Engineering and Design Journal

FTC Team 4278 de.evolution

> Ever since last spring, when their team members did a presentation at our school for Science Discovery Day, they have been an inspiration to our robotics students and are chiefly responsible for encouraging our 8th graders to start an FTC team this year.
>
> –David Warner, Coach for the RSF Eagles FTC Team

> De.evolution team members are committed to advancing understanding of robotics in middle and high schools and to spreading the message of the excitement, team spirit and intellectual stimulation that FIRST competitions provide.
>
> –The Rancho Santa Fe Review

> Historically, our team has been about building the best possible robot. This year, however, the focus of our team has shifted from winning the competition to helping others succeed, as we have realized that this, truly, is what FIRST is about.
>
> The teams at the competitions are wonderful, as always. The cooperative spirit of the events is fundamentally alters one's perspective, as it grows into more than simply a competition for awards.
>
> – Noah Sutton-Smolin, Co-president of de.evolution

Introduction

FTC Team 4278 proposes the creation of a robot to address all aspects of the FTC 2013-14 competition, "Block Party." The proposed design has the capacity to address, in the teleoperated period: (1) retrieval of blocks; (2) placement of blocks into the block crates; (3) raising the FTC flag; (4) hanging on the field bar. In addition, during the autonomous period, the robot will address the following elements: (1) placement of blocks in the correct bin, as indicated by an IR beacon; (2) maneuvering to the ramp.

In order to achieve these goals, it was determined that the robot should: (1) be very difficult to push; (2) be fast and reliable in movement; (3) have a low center of gravity; (4) have as few unique components as possible; (5) be as compact as possible; (6) pick up blocks with at most one motion; (7) place blocks in crates with at most one motion; (8) hang with at most either two motions on an existing component, or one motion on a new component; (9) be able to raise the flag with one component; (10) be electrically stable, generating as little static as possible.

Goals set for autonomous include: (1) being able to score in the crate denoted by the IR beacon; (2) being able to travel to the nearest ramp; (3) being able to address and recover from any problems which may arise on the field, including, but not limited to, obstructions caused by other robots; (4) potentially being able to score both autonomous blocks. For (3), the intent is to travel onto the ramp regardless of potential obstruction by other robots. Decision logic is used to avoid obstructions.

Over the course of this engineering document, we address these problems, determine how we solve them, and explain the process which lead to these conclusions. Most notably, the development of a prototype robot allowed us to identify critical problems with our design and address them before implementation.

The code which goes into this design will also be explained.

FTC Team 4278 also aims to create a foundation for all FIRST teams by: (1) helping teams online and in the pits where possible; (2) promoting the philosophy that the FIRST Tech Challenge is primarily about education and secondarily about competition; (3) raising awareness about FIRST in our high school and region; (4) supporting the success of teams through the distribution of knowledge and acquired experience.

We aim to, first and foremost, help teams succeed - the competition is not genuine if teams are not operating at maximum capacity - and secondarily, spread awareness about FIRST to students everywhere to bring them into the FIRST program.

Contents

1 Robot Design and History Overview

In this section, we detail each component of the robot, the design decisions that went into it, critical tests, and the general development of our design.

We have since changed our robot's design significantly from what follows. We have kept the following sections for two reasons: 1) for the sake of historical design records, and 2) for the sake of clarity. The changes to our robot from this version are detailed in section 1.4.

Over the past four years, we have gone through multiple paradigm shifts in our design process. Our first year, we simply threw ourselves at the problem and left the result to be decided. This worked wonderfully at the time, but in retrospect, much of that was simply sheer luck and is not sustainable in future years. We struck the correct idea, and executed it effectively, but not to the best of our ability.

As a result, the second year, our effectiveness slumped. Though we still won several awards for placing third in local competitions, we did not perform as we had hoped. The reasons for this are twofold: First, we failed to properly conceptualize what it is we intended to do before making an attempt at execution. In this way, the robot's design suffered significantly from problems which could have been solved with some forethought. Secondly, we failed to properly test the components of the design before implementation. We threw parts together without accurately testing each one for validity, and refused to change the parts that did not work.

We drew many lessons from that year, and in our third year, we developed a full SolidWorks model of our design, and created a defined intent for our robot before building. We prototyped several components before implementing them. We did still run into issues, but they were not issues we could necessarily have predicted or foreseen over such a short timescale. All things considered, we created a highly effective robot, breaking the world record score, and scoring a maximum of over 300 points on the field by ourselves.

This year, we had to think a bit more rigorously about our design. We hearkened back to the 2011-2012 FTC Challenge, "Bowled Over!" during which we struggled to actually lift the balls off the ground. We recognized going into this year that pulling objects out of a dispenser is much easier to design and engineer around than lifting objects off the ground. However, we wanted to do it all. When we set out to tackle this challenge, we decided we would find the IR beacon, move to the bridge, score tons of blocks, raise the flag, and hang our robot. We decided that the challenge was *doable*, and we would meet its challenge. This, below, is what we have done...

1.1 General Structure and Design

We decided very early on what the style of our robot's design would be. We recognized that it is possible to determine some components' effectiveness without actually having *designed* the other components necessarily.

We started with the frame structure. We decided there were a couple main approaches: we could either lift blocks straight off the ground, or drive into blocks and lift them up that way. One way supports a high, square frame with a tall clearance, and the other supports an "A-frame," so called for its resemblance to a block-letter A. This would allow us to pick up blocks with the front of our robot, while still leaving plenty of room for electronics. We chose the A-frame over the square frame for sheer simplicity and potentially synergistic design with other components of the robot.

At this time, we had imagined the flag spinning motor on the front of the robot, an arm to manipulate blocks, and various motors and errata on the sides of the robot. While we have significantly changed the vision from its original form, it is still true to these aspects in particular. We have placed a flag spinner on the front of our frame, we have an arm to manipulate blocks, and the gears are stored in the side panels.

As optimal as we may have decided the A-frame to be, there were a few considerations to think carefully about before actual implementation. These were lessons we learned from "Bowled Over!". The first topic we needed to consider was stability. When we designed the frame for the challenge two years ago, we paid very little attention to stability. As a result, the weight we placed on the center of the robot caused it to cave in where it should not have. The front wheels warped inward, and the electronics came loose under tension. Additionally, the muli-segment arm had pressure in places it was not designed to support. We will

reveal how we addressed this later in this section.

1.2 Considerations for Support, Structure, and Placement

We needed to make particular consideration for structure and weight distribution on the final design, as we had chosen a type of frame which becomes unstable if not properly constructed. We decided very early on that we would construct the robot mostly from raw materials, so the first consideration was of material. We were presented with several options, nobody on the team questioned the choice of Delrin as the appropriate material.

Material Selection Delrin was chosen primarily for its strength and resilience. It is an easy material to lathe and cut, both by laser and hand, and it does not damage easily. It is very strong in thick directions, and is also relatively lightweight. Tetrix aluminum frame pieces are far, far too flexible and pliable, and become warped or damaged easily during competition. Delrin does not often even so much as scratch.

Even so, we could not depend on Delrin entirely. After some consideration to structure, we decided that the easiest way to prevent the side panels from caving in was to both make them thicker and support them against each other. We decided that the wheels and gears could go in between two panels, which would be spaced for stability. These would act as our gear boxes. Additionally, the two gearbox panels would be supported by the entire middle section of the robot, which would prevent them from caving in. We decided support bars would be needed across the center of the robot. Thick aluminum standoffs were used to support the two panels of the gearboxes.

Center of Gravity Some consideration had to be given to center of gravity. In deciding we wanted to use an arm, we effectively filled most of the robot with air. This made the center of gravity incredibly critical. It needed to be very low and centered, so that we could not possibly be pushed over by another robot. This meant that the motors powering the arm and wheelbase could not be placed adjacent to their respective components.

To demonstrate this, consider the placement of the motors. Place the front drive motors in the front of the robot, the arm motors up by the arm, and suddenly the weight distribution becomes horribly imbalanced. As a result, we decided to place the motors directly next to the electronics in front of the robot. The center of gravity is thus centered along the width of our robot, and sufficiently low to enable difficult tipping. We have encountered a slight problem, where the center of gravity is too far forward, but this can either be corrected or may not be an issue. We have yet to see in competition, but preliminary, informal tests indicate this may not be an issue.

Gear Strength We have learned from past years that Tetrix axles are insufficient for high load. This is a result of two parts: first, their circular design makes them hard to lock into anything. Second, the locking pin/set screw, under high tension, actually cuts into the axle, both preventing movement of the attached part and preventing removal. The axles also rotate under moderate torque, as they are made of aluminum.

In order to alleviate this problem, we are using allowed hexagonal axles. These have several benefits. First, they are much thicker, and will not bend or twist. Second, objects that are locked into a hexagonal axle cannot rotate freely around the axle. Third, the hexagonal collars do not lock against the axle themselves, and as a result do not create an indentation in the axle past which nothing will move.

Arm Support We decided the arm needed to be strong enough to lift the robot. During the construction of the first prototype of the arm, we created a way for the arm to support the entire robot, but miscalculated the shearing strength of the wooden collars which locked the arm in place. As a result, the torque on the arm caused by gravity caused the wooden collars to strip internally. We have since replaced them with custom-built clamps, designed after hex wheel hubs. More on arm design to come.

1.3 Wheelbase, Arm Design, and a small word about Flag Spinner

Wheel selection In selecting the wheels for our robot this year, we wanted to prioritize two things: maneuverability and traction. Wheel layout and selection is the entire deciding component in how these two motivations are balanced, and our wheel selection reflects this. We decided that our design model - block grabbing in front, dumping in back - necessitated finite control while aligning the back, but did not require such control in the front. We decided that the front needed to be able to turn quickly, and the back needed to turn precisely. These requirements fit the model of a "fish tail" drive perfectly.

A fish tail drive consists of two traction wheels and two omni wheels, mirrored horizontally across the robot. The traction wheels we used are called Colson wheels, and the omni wheels are standard Tetrix parts. We capitalized on the new lenience in wheel selection this year by selecting a wheel which is both smooth, large, and high friction. As a result, it is nearly impossible for another robot to push us sideways, and only possible to push us along the direction of rotation with compliance from the motors.

Arm History: Roller Our original intent with the design, as can be seen in the "Solidworks and CAD Modeling" section, was to include a roller in front which would bring blocks into the robot. We had originally settled on this type of design as it gave us the ability to draw blocks in from anywhere on the floor. However, this component had serious problems. To list a few:

- We could not pick up blocks at the corners
- It became very difficult to control the exact number of blocks we drew in
- It took up an egregious amount of space in the front of the robot
- It often failed to pick up blocks entirely
- It required significant machining to maintain and modify, as it is an entirely custom part that requires heavy maintenance
- It often picked up more than four blocks; there was no way to tell how many blocks it had, however, since the robot was visually in the way

These items forced us to reconsider this design decision. Upon reconsideration, we realized that we would rarely, if ever, have a need to pick blocks up off the ground, and that it was probably wiser to pick up blocks against a wall. After all, every one of the blocks starts against or adjacent to a wall.

Arm Block Scoop The block scoop came with two simple revisions. During the first revision, we intended to create a small ramp, which the blocks would fall over when we hit them. This worked well, but did not work entirely as we intended. We noted a couple difficult problems, which would cause delays in the way we work in teleop. The first was that we had a chance to grab more than five blocks, and the second was that it required significant force to actually climb over the ramp.

To solve these problems, we took two actions: First, we replaced the solid Lexan panel with a thin sheet of plastic, which was thin enough to consistently slip underneath the blocks while also being able to support blocks. The second change was to add a metal bar to the inside of the arm, which prevents us from grabbing five blocks. With these two changes, we now consistently grab four blocks, even when there are very few blocks remaining at the edge of the field.

Robot length While one might normally consider this to be a small or trivial detail in robot design, this actually grew to high importance for us in our design deliberation. There are a couple reasons for this: first, fish tail drive is significantly impacted by the span of the wheels. Too short, and the robot does not turn well. Too long, and the front turns too quickly.

We have additional constraints on our length as well: we need to 1) be able to support a flag spinner in front of the robot, which cuts a couple inches off, and 2) we need to be able to pick up blocks with the

front of our robot. Our initial design for a roller necessitated a much larger margin in the front of the robot, however, as that has changed, the length of the robot changed with it.

Flag spinner This deserves an honorable mention. We used the dead space on the back panel to add a wheel with two prongs. We originally wanted to use a rubber panel and friction to turn the flag spinner, and our first tests told us this would work. However, in practice, it was too difficult to align, and was replaced with two prongs.

The rotational motor is geared 1:3. This is satisfactory, and provides a lift time of 4.5 seconds, which is good enough for our purposes.

1.4 Changes in Version 3

We decided after our last regional competition to make significant changes to the robot. We wanted to pick up blocks more consistently, improve the accuracy of autonomous, allow for a double hanging system, and increase the drive train's speed.

1.4.1 Drive train speed

To increase the drive train's speed, it was necessary to do two things: primarily, add another motor onto each side of the drive train; secondarily, decrease the overall weight of our robot. The first was a rather simple task to accomplish; simply place the motors in a triangle and gear them together. The gear ratios have been changed slightly, and as a result, we move across the field in approximately 2.75 seconds. Speed, previously a significant limiting factor, is no longer one for us.

1.4.2 Block pickup accuracy

As a result of our drive train changes, we were allowed the option to reposition the center of gravity of the robot. We have identified over the past few competitions a simple issue: our weight has mostly been at the back of the robot, away from the block scoop, so when we accelerate quickly the scoop kicks up slightly. This stops it from effectively picking up blocks. By changing the center of gravity to the approximate center of the robot, we have entirely stopped this problem from occurring.

1.4.3 Double hanging

One of the main benefits to our mechanism is that we do not have a dedicated mechancial process for hanging; instead, we simply use the torque of the arm to carry our robot. However, there is a significant disadvantage to this: our hooks are in a fixed position, separated by the width of the arm. This has made it incredibly difficult (and in most cases impossible) for us to effectively double hang.

We recognize that this is an important aspect of the game, and that, in order to become a more competitive robot, we should accommodate other roobts in this respect.

Instead of changing our arm, we decided to add an identical bar on *our* robot to the one which is used for hanging. This means that another team could hang on our arm the exact same way they might normally hang from the regular bar. They then pull themselves up on us, we pull ourselves off the ground, and both robots are hanging.

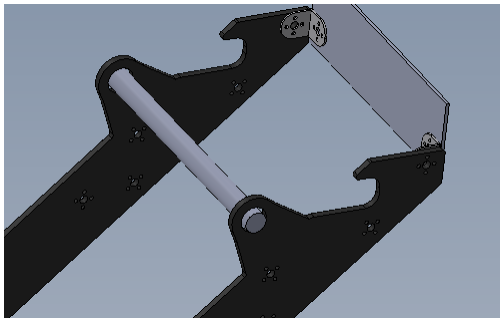


Figure 1: Double hanging bar - this is vertical when we are prepared to hang, so other robots may hang on us

1.4.4 Gearboxes! The Power Takeoff mechanism

In order to facilitate the additional increase in drive power we wished, we would have needed nine motors: six on drive, two on the arm, and one for the flag. To solve this problem, we decided to use a power takeoff mechanism, which allows us to transfer motor power between the flag spinner and the arm movement. This means that the flag spinner gets the additional power of an extra motor, while we conserve a motor.

