Arduino Tutorial

About Arduino

Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its products are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form or as do-it-yourself (DIY) kits.

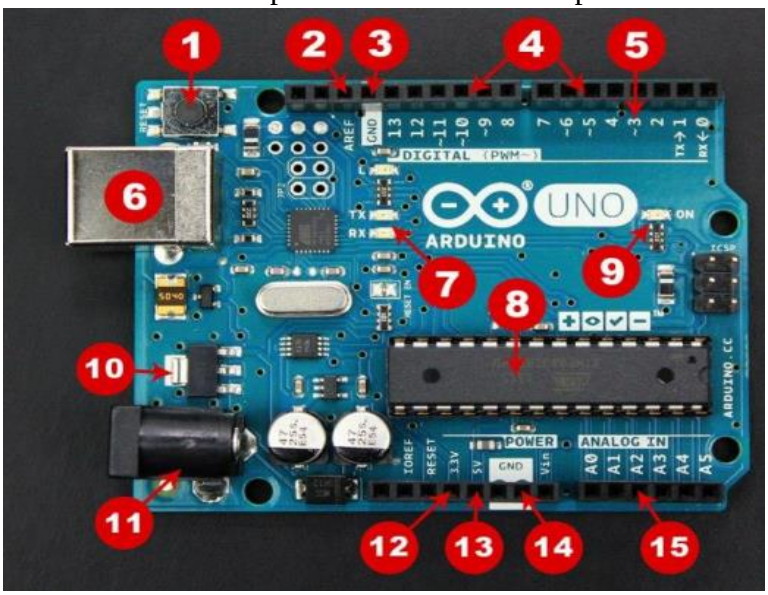
Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (For prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers can be programmed using C and C++ programming languages. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

Prerequisites

Before you start proceeding with this tutorial, we assume that you are already familiar with the basics of C and C++. If you are not well aware of these concepts, then we will suggest you go through our short tutorials on C and C++. A basic understanding of microcontrollers and electronics is also expected.

Arduino Overview

Here are the components that make up an Arduino board and what each of them



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.

Different type of Boards and their details

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Uno R3	5V	16MHz	14	6	6	1	USB via ATMega16U2
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB via ATMega16U2
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro 3.3v/8 MHz	3.3V	8 MHz	14	6	6	1	FTDICompatible Header
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDICompatible Header
Arduino mini 05	5V	16MHz	14	8	6	1	FTDICompatible Header
Arduino Pro mini 3.3v/8mhz	3.3V	8MHz	14	8	6	1	FTDICompatible Header
Arduino Pro mini 5v/16mhz	5V	16MHz	14	8	6	1	FTDICompatible Header
Arduino Ethernet	5V	16MHz	14	6	6	1	FTDICompatible Header
Arduino Fio	3.3V	8MHz	14	8	6	1	FTDICompatible Header
LilyPad Arduino 328 main board	3.3V	8MHz	14	6	6	1	FTDICompatible Header
LilyPad Arduino simply board	3.3V	8MHz	9	4	5	0	FTDICompatible Header

Arduino IDE – Installation

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.
In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1: First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

Step 2: Download Arduino IDE Software
You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system

(Windows, IOS, or Linux). After your file download is complete, unzip the file / install it according your downloaded file.

