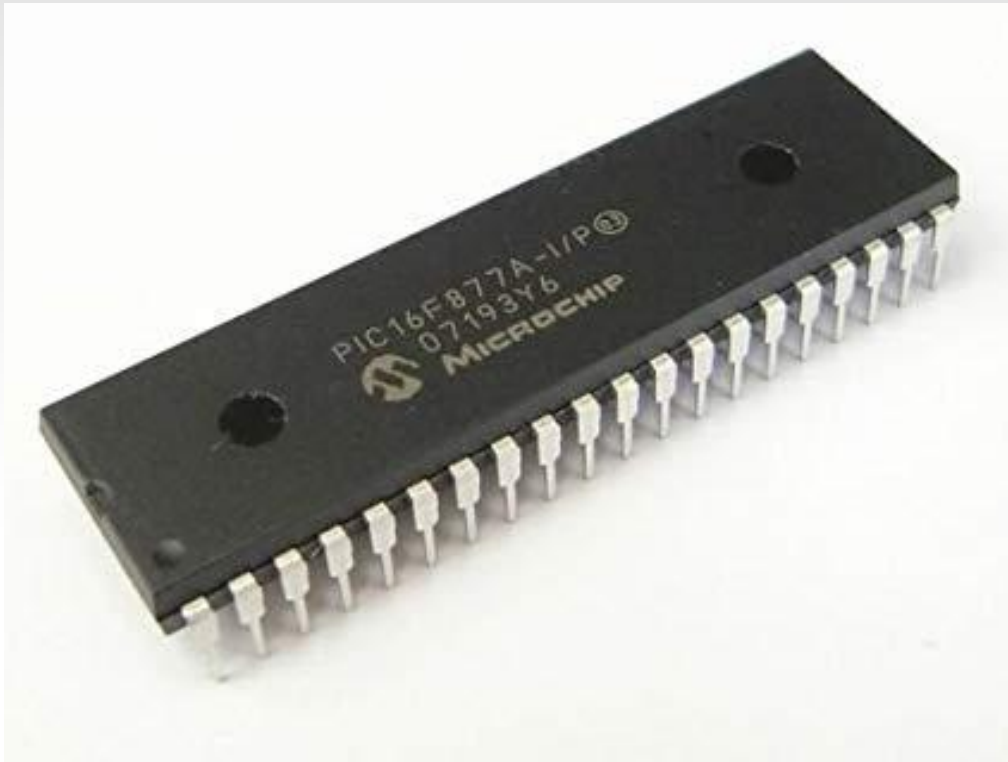


# Arduino & Friends

M1DST

# Back in the day...



```
1 list p=16f84a
2 include
3 COUNT1 EQU 08h
4 COUNT2 EQU 09h
5
6 org 0x00
7 goto start
8
9 start bsf STATUS, RP0 ;bank 1
10 movlw 0xFE
11 movwf TRISB ;set all PORTB input except for RB0
12 bcf STATUS, RP0 ;bank 0
13
14 main bsf PORTB, 0 ;make RB0 high
15 call delay ;delay subroutine
16 bcf PORTB, 0 ;make RB0 low
17 goto main
18
19 delay
20 loop1 decfsz COUNT1,1 ;decrement COUNT1 variable until zero
21 goto loop1
22 decfsz COUNT2,1 ;decrement COUNT2, if not zero, go back to
23 loop1
24 goto loop1
25 return
end
```

# Arduino Board

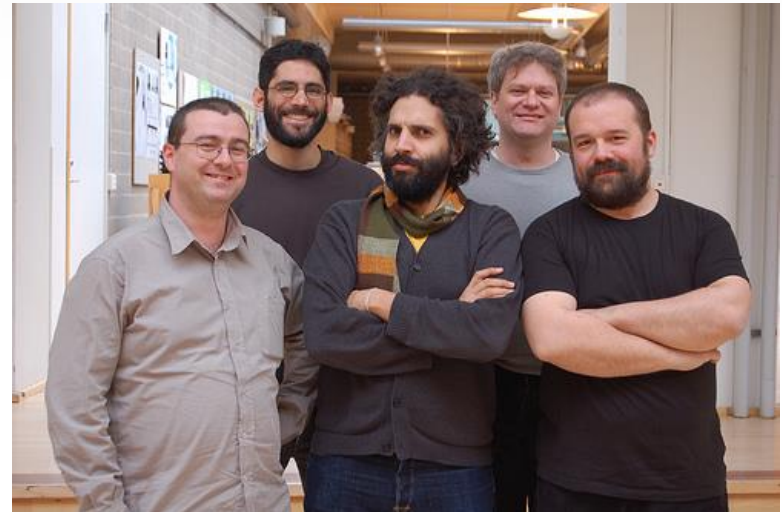
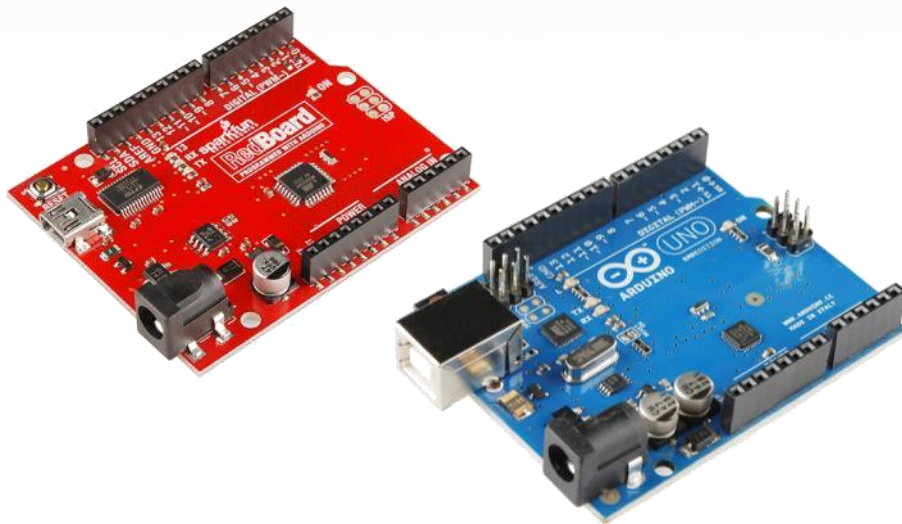
The name Arduino comes from a bar in Ivrea, Italy, where some of the founders of the project used to meet.

Created in 2005 by Massimo Banzi & David Cuartielles

Open Source Hardware

 Processor

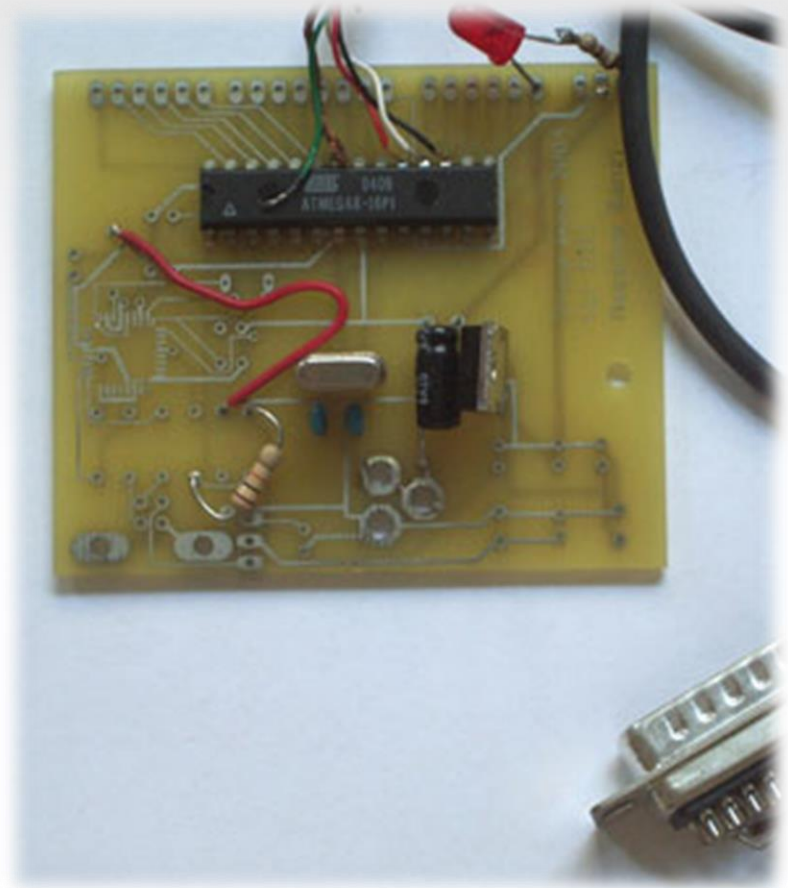
Coding is accessible & transferrable → (C++, Java etc)



