



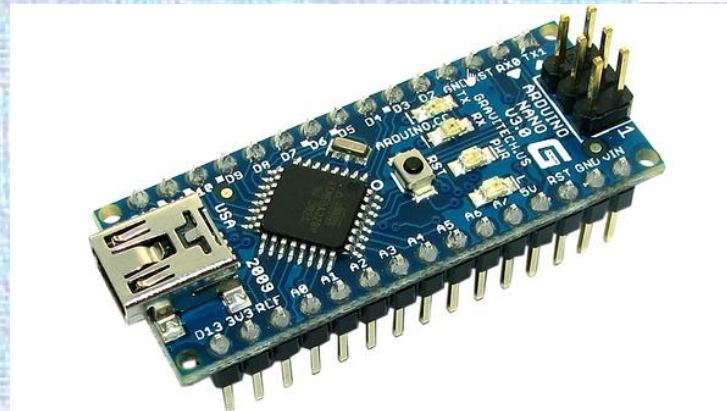
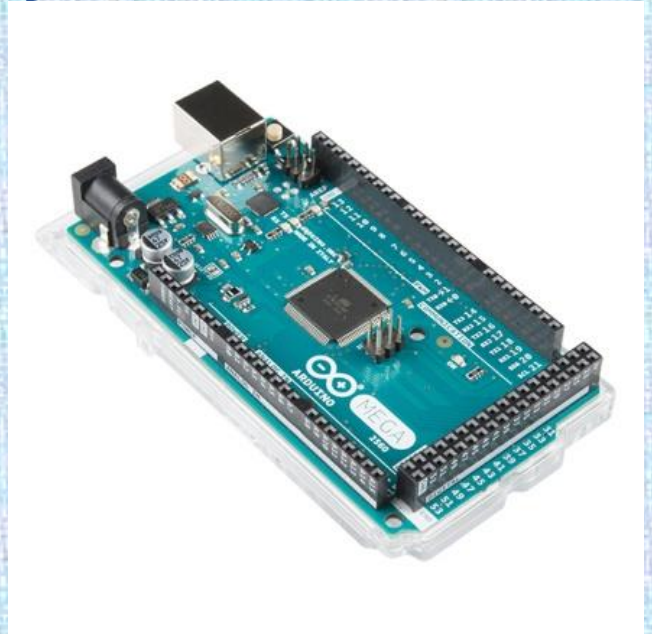
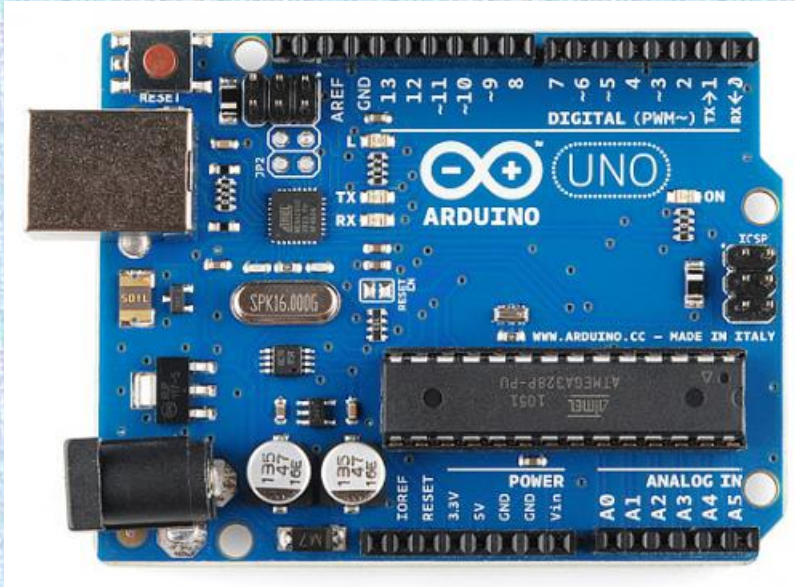
SAPIENZA
UNIVERSITÀ DI ROMA

Arduino



Alessandra de Vitis

Arduino types





Interfacing

Interfacing represents the link between devices that operate with different physical quantities.