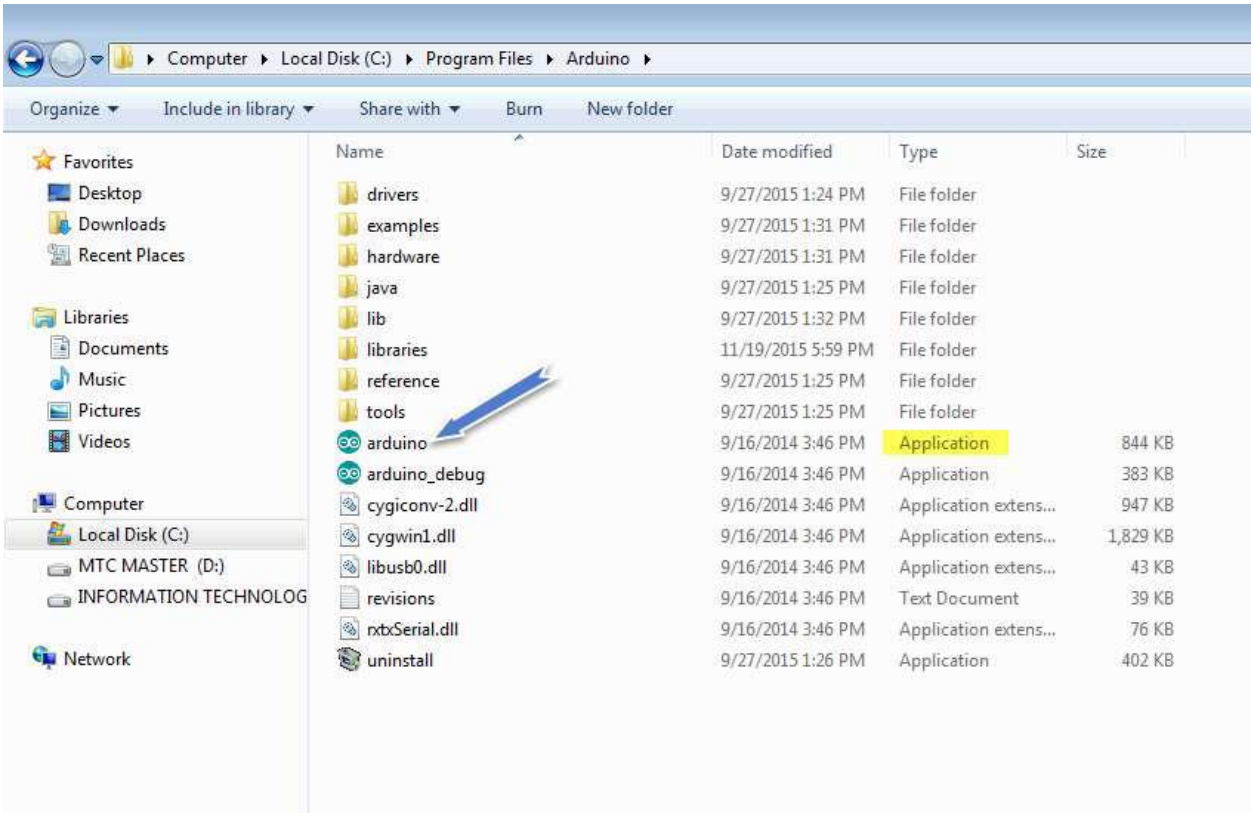
Step 3: Power up your board

The Arduino Uno, Mega and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4: Launch Arduino IDE

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double click the icon to start the IDE.

