



WPI

IMGD 3100 – Novel Interfaces for Interactive Environments: Physical Feedback

Robert W. Lindeman

Associate Professor

Interactive Media & Game Development

Human Interaction in Virtual Environments (HIVE) Lab

Department of Computer Science

Worcester Polytechnic Institute

gogo@wpi.edu

Motivation

- We've looked at how to get (some) physical input from the user
- Now we look at providing physical feedback
- Sound and vision are important
 - Often, though, they are all that are used by designers
- There is something special about receiving physical feedback
 - Different part of the brain
 - Different properties can be conveyed
 - E.g., the sound of wind vs. wind

Design Space of Physical Feedback

- We need to think about designing *machines*
 - Kind of cool!
- We need to glue mechanical and electrical engineering together with programming
 - Understanding things is tougher
 - Need EE background, and possibly some ME
 - Debugging is tougher
 - Need to analyze current, etc.
- Does this sound familiar?
 - This is what RBE is all about!
- Reward:
 - Design and build stuff that acts in the real world!

Design Tips

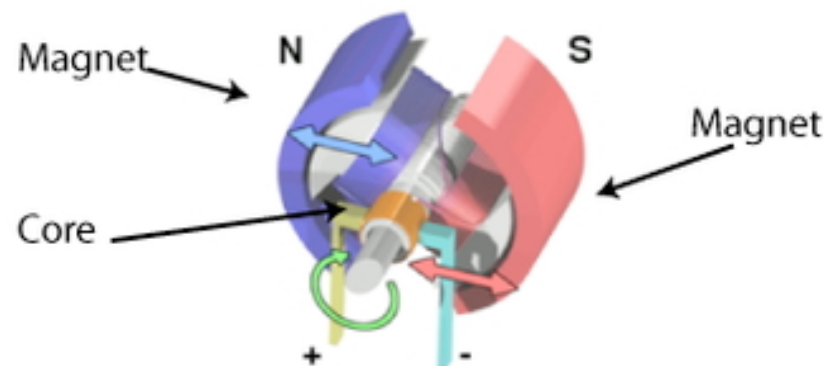
- ❑ Map analog (continuous) values to analog displays
- ❑ Map binary (discrete) values to binary displays
- ❑ Pay attention to user attention
- ❑ Measure and refine to improve user performance/experience
- ❑ Keep physical, visual, and audio feedback synchronized
- ❑ Be aware of the use environment
 - Car blinker

Primary Tool: Motors

- Many interesting feedback systems can be created using motors
 - DC motors
 - Servos motors
 - Stepper motors