# Arduino...

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

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



# Getting Started

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

**POWER**  
5V / 3.3V / GND

**Analog  
INPUTS**

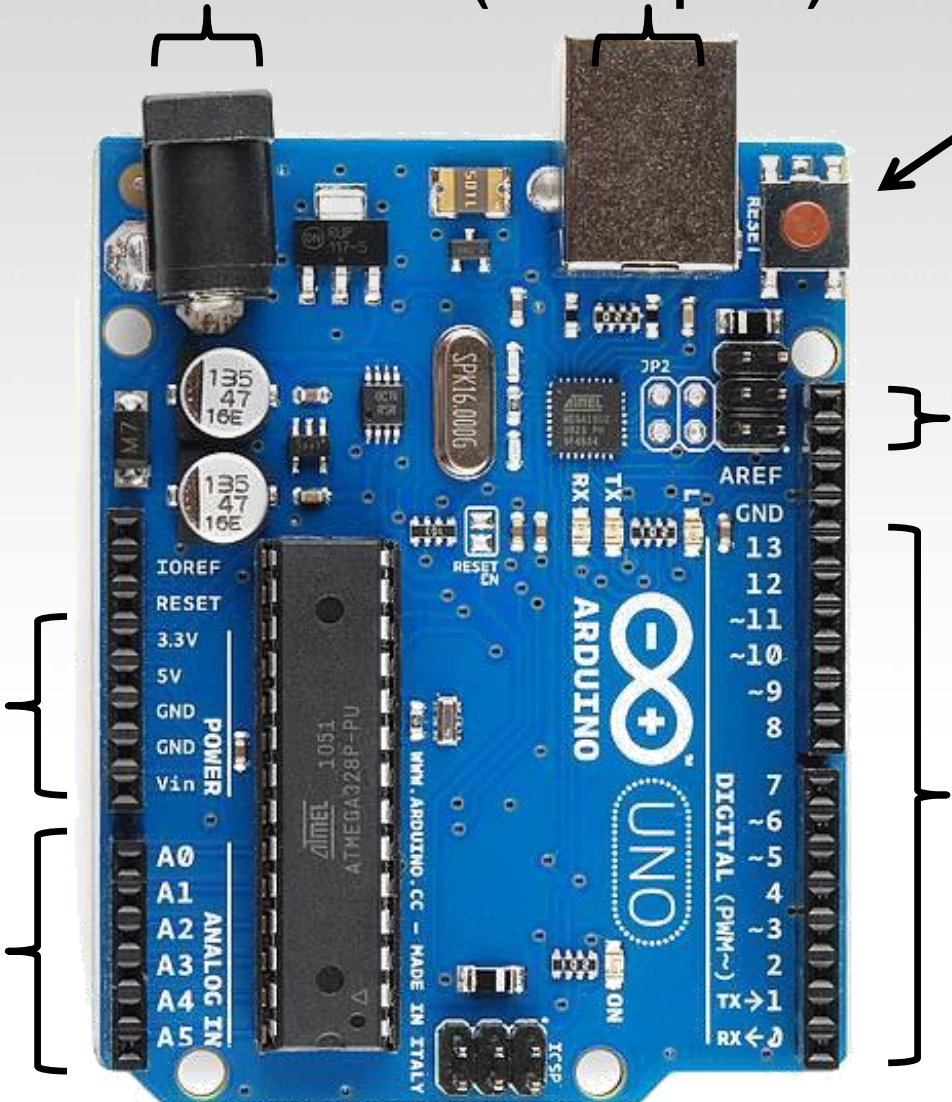
**PWR IN**

**USB  
(to Computer)**

**RESET**

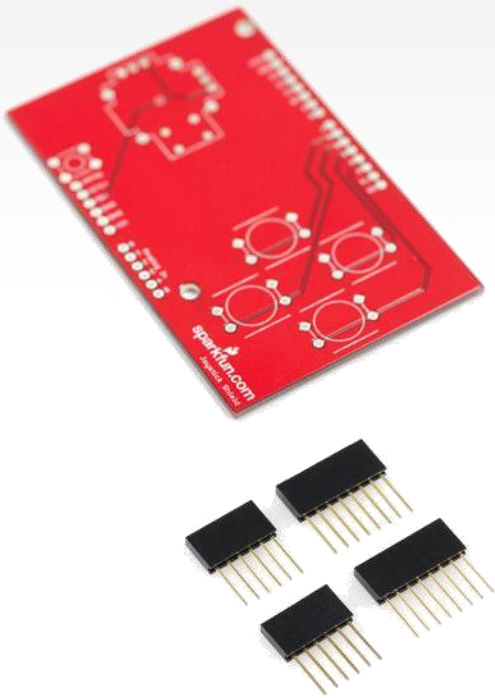
**SCL\SDA  
(I2C Bus)**

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



# Arduino Shields

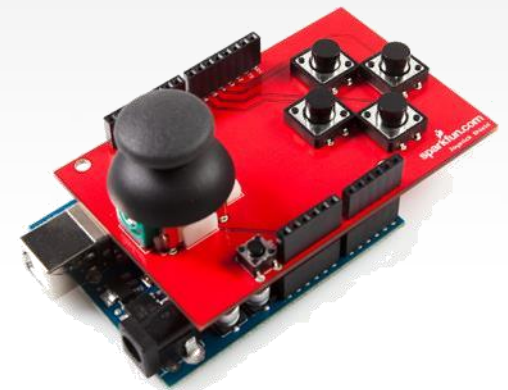
PCB



Built Shield

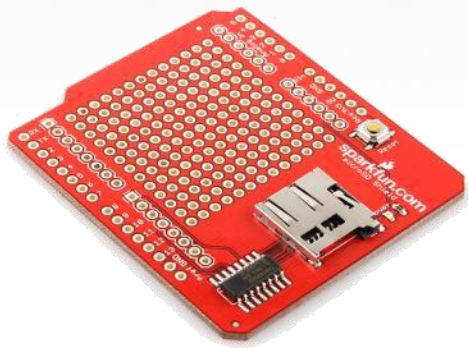


Inserted Shield

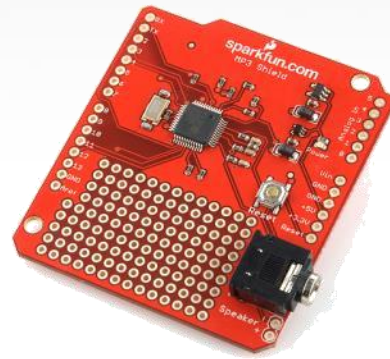


# Arduino Shields

Micro SD



MP3 Trigger











LCD

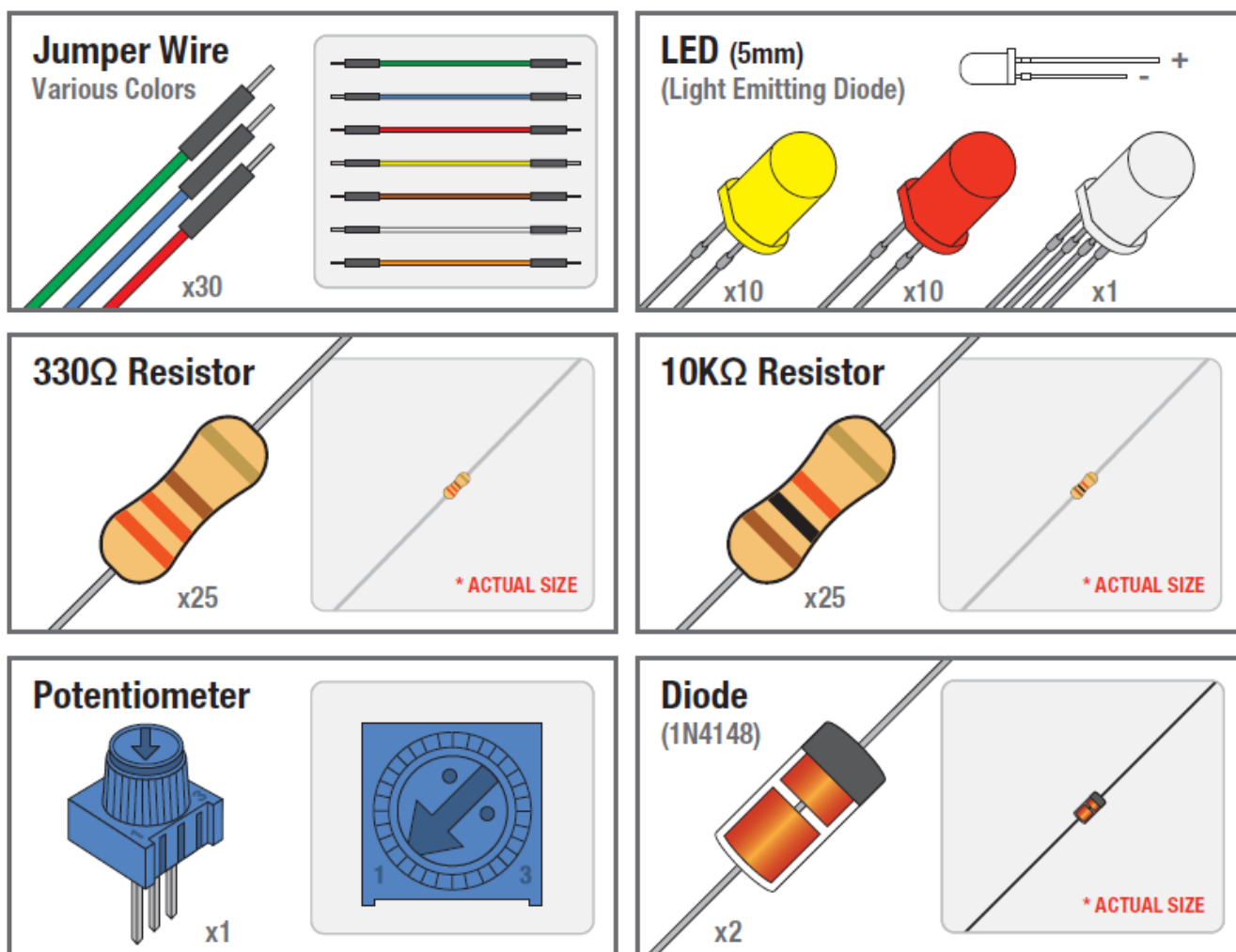




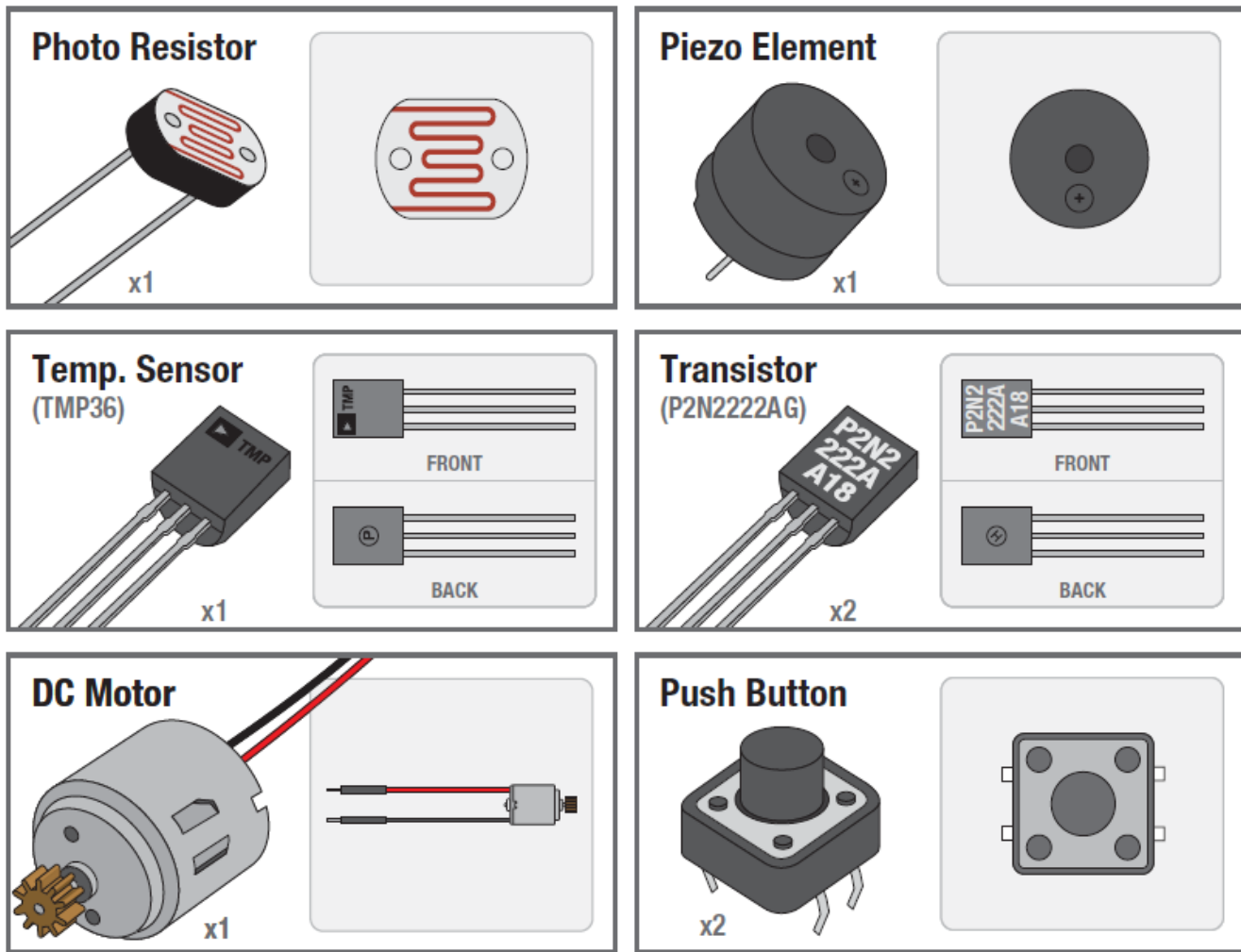
# Components

Name	Image	Type	Function	Notes
Push Button		Digital Input	Switch - Closes or opens circuit	Polarized, needs resistor
Trim potentiometer		Analog Input	Variable resistor	Also called a Trimpot.
Photoresistor		Analog Input	Light Dependent Resistor (LDR)	Resistance varies with light.
Relay		Digital Output	Switch driven by a small signal	Used to control larger voltages
Temp Sensor		Analog Input	Temp Dependent Resistor	
Flex Sensor		Analog Input	Variable resistor	
Soft Trimpot		Analog Input	Variable resistor	Careful of shorts
RGB LED		Dig & Analog Output	16,777,216 different colors	Ooh... So pretty.

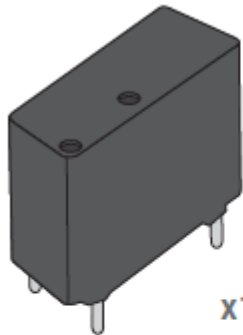
# Components



# Components

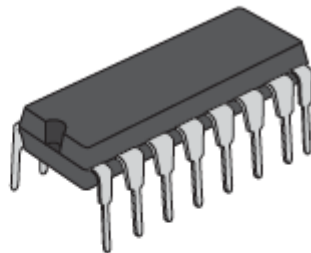


## Relay



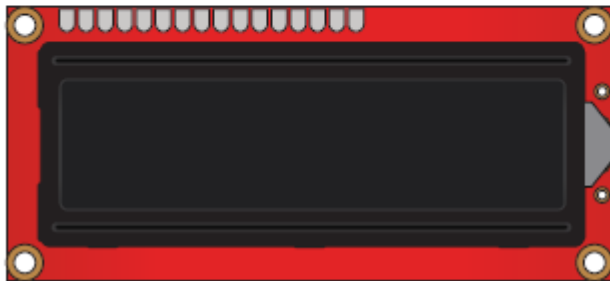
x1

## Integrated Circuit (IC)



x1

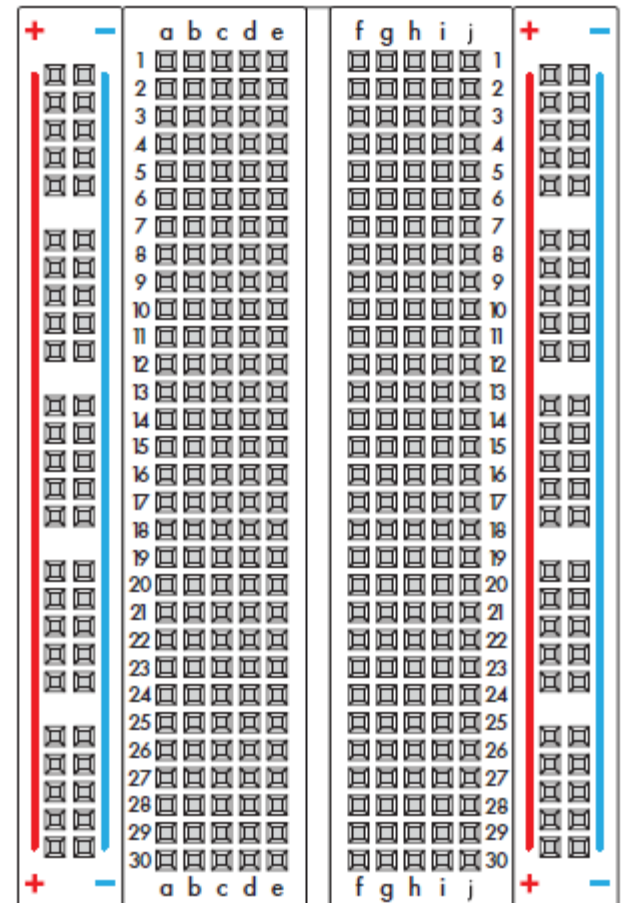
## LCD



x1

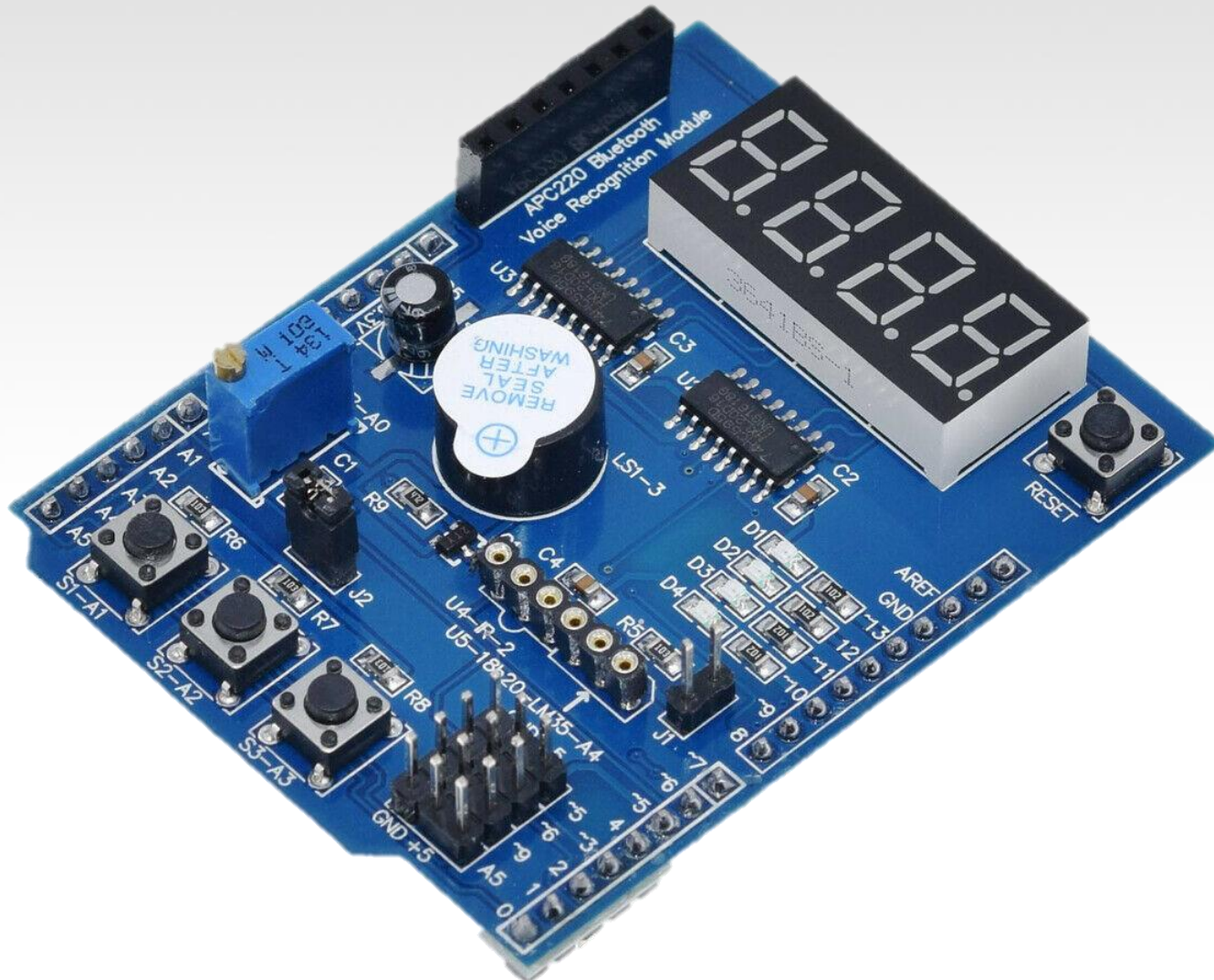
## Breadboard

Standard Solderless (Color may vary)



