

# **Arduino Beginner Kit**

## **User Manual**

## **Content**

<b>Introduction.....</b>	<b>3</b>
<b>Arduino Software (IDE).....</b>	<b>3</b>
<b>Hardware - Arduino Uno Board.....</b>	<b>8</b>
<b>Arduino Library.....</b>	<b>8</b>
<b>Language Reference.....</b>	<b>9</b>
<b>Tutorial.....</b>	<b>10</b>

## **Arduino Beginner Kit**

### **1. Introduction**

#### **What is Arduino?**

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

#### **Why Arduino?**

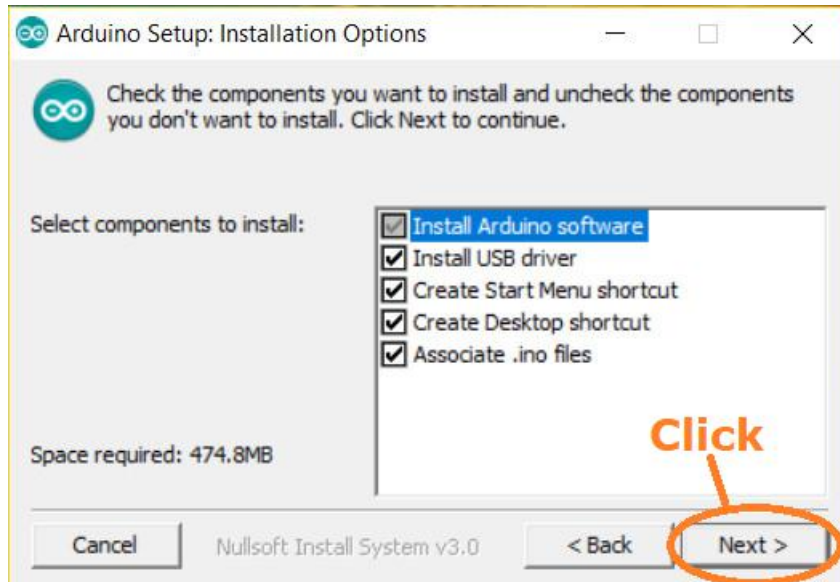
Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

### **2. Arduino Software (IDE)**

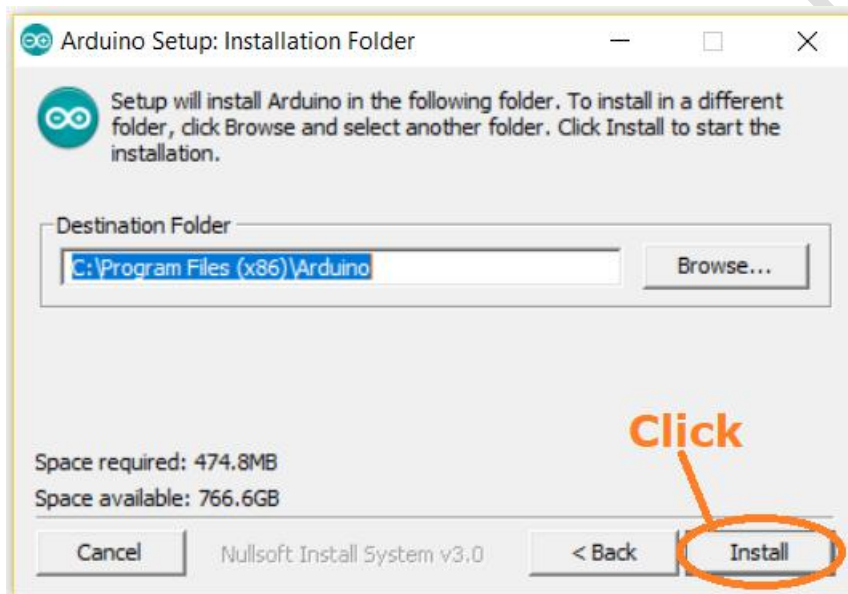
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

#### **Installation steps:**

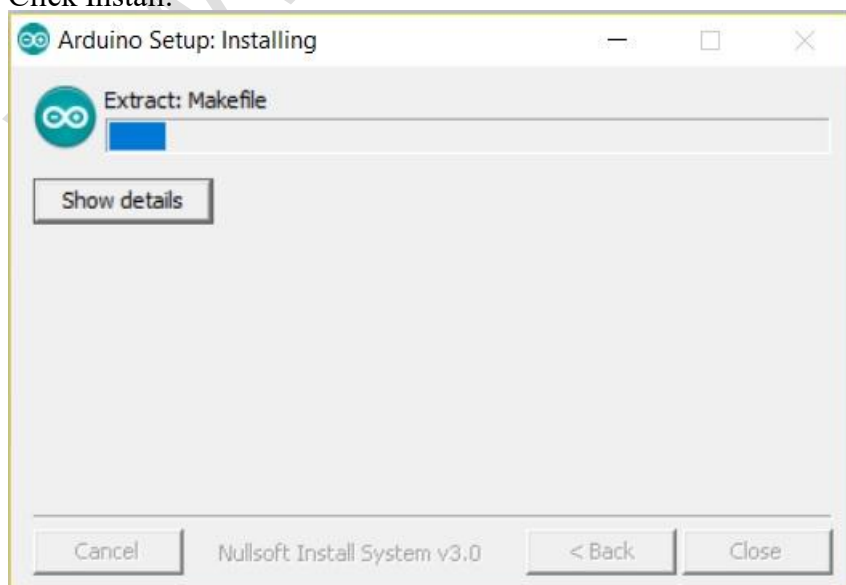
- a) Download the latest version from <https://www.arduino.cc/en/Main/Software>
- b) Proceed with the installation and follow the instructions as follow:



Click 'Next' to continue.



Click Install.

