Analog vs. Digital

- Digital is either **on** or **off**
  - HIGH or LOW, logic 1 or logic 0, +5v or 0v
  - No shades of grey...

- Analog is a continuous signal
- Can be used to sense a continuous range of values
  - Like a volume knob on a stereo
  - Or a heat setting on an oven
  - Or a steering wheel in a car
Analog vs. Digital

Analog

Digital

ADC

10 bit resolution

Analog Input

Sure sure, but how to make a varying voltage? With a potentiometer. Or just pot.

www.todbot.com
Wire up a Potentiometer

Analog Inputs and Arduino

```cpp
int sensorPin = A2;    // Analog pin 2
int ledPin = 13;
int sensorValue = 0;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    sensorValue = analogRead(sensorPin); // read ADC
    val = map(val, 0, 1023, 100, 255);    // Interpolate
    analogWrite(ledPin, val);   // write value to the LED
}
```
Analog Inputs and Arduino

```c
int sensorPin = A2;    // Analog pin 2
int ledPin = 13;
int sensorValue = 0;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    sensorValue = analogRead(sensorPin); // read ADC
    val = map(val, 0, 1023, 100, 255); // Interpolate
    analogWrite(ledPin, val);   // write value to the LED
}
```

Try this out with “potFade” in the DM examples

Analog Inputs and Arduino

```c
int sensorPin = A2;    // Analog pin 2
int ledPin = 13;
int sensorValue = 0;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    sensorValue = analogRead(sensorPin); // read ADC
    val = map(val, 0, 1023, 100, 255); // Interpolate
    analogWrite(ledPin, val);   // write value to the LED
}
```
Moving on... Servos

- Servo motors are small DC motors that have a range of motion of 0-180°
  - Internal feedback and gearing to make it work
  - Easy three-wire interface
  - Position is controlled by PWM signals
    - Same idea as LED fading...
  - It's all hidden in a library function for you!

Servo Control

- PWM freq is 50 Hz (i.e. every 20 milliseconds)
- Pulse width ranges from 1 to 2 milliseconds
  - 1 millisecond = full anti-clockwise position
  - 2 milliseconds = full clockwise position
# Servo Class Functions

- `#include <Servo.h>`  // include Servo library
- `Servo myservo;`  // creates an instance of Servo class
- `myservo.attach(pin);`  // attach to any digital output pin
- `myservo.write(pos);`  // moves servo from 0-179

- Servo library can control up to 12 servos on our boards
- Aside effect is that it disables the PWM on pins 9 and 10

---

## Servo movement

```cpp
#include <Servo.h>

Servo myservo;  // create servo object
int potpin = A2;  // analog pin for potentiometer
int val;          // variable to hold value from the ADC

void setup() {
  myservo.attach(10);  // attaches the servo object to pin 10
}

void loop() {
  val = analogRead(potpin);  // reads potentiometer (0 1023)
  val = map(val, 0, 1023, 0, 179);  // Interpolate val to 0-179
  myservo.write(val);  // sets the servo position to the scaled value
  delay(15);  // wait for the servo to get there
}
```

Load Sketchbook - DM - SimpleServo
Servo + Potentiometer

Wire this up! (Vdd is +5)  
Run with potFade from the DM examples
Servo + Potentiometer

End of Activity Two

- The pot and the servo are the basic building blocks for our drawing machine
- There are some additional slides that you can look at later
- There’s a summary at the end of the handout
Interpolation

- `value = map(val, 0, 1023, 0, 179);`
  - Interpolates “val” from 0-1023 to 0-179

- `value = constrain(val, 0, 179);`
  - Constrains value to whatever val is, but constrained to 0, 179 (i.e. anything over 179 goes to 179)

- In practice, the range of your analog sensor isn’t likely to be 0 – 1023.
  - Use calibration to check!

