6-12-2014

VEX robotics

Ho Joon Cha
Santa Clara University

Joshua Del Real
Santa Clara University

Jamie Kalb
Santa Clara University

Thomas Nance
Santa Clara University

Jenny Yang
Santa Clara University

Recommended Citation

Cha, Ho Joon; Del Real, Joshua; Kalb, Jamie; Nance, Thomas; and Yang, Jenny, "VEX robotics" (2014). *Mechanical Engineering Senior Theses.* Paper 17.

SANTA CLARA UNIVERSITY

Department of Mechanical Engineering

Date: June 12, 2014

I HEREBY RECOMMEND THAT THE THESIS PREPARED

UNDER MY SUPERVISION BY

Ho Joon Cha

Joshua Del Real

Jamie Kalb

Thomas Nance

Jenny Yang

ENTITLED

VEX Robotics

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING

M. A. Ayoubi

Thesis Advisor

Department Chair

Santa Clara University

School of Engineering Senior Thesis

VEX Robotics

Submitted By:

Ho Joon Cha

Joshua Del Real

Jamie Kalb

Thomas Nance

Jenny Yang

University Advisor:

Professor Mohammad A. Ayoubi

Submitted on the Date

12 of June, 2014

Submitted in Partial Fulfillment of the Requirements for the

Bachelor of Science

Degree in Mechanical Engineering in the School of Engineering

Santa Clara University, 2014

Santa Clara, California

Abstract

The objective of our project was to design and construct a control system for communication between two autonomous robots, of different size and capabilities, in an environment with numerous obstacles and challenges. These challenges, completed autonomously, involved moving over and under barriers along with the obtainment and transportation of spherical objects. Our project was tied to the VEX Robotics competition environment as we tested our robots by competing at the World Championships. Unfortunately, due to time and budget considerations, we had to scale back our sensor subsystem. This led to a simpler control system than originally intended; however, we were successful at the competition, and built robots that were efficient and economic in design. Our results proved that our robot designs were more agile and maneuverable than our opponents when it came to manipulating environment elements, and that our scaled back autonomous programs were still very effective in disrupting opponent robots' routines and strategies.

Acknowledgments

The VEX Robotics team would like to thank our advisor Dr. M. Ayoubi for his support, Santa Clara University's Engineering Undergraduate Programs, Santa Clara University's Mechanical Engineering Department, SMaRT Education, and NASA Ames Research Center for funding, as well as Dr. D. Riccomini, Dr. C. Kitts, Dr. R. Marks, Dr. D. Fabris, I. Kalb, and G. Chen for their assistance with the project.

Contents

0.1 Abstract i

0.2 Acknowledgments ii

Contents **iii**

List of Figures **vi**

1 Introduction **1**

1.1 Background Information 1

1.2 Literature Reviews 2

1.3 Project Goals 3

2 Project Architecture **5**

2.1 Customer Needs 5

2.2 Physical Sketch with User Scenario 6

2.3 Functional Analysis 7

2.4 Key System Level Issues 9

2.5 Team and Project Management 10

3 Subsystems **12**

3.1 System Level Requirements 12

3.2 Robot Subsystems 13

3.2.1 Drivetrain 14

3.2.2 Structure 16

3.2.3 Electronics 20

3.2.4 Software 20

3.3 Analysis 23

3.4 System Integration Test and Results 25

3.4.1 Design Iterations 26

3.5 Cost Analysis 29

4 Business Plan **31**

4.1 Executive Summary 31

4.2 Introduction 31

4.3 Goals 32

4.4 Product Description 32

4.5 Potential Markets 32
4.6 Competition 33
4.7 Sales Strategies 33
4.8 Pricing 34
4.9 Services 34
4.10 Financial Plan 34

5 Engineering Standards and Realistic Constraints 35

6 Conclusion 38

6.1 Problems Encountered 38
6.2 Competition Results 39
6.3 Improvements 40

Bibliography 41

A Product Design Specification 42

B Experimental Protocol 43

C Gantt Charts 45

D Budget and Sponsors 47

D.1 Budget 47
D.2 Order Invoices 47
D.3 Sponsors 53

E Copy of Questionnaire 54

F Organized Feedback 57

G Competition Strategy 60

G.1 Autonomous Strategies 60
G.2 Teleoperated Strategies 62

H 2D Drawings 63

I 3D Models 68

I.1 Robot 68
I.2 Chassis 70
I.3 Drive Train 71
I.4 Intake 73

J Robot Code 74

J.1 Competition.c 75
J.2 Subsystem_Functions.c 77
J.3 Utilities.c 84
J.4 Constants.c 87

J.5 Autonomous_Selection.c 89
J.6 Autonomous_Modes.c 92

K Test Data **94**

L Senior Design Conference Presentation **96**

List of Figures

2.1 The VEX robot system level sketch. 7
2.2 As the arm raises, its parallel bars approach each other, touching at the top of travel. 8
2.3 A photo of a mechanum wheel. [11] 10

3.1 The VEX robot system level sketch. [13] 12
3.2 The VEX robot system control block diagram. 14
3.3 Six-wheel skid steer drivetrain choosen for the final robots. 15
3.4 Robot chassis and superstructure prior to the addition of the hopper and the intake/outtake mechanism. 16
3.5 Four-bar linkage mechanism in raised state. [3] 17
3.6 Trajectory of designed six-bar linkage mechanism. 18
3.7 Gearing within the towers for the arm mechanism. 19
3.8 Optical encoder and potentiometer used on both robots. [11] 20
3.9 7.2V VEX battery used for both robots. [11] 20
3.10 Flowchart displaying the operation of the code used for autonomous selection. 23
3.11 Free body diagram of forces on the robot's arm while attempting to lift itself. 24
3.12 FEA of robot arm under lifting forces. Results are for total deformation in feet. 24
3.13 FEA analysis of robot arm under lifting forces. Results are for normal stress in pounds per square foot. 24
3.14 A VEX omni wheel. [11] 26
3.15 Skid-steer and holonomic drive trains. 27
3.16 H-shaped holonomic drive train prototype. 27
3.17 The anti-tip wheel when a) stowed and b) deployed. 28
3.18 Added hopper rails to prevent the elements from falling out. 28
3.19 Passive knock-off bars. 29

C.1 Gantt Chart for the VEX Robotics project. 46

E.1 First page of the general questionnaire administered to anonymous individuals and industry professionals. 55
E.2 Second page of the general questionnaire administered to anonymous individuals and industry professionals. 56

F.1 Processed background data of individuals taking survey. 58

F.2 Processed data for the viability of various approaches for multi-robot control system based on survey responses. 59

G.1 Initial brainstormed autonomous strategy. 60
G.2 Iteration of initial brainstormed autonomous strategy. 61
G.3 Final autonomous stragety for robots. 62

H.1 Drawing for the upper arm channel. 64
H.2 Drawing for the middle arm channel with cut. 65
H.3 Drawing for the inside channel of the tower structure with cut. 66
H.4 Drawing for complete robot assembly. 67

I.1 Solidworks rendering of the front side of the robot. 68
I.2 Solidworks rendering of the back side of the robot. 69
I.3 Solidworks rendering of the chassis with the tower structure. 70
I.4 Solidworks rendering of one side of the chassis. 70
I.5 Solidworks rendering of a side view of the drive train. 71
I.6 Isometric Solidworks rendering of the drive train with motors mounted. 71
I.7 Solidworks rendering of the top view of one side of the drive train. 72
I.8 Solidworks rendering of the intake system. 73
I.9 Solidworks rendering of a front view of one half of the intake system. 73

Chapter 1

Introduction

1.1 Background Information

Our team was motivated to design and build a control system to enable communication between two autonomous robots. The two robots, of varying size and capabilities, would have separate functions and goals; however, they needed to be able to complete tasks together. This placed the emphasis of our project on the control system to make sure that there was an optimum synergy and functionality between the two robots. The robots also needed to be able to navigate through environments with numerous obstacles and challenges: moving over and under barriers, obtaining and delivering spherical objects, and avoiding other robots.

Due to the complexity of these tasks, our system would be a simplified and less autonomous version of robots used for real-world application, among them search and rescue, entrapment, and other autonomous needs. Robots used for such situations would typically include the same subsystems that we used in our robots: sensors, arms, manipulators, and drivetrain. Compared to the system that we developed, the quantity and complexity of sensors required for larger scale robots would likely be greater; however, the budget and timeline of this project did not allow our team to develop the sensor system to a great extent.

The construction of the two robots was based on the VEX guidelines and rules for the 2013-2014 VEX Robotics competition game "Toss Up" [14]. The two robots were designed and built to maximize points during competition. Our team used the VEX Robotics Design System, a robotics construction kit, and programmed the robot using RobotC, a specialized C language developed by Carnegie Mellon University. RobotC is an "integrated development environment" used to program and control VEX robots. The software ran on a VEX-EDR CORTEX Microcontroller that used 802.11b/g protocol to communicate commands through a proprietary VEXnet implementation.

For the design of the robots, we built upon years of VEX competition data and championship-winning methods through thorough research of past winning designs and analysis of match videos. The testing and gauging of our robots and system performance was conducted at the 2014 VEX U World Championships. Each match features two colleges/universities from around the world facing off to earn more points in a limited time period. The head-to-head aspect provided for many more variables that we needed to account for in our design as the robot functions with field elements that could be disrupted and possibly hindered by the opposing team. As mentioned, our focus was on the autonomous portion of the competition; therefore, our team attempted to use additional sensors, not included in the VEX kit, in order to support our emphasis on the control system.

The intent of our autonomous control project was to demonstrate potential applications in larger scale, real life situations. Our control system utilized line tracking, optical shaft encoding, and potentiometer sensors in order to collect the data needed to communicate between both robots. Our team also enhanced and perfected our robots with custom electronics and motors. These were necessary in order to interface with the robots' surroundings and create an efficient and effective autonomous control system. The pairing of two robots allowed for different combinations of defense/offense, varying point-scoring strategies, etc. Most importantly, the autonomous programs increased our chances of earning bonus points, which would give us an advantage in all matches. An effective autonomous control system is essential for success at the World Championships.

Our overarching goal for this project was to develop a control system that would allow two autonomous robots to communicate with each other. This project served as a proof-of-concept for larger-scale applications of autonomous robotic systems. We demonstrated the capabilities of our completed robots at the Vex Robotics World Championships in the VEXU college division.

1.2 Literature Reviews

Early research for our project involved reading the VEX game manual for this year's competition and discussing the general nature of the competition. Two members of our team had previous VEX robotics competition experience, which facilitated our early discussions. This early phase involved an examination of the competition rules and layout of the playing field, and of scoring methods and robot designs seen in previous years. To better prepare for the competition,we also consulted with long-time VEX participants.

A major resource for our project is the VEX handbook and appendixes [13] that provide detailed information on game rules, layout, and specific dimension. The competition for this year was called "Toss-Up" and involved two robots that must perform in both an autonomous and a remote controlled period. The two robots must score points by moving balls into the scoring zones on the field, which were divided by a metal bump and a bridge. A series of appendixes were also provided detailing size specifications of game pieces and the playing field.

To help better understand the possibilities of cooperation with multiple robots, the conference paper "Entrapment/Escorting and Patrolling Missions in Multi-Robot Cluster Space Control" by S. Mas [8], proved to be an excellent source. The paper demonstrates a cluster space control method for controlling multiple robots simultaneously. This is accomplished as the robots record data and process it through various independent systems. This method creates a new level of abstracted data referred to as cluster space data. The robots or drivers can then use this data to better control and maneuver the robots. This proves to be highly effective when trying to have multiple robots navigate through various scenarios while avoiding collisions. Using this system of control reduces navigational difficulties that may be experienced while a robot is heading towards its destination.

The article, "Intelligent Route Planning for Fast Autonomous Vehicles Operating in a Large Natural Terrain" by Al-Hasan [1] provided an example of a large-scale controls experiment on robots. The journal article discusses creating robots that can travel over large predefined distances, around 4 square km, and calculate an optimal path from point A to B. The authors do this by weighing the variables of distance, safety, and maneuverability and calculating the best possible path through the area. This process is further optimized through an intelligent learning system onboard the robots that remembers previous attempts at navigation. This capability of "remembering" improves the calculation speed for the robots in determining an appropriate path.

We also discussed "Teamwork by Swarms of All-Terrain Vehicles" by Pettinaro Giovanni [6] which generally explores the concept of swarm robotics. Swarm robotics takes a large part of its inspiration from nature. Since a large independent robot required to perform complex tasks is both expensive and difficult to produce, why not create many small inexpensive robots that operate as one? This system functions more along the lines of an ant colony, where the ant colony as a whole is treated as one large organism. This makes the swarm much more robust and capable of withstanding mechanical and hardware failure in single swarm members without damaging the capabilities of the whole swarm. The applications of this system are numerous, ranging from space exploration to military to search and rescue in cramped conditions. Although the technology is still young, the potential and relatively low cost of swarm robots compared to large singular robots make it the economic choice as the technology continues to advance.

Discussing an application for using sensor systems on robots in different environments, the article by Kazuyoshi Miyazawa titled, "Fire Robots Developed by the Tokyo Fire Department" [10] outlines the current advantages and disadvantages of firefighting robots. Firefighting robots are an excellent addition to public safety since they replace for firemen in situations too dangerous for them, such as chemical fires and enclosed areas with high smoke. Despite their capabilities of reducing risks to humans and entering areas that could prove too dangerous for people, these robots still have shortcomings that must be addressed, such as heat resistance to protect control systems, mobility to address climbing stairs, reliable power sources, reliable signal transmission, and good autonomy. Despite these problems the potential impact of fire-fighting robots is something worth investing time and energy into.

In regards to sensors, the article titled, "Neuromorphic Vision Sensors" by Giacomo Indiveri [7] gives an interesting view on vision processing for robots. Vision in more advanced robots is usually done through the implementation of vision algorithms taking in data from cameras. However, these designs require a great deal of power and processing time, making them less viable in real world applications. A more viable approach would be to emulate insect eyesight, and use multiple sensors working in unison to reduce the volume of incoming data to the information that matters. This would decrease the processing time and still provide the robot with enough visual cues to navigate any hazards in an environment, similar to an insect operating on limited brainpower. This cluster of sensors is referred to as "a very large scale integration technology" or VLSI. Current experiments show great promise for this technology since it provides enough visual data for a robot to function at low power consumption and relatively small sensor size.

1.3 Project Goals

At present, autonomous multi-robotic systems agile enough to maneuver around obstacles with accuracy are rare. While there are numerous autonomous vehicles and robots being used around the world, multi-robot clusters are still in the early developmental phase. There are very few control systems for multi-robotic systems that can communicate to accomplish goals while moving to desired locations.

Our project therefore sought to develop a functioning multi-robot autonomous control system. Prior to any design, an anonymous survey was sent out to over 300 people for their opinions with regards to multi-robot autonomous systems (Appendix E). After a period of two weeks, sixteen responses were recorded, shown in Appendix F. The responses indicated that people believed the search and rescue viability for multi-robot autonomous systems was the highest. As a result, the team decided to pursue a control system that could navigate an obstacle-ridden environment. This led the team to participate in the VEX U robotics competition, providing a perfect platform to implement the control system being designed.

First off, the VEX U competition required the development of two robots, each built to specifications and rules of the VEX U competition. A control system was also required that would enable the robots to cooperate in sensing their environment and each other in order to move around obstacles. Testing of the robotic controls system's effectiveness was carried out in the VEX U robotics competition. Third, although not crucial to the success of the project, winning the VEX U competition was a desired outcome.

We hope that the development of our control system for two autonomous robots can be further expanded to include multiple robot cluster applications. The impact of this control system would not only help in the development of multi-robot clusters similar to the robots in this project, but could also assist in various hazardous missions, such as patrolling and entrapment missions in war zones, or natural disaster-stricken areas.

Chapter 2

Project Architecture

2.1 Customer Needs

Our project's customer can be defined in four main groups: our Engineering Advisor, the VEX Competition, the Military, and Search and Rescue. These groups are individually addressed below.

Engineering Advisor

The primary customer is Dr. Mohammad Ayoubi since his approval is necessary to continue work on our project. Our advisor needs the senior design requirements to be fulfilled and to ensure that our project is focused on a field in Mechanical Engineering, in this case Control Systems and Autonomous Robotics. The goal and motivation of our project specifically is to design and construct a control system and mechanical platform that will command two autonomous robots. By doing so, we are able not only to design autonomous robots that would compete in VEX Robotics but also further develop the use of robotics in real-world scenarios.

VEX Competition

Competition participants can watch and build upon our designs for future use, advancing robotic technologies whilst bringing more public attention to the VEX Robotics competition. The goal of the competition is to promote robotics to the general public and inspire young children to make their creative/innovative dreams come true through the use of robot technology.

Military

Today military forces all across the globe have shown us that one of the largest benefits of using robotics on the battlefield, such as aerial drones, is to remove the soldier physically from battle. These aerial drones are piloted miles away by qualified members of the military to accomplish the same task that soldiers would have to do in person before this specific technology came about. With that said, the military is looking for more ways to limit casualties of men and women in combat. Apart from surveillance, which has been a drone's primary objective, robots can be used on the ground to scan for hostile targets as well as other dangers, directly aiding the soldiers on foot. Robots working together to complete a task can accomplish many battlefield jobs that a combatant normally has to do, such as:

- Lift heavy objects to escape/evade the enemy
- Clear a room or identify targets in a building before military personnel need to enter
- Set up communication networks
- Search for/rescue wounded soldiers in hard to reach or dangerous environments (small, trapped locations)
- Clear heavy debris in disaster scenarios so that humans do not need to be involved
- Do all of these activities autonomously so that the soldier can focus on being a soldier during hostile situations (i.e. Firefight)

Search and Rescue

Search and Rescue is where robots can save time and lives without risking the lives of others. A Search and Rescue organization's primary need is a way of maneuvering through trapped environments while avoiding obstacles or even moving obstacles to get to a victim in a timely manner. They need to do this all while preventing injury or unnecessary accidents that result in fatalities of its members. A few other needs that these organizations have are:

- Assessing possible danger prior to human involvement in a disaster type of situation (i.e. Earthquakes, fires, floods, building collapses, etc...)
- Clearing heavy objects to reach a victim
- Protecting those who risk their lives to save others

We question why these individuals ever need to risk their lives to save others when we can develop methods to remove them harm's way. With the use of autonomous robotics we can accomplish this goal. Machinery is expendable; however, human lives are not. Designing a control system that allows for multiple robots to communicate and successfully master any of these needs is not only doable, but would be a huge advancement in engineering.

2.2 Physical Sketch with User Scenario

The robots we created interact with the environment via their sensors. Sensors, such as the potentiometer and the optical drive encoders, relay information regarding their movement on the playing field to the VEX Cortex microcontroller. The sensors aid in carrying out the autonomous control code that is relayed to the robots. The control code represents a closed-loop feedback control system, which receives data from the sensors and adjusts the input of the robot to achieve the desired output. For instance, if the robot needs to move twelve feet in a straight line, as the robot begins to move, the optical drive encoders relay the distance the robot has traveled back to the controller. From here, the difference (error) between the initial input and actual distance left to travel is calculated and becomes the new input. This loop continues until the output is reached and the robot stops.

Not only will the robots function autonomously but they have a driver-operated mode as well. In this case, the inputs come directly from the remote control and ultimately the driver. Inputs, such as moving forward/backward and lifting the arm system, all deliver their individual message from the remote control to the VEX Cortex, where there is an immediate and direct output. A system level sketch illustrating the control path can be seen below in Figure 2.1.

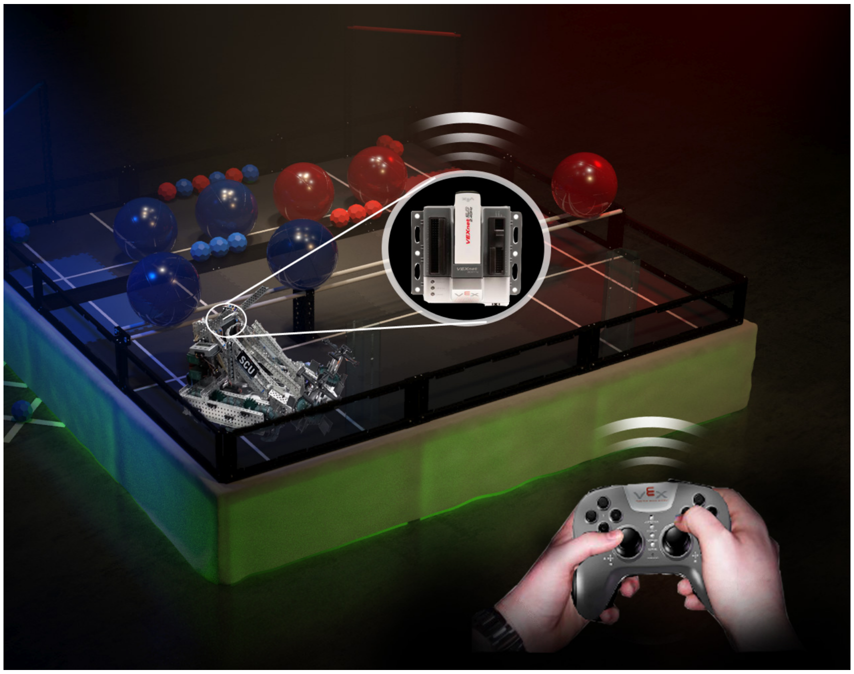


Figure 2.1: The VEX robot system level sketch.

2.3 Functional Analysis

The VEX Robotics Competition involves using two robots to maneuver around an obstacle-ridden environment to pick up and score spherical game elements, known as Bucky Balls, into scoring zones. In order to accomplish this, the robot had to be capable of moving around a 12'x12' field, intake and outtake Bucky Balls, lift the arm mechanism (to score in the tall goals), and operate autonomously. The following subsystems were necessary:

Drive Train

The completed robots were able to move around the field through the use of motors, which drive the wheels in the desired direction. During the teleoperated mode, the input required for the robot to drive came directly from the operator using the controller. The output was velocity and direction of the robot, depending on the input from the controllers. One thing constraining the motion of the robots was the gear ratio, which affected how the robot moved. The gear ratio inside the gearbox was 1.6:1 while the ratio was 1:1 between the motor and the wheels. The drivetrain gear ratio describes a trade off between speed and torque, or pushing power. A high gear ratio results in a higher top speed, but less overall pushing power. Both of these qualities are important to the robot's performance so an acceptable compromise had to be decided upon.

Arm Mechanism

Since the robot needed to score in various scoring zones and goals, including one being two feet above the playing field, it needed to be able to lift the balls to different heights; the lift mechanism designed accomplished this goal. This functionality was absolutely necessary in order to score the maximum number of points in a round.

The operator must be able to easily communicate his or her desired motion to the arm system.The input for the lift mechanism on the controller had been adjusted a few times and finalized as the right joystick. The operator would control the lift height via this joystick (up resulted in an upward motion of the arm and down resulted in a downward motion). The lift mechanism included a hopper to store the Bucky Balls and also had an up or down output motion. The six-bar arm design limited the maximum height because the parallel bars would touch at the maximum height. The height of the arm was controlled in order to avoid destroying the motors. Though the geometry limited the arm's motion, the code would turn off or limit motor power at the arm's extreme limits so as to not burn them out.

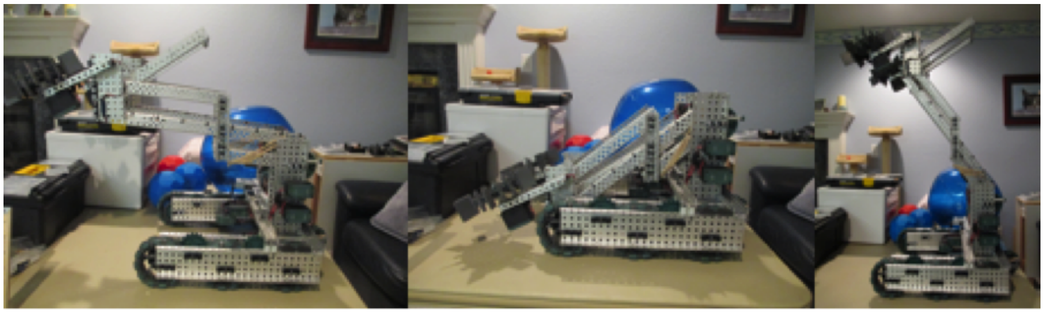


Figure 2.2: As the arm raises, its parallel bars approach each other, touching at the top of travel.

Intake or Outtake Mechanism

Without the ability to intake Bucky Balls it would have been extremely difficult to achieve maximum points in a round. For our robot designs, it was essential that one of the key functions was to intake the balls into a hopper so they could then be transported around the field and dropped off. The operator, using the two right bumper buttons on the controller, directly controlled the motor input for the intake. The output resulted in an angular velocity (spinning the intake wheels inward or outward for outtake) which was applied to the field elements. Essentially the motors would be activated and the motor shafts would rotate, spinning the intake wheels. From a constraints standpoint, there were a few limits on the use of the intake system other than positioning the intake wheels correctly to successfully bring one of the balls into the hopper.

Sense Environment

The ability to use sensors and operate autonomously wasn't required by the competition; however, the autonomous bonus provided a significant scoring advantage. The sensor's primary function was to take readings of either the environment or the robot itself (depends on the sensor); communicate this data to the robot's microcontroller (Cortex); correct the error from the desired output; and then send that information to the actuators. For the competition, this whole system had to be able to operate autonomously for at least one minute each match. To achieve autonomy, the code gave the robot an input, and the output would then be any one of the previous functions of the robot. Uniquely, this particular function of the robot can control all of the other functions that the robot can do during teleoperated mode.

2.4 Key System Level Issues

With any design there will come a point where certain issues arise. For our VEX Robotics project there are six primary subsystems, and each one can present various issues when designing and testing these robotic systems. Our main subsystems are: the Motors, VEX Sensors, Drivetrain, Controllers, Structure, and Coding.

Motors

The competition restricts the motor usage to 12 motors and the design that we implemented uses a limited amount of motors efficiently. Simply, motors add extra weight, and in order for the robots to be more agile and less "clunky", the number of motors had to be taken into careful consideration during the design process. Another issue with the motors is that they also take up space that could otherwise be used by another subsystem. For example, if a motor is located in a position that could potentially be replaced by a key system element such as another sensor then we would need to redesign a subsystem to accommodate both or prioritize our needs.

Sensors

The target robot design is one that can successfully navigate and score points in the competition environment, autonomously. In autonomous mode the sensors are the driving force for all the other subsystems. The sensors detect various field elements and game pieces around them, and send information back to the Cortex, power system, motors, and control systems to be interpreted into new commands which are sent back and translated into either motion in a certain direction or some other reaction based on the design of the robot. The main concern is making sure that the sensors are working properly and accurately in order for the robot to function correctly as a whole.

Drivetrain

There are a few different options for how these robots can move. The main choices that were discussed are the following:

- *Tank Drive*: There are a few things to consider when thinking about using a tank drive system to provide motion to a robot. One of these is the fact that it takes a considerable amount of power to turn a tracked robot because of the frictional forces that must be overcome in order to rotate. This type of steering is also called "skid steering." Tank drive systems usually result in slower movement speed as well, except for the mechanisms seen in today's military; the tanks in the military consist of a very complicated adaptation of the tank drivetrain not feasible for our design. However, the simplest drive system involving "tracking" is the dual drive system. By allocating an individual motor for each track this motion can be implemented. The drawbacks of a dual drive system are that two motors would need to be used just for the drivetrain itself, as well as the difficulty this design has of moving in a straight line since the two tracks move independently of one another.

- *Omnidirectional*: The omnidirectional drivetrains include swerve, holonomic and mecanum drive systems. Overall these types of drives are much more maneuverable than the tank drive. The holonomic drivetrain can move in a diagonal fashion whilst maintaining the direction that the front of the robot is facing, which adds another element into maneuverability. Mecanum drive operates by utilizing wheels composed of offset rollers, as shown below in Figure 2.3.

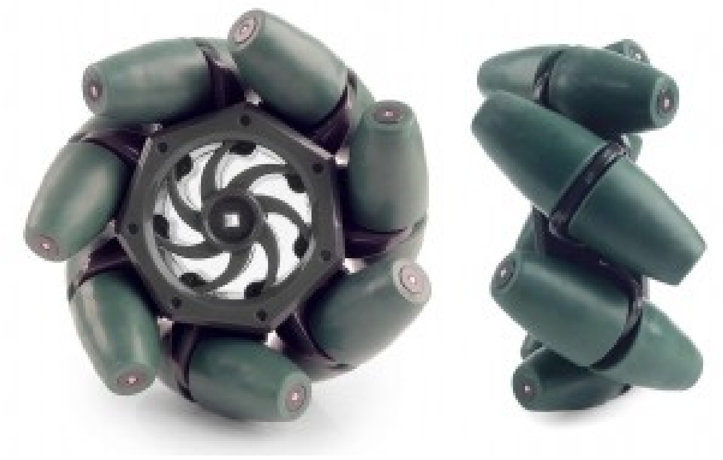


Figure 2.3: A photo of a mechanum wheel. [11]

When the front and back wheels of the drive train are spun in opposite directions, the robot can strafe to the right or left, depending on which way the rollers are made to spin. With the exception of the swerve drive, these drive systems are relatively complex to understand, which makes them complex to implement. The downside to omnidirectional drives is that they are difficult to control and program. Omnidirectional drives also have significantly less traction than tank drives.

Control

The control system is the link between the robot and the human operator's remote control as well as being the control system for the autonomous mode of the robots. One of the big issues that will be encountered when designing the control system is getting the robot-to-human user interface to feel natural and for commands to be carried out in a timely manner. Lag time between operator input and robot output could result in difficulties in control and increased driving errors.

Structure

Because weight is a significant issue for these two robots, we paid special attention to how the mechanical structure was designed. In order to build the most efficient robot, the design was condensed and simplified. Although one could argue that a bulky robot can be used strategically to block or obstruct the opponent during competition, this approach is simply not ideal. Both for the competition and in real-world applications, robots need to be tightly packed and simple. The addition of extraneous structural components also reduces the mounting space for other subsystems within the robot. The primary function of the structure subsystem is to provide a framework to mount the other subsystems as well as to protect them. Therefore, minimizing bulkiness and maximizing protection was the main focus when balancing the structural system to best fit our design.

2.5 Team and Project Management

For our project, there were numerous challenges and constraints to deal with. From budgeting to team management, from the design process to the timeline of our project, everything was carefully considered and dealt with as a group. Since two robots were needed, time and team management, along with budgeting, were critical. Weekly meetings with the team were held to make sure everyone was on the same page; weekly meetings with advisors and professors were also held to ensure that the department was up to date with what the team had accomplished. Email threads were also used to discuss issues when the team was not together, and any problems with group members were addressed directly in person at the beginning of team meetings. The team had one technical lead, Jamie Kalb, and one project lead, Jenny Yang, who were the main contacts who kept the project running. Gantt charts (Appendix B) were created at the start of the project to ensure that the team had a schedule to keep up with, while a detailed budget (Appendix D) was also created for a proper estimate of the cost of our project.

Chapter 3

Subsystems

3.1 System Level Requirements

For our system to be successful, it needed to excel in the VEX U competition. The competition consisted of field elements, rules, and opponent robots. Our focus was on the autonomous portion of the competition, to be achieved through the use of sensors and an accompanying control feedback system. Any sensors were allowed, including: ultrasonic, line tracking, optical shaft encoder, bumper switches, limit switches, and light sensors.

1. The playing field, shown below in Figure 3.1, is littered with different field elements that the robots must communicate and navigate through:

    - 20 Bucky Balls that are 0.25 pounds in weight (10 red, 10 blue)
    - 8 Large Balls that are 0.90 pounds in weight (4 red, 4 blue)
    - 12 inch tall overpasses
    - 2 inch high bumps

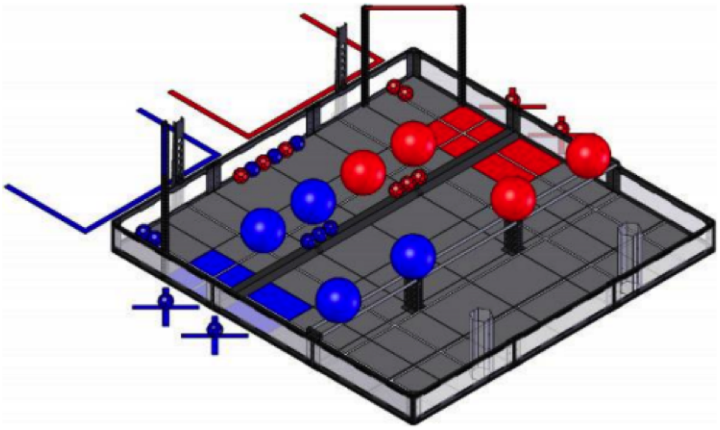


Figure 3.1: The VEX robot system level sketch. [13]

2. A 10 point bonus can be received for outscoring the opposing team during the first 60 seconds of the match, which is completely autonomous. The next 60 seconds consists of a teleoperated, human controlled, mode. A team can gain an advantage in the teleoperated mode by setting up the field in the autonomous mode.

3. A robot can score points by grabbing onto a bar and hoisting itself up in the competition field. The bar is located in the corner of the hanging zone, 40" above the ground. Low hanging is defined as being any distance above the ground short of the field perimeter height, while high hanging is defined as a robot hanging completely above the field perimeter. Scoring achieved by hanging is as follows:

   - 5 points for low hanging without a Large Ball
   - 15 points for low hanging with a Large Ball
   - 10 points for high hanging without a Large Ball
   - 20 points for high hanging with a Large Ball

4. Another method of scoring points involves moving the field elements into zones. The field is divided into three zones by the bumps and overpasses. The balls provide different points based on their size and placement:

   - 0 points for any ball in hanging zone
   - 1 point for any ball in middle zone
   - 2 points for the Bucky Ball in Goal Zone
   - 5 points for the Large Ball in Goal Zone
   - 5 points for scoring a Bucky Ball in a tall goal
   - 10 points for capping a Large Ball on a tall goal

5. An opponent's score can be lowered by tampering with their balls, zones, and robots:

   - Opponent balls can be moved into their hanging zone
   - Balls can be negated when pushed onto a barrier
   - Balls can be placed in contact with a robot to nullify the point value
   - Robots can be pinned for as long as 5 seconds
   - Robots can be blocked from hanging

3.2 Robot Subsystems

The robots developed for the VEX U robotics competition consist of several subsystems. Different subsystems allowed the robots to address different parts of the design challenge, from mobilizing the robots to manipulating game objects to controlling robot actions. Interactions between subsystems are outlined in the control block diagram shown below in Figure 3.1.