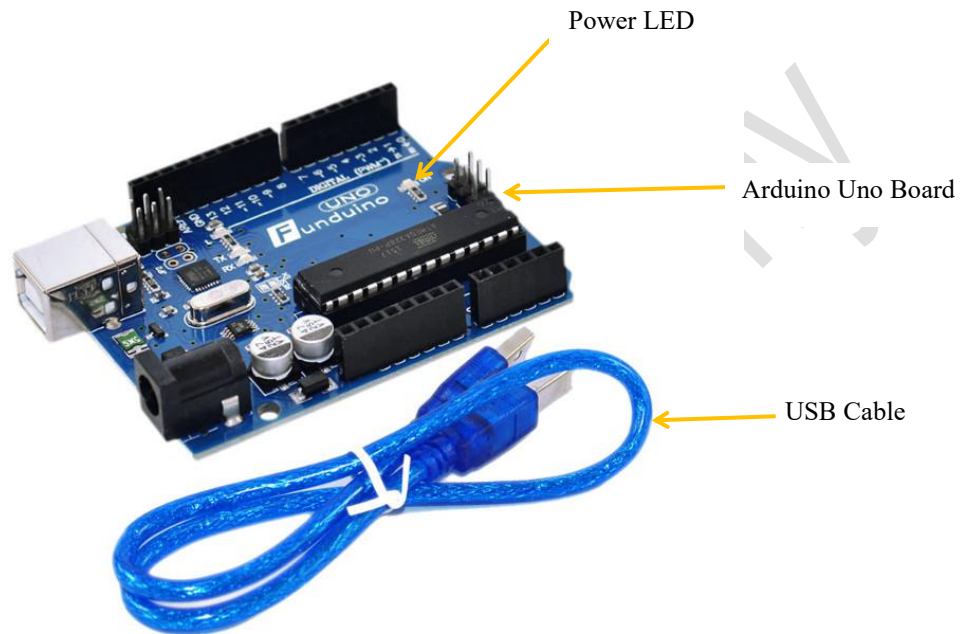


Extract and install all the required files. Click 'Yes' for install the driver.

c) Connect your Uno board with an A to B USB cable; sometimes this cable is called a USB printer cable.

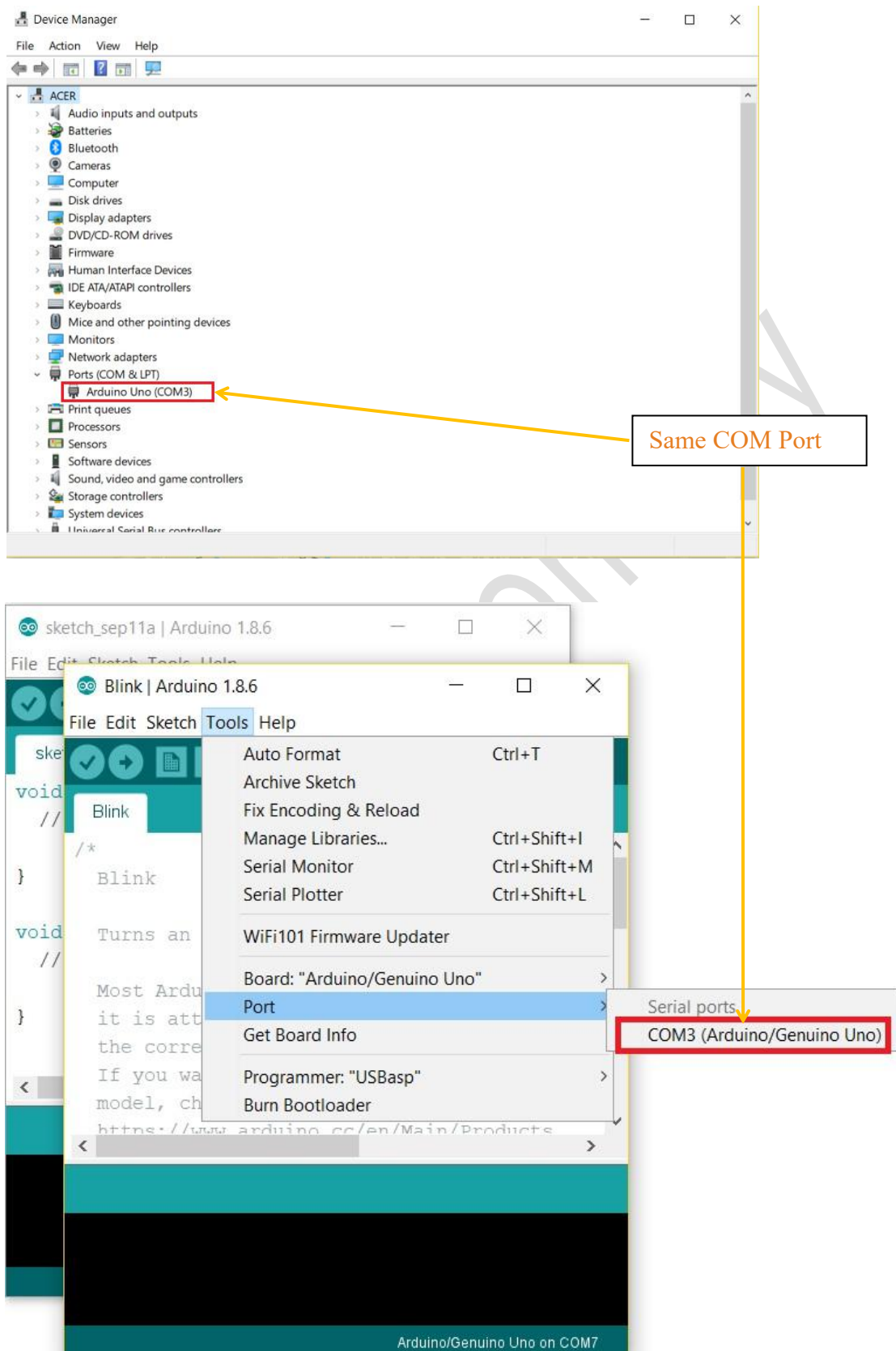
The USB connection with the PC is necessary to program the board and not just to power it up. The Uno automatically draw power from either the USB or an external power supply. Connect the board to your computer using the USB cable.

The green power LED (labelled PWR) should go on.

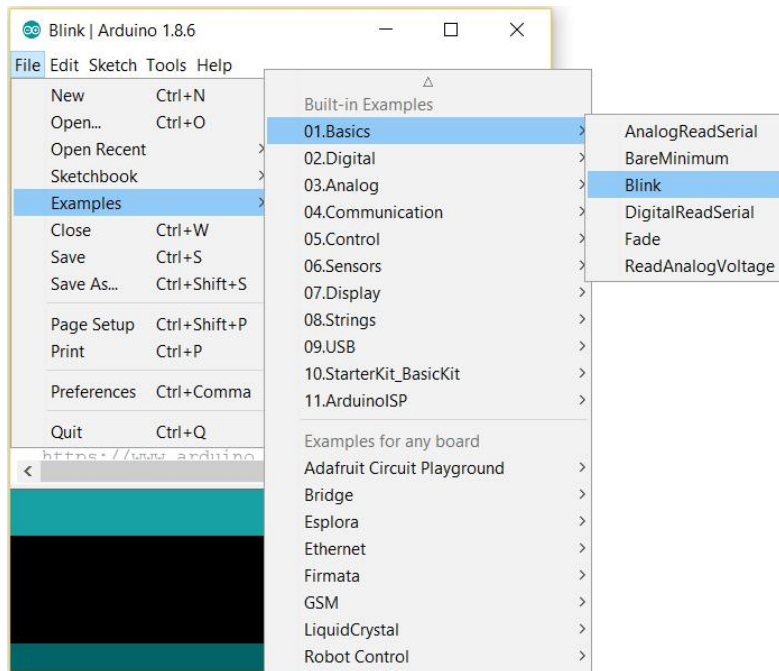


Connect Uno board to PC/Notebook.

