Introduction to Arduino

Maker Workshop

Raphael Kopala, Shawn Nock

Unlondon Digital Media Assoc.

Goals

By the end of this class, you'll:

- Know how to create programs for Arduino and run them.
- Have learned about digital input and output, reading switches and lighting LEDs
- Have created a *Whack-a-Mole* type game.
- Be prepared to follow Arduino tutorials online and continue exploring.

Unlondon

Enabling Exploration, Creativity, and Excellence In Art+Make+Tech

Challenging and embracing ideas related to new technologies and social platforms through the education, entertainment and engagement of our membership and the community-at-large.

- 121Studios: Coworking for Creatives
- Unlab: Hackerspace
- Events: STEAM Outreach & Edu., ExplodeConf, Nuit Blanche

Shawn: Day Job

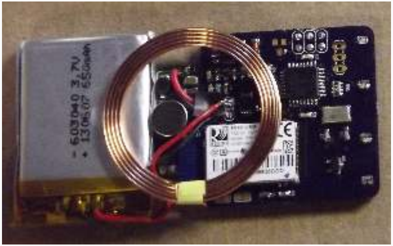
Freelance Embedded Systems Engineer

- 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!



Raphael: Day Job

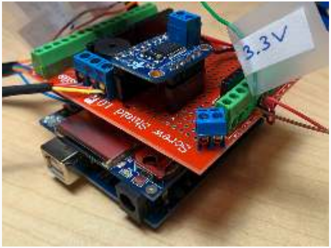
Mechanical Engineer

- Working in the medical device industry
- Experience in medical device R&D and Manufacturing
- Teaching SolidWorks CAD at Fanshawe

Raphael: The Fun Stuff

Thinker, Jack of all Trades - Master of None

- Arduino for Fun, and Odd Jobs
- 3D Printer Hobbyist
- PC Builder & Gamer
- Fish keeper



What’s in your kit?

Kit Contents

- Arduino Uno R3
- Solderless Breadboard
- Connecting wires
- LEDs
- Resistors, Potentiometer
- Buzzer

What is Arduino?

μC + reset button + led + USB

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

Arduino Software

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 μC.

Get the Arduino IDE:

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

Circuit Basics

Current

Current is the flow of charge through a circuit. Measured in Amperes (A).

Resistance / Impedance

Circuits have a resistance to current flow that depends on the parts in the circuit.

Measured in Ohms (Ω)

Voltage

Voltage is a potential (akin to a pressure) pushing the current through a circuit. Current is said to flow from higher (+) voltage to lower (-) voltage.

Measured in Volts (V)

Ohm’s Law; Light

Voltage, Current and Resistance are related to each other.

- As voltage increases, current increases
- As voltage decreases, current decreases
- As resistance increases, current decreases
- As resistance decreases, current increases

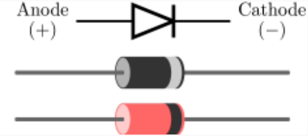
V, Ω, A: **The Water Analogy**

If charge were water, then:

- resistance = obstacles blocking flow
- current = flow rate
- voltage = change in height *or* pressure.

Diode

- One way value for current[1]
- LED $\equiv$ Light Emitting Diode
- Band marks (-)[2]
- Longer leg marks (+)



[1] https://learn.sparkfun.com/tutorials/diodes

[2] https://learn.sparkfun.com/tutorials/polarity/diode-and-led-polarity

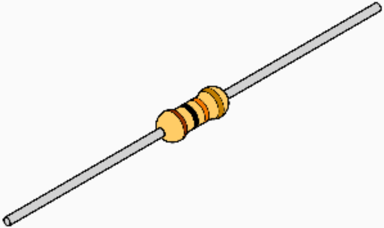
Diode Problems

- Diodes don’t limit current
- Diodes aren’t perfect (some current turned to heat)
- Too much current → Too much heat →

What’s that smell?

Resistor

- *Resists*/limits the flow of current
- Needed for LEDs: $\approx 1000\,\Omega$
- Button Pull-up/down: $\geq 10\,\mathrm{k}\Omega$
- Color coded, Google it



Buttons

- Buttons connect *or* disconnect two wires/parts
- Momentary Switch: Normally Closed (NC), Normally Open (NO)
- Toggle Switch

Circuits

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

Power

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.

