Team 10:
Matthew Haahr
Brian Shin
Nick Hom

RBE 2002: Unified Robotics II—Lab 1: Velocity Control Post-Lab

1. **PI Controller Block Diagram**

Figure 1.1:

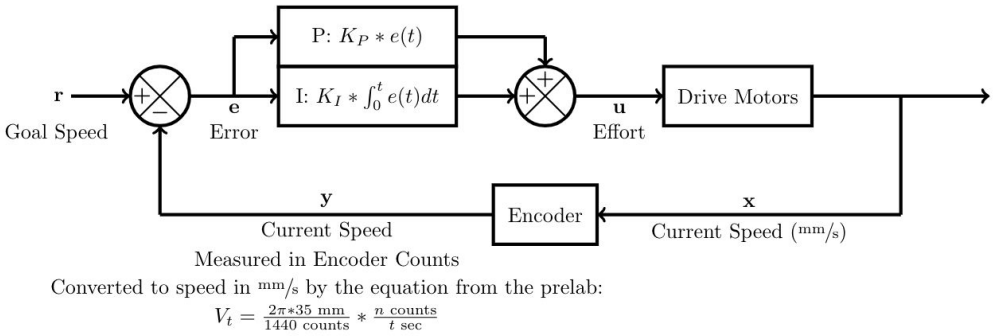

Figure 1.1: PI Controller Block Diagram

Figure 1.2:

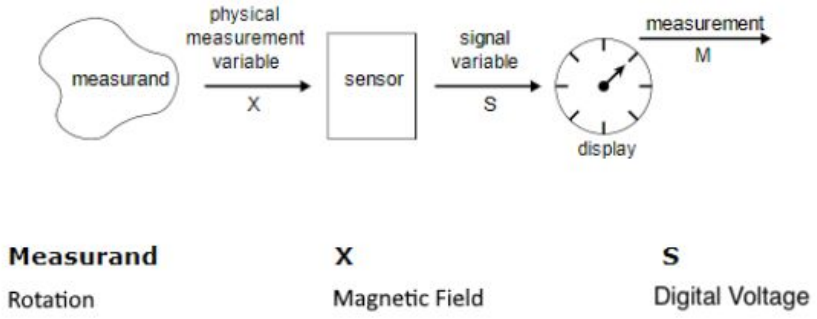

Figure 1.2: Measured Quantity

a. To measure the speed of the wheels for our robot, we measured the number of changes in the magnetic field of the disk attached to the back of the motor over a known time interval. To do this we are looking for changes on the encoder data lines, which indicate that the motor has turned a specific distance and by applying the known number of encoder counts per rotation to the number of changes over a given time, we can find the angular velocity of the wheel.

2. **Effort of 100 for 10 Seconds**
    a. Plots

Figure 2.1:

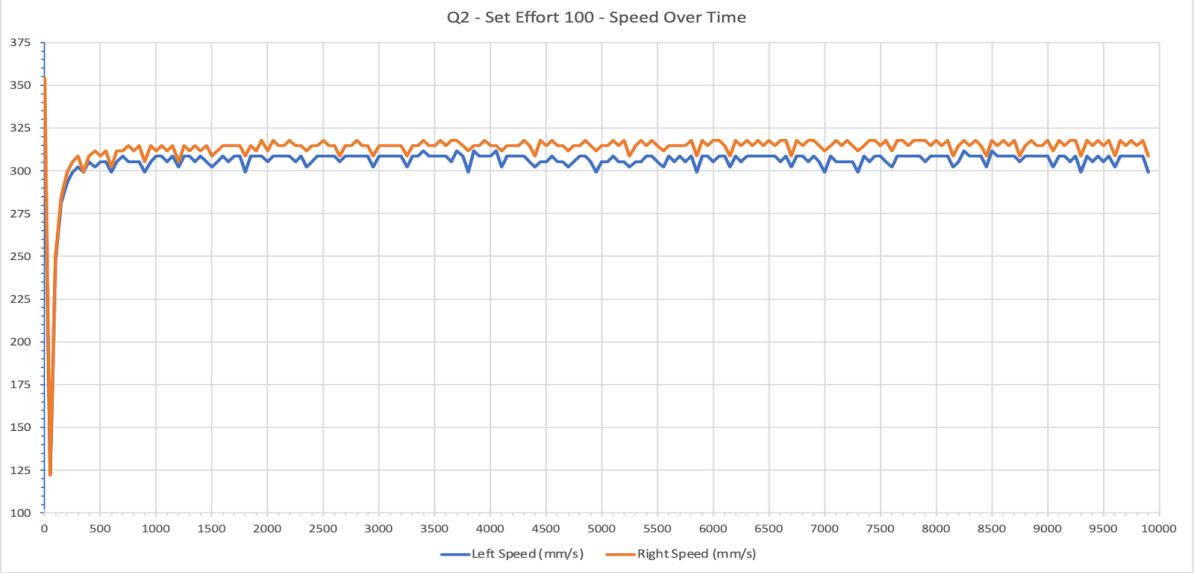

Figure 2.1: [Matthew] Wheel speed vs. time for applying a constant effort of 100 for 10 seconds.

Figure 2.2:

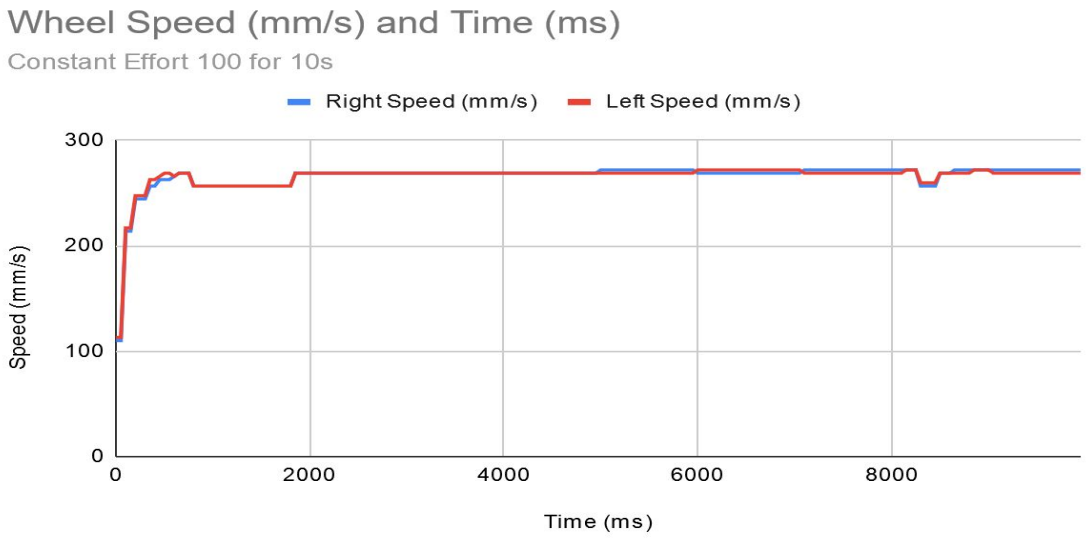

Figure 2.2: [Brian] Wheel speed vs. time for applying a constant effort of 100 for 10 seconds.

Figure 2.3:

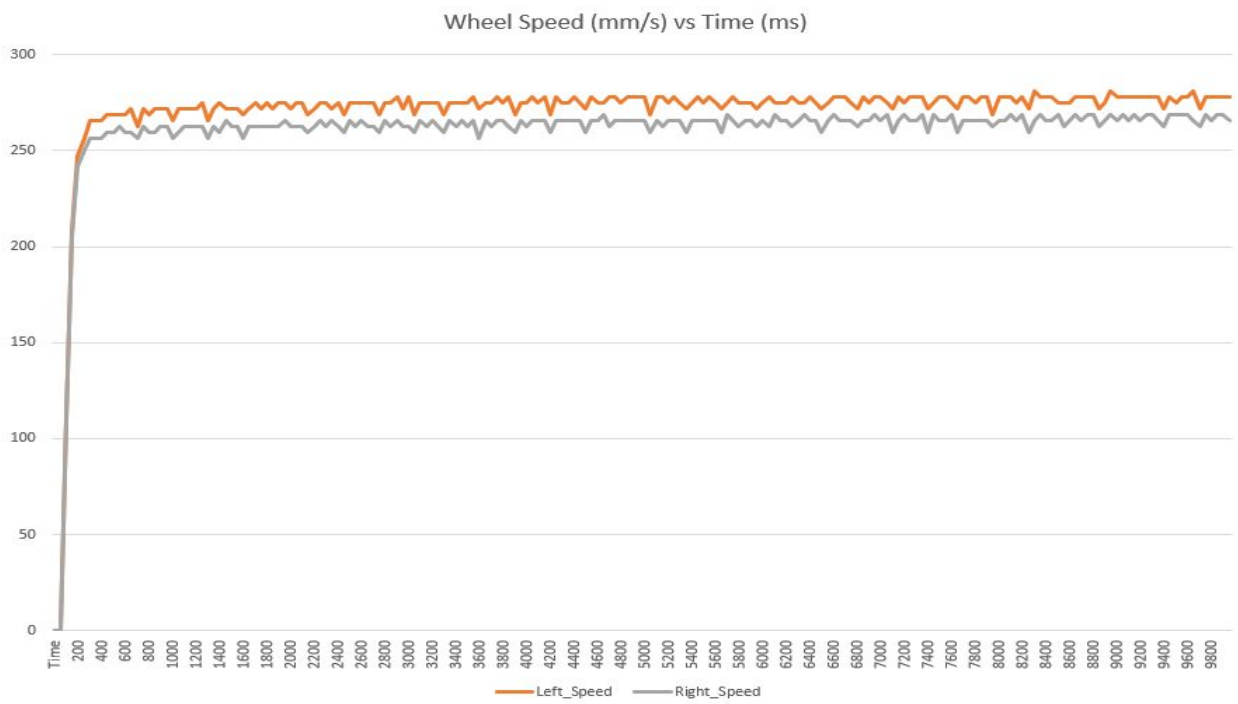

Figure 2.3: [Nick] Wheel speed vs. time for applying a constant effort of 100 for 10 seconds.

