Arduino For Beginners

More and more makerspaces around the world are looking to add coding and electronics to their maker education programs. One of the best ways to do this is by integrating an Arduino board into makerspace projects and lessons.
We've found that a lot of maker educators haven't taken the plunge into coding or Arduino because they think programming is scary. Because of this, we wanted to make sure this tutorial was written for the absolute beginner with no experience whatsoever.

This tutorial is a high level view of all the parts and pieces of the Arduino ecosystem. In future posts, we will take you step by step in creating your first simple Arduino project.

What Is Arduino?

Arduino is an open source programmable circuit board that can be integrated into a wide variety of makerspace projects both simple and complex. This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world.   By responding to sensors and inputs, the Arduino is able to interact with a large array of outputs such as LEDs, motors and displays. Because of it's flexibility and low cost, Arduino has become a very popular choice for makers and makerspaces looking to create interactive hardware projects.
Arduino was introduced back in 2005 in Italy by Massimo Banzi as a way for non-engineers to have access to a low cost, simple tool for creating hardware projects. Since the board is open-source, it is released under a Creative Commons license which allows anyone to produce their own board. If you search the web, you will find there are hundreds of Arduino compatible clones and variations available but the only official boards have Arduino in it's name.
In the next section, we're going to discuss a few of the Arduino boards available and how they differ from each other.

Types of Arduino Boards

Arduino is a great platform for prototyping projects and inventions but can be confusing when having to choose the right board. If you're brand new to this, you might have always thought that there was just one "Arduino" board and that's it. In reality, there are many variations of the official Arduino boards and then there are hundreds more from competitors who offer clones. But don't worry, we're going to show you which one to start with later on in this tutorial.

Below are a few examples of the different types of Arduino boards out there. The boards with the name Arduino on them are the official boards but there are also a lot of really great clones on the market as well. One of the best reasons to buy a clone is the fact they are generally less expensive than their official counterpart. Adafruit and Sparkfun for example, sell variations of the Arduino boards which cost less but still have the same quality of the originals. One word of caution, be careful when buying boards from companies you don't know.

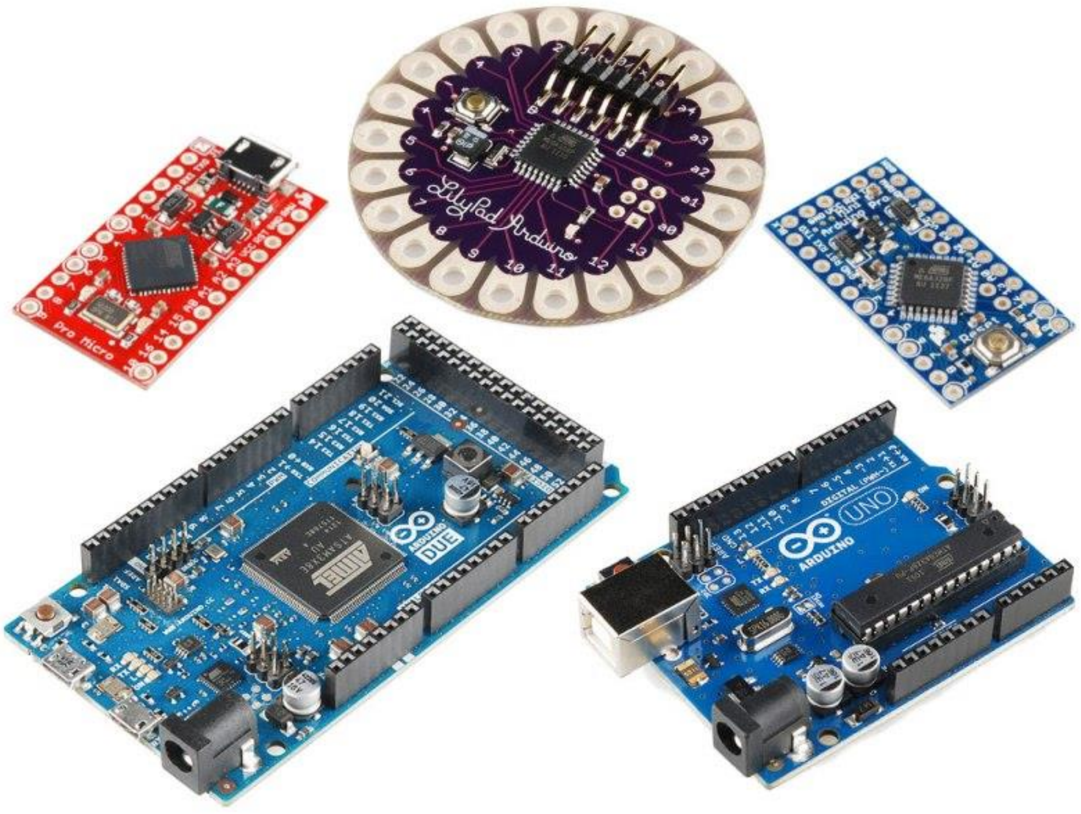


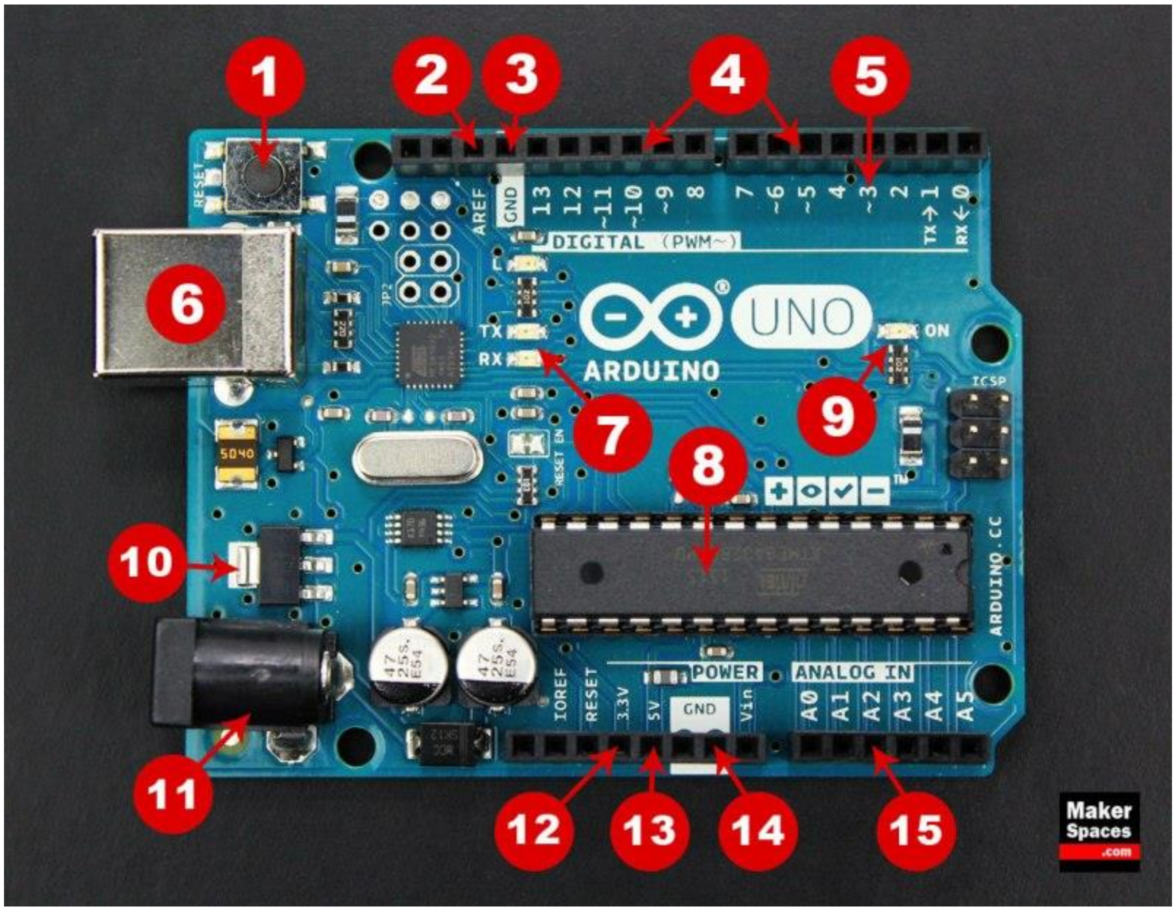
Image credit – Sparkfun.com

Another factor to consider when choosing a board is the type of project you are looking to do. For example, if you want to create a wearable electronic project, you

might want to consider the LilyPad board from Sparkfun. The LilyPad is designed to be easily sewn into e-textiles and wearable projects. If your project has a small form factor, you might want to use the Arduino Pro Mini which has a very small footprint compared to other boards. Check out Sparkfun's Arduino Comparison Guide for a breakdown and comparison of the top boards out there.
Next, we're going to focus on our favorite Arduino board which we recommend beginners start with.

Arduino Uno

One of the most popular Arduino boards out there is the Arduino Uno. While it was not actually the first board to be released, it remains to be the most actively used and most widely documented on the market. Because of its extreme popularity, the Arduino Uno has a ton of project tutorials and forums around the web that can help you get started or out of a jam. We're big fans of the Uno because of it's great features and ease of use.



Board Breakdown

Here are the components that make up an Arduino board and what each of their functions are.

