Self-Balancing Robot

“Lenny”

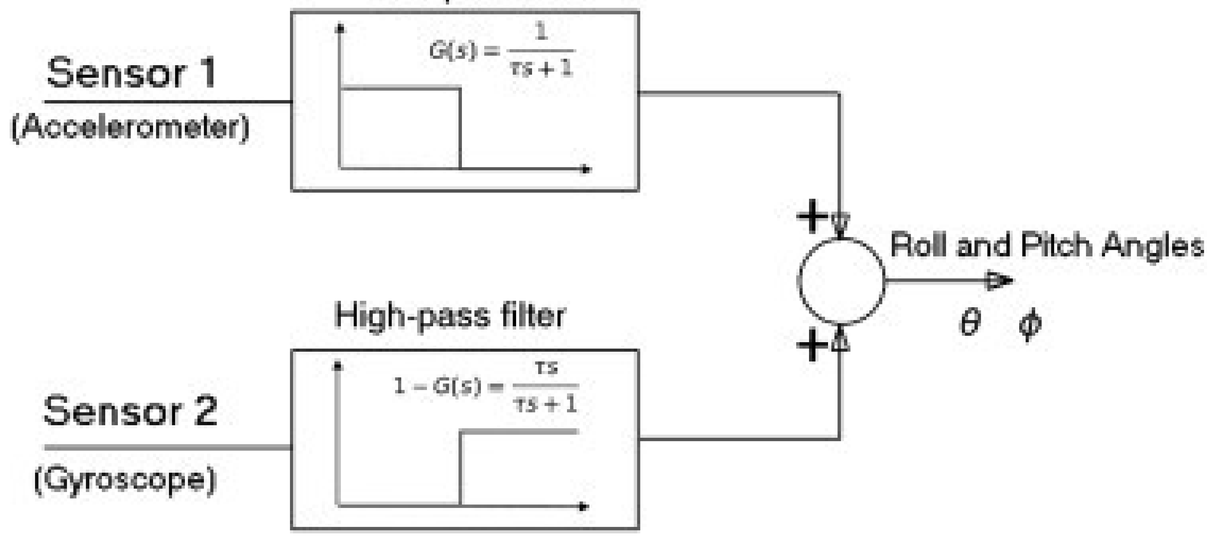
The Control Loop

The control loop is the main loop which runs the robot. First, the accelerometer and gyroscope data is loaded and passed through the complimentary filter. Then, the adjusted roll angle is passed to the PID configuration, which gives us a value of how much we need to move the wheels.

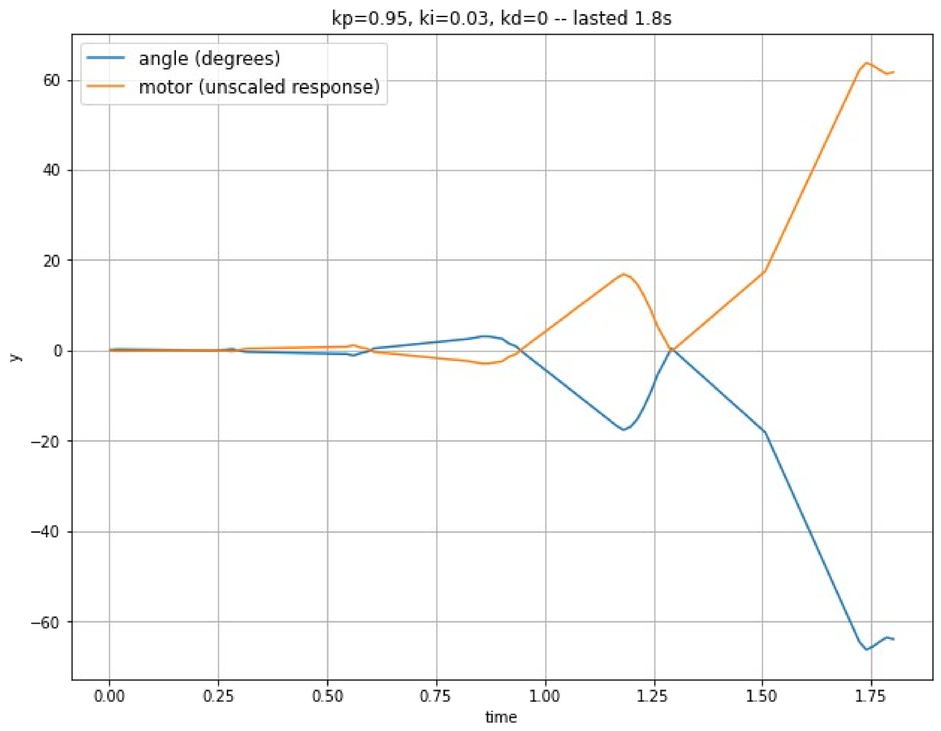
We set out to build a robot based on the inverted-pendulum model, that being having 2 wheels, and able to balance and self-correct on it’s own without outside help.