μController

Microcontroller (μC) 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[1]

[1] some commands change the address of the next fetched command

Digital Signals

- Vcc: The power supply of the circuit elements
- GND: The reference voltage (usually 0 V)
- Connecting a part to Vcc = Logical 1 or High
- Connecting to GND = Logical 0 or Low
- Connecting various pins to Vcc or Ground is all the μC can do to talk to the world [2]

[2] w/o fancy peripherals or dirty tricks

μController INPUT and OUTPUT

Most of the pins on the Arduino can be set for INPUT or OUTPUT mode.

- INPUT mode pins listen for a signal (0 or 1) from another device
- OUTPUT mode pins drive the pin High or Low

Floating Pins

What's happens if an INPUT mode pin tries to read the value of a pin that is connected to nothing? Is that a 1 or 0?

No one knows!

It's dependant of transient charges, static, nearby electric fields, the phase of the moon, . . . Whenever you want to check a digital signal, make sure that something is *driving* it (ensuring Vcc or GND).

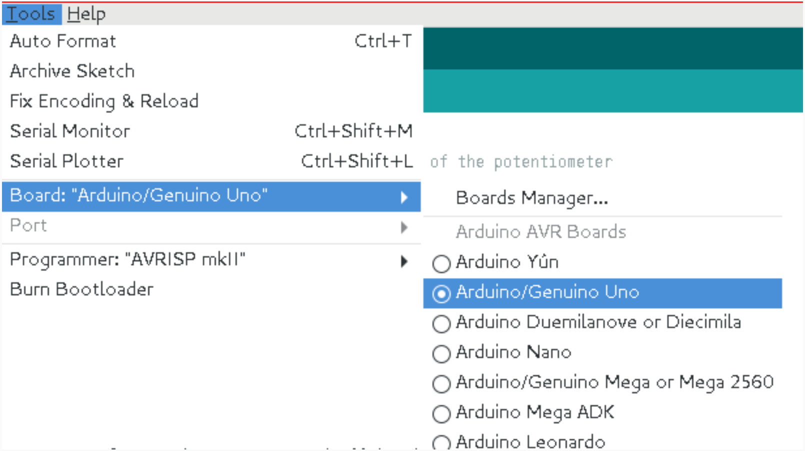
μC + Digital Signals as Switches

If one end of an LED is connected to ground, and the other end is connected to an OUTPUT pin on a μController, then:

- If the μC sets the pin High (Vcc, 5 V) then current will flow from the pin through the LED and turn it on.
- If μC sets the pin Low (GND, 0 V) then the current will not flow and the LED is off.

Let’s start programming

Configure Arduino

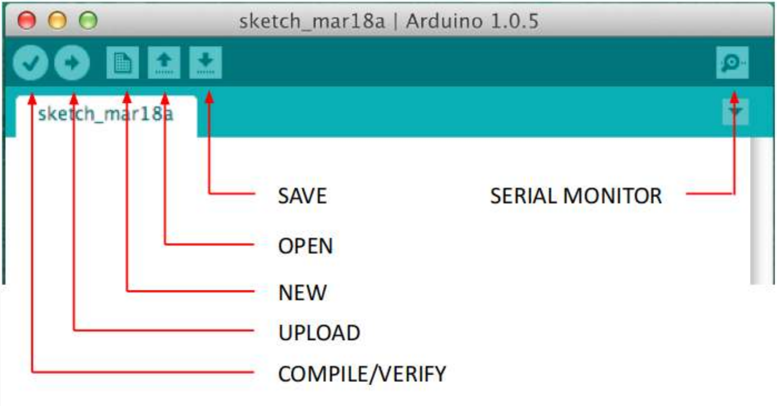


Board: "Arduino/Genuino Uno"

Programmer: "AVRISP mkII"

