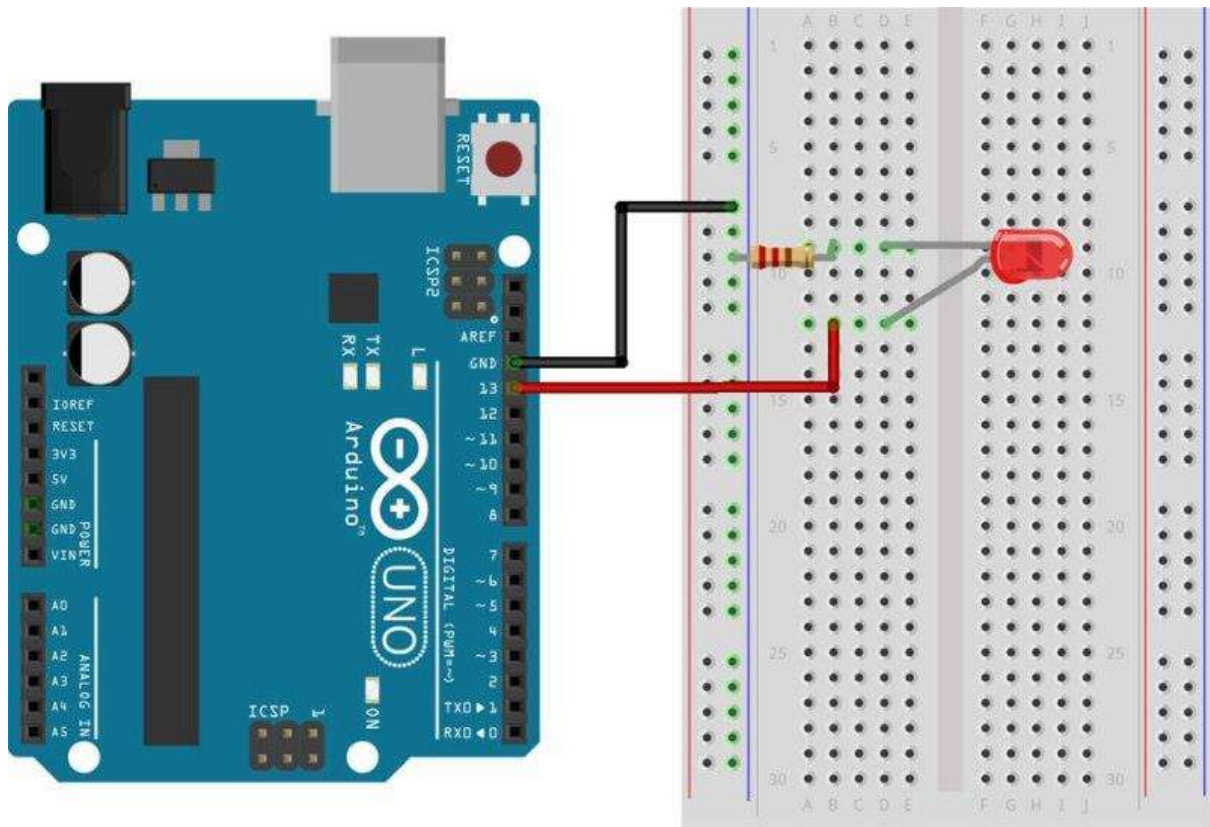


# Simple Arduino Projects For Beginners



## Arduino Projects

In this tutorial, we're going to help you create a few simple arduino projects that are perfect for beginners. These basic projects will help you understand how to set up the Arduino software and then connect the components to perform a specific action.

If you're completely brand new to Arduino, make sure you download our free ebook below. This guide was created for the absolute beginner and will help you to understand the Arduino board along with its parts and components.

# Tools and Parts Needed

In order to complete the projects in this tutorial, you'll need to make sure you have the following items.

- [Arduino Uno Board](#)
- [Breadboard](#) – half size
- [Jumper Wires](#)
- [USB Cable](#)
- [LED \(5mm\)](#)
- [Push button switch](#)
- [10k Ohm Resistor](#)
- [220 Ohm Resistor](#)

## Download The Software

At this point, we're ready to download the free software known as the IDE. The Arduino IDE is the interface where you will write the sketches that tell the board what to do.

You can find the latest version of this software on the [Arduino IDE download page](#).



