	Intro To Robotics LAB #4 Report	2/22/2024
Jaskin Kabir		**jkabir@uncc.edu**
Hunter Burnett		**hburnet7@uncc.edu**

Lab Objective:

The objective of this lab was to understand the motor encoders and their role in governing accurate movement of the robot. Additionally, this lab explores error correction methods. In part A, the encoders are used to ensure that the robot rotates in place the correct number of degrees. In Part B, the encoders are used to maintain the robot's movement in a straight line.

Lab Figures/Tables (Testing Data):

Rotate function:

Firs the rotated function algorithm is developed as follows:

1. Convert degrees to distance
2. Compute the amount of encoder pulses to travel the given distance
3. Set the turn direction (CCW or CW)
4. Stop the motors when the encoder count reaches the target number of pulses
5. Wait 2 seconds

	Diameter	Circumference
Wheel	7 cm	7π cm
Base	14 cm	14π cm

In order to convert degrees to distance we need to measure the robot's diameter which was measured to be about 14 cm and the wheel diameter was measured to be 7 cm.

Now to calculate the circumference of the circle we can use the following formula:

$$circumference = 2\pi r$$

So when a wheel makes 1 turn it travels a distance of 7$\boldsymbol{\pi}$ cm. It is given that a wheel takes 360 encoder pulses to turn completely. So the distance a wheel can turn per encoder count can be computed using equation 2

$$\frac{7\pi\ cm}{1\ Wheel\ Turn} * \frac{1\ Wheel\ Turn}{360\ encoder\ pulses} = \frac{7\pi\ cm}{360\ pulses}$$

We can then use this to calculate the amount of pulses to turn given a rotation degree

$$distance = 14\pi\ cm * \frac{rotat\ deg}{360\ deg} * \frac{360\ pulses}{7\pi\ cm} = \frac{14 * rotate\ deg}{7} pulses$$

A rotation function was designed to turn the robot counterclockwise or clockwise and used a switch statement to determine the directions of the motors

Drive Straight Function

The algorithm for driving straight is as follows:

1. Calculate the number of pulses required to travel the given distance using the same formula as was used in the rotation function
2. Turn on both motors in the forward direction at the same speed
3. Continuously read both motor encoder counts
4. Subtract the right motor count from the left motor count
5. Calculate the appropriate adjustment based on this error using a PID algorithm
6. Add this adjustment to the left motor speed
7. If the left motor speed is below the minimum speed, snap it to the minimum speed
8. If the left motor speed is above the maximum speed, snap it to the maximum speed
9. Once both motor encoder counts reach the target number of pulses, stop both motors and wait 2 seconds

The PID algorithm is as follows

1. Multiply the current error by the proportional constant
2. Subtract the error measured from the last call to this function from the current error, divide the result by the time elapsed between measurements and then multiply by the derivative constant
3. Multiply the current error by the time elapsed between the current and last measurements, then add this to the running total of cumulative error
4. Multiply this running total by the integral constant
5. Add these three products together and return the adjustment value

A minimum speed of 5 for the left motor was chosen, as this is the minimum value required for the motor to be able to turn the wheel. The maximum speed was chosen to be 65 to allow the motor to react to error quickly, but not so quickly that it causes excessive oscillation.

The three PID constants were chosen using the following process.

1. Set all three constants to 0
2. Increase Kp until the robot oscillates about a straight line at a reasonable speed
3. Increase Kd until the oscillation is damped
4. Increase Ki until steady-state error is filtered out

Test	Kp	Ki	Kd	Behavior and Notes
1	0	0	0	Constant curve right
2	0.001	0	0	Too slow to notice change
3	0.006	0	0	Too slow to notice change
4	0.02	0	0	Still slow, but now there is oscillation
5	0.03	0	0	Oscillation ist oo slow to be considered stable
6	0.07	0	0	Still too slow
7	0.1	0	0	Too slow
8	0.2	0	0	Too Slow
9	0.4	0	0	Reasonable speed of oscillation, good baseline
10	0.4	0	0.0004	Way too low to have any effect
11	0.4	0	0.02	Still too low
12	0.4	0	1	Much slower oscillation, but same amplitude
13	0.4	0	2	Very little change
14	0.4	0	4	Very little change
15	0.4	0	10	Much better, but oscillation does not approach 0

16	0.4	0	15	Still better but there seem to be diminishing returns
17	0.42	0	15	Increased kp to create faster oscillation. Robot shakes rapidly
18	0.42	0	30	Oscillation is damped but does not approach 0
19	0.42	0	38	Seems to approach 0 now
20	0.42	0	42	Better but still oscillating
21	0.42	0	55	Better but seeing diminishing returns, and steady state error to the right
22	0.34	0	55	Reduced kp to reduce oscillation. Oscillation was reduced but steady state error increased
23	0.42	0.02	55	Integral is way too high, causing rapid oscillation
24	0.42	0.0002	55	Reduced ss error
25	0.42	0.0004	55	Reduced ss error but increased oscillation

26	0.42	0.0006	60	Oscillation is too high, reduce kp and ki
27	0.396	0.0005	60	Drives very straight, but still oscillates
28	0.396	0.0005	64	More oscillation
29	0.396	0.0005	49.8	Drastic measures, decrease kd. Straightest the robot has ever been

With these values– Kp=0.396, Ki=0.0005, Kd=49.8, the robot moved very straight, with just a 3cm deviation from a straight path across a 5m distance.

Commentary and Conclusion:

The rotation function implemented in this lab did not use any error correction to account for the difference in rotation between the two wheels. Even though this function lacked this correction, the resulting rotation was quite accurate to the number of degrees entered into the function.

As for driving the robot forwards in a straight line, the error correction method used was somewhat overengineered. This could have been accomplished with a constant error correction that was not dependent on the amount of error at all, but we were interested in PID and saw this lab as an opportunity to learn more about this technique. When the three PID parameters were properly tuned, not only did the robot drive straight but it was very quick to react to extreme disturbances like abruptly pushing one of the wheels. After this disturbance, it returned to a straight line quickly and effectively.

Overall, this experiment was successful in fulfilling its objectives. The motor encoders were effectively used to govern the robot's motion, and we were able to explore PID control with a real-world application.