The Choate Robotics Guide to VEX EDR Competition Robotics

Revision History

Revision	Description	Author	Release Date
-	Initial Release	Adham O. Meguid	08/28/16
A			

Todo:

Significant editing (grammatical, section references, etc.)
Section on online challenges

Table of Contents (can click on page number to jump)

1.0 Introduction

1.1 Document Purpose

This document is meant to be a resource and guide for future Choate Robotics team members, group leaders, and team captains. This guide aims to cover all significant aspects of robot construction, programming, and testing for competition within the VEX Robotics Design System, and not covered by VEX's Inventor's Guide. At the time of writing, the Choate Robotics team has existed for three years. Next year will be the last year that members who were on the team from its initial founding will be participating. Our current mentors, Mr. Andrew Murgio and Mr. Kyle Di Tieri, will likely be a part of the team for longer, but regardless, when next year's season comes to a close there will likely be some techniques and knowledge that will be lost until it is rediscovered by future members. Engineering Notebooks provide insight into these techniques and lessons, however, they are fairly lengthy, and having to sift through every Engineering Notebook we have ever created would take an extremely long time. The aim of this guide is to summarize and allow techniques, knowledge, and each team's experiences to be passed on, creating a team that becomes stronger every year by building upon past lessons and discoveries. There are numerous learning experiences that we have already gone through, and to lose those experiences is to lose valuable lessons for future team members.

Finally, note that rules change from year to year. If a technique or setup in this guide conflicts with the game rules of the year, **follow the game rules.**

Most of all, always have fun and be positive!

1.2 VEX Inventors Guide

We recommend reading the VEX Inventors Guide for a solid background in the VEX Robotics Design System. This guide will not cover some of the basic aspects covered in the VEX Inventors Guide (names of parts, which screwdriver to use for which screws 3/32 vs 5/64, tightening screws without stripping them, how to use nylocks, basic motor knowledge, how and where to use collars and bearing flats, speed vs torque and gear ratios/compound gearing, wheel choice, basic motor gear replacement, etc), as it is geared towards giving general team and competition advice, and providing the reader with lessons learned from past years. The Inventors Guide is very useful, and is a fairly brief read with all of its diagrams and clear explanations, though keep in mind some information may be out of date. For example, the Cortex in the Inventor's Guide is an older model, and we no longer use radio frequency (RF) control with the Cortex, so do not read the control section (unless RF interests you!). This guide assumes knowledge of the Inventor's Guide. Finally, VEX's Curriculum Lesson Content: http://curriculum.vexrobotics.com/curriculum
Provides **a lot of very useful content**, with topics such as:

- Unit 1: Introduction to Engineering

- Unit 2: Introduction to Robotics
- Unit 3: Introduction to VEXnet
- Unit 4: Introduction to Autodesk Inventor
- Unit 5: THE GAME!
- Unit 6: Object Manipulation
- Unit 7: Speed, Power, Torque & DC Motors
- Unit 8: Mechanical Power Transmission
- Unit 9: Drivetrain Design
- Unit 10: Lifting Mechanisms
- Unit 11: Systems Integration
- Unit 12: Testing and the Iteration Process
- Unit 13: Design your Own Part [optional]

This curriculum also provides useful competition examples and contexts. You'll see a few examples from this curriculum throughout this guide.

1.3 The Vex Forum

The website http://www.vexforum.com/ is an extremely useful resource. The forum is a great place for gathering ideas and learning about different types of robots, drive bases, lifts, and intake mechanisms. Robot reveals, problem troubleshooting, drive base or lift construction, useful programming ideas; the forum provides a wealth of information that can help to brainstorm or think through a concept. There is a lot of valuable knowledge and advice that is likely applicable to mechanisms the team is considering or constructing. To give a sense of scope, there's even a thread on a very well done planetary/differential/pneumatic hybrid drive base! However, while the forum is very useful, it should not be your only source of information. If you are considering a mechanism, the forum can be useful to prove its viability, however, it should not be used to fully discount any ideas. For example, we have constructed a well functioning drive base which the forum indicated should burn out (stall) fairly frequently. Finally, you can also look at this document, 101 Things I wish I'd Known Before My First VEX Tournament, for even more information:
http://www.roboticseducation.org/documents/2013/06/101-things.pdf

2.0 The Engineering Process

2.1 The Engineering Notebook's Importance

The Engineering Notebook is a critical part of the design process. While it can appear to be a nuisance using up precious development time, and while it can be an easily forgotten part of building a robot, it is an essential part of any professional and experienced team's engineering process. It documents brainstorming sessions, decisions and why they were made, design discussions, problem-solving, and every aspect of the development of a robot. The Engineering Notebook allows all team members to be on the same page, working with the same level of knowledge of the robot, and allows teams to look back at decisions, understand why they were made, and avoid repeating mistakes or gain new insights. It also allows teams to understand how to repair the robot, rewire, and even how to rebuild the robot.

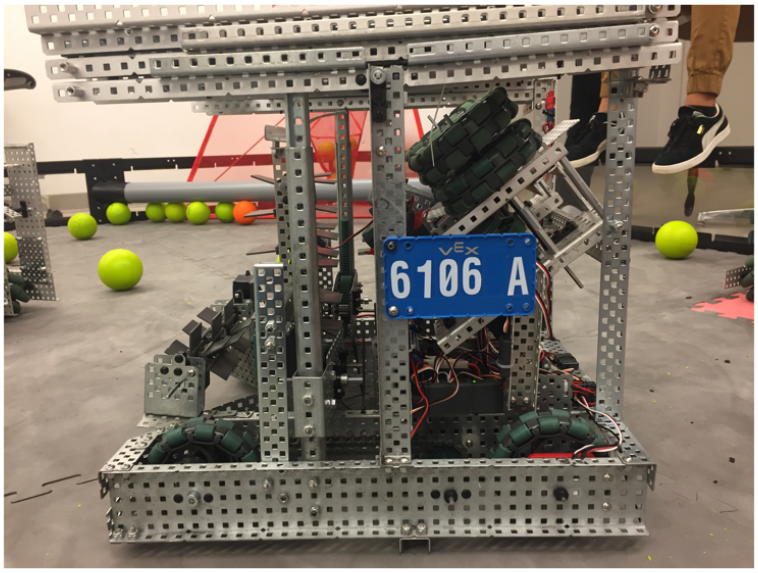
2.2 The Engineering Notebook Format

The Engineering Notebook's format will undoubtedly differ from year to year, but we found particular success this year with the format we used. While the method of recording the Engineering Notebook may change (we used Google Docs this year for collaboration, and later transferred those pages to a Microsoft Word document for a few extra features before printing), there are, of course, numerous aspects to include in a good Engineering Notebook.

2.21 The Cover Page and Index

Judges will be judging the Engineering Notebook from the moment they pick it up. While not specifically listed on the judges' rubric, the cover of an engineering notebook is a first impression, and can affect overall opinions. At a minimum, the cover should identify your team number and team name, and the competition season year and game. An example cover can be found below:

CHOATE ROBOTICS 6106A



Engineering Notebook

2015-2016 Season

VEX ROBOTICS COMPETITION

NOTHING BUT NET

2.22 The Table of Contents

We recommend having a table of contents in place for the notebook. Once again, while not strictly a requirement, a table of contents helps judges identify everything included in your notebook, and gives a sense of organization and professionalism. You can create the table of contents in multiple ways, for example using Microsoft Word's variety of header levels for each section title throughout the journal (Heading 1, Heading 2, etc.), and then inserting a table of contents with its built in table of contents feature. Regardless, the table of contents should allow judges to find different entries by their page numbers (usually recorded as a footer). Entries should be dated, and titled something descriptive and useful, and the date and title should be featured in the table of contents.

Additionally, this last year we actually color coded our entries and put a key in the header to show what each entry's color indicated. More information on that system can be found in 2.23: Our design process.

2.23 Introduction Section

An Introduction Section, once again not strictly required, offers a chance to show the judges your team and justify your design principles. We typically split the Introduction Section into a few parts

- About 6106_(team letter here): We like to put a brief introduction to our team here, such as:

  > We are 6106B, also known as Choate Robotics. We are a group of students from Choate Rosemary Hall, a private boarding school in Wallingford, Connecticut. There are currently 14 members on the team, with two mathematics teachers from our school, Mr. Murgio, and a new teacher Mr. Di Tieri, as our teacher mentors. Our team was founded in the 2013-2014 season, Toss Up, and made it to Nationals in our first year.

  We also like to put a meet the team, and meet the mentors section, with pictures of each team member, and their name and roles listed beneath their picture.

- Our Engineering Notebook: If using an online journaling format, it is often important to justify that decision, as judges may expect a completely hardcopy format. This section should also underline your format for each journal entry. An example is seen below:

Our Engineering Notebook

Our team strives to document every step of our journey, from our beginning brainstorms, to our fully built and field-tested robot. We document every challenge and solution throughout our season. We use Google Docs to journal our entries, then compile those into our final engineering notebook.

While Google Docs is not the same as a traditional engineering notebook, we set standards to ensure that when we journal on Google Docs, we follow the same principles as if we were journaling in an engineering notebook. We do not edit past entries, and if we do spot an error or wish to change a past entry, we will signal this change by writing in red text, and leaving the original text in place (with ~~strikethrough~~).

Google Docs allows our team to effectively communicate and journal whether we are in the lab upstairs, or on our test field in the basement.

Regarding coding in the Engineering Notebook, we usually do not include code in the Engineering Notebook. We do include autonomous plans, within the strategy set section, however putting all changes to the code in the notebook would quickly clutter it. **However,** we do use **source control** (GitHub) to keep track of all of our code changes and allow us to revert to any version, functioning both as a **journaling process, and as a backup method.** If you would like to see our GitHub page (username ChoateRobotics), please let us know!

Finally, **all ideas based upon or inspired by other sources are always credited.** However, we believe in building upon past ideas, and do not simply copy others' ideas.

Our team uses a standardized procedure for our journal entries.

1. Time Spent: The time we worked on the robot
2. Journaller: Who is creating the journal entry
3. Team Members: All of the team members involved in the work
4. Parts Used: A list of all parts used
5. Description of work: A bullet point description of work. Each top-level bullet point is a goal statement, and lower level bullets show what we did to fulfill that goal, how well that goal was fulfilled, what challenges came up, and future ideas and work.
6. Todo: A bullet point list of what to work on at the next meeting, allowing us to set our next goals and pick up right from where we left off.

7. Pictures: For particularly major changes, we take pictures of the development. We also take pictures of any sketches or brainstorming sessions. Finally, we take pictures gradually over time, to keep a good track of what our robot looks like

---

- Our Design Process: This past year, we used a color-coded table of contents system, color coding entries according to what type of development occurred. We also put a key for the different colors in the header of our table of contents section, such as the one seen at the top of this page (use section breaks to differentiate this header). An example of this section is seen below:

Our Design Process

Research and Design
During meetings focused upon research and design, our team examines game objectives, explores potential new mechanisms, searches the Vex Forums for posts, or researches engineering or scientific principles which apply to our work. We use white boards for conveying our ideas and in discussion, and analyze the benefits and flaws to various approaches.

Modeling and Prototyping
During meetings focused upon modeling and prototyping our team typically builds very basic designs which serve as a proof of concept for an idea. We typically use the sketches we created during our white board research and design as a basis for the prototype

Building
If a prototype is deemed effective, efficient, and feasible, we finalize the idea and begin building the prototype to be used on the final robot. We work on and finalize each section of our robot, drive base, intake, shooter, elevation mechanism, independently, then bring the components together in our final build. However, we always attempt to keep in mind whether different prototypes will fit well with one another.

Testing and Programming
After the robot has been built, our team begins testing and programming. The various mechanisms are tested in a game scenario, including time limits and field setup, and we develop our robot code and autonomous. Testing and programming also may sometimes encompass driver's practice, though these practices are usually separate

Reflection
After every competition, our team feels that it is important to take a step back, and reflect upon our performance. We discuss what problems we had, what we felt could have gone better, and how we want to proceed to further improve our robot.

Finally, every day that we meet to make progress, we first set initial goals for the day. These goals guide our meeting tasks, and we then progress to achieve those goals, and journal our progress towards them, problems reaching them, and potential future work.

- Our Season Goals and Competitions: This introduction section should talk about the goals the team has for this year's season, and the competitions that we will be attending.

And that concludes the 4 parts of the Engineering Notebook's introduction.

2.24 **Important Special Entries**

These are entries that judges specifically look for when evaluating an Engineering Notebook, as seen by the rubric in section 2.26. One of these entries may be put in the introduction section, and this entry is some form of overall plan. **A Calendar or list of project deadline dates** would show an excellent planning process. Ideally this calendar or list of deadlines would show that we leave at least a week for drivers practice and autonomous development, etc., and it would also show adjustments to the project timeline. We did not implement this in our past Engineering Notebook, so this is one notable potential improvement!

The first entry of the Engineering Notebook should be a description of this year's challenge, in both words and pictures. We recommend using pages from the year's game manual for pictures, and doing additional description with words. The team should also describe the team's goals towards achieving the challenge (what they want in a robot). We also recommend putting a quick section in about particularly important or new **rules** for the game, for example:

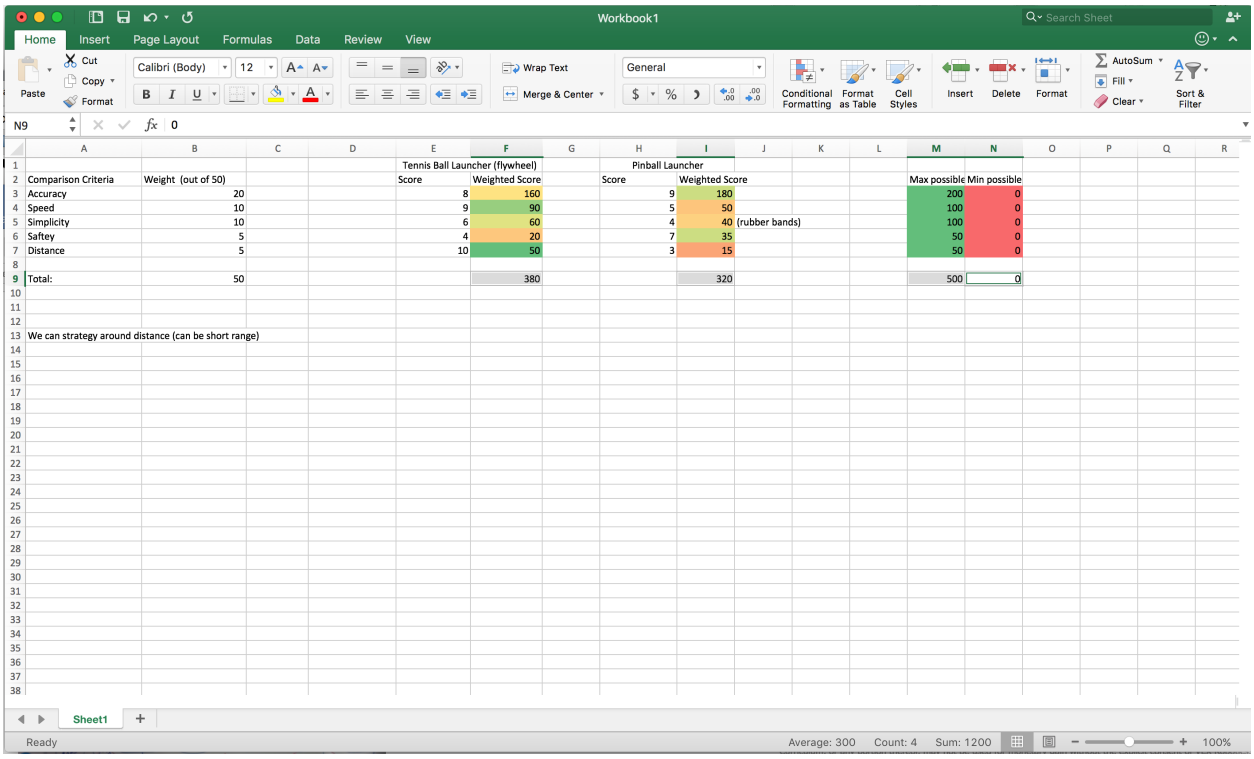
Rule Discussion:

We discussed the new rule-set, including the restrictions on vertical extension. In particular we examined:

> <G12> a.
> If a Robot has expanded beyond its 18"x18"x18" size limitation and is outside the Climbing Zone it is responsible for any type of Entanglement that occurs with an opponent. If an expanded Robot becomes Entangled while fully within its Climbing Zone, its opponent would be responsible
>
> <G13> Robots must be designed to permit easy removal of Scoring Objects from any mechanism without requiring the Robot to have power after a Match.
>
> <G14> Field tolerances may vary by as much as ±1", except where otherwise noted, so teams must design Robots accordingly. Please make sure to check Appendix A for more specific tolerances. Note: The foam field tiles should be entirely within the field perimeter. The field perimeter should not be resting on top of the foam field tiles.
>
> <SG6> Any Scoring Objects introduced during the Match as Driver Control Loads must be either gently placed on a Robot of your own color touching the Loading Zone or gently entered into the Loading Zone of your own color, by a Student Drive Team Member, without breaking the plane of the field, during the Driver Controlled Period. The intent of this rule is to allow teams to introduce objects into play, but not to impart energy on the Scoring Object which will cause it to end up in a position outside the Loading Zone.
>
> <SG7> Robots may not enter the opposing Alliance's Loading Zone at any time during the Match
>
> ***<SG8> Robots may not Possess more than four (4) Scoring Objects at once***
>
> <SG9> Robots may not enter (i.e. break the plane) of any Goal
>
> <SG10> Robots may not be in the opposing Alliance's Climbing Zone during the last thirty seconds (0:30) of the Match. Furthermore, during the period, Robots may not contact an opposing Robot that is contacting a partner Robot that is fully within the volume of the Climbing Zone.
>
> <SG11>Robots may not be in the opposing Alliance's Climbing Zone during the last thirty seconds (0:30) of the Match. Furthermore, during the period, Robots may not contact an opposing Robot that is contacting a partner Robot that is fully within the volume of the Climbing Zone.

Next, the team should show its brainstorming process, as covered by section 2.3. An additional recommendation for the brainstorming process is a Weighted Objectives Table (WOT). Vex provides a very useful explanation found here: http://curriculum.vexrobotics.com/appendices/appendix-7. You may consider using a "traffic color diagram" in conjunction with this table, with green box fills being assigned to criteria closer to the max score, yellow closer to the middle of the scoring range, and red when closer to the lowest score for the scoring range, and various shades in between if desired. Regardless of the brainstorming method, the team should have concrete reasons for why the selected approach was chosen verses the alternatives, and those should be clearly shown in the Engineering Notebook. Diagrams and tables are often the most effective means of showing this process to the judges.

An example of a WOT combined with the traffic color scale is seen below, using Microsoft Excel and its conditional formatting. (Selecting all scores of 1 category to be compared, then selecting the theoretical max and min score, going to conditional formatting, color scales, then green-yellow-red color scale. The sections that would be copied to the notebook are the comparison criteria, and the score and weighted score for each idea.



					Tennis Ball Launcher (flywheel)			Pinball Launcher						
	Comparison Criteria	Weight (out of 50)			Score	Weighted Score		Score	Weighted Score				Max possible	Min possible
	Accuracy	20			8	160		9	180				200	0
	Speed	10			9	90		5	50				100	0
	Simplicity	10			6	60		4	40	(rubber bands)			100	0
	Saftey	5			4	20		7	35				50	0
	Distance	5			10	50		3	15				50	0
	Total:	50				380			320				500	0
	We can strategy around distance (can be short range)													

Image 1: First, select the weighted score of all ideas being represented, **and** the maximum possible and minimum possible weighted score.

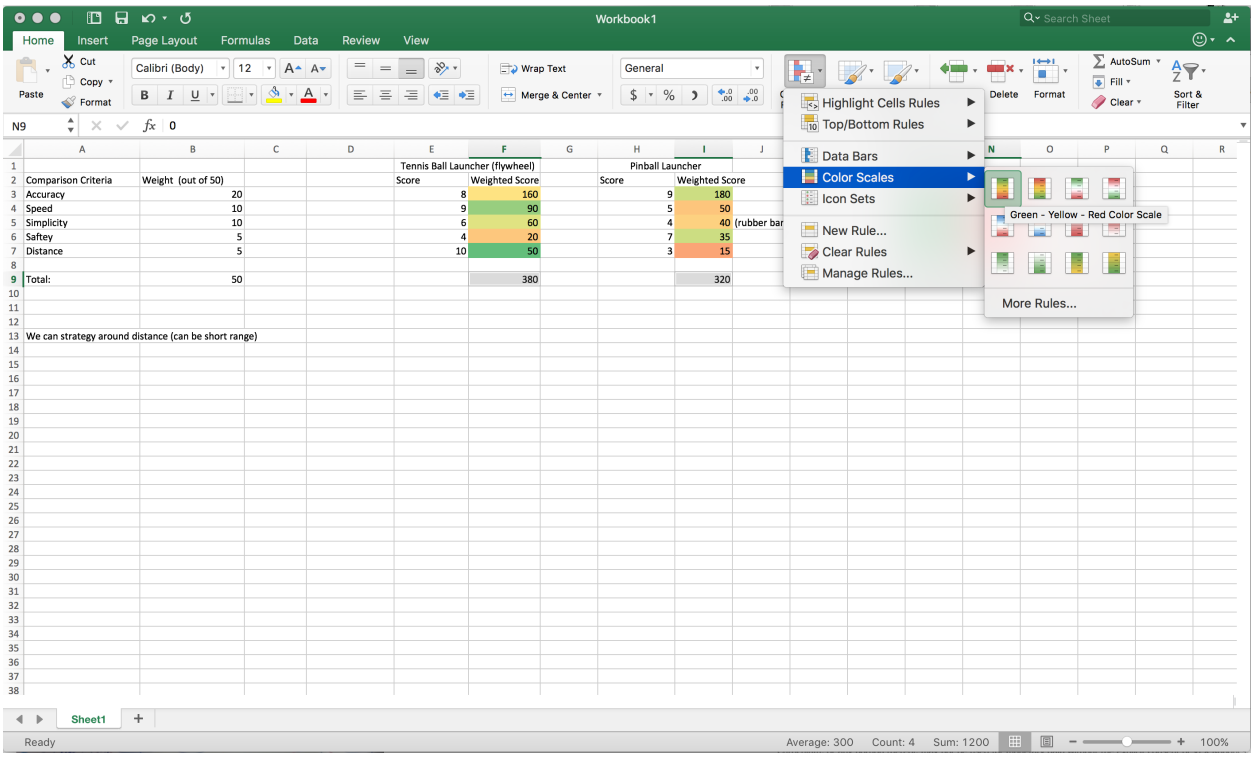


Image 2: Next, navigate to Excel's color scales menu, and choose the Green – Yellow – Red Color Scale.

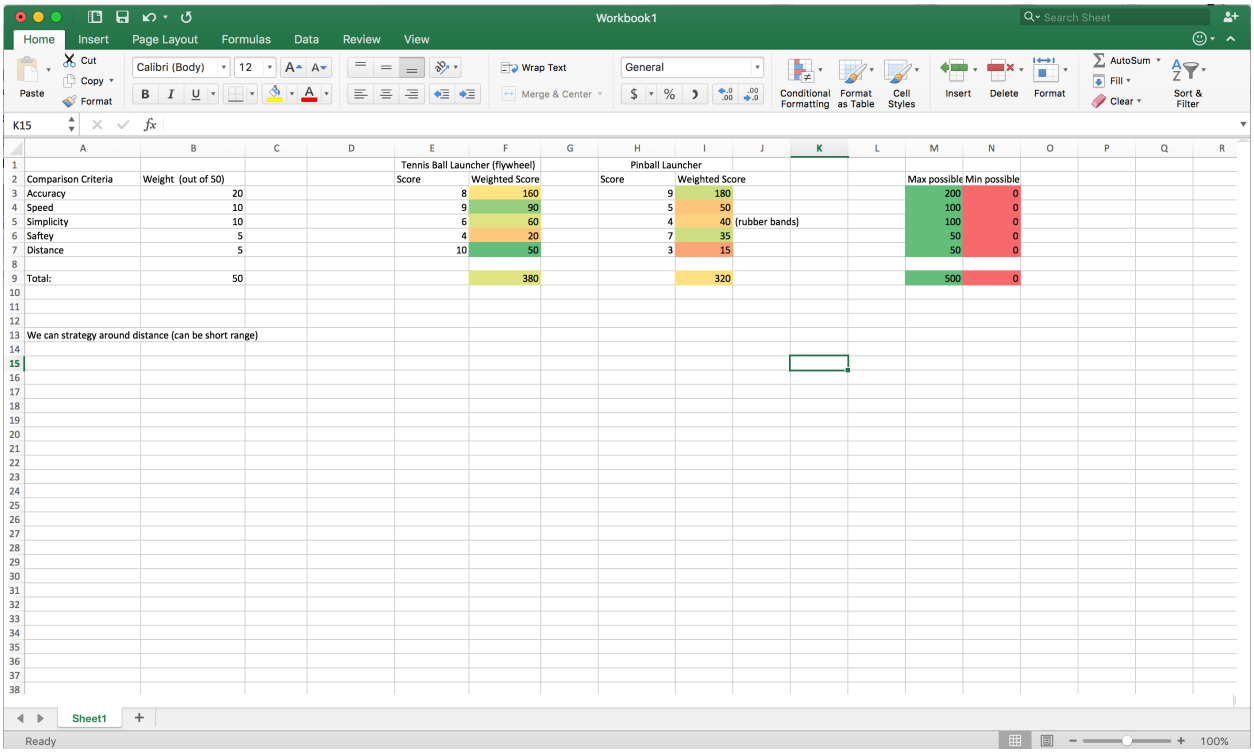


Image 3: Finally, the weighted score of the ideas in each comparison criteria should have a color shade.

