An Introduction the Arduino

Peter Beens,
DSBN 2016
@pbeens

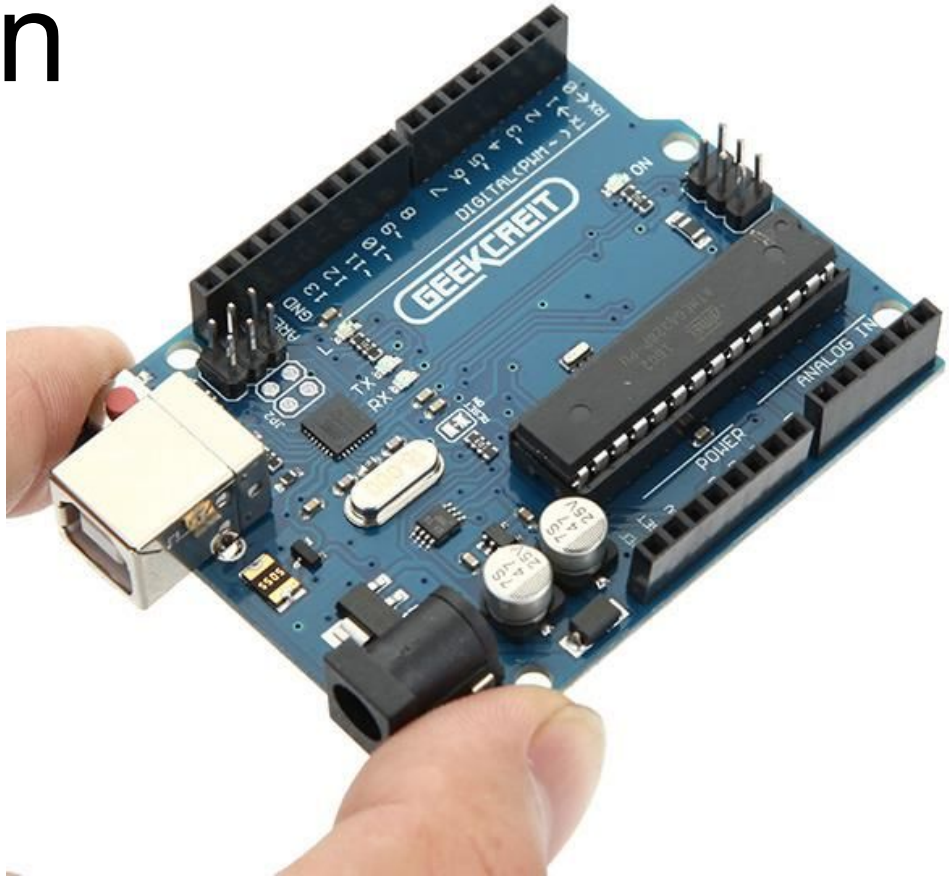


Image courtesy of http://goo.gl/8nJroG

URL for this presentation: https://goo.gl/97LryV

https://www.arduino.cc/

ARDUINO 1.6.12

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install

Windows app Get

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM (experimental)

Release Notes
Source Code
Checksums (sha512)

- Arduino is an open-source electronics platform based on easy-to-use hardware and software.

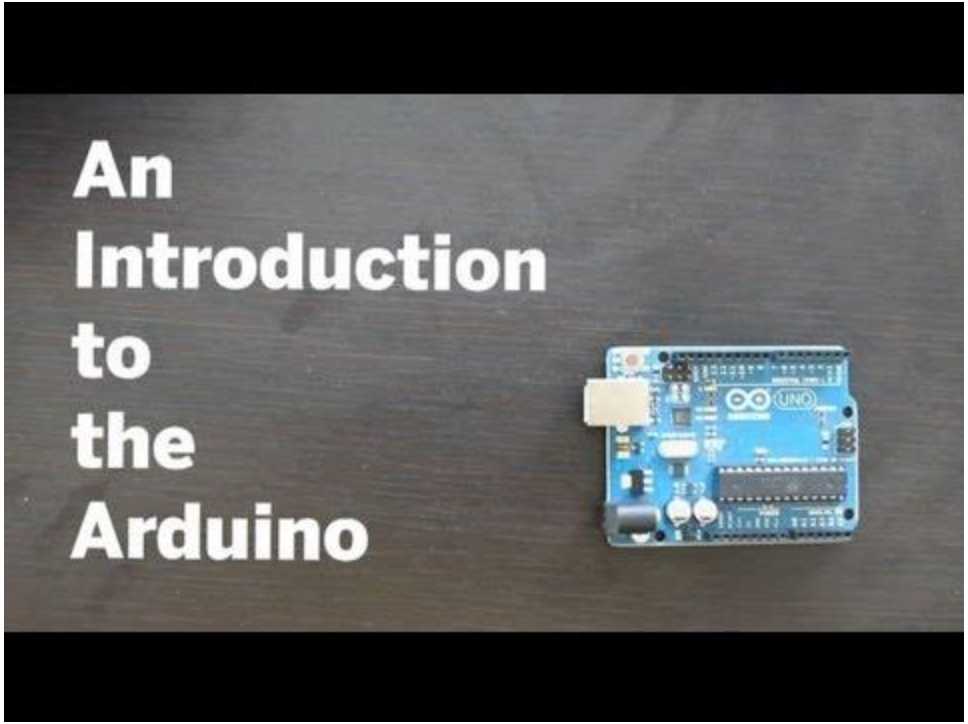
Credit: https://www.arduino.cc/en/Guide/Introduction

