

4/26/2019



Team 305: Southeast Con 2019
Hardware Competition Evidence Book

Daniel Delgado, Kyle Voycheske, Chase Sapp, Fabio Trinidad, Chengdong Yuan

FAMU-FSU College of Engineering 2525 Pottsdamer St. Tallahassee, FL. 32310

Abstract

The IEEE Southeast Con is a regional annual robotics competition that is hosted in Von Braun Center in Huntsville, Alabama this year, with the competition scheduled from April 11 through April 14, 2019. The robotics competition for this year involves designing an autonomous robot that can clean up debris, wooden cubes and plastic spheres, on a playing field. Points are gathered by completing orbits around the field, collecting the debris, sorting debris, and returning to the home base. Subsequent rounds after the first round are similar, with the only exception being that another robot will be sharing the field with the team, as they also attempt to gather the debris. The team has made their robot to be within the walls of the competition's rules, while optimizing it to gather the most possible points. The robot follows a planned route to gather the objects, and if necessary, it will go off route to align with the debris. The robot uses two brushes, spinning inward to the center of the robot, to gather objects. Once the robot has consumed an object, the object proceeds to the internal elevator. During this step, the robot sorts the debris into a sorting cube that has four color-coded cavities. After the robot has gathered all possible cubes and spheres, the robot performs counter-clockwise orbits until the last minute of the round. These orbits are vital to make up any loss points. During the final minute of the round, the robot places the debris into the proper color specific homes. Lastly, the robot returns home and raises the FAMU-FSU College of Engineering banner to inform the team that it has finished the round. The potential impact for this robot design is for public areas, gathering small-scale pieces of trash and sorting it by material.

Keywords: Robotics, Southeast Con Hardware Competition, and Color Sorting

Acknowledgement

A special thanks to the FAMU-FSU College of Engineering in sponsoring the Southeast Con 2019 Hardware Team. Providing the funds for the team to gather the proper material to design the robot. Allowed access to the required equipment to sculpt the robot as the team designed it.

A special thanks to reviewer Dr. Harvey for taking the time to take the robotics team to Huntsville Alabama for the competition, and valuable experience in past competitions. Also, a special thanks to reviewer Dr. Hooker, leaving an open door for the copious amount of questions that the team had over the year, supplying equipment when needed, crafting PCB boards, and support the overall team endeavors. Lastly, a special thanks to Dr. DeBrunner for her constructive feedback and support throughout the whole engineering process.

The project team would like to give a special thanks to the Innovation Hub for their tremendous amount of help with the 3D printing of various robotic components, such as the frame, gears, and bumper. The Innovation Hub provided professional service in making sure that each and every print came as designed, and guided the team in improving the preparation work for 3D printing.

Table of Contents

Abstract.....	2
Acknowledgement	3
List of Tables	8
List of Tables Figures	10
Chapter One: EEL 4914C	11
1.1 Southeast Con Project Scope.....	11
1.2 Southeast Con 2019 Team Customer Needs.....	13
Needs and Requirements.....	13
Needs/Customer Statements	14
Requirements	14
Constraints	15
1.3 Functional Decomposition.....	15
Movement	15
Obstacle Avoidance	16
Transport Debris	16
Navigation.....	17
1.4 Targets for the Southeast Con 2019 Team 305.....	18
1.4.1 Motion and Frame Module	18

1.4.2 Route Clearing Algorithm.....	21
1.4.3 Sorting Software and Hardware Module	22
1.4.4 Return Home Module	24
1.4.5 Detection and Avoidance Module	26
1.5 Concept Generation	28
1.5.1 Motion and Frame.....	29
1.5.2 Route Clearing Algorithm.....	33
1.6 Concept Selection	38
1.6.1 Movement Hardware and Software Selection	38
1.6.2 Route Clearing Algorithm Solution Selection	42
1.6.3 Microcontroller Selection	45
1.6.4 Sorting Hardware and Software Selection.....	47
1.6.5 Storage Selection	51
1.6.6 Return home algorithm	53
1.6.7 Wheels Selection.....	55
1.6.8 UFO Avoidance Algorithm Selection.....	57
1.7 Southeast Con 2019 Spring Plan.....	60
Chapter Two: EEL 4915C	60
2.1 Targets for the Southeast Con 2019 Team 305.....	60

2.1.1 Project Plan	60
2.1.2 Build Plan.....	63
<i>Final Design</i>	66
<i>Results</i>	71
2.2 Testing and Validation	72
2.2.1 Testing.....	72
2.2.2 Validation.....	75
2.3 Scholarship in Practice.....	76
2.3.1 Pixy2 Camera.....	76
2.3.2 DC Motor	78
2.3.3 IR Sensor and Encoder.....	78
2.3.4 3D Printing.....	80
2.3.5 Separating the Bumper from the Base	81
2.3.6 Work in Parallel not in Series	81
2.4 Conclusions.....	82
Future work.....	83
Appendices.....	84
Appendix A: Code of Conduct	84
A.1 Mission Statement.....	84

A.2 Roles.....	84
A.3 Communication.....	86
A.4 Team Dynamics	87
A.5 Ethics.....	88
A.6 Dress Code	88
A.7 Weekly and biweekly Tasks	88
A.8 Decision Making.....	88
A.9 Conflict Resolution	89
A.10 Statement of Understanding.....	90
Appendix B: Function Decomposition	91
Appendix C: Target Catalog	91
Appendix D: Testing Results	93
Appendix E: Operation Manual	94
Appendix F: Software.....	104
Appendix G: Spring 2019 Project Plan.....	144
Appendix H: Bill of Materials	147
Appendix I: Risk Assessment.....	148

List of Tables

Table 1- Costumer Needs.....	14
Table 2- Costumer Requirements	14
Table 3-Project Constraints.....	15
Table 4-Movement Decomposition	15
Table 5-Obstacle Avoidance.....	16
Table 6-Transport Debris	16
Table 7-Navigation	17
Table 8-Motor Comparison.....	30
Table 9-Frame Material Comparison.....	32
Table 10-Battery Comparison.....	33
Table 11-Different Algorithms Comparison.....	35
Table 12-Sensor Comparison.....	38
Table 13-Desired Material Features.....	39
Table 14-Material Considered Pugh Chart	40
Table 15-Battery Requirements	40
Table 16-Battery Feature Pugh Chart	41
Table 17-Motor Requirements	41
Table 18-Motor Feature Pugh Chart	42
Table 19-Route Clearing Requirements.....	43
Table 20-Route Clearing Pugh Chart.....	44
Table 21-Microcontroller Requirements.....	46
Team 305: Southeast Con 2019 Robotics Competition Team	8

Table 22-Microcontroller Feature Pugh Chart.....	47
Table 23-Gathering Criteria.....	48
Table 24-Gathering Solution Pugh Chart	49
Table 25-Sorting Criteria	50
Table 26-Sorting Criteria Pugh Chart.....	51
Table 27-Sorting Solution Requirements.....	52
Table 28-Sorting Solution Pugh Chart.....	52
Table 29-Localization Criteria.....	54
Table 30-Localization Criteria Pugh Chart.....	55
Table 31-Wheel Criteria	56
Table 32-Wheel Criteria Pugh Chart	57
Table 33-Avoidance Solution Criteria	58
Table 34-Avoidance Solution Pugh Chart	59
Table 35- Build Plan breakdown	63
Table 36-Key specs of the robot’s performance	71
Table 37- Southeast Con 2019 Test Results	73
Table 38-Target Catalog	91
Table 39- Southeast Con 2019 Test Results	93
Table 40-Spring Planning Chart	144

List of Tables Figures

Figure 1-Work Breakdown	60
Figure 2-Exploded view of the drive assembly	67
Figure 3-Exploded view of the elevator/control assembly	68
Figure 4-Exploded view of the bumper/brush assembly	69
Figure 5-Exploded view of the complete robot	70
Figure 6-Southeast Con 2019 Function Decomposition.....	91
Figure 7- Base Assembly	96
Figure 8-Bumper Assembly	97
Figure 9-Elevator Assembly	98
Figure 10-Overall Robot Assembly	99
Figure 11-Overall Circuit Design	100

Chapter One: EEL 4914C

1.1 Southeast Con Project Scope

The Southeast Con 2019 Hardware Team project scope was bounded based on the limitation set forth by the competition. The team went through the rules and regulations set forth selecting the rules that would affect the design process and limits of the robot.

- The robot must act on its own, no cables or communication can be sent to the robot.
- The size of the robot must be within 22.86-cm (9") x 22.86-cm (9") x 27.94-cm (11") [LxWxH].
- The robot must at least be one unit, it cannot break apart.
- The robot may wave a flag for extra points.
- The robot must be able drive around a carpeted plane.
- The bumper must present a vertical surface at least 2.54-cm (1") high and cover, at a minimum, the space from 3.81-cm (1 ½") to 6.35-cm (2 ½") above the playing field.
- Robots may be modified physically, reprogrammed, and/or recharged between each match. However, any physical modification will require a re-inspection for safety and overall size compliance
- A robot may not operate in a manner that excessively damages the playing field, causing stoppage of the competition, or require repair of the field for the next competition.
- The bumper may be of any shape around the robot and need not be outwardly convex on all surfaces but must not have any radius of curvature less than 1-cm.
- Pyrotechnics, compressed gas, hydrocarbons, toxic or corrosive materials are not allowed.

- The robot needs to recognize specific flashlight from other light.
- The robot must be able to deliver both cubes and balls.
- The robot needs to be steady enough with block.
- The robot needs a brake system react fast.
- The robot might be more efficient with a bucket to store.
- The robot must be completely autonomous.
- The robot must be able to complete the required tasks within 3 minutes.
- In the later rounds the robot must be able to avoid another robot also picking up trash.
- The robot will need to be able to differentiate between shapes and color.
- The on-board flag must contain the school logo (if affiliated with a university), State, territory or US flag.
- Robots must not damage the playing field, halt competition or require repair of the playing field.
- Each robot must have a bumper that surrounds 80% of its perimeter in a continuous stretch. This bumper must be the outermost structure at all times when the robot is moving.
- Any part of the robot that is deemed by contest officials to be dangerous or injurious to the participants, audience, staff, playing field or surroundings will result in disqualification.
- The robot may not exceed beyond its maximum allowable dimension and may not include any detachable or remoted extensions. In addition to meeting safe operation requirements, a robot will need to pass the size qualification.

1.2 Southeast Con 2019 Team Customer Needs

Needs and Requirements

The needs and requirements cover the specifications for a competing robot in the Southeast Con 2019 Hardware Competition. The needs break down the competition objective into tasks that the robot can do. The requirements take the needs and specify exact functions that the robot must complete to be competitive in the competition. Constraints was included to highlight the “Robot Specification” section of the competition rules, which decides if the robot may be permitted into the competition. This information was based on the hardware competition rules version 1.2 for student programs found on the IEEE website.

Objective reads:

“To clear orbital space debris while avoiding Spacetels and return to home base within the time limit of three minutes. The score is determined by a number of factors including: 1) the number of complete playing field orbits, 2) the number and type of space debris cleared, 3) sorting cleared debris, 4) returning to assigned corner square (home base), and 5) avoiding collision with Spacetel.” [1]

Needs/Customer Statements

Table 1- Costumer Needs

<i>Need Number</i>	<i>Objective Statement</i>	<i>Interpreted Need</i>
1	the number of complete playing field orbits	Must be able to move in a counter-clockwise direction.
2	the number and type of space debris cleared	To gather more points, the robot must be able to remove space debris from zone 2 to zone 1.
3	sorting cleared debris	Sort the space debris by color and deposit it in it's their respective color matching corner.
4	returning to assigned corner square (home base)	The robot must be able to return home from its current position.
5	avoiding collision with Spacetel	The robot must be able to avoid structures on the playing field.

Requirements

Table 2- Costumer Requirements

<i>Requirement Number</i>	<i>Need number</i>	<i>Requirement</i>
1	1	The robot must be able to move
2	1	Be able to turn on the playing field
3	1	Rotate counter-clockwise
4	2	Remove space debris from zone 2 to zone 1
5	3	Gather space debris
6	3	Sort space debris
7	4	Find a route to home base
8	5	Avoid structures on the playing field

Constraints

Table 3-Project Constraints

<i>Number</i>	<i>Constraints</i>
1	Robot must be autonomous
2	Power supply must be self-contained
3	Size restriction: 22.86-cm (9") x 22.86-cm (9") x 27.94-cm (11")
4	When not in motion the robot may extend f 7.62-cm (3") by 7.62-cm (3") in length and width direction at a time. However, the extension must always be connected to the robot
5	No weight limit
6	No pyrotechnics, compressed gas, hydrocarbons, toxic or corrosive materials are allowed.
7	If going for the bonus points for the flag, the flag must have the university logo.
8	The robot cannot damage the playing field.
9	A bumper must be present, that covers 80% of its perimeter in a continuous Stretch, is the outer most structure, and a vertical surface at least 2.54-cm (1") high and cover, at a minimum, the space from 3.81-cm (1 1/2") to 6.35-cm (2 1/2") above the playing field.
10	Team members must be IEEE members.

1.3 Functional Decomposition

Movement

Table 4-Movement Decomposition

Module	Propulsion
Inputs	PWM Power
Outputs	Wheel rotation
Functionality	Make the robot move as the PWM with command

Module	Steering
Inputs	Direction signal Power
Outputs	Wheel rotation
Functionality	Steer the robot turn to the right direction

Obstacle Avoidance

Table 5-Obstacle Avoidance

Module	Identify
Inputs	The color and the shape of objects
Outputs	The signal for different objects
Functionality	Recognize the objects around the robot in order to tell how to move

Transport Debris

Table 6-Transport Debris

Module	Locate Debris
Inputs	The location of each debris
Outputs	The route to approach the target debris
Functionality	Get close to target debris in order to move it

Module	Identify Debris
Inputs	The shape and color of target debris

Outputs	The signal for the target debris
Functionality	Let the robot know what kind of debris it is

Module	Grab debris
Inputs	The debris on the carpet field
Outputs	The debris in the storage of the robot
Functionality	Put the debris inside the storage to move it

Module	Move debris
Inputs	The debris in the storage of the robot
Outputs	The debris in a specific area
functionality	Move to the specific area and unload the debris

Navigation

Table 7-Navigation

Module	Localization
Inputs	The location of the robot and the destination
Outputs	The coordinate position of the robot and destination
functionality	Transfer the location into information which can be used by the robot
Module	Pathfinding

Inputs	The information of the location of starting point and terminal point
Outputs	The fastest route from starting point to terminal point
functionality	Find the fastest route to the destination without breaking the rule

1.4 Targets for the Southeast Con 2019 Team 305

1.4.1 Motion and Frame Module

In this section of the Target Summary covers the goals for the motion and frame module of the FAMU-FSU Southeast Con 2019 Robot. The need for this module was to move around the playing field in counter-clockwise orbit. The targets will breakdown the needed values for propelling, turning and wheel configuration of the robot. Also, design the frame to support future modules and meet the requirements of the Southeast Con Hardware competition 2019. This module is necessary to let the robot interact with the playing field.

Target 1 quantifies the necessary to move around the field at a competitive speed. This target is necessary to ensure that the robot can move around the field at its fastest rate of speed without tipping over. The marginal value for maximum speed of the robot was selected by calculating the top speed without slipping with a marginal error of 20%. The top speed before slipping was found by using the centripetal force ($f = \frac{mv^2}{r}$) and friction force formula ($f = \mu_f mg$) and solving for the velocity. The coefficient of friction was assumed to be 0.6, because average coefficient of friction was that. After finding the maximum speed (4.484 ft/s) it was given an error of 20 percent in case the carpet is more slippery then we expected. Leading to the value of 3.6 ft/s for the maximum speed of the robot. To assess whether the target has been successfully

achieved, the robot will be placed on the practice field and drive across the field. The time to cross the field will be recorded and the velocity will be calculated. The importance of this target is rated at 5 because this function allows the robot to interact with the competition. Failure to meet this target would result in moving at low rate of speed or not moving at all, causing the opponents to gather more space debris.

Target 2 quantifies an effective method of turning to maintain a steady counter-clockwise orbit, and responsive enough to avoid planned and unplanned objects. This target is necessary to ensure the robot can respond to a changing environment as well as maintain a steady path to collect debris. The marginal value of $\pi/3$ radians/sec was selected to ensure that the robot can readily turn to avoid obstacles. To assess whether the target has been successfully achieved, several measurements of the robot's angular velocity will be measured. The importance of this target is rated at 3 because the robot being able to respond to an unknown and changing environment is one of the core ideas of the competition. Failure to meet this target would result in missing debris and collisions with objects.

Target 3 quantifies the effectiveness of the wheel configuration. This target is necessary to ensure that the robot is efficient and capable of reliably navigating the playing field. The marginal value for be 3.5 ft/s was chosen to ensure the robot is capable of quickly traversing the playing field. This measurement will be taken when the robot is traveling in a straight path, after accelerating for 3 seconds, and the wheels suffer no slippage. To assess whether the target has been successfully achieved, several tests of different wheel configurations and combinations will be done, and subsequently compared to find an efficient wheel configuration. The importance of this target is rated at 4 because ensuring the robot has sufficient traction to navigate the playing

field is critical to the robots performance in the competition. Failure to meet this target would result in slow, inefficient robot that will struggle to keep track of its current position, and result in an extreme disadvantage during the competition.

Target 4 quantifies the regulated dimensions of FAMU-FSU Southeast Con Team's robot. This target is necessary to ensure that the final design of the Southeast Con robot qualifies to compete in the competition. The marginal value chosen was based on the maximum size of the robot given by the Southeast Con Hardware Competition. As specified below, the final robot dimension goal is set at 8.5-in x 8.5-in x 10-in, as this dimension allows for a robot design that is within the regulated size and prevents the downgrade of some sensors and microcontrollers that are essential for developing a properly working robot. To assess whether the target has been successfully achieved, measurements from a CAD design will be initial used to verify that the dimensions of the components chose for the final design will retain the robots desired dimension. Once this initial measure is a success, another measurement will be conducted after the final robot design is assembled. If the assembled robot remains within the desired dimensions, then it would be concluded that this target has been met. The importance of this target is rated at 5 because failing to meet the specified size dimensions would make the robot ineligible to compete in the competition. Therefore, failure to meet this target would put the FAMU-FSU Southeast Con Team of the competition, without having competed in any of the rounds.

1.4.2 Route Clearing Algorithm

This section of the Target Summary covers the goals for the route clearing algorithm of the FAMU-FSU Southeast Con 2019 Robot. The need for this module is to create an effective route for the robot to travel while searching for debris to clear. The targets will breakdown the needed values to create an effective route to find and remove debris from the playing field. Included in this module are the desired values of static object avoidance. This module is necessary to provide the route the robot will search to earn points.

Target 5 quantifies an efficient path that is necessary to quickly search the playing field for debris to clear. This target is necessary to ensure the robot has a path to search to clear debris from the field. This value was chosen based on the time limit of a single round of competition. The selected value of 2 minutes to complete a full search of the field without having to deviate to collect debris was chosen to ensure the robot had enough time to collect debris and return home. The primary assumption is that if the robot can complete the route within the set value, then it will have enough time to gather debris, avoid obstacles, and return to home base. To assess whether the target has been successfully achieved, the robot will be placed in the playing field and allowed to travel the route while being timed. The importance of this target is rated at a 3 because searching the playing field for debris to remove will be the primary source of points for the Southeast Con competition. Failure to meet this target would result in the robot traveling a route that will miss debris, however if some debris is missed by the robot it is still possible for the team to win the competition.

Target 6 quantifies the avoidance of planned objects. This target is necessary to ensure that the team does not lose points for hitting structures or damaging the playing field. The

marginal value was selected to ensure the robot is capable of safely avoiding planned obstacles. The value of 6 inches was chosen to ensure the robot will not accidentally hit any planned objects. Through the assistance of internal and external sensors the robot will keep track of its location relative to the planned obstacles and avoid them. To assess whether the target has been successfully achieved, the robot will be set into the playing field at various position, must localize where it is, and return home without any collisions. The importance of this target is rated at 3 because if the robot hits any object the points lost for the collision will greatly increase the difficulty of achieving advancement to the next round. Failure to meet this target would result in the FAMU-FSU Southeast Con team at a severe disadvantage, and if the damage to the playing field is extensive, possible disqualification.

1.4.3 Sorting Software and Hardware Module

In this section of the Target Summary covers the goals for the sorting component of the FAMU-FSU Southeast Con 2019 Robot. The need for this module was to find a way to take in the space debris and sort it within the robot. The targets will breakdown the needed values for gathering, sorting and storing the space debris within zone 2 of the playing field. This module is necessary to increase the possible points that the robot can earn during the competition.

Target 7 quantifies the gathering mechanism on the robot. This target is necessary to ensure that the gathering mechanism has obtain enough space debris to make it competitive during the competition. The marginal value was chosen so that if the opponent robot gathers about half the amount of space debris available, then FAMU-FSU Southeast Con 2019 robot gathers one more than half. This would put the FAMU-FSU Southeast Con 2019 Team ahead by two space debris. During the competition there will be 12 space debris in the zone 2 that the

teams can gather. To have the marginal amount would be 7 pieces of space debris. This assumes that the team will go to the second round which then they will compete against another robot. To assess whether the target has been successfully achieved, the robot will be placed in a practice field (design to mirror that within the competition) where it will try to gather 7 pieces of space debris. The importance of this target is rated at 4 because during the competition there are other ways to gather points, however it is necessary for sorting the space debris. Failure to meet this target would result in unable to properly sort the space debris, but the team can still win the competition without this module.

Target 8 quantifies the sorting mechanism on the robot. This target is necessary to ensure that the sorting mechanism takes the space debris from the gathering mechanism, organize them, and then deposit the space debris into the storage unit. The marginal value was chosen so that the robot would be able to sort the debris and be competitive to against the other competitors. The marginal value is 80% of the space debris is sorted correctly by color. The shape of the space debris does matter how it is sorted because there points gain for sorting base on that. This percentage was selected under the assumption that the opponent gathers half plus one of the available space debris, and return all of them home. If the opponent robot gathers 7 pieces of space debris (70 points) and returns home with them (70 points), then the FAMU-FSU Southeast Con 2019 robot must earn 140 points. If it gathers the remaining 5 pieces of space debris and return home, then it would have 100 points. To catch back up, the team must sort four out of five space debris and put it the proper corner. To assess whether the target has been successfully achieved, the robot would consume five pieces of space debris then sort them. The importance of this target is rated at 2 because the robot can still gather points without this function, but

increases the maximum points the robot can gather. Failure to meet this target would result in not storing sorted cubes.

Target 9 quantifies the necessary storage container for sorted space debris. This target is necessary to ensure that the space debris collected from the gather mechanism is held within the robot after being sorted by the sorting mechanism. The marginal value for this target was selected based on the size of the spherical space debris and wiggle room to slide into the container. The cube space debris was not the focus on the size of the storage container because the cube space debris (2") have smaller width compared to the pit balls (2.5"). An extra 1/2" is applied to the width, height and depth to prevent space debris from seizing in the storage bin. Resulting internal space of the storage bin should be able to support a volume of 242 in³ or the maximum amount of space debris, 12 pieces. 12 pieces of space debris was selected because the team assumes that robot will gather all the space debris during the competition for the module. To assess whether the target has been successfully achieved, 12 pieces of space debris will be place in the storage container to test that it can hold maximum space debris. The importance of this target is rated at 3 because even if the space debris is not able to be sorted by the sorting mechanism, the robot can still carry the space debris to home base for extra points. Failure to meet this target would result in unable to store the space debris, leading to maximum total points that can be earned during the competition drops by 240 points.

1.4.4 Return Home Module

In this section of the Target Summary covers the goals for the home returning of the FAMU-FSU Southeast Con 2019 Robot. The need for this module was to have the robot go back to the home base. The target will breakdown the needed value of how to find the best route to go

back to home base which is one of the colored corners where it begins. This module is necessary to get the points for getting back to home base and put the matched debris to colored corner.

Target 10 quantifies the dropping off mechanism of the robot. This target is necessary to ensure that the dropping off mechanism is able to drop off the sorted debris to a specific colored corner. The marginal value was chosen based on that it counts really much to deliver the matched debris to a colored corner, if the debris cannot be dropped off properly, sorting and gathering the robot did before would be useless. To assess whether the target has been successfully achieved, some similar debris will be loaded on the robot, and we will test the percentage of successful dropping off. The importance of this target is rated at 3 because only bonus points will be gotten if this target accomplished. Failure to meet this target would result in that we may transport all the debris back to home base which will cause 3/4 of the bonus points.

Target 11 quantifies detection and location mechanism of the robot. This target is necessary to ensure that the robot is able to locate where it is and then calculate the fastest route to home base. The marginal value was chosen based on that only if the robot gets the information of where it is and the location it is heading to, the robot can find the fastest way to get there. To assess whether the target has been successfully achieved, some tests will be done. In the test, the robot will be placed in a random location on a simulated playing field, and there will be four similar walls around the playing field and four LED obstacles on an orbit line. Then the robot will run this module to test how fast and accurate the robot can get the information about the location. The percentage of successful locating will be recorded. The importance of this target is rated at 3 because this module is going to be the last part of the whole game, the failure of this module will not be very influential to frontal parts of the game. Failure to meet this target would

result in that we will lose the points for transporting the matched debris to the specific colored corner and returning home base, though returning home base counts not very much but there may be a lot of bonus points to lose if the robot can't transport the debris to the colored corners.

Target 12 quantifies the returning home mechanism of the robot. This target is necessary to ensure the motor inside is able to drive to robot back to home base. The marginal value was chosen based on if the robot was able to return to the home base reliably. If the robot can return home 80% of the time, then it would be viewed as a success. To assess whether the target has been successfully achieved, we will test the robot if it is able to move to the home base without collision with other robot when knowing the location and the destination of the robot. The importance of this target is rated at 3 because only if the robot returned to the home base, we can get the points of returning home base and the last part of transporting debris. Failure to meet this target would result in that even if we successfully found the route back to the home base and sorted the debris, it would still be useless.