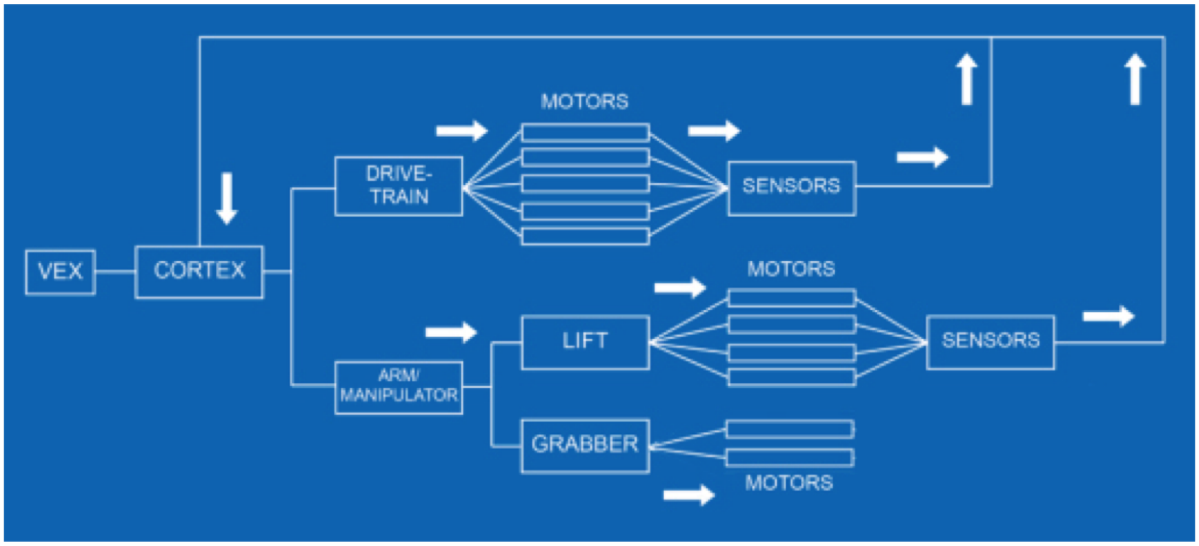


Figure 3.2: The VEX robot system control block diagram.

Dividing the robot into modular subsystems allowed for development in small, easily testable pieces, which were only combined after rigorous testing. In this way, each part of the robot was verified to work individually with a high degree of confidence prior to its integration into the overall system.

The subsystems include the drivetrain, structure, electronics, and software. A description of each follows, including its design requirements, possible options, the chosen solution, and how the chosen solution addresses the requirements.

3.2.1 Drivetrain

The drivetrain subsystem is what makes the robot mobile. It provides locomotion, which makes the robot useful by moving the rest of its mechanisms to where it can interact with objects. Although stationary robots have plenty of applications in manufacturing, this project is targeted towards robots that are inherently mobile, as in search and rescue operations. Unlike manufacturing, search and rescue robots face obstacles and interactions with objects in unknown initial locations.

The drivetrain system has a number of requirements. It must be fast, maneuverable, and able to hold position when forcibly opposed. Since the robot will be operated by a human at times, the drivetrain must be responsive and intuitive to control.

The drivetrain system consists of three major components:

- **Motors**
  The VEX Robotics Competition allows for the use of a set number of motors (12), and therefore has imposed an upper limit on power consumption. Our team must decide how many of these 12 motors to allocate for each subsystem. Our preliminary concepts suggested the use of 4-6 motors in the drivetrain subsystem. This was consistent with our research of previously designed robots under similar constraints [13]. However, when we decided to use a tank drive system, we needed six motors in the entire drivetrain.

- **Gear Train**
  The output speed of the motors (100RPM) needed to power the wheels directly. If not, a gear train

will have to be used to alter the output speed. The main trade off in the drivetrain design is speed versus torque. Our team needed to decide on a gear ratio that provides a sufficient amount of torque to move the robot while maintaining an appropriate speed. Our research led us to implement a 1.6:1 ratio. This ratio was found to provide the robots with enough torque to overcome driving over the bump on the field while also providing the robots with a relatively high maximum speed. Setting the gear ratio lower could have the risked overworking the motors and burning them out.

- **Wheels**
  We have several choices in wheel types, ranging from traction wheels, omni wheels, and mecanum wheels, as well as choices in wheel diameter, from 2.75" to 5". Our research and experience suggests that 4" diameter omni wheels will be effective in conjunction with the aforementioned 1.6:1 gear ratio, but it will be simple to substitute and test other wheel types and sizes. For the competition, we decided to go with the 4" diameter omni wheels for both of our robots since it was determined through experimentation that they possessed the best traction paired with maneuverability.

The drivetrain system that our two robots use is a tank drive system, shown in Figure 3.3. The tank drive system is used because of the benefits that it provides in maneuvering. It takes a considerable amount of power to turn a tracked robot because of the frictional forces that must be overcome in rotation. This rotation movement is called "skid steering". One of the disadvantages to using tank drive is that the robots will have a slower movement speed. Two motors would be needed just for the drivetrain and because this design uses two independently moving tracks, it would be difficult to move the robot in a straight line. The discrepancy was addressed and fixed through the use of code.

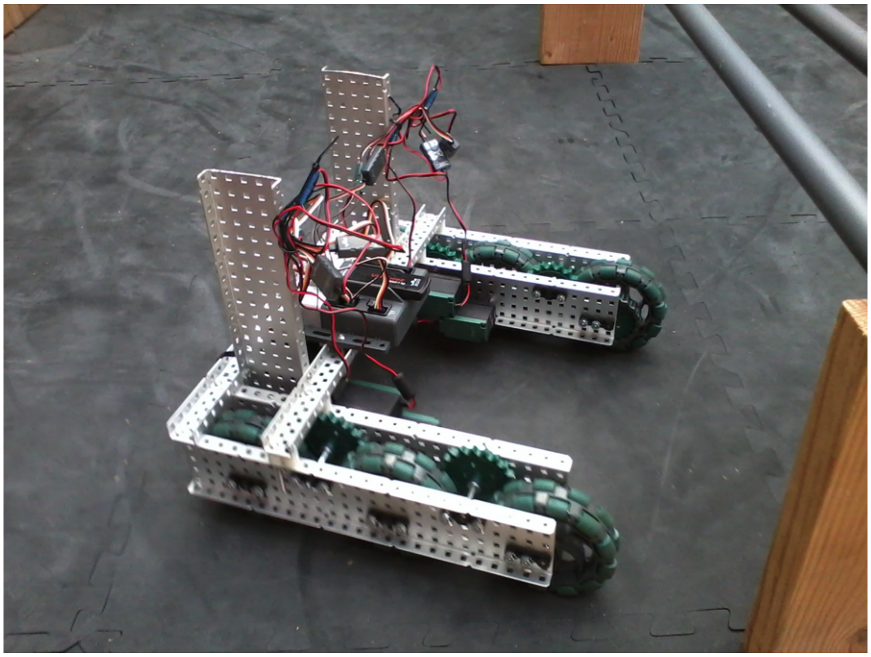


Figure 3.3: Six-wheel skid steer drivetrain choosen for the final robots.

3.2.2 Structure

While the competition does not impose a maximum weight limit, weight can be an issue when designing these two robots to efficiently navigate the field. The primary functions of the structure subsystem is to provide a framework to mount the other subsystems as well as protect them. Minimizing bulkiness and maximizing protection was the main focus when balancing the structural system to best fit our needs. To initially address this, we chose to use aluminum structural elements as opposed to the standard steel, which significantly decreased our initial frame weight.

For the competition, we decided to go with two identical robots, both of which had been designed to be agile and to be offensive, in the sense that both robots operated to score points during matches instead of playing defensively, such as blocking the opposing teams' robots. They were designed to quickly gather game elements and score. The chassis forms the mechanical structure of the drivetrain and the base of the entire robot, as shown in Figure 3.4. Because of its structural importance, the chassis must be very rigid, yet light to reduce the overall weight. Low weight has been shown in our experience to have a major effect on overall robot speed. Our aluminum chassis holds our two tank drivetrains and connects all of our major subsystems. It is attached in the center by a perpendicular bar which then leads to the arm system, sensors, and VEX Cortex.

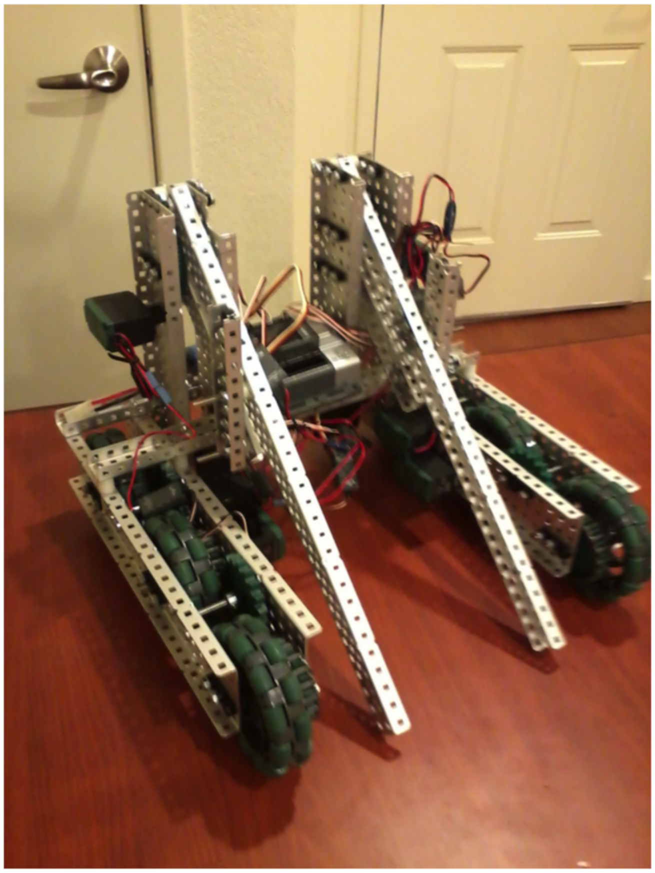


Figure 3.4: Robot chassis and superstructure prior to the addition of the hopper and the intake/outtake mechanism.

Intake

The intake system is what allows our two offensive robots to gather Bucky Balls and score. It is made up of twin intakes attached to the ends of the robots' arms. The intakes have both an upper and a lower manipulator. The bottom manipulators have rubber fins in order to grip and intake the smaller Bucky Balls. The upper manipulator, which has rubber grips as well, is designed to bend out in order to capture and release the larger inflated ball and is extended vertically from the lower manipulator and is attached with rubber stand-offs. This allows the upper manipulator to flex outward when picking up the large balls. If the intake could not flex, it would not be possible to pick up the large balls due to the difference in ball diameters.

Arm

The initial concept for the arm system consisted of two parallel four-bar linkage mechanisms, with the pivot point at the top of the two towers. It was determined that the maximum height that the arm could reach was 25 inches, as seen in Figure 3.5. The purpose of the arm was to score balls in a goal that was 24 inches tall. While the four-bar arm mechanism was able to clear the height of the goal, the addition of the designed intake mechanism caused the end of the robot arm to fall an inch short of the height of the goal. Since all parts used to build the robot came from a given set of VEX parts, a longer linkage could not be used because the four-bar linkage was modeled using the 1x2x1x35 C-Channel, the longest channel piece offered in the VEX catalog.

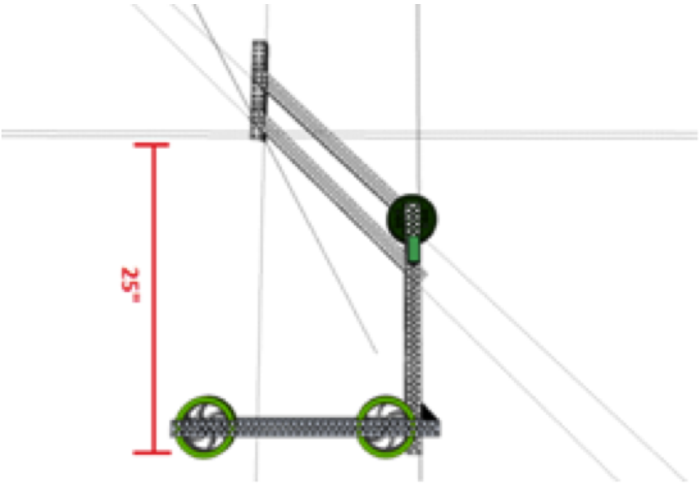


Figure 3.5: Four-bar linkage mechanism in raised state. [3]

To meet the height requirement of the goal, a six-bar mechanism was then designed. The six-bar mechanism consisted of two four-bar mechanisms superimposed on top of each other, trajectory shown below in Figure 3.6. The offset and division of the bars allowed for the system's intake to not only reach the height of the goal, but was also more space efficient when collapsed within the footprint of the chassis. Because the addition of material resulted in a heavier arm and higher torque effect, an extra motor was added to both sides in order to support the design modification.

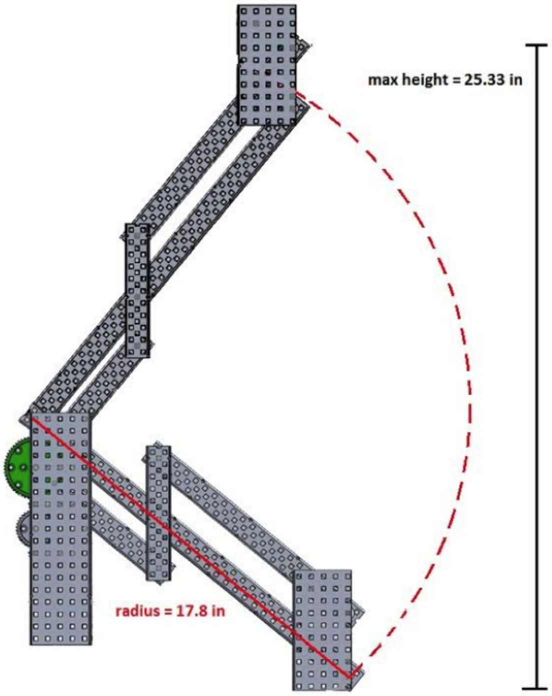


Figure 3.6: Trajectory of designed six-bar linkage mechanism.

One of the problems associated with the addition of a large arm system is the weight distribution and balance of the robot. When the arm was deployed, it was easy for a robot to tip over, especially with the added weight of the game elements. We implemented a number of measures to prevent tipping, such as a lower speed when the arm is in use, or maneuvers which allow the driver to force the robot to push itself upright; however, a robot is essentially out of play when it falls on its back. A tip wheel (see Figure 3.17) was added to the back of the chassis on our robots in order to make sure that tipping would not occur. It would self-deploy when the robot's weight shifts towards its back.

Gearing Calculations

Within the structure subsystem, the arm mechanism included two motors, which drove a large gear, as seen in Figure 3.7, moving the arm up and down. The motor shafts ran through the small 12 tooth gears; one shaft was connected to the green 60 tooth gear while the other was connected to the mid-sized 36 tooth gear. The 60 tooth gear was where the arm was attached to the main structure. It was important to determine the gear ratio to gauge whether the motors can handle the load.

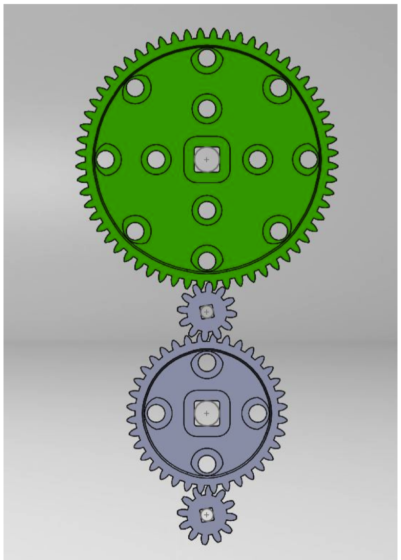


Figure 3.7: Gearing within the towers for the arm mechanism.

The gear ratio is simply the number of teeth in the output gear divided by the number of teeth on the input gear, with the input gear being the one being driven by the motor and the output being the second level gear that the input is powering. The gear ratio for the arm subsystem was determined by:

$$Ratio = \frac{\#oflargegearteeth}{\#ofsmallgearteeth} = \frac{60teeth}{12teeth} = 5 \qquad (3.1)$$

Therefore, the gear ratio for our system was 1:5. Using this value, the maximum allowable torque that the motors can handle before stalling was determined. Since the motors used on the robot came from the VEX catalog, an allowable stalling torque of 14.76 in-lb per motor was given from the specification sheet [11]. This number corresponds to the high torque configuration of the internal gearing for the motors, which was chosen since the motors had to be strong enough to lift the arm mechanism along with the balls. Using the 1:5 gear ratio and multiplying it by the allowable stall torque per motor, the actual allowable stall torque on one motor was determined to be 73.8 in-lb. Since there were four motors in the arm system, the 73.8 in-lb was multiplied by 4 to achieve the entire subsystem's maximum allowable stalling torque; this value resulted in 295.2 in-lb for the entire system.

The amount of torque applied to the motors while the system was operating could also be determined and compared to the stall torque. If this value is higher than the allowable then the motors will fail during robot operation. Figure 3.11 shows the free body diagram of the arm along with the equation:

$$M = Fd - (\frac{d}{2}) \qquad (3.2)$$

where F is 20 lbs, the applied force, or weight of the entire robot, d is 12.5 in, the length of the arm, and W 0.361 lbs, is the weight of the arm. The weight of the robot was over estimated to be 20 lbs. Substituting values into the equation above, it was determined that the torque was 247.74 in-lbs. This number was then divided by 5 to account for the gear change, the torque applied to all four motors was 49.55 in-lbs, which was much less than the allowable stall torque.

3.2.3 Electronics

The electrical components in both VEX robots are motors, a VEX Cortex, sensors, and a battery. The electric motors are used in the drive train, lift for the arm, and the intake. Sensors, shown in Figure 3.8, were two rotational encoders and a potentiometer in the lift mechanism. The two encoders are used in the drive train, one on each side of the robots, and are attached to the motors. The encoders are utilized in the autonomous operation of the robot to measure distance traveled in relation to the playing field. The potentiometer used in the lifting arm of the robot is used to measure and control the motors lifting the arm to prevent them from attempting to move the arm beyond its range of motion. Input from the potentiometer allows the arms to be properly controlled in autonomous mode and sets a limit on the arm when the robot is under teleoperated control. The sensors and motors are all wired to the VEX cortex which is used to implement autonomous coding and also functions as the receiver for teleoperated control of the robots. All of these systems are powered by one 7.2 volt VEX battery, as shown below in Figure 3.9.

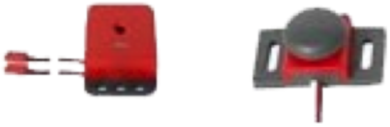


Figure 3.8: Optical encoder and potentiometer used on both robots. [11]

Figure 3.9: 7.2V VEX battery used for both robots. [11]

3.2.4 Software

All the software is contained in six fields: *Competition.c*, *Utilities.c*, *Constants.c*, *Subsystem_Functions.c*, *Autonomous_Selection.c*, and *Autonomous_Modes.c* (see Appendix J for full code files). The top-level file, *Competition.c*, references all the other files and is the only file actually downloaded to the robot. This same file is used for both robots; it can tell which robot is which by reading the value of digital I/O port 1. If it finds a jumper clip, the robot will know it is the robot dubbed Alive. Otherwise, the robot assumes it is the one dubbed Dead.

Competition.c

The *#pragma config* block is not written by hand but created with a GUI within the RobotC editor. It allows the user to control all the settings of the Cortex ports (both input and output), including naming, reversing directions of motors, and configuring inputs for various sensors. This is where the master list of motor/sensor port numbers is kept.

The next few *#pragma* statements are directly from the template and required for competition. The series of *#include* statements reference other files. They essentially import code from a separate file, which allows for better organization of code by separating different parts into different files.

VEX_ Competition_ Includes.c is the code underlying competition protocol, as provided and mandated by VEX. It manages things like going into autonomous or teleoperated mode, and disabling of the robot once the match is over. Since it is provided and required, it is not modified.

Utilities.c, *Constants.c*, *Subsystem_ Functions.c*, *Autonomous_ Selection.c*, and *Autonomous_ Modes.c* are our own files. The "# include" statement effectively copies and pastes the files here; they are only written in separate files for organizational purposes. The following is a list of each file along with a brief description of its contents:

- *Utilities.c* includes functions independent of any robot, like mathematical functions and pieces of the PID logic.
- *Subsystem_ Functions.c* defines all the functions of each subsystem of our robots. For instance, it contains both the teleoperated and autonomous driving functions.
- *Constants.c* is a listing of all the constants used. Instead of hard-coding these into the functions that use them, they all reference this file so that they can be easily tweaked.
- *Autonomous_ Selection.c* contains all the code required to make autonomous selection work. When the robot starts up, if an LCD button is pressed, the screen allows the user to scroll through avaliable autonomous routines and select run to run in the match.
- *Autonomous_ Modes.c* contains the actual routines to be run during autonomous mode. These can be modified at any time and are easy to build from the building blocks defined in *Subsystem_ Functions.c*.

The robot has three modes it can be set to: Pre-Autonomous, Autonomous, or User Control. It is preset to the Pre-Autonomous mode automatically when it boots up, so this mode is used for setting various things that occur at startup. In particular, the robot reads the port for the jumper clip and decides which of our two robots it is and sets the arm potentiometer limits accordingly. It ensures the launcher is in the retracted position. Finally, if an LCD button is pressed, the robot transitions into autonomous selection mode (autonomous selection allows the user to scroll through the available autonomous routines and choose one to run).

In the autonomous task, the only line is *runSelectedAutonomous();*. This calls a function stored in the *Autonomous_Modes.c* so that all the autonomous routines are stored in one place for easy editing.

The user control task contains a *while(true)* loop with functions for each subsystem to be controlled by the operator. All of these functions come from the *Subsystem_ Functions.c* file. This loop repeats the entire time the robot is in User Control mode and continuously updates all the motors based on the user's input each time through.

Subsystem_Functions.c

Subsystem_Functions.c contains the functions that control each subsystem. Each is built in layers. For instance, for the drivetrain, the first function defined is *setDriveMotors()*. This function allows for setting of all the drivetrain motor outputs with only two numbers, the left and right values. On top of this is built a framework for interpreting the user's inputs, filtering out noise, and computing the correct left and right values - the function *setDrive()*. In addition, *setDriveMotors()* is used in the autonomous driving function, *driveInches()*. *driveInches()* takes an input of a number of inches and controls the robot's drivetrain via PID to drive that many inches. Each subsystem follows a similar pattern, with a low level function handling the actual motor outputs, a user control function calling the low-level function based on user input, and an autonomous function using PID to command that system to a given position. Encapsulating the subsystem functions with descriptive function names makes the code readable; it is relatively obvious what each function does.

Utilities.c

Similar to *Subsystem_Functions.c*, *Utilities.c* contains encapsulated functions called in other functions or in Pre-Autonomous, Autonomous, or User Control. The difference is that while the subsystem functions are specific to our robot design, the utilities can be applied to any robot. There are generic deadband and clipping functions, and all the building blocks of the PID logic. The PID logic would have been written more simply as an object, but unfortunately RobotC is not object oriented. Using one set of functions and a global structure for each PID controller simulates the functionality of object-oriented programming. This file also contains a function for displaying the robot's battery voltage on the LCD screen (as this can be accessed remotely through the RobotC debugger and is thus robot-independent).

Constants.c

This file contains all the constants used by subsystem functions, for example, the potentiometer readings at the arm's upper and lower limits and the conversion factor between inches and drivetrain encoder counts. Instead of hard-coding these numbers in the functions, they all reference here so that they can be tweaked easily.

The arm limits are a special case, since these are the only numbers that are different for the two robots (due to the potentiometers not lining up exactly). The limits are defined separately for the two robots. In Pre-Autonomous, the code reads the value of digital I/O port 1. If it finds a jumper clip, the robot decides it is Alive and sets the limits to the Alive ones. Otherwise, the robot assumes it is Dead and sets the Dead limits.

Autonomous_Selection.c

In the VEX competition, it is almost necessary to prepare multiple autonomous routines, as the robot might start the match in one of four positions that is not assigned until shortly before the match. There is also strategic value to having multiple plans of attack from each starting point, depending on the opponents. This file contains all the code to allow a user to select an autonomous routine at the beginning of a match. The left and right LCD buttons scroll through the available options, displayed on the screen, and the center button enters the selection.

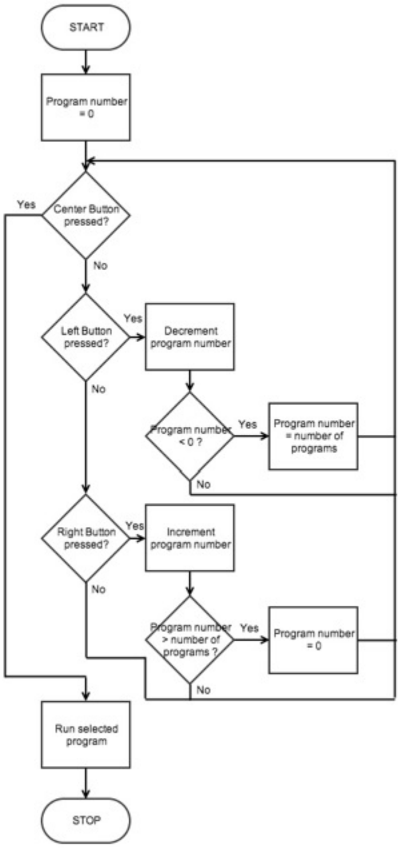


Figure 3.10: Flowchart displaying the operation of the code used for autonomous selection.

Autonomous_Modes.c

This file contains the autonomous routines themselves. They are highly flexible, being built from the autonomous functions established in *Subsystem_Functions.c*, and can be customized to suit different scenarios. For safety, the default option is to do nothing. This requires a user to intentionally select a routine if the robot is to move at all. There are separate routines for each starting position, and more can be added if necessary.

3.3 Analysis

Analysis was performed to determine whether or not the robot could hang without failure. This would be the highest stress configuration the robot would experience since it would need to lift itself off the floor. The motors were first analyzed to see if they could lift the robot without stalling. The arm of the robot was simplified to the free body-diagram seen below in Figure 3.11 and was used to calculate the moment on the arm joint where the motors are. The maximum force the motors could generate was found to be 295.2 in.lb., whereas the moment due to the weight of the robot was found to be 250 in-lb. Thus, though the motors came close to stalling, it was possible for the robot to hang.

After determining that it was possible for the robots to hang, an FEA analysis was performed to see if the materials used in the robots could withstand the stresses involved with hanging.

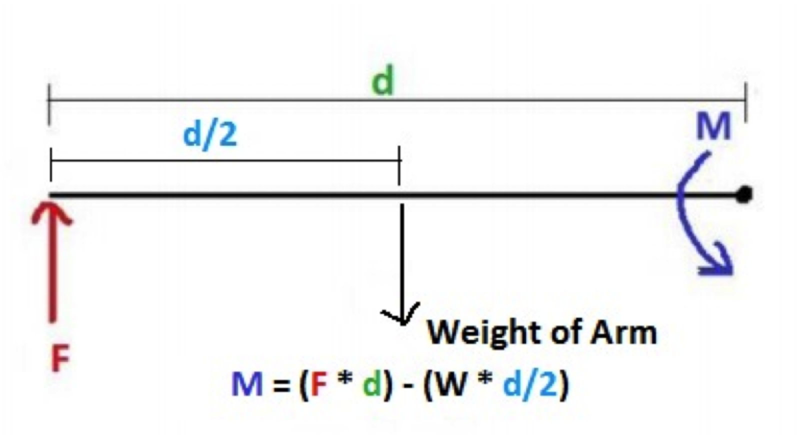
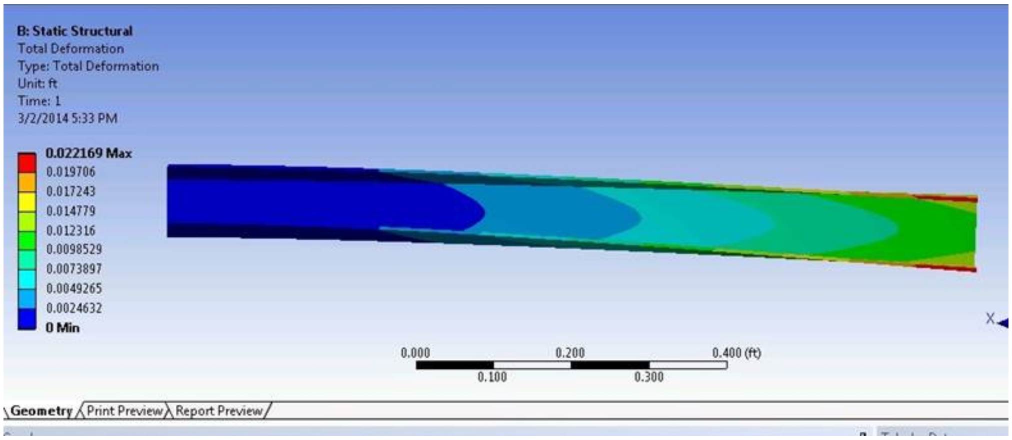
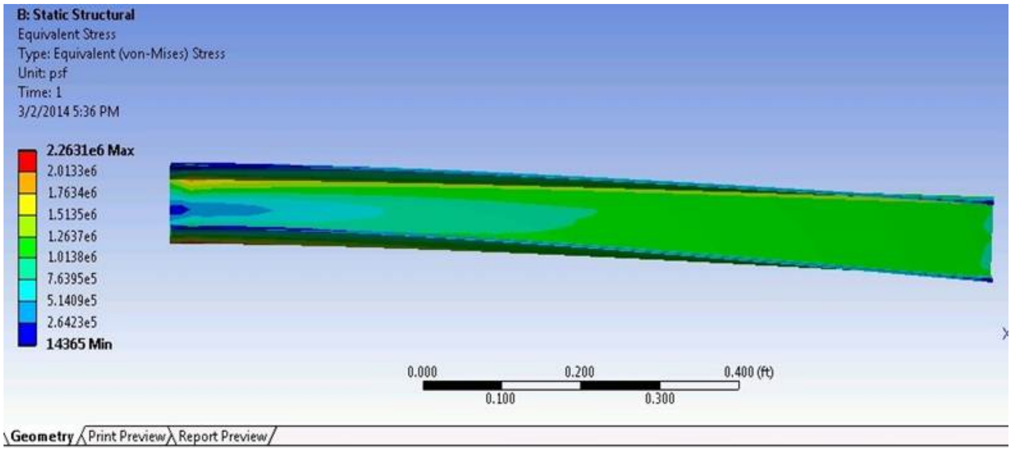


Figure 3.11: Free body diagram of forces on the robot's arm while attempting to lift itself.



B: Static Structural
Total Deformation
Type: Total Deformation
Unit: ft
Time: 1
3/2/2014 5:33 PM

Figure 3.12: FEA of robot arm under lifting forces. Results are for total deformation in feet.



B: Static Structural
Equivalent Stress
Type: Equivalent (von-Mises) Stress
Unit: psf
Time: 1
3/2/2014 5:36 PM

Figure 3.13: FEA analysis of robot arm under lifting forces. Results are for normal stress in pounds per square foot.

It was determined that the aluminum channels would be strong enough to support the robot with a minimal deflection of 0.022169 feet while the motors would not reach stall torque. Since the hanging mechanism would have had the most stresses, it was deduced that the rest of the robot would withstand the stresses during competition. Yield stress of the structural material used in the robots, 5052 H32 alloy aluminum, was found to have a yield strength of 28,000 PSI and an ultimate tensile strength of 33,000 PSI [9]. When compared to the FEA analysis of 2263100 PSF or 15,716 PSI we can see that we are 12,284 PSI away from the yield strength of the material. This result shows that the robots are not in danger of material failure while attempting to hang.

3.4 System Integration Test and Results

For our project, the system integration on our robots was tied to our autonomous mode. The autonomous mode would engage all of the robots' systems in order to complete tasks and different strategies. Those systems that were integrated by the code included the drivetrain, arm, manipulator, and sensors.

The sensors were the most essential part of our system integration. We used drive encoders to determine distance and angle traveled by our drivetrain by using live debugger values while running the robots. We utilized a potentiometer on the arm system and debugged different heights of the arm. These sensor values were used to find the physical limitations in the movement of the arm and the limits were programmed into the control code to prevent the motors from attempting to move the arms beyond their physical limits. The drivetrain and arm systems were then operated using a proportional-integral-derivative controller loop.

The important benchmark of our autonomous mode was based on how quickly and how successfully the robots were able to complete their tasks. Repeatability of tasks was crucial as well. We tested the robots on a replica VEX field and made multiple runs based on our experimental protocol and PDS plans. The autonomous mode strategies (Appendix G) were determined by the maximization of points as set by VEX scoring guidelines. An example of an autonomous strategy used can be seen in strategy G.1. Here robot 1 starting on the hanging side of the field begins facing the two balls underneath the hang bar. At the start of the match it drives forward and collects the two balls, then performs a 180 degree turn and lifts its arm to pass its two balls into the hopper of robot 2 on the other side of the bump. From here robot 1 turns with its back facing the big balls and drives into them to disrupt the big balls as well as knock the three balls on the bump into the goal zone. Robot 2 at this time does not begin activity until it receives the balls from robot 1. Once the balls have been deposited it drives forward and uses its passive knock off bars to knock two big balls off of the bridge into the goal zone. Robot 2 then drives towards the goal, lifting its arm and depositing the three held balls into the goal. After this the autonomous strategy G.1 is finished.

The results of the testing is shown in Appendix K. The "Test Data" table includes the requirement, the target range, the number of trials, and the final result of each test performed. These tests proved the effectiveness of the robots as a solution to the VEX design challenge. Based on the results of testing, changes were made to improve the system. One such change was the addition of the tip wheel as it was found that the robots would occasionally fall over backwards when they approached the bump too fast or at an angle. It also helped stabilize the robot when attempting to score balls in the stashed goals, since the robot would lurch backwards when driving forward with its arm fully raised. Other test results required the robots to be able to manipulate one large ball, 10 small balls, and then interfere with opponents movements while in autonomous. The capability of manipulating this many balls was programmed into the robots autonomous code and the capability of disrupting the opponent's robots was proven while playing practice matches against Bellarmine College Preparatory school. The majority of the other testing criteria, like descoring opponent's balls, manipulating a number of balls in a given time, and scoring those balls in a set time involved human control of the robots. To reach these results the operators for the robots practiced at Bellarmine over the course of a few months to achieve the results outlined in appendix K. Practice was carried out by operating the robots on an empty field as well as in practice matches to prepare the drivers for the challenge of playing against real opponents. Some test results, like collecting 5 balls in 30 seconds, could be done in 20 seconds. However, other requirements, like collecting 5 big balls in 25 seconds, could only be done in 30 seconds.

