Intro to Arduino
Zero to Prototyping
in a Flash!

Material designed by Linz Craig, Nick Poole, Prashanta Aryal, Theo Simpson, Tai Johnson, and Eli Santistevan
Overview of Class

Getting Started:
Installation, Applications and Materials

Electrical:
Components, Ohm's Law, Input and Output, Analog and Digital

-----------------------------

Programming:
Split into groups depending on experience

Serial Communication Basics:
Troubleshooting and Debugging

Virtual Prototyping:
Schematics and PCB Layout in Fritzing
Arduino Board

“Strong Friend” Created in Ivrea, Italy in 2005 by Massimo Banzi & David Cuartielles
Open Source Hardware
AIMEL Processor
Coding is accessible & transferrable → (C++, Processing, java)
Arduino…

is the go-to gear for artists, hobbyists, students, and anyone with a gadgetry dream.

rose out of another formidable challenge: how to teach students to create electronics, fast.

http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino
Getting Started

- **SW Installation:** Arduino (v.1.0+)
  Fritzing
  SIK Guide Code
  Drivers (FTDI)

- **Materials:**
  SIK Guide
  Analog I/O, Digital I/O, Serial, & Fritzing handouts
  Arduino CheatSheet
Analog INPUTS

POWER
5V / 3.3V / GND

Digital I/O
PWM(3, 5, 6, 9, 10, 11)

SCL\SDA
(I2C Bus)

PWR IN

USB (to Computer)

RESET
Go ahead and plug your board in!
Arduino Shields

PCB  Built Shield  Inserted Shield

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Arduino Shields

Micro SD

MP3 Trigger

LCD
<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>Type</th>
<th>Function</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push Button</td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Digital Input</td>
<td>Switch - Closes or opens circuit</td>
<td>Polarized, needs resistor</td>
</tr>
<tr>
<td>Trim potentiometer</td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Analog Input</td>
<td>Variable resistor</td>
<td>Also called a Trimpot.</td>
</tr>
<tr>
<td>Photoresistor</td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Analog Input</td>
<td>Light Dependent Resistor (LDR)</td>
<td>Resistance varies with light.</td>
</tr>
<tr>
<td>Relay</td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Digital Output</td>
<td>Switch driven by a small signal</td>
<td>Used to control larger voltages</td>
</tr>
<tr>
<td>Temp Sensor</td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Analog Input</td>
<td>Temp Dependent Resistor</td>
<td></td>
</tr>
<tr>
<td>Flex Sensor</td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Analog Input</td>
<td>Variable resistor</td>
<td></td>
</tr>
<tr>
<td>Soft Trimpot</td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Analog Input</td>
<td>Variable resistor</td>
<td>Careful of shorts</td>
</tr>
<tr>
<td>RGB LED</td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Dig &amp; Analog Output</td>
<td>16,777,216 different colors</td>
<td>Ooh... So pretty.</td>
</tr>
</tbody>
</table>
SIK Components

- **Photo Resistor**
  - Quantity: 1

- **Piezo Element**
  - Quantity: 1

- **Temp. Sensor (TMP36)**
  - Quantity: 1

- **Transistor (P2N2222AG)**
  - Quantity: 2

- **DC Motor**
  - Quantity: 1

- **Push Button**
  - Quantity: 2

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Electricity \ Electronics Basic Concept Review

- Ohms Law
- Voltage
- Current
- Resistance
- Using a Multi-meter
Ohm's Law

Ohm's Law describes the direct relationship between the Voltage (V), Current (I), and Resistance (R) of a circuit.

The three different forms of Ohm's Law are as follows:

\[ V = I \cdot R \]
\[ I = \frac{V}{R} \]
\[ R = \frac{V}{I} \]
Electrical Properties

### Voltage
- Defined as the amount of potential energy in a circuit.
- **Units**: Volts (V)

### Current
- The rate of charge flow in a circuit.
- **Units**: Amperes (A)

### Resistance
- Opposition to charge flow.
- **Units**: Ohms (Ω)

\[ V = I \cdot R \]
Current Flow Analogy

High Current

Low Current

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Voltage Analogy

More Energy == Higher Voltage

Less Energy == Lower Voltage

\[ V = I R \]
Resistance Analogy

Big Pipe == Lower Resistance

\[ V = I \cdot R \]

Small Pipe == Higher Resistance

\[ V = I \cdot R \]

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Continuity – Is it a Circuit?

The word “circuit” is derived from the circle. An Electrical Circuit must have a continuous LOOP from Power ($V_{cc}$) to Ground (GND).