1.4.5 Detection and Avoidance Module

This section of the Target Summary covers the goals for the detection and avoidance of UFOs and field architecture for the FAMU-FSU Southeast Con 2019 Robot. The need for this module is to find a way to detect and obstacles in the robot's path and avoid the detected obstacle by rerouting navigation path to a more suitable route. The targets will breakdown the needed percentage of detection and avoidance of UFOs and architectures within zone 2 of the playing field. This module is necessary to prevent point deduction or expulsion from the competition during each round of the competition.

Target 13 quantifies the detection and avoidance mechanism for the robot. This target is necessary to ensure that the robot is able to navigate the playing field freely, without colliding with other UFOs or field architects. The marginal value chosen for module 5 is based on the assumption that no other robot will intentionally attempting to block the pathway of this robot during the second round. As one of the stipulations for this competition, no robot should purposely attempt to collide with other robots in the playing field during each round, and so to ensure that this condition is met, the accepted marginal value is set at 95% to guarantee that the FAMU-FSU Southeast Con 2019 Team will not get points deducted or get disqualified from competition. To assess whether the target has been successfully achieved, various trails will be conducted on a practice field, where football size objects will be introduced at random times in random areas of the field to simulate playing condition at the competition. During each practice run, the amount of successful detection and avoidance of obstacles in the field will be record, then will be compared with the number of objects introduced to the field; and therefore, would give a reliable percentage of how well the robot can detect and avoid obstacles. The importance of this target is rated at 5 because failing to avoid any UFOs on the field could be viewed, by the judges, as a purposeful attempt to disable other robots in the field and therefore would result disqualification from the competition. Therefore, failure to meet this target would put the FAMU-FSU Southeast Con Team at a disadvantage, as this would greatly increase the possibility of having points deducted

Target 14 quantifies the detection and differentiation of the four colors (red, green, blue, yellow) associated with the debris scatter across the playing field. This target is crucial to ensure that the Southeast Con robot is able to accumulate as much points as possible during the

preliminary round, as each debris that is placed in the corner associated with its color will award 10 points; therefore, the marginal value chosen for this module is very high, since placing each debris collect to the area associated with its color will provide a substantial amount of point during each round. As stated in the competition rules, the separation of the debris based on color is not an essential requirement, however, color separation does provide a substantial amount of point, so therefore, and the importance of this module was set to 3. To ensure that the color differentiation algorithm is sufficient to score the Southeast Con robot sufficient points, this target will be assessed through the use of a numeric unit of measurement. During the assessment of this target, various practice runs will be conducted in which a record of the amount of debris detected and placed in the right colored corner will be collected. From the collected data, it will be determined whether the robot was able to detect the colors of the majority of the debris (8 or more debris). Although this target is not a crucial part of the overall robot design, it is important that the final robot design is able to meet this target, since the goal of the Southeast Con robot design is to score as many points as possible during each round of the competition.

1.5 Concept Generation

The Concept Generation is the process of developing solutions for design problems. This process is needed to produce massive amount of unique solutions to a problem, while using logic and research to back them. This gives the team options when selecting their approach to the problem at hand. In this Concept Generation, the solutions for the Southeast Con 2019 Hardware Competition were generated based on the functions found within its modules. The solutions are group together based on module and function. The solutions will breakdown

their benefits and downsides to their approach for that function. At the end of a group of solutions, a table will highlight essential data to differentiate between each solution.

1.5.1 Motion and Frame

In this section of the Concept Generation covers methods for motion during the competition at Southeast Con 2019. Also covers the solutions to battery and materials for the frame of the robot. These ideas highlight and take advantage of the rules provided by the competition. To be an effective a motion solution, it must rotate at high rpm, produce enough torque to drive the robot, while not consuming too much power to be an effective frame solution, the materials must be strong enough to support the components of the robot while not consuming too much of the budget. To be an effective battery solution, the battery must provide enough power to the system's electronics while lasting long enough to make it through the competition.

1.5.1.1 Motors

SparkFun 3.2V DC stepper motor

The SparkFun 3.2V DC stepper motor divides a full rotation into several steps, quantizing the amount of turns need to complete a full rotation. This motor is useful in making accurate turns and movements. This motor would be useful in the vehicle navigation and the debris picking up component of the device. One downside to the motor is its considerably low rpm of 2,000.

Vex robotics 12V DC CIM motor

The Vex robotics motor has an operating range of 5-12V and has a relatively low rpm of 5,330. This motor would be useful in the debris picking up component of the robot. This motor is

Team 305: Southeast Con 2019 Robotics Competition Team **29**

most notable for its relatively high stall torque of 2.41 N-m. One downside of this motor is its low rpm which would limit its use in high speed cases.

CE ROHS 7.2V CL-WS1512W gear brushless dc motor

A DC brushless motor is powered by DC current with an internal inverter that creates an AC current that powers a closed controller and controls the speed of the motor. DC brushless motors are most notable for having a high rpm of 33,000. These motors would be useful in controlling the speed and tires of the robot.

8VDC High Speed Motor

The 8V DC high speed motor has an operating voltage of 0-8 V DC and a relatively low current rating of 2 A. The motor is also notable for having a high rpm of 25,000. This motor would be useful in powering the movement and navigation of the robot. The motor also has a relative low cost of \$5.95

Table 8-Motor Comparison

Motor	Current Rating	Operating Voltage	RPM	Size
SparkFun Motor	2 A	3.2 V	2,000	2.22" x 2.22"
Vex Robotics CIM motor	2.7 A	12 V	5,330	2.5" x 2.5"
CE gear brushless motor	4.2 A	7.2 V	33,000	2.0" x 2.0"
8V DC High Speed Motor	2 A	8 V	25,000	1-3/8" x 15/16"

1.5.1.2 Frame Materials

Hard wood plywood

Hardwood plywood has a variety of uses but is most notable for its excellent strength and stiffness. Plywood is made by binding wood fiber sheets to form a composite. The material is also notable for having a relatively low density of 0.6 g/cm^3 . A relatively low weight and durable frame made out of plywood would be useful in maintain high speeds and stability.

6060 Aluminum

6060 aluminum is an alloy made of 98% aluminum and 2% various other metals. A notable trait of the alloy is its considerably high yield strength of 26,000 psi. One downside of using the aluminum frame would be its high conductivity which could lead to short circuits if any loose connections would make contact.

Polycarbonate

Polycarbonates are most notable for being durable, tough, and easy to cast into several shapes. They also have a relatively low density of 1.2 g/cm^3 when considering their relatively high yield strength of 10,152 psi. Polycarbonates also good electrical insulators which would eliminate the risk of short circuits with the frame.

Fiberglass

Fiberglass is a plastic that is reinforced using glass fibers and is notable for its relatively high yield strength of 30,000 psi. The material also has a notably low density of 2.55 g/cm^3 when compared to metals. A frame made of this material would be useful in maintaining a sturdy robot.

Table 9-Frame Material Comparison

Material	Density	Cost	Yield Strength
Hardwood Plywood	0.6 g/cm ³	\$13.27 per sheet	2,000 psi
6060 Aluminum	2.71 g/cm ³	\$2.50 per kilogram	26,000 psi
Polycarbonate	1.2 g/cm ³	\$10.50 per sheet	10,152 psi
Fiberglass	2.55 g/cm ³	\$2.50 per pound	30,000 psi

1.5.1.3 Battery

HitLights 12V DC / 5V DC (USB) Rechargeable Lithium-Ion Battery

The HitLights 12V DC battery has an operating voltage range of 5-12V DC and has around 3500 mAh. The battery is most notable for having a compact design and can be used to power motors, cameras, and other robotic components. The battery can also be recharged several times and has several output ports for supplying a 5V source as well.

Talentcell Rechargeable 6000mAh Li-Ion Battery Pack

The Talentcell battery has an operating voltage range of 12V and has around 6000mAh. The battery contains internal battery protection that improves its performance and life. The relatively small size of this battery would benefit when implementing it onto the overall design. One downfall of the device is its slow recharge rate of 10 hours.

Tenergy 12V 2000mAh NiMH Battery Pack w/ Bare Leads

The Tenergy battery has an operating voltage of 12 V and a low amp hour of 2000. One upside of this battery is its relatively small size which could be helpful in minimizing the overall

size of the robot and its weight. The low amp hours would make it less reliable when compared to other batteries when considering how often it would have to be charged.

Duracell Ultra 12V 8AH AGM SLA Battery

The Duracell Ultra battery was first intended to be used as backup power and has 50 to 150 cycles at 100% discharge. The battery has a relatively high amp hours of 8000 and has an operating voltage of about 12 V. The downside of the battery is its considerably large size. A battery like this would be useful in powering several components simultaneous without worry of it running out of power.

Table 10-Battery Comparison

Battery	Wattage	Operating Voltage	Amp hours	Size
HitLights	24 W	12 V	3500	3.5 x 2.4 x 1.1 inches
Talentcell	20 W	12 V	6000	1.1 x 3.35 x 5.7 inches
Tenergy	15 W	12 V	2000	72 x 50 x 299mm
Duracell Ultra	20 W	12 V	8000	3.78 x 0.98 x 2.4 inches

1.5.2 Route Clearing Algorithm

One of the main goals of the Southeast Con competition is to remove debris from a specified area of the playing field. To do this competitively an autonomous robot must be able to quickly identify debris, travel to it, and remove it from the specified area. The solutions investigated in this section are using a predetermined path to blindly remove debris, use sensors

to travel from debris to debris, use sensors to find debris along a predetermined route, and survey the field to plan an optimal route for collection.

1.5.2.1 Gathering Solutions

One of the main methods of gathering points in the Southeast Con competition is to remove debris from a specified area of the playing field. To be competitive, the robot must autonomously identify, travel to, and remove the debris quickly.

Predetermined Route

The first method investigated is for the robot to follow a path blindly and collect any debris it encounters. The strength of this method is it is simple, effective, and easy to implement. A major drawback to this design is that it is time consuming and requires more revolutions around the center structure than the other methods investigated.

Travel from Debris to Debris (No Predetermined Path)

Another method investigated is to ignore any type of predetermined path and simply have a sensor that searches for debris while the robot moves in a random direction. Once debris is identified the robot will travel to it, remove it, and resume searching for the next debris. This method relies on solely on the use of a sensor for navigating the playing field making collisions significantly more likely to occur. Like the first method the strength of this method is its simplicity, and ease of implementation.

Predetermined Route with Deviations for Debris

The next method investigated is a hybrid of the two previous methods. This method involves the robot traveling along a predetermined route while using a sensor to locate debris near it. Once a debris is spotted the robot removes it, and then resumes the predetermined route.

This method reduces some of the weaknesses exhibited by the other method while only moderately increasing the complexity of the robot.

Survey the Field and Plan an Optimal Route

This method involves the robot traveling to several predetermined locations and surveying the field. Form this survey the robot calculates an optimal route to gather all the objects identified as quickly as possible. This method is the most efficient of the algorithms and has the fewest number of revolutions around the center structure. However, it is significantly more complex and will take some time to survey the field.

Table 11-Different Algorithms Comparison

Method	Time Estimation	Complexity	Number of Revolutions	Ease of Implementation	Possibility of Collisions
Predetermined Route	4-6 minutes	Simple	Many	Easy	Low
Debris to Debris	4-6 minutes	Simple	Moderate	Easy	High
Predetermined Route with Debris Searching	2-3 minutes	Moderate	Moderate	Easy-Moderate	Low-moderate
Survey and Route Planning	2-3 minutes	Complex	Few	Difficult	Low

1.5.2.2 Microcontrollers

The brain of this project is the microcontroller. The selection of which microcontroller to use will determine a large portion of what the robot can accomplish. Many microcontrollers reviewed for this project behave more like small computers rather than just a microprocessor. Some of the advantages of using this style of microcontroller is the large number of integrated

features associated with them. Some of the features include memory, microprocessor, clock, variety of I/O ports, PWMs, analog to digital converters, and built in signal conditioning.

For this project a microcontroller will be selected primarily on processing performance, operating temperature, voltage requirements, and memory available. These attributes were selected to ensure the robot has the required computing power and physical properties to operate sufficiently for the 3-minute round of competition.

Concept 1: Pixy2 CMUcam5 Sensor

The Pixy2 camera allows for the observation of both flat and 3d objects, as this Arduino compatible camera is able to provide a 3-dimension feedback of what is being observed. Furthermore, this video camera allows for categorizing each object detected in relation to the color and size of the object at a faster time frame, due to its 60-fps frame rate. Because this camera has a companion web application, it is possible to program camera send a specific feedback to the Arduino microcontroller, and therefore increase the potential utility of this device. With a 60-degree horizontal and 40 degree vertical, the Pixy2 cam allows for wider viewing angle, which would allow for better detection of objects in the field.

Concept 2: TTL Serial JPEG Camera with NTSC Video

The TTL JPEG camera allows for the capturing of objects in jpeg format or videos at a frame rate of 30 fps, allowing both fast image process. Similar to the Pixy2 camera, this JPEG camera voltage requirement ranges around 5V, which is ideal for the use with any microcontroller. A notable feature for this device is its 60-degree vertical and horizontal viewing angle, which is wider when compared to the Pixy2 camera; however, due to the frame rate of 30

fps there may be some slight time increase in analyzing the object detected and categorizing it as a UFO or debris.

Concept 3: IR Distance Sensor (20cm-150cm Range)

This 20 – 150 mm range IR distance sensor provides enough detection range to allow for a reasonable response times for the robot. With a 150 mm max detection distance, this IR sensor allows for good range detecting an object and avoiding the detected object. One drawback with using this sensor is that it is only capable of objects straight ahead of it, so this will require the robot to rotate to scan the surrounding area.

Concept 4: IR Distance Sensor (100cm-500cm Range)

The 100 – 500 cm range IR distance sensor provides a very reliable detection distance, allowing for quicker time to examining objected detected and categorizing the item as debris or a UFO that needs to be avoid. This sensor has the same voltage requirement is the sensor previously examined, however, it is a bit bigger, with more than twice the max detection range. As with the previous IR sensor, this distance sensor can only detect an object at a linear path or in other words it can only detect something that is straight ahead of it.

Table 12-Sensor Comparison

Sensors	Voltage Requirement (Volts)	Output Format	Size	Frame Speed	Viewing Angle
Pixy2 CMUcam5	5 V	Video	38mm x 41.91mm x 15.24mm	60 fps	60 Degree (Horizontal), 40 Degree (Vertical)
TTL Serial JPEG Camera with NTSC Video	+5 V	JPEG	32mm x 32mm x 32mm	30 fps	60 Degrees
IR Distance Sensor (20cm- 150cm Range)	5 V	analog voltage	21.66mm x 44.39mm x 18.67mm	N/A	0 Degrees (Linear)
IR Distance Sensor (100cm-500cm Range)	5 V	analog voltage	58mm x 17.6mm x 22.5mm	N/A	0 Degrees (Linear)

1.6 Concept Selection

1.6.1 Movement Hardware and Software Selection

For this module, the focus is to ensure the robot can move quickly while still maintaining stability and control. The frame chosen for this design will have to be lightweight as well as durable and sturdy. 6060 Aluminum was chosen for the frame material due to its relatively inexpensive cost of \$2.50 per kilogram and high yield strength of 26,000 psi. The frame will be taking up most of the weight and ensuring it has a lightweight frame is vital in ensuring its speed. For the battery, the Talentcell battery was chosen due to its 6000-amp hour rating and at 20W with 12V operating voltage. The battery was chosen by considering which ones would need the least amount of recharging after use. Also, the overall size of the battery is beneficial in

conserving space. The motor chosen for the design was the CE gear brushless motor. It has a max rpm of 33,000 and an operating voltage of 7.2 V at 4.2 A. The motor would be useful in ensuring the robot is able to reach top speeds quickly.

1.6.1.1 Material Solution Selection

For the material selection, the 6060 Aluminum was chosen on the basis of light density, yield strength, and overall cost. The aluminum has an impressive yield strength of 26,000 psi. The one downside to the aluminum its heavy density of 2.71 g/cm³. The Aluminum has a low cost of \$2.50 per kilogram. Compared to the fiberglass, the aluminum has a more reasonable price range but lacks the light density that is maintained in it. When compared to the hardwood plywood, the wood has a significantly lower yield strength and slightly pricier cost. However in the end, the team ended up using PLA as the selected material because of its low cost, rapid prototyping with 3D printing, and rigid structure for the force the robot was expected to take.

Table 13-Desired Material Features

	Weight	Baseline	Optimal	Description
Light Density	4	1.5 g/cm ³	1.0 g/m ³	Desired density for lightweight material
Yield Strength	3	15,000 psi	25,000 psi	Overall strength of material
Cost	5	\$10.00	\$7.50 per sheet	Cost per sheet of material

Table 14-Material Considered Pugh Chart

Selection of Criteria	Hardwood Plywood	6060 Aluminum	Fiberglass
Light Density	+	-	+
Yield Strength	-	+	+
Cost	-	+	-
Score	-2	4	2

1.6.1.2 Battery Solution Selection

For the battery, it was decided that the Talentcell provided the most optimal specifications when considering its compact design, vast amount of amp hours, and wattage. The Talentcell battery had 6000-amp hours, a compact 1.1 x 3.35 x 5.7 in dimension, and 20 watts. When compared to the HitLights battery, the Talentcell had optimal amp hours but not as much wattage. When compared to the Duracell, the Talentcell had a significantly smaller and more compact design making it useful in the final implementation of the design. Although, the Duracell was able to supply an impressive 8000-amp hours per battery life.

Table 15-Battery Requirements

Selection of Criteria	Weight	Baseline	Optimal	Description
Wattage	4	20	30	Overall power supply of the battery
Size	5	2.5 x 2.5 x 2.5	1.0 x 1.0 x 1.0	The dimensions of the battery
Amp Hours	4	4000	6000	Duration of battery life

Table 16-Battery Feature Pugh Chart

Selection of Criteria	HitLights	Talencell	Duracell Ultra
Wattage	+	0	+
Size	-	+	-
Amp Hours	-	+	+
Score	-5	9	3

1.6.1.3 Motor Solution Selection

When selecting the motor for the final design, it was decided the CE gear brushless motor would be the best based on its optimal operating voltage of 7.2V, high RPM of 33,000, and compact size of 2.0" x 2.0". When compared to the spark fun motor, the CE motor had a significantly higher RPM which would be helpful in attaining high speeds with the robot in a short amount of time and, although the operating voltage of the SparkFun motor is better. When compared to the 8V DC high speed motor, the size of the 8V dc motor was slightly less than the CE motor but its max RPM was slightly less than the CE motor.

Table 17-Motor Requirements

Selection of Criteria	Weight	Baseline	Optimal	Description
RPM	5	10,000	30,000	Revolutions per minute of motor
Operating Voltage	4	8 V	7.0 V	Max operating of the motor
Size	3	2.00" x 2.00" X 2.00"	1.5" x 1.5" x 1.5"	Overall Size of motor

Table 18-Motor Feature Pugh Chart

Selection of Criteria	SparkFun Motor	CE gear Brushless Motor	8V DC High Speed Motor
RPM	-	+	+
Operating Voltage	+	+	0
Size	-	0	+
Score	-4	9	8

1.6.2 Route Clearing Algorithm Solution Selection

This selection process will compare the previously discussed methods of route planning and how to best implement them with respect to the qualities desired of the robot. This process will involve defining the qualities desired for the robot to perform competitively at the Southeast Con competition. Then these qualities will be compared to each method and a Pugh Matrix will be used to select the optimal route clearing module.

By using the Pugh Matrix, the concept of predetermined route with debris searching was selected due to its strengths in searching the playing field and avoiding known objects. However, this method only has moderate resistance to environmental noise and is slow at implementing the UFO avoidance code.

1.6.2.1 Route Clearing Algorithm Solution Selection

For the Southeast Con competition an automated robot will be tasked with moving randomly placed debris from a center zone to an outer zone. To do this in under the allotted round time of 3 minutes a robot must be able to efficiently search the playing field for debris. The function of this route clearing algorithm is to find an effective way to quickly search the field and remove

debris. The route clearing algorithm will provide the robot with a method of quickly searching the playing field, identifying debris, and avoiding known obstacles. The solution to this selection will possess the following qualities:

- Quickly search the playing field
- Avoid all known objects
- Resistant to environmental noise
- Allow for adjustments to avoid UFOs

To decide which design best fits the requirements for the Southeast Con robot a Pugh Matrix was used. The table below (table 19) breaks down the qualities listed above for use in the Pugh Matrix. The weight of each quality was selected on a scale of 1 to 5, with 5 being a critical quality to the overall design of the robot. Each baseline was selected based on the nominal operation of each method. While the optimal is based on the targets previously mentioned.

Table 19-Route Clearing Requirements

Selection Criteria	Weight	Baseline	Optimal	Description
Searching the Playing Field	5	2.5 min	1.5 min	Time to completely search the playing field for debris
Known Object Avoidance	4	3 in	6 in	Smallest distance robot comes to collision with a known obstacle
Resistance to Environmental Noise	2	3 times	0 times	Number of times the robot malfunctions due to environmental noise
Time to Implement UFO Avoidance Code	3	2 sec	< 1 sec	Time for the robot to recognize a UFO and implement UFO avoidance code

Table 20 below is the Pugh Matrix to assist with the decision of which route clearing algorithm best fits the Southeast Con robot. The Pugh Matrix is a comparison of each route

clearing concept and the qualities desired in the robot to the baseline in table 19. The “+” represent an increased performance, “-” represent a decreased performance, and “0” represents no significant difference. The score for each method is the sum of the weight of each quality multiplied by the comparison symbol. For example, the predetermined route is $(4+2) + (-5-3)$ which results in a score of -2. The highest scoring concept was a predetermined route with debris searching. This concept allowed for quick searching of the playing field, avoidance of known objects, moderate resistance to environmental noise, and a moderate response time to avoid UFOs.

Table 20-Route Clearing Pugh Chart

Selection of Criteria	Predetermined Route	Debris to Debris	Predetermined Route with Debris Searching	Survey and Route Planning
Searching the Playing Field	-	0	+	+
Known Object Avoidance	+	-	+	+
Resistance to Environmental Noise	+	-	0	-
Time to Implement UFO Avoidance Code	-	+	0	-
Score	-2	-3	9	3

1.6.3 Microcontroller Selection

This selection process will compare the previously discussed microcontrollers with the qualities desired to create a competitive robot for Southeast Con. This process will involve defining the qualities desired for the robot to perform competitively. Then these qualities will be compared to each microcontroller in a Pugh Matrix and an optimal microcontroller will be selected.

By using the Pugh Matrix, the microcontrollers Raspberry Pi B+ and BeagleBone Blue were selected. These microcontrollers were both selected because they scored the same in the Pugh Matrix and only have subtle differences between them. The BeagleBone Blue will most likely be used in this robot due to being specifically designed to operate robots.

1.6.3.1 Microcontroller Solution Selection

For the Southeast Con robot to be competitive it needs to be able to make complex decisions. The most efficient and simplest way to accomplish this is using a microcontroller. The microcontroller for this robot will need to have enough processing performance, memory storage, number of pins, and be small enough to fit on the robot. Table 21 below breaks down the selection criteria needed to compare each microcontroller in a Pugh Matrix. The table quantifies each selection criteria by weight, baseline value, and optimal value. The weight is a number between 1 and 5 describing the importance of that criteria to the robot's overall performance. Baseline represents the nominal value desired for that criteria, while optimal describes the desired value for competitive performance.

Table 21-Microcontroller Requirements

Selection Criteria	Weight	Baseline	Optimal	Description
Processing Performance	4	180 MHz	1 GHz	The speed of the microcontrollers internal clock
Memory Storage	3	256 KB	512 MB	The memory storage capacity of the microcontroller
Size	2	85 mm	56 mm	The physical size of the microcontroller
Number of Pins	4	40	50	The number of available I/O pins on the microcontroller

Table 22 below is the Pugh Matrix to assist with the decision of which microcontroller best fits the Southeast Con robot. The Pugh Matrix is a comparison of each microcontroller and the qualities desired in the robot to the baseline in table 22. The “+” represent an improved performance, “-” represent a decreased performance, and “0” represents no significant difference. The score for each method is the sum of the weight of each quality multiplied by the comparison symbol. For example, the Teensy 3.6 is $(0+2+4) + (-3)$ which results in a score of 3. The highest scoring concept was tie between the Raspberry Pi B+ and BeagleBone Blue. These two microcontrollers behave similarly in both processing performance, memory storage, size, and number of pins. A closer inspection of these two microcontrollers show that the Raspberry Pi B+ has a slightly better processor and significantly better storage. However, the BeagleBone Blue has more pins, is slightly smaller, and is specifically designed to be used in robotics.

Table 22-Microcontroller Feature Pugh Chart

Selection of Criteria	Raspberry Pi B+	Arduino Mega 2560	Teensy 3.6	BeagleBone Blue
Processing Performance	+	-	0	+
Memory Storage	+	0	-	+
Size	0	-	+	0
Number of Pins	0	+	+	0
Score	7	-2	3	7

1.6.4 Sorting Hardware and Software Selection

In this section the solutions from the concept generation for the sorting hardware and software solution was analyzed. These solutions included gathering, sorting and storage methods. Making the proper selection for this module increases the maximum point for the threshold. However, as each function's solution was selected, power consumption was kept in mind, so this module does not pull too much power from the battery. These solutions were picked based on the most engineering benefit through a Pugh chart and other information that could not have a numerical value.

