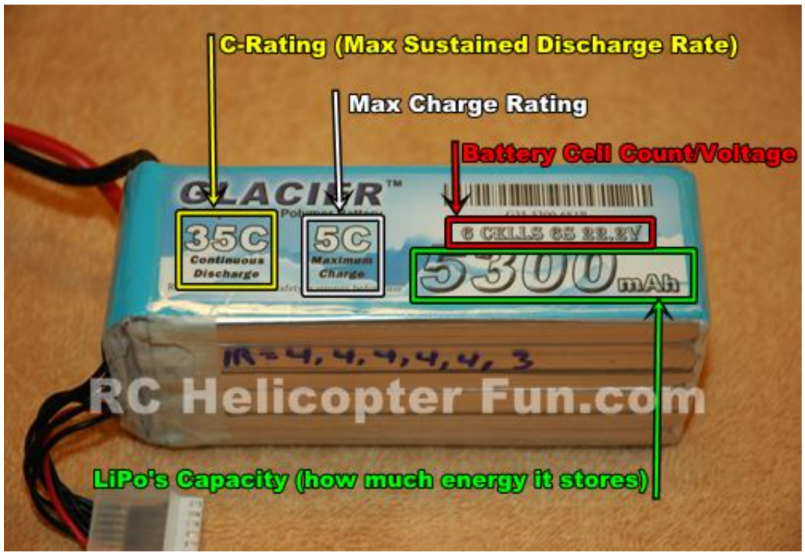


Figure 68. LiPo Battery Code [26].

Discharge Rate

The discharge rate of a LiPo battery will be denoted as, "**xx-C**" where "**xx**" is the multiplied value of the capacity of the pack. In **Figure 7** "35C" means it can discharge 35 times its capacity safely [26].

Max Charge Rating

The charge rate of a LiPo battery will be denoted as, "**x-C**" where x is the multiplied value of the capacity of the pack. In **Figure 7** "5C" means it can charge at 5 times it's capacity safely [26].

Battery Cell Count

The cell rating of a LiPo battery will be denoted as, "**X-S**" Where "**X"** Is the number of cells in the LiPo battery pack. In **Figure 7** "6S" means it has 6 cells in the battery pack [26].

Capacity

Capacity in LiPo batteries are measured in milliamp-hours [26].

LIPO Discharge Voltage Drop

As LiPo batteries discharge, the battery cell voltage will drop. The correlation between capacity and cell voltage is listed below in Table 17.

Table 17. LiPo Battery Drop Off [27].

Capacity %	1S	2S	3S	4S	5S	6S
100	4.2	8.4	12.6	16.8	21	25.2
95	4.15	8.3	12.45	16.6	20.75	24.9
90	4.11	8.22	12.33	16.45	20.56	24.67
85	4.08	8.16	12.25	16.33	20.41	24.49
80	4.02	8.05	12.07	16.09	20.11	24.14
75	3.98	7.97	11.95	15.93	19.92	23.9
70	3.95	7.91	11.86	15.81	19.77	23.72
65	3.91	7.83	11.74	15.66	19.57	23.48
60	3.87	7.75	11.62	15.5	19.37	23.25
55	3.85	7.71	11.56	15.42	19.27	23.13
50	3.84	7.67	11.51	15.34	19.18	23.01
45	3.82	7.63	11.45	15.26	19.08	22.89
40	3.8	7.59	11.39	15.18	18.98	22.77
35	3.79	7.57	11.36	15.14	18.93	22.72
30	3.77	7.53	11.3	15.06	18.83	22.6
25	3.75	7.49	11.24	14.99	18.73	22.48
20	3.73	7.45	11.18	14.91	18.63	22.36
15	3.71	7.41	11.12	14.83	18.54	22.24
10	3.69	7.37	11.06	14.75	18.44	22.12
5	3.61	7.22	10.83	14.43	18.04	21.65
0	3.27	6.55	9.82	13.09	16.37	19.64

iv. Gear Standards

One of the main mechanical elements is gears which are known as the transmission components of the robot and are responsible for increasing or decreasing speed, amplifying torque, changing the direction of rotation, and so on. With a structure that meshes with each other through teeth, gears used to transmit rotational motion from one shaft to another are mounted or linked to other parts through shafts or bases. Depending on the application needs, suitable gears are selected and assembled in a variety of models such as planetary gear mechanisms or rack & pinion gears.

In robotics, gears are used to transmit rotational forces between axles and are often used as the speed reducer of control mechanisms. They can change speed and direction. Gears may need to be treated depending on the application, such as grinding the tooth surface where high precision and fineness are required. There are two types of gear that are most commonly used in robots: spur gears and helical gears [28]. Accordingly, there are technical standards for gears established by development regulatory bodies that organize the rules, principles, and definitions for the manufacture, assembly, and inspection of gears. Depending on different regions or countries, these standards are also slightly different. In the United States, we have the AISI/AGMA standards for gears developed by the American Gear Manufacturers Association (AGMA). These standards may differ from equipment standards in Germany, the United Kingdom, or Japan, which have their own associations. However, the standard that is regularly revised, updated, and widely applied the most is the ISO standard which belongs to the International Organization for Standardization [29].

With different standards, an evaluation and comparison table are needed to provide suitable solutions for engineers who are working on the gear sets. Typically, a paper called "A Comparison of Current AGMA, ISO and API Gear Rating Methods" by John M. Rinaldo (a member of AGMA) makes it easy to understand the choices and the impact the choices have on gearbox design [30].

The gear standards necessary and practical for this autonomous climbing robot project are the gear ratio standards, load capacity standards, and pitting resistance standards. Below are some documents about the gear standards that are regulated by ISO and AGMA.

ISO 6336

ISO 6336 is a series of topics in Calculation of Load Capacity of Spur and Helical Gears. Each part focuses on different issues such as surface durability (pitting) (ISO 6336-2:2019) [31], tooth bending strength (ISO 6336-3:2019) [32], service life under variable load (ISO/TS 6336-6:2019) [33], etc.

ANSI/AGMA 2001--D04

ANSI/AGMA 2001-D04 is a standard about Fundamental Rating Factors and Calculation Methods for Involute Spur and Helical Gear Teeth. This standard specifies a method for assessing the pitting resistance and bending strength of spur and helical gear pairs. Factors affecting equipment survival and the calculation methods provided are also discussed [34].

b. Safety

The main concern regarding safety is that the robot will fall while climbing by veering off to the side and off the beam or other magnetic surface. There would likely need to be an agreement with the consumer that they understand not to stand below the robot while it climbs as a safety precaution, potentially along with some new additions to the robot to increase safety. It is also worth considering the fact that this technology is replacing the need for humans to climb or be taken by machines up to higher elevations to directly inspect structures, so any amount of risk is relative to that alternative.

As for the machine itself, we attempted to keep the wiring and movement parts of the robot as safe as feasible. The wiring utilizes sinches that prevent any contact with exposed wire. We also place the battery at the center of the machine, secured strongly to prevent it from coming loose, and the vast majority of components are fastened using nylon nuts to prevent them from coming loose over time and falling apart due to vibrations. If this type of design were to be used on a wider scale, the possibility of creating machinery to produce more custom parts that fit together more securely would open up, further reducing this concern. The only concern regarding environmental impact in this design and its manufacturing is the battery, which is not easily disposable.

c. Ethical Considerations

As mentioned, the battery in the robot's design is not an easily disposable component, and generally the purchasing of component material from ethical sources is a concern. If robots like this were to be produced on a wider scale, work would need to be done to ensure the ethical sourcing of material components.

For the creation of the CAD models, the use of GrabCAD models was utilized in order to help in the organizing of the electrical components in CATIA. Additionally, initial brainstorming was impacted largely by research done on YouTube, although the only aspects of the final design impacted by this are the magnets in the wheels and the differential drive, which are both common in autonomous climbing machines.

d. Economic Factors

When deciding how to purchase parts we would need multiple instances of, such as our fasteners, options for buying in bulk were leaned towards for the increased cost effectiveness. When needing to produce custom parts, we utilized the 3D printing available to us as a cheap, effective, and quick manufacturing option. If the robot were to be mass produced, the battery and magnets we used in the design may need to be reconsidered due to sustainability concerns.