**ARDUINO**

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

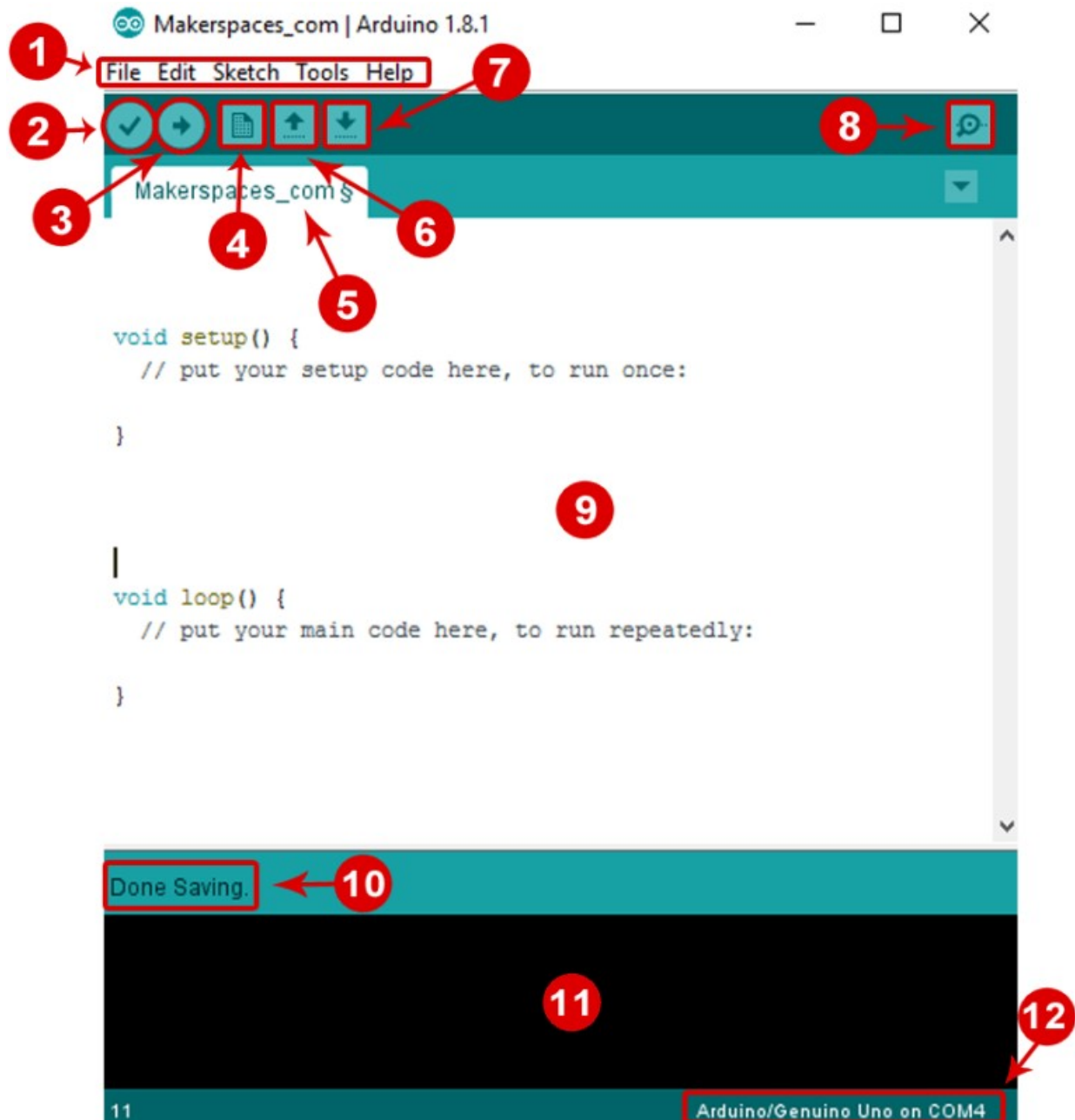
[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)

To install the software, you will need to click on the link that corresponds with your computer's operating system.

# Arduino IDE

Once the software has been installed on your computer, go ahead and open it up. This is the Arduino IDE and is the place where all the programming will happen.

Take some time to look around and get comfortable with it.