1.6.4.1 Gathering Solution Selection

During the concept generation phase, several gathering solutions were considered. These solutions were the not gathering space debris, extendable arm system, and the dual brush system. To decide what solution was the best fit for the robot, a Pugh chart was used to assist with

decision. Table 23 breaks down the criteria used in the Pugh chart. Maximum possible points were given the highest weight of 5 because this limits the overall points the robot can gathered. 210 points was chosen as the baseline because of target 7, space debris gathered, marginal value was seven space debris gathered. Then assumed that the robot was able to get the space debris out of zone 2 to zone 1, sort all space debris gathered, and then turn them into the proper color coded home. Which is 30 points per space debris, or 210 points for all 7 pieces of space debris. For the optimal value was based on the all available pieces of space debris was properly taken care of, or 360 points. Power consumption was baseline was based on two average stepper motors, because we did not want to draw too much power for the sorting hardware and software module. Gathering time baseline was chosen on if an opponent robot were to take the shortest orbit possible, and was traveling at maximum speed of 4.5 ft/s. which would cause the opponent robot to complete one in about 1.5 seconds. Then this would lead the robot needs to spend five seconds rotating about the shortest orbit to gather same amount of for maximum threshold for points of one piece of space debris. Two seconds was chosen as the optimal value because then the robot gather two pieces of space debris for every five seconds another robot spend rotating, effectively increasing overall possible point threshold for one round.

Table 23-Gathering Criteria

	Weight	Baseline	Optimal	Description
Max Possible Points	5	210	360	Enough points to be competitive
Power Consumption	2	100W	60W	Amount of power the solution consumes
Gathering Time	4	5s	2s	Time to gather 1 piece of Space Debris

In table 24 is Pugh chart for the gathering solution selection. The information gathered in the concept generation was applied to the criteria, then scored to its respective weight. Dual Brush System ended up having the highest score, because it exceeds all baselines in this selection. It should be noted that even though the “Don’t Gather Space Debris” solution had four times the maximum points to gathered, the power consumption was high compared to the other solutions, because the need of either more motors or higher quality motors which lead to more power consumption. The final decision is a combination of “Don’t Gather Space Debris” and “Dual Brush System” methods. The idea is while there is space debris is present, the robot will try to focus on gathering space debris. However, when the field is void of space debris, the robot will focus on orbits.

Table 24-Gathering Solution Pugh Chart

Selection of Criteria	Don’t Gather Space Debris	Extendable Arm System	Dual Brush System
Max Possible Points	+	+	+
Power Consumption	-	0	+
Gathering time	+	-	+
Score	7	1	11

1.6.4.2 Sorting Solution Selection

During the concept generation phase, several sorting solutions were considered. These solutions were the Elevator meets Rubik’s cube approach, Lane Driver, and the Linear Memory

methods. To decide what solution was the best fit for the robot, a Pugh chart was used to assist with decision. Table 25 breaks down the criteria used in the Pugh chart. The sorting time baseline and optimal value was the same as gathering time because as the piece is moving to another space debris or gathering, the robot should have sorted and deposited the recent piece of space debris. The baseline for the working volume was selected to be two pieces of space debris, because that would take up 15% of space on the first level of the robot. This area is valuable for the fact that is where the important components will live. Any more space taken up for space prior to sorting is not possible. Little to no space is the target for this criteria. Two motors were selected to be the baseline because to avoid too much power consumption by this module. However at least one motor will be necessary to move the space debris from the gathering module to the sorting module.

Table 25-Sorting Criteria

	Weight	Baseline	Optimal	Description
Sorting Time	5	5s	2s	The time to take to sort the space debris
Working Volume	3	2 pieces of space debris	1 pieces of space debris	The space available prior sorted needed
Motors	2	2	1	Amount of motors needed to sort

In Table 26 is the Pugh chart for the sorting selection. The information gathered in the concept generation was applied to the criteria, then scored to its respective weight. The Linear Memory solution gather the most amount of points. This method would superior in sorting time due to how simple the process is. Also, since the sorting time is so short, the working can be minimal compared to the other ideas. Lastly one motor is need since the sorting method will

simply check the color, and drive the space debris straight to the storage chamber. However, the team has decided to go with “Elevator meets Rubik’s cube approach” because the need of a linear storage capacity for the Linear Memory method. The linear storage refers to storage chamber that must keep the space in the same order that it entered so that the robot does not lose track of where it is located. Having the space debris jumbled but color coded allows during the period in which the robot has to deposit the space debris in the proper color coded corner to take up less time. In the linear storage it must rush side to side to place the space debris in the right corner, while the other method only has to do one loop to drop off the space debris.

Table 26-Sorting Criteria Pugh Chart

Selection of Criteria	Elevator meets Rubik’s cube approach	Lane Driver	Linear Memory
Sorting Time	-	-	+
Working Volume	+	0	+
Motors	0	-	+
Score	-2	-7	10

1.6.5 Storage Selection

During the concept generation phase, several storage solutions were considered. These solutions were the box storage system, simple sack storage and horizontal lane storage system methods. To decide what solution was the best fit for the robot, a Pugh chart was used to assist with decision. Table 27 breaks down the criteria used in the Pugh chart. Volume baseline was decided base on the assumption that the team was able to gather all pieces of space debris. Also, the idea that all the space debris were all spheres was assumed because to avoid one getting stuck

in the specified chamber. Also, the storage solution must keep all the space debris color coded, it would make the sorting solution pointless and decrease the maximum point threshold.

Table 27-Sorting Solution Requirements

	Weight	Baseline	Optimal	Description
Volume (LxWxH)	3	5” x 5” x 6.5”	5” x 5” x 5.25”	Internal volume that the storage unit should hold
Keeps the Space Debris Sorted	5	Yes	---	Keeps the space debris sorted

In Table 28 is the Pugh chart for the sorting selection. The information gathered in the concept generation was applied to the criteria, then scored to its respective weight. The boxed storage system solution gathered the most amount of points and will be the team’s choice for storage.

Table 28-Sorting Solution Pugh Chart

Selection of Criteria	Boxed Storage System	Simple Sack Storage	Horizontal Lane Storage System
Volume (LxWxH)	+	+	-
Keeps the Space Debris Sorted	0	-	0
Score	3	-2	-3

1.6.6 Return home algorithm

This is the selection process which will compare the methods of home returning that mentioned before. In this section, several qualities which are required for robot of Southeast Con competition to perform well and score more points will be defined. These all qualities will be discussed and assessed for all four methods of returning home. To find out which method is the best one for our robot to compete and win the competition, a rating mechanism will be established. The method which scores most is the best method for returning home.

By using this mechanism, the method of locate and go is selected as the best method. It is because with this method applied, the robot can be precisely and quickly going back to home and the sensor using for the method is common. However, it could be a little bit difficult to apply this method as it is a combination of two methods.

1.6.6.1 Localization solution

In the concept generation phase, we discussed some solutions for localization. These solutions were Distance from walls measuring, color of spacetel detecting, localize by gyro and code disk and locating and go. To decide which solution to be used for our robot, a Pugh matrix is used to help us. Table 29 brake down the criteria used in the Pugh matrix. Each criterion has its weight which represent the importance of the criteria. Estimated time is given a weight of 2 because it is a limited time competition, but after estimated the total time for all modules, we concluded that we may have enough time to do all modules, time is not the thing we worry mostly. Thus, the baseline is decided as 35. Accuracy is the most important quality for this module, thus the weight of accuracy is the highest 5. It is because that if the robot cannot

precisely go to the colored corner, we will lose a big amount of points. For possibility of error, it is also given a high weight which is 4 since the error could lead to lose points. The ease of implementation is given the weight of 2 as it could be improved if we take efforts on it. The size of the sensor is given a weight of 2, the baseline is chosen as 3’*3’*3’ since the maximum size of the robot is 9’*9’*11’, we decide to try to limit the size of each sensor under 1/3 of the size of the robot. The baseline for accuracy and possibility of error is chosen as moderate because if big mistake occurs, we might lose too many points to lose the competition.

Table 29-Localization Criteria

	Weight	Baseline	Optimal	Description
Time estimate	2	35	15	Time for returning home
Accuracy	5	moderate	high	The accuracy of the solution
Ease of implementation	2	moderate	easy	Easy or not easy to implement
Possibility of error	4	moderate	low	The possibility of error of the solution
Size of sensor required	2	3’x3’x3’	none	The size of the sensor of the solution

In table 30 is the Pugh chart for the localization solution selection. The information gathered in the concept generation was applied to the criteria, then scored to its weight respectively. Locate and go solution ended up having the highest score since it exceeds most of the baselines in this selection. For the solution with a gyro and code disk, it should be noted that it is even better than locate and go solution in some respects like accuracy and estimated time,

however, this solution requires a big assembly of sensors and it is rather difficult to implement.

The final decision is locating and go solution. This idea absorbed the advantages of two methods which are distance from wall solution and color of spacetel solution. So, this is a both accuracy and feasible solution.

Table 30-Localization Criteria Pugh Chart

Selection of Criteria	Distance from walls	Color of spacetel	With a gyro and code disk	Locate and go
Time estimate	+	+	+	+
Accuracy	0	-	+	+
Ease of implementation	0	+	-	-
Possibility of error	0	-	+	+
Size of sensor required	+	+	-	+
Score	4	-3	9	13

1.6.7 Wheels Selection

This selection process will compare the previously discussed wheels with the qualities desired to create a swift robot for Southeast Con. This process will involve defining the qualities desired for selecting wheels. Then these qualities will be compared to each wheel in a Pugh Matrix and a final decision will be made.

By using the Pugh chart, the wheels with medium pressure were selected. This kind of wheels was selected because it scored highest in the Pugh chart. It is not only providing enough friction to the robot but also easy enough to access, further it is also enough to carry the robot.

To win the competition of southeast con which is time-limited, it is important to find a kind of wheels that can provide a big friction value which is good to make the robot move quicker. So, the friction value is chosen as the most important criteria which is given weight of 5. The baseline for the criteria is selected as 0.4 so that our robot can move by the center column while not drifting. The price is given the weight of 3 and the baseline for it is 20 dollars since normal wheels usually are not expensive. The baseline of load is defined as 5 pounds as the weight of the robot we designed so far. The optimal value is chosen in case of that we may add more parts to the robot.

Table 31-Wheel Criteria

Selection Criteria	Weight	Baseline	Optimal	Description
price	3	20	10	The total price of the wheels
Friction value	5	0.4	0.6	The friction value between wheels and field
load	2	5 pounds	10 pounds	The highest load of the wheels
Ease of implementation	2	Moderate	Easy	Easy or not easy to implement

Table 32 below is the Pugh Matrix to assist with the decision of which kind of wheels best fits the Southeast Con robot. The Pugh Matrix is a comparison of each wheels and the qualities desired in the robot to the baseline in table 31. The “+” represent an improved

performance, “-” represent a decreased performance, and “0” represents no significant difference. The score for each method is the sum of the weight of each quality multiplied by the comparison symbol. For example, the Meconium wheels is $(5+2) + (-3+2)$ which results in a score of 2. The highest scoring concept was the medium pressure wheels. Though this kind of wheels cannot provide a precise and swift performance as meconium wheels for the robot moving, this kind of wheel is cheap and easy to access and program. The load that the wheels can carry is enough for our robot and it has a high friction value which is conducive for high speed which can make our robot competitive.

Table 32-Wheel Criteria Pugh Chart

Selection of Criteria	Airless wheel	High pressure wheels	Medium Pressure wheels	Meconium wheels
price	+	+	+	-
Friction value	-	-	+	+
load	+	+	+	+
Ease of implementation	+	+	+	-
Score	2	2	12	2

1.6.8 UFO Avoidance Algorithm Selection

In this section the solutions from the concept generation for the avoidance algorithm was analyzed. These solutions included avoidance method and sensor solution. Making the proper selection for this module helps avoid the loss of points. However, as each function’s solution was selected, power consumption was kept in mind, so this module does not pull too much power

from the battery. These solutions were picked based on the most engineering benefit through a Pugh chart and other information that could not have a numerical value.

1.6.8.1 Avoidance Solution Selection

During the concept generation phase, several avoidance solutions were considered. These solutions were the predetermined object size, avoidance percentage and ease of implementation. To decide what solution was the best fit for the robot, a Pugh chart was used to assist with decision. Table 33 breaks down the criteria used in the Pugh chart. Time estimation baseline was chosen to be three minutes due to the length of one round. The optimal value for time estimation was selected, so the robot would have completed the setup for avoidance prior the minute used to deposit space debris and return home. The avoidance was selected to be 90% because a large amount of points can be loosed by running into structures and run risk of being disqualified from the tournament. 99% was selected because it is truly impossible to have a design that can avoid another structure 100% of the time. Ease of implementation was selected to be moderate because this module will be the last to be installed into the robot, so it should be able to be applied easily.

Table 33-Avoidance Solution Criteria

	Weight	Baseline	Optimal	Description
Time Estimation	3	3 mins	2 mins	Internal volume that the storage unit should hold
Avoidance Percentage	5	90%	99%	Odds that the robot will avoid an oncoming structure
Ease of Implementation	2	Moderate	Easy	Installing and modifying the solution to the robot

In Table 34 is the Pugh chart for the avoidance solution selection. The information gathered in the concept generation was applied to the criteria, then scored to its respective

weight. The stop and scan solution gathered the most amount of points, however It will not be the team’s pick. The team will go with a mixture of both other choices because the stop and scan would cause the team too much time while the opponent is gathering points or making orbits. The combination of predetermined object size and surveying field while moving, allows the robot to keep moving to pick up space debris, while prepare for an oncoming piece of space debris.

Table 34-Avoidance Solution Pugh Chart

Selection of Criteria	Predetermined Object Size	Stop and Scan Surround	Surveying Field While Moving
Time Estimation	0	+	-
Avoidance Percentage	0	+	-
Ease of Implementation	0	+	0
Score	0	10	-7

1.7 Southeast Con 2019 Spring Plan

Please Refer to Appendix G for the breakdown of each part.

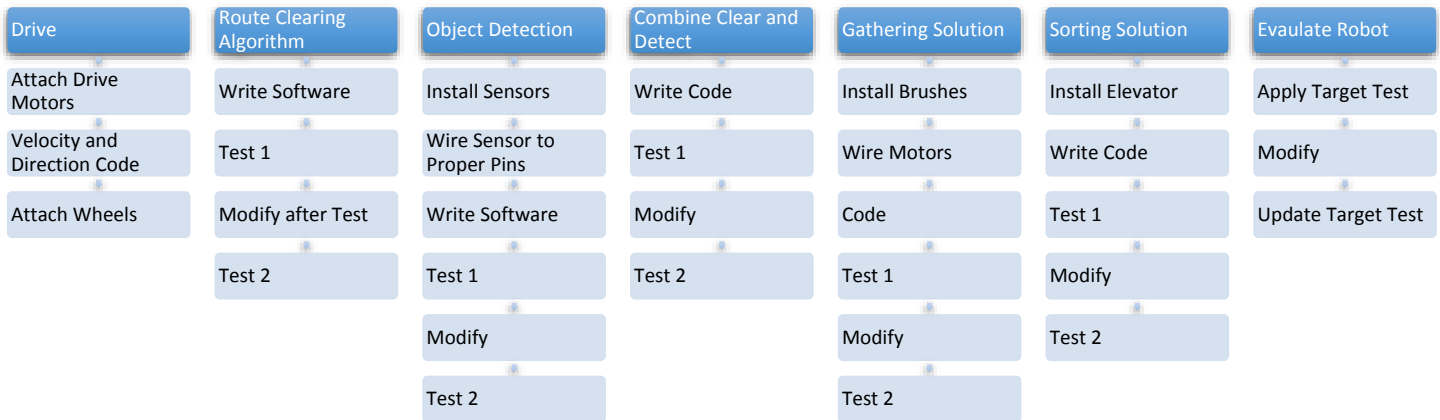


Figure 1-Work Breakdown

Chapter Two: EEL 4915C

2.1 Targets for the Southeast Con 2019 Team 305

2.1.1 Project Plan

As Southeast 2019 Team entered their second semester, months prior to the competition, more emphasis was put onto the building the robot, rather than concept generation and selection. After the end of last semester, the students had to find a more reliable way to develop the frame of the robot compared to relying on the machine shop. The machine shop was a profession team in the FAMU-FSU College of Engineering building that was able to sculpt materials for students and professors at no charge. However, with multiple teams approaching for product development, as seen at the end of fall 2018 semester, simple aluminum designs took over two weeks to be produced. This meant that if any designs were modified or errors while developing the product, would put the team two to three weeks behind. That is why the team turned to 3D printing.

On the FSU main campus, by Landis Green is the Innovation Hub. At the Innovation Hub, students are able 3D print, laser cut, and practice with virtual reality at no expense. The team shifted their major part of fabrication to 3D printing through the Innovation Hub. The Innovation Hub support over 15 small, 2 medium and 1 large scale 3D printers. When 3D printing, the design can be with millimeters of accuracy, rather than at Machine Shop was within centimeters. Also, when 3D printing the structure was printed as a whole, compared to being welded together, which increase the overall strength of the structure. Lastly, the time to do a complete print of the whole robots frame, bumper, and other small parts, could be done with two days. This shift to 3D printing allowed the team to do multiple iterations of the robot, to tune the design received the target results.

3D printing drastically change the original project plan crafted at the end of the fall 2019 semester of the robot. The new project plan required the team to change the frequency of meeting from two to three times a week. In the prior semester, the team only met during the weekend and/or whenever an extra meeting was needed. Another day was added, Wednesday, to allow the team discuss the current iteration, breakdown expectation that needed to be met before the weekend, and be able to meet with reviewers or advisors if necessary. Next, the team decide to break-up into minor sections, based on module. Chase and Kyle, designed and tested the frame, motors, and electronic components. This was the first module that had to be perfected so the software team would be able to reliably run their code in the real world. Chase and Kyle made the decision to 3D print all small components (gears, hubs, and internal elevator), because they were able to hone in press fit the wheel hubs to the motors and get the proper gear ratio for the proper parts. They used Creo and Cura to develop the 3D design. Fabio, and Daniel were the

major software development. Their job was to implement the each module through the selected hardware. As each iteration came available, they would implement and test respective module. The first step would have develop a code for the robot to be able to drive around the center structure in a circle. This would prepare the team for the predetermined route. Second step, was to develop code to exit the home corner. This step must be repeatable for each corner. Third step, use the on board Pixy2 to go off the predetermined route and gather the debris. As the robot gather debris through the front brushes, it would plan how to get back on route. It was important that the robot return to the predetermine route before gathering another debris, to avoid confusing the robot. Fourth step was to implement the sorting method into the robot. Chase would step away from the fabrication and iteration process to assist with developing the code for the sorting box. The fifth step would implement return home. The robot would make one more complete orbit in the last thirty seconds of the round. While doing so the robot would find where it is, and find the proper launch spot. This was found by going to the color line in zone 2 that came before the home corner color. To do this, the robot read the color lines in zone 2 till it reach the target color, stop, and then prepare for launch. It would launch at the specified angle, drive home, stop before hitting the wall, and raise the on board flag. The last step, was to develop code to detect and avoid other robots. This step was important, so if the robot passed the qualifying rounds, it would not hit other robots. As the team finish each section they would test and update the overall system to meet the targets.

2.1.2 Build Plan

Table 35- Build Plan breakdown

Iteration	Hardware Implementation	Software Implementation	Time Allotted
Version 0	<ul style="list-style-type: none"> • Base • Drive Motor • Microcontroller [Use old team robot] 	<ul style="list-style-type: none"> • Have the robot drive in a straight line 	August 27, 2018 To December 15, 2018
Version 1	<ul style="list-style-type: none"> • Base [No elevator walls] • Bumper • Bumper Supports • Pixy2 & 2 IR Sensors 	<ul style="list-style-type: none"> • Exit Home • Follow Pre-Determine Route 	January 7, 2019 To January 31, 2019
Version 2	<ul style="list-style-type: none"> • Internal Elevator Walls • Wheel Hubs • Brush Motors • Brushes 	<ul style="list-style-type: none"> • Go off Pre-Determine Route to Debris • Gather Debris 	February 1, 2019 To February 28, 2019
Version 3	<ul style="list-style-type: none"> • Internal Elevator • Elevator Motor • Second Base Plate • Sorting Box Motor • Sorting Box 	<ul style="list-style-type: none"> • Sorting Code • Return Home 	March 1, 2019 To March 23, 2019
Version 4	<ul style="list-style-type: none"> • Break Base into Parts • Flag 	<ul style="list-style-type: none"> • Raise Flag • Detect and Avoid other Robots 	March 24, 2019 To April 11, 2019

Version 0

This was the starting version of the robot. At this time the students were focused on concept generation and selection. However, to get an understanding of how the robot would interact with real world, the team broke down a past team's design and build basic motion robot. This robot had two drive motors powered by a 12V battery and an Arduino Mega for the drive

logic. Using the Arduino IDE, the team would develop basic drive code, to make the robot drive in a straight line.

Version 1

At this point the students would have made all major selections for the final design of the robot and would prepare to implement at the beginning of the Spring 2019 Semester. In this version, the team would develop a base of the robot and the predetermined route. The hardware include two drive motors to propel the robot. A bumper to make sure that the robot qualifies for the competition, support the future brushes and hold the sensors. A Pixy2 would be installed at the front of the robot to recognize the debris. Two IR sensors would be placed in the front left corner of the robot to locate the center structure of the robot and walls the robot would may collide in. For the code the software section of team would use the side IR sensor to recognize where the center structure is and drive a counter clockwise orbit about it. The code would cause the robot to come closer if it is too far away, drive away from the center if it is to close, and keep proper orbit if it is at a safe distance. The exit home section of the code would simply recognize home color and then enter the second zone.

Version 2

The robot at this point can gather most of the points through exiting and orbits, but to make the robot competitive, it must be able to gather debris. The Hardware section of the team would start to implement brushes into the robot. The brushes would spin inward to take in the debris. Also the internal elevator walls would be added so the gathered debris would not hit the internal electronic components. The gears would be used to make a connection from the motors to the brushes. Wheel hubs would be designed to make sure that the wheels are the proper

distance away from base walls but do not collide with the bumper. The wheels were placed within the bumper region based on the rules of the competition. The software team would develop the major gather debris section. First the robot would be programmed to go off route when the Pixy2 sees a colored debris. It would center the mouth of the robot about debris, turn on the brushes and gather the debris. If there were two debris in the elevator that has not been sorted, then it would not go off route.

Version 3

At this point the team would be about a month out till competition and should have most of the major subsystems operating. The hardware team would add the sorting module to the robot, this would include the sorting box, base plate two and elevator. The elevator would be three major parts: elevator pad, elevator screw, and guide beam. The debris would rest on the elevator pad, the elevator screw would raise the pad, and the guide beam would make sure it didn't move around too much. The sorting box, would be made up of two major parts, the four chamber box and sorting gear. The box would hold up to twelve debris, both cubes and spheres, and hold them based on respective color. The sorting gear would mesh with the driving sorting box stepper motor. The base plate two would make sure that the debris does not fall back into the robot. The software team would add the code to make sure that the robot turns home and sorts the debris. . The robot would make one more complete orbit in the last thirty seconds of the round. While doing so the robot would find where it is, and find the proper launch spot. This was found by going to the color line in zone 2 that came before the home corner color. To do this, the robot reads the color lines in zone 2 till it reaches the target color, stops, and then prepares for launch. It would launch at the specified angle, drive home, and stop before hitting the wall.

Version 4

The would now be a couple weeks away from the competition and should be double checking each system and preparing for the scenario if they pass the qualifying rounds. This version of hardware update was added later in the semester due to issues of fixing the robot. The robot main base was one whole part, making it difficult to add electronics and add print time to make backups. The base plate was broken up to ten parts to resolve the issues. The print time drop from two days to three hours, when using multiple printers. The team was able to assemble the robot within one day, compare to prior taking four days. A flag would have been added, to make sure the robot had the opportunity to gather all possible points. The software team would add code to raise the flag and avoid other robots in rounds after the qualifying rounds. The flag would raise simply from horizontal to vertical. For the robot to avoid other robots, it would use the front IR sensor as it approached the other robot. If the robot was in proper orbit, the robot would speed up and go around the other robot. Otherwise, it stop behind the other robot.

Final Design

The final design of the 2019 Southeast Con Hardware Competition robot consists of the drive, bumper/brush, elevator/control and sorting assemblies. The primary material used to build this robot is 3D printed PLA. This material was chosen for its low impact resistance as well as its ability to be used in rapid prototyping.

