Lab 3 Report

[Task 1 – Motion with PID - Front Distance Sensor] Task one is to design a controller that will use PID proportional control to drive our e-puck robot motors to a 'desired distance' from a wall. The desired distance and current distance readings from the front distance sensor lets us calculate what velocity the motor should be running to achieve the desired distance. In our case we want the robot to stop 10 inches from the wall so first our control function compares the desired 10 inches to whatever the sensors say we are currently at to get a distance error, then multiply that error by a proportional gain to see what velocity the motor should be running at to get there. If the calculations result in velocities that exceed the capabilities of our e-puck motors, we saturate the values down to the minimum or maximum permitted, ±6.28.

[Task 2 – Motion with PID - Side Distance Sensors] Task two is to design a controller that will use PID proportional control on both the left and right motors at the same time to make sure that while we drive our e-puck robot down a path, the motors turn left or right to keep a 'desired distance' from both side walls. This task was a giant time sink not only because of the large amount of time spent with a seemingly faulty world build or the time wasted tweaking world files and robot components, but because the control functions constantly wanted to do different things with the left and right motors. After emailing a TA about the world issue (it wasn't just me going crazy, the sensors were misplaced) and speaking to the professor about our controllers' limitations I found that I was only limiting myself; we have obviously been learning about all these different e-puck sensors to use them concurrently and build on our designs. In the next tasks I'll use more of the tools at my disposal.

[Task 3 – Wall Following - Corridor] Task three is to design a controller that will drive our e-puck robot along the walls of a corridor keeping away from the wall using only the left and right sensors in out PID controller. I assume this would be done easiest by having one of the side sensors almost completely ignored and just keeping a desired wall distance from say, just the left sensor always. If the left sensor had to always stay 6 inches from the wall for example, we could probably just ignore the right sensor most of the time. If both left and right sensors want to stay at a specific distance and they are run through a control function at the same time they are going to tug back and forth. I decided that the best way to split the difference is to always find the average of the two readings as the 'center' of the hall. Part of the lab details said that regardless of where the robot starts it should be able to stop at that initial starting point. After discussing it with the professor I decided to focus my stopping point on the initial robot location at world reset. I will do the same for the maze in Task 4.

[Task 4 – Wall Following - Maze] The maze that's given in lab 4 is supposed to test if our robot can correctly wall follow in trickier environments. The first 'bug' I experienced was kind of funny in hindsight, my robot was NOT performing in the way I programmed it to during my first several minutes of experimenting with code I had copied over from task 3 and I could not find out why anywhere in my syntax. It turns out I had set the proportional gain [Kp] value to 0.1 during a previous test and forgot to change it back *facepalm*. Despite spending more time on this lab than the last, a lot of that time was spent in frustration with perceived limitations and technical issues. Task 1 was enjoyable and easy enough having been given the formulas for distance control for one sensor but trying to keep two different sensors from fighting over motor control while applying the same control values to the two sensors at the same time proved difficult. I imagine the stuttering and/or zigzagging during most of my experience with the lab is part of why Integral and Derivative components exist for full PID controllers to really smooth out the motions you're trying to achieve. Being able to use the IMU readings and all 8 sensors would have made each controller better and more compelling to experiment with; I'll have to make a point to use everything I've learned in these last three labs for the next ones. That said even when time feels wasted, I must appreciate that I've newly discovered dozens of ways NOT to accomplish my goal.

Calculations

```
//test cases for p component
    Kp = [0.1, 0.5, 1.0, 2.0, 2.5, 5.0]
// proportional control
    distance error = -desired distance - -front
    new speed = proportional gain * distance error
// saturation
    if new speed > max:
        new speed = max
    if new speed < - max:
        new speed = -max
// left right control motor via orientation
    left = left Distance Sensor / 0.0254        //meters to inches
    right = right Distance Sensor / 0.0254
    wall distance = (left + right) / 2
    error left = wall distance - -left
    error right = wall distance - -right
    new left speed = proportional gain * error left
    new right speed = proportional gain * error right
// 90 degree turns left or right
    right spin IMU endpoint = (START_THETA - 90) % 360
    left spin IMU endpoint = (START_THETA + 90) % 360

// stop at initial start position
    Start position = 0
    straight hallway run = 50
    one 90-degree turn = 33
    50 + 33 = 83
    83 * (4 straight aways & turns) = 332
    if position sensor = 332:
        left motor velocity = 0
        right motor velocity = 0
        exit(0)
```