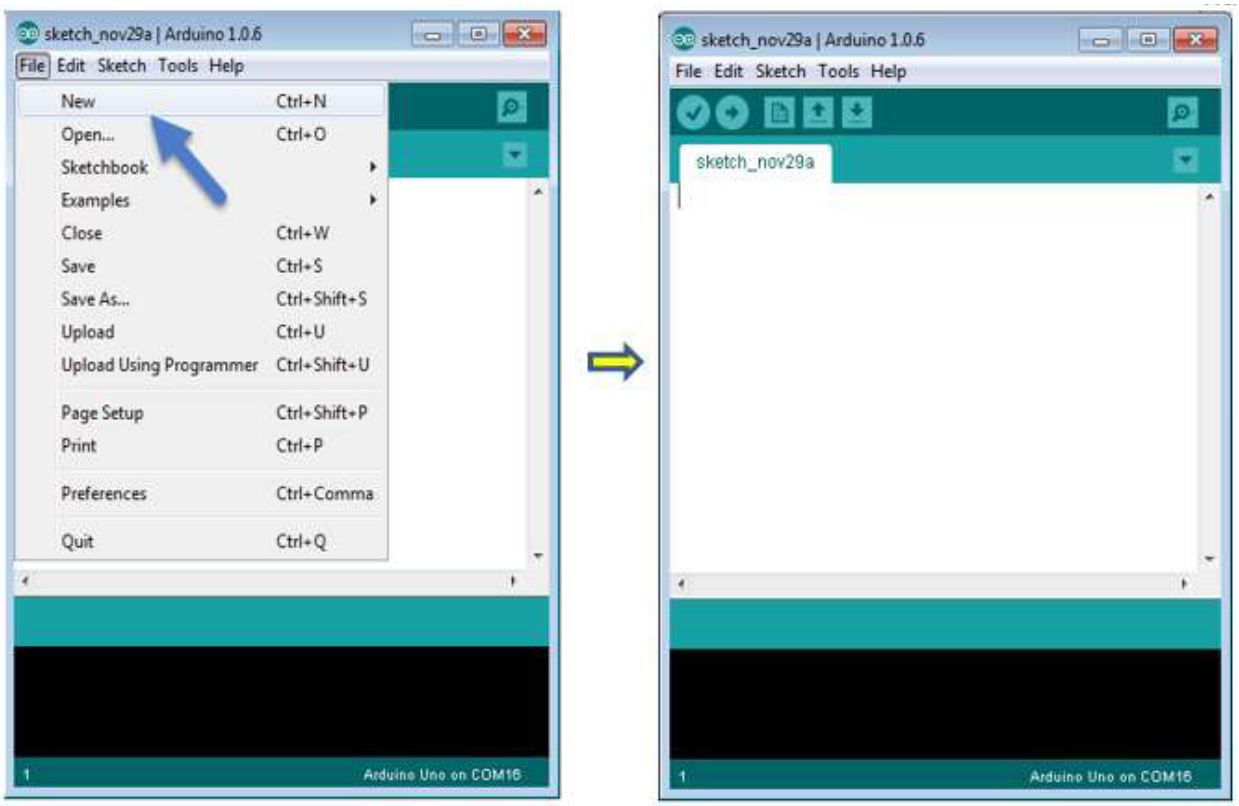


Step 5: Open your first project

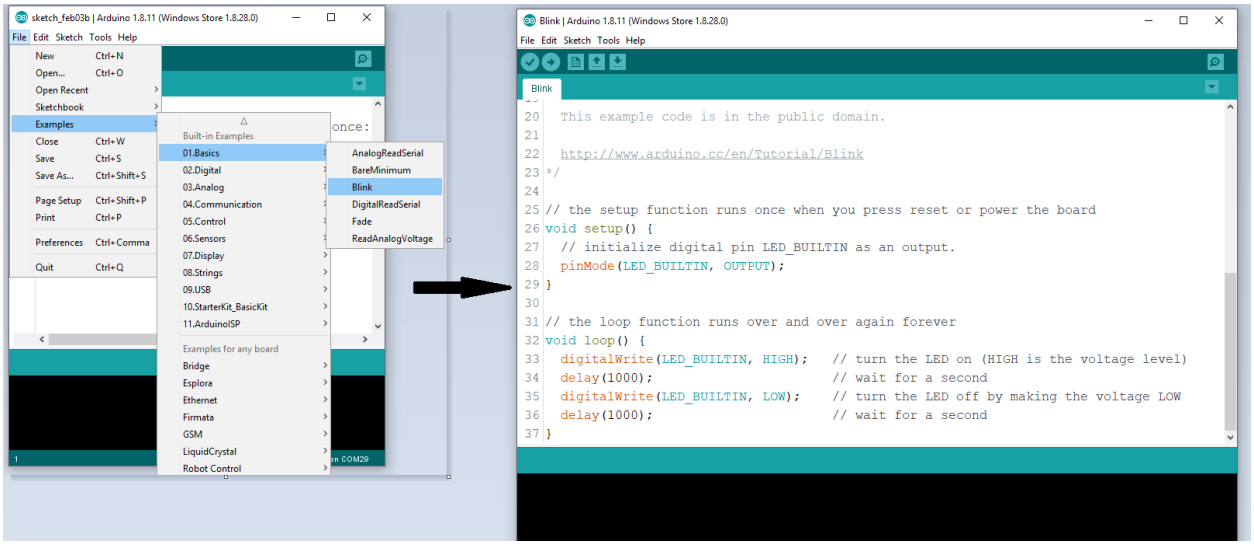
Once the software starts, you have two options:

- Create a new project.
- Open an existing project example.