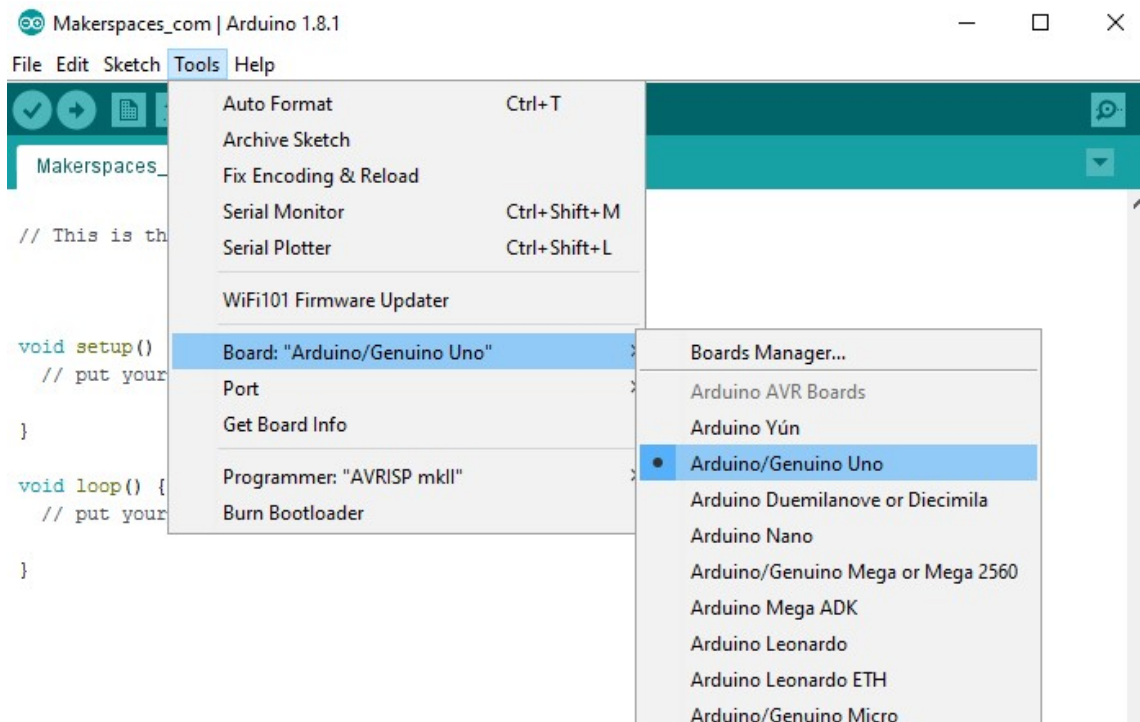


1. **Menu Bar:** Gives you access to the tools needed for creating and saving Arduino sketches.
2. **Verify Button:** Compiles your code and checks for errors in spelling or syntax.
3. **Upload Button:** Sends the code to the board that's connected such as Arduino Uno in this case. Lights on the board will blink rapidly when uploading.
4. **New Sketch:** Opens up a new window containing a blank sketch.
5. **Sketch Name:** When the sketch is saved, the name of the sketch is displayed here.
6. **Open Existing Sketch:** Allows you to open a saved sketch or one from the stored examples.
7. **Save Sketch:** This saves the sketch you currently have open.
8. **Serial Monitor:** When the board is connected, this will display the serial information of your Arduino
9. **Code Area:** This area is where you compose the code of the sketch that tells the board what to do.
10. **Message Area:** This area tells you the status on saving, code compiling, errors and more.
11. **Text Console:** Shows the details of an error messages, size of the program that was compiled and additional info.
12. **Board and Serial Port:** Tells you what board is being used and what serial port it's connected to.

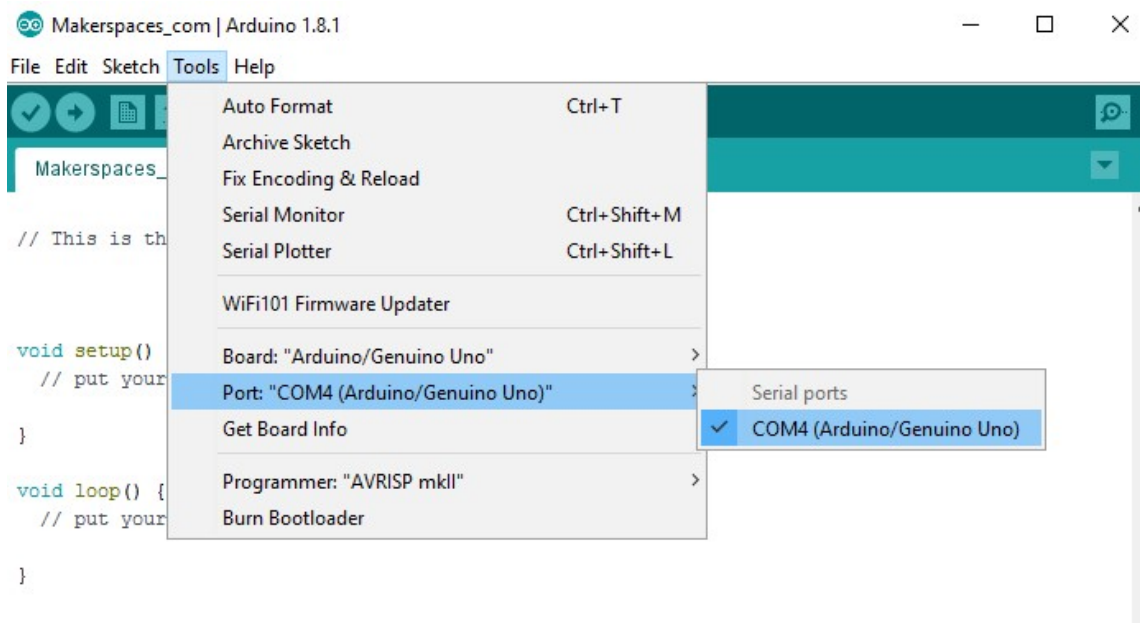
## Connect Your Arduino Uno

At this point you are ready to connect your Arduino to your computer. Plug one end of the USB cable to the Arduino Uno and then the other end of the USB to your computer's USB port.

Once the board is connected, you will need to go to **Tools** then **Board** then finally select **Arduino Uno**.



Next, you have to tell the Arduino which port you are using on your computer. To select the port, go to **Tools** then **Port** then select the port that says **Arduino**.

