Lab 1 – First Swim

You might remember that Ethernet cables ran between your control box and your E79 robot. The cables were probably cumbersome and limited the range of the robot. In E80, you will build an autonomous robot with an onboard Teensy microcontroller that controls its motors and collects data from its sensors. In this lab, you will get acquainted with the Teensy and build your first autonomous robot.

Assembling Robot

Your kit should have an undrilled box, one assembled penetrator, two PCBs and components, and PVC for the robot frame. From E79, you should already know how to assemble the robot frame. Go here to access the pdfs describing how to assemble the box and PCB.

Open Loop Control

You will need several software tools to run C programs written in the Arduino IDE on the Teensy. Go here: follow the software download instructions and run through the blink tutorials. You will end up downloading the Teensy bootloader, Arduino (note the compatible version), Teensyduino, and possibly some OS-specific software.

After downloading the necessary software, you will go through several tutorials on writing Arduino sketches and running them on your Teensy. Complete Tutorial 2: RGB LED (here), Tutorial 3: Serial Monitor & Input (here) up to "Built In Pullup Resistor," and Tutorial 4: Analog Input (here) up to "User Controlled LED Color." You might want to still at least read through the rest of Tutorials 3 and 4.

After completing the tutorials, go here to download the library files for this lab. Open the "libraries" folder and save the files inside in your Arduino/library working directory. Save the "lab1" folder in your Arduino working directory. The lab1 folder contains the starter code for this lab.

Using the starter code, you will write a C program for your Teensy that sends PWM signals to the robot's left, right, and vertical motors. The program should log the PWM values sent to each motor and navigate the robot through a specific course. The starter code already has the necessary library headers, global variables, and logging code. You will implement PWM output by completing Todo #1-3. Note: Reading the MotorDriver files might help.

You might notice that an IntervalTimer object was initialized in the starter code. This object is used to switch between logging data to a memory buffer and writing that data out to the SD card. More information about IntervalTimer can be found here.

Thrust Calibration

The C program will send a sequence of PWM signals to the left, right, and vertical motors to navigate the robot through an assigned course. To determine what PWM values are needed, you will need to figure out how the robot responds to different PWM values.

First, each motor will have a deadzone, or a minimum PWM value that is needed to get the motor to turn. Determine the deadzone for each of your motors and update the deadzone value in the Params.h library file.

You will also want to know how the robot moves in response to PWM inputs above the deadzone. It might be useful to look back at E79 Practicum 1B/1C, in which you found your motor's thrust for various PWM values and used a Labview simulation to relate thrust to the robot's step-response. (Hint: you can create a rough thrust curve, find the step-response for some of those thrusts, and then extrapolate.) Keep in mind that you are dealing with off-centered left and right motors this time in addition to the vertical motor. You might also want to experiment with turning and empirically determine the best way to rotate your robot. For example, what left/right PWM combination is needed and for how long should is it needed to get the robot to turn 90 degrees?

The Swim

On the second day, you will test your robot in the tank. How well does statically programming the robot work? Does the robot move in the way you expected it to?