

Lab 3: Arduino Basics

Prepared by senior-lecturer,
B.B. Imankulova

Goals of Lab #3

- Learn how the programming takes place
- Exercises about:
 - Learn about Arduino electronics
 - Learn about LEDs
 - Learn how to code

Intro to Arduino

Installing the IDE



Windows Installation Process

Go to the web address below to access the instructions for installations on a Windows-based computer.

[*http://arduino.cc/en/Guide/Windows*](http://arduino.cc/en/Guide/Windows)



Macintosh OS X Installation Process

Macs do not require you to install drivers. Enter the following URL if you have questions. Otherwise proceed to next page.

[*http://arduino.cc/en/Guide/MacOSX*](http://arduino.cc/en/Guide/MacOSX)



Linux: 32 bit / 64 bit, Installation Process

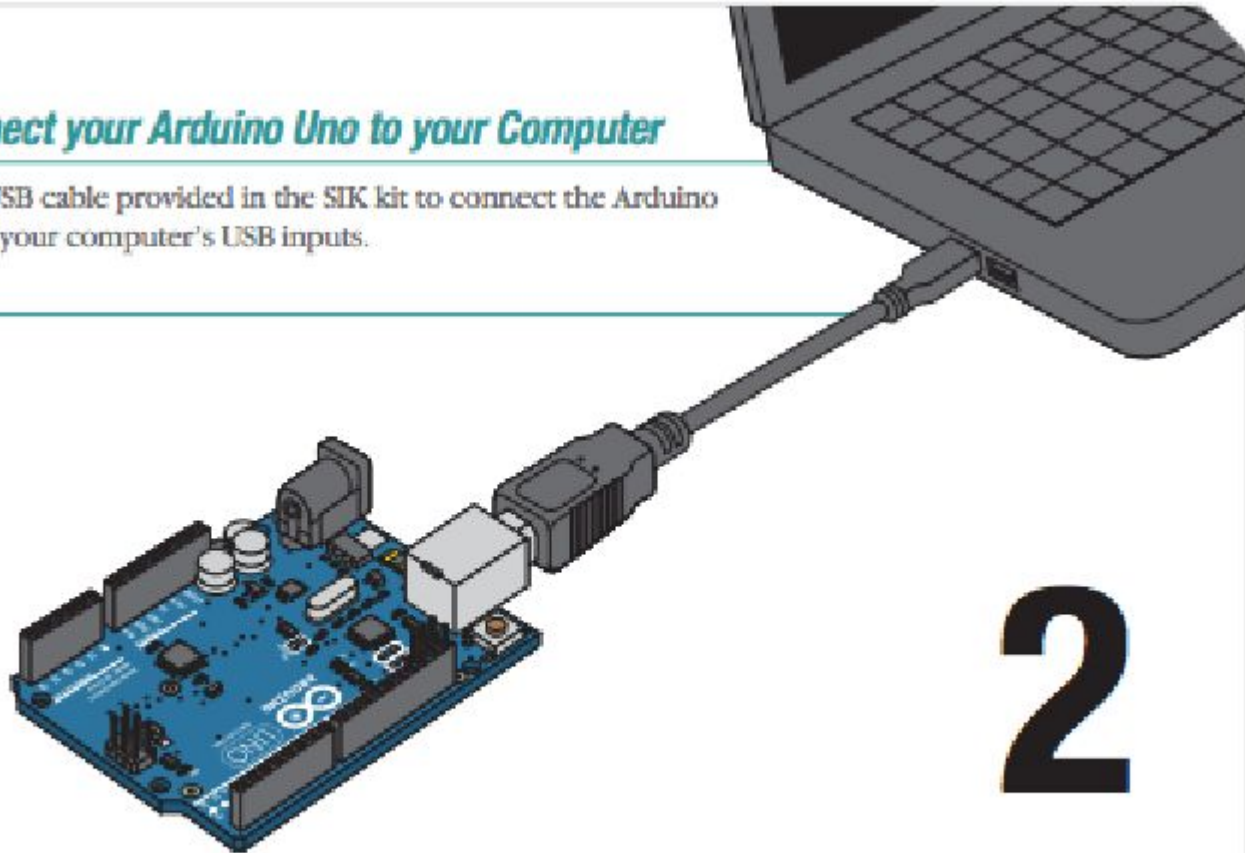
Go to the web address below to access the instructions for installations on a Linux-based computer.