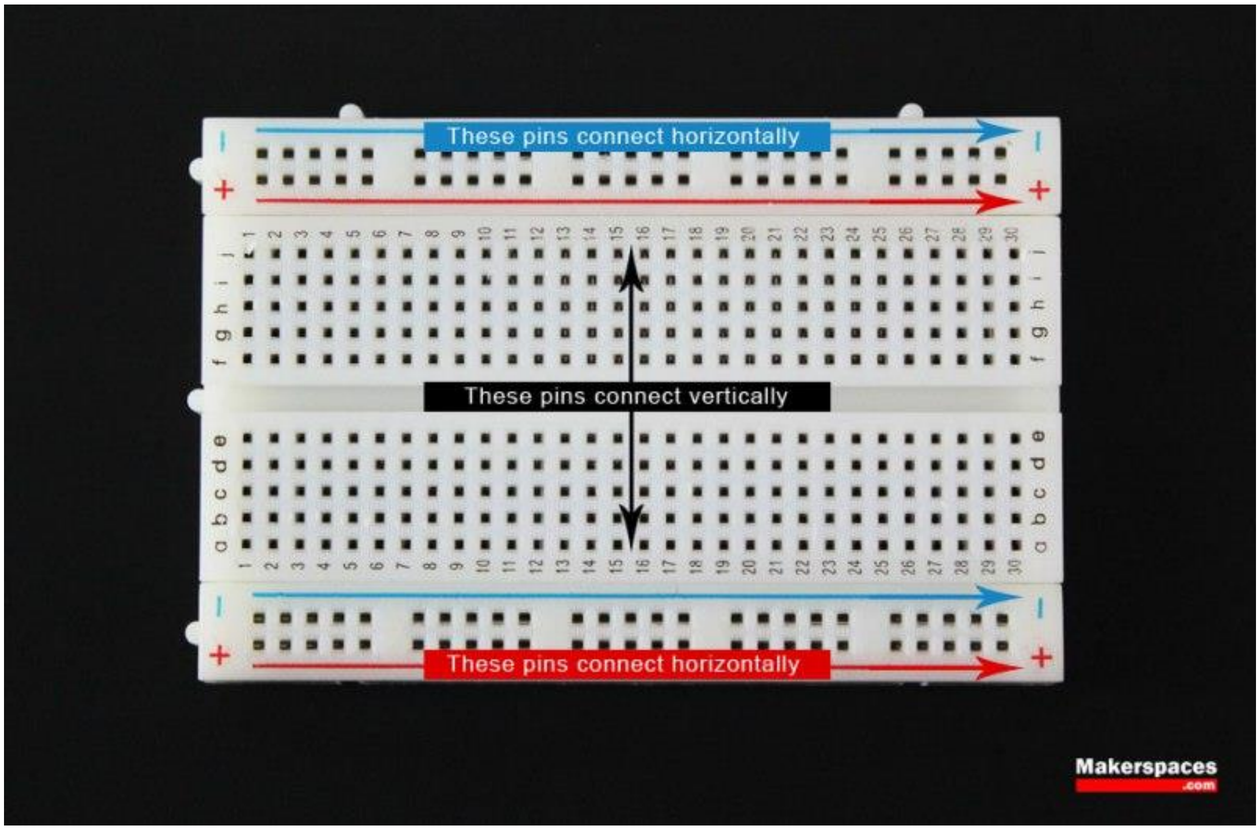
1. **Reset Button** – This will restart any code that is loaded to the Arduino board
2. **AREF** – Stands for “Analog Reference” and is used to set an external reference voltage
3. **Ground Pin** – There are a few ground pins on the Arduino and they all work the same
4. **Digital Input/Output** – Pins 0-13 can be used for digital input or output
5. **PWM** – The pins marked with the (~) symbol can simulate analog output
6. **USB Connection** – Used for powering up your Arduino and uploading sketches
7. **TX/RX** – Transmit and receive data indication LEDs
8. **ATmega Microcontroller** – This is the brains and is where the programs are stored
9. **Power LED Indicator** – This LED lights up anytime the board is plugged in a power source
10. **Voltage Regulator** – This controls the amount of voltage going into the Arduino board
11. **DC Power Barrel Jack** – This is used for powering your Arduino with a power supply
12. **3.3V Pin** – This pin supplies 3.3 volts of power to your projects
13. **5V Pin** – This pin supplies 5 volts of power to your projects
14. **Ground Pins** – There are a few ground pins on the Arduino and they all work the same
15. **Analog Pins** – These pins can read the signal from an analog sensor and convert it to digital

Arduino Power Supply

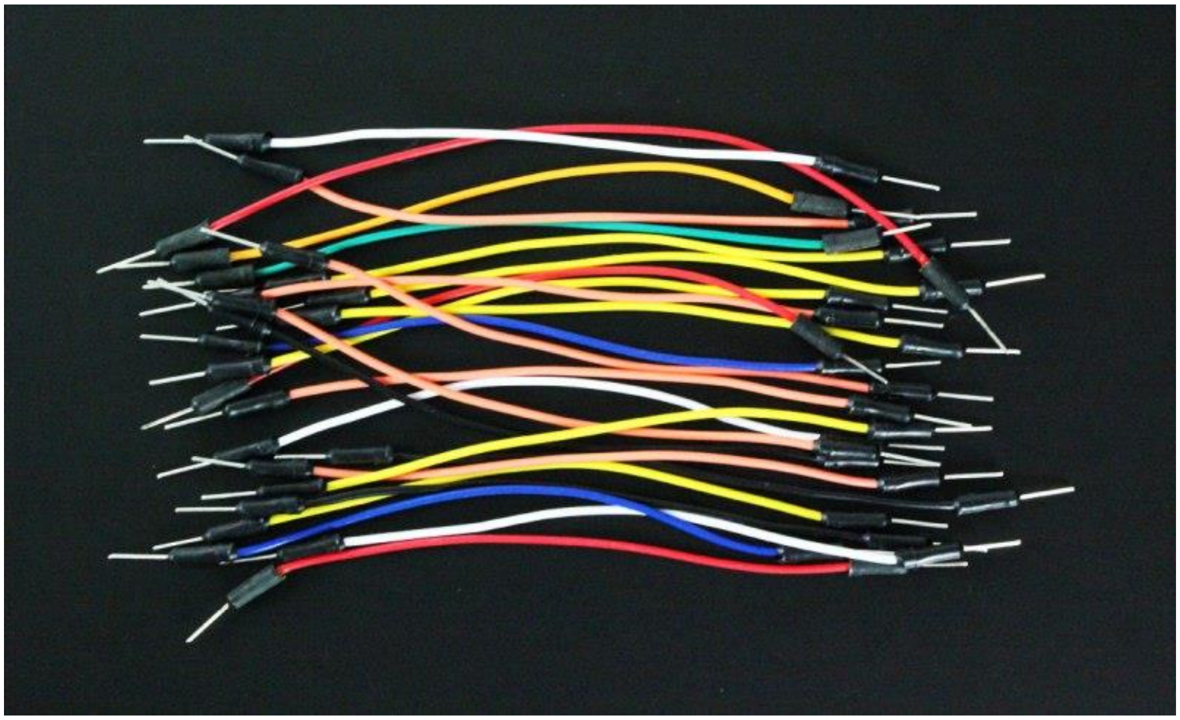
The Arduino Uno needs a power source in order for it to operate and can be powered in a variety of ways. You can do what most people do and connect the board directly to your computer via a USB cable. If you want your project to be mobile, consider using a 9V battery pack to give it juice. The last method would be to use a 9V AC power supply.

Arduino Breadboard

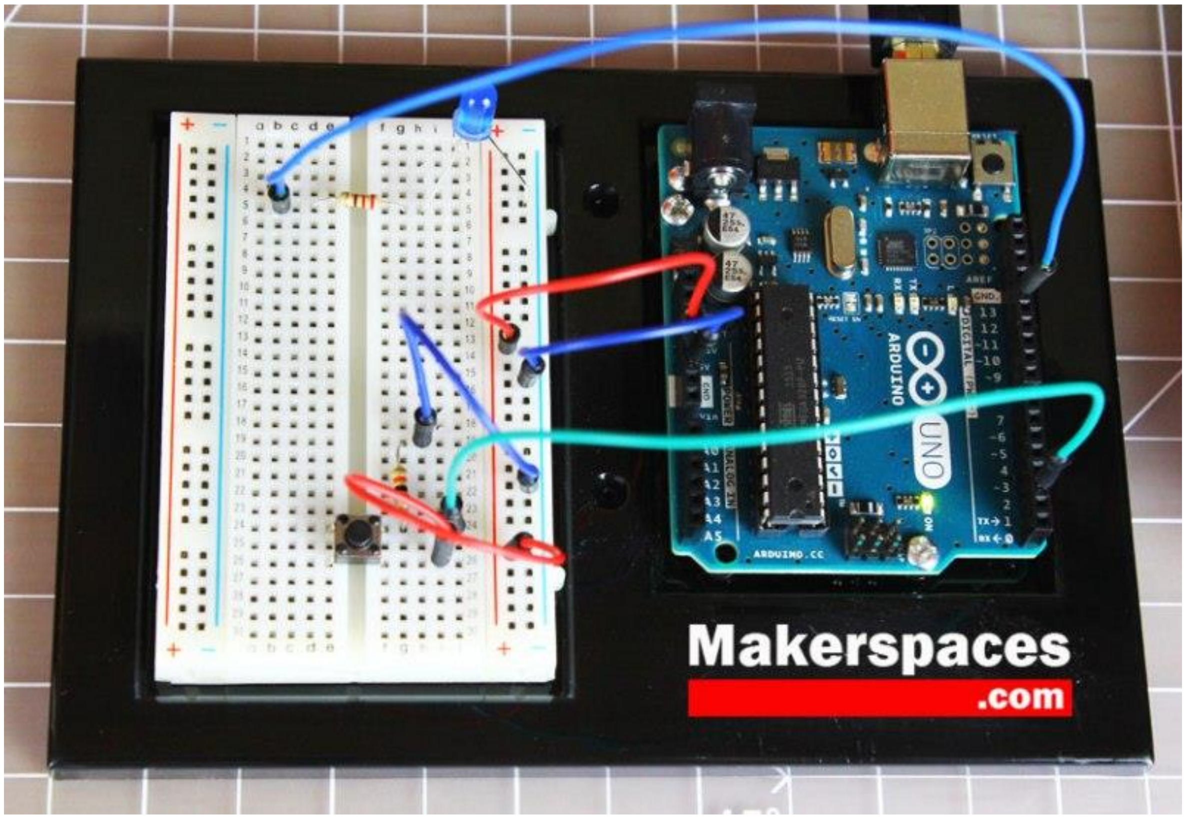
Another very important item when working with Arduino is a solderless breadboard. This device allows you to prototype your Arduino project without having to permanently solder the circuit together. Using a breadboard allows you to create temporary prototypes and experiment with different circuit designs. Inside the holes (tie points) of the plastic housing, are metal clips which are connected to each other by strips of conductive material.


These pins connect horizontally
These pins connect vertically
These pins connect horizontally
Makerspaces .com

On a side note, the breadboard is not powered on its own and needs power brought to it from the Arduino board using jumper wires. These wires are also used to form the circuit by connecting resistors, switches and other components together.