If robots like the one in this project were to be used in place of humans in infrastructure inspection on a broader scale, it would provide the companies with a quicker, cheaper, and safer way to perform this task. It may decrease the number of jobs available in this regard, but there would still need to be people hired by these companies to deploy and transport these robots, so the jobs wouldn't disappear completely. On top of this, the decreased exposure to harmful environments might help people to accept the automation of these tasks.

e. Reliability

If this robot were to be widely produced and distributed, it would need to have a relatively low rate of failure. If there are any humans, other living organisms, or fragile objects below the robot while it is climbing the consequences of failure could be extreme. If the robot were to become detached from the beam and fall while a human was underneath it, it could cause serious injury.

f. Aesthetics

The current model is functional but may need a few alterations to make it more desirable as a product. While it did not make an appearance in the final model, there was a design concept to make the robot more aesthetically pleasing while also providing protection to the motors and electrical components. The idea was to create an outer casing for the robot resembling the body of a bee, an aesthetic that would be helped by the two antenna-like arms at the front of the robot. Likening the robot to such a small creature would be appropriate considering the relatively small size of this robot to similar devices in this field of study, as well as its maneuverability.

g. Potential Customers

Given that it is intended to inspect infrastructure, the use of this robot would likely be fairly professional, by companies looking to inspect the structural integrity of buildings, bridges, and the like. The most important aspect of the customer's interaction with the robot would be ease of use. In this regard, the fact that a system for vision and distance sensing is already in place will help to enable this machine in a wider variety of applications. The light compact nature of the robot allows it to be conveniently carried by a human while not in use, allowing for easier transport and deployment.

h. Societal and Global Impact

The concept of an autonomous robot climbing beams in order to inspect infrastructure is one that has been explored a fair deal, but this machine does add to that conversation. This machine is relatively small and has a good deal of maneuverability compared to machines from similar projects. This allows it to potentially be used in applications that are more cramped, crowded, or otherwise provide less area for the robot to maneuver. The compact and lightweight design of the robot also allows for easier transportation and placement of the machine.

VIII. CONCLUSIONS AND RECOMMENDATIONS

1. Comment on the Overall Solution to the Project.

The autonomous climbing robot is designed to meet customer requirements and intended use. In phase one, the research on material, design, and market is the priority issue. Assessments of customer requirements and engineering specifications to meet those needs are discussed and agreed upon. On the other hand, the necessary engineering standards in terms of mechanical, electrical, and computer/software are researched and organized.

In phase two, all the initial ideas and sketches are gathered and analyzed for pros and cons. The team developed the technical solution for the design. The ratings and options for each solution and component are discussed by the Pugh Matrix. From there, the team concluded by preliminary cost summaries.

In phase three, the initial design of the robot was a good combination of the sketches and ideas of all members. Initial kinematic analysis was done with an estimated weight to find the suitable model of motors. The official real model deployed with the design of the bottom frame which is manufactured by 3D-printing and some initially ordered components. Initial components were assembled and tested with the oscilloscopes and function generators borrowed from the school. The programming language was chosen. The coding was set up and run to make sure there is no error issue.

In phase four, the full model was assembled and tested. The design has been revised and improved step by step in parallel with testing. Design changes and additions are decided based on the test results to provide the most efficient solution possible. Kinematic analysis was performed for every essential movement of the robot. The CAD model was assembled and executed in the CAD simulation. FEA analysis was done based on the CAD model.

2. Comment on the Project

The assignment of this project presented many challenges that applied equally to all the different subdivisions of engineering. On the mechanical side, strict requirements had to be determined and met to allow the robot to be light enough to travel up and down the beam safely and efficiently while also being robust enough to withstand the full development process and accomplish the end goal.

The electrical work was also a meticulous task because of the need to implement a design that provided sufficient power to the system while also complying with the mechanical design specifications. This robot utilized four different motors, motor drivers, buck converters, two controllers, various sensors, and a camera which all had to be wired securely and safely while all fitting inside a 1 square foot area.

The software implementation was the biggest hurdle that had to be overcome. Utilizing the Jetson Nano meant that Linux was required to configure and run various scripts through ROS

and the NVIDIA object detection model. Linux was not something that anyone in the group had experience using so there was a big learning curve when initially working with ROS and the Jetson Nano. Programming with the much more familiar Arduino Mega also presented its own challenges though. Communication protocol had to be configured between the Arduino Mega and Jetson Nano so they could work together to control the motors and ultimately guide the robot throughout the task. PID controls also had to be implemented and continuously tweaked to allow the robot to adjust its position appropriately.

If given the opportunity to start again from scratch, there are definitely a few things we would approach differently for this project. One major aspect that our group got hung up on was the level of autonomy required for this robot. The idea to implement such a robust vision system was to create a robot that could find the beam in the lab no matter how the robot was oriented at the start or how far from the beam it was at the start. As the semester progressed, we realized this level of autonomy was not required to meet the expectations of this robot and a much simpler and cheaper vision system would have sufficed. This would have led to much quicker advancement in the development of the other subsystems required in the robot.

IX. REFERENCES

[1] *Amazon.com: Dingguanghe-Cup Ball Casters 5/10/20pcs Steel Ball Swivel ...* https://www.amazon.com/DINGGUANGHE-CUP-Universal-Machinery-Furniture-Industrial/dp/B08GFRCMQP.

[2] L. Trax. *Wheels vs Tracks: Advantages and Disadvantages.,* 1 Mar 2018. [Online]. Available: https://litetrax.com/wheels-vs-tracks-advantages-disadvantages/#:~:text=The%20traction%20is%20greater%20if,more%20suited%20to%20soft%20surfaces.. [Accessed 29 Jan 2023].

[3] Pye, Andy, and About Andy PyeAndy Pye is a technologist. "The Basics of Acrylonitrile Butadiene Styrene (ABS)." *Prospector Knowledge Center*, 17 Aug. 2022, https://knowledge.ulprospector.com/11655/pe-the-basics-of-acrylonitrile-butadiene-styrene-abs/.

[4] Britannica, The Editors of Encyclopaedia. "*magnetic field*". Encyclopedia Britannica, 5 Jan. 2023, https://www.britannica.com/science/magnetic-field. Accessed 5 February 2023.

[5] Williams, M. (2016, May 17). *What is air resistance*? Universe Today. Retrieved February 5, 2023, from https://www.universetoday.com/73315/what-is-air-resistance/.

[6] Edge, Engineers. "Coefficient of Friction Equation and Table Chart." *Engineers Edge - Engineering, Design and Manufacturing Solutions*, https://www.engineersedge.com/coeffients_of_friction.htm

[7] Guo, Qi, et al. "Experimental Study of Friction Resistance between Steel and Concrete in Prefabricated Composite Beam with High-Strength Frictional Bolt." *Advances in Materials Science and Engineering*, Hindawi, 19 Feb. 2020, https://www.hindawi.com/journals/amse/2020/1292513/.

[8] "Strength at Break (Tensile)." *Tensile Strength - Definition, Units, Formula & Test Methods*, https://omnexus.specialchem.com/polymer-properties/properties/strength-at-break-tensile.

[9] "BC64." *K&J Magnetics: BC64*, https://www.kjmagnetics.com/proddetail.asp?prod=BC64.

[10] "5mm Flange Coupling Optical Axis Support Fixed Seat Steel Rigid Flange Plate Shaft Connector." *Alexnld.com - Free Worldwide Shipping*, https://alexnld.com/product/5mm-flange-coupling-optical-axis-support-fixed-seat-steel-rigid-flange-plate-shaft-connector/.

[11] *Amazon.com: Dingguanghe-Cup Ball Casters 5/10/20pcs Steel Ball Swivel ...* https://www.amazon.com/DINGGUANGHE-CUP-Universal-Machinery-Furniture-Industrial/dp/B08GFRCMQP.

[12] *Amazon.com: Tattu 14.8V 1300mah Lipo Battery Pack 45c 4s with XT60 Plug ...* https://www.amazon.com/1300mAh-Battery-Freestyle-Airplane-Quadcopter/dp/B013I9RSEU.

[13] "Metal DC Geared Motor w/Encoder - 6V 210rpm 10kg.Cm." *DFRobot*, https://www.dfrobot.com/product-1617.html.

[14] "Buck Converter XL4015 - 5A (DC-DC)." *Partsbuilt 3D*, https://www.partsbuilt.com/buck-converter-xl4015-5a-dc-dc/.

[15] *Amazon.com: Hiletgo BTS7960 43A High Power Motor Driver Module/Smart ...* https://www.amazon.com/HiLetgo-BTS7960-Driver-Arduino-Current/dp/B00WSN98DC.

[16] SAVAGE, ARTHUR DEKKER. "DP." *Amazon*, KRILL PRESS, 2016, https://www.amazon.com/dp/B0BNQRV7SM?ref=ppx_yo2ov_dt_b_product_details&th=1.

[17] iGAM, "4 key differences between ABB, Kuka, and Fanuc Robots," *4 Key Differences Between ABB, KUKA, and FANUC Robots*. [Online]. Available: https://blog.igam.com/4-key-differences-between-abb-kuka-and-fanuc-robots..

[18] "Robot operating system," *ROS*. [Online]. Available: https://www.ros.org/.

[19] "The Orocos toolchain," *The Orocos Toolchain by orocos*. [Online]. Available: https://orocos.github.io/.

[20] "Getting set up," *YARP*. [Online]. Available: https://www.yarp.it/latest/index.html.

[21]"Unified architecture," *OPC Foundation*, 26-Sep-2019. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[22] "To float or not to float? analysis of a floating vs. grounded output ..." [Online]. Available: https://aptsources.com/wp-content/uploads/pdfs/Floating-Output.pdf.