[*http://www.arduino.cc/playground/Learning/Linux*](http://www.arduino.cc/playground/Learning/Linux)

Connect your Arduino

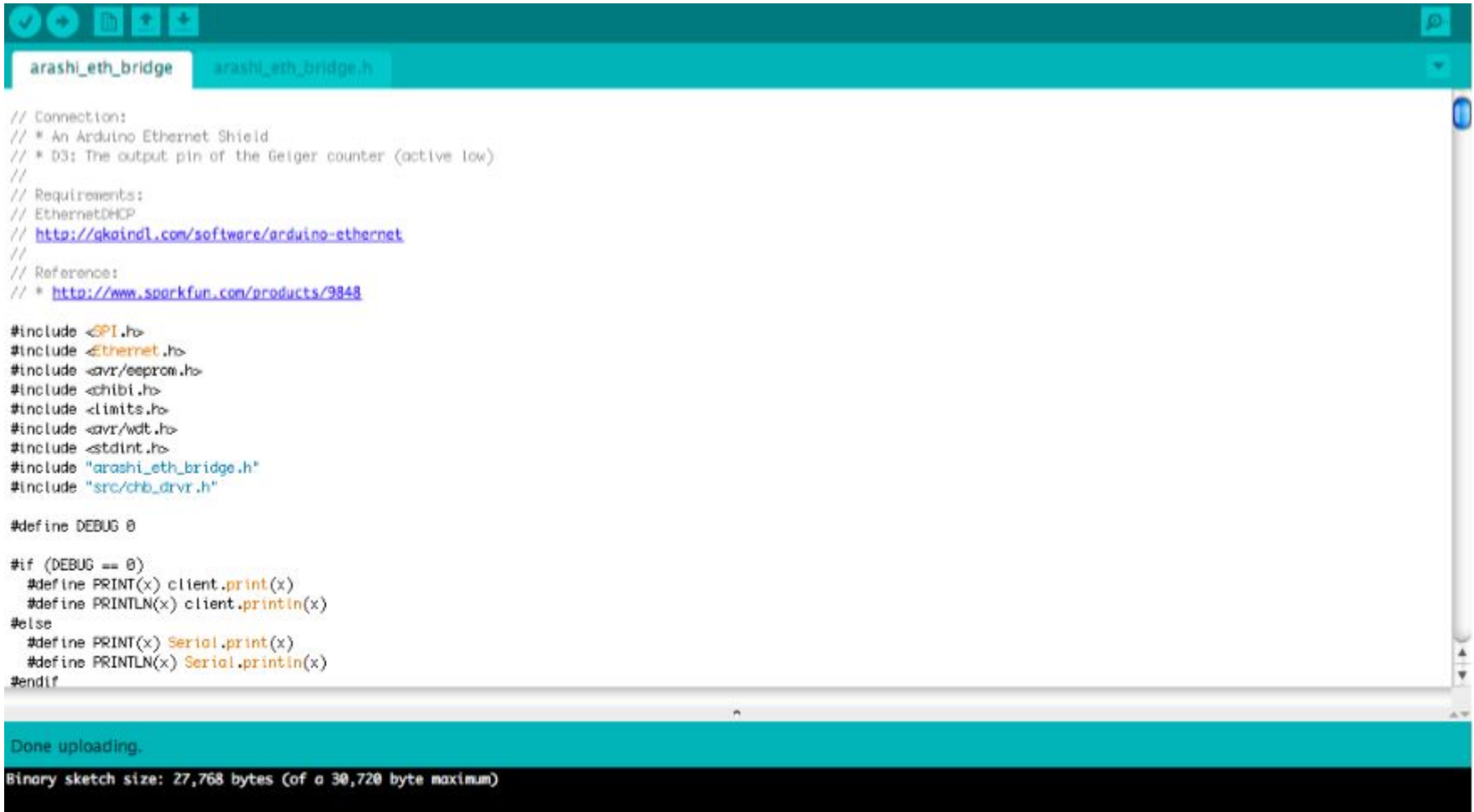
// *Connect your Arduino Uno to your Computer*

Use the USB cable provided in the SIK kit to connect the Arduino to one of your computer's USB inputs.



