Blink

Blink | Arduino 1.6.12

Blink §

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

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

14 Arduino/Genuino Uno on /dev/cu.usbmodem1411

At the end of the previous lecture slides, we loaded and ran the blink program. When the program is running, the built-in LED blinks on and off — on for one second and off for one second. It is very simple, but it consists of functions that are not familiar: pinMode, digitalWrite and delay. However, it is not hard to guess the purpose of these functions. We will look at them in more detail.

Arduino program commands

The official Arduino web site (arduino.cc) has tons of information about using Arduinos. In addition, there are a seemingly infinite number of other web sites with all manner of Arduino-related content. Be sure to explore.

For writing programs, a good place to start is the language reference page of the Arduino web site: https://www.arduino.cc/en/Reference/HomePage. Many of items listed there should be familiar to us:

- if…else conditionals
- for and while loops
- basic syntax and punctuation
- arithmetic, comparison, and Boolean operators
- standard variable types (int, char, long, double, string, etc.)
- math functions (sin, cos, sqrt, abs, etc.)

On the right of the page are functions that are less familiar. These include the commands that set up and control the operation of the input and output pins. There are also some functions that are related to timing. These are the functions that are essential to using microcontrollers.

Blink

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

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Let's start by looking at the two functions in the program that control the blinking:

`pinMode()` and `digitalWrite()`.

`pinMode()`

The built-in LED is hard-wired to pin 13. (We could replace `LED_BUILTIN` with the literal 13 and the program would function identically. Try it.) But let's face it — that is not a very exciting LED.

We might want to connect a brighter LED and make it blink. We can change to a different pin and connect our own LED to glitz things up a bit. To use a different pin, we must modify the `pinMode()` statement.

First, note that pins 0 thru 13 are the digital connections. We can use any of those. Let's use pin 6. We could just plug the literal number "6" into the pinMode function, but a bit of abstraction is usually a good idea. So define an integer constant `RED_LED_PIN` and assign it the number 6.

The `pinMode()` function takes two values, the pin number (as we just defined) and a value that sets how the pin will work. Digital pins can either be outputs or inputs. If it is an output, the voltage on the pin will be set to either `HIGH` (meaning 5 V) or `LOW` (meaning 0 V). If the pin is set to be a input, it will *read* the voltage that is on the pin, and return `LOW` if the voltage on the pin is low (< 2.5 V) and return HIGH if the voltage on the pin is hight (> 2.5 V). (We will look at using digital pins as inputs later.)

`digitalWrite()`

Once a pin has been defined as digital output, the output value (0 V or 5 V) can be set with the `digitalWrite()` function.

`digitalWrite()` takes two values as parameters. The first is the number of the particular pin and the second is output level, high or low. The Arduino language provides the keywords, `HIGH` and `LOW` that can be used in the function. The numbers 1 and 0 would work just as well.

For example, the function call

```
digitalWrite( 9, HIGH);
```

sets the output voltage on pin 9 to the high value (5 V).

It is quite easy.

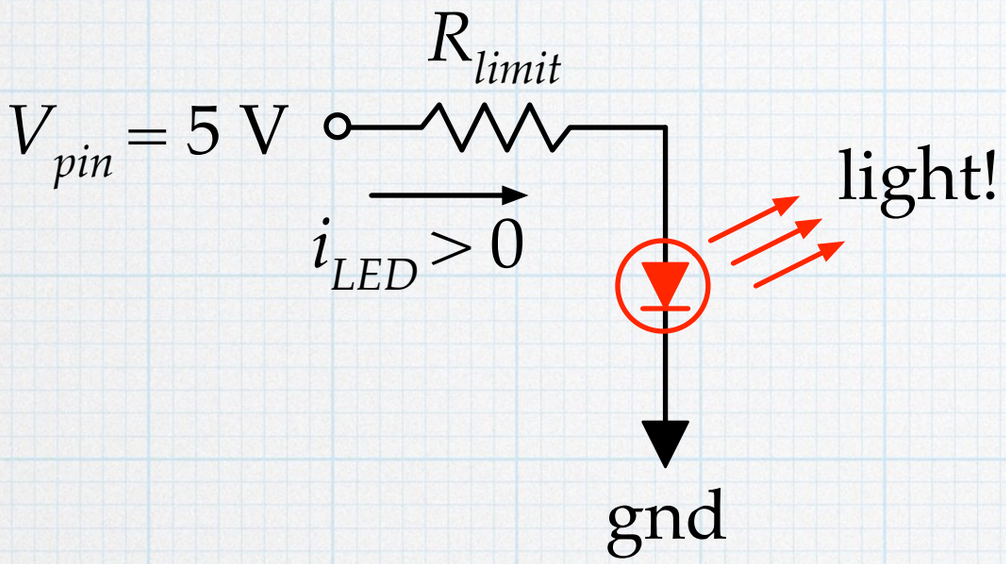
For further details about `pinMode()` and `digitalWrite()` check the Language Reference.

