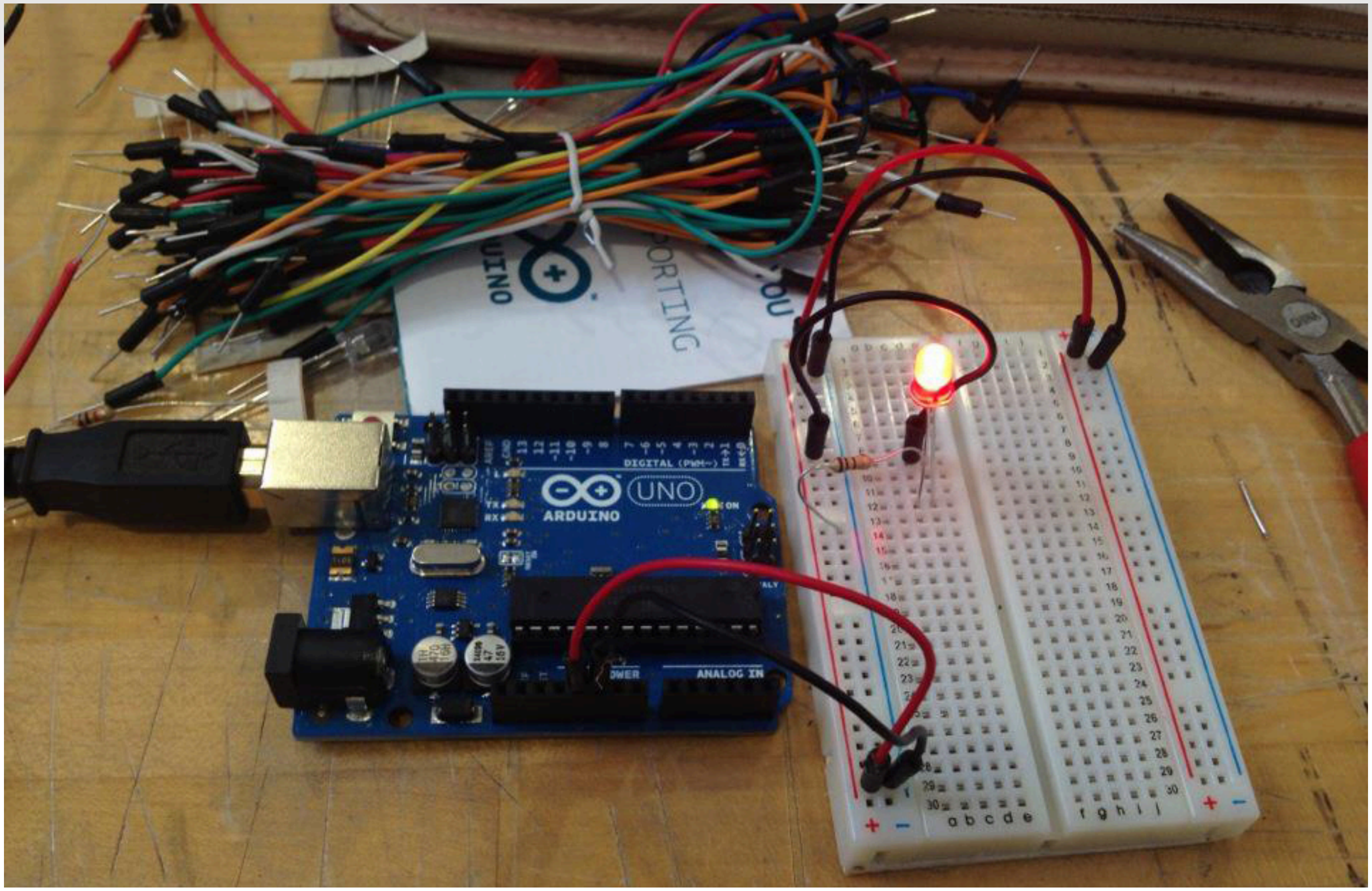
Adding control – let's use the Arduino and start programming!!!

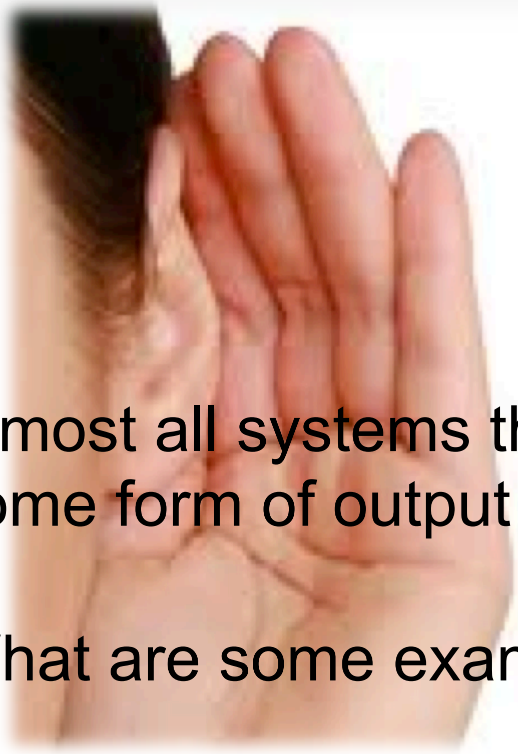


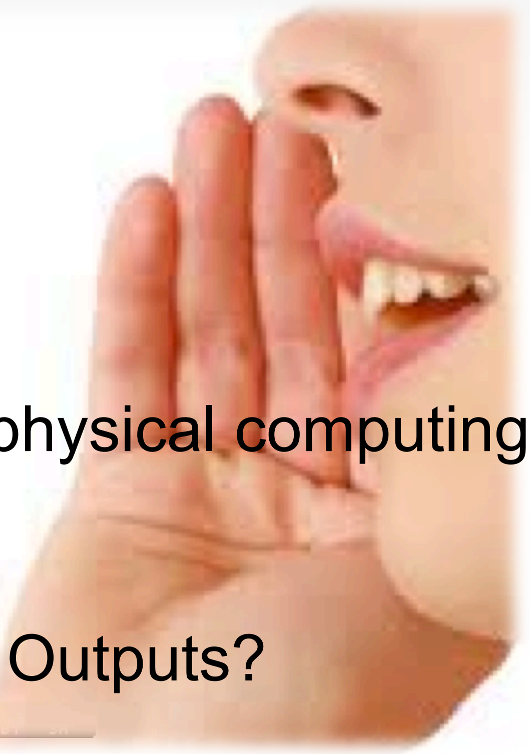
Concepts: INPUT vs. OUTPUT

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.

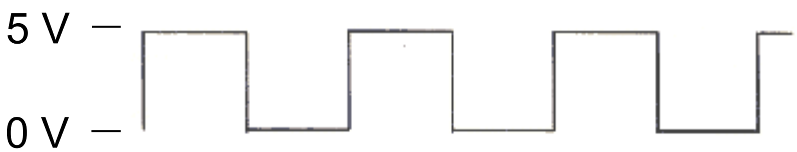
Output is any signal exiting the board.

Examples: Buttons Switches, Light Sensors, Flex Sensors, Humidity Sensors, Temperature Sensors…	Examples: LEDs, DC motor, servo motor, a piezo buzzer, relay, an RGB LED

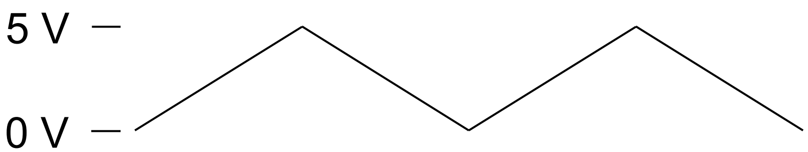
Concepts: Analog vs. Digital

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



Analog Signal

Open up Arduino

Hints:

For PC Users →

1.Let the installer copy and move the files to the appropriate locations, or

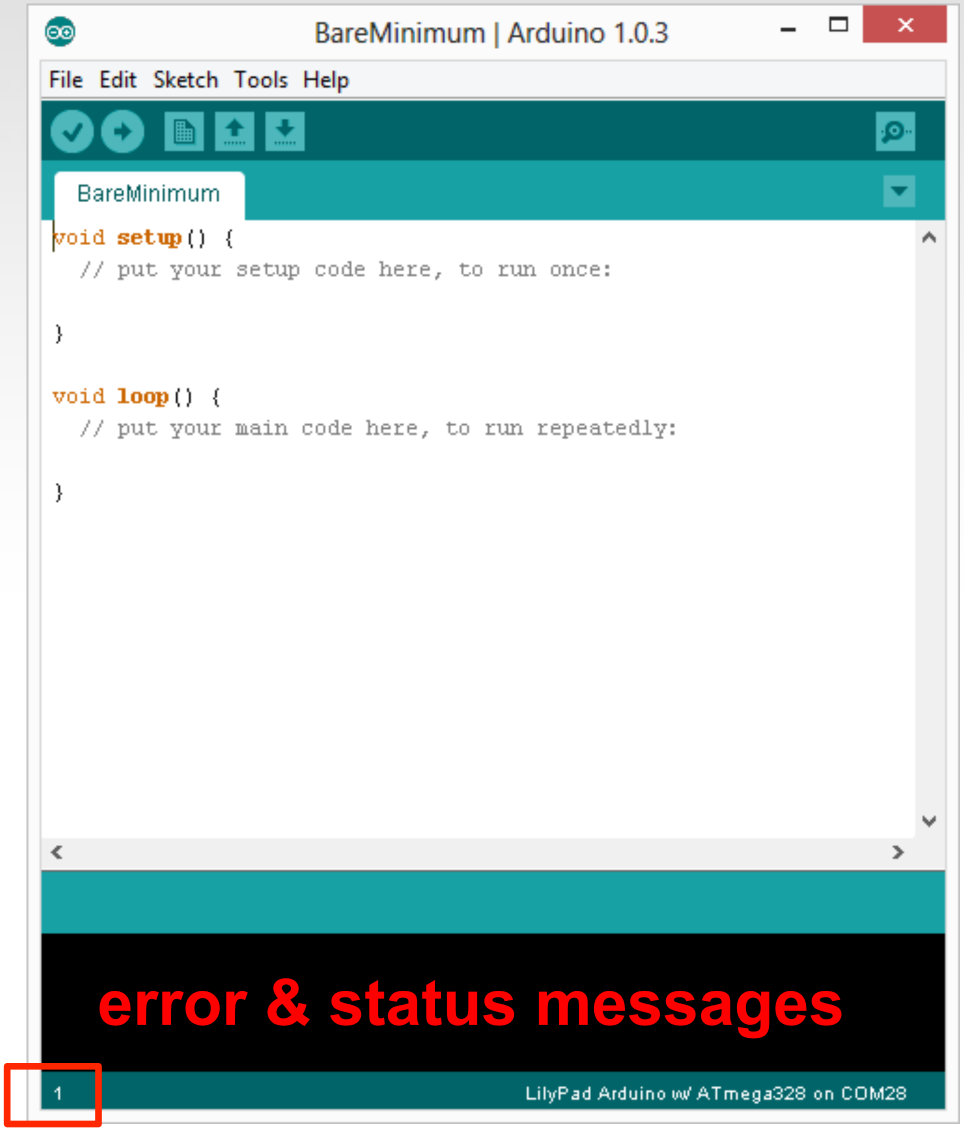
2.Create a folder under C:\Program Files (x86) called Arduino. Move the entire Arduino program folder here.

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)