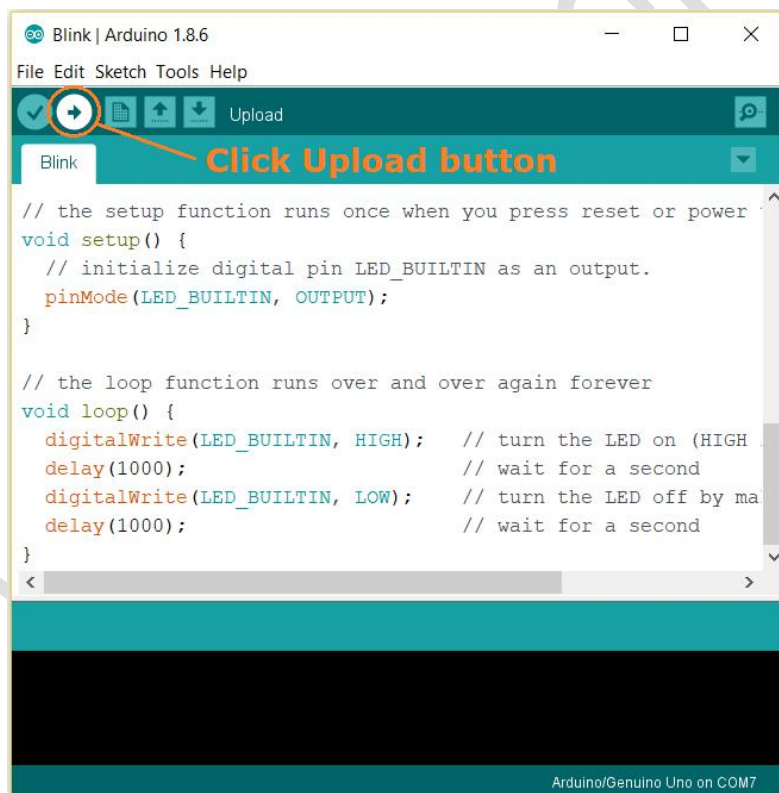
d) Run Arduino IDE, then select the COM Port.



e) Open Project “Blink” (Click menu → File → Examples → 01.Basic → Blink)



f) Click Upload.

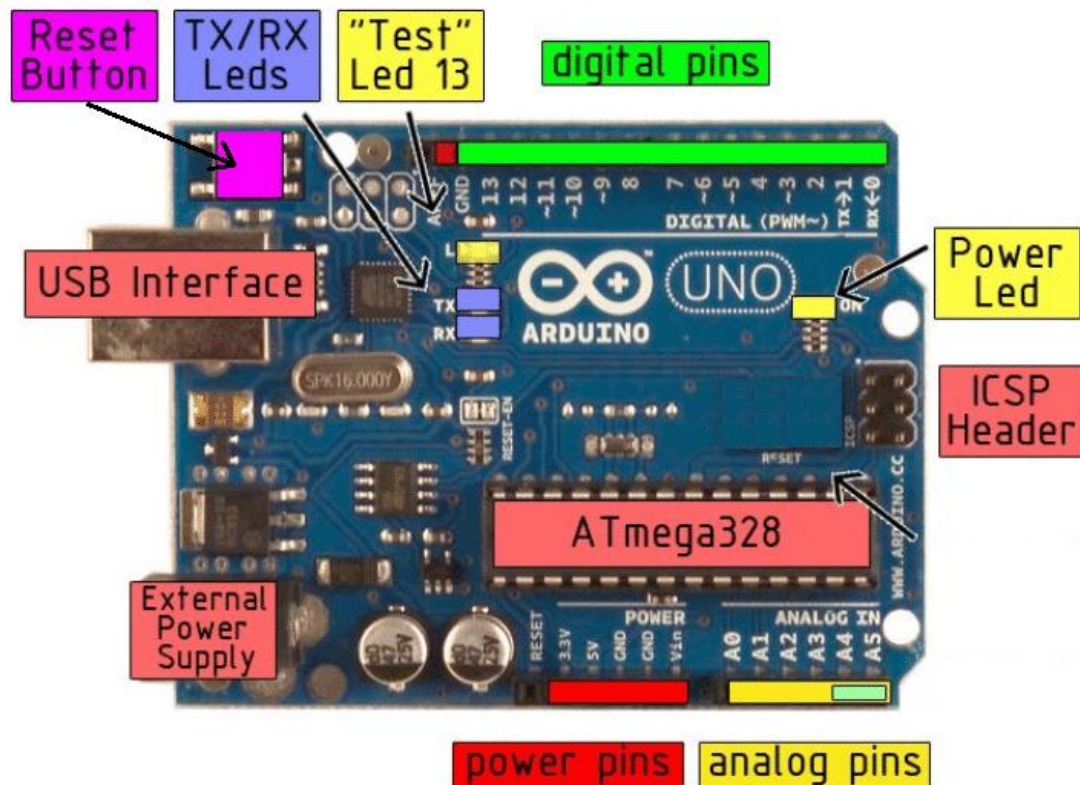


g) Result

After uploaded, you should see the pin 13 (L) LED on the board start to blink (in orange). If it does, congratulations! You've gotten Arduino up-and-running.