Complimentary Filter

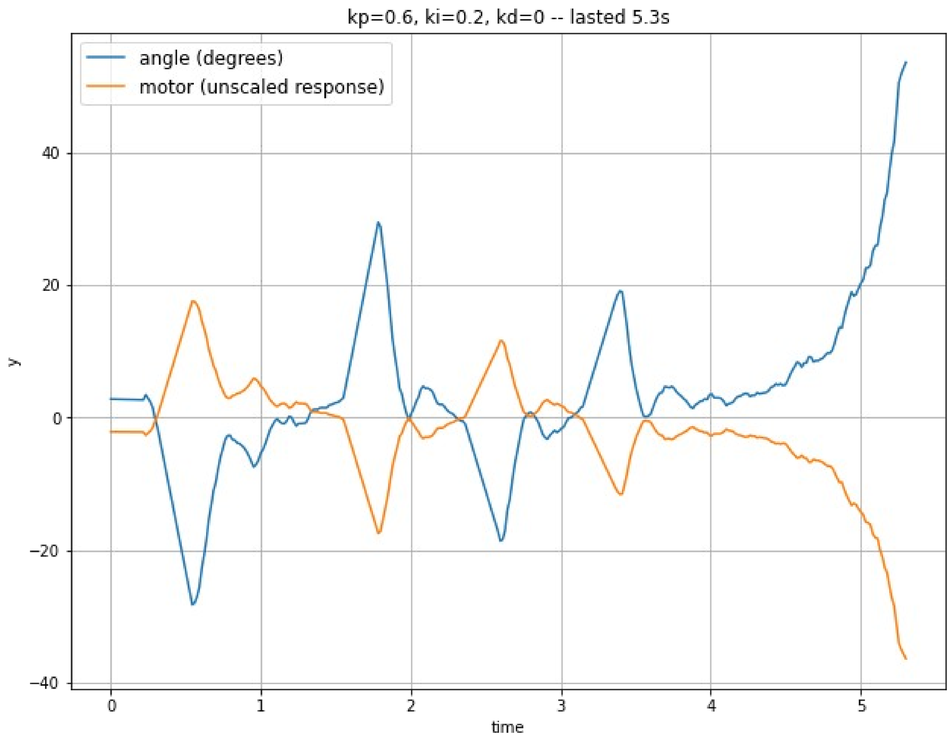
A complimentary filter takes the gyroscope and accelerometer data in real-time to filter out the forces caused by the movement of the robot. It uses the acceleration due to gravity as a reference.



