

Sponsor: University of Texas at Tyler Electrical Engineering Department

Sponsor Contact: Dr. Mukul Shirvaikar

Advisor: Dr. Prabha Sundaravadivel

By: Richard Sij

Michelle Tamdjo

Wyatt Snare

Josue Rios

Joshua Luker

Acknowledgements

The members of the Tyler Team 9 would like to thank Dr. Mukul Shirvaikar and the Electrical Engineering Department at UT Tyler for their support as sponsors and for their input and ideas on the designing of this project. We would also like to thank Dr. Prabha Sundaravadivel and Dr. Hassan El-Kishky for being our advisor and guiding us in the right direction, providing valuable feedback and expertise. Lastly, we would like Ed Farina for all the help and advice given when we were using the 3D printer and laser cutter.

Executive Summary

The focus of this project is to create an autonomous recycling robot able to pick up different types of litter that include plastic, aluminum, paper then dispense it to their respective trash bin. Raspberry Pi camera was used to detect the direction to help find and dispose of trash. Servos will be used to move the bumpers and close the claw. DC motors will be used for the movement the entire robot. Raspberry Pi will be the brain of the robot. The Pi board will process the pictures from the Raspberry pi camera which will be used for object detection. The claw will be used to pick up the trash once the robot identifies which trash it has come to. Results expected for this robot is a fully working robot able to detect trash and properly dispense it to their proper bins. While this was the full intent of the project due to unseen circumstances the robot had to be changed from autonomous to remote control. While the robot is one hundred percent designed to run autonomously with adding a remote control the robot will now be controlled by any user.

Table of Contents

Introduction 1

Requirements, specifications, and constraints 2

List of relevant standards used 3

Final design, details, and consideration 4

Final System, Subsystem Level Black Box, and Functional Diagram 5

Final design calculation 10

List of risks involved and Failure mode analysis 15

Final Test Results 17

Final Simulations Testing Results 17

B.O.M. and Budget 18

Economic Feasibility 19

Flow Chart of Software 20

Final Acceptance Criteria 21

Discussion 22

Recommendations 24

Conclusion 25

Appendices 26

- Poster 26
- Sign off Sheet 27
- Economic feasibility (Cost of prototype vs. Mass production cost) 30
- Timeline and milestones with dates 31
- ARCI Chart 32
- Mechanical Drawings 33
- Electric Circuits Diagrams 40
- Any Code or Program Listing 48
- Detailed design calculations, formulae derivations, and software extra analysis output 65
- Design of experiments, experimental procedures, and results 67
- Final Assembly Drawings 68

User Manual 83

List Of Figures

Figure 1. Final robot 4
Figure 2. System Black Box Diagram 5
Figure 3. Bumpers and Claw Sub-System Black Box Diagrams 7
Figure 4. Movement Sub-System Black Box Diagram 8
Figure 5. Function Structure Diagram 9
Figure 6. Torque for the motor diagram 11
Figure 7. Torque for Servo HS-422 12
Figure 8. Torque for High Torque 13
Figure 9. Risk Assessment Matrix 15
Figure 10. Software Flowchart 20

List of Tables

Table 1. IEEE Standards for Equipment to be Used 3
Table 2. Torque Calculations for RPM Gear Motor 11
Table 3. Torque Calculations for HS- 422 12
Table 4. Torque Calculations for High Torque Servo 13
Table 5. Entire System Power Calculation 14
Table 6. Risk Assessment and Contingency Plan. 16
Table 7. B.O.M. 18
Table 8. Totals 19

Introduction

Pollution of various forms is an ever-present problem in our modern world. For the last 50 years, considerable attention has been focused on larger institutional sources of pollution, and while there is more to do, significant progress has been made. Many individuals have also learned to make better personal choices, yet the basic problem of litter remains pervasive worldwide. There are still large numbers of people that feel entitled to simply drop or toss waste items they are through with wherever they are. This is particularly evident walking along a roadside or looking in or underneath the stands after a sporting event. In much of IEEE region 5, high winds and open country allow outdoor litter to be dispersed from its source into other settings. It commonly finds its way into streams and lakes, where it causes additional environmental problems. It also commonly appears on the grounds of people that have personally made efforts to avoid creating litter, insulting them, and adding to their burdens.

There are three different approaches required to address this problem. It is important to educate those who do not already know better about the consequences of their behavior. It is also important to create a form of accountability for those that continue polluting behavior. Finally, it is important to responsibly clean up the mess before it creates greater problems. This year's game has elements representing each of these areas. But unlike many former games, we are not merely cleaning up some symbolic representation of trash; we will be picking up and properly disposing of four extremely common articles of actual litter. With a modest amount of additional work, we expect a successful robot from this year's competition could be used to help clean up areas that have been badly littered, such as grounds after gatherings of any types of entertainment.

Requirements, specifications, and constraints

1. Robot can be no larger than 24” x 24” x 24”. Once the match has started, it may extend out past the starting limit to 36” diameter

2. The robot must be a single entity and remain that way. Parts must not be detached and left on the field.

3. The robot, as it will be placed on the field, must not weight more than 25 pounds.

4. The robot must have a prominent stop button that will stop all motion by and within the robot, but not necessarily remove power from the control elements. It must be possible for team members or the referee to access this button easily and rapidly at any time regardless of the position of any other mechanisms of the robot. This could be used to stop the robot at the end of the match or in instances where the referee has determined it to be a hazard.

5. The robot must not contain any hazards to people or the litterbug, or cause damage to the field or game pieces. Example hazards could include (but are not limited to) sharp edges, corners, or points that could cut or tear skin or the field tiles, exposed electrical contacts that could deliver a significant shock, mechanisms that could throw litter significantly outside the playing field, or wedge shaped elements that could tip the litterbug. Under no circumstances may highly flammable liquids or gasses, or explosive materials be used in the robot.

6. No pneumatic pressures greater than 100 Psi shall be used. At inspection teams using pneumatics should be able to provide documentation that all components are rated for the operating pressure used.

7. Robots must not leak or drip any fluids, including lubricants. The only exception would be for residual fluids present in the game piece litter.

List of relevant standards used

Applicable standards have been set by IEEE competition and advisors. These standards rely on the mainly on the safety of the product. The standards set by the IEEE competition was straightforward for this project. Team research and design has been with these concepts in mind.

Table 1. IEEE Standards for Equipment to be Used

Standard No.	Description
1	No open driving mechanisms (chains)
2	No sharp corners or edges exposed
3	No wires exposed
4	No spillable or flammable battery
5	Obvious and simple to operate kill switch (if powered)
6	No leak and drip any fluids

Final design, details, and consideration

Due to recent events of the COVID-19 outbreak, our project has changed up a little. The IEEE Robotics competition is no longer able to host the event. With this unfortunate event we have been instructed to adapt to our surroundings and produce a robot to instead work from a controller that can be used for instructional use at the university. Although these events have been difficult, we as a team have come together and fabricated a fully functioning robot that can be controlled from a Bluetooth controller. All the moving components on the robot are able to function correctly and move as intended.

Of course, the rules for our design have changed. We no longer need to compete against other robots or pick up a certain amount of trash in a timely manner. We have kept every portion of the robot the same, but the only difference is that it is being controlled from a controller.

Figure 1 below is a diagram of the Final robot.

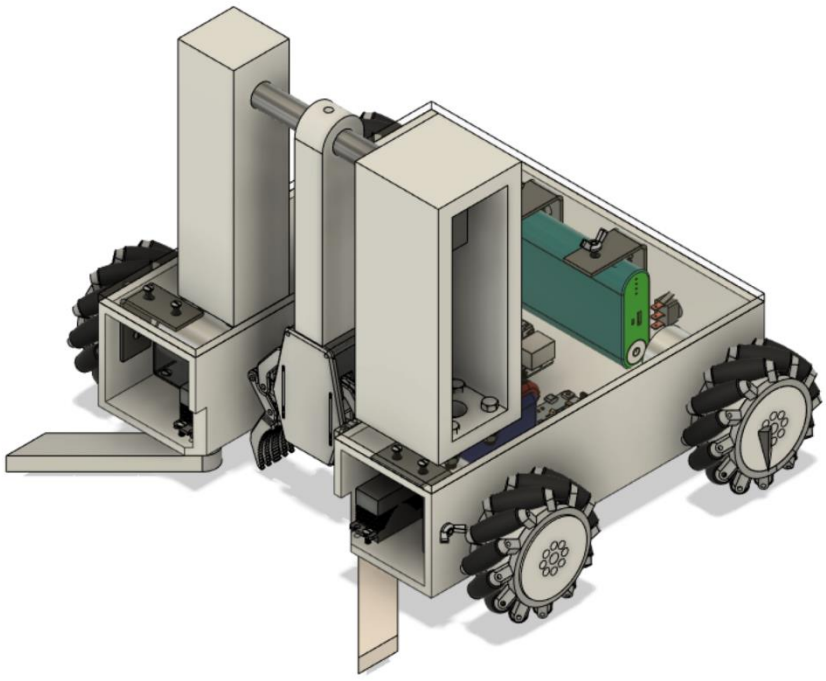


Figure 1. Final robot

Final System, Subsystem Level Black Box, and Functional Diagram

The robotic design created to meet specifications of the IEEE region 5 competition includes a large system which encompasses many smaller sub-systems to accomplish competition objectives. The larger system of the robotic design is made up of the following elements: power, claw, wheels, motors, remote, body, raspberry pi, and Arduino. These elements of the robotic design working in unison can generate the desired outputs of the following: movement, trash pickup, trash recycling, and noise. The system described can be found in Black Box level detail below in Figure 2, the system black box diagram.

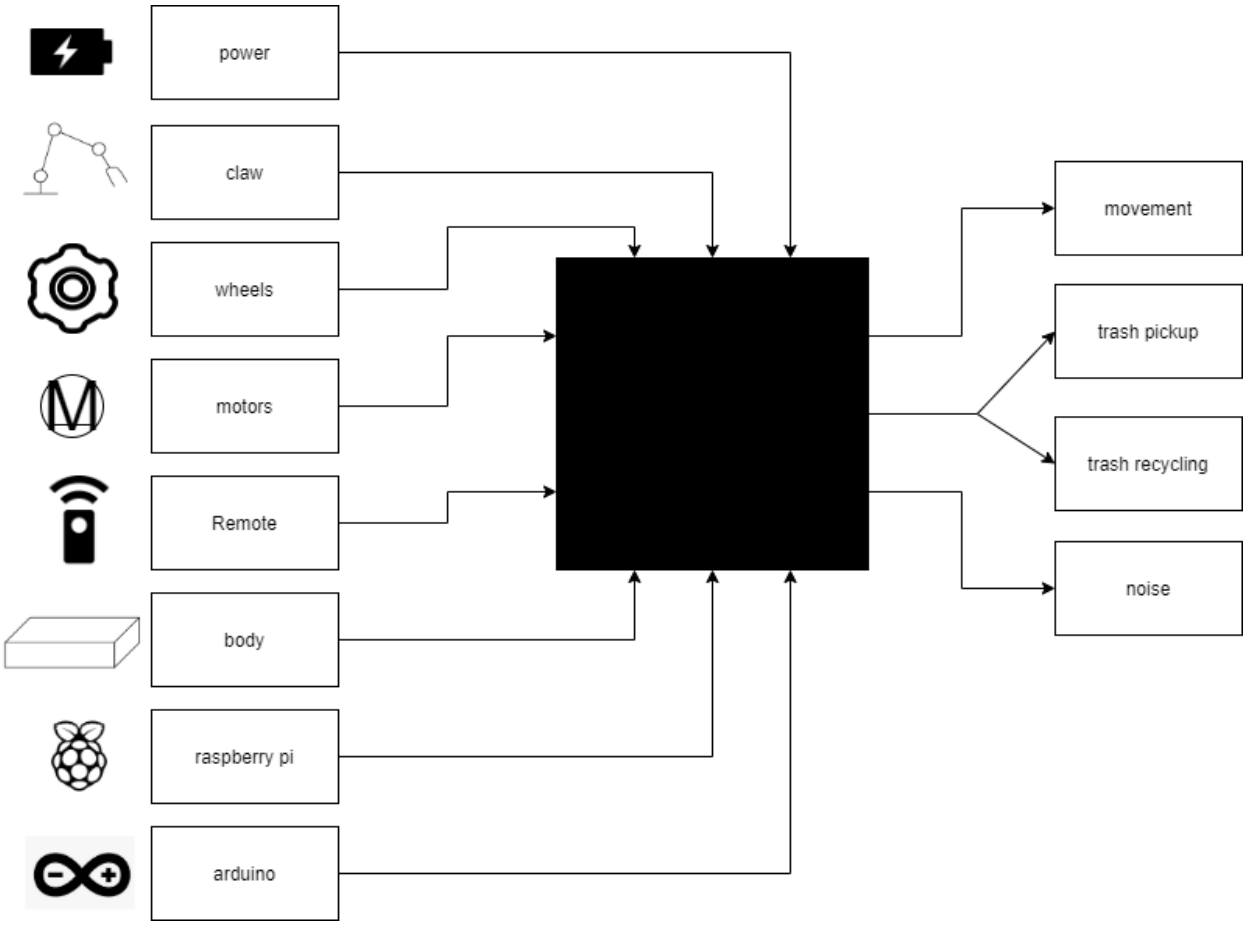


Figure 2. System Black Box Diagram

The smaller sub-systems involved in the design of the IEEE region 5 robotics competition robot include those of the following: Bumpers, Claw, and Movement. The Bumpers sub-system requires inputs such as power, 3-D printed bumpers that are capable of moving trash to more desired locations, servos with required torque that will move the 3-D printed bumpers, a newly implemented remote control for operations outside of the competition, and a single Arduino Mega 2560 model that will handle all electrical communications for the bumpers sub-system. The Bumpers sub-system output is that of object positioning. The Claw sub-system requires inputs such as power, a mechanical claw that will grab and hold onto desired items, a 3-D printed arm that will attach to the claw, 3-D printed supports that will hold the claw arm, a body that will mount the claw arm and supports, a High Torque servo which is capable of moving the 3-D printed arm, A single Arduino Mega 2560 model that will handle all servo based operations and electrical communication for the claw sub-system, a servo-adapter-board capable of establishing communication between the high torque servo and the Arduino, and a newly implemented remote for use outside of competition operations. The output generated by the claw subsystem is Object Pickup. The sub-systems described can be found in Black Box level detail below in Figure 3, the Bumpers and Claw sub-system black box diagrams.

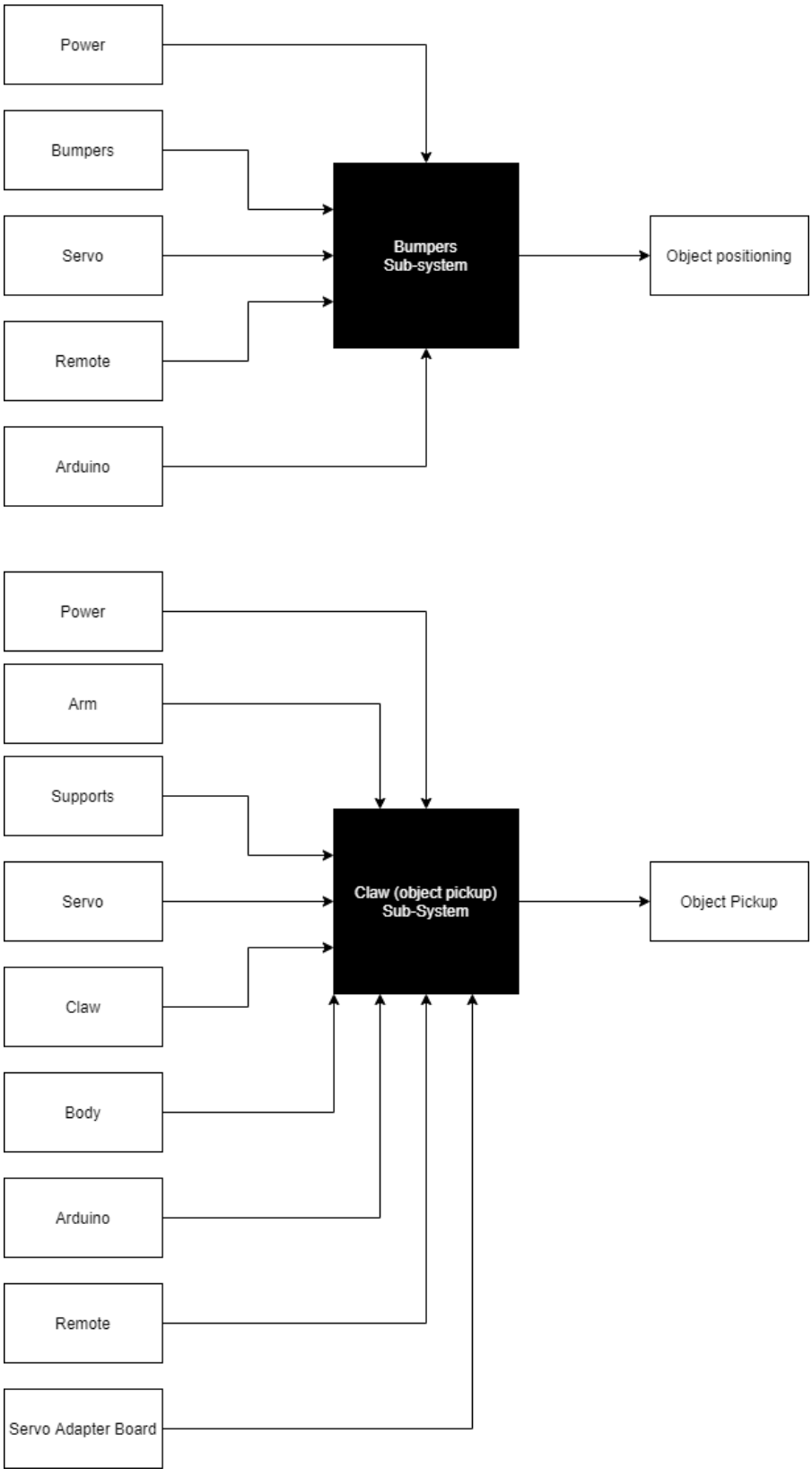


Figure 3. Bumpers and Claw Sub-System Black Box Diagrams

The other smaller sub-system involved in the design of the IEEE region 5 robotics competition robot is the Movement sub-system. The Movement sub-system requires inputs such as power, wheels capable of moving a robot under 25 lbs., Motors with enough torque to move the wheels of the robot without failure, a single Arduino responsible for the communications of the movement sub-system, a driver shield capable of establishing communication between the Arduino and the motors which will turn the wheels, and a newly implemented remote for operations outside of the competition. The outputs of the movement sub-system generated by the simultaneously operating inputs include obstacle avoidance and successful navigation. The sub-system described can be found in Black Box level detail below in Figure 4, the Movement sub-system black box diagram.

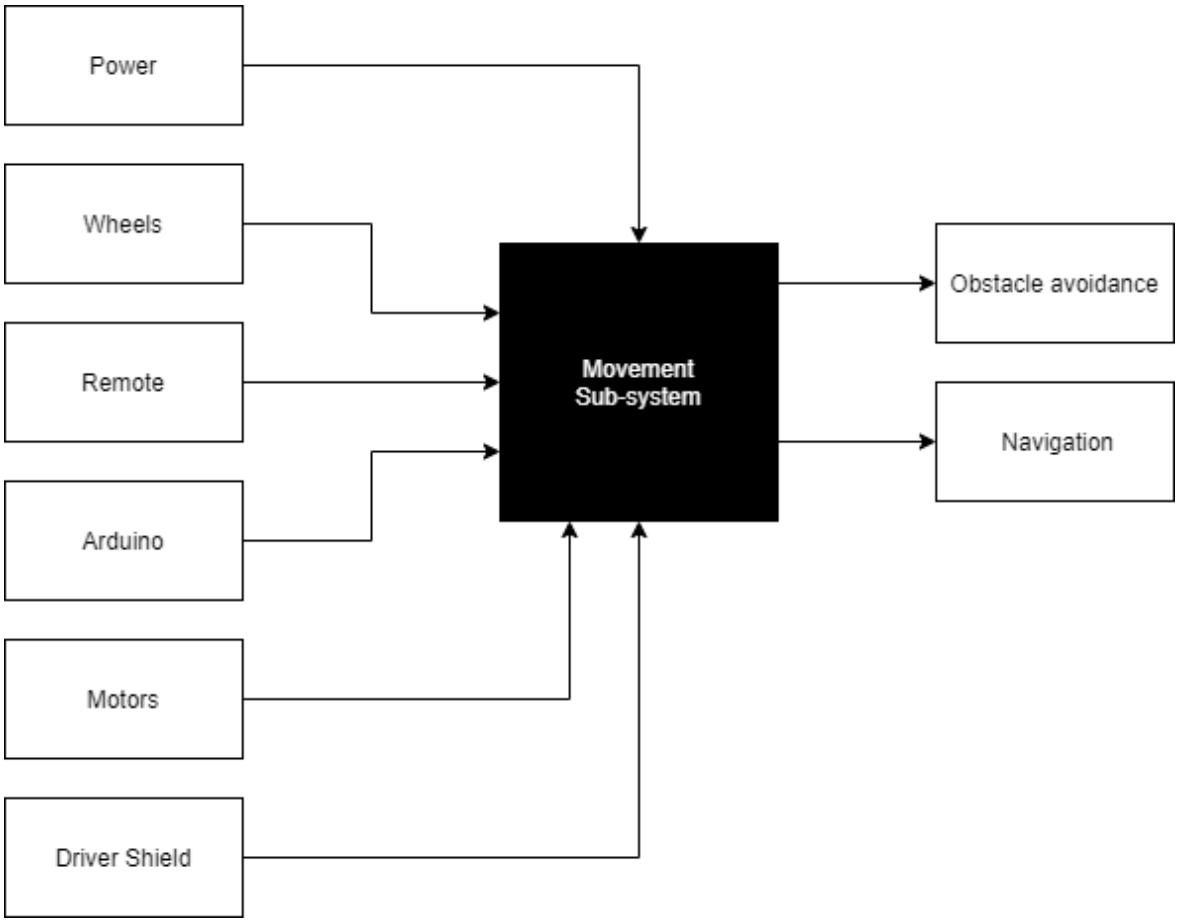


Figure 4. Movement Sub-System Black Box Diagram

The design of the IEEE region 5 robotics competition robot involves a flow of both several functions and structures. The flow of Program Code will allow larger structures of function to occur such as wheel and claw movement. The flow of Program code and functioning structure of claw movement allows for subsequent structure of Trash Pickup to occur. This structure of trash pickup subsequently generates a new flow of trash placement into appropriate bins. The flow of program code is also a larger structure which operates from the flow of the newly implemented remote data for outside of competition operations. Another flow involved in the operation of greater structures within the design of the IEEE region 5 robotics competition robot is Electricity. The flow of electricity allows the structures of wheel and claw movement. Without both the flow of program code and electricity the structures of wheel and claw movement are not possible for this design. The flow of electricity along with the functioning wheel and claw movement generates the output of noise. This relationship of functions and structures can be found in further detail below in Figure 5, the function structure diagram.

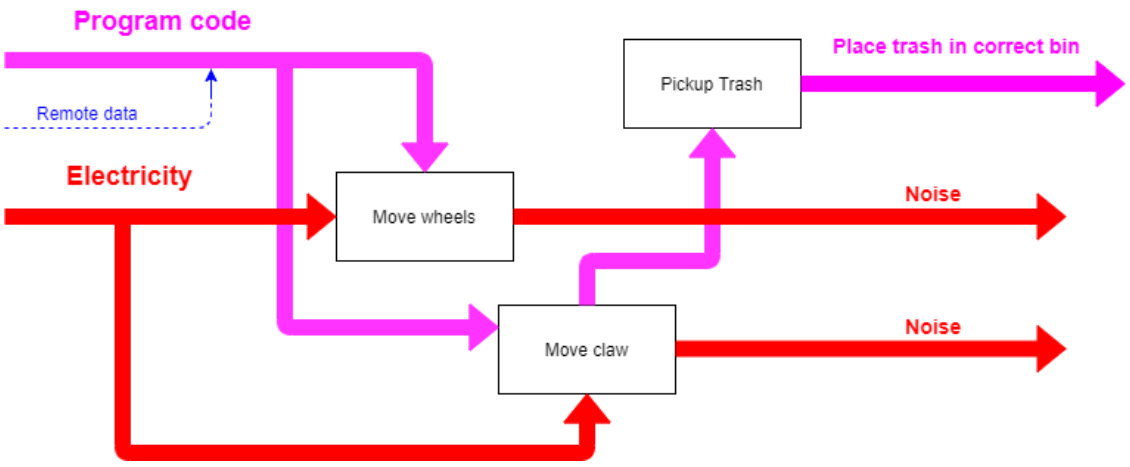


Figure 5. Function Structure Diagram

Final design calculation

Analysis of the concept is the bottom to the top. There is a Raspberry Pie camera right at the center of the claw to detect the types of object in front of the robot. As the robot moves forward to grab the object facing downward the claw will be able to grab it. If the object is upward, the bumpers are used to pull flip the object downward to position it right at the front of the claw for the claw to properly grab it. The claw will move upward at 145° angle to dump into trash bin. Motor chosen are 101 RPM gear motor because the motors needed to have a lower RPM while maintaining a high torque ratio. The maximum ratio of the gear motor is 45.6 oz*in, Table 2 shows the calculations of the torque for the gear motor and Figure 6 demonstrate the free body diagram. Results show that the torque is 35.12 oz*in which is good because it is lower than the require torque. Hs-422 servo was use for the bumpers; Table 3 demonstrates the torque calculation for the servo. Figure 7 shows the free body diagram of the HS-422. The maximum torque for the servo is 45.82 oz*in and the calculate torque is 0.0048 oz*in which is good. Hs-422 servo was chosen because the flippers needed to be able to move slowly to change the detected object position.

Table 4 demonstrated the calculation of the high torque servo. The servo was chosen because the claw needed more force to lift the object at 145°. Figure 8 shows the free body diagram of the servo and the claw. The maximum torque require is 18 kg-cm which is less than the maximum 29 kg-cm. The material chosen for the body of the robot is ABS plastic and it was 3D printed; it allowed the robot to be a less than 25 pounds. The wheels are used as recycling based on the previous year's design. DC motors will be used as movement for the robot. Table 5 describes the calculations for the entire power system of the robot.

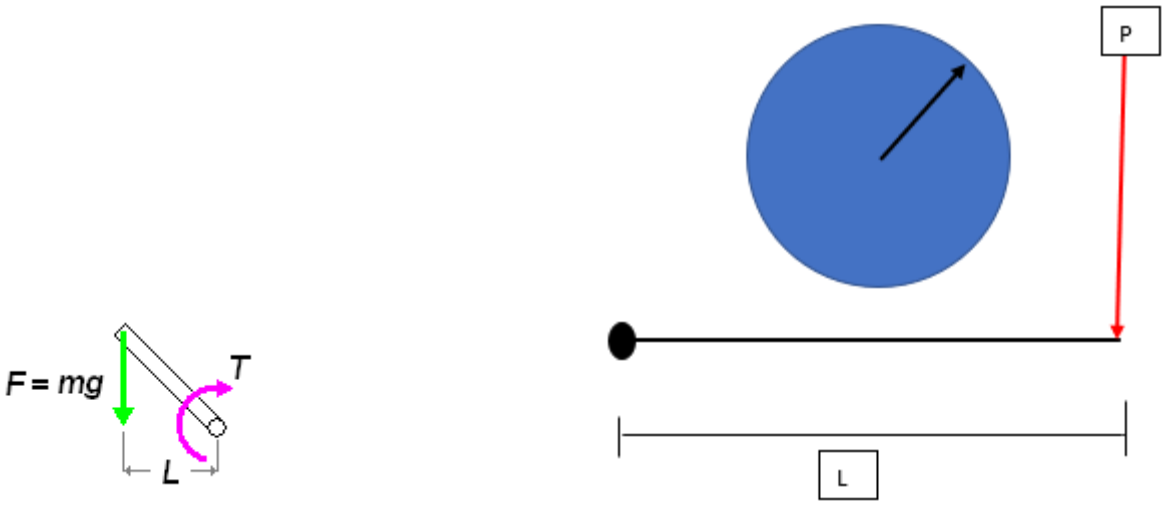


Figure 6. Torque for the motor diagram

Table 2. Torque Calculations for RPM Gear Motor

No.	Specifications	Symbol	Unit	Formula	Result
1	Radius	R	ft	Set Values	.21
2	Length	L	ft		0.42
3	Mu	μ	(not sure)		.72
4	Gear Ratio	GR	(not sure)		2x30
5	Gravity	g	ft/s^2		32.2
6	Mass	m	lb.		25
7	Force	F	N	$F = m * g$	805
8	Load	P	(Not sure)	$P = R * F$	7.5
9	Force (not sure where this comes from or what is the name of it. it was on the previous report)	Fj	N	$F_j = P * R$	1.56
10	Torque	T	Ft x lb. or Oz X in	$T = \frac{(\mu * F + mg) x L}{GR}$	0.1829/35.12

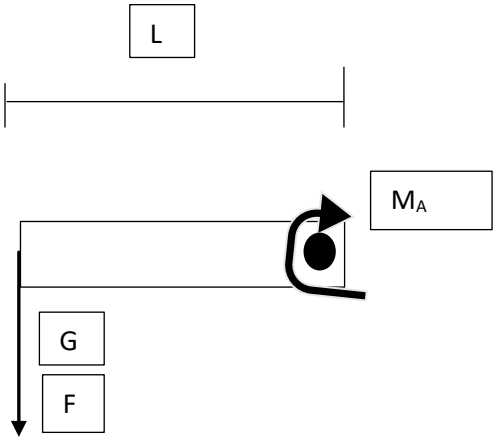


Figure 7. Torque for Servo HS-422

Table 3. Torque Calculations for HS- 422

No.	Specifications	Symbol	Unit	Formula	Result
1	Length	L	ft	Set Values	.42
2	Gravity	g	ft/s^2		32.2
3	Mass	m	lb.		0.675
4	Force	F	N	$F = m * g$	2.17
5	Moment at A/ Torque	M_A	lb. *ft/ oz*in	$M_A = L * F$	0.92/ 0.048

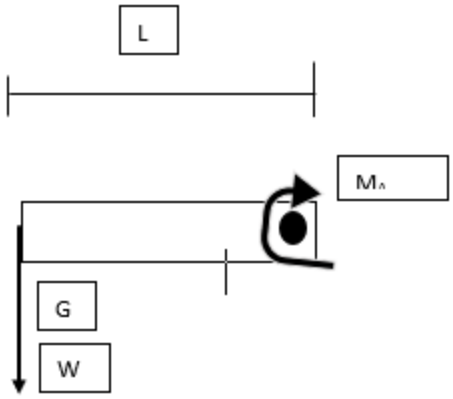


Figure 8. Torque for High Torque

Table 4. Torque Calculations for High Torque Servo

No.	Specifications	Symbol	Unit	Formula	Result
1	Length	L	cm	Set Values	25.4
2	Gravity	g	cm/s^2		980
3	Mass	m	Kg		0.725
4	Moment at A/ Torque	M_A	Kg-cm	$M_A = L * m * g * \cos(90)$	18.046

Table 5. Entire System Power Calculation

No.	Specifications	Symbol	Unit	Formula	Result
1	Driver Current	A	A	Set Values	0.036
2	Battery	B	mAh		1500
3	Time	t	min		2
4	Battery voltage	P_b	V		11.1
5	Driver voltage	P_d	V		0.5
6	Arduino voltage	P_A	V		5
7	Motor voltage	P_m	V		11.1
8	Arduino Input	A_I	inputs		0.24
9	Power Input	A_m	A		0.25
10	Maximum Amp	A_{max}	A	$A_{max}= \frac{B}{(t * 1000)}$	45.45
11	Total P_b	T_{Pb}	W	$T_{Pb}= P_b \times A_{max}$	504.95
12	Driver Power	T_{Pd}	W	$T_{Pd} = P_d \times A$	0.072
13	Arduino Power	T_{PA}	W	$T_{PA} = P_A \times A_I$	1.2
14	Motor Power	T_{Pm}	W	$T_{PA} = P_m \times A_m$	11.1
15	Total power	T	W	$T= T_{Pd} + T_{Pm} + T_{PA}$	12.73
16	Overall Power System	T_T	W	$T_T = T_{Pb - T}$	492.12

List of risks involved and Failure mode analysis

Risks are in everything we do as engineers. Since there are so many risks, as engineers we have come up with rick assessments to justify the risk involved to see if the project is worth going forward. There are three main components to a risk. The first one is the impact risk, which means what impact on environment or people around would the project have? The second one is the probability of occurrence. This one means that how many times will the project mess up or cause a problem in a given time. Finally, are the consequences that occur if the risk is not mitigated. Which means what will happen if I go ahead with the project and the risks outweigh the reward. Figure 9 shows the risk assessment matrix's comparing likelihood to severity. Table 6 was used to determined level of risks in this project.