In order to achieve this, we moved the gearboxes from inside the arm panels to the center of our robot. This, additionally, allowed us to increase the centering of our robot's mass. The power takeoff mechanism relies on a special gear called a "dog" and a servo shifter. The mechanism can be seen here:

The highlighted gear below is the dog gear:

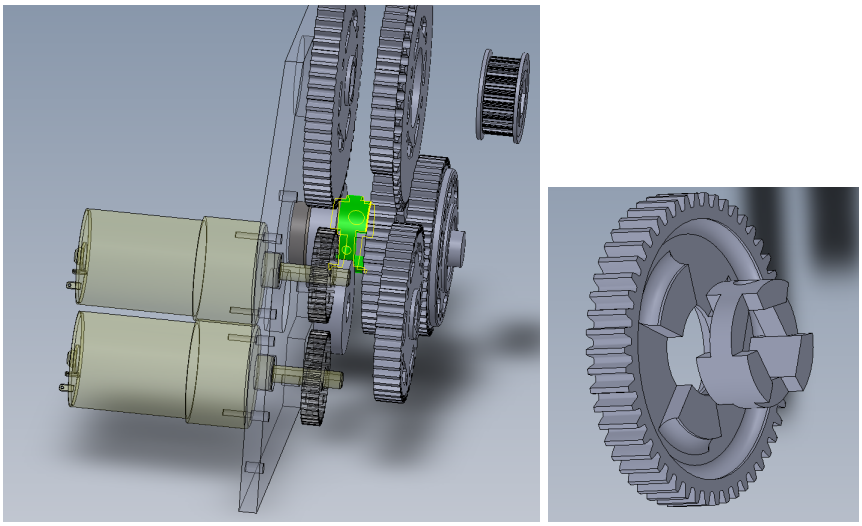


Figure 2: Left: dog gear in the neutral position; right: demonstration of how the dog gear locks

This gear turns with the axle. The other two gears are on bearings, so they spin freely around the axle - however, when the dog gear is locked into one of the two gears, they rotate simultaneously. The dog gear has the capacity to shift between the two gears, causing power transfer to change from one gear to the other - i.e. from the flag spinner to the arm and back. This happens automatically in code when necessary. The driver does not need to think about switching; power will be directed where the driver indicates it should.

2 Meetings and Proceedings

In this section, we detail exactly what happened at select meetings, whose events bear particular significance over the development of our robot.

2.1 Preseason Overview – Meeting 1: 2013-08-31

We held a preseason meeting in order to go over scheduling, recruitment of new members, and hold a quick review of what we learned from our last season. One of the main items we discussed was our scheduling. We decided we may need to cut back on scoring the maximum number of points and instead focus on scoring a high, yet consistent number of points. We found out that we need to secure our electronics and work diligently to rectify issues we have previously had with the field control system. We also plan on placing a much larger emphasis on the CAD design of our robot than we have in our two previous years as it is an efficient way to quickly discover and troubleshoot problems prior to building the system.

Our final goal is to have two weeks of drive practice before our first regional on December 14th, important for both driver and coaches, to figure out the timing of the game, as well as things that we can or cannot do in game. Organization of the team this year will be facilitated through the use of Google Groups, which will allow all the team members and their parents to be easily contacted for meetings, and will hopefully foster some discussion over build design or game strategy.

We also decided to meet a few hours after the game was released next week so team members could think about strategies ahead of time and add to the strategy discussion of our first meeting of the season. The meeting was fairly short, but got the team in the right mindset for the upcoming season, and got everyone excited for the new season!

2.2 Initial Design – Meeting 2: 2013-09-07

The team decided last week to meet a few hours after the game video and rules were released, so by the time our meeting started, everyone had an idea of how they thought the robot should work, and be built. Fortunately, most team members were on the same page and wanted to focus on scoring as many blocks into the baskets as possible, in a manner that would allow for a integration in the lifting and scoring mechanism. We decided to focus on every aspect of the game for our qualifiers, since it seems like we could consolidate all of the mechanisms into very few.

After a fairly long strategy discussion, some ideas were thrown around as to how to pick up and score the blocks, since we always have a difficult time getting our game pieces, and a rough idea of a mechanism was developed that would use a roller system and a block hopper that would pick up and score the blocks. There is still significant discussion considering the way we will go about constructing a lifting mechanism, with different arguments for and against a rotating arm and a more standard but perhaps less efficient forklift, similar to "Ring It Up!". Some of our team members have some experience with constructing forklifts from prior FTC seasons, so we have some idea of what kind of issues we might come across with that kind of design. One thing we discussed was making sure we construct and purchase our materials with a little more regard for quality than we have in previous years, but we wanted to focus on simplicity this season as it worked out very well for us three years ago, and the ideas we are thinking of currently all seem to fit this idea.

Members of the team have assumed different jobs to complete before the next meeting, such as researching lifting mechanisms, getting a BOM for the field and purchasing the materials, and researching the specifications of an IR beacon and sensor, since the team also decided that scoring the block during autonomous is absolutely imperative to the outcome of this game. It is essentially free points that even a defensive robot cannot stop. As of right now we are choosing to focus the endgame period, since lifting and raising the flag are relatively simple tasks.

We are considering purchasing the AndyMark field, as it would provide us a standardized field and a better replication of the interaction of the robot with the field during competition. We have previously used the PVC field walls, which may not be as effective in the upcoming competitions.

2.3 General Design – Meeting 3: 2013-09-14

Today was our second main meeting; we discussed several different raising mechanisms, since we as a team have decided that it was the most important system of our robot.

Our main idea for lifting is an arm that is centered around the top corner. We are designing it to be just long enough to allow us to climb the bar as well as to clear the front block scooper. As for the material, we were looking at Delrin, and wood for our prototype. Delrin has a particularly high tensile strength. If a material is homogeneous then the tensile and flexural strengths are identical. However, most materials have defects in them which act to concentrate the stresses locally, which in turn cause a localized weakness. We have determined that the flexural strength is greater in the Delrin over the wood as follows: We first see that method for calculating flexural strength is:

$$\sigma = \frac{3FL}{2bd^2}$$

and we are looking to see which sheet can take in the maximum force, F. It directly follows that as we increase b, the width measured in in, and d, the thickness measured in in, the sheet can withstand a greater force. Elementary analysis allows us to see that the Delrin is superior than the wood. This will assure us that if our wood prototype works, so should our final design. It will also be able to withstand in competition.

For the flag spinner, we discussed using some external pins or some type of rubber-esque substance to push against the edge and raise it up. We believe that a rubber-type material will work. We plan on using 3 NXT motors to provide enough torque and speed to raise the flag. Our goal is to raise the flag in under 5 seconds.

We also spent a lot of time discussing the importance of autonomous and the structure of our code. We are looking at dynamic autonomous programs that are able to dodge other robots if need be. We are looking forward to prototyping within the next several weeks. Our meetings have mainly comprised of theorizing about code and 3D modeling our robot. We are sharing with the younger students much of the knowledge we have learned over the years as well.

2.4 Initial Field Build – Meeting 4: 2013-09-21

Today we began building the field, discussed the lifting, got mail, worked on autonomous theory, cut PVC with safety glasses, and had some fun moving our very mobile robot for a while. In the mail we received one half of the blocks necessary for the field. For building we:

1. Painted the wood
2. Cut all the pvc pipes for the flag spinner
3. Wore safety glasses
4. Cleaned up a lot of pvc dust
5. Screwed temporary screws to the flanges onto the wood

Flag Spinner:

- Involves NXT motors
- Gearing 6:1
- Rubber-esque material did not work out
- Using extruding pins

2.5 First Prototype – Meeting 5: 2013-09-28

Today was a day for laying out drawings, finalizing designs, and testing some of the flag spinner prototypes.

We started the day with the usual bit of socialization, before returning to work. The primary topic of discussion is the acquisition mechanism for the blocks. It has been speculated up until this point that we are going to be using a roller, and it looks like those plans may come to fruition. We received several items today, including some Tetrix motors, as well as M3 screws and the other half of the blocks.

A to-scale side view drawing shows that the arm we want to design and guiding channels will take up about 7 inches on either side, leaving 8 inches in the middle to mount the block grabbing mechanism on. This fits within our expectations.

It is worth noting that the since we are designing the robot out of raw materials we are using metric, as opposed to the imperial u-channels, so there is some difficulty in lining up the holes if we plan to use any Tetrix. This is a prototype however, and the arm we plan to design for the final design would be out of Delrin, and can be designed to fit our needs. The prototype wooden arm also has potential tolerance issues. There is about a 1 block tolerance on the size of the spinning blocks. Stability is key.

We constructed our first prototype, we assembled the outside arms from plywood. From left to right, there are two arm chassis pieces, the arm pieces and then two arm chassis pieces. The frame design is an A-frame and it works pretty well.

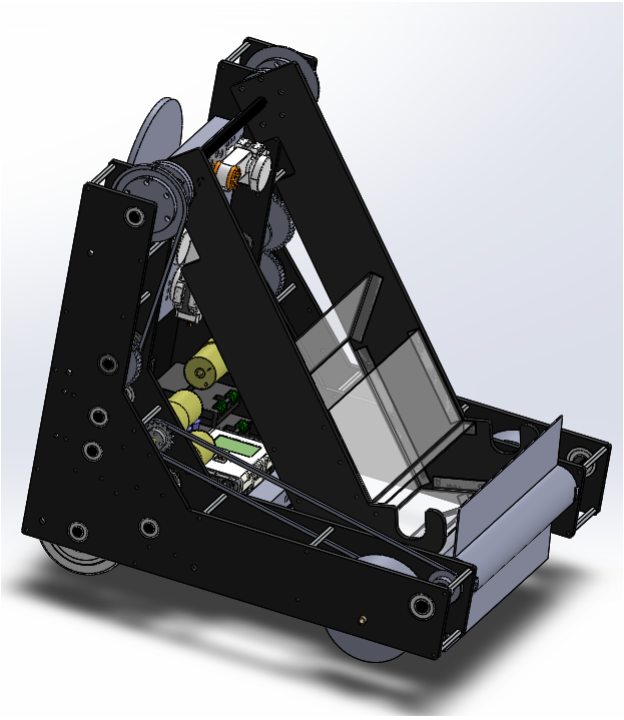


Figure 3: This is the first revision of our Solidworks. We can see the block acquisition device that is essentially a roller that will inhale blocks into our robot

Force is not an issue, and remains at a 16:1 ratio with torque gained. It is important to note that the force DOES double when an additional layer is added, however, with the 16:1 ratio between the movement of the bars, the speed with which the bars are raised ALSO doubles. This means that, though the force doubles, the speed at which the arm go up is also doubled. That implies that any torque issue created by adding another layer to the mechanism can be resolved by multiplying the gear ratio of the chain motor by two. The same vertical speed will be achieved.

The prototype is very functional. We connected it to the (makeshift) motor, and ran it. The mechanism lifted very well. The following problems were noted:

- Block pickup does not work
- Lifting does not lock after power is lost
- The wooden shaft collars strip
- It does not move fast enough
- It makes the robot very back heavy

Possible solutions to these problems (in order) are:

- Switch from the roller
- Jam the gears
- Change the wood to metal
- Add steel to the front

Despite these issues, the lifting mechanism works well, and glides smoothly as long as they aren't stripped. Granted it is currently in prototype form, the final version (which will be produced much more carefully), should be very effective.

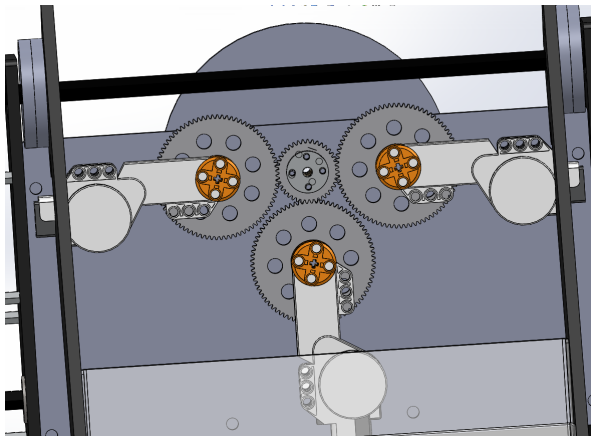
We are currently looking into redesigning the block pickup mechanism as it does not seem to be effective. We plan on lowering the roller to the ground and seeing what happens.

Currently, our projected maximum height is 32, just over what we need to hang on the bar.

2.6 Flag Spinner Initial Test – Meeting 6: 2013-10-6

Today's meeting was short - it was only two hours. We had a couple ideas we wanted to test, and the tests were successful. We verified our idea would work, and redrew the pictures from two days ago.

The idea we were testing today was whether or not we could raise the flag by using a rubber-like substance to mesh within it and spin. We soon saw that the NXT motors were not as powerful as we would like and the force that the rubber plate was inducing was off center. This makes it unfeasible to use.



We plan on switching to the seemingly common extruded pins design. We plan on having two pins attached off center that are 180° apart. We also plan on switching to a Tetrix motor with a 6 : 1 gearing ratio. With this ratio, we will be able to raise the flag, theoretically, in under 10 seconds.

However, this method adds an extra 1.5″ to our length. As we must stay within the 18″ length constraint, we may have to shorten our roller (block pickup mechanism). This may become problematic, but as far as we can see we should be able to work around it.

For the pins, we had two ideas: the first is to use cut Tetrix axles. The second is to use longer Tetrix screws. We plan on using the Tetrix screws first to test and we may end up keeping them on afterwards.

We also did some more testing with the roller. We experimented with lowering it and shrinking the radius of the aluminum tube. We chose a smaller inner radius due to the fact that it would weigh less and it would be easier to place near the ground. We are still running into issues of picking up much more than 4 blocks at a time.

We decided on mounting it 1.5" above the ground because this is just enough to let the blocks slide under as we roll.

It is the popular idea that we want to completely redesign the block grabbing mechanism due to its inability to give us the accuracy and visibility that we want. Visibility is an issue when we go to the opposite side of the field.

2.7 Block Acquisition Redesign – Meeting 7: 2013-10-11

We have spent a lot of time working on scoring blocks, but we overlooked the difficulty of acquiring them. We decided to spend the day looking over our acquisition device as we found it was not very effective. We are looking into changing the roller into a much lower roller made of rubber. We are also looking into changing the entire design altogether. The main ideas revolve around driving into blocks and then flipping them into our arm. However, many members of the team do not like the idea of multiple moving parts.

We decided to extend the final aspect of the roller and found that it was completely ineffective. There is currently no member that continuously supports this idea.

Today, Garrison proposed that we use a bulldozer like system. A flat plate that we would use to ram into the blocks, which essentially extends our arm to be outside of the frame. We built a first prototype with this and it seems to show some promise. We are still unsure of the implications that this has for the a final working idea. It may also be easy for blocks to slip outside of our robot.

We currently plan to go home and think about better ideas. We will probably have an in-depth Skype conversation over this idea. We look forward to improving this design as it needs a lot of work.

The flag spinner was also entirely redesigned as follows:

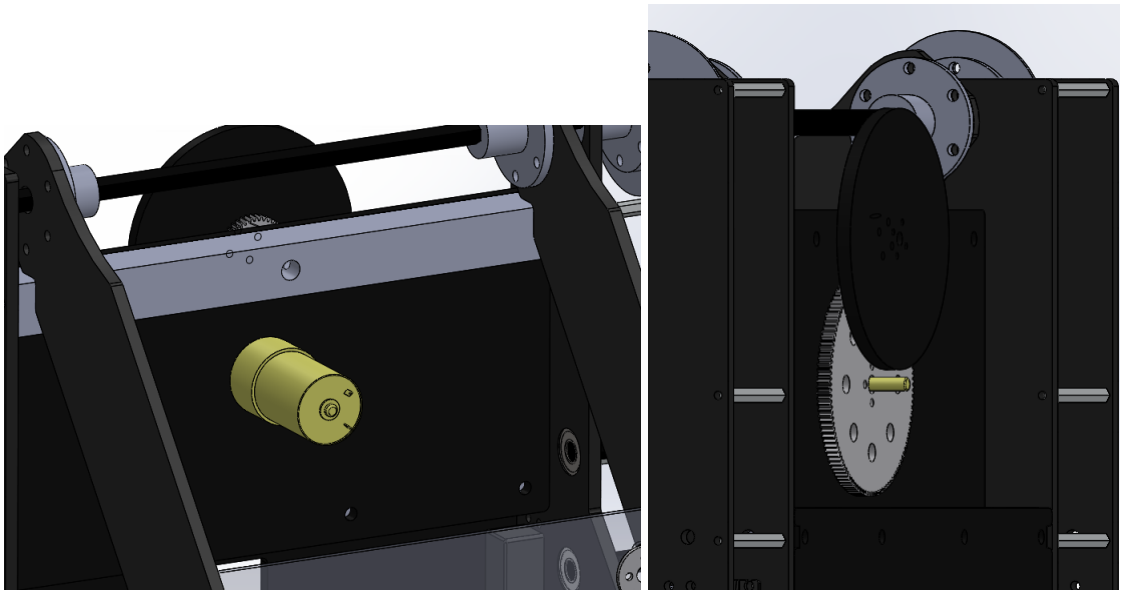


Figure 4: Here we can see both sides of the new, redesigned, flag spinner. The notable changes include the switch to the Tetrix motor and the removal of the rubber backing in turn for extruding pins (not shown above).

