

Hello Makers!

How are the recent innovations coming along? We're back with more useful content. We have been getting a lot of enquiries from our community about getting started with the Arduino IDE. Here's a tutorial that helps you get started with arduino programming.

Haven't been introduced to arduino boards yet? You should probably check out this article that gives you a great insight into arduino basics.

Introduction and Installation of the Arduino Integrated Development Environment

IDE stands for Integrated Development Environment. It is a text editor that lets you upload code on to arduino. Every program file is called a sketch and contains all the code that you write for your projects. Every file has an extension of .ino which used to be a .pde!

The code is written in basic c++ format and is human readable. So if the code is written in the readable form, how does the machine understand it? Well, that is precisely what the IDE is responsible for. The internal process of compiling translates the code that you write to a format that the machine understands.

How you can install Arduino Integrated Development Environment:

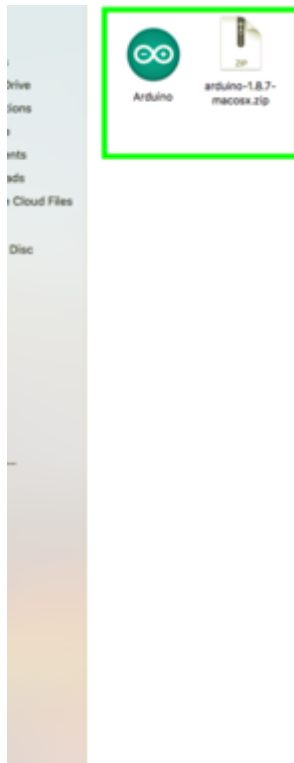
- Head over to arduino.cc
- Go to the **Software** tab and click on **Download**



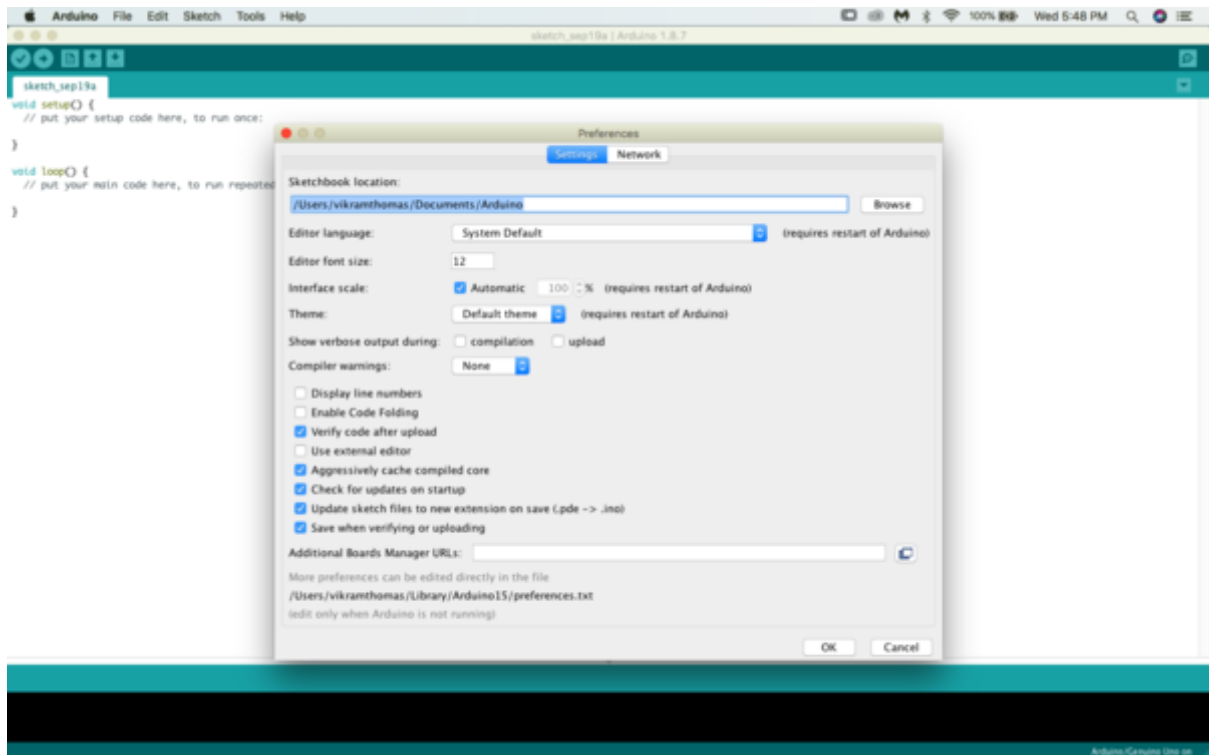
- Scroll down and click on the download option for your operating system



- Unzip the downloaded compressed zip file, and voila, you have your arduino IDE



Once you install the IDE the first thing to do is set up your preferences. You can do that by heading over to the **preferences** section under the **Arduino** tab. Over here you can control various different settings like where you want your sketches to be saved, font size, and a lot more. Go crazy!



