Name: Zixiao Wang, Yue Zhang

NetId: zw579, yz2455

Lab Secssion: Monday

Submission Date: 10/16/2019

1. **Describe the required steps for calibration of a continuous rotation servo. In particular, explain the signal used for calibration and the technique used to calibrate the servo. What additional step is required to verify that the calibration signal generated by the Raspberry Pi is indeed the correct signal for calibration (hint: how would you check that the signal from the Pi is correct?)**

   First, program host device(Raspberry Pi) to deliver a 1.5ms pulse through a specified I/O pin, continually refreshed every 20ms, this technique is named Pulse Width Modulation(PWM).

   Second, connect the white pin on servo to I/O pin on host device(Raspberry Pi).

   Third, run the program to observe the motion of the servo. Twist the potentiometer gently until servo does not turn or vibrate, which means calibration is completed.

   To verify the calibration signal generated from Pi, use oscilloscope to observe the output signal from pin.

2. **Describe continuous rotation servo control; What are the parameters used for calibration (motor is still), full speed clockwise rotation and full speed counter-clockwise rotation? Express these parameters in terms that would work with RPi.GPIO calls (hint: frequency and duty cycle)**

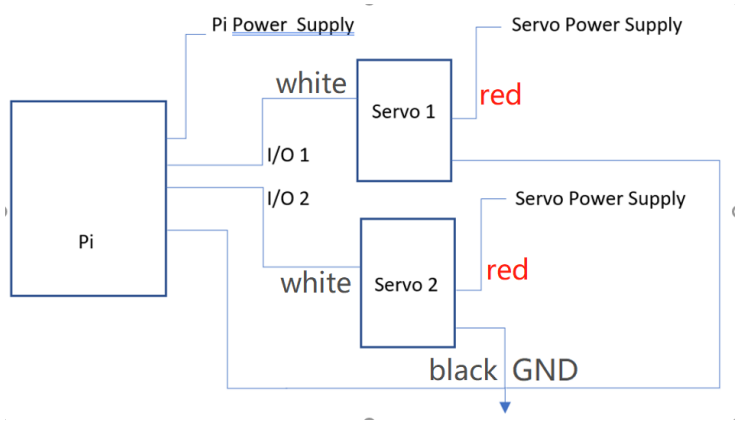
   Continuous servo rotation is controlled by pulse width modulation. Rotational direction and speed are determined means high pulse from 1.3 to 1.7ms with 20ms pause.

   High pulse in 1.5ms with 20ms pause is used for calibration, whose duty cycle is 1.5/(1.5 + 20) = 6.97%, frequency is 1/(1.5 + 20) = 46.5Hz;

   High pulse in 1.3ms with 20ms pause is used for full speed clockwise rotation, whose duty cycle is 1.3/(1.3 + 20) = 6.1%; %, frequency is 1/(1.3 + 20) = 46.9Hz;

   High pulse in 1.7ms with 20ms pause is used for full speed counterclockwise rotation, whose duty cycle is 1.7/(1.7 + 20) = 7.83%, frequency is 1/(1.7 + 20) = 46.1Hz;

3. **What is the best way to power the servo motors and the RaspberryPi. Show a sketch of the circuit indicating power and ground for two servos, the microprocessor plus control line connection for the PWM to servos.**



4. **With the servos attached to the robot frame, the robot and Raspberry Pi, become elements of a mobile embedded system. Suppose you want the robot to travel in a straight line for a specific distance. Describe one possible method for achieving straight-path travel, over some specific distance. What other considerations might you have to keep in mind (hint: think robot physical safety)?**

Denote specific distance as D, full speed of rotation as V. Program the host device Pi to generate two channels of PWM waves to drive two servo at full speed(same speed!) but opposite direction. One channel is 1.3ms high pulse with 20ms pause and another is 1.7ms high pulse with 20ms pause. Also set a counter to limit the output time as D/V, after D/V(ms) Pi should change the high pulse time duration to 1.5 to stop the rotation of both servos. To stop at a specific distance, we also need to know the diameter of the wheel!

In considering of physical safety, the time duration of high pulse should change gradually instead of jumping to another number directly. Because abrupt change in duty cycle will cause abrupt change in speed, which may become a risk for robot safety. Some elements would fall apart because of inertia.

5. **One solution for displaying the right and left logs for the rolling_control.py program from lab three would be to use a python dictionary. The dictionary for the left log might be organized as: Left_log = { stop:(80,140), clockwise:(80,160), counter_clock:(80,180) } Where the keys represent the log event and the values represent the position of the log on the screen in an (x,y) coordinate tuple. Describe the flaw when using this structure and suggest one method to correct it.**

One possible flaw of this structure could be that if history entry repeat itself, for example, "stop" could repeat 2 or 3 times out of 3 entries. In such case, we are going to have identical keys with different values in log dictionary, which will cause traceback. One possible solution could be

using queue instead of dictionary. Making use of the First-In-First-Out character, we pop the first element(like a “stop”) when number of elements reaches 3 and new entry will be pushed in. Then display elements in order on TFT screen to rolling the entries.