

Arduino Lab – Blinking LEDs

This lab will teach you a little about basic electronics circuits and how to use a small computer to control lights and make music. The small computer is called an Arduino.

Top view of Arduino Uno board

Pin 13 input to Arduino also has an LED connected to it.

Digital input/output (I/O) pins. The ones with ~ have intensity that can be controlled through pulsed wave modulation (PWM).

Reset button

Plug in USB cable here. Plug the other end into the PC.

CPU Chip – computer “brain” — ATmega 328P chip.

Voltage pins.

Analog input pins.

1. Using the Arduino Software

Now we are ready to look at the software. Open the Arduino program by going to upper left icon

on Linux desktop, click on the Electronics folder, and then click on the Arduino IDE icon.



Plug one end of the USB cable into the Arduino and the other end into your Linux PC.

Check that Tools->Board is set to “Arduino Uno.” There should be a dot next to the item. If not, please click on this item with the trackpad, and then the dot should appear.

Check that Tools->Serial Port is set to something like “dev/ttyACM0.” There should be a dot next to the item. If not, please click on this item with the trackpad, and then the dot should appear.

At the lower right corner of the window, it should now say “Arduino Uno on /dev/ttyACM0”

2. Blink Program

Open the first example program, called Blink by going to the File menu, select “Examples”, then “01. Basics” and then “Blink”.

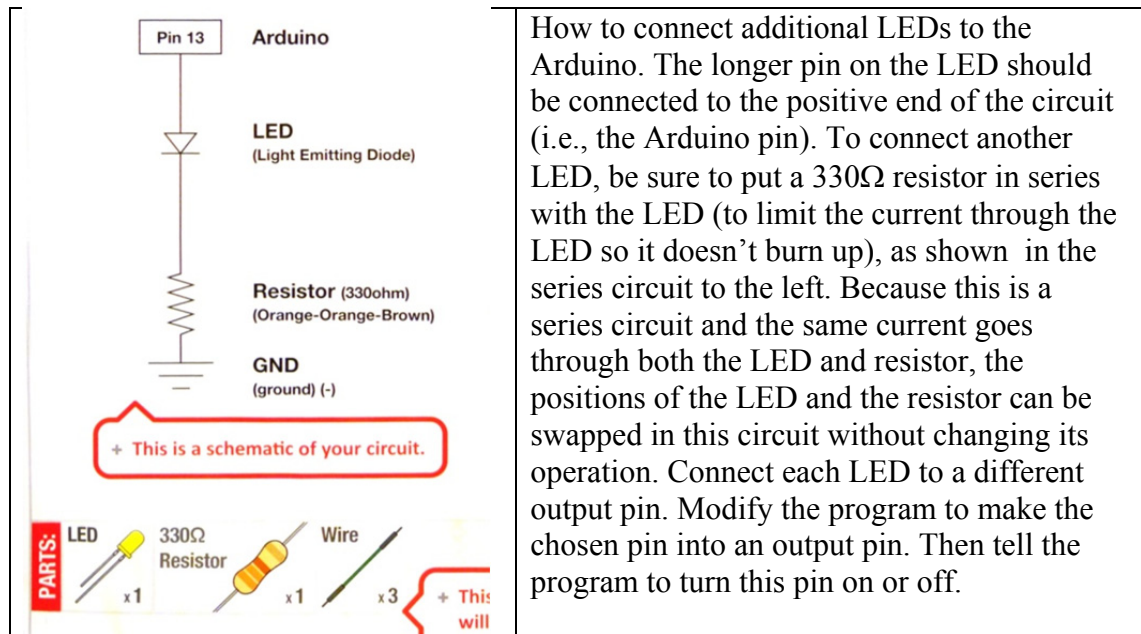
This will load the Blink program, which will make the Red light emitting diode (LED), which is connected to Pin 13, turn on for a second and then turn off for a second. Arduino programs are all written in C (actually in C++). Read the program and see if it makes sense. The part inside setup is executed once, at the beginning. It initializes pin 13 to be an output pin. The part inside loop executes over and over again, forever, or until the Arduino is reset or reprogrammed. It turns the LED on (sends a high voltage out to pin 13), waits 1000 mS=1 sec, turns the LED off (sends a low voltage out to pin 13), and waits 1000 mS.