b. Calculations

	Average Left Speed	Left Speed Standard Dev	Average Right Speed	Right Speed Standard Dev
Matthew	305.34 (mm/s)	14.30 (mm/s)	312.99 (mm/s)	15.15 (mm/s)
Brian	265.27 (mm/s)	16.87 (mm/s)	265.61 (mm/s)	17.5 (mm/s)
Nick	271.97 (mm/s)	23.45 (mm/s)	261.72 (mm/s)	22.31 (mm/s)

3. **50 mm/s for 10 Seconds with $K_P = 0.3$, $K_I = 0$**
    a. Plots

Figure 3.1.1:

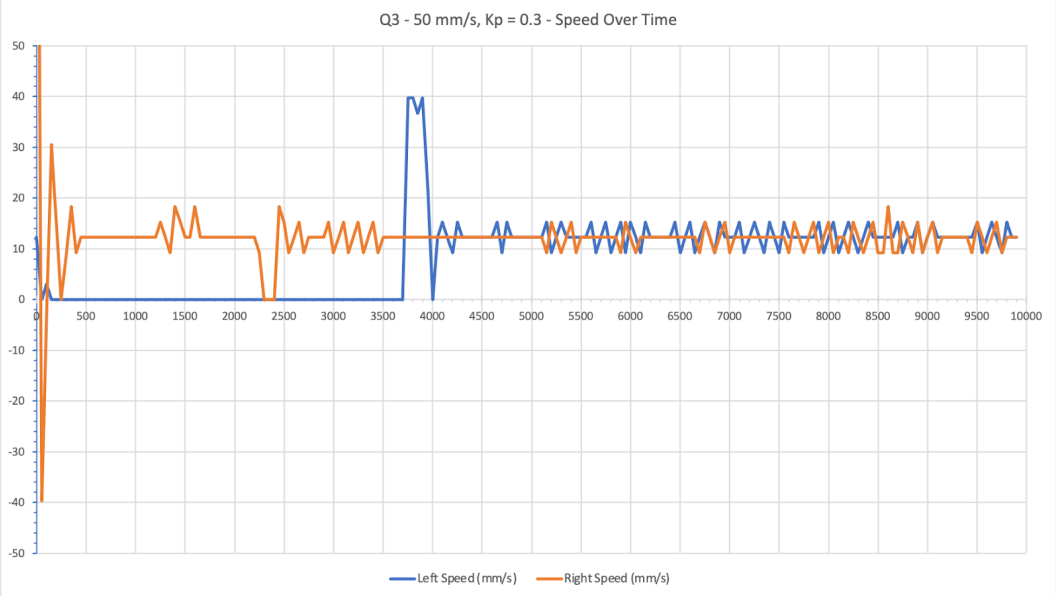

Figure 3.1.1: [Matthew] Wheel speed vs. time for 50mm/s target at PI gain of {0.3, 0}

Figure 3.1.2:

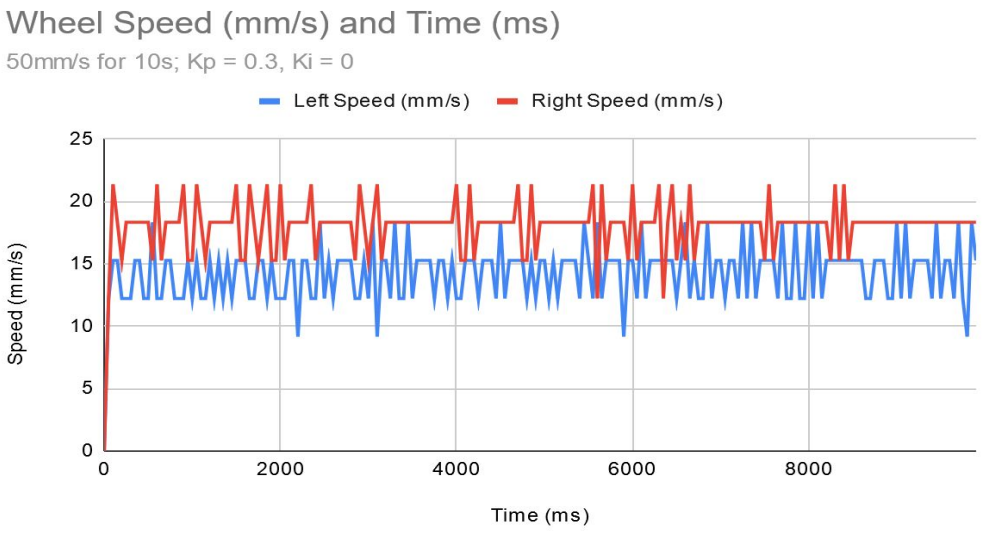

Figure 3.1.2: [Brian] Wheel speed vs. time for 50mm/s target at PI gain of {0.3, 0}

Figure 3.1.3:

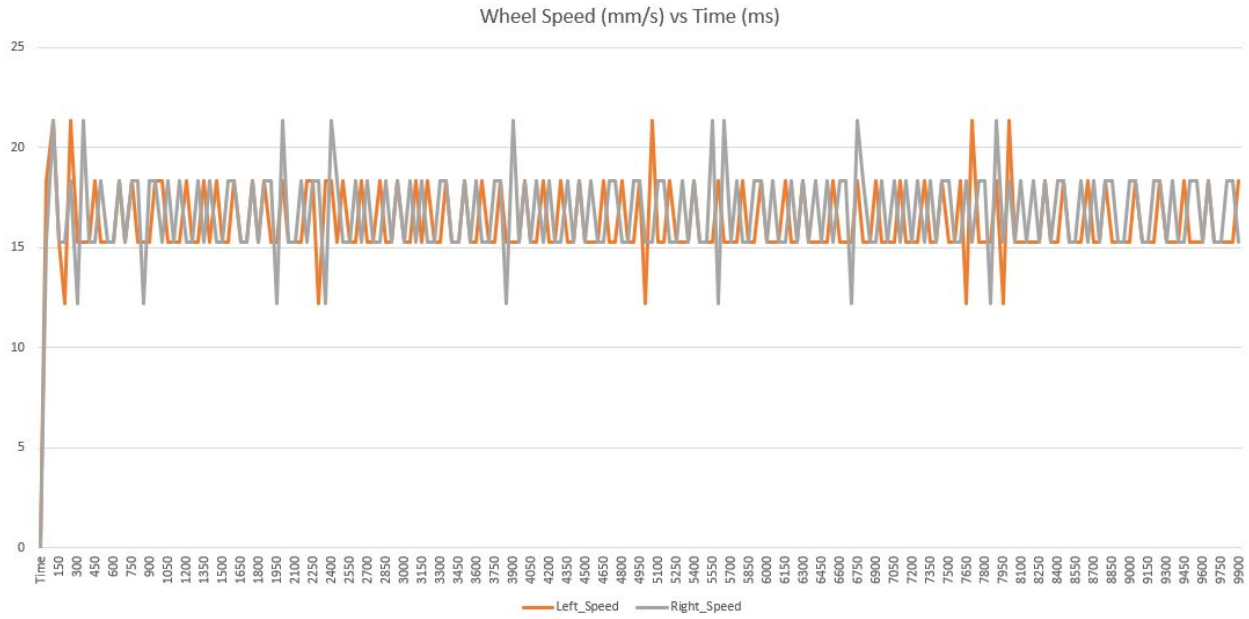

Figure 3.1.3: [Nick] Wheel speed vs. time for 50mm/s target at PI gain of {0.4, 0} (Increased Kp until both wheels were able to spin which is why the value is higher than the original instruction)

Figure 3.2.1:

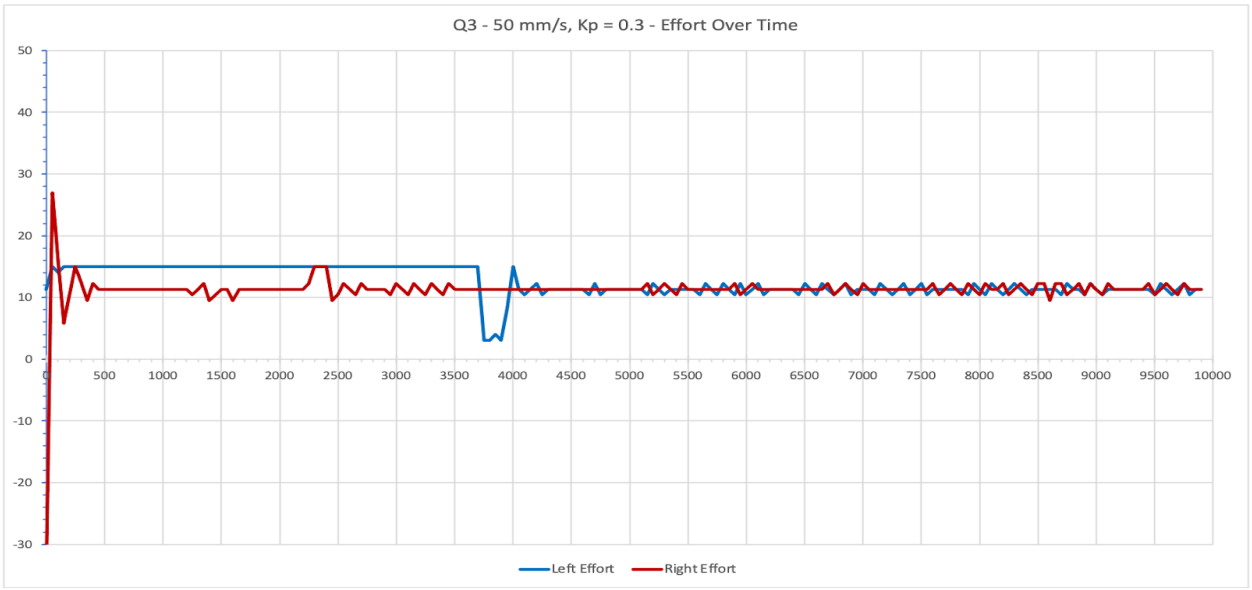

Figure 3.2.1: [Matthew] Effort for both wheels vs. time for 50mm/s target at PI gain of {0.3, 0}