*Interface board or simply or **interface** is what realizes interfacing*



interfacing

Interfacing make communication, between PC and other devices , possible.

Interface boards are typical device that allow to transmit or to receive informations from other device.



interfacing

An interface is made by a channel or transmission media (ie a cable), 2 connectors and 2 ports at the transmission ends. It is necessary to establish a **way of transmission (serial or parallel).**



interfacing

There are 2 kind of interfaces:
Proprietary interface (like a video board) **and Custom interface** for sensors and actuator.



Physical computing

One important interface board application field is **Physical computing.**



Physical computing

Physical computing is useful to realize systems which can interact with external environment using both hardware & software. The system interacts in both directions. It acquires and measures **Physical quantities** using sensors and actuators.



Sensor

In the broadest definition, a **SENSOR** is a device, module, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, frequently a computer processor. A sensor is always used with other electronics, whether as simple as a light or as complex as a computer.

Sensors are used in everyday objects such as touch-sensitive elevator buttons (tactile sensor) and lamps which dim or brighten by touching the base, besides innumerable applications of which most people are never aware.



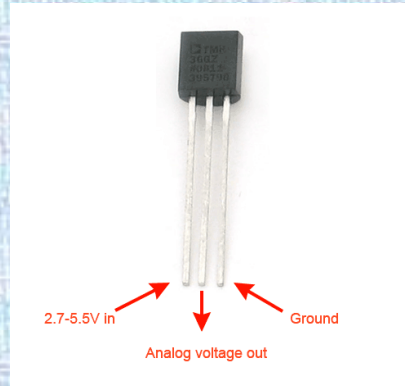
Sensor

A **good sensor** obeys the following rules:

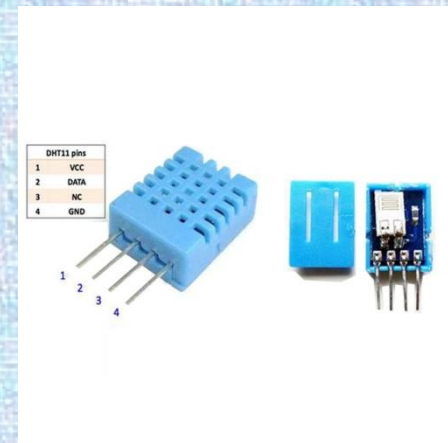
- it is **sensitive** to the measured property
- it is **insensitive** to any other property likely to be encountered in its application, and
- it **does not influence** the measured property.

Sensor examples

Temperature sensor



Humidity sensor



Light sensor





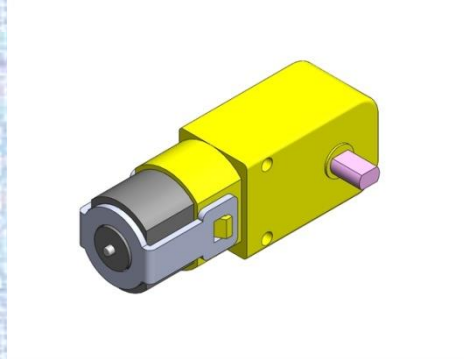
Actuator

An **actuator** is a component of a machine that is responsible for moving and controlling a mechanism or system, for example by opening a valve. In simple terms, it is a "mover".

An actuator requires a control signal and a source of energy. The control signal is relatively low energy and may be electric voltage or current, pneumatic or hydraulic pressure, or even human power. Its main energy source may be an electric current, hydraulic fluid pressure, or pneumatic pressure. When it receives a control signal, an actuator responds by converting the signal's energy into mechanical motion.