BareMinimum | Arduino 1.0.3

File Edit Sketch Tools Help
BareMinimum

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

}

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

}
```

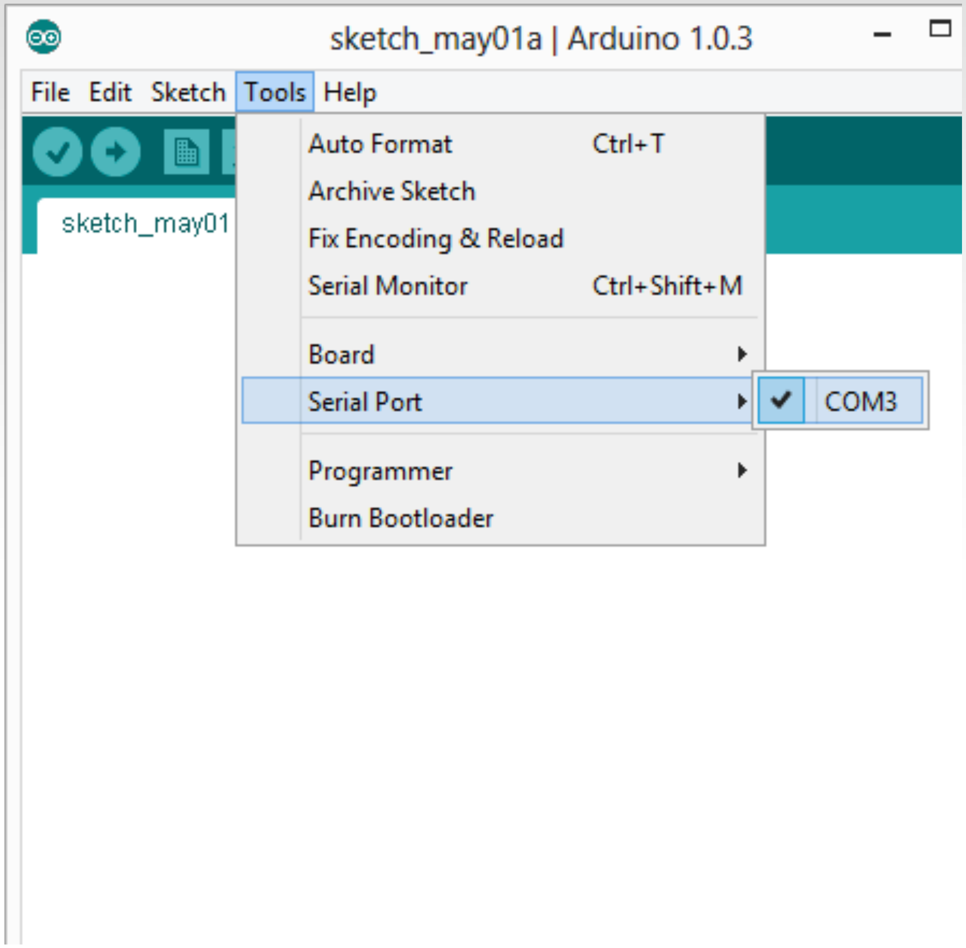
error & status messages

1
LilyPad Arduino w/ ATmega328 on COM28

Two required functions / methods / routines:

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

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

Settings: Tools → Serial Port



sketch_may01a | Arduino 1.0.3

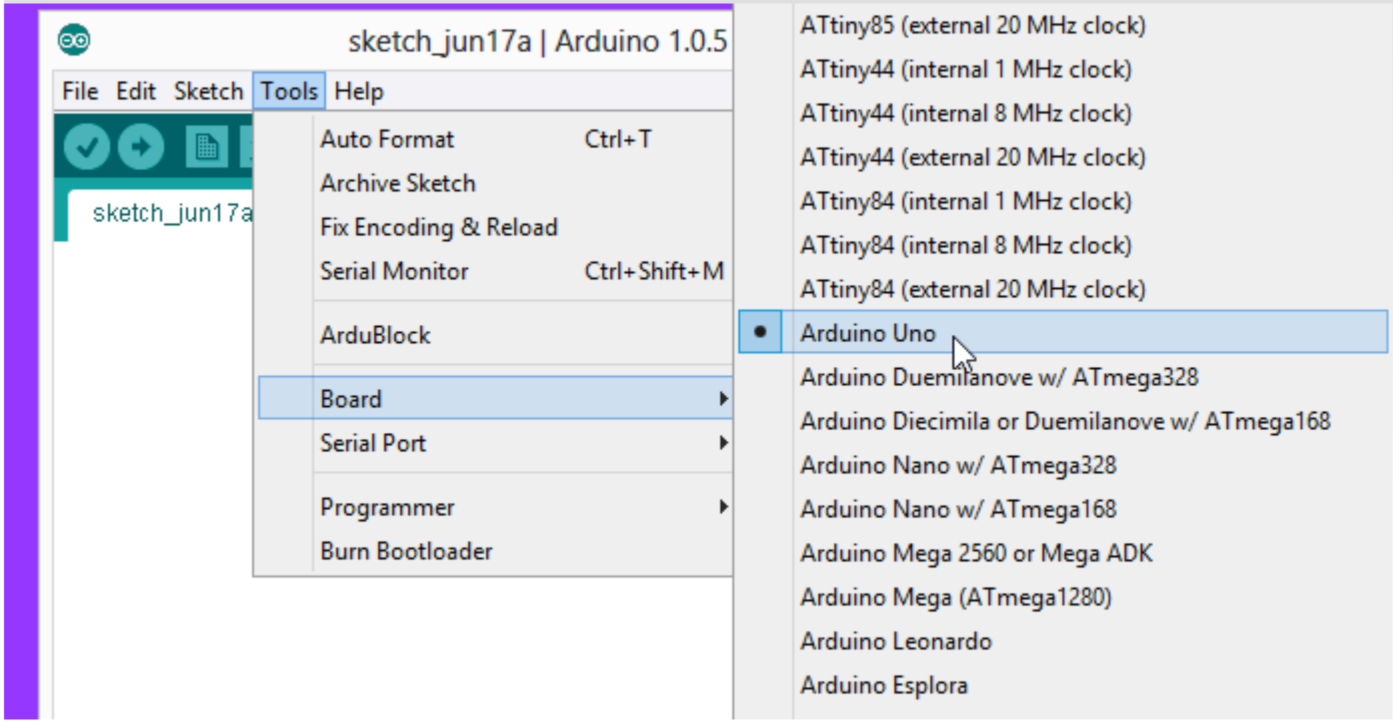
File Edit Sketch Tools Help

sketch_may01

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

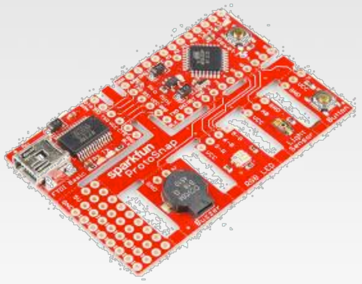
Check to make sure that the drivers are properly installed.

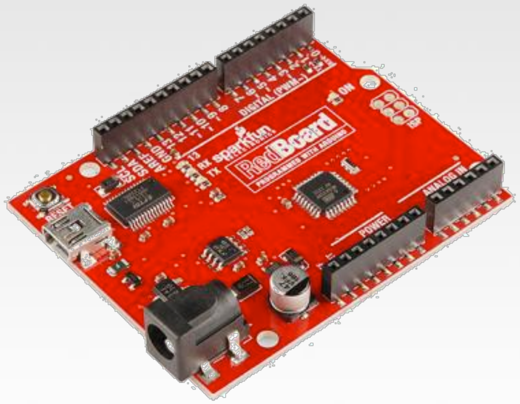
Settings: Tools → Board

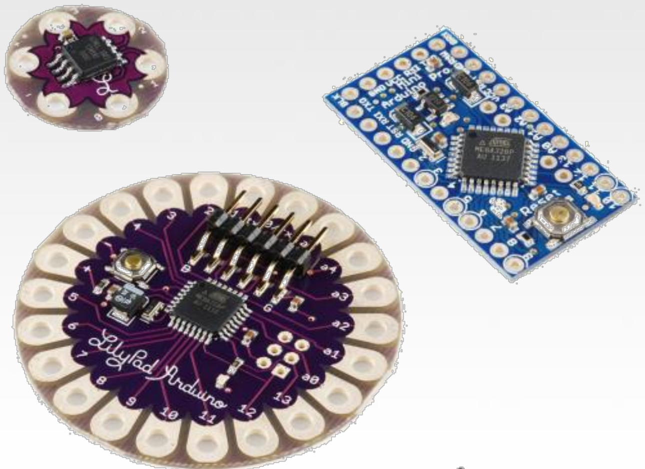

sketch_jun17a | Arduino 1.0.5
File
Edit
Sketch
Tools
Help
sketch_jun17a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
ArduBlock
Board
Serial Port
Programmer
Burn Bootloader
ATtiny85 (external 20 MHz clock)
ATtiny44 (internal 1 MHz clock)
ATtiny44 (internal 8 MHz clock)
ATtiny44 (external 20 MHz clock)
ATtiny84 (internal 1 MHz clock)
ATtiny84 (internal 8 MHz clock)
ATtiny84 (external 20 MHz clock)
● Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora

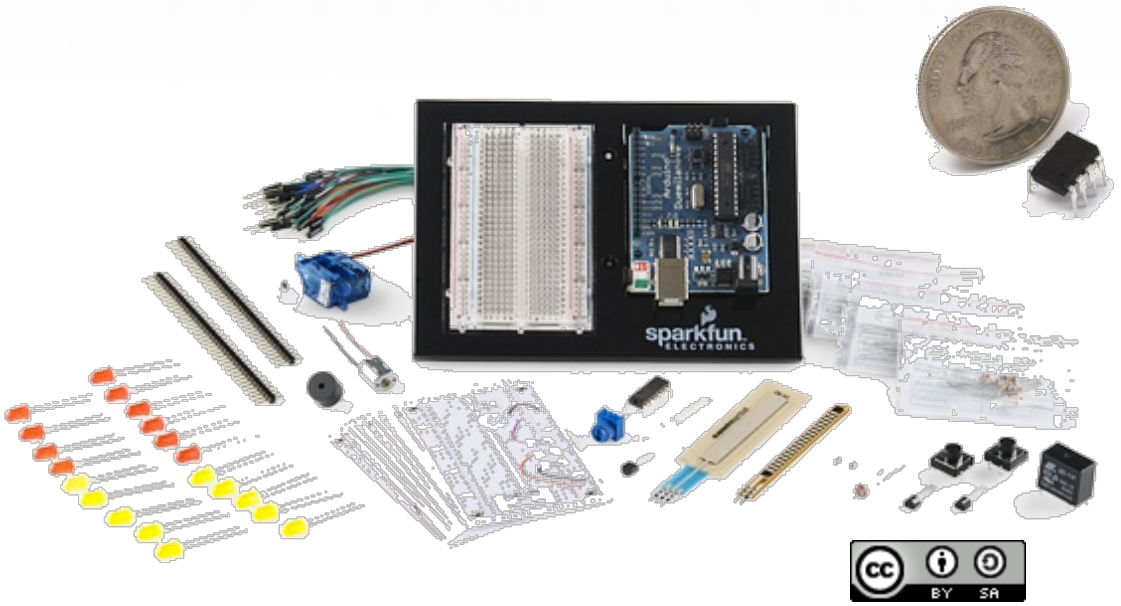
Next, double-check that the proper board is selected under the Tools→Board menu.

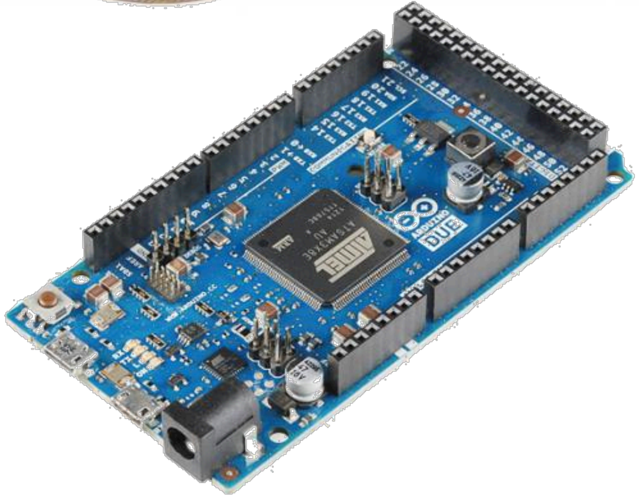
Arduino & Arduino Compatible Boards



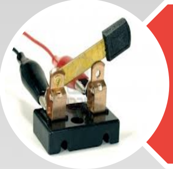






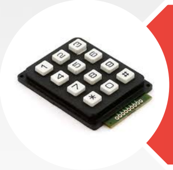


BIG 6 CONCEPTS

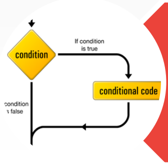


digitalWrite()

analogWrite()

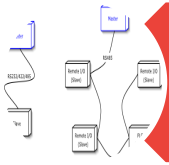


digitalRead()



if() statements / Boolean

analogRead()



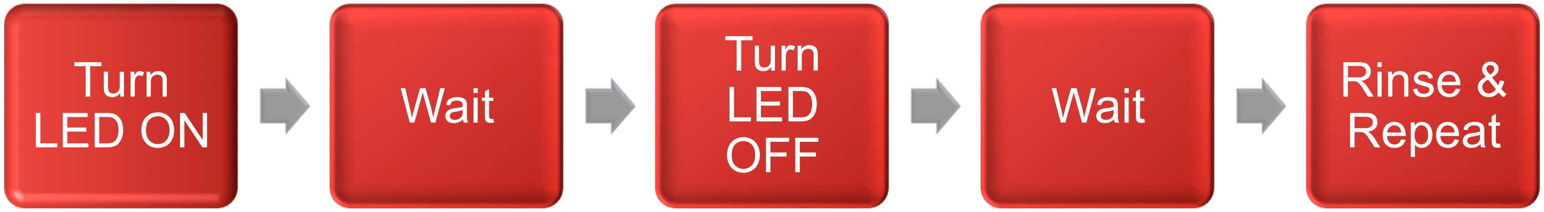
Serial communication

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

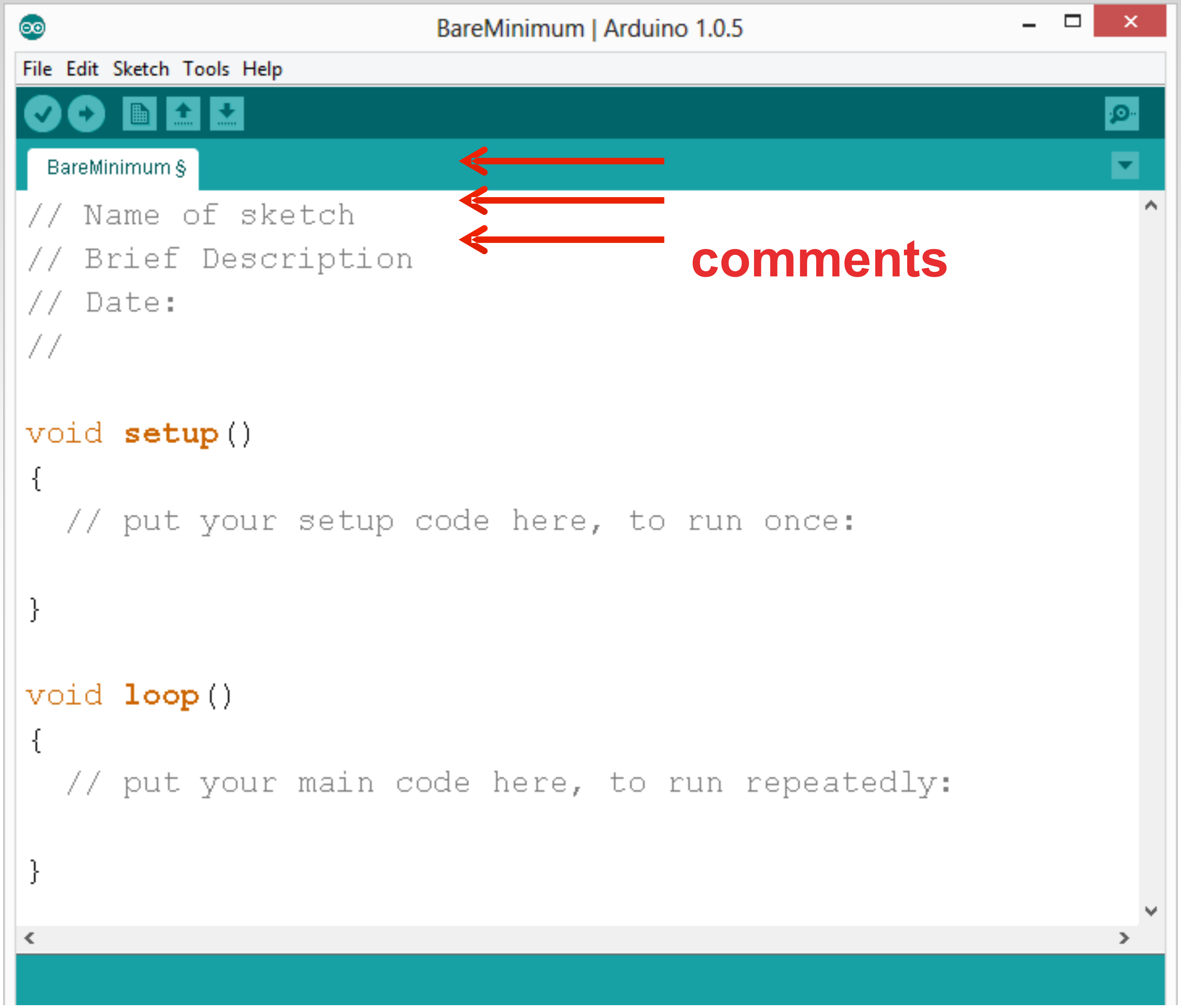
Wait

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….
*/
```



```
// 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:

}
```