Then, in the notebook, the table would look something like this:

Our weighted objectives for the launcher: **(accompanied by a bulleted explanation for each one, ex: distance is lower because we can design our strategy around that to an extent, become a short range field-robot, etc).**

Comparison Criteria	Weight (out of 50)
Accuracy	20
Speed	10
Simplicity	10
Safety	5
Distance	5
Total:	50

Our potential launcher ideas: **(preceded by a description and sketches of all of these ideas from the brainstorming session)**

	Tennis Ball Launcher (flywheel)			Pinball Launcher	
	Score	Weighted Score		Score	Weighted Score
Accuracy	8	160		9	180
Speed	9	90		5	50
Simplicity	6	60		4	40
Safety	4	20		7	35
Distance	10	50		3	15
		380			320

2.25 Entry Format

Judges like to see an organized, consistent, entry format for every development that includes goals, pictures, and tests and redesigns. We talked about a theoretical format in the "Our Engineering Notebook" subsection of 2.23, and below are a few example entries.

Final Construction and Competition Preparation, Friday November 13th

Time Spent: 3:00- 9:00pm

Journaller: Adham, Katrina
Team Members: Katrina, Adham, Elise, Nandini

Parts Used:
Two 35 length L-bars (to replace the two 1x2x1x35 C-channels), standoffs of assorted sizes

Description of Work:

- Goal: Fix Diagonal Conveyor
  - Got rid of C-channels, replaced with L-bar system, removed tilt of the conveyor system, also decreased compression at the two ends of the conveyor
    The flywheel is now functioning quite well!
- Goal: Driver's Practice
  - Nandini is driving, and doing very well! Scoring 100 points in drivers control consistently.
- Goal: Fix ramp so extends all the way
  - Could not fix the ramp so that it fully extends when we activate it, and does not extend too early in the match (from a potential shove). Will need to fix either at the competition, or not elevate, or find a partner who can get over the gap
- Goal: Get robot ready to be inspected
  - Grinding and sanding of edges, getting robot into size limitations (was out by very small measurements)
- Goal: Packing!
  - Packed up supplies and robot, ready to go for the competition tomorrow!

Pictures: (not included in this guide to conserve space, just pictures of the robot.)

Elevation Discussion Part 2, Saturday March 26

Time Spent: 3:00 pm- 6:00pm

Journaller: Ausar
Team members: Katrina, Adham, Max, Ausar, Elise, Nandini, Weston, Nikhil, Kristen

Parts used: N/A Today was discussion

Description of Work:

- Goal:Met as a larger group to talk about elevation and its incorporation on the new robot
    - Went through the pros and cons of each of the elevation ideas that we have in order to select one idea and start building it.

	PROS	CONS
Forklift/Pulley	• Can easily accomplish a low elevation • Mixture of all ideas given • Original Idea • Lots of Torque • Easy to Attach • Takes A Little Space • Almost no reliance on partner	• Need to learn how to use the vex pulleys and the math behind it (extra time)
Ramp	• We have previous experience • Quickest to implement • No motors needed • No counterweights needed	• Have not had good previous experiences with current ramp • Could fail to work based on how low partner's drive is • The steepness of the ramp could cause some robots to fail to drive up it
“Cyber Brains” Type Lift	• Have experience building a system similar to it • Very little reliance on partner • Can be made with either pneumatics or motors • Very Fast Elevation • Looks really cool	• Not a lot of space for it • Not much idea on how to use pneumatics effectively • Only a basis for how to build, but that won't get most of the lift to work
Self Elevation	• Should be able to elevate on almost every robot • Does not take up much space at all • Simple to implement on robot	• Not fully sure if it is legal since it hooks on the outside of the wall • Need to land on a flat robot • Would not be able to get a full 18in of the ground which would be problematic with larger partners

- Based on the pros and cons of each of the elevation ideas we decided to use a forklift/pulley type elevations
    - Have two pulleys on the front and back of a platform in order to make a small ramp
        - Then pull the ramp up 8 inches in order to reach high elevation

- Use a rachet and pawl possible for a locking mechanism

Regarding coding in the Engineering Notebook, we usually do not include code in the Engineering Notebook. We do include autonomous plans, with the strategy set section, however putting all changes to the code in the notebook would quickly clutter it. **However,** we do use **source control** (GitHub, see section 5.9) to keep track of all of our code changes and allow us to revert to any version, functioning both as a **journaling process, and as a backup method.** When talking with judges you often want to bring this point up, and it should be mentioned in the notebook (for example in the introduction section).

2.26 The Design Award Rubric

Please find below the Design Award Rubric which judges use to evaluate a team and its Engineering Notebook! This is a very useful resource which we have designed our notebook around.

Design Award Rubric

Team # _____

Judges _____

For Design Award details, review the Awards Appendix on http://www.roboticseducation.org/vex-robotics-competitionvrc/game-day-running-an-event/

Directions: Mark the descriptor that best describes the team's performance for each criterion. Write the best features of the team's Engineering Notebook and Student Interview and Discussion on the back of this page.

Engineering Notebook: The notebook...			**See Student Interview and Discussion Criteria on Next Page**	
Criteria	**Expert (3 points)**	**Proficient (2 points)**	**Emerging (1 point)**	**Points**
Design Process: Challenge	Describes the challenge at the beginning of the notebook with words and pictures and states the teams' goals toward accomplishing that challenge.	Identifies the challenge at the beginning of the notebook.	Neglects to clearly identify the challenge.	
Design Process: Brainstorming	Generates an extensive list of possible approaches to the challenge with labeled diagrams.	Provides an extensive list of possible approaches to the challenge.	Contains a very short list or does not list the results of the brainstorming sessions.	
Design Process: Select Approach	Explains why the selected approach was chosen and why the other alternatives were not chosen.	Explains why the selected approach was chosen.	Does not document why the team selected the approach they did.	
Design Process: Build & Program	Records the building and programming process in such detail that someone outside the team could recreate the robot by following the steps in the notebook.	Documents the key steps in the process of building and programming.	Seems to skip some important steps in the process of building and programming the robot.	
Test & Redesign	Describes in great detail the process of troubleshooting, testing, and redesigning through all iterations (cycles) of the process.	Captures the key results of the troubleshooting, testing, and redesign cycle.	Leaves out important information about the troubleshooting, testing and redesign cycle.	
Usefulness	Is such a detailed account of the team's design process that the reader could recreate the project's history. It is a useful engineering tool. It contains evidence that team made decisions about design process based on previous entries. The team can explain why the notebook is organized the way it is.	Is a complete record of the process, documenting the key events of each work session. It is organized in a way that any team member can locate needed information.	Is missing, or lacks the detail needed for the reader to understand the team's history, and/or is not organized in a way that an outsider can make sense of it.	
Resources	Shows the team's efficient use of time with an overall project timeline. The team uses checkpoints to help them know how well they are staying on schedule and readjusts their schedule as needed. The notebook illustrates the good use of human resources by assigning members roles based on their strengths.	Documents the team's efficient use of time with planning and goal-setting for each day's session. It shows that the team used its human resources wisely by assigning members specific tasks.	Does not provide evidence of the team's wise use of the team's time or talents.	
Teamwork	Provides evidence that all team members were consistently involved in the process, that individual team members were self-directed enough to finish what needed to be done, and that all team members consistently shared ideas and respectfully considered each other's input.	Shows that all team members' were involved in the process, that members could be counted on because they did what they were supposed to, and that the whole team shared ideas and supported ideas of others.	Suggests that perhaps some team members did most or all the work, that one or more individuals had to be nagged or reminded to do their work, and/or that some team members did not contribute ideas or that their ideas were not considered.	

VRC Design Award Rubric 7/10/2016

2.27 The Design Award Rubric Part 2: The Interview

When judges find well-done Engineering Notebooks, they will often ask a team to interview (or sometimes they ask all teams to interview). The team should practice for judging just like you practice with your robot. For example, have a mentor run a mock judging session. Each person should know the whole story of your team and your robot, and should be able to point to their contribution to the robot. Additionally, members should be able to describe the goals of their design process and how the team accomplished these goals, should be able to describe and show (in the notebook!) our brainstorming session and multiple considered strategies and mechanisms. The team can point to their strategy set section (2.28) and explain why they chose their current strategy. Each member should be able to **independently** answer judges' questions! Therefore every student should know about the robot, the journal, and their contributions.

Finally, team members should be respectful and courteous to the judges and to each other! Ensure that you do not interrupt each other (or the judges!), and if you do accidentally apologize and let the other person continue.

And overall, when talking to Judges be enthusiastic and coherent! (This line is from 101 Things You Should Know Before Your First VEX Tournament)

Please find the Design Award Rubric for the Judge Interview below.

Design Award Rubric

Team # _____

Judges _____

Student Interview and Discussion: During the interview...			See Engineering Notebook Criteria on Previous Page	
Criteria	**Expert (3 points)**	**Proficient (2 points)**	**Emerging (1 point)**	**Points**
Design Process	Students describe the goals of the design process and how the team accomplished the challenge.	Students provide possible goals of the design process but do not clearly identify how team accomplished the challenge.	Students neglect to identify any goals of the design process and cannot describe how the team accomplishes the challenge.	
Design: Methods & Strategies	Students describe multiple design methods and strategies considered; explaining both how and why the current design strategy was selected	Students only describe their current design methods and strategy; explaining only one of either how or why the current design strategy was selected	Students do not describe any of the design methods or strategies considered; do not explain why or how the current design strategy was selected	
Team Work: Contributions	Students explain how each team member contributed to the design and strategy.	Students explain how some team members contributed to the design and strategy.	Students only explain how 1-2 members contributed to the design and strategy.	
Interview: Individual Contributions	All students independently answer the Judges' questions.	Students support each other as needed to answer the Judges' questions.	Students rely on one or two members to answer all the questions.	
Interview: Professionalism	Students present their answers in a respectful and courteous manner to the Judges and other team members, making sure each team member has a chance to contribute and waiting to speak until the other person has finished.	Students present their answers in a respectful and courteous manner to either the team members or the Judges.	Students do not present themselves in a respectful and courteous manner.	
		Total the number of points earned from Student Interview and Discussion:		
		Total the number of points earned from Notebook:		
		Total the number of points combined:		

The REC Foundation thanks Northeastern State University, Oklahoma teacher training program for developing these rubrics.

Comments:

2.28 The Strategy Set Section

As covered in 6.1, there should be a final Strategy Set section within your notebook (potentially marked off from the rest of the Engineering Notebook with a binder divider). This section should use diagrams (drawings on the field page of the game manual, sketches, etc.) to illustrate a variety of strategies that your Drive Team may implement in a match. The strategy set should cover scenarios such as what to do if we are behind the opposing team near the beginning of a match, or how to counter opponents which perform better than us, etc.

Additionally, the Strategy Set section should cover all autonomous strategies as well, similarly by using arrows and drawings on the field page of the manual, sketches, etc.

2.29 Wiring Diagram

We highly recommend having a wiring diagram page in both your Engineering Notebook, and competition notebook. This page bookmarked

2.3 Brainstorming and Design Discussion Principles

Everyone has a voice. This means listening to and taking everyone's ideas or input seriously, patiently explaining ideas, and ensuring that nobody dominates a conversation. The team captain (for most discussions) or a group leader (for individual group discussions which the captain cannot attend) is responsible for ensuring this crucial part of the design process is met. **Keep in mind, letting unique, crazy, or bizarre ideas develop during brainstorming sessions is the most likely way the team will come up with new and innovative solutions!** Brainstorming at the beginning of the year is usually a full team endeavor. This movement should begin with an overview of the year's game, watching the video, talking a bit about past robots for new members, and discussion of this year's rules and any standouts (EX: No expansion out of the size limit, new motor/pneumatic cap, etc.). **The team needs to know the rules of the game in order to effectively brainstorm, design, and compete. Therefore, each team member should fully read the year's rulebook (including any revisions made over the course of the season).** After discussing the game and the rules, the team should begin to decide what is really key to a robot. In general, the team can narrow it down to a few modules. For example, the wheelbase, an intake, a scoring mechanism/launcher, and sometimes a lift. These modules will depend on the year's game.

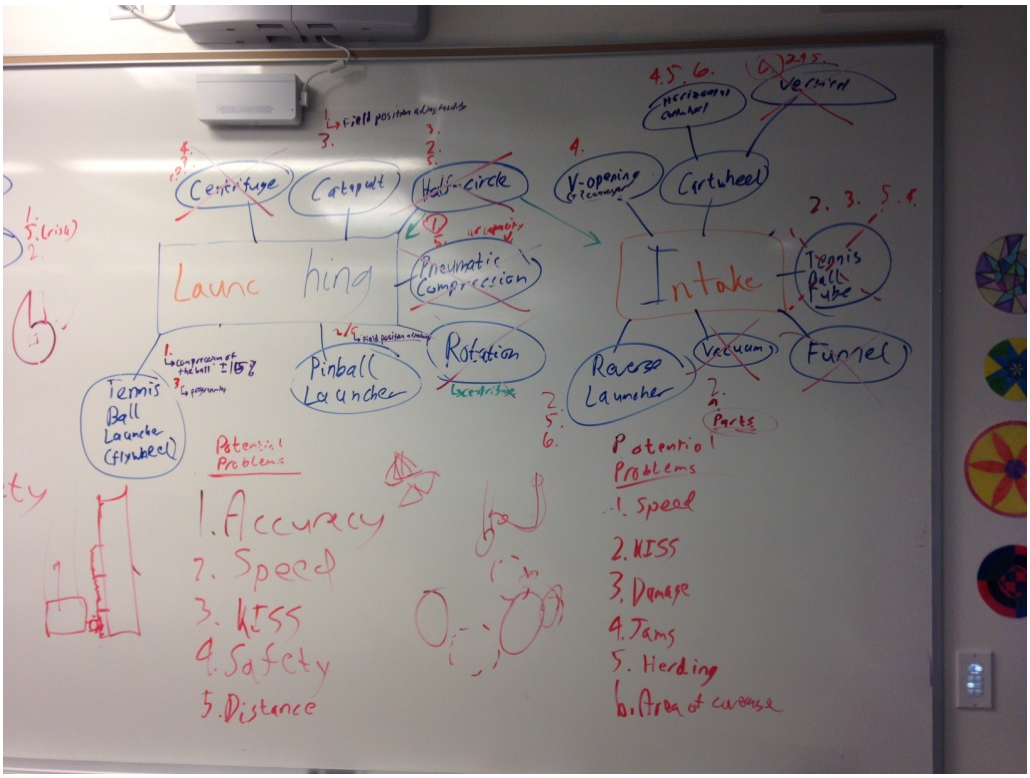
It is also important to set a few rules/guidelines for the brainstorming session. We found this list below (from [http://curriculum.vexrobotics.com/curriculum/intro-to-engineering/engineering-design-teams)](http://curriculum.vexrobotics.com/curriculum/intro-to-engineering/engineering-design-teams) to be particularly useful.

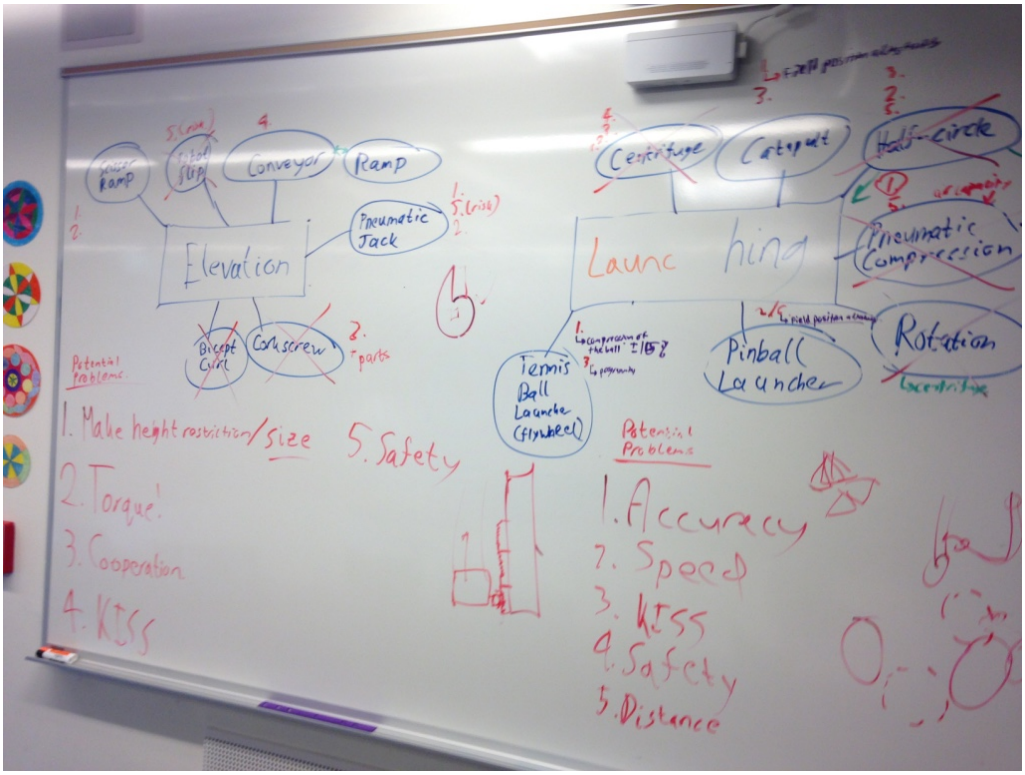
- One should always keep an open mind. It is important to allow crazy ideas to develop. The most likely time for a creative solution to be found is early in the design process when wild ideas are expressed.

- No one should become overly attached to any single idea - especially one they created. It is easy to become blinded to other ideas simply because "they aren't mine."
- One should not become defensive regarding the opinions of others. Defend one's own opinions and ideas but always focus on the ultimate goal of providing the best solution possible.
- One should always stay positive, even when discussing negatives.
- Engineering is based in logic. One should focus on factual arguments, not those based on opinions. Emotion should not be allowed to interfere with the process.
- It is important not to be offended if disagreements occur, even if things get heated and criticisms are overly harsh. Most engineers get passionate during design discussions and will often be very blunt. It is important not to take this personally.
- An unjustified opinion is not useful. Team members must be able to describe WHY they like or dislike something.
- *This is NOT rhetoric, it is engineering.* In rhetoric, the person who argues best will be most persuasive. In engineering, the person who has the best argument will be most persuasive. It is not the one who can speak the best but the one who can provide quantitative proof that will win an argument and prove their idea is better! It is important to be quantitative wherever possible.

From here, it should be noted that **brainstorming is particularly great for allowing new members to become engaged right from the start of the year.** If experienced members begin pitching ideas, new members are likely to just go along with them. That's why we may separate the new members and have a group of new members just brainstorming ideas for the year's key modules. This decision is ultimately up to the year's leadership. This separate brainstorming keeps new members engaged before they have any experience, and allows them to see their ideas become implemented. When doing separate brainstorming, it is often useful to list out modules, and to come up with potential problems for each module. Then, write out all ideas in a "web", and ask new members to list which potential problems apply to which ideas. Finally, ask new members to eliminate ideas that have too many potential problems, or are infeasible.

An example can be seen below, from idea brainstorming from the 2015-2016 season Nothing But Net. The modules were elevation, launching, and intake (wheelbases was not as major a consideration). Each module is boxed and has potential ideas (which smaller groups of new members came up with and diagrammed on boards earlier with a little guidance from the captain, then "pitched" to the group of all new members) in circles coming off from the module. Listed below are potential problems, which the captain worked with new members to come up with. Then the corresponding number for each potential problem was listed next to ideas, and any idea with too many potential problems, or with a potential problem too drastic to be addressed, was crossed off. The top three ideas from each category were then brought to the team as a whole (including experienced members). At that point, members separated into groups. The groups (which become 6106A, 6106B, 6106C, etc.) were organized by the team captain, based upon giving equal experience levels to each group, personality match-ups, and ensuring each group had some members in the afternoon activity (robotics as a sports exemption). The group formats are apt to change from year to year (EX. Having a group of new members led by 1 more experienced member, etc.).


Launching
Centrifuge
Catapult
Half-circle
Pneumatic Compression
Rotation
Pinball Launcher
Tennis Ball Launcher (Flywheel)
Intake
Horizontal
Vertical
V-opening
Cartwheel
Tennis Ball Tube
Funnel
Vacuum
Reverse Launcher
Potential Problems
1. Accuracy
2. Speed
3. KISS
4. Safety
5. Distance
Potential Problems
1. Speed
2. KISS
3. Damage
4. Jams
5. Herding
6. Area of coverage


Elevation
Ramp
Conveyor
Pneumatic Jack
Bicept Curl
Corkscrew
Launching
Centrifuge
Catapult
Half-circle
Pneumatic Compression
Rotation
Pinball Launcher
Tennis Ball Launcher (Flywheel)
Potential Problems
1. Make height restriction/size
2. Torque
3. Cooperation
4. KISS
5. Safety
Potential Problems
1. Accuracy
2. Speed
3. KISS
4. Safety
5. Distance

Preferably, the new members even get to build small-scale prototypes of the ideas.

The separate groups then had group discussions of the modules and their vision for the robot. They discussed what features they wanted, whether they wanted to specialize or do really well at one task**, and what sensors will they use and potential autonomous ideas** (often forgotten)**.** Overall, the group should **design around their set of strategies (strategies for different scenarios) for the year's competition.**

This process is likely to change from year to year, but regardless, the brainstorming and design process **should always be documented and stored in the engineering notebook, with pictures such as those shown above, and drawings diagramming ideas, and overall robot design.**

Finally, "traffic color" diagrams are often useful, with designations as to the desirability of each potential capability of a robot. (See section 2.2 for an example).

An additional recommendation for the brainstorming process is a Weighted Objectives Table (WOT). Vex provides a very useful explanation found here: http://curriculum.vexrobotics.com/appendices/appendix-7. You may consider using a "traffic color diagram" in conjunction with this table, with green box fills being assigned to criteria closer to the max score, yellow closer to the middle of the scoring range, and red when closer to the lowest score for the scoring range, and various shades in between if desired. Regardless of the brainstorming method, the team should have concrete reasons for why the selected approach was chosen verses the alternatives, and those should be clearly shown in the Engineering Notebook. Diagrams and tables are often the most effective means of showing this process to the judges.

2.4 Awards

A strong Engineering Design process will be rewarded by judges at competitions. Awards that the Engineering Notebook and Design process play a part in are:

- The Excellence Award
- The Design Award
- The Judges Award (not stated, but likely)

In addition, **please note that for some awards at Worlds** teams need to sign up in advance. These awards include the **Excellence and Design awards among others! The signup page is traditionally located at:** https://www.robotevents.com/vexawards/index.php and the **deadline** is **well in advance of the tournament.**

However, please remember that we do not use the engineering design process to win awards, but rather because of its importance to the team and its usage in the engineering world at large. Awards at competitions are to commend us for our dedication to professional development and for following an important process used in professional engineering. Awards acknowledge our hard work and our dedication to learning from our process and mistakes.

A full list of awards (with descriptions) offered for each year's competition can typically be found in an appendix of the competition game manual. For example, in the 2015-2016 competition Nothing But Net, this page was found at: http://www.roboticseducation.org/documents/2015/05/vex-nothing-but-net-appendix-d-awards.pdf.

Finally, this page: http://www.roboticseducation.org/vex-robotics-competitionvrc/game-day-running-an-event/ provides rubrics and other documents that judges use when judging teams and robots at the competition. This guide: http://www.roboticseducation.org/documents/2014/11/local-judges-guide-vex-robotics-competition-2.pdf is a judges guide which may also prove useful.

2.5 Safety

Safety is an extremely important part of our team. We strive to be safe in all that we do when working on our robots to ensure that everyone has an enjoyable experience and nobody is hurt.

2.51 Power Tool Certification

Power tools, while very useful, can also be dangerous if used improperly. For this reason, any time a student wishes to use a power tool that they have not used before, they need to be "certified" by one of the team mentors, or by the team captain or a group leader (assuming they have experience with, and are certified to use, the tool in question). Certification includes instruction on proper use of the tool, common mistakes and how to avoid them, what safety equipment to wear when operating a tool (always includes eye protection!), and supervised practice uses of the tool. We most frequently use tools such as our Dremel (with cutting and sanding disks), power drill, band saw, and disc sander.

2.52 General Safety Information

General safety information from: https://www.vexrobotics.com/wiki/VEX_Safety

While the VEX development system itself is not inherently dangerous to use, anyone working with metal parts and tools needs to have a safe attitude and work in safe ways. The most important aspect of safety is to think safety at all times. It is well-known in workshops that safety starts with being aware of your work space and being sensitive to dangerous behaviors and materials. Purposefully use safety gear and follow safe practices to insure your own well-being.

Safety is not just about following rules, it's about doing work in a way that is safe. Most of the time if you think you are behaving unsafely, you are. Stop and think

before you act. **The most important safety tool of all is your head.** Remember: your attitude affects your behavior. If you have a positive attitude you will likely exhibit safe behavior. A negative attitude toward safety will cause carelessness, inattention, and injury.