---

Communicating with Others

- Arduino can use same USB cable for programming and to talk with computers
- Talking to other devices uses the “Serial” commands
  - `Serial.begin()` – prepare to use serial
  - `Serial.print()` – send data to computer
  - `Serial.read()` – read data from computer
Serial from Arduino to PC

- **Serial.begin(baud-rate);**
  - baud-rate is 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 57600, or 115200
  - Sets serial bit rate - Use 9600 to start...

- **Serial.print(arg);**
  - Sends arg to the serial output – can be number or string

- **Serial.println(arg);**
  - Same, but also prints a newline to the output

Send data to PC

```java
void setup() {
  Serial.begin(9600);  // init the serial port
}

void loop() {
  Serial.println("Hello World!");  // print to the screen!
  delay(500);  // Wait so you don’t print too fast
}
```

Opens the "serial monitor" on the host
Checking on Analog Inputs (Calibration)

```cpp
int sensorPin = A0;      // select the input pin for the potentiometer
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  Serial.begin(9600);              // Init serial communication at 9600 baud
}

void loop() {
  sensorValue = analogRead(sensorPin); // read the value from the sensor:
  Serial.print("Sensor value is: ");            // print a message
  Serial.println(sensorValue);                    // print the value you got
  delay(50);                                                // wait so you don’t print too much!
} // VERY useful for getting a feel for the range of values coming in
  // Remember to open the Serial Monitor to see the values
```

Sensing the Dark

- Pots are example of a voltage divider
- Voltage divider splits a voltage in two
- Same as two resistors, but you can vary them

![Image of voltage divider](www.todbot.com)
Sensing the Dark: Photocells

- aka. photoresistor, light-dependent resistor
- A variable resistor
- Brighter light == lower resistance
- Photocells you have range approx. 0-10k

Photocell Circuit
CDS light sensor

Resistive sensors

thermistor (temperature)

flex sensor (bend, deflection)

force sensors (pressure)

photocell (light)

also air pressure and others

circuit is the same for all these
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;          // select the pin for the LED
int sensorValue;        // variable to store the value coming from the sensor

void setup() {
  pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT:
  // Note that you don’t need to declare the Analog pin – it’s always input
}

void loop() {
  sensorValue = analogRead(sensorPin); // read the value from the sensor:

  digitalWrite(ledPin, HIGH); // turn the ledPin on
  delay(sensorValue); // stop the program for <sensorValue> milliseconds:
  digitalWrite(ledPin, LOW); // turn the ledPin off:
  delay(sensorValue); // stop the program for for <sensorValue> milliseconds:
}
Servo/Light Practice

- Use a photocell on the input
  - put in series with 10k ohm resistor
- Use a servo on the output
  - create a servo object
- make the servo do something in response to the amount of light falling on the photocell

With Calibration

```
#include <Servo.h>
Servo myservo; // create servo object to control a servo
int sensorPin = A0; // analog pin used to connect the potentiometer
int sensorVal; // variable to read the value from the analog pin
int scaledVal; // variable to hold the mapped and constrained value

void setup() {
  myservo.attach(9); // attaches the servo object control wire to pin 9
  Serial.begin(9600); // init the serial port at 9600 baud
}

void loop() {
  sensorVal = analogRead(sensorPin); // read the value of the sensor
  scaledVal = map(sensorVal, 0, 1023, 0, 179); // scale it to use it with the servo
  scaledVal = constrain(scaledVal, 0, 179); // make sure it stays in range
  Serial.print("sensor = "); // This print section is used for calibration
  Serial.print(sensorVal); // Write down the values you see from the sensor
  Serial.print(" output = "); // and replace the “0, 1023” above with the
  Serial.print(scaledVal); // range of values you actually see
  myservo.write(scaledVal); // sets the servo position according to the scaled value
  delay(20); // wait for the servo to get there
}
```

Load Sketchbook - ServoCalibration
Sensor/Servo Coordination

Getting Input (Digital)

- Switches make or break a connection
- But Arduino wants to see a voltage
  - Specifically, a “HIGH” (5 volts)
  - or a “LOW” (0 volts)

How do you go from make/break to high/low?
Switches

- Digital inputs can “float” between 0 and 5 volts
- Resistor “pulls down” input to ground (0 volts)
- Pressing switch sets input to 5 volts
- Press is HIGH
  Release is LOW

Why do we need the “pull down” resistor?

