Lab 1 - Activities

RBE 2002

10/27/2024

Elliot Scher, Evan Kaba

Contents

Abstract

Using the Pololu Romi platform and programming framework provided in class to accomplish the following goals: PID Controller-based Motor control using encoder ticks per interval feedback; Understand and customize the Robot Remote state machine framework; Connect Line following array and write PID feedback controller to line follow; Tune PID gains for smooth and fast motion.

Equipment

- RBE 2002 Romi Kit
- IR Remote and IR Receiver
- Line Sensor Array board (And jumper wires)
- Tape Measure, Smartphone (stopwatch)
- Line Following Course

Introduction

Handout Worksheet

1. Encoder Capture

Forward Direction - Clockwise

A	B	A^B	lastA	newA	lastB	newB	newA^lastB	lastA^newB	ΔencCount
↑	H	0	0	1	1	1	0	1	-1
↓	H	1	1	0	1	1	1	0	1
↑	L	1	0	1	0	0	1	0	1
↓	L	0	1	0	0	0	0	1	-1
H	↑	0	1	1	0	1	1	0	1
H	↓	1	1	1	1	0	0	1	-1
L	↑	1	0	0	0	1	0	1	-1
L	↓	0	0	0	1	0	1	0	1

2. Encoder Processing

Pololu uses an XOR gate across channel A and B so that they don't need to use two interrupt pins to calculate the change in encoder state.

3. Encoder Technology

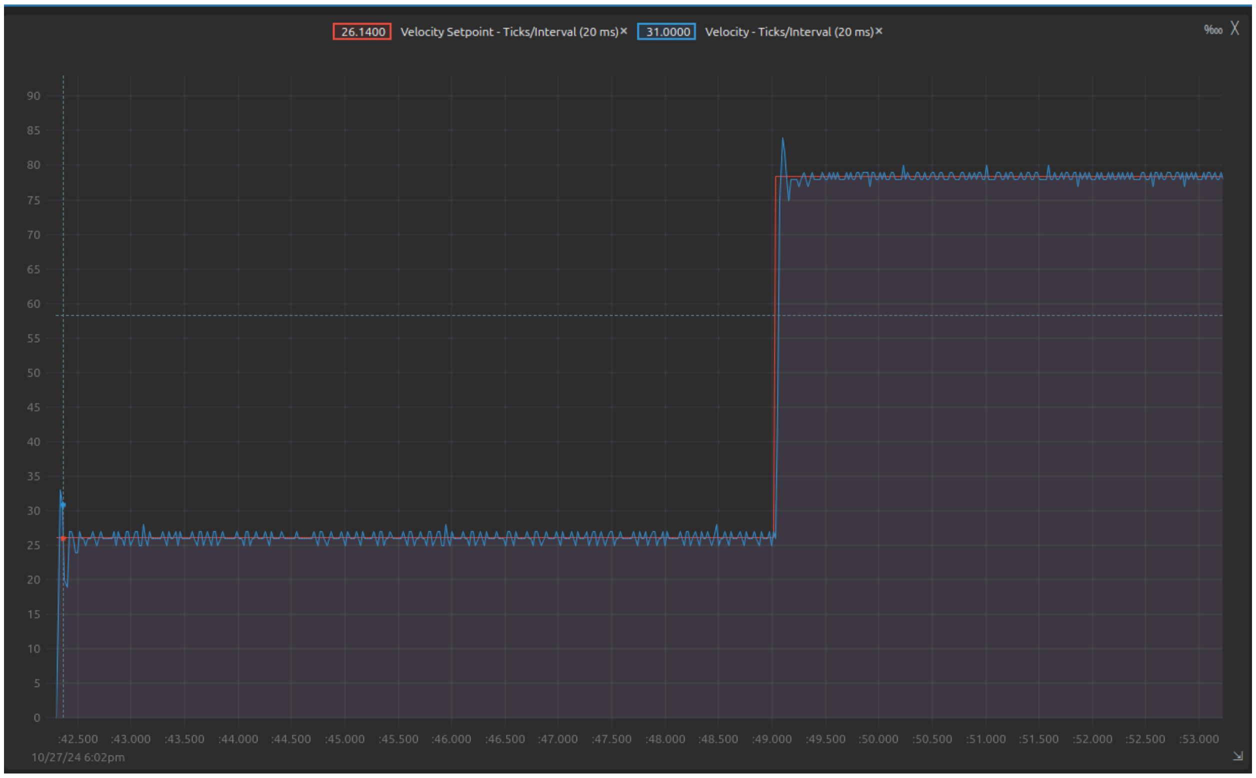
The romi's motor encoders use hall effect sensors to detect the motion of magnets on a wheel attached to the motor's rotor. The hall effect is a change in voltage as a result of a magnetic field moving over a wire with a constant current. The change in voltage is interpreted (through a XOR gate) by the Romi as some motion of the motor.