x1

Super cheap shield £4.50

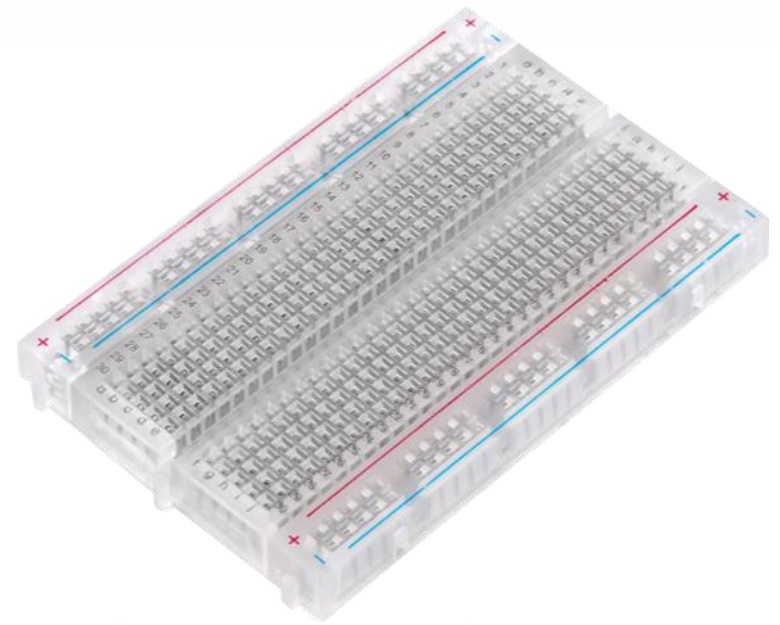


# Prototyping Circuits

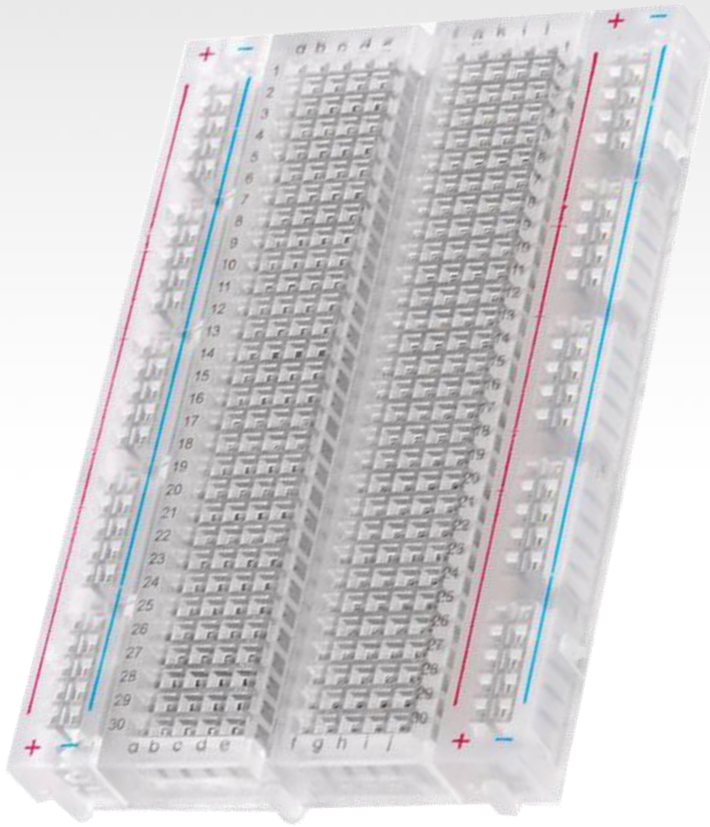
## Solderless Breadboard

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

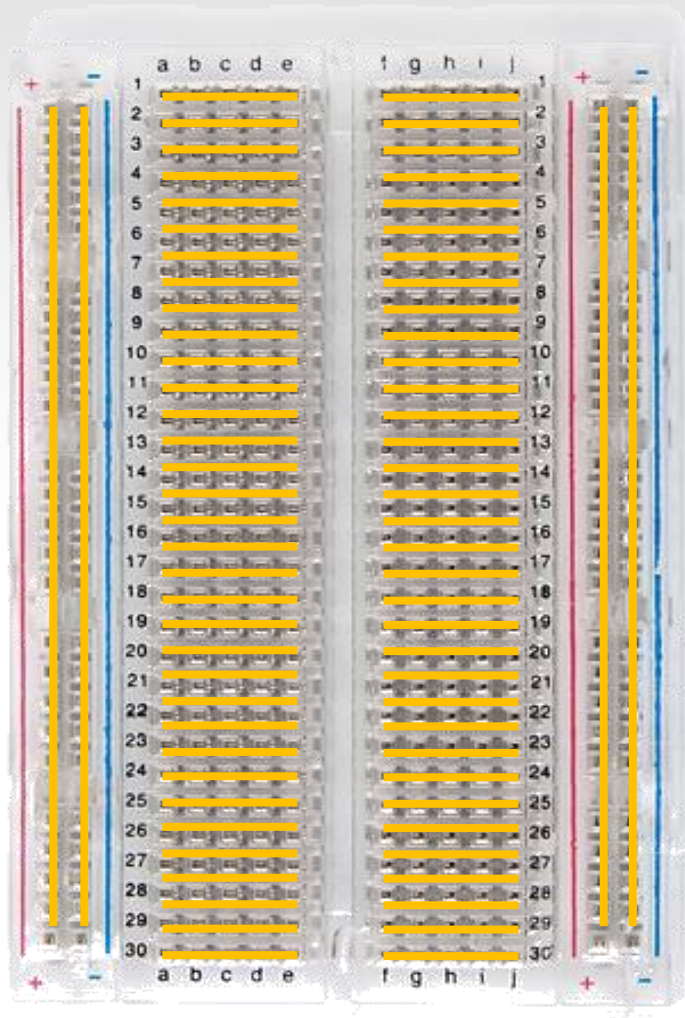
- A breadboard is easier than soldering
- A lot of those little holes are connected, which ones?
- Sometimes breadboards break



# What's a Breadboard?



# Solderless Breadboard

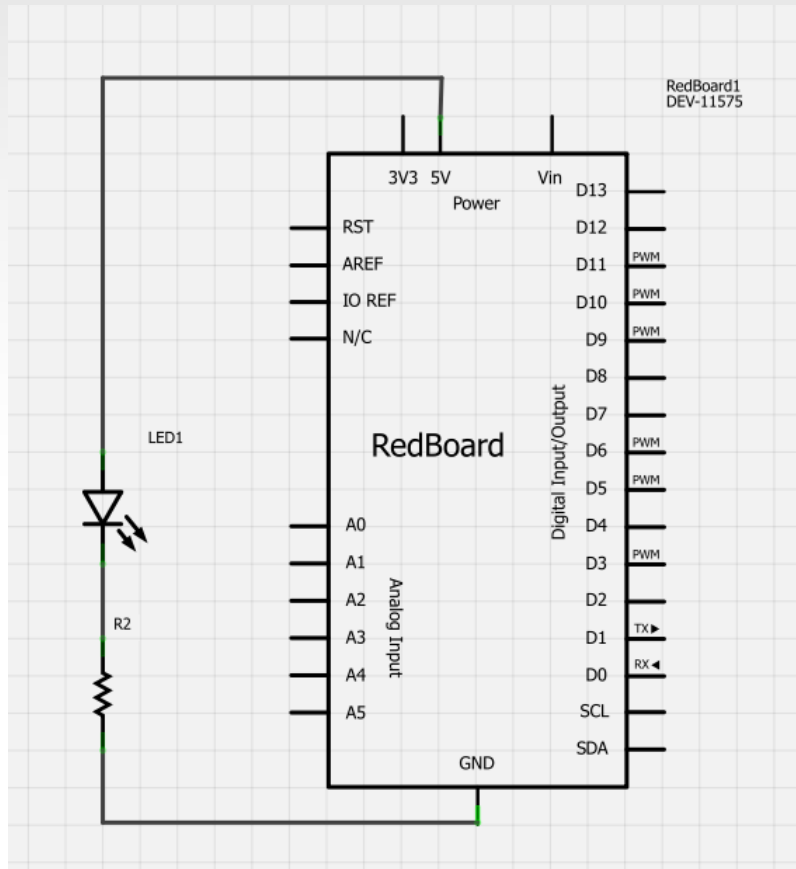


Each row (horiz.) of 5 holes are connected.

Vertical columns – called power bus are connected vertically



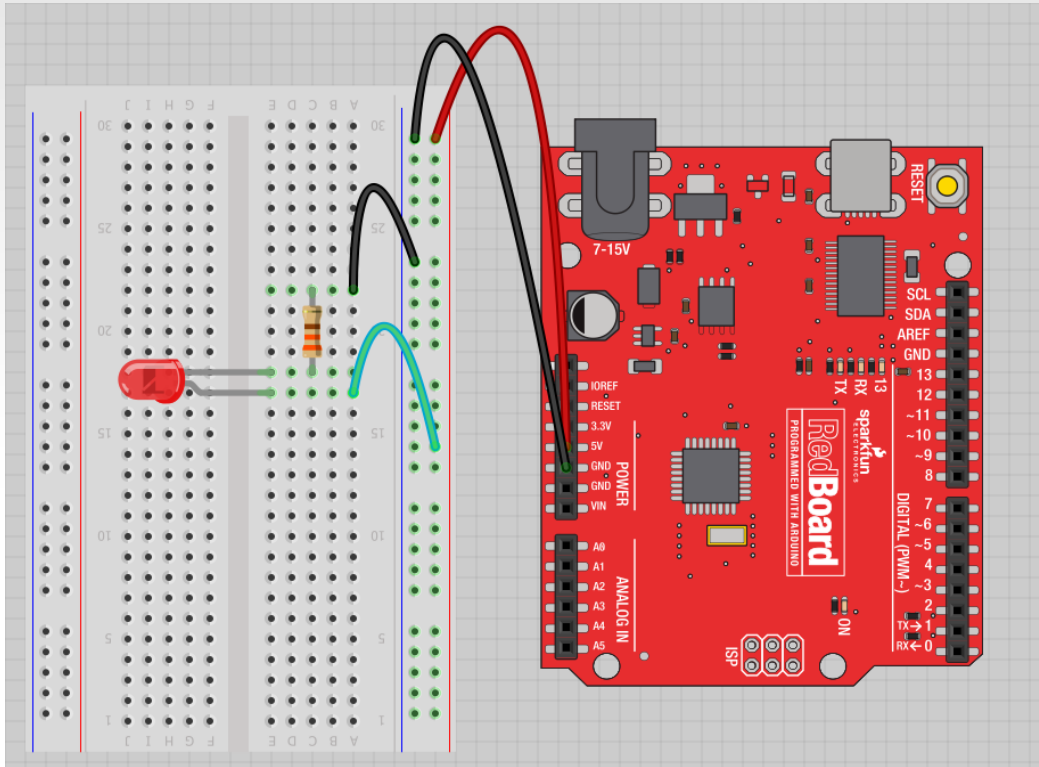
# Using the Breadboard to built a simple circuit



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

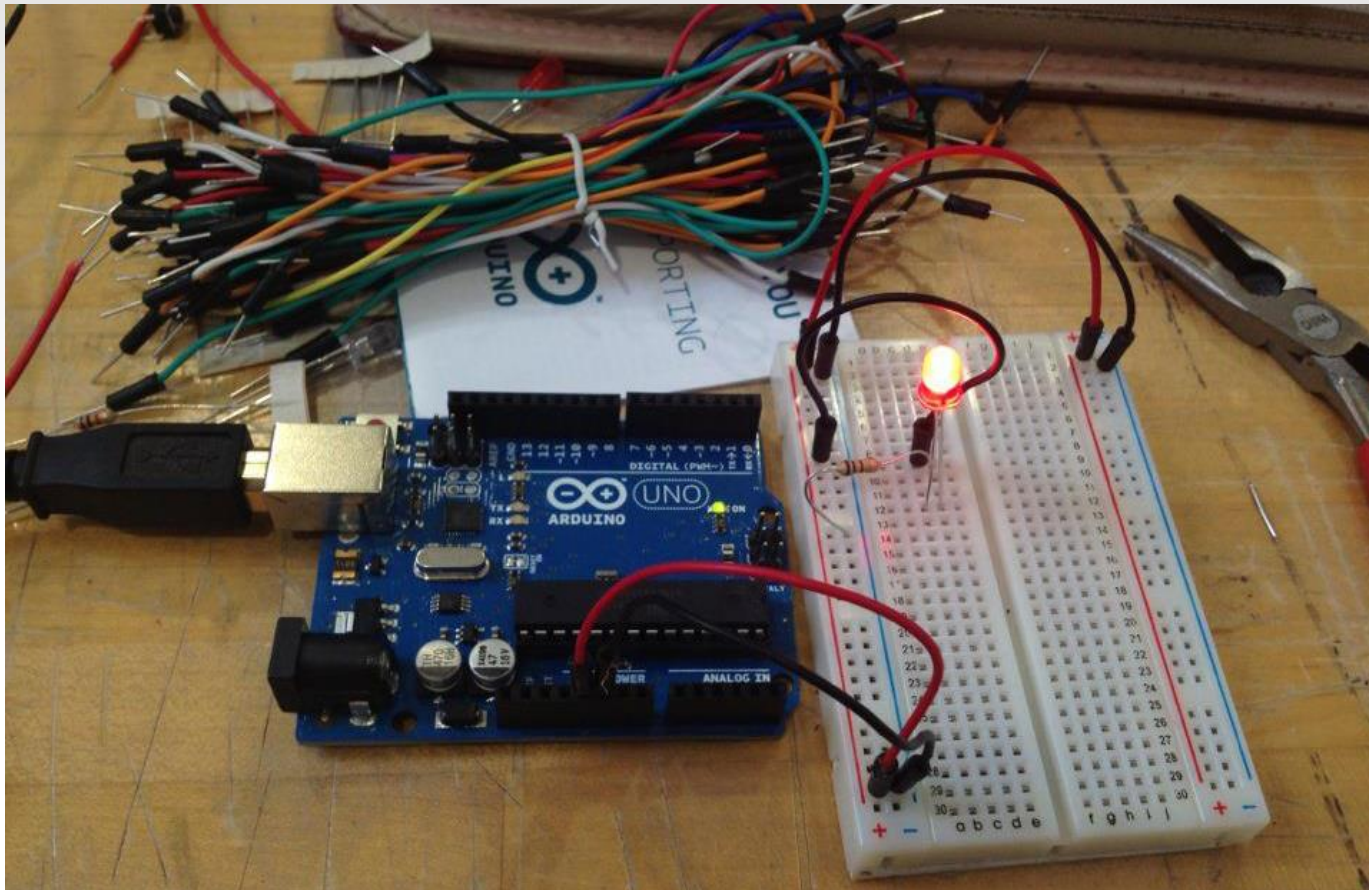
# Fritzing View of Breadboard Circuit



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.

