Programming Struggles and Challenges

Overview: This document will describe all the programming challenges, struggles, and issues that Tobor's programming team encountered during this Tobor programming season. It will not only cover the difficulties I faced but also outline the solutions that were found to overcome them.

Overall Biggest Challenge: I recently had a new experience where I had to learn and solve problems independently as the only programmer for Tobor. While I could still take input from the team, I had to approach problem-solving differently. Although I enjoy figuring things out, being the only returning programmer presented a challenge as I had to solve problems alone. If I were to identify my biggest struggle, it would be the new experience of programming independently this year and stepping up the leadership role on the team.

1. SDK issues
   1.1. Android Studio does not allow me to create a Java class under team code.
   1.2. Android Studio does not pre-write my package name along with the head class for the project.
   1.3. When running the code, the SDK gives a strange error relating to the team code. There is nothing wrong with the code under the team code. I can not upload any code to the robot until this is fixed.

   Problem 1.3 solved: this problem was solved by doing the same method explained below, I copied and pasted the team code folder to make sure nothing would be lost. Then I reopened the software development kit and pasted it into the team code. It worked afterward.

   Issue 1 resolved: *opening the SDK was done incorrectly, try opening the SDK by opening "FtcRobotController-master," this project should have the green android mascot on the file, this shows that the file is good to open. If any further errors are held with the SDK try redownloading it and reopening it.*

2. Organizing the code
   2.1. Using both LinearOpMode and OpMode in one Java class

   Struggle 2.1 resolved: *this was a struggle at first, but I later resolved this struggle by extending LinearOpMode and creating an interface with multiple required methods that I can call within the runOpMode method. This way I am using LinearOpMode to make the code layout look like both LinearOpMode and OpMode.*

2.2. Can not call dcmotor, telemetry, hardwareMap, or any other hardware without extending a class.

Issue 2.2 resolved: *It took me a little while to figure this one out, but finally, I found out that if I wanted to use hardware Map in a class that had no extension, I could create a method that has a 'HardwareMap' type inside, like this: public static void mapMotors (HardwareMap mapHardware, String frontLeftMotor, String frontRightMotor, String backLeftMotor, String backRightMotor). This way when I call the hardware map from the Java class that does extend, I can call it like so: mapMotors (hardwareMap, "fr","fl","br","bl");.*

2.3. Outputs, Inputs, and In-Out

Struggle 2.3 resolved: As the code was created it was found to be difficult to figure out where to begin organization-wise. I knew everything needed to be organized. I thought about how everything could be divided. Any technology can be divided into inputs and outputs, I thought, so that is what I did. I decided to separate the inputs from the outputs. I made an output folder and an input folder to hold inputs and outputs. Later an "inOut" folder was added, this way if inputs and outputs are in the same java class they can be put in this folder. I only had one input and output class at one point in time. Later, as the code expanded, I found a larger use for the inOut folder. Now it holds many inputs and outputs combined, such as TeleOp and autonomous functions.

2.4. Uploading the entire SDK team code to Git Hub cannot be done, because the file is too big.

Issue 2.4 resolved: Uploading big files such as the edited SDK team code to your GitHub repository cannot be done by simply dragging the files. It must be done from Android Studio itself. To resolve this problem, I looked up how to upload code from Android Studio to Git Hub. Before using Android Studio to upload code to GitHub, git must be installed on your computer, and then you can push your code to your GitHub repository. When managing code in GitHub, I found it is best to use Android Studio and Git to push, pull, clone, comment, manage version control, and more! I have not found out all about these helpful tools, but I will investigate learning more about GitHub and Git.

2.5. Allowing other GitHub users to edit code.

Struggle 2.5 resolved: this was a bit of a struggle at first, but I found out that to do this you have to go to permissions and select what account you wish to allow you to edit the files.

2.6. Managing branches

Struggle 2.6 resolved: At first this seemed to be a struggle for me, but later I realized the problem was with the GitHub account I was using. I was using my personal GitHub account when the Tobor account was the one that created the repository. To rename or edit the default branch I had to be using the Tobor account, the account that created the repository. With my account, I can still edit and delete sub-branches, but just not the default branch. I need to use the team account to do that.