Figure 3.2.2:

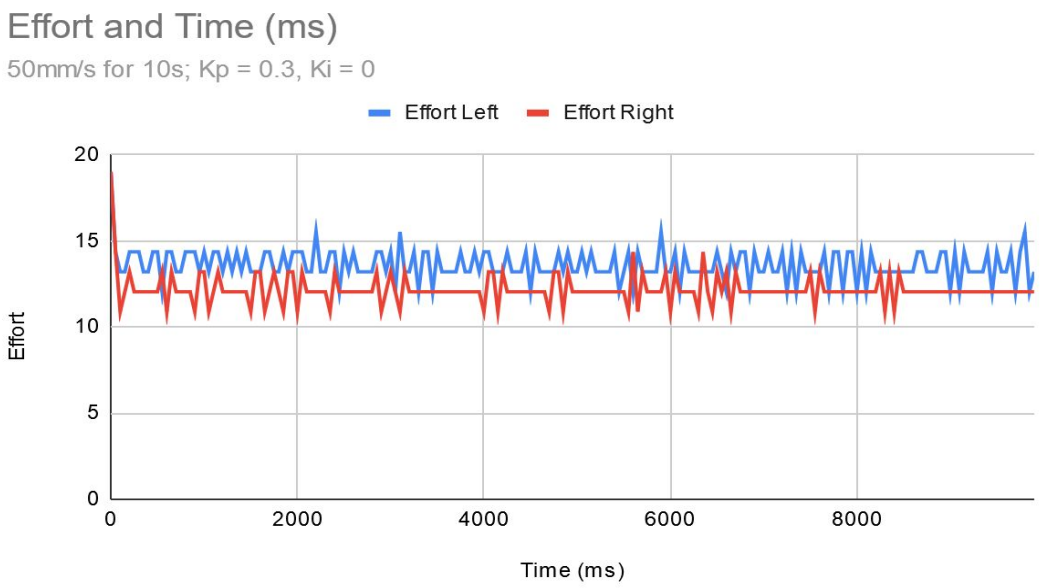

Figure 3.2.2: [Brian] Effort for both wheels vs. time for 50mm/s target at PI gain of {0.3, 0}

Figure 3.2.3:

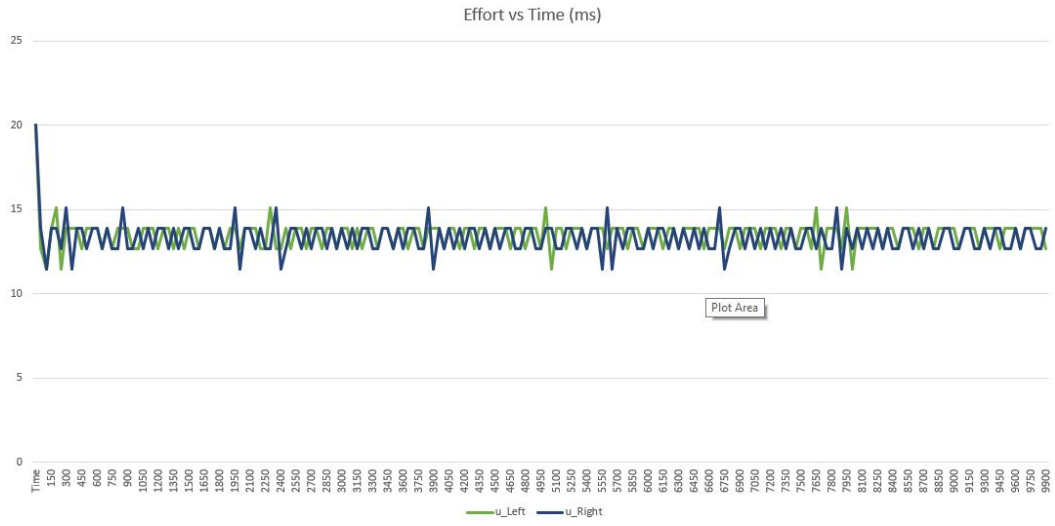

Figure 3.2.3: [Nick] Effort for both wheels vs time for 50mm/s target at PI gain of {0.4, 0} (Increased Kp until both wheels were able to spin which is why the value is higher than the original instruction)

Figure 3.3.1:

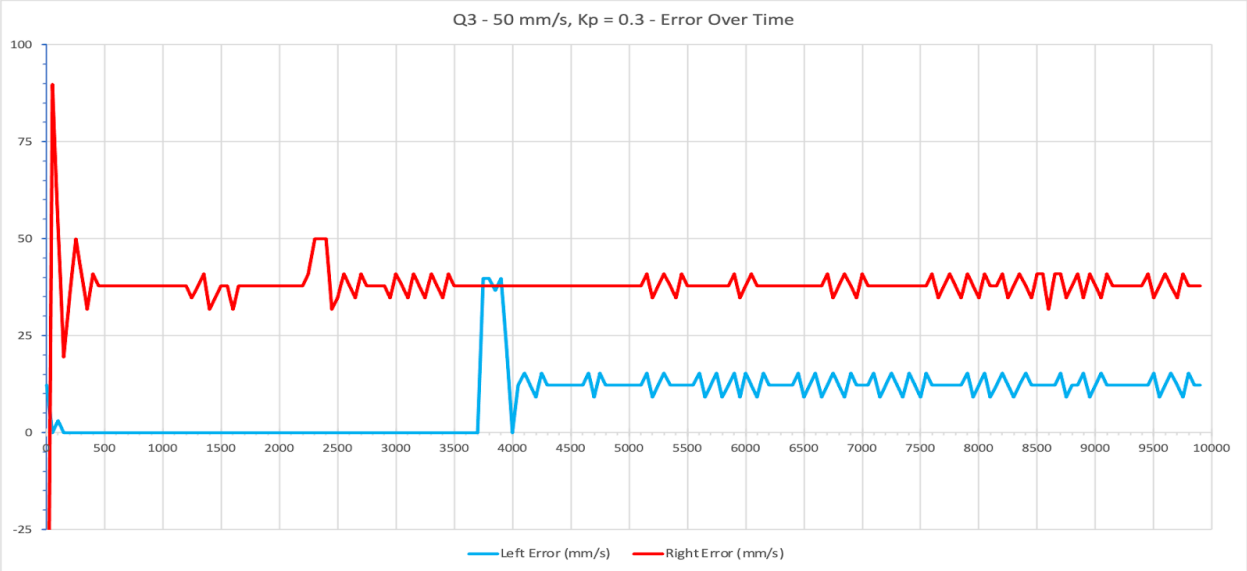

Figure 3.3.1: [Matthew] Velocity error vs. time for 50mm/s target at PI gain of {0.3, 0}

Figure 3.3.2:

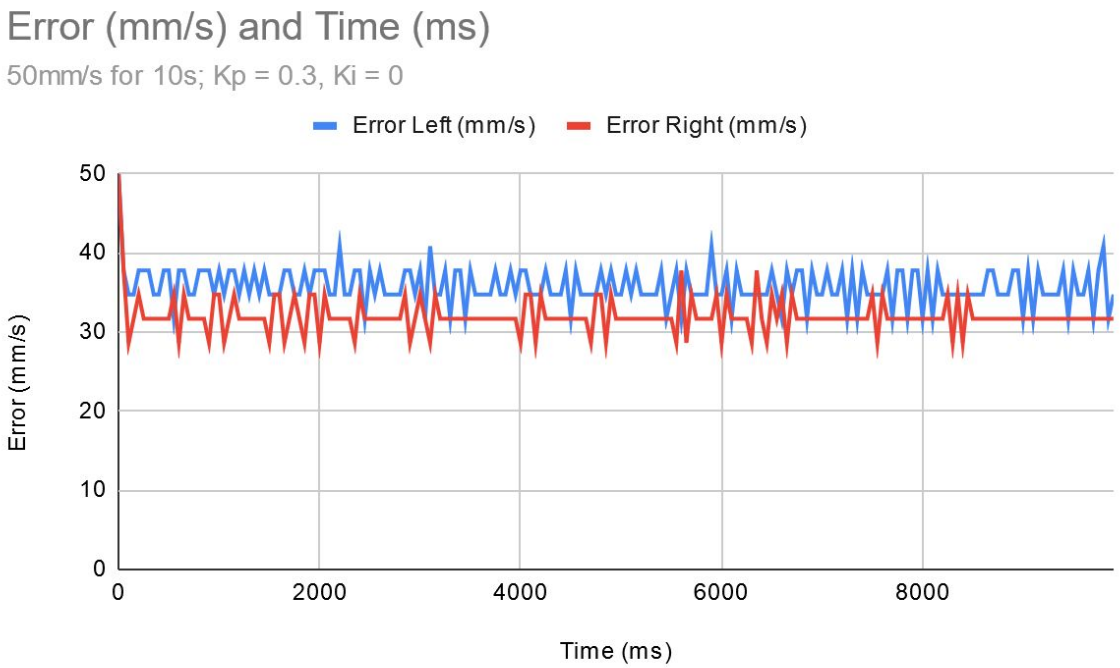

Figure 3.3.2: [Brian] Velocity error vs. time for 50mm/s target at PI gain of {0.3, 0}