2.8 SolidWorks Redesign – Meeting 8: 2013-10-18

After coming back with new ideas, we spent the first hour of our meeting discussing the pros and cons of our new block acquisition mechanism. We all feel that a flat sheet on the ground that we can use to run into the blocks as a bulldozer is the optimal solution.

We are designing it to be fully flush with the ground. We look forward to testing this out. In SolidWorks we are redesigning the base of the robot to be shorter to allow the bulldozer system to lie in front.

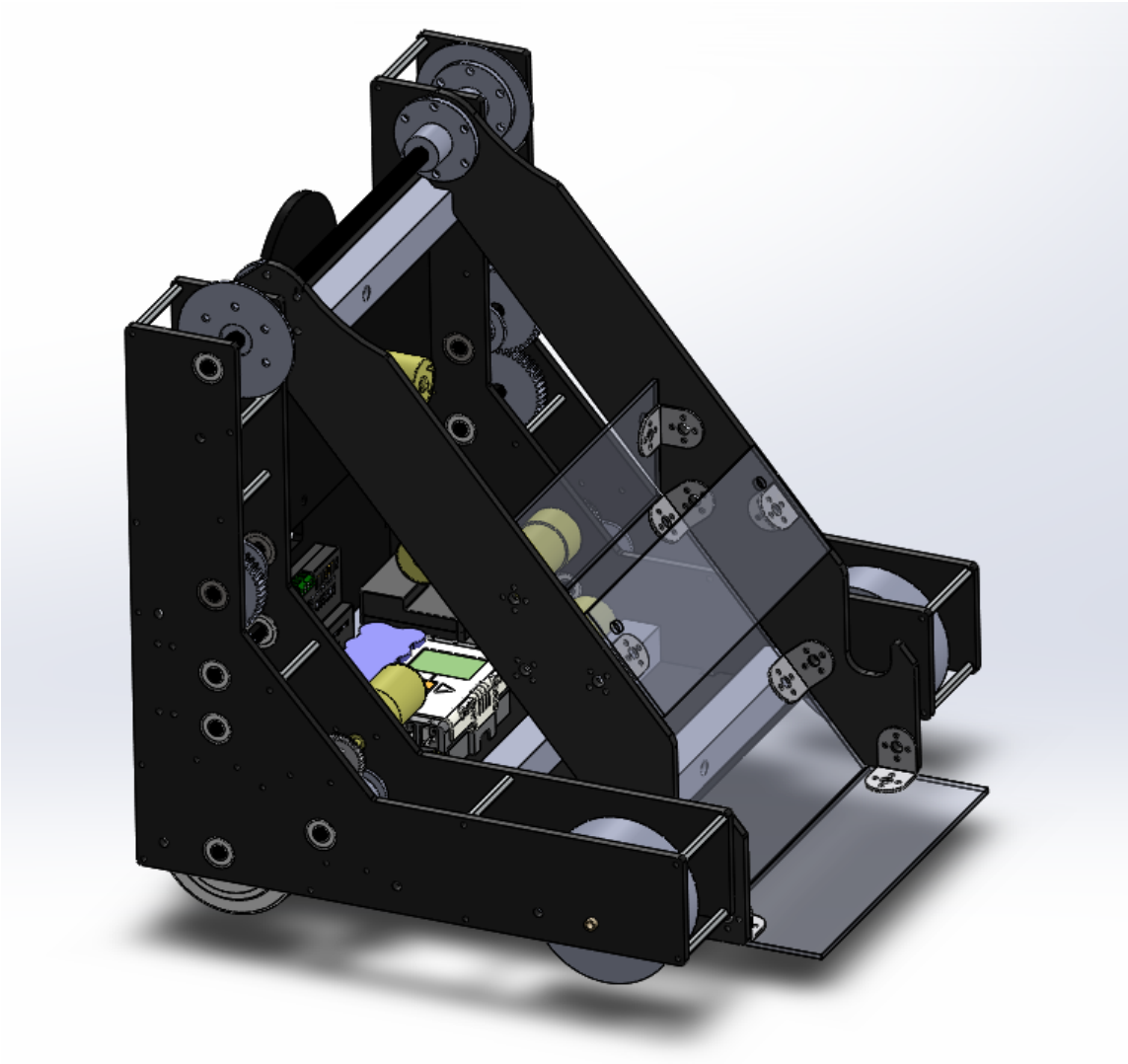


Figure 5: This is the second revision of our SolidWorks which includes the redesigned block acquisition device.

2.9 First Full Test and Match – Meeting 9: 2013-10-25

Today was our first day with a complete robot. Although we had done several tests of our prototype before, this was the first day we had everything working together. The excitement within our team was immense. We tested our drive train with the added weight of the arm on the motor. We tied the flag mechanism with nylon as our Kevlar rope has not yet been received. The Kevlar provides less friction and is more like the competition.

Near the bottom, we have a lot of empty space to place the electronics. We hope to use the HiTechnic SuperPro this year to do much more sensor fusion. We want to add an Arduino or Atmel processor (ATUC3b0128) to do much more powerful computation without having to deal with the timing issues that are imposed by using the task system in RobotC and the slow clock of the NXT. We have plans to include gyroscopes, accelerometers and ultrasonic sensors. We are also looking into developing our own infrared seeker, although we are still uncertain about the feasibility of that task.

With the A-frame design that we have we are able to distribute all of our weight to the back of the robot. This makes it very stable and very difficult to push us.

When we scored our first block, the entire team was ecstatic. We all felt relief that we were able to lift something and score our blocks. We were somewhat surprised as to how simple it was for us to score a block - simply by driving backwards and raising the arm.

The arm went up very smoothly and we had no issues scoring on any of the blocks. We decided to play our first match to see how many points we were able to score.

We gave Tristan the controller and on our first run we were able to score about 200 points. We scored only on the outside baskets. This added up to the balanced bonus bonuses. Overall, we are very happy with the current design of the robot. Although, we do feel that a lot of driver practice is needed, along with an improved block acquisition device.

Once we had the entire robot built we started testing the arm mechanism and its ability to remove and score blocks. We found that it requires almost no effort to manipulate, and that it has the force required to easily push under blocks.

Tristan, our driver, says that with the new “fishtail” drive train that we incorporated along with the ease of use of scoring blocks and removing them from the ground. We are still unable to consistently pickup four blocks.

Now there is a lot of talk among the team members about autonomous and strategies for approaching the autonomous program. We have the ability to determine which column the Infrared beacon is one within one second of our robot starting up.

After we scored our first block and played our first game we decided that we had accomplished a decent amount for this extra Sunday meeting and decided to call it a day. We have high hopes for the competition and expect to place well at our first qualifier.

2.10 Short Design Meeting – Meeting 10: 2013-11-2

Today was mostly a day for driver practice and minor repairs. We did several hours of practice, then moved the grabbing mechanism down an inch because it was barely outside the sizing box. This created a couple functionality issues, which will be resolved in the next meeting. We also tested the autonomous accelerometer and gyroscope, but didn't get much farther than measuring the (mostly insignificant) error before we had to leave.

Today was short.

2.11 Tweaking and Autonomous Programming – Meeting 11: 2013-11-8

Today several mechanical changes were made. The flat block grabber was moved down more, but the attached aligning bars were kept in the same place. The bars were also melted into a curve to stop the blocks from catching on them, but at the same time keeping them from falling out. In addition, we made our hex arm shafts, designed after the AndyMark hex wheel shaft. We used a lathe to create the same shape and then used a hex broach of 3/8" to allow it to slide over.

Another notable change was somewhat small, but it was entirely necessary. We added a funnel system onto our arm to stop blocks from flying out as we tried to score. This was done with two small bent Tetrix pieces that redirect the blocks into the center of our scoring mechanism.

After implementing this, the amount of blocks that we missed dropped from almost 50% to nearly 2%. This worked phenomenally and outdid all of our expectations.

Today was a rather complex code day. Yousuf and Kian woke up early to start work at 7 AM on autonomous. Other team members were arriving past 10. Our progress was significant and extensive, leading us into a few minor problems but also into several solutions.

We decided immediately to write the code again from the bottom up, testing each component as we added it. The old code would have taken too long to get working properly. We struggled for around half an hour with a few simple glitches. We had a very silly error where RobotC simply refused to function properly. The eventual solution was ridiculous: the line of code with the error was replaced via copy and paste with a duplicate line from an old code file. This resolved the issue.

Accelerometric integration now works, and we can properly integrate our position with very little noise and error. There are no issues here.

The gyroscope integration, on the other hand, is returning garbage output. It's moderately representative of our actual rotation, but in actuality it is totally useless. The tolerance is +/- 10 degrees. I surmise this is a tasking issue, and δt is not being set properly, resulting in error.

We attached the infrared seeker sensors, and can now start the game consistently knowing which column the tag is in. We have a 540 degree field of view, since the two sensors face opposite directions. As such, our first movement command is to go directly to the column with the IR beacon. We may need to redesign our autonomous though since we are getting a lot of noise.

Our plan is to turn perpendicular to the pendulum. Everything except the gyroscope is functional.

2.12 Autonomous Working (No IR) – Meeting 12: 2013-11-10

This weekend, we successfully programmed our autonomous. The code allows the robot to deposit the autonomous block in any basket, as selected by the user interface. The logic for this decision works as follows:

- The user selects which row they would like the block placed in, if the IR beacon is in each column.
- The robot uses triangulation with two outward-facing IR seekers to start the game immediately knowing which column contains the beacon. (This still doesn't work yet)
- The robot angles to the correct basket.
- The robot moves and rotates, aligning itself to score with the block.
- The robot raises the arm, scoring the block.
- It then backs off to move onto the ramp.

The program for this has been written, and works with an (approximated) 90% consistency.

2.13 Driver Practice and Lifting Locking – Meeting 13: 2013-11-17

The plan for the meeting was to spend an hour or two testing the autonomous for consistency at getting on top of the ramp. Then the plan was to do driver practice for the rest of the day.

Early tests of the autonomous showed some problems getting on top of the board, with the placement of the sensor being off about 30% of the time. The most common issue occurred when the IR mounting was off center. This caused us major error. After mounting the beacon much more securely, the autonomous became very consistent at getting to the pendulum, scoring the vast majority of the time in several trials.

As we began driver practice, however, some structural problems were discovered. The cross bars that we were using to brace our lift torqued during one run, causing us concern when lifting. Tristan volunteered to create the crossbars in SolidWorks and spent some time taking careful measurements of the spacing between the two halves of the lift. The other major fix that was incorporated was to add two points of contact on each side in order to stop the arm from torquing under major force.

We also developed a lifting locking mechanism which works by jamming a third gear into our system. This induces a lock that cannot be broken and does not need power. Figure 7 shows this system in action.

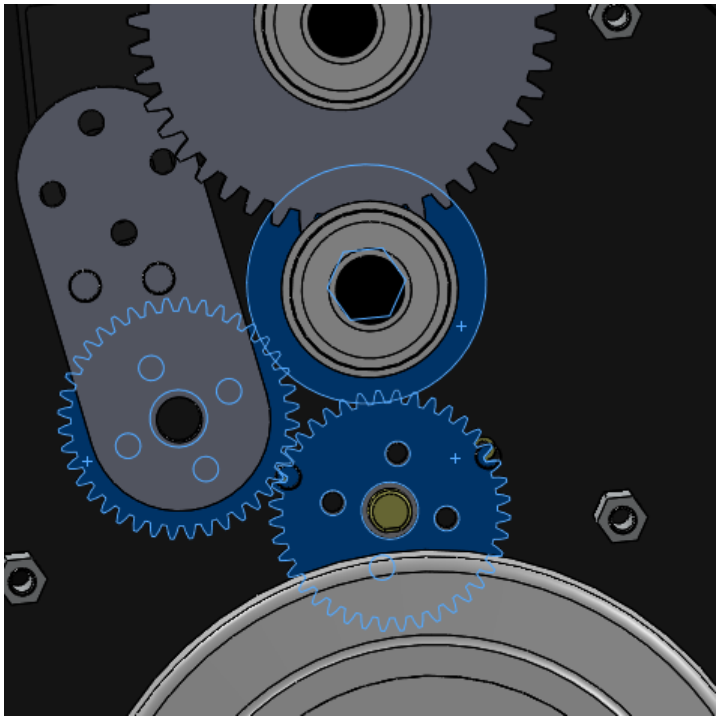


Figure 6: The blue sections highlighted show the three gears that are all locked together. It restricts any movement and works effectively.

2.14 Outreach – Meeting 14: 2013-11-29

This year we decided to mentor an FLL team and we mentored two FLL teams. We helped them program the new EV3 and taught them some new design techniques for the attachments that they had on their robot. We spent about 4 hours running through all of the programs they had and suggesting different ways that they could go about it. We also taught them how to use sensors as a lot of their programs were unreliable. We hope to start an FTC team with these students in the upcoming years. The FLL team we mentored just won their qualifier competition and has a regional competition at LegoLand on December 7th.

2.15 Driver Practice – Meeting 15: 2013-12-6

Today we mainly tested our robots abilities using time trials. Our robot managed to pick up two to three blocks per trip, which was below our expectations. Because of this we began experimenting with different methods of collecting blocks, such as a horizontal roller. The first material being a sheet of extremely thin sheet metal which was low enough to the ground but it was too weak, When we tried to pick up blocks the metal would buckle and make a wall impossible for anymore blocks to get up. Our next solution was to use a thicker, flexible sheet of plastic that slid with really low friction across the ground. The plastic sheet was just the right height to where it would fit perfectly under all of the blocks without getting deformed.

Our next task was to be able to fit three or four blocks - no more, no less. The way we solved this greatly important problem was by using a metal deterrent to block more than five blocks from being collected. The combination of the ramp and the deterrent made it so that we extremely rarely picked up two blocks but we also very rarely picked up more than four blocks.

Overall, we are currently scoring approximately 300 points alone per round. We hope that our speed will be unmatched and we will be able to score many points.

2.16 After Competition Meeting and Revisions – Meeting 16: 2014-01-06

We took a break after the first competition to relax from robotics and think about how we did at our competition. We did very well being the winning team alliance and the Inspire Award Winners at the 2nd San Diego Qualifier. Once we met again we decided to fix the following issues:

- Consistently pick up 4 blocks
- Make raising our robot much quicker
- Change the position of the autonomous block
- Fix autonomous

We decided to work on moving the IR sensors to the side of the robot to make the control much more effective. To fix our consistency issues we widened the arm making it almost always pick up 4 blocks if we go on straight on. We also made it that we cannot pick up 5 or more blocks by adding an aluminum ℓ bar to prevent multiple rows of blocks.

We redid the logic of the autonomous program and it is now far more consistent.

2.17 Pacific Ridge Qualifier Reflection

The day didn't start off the best for our team. We started off the day with a scoop that wasn't exactly in the greatest condition, and our autonomous program wasnt running optimally. By the time our first match came around, we started autonomous without a block scored, then were stuck on the walls and mangled two motors, leaving us with disabled robot. Needless to say, we lost that round. We needed to detach the side plates of our robot to remove and replace the two burnt motors. Thankfully, the kind teams we were competing with let us borrow motors. We have since given them replacement motors, and thanked them profusely for their generosity.

Our next match went slightly better, but we managed to completely break our scoop. We could not fix it at the competition, as we had not brought spare plastic. Instead, we made an on the fly decision to cover our block scoop in duct tape. This temporary fix wasnt very effective, but it worked, we won our next game but we were still in close to last. During our next game, we somehow managed to memory fault an NXT, destroying it completely. We were in around eight place during alliance selection. First picked second, then third picked us. In the elimination round of the qualifier we lost our first game but then won the next two. After that, we won our first game and lost our second, and when it finally came to our last match, we pulled the victory by quite a large gap.