### 3. Hardware - Arduino Uno Board



#### Reference Information:

- Schematic ([https://www.arduino.cc/en/uploads/Main/Arduino\\_Uno\\_Rev3-schematic.pdf](https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf))
- Microcontroller ATmega328 ([http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P\\_Datasheet.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf))
- Arduino Pin Mapping (<https://www.arduino.cc/en/Hacking/PinMapping168>)

### 4. Arduino Library

#### What are Libraries?

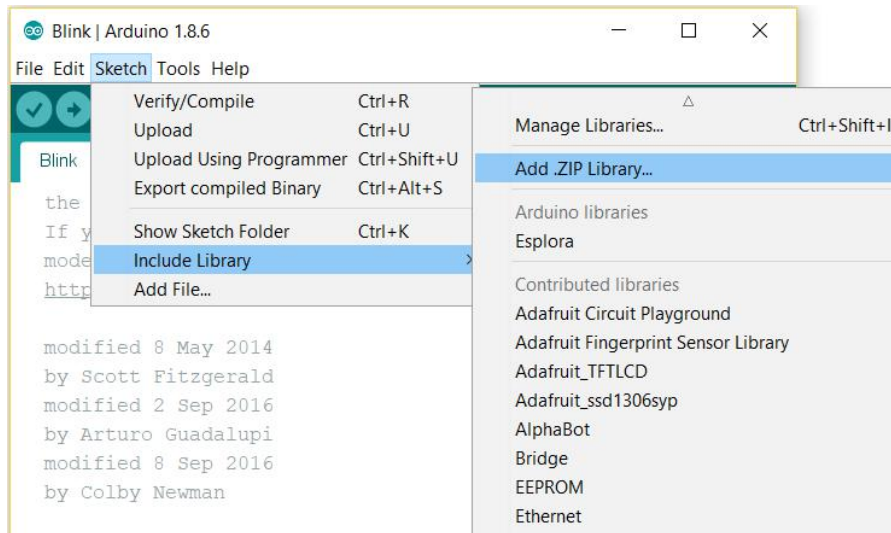
Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

#### Importing a .zip Library?

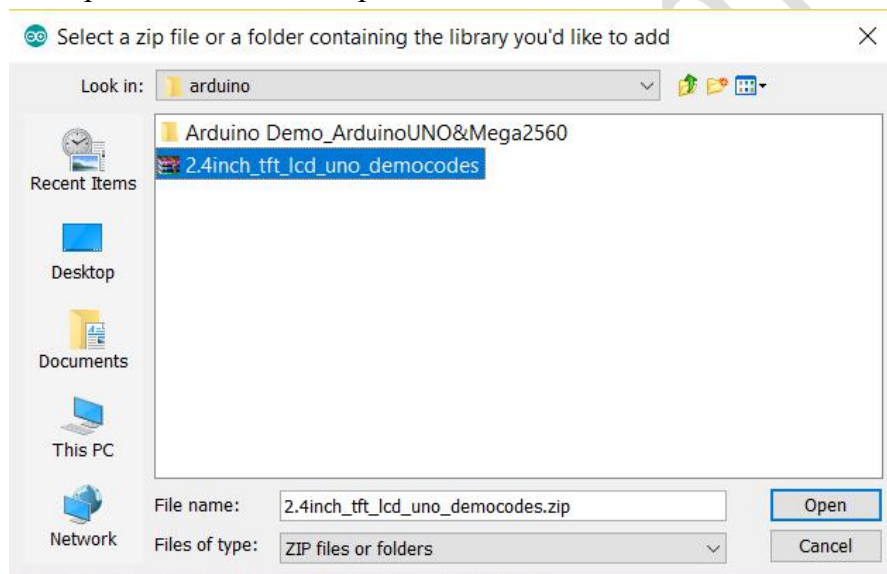
Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, leave it as is.



In the Arduino IDE, navigate to Sketch > Include Library > Add .ZIP Library. At the top of the drop down list, select the option to "Add .ZIP Library".



You will be prompted to select the library you would like to add. Navigate to the .zip file's location and open it.



Return to the Sketch > Include Library menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory.

For more details, refer <https://www.arduino.cc/en/Guide/Libraries>

## 5. Language Reference

Arduino programming language can be divided in three main parts: structure, values (variables and constants), and functions.

Please refer <https://www.arduino.cc/reference/en/> for more details.

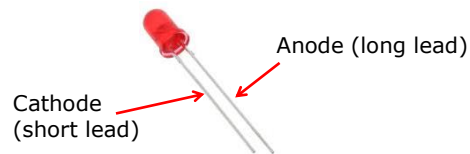
## 6. Tutorials

### Example 1 - Blink

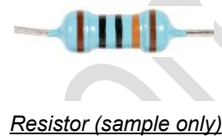
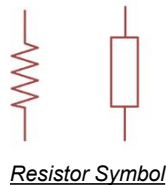
This example shows the simplest thing you can do with an Arduino board to see physical output: it blinks the external LED by using I/O pins.

#### Hardware Required

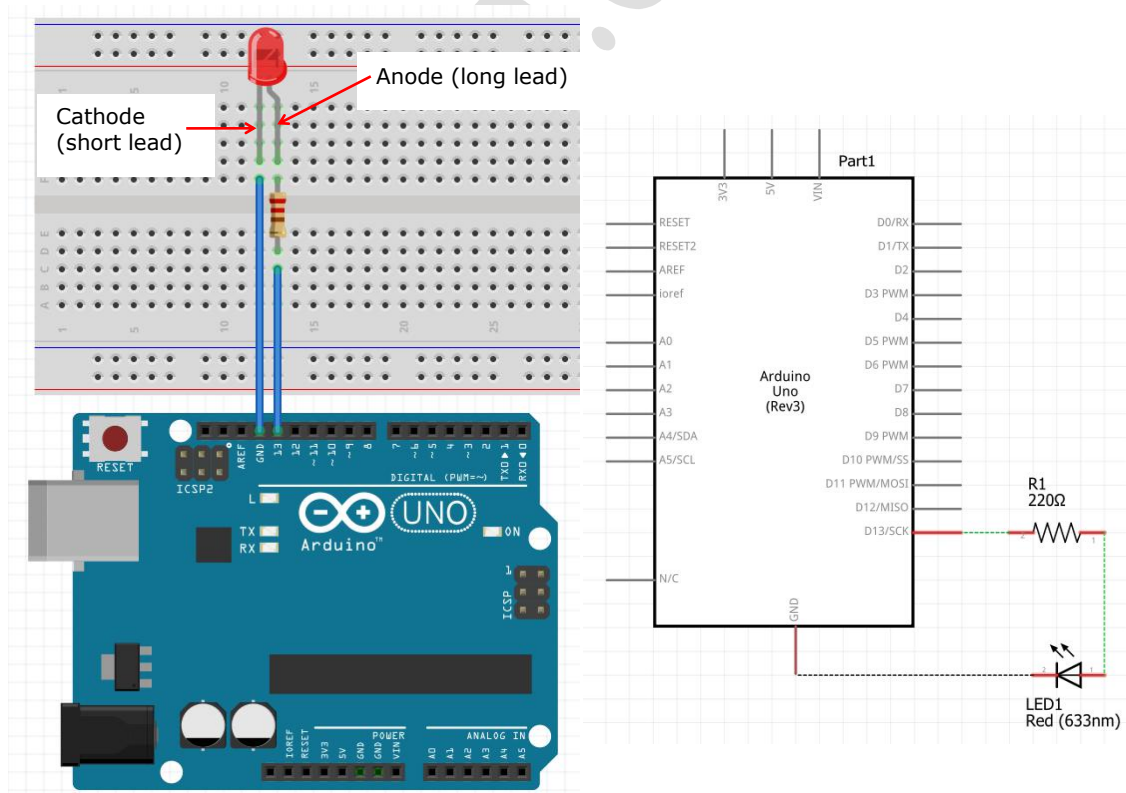
- Arduino Uno Board and Bread Board
- LED (light-emitting diode), is a two-lead semiconductor light source.