3.4.1 Design Iterations

The robot design progressed through several iterations as mechanisms were tested and discoveries made. The first major instance of this occurred in the drive train subsystem. There are two major classes of drive trains used in the VEX competition: skid-steer (or "tank") drive and holonomic (or "omni") drive. In a skid-steer drivetrain, two parallel sets of wheels are separately powered, allowing for forward and backward motion and turning. Holonomic drive utilizes omni wheels (Figure 3.14). Omni wheels have free-spinning rollers mounted orthogonal to their axis of rotation, allowing for uninhibited side-to-side motion. Mounting separately powered omni wheels perpendicular to each other allows for controlled motion in any direction as well as rotation. A visualization of these drive systems is shown in Figure 3.15.

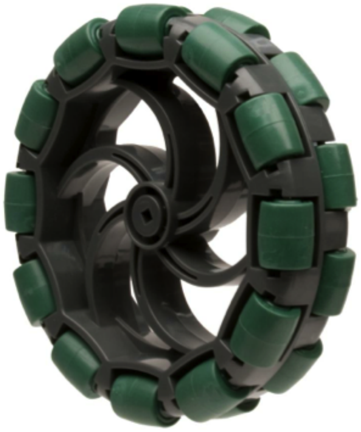


Figure 3.14: A VEX omni wheel. [11]

The initial drivetrain prototype was an H-shaped omni drive configuration, with five powered wheels, as seen in Figure 3.16. However, the perpendicular wheel through the middle of the robot presented difficulties in traversing the bump dividing the sections of the playing field. In order to solve this, the drivetrain configuration was changed to a six-wheel skid-steer drive. The simplicity and robustness of this solution far outweighed the advantage of the omni drive's increased maneuverability.

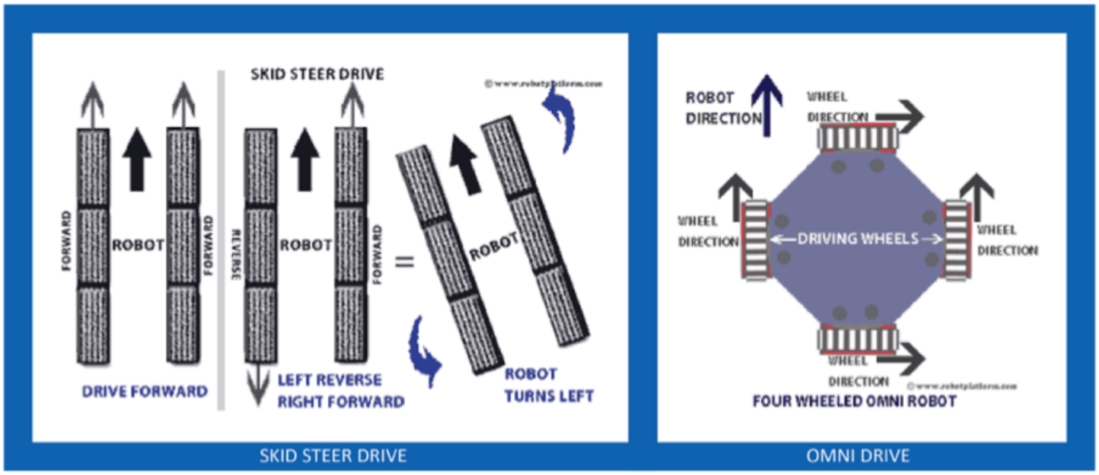


Figure 3.15: Skid-steer and holonomic drive trains.

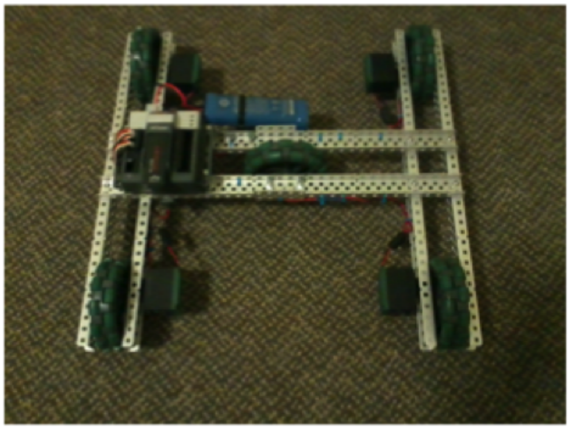


Figure 3.16: H-shaped holonomic drive train prototype.

During testing, it became apparent that large accelerations hence the robot's center of gravity was elevated had the potential to tip the robot over. An "anti-tip wheel" was added to the base to prevent this (Figure 3.17). The wheel is mounted on an arm that folds into the base to fit within the allowed volume at the start of a match, but is spring-loaded to pop out at a slight vibration. When the robot begins to tip backward, this wheel contacts the ground before the robot can fall all the way over.

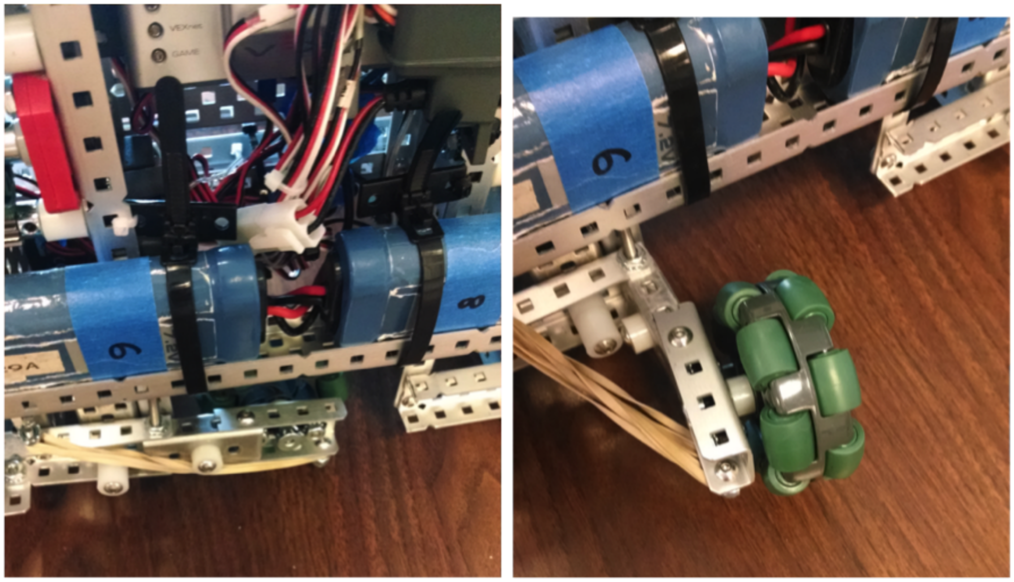


Figure 3.17: The anti-tip wheel when a) stowed and b) deployed.

Another discovery from testing was that captured game objects were not well controlled when the arm was raised, often falling out the sides of the hopper as the robot turned. To solve this, rails were added along the outside of the hopper that fit within the arm linkages but controlled the balls during lifting (Figure 3.18).

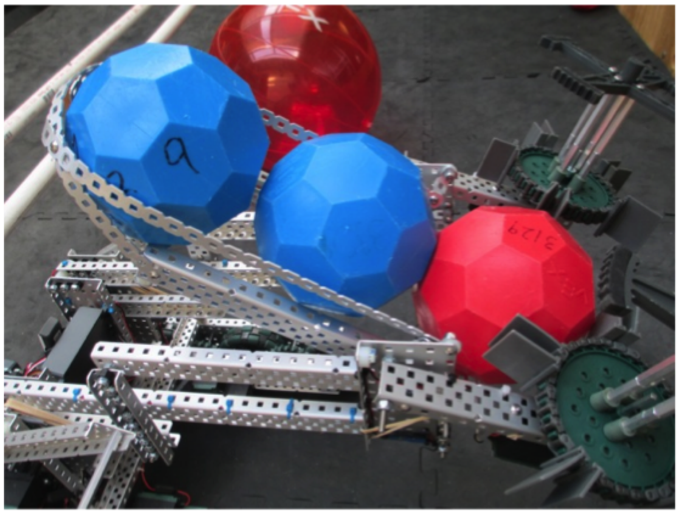


Figure 3.18: Added hopper rails to prevent the elements from falling out.

Passive, spring-loaded pivoting bars were also added to the top of the arm towers during testing (Figure 3.19). These bars allowed the robot to passively knock balls off of the barrier without having to raise the arm. The spring-loaded pivot allowed the bars to swivel in order to fit under the barrier but return to their default position afterwards.

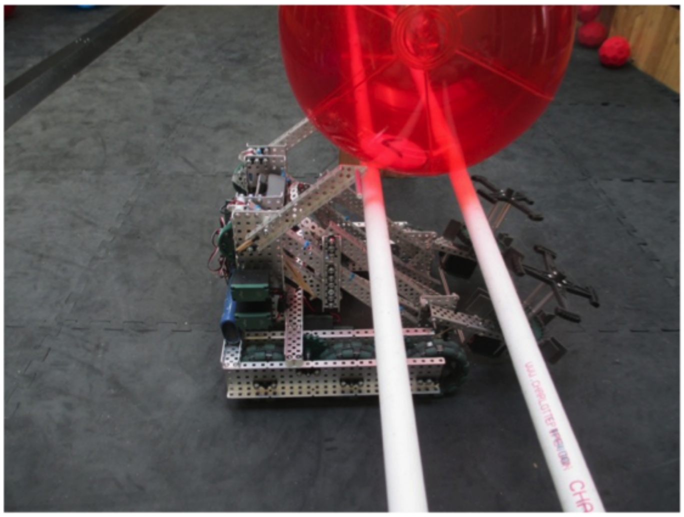


Figure 3.19: Passive knock-off bars.

In the final phases of testing, the robots were observed to lose power after intense operation. Much research traced the issue to current being limited by fuses in the Cortex microprocessor. To work around this, a power expander was added to each robot. The power expander allows for the use of an additional battery and, more importantly, contains a separate fuse. By splitting the current supply through the Cortex fuses as well as the power expander fuse, consistent operation with no sudden losses of power was achieved.

3.5 Cost Analysis

The project budget was determined by analyzing budgets of previous VEX competition teams and adjusting for the two robots required in VEX U and the costs associated with starting a new team from scratch, such as purchasing tools. The proposed budget is shown in Appendix D1. The total cost was $6,055. $1,000 of this was reserved for travel costs; the remaining $5,055 was used for tournament registration as well as all expenses associated with the robots, including parts, tools, and software.

The choice of the VEX U robotics competition as the vehicle for the project left relatively few choices for cost analysis. In order to abide by competition rules, all parts must be purchased from VEX's catalog, and many parts are de facto requirements with no alternatives. For instance, the Cortex microcontroller, VEXNet wireless communication keys, Joystick controller, RobotC software, and VEX 7.2V batteries are all required for competition.

The VEX competition's model of limiting teams to one parts supplier is a mechanism for leveling the playing field. Teams who can afford to "throw money at their robots" do not have a large advantage over teams who are not as well-funded, because all team have access to the same parts. Rather, ingenuity and clever use of parts is encouraged. Because of this, cost analysis is mostly about determining whether or not certain parts are needed for the team's chosen design.

Late in the project the team had to decide whether or not to pursue a hanging mechanism. As discussed in the analysis sections, in the 2014 VEX challenge, Toss Up [12], robots could suspend themselves from a bar elevated forty inches above the playing field to earn bonus points at the end of matches. The SCU VEX team's robots performed all of the challenge tasks except this one, and the idea of adding a mechanism for the purpose was discussed. In addition to the technical challenges presented by such a high-stress application, developing a hanging mechanism would have introduced significant cost to the project budget. With the motor budget already exhausted, another active system would have necessitated the use of a pneumatics system. This is a very costly system; VEX's pneumatics kit costs over $220. The mechanism would have required at least one kit per robot for a total of nearly $500, not including any extra structural parts and parts used in prototyping and development. Due to this unexpectedly high cost and analysis of the value of the hanging bonus, the team decided that it made more sense to use the remaining time to improve the existing systems of the robot, reserving the rest of the budget for emergency purchases such as replacement parts.

Chapter 4

Business Plan

4.1 Executive Summary

This business plan outlines the strategy of the VEX team to create a company or organization which offers assistance to aspiring future VEX teams around the world. In an effort to educate more and more people in the fields of science, technology, engineering and mathematics (STEM), VEX lays out a strong platform for increases in knowledge of these fields. VEX allows students to get hands-on experience with robotics technology in both an educational setting as well as a competitive one. With the exponential growth in robotics technology [4] today the Santa Clara University VEX robotics team would like to potentially partner with VEX to offer assistance to teams around the world so that they too can participate and learn through VEX. The proposed organization will expand the educational platform by offering robotics workshops to anyone wanting to take on a VEX team in the future and even to those who simply want to learn more. The company will also offer mentoring, funding, and sponsorship of teams who are having a difficult time starting up or who need guidance from experienced individuals.

4.2 Introduction

What is VEX? The VEX Robotics Design System offers students an exciting platform for learning about areas rich with career opportunities spanning science, technology, engineering and math (STEM). These are just a few of the many fields students can explore by creating with VEX Robotics technology. Beyond science and engineering principles, a VEX Robotics project encourages teamwork, leadership and problem solving among groups. It also allows educators to easily customize projects to meet the level of students' abilities. The affordable VEX platform is expanding rapidly and is now found in middle schools, high schools, and university labs around the globe.

The primary goal of VEX is education and the SCU VEX Robotics team would like to enhance this goal even further by providing assistance to VEX teams that need it. At times it becomes difficult to get the necessary funding to start or even finish a VEX Robotics project so this would be one way that the SCU team can help. This was actually an issue that the team faced during the process but with the help of those willing to give it our wishes of competing in the VEX competition came true.

VEX designs change each year as the competition game changes completely from year to year. In order to accommodate these changes each year the services provided (funding, sponsorship, mentorship and workshops) will remain with minor adaptations based on each year's new game design. Again, the VEX team's desire is to guide young and enthusiastic individuals to become inspired by the rapid evolution of technology to create something truly special. The experience as a whole is one that is unforgettable and something every student should be able to pursue, and this organization proposes to make that a possibility for all.

The members of the SCU VEX Robotics team will be the co-owners of the company and be the leading forces in getting the company off the ground.

4.3 Goals

The VEX Robotics team has a goal to enlighten and educate students about the field of robotics while providing support in their aspirations to compete in the VEX Robotics competition. Funding, sponsorship, and mentorship will be provided to teams who need the extra help as well as to the teams that need guidance from the ground up.

4.4 Product Description

For this year's VEX competition the game involved traversing an obstacle ridden environment to score bucky balls into designated scoring zones. The two robots were designed to accommodate these game parameters to succeed in scoring more points than the other VEX teams in the most effective and efficient manner. The interesting part is that next year the game changes to something completely different therefore the potential product to be marketed is rather broad, depending on the game design. What the SCU VEX team would like to then focus on is the service to others. In essence the product is more of an entire package sort of deal where this team would lend its service to other teams around the world.

One of the aspects of this "product" is the team's ability to design efficient, cheap and effective robots each year to meet the game design requirements and game parameters. These designs can then be offered to other teams who need help with ideas as well as part of potential sponsorship. With a full sponsorship from the proposed organization/company funding may be provided along with mentorship, that is, provide guidance to teams based on the SCU VEX team's first-hand experience with the competition.

Another service that will be provided is access to parts from a large surplus of VEX parts, ranging from brand new to hand-me-down parts. One goal of this organization is to get ahold of old VEX parts from previous teams as well as donation kits much like the ones that the SCU team received during the making of its robots. Again, the utmost important goal is to enhance student's education in the creation and operation of robotics technology through the VEX competition. Since the VEX competition incorporates a wide age and education level range (elementary school up to college) each individual team's skill and experience level will be managed accordingly. Workshops will be held to further push the educational side of VEX and robotics in general.

4.5 Potential Markets

The potential markets for this proposed company are broad within the educational technology market. With the growth of robotics in society today [4] the need for robotics education and immersion will continue to grow along with it. The markets could range from setting up workshops for elementary school and middle school students to financial sponsorship of high school and college teams. Location is not an issue either since this competition draws in teams from many countries around the world. At this year's world championship there were teams from Mexico, New Zealand, and 20 other countries. VEX has a very large reach as the largest educational robotics program in the world. The potential for this company to expand and grow is very likely as VEX continues to expand globally. Finally, since we propose to sell reused parts for a much lower price than brand new, the teams with a larger need for parts like the ones from less privileged countries, our organization would be a perfect outlet.

This company's hope is for rapid mass expansion. From a first-hand experience point of view at times it is very difficult to get that last bit of funding for the project since big corporations such as NASA or Lockheed Martin allocate some funding to teams but once that funding is used then that outlet is essentially closed off to all other teams. With a vast majority of the leftover teams needing aid this company would be a prime candidate to hear their call for help.

4.6 Competition

The SCU team is not proposing to market any one specific product but rather provide VEX teams with the necessary means to learn, create and compete in the competition each year; therefore the market competition is practically non-existent. Companies devoted solely to helping VEX teams that need funding, parts, designs, and even mentoring are not prevalent in the VEX world at the moment.

4.7 Sales Strategies

While there is the potential to profit from this organization and this service, this organization/company would primarily operate as a non-profit type of corporation. The sale strategies for SCU VEX would revolve around a good marketing platform, one which gets the word out to the world that there is a cheaper alternative for the construction of these VEX robots than buying brand new parts. Also, as an added bonus, this company is offering up its services in the field of mentorship and further robotics education.

Based on the educational background and experience of VEX team members, workshops will be held to teach and expand the knowledge of robotics enthusiasts and VEX competition hopefuls at a reasonable price to attend if not free. Either way, the workshops will provide a perfect place to further market the company where we will advertise refurbished and donated VEX parts as well as promote our other services. Hopefully this will get the company's name out there for others to find.

The SCU VEX team would also like to promote and market the company via social networking applications. A Facebook and Instagram profile will be created where new deals and services can be advertised. A page specifically designed for those seeking extra funding to finish their projects will be available as well where teams can fill out forms requesting funding, and even teams looking for mentoring can come and request that as well.

A few of our team members have been participating in VEX and other robotics events like it for many years and have built lasting relationships with the VEX community. We would like to reach out to this community as well to spread the news of this company's existence and its importance in the advancement of robotics technology education. Since VEX itself is designed to educate, our organization would only further educate society.

From a strictly sales point of view, the SCU VEX team's strategy in terms of making a profit would be from selling reused VEX parts for a much lower price as well as sell design plans each year to help newly formed teams get started with their robots.

4.8 Pricing

From a VEX parts point of view the pricing will vary based on the condition each part is in. With all brand new parts the typical cost of each robot is usually around $3,000, where the bulk of that cost comes from the VEX microcontrollers. Since a new microcontroller is somewhat essential, the cost of those will most likely remain a constant from VEX's online catalogue, roughly $700. The building pieces are much more likely to change from their manufacturer's original pricing and these may vary depending on their condition as stated previously. The aluminum and steel framework pieces that make up the bulk of the actual physical robots are reusable to some extent, provided that the pieces have not been bent or tampered with too much. What this team would like to do is attempt to cut the cost of building each robot by a third, but since this company is not selling fully assembled robots that price will depend on the design and monetary availability of each individual VEX team.

4.9 Services

Our service primarily focuses on building awareness of the VEX program and also furthering robotics in education. This means that customer service and relations is essential. Our clients must see the value in what we are attempting to grow: Robotics and STEM. This can be done as our team now has great experience in VEX and have all been a part of the large scope of such a project. Our advantage is in the fact that we all closely worked on each process of the competition. Our knowledge is a key selling point; how we market ourselves will bring in customers.

As mentioned above, our business would also deal with reselling VEX used parts. This means that the parts we have on hand must be well organized and sorted to make sure that orders will be quickly processed and fulfilled. The business would try to ensure that all parts are of a certain standard; however, in case the condition stated is not met, the business will do a full refund. This of course means that the process of inspection of old parts must be very rigorous. The importance of quality control and customer relations must not be taken lightly.

The analysis of stresses on aluminum and steel framework pieces will also be performed to make sure that they will not fail due to previous competition use. Motors can also be tested to determine reusability and sensor values can be evaluated for accuracy. All in all, the business will attempt to make sure that the experience of using older parts will come with no worries or a lack of quality.

4.10 Financial Plan

Initial investments will go towards purchasing a wide array of VEX parts directly from the VEX store as well as finding a suitable location to store them as well as provide a good place of work for the company to start out. This company will act primarily as a non-profit organization but at the same time there will be slight profits from the workshops as well as resale of old VEX parts in order to keep the funding and sponsorship aspirations alive.

Chapter 5

Engineering Standards and Realistic Constraints

Our project focus was on developing a multi-robot autonomous system and testing it by competing in the VEX Robotics competition. The goal was to create two autonomous robots with the intention of applying it to search and rescue applications. We believed that the VEX competition, which requires that robots maneuver through an obstacle ridden environment while avoiding other robots, would provide a situation analogous with situations encountered during search and rescue operations.

Unfortunately, the scale of our project was reduced after realizing that our budget was not optimal for it. This drastically limited the types and quantity of sensors we could use on our robots. This caused our focus for the project to shift from developing a fully autonomous control system for the two robots (with the use of sensors) to building robots that could handle simple autonomous modes and win the VEX Robotics competition. Potential areas our project could impact are the VEX robotics society and those interested in joining future VEX robotics competitions.

Economic

Our project required that we design and build two robots for the VEX competition. Economic considerations regarding the project were a major factor in the design and construction of our two robots. Complications regarding the budget for the project arose when the requested budget of $5000 from Santa Clara University School of Engineering was cut in half and an additional $1000 from NASA Ames Research Center was delayed. This restricted early progress to only strategy and initial design of the robots. Despite this setback construction was carried out in earnest in December despite not enough funding to complete the second robot. Luckily Santa Clara University's Mechanical Engineering department donated an additional $2500 to our project in February, which allowed the team to operate under the expected budget for the project and parts for the second robot could be ordered. However, since the competition was in April, this resulted in only two months to complete both robots and have them ready for competition. The initial budget constraints limited our choices of sensors used in the robots as well as causing us to abandon implementing a hanging mechanism in the robots.

Although the initial motivation for our project was for search and rescue applications, this had to be scaled back due to the aforementioned budgeting issue faced by our group. However, what can be examined is the accessibility of VEX for outside schools or teams who are interested in competing. Looking at the costs of our two robots it should cost approximately $5000 to cover the costs of materials($4000) and registration($1000). Additionally, money should be saved for the cost of travel since this is where the additional $1000 from NASA Ames was utilized. Although examining only the cost of robot construction it should cost approximately $4000 for a college team and $2000 for a high school team since college teams must build two robots whereas high school only need to construct one. A cost of around $5000 for college and $3000 for high school could be difficult for some teams to afford; however, there is a large number of companies who are willing to sponsor teams so they can participate within VEX. From this we can hopefully show that the economic costs of competing in VEX are not too high and thus more people may be willing to participate further.

Manufacturability

For our project, manufacturability was a major concern which luckily the rules of the VEX U competition helped to facilitate. Since the project competed at the VEX U robotics competition to determine the feasibility and execution of the system, it was required that we only use parts that can be ordered from the VEX catalogue. This was done to insure an even playing field between all competitors. Despite the need to order stock parts from VEX there is still a likely chance that a part may be damaged by the opposing team, resulting in a need to replace the part as quickly as possible. Therefore, the easier it is for the product to be manufactured, the easier it will be to swap out the part. To this extent designing modular robotics systems was done to ease repair and modification of the systems on the robots.

Ethical Issues

While working towards the VEX robotics competition a number of ethical considerations needed to be addressed. These considerations were directed at how the team interacted within itself as well, how the team interacted with outside groups, and the ethical nature of the robots we were constructing. When addressing the team aspects of the project, one of the advantages in taking up a team/competition-based project is that the team roles could be clearly defined. We have members that have been participants and volunteers of the VEX Robotics Competitions for a long time, which has helped to guide our project. The team makes sure to have agendas and weekly notifications for all of our meetings, along with meeting minutes taken for documentation. Furthermore, the locations, times, and tasks for each are well defined. We also follow the three C's (competence, conscience, and compassion) of Santa Clara University in our interactions with both our sponsors and Bellarmine College Preparatory robotics team.

The ethical nature of the operation of our robots was carried out by following the ASME Code of Ethics [2]. This was performed by making sure our robots could not inadvertently harm others by programming the robots with certain restrictions. Each time the robot was turned on into autonomous mode, 0 actions would be taken. In order for the robot to move autonomously, it would wait for a human to input which program to perform. Within the human controlled period, the robot would always perform the actions that a human driver would command to the robot. If it were told to drive outside of the range of connectivity, the robot would cease all action. After every match the robots would be serviced to make sure they would not suffer from a mechanical failure which could harm bystanders.

Health and Safety

There were a few health and safety considerations when designing and thinking about our project. When building the two robots we had to ensure that the finished product would be safe to operate as well as safe to be around other people. To this extent the need of the two robots to operate efficiently with one another and operate effectively in their environment.

Since our project focuses on competing in the VEX competition there are many safety regulations that are set by the company which must be met prior to competing as well as during competition. The primary safety rule for the VEX competition involves the protection of other individuals around the playing field. "If at any time the Robot operation or team actions are deemed unsafe or have damaged the Field Elements or Scoring Objects, by the determination of the referees, the offending team may be Disqualified. The Robot will require re-inspection before it may again take the field." [13] It must always be assumed that the worst scenario can happen, worst being a robot fails to adhere to the inputs given by the driver during competition and goes out of control.

With functionality and operational safety in mind, the two robots were assumed to be the most "out of our control," when developing the autonomous modes. To develop safe autonomous modes we had to write the code in such a way that it would stop the robots once the code finished running. Also, without a default autonomous mode the robots would jump into the code as soon as the robots were turned on. In order to limit injuries to the person the initial default autonomous mode has the robots sit still so that the operational modes can be selected afterward and then activated by a switch on the field.

The health and safety impacts on our project were considered and dealt with accordingly. From a mechanical viewpoint, the parts used needed to be safe for handling purposes. This was done by filing down the sharp edges on the metal components. From a coding point of view, the control code was developed so that there was a safe default mode, which the robots went into. In order to get out of the default mode and into the desired one had to be done by hand that way the human element remained in the equation. In the end, knowing the safety regulations for the VEX competition we constructed the two robots to adhere to these rules to make the experience for everyone involved safe and enjoyable.

Civic Engagement

The greatest social aspect our project affected was the interest in robots that was generated while at the VEX competition. We noticed that most people were very interested in finding out how our physical robots actually operated. The VEX program also has grown tremendously recently and has increased its outreach and education efforts.

The 2014 World Championship was held at the Anaheim Convention Center and hosted more than "15,000 participants from 27 countries, as 760 of the world's best student-run robotics teams demonstrated their Science Technology, Engineering, and Math (STEM) skills" [5]. Outside of the World Championship, in 2013-2014, the program has had "over 10,100 teams from 33 countries [participating] in over 750 competitions worldwide" [5].

The consequences for such a project are very positive. The increase in awareness for such a project can lead to a life-long interest in robotics. According to magazine, *Visions Systems Design*, "in 2013, global robot sales will increase by about 2%, but from 2014 to 2016, the industry will see worldwide sales increase by about 6% on average per year" [4]. This shows that the robotics industry will soon become a larger and a more crucial part of engineering. It was also noticed at the Senior Design Showcase that people of all ages crowded around our demonstration. The live presentation of a robot scoring goals and maneuvering with ease was both exciting and eye-catching.

Chapter 6

Conclusion

The VEX World Championships competition proved that our robots were successfully designed. We excelled due to our streamlined process and strategic autonomous modes. This was also on the back of budget constraints and less time than initially planned. We would have liked to improve on many aspects of our project; however, we were satisfied with our results and our robots were optimally built. At the competition, we placed seventeenth out of sixty teams during qualification matches which earned us a place in the elimination rounds. We unfortunately lost in our second match, ending up at sixteenth place, but we met our expectations in building a solidly designed robot that economically strived to achieve most of the requirements of the competition.

The VEX World Championships were a great benchmark as we were pitted against other University teams in similar situations to ours; however, a majority of the other teams have had foundations in the VEX program and had an advantage in time allotted for the robots. Our budget constraints delayed the start of our robot building and testing until late February; other teams have been working on their robots for more than an year. This was daunting, but we were able to come up with a design which focused on enabling our robots to manipulate field elements with great speed. Our opponent robots were on average much slower than ours and our simple design allowed us to defeat robots who had been built for an excessive amount of functions.

During our testing phase, we used our experimental protocol to maximize the amount of points that the robots could win in a match. At the practice fields of Bellarmine College Prepatory, we planned out our autonomous modes to disrupt opponent maneuvers as much as possible. This was once again possible due to our robots' speed. We would be able to reach parts of the field much more quickly than others. This allowed for our robots to act as barriers with the use of field elements (Appendix: Autonomous Strategies). Our robots were roughly twelve pounds and had just the essentials; we did not include heavy pneumatic systems and had no excess subsystems.

6.1 Problems Encountered

The team did face a couple of disadvantages: a lack of hanging and communication errors. Due to economic considerations, our robots were not able to hang. The hanging aspect of the project required robots to hoist themselves off the field, onto a pipe. The group found that this was not practical for the scope of our project. We calculated that hanging causes extreme stress and stalling on the arm system motors so it was abandoned; furthermore, it would have required extra motors and aluminum channels which we did not have the funds for. The competition bonus for hanging was substantial, but winning the autonomous mode evened out this point advantage in most of our matches.

Another problem was caused by the VEX Cortex to controller communication. We found that our robots' would sometimes have difficulty communicating during the tele-operated modes. This caused our robots to simply stop operating on the field during crucial moments. We troubleshooted this error by testing different wireless communication keys and replacing batteries frequently; however, the cause of this issue was never fully solved. We now believe that it was a firmware error on VEX's part. Unfortunately, VEX had decided to launch new keys halfway during the project process. This required a less than optimal update of software and firmware on the Cortex and wireless communication dongles.

6.2 Competition Results

Even with our setbacks, the SCU team was able to perform better than other prestigious engineering school teams. Rice University and Purdue University were ranked below our team despite their strong background in engineering. In fact, we surmised that we would have placed much higher if our project had been able to meet our initial goals. We observed that Purdue and Rice met their downfall due to the construction of overly complex robots. While their robots had much more subsystems and functionality, their complexity and inefficiency proved to be their Achilles' heel.

We discovered that troubleshooting larger systems in a competition setting proved to be disastrous. Issues with wiring and mechanical malfunctions almost affected the performance of our robots, but we were able to quickly discern the problems and solve them due to our thoughtful design. Other teams were not as lucky. NAR's team had created a large robot which seemingly maxed out the 24"x24"x24" size limit, but it spent most of its time at the competition in their pit area. Purdue's team created robots similar in design to ours (arm systems with manipulators on top of drivetrains), but they did not maneuver as well due to their attempt to encompass all aspects of the competition. Other teams had to use their backup robots or were spending excessive amounts of time in their pits to work on their subsystems.

The important takeaway from the competition was in the design philosophy. We could have followed the mold set by other VEX teams by incorporating more subsystems to gain more points; however, more was not always better. We specialized in what we were able to streamline: speed, autonomous strategies, and maneuverability. We believe that this optimization led to our moderate success. In the real world, not everything can be a Swiss Army pocket knife. Products have to be created to meet a purpose and must perform that function well in order to be beneficial and have appeal.

Table 6.1: Top 20 teams at the VEX World Championship Event.

Rank	**Team**	**Wins**	**Losses**
1	Universidad Tecnologica De Gutierrez Zamora	10	0
2	OYES Robotics	10	0
3	Vaughn College of Aeronautics and Technology	10	0
4	University of Vermont	9	1
5	Universidad Politecnica de Tapachula	9	1
6	University of Colorado Boulder	9	1
7	Universidad Politecnica de San Luis Potosi	8	1
8	Universidad Technologica de La Huasteca Hidalguense	8	2
9	Universidad Tecnologica de Gutierrez Zamora	8	2
10	FEARLESS Robotics (College) Competition Club	8	2
11	New York Institue of Technology	8	2
12	Massey University	8	2
13	PR Institute of Robotics	7	2
14	Universidad Tecnologica Emiliano Zapata	7	3
15	Universidad Tecnologica de Matamoros	7	3
16	Pacific Youth Robotics Society	7	3
17	**Santa Clara University**	**7**	**3**
18	Riobotica	7	3
19	Rice University: Rice Robotics Club	6	4
20	Moorpark College Engineering Club	6	4

6.3 Improvements

If the group had extra funding and time, we would have focused more on the sensor subsystem. A more robust sensor subsystem would have allowed us for a more complex autonomous mode. Our robots currently only feature potentiometers and line encoders. This meant that our autonomous mode was programmed to be basic; the robots simply followed move commands that we measured out by matching sensor debug values with distance or angle traveled. This meant that the robots could not account for changes in field elements or opponent robot movements. Fortunately, our autonomous modes were meticulously written in order for the robots to perform the most efficient routines.

The addition of other sensors such as line trackers, bumper switches, and ultrasonic range sensors would have provided great improvements to our autonomous system. Line trackers, bumper switches, and ultrasonic range sensors would have allowed the robots to better traverse the field and with less human input. Bumper switches would have allowed for robots to avoid obstacles, opponent robots, and the field walls. The robots would have been able to locate and manipulate field elements by determining their proximity with ultrasonic range sensors. An improved sensor subsystem would have further ensured our ability to gain the autonomous bonus, and would have been a greater accomplishment to feature for our project.

All in all, we were very satisfied with our project. We met the revised goals that we set and the VEX World Championships were exhilarating; seeing our robots in action against others provided a great reward for our hard work. There were many aspects we did well, especially in design, but we felt that we could have had a more robust sensor subsystem. Through rigorous testing and determination, we were able to create robots which performed on scale with more expensive and more complicated counterparts.

Bibliography

[1] S. Al-Hasan and G. Vachtsevanos. Intelligent route planning for fast autonomous vehicles operating in a large natural terrain. *Robotics and Autonomous Systems*, 40:1–24, 2002.

[2] ASME. Code of ethics of engineers, November 2006. https://www.asme.org/getmedia/9EB36017-FA98-477E-8A73-77B04B36D410/P157_Ethics.aspx.