To create a new project, select **File --> New.**



To open an existing project example, select **File -> Example -> Basics -> Blink.**

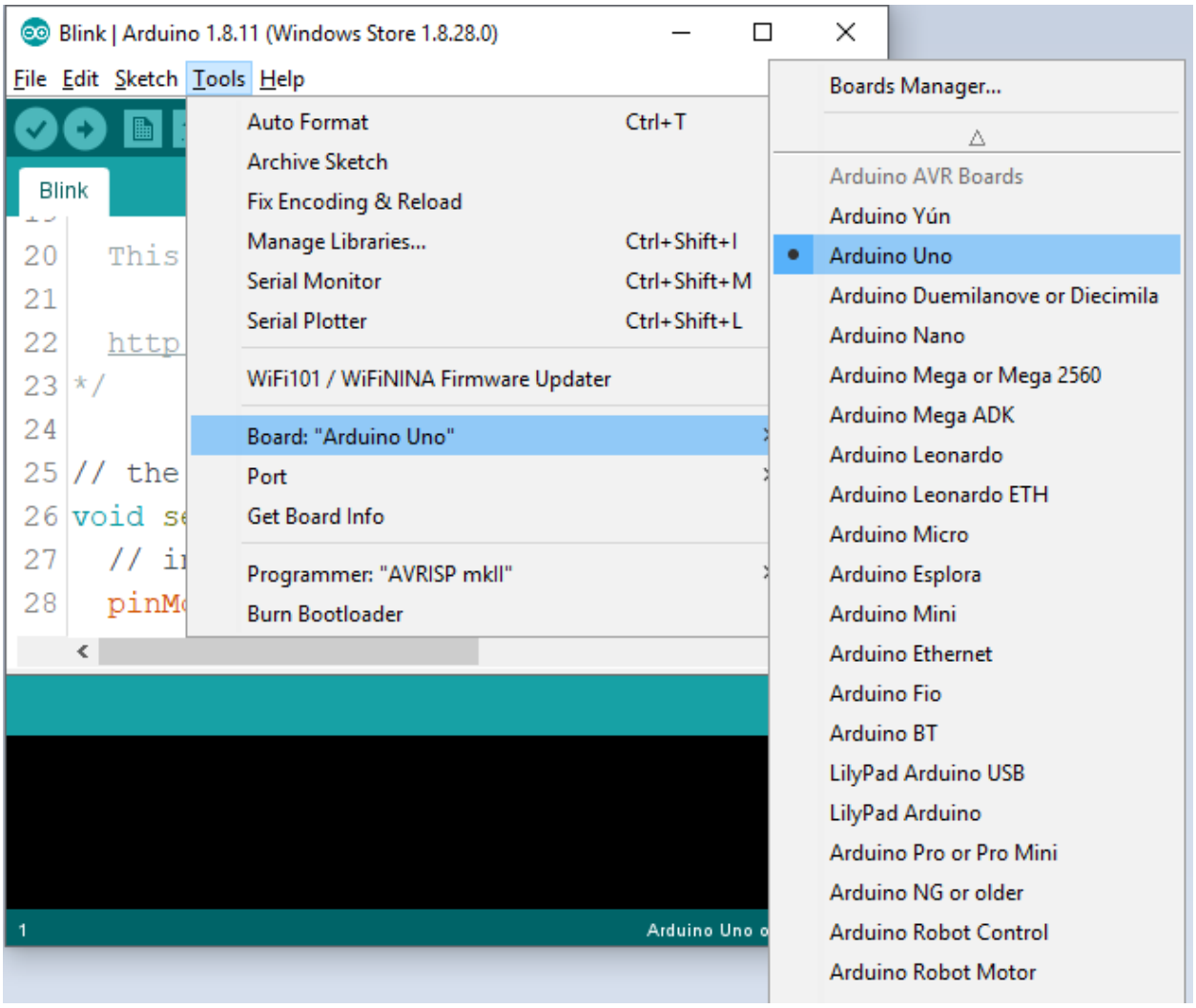


Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6: Select your Arduino board