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

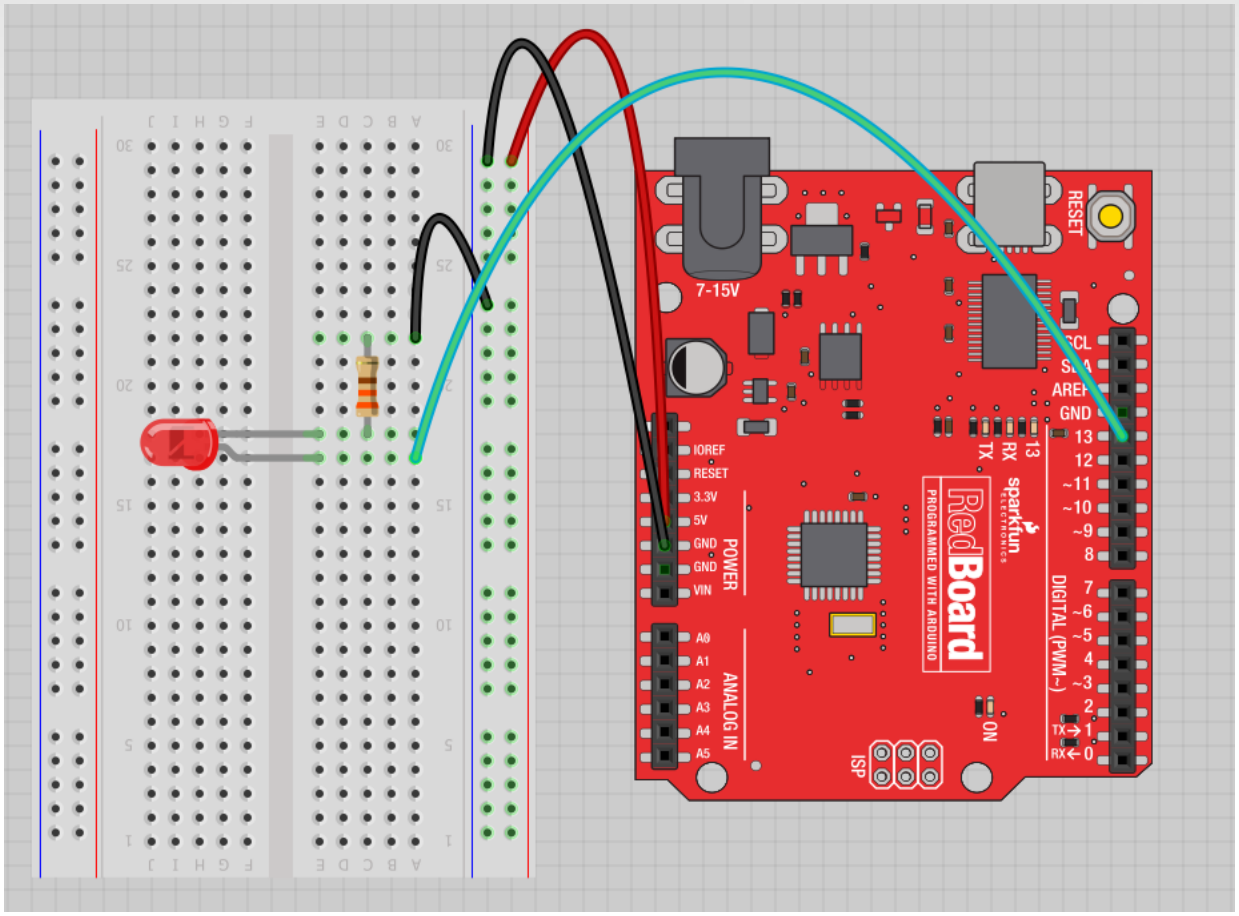


Image created in Fritzing

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

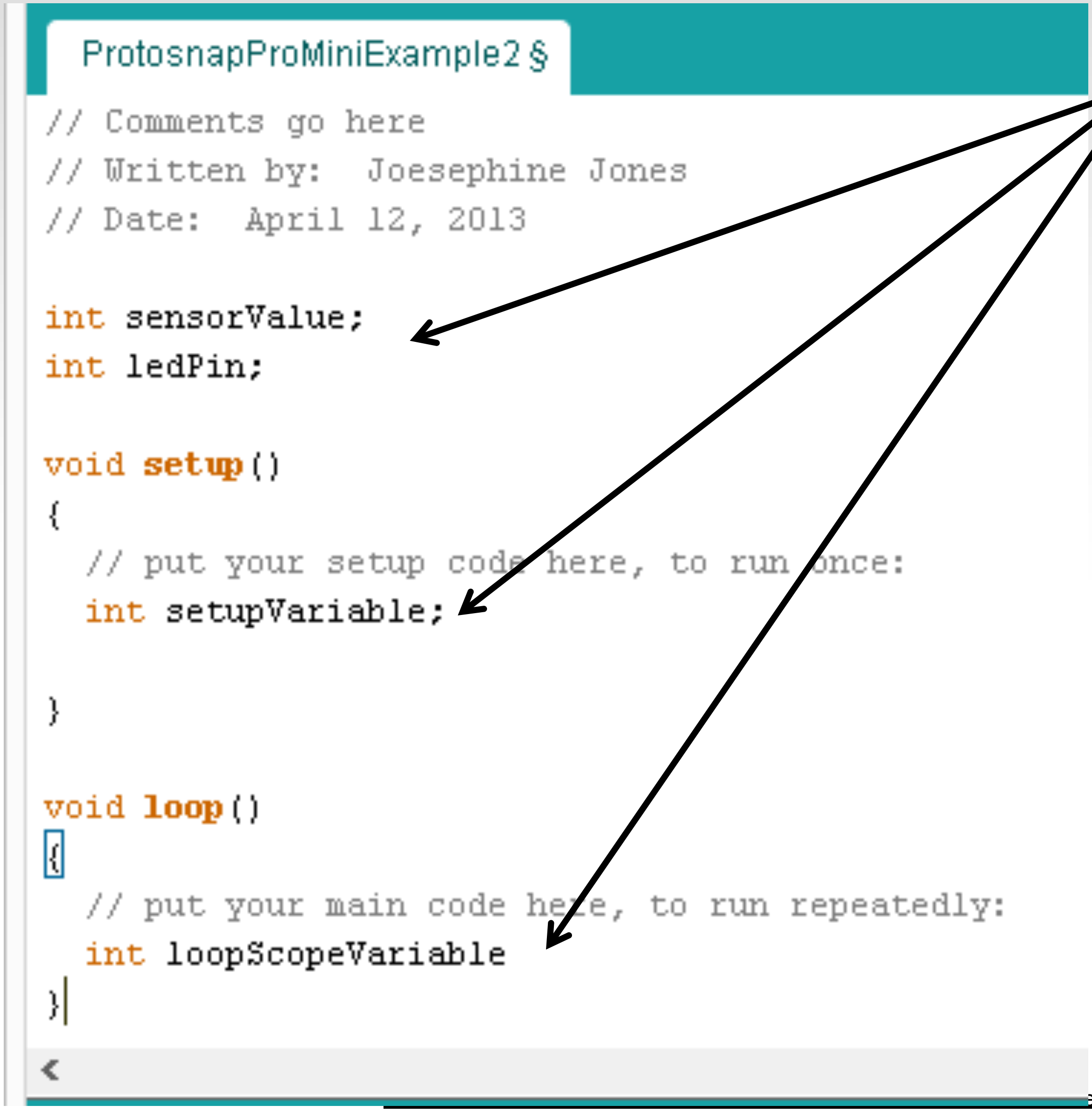
Can you blink two, three, or four LEDs?

(Hint: Each LED will need it’s own 330Ω resistor.)

Generate your own morse code flashing

How about → Knight Rider? Disco? Police Light?

Programming Concepts: Variables



```
// 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	16 bits	32 bits
byte	int	long
char	unsigned int	unsigned long
		float

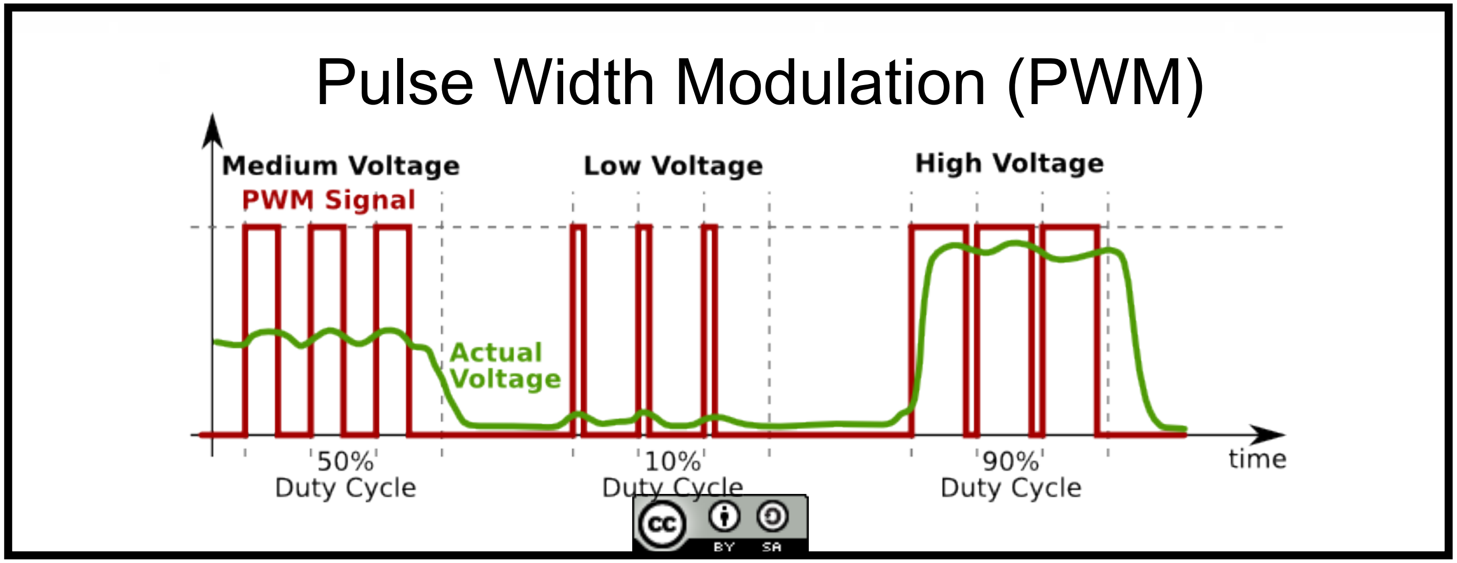
Fading in and Fading Out (Analog or Digital?)

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.



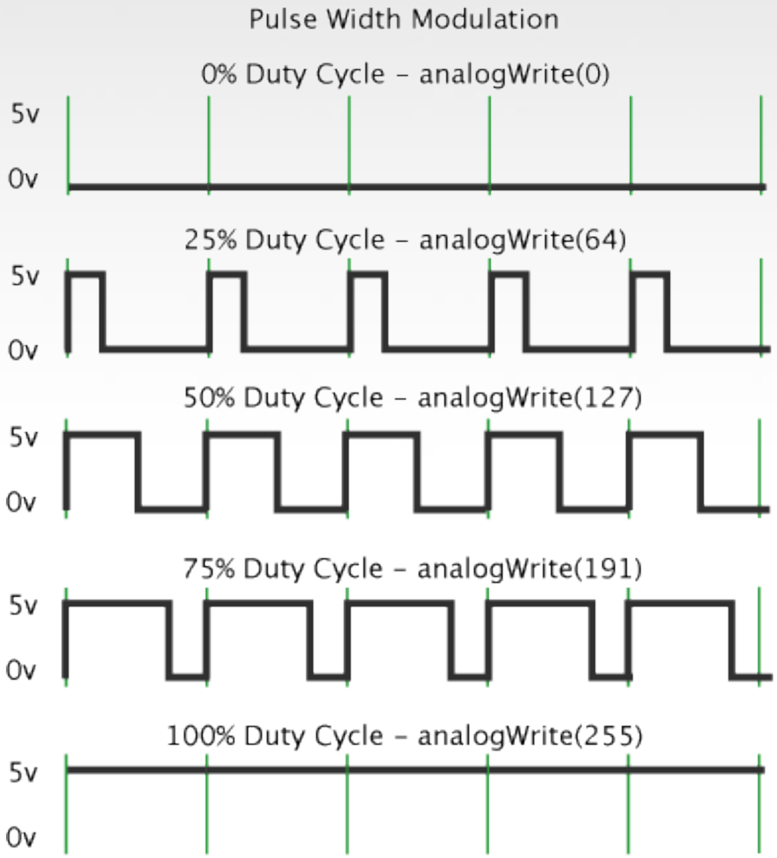
Project #2 – Fading
Introducing a new command…

```
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).

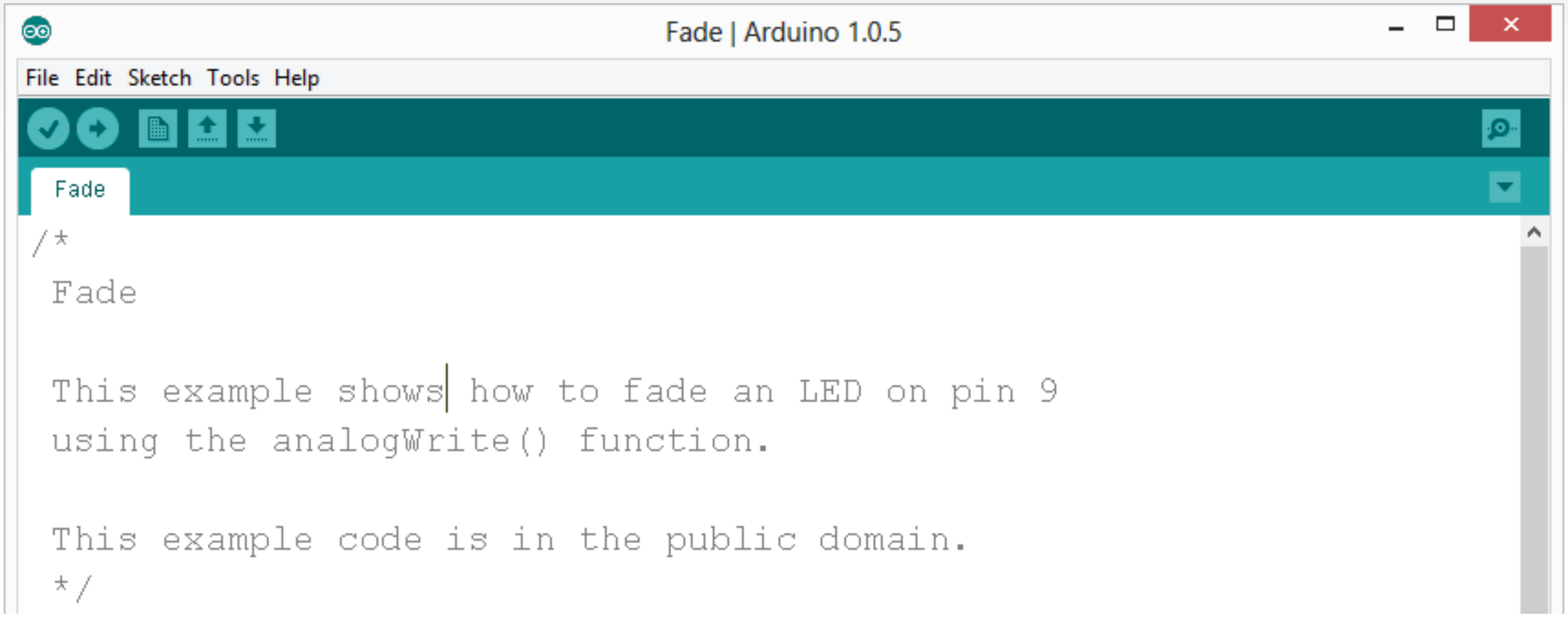
0 => 0V | 255 => 5V


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Move one of your LED pins over to Pin 9

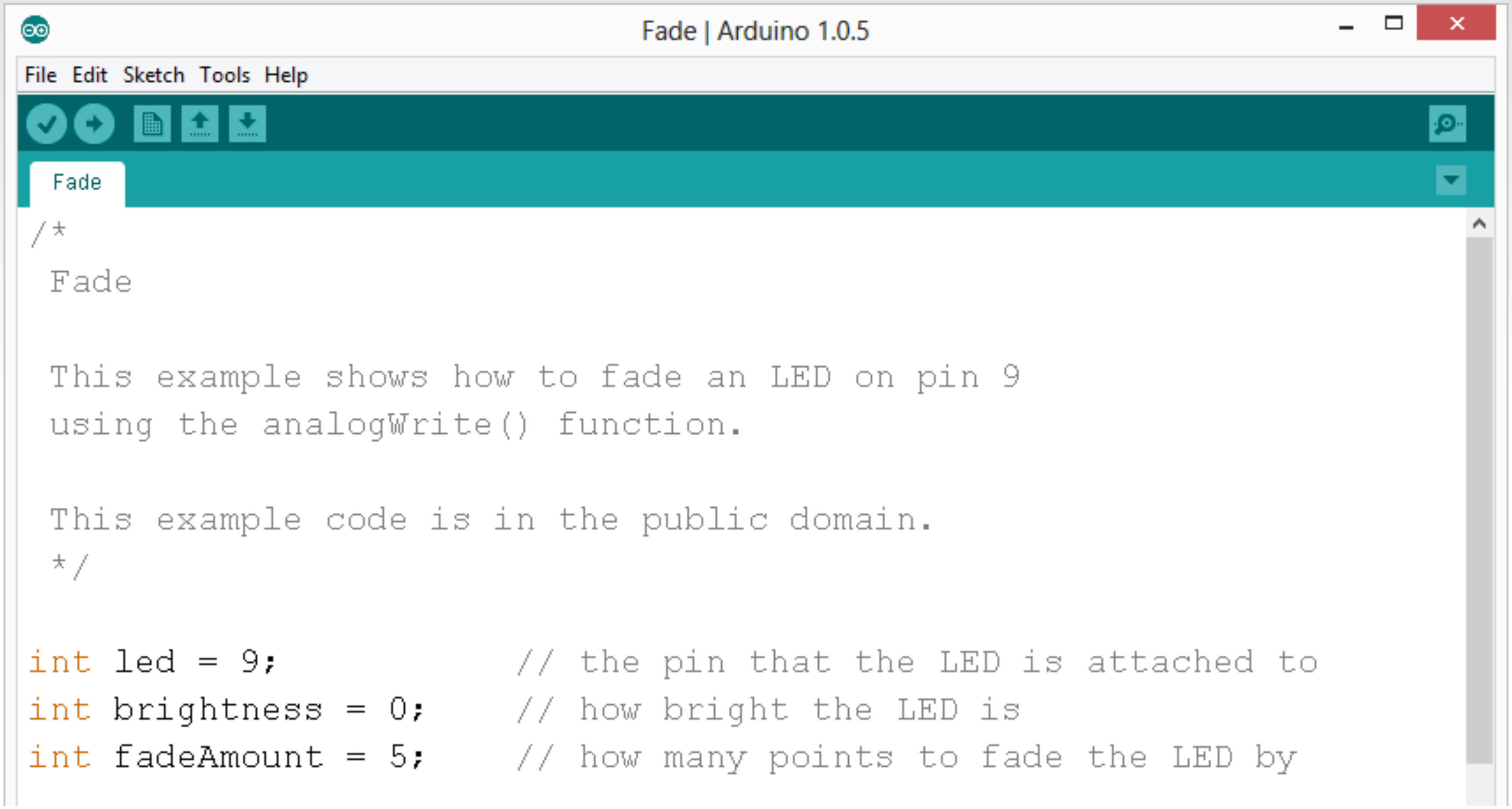
In Arduino, open up:

File → Examples → 01.Basics → Fade



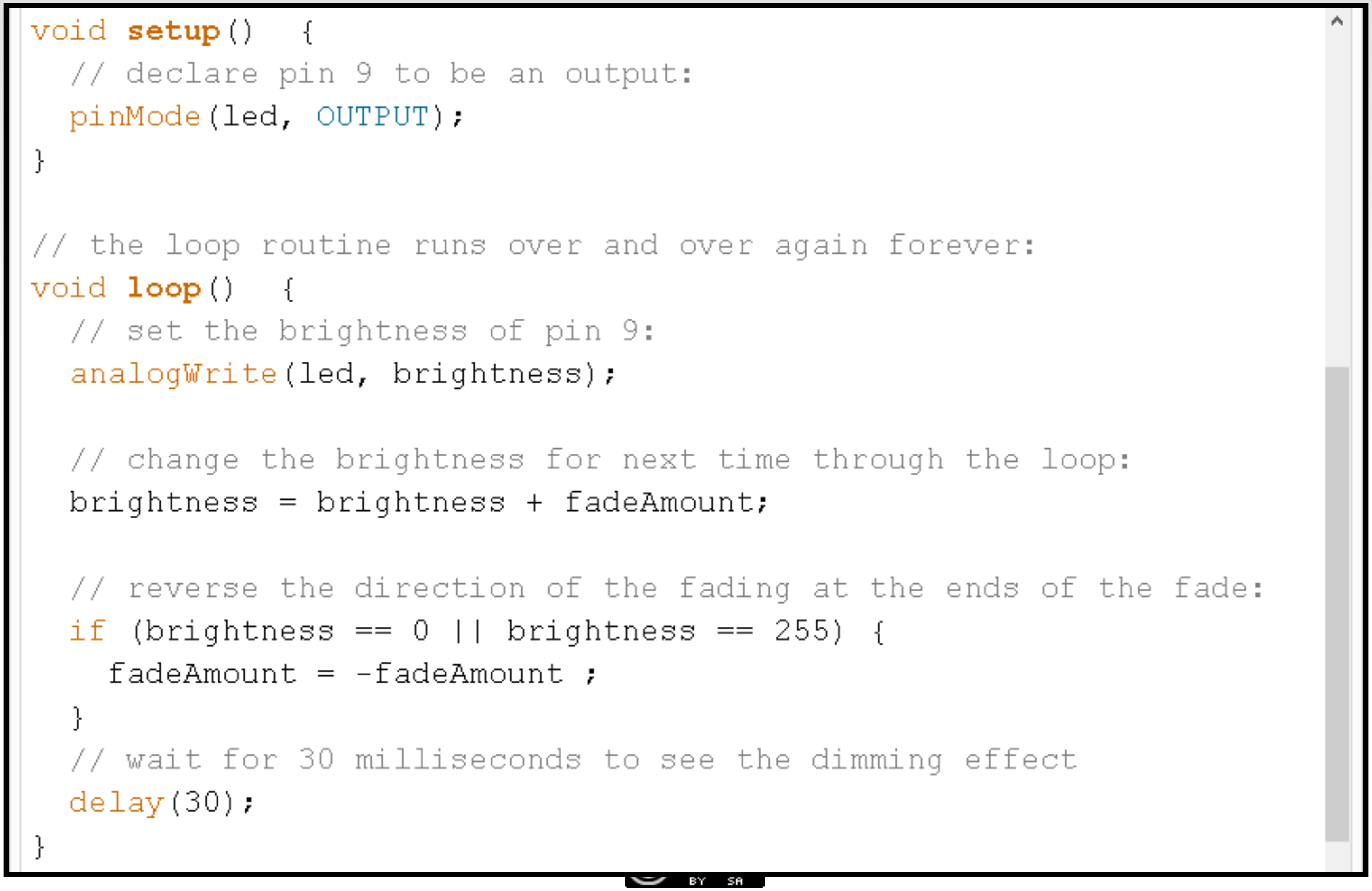
```
/*
 Fade

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

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

Fade - Code Review



```
/*
 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

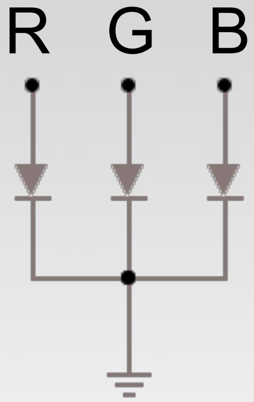


```
void setup()  {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// 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);
}
```

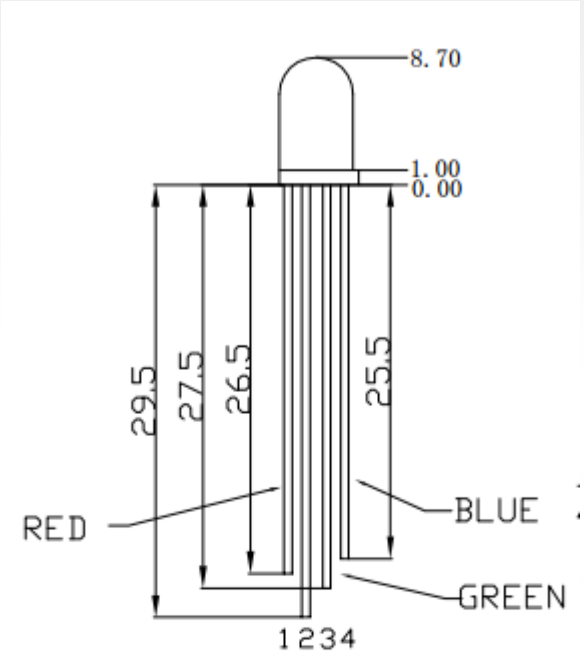
Project# 2 -- Fading

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.



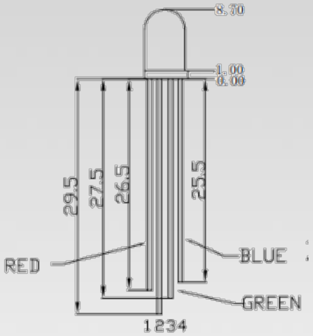
Color Mixing
Tri-color LED

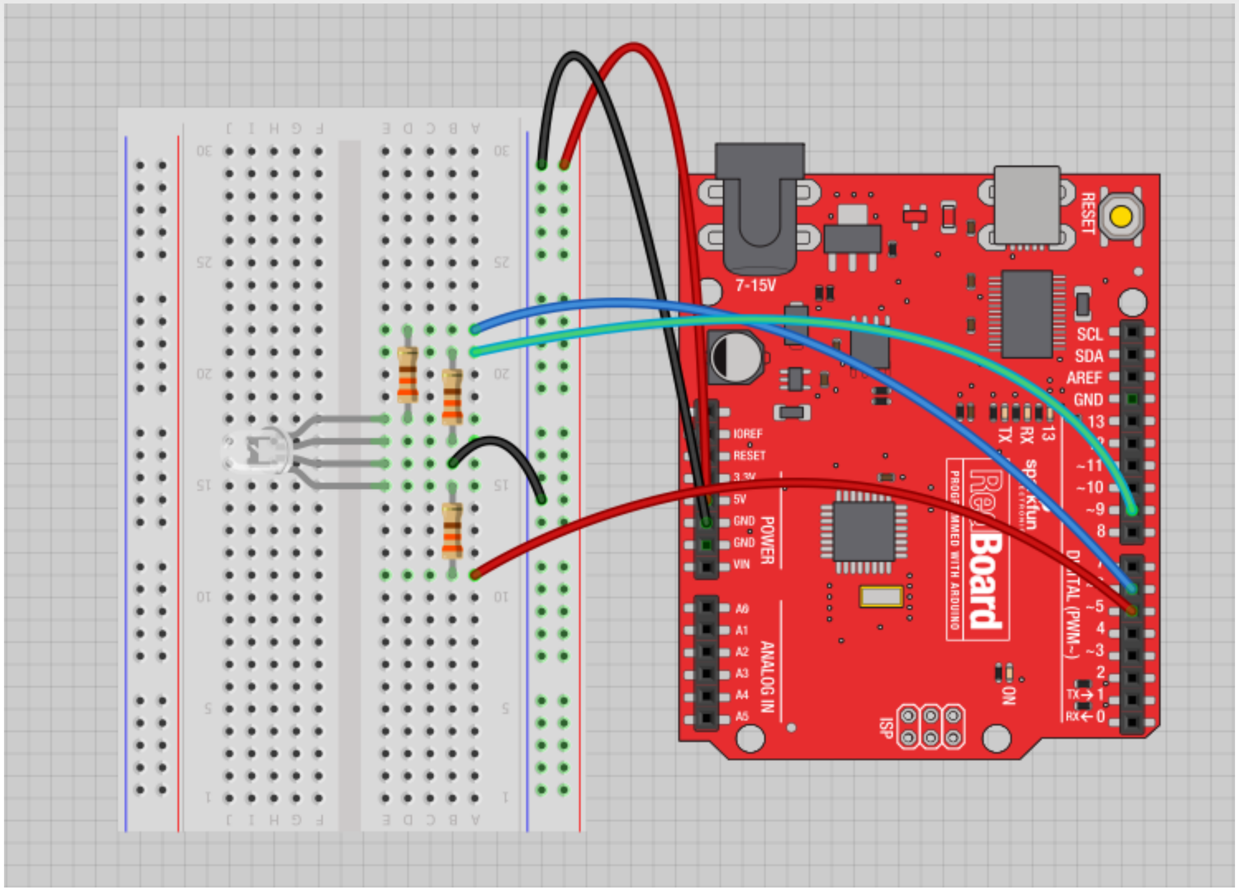


In the SIK, this is a standard – Common Cathode LED

This means the negative side of the LED is all tied to Ground.

Project 3 – RGB LED



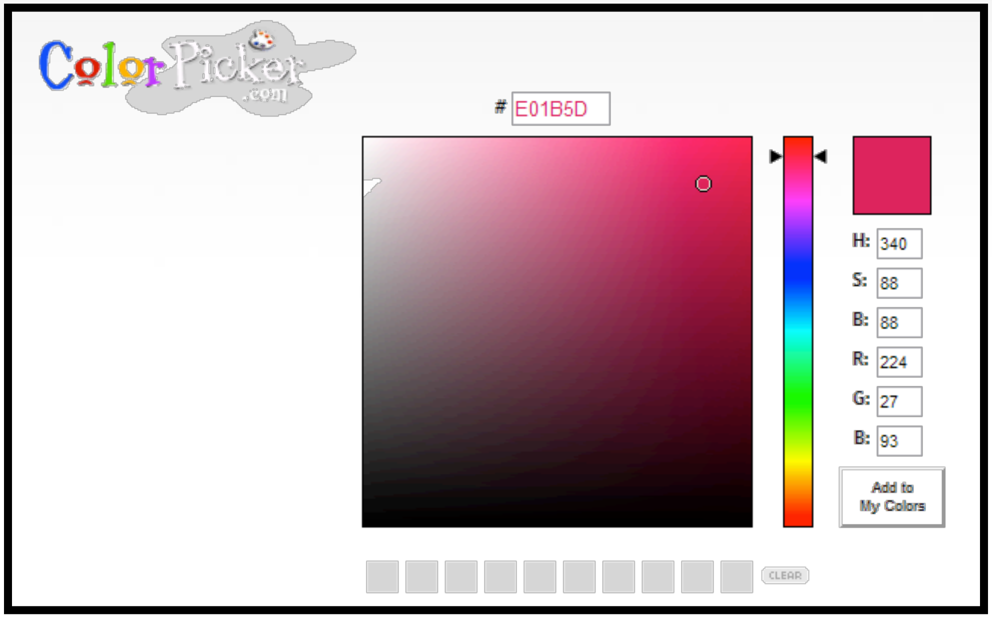


Note: The longest leg of the RGB LED is the Common Cathode. This goes to GND.

Use pins 5, 6, & 9

How many unique colors can you create?

$$\# \ of \ unique \ colors = 256 \cdot 256 \cdot 256$$
$$= 16{,}777{,}216 \ colors!$$



E01B5D

H: 340
S: 88
B: 88
R: 224
G: 27
B: 93

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.