- Can read inputs (e.g., light sensor, button, a Tweet) and provide outputs (e.g., activate a motor, turn on an LED, make sounds, send a Tweet, put characters on an LCD display)

- Can be used to learn about electronics *and* programming (ideal for Computer Technology or Maker Clubs and a great way to make CS more fun)
- Interesting fact: Arduino programs are called "**sketches**", not programs

Credit: https://en.wikipedia.org/wiki/Arduino

Intro Videos



An Introduction to the Arduino (from Make Magazine) (4:25)
https://www.youtube.com/watch?v=CqrQmQqpHXc

Massimo Banzi: How Arduino is open-sourcing imagination (15:46)
https://www.youtube.com/watch?v=UoBUXOOdLXY

“Tutorial 01 for Arduino: Getting Acquainted with Arduino” (14:31)
https://www.youtube.com/watch?v=fCxzA9_kg6s

Our First Program!

“Empty Program”

This program will allow you to:

- See the basic functions of an Arduino program
- Learn how to configure the editor (IDE) for the Arduino
- Learn how to check the syntax of a program
- Learn how to upload the program into the Arduino

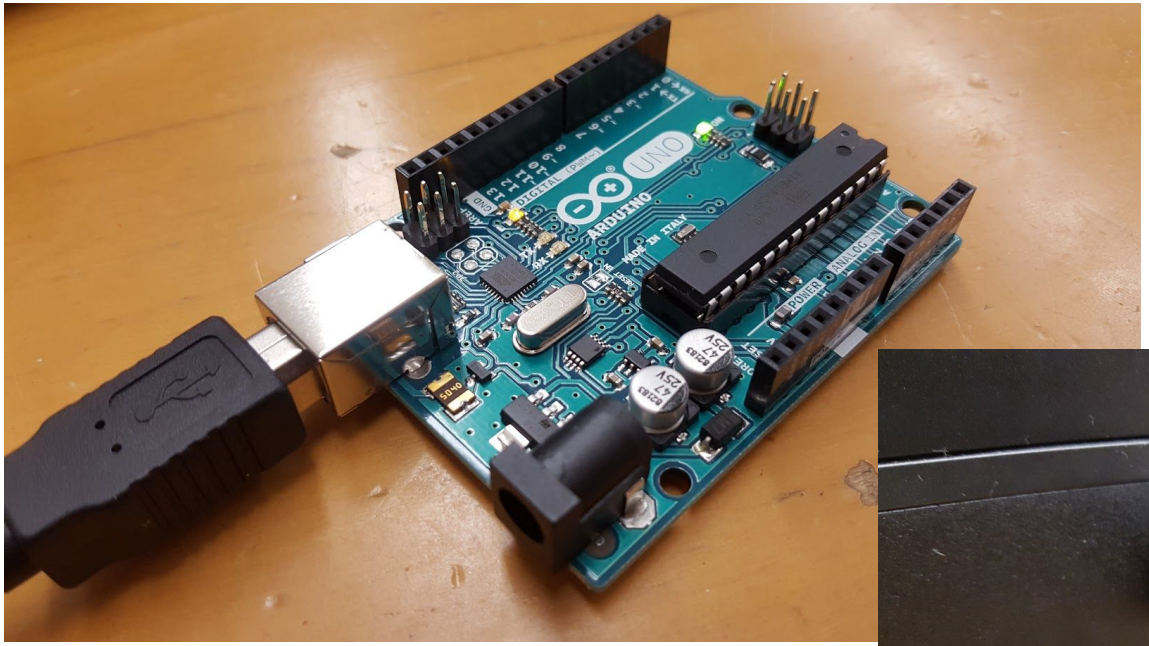
Save as “Empty”

```
void setup() {

}

void loop() {

}
```