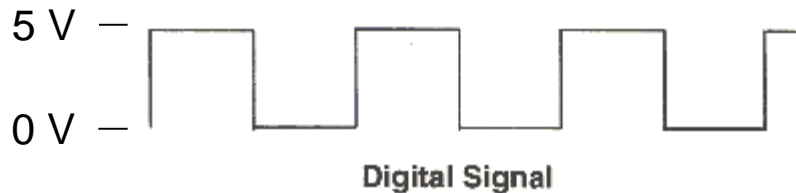
**Output** is any signal exiting the board.

<p><u>Examples</u>: Buttons Switches, Light Sensors, Flex Sensors, Humidity Sensors, Temperature Sensors...</p>	<p><u>Examples</u>: LEDs, DC motor, servo motor, a piezo buzzer, relay, an RGB LED</p>
---	--

# 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...



# Arduino

## Integrated Development Environment (IDE)

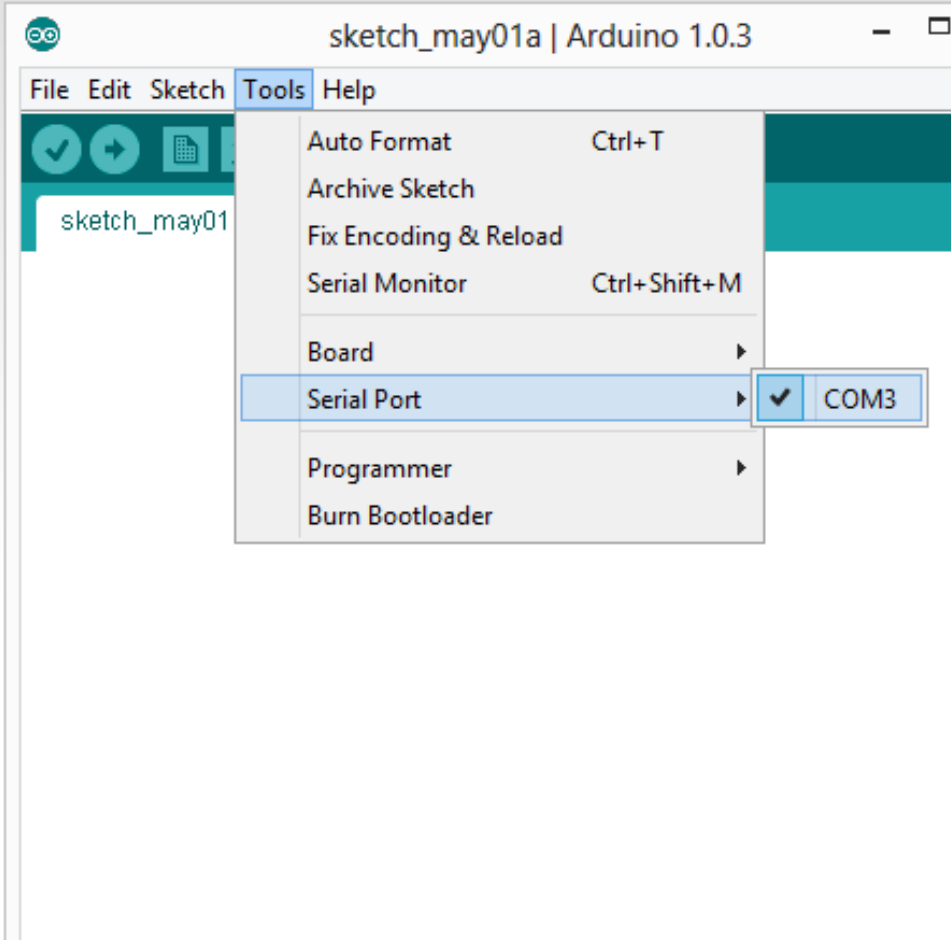


Two required functions /  
methods / routines:

```
void setup()  
{  
    // runs once  
}
```

```
void loop()  
{  
    // repeats  
}
```

# Settings: Tools → Serial Port

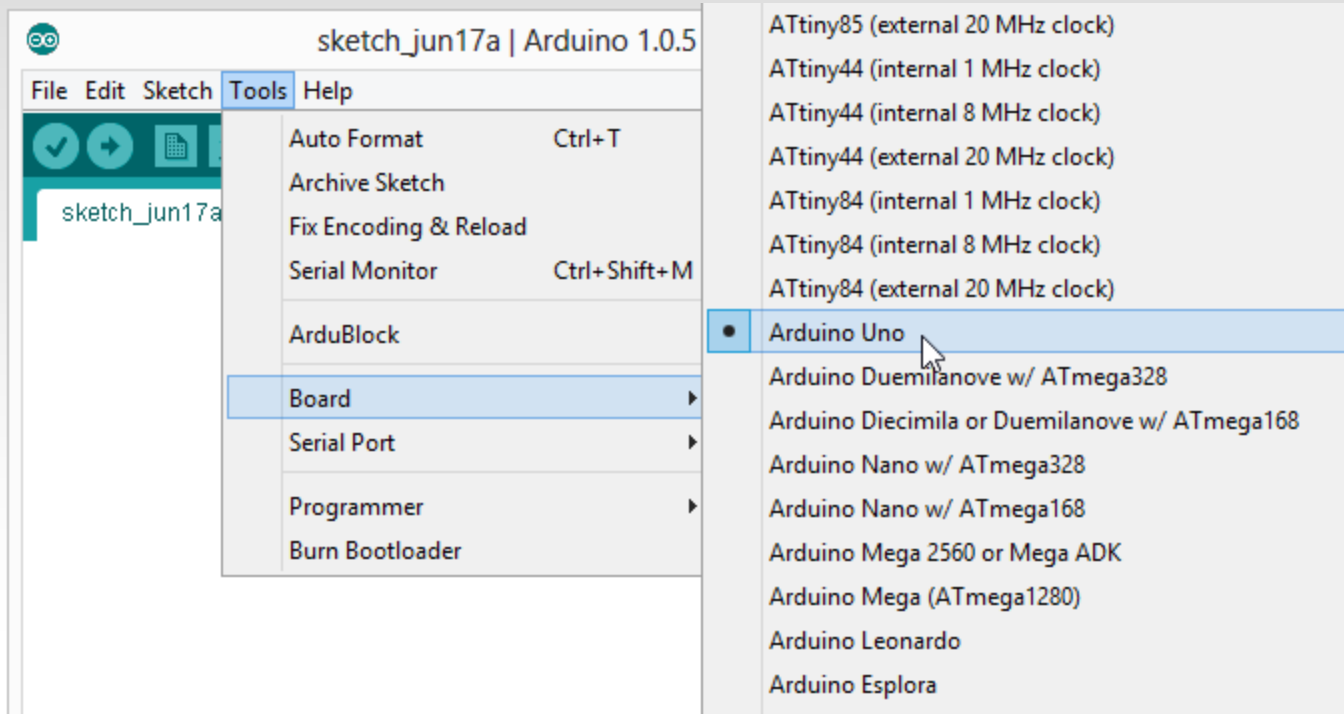


Your computer communicates to the Arduino microcontroller via a serial port → through a USB-Serial adapter.

Check to make sure that the drivers are properly installed.



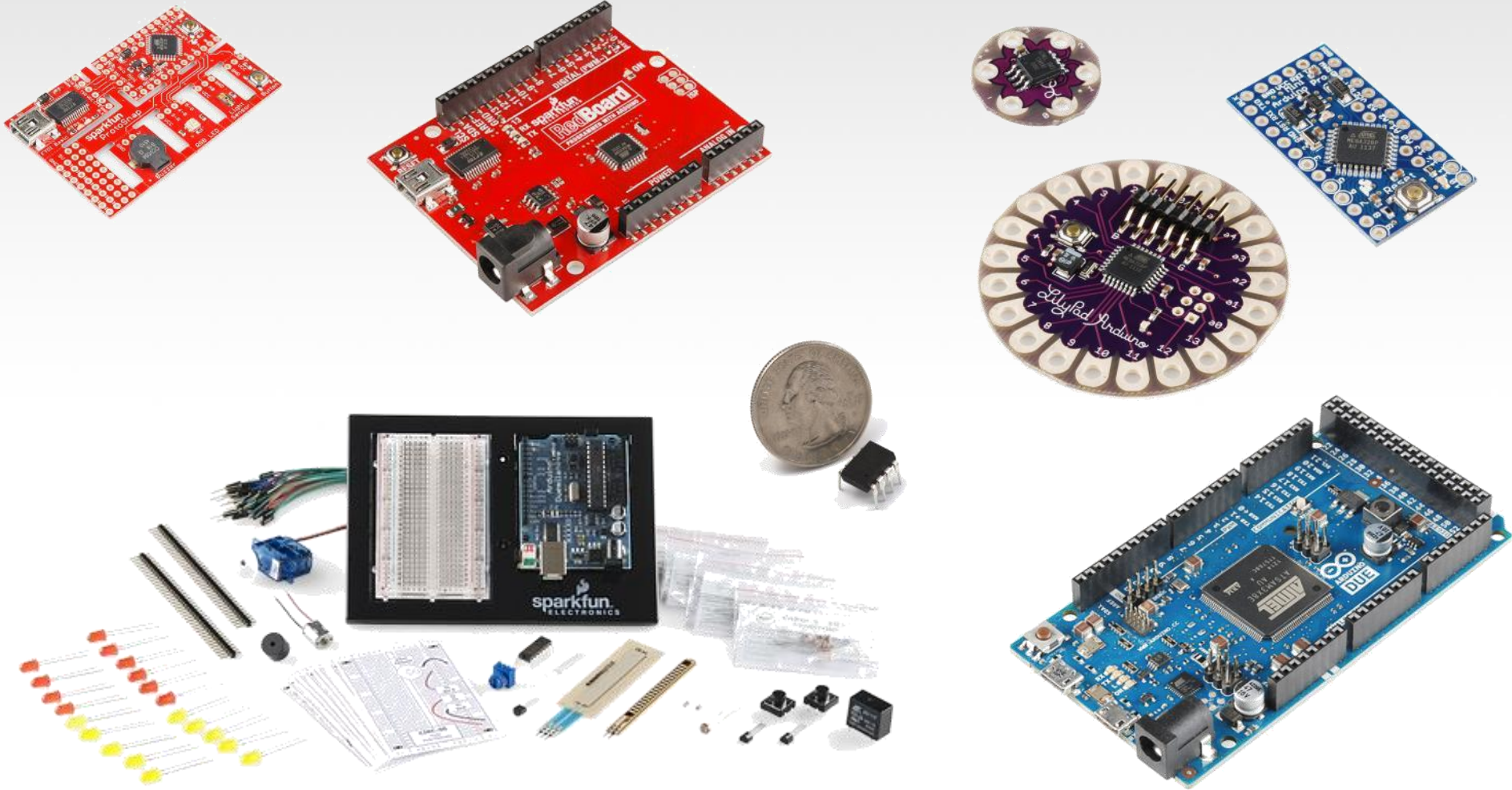
# Settings: Tools → Board



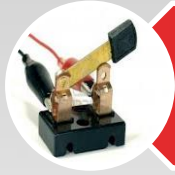
Next, double-check that the proper board is selected under the Tools→Board menu.



# Arduino & Arduino Compatible Boards



# BIG 6 CONCEPTS



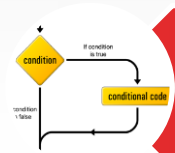
`digitalWrite()`



`analogWrite()`



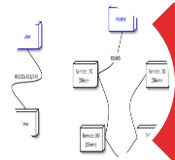
`digitalRead()`



`if()` statements / Boolean



`analogRead()`



Serial communication

Let's get to coding...

## Project #1 – Blink

“Hello World” of Physical Computing

*Pseudo-code – how should this work?*



# 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...
*/
```

```
BareMinimum | Arduino 1.0.5
File Edit Sketch Tools Help
BareMinimum $
// Name of sketch
// Brief Description
// Date:
//

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

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



**comments**

## Three commands to know...

```
pinMode (pin, INPUT/OUTPUT);
```

```
ex: pinMode (13, OUTPUT);
```

```
digitalWrite (pin, HIGH/LOW);
```

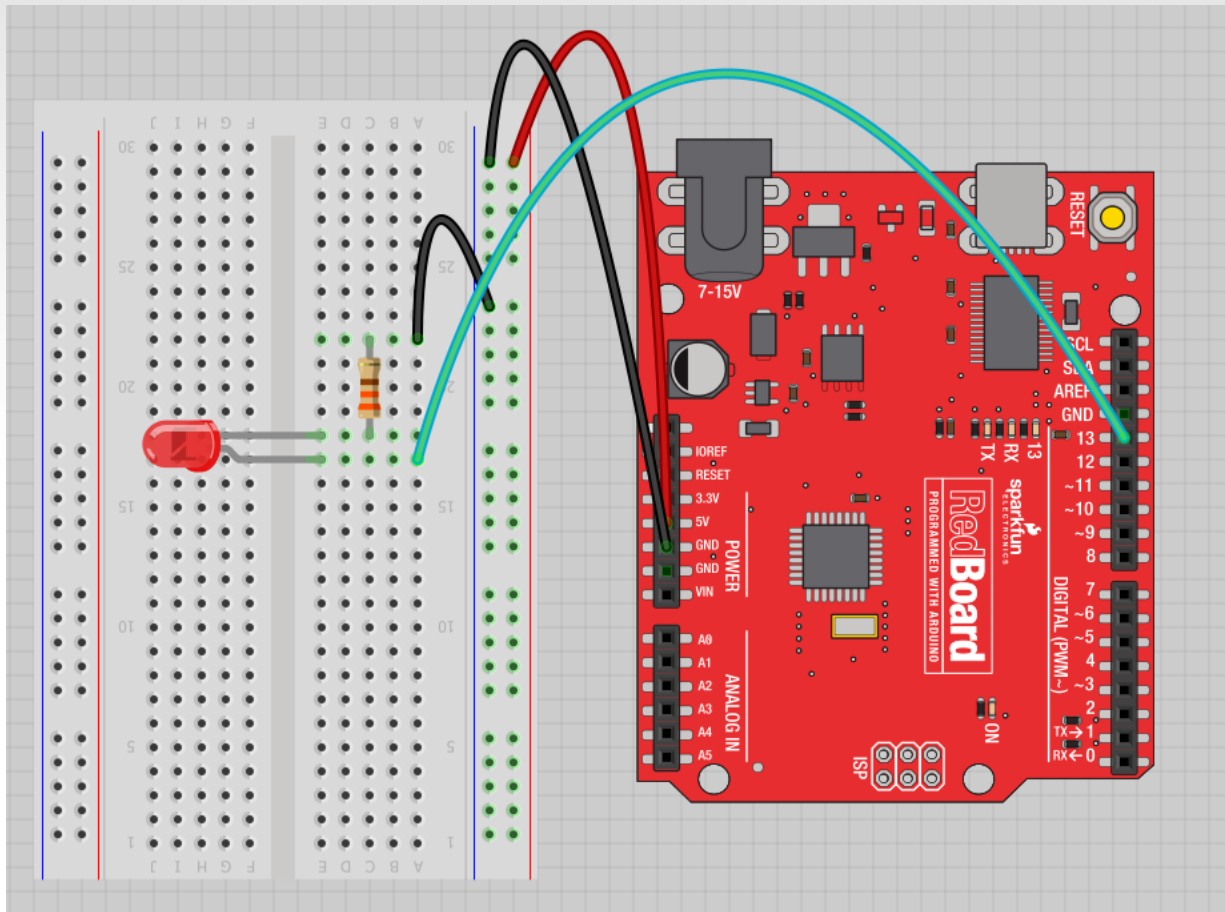
```
ex: digitalWrite (13, HIGH);
```

```
delay (time_ms);
```

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

```
// NOTE: -> commands are CASE-sensitive
```

# Project #1: Wiring Diagram



Move the green wire from the power bus to pin 13 (or any other Digital I/O pin on the Arduino board).