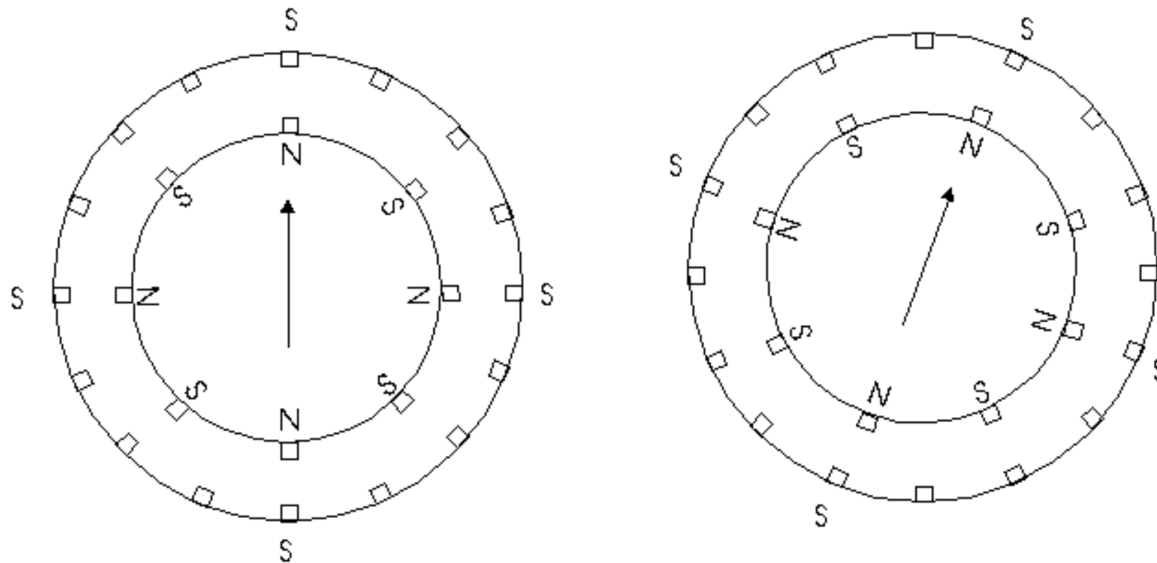
DC Motors

- Motor spins using magnetism
 - Electromagnetic coil + fixed magnets
- Switch the polarity every half-turn
- Can reverse direction using an H-Bridge



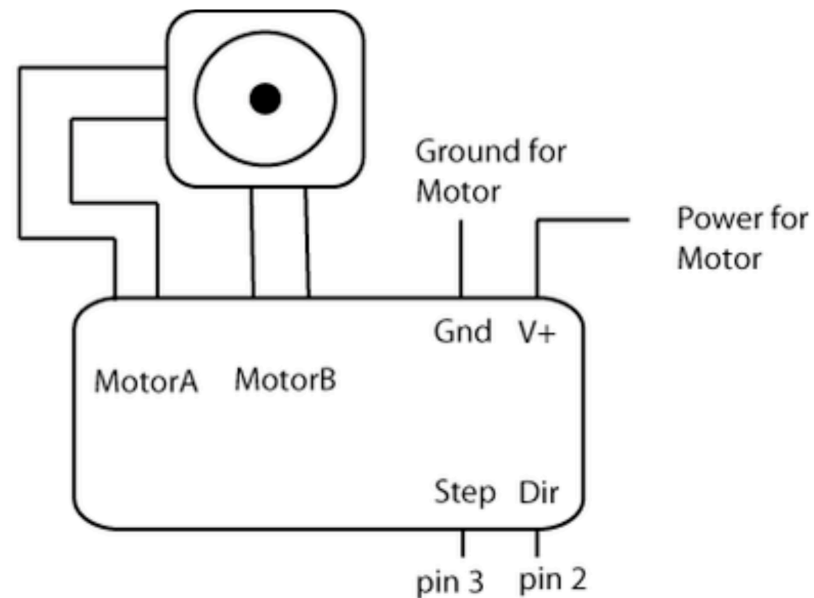
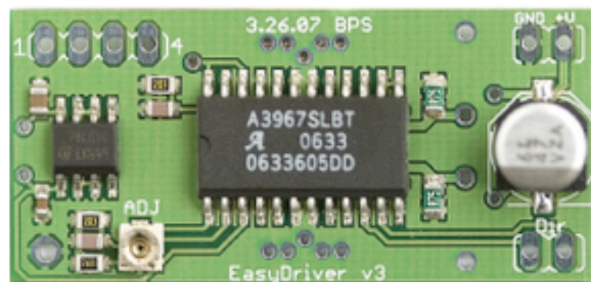
Stepper Motors

- ❑ Motor (again) spins using magnetism
- ❑ Multiple electromagnets in a circle allow the motor to "step" to a desired position



Stepper Motors (cont.)