We are looking to improve the consistency of our design so that similar errors do not occur in future competitions. The quantity of parts we had to borrow from other teams is simply inexcusable, and it was only the bounds of their generosity supporting our team. We are forever indebted to them, and will send back new parts posthaste.

2.18 Glendale Qualifier Reflection

We woke up at the crack of dawn the day of the competition. The robot had been moved to another team members house because Yousuf was out of town for an international science fair. We carpooled for the two hour drive up to the competition and stopped for some food.

We were one of the first teams to arrive and quickly set up our pit for the competition. We had finished what we needed the day before, and were prepared for arrival. We passed inspection easily. However, our first game seemed to follow a familiar trend for destroying motors early in the competition. We successfully ran autonomous, but, one of our programmers left a piece of debug codethat caused the arm motors to burn out during teleop. As a result, we lost the two motors driving the arm. However, our mechanical problems stopped there. We remained undefeated for the rest competition. During alliance selection, we picked Syntax Error and The Millionaire Mind Kids for our partners.

We remained undefeated for the rest of the elimination rounds and reached a personal high score with the Millionaire Mind Kids of 342 points. We also won the Inspire Award and qualified for the Los Angeles Regional Competition.

For futre competitions, we are looking to change the material of the platform used to pick up blocks, so that the mechanism becomes far more consistent and easier. We are also looking to widen the arm - while this could potentialy allow us to pick up five blocks, it is unlikely this will acutally happen in game. Hopefully, we will be able to pick up four blocks more consistently with these changes. We're looking forward to the regional competitions!

3 Software Architecture and Implementation

Detailed in this section is the method and reason to our software architecture. In particular, we aimed for our architecture to be stateless with respect to the current operation, to maximize efficiency of joystick checks (which are normally slow in RobotC), and to allow dynamic and easy reassignment of both controller information during teleop, as well as both movement sequencing and heuristic decision trees during autonomous.

We have chosen a particular structure for our code. We have rewritten `JoystickDriver.c` to better express our needs. By removing superfluous tasks from the driver, we maintained functionality while increasing our efficiency by approximately threefold. We have created a set of utility headers: `teleoputils.h`, `autoutils.h`, and `sharedutils.h` to address the need for macros, `#define` statements, and standardized functions. Both `teleoputils.h` and `autoutils.h` import `sharedutils.h`, so that file is never explicitly included in the main code body. Drivers were pulled from HiTechnic's online library for 3rd party sensors.

Additionally, it may be worth taking a peek into the code appendices in the back of this engineering notebook.

3.1 Teleoperated Mode

Teleoperated mode has the following requirements:

1. Smooth arcade driving
2. Easy reassignment of buttons
3. Stateless control of motors and robot state
4. Efficient button control and loop checking
5. **Must** use only one controller

In order to implement this effectively, we have implemented several macros. These macros allow us to later set the powers of the motors without much effort.

3.1.1 Drive Code and Reasoning

Our main body of code is run through the following:

```
task main() {
    while(true) {
        getJoystickSettings(joystick);
        checkJoystickButtons();
        setLeftMotors(powscl(JOY_Y1)-powscl(JOY_X1)/1.75);
        setRightMotors(powscl(JOY_Y1)+powscl(JOY_X1)/1.75);
    }
}
```

First, grab the current joystick configuration from the controllers. Then, check to see if any buttons have changed (`checkJoystickButtons();`). Then, set the motor powers for arcade drive.

Power scaling The `powscl(int)` function's definition is intended to compensate for the large deadband range which occurs under standard drive conditions. The controller's user really only needs two ranges: a high-precision, low power range near zero, and a low-precision, high power range near the maxima. While an exponential function could be used, it is much slower, and much more hard to tune. Instead, we draw two lines: a shallow slope for the first segment, then a large slope for the second segment of the controller.

This provides both high precision and high power where needed. As the driver does not generally use the range in $[60, 85]\%$, there is no concern about the nonlinearity. The function is defined as follows:

```
float powscl(int xz) {
    float sign = (float)sgn(xz);
    float x = abs(xz)/128.0;
    if(x < DISTA)
        return 100* sign * (x*SLOPE);
    else
        return 100* sign * ((DISTA*SLOPE*(x-1.0) - x + DISTA) / (DISTA - 1.0));
}
```

Compensation for the Old and New Joystick Configuration It is necessary to compensate for both the old and new controller configurations. As the controller has been updated, the buttons have changed - however, competition rules permit the use of both controllers. Therefore, we must be able to accommodate this change if necessary. We have done so through the use of a define statement: if we have `#define ALTLOG`, then we switch to the old button layout.

Button Press Checking Requires a Stateless Organization In order to easily and effectively change the functionality of the controller, a particular design was implemented:

```
void invokeButton(int button, bool pressed) {
    switch(button) {
        case JOY_X:  if(pressed) {servo[servoL1] = 156; servo[servoL2] = 26;}
            else {} break;
        case JOY_Y:  if(pressed) {servo[servoL1] = 120; servo[servoL2] = 40;}
            else {} break;
        case JOY_A:  if(pressed) {} else {} break;
        case JOY_B:  if(pressed) {motor[mSpin] = 100;} else {motor[mSpin] = 0;}
            break;
        case JOY_RB: if(pressed) {setArmMotors(100);}  else {setArmMotors(0);}
            break;
        case JOY_LB: if(pressed) {setArmMotors(-100);} else {setArmMotors(0);}
            break;
        case JOY_R3: if(pressed) {} else {} break;
        case JOY_L3: if(pressed) {} else {} break;
    }
}

bool t[8];
void checkJoystickButtons() {
    for(int i = 0; i < 8; i++) {
        if(joy1Btn(i) != t[i]) {
            invokeButton(i, !t[i]);
            t[i] = !t[i];
        }
    }
}
```

This may appear confusing at first, however, there are a couple points: `checkJoystickButtons()` is actually called from the main loop. It simply checks to see what buttons have changed on the controller, and calls the appropriate `invokeButton(int, bool)` arguments. In doing so, we can determine exact behavior on button presses with ease. As our robot is very simple, we do not need more than a handful of buttons, so most of them remain unassigned.

Code Optimizations Although this method works for determining which button is being pressed, we quickly found it is not the most elegant way of doing so. First off, we use a `bool[]` to hold the state of each button. This seems intuitive, but the RobotC compiler actually creates an entire `char` to hold either `true` or `false` for every `bool`; this is a waste of memory. The other issue can be found in the processing structure. Each iteration of the program checks all possible button conditions; this is a waste of processing time if there has been no change since the last iteration.

```
short btn = JOY_BTN;//local store = live store, initially
void checkJoystickButtons() {
    if(btn == JOY_BTN) return;//checksum
    for(short i = 11; i >= 0; i--) {
        if((btn>>i) ^ (JOY_BTN>>i)) {//check each button for a change
            invokeButton(i, ((btn & (1 << i)) == 0));//trigger event (#, down|up)
            btn ^= 1<<i;//mirror changes in local store
        }
    }
}
```

We solve the former issue by means of storing each button state in a single bit of data, reducing our memory footprint. Teams are encouraged to call the `joy1Btn(int)` function each time they wish to check the state of a particular button, but this can become cumbersome of one wishes check multiple buttons in real-time. The "JoystickDriver.c" file that we must use for field communications stores each button state in a single `short`; this means that we are capable of running a checksum of the joystick state before we check each button. This not only reduces our time per iteration, but also allows for our robot to be more responsive to joystick changes due to the inherent speed of bitwise operations.

3.2 Autonomous Mode

Autonomous, much like teleop, must meet certain criteria:

1. Recognize which crate has been chosen with the IR beacon
2. Move to said crate in a timely manor
3. Move to onto the bridge after scoring our block
4. Avoid any other robots on the way

Crate Detection Much like any other team, we use the HiTechnic IR Seeker to determine which crate our robot should pursue. We use a function that reads not only the position of the IR beacon relative to our location on the field, but also the strength at which it is reading. Having more than one variable to check greatly reduces the generally unavoidable environment-error the comes hand-in-hand with Infrared Light detection.

Movement Functions Using motor encoders, sensing how far a robot has moved is relatively simple in theory. Generally, the code becomes messy when the programmer has to remember motor encoder values and direction. To solve this issue, we have implemented general movement functions based on inches.

```
void rbtMoveFdTime(float inches, int msec) {
    int enc = getEncoderByInches(inches); clearEncoders();
    int norm = -1.0*sgn(inches);
    ClearTimer(DrTimer);
    while(leftEncoder < enc && rightEncoder < enc && time1[DrTimer] < msec) {
        setLeftMotors (100*norm);
        setRightMotors(100*norm);
    }
    setLeftMotors(0); setRightMotors(0);
}
```

This function allows our robot to move forward by a arbitrary number if inches and finish the motion in less than the specified time. Generally one does not want to move forward for an amount of time because the power fluctuates with battery levels. On the same hand, if one moves forwards based on just encoder values, the motor have the potential to burn out if the robot incurs a collision. Allowing the function to reach the specified distance before a certain amount of time insures that neither of these situations have a high probability of surfacing.

Object Avoidance Through the use of the HiTechnic SuperPro that was made legal by this year's game manual, we have been able to mount a high performance ultrasonic sensor to validate that the path we wish to take is open. If there is an obstruction, the robot moves onto the bridge via an alternate route. This feature increases the probability of a higher average score.

```
bool pathClear(float dist){
    pause();
    float read = 0;
    for(int i=0;i<10;i++){read+=(analogRead(A3)*0.4);wait1Msec(5);}
    nxtDisplayBigTextLine(3,"%f", read/10.0);
    wait1Msec(2000);
    return ((read/10)<dist?false:true);
}
```

3.3 Algorithms and Cartesian Mathematics

3.3.1 Hybrid Localization Using Gyroscopes & Odometers

Odometric Data We begin by assigning the following constants:

$$D_{ot} = \frac{\text{Distance}}{\text{odometer tick}} = \pi(\text{wheel diameter})/(\text{ticks/revolution})$$

$$\theta_{ot} = \frac{\theta}{\text{odometer tick}} = \pi\left(\frac{\text{wheel diameter}}{\text{distance between wheels}}\right)/(\text{ticks/revolution})$$

We can calculate $(x_{\text{enc}}, y_{\text{enc}}, \theta_{\text{enc}})$ from the odometer as follows:

$$\begin{aligned}
\mathrm{d}l &= l^t_{\text{enc}} - l^{t-1}_{\text{enc}} \\
\mathrm{d}r &= r^t_{\text{enc}} - r^{t-1}_{\text{enc}} \\
\mathrm{d}D &= \frac{1}{2}(\mathrm{d}l + \mathrm{d}r)D_{ot} \\
\mathrm{d}x_{\text{enc}} &= \mathrm{d}D\cos(\theta^t_{enc}) \\
\mathrm{d}y_{\text{enc}} &= \mathrm{d}D\sin(\theta^t_{enc}) \\
\mathrm{d}\theta_{\text{enc}} &= (\mathrm{d}r - \mathrm{d}l)\theta_{ot} \\
x_{\text{enc}} &= x^{t-1}_{\text{enc}} + \mathrm{d}x_{\text{enc}} \\
y_{\text{enc}} &= y^{t-1}_{\text{enc}} + \mathrm{d}y\text{enc} \\
\theta_{\text{enc}} &= \theta^{t-1}_{\text{enc}} + \mathrm{d}\theta_{\text{enc}}
\end{aligned}$$

l denotes the left side of the robot, and r denotes the right side of the robot. D denotes the distance.

Localization Algorithm The robots motor controller calculates position and orientation $(x_{\text{enc}}, y_{\text{enc}}, \theta_{\text{enc}})$ from encoder ticks and sends the data to an on-board computer. The mounted gyroscope communicates with a gyro driver which integrates the rate values into an absolute angle (θ_{gyro}). Global position $(x_{\text{rbt}}, y_{\text{rbt}})$ is found by transforming the translation vector from encoder space to gyroscope space. Global angle (θ_{rbt}) is the gyro angle (θ_{gyro}). The following describes the computation as an iterative algorithm:

$$\begin{aligned}
\mathrm{d}x &= x^t_{\text{enc}} - x^{t-1}_{\text{enc}} \\
\mathrm{d}y &= y^t_{\text{enc}} - y^{t-1}_{\text{enc}} \\
\mathrm{d}\theta &= \theta^t_{\text{gyro}} - \theta^t_{\text{enc}} \\
x^t_{\text{rbt}} &= x^{t-1}_{\text{rbt}} + \cos(\mathrm{d}\theta)\mathrm{d}x - \sin(\mathrm{d}\theta)\mathrm{d}y \\
y^t_{\text{rbt}} &= y^{t-1}_{\text{rbt}} + \sin(\mathrm{d}\theta)\mathrm{d}x + \cos(\mathrm{d}\theta)\mathrm{d}y \\
\theta^t_{\text{rbt}} &= \theta^t_{\text{gyro}}
\end{aligned}$$

4 Outreach

As a team, de.evolution believes that robotics is more than just the engineering. Our first year as a team we saw that the other FTC team at our school, Domo Arigato, was entirely seniors and we were sad that it was going to disappear. This inspired us to try and spread robotics to the younger students and have people involved to keep robotics teams going and to propogate the message of FIRST throughout our community. We have started a robotics class at our high school which has turned into the team Domo Arigato. Due to the popularity of the class a second class, and subsequently an FTC team, has been started. Both the Robo Ravens and Domo Arigato, along with our own team, de.evolution, work hard to start a robotics and engineering culture at our school.

4.1 Community Outreach

As a team, de.evolution has tried to do a lot to help spread the ideas of FIRST, robotics and engineering. The following is an extensive list of some of the community outreach that we have done:

1. Robotics demos to elementary school students at Solana Pacific Elementary, Del Mar Heights Elementary, Rancho Santa Fe Elementary
2. PTC World demo in June 2013
3. Mentoring of the three Islamic School of San Diego FLL teams
4. Sister team of the San Dieguito Academy FTC teams Paradox Squared and Paradox Cubed
5. Demo for school board & superintendent sponsored building/engineering production district-wide with bond measure
6. Hosted Robotics Week at CCA in coordination with Domo Arigato (3513)
7. Publication in newspapers & press releases about STEM/robotics
8. Publication in school's magazines
9. Over 20% of school has become involved with the FIRST robotics programs after the inception of our team
10. Inspired the FIRST robotics classes
11. Incorporating art and conservatory at CCA with the robotics program
12. Incorporating the humanities conservatory at CCA with the robotics teams
13. Working with TEDxYouth@SanDiego to spread robotics to the students in San Diego
14. Attended and presented at Rotary meetings to show our community the influence of robotics

Additionally, in the pits, we offer to help any and all teams who are in need of assistance - including offering our contact information to pit admin if any team needs help.

4.2 In-person demos at middle and elementary schools

We have presented the FIRST Tech Challenge to three separate elementary schools: Solana Pacific Elementary, Del Mar Heights Elementary, and Rancho Santa Fe Elementary. We inspired the creation of teams at these schools this FTC season, and will bring them into our robotics program at Canyon Crest Academy if they choose to attend our school.

> Ever since last spring, when their team members did a presentation at our school for Science Discovery Day, they have been an inspiration to our robotics students and are chiefly responsible for encouraging our 8th graders to start an FTC team this year.
>
> –David Warner, Coach for the RSF Eagles FTC Team

When we showed our robot to the students at these schools we had their teachers drive the robot. The kids loved it! It was definitely a great experience for us to be able to show so many students how much we loved robotics and the subtle beauty that is found in engineering and FIRST competitions.

The students at the schools are interested in robotics, and are now aware of what the FIRST program can provide. We are thrilled to have the opportunity to interest more students in FTC, and look to continue doing so next year!