Figure 3.3.3:

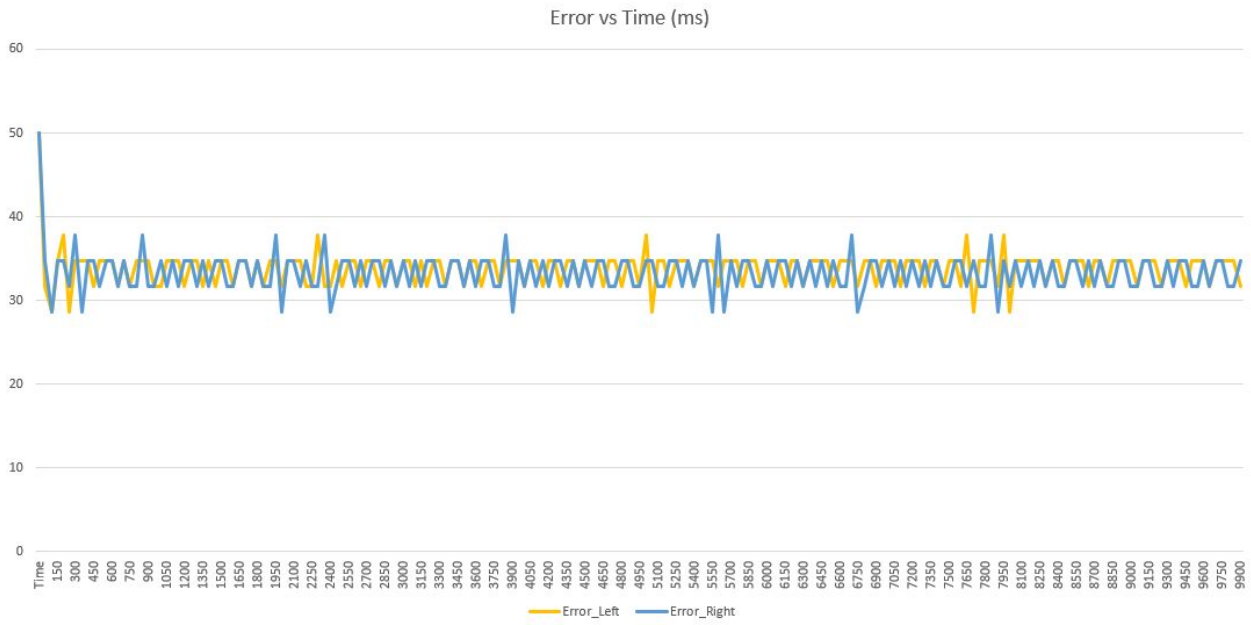

Figure 3.3.3: [Nick] Velocity error vs. time for 50mm/s target at PI gain of {0.4, 0} (Increased Kp until both wheels were able to spin which is why the value is higher than the original instruction)

b. Why does the system not reach the target velocity?
    i. The system does not reach the target velocity due the lack of steady-state error compensation. A purely proportional system will decrease in error and thus output effort as the error decreases, eventually falling to within the motor deadband and effectively stopping the motors. To correct a steady-state error of this type, the integral term is introduced, which compensates for the difference between the goal state and zero state so that the proportional part of the controller deals with difference from the goal.

4. **50 mm/s for 10 Seconds with K_P = 10, K_I = 0**
    a. Plots

Figure 4.1.1:

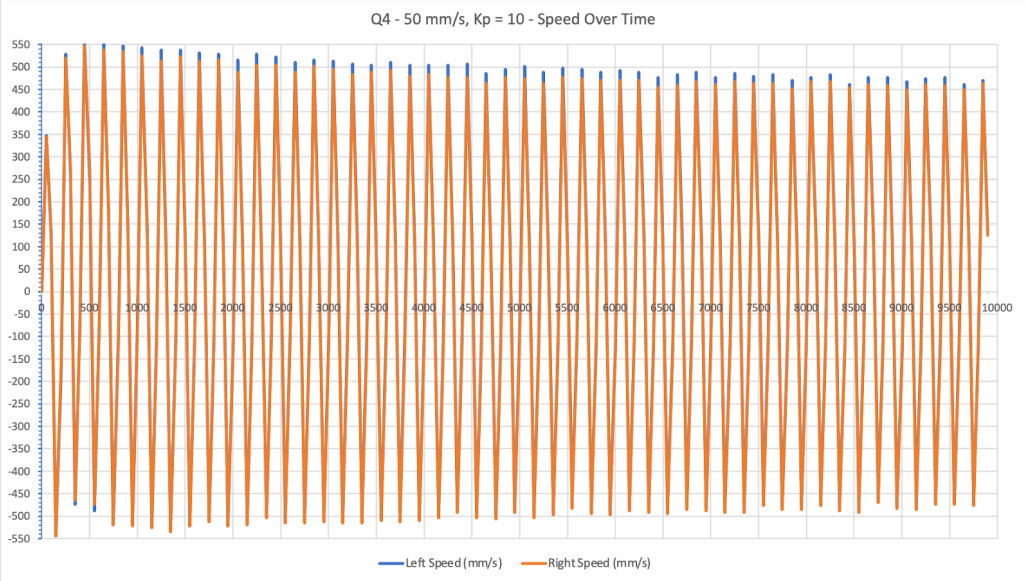

Figure 4.1.1: [Matthew] Wheel speed vs. time for 50mm/s target at PI gain of {10, 0}

Figure 4.1.2:

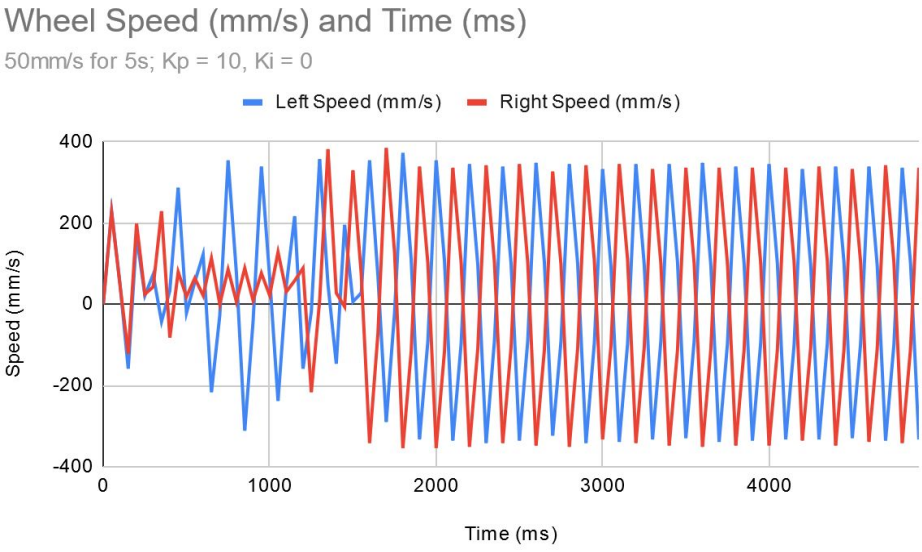

Figure 4.1.2: [Brian] Wheel speed vs. time for 50mm/s target at PI gain of {10, 0}

Figure 4.1.3:

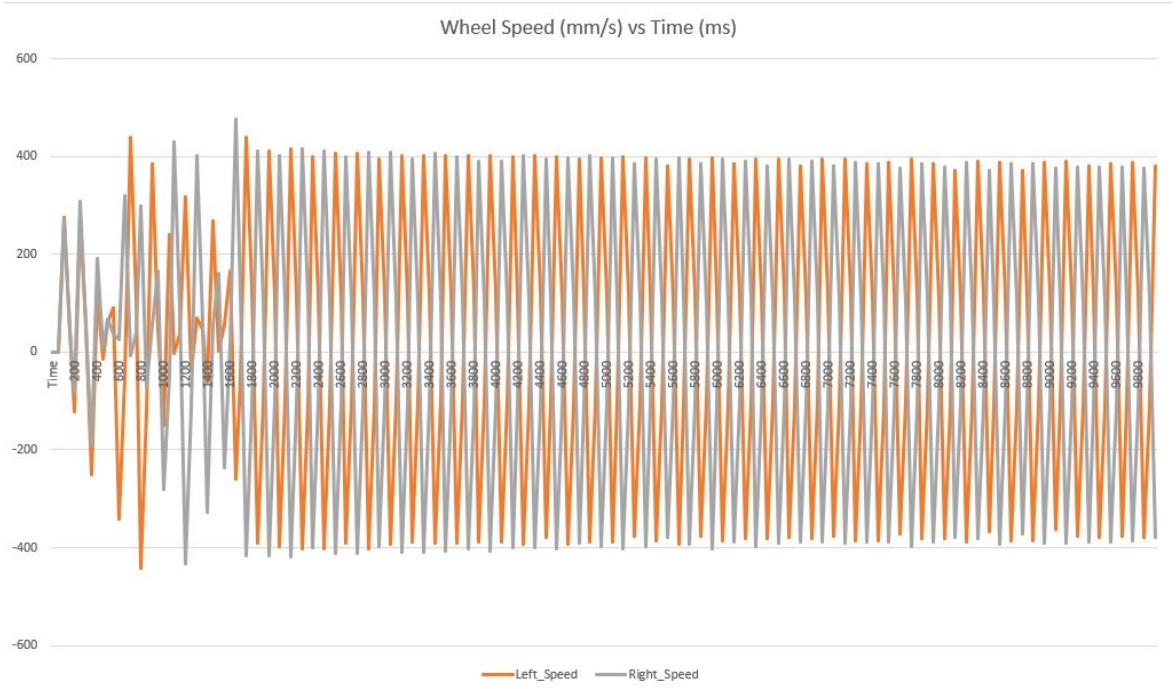

Figure 4.1.3: [Nick] Wheel speed vs. time for 50 mm/s target at PI gain of {10, 0}