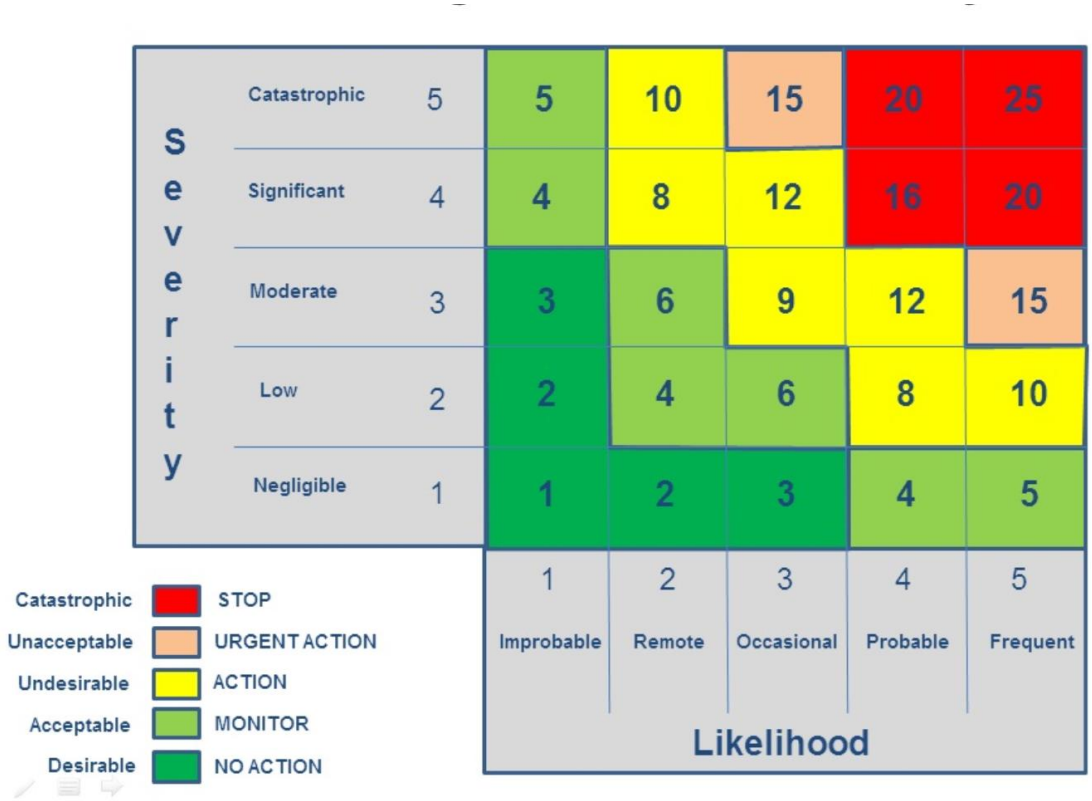


Figure 9. Risk Assessment Matrix

Table 6. Risk Assessment and Contingency Plan.

Risk	Description	Level	Contingency
1	High Toque Servo coming unmounted due to repetitive motion	4	Use locking washers and nuts to make sure they do not come lose
2	Battery pack dying during the competition	12	Changed out battery pack to one with a greater charge that will last longer
3	Wheels slipping on surface	6	Moved most of the weight of the bot as close wheels so the down force will be greater over wheels
4	Wire coming lose during operation	8	Designed the lid to the robot to be easily removable
5	3D print not being strong enough to hold screw	20	When Printing added 80% infill to parts with screws
6	Arduino board buns out	5	Add static protective case to help

Final Test Results

The robot was built to meet the following criteria: Size less than 24” by 24” by 24”, Pneumatic pressures less than 100 Psi, must have a prominent stop button, must not contain any hazards to people, must be a single entity and remain that way, must not leak or drip any fluids, must not weigh more than 25 pounds. The robot at the end of the testing phase met all criteria listed above. The robot was weighed using a hang scale and was suspended using rope. The robot weighs 13.5 lb. after completed assembly. The robot does not leak or drip fluids of any kind and does not operate using pneumatic pressure. The robot was originally designed and built to work autonomously for competition purposes. While the robot was able to successfully complete all desired operations autonomously, it was later converted to operate remotely for future use by the department of Electrical Engineering at UT Tyler. Both remote and autonomous modes of operation succeeded in attempts to locate, grab, and lift different articles of trash as the robot was designed to. The robot while operating autonomously uses the “candy-edge” object detection method. This method allows the robot to blur the background and highlight edges of objects in the foreground of the visual field. Initial attempts of object finding were unsuccessful, but after further calibration as the robot entered the final stages of testing it was able to successfully identify all desired trash articles, I.E.: water bottles, paper trays, soda cans, and chip bags.

Final Simulations Testing Results

As mentioned previously due to circumstances out of our control, our robot has been changed over to work from a Bluetooth controller rather than working autonomously. Testing the robot was quite simple due to the change of operation. The team was able to place our robot on an arena floor with various types of trash scattered around the arena. While one team member took control, we were able to have the robot flawlessly move across the floor and pick up each piece of trash and deposit it in the different trash cans.

The tests took place over several days. As a team we decided that we wanted to ensure that the robot was up to par will all standards and could work for the original 2-minute time period. The tests were performed in 2-minute intervals which were 5-minutes apart. The robot kept a charge and was able to complete all tasks as expected. Also tested on the robot was the arm/claw. The arm and claw were tested with several different weights in order to ensure that it was able to pick up every piece of trash that was needed and also to ensure that it was durable enough to last as long as we needed it to. We were able to pick up a full water bottle with ease, which is much heavier than any piece of trash that is required.

B.O.M. and Budget

Listed below in Table 7, is the bill of materials (B.O.M.) which is used to list out every component required to build the robot.

Table 7. B.O.M.

Qty.	Component list Parts Name	Part #	Cost
1	Remote Control	Logitech F710	$40.00
2	Arduino	Mega 2560	$37.98
1	Raspberry Pi 4 Board	16 Gig	$116.32
1	Claw Bracket w/ servo	N/A	$48.72
4	Omni Wheels	B01CTUTDB0	$135.95
4	DC motors	638194	$120.99
1	Raspberry Pi Camera	B0031	$18.99
1	High-torque servo	HT1-retail	$109.95
1	Servo Adapter Board	Lynxmotion	$42.00
3	Traditional Servo	Hitec HS-422	$25.00
1	Driver shield	1438	$21.54
2	Rocker Switch	12VDC 3P SPST ON-OFF Oval Rocker Switch, 30A	$8.00
2	Batteries	Floureon 11.1 Volt, 35C, 1500mAh Battery Pack	$40.00
1	3-D Printed Base	N/A	$0.00
2	3-D Printed Flippers	N/A	$0.00
1	3-D Printed Arm	N/A	$0.00
1	Acrylic sheet	12 x 12 x 12 in	$12.00
30	Screws	3/8 by 1 long	$20.00
15	Screws	3/8 by 12 long	$7.20

From the B.O.M. you can get the information that is in Table 8. The important information is the total number of parts and total price of the robot.

Table 8. Totals

Totals	
Parts	74
Price	$804.10

Economic Feasibility

The robot designed had the body completely 3D printed, although some the materials of the robot were purchased other materials were already available for the team. The assigned budget for this prototype was $500 but to build the prototype the cost was about $800. The body of the robot was made of PLA filament, mecanum wheels and battery and those materials was already provided. The electrical components such as Arduino, Raspberry Pi camera, Raspberry, DC motors, servos and battery were all purchased. Table 7 describes a detailed summary of how much each components price. For the mass production cost for this prototype, there are certain steps to take to be able to achieve it. One major cost for scaling a product is the electrical certifications. FCC (Federal Communications Commission) certification is one of the certifications, having an FCC certification ensure the product is not emitting electromagnetic radiation radio waves that will interfere with other forms of communication with a cost of $2500. Battery certification, UN 38.3 is for the lithium battery certification cost for $1000. To build the body, the cost is around $10,000 depending on the size of the material. In conclusion, the final scaling cost needs to include all of the electrical certification costs required for the market area, the manufacturing setup costs – which includes the molds and assembly fixtures – and any miscellaneous costs like patents, liability insurance, and factory audits

Flow Chart of Software

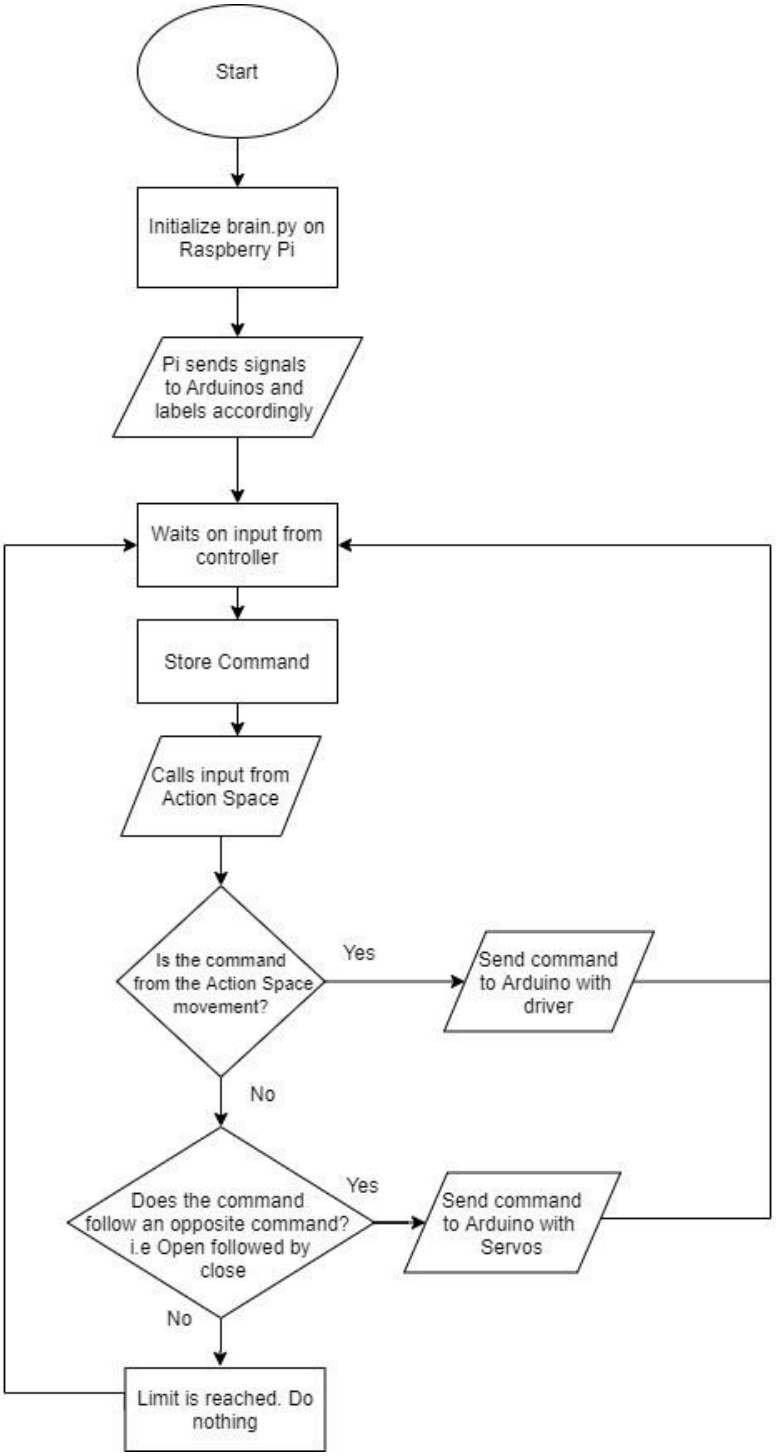


Figure 10. Software Flowchart

Final Acceptance Criteria

The final acceptance criteria for the IEEE region five robotics competition robot includes a fully functional prototype that meets requirements, specifications and constraints listed below. These requirements, specifications, and constraints include the following:

- Robot can be no larger than 24” x 24” x 24”. Once the match has started, it may extend out past the starting limit to 36” diameter

- The robot must be a single entity and remain that way. Parts must not be detached and left on the field.

- The robot, as it will be placed on the field, must not weigh more than 25 pounds.

- The robot must have a prominent stop button that will stop all motion by and within the robot, but not necessarily remove power from the control elements. It must be possible for team members or the referee to access this button easily and rapidly at any time regardless of the position of any other mechanisms of the robot. This could be used to stop the robot at the end of the match or in instances where the referee has determined it to be a hazard.

- The robot must not contain any hazards to people or the litterbug, or cause damage to the field or game pieces. Example hazards could include (but are not limited to) sharp edges, corners, or points that could cut or tear skin or the field tiles, exposed electrical contacts that could deliver a significant shock, mechanisms that could throw litter significantly outside the playing field, or wedge shaped elements that could tip the litterbug. Under no circumstances may highly flammable liquids or gasses, or explosive materials be used in the robot.

- No pneumatic pressures greater than 100 Psi shall be used. At inspection teams using pneumatics should be able to provide documentation that all components are rated for the operating pressure used.

- Robots must not leak or drip any fluids, including lubricants. The only exception would be for residual fluids present in the game piece litter.

- The robot must be capable of picking up four different types of trash: paper trays, soda cans, chip bags, and water bottles.

- The robot must be capable of disposing all trash items collected, I.E. picking up, moving, and dropping items.

Other acceptance criteria include the robot having the robot converted to work remotely rather than autonomously for future use by the department of electrical engineering at UT Tyler.

Discussion

The IEEE region five competition robot created and described in detail above has served as an excellent research project in the field of developing autonomous machines for large scale litter removal purposes. By creating a reliable machine that removes trash from its environment of operation it will greatly reduce the scale on which that environment is affected by litter. When creating such a machine there are several aspects to consider including the following: desired operation, electrical design, mechanical design, system programming, environment of operation, cost, budget, availability of parts, quality of parts, timeline, feasibility, and etc. The first and foremost element to consider is the desired operation of the machine. It is important to have a well-defined purpose for a machine before it goes into the electrical and mechanical design phases. It is also essential to consider the environment that the machine will be operating in to complete its desired purpose. Keeping such ideas in mind will help to avoid any design flaws from being included in the finished product. One example of a design flaw could include an autonomous robot that detects color to achieve its purpose but operates in an environment with poor lighting. If the robot is unable to detect color due to poor lighting than it is not able to achieve its purpose. This is a design flaw that could have been avoided if the environment of operation had been given more consideration. The design process for the IEEE region five competition robot was a difficult task to accomplish because the environment of operation was partially defined but constantly changing. For example, the original arena design where the competition robot would be operating was to have dark gray walls but later the walls were changed to be clear. This change to the environment of operation presented an issue to the prototype of the design that had already been created. The prototype based on the original design was created to account for the dark gray walls and ignore them but with the newly implemented clear walls the prototype robot was detecting objects in the background which interfered with successful identification of items in the foreground. To fix this issue the prototype robot required a finer tuning to the aspect of system programming. More specifically, the IEEE region five competition robot operates using the "candy-edge" object detection method. This method of object detection emphasizes on the outlines of objects in the foreground while blurring all areas of the background at specified intensities. By increasing the intensity of the blurred background, the newly implemented clear walls of the operating environment no longer presented an issue to identifying objects in the foreground. The electrical and mechanical designs of a machine are of substantial importance and require consideration of the following previously listed criteria: desired operation, system programming, environment of operation, cost, budget, availability of parts, and quality of parts, timeline, and feasibility. When creating an electrical or mechanical design feasibility is of paramount importance. It is often times easy to create a design that looks good on paper but fails to meet expectations when put into the prototype stages. These failures can often times be related to negligence in the categories listed above. Availability of parts is not to be overlooked in the design stages and it can affect the timeline of production. If the parts included in a design are not available during the prototype stage it may prevent the project from meeting deadlines. The cost of parts is also to be held into account during the design process as it directly involves the budget and future status of the prototype model(s). If the cost of a part is too expensive for a planned budget it may not be able to be replaced if it fails. At the same time, it is crucial not to sacrifice the quality of the parts that are included in the design process. If the parts included in a design are of poor quality, then they will often require

replacement which can impact the timeline of the project. If parts constantly fail and require replacement it may take days or weeks to replace them. These instances can cause a project to fall behind its scheduled deadlines. Instead it is important to consider high-quality low-cost alternatives during the design process. During the electrical and mechanical design process of the IEEE region 5 competition robot a part was included in the final design that presented several issues during the prototype stage in the areas of: timeline, budget, cost, part quality, and feasibility. The part included that presented these many issues is the Lynxmotion High Torque Smart Servo Motor. The Lynxmotion High Torque Smart Servo Motor is the unit that was chosen to lift the mechanical arm (see Appendix) included in the design was not capable of communicating directly with the Arduino Mega 2560 as advertised and required additional parts to establish this connection. The additional unit required to establish connection between the Arduino mega 2560 and the servo motor is the Lynxmotion Smart Servo Adapter Board. Ordering this smart servo adapter board required further expenses affecting the areas of both cost, budget, and timeline. With the smart servo motor incapable of operation upon arrival it delayed a portion of the testing and prototype phase of the design. Upon arrival of the Lynxmotion Smart Servo Adapter Board the servo motor entered into the testing phase where it was discovered to be defective. The servo motor when connected and at rest appeared to be pulling 2000 mA of current according to the Lynxmotion software. This current measurement while at rest indicates a faulty current sensor within the servo motor. The manufacturer was then contacted to order a replacement which affected the aspects of timeline and the quality of parts. Waiting for a replacement part from the manufacturer further delayed a portion of the testing and prototype phases of the design. Shortly after the arrival of the replacement part from the manufacturer the Lynxmotion Smart Servo Adapter Board began to malfunction for unknown reasons. This adapter board also required replacement from the manufacturer which again further delayed a portion of the testing and prototype phases of the design. The delay in the scheduled timeline for the IEEE region five competition robot could possibly have been avoided had a lower-cost higher-quality alternative such as a stepper motor been selected for the design instead.

Recommendations

The biggest setback to the robot was that the rules were changed continuously and were not finalized until all parts included in the original design were ordered. When the rules updated the big setback was that instead of only picking up four items of trash and having to stop the litter bug from throwing out more items, the rules were switched to picking up twelve items. With this complication I would recommend that the robot needs an updated higher capacity of storage. Right now, the robot can hold one item of trash at a time. If there was a type of storage system on the robot it would make picking items up faster and it could deposit trash at the end of the match instead of after picking up each item.

Communication is somewhere this robot could improve. The competition rules for communication were released way too late so there was no time to perfect the communication process. The communication device that is on the robot is only compatible with three out of the four modulation types included in the competition rules. The recommendation is to research and find a better transceiver and receiver that is capable of performing all four modulation types.

Improving the movement system is the last recommendation for the robot. The wheels on the robot are omni-type which gives the robot the potential to strafe, which means the robot could move in the X-direction (All directions) without turning. Unfortunately, the robot was not programmed to move in this manner. Strafing with this robot could be a better way to get from point A to point B quickly.

Conclusion

The purpose of the IEEE region 5 robotics competition is to draw attention to the increasing amounts of litter affecting the world's environment. This research project in the field of using autonomous machines to reduce the volume of litter that affects the environment helps to create both environmental awareness and an understanding of how such a machine can be created. The design phases of creating an autonomous machine for environmental cleanup require great consideration in the areas of desired operation, electrical design, mechanical design, system programming, environment of operation, cost, budget, availability of parts, quality of parts, timeline, feasibility, and etc. If any of these areas are neglected during the design process it is likely that there will be flaws in the final product and it will not perform properly. During the design process of the IEEE region 5 robotics competition robot the quality of the Lynxmotion Smart Servo Motor was overlooked which led to many delays in the timeline of testing and prototype phases. While the IEEE region 5 robotics competition robot did meet competition criteria it was unable to compete due to the cancelation of the competition. Since the cancelation of the competition the robot has been converted to operate with a remote control rather than autonomously for future use by the UT Tyler department of electrical engineering.

Appendices

Poster

IEEE Robotics Competition

TYL 9: ROBOTECH

Members: Richard Siu, Joshua Luker, Michelle Tamdjo, Wyatt Snare, and Josue Rios

Sponsor: University of Texas at Tyler Department of Electrical Engineering

Advisor Dr. Prabha Sundaravadivel

Introduction

The IEEE Region 5 organizes an annual robotics competition open to schools in the region. The overall goal is to design and build an autonomous robot that has the ability to identify and pick up objects to place them into their respective bins. Objects placed in the correct bins within the two-minute time limit will score points and the team with the most point wins the competition.

The list below are rules and design constraints that must be followed:

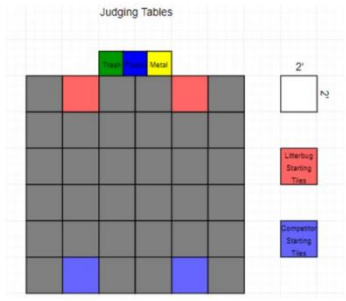
- Robot must be within 24"x24"x24"
- Must have a start and stop button
- Must weigh less than 25lbs
- Must not separate into multiple parts
- No hazardous materials

The Playing Field

The field will be 12' X 12' square with an 11" high border wall around the arena. The arena will be made up of 20 grey tiles, 2 red tiles, and 2 blue tiles. The bins are placed outside the area which will be green, blue, and yellow. Each robot must start on a blue tile opposite of a red tile with Litterbug, which is a competition owned robot that will dispense different objects throughout the round and must be stopped by sending different RF signals using FSK, GFSK, and MSK modulation schemes.

The types of objects on the field will be:

- Aluminum soft drink cans
- Clear plastic water bottles
- Paper trays (commonly used for serving concession food)
- Snack size chip bags

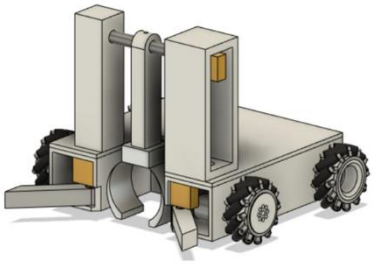


Mechanical Design

Our design was created with a single servo-based arm and servo-actuated claw to minimize movement of trash items to a single axis and ensure precise item capture.

Other mechanical components consist of:

- Four drive wheels
- Four drive motors
- Two servo actuated flippers





Hardware Design

The brain of this design relies on a Raspberry Pi and borrows the processing power of two Arduino Mega boards. Each Arduino board serves a specific purpose: one is responsible for processing movement, and the other is responsible for processing all servo-based operations. The Arduino responsible for movement is connected via I2C to a driver shield while the Arduino responsible for servo-based operations is connected via I2C to an LSS smart servo adapter.

Other hardware items connected to the Raspberry Pi and Arduinos:

- LSS Smart Servo
- Raspberry Pi Camera
- Two power switches
- Code initializer switch
- Adafruit Feather 32u4 RFM96 LoRa Radio



Raspberry Pi 4



LSS Smart Servo Adapter



Arduino Mega 2560



Adafruit motor shield V2

Conclusions

All of our designs came together to create a functioning robot. However, due to unforeseen current events, we were not able to attend and compete.

Sign off Sheet

The University of Texas at TYLER COLLEGE OF ENGINEERING

Department of Mechanical Engineering
Phone: +1.903.566.7003
Fax: +1.903.566.7148
Uttyler.edu/engineering

Senior Design Project II

MENG 4216 / EENG 4315

Project RoboTech and Acceptance

Senior design team RoboTech, from University of Texas at Tyler College of Engineering, has worked with a sponsor: University of Texas at Tyler's Electrical Engineering Department to compete at the IEEE Region 5 Robotics competition in April as part of the senior students' academic work during the academic year of 2019-2020. This document includes the sign-off list and University of Texas at Tyler's Electrical Engineering Department final approval of the outcome of the robot delivered by RoboTech, based on the previously set and mutually approved scope and specifications of the project. An authorized signature indicates this approval.

Table 1: Sponsor Sign OFF list for Deliverables of the Senior Project RoboTech

Item No.	Description of Deliverable	Comments:	Sponsor Sign Off: Authorized Initials
Robot	Sort trash to the correct recycle bin in a two-minute timeframe	Supposed to be autonomous but changed to remote control due to cancellation of competition	
DC Motor X 4	101 RPM		
Raspberry Pi	The Brain of the robot		
Raspberry Pi Camera	Camera will be used for machine learning		
Arduino Mega X 2	Operate motors and Arm		
Claw	Must be able to pick up all sizes of trash in competition		

Smart Servo	Must be able to strong enough to withstand a moment of 18,046 kg-cm	
Communication Device	RF Development Tools Adafruit Feather 32u4 RFM96 LoRa Radio – 433 MHz	Rules Official Changed 2/27/2020
Smart Servo Board	Sits on Arduino board to connect Smart Servo	
3D printed body	Printed entire body and arm using PLA	
Remote Controlled	This is added so the robot will be controlled though a remote	Changed 3/26/20 Due to Competition cancelling
Production Model Documentation	Technical Documentation	
B.O.M.	Bill of Materials	
I.O.M.	Installation and Operations Manual	
Electrical Schematic	Electrical Layout	
Mechanical Drawings	Mechanical Layout	
Testing	Testing of system	
Trial Competition	Travel to Oklahoma To test and observe the litter bug	
Competition on April 4th	This is the final test for the entire system	Competition cancelled due to Covid-19

The following signatures indicate that University of Texas at Tyler's Electrical Engineering Department approves of the completion of this project and deliverables stated above which included all sponsor changes from original design and any consequent changes in the project, as mutually approved by all signing parties.
Also indicated is the acknowledgement by University of Texas at Tyler's Electrical Engineering Department of reception of the final product of the project.

Dr. Hassan El-Kishky	Date
Dr. Mukul Shirvaikar	Date
Dr. Prabha Sundaravadivel	Date
Josue Rios	Date

Economic feasibility (Cost of prototype vs. Mass production cost)

Qty.	Component list Parts Name	Prototype Cost	Mass Production Cost
2	Arduino	$37.98	$0
1	Raspberry Pi	$22.99	$0
1	Claw	$48.72	$15.78
4	Wheels	$135.95	$100
4	DC motors	$99.96	$50.45
1	Raspberry Pi Camera	$18.99	$11.99
1	High-torque servo	$109.95	$47.95
6	Traditional Servo	$77.94	$40.60
4	Driver Circuits	$5.59	$1.99
1	Start Button	$5.18	$1.99
1	Stop Button	$5.18	$1.99
1	Battery	$19.99	$7.56
3	Electromagnets	$47.67	$20.78
1	Light Receiver	$9.99	$3.55
1	3-D Printed Base	$0.00	$4000
2	3-D Printed Flippers	$0.00	$4000
1	3-D Printed Arm	$0.00	$2000
25	Assembly	$0.00	$2500
1	FFC certification	$0	$2500
	UN 38.3	$0	$1000
	Total	$646.08	$16,338

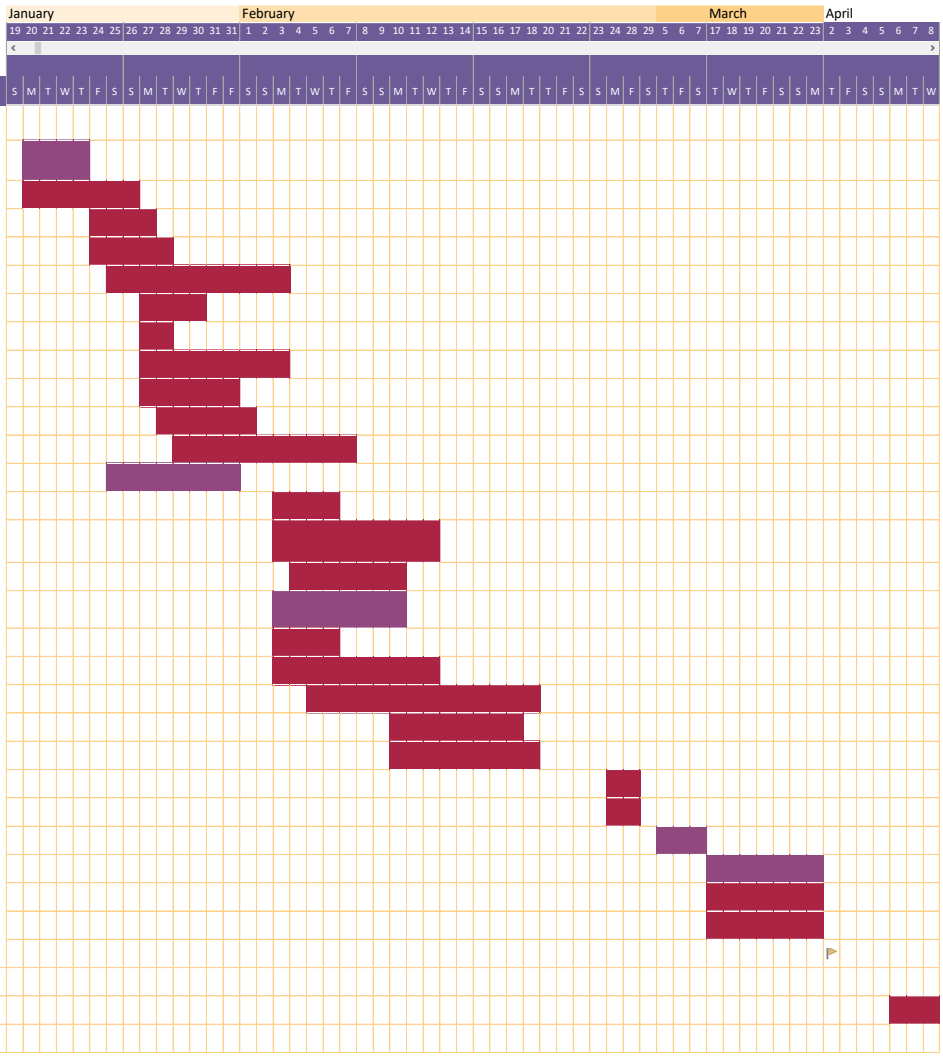
Timeline and milestones with dates

Robot Competition

University of Texas at Tyler
Team members: Richard Sij
Wyatt Snare , Joshua Luker
Michelle Tamdjo, Josue Rios

Project Start Date: 1/13/2020
Scrolling Increment: 6

Legend: On Track | Low Risk | Med Risk | High Risk | Unassigned

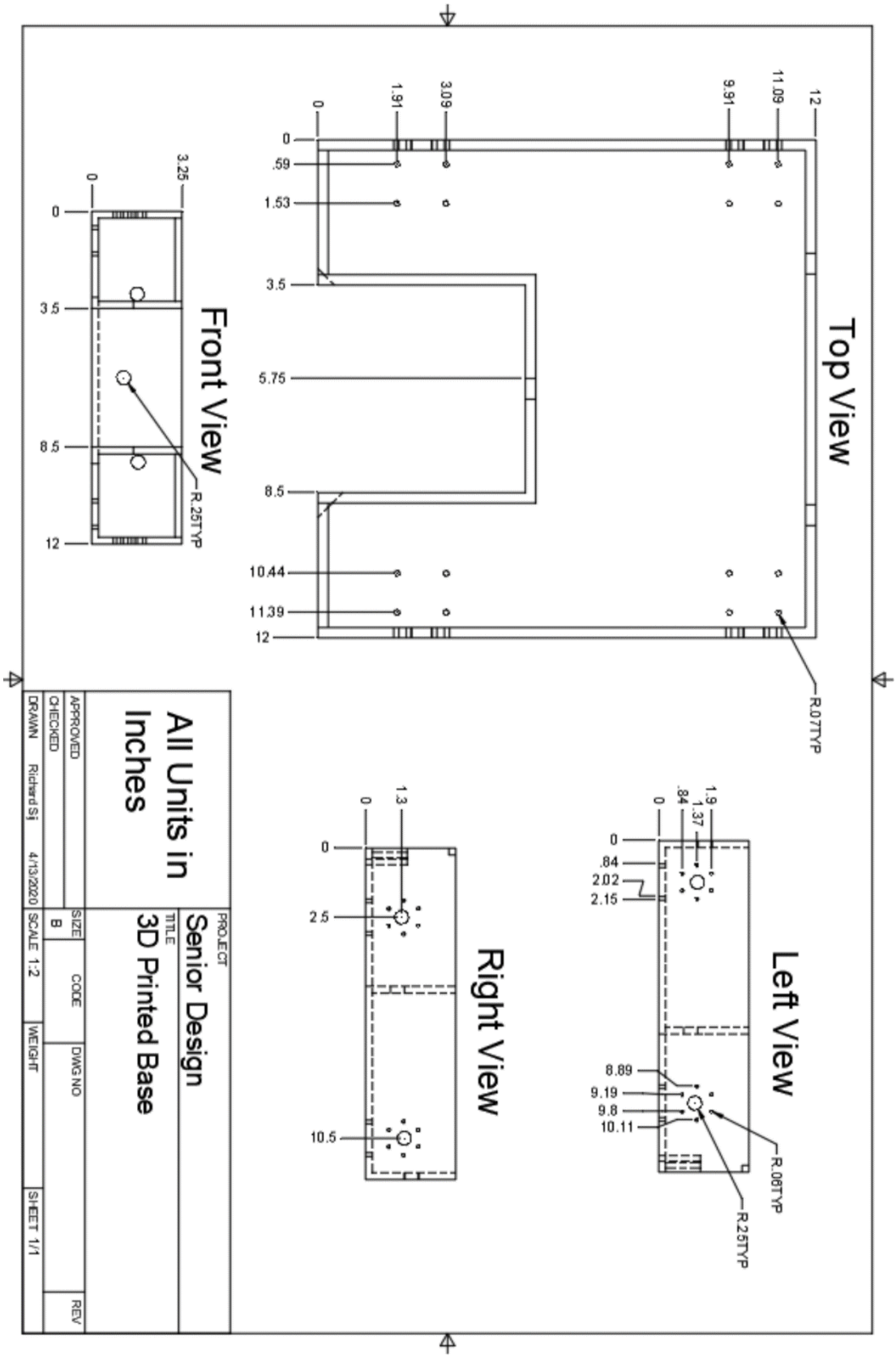


Task Name	Classification	Assigned To	Progress	Start	No. Days
Project Assignments/ Robot					
Week 1 assigment due	Med Risk	Entire team	100%	1/20/2020	4
Base of the body construction	High Risk	Richard,Michelle	100%	1/20/2020	7
Mount motor and pi with power	High Risk	Richard	100%	1/24/2020	4
Mount Claw system to base	High Risk	Richard, Michelle	100%	1/24/2020	5
Mount Trays/Wiring diagram	High Risk	Richard, Michelle, Josh, Wyatt	100%	1/25/2020	10
Week 2 assigment due	High Risk	Entire team	100%	1/27/2020	4
Power of all the robot components	High Risk	Entire team	100%	1/27/2020	2
Coding Vsion of the robot	High Risk	Wyatt, Josue	100%	1/27/2020	8
Adding the base	High Risk	Josh	100%	1/27/2020	5
Coding Claw	High Risk	Wyatt	100%	1/28/2020	5
Coding Trays	High Risk	Michelle	100%	1/29/2020	10
Mount motor and pi with power	Med Risk	Richard	100%	1/25/2020	7
Week 3 assigment due	High Risk	Entire team	100%	2/3/2020	4
Compile a test set of images (plastic, paper, alumminum)	High Risk	Wyatt, Michelle	100%	2/3/2020	10
Receive video and feed and run prototyte detector	High Risk	Richarch Sij	100%	2/4/2020	7
Build object locator	Med Risk	Entire team	100%	2/3/2020	8
Image detection complete	High Risk	Entire team	100%	2/3/2020	4
Body parts attachment	High Risk	Entire team	100%	2/3/2020	10
Operators testing and evaluation	High Risk	Entire team	100%	2/5/2020	15
Week 4 assigment due	High Risk	Entire team	100%	2/10/2020	8
Testing for robot ability to pick up different type of trash	High Risk	Entire team	100%	2/10/2020	10
Week 6 assigment due (No assigment week 5	High Risk	Entire team	100%	2/24/2020	5
Testing all parts/ fixing errors	High Risk	entire team	100%	2/24/2020	5
Week 7&8 No assigment due	Med Risk	Joshua, Richard	100%	3/2/2020	10
Test the robot on the mini floor display for the interface	Med Risk	Entire team	100%	3/9/2020	15
Week 9 assigment due	High Risk	Entire Team	100%	3/16/2020	8
Trial errors testing	High Risk	Entire team	100%	3/16/2020	10
Week 11 assigment due (No assigment week 10)	Milestone	Entire team	100%	3/30/2020	4
Robot done	High Risk	Entire team	100%	3/30/2020	3
Week 12&13 assigment due	High Risk	Entire team	100%	4/6/2020	14
Week 14 assigment due	Med Risk	Entire team	100%	4/27/2020	7