Actuator examples

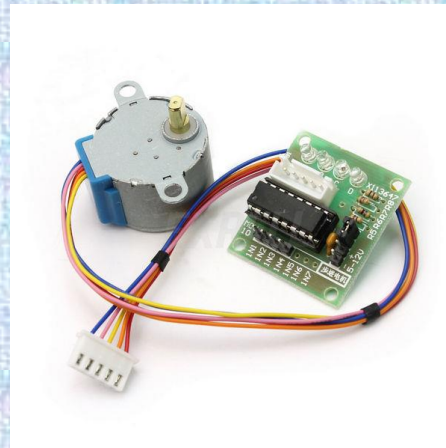
DC Motor



Servo Motor



Step Motor



Actuator examples

Led



speaker

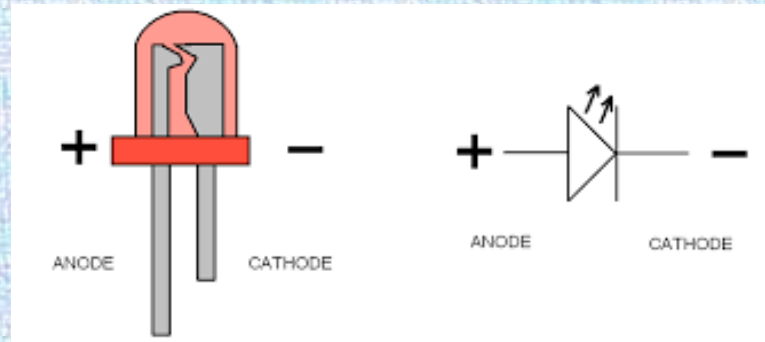


Buzzer

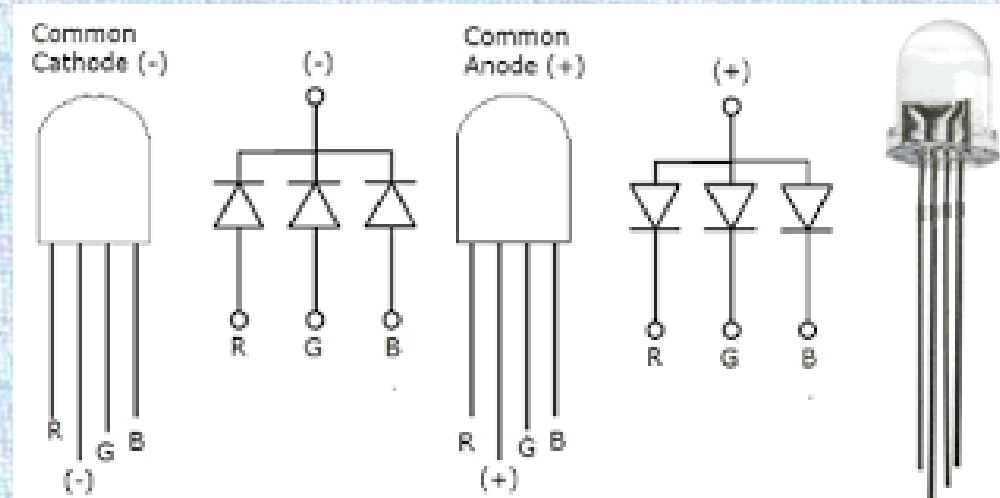


Led examples

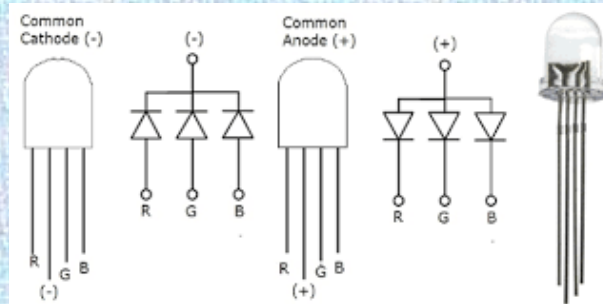
Led



RGB LED



RGB Led



Connect common cathode RGB to Ground

Connect common anode RGB to 5V

Always use a 220 ohm resistor for every pin.



Custom interface

Making custom interfaces is a complex process and you have to know:

- **How computer communication ports work.**
- **Which is the right protocol to use,**
- **Electronics**
- **Low level Programming language**



Developing board

Using a developing board we can easily make computer communicate with other devices thanks to a generic interface called **microcontroller.**



microcontroller

A microcontroller is an integrated microprocessor system in a single chip projected to work stand alone and to work in a specific application.



microcontroller

They have an execution unit, a memory module (RAM & ROM) and other I/O peripheral like analog to digital converters, timer, serial and parallel interface.



microcontroller

Microcontrollers are used in specific applications, typical of industrial control.



