

**Physics 219 – Fall, 2005**  
**LabNotes 7 – Feedback and Control**

**Contents**

<b>Wednesday, November 9 .....</b>	<b>2</b>
What's the big idea? .....	2
Historical examples of feedback and control .....	2
A Showcase for Feedback: Controlling the position of a motor.....	3
Sensing position using "shaft encoding" .....	3
Quadrature shaft encoder.....	5
Go exactly where I tell you.....	6
<b>Friday, November 11 .....</b>	<b>6</b>
H-bridge driver in a chip .....	6
Two Types of Control.....	8
Bang–Bang Control of a Motor .....	8
Proportional Control of a Motor .....	9
Power control using pulse width modulation.....	9
It feels like spring.....	10

## Wednesday, November 9

### What's the big idea?

Every field has its “big ideas”: key concepts necessary for understanding the field that often shed light on other disciplines as well. In this course we encounter – and pay close attention to – a number of big ideas from the world of electronics. Often, these big ideas are important for understanding not only electronic systems but also for understanding social systems and the natural world. A great example of this is the idea of **feedback and control**. We just saw an example of feedback in our discussion of op amps. In this lab we'll see how circuits that use a LogoChip to further explore ideas related to feedback.

You have already seen many examples that used “sensors” and “actuators”, mediated by the LogoChip. For example the code :

```
waituntil[touch?] flash
```

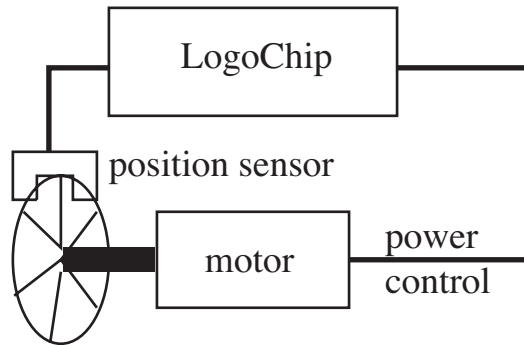
constantly monitors a (touch) sensor and uses this information to **control** an actuator (flashing LED). We use the term **feedback** to describe situation where we somehow measure the current state of the system and then make use of this information in deciding what sort of signal we end up giving to the thing we are trying to control. Often we use feedback to enable us to “home in” on a desired goal (and stay there!).

### Historical examples of feedback and control

- **Watt's governor** - is a mechanical device used to keep the rate of rotation of a shaft constant in the face of a motor that is delivering a fluctuating amount of power. The Watt's governor increases the friction in the friction in the system if the rotation rate speeds up above the desired rate and lowers the friction if the rotation rate drops below the desired rate.
- **toilet float** - senses the level of water in the tank and causes a valve to open when the tank is empty and close when the tank is full.
- **thermostat** - measures the temperature in your house. It then turns on the heat when it gets too cold and urns off the heat when it gets too hot.

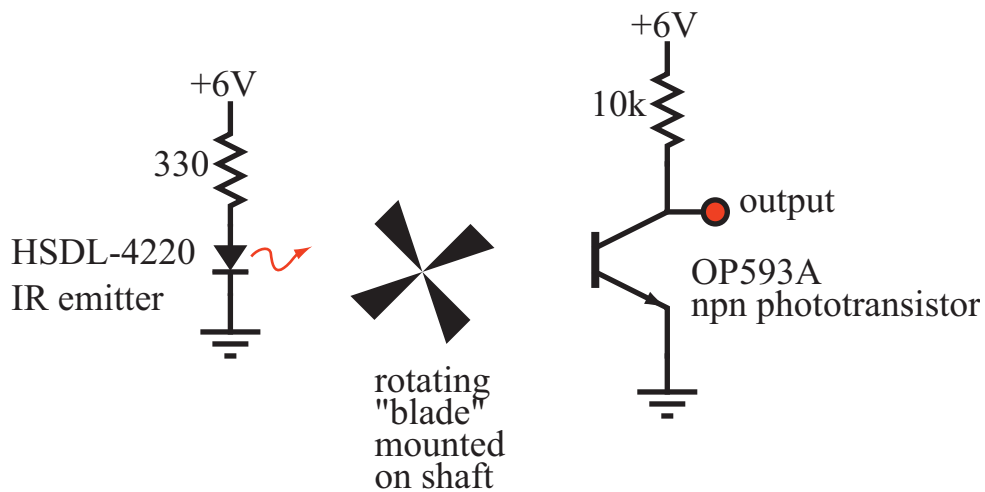
**A Showcase for Feedback: Controlling the position of a motor**

The challenge that you'll address in this lab is to construct a system that uses feedback and control to cause a wheel turn to *and maintain* a desired position. The basic scheme is sketched in the figure below.