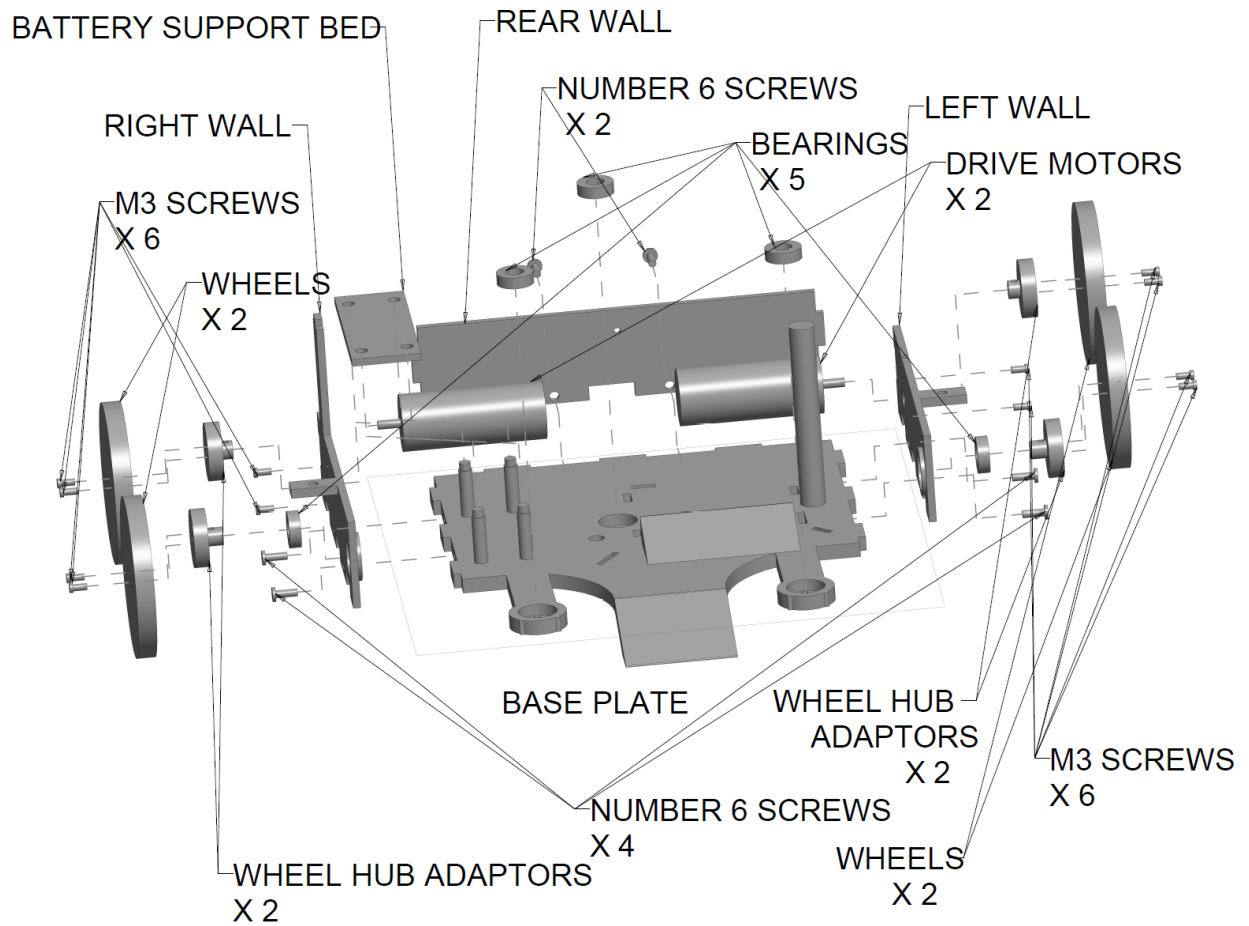


Figure 2-Exploded view of the drive assembly

The drive assembly consists of two drive motors, a base plate containing supports for the elevator assembly, mounting walls for the wheels and wheel hubs. The primary function of this assembly is to reliably drive and control the robot's position.

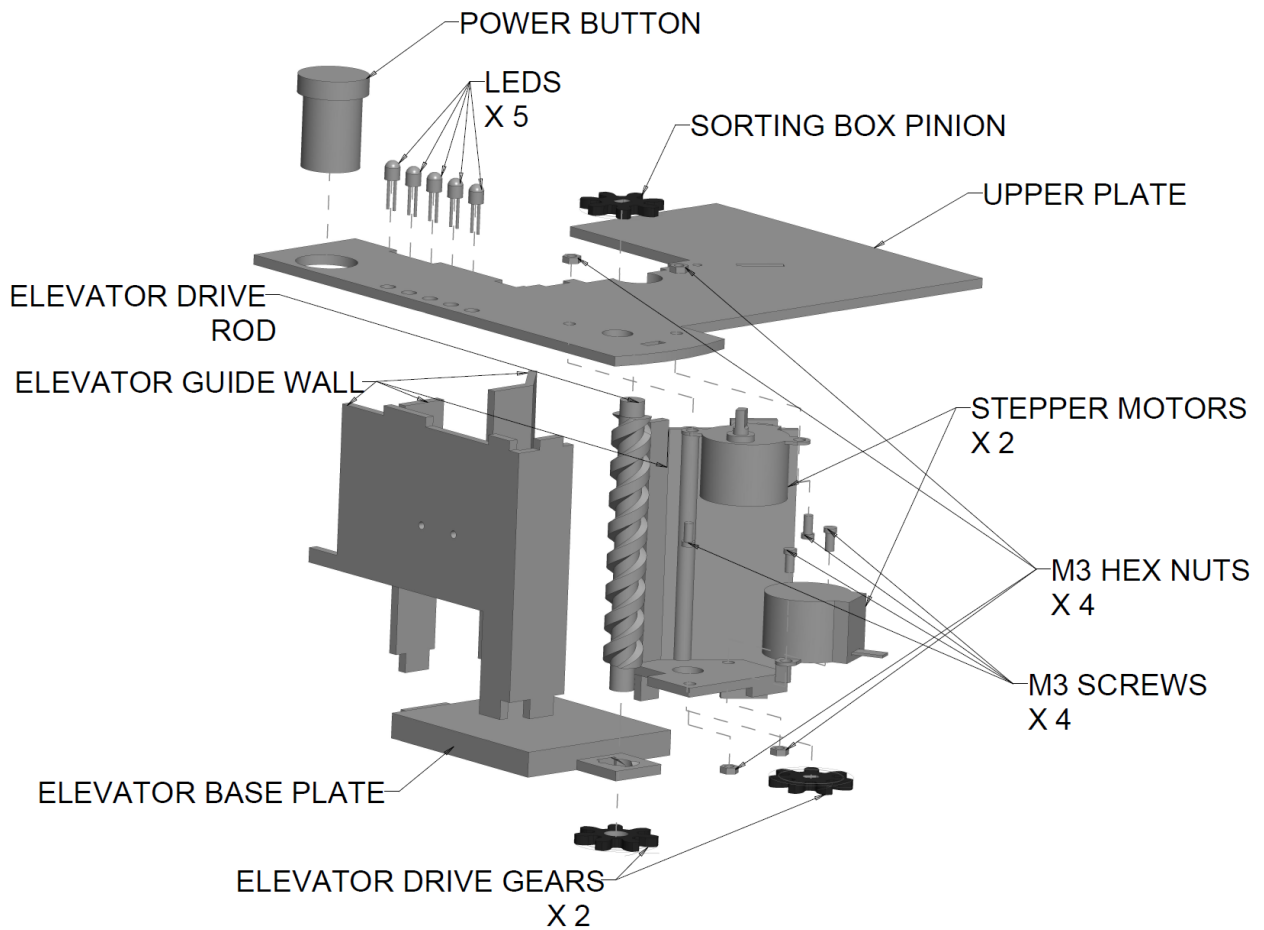


Figure 3-Exploded view of the elevator/control assembly

The key components of the elevator/control assembly are the power button, five LEDs, elevator drive rod, two stepper motors, and elevator guide wall. The primary functions of this assembly to raise the collected debris to the sorting system, drive the sorting box, turn the robot off and on, and display the robot's current state using the LEDs.

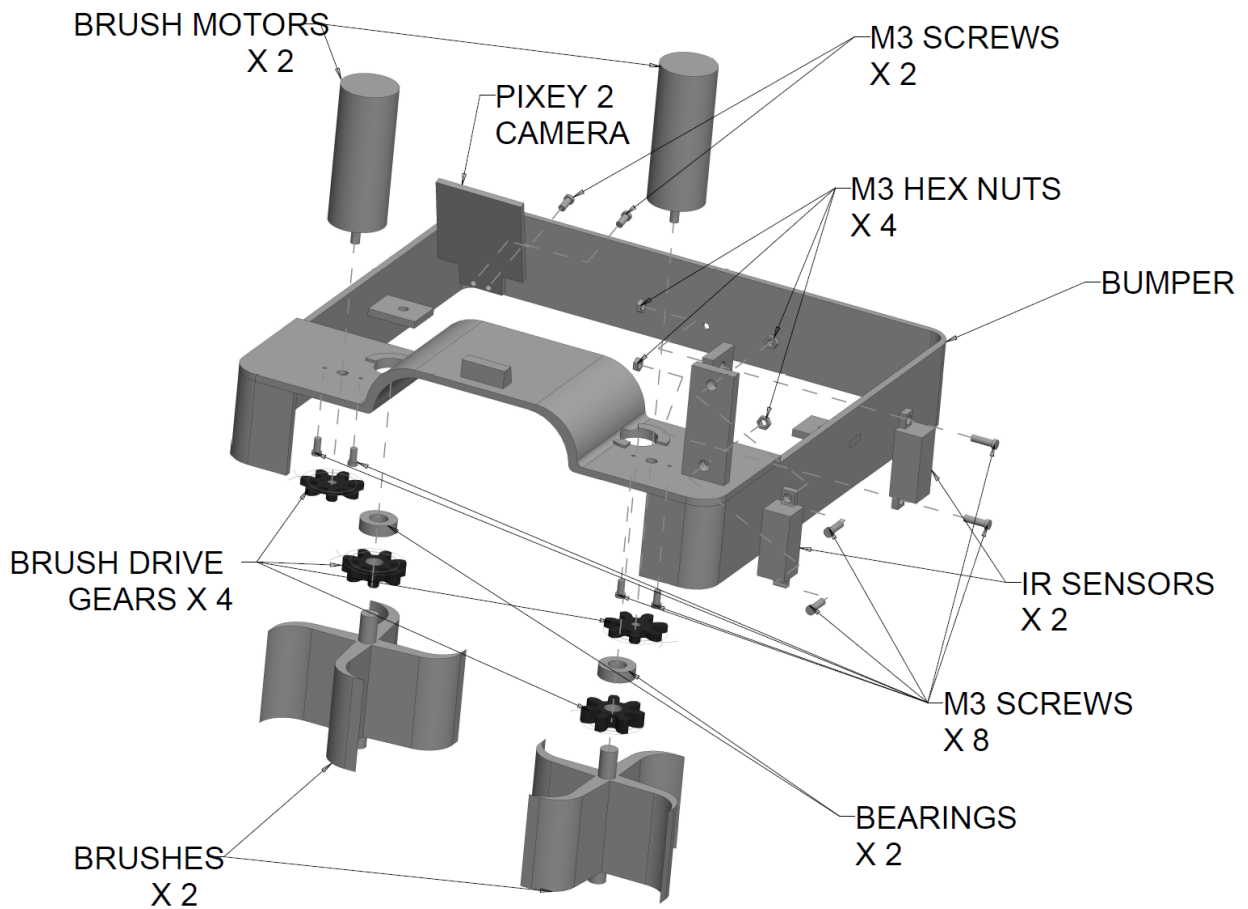


Figure 4-Exploded view of the bumper/brush assembly

The bumper/brush assembly consists of two brushes, two IR sensors, a Pixey 2 camera, and a bumper. The bumper serves as a mounting surface for the brushes, sensors and camera. The IR sensors and Pixey 2 camera are used to localize the robot's position and to located debris on the field. The primary functions of this assembly is to protect the robot from collision, located debris and gather the debris using the brushes.

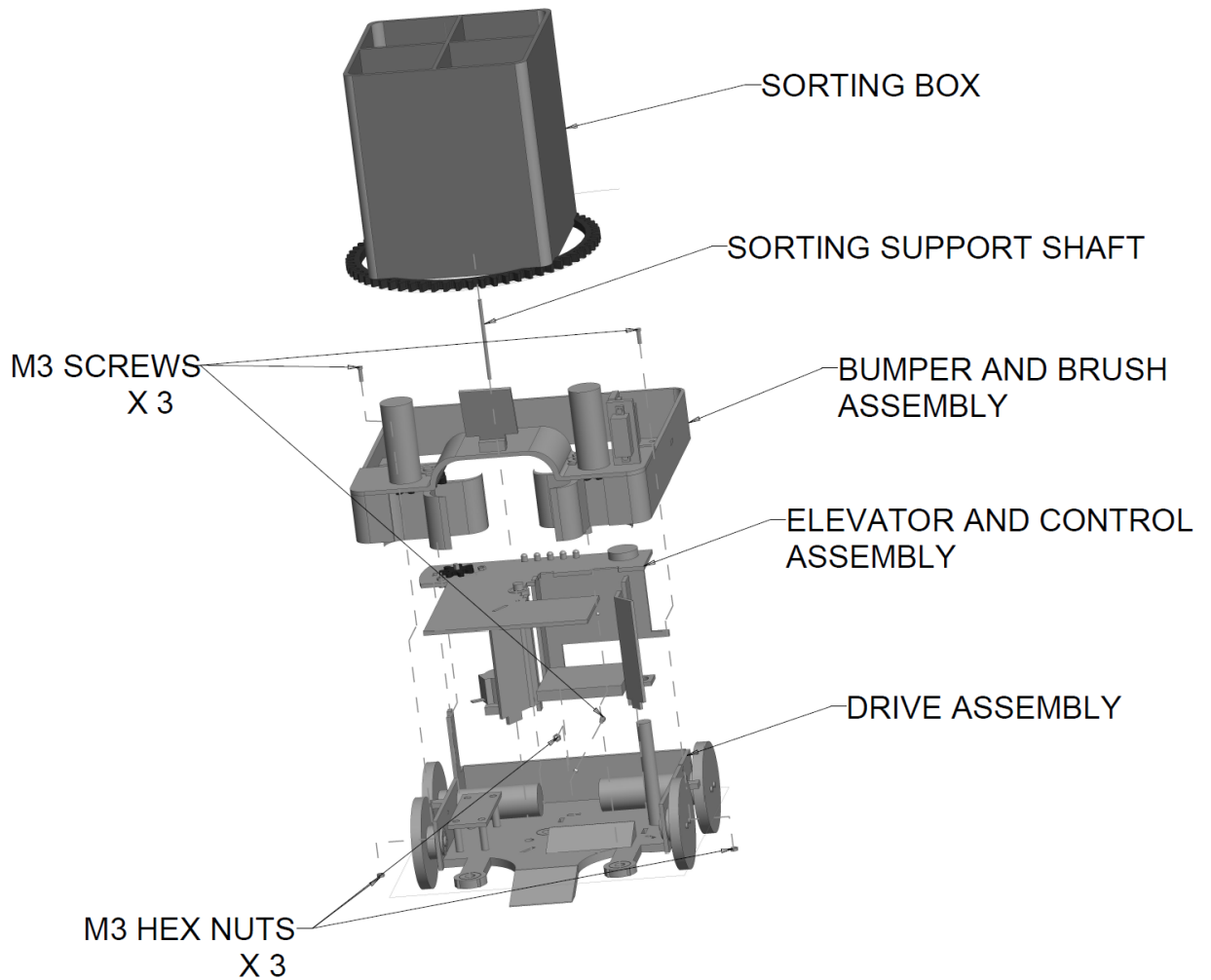


Figure 5-Exploded view of the complete robot

Using the tabs and brackets the robot can be quickly assembled or modified. The key component of the above figure is how to mount the sorting box to the top of the elevator assembly.

Results

Table 36-Key specs of the robot's performance

Max Speed	2.67 ft/s
Total Weight	3.8 lbs.
Debris Storage Capacity	12 Debris
IR Sensor Max Range	150 cm
Battery Life	27 Min.
Brush Speed	60 RPM

The table above displays some of the key robot specs used to determine the quality of the robot compared to the targets listed in chapter 1. The max speed of the robot is 1 ft/s less than the marginal value of 3.6 ft/s and almost 2 ft/s less than the ideal value 4.5 ft/s. While this is a short coming regarding the original target the maximum speed was reduced to ensure that the robot could move precisely around the field and could carry all debris placed on the field. The original marginal target for total weight was 4.4 lbs. with an ideal target of 3.5 lbs. The completed robot with all components and wires assembled weighed 3.8 lbs. which is well within the targeted parameters. For debris storage the ideal target was 10 debris and the completed robot was able to surpass this target and can gather all 12 debris. While the max IR sensor range, battery life, and brush speed are not listed in the targets these values are crucial to evaluating the robot's overall performance.

During the 2019 Southeast Con Hardware Competition there were some setbacks to the robot's hardware that caused a significant decrease in overall performance. The first of these setbacks was both the primary and back up lithium polymer batteries failing. These had to be replaced with two lead acid batteries. The main problem with the replacement of the batteries is the lead acid batteries weighed roughly 5 lbs. (the LiPo weighed 0.25 lbs.) and to fit the

new batteries on the robot the elevator and sorting systems needed to be removed. The second major setback was when the batteries failed so did the main processor. Fortunately, the modularity of the system allowed for the processor to be quickly replaced with a backup. The final setback to the competition was recalibrating the robot to the field and light conditions after replacing the batteries. This recalibration had to factor in the additional weight of the robot as well as the change in voltage from the new batteries.

2.2 Testing and Validation

2.2.1 Testing

The table below shows the expected results for the robot compared to the results of the final robot. The final robot excelled in debris storage, debris identification, and closest distance to collision. The initial target for debris storage was to be able to store a portion of the debris on the field, while the final robot was able to hold all the debris on the field. Similarly, the initial target for debris location was 10 debris and the final robot was able to successfully locate all the debris on the field. Optimization of the closest distance to collision was found through serendipity. During the initial calibration of the object detection and stopping code arbitrary constants were chosen and used in the brake code, and with minor adjustments the robot could reliably stop 3 inches from collision.

The key targets the robot met or came close to meeting are minimizing the sized of the robot and detection/avoidance reliability. The target for minimized size of the robot was the maximum size to qualify for competition, and the final robot met those dimensions precisely and exceeded the height dimension by being a half inch shorter than the target dimension. The

Team 305: Southeast Con 2019 Robotics Competition Team 72

Other key target was the robot’s detection and avoidance reliability. This target called for 95 % accuracy in detecting and avoiding obstacles (walls and spacetels), and due to signal noise in the detection sensors the final robot only had a 90 % reliability of detecting obstacles.

The targets that the final robot fell short on were the speed of the robot and debris gathered. Unfortunately, the maximum speed of the robot was only 2.67 ft/s falling a full ft/s short of the targeted value of 3.6 ft/s. This was primarily due to the need to choose between a dc motor that had enough speed to reach the target value but lacked the torque to move the robot or a motor that had enough torque to move the robot. The final key component that was unachieved was how many debris the robot could gather. The initial target was 7 debris gathered; however, the final robot was only able to gather 3 debris in a round.

Table 37- Southeast Con 2019 Test Results

Target	Expected	Result
Speed of the robot	3.6 ft/s	2.67 ft/s
Minimized size of the robot	9 x 9 x 11 in	9 x 9 x 10.5 in
Closest distance to collision	6 in	3 in
Space debris gathered	7 debris	3 debris
Space debris storage capacity	10 debris	12 debris
Detection/avoidance reliability	95 %	90 % (walls and spacetels)
Successfully located debris	10 debris	12 debris

The tests to collect the necessary data for the robot’s performance to be compared against the initial targets designed by the IEEE Southeast Con 2019 Hardware Competition team were set up to gather data as if the robot was competing. For example, the speed of the robot was tested on similar carpet specified by the competition and the robot’s drive system was set to the same specifications as if it were competing. While this led to a lower top speed of the robot it is a

more accurate representation of the robot's capability. The dimensions and debris storage capacity were tested by using a tape measurer and placing debris into the storage mechanism manually until it was full. The closest distance to collision was tested by examining how quickly the robot could break once an obstacle was detected (primarily walls and spacetels). This was also coupled with the detection/avoidance reliability test which simply tested whether the robot could successfully identify an obstacle. To determine if the robot could successfully locate debris the Pixy 2 camera used by the robot was connected to a laptop and set to stream what it was seeing as well as what it was identifying as debris. This test was conducted while the robot was in motion around the field, and in the course of several orbits the robot was able to identify each piece of debris. The final test conducted by the team was debris gathered in a round. This was the final phase of testing that occurred once all the hardware and core programming had been completed. To test how many debris the robot could gather in a round of competition the robot preformed a series of rounds and the number of debris gathered each round was averaged to obtain the result listed in the table.

A few of the major changes to the robot's design came from trying to meet these targets. For example, in order to attempt to reach the targeted speed the primary material of the robot was changed from steel and aluminum to 3D printed PLA. This change allowed for the robot's weight to drop significantly to a final value of 3.8 lbs. Another major change inspired by the targets switching to a more modular design in order to further minimize the size of the robot while still being able to easily preform necessary maintenance and troubleshooting.

2.2.2 Validation

The validation of the robot was conducted in Huntsville, Alabama at the IEEE Southeast Con 2019 Hardware Competition. This competition served as the ideal testing ground for the team to examine how well the robot could perform in comparison to similar robots. There was a total of 41 teams competing, and the difference in the approach to the competition varied greatly between each of the teams. Unfortunately, the FAMU-FSU team placed 31 in the competition.

A key piece of advice to anyone going to a competition is Murphy's Law ("If anything can go wrong it will go wrong.") is always in effect. The night before the competition began the team suffered a catastrophic failure of the primary LiPo battery, secondary LiPo battery, and Arduino board all failing. To fix these issues in time for the competition two lead-acid batteries were purchased, and a back-up Arduino board was connected to the robot. Due to the increased size of the batteries some of the components of the robot had to be removed to remain within the size constraint of the competition. This led to the removal of the sorting system, debris storage, and elevator assembly. Another major setback was the time required to recalibrate the robot's drive system to account for the increased weight (roughly 5 lbs. of additional weight).

Despite these numerous problems the robot was still able to successfully qualify, and complete two rounds of the competition. During the numerous practice runs the robot performed, both at the competition and away from it, the 3D printed structure of the robot did not fracture or break. Even the bumper which was occasionally slammed into a wall or the center structure of the field maintained its structural integrity. It is also important to note that base of the robot was able to support the increased weight from the batteries without fracturing or

warping. Though the robot suffered some catastrophic failures shortly before being able to complete the team did successfully preform on the spot modifications and adjustments to ensure it could compete. As a result of this dedication the robot was able to earn more points than 10 other competitors.

2.3 Scholarship in Practice

The Southeast Con Robot Design Team main target throughout the development process of the Southeast Con robot (SCR) was to have a completely autonomous robot, capable of navigating a playing field and collecting various debris (cubes and balls) of varying colors (red, blue, green, yellow) scattered across the field. Additionally, the robot needed to have the capability to travel in a counterclockwise orbit on the circular playing field, in which a colored column is place at the center of the field and four LED lights placed at the outer edge of the field at set distance so that the field is evenly divided into four quadrants. With colored lines placed between the center column and LED lights marking quadrant division, the robot had to be able to navigate the field while collecting debris and avoiding both the column, the LED lights, and other unidentified objects. In order to handle all these obstacles on the field, the final robot design required various revisions throughout the development process; with some aspect of the robot having some minor modifications to correct problems encountered during testing, while other sections being completely redesign or replacing due to their failure to meet the goals set forth by the team and/or competition rules.

2.3.1 Pixy2 Camera

One of the main functionalities that the autonomous robot had to possess was the ability detect colored debris (cubes and balls) and differentiate between the four possible colors those

objects could be (red, blue, green, yellow). At the early stages of the design, the team initially considered using various color sensors placed at different locations at the base of the robot to increase the probability of detecting a colored object. However, this approach would be extremely unreliable, as both the center column and the dividing lines between the four quadrants on the playing field were of the same colors as the debris; so therefore, it would prove to be very difficult to determine whether the sensor detected debris or parts of the playing field. Revising the approach to this problem, it was determined that the best method for dealing with detecting the correct objects was through the use of a camera sensor/module. Initially the TTL Serial JPEG Camera was considered for resolving with the detection problem, but, because this camera could only produce NTSC video and take color snapshots, a substantial amount of image process would be required which in itself required substantially more processing power than what the selected microcontroller could provide; so therefore, an additional processor solely dedicated to images processing would be required for this method to work. After comparing the TTL camera with other similar cameras compatible with the selected microcontroller, the Pixy2 CMUcam5 Sensor was selected since this sensor proved to be a substantially more powerful camera sensor than anything else, due its dedicated onboard image processor and its ability to determine the color, width, height, and distance of an object; which significantly improved the time for configuring the robot to only collect specified debris. With some minor changes to the setting of the Pixy2 camera, it was possible to accurately differentiate between debris and non-debris of the same color.

2.3.2 DC Motor

As one of the most important part to enable movement, choosing the correct motor with specifications that did not exceed any constraints set forth for the robot proved to be one of the more time-consuming process. One of the main constraints for the robot was that it had to be completely self-contained, so therefore, the possible power supply for the motors were limited to the battery capacity selected. Because the selected battery (2200 mAh) also had to supply power to other components of the robot, a 12v stepper motor (400 mA) was initially considered, as this motor had sufficient torque (40Ncm) to move the robot, without reducing the battery life significantly. However, this chose failed to meet the set target, as the dimensions of the motor exceeded the size constraint set forth to guarantee that all electronic components would fit within the body. Furthermore, it was determined that a stepper motor would not provide sufficient speed to meet the desired speed of 4 ft/s. After ruling out the use of stepper motor, the next chose was the use of brushless DC motor. With the need for only two 12V DC motors (450 mA) operating at 500 RPM to move the robot at speed that meets the target speed.

2.3.3 IR Sensor and Encoder

With the robot being completely autonomous, one of the main issues that had to be addressed early in the design process was the issue of preventing any collision with obstacles on the field. The robot had to be able to determine its position on the playing field at all time and adjust its travel direction if was too close to central column or if was about to collide with any unidentified object. To alleviate this problem, IR proximity sensors with a detection range of 10 to 80 cm were chosen. Initially, only one IR sensor was placed at the left side of the robot, as this would guarantee that the robot would adjust its travel direction if it traverses too close or too far

from the central column and therefore avoid any collision with the LED lights. The initial thought process of using only one IR sensor was that the Pixy2 camera at the front of the robot could be used to avoiding anything that was not a colored debris. However, this approach failed to meet expectations, as it was soon discovered that the pixy would also need to know the color of the object that it should avoid to properly avoid the object. Now this would not be an issue if all possible obstacles were known before the completion of robot, and therefore, it would be easy to program instructions that would take the color of those obstacles into consideration when traversing the field and avoid those objects. However, this is unrealistic due to the countless possibilities in color combinations that an obstacle could be. To correct this problem, an IR was placed at the front of the robot at a height that would prevent any unnecessary debris detection but low enough to detect the LED lights and any other object with the same size or greater than the lights. However, even with the introduction of an additional IR sensor, another issue arose during testing, which mainly dealt with the unreliability of IR sensor when it came too close to an object, as this prevented the receiver end of the sensor to detect light being emitted by the light emitter. Since the position of the two IR sensors differ, one solution was not possible, so therefore, an encoder was used in conjunction with the side IR sensor, while a break method approach was used for the front facing IR sensor. With the assistance of an encoder, it was possible to monitor the rotation speed of the left motor and therefore, determine the turning angle of the robot on the field. Since the right motor rotated at a constant speed and the left motor varied in speed, detecting a slower rotation speed of the left motor indicated that it was orbiting closer the central column, so therefore, the robot had to adjust position so that the IR sensor can still detect the central column correctly. The front sensor proved to be far easier to correct with

the use a break sequence, in which the robot would stop for a few seconds, then turn left a little more if the front sensor reading reach the critical detection distance; which in this instance was set to 30 cm.