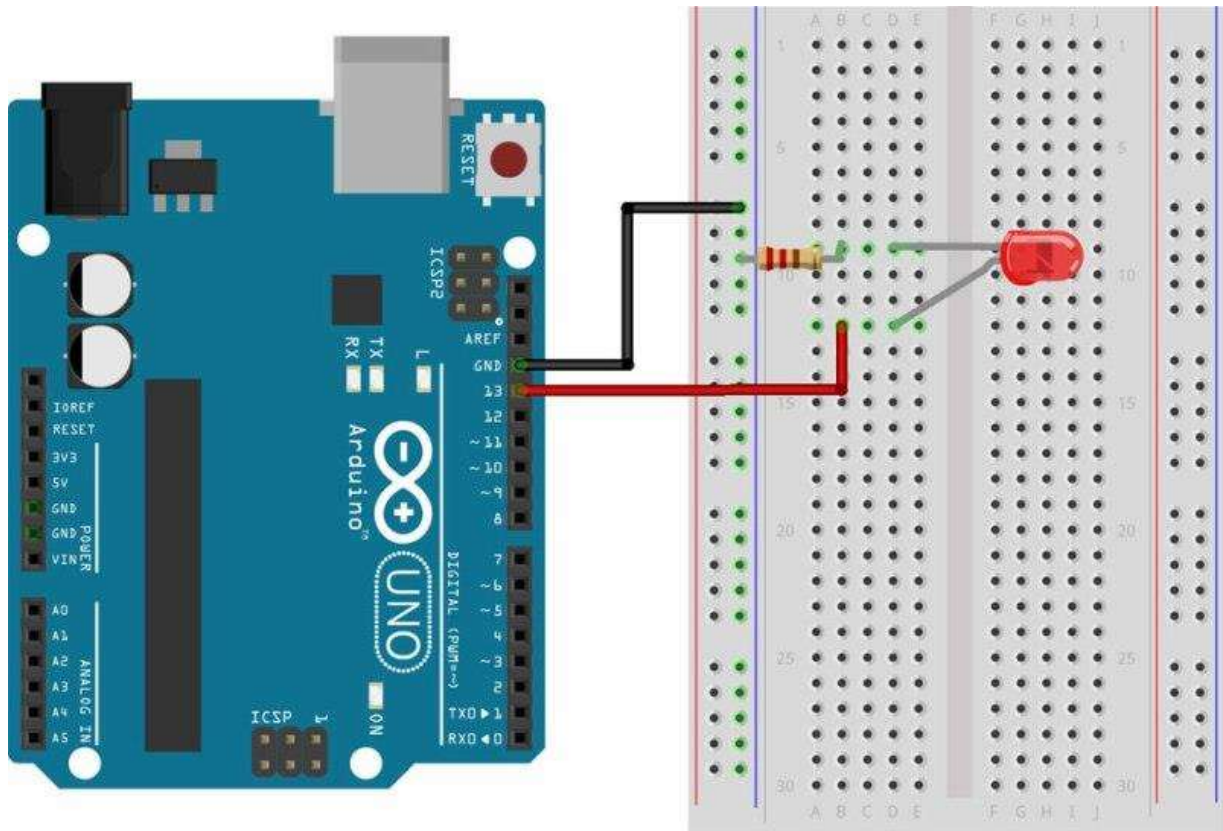




# Arduino Project 1: Blink an LED

It's finally time to do your first Arduino project. In this example, we are going to make your Arduino board blink an LED.

If you need a refresher on the parts of the Arduino or how a breadboard works, check out our previous tutorial called [Arduino For Beginners](#).



## Required Parts

- [Arduino Uno Board](#)
- [Breadboard](#) – half size
- [Jumper Wires](#)
- [USB Cable](#)
- [LED \(5mm\)](#)
- [220 Ohm Resistor](#)

## Connect The Parts

You can build your Arduino circuit by looking at the breadboard image above or by using the written description below. In the written description, we will use a letter/number combo that refers to the location of the component. If we mention H19 for example, that refers to column H, row 19 on the breadboard.

**Step 1** – Insert black jumper wire into the GND (Ground) pin on the Arduino and then in the GND rail of the breadboard row 15

**Step 2** – Insert red jumper wire into pin 13 on the Arduino and then the other end into F7 on the breadboard

**Step 3** – Place the LONG leg of the LED into H7

**Step 4** – Place the SHORT leg of the LED into H4

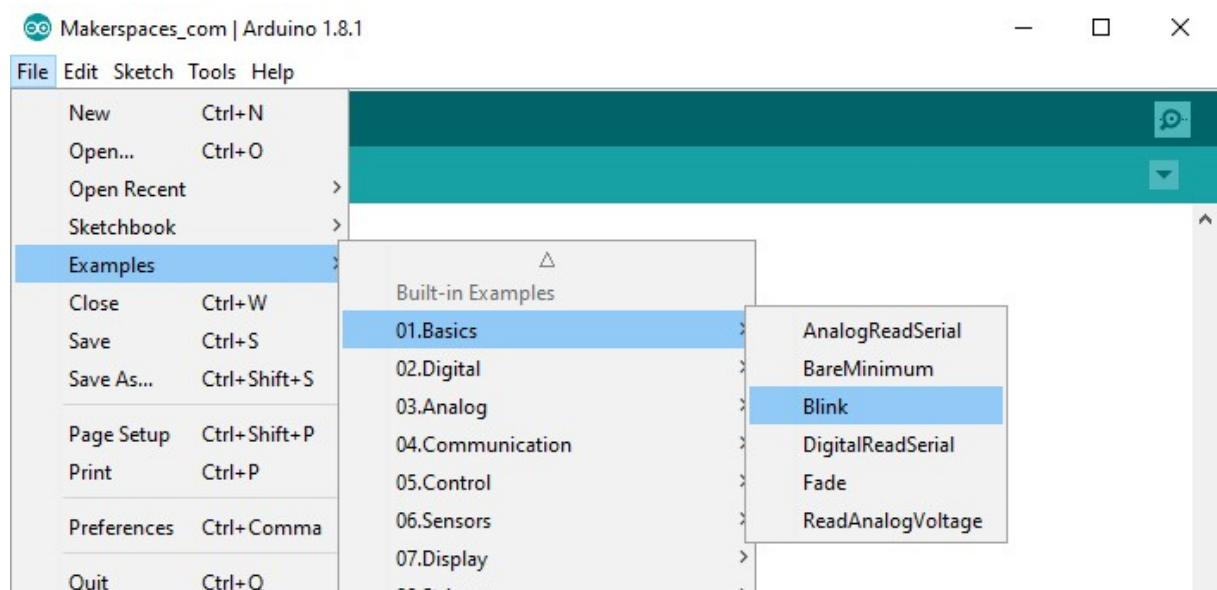
**Step 5** – Bend both legs of a 220 Ohm resistor and place one leg in the GND rail around row 4 and other leg in I4