Note: we will use a slightly different board, which you connect through one additional small board, the programmer

The Arduino IDE



```
// Connection:
// * An Arduino Ethernet Shield
// * D3: The output pin of the Geiger counter (active low)
//
// Requirements:
// EthernetDHCP
// http://gkaindl.com/software/arduino-ethernet
//
// Reference:
// * http://www.sparkfun.com/products/9848

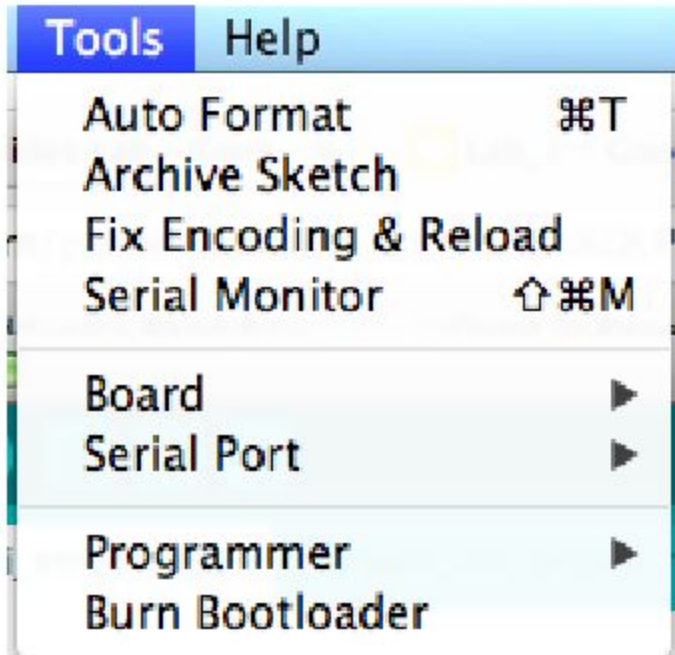
#include <SPI.h>
#include <Ethernet.h>
#include <avr/eeprom.h>
#include <chibi.h>
#include <limits.h>
#include <avr/wdt.h>
#include <stdint.h>
#include "arashi_eth_bridge.h"
#include "src/chb_drvr.h"

#define DEBUG 0

#if (DEBUG == 0)
  #define PRINT(x) client.print(x)
  #define PRINTLN(x) client.println(x)
#else
  #define PRINT(x) Serial.print(x)
  #define PRINTLN(x) Serial.println(x)
#endif
#endif

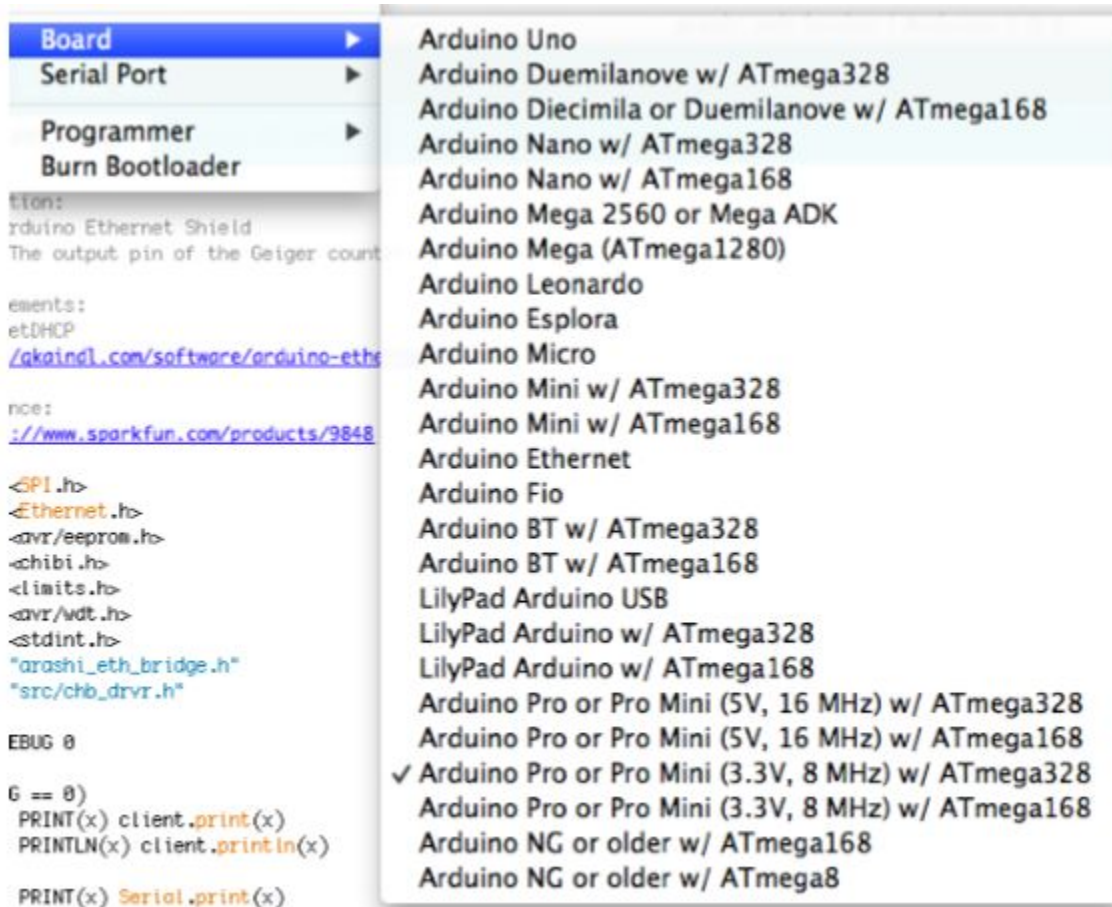
Done uploading.
Binary sketch size: 27,768 bytes (of a 30,720 byte maximum)
```

