Autonomous Overview

Sequential and Parallel Commands

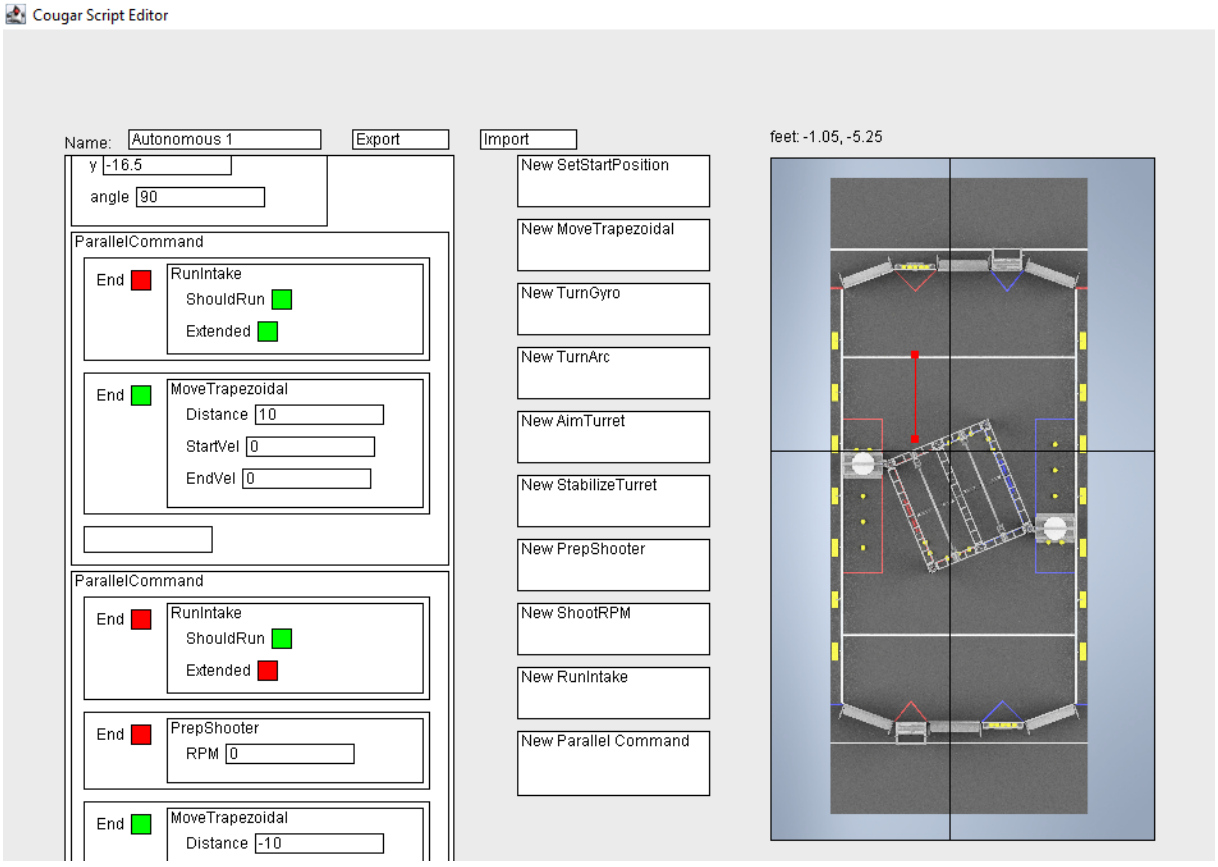
The Commands for autonomous are controlled by sequential command groups and parallel commands. The sequential command group contains a list of commands to run. When the group is scheduled, it handles the scheduling and ending of the commands in its list in sequential order. Parallel commands contain a list of commands to run as well as a list of “end commands” that determine when the parallel command is finished. Parallel commands can be run in the sequential command group and can be used to run several commands at the same time, such as shooting while aiming the turret. The parallel command ends when all of the commands that are labeled as end commands are finished, which allows for one command to easily control the termination of the rest of the commands in the parallel command.

Cougar Scripting

Team 1403’s Cougar Scripting system is a robust system that is used to create sequences for autonomous control of the robot. Scripts can be created through the use of the Cougar Script Editor which is a drag and drop graphical user interface that allows for creation and modification of cougar scripts and provides a preview of the projected actions of the robot. The script is then exported to the robot and can be interpreted and executed by the robot. The system allows for easy modification of the game or robot.