2.7. Tobor Programming Website

Project 2.7 resolved: A phone number was created for team 535 Tobor. Therefore, it is now possible to call or text Tobor. The texts get forwarded to Tobor's email and I think the missed calls and voicemails do as well. I was having a little bit of trouble trying to figure out how to add an HTML link to the website where when a link is clicked the user's phone redirects the phone to the texting app to text Tobor. It was later figured out to use the keyword 'SMS'. This is how the HTML was written:

```
<a href="sms:+17652678525">
```

Struggle 2.7 resolved: After working with this document for a while and noting down struggles and challenges that were faced, I realized that there wasn't any way to strongly archive it. That is what gave me the idea. Tobor has its website for the entire team, but why shouldn't the programming category of Tobor have a programming website specifically for programming information on Tobor? Sense GitHub offers the ability to program a website for free by using a GitHub username and github.io domain name, and Tobor Programming is the only one that uses Tobor's GitHub account, I used the GitHub website creator. (https://535tobor.github.io/CenterStage/)

3. Test robot issues.

3.1. Non-responding encoders

Issue 3.1 resolved: *this issue was rather ridiculous, I kept trying to run the test robot encoders but later I realized that the encoders were not plugged in. (this happened 2xs. I learned my lesson the second time.)*

3.2. The test robot slowly moves leftward when told to move forward, it moves forward for a little while but slowly moves leftward after a while.

Issue 3.2 resolved: *mechanical fixes will not be performed since the test bot is not what the team is using for this season's competition, however possible programming fixes could be obtained. For example, if the robot is slowly moving left, maybe put more power on the left wheels to make the robot turn right again.*

4. Mecanum Drive

4.1. Mecanum Drive is not working properly at all.

Issue 4.1 resolved: *definitions in the code were not set correctly, I told the code to set the front left wheel to the front right power and set the back left wheel to the back right power. I changed that and the robot worked a lot better.*

4.2. Mecanum Drive program strafing left when it should strafe right, cannot find any error with the code.

Issue 4.2 resolved: *To fix this issue I finally found where in my code I was accidentally calling the wrong motors. What I had going wrong was I had a variable of a wheelset to the wrong wheel somewhere in the code. If someone looks over all their code, the code is good, but everything is not working correctly still then take it back to the issue. If no error is showing on the driver's station, but the robot is not doing as expected it could be incorrectly configured. An incorrect configuration was also my issue in this case. The wheel direction should also be checked, but in my case, that was not the problem, the problem was I was calling the wrong motors.*

5. Working From Home
   5.1. Setting up the Tobor Google email so that I can work with it from home. **Not yet resolved.** Later in this season, this has not been found necessary, (or at least not a high priority) GitHub can be used by cross accounts.
   5.2. Defining distance.
      5.2.1. Motor encoders
      5.2.2. Free spinning wheel odometry

Challenge 5.2 solved: To solve both 5.2.1 and 5.2.2 I had the robot move a specific distance (a specific amount of encoder ticks). After doing this I took the number of ticks it moved divided by the distance it went. That number I put in a variable called, "one inch" (or something like that). That variable I used to move inches in different methods. If I want to have the robot move 5 inches instead of one I put five inches in and inside the java method it multiplies 5 by the one-inch value that was previously calculated.

6. Team Prop and Pixel Detection
   6.1. What are the best objects to scan when using AI?

Challenge Question 6.1 solved: I found out from doing research that when using tensor flow AI, the best objects to use for a team prop are complicated. The more complicated the shape the better. If you can add complicated colors too that will be even better. This is because AI needs something unique apart from the rest of the field to recognize what it is.

   6.2. The AI is saying that the edge of the laptop is a pixel.

Issue 6.2 solved: The AI technology is not meeting my accuracy expectations. This may be because the pixels are created with basic colors and shapes, limiting their capabilities. However, there is a solution to this problem - using team props can boost your move points and allow for more control over the game.