Developing boards

The most famous developing boards (Arduino, Raspberry, Edison & others) represent a good option to custom boards because they easily allow to develop prototype.

These boards are made by a microcontroller plus electronics to connect to PC communication ports.



Arduino Project

Arduino is a programmable hardware platform that can be easily connected with a PC.

Arduino is represented with a board and a software.

Arduino board has a memory in which you can store the program executed by ATmega328 microcontroller.

<https://www.arduino.cc>

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.



<https://www.arduino.cc>

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, and the Arduino Software (IDE).



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

Why Arduino?

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.

Why Arduino?

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.

Why Arduino?

Arduino is inexpensive

- Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than \$50.



Why Arduino?

Arduino is a Cross-platform

- The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems.

Why Arduino?

Arduino is a simple, clear programming environment

- The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

Why Arduino?

Arduino is an open source and extensible software & hardware platform

- The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries,
- The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.



key features

Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.

You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).



key features

Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.

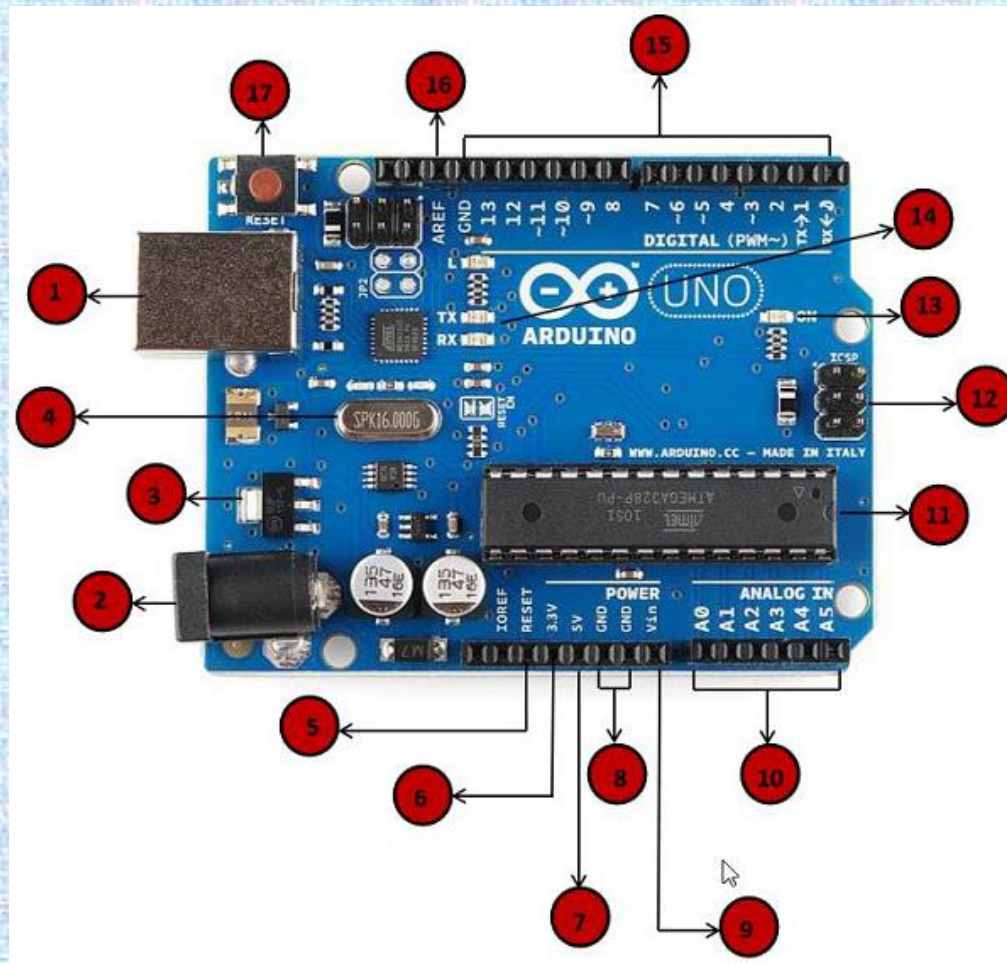
Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.



