

# Introduction to Art Engineering

Medway High School

---

Fred Cahill, Shawn Nock

Unlondon Digital Media Assoc.

# Goals

---

- Science, technology, engineering and maths; in service of Art.
- Allows unprecedented interactivity
- Reach Kids, non-traditional art audiences
- Opens doors to new funding sources, non-traditional gallery space.

- 121Studios: Coworking for Creatives
- Unlab: Hackerspace
- Events: STEAM Outreach & Edu., ExplodeConf, X, Y, Z

## Freelance Engineer, Father

- Indoor location tracking w/ Bluetooth
- Keychain / Fitness Band Widgets
- Joystick for VR
- Remote Controls
- Internet of S\*#t

# Shawn: The Fun Stuff

Hacker, Church of the Weird Machine, Odd Duck

- Arduino compatible implant
- EEG Games / Toy Hacking
- Brain Stimulation
- Be Weird, Make Weird, Have Fun!
- Bad at "Art"



*“... she explores the boundaries between technology, society, and creative expression, using her unique perspective to try and help illuminate what makes us human.”<sup>1</sup>*

- Eclipse
- Forest
- Zen Photon Garden

---

<sup>1</sup>Micah's Portfolio Website: [misc.name](#)

*“With a background in fine art, world music, and carpentry, Kim Alpert brings an attention to detail and diverse style to her work.”<sup>2</sup>*

- Bodyphonic @ National Music Center, Calgary

---

<sup>2</sup>Kim's Portfolio Website: <http://aestheticengineer.com>



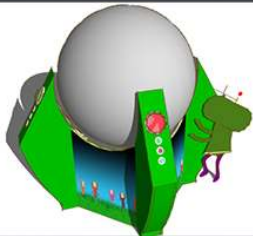
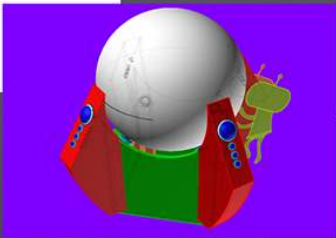
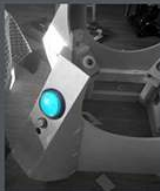
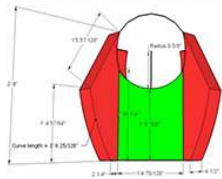
Hey Fred, how about a bio? Then your images follow. Seemed like a logical flow. . . intro shawn; shawn talks about his heros, then switch.



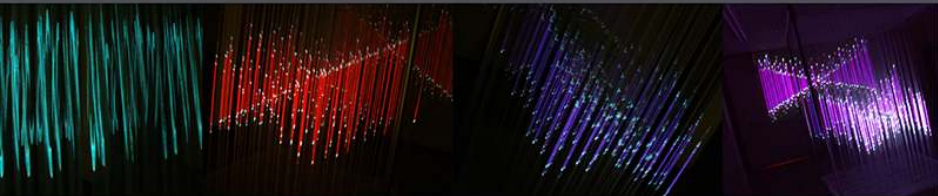
# WORLD RECORD TOWER



# ROLL UP THE COSMOS



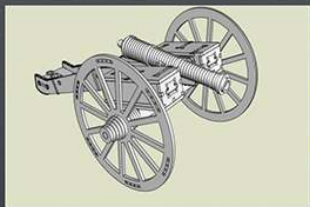
# LUMARCA



# K'NEX EXHIBITIONS



# LASER CUT CANNON



**What's in your kit?**

---



# Kit Contents

- Arduino Uno R3 Clone
- Solderless Breadboard
- Connecting wires
- LEDs
- Resistors, Potentiometer
- Buzzer
- IR Remote
- IR Receiver

# What is Arduino?

$\mu$ C + reset button + led + USB

It's a kit (on a board) with the bare minimum components to easily use the  $\mu$ C hardware. They do the basic, boring design needed for any board, so users only need to add the neat stuff.

The Arduino folks also adapted an *Integrated Development Environment* (IDE) to their boards. This IDE allows us to easily write programs for their boards and then write the programs to the  $\mu\text{C}$ .