Another Switch

- Resistor pulls up input to 5 volts
- Switch sets input to 0 volts
- But now the sense is inverted
  - Press is LOW
  - Release is HIGH

“pull-up”
A Switch

Pressing the button, “closes the gap”

Using a Switch
Using digitalRead()

- Assume `int myPin = 5;` // pick a pin
- in setup() – use `pinMode(myPin, INPUT);`
- in loop() – use `digitalRead(myPin)`
  - `int foo;` // variable to hold input
  - `foo = digitalRead(myPin);` // Read the value from pin 5
  - `if (foo == 1)` // check the value
  - `{do something}` // only “do something” when
  - // the button is high

Load Sketchbook - DM - SimpleButton

digitalRead(pin);

// constants won’t change. They’re used here to set pin numbers:
const int buttonPin = 2; // the number of the pushbutton pin
const int ledPin = 13; // the number of the LED pin

// variables hold values that will change:
int buttonState = 0; // variable for reading the pushbutton status

void setup() {
  pinMode(ledPin, OUTPUT); // initialize the LED pin as an output:
  pinMode(buttonPin, INPUT); // initialize the pushbutton pin as an input:
}

void loop() {
  buttonState = digitalRead(buttonPin); // read the state of the pushbutton value:
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH); // turn LED on:
  }
  else { digitalWrite(ledPin, LOW); } // turn LED off:
}
Moving on...

```cpp
int ledPin = 13; // choose the pin for the LED
int inPin = 7; // choose the input pin (for a pushbutton)
int val = 0; // variable for reading the pin status
int delayval = 100;

void setup()
{
    pinMode(ledPin, OUTPUT); // declare LED as output
    pinMode(inPin, INPUT); // declare pushbutton as input
}

void loop()
{
    val = digitalWrite(inPin); // read input value
    if (val == HIGH)
        delayval = 1000;
    else
        delayval = 100;
    digitalWrite(ledPin, HIGH); // blink the LED and go OFF
    delay(delayval);
    digitalWrite(ledPin, LOW);
    delay(delayval);
}
```

Load Sketchbook – DM - ButtonDelay

Multiple Switches

Just like an LED – each switch needs its own resistor.

Same sub-circuit, just duplicate
Make Your Own Switches

- Anything that makes a connection
- Wires, tin foil, tinfoil balls, ball bearings
- Pennies!
- Nails, bolts, screws

- Or repurpose these tiny switches as bump detectors or closure detectors
Side Note - Power

- Servos can consume a bit of power
  - We need to make sure that we don’t draw so much power out of the Arduino that it fizzes
  - If you drive more than a few servos, you probably should put the servo power pins on a separate power supply from the Arduino
  - Use a wall-wart 5v DC supply, for example

- Not necessary for what we’re up to today!

Summary — Whew!

- LEDs – use current limiting resistors (220Ω to 470Ω) (remember color code!)
  - drive from `digitalWrite(pin, val);` for on/off
  - drive from `analogWrite(pin, val);` for PWM dimming (values from 0-255)

- buttons – current limiting resistors again (10k Ω)
  - active-high or active low (pullup or pulldown)
  - read with `digitalRead(pin);`

- potentiometers (pots)—voltage dividers with a knob
  - use with `analogRead(pin);` for values from 0-1023
Summary – Whew!

- **photocells – variable resistors**
  - use with current-limiting resistors (1k-10k)
    - (to make voltage divider)

- **Serial communications – write a value to the host**
  - communicate to the Arduino environment, or your own program

- **Servos – use Servo library to control motion**
  - might need external power supply
  - range of motion 0-180°

- Also **setup() and loop()** functions, and various libraries

Contact Information

- **Erik Brunvand**
  - School of Computing
  - University of Utah
  - Salt Lake City, UT 84112

  elb@cs.utah.edu
  http://www.cs.utah.edu/~elb