Colorado School of Mines

Memo

To: Instructor and TA

From: Christian Prather

Team #: M420

Date: 10-22-20

Re: Lab 3: Closed loop control

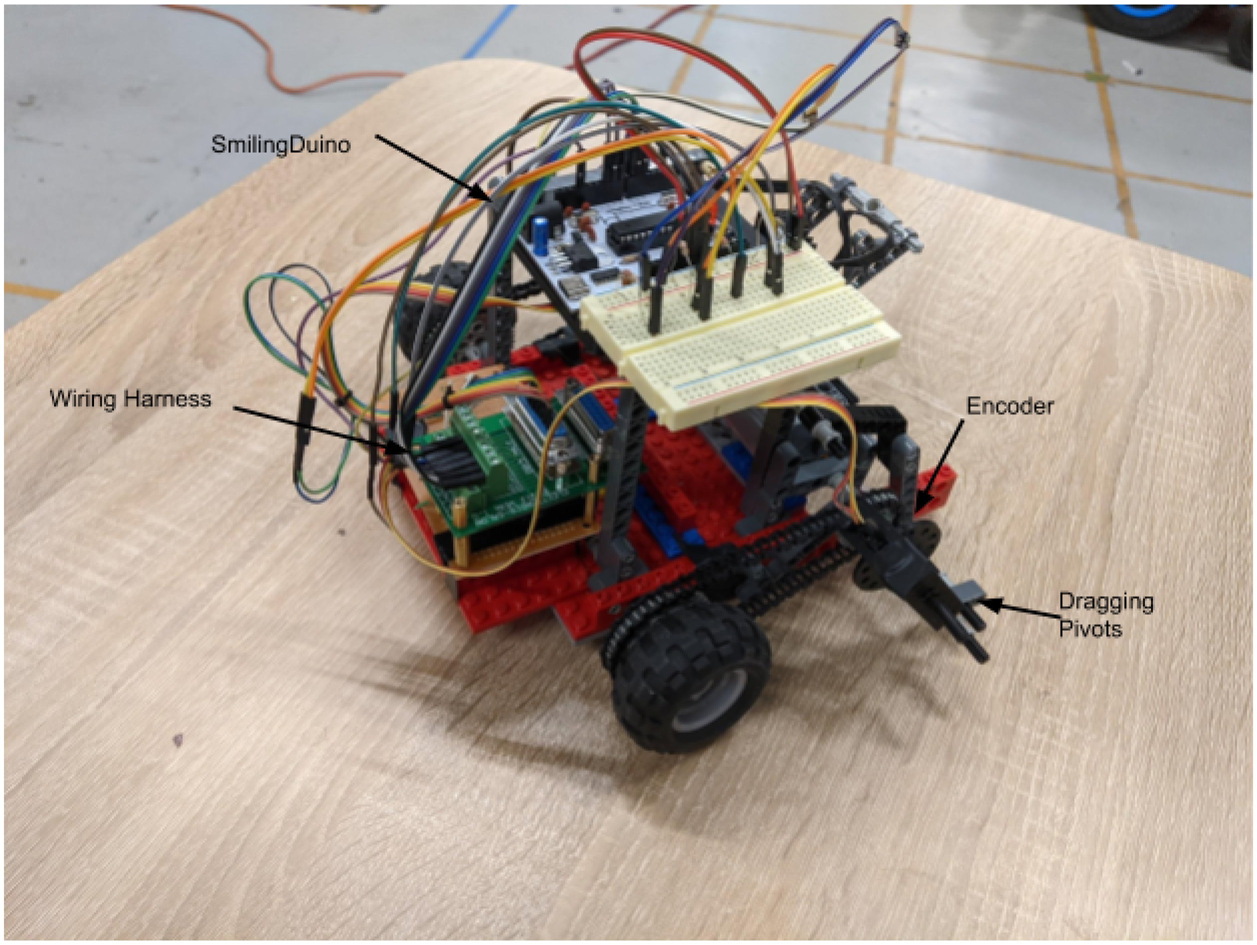
Problem Statement:

Robotic control can be simulated perfectly all day however once you enter the real world variables arise that you can not model. It is the designers job to build a platform that can handle such disturbances. One method of doing so is through sensor feedback. This feedback method closes the loop in the robot's logic and tells it to take into consideration what has happened in the past and adjust to meet your goal. In this lab's case we have introduced encoders to help give feedback as to actual motor rotation.

Methods:

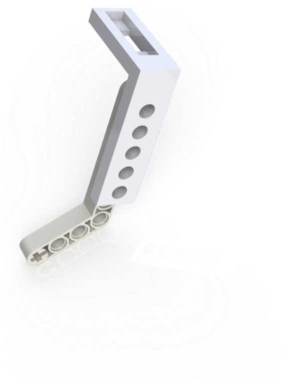
Sensors:

To offer the highest resolution in the encoders values their placement was set at the motor shaft as any placement at the geared down end of my drive train would have resulted in fewer revolutions per linear distance traveled and this would have limited resolution.