[3] AURA. Six-bar linkages. http://www.aura.org.nz/archives/672, 2011.

[4] James Carroll. Report: Robotics industry growth on the horizon. http://www.vision-systems.com/articles/2013/09/report-robotics-industry-growth-on-the-horizon.html.

[5] REC Foundation. After three action-packed days, stem champions are crowned at the 2014 vex robotics world championship, April 2014. http://www.roboticseducation.org/after-three-action-packed-days-stem-champions-are-crowned-at-the-2014-vex-robotics-world-championship/.

[6] Pettinaro C Giovanni. Teamwork by swarms of all terrain mobile robots. *Industrial Robot: An International Journal*, 31:519–26, October 2013.

[7] Giacomo Indiveri and Rodney Douglas. Neuromorphic vision sensors. *Science*, 288:1189–190, May 2000.

[8] J. Acain Mas, S. Li and C. Kitts. Entrapment/escorting and patrolling missions in multi-robotcluster space control. *Proc IEEE International Conference on Intelligent Robots and Systems*, pages 5855–61, October 2009.

[9] ADI Metal. Flat sheet (5052-h32). http://aluminum.adimetal.com/viewitems/5052-alloy-aluminum/flat-sheet-5052-h32-, 2014.

[10] Kazuyoshi Miyazawa. Fire robots developed by the tokyo fire department. *Advanced Robotics*, 16:553–56, 2002.

[11] VEX Robotics Design System. Vex product catalog. http://www.vexrobotics.com/vex/products/, 2014.

[12] VEX. Vex competition overview, 2002-2014.

[13] VEX. Toss up, August 2013.

[14] VEX. *VEX Toss Up Game Manual*. VEX Robotics Inc., May 2013.

Appendix A

Product Design Specification

Table A.1: Specifications that were taken into consideration throughout the design process.

Elements/Requirements	**Units**	**Datum**	**Target Range**
Total Robots	Number of Robots	2 Robots	2 Robots
Small Robot Size	Inch	15" Cube	15" Cube
Large Robot Size	Inch	24" Cube	24" Cube
Motors	Number of Motors	12 per Robot	<12 per Robot
Robots Hanging w/ Ball	Inch	12" Hanging Height	10"-12" Hanging Height
20 Bucky Balls	N/A	N/A	Collect 5 balls in 30 seconds
8 Large Balls	N/A	N/A	Collect 5 balls in 25 seconds
12" Barrier	N/A	N/A	Maneuver around barrier for 60 seconds
2" Bump	N/A	N/A	Maneuver across bump for 60 seconds
Middle Zone Balls	N/A	N/A	Move 10 balls in 30 seconds into zone
Goal Zone Bucky Balls	N/A	N/A	Move 10 balls in 30 seconds into zone
Goal Zone Large Balls	N/A	N/A	Move 4 balls in 30 seconds into zone
Stashed Bucky Balls	N/A	N/A	Stash 10 balls in 30 seconds
Stashed Large Balls	N/A	N/A	Cap 4 balls in 30 seconds
De-Score Balls	N/A	N/A	Descore 5 small balls and 4 large balls in 60 seconds
Autonomous	N/A	N/A	Perform maneuvers autonomously
Center of Gravity	N/A	N/A	Low center of gravity for stability

Appendix B

Experimental Protocol

Table B.1: Protocol created as guidelines and basis for testing of the prototyped systems.

Evaluation	**Equipment**	**Accuracy**	**Expected Outcome**
Small Robot Size	Measuring Tape	The small robot must be no bigger than 15"x15"x15" in size	15"x15"x15" in size
Larger Robot Size	Measuring Tape	The small robot must be no bigger than 24"x24"x24" in size	15"x15"x15" in size
Robot Hanging w/o Ball	Replicate bar on practice field	Subjective	8"-10" hanging height
Robot Hanging w/ Ball	Replicate bar on practice field, VEX practice field large ball	Subjective	8"-10" hanging height
Scoring Bucky Balls	Replicate VEX practice field and game elements	The robots should be able to autonomously collect and gather field element balls (3 inches)	Collect 5 balls in 45 seconds
Scoring Large Balls	Replicate VEX practice field and game elements	The robots should be able to autonomously collect and gather field element balls (3 inches)	Collect 5 balls in 30 seconds
12" Barrier	Replicate VEX practice field and game elements	The robots should be able to autonomously function and move within the competition arena (6 inches)	Maneuver around the barrier repeatedly for 50 seconds
2" Bump	Replicate VEX practice field and game elements	The robots should be able to autonomously function and move within the competition arena (6 inches)	Maneuver around the bump repeatedly for 50 seconds

Continued on next page

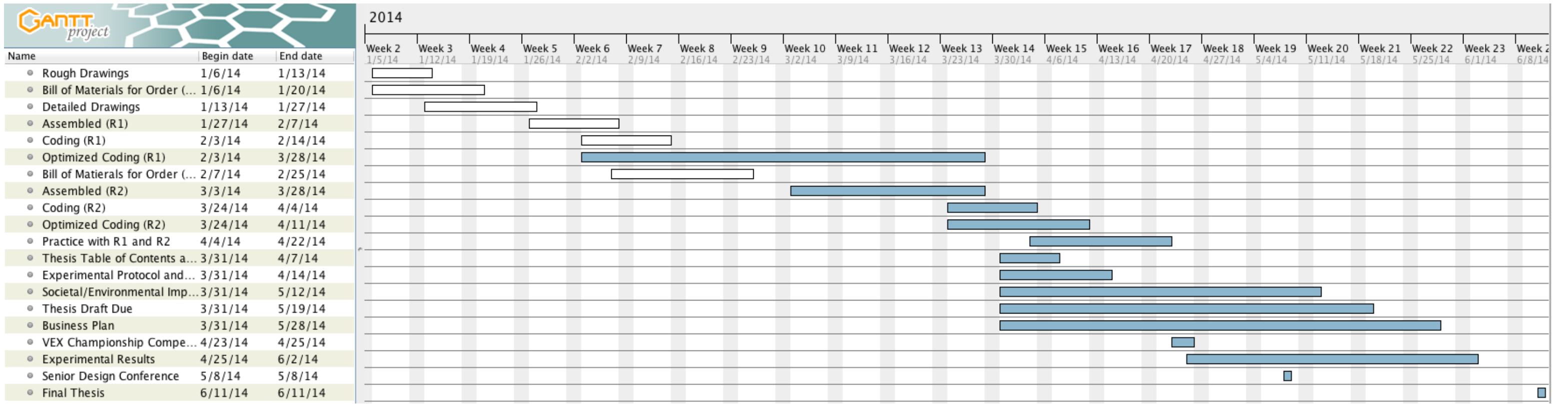
Table B.1 – *Continued from previous page*

Evaluation	**Equipment**	**Accuracy**	**Expected Outcome**
Middle Zone Balls	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 10 balls into zone in 45 seconds
Goal Zone Small	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 10 balls into zone in 45 seconds
Goal Zone Large	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 4 balls into zone in 45 seconds
Stashed Zone Small	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 10 balls into zone in 45 seconds
Stashed Zone Large	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 4 balls into zone in 45 seconds
Autonomous Bonus	Pratice field, laptop with Robot C	Perform tasks within 4 inches of tolerance	Operate autonomously and collect 4 large balls and 10 small balls
De-score Enemy Balls	Practice field with game elements, practice partners	The robots should be able to move collected opponent balls out of scoring zones (4 inches)	Able to collect and move out 5 opponent small balls
Block From Hanging	Practice field with game elements, practice partners	The robots should be able to disrupt opponent robots (10 inches)	Able to interrupt opponent robot from hanging
Low Center of Gravity	Practice field, stability checks through experiments	Robot should be able to hang and stay balanced during maneuvers	Able to collect and move balls in motion without losing stability
Weight	Practice field, stability checks through experiments	Robot should not be hindered by weight due to systems added	15 pounds

Appendix C

Gantt Charts

The following page contains the VEX project gantt chart.



Name	Begin date	End date
Rough Drawings	1/6/14	1/13/14
Bill of Materials for Order (...	1/6/14	1/20/14
Detailed Drawings	1/13/14	1/27/14
Assembled (R1)	1/27/14	2/7/14
Coding (R1)	2/3/14	2/14/14
Optimized Coding (R1)	2/3/14	3/28/14
Bill of Matierals for Order (...	2/7/14	2/25/14
Assembled (R2)	3/3/14	3/28/14
Coding (R2)	3/24/14	4/4/14
Optimized Coding (R2)	3/24/14	4/11/14
Practice with R1 and R2	4/4/14	4/22/14
Thesis Table of Contents a...	3/31/14	4/7/14
Experimental Protocol and...	3/31/14	4/14/14
Societal/Environmental Imp...	3/31/14	5/12/14
Thesis Draft Due	3/31/14	5/19/14
Business Plan	3/31/14	5/28/14
VEX Championship Compe...	4/23/14	4/25/14
Experimental Results	4/25/14	6/2/14
Senior Design Conference	5/8/14	5/8/14
Final Thesis	6/11/14	6/11/14

Figure C.1: Gantt Chart for the VEX Robotics project.

Appendix D

Budget and Sponsors

D.1 Budget

Table D.1: Categorized budget breakdown for the VEX Robotics project.

Item	**Amount**	**Description**
Robot Parts		
	$800	(2) VEXnet System Bundle (Cortex, joystick, VEXnet)
	$800	Aluminum
	$150	Wheels
	$850	(30) Motors
	$500	Pneumatics
	$160	Gears, Sprockets, Chain, Shafts
	$250	Sensors
	$100	Electronics (LCD)Wires
	$100	Fasteners
	$250	Batteries and Chargers
Software		
	$75	RobotC
	$50	Programming Cable
Tools		
	$60	Allen Wrenches/Snips, etc.
	$60	Cabinets and Parts Sorter
Registration		
	$100	Team Registration
	$750	World Championship Registration
Travel		
	$300	Transportaion (Car)
	$700	Lodging
Total	$6055	

D.2 Order Invoices

The following pages contains invoices for all orders placed through VEX Robotics for the team.

VEX Robotics, Inc.
1519 Int. 30 W.
Greenville, TX 75402
United States
+1 903 453 0802 phone
+1 214 722 1284 fax

INVOICE

Order # 11079638
Order Date: Nov 21, 2013

SOLD TO:	SHIP TO:
Irv Kalb 735 Glenborough Drive Mountain View, California, 94041 United States T: 650 964 6631	Irv Kalb 735 Glenborough Drive Mountain View, California, 94041 United States T: 650 964 6631
Payment Method	**Shipping Method:**
Name on the Card: Credit Card Type: Visa Credit Card Number: xxxx-8671 Expiration Date: 06/2014	UPS - UPS Ground (Total Shipping Charges $18.77)

Products	SKU	Price	QTY	Tax	Subtotal
VEXnet System Bundle	276-1604	**$399.99**	1	**$0.00**	**$399.99**
Programming Hardware Kit	276-2186	**$49.99**	1	**$0.00**	**$49.99**
ROBOTC for Cortex & PIC (1-seat)	276-1739	**$79.00**	1	**$0.00**	**$79.00**
2-Wire Motor 393 w/ Motor Controller 29	276-1668	**$24.98**	6	**$0.00**	**$149.88**
4" Omni-Directional Wheel (2-pack)	276-2185	**$24.99**	3	**$0.00**	**$74.97**
Smart Charger v2 w/ Power Cord Options	SMART-CHARGER-V2-G-276-2519-276-2520				
Charger					
1 x Smart Charger v2		**$16.99**	**1**	**$0.00**	**$16.99**
Power Cord					
1 x Battery Charger Power Cord - North America (Type A)		**$1.99**	**1**	**$0.00**	**$1.99**
7.2V Robot Battery NiMH 3000mAh	276-1491	**$29.99**	2	**$0.00**	**$59.98**
VEXnet Joystick Power Adapter	276-1701	**$12.99**	1	**$0.00**	**$12.99**
Aluminum C-Channel 1x2x1x35 (6-pack)	276-2289	**$49.99**	1	**$0.00**	**$49.99**
C-Channel Coupler Gusset (8-pack)	276-2575	**$19.99**	1	**$0.00**	**$19.99**
Aluminum Plate 25x5 (6-pack)	276-2311	**$34.99**	1	**$0.00**	**$34.99**
Drive Shaft 12" (4-pack)	276-1149	**$8.96**	1	**$0.00**	**$8.96**
Bearing Flat, Delrin (10-pack)	276-1209	**$4.99**	3	**$0.00**	**$14.97**
Shaft Collar (16-pack)	276-2010	**$10.49**	2	**$0.00**	**$20.98**
Screw 8-32 x 0.500" (100-pack)	275-1004	**$7.49**	1	**$0.00**	**$7.49**
Nut 8-32 Keps (100-pack)	275-1026	**$2.99**	1	**$0.00**	**$2.99**
Nylon Spacer Variety Pack	275-1066	**$4.95**	1	**$0.00**	**$4.95**

Subtotal:	**$1,011.09**
Shipping & Handling:	**$18.77**
Grand Total:	**$1,029.86**
Total Paid:	**$1,029.86**
Total Due:	**$0.00**

VEX Robotics, Inc.
1519 Int. 30 W.
Greenville, TX 75402
United States
+1 903 453 0802 phone
+1 214 722 1284 fax

INVOICE

Order # 11086997
Order Date: Jan 20, 2014

SOLD TO:	SHIP TO:
Irv Kalb 735 Glenborough Drive Mountain View, California, 94041 United States T: 650 964 6631	Irv Kalb 735 Glenborough Drive Mountain View, California, 94041 United States T: 650 964 6631

Payment Method	**Shipping Method:**
Name on the Card: Credit Card Type: Visa Credit Card Number: xxxx-8671 Expiration Date: 06/2014	UPS - UPS Three-Day Select (Total Shipping Charges $42.30)

Products	SKU	Price	QTY	Tax	Subtotal
2-Wire Motor 393 w/ Motor Controller 29	276-1668	**$24.98**	4	**$0.00**	**$99.92**
2-Wire Motor 393	276-2177	**$14.99**	2	**$0.00**	**$29.98**
3-Wire "Y"-Cable 6" (2-pack)	276-1423	**$14.95**	1	**$0.00**	**$14.95**
2-Wire Extension Cable 36" (4 Pack)	276-1430	**$9.99**	1	**$0.00**	**$9.99**
Bearing Flat, Delrin (10-pack)	276-1209	**$4.99**	6	**$0.00**	**$29.94**
Shaft Collar (16-pack)	276-2010	**$10.49**	2	**$0.00**	**$20.98**
High Strength Sprocket & Chain Kit	276-2252	**$39.99**	1	**$0.00**	**$39.99**
High Strength Gear Kit	276-2250	**$29.99**	1	**$0.00**	**$29.99**
Potentiometer (2-pack)	276-2216	**$12.99**	1	**$0.00**	**$12.99**
Motor 393 Integrated Encoder Module (2-pack)	276-1321	**$29.99**	1	**$0.00**	**$29.99**
Standoff 2.00" (10-pack)	275-1018	**$7.95**	1	**$0.00**	**$7.95**
Aluminum C-Channel 1x5x1x25 (6-pack)	276-2290	**$49.99**	1	**$0.00**	**$49.99**
Aluminum C-Channel 1x2x1x35 (6-pack)	276-2289	**$49.99**	2	**$0.00**	**$99.98**
Drive Shaft 12" (4-pack)	276-1149	**$8.96**	1	**$0.00**	**$8.96**
Screw 8-32 x 0.500" (100-pack)	275-1004	**$7.49**	1	**$0.00**	**$7.49**
Screw 8-32 x 0.250" (100-pack)	275-1002	**$7.49**	1	**$0.00**	**$7.49**
Screw 8-32 x 2.000" (25-pack)	275-1012	**$7.49**	1	**$0.00**	**$7.49**
Nut 8-32 Keps (100-pack)	275-1026	**$2.99**	1	**$0.00**	**$2.99**
Nut 8-32 Nylock (100-pack)	275-1027	**$3.99**	1	**$0.00**	**$3.99**
				Subtotal:	**$515.05**
				Shipping & Handling:	**$42.30**
				Grand Total:	**$557.35**
				Total Paid:	**$557.35**
				Total Due:	**$0.00**

VEX Robotics, Inc.
1519 Int. 30 W.
Greenville, TX 75402
United States
+1 903 453 0802 phone
+1 214 722 1284 fax

INVOICE

Order # 11093366
Order Date: Feb 24, 2014

SOLD TO:	SHIP TO:
Irv Kalb 735 Glenborough Drive Mountain View, California, 94041 United States T: 650 964 6631	Irv Kalb 735 Glenborough Drive Mountain View, California, 94041 United States T: 650 964 6631
Payment Method	**Shipping Method:**
Name on the Card: Credit Card Type: Visa Credit Card Number: xxxx-8671 Expiration Date: 06/2014	UPS - UPS Ground (Total Shipping Charges $25.36)

Products	SKU	Price	QTY	Tax	Subtotal
2.75" Omni Directional Wheel - Double Roller (2-pack)	276-1902	**$19.99**	1	**$0.00**	**$19.99**
2-Wire Motor 393	276-2177	**$14.99**	2	**$0.00**	**$29.98**
2-Wire Motor 393 w/ Motor Controller 29	276-1668	**$24.98**	10	**$0.00**	**$249.80**
VEXnet System Bundle	276-1604	**$399.99**	1	**$0.00**	**$399.99**
LCD Display	276-2273	**$49.99**	1	**$0.00**	**$49.99**
Serial Y-Cable	276-1579	**$7.99**	2	**$0.00**	**$15.98**
2-Wire Extension Cable 36" (4 Pack)	276-1430	**$7.99**	3	**$0.00**	**$23.97**
3-Wire Extension Cable 36" (4-pack)	276-1976	**$7.99**	1	**$0.00**	**$7.99**
3-Wire "Y"-Cable 6" (2-pack)	276-1423	**$4.99**	1	**$0.00**	**$4.99**
Potentiometer (2-pack)	276-2216	**$12.99**	1	**$0.00**	**$12.99**
Nylon Spacer Variety Pack	275-1066	**$4.95**	4	**$0.00**	**$19.80**
Screw 8-32 x 2.000" (25-pack)	275-1012	**$7.49**	1	**$0.00**	**$7.49**
Screw 8-32 x 0.250" (100-pack)	275-1002	**$7.49**	1	**$0.00**	**$7.49**
Screw 8-32 x 0.500" (100-pack)	275-1004	**$7.49**	1	**$0.00**	**$7.49**
Nut 8-32 Keps (100-pack)	275-1026	**$2.99**	1	**$0.00**	**$2.99**
Standoff 1.50" (10-pack)	275-1017	**$7.95**	1	**$0.00**	**$7.95**
Aluminum C-Channel 1x5x1x35 (6-pack)	276-2298	**$44.99**	2	**$0.00**	**$89.98**
Aluminum C-Channel 1x2x1x35 (6-pack)	276-2289	**$34.99**	1	**$0.00**	**$34.99**
Linear Motion Kit	276-1926	**$24.99**	3	**$0.00**	**$74.97**
High Strength Gear Kit	276-2250	**$29.99**	1	**$0.00**	**$29.99**
Drive Shaft 12" (4-pack)	276-1149	**$8.96**	2	**$0.00**	**$17.92**
Bearing Flat, Delrin (10-pack)	276-1209	**$4.99**	4	**$0.00**	**$19.96**
Shaft Collar (16-pack)	276-2010	**$10.49**	2	**$0.00**	**$20.98**
Rubber Link (4-pack)	275-1029	**$7.49**	1	**$0.00**	**$7.49**
Aluminum Bar 1x25 (16-pack)	276-2307	**$29.99**	1	**$0.00**	**$29.99**

Subtotal:	**$1,195.15**
Shipping & Handling:	**$25.36**
Grand Total:	**$1,220.51**
Total Paid:	**$1,220.51**
Total Due:	**$0.00**

VEX Robotics, Inc.
1519 Int. 30 W.
Greenville, TX 75402
United States
+1 903 453 0802 phone
+1 214 722 1284 fax

INVOICE

Order # 11095662
Order Date: Mar 16, 2014

SOLD TO:

Irv Kalb
735 Glenborough Drive
Mountain View, California, 94041
United States
T: 650 964 6631

SHIP TO:

Irv Kalb
735 Glenborough Drive
Mountain View, California, 94041
United States
T: 650 964 6631

Payment Method

Name on the Card:
Credit Card Type: Visa
Credit Card Number: xxxx-8671
Expiration Date: 06/2014

Shipping Method:

UPS - UPS Ground
(Total Shipping Charges $18.10)

Products	SKU	Price	QTY	Tax	Subtotal
Pneumatics Kit 1 - Single Acting Cylinders	275-0274	**$179.95**	2	**$0.00**	**$359.90**
4" Omni-Directional Wheel (2-pack)	276-2185	**$24.99**	2	**$0.00**	**$49.98**
Motor 393 Integrated Encoder Module (2-pack)	276-1321	**$29.99**	1	**$0.00**	**$29.99**
VEXnet Backup Battery Holder	276-2243	**$9.99**	1	**$0.00**	**$9.99**
High Strength Sprocket & Chain Kit	276-2252	**$39.99**	3	**$0.00**	**$119.97**
Hinge (2-pack)	275-1272	**$9.99**	2	**$0.00**	**$19.98**
Rubber Link (4-pack)	275-1029	**$7.49**	4	**$0.00**	**$29.96**
Aluminum Plate 25x5 (6-pack)	276-2311	**$24.99**	1	**$0.00**	**$24.99**
6" Wheel Leg (4-pack)	276-2218	**$24.99**	1	**$0.00**	**$24.99**
Bearing Flat, Delrin (10-pack)	276-1209	**$4.99**	4	**$0.00**	**$19.96**
Smart Charger v2 w/ Power Cord Options	SMART-CHARGER-V2-G-276-2519-276-2520				
Charger					
1 x Smart Charger v2		**$16.99**	**1**	**$0.00**	**$16.99**
Power Cord					
1 x Battery Charger Power Cord - North America (Type A)		**$1.99**	**1**	**$0.00**	**$1.99**
7.2V Robot Battery NiMH 3000mAh	276-1491	**$29.99**	2	**$0.00**	**$59.98**
				Subtotal:	**$768.67**
				Shipping & Handling:	**$18.10**
				Grand Total:	**$786.77**
				Total Paid:	**$786.77**
				Total Due:	**$0.00**

VEX Robotics, Inc.
1519 Int. 30 W.
Greenville, TX 75402
United States
+1 903 453 0802 phone
+1 214 722 1284 fax

INVOICE

Order # 11097493
Order Date: Apr 6, 2014

SOLD TO:	SHIP TO:
Irv Kalb 735 Glenborough Drive Mountain View, California, 94041 United States T: 650 964 6631	Irv Kalb 735 Glenborough Drive Mountain View, California, 94041 United States T: 650 964 6631

Payment Method	**Shipping Method:**
Name on the Card: Credit Card Type: Visa Credit Card Number: xxxx-0130 Expiration Date: 06/2014	UPS - UPS Ground (Total Shipping Charges $16.41)

Products	SKU	Price	QTY	Tax	Subtotal
LCD Display	276-2273	**$49.99**	1	**$0.00**	**$49.99**
Screw 8-32 x 2.000" (25-pack)	275-1012	**$7.49**	1	**$0.00**	**$7.49**
Aluminum C-Channel 1x2x1x25 (6-pack)	276-2288	**$29.99**	1	**$0.00**	**$29.99**
Pneumatics Kit 2 - Double Acting Cylinders	275-0276	**$229.95**	2	**$0.00**	**$459.90**
Nut 8-32 Keps (100-pack)	275-1026	**$2.99**	2	**$0.00**	**$5.98**
Rubber Link (4-pack)	275-1029	**$7.49**	1	**$0.00**	**$7.49**
Smart Charger v2 w/ Power Cord Options	SMART-CHARGER-V2-G-276-2519-276-2520				
Charger					
1 x Smart Charger v2		**$16.99**	**2**	**$0.00**	**$33.98**
Power Cord					
1 x Battery Charger Power Cord - North America (Type A)		**$1.99**	**2**	**$0.00**	**$3.98**
7.2V Robot Battery NiMH 3000mAh	276-1491	**$29.99**	4	**$0.00**	**$119.96**
Standoff Pack	276-2013	**$15.99**	1	**$0.00**	**$15.99**
				Subtotal:	**$734.75**
				Shipping & Handling:	**$16.41**
				Grand Total:	**$751.16**
				Total Paid:	**$751.16**
				Total Due:	**$0.00**

D.3 Sponsors

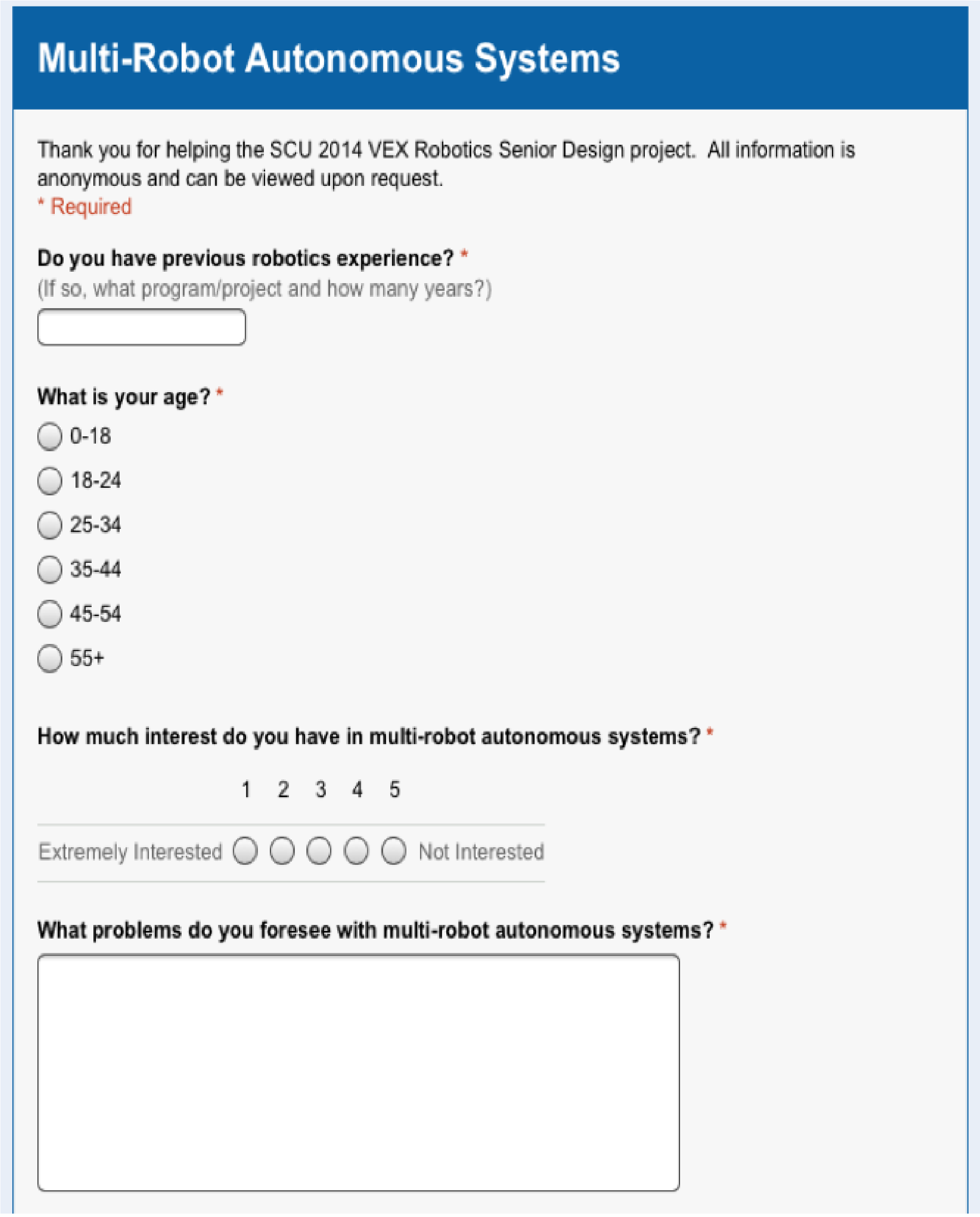
Table D.2: Organizations and corporations who provided support for the VEX Robotics team.

Sponsor or Partner	**Involvement**
Santa Clara University School of Engineering	$2500 Monetary Donation
Santa Clara University Mechanical Engineering Department	$2500 Monetary Donation
NASA Robotics Alliance Project	$1000 Monetary Donation
SMaRT Education	$̃500 Donation in Kind
Bellarmine College Preparatory Robotics Team	Regulation Competition Field Access
Green MacHHHHine Robotics Team	Regulation Competition Field Access

Appendix E

Copy of Questionnaire

The following pages contains a copy of the online questionnaire that was sent out to over 200 individuals.



Multi-Robot Autonomous Systems

Thank you for helping the SCU 2014 VEX Robotics Senior Design project. All information is anonymous and can be viewed upon request.

* Required

Do you have previous robotics experience? *

(If so, what program/project and how many years?)

What is your age? *

○ 0-18

○ 18-24

○ 25-34

○ 35-44

○ 45-54

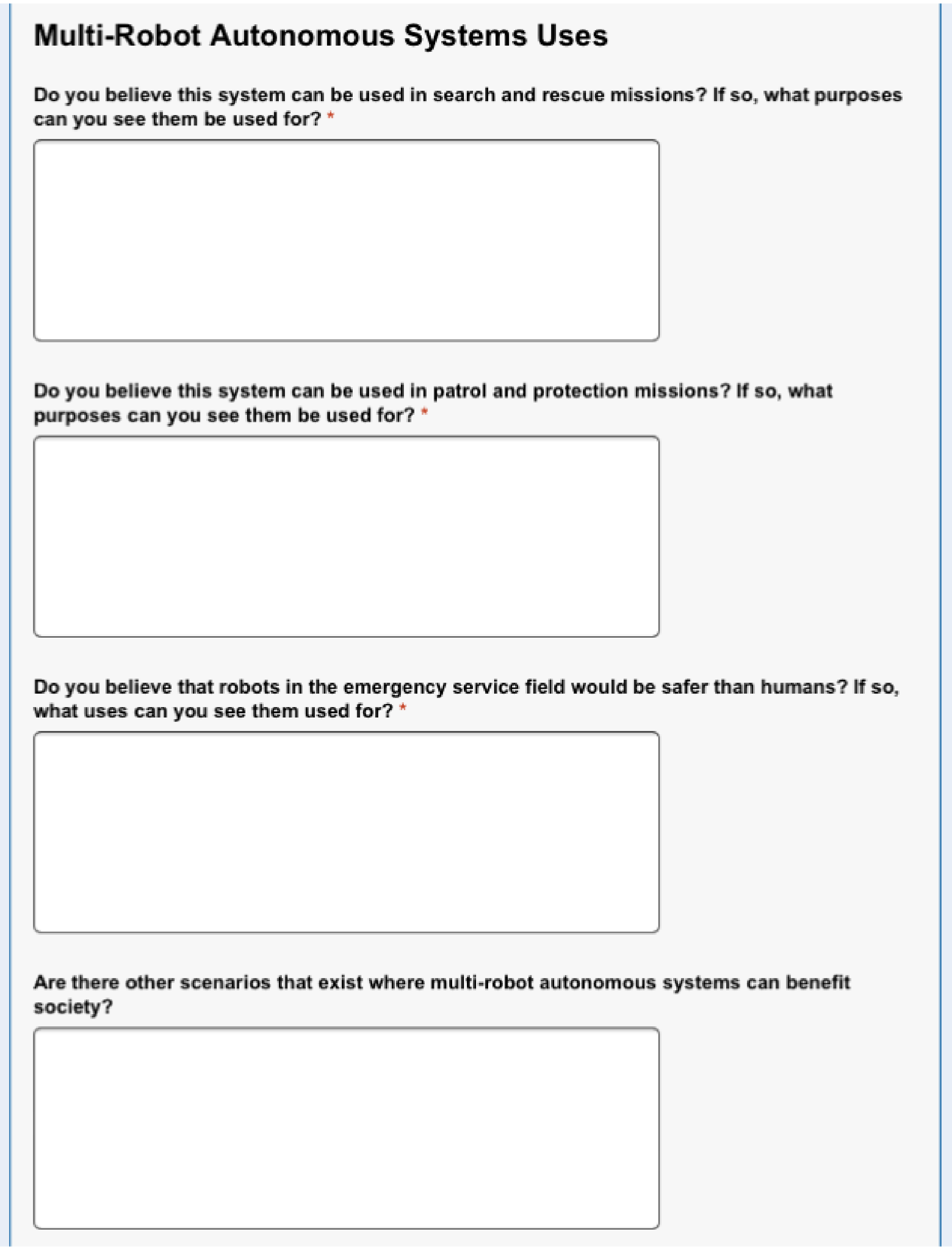
○ 55+

How much interest do you have in multi-robot autonomous systems? *

	1	2	3	4	5	
Extremely Interested	○	○	○	○	○	Not Interested

What problems do you foresee with multi-robot autonomous systems? *

Figure E.1: First page of the general questionnaire administered to anonymous individuals and industry professionals.