- 220 ohm resistor



#### Circuit



## Code

```

Blink | Arduino 1.8.6
File Edit Sketch Tools Help

Blink $


const int ledPin = 13; // set I/O Pin 13 as LED pin

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

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

```

## Compile and Upload

Click  (upload button) to compile the code and then upload to Uno board.

## Result

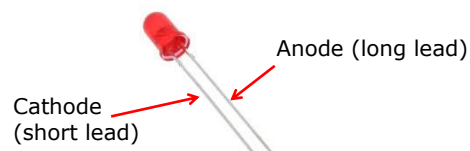
Turn on an LED for one second, then turn off one second, repeatedly.

## Example 2 - LED Display Effects

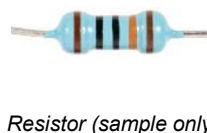
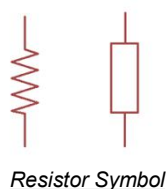
This example show various of LED display effect by using 8 LEDs.

## Hardware Required

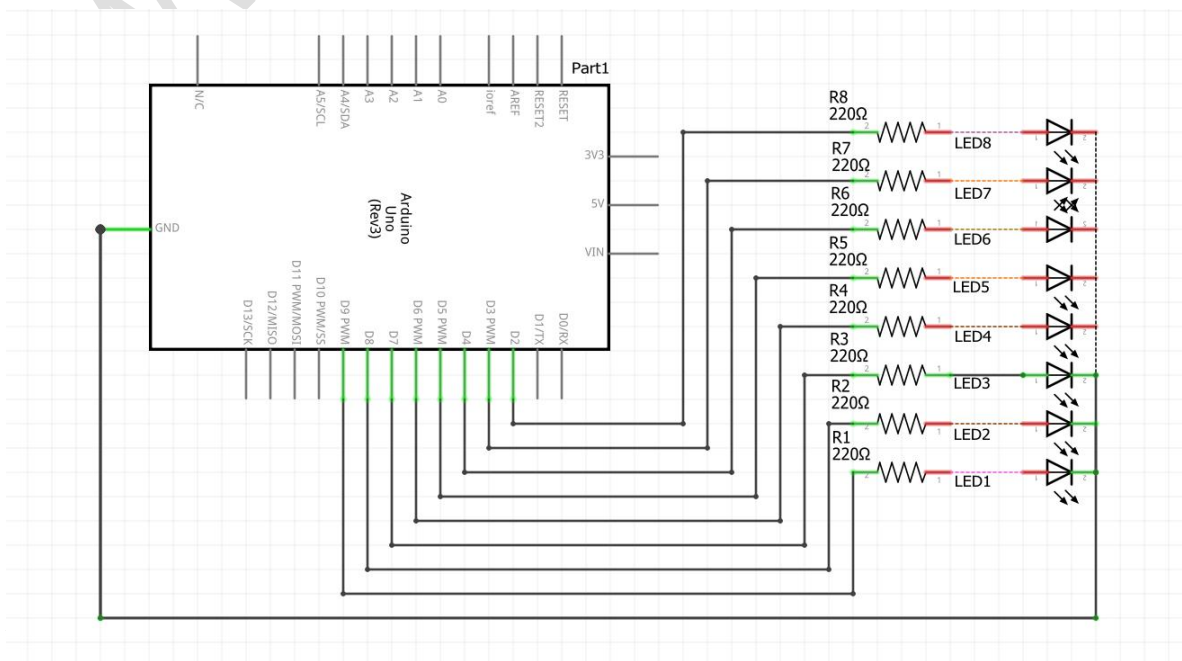
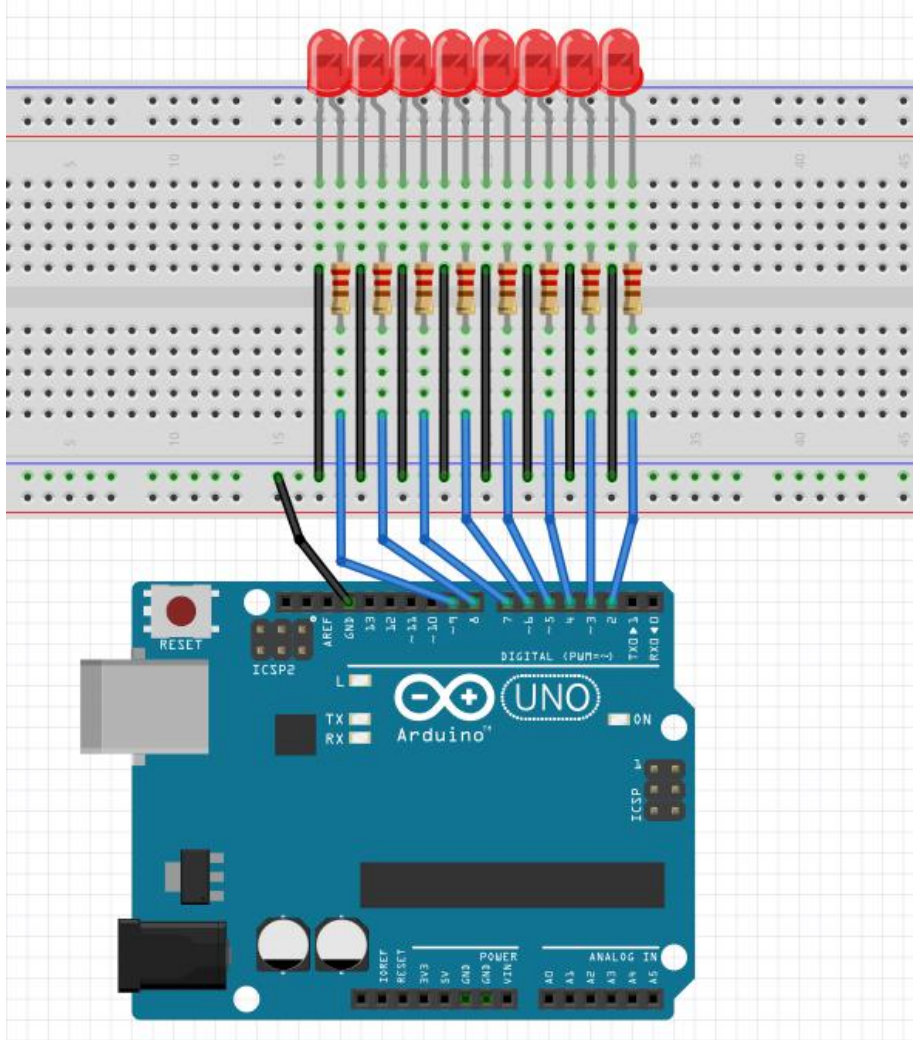
- Arduino Uno Board and Bread Board
- 8 x LEDs (light-emitting diode)



