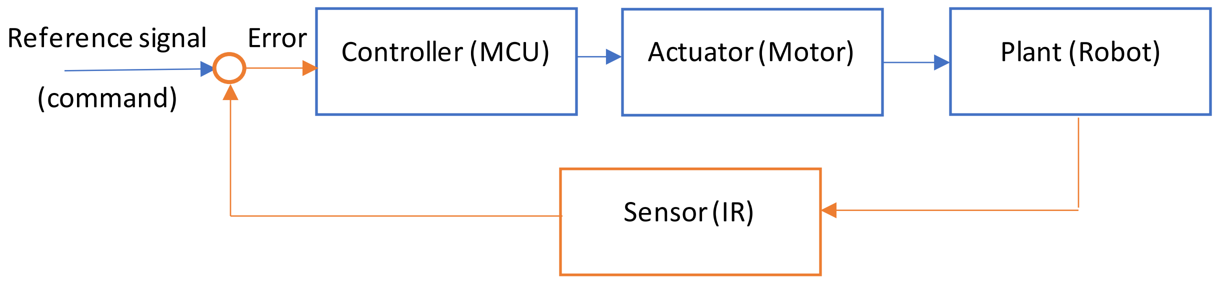
Sample codes to help starting Final Project

Part 1: Line tracking with PID control

In the previous lab, the feedback control of motors has been achieved. This time, the expected outcome is line tracking, the ability of the robot to following a line, using a series of 8 infra-red (IR) sensors.



If we pre-program the route of the robot by timing how long each movement or action takes (open-loop control), the outcome is usually poor. This degradation is due to component tolerance, declining battery level and other surrounding factors. Therefore, the robot should be equipped with sensors such that it can know "how close" between the desired and actual outcome, or "how well" itself has behaved. In our case, it measures the derivation between the current position and the mid-point of the line. This error is the difference between the reference signal and the actual outcome. With the error computed utilizing sensor information, fine adjustments and steering can be done. In control systems, this is known as feedback control or closed-loop control.

In this lab, you are going to use a provided proportional-integral-derivative (PID) controller. The mathematical expression is listed. By tracking the error dynamics $e(k)$ at present, and its accumulating and differential values, a feedback control signal $u(k)$ can be solved.

$$u(k) = K_p\big(e(k)\big) + K_i\big(e(k) + e(k-1)\big) + K_d\big(e(k) - e(k-1)\big)$$

Next, we have to conduct experiment observing the actual plant behavior and tuning the gain constants $\{K_p, K_i, K_d\}$, to improve the overall performance.

Task 1 – Obtain current position from IR sensor readings.

Finish float obtain_position_fromIR()

Task 2 – Implement motor driver

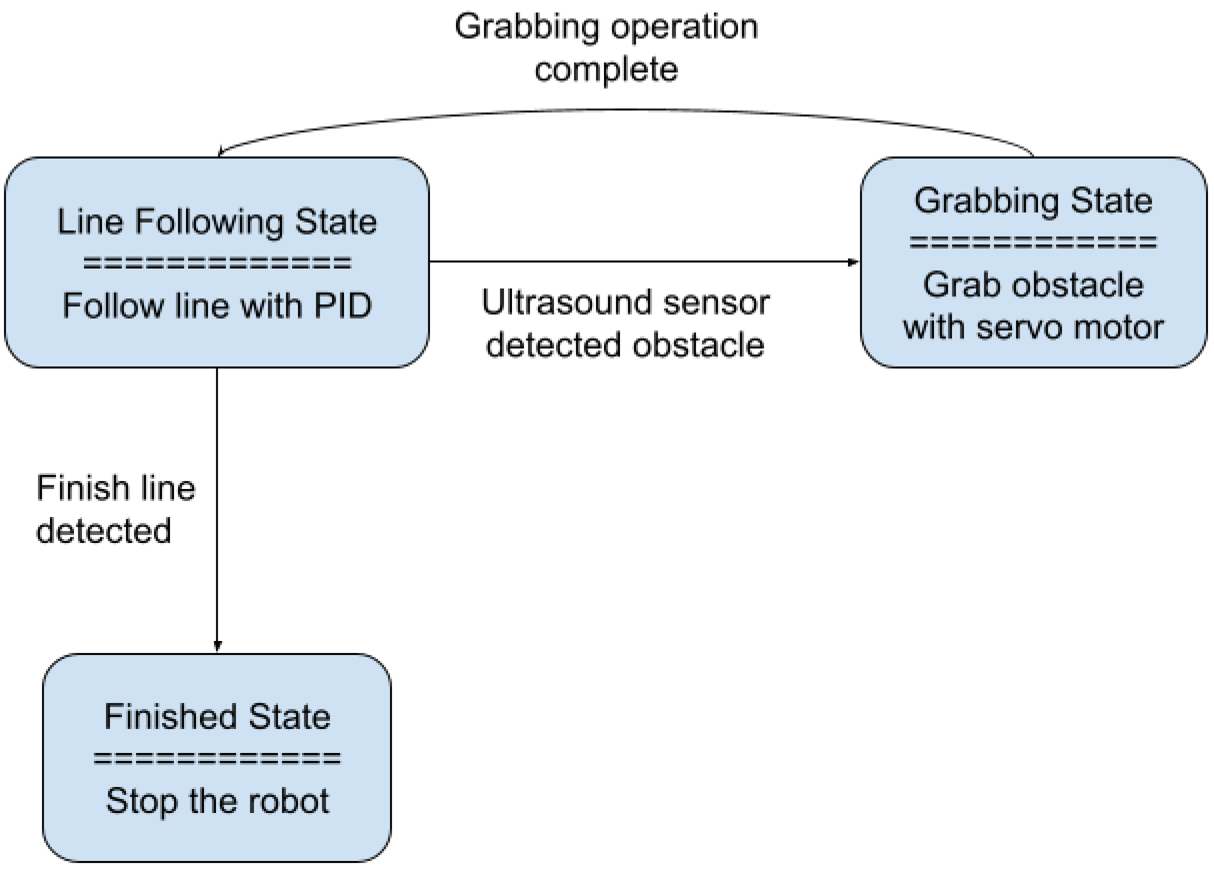
Finish void motorDrive(int power, pin_size_t in1, pin_size_t in2)

Task 3 – Finish the main control flow

Change all function inputs and parameters followed by //TODO

Part 2: Finite State Machine for obstacle avoidance

The fsm.ino skeleton code provides a framework to implement FSM for your project. This is skeleton code FSM diagram:



The skeleton code uses a very simplified state diagram for the FSM design, and you will have to add more states for the code to work well in the final project.

Following the outline of the skeleton code, complete the functions using code written in previous labs and test the simplified robot FSM to see if it reacts to external events correctly.