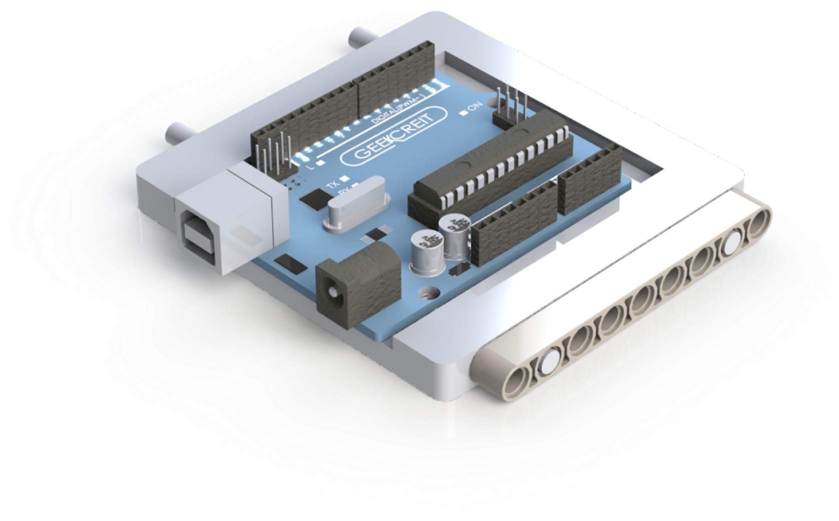
(Fig 1)

I decided that rather than dealing with mounting the encoder using a combination of legos I would design and 3D print a small bracket to mount the encoder to my motor wheel. This went through a couple of revisions but proved a good choice as my largest area of hindrance is my robots chaise.



(Fig 2)

I also designed and printed a mount for my Arduino board that allowed for better mounting then I had in the last lab and added structural integrity to my bot.



(Fig 3)

Algorithm:

I chose to implement my own code from scratch for two primary reasons.

1) I wanted the experience as I am very familiar with Arduino programming and wanted to have some more fun with it.

2) As labs progress I do not want to be reliant on another person's code (regardless of how good it is) as incorporating new features will be far easier for myself if I am familiar with the code base.

This also allowed me to break out the code to a higher degree utilizing multiple simple functions as well as a standard PID library offered by Arduino which I have used before. A simple implementation (documented in detail in code) follows the flow of taking in next direction command from a list of (command, distance) pairs, calculating the desired encoder counts given constants such as distance/ degrees per rotation and adjusting PWM values of each motor every cycle based on their respective encoder count values (incremented by separate ISRs) compared to a desired count and fed through their individual PID loops. (As a side note I also made a simple wiring harness as I was sick of how delicate my wiring was with the breadboard and connectors, while this is not a great solution in this lab the goal is to order a simple shield from jlcpcb for the next lab. I also replaced the standard Arduino Uno board with a custom one I designed as it has a type c USB port which is only important because it offers less resistance when removing it as I was damaging my bot removing the cable so many times).

Results:

I ran my robot through the test maze a surplus of 20 times and had a mix set of results. The primary conclusion that can be scrubbed is that the software was performing very well, when tested on a bench top with the robot immobile or on hardwood floor the bot would move to the same spot almost 100% of the time and the encoder counts would always be with in 2 away from goal as the deceleration curve was performing great. These results did not hold well when maze testing however as this lab seriously highlighted the shortcoming of the chaise design. Its simple dragging pivots in the back seemed to get caught on anything and everything while the wheels would continue spinning, tricking the feedback loop and throwing the bot entirely off. In runs that proved successful my average completion time was 42.16 seconds making it to square 10.

Table 1 shows the results of 3 relatively successful runs

Note: As my chassis proved a large hindrance I allowed mild overlap of robot – wall boundaries in successful runs)

	Run 1	Run 2	Run 3
Time (seconds)	41	40.5	45

I did need to adjust my kP, kD, and kI values from the starting points provided in the library to compensate for the steady state error. I increased my integral value to allow for better stopping at exact distance with limited overshoot. I also set a boundary for the PID output to max at different values per motor as if both hit 255 early on the variance between them is great enough that I would get drift before it had time to compensate. I also set the lowest to 150 as this allows a slow crawl to move the encoders while not getting stuck in an infinite loop before it enters a deadband.

Conclusions:

There are two major conclusions to be drawn from this lab.

1) A simple encoder based feedback system offers an increase in precision given no major disturbance that are capable of tricking them (motor spinning but robot stationary) but the accuracy of the robots final destination is not guaranteed without better world placement sensors.

2) My chaise needs a redesign to allow a roller in the back as apposed to two dragging pivots, these pivots get caught far too easily and this lab became an order of magnitude harder due to it. My top priority in the following lab will be to adjust for this issue as well as do a better job accounting for collisions. I would like to add a response handler that utilizes bump sensors to adjust in the event of error beyond what can be handled by the encoders.

I feel my software design and algorithm approach went very well as when tested modularly without impact of non detectable chasie eros its ramping, precision, and accuracy were great. I was surprised however at how bad my chassis is with turning as the pivots get stuck on everything and would cause it to not move while wheels maintained rotation, tricking the encoders. I plan on rectify this with a rear pivot roller in the next lab.

References

PID library documentation: https://playground.arduino.cc/Code/PIDLibrary/

Starter Code

Appendices:

Fig 1: Encoder placement and robot designed

Fig 2: Encoder mount render

Fig 3: Arduino mount render

Table 1: Times of successful maze runs