Multi-Robot Autonomous Systems Uses

Do you believe this system can be used in search and rescue missions? If so, what purposes can you see them be used for? *

Do you believe this system can be used in patrol and protection missions? If so, what purposes can you see them be used for? *

Do you believe that robots in the emergency service field would be safer than humans? If so, what uses can you see them used for? *

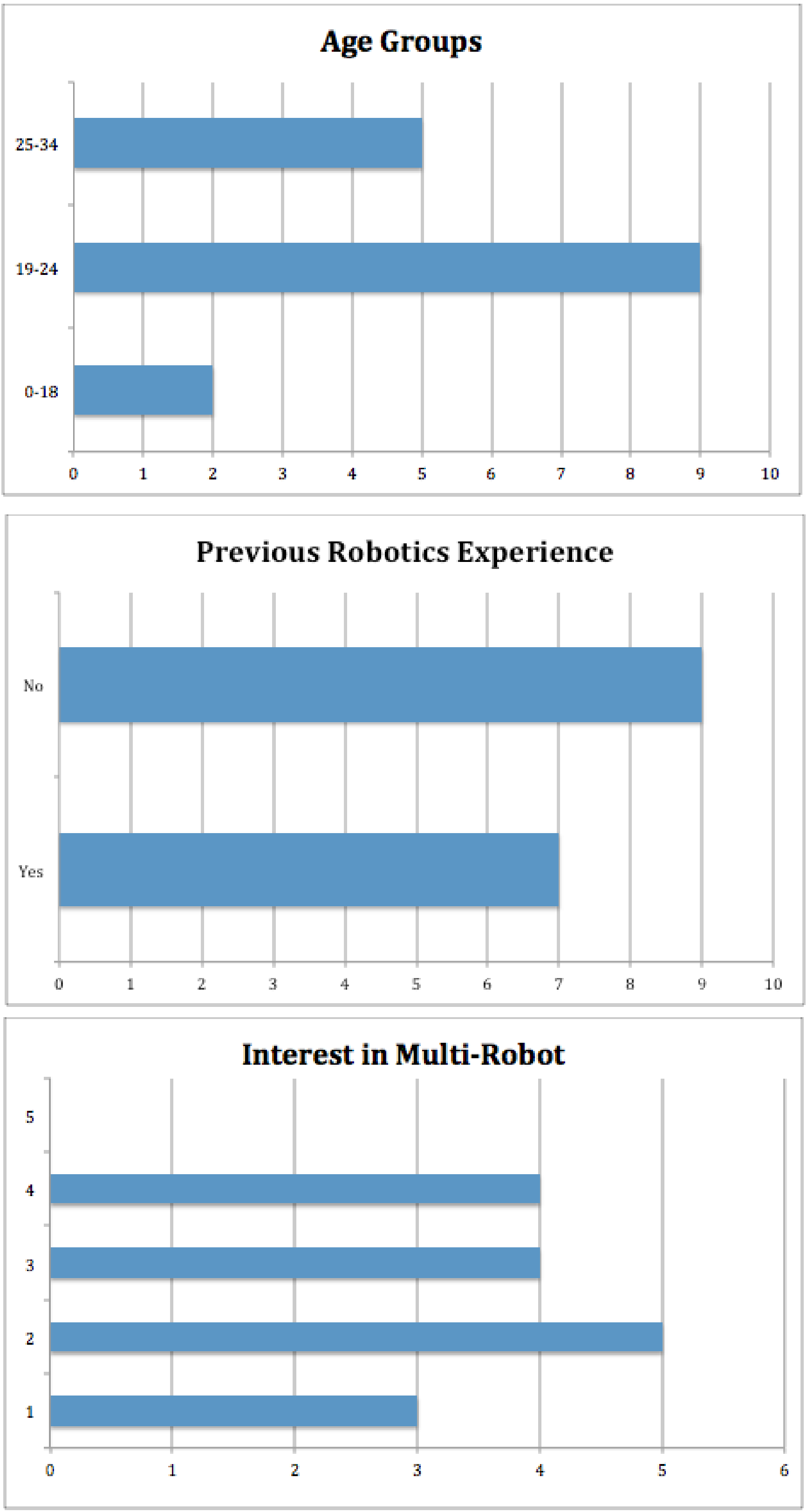
Are there other scenarios that exist where multi-robot autonomous systems can benefit society?

Figure E.2: Second page of the general questionnaire administered to anonymous individuals and industry professionals.

Appendix F

Organized Feedback

The following pages contains the questionnaire responses organized in graphical form.



Age Groups

Previous Robotics Experience

Interest in Multi-Robot

Figure F.1: Processed background data of individuals taking survey.

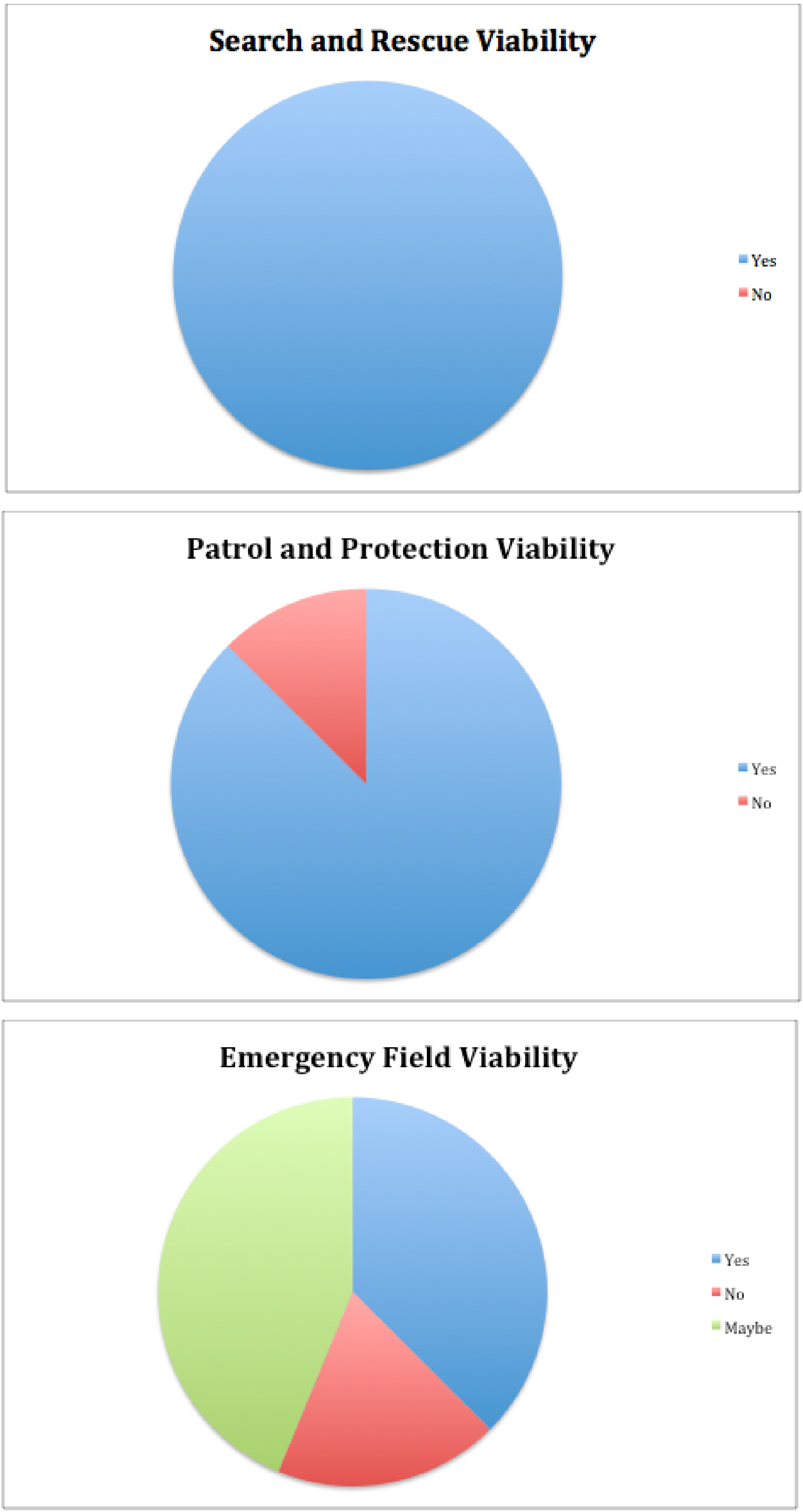


Figure F.2: Processed data for the viability of various approaches for multi-robot control system based on survey responses.

Appendix G

Competition Strategy

G.1 Autonomous Strategies

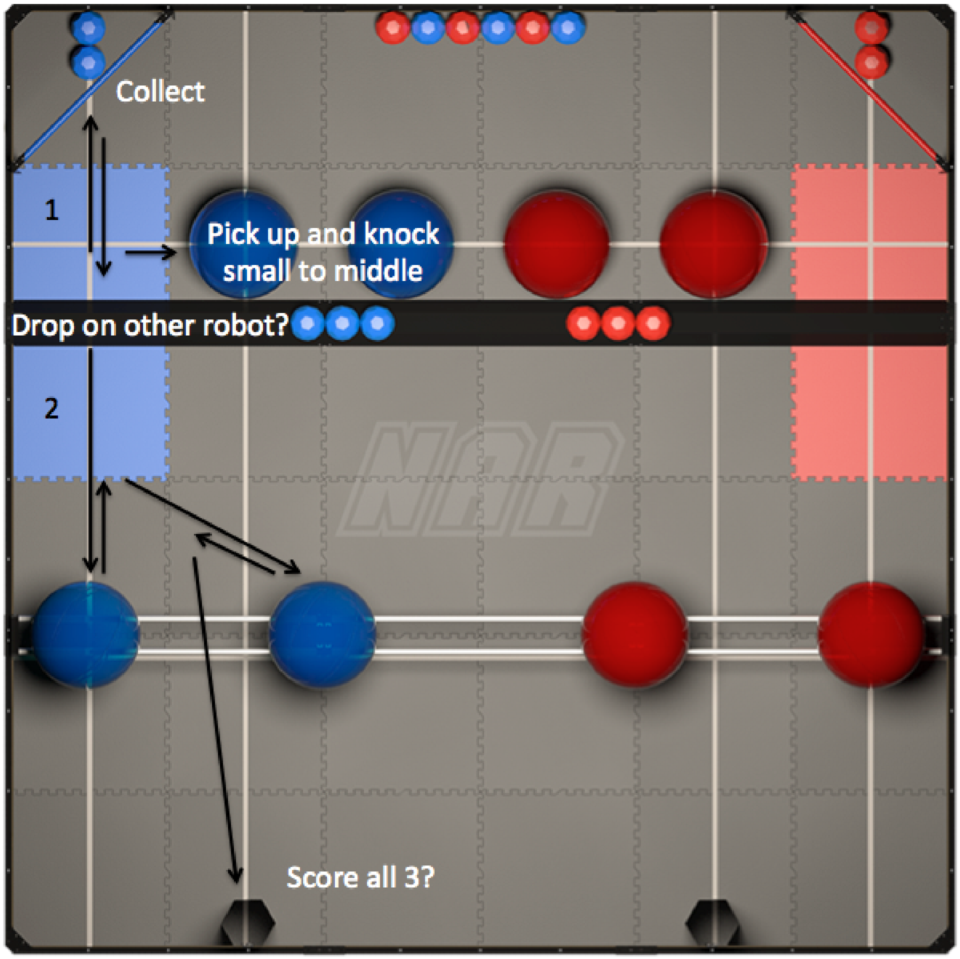


Figure G.1: Initial brainstormed autonomous strategy.

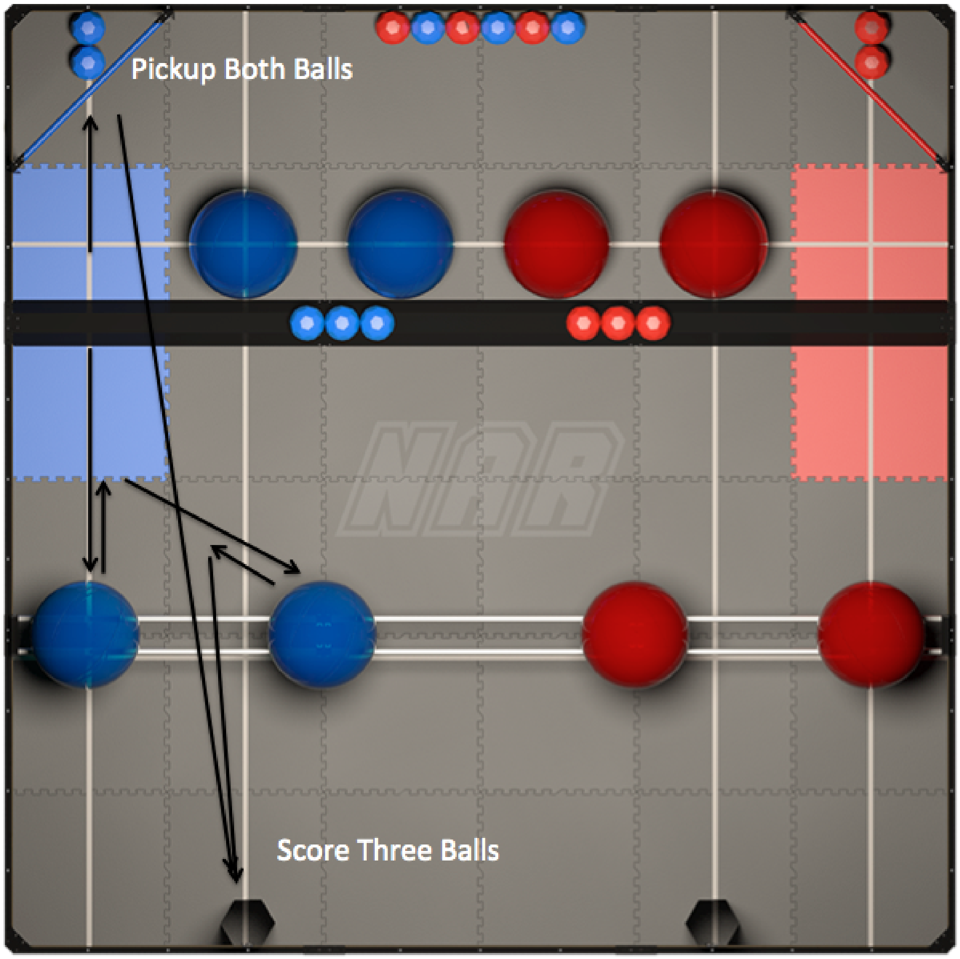


Figure G.2: Iteration of initial brainstormed autonomous strategy.

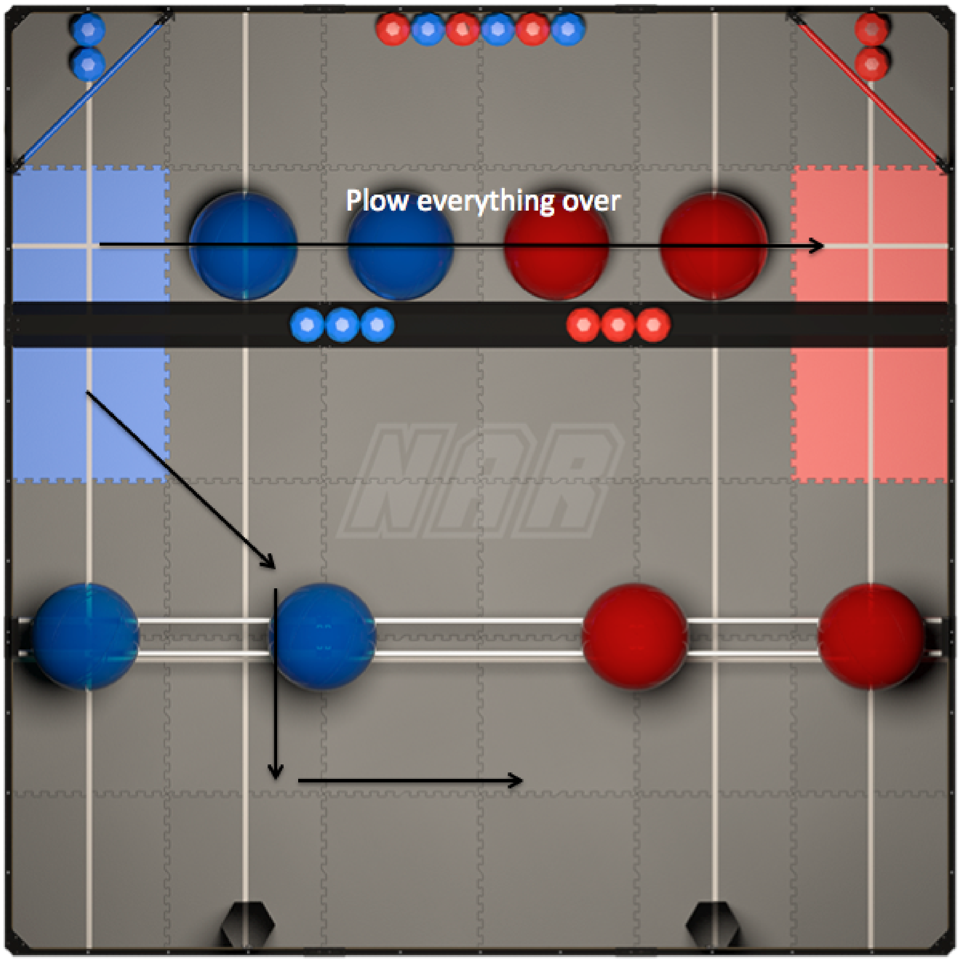


Figure G.3: Final autonomous stragety for robots.

G.2 Teleoperated Strategies

- One defense robot (larger and heavier robot), one offense robot (smaller and faster robot)
- Ball launching robot (able to be offense and defense)
- Two offensive scoring robots
- Two purely defensive robots

Appendix H

2D Drawings

The following pages contain 2D drawings of various structural components.

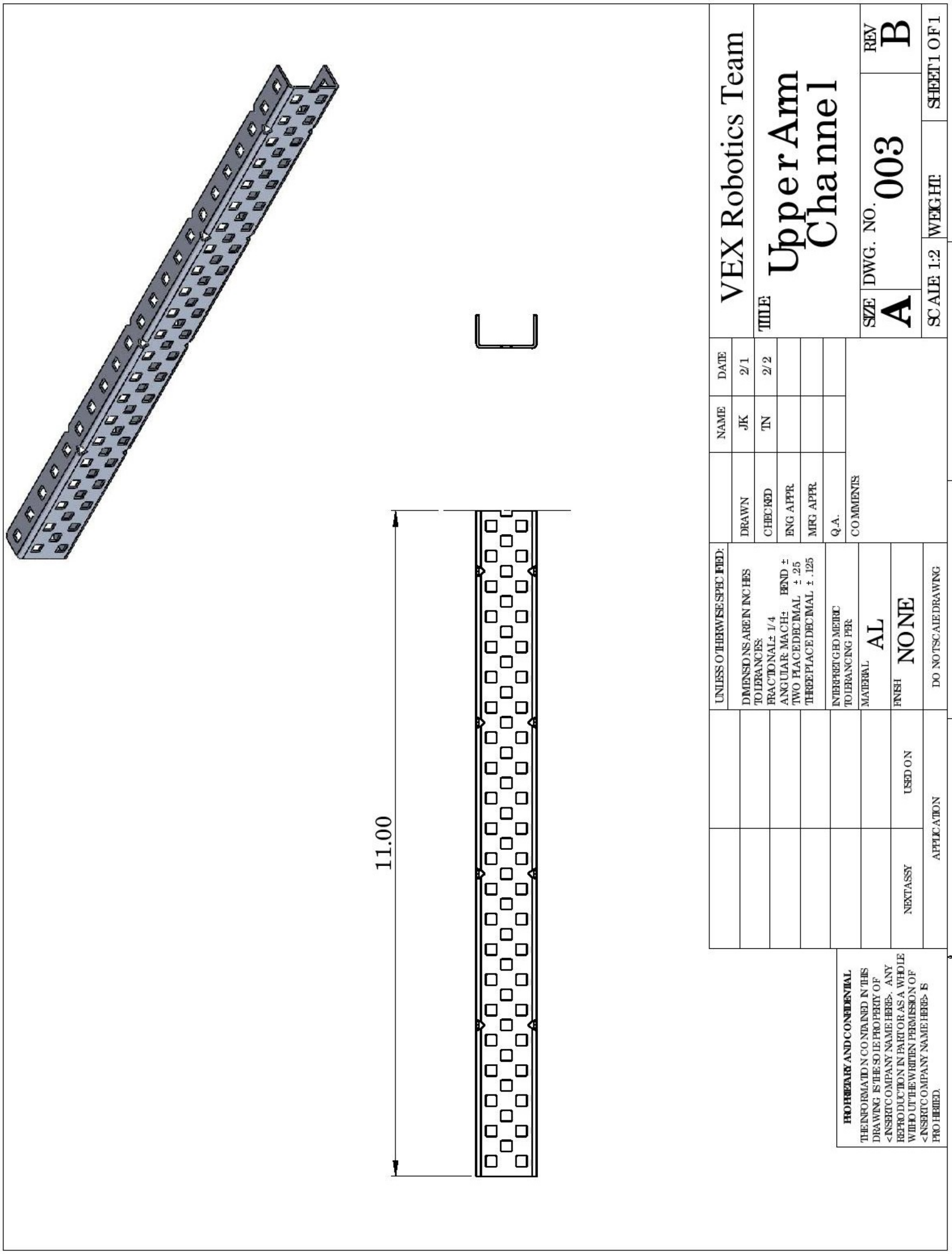

11.00
VEX Robotics Team
TITLE:
Upper Arm Channel
SIZE
A
DWG. NO.
003
REV
B
SCALE 1:2
WEIGHT:
SHEET 1 OF 1
NAME
DATE
DRAWN
JK
2/1
CHECKED
TN
2/2
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL± 1/4
ANGULAR: MACH± BEND ±
TWO PLACE DECIMAL ± .25
THREE PLACE DECIMAL ± .125
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
AL
FINISH
NONE
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
5
4
3
2
1

Figure H.1: Drawing for the upper arm channel.

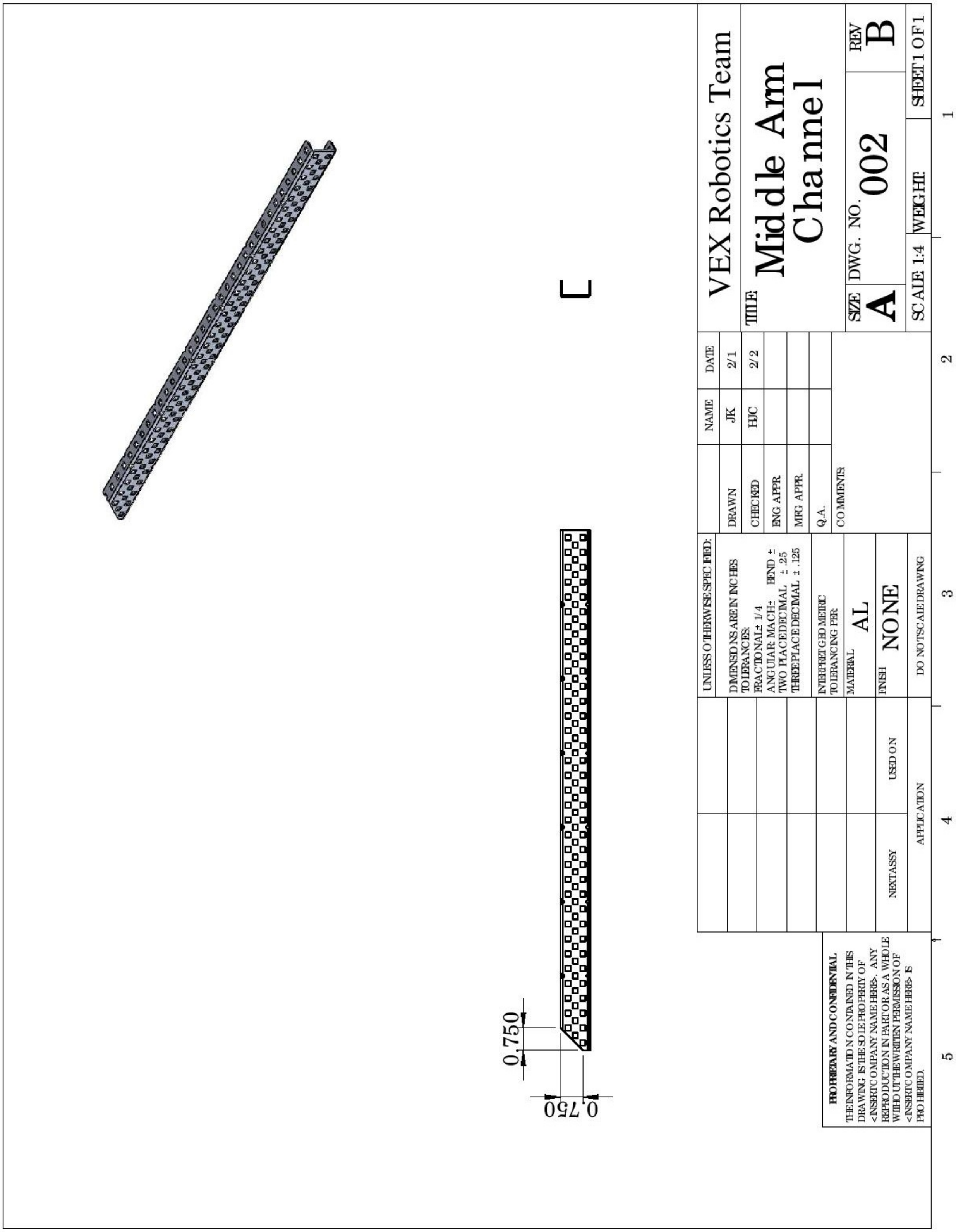


Figure H.2: Drawing for the middle arm channel with cut.

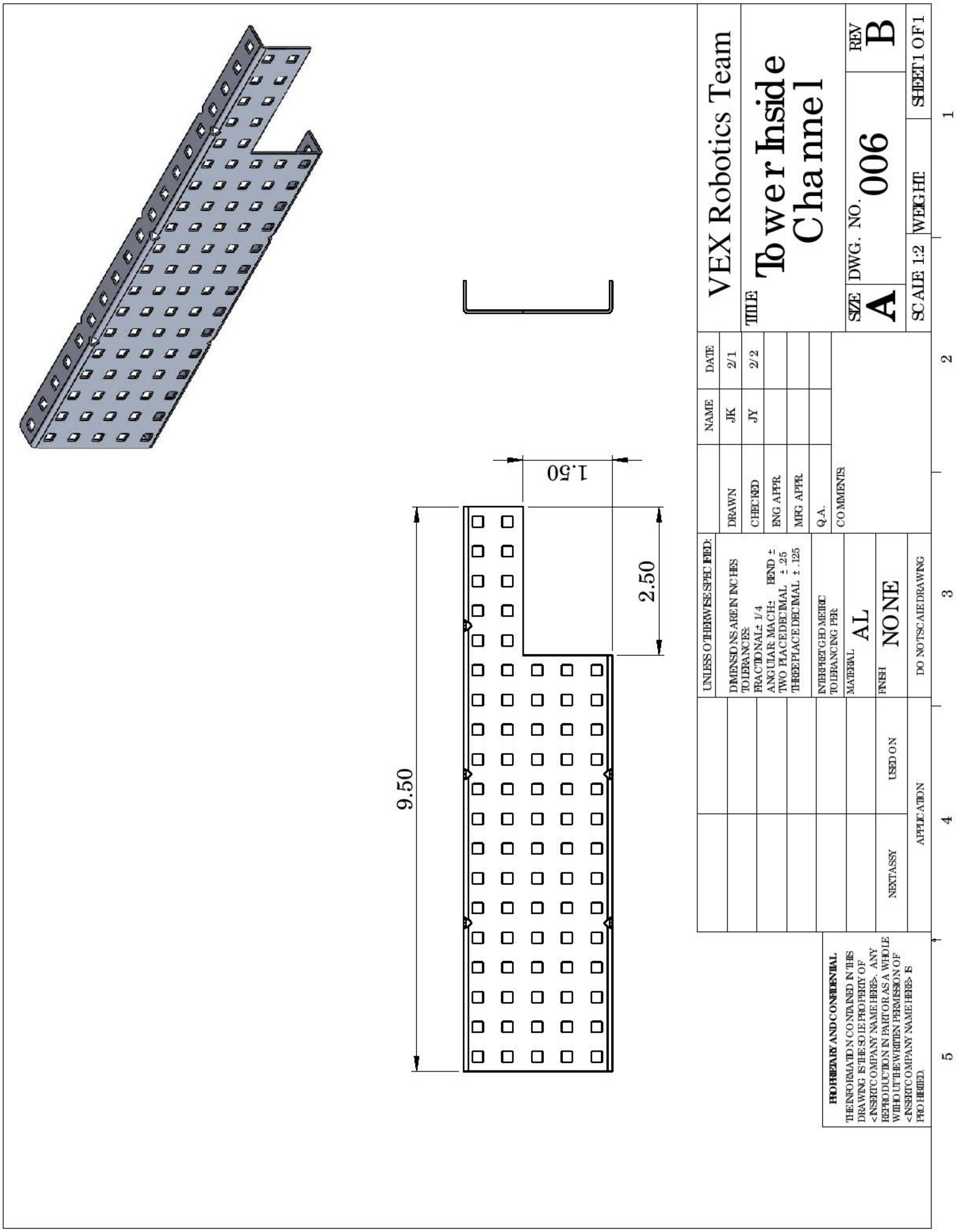


Figure H.3: Drawing for the inside channel of the tower structure with cut.

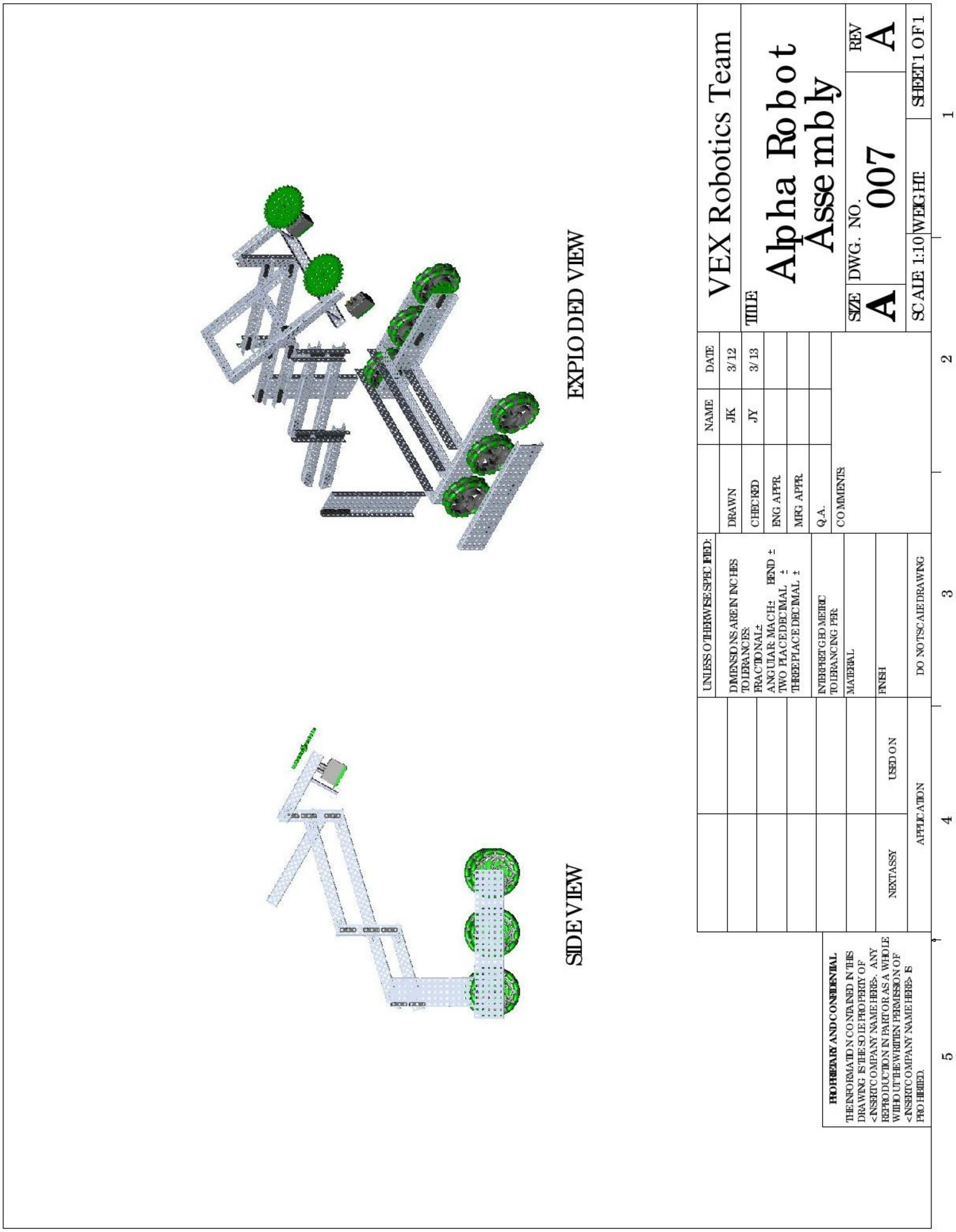


Figure H.4: Drawing for complete robot assembly.

Appendix I

3D Models

I.1 Robot

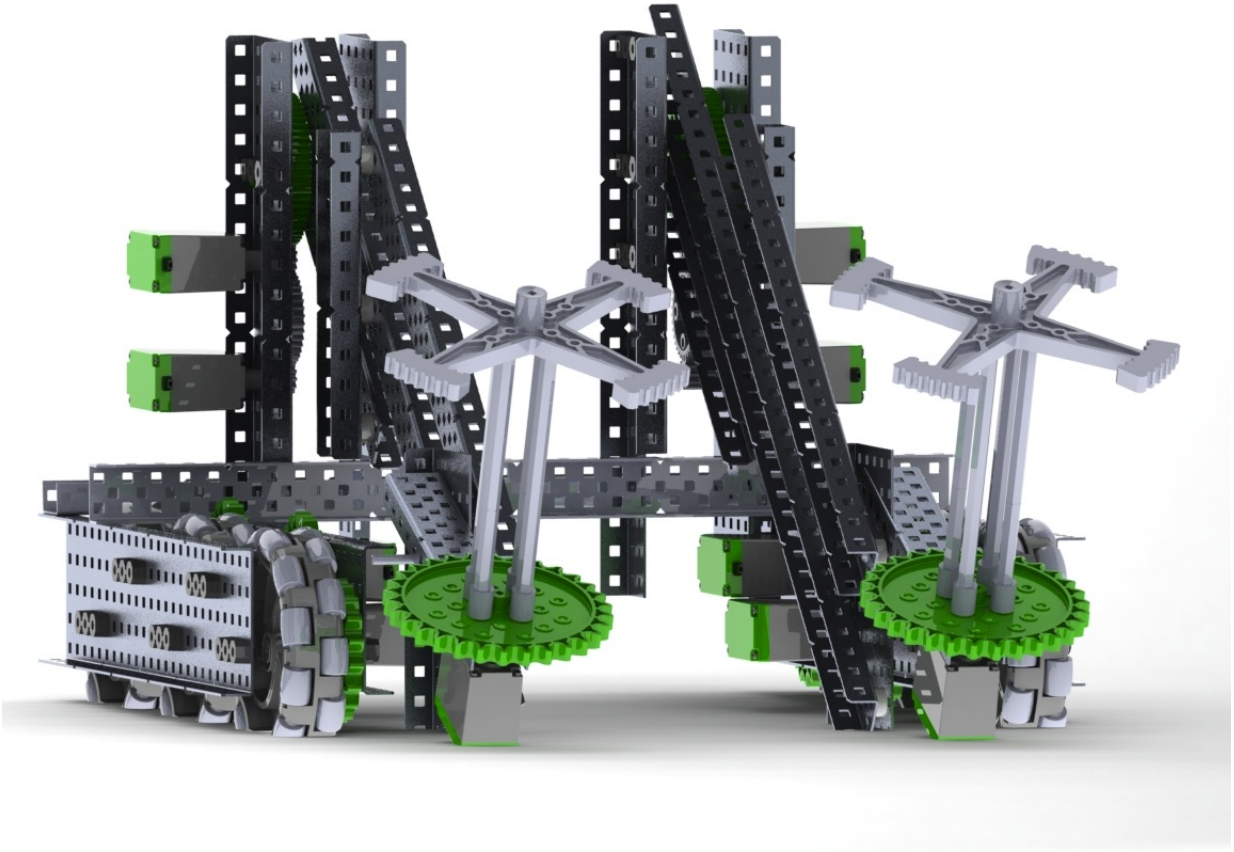


Figure I.1: Solidworks rendering of the front side of the robot.