Image created in Fritzing



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 it's own  $330\Omega$  resistor.)

Generate your own morse code flashing

How about → Knight Rider? Disco? Police Light?

# Programming Concepts: Variables

```
ProtosnapProMiniExample2$
```

```
// Comments go here
// Written by:  Joesephine Jones
// Date:  April 12, 2013

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

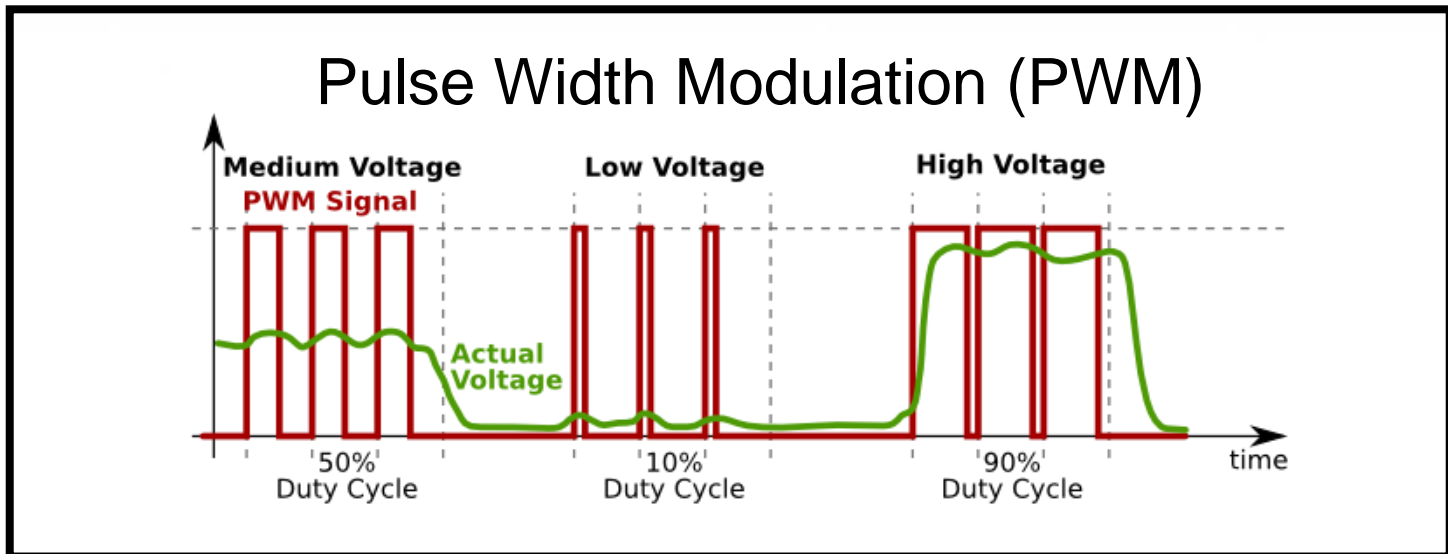
# Fading in and Fading Out (Analog or Digital?)

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.



# Project #2 – Fading

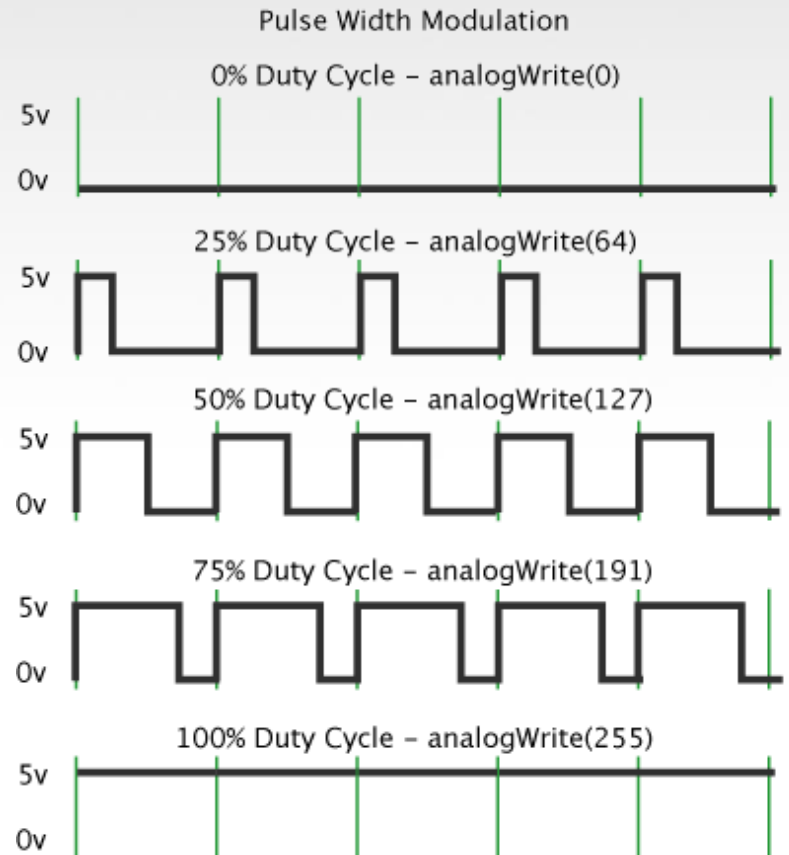
## Introducing a new command...

```
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).

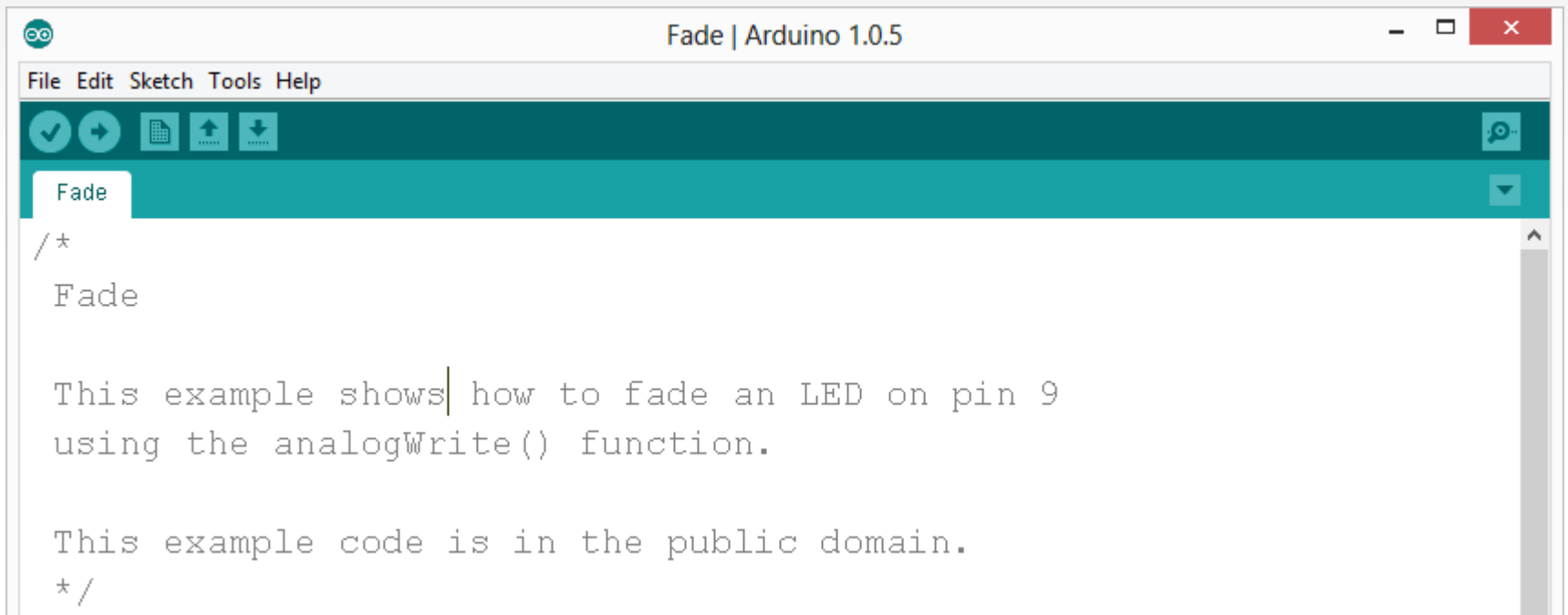
0 => 0V    |    255 => 5V



# Move one of your LED pins over to Pin 9

In Arduino, open up:

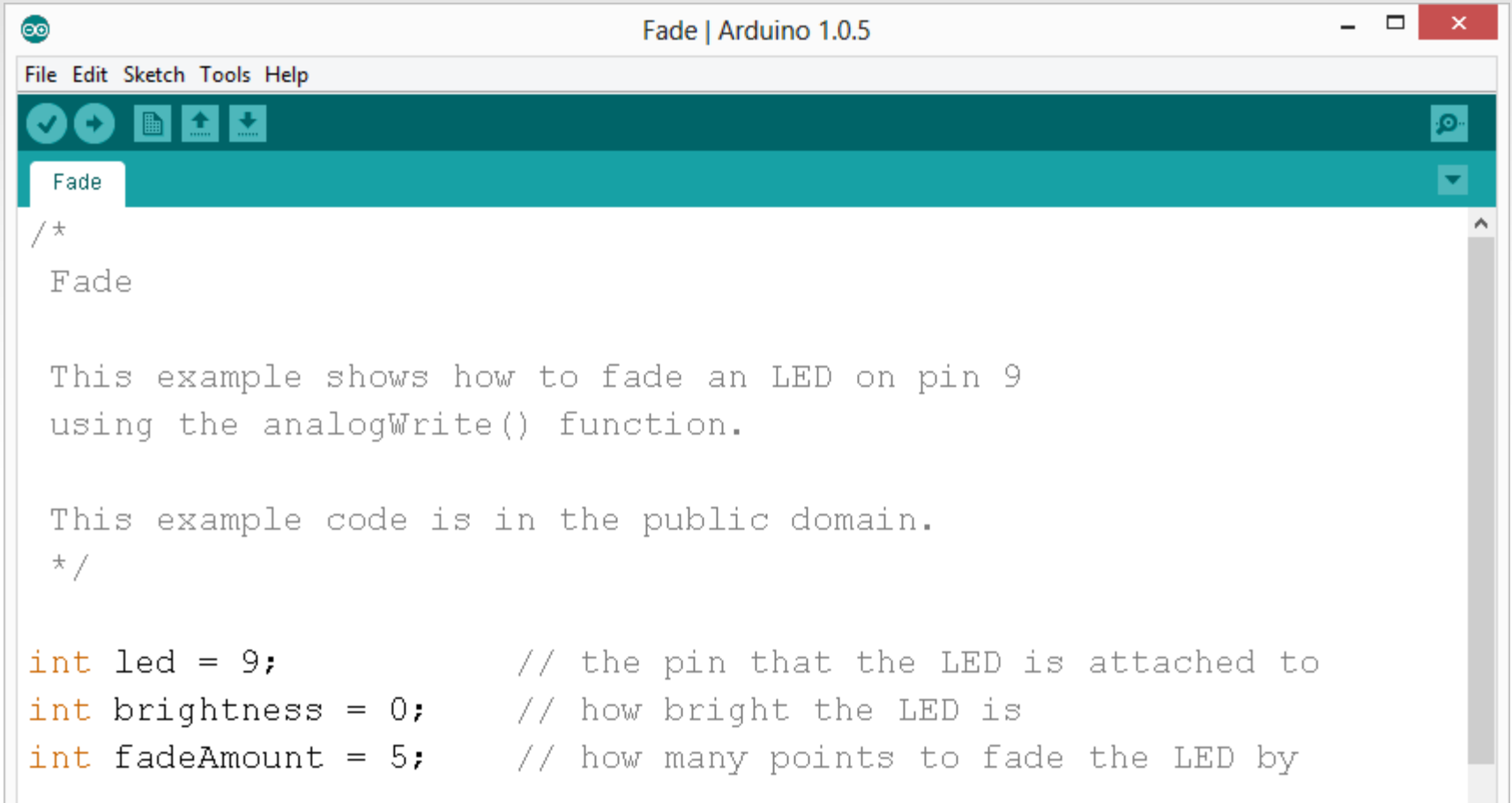
File → Examples → 01.Basics → Fade

A screenshot of the Arduino IDE interface. The window title is "Fade | Arduino 1.0.5". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for a checkmark, a right arrow, a document, an upload arrow, and a download arrow. A tab labeled "Fade" is active. The main text area contains the following code:

```
/*  
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 - Code Review



```
File Edit Sketch Tools Help
Fade
/*
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
```

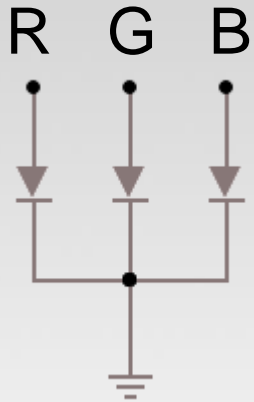
# Fade - Code Review