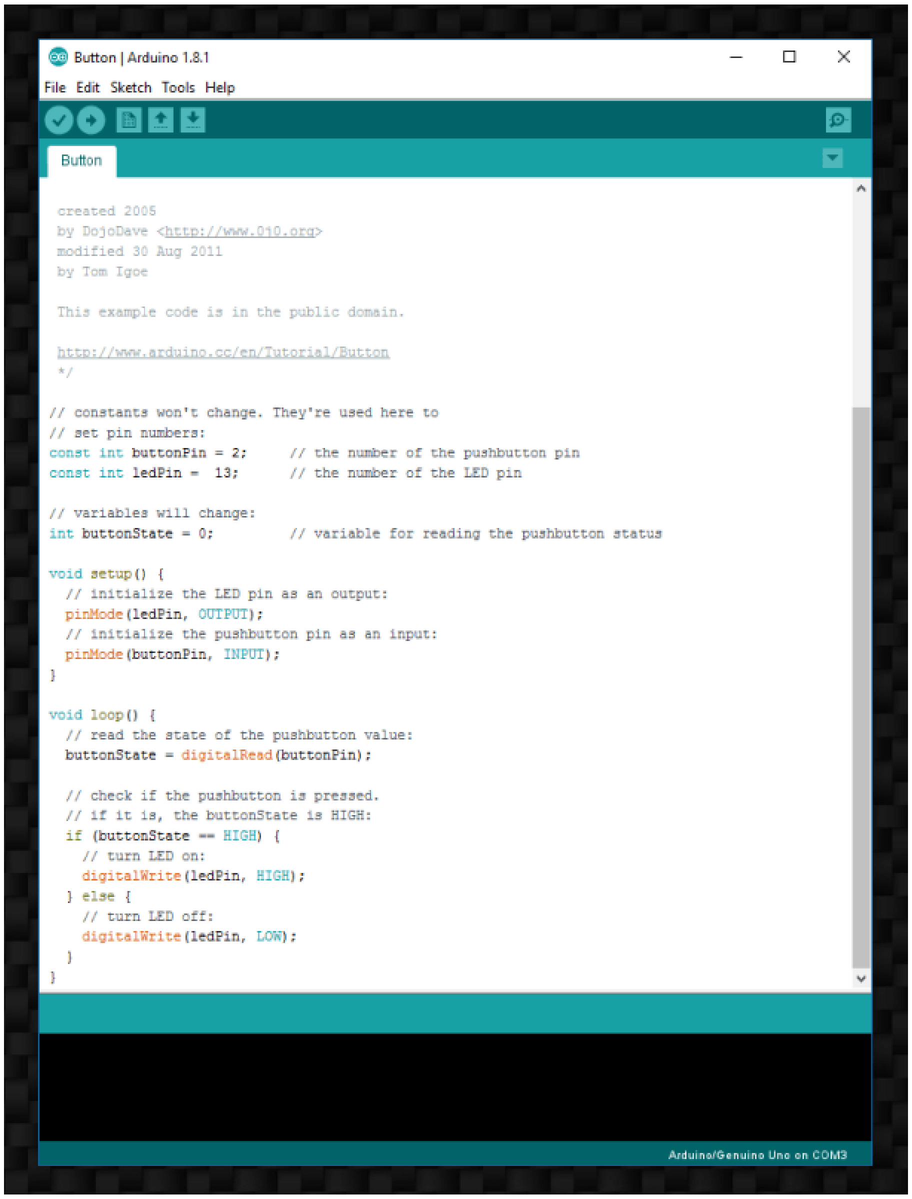


Here is a visual of what a completed Arduino circuit looks like when connected to a breadboard.



How To Program Arduino

Once the circuit has been created on the breadboard, you’ll need to upload the program (known as a sketch) to the Arduino. The sketch is a set of instructions that tells the board what functions it needs to perform. An Arduino board can only hold and perform one sketch at a time. The software used to create Arduino sketches is called the IDE which stands for Integrated Development Environment. The software is free to download and can be found at https://www.arduino.cc/en/Main/Software



```
 created 2005
 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);
  }
}
```

Every Arduino sketch has two main parts to the program:

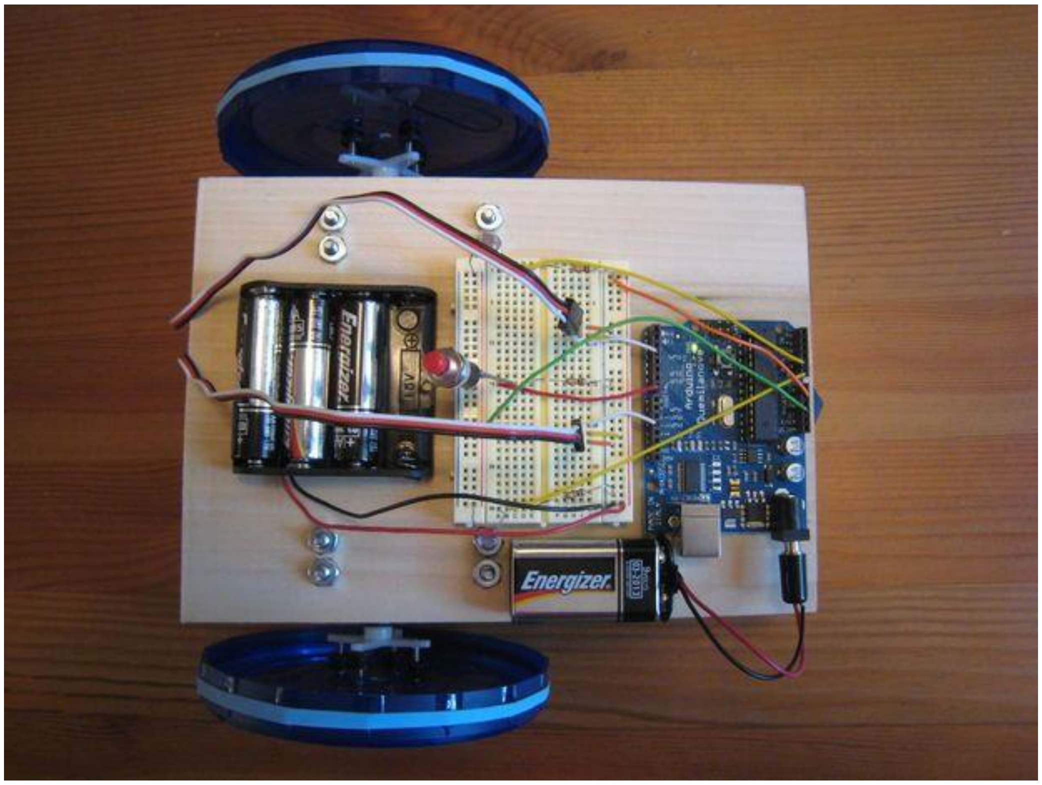
void setup() – Sets things up that have to be done once and then don't happen again.
void loop() – Contains the instructions that get repeated over and over until the board is turned off.

How Everything Works Together

Check out our quick Arduino video to see how a breadboard, Arduino, jumper wires and the sketch work together to perform a function. In this video, we use a momentary push button switch to blink an LED.

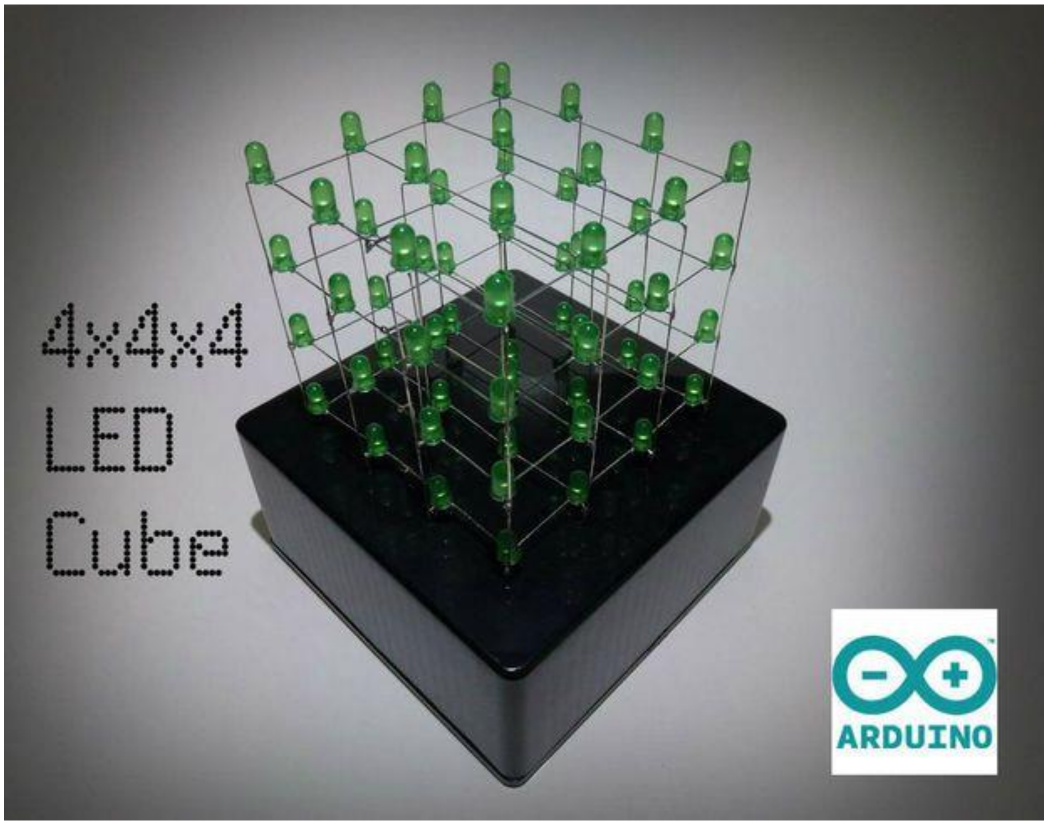
Arduino Projects

You may be wondering what an Arduino board can do besides blink an LED. Below are some example projects which help to showcase how truly amazing this board is and the capabilities of it. If you're looking for more project ideas, check out sites such as Instructables or Make Magazine which are loaded with helpful tutorials.