Before working on your robot, go through these steps in your mind:

- Do I have necessary safety gear such as safety glasses and hearing protection?
- Is this the safest way to do this task?
- Am I using the correct tools? Using the wrong tool is frequently more dangerous than using the correct one.
- Are the parts secured correctly when cutting or bending? Use a vise or clamp – not your fingers.

2.53 Safety Rules

Safety rules from https://www.vexrobotics.com/wiki/VEX_Safety (slight modifications)

Know where the lab safety equipment is. We have a first aid kit, and a fire extinguisher in the lab, know where they are kept.

No horseplay! Workshops are no place for irresponsible behavior such as running around, "swordfights," throwing parts, or pushing people.

Always wear safety glasses when cutting or bending structural parts, especially when cutting with the Dremel and other power tools!

Wear safety glasses, even if you wear normal eyeglasses. Unless you have had them specially made (which would be pretty cool), eyeglasses do not offer sufficient protection from shrapnel or flying debris, and do not usually protect the eyes against debris coming from the side.

Always wear safety glasses around operating robots, and any time a battery is hooked up. Moving robot mechanisms can "sneak up on you" when you do not expect it.

Wear hearing protection when using power tools such as a Dremel or grinder. It does not take much noise to cause hearing loss. You may also consider wearing a face mask, the metal particles from a lot of cut metal can fly up and be inhaled quite easily.

Study all safety rules carefully and constantly apply them. When in doubt about any task, get help! DO NOT take chances!

Read, understand and follow the safety instructions that come with your tools. Complete safety rules for power tools would take hundreds of pages, so please refer to your tool's manual or the OSHA power tool safety guidelines at www.osha.gov.

Always dress properly for the shop. Avoid wearing loose fitting clothing, tie up or cover long hair, and remove rings and other jewelry when using power tools. **Wear closed-toed shoes in the lab!** Do not wear gloves when operating power tools.

Watch your hair, fingers, and tools around robots. Even the plastic gears and chains in a VEX robot can cause painful injuries if you are not careful. Long hair is especially dangerous around operating robots so keep it under control.

Be aware of what others are doing. Safety is everyone's responsibility – both individually and collectively.

Report unsafe conditions or practices.

Keep the lab clean. Metal scraps should be disposed of in a waste bin or recycling container. Never allow scraps to remain on the bench or floor. Removing trash and debris reduces slipping and falling and generally promotes a safe work environment.

Learn how to use your tools properly. It is foolish to operate tools without first receiving proper instructions. This applies to even the small power and hand tools used in VEX robotics. Get help if you are not sure what must be done or how a task should be performed, see section 2.51 for Power Tool Certification.

Keep the VEX parts out of your mouth, ears, nose, etc. It's a robot, not a snack.

Do not short out the batteries by running a wire between the battery terminals. This can cause fire, ruined batteries, and/or explosion.

Use batteries wisely. Do not lick batteries, or put terminals in your mouth. If a battery appears to be leaking or has a crystalline deposit on the outside, dispose of it immediately and wear gloves – preferably made of nitrile or other non-reactive material – when handling.

Dispose of worn-out batteries responsibly and safely. Do not throw batteries into the trash, especially rechargeable batteries. Contact your local waste disposal office for information on battery disposal. Batteries should be stored as directed by your local hazardous materials disposal office until pickup, usually in a hard-sided, waterproof, non-conductive container, such as a plastic bucket.

Antennas are not a toy. Do not use them as whips or swords, and do not put them into your mouth, eyes, ears, or nose. If the blunt tip breaks off the tip of your antenna, dispose of the antenna and replace it with a new one.

Don't drive robots on tables. A robot that drives off of a table is, needless to say, very dangerous.

Be careful with autonomous. A robot in autonomous may do unexpected things, keep hands clear and stay a sufficient distance away.

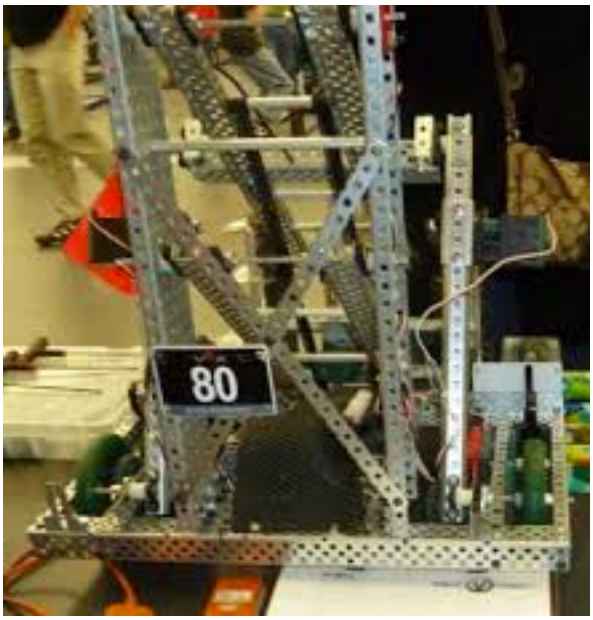
3.0 Robot Mechanical Engineering

3.1 General Advice

Almost everyone will initially work on the robot. Programmers will have some pre-robot programming they can do but that work may be limited. If programmers don't have something to do, they should be helping with construction too! While people may have jobs and roles, they can go outside of those and get experience in other areas as well. Everyone can contribute with ideas and building. On to the general advice:

Structural/Design

- If prototyping a mechanism which will not go on the robot, it is useful to use KEPS nuts (the nuts with teeth) to secure screws. However, on final robot mechanisms it is strongly recommended that Nylock nuts are used, as they have an internal nylon coating which prevents them from unscrewing, even despite strong vibrations (nylock nuts need to be held, often by a wrench, while a screw is being screwed in).
- Build robots half an inch or an inch smaller than the size limit! It's very easy for a design to take up a little more space than planned.
- The robot's overall structure should be very accessible. Motors should be fairly easy to get to for replacements (after stripping a motor, pinching a wire, etc.) or for re-gearing.
- Modular design is ideal. Being able to add extra features or switch out important components easily is valuable, and modular design lends itself to more flexibility overall.
- VEX refers to parts in units of half inches. For example, a 35 length aluminum c-channel is 17.5 inches in length. If the c-channel has a width of the 2 units, it is a 1x2x1x35 aluminum c-channel. Learn these classifications, it will make talking about designs and journaling much easier.
- To keep structure robust, use c-channels, and angle bars instead of bars or chassis rails. In terms of strength, bars < chassis rails < angles < 1x2x1x__ c-channels < 1x3x1x__ c-channels < 1x5x1x___ c-channels.
- Also use cross bracing (often diagonal), and connect parts smartly to keep a structure robust. An example of diagonal cross bracing with bars is shown below.



(image from https://www.vexmen.com/wp-content/uploads/2013/05/Vexmen-Spring-Educational-Sessions-week2.pdf)

- Sometimes you need to use standoffs instead of metal, to conserve space. They are not as strong as metal, but can be very useful for secondary support.
- You can also sometimes simply use a long screw with spacers as another method of attachment.
- Triangles are very strong structures.
- Vex parts don't always align perfectly in the holes. Sometimes you need to loosen a screw, move the screw and nut around in the hole, then retighten, to make structures fit.
- More metal is often stronger, but also heavier.
- Keep your robot's center of gravity in mind, especially in games involving lifting. You can move heavier components (like batteries, the cortex, steel parts) so that even after lifting your robot's center of gravity is still within its support polygon.
- A lighter robot means that motors have to do less work to move the robot. It is often useful to use aluminum parts, especially for arms and lifting mechanisms, tall towers, and when using a lot of c-channels (which are fairly heavy), as aluminum is about 40% less heavy than steel. Aluminum is quite competitive with the strength of steel (its light weight means that aluminum parts are thicker than steel while still being lighter than steel), though aluminum is softer and can be scratched up more easily.
- Generally, a simple design is the best option. Complex mechanisms can work quite well for certain applications, but look for a simpler solution first. Reliability is king.
- Log everything in the Engineering Notebook, and take lots of pictures!

Motors, Shafts, and Gearing

- You should almost always use a bearing wherever a shaft passes through metal. The bearing flats reduce friction and let you get more out of your motors. Even if there's some spacing difficulty, it is almost always worth putting a bearing flat.
- Consider pillow Bearing Blocks if you need to mount a shaft above or below metal (C-channels on a drive base for example). This application could work very well if you need to get over a ramp as a part of the competition for example.
- Motors should be fairly easy to get to for replacements (after stripping a motor, pinching a wire, etc.) or for re-gearing.
- Consider High Strength Gears for high-load gearing applications.
- Chain and sprockets are great for long distance transmission of loads or moving motors to different locations to conserve space. They can also be used on a drive base to move motors away from the wheels of your drive base and make space for intakes at the front of the robot for example.
- There are very limited safe applications for the regular chain and sprockets, which are weak and break very easily. You almost always want to use high strength chain and sprockets.
- Use spacers and collars on shafts with gears to keep them in place and prevent them from moving accidentally during competition (shafts should almost always be collared on the motor side, up against the metal the motor is mounted to).
- When using shaft lock bars, the metal shaft lock bars are far less prone to breakage than the plastic ones, however it can sometimes be difficult to get the shaft through. You may want to use grease or WD-40, or use those with a hammer.

Part Modifications (cuts, sanding, etc.)

- In general, aim to cut parts in 5's, you'll usually find that they're easier to reuse later.
- The bandsaw works very well for cutting parts, but keep your hands well clear of the blade, and be careful on the "follow-through" when you finish cutting a part.
- The Dremel is very useful for small cuts, making cuts and sanding parts already mounted on the robot (that would be difficult to remove).
- After cutting and sanding a part with the sanding wheel or Dremel, proceed to use higher and higher grit sanding sponges to ensure the part is not sharp (sharp parts on the robot are dangerous to our team, and will also not pass inspection).

And of course, it is also important to understand parts of the Software Engineering and Electrical Engineering jobs as well. Knowing how to wire and which pins go where is always useful on the robot, and being able to change that constant within the code when a programmer isn't available can save a significant amount of time.

3.2 Drive Trains

We highly recommend taking a look through unit 9 of the VEX EDR Curriculum (http://curriculum.vexrobotics.com/curriculum/drivetrain-design) This unit provides an excellent and clear introduction to drivetrain design and concepts (another useful resource, more geared towards FRC teams but with similar principles can be found here: http://www.simbotics.org/files/pdf/drivetraindesign.pdf). Finally, another FRC guide with more math and equations, and on more general drive train basics (which also apply to VEX) is available here: https://www.chiefdelphi.com/media/papers/1443 .

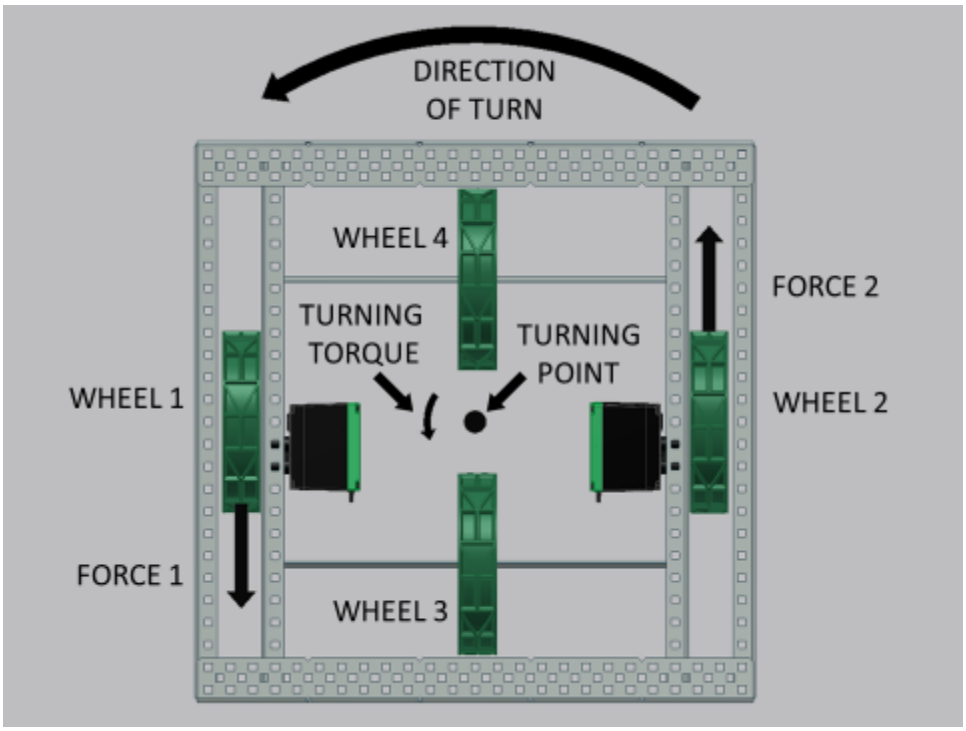
We have copied one particularly important section on turning torque and turning scrub from http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology-and-turning, seen below:

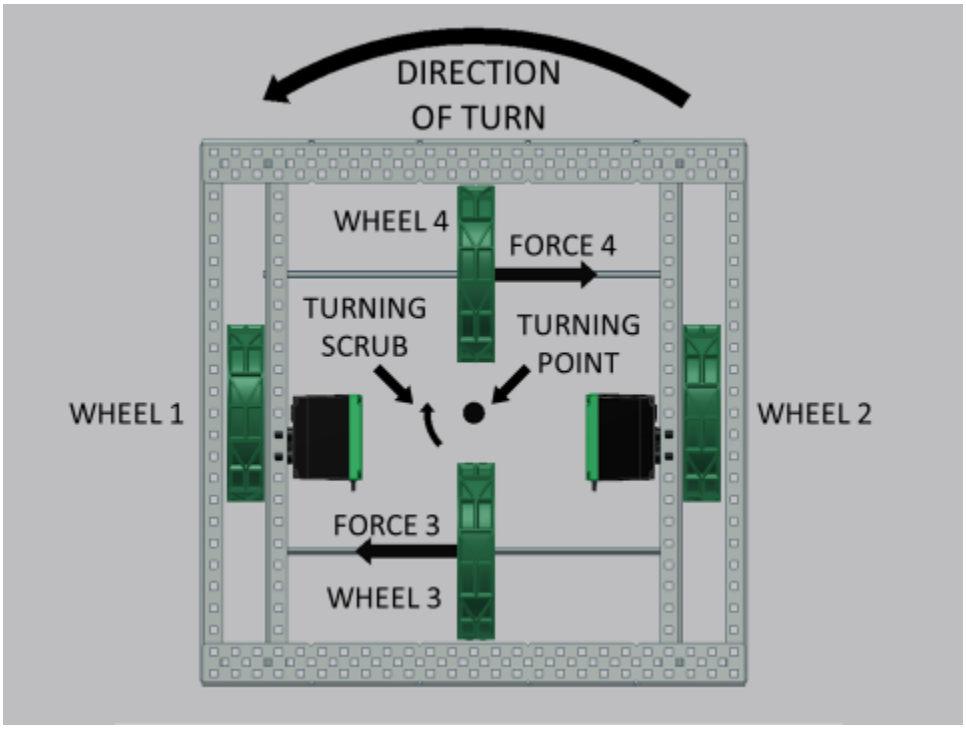
“

One of the major attributes that defines a drivetrain’s performance is how well it turns. There are two main properties which affect drivetrain turning: Turning Torque and Turning Scrub.

Turning Torque is the torque about the turning point which causes the robot to turn.
Turning Scrub is the friction which resists the robot turning. This is caused by wheels dragging sideways on the ground as the robot turns, resisting the motion of the turn. Turning Scrub is also expressed as a torque about the turning point of the robot, opposing the turning torque.

In a typical skid-steer drivetrain (specifically one in which all the wheels are drive wheels), ALL the wheels will exert force that contributes to the turning torque, and they will ALL drag sideways and contribute to the turning scrub. To help visualize things more simply, one can think of a robot in the odd configuration seen below:

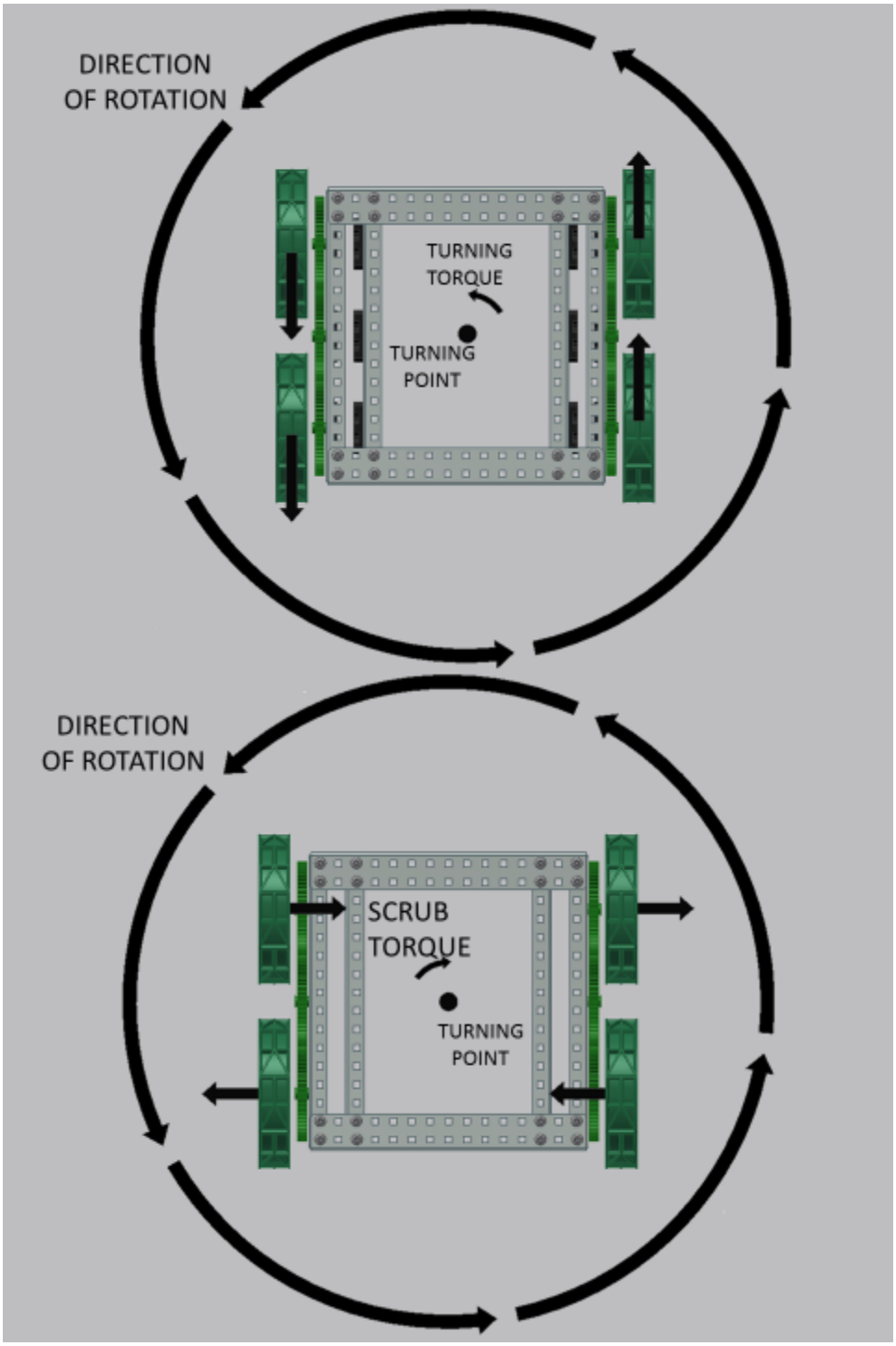




One can see that Wheel 1 and Wheel 2 contribute to the turning torque. They each exert a linear force (Force 1 & Force 2) that creates a torque about the turning point. These wheels do not drag at all as the robot turns, so they don't contribute to the turning scrub.

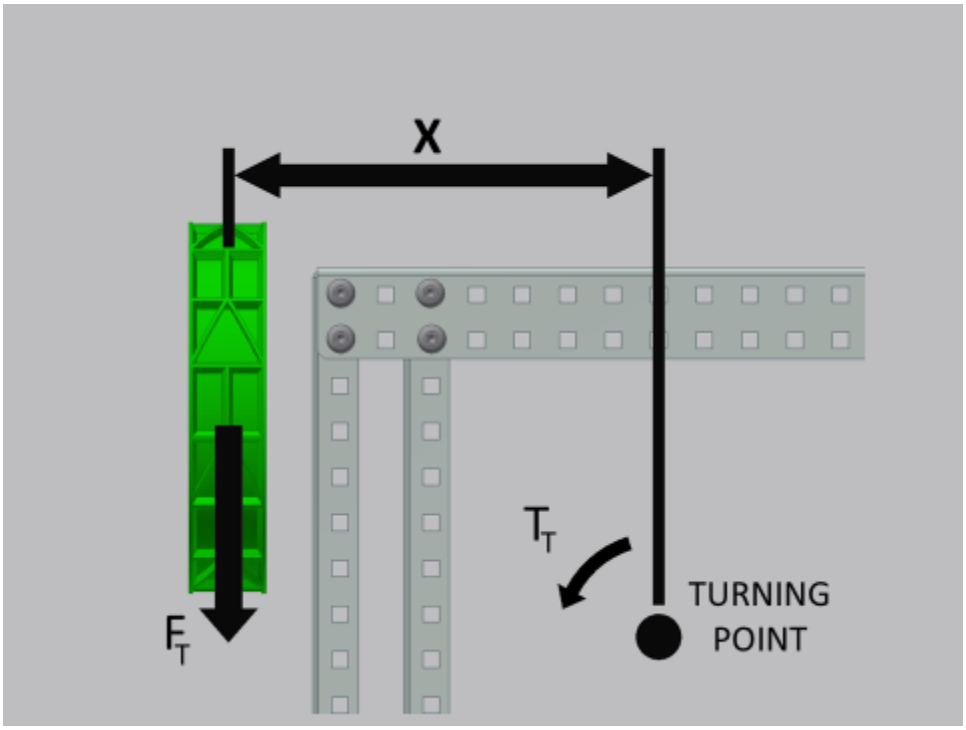
Wheel 3 and wheel 4 do not contribute to the turning torque, but they slide sideways as the robot turns and significantly contribute to the turning scrub. Force 3 and Force 4 are frictional forces from the wheels on the ground; this friction results in the turning scrub.

Now, in a more traditional drivetrain configuration, all the wheels would both contribute to the turning torque, AND the turning scrub:

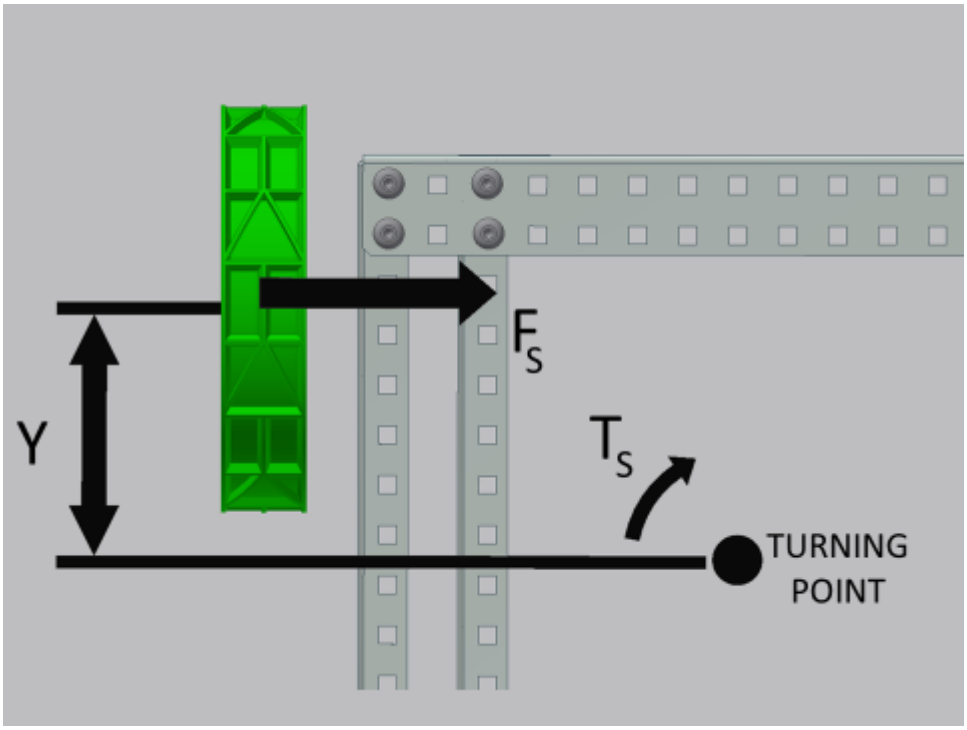


In the above case, all four wheels contribute to the Turning Torque, and all four wheels contribute to the Turning Scrub. Each wheel applies some force that contributes to turning, and each wheel needs to slide sideways and contributes some friction to scrub.

Turning Torque and Turning Scrub are both torques about the robot's turning point. As discussed in Unit 7: a torque is a turning force, defined by a linear force at a distance from some center of rotation. The below diagrams show how the frictional force of the wheel rolling forward contribute turning torque of the robot, and the frictional force of the wheel sliding sideways contributes to the turning scrub of the robot.