4.3 Helping other teams online

As the majority of the members on our team are also on the FRC team The Aluminum Narwhals we realize that having online support is one of the best ways to debug and learn about robotics. Chief Delphi, one of the prominent places for online FIRST discussion, is very popular among FRC teams. However, FTC teams do not have nearly as much online support as they may need or like. As a team we have shifted a lot of our focus to helping teams online. We have open-sourced all of our code and have created code tutorials to help out many other teams. We have helped teams in the following ways

4.3.1 Syntax Error code optimization assistance

At a San Diego regional qualification competition, we entered a discussion with team 6077, Syntax Error, about code optimization. We were sharing our varied tricks which were used in code, and through discussion, it became clear that there were a couple points they were particularly interested in implementing. We passed them some contact information, and later received an email from them:

> Hey Noah,
> We're interested in optimizing the speed of our teleop code, and I know you had mentioned at the tournament that you guys XOR-ed the joystick values to determine when the values had changed–allowing you to skip checks when things weren't changing. Could you please go into more depth as to how you achieved this? Did you directly modify joystickdriver.c? Which values should we be XOR-ing exactly?
> Thanks,
> -Collin

We wrote out a document generally detailing the approaches we took when optimizing our teleoperated code. My response was the following PDF file:

Hello Collin/Syntax Error/6077!

There are a couple main things that we've done to improve our teleop code's responsiveness. The first has to do with button checking and stateless button operations, and the second concerns significant, er, *improvements* to the JoystickDriver file.

First, if you'd like to view our source code, all of it is available on Google Code (project ftc-team-4278). This will link you to the file browser, but if you use Subversion, feel free to checkout a copy and look through it! It might help to see specifically what I'm talking about in the files.

The first improvement to teleop comes from the simple idea that code run fewer times runs faster. Cut down on the number of iterations and checks, and you simultaneously cut down on the number of cycles used for dead processing. Our teleop code is laid out in a pretty simple structure:

```
task main() {
    unlockArmMotors(); clearEncoders();
    waitForStart();

    while(true) {
        getJoystickSettings(joystick);
        checkJoystickButtons();

        setRightMotors(powscl(JOY_Y1)-powscl(JOY_X1)/1.1);
        setLeftMotors(powscl(JOY_Y1)+powscl(JOY_X1)/1.1);
    }
}
```

We do two things: check to see if the buttons have changed, and set the motor powers. The core principle here is actually the same as I imagine most robots' code is: do things with the buttons, then do things with the joysticks (or vice versa). The optimization comes from the `checkJoystickButtons()` function; it only runs code if the buttons have changed. Here's how that function works:

```
#define JOY_BTN joystick.joy1_Buttons //from teleoputils.h

short btn = JOY_BTN;
void checkJoystickButtons() {
    if(btn == JOY_BTN) return;
    for(short i = 11; i >= 0; i--) {
        if((btn>>i) ^ (JOY_BTN>>i)) {
            invokeButton(i, ((btn & (1 << i)) == 0));
            btn ^= 1<<i;
        }
```

```
    }
  }
```

The first line is the most important. `if(btn == JOY_BTN) return`; i.e. if the button state has not changed, do nothing. This is important, as checking the state of each button is (comparatively) expensive. The rest of this is just a pile of binary math, which results in the function `invokeButton()` being called with the arguments `int button, bool pressed` to indicate which button has changed to which state.

If you're really interested, here's what's going on in a bit more detail (I'll assume you're familiar with XOR):

In essence, the `if` statement is using the XOR operator to check if the previous state (`btn`) is different than the current state (`JOY_BTN`). The statement `btn>>i` shifts the button value right by the button number; in essence, it makes the i^{th} button the first digit of the binary `btn`. It does the same thing on `JOY_BTN`. If the first digits differ, the XOR operator will make the first digit of the output 1. The if statement will check this digit for true/false value. And, if this value is 1 (i.e. differing values), then it calls `invokeButton`. The next line simply sets that value to 0 in `btn` so it isn't called again.

The second set of optimizations actually occurred within the JoystickDriver itself. It turns out that (unsurprisingly) the current JoystickDriver is actually relatively inefficient. *(A side note: We've optimized our JoystickDriver.c for use with one controller only - you probably don't want to use our driver if you want two controllers)* Our philosophy with this driver has been simple: **if you keep removing things, and it still works, then good. You're improving it.** So, I'm going to point out a couple things which can be safely trimmed, but I challenge you to see how much you can trim without breaking it. I guarantee it will be at least twice as fast when you're done with it. We got ours down to 126 lines... :]

I guess the main thing here is the `displayDiagnostics` task, which, it turns out, actually uses a significant number of cycles. You know that debug info that shows up on your screen with voltages? Turns out, that actually comes from the `JoystickDriver` - and it isn't actually all that useful. You only really need to see it once, and don't need it running constantly in the background. You can simply delete the entire task. As well as the `disableDiagnosticsDisplay` function. And pretty much everything concerning diagnostic output.

I guess my main advice with this file is: most of it isn't needed. Make sure you understand what you're removing, but it's pretty much safe to remove most of it.

Hopefully this helps! If you have any questions, feel free to contact me personally. I'd be happy to help!

The aim was to create an easy-to-follow guide for someone with experience to optimize the teleoperated program. To our enjoyment, they reported significant increases in response time on their controller.

4.3.2 Reddit

To our pleasant surprise the FTC Reddit subforum was started this year and our team has jumped completely on board to help other teams with both programming and mechanical help. We often will have Skype calls with other teams around the country or just send long documents to them. We have also created extensive and comprehensive documents that detail all of the details, including the advantages and disadvantages, of the different drive trains. The following pages have some of our biggest discussions with teams on Reddit:

8

Offering programming assistance to all teams!

self.FTC

submitted 5 months ago* (last edited 5 months ago) by Telthien FTC 4278 (Software)

If any team would like live assistance with their code in RobotC, Team 4278, de.evolution, is offering to assist.

We are well-versed in all available RobotC capabilities, and will help with any challenge. We received second at the world championship during the Get Over It! challenge as a rookie team, and received two division awards for the Bowled Over challenge. We earned six total awards last year in the Ring It Up challenge.

We can assist with:

- Conceptualization of code
- Autonomous development and testing
- Manual code and control theory/logic
- All HiTechnic and NXT sensors
- Code structure and design
- NXT/RobotC/Samantha debugging and configuration
- Code formatting
- If you think of something else you'd like help with, don't hesitate to ask!

We will help teams develop their own programming skills-not just write code for you. Our goal is to assist teams by teaching them the skills they need to progress on their own.

Note: I have passed my contact information through ROT13. To find my actual contact information, go here: http://www.rot13.com/index.php.

To get in contact with me, you can add my Skype (nygubea.rfgina, through ROT13). I request that, if you contact me by Skype, you put that you put your team number and your name in the contact request - contact requests without team numbers and names will be ignored, for obvious reasons.

Thank you, and good luck! --Noah, 4278 de.evolution co-programmer

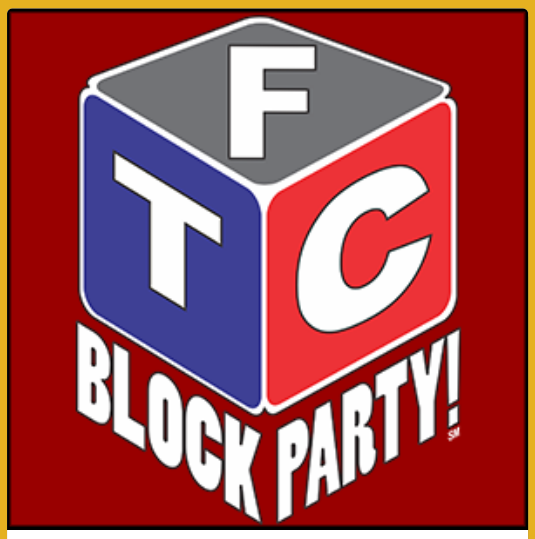
5 comments share

sorted by: best

[–] idoescompooters 1 point 4 months ago

Are there any specific websites, videos, etc. that you would suggest for learning RobotC? I've found a lot RobotC learning resources, but I was wondering if there were any that you think really stick out and would help people beginning with RobotC and be able to program with it. By the way, I have previous programming experience.

permalink



FTC

search reddit

this post was submitted on 10 Sep 2013

8 points (78% like it)

11 upvotes 3 downvotes

shortlink: http://redd.it/1m5k1t

remember me reset password Login

Subscribe 313 Roboticists *(~6 here)*

Note

If your submission doesn't seem to show up, message the mods. It's very likely the submission accidentally got flagged as spam by the filter.

What is /r/FTC?

Anything and everything FTC related! If you've stumbled into this sub, read about the FIRST Tech Challenge here: http://www.usfirst.org/roboticsprograms/ftc

Add Your Team Number!

You can now apply your own user flair for your account! Click edit flair and select the flair that applies to you! Then just edit the numbers to your team numbers!

e.g. *mozrila FTC 3415*

[–] Telthien FTC 4278 (Software) [S] 1 point 4 months ago

Actually, for new programmers, the standard RobotC introduction videos and documents are rather useful. Let me know if those help, or if you need a pointer to where they're located (they can be somewhat difficult to find).

They're primarily useful in that they provide tasks. However, the actual code instruction is good when supplemented by a knowledgeable team member/mentor.

permalink parent

[–] idoescompooters 1 point 4 months ago

Are these the training videos? Could you possibly link me to these. In the past, I've discovered training videos and a huge amount of webinar videos that I may give a try.

permalink parent

[–] Telthien FTC 4278 (Software) [S] 1 point 4 months ago

http://www.robotc.net/teachingmindstorms/

You don't actually *need* to build the robot they have there, but it may be helpful. The introduction can be done conceptually.

permalink parent

[–] idoescompooters 1 point 4 months ago

Yeah, that's what I was talking about.

permalink parent

e.g. *Anton338 FTC 3415 (Alum)*

Important Links

US FIRST
Offical FTC Forum
FTC Blog
10% off Tetrix Parts
AndyMark

Related Subreddits

/r/FRC
/r/vex
/r/Robotics

created by FTC Team 3415 -- The Lancers a community for 2 years

Message The Moderators

COMPETITION JUDGES

rainydayplaylist
thebootsiest
mozrila [MOD[FTC 3415
Anton338 FTC 3415 (Alum | Mentor)
stanleychg FTC 3415
TurnRobotOn Volunteer
robertf224 FTC 3415 (Alum)

5

What Gear Ratio Are You Using For Hanging // Weight?

self.FTC

submitted 2 months ago* (last edited 2 months ago) by Arctic_Wind

FTC 6112 Wolfbyte

Hey guys,

What gear ratio are you guys using for your hanging system, and how much does your robot weigh?

Is there an easy way to solve what ratio you should use mathematically? Or is it more of a "safe bet" sort of thing?

Our robot is about 35 pounds, but we're looking into also supporting another robot to hang. How does 9:1 gear ratio sound?

Thanks! FTC 6112 Wolfbyte Alaska

P.S. If you're wondering what some others are doing too, I posted this topic on the FTC Forum as well.

FTC Forum Thread: What Gear Ratio Are You Using For Hanging // Weight?

7 comments

sorted by: best

[–] Oriek 6376 2 points 2 months ago

We're using a rack and pinion system with a... well I don't know the gear ratio but it's the small textrix gear to the medium one. Works super well.

permalink

[–] Dastyruck 4278 2 points 2 months ago

We are using an arm to hang, so a large part of deciding our gear ratio was using torque calculations to figure out how much torque we would need in order to raise our robot. I'll spare you the complexities of calculating this but I can say that our arm is geared 7.6:1 and the arm hooks onto the bar about 17 inches from the rotational axis. We weigh about 30 pounds and use two tetrix motors to lift ourselves.

9:1 with two tetrix motors should be more than enough to lift a 35 pound robot, depending on your method of hanging from the bar. If you are using an arm, the further away your "hook" is from the rotational axis, the more torque required, thus the higher the reduction you will need.

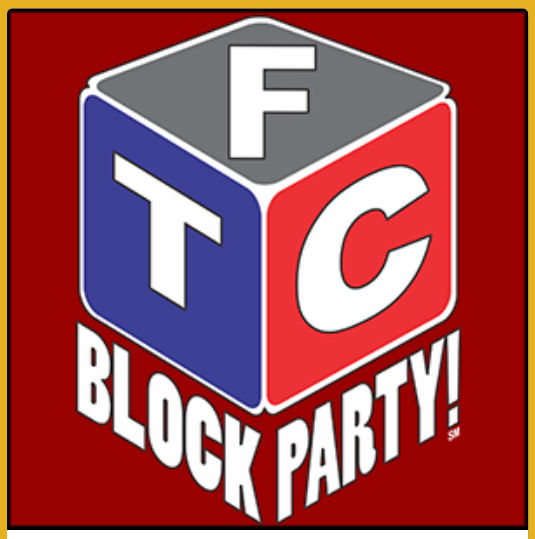
If you post more specifics, I would be happy to help you figure out an optimal solution.

permalink

[–] Arctic_Wind FTC 6112 Wolfbyte [S] 1 point 2 months ago

The current plan is to use a "grappling hook," with the length of the line being 28 inches (though this could be moved to the second level, so could be 16 inches). We were thinking about using the tetrix worm gears, which have options of 4:1, 10:1, 20:1. We ideally would like to use 1 motor here; using 2 would be rather costly.

The idea is to have an arm which would have a fulcrum length of about 5-6 inches. It would



FTC

search reddit

this post was submitted on 22 Dec 2013

5 points (85% like it)

6 upvotes 1 downvote

shortlink: http://redd.it/1thtro

remember me reset password Login

Subscribe 313 Roboticists *(~6 here)*

Note

If your submission doesn't seem to show up, message the mods. It's very likely the submission accidentally got flagged as spam by the filter.

What is /r/FTC?

Anything and everything FTC related! If you've stumbled into this sub, read about the FIRST Tech Challenge here: http://www.usfirst.org/roboticsprograms/ftc

Add Your Team Number!

You can now apply your own user flair for your account! Click edit flair and select the flair that applies to you! Then just edit the numbers to your team numbers!

e.g. *mozrila FTC 3415*

portrude a "second bar" for another robot to hang on. Ideally again 1 motor used here, and we're looking into worm gears again for this.

If the arm were supporting another robot, would the line also take some of the stress? Would we need to gear for about 70 some pounds on our line? As for each individually, would the 10:1 worm gears work? We appreciate any help you can give!

permalink parent

[–] **Dastyruck** 4278 1 point 2 months ago

From what it sounds like, you would like to first hang yourselves using a grappling hook method using a worm gear and then a second arm would then be used to grab and then lift another robot off of the ground.

If this is the case, your line would indeed need to be able to handle the force of two robots statically. The worm gear used for the first stage would also need to not back drive under the stress of two robots. Though, neither stage would need to pull with the force required to lift two robots.

Starting with the first stage, the grappling hook, the amount of force the "winch" system will have to pull with is going to be the weight of your robot, in this case 35 lbs. The torque that a tetrix motor can produce at stall current is 325 Oz - in. The force that the winch system pulls is F = T / (r * sin(A)). In this case, the angle of lever arm will always be 90 degrees, which simplifies the expression to F = T / r where F is the force, T is the torque, and r is the radius of the axle that your winch cable is wrapped around. Since our torque is given in Oz-in, we want to use ounces for the force and inches for the radius. Converting our robot weight of 35 pounds into ounces gives us a force required of 560 ounces. If we used no added gearing, our formula would be 560 oz = 325 Oz-in / r. Which solves to r = .58 inches. If you move up to the 10:1 worm gear, the effective torque would be multiplied by 10. Which would make the formula 560 oz = 3250 oz-in / r. Which solves to r = 5.8 inches. So, if you choose an axle for your winch smaller than this, you will be able to lift yourself. I would stay on the side of caution and use something smaller than this, such as a 1.5" radius axle. The larger the axle, the faster you will go up.