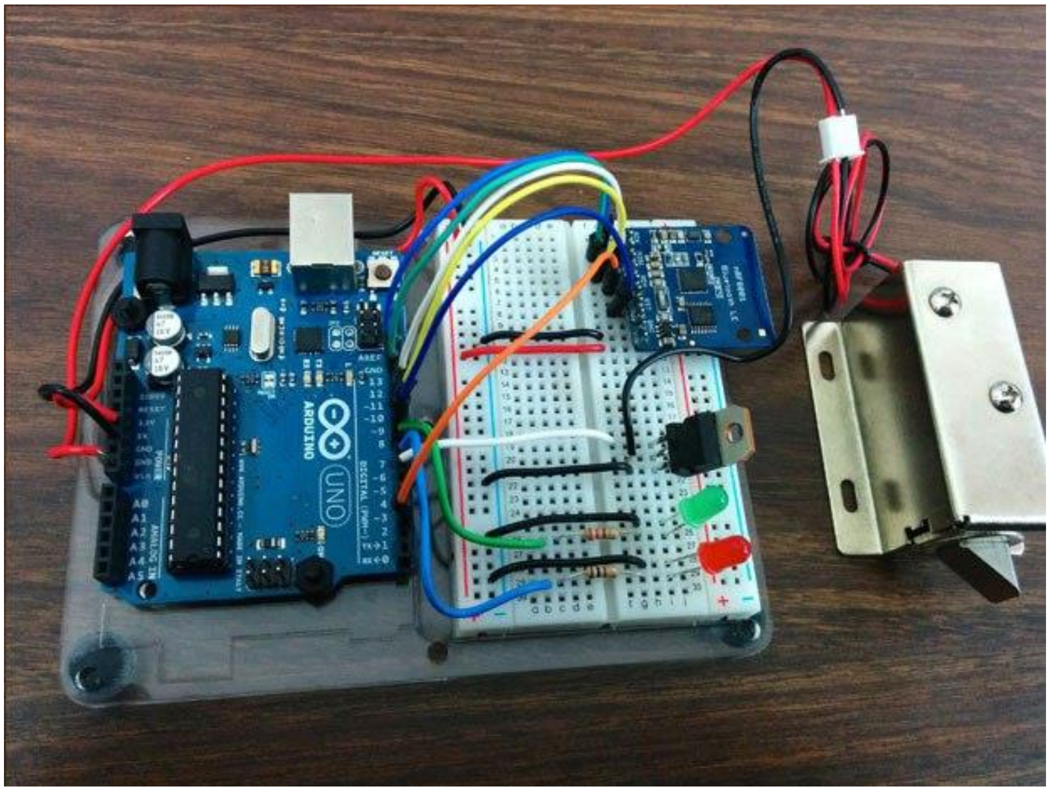


Arduino Light Following Robot – Instructables

Arduino Drone That Follows You – Instructables



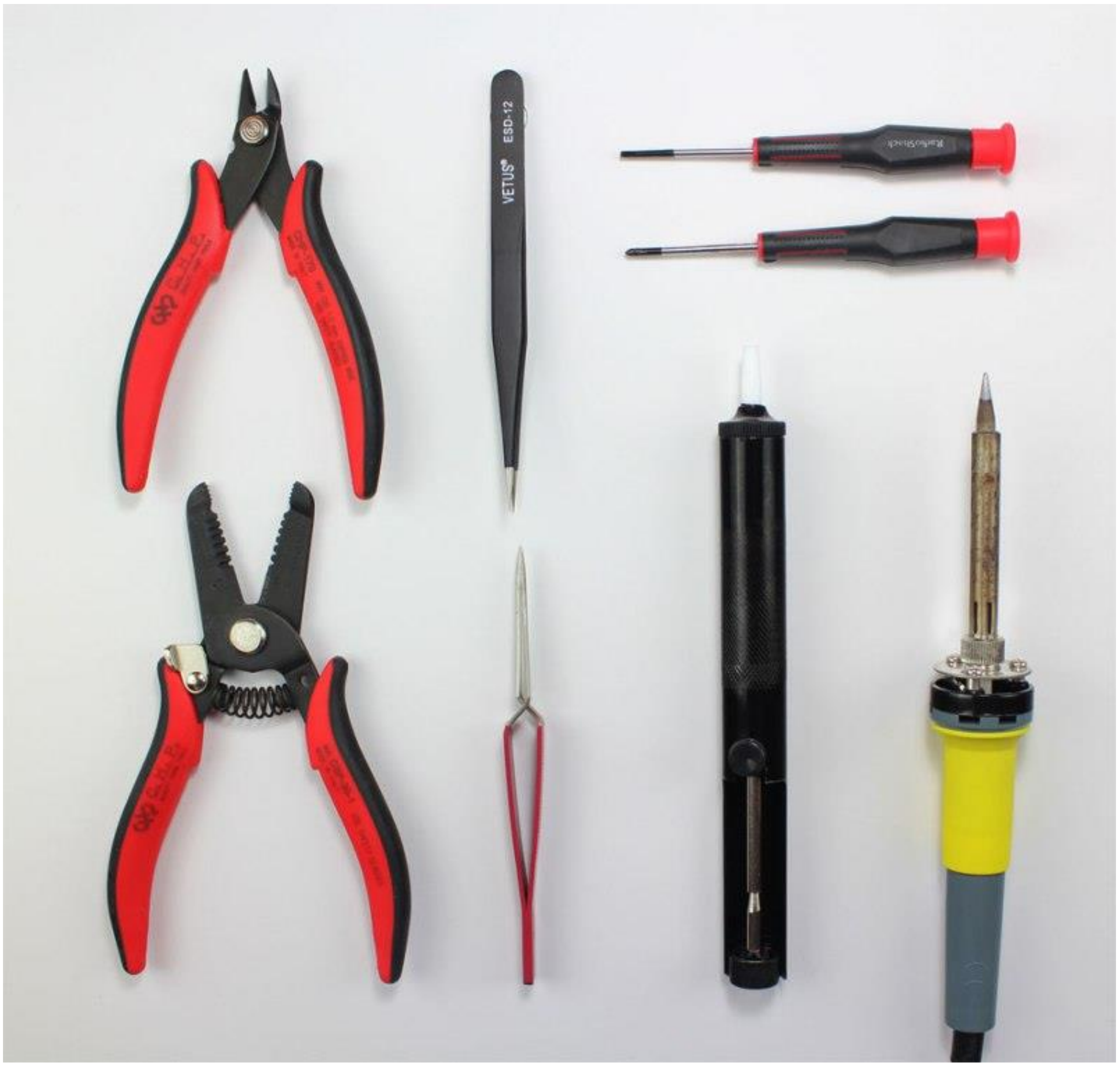
LED Cube w/ Arduino Uno – Instructables



Control A Doorlock Using Arduino and Bluetooth – MAKE Magazine

Tools Needed

Next we will help to highlight some of the most common tools you will need when working with Arduino projects.

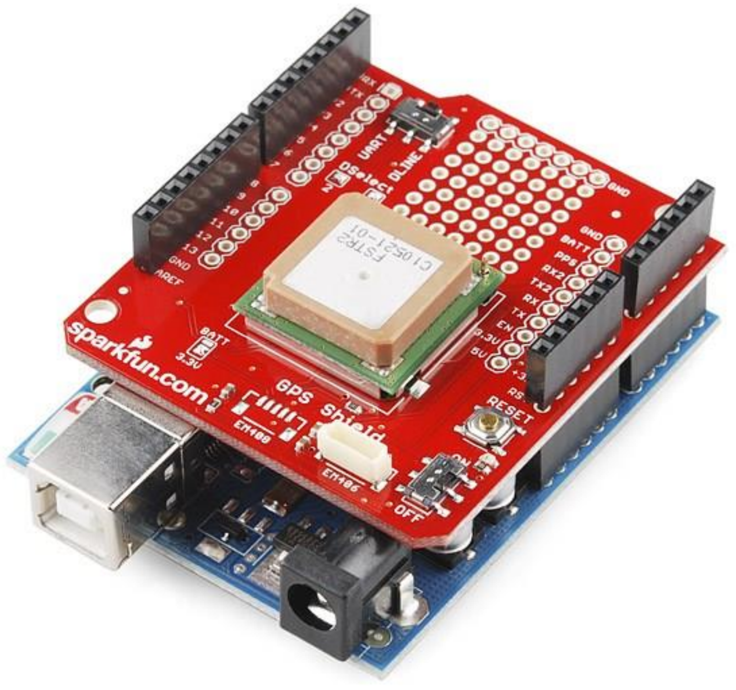


- [Needle-nose Pliers]
- [Wire Strippers]
- [Precision Screwdriver Set]
- [Flush Cutters]
- [Fine Tip Straight Tweezers]
- [Digital Multimeter]
- [Soldering Iron]
- [Panavise Jr]
- [Solder Sucker]

Arduino Shields

If you want to add a very specific functionality to your Arduino, you will need to use a shield. Arduino shields plug into the top of the Arduino board and can add capabilities such as WiFi, Bluetooth, GPS and much more. There are literally hundreds of shields to choose from and here are a few examples.

- WiFi Shield
- LCD Shield
- GPS Logger Shield
- MP3 Music Maker Shield
- Ethernet Shield
- Motor/Stepper/Servo Shield

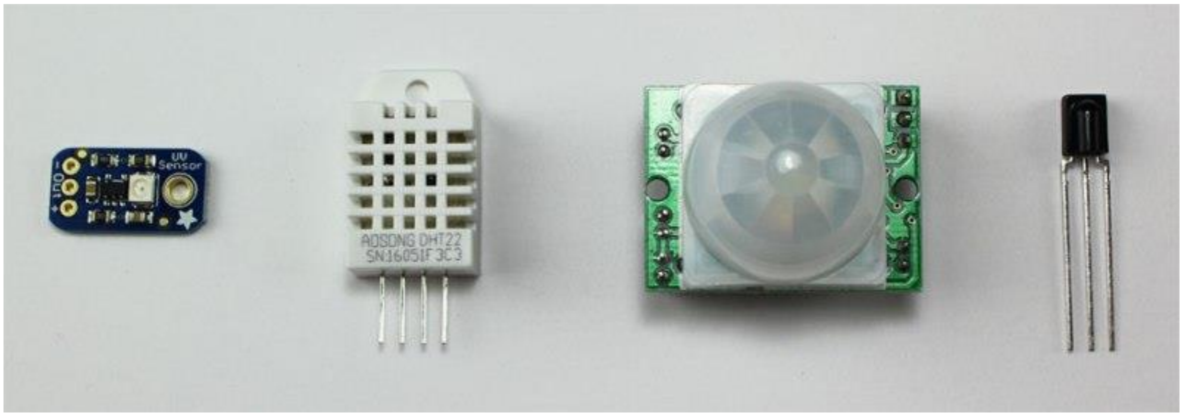


GPS Shield Plugged into Arduino Uno – Sparkfun.com

Arduino Sensors

If you want your Arduino to sense the world around it, you will need to add a sensor. There are a wide range of sensors to choose from and they each have a specific purpose. Below you will find some of the commonly used sensors in projects.

- Distance Ranging Sensor
- PIR Motion Sensor
- Light Sensor
- Degree of Flex Sensor
- Pressure Sensor
- Proximity Sensor
- Acceleration Sensor
- Sound Detecting Sensor
- RGB and Gesture Sensor
- Humidity and Temperature Sensor