[23] E. Technology, "Electrical Wiring Color Codes for AC & DC - NEC & IEC," *ELECTRICAL TECHNOLOGY*, 29-Sep-2022. [Online]. Available: https://www.electricaltechnology.org/2020/07/electrical-wiring-color-codes-nec-iec.html#dc-wiring-color-codes-in-other-countries.

[24] "MetaTek," *reference:awg-iec-wire [MetaTek]*. [Online]. Available:

[25] “Fuseology Application Guide - Littelfuse.” [Online]. Available: https://www.littelfuse.com/~/media/electronics/application_guides/littelfuse_fuseology_application_guide.pdf.pdf.

[26] J. Salt, “RC LiPo Batteries How to get the most life & fun out of them!” Oct-2022 Available: https://www.rchelicopterfun.com/lipo-batteries.html.

[27] Lipo Voltage Chart: Show the Relationship of Voltage and Capacity (2018) ampow. Available at: https://blog.ampow.com/lipo-voltage-chart/.

[28] K. S. Gears, "GEARS FOR ROBOTICS," [Online]. Available: https://khkgears.net/new/gears-for-robotics.html.

[29] B. Dengel, "What is a standard anyway?," 15 Mar 2020. [Online]. Available: https://gearsolutions.com/departments/tooth-tips/what-is-a-standard-anyway/.

[30] J. M. Rinaldo, "A Comparison of Current AGMA, ISO and API Gear Rating Methods," 1 July 2018. [Online]. Available: https://www.geartechnology.com/ext/resources/issues/0718x/gear-rating.pdf.

[31] ISO, "Calculation of load capacity of spur and helical gears — Part 2: Calculation of surface durability (pitting)," 11 2019. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:6336:-2:ed-3:v2:en.

[32] ISO, "Calculation of load capacity of spur and helical gears — Part 3: Calculation of tooth bending strength," 11 2019. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:6336:-3:ed-3:v2:en.

[33] ISO, "Calculation of load capacity of spur and helical gears — Part 6: Calculation of service life under variable load," 11 2019. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:6336:-6:ed-2:v1:en.

[34] AGMA, "Fundamental Rating Factors and Calculation Methods for Involute Spur and Helical Gear Teeth," 7 Jun 2005. [Online]. Available: https://wp.kntu.ac.ir/asgari/AGMA%202001-D04.pdf.

X. APPENDICES

Appendix A

1. Design Proposal Outline

The Senior Design final reports must explicitly follow the outline below.

Additional material may be added to appropriate sections, but all of the sections below, and the content indicated in each section, must be presented in the order given.

Title page

a. Title of the project

b. Date – Semester and year

c. Group number

d. Names of all group members - next to each name list the student's major and section(s) of the report for which the student was primarily responsible, clearly label the ME and ECE lead members. Also list the primary project role for each student.

e. Names of instructor(s), course number

Abstract

A very brief overview of the project's purpose and major results. May appear on the title page or at the top of the next page.

Table of contents

At a minimum must contain the headings of this outline – that is, everything in **bold** - as distinct sections of the report, with appropriate heading levels. Note that the title page and table of contents never appear in the table of contents.

1. Introduction

a. Description of the project

b. Brief description of the design solution, with illustrations

2. Design specifications

a. Detailed specifications for the design. Include overall specifications for the design as well as for each subsystem and component. These are the specifications that you were seeking to meet with your design solution and should refer to both the project description and any applicable professional standards. In particular, pay close attention to multidisciplinary specifications, that is, specifications that involve more than one field. For example, the torque and speed of a motor required in order for a particular motion to take place (mechanical) determine the power requirements to drive that motor (electrical) and the control requirements to make sure it all happens properly (computer). Refer to the design description and your design proposal and document any changes in the design specifications that were necessary during the course of the project, in chronological order.

3. Design overview

a. A non-technical overview of the complete design solution, complete with illustrations and descriptions of the function of the overall system and each subsystem.

4. Mechanical Subsystem

a. A detailed description of the mechanical subsystem, including but not limited to initial sketches and concepts, theory and hand calculations (typed), complete FEA of each component, assembly drawings and the results of motion analysis. Major decisions and revisions to the mechanical subsystem are to be presented in chronological order and must be accompanied by justifications and rationale for each change in decision or Pugh matrix format. Discuss how closely your simulations matched the performance of the physical mechanical subsystem. Include a bill of materials and costs for each mechanical component. Discuss any challenges or hurdles encountered with integration of the mechanical subsystem to the rest of the design solution.

b. Detailed engineering drawings of each component that was designed and built, as well as data sheets and/or specifications for any mechanical component that was purchased, are to be included in the appendices.

5. Electrical/Electronic Subsystem

a. A detailed description of the electrical and/or electronic subsystem, including but not limited to initial concepts, theory and hand calculations (typed), complete simulation of each electrical component, circuit diagrams and analyses of the electrical subsystem. Major decisions and revisions to the electrical subsystem are to be presented in chronological order and must be accompanied by justifications and rationale for each change in decision or Pugh matrix format. Discuss how closely your simulations matched the performance of the electrical subsystem. Include a bill of materials and costs for each electrical component. Discuss any challenges or hurdles encountered with integration of the electrical subsystem to the rest of the design solution.

b. Detailed circuit schematics of each component that was designed and built, as well as data sheets and/or specifications for any electrical components that were purchased, are to be included in the appendices.

6. Computer/Software Subsystem

a. A detailed description of the computer and/or software subsystem, including but not limited to initial concepts and theory employed, complete state machine diagrams, flow charts and block diagrams that describe the function of each portion of the computer code. Major decisions and revisions to the computer subsystem are to be presented in chronological order and must be accompanied by justifications and rationale for each change in decision or Pugh matrix format. Discuss the performance of the computer subsystem, in particular the performance of sensors and actuators, as compared to your simulations and/or manufacturer's specifications. Include a bill of materials and costs for the computer subsystem. Discuss any challenges or hurdles encountered with integration of the computer subsystem to the rest of the design solution.

b. Complete code listings, clearly documented as either original and/or reused/altered code, must be included in the appendices.

7. Discussion

a. **Technical Discussion** of the complete design details, focusing on how the design solution works and how well it addresses the design specifications of Section 4 above. Refer to the design specifications and the project proposal in the discussion, indicate which specifications were met and which were not, and how and why the final design differs from the solution proposed at the beginning of the semester. Discuss overall challenges or hurdles encountered with the physical integration of the various subsystems. An overall bill of materials and costs for the project are to be included.

b. **Professional and Societal Context** - explicitly address each of these topics:

i. **Engineering Standards** - Describe the process in which you searched for applicable professional engineering standards and applied them in your design. Even if you could not access the standards due to restricted content or cost, you must at least identify the government, industry or professional organization standards that govern the development, analysis, distribution, manufacture and/or sales of the technology involved in your project.

ii. **Safety** – Discuss safety concerns that you had to address in the development and execution of your design. Demonstrate that you have the understanding that the vast majority of engineering products are used by those without technical or engineering knowledge to design and produce them, and that engineers have the ultimate professional responsibility to keep end users safe (using the broadest definition of the word "safe") by explicitly accounting for safety in design. What sort of constraints must be accounted for if the technology used in your design was used on a large scale? Are there significant environmental impacts surrounding the manufacture, use or disposal of your design?

iii. **Ethical Considerations** - Discuss your research into the various technologies that you have used in your design, and in particular any documented or potential safety or environmental issues that exist. Have you borrowed ideas from existing products or patents that may need to be licensed if the technology you developed was adopted in the marketplace? Have you borrowed ideas from other design groups and properly given credit to those ideas?

iv. **Economic factors** - Discuss any economic factors accounted for in your design. Talk about the sustainability and manufacturability of your design. How much would it cost to implement the technology and what sort of jobs would it impact (consider improvements to safety as well as potentially displacing jobs)? What sort of impact would this have on the acceptance of such technology?

v. **Reliability** - If this technology was to be used in mass applications, what sort of reliability and failure rates would be acceptable? What safety issues would arise if the technology you developed failed?

vi. **Aesthetics** – How close have you come to producing a prototype of a marketable, viable product? What sort of additional devices or accessories could be used to make this technology as aesthetically attractive as possible while retaining its essential functionality?

vii. **Potential customers** – Describe how you have taken into account how people live and work (culturally, economically, etc) and have responsibly, consciously and professionally developed a solution that accommodates the end user? Think broadly about this question and consider applications of the general technologies you have used in your project; do not focus on the specific application you have designed.

viii. **Societal and global impact** - What would be the potential benefits to society for the applications that you developed? What would be the potential drawbacks? Are there any global ramifications to your design or the technology it uses?

c. **Information Literacy** - Describe how, in the course of your project, you learned new things on your own, found and assessed sources of information, and the overall need to seek out and use new ideas and information to solve engineering problems, all in a responsible and professional manner.

8. Conclusions and recommendations

a. Comment on the overall solution to the project.

b. Comment on the project assigned and the process by which you solved the challenge, particularly issues with integrating the various subsystems of the physical model and including any recommendations for change or for future projects.