As seen above, the turning torque is contributed to by the friction force of the wheel at a distance from the turning point.



As seen above, the turning scrub is contributed to by the force of the wheel, around the robot turning point.

If a drivetrain has multiple wheels on the ground, all of these wheels will contribute based on their location in the drivetrain relative to the turning point.

DESIGNING A TURNING DRIVETRAIN:

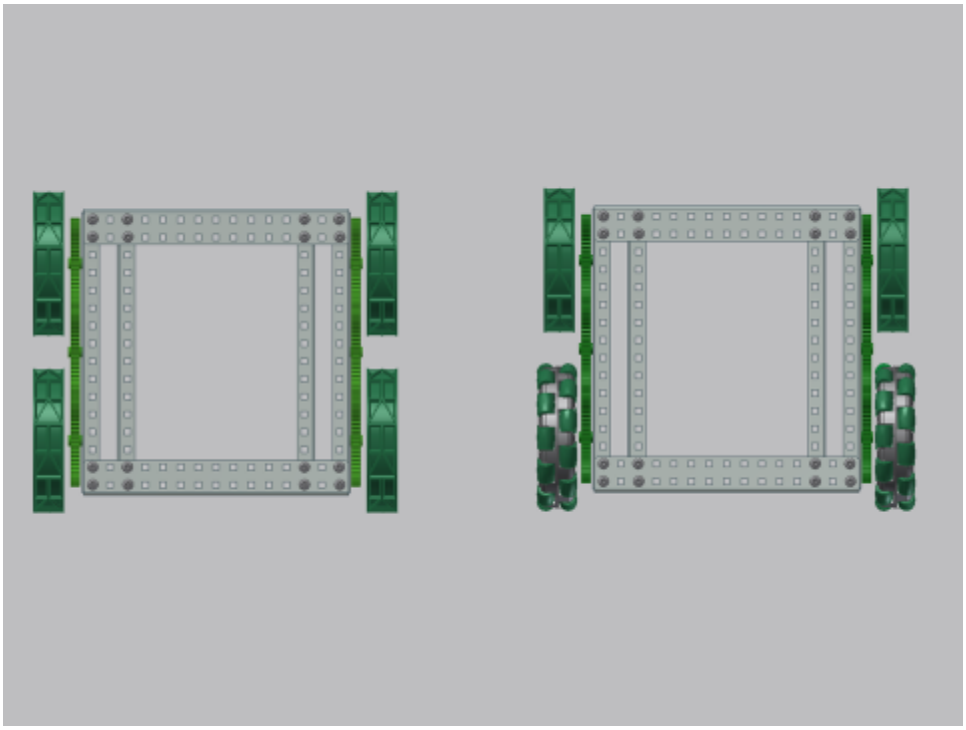
Once one understands the concepts of turning torque, turning scrub, and how they affect robot turning, one can begin to understand how to alter them to make a robot turn more effectively.

How does one reduce turning scrub?

Turning scrub is driven by the force of friction of the wheel sliding sideways on the floor. By reducing this frictional force, one reduces the turning scrub. One may also decrease the distance the wheel is from the turning point.

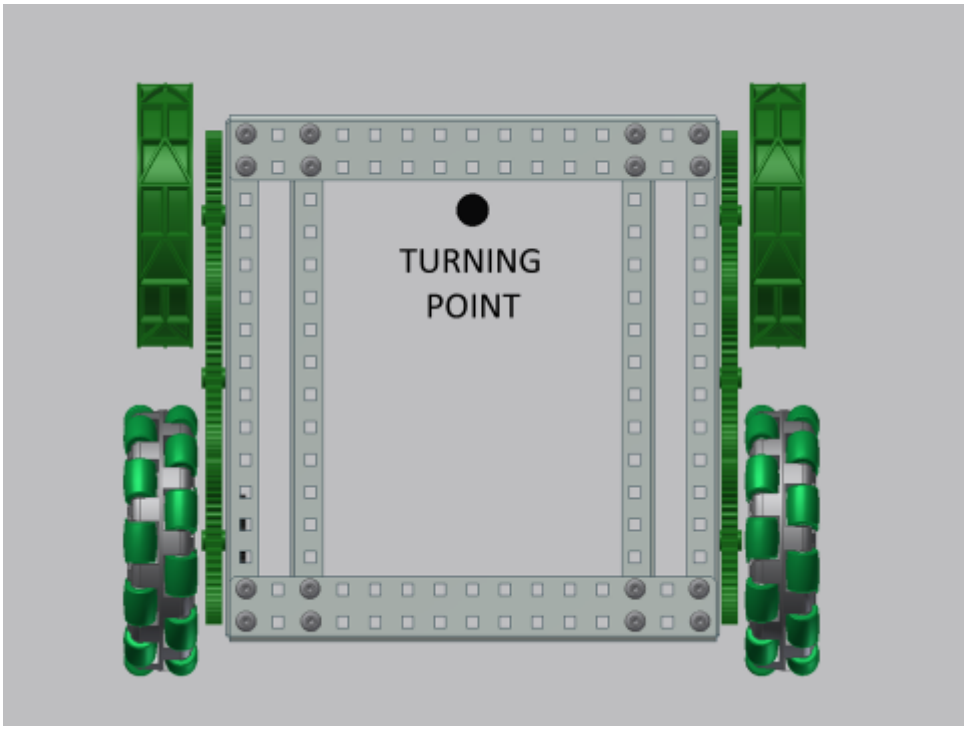
Similarly, one could increase turning torque with the opposite approach; by increasing the frictional force, or increasing the distance from the turning point.

Notice that to decrease turning scrub, one must decrease the wheel friction in the left/right direction. To increase turning torque, one must increase the friction of the wheel in the front/back direction. It is difficult to modify the friction of a wheel in one direction without affecting the other, so it is usually best to modify the geometry of the robot chassis to help improve robot turning.



However, designers should note that omni-directional wheels have ZERO sideways friction. This means a drivetrain with an omni-wheel would have NO turning scrub caused by that wheel. A drivetrain with ALL omni-wheels would have almost ZERO turning scrub!

It is interesting to note that a drivetrain with two omni-wheels and two traction wheels would have a turning point directly between the two traction wheels. This drivetrain would also have no turning scrub, since the traction wheels would not need to slide sideways.



”

Now, found below are the most common types of drivetrains.

Tank Drive- 2 Wheels

Info from page 80 of: http://www.simbotics.org/files/pdf/drivetraindesign.pdf

This drivetrain is rarely seen. It uses 2 driven wheels, and either self-balances, or uses other low traction, non-driven, wheels. Turning with this drivetrain offers almost no scrub, or resistance, which can be advantageous, but also disadvantageous, as the drivetrain will be difficult to control, and easily blocked. The drive train will also not have much pushing force, and may not accomplish a “centered” turn.

Tank Drive- 4 Wheels (Skid Steer)

Info from: http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology
http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology-and-turning

This drivetrain is one of the most common ones out there, and also known as a skid steer. The wheels on the left and right side are powered by separate motors, and are locked pointing forward or backward (cannot turn/steer).

“Steering is accomplished by varying the speed of the different sides (i.e. if the right side goes forward very fast, and the left side goes forward slowly – the robot turns left).

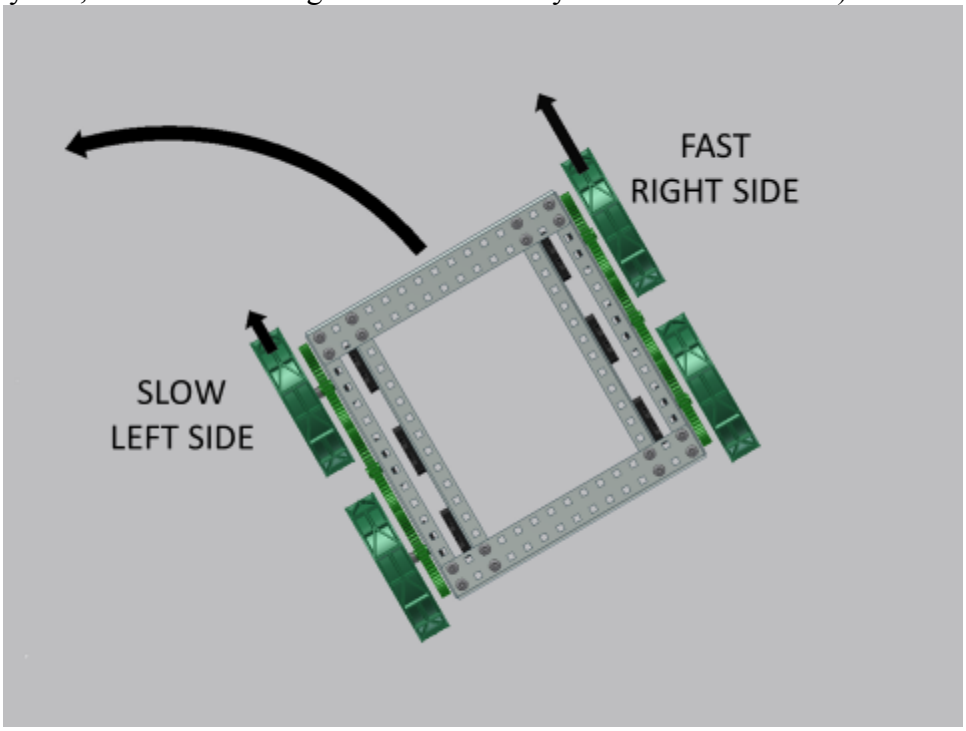
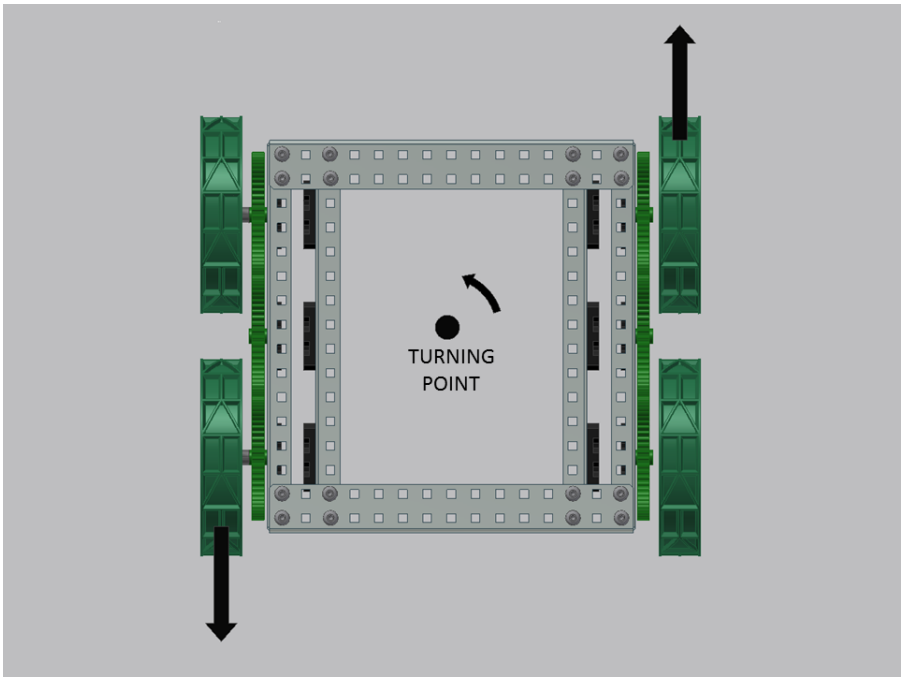
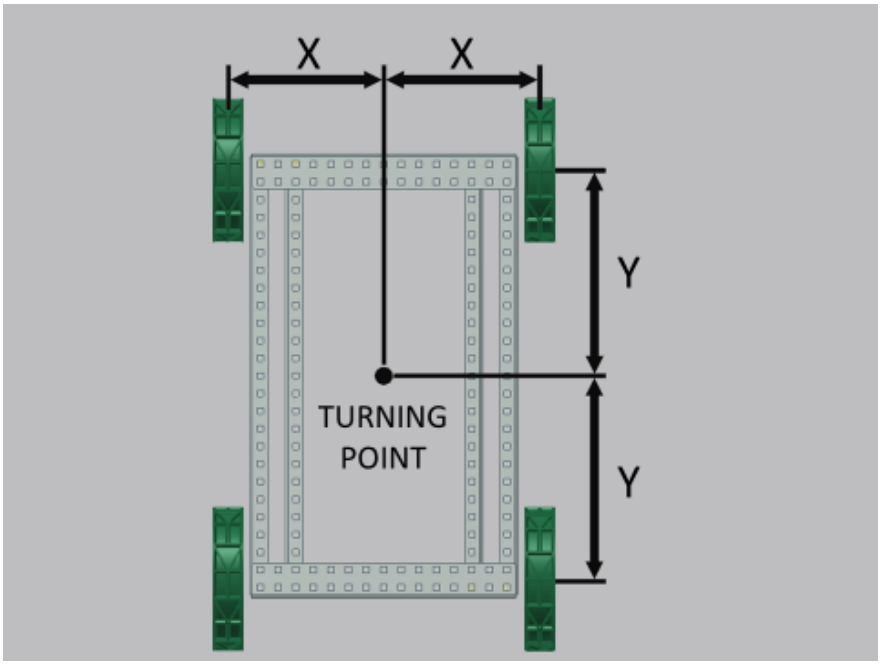


Image credit to curriculum.vexrobotics.com

This type of drivetrain is capable of zero radius turns; the robot driver would simply power one side forward and the other side in reverse.” (http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology)

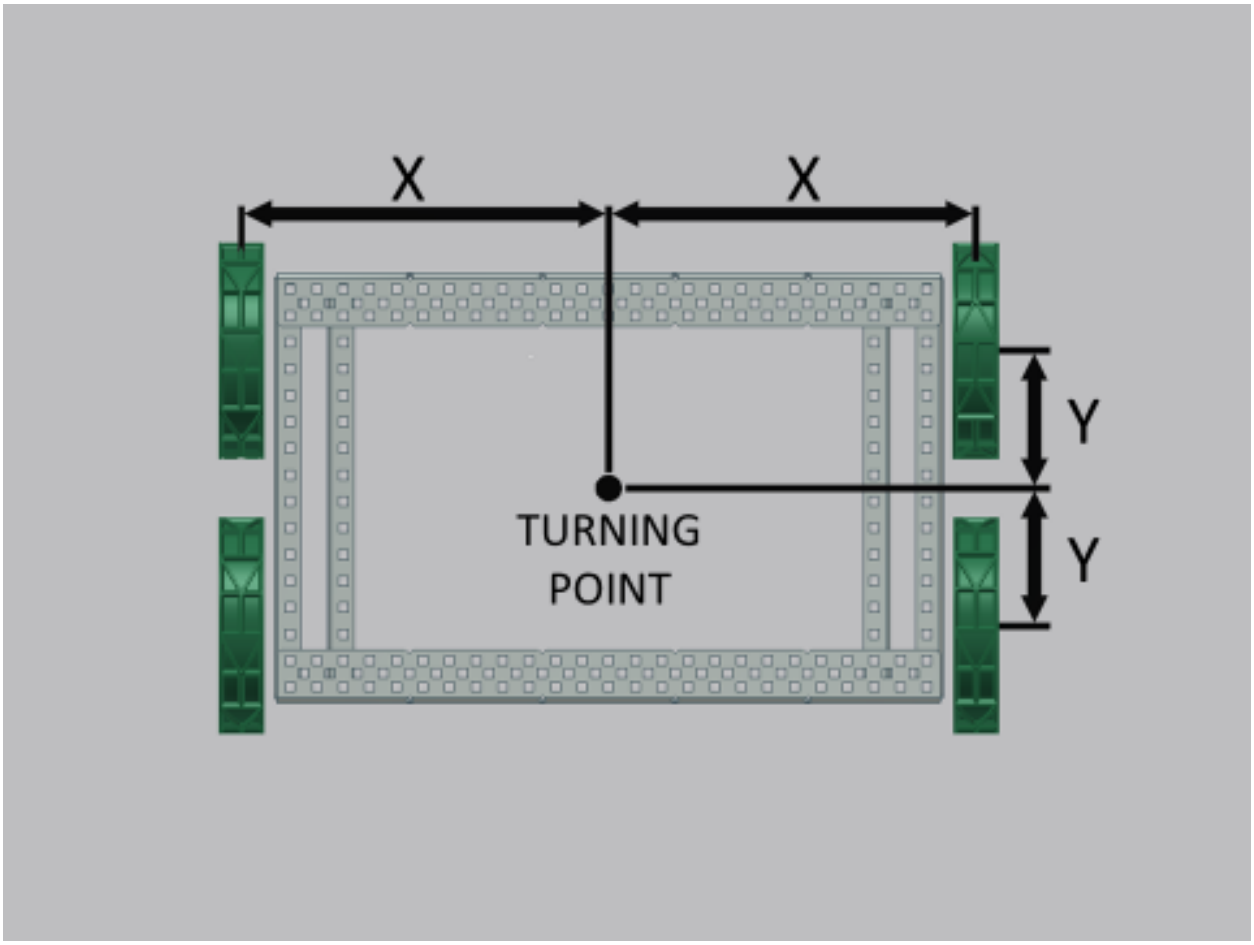


All power from the motors on the drivetrain are used for acceleration or pushing. However, if this configuration is used in a particular long and narrow drivetrain, note that it will have poor turning characteristics as a result of high turning scrub, and low turning torque.



Credit to: http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology-and-turning

On the other hand, a short and wide drivetrain will have favorable turning characteristics.



Credit to: http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology-and-turning

Tank Drive-6 wheels
Info from: http://www.simbotics.org/files/pdf/drivetraindesign.pdf
http://www.vexforum.com/index.php/6715-6-wheel-base/0
http://www.vexforum.com/index.php/5853-vex-6-wheel-drive/0

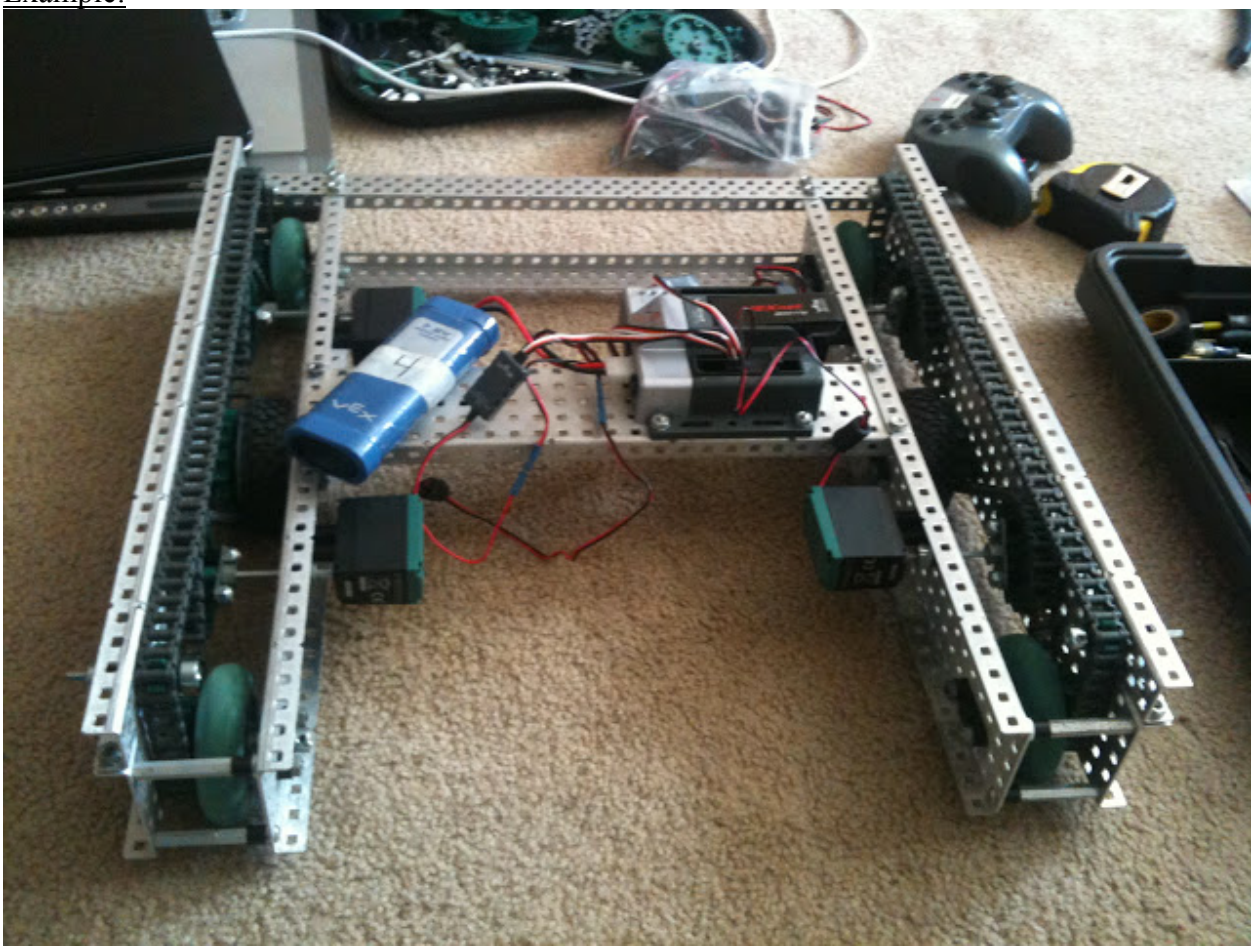
In VEX a 6-wheel drivetrain with all high traction wheels tend to have too much turning scrub to cleanly turn (depending, of course, on the robot). You could have the outer wheels be omni wheels to reduce the scrub, or use an all omni 6-wheel drive train (which has worked well for us before).

Another option which we have not yet tried is the popular “drop-center” 6-wheel drive train, more common in FRC. This drive train involves lowering the center wheels a bit lower than the outer wheels. Lowering this center wheel reduces the length of the robot (makes turning easier, this is more of an “effective length”). Some teams use the 2.85” VEXplorer wheels in the center with 2.75 traction wheels on the outside.

http://www.vexrobotics.com/edr-wheels.html
(The VEXplorer wheel has been discontinued, but is still legal for competition).
A 6-wheel drivetrain can be done with the dropped center wheel in order to provide the very useful advantage of clean turning **with high traction wheels (only if there is a centered center of gravity on the robot, and there is enough of a drop of the center wheel),** because of its reduction in the effective length of the drivetrain for turns. Clean turning with high traction wheels provides both clean turns, and strong pushing force against other robots.

If the team ever extensively tests the drop-center drivetrain, feel free to write a bit about it, the wheels you used, the measurements, the difficulties, its performance, and anything you feel is relevant!

Example:

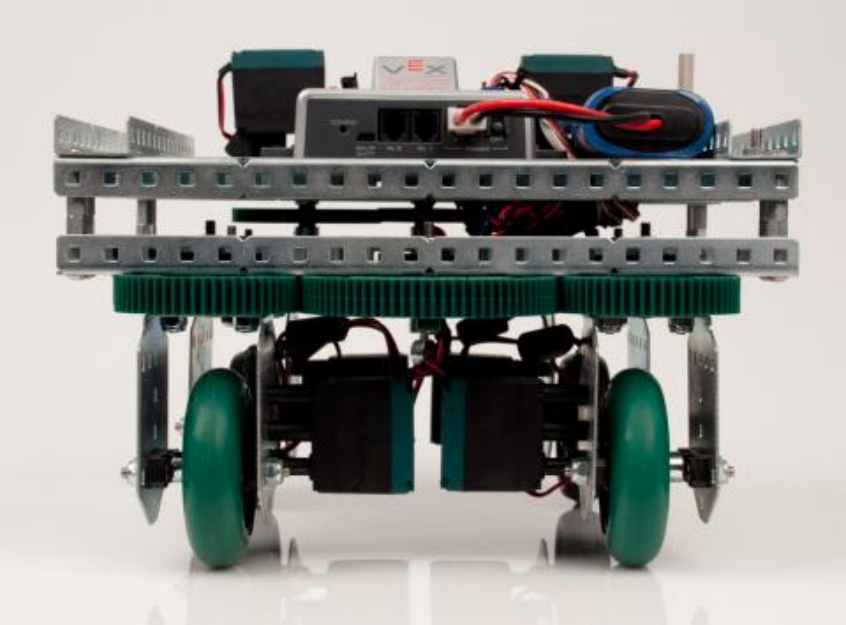


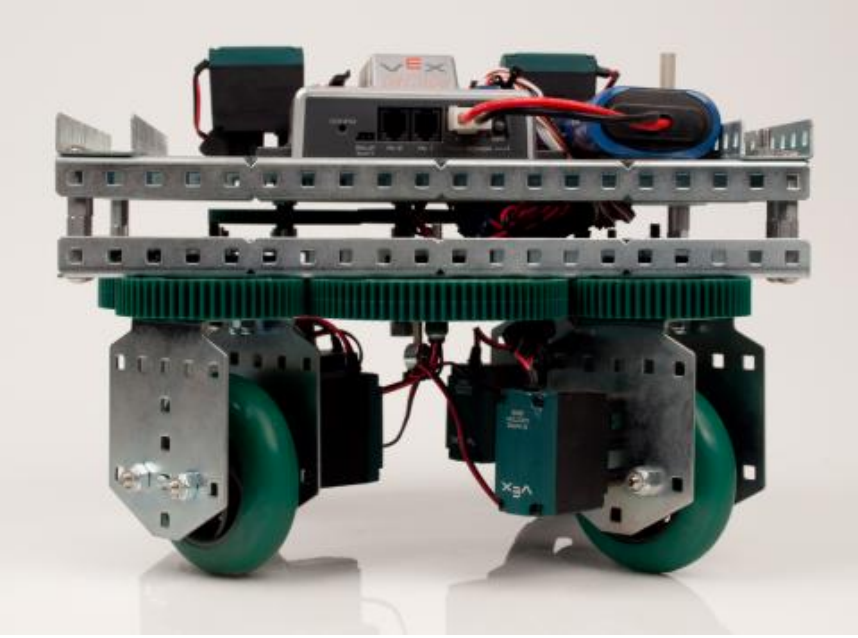
Credit Vexforum user: Ryan Dognaux

Swerve Drive

Swerve drives use wheels which are powered not only forwards and backwards, but also are powered so they can move in any direction by steering the wheels. These drivetrains are rarely seen in competition, primarily because of the extra complexity they add to the drivetrain, and the extra space they take up.