2.3.4 3D Printing

Early in the development of the autonomous robot, a steel base and aluminum walls were used for testing purpose, however, due to the weight of the both the steel and aluminum the robot suffered a decrease in movement speed. Although the steel and aluminum provide excellent protection for the robot if it were to collide with any object, the high strength of the metal made bending its form very difficult. Since one of goal set forth by design team was to produce a robot that had a light weight and flexibility body, yet durable enough to withstand low impact collisions. The initial prototype used for testing was not enough to meet the criteria. The durability and corrosion resistance of steel and aluminum made it ideal to work with. Various grades of aluminum were examined for this purpose until arriving to the selection of aluminum 6060 due to its lower rigidity. Initially, this material proved to work flawlessly, allowing for the bending and cutting off section of the frame during the assembly of the first prototype. However, as various major design modifications were made to address issues encountered during testing, such as the placement of motors and microcontroller, the time spent on having the metal reassembled greatly reduced the development progress of the robot; as each piece of the metal had to be re-welded to the desired form. To increase productivity, it was determined that the use of 3D printed parts would be the ideal approach; each component would require less than half the time to produce, and the material used for printing also had the desired durability required.

2.3.5 Separating the Bumper from the Base

At the start of the robot design, it was decided that both the base and side bumper of the robot were to be one piece, therefore, reducing the amount parts needed for assembling the robot. This decision brought forth some issues when it came time to place certain electronic parts on the base of the robot. Due to the limited amount of space available to orientate the microcontroller, motors, and battery the robot, it proved to be at times difficult to wire up and test each electronic component. Furthermore, there were various electronic components needed to run the robot correctly. Due to space constraints disassembling and reassembling the robot for maintenance and troubleshooting became a time-consuming process. To alleviate this issue the body of the robot was printed in small pieces connected by a system of snap-in tabs and screws. This design proved to be more modular and allowed for quicker assemble and disassemble when need.

2.3.6 Work in Parallel not in Series

Parallel and series are terms used to describe how electronic components are connected to each other. Series is when a two or more components are place end to end of each other, while parallel shares a common node. This can be used to describe the how the flow of workload goes. The team planned to work in parallel, working at multiple sections with different team members at the same time. However, within a few hours, the team work in series, where everyone was working on the same team. This lead to less being accomplished prior to going to the competition and idle hands while working on the robot. For future team projects would recommend a physical barrier to divide up the team. For physical barrier, would be putting the team members in separate rooms or on simply on different sides of the room. However keep productive communication lines open so the robot does not become disjointed.

2.4 Conclusions

The choice to use 3D printed PLA allowed for quicker prototyping than a machine shop would allow, and the use of multiple 3D printers allowed for the parts to be made in parallel. There were some issues with the 3D printed wheel hubs connected to the drive motors, because of the torque of the drive motors the wheel hubs d-shaft connector tended to be rounded out after roughly 20 runs. This problem was overcome by printing several extra wheel hubs and using the modularity of the system to easily replace them.

The gathering and sorting mechanism of the robot was mostly successful. The brushes can reliably pick up the debris, the elevator can raise and lower a full stack (3) of debris, and the sorting box can hold all the debris while still being able to rotate. However, the color sensor was not implemented due to time constraints, and the revised elevator and base plate were unable to be printed in time for the competition. Without these updated parts the sorting box is prone to jamming when taking the debris from the elevator.

The remaining components (drive assembly, bumper, IR sensors, Pixy2 camera) function reliably. The drive assembly occasionally experiences slip, but the IR sensors and Pixey 2 camera provide enough localization that the robot can correct for those errors. The Pixey 2 camera can reliably identify the debris and the colored lines around the field. Overall the final robot still has some minor issues but still meets many of the original targets selected for this project.

Future work

To further improve upon the robot design and function it is recommended to implement a color sensor inside of the elevator and update the design of the elevator base and the base plate to prevent the sorting box from jamming. Another recommended improvement is to implement more sensors for localization. Sometimes the IR sensors would give a false reading and the Pixey 2 camera was unable to correct the error before the robot ran into a wall or spacetels. General advice for any robotics competition team is start early, make several prototypes, always have a backup, and program in parallel with the construction of the robot.

Appendices

Appendix A: Code of Conduct

A.1 Mission Statement

Southeast Con Team is committed to ensuring a positive work environment that supports professionalism, integrity, respect, and trust. Every member of this team will contribute a full effort to the creation and maintenance of such an environment to bring out the best in all of us as well as this project.

A.2 Roles

Each team member is delegated a specific role based on their experience and skill sets. In an event where someone needs to step down from his/her role, the team will collectively vote on someone else to fill that position. Each role description and responsibility are laid out here-within:

Team Leader – Chase Sapp

Manages the team as a whole; develops a plan and timeline for the project, delegates tasks among group member according to their skill sets; finalizes all documents and provides input on other positions where needed. The team leader is responsible for promoting synergy and increased teamwork. If a problem arises, the team leader will act in the best interest of the project.

He keeps the communication flowing, both between team members and Sponsor. The team leader takes the lead in organizing, planning, and setting up of meetings. In addition, he is responsible for keeping a record of all correspondence between the group and ‘minutes’ for the meetings. During the Southeast Con competition, he can apply for an appeal if the team

disagrees with the judge on a ruling. Finally, he gives or facilitates presentations by individual team members and is responsible for overall project plans and progress

Financial Advisor and Lead Systems Engineer – Chendong Yuan

Manages the budget and maintains a record of all credits and debits to project account. Any product or expenditure requests must be presented to the advisor, whom is then responsible for reviewing and the analysis of equivalent/alternate solutions. They then relay the information to the team and if the request is granted, order the selection. A record of these analyses and budget adjustments must be kept. Takes charge of the control design aspects of the project. Keeps line of communication with the other leads.

Lead Mechanical Engineer – Kyle Voycheske

Takes charge of the mechanical design aspects of the project. Keeps line of communication with the other leads. He is responsible for knowing details of the design and presenting the options for each aspect to the team for the decision process. Work load that falls under the mechanical engineering umbrella, the Lead ME may break up the work amongst the team. Keeps all design documentation for record and is responsible for gathering all reports.

Lead Software Engineer – Fabio Trinidad

He is responsible of the EE, or IE design part in support of the project. He maintains line of communication with the other leads. Work load that falls under the electrical engineering umbrella, the Lead EE may break up the work amongst the team. He keeps all design documentation for record.

Navigation Specialist – Daniel Delgado

He is responsible of the CE design part in support of the project. He maintains line of communication with the other leads. Work load that falls under the computer engineering umbrella, the Lead Software Engineer may break up the work amongst the team. He keeps all design documentation for record.

All Team Members

- Work on certain tasks of the project
- Buys into the project goals and success
- Delivers on commitments
- Adopt team spirit
- Listen and contribute constructively (feedback)
- Be effective in trying to get message across
- Be open minded to other ideas
- Respect other roles and ideas
- Be ambassador to the outside world in own tasks
- Record what happens during meetings

A.3 Communication

The main form of communication will be over phone, slack, and text-messaging among the group, preferably phone as well as through regular meetings of the whole team. Email will be a secondary form of communication for issues not being time-sensitive. For the passing of information, i.e. files and presentations, slack will be the main form of file transfer and proliferation.

Each group member must have a working email and slack for the purposes of communication and file transference. Members must check their slack and emails at least twice a day to check for important information and updates from the group. Although members will be initially informed via a phone call, meeting dates and pertinent information from the sponsor will additionally be sent over email so it is very important that each group member checks their email frequently.

If a meeting must be canceled, an email must be sent to the group at least 24 hours in advance.

Any team member that cannot attend a meeting must give advance notice of 24 hours informing the group of his absence. Reason for absence will be appreciated but not required if personal. Repeated absences in violation with this agreement will not be tolerated. If three unexcused absences or five excused absences occur during the time that the project is active, the team member will receive a negative mark on the peer evaluation.

A.4 Team Dynamics

The students will work as a team while allowing one another to feel free to make any suggestions or constructive criticisms without fear of being ridiculed and/or embarrassed. If any member on this team finds a task to be too difficult it is expected that the member should ask for help from the other teammates. If any member of the team feels they are not being respected or taken seriously, that member must bring it to the attention of the team leader in order for the issue to be resolved. We shall NOT let emotions dictate our actions. Everything done is for the benefit of the project and together everyone achieves more.

A.5 Ethics

Team members are required to be familiar with the IEEE and NSPE Engineering Code of ethics as they are responsible for their obligations to the public, the client, the employer, and the profession. There will be stringent following of the IEEE and NSPE Engineering Code of Ethics.

A.6 Dress Code

Team meetings will be held in casual attire. Sponsor meetings and group presentations will be business casual to formal as decided by the team per the event.

A.7 Weekly and biweekly Tasks

Team members will participate in all meetings with the sponsor, adviser and instructor. During said times ideas, project progress, budget, conflicts, timelines and due dates will be discussed. In addition, tasks will be delegated to team members during these meetings. Repeat absences will not be tolerated, please refer to Communication for how repeated absences are handled.

A.8 Decision Making

It is conducted by consensus and majority of the team members. Should ethical/moral reasons be cited for dissenting reason, then the ethics/morals shall be evaluated as a group and the majority will decide on the plan of action. Individuals with conflicts of interest should not participate in decision-making processes but do not need to announce said conflict. It is up to each individual to act ethically and for the interests of the group and the goals of the project.

Achieving the goal of the project will be the top priority for each group member. Below are the steps to be followed for each decision-making process:

- Problem Definition – Define the problem and understand it. Discuss among the group.
- Tentative Solutions – Brainstorms possible solutions. Discuss among group most plausible.
- Data/History Gathering and Analyses – Gather necessary data required for implementing Tentative Solution. Re-evaluate Tentative Solution for plausibility and effectiveness.
- Design – Design the Tentative Solution product and construct it. Re-evaluate for plausibility and effectiveness.
- Test and Simulation/Observation – Test design for Tentative Solution and gather data. Re-evaluate for plausibility and effectiveness.
- Final Evaluation – Evaluate the testing phase and determine its level of success. Decide if design can be improved and if time/budget allows for it.

A.9 Conflict Resolution

In the event of discord amongst team members the following steps shall be respectfully employed:

- Communication of points of interest from both parties which may include demonstration of active listening by both parties through paraphrasing and other tool acknowledging clear understanding.
- Administration of a vote, if needed, favoring majority rule.
- Lead of the field in which the conflict falls, decides the outcome.
- Team Leader intervention.
- Instructor will facilitate the resolution of conflicts.



A.10 Statement of Understanding

By signing this document, the members of Southeast Con Team agree to all the above and will abide by the code of conduct set forth by the group.

CHASE SAPP
KYLE VOYCHESKE
Fabio Trinidad
Chendong Juan
Daniel Dodgato

Appendix B: Function Decomposition

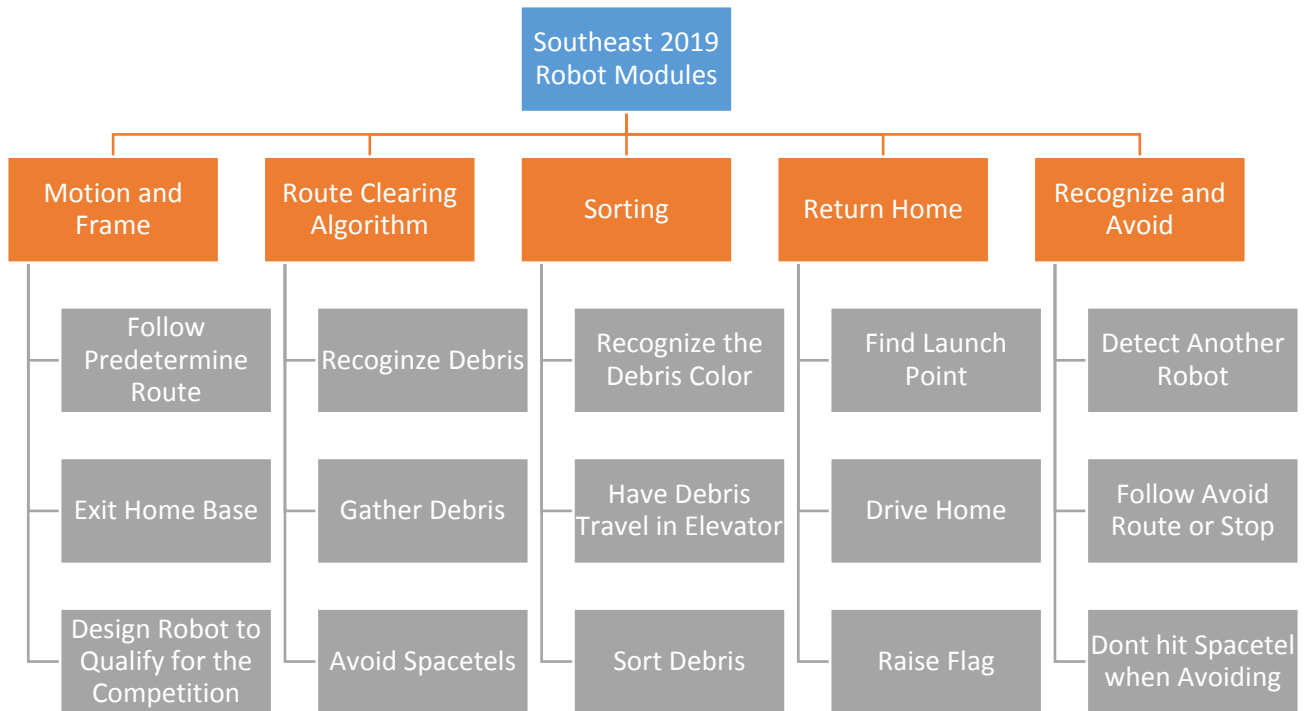


Figure 6-Southeast Con 2019 Function Decomposition

Appendix C: Target Catalog

Table 38-Target Catalog

Target No.	Need	Metric	Imp.	Units	Marginal Value	Ideal Value
1	1	Speed of the Robot	5	Ft/s	3.6	4.5
2	1	Maximum angular velocity of the robot	3	Rad/sec	Pi/3	Pi/2
3	1	Acceleration of the robot with various wheel configurations	4	Ft/sec	3.5	4.5



4	1	Minimized Size of the robot	5	inches	9 x 9 x 11	8.5 x 8.5 x 10
5	2	Time to search the field	3	Seconds	120	90
6	2	The closest distance the robot comes to collision	3	Inches	6	3
7	3	Space debris gathered	4	Space debris	7	10
8	3	Amount space debris sorted versus how many were gathered	2	%	80	90
9	3	Storage Capacity	3	In ³	242	192
10	4	Amount of debris being dropped off	3	debris	8	12
11	4	The percentage of successfully locating	3	%	80	100
12	4	The percentage of successfully returning	3	%	80	100
13	5	Detection and Avoidance	5	%	95	99
14	5	Color Differentiation	3	color of debris	10	12

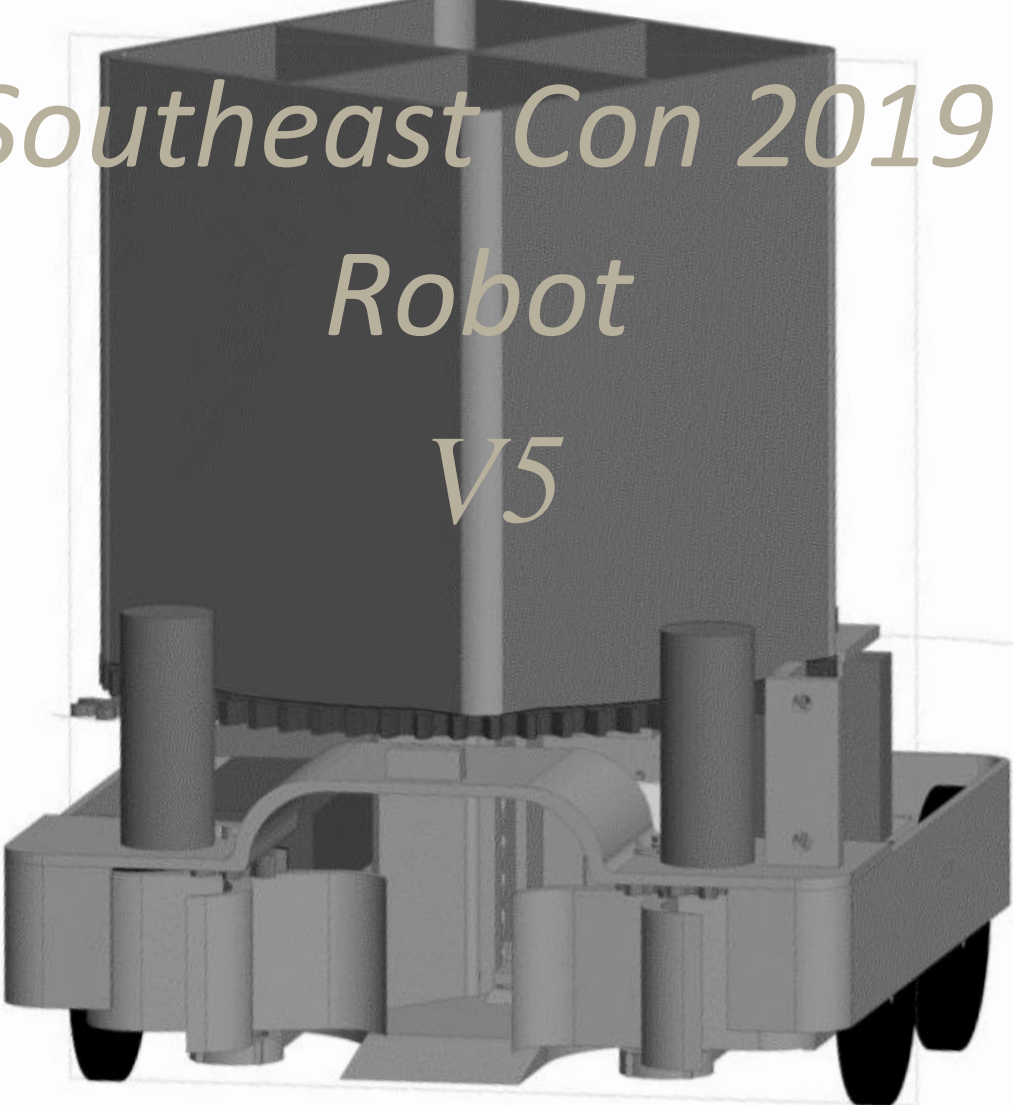
Appendix D: Testing Results

Table 39- Southeast Con 2019 Test Results

Target	Expected	Result
Speed of the robot	3.6 ft/s	2.67 ft/s
Minimized size of the robot	9 x 9 x 11 in	9 x 9 x 10.5 in
Closest distance to collision	6 in	3 in
Space debris gathered	7 debris	3 debris
Space debris storage capacity	10 debris	12 debris
Detection/avoidance reliability	95 %	90 % (walls and spacetels)
Successfully located debris	10 debris	12 debris

Appendix E: Operation Manual

Operational Manual



*Southeast Con 2019
Robot
V5*

By Chase Sapp, Kyle Voycheske, Chendong Yuan, Fabio Trinidad, and Daniel

Delgado

Team 305: Southeast Con 2019 Robotics Competition Team

94

2019

Operation Manual

Assembling the Robot	1
Circuit Layout	6
Code Guidance	7
Troubleshooting	9

Assembling the Robot

The Southeast Con 2019 Robot has a complex structural design due to the size restrictions of the competition and incorporation the mechanisms required to quickly identify, gather and sort debris. To assemble the robot it recommended to follow these steps to ensure all the parts attached correctly so that the robot is structurally sound, and all the necessary mechanisms work correctly.

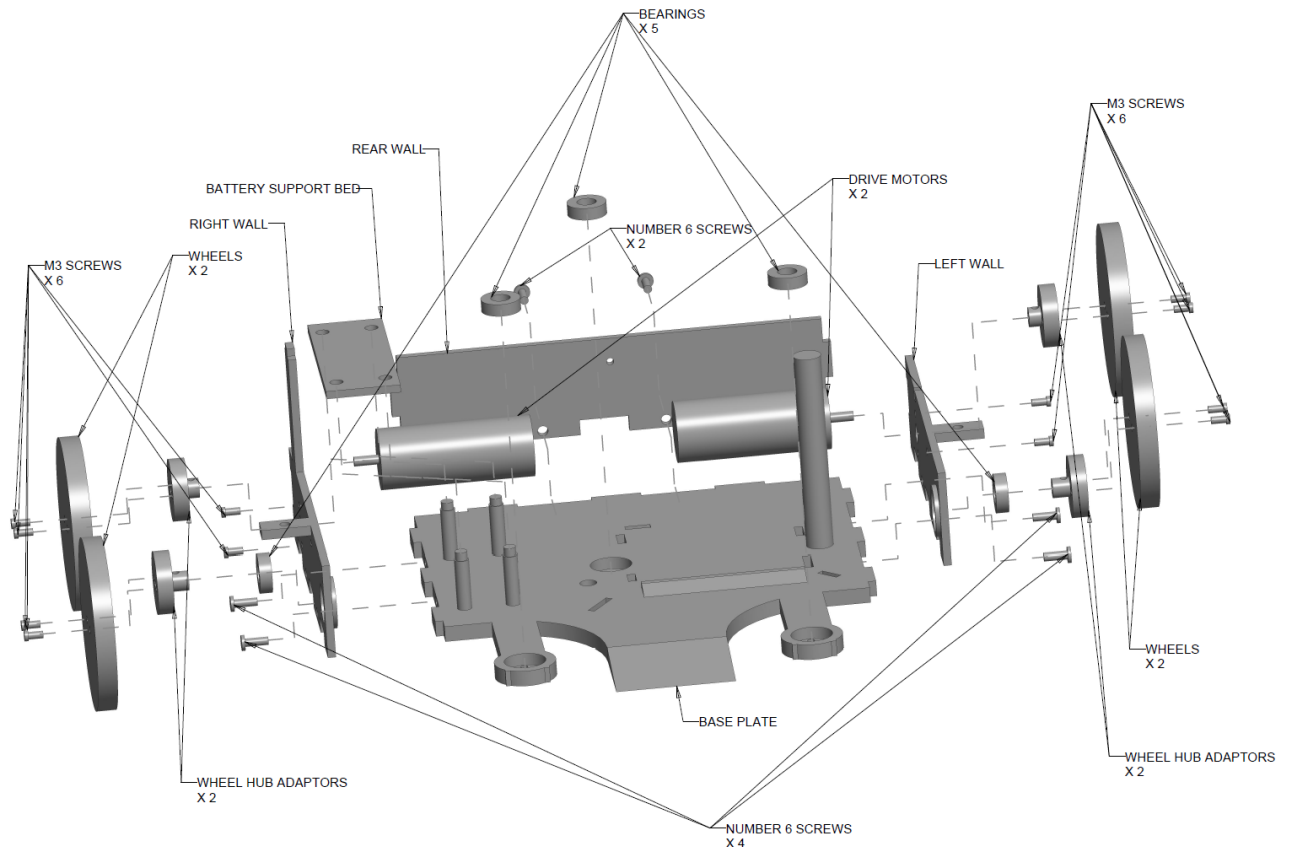


Figure 7- Base Assembly

To assemble the robot's drive assembly please follow these steps:

1. Attach the drive motors to the left and right walls using M3 machine screws.
2. Press fit the bearings into the left and right walls.
3. Attach the wheels to the wheel hub adaptors using M3 machine screws.
4. Press fit the wheel hub adaptors into the bearings and onto the drive motors shaft.
5. Press fit three bearings into the base plate.
6. Screw the right, left and rear walls to the base plate using number 6 sheet metal screws.
7. Place batter support bed on top of four support beams near the right wall.