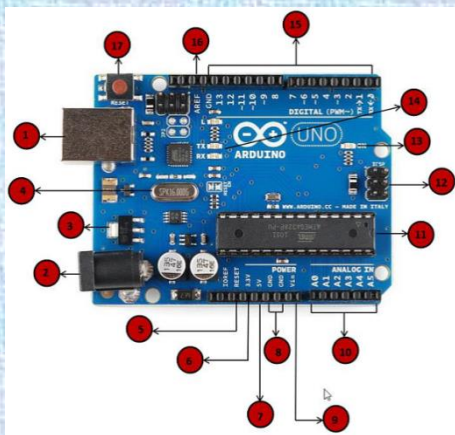
key features

Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

Arduino - Board Description



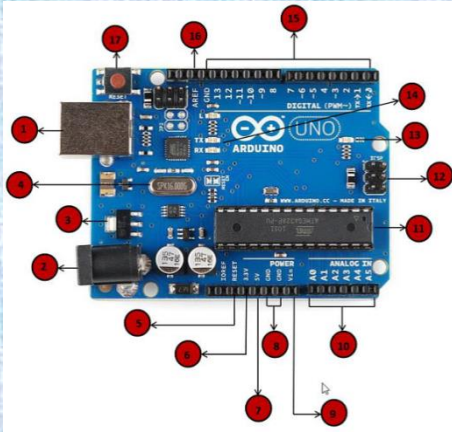
Arduino - Board Description



1) Power USB

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection.

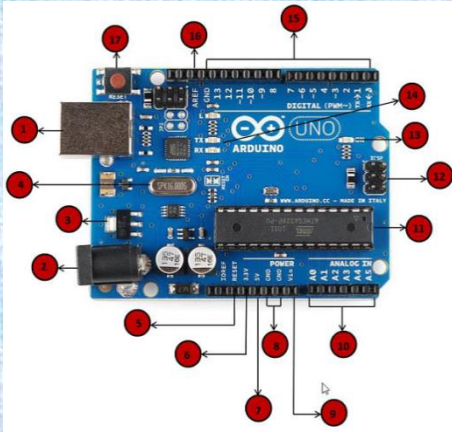
Arduino - Board Description



2) Power (Barrel Jack)

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack .