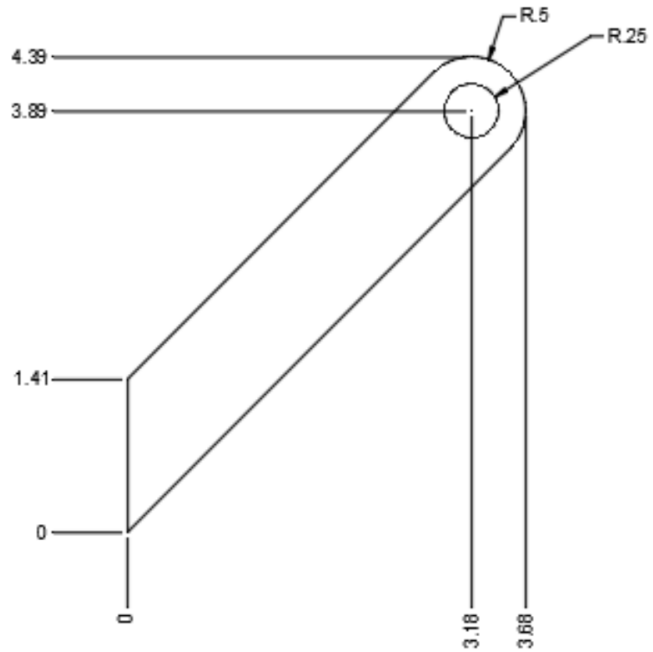
ARCI Chart

Tasks	Richard	Wyatt	Josh	Michelle	Josue
Design of the entire Robot	•	•	•	•	•
Drawing Dimensions	•			•	
Structural Analysis	•			•	
Camera /Image testing and calibration		•		•	
Wiring connection/ verification of physical components.			•		•
Documentation/reports	•	•	•	•	•
3D Printing	•			•	
Build the base of the robot	•		•		•
Build the entire body	•			•	
Sensors testing /calibration		•	•		•
Motor testing	•		•	•	
Arduino Mega testing/ calibration/coding		•			
Raspberry Pi testing /calibration/coding				•	•
Arduino & Raspberry Pi coding		•	•	•	•
Overall programing/testing/verification	•	•	•	•	•
Testing base movement	•	•	•	•	•
Research on all sub-systems	•	•	•	•	•
Movement Debugging			•		•
Smart Servo Code				•	
Remote Control Coding		•	•		•
Manual				•	

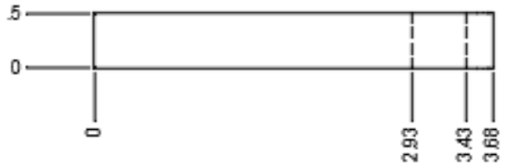
Mechanical Drawings



Top View



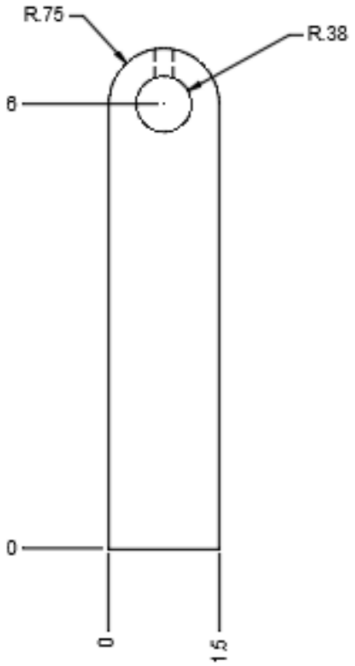
Side View



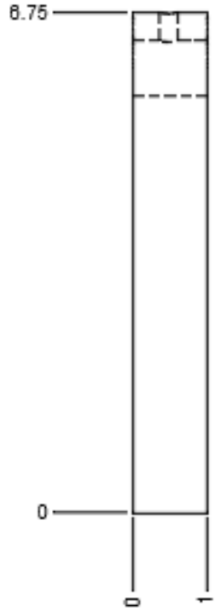
All Units in Inches QTY= 2	PROJECT: Senior Design				
	TITLE: 3D Printed Bumpers				
APPROVED	SIZE: B	CODE	DWG NO		REV
CHECKED					
DRAWN Richard Sij 4/13/2020	SCALE 1:1	WEIGHT		SHEET 1/1	

34

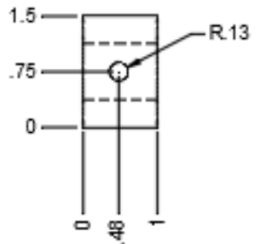
Side View



Front View

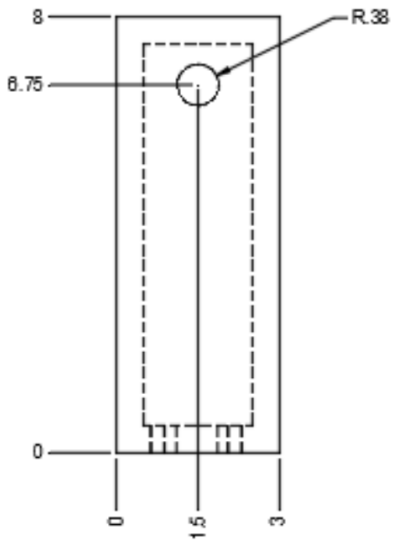


Top View

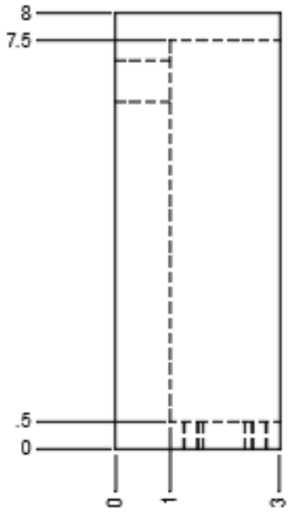


All Units in Inches	PROJECT: Senior Design				
	TITLE: 3D Printed Claw Arm				
APPROVED	SIZE: B	CODE	DWG NO		REV
CHECKED					
DRAWN: Richard Sij 4/13/2020	SCALE: 1:1.5	WEIGHT	SHEET: 1/1		

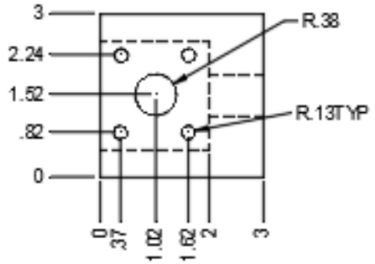
Left Side



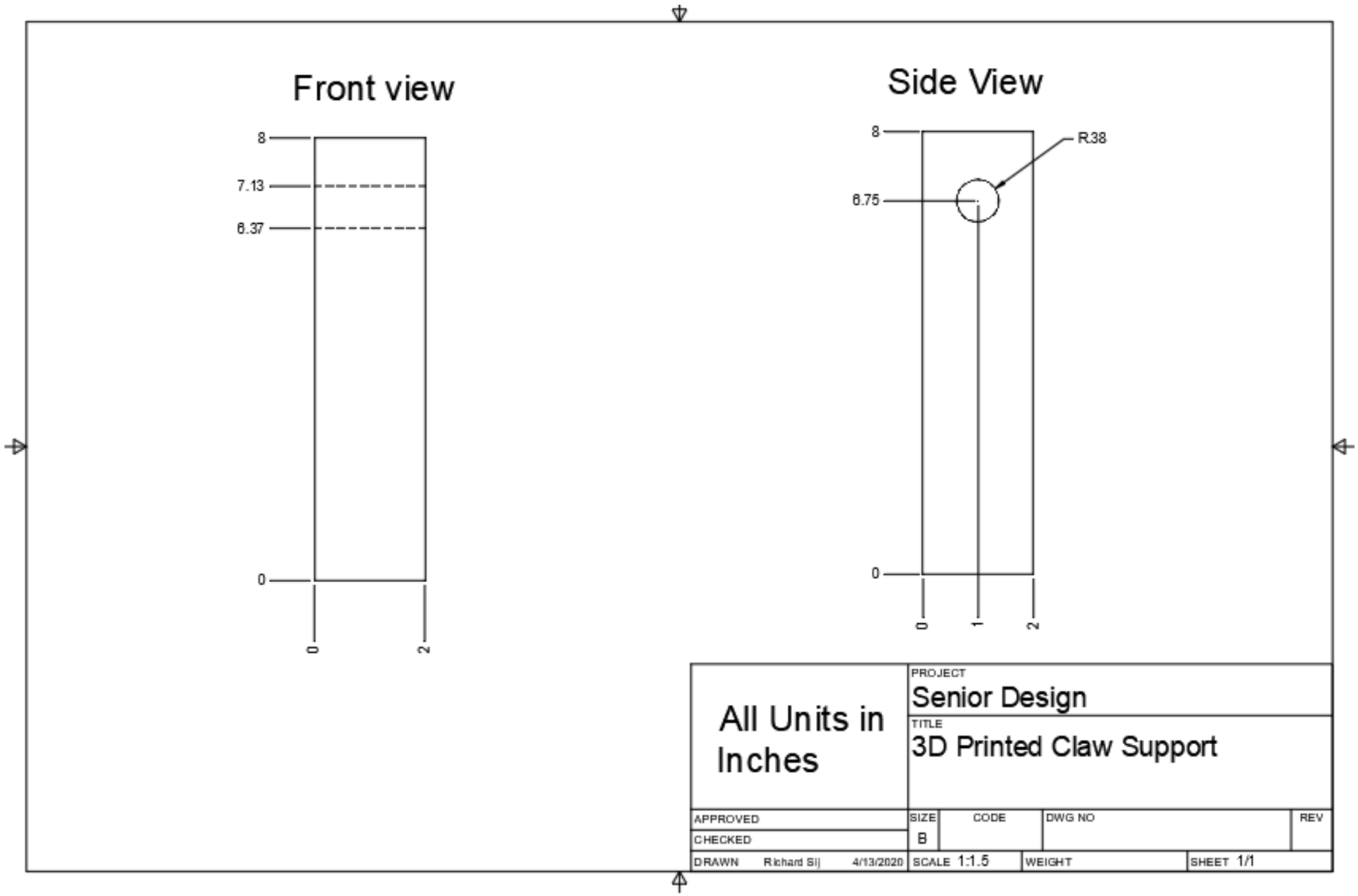
Front Side



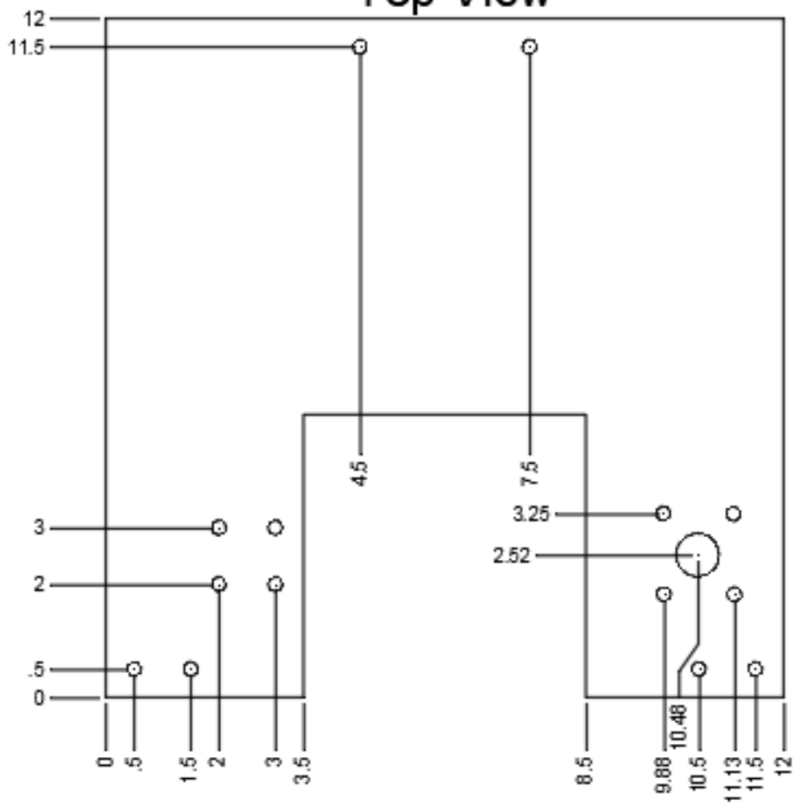
Bottom Side



All Units in Inches	PROJECT: Senior Design				
	TITLE: 3D Printed Power Claw Arm Support				
APPROVED	SIZE: B	CODE	DWG NO		REV
CHECKED					
DRAWN Richard Sij 4/13/2020	SCALE 1:3	WEIGHT	SHEET 1/1		


Front view
8
7.13
6.37
0
0
2
Side View
8
6.75
0
R.38
0
1
2
All Units in Inches
PROJECT
Senior Design
TITLE
3D Printed Claw Support
APPROVED
CHECKED
DRAWN Richard Sij 4/13/2020
SIZE
B
CODE
DWG NO
REV
SCALE 1:1.5
WEIGHT
SHEET 1/1

Top View



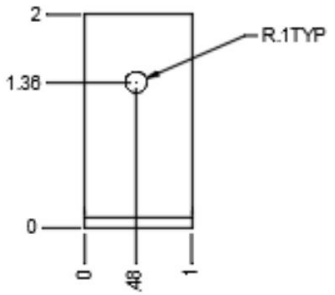
All Units in Inches

PROJECT: Senior Design

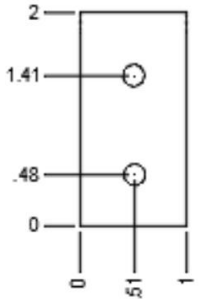
TITLE: Lazer Cut Plexiglass Lid .25in thick

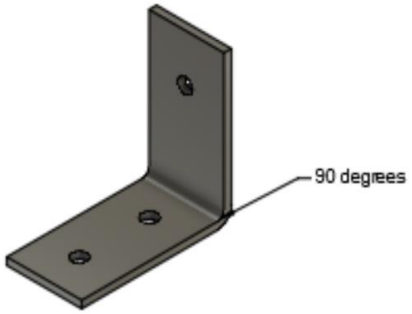
APPROVED		SIZE	CODE	DWG NO	REV
CHECKED		B			
DRAWN Richard Sij 4/13/2020	SCALE 1:2	WEIGHT		SHEET 1/1	

Front



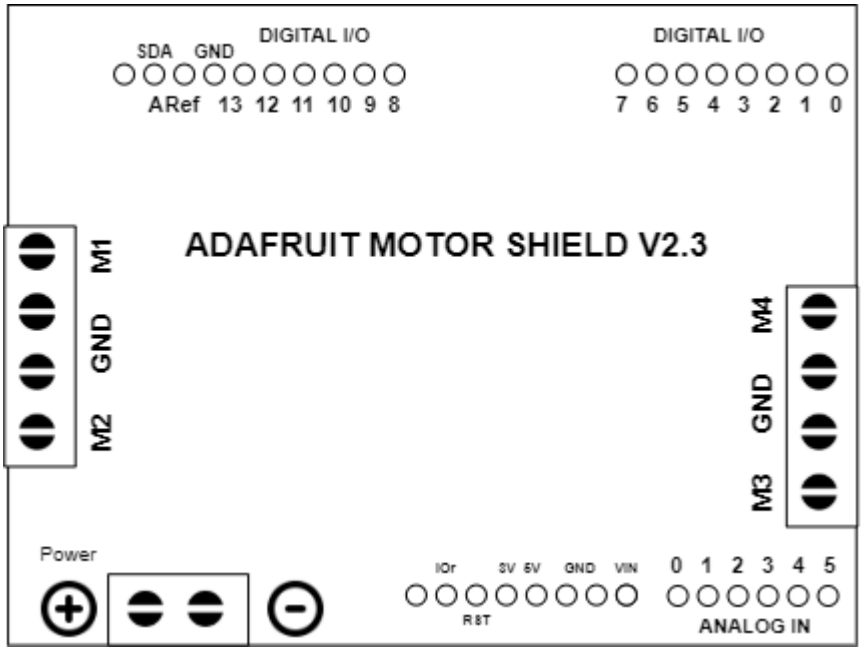
Bottom



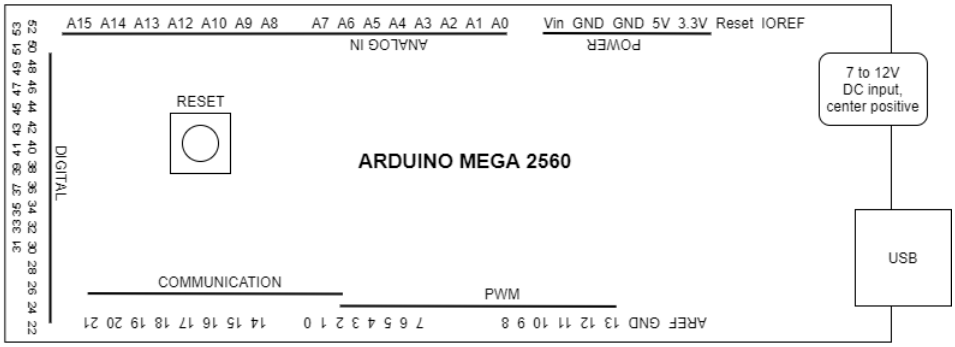


All Units is Inches	PROJECT: Senior Design				
	TITLE: Lid Brackets				
APPROVED	SIZE: B	CODE	DWG NO		REV
CHECKED					
DRAWN Richard Sj 4/18/2020	SCALE 1:1	WEIGHT		SHEET 1/1	

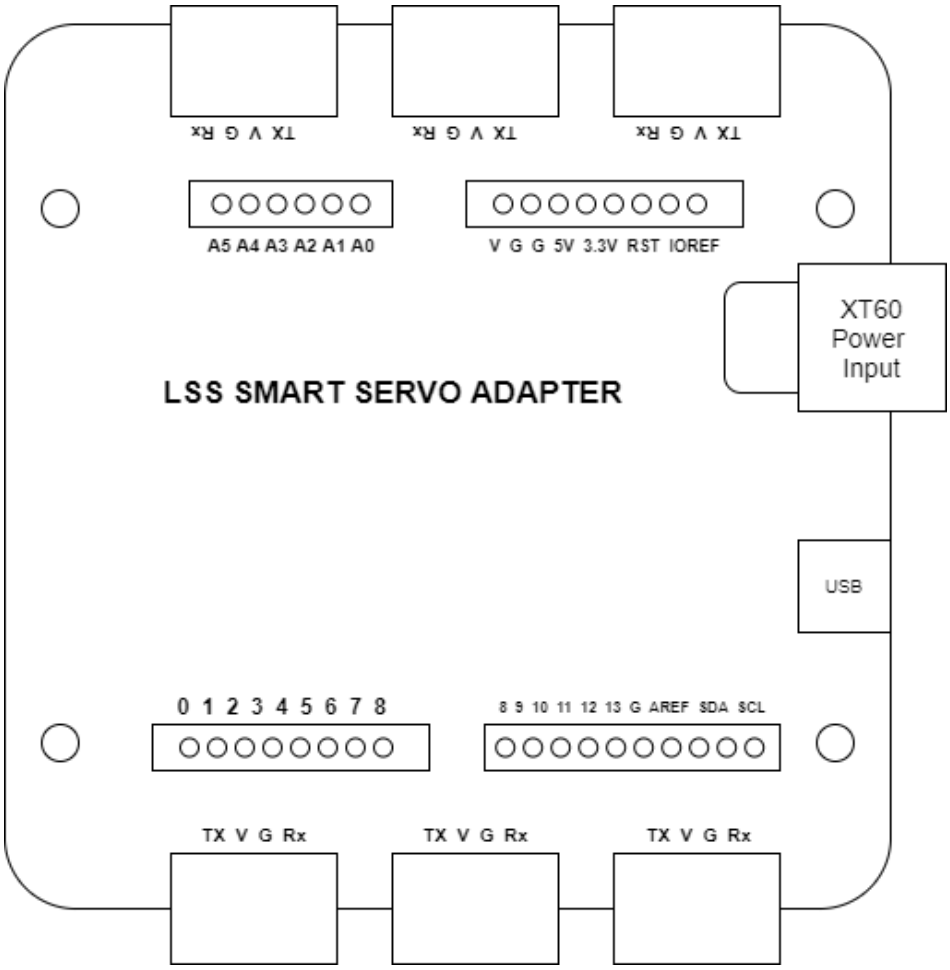
Electric Circuits Diagrams



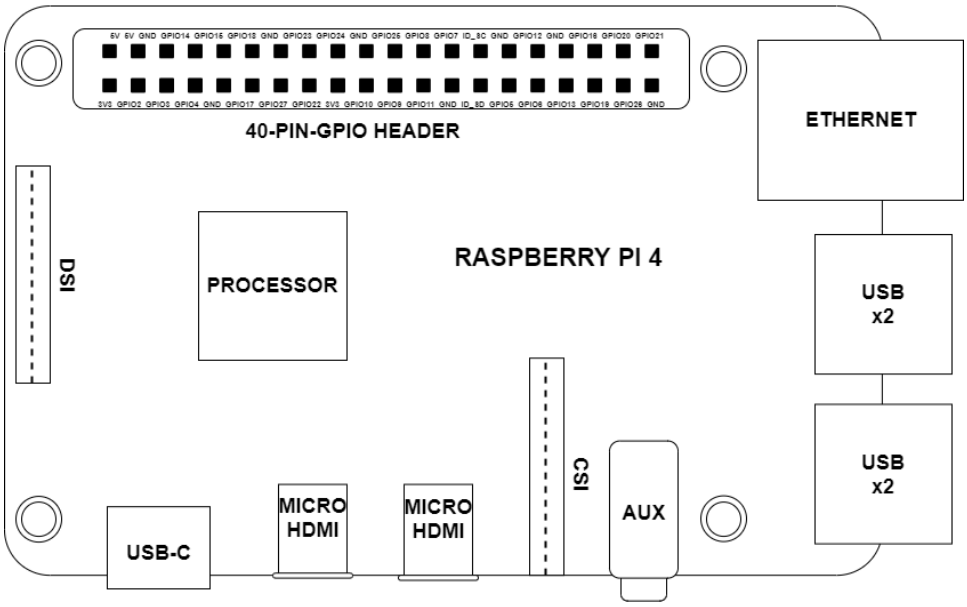
Adafruit Motor Shield Schematic Diagram



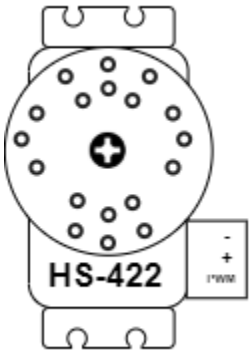
Arduino Mega 2560 Schematic Diagram



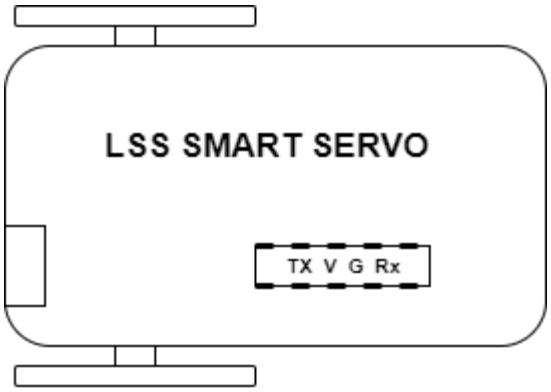
LSS Smart Servo Adapter Schematic Diagram