6.3. After training the team's team prop it detects good, but not as good as I hoped.

Issue 6.3 solved: I have decided to switch to using a color/distance sensor instead of AI for detecting the team prop, as you believe that AI has its limitations. I plan to use a distance sensor to determine whether the robot sees the team prop and knows how to get it working efficiently. I believe that if the team can get points that way, then it's worth going for it. Another reason I decided to switch is that FIRST only gives a team a limited amount of training time, and I don't want to waste that time on training repeatedly if the AI has trouble detecting the team prop due to color issues. Last year, the AI didn't work well either, so this year I was hoping to get it working better, but it didn't turn out as perfect as I wanted. Therefore, I have decided to switch to the distance solution, which I believe will be more reliable than using a color sensor due to color variations and lighting conditions.

...But now after using the distance sensor and mounting the camera to the robot with a camera case, it was thought that the camera and distance sensor could detect the team prop for higher chances of scanning the prop. If the camera does detect the prop check with the more reliable operation. The camera will be used for April Tags.

7. Robot Not Moving Straight!

7.1. TeleOp not moving straight.

Issue 7.1 solved: This doesn't need high attention due to joysticks being able to straighten out the movements of the robot. So, I can fix this later, however it isn't that urgent.

7.2. Autonomous not moving straight.

Issue 7.2 solved: This is more of an issue. If the robot cannot go straight in autonomous, the code can not work reliably. I fixed this by using a setVolocity command instead of using a setPower command. What this is doing is setting the speed of each motor instead of sending power or raw voltage. Sending the power can be unreliable, however, using speed is more consistent. I also will be using ticks with encoders and odometers.

8. Arm, Lift/Shaft, and claw are not responding to code!

Issue 8 solved: Just as of 3.1 the reason for this issue was because for some reason the motors got unplugged. It was too bad because we were late to the match and the pixels never did get on the board. I keep telling myself that if only I had figured out that the motors were not plugged in, then we might have won that match. We were one match away from winning and going to state that way.

9. Robot shaking dramatically.
   Issue 9 thought through: This issue isn't technically resolved yet; however it has been thought through, and the issue has been thought to be the belt for the wheel motors has been getting loose and causing the motor encoders to not get the correct measurements.

10. Odometry does not give accurate measurements.
    Problem 10 solved: The solution here was found to be that the robot needed to be run in TeleOp, and the encoder readings had to be sent to the driver station by telemetry. Instead of driving the robot, pushing the robot was more reliable, due to the robot not perfectly going straight, even with setVolocity. One degree off isn't bad on a small field but configuring measurements in a long hallway, one degree makes a big difference.

11. Claw Issues.
    11.1. Claw Scrapes on Ground. When moving a robot, the claw gets sucked into the wheels.

    Issue 11.1 resolved: This is a major issue, when the claw gets stuck the robot drives over the claw, which wasn't fun for us in matches. This is why a push button was installed, afterwords the light would come on when the drivers get too far down with their arm. The code also has a program that makes the arm go in reverse when the arm gets too close.

    11.2. Claw gears coming unmeshed!

    Problem 11.2 in the process: This is more of a build issue than a programming issue, so there isn't much to say about this here, except that the build team is redesigning the claw to be more accidental in picking up pixels. The claw was reprinted, and the claw holder was rebuilt, the enmesh issue shouldn't happen as often or as easily.

    11.3. Getting the claw into the correct position was hard.

    Difficulty 11.3 resolved: this was before we were using a smart servo. When we used the non-smart servo, I had to screw and unscrew a bolt which withered away the holder for the claw, it wasn't a good thing to be doing. After switching to a smart servo, that difficulty was resolved.

    11.4. Picking up pixels was sometimes difficult.

    Difficulty 11.4 resolved: mechanical. Will hopefully be permanently fixed with the new claw design, discom, and foam can always help.

12. Limited time in Autonomous.
    Challenge 12 solved: since teams have only 30 seconds to do autonomous, things must be fast. Sometimes the robot would not finish autonomous quick enough, this was fixed by increasing the speed and decreasing the wait time between each movement.