9. References

a. Must be in proper ASME or IEEE format – list here the sources of everything that you have used in the process of your design solution that was not original to you. **Everything that is not yours – ideas, equations, concepts, photos, graphic, everything – must be documented and properly referenced**. This is major evidence of the ethical and professional manner in which you carried out your design project. All references must be properly cited in the text, and care must be taken to ensure that all information comes from reliable and trustworthy sources.

10. Appendices

a. Everything that interrupts the flow of the report belongs in the appendices. Lengthy mathematical derivations, component mechanical drawings, detailed electrical and electronic circuits, listings of computer code, product specifications and data sheets, etc. are some of the items that typically are included in appendices.

Grading Rubric

POINTS	3	4	5
Mechanics	Some sections missing and/or incomplete, multiple typos and/or use of non-standard English	All sections included, some sections incomplete, some typos and/or use of non-standard English	All sections included and complete, free of typos, standard English
Introduction	Unclear, wordy, not descriptive, not illustrated	Somewhat clear, concise, and descriptive, poorly illustrated	Clear, concise, descriptive, well-illustrated
Design Specifications	No clear list of requirements or goals for the project	List of requirements, lack detail and specificity	Clear list of specific requirements, goals and options for the project
Design Overview	Unclear, wordy, not descriptive, not illustrated	Somewhat clear, concise, and descriptive, poorly illustrated	Clear, concise, descriptive, well-illustrated
Subsystem and System Discussions	Unclear, wordy, does not convey understanding of concepts and challenges, decisions not documented with Pugh matrices	Somewhat clear and/or concise, sometimes unclear on concepts and challenges, some major decisions documented with Pugh matrices	Clear, concise, clearly demonstrates understanding of concepts and challenges, all major decisions documented with Pugh matrices
Professional and Societal Context	Unclear, wordy, superficial, does not convey understanding of concepts and responsibilities	Somewhat clear and/or concise, sometimes unclear on concepts and responsibilities	Clear, concise and thoughtfully demonstrates understanding of concepts and responsibilities
Conclusions and Recommendations	Missing or superficial assessment of project, goals and/or deliverables, no useful conclusions and/or recommendations	Assessment of project, goals and deliverables, somewhat useful conclusions and recommendations	Thoughtful assessment of project, goals and deliverables, useful conclusions and recommendations
References	Sources missing and/or not in acceptable format	List of sources incomplete and/or not properly formatted	List of sources complete and properly formatted
Appendices	Appendices missing and/or used inappropriately	Some appropriate information in appendices but some remain in text	All appropriate information contained in appendices

2. Product Development Process

Outline and tools adapted from "*Product Development and Production Handbook, First Edition*," Steven W. Trimble and Abdelrahman N. Shuaib, Cognella Academic Publishing, 2022

Phase 1 – Definition of the Design Project

The purpose of this phase of a design project is to completely define the project goals and functions, establish measurable engineering requirements, and identify and plan for all solution and problem-solving constraints such as schedule and budget. These steps must be completed before any conceptual or preliminary designs are explored. Changes to the design requirements after this phase is complete require that the project definition be revisited and revised.

- Project team is formed
- Project needs and functions are defined and prioritized through Customer interaction
- Develop formal project statement
- Convert customer needs to defined and measurable engineering design requirements with Customer input and approval – this list defines the design project and is the basis for measuring the progress and successful completion of the project
- Identify high priority design goals and requirements for optimization
- Search for all applicable regulatory, government, industry, and/or professional codes and/or standards that may apply to potential design solutions. Search for applicable patents that may need to be accommodated with potential design solutions
- Complete project plan including task list, schedule (Gantt chart), and resource budget with Customer input

Deliverables:

- Customer Needs and Functions Worksheet
- Engineering Design Requirements and Validation Worksheet
- Project Task List Worksheet
- Preliminary Project Budget Worksheet
- Documentation of applicable codes, standards, and/or patents • Project schedule Gantt Chart

Phase 2 – Conceptual Design

The purpose of this phase is to develop (usually with the customer) several conceptual design solutions, to rank them in terms of their potential to meet or exceed all the design requirements and to begin to analyze the most promising one for further exploration and development. It may be necessary to revisit some of the Phase 1 tasks and deliverables to develop a promising conceptual design.

- Explore conceptual ideas with the Customer
- Explore the design space through research and brainstorming to identify additional candidate concepts
- Construct block diagrams or flowcharts for each conceptual design to identify the functions required to meet all the design requirements

- Filter the concepts down to the best two or three using decision matrices based on the design requirements. Analysis and proof of concept testing may also be needed
- Define most promising concept with sketches and estimates of all important physical parameters such as size, weight, cost, reliability, etc.
- Choose initial components, identify whether components will be purchased or designed and fabricated, develop initial Bills of Material (BOM)
- Update the Engineering Design Requirements and Validation Worksheet to begin to track the validation of each design requirement

Deliverables:

- Sketches and Schematics
- Decision Matrices
- Bills of Material for each subsystem and the complete system
- Updates of Phase 1 Deliverables

Senior Design Class Deliverable – Design Proposal

Phase 3 – Preliminary Design

The purpose of this stage is to develop the most promising conceptual idea into a robust design that has the potential to meet or exceed all the engineering design requirements.

- Analysis is required to prove that the design will satisfy the design requirements. Types of analysis include but are not limited to strength and deflection (FEA); microprocessor requirements; voltage and current, power; reliability; assembly; maintainability; safety (RMS) analysis; failure modes and effects (FMEA); design to cost (DTC) analysis; design for manufacturing and assembly (DFMA); component and material selection
- Simulations, tests, and prototypes must be documented through the Engineering Simulations, Tests and Prototypes procedure (below)
- Changes to the conceptual design(s) must be documented through using decision matrices based on the design requirements
- Key deliverables are dimensioned solid models, electrical schematics, software flow charts and pseudocode

Deliverables:

- Solid Models, electrical schematics, software flow charts and pseudocode
- Decision Matrices
- Results of analysis and preliminary testing
- Updates of Phase 2 Deliverables

Major Senior Design Class Deliverables – Mid-Term Design Review, Mid-Term Presentation Note that for most groups these deliverables will occur within Phase 3 or Phase 4.

Phase 4 – Detailed Design

The purpose of this design phase is to define a complete and robust design solution that satisfies all the design requirements, to validate the design through a functional prototype, and to deliver a set of plans and assembly procedures that is ready for pre-production prototyping.

- The Preliminary Design from Phase 3 is further analyzed and defined to develop the detailed package for production
- This process typically entails a series of simulations, physical tests and/or prototypes (see below) of increasing complexity and completeness, culminating in a complete and validated model of the entire design
- Results of each simulation, test and prototype are iterated back to the preliminary design phase and used to validate the detailed design of the production unit
- The deliverables of this phase are a complete specification for a production solution to the deign problem and a fully functioning prototype

Deliverables:

- Phase 3 Deliverables Updated
- Detailed Component, Assembly and System specifications
- Complete design prototype with full functionality

Major Senior Design Class Deliverables – Final Design Review, Final Presentation, Design Expo and Final Report

---------- Senior design projects typically end here ----------

For design projects result in products that are produced, marketed, distributed, sold, and supported after purchase, the design process continues as below.

Phase 5 – Production Fabrication, Assembly and Testing

The purpose of this phase is to build and test a pre-production prototype from the detailed design of Phase 4, with complete functionality and including all non-functional goals that were not included in the design prototype. Very little additional development should be required at this stage. If this pre-production prototype fails any of its development or functional tests, it must be reworked and modified until it can pass all validation tests. Any rework must be documented in the production drawings.

- Fabrication and Assembly
- Development Testing <-> Rework • Validation Testing • Deliverables:
  - Complete Component, Assembly and System specifications o Qualified Prototype
  - Production Drawings

Phase 6 – Production Unit and Support

This phase includes all post-design product production, support of the product once it is in the field, upgrades of the product design based on customer feedback on field units.

Deliverables:

- Production Units
- Support Resources

Engineering Simulations, Tests and Prototypes procedure

All engineering simulations, tests and prototypes are to be documented in a Test Report (see worksheet). Each simulation, test or prototype is itself small design project, with the same goals, engineering requirements, preliminary design, fabrication, testing, and documentation requirements as the main project. Note that there

are often many simulations, tests and prototypes for a given engineering project, at least one for every major subsystem and another for the complete system.

The main deliverable for any simulation, test or prototype is information to be fed back to the preliminary or detailed design.

Phase 1T – Define and Design

Sub-group of design team is identified and tasked with the simulation, test, or prototype. Document the information required from the simulation, test or prototype and the features of the product design that must be included in the simulation, test or prototype, accounting especially for budget and schedule constraints. Document the resources required for the test. Simulations (FEA. MATLAB, SimuLink, etc.) are often necessary to show that the physical tests or prototype units will meet the requirements of the test. The main deliverable is a simulation plan or an engineering design package and a set of plans and procedures describing how the simulation will be conducted or engineering prototype will be manufactured, developed, validated, and tested.