**Step 6** – Connect the Arduino Uno to your computer via USB cable

## Upload The Blink Sketch

Now it's time to upload the sketch (program) to the Arduino and tell it what to do. In the IDE, there are built-in example sketches that you can use which make it easy for beginners.

To open the blink sketch, you will need to go to **File > Examples > Basics > Blink**



Now you should have a fully coded blink sketch that looks like the image below.



```

Blink
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, MEGA and ZERO
it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
the correct LED pin independent of which board is used.
If you want to know what pin the on-board LED is connected to on your Arduino model, check
the Technical Specs of your board at https://www.arduino.cc/en/Main/Products

This example code is in the public domain.

modified 8 May 2014
by Scott Fitzgerald

modified 2 Sep 2016
by Arturo Guadalupi

modified 8 Sep 2016
by Colby Newman
*/

// 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
}

```

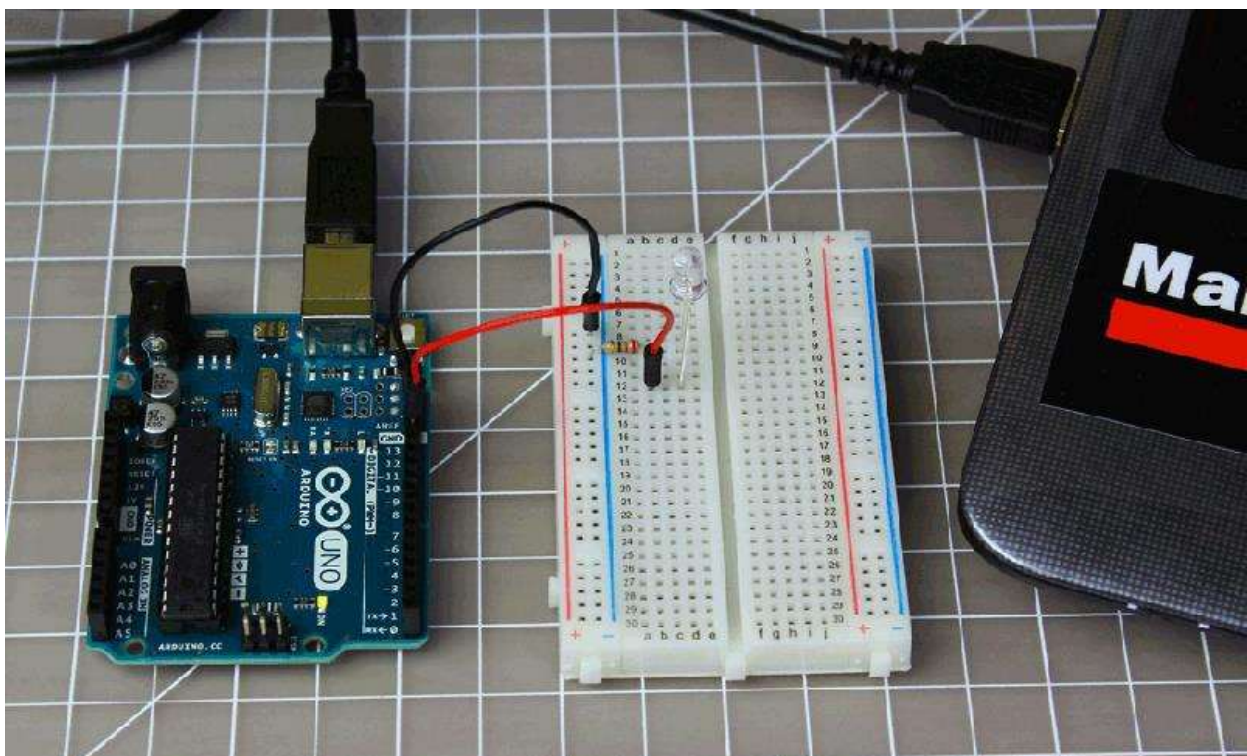
Arduino/Genuino Uno on COM3



Next, you need to click on the verify button (check mark) that's located in the top left of the IDE box. This will compile the sketch and look for errors. Once it says "Done Compiling" you are ready to upload it. Click the upload button (forward arrow) to send the program to the Arduino board.



