Basic Information

Team Number
1736

Team Name
Robot Casserole

Programming Language

What programming language do you use?
Java

Public Code

Is your code public?
Yes

What is your team's GitHub account?
https://github.com/RobotCasserole1736

Vision

What camera do you use?
Homemade Raspberry PI-based solution

What do you like about your camera?
Fully customizable, ground-up integrated design around 2020's unique challenges. Historically we've used JeVois with good success, but the usb connector and weirdness of interfacing made it a less than awesome solution (ethernet FTW). Getting the far shot required good accuracy, which I believe we were able to achieve with a solution designed around high resolution (at the cost of slightly higher latency)

What do you dislike about your camera?
Required specialized knowledge to get functional.

How would you compare this camera to other cameras you've used in the past?
Best solution we've used so far!

If you could do vision differently, what would you change?
pick a slightly different mounting location. Shooter wheel vibrations caused weird artifacts sometimes.

How are you planning to do vision next year?
Maybe a limelight, maybe something custom. Depends on where we end up at this summer team-skills wise, and what the game requires.

Path Planning

How do you design your paths?
Hand-selected waypoints

What library(ies) do you use to make your paths?
Jaci's Pathplanner (V1)

What geometry do you use for your paths? (Splines, piecewise, circle, etc)

erm.... whatever Jaci's thing spits out. It's a total black box to me.

What forms of path planning have you done in the past and if you have changed them, why?
Used to use falcon pathplanner. It's got a lot of quirks, but we knew how to deal with them. On a whim we tried updating to Jaci's this year, and it went well.

How do you integrate sensors with your paths?
We do the cheat method - pathplananer generates left and right wheel velocity profiles. We interpolate these over time to set the desired velocity of each side of the drivetrain. This means we're using encoders as the primary feedback sensor. In addition, we use a gyro along with pathplanner-generated "desired heading" values to offset the wheel velocities - a simple P controller to keep us pointed in the right direction. Really, it just helps compensate for wheel scrub dynamics that the pathplanner doesn't have built in. Totally not the optimal solution, but hey, we can get accurate placement within an inch or two out of it, so who cares?

If you could do path planning differently, what would you change?
Would like to try ramaste at some point, whenever we have time. This solution has been working just fine for us for now though, so no real complaints. Ya don't _need_ ramaste to score points.

<u>Training</u>
How do new programmers get trained?
Fall off season training. Lots of presentations, re-coding old robots, simulators, general chit-chat. Is "osmosis" a valid answer?

Do new programmers have to work outside of practice?
Only if they want to, we'll keep supplying projects till we''re blue in the face.

What is your general training order? (ex. Classes, functions, data types, reading documentation)
Totally random and varies year to year. Ideally it's based on what student already know vs. need help on. There's also a huge element of "hey this is relevant today, let's chat about it".

What do you do if there is not enough work for all the programmers?
Spend time at home writing software. I like writing software.

On average, how many programmers do you have?
5-10, depends on the night.

<u>GitHub</u>
How do you control access to the team GitHub?
we have an organization and add students to one or more teams. Generally it's split between "admin" and "dev" permissions.

How do you delegate using GitHub?
Mentors have all the admin account passwords, but students generally get access on an as-need basis.

How do you handle merge issues and multiple people working on the same file with GitHub?

FYI: Git is the tool that is used for doing this (Github is just the server repository with some extra bells & whistles). Someone with experience is responsible for doing the merge and usually collaborates with the contributors. Also, with proper software architecture, and by carefully sequencing the order in which tasks are completed, and doing one task per branch, the majority of the nastiness can be eliminated.

If your team uses private repositories: what are the advantages of this/why did you start doing it?
We generally keep "this year's" repo private until we reveal the robot, and feel comfortable having people look at our code and ask questions about it. This is partially to help keep some of our development "proprietary" in the early part of the season (though we'd readily give out info to anyone who asks), and mostly just because we wouldn't want someone referencing code that we have no proof actually works.

How does your team make ReadME.md documents?
Usually it's just an activity that's part of the software dev. The student gets to learn how to write a "front page" summary to entice people and make them want to read more.

<u>Other Sensors</u>
What other types of sensors do you use?
Whatever type of sensor the game requires! Encoders are the most common. I'm a huge fan of the REV through-hole hex shaft mounted model. Gyro is almost always a given, as is an analog pressure sensor if we've got a pneumatic system.

How do these sensors help your robot?
Sensors serve two purposes: #1 is part of a closed-loop feedback system, #2 is data gathering to help measure robot performance.

Of those sensors, which are you planning to use again in the future (if any)?
Any and all of them! We would want to use as few as possible to get the job done on the robot (fewest points of failure). But, as soon as the robot requires one, we'll put one on there!

How do you learn what new sensors to try and how to use them?
Online research, new product announcements, ChiefDelphi.

<u>Off Season</u>
What do you do in the offseason to prepare for build season?
Teach programming projects, do formal classroom learning, participate in team "pet" projects for learning purposes.

How does programming interact with mechanical for off season activities?
It's more separate than during the build season, but part of the training process involves "find projects to force interaction", so students on both sides get experience "bridging the worlds".