- ❑ Stepper driver board makes things easier
- ❑ Connect to Arduino pins

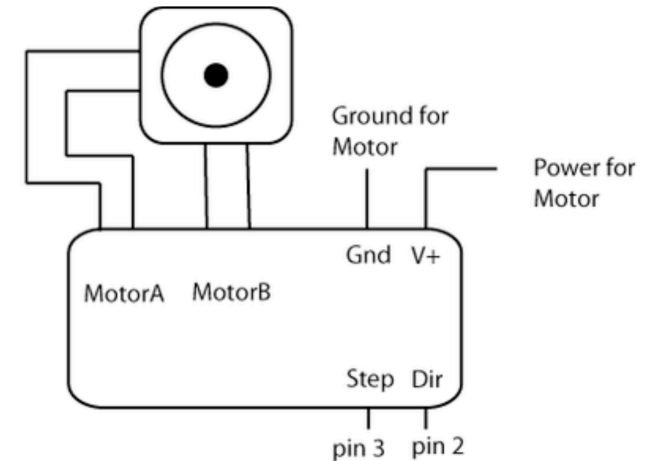


Stepper-Motor Code

```

int dirPin = 2;
int stepperPin = 3;
void setup( ) {
    pinMode( dirPin, OUTPUT );
    pinMode( stepperPin, OUTPUT );
}
void step( boolean dir, int steps ) {
    digitalWrite( dirPin, dir );
    delay( 50 );
    for( int i = 0; i < steps; i++ ) {
        digitalWrite( stepperPin, HIGH );
        delayMicroseconds( 100 );
        digitalWrite( stepperPin, LOW );
        delayMicroseconds( 100 );
    }
}

```



```

void loop( ) {
    step( true, 1600 );
    delay( 500 );
    step( false, 1600*5 );
    delay( 500 );
}

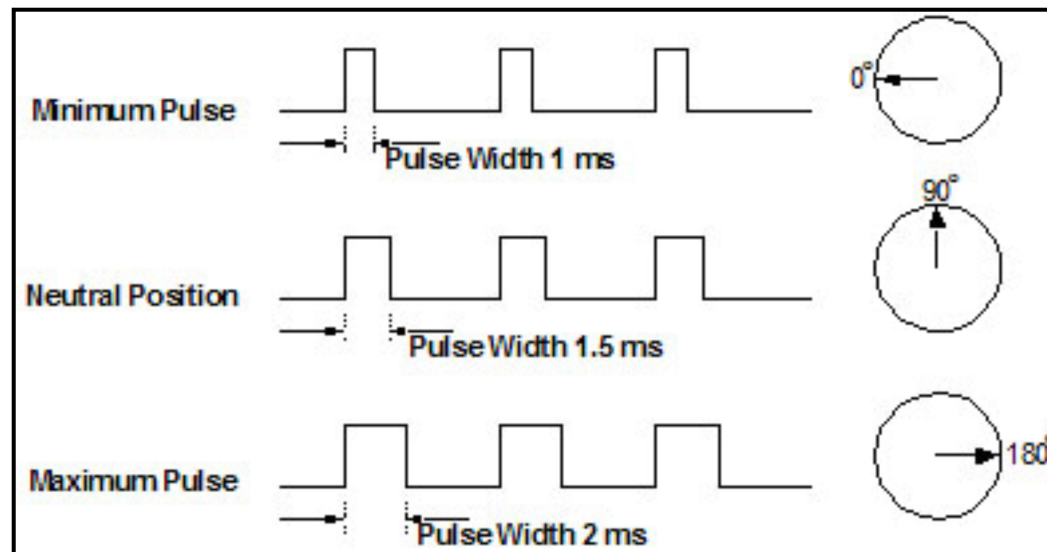
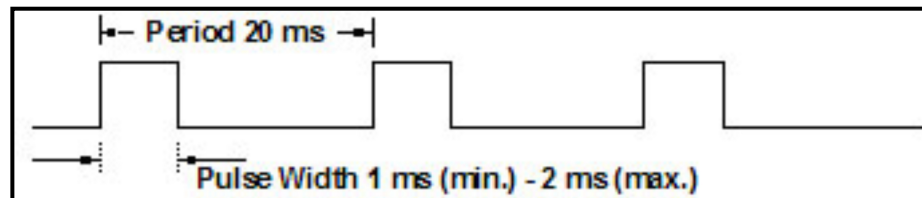
```