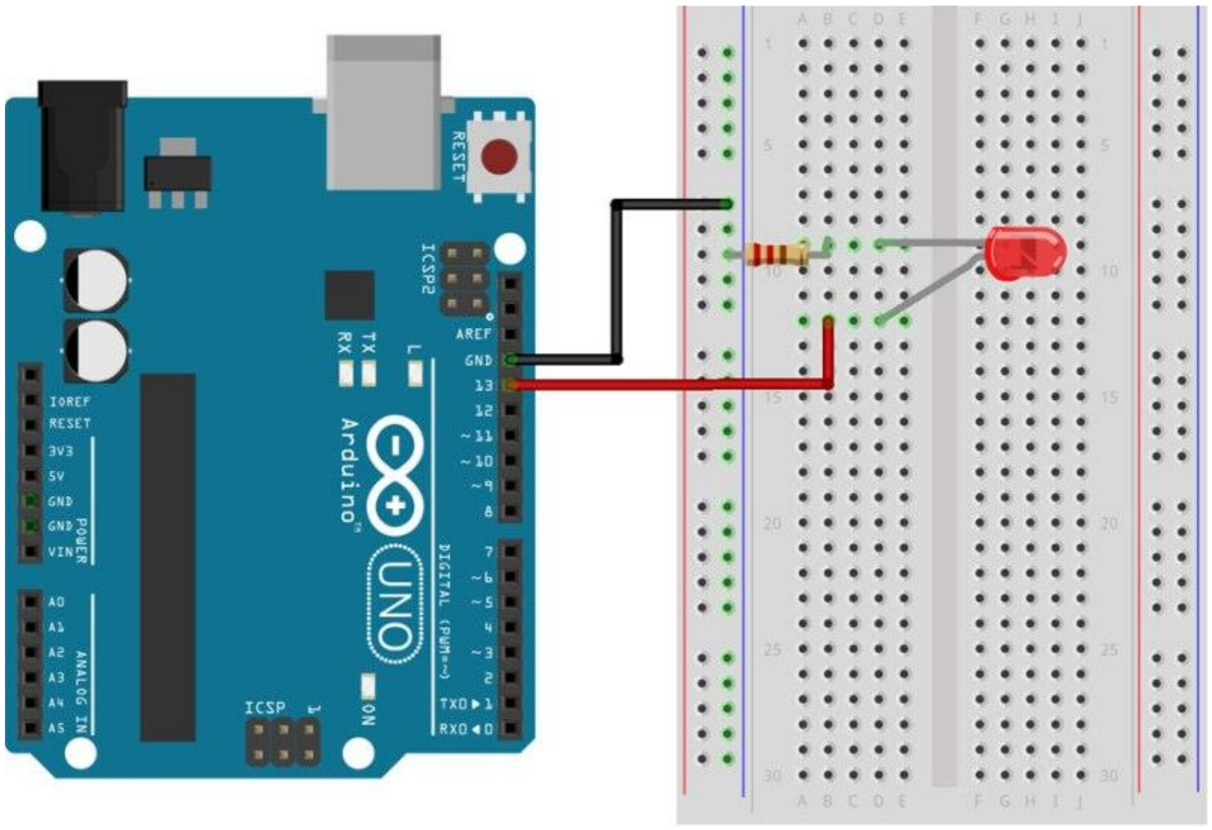
Examples of Arduino Sensors

Electronics Retailers

Below are some of our favorite places we tend to go when we need makerspace materials or electronic components.

- Adafruit
- Sparkfun
- Makershed
- Radio Shack
- Jameco
- Digikey
- MCM Electronics
- Mouser

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

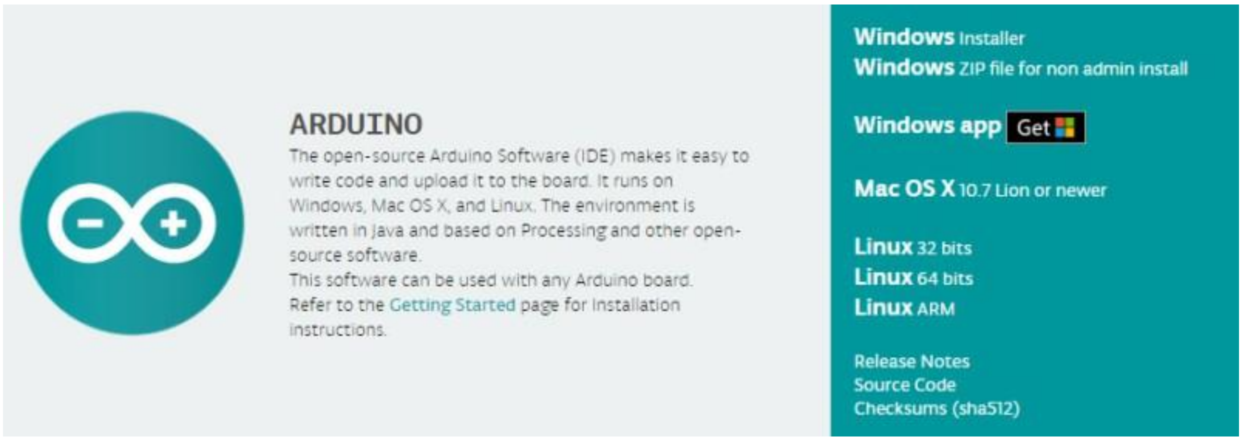
In order to complete the projects in this tutorial, you'll need to make sure you have the following items.

- 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

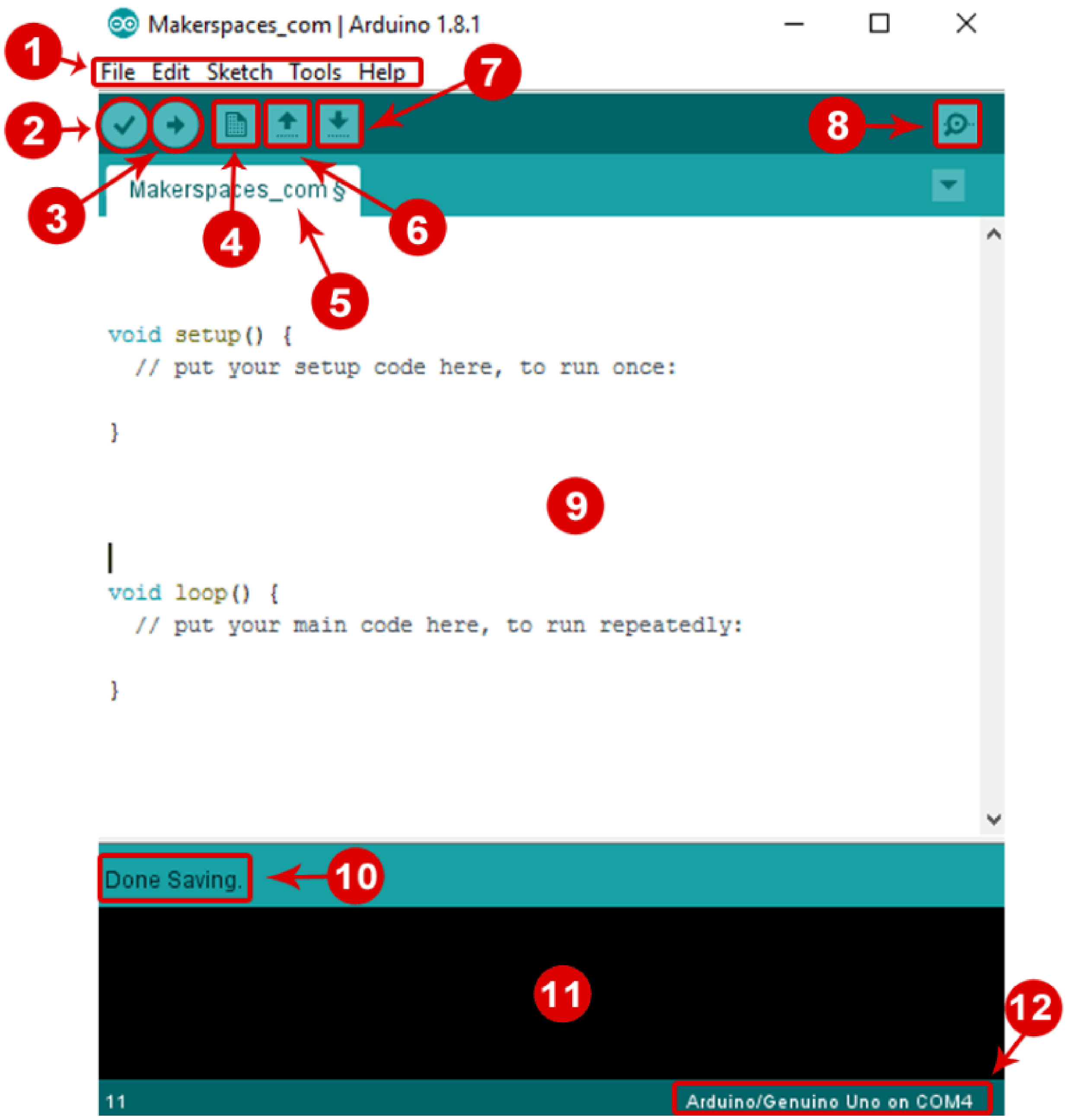
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

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.


Makerspaces_com | Arduino 1.8.1
File Edit Sketch Tools Help
Makerspaces_com §
Done Saving.
Arduino/Genuino Uno on COM4
1
2
3
4
5
6
7
8
9
10
11
12

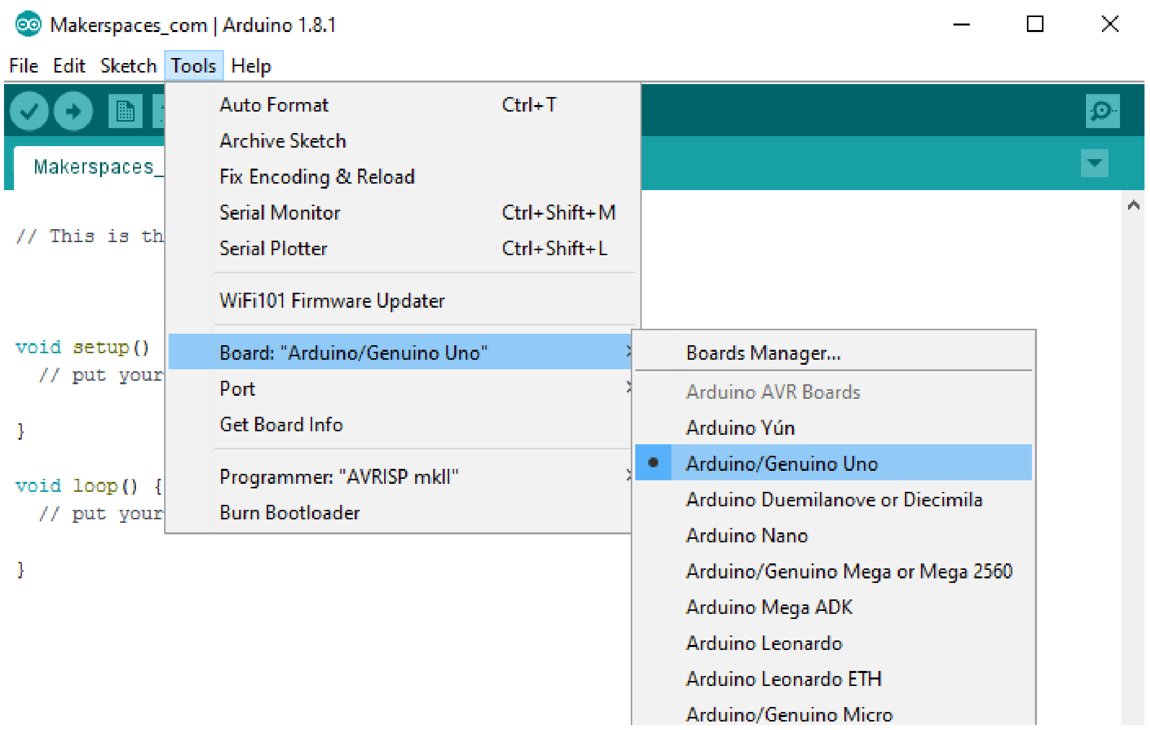
```
void setup() {
  // put your setup code here, to run once:

}


void loop() {
  // put your main code here, to run repeatedly:

}
```

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.