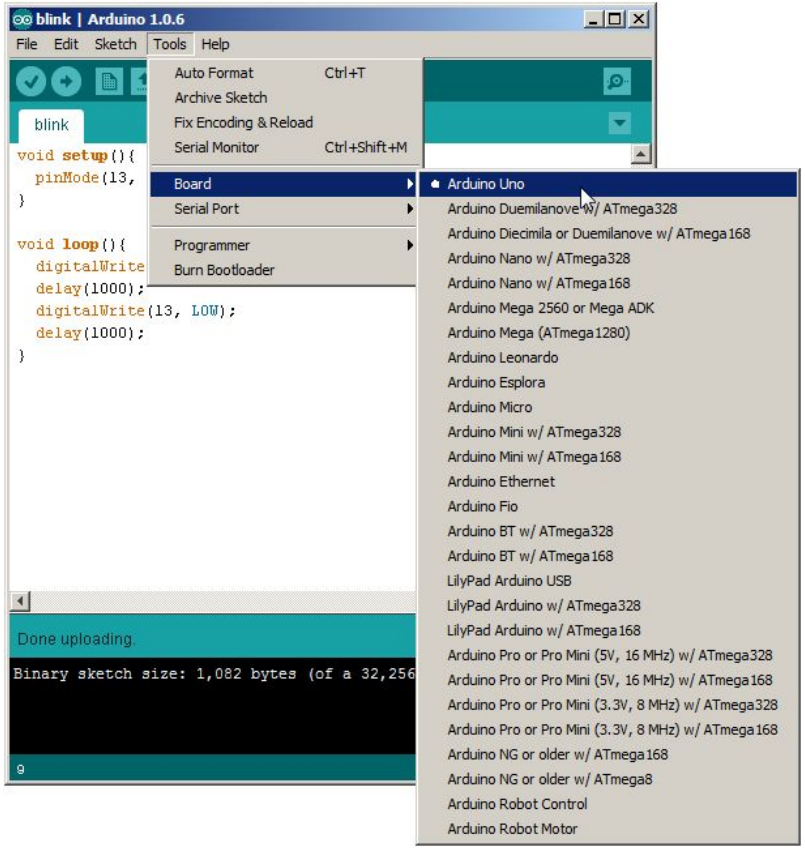
Note!

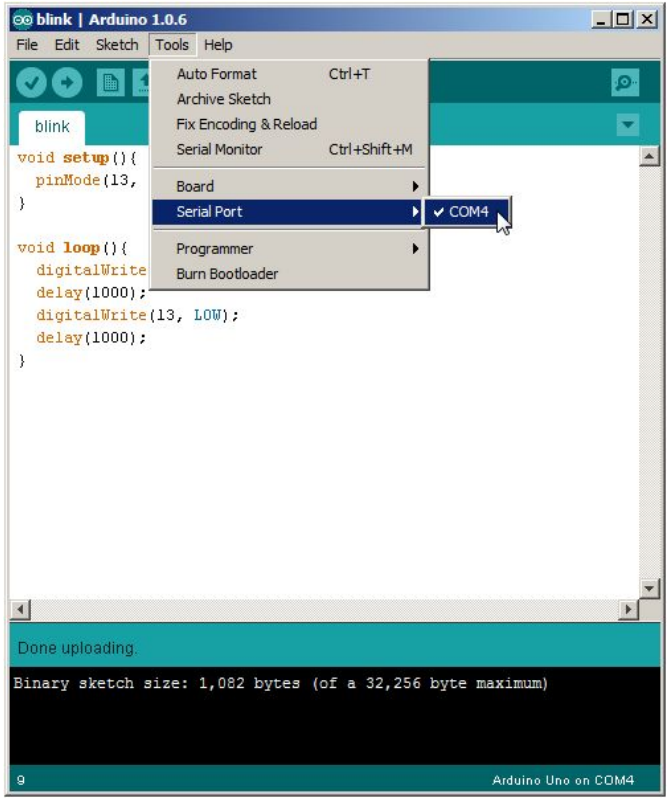
Save the code for all your programs in a folder named “Arduino” in your home folder.