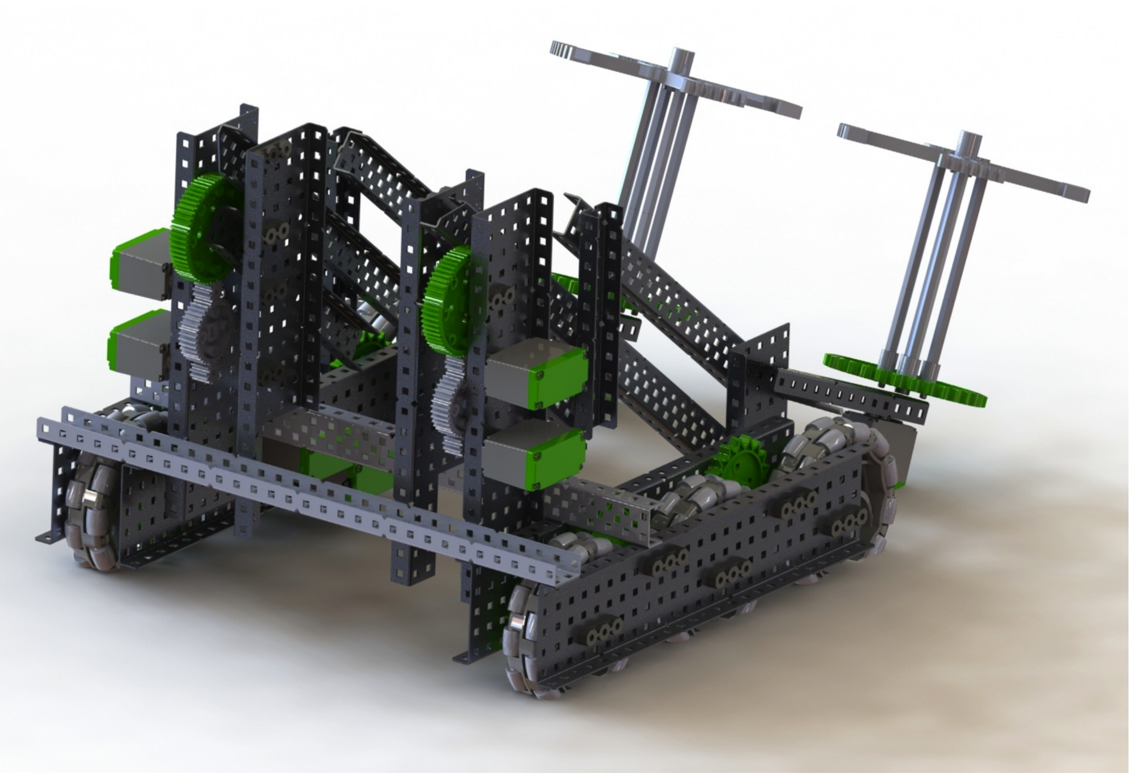


Figure I.2: Solidworks rendering of the back side of the robot.

I.2 Chassis

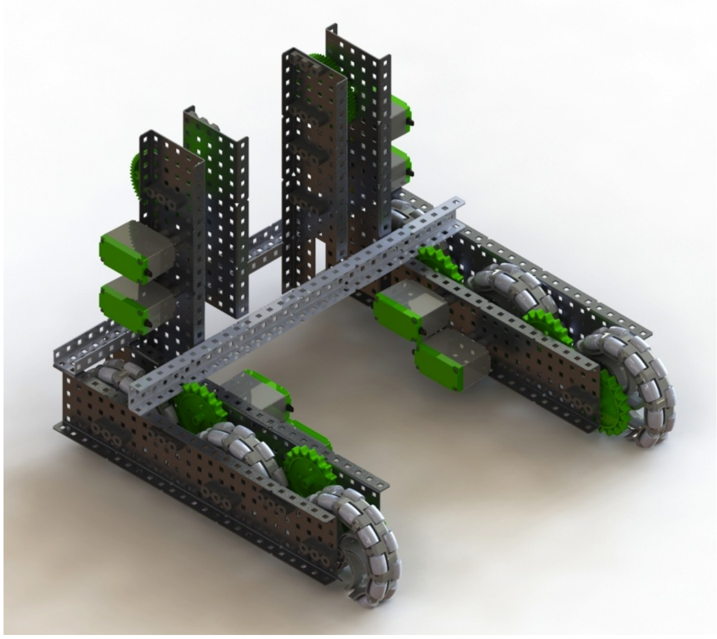


Figure I.3: Solidworks rendering of the chassis with the tower structure.

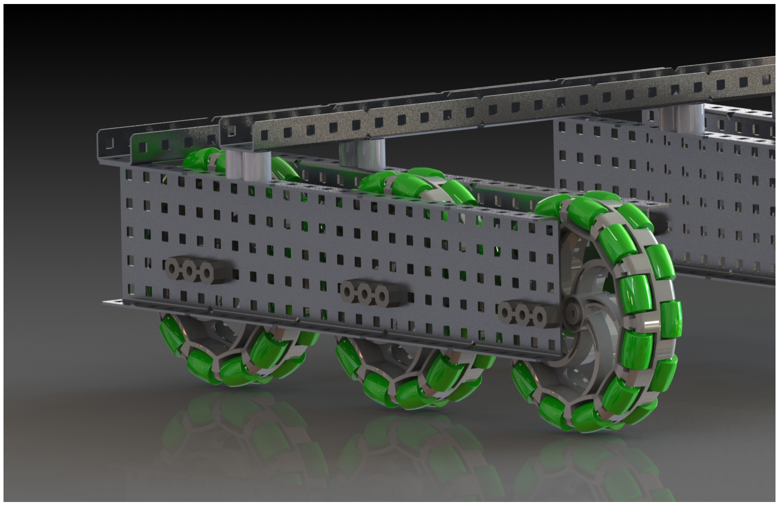


Figure I.4: Solidworks rendering of one side of the chassis.

I.3 Drive Train

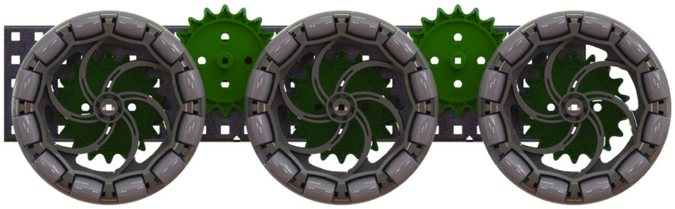


Figure I.5: Solidworks rendering of a side view of the drive train.

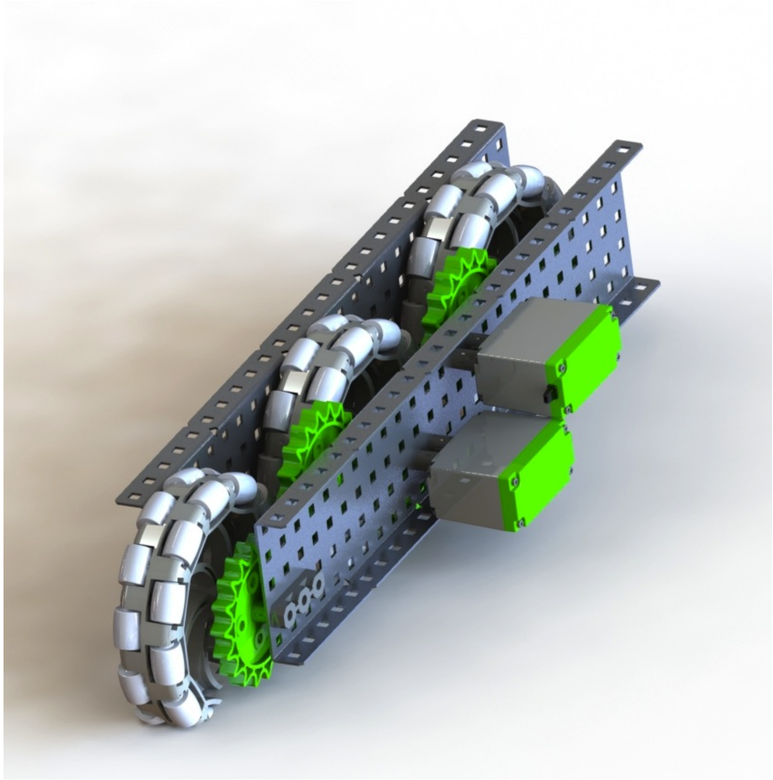


Figure I.6: Isometric Solidworks rendering of the drive train with motors mounted.

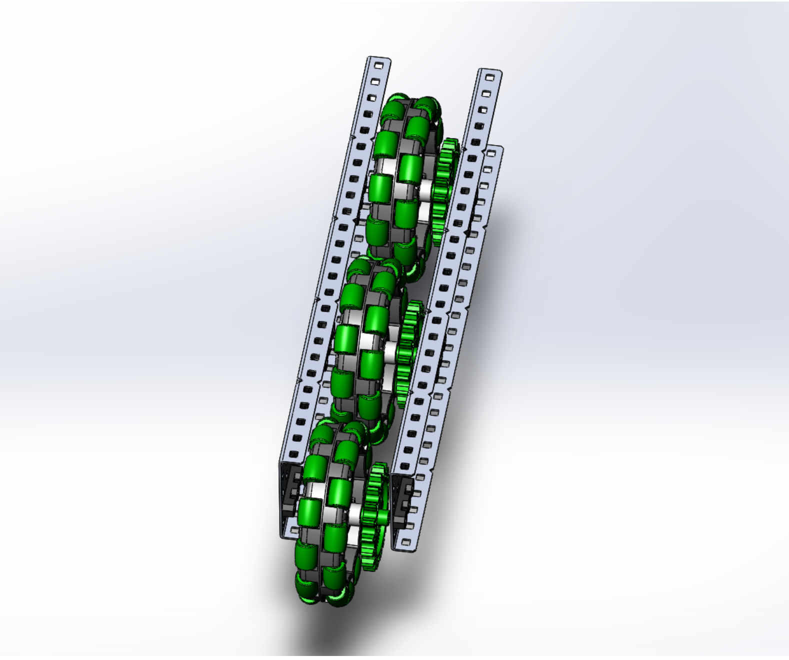


Figure I.7: Solidworks rendering of the top view of one side of the drive train.

I.4 Intake

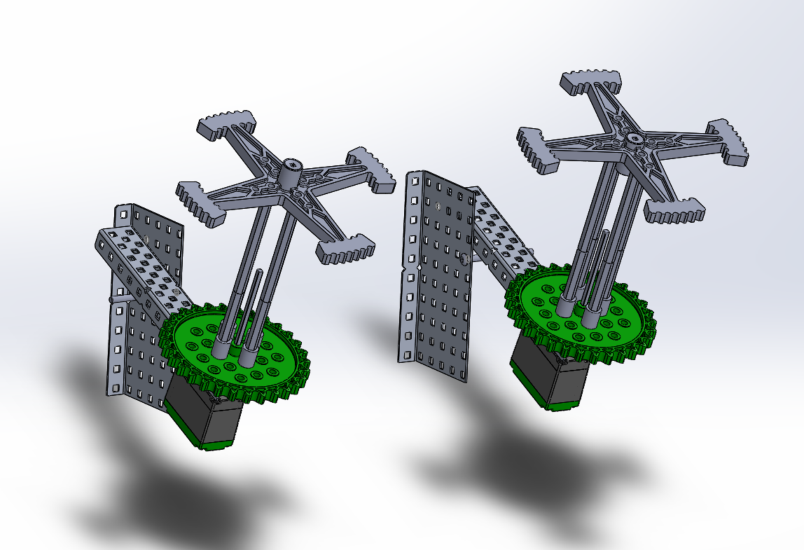


Figure I.8: Solidworks rendering of the intake system.

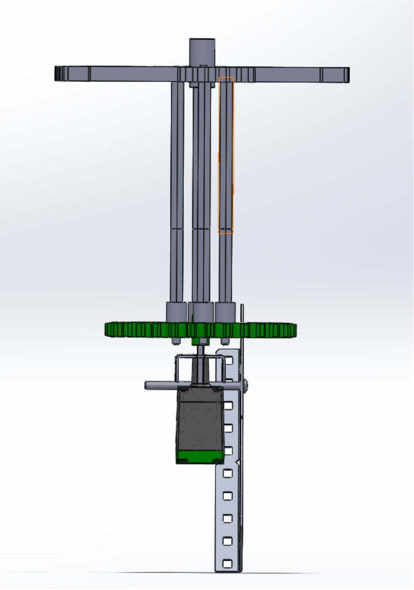


Figure I.9: Solidworks rendering of a front view of one half of the intake system.

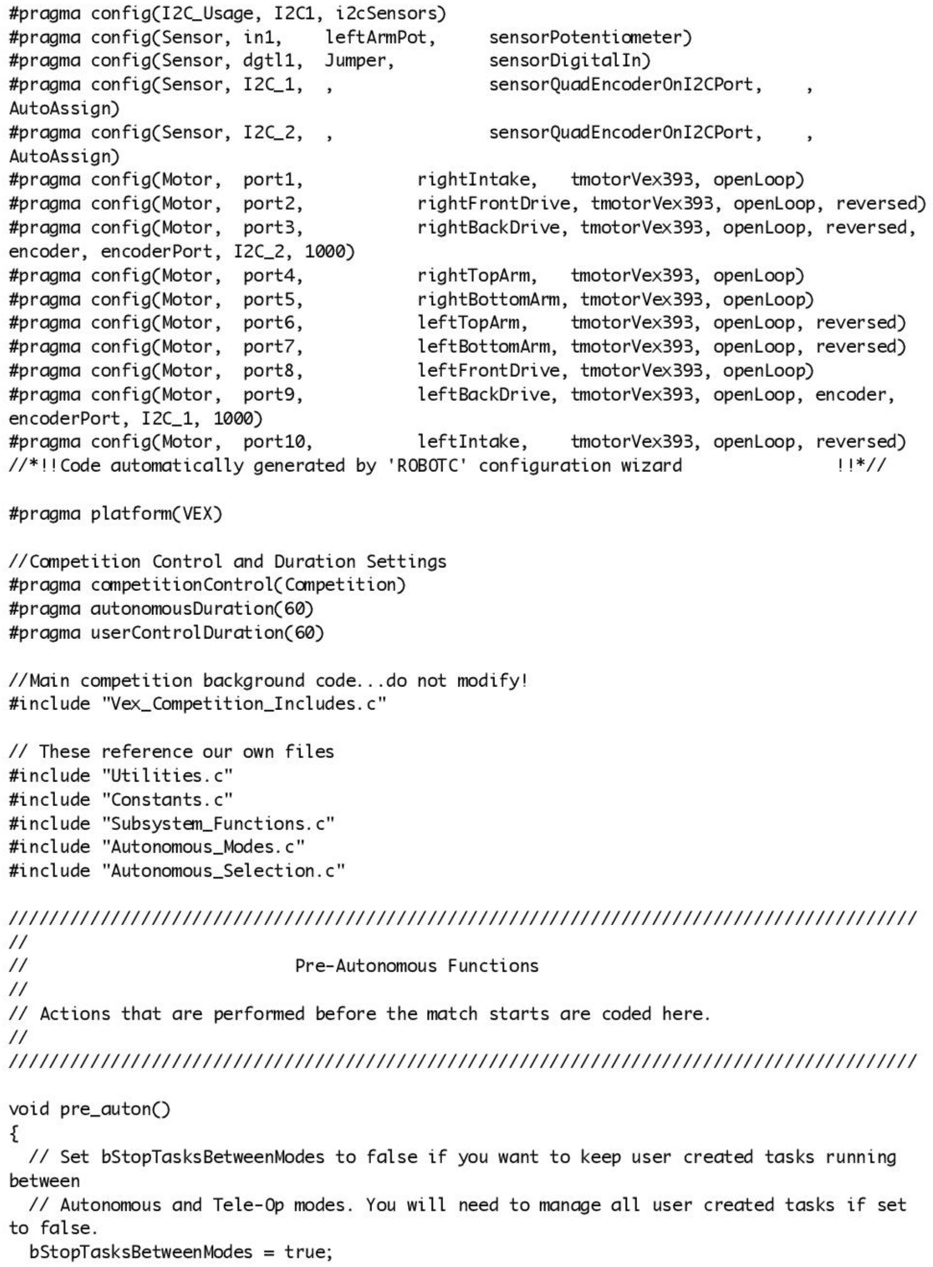
Appendix J

Robot Code

The following pages contains the code for both robots in the following order:

1. Competition.c
2. Subsystem_Functions.c
3. Utilities.c
4. Constants.c
5. Autonomous_Selection.c
6. Autonomous_Modes.c

J.1 Competition.c



```
#pragma config(I2C_Usage, I2C1, i2cSensors)
#pragma config(Sensor, in1,    leftArmPot,     sensorPotentiometer)
#pragma config(Sensor, dgtl1,  Jumper,         sensorDigitalIn)
#pragma config(Sensor, I2C_1,  ,               sensorQuadEncoderOnI2CPort,    ,
AutoAssign)
#pragma config(Sensor, I2C_2,  ,               sensorQuadEncoderOnI2CPort,    ,
AutoAssign)
#pragma config(Motor,  port1,           rightIntake,   tmotorVex393, openLoop)
#pragma config(Motor,  port2,           rightFrontDrive, tmotorVex393, openLoop, reversed)
#pragma config(Motor,  port3,           rightBackDrive, tmotorVex393, openLoop, reversed,
encoder, encoderPort, I2C_2, 1000)
#pragma config(Motor,  port4,           rightTopArm,   tmotorVex393, openLoop)
#pragma config(Motor,  port5,           rightBottomArm, tmotorVex393, openLoop)
#pragma config(Motor,  port6,           leftTopArm,    tmotorVex393, openLoop, reversed)
#pragma config(Motor,  port7,           leftBottomArm, tmotorVex393, openLoop, reversed)
#pragma config(Motor,  port8,           leftFrontDrive, tmotorVex393, openLoop)
#pragma config(Motor,  port9,           leftBackDrive, tmotorVex393, openLoop, encoder,
encoderPort, I2C_1, 1000)
#pragma config(Motor,  port10,          leftIntake,    tmotorVex393, openLoop, reversed)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

#pragma platform(VEX)

//Competition Control and Duration Settings
#pragma competitionControl(Competition)
#pragma autonomousDuration(60)
#pragma userControlDuration(60)

//Main competition background code...do not modify!
#include "Vex_Competition_Includes.c"

// These reference our own files
#include "Utilities.c"
#include "Constants.c"
#include "Subsystem_Functions.c"
#include "Autonomous_Modes.c"
#include "Autonomous_Selection.c"

/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Pre-Autonomous Functions
//
// Actions that are performed before the match starts are coded here.
//
/////////////////////////////////////////////////////////////////////////////////////////

void pre_auton()
{
  // Set bStopTasksBetweenModes to false if you want to keep user created tasks running
between
  // Autonomous and Tele-Op modes. You will need to manage all user created tasks if set
to false.
  bStopTasksBetweenModes = true;
```

```
        // All activities that occur before the competition starts
        // Example: clearing encoders, setting servo positions, ...

  initArmLimits();        // determine Dead or Alive robot and set arm limits accordingly
  initAllPids();          // set all the PID constants
  if (nLCDButtons != 0)   // if any LCD buttons are pressed during startup...
        {selectAutonomous();} // ...allow the user to select which autonomous to run.
}

/////////////////////////////////////////////////////////////////////////////////////////
//
//                                 Autonomous Task
//
// This task is used to control the robot during the autonomous phase of competition.
//
/////////////////////////////////////////////////////////////////////////////////////////

task autonomous()
{
        runSelectedAutonomous(); // runs an autonomous routine based on the selection from
the LCD
                                 // see "Autonomous_Modes.c" for the different routines
}

/////////////////////////////////////////////////////////////////////////////////////////
//
//                                 User Control Task
//
// This task is used to control the robot during the user control phase of competition.
//
/////////////////////////////////////////////////////////////////////////////////////////

task usercontrol()
{
        while (true)  // BLT (Big Loop Technology)
        {
          // This is the main execution loop for user control. Each time through the loop
          // the program updates motor values based on feedback from the Joystick.

          setDrive();
          setArm();
          setIntake();
          displayBatteryVoltage();
        }
}
```

J.2 Subsystem_Functions.c

```
/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Subsystem Functions
//
// All the robot-specific subsystem functions are defined here.
// Note: this file will not compile.  Only try to compile the top-level code that
// "#include"'s this file.
//
/////////////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Initializations/Init Functions
//
// These are called in Pre-Atuon (during startup).
//
/////////////////////////////////////////////////////////////////////////////////////////

PidState armPid;         // create the arm PID struct as a global so we can see the values
in the debugger
PidState drivePid; // create the drive PID struct as a global so we can see the values in
the debugger
PidState turnPid;  // create the turn PID struct as a global so we can see the values in
the debugger
// a second (more important) reason to make these globals is so that all functions can
reference them

void initAllPids() // sets all the PID struct constants - to be called in pre-auton
{
        InitPidConstants(drivePid, driveKp, driveKi, driveKd);
        InitPidConstants(armPid, armKp, armKi, armKd);
        InitPidConstants(turnPid, turnKp, turnKi, turnKd);
}

float LEFT_ARM_UPPER_LIMIT = DEAD_ARM_UPPER_LIMIT;  // upper limit for arm potentiometer
(default to Dead)
float LEFT_ARM_LOWER_LIMIT = DEAD_ARM_LOWER_LIMIT;  // lower limit for arm potentiometer
(default to Dead)

void initArmLimits() // checks for Alive jumper and if so, sets arm upper and lower limits
appropriately
{
        if (SensorValue(Jumper) == 0) // if the Alive jumper clip is in
        {
                LEFT_ARM_UPPER_LIMIT = ALIVE_ARM_UPPER_LIMIT;  // change upper limit for
arm potentiometer to Alive limit
                LEFT_ARM_LOWER_LIMIT = ALIVE_ARM_LOWER_LIMIT;  // change lower limit for
arm potentiometer to Alive limit
        }
}


/////////////////////////////////////////////////////////////////////////////////////////
//
```

```
//                          Drive Functions
//
////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////
//            Low-Level                  //
///////////////////////////////////////////

void setDriveMotors(float rightValue, float leftValue) // lets you set all the drive
motors in one call (skid-steer)
{
        if (SensorValue(leftArmPot) > LEFT_ARM_LOWER_LIMIT + 100) // if the arm is lifted
up...
        {
                rightValue = rightValue * SPEED_REDUCTION_FACTOR; // ...reduce the right
speed by our constant factor...
                leftValue = leftValue * SPEED_REDUCTION_FACTOR;   // ...reduce the left
speed by our constant factor.
        }
        motor[rightFrontDrive] = rightValue; // rightFront - 2
        motor[rightBackDrive] = rightValue;  // rightBack - 3
        motor[leftFrontDrive] = leftValue;   // leftFront - 8
        motor[leftBackDrive] = leftValue;    // leftBack - 9
}

///////////////////////////////////////////
//            Driver Control             //
///////////////////////////////////////////

void setDrive()                 // reads Joystick, calculates drive motor values and sends
values to motors - used in driver control
{
        int driveJoystickY = deadband(vexRT[Ch3], DRIVE_DEADBAND_THRESHOLD);    // read
from the left josytick Y-axis (channel 3) and deadband value
        int driveJoystickX = deadband(vexRT[Ch4], DRIVE_DEADBAND_THRESHOLD);    // read
from the left josytick X-axis (channel 4) and deadband value
        int rightMotorValue = driveJoystickY - driveJoystickX;
                                  // arcade drive left side value
        int leftMotorValue = driveJoystickY + driveJoystickX;
                                          // arcade drive right side value
        setDriveMotors(rightMotorValue, leftMotorValue);        // send the calculated
motor values to the drive sides
}

///////////////////////////////////////////
//            Autonomous                 //
///////////////////////////////////////////

void initDriveEncoders() // resets both drive encoder counts to zero
{
        nMotorEncoder[leftBackDrive] = 0;
  nMotorEncoder[rightBackDrive] = 0;
}
```

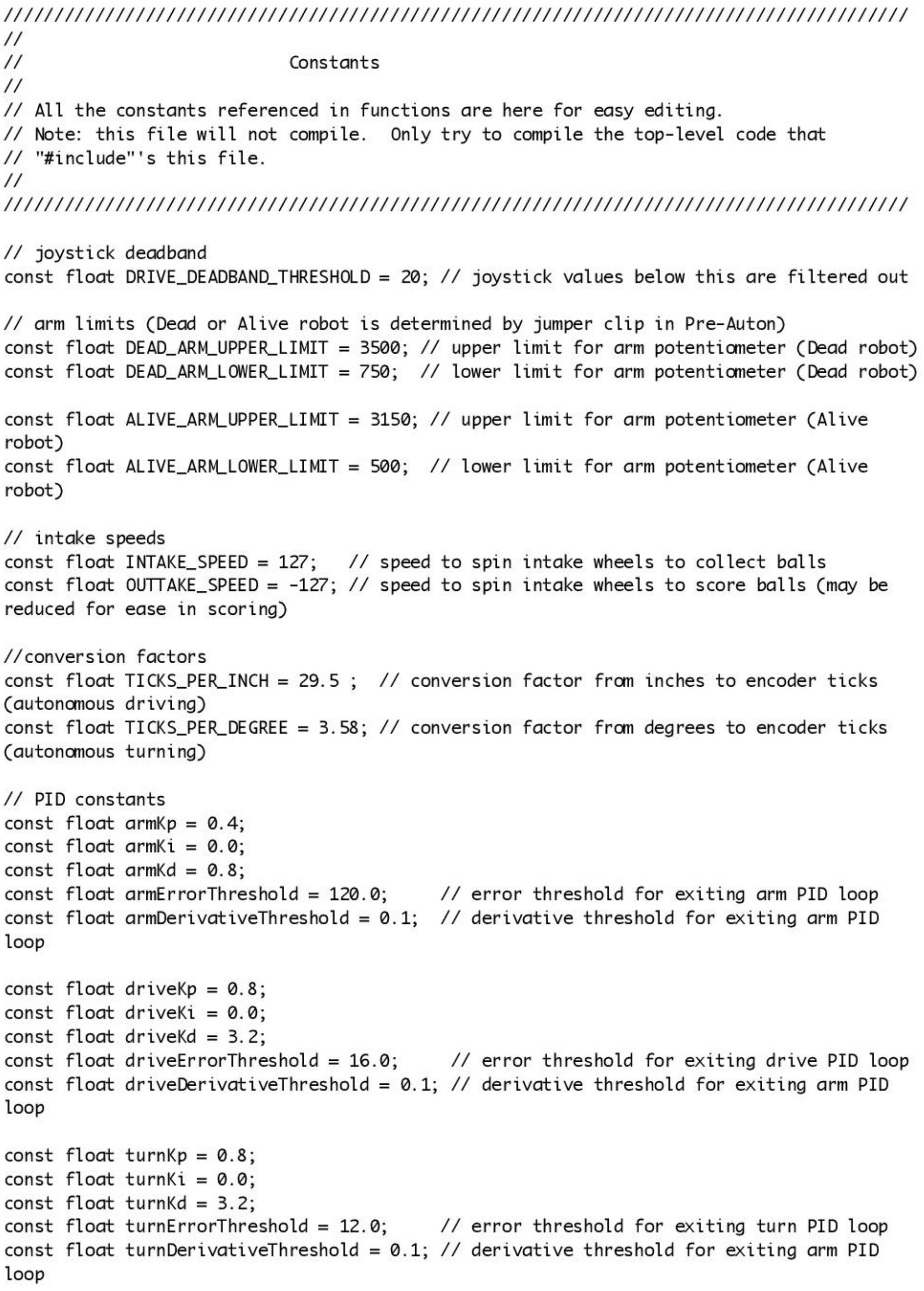
```
void driveInches(int inches)  // uses a PID loop to drive (straight) a given distance -
used in autonomous
{
        float goal = inches * TICKS_PER_INCH; // conversion factor from inches to encoder
ticks
        int output;                                     // speed to send to the drive
motots, set in the loop
        initDriveEncoders();         // reset drive encoder counts to zero
        InitPidGoal(drivePid, goal);  // initialize the drive PID with the goal
        ClearTimer(T1);              // clear the timer

        while( (abs(drivePid.error) > driveErrorThreshold)
                 || (abs(drivePid.derivative) > driveDerivativeThreshold) ) // until both
the error and speed drop below acceptable thresholds...
        {
                if (time1(T1) > PID_UPDATE_TIME)        // if enough time has passed since
the last PID update...
                {
                  float currentPosition = (-nMotorEncoder[leftBackDrive]-
nMotorEncoder[rightBackDrive]) / 2.0; // (the current position is the average of the drive
encoders)
                  output = UpdatePid(drivePid, currentPosition);  // ...update the motor
speed with the drive PID...
                  float offset = DRIVE_STRAIGHT_FACTOR * ( (-nMotorEncoder[leftBackDrive])
- (-nMotorEncoder[rightBackDrive]) ) ; // (proportional correction to straighten out)
                  setDriveMotors(output - offset, output + offset);  // ...and send that
speed to the drive motors (left and right)
                  ClearTimer(T1); // reset the timer so it starts counting again
                }
        }
  setDriveMotors(0, 0);        // remember to stop the motors!
  wait1Msec(TIME_BETWEEN_AUTONOMOUS_ACTIONS); // lets things settle before moving to next
action
}

void turnDegrees(int degrees)  // uses a PID loop to turn a given angle (clockwise) - used
in autonomous
{
        float goal = degrees * TICKS_PER_DEGREE; // conversion factor from degrees to
encoder ticks
        int output;                                     // speed to send to the drive
motots, set in the loop
        initDriveEncoders();        // reset drive encoder counts to zero
        InitPidGoal(turnPid, goal); // initialize the drive PID with the goal
        ClearTimer(T2);             // clear the timer

        while( (abs(turnPid.error) > turnErrorThreshold)
                 || (abs(turnPid.derivative) > turnDerivativeThreshold) ) // until both
the error and speed drop below acceptable thresholds...
        {
                if (time1(T2) > PID_UPDATE_TIME)        // if enough time has passed since
the last PID update...
                {
                  float currentPosition = (nMotorEncoder[leftBackDrive]-
nMotorEncoder[rightBackDrive]) / 2.0; // (the current position is the (+) average of the
drive encoders)
                  output = UpdatePid(turnPid, currentPosition);  // ...update the motor
speed with the drive PID...
                  setDriveMotors(-output, output);  // ...and send that speed to the drive
motors (left and right)
                  ClearTimer(T2); // clear the timer
                }
        }
  setDriveMotors(0, 0); // remember to stop the motors!
  wait1Msec(TIME_BETWEEN_AUTONOMOUS_ACTIONS); // lets things settle before moving to next
action
}


//////////////////////////////////////////////////////////////////////////////////////////
//
//                          Arm Functions
//
//////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////
//            Low-Level                   //
////////////////////////////////////////////

void setArmMotors(int value)  // lets you set the arm motor values in one call
{
                                // this function also limits the motors at the top/bottom
of arm travel for safety
        if (value > 0 && SensorValue(leftArmPot) > LEFT_ARM_UPPER_LIMIT)      // if we're
above the upper limit and going up...
        { value = 0; }

                                                  // ...set the arm motor value to 0
  else if (value < 0 && SensorValue(leftArmPot) < LEFT_ARM_LOWER_LIMIT) // if we're below
the lower limit and going down...
        { value = 0; }

                                                  // ...set the arm motor value to 0
        motor[rightTopArm] = value;               // rightTop - 4
        motor[rightBottomArm] = value;  // rightBottom - 5
        motor[leftTopArm] = value;                // leftTop - 6
        motor[leftBottomArm] = value;     // leftBottom - 7
}

////////////////////////////////////////////
//            Driver Control              //
////////////////////////////////////////////

bool bHoldHeight = false; // this tells whether or not we want the arm to hold its
position

void setArm()             // reads Joystick, determines arm motor values and sends values
to motors - used in driver control
```

```
{
          // uses PID to hold the arm up when buttons are released
                                 // note: this function does not decide where the upper/
lower limits are, setArmMotors does
        int currentPosition = SensorValue(leftArmPot);
        if (vexRT[Btn5U] == 1)                                // if button 5 Up is
pressed...
                  {
                        bHoldHeight = false;                // ...don't hold position,
instead...
                    armPid.goal = LEFT_ARM_UPPER_LIMIT; // ...set the goal to the upper
limit.
            }
        else if (vexRT[Btn5D] == 1)               // if button 5 Down is pressed...
                  {
                        bHoldHeight = false;                // ...don't hold position,
instead...
                        armPid.goal = LEFT_ARM_LOWER_LIMIT; // ...set the goal to the
lower limit.
                   }
  else
                          // if neither button is pressed...
                  {
                   if (bHoldHeight == false)                // ...and we weren't
previously holding position...
                   {
                     bHoldHeight = true;                    // ...now we want to hold
position, so...
                     armPid.goal = SensorValue(leftArmPot); // ...set the goal to the
current height.
                    }
                  }
        if (time1(T3) > PID_UPDATE_TIME)                             // if enough time has
passed since the last PID update...
                {
                        float output = UpdatePid(armPid, currentPosition); // ...update
the motor speed with the arm PID...
                  setArmMotors(output);                              // ...and send that
speed to the arm motors...
                  ClearTimer(T3);                                    // ...and reset the
timer for the next update.
                 }
}

////////////////////////////////////////////
//            Autonomous                  //
////////////////////////////////////////////

void armHeight(float percent) // uses a PID loop to drive the arm to a given height - used
in autonomous
{
        float goal = ( (percent/100) * (LEFT_ARM_UPPER_LIMIT - LEFT_ARM_LOWER_LIMIT) ) +
LEFT_ARM_LOWER_LIMIT;
        int output;                // speed to send to the arm motots, set in the loop
        InitPidGoal(armPid, goal); // initialize the arm PID with the goal
        ClearTimer(T3);            // clear the timer

        while( (abs(armPid.error) > armErrorThreshold)
                 || (abs(armPid.derivative) > armDerivativeThreshold) ) // until both the
error and speed drop below acceptable thresholds...
        {
                if (time1(T3) > PID_UPDATE_TIME)        // if enough time has passed since
the last PID update...
                {
                        output = UpdatePid(armPid, SensorValue(leftArmPot));  // ...update
the motor speed with the arm PID...
                  setArmMotors(output); // ...and send that speed to the arm motors
                  ClearTimer(T3);
                 }
        }
        // don't set the motors back to 0 afterwards, we want them to hold position
        wait1Msec(TIME_BETWEEN_AUTONOMOUS_ACTIONS); // lets things settle before moving to
next action
}


/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Intake Functions
//
/////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////
//            Low-Level                   //
////////////////////////////////////////////

void setIntakeMotors(int value) // lets you set both intake motor values in one call
{
        motor[rightIntake]= value;      // rightIntake - 1
        motor[leftIntake]= value;         // rightIntake - 10
}

void intakeIn()
{ setIntakeMotors(INTAKE_SPEED); } // this is full speed in

void intakeOut()
{ setIntakeMotors(OUTTAKE_SPEED); } // speed out can be reduced for ease of scoring

void intakeStop()
{ setIntakeMotors(0); } // stop the intake motors

////////////////////////////////////////////
//            Driver Control              //
////////////////////////////////////////////

void setIntake()        // reads the Joystick, determines the state of the intake and
sends values to motors - used in driver control
{
        if(vexRT[Btn6U] == 1)                   // if button 6 Up is pressed...
                {intakeIn();}                                       // ...intake
balls.
        else if(vexRT[Btn6D] == 1)      // if button 6 Down is pressed...
                {intakeOut();}                                      // ...outtake
balls (possibly at reduced speed).
        else
                // if neither button is pressed...
                {intakeStop();}           // ...stop the motors.
}

////////////////////////////////////////////
//            Autonomous                  //
////////////////////////////////////////////

void deployIntake()  // jerks the robot back and forth to deploy the intake arms
{
        setDriveMotors(127,127);
        wait1Msec(100);
        setDriveMotors(-127,-127);
        wait1Msec(100);
        setDriveMotors(0,0);
        wait1Msec(TIME_BETWEEN_AUTONOMOUS_ACTIONS); // lets things settle before moving to
next action
}
```