Anatomy of the Arduino Development Environment

Can't wait to create your first sketch? The first step is to name your sketch. Go ahead and give it a human-friendly name. The name that you give to your sketch appears on the sketch tab and all the way at the top. Every sketch is given a default name, you can choose to override this name by choosing to save the file, which lets you rename the file.

Pro Tip - Keep saving your files so that you have instant backups in case of an emergency.

Verify Button

The first button is the verify button. The verify button is used for compiling your code and checking for errors. It highlights all the errors that you have in the sketch. If there are no errors there won't be any highlights and you are good to go. The shortcut key for the verify button is 'cntrl + r'.

Upload Button

The second button is the upload button and it is used for uploading your code to the arduino board. The shortcut key for this button is 'cntrl + u'. When you use this button you will normally see two LED's light up on your board, the TX and RX. These LED's light up when there is information being passed between the board and the IDE.

New Editor Button

The new editor button opens up a new code editing window that you can use instead of the current one. You can use the shortcut key 'cntrl + n'.

Open-File Button

The open file button is used for opening a pre-existing file. You can use the shortcut key 'cntrl + o' for opening a file.

Save Button

The save button can be used for saving your sketch. You can use the shortcut key 'cntrl + s'.

Serial Monitor

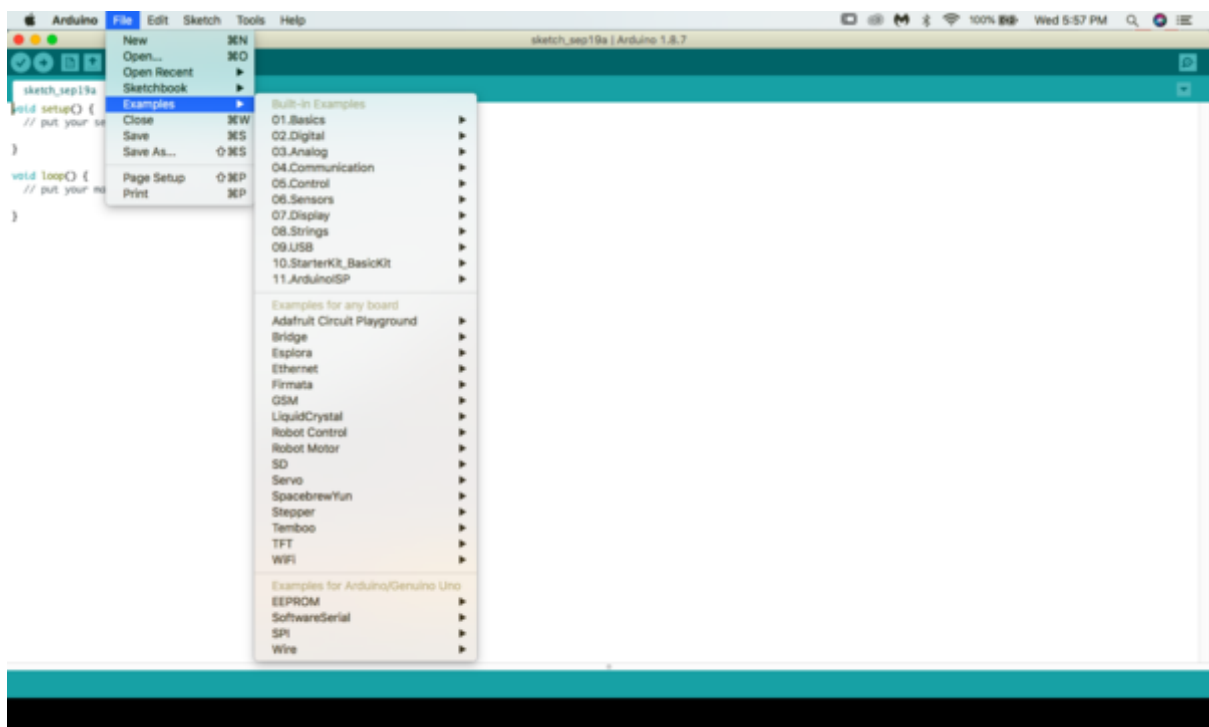
The Serial Monitor window is used for checking the level of communication between your arduino board and the IDE. The shortcut key for your serial monitor window is 'cntrl + shift + m'.

The Down Arrow Button

Down arrow gives you options like adding a sketch to the current project. It opens as a new tab in the current code editor, that is useful for organizing your code into logical files.

