Engineering and Design Journal

FTC Team 4278 de.evolution

Abstract

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 against 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.

Additionally, the code which goes into this design will be explained.

Contents

1 Robot Design and History Overview

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

Over the past four years, our design strategy has changed significantly. 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, but in retrospect, much of that was simply sheer luck. 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 slightly. Though we still won several awards for placing third in local competitions, we did not perform to our expected standard. 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 forseen 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 lifing objects off the ground. However, in the usual spirit of our team, 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 compoments 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. This, we decided, is doable.

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 TODO: Final Design Overview

1.4 Wheelbase and Arm Design

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, and as a team we decided to take an aggressive approach towards our first couple of regional competitions in order to secure a place in St. Louis. Because of this, 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 with the field control issues that we were issues. 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.

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 incredible 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-esque 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 Autonomous STUFF

	Crate 1	Crate 2	Crate 3	Crate 4
Object	0/0	0/0	0/0	0/0
Nobjct	0/0	0/0	0/0	0/0

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.

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 premit the use of both controllers. Therefore, we must be able to accomodate 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.

3.2 Autonomous Mode

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:

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