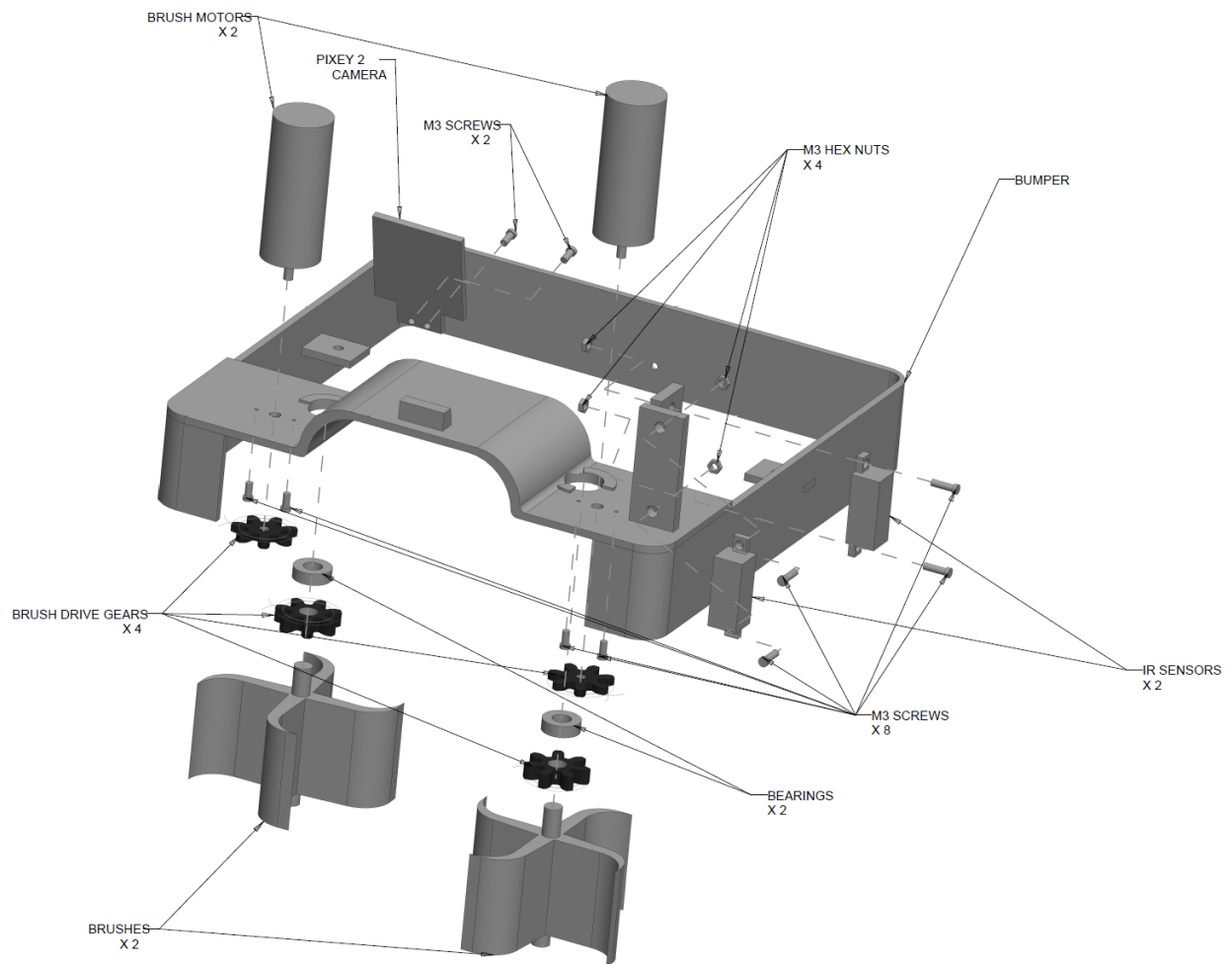


Figure 8-Bumper Assembly

To assemble the robot's bumper and brush assembly please follow these steps:

1. Attach the IR sensors to the bumper using M3 machine screws and hex nuts.
2. Press fit the bearings to the bumper.
3. Attach the brush motors to the assembly using M3 machine screws.
4. Press fit the brush drive gears to the brush motors and brushes.
5. Press fit the brushes into the bearings ensuring the gears mesh properly.
6. Attach the Pixey 2 camera to the bumper using M3 machine screws.

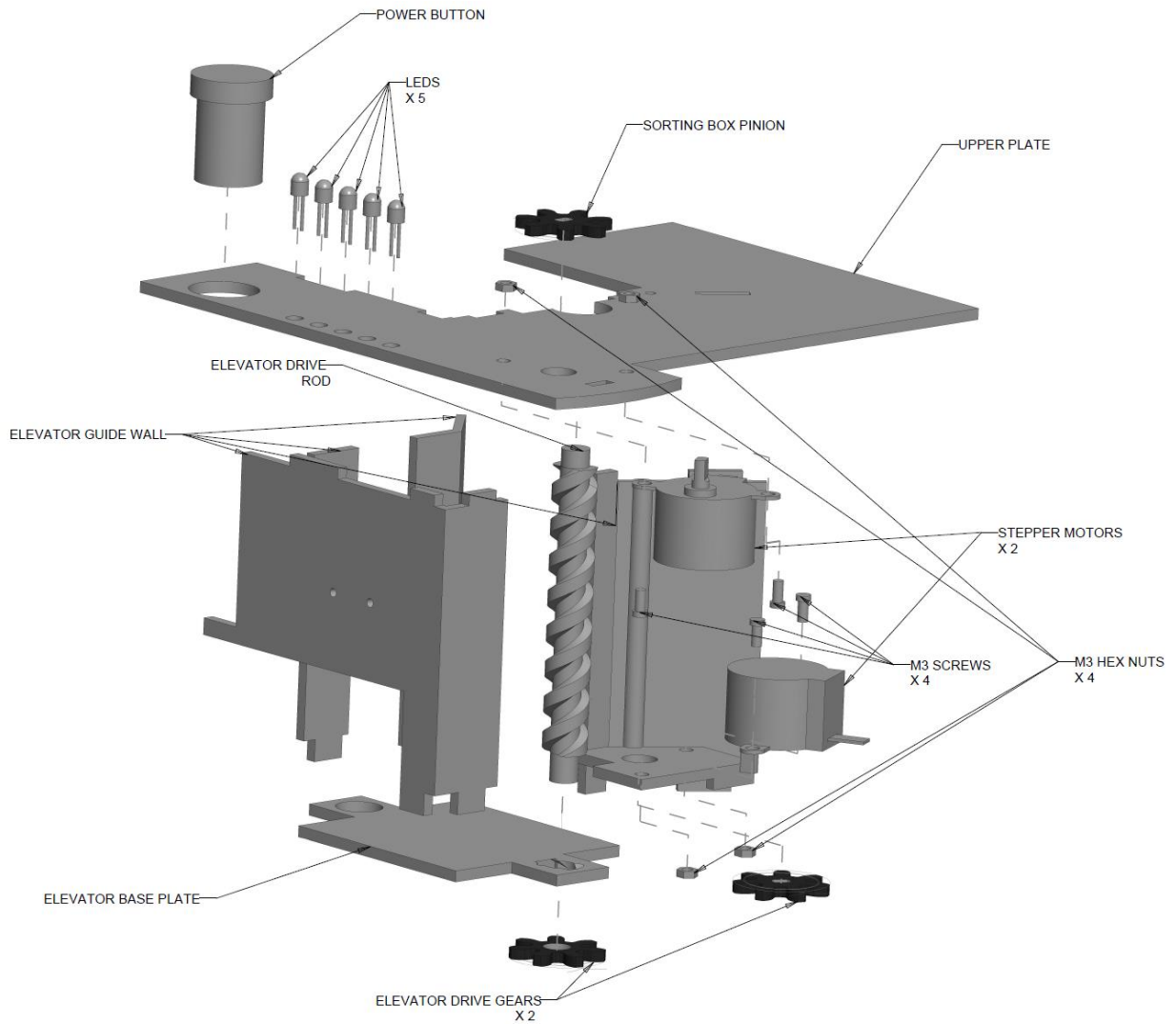


Figure 9-Elevator Assembly

To assemble the robot's elevator and control assembly please follow these steps:

1. Attach the stepper motors to the upper plate and elevator guide wall using M3 machine screws and hex nuts.
2. Press fit the elevator drive gear and the sorting box pinion to the stepper motors.
3. Thread the elevator drive rod through the elevator base plate.
4. Press fit the remaining elevator drive gear to the elevator drive rod.
5. Insert elevator guide wall tabs into the upper plate.
6. Insert the LEDs and power button into the upper plate.

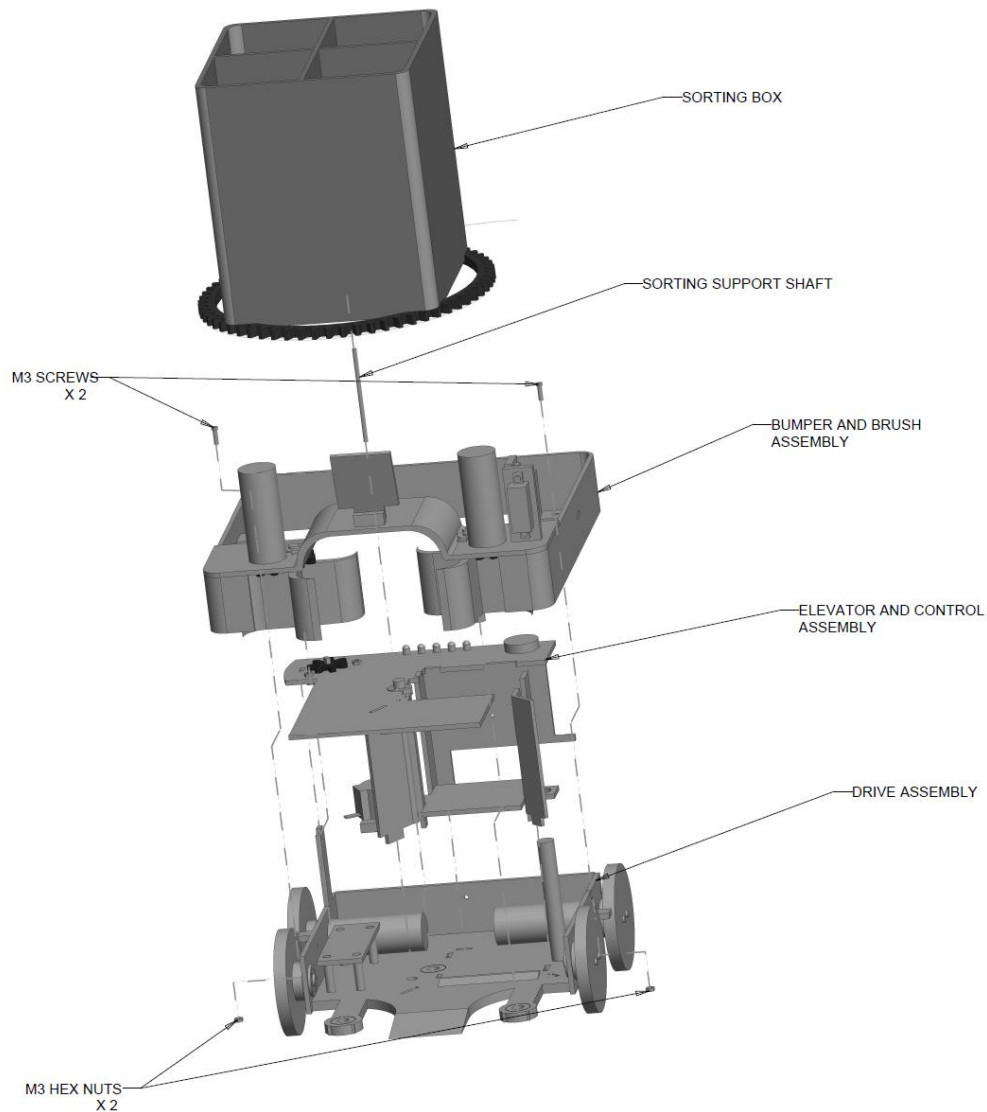


Figure 10-Overall Robot Assembly

To assemble final robot please follow these steps:

1. Attach the elevator and control assembly to the drive assembly using the tabs on the elevator guide wall.
2. Attach the bumper assembly to the drive assembly using M3 machine screws and hex nuts.
3. Insert the sorting support shaft through the upper plate and into the guide wall.
4. Place the sorting box on top of the sorting support shaft ensuring that the gears mesh properly.

Circuit Layout

Code Guidance

The Southeast Con 2019 Robot has a complex circuit design. This is due to having six motors, five LED's, three sensors, five driver chips, 12V battery, and an Arduino Mega on board. Also the circuit is compressed into a small box of about 9in by 9in by 4in. Lastly the components are driven with different voltages, so connecting wire in the wrong location can lead to smoke and/or burning out parts. If a wire would disconnect during the competition or show case follow these steps:

1. Locate each end of the wire.
2. Write down the component that it was connected to. If this side does not have a component connected, then check nearby devices while referencing this circuit layout if any pins are missing a wire. Write down that component.
3. Write down the pin number on the Arduino Mega where the other side of the wire was connected. If this side is not connected to a pin, then check nearby pins while referencing this circuit layout if any pins are missing a wire. Write down that that pin number.
4. Highlight the route on the Circuit Layout where it should be connected.
5. Follow the highlighted route and connect the pins to the proper location.

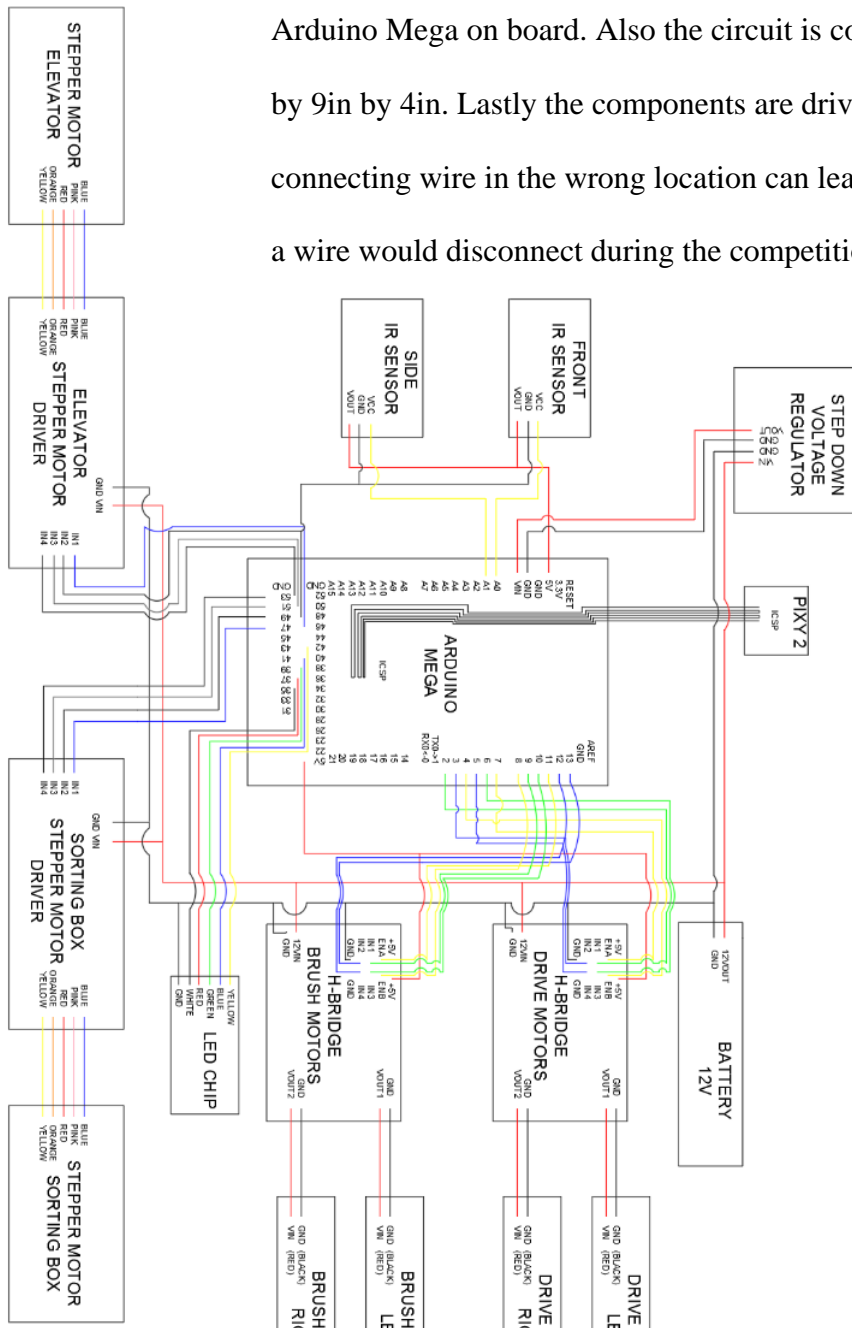


Figure 11-Overall Circuit Design

During the competition or when making adjustments to the robot a test is necessary for any components. In this section will provide test code for the Arduino IDE. It should be noted to **TURN OFF THE ROBOT BY THE ON/OFF SWITCH BEFORE CONNECTING THE ROBOT TO AN COMPUTER**. This avoids overloading the robot with too much voltage and/or current.

Brush Motor Test

Brush Motor test code is design to test the front brush motors of the Southeast Con 2019 Robot. These motor are connected at the front of the robot on the bumper. The supply and ground is connected to the *H-Bridge Brush Motor*, with a voltage that can range from zero to twelve volts. To control the speed of the brushes, simply set **brush_speed** from zero to one hundred percent. One hundred percent is the maximum speed the motor can go before damaging the motor and/or wired connected to the motor. It is recommended not to go above forty percent to avoid damage to the brush gear, bearings, bumper, and/or brushes. The **brush_direction** affects the direction in which the brushes are spinning. Setting it equal to two will cause the brushes to spin inward. Setting it equal to one will cause the brushes to spin outward. Setting it equal to zero will cause the brushes to stop. Only change the global variables. To view the code, please go to the end of the document.

Drive Motor Test

Drive Motor test code is design to test the drive motors of the Southeast Con 2019 Robot. These motor are connected to the side walls that are attached to base plate one. The supply and ground is connected to the *H-Bridge Drive Motor*, with a voltage that can range from zero to twelve volts. To control the speed of the wheels, simply set **driver_speed** from zero to one hundred percent. One hundred percent is the maximum speed the motor can go before damaging the motor and/or wired connected to the motor. It is recommended not to go below fifty percent, or the robot will not move. The **driver_direction** affects the direction in which the wheels are spinning. Setting it equal to two will cause the wheels to go forward. Setting it equal to one will cause the wheels to spin backward. Setting it equal to zero will cause the wheels to stop. Only change the global variables. To view the code, please go to the end of the document.

IR Sensor Test

IR sensor test code is design to test the drive motors of the Southeast Con 2019 Robot. These IR Sensors are connected to the bumper located near the left brush motor. The supply and ground is connected to the *Arduino Mega*, with a voltage that can range from zero to five volts. This code has two variables that return the distance from in front of the IR sensors. **SIR** or side IR sensor is located ninety degrees away from the front facing of the robot. This sensor is used to locate how far the sensor mass is away from the robot. **FIR** or front IR sensor is facing in the same direction of the opening of the robot. This sensor is used to tell the distance between the walls of the field and any opposing robots. Simply upload the code and open the serial monitor found in the Arduino IDE to see what the IR sensors are reading. To view the code, please go to the end of the document.

LED Color Test

LED Color Test code is design to test the LED's on the Southeast Con 2019 Robot. These LED's are connected to the bumper located near the rear by the ON/OFF switch. The supply and ground is connected to the *LED Chip*, with a voltage that can range from zero to five volts. This code sends on and off signals to each LED to cause them to flash. Upload the code and observe the LED light flash. To view the code, please go to the end of the document.

Elevator/Sorting Box Test

Elevator/Sorting Box Test code is design to test the drive motors of the Southeast Con 2019 Robot. These mechanisms are connected to the center of the robot. The stepper motors are connected to their respective drivers, with a voltage that can range from zero to twelve volts. Both of these modules are tested with the same code. The code causes each module to travel at the same speed. To change the direction of the elevator or sorting box change the global variable **rotationController**. The **rotationController** is set to "true", then the stepper motors turn clockwise. If the **rotationController** is set to "false", then the stepper motors turn counter clockwise. To view the code, please go to the end of the document.

Troubleshooting

The Southeast con2019 Robotics is a tightly enclosed complex system that has a mixture of mechanical and electrical components. During competition or showcasing the robot errors are bound to occur. This section will cover how to recognize and resolve these errors.

Brushes

- Brushes are not rotating: If this error occurs then there are two possible issues that are occurring with the robot. The first issue is that the brush gear has fallen off the brush DC motor. To resolve this issue first, check the original gear for any damage. If there is no damage replace the gear back onto the brush motor while still grinding with the gear above the brushes. If the gear is broken, replace the part by 3-D printing it. If this does not resolve the issue then check if there are any issues with the H-bridge or brush motor. To test this upload the *Brush Motor Test* to the robot and observe what occurs. If the brushes do not rotate still, check if any debris is obstructing the rotation. Lastly, if there is no debris obstructing the brushes then the motors are burned out and need to be replaced.
- Brushes are spinning in wrong direction: If the brushes are spinning in the wrong direction then there are one possible errors that could be occurring. The issue can occur because the supply and ground were improperly connected to the motor from the *Brush Motor H-bridge*. To resolve this issue flip the wires connected to the motor.

Wheels

- Wheels are spinning freely: This error occurs when the internal cavity of the hub attached to the motor is stripped. To resolve this issue, replace the hub.
- Wheels are not spinning: This error occurs when the motor is not receiving power, the robot has hit a wall and/or the wheels have become stuck. To test if the motor is burned out remove the wheel and hub from the motor and upload the *Driver Motor Test*. If the motor's rotor is not rotating then the motor is stalled out and needs to be replaced. If the wheel has something clogged in the wheel, remove the debris

IR Sensors

- No values being returned: This error occurs when the wires connected wrong or fallen out. Compare the connection to the Circuit Layout and replace the wires as needed.
- Wrong Values being returned: This error occurs when data yellow wires is not connect properly. Compare to the connection to the Circuit Layout and replace as needed.

Pixy 2

- The Pixy 2 is not returning values: This error occurs when the connection to the Arduino Mega is not connected properly. A sign of this error can be by a small LED below the camera of the Pixy 2 producing a red light. Rotate the pin connection 180 degrees, and the Pixy 2 will produce a multi-color lights.

Appendix F: Software

Showcase

```

#include <SharpIR.h>
#include <PIDLoop.h>

/*
sig 1 = red object
sig 2 = yellow object
sig 3 = green object
sig 5 = blue object
*/

// limits how fast to travel forward
#define MAX_VELOCITY 100

// bool kickstart = true;

// left brush pins
const int ENC = 8;
const int IN5 = 9;
const int IN6 = 10;

// right brush pins
const int END = 11;
const int IN7 = 12;
const int IN8 = 13;

// Elevator pins
int bluePin = 22; // IN1 on the ULN2003 Board, BLUE end of the Blue/Yellow motor
coil
int pinkPin = 24; // IN2 on the ULN2003 Board, PINK end of the Pink/Orange motor
coil
int yellowPin = 26; // IN3 on the ULN2003 Board, YELLOW end of the Blue/Yellow
motor coil
int whitePin = 28; // IN4 on the ULN2003 Board, ORANGE end of the Pink/Orange
motor coil

// Sort box pins

```



```

int SortIN1 = 23;
int SortIN2 = 25;
int SortIN3 = 27;
int SortIN4 = 29;

// LED pins
#define yellow 30
#define blue 32
#define green 34
#define red 36
#define white 38

// PI FOR DRIVE MOTOR
#define PI 3.1415926535897932384626433832795
int currentStep_elevator = 0;
int currentStep_box = 0;
unsigned long StartTime = 0; // track orbital time
int brushDirection=0;
int brushSpeed=0;
int current_position = 0;
int targetPosition = 0;

SharpIR FrontSensor(SharpIR::GP2Y0A41SK0F, A1); // set model of sensor and output
pin (A0)
SharpIR SideSensor(SharpIR::GP2Y0A41SK0F, A0); // set model of sensor and output
pin (A1)
int DistanceFront = 0; // distance dectection FrontSensor
int DistanceSide = 0; // distance dectection SideSensor

bool LeaveBase = false;
bool EndOrbit = false;
bool Return2Base = false;

void setup()
{
    // left brush pins
    pinMode(IN5, OUTPUT);
    pinMode(IN6, OUTPUT);
    pinMode(ENC, OUTPUT);

    // right brush pins
    pinMode(IN7, OUTPUT);
    pinMode(IN8, OUTPUT);

```

```

pinMode(END, OUTPUT);

// LED pins
pinMode(yellow, OUTPUT);
pinMode(blue, OUTPUT);
pinMode(green, OUTPUT);
pinMode(red, OUTPUT);
pinMode(white, OUTPUT);

// elevator pin
pinMode(bluePin, OUTPUT);
pinMode(pinkPin, OUTPUT);
pinMode(yellowPin, OUTPUT);
pinMode(whitePin,OUTPUT);

digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, LOW);
// sorting box
pinMode(SortIN1, OUTPUT);
pinMode(SortIN2, OUTPUT);
pinMode(SortIN3, OUTPUT);
pinMode(SortIN4,OUTPUT);

digitalWrite(SortIN1, LOW);
digitalWrite(SortIN2, LOW);
digitalWrite(SortIN3, LOW);
digitalWrite(SortIN4, LOW);

}

void loop()
{
  DistanceFront = FrontSensor.getDistance(); // reads the value of front sensor (in cm)
  DistanceSide = SideSensor.getDistance(); // reads the value of Side sensor (in cm)
  box_driver(2); // rotate sorting box ccw forever

  // elevator_smart_driver(7550);

  if((DistanceFront >= 4)&&(DistanceFront <= 9))
  {
    brushDirection = 1;
    brushSpeed = 90;
  }
}