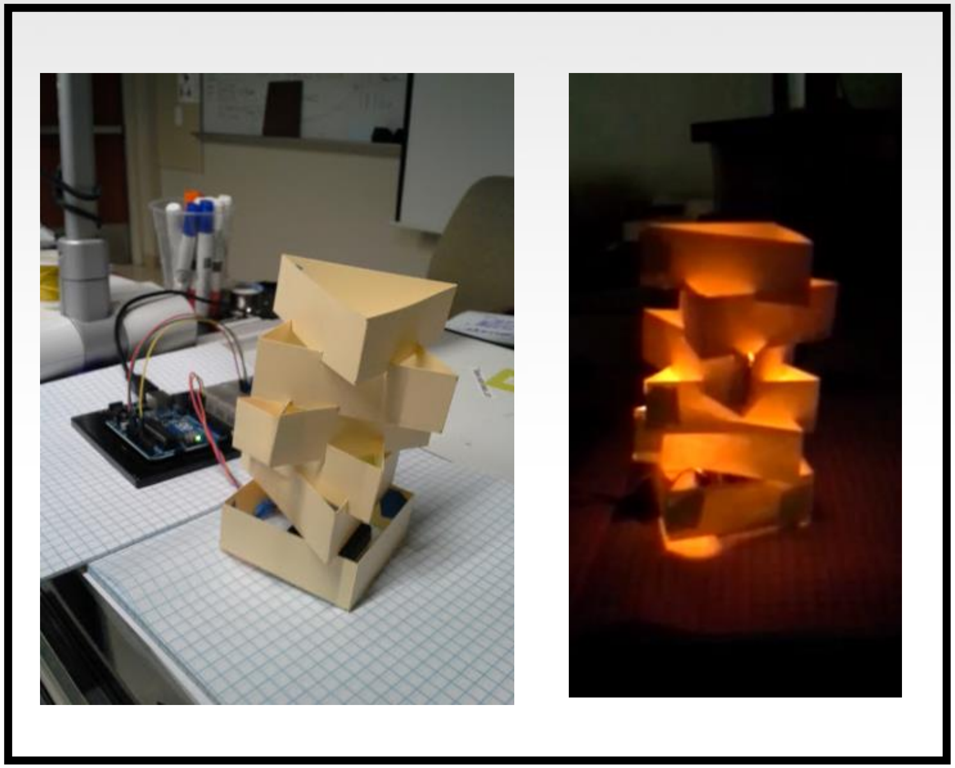
RGB LED Color Mixing

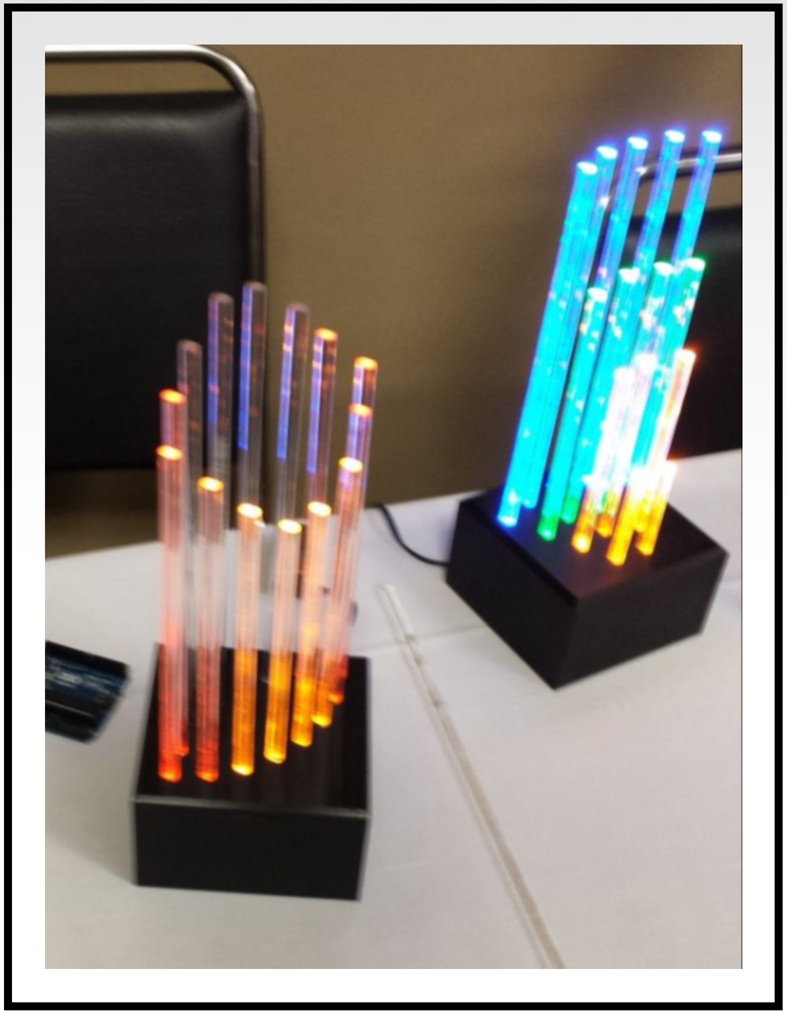
```
int redPin = 5;
int greenPin = 6;
int bluePin = 9;

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);
}
```

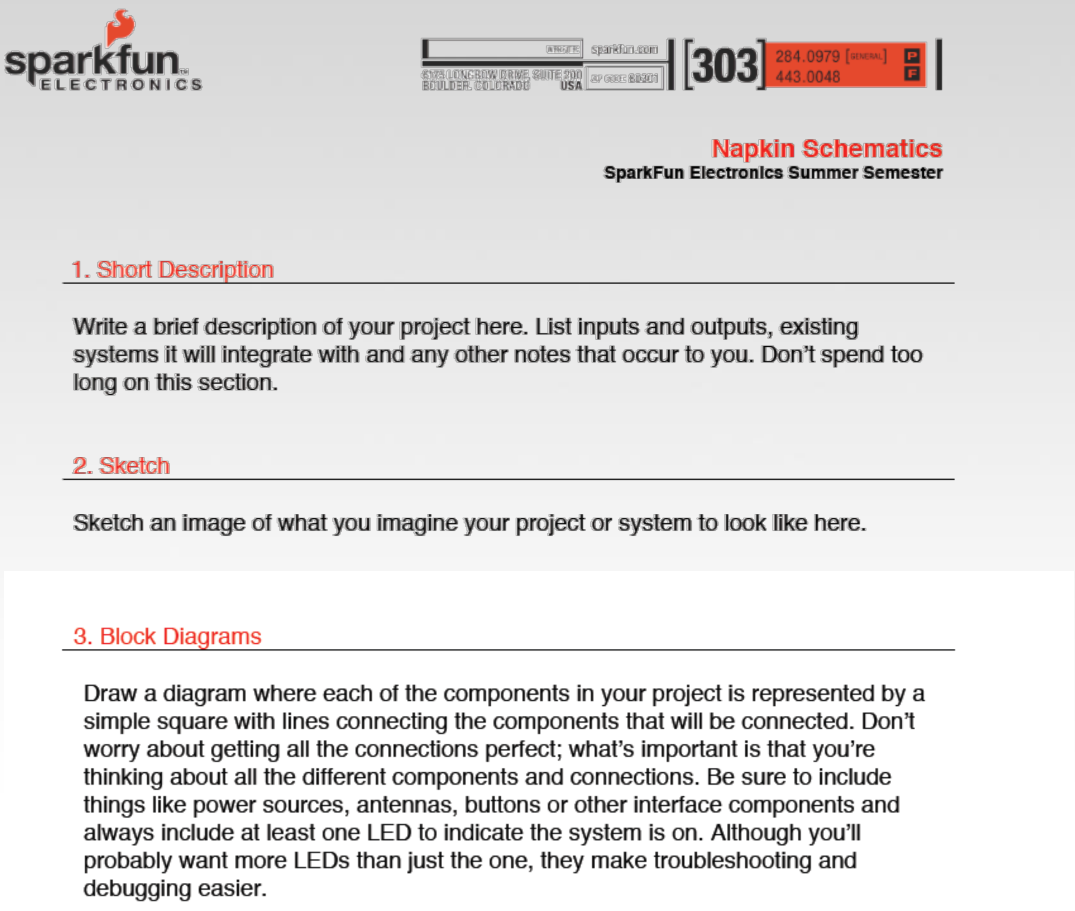
Project: Mood Lamp / Light Sculpture





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.**



sparkfun ELECTRONICS

WEBSITE sparkfun.com
6175 LONGBOW DRIVE, SUITE 200
BOULDER, COLORADO USA
ZIP CODE 80301
[303] 284.0979 [GENERAL] P
443.0048 F

Napkin Schematics
SparkFun Electronics Summer Semester

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

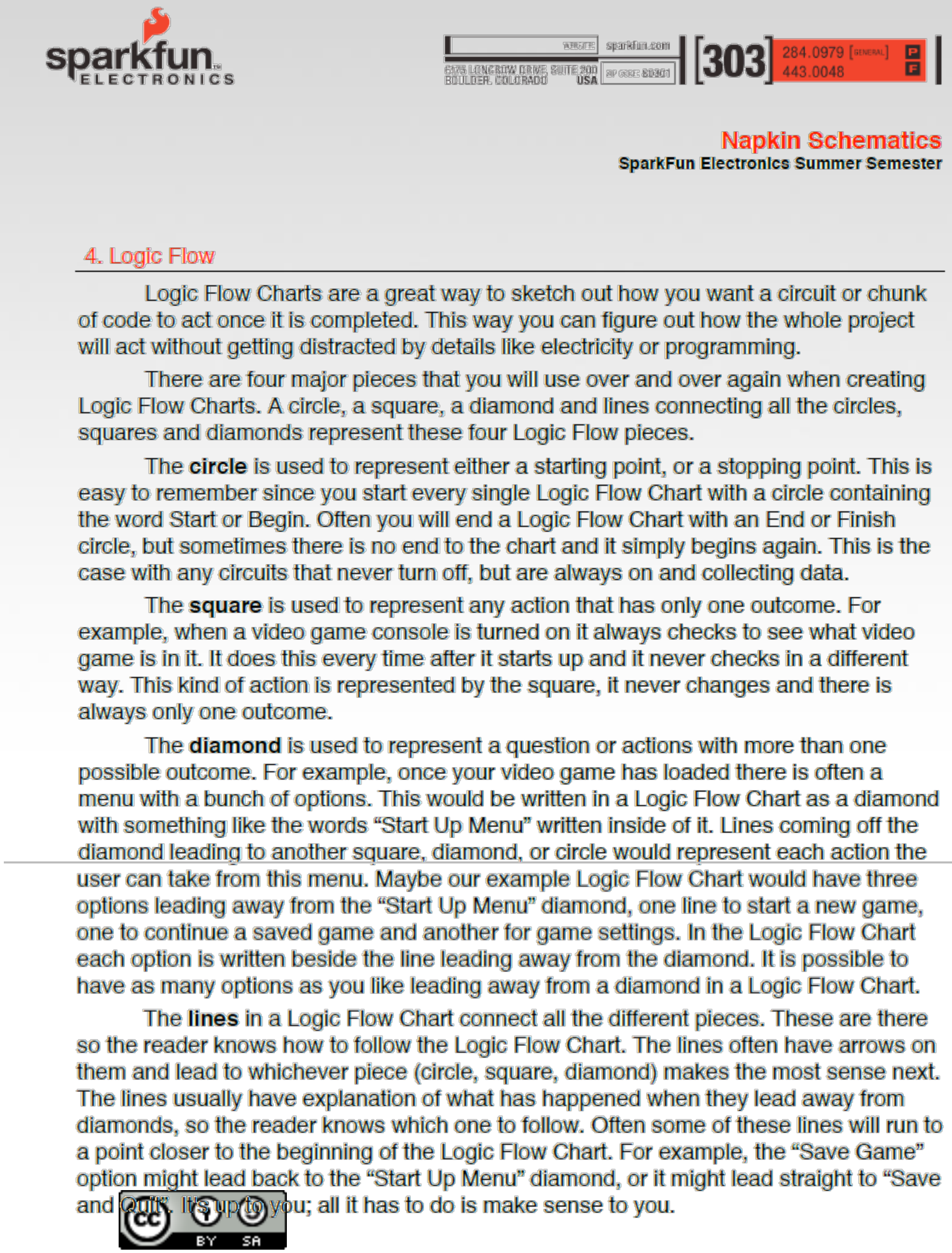
Sketch an image of what you imagine your project or system to look like here.

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.**



sparkfun ELECTRONICS

WEBSITE sparkfun.com
6175 LONGBOW DRIVE, SUITE 200
BOULDER, COLORADO USA
ZIP CODE 80301
[303] 284.0979 [GENERAL] P
443.0048 F

Napkin Schematics
SparkFun Electronics Summer Semester

4. 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.

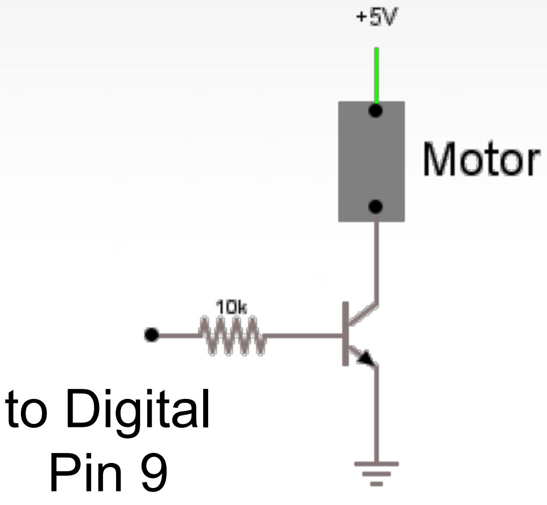
The **square** is used to represent any action that has only one outcome. For example, when a video game console is turned on it always checks to see what video game is in it. It does this every time after it starts up and it never checks in a different way. This kind of action is represented by the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

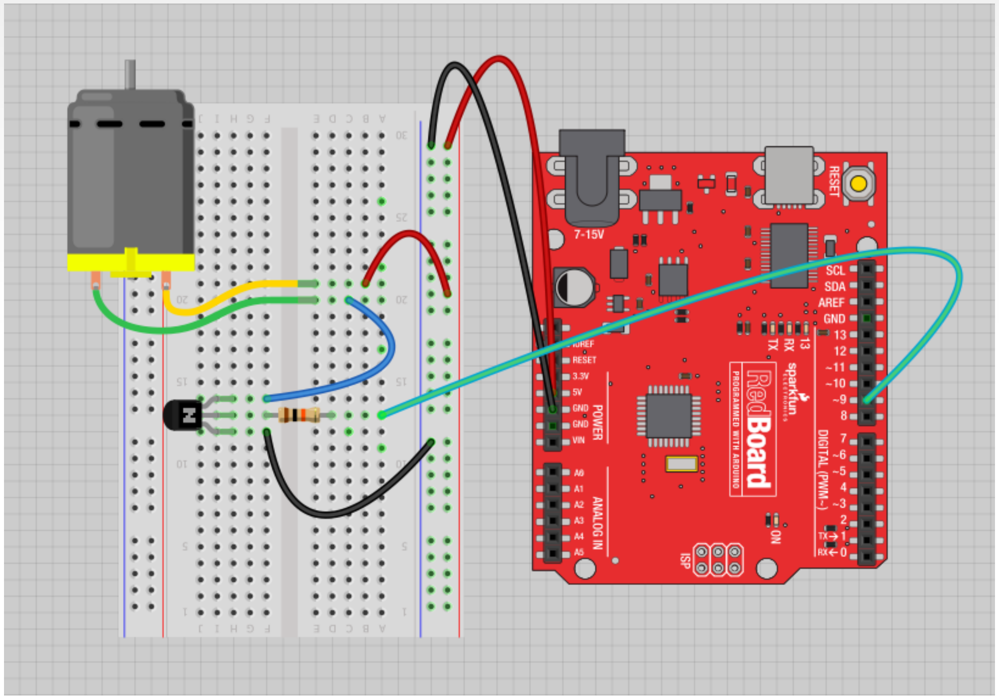
The **lines** in a Logic Flow Chart connect all the different pieces. These are there so the reader knows how to follow the Logic Flow Chart. The lines often have arrows on them and lead to whichever piece (circle, square, diamond) makes the most sense next. The lines usually have explanation of what 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.

