**8/30 Entry**

Goals:

* Test code that I wrote at home
  + Fixing errors
  + Improve the code
  + Note results
* Finish basic claw control

Progress Updates:

* The claw function I had prepared worked! After a few syntax corrections made.
  + I still have to add holding power based on the position of the claw so it doesn’t burn out. That way I can actually increase the holding power when it is necessary as well which is a really nice added bonus.
* I ran the code to test how long each iteration takes and the result came out at 27ms. I believe that a good portion of this was actually just printing these results to the debug stream so I’ll see how I can try to get more accurate results next time.
  + These results do provide useful insight as knowing the average wait time between each call of a function means I can make realistic predictions and adjust my code accordingly.
* I managed to get the potentiometer based holding power to work!
  + But the potentiometer has a problem. The readouts of the sensor seem to shift a little each time but I thought of a solution I might be able to implement next time. I can record at what point it starts at the beginning of the program, and base my holding power thresholds on that value.

Reflection

* Today was a very productive testing day
  + Got through claw testing
  + Loop testing to find out how long the code takes to run
  + Potentiometer based holding power was tested
    - Found problems with the potentiometer
* I may try adding a struct for sensors later for easier manipulation and access to values of the sensor. It also means I could later add things like calculating velocities and weighted averages behind the scenes and access those really easily, making the code look neater.

To start, I ran the two pieces of code that I prepared at home (after fixing a few errors). The claw function works! Yesssss. But, I'm going to add a way to have holding power applied based on how closed the claw is. If it's closed all the way then I can just cut the holding power because there's no point in it. I also ran the function to test how long each loop takes and I found that it takes around 27 ms for v0.1.4. I'll run these tests periodically to see how much the code influences the run time of each iteration of the program.

After this I got the claw function to work well and made it so that the application of holding power was based on how open the claw is. If it's completely closed I know there is no cap so I just release the power. Otherwise I apply holding power so that the cap doesn't fall out.

Then we ran into a build problem. The intake has stopped working properly. Our intake is structured such that the balls go up at one central collection point and the roller on the front has a metal piece sloped so that the balls go towards the collection point. The problem is that the slope of the metal piece is too shallow and the balls get stuck on the side far from the collection point which in turn burns out the motor for the intake. So it's a pretty big problem. Tomorrow we'll try solving this by changing the way the rubber bands are put on, and by adjusting the curve of the metal.

I figured out another problem, a programming problem. This time the problem is that the potentiometer values for the claw aren't consistent and can shift. At first I just read the value before starting the program, adjusted the constant, then ran the program. My solution that I'll integrate after I add the structs for the sensors is to take note of the starting position of the potentiometer and then base my thresholds off of that.

So I'm doing a bit of work at home and I've set up a struct for the potentiometer and based on that I implemented the system of thresholds based on starting value. Now I just need to test that next time I have robotics which is Monday of next week. I also condensed some nested if else I had into a bunch of one line ternary statements so the code looks much denser but at least easier to see the various bits.

**8/29 Entry**

Goals:

* Continuing development of user control from yesterday
* Work on developing a joystick structs
  + I made this decision because this makes it easier to update and get joystick values without having to write out code that makes the code messy. It also makes it easier to integrate this
  + Eventually develop this into a separate file that I can use for other programs

Progress Updates:

* Compiled the code for the first time after writing 100+ lines of code. Naturally there were a lot a lot of errors.
  + I should try to be better about testing code more regularly and test the code part by part so that at least you know that parts of it work independently of the rest.
  + I have to initialize variables for the structs but since some of that can’t be in the main execution block I had to put them in functions.
  + I also tried defining functions within functions to try to simulate the kind of OOP programming that I was used to but that doesn’t work in robotc so I had to fix that.
  + Eventually ended up with 311 lines of bug free code!
* Using the functions and structs I wrote, I set up the drive, intake, and lift controls in user control.
  + The structs I made facilitated the process by allowing me to approach the code from a more abstracted level which really speeds things up.

Reflection:

* Setting up the structs were a productive endeavor in many different ways
  + Expanded my coding ability. I now am more comfortable with how robotc can be further abstracted for easier and more complex coding
  + Makes the code easier to read and understand
  + Perhapes can be expanded to a file that can be included in many other programs
  + It will provide a very good framework for easily adding pid, slew rate, velocity control, etc, later on.

Notes for Tomorrow:

* Test claw control and refine the code and timings for that

Original Text:

Today I worked on redesigning the user control based on basic motor testing from the previous day and also designed to integrate a joystick struct into the mix to make my code more modular.

Just compiled for the first time. Soooo many errors.

Took me a while to fix all the errors but there were a few problems. One was that to initialize the structs I had to assign some variable values but I can't do that in the main execution block so I had to hide those away in functions. I also had defined some functions in the same functions so that I could simulate methods of OOP programming languages but turns out I can't define functions within functions in robotc so I had to fix that. In the end, 311 lines of code with only warnings. Yes!

It's the end of today's robotics, I've implemented control for the drive, intake, and lift using the functions I wrote earlier. The functions made it easier to be more direct with my thought process to the code, not having to go into really detailed parts of the procedure because the functions had already taken care of them.

I did a little bit of work at home. I implemented a motor struct with just simple power setting functions but it's just a framework so that I can things like pid, slew rate, velocity control, etc., as those things become necessary.

I also added some claw control code so that I can test and refine that at robotics tomorrow.