All the way at the bottom you can find out information about the type of board that is connected with the development environment.

Pro Tip - Have a look at the **examples** section under **files** to find pre-existing code that you can play around with. If you are a beginner this is a great way to get familiar with Arduino coding.



Arduino IDE Syntax Rules

There are basic syntax rules that you would have to follow while writing code on the arduino IDE.

Let's start with comments. Comments are statements that are written to help the reader understand the code. They aren't statements of code themselves, rather they are used to make the reader understand what the code is all about. There are two different types of comments, single line comments, and multi-line comments.

Single line comments are written using `//` and do not need a termination. Multiline comments start with `/*` and end with `*/`

```
/*
LED bar graph

Turns on a series of LEDs based on the value of an analog sensor.
This is a simple way to make a bar graph display. Though this graph uses 10
LEDs, you can use any number by changing the LED count and the pins in the
array.

This method can be used to control any series of digital outputs that depends
on an analog input.

The circuit:
- LEDs from pins 2 through 11 to ground

created 4 Sep 2010
by Tom Igoe

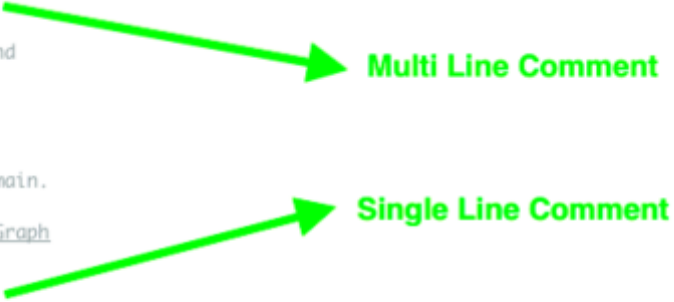
This example code is in the public domain.

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

// these constants won't change:
const int analogPin = A0; // the pin that the potentiometer is attached to
const int ledCount = 10;  // the number of LEDs in the bar graph

int ledPins[] = {
  2, 3, 4, 5, 6, 7, 8, 9, 10, 11
}; // an array of pin numbers to which LEDs are attached

void setup() {
  // loop over the pin array and set them all to output:
  for (int thisLed = 0; thisLed < ledCount; thisLed++) {
    pinMode(ledPins[thisLed], OUTPUT);
  }
}
```



The diagram illustrates the two types of comments. A green arrow points from the multi-line comment block (enclosed in `/*` and `*/`) to the label **Multi Line Comment**. Another green arrow points from a single-line comment (starting with `//`) to the label **Single Line Comment**.

The next important syntactic rule is the usage of the semicolon (;) after every statement. The IDE will throw an error if a semicolon is not used at the end of a statement.

You will be using the setup and loop functions in every programming file. The setup function is used to define the setup parameters for the arduino board. The code in the loop function is executed by the arduino board, yes you guessed it, in a loop.

Here's a visual example of how you can use these two functions

```
http://www.arduino.cc/en/Tutorial/AnalogReadSerial
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

Introduction to Variables in Arduino Programming

Variables can be used to carry information from one part of the program to another part. Think of a variable as a cup. You can fill the cup with anything you want. The cup that you have remains as it is, although the contents of the cup can change.

Now let's have a look at variable types. A cup of a certain type can only be used for holding a certain item similarly a variable of a particular type, let's say integer, can only be used for storing integer values. You can read up on the different types of variables over here.

Arduino data types and constants.


Constants	String	Variable Scope & Qualifiers
Floating Point Constants	String()	const
Integer Constants	array	scope
HIGH LOW	bool	static
INPUT OUTPUT INPUT_PULLUP	boolean	volatile
LED_BUILTIN	byte	
true false	char	Utilities
	double	PROGMEM
	float	sizeof()
	int	
Conversion	long	
byte()	short	
char()	unsigned char	
float()	unsigned int	
int()	unsigned long	
long()	void	
word()	word	
Data Types		

A variable comes into existence through variable declaration. Every variable declaration has a data type and a variable name.

Variable declaration rules:

- A variable name cannot have a space or special characters, except for underscore(_).
- A variable name can be alphanumeric in nature
- A variable name cannot start with a number. It would have to start with an alphabet character or ''
- A variable name can contain uppercase and lowercase characters
- You cannot use a keyword (example 'int', 'void' and so on) as a variable name

You can give a variable value through variable initialization. It can be done at the time of variable declaration or at a later point.



```
sketch_sep19a  
  
//Initialization at the time of declaration  
int sample = 10;  
  
//Initialization at a later point  
int sample;  
sample = 10;
```

Now that you understand the basics of Arduino IDE, you can ahead and check out arduino programming tutorials for easy-to-implement projects.