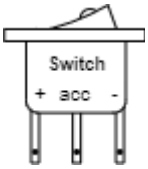
Raspberry Pi 4 Schematic Diagram



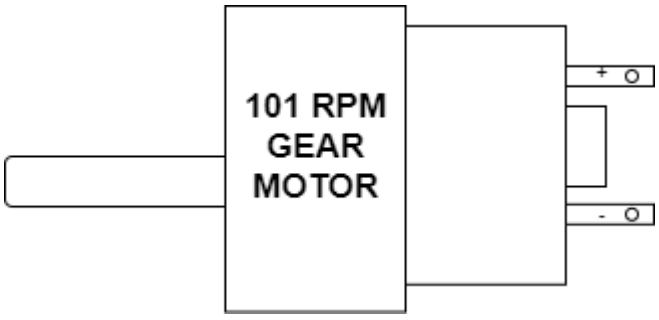
HS-422 Servo Schematic Diagram



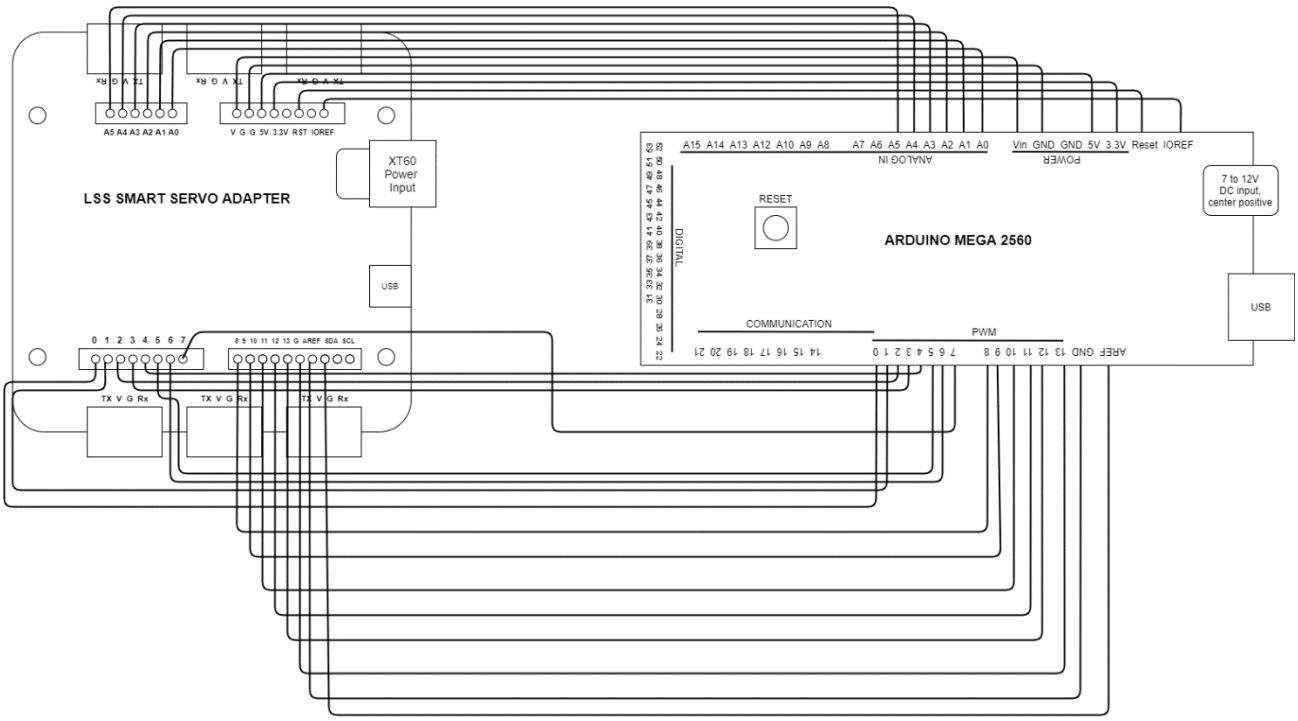
LSS Smart Servo Schematic Diagram



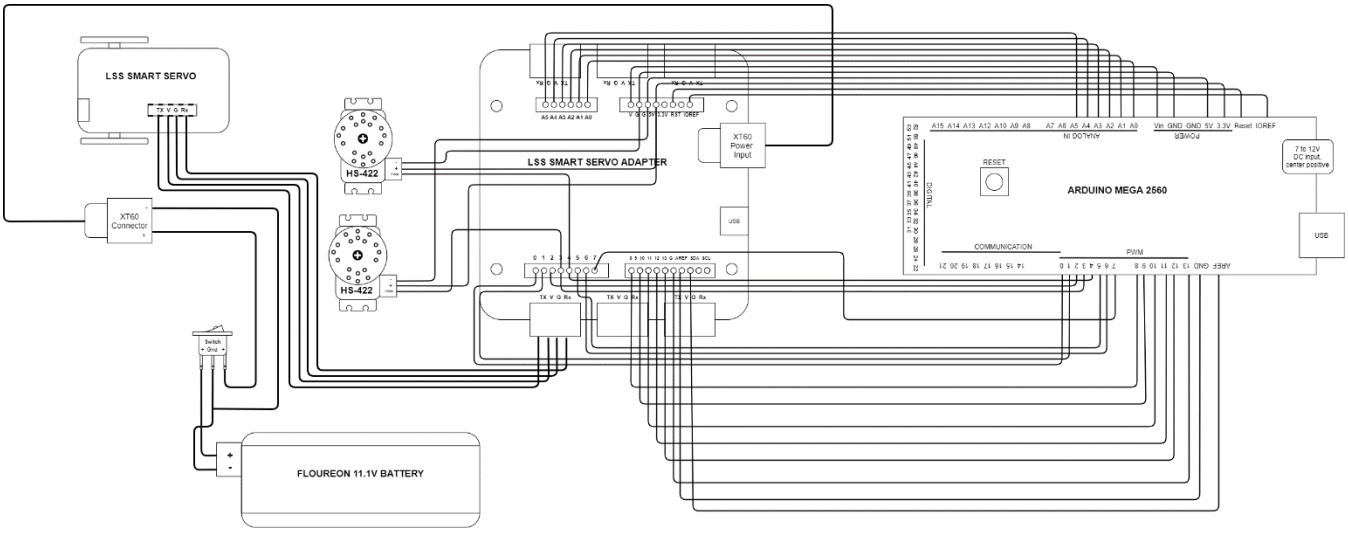
Rocker Switch Schematic Diagram



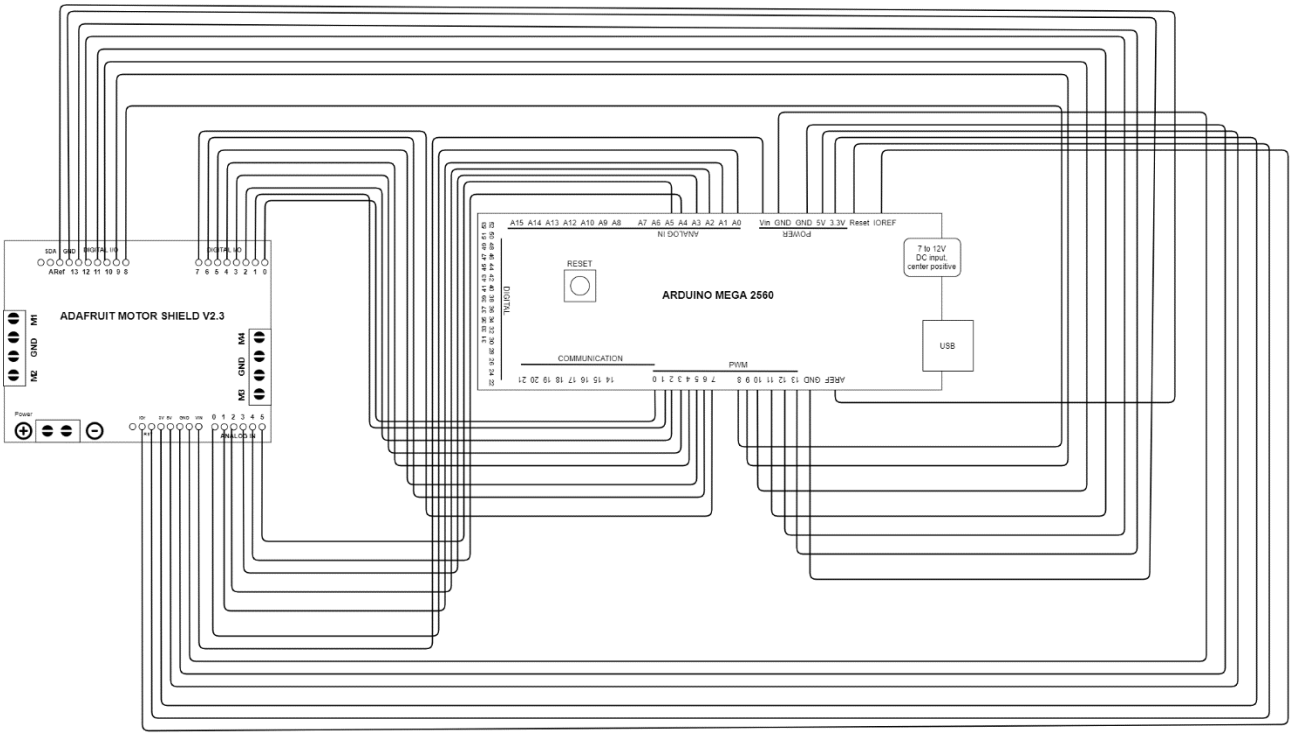
101 RPM Gear Motor Schematic Diagram



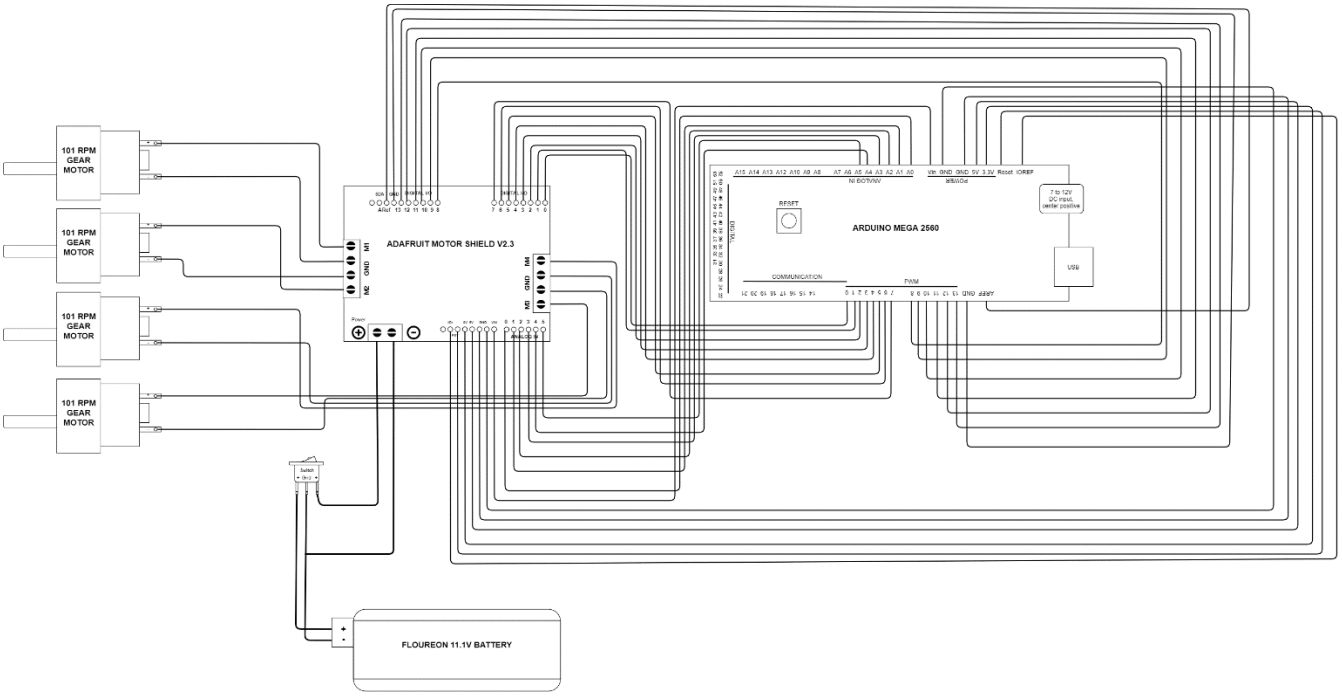
LSS Smart Servo Adapter Connected to Arduino Mega 2560



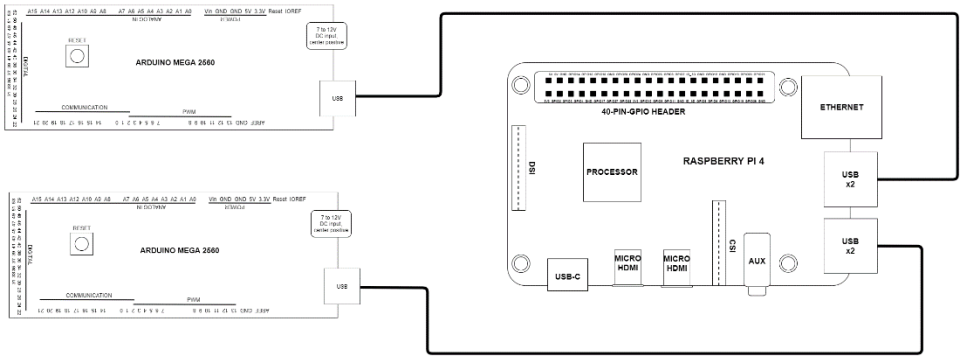
Powered LSS Smart Servo Adapter Connected to Arduino Mega 2560 and Servos



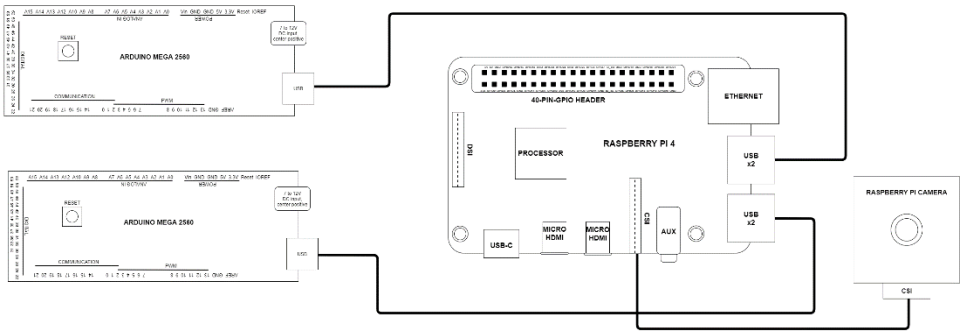
Adafruit Motor Shield V2.3 Connected to Arduino Mega 2560



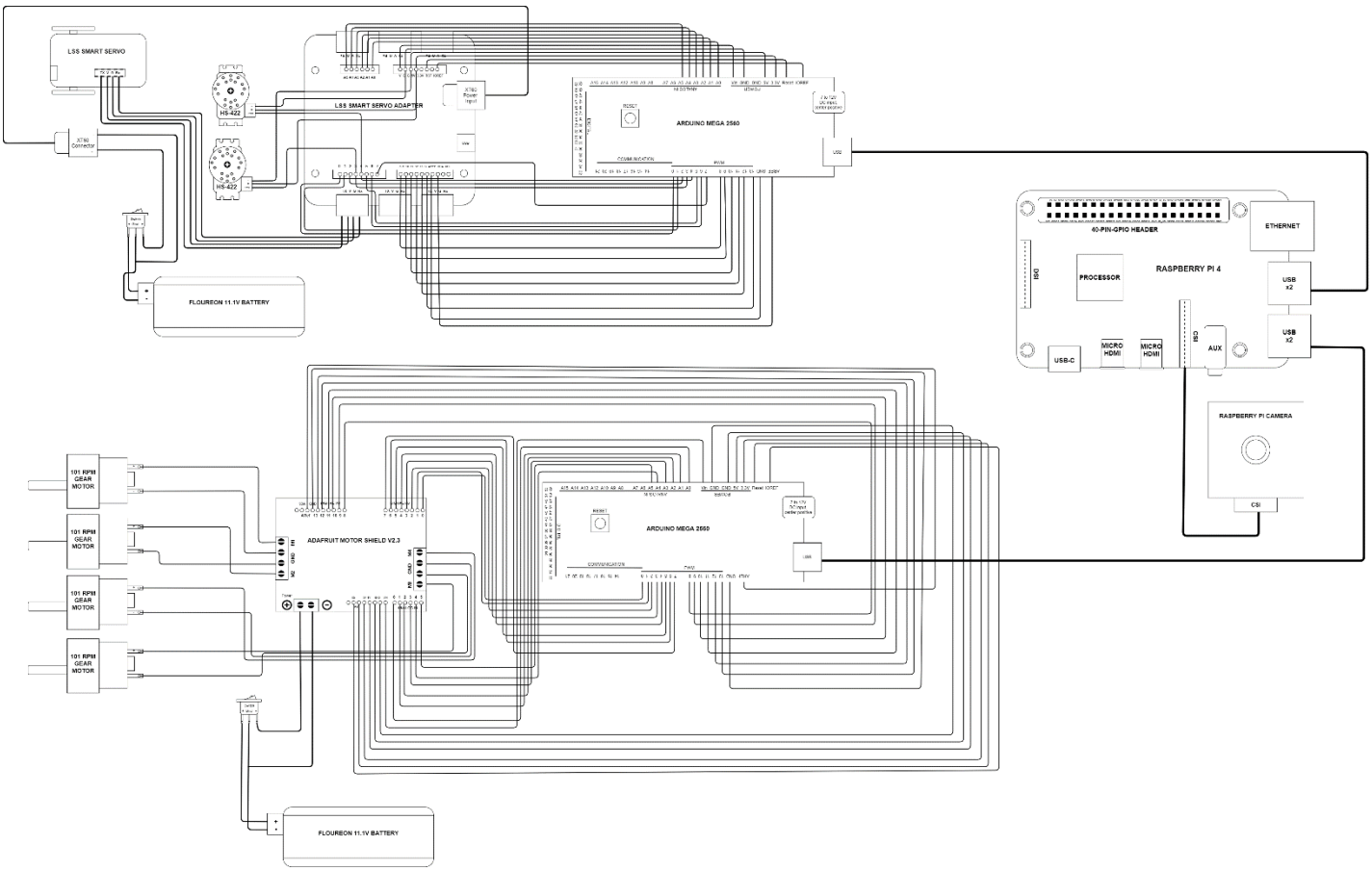
Powered Adafruit Motor Shield V2.3 Connected to Arduino Mega 2560 and 101 RPM Motors



Raspberry Pi 4 Connected to Two Arduino Mega 2560



Raspberry Pi 4 Connected to Two Arduino Mega 2560 and Raspberry Pi Camera



System Electric Circuit Diagram

Any Code or Program Listing

Folder: Ard

File: Ard

```
#include <Servo.h>
#include "Motion.h"
#include "Servos.h"
#define MAX_BUFFER_LENGTH 2

const int NAME = 1;
String active = "pi";
char buffer[MAX_BUFFER_LENGTH]; //instruction buffer

//Attach DC motors
Motion motors(
  3,          //left front wheel
  4,          //left back wheel
  2,          //right front wheel
  1           //right back wheel
);

//Attach Servos for claw and flippers
Servos servos(
  9,          //claw pin
  4,          //left flipper
  3           //right flipper
);

void setup() {
  Serial.begin(9600);
```

```
  motors.begin();
  servos.begin();
  servos.openClaw();
  servos.openFlippers();

  //wait for Pi to configure COMs
  while(active == "pi"){
    Serial.println(NAME);
    delay(1000);
    if(Serial.available() > 0){
      active = "ard";
    }
  }

  // Serial.println("Setup Complete.");
}

void loop() {
  // testServos();
  // testMotion();
  recieveData();
}

//received data from Raspberry Pi
void recieveData(){
  int len = 0;
  if (Serial.available() >= 0){
    while(Serial.available()>len){//is a char available?
      len = Serial.available();
      delay(5); //delay 5ms for new characters to arrive
```

```
    if(len>10){
      break;
    }
   }
  }
  String instr = ""; //instruction
  if(len>0){
   Serial.readBytes(buffer, len); //get the characters
   Serial.println(len);
   for(int i = 0; i<len; i++){ //exclude terminal character
    instr += String(int(buffer[i])-48);
   }
   Serial.println(instr);
   Serial.println(instr.toInt());
   runCommand(instr.toInt());
  }
}

void runCommand(int action){
  switch(action%48){
   case 0:          //STOP MOTORS
    motors.stop();
    break;
   case 1:          //MOVE FORWARD
    motors.forward();
    break;
   case 2:          //MOVE BACKWARD
    motors.backward();
    break;
   case 3:          //ROTATE RIGHT
```

```
      motors.turnRight();
      break;
    case 4:             //ROTATE LEFT
      motors.turnLeft();
      break;
    case 5:             //CLOSE FLIPPERS
      servos.closeFlippers();
      break;
    case 6:             //OPEN FLIPPERS
      servos.openFlippers();
      break;
    case 7:             //CLOSE CLAW
      servos.closeClaw();
      break;
    case 8:             //OPEN CLAW
      servos.openClaw();
      break;
    case 9:             //RAISE ARM
      servos.raiseArm();
      break;
    case 10:            //LOWER ARM
      servos.lowerArm();
      break;
  }
}

void testMotion(){
  runCommand(1);
  delay(1000);
  runCommand(0);
```

```
  delay(1000);
  runCommand(2);
  delay(1000);
  runCommand(0);
  delay(1000);

  runCommand(3);
  delay(1000);
  runCommand(0);
  delay(1000);
  runCommand(4);
  delay(1000);
  runCommand(0);
  delay(1000);
}

void testServos(){
  servos.closeClaw();
  delay(2000);
  servos.openClaw();
  delay(2000);
  servos.closeFlippers();
  delay(2000);
  servos.openFlippers();
  delay(2000);
  servos.raiseArm();
  delay(2000);
  servos.lowerArm();
  delay(2000);
}
```

File: Motion.cpp

```
#include <Wire.h>
#include "Motion.h"
#include <Adafruit_MotorShield.h>

Motion::Motion(int lf,  int lb, int rf, int rb){
  _AFMS = Adafruit_MotorShield();

  //initialize motors
  //_lf - left front motor
  //_rf - right front motor
  //_rb - right back motor
  //_lb - left back motor
  _lf = _AFMS.getMotor(lf);
  _lb = _AFMS.getMotor(lb);
  _rf = _AFMS.getMotor(rf);
  _rb = _AFMS.getMotor(rb);
}

//stop ceases all motion of motors
Motion::stop(){
    _lf->run(RELEASE);
    _lb->run(RELEASE);
    _rf->run(RELEASE);
    _rb->run(RELEASE);
}

Motion::rotate(bool right){
    this->stop();
    if(right == true){
```

```
        _lf->run(FORWARD);
        _lb->run(FORWARD);
        _rf->run(BACKWARD);
        _rb->run(BACKWARD);
    }else{
        _lf->run(BACKWARD);
        _lb->run(BACKWARD);
        _rf->run(FORWARD);
        _rb->run(FORWARD);
    }
}

Motion::linear(bool forward){
    if(forward == true){
        _lf->run(FORWARD);
        _lb->run(FORWARD);
        _rf->run(FORWARD);
        _rb->run(FORWARD);
    }else{
        _lf->run(BACKWARD);
        _lb->run(BACKWARD);
        _rf->run(BACKWARD);
        _rb->run(BACKWARD);
    }
}

Motion::forward(){
    this->linear(true);
}
```

```
Motion::backward(){
   this->linear(false);
}

Motion::turnRight(){
   this->rotate(true);
}

Motion::turnLeft(){
   this->rotate(false);
}

Motion::begin(){
   int speed = 200;
   _AFMS.begin();
   _lf->setSpeed(speed);
   _lb->setSpeed(speed);
   _rf->setSpeed(speed);
   _rb->setSpeed(speed);
   this->stop();
}
```

File: Motion.h

```
/*
   Initializes motors for movement for V.I.C.T.O.R.
*/
#ifndef Motion_h
#define Motion_h
#include <Adafruit_MotorShield.h>

class Motion{
```

```
  public:
    Motion(int lf,  int lb, int rf, int rb);
    stop();
    begin();
    forward();
    backward();
    turnLeft();
    turnRight();
  private:
    Adafruit_MotorShield _AFMS;
    Adafruit_DCMotor *_lf, *_rf, *_rb, *_lb;
    rotate(bool right);
    linear(bool forward);
};
#endif
```

File: Servos.cpp

```
#include <Servo.h>
#include "Servos.h"
#include <Arduino.h>
#include <LSS.h>

// ID set to default LSS ID = 0
#define LSS_ID              (0)
#define LSS_BAUD            (LSS_DefaultBaud)

Servos::Servos(int clawPin, int leftFlipper, int rightFlipper){
  isClawClosed      = true;
  isFlippersClosed  = true;
  isArmLowered      = true;
```

```
  _openClawAngle     = 135;
  _closeClawAngle    = 180;
  _openFlipperAngle  = 50;
  _closeFlipperAngle =  1;

  _clawPin      = clawPin;
  _leftFlipper  = leftFlipper;
  _rightFlipper = rightFlipper;
}

Servos::raiseArm(){
  if(isArmLowered){
    _arm.moveT(700, 2000);
    delay(2050);
    isArmLowered = false;
  }
}

Servos::lowerArm(){
  if(!isArmLowered){
    _arm.moveT(-700, 2000);
    delay(2050);
    isArmLowered = true;
  }
}

Servos::openClaw(){
  if(isClawClosed){
    for(int pos = _closeClawAngle; pos >= _openClawAngle; pos -= 1) {
```

```
    _claw.write(pos);
    delay(20);
   }
  }
  isClawClosed = false;
}

Servos::closeClaw(){
  if(!isClawClosed){
   for (int pos = _openClawAngle; pos <= _closeClawAngle; pos += 1) {
    _claw.write(pos);
    delay(20);
   }
  }
  isClawClosed = true;
}

Servos::openFlippers(){
  int opened = _openFlipperAngle;
  int closed = _closeFlipperAngle;

  if(isFlippersClosed){
   for (int pos = closed; pos <= opened; pos += 1) {
    _lFlipper.write(pos);
    _rFlipper.write(max(opened-pos, 1));
    delay(20);
   }
  }
```

```
  isFlippersClosed = false;
}

Servos::closeFlippers(){
  int opened = _openFlipperAngle;
  int closed = _closeFlipperAngle;

  if(!isFlippersClosed){
    for (int pos = opened; pos >= closed; pos -= 1) {
      _lFlipper.write(pos);
      _rFlipper.write(opened-pos);
      delay(40);
    }
  }

  isFlippersClosed = true;
}

Servos::begin(){
  _claw.attach(_clawPin);
  _lFlipper.attach(_leftFlipper);
  _rFlipper.attach(_rightFlipper);

  //Initialize arm and move to 0deg
  _arm = LSS(LSS_ID);
  LSS::initBus(Serial1, LSS_BAUD);
  _arm.move(0);
  delay(2000);
}
```

File: Servos.h

```
/*
   Initializes servos for flippers and claw V.I.C.T.O.R.
*/
#ifndef Servos_h
#define Servos_h
#include <Servo.h>
#include <LSS.h>
class Servos{
  public:
    bool isClawClosed;
    bool isFlippersClosed;
    bool isArmLowered;

    Servos(int clawPin,  int leftFlipper, int rightFlipper);
    begin();
    raiseArm();
    lowerArm();
    openClaw();
    closeClaw();
    openFlippers();
    closeFlippers();
  private:
    LSS _arm;
    int _clawPin, _leftFlipper, _rightFlipper;
    Servo _claw, _lFlipper, _rFlipper;
    int _openClawAngle, _closeClawAngle; //claw closed and open angles
    int _openFlipperAngle, _closeFlipperAngle; //flippers closed and open angles
};
#endif
```

Folder: utils

File: State

```
 import os
#State class containing active arduinos and active running components
class StateManager():


  #Determine OS
  OS="linux"
  if os.name == "nt":
    OS = 'windows'


  def __init__(self, num_devices=2):
    self.nodes = {}                              #an arduino
    self.active = 'pi'                    #
    self.added_devices = {} #


    self.valid_ids = [str(i) for i in range(num_devices)]


  def add_node(self, arduino, device_name, baudrate=9600):
    _id = None
    while _id not in self.valid_ids:
      _id = self.getId(arduino)
    self.added_devices.update({_id:device_name})


    #add new node
    arduino.baudrate=baudrate
    newNode = {str(_id):arduino}
    self.nodes.update(newNode)
    return _id
```

```
 def getId(self, arduino):
  # return '1'
  '''gets message from arduino serial.Serial() input stream. interprets the message as an _id'''
  # print("Attempting to read line")
  read_serial=arduino.readline()
  # print("read serial",read_serial)
  _id = read_serial.decode('utf-8') #returns initial message from arduino

  if StateManager.OS == "windows":
    _id = _id[:-2]  #remove \r\n character
  else:
    _id = _id[:-2]  #remove \r\n character

  print('Got id', _id)
  return _id

 def __str__(self):
  return str(self.added_devices)
```

Folder: Test

File: Gamepad

```
import time
import inputs
from victor import Brain

brain = Brain(2)

print(inputs.devices.gamepads)

pads = inputs.devices.gamepads
```

```
if len(pads) == 0:
    raise Exception("Couldn't find any Gamepads!")

action_space = {
    'ABS_HAT0X 0':  "STOP",
    'ABS_HAT0X -1': "TURN_LEFT",       #LEft  D Pad
    'ABS_HAT0X 1':  "TURN_RIGHT",      #Right D Pad
    'ABS_HAT0Y 1':  "MOVE_BACKWARD",   #Down  D Pad
    'ABS_HAT0Y -1': "MOVE_FORWARD",    #Up    D Pad
    'ABS_HAT0Y 0':  "STOP",
    'BTN_TL 1':     "OPEN_CLAW",       #RIGHT BUMPER
    'BTN_TR 1':     "CLOSE_CLAW",      #LEFT BUMPER
    'BTN_SOUTH 1':  "CLOSE_FLIPPERS",  #A
    'BTN_EAST 1':   "OPEN_FLIPPERS",   #B
    'BTN_NORTH 1':  "RAISE_ARM",       #X
    'BTN_WEST 1':   "LOWER_ARM",       #Y
}

print("Ready to receive")
while True:
    events = inputs.get_gamepad()
    for event in events:
        # print(event.ev_type, event.code, event.state)
        key = event.code + ' ' + str(event.state)
        action = action_space.get(key)
        print('key', key, 'Action', action)
        if action is not None:
            ard = "0"
            if "ABS" not in key:
```

```
            ard = "1"
        brain.send_act(action, ard)
File: action_space
ACTION_SPACE = {
  'STOP' : 0,
  'MOVE_FORWARD' : 1,
  'MOVE_BACKWARD' : 2,
  'TURN_RIGHT' : 3,
  'TURN_LEFT' : 4,
  'CLOSE_FLIPPERS': 5,
  'OPEN_FLIPPERS': 6,
  'CLOSE_CLAW': 7,
  'OPEN_CLAW': 8,
  'RAISE_ARM': 9,
  'LOWER_ARM': 10,
}
```

Detailed design calculations, formulae derivations, and software extra analysis output

Torque Servo Calculations for RPM Gear Motor

No.	Specifications	Symbol	Unit	Formula	Result
1	Radius	R	ft.	Set Values	.21
2	Length	L	ft		0.42
3	Mu	μ	(not sure)		.72
4	Gear Ratio	GR	(not sure)		2x30
5	Gravity	g	ft/s^2		32.2
6	Mass	m	lb.		25
7	Force	F	N	$F = m * g$	805
8	Load	P	(Not sure)	$P = R * F$	7.5
9	Force (not sure where this comes from or what is the name of it. it was on the previous report)	Fj	N	$F_j = P * R$	1.56
10	Torque	T	Ft x lb. or Oz X in	$T= \frac{(\mu * F + mg) x L}{GR}$	0.1829/35.12

Torque Servo Calculations for RPM Gear Motor

No.	Specifications	Symbol	Unit	Formula	Result
1	Length	L	cm	Set Values	25.4
2	Gravity	g	cm/s^2		980
3	Mass	m	Kg		0.725
4	Moment at A/ Torque	M_A	Kg-cm	$M_A = L * m * g * \cos(90)$	18.046

Entire System Power

No.	Specifications	Symbol	Unit	Formula	Result
1	Driver Current	A	A	Set Values	0.036
2	Battery	B	mAh		1500
3	Time	t	min		2
4	Battery voltage	P_b	V		11.1
5	Driver voltage	P_d	V		0.5
6	Arduino voltage	P_A	V		5
7	Motor voltage	P_m	V		11.1
8	Arduino Input	A_I	inputs		0.24
9	Power Input	A_m	A		0.25
10	Maximum Amp	A_{max}	A	$A_{max}= \frac{B}{(t * 1000)}$	45.45
11	Total P_b	T_{Pb}	W	$T_{Pb}= P_b \times A_{max}$	504.95
12	Driver Power	T_{Pd}	W	$T_{Pd} = P_d \times A$	0.072
13	Arduino Power	T_{PA}	W	$T_{PA} = P_A \times A_I$	1.2
14	Motor Power	T_{Pm}	W	$T_{PA} = P_m \times A_m$	11.1
15	Total power	T	W	$T= T_{Pd} + T_{Pm} + T_{PA}$	12.73
16	Overall Power System	T_T	W	$T_T = T_{Pb} - T$	492.12

Design of experiments, experimental procedures, and results

- DC Motor shield

Testing the DC motor shield consisted of connecting it to the Arduino, applying the rated voltage and running code to ensure the correct voltage and current was being produced at the output terminals. Taking several measurements with a multimeter in order to confirm that the output from the motor shield was able to produce a constant and reliable source for the DC Motors to be connected to. Once this was confirmed and the DC Motors were tested and connected to the shields. After running code, the shield proved that it was producing the correct output in order to power the motors in any direction needed, along with the 25 lbs. weight limit.

- DC Motor

Testing the DC Motor was simple. Taking a dc battery and connecting it directly to the motor was the most efficient way of testing the max capabilities of the motor. Once connected we were able to measure the current draw and evaluate the pace at which the shaft spins. Once we tested each individual motor with the battery, we were able to attach all components to the base of the robot and wire the motors into the DC Motor Shield. With everything connected, a code was running in order to test functionality of forward and reverse motion. Once forward and reverse movement was tested with no external weight added we added 25 lbs. to the top of the base and tested movement again.

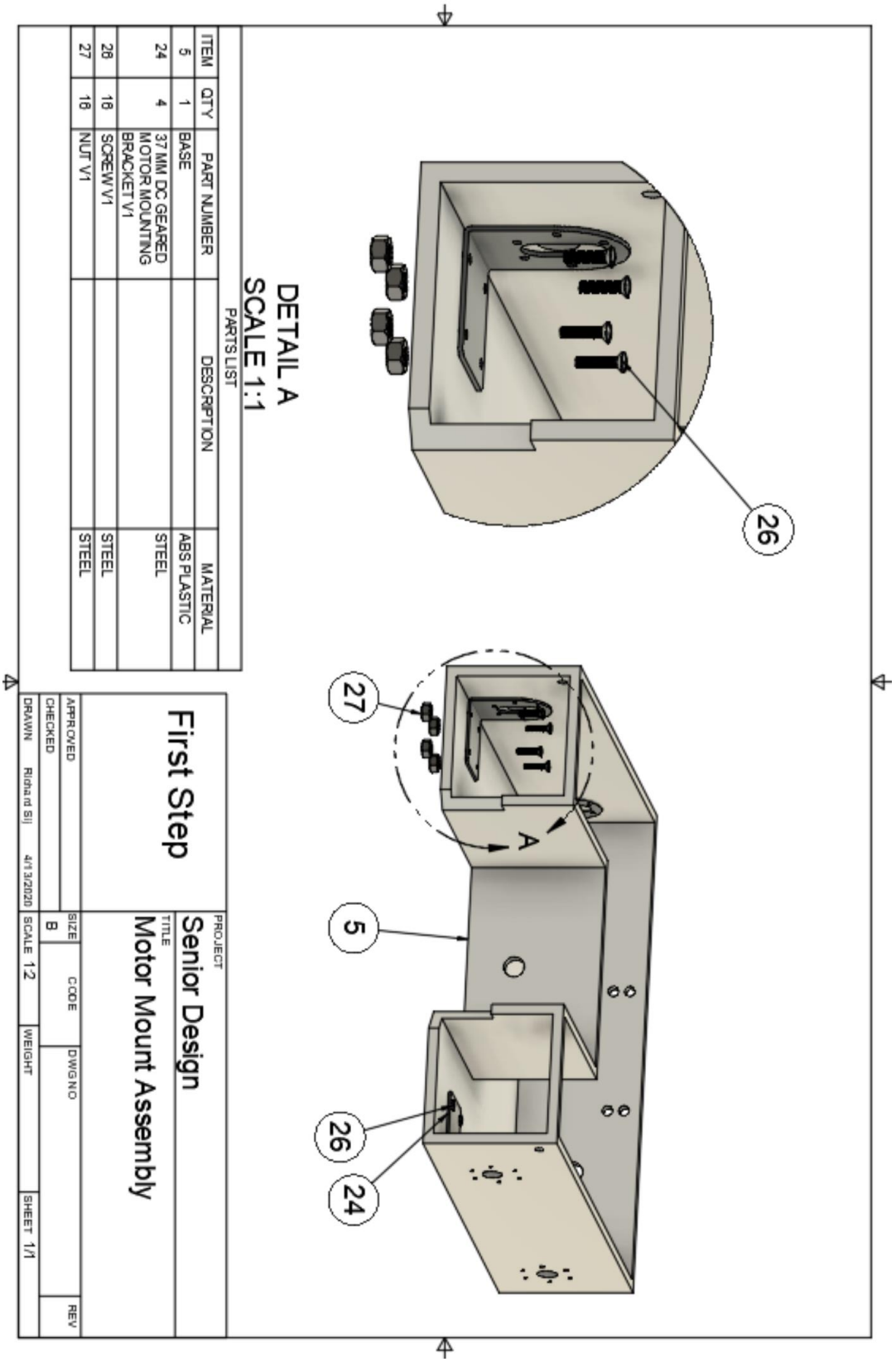
- Claw movement

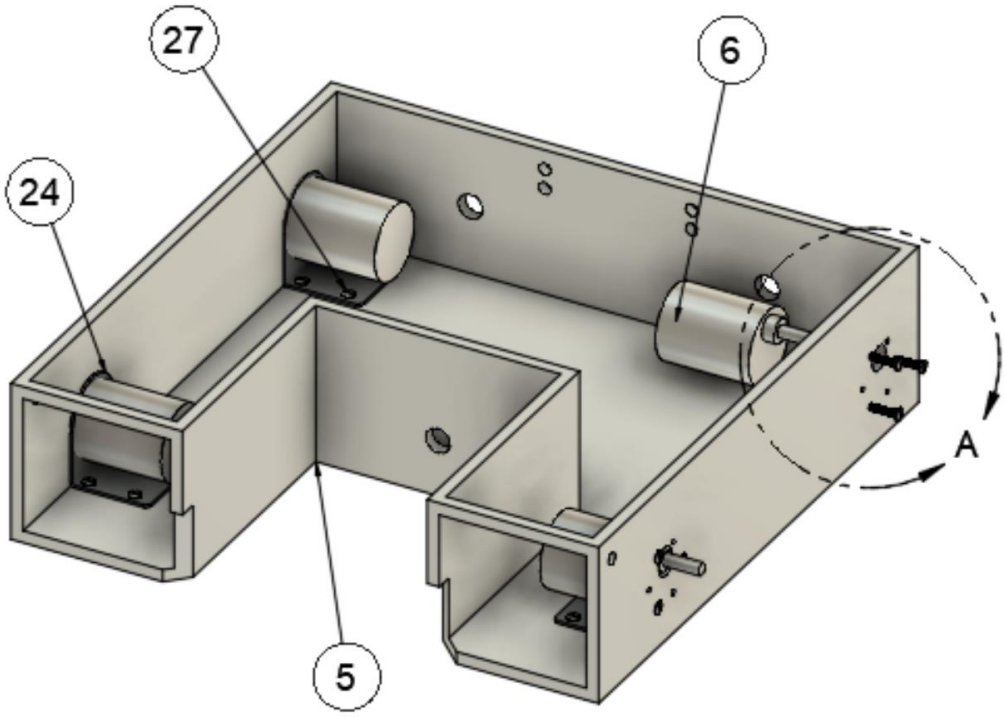
The claw was tested in two ways. First, using several different sized objects, we tested the claws ability to close and grip each object. Secondly, we did a weight test. We placed several different weighted objects on the playing field and had the claw grab each object. Once each object was gripped by the claw, we slowly picked the arm of the claw straight up away from the ground in order to see if the claw grip was able to withstand the free weight of the object without losing grip.

