CSCE 236 Embedded Systems, Spring 2012
Lab 7

In Class: Thursday, April 5, 2012

Names of Group Members:

1 Instructions

This is a group assignment to work on during class. Complete all of the sections below and make sure to get the instructor or TA to sign off where required.

2 Wheel Sensors

For this part you should mount the wheel sensors as demonstrated. These are break-beam sensors that can detect the gaps in the wheels. By counting how many gaps you see you can determine how far each wheel has rotated. Connect your sensors to the `INT0` and `INT1` input pins of the Atmel (refer to the schematic to determine which pins these are). Now, connect your servos to PWM outputs so you can control them like you did in the previous project. In your main loop, write code that moves your robot straight forward whenever the button is pressed and then returns it to the starting point when the button is released. Do this by counting the number of transitions that occur on each wheel while driving forward and then moving that many backwards once the button is released.
Checkoff: *Demonstrate moving forward while the button is pressed and then returning to the starting location when the button is released.*

3 Wheel Sensors with Interrupts

Monitoring the sensors in the main loop is inefficient, so now we will move this monitoring to an interrupt handler. Review Lab 5 if you do not remember how to configure interrupts on pins `INT0` and `INT1`. Now implement the following approach for both wheels. There should be global variables: `rightDir`, `leftDir`, `rightTicks`, and `leftTicks` that maintain the current rotational direction of each wheel (forwards or backwards) and the number of ticks that have been detected. The interrupts should increment the tick counters appropriately based on the specified rotational direction (e.g. increase when moving forward and decrease when moving backwards).

Now create a function takes as an argument the number of ticks (positive or negative) you want that wheel to turn. This function should then set another global variable (e.g. `rightTargetTicks` and `leftTargetTicks`) to the appropriate target value based on the current number of ticks. It should also set the direction variables appropriately and then turn on the motor in the correct direction. The interrupt code should disable PWM whenever the current number of ticks equals the target ticks. By doing this, you will be able to specify a target distance and then it will automatically go there and stop.

When accessing these global variables outside of the interrupt code (where interrupts are already disabled) you must make sure to **disable interrupts** so that they are always in a consistent state. I suggest you create additional functions to set and read the variables with interrupts disabled and only use those functions.
Checkoff: *Demonstrate code that will drive your robot forward 12 inches and then back 12 inches* [1].

[1] The wheel diameter is 2.7 inches, so the circumference is approximately 8.48 inches.

For the above, you may need to calibrate your PWM values so that your robot drives straight. You can do this manually (by tweaking the values) or you can try to do this automatically by monitoring the relative ticks between the wheels and adjusting the PWM output value if they differ by too much.
Checkoff: *Now write code that makes your robot follow the arc of a circle forward and then backwards. How did you do this?*

Finally, write code (similar to Section 2) that moves the robot forward while the button is pressed and then returns it to the same spot when it is released. To do this, you should record the current ticks when the button is initially pressed. Then enable the motors moving forward. When the button is released, use the target tick setting function to return to the initial tick value. Just make sure the target tick value isn't set inappropriately to start with as this could cause the interrupt handler to disable the PWM prematurely.
Checkoff: *Demonstrate moving forward while the button is pressed and then returning to the starting location when the button is released.*