<u>Documentation</u>
How do you document your code?
github wiki, comments, tribal knowledge.

Have you documented differently in the past? What do you like better now vs then?

Nope, admittedly we don't track much year to year. Lots of the documentation is more of "hey, we did something similar in year XXXX, reference that and make something similar". We rely far more on WPILIB's documentation than our own, as we're usually creating from scratch.

Build Season

What do your programmers do at the start of build season?
First priority: Participate in game analysis and design discussions.
Second priority: Set up base robot project and add in data logging, website utilities, basic drivetrain code.... whatever we know _for sure_ will be needed.
Third priority: Start prototyping sensor & vision solutions

How useful are the tasks that they do at the start of build season? (from 1 - 10)
10

How much time does programming get to program the robot (without mechanical intervention)?
Varies year to year. Usually we set expectations that we get about ten hours total, spread out over many days.

How do you divide up the time programming gets on the robot between different mechanisms, tuning, and autonomous?
Ground-up. First priority is always basic functionality - making mechanisms move based on operator commands. This includes PID & other tuning required for arm or manipulator mechanisms. Second priory is autonomous - tuning drivetrain PID's and tweaking path timings and sequencing. Third priority is optimization - returning and tweaking the mechanisms, changing the software to push the robot to its mechanical limits.

During programming's time on the robot, how does your team handle mechanical failures and imperfections?
Usually both teams are present, meaning that minor repairs can be done quickly. We do rotating time on the robot -software development is usually 30 seconds of actually moving the robot, followed by 5 minutes of redesign and software writing. Those 5 mintues are where mechanical and electrical hop in to make repairs or improvements. We work together to coordinate larger tasks, occasionally scheduling extra meetings to ensure the robot is always left in a state where the other team can make progress.

How do you make the schedule for programming?
We string it around the robot's schedule. If drivetrain is due to be completed (mechanical+electrical) on date XX, we ensure we have a software release that can run a drivetrain by date XX. From there, we work backward to figure out all the parts that go into it.Since not every student is around every night, it's hard to put hard deadlines on particular students and tasks. Usually it's just a "well, here's top priority, if no one's working it you get to jump in on it!". This is also why we encourage regular attendance - if we know when you're gonna be here, we can better schedule tasks and avoid handing your work off to someone else.

How does your team use gearbox ratios with encoder counts?
We make constants and utility functions to do conversion between different shaft speeds. In code we always use the constant or utility to transform one shaft speed into another. Then, if the gearbox ratio changes, the constants file gets updated.

How does your team define code standards?

Standard 0: Communicate. Talk about ideas, ask about implementation, review your code with someone experienced before merging it.
Standard 1: Make it work
Standard 2: Make it work well
Standard 3: Make the code look nice (auto-formatter is good)
Standard 4: Good architecture makes things nice. Experienced students and mentors create the sandboxes for people to play in, then any student can play within their sandbox. As soon as you step outside your sandbox, you know it's a conversation that has to be had with others.

Creating from Scratch vs Inheritance

How does your team balance inheriting WPILib functions with writing custom functions?
Generally bias toward "already written". However, the underlying requirement is just "get the robot functional as fast as possible". Pick per-written or custom depending on which you believe will achieve that requirement the best.

What are some examples of custom functions that your team has made?
Custom webserver for calibration & data viewing. roboRIO memory/CPU load tracking classes. Integral/derivative/Filter classes. Autonomous event timeline-style sequencer

Interesting WPILib Functions

Are there any WPILib functions that are unusual and make your life a lot cooler or easier?
Really the biggest thing is the underlying HAL interface to the roboRIO's hardware. It's so hard to get right on your own, they took the vast majority of the hard part out of software development.

What class do you use for joystick control?
XBOX controller, with a wrapper to translate concepts of "X button pressed" to "arm extension commanded".

What class do you use for automating actions?
Hmmmm, I"m not sure what "Automating" means in this context.

Joystick Layout

Who determines the layout of the joystick for your team?
Software team takes a first pass at it, driveteam evaluates it and proposes changes as needed.

How do you manage changes to the joystick layout?
Only driveteam is allowed to request changes after initial development. Changes are clearly communicated to them as soon as they go on the robot. In general, it's important to spend a good amount of time prior to handing it off to drive team to ensure the control scheme is reasonable, to minimize drastic or frequent changes.

How do you test the joystick layout?
Same as we test the rest of the robot - just rack up hours of driving the robot with driveteam at the helm.

PID Tuning

When you get the robot, what is the first thing your programming team does with it?
Put the robot up on blocks, and make sure all the hardware is connected and communicating with the software properly. Fix electrical issues or port constants as needed.

How does your team determine if motors should have encoders or not?
It depends on the mechanism the motor is hooked up to. Is "guessing at voltage" a good enough control strategy to make the mechanism move as desired? Or is finer control required?

When you PID tune a motor for position control, what is your procedure?
Arbitrary feed-forward first. Then P. Then D or I, depends on the system. See https://trickingrockstothink.com/blog_posts/2019/10/26/controls_supp_arm.html

When you PID tune a motor for velocity control, what is your procedure?
Feed forward, P, D, I. See https://trickingrockstothink.com/blog_posts/2019/10/19/tuning_pid.html