Connect Your Arduino...

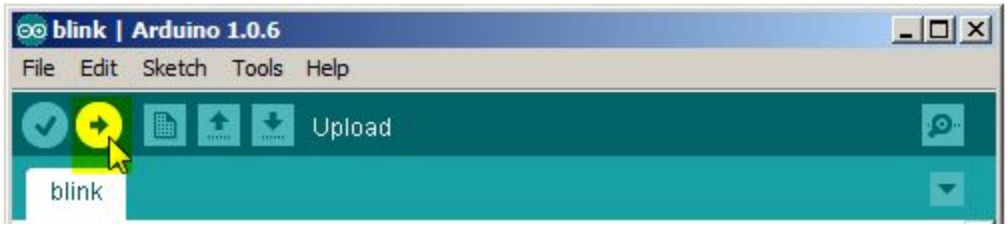


Before Uploading...





Upload!



Watch the LEDs on the Arduino while the program is uploading...

Program #2: “Blink”

“Blink”

This program turns on an LED on for one second, then off for one second, repeatedly.

Most Arduinos have an on-board LED you can control. On the Uno and Leonardo, it is attached to digital pin 13.

Credit(s): http://labs.arduino.org/STARTER+KIT+-+GET+TO+KNOW+YOUR+TOOLS

Save as “Blink”

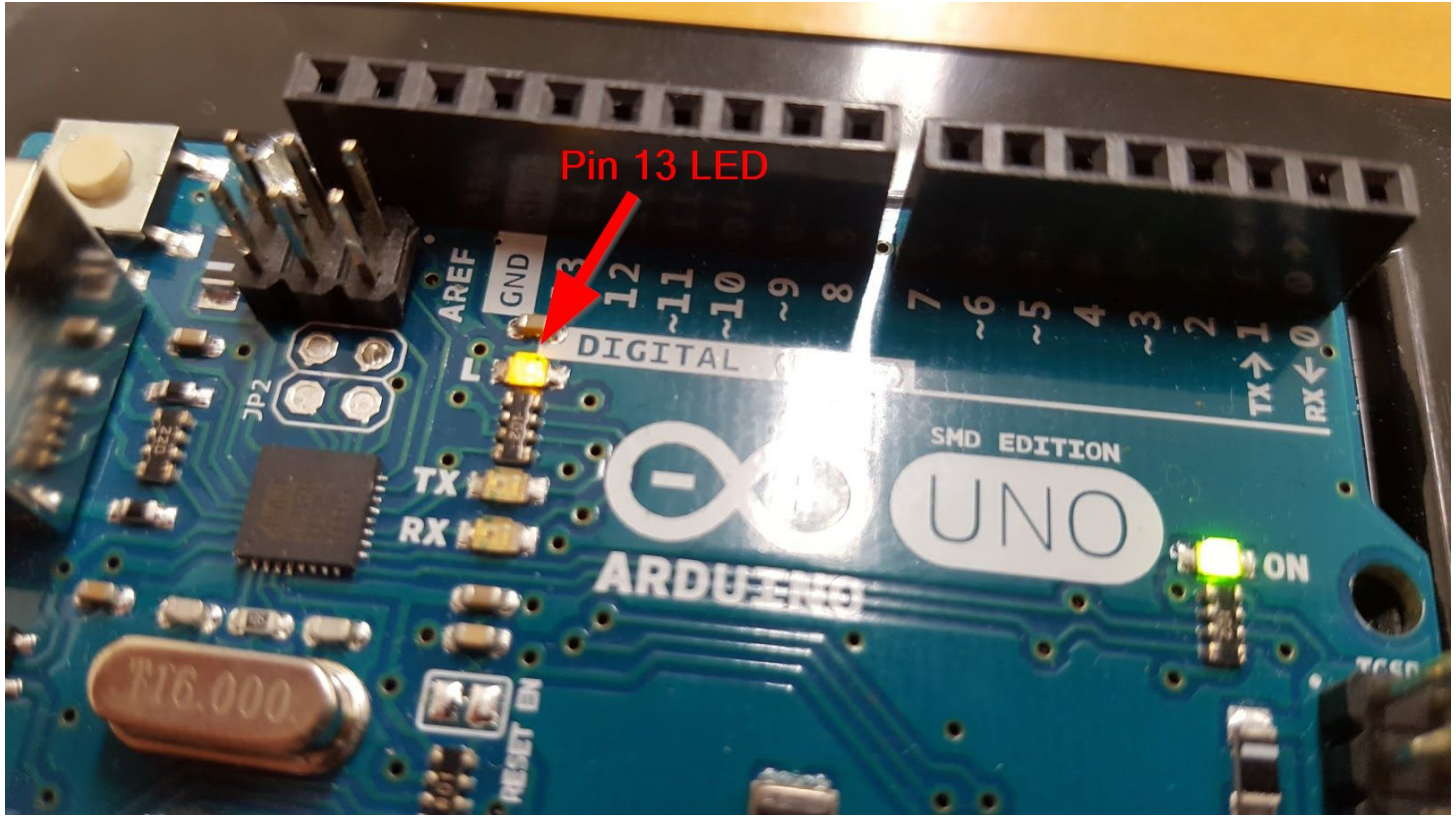
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off
  delay(1000);              // wait for a second
}
```

Note!

The highlighted comments are not necessary for your program, but it’s good programming practice for you to include lots of them to help document your program.

Credit(s): http://labs.arduino.org/STARTER+KIT+-+GET+TO+KNOW+YOUR+TOOLS

Did it Work?!!


Pin 13 LED
AREF
GND
DIGITAL
TX
RX
SMD EDITION
UNO
ON
JP2
RESET EN

Some Arduino Programming Basics

Note!

The #1 rule in programming is:

Save Early, Save Often!

Variables

A variable is a place for storing a piece of data. It has a name, a type, and a value.

Example:

```
int ledPin = 13; // LED connected to digital pin 13
```


Credit(s): http://labs.arduino.org/What+is+a+Sketch

Variables (cont'd)

This example declares a variable with the name *ledPin*, the type *int*, and an initial value of *13*.

```
int ledPin = 13; // LED connected to digital pin 13
```

It's being used to indicate which Arduino pin the LED is connected to.

Every time the variable "ledPin" appears in the code, its value will be retrieved (i.e. the value 13).

Credit(s): http://labs.arduino.org/What+is+a+Sketch

Program #3: Blink with a Variable

Task:

Modify your Blink program by inserting the ledPin declaration (as shown below) and substituting “13” with “ledPin” everywhere in your program.

```
int ledPin = 13; // LED connected to digital pin 13
```

The declaration line should go above the setup() function.

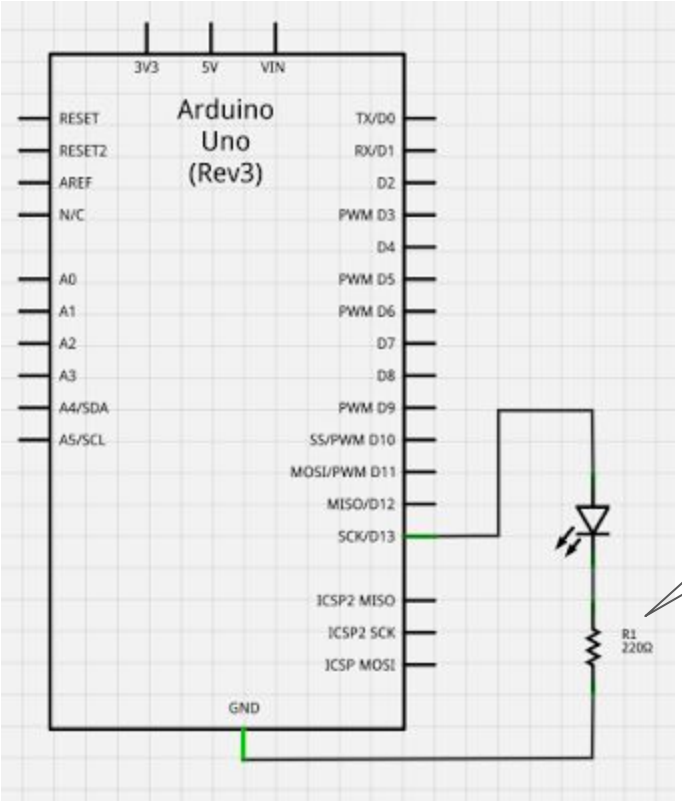
Don’t forget to save your program!

Your code should look like this:

```
int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Let’s Hook Up an External LED!