1: Select serial port



Select the Serial Port:
tty/USBx on Linux,
COMx on Windows

2: Select Arduino model



Select Board:
Arduino Pro
3.3V, ATmega328

Programming an Arduino

From the File menu, choose Open and select the code you want to open.

The source code will appear in the IDE window.

Programming workflow

1. Opening



Open

2. Verifying



Verify

3. Uploading



Upload

Programming an Arduino

Click on the upload button and wait until the code has been compiled and uploaded.

At the end you will see in the bottom right corner:

Done uploading.

Programming an Arduino

This is the template of a basic Arduino program:

```
void setup()
{
  Initialize variables, open USB, open WiFi, etc
}
```

SETUP
(once)

```
void loop()
{
  Perform some action

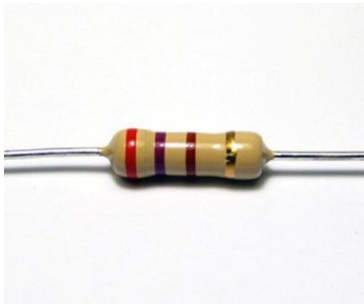
  Wait for a certain number of msecs or wait for an alarm
}
```

LOOP
(forever)

Pre-lab #3

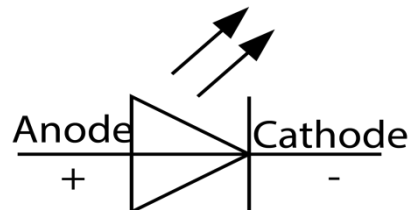
Resistor

Resistors "resist" the *flow of electricity*, or current. The orientation in the circuit does not matter; they have no *polarity*. The resistance in Ohms can be determined by the colors of the bands. [Here](#) is a chart to see how the bands convert to the resistor's value.