Continuity is important to make portions of circuits are connect. Continuity is the simplest and possibly the most important setting on your multi-meter. Sometimes we call this “ringing out” a circuit.
Measuring Electricity – Voltage

Voltage is a measure of potential electrical energy. A voltage is also called a potential difference – it is measured between two points in a circuit – across a device.
Measuring Electricity -- Current

Current is the measure of the rate of charge flow. For Electrical Engineers – we consider this to be the movement of electrons.

In order to measure this – you must break the circuit or insert the meter in-line (series).
Measuring Electricity -- Resistance

Resistance is the measure of how much opposition to current flow is in a circuit.

Components should be removed entirely from the circuit to measure resistance. Note the settings on the multimeter. Make sure that you are set for the appropriate range.
Prototyping Circuits
Solderless Breadboard

One of the most useful tools in an engineer or Maker’s toolkit. The three most important things:

• A breadboard is easier than soldering
• A lot of those little holes are connected, which ones?
• Sometimes breadboards break
What’s a Breadboard?
Solderless Breadboard

Each row (horiz.) of 5 holes are connected.

Vertical columns – called power bus are connected vertically.
Using the Breadboard to build a simple circuit

Use the breadboard to wire up a single LED with a 330 Ohm Resistor (Orange-Orange-Brown).

Note: the longer leg on the LED is the positive leg and the shorter leg is the negative.
Fritzing View of Breadboard Circuit

What happens when you break the circuit?

What if you wanted to add more than one LED?
Adding control – let’s use the Arduino and start programming!!!
Concepts: INPUT vs. OUTPUT

Referenced from the perspective of the microcontroller (electrical board).

**Inputs** is a signal / information going into the board.  
**Output** is any signal exiting the board.

Almost all systems that use physical computing will have some form of output.

What are some examples of Outputs?
Concepts: INPUT vs. OUTPUT

Referenced from the perspective of the microcontroller (electrical board).

**Inputs** is a signal / information going into the board. **Output** is any signal exiting the board.

**Examples:** Buttons Switches, Light Sensors, Flex Sensors, Humidity Sensors, Temperature Sensors…

**Examples:** LEDs, DC motor, servo motor, a piezo buzzer, relay, an RGB LED
Concepts: Analog vs. Digital

Microcontrollers are **digital** devices – ON or OFF. Also called – discrete.

**analog** signals are anything that can be a full range of values. What are some examples? More on this later…

---

![Digital Signal](image1)

![Analog Signal](image2)
Open up Arduino

Hints:

For PC Users ➔
1. Let the installer copy and move the files to the appropriate locations, or
2. Create a folder under C:\Program Files (x86) called Arduino. Move the entire Arduino program folder here.

For Mac Users ➔
1. Move the Arduino executable to the dock for ease of access.
2. Resist the temptation to run these from your desktop.
Arduino
Integrated Development Environment (IDE)

Two required functions / methods / routines:

```cpp
void setup() {
    // put your setup code here, to run once:
}

void loop() {
    // put your main code here, to run repeatedly:
    // runs once
}

// repeats
```
Settings: Tools → Serial Port

Your computer communicates to the Arduino microcontroller via a serial port → through a USB-Serial adapter.

Check to make sure that the drivers are properly installed.
Next, double-check that the proper board is selected under the Tools → Board menu.
Arduino & Arduino Compatible Boards
BIG 6 CONCEPTS

- `digitalWrite()`
- `analogWrite()`
- `digitalRead()`
- `if() statements / Boolean`
- `analogRead()`
- `Serial communication`
Let’s get to coding…

Project #1 – Blink
“Hello World” of Physical Computing

Psuedo-code – how should this work?

Turn LED ON → Wait → Turn LED OFF → Wait → Rinse & Repeat
Comments, Comments, Comments

Comments are for you – the programmer and your friends...or anyone else human that might read your code.

// this is for single line comments
// it’s good to put a description at the top and before anything ‘tricky’
/* this is for multi-line comments
   Like this...
   And this.... */

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
void setup()
{
    // put your setup code here, to run once:
}

void loop()
{
    // put your main code here, to run repeatedly:
}
Three commands to know…

```cpp
pinMode(pin, INPUT/OUTPUT);
 ex: pinMode(13, OUTPUT);

digitalWrite(pin, HIGH/LOW);
 ex: digitalWrite(13, HIGH);

delay(time_ms);
 ex: delay(2500); // delay of 2.5 sec.

// NOTE: -> commands are CASE-sensitive
```
Move the green wire from the power bus to pin 13 (or any other Digital I/O pin on the Arduino board.)
A few simple challenges
Let’s make LED#13 blink!