```
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);
}
```

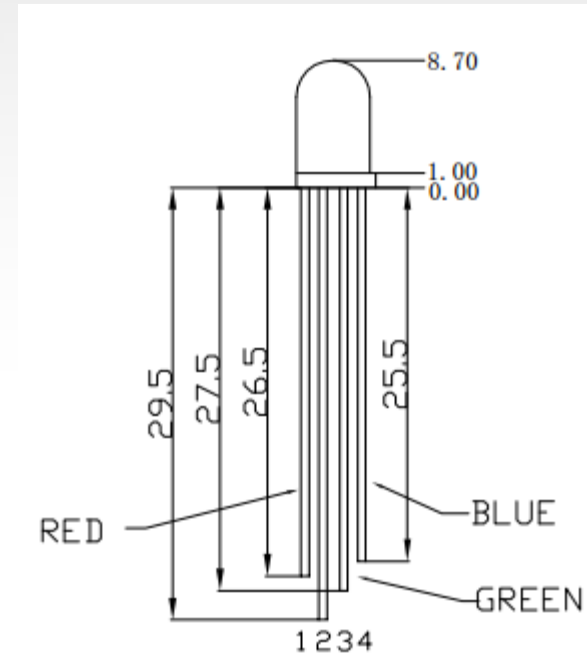


## Color Mixing Tri-color LED

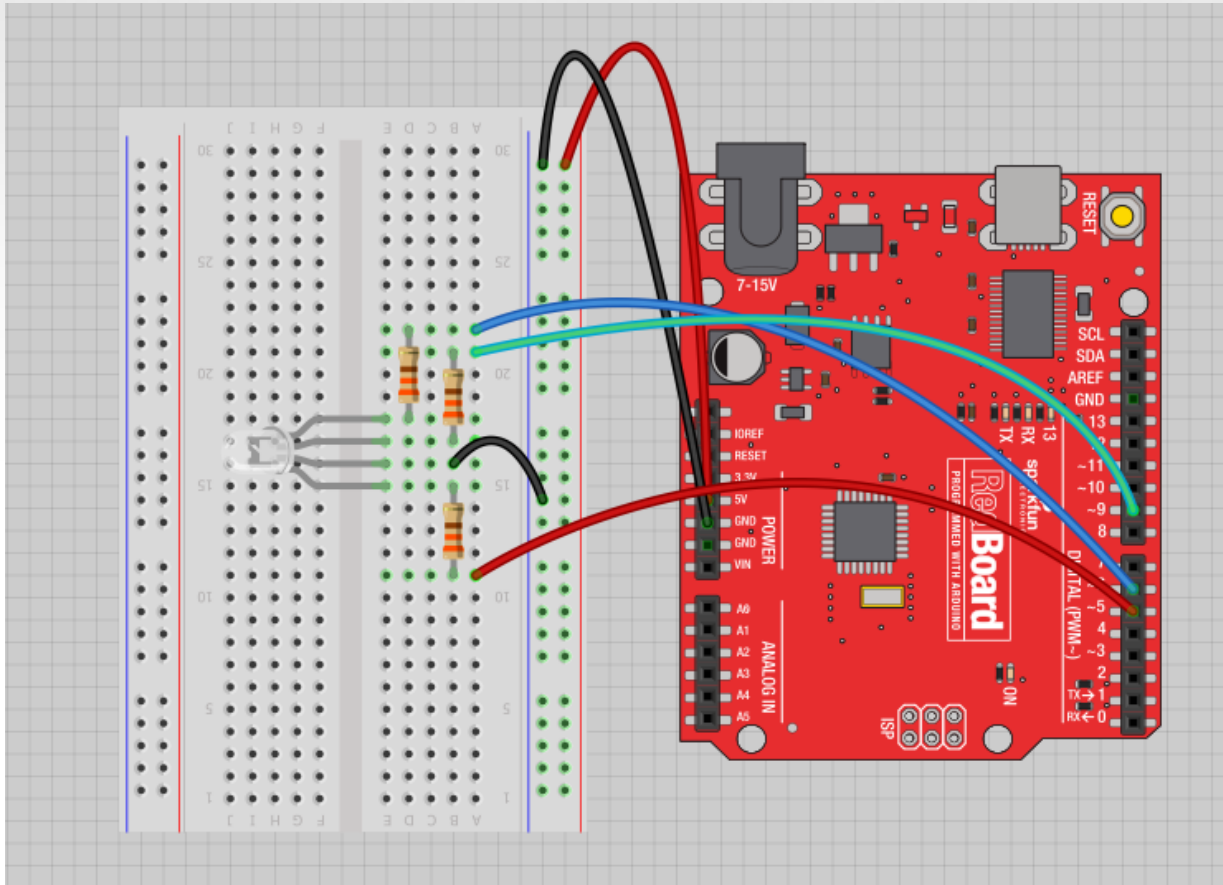
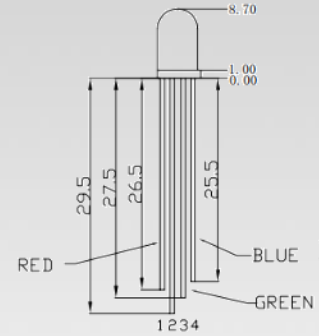


This a standard – Common  
Cathode LED

This means the negative side of  
the LEDs are all tied to Ground.



# Project 3 – RGB LED

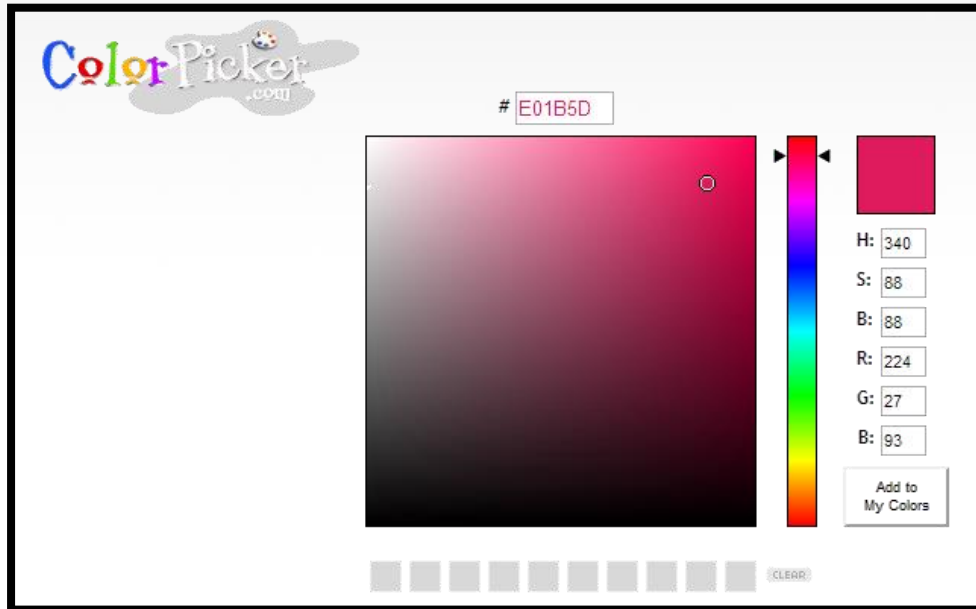


Note: The longest leg of the RGB LED is the Common Cathode. This goes to GND.

Use pins 5, 6, & 9

# How many unique colors can you create?

$$\begin{aligned}\# \text{ of unique colors} &= 256 \cdot 256 \cdot 256 \\ &= 16,777,216 \text{ colors!}\end{aligned}$$



Use Colorpicker.com or experiment on your own.

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

Play around with this with the `analogWrite()` command.

# RGB LED Color Mixing

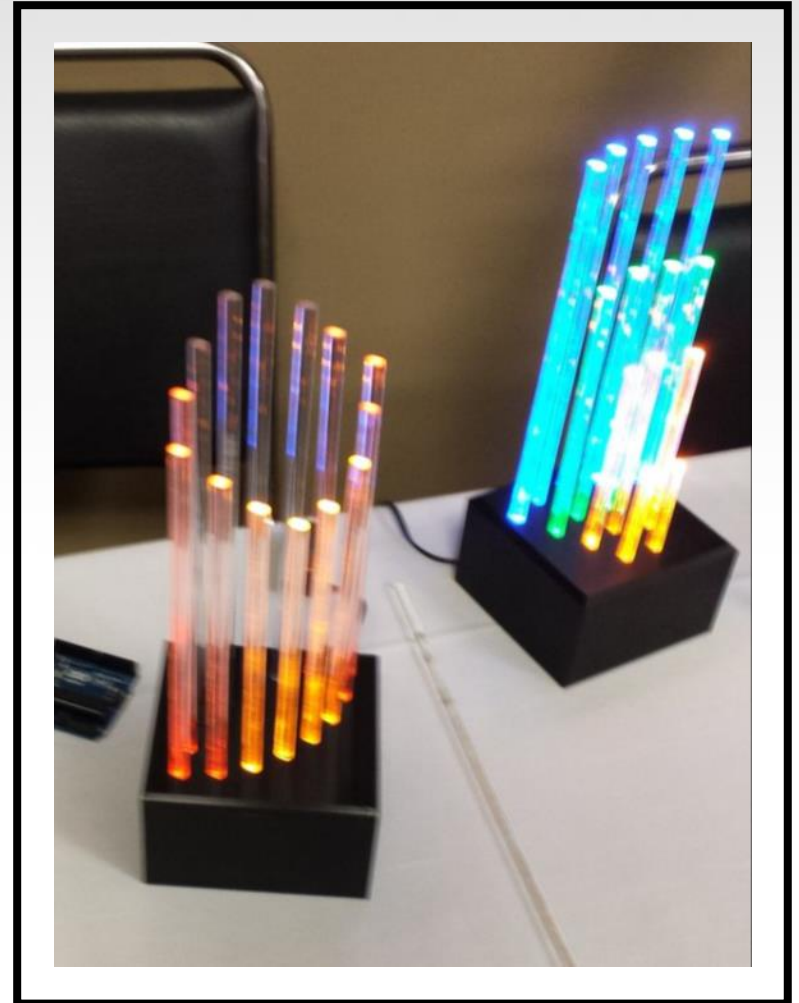
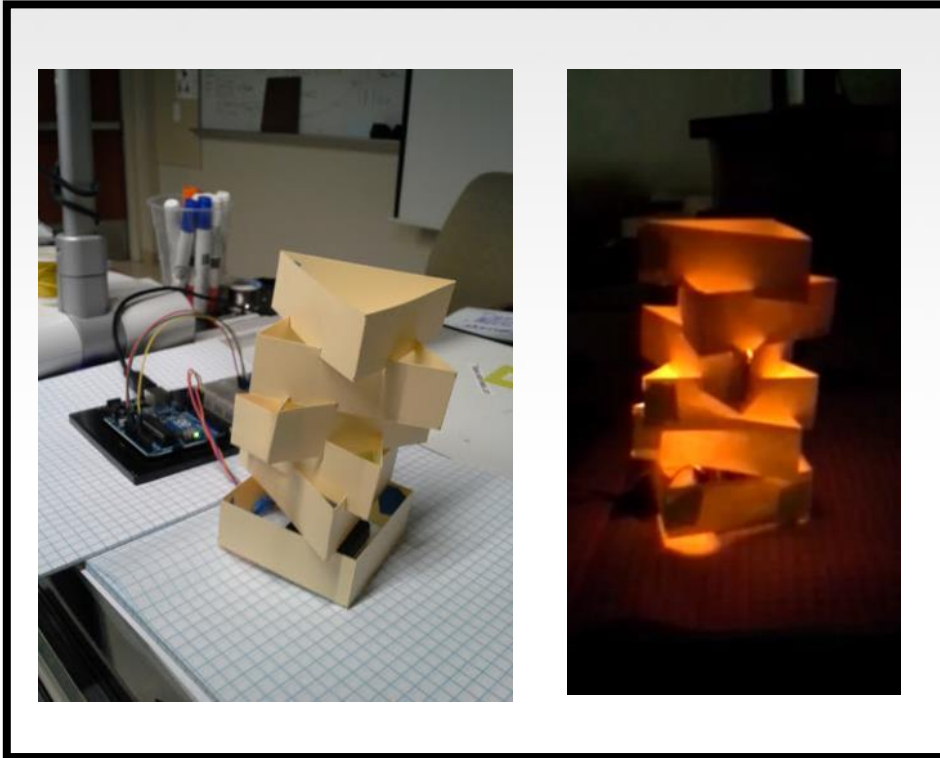
```
int redPin = 5;
int greenPin = 6;
int bluePin = 9;

void setup()
{
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}
```

# RGB LED Color Mixing