For the next part, you have already said that your lever arm will be 5-6 inches. Staying on the side of caution, we should say that it is at 6 inches in radius. The force required to lift is again 560 ounces if the robot is of similar weight. Though, this time, the formula does not simplify out as the angle of the lever arm will change as the arm is rotated upwards to lift the robot off of the ground. Thus, F = T / (r * sin(A)). Solving this equation for torque, we get T = r*sin(A)*F. As the arm moves towards 90 degrees, or parallel to the ground, the torque required increases to its maximum. So, if we set the angle equal to 90 again, we are left with T = r*F. Since our radius is 6 inches and our force is 560 ounces, we find that we need a maximum torque of 3360 Oz-in. With the 10:1 worm gear ratio, we only get 3250 Oz-in of torque when the motors stall. So, increasing this to the 20:1 would be the best way to go if you wanted to use one of the worm gear options.

I am unsure of how much force is required to make the worm gears back drive or if they can even hold up to that amount of force on them. For a line that would most likely be able to hold up to 70 pounds of tension, I recommend a modern bow string material. We used this last year for our forklift cables. It has the benefit of being very strong while also being very thin, if you need it to hold more, braiding it is very easy to do and will give quite a bit more strength.

If you have any questions or want me to clarify anything, don't hesitate to ask.

permalink parent

[–] **Arctic_Wind** FTC 6112 Wolfbyte [S] 1 point 2 months ago

Geez thanks for all the help on that! Shoutout to Team 4278.

As for our secondary lever arm, it is the last thing we are really looking into but we could also shorten the length of it to perhaps 3 or 4 inches.

The first ideal situation would be to hopefully use our grappling hook to hook, then move away from the hanging bar – give another robot space to get hold of the bar – and then come back and have both robots lift on their own simultaneously (brushing up against each other would prevent swinging). Sort of a concept that I've seen work on Youtube, not particularly sure whether it's consistent.

Thanks also for the tip on the bow string!

e.g. *Anton338 FTC 3415 (Alum)*

Important Links

US FIRST
Offical FTC Forum
FTC Blog
10% off Tetrix Parts
AndyMark

Related Subreddits

/r/FRC
/r/vex
/r/Robotics

created by FTC Team 3415 -- The Lancers a community for 2 years

Message The Moderators

COMPETITION JUDGES

rainydayplaylist
thebootsiest
mozrila [MOD[FTC 3415
Anton338 FTC 3415 (Alum|Mentor)
stanleychg FTC 3415
TurnRobotOn Volunteer
robertf224 FTC 3415 (Alum)

permalink parent

[–] limellipsis FTC 5421 1 point 2 months ago

One word of warning about a grappling hook: you can't make your robot throw it. It's illegal for the robot to launch pieces of itself (as the GDC ruled here). A team got called out on it at our qualifier, and had to take their assembly off- don't let that be you!

permalink parent

[–] tmbrudy 1 point 1 month ago

I use JVNs mechanism calculator for this kind of stuff:
http://www.chiefdelphi.com/media/papers/2755?langid=2

It doesn't have the tetrix motors listed in the list but you can get the new and old motor spec sheets here: http://cheer4ftc.blogspot.com/p/motors.html

permalink

2

Hanging with worm gears

self.FTC

submitted 1 month ago by rohyourb0at 5205 SyBorgs

Hey guys. My team was wondering how we should utilize our (10:1) worm gear set for hanging. We want to use our six bar to hang but were contemplating whether to replace the driving gear(s) with the worm gear or not. Thanks & Merry Christmas!

5 comments

sorted by: best

[–] Dastyruck 4278 2 points 1 month ago

More details would be needed to fully determine whether or not you could hang with your 6 bar linkage. But, if your robot is fairly light, and your 6 bar isn't too long, hanging with it shouldn't be a problem with only one motor. The worm gear might make it so your arm wouldn't move when it wasn't powered, which you want if you plan on hanging with it.

I would make sure that with the gearing that you choose, that your robot could actually lift itself. I have seen quite a few teams attempt to hang and are unable to due to the lack of torque in their current gearing. If you give more details on your robot, I would be happy to help you figure out adequate gearing for your system.

permalink

- [–] rohyourb0at 5205 SyBorgs [S] 1 point 1 month ago

  Thanks for the quick reply, we're using two motors to power our six bar linkage so that's what I was having trouble with figuring out

  permalink parent

- [–] rohyourb0at 5205 SyBorgs [S] 1 point 1 month ago

  I'm unable to send pictures right now because I'm in Florida for vacation lol

  permalink parent

  - [–] Dastyruck 4278 1 point 1 month ago

    No problem, the easiest way to link two motors into one worm gear box is to add an additional gear onto the motor that is the input to the worm box and put a second gear of equal size onto the second motor and have both of these gears mesh. Both motors will be linked in a 1:1 ratio and you will get the torque from both motors into one worm box.

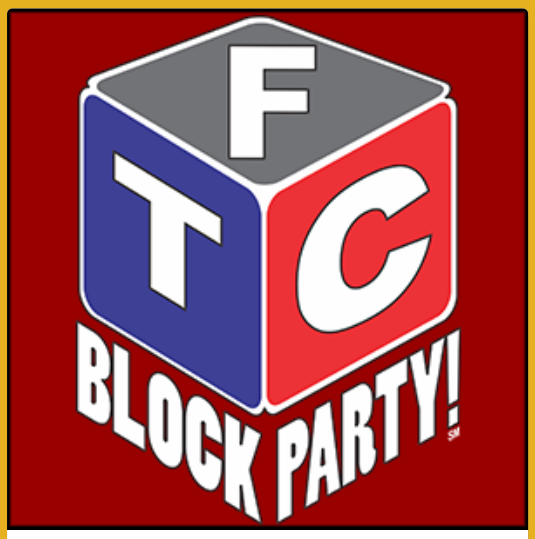
    permalink parent

[–] _TrafalgarLaw 1 point 1 month ago

Something you might want to watch out for is the warping of metal. If you put too much stress onto a single point of an axle (which is easily possible with a high gear ratio) your axle will warp, rendering it useless if you ever want to remove it or replace it.

I don't know how you set up your robot, but hanging should definitely be possible, you just need to be careful with how you distribute your force.

permalink



FTC

search reddit

this post was submitted on 24 Dec 2013

2 points (66% like it)

4 upvotes 2 downvotes

shortlink: http://redd.it/1tnigu

username password

remember me reset password Login

Subscribe 313 Roboticists *(~6 here)*

Note

If your submission doesn't seem to show up, message the mods. It's very likely the submission accidentally got flagged as spam by the filter.

What is /r/FTC?

Anything and everything FTC related! If you've stumbled into this sub, read about the FIRST Tech Challenge here: http://www.usfirst.org/roboticsprograms/ftc

Add Your Team Number!

You can now apply your own user flair for your account! Click edit flair and select the flair that applies to you! Then just edit the numbers to your team numbers!

e.g. *mozrila FTC 3415*

e.g. *Anton338 FTC 3415 (Alum)*

Important Links

US FIRST
Offical FTC Forum
FTC Blog
10% off Tetrix Parts
AndyMark

Related Subreddits

/r/FRC
/r/vex
/r/Robotics

created by FTC Team 3415 -- The Lancers a community for 2 years

Message The Moderators

COMPETITION JUDGES

rainydayplaylist
thebootsiest
mozrila [MOD[FTC 3415
Anton338 FTC 3415 (Alum | Mentor)
stanleychg FTC 3415
TurnRobotOn Volunteer
robertf224 FTC 3415 (Alum)

Additionally, the code assistance post was also mirrored on the Chief Delphi FRC forums. We have contacted six teams through these mediums.

4.3.3 FTC Team NESI (6033)

One of the many responses we received for our online posts has been copied here. In short: a team was in need of holonomic drive assistance, and we responded with details about how the algorithm might best be implemented. We did so with the intent to provide a conceptual structure, while still leaving an educational problem for the team to understand and develop. In this way, we are not simply giving them the solution; we are instead helping them understand the tools they need to implement a solution, furthering their understanding and education.

> “I saw your offer to help teams with programming. We are a new FTC all girls team and could use some help with programming our robot. We originally had tank drive and were able to modify the code to work. However, we thought omni direction would be be better for us and built a [chassis] with omni directional wheels mounted at 45 degrees in each corner with a motor on each. There is an encoder on each. I know there has to be some conversion to establish drive vectors, but having some challenges with that. If you had some sample RobotC code that would be great. The girls are very sharp and could make it work, we just need a little help. Ideally we want to use one joystick to go forward, back, left, right. Then the other joystick to rotate left or rotate right.
> Anything you can do to provide sample code would be VERY much appreciated! Thanks!
> *Coach Jerry, Team NESI (6033)* ”

Our approach to helping teams in this situation is not to outright give the solution, but help provide a conceptual framework for thinking about approaching a solution. That way, teams still gain the educational experience of solving problems themselves, but aren't working in the dark when they attempt to solve the problem.

“I'd be happy to explain conceptually! I'm sure you and I would both agree that, even though I could simply give source code, it's better for the programmers to come up with it themselves to gain a conceptual understanding. Of course, holonomic drive (45 degree omni) is confusing, so feel free to ask questions!

First, the programmers will need to have an understanding of polar coordinates. Nothing too deep, just the basics: that they are expressed with an angle and a radius. The best holonomic drive will follow sines and cosines, so I recommend converting rectangular coordinates to polar coordinates. In this way, the theta of the joystick position becomes the direction the robot is to move in, and the “r” magnitude becomes the speed.

There are two parts to a holonomic drive: rotation and translation. Rotation allows the robot to turn to any bearing, and translation allows the robot to move in any direction at a time. Make sure the wheels are aligned such that, in positive power, both the left and right side motors move forward.

Rotation should be added to translation, since translation is more important and complex. Thus, the goal is to work out a translation algorithm. Remember that your controller should be stored in polar coordinates. It is helpful to plot out on paper which direction you want the motors to move for each angle; for instance, at 45 degrees, you want the front left (FL) and bottom right (BR) at 100% power, and FR/BL at 0% power. You can use these points to build a sine/cosine graph for each wheel.

Once a translation algorithm has been tested and implemented, rotation is the next step. Conceptually, to rotate the robot, all the wheels need to move the same direction. Thus, you can simply add whatever your rotational value is (we use the x axis on the second joystick, but be sure to divide by 128 to get a number from -1 to 1) directly to the output from the sines and cosines.

When you add rotation, you will encounter another problem: sometimes motors will be set to 200% power, while others are at 100%. Yet the robot still needs to turn. The cause of this problem lies in the rotation. If the robot is going at 45 degrees, FL/BR = 100%, FB/BL = 0%. However, if you are rotating at full power, you'll end up with a situation where FL/BR = 200%, FB/BL = 100%. The robot will just drive forward. What you need to do at this point is find the maximum power, and divide all the powers by it. They will stay proportional, and you'll get the expected output.

Please let me know if you have any questions! It's certainly not a simple algorithm to think about. I've tried to give a general conceptual outline of how it works, since we'd probably both agree it's better to write the code oneself. Still, if you would like further help, feel free to ask!”

While we have removed most of the discussion from this text, the conversation was left with:

> Your explanation will be a big help to the programmers! I anticipate they will have some questions for you about how the theory gets put into practice with RobotC. Thanks for pointing out some of the troublesome aspects of how translation and rotation come together. We will be in touch. Thanks!

4.4 Mentoring of FLL teams

Our team aided the Islamic School of San Diego's FLL Teams during their rookie years by mentoring the young kids on robotics and the engineering design process in general. Our students voluntarily watched over the excited and energetic children, assisted in teaching them key concepts, and created useful PowerPoint presentations that were presented to further educate the FLL teams.

In each meeting, the FLL teams was taught a concept and then given time to build something from LEGOs using their newfound knowledge. Since our de.evolution volunteers were there, the FLL leader could split the team into small groups with a mentor or leader helping each group. Another advantage of having an FTC mentor available was that the volunteer could lead the team while the leader temporarily prepared the next lesson or dealt with other essential tasks. It relieved the leader of FLL and kept everything running smoothly.

The lessons taught the eager and energetic students about what causes earthquakes and the destruction that they inflict. This understanding supported the kids in cooperatively creating ideas for a machine to aid anyone who recently experienced a devastating disaster. The mentors also educated the group about the engineering design process to assist them in efficiently building a better solution when they were ready to begin fabrication of their product.

We found that mentoring FLL teams has not only allowed us to teach the students a lot about robotics, but we gained valuable experiences by doing so.

4.5 Meeting with the SDUHSD Disctrict Office

A while back, we met with the San Diego Union High School District (SDUHSD) district office to present the FIRST Tech Challenge program. It was a significant demonstration, as our ultimate goal was to inspire further funding for robotics in regional high schools.

Our efforts paid off. Recently, Proposition AA, an educational bond bill, was passed in the San Diego area. This included significant funding for the construction of a robotics building at our host high school, Canyon Crest Academy. We have already brought approximately 15 to 20% of the high school population through some form of FIRST's programs, and this will hopefully expand both the educational capacity and general awareness of FIRST among the high school students.

4.6 TEDxYouth@SanDiego

As our team is greatly involved with Canyon Crest Academy as an academic instition we teamed up with the FRC team at our school and began creating promotional videos to show robotics to our school. At TEDxYouth@SanDiego we were invited to put on a robotics and dance choreography. We worked with the FRC team at our school and the dance conservatory to put on a truly magnificent performance. It was the joining of exceptional engineering and amazing talent in terms of the arts.

At TEDxYouth@SanDiego, over 500 students from high school in the San Diego region all came together to be inspired. By being honored to present our robot on the TED stage we were able to reach many students and show them the elogance of robotics. By integrating it with the arts we are really able to show the practical and artistic aspect of robotics.

4.7 Press Releases

To really reach the community of professionals and the general public, we turned towards press releases. By having the media come to us to cover not only our successes as a team, but our ability to inspire and reach out to many other students has given us many new opportunities to go to middle schools and mentor other FTC and FLL teams. The following pages are excerpts of some of the media and press that has covered our team:

CCA De-Evolution robotics team members: (L-R) Kian Sheik, Noah Sutton-Smolin, Tristan Murphy, Alex Quan, Christian Cooper. Not pictured: Mariella Gauvreau, Ryan Lee, and Yousuf Soliman. Noah and Tristan are holding the two trophies for winning the Inspire Award and for Captain of the Winning Alliance.

Double win for CCA's De-Evolution robotics team

Canyon Crest Academy's robotics team, De-Evolution, has now qualified to compete in the Los Angeles regional competition, after the team's double win at the L.A. qualifying tournament held Jan. 25 in Glendale.

De-Evolution has already qualified to compete in the upcoming San Diego regional competition with the team's double win in December at the Escondido qualifying tournament.

Going into the semi-finals at the L.A. tournament, De-Evolution was ranked Number 1 out of 30 teams and maintained its top position, remaining undefeated the entire day.

A double win means the team qualified twice for Regionals, by being named the winner of the coveted Inspire award as well as being the captain of the Winning Alliance.

De-Evolution placed second internationally three years ago, as a rookie team. Four members of that rookie team, now CCA seniors, remain on the team.

De-Evolution has now won the Inspire Award at both qualifying tournaments, which is considered more prestigious than winning on the field. The Inspire Award is described by tournament organizers as a team that is "a top contender for all other judging categories and is a strong competitor on the field."

The Inspire Award winner, organizers say, "is able to communicate their experiences, enthusiasm and knowledge to other teams, sponsors, and the judges, [and] will have demonstrated success in accomplishing the task of creating a working and competitive robot."

De-Evolution received a standing ovation from the other teams when it was announced the team had won the Inspire Award.

"Winning the Inspire Award is an unexpected honor," said Noah Sutton-Smolin, De-Evolution's co-president and programmer. "We never strive for the Inspire Award for its own sake. Historically, our team has been about building the best robot possible. This year, though, the focus of our team has shifted from winning the competition to helping others succeed.

"The teams at the competitions are wonderful, as always. The cooperative spirit of the events is fundamentally perspective-altering, as it grows into more than simply a competition for awards."

De-Evolution is a FIRST Tech Challenge (FTC) team, with eight members this year, and is CCA's after-school FTC robotics team. FTC teams are limited to 10 students in grades 7-12.

Based in Manchester, New Hampshire, FIRST (For Inspiration and Recognition of Science and Technology) is an international robotics competition founded by inventor Dean Kamen in 1989. A non-profit organization, FIRST [www.usfirst.org] was created to inspire and motivate students to excel and pursue careers in engineering, science and technology.