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

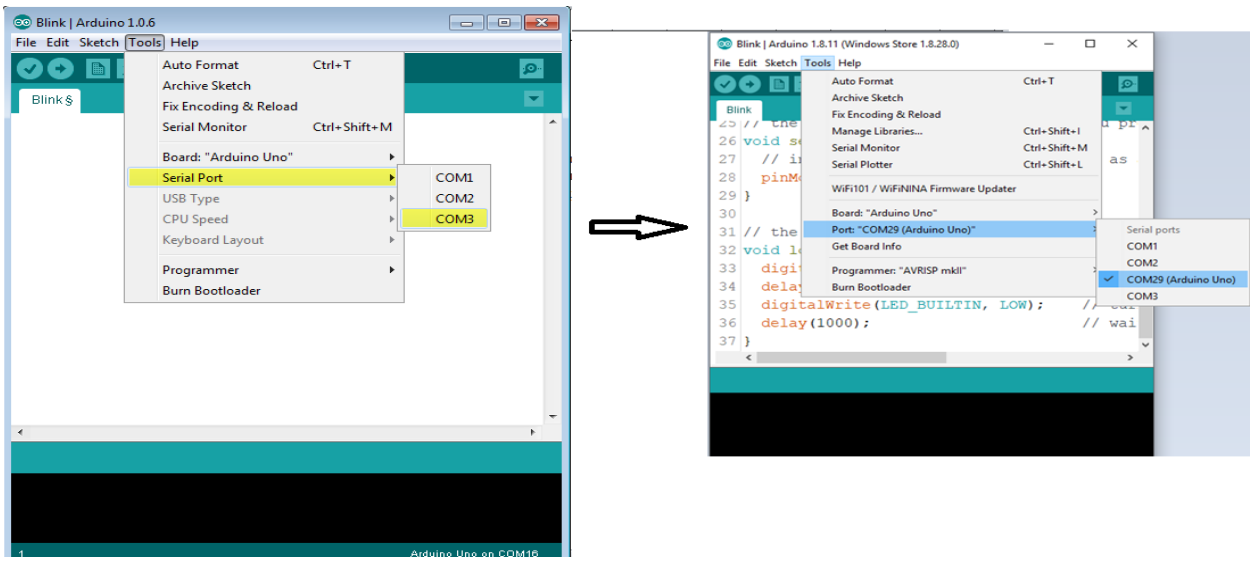
Go to **Tools -> Board** and select your board.



Here, we have selected **Arduino Uno** board according to our tutorial, but you must select the name matching the board that you are using.