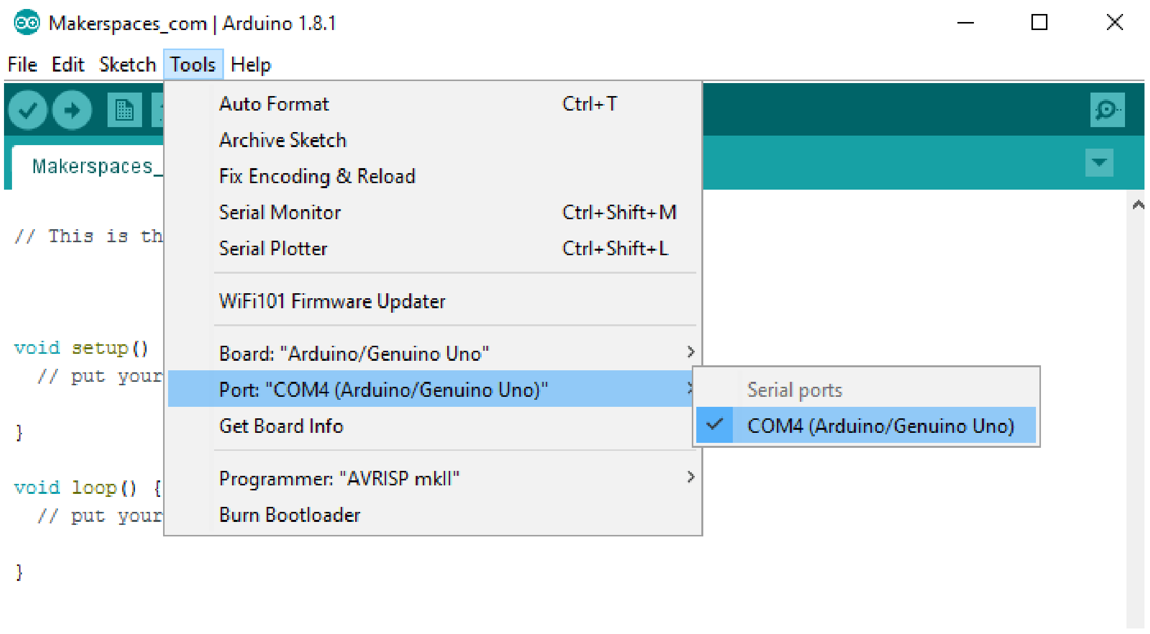
Once the board is connected, you will need to go to **Tools** then **Board** then finally select **Arduino Uno.**



```
// This is th


void setup()
  // put your

}

void loop() {
  // put your

}
```

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.**

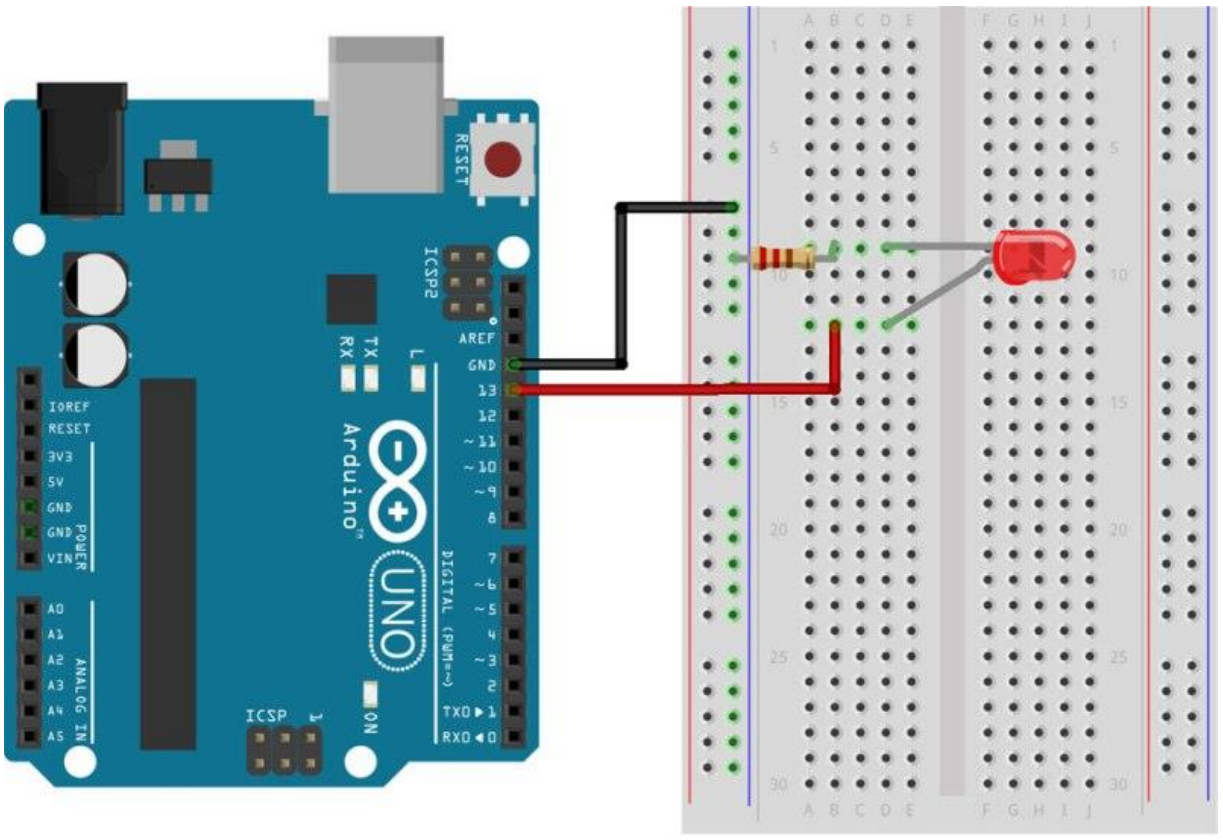

Makerspaces_com | Arduino 1.8.1
File
Edit
Sketch
Tools
Help
Makerspaces_
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM4 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM4 (Arduino/Genuino Uno)

```
// This is th


void setup()
  // put your

}

void loop() {
  // put your

}
```

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

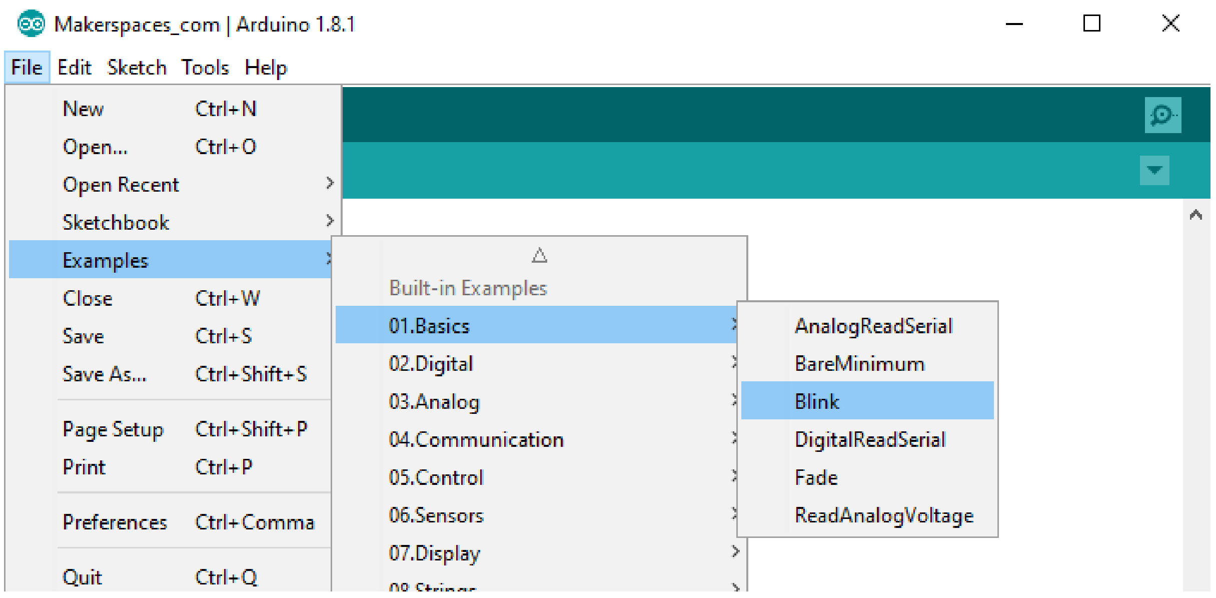
You can build your Arduino circuit by looking at the breadboard image above or by using the written description below. In the written description, we will use a letter/number combo that refers to the location of the component. If we mention H19 for example, that refers to column H, row 19 on the breadboard.