4. Encoder Hysteresis

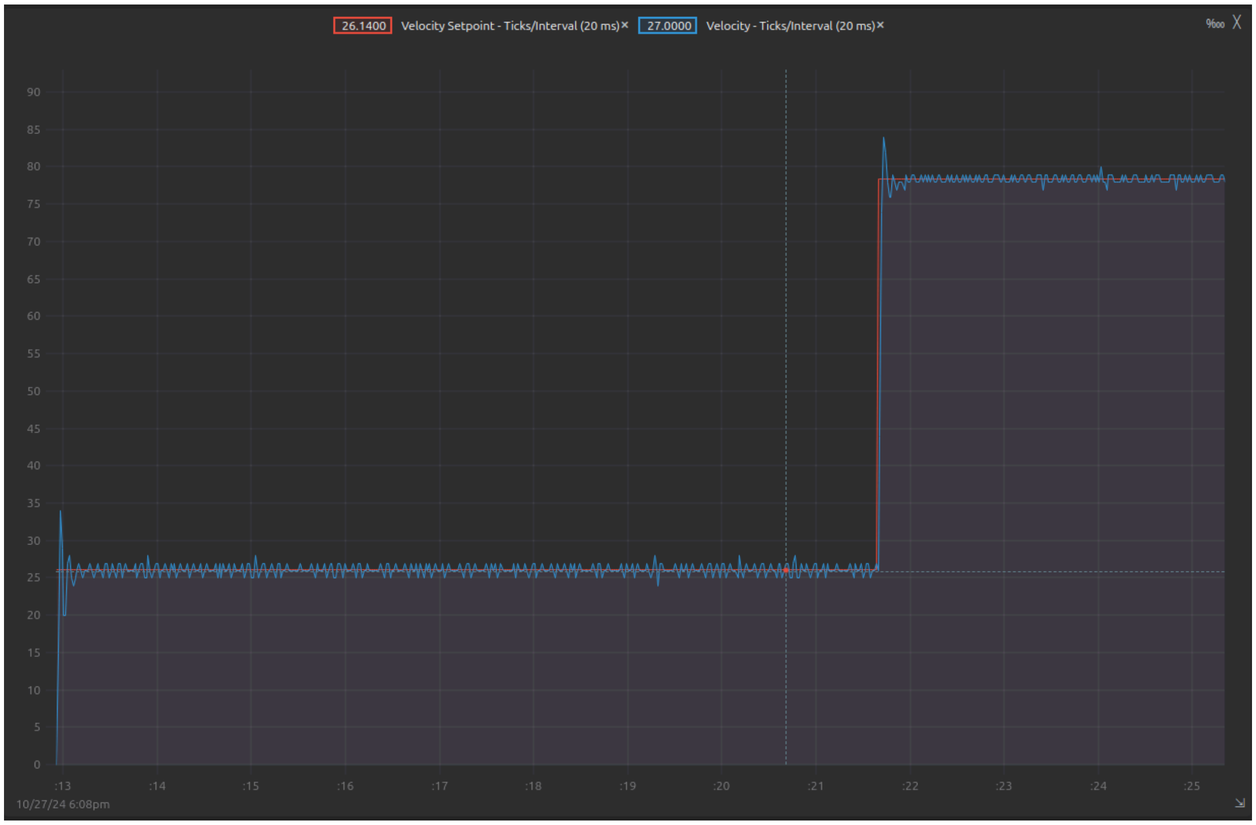
The encoders as a system do need knowledge of the previous state in order to find the encoder delta because if A or B is falling or rising, the direction of the motor is different.

5. Motor Control Algorithm

The motor control algorithm is a PID controller using the motor's velocity as the process variable. We increased Kp until the wheels reached a steady-state velocity error. We then increased Ki until the robot reaches the velocity setpoint, we then decrease Kp until the robot stops oscillating around the velocity setpoint. We used the gains Kp = 7.5, and Ki = 2.0 to achieve a satisfactory response from the system.



When we added the derivative term, with a gain of Kd = 0.5, we saw that the system settled to the setpoint slower, but the oscillations once at steady state were smoother and less noticeable. The Kd gain was tuned through experimentation.



6. Line Following Error

We calculated the line following error by subtracting the left sensor's ADC value from the right sensor's ADC value. This gave an integer range of values between -1023 and +1023.

7. Line following Derivative term

Yes we added a derivative term to the effort calculations. The derivative term lets the controller react to the change in error over time, which is important in handling corners when line following. The mathematical expression for effort is: *effort = Kp * error + Kd * (error - prev_error);* which is returned from the function in a separate line.

The variable *prev_error* is set to the value of *error* before the effort is returned.

8. Links to GitHub repos

Elliot's Repo, Evan's Repo

9. Link to robot driving video

Elliot's Run, Evan's Run

10. Experimental Data for Target Speed v. Measured Speed

Each robot was set at the end of a tape measure stretched out to 100 cm. Then, the robot was commanded to go the requested speed, and at the same time, a stopwatch was started. Once the robot reached the 50 cm (for 5-10 cm/s) or 100 cm (for 15-25 cm/s) marks, the stopwatch was stopped, and the measured speed was found with the following formula: *measured speed = covered distance / total time*. Each test was performed once per robot for the respective times.

Target Speed	Measured Speed (Robot A)	Measured Speed (Robot B)
5 cm/s	5.08 cm/s	4.97 cm/s
10 cm/s	9.70 cm/s	9.80 cm/s
15 cm/s	14.57 cm/s	14.88 cm/s
20 cm/s	19.88 cm/s	19.96 cm/s
25 cm/s	24.50 cm/s	25.12 cm/s

11. Extra: Control

N/A

12. Extra: Pursuit Challenge

N/A

13. Wrapping up: Team Member Contribution

Student	Percentage
Elliot Scher	50.0 %
Evan Kaba	50.0 %

Both participants implemented and tuned feedback controllers separately and came together to answer the conceptual questions and find experimental data. Both participants did the same thing, on their own code.