- 8 x 220 ohm resistors



## Circuit



## Code

```

LED_Effects
//set ledPin from 2 to 9
const int ledPinLowest = 2;
const int ledPinHighest = 9;
const int DELAYTIME = 200;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin ledPin as an output.
  for (int i=0; i<8; i++)
    pinMode(ledPinLowest + i, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  int i;

  for (i=0; i<8; i++) {
    digitalWrite(ledPinHighest - i, HIGH);    // turn the LED on one by one from left to right
    delay(DELAYTIME);                          // wait for 200ms
  }

  for (i=0; i<8; i++) {
    digitalWrite(ledPinLowest + i, LOW);       // turn the LED off one by one from right to left
    delay(DELAYTIME);                          // wait for 200ms
  }

  for (i=0; i<8; i++) {
    digitalWrite(ledPinHighest - i, HIGH);    // turn the LED on one by one from left to right
    if (i != 0)                                // if not the first LED
      digitalWrite(ledPinHighest - i + 1, LOW); // turn the pre-LED off


    delay(DELAYTIME);                          // wait for 200ms
  }

  for (i=0; i<8; i++) {
    digitalWrite(ledPinLowest + i, HIGH);     // turn the LED on one by one from right to left
    if (i != 0)                                // if not the first LED
      digitalWrite(ledPinLowest + i - 1, LOW); // turn the next LED off

    delay(DELAYTIME);                          // wait for 200ms
  }
}

```

## Compile and Upload

Click  (upload button) to compile the code and then upload to Uno board.

## Result

Show various of LED display effect.

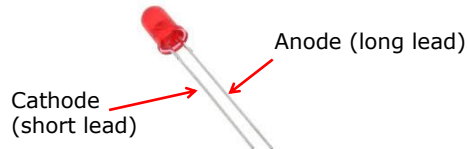


### Example 3 - Button

Button as a input device. This example turns on and off a light emitting diode(LED) connected to digital pin 13, when pressing a pushbutton attached to pin 2.

#### Hardware Required

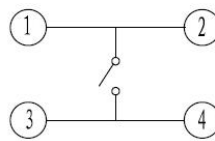
- Arduino Uno Board and Bread Board
- 1 x LEDs (light-emitting diode)