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

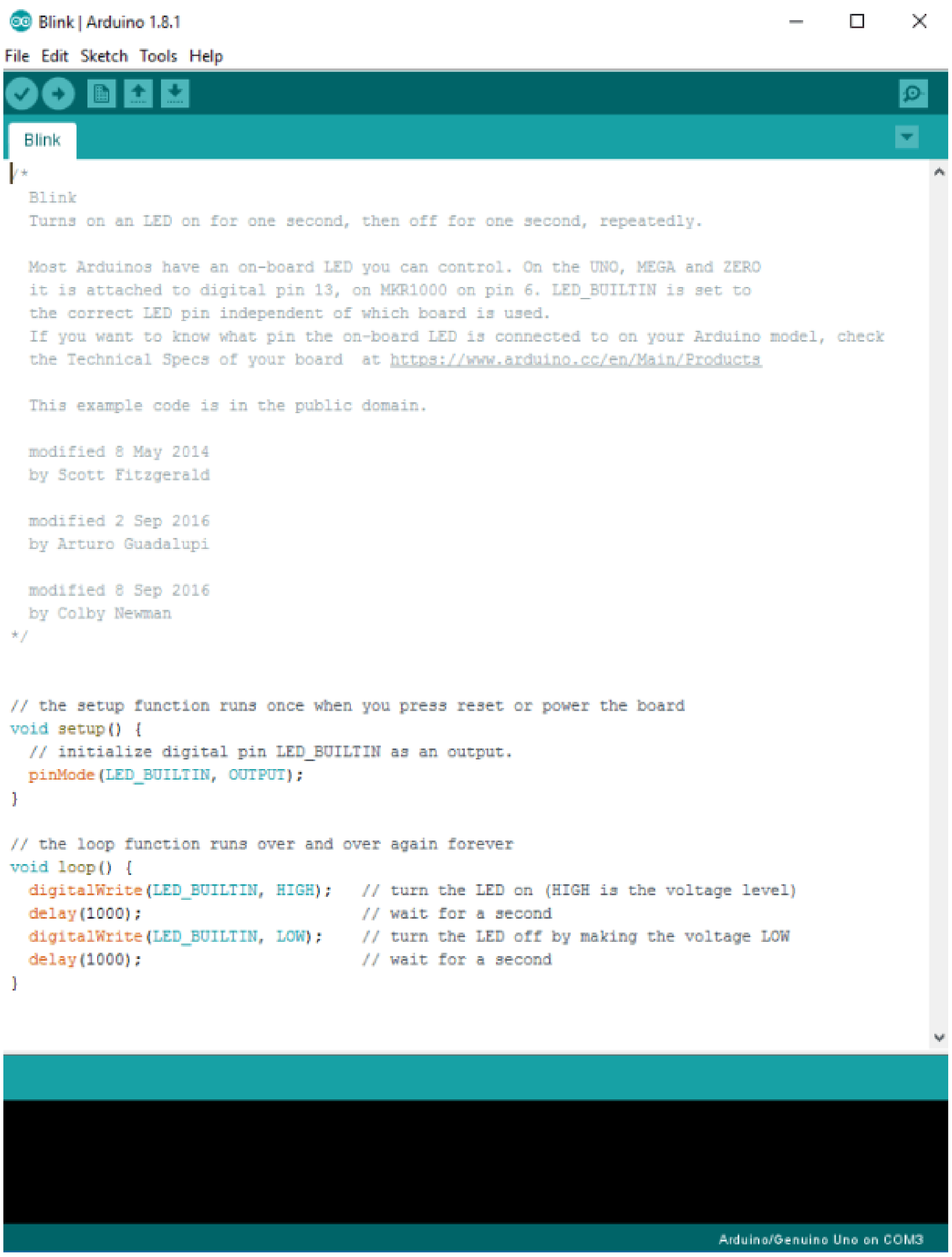
Upload The Blink Sketch

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**

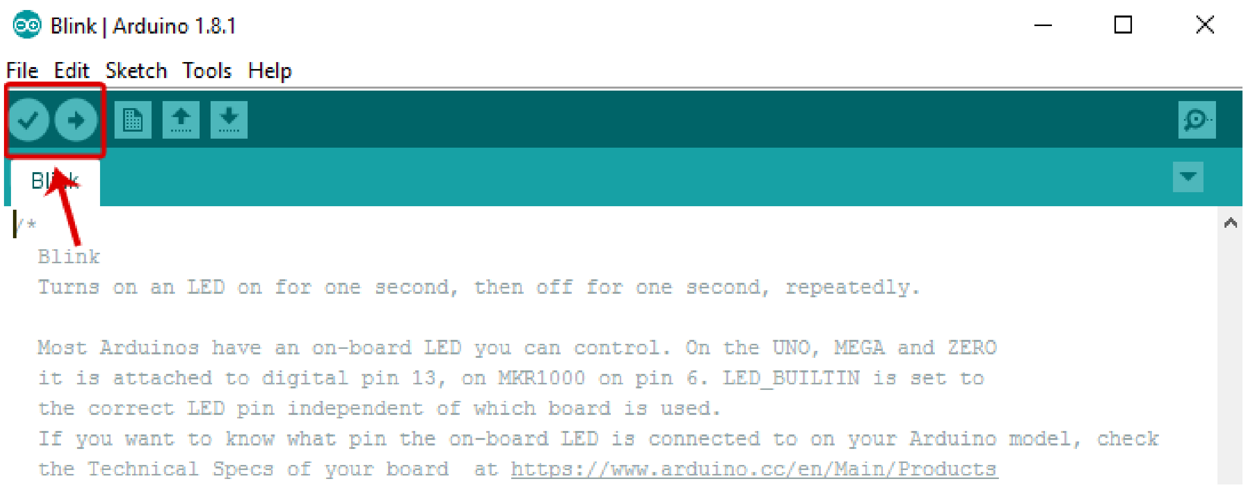


Now you should have a fully coded blink sketch that looks like the image below.



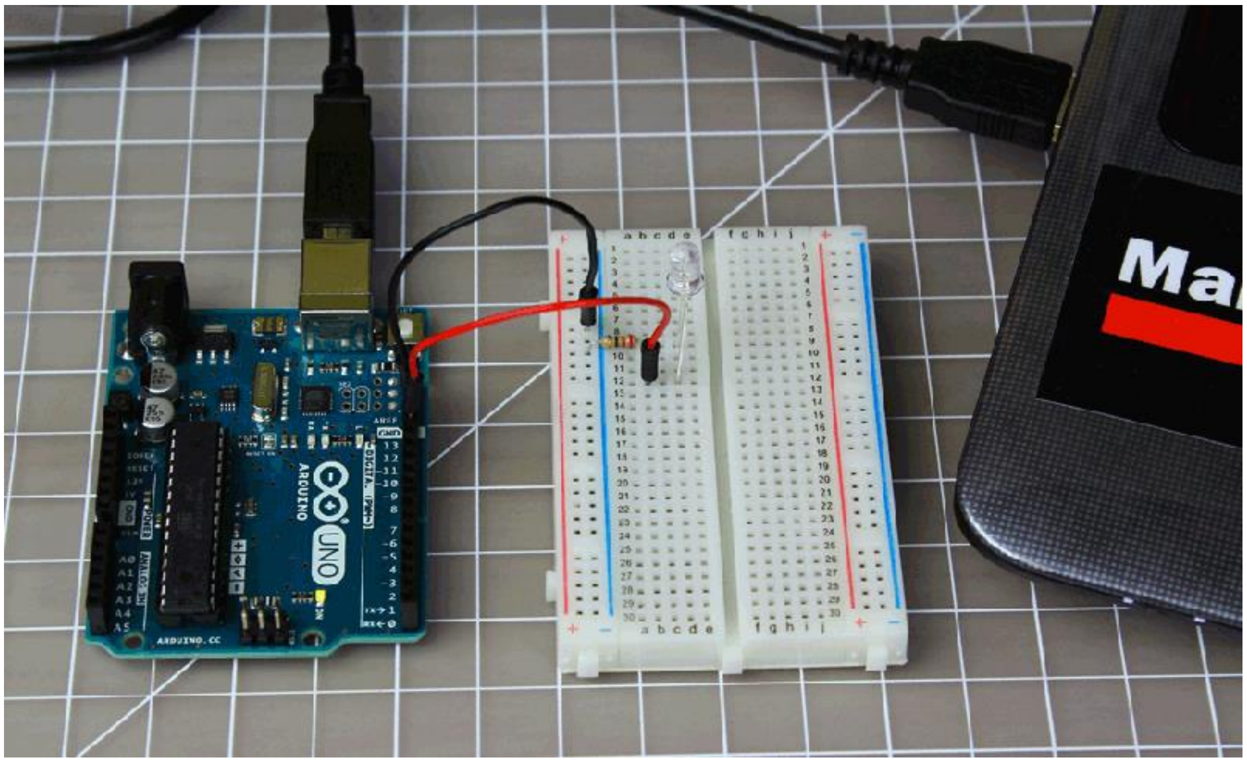
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  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
}
```

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.