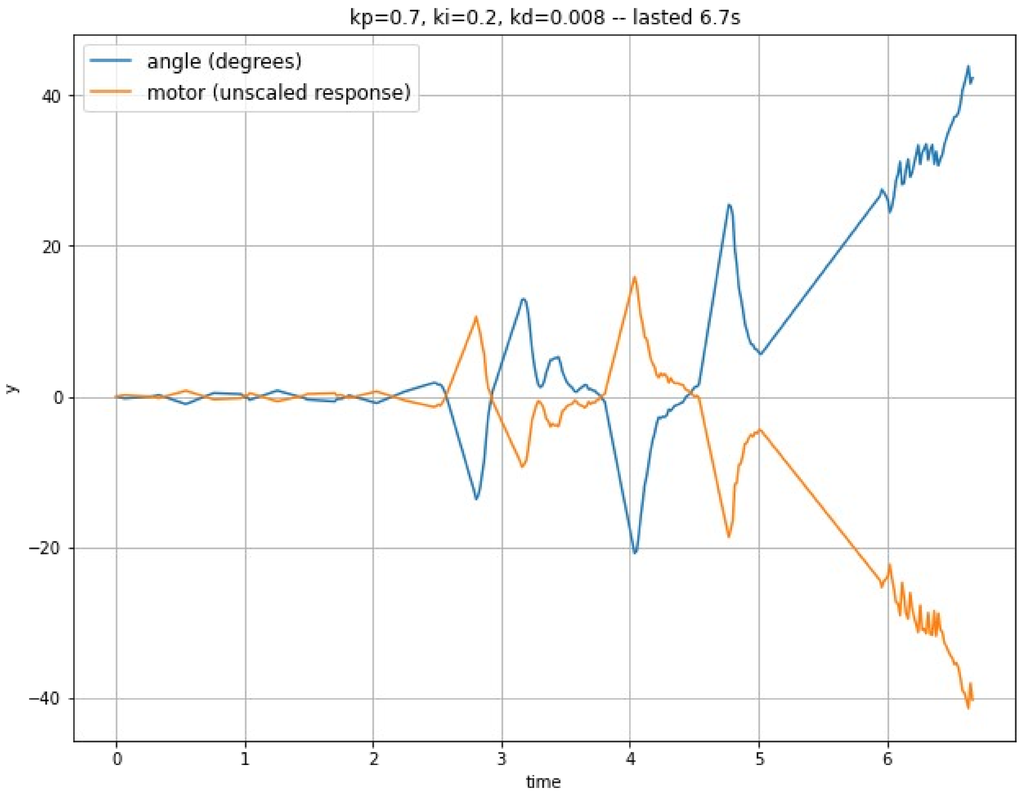
This graph shows a PID configuration that over-corrects and quickly loses control



This graph shows a PID configuration that corrects too quicly, yet remains balanced for a decent amount of time

