91.450, Robotics I
Fall 2003
Prof. F. Martin

Lab 5: Shaft Encoders and Gears

Out: Friday, 3 October 2002
Due: Friday, 10 October 2002

Shaft encoders can be used to count the number of times a wheel spins, or in general the number of digital pulses seen by an input. For this lab, we will use optical encoders that use optical switches whose beam is periodically broken by a slotted wheel (we will use a LEGO pulley wheel with holes as our slotted wheel).

Also in this lab, you will experiment with gears and gear trains.

Part I: Shaft Encoders

Read Martin 3.8 (starting on page 130) on shaft encoders.

We will be using a slightly different implementation of the driver software. Interactive C v4.2 comes with encoder routines that require digital signals (rather than analog ones as described in the book). This is OK because the break-beam sensors we have yield nice clean highs and lows.

IC's encoder functions are the following:

```
void enable_encoder(int encoder)
```
Enables the given encoder to start counting pulses and resets its counter to zero. By default encoders start in the disabled state and must be enabled before they start counting.

```
void disable_encoder(int encoder)
```
Disables the given encoder and stops it from counting. Shaft encoders use processing time every time a pulse is received while enabled, so they should be disabled when you no longer need the encoder's data. Do not re-enable an encoder that is already on.

```
void reset_encoder(int encoder)
```
Resets the counter of the given encoder to zero.

```
int read_encoder(int encoder)
```
Returns the number of pulses counted by the given encoder since the last reset.

The arguments to these functions must be in the range of 0 to 3, which correspond to encoders plugged into the Handy Board's digital inputs 7, 8, 12, and 13 (respectively).

Encoders 0 and 1 (inputs 7 and 8) are implemented using the input timer capture feature on the 68HC11 chip. Therefore processing time is only used when a pulse is actually being recorded, and even very fast pulses can be counted. These routines can notice a pulse as short as 1/2 of a microsecond. This can actually be liability—if you use a switch as an encoder, hoping to count switch clicks, you may get a dozen or more encoder counts per switch press (due to switch bounce).

Encoders 2 and 3 (inputs 12 and 13) are implemented by periodically sampling the input bits. These are not susceptible to switch bounce, but require a pulse of at least 1 millisecond to register a count (2000x slower than the hardware-based encoders).

The encoding software keeps a running count of the number of pulses each enabled encoder has seen. The number of counts is set to 0 when a channel is first enabled and when a user resets that channel. Because the counters are only 16-bits wide, they will overflow and the value will appear negative after 32,767 counts have been accumulated without a reset.

Activities:

1. Install encoders on your robot's wheels. Enable the encoders and play with reading and resetting them as the robot moves.

2. These encoder routines do not provide inherent velocity measures. Write code to generate velocity information from repetitive counts readings.

3. Now write a program that attempts to maintain constant velocity on the drive wheel by varying the power level delivery to the motor (Martin p. 137, Exercise 4). Experiment with the system by holding the robot in the air and applying pressure to the drive wheel. Is the system able to maintain the velocity? Why happens if you suddenly remove the pressure?

4. A good way to make a robot stall detector is to have a non-powered wheel that is dragged along as the robot moves, and checking for a zero velocity on this when the robot should be moving. I am just planting this idea in your head for later.

Part II: Gears and Gear Trains

Read Martin, Sections 4.2 and 4.5. Do Exercise 4.2.2 on p. 146.

Do Exercise 3 in Section 4.5.4 on p. 173 of Martin. Show me your two designs. (You should have enough Lego to build two different chasses.)

I'd also recommend that you try Exercise 2 on p. 173, but there's nothing to turn in for this.