To load this program into the Arduino, put your trackpad cursor over the button which looks like a right arrow. You will see the white word “Upload.” Click this Upload button. You should see the program say “Compiling sketch” and then “Uploading” followed by a message about the number of bytes (>1000) uploaded into the Arduino.

Then the program should start running and blinking the LED! If nothing is happening, you can try to push the Reset button (near the USB connector, upper left corner of photo).

Now try modifying the program. Save your trial program under another name, such as Blink2, by using the menu item “File, Save As.” You can save it again under the same name (Blink2) by clicking the menu item “File, Save” or using the keyboard shortcut “Control – S.”

1. Change the program to change the times for the LED to be on versus off. For example, you can try turning it on for 1.5sec, and off for 0.5sec.
2. Duplicate some lines of the code so that the LED turns on for 2 sec, off for 1 sec, on for 1 sec, and off for 0.5 sec.
3. Make up your own variation of times for LED on and off.
4. Connect additional LEDS (several colors available) to other pins (see figure on next page). Rewrite the program to turn them on or off.



How to connect additional LEDs to the Arduino. The longer pin on the LED should be connected to the positive end of the circuit (i.e., the Arduino pin). To connect another LED, be sure to put a 330Ω resistor in series with the LED (to limit the current through the LED so it doesn't burn up), as shown in the series circuit to the left. Because this is a series circuit and the same current goes through both the LED and resistor, the positions of the LED and the resistor can be swapped in this circuit without changing its operation. Connect each LED to a different output pin. Modify the program to make the chosen pin into an output pin. Then tell the program to turn this pin on or off.

Changing the intensity of the light from the LEDs.

The Arduino cannot write an analog voltage output. Instead, it can quickly turn the digital inputs on and off. You call the `analogWrite` subroutine and use Pulsed Wave Modulation.

This only works on the pins labeled with the symbol ~ on the board.

The C code is:

```
analogWrite(pin, value); // where value is an integer between 0 and 255.
```

Value is the duty cycle, i.e., how much of the time the pin is on or off.

Value = 255; 100% duty cycle means the pin is always on.

Value = 192; 75% duty cycle means the pin is on 75% of the time. Think of a square wave which is high 75% of the time and low 25% of the time.

Value = 128; 50% duty cycle means the pin is on 50% of the time.

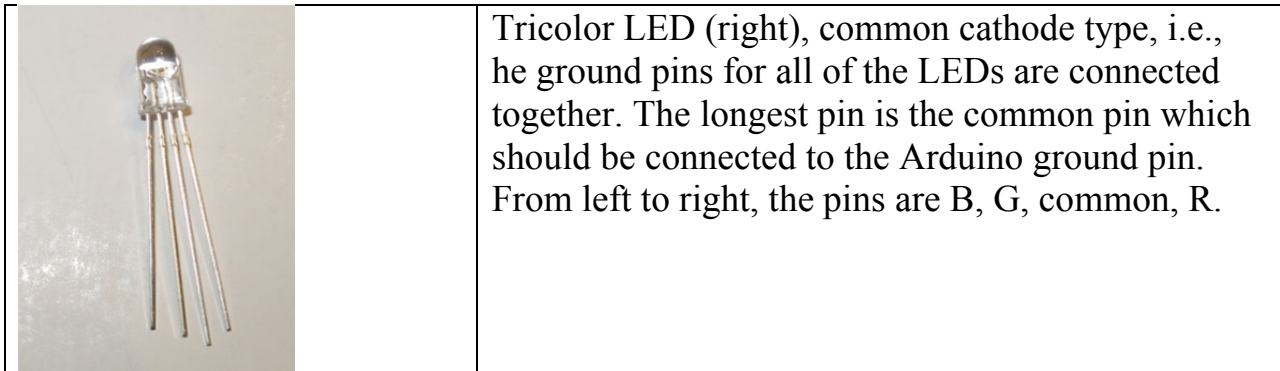
Value = 64; 25% duty cycle means the pin is on 25% of the time; i.e., square wave which is high 25% of the time, and low 75% of the time.