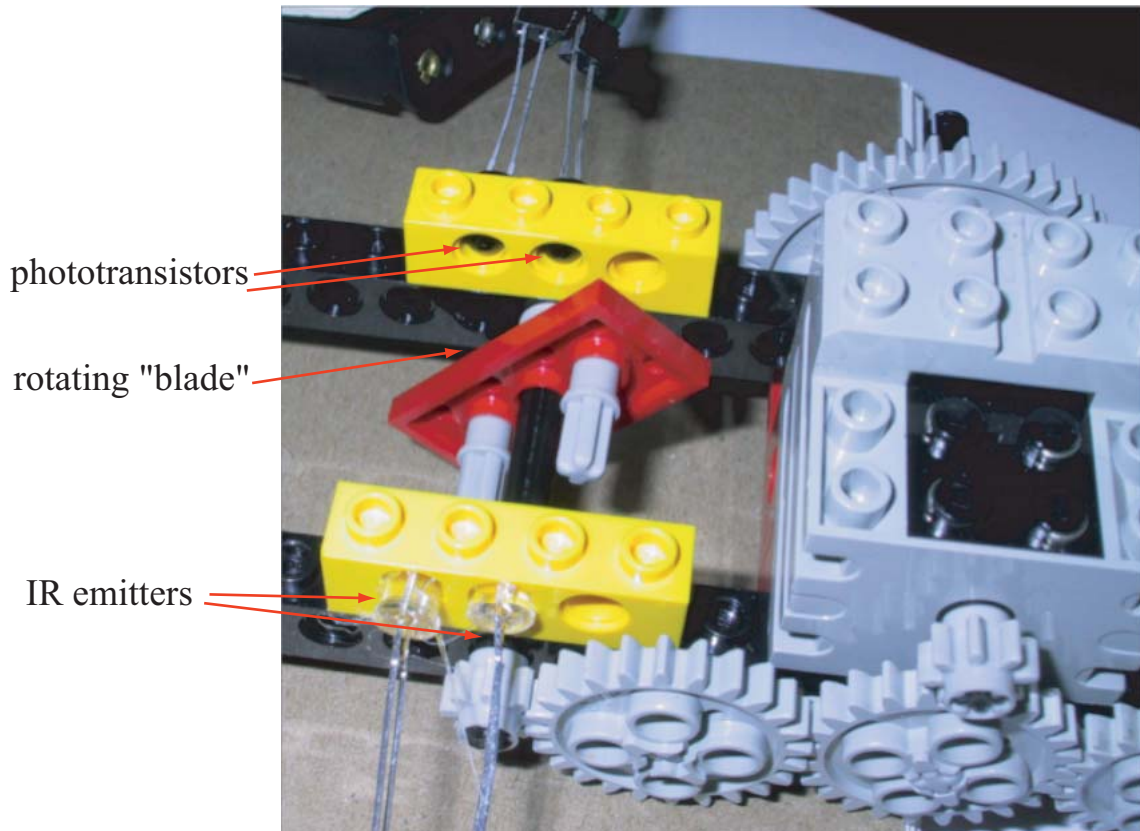


**Sensing position using “shaft encoding”**

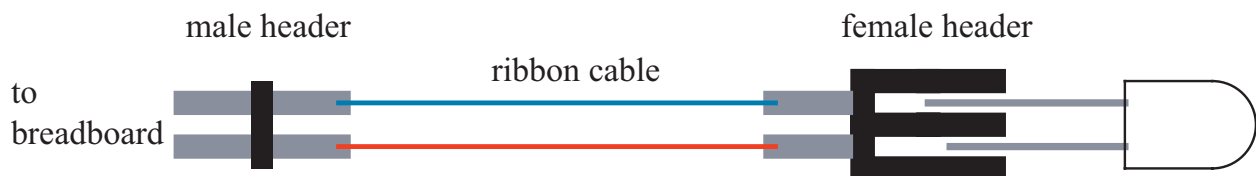
- Start by using a “break beam sensor” and some LEGO parts to build a **shaft encoder** to sense the angular position of a rotating wheel. The circuit for the break beam sensor is shown in the figure below.