Examples:





Credit to: http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology

Crab Drive

A crab drive uses two sets of tank drive drivetrains, each pointed in different directions. Only one of these drivetrains touches the ground at once.

“

> For instance, a crab drive would have a primary skid-steer drive pointed forwards/backwards, and could drive around normally on this. Then it could drop a secondary drivetrain pointed right/left, lift its primary drive off the ground, and be able to move right or left (like a crab).

”

(http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology)

Example:

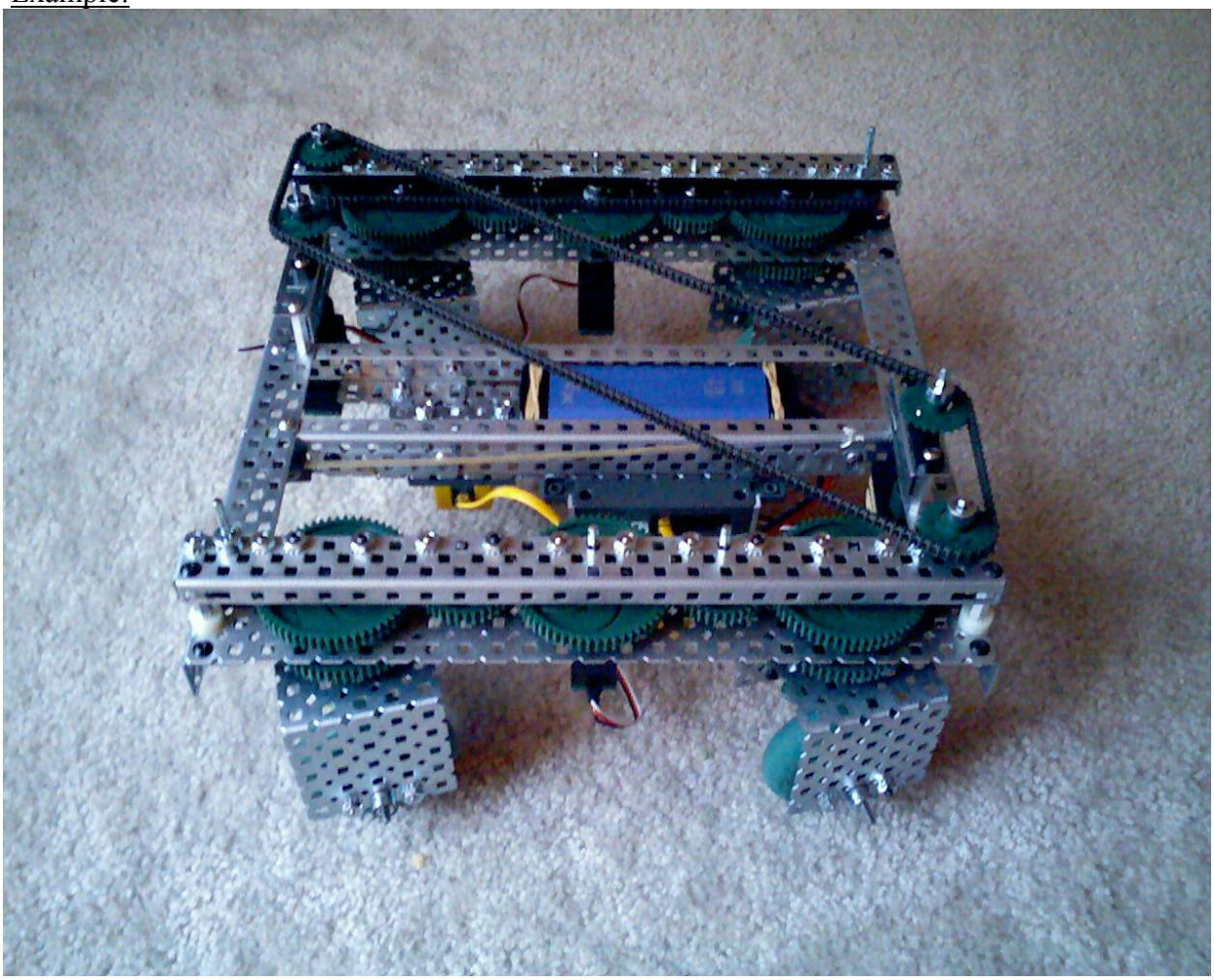


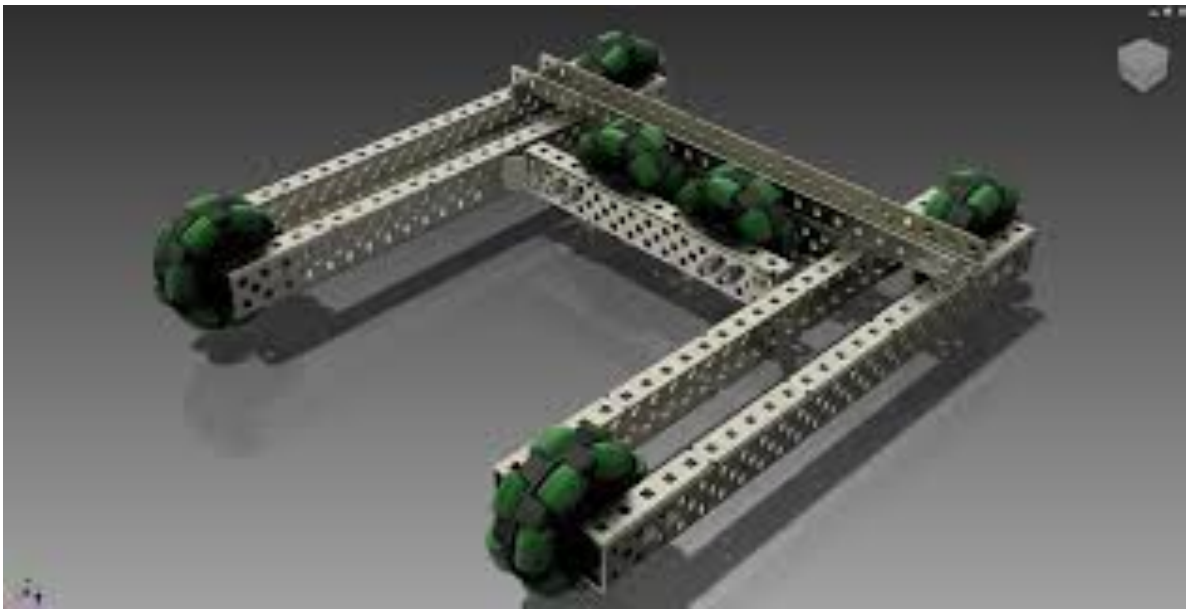
Image credit: https://www.chiefdelphi.com/media/img/65b/65bd33ea004417520738b7a81e49d2f7_l.jpg

Of course, the drivetrains above can be modified. Changing the types of wheels used affects a number of characteristics, for example using omni-wheels on a typical 4-wheel tank drive decreases the turning scrub significantly! The next few drivetrains require omnis or mechanums in certain orientations.

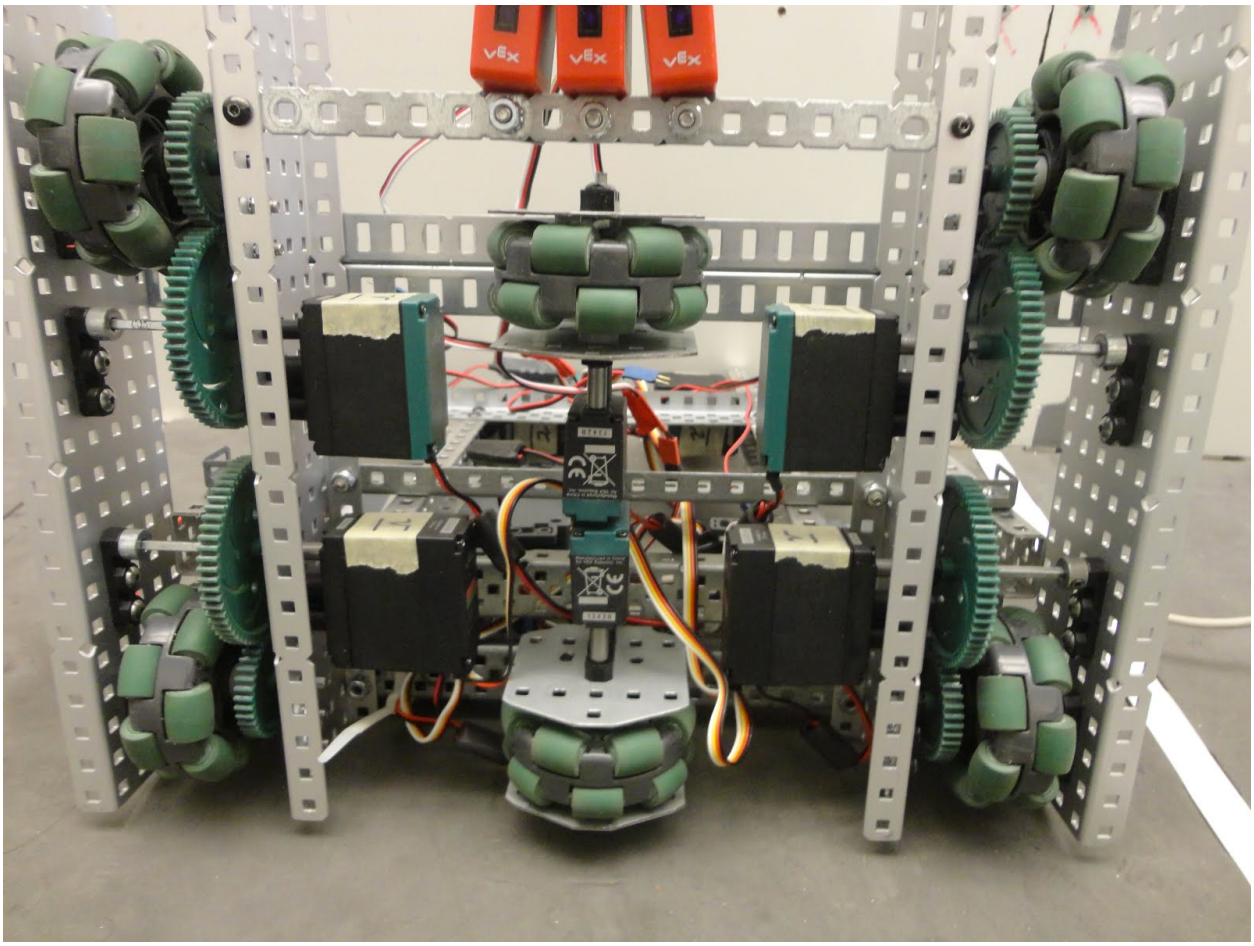
H-Drive or Slide Drive

This drive train is similar to tank drives, except it adds an extra wheel (or more than one) near the center of the robot perpendicular to the other wheels. This type of drivetrain requires all omni-wheels, and provides the advantage of being able to strafe (move left and right without turning). This drivetrain augments a robot's agility while strongly decreasing its pushing potential. This drivetrain also requires an extra motor (5 on a 4 motor tank drive) to strafe, while other drives (Holonomic Drives and Mecanum Drives) do not require that motor.

Examples:



Credit: Team 323Z (http://www.team323z.com/uploads/1/0/0/6/10062475/7494614_orig.png)

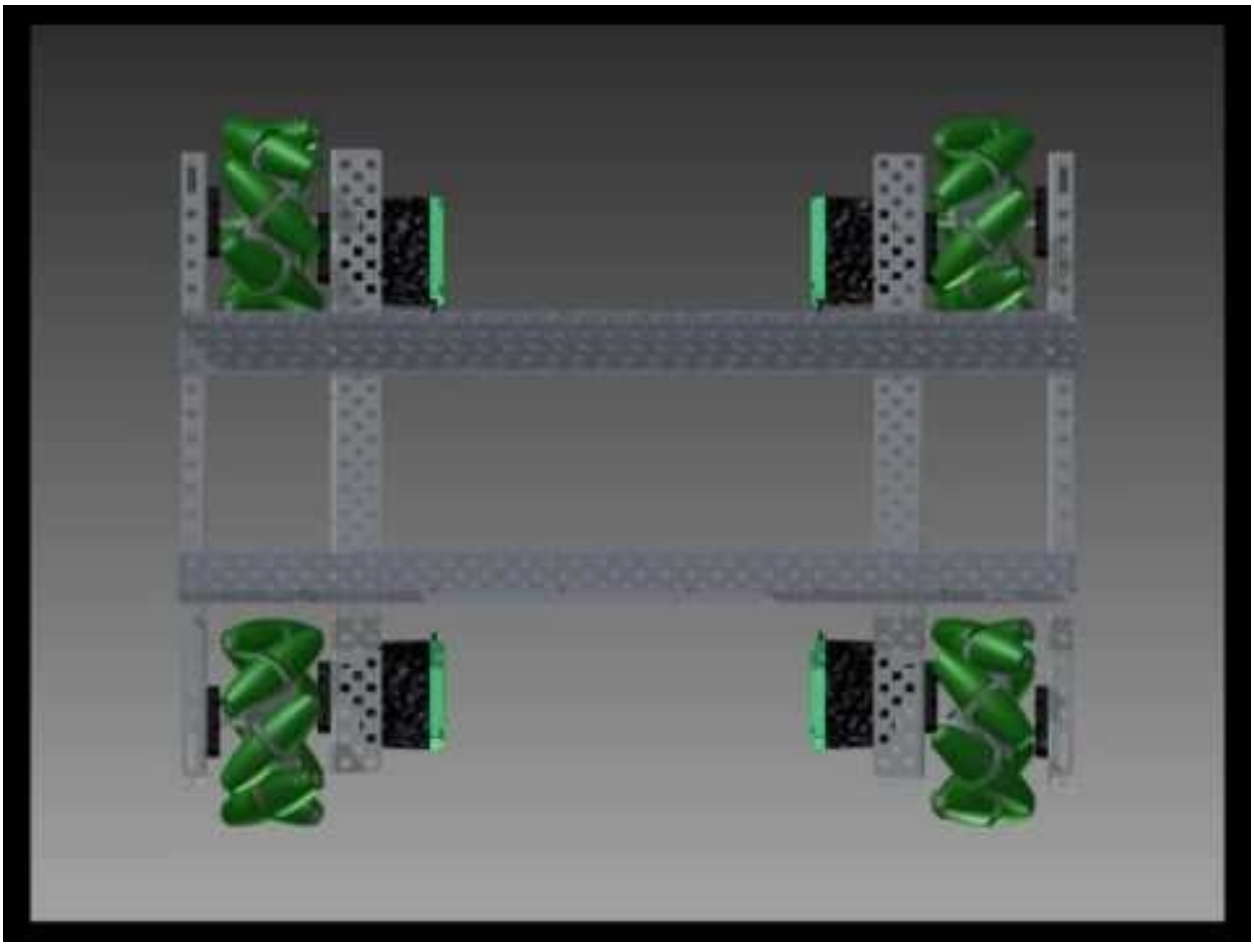


Credit: http://2.bp.blogspot.com/-Ndm22tI80tw/UNPgOberPHI/AAAAAAAAADk/7VsPtG2-xHk/s1600/1.JPG

Mecanum Drive

Mecanum drives are similar to a 4-wheel tank drive, except they must use 4 mecanum wheels. These drivetrains are fairly easy to design, and allow for strafing capability without sacrificing an extra motor. However, in VEX, the mecanum wheels themselves take up quite a bit of space. Additionally, mecanums do not have the potential to provide strong pushing force, and can provide some difficulty when on a robot that does not have a centered center of gravity.

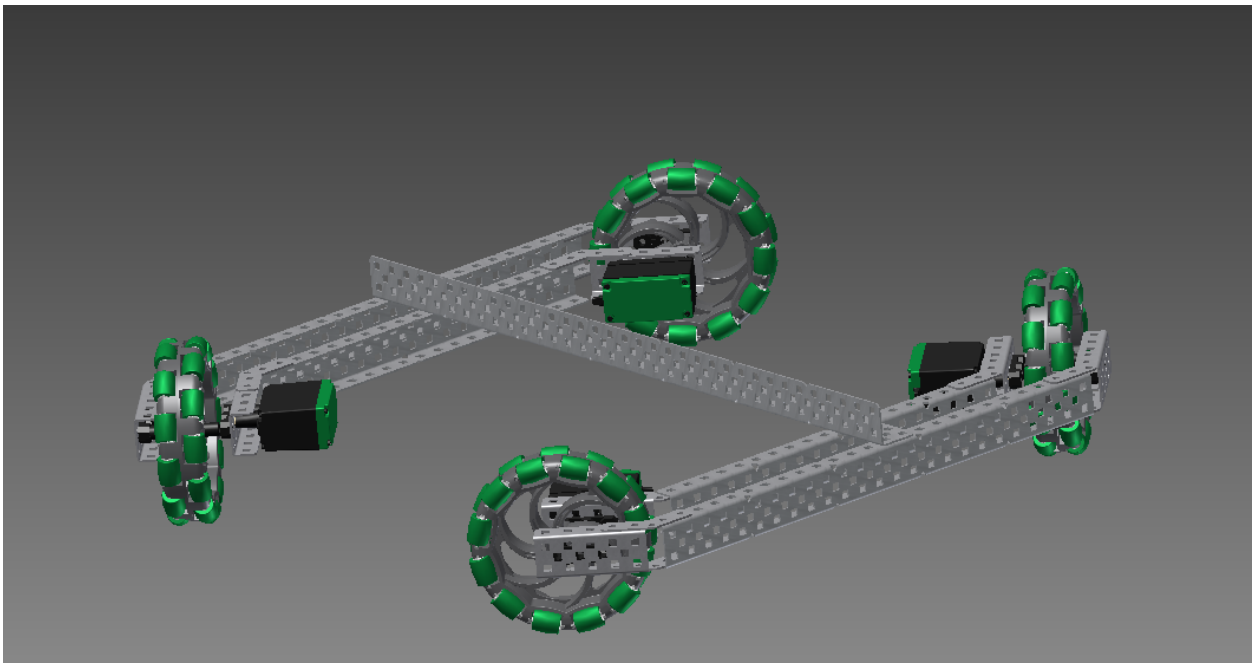
When using mecanum wheels, it is very important to orient the wheels correctly. The image below shows proper mecanum orientation from a top-view. You'll note that each mecanum wheel has the rollers oriented so they form an "x" if you follow the roller orientation and extend it to the middle of the drive base. **If mecanum wheels are not installed in this orientation, they will not work properly!**



Credit: https://www.youtube.com/watch?v=5Ge_9rwQTqA

Holonomic Drive- X-Drive

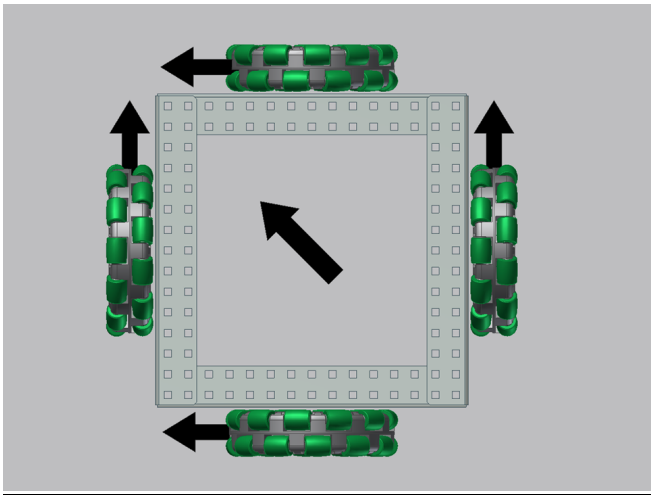
This drivetrain uses 4 omni wheels positioned at a 45 degree angle in the corner of the drive base. Each wheel needs to be independently powered. The big advantage of this drivetrain is its agility. The drivetrain can move in any direction, move diagonally, and can strafe. X-drives actually have a higher maximum speed than tank drives with the same gear ratio. See http://www.aura.org.nz/archives/1137 for an in-depth explanation. However, the X-Drive will likely have more losses due to friction than a tank drive will because of the friction between the omniwheels and the omniwheel rollers, and will push with less force than a tank drive (http://www.aura.org.nz/archives/1137).



Credit: JVS on Vexforum.com

Holonomic Drive- Plus Drive

Similar to an X-drive, but with the wheels oriented perpendicular at the ends of a + shape. The plus drive has similar advantages and disadvantages, and is able to move in any direction, but its formation can make it difficult to leave a large enough space to intake scoring objects.



Credit: http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology

Summary of links used/recommended:
http://curriculum.vexrobotics.com/curriculum/drivetrain-design/friction-and-traction
http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology
http://curriculum.vexrobotics.com/curriculum/drivetrain-design/drivetrain-terminology-and-turning
http://www.simbotics.org/files/pdf/drivetraindesign.pdf
http://www.aura.org.nz/archives/1137
http://www.vexforum.com/index.php/20640-x-drive-blueprints

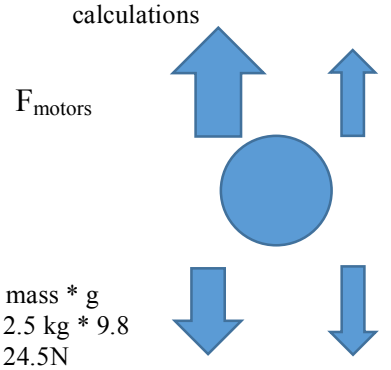
And always test your drivetrain extensively! Put some weight on it before you've made the other mechanisms, have the drivers drive like crazy and push against other robots to make sure the drivetrain does not stall out.

3.3 Past Lifts, and Lessons Learned

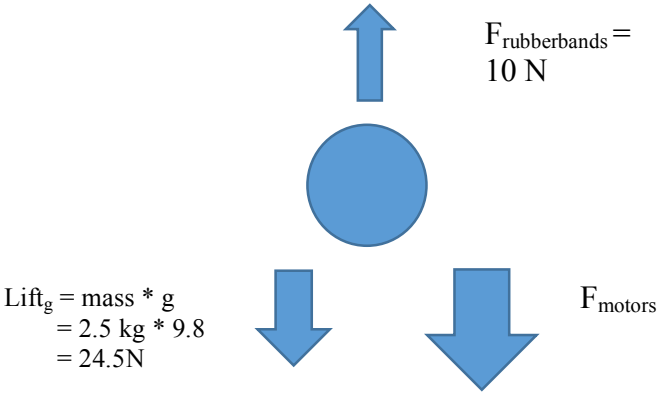
It is important in many games to lift the game objects to various heights. We've compiled a few useful tips for creating lifting mechanisms.