```
/*
  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
```

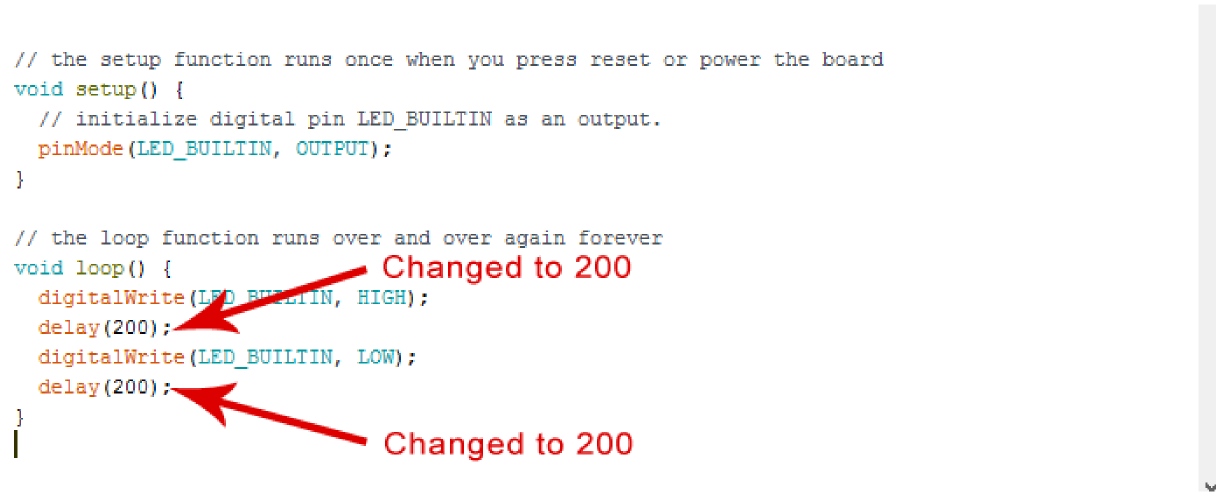
The built-in LEDs on the Arduino board will flash rapidly for a few seconds and then the program will execute. If everything went correctly, the LED on the breadboard should turn on for a second and then off for a second and continue in a loop.



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, lets 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.



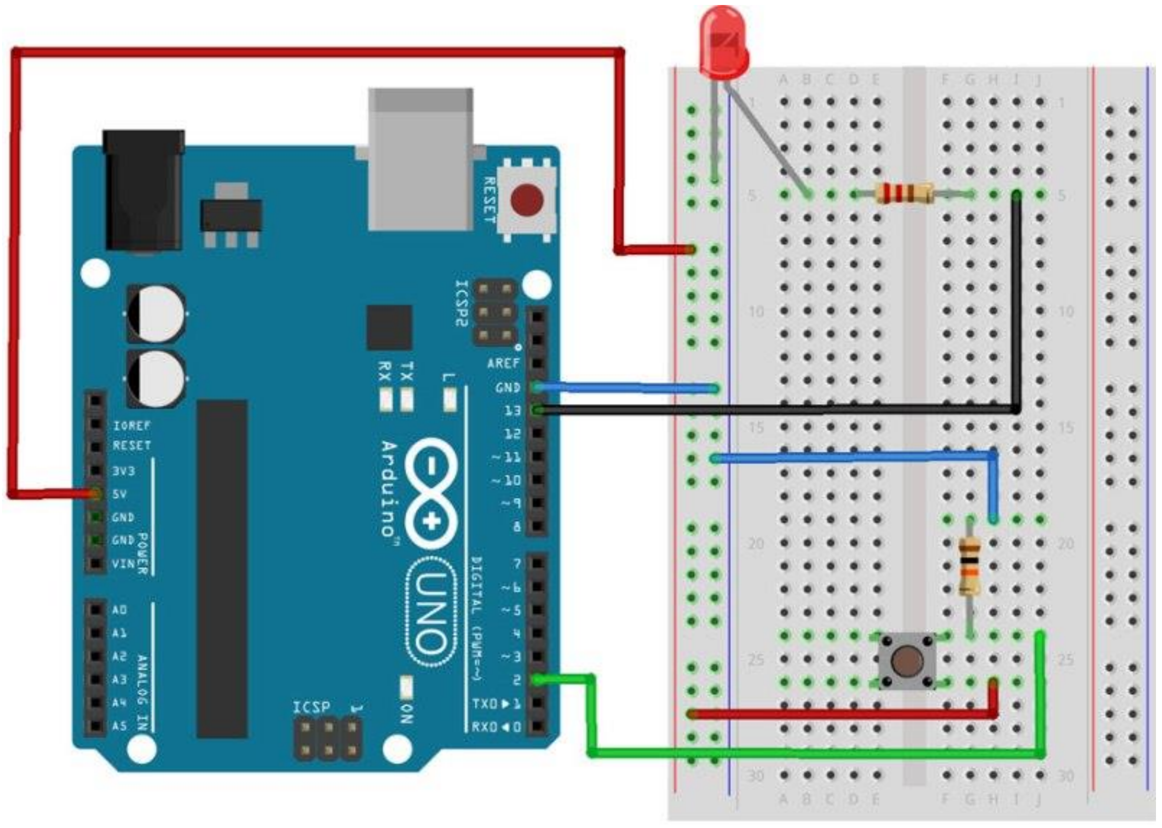
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// 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 a 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

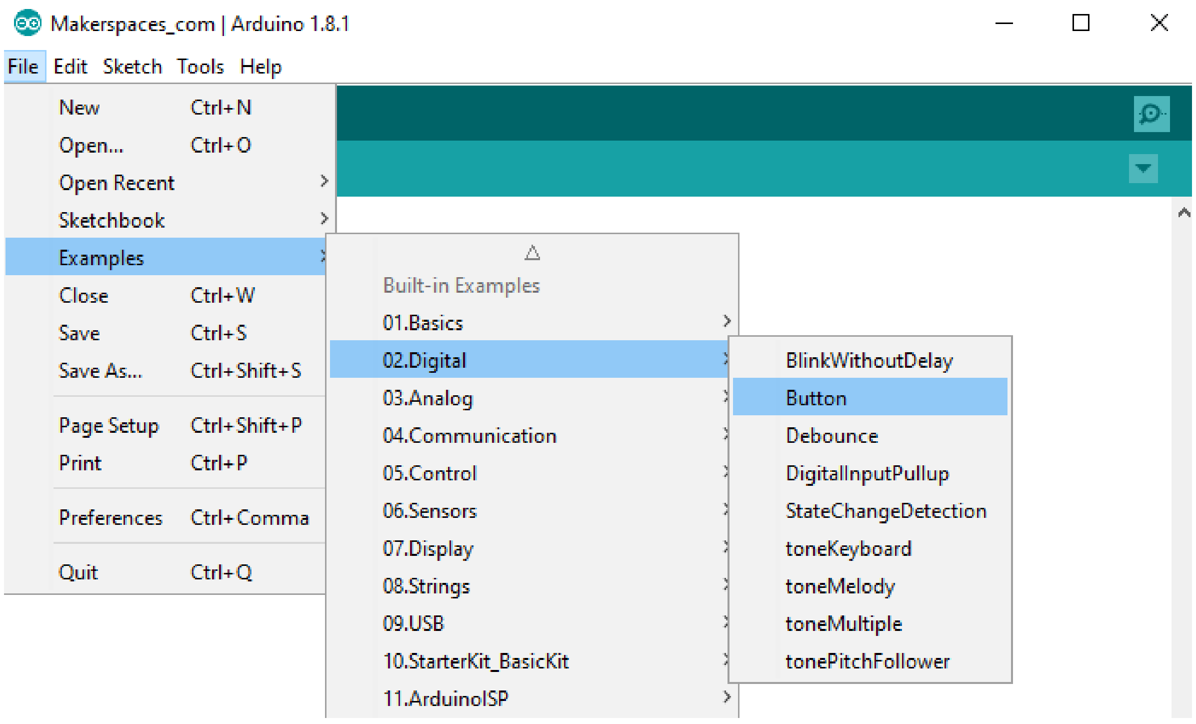
You can build your Arduino circuit by looking at the breadboard image above or by using the written description below. In the written description, we will use a letter/number combo that refers to the location of the component. If we mention H19 for example, that refers to column H, row 19 on the breadboard.

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

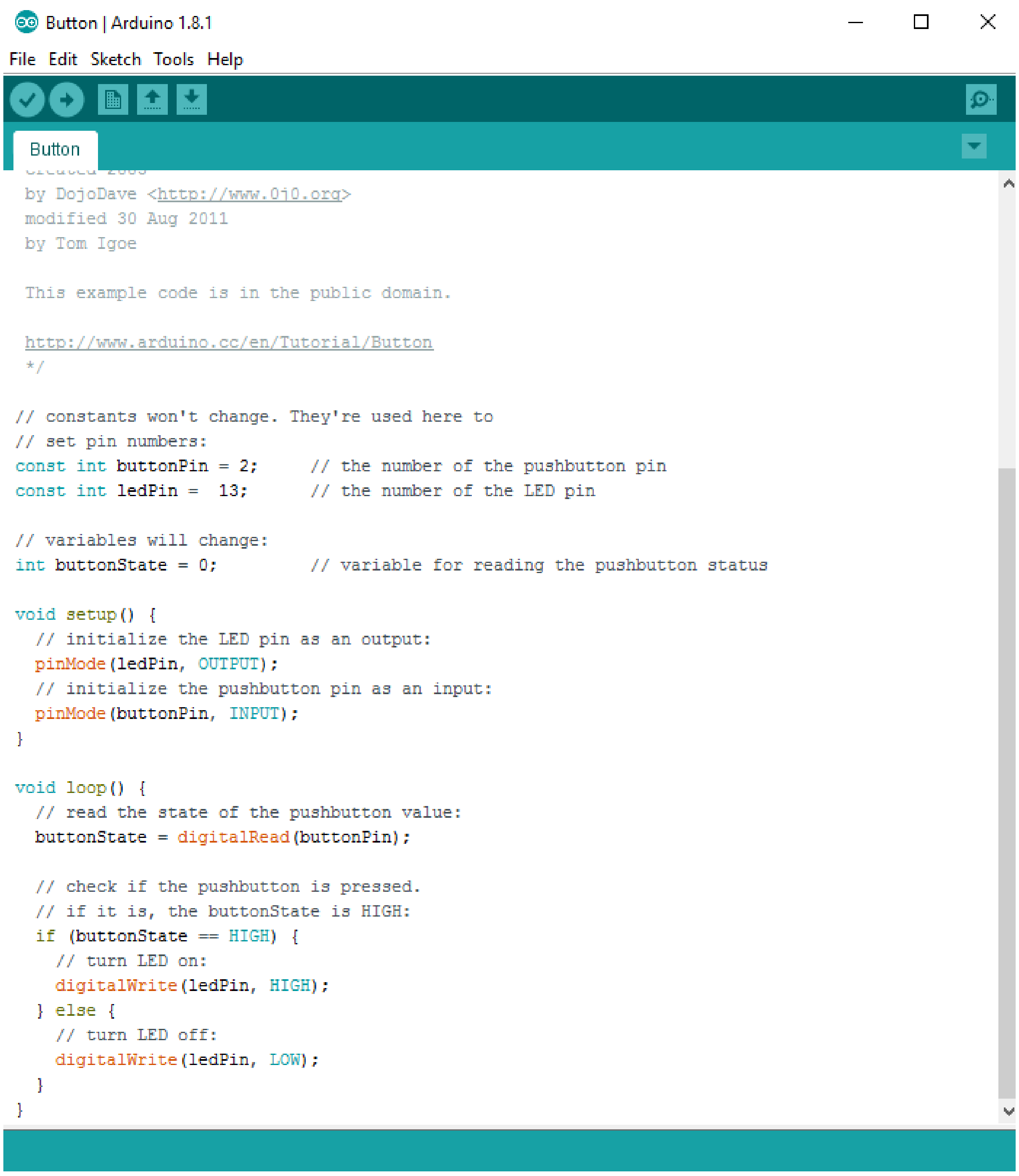
Upload The Switch Sketch

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**


Makerspaces_com | Arduino 1.8.1
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.



```
  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

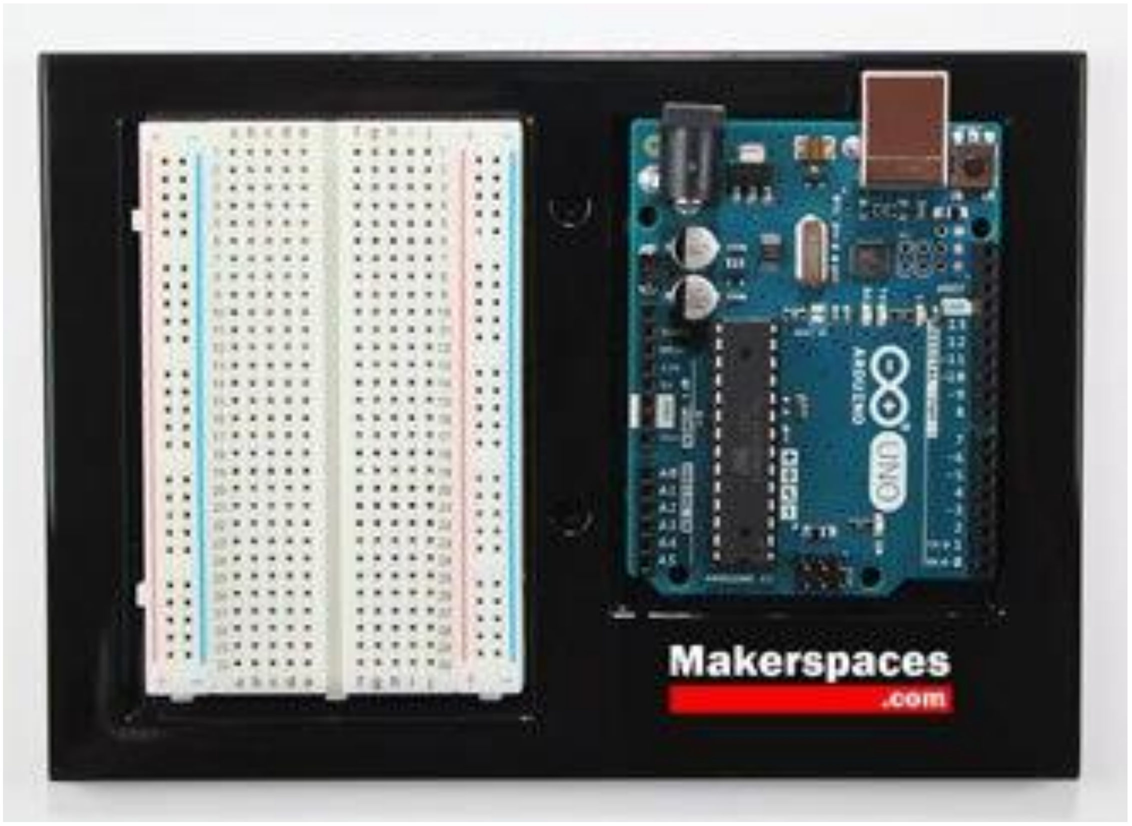
If you are having any problems with the projects we did, make sure the following has been checked.

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

Conclusion



This may be the end of this tutorial, but it’s just the beginning of your journey working with Arduino. Stay tuned for future posts and guides on Arduino.

Maker Spaces .com

Makerspaces.com