Challenge 1a – blink with a 200 ms second interval.

Challenge 1b – blink to mimic a heartbeat

Challenge 1c – find the fastest blink that the human eye can still detect…

1 ms delay? 2 ms delay? 3 ms delay???
Try adding other LEDs

Can you blink two, three, or four LEDs?
(Hint: Each LED will need its own 330Ω resistor.)

Generate your own morse code flashing

How about Knight Rider? Disco? Police Light?
Programming Concepts: Variables

```c
int sensorValue;
int ledPin;

void setup()
{
    // put your setup code here, to run once:
    int setupVariable;
}

void loop()
{
    // put your main code here, to run repeatedly:
    int loopScopeVariable
}
```

Variable Scope

- **Global**
- *Function-level*
Programming Concepts: Variable Types

Variable Types:

- **8 bits**
  - byte
  - char

- **16 bits**
  - int
  - unsigned int

- **32 bits**
  - long
  - unsigned long
  - float
Fading in and Fading Out
(Analog or Digital?)

A few pins on the Arduino allow for us to modify the output to mimic an analog signal.

This is done by a technique called:

Pulse Width Modulation (PWM)
Concepts: Analog vs. Digital

To create an analog signal, the microcontroller uses a technique called PWM. By varying the duty cycle, we can mimic an “average” analog voltage.

Pulse Width Modulation (PWM)
Project #2 – Fading
Introducing a new command…

`analogWrite(pin, val);`

**pin** – refers to the OUTPUT pin (limited to pins 3, 5, 6, 9, 10, 11.) – denoted by a ~ symbol

**val** – 8 bit value (0 – 255).

<table>
<thead>
<tr>
<th>0</th>
<th>0V</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>5V</td>
</tr>
</tbody>
</table>
Move one of your LED pins over to Pin 9

In Arduino, open up:

File → Examples → 01.Basics → Fade

```cpp
/*
Fade

This example shows how to fade an LED on pin 9 using the analogWrite() function.

This example code is in the public domain.
*/
```
/*

Fade

This example shows how to fade an LED on pin 9 using the analogWrite() function.

This example code is in the public domain.
*/

int led = 9;       // the pin that the LED is attached to
int brightness = 0; // how bright the LED is
int fadeAmount = 5; // how many points to fade the LED by
void setup() {  
  // declare pin 9 to be an output:  
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:  
void loop() {  
  // set the brightness of pin 9:  
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:  
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:  
  if (brightness == 0 || brightness == 255) {  
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect  
  delay(30);
}
Project# 2 -- Fading

Challenge 2a – Change the rate of the fading in and out. There are at least two different ways to do this – can you figure them out?

Challenge 2b – Use 2 (or more) LEDs – so that one fades in as the other one fades out.
Color Mixing
Tri-color LED

In the SIK, this is a standard – Common Cathode LED

This means the negative side of the LED is all tied to Ground.
Project 3 – RGB LED

Note: The longest leg of the RGB LED is the Common Cathode. This goes to GND.

Use pins 5, 6, & 9
How many unique colors can you create?

\[
# \text{ of unique colors} = 256 \cdot 256 \cdot 256 = 16,777,216 \text{ colors!}
\]

Use Colorpicker.com or experiment on your own.

Pick out a few colors that you want to try recreating for a lamp or lighting display...

Play around with this with the \texttt{analogWrite()} command.
RGB LED Color Mixing

```cpp
int redPin = 5;
int greenPin = 6;
int bluePin = 9;

void setup()
{
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}
```
RGB LED Color Mixing

```c
void loop()
{
    analogWrite(redPin, 255);
    analogWrite(greenPin, 255);
    analogWrite(bluePin, 255);
}
```
Project: Mood Lamp / Light Sculpture
Napkin Schematics

Emphasize the engineering design process with students. We like to skirt the line between formal and informal with a tool called Napkin Schematics.

1. Short Description

Write a brief description of your project here. List inputs and outputs, existing systems it will integrate with and any other notes that occur to you. Don't spend too long on this section.

2. Sketch

Sketch an image of what you imagine your project or system to look like here.

3. Block Diagrams

Draw a diagram where each of the components in your project is represented by a simple square with lines connecting the components that will be connected. Don't worry about getting all the connections perfect; what's important is that you're thinking about all the different components and connections. Be sure to include things like power sources, antennas, buttons or other interface components and always include at least one LED to indicate the system is on. Although you'll probably want more LEDs than just the one, they make troubleshooting and debugging easier.
Napkin Schematics

Emphasize the engineering design process with students. We like to skirt the line between formal and informal with a tool called Napkin Schematics.