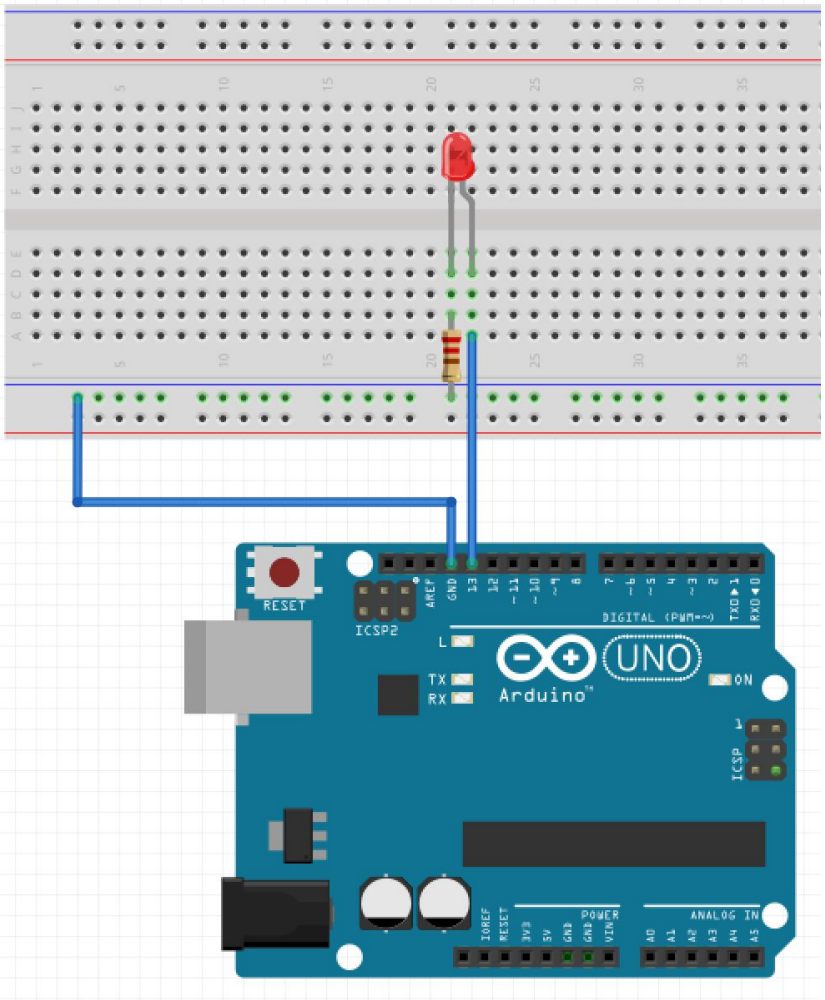
Schematic



The resistor is needed to protect the LED. It limits the current to a safe amount and drops the voltage across the LED to about 2 volts.

Image source: http://www2.beens.org/computer-technology/arduino/activities/blinking-led

Pictorial



Note!

Unplug your Arduino while hooking up the electronic components. Get the circuit checked before applying power.

Image source: http://www2.beens.org/computer-technology/arduino/activities/blinking-led

Program #4: A Traffic Light

Let’s Build a Traffic Light!

Save your Blink program as “TrafficLight”.

Modify “ledPin” to be “redLED” throughout your program.

Add two variables “yellowLED” and “greenLED”, using pins 12 and 11, respectively.

Modify your program and your circuitry to simulate a traffic light. There should be no delay between each LED being lit.

Program #5, 6, and 7: 8 LEDs

8 LED Circuit and Programs

First, wire up 8 LEDs. Remember that each LED requires its own resistor. For convenience, use pins 6-13. (Hint: use pin 6 for LED 1, 7 for LED 2, etc.)

Program 5: Simply flash all 8 LEDs on and off.

Program 6: Light the LEDs in sequence.

Program 7: Create a binary counter, with each LED representing one bit in a byte.