Get the Arduino IDE:

<https://www.arduino.cc/en/Main/Software>

Get installing

# Circuit Basics

---

Current is the flow of charge through a circuit. Conventionally we think of this as happening from HIGH (+) to LOW (–)

# Voltage / Potential / Resistance

Voltage is how fast the current can move in the circuit. River metaphor:

- current = flow rate: ( $\text{L s}^{-1}$ )
- voltage = change in height: (m)

Other devices in a circuit can impede / effect current flow. We'll call them resistance(s).

A circuit is a completed loop from HIGH potential (voltage) to LOW, which causes current to flow through some other components along the way.



# Transducers

Often these *other* components are *transducers*, which convert electrical energy into another sort of energy:

---

Speaker	Electrical → Sound
Microphone	Sound → Electrical
LED	Electrical → Light
LED	Light → Electrical
Piezoelectric	Electrical → Motion

---

The power supply provides the energy to drive the system.

Can be a:

- Voltage Regulator (converts one potential to another)
- Batteries
- Solar Panel

In our circuits, your laptop is converting it's power source to 5 V and delivering power to our circuit via USB. You also have a battery pack for computer-free shenanigans (6 V).

Microcontroller ( $\mu$ C) is a *processor*, *memory* and a few *peripherals* on a standalone chip.

**Processor** is a group of transistors that understands a dozen or so commands (ADD, SUB, JUMP..)

**Memory** a circuit that can hold values.

**Peripherals** Vary chip to chip, but often include timers, communications and ADC, DAC.

Seems complicated, but really simple. They read a command from the start of memory, then execute the command. At the end of the command, read the next command from the next memory cell and repeat<sup>3</sup>

---

<sup>3</sup>some commands change the address of the next fetched command

- Vcc: The power supply of the circuit elements
- GND: The reference voltage (usually 0V)
- Connecting a part to Vcc = Logical 1
- Connecting to GND = Logical 0
- Connecting to Vcc & Ground is all the  $\mu\text{C}$  can do to talk to the world <sup>4</sup>

---

<sup>4</sup>w/o fancy peripherals

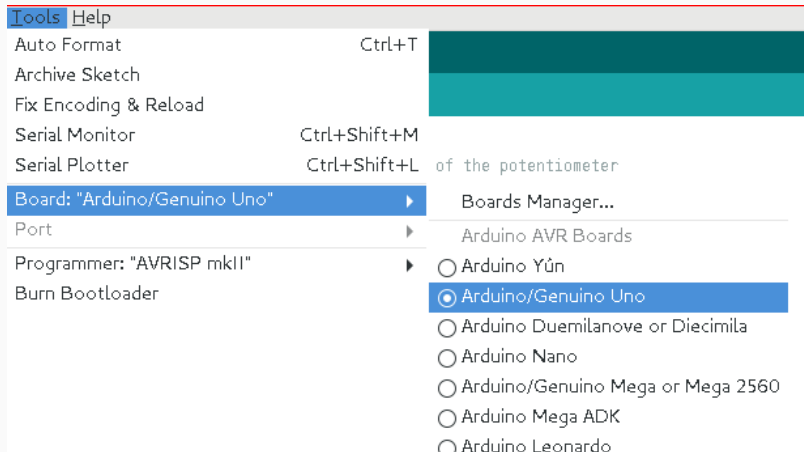
If one end of an LED is connected to ground, and the other end is connected to a pin on a  $\mu$ Controller, then:

- If the  $\mu$ C sets the pin HIGH ( $V_{cc}$ , 5V) then current will flow from the pin through the LED and turn it on.
- If  $\mu$ C sets the pin LOW (GND, 0V) then the current will not flow and the LED is off.

**Let's start programming**

---

# Configure Arduino



- Board: Arduino/Genuino UNO
- Port: ...

## Fetch the Class Code

- Download and extract:  
<https://nocko.se/assets/arduino-medway.zip>
- File→Preferences
- Browse for sketchbook
- Point it at the sketchbook subfolder of the extracted download
- You should now see a list of projects in the File→Sketchbook menu.