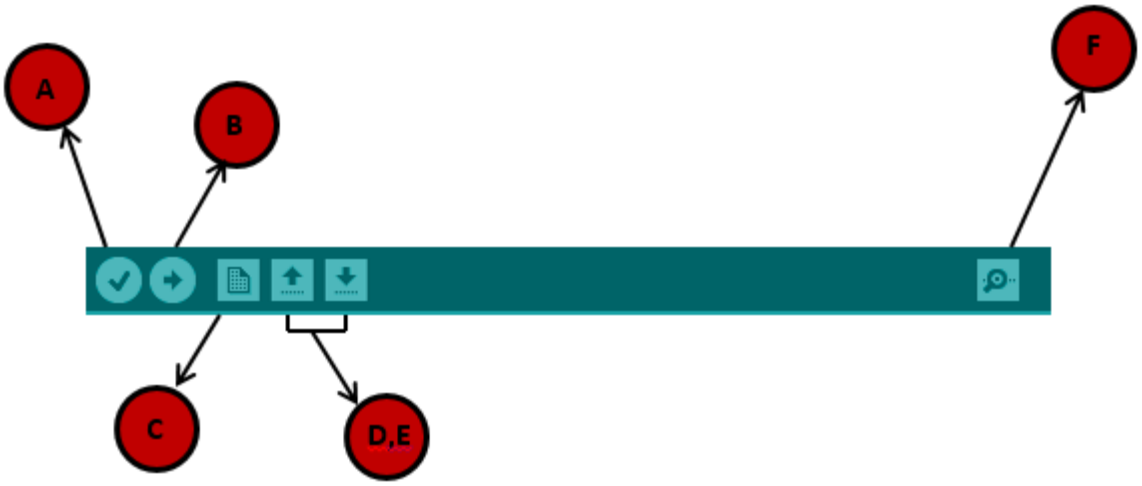
Step 7: Select your serial port

Select the serial device of the Arduino board. Go to **Tools -> Serial Port** menu. This is likely to be **COM3** or higher (**COM1** and **COM2** are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8: Upload the program to your board

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



A- Used to check if there is any compilation error.
B- Used to upload a program to the Arduino board.
C- Shortcut used to create a new sketch.
D- Used to directly open one of the example sketch.
E- Used to save your sketch.
F- Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "**Upload**" button in the environment. Wait a few seconds; you will see the **RX** and **TX** LEDs on the board, flashing. If the upload is successful, the message "**Done uploading**" will appear in the status bar.

[Note: If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.**]**

Arduino – Program Structure

In this chapter, we will study in depth, the Arduino program structure and we will learn more new terminologies used in the Arduino world. The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

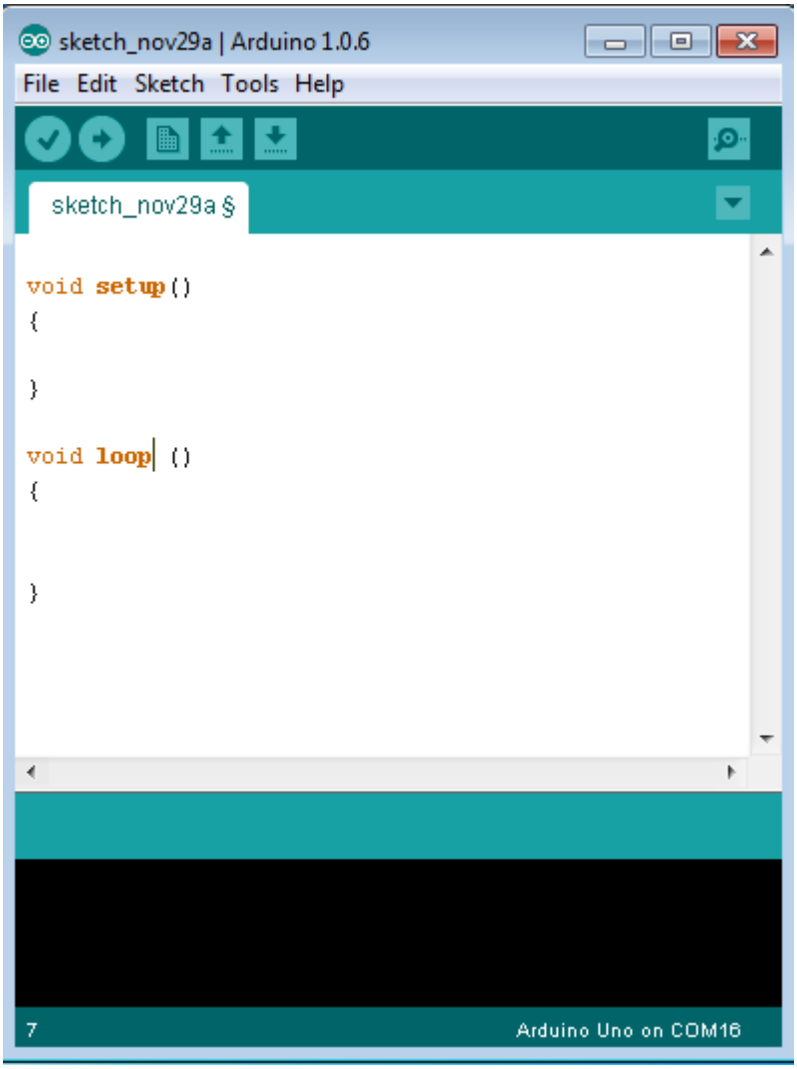
Sketch: The first new terminology is the Arduino program called "**sketch**".