The built-in LEDs on the Arduino board will flash rapidly for a few seconds and then the program will execute. If everything went correctly, the LED on the breadboard should turn on for a second and then off for a second and continue in a loop.



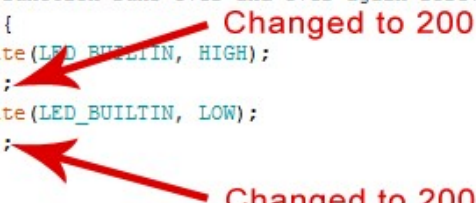
Congrats! You just completed your first Arduino project. **Troubleshooting** – If you ran into a problem don't give up, check out the troubleshooting section at the end for common ways to fix problems.

## Change The Code

Before we go to the next project, let's change some of the code in the "Blink" sketch to make it do something different. Playing around with the sketch will help you start to learn how the code controls the board.

```
// 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);
  delay(200);
  digitalWrite(LED_BUILTIN, LOW);
  delay(200);
}
|
```

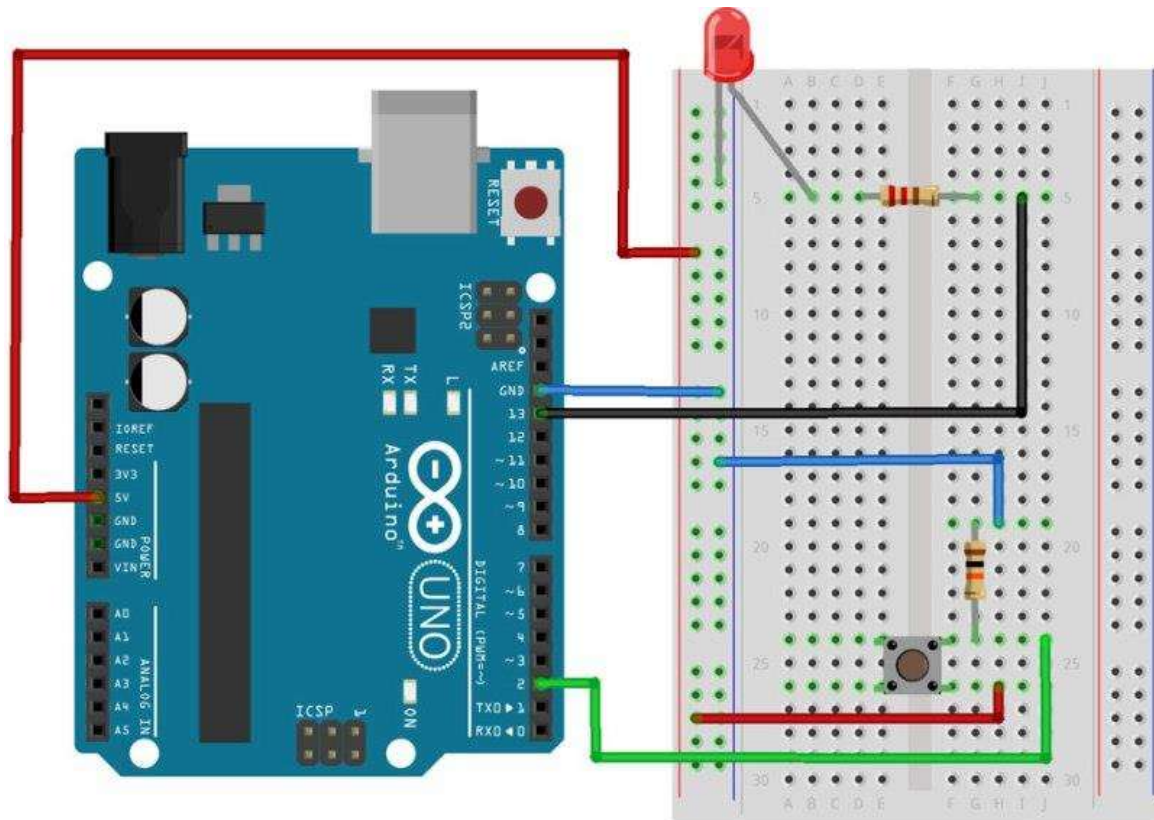


Keep the Arduino board connected and change the delay portion of the code from (1000) to (200). Click the verify button on the top left of the IDE and then click upload. This should make the LED on the breadboard blink faster.

**NOTE** – Arduino measures time in milliseconds and 1000 milliseconds = 1 second. The original code (1000) turns on the LED for 1 second and then off for 1 second. By adjusting the code from (1000) to (200) it shortens the time between on and off which makes it blink faster.

# Arduino Project 2: LED w/ Switch

Now it's time to talk switches and how they can be incorporated into Arduino projects. A switch is an electrical component that completes a circuit when pushed and breaks the circuit when released. In this project, we will be using a [small pushbutton switch](#) to control an LED.



## Required Parts

- [Arduino Uno Board](#)
- [Breadboard](#) – half size
- [Jumper Wires](#)
- [USB Cable](#)
- [LED \(5mm\)](#)
- [Push button switch](#)
- [10k Ohm Resistor](#)
- [220 Ohm Resistor](#)

## Connect The Parts

You can build your Arduino circuit by looking at the breadboard image above or by using the written description below. In the written description, we will use a letter/number combo that refers to the location of the component. If we mention H19 for example, that refers to column H, row 19 on the breadboard.