- You will need to wire up two of these break beam sensors, as shown in the photo below:



- You will have to do some soldering to make cables to connect the IR emitter and the phototransistor to your breadboards. Since each of the two break beam sensors has a transmitter and a detector, you'll need to make a total of four cables. (**Hint:** Make sure to use heat shrinkable tubing to insulate and provide mechanical support for your solder connections.)



- Now construct the mechanical arrangement shown in the photo below. The arrangement includes a pair of break beam sensors, arranged so that as the shaft rotates the beams are successively broken by a LEGO plate mounted on the axle. The geometry is such that there is a small range of orientations where the plate is blocking both beams at the same time. (See photo on previous page.)
- Test your shaft encoders by viewing the phototransistor outputs on an oscilloscope as you rotate the shaft. You should observe the output switch between 0 V and 6 V as you rotate the shaft, making nice square waves .

- Connect one of the phototransistor outputs to pin B0 of a LogoChip and use the following program to print out a “count” of the number of shaft rotations. The program works by counting “edges” as the beam is blocked by the rotating blade. For a blade made of a rectangular LEGO plate, you should get two “counts” per revolution.

```

to count
  setn 0
  loop [wait-for-edge
        setn n + 1
        print n]
  end

to wait-for-edge
  waituntil [not blocked?]
  waituntil [blocked?]
  end

to blocked?
  output (testbit 0 portb)
  end

```

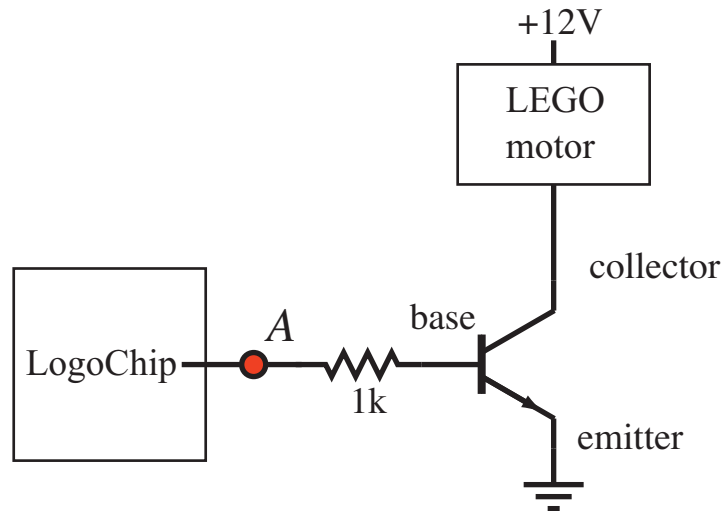
### Quadrature shaft encoder

- The above program cannot detect the “sense” of the axle’s rotation. It doesn’t matter whether the shaft is rotating clockwise or counter-clockwise, the count always increases. But with a little cleverness, you can use a second break beam sensor to add the ability to keep track of the direction of the rotation. Use the two channels of your oscilloscope to look simultaneously the outputs of the two break beam sensors as you rotate the shaft. What changes do you observe in the traces as you change the sense of the shaft’s rotation?
- Modify the program so that it prints “net” count, counting up as the shaft rotates one way and counting down when it rotates the other way.

This scheme of using a pair of sensors, positioned “about 90 degrees out of phase from one another” to measure both the number and the sense of a shaft’s rotations, is known as **quadrature shaft encoding**.

**Go exactly where I tell you**

- connect a LEGO motor, driven by a transistor switch, to your LogoChip as shown.