Value = 0; 0% duty cycle, i.e, signal is always low.

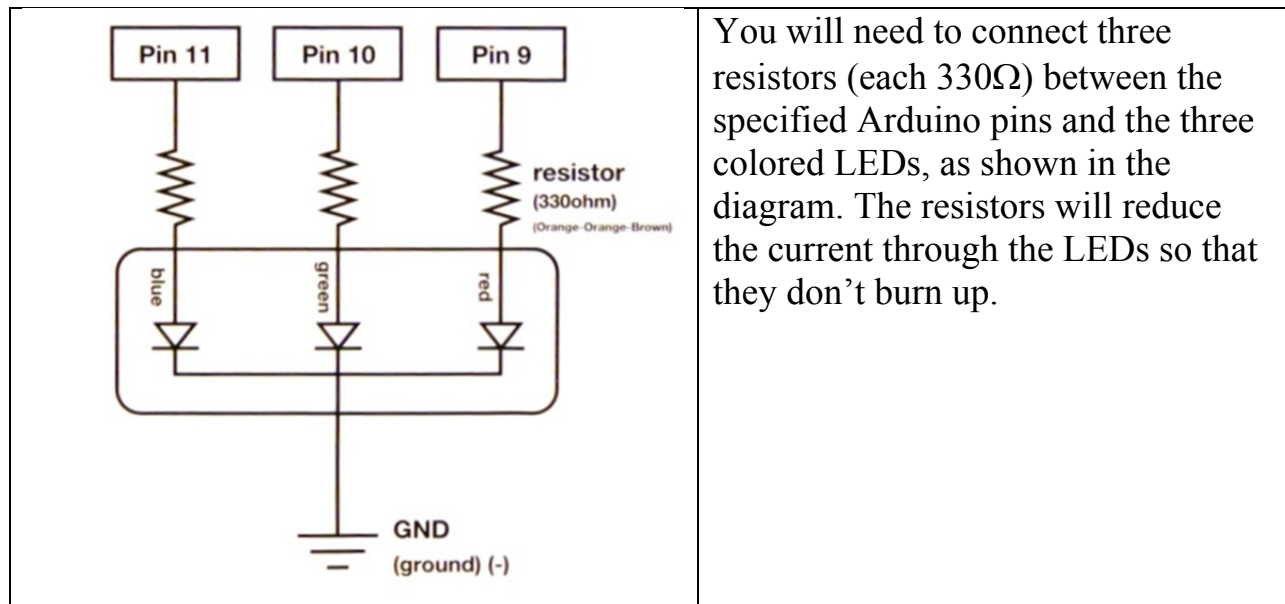
Write a "for" loop to change the intensity of one of your LEDs from 0 to 255, incrementing by 1 each time the "for" loop executes. Try a delay of 10mS between LED intensity steps of 1. What happens if the delay time is too short?

3. TriColor LED

The tricolor LED has 3 LEDs (Red, Green, Blue, i.e., RGB) all inside the same package.



Here is the circuit diagram for connecting the tricolor LED to the Arduino.



Login to “smartsite.ucdavis.edu” using your UC Davis userid and passphrase. Click the site for “COSMOS Cluster 1 2016.” Click on Resources at the left, then click on the folder icon for “Arduino lab.” Click on the program “TriColorLED_start.ino” to download it. Tell the computer to save the file. In the browser, click on the download arrow. The computer will tell you it wants to move the program into a directory of the same name. Tell it “OK” to do that. Then it will open the program in the Arduino IDE.

Read the program. The part inside loop calls 2 subroutines: mainColors, and showSpectrum.

Write the subroutine mainColors, which turns on the different colors in turn, each for a second: red, green, blue, then yellow, cyan, purple, and finally white.

Write the subroutine showSpectrum, that changes the intensity (brightness) of the LED and steps through all of the colors of the rainbow. Upload the program into the Arduino. Enjoy the colors!