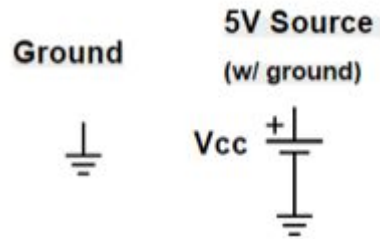
LED

Light Emitting Diodes (LEDs) only let current pass in one direction, so we say they have polarity; they have a positive and negative lead. The longer lead is the anode, and it is positive. The shorter lead is the cathode, which is negative. Usually the negative side of the plastic part of the LED will be flat.



Voltage Sources

The Arduino has two constant voltage sources that we will use. The first is called "ground" which is our reference for 0 volts. It has the schematic symbol shown above. The second reference we will use is called V_{cc} and we will represent it as a voltage source, or battery, with one side attached to ground, as shown. For the Arduino, V_{cc} is always at 5 Volts.



Please prepare for the Post lab using lecture notes.

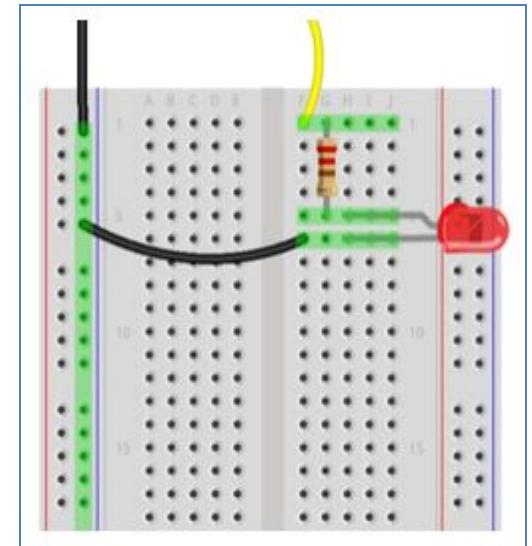
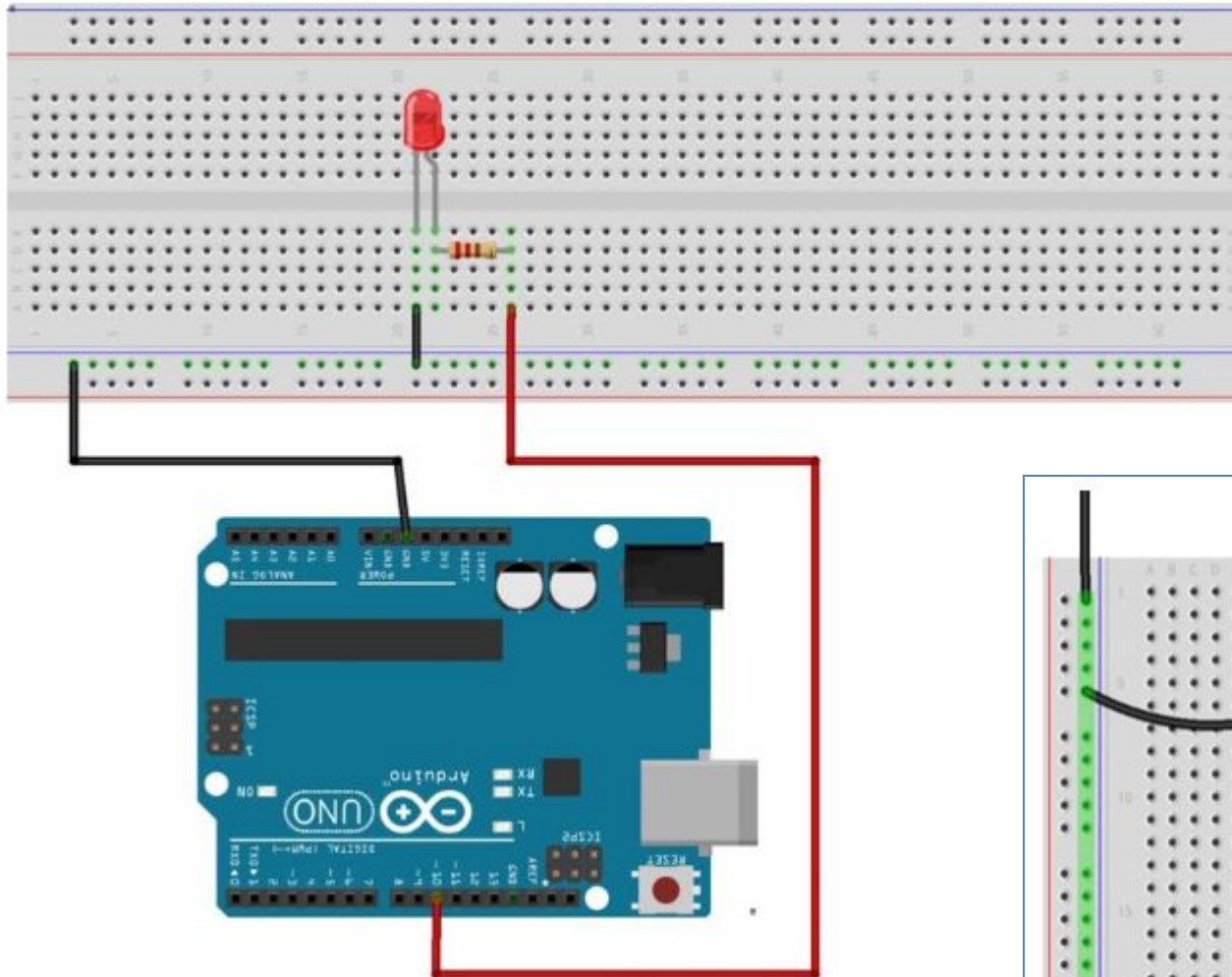
Lab #3