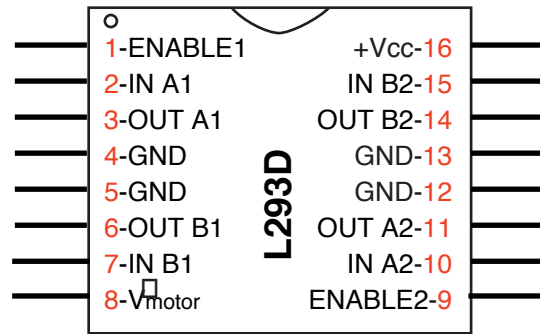


- Build a “gear train”, as shown in the photo on the previous page, that connects the motor to your shaft encoder and also to a wheel whose angular position to will try to control.
- Write a Logo program that can make the motor shaft turn a specified number of revolutions.

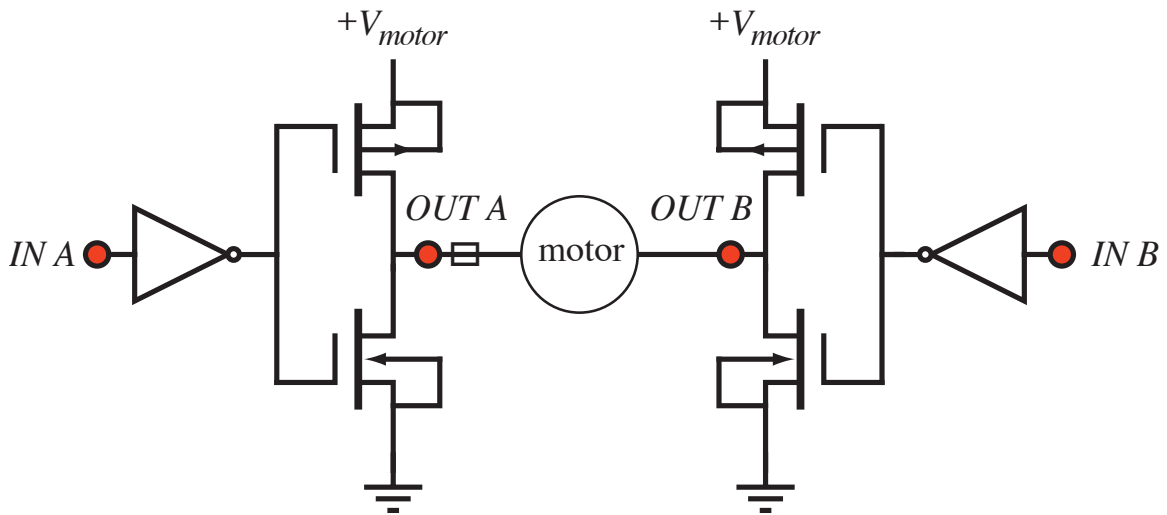
**Friday, November 11****H-bridge driver in a chip**

With your quadrature shaft encoder you now have a good way to sense the angular position of a wheel. In order to fully control the wheel’s position the motor needs to be able to turn the wheel in both directions.

You know how to do this; with an H–bridge circuit. You’ve already built your own H–bridge using 4 MOSFETs. In this lab, for convenience, you should use a LD293 motor driver chip instead. These integrated circuits contain two built-in H–bridges, and also built-in diodes to dissipate the “back emfs” that develop when switching inductive loads such as motors. (See page 52 of Horowitz and Hill.)



**L293D pinout diagram**



The operation of the L293D can be understood from the following **truth table**:

**L293D Truth Table:**

ENABLE	IN A	IN B	OUT A	OUT B	MOTOR STATE
L	X	X	X	X	coast
H	L	L	L	L	brake
H	L	H	L	H	on cw
H	H	L	H	L	on ccw
H	H	H	H	H	brake

- Wire up a L293D motor driver and interface it to the LogoChip, connecting two LogoChip output pins to the inputs of the motor driver chip. Note that, unlike the CMOS-based H-bridge that you built, the L293D motor driver chip allows you to use supply your motor with a voltage that is different from the LogoChip's power supply voltage. Thus, although not necessary here, you could run the LEGO motor at 9 V (the specified value) or, if you really want to go fast, 12 V (this voids the warranty!)
- Write a few simple Logo procedures to turn the motor on this way and that way, to turn it off and to make it brake.