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

The Code Environment



Your first Program

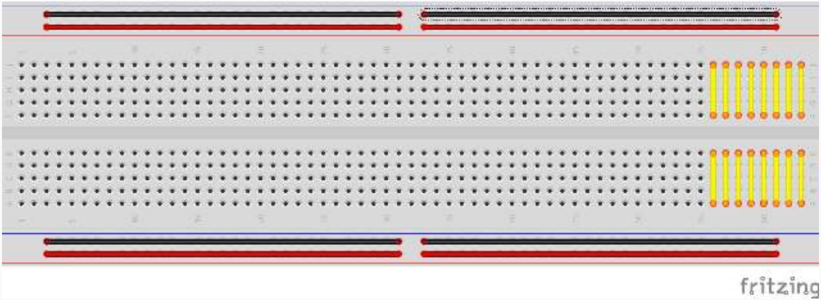
```
#define LED 13

/* the setup function runs once on reset / power */
void setup() {
  pinMode(LED, OUTPUT);
}

/* loop() repeats until reset or power off */
void loop() {
  digitalWrite(LED, HIGH);   // turn on LED
  delay(1000);              // wait for a second
  digitalWrite(LED, LOW);    // turn the off LED
  delay(1000);
}
```

Add Some Parts

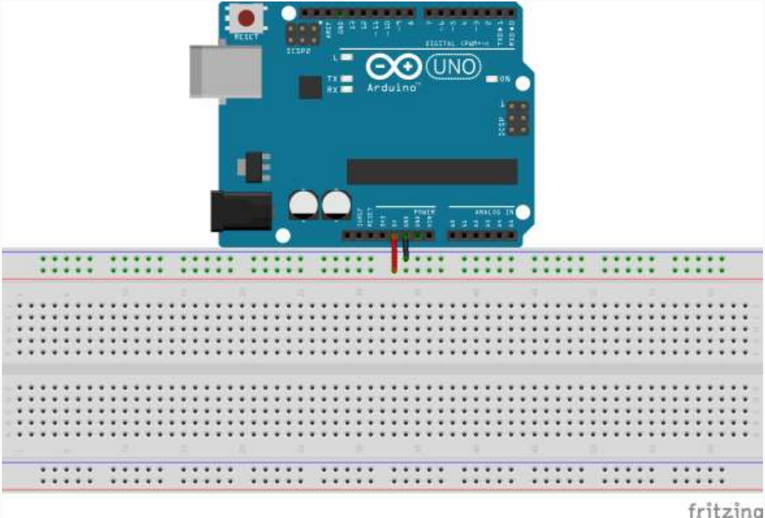
Breadboard



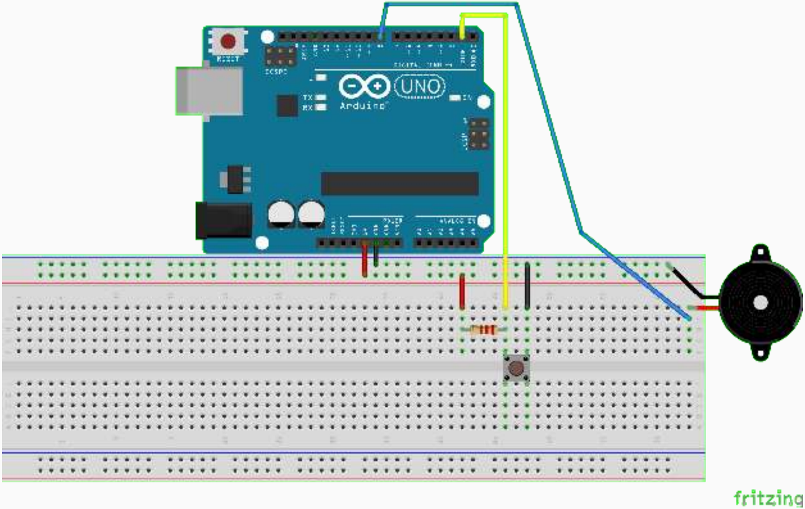
- Connectors gently pinch component leads, wires.
- Have internal connections