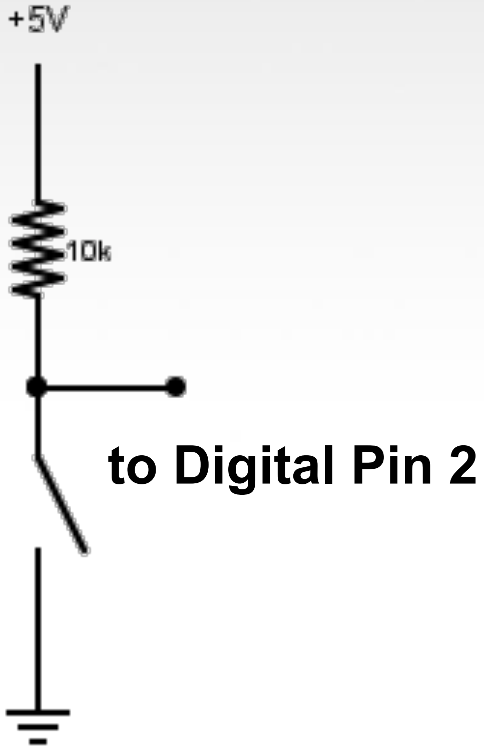
- Both digital and analog sensors are forms of input
- Input can also take many other forms: Keyboards, a mouse, infrared sensors, biometric sensors, or just plain voltage from a circuit

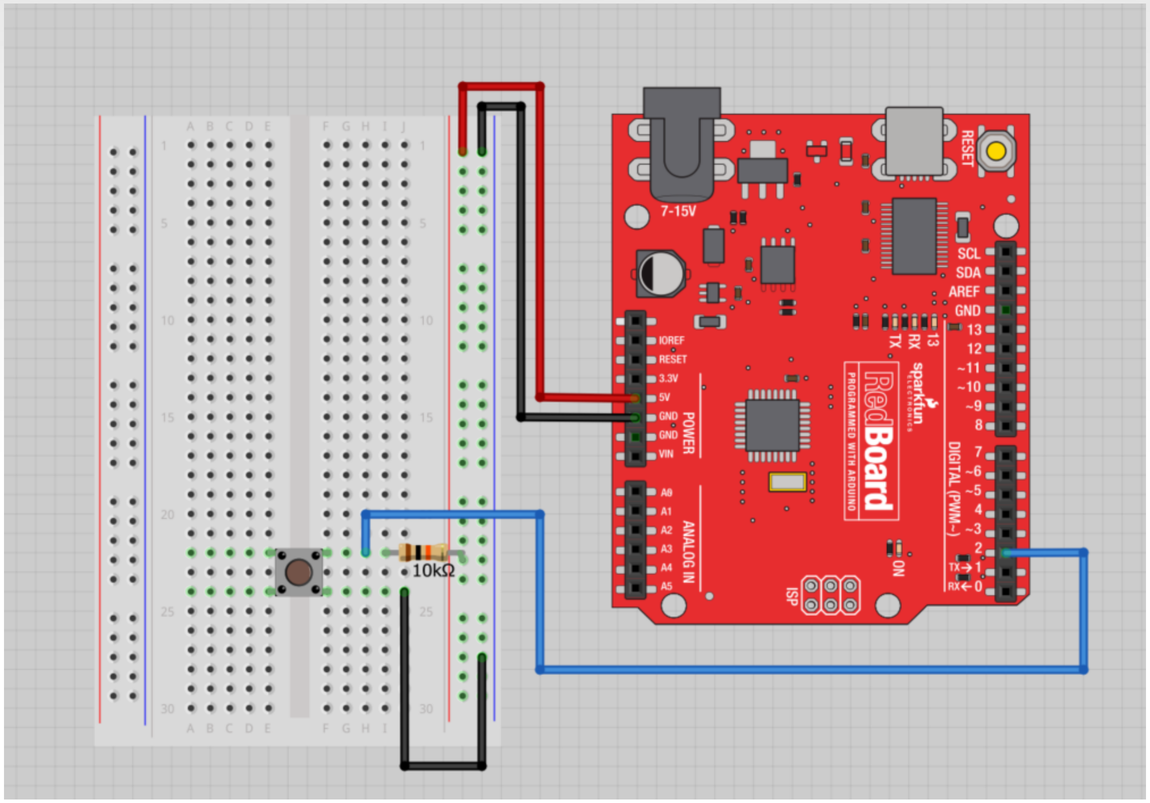
Project #4 – Digital Input

In Arduino, open up:

File → Examples → 02.Digital → Button

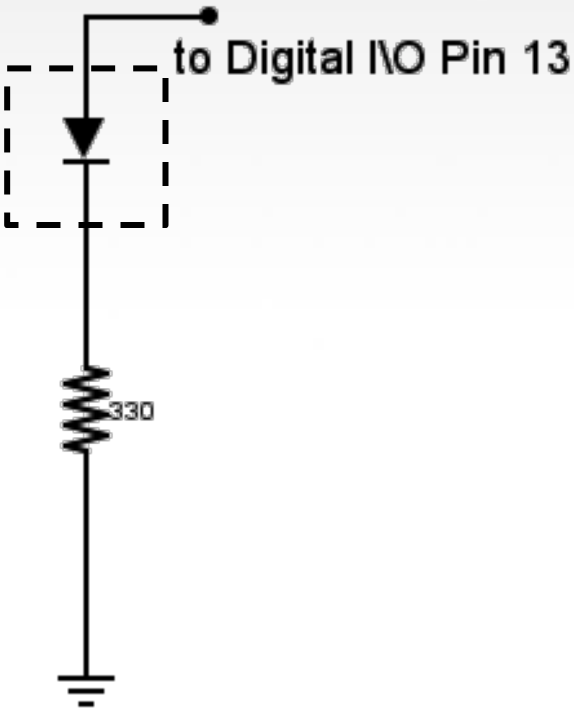
Digital Sensors (a.k.a. Switches)
Pull-up Resistor (circuit)

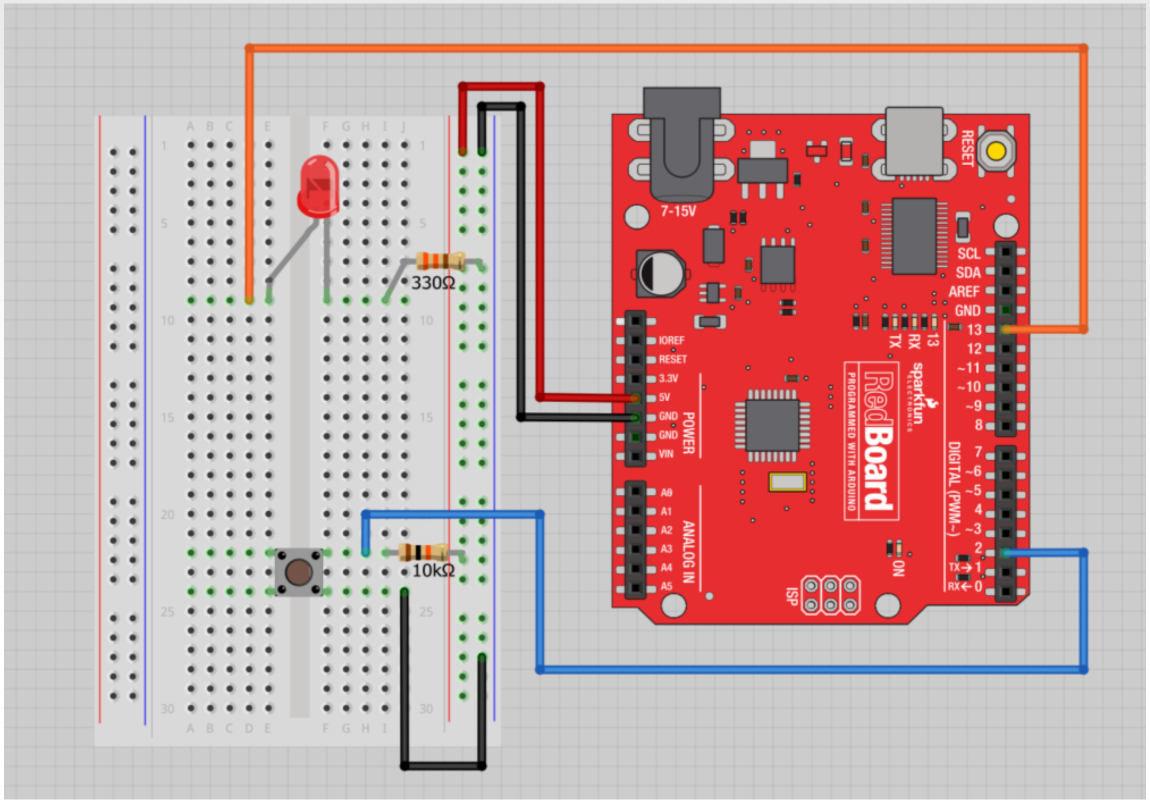




Digital Sensors (a.k.a. Switches)
Add an indicator LED to Pin 13

This is just like our 1^{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***

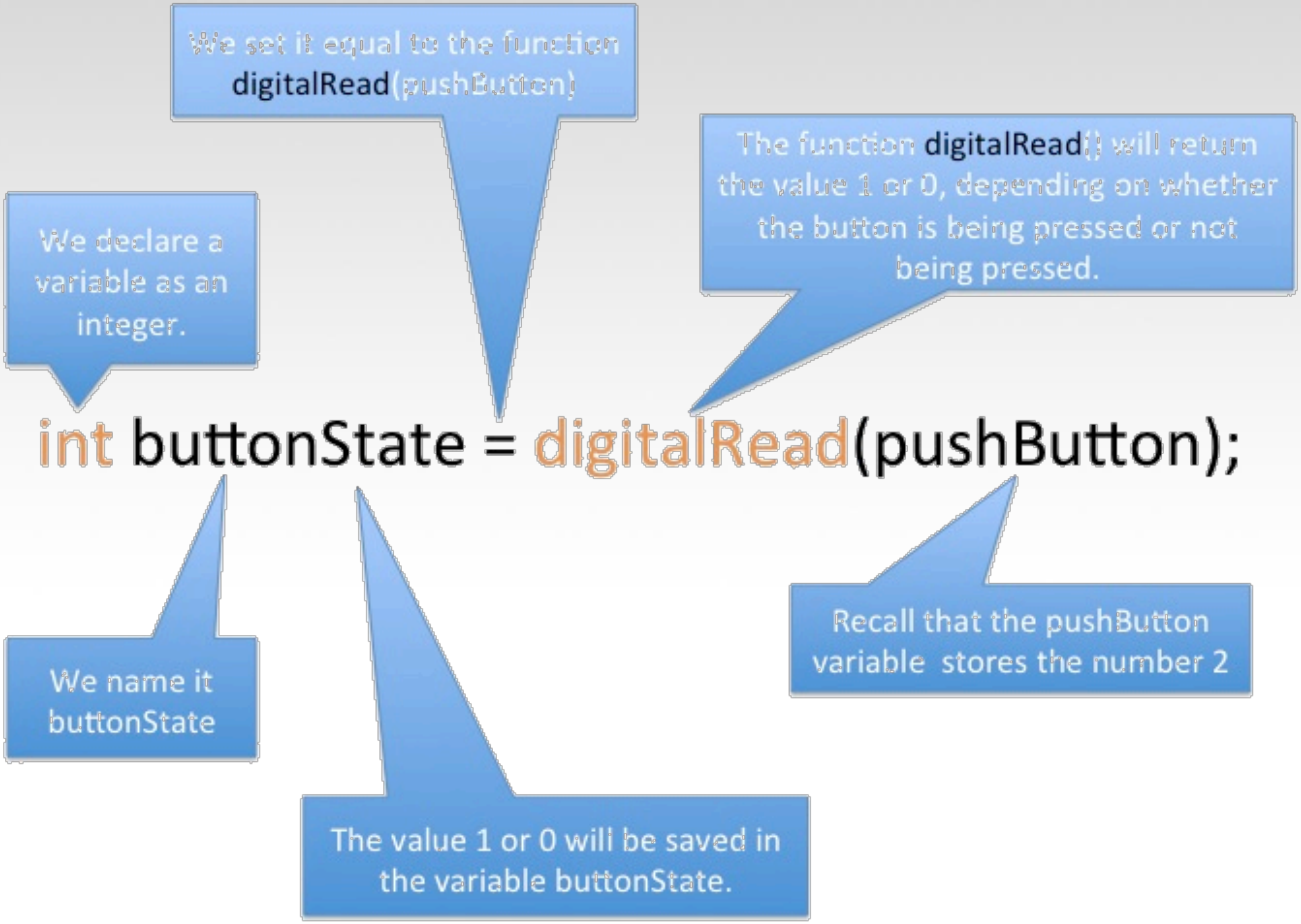
- To get a digital reading:

  ```
  int buttonState = digitalRead (pinNumber);
  ```

- Digital Input values are only **HIGH** (On) or **LOW** (Off)

Digital Sensors

- Digital sensors are more 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)

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

We declare a variable as an integer.

```
int buttonState = digitalRead(pushButton);
```

We name it buttonState

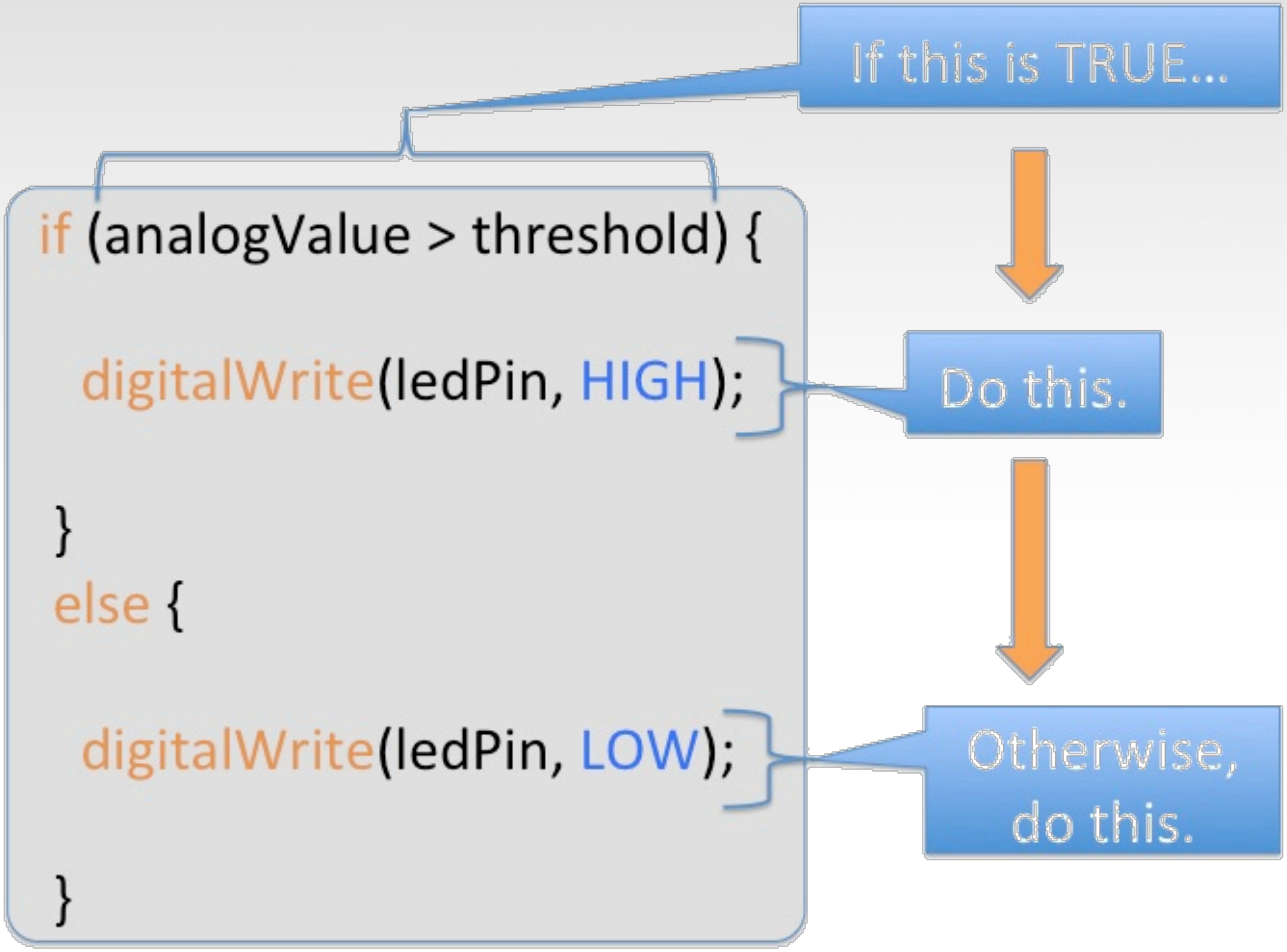
The value 1 or 0 will be saved in the variable buttonState.