Step 1: Connect the First LED to the Arduino

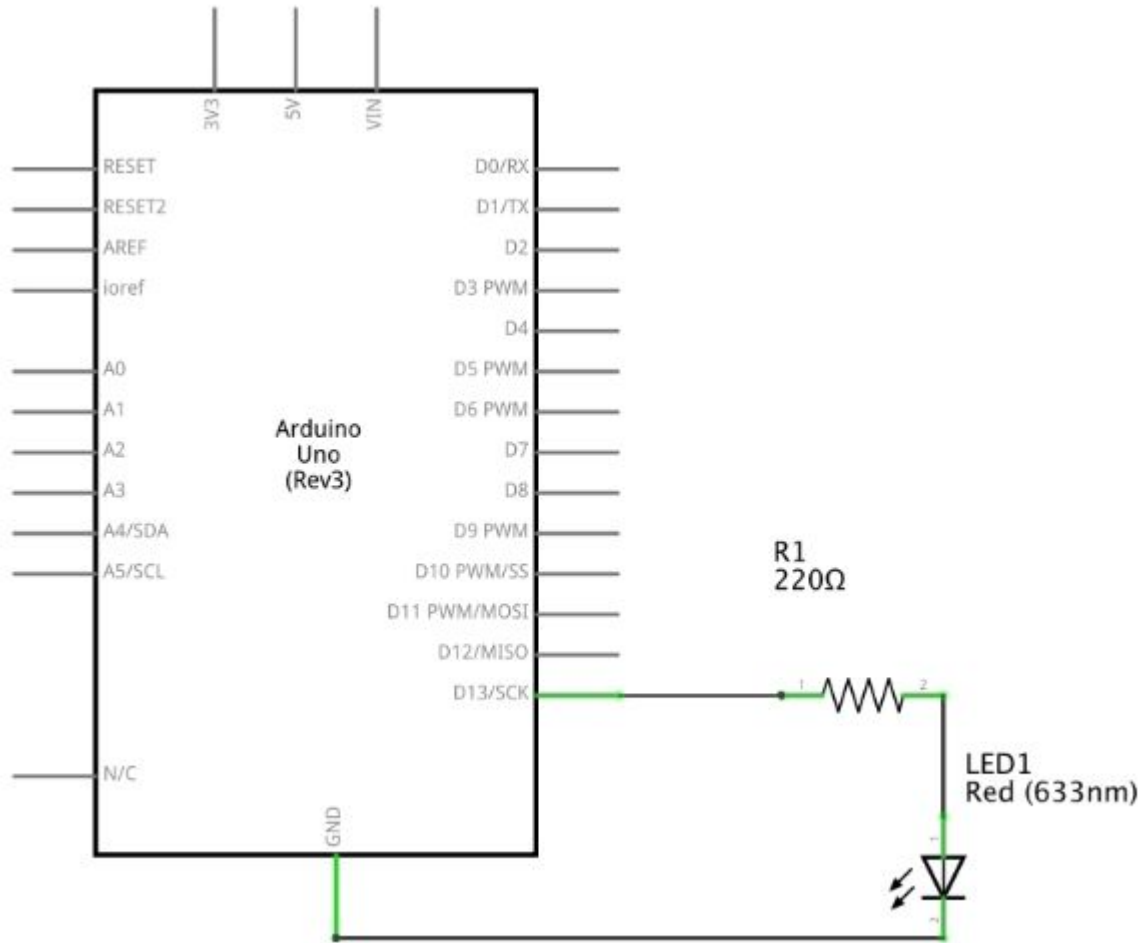
The first step is to connect the first LED to the circuit and Arduino.