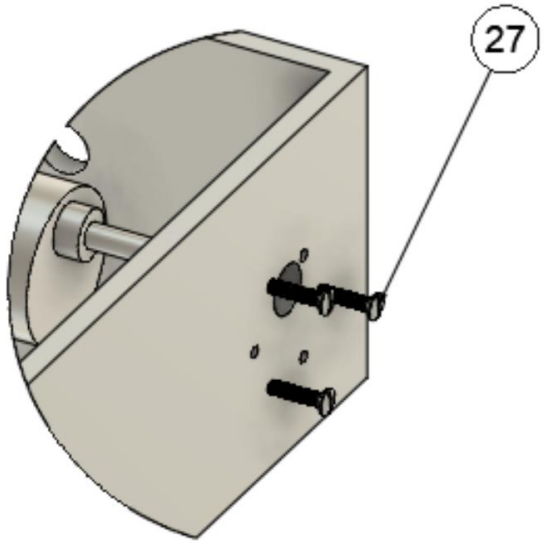
- High Torque Servo

The High Torque Servo was tested with several different weights in order to confirm the durability of the servo. We placed different sized objects with different weights at the end of the arm which is attached to the servo and had the servo lift each object at the same pace. Every object and weight were able to be picked up effortlessly.

Final Assembly Drawings



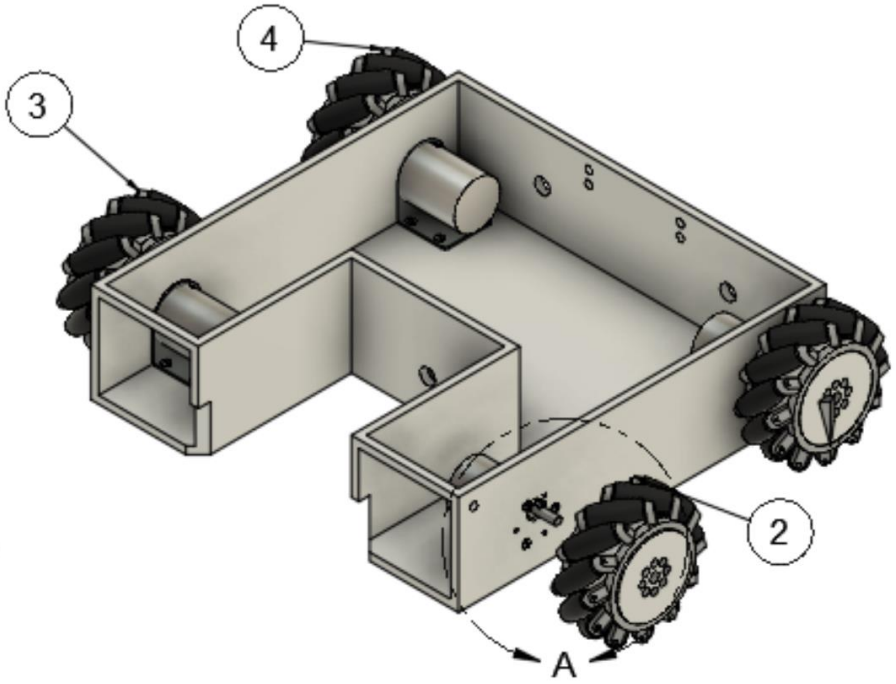


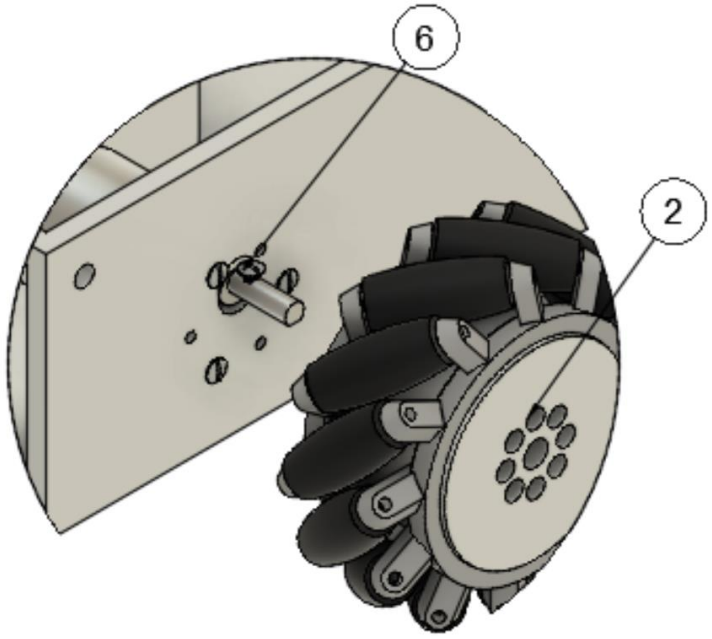


DETAIL A
SCALE 1:1

PARTS LIST				
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
5	1	BASE		ABS PLASTIC
6	4	MOTORS		ALUMINUM
24	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
26	16	NUT V1		STEEL
27	28	SCREW V1		STEEL

Second Step	PROJECT: Senior Design			
	TITLE: Mounting Motor Assembly			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Richard Sij 4/13/2020	SCALE 1:2.5	WEIGHT	SHEET 1/1	





DETAIL A
SCALE 1:1

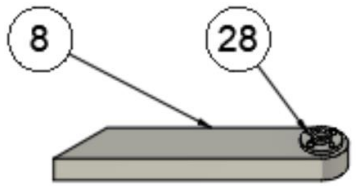
PARTS LIST				
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	BACK RIGHT TIRE		STEEL
2	1	FRONT RIGHT TIRE		STEEL
3	1	FRONT LEFT TIRE		STEEL
4	1	BACK LEFT TIRE		STEEL
5	1	BASE		ABS PLASTIC
6	4	MOTORS		ALUMINUM
24	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
26	16	NUT V1		STEEL
27	28	SCREW V1		STEEL
28	4	SET SCREW		STEEL

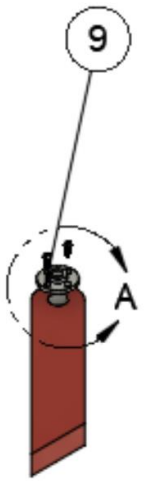
Third Step

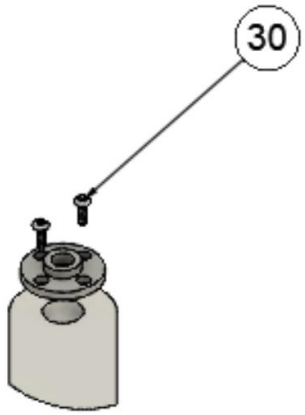
PROJECT: Senior Design

TITLE: Mounting Wheel Assembly

APPROVED	SIZE	CODE	DWG NO	REV
CHECKED	B			
DRAWN Richard Sij 4/13/2020	SCALE 1:1.75	WEIGHT	SHEET 1/1	



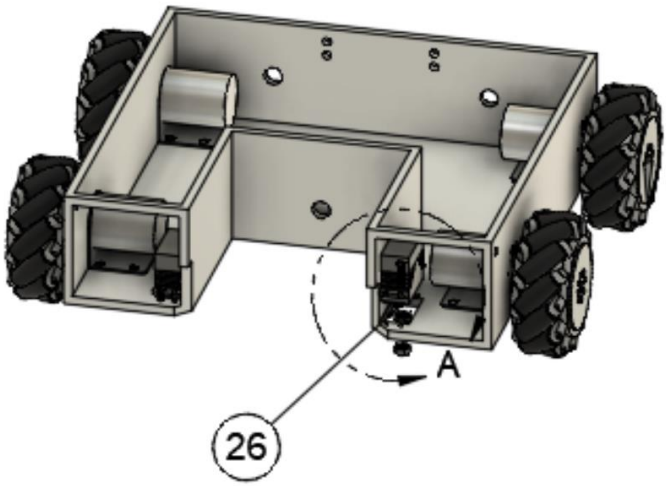


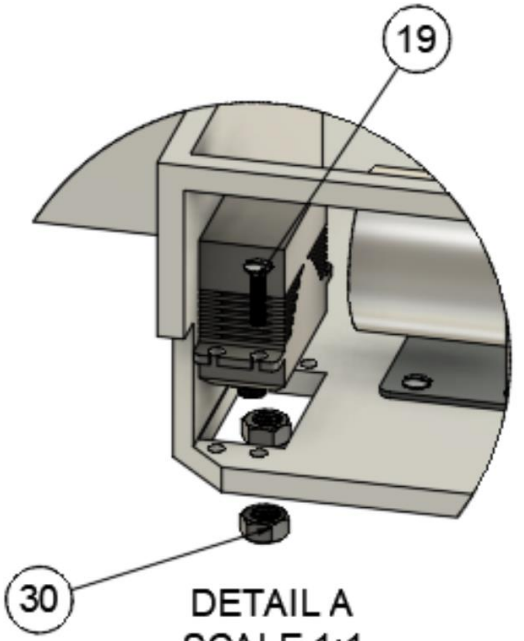


DETAIL A
SCALE 1:1

PARTS LIST				
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
8	1	BUMPER LEFT (1)		ABS PLASTIC
9	1	BUMPER RIGHT		ABS PLASTIC
28	2	BUMPER HUB		GENERIC
30	4	1.6 X 6MM SCREW V1		GENERIC

Fourth Step	PROJECT: Senior Design			
	TITLE: Bumper Assembly			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Richard Sij 4/13/2020	SCALE 1:2	WEIGHT	SHEET 1/1	





DETAIL A
SCALE 1:1

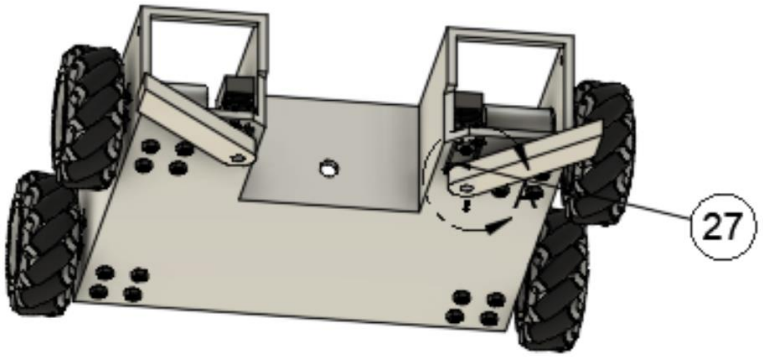
PARTS LIST				
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	BACK RIGHT TIRE		STEEL
2	1	FRONT RIGHT TIRE		STEEL
3	1	FRONT LEFT TIRE		STEEL
4	1	BACK LEFT TIRE		STEEL
5	1	BASE		ABS PLASTIC
6	4	MOTORS		ALUMINUM
16	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
18	16	NUT V1		STEEL
19	32	SCREW V1		STEEL
20	4	SET SCREW		STEEL
26	2	SERVO V1		STEEL
30	4	SMALL NUT		SOLIDWORKS MATERIALS\AISI 304

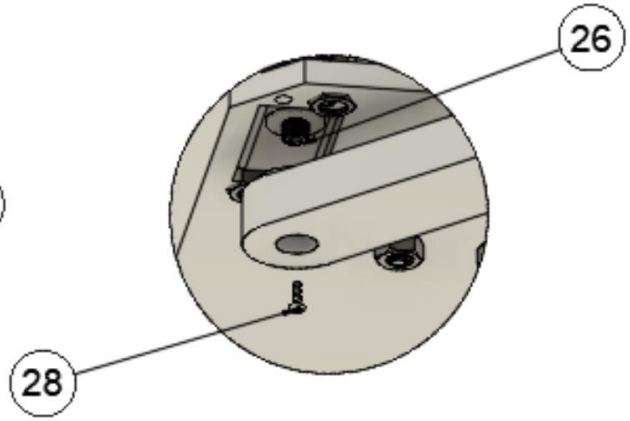
Fifth Step

PROJECT
Senior Design

TITLE
Bumper Servo Assembly

APPROVED	SIZE	CODE	DWG NO	REV
CHECKED	B			
DRAWN Richard Sij 4/13/2020	SCALE 1:3	WEIGHT	SHEET 1/1	





DETAIL A
SCALE 1:1

ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
29	4	SMALL NUT		SOLIDWORKS MATERIALS\AISI 304
28	6	1.6 X 6MM SCREW V1		GENERIC
27	2	BUMPER HUB		GENERIC
26	2	SERVO V1		STEEL
20	4	SET SCREW		STEEL
19	32	SCREWV1		STEEL
18	16	NUT V1		STEEL
16	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
8	1	BUMPER RIGHT		ABS PLASTIC
7	1	BUMPER LEFT (1)		ABS PLASTIC
6	4	MOTORS		ALUMINUM
5	1	BASE		ABS PLASTIC
4	1	BACK LEFT TIRE		STEEL
3	1	FRONT LEFT TIRE		STEEL
2	1	FRONT RIGHT TIRE		STEEL
1	1	BACK RIGHT TIRE		STEEL

PARTS LIST

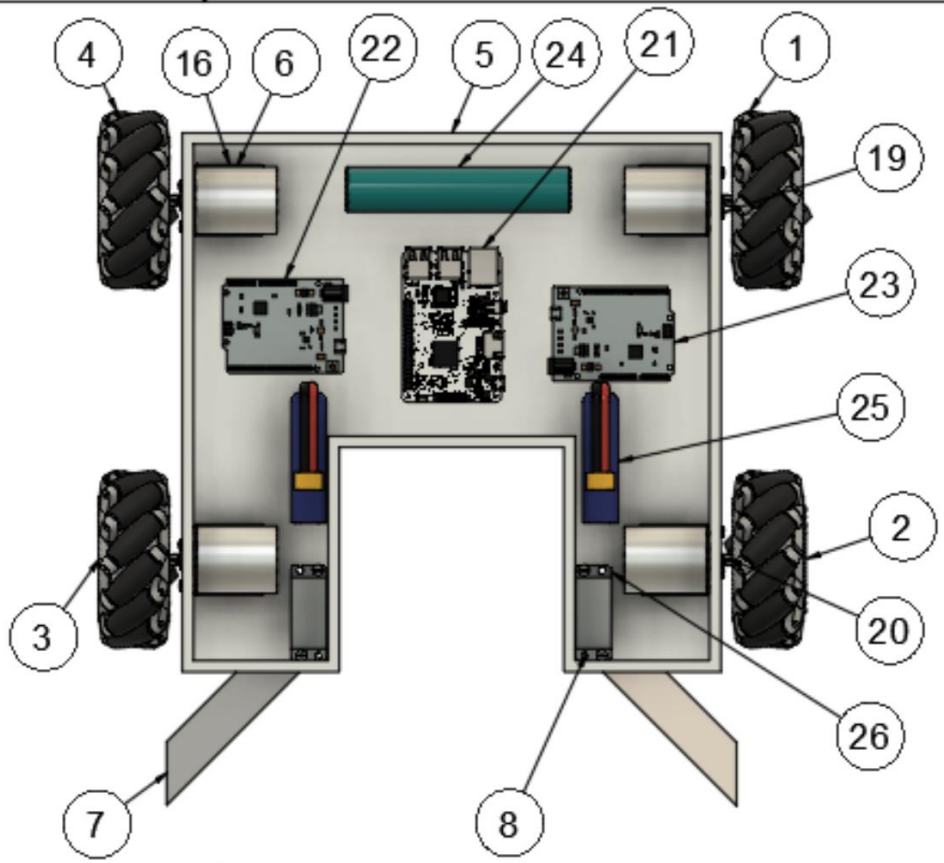
Sixth Step

PROJECT: Senior Design

TITLE: Full Bumper Assembly

APPROVED	SIZE	CODE	DWG NO	REV
CHECKED	B			
DRAWN Richard Sij 4/13/2020	SCALE 1:3	WEIGHT	SHEET 1/1	

PARTS LIST			
ITEM	QTY	PART NUMBER	MATERIAL
1	1	BACK RIGHT TIRE	STEEL
2	1	FRONT RIGHT TIRE	STEEL
3	1	FRONT LEFT TIRE	STEEL
4	1	BACK LEFT TIRE	STEEL
5	1	BASE	ABS PLASTIC
6	4	MOTORS	ALUMINUM
7	1	BUMPER LEFT (1)	ABS PLASTIC
8	1	BUMPER RIGHT	ABS PLASTIC
16	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1	STEEL
18	16	NUT V1	STEEL
19	32	SCREW V1	STEEL
20	4	SET SCREW	STEEL
21	1	RASPBERRYPI V1	STEEL
22	1	ARDUINO V1	STEEL
23	1	ARDUINO V1 (1)	STEEL
24	1	POWERBANK V1	STEEL
25	2	BATTERY V1 (1)	STEEL
26	2	SERVO V1	STEEL
27	2	BUMPER HUB	GENERIC
28	6	1.6 X 6MM SCREW V1	GENERIC
29	4	SMALL NUT	SOLIDWORKS MATERIALS|AISI 304

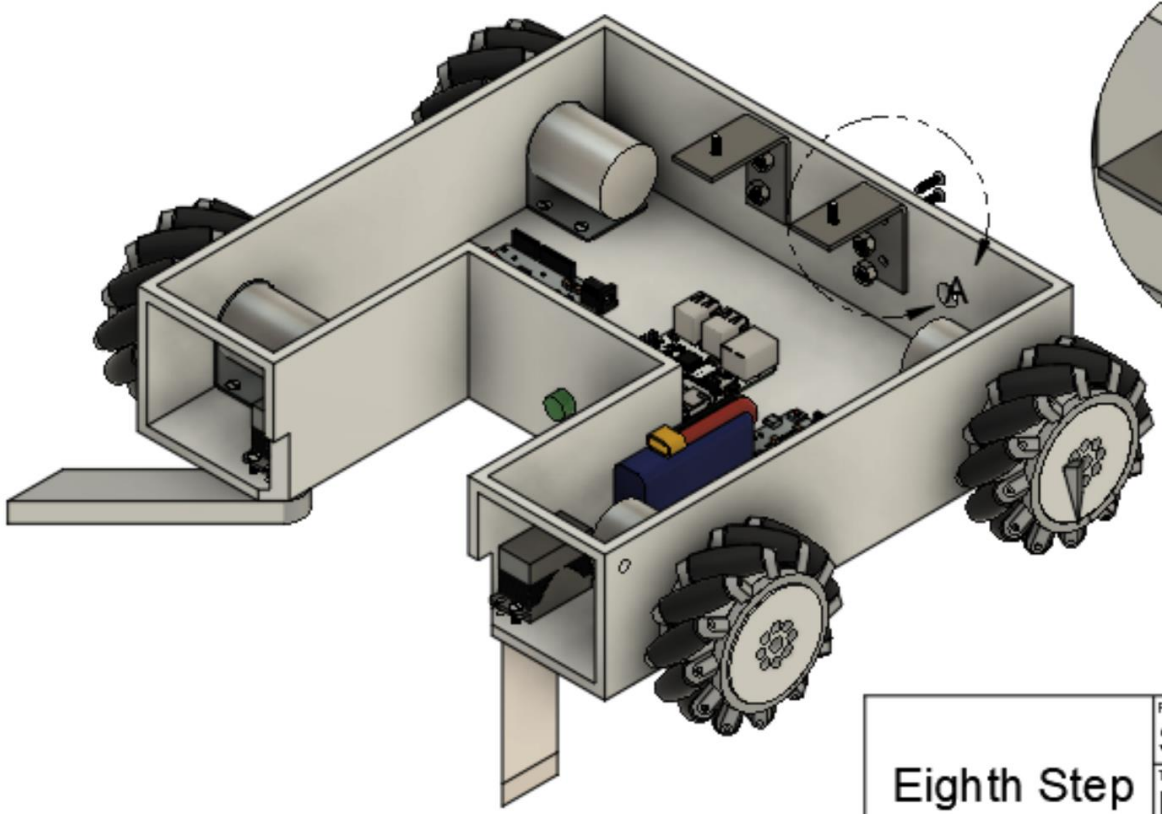


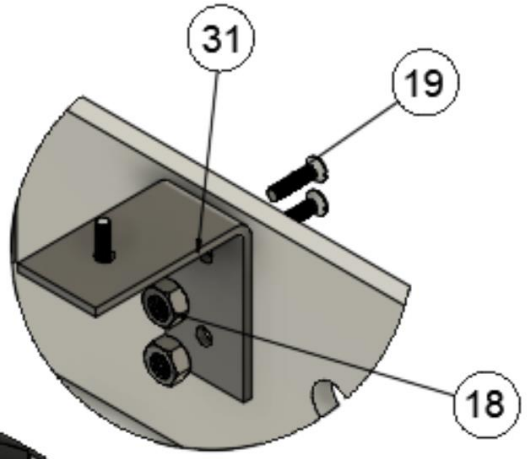
Seventh Step

PROJECT
Senior Design

TITLE
Electrical Component Assembly

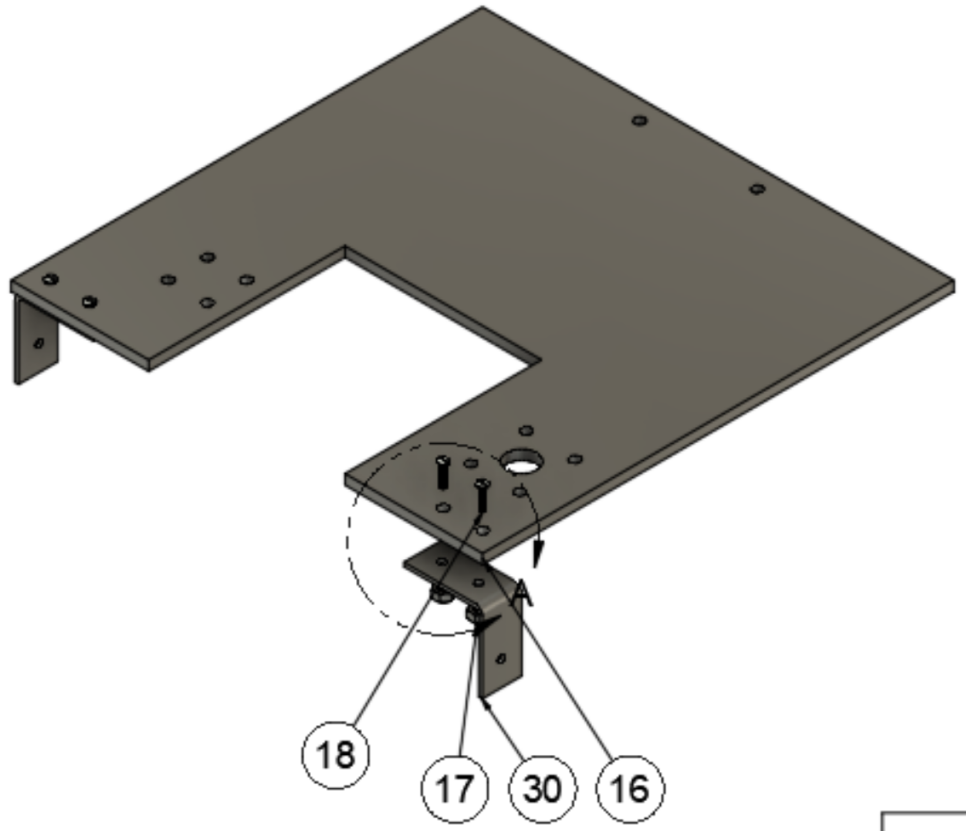
APPROVED	SIZE	CODE	DWG NO	REV
CHECKED	B			
DRAWN Richard Sij 4/13/2020	SCALE 1:3	WEIGHT	SHEET 1/1	

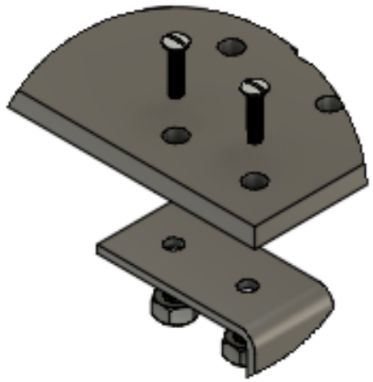




DETAIL A
SCALE 1:1

Eighth Step	PROJECT: Senior Design			
	TITLE: Lid Brackets			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Richard Sij 4/18/2020	SCALE 1:3	WEIGHT	SHEET 1/1	

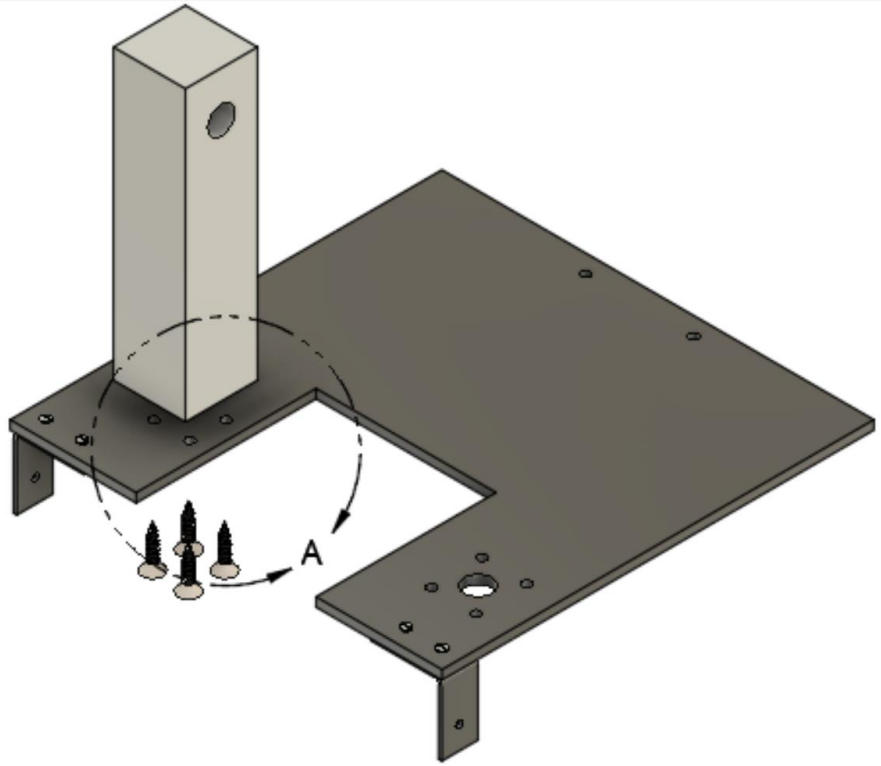


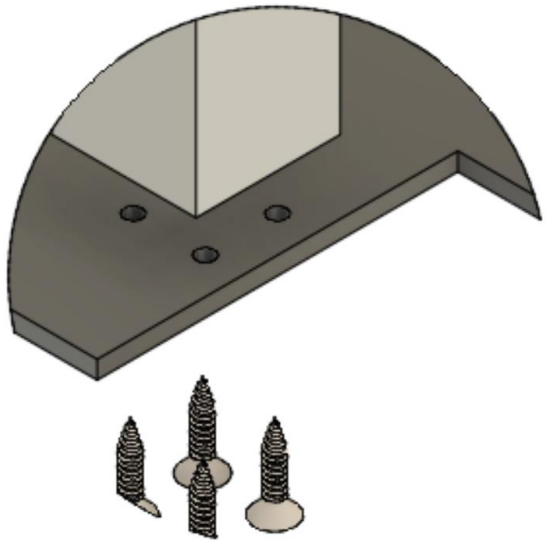


DETAIL A
SCALE 1:1

PARTS LIST			
ITEM	QTY	PART NUMBER	MATERIAL
16	1	LID (1)	STEEL
17	24	NUT V1	STEEL
18	42	SCREW V1	STEEL
30	4	FINAL DRAWING CLAW ARM	STEEL

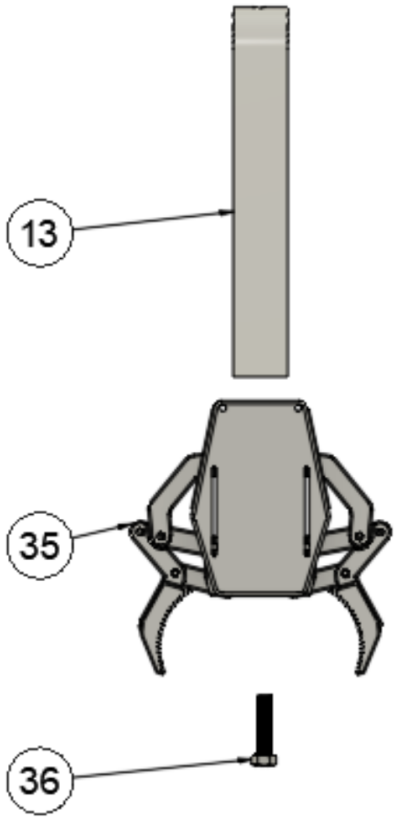
Ninth Step	PROJECT: Senior Design			
	TITLE: Lid Front Mounts			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Richard Sij 4/18/2020	SCALE 1:2	WEIGHT	SHEET 1/1	

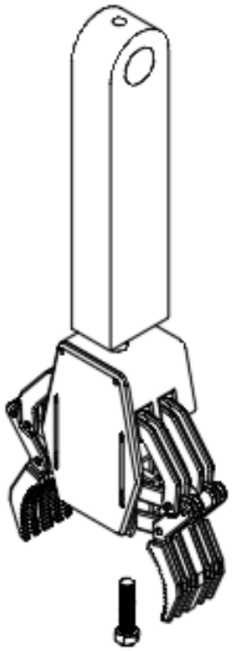




DETAIL A
SCALE 1:1

Tenth Step	PROJECT: Senior Design			
	TITLE: Support Arm Assembly			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Richard Sij 4/18/2020	SCALE 1:2	WEIGHT	SHEET 1/1	

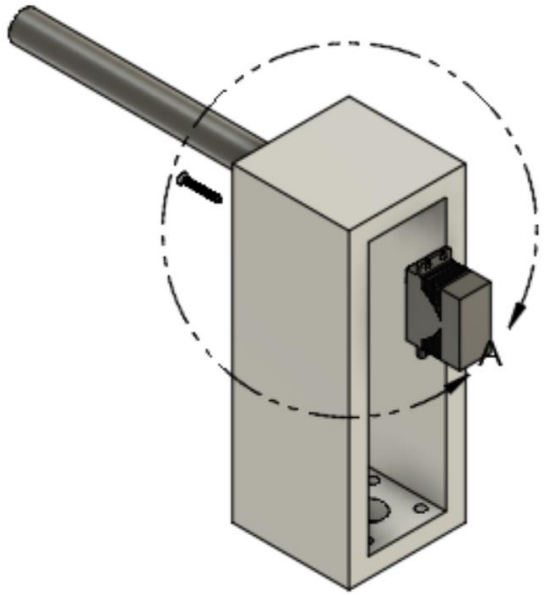


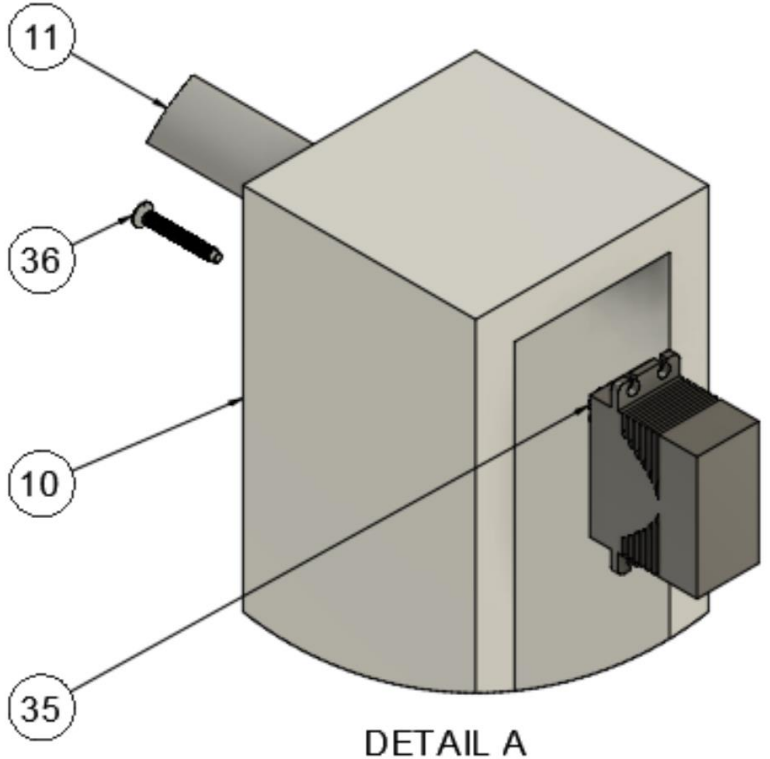


ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
36	1	SCREW1 V1		STEEL
35	1	THE CLAWV1		STEEL
13	1	ARM		ABS PLASTIC

PARTS LIST

Eleventh Step	PROJECT: Senior Design			
	TITLE: Claw Arm Assembly			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Richard Sij 4/19/2020	SCALE 1:2	WEIGHT	SHEET 1/1	