Deliverables:

- Documentation of who is responsible for the test and the resources required
- Solid model, engineering drawings, electrical schematics, software flowcharts for the test
- Simulation or physical test plan

Phase 2T – Build, Test and Report

A simulation is carried out, or a physical unit is fabricated, assembled, validated, and tested according to the design and test plan from Phase 1T. Documented changes may be made to the list of test requirements, prototype design package, constructed unit and/or test plan as necessary to achieve required information from the simulation, test or prototype. Results from the simulation, test or engineering prototype are documented and integrated into the preliminary or detailed design as appropriate.

Deliverables:

- Results of simulation, test, or prototype
- Documentation of how the results of the test will be integrated into the preliminary or detailed design

Appendix B

1. Pugh Matrix

Vision Hardware

Criteria	Weight	Pixy 2	ESP-32 CAMERA	RPIv 2	ARDUCAM	DFRobot FIT0701
Max Power Draw (W)	2	1	1	2	0	0
Cost	4	1	3	2	4	4
Still Frame Resolution	2	1	1	0	0	2
Frame rate (fps)	3	2	2	3	1	0
Size	1	0	1	1	1	0
Weight	2	0	1	2	1	0
MC Compatibility	4	2	2	4	2	4
Interface	4	4	0	4	4	4
Total	22	11	11	18	13	14

Vision controller Pugh Matrix

Criteria	Baseline	Raspberry Pi	Jetson
Difficulty	1	0	1
Accuracy	2	1	2
Speed	1	0	1
Functionality	3	1	3
Cost	1	2	0
Total		4	7

Detection Solution Pugh Matrix

Criteria	Baseline	Object Detection	Color Detection	Image Classification
Difficulty	1	-1	0	1
Accuracy	2	2	-2	0

Speed	1	-1	0	1
Functionality	2	2	1	-2
Precision	1	1	-1	0
Repeatable	2	2	-2	2
Total		**5**	**-4**	**3**

Robot's Rear Motion

Criteria	Baseline	Ball Universal Wheel	3D-printing wheel	Non-Swivel Caster Wheel
Speed	4	4	2	3
Weight	4	3	4	2
Cost	3	2	3	-1
Mobility	4	3	1	2
Simplicity	2	2	1	1
Power Efficiency	3	3	1	2
Traction	3	1	2	3
Aesthetics	1	1	0	1
Getting off the beam	4	-1	2	4
Total		18	16	17

Robot's Movement Methods

Criteria	Baseline	Magnetic Tank Tread	Magnetic Wheels
Speed	4	0	3
Weight	3	0	2
Cost	3	3	3
Maneuverability	2	0	2
Mobility	3	2	1
Materials	2	0	1
Simplicity	2	-1	2
Power Efficiency	4	1	0

Traction	3	3	0
Aesthetics	0	2	1
Ground Impact	0	2	1
Weight Growth Potential	2	2	0
Total	0	14	16

Vision Controllers

Criteria	Baseline	Raspberry Pi	Jetson
Difficulty	0	-1	0
Accuracy	0	0	1
Speed	0	0	1
Functionality	0	0	1
Cost	0	0	-1
Total	0	-1	2

Detection Solutions

Criteria	Baseline	Object Detection	Color Detection	Image Classification
Difficulty	0	-1	0	1
Accuracy	0	1	-1	0
Speed	0	-1	0	1
Functionality	0	2	1	-2
Precision	0	1	-1	0
Repeatable	0	1	-1	1
Total	0	3	-2	1

Microcontroller

Criteria	Baseline	Arduino Uno	Arduino Nano	Arduino Micro	ESP-32
Size	2	0	2	2	1

GPIO - Digital	2	-1	2	0	1
GPIO - Analog	1	-1	-1	0	1
Logic Level	1	1	1	1	0
Power In	1	0	0	0	-1
Power Draw	1	0	-1	2	1
Memory	3	2	1	0	3
Communication	1	1	1	1	1
Speed	1	0	0	0	1
Price	0	-1	0	0	1
Total	0	1	5	6	9

2. Gantt Chart

Beam Climbing Robot

Group 12

SIMPLE GANTT CHART by Vertex42.com

https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Start: Mon, 1/9/2023

Display Week: 8

Phase 1 - Definition of the Design Project	Assigned To:			
Customer Needs and Functions Worksheet		100%	1/9/23	1/19/23
Engineering Design Requirement and Validation Worksheet		100%	1/9/23	1/19/23
Project Task List Worksheet		100%	1/9/23	1/19/23
Preliminary Project Budget Worksheet		100%	1/9/23	1/19/23
Documentation of codes, standards, patents		100%	1/19/23	1/28/23
Project Schedule Gantt Chart		100%	1/19/23	1/25/23
Phase 2 - Conceptual Design				
Sketches and Schematics		100%	1/20/23	1/26/23
Decision Matrices		100%	1/22/23	1/26/23
Bill of Material for Each Subsystem and Complete		100%	1/26/23	1/29/23
Updating Phase 1 Deliverables		100%	1/25/23	1/27/23
Design Proposal		100%	1/24/23	1/30/23

Feb 27, 2023 | Mar 6, 2023 | Mar 13, 2023 | Mar 20, 2023 | Mar 27, 2023 | Apr 3, 2023 | Apr 10, 2023 | Apr 17, 2023

27 28 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

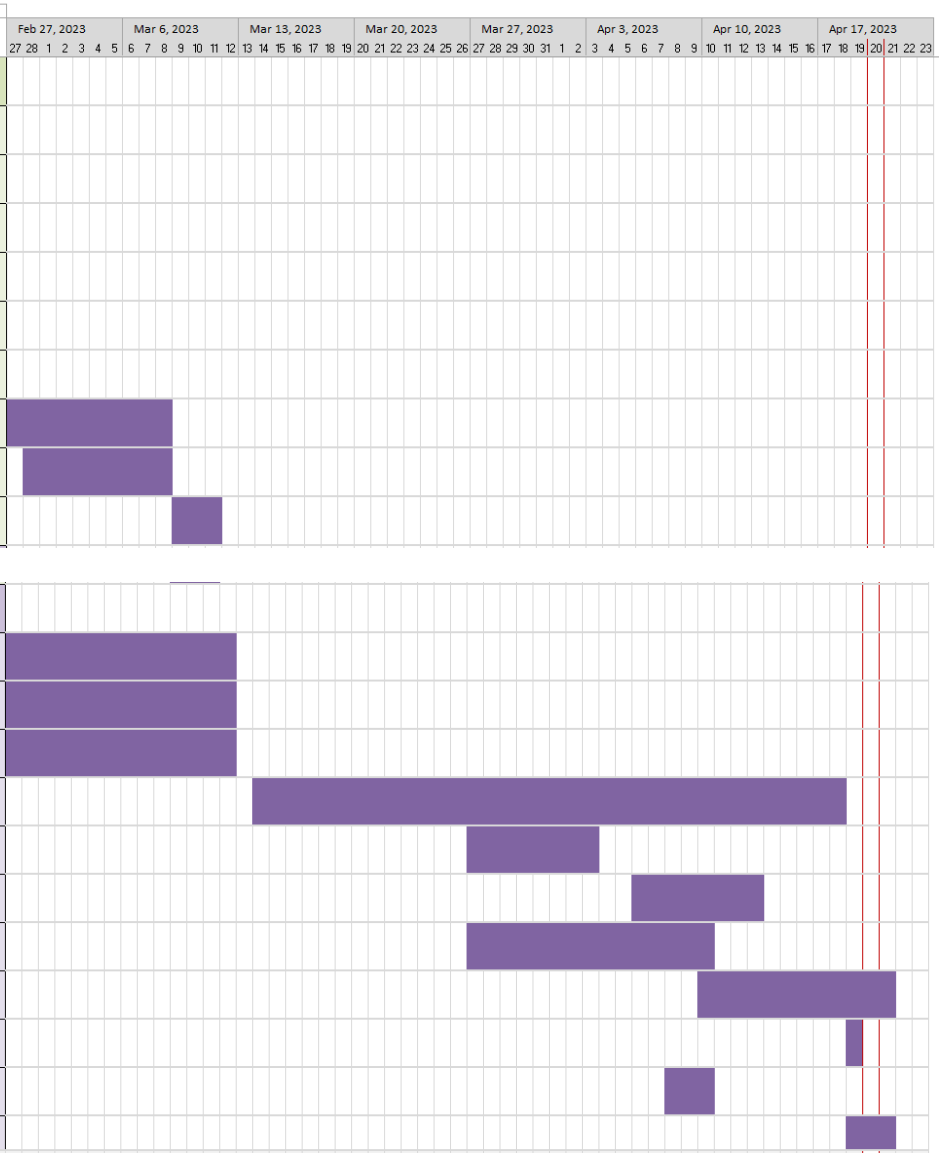
Beam Climbing Robot

Group 12

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

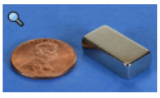
Project Start: Mon, 1/9/2023
Display Week: 8