**Step 1** – Connect the blue jumper wire from the GND on the Arduino to the GND rail (blue line) on the breadboard near A13

**Step 2** – Connect the blue jumper wire from the GND rail on the breadboard near A17 to H19

**Step 3** – Connect the red jumper wire from the power rail on the breadboard around row A27 to H26

**Step 4** – Connect the green jumper wire from pin 2 on Arduino to J24 on the breadboard

**Step 5** – Place one leg of a 10k Ohm resistor in G19 and the other leg in G24

**Step 6** – Place the pushbutton switch into F24, F26, E24 and E26

**Step 7** – Place one leg of a 220 Ohm resistor in D5 and the other leg in G5

**Step 8** – Insert the short leg of the LED in the GND rail around A5 and the long leg in B5

**Step 9** – Connect the black jumper wire from pin 13 on the Arduino to I5 on the breadboard

**Step 10** – Connect the red jumper wire from 5V on the Arduino to power rail (+) near A8

**Step 11** – Connect the Arduino Uno to your computer via USB cable

## Upload The Switch Sketch

Now it's time to upload the sketch to the Arduino that will allow us to use a switch. As with the blink sketch, there are example programs already loaded in the Arduino IDE that we will be using.

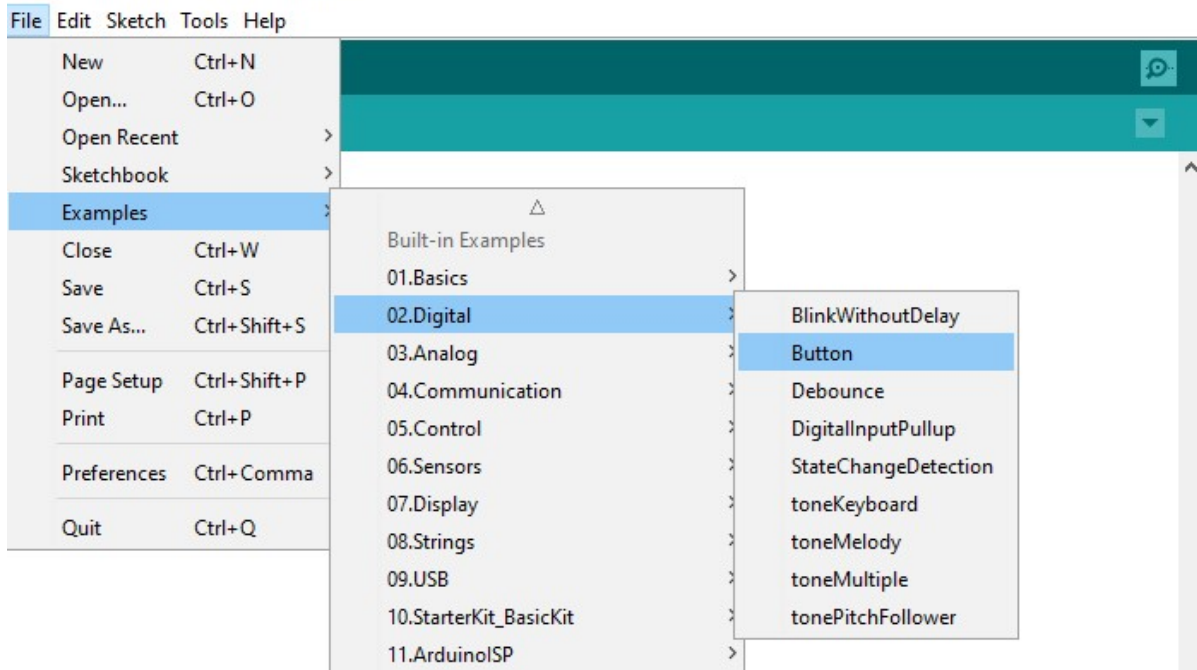
In order to use a switch, we have to load the file called "Button" which can be found here: **File > Examples > Digital > Button**

File Edit Sketch Tools Help

- New Ctrl+N
- Open... Ctrl+O
- Open Recent >
- Sketchbook >
- Examples** >
- Close Ctrl+W
- Save Ctrl+S
- Save As... Ctrl+Shift+S
- Page Setup Ctrl+Shift+P
- Print Ctrl+P
- Preferences Ctrl+Comma
- Quit Ctrl+Q