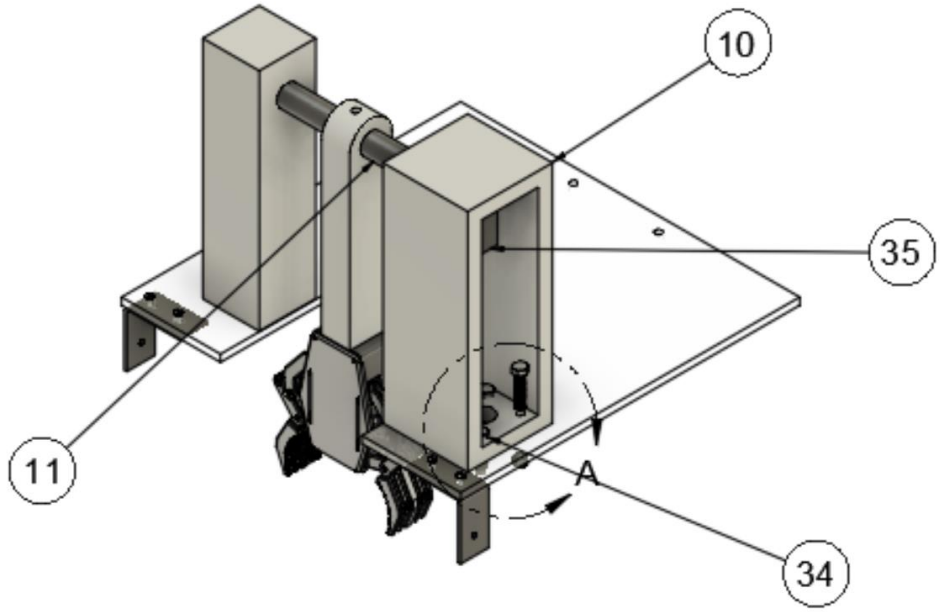


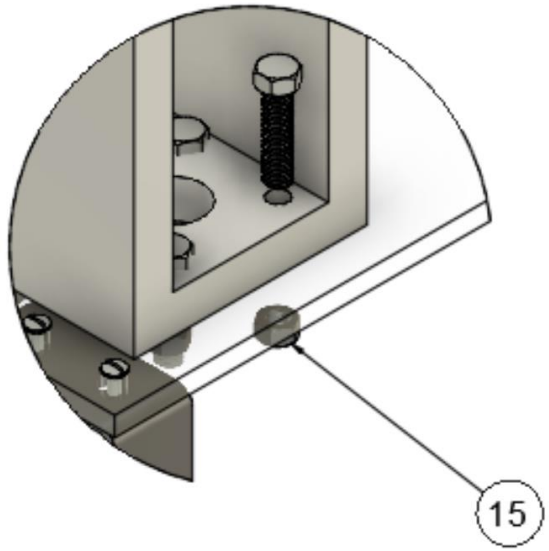


DETAIL A
SCALE 1:1

PARTS LIST			
ITEM	QTY	PART NUMBER	MATERIAL
10	1	POWER SUPPORT	ABS PLASTIC
11	1	ARM BAR	ASH
35	1	HIGH TORQUE SERVO	STEEL
36	1	SCREWV1 (2)	STEEL

Twelfth Step	PROJECT: Senior Design			
	TITLE: High Torque Servo Assembly			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Richard Sij 4/19/2020	SCALE 1:2	WEIGHT	SHEET 1/1	





DETAIL A
SCALE 1:1

ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
36	1	SCREW V1 (2)		STEEL
35	1	HIGH TORQUE SERVO		STEEL
34	5	SCREW1 V1		STEEL
33	1	THE CLAW V1		STEEL
28	4	FINAL DRAWING CLAW ARM		STEEL
16	42	SCREW V1		STEEL
15	28	NUT V1		STEEL
14	1	LID (1)		STEEL
12	1	ARM		ABS PLASTIC
11	1	ARM BAR		ASH
10	1	POWER SUPPORT		ABS PLASTIC
9	1	SUPPORT LEFT ARM		ABS PLASTIC

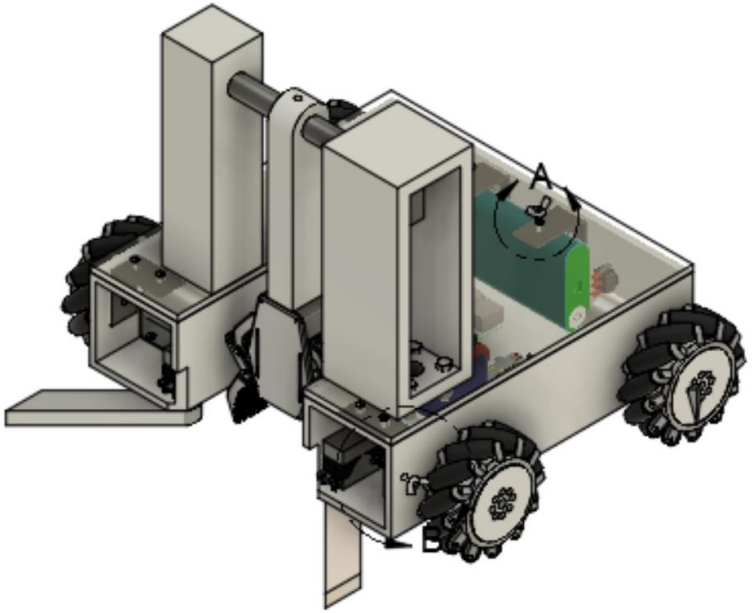
PARTS LIST

Thirteenth Step

PROJECT: Senior Design

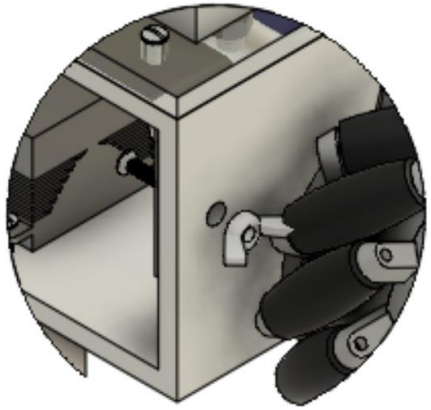
TITLE: Final Lid Assembly

APPROVED	SIZE	CODE	DWG NO	REV
CHECKED	B			
DRAWN Richard Sij 4/19/2020	SCALE 1:2	WEIGHT	SHEET 1/1	



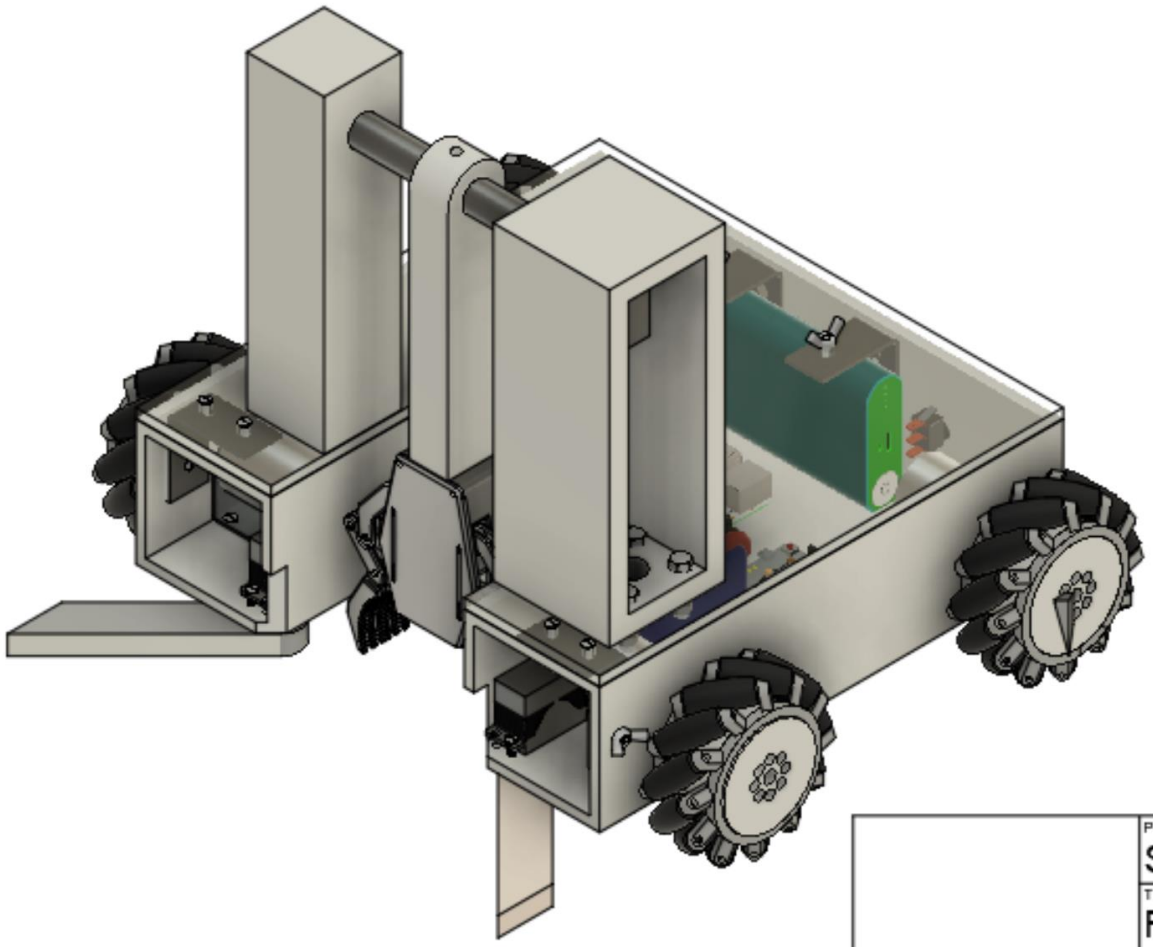


DETAIL A
SCALE 1:1



DETAIL B
SCALE 1:1

Final Step	PROJECT: Senior Design			
	TITLE: Lid to Body Assembly			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Richard Sij 4/19/2020	SCALE 1:3	WEIGHT	SHEET 1/1	



	PROJECT: Senior Design				
	TITLE: Full Assembly				
APPROVED	SIZE: B	CODE	DWG NO		REV
CHECKED					
DRAWN: Richard Sij 4/19/2020	SCALE: 1:2	WEIGHT		SHEET: 1/1	



User Manual

By: Richard Sij
Michelle Tamdjo
Wyatt Snare
Josue Rios
Joshua Luker

Contents Page

1. Getting Started
2. Budget
3. Items needed
    A. 3D design of the body
    B. Mecanum Wheels
    C. Gear Motor
    D. Driver
    E. Battery
    F. Servo HS-422 and High Torque Servo
    G. Camera
    H. Arduino Uno / Raspberry Pi / High Torque Servo
    I. Claws / Wires
    J. Soldering Iron/ Multimeter/ Lighter and cutter
4. Step 1. Mount Motors Support
5. Step 2. Mount Gear Motors
6. Step 3. Attach Wheels
7. Step 4. Wire Gear Motors to Arduino and Driver
8. Step 5. Mount of the Entire Base with Raspberry Pi Camera Mounted
9. Step 6. Movement Code Implementation
10. Step 7. Mount Flippers with Screws
11. Step 8. Mount HS-422 To the Base
12. Step 9. Mount the Flippers to the Base
13. Step 10. Flippers Code
14. Step 11. Base Cover Mounting
15. Step 12. Mount the Claw Support to the Base Cover
16. Step 13. Mount the Arm and the Claw
17. Step 14. Mount of the Arm Bar, High Torque Servo and Claw Arm Support
18. Step 15. Connect the Arm Support and Cover Base
19. Step 17. Connect the Support to the Base

20. Step 18. Connect the Arm Support and Cover Base
21. Step 19. Arm Code
22. Step 20. Image Detection Code
23. Step 21. Pathing Code
24. Step 23. State Class Code
25. Step 24. Utils Code
26. Step 25. The Code for the Entire System
27. Step 26. Full Body

1. Getting Started

Welcome to the tutorial that will instruct you on making your own robot. This tutorial will help you get started on programing and constructing the robot to make it functional. Although a basic knowledge of programing, and 3D design is recommended, this manual will be a guide for successful construction. The tutorial will start with the 3D design of the robot. The budget of the robot will be detailed with the link of each part of the robot. Each part of the robot will be described with their specifications. Finally, a step by step procedure on how to build the robot followed by the programming required.

What does the robot do?

This robot functions as a litter bot, it will detect and pick up different types of trash such as 16.9 FL OZ plastic bottles, coke cans, chip bags, hot dog type paper plates and put each article in its respectful receptacle. The robot is controlled remotely from a Bluetooth gaming controller.

2. Budget

Table 1. Items Prices

Qty.	Component list Parts Name	Part #	Cost
1	Remote Control	Logitech F710	$40.00
2	Arduino	Mega 2560	$37.98
1	Raspberry Pi 4 Board	16 Gig	$116.32
1	Claw Bracket w/ servo	N/A	$48.72
4	Omni Wheels	B01CTUTDB0	$135.95
4	DC motors	638194	$120.99
1	High-torque servo	HT1-retail	$109.95
1	Servo Adapter Board	Lynxmotion	$42.00
3	Traditional Servo	Hitec HS-422	$25.00
1	Driver shield	1438	$21.54
2	Rocker Switch	12VDC 3P SPST ON-OFF Oval Rocker Switch, 30A	$8.00
2	Batteries	Floureon 11.1 Volt, 35C, 1500mAh Battery Pack	$40.00
1	3-D Printed Base	N/A	$0.00
2	3-D Printed Flippers	N/A	$0.00
1	3-D Printed Arm	N/A	$0.00
1	Acrylic sheet	12 x 12 x 12 in	$12.00
30	Screws	3/8 by 1 long	$20.00
15	Screws	3/8 by 12 long	$7.20

Table 2. Table Cost

Totals	
Parts	74
Price	$804.10

Table 3. Website links for Items list

Qty.	Component list Parts Name	Cost	Hyperlinks
2	Arduino	$37.98	Amazon
1	Raspberry Pi	$22.99	Amazon
1	Claw	$48.72	AliExpress
4	Wheels	$135.95	Amazon
4	DC motors	$99.96	ServoCity
1	Raspberry Pi Camera	$18.99	Amazon
1	High-torque servo	$109.95	RobotShop
6	Traditional Servo	$77.94	RobotShop
4	Driver Circuits	$5.59	Amazon
1	Start Button	$5.18	Delcity
1	Stop Button	$5.18	Delcity
1	Battery	$19.99	Amazon
3	Electromagnets	$47.67	Amazon
1	Light Receiver	$9.99	Amazon
1	3-D Printer Tray	$0.00	Custom made from PLA Filament
1	3-D Printed Base	$0.00	Custom made from PLA Filament
2	3-D Printed Flippers	$0.00	Custom made from PLA Filament
1	3-D Printed Arm	$0.00	Custom made from PLA Filament
25	Screws		Home Depot

3. Items Needed

A. Body

The body of the robot was composed into multiple parts that include the base, claw arm, flippers, and claw support. Figure 1 below shows the body designed with Fusion 360 software. The main challenge with the body was the restrictions of the dimensions and weight. The robot cannot be bigger than 24 inches long, 24 inches wide and 24 inches tall and it can be no more than 25 pounds. The entire body of the robot was 3D printed with PLA filament material. With the body 3D printed, it is very light weight and easy to modify if needed.

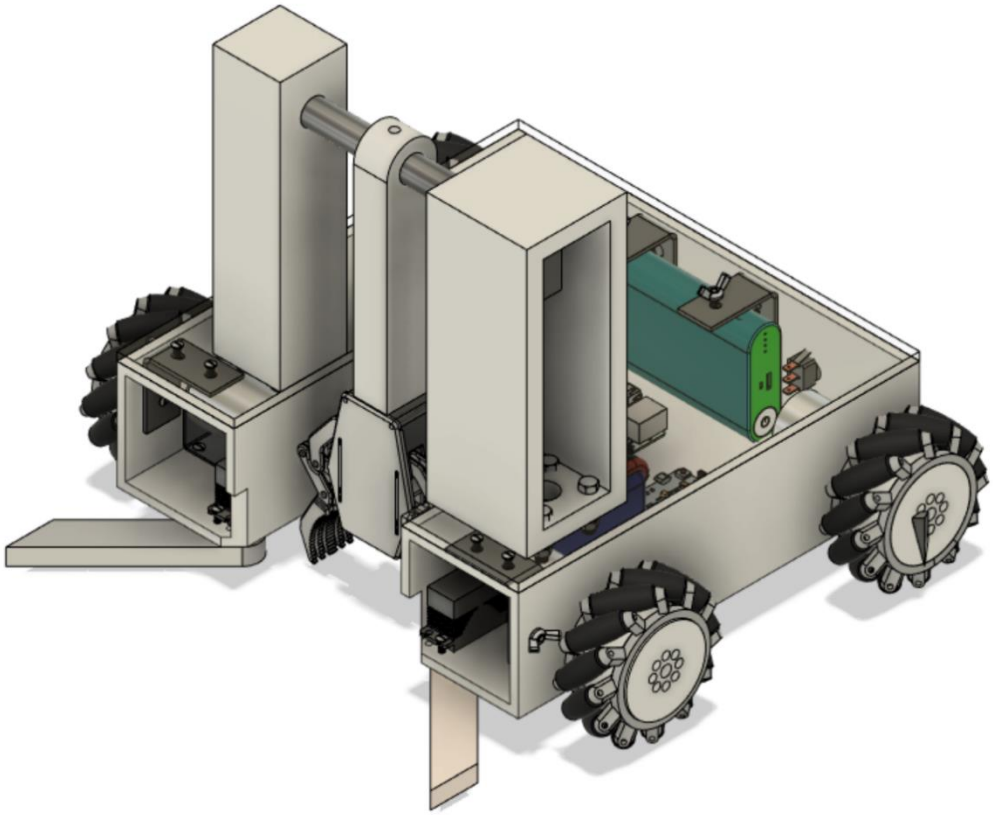


Figure 1. 3D design of the robot developed in Fusion 360

Figure 2. And Figure 3. Demonstrate the design of the base and the base top with their corresponding measurements.

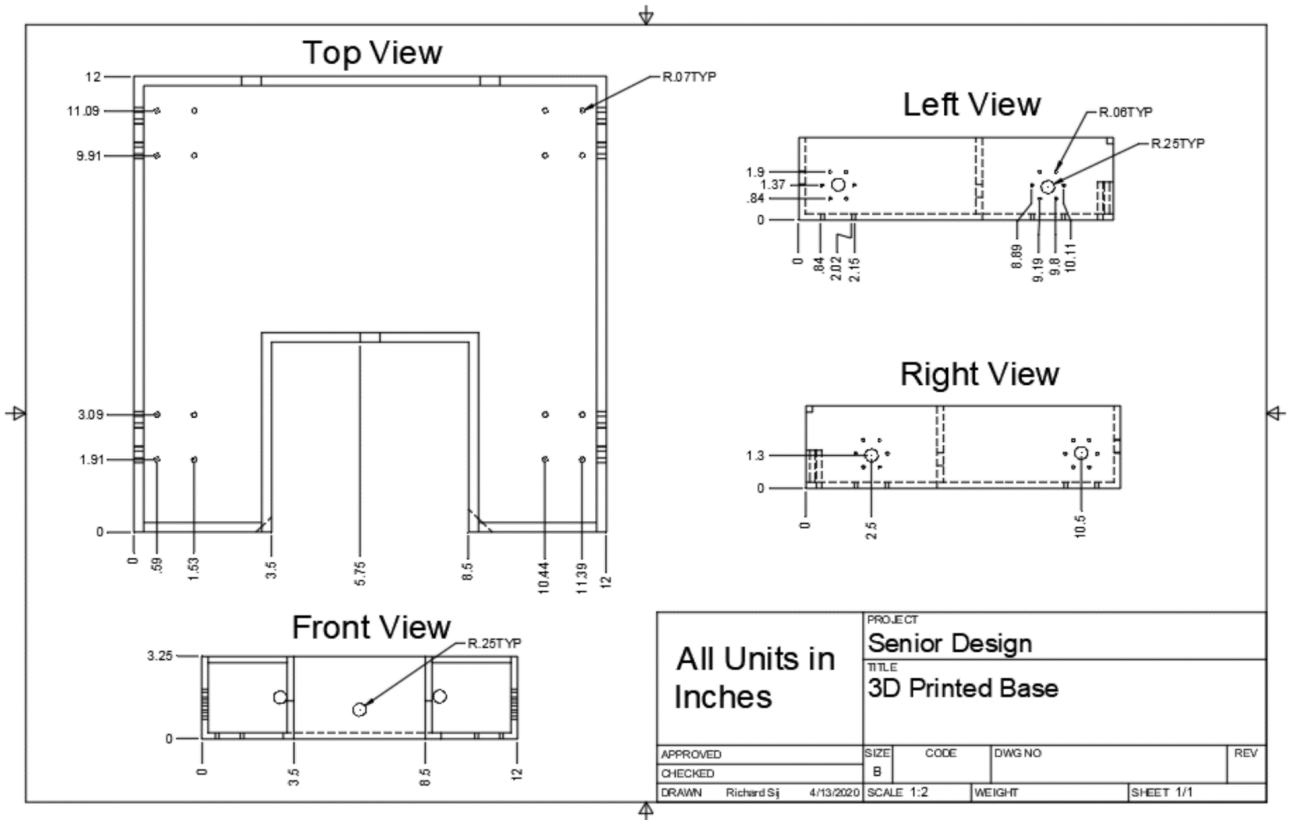


Figure 2. 3D design of the base of the robot

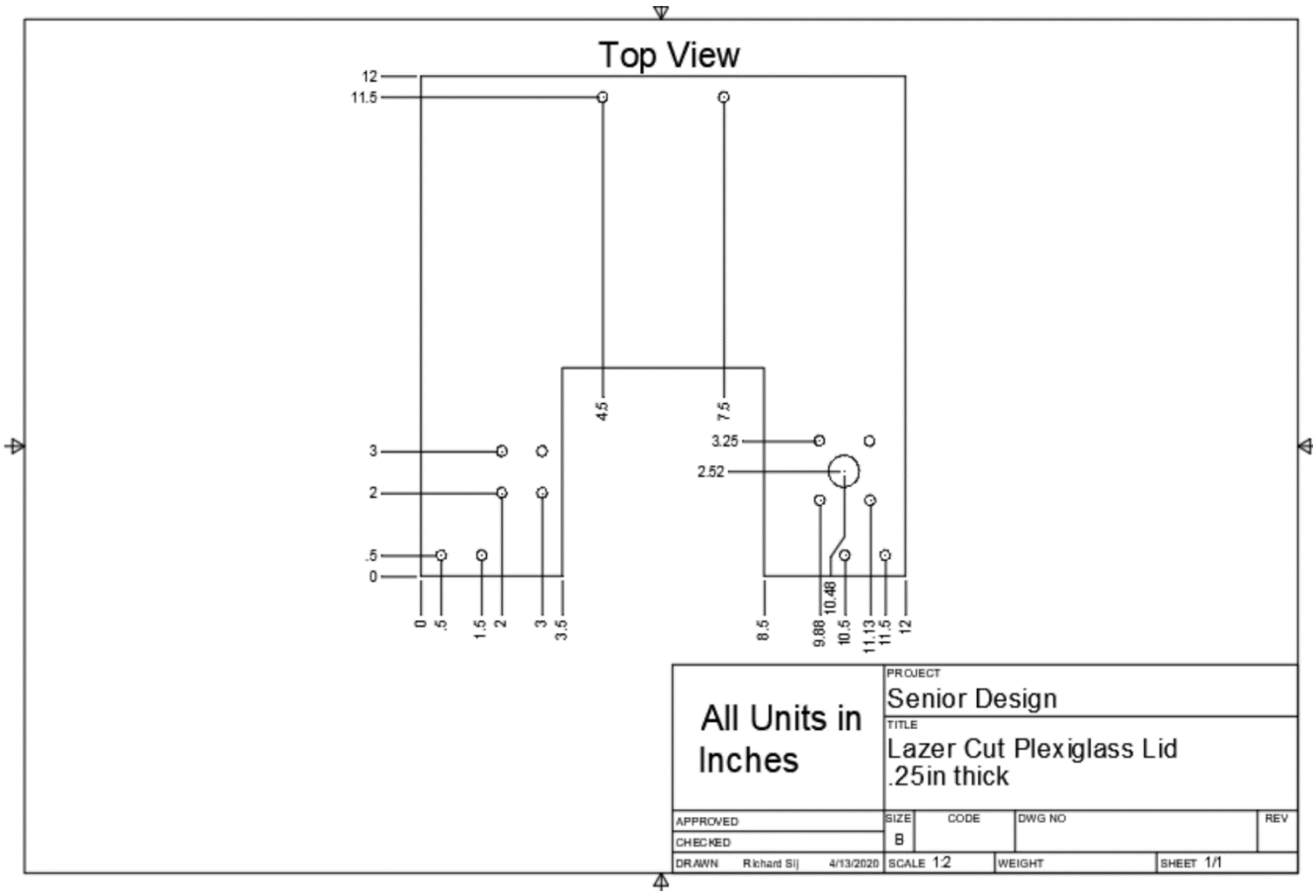


Figure 3. 3D design of the base top

The claw is attached to a 3-D printed arm which is also connected to a high torque servo. The high torque servo is used to lift the arm once the claw has been closed and has successfully captured an item. Figure 4 shows the dimensions of the claw arm.

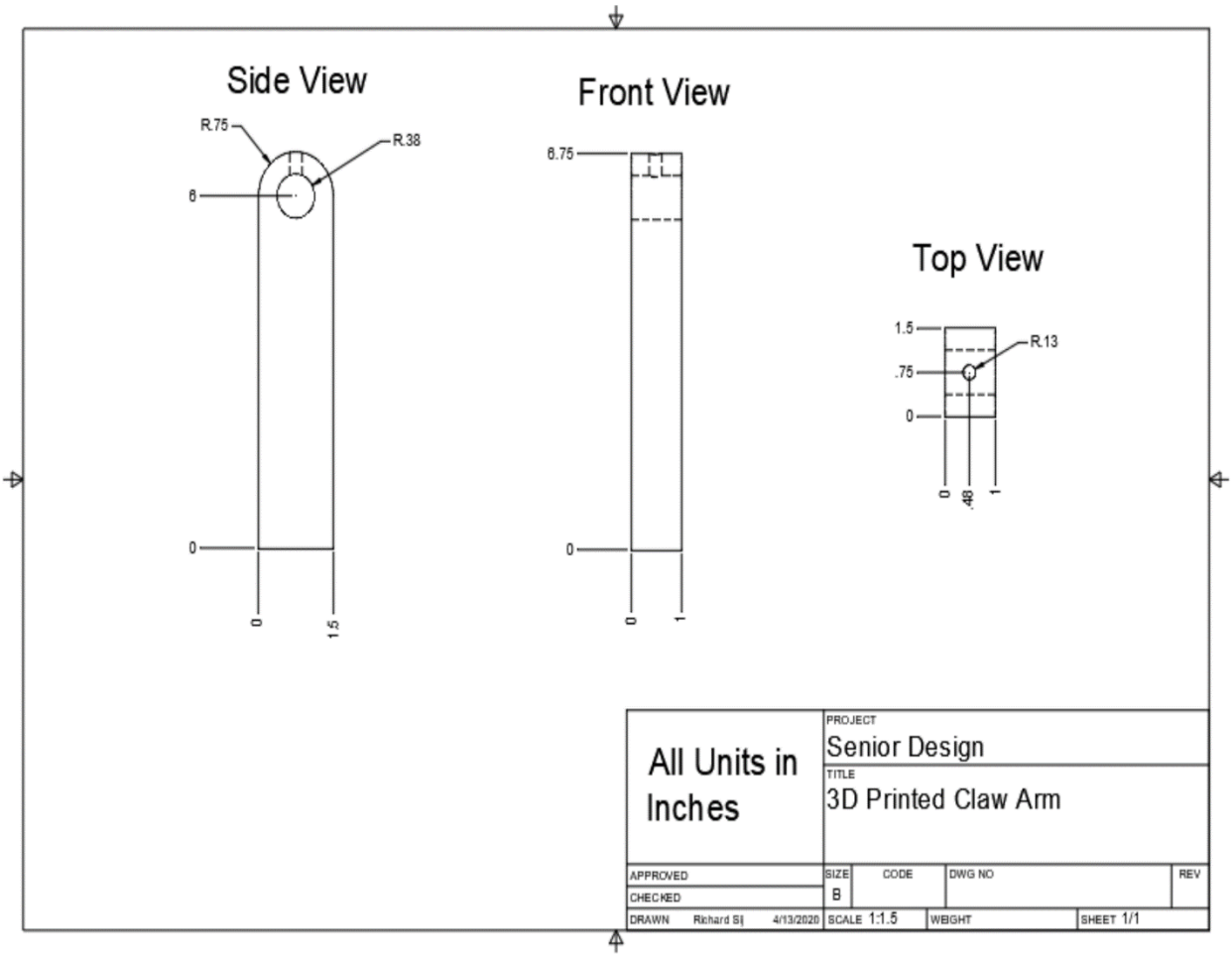


Figure 4. claw arm.

B. Wheels

Figure 6 shows 152 mm Aluminum Mecanum Wheel Set (2x Left, 2x Right) includes 4 pieces that are 2 right wheels and 2 left wheels. The wheels were provided by the IEEE department. Each wheel is connected to its own, independent motor and driven independently. Each of the four wheels include a series of rollers attached to the outer edge of the wheel, these rollers having an axis of rotation at 45° to the plane parallel to the axis of rotation of the wheel. Mecanum wheels use a standard chassis shape and by powering each wheel with a different motor, the robot can be capable of moving in any direction. This design enables the wheels to propel the robot forward, backward and side to side.



Figure 6. Mecanum Wheels

C. Gear Motor

Figure 7 shows the 101 RPM gear motor that is used to drive each wheel. Each of the four motors operate while connected to the Adafruit Motor shield version 2.3 and with a voltage supply of twelve volts.



Figure 7. 101 RPM motor

D. Motor Shield

The Adafruit Motor Shield Version 2.3 board module for Arduino is used to establish communication between Arduino and the 101 RPM motors. This motor shield was chosen because of its four-motor design. Other motor shields and motor controlling boards are typically created with a two-motor design which results in a loss of consistent voltages being supplied to each motor. With the Adafruit Motor Shield all four motors equipped on the robot can be controlled by this single board. A diagram of the Adafruit Motor Shield Version 2.3 board can be found below in Figure 8.

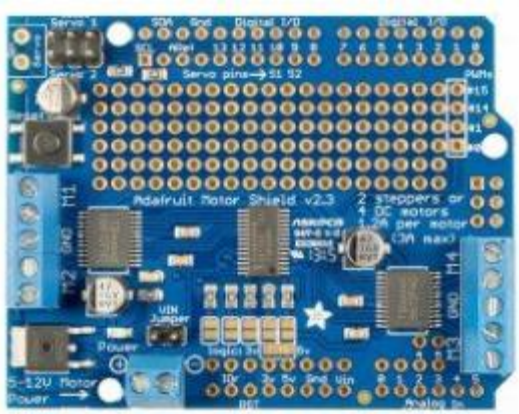


Figure 8. L298N Dual H-bridge DC stepper motor driver controller board

E. Battery

The dc batteries, drivers, motor, batteries, and Arduino used for this project are described below along with all necessary calculations used to identify if these parts will work together properly at max load. Showing in figure 9 is battery is a Floureon 11.1 Volt, 35C, 1500mAh. It is a lithium-ion Polymer RC battery pack.



Figure 9. Floureon 11.1 Volt, 35C, 1500mAh Battery Pack

F. Servos/ High torque servo

A Servo is a cornerstone in most robotic appliances. To put it short it is a little box with wires to it, and an axle that can turn some 200 degrees. on this axle you can mount a disc or some other peripheral that comes with the servo. The 3 wires are: 2 for power, and one for signal. The signal-wire goes to something that controls a servo, in this case that is the micro-controller. The micro-controller can decide which angle the axle will turn to*;* Figure 10 shows the servos used for the flippers with the necessary specifications. The servos will control the flippers which will allow the trash to flip vertically if the robot detect it facing horizontally. Figure 11. shows the high torque servo with the necessary specifications needed to lift the arm. The use of this servo was needed due to the weight of the claw.



ANNOUNCED SPECIFICATION OF
HS-422 STANDARD DELUXE SERVO

1.TECHNICAL VALUES		
CONTROL SYSTEM	: +PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	: 4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	: -20 T0 +60° C	
TEST VOLTAGE	: AT 4.8V	AT 6.0V
OPERATING SPEED	: 0.21sec/60° AT NO LOAD	0.16sec/60° AT NO LOAD
STALL TORQUE	: 3.3kg.cm(45.82oz.in)	4.1kg.cm(56.93oz.in)
OPERATING ANGLE	: 45°/ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	: CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
CURRENT DRAIN	: 8mA/IDLE AND 150mA/NO LOAD RUNNING	
DEAD BAND WIDTH	: 8usec	
CONNECTOR WIRE LENGTH	: 300mm(11.81in)	
DIMENSIONS	: 40.6x19.8x36.6mm(1.59x0.77x1.44in)	
WEIGHT	: 45.5g(1.6oz)	

Figure 10. Servo HS-422 with specification



Figure 11. High Torque Servo with specifications

Specifications

- Nominal Voltage (recommended): 12 V (6 V min, 12.6 V max)
- Torque, max. static at 12 V: 29 kg-cm
- Torque, max dynamic at 12 V: 5.8 kg-cm
- Speed, no load at 12 V: 60 RPM
- Current, no load, standby at 12 V: 70 mA
- Current, stall at 12 V: 800 mA
- Operating temperature range: 45°C~85°C
- Spline: 24 Teeth (compatible with Hitec standard spline)
- DC Motor: Coreless
- Gear ratio: 1:320
- Gear material: Steel
- Operating angle: 360° absolute and virtual multi-turn
- Communication type: TTL full duplex asynchronous serial or RC PWM.
- Communication protocol: Custom Lynxmotion Smart Servo (LSS)
- Baudrate range: 9600 bps ~ 921600 bps
- Baudrate (recommended): 115200 bps
- Connector: Molex 4-pin, 2.54mm Low Profile (pinout: Rx | Vcc | GND | Tx)
- Weight: 63 g

G. Raspberry Pi Camera

https://opencv-python-tutorials.readthedocs.io. Python was installed with Visual Studio as well Numpy. A detailed procedure on how to the robot is detecting images will be discussed later.