The Cougar Script editor is used to create, modify, preview, and export Cougar Scripts. In the editor, there are three key sections. The queue, which is located on the left, contains a list of RobotCommands that can have commands inserted or taken out. The fields of the commands can be edited by clicking on them while they are in the queue. The command list, located to the right of the queue, is used to create new commands. The preview, located to the right of the command list, shows a preview of the motion and actions of the robot for the current script. When the mouse is hovered over the image, the position of the cursor is reported above the preview in feet, with 0, 0 being in the center. Autonomous sequences can be easily expanded or changed to accommodate changes to the robot and its commands.

The script will be exported and read from top down order into a JSON file, which is then deployed onto the RIO. When the Robot is initialized, the script is parsed and stored in a hashmap.


Cougar Script Editor
Name: Autonomous 1
Export
Import
y -16.5
angle 90
ParallelCommand
End
RunIntake
ShouldRun
Extended
End
MoveTrapezoidal
Distance 10
StartVel 0
EndVel 0
ParallelCommand
End
RunIntake
ShouldRun
Extended
End
PrepShooter
RPM 0
End
MoveTrapezoidal
Distance -10
New SetStartPosition
New MoveTrapezoidal
New TurnGyro
New TurnArc
New AimTurret
New StabilizeTurret
New PrepShooter
New ShootRPM
New RunIntake
New Parallel Command
feet: -1.05, -5.25

Hardware Overview

Sensors and Actuators

Each of the subsystems on the robot uses a variety of sensors in order to record the happenings of the physical system to allow for useful control systems. Following are a list of subsystems, the sensors they use, the ports of them. Their purpose will be explained later with each subsystem.

Subsystem	Port	Device	Use
Drivetrain	10	REV SparkMax Controller	Front Left
	9	REV SparkMax Controller	Back Left (Follows Front Left)
	2	REV SparkMax Controller	Front Right
	1	REV SparkMax Controller	Back Right (Follows Front Right)
Turret	3	REV Through Bore Encoder (Absolute)	Reading Angle
	7	TalonSRX	Yaw Control
	1	Hall effect	Resetting
Shooter	14	TalonSRX Grayhill Encoder	Left Shooter Read Velocity
	13	TalonSRX	Right Shooter (Follows Left Shooter)
Hopper	8	TalonSRX	Left Motor

	1	TalonSRX	Right Motor
Upright	4	TalonSRX	Motor
Color Wheel	2	RevRobotics ColorSensor V3	Read Color
	5	TalonSRX	Wheel Motor
Intake	3	VictorSPX	Run Belts
	4	VictorSPX	Extension Motor
	9	Through Bore Encoder	Read Extension Angle
Climber	2	TalonSRX	Left Climb
	3	TalonSRX	Right Climb
		Servo	Left Ratchet
		Servo	Right Ratchet
		Photogate	Top Sensor
		Photogate	Bottom Sensor

Joystick Input

Joystick	Button/Joystick	Function
Driver	Left Stick	Left Drivetrain
	Right Stick	Right Drivetrain
	Hold LB	Toggle Arcade Drive
	Hold Back	Color Wheel
	Hold LT	Manual Color Wheel Control
Operator	Left Stick	Control Turret
	Hold B	Activate Field Relative Control for Stick
	Hold Y	Activate Manual Control for Stick
	Hold A	Activate Limelight Tracking
	LB	Run Upright Reversed
	RB	Run Intake
	RT	Intake Extension Out
	LT	Intake Extension In
	Right Stick	Run Feeder Manual
	Hold X	Run Upright When Shooter is Ready

Drivetrain

Overview of Functionality

The code for the drivetrain consists of four REV SPARK MAX motor controllers (two on each side), with the back controllers following the front ones. The front two use the built in PID

to power, to ensure that when the battery is at low voltage, it will still drive straight, and it allows for more consistent autonomous driving capabilities.

The driver is able to use two modes of control to drive: tank drive, and arcade drive. Tank drive allows for each joystick to control the velocity for its respective side of the drivetrain, while arcade drive uses one joystick to control the forward velocity and the other to control angular velocity. This makes it easier for the driver to go directly forward/backwards and achieve reliable point turns.

Commands

NEODrivetank is the default teleoperated driving command, and allows the driver to use both tank drive and arcade drive.

For autonomous, there are three main commands which allow for driving in autonomous: TurnGyro, DriveTrapezoidal, and TurnArc. TurnGyro uses an ADXRS450 gyro in tandem with PID to make the robot use a point turn to orient itself to a desired angle. DriveTrapezoidal uses our custom-made trapezoidal profile generator to generate velocity profiles given the goal distance, and start and end velocities (to allow for smoothly chaining commands together), and sets the motor setpoint speeds to the output of the profile. It also uses PID on the distance travelled to account for error in overshooting/undershooting its current vs. expected distance. Finally, TurnArc takes in an arc radius and angle to turn in a circular arc, and it can be smoothly chained with DriveTrapezoidal (or other TurnArcs) to have versatile driving.