# The Code Environment



## Your first Program

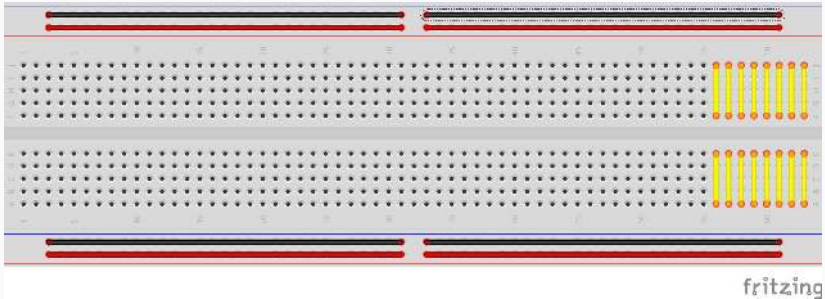
```
/* the setup function runs once on reset / power */
void setup() {
    /* set pin 13 as an output */
    pinMode(13, OUTPUT);
}

/* the loop function repeats forever */
void loop() {
    digitalWrite(13, HIGH);    // turn on LED
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);    // turn the off LED
    delay(1000);              // wait for a second
}
```

## Add Some Parts

---

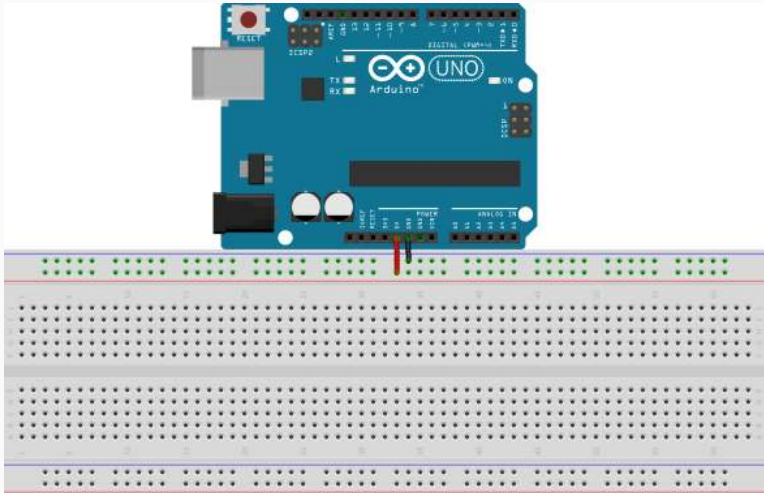
# Breadboard



- Connectors gently pinch component leads, wires.
- Have internal connections

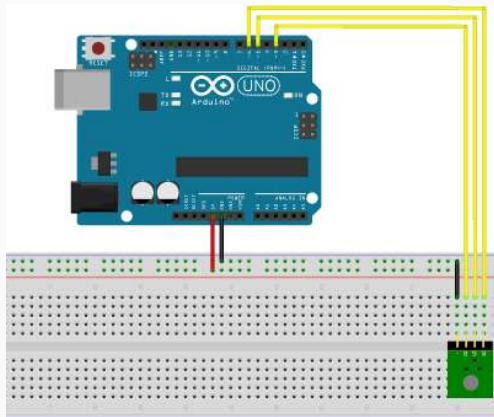
# Power Up the Rails

We use the long rows to distribute power. The Arduino outputs 5V on the pin marked 5V, the reference (GND) is marked GND.



# RGB LED

- Three LEDs in the same package.
- LEDs share the same GND (–) pin, one (+) side of each LED
- Connect – to negative rail, R, G, & B to pins 3,5, & 6 on Arduino



## RGB Blink, pt. 1

```
#define RED 6
...
#define DELAY_MS 1000

void setup() {
    /* initialize digital pin functions */
    pinMode(RED, OUTPUT);
    ...
}
```

## RGB Blink, pt. 2

```
/* turn the RED LED */  
digitalWrite(RED, HIGH);  
/* Do nothing for a while */  
delay(DELAY_MS);  
/* turn the LED off */  
digitalWrite(RED, LOW);  
delay(DELAY_MS);  
/* Continue on to green LED */  
digitalWrite(GREEN, HIGH);  
...
```