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);

}
```

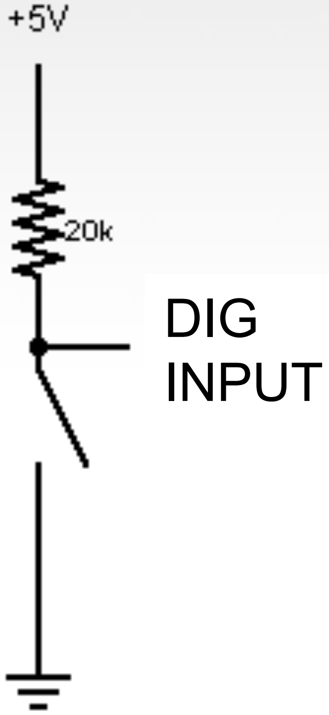
If this is TRUE...

Do this.

Otherwise, do this.

Programming: Conditional Statements
`if()`

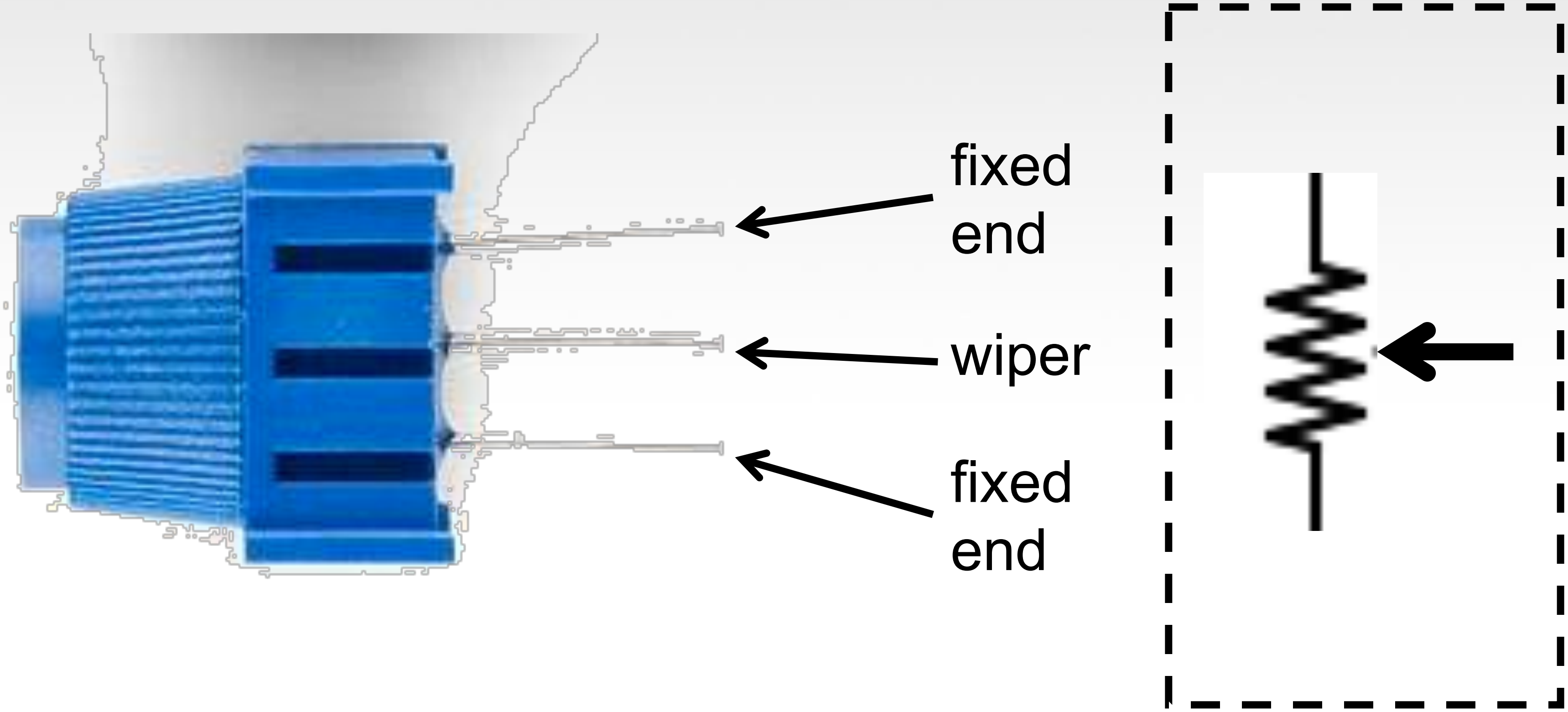
```
void loop()
{
  int buttonState = digitalRead(5);
  if(buttonState == LOW)
  {  // do something
  }
  else
  {  // do something else
  }
}
```



Boolean Operators

<Boolean>	Description
( ) == ( )	is equal?
( ) != ( )	is not equal?
( ) > ( )	greater than
( ) >= ( )	greater than or equal
( ) < ( )	less than
( ) <= ( )	less than or equal

Trimpot (Potentiometer)
Variable Resistor



fixed end

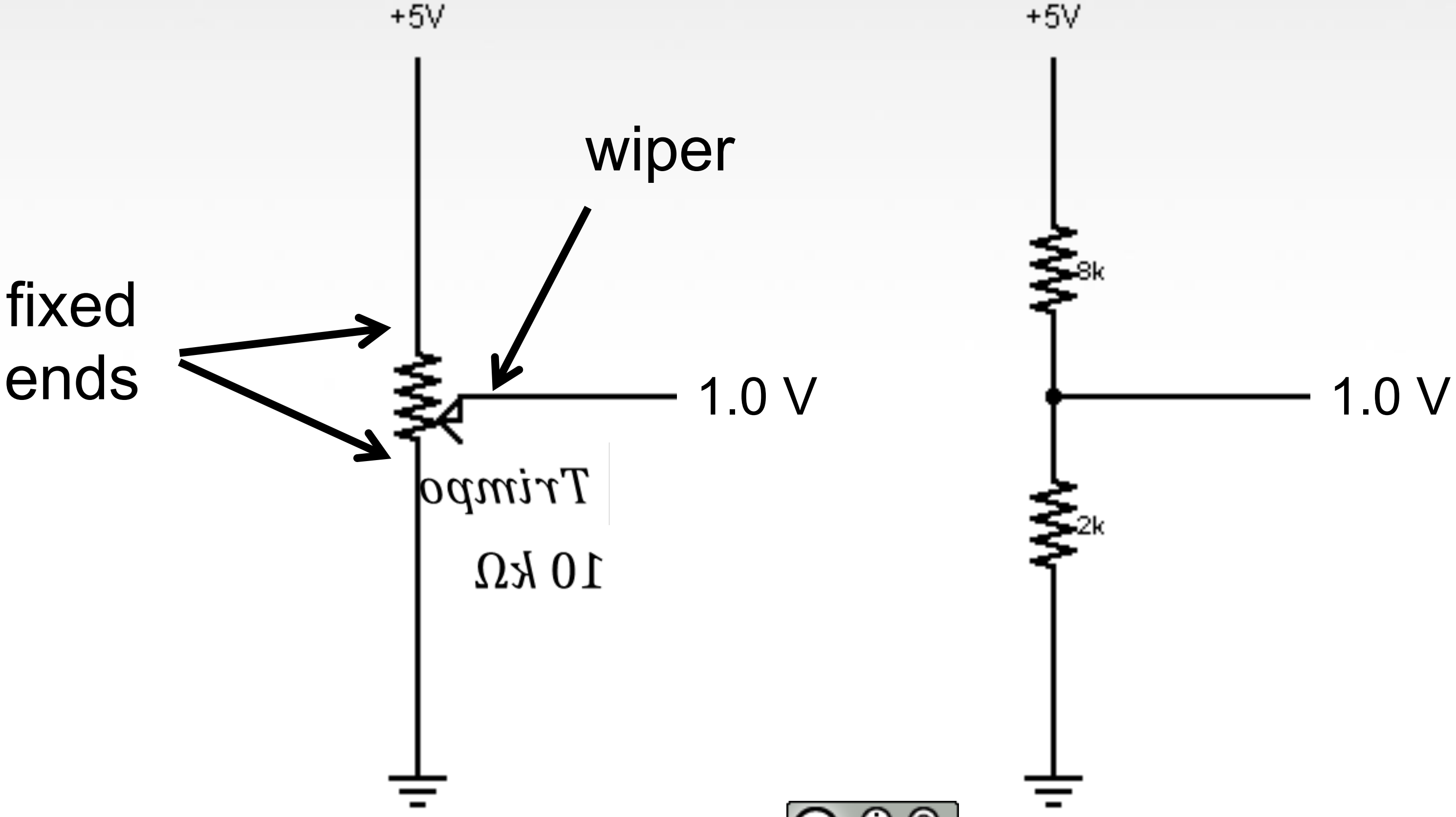
wiper

fixed end

Analog Sensors

3 Pin Potentiometer = var. resistor (circuit)

a.k.a. Voltage Divider Circuit



fixed ends

wiper

1.0 V

1.0 V

Ohms Law… (just the basics)
Actually, this is the “voltage divider”

$$V_{R1} = V_{CC} \cdot \left(\frac{R_1}{\mathrm{R}_{Total}} \right)$$

$$V_{R2} = V_{CC} \cdot \left(\frac{R_2}{\mathrm{R}_{Total}} \right)$$

$$R_{Total} = R_1 + R_2$$

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

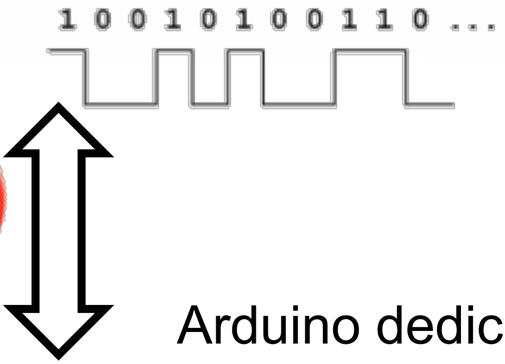
Ex:

```
int sensorValue = analogRead(A0);
```

Using Serial Communication

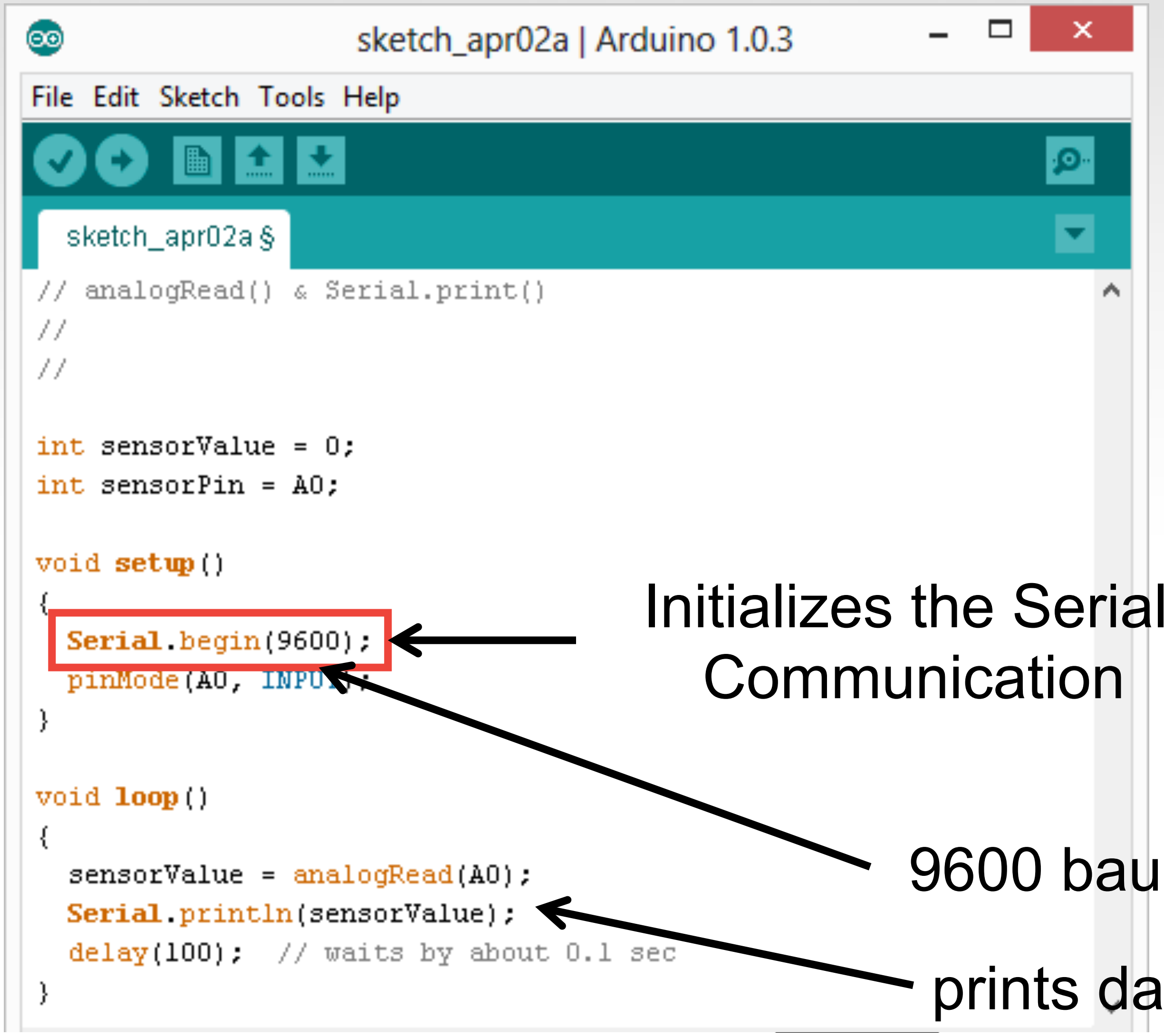
Method used to transfer data between two devices.

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()



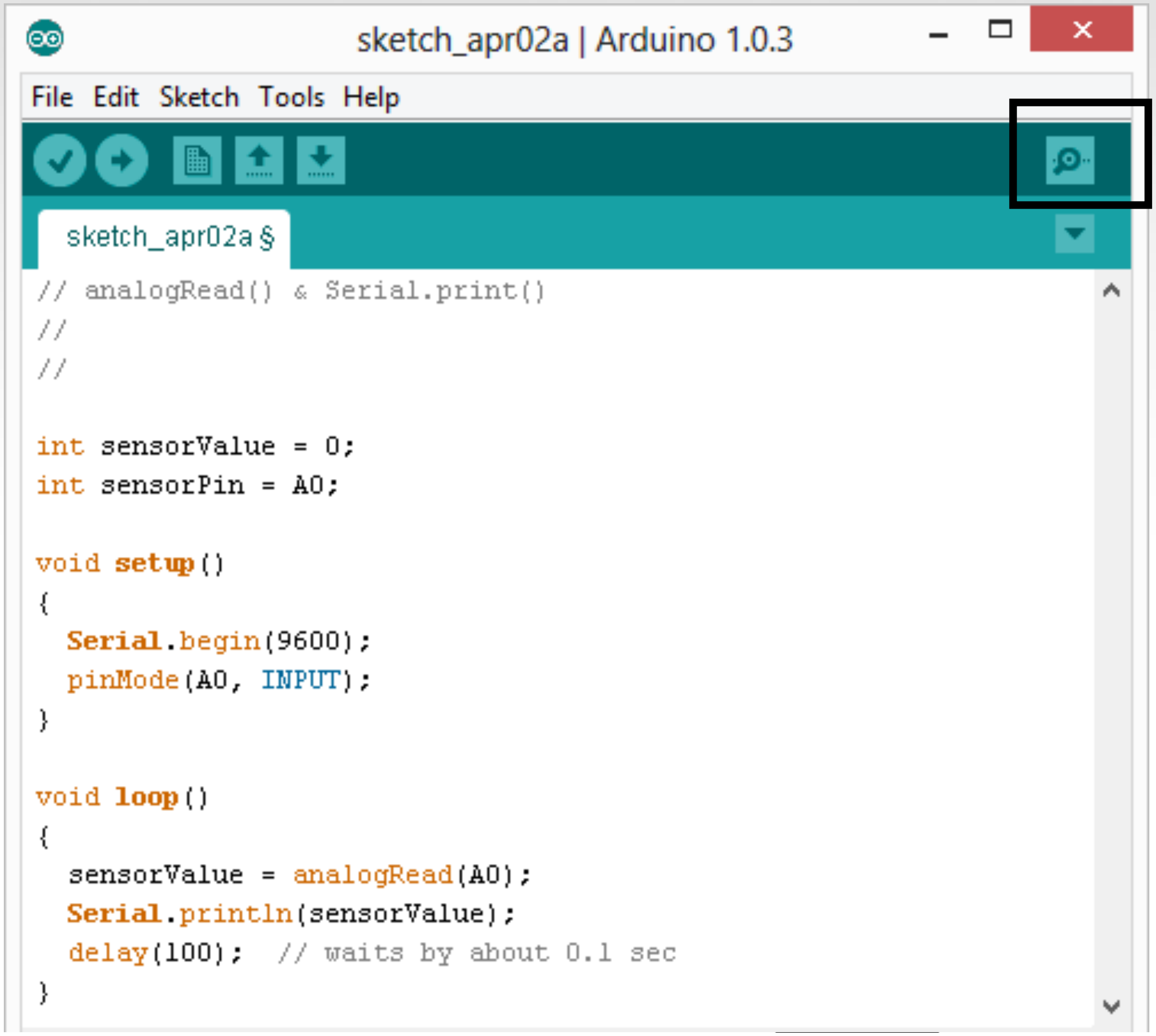
```
// analogRead() & Serial.print()
//
//