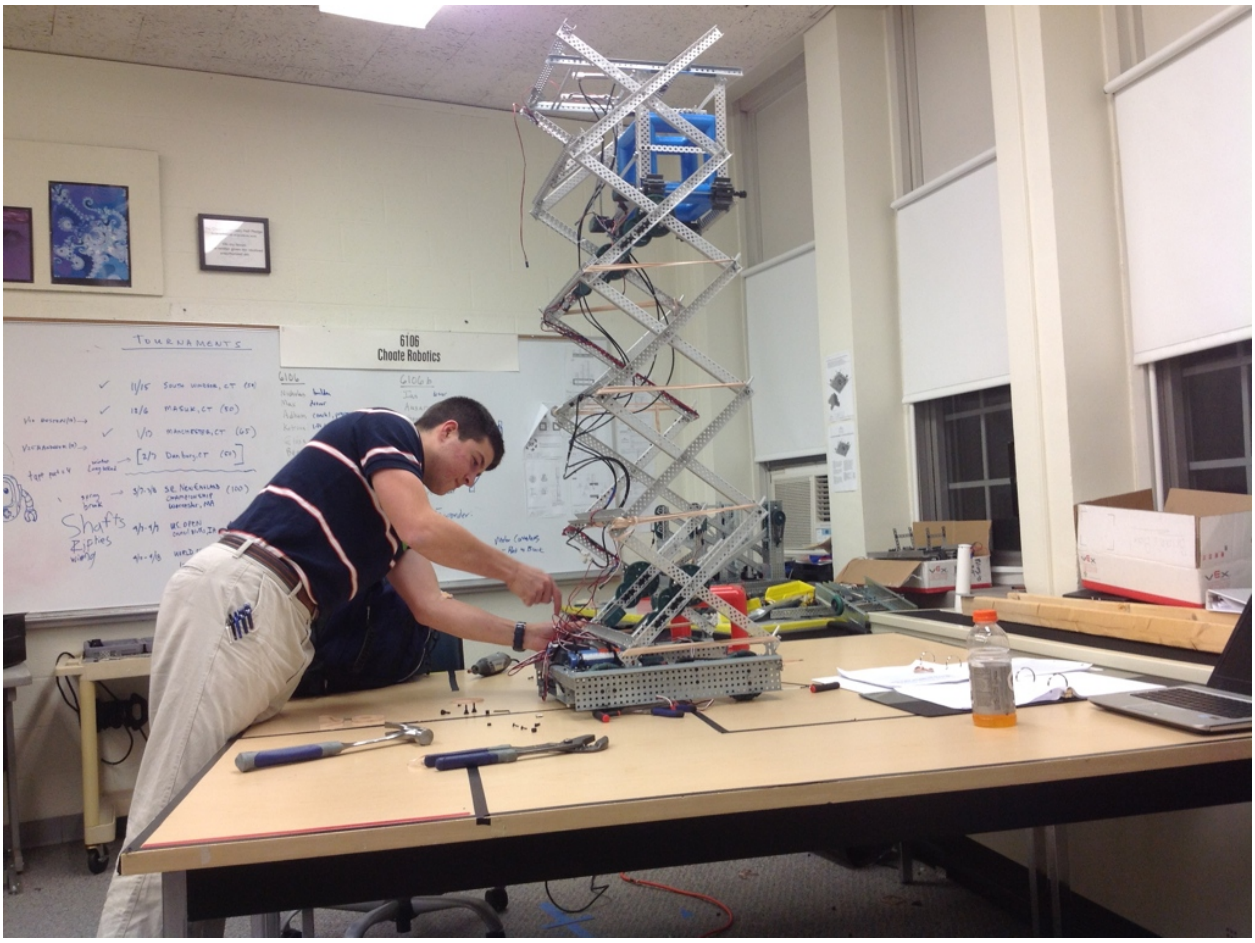
One general tip however, it is often extremely useful to use counterbalancing with lifts. Counterbalancing refers to using rubber bands or latex tubing to assist your lift in extending vertically. If you are trying to raise a scissor lift holding 5 lbs. of weight for example, you may want to use rubber bands across the scissor stages. The rubber bands will aid in compressing the scissor lift, so that extension with that load weight **and** the weight of the lift itself is easier. However, lowering the scissor lift with rubber bands will be more difficult, though you will have the weight of the scissor lift itself working with you when trying to lower the scissor lift. Counterbalancing is about finding the right amount of spring force to allow your lift to raise with weight easier, but still allow it to lower back down. See the example force diagrams below, and the example picture below. You would design the number of motors and gear ratio so that the motors provide enough just enough force (with a margin of course) to raise the lift up, and compress it. Providing just enough force means you use a minimum number of motors, and have the lift geared as fast as you can for that number of motors (give a very generous margin though!).

Raising (extension) of lift
Using example numbers to Illustrate idea, NOT actual calculations



Lowering of lift
Using example numbers to illustrate idea, NOT actual calculations





Example: Rubber bands are strung across the stages of the scissor lift, making compression of the lift easier so that it can raise with the weight of itself, and the weight of the cube, but still compress back down.

Another few pieces of general lift advice:

- Always use bearing flats wherever shafts go through metal
- Lightly and legally grease any moving parts
- Check the Vexforums for advice on building whatever lift you are trying out

Though giving a guide into building every type of lift is out of the scope of this guide, we can give some advice for the lifts we've tried.

Bar Linkages

See: http://curriculum.vexrobotics.com/curriculum/lifting-mechanisms/linkages for advice and http://www.aura.org.nz/archives/672
Also on chain lifts (good way to use sprockets with tensioner):
http://www.aura.org.nz/archives/991

Scissor Lift (gear and linear slide driven, not rack and pinion)

- Create a prototype of the lift and test before mounting it to the robot
- Connect the two sides of the scissor lift with a shaft and shaft couplers to the shafts directly driven by the motors on either side. Connecting both sides with the shaft will decrease the probability that the lift tilts to one side (the two sides are unevenly driven).
- Use greased inner linear slide inserts (Inner Acetal Slide Trucks rather than the outer acetal slide trucks). The outer tracks tend to allow the linear slide to pop out if the lift begins to tilt.
- Use calculated counterbalancing with rubber bands or latex tubing (latex tubing may work better as it does not need to be replaced nearly as frequently as rubber bands do)!

3.4 Intakes and Object Manipulators and Accumulators

The Vex curriculum links below extensively cover the topics of intakes and object manipulators and accumulators.

http://curriculum.vexrobotics.com/curriculum/object-manipulation/manipulators
http://curriculum.vexrobotics.com/curriculum/object-manipulation/accumulators

3.5 Pneumatics

The VEX Inventor's Guide has a very useful guide on pneumatics:
http://content.vexrobotics.com/docs/instructions/VEX_PneumaticsKit-INST-0712.pdf

As the guide mentions, pneumatics are best for applications which need fast deployment and considerable force (catapults are one example). You should be aware however, that the competition limits you to using 2 air reservoirs pumped up to 100 psi (air reservoirs can be joined by pneumatic tubing), meaning you will have limited strokes in a match. Additionally, these strokes will decrease in strength as the air pressure in the pneumatic system decreases from each stroke. You will need to pressurize the air reservoir before every match (preferably right before you put the robot on the field). Currently, the competition also gives teams a choice between 2 extra motors (for a total of 12) or 10 motors with a pneumatic system. Often times the 2 extra motors and a gearing system can work better than pneumatics, but this decision is situational.

There are both single action and double action pistons. Single action pistons extend with pneumatic pressure, and are automatically retracted by an internal spring; they therefore extend for only as long as they are signaled. Double action pistons extend with pneumatic pressure, and retract with pneumatic pressure; they therefore extend when they are signaled, and remain extended until they receive a signal to retract.

We typically recommend double action pistons. Because these pistons do not have to fight against an internal spring, they have a stronger force on their extend stroke. Additionally, double action pistons have a stronger return stroke (the single action only uses the spring's strength, the

pneumatic pressure of the double action is stronger than just that spring). Finally, double action cylinders have the advantage of allowing one to control the strength of the stroke in both directions via the pneumatic regulator (the single only works with a regulator for the extend stroke).

3.6 CAD

CAD, or Computer-aided design (or drafting) involves using computers to create, draft, modify, and/or optimize a design. CAD can prove to be very useful in robotics, both to detect potential design problems in advance of construction, and even to model mechanisms. One very practical and impressive examples of CAD in VEX involves using CAD to predict whether an arm will be able to lift both itself (using gravity in the model) and a certain amount of weight. This model can be extended to determine the maximum amount of weight an arm or lift will be able to hold before stalling, and what components in the mechanisms (such as a shaft) might break or bend. The videos at http://curriculum.vexrobotics.com/curriculum/speed-power-torque-and-dc-motors/simulate_and_size_a_dc_motor demonstrate these advantages clearly.

The VEX curriculum provides an excellent CAD introduction using Autodesk Inventor (normally a paid software, you can get a student license for free) in the unit: http://curriculum.vexrobotics.com/curriculum/intro-to-autodesk-inventor, and more advanced topics can be found later on throughout the curriculum. There's another set of videos you can also use, at: https://www.youtube.com/watch?v=C-vSc79XsO8.

As for the part libraries, you can either download the STEP files from each of their pages at the VEX store, (vexrobotics.com). Another option involves downloading a parts library (a large set of available parts). Parts libraries may also have imates (a feature on parts that makes it much easier to constrain parts to each other). We've used Team 24, Super Sonic Sparks's VEX CAD Library, which has worked quite well. The library can be found in the forum post at: http://www.vexforum.com/index.php/5635-team-24-super-sonic-sparks-vex-cad-library/0.
Unless it has been updated recently, it still requires **Autodesk Inventor 2012.**
If that library ever becomes outdated, Aperture Robotics also has a CAD library: http://www.vexforum.com/index.php/5625-aperture-robotics-vex-cad-library/0 (similarly, unless it has been updated, requires **Autodesk Inventor 2012.)**

If you complete CAD diagrams of your robot, you should definitely include them, (multiple views, exploded view, bill of materials, etc.) in your **Engineering Notebook, and make a note of the location of the diagrams in the index!** Even CAD diagrams of certain mechanisms are very useful, professional, and impressive.

3.7 Learning Mechanical Engineering on the Team

One excellent way to learn about robot mechanisms and the way a robot comes together is to help with the disassembly of the previous year's robots! This process can be very rewarding for new members, especially when you have an experienced member directing the disassembly process, as seen below.

Example: Directed disassembly of a previous robot.

4.0 Robot Electrical Engineering

4.1 Key Electrical Elements

There are a few key elements to keep in mind when planning out the power subsystem of the robot.

- The cortex
- Two batteries
- A power expander
- The backup battery and its holder

We almost always recommend using a power expander. A power expander allows a robot to use two batteries, with one battery directly powering four motors, and the other powering the cortex and the other 6-8 motors. The motors put on the power expander should often need the most power, and should preferably be related (for example the left and right sides of the drive train could be on a power expander if the drive train torque and speed is particularly important).

Also, please do not hold batteries by their wires. Holding batteries by their wires can pull the wires out of the plastic battery connection socket and render the battery unusable.

4.2 Placement of Key Electrical Elements

The Cortex, two batteries, power expander, and backup battery are the key electrical components of a VEX robot (besides the motors and wires themselves). When placing the cortex, it should be put in a position where it is easy to remove the Vexnet key and replace it with a programming cable and vice-versa. In addition, ideally the cortex should have some distance from nearby metal (which can interfere with the Vexnet connection), and should be easy to power on and off. Finally, the cortex should not be vulnerable or in a position in which it could be accidentally damaged by opponents or by driving around, and all cortex lights should be visible from the outside of the robot. After finding a spot for the cortex, the power expander should be placed so that wires can run from it to the cortex cleanly. Next, the two batteries should be placed so that they can connect to the cortex and the power expander without having their wires pinched or bent. In addition, these batteries should be in spots that allow them to be easily replaceable, so that batteries can be switched out before matches. The VEX Battery Extension Cable can often help a lot with this, giving more freedom to battery placement. Finally, the backup battery and its holder should be placed in a spot where it can easily connect to the cortex without its wires being pinched. The backup battery should be somewhat accessible, but will not need to be replaced as frequently as the Cortex and power expander batteries.

It is often useful to simply place these key electrical components, or screw them in lightly with a few KEPS nuts, and visualize how they will be laid out before setting their final attachment points with nylock nuts. In addition, (assuming the Cortex is in a safe position, batteries are not pinched, etc. as described above) the members wiring should test how easy it is to:

- Turn the Cortex on and off
- Switch out robot and power expander batteries
- Take out the Vexnet key and plug in a programming cable (and vice-versa)
- Plug in and switch out the backup battery

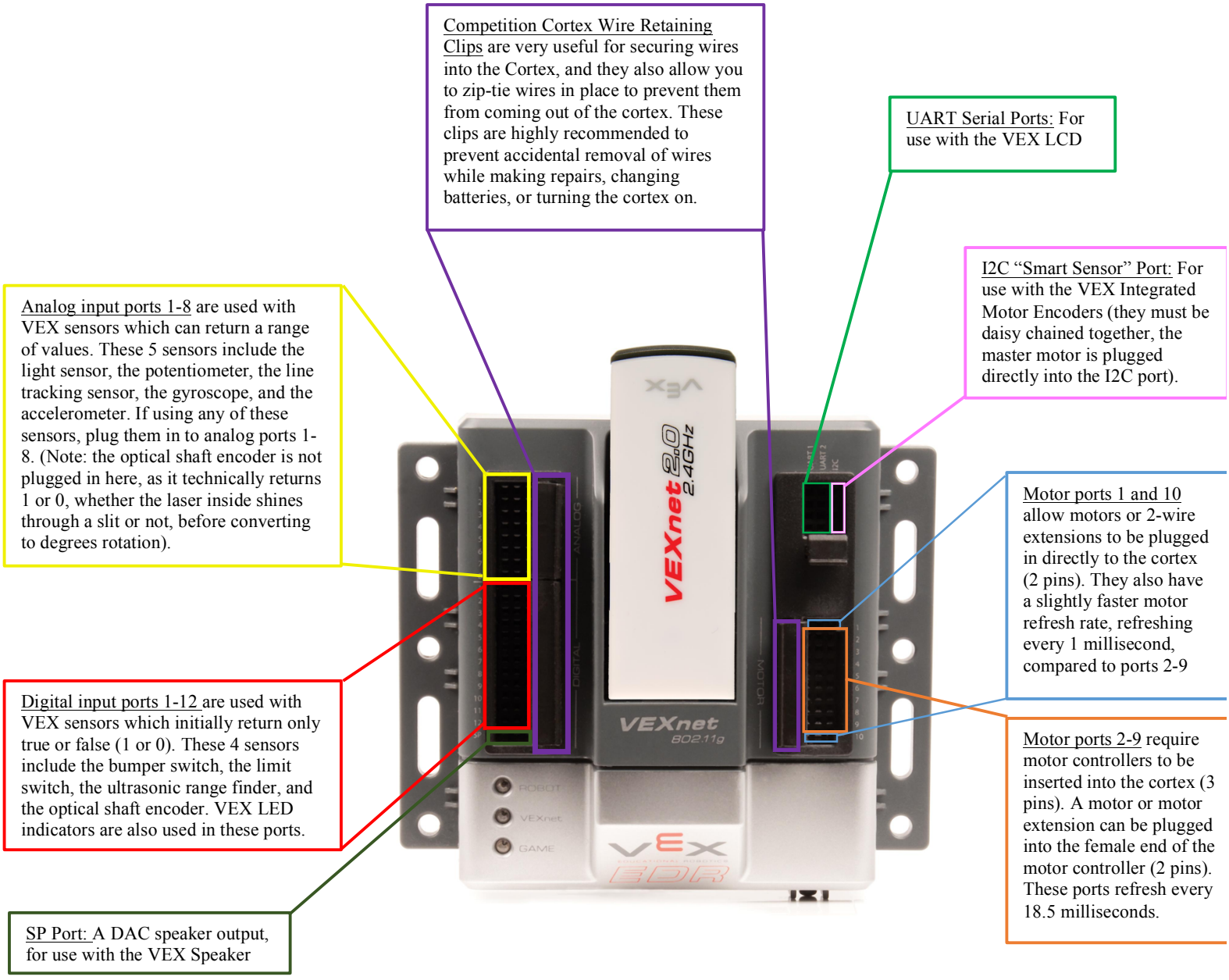
These operations should be quick, easy, and risk free (easy to perform without accidentally unplugging wires or pinching battery wires).

4.3 Wiring

When wiring, there are three main goals you should be striving for (don't be surprised if wiring takes a long time! Good wiring on a competition robot can take at least a few hours worth of time!)

1. Wires are out of the way, not pinched around corners or held down too tightly, and unlikely to be pinched by mechanisms. A team member (driver or engineer) should be able to turn the cortex on, replace batteries, or make a quick repair without interfering with wiring or risk pulling wires out. We often zip tie wires to the side of metal on the robot to run wires across the robot.
2. Wires have a bit of slack at each end (the motor and the cortex end). At the cortex the wires should have enough slack to move around a little, so that if someone's hand accidentally brushes against wires they are not pulled out. At the motor end the wires should have enough slack so they are not taut and stressed. In fact, there should be a small amount of slack everywhere wiring is done, but small enough that wiring stays organized, in place, and out of the way.
3. Wires should be labelled! This step is crucial. If a few wires come out right before a match and the members nearby cannot remember the wiring, there could be a serious problem. Place a fairly large piece of tape on each motor on the robot, and use a sharpie to write which port the motor goes to on the tape (or use a labelmaker). This port could be the power expander letter, or the cortex port. In addition, put a piece of tape on motor controllers and write which port the wire goes to on there too. Finally, if the wiring is particular complicated, it is often useful to put a piece of tape on the wire itself about 2 inches before it enters a port, so that if a wire comes out a team member can plug it back in quickly and directly without needing to trace the wire back to a motor or motor controller.

4.4 Wiring Tips

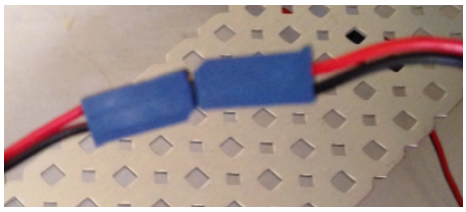


Base Cortex image (without annotations) credit to vexrobotics.com

- Before using a sensor, read the Vexrobotics manual, lookup wiki info (ex: https://www.vexrobotics.com/wiki/Optical_Shaft_Encoder) , and look up best practices, for example on Vexforum.com.
- Stick to a convention on a robot for wiring when using extensions cables (which allow you to wire over greater distances) and plugging in to the cortex. For example, we choose black to black at every wire connection to keep them consistent, and we reverse values in

code as needed. This convention means there is never doubt if a wire is unplugged or needs to be replaced. For plugging in to the cortex, the black wire should always face the "outside" of the cortex (away from the center). See the image below for these two conventions. Also, on a power expander the black wire should be facing to the outside when in the OUT and IN ports (shown in section 4.5).

Black wire faces "outside" of Cortex (Away from center)



Wiring black to black (image credit: Renegade Robotics)

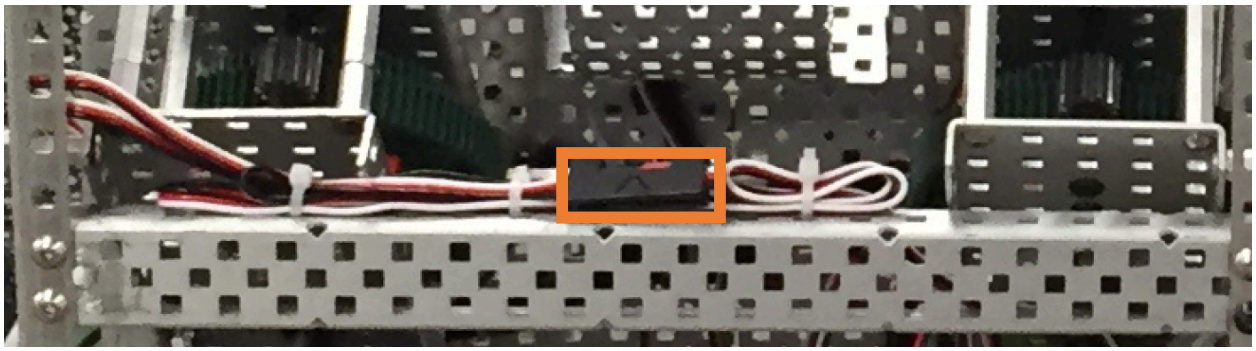
- When deciding what motor should go where, **be aware that Cortex motor ports 1-5 are controlled by one 4 Amp PTC, and motor ports 6-10 by a different 4 Amp PTC**. Therefore, it's better to split motors so half are in motor ports 1-5 and the other half in motor ports 6-10. Additionally, if using all 10 motors, it is best to spread the theoretical load across the two Cortex PTC's. For example, if it is suspected that a 4 motor drive train will come under significant stress, you can wire the left motors to motor ports 1 and 2, and the right motors to motor ports 9 and 10. The part of the robot under significant stress now has its load spread across the two Cortex PTC's, and is therefore less likely to trip a PTC in a match, which would stall (stop) the robot's motors, regardless of driver input.
- There are also PTC's within the Vex 393 motors themselves.
- It is also often important to talk with the software engineers on the team when deciding on a wiring plan. Software engineers may have input on an important wiring factor, particularly when dealing with PID Loops and autonomous (motor ports 1 and 10 update

motor speed at around 60 Hz, but motor ports 2-9 use motor controllers, which in practice makes updated motor speed notably slower).

- Still, the motor load should be spread across the Cortex circuit breakers, and remember it is often easier to change code rather than wiring.
- It is often useful to wire one motor at a time, tracing your planned path without zip ties, then zip-tying wires down, then moving on to the next motor.
- Be very careful when removing zip ties! It is very easy to accidentally cut into wires or nick them while cutting zip ties.
- When wiring on or near a portion of the robot that moves (a lift for example), physically move (slowly!) that portion of the robot back and forth and to its full extent to ensure the wire does not get pulled or pinched.
- If there is any metal exposed in a wire or fraying, the wire may short, and could fry the motor or cortex motor port (motor ports 1 and 10 in particular). Replace these wires/motors as soon as possible.
- Y-cables are used to allow one motor port to run multiple motors. You may only use one y-cable per motor port, and may not exceed the motor limit for the year's competition (or y-cable limit if one is in place). Y-cables are often used if the year's competition motor limit is greater than 10 (as there are only 10 motor ports on the Cortex). However, it is important to understand that whatever motor value is sent from the Cortex will then be applied to **both** motors. Thus y-cables can be used for each side on a typical tank drive train, as theoretically the motors on each side of the drive will always be spinning in the same direction (unless you want the motors on one side turning at different speeds from one another).
- You may need to break convention (black-black wiring) when using a y-cable to reverse the polarity of one of the two motors. Try to avoid this whenever possible (use the y-cable on a different pair of motors), but if unavoidable you can wire red to black to reverse the polarity of a motor (a motor being sent a signal of 127 from the cortex will instead output -127 with red-black wiring). You cannot reverse polarity **at** the Y-cable to wire connection though, only in the wires leading up to it.
- Motor controllers have two bumps on the bottom which fit nicely into VEX metal square holes. Additionally, notches along motor controllers fit zip ties nicely for attachment to metal when wiring.

4.5 Some Examples

The image below shows how wiring was done using zip ties to run wires, and how to deal with excess wire that would normally get in the way. The excess wire was bundled and zip tied to metal just as if it was being run like normal. Additionally, the zip ties were not overly tightened anywhere along the wire, but were not so slack that the wire could slip out. Wires were kept well out of the way of the gears shown, as the metal 12 tooth gears (and the flywheels on the same shaft) spun at 1633 RPM. Finally, this image shows a wire retention clip (in the orange box) being used to hold extension cables connected to encoder wires. Use a retention clip anywhere you join a wire to an extension (wherever the male and female ends of wires meet).



VEX's wiring diagram for a motor controller (outdated microcontroller shown with only 8 motor ports, and only motor 8, the top motor, is wired to the power expander; motor 1 is wired normally). Also notice the black wires are facing to the outside at the OUT and IN ports, and the wire only fits one way in the STATUS port (because of the small notch inside the port).

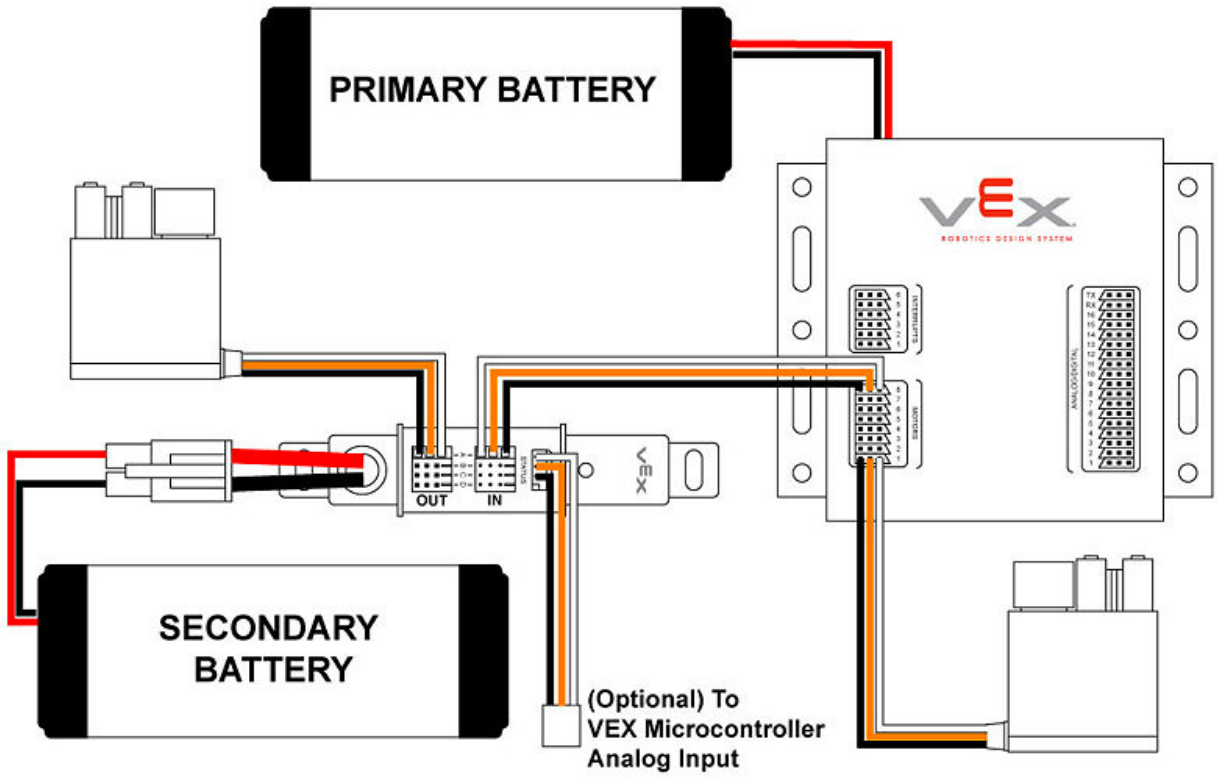
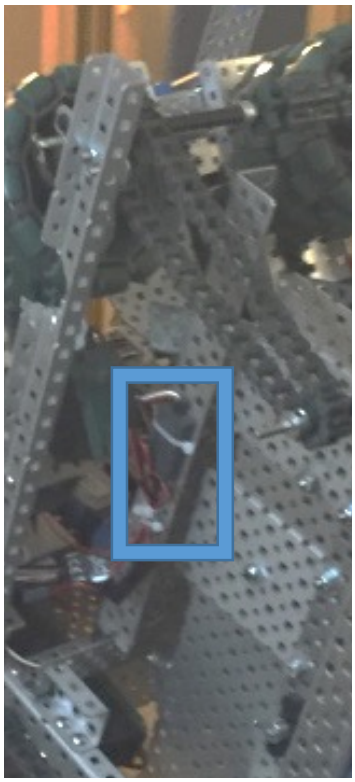


image credit to vexrobotics.com

Using zip-ties to hold down a motor controller:



5.0 Robot Software Engineering

5.1 An Overview

Software Engineering in robotics is deceptively simple. Autonomous routines seem like a breeze, drive straight for this distance, then stop, turn, intake a scoring object, turn, and score. However, simple commands such as driving straight for a set distance, and executing a perfect turn, can actually become quite complex very quickly. Sure, you could set the robot to drive forwards for a set amount of time, but the distance travelled will vary based upon battery voltage, friction, motor and physical inconsistencies. Even using encoders to measure the number of rotations, and multiplying that number by the circumference of the robot's wheels, is not perfect, as there are still variances in friction, and most importantly in the way the robot drifts when it comes to a complete stop from a high speed.