Figure 4.2.1:

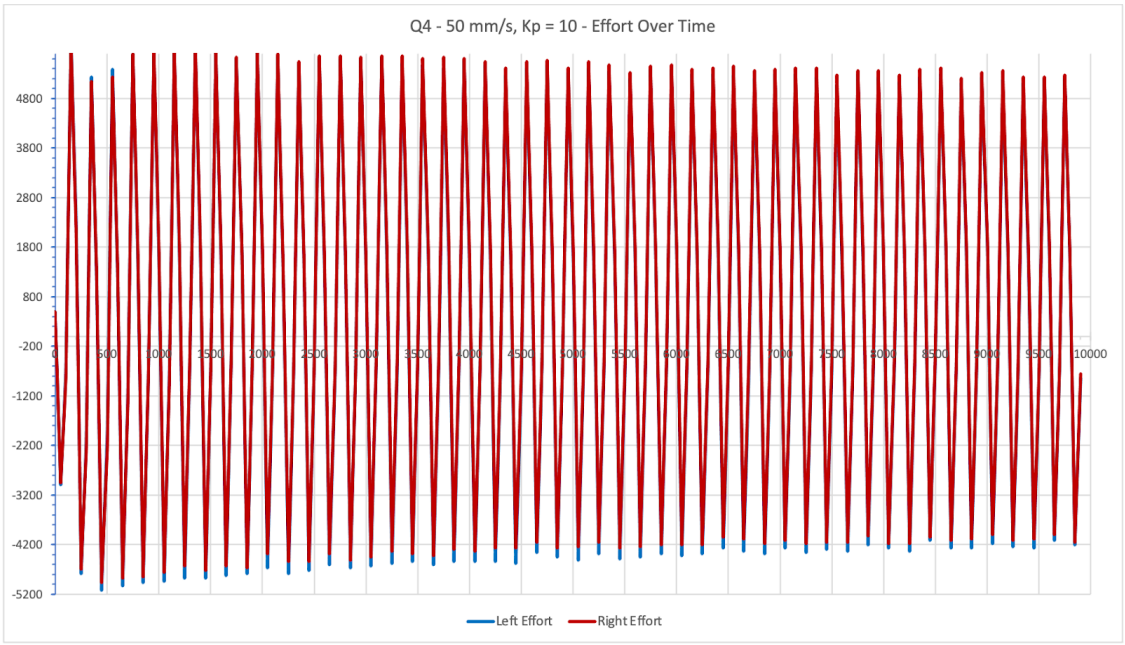

Figure 4.2.1: [Matthew] Effort for both wheels vs. time for 50mm/s target at PI gain of {10, 0}

Figure 4.2.2:

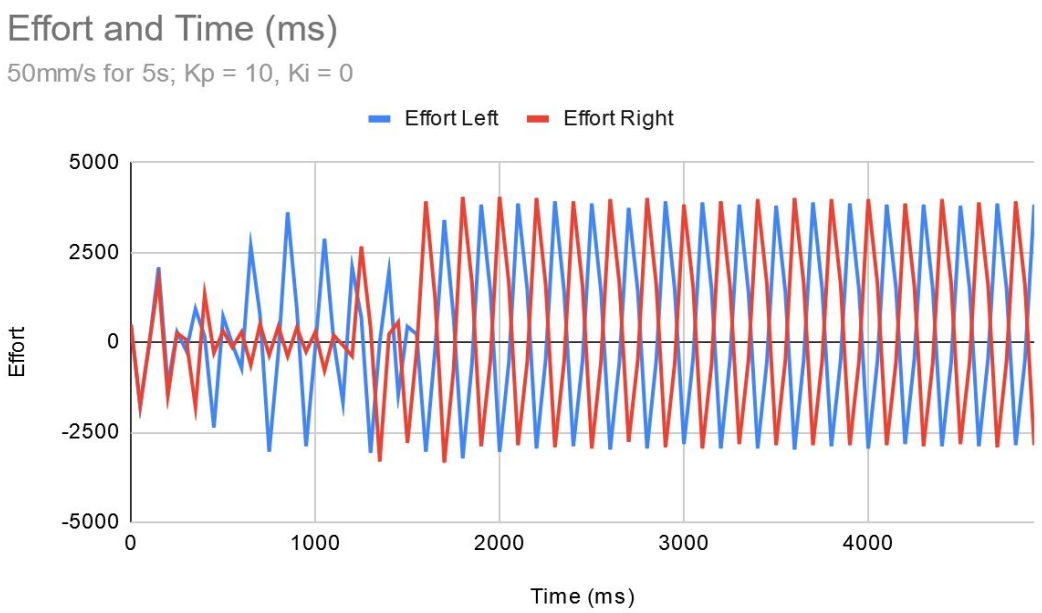

Figure 4.2.2: [Brian] Effort for both wheels vs. time for 50mm/s target at PI gain of {10, 0}

Figure 4.2.3:

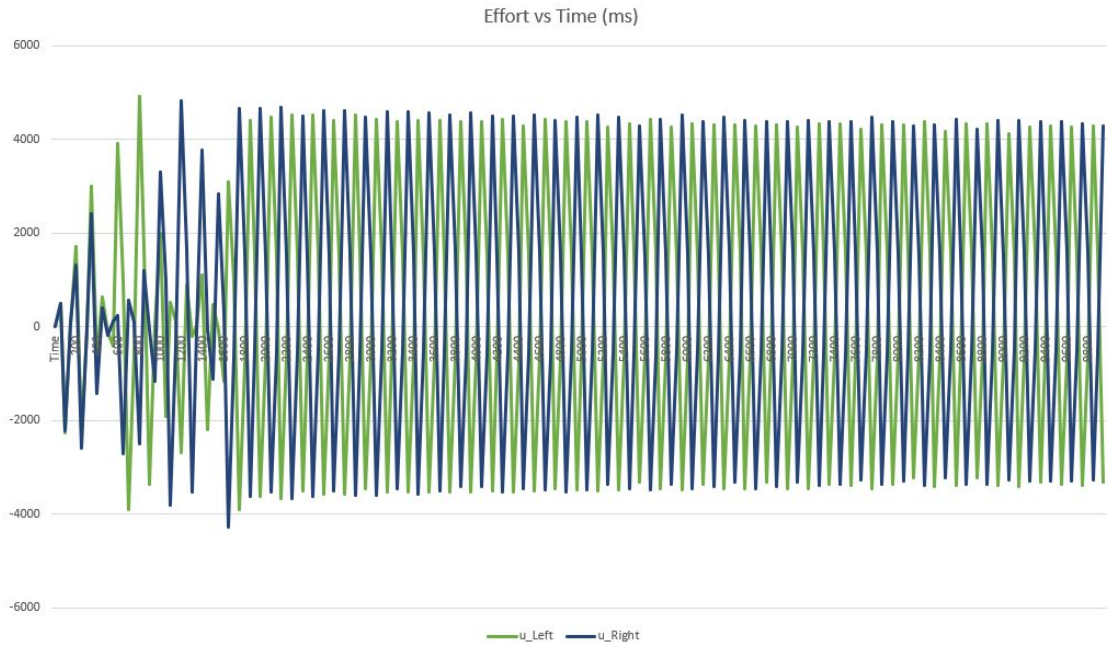

Figure 4.2.3: [Nick] Effort for both wheels vs. time for 50mm/s target at PI gain of {10, 0}

Figure 4.3.1:

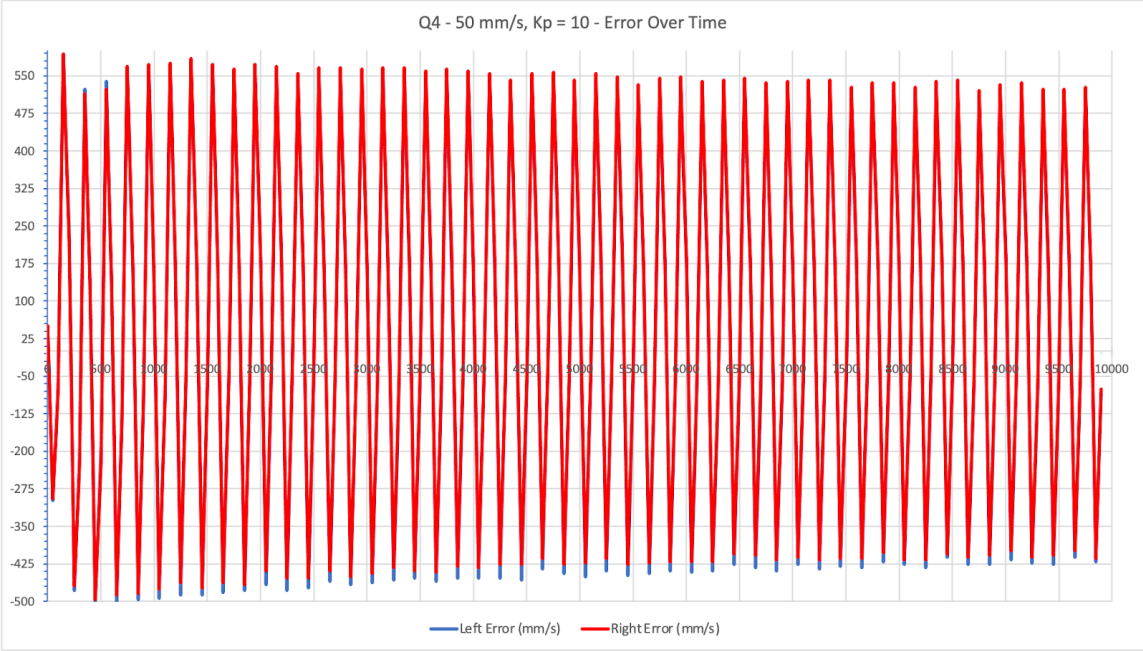

Figure 4.3.1: [Matthew] Velocity error vs. time for 50mm/s target at PI gain of {10, 0}