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

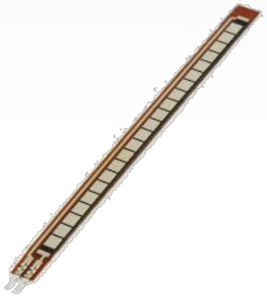
Serial Monitor & analogRead()

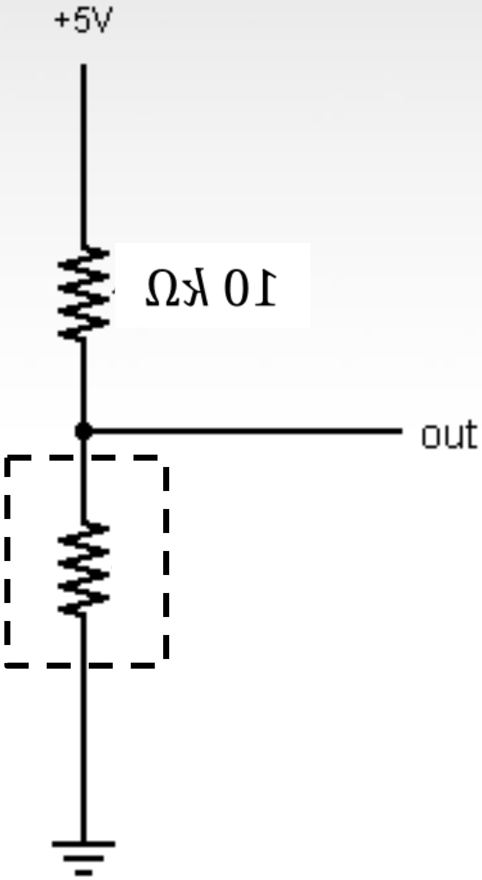


```
// analogRead() & Serial.print()
//
//

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
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 = __________

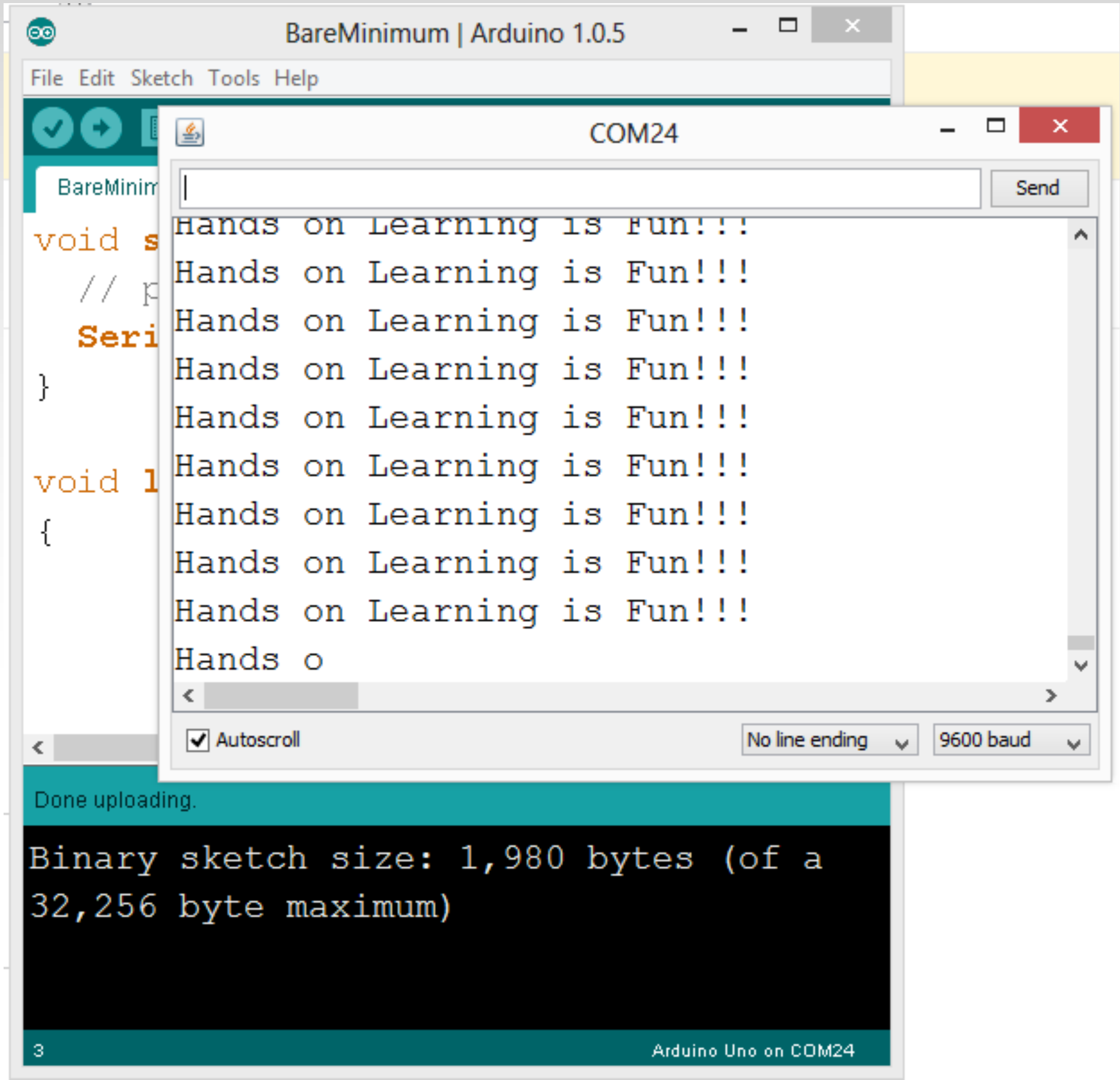
Analog Sensors

Examples:

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!!!”) ;

}
```



```
void s
  // p
  Seri
}

void l
{
```

```
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
```

☑ Autoscroll   No line ending   9600 baud

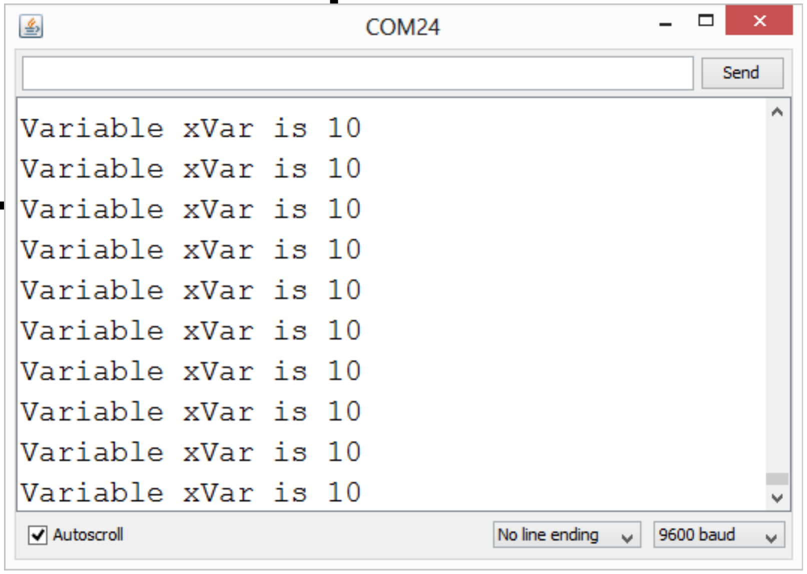
Done uploading.

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

3   Arduino Uno on COM24

Serial Communication: Serial Debugging

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

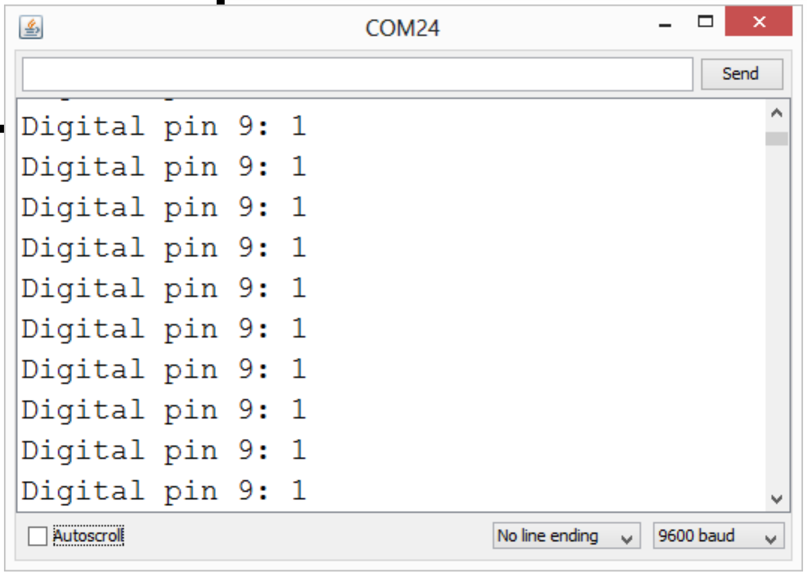


```
Variable xVar is 10
Variable xVar is 10
Variable xVar is 10
Variable xVar is 10
Variable xVar is 10
Variable xVar is 10
Variable xVar is 10
Variable xVar is 10
Variable xVar is 10
Variable xVar is 10
```

☑ Autoscroll   No line ending   9600 baud

Serial Communication:
Serial Troubleshooting

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

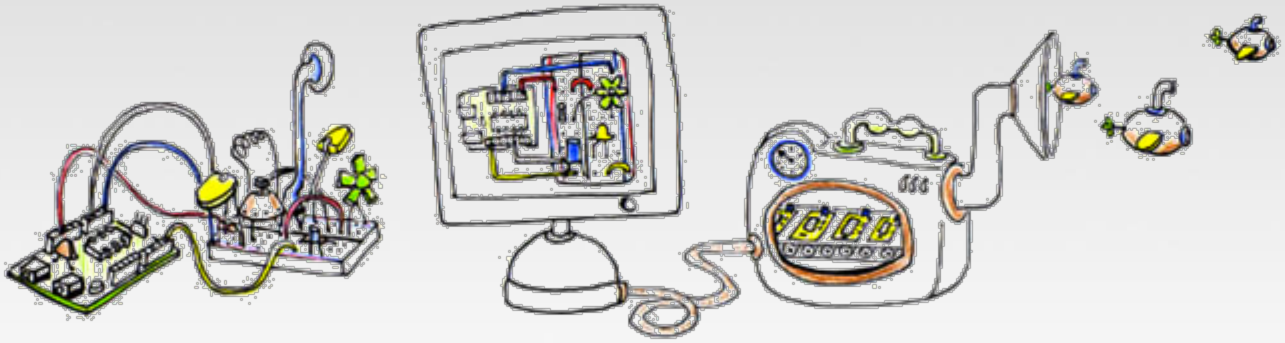


```
Digital pin 9: 1
Digital pin 9: 1
Digital pin 9: 1
Digital pin 9: 1
Digital pin 9: 1
Digital pin 9: 1
Digital pin 9: 1
Digital pin 9: 1
Digital pin 9: 1
Digital pin 9: 1
```

No line ending

9600 baud

FRITZING



Virtual Electrical Prototyping Project

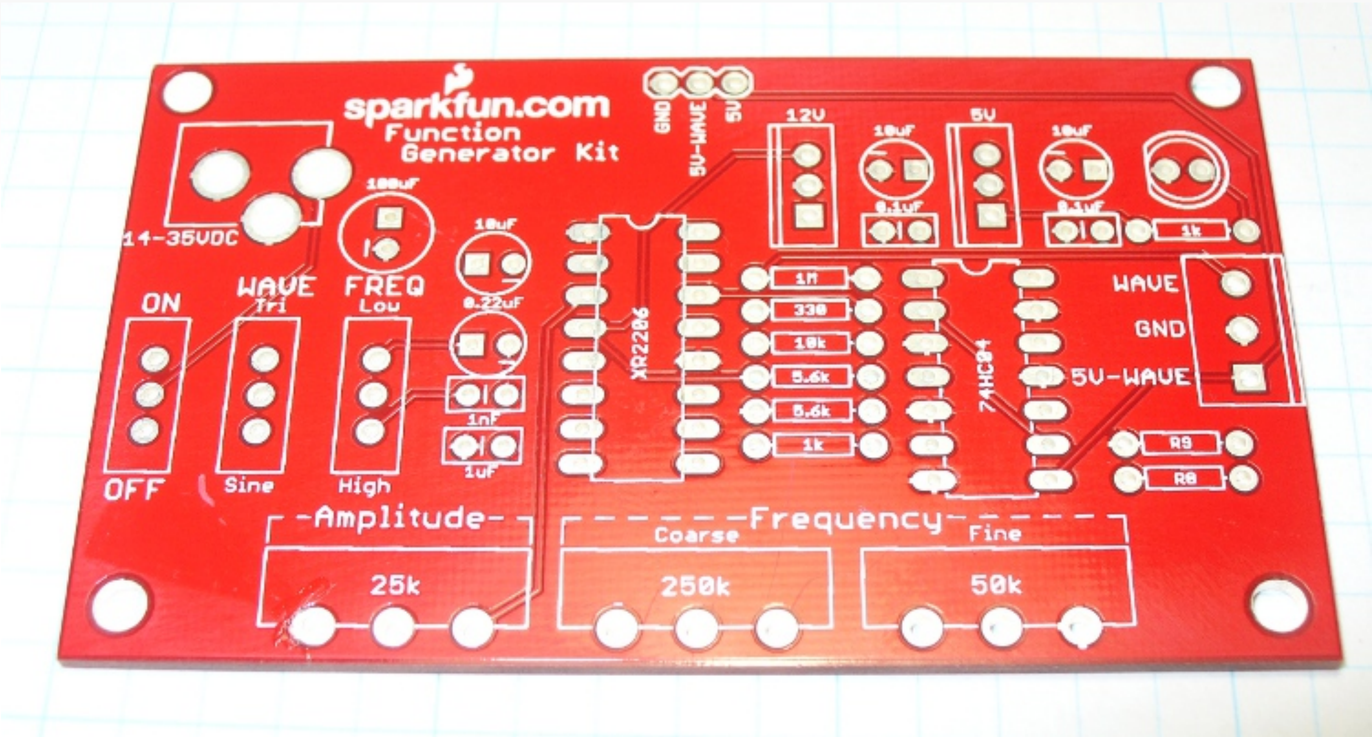
started in 2007 by the Interaction Design Lab

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?

sparkfun™ ELECTRONICS

www.sparkfun.com
6175 Longbow Drive, Suite 200
Boulder, Colorado 80301