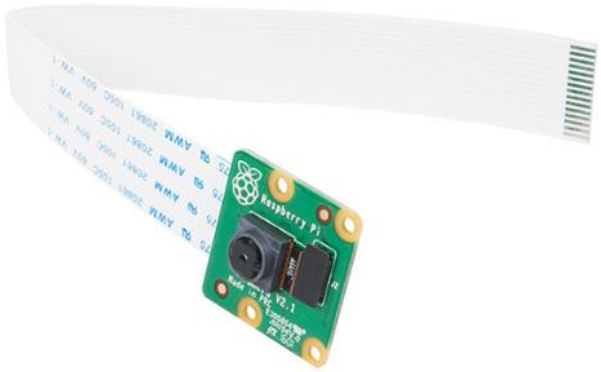


Figure 12. Raspberry Pi Camera

H. Arduino Uno/Raspberry Pi/ High Torque Servo Adapter

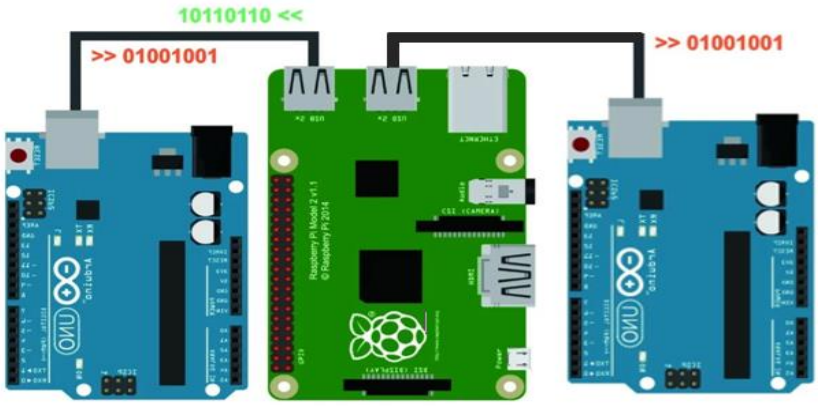


Figure 13. Connection Between Raspberry Pi and Arduino

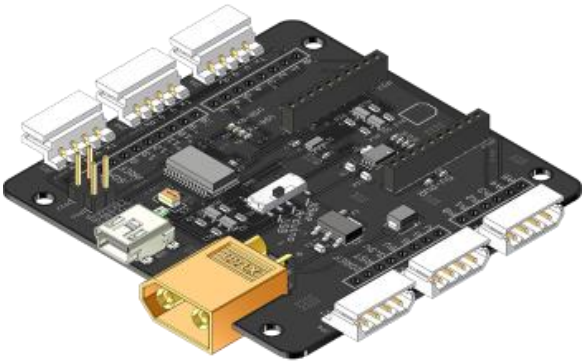


Figure 14. High Torque servo adapter

I. Claw/Wires

To connect things smooth, the wire used is a 20-22-gauge single conductor stranded copper wire, that has an insulated jacket. These two sizes were chosen because the current rating on the 20-gauge is 1.5 A and the 22-gauge is 0.92A. figure 16.

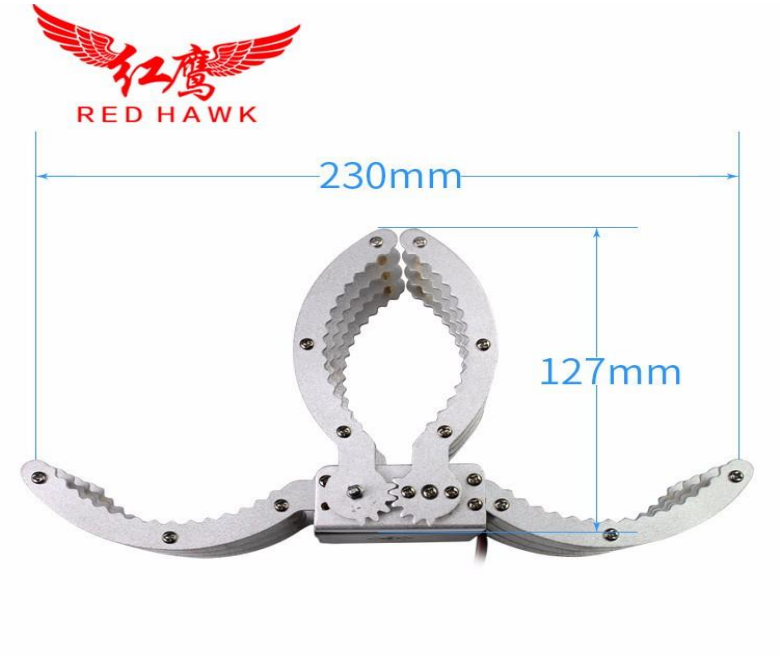


Figure 16. Claw



Figure 17. 20-22-gauge wire

J. Soldering iron /Solder / A lighter and Cutter/Multimeter

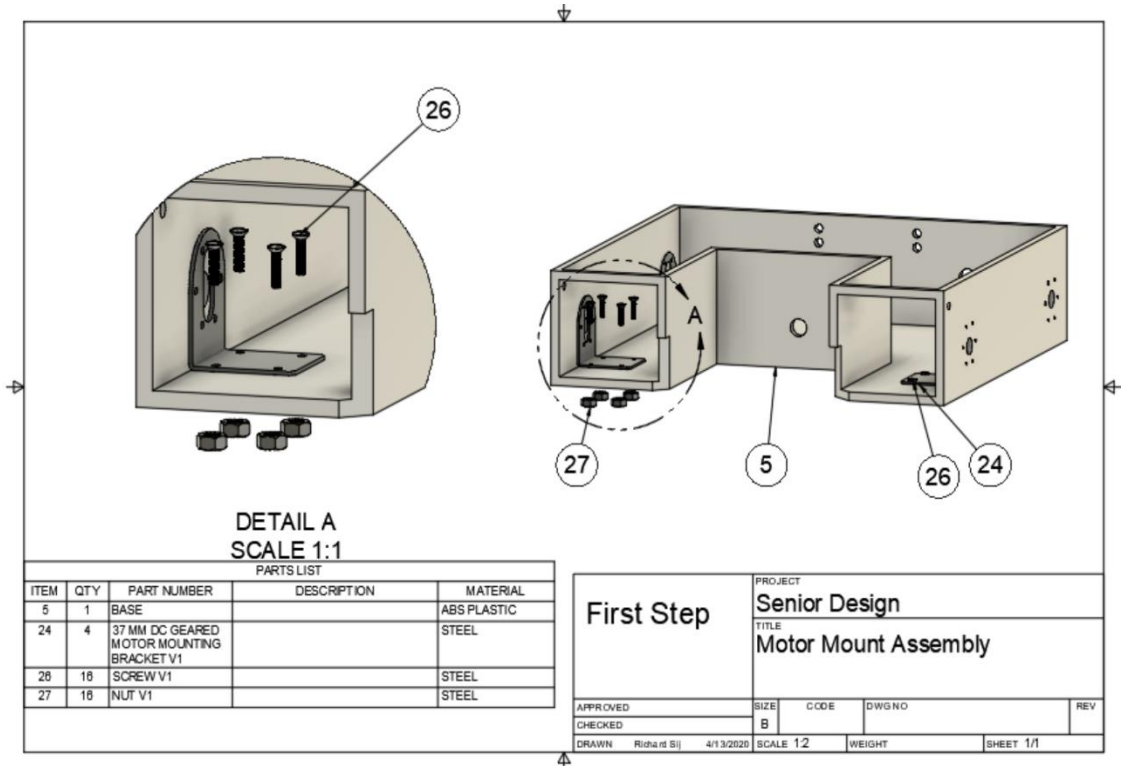
To form a permanent connection between wires and electric components soldering is necessary. A lighter is used for heat shrinking and the cutter to cut. The use of the multimeter is to measure the voltage for safety. Figure 18 shows pictures of all those pictures.



Figure 18. Soldering Iron (top left), Lighter and cutter (top right), Multimeter (bottom)

Step 1. Mount Motors Support

Wheels and motor driver are connected with 3/8 by 1 long screws via screw support. The holes of the base for each wheel were drawn on the 3D design based on the dimensions of the driver. Figure 20 shows the mounting of the motor and figure 21 describes the mounting of the top.



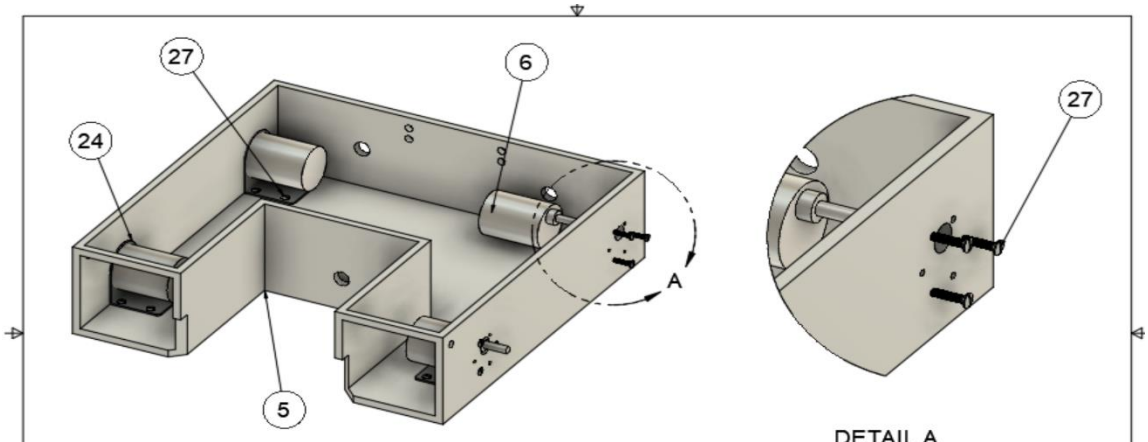
PARTS LIST

ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
5	1	BASE		ABS PLASTIC
24	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
26	16	SCREW V1		STEEL
27	16	NUT V1		STEEL

First Step

PROJECT: Senior Design
TITLE: Motor Mount Assembly
APPROVED
CHECKED
DRAWN Richard Sij 4/13/2020
SIZE B | CODE | DWG NO | REV
SCALE 1:2 | WEIGHT | SHEET 1/1

Step 2. Mount Gear Motors



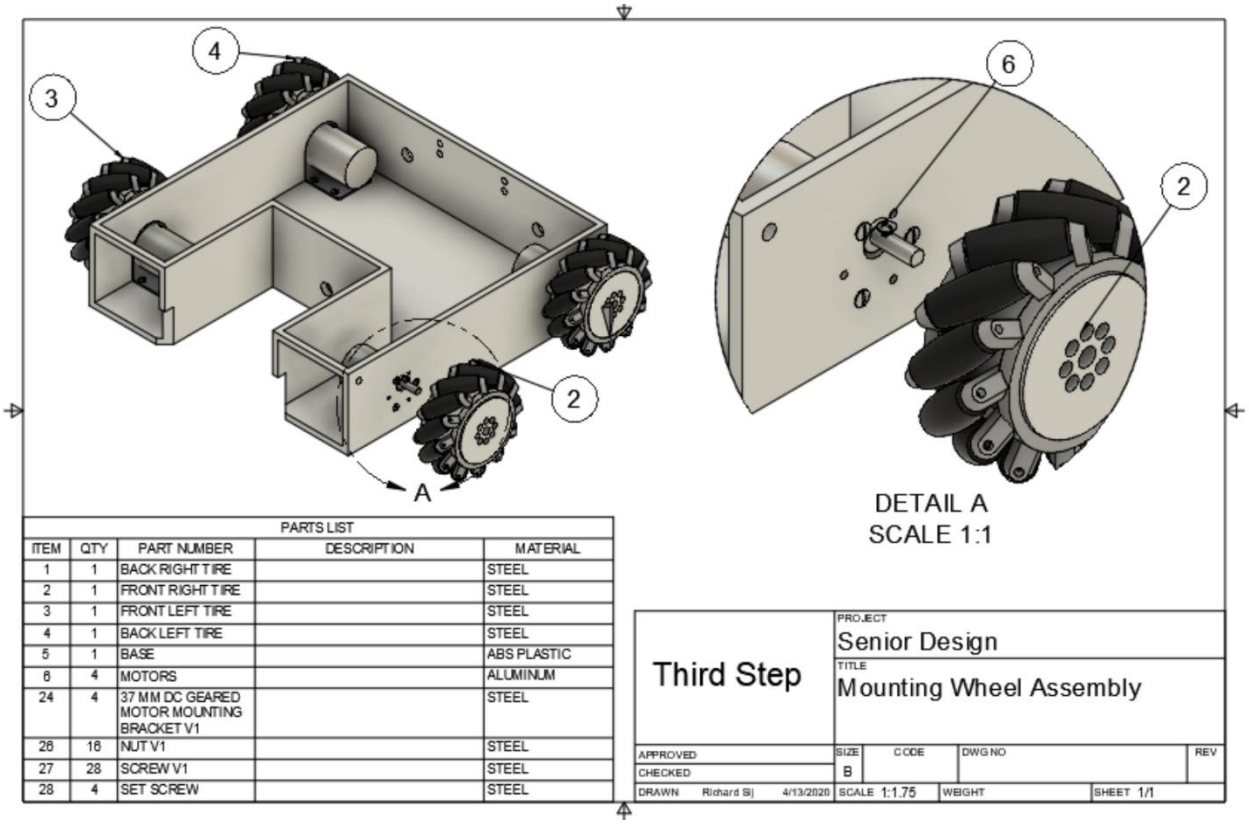
PARTS LIST

ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
5	1	BASE		ABS PLASTIC
6	4	MOTORS		ALUMINUM
24	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
26	16	NUT V1		STEEL
27	28	SCREW V1		STEEL

Second Step

PROJECT: Senior Design
TITLE: Mounting Motor Assembly
APPROVED
CHECKED
DRAWN Richard Sij 4/13/2020
SIZE B | CODE | DWG NO | REV
SCALE 1:2.5 | WEIGHT | SHEET 1/1

Step 3. Attach Wheels



PARTS LIST

ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	BACK RIGHT TIRE		STEEL
2	1	FRONT RIGHT TIRE		STEEL
3	1	FRONT LEFT TIRE		STEEL
4	1	BACK LEFT TIRE		STEEL
5	1	BASE		ABS PLASTIC
6	4	MOTORS		ALUMINUM
24	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
26	16	NUT V1		STEEL
27	28	SCREW V1		STEEL
28	4	SET SCREW		STEEL

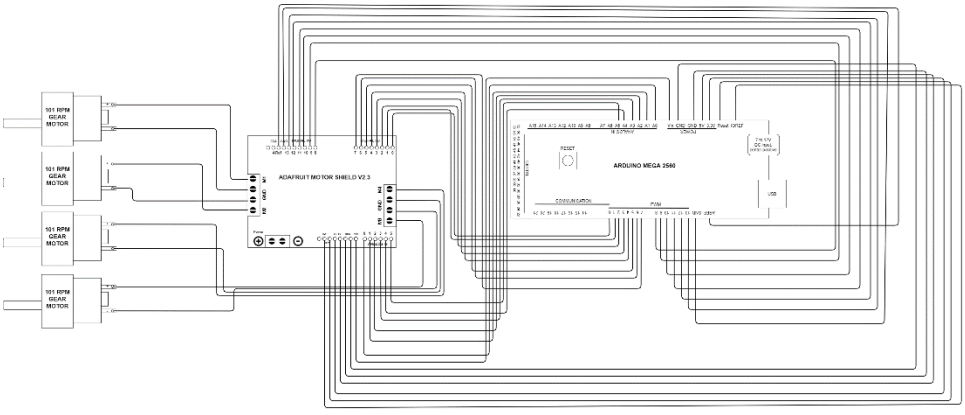
Third Step

PROJECT: Senior Design

TITLE: Mounting Wheel Assembly

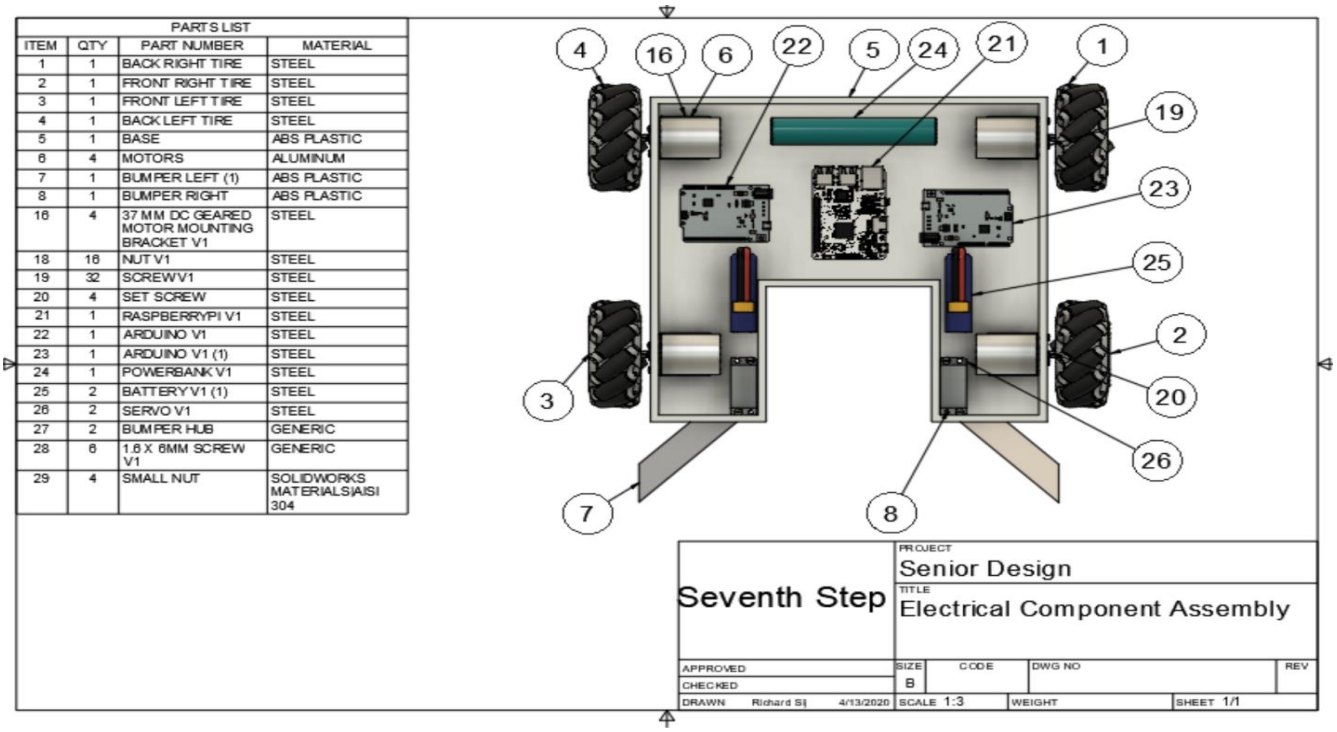
APPROVED	SIZE	CODE	DWG NO	REV
CHECKED	B			
DRAWN Richard Sj 4/13/2020	SCALE 1:1.75	WEIGHT	SHEET 1/1	

Step 4. Wire Gear Motors to Arduino and Motor Shield



Above the figure shows the circuit connection use the on-board +5V power supply from Arduino board, and should be done without the 5V Enable Jumper on (Active 5V). This connection can drive two 5V DC motors simultaneously. The other two motors should be wired the same way.

Step 5. Mount of the Entire Base with Raspberry Pi Camera Mounted



PART'S LIST

ITEM	QTY	PART NUMBER	MATERIAL
1	1	BACK RIGHT TIRE	STEEL
2	1	FRONT RIGHT TIRE	STEEL
3	1	FRONT LEFT TIRE	STEEL
4	1	BACK LEFT TIRE	STEEL
5	1	BASE	ABS PLASTIC
6	4	MOTORS	ALUMINUM
7	1	BUMPER LEFT (1)	ABS PLASTIC
8	1	BUMPER RIGHT	ABS PLASTIC
16	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1	STEEL
18	16	NUT V1	STEEL
19	32	SCREW V1	STEEL
20	4	SET SCREW	STEEL
21	1	RASPBERRYPI V1	STEEL
22	1	ARDUINO V1	STEEL
23	1	ARDUINO V1 (1)	STEEL
24	1	POWERBANK V1	STEEL
25	2	BATTERY V1 (1)	STEEL
26	2	SERVO V1	STEEL
27	2	BUMPER HUB	GENERIC
28	6	1.6 X 6MM SCREW V1	GENERIC
29	4	SMALL NUT	SOLIDWORKS MATERIALS\AISI 304

Step 6. Movement Code Implementation

File: Motion.cpp

```
#include <Wire.h>
#include "Motion.h"
#include <Adafruit_MotorShield.h>

Motion::Motion(int lf,  int lb, int rf, int rb){
  _AFMS = Adafruit_MotorShield();

  //initialize motors
  //_lf - left front motor
  //_rf - right front motor
  //_rb - right back motor
  //_lb - left back motor
  _lf = _AFMS.getMotor(lf);
  _lb = _AFMS.getMotor(lb);
  _rf = _AFMS.getMotor(rf);
  _rb = _AFMS.getMotor(rb);
}

//stop ceases all motion of motors
Motion::stop(){
    _lf->run(RELEASE);
    _lb->run(RELEASE);
    _rf->run(RELEASE);
    _rb->run(RELEASE);
}
```

```
Motion::rotate(bool right){
    this->stop();
    if(right == true){
        _lf->run(FORWARD);
        _lb->run(FORWARD);
        _rf->run(BACKWARD);
        _rb->run(BACKWARD);
    }else{
        _lf->run(BACKWARD);
        _lb->run(BACKWARD);
        _rf->run(FORWARD);
        _rb->run(FORWARD);
    }
}

Motion::linear(bool forward){
    if(forward == true){
        _lf->run(FORWARD);
        _lb->run(FORWARD);
        _rf->run(FORWARD);
        _rb->run(FORWARD);
    }else{
        _lf->run(BACKWARD);
        _lb->run(BACKWARD);
        _rf->run(BACKWARD);
        _rb->run(BACKWARD);
    }
}
```

```
Motion::forward(){
   this->linear(true);
}

Motion::backward(){
   this->linear(false);
}

Motion::turnRight(){
   this->rotate(true);
}

Motion::turnLeft(){
   this->rotate(false);
}

Motion::begin(){
   int speed = 200;
   _AFMS.begin();
   _lf->setSpeed(speed);
   _lb->setSpeed(speed);
   _rf->setSpeed(speed);
   _rb->setSpeed(speed);
   this->stop();
}
```

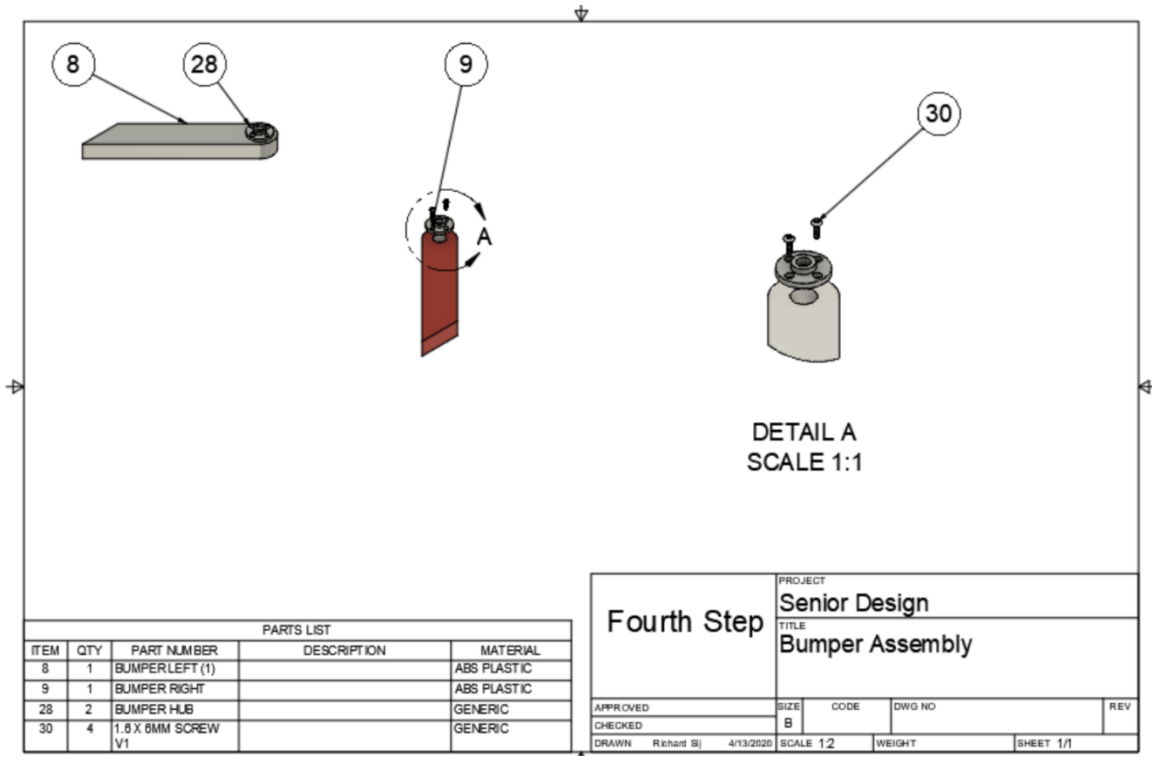
File: motion.h

```
/*
   Initializes motors for movement for V.I.C.T.O.R.
*/
#ifndef Motion_h
#define Motion_h
#include <Adafruit_MotorShield.h>

class Motion{
  public:
    Motion(int lf,  int lb, int rf, int rb);
    stop();
    begin();
    forward();
    backward();
    turnLeft();
    turnRight();
  private:
    Adafruit_MotorShield _AFMS;
    Adafruit_DCMotor *_lf, *_rf, *_rb, *_lb;
    rotate(bool right);
    linear(bool forward);
};

#endif
```

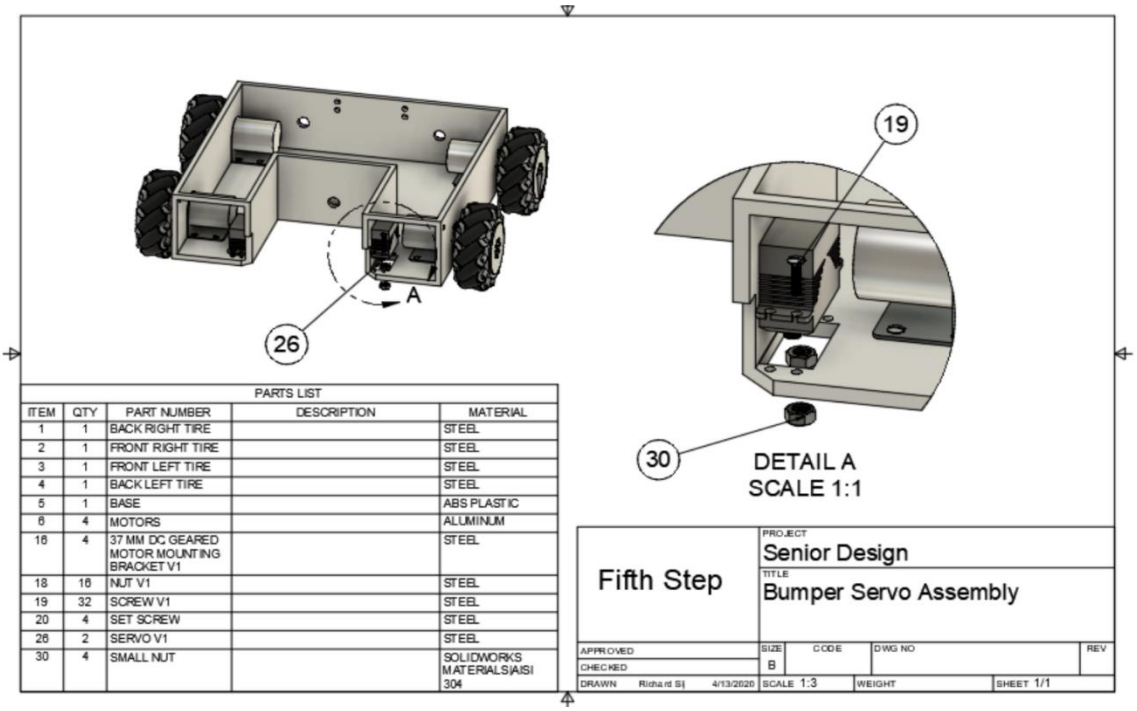
Step 7. Mount Flippers with Screws



PARTS LIST

ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
8	1	BUMPER LEFT (1)		ABS PLASTIC
9	1	BUMPER RIGHT		ABS PLASTIC
28	2	BUMPER HUB		GENERIC
30	4	1.6 X 6MM SCREW V1		GENERIC

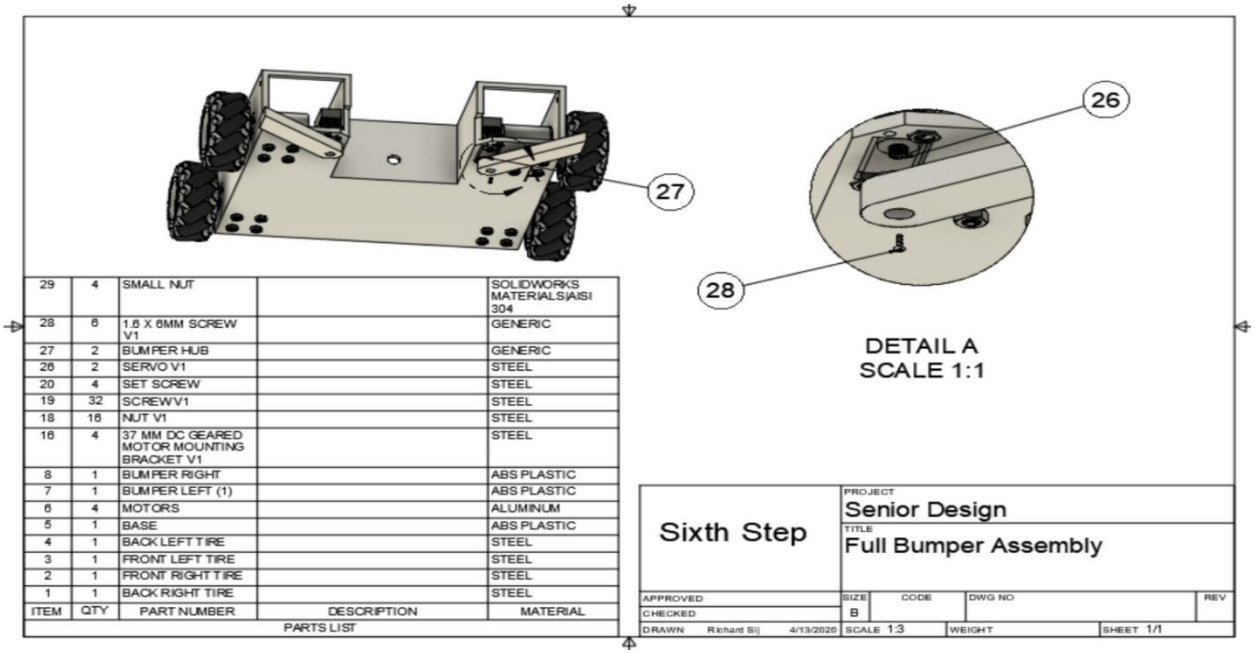
Step 8. Mount HS-422 To the Base



PARTS LIST

ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	BACK RIGHT TIRE		STEEL
2	1	FRONT RIGHT TIRE		STEEL
3	1	FRONT LEFT TIRE		STEEL
4	1	BACK LEFT TIRE		STEEL
5	1	BASE		ABS PLASTIC
6	4	MOTORS		ALUMINUM
16	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
18	16	NUT V1		STEEL
19	32	SCREW V1		STEEL
20	4	SET SCREW		STEEL
26	2	SERVO V1		STEEL
30	4	SMALL NUT		SOLIDWORKS MATERIALS\AISI 304