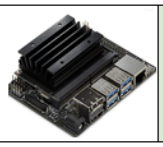
Phase 3 - Preliminary Design				
Solid Models			1/31/23	2/14/23
Electrical Models			1/31/23	2/14/23
Software Flowchart & Psuedocode			1/31/23	2/18/23
Decision Matrices			2/5/23	2/10/23
Preliminary Testing			2/7/23	2/21/23
Results of Analysis and Preliminary Testing			2/21/23	2/25/23
Midterm Design Review			2/26/23	3/8/23
Midterm Presentation			2/28/23	3/8/23
Updates of Phase 2 Deliverables			3/9/23	3/11/23
Phase 4 - Detailed Design				
Component Specs			2/19/23	3/12/23
Assembly Specs			2/19/23	3/12/23
System Specs			2/19/23	3/12/23
Complete Design Prototype			3/14/23	4/18/23
Final Design Review (prep)			3/27/23	4/3/23
Final Presentation			4/6/23	4/13/23
Final Report DRAFT			3/27/23	4/10/23
Final Report			4/10/23	4/21/23
Design Expo			4/19/23	4/19/23
Updates of Phase 3 Deliverables			4/8/23	4/10/23
Final Deliveries			4/19/23	4/21/23



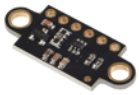
3. Detailed Budgets of Materials

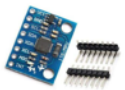
Final Build BOM								
Category	Quantity	Picture	Part	Model	Price	Items Used	Cost of item used	Notes
ORDER 1								
Electrical Sub-System	1		Battery	14V 4 Cell	$19.99	1	$19.99	
Electrical Sub-System	1		Charger	Charger	$33.99	1	$33.99	
Computer/Software Sub-System	4		Buck Converter	4 Buck Converters	$14.99	4	$14.99	
Computer/Software Sub-System	5		Driver	BTS7960 Motor Driver	$26.00	2	$10.40	
ORDER 1 Total = 100.66								
ORDER 2								
Electrical Sub-System	2		Motor	Geared DC Motor 210RPM 10Kg.cm	$39.80	2	$39.80	
ORDER 2 Total = 52.78								

ORDER 3								
Mechanical Sub-System	20		Magnets	BC64	$40.32	20	$40.32	
ORDER 3 Total = 51.75								
ORDER 4								
Computer/Software Sub-System	1		Camera	HD Pro Webcam C920	$64.00	1	$64.00	We had it on hand
ORDER 4 = 64								
ORDER 5								
Mechanical Sub-System	2		Rigid Flange Coupling	Motor Shaft Flange	$7.99	2	$7.99	
ORDER 5 Total = 8.47								

ORDER 6								
Computer/Software Sub-System	1		Compuer	Jetson Nano	$149.00	1	$149.00	
ORDER 6 Total = 157.94								
ORDER 7								
Computer/Software Sub-System	1		Module	Wifi Module	$22.94	1	$22.94	
ORDER 7 Total = 24.32								
ORDER 8								
Computer/Software Sub-System	4		Connectors	Wires	$10.59	1	$2.65	
ORDER 8 Total = 11.23								
ORDER 9								
Computer/Software Sub-System	100		Fuses		$9.99	2	$0.20	
ORDER 9 Total = 10.59								

ORDER 10								
Mechanical Sub-System	300		Bolts		$3.99	60	$0.80	Available item
ORDER 10 Total = 4.22								
ORDER 11								
Computer/Software Sub-System	1		Microcontroller	Intel Mega2560	$14.99	1	$14.99	Available item
ORDER 11 Total = 15.89								
ORDER 12								
Electrical Sub-System	3		Proto Board		$16.99	1	$5.66	
ORDER 12 Total = 17.08								
ORDER 13								
Mechanical Sub-System	100		Nylon Nuts		$5.99	20	$1.20	
ORDER 13 Total = 6.35								

ORDER 14								
Electrical Sub-System	26		JSTs		$45.99	1	$1.77	
ORDER 14 Total = 48.75								
ORDER 15								
Mechanical Sub-System	300		Zipties		$11.00	15	$0.55	
ORDER 15 Total = 11								
ORDER 16								
Electrical Sub-System	2		Servos	DS3225 25KG	$33.99	2	$33.99	
Computer/Software Sub-System	2		Distance Sensor	VL53L0X	$9.99	1	$5.00	
ORDER 16 Total = 46.62								

ORDER 17								
Computer/Software Sub-System	1		Gyro Sensor		$3.00	1	$3.00	We had it on hand
ORDER 17 Total = 3								
ORDER 18								
Mechanical Sub-System	1		Steel Ball Universal Wheel		$2.00	1	$2.00	We had it on hand
ORDER 18 Total = 2								
ORDER 19								
Electrical Sub-System	2		Micro Servo Motor	SG90 9g	$8.79	1	$4.40	
ORDER 19 Total = 9.32								

TOOLING COST								
Mechanical Sub-System	2	Wheel		5 Mag		7.03	$0.14	TPU
	1			9 Mag		13.35	$0.33	PLA
	1			9 Mag		13.35	$0.27	TPU
	2			9 Mag - Mod		11.95	$0.24	TPU
	2			10 Mag - Mod		14.71	$0.29	TPU
	2			10 Mag - Mod v2		14.67	$0.29	TPU
	1	Frame		Top Frame v1		119.82	$2.99	PLA
	1			Bottom v1		114.73	$2.29	PLA
	1			Top Frame v2		115.32	$2.30	PLA
	1			Bottom v2		112.76	$2.24	PLA
	1			Top Frame v3		113.23	$2.25	PLA
				TOTAL			13.63	
				Total	**659.61**	**Total on Robot**	**493.25**	

Appendix C

1. Detailed Electrical Circuit

2. Listings of Computer Code

ledon.launch

```
<?xml version="1.0"?>
<launch>
    <node pkg="rosserial_python" type="serial_node.py" name="serial">
        <param name="~port" value="/dev/ttyACM0"/>
        <param name="~baud" value="115200"/>
    </node>
    <node pkg="arduino" type="beam+bell.py" name="jetson">
    </node>
</launch>
```

```
#!/usr/bin/env python3
import jetson.inference
import jetson.utils
import rospy
from std_msgs.msg import Int8
from std_msgs.msg import Int32


cam = jetson.utils.videoSource("/dev/video0")
disp = jetson.utils.videoOutput("display://0")
net = jetson.inference.detectNet(argv=['--model=/home/jb/catkin_ws/src/jetson-inference/python/training/detection/ssd/models/beam-v2/ssd-mobilenet.onnx',
                                       '--labels=/home/jb/catkin_ws/src/jetson-inference/python/training/detection/ssd/models/beam-v2/labels.txt',
                                       '--input-blob=input_0','--output-cvg=scores', '--output-bbox=boxes'],
                                        threshold=0.75)
net2 = jetson.inference.detectNet(argv=['--model=/home/jb/catkin_ws/src/jetson-inference/python/training/detection/ssd/models/bell/ssd-mobilenet.onnx',
                                        '--labels=/home/jb/catkin_ws/src/jetson-inference/python/training/detection/ssd/models/bell/labels.txt',
                                        '--input-blob=input_0','--output-cvg=scores', '--output-bbox=boxes'],
                                         threshold=0.75)
```

```
#cap=cv2.VideoCapture("csi://0")
detect = 0
approach = 0
def outputCallback(data):
        global detect
        detect = data.data
def outputCallback2(data2):
        global detect
        approach = data2.data

while not rospy.is_shutdown():
    #while disp.IsStreaming():
    pub = rospy.Publisher('data',Int8,queue_size=10)
    sub = rospy.Subscriber('detect', Int32, outputCallback)
    sub2 = rospy.Subscriber('approach', Int32, outputCallback2)
    rospy.init_node('jetson', anonymous=True)

    rate=rospy.Rate(8)
```

```
38      if detect == 0 and approach == 0:
39          img=cam.Capture()
40          detections = net.Detect(img)
41          disp.Render(img)
42          disp.SetStatus("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
43          flag = 0
44
45          #Find object
46          for det in detections:
47              center_x = int(det.Left + det.Right) // 2
48              center_y = int(det.Top + det.Bottom) // 2
49              #jetson.utils.cudaDrawCircle(img, (center_x, center_y,), 5,(0,255,0,200))
50              flag = 1
51
52          if(flag == 1):
53              if center_x < img.width // 2:
54                  #print("Move Left")
55                  #print("Detect: %s", detect)
56                  rospy.loginfo(0)
57                  pub.publish(0)
58                  rate.sleep()
59              if abs(center_x - img.width // 2) <= 35:
60                  #print("Center")
61                  #print("Detect: %s", detect)
62                  rospy.loginfo(1)
63                  pub.publish(1)
64                  rate.sleep()
65              if center_x > img.width // 2:
66                  #print("Move Right")
67                  #print("Detect: %s", detect)
68                  rospy.loginfo(2)
69                  pub.publish(2)
70                  rate.sleep()
```

```
    if detect == 0 and approach == 1:
        img=cam.Capture()
        detections = net.Detect(img)
        disp.Render(img)
        disp.SetStatus("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
        flag = 0
        print("Approach: %s", approach)
    #Find object
    for det in detections:
        center_x = int(det.Left + det.Right) // 2
        center_y = int(det.Top + det.Bottom) // 2
        #jetson.utils.cudaDrawCircle(img, (center_x, center_y,), 5,(0,255,0,200))
        flag = 1

    if(flag == 1):
        if center_x < img.width // 2:
            #print("Move Left")
            #print("Detect: %s", detect)
            rospy.loginfo(0)
            pub.publish(0)
            rate.sleep()
        if abs(center_x - img.width // 2) <= 10:
            #print("Center")
            #print("Detect: %s", detect)
            rospy.loginfo(1)
            pub.publish(1)
            rate.sleep()
        if center_x > img.width // 2:
            #print("Move Right")
            #print("Detect: %s", detect)
            rospy.loginfo(2)
            pub.publish(2)
            rate.sleep()

        if(flag == 0):
            #print("No object")  #No object
            #print("Detect: %s", detect)
            rospy.loginfo(2)
            pub.publish(2)
            rate.sleep()