A. Logic Flow

Logic Flow Charts are a great way to sketch out how you want a circuit or chunk of code to act once it is completed. This way you can figure out how the whole project will act without getting distracted by details like electricity or programming.

There are four major pieces that you will use over and over again when creating Logic Flow Charts. A circle, a square, a diamond and lines connecting all the circles, squares and diamonds represent these four Logic Flow pieces.

The circle is used to represent either a starting point, or a stopping point. This is easy to remember since you start every single Logic Flow Chart with a circle containing the word Start or Begin. Often you will end a Logic Flow Chart with an End or Finish circle, but sometimes there is no end to the chart and it simply begins again. This is the case with any circuits that never turn off, but are always on and collecting data.

The square is used to represent any action that has only one outcome. For example, when a video game console is turned on it always checks to see what video game is in it. It does this every time after it starts up and it never checks in a different way. This kind of action is represented by the square. It never changes and there is always only one outcome.

The diamond is used to represent a question or actions with more than one possible outcome. For example, once your video game has loaded there is often a menu with a bunch of options. This would be written in a Logic Flow Chart as a diamond with something like the words “Start Up Menu” written inside of it. Lines coming off the diamond leading to another square, diamond, or circle would represent each action the user can take from this menu. Maybe our example Logic Flow Chart would have three options leading away from the “Start Up Menu” diamond, one line to start a new game, one to continue a saved game and another for game settings. In the Logic Flow Chart each option is written beside the line leading away from the diamond. It is possible to have as many options as you like leading away from a diamond in a Logic Flow Chart.

The lines in a Logic Flow Chart connect all the different pieces. These are there so the reader knows how to follow the Logic Flow Chart. The lines often have arrows on them and lead to whichever piece (circle, square, diamond) makes the most sense next. The lines usually have explanation of what has happened when they lead away from diamonds, so the reader knows which one to follow. Often some of these lines will run to a point closer to the beginning of the Logic Flow Chart. For example, the “Save Game” option might lead back to the “Start Up Menu” diamond, or it might lead straight to “Save and Quit,” it’s up to you; all it has to do is make sense to you.
Driving Motors or other High Current Loads

NPN Transistor (Common Emitter “Amplifier” Circuit)

to Digital Pin 9
Input

Input is any signal entering an electrical system.

• Both digital and analog sensors are forms of input
• Input can also take many other forms: Keyboards, a mouse, infrared sensors, biometric sensors, or just plain voltage from a circuit
Project #4 – Digital Input

In Arduino, open up:

File → Examples → 02.Digital → Button
Digital Sensors (a.k.a. Switches)
Pull-up Resistor (circuit)
to Digital Pin 2
Digital Sensors (a.k.a. Switches)
Add an indicator LED to Pin 13

This is just like our 1\textsuperscript{st} circuit!
Digital Input

• Connect digital input to your Arduino using Pins # 0 – 13 (Although pins # 0 & 1 are also used for programming)

• Digital Input needs a `pinMode` command:
  ```
  pinMode (pinNumber, INPUT);
  ```
  Make sure to use ALL CAPS for `INPUT`

• To get a digital reading:
  ```
  int buttonState = digitalRead (pinNumber);
  ```

• Digital Input values are only `HIGH` (On) or `LOW` (Off)
Digital Sensors

- Digital sensors are more straightforward than Analog.
- No matter what the sensor there are only two settings: On and Off.
- Signal is always either HIGH (On) or LOW (Off).
- Voltage signal for HIGH will be a little less than 5V on your Uno.
- Voltage signal for LOW will be 0V on most systems.
We declare a variable as an integer.

We set it equal to the function `digitalRead(pushButton)`

The function `digitalRead()` will return the value 1 or 0, depending on whether the button is being pressed or not being pressed.

`int buttonState = digitalRead(pushButton);`

We name it `buttonState`

The value 1 or 0 will be saved in the variable `buttonState`.

Recall that the `pushButton` variable stores the number 2

http://opensourcehardwarejunkies.com/tutorial-03-digitalread-and-
Programming: Conditional Statements

if ()

if (analogValue > threshold) {
    digitalWrite(ledPin, HIGH);
}
else {
    digitalWrite(ledPin, LOW);
}

If this is TRUE...

Do this.