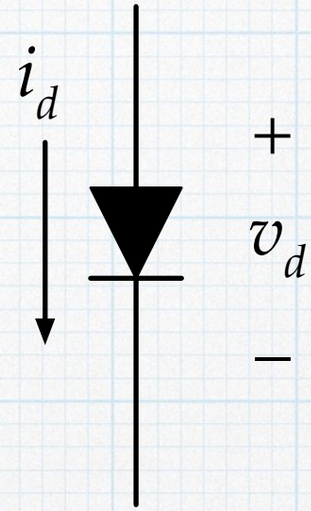
light-emitting diodes (LEDs)

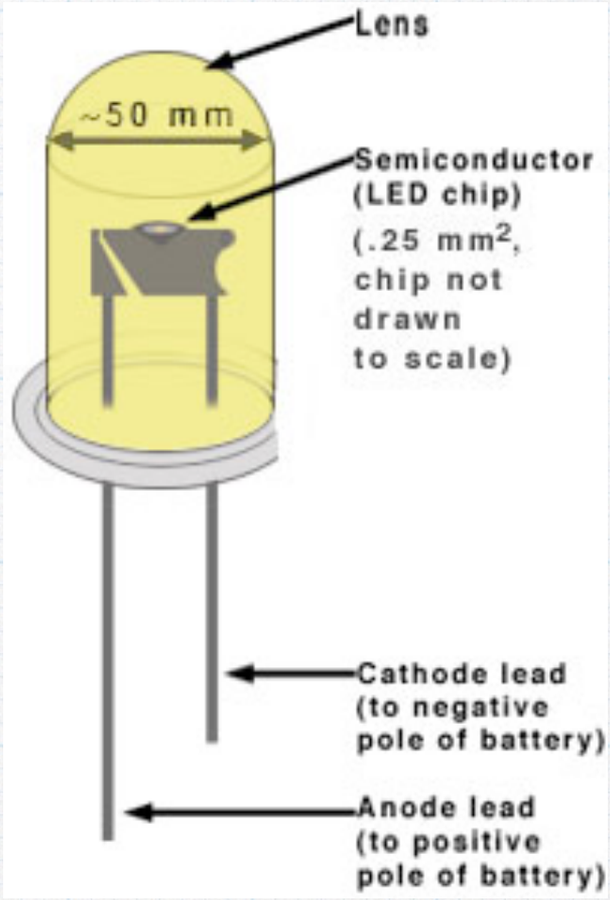
A light-emitting diode is similar to a regular rectifying diode except that it gives off light when a forward current is passing through it. LEDs come in a wide variety of colors. Red, green, yellow, and blue are very common, but other colors, including infrared and ultraviolet are available.

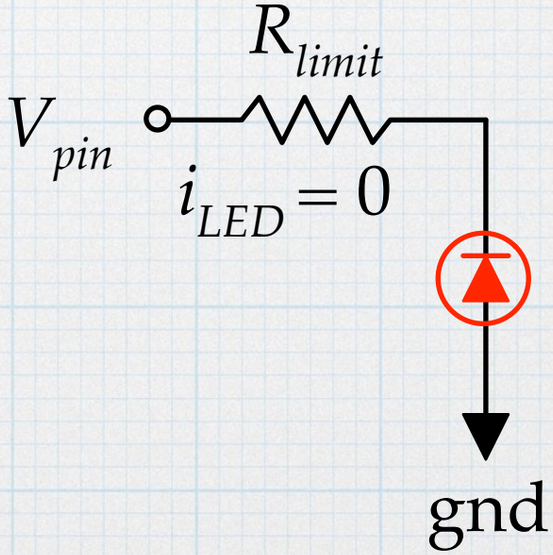
Using an LED is very simple, but there are a couple of precautions. First, you must *always* have a resistor in series with it. Without the resistor, you run the risk of having the diode current become too high, causing the diode to burn out. The value of the resistor is not critical — anything between 100 Ω and 1 kΩ is probably OK, although lower resistance values will give a brighter output.