Servo Motors

- A servo is a motor with some "extra" features
 - It reads the voltage passed to it, and decides how far to rotate within a given range (e.g., 180°)
- Cool fact:
 - The same code used to control small servos can be used to control honkin' servos
 - Think big!
- Not-so-cool fact:
 - You can't control servos using the "normal" PWM outputs on the Arduino
 - You have to "roll-your-own" PWM

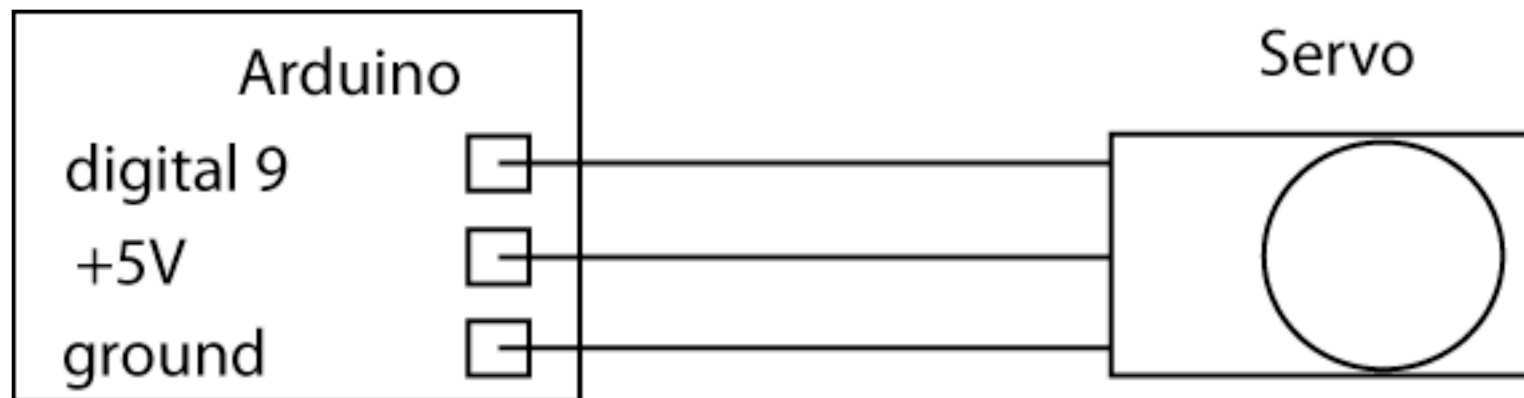
Servo Motors (cont.)

- Actually, it's not that bad



Servo Motors (cont.)

- Three wires
 - Red (usually) is power
 - Black (usually) is ground
 - Yellow (or white) goes to a digital pin



Servo-Motor Code

```
#include <Servo.h>

Servo myservo; // create the servo object
int potpin = 0; // analog pin used to connect the potentiometer
int val; // variable to read the value from the analog pin

void setup( ) {
  myservo.attach( 9 ); // attaches the servo on pin 9 to the servo object
}

void loop( ) {
  // Read the value of the potentiometer
  // (value between 0 and 1023)
  val = analogRead( potpin );

  // Scale it to use it with the servo
  // (value between 0 and 180)
  val = map( val, 0, 1023, 0, 180 );

  // Sets the servo position according to the scaled value
  myservo.write( val );

  // Wait for the servo to get there
  delay( 15 );
}
```

Steppers and Servos

- Servos are similar to Steppers
- Servos are smoother than Steppers
 - Better for continuous motion
- Steppers are better for "locking" in place or moving to a predefined position
- Can get multipurpose Arduino shields (AdaFruit)
 - 2 Servos
 - 4 DC motors
 - 2 Steppers
 - Screw-down terminals