1. Place one LED on the breadboard and connect the cathode to the negative rail of the breadboard.
2. Place a 220Ω resistor on the anode leg of the LED and connect the other end to pin 13 on the Arduino.
3. Connect the negative rail on the breadboard to the grounding pin on the Arduino.

Step 1: Connect the First LED to the Arduino



Schematic



CODE

```
1  int led = 13;           // The LED is attached to pin 13
2
3  /*
4  The statements in the "setup()" function are executed once when the
5  program starts.
6  */
7  void setup(){
8      pinMode(led, OUTPUT);    // Make pin 13 an output.
9  }
10
11 /* The statements in "loop()" are executed sequentially forever. */
12 void loop(){
13     digitalWrite(led, HIGH); // Turn the LED on by making pin 13 a high voltage.
14     delay(1000);             // Delay for 1 second
15     digitalWrite(led, LOW);  // Turn the LED off
16     delay(1000);             // Delay for 1 second
17 }
```

Task #2 - Traffic Light Circuit

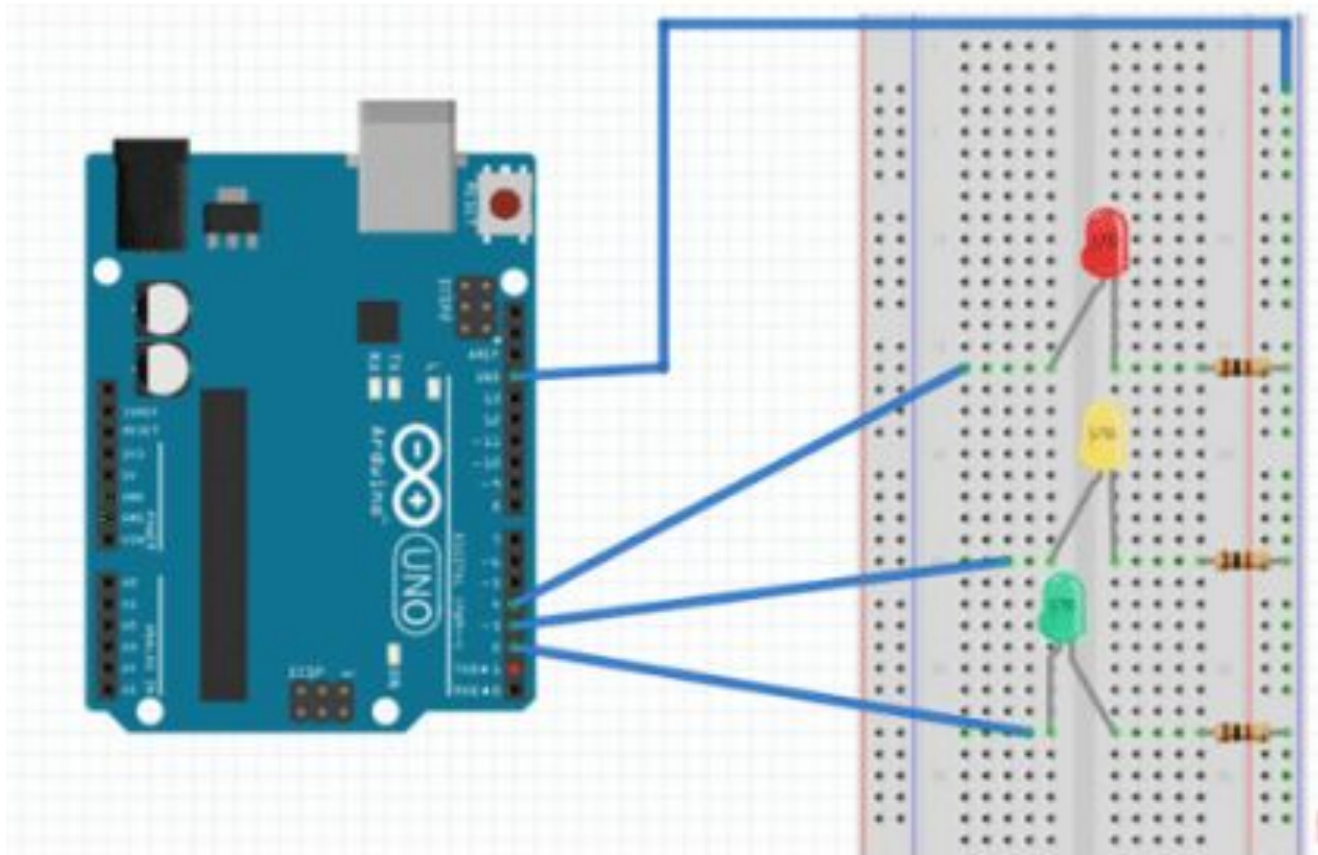
The equipment I have used in this Arduino traffic light project is listed right below.

- Arduino Uno
- Red, Yellow and green LED
- 3 x 220 Ohm resistors (Color = Brown Black Brown)
- Breadboard wires
- Breadboard

Step 1

- The circuit that we need to set up is really simple and shouldn't take you too long to do. It's a fairly simple setup with each pin controlling an LED.