Structure

Arduino programs can be divided in three main parts: **Structure**, **Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the Structure. Software structure consist of two main functions:

- ✓ Setup( ) function
- ✓ Loop( ) function




```
Void setup ( )
{
}
```

PURPOSE: The **setup()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

```
Void Loop ( )
{
}
```

PURPOSE: After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

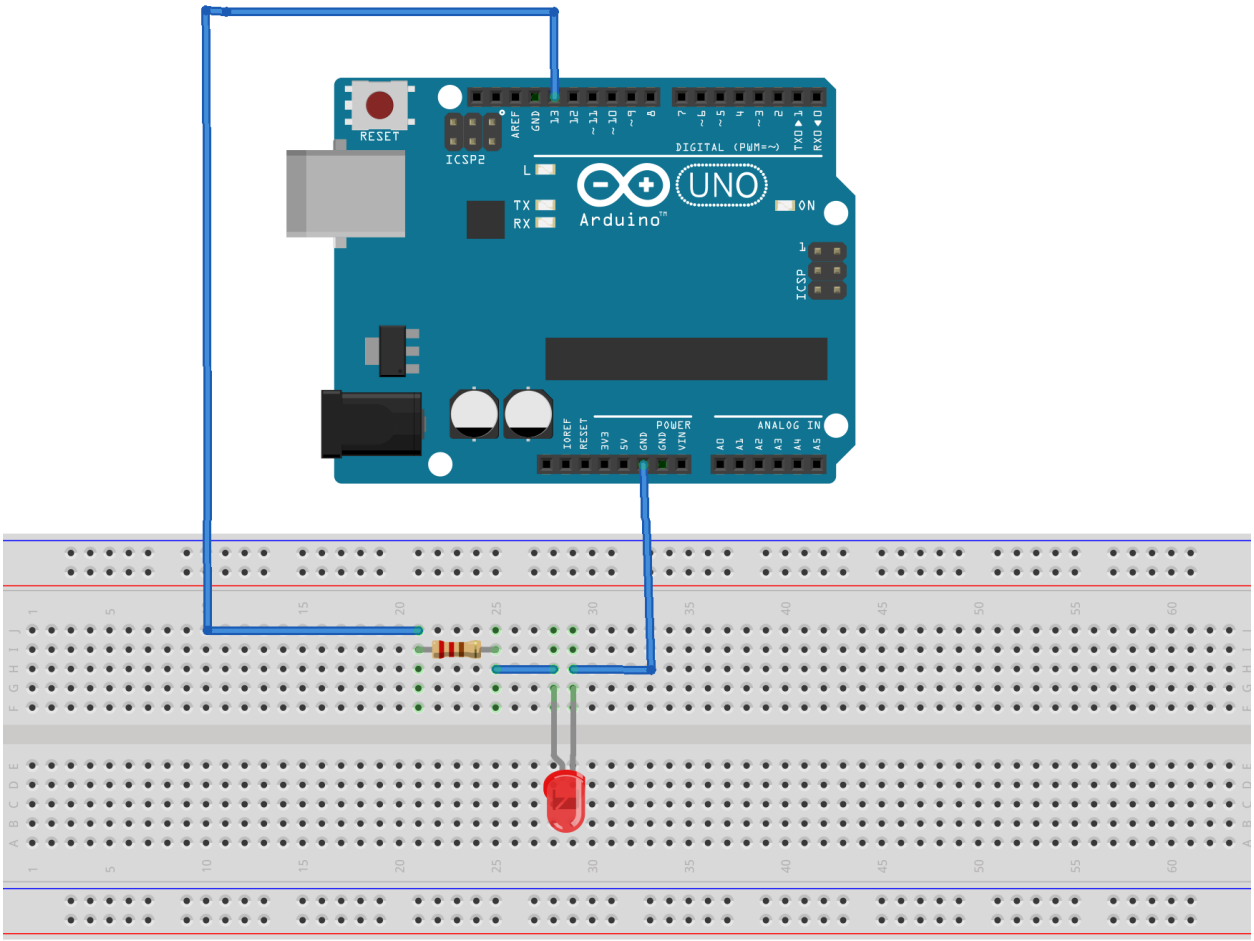
Simple LED Blinking Program

/**********Program**********/

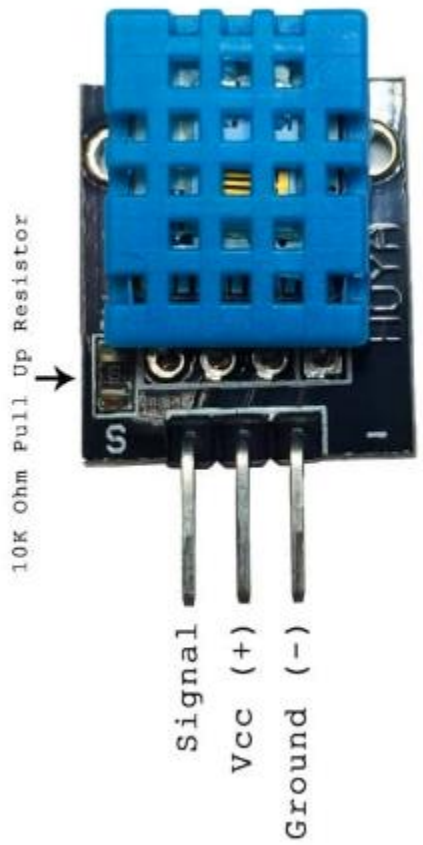
```
int led = 13; // the pin the LED is connected to
void setup() {
  pinMode(led, OUTPUT) // Declare the LED as an output
}