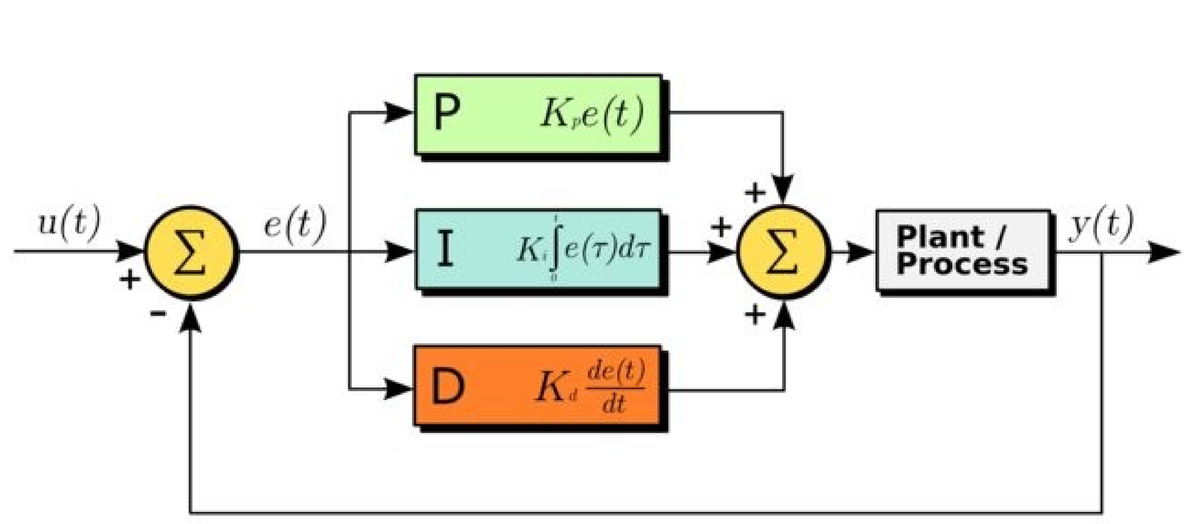


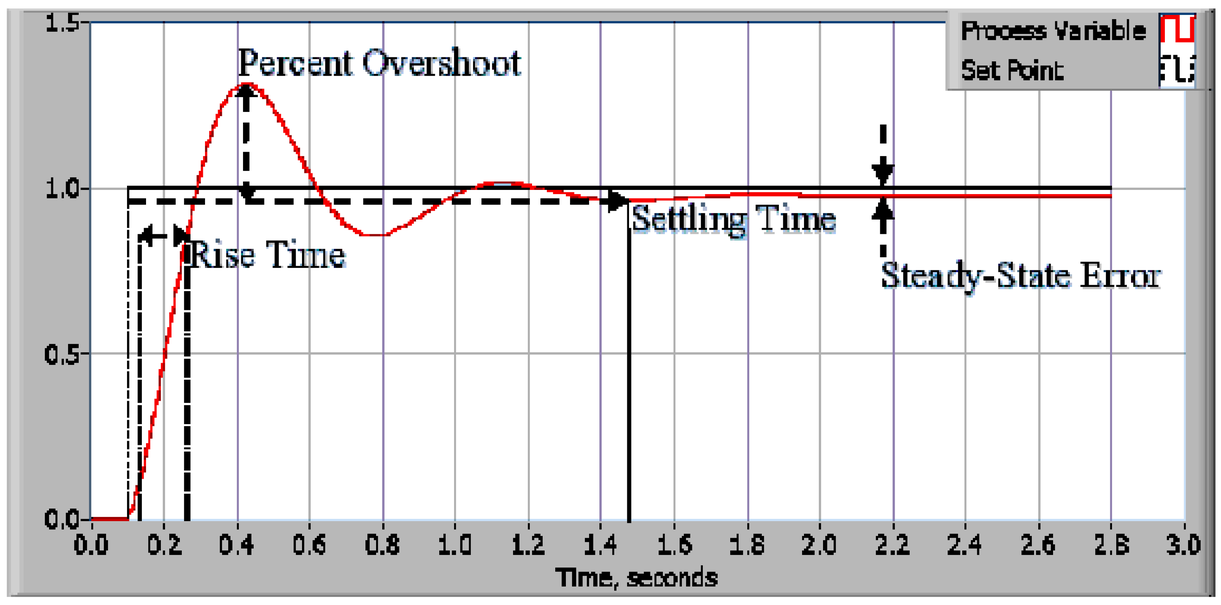
This graph shows a PID configuration that self-corrects well, but under-corrects just a little



PID

PID, or Proportial-Integral-Derivative, is a method of gradually aligning one variable to some value via some process. In this case, we are trying to align the accelerometer data (after being passed through the complimentary filter) to 0 (a balanced standing position) via moving the wheels.





Conclusion

In the end, we were able to get the robot to balance on it’s own and self correct, but only for a limited amount of time. We learned that while the complimentary filter and PID were relatively easy to implement, tuning the PID parameters to not over or under correct proved difficult. We think that if we were to do this project again, testing the code rigorously at every step might prove valuable, as well as converting our variables to pre-defined units. We discovered that a lot of our values were arbitrary data that wasn’t particularly easy to understand how to debug.