- 1 x 220 ohm resistors and 1 10K resistor
- 1 Button/Tact Switch

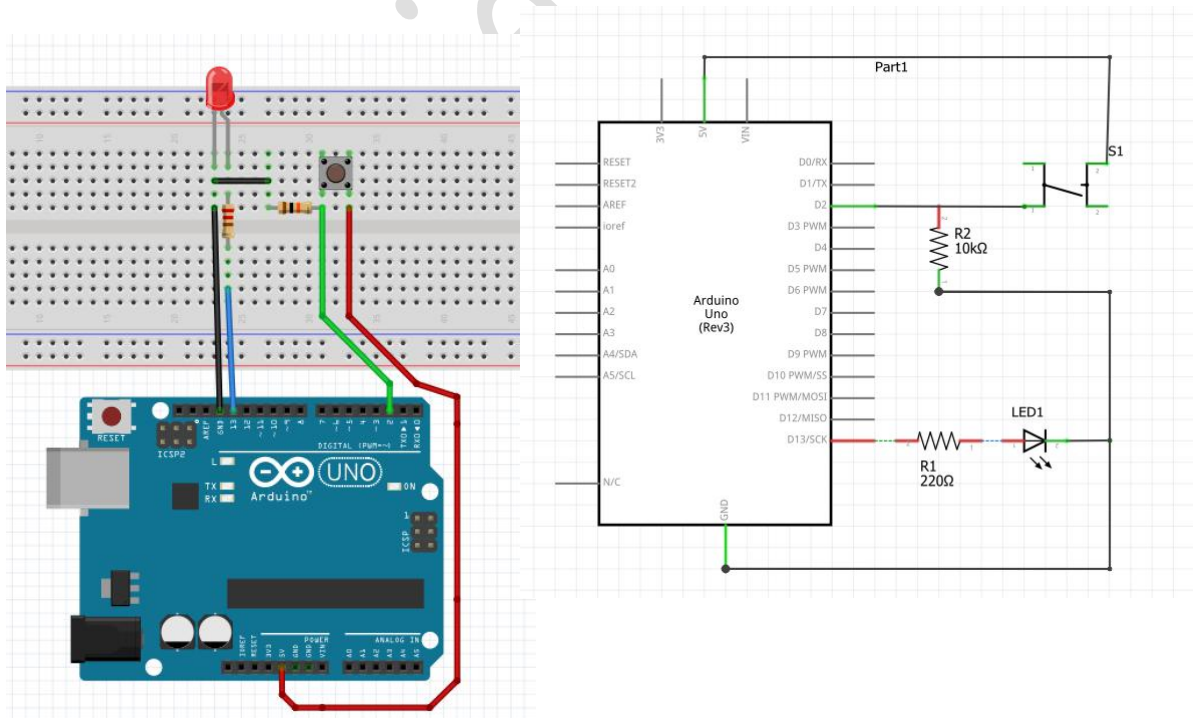


Tact Switch



Circuit

#### Circuit





## Code

```
Button §

/*
  Button
  Turns on and off a light emitting diode(LED) connected to digital pin 13,
  when pressing a pushbutton attached to pin 2.

  The circuit:
  - LED attached from pin 13 to ground
  - pushbutton attached to pin 2 from +5V
  - 10K resistor attached to pin 2 from ground

  - Note: on most Arduinos there is already an LED on the board
    attached to pin 13.
*/

// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;      // the number of the LED pin


// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

## Compile and Upload

Click  (upload button) to compile the code and then upload to Uno board.

## Result

LED turn on while pushbutton is pressed and LED turn off while button is release.

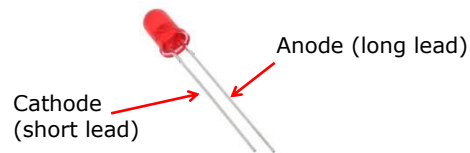
## Example 4 - Analog Input

Potentiometer as a input device. This example demonstrates analog input by reading an analog sensor on analog pin 0 and turning on and off a light emitting diode(LED) connected to digital pin 13.

The amount of time the LED will be on and off depends on the value obtained by analogRead().

### Hardware Required

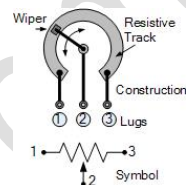
- Arduino Uno Board and Bread Board
- 1 x LEDs (light-emitting diode)



- 1 x 220 ohm resistors
- 1 Potentiometer/Variable Resistor

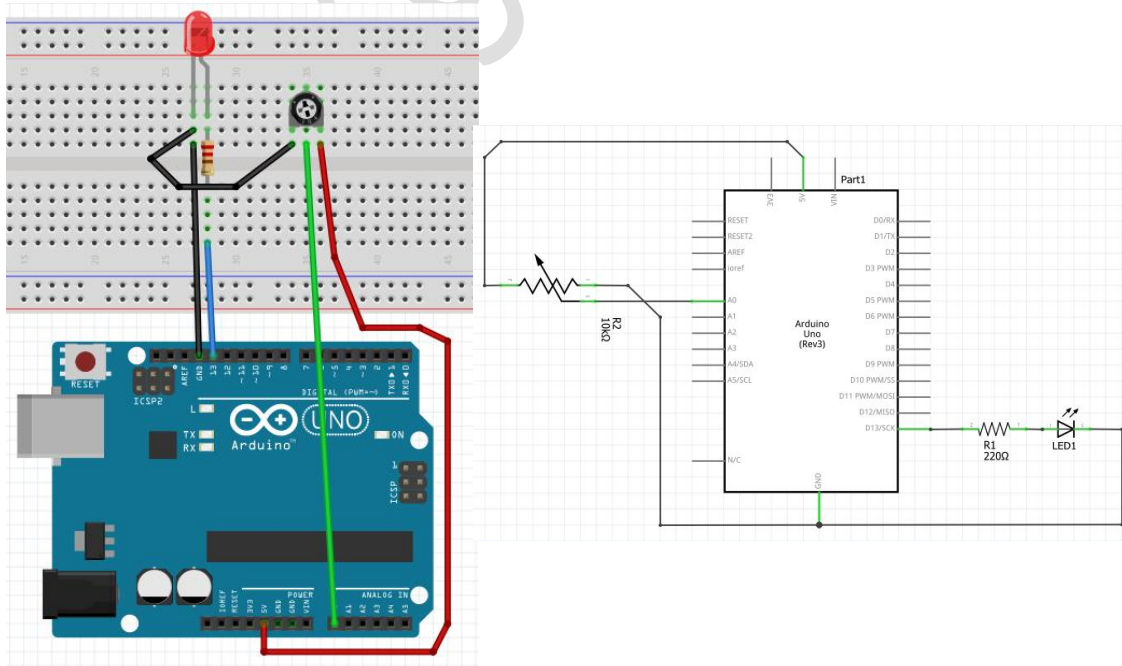


Potentiometer



Symbol

### Circuit



## Code



```

/*
  Analog Input

  Demonstrates analog input by reading an analog sensor on analog pin 0 and
  turning on and off a light emitting diode(LED) connected to digital pin 13.
  The amount of time the LED will be on and off depends on the value obtained
  by analogRead().

  The circuit:
  - potentiometer
    center pin of the potentiometer to the analog input 0
    one side pin (either one) to ground
    the other side pin to +5V
  - LED
    anode (long leg) attached to digital output 13
    cathode (short leg) attached to ground

  - Note: because most Arduinos have a built-in LED attached to pin 13 on the
    board, the LED is optional.

  created by David Cuatrecasas
  modified 30 Aug 2011
  By Tom Igoe

  This example code is in the public domain.