Figure 4.3.2:

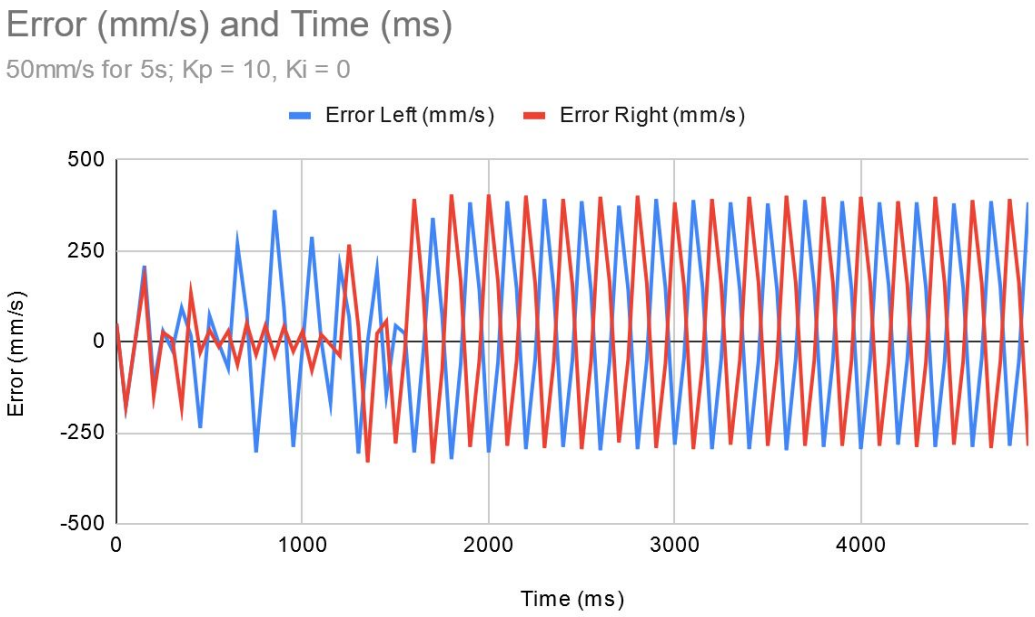

Figure 4.3.2: [Brian] Velocity error vs. time for 50mm/s target at PI gain of {10, 0}

Figure 4.3.3:

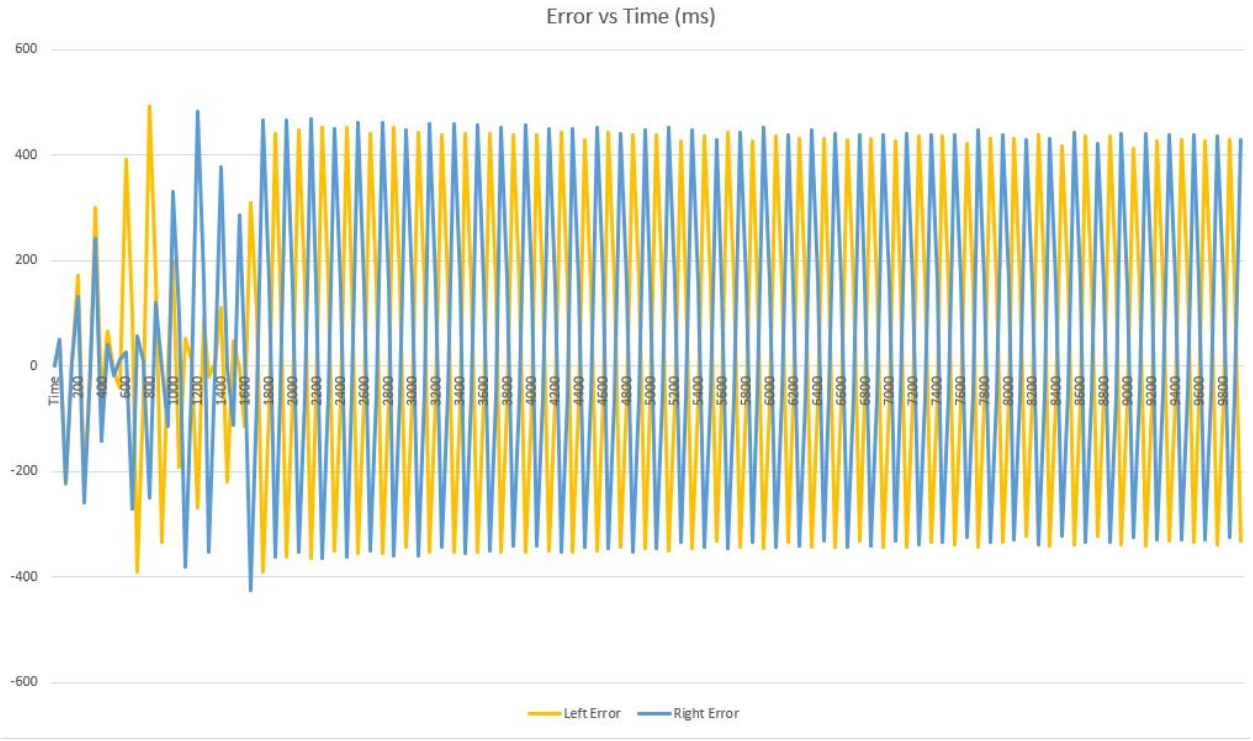

Figure 4.3.3: [Nick] Velocity error vs. time for 50mm/s at PI gain of {10, 0}

b. Why does the system oscillate?

   i. The system oscillates because the proportional gain is set so high, so any error is increased in magnitude and thus the corresponding effort is even higher, driving the motors faster and faster. This oscillation continues, it grows and grows until it is unstoppable. This is an example of a positive feedback loop. A system like this is called “unstable.”

5. **50 mm/s for 10 Seconds with K_P = 0.3, K_I = 0.1**
    a. Plots

Figure 5.1.1:

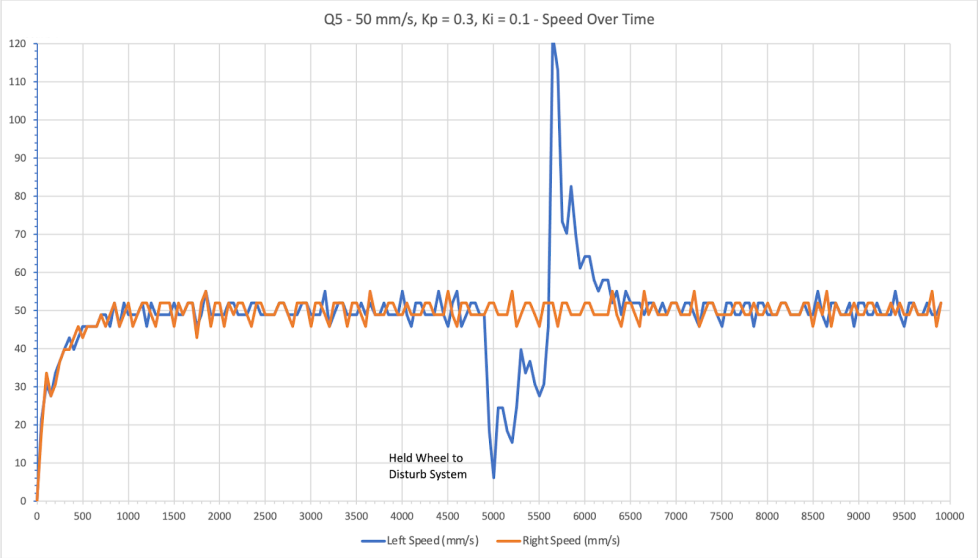

Figure 5.1.1: [Matthew] Wheel speed vs. time for 50mm/s target at PI gain of {0.3, 0.1}

Figure 5.1.2:

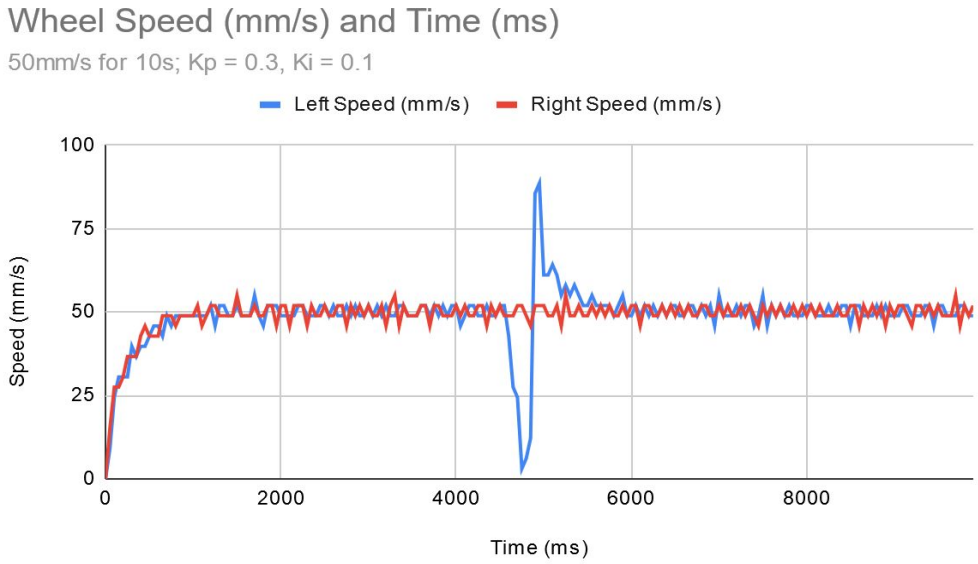

Figure 5.1.2: [Brian] Wheel speed vs. time for 50mm/s target at PI gain of {0.3, 0.1}