Secondly, the diode passes current in only one direction — it is not like a resistor. It must be connected with the right polarity. If you install it backwards, it will never pass current and so will never lights up.









oops!
Backwards
always dark

The current through the LED is

$$i_D = \frac{V_{out} - v_D}{R_{limit}} \approx \frac{5\text{ V} - 1\text{ V}}{R_{limit}}$$

What value for the current-limiting resistor?

From the equation, we see that the LED current is inversely related to the resistance — the current goes up as the resistance is reduced. A, smaller resistor will lead to a brighter LED output.

However, the digital outputs have limited current sourcing / sinking capability. Referring to the Arduino pin current limitations web page

http://playground.arduino.cc/Main/ArduinoPinCurrentLimitations

we note that the maximum current that can flow in or out of a digital pin is 40 mA. However, this is the limit where damage might occur. Therefore, a recommended safe limit is 20 mA. From the diode current equation, this would correspond to a limiting resistance value of about 200 Ω. Since 220-Ω resistors are a standard value, we will use that as a our typical diode current limiting resistance value — giving an LED current of about 18 mA.

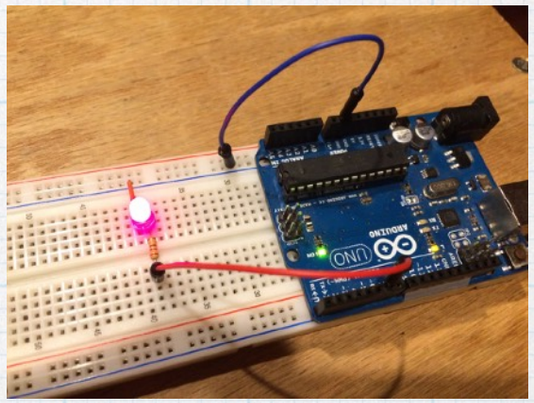
The web page also gives a maximum value of for the total Arduino current of 200 mA. We should be aware of that limit when using multiple devices on the digital pins.

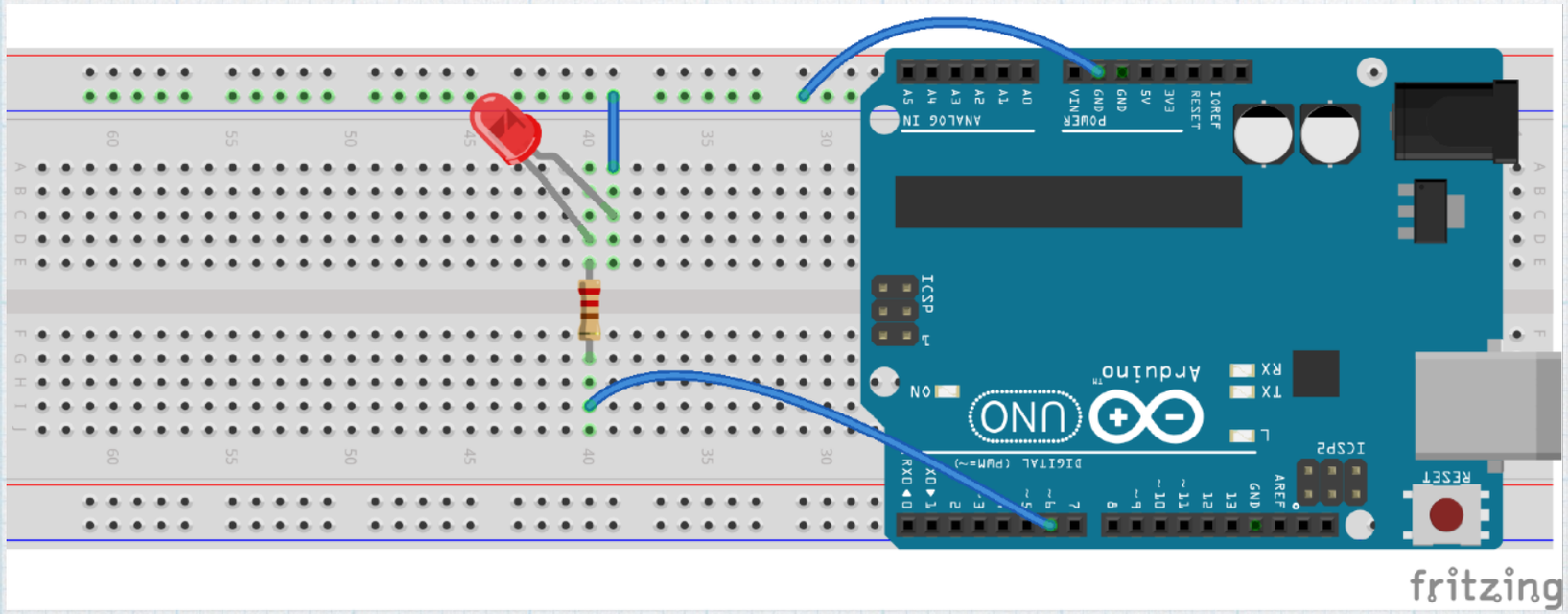
Finally, in battery-powered applications, conserving energy may be more important than LED brightness. In those cases, we might opt for larger resistance (lower currents) to extend battery life.