J.3 Utilities.c

```
/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Utilities
//
// Functions that are not specific to one robot, like math, PID, and LCD functions, are
here.
// This file will be "include"ed in both robots' top-level files.
// Note: this file will not compile.  Only try to compile the top-level code that
// "#include"'s this file.
//
/////////////////////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Math Functions
//
/////////////////////////////////////////////////////////////////////////////////////////


// deadband function
int deadband(int input, int threshold)
{
        if (abs(input) < threshold) // if the input value is less than the threshold
value...
                {input = 0;}                                                    // ...set
it to zero...
        return input;                                                           // ...then
return the value either way.
}

// clipping function (note: this is done automatically anyway for motor values)
int clip(int input, int limit = 127)
{
        if (input > limit)       // if the input is above the limit...
                {input = limit;}                         // set it to the lmimt.
  else if (input < -limit) // if the input is below the negative of the limit...
        {input = -limit;}                        // set it to the negative of the limit.
  return input;                                          // ...then return the value
either way.
}


/////////////////////////////////////////////////////////////////////////////////////////
//
//                          PID functions (and struct definition)
//
/////////////////////////////////////////////////////////////////////////////////////////


// PID struct
typedef struct
{
  float kP;
  float kI;
  float kD;
  float goal;
  float integral;
  float error;
  float previousError;
  float derivative;

} PidState;

// PID constant initialization function
void InitPidConstants(PidState* pidState, float kP, float kI, float kD)
{
  pidState->kP = kP;
  pidState->kI = kI;
  pidState->kD = kD;
}

// sets the goal separately
void InitPidGoal(PidState* pidState, float goal)
{
        pidState->goal = goal;
  pidState->error = goal;
  pidState->previousError = goal;
  pidState->derivative = 0;
  pidState->integral = 0;
}

// PID update function
int UpdatePid(PidState* pidState, float currentPosition)
{
  pidState->error = pidState->goal - currentPosition;
  pidState->integral += pidState->error;
  pidState->derivative = pidState->error - pidState->previousError;
  pidState->previousError = pidState->error;

  int output = (pidState->kP * pidState->error)
             + (pidState->kI * pidState->integral)
             + (pidState->kD * pidState->derivative);

  return output;
}


//////////////////////////////////////////////////////////////////////////////////////////
//
//                          LCD functions
//
//////////////////////////////////////////////////////////////////////////////////////////

void displayBatteryVoltage()    // displays the current battery voltages
{
  bLCDBacklight = true; // turns on the backlight
  string mainBattery, backupBattery;   // these strings will be written with the battery
voltages

  displayLCDString(0, 0, "Primary: "); // displays "Primary: "
  sprintf(mainBattery, "%1.2f%c", nImmediateBatteryLevel/1000.0,'V'); // creates a string
of the main battery voltage
  displayNextLCDString(mainBattery);   // displays the string on the screen in the next
avaliable row

  displayLCDString(1, 0, "Backup: ");  // displays "Backup: "
  sprintf(backupBattery, "%1.2f%c", BackupBatteryLevel/1000.0, 'V');  // creates a string
of the backup battery voltage
  displayNextLCDString(backupBattery); // displays the string on the screen in the next
avaliable row
 }
```

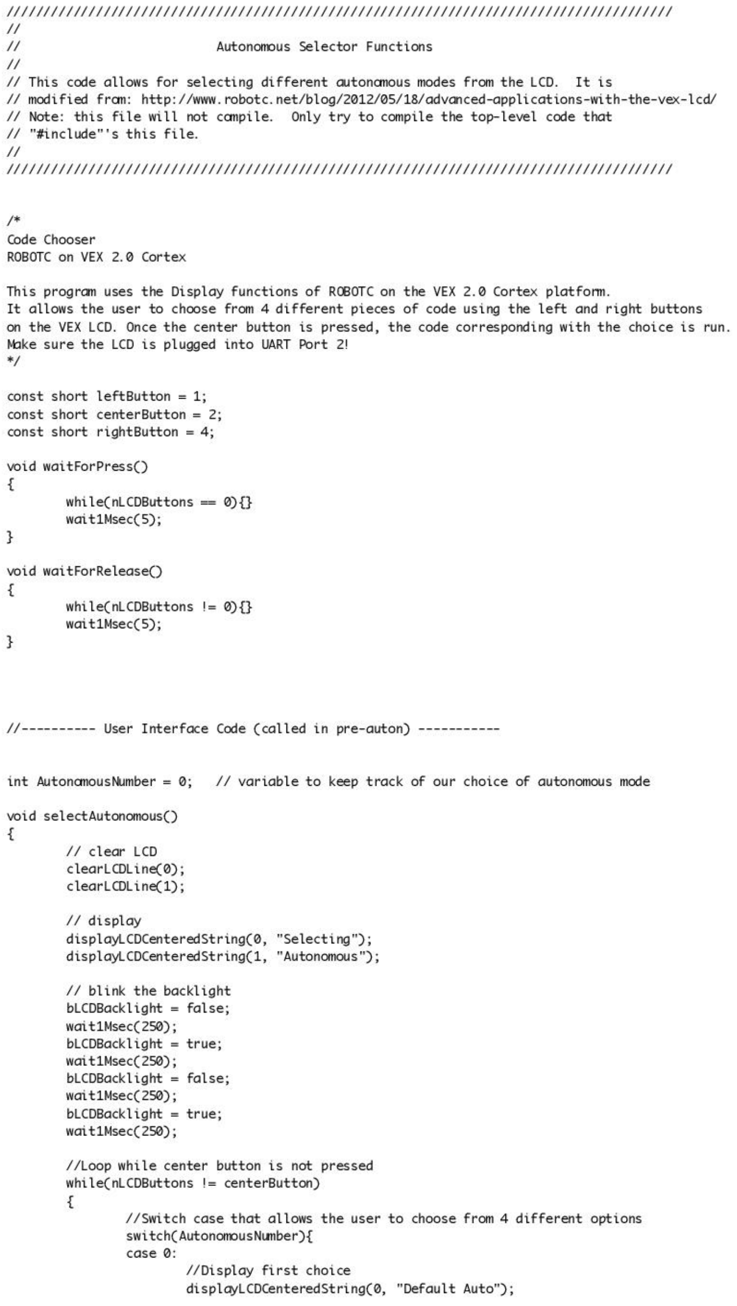
J.4 Constants.c



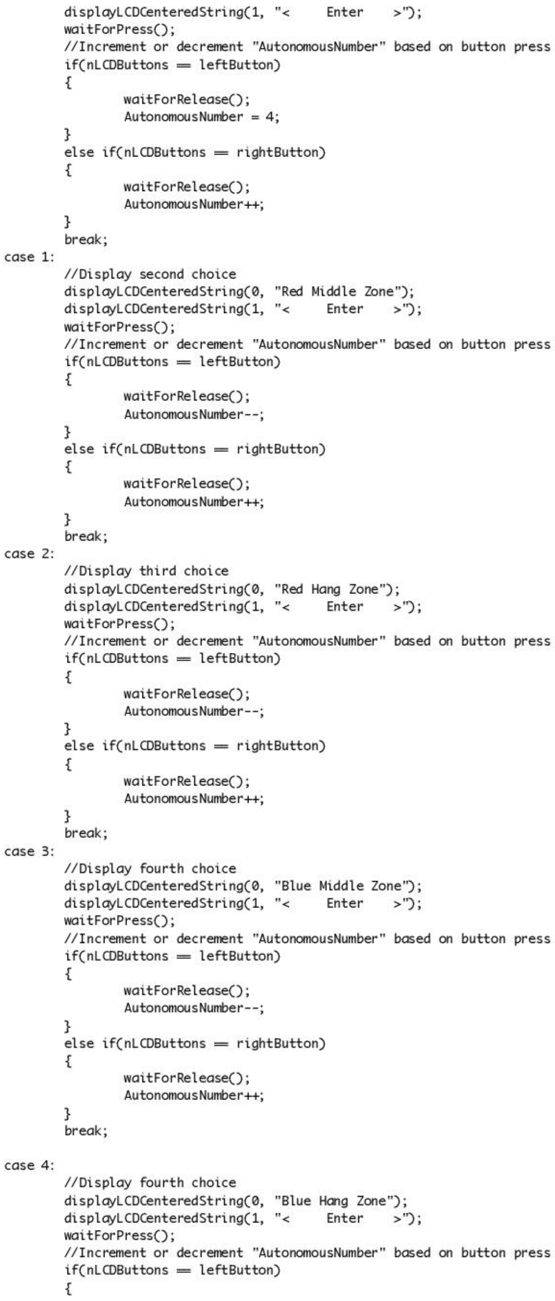
```
/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Constants
//
// All the constants referenced in functions are here for easy editing.
// Note: this file will not compile.  Only try to compile the top-level code that
// "#include"'s this file.
//
/////////////////////////////////////////////////////////////////////////////////////////

// joystick deadband
const float DRIVE_DEADBAND_THRESHOLD = 20; // joystick values below this are filtered out

// arm limits (Dead or Alive robot is determined by jumper clip in Pre-Auton)
const float DEAD_ARM_UPPER_LIMIT = 3500; // upper limit for arm potentiometer (Dead robot)
const float DEAD_ARM_LOWER_LIMIT = 750;  // lower limit for arm potentiometer (Dead robot)

const float ALIVE_ARM_UPPER_LIMIT = 3150; // upper limit for arm potentiometer (Alive
robot)
const float ALIVE_ARM_LOWER_LIMIT = 500;  // lower limit for arm potentiometer (Alive
robot)

// intake speeds
const float INTAKE_SPEED = 127;   // speed to spin intake wheels to collect balls
const float OUTTAKE_SPEED = -127; // speed to spin intake wheels to score balls (may be
reduced for ease in scoring)

//conversion factors
const float TICKS_PER_INCH = 29.5 ;  // conversion factor from inches to encoder ticks
(autonomous driving)
const float TICKS_PER_DEGREE = 3.58; // conversion factor from degrees to encoder ticks
(autonomous turning)

// PID constants
const float armKp = 0.4;
const float armKi = 0.0;
const float armKd = 0.8;
const float armErrorThreshold = 120.0;     // error threshold for exiting arm PID loop
const float armDerivativeThreshold = 0.1;  // derivative threshold for exiting arm PID
loop

const float driveKp = 0.8;
const float driveKi = 0.0;
const float driveKd = 3.2;
const float driveErrorThreshold = 16.0;     // error threshold for exiting drive PID loop
const float driveDerivativeThreshold = 0.1; // derivative threshold for exiting arm PID
loop

const float turnKp = 0.8;
const float turnKi = 0.0;
const float turnKd = 3.2;
const float turnErrorThreshold = 12.0;     // error threshold for exiting turn PID loop
const float turnDerivativeThreshold = 0.1; // derivative threshold for exiting arm PID
loop
```

```
// note: thresholds are only used in autonomous; Kp, Ki, Kd are for both auto and user
control

const float PID_UPDATE_TIME = 30;          // time to wait between PID updates (in ms)

//other factors
const float DRIVE_STRAIGHT_FACTOR = 0.4;   // factor for driving straight proportioanl
correction in autonomous
const float SPEED_REDUCTION_FACTOR = 0.4;  // scaling factor for drive speed when the arm
is up
const float TIME_BETWEEN_AUTONOMOUS_ACTIONS = 50; // wait this long at the end of each
autonomous step to let things settle
const float ARM_PASS_HEIGHT = 55; // percent of max arm height to raise to in order to
pass in autonomous

/*
Power Expander assignments:
port 3 -> A
port 5 -> B
port 7 -> C
port 9 -> D
*/
```

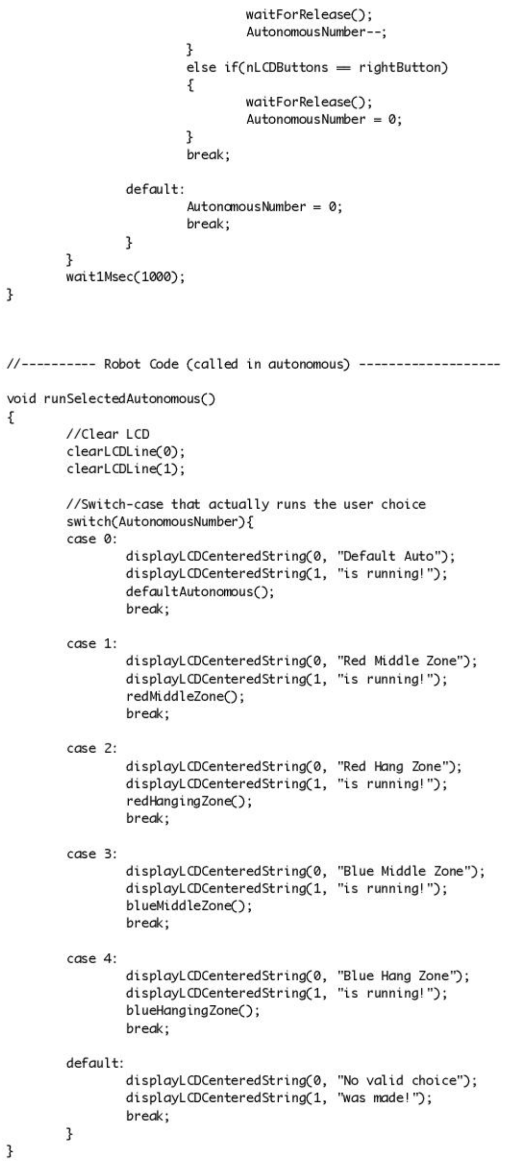
J.5 Autonomous_Selection.c



```
////////////////////////////////////////////////////////////////////////////////////////
//
//                          Autonomous Selector Functions
//
// This code allows for selecting different autonamous modes from the LCD.  It is
// modified fram: http://www.robotc.net/blog/2012/05/18/advanced-applications-with-the-vex-lcd/
// Note: this file will not campile.  Only try to compile the top-level code that
// "#include"'s this file.
//
////////////////////////////////////////////////////////////////////////////////////////


/*
Code Chooser
ROBOTC on VEX 2.0 Cortex

This program uses the Display functions of ROBOTC on the VEX 2.0 Cortex platform.
It allows the user to choose from 4 different pieces of code using the left and right buttons
on the VEX LCD. Once the center button is pressed, the code corresponding with the choice is run.
Make sure the LCD is plugged into UART Port 2!
*/

const short leftButton = 1;
const short centerButton = 2;
const short rightButton = 4;

void waitForPress()
{
        while(nLCDButtons == 0){}
        wait1Msec(5);
}

void waitForRelease()
{
        while(nLCDButtons != 0){}
        wait1Msec(5);
}



//---------- User Interface Code (called in pre-auton) -----------


int AutonamousNumber = 0;   // variable to keep track of our choice of autonomous mode

void selectAutonomous()
{
        // clear LCD
        clearLCDLine(0);
        clearLCDLine(1);

        // display
        displayLCDCenteredString(0, "Selecting");
        displayLCDCenteredString(1, "Autonomous");

        // blink the backlight
        bLCDBacklight = false;
        wait1Msec(250);
        bLCDBacklight = true;
        wait1Msec(250);
        bLCDBacklight = false;
        wait1Msec(250);
        bLCDBacklight = true;
        wait1Msec(250);

        //Loop while center button is not pressed
        while(nLCDButtons != centerButton)
        {
                //Switch case that allows the user to choose from 4 different options
                switch(AutonomousNumber){
                case 0:
                        //Display first choice
                        displayLCDCenteredString(0, "Default Auto");
```

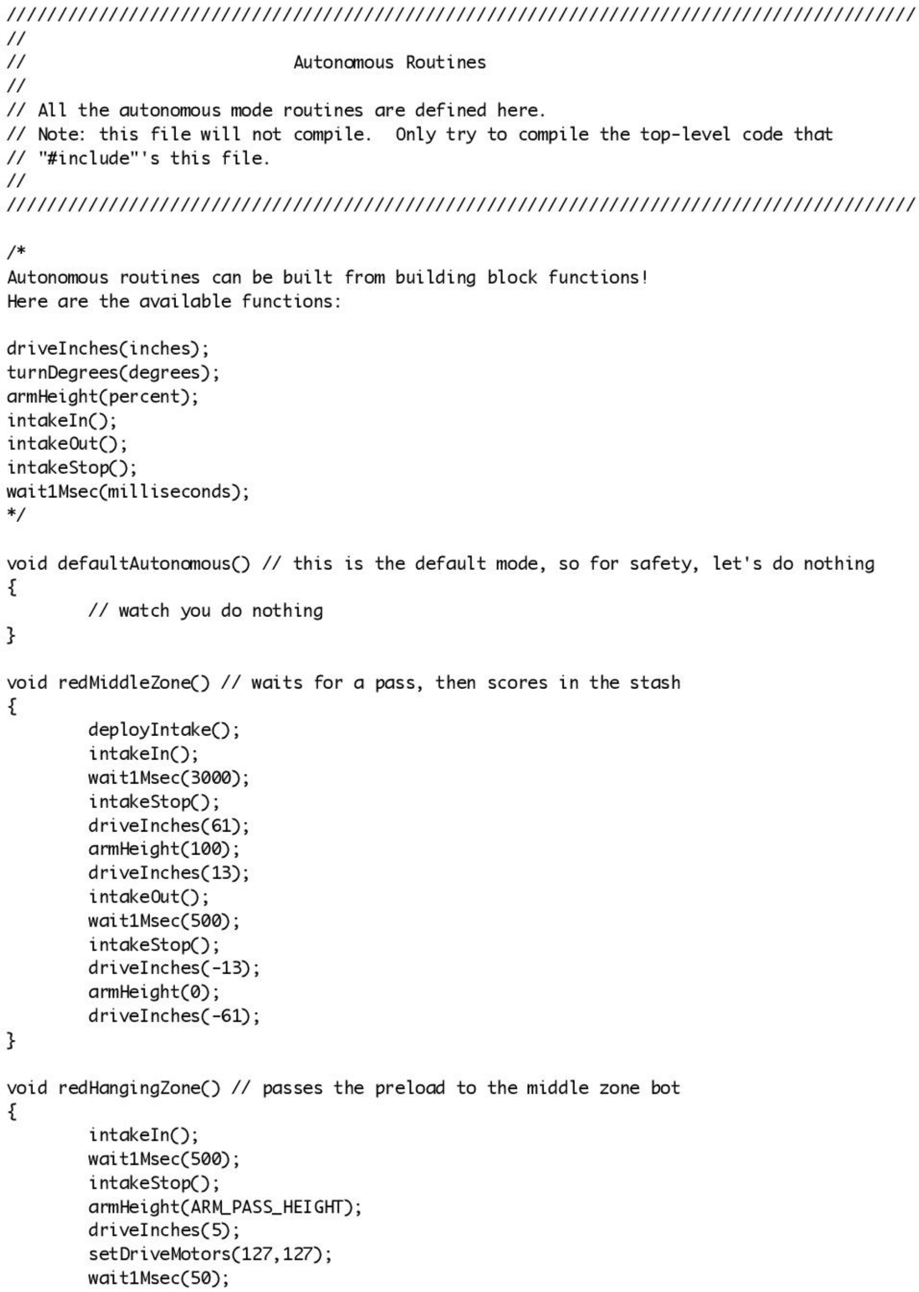


```
        displayLCDCenteredString(1, "<     Enter    >");
        waitForPress();
        //Increment or decrement "AutonomousNumber" based on button press
        if(nLCDButtons == leftButton)
        {
                waitForRelease();
                AutonomousNumber = 4;
        }
        else if(nLCDButtons == rightButton)
        {
                waitForRelease();
                AutonomousNumber++;
        }
        break;
case 1:
        //Display second choice
        displayLCDCenteredString(0, "Red Middle Zone");
        displayLCDCenteredString(1, "<     Enter    >");
        waitForPress();
        //Increment or decrement "AutonomousNumber" based on button press
        if(nLCDButtons == leftButton)
        {
                waitForRelease();
                AutonomousNumber--;
        }
        else if(nLCDButtons == rightButton)
        {
                waitForRelease();
                AutonomousNumber++;
        }
        break;
case 2:
        //Display third choice
        displayLCDCenteredString(0, "Red Hang Zone");
        displayLCDCenteredString(1, "<     Enter    >");
        waitForPress();
        //Increment or decrement "AutonomousNumber" based on button press
        if(nLCDButtons == leftButton)
        {
                waitForRelease();
                AutonomousNumber--;
        }
        else if(nLCDButtons == rightButton)
        {
                waitForRelease();
                AutonomousNumber++;
        }
        break;
case 3:
        //Display fourth choice
        displayLCDCenteredString(0, "Blue Middle Zone");
        displayLCDCenteredString(1, "<     Enter    >");
        waitForPress();
        //Increment or decrement "AutonomousNumber" based on button press
        if(nLCDButtons == leftButton)
        {
                waitForRelease();
                AutonomousNumber--;
        }
        else if(nLCDButtons == rightButton)
        {
                waitForRelease();
                AutonomousNumber++;
        }
        break;

case 4:
        //Display fourth choice
        displayLCDCenteredString(0, "Blue Hang Zone");
        displayLCDCenteredString(1, "<     Enter    >");
        waitForPress();
        //Increment or decrement "AutonomousNumber" based on button press
        if(nLCDButtons == leftButton)
        {
```

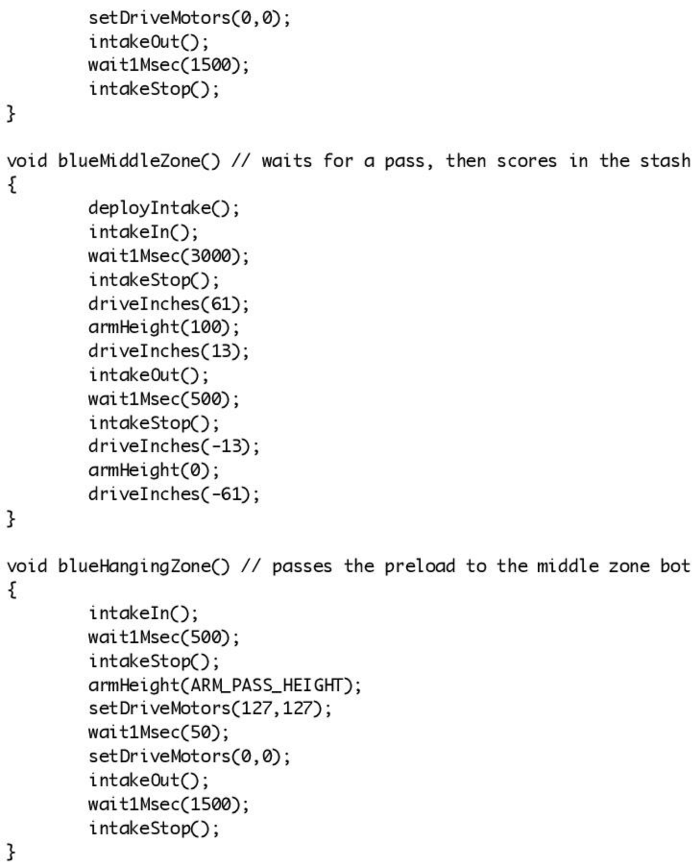


```
                                        waitForRelease();
                                        AutonomousNumber--;
                                }
                                else if(nLCDButtons == rightButton)
                                {
                                        waitForRelease();
                                        AutonomousNumber = 0;
                                }
                                break;

                        default:
                                AutonomousNumber = 0;
                                break;
                        }
                }
                wait1Msec(1000);
}


//---------- Robot Code (called in autonomous) -------------------

void runSelectedAutonomous()
{
        //Clear LCD
        clearLCDLine(0);
        clearLCDLine(1);

        //Switch-case that actually runs the user choice
        switch(AutonomousNumber){
        case 0:
                displayLCDCenteredString(0, "Default Auto");
                displayLCDCenteredString(1, "is running!");
                defaultAutonomous();
                break;

        case 1:
                displayLCDCenteredString(0, "Red Middle Zone");
                displayLCDCenteredString(1, "is running!");
                redMiddleZone();
                break;

        case 2:
                displayLCDCenteredString(0, "Red Hang Zone");
                displayLCDCenteredString(1, "is running!");
                redHangingZone();
                break;

        case 3:
                displayLCDCenteredString(0, "Blue Middle Zone");
                displayLCDCenteredString(1, "is running!");
                blueMiddleZone();
                break;

        case 4:
                displayLCDCenteredString(0, "Blue Hang Zone");
                displayLCDCenteredString(1, "is running!");
                blueHangingZone();
                break;

        default:
                displayLCDCenteredString(0, "No valid choice");
                displayLCDCenteredString(1, "was made!");
                break;
        }
}
```

J.6 Autonomous_Modes.c



```
//////////////////////////////////////////////////////////////////////////////////////////
//
//                            Autonomous Routines
//
// All the autonomous mode routines are defined here.
// Note: this file will not compile.  Only try to compile the top-level code that
// "#include"'s this file.
//
//////////////////////////////////////////////////////////////////////////////////////////

/*
Autonomous routines can be built from building block functions!
Here are the available functions:

driveInches(inches);
turnDegrees(degrees);
armHeight(percent);
intakeIn();
intakeOut();
intakeStop();
wait1Msec(milliseconds);
*/

void defaultAutonomous() // this is the default mode, so for safety, let's do nothing
{
        // watch you do nothing
}

void redMiddleZone() // waits for a pass, then scores in the stash
{
        deployIntake();
        intakeIn();
        wait1Msec(3000);
        intakeStop();
        driveInches(61);
        armHeight(100);
        driveInches(13);
        intakeOut();
        wait1Msec(500);
        intakeStop();
        driveInches(-13);
        armHeight(0);
        driveInches(-61);
}

void redHangingZone() // passes the preload to the middle zone bot
{
        intakeIn();
        wait1Msec(500);
        intakeStop();
        armHeight(ARM_PASS_HEIGHT);
        driveInches(5);
        setDriveMotors(127,127);
        wait1Msec(50);
```



```
        setDriveMotors(0,0);
        intakeOut();
        wait1Msec(1500);
        intakeStop();
}

void blueMiddleZone() // waits for a pass, then scores in the stash
{
        deployIntake();
        intakeIn();
        wait1Msec(3000);
        intakeStop();
        driveInches(61);
        armHeight(100);
        driveInches(13);
        intakeOut();
        wait1Msec(500);
        intakeStop();
        driveInches(-13);
        armHeight(0);
        driveInches(-61);
}

void blueHangingZone() // passes the preload to the middle zone bot
{
        intakeIn();
        wait1Msec(500);
        intakeStop();
        armHeight(ARM_PASS_HEIGHT);
        setDriveMotors(127,127);
        wait1Msec(50);
        setDriveMotors(0,0);
        intakeOut();
        wait1Msec(1500);
        intakeStop();
}
```

Appendix K

Test Data

Table K.1: Results obtained after numerous tests based on the experimental protocol.

Elements/Requirements	**Target Range**	**Trials**	**Results**
Robot hanging w/o Ball	10"-12" hanging height	0	Robot hanging was abandoned due to time and cost
Robot hanging w/ Ball	10"-12" hanging height	0	Robot hanging was abandoned due to time and cost
20 Bucky Balls	Collect 5 balls in 30 seconds	10	Robot collected 5 balls in 20 seconds
8 Large Balls	Collect 5 balls in 25 seconds	10	Robot collected 5 balls in 30 seconds
12" Barrier	Maneuver around barrier repeatedly for a minute	10	Robot maneuvered around barrier repeatedly without tipping or entanglement
2" Bump	Maneuver around the bump repeatedly for 60 seconds	10	Robot maneuvered around bump repeatedly without tipping over
Middle Zone Balls	Move 10 balls into zone in 30 seconds	15	Robot moved 10 balls into zone in 30 seconds
Goal Zone Small	Move 10 balls into zone in 30 seconds	15	Robot was able to move 10 balls into zone in 20 seconds
Goal Zone Large	Move 4 balls into zone in 30 seconds	15	Robot moved 4 balls into zone in 30 seconds
Stashed Zone Small	Move 10 balls into zone in 30 seconds	15	Robot moved 10 balls into zone in 30 seconds
Stashed Zone Large	Move 4 balls into zone in 30 seconds	15	Robot moved 4 balls into zone in 30 seconds

Continued on next page

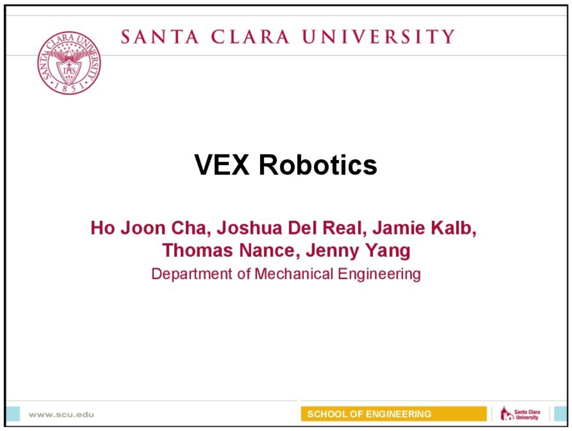
Table K.1 – *Continued from previous page*

Elements/Requirements	**Target Range**	**Trials**	**Results**
Autonomous Bonus	Collect balls and move balls into various zones autonomosuly	30	Robots could autonomously collect 1 large ball, manipulate 10 small balls, and disrupt opponents movements
De-score Opponents Balls	Collect 5 opponent small balls and move out 4 large balls in 60 seconds	20	Robots could move out 6 opponent small balls and move out 4 opponent large balls
Low Center of Gravity	Perform all tasks with stability and precision	30	Anti-tip wheel incorportated to ensure stability and precision

Appendix L

Senior Design Conference Presentation

The following pages contains the senior design conference presentation slides presented on May 8, 2014.