Power Up the Rails

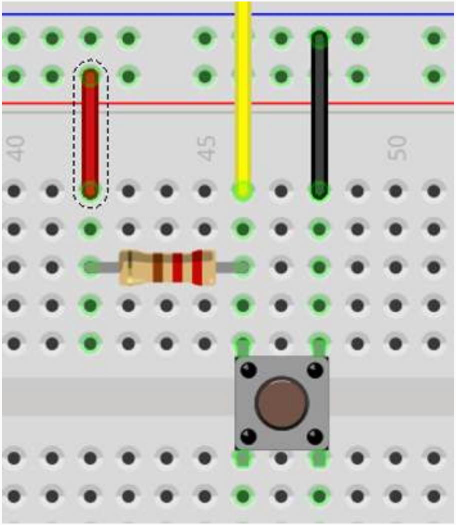
We use the long rows to distribute power. The Arduino outputs 5 V on the pin marked `5V`, the reference (GND) is marked `GND`.



Buzzer & Button: Hardware



Push Button: Zoom



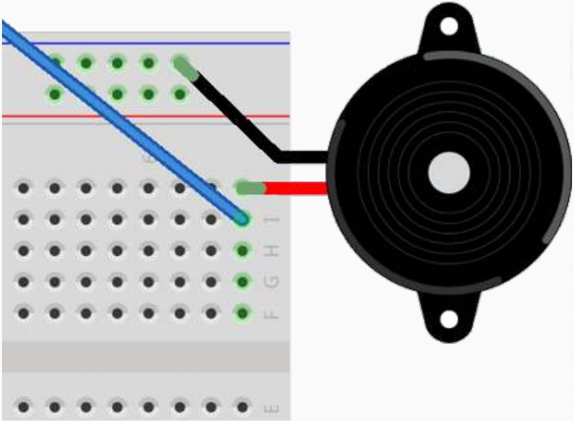
Pullup / Pulldown Resistors

Reading a floating pin is **bad**. A switch only connects and disconnects a wire. When the wire is disconnected… the INPUT pin is floating!

Solution:

Connect the pin to Vcc so that it reads High; use a resistor to prevent short circuit (limit current).

Buzzer: Zoom



Buzzer / Button: Software (Part 1)

```
#define BUTTON 2
#define BUZZER 8

int button_state;

void setup() {
  pinMode(BUTTON, INPUT);
  pinMode(BUZZER, OUTPUT);
  digitalWrite(BUZZER, LOW); /* Start w/ LED off */
}
```

Programming Note: Variables

Declare a variable:

```
int button_state = HIGH;
```

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

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

Programming Note: *If* Statement

```
if (condition) {
    // body: Runs if condition true ( != 0)
} else {
    // Runs if condition false ( == 0 )
}
```

- body code inside curly braces: `{ }`
- **condition** evaluates to 0 → body code skipped
- else section is optional, runs if **condition** evaluates to 0
- **condition** evaluates to *not* 0 → body code runs

Programming Note: ==

In C-like languages, the == operator checks if two things (statements, variables, . . .) are equal to each other.

- It returns 1 if the items are equal, *or*
- It returns 0 if the items are not equal

Programming Note: Functions

Functions make it easy to reuse code. You already know / use several functions:

- pinMode
- digitalWrite
- delay

digitalRead(pin number) returns HIGH or LOW depending on current state of any **INPUT** pin.

You can write your own functions!

Programming Note: Writing Functions

```
void my_function(int arg1, ...) {
    // Do fun things
}
```

void: Return type. Void means nothing returned. Can be any type.

my_function: A name for your function

arguments: A type and name for any parameters you want to use in your function from the outside.

Define a function once, you can use it again and again. Better than copy/pasting.

Push Button: Software (Part 2)

```
void buzz(int ms) {
    digitalWrite(BUZZER, HIGH);
    delay(ms);
    digitalWrite(BUZZER, LOW);
}
void loop() {
  button_state = digitalRead(BUTTON);
  if (button_state == LOW) {
    buzz(100);
  }
}
```