```

```
112
113     if detect == 1:
114         img2=cam.Capture()
115         detections2 = net2.Detect(img2)
116         disp.Render(img)
117         disp.SetStatus("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
118         flag = 0
119
120         #Find object
121         for det2 in detections2:
122             center_x = int(det2.Left + det2.Right) // 2
123             center_y = int(det2.Top + det2.Bottom) // 2
124             #jetson.utils.cudaDrawCircle(img2, (center_x, center_y,), 5,(0,255,0,200))
125             flag = 1
126
127         if(flag == 1):
128             if center_x < img2.width // 2:
129                 #print("Move Left")
130                 #print("Detect: %s", detect)
131                 rospy.loginfo(5)
132                 pub.publish(5)
133                 rate.sleep()
134             if center_x > img2.width // 2:
135                 #print("Move Right")
136                 #print("Detect: %s", detect)
137                 rospy.loginfo(4)
138                 pub.publish(4)
139                 rate.sleep()
140             if abs(center_x - img2.width // 2) <= 5:
141                 #print("Center")
142                 #print("Detect: %s", detect)
143                 rospy.loginfo(6)
144                 pub.publish(6)
145                 rate.sleep()
146         if(flag == 0):
147             #print("No object")  #No object
148             #print("Detect: %s", detect)
149             rospy.loginfo(2)
150             pub.publish(2)
151             rate.sleep()
152
153
154
```

```
318 currentMillis = millis();
319 if (currentMillis - previousMillis >= interval) { //takes RPM reading every 150 ms
320   previousMillis = currentMillis;
321   double LENCA_meas = LENCA - prevLENCA;
322   double RENCA_meas = RENCA - prevRENCA;
323   prevLENCA = LENCA;
324   prevRENCA = RENCA;
325   rpm_L = (double)((LENCA_meas / 374) * 400);
326   rpm_R = (double)((RENCA_meas / 374) * 400);
327   get_MPU_data(); //pulls pitch yaw roll values from gy 521
328 }
```

```
switch (Robot_Sequence) {
  case 0: //Wait for bootup to occur
    servoT.write(10);
    valueDetect = 0;
    approachValue = 0;
    visionState.data = valueDetect;
    approach.data = approachValue;
    if (s >= 0) {
      pub.publish(&visionState);
      pub2.publish(&approach);
      Robot_Sequence = 1;
    }
    break;
```

```
355       case 1:   //Find Beam
356         servoT.write(10);
357         if (s == 2) { // Move forward with PID if not center
358           digitalWrite(RMotorEN_1, HIGH);
359           digitalWrite(RMotorEN_2, HIGH);
360           digitalWrite(LMotorEN_1, HIGH);
361           digitalWrite(LMotorEN_2, HIGH);
362 
363           analogWrite(RMotorFWD, Rforward_SPEED);
364           analogWrite(LMotorFWD, Lforward_SPEED);
365         }
366         if (s == 0) { // Move forward with PID if not center
367           digitalWrite(RMotorEN_1, HIGH);
368           digitalWrite(RMotorEN_2, HIGH);
369           digitalWrite(LMotorEN_1, HIGH);
370           digitalWrite(LMotorEN_2, HIGH);
371 
372           analogWrite(RMotorFWD, Rforward_SPEED);
373           analogWrite(LMotorFWD, Lforward_SPEED);
374         }
375         if (s == 1) { // Stop if center
376           Robot_Sequence = 2;
377           digitalWrite(RMotorEN_1, LOW);
378           digitalWrite(RMotorEN_2, LOW);
379           digitalWrite(LMotorEN_1, LOW);
380           digitalWrite(LMotorEN_2, LOW);
381 
382           analogWrite(RMotorFWD, 0);
383           analogWrite(LMotorFWD, 0);
384           statepreviousMillis = currentMillis;
385           LENCAx = LENCA; // Record distance traveled
386           RENCAx = RENCA;
387         }
388         break;
```

```
389     case 2:   //Wait 2 Seconds
390       if (currentMillis - statepreviousMillis > 2000) {
391         Robot_Sequence = 3;
392         turnLENCA = LENCA;
393         turnRENCA = RENCA;
394       }
395       break;

396     case 3: //Backup to center
397       if ((turnLENCA - LENCA > 400) && (turnLENCA - RENCA > 400)) {
398         digitalWrite(RMotorEN_1, LOW);
399         digitalWrite(RMotorEN_2, LOW);//If rotation distance is 400, stop
400         digitalWrite(LMotorEN_1, LOW);
401         digitalWrite(LMotorEN_2, LOW);
402
403         analogWrite(RMotorFWD, 0);
404         analogWrite(LMotorFWD, 0);
405         analogWrite(RMotorBWD, 0);
406         analogWrite(LMotorBWD, 0);
407         statepreviousMillis = currentMillis;
408         Robot_Sequence = 4;
409       }
410       else if ((turnLENCA - LENCA < 400) && (turnLENCA - RENCA < 400)) {
411         digitalWrite(RMotorEN_1, HIGH);
412         digitalWrite(RMotorEN_2, HIGH);//If rotation distance is not 400, continue
413         digitalWrite(LMotorEN_1, HIGH);
414         digitalWrite(LMotorEN_2, HIGH);
415
416         analogWrite(RMotorBWD, 100);
417         analogWrite(LMotorBWD, 100);
418       }
419
420       break;
```

```
828  void leftENCA() {
829
830    if (digitalRead(LEncoderB) == HIGH) {
831
832      LENCA++; //If B side goes high, going forward so increment
833    }
834
835
836    else {
837      LENCA--; //Else if B is LOW, going backward so decrement
838    }
839  }
840
841
842  void rightENCA() {
843
844
845    if (digitalRead(REncoderB) == HIGH) {
846
847      RENCA--;//If B is LOW, going backward so decrement
848    }
849
850    else {
851      RENCA++;//Else B side goes high, going forward so increment
852    }
853  }
```

3. Product Specifications

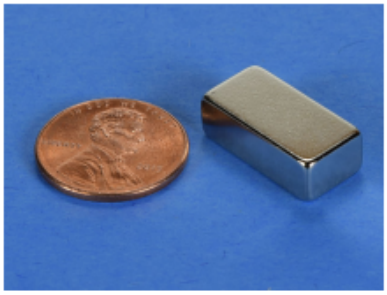
a. Magnets

K&J Magnetics, Inc.

Your Source for the World's Strongest Magnets

BC64 Specification Sheet

Product Specifications



Type:	BLOCK
Dimensions:	0.75 x 0.375 x 0.25 thk (in)
Tolerance:	All dimensions ± 0.004 in
Material:	NdFeB, Grade N42
Plating:	NiCuNi
Max Op Temp:	176°F (80°C)
Br max:	13,200 Gauss
BH max:	42 MGOe

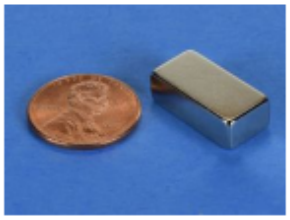
Performance Specifications

Pull Force, Case 1, Magnet to a Steel Plate:	14.36 lb
Surface Field:	4404 Gauss

Surface Field values are derived from calculation and verification with experimental testing. These values are the field values at the surface of the magnet, centered on the axis of magnetization. Measurement of the B field with a magnetometer may yield varying results, depending on the geometry of your sensor. Pull Force values are based on extensive product testing in our laboratory. Different configurations of magnets and surrounding ferromagnetic materials may substantially alter your results.

BC64 Safety Data Sheet (SDS/MSDS)

1. Identification, Company & Product



Product Name: BC64 Sintered Neodymium-Iron-Boron (NdFeB) Permanent Magnet

Product Use: Permanent magnet for various uses.