## More Parts

---

# Potentiometer

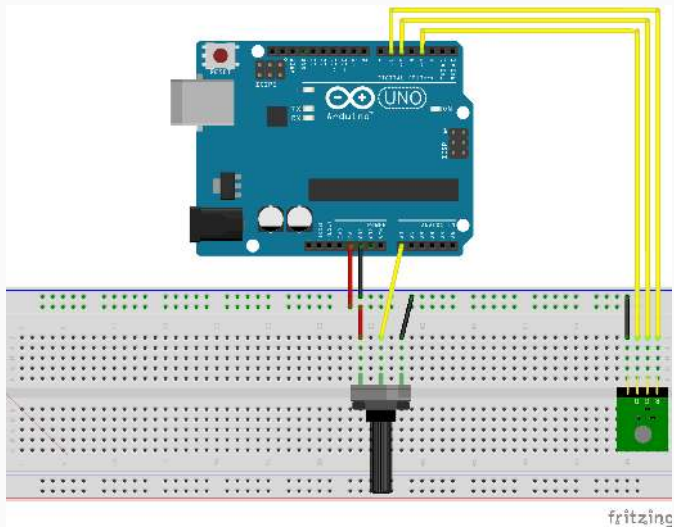
*Puh - ten - she - omer*

- *Pot* for short
- A Voltage Divider
- Potential at *Wiper* varies between the two terminals

# ADC: Analog to Digital Converter

- A peripheral of the  $\mu$ Controller
- Measures Potential, outputs a number
- In our case,  $0\text{ V} \rightarrow 0$  and  $5\text{ V} \rightarrow 1023$
- A0-A5 pins on Arduino can be used
- Fun uses: Reading pot position, sampling audio, reading from sensors

# The Pot Hookup



Connect center pin to A0, outer pins to (+) and (-) rails

## Pot Code, pt. 1: Variable

Declare a variable:

```
int delay_ms = 1000;
```

<type> <name> [= <initial value>]; (value optional)

It's a name, like a preprocessor #define, but the value can change at *runtime*

## Pot Code, pt. 2: ADC

`analogRead(pin)` returns the current state of the pin (0–1023), it can be assigned to a variable.

```
void loop() {  
    delay_ms = analogRead(A0);  
    digitalWrite(REDA, HIGH);  
    delay(delay_ms);  
    ...  
}
```

Each time through the loop, a new `delay_ms` value is read. Since the subsequent delay calls use `delay_ms`, the blink rate changes.

**What else can you do with a light?**

---

New variable:

```
int brightness = 0;
```



## Dimmer Code, pt. 2

`analogWrite(pin, <0-255>)`, sets the *average* voltage to 0 V @ 0 to 5 V @ 255.

```
void loop() {  
  brightness = analogRead(A0) >> 2;  
  analogWrite(RED, brightness);  
  ...  
}
```

## Bitwise Shift

Then number 100 is the number 4 in binary.

$$0c100 \ll 1 = 0b1000 = 8$$

$$0b100 \gg 1 = 0b10 = 2$$

$$0b10 \gg 1 = 0b1 = 1$$

Many  $\mu C$  do not have multiplication/division hardware, and they take a lot of time and power to fake it. For powers of two, shifting is faster/better.

# PWM

If  $\mu C$  can only output 0 and 1, how does "analogWrite" work?

We can turn the pin on and off very quickly and vary the \*duty cycle\* (the percentage of time the pin is HIGH).