More Parts

Potentiometer

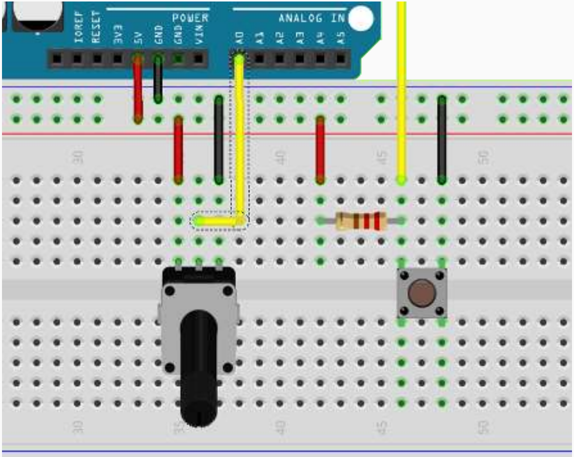
Puh - ten - she - ometer

- *Pot* for short
- A Voltage Divider
- Voltage at *Wiper* is somewhere between potential at the two terminals.
- The exact wiper potential depends on the position of the knob/lever.

ADC: Analog to Digital Converter

- A peripheral of the μController
- Measures Potential, outputs a number
- In our case, $0\,\mathrm{V} \rightarrow 0$ and $5\,\mathrm{V} \rightarrow 1023$
- 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

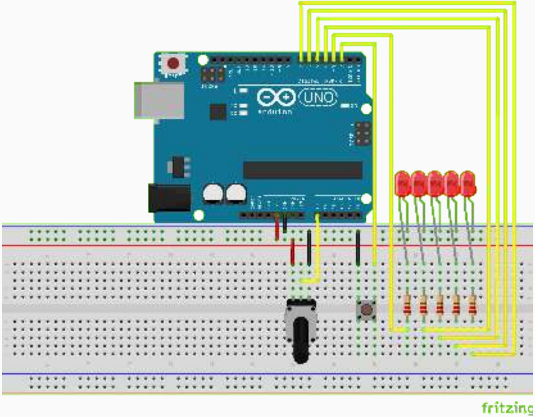
analogRead(*pin*) returns the voltage at the pin (0–1023), it can be used directly or via variable.

```
#define LED 13
void setup() {
  pinMode(LED, OUTPUT);
}
void loop() {
  digitalWrite(LED, HIGH);
  delay(analogRead(A0));
  digitalWrite(LED, LOW);
  delay(analogRead(A0));
}
```

Since the delay() calls use the result of analogRead (0-1023), the blink rate changes with knob position.

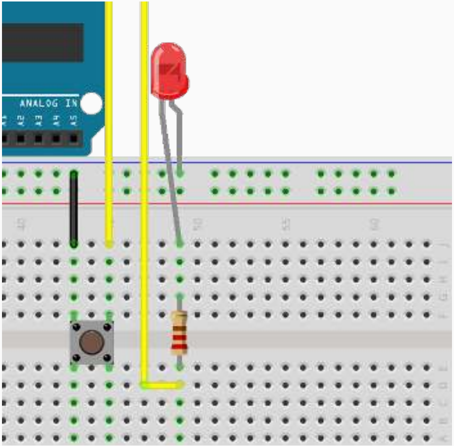
Shall we play a game?

Hooking up a bunch of LEDs



LEDs, the first one

Looks complicated, but for each LED: The short leg goes to ground, the long leg goes to one end of a resistor, and the other end of the resistor goes to the arduino pin.



Programming Note: `for` Loop

```
for ( initializer ; condition; increment ) {
    // This body will repeat while condition != 0
}
```

initializer Executed once at beginning of loop. Often used to declare a local variable.

condition Loop will repeat while condition is true ($\neq 0$)

increment Runs *after* each loop. Often used to increment variables.

All fields are optional

Cylon Simulator: Part. 1; Pin Setup

```
setup() {
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(7, OUTPUT);
}
```

Cylon Simulator: Part. 2

```
loop () {
    for (int i = 4; i <= 7; i++) {
        delay_ms = analogRead(A0);
        digitalWrite(i - 1, LOW);
        digitalWrite(i, HIGH);
        delay(delay_ms);
    }
    for (int i = 6; i >= 3; i--) {
        delay_ms = analogRead(A0);
        digitalWrite(i + 1, LOW);
        digitalWrite(i, HIGH);
        delay(delay_ms);
    }
}
```

Programming Note: `while` Loop

```
while ( statement ) {
    // This body will repeat while condition is true
    // True means statement != 0
}
```

initializer Executed once at beginning of loop. Often used to declare a local variable.

condition Loop will repeat while condition is true ($\neq 0$)

increment Runs *after* each loop. Often used to increment variables.