Turret

Overview of Functionality

The turret consists of an absolute encoder to record its position, a hall effect to allow the encoder to reset on startup, a Limelight 2+ for tracking and distance calculation, and a TalonSRX to turn it. The code will constantly track the orientation of the turret to ensure that it stays within the valid area (due to their being a mechanical deadzone). The turret will automatically

ignore setpoints to the PID that will enter this deadzone, and will go to the nearest point outside the deadzone if it is given a point within it.

It has field relative capabilities, meaning it can maintain an angle relative to the orientation of both the robot and the field, and orientations given by the operator automatically are converted to become field relative.

Commands

At the start of every match, the turret will automatically reset itself using a hall effect sensor and two magnets to ensure that it will never sweep past the deadzone. It will sweep in one direction until one magnet sees the hall effect, and it will sweep in the other direction for a short amount of time to ensure that it did not see the wrong magnet, and finally it will apply the reading as an offset to the turret angle PID. The purpose of the second magnet is to give it a lower chance of failing on startup.

For operator control, the TurretPassiveControl allows for the varying forms of functionality using a state machine to switch between control modes. There are 6 states in this machine: Idle, percent output, vision, field relative, reset1, and reset2. After resetting on startup, the default state is idle in which the turret will maintain its current robot relative angle, and the operator can switch through the other control modes using buttons. The variety of controls ensure that the operator is able to precisely manipulate the turret to easily aim it at the target without needing to worry about the deadzone.

For autonomous scripting, SetTurretAngle, StabilizeTurret, and AimTurret allow for setting the field relative angle of the turret, stabilizing it so it will offset the gyro turning of the robot, and limelight vision tracking respectively.

Shooter

Overview of Functionality

The shooter consists of two TalonSRXs, one with a grayhill encoder to measure velocity. It also keeps track of the hopper and upright, and allows them to run simultaneously when the limelight is aligned and when the RPM is approximately matched to the setpoint, to ensure that no balls are fed while the shooter is still accelerating so the shots can be consistent.

Commands

For teleoperated control, the ControlledShoot command allows the operator to use a button to set the speed of the shooter based on the distance reading of the limelight, and the shooter is able to convert that reading into an RPM utilizing interpolated ideal data points. The operator is able to prep the shooter up to the desired RPM and hold a button to have the upright feed the shooter balls when the limelight is aligned and when the velocity is approximately matched, so that they will not need to worry about needing to stop holding the button. Additionally, the limelight is used to approximate the robot's distance from the target, and the field relative angle is used to adjust so that the limelight centers such that it is aiming at the inner port, allowing the operator to easily shoot from anywhere by the press of a button.

The autonomous commands consist of one to prep the shooter while other commands are running, and a matched RPM mode which will run for a set amount of time, feeding the shooter when it is ready for a set amount of time.

Intake

Overview of Functionality

Team 1403's Intake subsystem allows the robot to manipulate the game pieces on the field. Using two motors and an absolute encoder, Team 1403 is able to extend the intake beyond the bumpers as well as control the belts to retrieve the power cells on the game field.

Commands

The IntakeCommand allows the Intake subsystem to manipulate the power cells as well as extend beyond the bumpers. When executing, the belts run through a percent output when a joystick button is pressed. However, if the button is lightly pressed, then the intake would not run because there are instances when a button is not completely deactivated, which causes the intake to continue to run. As a result, we implemented a deadzone so that if the speed of the belts were less than the dead zone, it would stop the intake. Another button would activate the intake extension, which runs a method which determines if the intake is extended or tucked in. If the intake is tucked in, the button would cause the intake to extend to a specific angle, which we determined by getting the angle from the absolute encoder. The intake would stay on the angle due to the position PID.

Control Panel

Overview of Functionality

The Control Panel subsystem makes sure that the robot is able to rotate the color wheel 3-5 rotations and spot the desired color. Using the color sensor and a motor, we use a wheel to control the rotations of the color wheel while the color sensor observes the color on the wheel.

Commands

First, each color had a certain RGB value according to the color sensor. So, we used string types to determine the color according to the sensor. However, since the wheel is on a separate location compared to the goal location, we had to state that the color sensed on the control panel is two colors behind the goal color. For instance, if the target color is yellow, the control panel would stop rotating the color wheel when the color sensor senses green.