File: excavation\_node.py

Author(s): Andrew Burroughs

Detailed Description:

This file is the implementation of the ROS2 node responsible for controlling the excavation motors. The motors used for the excavation system on the 2021-2022 were the ODrive 270 motors. The purpose of this file was to convert the signals sent from the logic node into movement by the motors.

Issues:

There are currently issues with the error handling of this file. If the motors go into an error state, the program will attempt to restart before it shuts down and has the user take over. The program is occasionally sluggish to respond and doesn’t always move the motors as intended.

Software Documentation

Global Variables

**calibrated**

This variable was created to track whether the motors had successfully performed the calibration sequence. If something was wrong with the motors, they would return an error state and fail to calibrate.

**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 subscribing nodes the motors are safe to run.

**errorState**

This variable is the error state of the motors. If a motor is not functioning properly, the errorState flag is set to high and the errorChecking function is called.

**restarted**

This variable tracks whether the motors have been restarted before. To correct the errors detected when errorState is high, the motors will recalibrate. However, if errorState is high after a second calibration, the error is one that will not be solved by calibration and requires user interaction to solve.

Function Documentation

**\_\_init\_\_**(self):

The init function is called when the program first runs. It creates an instance of the super class with the name ‘excavation\_node’, then sets all four global variables (calibrated, GO, errorState, and restarted) to false. It then calls the functions findODriveObjects, calibrate, and setRequestedState. Next, it creates subscriptions to excavation\_arm, excavation\_drum, STOP, and GO.

Expected Input:

None

Expected Results:

The function will print several statements on the screen allowing the user to track the progress of the function and will have called functions to find the ODrive objects, calibrate them, set the motor states accordingly, then create the ROS2 subscriptions as needed.

**errorChecking**(self, odrv, num):

The errorChecking function has an ODrive object passed to it and checks if there are errors for the system, axis0, axis0.motor, axis1, and axis1 motor. If errors are detected at any level, then the errorState variable is set to True and the ODrive object is passed to the calibrate function. The num parameter is only used for printing information to the screen about which ODrive object is being checked for errors.

Expected Input:

odrv – ODrive Object

num – Float, should be either 0 or 1

Expected Results:

This function will print a statement about where the error has occurred, set errorState to True, then pass the odrv object to calibrate to fix the issue.

**calibrate**(self):

This calibrate function only takes the self parameter and is called by the init function to calibrate the motors upon startup. It is hardcoded to go through the sequence axis0.motor0, axis1.motor0, axis0.motor1, then axis1.motor1. The motors are set to the AXIS\_STATE\_FULL\_CALIBRATION\_SEQUENCE state, which causes the motors to emit a beep, then make a full rotation in each direction. The program waits until the motors finish turning to begin the calibration sequence on the next motor to avoid drawing too much current.

Expected Input:

None

Expected Output:

The function will print a statement that is has started the calibration sequence, calibrate all motors, then print a second statement that the motors have been calibrated.

**calibrate**(self, odrv, num):

The calibrate function that requires the self, odrv and num parameters is intended to reduce the reliance on the hardcoded values in the functions calibrate0 and calibrate1. This function was intended to clear the errors and then calibrate both axes of the ODrive object that was passed to it. When the axes entered the AXIS\_STATE\_IDLE after calibration, axis0 was set to AXIS\_STATE\_CLOSED\_LOOP\_CONTROL, which allowed the motor to be controlled using velocity control. The errorState variable was then set to False.

Expected Input:

odrv – ODrive Object

num – Float, should be either 0 or 1

Expected Results:

The function will print a statement to the screen about which ODrive object was being calibrated, clear the errors, calibrate the motors, then set errorState to False.

**calibrate0**(self):

This function was intended to be called when there was an error detected on odrv0. The function was to clear the errors, recalibrate both axes, then set the state of axis0 to AXIS\_STATE\_CLOSED\_LOOP\_CONTROL, which allows the motor to be controlled using velocity control.

Expected Input:

None

Expected Output:

The function will print a statement to the screen about calibrating odrv0, clear the errors, calibrate the motors, then set axis0 to the closed loop control.

**calibrate1**(self):

This function is identical to calibrate0, but resets odrv1.

Expected Input:

None

Expected Output:

The function will print a statement to the screen about calibrating odrv1, clear the errors, calibrate the motors, then set axis0 to the closed loop control.

**findODriveObjects**(self):

This function is called by the init function on startup. It is assumed that both ODrive motor controller boards are attached to the Jetson Nano using the USB cable. This function looks for a board with the serial number 2083367F424D and assigns this board to odrv0 when it is found and 20773881304E as odrv1. This allows the boards to be assigned consistently and prevents issues from arising with the wrong motors responding to commands.

Expected Input:

None

Expected Results:

The function will find both ODrive objects and set them to the correct designation based on the serial number.

**setRequestedState**(self):

The motors must be calibrated and then set to the correct states to run correctly.  For axis1 on both odrv0 and odrv1, the motor is configured to mirror axis 0, then put into input mode 7, which is mirrored input.  This allows the motor to exactly mirror the output of axis 0 and prevents the motors from opposing each other.  For both ODrive objects, axis 0 is set to control mode 2, velocity control, and initialized with a velocity of 0. They are then set to AXIS\_STATE\_CLOSED\_LOOP\_CONTROL, which allows the motors to run. NOTE: This setup has not been proven to work, it relies on information from the ODrive website that might be outdated.

Expected Input:

None

Expected Results:

The function will print a statement to the screen that the function has started, set the various settings to the values specified, then set calibrated to True.

**excavationDrumCallback**(self, msg):

This is the callback function of the excavationDrumSubscriber and is triggered when the node receives a message with the topic excavationDrum. The message contains a float that should be the new speed of the drum. If the motors are calibrated, are not in an error state, and the GO flag is still high, the new speed is calculated by multiplying the float by 144, which is the maximum RPM of the motor. To ensure that the motors do not use too much current and throw an error, the old speed is calculated as well, and the motor is stepped up to the new speed if it is greatly different from the old speed.

Expected Input:

msg – Message of type Float32 with range -1 to 1

Expected Results:

This function will adjust the speed of the motors controlling the excavation drum.

**excavationArmCallback**(self, msg):

This is the callback function of the excavationArmSubscriber and is triggered when the node receives a message with the topic excavationArm. The message contains a float that should be the new speed of the arm. If the motors are calibrated, are not in an error state, and the GO flag is still high, the new speed is calculated by multiplying the float by 144, which is the maximum RPM of the motor. To ensure that the motors do not use too much current and throw an error, the old speed is calculated as well, and the motor is stepped up to the new speed if it is greatly different from the old speed.

Expected Input:

msg – Message of type Float32 with range -1 to 1

Expected Results:

This function will adjust the speed of the motors controlling the excavation arm.

**stopCallback**(self, msg):

The stopCallback function is called when the program receives a ROS2 message with the topic STOP. This message is sent when the user releases the trigger on the joystick. The function sets the GO variable to false, then sets the velocity of both the excavation arm and drum motors to zero.

Expected Input:

msg – Message of type Empty with topic STOP

Expected Results:

This function will set the speeds of all motors to zero and prevent the robot from moving.

**goCallback**(self, msg):

The goCallback function is called when the program receives a ROS2 message with the topic GO. This message is sent when the user presses the trigger and is a signal to the motors that the user is in control and the motors can move. The function sets the GO variable to true.

Expected Input:

msg – Message of type Empty with topic GO

Expected Results:

This function will allow the motors to move when they receive a valid speed command.

Change Log:

7/1/2022: Documentation is created