All fields are optional

Winner, Winner, Chicken Dinner

```
#define WINNER 5
void check_delay(int cur_led, int delay_ms) {
  unsigned long start = millis();
  while (millis() < start+delay_ms) {
    if (digitalRead(BUTTON) == LOW) {
      if (cur_led == WINNER) {
        do_winner();
      } else {
        while (digitalRead(BUTTON) == LOW) {
          do_loser();
        }
      }
    }
  }
}
```

More functions, pt. 1

```
void set_all_leds(int state) {
  for (int i = 3; i <= 7; i++) {
    digitalWrite(i, state);
  }
}

void do_loser(void) {
  buzz(500);
}
```

More functions, pt. 2

```
void do_winner(void) {
  set_all_leds(HIGH);
  buzz(100);
  delay(100);
  buzz(100);
  set_all_leds(LOW);
}
```

Putting it Together

```
loop () {
    for (int i = 4; i <= 7; i++) {
        delay_ms = analogRead(A0);
        digitalWrite(i - 1, LOW);
        digitalWrite(i, HIGH);
        check_delay(i, delay_ms);
    }
    for (int i = 6; i >= 3; i--) {
        ...
        check_delay(i, delay_ms);
    }
}
```

The End?

Extra Credit

Ohm's Law

Ohm's Law relates current to potential and resistance.

$$V = IR$$
$$I = \frac{V}{R}$$
$$R = \frac{V}{I}$$

- V = Potential in Volts (V)
- I = Current in Amperes (A)
- R = Resistance in Ohms (Ω)

Ohm's Law: Example

The datasheet for an LED says that the maximum continuous current is 15 mA. Your circuit operates at 5 V[1]. How big should your resistor be?

$$\Omega = \frac{5\,\text{V}}{0.015\,\text{A}} = 333.\overline{3}\Omega$$

How much current for our *cheet sheet* value?

$$\text{A} = \frac{5\,\text{V}}{1\,\text{k}\Omega} = 5\,\text{mA}$$

[1] Actually, this calculation is inaccurate. LEDs will have a *forward voltage drop* of between 1.8 V and 3.3 V this should be subtracted from V above... but it's not critical.

Current Limits, Arduino

- No single pin should source more that 20 mA (40 mA is absolute max)
- Pins are ganged together in groups of 8, no group should source more than 150 mA total
- The whole board cannot source more than 200 mA total

Practically speaking, this means that the Arduino cannot drive speakers, most motors, or anything normally mains powered.

So. . . no Arduino smart blender?

You can control almost anything with an arduino, you just can't power it with the Arduino. There are various devices that let you switch highier powered devices:

- Transistors
- Relays
- Solid State Relays
- Triac

HIGHs and LOWs

Many different logic levels are in common use: 1.2 V, 1.8 V, 2.5 V, 3.3 V, and 5 V. The voltage cited is the *nominal* Vcc of the system.

A HIGH signal is generally any voltage $\geq \frac{2}{3} V_{cc}$.

A LOW signal is generally any voltage $\leq \frac{1}{3} V_{cc}$.

HIGHs and LOWs, pt. 2

In your travels, you're likely to see both 5 V and 3.3 V sensors and peripherals.

Since $3.3\,\mathrm{V} \geq \frac{2}{3} V_c c$ your Arduino will accept input from a 3.3 V peripheral without issue.

If you drive an output to 5 V while it's connected to a 3.3 V peripheral with an Arduino **it will blow up your peripheral**.[3]

[3] In the datasheet for the sensor, it'll have a section called *Absolute Maximums*. Generally 3.3 V parts won't accept more that $\approx 3.6\,\mathrm{V}$, but some will.

HIGHs and LOWs, pt. 3

Solutions:

- Level Shifter: A dedicated chip that translates between voltages. Available as uni or bidirectional.
- Buy a 3.3V Arduino Compatible. Arduinos are available that operate at the lower voltage.