The main takeaway is that timed motor movements usually do not work well. Sensors are key for automation, and autonomous can make the difference.

Finally, every Software Engineer on the team should also have a solid base in Mechanical Engineering as well. Some mechanism or part of the robot will often fail over the course of autonomous testing. If a Software Engineer knows how to perform basic repairs on the robot, then they do not have to completely halt progress and wait until a Mechanical Engineer is in the lab to fix a small problem that came up during autonomous testing.

If you're looking for our past team's code, including **a fully annotated version** of one team's code that's **great for example code,** you can go to our github!

https://github.com/login

Our username is: ChoateRobotics

Our password is: Hugo1_Charlie2

(Please, do not share with non-team members. They can view our code fine without the password, the repository is public. We do not want non-team members editing our code.)

Regarding the annotated code, the repository is titled: 2015-2016-6106A-Annotated. Read the README markdown for more information on the individual files (the main code is commented throughout).

5.2 During the Build Period

Software Engineers will likely spend most of the build period brainstorming ideas, and helping Mechanical Engineers construct the robot. If there are plenty of people helping with the robot already, Software Engineers can begin to create code that can be run to test robot mechanisms as they are built, and eventually work towards creating driver control code. Generally, driver

control code should be made as soon as possible, so that when the robot is constructed drivers can begin practicing as soon as possible.

Additionally, Software Engineers can look into ensuring that sensors are installed on the robot in appropriate places (such as encoders!), and getting more complex code, such as PID loop control, ready for tuning. Software Engineers can **learn Github (highly recommended, almost a necessity! See section 5.9)**, test new algorithms or learn about other code being used for PID controllers, gyroscopes, gyroscope PID, and even create new sensor code.

Software Engineers can also start to plan out an autonomous strategy set and discuss their ideas and strategies with the whole team.

5.3 Your Choice of Integrated Development Environment (IDE)

There are a few different options to consider when you are going to program a VEX Robot. Namely, which IDE you will use. EasyC and ROBOTC, offered by Robomatter, are both designed to work with the VEX Robotics Design System for programming. These IDE's use a modified version of c-based programming, and have different environments. EasyC is primarily block-based programming. It is designed for beginners just learning how to code, and works fairly well for that learning process, but it can also be switched to text-based programming. ROBOTC is typical text-based programming, but is still geared towards beginner programmers.

As for the debate between EasyC and ROBOTC, both are viable. In general, we have chosen to use ROBOTC for a few reasons. One large reason is ROBOTC's real time debugger. This debugger allows us to view all sensors, motors, variables, and more while code is running. This debugger has proven to be very useful for us, particularly for tuning PID code and, of course, for general debugging. ROBOTC also offers multitasking, a more advanced programming feature which offers a lot of capability to those who learn about it (easy-to-use slew motor control for example).

There are other options, namely PROS (Purdue Robotics Operating System) and ConVEX. These languages are not officially supported by any companies, PROS being made by the Purdue Robotics team, and ConVEX being open source.

We did use PROS for a year (at first primarily for its Mac OS X support), however now that the team has more up to date Windows laptops specifically for team use, and because of ROBOTC's real time debugger and more in depth documentation, we have switched to ROBOTC.

PROS features include (from http://www.vexforum.com/index.php/10048-purdue-robotics-os-pros-now-available/0):

• Development on Windows, Linux, or Mac OS X

• Eclipse-based environment with one-click download and integrated source control for easy collaboration

• Runtime library written from scratch that takes full advantage of the Cortex Microcontroller's power

• Direct compilation to native C allows use of structures, pointers, dynamic memory allocation, and functions familiar to students

• Use of standard tools (GCC), coding standards (ISO C99), and function names (libc, Arduino) minimizes learning curve

ConVEX is **not** recommended for beginners, or even intermediate programmers. ConVEX does not have an IDE, or an installer, and the creator, jpearman (on the VEX forums, an excellent programmer who is also currently VP of Technology at Robomatter) outright says beginners should not attempt to use it (at risk of bricking your cortex!). ConVEX is for those who want to program with real time operating system (RTOS) concepts and who can program micro-controllers without intermediate software making programming easier. For our robotics team, ConVEX is not suitable, as it does not allow beginning programmers a chance to program, and it can make communication between Software Engineers of different levels very difficult.

5.4 Using ROBOTC

There are many resources for ROBOTC for beginning programmers. Please go through the tutorial linked below to get a very solid introduction to programming in ROBOTC.

http://www.education.rec.ri.cmu.edu/products/cortex_video_trainer/index.html

The modules at the top are different topics the tutorial covers. All modules contain useful information, both for programming and for building!

This introduction will serve you very well and provide a solid basis so you are ready to program competition robots. Of course, practice is essential to become a good programmer, particularly with concepts such as PID loop tuning.

5.5 General Coding Practices

Please, please, **comment code thoroughly**! It makes it much easier for a team to collaborate, spot errors, and work together on code if everyone comments their code explaining what different lines or functions do. If there's a line that took a while to figure out or could potentially be confusing, write a comment explaining it and the thought process.

Use functions: If you're writing code and you see yourself repeating multiple lines even a few times in your code, write a function for it. It also allows us to easily re-use code in later years!

Translate units: Rather than measuring in encoder ticks, or ticks per second, put in a line converting ticks to degrees, or ticks per second to rotations per minute. It is much clearer, and much easier to find errors when you are programming with units you are familiar with.

Use reasonable and understandable variable names: You and others on the team will be thankful for clear, understandable variable and operation names.

Program clearly (partially adapted from AURA): When working in a team setting, even if working on code by yourself, please keep code simple. Though you could save some memory by avoiding a unit conversion, or you could save a few lines by using a ternary operator, it's simply not worth the sacrifice in code clarity. If you've been away from the code for a while, you may forget why you did what, and others on the team won't even be able to understand what's going on, or will waste time figuring out code which could have just been kept clear in the first place.

5.6 Optical Shaft Encoders (also called Quadrature Encoders)

Optical Shaft Encoders are likely to be one of the most important sensors on your robot. **The first thing to note about encoders is that they should be placed as close to your wheels as possible.** The closer the encoders are to the wheels, the less backlash the wheels will have (the less they will be able to turn without the encoder measuring a tick).

Additionally, it may be **extremely beneficial** to replace your wheels' plastic square inserts with the metal inserts used on high strength gears. **As a warning,** it can be very difficult to get these plastic inserts out, and you should check if the wheel you're trying to modify can have its insert safely removed. However, the encoders will likely keep a better track of the wheel's position, and it may also make a large difference in driver control, and how well the robot responds to acceleration or deceleration.

One team has used a pair of shaft lock bars to avoid having to modify the wheels themselves. Be warned that it can be difficult to push shafts through, or get shafts out of, shaft lock bars, however this method is suitable if you wish to avoid modifying the wheels themselves.

Credit to George Gillard of Team AURA

As for wiring Optical Shaft Encoders, we usually plug wires from the same encoder right next to each other (for example into digital ports 1 and 2). When using the Motors and Sensors Setup menu on ROBOTC, after specifying one of the two digital ports (ex. digital 1), ROBOTC will prompt for the "second port," the port the second wire is plugged into (ex. digital 2). We usually input the lower port number wire first, and put the higher port number in as the "second port,"

for consistency's sake. Definitely label the wires on the robot, as if both of an encoder's wires come out, it can be easy to mistake which one went into which port, and that wiring needs to be consistent.

If the encoder is outputting strange values (for example, always 0 or a low number regardless of rotation), you will first want to check your wiring (you may need to flip the ports each encoder wire is going into). After testing again, if unusual values are still being output, you may want to open the lid of the encoder. If there are scratches on the encoder disk, or if there is dust, that may be interfering with the laser and giving odd values. You may want to replace the encoder, or blow the dust out.

If you are going to use PID Control with an encoder, we highly recommend putting the motor being measured on ports 1 or 10. Ports 1 and 10 have a higher motor refresh rate, and will thus allow the PID loop to update motor values more quickly.

5.7 Proportional-Integral-Derivative Control

There are a variety of guides which excellently explain PID control and PID tuning. Rather than reinventing the wheel, we would redirect you to read through the guides these sites have available, which make PID loops very accessible. Going through these guides and past code will allow the creation and tuning of a PID loop:

This guide explains the concept of PID control very well:
https://www.dropbox.com/s/yg25rtawb0sboxw/PID%20Guide.pdf

This guide provides a very good explanation of how to tune a PID Loop:
https://docs.google.com/document/d/1D61dinGqP__CHzRfc9yNugnS9K-JTa395_r0qP7L5W8/edit?pref=2&pli=1

The guide mentioned in section 5.4 does also go over PID loop control, using integrated motor encoders (IME's). Because IME's have known to experience problems due to static electricity, we generally tend to stay away from IME's and use encoders. And, once again, **if you are going to use PID Control with an encoder, we highly recommend using ports 1 and 10.** Ports 1 and 10 have a higher motor refresh rate, and will thus allow the PID loop to update motor values more quickly.

You can use the example code and build-in PID algorithms that ROBOTC provides, as covered by the tutorial. We have never actually tested ROBOTC's built in PID algorithm, so feel free to try it out, and **add a few paragraphs about how it went**. We typically use algorithms we have created or modified from other teams, so that we can fine tune the algorithm for our robot.

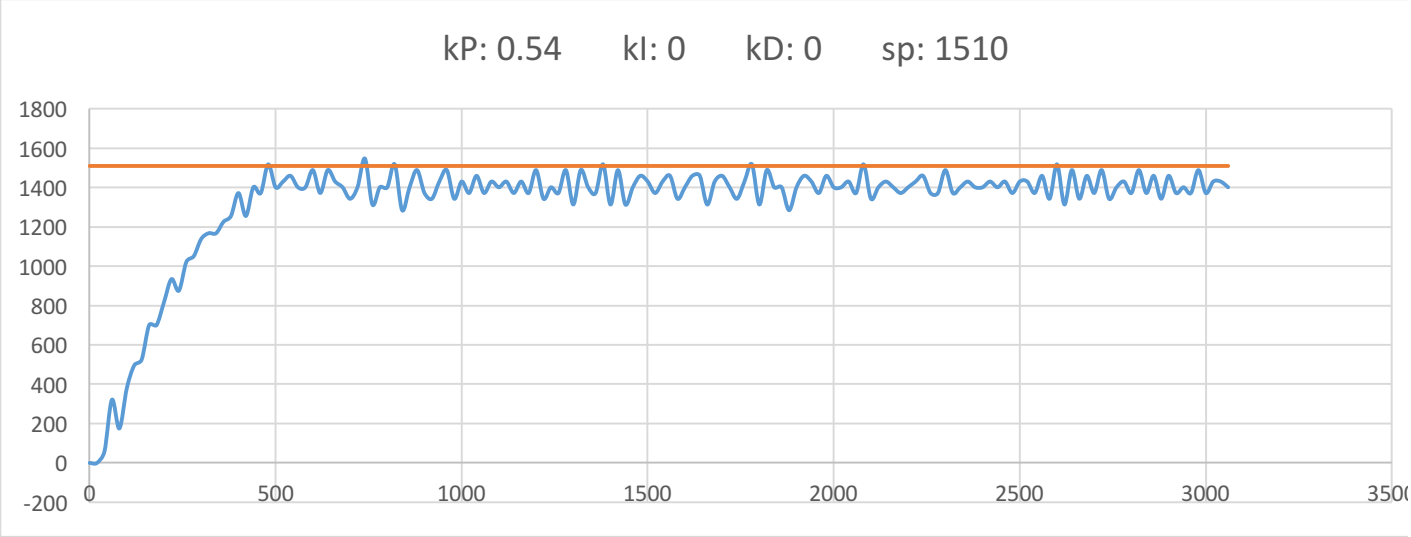
PID Loop tuning can require hours of time, especially if it is your first time. Gradually you will get a feel for the tuning process, and will be able to tune faster, but it does take a significant amount of time.

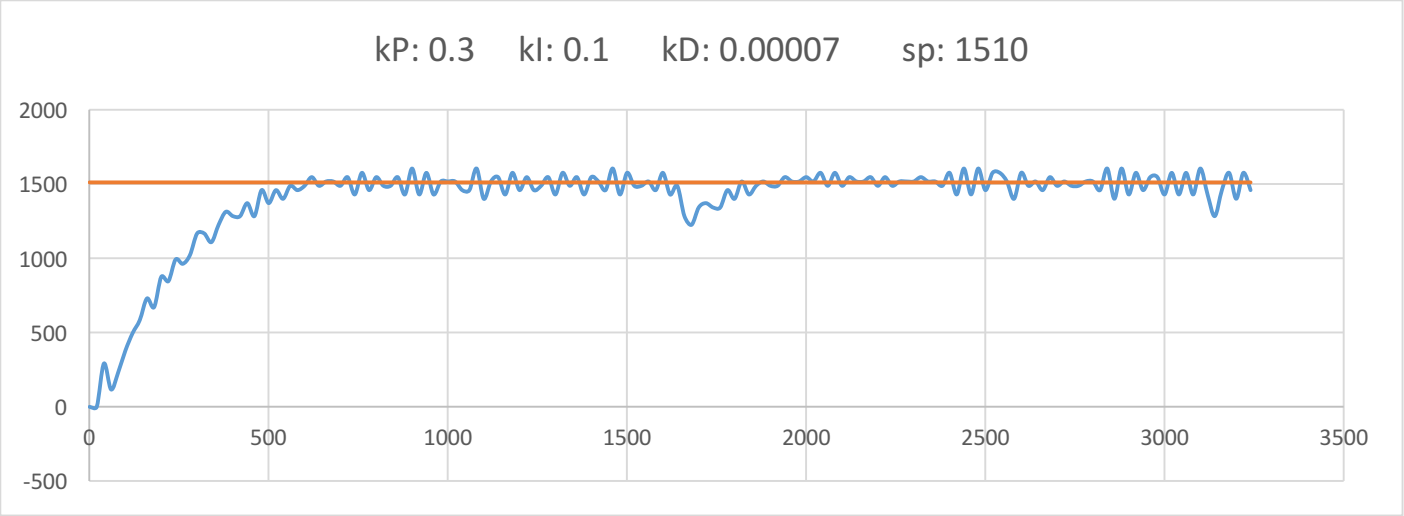
In order to see how well tuning is going, you can use ROBOTC's real time debugger. The t code on our Github shows an example of us simply outputting values to the debugger stream with the command: writeDebugStreamLine("%f", degreesTravelledR);
These values can then be copied over to an excel document for graphing. For the time for each measurement, we usually simply increment 20 milliseconds graphing in encoder measurement vs milliseconds (20 milliseconds from the 20 millisecond wait we use at the end of the while loop). More ideally, we can also use log nSysTime to the debug stream as well right before outputting the RPM.

Example:

Time (milliseconds)	RPM
0	0
20	0
40	291.666656
60	116.666664
80	233.333328
100	379.166656
120	495.833344

And so on. Then create a graph (we typically use scatter with smooth lines). The graph should look something like this:
Shown below are two example graphs. The orange line is our setpoint, or sp (our target RPM), and the blue line shows what the encoders measure as our flywheel's current RPM. The titles contain the constants which were used to produce this graph, and our setpoint. (The graphs have been shortened in width to fit on the page).





As you can see, tuning by even small amounts can make a big difference! You can also see where we actually launched a ball (at about time = 1600 milliseconds) and that we had a recovery time (time until the RPM stabilized again after firing) of less than half a second. Once again, the sites linked above have very good example graphs showing how tuning the constants affects the output. Those graphs may be different from the ones shown here, as flywheel tuning involves keeping the wheel spinning at a constant RPM, other applications for PID are driving a set distance, which would likely have a different graph (likely would not oscillate nearly as much).

5.8 Yaw Rate Gyroscope Sensor

The Yaw Rate Gyroscope is a single-axis gyroscopic sensor used to measure rotation in degrees per second, and outputs the degrees a robot has rotated.

The Gyroscope should be placed in a location on the robot with the **least amount of vibration and motion.** Vibration can lead the Gyroscope to accumulate error, and drift over time. In addition, we usually like to **mount the Gyroscope by using VEX Rubber Links**, to minimize drift by absorbing some of the vibration and shock that may otherwise be transferred to the Gyroscope.

It is also important that the gyro has a solid amount, that you use a short cable to the cortex, that it is stationary when initialized, and that you keep interference away from it (do not keep motor wires near it).
(http://www.vexforum.com/index.php/8820-gyroscope-kalman-filter/0)

The Gyroscope should be wired so that the ground wire (black) is next to the white letter B that is silkscreened on the Gyroscope. On the Cortex end, the Gyroscope can be connected to any analog input on the Cortex.

Though we typically use the Gyroscope with QCC2's Gyro PID Code, to program the Gyroscope on its own, you can follow the guide at this page:
http://www.robotc.net/blog/2011/10/13/programming-the-vex-gyro-in-robotc/

The library that we currently use with the Gyroscope sensor is Jason McKinney's QCC2 Gyro PID code.
http://www.vexforum.com/index.php/14219-qcc2-gyro-and-pid-control-code/0
It also requires tuning a PID controller loop, as covered by section 5.7. Example implementation is again found on our Github, under the 2015-2016 annotated code repository.

Additionally, you may want to try this code by James Pearman (jpearman) made back in 2012:
https://github.com/jpearman/RobotCLibs/tree/master/gyroLib
We've not yet tried this code, so if you do try it, test it against the library we currently use! Then you can **write about how it went in this section.**

You may also try using a modified PID Drive function (different sides of the drive turn different directions) with the wheels turning the appropriate number of degrees for the turn. In fact, this approach may be simpler and work better than using the Gyroscope if it can be tuned so that different degree turns do not require different PID constants (as can happen with Jason McKinney's QCC2 Gyro PID code).

5.9 Autonomous Development and **Source Control**

When developing autonomous, the autonomous plan should have been developed as a part of the team's strategy set (section 2.28 and 6.1). When creating the team's final competition autonomous code, before every test run of the autonomous code they should ensure that:

1. The robot is in the same starting position every time and this position is repeatable, either with an alignment tool or alignment with field tiles in some way.
2. The robot and power expander have fully charged batteries at a consistent voltage. There is some variation in how quickly/powerfully the robots motors respond depending upon battery level, so it is important to ensure that battery levels are consistent across

Batterys should be recharged and switched every few autonomous runs.

Regarding code **source control, or version control,** our team has a Github. This site lets us **backup our code to Github (please always back up code!!!)**, and also provides source control (logs every "committed" version of code, and allows us to revert to past versions of the code). Source control acts as our programming "Engineering Notebook" and shows the history and development of our code. Additoinally, source control lets us revert to previous versions of code, in case a member accidentally changes the code and it does not work. Github also lets us see the difference between multiple versions of code, see what changes may have broken code, lets us collaborate and communicate (everyone on the team can commit to Github).

For a few tutorials on Github, and how to use it, we recommend these two sources below:
http://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1/
http://readwrite.com/2013/10/02/github-for-beginners-part-2/

Or:
https://guides.github.com/activities/hello-world/

Or:
http://product.hubspot.com/blog/git-and-github-tutorial-for-beginners

5.10 Example Code

If you're looking for our past team's code, including **a fully annotated version** of one team's code that's **great for example code,** you can go to our github!

https://github.com/login

Our username is: ChoateRobotics

Our password is: Hugo1_Charlie2

(Please, do not share with non-team members. They can view our code fine without the password, the repository is public. We do not want non-team members editing our code.)

Regarding the annotated code, the repository is titled: 2015-2016-6106A-Annotated. Read the README markdown for more information on the individual files (the main code is commented throughout, providing examples of PID Loop Controllers, GyroPID, etc.).

5.11 Further Reading

We can not hope to cover every useful tip for competition programming in this guide. There are simply far too many ways to improve robot drive and autonomous performance through programming, or even just convenience features (LCD autonomous chooser for example). We've compiled a selection of sources which provide further reading on the subject of Software Engineering in VEX Robotics.

http://www.vexforum.com/index.php/6146-robotc-programming-tips
http://www.aura.org.nz/educational
https://vamfun.wordpress.com/category/vex/
http://www.vexforum.com/index.php/10949-gyro-drift-mechanical-improvements/0
http://www.vexforum.com/index.php/8197-smart-software-monitor-keeps-ptc-fuses-from-tripping/0
http://www.vexforum.com/index.php/7868-motor-torque-speed-curves/0#post300280
http://www.vexforum.com/index.php/8089-motor-torque-speed-curves-rev2/0
http://www.vexforum.com/index.php/7942-cortex-motor-speed-testing/0#post302395
http://www.vexforum.com/index.php/7955-estimating-motor-current/0
http://www.vexforum.com/index.php/8390-estimating-motor-current-part-3/0
http://www.vexforum.com/index.php/8431-smart-motor-library/0
http://www.vexforum.com/index.php/6465-a-pid-controller-in-robotc/0
http://www.vexforum.com/index.php/17928-discussion-on-using-tasks-in-robotc/0
http://www.robotc.net/wikiarchive/Multitasking_tips
http://www.vexforum.com/index.php/9800-24c-s-motor-control-value-remapping/0
http://www.vexforum.com/index.php/13416-introducing-bnslib-an-advanced-programming-library-from-team-bn/0
http://www.vexforum.com/index.php/15128-robot-position-calculation-using-simplified-filter-smoother/0
(Kalman filters are likely overkill for VEX, but still interesting to learn about)
http://home.wlu.edu/~levys/kalman_tutorial/
https://en.wikipedia.org/wiki/Kalman_filter
https://en.wikipedia.org/wiki/Extended_Kalman_filter

6.0 Testing and Practice

6.1 Strategy Set

By the end of the build period, the robot will likely not function exactly as planned. It may be faster than expected, a mechanism may not be entirely reliable, or maybe the robot stalls out if turning too much. Regardless, the set of strategies envisioned at the beginning of the season probably needs to be modified after the robot is driven on the field and tested. For each robot, its whole group should be present after a test session, or even the first Drivers' Practice (covered in 6.2) to evaluate the robot's performance, and discuss modifications to the set of strategies. It is the coach's job to fully understand the final set of strategies, find the best way to achieve one of them during actual matches, and direct the driver(s). Everyone on the team should be aware of their final robot strategies going in to a competition. The strategy set should cover scenarios such as what to do if we are behind the opposing team near the beginning of a match, or how to counter opponents which perform better than us, etc. **In addition, all strategies used should be recorded in the Engineering Notebook, preferably in a dedicated strategy section.**

6.2 Drivers' Practice

Being a member of the Drive Team is a large responsibility. The drive team has to be able to communicate effectively and clearly with one another, and drivers have to be quick and well-practiced with the robot. The Drive Team is therefore expected to practice driving and manipulating scoring objects, as they would in a real match. The coach, and both drivers (if there are two) should be present at the practice. The coach should have a clear strategy set in mind that was previously discussed with the whole team, and the drivers should be aware of the strategy set as well. However, drivers should focus on driving, and should always follow the coach's directions, as the coach also needs to choose which strategy to follow. The Drive Team should begin practicing as soon as the robot is ready for driver control (mechanically and programmatically), and should ideally rehearse each strategy in the strategy set. The software engineers can work on the autonomous after the Drive Team is all set for practice in operator control. Ideally, the Drive Team should practice on fully charged, or almost fully charged batteries. It is also highly recommended that the team have a Backup Drive Team, a completely different set of Coach and Driver(s) also getting practice in. If for some reason any member of the Drive Team is unable to attend a competition at the last minute, a member of the Backup Drive Team can fill in.