Otherwise, do this.
void loop() {
    int buttonState = digitalRead(5);
    if (buttonState == LOW) {
        // do something
    }
    else {
        // do something else
    }
}
## Boolean Operators

<table>
<thead>
<tr>
<th>&lt;Boolean&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ) == ( )</td>
<td>is equal?</td>
</tr>
<tr>
<td>( ) != ( )</td>
<td>is not equal?</td>
</tr>
<tr>
<td>( ) &gt; ( )</td>
<td>greater than</td>
</tr>
<tr>
<td>( ) &gt;= ( )</td>
<td>greater than or equal</td>
</tr>
<tr>
<td>( ) &lt; ( )</td>
<td>less than</td>
</tr>
<tr>
<td>( ) &lt;= ( )</td>
<td>less than or equal</td>
</tr>
</tbody>
</table>
Trimpot (Potentiometer)  
Variable Resistor

fixed end

wiper

fixed end

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Analog Sensors
3 Pin Potentiometer = var. resistor (circuit)
a.k.a. Voltage Divider Circuit

Trimpot

10 kΩ

1.0 V

fixed ends

wiper

+5V

1.0 V

+5V

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Ohms Law… (just the basics)
Actually, this is the “voltage divider”

\[ V_{R1} = V_{CC} \cdot \left( \frac{R_1}{R_{Total}} \right) \]

\[ V_{R2} = V_{CC} \cdot \left( \frac{R_2}{R_{Total}} \right) \]

\[ R_{Total} = R_1 + R_2 \]
analogRead()

Arduino uses a 10-bit A/D Converter:

• this means that you get input values from 0 to 1023
  • 0 V → 0
  • 5 V → 1023

Ex:

```c
int sensorValue = analogRead(A0);
```
Using Serial Communication

Method used to transfer data between two devices.

Data passes between the computer and Arduino through the USB cable. Data is transmitted as zeros ('0') and ones ('1') sequentially.

Arduino dedicates Digital I/O pin #0 to receiving and Digital I/O pin #1 to transmit.
Serial Monitor & analogRead()

```cpp
int sensorValue = 0;
int sensorPin = A0;

void setup()
{
    Serial.begin(9600);
    pinMode(A0, INPUT);
}

void loop()
{
    sensorValue = analogRead(A0);
    Serial.println(sensorValue);
    delay(100); // waits by about 0.1 sec
}
```

- Initializes the Serial Communication
- 9600 baud data rate
- prints data to serial bus
Serial Monitor & analogRead()

```
int sensorValue = 0;
int sensorPin = A0;

void setup()
{
    Serial.begin(9600);
    pinMode(A0, INPUT);
}

void loop()
{
    sensorValue = analogRead(A0);
    Serial.println(sensorValue);
    delay(100);  // waits by about 0.1 sec
}
```
Analog Sensors

2 Pin Analog Sensors = var. resistor

Take two sensors -- Use the Serial Monitor and find the range of input values you get for each sensor.

MaxAnalogRead = __________

MinAnalogRead = __________
# Analog Sensors

## Examples:

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mic</td>
<td>soundVolume</td>
</tr>
<tr>
<td>Photoresistor</td>
<td>lightLevel</td>
</tr>
<tr>
<td>Potentiometer</td>
<td>dialPosition</td>
</tr>
<tr>
<td>Temp Sensor</td>
<td>temperature</td>
</tr>
<tr>
<td>Flex Sensor</td>
<td>bend</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>tilt/acceleration</td>
</tr>
</tbody>
</table>
Additional Serial Communication
Sending a Message

```cpp
void loop ()
{
    Serial.print("Hands on ");
    Serial.print("Learning ");
    Serial.println("is Fun!!!");
}
```
void setup()
{
  //
  Serial.begin(9600);
}

void loop()
{
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands on Learning is Fun!!!
  Hands o
}

Done uploading.

Binary sketch size: 1,980 bytes (of a 32,256 byte maximum)
Serial Communication: Serial Debugging

```c
void loop()
{
    int xVar = 10;
    Serial.print ( "Variable xVar is ");
    Serial.println ( xVar );
}
```

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Serial Communication: Serial Troubleshooting

```c
void loop ()
{
    Serial.print ("Digital pin 9: ");
    Serial.println (digitalRead(9));
}
```
Virtual Electrical Prototyping Project
started in 2007 by the Interaction Design Lab
at the University of Applied Science Potsdam, Germany

Open Source

Prototypes: Document, Share, Teach, Manufacture
Now that you feel comfortable putting together circuits with your breadboard let’s talk about how to go from the breadboard to a PCB
Free Time

The rest of the class is dedicated to free pursuit

Experiment with the various circuits and lessons in the SIK.

Explore the additional tutorials available on learn.sparkfun.com

Thank you for attending our Intro to Arduino class
Questions?