Figure 5.1.3:

Figure 5.1.3: [Nick] Wheel speed vs time for 50mm/s target at a PI gain of {0.3, 0.1}

Figure 5.2.1:

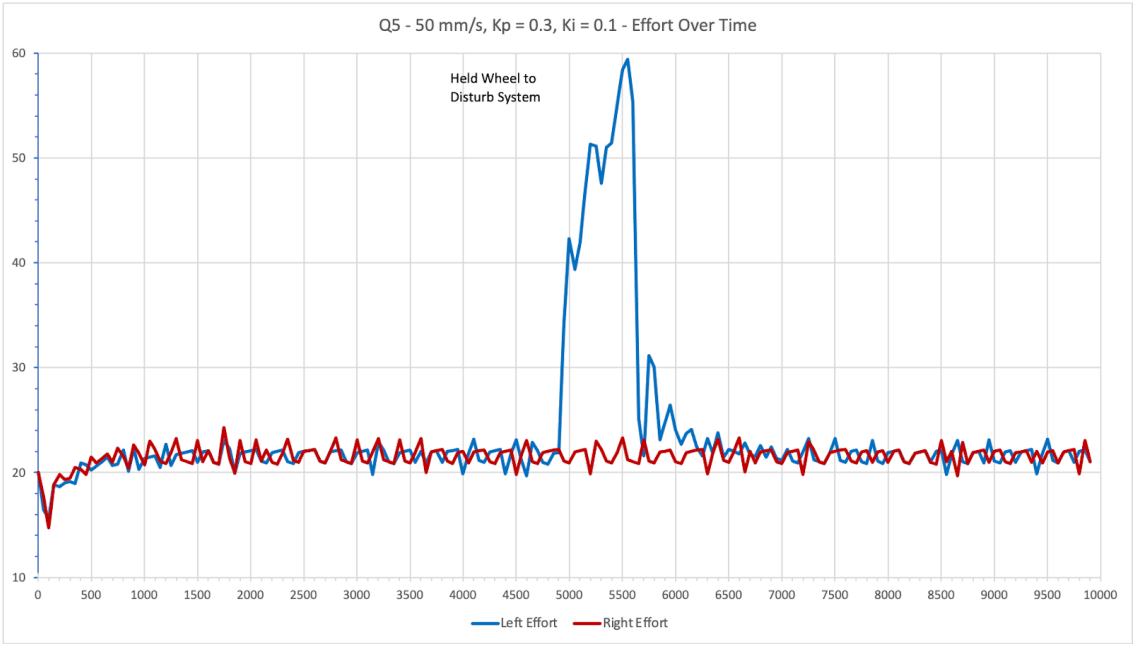

Figure 5.2.1: [Matthew] Effort for both wheels vs. time for 50mm/s target at PI gain of {0.3, 0.1}

Figure 5.2.2:

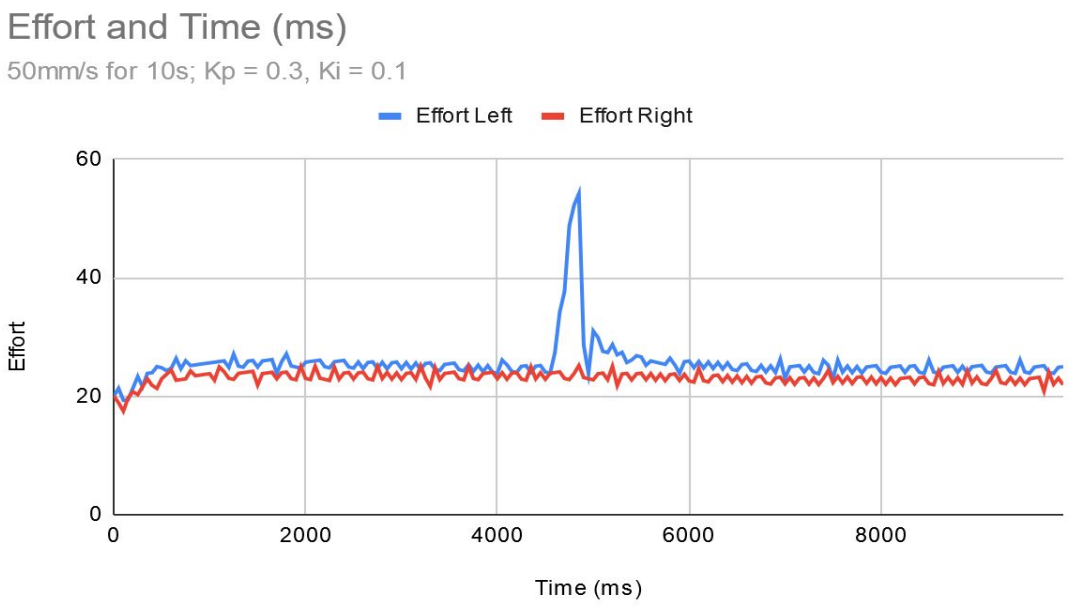

Figure 5.2.2: [Brian] Effort for both wheels vs. time for 50mm/s target at PI gain of {0.3, 0.1}

Figure 5.2.3:

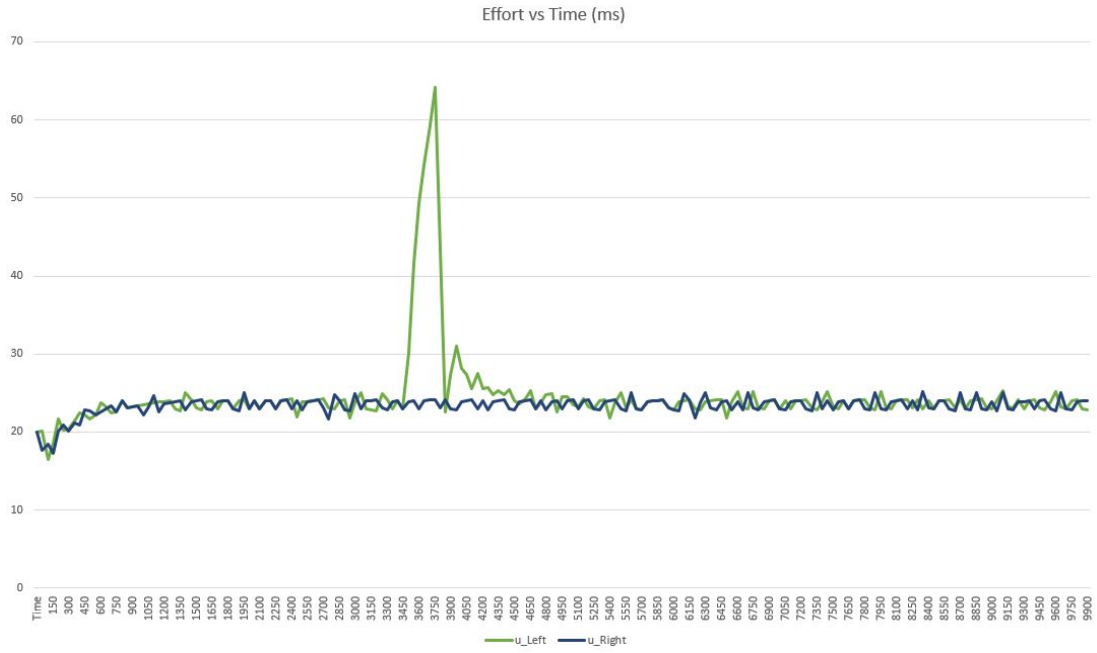

Figure 5.2.3: [Nick] Effort for both wheels vs. time for 50mm/s target at PI gain of {0.3, 0.1}

Figure 5.3.1:

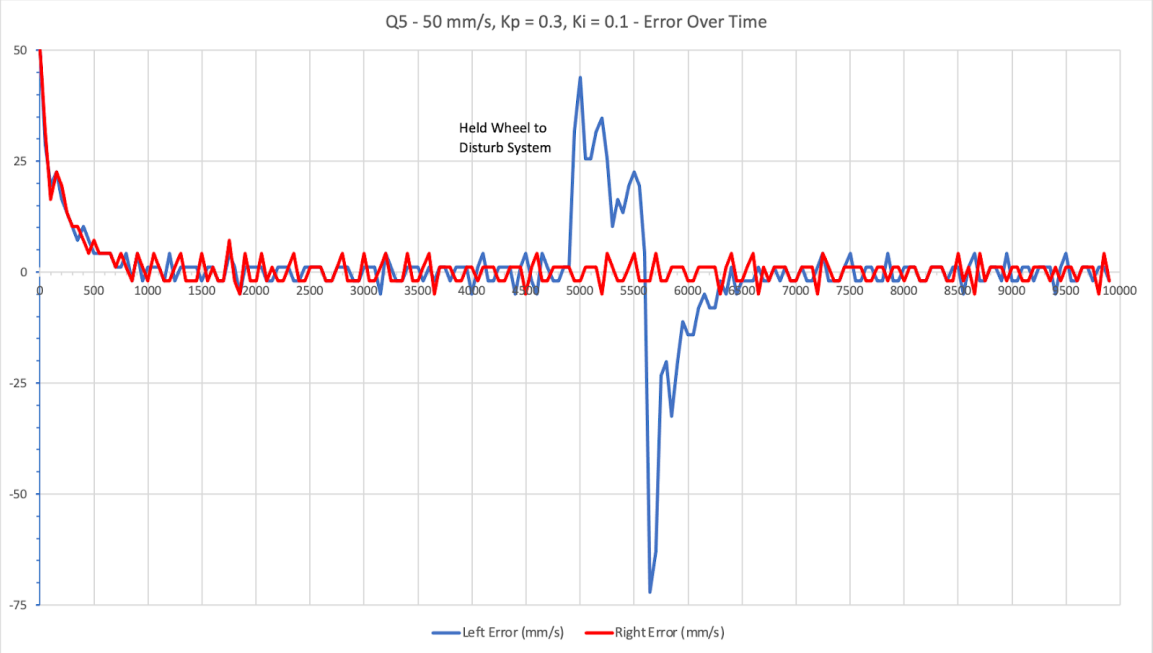

Figure 5.3.1: [Matthew] Velocity error vs. time for 50mm/s target at PI gain of {0.3, 0.1}