It is up to the team captain/the group leader to decide how much time the Drive Team needs to practice, however, they should practice a significant amount, and should get practice the week leading up to a competition. **It is often better that we get practice and testing in instead of adding extra features in at the last minute.** Additionally, the team's backup Drive Team should also get practice in, just in case anyone on the main Drive Team is unable to make the competition at the last minute. In a situation where Backup Drive Team and Drive Team practice may interfere, preference is usually given to the main Drive Team for practice.

Finally, someone on the Drive Team, usually the coach, should be aware of how the robot needs to be set on the field for autonomous, see section 6.3 for more information.

6.3 Autonomous Practice

As stated in 6.1, someone on the Drive Team, usually the coach, should be aware of how the robot needs to be set on the field for autonomous. The software engineers will usually assume a certain position for the robot to begin the routine, and this position is typically very important for the autonomous to perform well. This position may require precise alignment or use of an alignment tool. The member of the Drive Team who is to setup the robot on the field should practice this alignment, preferably with a software engineer, to ensure that they understand the positioning, and that autonomous is functional and ready. Autonomous should also be performed with full batteries.

7.0 Team Structure

The team structure will likely differ from year to year, and general organization is up to the team's captain and mentors.

7.1 General Advice

Despite the changing team structure, there are a couple of points we would like to stress:

- **Communication** is critical. This point applies to all aspects of the robot, if a group is to function well, it needs to communicate constantly. A member cannot simply expect fellow team members to be in the lab or show up when they are in the lab. Teams should communicate, both through their group leader and however the leader chooses to organize (Google Docs, group chat, etc.). All communication should include every member, the group leader, and the team captain for organizational purposes.
- **Be friendly, accepting, understanding, positive, and kind.** We want to create an encouraging environment that we enjoy and where we feel free to share ideas.

8.0 The Competition

8.1 Our Attitude and Representation

When we attend a competition, the entire team should know that each member represents themselves, represents their team, and represents this school. Each member should know to comport themselves well, be courteous, kind, respectful, helpful and polite. In addition, each member should be a good sport regardless of a win or loss. The drive team should shake hands with their teammates and opponents. If the team had a big win, they should feel free to be excited, but they should not be discourteous or excited to the point that the other team does not feel good. The team can share some of that excitement with teammates back at the pits. School rules are also still in effect at competitions or in hotels if at a longer competition. Finally, particularly at larger competitions, team members should store the phone numbers of all other team members (their group leader, the team captain, and team mentors at a minimum). The captain or mentors should distribute a list of phone numbers with hotel room numbers at larger trips.

Overall, we want to be a team that is kind, courteous, professional, and helpful to everyone.

8.2 What to Bring (Packing)

Going to competitions is a great experience, and all team members are highly encouraged to attend! Competitions can be hectic, and often motors will strip, gears will snap, chain will pop, and repairs or changes will need to be made. Hopefully the previous captain or other team member has passed on a packing checklist from last year. A packing checklist is important, as if you go through the checklist properly it almost ensures that you will not forget anything important at a competition. The packing checklist should include every key part of the robot, from spare wheels to spare treads or chain, lots of robot batteries and chargers, spare motors, motor gearing kits, extension cables, sensors, and even spare metal. The checklist should also include the robots(!), power tools such as the dremel, a vice, and anything needed for competition repairs. The packing checklist has been in circulation since the end of our 2014-2015 season, and should be passed down year to year and modified as robots change. Important parts, and parts that may be mixed up, such as batteries, multi-meters, **joystick controllers**, power strips, etc., should be labelled! Whenever possible, pack robots in inspection-ready condition! An inspection checklist can be found in section 8.5.

For storage containers, we like to use tackle boxes for storing a lot of our parts. We have a smaller tackle box for our allen keys, wrenches, shafts, Nylock and KEPS nuts, gussets, etc. We have a toolbox to store our larger allen screwdrivers, Dremel, safety glasses, multimeters, files, sanding blocks, etc. We have 2 larger tackle boxes which we use to store our gears, sprockets, spare motors, spare motor gears, spare wires, etc. And finally, we bring a couple of larger containers to store our spare metal, joysticks, vicegrip, etc.

8.3 Competition Jobs

The roles that team members hold in the lab will likely change slightly when at a competition. Therefore, it is important to hold a team meeting before a competition to discuss which roles everyone will fill, who will hold walkie talkies for communication, discuss strategy, and for any other organization before the competition.

The roles that can be filled at a competition are as follows:

Driver(s): The team members who drive the robot during competition matches. There can be one or two drivers depending on what the team decides, and the drivers should have gotten sufficient practice with each other and the coach during Drivers' Practice leading up to the competition. Drivers follow the instructions of their coach, and with the coach make up the Drive Team. The Drive Team should also keep in good contact with the Pit Mechanics, being able to describe what went wrong (or went well) with the robot in a match.

Coach: The team member who is a member of the Drive Team and must choose which strategy to use in a match. The coach stands with the drivers, and directs them over the course of a match. The coach should also communicate with scouts to decide which strategy of the strategy set to use in a match (or to make and record a new one). The coach also decides if the match is going well enough to score Strength of Schedule Points (SP), covered in section 8.7. The coach, along with Pit Mechanics, go through the pre-match checklist before competing. Finally, coaches should have a match list, and know exactly when the Drive Team needs to be ready for a match! The coach usually keeps a walkie talkie on them to let others know they have queued (this also means that the coach should always stick with with Drive Team).

Pit Mechanic: Pit mechanics should be fast builders with significant mechanical knowledge of the robot. The pit mechanics should be prepared to fix absolutely anything on the robot should it break. Additionally, the Pit Mechanics and the Drive Team should develop good communication working together. Often times the Drive Team knows best what went wrong, and they need to communicate clearly to the Pit Mechanics what happened so they can figure out how best to fix the problem. Pit Mechanics also work with the coach to go through the pre-match checklist, to ensure the robot is ready to compete before a match. There can any number of Pit Mechanics, but likely one to three per robot should suffice. Usualy there is a pit walkie talkie, to prepare the Mechanics for future changes, or the Battery Boss to provide batteries.

Head Scout: The Head Scout is the most experienced scout. This scout is responsible for directing the other scouts and organizing which scocuts will watch which robot in each match, and when they will talk with our future alliance partners in qualification matches. The Head Scout will also compile all data collected by the other scouts on their scouting sheets. Finally, the Head Scout will also scout strongly viable alliance partners as per the compiled data. The Head Scout must have very good knowledge of the robot's capabilities, and is also likely the person to talk to other teams about alliancing. The head scout may also be a designated spokespearson for the robot, ready to talk with teams at the pit, or to judges along with the pit mechanics and whoever is in the pit at the time judges come around.

Scouts: These team members will watch matches and robots as directed by the Head Scout. They will record all information in a scouting sheet developed by the team for each year's competition. They will also talk with the coach if there is a particularly difficult match to provide strategy advice against particular opponents that they have scouted. Finally, the scouts may be asked to scout our team's next alliance partner in qualification matches, to determine what autonomous our partner has (and which tile they prefer to start on), and which strategy we should use to work best together. One of the scouts can potentially have something to record with, and can record our matches for analysis. A scout may also take a look at a few standout robots or mechanisms; you can also learn a lot from other teams at the competition too!

Battery Boss: The battery boss is responsible for charging and providing every one of our robots with charged batteries before their matches. This job is critical for the team; different robots may have different preferences for battery level (including different preferences for the cortex and the power expander battery), and robots often need a charged battery to perform at their best. The battery boss can be managing around 20 different batteries, 12 chargers, or even more (as power expanders allow for 2 batteries per robot)! The battery boss should be charging as many batteries as possible, checking battery voltage with a voltmeter, and organizing batteries by their charge so they are ready for robots before their matches. The battery boss may also want a match list, with **our team's matches circled or highlighted** (different colors for each of our robots, **the team captain and group mentors should do this too**) to know when they should have batteries ready. There may be one or two battery bosses, and one of them can be a Battery Runner, who brings batteries to the Drive Team when in need of a last minute replacement, or between elimination matches.

Software Engineer: The team's Software Engineers will be charging laptops and ready to modify autonomous or driver control code as needed. These members will also ensure that the correct autonomous code (for different sides of the field, different tiles, or different strategies) is loaded when the coach and pit mechanics are going through the pre-match checklist. There are often autonomous tweaks that need to be made, so these members should be able to modify code quickly on the go. If there are more Software Engineers than needed at a competition, they can become scouts.

Each Group Leader may take on one of these roles, or may be a general organizer for their group. The Team Captain can also take on a role, such as Pit Mechanic, and typically ensures all teams are queued, have everything they need, and can often fill any role, be it emergency repairs, scouting, or charging batteries.

8.4 Pit Setup

Upon arrival at a competition, the first thing the team should do is locate their pits. The pit is where the team prepares the robots for competition, makes repairs, charges batteries, stores tools, and stores the robots in-between matches. When the pit has been found, the team should unload the robots, unload the main tool bins, find the nearest outlet (we bring an extension cord in case the outlet is far) and setup all battery chargers and power strips, and get the batteries charging. The battery boss will be in charge of monitoring and charging batteries. Any changes to the robots that are needed before inspection should be made as soon as possible, and unneeded

jackets, backpacks, etc., should be stowed out of the way (one option is to store them inside the robot carrying cases). The pit should be kept neat and organized. This setup allows repairs to be finished faster, and judges may prefer to see a clean pit.

8.5 Inspection

While the pit is being setup, team members should be bringing any inspection-ready robots to inspection. Robots that are inspection-ready can go through the checklist below (don't forget batteries and nameplates!). Usually, one or two experienced members (ones who know the inspection process) will go to inspection and bring a couple of less experienced members so they can learn about the process and understand what is asked at inspection. In addition, robots will be size checked at inspection, either with a Vex sizing tool, or with an 18" x 18" x 18" box. Below are the questions that members will be asked about their robot at inspection (**list updated as of 06/17/2016).**

For clarification, PTC testing involves testing of the positive temperature coefficient thermistor within VEX 393 motors to ensure that they have not been tampered with to increase performance. The PTC's within motors are there to stop motors if they are running with currents greater than 25% of the maximum torque for the motor. PTC's are a safety precaution and prevent damage to motors. If a PTC is tripped, the motor will "stall out" and be unable to run. If this happens in a match, the best option is usually to stop trying ot run the motors, and wait to let the motors cool down before trying to run them again.

Robot Inspection Checklist

Team Number_____

Size Inspection

☐	Robot fits within starting size restrictions (18" x 18" x 18") does not touch walls or ceiling of the sizing box! *Robot should be measured WITH Team ID # Plates installed.*	R4

Overall Inspection

☐	Team is only competing with ONE robot - they have no spare or replacement robots.	R1
☐	Robot displays colored VEX Team Identification plates on at least (2) opposing sides.	R19
☐	Robot does NOT contain any components which will be intentionally detached on the playing-field.	G11
☐	Robot does NOT contain any components that could entangle or damage the playing-field or other robots.	R3
☐	Robot does NOT contain any sharp edges or corners.	R3
☐	Robot on/off switch is accessible & Microcontroller lights are visible without moving or lifting the robot.	R16

VEX Parts Inspection

☐	ALL Robot components are (or are IDENTICAL to) OFFICIAL VEX Products as sold on VEXrobotics.com (No 3D printed functional parts are allowed)	R5 R6 R7
☐	Robot does not use VEX products not intended for use as a robot component or any VEX packaging.	R5b
☐	ALL Components on the Robot NOT meeting VRC Inspection Criteria are NON-FUNCTIONAL decorations	R7d
☐	Any grease is used only in moderation on components that do not contact the field, objects, or other robots.	R7e
☐	Any non shattering plastic on the robot was cut from a single sheet of 0.070" material not larger than 12"x24".	R7f
☐	Robot has only (1) VEX EDR Microcontroller.	R9
☐	Robot utilizes the VEXnet wireless communication system.	R10
☐	None of the *electronics* are from the VEXplorer, VEXpro, VEX-RCR, VEX IQ, or VEX Robotics by Hexbug.	R10b
☐	Total number of Servos and Motors is not more than twelve (12) without use of pneumatics or ten (10) with use of pneumatics.	R11
☐	Each 2-wire motor is plugged into its own 2-wire port or into a Model 29 motor controller	R11-2a
☐	A motor may only be controlled by a single controller port	R11-2b
☐	Robot uses a maximum of (1) Y-Cable per each 3-wire Motor Port (cannot "Y" off a 2-wire Motor Port)	R12
☐	Robot uses (1) VEX 7.2V (Robot) Power Pack as the primary power source.	R13
☐	If the Robot has a Power Expander, it has a 2nd 7.2V (Robot) Power Pack	R13
☐	Robot uses a maximum of (1) VEX Power Expander	R13b
☐	Robot has a charged 9V Backup Battery connected	R13c
☐	Team only utilize VEX Battery Chargers for charging VEX 7.2V Battery Packs	R13e
☐	Robot is not controlled by more than (2) VEX hand-held transmitters.	R14
☐	NO VEX electrical components have been modified from their original state.	R15a
☐	NO Method of attachment NOT provided by the VEX Design System is used. (Welding, Gluing, etc.)	R15b
☐	Robot uses a maximum of two (2) VEX pneumatic air reservoirs. (Maximum 100 psi per air reservoir)	R18

Field Control Check

☐	Robot successfully completes the "Field Control Check" Procedure. See Inspection Guidelines.	R21
☐	Robot enters Autonomous mode when prompted with no driver control for duration of Autonomous.	R20
☐	The Hand-held Controller(s) ONLY control the robot when robot is in Driver mode.	R20

PTC Verification Testing

Failure to pass this test will result in immediate Event Disqualification	**Pass / Fail:** _____	**Tested By:** _____	R15 R21

Final Inspection Pass / Fail: _____ **Inspector Signature:** _____ **Team Initials:** _____

8.6 Practice Fields

When the team arrives at a competition, the robots will often need calibration. This calibration can be in the form of tuning a PID loop on the robot for the fields and scoring objects at the competition (there are sometimes variances in scoring object size/density), checking that autonomous works on the field and with the objects at the competition on both sides, and even some more practice for the drive team to get them ready for their first match. The best place to do calibration and practice on competition day is at a practice field. In general, there are at least one or two practice fields at a competition. These fields are often very busy, as a lot of teams will want to edit autonomous codes or practice. It is often beneficial to save a spot in line, especially if the queue for the field is very long or a robot greatly needs the calibration. Sending a few members to wait in line while the robot is prepared or going through inspection can be a good approach. However, if the robot is not ready by the time our team members are next in line, they should let other teams behind them go ahead.

8.7 Competition Format (WP's, AP's, and SP's)

The official rulebook for each year describes the year's competition format and rules, and each team member should read and become familiar with the rules. One aspect currently in the rules (it has been for a while) worth noting here is Strength of Schedule Points, or SP. SP are what differentiate robots with identical Win Points (WP), and is equivalent to the number of points the losing alliance has scored (or the tie score in a tie, see the game rulebook for other cases such as disqualification). SP are interesting, because if your alliance is ahead by a significant margin in a match, they can sometimes score for **the opponent,** thereby increasing the SP they earn for a match. **The coach** (occasionally scouts may advise the coach about SP) is usually in charge of determining whether the Drive Team should go for SP in any given match.

And, recently, the VEX Starstruck game added AP's (Autonomous Points). AP's mean that SP's become the **third basis** for ranking teams, and **AP's are the second basis.**
According to the manual:
"
All teams in each Qualifying Match will also receive Autonomous Points (AP).

- Teams who earn the autonomous bonus in a Qualifying Match receive four (4) AP
- Teams who do not earn the autonomous bonus in a Qualifying Match receive zero (0) AP

"

This change makes autonomous significantly more important, and may stick around in future competitions.

8.8 Scouting and Strategy

Scouting at competitions is critical, especially at larger competitions. Scouting refers to sending team members to watch matches and take notes on other robots. In particular, if our team is not performing well enough to be a top 8 (or 12 depending on some competitions) robot, then scouts should be advocating for their group's robot to teams which they believe will work well with, and will be in the top 8. On the flip side, if our robot is likely to be in the top 8, scouts should be

looking for any robot they believe performs very well and will also pair up well with us. Scouts should work with one another to take notes, and eventually to created a ranked list of teams to pick if we are in the top 8, or teams to advocate to if we are below the top 8. This list should be fairly deep, just in case your favorite teams are picked ahead, and it should be given to the group leader so everyone in the group can discuss what they've seen and who the final picks will be before alliance selection (whoever goes up to represent the team at alliance selection should take the ranked list with them, as their top choices may be picked first!) **If anyone doesn't currently have a job at a competition, they can probably be scouting!** Team members in this position should talk with the scouts/lead scout (if one exists) for what they can do to help. Make sure to take notes as discussed below!

On the subject of strategy, scouts also play a critical role. While the Drive Team coach is the strategist during each match, scouts can provide the coach with valuable insight going into a match. Scouts should keep track of when their robot will be in a match, and if they have information on their ally or their opponents that may prove useful they should provide that to the coach. Scouts and the coach can work to change their strategy set choice to combat a particularly difficult or elaborate opponent. Even if the opposing alliance has robots which function better or score faster, we can still win with strong and clear strategy! Building a good robot is only one facet of the competition.

At: http://curriculum.vexrobotics.com/appendices/appendix-4, they suggest the following be noted during scouting:

Robot Functionality

- How well does the robot move?
- Does the robot have the capability to pick up different types of objects?
- How does the robot score the different types of objects?

Robot Behavior

- How many points did they score in each match? (An exact number)
- How many of each type of object did they score?
- What do they do at the start of the match? At the end of the match?
- What is their general strategy?
- When do they like to score?
- What is their general match flow?

We would suggest adding another category to the list:

Robot Autonomous

- Did their autonomous function, and is it reliable?
- What does the robot do during autonomous?
- How many points does the robot score during autonomous?
- Where on the field does the robot travel? (to avoid collisions with our autonomous)
- Do they have multiple autonomous modes? (if viewing multiple matches, or talking with the team)

This detailed list of notes not only helps with strategy when going up against a robot, but also helps when compiling a ranked list of teams to pick or to talk to before alliance selection! Also, if the robot malfunctioned, make sure to make a note of that as well! That may indicate the robot is unreliable, or that they had some form of difficulty which prevented them from playing as well as they can.

To summarize the overall process, each scout should be assigned (likely by the head scout, if one exists) a certain robot in each match to watch. Each scout should take a scouting sheet the team has created for the year (using criteria seen above) and modified for the year of competition to every match they watch and fill it out. They should then bring that list back to the head scout (if one exists, else whoever is in charge of organizing the sheets), who should be compiling all of the data on each robot.

8.9 Preparing a Robot for a Match and Joystick/Robot VexNet Light Troubleshooting

When preparing a robot for a match, it is important to ensure that nothing is forgotten. For this reason, we usually choose to create a pre-match checklist unique to each robot (with some shared features). Below is a sample pre-match checklist:

Item	Checkbox	Name
Battery voltage full (battery boss only) Backup Battery full		
Motors and motor screws tightened, motor collars tightened		
Nameplate: Right color for match		
Autonomous: Correct code loaded		
Bearing flats: In place and pop rivets		
Linear slides: greased and secured		
Ramp: behind rubber screws so does not actuate by accident		
Rubber bands: On wheels, and on ramp		
Motor port wiring: Connected, no pinches.		
Safety glasses, controllers, and controller battery level		
Standoffs: Tightened appropriately so ramp does not bend and conveyor is not crooked		
Ready to have fun! :)		
Final sign-off	All	

The coach, and one of the pit mechanics (and a software engineer for code) should go through this pre-match checklist, preferably at least 4 matches in advance if possible. We usually print out the checklist, and put it behind a clear sheet protector or dry erase pocket, so that we can use a dry erase marker to check off the items. We leave this sheet in a binder, the robot's "competition" binder. **It also helps to bring a "rules" binder to competitions** (a binder with a printed copy of the current rulebook).

Additionally, the coach or a scout should check with our alliance partner for each match to form a strategy, and to understand each others' autonomous! **We need to coordinate autonomous codes so that we do not interfere with one another!!**

Finally, when the Drive Team arrives at the field, the coach should do a last check after placing the robot on the field in the correct autonomous position to ensure that **batteries are plugged in and fastened, no wires are unplugged, and <u>the robot and joysticks are turned on and paired.</u>** It may be useful to bring a sheet like the one found below (credit to VexNet Cortex User Guide) to the field for troubleshooting the blinking lights on the robot or joysticks. This sheet may also prove useful when troubleshooting in the lab as well.

5. Diagnostics Information: refer to the following chart for Joystick and Cortex LED patterns and meanings.

Joystick [5]	Robot	VEXnet	Game	
		Medium (yellow)		Initialize - Looking for PC or Tether Mate
		Blip (yellow)		Startup - Looking for USB Key
		Fast (yellow)		Linking - Searching for VEXnet Mate
		Fast (green)		Linked
		Slow (green / yellow)		Linked - Data quality reduced
		Slow (green / red)		Linked - Poor Data quality reduced
		Solid (green)		Tethered to Mate or PC
		Slow (red) single blink		Fault: Lost Link - Searching for VEXnet Mate
		Slow (green)		Downloading User Code [1]

Note 1: Does not apply to ROBOTC User Code Downloads

Joystick [5]	**Robot** [1]	VEXnet	Game	
	(red)			Main Battery = **Dead** (<5.5v) or CORTEX Off [2]
	(yellow)			Main Battery = **Low** (<6.5v) [2]
	(green)			Main Battery = **Good**
	Solid			All Good: Both Joysticks connected
	Solid + 1 Blink			All Good: Tx1 Joystick connected
	Fast			Autonomous only mode
	Fast (red) [3]			Fault: Low Backup Battery (0v-8v)
	Slow (red)			Fault: User Microprocessor Issue

Note 1: Robot LED only work when Linked
Note 2: Lowest CORTEX battery color latched at Joystick and CORTEX
Note 3: No Backup Battery only indicated if competition cable is connected.

Joystick [5]	Robot	VEXnet	**Game**	
			Off	No Competition connection
			Solid (green)	Driver [4]
			Fast (green)	Autonomous
			Fast (yellow)	Disabled

Note 4 : Game LED Driver Indicator is only used when the competition cable is connected.

Joystick [5]	Robot	VEXnet	Game	
(red)				Joystick Battery = **Dead** (<5.5v)
(yellow)				Joystick Battery = **Low** (<6.5v)
(green)				Joystick Battery = **Good**
Fast				Two Joysticks in use
Solid				One Joystick in use

Note 5 : Joystick LED only on Joystick.

Update Utility Tool Indicators

Joystick [5]	**Robot**	**VEXnet**	**Game**	
		Solid (green)		Tether to PC
	Slow (green)		Flickering (green)	Bootload Mode - Ready to update firmware
	Slow (green)	Slow (red)	Flickering (green)	Downloading Master Code

Other Indicators

Joystick [5]	**Robot**	**VEXnet**	**Game**	
(red)	(red)	(red)	(red)	Flash on all 3 indicates a Reset
		Slow (red) double blink		NO VEXnet Key detected
	Slow (red) double blink			Invaild ID in the CORTEX
Slow (red) double blink				Invaild ID in the Joystick

ROBOTICS DESIGN SYSTEM

8.10 How to Avoid Missing Matches

There should be at least one person on each group, usually the coach, **who has match list with their group matches highlighted so we never miss a match**! In addition, the team captain, and the team mentors should have match lists as well, for all groups. Walkie talkies are very useful here as well, to ensure that each group is queued and to find drivers or coaches before matches. Groups should be informed at least 4 matches in advance of their match if possible, to allow time to gather the drivers and coach and to prepare the robot for a match. The group coach or mentors or the team captain should ensure each group is in the queue two or three matches in advance.

8.11 Throwing Matches

We never throw matches. It is very likely that throwing a match will hurt your partner team, and throwing matches is an unethical, unsportsmanlike, and unfair tactic that goes against the spirit of the entire competition. Our team tries our best and never gives up, no matter what.

8.12 Skills Challenges

The coach, and whoever else keeps track of match schedules should also find good spots in between matches for skills challenges. Good spots typically have a fairly large gap between matches, leaving enough time to wait in line, do the skill challenge, and get back to the pit to prepare for the next match. Additionally, the should make sure this slot does not interfere with lunch time for the Drive Team.

Skills Challenges are very important, provide an opportunity to demonstrate your robots' skills without outside interference, and are usually needed to be a candidate for the Excellence Award.

8.13 Reflection

Throughout the competition, team members should be reflecting upon their performance. Questions to ask include (as noted by http://curriculum.vexrobotics.com/appendices/appendix-4):

- Could we be doing better?
- Do we need more strategies? Different strategies?
- Are we using our scouting data effectively?
- Is the robot working the way it is supposed to?
- Is it too slow? Is it being unreliable?
- Are teams picking up on our strategies? How do we adapt?
- Are our opponents better than us? Why? What are they doing that we are not?

These questions, and responses, should be recorded in the Engineering Notebook for discussion after the competition.

Additionally, if any problems are addressable at the competition, then the team can definitely work to make slight modifications to fix problems. However, this process requires all of the team

to be on the same page, and good communication, so that mechanics at the pit can make repairs, or programmers make changes to the code.

However, **no matter what is going wrong, never give up and don't panic!** Do what you can to fix problems as they come, don't stress about what you can't control, and enjoy the experience!

8.14 Helping Other Teams

If teams around us are struggling with a problem, ask if you can help! You'll broaden your experience with different robots, test your knowledge and experience, and most importantly you'll help a team out!

Questions? Ask your group leader, team captain, or team mentor!

Good luck, and have fun! ☺