File: logic\_node.cpp

Authors: Andrew Burroughs, Bill Johnson

Documentation Author: Andrew Burroughs

Detailed Description:

This file implements the logic needed to create the logic node. It receives information published by the communication node, then wraps that information into topics, and publishes the topics. The node listens for the information sent by the user and received by the communication node, then translates this information into instructions that are then implemented by the motors.

Issues:

Software Documentation

Global Variables

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

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

**float joystick1Roll**

Float variable used to store the value of the roll component of the joystick input.

**float joystick1Pitch**

Float variable used to store the value of the pitch component of the joystick input.

**float joystick1Yaw**

Float variable used to store the value of the yaw component of the joystick input.

**float joystick1Throttle**

Float variable used to store the value of the throttle component of the joystick input.

**bool automationGo**

Variable used to track whether the user has toggled automation on or off.

**bool invertDrum**

Variable used to track the direction the user wants the drum to spin.

**bool excavationGo**

Variable used to track whether the robot is in drive mode or excavation mode.

**Automation\* automation**

Description of variable

**driveLeftSpeedPublisher**

ROS2 publisher that publishes the speed of the left drive train motors.

**driveRightSpeedPublisher**

ROS2 publisher that publishes the speed of the right drive train motors.

**dumpBinSpeedPublisher**

ROS2 publisher that publishes the speed of the dump bin linear actuator.

**shoulderSpeedPublisher**

ROS2 publisher that publishes the speed of the excavation linear actuator.

**excavationArmPublisher**

ROS2 publisher that publishes the speed of the excavation motors controlling the rotation of the arm.

**excavationDrumPublisher**

ROS2 publisher that publishes the speed of the excavation drum motors.

**servoStatePublisher**

ROS2 publisher that publishes the state of the dump bin servos.

Function Documentation

**void initSetSpeed**():

Description of function

This function is used to set the motor speeds to zero. It was assumed that the motors would automatically zero themselves out upon the robot restart, but this wasn’t always the case. Thus, this function ensures that the robot won’t start up and do odd things, like run into walls.

Expected Input

N/A

Expected Results

All motors will receive a zero message and won’t move.

**void updateSpeed**():

Description of function

This function is called by the joystickAxisCallback to update the speed to the wheels. It uses the data published to compute the speed of the wheels based on the joystick information and limited by the maxSpeed variable. This variable is used as a safety feature and can be adjusted as needed to ensure the speed of the robot isn’t too extreme.

Expected Input

N/A

Expected Results

The wheels should respond to the new values generated by the user inputs to the joystick.

**void stopSpeed**():

Description of function

This function is called when the user presses the toggle excavation button and switches to the excavation mode. This creates a speed message with a speed of zero and publishes it to the drive train motors. Without this function, it was assumed that the motors would stop when they received a zero message due to the joystick not being pressed. However, due to rate limiting or other issues, they wouldn’t always stop, and the rover would occasionally spin whilst attempting to dig.

Expected Input

N/A

Expected Results

The drive train motors should stop moving and shouldn’t move during the excavation sequence.

**void updateExcavation**():

Description of function

This function is called when the node receives joystick information and is in excavation mode. The function publishes the shoulder speed, arm speed, and drum speed information. The shoulder speed is based on the joystick pitch value, the arm speed is based on the joystick yaw, and the drum speed is based on the joystick throttle.

Expected Input

N/A

Expected Results

The excavation motors should respond as expected given the user’s input to the various joystick axes.

**void stopExcavation**():

Description of function

This function is called when the toggle excavation button is pressed and switches back to the drive mode. There were no issues encountered prior to this function, but given the issues with the drive train motors, it was created out of an abundance of caution and to guarantee the motors performed the expected actions.

Expected Input

N/A

Expected Results

The excavation motors should stop and shouldn’t move until the bot is placed back into excavation mode.

**flota transformJoystickInfo**(float info, float deadZone):

Description of function

This function is used to transform the joystick input into the desired output. Because the joystick doesn’t return perfectly to the zero position when the user releases the joystick, the bot will continue to receive small commands and cause the motors to spin very slowly, rather than stop. To solve this, we introduced a dead zone variable that is used to set the zero position. This function will change any values in the range [-deadZone, deadZone] to zero, allowing the user to have more precise control of when the robot is moving.

Expected Input

**float info** – expected to be in the range[-1, 1], but can technically be any float value

**float deadZone** – expected to be positive and roughly 0.1

Expected Outputs

**float transformed** – transformed float value, expected to be in the range [-0.9, 0.9].

**void joystickAxisCallback**(const messages::msg::AxisState::SharedPtr axisState):

Description of function

This function is called when the node receives a message with the topic name joystick\_axis, then converts the joystick input using transformJoystickInfo. After getting the transformed values, the function then calls either updateSpeed() or updateExcavation() depending on the mode.

Expected Input

axisState – Message containing axis and axis values

Expected Results

The excavation or drive train motors should respond as expected.

**void joystickButtonCallback**(const messages::msg::ButtonState::SharedPtr buttonState):

Description of function

This function is called when the node receives a topic with the name joystick\_button. Button 2 toggles the drive and excavation states. Button 3 inverts the direction of the drum. Buttons 6 and 7 control the locking servo.

Expected Input

buttonState – Message containing the button value and current state of the button.

Expected Results

The various functions should be performed according to which button was pressed.

**void joystickHatCallback**(const messages::msg::HatState::SharedPtr hatState):

Description of function

This callback function was called when the logic node received a message with the topic name joystick\_hat containing the data for the current state of the joystick hat. If the user was pressing the hat up, the function publishes data to the dump bin, controlled by a talon node, to cause the bin to rise. If the user was pressing the hat down, the dump bin was lowered.

Expected Input

hatState – Int values

Expected Results

The dump bin will respond as expected given the joystick hat inputs.

**void keyCallback**(const messages::msg::KeyState::SharedPtr keyState):

Description of function

This function is called when the node receives a topic with the name key\_state. Currently, this function only toggles the automationGo variable if the user presses the ‘s’ key.

Expected Input

keyState – message containing the key pressed and current state of the keypress

Expected Results

Toggles the automationGo variable if the user presses the s key.

**void zedPositionCallback**(const messages::msg::ZedPosition::SharedPtr zedPosition):

Description of function

This function is called when the node receives a topic with the name zed\_position. It was the first attempt at getting the various information needed for autonomy. It does some math to figure out various values.

Expected Input

zedPosition – ROS2 message

Expected Results

The function displays various values to the screen.

Change Log:

7/3/2022: Documentation was created

10/3/2022: Descriptions and issues were added by Andrew Burroughs

1/23/2023: Variable and function descriptions were added by Andrew Burroughs