Lab 8: Basic Self Balancing Robot

A) Objectives

- To choose the right IMU data and set-point as the feedback for motion control
- To tune PID controller in maintaining balance for the robot on a spot
- To pick different set-point(s) to move the robot forward and backward without toppling
- (optional) To cascade wheel velocity control loop to achieve forward and backward motion

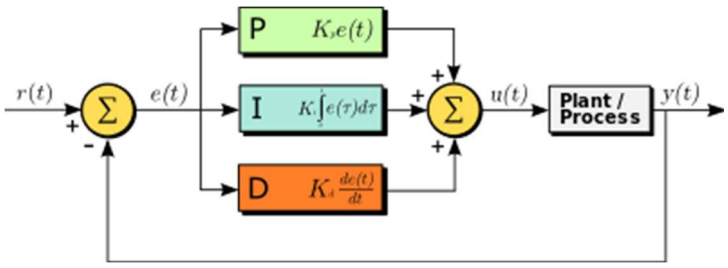

Source: Wikimedia

B) Background

In this lab, we are going to practice tuning a PID control loop with the goal of having our robot balances on its own. We have already learned how to control motors and read IMU data through I2C in previous labs, so this time, starting from skeleton codes performing those functions, you only need to focus on tuning the PID loop.

PID control continuously evaluates the error between a set point and the variable being controlled and applies a correction based on proportional, integral, and derivative terms. The output speed of the motor from the encoder is compared to the set point and fed to the controller. The controller uses the PID control algorithm to determine a new output (PWM) if needed to reduce the error and a new output from the encoder starts the loop over again.

A PID controller produces an output signal consisting of three terms: one proportional to error signal, another one proportional to integral of error signal and third one proportional to the derivative of the error signal.



- The proportional controller stabilizes the gain but produces a steady state error.
- The integral controller reduces or eliminates the steady state error, but introduces overshoot and longer settling time.
- The derivative controller reduces the rate of change of error.

Source: Wikipedia

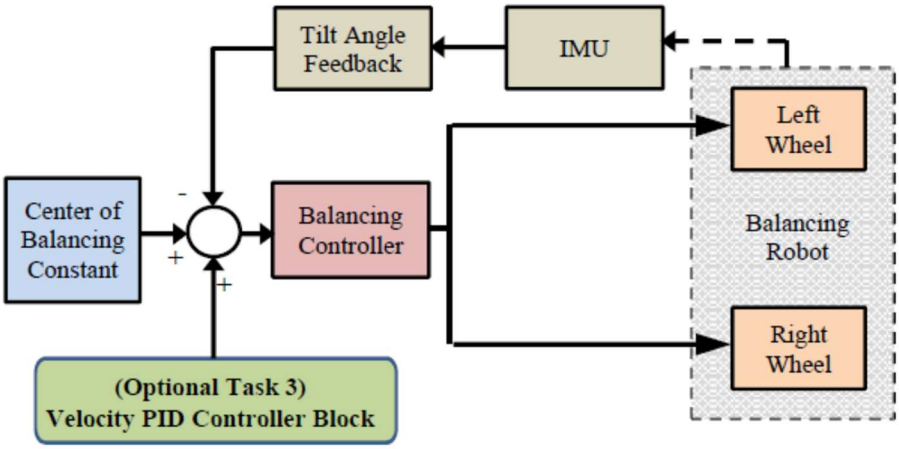
Task 1 – Complete the function call to the PID update function

The first parameter is the feedback term to the controller. The array ***ypr[]*** stores the yaw, pitch and roll quaternion angles of the robot attitude. Determine which one out of the three angles you should use as the feedback for the control loop.

The second parameter is the target or set point of the control loop. This is the target output state of your robot. Determine the ideal set point for the robot to maintain balance and pass it as the parameter to the PID update function call.

Task 2 – Tuning of the PID controller

You are given 5 parameters to tune, ***kp, ki, kd, maxIntegral*** and ***maxOutput***. Referring to the code, your course materials and the above explanation of these coefficients, adjust the parameters systematically and iteratively until the robot balances on its own.



Task 3 – Forward/ Backward: Finding New Set-Points or Cascading Control Loops

Once you achieved balancing the robot upright, to move forward and backward, choosing slightly different set-points could create short spurts and jittery forward and backward motion.

A more elegant and robust way is to create a second control loop in controlling the wheel velocity (same for both left and right wheel).