De-Evolution team members are committed to advancing understanding of robotics in middle and high schools and to spreading the message of the excitement, team spirit and intellectual stimulation that FIRST competitions provide. Any local schools wishing to start a robotics program are encouraged to contact De-Evolution to schedule a visit or demonstration.

De-Evolution will now compete at the San Diego regional competition Feb. 15 and at the Los Angeles regional competition Feb. 22. The winning teams at Regionals will advance to the Super-Regionals in northern California in March. The winner there will compete internationally in April.

The public is welcome to attend and cheer on De-Evolution at the San Diego Regionals on Feb. 15 at Madison High School in San Diego.

Canyon Crest Academy's De-Evolution robotics team takes top prizes at tournament

Canyon Crest Academy's robotics team, De-Evolution, was named the winner in the team's first appearance in this season's qualifying matches.

Going into the semi-finals, De-Evolution was ranked Number 1 out of 28 teams from several Southern California counties. The team maintained its top position and remained undefeated at the tournament, held Dec. 14 at Escondido Charter High School.

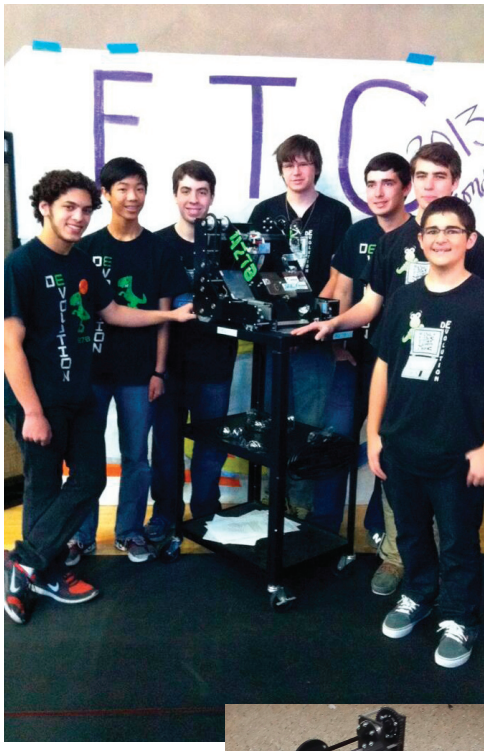
In the finals, the De-Evolution alliance scored the highest points, 256, in any one match of the day.

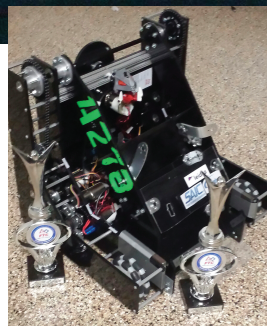
As the Winning Alliance Captain team, De-Evolution now qualifies to advance to the San Diego regional competition to be held Feb. 15 at Madison High School in San Diego. The winning team at Regionals will advance to the Super-Regionals in Northern California in March. The winner there will compete internationally in April.

De-Evolution placed second internationally three years ago, as a rookie team. Four members of that rookie team, now CCA seniors, remain on the team.

De-Evolution was also given the Inspire Award, which is considered more prestigious than winning on the field. The Inspire Award is described by tournament organizers as follows:

"The team that receives this award is chosen by the judges as having best represented a role-model FTC team. This team is a top contender for all other judging categories and is a strong competitor on the field. The Inspire Award Winner is an inspiration to other teams, acting with Gracious Professionalism™ both on and off the playing field. This team is able to communicate their experiences, enthusiasm and knowledge to other teams, sponsors, and the judges. Working as a unit, this team will have demonstrated success in accomplishing the task of creating a working and competitive robot."





(Above) CCA's winning De-Evolution robotics team members (left to right): Yousuf Soliman, Alex Quan, Tristan Murphy, Noah Sutton-Smolin, Ryan Lee, Kian Sheik, Christian Cooper. [not pictured: Mariella Gauvreau]; (Right) The robot with its trophies!

De-Evolution is a FIRST Tech Challenge team, with eight members this year, and is CCA's after-school FTC robotics team. FTC teams are limited to 10 students in grades 7-12.

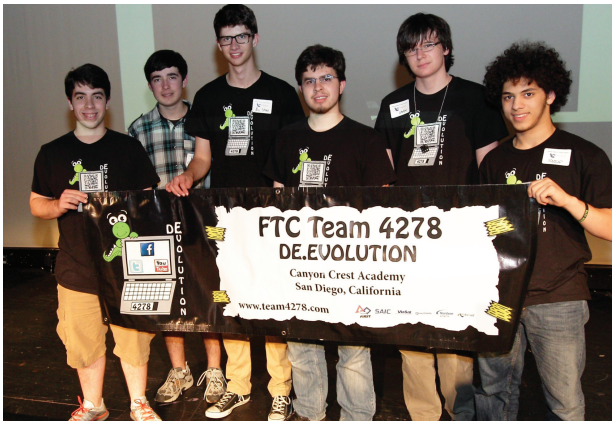
Based in Manchester, New Hampshire, FIRST (For Inspiration and Recognition of Science and Technology) is an international robotics competition founded by inventor Dean Kamen in 1989. A nonprofit organization, FIRST [www.usfirst.org] was created to inspire and motivate students to excel and pursue careers in engineering, science and technology.

De-Evolution team members are committed to advancing understanding of robotics in middle and high schools and to spreading the message of the excitement, team spirit and intellectual stimulation that FIRST competitions provide. Any local schools wishing to start a robotics program are encouraged to contact De-Evolution to schedule a visit or demonstration.

On the Web: December's contest is 'Cutest Kid Photo'

December's On the Web photo contest theme is "Cutest Kid Photo." Submit yours today at DelMarTimes/net/Contests for a chance to win a prize.

Accomplished CCA robotics team visits R. Roger Rowe School



Canyon Crest Academy's robotics team De-Evolution demonstrates its robot's design and prowess to students in 5th through 8th grades at Rancho Santa Fe School on April 5 during Science Discovery Day. The team has achieved success at several regional and local competitions this year.

(Right) CCA Robotics team DeEvolution: Tristan Murphy, Ryan Lee, Nic Stone, Colin Murphy, Noah Sutton-Smolin, Yousuf Soliman; (Left) RSF School students get a close encounter with Robot T-Payne. Photos/Jon Clark

5 Team scoring and ranking application

It is difficult to deny a certain degree of randomness in the FTC competitions. While the competitions are fair and the sample size is large, the ranking points and qualifying points do not necessarily give a clear picture as to the difficulty of the competition or the overall effectiveness of a specific team.

In order to mitigate this problem, we have created our own method for ranking teams fairly based on how well they perform in each match. This allows us to provide a cleaner and easier ranking system for teams, which gives them due credit for the quality of their robot and design.

Let us assume, for instance, that two very good teams are up against one very good team and a moderately good team. Statistically speaking, it is more likely for the two very good teams to win the match. The current ranking point system accomodates this by only giving the winning teams ranking points. While this is, for the most part, fair and effective, there are certain cases under which teams are not necessarily given equal competition - or, even, that a team was lucky in their match selection.

With our scoring and ranking app, we mitigate this problem in a few critical ways. While the FTC competition's scoring measurements are fair and accurate, the program we designed gives better insight into how much specific teams are actually helping their alliance partners, and allows teams to make more clear decisions about who they may wish to pick during alliance selection.

The algorithm is composed of three steps:

1. The algorithm finds the average match score for each team
2. The algorithm then finds how much that average score was changed by a specific alliance
3. The algorithm then adds those together to produce a final score - the "power rank" for each team

These three steps combined do several unique things. However, the most important of these is that **the algorithm gives teams credit for how much they assist their alliance's overall score.** This means that if you help your alliance - even if you don't win the match - you're still credited for your assistance in that match.

In this way, it becomes irrelevant how well the players do in ther competition; rather, it simply becomes relevant how well the individual team does when contributing to their alliance score. This is the ultimate goal of the algorithm: instead of measuring the competition, measure the individual robot.

Over the course of the program's use and execution, we have noted several distinct cases where teams who are, in all honesty, quite good, do not end up as a match seed or even necessarily near the top - instead, they have lost many of their matches. It becomes clear when looking at match records that such teams are often pitted against the hardest teams to beat, and so the ranking points may not necessarily be the best representation of their success as an FTC team.

We do appreciate FIRST's method for ranking teams at the competiton, however. It's very straightforward, clear-cut, and promotes healthy and friendly competition. The method we use, however, is designed more so for fairness to the individual teams than it is for the actual rankings themselves.

We make the results of these analyses public. At each competition, we have had multiple teams ask to see our ranking results, simply because they would like to know where their team lies, or because they would like to gather a clearer picture of which teams may be better to pick. Either way, it would be unfair of us to keep these results to ourselves, so we make them available for broader use.

6 Code Appendices

The following code appendices detail our code, and a very high-level explanation of the code's intent and architecture.

Additionally, all of our code is available on Google Code.

6.1 Teleoperated code

```
#include "drivers/teleoputils.h"

void invokeButton(int button, bool pressed) {
        switch(button) {
                case JOY_X:  if(pressed) {setSpinMotor(100);}  else
                    {setSpinMotor(0);} break;
                case JOY_Y:  if(pressed) {setSpinMotor(100);}  else
                    {setSpinMotor(0);} break;
                case JOY_A:  if(pressed) {setSpinMotor(100);}  else
                    {setSpinMotor(0);} break;
                case JOY_B:  if(pressed) {setSpinMotor(100);}  else
                    {setSpinMotor(0);} break;

                case JOY_RB: if(pressed) {setArmMotors(50);}  else
                    {setArmMotors(0);} break;
                case JOY_LB: if(pressed) {setArmMotors(-100);} else
                    {setArmMotors(0);} break;
                case JOY_RT: if(pressed) {unlockArmMotors();} else {} break;
                case JOY_LT: if(pressed) {lockArmMotors();} else {} break;

                case JOY_R3: if(pressed) {} else {} break;
                case JOY_L3: if(pressed) {} else {} break;
                case JOY_ST: if(pressed) {} else {} break;
                case JOY_BA: if(pressed) {} else {} break;
        }
}

short btn = JOY_BTN;
void checkJoystickButtons() {
        if(btn == JOY_BTN) return;
        for(short i = 11; i >= 0; i--) {
                if((btn>>i) ^ (JOY_BTN>>i)) {
                        invokeButton(i, ((btn & (1 << i)) == 0));
                        btn ^= 1<<i;
                }
        }
}

void debugBlock() {
        nxtDisplayTextLine(0,"%d",rightEncoder);
        nxtDisplayTextLine(4,"%d",joystick.joy1_TopHat);
        int dirIR, strIR;
        HTIRS2readEnhanced(sensorIR, dirIR, strIR);
        nxtDisplayTextLine(1,"IR: %i", dirIR);
        if(dirIR == 5) {
                PlaySound(soundBeepBeep);
                setRightMotors(0); setLeftMotors(0);
                setArmMotors(0);
                while(1 == 1) wait1Msec(5);
```

```
        }
}

task main() {
        unlockArmMotors();
        clearEncoders();
        waitForStart();

        while(true) {
                getJoystickSettings(joystick);
                checkJoystickButtons();
                //debugBlock();

                setRightMotors((powscl(JOY_Y1)-powscl(JOY_X1)/1.1));
                setLeftMotors(0.8*(powscl(JOY_Y1)+powscl(JOY_X1)/1.1));
        }
}
```

In essence, our code has two sections: check to see if the buttons have changed, and set the power of the drive motors. This has the effect of keeping the speed of execution high - it doesn't run the button code every iteration, but only if the buttons have actually changed.