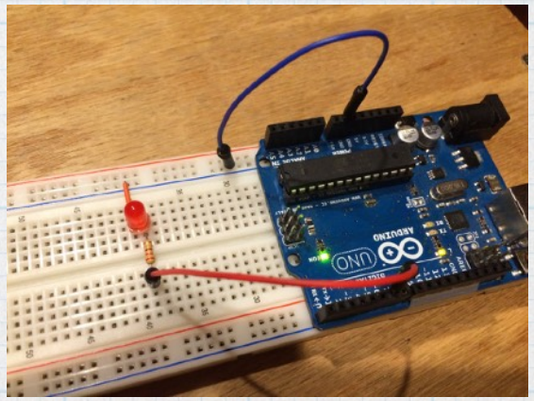
Blink one red LED.

Blink1 §

```
// Uses pin 6 to control the blinking of an external red LED.

const int RED_LED_PIN = 6;

void setup() {

  pinMode(RED_LED_PIN, OUTPUT);
}

void loop() {

  digitalWrite(RED_LED_PIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(RED_LED_PIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

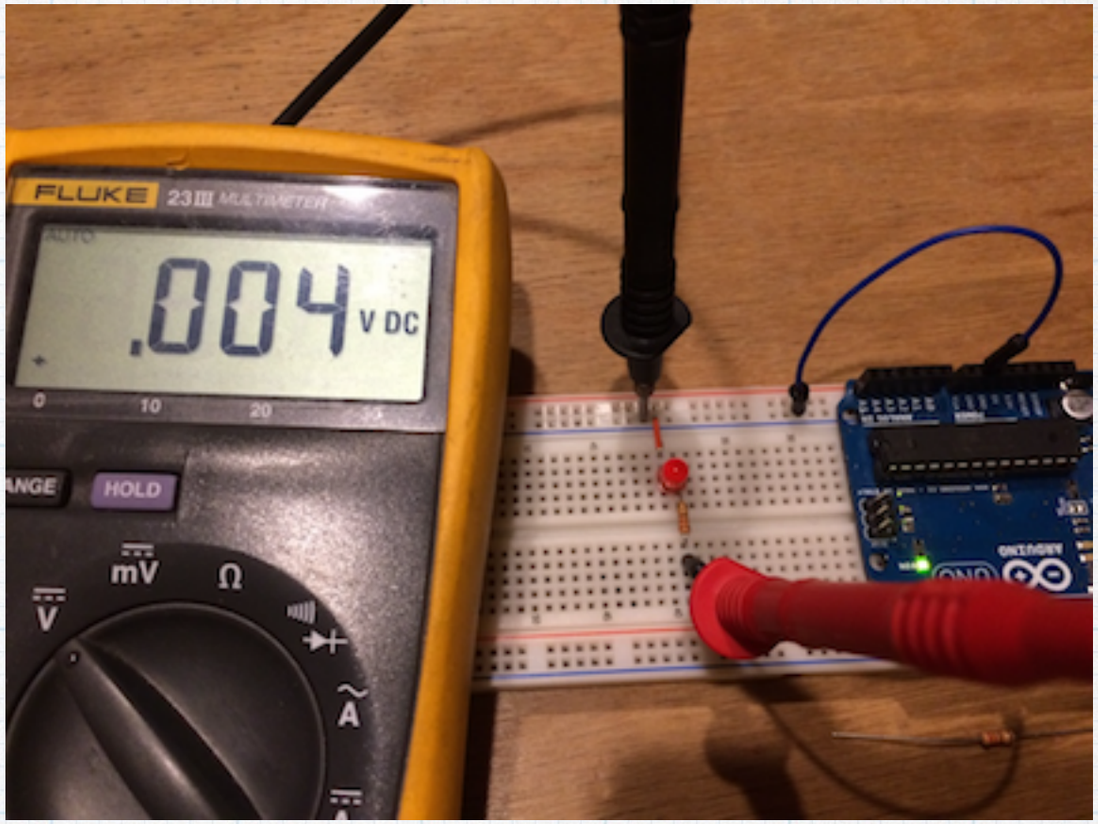




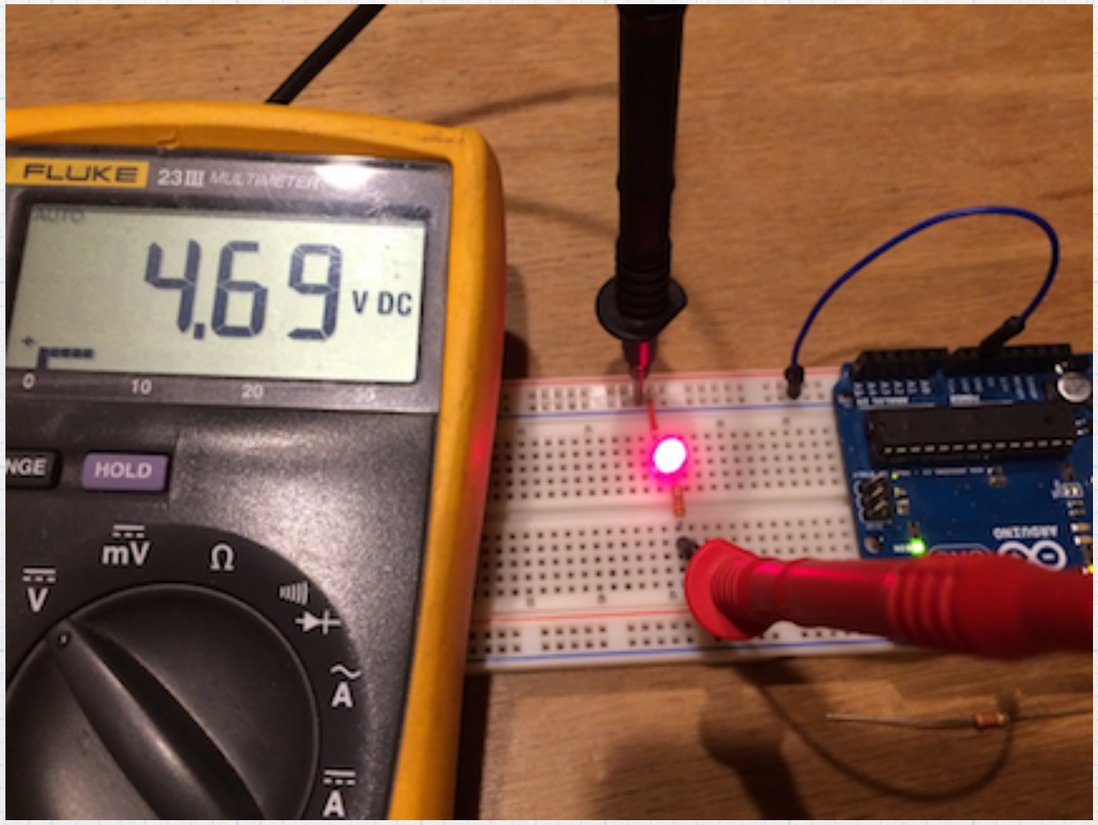


(To make these diagrams: http://fritzing.org/home/)

The low output is pretty close to zero, indeed.



The high output is not exactly 5 V, but it is probably “high enough”. This amount of variation is typical.



Blink two LEDs — alternating red and green.

Blink2

```
// Uses pins 6 & 7 to control the blinking of external red and greeen LEDs.

const int RED_LED_PIN = 6;
const int GREEN_LED_PIN = 7;

void setup() {

  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
}

void loop() {

  digitalWrite(RED_LED_PIN, HIGH);    // Turn the red on...
  digitalWrite(GREEN_LED_PIN, LOW);   // ...and green off.
  delay(1000);                        // Wait for a second.

  digitalWrite(RED_LED_PIN, LOW);     // Now do the opposite.
  digitalWrite(GREEN_LED_PIN, HIGH);
  delay(1000);
}
```