void loop() {
  digitalWrite(led, HIGH) // Turn the LED on
  delay(1000);
  digitalWrite(led, LOW) // Turn the LED off
  delay(1000);
}
```

/**********Program**********/

Connections

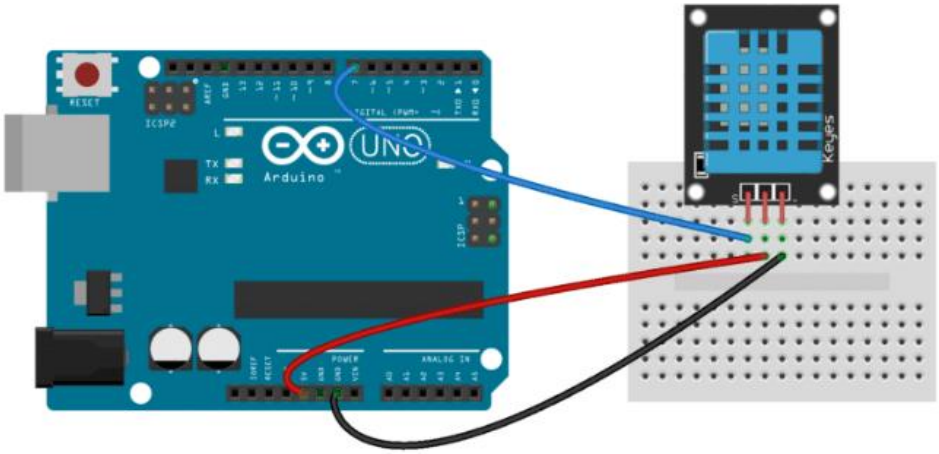


DHT11 Humidity and Temperature Sensor



The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.

Connections



------Program-------

```
#include <dht.h>

dht DHT;

#define DHT11_PIN 7

void setup(){
  Serial.begin(9600);
}

void loop(){
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(1000);
}
```

If we are using the pin other than pin 7, we need to modify the pin value in the statement #define DHT11_PIN 7

References:

1. https://en.wikipedia.org/wiki/Arduino
2. https://www.arduino.cc/