Robotics Lab 5

Motor Balancing

Joel Boynton

July 15, 2025

1 Analyzing the Code

1.1 What lines of code or variable represents the reference signal / input?

```
float targetSpeed = 15;
```

- Above represents our reference signal. It is the speed we want our right motor to achieve

1.2 What lines of code represent the controller?

```
float rightError = leftCount - rightCount;

int rightSpeed = targetSpeed + rightError * kp;
```

- These lines listed above represent the controllers for the right motor. They control how the right speed is corrected in speed based on error between the left and right motors

1.3 What represents the feedback? How does this work into the error term?

```
float rightCount = getEncoderRightCnt();
```

- The line stated above represents our feedback. These are actual measurements of the wheel movement. This feedback of wheel movements allows us to actually compute the error. This can then be used in our corrections.

1.4 Which variable represents the system gain? How does the controller use the system gain?

```
float kp = 0.1;
```

- The variable listed above represents the system gain. This determines how strongly the controller reacts to an error. The controller uses this by adding the product of the error and gain to the target speed. Thus creating the correction.

2 Experiment 1

`Set the gain variable to 0, build, upload, and run the code. Consider how the robot is driving.`

2.1 What effect does setting the gain to 0 have on the system? Does this completely negate the effects of the control system?

- Setting the gain to 0 makes it so the robot will not adjust if one motor moves faster/slower than the other. So, yes. It effectively disables the corrective mechanism that the control uses.

3 Experiment 2

`Set the gain variable back to 0.1, build, upload, and run the code again. Contrast the results of this value with the results of experiment 1.`

3.1 Does this drive more straight, or nearly the same? What effect did this value have on the code?

- It definitely drives more straight. By giving the gain a value above zero we provide the robot with a corrective mechanism that can make minute adjustments.

4 Experiment 3

`Set the gain variable to 0.25, build, upload, and run the code again. If you felt like last time appeared about the same, you should now observe a noticeable difference. However, you might occasionally notice the robot jerk quickly from time to time when compensating.`

4.1 Does this perform better or worse than the results in Experiment 2?

- To me I believe it performs a little worse than the results in Experiment 2, as you can see it overcompensate a little bit.

5 Experiment 4

Set the gain variable to 0.5, build, upload, and run the code again. Note the behavior of the robot. It is likely jerking around wildly as each wheel attempts to catch up to the other in short bursts. This is what we would call an unstable control system, brought about by the gain value. The error is being amplified too much and the motors are now overcompensating for their errors.

This is an issue that can be solved with more complex controllers. In many cases, a proportional control won't be able to satisfy the requirements for a control system. In this application, we can likely adjust the value of the gain empirically through trial and error to find the "sweet spot" value, where the system reacts enough to make the robot drive straight, but not so much that the system becomes unstable.

5.1 What value would you guess would be a sweet spot from your observations in these experiments? Try that value on the robot in one last run and record the results.

- I thikn **0.15** would be a good sweet spot. Upon trying the value it seemed to drive pretty straight. There were vert few instances of compensation that were particularly noticeabe.