- Pin 2 goes to the positive leg of the green LED.
- Pin 3 Goes to the positive leg of the yellow LED.
- Pin 4 goes to the positive leg of the red LED.
- Add a 220-ohm resistor to each of the negative LED legs and have it go to GND.

Diagram



Code

- `// variables`
- `int GREEN = 2;`
- `int YELLOW = 3;`
- `int RED = 4;`
- `int DELAY_GREEN = 5000;`
- `int DELAY_YELLOW = 2000;`
- `int DELAY_RED = 5000;`
- `// basic functions`
- `void setup()`
- `{ pinMode(GREEN, OUTPUT);`
- `pinMode(YELLOW, OUTPUT);`
- `pinMode(RED, OUTPUT);`
- `}`
- `void loop() {`
- `green_light();`
- `delay(DELAY_GREEN);`
- `yellow_light();`
- `delay(DELAY_YELLOW);`
- `red_light();`
- `delay(DELAY_RED);`
- `}`

Code

- Now for each LED, we will need to create a function. As you can see the **green_light()** function will turn the green LED on while turning the yellow and red LEDs off.
- `void green_light()`
- `{`
- `digitalWrite(GREEN, HIGH);`
- `digitalWrite(YELLOW, LOW);`
- `digitalWrite(RED, LOW);`
- `}`

- `void yellow_light()`
- `{`
- `digitalWrite(GREEN, LOW);`
- `digitalWrite(YELLOW, HIGH);`
- `digitalWrite(RED, LOW);`
- `}`

- `void red_light()`
- `{`
- `digitalWrite(GREEN, LOW);`
- `digitalWrite(YELLOW, LOW);`
- `digitalWrite(RED, HIGH);`
- `}`

Report

The report should include:

1. front page and title
2. a code
2. photos of the scheme
3. conclusion

Please prepare for the Post lab using lecture notes.