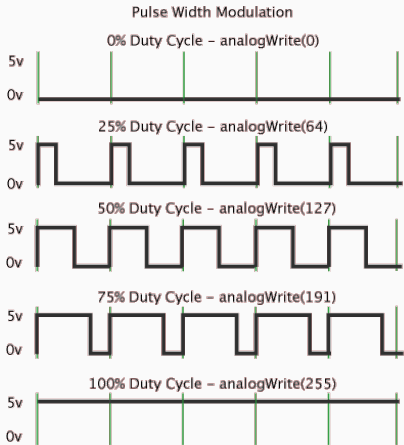
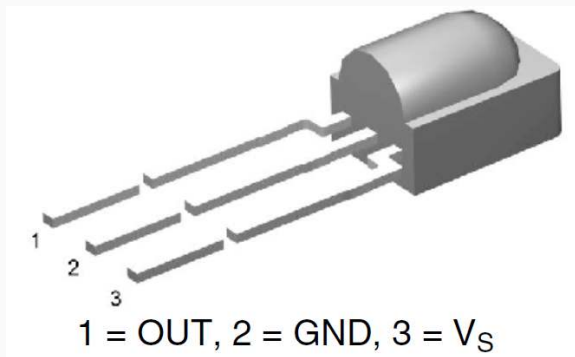


Image courtesy of Arduino.cc

# Remote Control

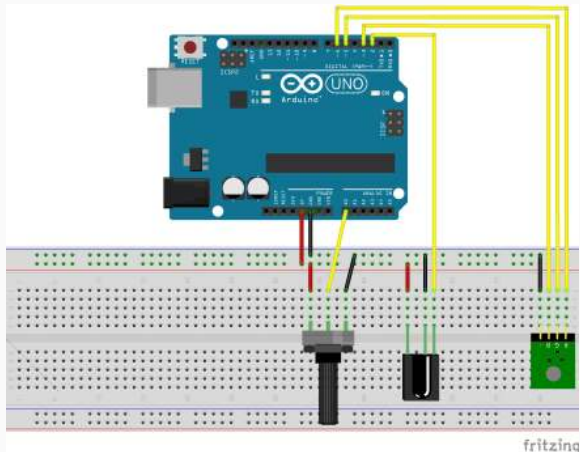
---

## IR Receiver



Neat piece of kit. Internally demodulates IR signal @ 38 kHz and outputs 16 bit code (unique to each button).

# Remote Control: Hardware Setup



OUT of IR Receiver connects to pin 2 of Arduino, GND (middle) GND rail, remaining pin to Vcc rail.

## Remote Control: Code, pt. 1

Imports and libraries. You can include code from other files using `#include`. These are often used to include library code, for example below the `IRremote.h` file includes declarations that allow us to use objects / methods from the `IRremote` library.

```
#include <IRremote.h>  
#include "medway-remote.h"
```

## Remote Control: Code, pt. 2

```
bool power = false;

/* Initialize the irrecv part of the IRremote library */
IRrecv irrecv(IR_PIN);
decode_results results;

void setup() {
    ...
    irrecv.enableIRIn();
}
```



## Bitwise AND

A function that compares the bit-by-bit two numbers. For each bit, it returns 1 if both input bits are 1, else 0. Examples

- $5 \& 1 = 1$ ;  $0b101 \& 0b001 = 0b001$
- $241 \& 133 = 129$ ;  $0b11110001 \& 0b10000101 = 0b10000001$

Why use this? Setting or clearing ranges of bits.

$0xF = 0b00001111$ , so anything  $\& 0xF$  will clear any bits *left* of the last four.

## Remote Control: Code, pt. 3

```
void loop() {
  if (irrecv.decode(&results)) {
    uint16_t resultCode = (results.value & 0xFFFF);
    switch (resultCode) {
      ...
      case ONE:
        digitalWrite(RED, !digitalRead(RED));
        break;
      case TWO:
        digitalWrite(GREEN, !digitalRead(GREEN));
        break;
    }
  }
}
```

```
irrecv.resume();
```

This tells the IR Receiver library that we've processed the current code, and it can provide (or wait) for the next one.

**Where to go next?**

---

# Light Painting

Program a blink / fade (see Fade example in sketchbook) / colour pattern into your led(s). Take a long exposure / multiple exposure photograph as you move the project around the room.

Hook up one end of the Piezo buzzer (black cylinder) to GND rail and the other to a  $\mu$ Controller pin (sample code in remote sketch). Turn on the buzzer and watch Fred and I squirm.

Your kit also contains a photoresistor, try hooking it up to an ADC pin. Your kit also has a modified LED that can act as a *flame sensor*, it may be fun to play with. . . .

Let's build some cool stuff!