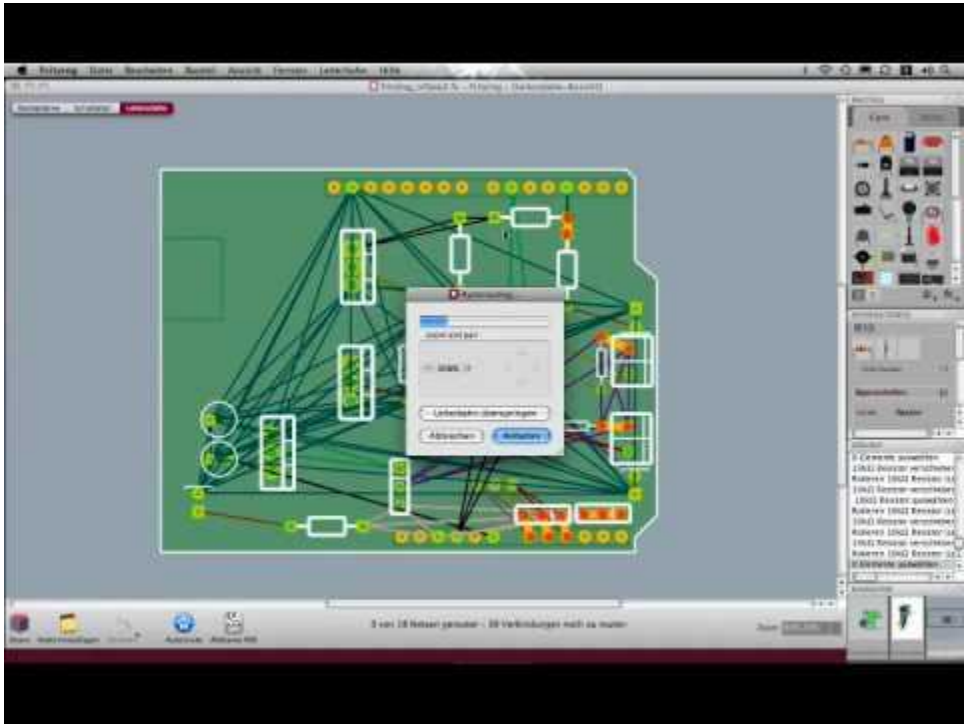
Supplemental Material

Fritzing

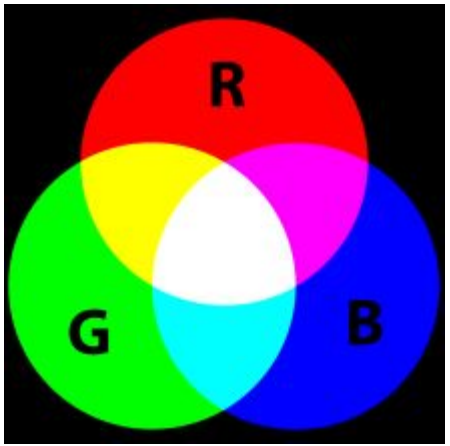
“Fritzing is an open-source hardware initiative that makes electronics accessible as a creative material for anyone. We offer a software tool, a community website and services in the spirit of Processing and Arduino, fostering a creative ecosystem that allows users to document their prototypes, share them with others, teach electronics in a classroom, and layout and manufacture professional PCBs.”

http://fritzing.org

Fritzing cont’d



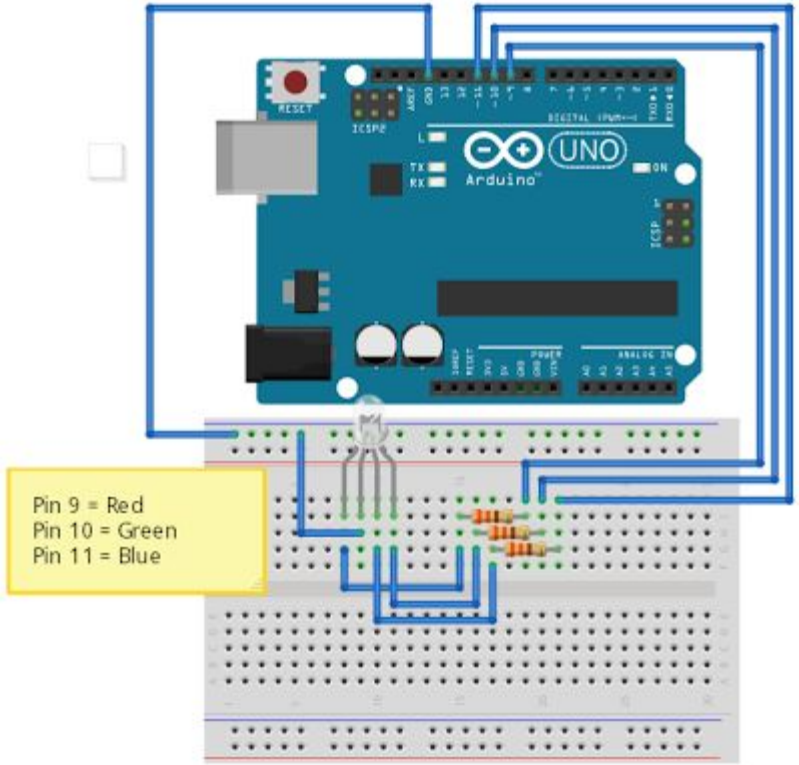
From a Traffic Light to RGB!