6.2 Autonomous

```
#include "autoconst.h"
#include "drivers/autoutils.h"
//#include "drivers/autodummy.h"

int OPT_SIDE = 0; int OPT_AUTO = 0; int OPT_DELAY = 0; int OPT_BRIDGE = 0;

void initializeRobot() {unlockArmMotors();}
void moveToBridge() {
        setLeftMotors(60*LEFT_POW_DIFF);
        setRightMotors(60*RIGHT_POW_DIFF);
        wait1Msec(3000);
        setLeftMotors(0);
        setRightMotors(0);
}

void runAutoLeft() {
        int irEncDist = -1;
             if(OPT_AUTO == 0) {irEncDist = rbtMoveToIR(C4_ENC, 6000) -
                 getEncoderByInches(IR_REALIGN); rbtMoveFdEnc(IR_REALIGN, 2000);}
        else if(OPT_AUTO == 1) {rbtMoveFdEnc(C1_ENC, 5000); irEncDist = C1_ENC;}
        else if(OPT_AUTO == 2) {rbtMoveFdEnc(C2_ENC, 5000); irEncDist = C2_ENC;}
        else if(OPT_AUTO == 3) {rbtMoveFdEnc(C3_ENC, 5000); irEncDist = C3_ENC;}
        else if(OPT_AUTO == 4) {rbtMoveFdEnc(C4_ENC, 5000); irEncDist = C4_ENC;}

        rbtArcRight(90);          //Turn to crate
        rbtMoveFdDist(4, 3000); //Against crate
        dumpArm();                //Dump blocks
        rbtMoveFdDist(-1, 1000);  //Back away

        if(OPT_BRIDGE == 0) OPT_BRIDGE = C23_THRESH < irEncDist ? 2 : 1;
        if(OPT_BRIDGE == 1) { //Left
                rbtArcLeft(90);
                rbtMoveFdEnc(irEncDist+getEncoderByInches(WHEELBASE+1), 6000);
```

```
                rbtArcLeft(-90);
                rbtMoveFdDist(18, 3000);
                rbtArcLeft(-90);
                rbtMoveFdDist(30, 4000);
        }
        if(OPT_BRIDGE == 2) { //Right
                rbtArcRight(-90);
                rbtMoveFdEnc(BRIDGE_ENC-irEncDist+getEncoderByInches(1), 6000);
                rbtArcRight(90);
                rbtMoveFdDist(18, 3000);
                rbtArcRight(90);
                rbtMoveFdDist(30, 4000);
        }
        if(OPT_BRIDGE == 3) //Back off
                rbtMoveFdDist(-24, 6000);
        //if(OPT_BRIDGE == 4); //None
}

void runAutoRight() {
        int irEncDist = -1;
             if(OPT_AUTO == 0) {irEncDist = rbtMoveToIR(C4_ENC, 6000) -
                 getEncoderByInches(IR_REALIGN); rbtMoveFdEnc(IR_REALIGN, 2000);}
        else if(OPT_AUTO == 1) {rbtMoveFdEnc(C1_ENC, 5000); irEncDist = C1_ENC;}
        else if(OPT_AUTO == 2) {rbtMoveFdEnc(C2_ENC, 5000); irEncDist = C2_ENC;}
        else if(OPT_AUTO == 3) {rbtMoveFdEnc(C3_ENC, 5000); irEncDist = C3_ENC;}
        else if(OPT_AUTO == 4) {rbtMoveFdEnc(C4_ENC, 5000); irEncDist = C4_ENC;}

        rbtArcLeft(-90);
        rbtMoveFdDist(4, 3000);
        dumpArm();
        rbtMoveFdDist(-1, 1000);

        if(OPT_BRIDGE == 0) OPT_BRIDGE = C23_THRESH < irEncDist ? 1 : 2;
        if(OPT_BRIDGE == 1) { //Left
                rbtArcLeft(90);
                rbtMoveFdEnc(BRIDGE_ENC-irEncDist+getEncoderByInches(1), 6000);
                rbtArcLeft(-90);
                rbtMoveFdDist(18, 3000);
                rbtArcLeft(-90);
                rbtMoveFdDist(30, 4000);
        }
        if(OPT_BRIDGE == 2) { //Right
                rbtArcRight(-90);
                rbtMoveFdEnc(irEncDist+getEncoderByInches(WHEELBASE+1), 6000);
                rbtArcRight(90);
                rbtMoveFdDist(18, 3000);
                rbtArcRight(90);
                rbtMoveFdDist(30, 4000);
        }
        if(OPT_BRIDGE == 3) //Back off
                rbtMoveFdDist(-24, 6000);
        //if(OPT_BRIDGE == 4); //None
}

void optionScreen() {
        nxtDisplayTextLine(0, "NXT:  %.2f V", ((float)nAvgBatteryLevel)/1000.0);
        if(externalBatteryAvg > 0) nxtDisplayTextLine(1, "EXT: %.2f V",
            ((float)externalBatteryAvg)/1000.0);
                else nxtDisplayTextLine(1, "EXT: OFF");
```

```
if(nAvgBatteryLevel < NXT_LOW_BAT) nxtDisplayTextLine(2, "***NXT
    LOW***");
if(externalBatteryAvg < EXT_LOW_BAT) nxtDisplayTextLine(2, "***    EXT
    LOW***");
if(nAvgBatteryLevel < NXT_LOW_BAT && externalBatteryAvg < EXT_LOW_BAT)
    nxtDisplayTextLine(2, "***NXT EXT LOW***");

while(nNxtButtonPressed != BTN_CENTER) { // SIDE: Left | Right | Bridge |
    None
             if(OPT_SIDE == 0) nxtDisplayTextLine(3, "SIDE: Left");
        else if(OPT_SIDE == 1) nxtDisplayTextLine(3, "SIDE: Right");
        else if(OPT_SIDE == 2) nxtDisplayTextLine(3, "SIDE: Bridge");
        else if(OPT_SIDE == 3) nxtDisplayTextLine(3, "SIDE: None");

        if(nNxtButtonPressed == BTN_LEFT || nNxtButtonPressed ==
            BTN_RIGHT) {
                PlaySound(soundShortBlip);
                if(nNxtButtonPressed == BTN_LEFT) OPT_SIDE--;
                if(nNxtButtonPressed == BTN_RIGHT) OPT_SIDE ++;
                if(OPT_SIDE > 3) OPT_SIDE = 0;
                if(OPT_SIDE < 0) OPT_SIDE = 3;

                while(nNxtButtonPressed == BTN_LEFT || nNxtButtonPressed
                    == BTN_RIGHT) wait1Msec(5);
        }
} PlaySound(soundShortBlip); while(nNxtButtonPressed == BTN_CENTER)
    wait1Msec(5);

if(OPT_SIDE != 3) // DELAY: 0 - 25000
        while(nNxtButtonPressed != BTN_CENTER) {
                nxtDisplayTextLine(4, "DELY: %i", OPT_DELAY);
                if(nNxtButtonPressed == 1 || nNxtButtonPressed == 2) {
                        PlaySound(soundShortBlip);
                        if(nNxtButtonPressed == 2) OPT_DELAY -=
                            (time1[T1] < 200 ? 5000 : 1000);
                        if(nNxtButtonPressed == 1) OPT_DELAY +=
                            (time1[T1] < 200 ? 5000 : 1000);
                        if(OPT_DELAY < 0)     OPT_DELAY = 25000;
                        if(OPT_DELAY > 25000) OPT_DELAY = 0;

                        while(nNxtButtonPressed == BTN_LEFT ||
                            nNxtButtonPressed == BTN_RIGHT) wait1Msec(5);
                        ClearTimer(T1);
                }
        } if(OPT_SIDE != 3) PlaySound(soundShortBlip);
            while(nNxtButtonPressed == BTN_CENTER) wait1Msec(5);

if(OPT_SIDE < 2) // AUTO: IR | Crate 1 | Crate 2 | Crate 3 | Crate 4
        while(nNxtButtonPressed != BTN_CENTER) {
                if(OPT_AUTO == 0) nxtDisplayTextLine(5, "AUTO: IR");
                else nxtDisplayTextLine(5, "AUTO: Crate %i", OPT_AUTO);

                if(nNxtButtonPressed == BTN_LEFT || nNxtButtonPressed ==
                    BTN_RIGHT) {
                        PlaySound(soundShortBlip);
                        if(nNxtButtonPressed == BTN_LEFT) OPT_AUTO--;
                        if(nNxtButtonPressed == BTN_RIGHT) OPT_AUTO++;
                        if(OPT_AUTO > 4) OPT_AUTO = 0;
```

```
                                if(OPT_AUTO < 0) OPT_AUTO = 4;

                                while(nNxtButtonPressed == BTN_LEFT ||
                                    nNxtButtonPressed == BTN_RIGHT) wait1Msec(5);
                        }
                } if(OPT_SIDE < 2) PlaySound(soundShortBlip);
                    while(nNxtButtonPressed == BTN_CENTER) wait1Msec(5);

        if(OPT_SIDE < 2) // BRIDGE: Closest | Left | Right | Back up | None
                while(nNxtButtonPressed != BTN_CENTER) {
                            if(OPT_BRIDGE == 0) nxtDisplayTextLine(6, "BRDG:
                                Closest");
                        else if(OPT_BRIDGE == 1) nxtDisplayTextLine(6, "BRDG:
                            Left");
                        else if(OPT_BRIDGE == 2) nxtDisplayTextLine(6, "BRDG:
                            Right");
                        else if(OPT_BRIDGE == 3) nxtDisplayTextLine(6, "BRDG:
                            Back up");
                        else if(OPT_BRIDGE == 4) nxtDisplayTextLine(6, "BRDG:
                            None");

                        if(nNxtButtonPressed == BTN_LEFT || nNxtButtonPressed ==
                            BTN_RIGHT) {
                                PlaySound(soundShortBlip);
                                if(nNxtButtonPressed == BTN_LEFT) OPT_BRIDGE--;
                                if(nNxtButtonPressed == BTN_RIGHT) OPT_BRIDGE++;
                                if(OPT_BRIDGE > 4) OPT_BRIDGE = 0;
                                if(OPT_BRIDGE < 0) OPT_BRIDGE = 4;

                                while(nNxtButtonPressed == BTN_LEFT ||
                                    nNxtButtonPressed == BTN_RIGHT) wait1Msec(5);
                        }
                } if(OPT_SIDE < 2) PlaySound(soundShortBlip);
                    while(nNxtButtonPressed == BTN_CENTER) wait1Msec(5);

        nxtDisplayTextLine(7, "*** LOCKED ***");
}

task main() {
        initializeRobot();
        optionScreen();
        waitForStart();
        wait1Msec(OPT_DELAY);
             if(OPT_SIDE == 0) runAutoLeft();
        else if(OPT_SIDE == 1) runAutoRight();
        else if(OPT_SIDE == 2) moveToBridge();
        lockdownRobot();
}
```

The autonomous program has a few relevant steps. The first is to display the option screen for the robot. This is a user interface control which allows us to dynamically assign various parameters for autonomous. Namely, we can select our side (or simply indicate we want to go to the bridge), assign a delay to startup (for inter-team coordination), select whether to go to a specific crate or the IR beacon, and finally which side of the bridge the code should go on (closest, right, left, or none).

The code has been generically created so that no matter which options we select, the code will execute and follow through properly.

6.3 Drivers and utility files

There are three utility files in our archive: `sharedutils.h`, `teleoputils.h`, and `autoutils.h`. Additionally, the file `autodummy.h` is a temporary autonomous file used for visualizing the output from autonomous without actually running any motors.

Additionally, we created several utility files which are used before and after matches, and for diagnostics and testing purposes.

6.3.1 The shared utility file

```
#ifndef __SHAREDUTILS__
#define __SHAREDUTILS__

#include "JoystickDriver4278.c"
#include "hitechnic-irseeker-v2.h"
#include "wiringnxt.h"

#define setLeftMotors(x)  {motor[mLeft1]  = x; motor[mLeft2]  =  x;}
#define setRightMotors(x) {motor[mRight1] = x; motor[mRight2] =  x;}
#define setArmMotors(x)   {motor[mArm1]   = x; motor[mArm2]   = -x;}
#define setSpinMotor(x)   {motor[mSpin]   = x;}
#define lockArmMotors()   {servo[servoL1] = 155; servo[servoL2] = 20;}
#define unlockArmMotors() {servo[servoL1] = 120; servo[servoL2] = 70;}

#define leftEncoder     abs(nMotorEncoder[mArm2])
#define rightEncoder    abs(nMotorEncoder[mArm1])
#define clearEncoders() {nMotorEncoder[mArm1] = 0; nMotorEncoder[mArm2] = 0;}

//Distance Macros
#define INCH   1.0
#define CM     0.3937
#define MM    39.370
#define YARD  36.0
#define FOOT  12.0
#define METER 39.370

#define WHEELCIRC 12.566
#define WHEELBASE 15.309
#define FLOORMAT  24.0

#define BTN_CENTER 3
#define BTN_LEFT   2
#define BTN_RIGHT  1
#define BTN_BACK   0

#define LEFT_POW_DIFF 0.563
#define RIGHT_POW_DIFF 1.0

void waitForStart() {
  while(true) {
    getJoystickSettings(joystick);
    if(!joystick.StopPgm) break;
  }
}

#endif //__SHAREDUTILS__
```

You may notice that most of this code is defined by `#define` statements. This is intentional - these values are intended to be used as conversions and small macros. For instance, wherever the compiler sees `setSpinMotor(float power)`, it will automatically replace it with the command `motor[mSpin] = power;`. This way, we minimize the amount of clutter in our code and maximize its readability.

6.3.2 The teleop utilities file

```
#ifndef __TELEOPDRIVER__
#define __TELEOPDRIVER__

#include "sharedutils.h"

//Allows threshold to be defined in teleop-file
#define THRESH 10.0
#define MINX   10.0
#define SLOPE   0.5
#define DISTA   0.6

float powscl(int xz) {
        float sign = (float)sgn(xz);
        float x = abs(xz)/128.0;
        if(x < DISTA) {return 100 * sign * (x*SLOPE);}
                else {return 100 * sign * ((DISTA*SLOPE*(x-1.0) - x + DISTA) /
                    (DISTA - 1.0));}
}

//Controller 1 - Left Joystick - Linear
#define JOY_X1 (abs(joystick.joy1_x1) > THRESH ? joystick.joy1_x1 : 0)
#define JOY_Y1 (abs(joystick.joy1_y1) > THRESH ? -1.0*joystick.joy1_y1 : 0)

//Defines current button map layout
#define JOY_X 0
#define JOY_Y 3
#define JOY_B 2
#define JOY_A 1

#define JOY_RB 5
#define JOY_LB 4
#define JOY_LT 6
#define JOY_RT 7

#define JOY_R3 10
#define JOY_L3 11

#define JOY_ST 9
#define JOY_BA 8

#define JOY_BTN joystick.joy1_Buttons

int getLeftPowTopHat(int topHat) {
        if(topHat == 0) return 100;
        if(topHat == 6) return -100;
        if(topHat == 4) return -100;
        if(topHat == 2) return 100;
        return 0;
}
```

```
int getRightPowTopHat(int topHat) {
        //topHat--;
        if(topHat == 0) return 100;
        if(topHat == 6) return 100;
        if(topHat == 4) return -100;
        if(topHat == 2) return -100;
        return 0;
}

#endif //__TELEOPUTILS__
```

This file is intended to expand upon the `sharedutils.h` file. It includes definitions for the buttons, so that instead of checking button 5, we check button `JOY\_RB`; it keeps everything cleaner and more logical. Additionally, we created a power scalar function to more easily transition from ranges of high power to those of low power in movement.

6.3.3 The autonomous utilities file

```
#ifndef __AUTODRIVER__
#define __AUTODRIVER__

#include "sharedutils.h"

#define DRV_TIMER T3
#define MAX_TURN_TIME 3000
#define PAUSE_TIME 160

void pause() {wait1Msec(PAUSE_TIME);}
void pause(int n) {for(int i = 0; i < n; i++) pause();}
void estop() {StopAllTasks();}

int getEncoderByInches(float inches) {return floor((1440)*(inches)/WHEELCIRC);}
float getInchesByEncoder(int encode) {return (((float)encode)/1440.0)*WHEELCIRC;}

void dumpArm() {
        //PlaySound(soundBlip);
        setArmMotors(50);
        wait1Msec(1550);

        //PlaySound(soundBlip);
        setArmMotors(0);
        wait1Msec(400);

        //PlaySound(soundBlip);
        setArmMotors(-50);
        wait1Msec(1100);

        //PlaySound(soundBeepBeep);
        setArmMotors(0);
}

void lockdownRobot() {
        setLeftMotors(0);
        setRightMotors(0);
        setArmMotors(0);
        setSpinMotor(0);
        unlockArmMotors();
```

```
        while(true) wait1Msec(5);
}

int rbtMoveToIR(int max, int timeout) {
        int dirIR, strIR; float stopRightEnc;
        HTIRS2readEnhanced(sensorIR, dirIR, strIR);
        clearEncoders();

        ClearTimer(DRV_TIMER);
        while(dirIR != 5 && rightEncoder < max) {
                HTIRS2readEnhanced(sensorIR, dirIR, strIR);
                stopRightEnc = rightEncoder;
                if(dirIR != 5) {setLeftMotors(40); setRightMotors(40);}
                if(time1[DRV_TIMER] > timeout) lockdownRobot();
        }
        setLeftMotors(0); setRightMotors(0); pause(3);
        return rightEncoder;
}

void rbtMoveFdDist(float inches, int msec) {
        clearEncoders();
        int enc = abs(getEncoderByInches(inches));
        int norm = 1.0*sgn(inches);
        ClearTimer(DRV_TIMER);
        int lEnc = leftEncoder; int rEnc = rightEncoder;
        while(abs(lEnc) < enc && abs(rEnc) < enc) {
                if(time1[DRV_TIMER] > msec) lockdownRobot();
                lEnc = leftEncoder; rEnc = rightEncoder;
                setLeftMotors (100.0*norm*LEFT_POW_DIFF);
                setRightMotors(100.0*norm*RIGHT_POW_DIFF);
        }
        if(time1[DRV_TIMER] > msec) lockdownRobot();
        setLeftMotors(0); setRightMotors(0); pause();
}
void rbtMoveFdEnc(int enc, int msec) {rbtMoveFdDist(getInchesByEncoder(enc),
    msec);}

void rbtArcLeft(float degs) {
        int enc = getEncoderByInches((2.0*PI*WHEELBASE)*(abs(degs)/360.0));
        clearEncoders();
        setLeftMotors(-1*sgn(degs)*90);
        ClearTimer(DRV_TIMER);
        while(leftEncoder < enc) if(time1[DRV_TIMER] > MAX_TURN_TIME)
            lockdownRobot();
        setLeftMotors(0); pause();
}

void rbtArcRight(float degs) {
        int enc = getEncoderByInches((2.0*PI*WHEELBASE)*(abs(degs)/360.0));
        clearEncoders();
        setRightMotors(sgn(degs)*60);
        ClearTimer(DRV_TIMER);
        while(rightEncoder < enc) if(time1[DRV_TIMER] > MAX_TURN_TIME)
            lockdownRobot();
        setRightMotors(0); pause();
}

void rbtTurnRight(float degs) {
        int enc = getEncoderByInches((PI*WHEELBASE)*(abs(degs)/360.0));
```

```
        clearEncoders();
        setLeftMotors( -1*sgn(degs)*40);
        setRightMotors(sgn(degs)*30);
        ClearTimer(DRV_TIMER);
        while(rightEncoder < enc) if(time1[DRV_TIMER] > MAX_TURN_TIME)
            lockdownRobot();
        setLeftMotors(0); setRightMotors(0); pause();
}

void rbtTurnLeft(float degs) {
        int enc = getEncoderByInches((PI*WHEELBASE)*(abs(degs)/360.0));
        clearEncoders();
        setLeftMotors(sgn(degs)*60);
        setRightMotors(-1*sgn(degs)*60);
        ClearTimer(DRV_TIMER);
        while(leftEncoder < enc) if(time1[DRV_TIMER] > MAX_TURN_TIME)
            lockdownRobot();
        setLeftMotors(0); setRightMotors(0); pause();
}

#endif //__AUTODRIVER__
```

The autonomous utilities file is designed to make the logic flow easily. Recall that statements in our autonomous program come in the form of `rbtTurnLeft();`, not a sequence of commands with the same effect. That is, in essence, the entire purpose of this file.