Step 9. Mount the Flippers to the Base



ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
29	4	SMALL NUT		SOLIDWORKS MATERIALS\AISI 304
28	6	1.6 X 6MM SCREW V1		GENERIC
27	2	BUMPER HUB		GENERIC
26	2	SERVO V1		STEEL
20	4	SET SCREW		STEEL
19	32	SCREW V1		STEEL
18	16	NUT V1		STEEL
16	4	37 MM DC GEARED MOTOR MOUNTING BRACKET V1		STEEL
8	1	BUMPER RIGHT		ABS PLASTIC
7	1	BUMPER LEFT (1)		ABS PLASTIC
6	4	MOTORS		ALUMINUM
5	1	BASE		ABS PLASTIC
4	1	BACK LEFT TIRE		STEEL
3	1	FRONT LEFT TIRE		STEEL
2	1	FRONT RIGHT TIRE		STEEL
1	1	BACK RIGHT TIRE		STEEL

PARTS LIST

Sixth Step	PROJECT: Senior Design			
	TITLE: Full Bumper Assembly			
APPROVED	SIZE: B	CODE	DWG NO	REV
CHECKED				
DRAWN Rishant Sij 4/13/2020	SCALE 1:3	WEIGHT	SHEET 1/1	

Step 10. Overall Servo Code

File: Servo.cpp

```
#include <Servo.h>
#include "Servos.h"
#include <Arduino.h>
#include <LSS.h>

// ID set to default LSS ID = 0
#define LSS_ID          (0)
#define LSS_BAUD        (LSS_DefaultBaud)

Servos::Servos(int clawPin, int leftFlipper, int rightFlipper){
  isClawClosed      = true;
  isFlippersClosed  = true;
  isArmLowered      = true;

  _openClawAngle    = 135;
  _closeClawAngle   = 180;
  _openFlipperAngle = 50;
  _closeFlipperAngle =  1;

  _clawPin      = clawPin;
  _leftFlipper  = leftFlipper;
  _rightFlipper = rightFlipper;
}

Servos::raiseArm(){
  if(isArmLowered){
```

```
    _arm.moveT(700, 2000);
    delay(2050);
    isArmLowered = false;
  }
}

Servos::lowerArm(){
  if(!isArmLowered){
    _arm.moveT(-700, 2000);
    delay(2050);
    isArmLowered = true;
  }
}

Servos::openClaw(){
  if(isClawClosed){
    for(int pos = _closeClawAngle; pos >= _openClawAngle; pos -= 1) {
      _claw.write(pos);
      delay(20);
    }
  }
  isClawClosed = false;
}

Servos::closeClaw(){
  if(!isClawClosed){
    for (int pos = _openClawAngle; pos <= _closeClawAngle; pos += 1) {
      _claw.write(pos);
```

```
    delay(20);
   }
  }
  isClawClosed = true;
}

Servos::openFlippers(){
  int opened = _openFlipperAngle;
  int closed = _closeFlipperAngle;

  if(isFlippersClosed){
   for (int pos = closed; pos <= opened; pos += 1) {
    _lFlipper.write(pos);
    _rFlipper.write(max(opened-pos, 1));
    delay(20);
   }
  }

  isFlippersClosed = false;
}

Servos::closeFlippers(){
  int opened = _openFlipperAngle;
  int closed = _closeFlipperAngle;

  if(!isFlippersClosed){
   for (int pos = opened; pos >= closed; pos -= 1) {
    _lFlipper.write(pos);
```

```
      _rFlipper.write(opened-pos);
      delay(40);
    }
  }

  isFlippersClosed = true;
}

Servos::begin(){
  _claw.attach(_clawPin);
  _lFlipper.attach(_leftFlipper);
  _rFlipper.attach(_rightFlipper);

  //Initialize arm and move to 0deg
  _arm = LSS(LSS_ID);
  LSS::initBus(Serial1, LSS_BAUD);
  _arm.move(0);
  delay(2000);
}
```

File: Servos.h

```
#ifndef Servos_h
#define Servos_h
#include <Servo.h>
#include <LSS.h>

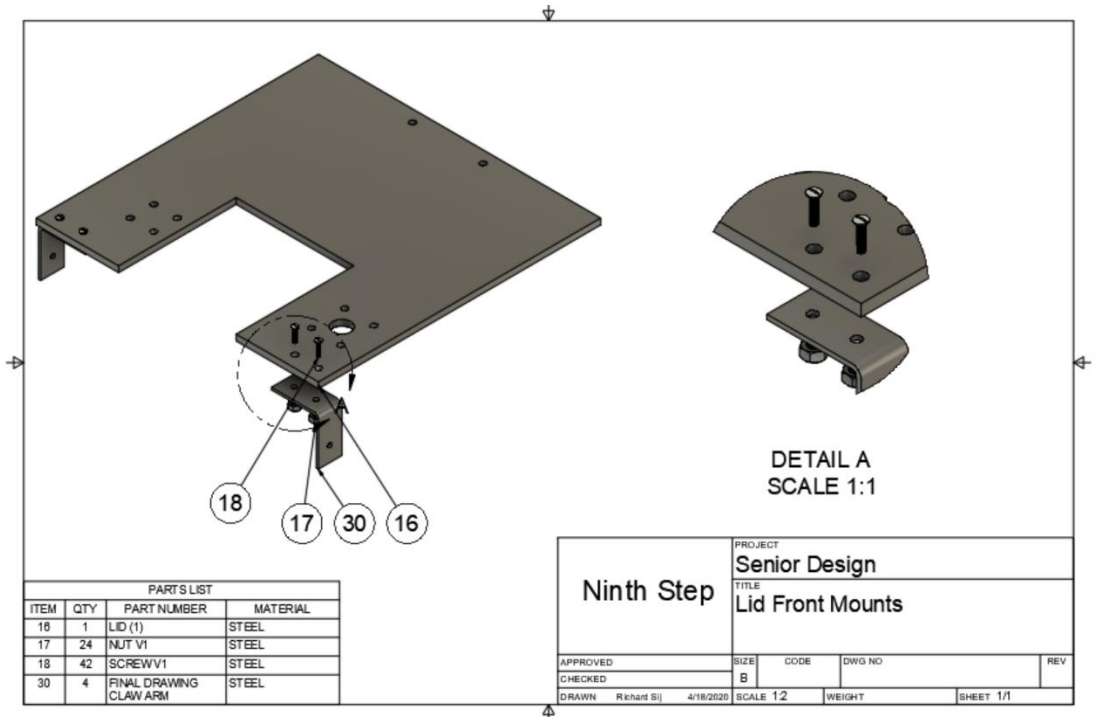
class Servos{
  public:
    bool isClawClosed;
```

```
    bool isFlippersClosed;
    bool isArmLowered;

    Servos(int clawPin,  int leftFlipper, int rightFlipper);
    begin();
    raiseArm();
    lowerArm();
    openClaw();
    closeClaw();
    openFlippers();
    closeFlippers();
  private:
    LSS _arm;
    int _clawPin, _leftFlipper, _rightFlipper;
    Servo _claw, _lFlipper, _rFlipper;
    int _openClawAngle, _closeClawAngle; //claw closed and open angles
    int _openFlipperAngle, _closeFlipperAngle; //flippers closed and open angles
};

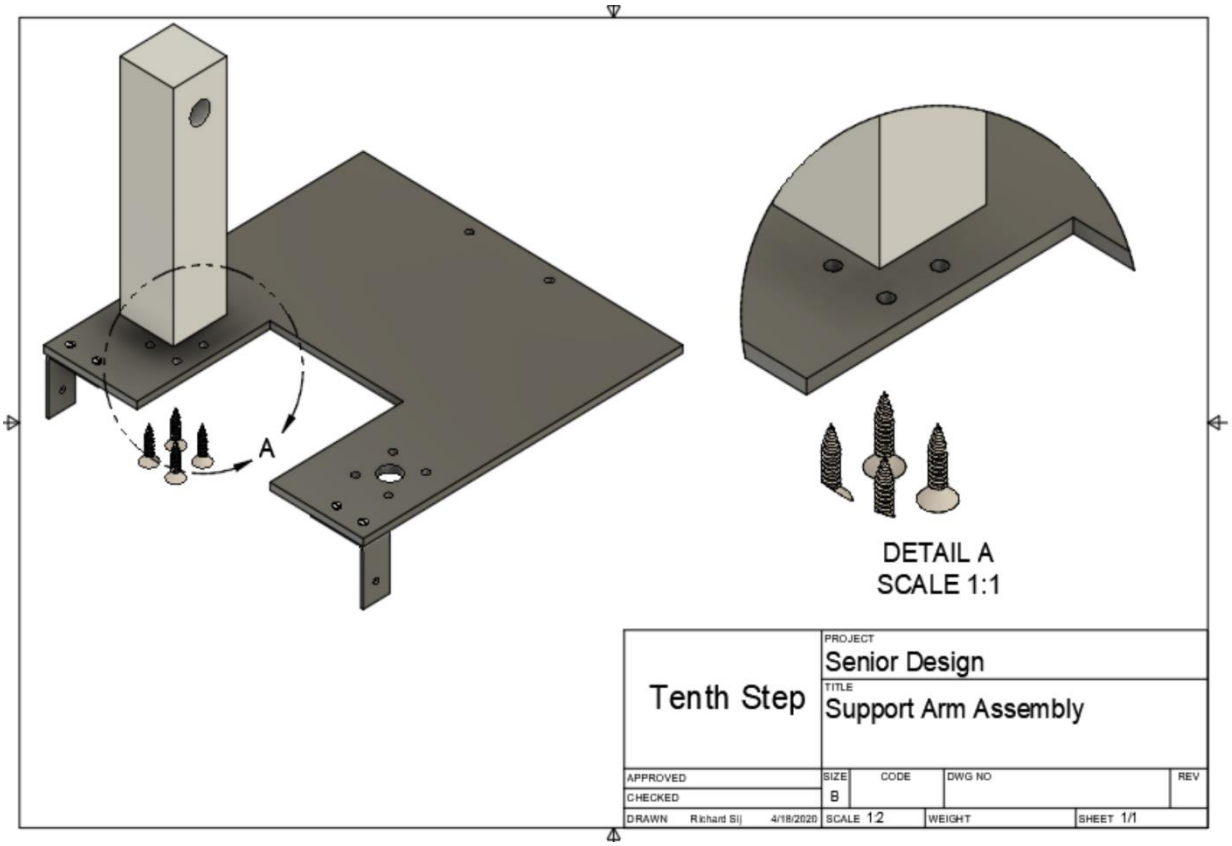
#endif
```

Step 11. Base Cover Mounting

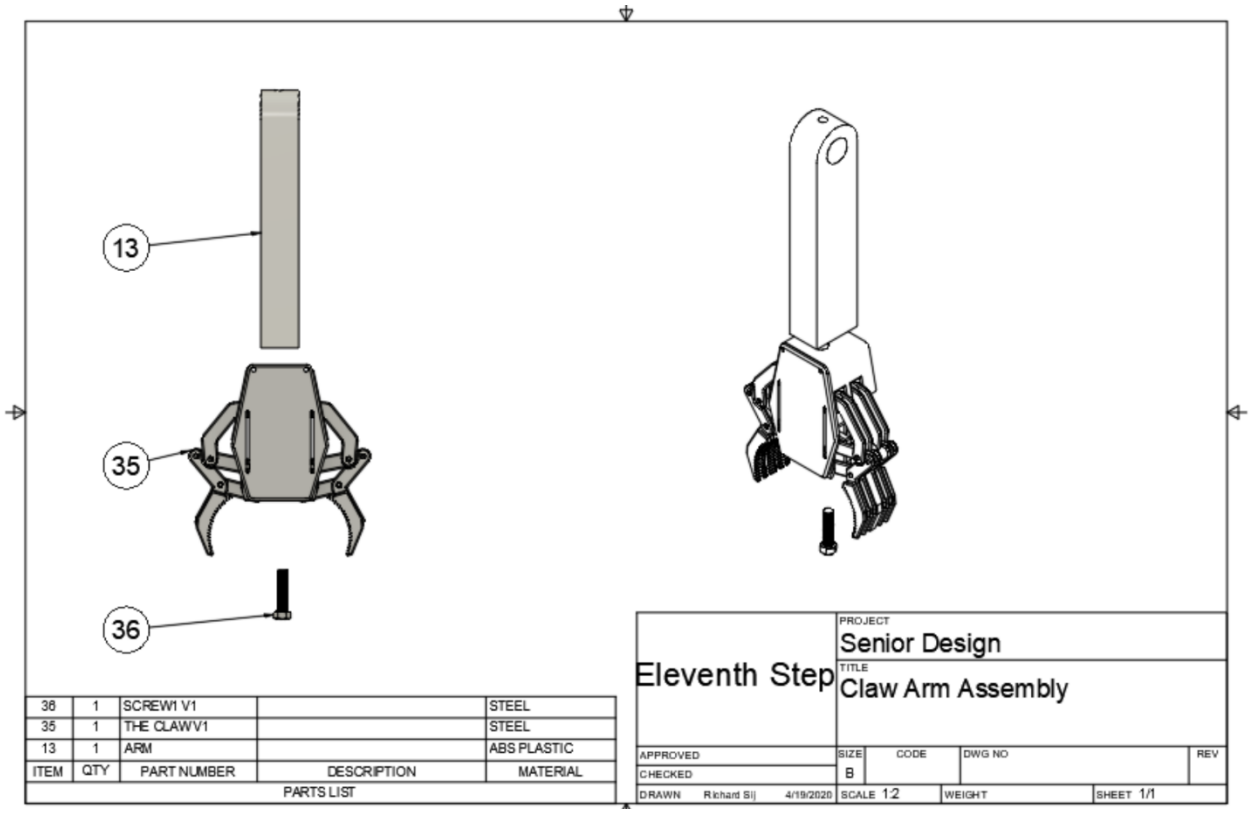


PARTS LIST			
ITEM	QTY	PART NUMBER	MATERIAL
16	1	LID (1)	STEEL
17	24	NUT V1	STEEL
18	42	SCREWV1	STEEL
30	4	FINAL DRAWING CLAW ARM	STEEL

Step 12. Mount the Claw Support to the Base Cover



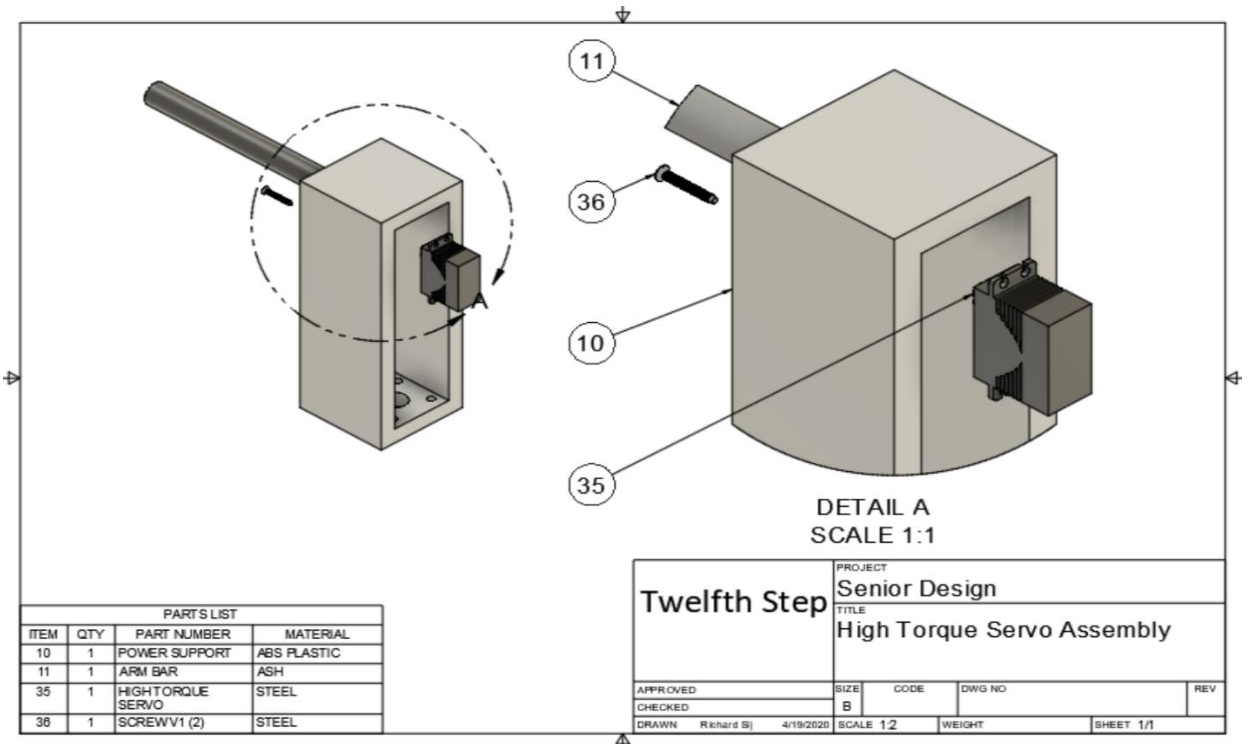
Step 13. Mount the Arm and the Claw



ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
36	1	SCREW1 V1		STEEL
35	1	THE CLAWV1		STEEL
13	1	ARM		ABS PLASTIC

PARTS LIST

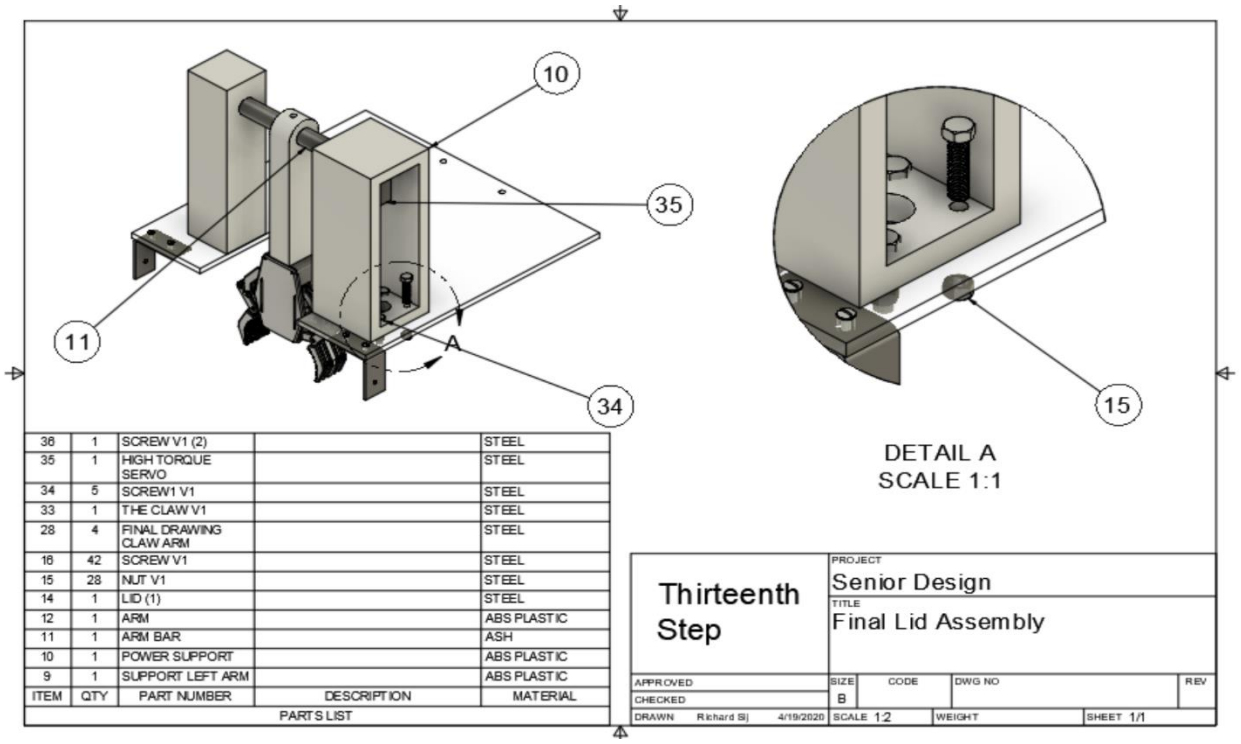
Step 14. Mount of the Arm Bar, High Torque Servo and Claw Arm Support



PARTS LIST

ITEM	QTY	PART NUMBER	MATERIAL
10	1	POWER SUPPORT	ABS PLASTIC
11	1	ARM BAR	ASH
35	1	HIGH TORQUE SERVO	STEEL
36	1	SCREWV1 (2)	STEEL

Step 15. Connect the Arm Support and Cover Base



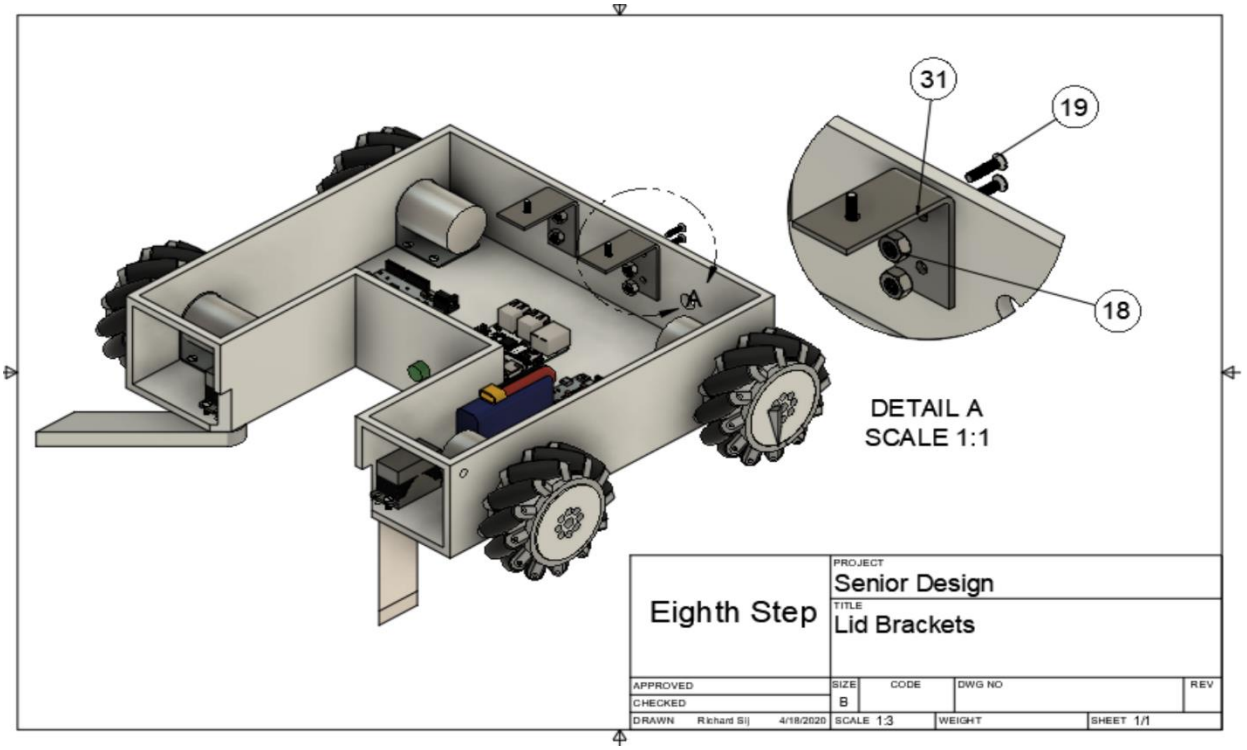
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
36	1	SCREW V1 (2)		STEEL
35	1	HIGH TORQUE SERVO		STEEL
34	5	SCREW1 V1		STEEL
33	1	THE CLAW V1		STEEL
28	4	FINAL DRAWING CLAW ARM		STEEL
16	42	SCREW V1		STEEL
15	28	NUT V1		STEEL
14	1	LID (1)		STEEL
12	1	ARM		ABS PLASTIC
11	1	ARM BAR		ASH
10	1	POWER SUPPORT		ABS PLASTIC
9	1	SUPPORT LEFT ARM		ABS PLASTIC

PARTS LIST

Thirteenth Step

PROJECT: Senior Design
TITLE: Final Lid Assembly
APPROVED
CHECKED
DRAWN Richard Sij 4/19/2020
SIZE B | CODE | DWG NO | REV
SCALE 1:2 | WEIGHT | SHEET 1/1

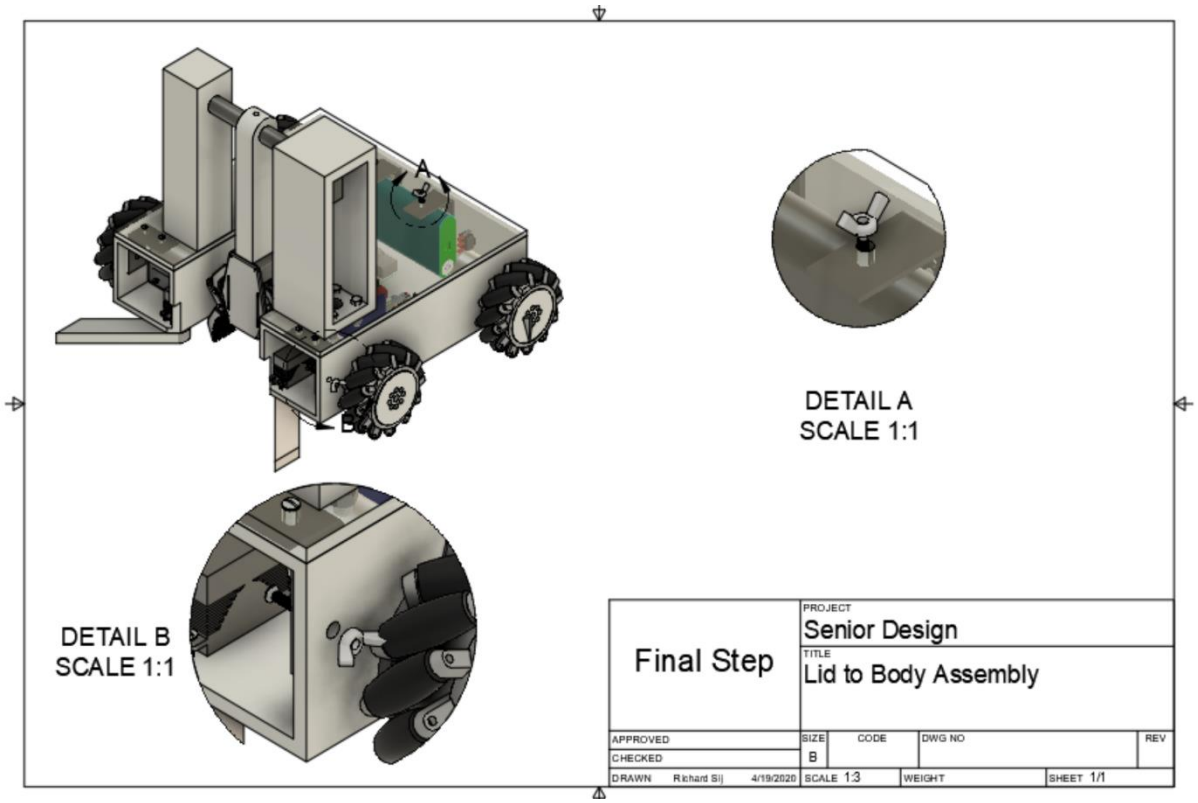
Step 17. Connect the Support to the Base



Eighth Step

PROJECT: Senior Design
TITLE: Lid Brackets
APPROVED
CHECKED
DRAWN Richard Sij 4/18/2020
SIZE B | CODE | DWG NO | REV
SCALE 1:3 | WEIGHT | SHEET 1/1

Step 18. Connect the Arm Support and Cover Base



Step 19. State Class Code

File: state.py

```
import os

#State class containing active arduinos and active running components
class StateManager():

  #Determine OS
  OS="linux"
  if os.name == "nt":
    OS = 'windows'

  def __init__(self, num_devices=2):
    self.nodes = {}                          #an arduino
    self.active = 'pi'                  #
    self.added_devices = {} #

    self.valid_ids = [str(i) for i in range(num_devices)]

  def add_node(self, arduino, device_name, baudrate=9600):
    _id = None
    while _id not in self.valid_ids:
      _id = self.getId(arduino)
    self.added_devices.update({_id:device_name})

    #add new node
    arduino.baudrate=baudrate
```

```
  newNode = {str(_id):arduino}
  self.nodes.update(newNode)
  return _id

def getId(self, arduino):
  # return '1'
  '''gets message from arduino serial.Serial() input stream. interprets the message as an _id'''
  # print("Attempting to read line")
  read_serial=arduino.readline()
  # print("read serial",read_serial)
  _id = read_serial.decode('utf-8') #returns initial message from arduino

  if StateManager.OS == "windows":
    _id = _id[:-2]  #remove \r\n character
  else:
    _id = _id[:-2]  #remove \r\n character

  print('Got id', _id)
  return _id

def __str__(self):
  return str(self.added_devices)
```

Step 20: Arduino Code for Communication with Raspberry Pi

File: Ard,ino

```
#include <Servo.h>
#include "Motion.h"
#include "Servos.h"
#define MAX_BUFFER_LENGTH 2



const int NAME = 1;
String active = "pi";
char buffer[MAX_BUFFER_LENGTH]; //instruction buffer

//Attach DC motors
Motion motors(
  3,          //left front wheel
  4,          //left back wheel
  2,          //right front wheel
  1           //right back wheel
);

//Attach Servos for claw and flippers
Servos servos(
  9,          //claw pin
  4,          //left flipper
  3           //right flipper
);
```

```
void setup() {
  Serial.begin(9600);
  motors.begin();
  servos.begin();
  servos.openClaw();
  servos.openFlippers();

  //wait for Pi to configure COMs
  while(active == "pi"){
    Serial.println(NAME);
    delay(1000);
    if(Serial.available() > 0){
      active = "ard";
    }
  }

  // Serial.println("Setup Complete.");
}

void loop() {
  // testServos();
  // testMotion();
  recieveData();
}

//received data from Raspberry Pi
void recieveData(){
```

```
  int len = 0;
  if (Serial.available() >= 0){
    while(Serial.available()>len){//is a char available?
      len = Serial.available();
      delay(5); //delay 5ms for new characters to arrive
      if(len>10){
        break;
      }
    }
  }
  String instr = ""; //instruction
  if(len>0){
    Serial.readBytes(buffer, len); //get the characters
    Serial.println(len);
    for(int i = 0; i<len; i++){ //exclude terminal character
      instr += String(int(buffer[i])-48);
    }
    Serial.println(instr);
    Serial.println(instr.toInt());
    runCommand(instr.toInt());
  }
}

void runCommand(int action){
  switch(action%48){
    case 0:             //STOP MOTORS
      motors.stop();
      break;
```

```
case 1:          //MOVE FORWARD
  motors.forward();
  break;
case 2:          //MOVE BACKWARD
  motors.backward();
  break;
case 3:          //ROTATE RIGHT
  motors.turnRight();
  break;
case 4:          //ROTATE LEFT
  motors.turnLeft();
  break;
case 5:          //CLOSE FLIPPERS
  servos.closeFlippers();
  break;
case 6:          //OPEN FLIPPERS
  servos.openFlippers();
  break;
case 7:          //CLOSE CLAW
  servos.closeClaw();
  break;
case 8:          //OPEN CLAW
  servos.openClaw();
  break;
case 9:          //RAISE ARM
  servos.raiseArm();
  break;
case 10:          //LOWER ARM
```

```
      servos.lowerArm();
      break;
   }
}

void testMotion(){
  runCommand(1);
  delay(1000);
  runCommand(0);
  delay(1000);
  runCommand(2);
  delay(1000);
  runCommand(0);
  delay(1000);

  runCommand(3);
  delay(1000);
  runCommand(0);
  delay(1000);
  runCommand(4);
  delay(1000);
  runCommand(0);
  delay(1000);
}

void testServos(){
  servos.closeClaw();
  delay(2000);
```

```
  servos.openClaw();
  delay(2000);
  servos.closeFlippers();
  delay(2000);
  servos.openFlippers();
  delay(2000);
  servos.raiseArm();
  delay(2000);
  servos.lowerArm();
  delay(2000);
}
```

Step 21. The Code for the Entire System

```
import time
import inputs
from victor import Brain

brain = Brain(2)

print(inputs.devices.gamepads)

pads = inputs.devices.gamepads

if len(pads) == 0:
    raise Exception("Couldn't find any Gamepads!")

action_space = {
    'ABS_HAT0X 0':  "STOP",
    'ABS_HAT0X -1': "TURN_LEFT",        #LEft  D Pad
    'ABS_HAT0X 1':  "TURN_RIGHT",       #Right D Pad
    'ABS_HAT0Y 1':  "MOVE_BACKWARD",    #Down  D Pad
    'ABS_HAT0Y -1': "MOVE_FORWARD",     #Up    D Pad
    'ABS_HAT0Y 0':  "STOP",
    'BTN_TL 1':     "OPEN_CLAW",        #RIGHT BUMPER
    'BTN_TR 1':     "CLOSE_CLAW",       #LEFT BUMPER
    'BTN_SOUTH 1':  "CLOSE_FLIPPERS",   #A
```

```
    'BTN_EAST 1':   "OPEN_FLIPPERS",    #B
    'BTN_NORTH 1':  "RAISE_ARM",        #X
    'BTN_WEST 1':   "LOWER_ARM",        #Y
}


print("Ready to receive")
while True:
    events = inputs.get_gamepad()
    for event in events:
        # print(event.ev_type, event.code, event.state)
        key = event.code + ' ' + str(event.state)
        action = action_space.get(key)
        print('key', key, 'Action', action)
        if action is not None:
            ard = "0"
            if "ABS" not in key:
                ard = "1"
            brain.send_act(action, ard)
```

Step 22. Full Body