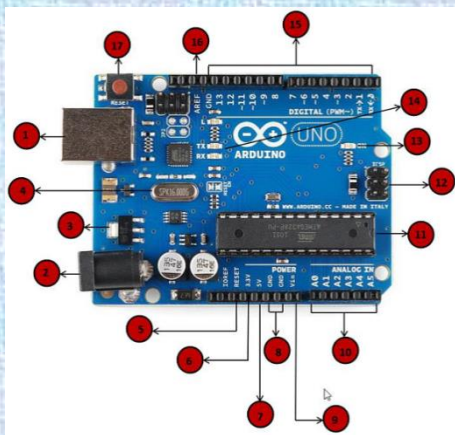
Arduino - Board Description



3) Voltage Regulator

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

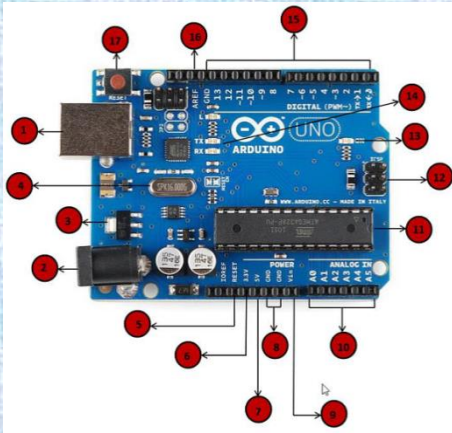
Arduino - Board Description



4) Crystal Oscillator

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

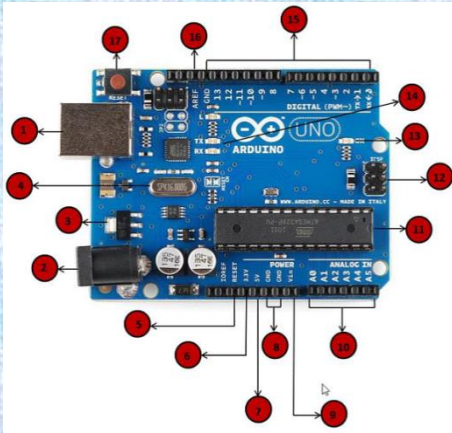
Arduino - Board Description



5 -17) Arduino Reset

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

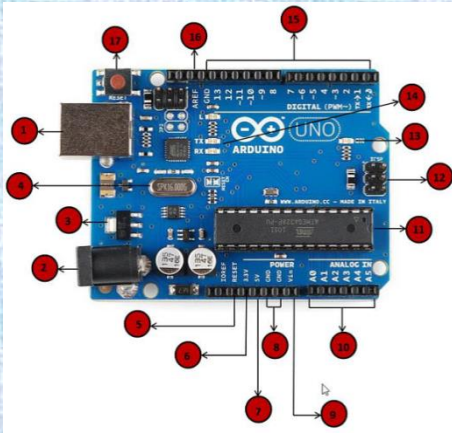
Arduino - Board Description



6-7-8-9) Pins (3.3, 5, GND, Vin)

- **3.3V (6) – Supply 3.3 output volt**
- **5V (7) – Supply 5 output volt**
- **Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.**
- **GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.**
- **Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.**

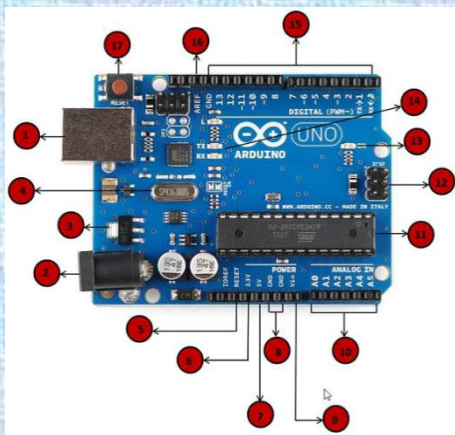
Arduino - Board Description



11) Main microcontroller

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

Arduino - Board Description



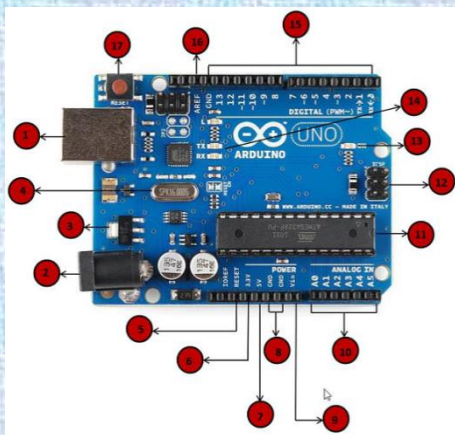
12) ICSP pin

In Circuit Serial Programming

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

It is one of the several methods available for programming Arduino boards. Ordinarily, an Arduino bootloader program is used to program an Arduino board, but if the bootloader is missing or damaged, ICSP can be used instead. ICSP can be used to restore a missing or damaged bootloader.