### Two Types of Control

A thermostat is a device that uses feedback and control to try to maintain a constant temperature in your house. Most thermostats employ a type of control known as **bang-bang control**. Bang-bang control is a kind of all or nothing deal. When the temperature drops too low, the furnace comes on, full blast, to heat up the place. When it gets too hot, the air conditioner's unleashed, flat out, to chill things out. Bang-bang control can work pretty well in some instances, but its abruptness sometimes causes problems. Think about what would happen if you tried to use bang-bang control to drive your car down the middle of the road: In such a scheme, the instant your eyes detected that you had drifted a bit too far to the right, your brain would have you turn the steering wheel sharply to the left. As soon as you crossed a bit too far to the left you'd slam the wheel to the right. Swerving down the road like this, you'd be pulled over in no time.

An often better idea is to use **proportional control**, where the action taken, the feedback, is *in proportion* to the degree the system diverges from the ideal point. Proportional control is how you usually drive; every time you notice the car drifting a tiny bit to the left, you steer *slightly* to the right and vice versa.

### Bang-Bang Control of a Motor

In a bang-bang control scheme we expect the motor to turn completely on or completely off in an attempt to maintain the output wheel at the desired position.

- Write a program that uses bang-bang control to maintain the output wheel at a constant angular position. As you try to twist the output wheel with your hand, the motor should do everything in its power to thwart your efforts.

### Proportional Control of a Motor

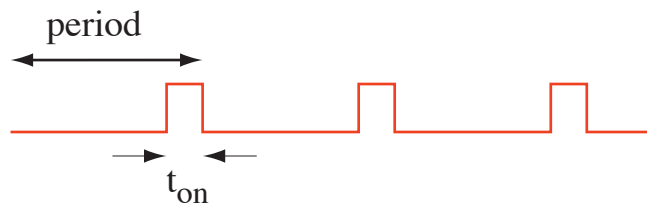
With proportional control the goal is to have the motor provide a “spring-like” restoring force that increases in strength as the “deviation” from the desired position increases. This strategy is summarized in the following expression:

$$\text{MOTOR POWER} = \text{GAIN} \times (\text{CURRENT POSITION} - \text{DESIRED POSITION})$$

Note that the sign of the right hand side indicates the direction in which the motor should rotate.

### Power control using pulse width modulation

In order to implement proportional control you need to be able to control the amount of power delivered to the motor. The average power delivered to the motor can be varied using the technique of **pulse width modulation**. Conveniently, pin C2 on the LogoChip can be configured to generate a square wave with a user determined period and duty cycle (fraction of time on) using the following code:



(Note: This feature is *very* nice; it allows you to do PWM without completely tying up your program.)

- Try using the code to generate a PWM signal on pin C2. Use an oscilloscope to monitor the signal as you change the duty cycle. What is the period of the signal?

```
; Pulse Width Modulation code. This program
; causes a square wave of user selected period and
; duty cycle to appear on pin C2. :val determines
; the duty cycle Values from 0 to 100 can be used.
```

```
constants [[ccp1con $fbd] [t2con $fca] [ccpr1l $fbe]
[pr2 $fcb]]
```

```
to initialize
clearbit 2 portc-ddr
write t2con 6
```

```
write pr2 100
write ccplcon 12
end

to pwm :val
write ccpr11 :val
end
```

- Can you make your motor slow down and speed up by using pulse width modulation on the LD293 enable line?

### **It feels like spring**

- Now it's time to put everything together. Write a program that uses proportional control to try to maintain the output wheel at a constant position. If things are working properly you should feel a restoring torque that grows in strength as you twist the output wheel away from the set position. Try varying the "spring constant" by adjusting the value of the gain used in your program.

(**Hint:** The "feedback and control loop" will tend to work better the shorter the amount of time it takes to go through the loop. If you take too long to get through the loop the response gets too "sluggish". Imagine what driving would be like if it took 10 seconds between when you see the road curve and when you turn the wheel.)

We're going to spend the next lab session talking about programming in Logo, so if you're having trouble with writing this program, help is on the way.