Figure 5.3.2:

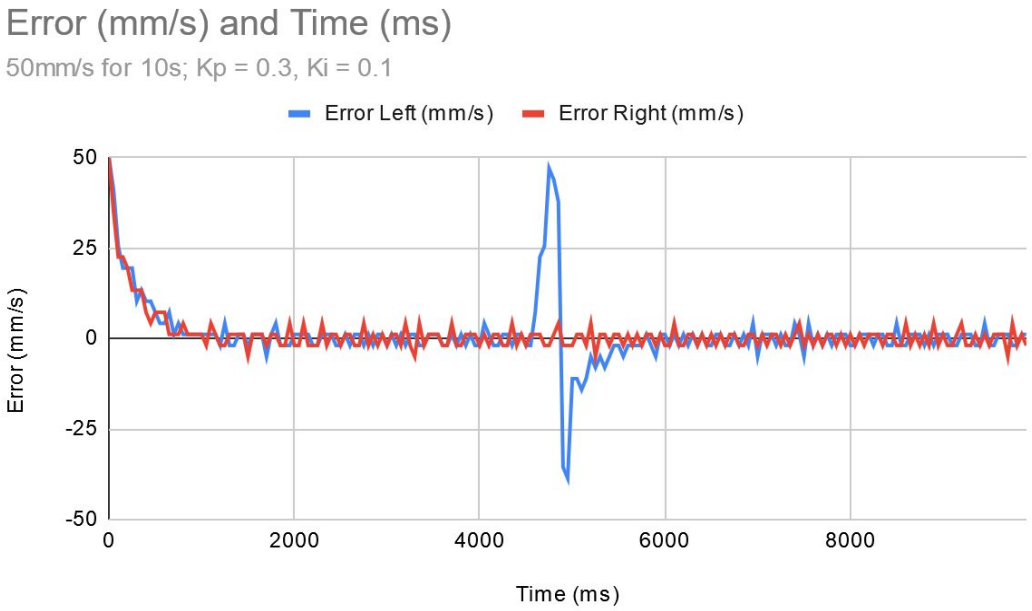

Figure 5.3.2: [Brian] Velocity error vs. time for 50mm/s target at PI gain of {0.3, 0.1}

Figure 5.3.3:

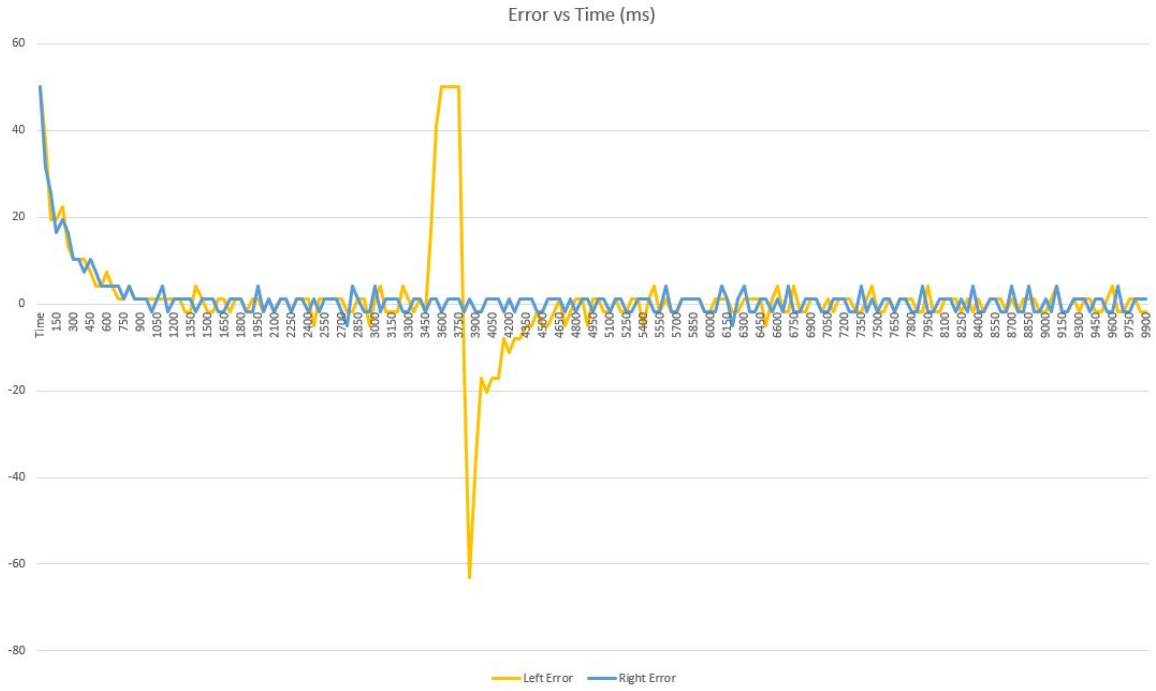

Figure 5.3.3: [Nick] Velocity error vs. time for 50mm/s target at PI gain of {0.3, 0.1}

b. Explain behavior of Speed, Error, and Effort before and after disturbing the system

   i. As reflected in the graphs, the motors start from no motion and ramp up to the desired velocity, as seen in Figures 5.1.X, and this initial start-up condition is also reflected in the increase at the beginning of the Effort graphs, or the decrease in the Error graphs. The system has achieved a steady state after this initial start-up, as seen in all three properties roughly aligning to a constant value. Upon disturbing the system, there is a “spike” in the Speed where it swiftly decreases, and accompanying increase “spikes” in Error and Effort at this moment. The system must compensate for this disturbance and within the next moment, there are peaks in the opposite direction in each respective property: the Speed overshoots, effort increases in the direction opposite the disturbance, and the Error increases in the negative direction. After this compensation, the

system ramps back down in a series of increasingly smaller peaks and returns to a steady state.

6. **10 mm/s for 2, 4, 6, 8, and 10 Seconds Distances**
    a. Matthew

Time	2 Sec	4 Sec	6 Sec	8 Sec	10 Sec
Distance Travelled	15 mm	31 mm	57 mm	73 mm	100 mm
Error	5 mm	9 mm	3 mm	7 mm	0 mm

    b. Brian

Time	2 Sec	4 Sec	6 Sec	8 Sec	10 Sec
Distance Travelled	14 mm	30 mm	51 mm	72 mm	93 mm
Error	6 mm	10 mm	9 mm	8 mm	7 mm

    c. Nick

Time	2 Sec	4 Sec	6 Sec	8 Sec	10 Sec
Distance Travelled	15 mm	33 mm	53 mm	69 mm	90 mm
Error	5 mm	7 mm	mm	11 mm	10 mm

    d. Explanation For Error
        i. As seen in the tables for each trial of this experiment, the robot never quite made it to the target distances. This is due to the speed of the robot being considerably low (almost within the deadband of these Pololu motors), and the majority of the driving being completed by the integral controller accumulating. This also produced an oscillation in the speed of the robot. Even with both proportional and integral control, the robot was not fast enough nor traveling at a consistent enough speed to make the distance in the given time.

7. **Turning Calculations with One Motor Driving and One Stopped**

Wheel diameter: 70mm (35mm radius); wheel track: 140.34mm; 1440 counts per revolution of wheel. Conceptually, when turning with only one wheel (and the other stopped), the moving wheel is tracing a fraction of the circumference of a circle with a radius equal to the width of the robot (wheel track). Arc length: S = rΘ (in radians)

a. π Radian Turn

$$Counts = \pi * 140.34mm * \frac{1\ rev}{2\pi * 35mm} * \frac{1440\ counts}{1\ rev}$$

2887 counts [2886.99]

b. π/2 Radian Turn

$$Counts = \pi/2 * 140.34mm * \frac{1\ rev}{2\pi * 35mm} * \frac{1440\ counts}{1\ rev}$$

1444 counts [1443.49]

c. π/4 Radian Turn

$$Counts = \pi/4 * 140.34mm * \frac{1\ rev}{2\pi * 35mm} * \frac{1440\ counts}{1\ rev}$$

722 counts [721.75]

8. **How Does a PI Controller Compensate for Motors Spinning at Different Speeds for the Same Applied Effort**

a. Unfortunately, not all motors are made the same and the motors on the Romi do not spin at the same speeds regardless if they are set to the same effort. A proportional-integral controller could compensate for this error by ensuring each wheel was moving at the same *velocity*. By doing so, the differences between the motors can be eliminated since this controller adjusts the effort for each wheel independently based on the angular speeds of each wheel. For example, if Motor A moved at 90% of the speed of Motor B at the same effort, then the PI controller would have Motor A move an extra 10% faster to compensate for the difference, because the controller calculates an appropriate effort for each wheel such that each wheel spins at the same speed. In this example, Motor A would be set to a higher effort than Motor B to have the robot drive straight. This behavior is part of

the function of the integral portion of the controller, eliminating a steady-state error.