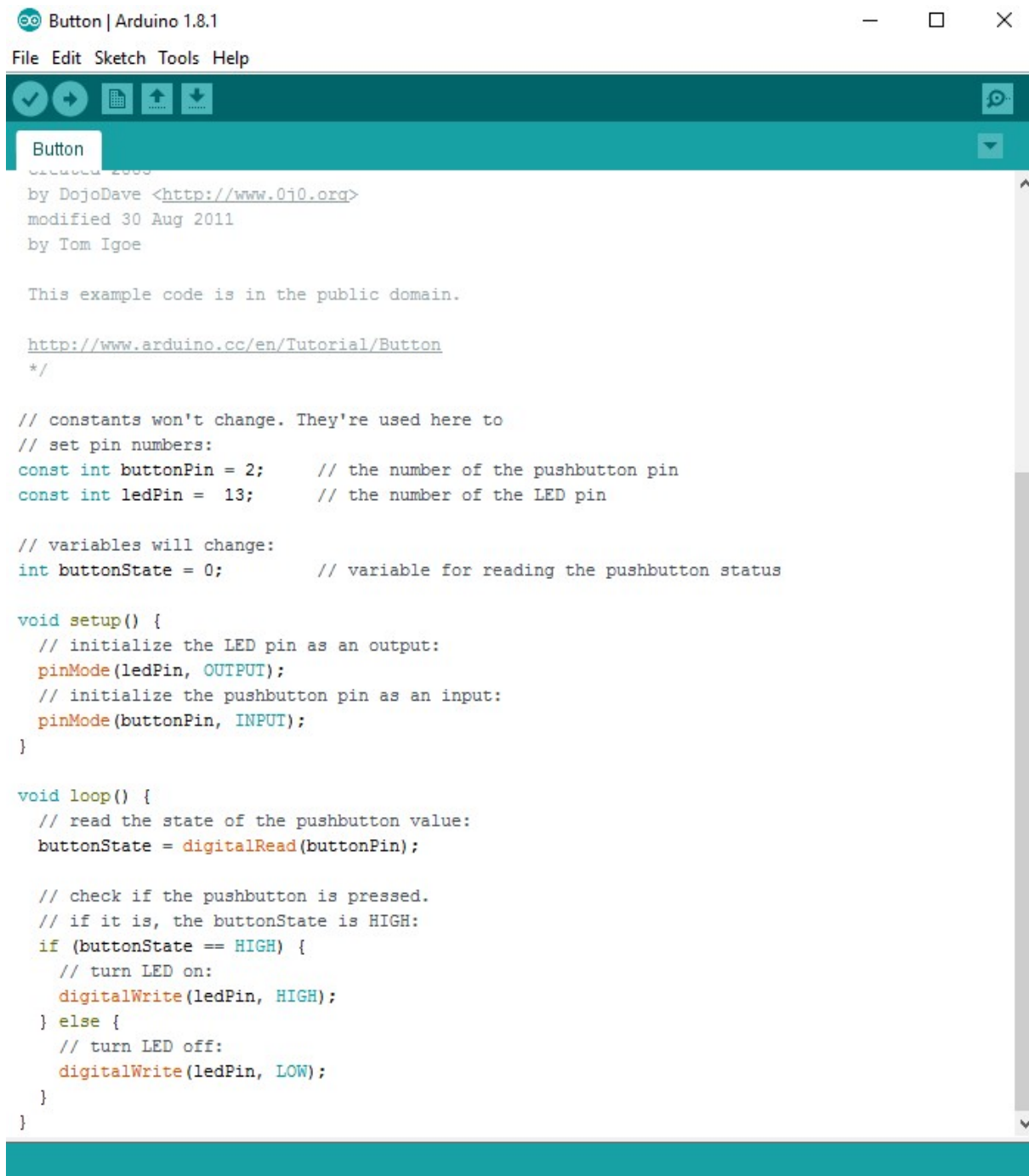
- Built-in Examples
- 01.Basics >
- 02.Digital** >
- 03.Analog >
- 04.Communication >
- 05.Control >
- 06.Sensors >
- 07.Display >
- 08.Strings >
- 09.USB >
- 10.StarterKit\_BasicKit >
- 11.ArduinoISP >

- BlinkWithoutDelay
- Button**
- Debounce
- DigitalInputPullup
- StateChangeDetection
- toneKeyboard
- toneMelody
- toneMultiple
- tonePitchFollower





Now you should have a fully coded button sketch that looks like the image below.



```
Button | Arduino 1.8.1
File Edit Sketch Tools Help

Button
Created 2009
by DojoDave <http://www.0j0.org>
modified 30 Aug 2011
by Tom Igoe

This example code is in the public domain.

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

// 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);
  }
}
```



Next, you need to click on the verify button (check mark) that's located in the top left of the IDE box. Once it says "Done Compiling" you are ready to upload it. Click the upload button (forward arrow) to send the program to the Arduino board.

Press the button switch on the breadboard and you should be able to turn on and off the LED as shown in this [Youtube video](#).

## Troubleshooting

If you are having any problems with the projects we did, make sure the following has been checked.

1. Verify the LED is actually functional. Use a 3v coin cell battery and connect the LONG leg of the LED to the (+) and SHORT leg to the (-) of the battery.
2. Verify the correct leg of the LED is connected properly. LONG leg to positive and SHORT leg to negative.
3. Make sure the Arduino IDE shows the correct board. Go to **Tools > Board** then select **Arduino Uno**.
4. Make sure the Arduino IDE shows the correct port. Go to **Tools > Port** then select the port that says **Arduino**.
5. Verify all component connections are secure with the Arduino board and breadboard.

## Resources

- This [instructable](#) and [LED calculator](#) will help you determine which size resistor to use for projects involving LEDs
- This [resistor color code calculator](#) will help you decode what size resistor you have based on the color bands
- Download our FREE Ebook (PDF) – [Beginners Guide to Arduino](#) for more info on the basics of Arduino