Part I – The Basics
Lutz Hamel
hamel@cs.uri.edu
www.cs.uri.edu/~hamel/as220
Workshop Overview

- Part I – *The Basics*
- Part II – *Interactive Design with advanced Transducers*
- Part III – *Multimedia Applications*
- Part IV – *Communication and Project Presentations*
Physical Computing

- The discipline of creating highly interactive objects using electronics and microcontrollers.
- Encourages an experimental approach.
- Values new experiences over precise theoretical foundations.
- Sometimes also called *Physical Interactive Design*
Arduino Microcontrollers

- Very small computers on a single chip.
- Designed to interface efficiently with the physical world
  - Serial Comm. Ports
  - Digital IO Pins
  - A/D Converters
Interactive Applications

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software.

- It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.
- It includes all the electronic components to perform basic experiments.
Developing Applications

- Develop programs called “Sketches”
- The Arduino IDE compiles these programs for the μC
- The IDE has tools to load the compiled programs onto the μC

Installation

- You will need to download the Arduino IDE and drivers for your development computer
  - Windows, Mac OX, Linux are all supported
- There is nothing you need to do for the Arduino board 😊
Mac OS X Install

- Download and install the IDE
- Download and install USB drivers
- Connect Arduino board
- Start IDE and select serial port
- Select type of Arduino (Duemilanove)
Download and install the IDE
Download USB drivers
Connect Arduino board
Use Wizard to install drivers
Start IDE and select serial port
Select type of Arduino (Duemilanove)
Blink – Our First Application

- Idea:
  - Connect a LED (light emitting diode) to the Arduino board
  - Write a sketch that turns the LED on and off periodically.
Arduino

Blink – Our First Application

- **Hardware:**
  - 1 LED – Polarized, long leg (+)
  - 1 Resistor (1KΩ) – Color coded: brown, black red
  - Breadboard
  - 2 Long wires
NOTE: The legs of the same component should **never** be connected to the same metal strip – short circuit!
Arduino

Blink – Hardware

Schematic

Complete System

Breadboard Layout
Sketches consist of two sections:
- setup
  - initialize µC
  - init. IO ports
  - etc.
- loop
  - this is where the processing happens.

Once your sketch is done
- verify it (compile it)
- upload it (runs automatically once uploaded)
Arduino

Blink – Our First Application

```c
/*
 * Blink
 *
 * The basic Arduino example. Turns on an LED on for one second,
 * then off for one second, and so on...
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 8; // LED connected to digital pin 8

void setup()
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000); // waits for a second
  digitalWrite(ledPin, LOW); // sets the LED off
  delay(1000); // waits for a second
}
```
Virtually every electronic circuit can be represented as either this simple circuit or a combination of these simple circuits.

Arduino

Basic Electronics

Flash Light

Switch

Battery

Lamp

Voltage Source

Load

Our Blink Circuit
The dreaded *short circuit*:  
- this is a circuit with a load equal to zero  
- this allows “infinite” current to flow from the positive terminal of the voltage source to the negative terminal  
- *it will break stuff!*

- Always check your circuits carefully before applying power
- *Never* connect an Arduino output pin directly to ground, always use a *load resistor*
Basic Electronics

Some Electronic Symbols

Reading Digital Input

• Idea:
  • Read the input signal produced by a pushbutton on a digital input pin of the Arduino board
  • Switch LED on/off on digital output pin depending on the signal on the input pin

• Specifics:
  • We tie the input pin to ground in order to generate a digital zero or LOW signal
  • We tie the input pin to +5V in order to generate a digital one or HIGH signal
  • Be careful with short circuits!
Reading Digital Input

Hardware Layout

Arduino

GND Bus
/*
 * Based on Button
 * by DojoDave <http://www.0j0.org>
 *
 * Turns on and off a light emitting diode(LED) connected to digital
 * pin 0, when pressing a pushbutton attached to pin 2.
 *
 */

int ledPin = 0; // pin for the LED
int inputPin = 2; // input pin (for a pushbutton)
int val = 0; // variable for input pin status

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

void loop(){
  val = digitalRead(inputPin); // read input value
  if (val == HIGH) {
    digitalWrite(ledPin, HIGH); // turn LED ON
  } else {
    digitalWrite(ledPin, LOW); // turn LED OFF
  }
}
Our circuit consists of two “simple circuits”

The load of the first circuit controls the voltage source of the second circuit (indicated by red arrow)

Notice that if we had tied D2 in the first circuit directly to ground then the load would have had a short circuit
A/D converters take analog signals and convert them into sequences of numbers.

