File: falcon\_node.cpp

Author: Andrew Burroughs

Detailed Description:

The falcon node controls the Talon FX motor controllers and sets the speeds of the Falcon motors attached to them. The TalonFX motor controllers are built into the Falcon 500 motors and are used to set the speeds of the drive train motors. Two motors are attached to a single gearbox, necessitating the Falcons be synced.

Issues:

The second Falcon motor might not be following the first. Additionally, there are two Falcons hardcoded into the node because we couldn’t get a reference to the second to function properly in the time left.

Software Documentation

Global Variables

**rclcpp::Node::SharedPtr nodeHandle**

This is an instance of the ROS2 pointer and is used to create the ROS2 node.

**bool GO**

This variable was created as a safety feature. When the user presses the trigger on the joystick, the logic node publishes a message called GO that alerts

**bool useVelocity**

This variable determines whether the program uses the velocity or percent output control modes. If it is true, the control mode used is the velocity. If it is false, the control mode used is percent output. It is not currently in use.

**int velocityMultiplier**

This variable is only used when the control mode used is velocity. The speed data is multiplied by the velocityMultiplier. It is not currently in use.

**int testSpeed**

This variable is used for debugging purposes. It is intended to set the motor speeds to the speed of this value. It is not currently in use.

**TalonFX\* talonFX**

This is a pointer to a talonSRX object that is used to call the functions needed to set the various data.

**TalonFX\* talonFX2**

This is a pointer to a talonSRX object that is used to call the functions needed to set the various data.

Function Documentation

**void stopCallback**(std\_msgs::msg::Empty::SharedPtr empty):

Callback function triggered when the node receives a topic with the topic name of STOP. This function sets a Boolean value GO to false, which prevents the robot from moving.

Expected Input

empty – This is a ROS2 message of type empty with the name STOP

Expected Results

The GO variable will be set to false, and the motors will stop any movement.

**void goCallback**(std\_msgs::msg::Empty::SharedPtr empty):

Callback function triggered when the node receives a topic with the topic name of GO. This function sets a Boolean value GO to true, which allows the robot to drive.

Expected Input

empty – This is a ROS2 message of type empty with the name GO

Expected Results

The GO variable will be set to true, and the motors will be able to run.

**void speedCallback**(std\_msgs::msg::Float32::SharedPtr speed):

Callback function triggered when the node receives a topic with the topic name of drive\_left\_speed or drive\_right\_speed. This function takes the data from the topic and sets the motor to the speed specified.

Expected Input

speed – Float32 with range [-1, 1]

Expected Results

The node will set the motors to the speed specified.

**template <typename T>**

**T getParameter**(std::string parameterName, std::string initialValue):

This function takes a string as a parameter containing the name of the parameter that is being parsed from the launch file and the initial value of the parameter as inputs, then gets the parameter, casts it as a string, displays the value of the parameter on the command line and the log file, then returns the parsed value of the parameter.

Expected Input

parameterName – This is the name of the parameter included in the launch file

initialValue – This is a string value and is the initial value of the parameter

Expected Results

This function will cast the parameter and return the value of the parameter

**template <typename T>**

**T getParameter**(std::string parameterName, int initialValue):

This function takes a string as a parameter containing the name of the parameter that is being parsed from the launch file and the initial value of the parameter as inputs, then gets the parameter, casts it as a string, displays the value of the parameter on the command line and the log file, then returns the parsed value of the parameter.

Expected Input

parameterName – This is the name of the parameter included in the launch file

initialValue – This is an int value and is the initial value of the parameter

Expected Results

This function will cast the parameter and return the value of the parameter

Change Log:

7/3/2022: Documentation was created

10/1/2022: Description and issues were added by Andrew Burroughs