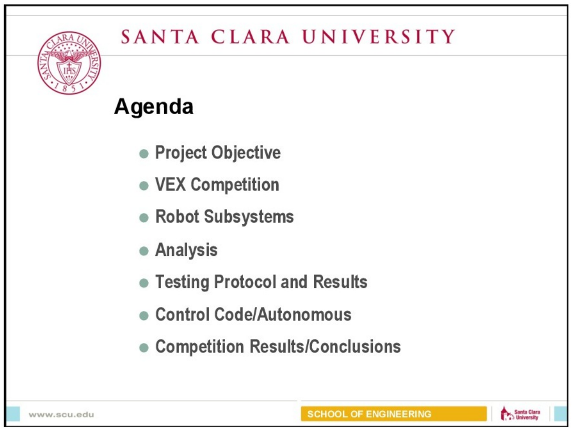
SANTA CLARA UNIVERSITY

VEX Robotics

Ho Joon Cha, Joshua Del Real, Jamie Kalb, Thomas Nance, Jenny Yang

Department of Mechanical Engineering

www.scu.edu SCHOOL OF ENGINEERING

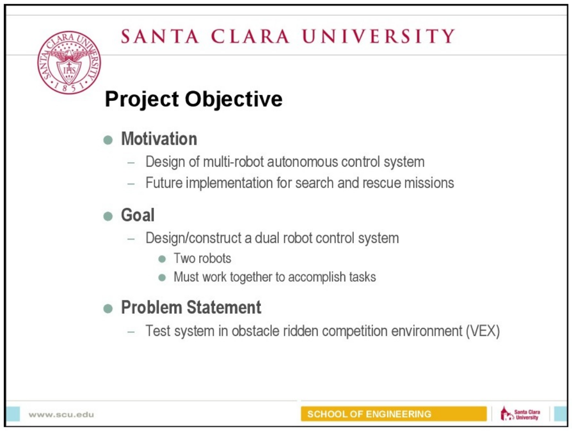


SANTA CLARA UNIVERSITY

Agenda

- Project Objective
- VEX Competition
- Robot Subsystems
- Analysis
- Testing Protocol and Results
- Control Code/Autonomous
- Competition Results/Conclusions

www.scu.edu SCHOOL OF ENGINEERING

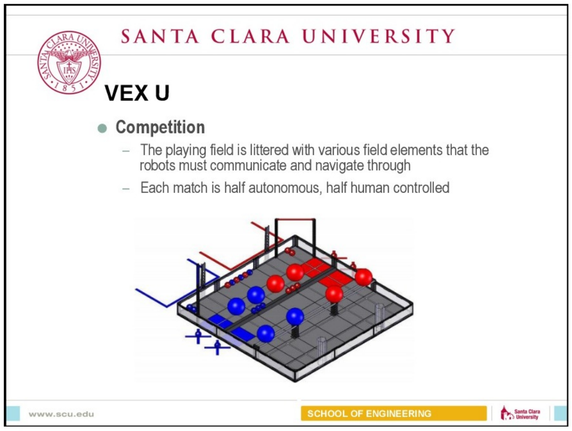


SANTA CLARA UNIVERSITY

Project Objective

- Motivation
  - Design of multi-robot autonomous control system
  - Future implementation for search and rescue missions
- Goal
  - Design/construct a dual robot control system
    - Two robots
    - Must work together to accomplish tasks
- Problem Statement
  - Test system in obstacle ridden competition environment (VEX)

www.scu.edu SCHOOL OF ENGINEERING

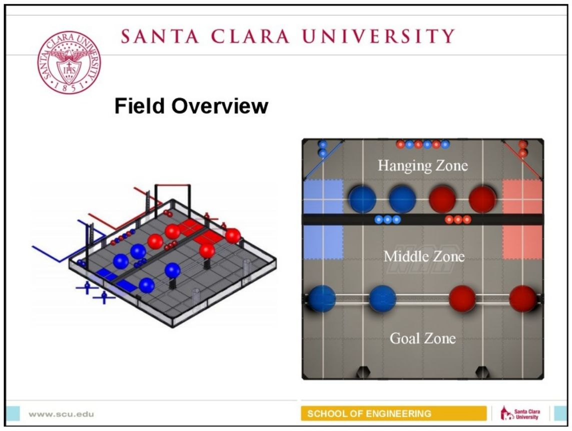


SANTA CLARA UNIVERSITY

VEX U

- Competition
  - The playing field is littered with various field elements that the robots must communicate and navigate through
  - Each match is half autonomous, half human controlled

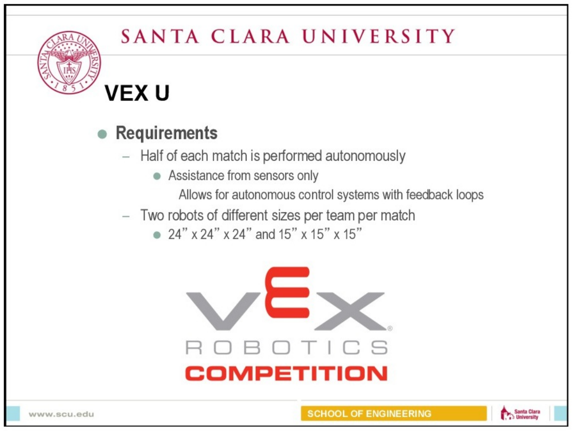
www.scu.edu SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

Field Overview

www.scu.edu SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

VEX U

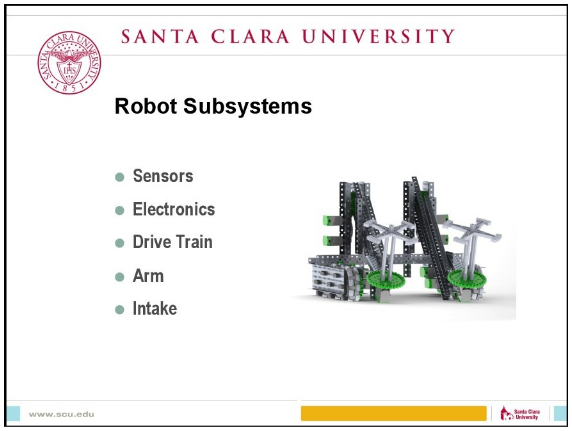
- Requirements
  - Half of each match is performed autonomously
    - Assistance from sensors only
      Allows for autonomous control systems with feedback loops
  - Two robots of different sizes per team per match
    - 24" x 24" x 24" and 15" x 15" x 15"

VEX ROBOTICS COMPETITION

www.scu.edu SCHOOL OF ENGINEERING

SANTA CLARA UNIVERSITY

www.scu.edu SCHOOL OF ENGINEERING

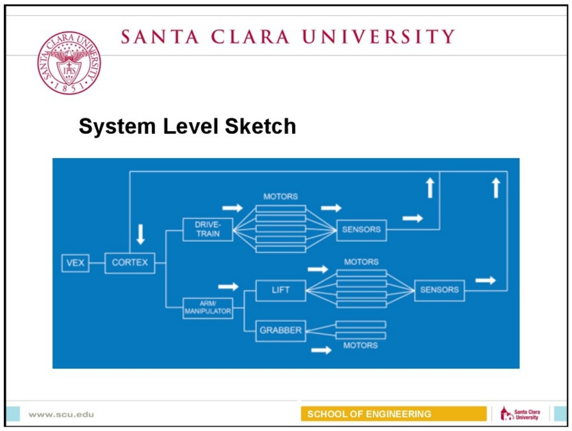


SANTA CLARA UNIVERSITY

Robot Subsystems

- Sensors
- Electronics
- Drive Train
- Arm
- Intake

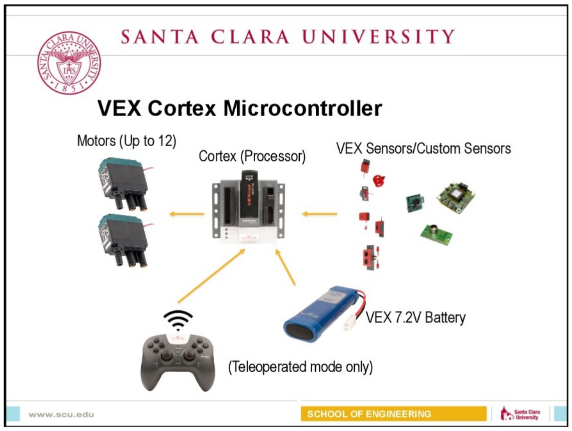
www.scu.edu



SANTA CLARA UNIVERSITY

System Level Sketch

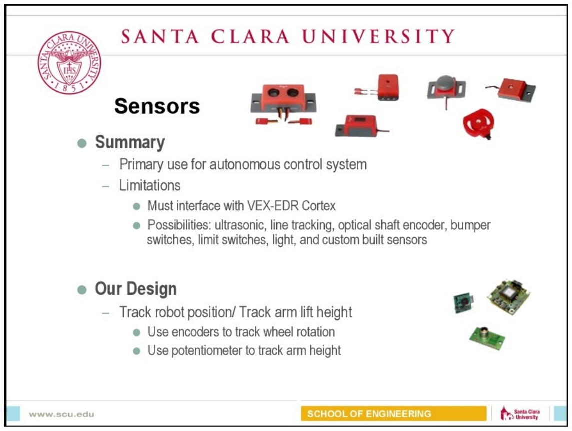
www.scu.edu SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

VEX Cortex Microcontroller

www.scu.edu SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

Sensors

- Summary
  - Primary use for autonomous control system
  - Limitations
    - Must interface with VEX-EDR Cortex
    - Possibilities: ultrasonic, line tracking, optical shaft encoder, bumper switches, limit switches, light, and custom built sensors
- Our Design
  - Track robot position/ Track arm lift height
    - Use encoders to track wheel rotation
    - Use potentiometer to track arm height

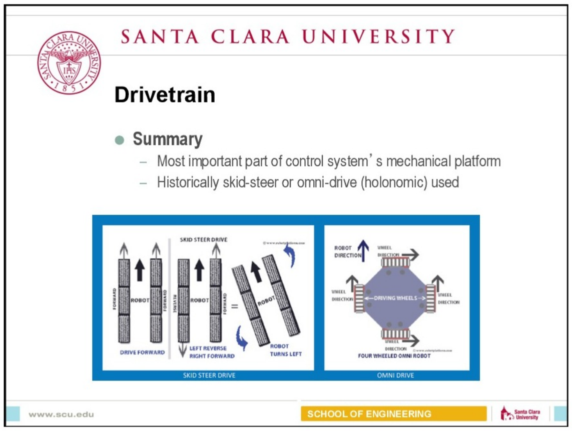
www.scu.edu SCHOOL OF ENGINEERING

SANTA CLARA UNIVERSITY

Electronics

- Available Parts
  - Motors: 12 motors maximum allowed
  - Processor: VEX-EDR CORTEX Microcontroller
    - 10 motor outputs
    - 8 Analog input
    - 12 digital I/O Ports
- Power Management
  - Power Supply: 7 volt 3000 mAh NIMH Power Pack
  - Power Expander: Allows for two batteries

www.scu.edu SCHOOL OF ENGINEERING

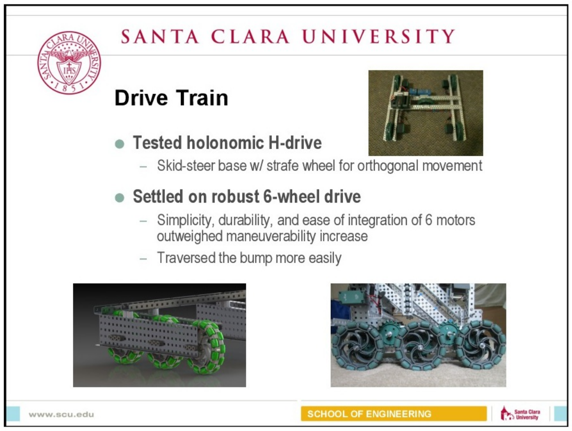


SANTA CLARA UNIVERSITY

Drivetrain

- **Summary**
  - Most important part of control system' s mechanical platform
  - Historically skid-steer or omni-drive (holonomic) used

www.scu.edu SCHOOL OF ENGINEERING

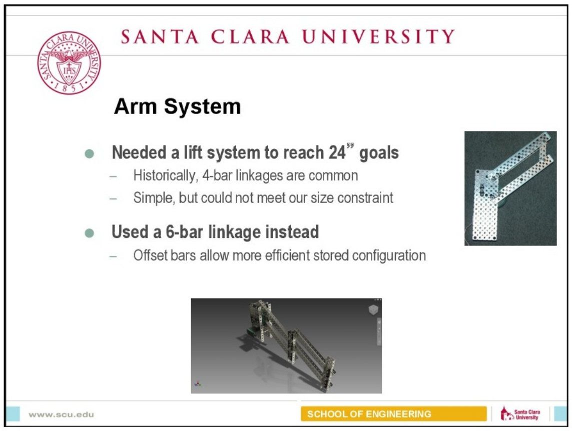


SANTA CLARA UNIVERSITY

Drive Train

- **Tested holonomic H-drive**
  - Skid-steer base w/ strafe wheel for orthogonal movement
- **Settled on robust 6-wheel drive**
  - Simplicity, durability, and ease of integration of 6 motors outweighed maneuverability increase
  - Traversed the bump more easily

www.scu.edu SCHOOL OF ENGINEERING

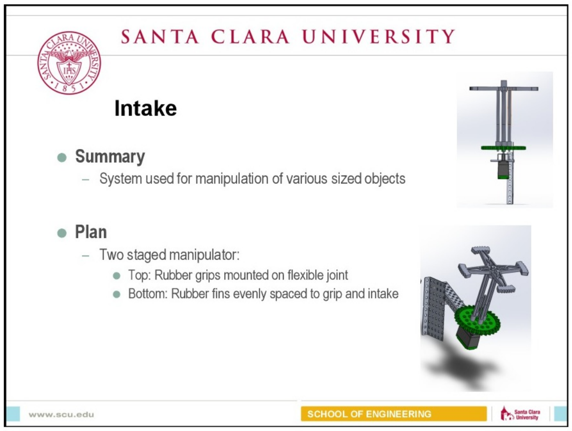


SANTA CLARA UNIVERSITY

Arm System

- **Needed a lift system to reach 24" goals**
  - Historically, 4-bar linkages are common
  - Simple, but could not meet our size constraint
- **Used a 6-bar linkage instead**
  - Offset bars allow more efficient stored configuration

www.scu.edu SCHOOL OF ENGINEERING

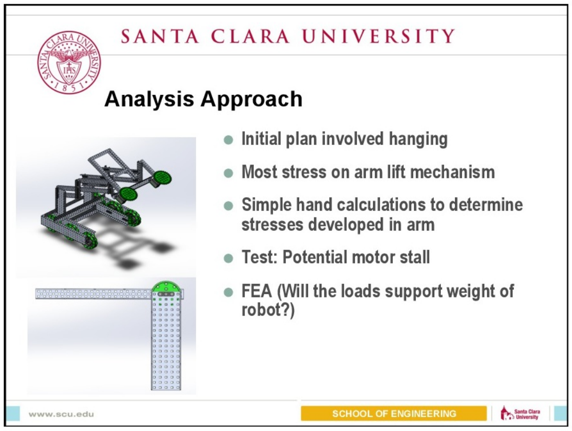


SANTA CLARA UNIVERSITY

Intake

- **Summary**
  - System used for manipulation of various sized objects
- **Plan**
  - Two staged manipulator:
    - Top: Rubber grips mounted on flexible joint
    - Bottom: Rubber fins evenly spaced to grip and intake

www.scu.edu SCHOOL OF ENGINEERING

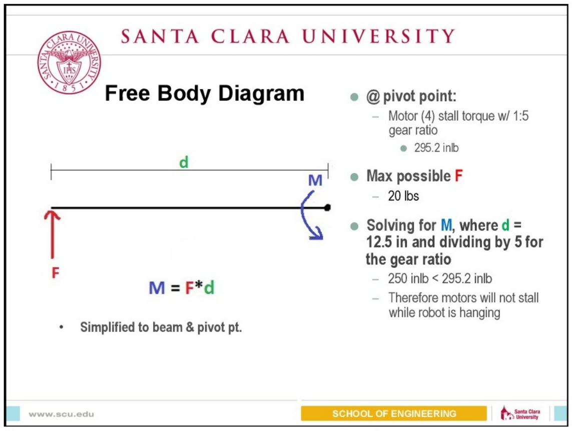


SANTA CLARA UNIVERSITY

Analysis Approach

- **Initial plan involved hanging**
- **Most stress on arm lift mechanism**
- **Simple hand calculations to determine stresses developed in arm**
- **Test: Potential motor stall**
- **FEA (Will the loads support weight of robot?)**

www.scu.edu SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

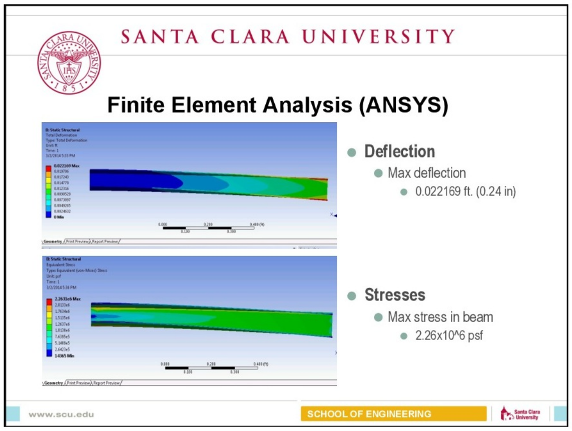
Free Body Diagram

$M = F*d$

- **Simplified to beam & pivot pt.**

- **@ pivot point:**
  - Motor (4) stall torque w/ 1:5 gear ratio
    - 295.2 inlb
- **Max possible F**
  - 20 lbs
- **Solving for M, where d = 12.5 in and dividing by 5 for the gear ratio**
  - 250 inlb < 295.2 inlb
  - Therefore motors will not stall while robot is hanging

www.scu.edu SCHOOL OF ENGINEERING



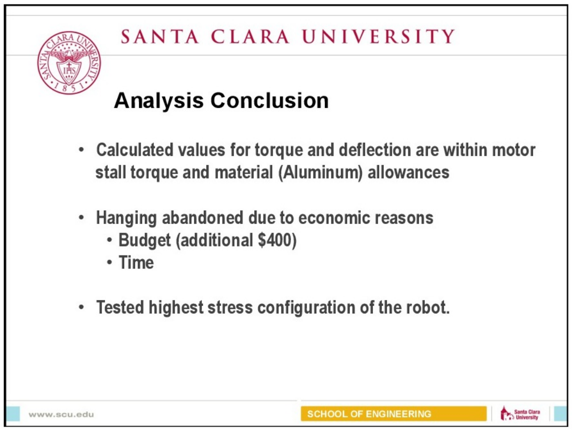
SANTA CLARA UNIVERSITY

Finite Element Analysis (ANSYS)

- Deflection
  - Max deflection
    - 0.022169 ft. (0.24 in)
- Stresses
  - Max stress in beam
    - 2.26x10^6 psf

www.scu.edu

SCHOOL OF ENGINEERING



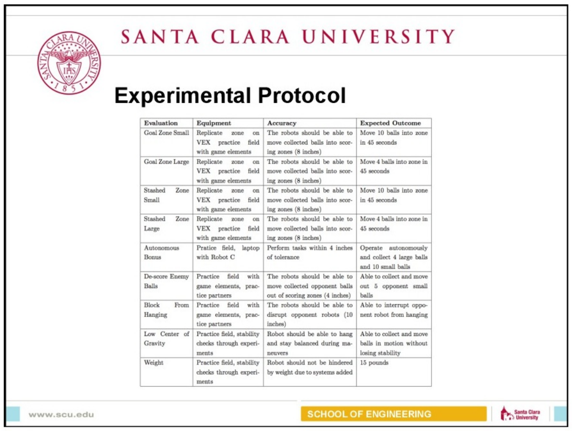
SANTA CLARA UNIVERSITY

Analysis Conclusion

- Calculated values for torque and deflection are within motor stall torque and material (Aluminum) allowances
- Hanging abandoned due to economic reasons
  - Budget (additional $400)
  - Time
- Tested highest stress configuration of the robot.

www.scu.edu

SCHOOL OF ENGINEERING



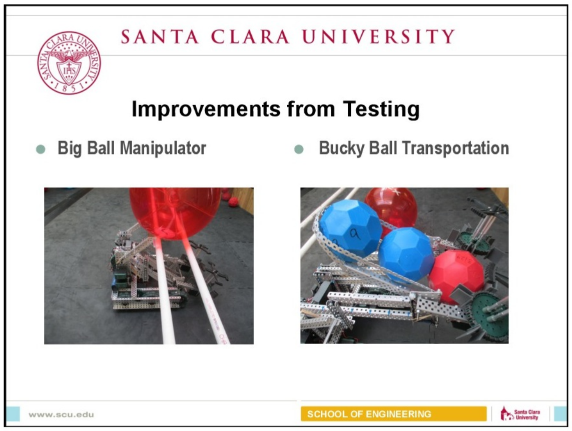
SANTA CLARA UNIVERSITY

Experimental Protocol

Evaluation	Equipment	Accuracy	Expected Outcome
Goal Zone Small	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 10 balls into zone in 45 seconds
Goal Zone Large	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 4 balls into zone in 45 seconds
Stashed Zone Small	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 10 balls into zone in 45 seconds
Stashed Zone Large	Replicate zone on VEX practice field with game elements	The robots should be able to move collected balls into scoring zones (8 inches)	Move 4 balls into zone in 45 seconds
Autonomous Bonus	Pratice field, laptop with Robot C	Perform tasks within 4 inches of tolerance	Operate autonomously and collect 4 large balls and 10 small balls
De-score Enemy Balls	Practice field with game elements, practice partners	The robots should be able to move collected opponent balls out of scoring zones (4 inches)	Able to collect and move out 5 opponent small balls
Block From Hanging	Practice field with game elements, practice partners	The robots should be able to disrupt opponent robots (10 inches)	Able to interrupt opponent robot from hanging
Low Center of Gravity	Practice field, stability checks through experiments	Robot should be able to hang and stay balanced during maneuvers	Able to collect and move balls in motion without losing stability
Weight	Practice field, stability checks through experiments	Robot should not be hindered by weight due to systems added	15 pounds

www.scu.edu

SCHOOL OF ENGINEERING



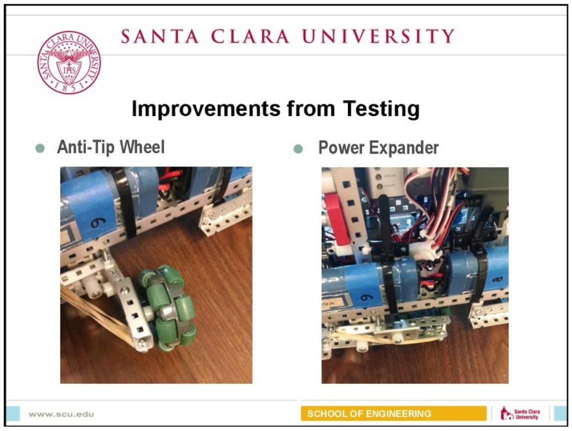
SANTA CLARA UNIVERSITY

Improvements from Testing

- Big Ball Manipulator
- Bucky Ball Transportation

www.scu.edu

SCHOOL OF ENGINEERING



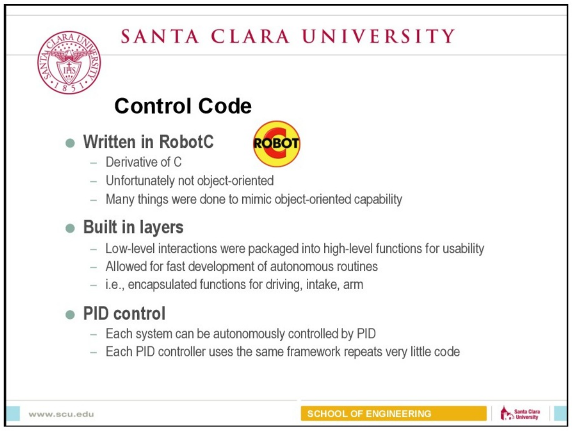
SANTA CLARA UNIVERSITY

Improvements from Testing

- Anti-Tip Wheel
- Power Expander

www.scu.edu

SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

Control Code

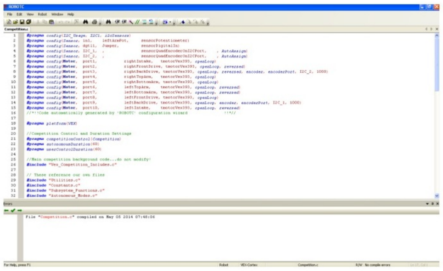
- Written in RobotC
  - Derivative of C
  - Unfortunately not object-oriented
  - Many things were done to mimic object-oriented capability
- Built in layers
  - Low-level interactions were packaged into high-level functions for usability
  - Allowed for fast development of autonomous routines
  - i.e., encapsulated functions for driving, intake, arm
- PID control
  - Each system can be autonomously controlled by PID
  - Each PID controller uses the same framework repeats very little code

www.scu.edu

SCHOOL OF ENGINEERING

SANTA CLARA UNIVERSITY

Control Code



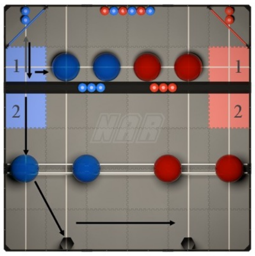
```
#pragma config(I2C_Usage, I2C1, i2cSensors)
#pragma config(Sensor, in1,    leftArmPot,     sensorPotentiometer)
#pragma config(Sensor, dgtl1,  Jumper,         sensorDigitalIn)
#pragma config(Sensor, I2C_1,  ,               sensorQuadEncoderOnI2CPort,    , AutoAssign)
#pragma config(Sensor, I2C_2,  ,               sensorQuadEncoderOnI2CPort,    , AutoAssign)
#pragma config(Motor,  port1,           rightIntake,   tmotorVex393, openLoop)
#pragma config(Motor,  port2,           rightFrontDrive, tmotorVex393, openLoop, reversed)
#pragma config(Motor,  port3,           rightBackDrive, tmotorVex393, openLoop, reversed, encoder, encoderPort, I2C_2, 1000)
#pragma config(Motor,  port4,           rightTopArm,   tmotorVex393, openLoop)
#pragma config(Motor,  port5,           rightBottomArm, tmotorVex393, openLoop)
#pragma config(Motor,  port6,           leftTopArm,    tmotorVex393, openLoop, reversed)
#pragma config(Motor,  port7,           leftBottomArm, tmotorVex393, openLoop, reversed)
#pragma config(Motor,  port8,           leftFrontDrive, tmotorVex393, openLoop)
#pragma config(Motor,  port9,           leftBackDrive, tmotorVex393, openLoop, encoder, encoderPort, I2C_1, 1000)
#pragma config(Motor,  port10,          leftIntake,    tmotorVex393, openLoop, reversed)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

#pragma platform(VEX)

//Competition Control and Duration Settings
#pragma competitionControl(Competition)
#pragma autonomousDuration(60)
#pragma userControlDuration(60)

//Main competition background code...do not modify!
#include "Vex_Competition_Includes.c"

// These reference our own files
#include "Utilities.c"
#include "Constants.c"
#include "Subsystem_Functions.c"
#include "Autonomous_Modes.c"
```

```
File "Competition.c" compiled on May 05 2014 07:48:06
```

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

SANTA CLARA UNIVERSITY

Autonomous Strategies



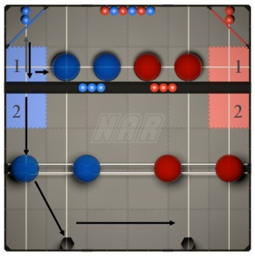
- Safety
- Bot 1:

1. Collects two small bucky balls.
2. Pushes big balls into opponents' #1 to disrupt function.
3. Knocks off bucky balls from barrier.
4. Shoots out three bucky balls into goal/middle zone.

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

SANTA CLARA UNIVERSITY

Autonomous Strategies



- Bot 2:

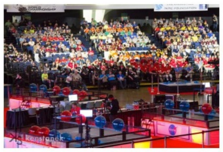
1. Sprints towards goal.
2. Knocks off big ball into goal zone.
3. Scores a bucky ball into goal.
4. Aligns itself to opponents' goal.
5. Drives straight across to disrupt opponent autonomous mode.

www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

SANTA CLARA UNIVERSITY

Competition

- VEX World Championships
- Anaheim Convention Center
- 60 Teams Internationally
- 7000+ People



www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

SANTA CLARA UNIVERSITY

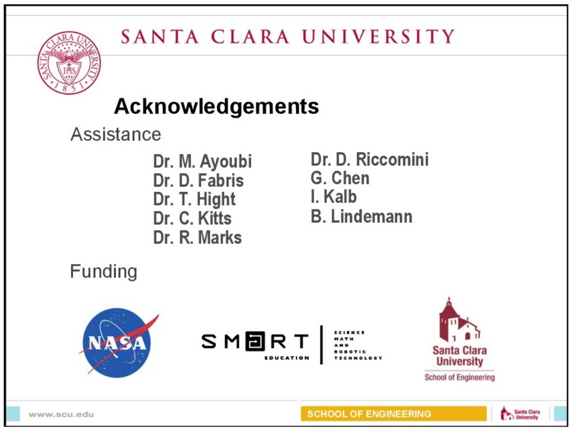
www.scu.edu SCHOOL OF ENGINEERING Santa Clara University

SANTA CLARA UNIVERSITY

Results and Conclusions

- 17th Rank After 10 Qualification Matches
  - Guaranteed Spot in Elimination Rounds
- Lightweight and Maneuverability Contributed to Success
- Autonomous Portion Crucial

www.scu.edu Santa Clara University



SANTA CLARA UNIVERSITY

Acknowledgements

Assistance

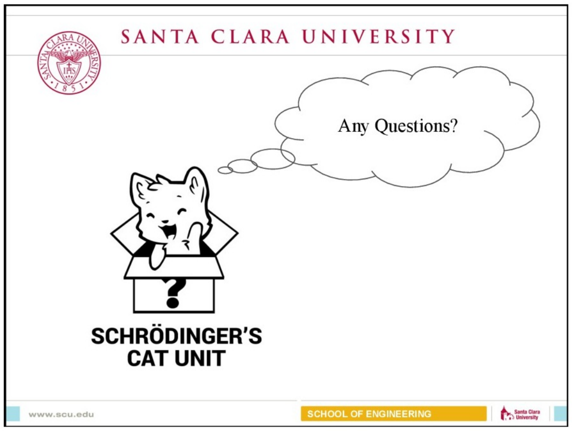
Dr. M. Ayoubi
Dr. D. Fabris
Dr. T. Hight
Dr. C. Kitts
Dr. R. Marks

Dr. D. Riccomini
G. Chen
I. Kalb
B. Lindemann

Funding

NASA
SMART EDUCATION
SCIENCE MATH AND ROBOTIC TECHNOLOGY
Santa Clara University
School of Engineering

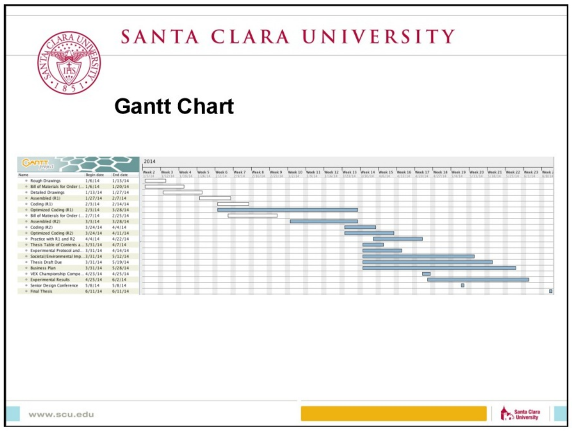
www.scu.edu SCHOOL OF ENGINEERING



SANTA CLARA UNIVERSITY

SCHRÖDINGER'S
CAT UNIT

www.scu.edu SCHOOL OF ENGINEERING

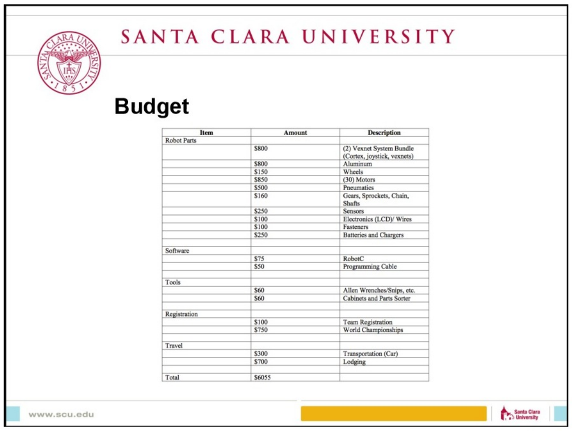


SANTA CLARA UNIVERSITY

Gantt Chart

Name	Begin date	End date
Rough Drawings	1/6/14	1/13/14
Bill of Materials for Order (...	1/6/14	1/20/14
Detailed Drawings	1/13/14	1/27/14
Assembled (R1)	1/27/14	2/7/14
Coding (R1)	2/3/14	2/14/14
Optimized Coding (R1)	2/3/14	3/28/14
Bill of Matierals for Order (...	2/7/14	2/25/14
Assembled (R2)	3/3/14	3/28/14
Coding (R2)	3/24/14	4/4/14
Optimized Coding (R2)	3/24/14	4/11/14
Practice with R1 and R2	4/4/14	4/22/14
Thesis Table of Contents a...	3/31/14	4/7/14
Experimental Protocol and...	3/31/14	4/14/14
Societal/Environmental Imp...	3/31/14	5/12/14
Thesis Draft Due	3/31/14	5/19/14
Business Plan	3/31/14	5/28/14
VEX Championship Compe...	4/23/14	4/25/14
Experimental Results	4/25/14	6/2/14
Senior Design Conference	5/8/14	5/8/14
Final Thesis	6/11/14	6/11/14

www.scu.edu

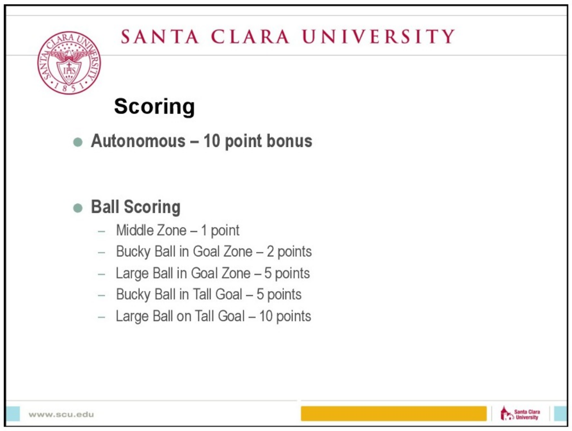


SANTA CLARA UNIVERSITY

Budget

Item	Amount	Description
Robot Parts		
	$800	(2) Vexnet System Bundle (Cortex, joystick, vexnets)
	$800	Aluminum
	$150	Wheels
	$850	(30) Motors
	$500	Pneumatics
	$160	Gears, Sprockets, Chain, Shafts
	$250	Sensors
	$100	Electronics (LCD)/ Wires
	$100	Fasteners
	$250	Batteries and Chargers
Software		
	$75	RobotC
	$50	Programming Cable
Tools		
	$60	Allen Wrenches/Snips, etc.
	$60	Cabinets and Parts Sorter
Registration		
	$100	Team Registration
	$750	World Championships
Travel		
	$300	Transportation (Car)
	$700	Lodging
Total	$6055	

www.scu.edu

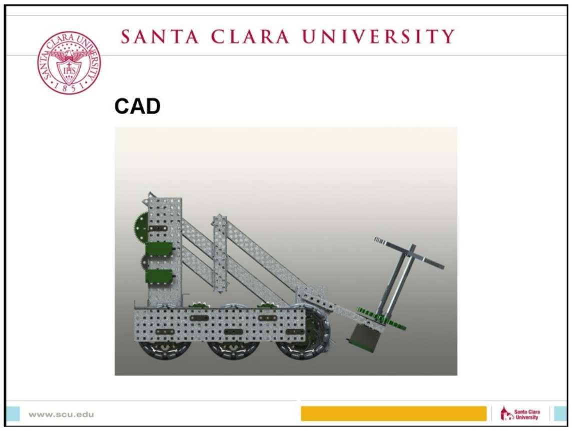


SANTA CLARA UNIVERSITY

Scoring

- Autonomous – 10 point bonus
- Ball Scoring
  - Middle Zone – 1 point
  - Bucky Ball in Goal Zone – 2 points
  - Large Ball in Goal Zone – 5 points
  - Bucky Ball in Tall Goal – 5 points
  - Large Ball on Tall Goal – 10 points

www.scu.edu



SANTA CLARA UNIVERSITY

CAD

www.scu.edu

SANTA CLARA UNIVERSITY

www.scu.edu

Santa Clara University

SANTA CLARA UNIVERSITY

www.scu.edu

Santa Clara University