```



```
brush_driver(brushSpeed, brushDirection);
}
else
{
brushDirection = 0;
brushSpeed = 0;
brush_driver(brushSpeed, brushDirection);
}

////////// Elevator ////////////
int LEDSelector = 0;
if ((DistanceSide >= 4)&&(DistanceSide <= 9))
{
elevator_smart_driver((1510*(DistanceSide - 4)));
LEDSelector = DistanceSide -4;

} else
{
elevator_smart_driver(0);
LEDSelector = 0;
}

LED_driver((LEDSelector));
}

////////// FUNCTIONS ////////////
////////// BRUSH MOTOR FUNCTION ////////////
// ALLOWS THE USER TO CONTROL THE OF BRUSHES
// DIRECTION AND SPEED.
// INPUT | DESCRIPTION
// -----
// brush_speed | THE SPEED THE ROBOT IS TRAVELING FROM STOP [0%] TO
MAX [100%]
// brush_direction | 2 = FORWARD || 1 = BACKWARD || 0 = STOP
void brush_driver (int brush_speed,int brush_direction)
{
int Speed = brush_speed*2;
if(brush_direction == 2)
{
Motor3_Forward(Speed);
Motor4_Backward(Speed);
}
if(brush_direction == 1)
{
Motor3_Backward(Speed);
```



```
    Motor4_Forward(Speed);
  } else
  {
    Motor3_Brake();
    Motor4_Brake();
  }
}
void Motor3_Forward (int Speed)
{digitalWrite(IN5,HIGH);
digitalWrite(IN6,LOW);
analogWrite(ENC,Speed);
}
void Motor3_Backward (int Speed)
{
digitalWrite(IN5,LOW);
digitalWrite(IN6,HIGH);
analogWrite(ENC,Speed);
}
void Motor3_Brake ()
{
digitalWrite(IN5,LOW);
digitalWrite(IN6,LOW);
}
void Motor4_Forward (int Speed)
{
digitalWrite(IN7,HIGH);
digitalWrite(IN8,LOW);
analogWrite(END,Speed);
}
void Motor4_Backward (int Speed)
{
digitalWrite(IN7,LOW);
digitalWrite(IN8,HIGH);
analogWrite(END,Speed);
}
void Motor4_Brake()
{
digitalWrite(IN7,LOW);
digitalWrite(IN8,LOW);
}
////////////////////////////////////
////////// ELEVATOR CONTROLLER FUNCTION //////////
void elevator_smart_driver(int target_position)
{
```



```
//drive elevetor to wanted position
if (target_position < current_position)
{
  if(current_position == 0)
  {
    elevetor_driver(0);
  }
  else
  {
    current_position = current_position - 1;
    elevetor_driver(2);
  }
}
else if (target_position > current_position)
{
  if(current_position == 7550)
  {
    elevetor_driver(0);
  }
  else
  {
    current_position++;
    elevetor_driver(1);
  }
}
else if (target_position = current_position)
{
  elevetor_driver(0);
}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////// ELEVATOR FUNCTION //////////
void elevetor_driver(int rotationController)
{
  if (rotationController == 1)
  {
    switch(currentStep_elevetor){
      case 0:
        digitalWrite(bluePin, HIGH);
        digitalWrite(pinkPin, LOW);
        digitalWrite(yellowPin, LOW);
        digitalWrite(whitePin, LOW);
        break;
      case 1:
```



```
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, HIGH);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, LOW);
break;
case 2:
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, HIGH);
digitalWrite(whitePin, LOW);
break;
case 3:
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, HIGH);
break;
}
}
else if(rotationController == 2)
{
switch(currentStep_elevator){
case 0:
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, HIGH);
break;
case 1:
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, HIGH);
digitalWrite(whitePin, LOW);
break;
case 2:
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, HIGH);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, LOW);
break;
case 3:
digitalWrite(bluePin, HIGH);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
```

```

    digitalWrite(whitePin, LOW);
    break;
  }
}
else if(rotationController == 0)
{
  digitalWrite(bluePin, LOW);
  digitalWrite(pinkPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(whitePin, LOW);

}
currentStep_elevator = (++currentStep_elevator < 4) ? currentStep_elevator : 0;

//2000 microseconds, or 2 milliseconds seems to be
//about the shortest delay that is usable. Anything
//lower and the motor starts to freeze.
//delayMicroseconds(2250);
delay(2);
}
////////////////////////////////////
////////// SORTING BOX FUNCTION //////////
void box_driver(int rotationController)
{
if (rotationController == 1)
{
switch(currentStep_box){
case 0:
  digitalWrite(SortIN1, HIGH);
  digitalWrite(SortIN2, LOW);
  digitalWrite(SortIN3, LOW);
  digitalWrite(SortIN4, LOW);
  break;
case 1:
  digitalWrite(SortIN1, LOW);
  digitalWrite(SortIN2, HIGH);
  digitalWrite(SortIN3, LOW);
  digitalWrite(SortIN4, LOW);
  break;
case 2:
  digitalWrite(SortIN1, LOW);
  digitalWrite(SortIN2, LOW);
  digitalWrite(SortIN3, HIGH);
  digitalWrite(SortIN4, LOW);

```



```
        break;
    case 3:
        digitalWrite(SortIN1, LOW);
        digitalWrite(SortIN2, LOW);
        digitalWrite(SortIN3, LOW);
        digitalWrite(SortIN4, HIGH);
        break;
    }
}
else if(rotationController == 2)
{
    switch(currentStep_box){
    case 0:
        digitalWrite(SortIN1, LOW);
        digitalWrite(SortIN2, LOW);
        digitalWrite(SortIN3, LOW);
        digitalWrite(SortIN4, HIGH);
        break;
    case 1:
        digitalWrite(SortIN1, LOW);
        digitalWrite(SortIN2, LOW);
        digitalWrite(SortIN3, HIGH);
        digitalWrite(SortIN4, LOW);
        break;
    case 2:
        digitalWrite(SortIN1, LOW);
        digitalWrite(SortIN2, HIGH);
        digitalWrite(SortIN3, LOW);
        digitalWrite(SortIN4, LOW);
        break;
    case 3:
        digitalWrite(SortIN1, HIGH);
        digitalWrite(SortIN2, LOW);
        digitalWrite(SortIN3, LOW);
        digitalWrite(SortIN4, LOW);
        break;
    }
}
else if(rotationController == 0)
{
    digitalWrite(SortIN1, LOW);
    digitalWrite(SortIN2, LOW);
    digitalWrite(SortIN3, LOW);
    digitalWrite(SortIN4, LOW);
}
```




```
}
currentStep_box = (++currentStep_box < 4) ? currentStep_box : 0;

//2000 microseconds, or 2 milliseconds seems to be
//about the shortest delay that is usable. Anything
//lower and the motor starts to freeze.
//delayMicroseconds(2250);
delay(2);
}
/////////////////////////////////////////////////////////////////
////////// SORTING BOX FUNCTION //////////
void LED_driver(int color_select)
{
  if(color_select == 0)
  {
    digitalWrite(yellow, LOW);
    digitalWrite(blue, LOW);
    digitalWrite(green, LOW);
    digitalWrite(red, LOW);
    digitalWrite(white, LOW);
  }
  else if(color_select == 1)
  {
    digitalWrite(yellow, HIGH);
    digitalWrite(blue, LOW);
    digitalWrite(green, LOW);
    digitalWrite(red, LOW);
    digitalWrite(white, LOW);
  }
  else if(color_select == 2)
  {
    digitalWrite(yellow, LOW);
    digitalWrite(blue, HIGH);
    digitalWrite(green, LOW);
    digitalWrite(red, LOW);
    digitalWrite(white, LOW);
  }
  else if(color_select == 3)
  {
    digitalWrite(yellow, LOW);
    digitalWrite(blue, LOW);
    digitalWrite(green, HIGH);
    digitalWrite(red, LOW);
  }
}
```



```
    digitalWrite(white, LOW);
  }
  else if(color_select == 4)
  {
    digitalWrite(yellow, LOW);
    digitalWrite(blue, LOW);
    digitalWrite(green, LOW);
    digitalWrite(red, HIGH);
    digitalWrite(white, LOW);
  }
  else if(color_select == 5)
  {
    digitalWrite(yellow, LOW);
    digitalWrite(blue, LOW);
    digitalWrite(green, LOW);
    digitalWrite(red, LOW);
    digitalWrite(white, HIGH);
  }
}

Elevator Test
int bluePin = 22; //23
int pinkPin = 24; //25
int yellowPin =26; //27
int orangePin = 28; //29

//Keeps track of the current step.
//We'll use a zero based index.
int currentStep = 0;
bool rotationController = false;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  pinMode(bluePin, OUTPUT);
  pinMode(pinkPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(orangePin,OUTPUT);

  digitalWrite(bluePin, LOW);
  digitalWrite(pinkPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(orangePin, LOW);
}
```



```
void loop() {

    //Comment out the Serial prints to speed things up
    //Serial.print("Step: ");
    //Serial.println(currentStep);
    if (rotationController == true)
    {
        switch(currentStep){
            case 0:
                digitalWrite(bluePin, HIGH);
                digitalWrite(pinkPin, LOW);
                digitalWrite(yellowPin, LOW);
                digitalWrite(orangePin, LOW);
                break;
            case 1:
                digitalWrite(bluePin, LOW);
                digitalWrite(pinkPin, HIGH);
                digitalWrite(yellowPin, LOW);
                digitalWrite(orangePin, LOW);
                break;
            case 2:
                digitalWrite(bluePin, LOW);
                digitalWrite(pinkPin, LOW);
                digitalWrite(yellowPin, HIGH);
                digitalWrite(orangePin, LOW);
                break;
            case 3:
                digitalWrite(bluePin, LOW);
                digitalWrite(pinkPin, LOW);
                digitalWrite(yellowPin, LOW);
                digitalWrite(orangePin, HIGH);
                break;
        }
    }
    else
    {
        switch(currentStep){
            case 0:
                digitalWrite(bluePin, LOW);
                digitalWrite(pinkPin, LOW);
                digitalWrite(yellowPin, LOW);
                digitalWrite(orangePin, HIGH);
                break;
            case 1:
```



```
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, HIGH);
digitalWrite(orangePin, LOW);
break;
case 2:
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, HIGH);
digitalWrite(yellowPin, LOW);
digitalWrite(orangePin, LOW);
break;
case 3:
digitalWrite(bluePin, HIGH);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
digitalWrite(orangePin, LOW);
break;
}
}
currentStep = (++currentStep < 4) ? currentStep : 0;

//2000 microseconds, or 2 milliseconds seems to be
//about the shortest delay that is usable. Anything
//lower and the motor starts to freeze.
//delayMicroseconds(2250);
delay(2);
}

    Sort Test
#include <SharpIR.h>
#include <PIDLoop.h>

/*
sig 1 = red object
sig 2 = yellow object
sig 3 = green object
sig 5 = blue object
*/

// limits how fast to travel forward
#define MAX_VELOCITY 100
```

```

// bool kickstart = true;

// left brush pins
const int ENC = 8;
const int IN5 = 9;
const int IN6 = 10;

// right brush pins
const int END = 11;
const int IN7 = 12;
const int IN8 = 13;

// Elevator pins
int bluePin = 22; // IN1 on the ULN2003 Board, BLUE end of the Blue/Yellow motor
coil
int pinkPin = 24; // IN2 on the ULN2003 Board, PINK end of the Pink/Orange motor
coil
int yellowPin = 26; // IN3 on the ULN2003 Board, YELLOW end of the Blue/Yellow
motor coil
int whitePin = 28; // IN4 on the ULN2003 Board, ORANGE end of the Pink/Orange
motor coil

// Sort box pins
int SortIN1 = 23;
int SortIN2 = 25;
int SortIN3 = 27;
int SortIN4 = 29;

// LED pins
#define yellow 30
#define blue 32
#define green 34
#define red 36
#define white 38

// PI FOR DRIVE MOTOR
#define PI 3.1415926535897932384626433832795
int currentStep_elevator = 0;
int currentStep_box = 0;
unsigned long StartTime = 0; // track orbital time
int brushDirection=0;
int brushSpeed=0;
int box_current_position = 0;

```

```

int current_position = 0;
int targetPosition = 0;
int targetBoxPosition = 0;
int targetColor = 0;
int oldColor = 1;
bool elevatorRaisePT1 = false;
bool debrisSorted = false;
bool debrisSortedFull = false;

SharpIR FrontSensor(SharpIR::GP2Y0A41SK0F, A1); // set model of sensor and output
pin (A0)
SharpIR SideSensor(SharpIR::GP2Y0A41SK0F, A0); // set model of sensor and output
pin (A1)
int DistanceFront = 0; // distance dectection FrontSensor
int DistanceSide = 0; // distance dectection SideSensor

bool LeaveBase = false;
bool EndOrbit = false;
bool Return2Base = false;

void setup()
{
    // left brush pins
    pinMode(IN5, OUTPUT);
    pinMode(IN6, OUTPUT);
    pinMode(ENC, OUTPUT);

    // right brush pins
    pinMode(IN7, OUTPUT);
    pinMode(IN8, OUTPUT);
    pinMode(END, OUTPUT);

    // LED pins
    pinMode(yellow, OUTPUT);
    pinMode(blue, OUTPUT);
    pinMode(green, OUTPUT);
    pinMode(red, OUTPUT);
    pinMode(white, OUTPUT);

    // elevator pin
    pinMode(bluePin, OUTPUT);
    pinMode(pinkPin, OUTPUT);
    pinMode(yellowPin, OUTPUT);
    pinMode(whitePin,OUTPUT);

```



```
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, LOW);
// sorting box
pinMode(SortIN1, OUTPUT);
pinMode(SortIN2, OUTPUT);
pinMode(SortIN3, OUTPUT);
pinMode(SortIN4,OUTPUT);

digitalWrite(SortIN1, LOW);
digitalWrite(SortIN2, LOW);
digitalWrite(SortIN3, LOW);
digitalWrite(SortIN4, LOW);

Serial.begin(9600);
}

void loop()
{
  DistanceFront = FrontSensor.getDistance(); // reads the value of front sensor (in cm)
  DistanceSide = SideSensor.getDistance(); // reads the value of Side sensor (in cm)
  ///////////////////////////////////
  ///////////////////////////////////
  targetColor = 1;
  ///////////////////////////////////
  ///////////////////////////////////
  if ((elevatorRaisePT1 == false)&&(debrisSortedFull == false))
  {
    elevator_smart_driver(2600);// max: 7550
    if (current_position == 2200)
      elevatorRaisePT1 = true;
  }
  // what color to sort //
  // red = 1
  // blue = 2
  // green = 3
  // yellow= 4
  targetColor = 1;
  targetBoxPosition = targetColor*4600;
  if ((elevatorRaisePT1 == true)&&(debrisSortedFull == false))
  {
    if (box_current_position < targetBoxPosition)// one rotation takes 4600
```



```
    box_driver(1); // rotate sorting box ccw forever
else if (box_current_position == targetBoxPosition)
{
    box_driver(0);
    // raise debris into chamber //
    if (current_position < 7550)
    {
        elevator_smart_driver(7550); // max: 7550
    }
    else if (current_position == 7550)
    {
        debrisSorted = true;
    }
}
}
// put box in hold position and lower elevator
if (debrisSorted == true)
{

    /// hold debris in chamber ///
    if (box_current_position != targetBoxPosition + 2300) // one rotation takes 4600
    {
        LED_driver((4));
        box_driver(1); // rotate sorting box ccw
    }
    else
    {
        box_driver(0);
    }
    /// lower elevator ///
    if (current_position > 1)
    {
        elevator_smart_driver(0); // max: 7550
    }
    else
    {
        debrisSortedFull = true;
    }
}

}

int LEDSelector = (current_position/1000);
LED_driver((LEDSelector));
}
```




```
//////////////////// FUNCTIONS //////////////////////////////////////
//////// BRUSH MOTOR FUNCTION //////////
// ALLOWS THE USER TO CONTROL THE OF BRUSHES
// DIRECTION AND SPEED.
// INPUT | DESCRIPTION
// -----
// brush_speed | THE SPEED THE ROBOT IS TRAVELING FROM STOP [0%] TO
MAX [100%]
// brush_direction | 2 = FORWARD || 1 = BACKWARD || 0 = STOP
void brush_driver (int brush_speed,int brush_direction)
{
  int Speed = brush_speed*2;
  if(brush_direction == 2)
  {
    Motor3_Forward(Speed);
    Motor4_Backward(Speed);
  }
  if(brush_direction == 1)
  {
    Motor3_Backward(Speed);
    Motor4_Forward(Speed);
  } else
  {
    Motor3_Brake();
    Motor4_Brake();
  }
}
void Motor3_Forward (int Speed)
{digitalWrite(IN5,HIGH);
digitalWrite(IN6,LOW);
analogWrite(ENC,Speed);
}
void Motor3_Backward (int Speed)
{
  digitalWrite(IN5,LOW);
  digitalWrite(IN6,HIGH);
  analogWrite(ENC,Speed);
}
void Motor3_Brake ()
{
  digitalWrite(IN5,LOW);
  digitalWrite(IN6,LOW);
}
void Motor4_Forward (int Speed)
```



```
{
  digitalWrite(IN7,HIGH);
  digitalWrite(IN8,LOW);
  analogWrite(END,Speed);
}
void Motor4_Backward (int Speed)
{
  digitalWrite(IN7,LOW);
  digitalWrite(IN8,HIGH);
  analogWrite(END,Speed);
}
void Motor4_Brake()
{
  digitalWrite(IN7,LOW);
  digitalWrite(IN8,LOW);
}
////////////////////////////////////
////////// ELEVATOR CONTROLLER FUNCTION //////////
void elevator_smart_driver(int target_position)
{
  //drive elevetor to wanted position
  if (target_position < current_position)
  {
    if(current_position == 0)
    {
      elevator_driver(0);
    }
    else
    {
      current_position = current_position - 1;
      elevator_driver(2);
    }
  }
  else if (target_position > current_position)
  {
    if(current_position == 7550)
    {
      elevator_driver(0);
    }
    else
    {
      current_position++;
      elevator_driver(1);
    }
  }
}
```



```
}
else if (target_position == current_position)
{
    elevator_driver(0);
}
}
////////////////////////////////////
////////// ELEVATOR FUNCTION //////////
void elevator_driver(int rotationController)
{
if (rotationController == 1)
{
    switch(currentStep_elevator){
    case 0:
        digitalWrite(bluePin, HIGH);
        digitalWrite(pinkPin, LOW);
        digitalWrite(yellowPin, LOW);
        digitalWrite(whitePin, LOW);
        break;
    case 1:
        digitalWrite(bluePin, LOW);
        digitalWrite(pinkPin, HIGH);
        digitalWrite(yellowPin, LOW);
        digitalWrite(whitePin, LOW);
        break;
    case 2:
        digitalWrite(bluePin, LOW);
        digitalWrite(pinkPin, LOW);
        digitalWrite(yellowPin, HIGH);
        digitalWrite(whitePin, LOW);
        break;
    case 3:
        digitalWrite(bluePin, LOW);
        digitalWrite(pinkPin, LOW);
        digitalWrite(yellowPin, LOW);
        digitalWrite(whitePin, HIGH);
        break;
    }
}
else if(rotationController == 2)
{
    switch(currentStep_elevator){
    case 0:
        digitalWrite(bluePin, LOW);
```



```
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, HIGH);
break;
case 1:
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, HIGH);
digitalWrite(whitePin, LOW);
break;
case 2:
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, HIGH);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, LOW);
break;
case 3:
digitalWrite(bluePin, HIGH);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, LOW);
break;
}
}
else if(rotationController == 0)
{
digitalWrite(bluePin, LOW);
digitalWrite(pinkPin, LOW);
digitalWrite(yellowPin, LOW);
digitalWrite(whitePin, LOW);
}
currentStep_elevator = (++currentStep_elevator < 4) ? currentStep_elevator : 0;

//2000 microseconds, or 2 milliseconds seems to be
//about the shortest delay that is usable. Anything
//lower and the motor starts to freeze.
//delayMicroseconds(2250);
delay(2);
}
////////////////////////////////////
///////// SORTING BOX FUNCTION //////////
void box_driver(int rotationController)
{
```



```
if (rotationController == 1)
{
  switch(currentStep_box){
    case 0:
      digitalWrite(SortIN1, HIGH);
      digitalWrite(SortIN2, LOW);
      digitalWrite(SortIN3, LOW);
      digitalWrite(SortIN4, LOW);
      box_current_position++;
      break;
    case 1:
      digitalWrite(SortIN1, LOW);
      digitalWrite(SortIN2, HIGH);
      digitalWrite(SortIN3, LOW);
      digitalWrite(SortIN4, LOW);
      box_current_position++;
      break;
    case 2:
      digitalWrite(SortIN1, LOW);
      digitalWrite(SortIN2, LOW);
      digitalWrite(SortIN3, HIGH);
      digitalWrite(SortIN4, LOW);
      box_current_position++;
      break;
    case 3:
      digitalWrite(SortIN1, LOW);
      digitalWrite(SortIN2, LOW);
      digitalWrite(SortIN3, LOW);
      digitalWrite(SortIN4, HIGH);
      box_current_position++;
      break;
  }
}
else if(rotationController == 2)
{
  switch(currentStep_box){
    case 0:
      digitalWrite(SortIN1, LOW);
      digitalWrite(SortIN2, LOW);
      digitalWrite(SortIN3, LOW);
      digitalWrite(SortIN4, HIGH);
      box_current_position = box_current_position -1;
      break;
    case 1:
```



```
digitalWrite(SortIN1, LOW);
digitalWrite(SortIN2, LOW);
digitalWrite(SortIN3, HIGH);
digitalWrite(SortIN4, LOW);
box_current_position = box_current_position -1;
break;
case 2:
digitalWrite(SortIN1, LOW);
digitalWrite(SortIN2, HIGH);
digitalWrite(SortIN3, LOW);
digitalWrite(SortIN4, LOW);
box_current_position = box_current_position -1;
break;
case 3:
digitalWrite(SortIN1, HIGH);
digitalWrite(SortIN2, LOW);
digitalWrite(SortIN3, LOW);
digitalWrite(SortIN4, LOW);
box_current_position = box_current_position -1;
break;
}
}
else if(rotationController == 0)
{
digitalWrite(SortIN1, LOW);
digitalWrite(SortIN2, LOW);
digitalWrite(SortIN3, LOW);
digitalWrite(SortIN4, LOW);
}
currentStep_box = (++currentStep_box < 4) ? currentStep_box : 0;

//2000 microseconds, or 2 milliseconds seems to be
//about the shortest delay that is usable. Anything
//lower and the motor starts to freeze.
//delayMicroseconds(2250);
delay(2);
}
////////////////////////////////////
////////// LED FUNCTION //////////
void LED_driver(int color_select)
{
if(color_select == 0)
{
```



```
digitalWrite(yellow, LOW);
digitalWrite(blue, LOW);
digitalWrite(green, LOW);
digitalWrite(red, LOW);
digitalWrite(white, LOW);
}
else if(color_select == 1)
{
digitalWrite(yellow, HIGH);
digitalWrite(blue, LOW);
digitalWrite(green, LOW);
digitalWrite(red, LOW);
digitalWrite(white, LOW);
}
else if(color_select == 2)
{
digitalWrite(yellow, LOW);
digitalWrite(blue, HIGH);
digitalWrite(green, LOW);
digitalWrite(red, LOW);
digitalWrite(white, LOW);
}
else if(color_select == 3)
{
digitalWrite(yellow, LOW);
digitalWrite(blue, LOW);
digitalWrite(green, HIGH);
digitalWrite(red, LOW);
digitalWrite(white, LOW);
}
else if(color_select == 4)
{
digitalWrite(yellow, LOW);
digitalWrite(blue, LOW);
digitalWrite(green, LOW);
digitalWrite(red, HIGH);
digitalWrite(white, LOW);
}
else if(color_select == 5)
{
digitalWrite(yellow, LOW);
digitalWrite(blue, LOW);
digitalWrite(green, LOW);
digitalWrite(red, LOW);
```



```
    digitalWrite(white, HIGH);
  }
}

    Drive Test
    const int ENA=2;
    const int IN1=3;
    const int IN2=4;
    const int ENB=5;
    const int IN3=6;
    const int IN4=7;
    #define PI 3.1415926535897932384626433832795

    const int driver_speed =70;// 40; 0% to 100%
    const int driver_direction = 1; // 1 = forward || 2 = backward || 0 = stop
    const int angle = 45;//Range from 0 to 90 degrees || 45 is straight || Greater then 45 is
right || less then 45 is left

    void setup()
    {pinMode(IN1,OUTPUT);
pinMode(IN2,OUTPUT);
pinMode(ENA,OUTPUT);
pinMode(IN4,OUTPUT);
pinMode(IN3,OUTPUT);
pinMode(ENB,OUTPUT);

    Serial.begin(9600);
    }

    void loop()
    {

        driver(driver_speed,driver_direction,angle);
        delay(100);
    }

    void driver (int driver_speed,int driver_direction, int angle)
    {
        int Speed;

        if (driver_speed <= 100)
            Speed = driver_speed*2;
        else
            Speed = 200;
    }
}
```




```
double rad=(angle*PI)/180;  
int Speed_left = Speed*cos(rad);  
int Speed_right = Speed*sin(rad);
```

```
if(driver_direction == 2)  
{  
    Motor1_Forward(Speed_left);  
    Motor2_Backward(Speed_right);  
}  
else if(driver_direction == 1)  
{  
    Motor1_Backward(Speed_left);  
    Motor2_Forward(Speed_right);  
} else  
{  
    Motor1_Brake();  
    Motor2_Brake();  
}  
}
```

```
void Motor1_Forward (int Speed)  
{digitalWrite(IN1,HIGH);  
digitalWrite(IN2,LOW);  
analogWrite(ENA,Speed);  
}  
void Motor1_Backward (int Speed)  
{  
    digitalWrite(IN1,LOW);  
    digitalWrite(IN2,HIGH);  
    analogWrite(ENA,Speed);  
}  
void Motor1_Brake ()  
{  
    digitalWrite(IN1,LOW);  
    digitalWrite(IN2,LOW);  
}  
void Motor2_Forward (int Speed)  
{  
    digitalWrite(IN3,LOW);  
    digitalWrite(IN4,HIGH);  
    analogWrite(ENB,Speed);  
}
```



```
void Motor2_Backward (int Speed)
{
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,Speed);
}
void Motor2_Brake()
{
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
}

  Brush Test
  const int ENC=8;// wheel driver: 9
  const int IN5=9;// wheel driver: 8
  const int IN6=10;// wheel driver: 13
  const int END=11;// wheel driver: 9
  const int IN7=12;// wheel driver: 11
  const int IN8=13;// wheel driver: 12

int brush_speed = 100; // 0% to 100%
int brush_direction = 1; // 2 = pull in || 1 = pull out || 0 = stop

void setup()
{pinMode(IN5,OUTPUT);
pinMode(IN6,OUTPUT);
pinMode(ENC,OUTPUT);
pinMode(IN7,OUTPUT);
pinMode(IN8,OUTPUT);
pinMode(END,OUTPUT);}

void loop()
{
  brush_driver(brush_speed,brush_direction);
  delay(100);
}

void brush_driver (int brush_speed,bool brush_direction)
{
  int Speed = brush_speed*2;
  if(brush_direction == 2)
  {
    Motor1_Forward(Speed);
```



```
    Motor2_Backward(Speed);
}
if(brush_direction == 1)
{
    Motor1_Backward(Speed);
    Motor2_Forward(Speed);
} else
{
    Motor1_Brake();
    Motor2_Brake();
}
}
```

```
void Motor1_Forward (int Speed)
{digitalWrite(IN5,HIGH);
digitalWrite(IN6,LOW);
analogWrite(ENC,Speed);
}
void Motor1_Backward (int Speed)
{
    digitalWrite(IN5,LOW);
    digitalWrite(IN6,HIGH);
    analogWrite(ENC,Speed);
}
void Motor1_Brake ()
{
    digitalWrite(IN5,LOW);
    digitalWrite(IN6,LOW);
}
void Motor2_Forward (int Speed)
{
    digitalWrite(IN7,HIGH);
    digitalWrite(IN8,LOW);
    analogWrite(END,Speed);
}
void Motor2_Backward (int Speed)
{
    digitalWrite(IN7,LOW);
    digitalWrite(IN8,HIGH);
    analogWrite(END,Speed);
}
void Motor2_Brake()
{
```



```
digitalWrite(IN7,LOW);
digitalWrite(IN8,LOW);
}
    Flag Test
    int bluePin = 22; //IN1 on the ULN2003 Board, BLUE end of the Blue/Yellow motor
coil
    int pinkPin = 24; //IN2 on the ULN2003 Board, PINK end of the Pink/Orange motor
coil
    int yellowPin =26; //IN3 on the ULN2003 Board, YELLOW end of the Blue/Yellow
motor coil
    int orangePin = 28; //IN4 on the ULN2003 Board, ORANGE end of the Pink/Orange
motor coil

//Keeps track of the current step.
//We'll use a zero based index.
int currentStep = 0;
bool rotationController = false;
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);

    pinMode(bluePin, OUTPUT);
    pinMode(pinkPin, OUTPUT);
    pinMode(yellowPin, OUTPUT);
    pinMode(orangePin,OUTPUT);

    digitalWrite(bluePin, LOW);
    digitalWrite(pinkPin, LOW);
    digitalWrite(yellowPin, LOW);
    digitalWrite(orangePin, LOW);
}

void loop() {
    int timer = millis();
    if (timer <= 1000)
    flag_driver(2);
    else
    {
        flag_driver(0);
    }
}

void flag_driver(int rotationController)
{
```



```
if (rotationController == 1)
{
  switch(currentStep){
    case 0:
      digitalWrite(bluePin, HIGH);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, LOW);
      break;
    case 1:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, HIGH);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, LOW);
      break;
    case 2:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, HIGH);
      digitalWrite(orangePin, LOW);
      break;
    case 3:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, HIGH);
      break;
  }
}
else if (rotationController == 2)
{
  switch(currentStep){
    case 0:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, HIGH);
      break;
    case 1:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, HIGH);
      digitalWrite(orangePin, LOW);
      break;
  }
}
```



```
case 2:
  digitalWrite(bluePin, LOW);
  digitalWrite(pinkPin, HIGH);
  digitalWrite(yellowPin, LOW);
  digitalWrite(orangePin, LOW);
  break;
case 3:
  digitalWrite(bluePin, HIGH);
  digitalWrite(pinkPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(orangePin, LOW);
  break;
}
}
else if (rotationController == 0)
{

  digitalWrite(bluePin, LOW);
  digitalWrite(pinkPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(orangePin, LOW);
}
currentStep = (++currentStep < 4) ? currentStep : 0;

//2000 microseconds, or 2 milliseconds seems to be
//about the shortest delay that is usable. Anything
//lower and the motor starts to freeze.
//delayMicroseconds(2250);
delay(2);
}

  IR Sensor Test
  #include <SharpIR.h>
  #define side_IR A1
  #define front_IR A0

  int FIR = 0;
  int SIR = 0;

  SharpIR FrontSensor(SharpIR::GP2Y0A41SK0F, A1); // set model of sensor and output
pin (A0)
  SharpIR SideSensor(SharpIR::GP2Y0A41SK0F, A0); // set model of sensor and output
pin (A1)
  void setup() {
    pinMode(A0, INPUT);
```



```
pinMode(A1, INPUT);
Serial.begin(9600);
}

void loop() {
  FIR = FrontSensor.getDistance(); // reads the value of front sensor (in cm)
  SIR = SideSensor.getDistance(); // reads the value of Side sensor (in cm)

  Serial.print("SIDE IR: ");
  Serial.print(SIR);

  Serial.print("\n");

  Serial.print("FRONT IR: ");
  Serial.print(FIR);

  Serial.print("\n");
  delay(1000);

}

  LED Color Test
  #define yellow 30
  #define blue 32
  #define green 34
  #define red 36
  #define white 38

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(yellow, OUTPUT);
  pinMode(blue, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(white, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  //////////////////////////////////// YELLOW ////////////////////////////////////
  digitalWrite(yellow, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(yellow, LOW); // turn the LED off by making the voltage LOW
```

```

delay(1000);           // wait for a second
////////////////////// BLUE ////////////////////////////////////////
digitalWrite(blue, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000);           // wait for a second
digitalWrite(blue, LOW); // turn the LED off by making the voltage LOW
delay(1000);
////////////////////// GREEN ////////////////////////////////////////
digitalWrite(green, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000);           // wait for a second
digitalWrite(green, LOW); // turn the LED off by making the voltage LOW
delay(1000);
////////////////////// RED ////////////////////////////////////////
digitalWrite(red, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000);           // wait for a second
digitalWrite(red, LOW); // turn the LED off by making the voltage LOW
delay(1000);
////////////////////// WHITE ////////////////////////////////////////
digitalWrite(white, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000);           // wait for a second
digitalWrite(white, LOW); // turn the LED off by making the voltage LOW
delay(1000);
}