Assignment

Write a program that loops through the following colours, with each colour being displayed for 500 mS:

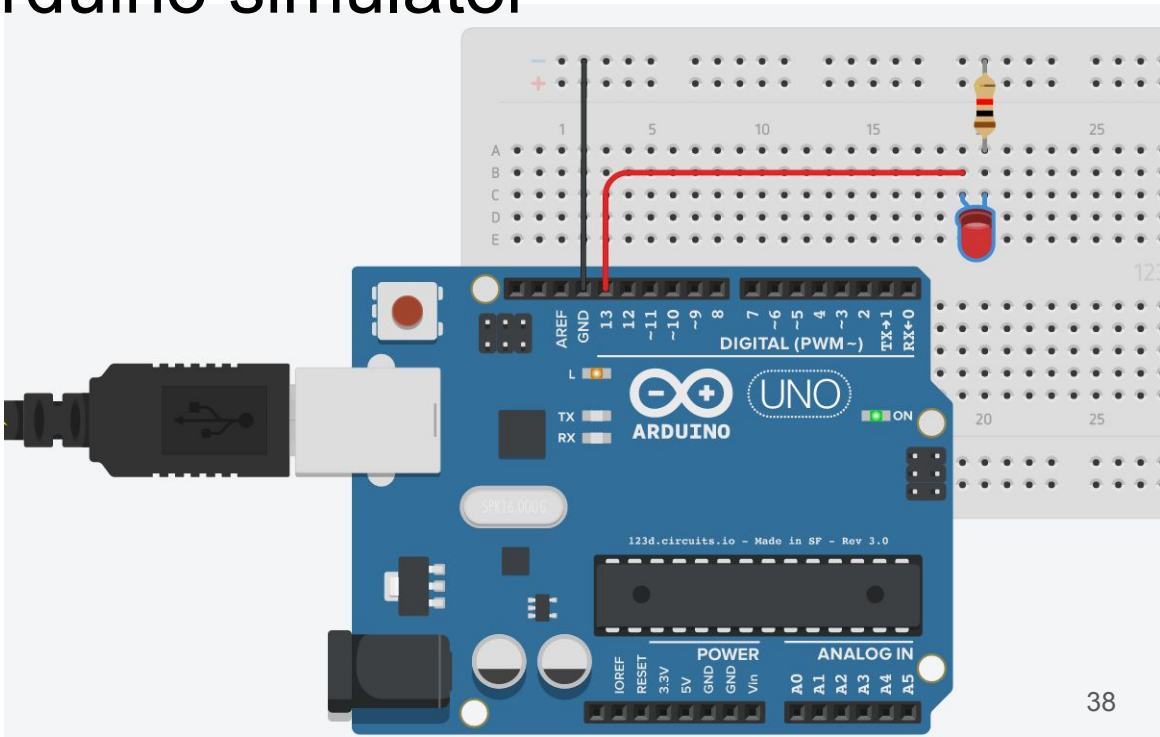
- green
- yellow
- red
- magenta
- blue
- cyan
- white



Frtizing diagram source: http://www2.beens.org/computer-technology/arduino/activities/rgb-leds
RGB image source: http://upload.wikimedia.org/wikipedia/commons/thumb/c/c2/AdditiveColor.svg/220px-AdditiveColor.svg.png

Further Exploration:

- Electronics and Arduino simulator https://circuits.io/



Further Exploration (cont'd):

- Arduino Tutorials
  http://www.arduino.org (Learning tab)
- Arduino Reference
  https://www.arduino.cc/en/Reference

My Electronics Presentation

https://goo.gl/cfO08H
(on GitHub)

Thanks!

Follow Up questions or comments?
Email me at pbeens@gmail.com.