Company: K&J Magnetics, Inc.
18 Appletree Ln.
Pipersville, PA 18947
1-888-746-7556 (215-766-8055)
http://www.kjmagnetics.com

2. Hazards Identification

Neodymium magnets are extremely powerful. The incredibly strong force between magnets can cause injury. Fingers and other body parts can be pinched between two magnets. With large magnets, injuries of this type can be severe.

Strong magnets may affect the operation of pacemakers or other implanted medical devices.

3. Composition / Information on Ingredients

Chemical Name: Sintered Neodymium-Iron-Boron (NdFeB) Permanent Magnet

Material	Weight %	CAS #:	ACGIH TLV (mg/m^3)	Notes
Neodymium	approx. 33%	7440-00-8	Not established	
Iron	approx. 65%	7439-89-6	10 (oxide)	
Boron	approx. 1.3%	7440-42-8	10	
Nickel	0.01-0.4%	7440-02-0	1(dust)/0.1(fume)	plating
Copper	0.01-0.2%	7440-50-8	1(dust)/0.1(fume)	plating
Dysprosium	0-4%	7429-91-6	Not established	hi-temp grades
Cobalt	0-5%	7440-48-4	0.02	hi-temp grades

4. First-Aid Measures

Eye Contact:
Small pieces, chips or dust from magnet material may cause irritation. Wash eyes gently under

running water for 15 minutes or more to remove. If symptoms persist, seek medical attention.

Skin Contact:
Prolonged skin contact may cause irritation or allergenic dermatitis, especially for individuals with nickel allergy.

In the case of contact with chips or dust from a broken magnet, brush off powders and wash well with soap and water.

Inhalation:
Rare. If vapors or dusts have been inhaled, move individual to fresh air and seek medical attention.

Ingestion:
If a magnet is swallowed, seek medical attention immediately. If multiple magnets are ingested, magnets can stick together through intestinal walls, causing serious infections and death. Seek immediate medical attention.

Information for Doctors:
Ingestion of multiple magnets can pose a serious risk. Consider consulting the algorithm presented in, "Management of Ingested Magnets in Children," (Hussain et al., 2012).

Strong magnetic fields found near neodymium magnets can interfere with the operation of implanted medical devices such as pacemakers and ICDs.

5. Fire-fighting Measures

Flammable Properties:
Dry powders of neodymium magnets will oxidize, smolder, and burn rapidly in the presence of air or oxygen. Maintain powders in water slurry or in inert atmospheres of nitrogen or argon to prevent spontaneous combustion. Magnets may spark on impact. Handle carefully in explosive atmospheres.

Extinguishing media:

Suitable:	Sand or dry chemicals without oxygen compounds
Unsuitable:	Do not use Halon agents or water on smoldering or burning powder.

6. Accidental Release Measures

Personal precautions, protective equipment and emergency procedures:
No special measures required. See Section 7 for information on safe handling.

Environmental precautions:
Not applicable.

Methods and Material for containment and cleaning up:
No special measures required. Pick up mechanically.

Reference to other sections:
See Section 7 for information on safe handling.

7. Handling and Storage

Large magnets can attract to one another. Strong attractive forces can cause injury. Impacts of magnets can eject chips or bits of magnet material at speed - eye protection should be used.

Strong magnetic fields may affect the operation of implanted medical devices such as pacemakers and ICDs.

If magnetic dust is formed, sweep up dust and store in water slurry or sealed containers utilizing inert atmosphere such as argon or nitrogen to prevent spontaneous combustion.

8. Exposure Controls / Personal Protection

This product is considered to be an article which does not release or otherwise result in exposure to a hazardous chemical under normal use conditions. No engineering controls are necessary.

Protection of hands:
Avoid repeated and prolonged contact with the skin, especially if user has known nickel allergies. Protective gloves may be used.

Eye protection:
Safety goggles should be worn when handling magnetized magnets.

Keep mechanical/electrical instruments which may be damaged by high magnetic fields at some distance away from neodymium magnets.

9. Physical and Chemical Properties

Information on basic physical and chemical properties.

Physical state:	Solid
Color:	Silver/gray metal
Odor:	Odorless
Density:	7.5 grams per cubic centimeter
Specific Gravity:	7.5 (H_2O = 1)
Solubility:	Not water soluble
Melting Point:	Above 1000°C (1832°F)

10. Stability and Reactivity

Possibility of hazardous reactions:
Hydrogen maybe released in contact with acid, which can cause explosive gas mixtures. Reacts with strong acids to form hydrogen gas. Do not store near strong oxidizers.

11. Toxicological Information

Neodymium compounds are of low to moderate toxicity, yet its toxicity has not been thoroughly investigated. Neodymium dust and salts are very irritating to the eyes.

Roughly 2/3 of a neodymium magnet is composed of iron. Inhalation overexposure of iron dust may cause siderosis, a benign pneumoconiosis.

Most neodymium magnets are plated with nickel. Prolonged contact with nickel may cause sensitization dermatitis, or nickel allergy. Nickel is a listed carcinogen.

12. Ecological Information

No specific information available for this product.

13. Disposal considerations

Dispose in accordance with federal, state and local regulations.

Large, powerful magnets may be demagnetized with high temperatures before disposal to prevent possible handling injury.

14. Transport Information

Magnets can generate magnetic fields that may affect navigation equipment. Magnets are able to attract ferromagnetic materials.

For air transport, neodymium magnets may or may not require a hazardous material label. See the International Air Transport Association's (IATA) Dangerous Goods Regulations (DGR) and FAA Title 49, Part 173.21.

15. Regulatory Information

The following list of regulations may not be complete and should not be solely relied upon for all regulatory compliance responsibilities.

This product is an article as defined by TSCA regulations, and is exempt from TSCA Inventory

requirements.

RoHS: This product is in compliance and conforms to the European Union's Restriction of Use of Hazardous Substances in Electrical and Electronic Equipment (RoHS) Directive of 2002/95/EC, 2011/65/EU (RoHS2), and 2015/863 (RoHS3).

REACH: This product does not contain, or contains less than 0.1% (by weight), any substances of concern as detailed in REACH EC-1907/2006. This declaration includes the 224 REACH SVHC items as of June 10, 2022. As the list of SVHC substances keeps expanding, K&J Magnetics will continue to monitor and test this statement's applicability.

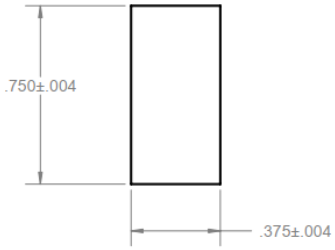
California Prop. 65: This product may contain ingredients which the State of California has determined may cause cancer.

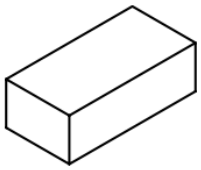
16. Other Information

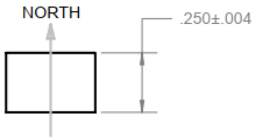
WARNING:

RARE EARTH MAGNETS ARE EXTREMELY POWERFUL!
They exhibit very strong magnetic forces which make them attract to other magnets and other ferromagnetic materials such as iron or steel.
HANDLE WITH EXTREME CAUTION!

The above information is believed to be correct but does not purport to be all-inclusive and shall be used only as a guide. K&J Magnetics, Inc. shall not be held liable for any damage resulting from handling or from contact with the above product.







GRADE:	N42
Br:	13,000-13,200 GAUSS
BHmax:	40-42 MGOe
CURIE TEMP:	310 °C
PLATING:	NICKEL-COPPER-NICKEL PLATING

NOTE: ALL EDGES HAVE A CHAMFER/RADIUS NOT TO EXCEED 1/32"

UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES

TOLERANCES:
FRACTIONAL ± 0.030
ANGULAR: ± 1°
TWO PLACE DECIMAL ± 0.004
THREE PLACE DECIMAL ± 0.004

MATERIAL SEE NOTES
FINISH SEE NOTES

K&J Magnetics, Inc.

TITLE: BLOCK MAGNET

SIZE	DWG. NO.	REV
A	BC64	1

SCALE: 2:1 | DO NOT SCALE DRAWING | SHEET 1 OF 1

5 4 3 2 1

b. Microcontroller

PRODUCT INFORMATION	**SKU**	422238
	Mfr Part#	MEGA 2560 BOARD
	UPC	618996977871
GENERAL	**Board Type**	Mega2560
	Component Type	Development Board
	Components	Mainboards
MAINBOARDS	**Board Color**	Blue
	Processor	ATmega2560 Microcontroller
	Clock Rate	16MHz
	Operating Voltage	5V
	Input Voltage	7V - 12V
	Input Voltage (Limits)	6V - 20V
	Digital I/O Pins	54 (of which 15 provide PWM output)
	Analog Input Pins	16 Pins
	DC Current per I/O Pin	40mA
	DC Current for 3.3V Pin	50mA
	Flash Memory	256KB
	SDRAM	8KB
	EEPROM	4KB