The Arduino has six onboard A/D converters.

Each A/D converter converts voltages between 0V and 5V into numbers between 0 and 1023.
Idea:

- Use a *potentiometer* to generate an analog signal
- Use a A/D converter to convert it to a digital signal
- Display the digital signal on the serial monitor in the Arduino IDE
- Our sampling frequency is one sample per second
Reading Analog Input
// Reading analog input
// sample an analog signal on analogPin
// write the digitized signal to the USB
// serial line
int analogPin = 0;
int val = 0;

void setup() {
    // initialize the USB serial line
    Serial.begin(9600);
}

void loop() {
    // get a sample from the A/D converter
    val = analogRead(analogPin);
    // write the value to the serial line
    Serial.print(val);
    Serial.print(" ");
    // wait a second until our next sample
    delay(1000);
}
Pulse Width Modulated (PWM) Signals

- μCs cannot generate analog output, but we can fake it by creating digital signals with different “duty cycles” - signals with different pulse widths.

- To the analog world the different duty cycles create different effective voltages.
Arduino

PWM Signals

50% Duty Cycle

25% Duty Cycle

75% Duty Cycle

Effective Voltage

Arduino Dimmer

- Idea:
  - Read an analog signal from an analog input
  - Use this input to set the brightness of a LED

- NOTE: the LED needs to be connected to a PWM capable digital output (Duemilanove: 3, 5, 6, 9, 10, or 11)
Arduino Dimmer
// LED dimmer

int ledPin = 9;  // LED connected to digital pin 9, this is a PWM capable output port
int analogPin = 0;  // potentiometer connected to analog pin 0
int val = 0;  // variable to store the read value

void setup()
{
    pinMode(ledPin, OUTPUT);  // sets the pin as output
}

void loop()
{
    val = analogRead(analogPin);  // read the input pin
    analogWrite(ledPin, val / 4);  // analogRead values go from 0 to 1023,
                                  // analogWrite values from 0 to 255
}
Simulating Soundwaves

Note: For a 20Hz sound wave we have $T = 50\text{ms}$, for a 200Hz sound wave we have $T = 5\text{ms}$.

Low pitched tone – long period $T$

High pitched tone – short period $T$
Arduino

Optical Theremin

- Idea:
  - Read an analog signal generated through a photoresistor
  - We interpret the digitized value from the A/D conversion as the period of the sound wave we want to generate
  - Generate one period of the sound wave, output it to the speaker and then sample the input again
Arduino Optical Theremin
Arduino

Optical Theremin

// Optical Theremin
// It will generate square wave on soundPin.
// The period/frequency of the wave is governed by
// the value read from the pot. It will generate a
// wave from roughly 20Hz to 200Hz

int soundPin = 9; // output on digital pin 9
int freqPin = 0;  // photoresistor connected to analog pin 0
int interval = 0; // variable to store the read value

void setup()
{
  pinMode(soundPin, OUTPUT); // sets the pin as output
}

void loop()
{
  // read the interval value - an interval value is
  // half a period of the sound wave
  interval = (analogRead(freqPin)/25 + 5)/2;

  // generate one whole period of the wave
  digitalWrite(soundPin, HIGH);
  delay(interval);
  digitalWrite(soundPin, LOW);
  delay(interval);
}
Things to do for Next Time

- Design a concept for an interactive object
  - for inspiration check out:
    http://www.arduino.cc/playground/Projects/ArduinoUsers
    http://www.instructables.com/tag/?q=arduino
  - Notice how many interactive objects there are in your everyday environment

- Individual or group projects

- Read “Getting Started with Arduino”, Chapters 1 through 4, and the Appendices