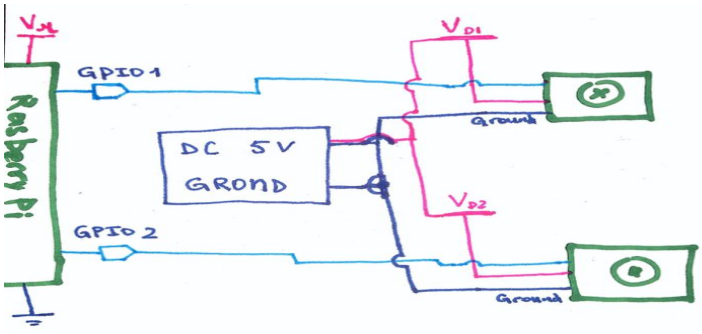
Homework 3 Part 1

Alec Newport (acn55) and Eldor Bekpulatov (eb654)

1. The continuous rotation servo is controlled through pulse width modulation. To center the servo, we must program the Raspberry Pi to deliver a 1.5 ms pulse, and continually refreshed every 20 ms. To make the servo hold completely still when receiving a 1.5 ms pulse, we can adjust the potentiometer that is placed right above the place where the cable attaches to the case. We can test our raspberry pi is producing the correct signal by hooking it up to an oscilloscope and measuring the pulse-width average. We can also use LEDs to see if our pins are updating.
2. The continuous rotation servo is controlled through pulse width modulation. Rotational speed and direction are determined by the duration of a high pulse, in the 1.3–-1.7 ms range. In order to have smooth rotation, the servo needs a 20 ms pause between pulses. To center the servo, we must program the Raspberry Pi to deliver a 1.5 ms pulse. To make it gradually rotate faster in the clockwise/counter-clockwise direction, we can decrease/increase the length of the pulse from 1.5 ms. The servo will rotate full speed clockwise at 1.3 ms pulse-width. The servo will rotate full speed counter-clockwise at 1.7 ms pulse-width.
   PWM can be controlled using two parameters: frequency and duty cycle. To fully explore the relationship between these parameters let's define a range that defines the speed and direction of the servo and call it **dt = [-0.2 … 0.2]**. Then:
   duty cycle = (1.5 + dt) / (21.5 + dt) * 100 %
   frequency = 1000 / (21.5 + dt) Hz
3. 

4. It can be very difficult to impossible to drive multiple servos at the exact same speed, whether due to calibration differences or slight variations in the drive signal or power supply. Therefore, we need some sort of feedback mechanism to keep the robot moving in a straight line. One way to do this would be to set up walls along the path, attach ultrasonic distance sensors to either side of the robot, and implement a feedback algorithm that keeps the robot an equal distance from either wall. Other considerations include potential obstacles in the path, which may be avoided using a camera or distance sensor on the front of the robot, or slippery or bumpy terrain causing the wheels to spin at different speeds, which will be compensated for by the feedback algorithm.

5. When dealing with logs or time based data, dictionaries are not good data structures. They do not keep the sequential order of data. To correct this we can store the events in a list or a queue and map the last three entries to a dictionary that stores the (x,y) positions that correspond to where to show the entries on screen.