Main.c

The main file with all the sensor, motor settings and the default tasks. Other files are included here. Some variables are declared here

Autonomous:

* autonomous .h serves as a auton selector(although I didn’t this part of the code so I usually just override and only download one auto)
* Other files in auton folder are the actual commands of each auto, as you can see, we have a lot of them
* autoplace.h-> was suppose to automate the placing mogo action but I didn’t have time to do it
* Similarly for other empty files
* Autofunction has several functions that include some reusable part of autonomous such as picking up mogo and line up for loader

Usercontrol: void usercontrol body serves as the overall function that controls the robot under task usercontrol. It has the ability to run preset actions (such as autoplace which was not written) without freaking out other code. Also actions like such, can be stopped mid-action. This was mainly used to run auto in a safe manner without a comm switch.(action2task is used for this purpose too)

* By default, the void driver func is being looped which interfaces the driver with the robot through the controller inputs.

Subsystems:

* Because of the feature requested, the code need to be able to switch between several different control schemes(joystick value, pid, autonomous(with or without pid) and more if needed). To that end I have dedicated tasks to control each subsystem. The tasks are the only way to interact with the said system. Within the task, a control state value is used to determine whether the robot is in autonomous or driver control or other states so the subsystem can be used accordingly.
* The motors are not directly controlled by drivefunc, instead, a position or motor value is requested and the subsystem task handles the actuation.
* Another benefit of this is that in autonomous all the subsystems are truly independent of each other and complex multitasking can be done to save valuable time.
* These files are named pid\_\*subsystem\*.h

LCD.h is used to manage LCD, however, my teammate wrote some more complex not really competition ready good looking code which can be found in folder Alexander

Pid\_control and pid\_master are pid related stuff usually functions for autonomous.

Linear\_drive can linearize our drive train but it was set to off due to driver request

Slew\_rate is jpearman’s code used in conjunction with pid in autonomous to prevent wheel slipping

Essential\_functions is there to make the code more easily adaptable to other robots(all subsystems except drivetrain is defined here)

Folder Legacy is our old code(from states?)

libGyro GCC2 gyro library, RECOMMEND

libPID GCC2 pid library, I don’t use

Folder Bennett ignore

Audio files-> for our speaker, never got to use them