  http://www.arduino.cc/en/Tutorial/AnalogInput
*/

int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;       // select the pin for the LED
int sensorValue = 0;   // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
  
```

## Compile and Upload

Click  (upload button) to compile the code and then upload to Uno board.

## Result

LED will be on and off, the amount of on/off time will depends on the potentiometer value.

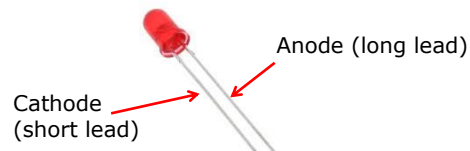
## Example 5 - Fading

This example demonstrates the use of the `analogWrite()` function in fading an LED off and on.

`AnalogWrite` uses pulse width modulation (PWM), turning a digital pin on and off very quickly with different ratio between on and off, to create a fading effect.

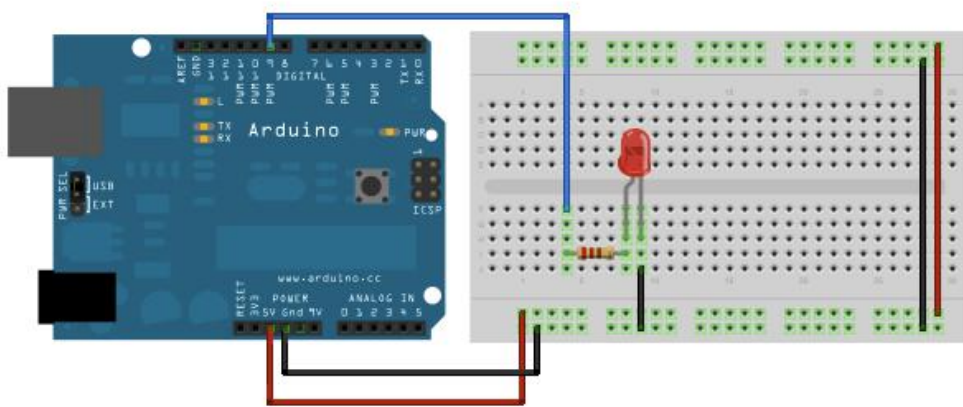
### Hardware Required

- Arduino Uno Board and Bread Board
- 1 x LEDs (light-emitting diode)

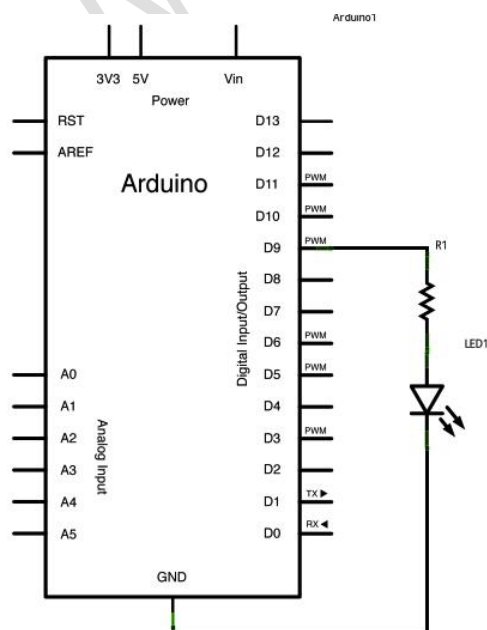


- 1 x 220 ohm resistors

### Circuit



### Schematic



## Code

```

sketch_sep15b $
/*
  Fade

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

  The analogWrite() function uses PWM, so if you want to change the pin you're
  using, be sure to use another PWM capable pin. On most Arduino, the PWM pins
  are identified with a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Fade
*/

int led = 9;          // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
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);
}

```

## Compile and Upload

Click  (upload button) to compile the code and then upload to Uno board.

## Result

LED brightness will change from dim to bright gradually and then change from bright to dim gradually, repeatedly.

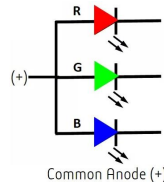
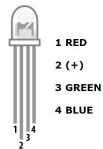
**\*Note: For more details, refer to <https://www.arduino.cc/en/tutorial/fade>**

## Example 6 - RGB LED

This example shows how to control the color of a RGB (Red, Green, Blue) LED with an Arduino by using analogWrite() function.

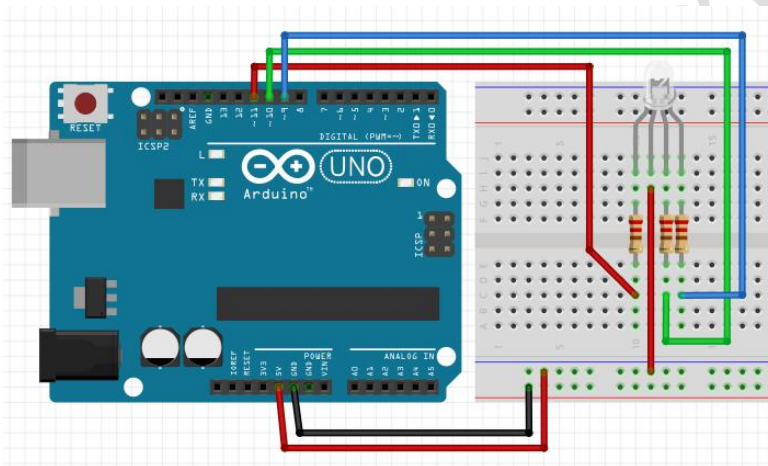
### Hardware Required

- Arduino Uno Board and Bread Board
- 1 x RGB (Red, Green, Blue) LEDs common anode

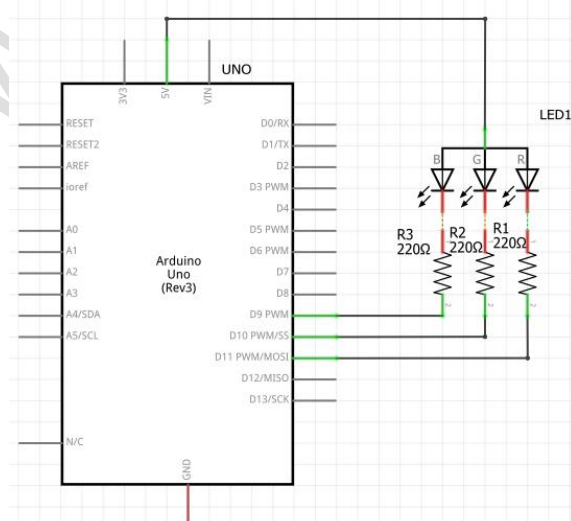


- 3 x 220 ohm resistors

### Circuit



### Schematic





## Code

```
RGB_LED

const int redPin = 11;
const int greenPin = 10;
const int bluePin = 9;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  showLEDColour(255, 0, 0); //red
  delay(1000);
  showLEDColour(0, 255, 0); //green
  delay(1000);
  showLEDColour(0, 0, 255); //blue
  delay(1000);
  showLEDColour(255, 255, 0); //yellow
  delay(1000);
  showLEDColour(80, 0, 80); //purple
  delay(1000);
  showLEDColour(0, 255, 255); //aqua
  delay(1000);
}

void showLEDColour(int red, int green, int blue) {
  red = 255 - red;
  green = 255 - green;
  blue = 255 - blue;

  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

## Compile and Upload

Click  (upload button) to compile the code and then upload to Uno board.

## Result

RGB LED will cycle through the colors red, green, blue, yellow, purple and aqua.

**\*Note:** For more details, please refer <https://learn.adafruit.com/adafruit-arduino-lesson-3-rgb-leds/overview>

### Example 7 - Buzzer

This example shows how to use a buzzer with Arduino. Buzzers can be found in alarm devices, computers, timers and confirmation of user input such as a mouse click or keystroke.

#### Hardware Required

- Arduino Uno Board
- 1 x Buzzer



- 1 x 100 ohm resistors

**\*Note:** For more details, please refer <https://www.instructables.com/id/How-to-use-a-Buzzer-Arduino-Tutorial/>