```

Main Code

```

////////////////////// LIBRARY'S ////////////////////////////////////////

//////////////////////////////////////
////////////////////////////////////// PIN NAMES ////////////////////////////////////////
//// IR SENSORS ASSIGNMENT //////////
#define side_IR A1
#define front_IR A0
//// DRIVE MOTOR ASSIGNMENT //////////
#define ENA 2
#define IN1 3
#define IN2 4
#define ENB 5
#define IN3 6
#define IN4 7
//// BRUSH MOTOR ASSIGNMENT //////////
#define ENC 8// wheel driver: 8
#define IN5 9// wheel driver: 13
#define IN6 10// wheel driver: 9
#define END 11// wheel driver: 11
#define IN7 12// wheel driver: 12
#define IN8 13// wheel driver: 10

```



```

//////// LED ASSIGNMENT //////////
#define yellow 30
#define blue 32
#define green 34
#define red 36
#define white 38
// Elevator pins
int bluePin = 22; // IN1 on the ULN2003 Board, BLUE end of the Blue/Yellow motor
coil
int pinkPin = 24; // IN2 on the ULN2003 Board, PINK end of the Pink/Orange motor
coil
int yellowPin = 26; // IN3 on the ULN2003 Board, YELLOW end of the Blue/Yellow
motor coil
int whitePin = 28; // IN4 on the ULN2003 Board, ORANGE end of the Pink/Orange
motor coil

// Sort box pins
int SortIN1 = 23;
int SortIN2 = 25;
int SortIN3 = 27;
int SortIN4 = 29;
////////////////////////////////////
//////////////////////////////////// GLOBAL VARIABLES //////////////////////////////////////
//////////////////////////////////// PI FOR DRIVE MOTOR //////////////////////////////////////
#define PI 3.1415926535897932384626433832795
//////////////////////////////////// DRIVE MOTOR VARIABLES //////////////////////////////////////
int driver_speed = 50; // 40; 0% to 100%
int driver_direction = 2; // 1 = forward || 2 = backward || 0 = stop
int angle = 0; // Range from 0 to 90 degrees || 45 is straight || Greater than 45 is left || less
then 45 is right
//////////////////////////////////// BRUSH MOTOR VARIABLES //////////////////////////////////////
int brush_speed = 35; // 0% to 100%
int brush_direction = 0; // 2 = pull in || 1 = pull out || 0 = stop
//////////////////////////////////// IR SENSOR PHASE 2 BUFFER //////////////////////////////////////
// RANGE OF IR BEFORE HAVE TO AVOID CENTER
const int SideIRMinP2 = 279;
const int SideIRMaxP2 = 230;
// TIME WHEN ROBOT SHOULD EXIT PHASE 2 TO PHASE 3
const unsigned long phase2ToPhase3Timer = 120000; // AFTER 2M INS GO TO PHASE
3
//// informs robot that a phase has been completed //////////
bool phase1Complete = false;
bool phase2Complete = false;
bool phase3Complete = false;

```

//

```

void setup()
{
  /// IR SENSORS PIN SETUP ///
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  /// DRIVE MOTOR PIN SETUP ///
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(ENA,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(ENB,OUTPUT);
  /// BRUSH MOTOR PIN SETUP ///
  pinMode(IN5,OUTPUT);
  pinMode(IN6,OUTPUT);
  pinMode(ENC,OUTPUT);
  pinMode(IN7,OUTPUT);
  pinMode(IN8,OUTPUT);
  pinMode(END,OUTPUT);
  /// LED PIN SETUP ///
  pinMode(yellow, OUTPUT);
  pinMode(blue, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(white, OUTPUT);
}

void loop() {
  /// LOOP VARIABLES ///
  /// clock ///
  unsigned long timer = millis();
  /// IR SENSORS VARIABLES ///
  int SIR = analogRead(A1);// SIDE IR SENSOR VALUE
  int FIR = analogRead(A0);// FRONT IR SENSOR VALUE
  ///
  /// PHASE 1 ///
  // IN THIS PHASE THE ROBOT WILL EXIT FROM THE HOME CORNER,
  RECONGIZE WHAT
  // COLOR ITS HOME BASE IS, AND ENTER ZONE 2 AT A ANGLE.
  //-----
  // TURN ON YELLOW LIGHT TO INFORM EXITING HOME

```

```

if (phase1Complete == false)
{
  digitalWrite(yellow, HIGH);
  // EXIT OUT OF HOME BASE AT A ANGLE
  driver_speed = 100;
  driver_direction = 2;
  angle = 45;
  driver(driver_speed,driver_direction,angle);
  delay(10);
  driver_speed = 80;
  angle = 66;
  driver(driver_speed,driver_direction,angle);
  delay(1100);//1200
  driver_speed = 80;
  angle = 38;
  driver(driver_speed,driver_direction,angle);
  delay(100);
  // Phase 1 is complete
  phase1Complete = true;
}
////////////////////////////////////
//////////////////////////////////// PHASE 2 //////////////////////////////////////
// IN THIS PHASE THE ROBOT WILL PERFORM COUNTER-CLOCKWISE
ORBITS AND
// GATHER SPACE DEBRIS UNTIL THE ROUND HAS LASTED FOR 2 MINUTES.
THE ROBOT
// MUST BE PREPARED TO AVOID ONCOMING ROBOTS AND OBSTCALES.
if (phase1Complete == true && phase2Complete == false)
{
  digitalWrite(yellow, LOW);
  digitalWrite(green, HIGH);

  if (SIR < SideIRMinP2 && SIR > SideIRMaxP2) // IN PROPER ORBIT
  {
    driver_speed = 60;//work on me!!!
    driver_direction = 2;
    angle = 75;//work on me!!!
    driver(driver_speed,driver_direction,angle);
  }
  else if (SIR >= SideIRMinP2) // TOO CLOSE TO CENTER MASS, MOVE ROBOT
  AWAY FROM CENTER
  {
    driver_speed = 56;//work on me!!!
    driver_direction = 2;

```

```

    angle = 70;//work on me!!!
    driver(driver_speed,driver_direction,angle);
  }
  else if (SIR <= SideIRMaxP2) // GETTING OUT OF ORBIT, BRING ROBOT
CLOSER
  {
    driver_speed = 60;//work on me!!!
    driver_direction = 2;
    angle = 75;//work on me!!!/82
    driver(driver_speed,driver_direction,angle);
  }

  ////////// CHECK IF PHASE 2 IS COMPLETE //////////
  if(timer >= phase2ToPhase3Timer)
  { // AFTER 2 MINS GO TO PHASE 3 //
    phase2Complete = true;
  }
  ////////// WALL CRASH PREVENTION //////////
  if(FIR >= 200)
  {
    driver_speed = 0;
    driver_direction = 0;
  }
}
////////////////////////////////////
//////////////////////////////////// PHASE 3 //////////////////////////////////////
// IN THIS PHASE THE ROBOT WILL RETURN HOME IN THE LAST MINUTE OF
THE
// ROUND AND RAISE A FLAG TO NOTIFY THAT IT HAS COMPLETED THE
ROUND. THE
// ROBOT MUST BE PREPARED TO AVOID OTHER ROBOTS.
if (phase2Complete == true)
{
  digitalWrite(green, LOW);
  digitalWrite(white, HIGH);
  // stop robot
  driver_speed = 0;
  driver_direction = 0;
  angle = 70;
  driver (driver_speed,driver_direction,angle);
  delay(1000000);
}
//////////////////////////////////// END OF MAIN LOOP //////////////////////////////////////
}

```



```

//////////////////// FUNCTIONS //////////////////////////////////////
////////// DRIVE MOTOR FUNCTION //////////
// ALLOWS THE USER TO CONTROL THE SPEED,
// DIRECTION, AND/OR IF IN REVERSE.
//   INPUT   | DESCRIPTION
// -----
// driver_speed | THE SPEED THE ROBOT IS TRAVELING FROM STOP [0%] TO
MAX [100%]
// driver_direction | 1 = FORWARD || 2 = BACKWARD || 0 = STOP
//   angle   | RANGE FROM 0 TO 90 DEGREES || 45 = STRAIGHT || GREATER
THEN 45
//           | IS RIGHT || LESS THEN 45 DEGREES IS LEFT
void driver (int driver_speed,int driver_direction, int angle)
{
  int Speed;

  if (driver_speed <= 100)
    Speed = driver_speed*2;
  else
    Speed = 200;

  double rad=(angle*PI)/180;// PUT ANGLE INTO RADIAN
  int Speed_left = Speed*cos(rad);
  int Speed_right = Speed*sin(rad);

  if(driver_direction == 2)
  {
    Motor1_Forward(Speed_left);
    Motor2_Backward(Speed_right);
  }
  else if(driver_direction == 1)
  {
    Motor1_Backward(Speed_left);
    Motor2_Forward(Speed_right);
  } else
  {
    Motor1_Brake();
    Motor2_Brake();
  }
}
void Motor1_Forward (int Speed)
{digitalWrite(IN1,HIGH);
digitalWrite(IN2,LOW);

```



```
analogWrite(ENA,Speed);
}
void Motor1_Backward (int Speed)
{
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  analogWrite(ENA,Speed);
}
void Motor1_Brake ()
{
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
}
void Motor2_Forward (int Speed)
{
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,Speed);
}
void Motor2_Backward (int Speed)
{
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  analogWrite(ENB,Speed);
}
void Motor2_Brake()
{
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
}
//////// BRUSH MOTOR FUNCTION //////////
// ALLOWS THE USER TO CONTROL THE OF BRUSHES
// DIRECTION AND SPEED.
//   INPUT   | DESCRIPTION
// -----
//  brush_speed | THE SPEED THE ROBOT IS TRAVELING FROM STOP [0%] TO
MAX [100%]
// brush_direction | 2 = FORWARD || 1 = BACKWARD || 0 = STOP
void brush_driver (int brush_speed,bool brush_direction)
{
  int Speed = brush_speed*2;
  if(brush_direction == 2)
  {
    Motor3_Forward(Speed);
```



```
    Motor4_Backward(Speed);
}
if(brush_direction == 1)
{
    Motor3_Backward(Speed);
    Motor4_Forward(Speed);
} else
{
    Motor3_Brake();
    Motor4_Brake();
}
}
void Motor3_Forward (int Speed)
{digitalWrite(IN5,HIGH);
digitalWrite(IN6,LOW);
analogWrite(ENC,Speed);
}
void Motor3_Backward (int Speed)
{
    digitalWrite(IN5,LOW);
    digitalWrite(IN6,HIGH);
    analogWrite(ENC,Speed);
}
void Motor3_Brake ()
{
    digitalWrite(IN5,LOW);
    digitalWrite(IN6,LOW);
}
void Motor4_Forward (int Speed)
{
    digitalWrite(IN7,HIGH);
    digitalWrite(IN8,LOW);
    analogWrite(END,Speed);
}
void Motor4_Backward (int Speed)
{
    digitalWrite(IN7,LOW);
    digitalWrite(IN8,HIGH);
    analogWrite(END,Speed);
}
void Motor4_Brake()
{
    digitalWrite(IN7,LOW);
    digitalWrite(IN8,LOW);
}
```

}

Appendix G: Spring 2019 Project Plan

Table 40-Spring Planning Chart

Objectives	Team Member	Deliverable
Drive		
Code	Daniel	Motor control implementation
Attach Wheels	Kyle	Configured wheels and turning mechanisms
Wire	Chase, Kyle	Soldered components and connections
Route Test		
Write Algorithm	Daniel	Predetermined route algorithm
Test Algorithm	Daniel	Function testing and accuracy results
Debug	Daniel	Optimized route
Object Detection		
Install Sensors	Fabio	Configured IR/ Pixy Sensor
Write Code	Fabio	Optimized image processing algorithms
Debug	Fabio	Tested algorithms in different environments
Apply Route Clearing + Object Detection		

Write Code	Daniel	
Hardware/software Integration	Daniel	Configured algorithm for focusing on detected objects/steering
Debug	Daniel	Optimized algorithms
Apply Gathering Solution		
Install Parts	Yuan	Configured motors with power and microprocessor
Wire Motors	Yuan	Soldered components
Debug	Yuan	Optimized algorithms
Sorting		
Install storage	Chase	Configure storage unit to allow for storage rotation
Write code	Chase	
Debug	Chase	Test code and optimize for a reliable storage rotation and collection of objects
Evaluate Robot		
Evaluate Target	All team members	Evaluate final design to see if all targets are met
Modify		
Code	Daniel , Fabio	

Structure	Chase, Kyle	Examine structure of final design to determine if design met competition constraints
Finalize		
Gather	Fabio	Verify that final gathering algorithm works
Sort	Chase	Finalize sorting method of robot
Route	Yuan, Kyle	Finalize route for robot to navigate
Drive	Daniel	Verify that robot is able to navigate the field at the desired speed

Appendix H: Bill of Materials

Bill of Materials					
amount	item	Type	-5.14	walmart glue	<i>parts</i>
-12.99	tired wheels	<i>parts</i>	-39.9	2*gearmotors	<i>motors</i>
-59.9	pixy2	<i>electronics</i>	-37.38	4*battery	<i>backup</i>
-17.99	wooden blocks	<i>pf</i>	-18.55	battery pack	<i>electronics</i>
-13.85	pit balls	<i>pf</i>	-40.85	2*voltage regulator	<i>backup</i>
-35.98	DC gearmotor*2	<i>motors</i>	Total Spent		\$663.8
-16.99	H-bridge	<i>electronics</i>	Remainder		\$1336.2
-30.44	aluminum sheet and rod	<i>parts</i>			
-33.98	2*h-bridges	<i>electronics</i>			
-73.9	2*gearmotors	<i>motors</i>			
-39.54	4*amber lights	<i>pf</i>			
-29.27	2*plastic sheets	<i>parts</i>			
-16.99	H-bridge	<i>electronics</i>			
-7.01	H-bridge	<i>backup</i>			
-10.48	Bearings&rubber seals	<i>parts</i>			
-9.97	RGB sensor	<i>electronics</i>			
-5.79	cable	<i>parts</i>			
-12.99	LED	<i>parts</i>			
-18.99	IR sensor	<i>electronics</i>			
-6.78	switch	<i>parts</i>			
-11.95	voltage regulator	<i>electronics</i>			
-36.95	backup gearmotor	<i>backup</i>			
-16.98	2*wheel set	<i>backup</i>			
-2.27	tax	<i>Tax</i>			

Appendix I: Risk Assessment

**Risk Assessment
Safety Plan**

Project information:

Southeast Con 2019 Hardware Competition	02/28/2019
Name of Project	Date of submission

Team Member	Phone Number	e-mail
Fabio Trinidad	(863)604-7645	ft16d@my.fsu.edu
Daniel Delgado	(786)514-7161	Dad15c@my.fsu.edu
Kyle Voycheske	(321)272-0627	Kav14d@my.fsu.edu
Chendong Yuan	(850)345-0582	Cy18o@my.fsu.edu
Chase Sapp	(850)303-9874	ccs16c@my.fsu.edu

Faculty mentor	Phone Number	e-mail
Bruce Harvey	(850)410-6451	bharvery@eng.famu.fsu.edu
Linda DeBrunner	(850)410-6462	linda.debrunner@eng.famu.fsu.edu
Leonard Tung	(850)410-6469	tung@eng.famu.fsu.edu

I. Project description:

This project revolves around a robotics competition, in which a completely autonomous vehicle will be developed for the purpose collecting various colored debris scattered across a predefined playing field. As described in the competition rules, this robot must Be able to clear debris while avoiding other vehicles in the playing field. This robotics competition will utilize a point system to determine the winner of the competition, and therefore most of

II. Describe the steps for your project:

1. Design the robot to meet Southeast Con 2019 Hardware Competitions
2. Cut out frames for the robot's final design
3. Solder wires to electronic components
4. Attach electronic components to robot
5. Program microcontroller
6. Connect electronic components to microcontroller
7. Test and debug robot

III. Given that many accidents result from an unexpected reaction or event, go back through the steps of the project and imagine what could go wrong to make what seems to be a safe and well-regulated process turn into one that could result in an accident. (See examples)

1. While cutting out the frame of the robot, there is a possibility of getting cut by the machine used.
2. Soldering wires to all electronic components introduces the risk of getting burned if the soldering tool is improperly handles.
3. While soldering wires to electronic parts, there is also a possibility of getting shocked.



- 4. Exposing the battery excessive heat or puncturing the package, runs the risk of explosion.
- 5. Accidentally connect the battery terminals causing burns and/or shock.
- 6. Loose hair that is expose can get caught in motors during operation.
- 7. Exposed wires while connecting to an active device, can lead to shock and/or burns.

IV. Perform online research to identify any accidents that have occurred using your materials, equipment or process. State how you could avoid having this hazardous situation arise in your project.

- 1) Handling un-shielded wires that are carrying a voltage can lead to shock.
- 2) Having materials on self that can conduct a current.
- 3) Hair or loose clothing that can get caught in motors.
- 4) Cutting materials toward the user or others.
- 5) Carrying heavy equipment by yourself.
- 6) Alone when someone was injured.
- 7) Rule 410A2: Supply and Communication Systems: Employers shall provide training to all employees who work on or

in the Vicinity of exposed energized lines and pars – “IEEE-SA - Program & Structure - The National Electrical Safety Code® (NESC®).” IEEE-SA - The IEEE Standards Association - Home, standards.ieee.org/products-services/nesc/program.html.

V. For each identified hazard or “what if” situation noted above, describe one or more measures that will be taken to mitigate the hazard. (See examples of engineering controls, administrative controls, special work practices and PPE).

- 1) when working with live wire, have the proper PPE.
- 2) Remover all Jewelry and conductive materials before entering the lab.
- 3) Tie up hair and wear tight but flexible clothing.
- 4) Cut away from self and others
- 5) Work in groups when carrying heavy supplies.
- 6) Make sure to have another person present when working in the lab.
- 7) Turn off all active devices before connecting wire(s) to them.

VI. Rewrite the project steps to include all safety measures taken for each step or combination of steps. Be specific (don’t just state “be careful”).

- 1. Design the robot to meet Southeast Con 2019 Hardware Competitions and safety rules
- 2. Have trained individuals cut out frames for the robot’s final design
- 3. Where PPE when soldering wires to electronic components
- 4. Attach electronic components to robot without having a presence of a live current.
- 5. Program microcontroller while having no unexpected equipment attached.
- 6. Connect electronic components to microcontroller, while the circuit is not live
- 7. Test and debug robot with at least another student is present.

VII. Thinking about the accidents that have occurred or that you have identified as a risk, describe emergency response procedures to use.

- If accident occurs:
- 1) Notify 911
 - 2) Contact faculty
 - 3) Turn off all devices if possible
 - 4) If someone is being shocked do not lay hands on them, let trained staff proceed.
 - 5) Stay with situation until told otherwise.

6) Prepare to avoid the situation in the future.

VIII. List emergency response contact information:

- Call 911 for injuries, fires or other emergency situations
- Call your department representative to report a facility concern

Name	Phone Number	Faculty or other COE emergency contact	Phone Number
Chase Sapp	850-303-9879	FSU Police	850-644-1234
Kyle Voycheske	321-272-0627	Emergency	911
Fabio Trinidad	863-604-7645	Dr. Hooker	850-410-6463

IX. Safety review signatures

- Faculty Review update (required for project changes and as specified by faculty mentor)
- Updated safety reviews should occur for the following reasons:
 1. Faculty requires second review by this date:
 2. Faculty requires discussion and possibly a new safety review BEFORE proceeding with step(s)
 3. An accident or unexpected event has occurred (these must be reported to the faculty, who will decide if a new safety review should be performed.
 4. Changes have been made to the project.

Team Member	Date	Faculty mentor	Date
Fabio Trinidad	02/28/2019		
Daniel Delgado	02/28/2019		
Kyle Voycheske	02/28/2019		
Chendong Yuan	02/28/2019		
Chase Sapp	02/28/2019		

Report all accidents and near misses to faculty mentor.

References

- [1] IEEE Future Directions, sites.ieee.org/southeastcon2019/program/student-program/