```
void loop()  
{  
  analogWrite(redPin, 255);  
  analogWrite (greenPin, 255);  
  analogWrite (bluePin, 255);  
}
```

# Project: Mood Lamp / Light Sculpture







# 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

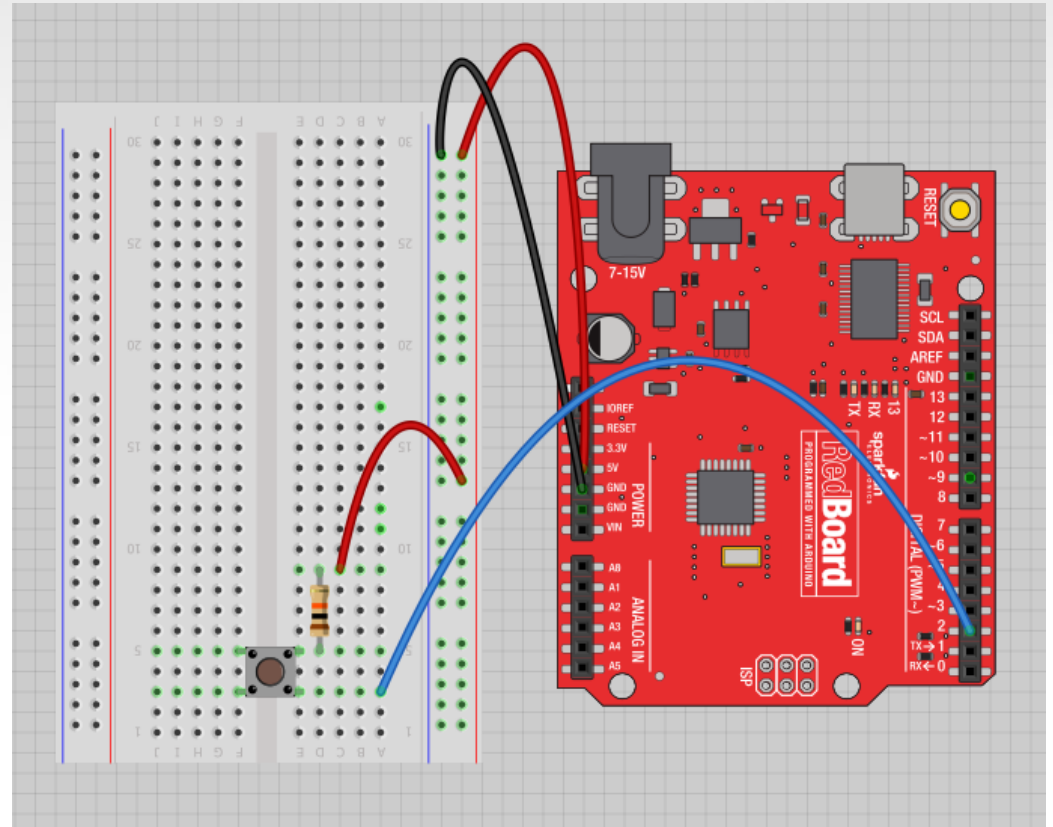
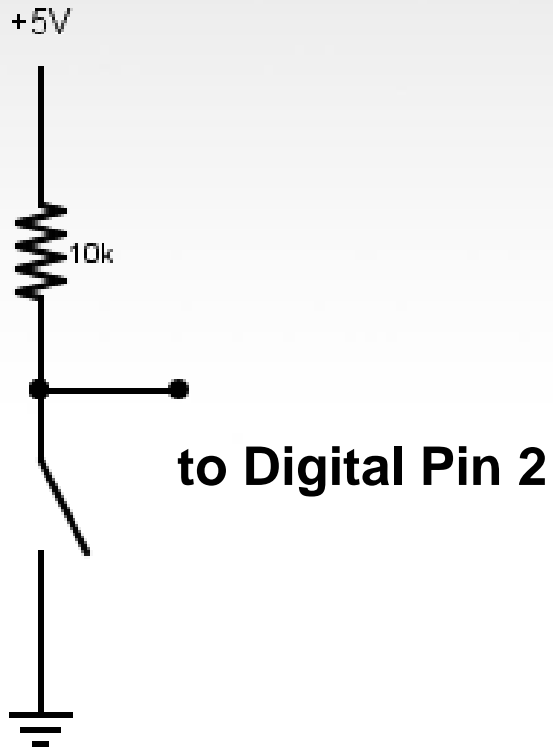


# Project #4 – Digital Input

In Arduino, open up:

File → Examples → 02.Digital → Button

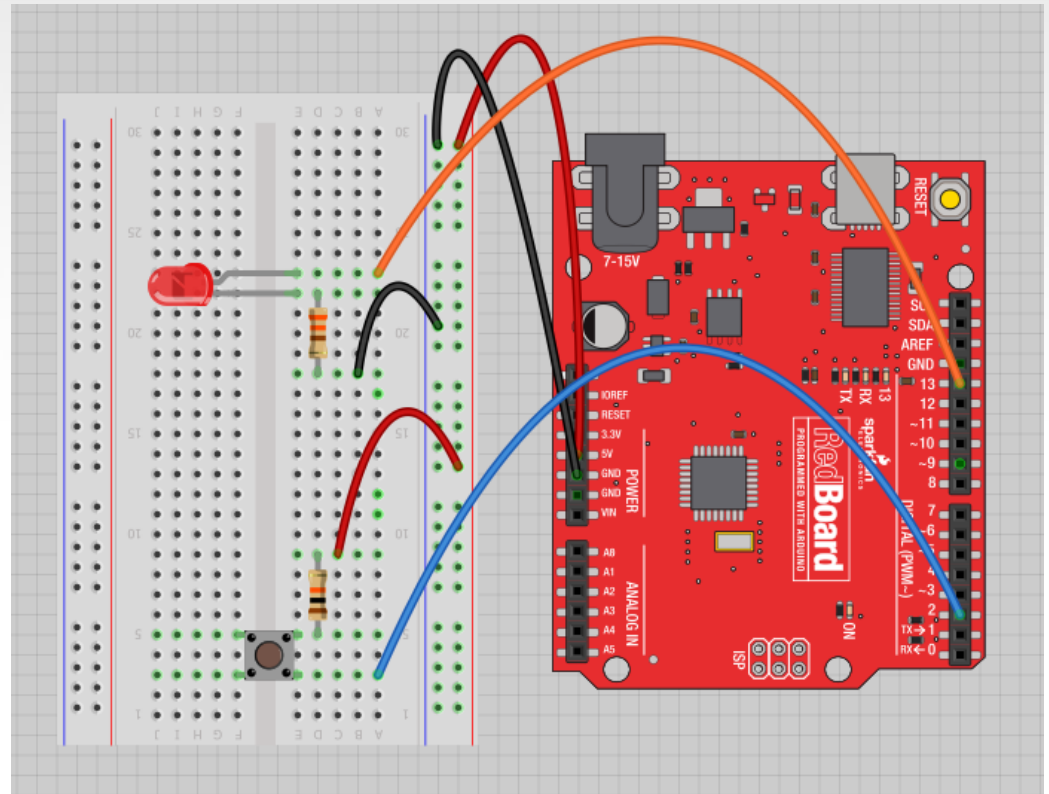
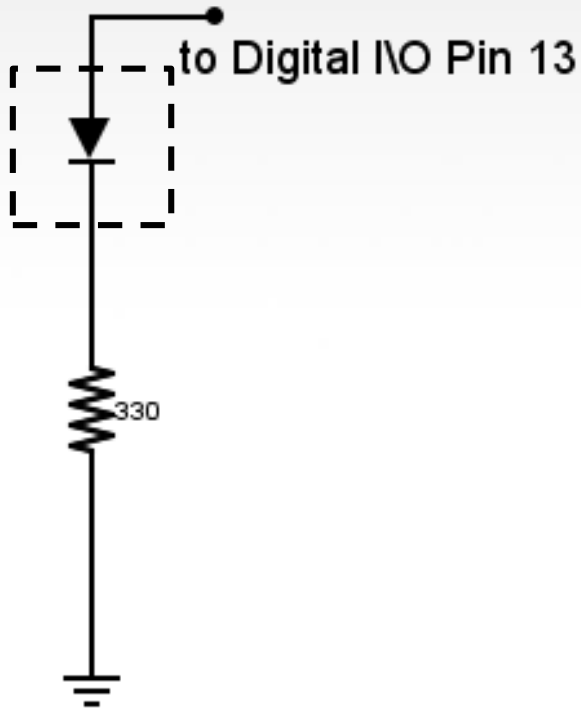
# Digital Sensors (a.k.a. Switches) Pull-up Resistor (circuit)



# Digital Sensors (a.k.a. Switches)

## Add an indicator LED to Pin 13

This is just like our  
1<sup>st</sup> 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 straight forward 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 set it equal to the function `digitalRead(pushButton)`

We declare a variable as an integer.

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`

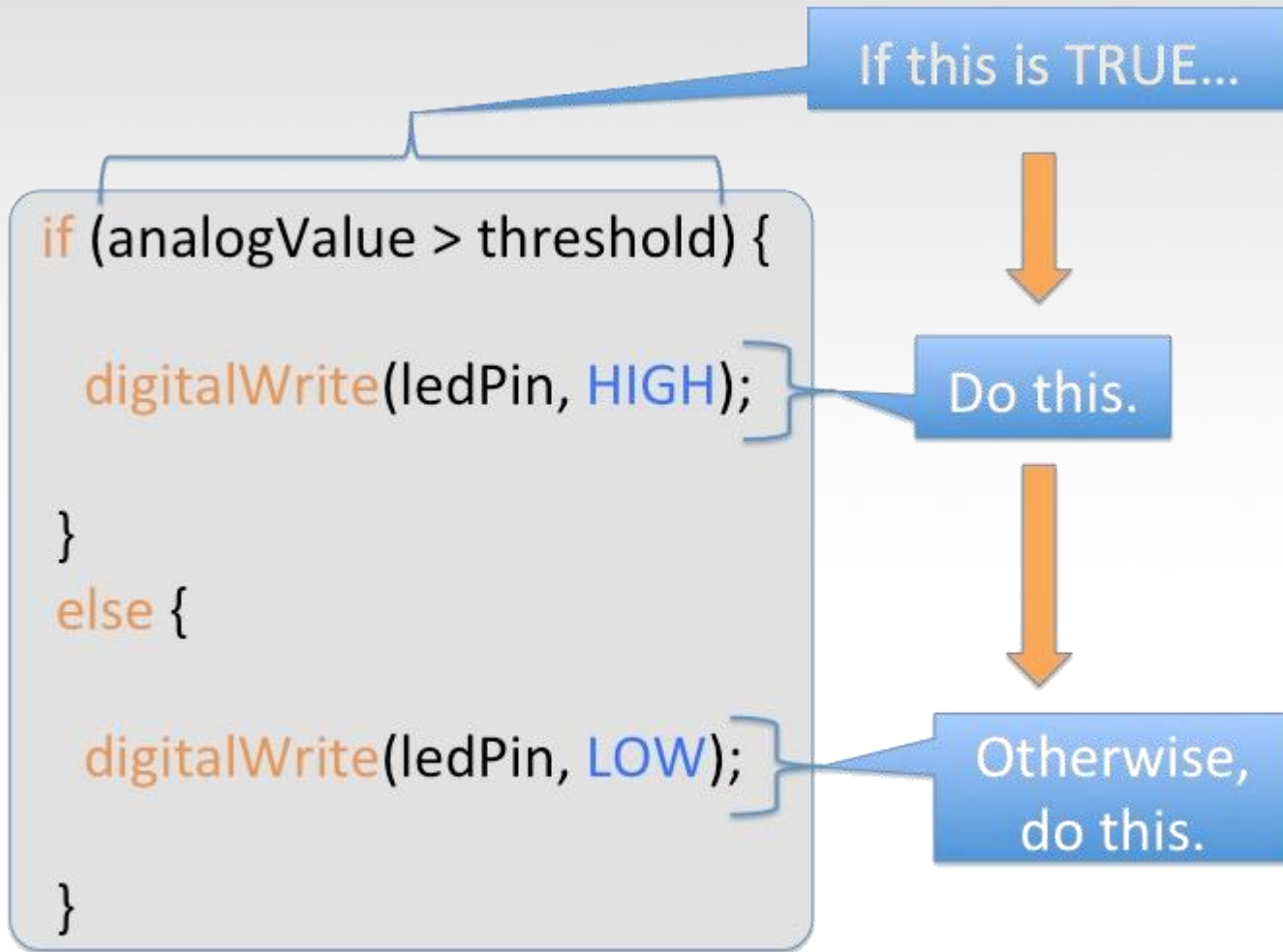
Recall that the `pushButton` variable stores the number 2

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



# Programming: Conditional Statements

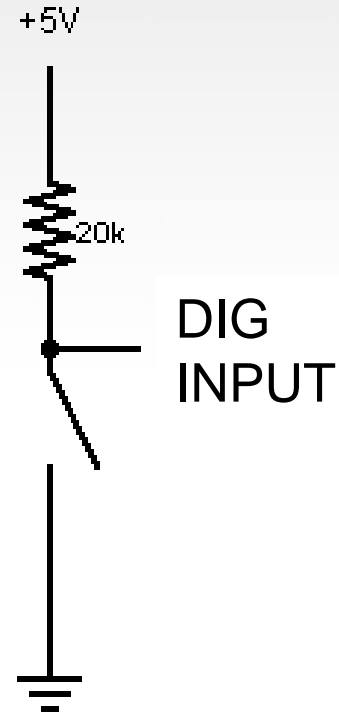
`if ()`



# Programming: Conditional Statements

`if ()`

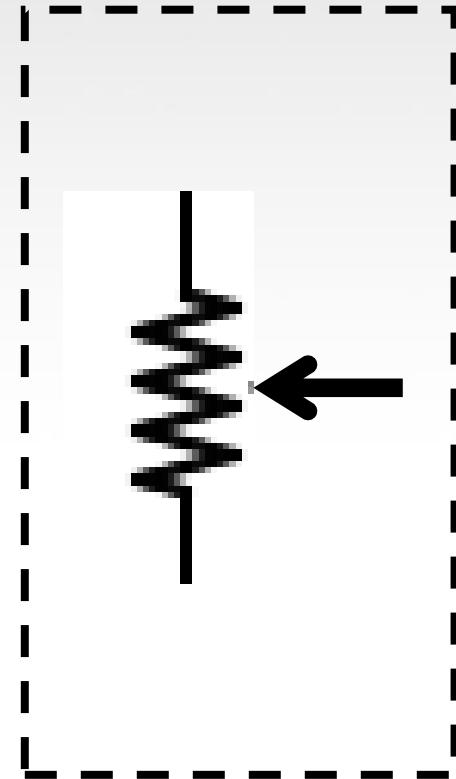
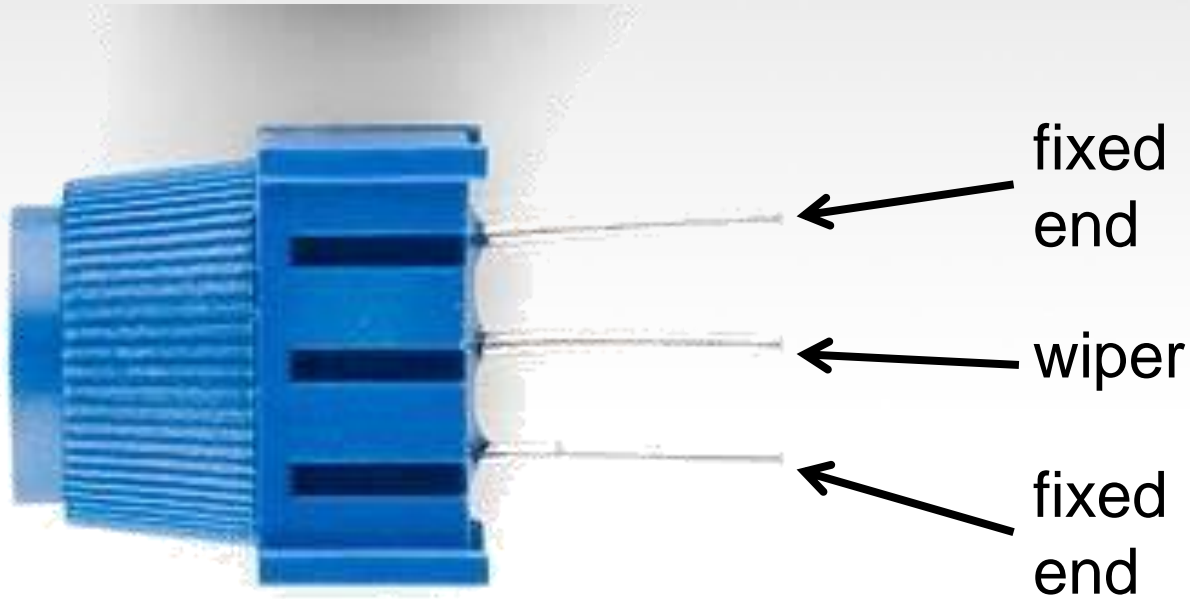
```
void loop()  
{  
  int buttonState = digitalRead(5);  
  if(buttonState == LOW)  
  { // do something  
  }  
  else  
  { // do something else  
  }  
}
```



# Boolean Operators

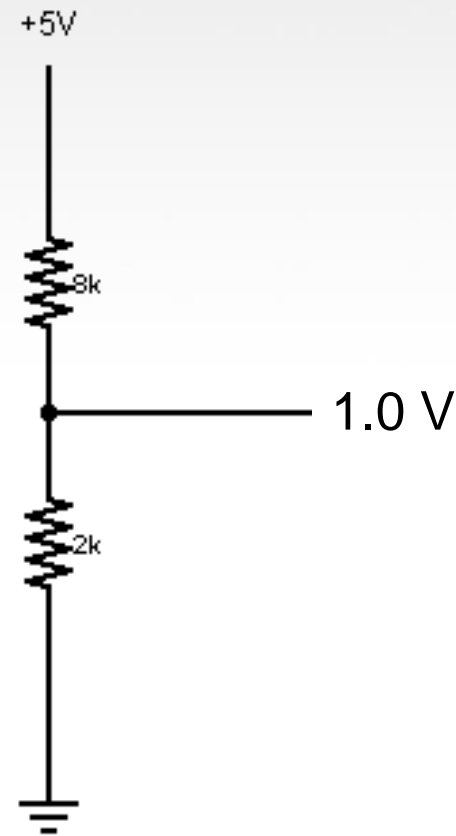
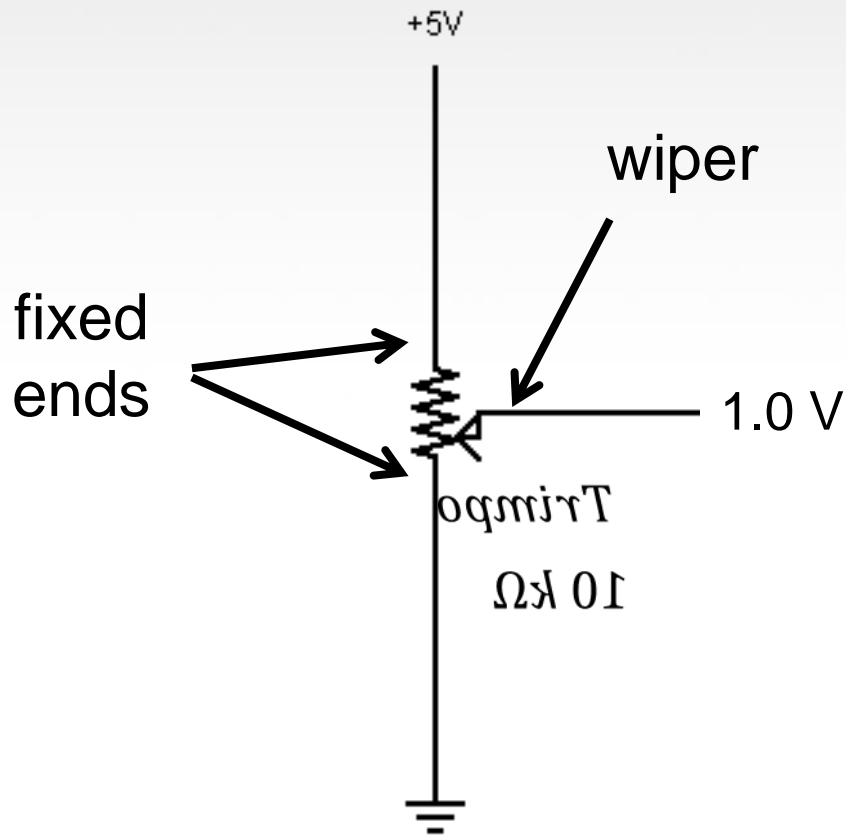
<b>&lt;Boolean&gt;</b>	<b>Description</b>
( ) == ( )	is equal?
( ) != ( )	is not equal?
( ) > ( )	greater than
( ) >= ( )	greater than or equal
( ) < ( )	less than
( ) <= ( )	less than or equal

# Trimpot (Potentiometer) Variable Resistor



# Analog Sensors

3 Pin Potentiometer = var. resistor (circuit)  
*a.k.a. Voltage Divider Circuit*



# analogRead()

Arduino uses a 10-bit A/D Converter:

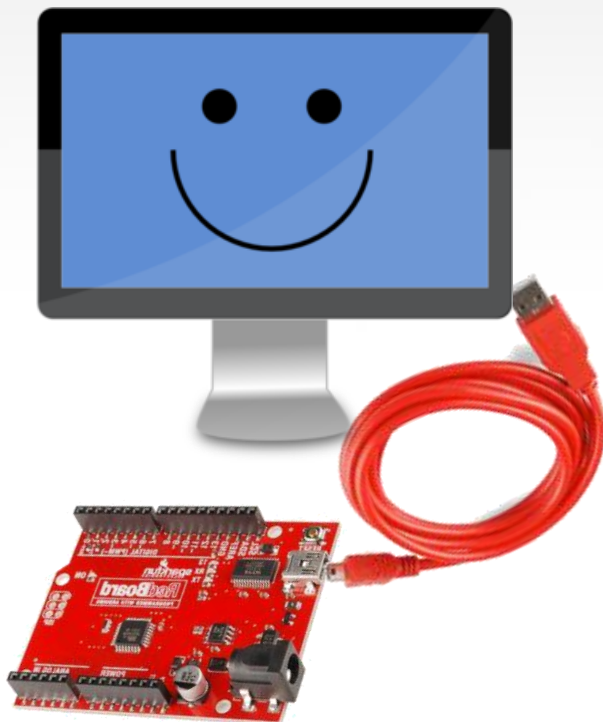
- this means that you get input values from 0 to 1023
  - $0\text{ V} \rightarrow 0$
  - $5\text{ V} \rightarrow 1023$

Ex:

```
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.

1 0 0 1 0 1 0 0 1 1 0 ...



Arduino dedicates Digital I/O pin # 0 to receiving and Digital I/O pin #1 to transmit.

# Serial Monitor & analogRead()

```
sketch_apr02a | Arduino 1.0.3
File Edit Sketch Tools Help
sketch_apr02a $
// analogRead() & Serial.print()
//
//
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
}
```

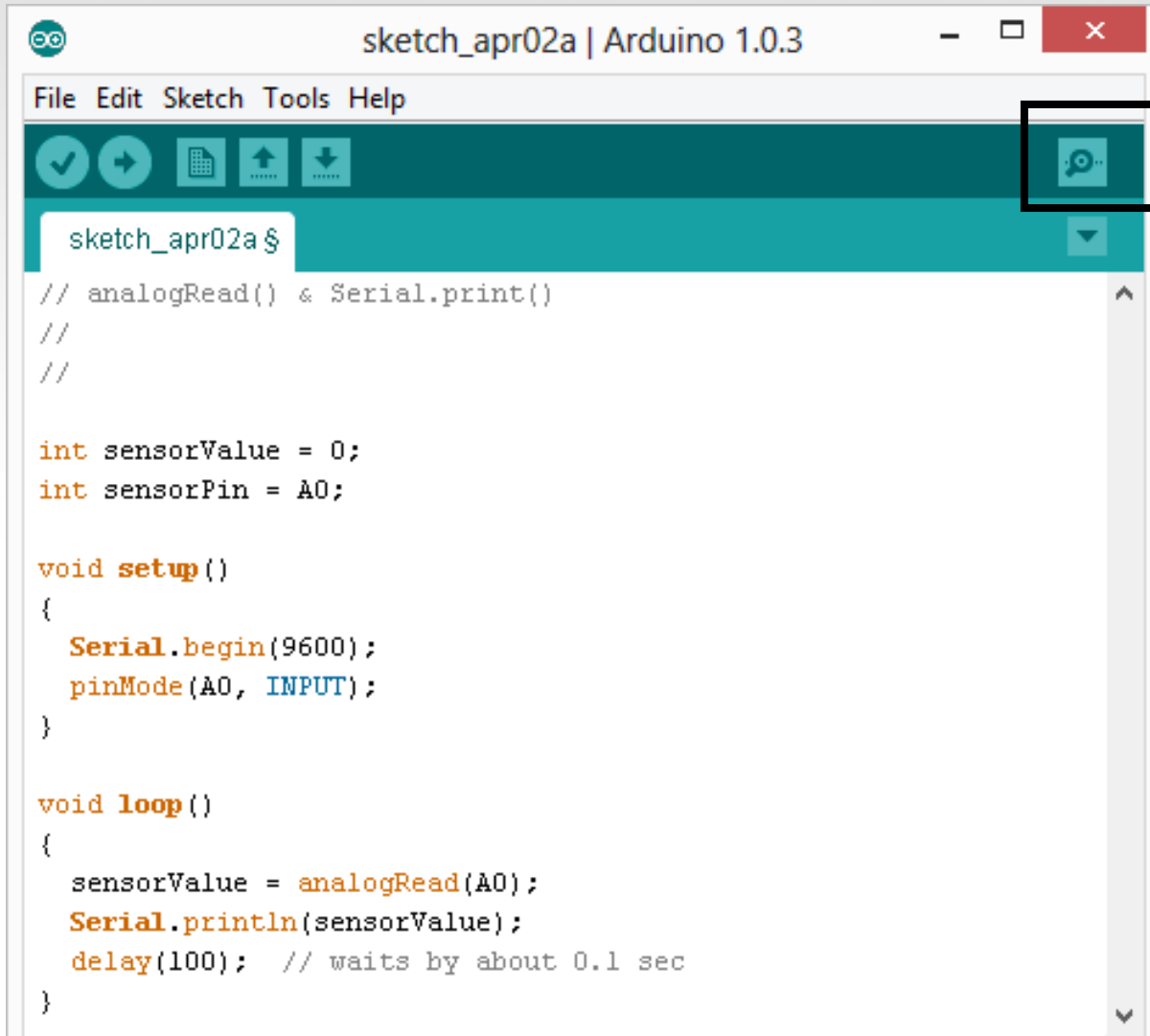
Initializes the Serial  
Communication

9600 baud data rate

prints data to serial bus



# Serial Monitor & analogRead()



```
sketch_apr02a | Arduino 1.0.3
File Edit Sketch Tools Help
sketch_apr02a $
// analogRead() & Serial.print()
//
//
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
}
```

Opens up a  
Serial Terminal  
Window

# Analog Sensors

Examples:

Sensors	Variables
Mic	soundVolume
Photoresistor	lightLevel
Potentiometer	dialPosition
Temp Sensor	temperature
Flex Sensor	bend
Accelerometer	tilt/acceleration

# Additional Serial Communication

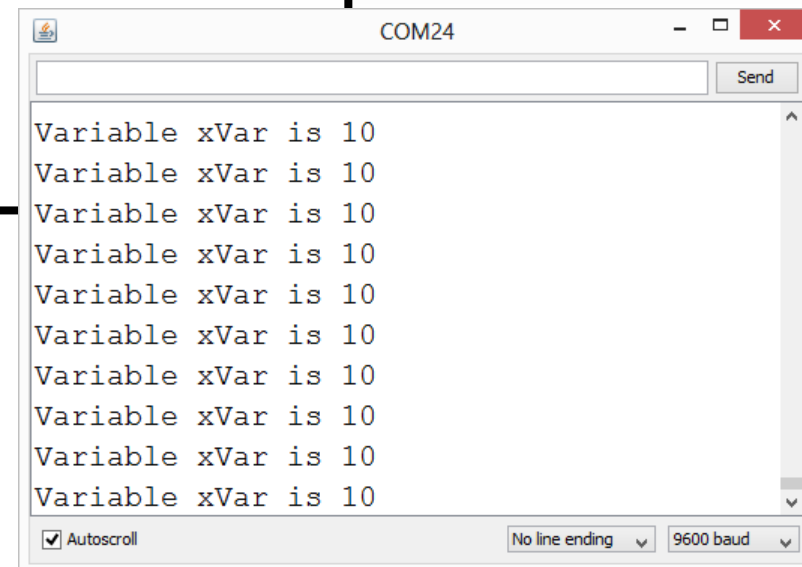
## Sending a Message

```
void loop ( )  
{  
  Serial.print("Hands on ") ;  
  Serial.print("Learning ") ;  
  Serial.println("is Fun!!!") ;  
  
}
```



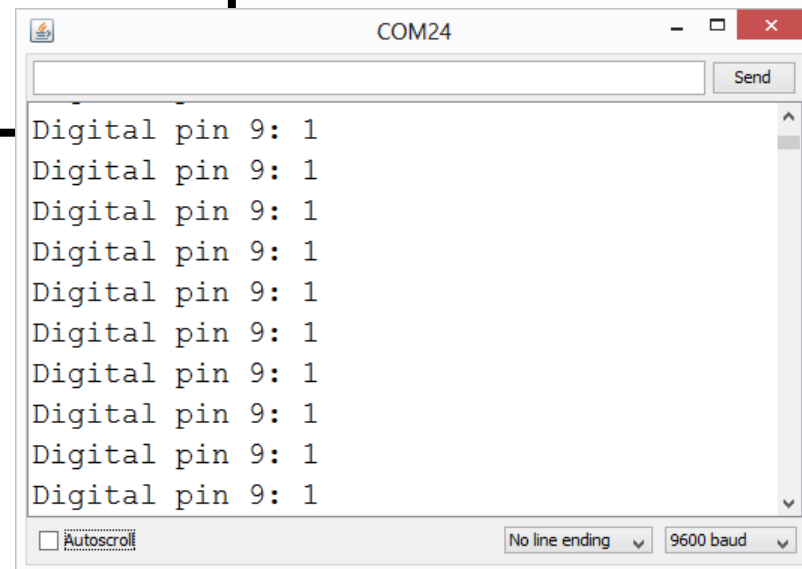
# Serial Communication: Serial Debugging

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



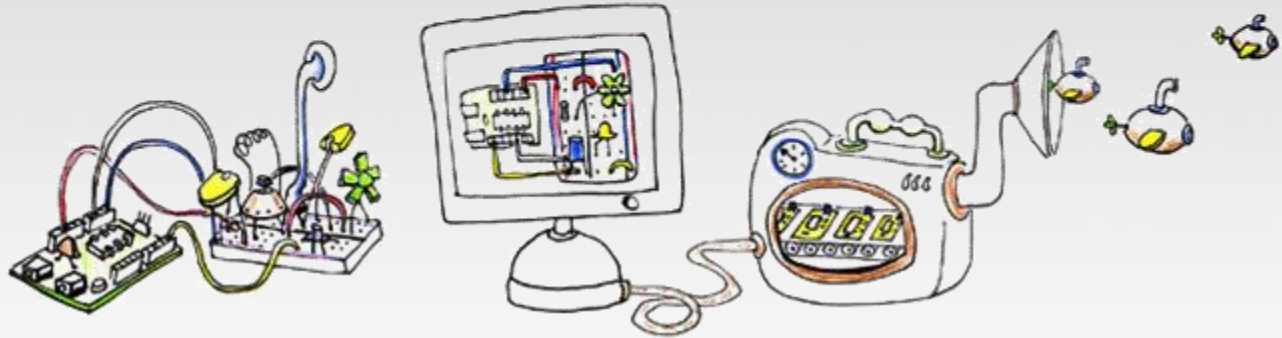
# Serial Communication: Serial Troubleshooting

```
void loop ( )  
{  
  Serial.print ("Digital pin 9: ");  
  Serial.println (digitalRead(9));  
}
```





# FRICTZING



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

## Other Boards

Each has their own advantages and costs.

Netduino

GHI Fez

ESP8266

ESP32

Teensy

Picaxe



**QUESTIONS?**