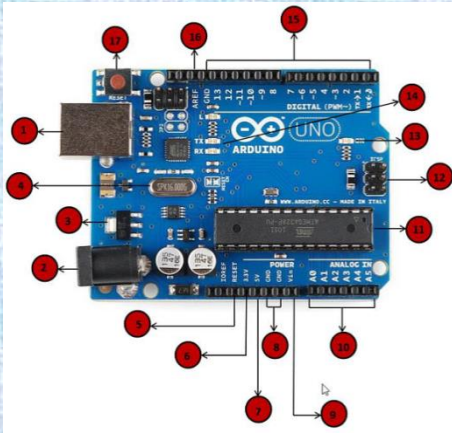
Arduino - Board Description



13) Power LED indicator

This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

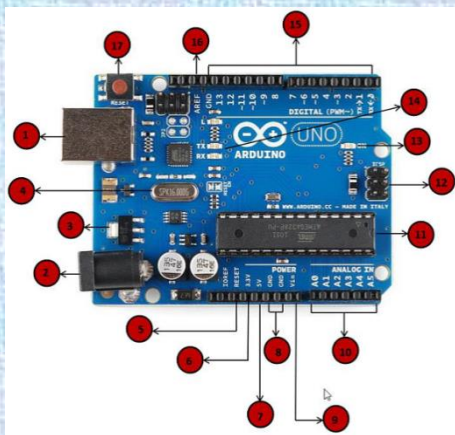
Arduino - Board Description



14) TX and RX LEDs

On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

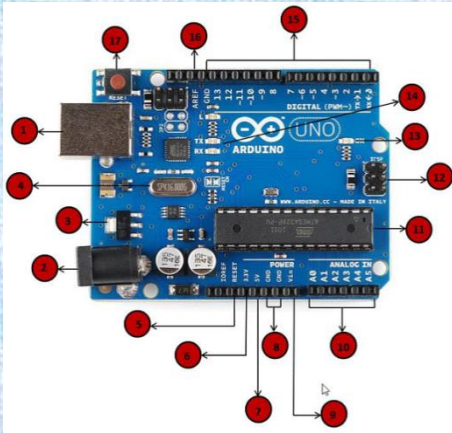
Arduino - Board Description



15) Digital I/O

The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.

Arduino - Board Description



16) AREF

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.



Arduino – Tech Specs

| | |
|---------------------------|---|
| Microcontroller | ATmega328P – 8 bit AVR family microcontroller |
| Operating Voltage | 5V |
| Recommended Input Voltage | 7-12V |
| Input Voltage Limits | 6-20V |
| Analog Input Pins | 6 (A0 – A5) |
| Digital I/O Pins | 14 (Out of which 6 provide PWM output) |
| DC Current on I/O Pins | 40 mA |
| DC Current on 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (0.5 KB is used for Bootloader) |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Frequency (Clock Speed) | 16 MHz |



Arduino - Installation

– 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.

In case you use Arduino Nano, you will need an A to Mini-B cable instead.



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).



SAPIENZA
UNIVERSITÀ DI ROMA

Download Arduino IDE Software.

The screenshot shows the Arduino website homepage. At the top is a teal navigation bar with the Arduino logo on the left and search, cart, and 'SIGN IN' links on the right. Below the navigation bar is a main content area with several sections. On the left, there's a 'WHAT IS ARDUINO?' section with an image of an Arduino Uno board and three buttons: 'BUY AN ARDUINO' (orange), 'LEARN ARDUINO' (teal), and 'DONATE' (brown). In the center, there's a 'BLOG' section with an image of a robotic arm and the text 'ANIMATE A SODA BOTTLE STRUCTURE WITH TRUSSFORMER AND ARDUINO'. On the right, there's a section with the Arduino logo and links: '> ARDUINO', '> WEB EDITOR', and '> CODE ONLINE!'. Below this is a screenshot of the Arduino IDE web editor showing code for 'void setup()' and 'void loop()'. At the bottom, there are two more sections: 'THE ARDUINO' with an image of a circuit board and 'NEW BOARDS AND' with an image of a robotic arm.

ARDUINO

HOME STORE SOFTWARE EDUCATION RESOURCES COMMUNITY HELP

WHAT IS ARDUINO?

BUY AN ARDUINO

LEARN ARDUINO

DONATE

BLOG

ANIMATE A SODA BOTTLE STRUCTURE WITH TRUSSFORMER AND ARDUINO

> ARDUINO
> WEB EDITOR
> CODE ONLINE!

void setup(){
}

void loop(){
}

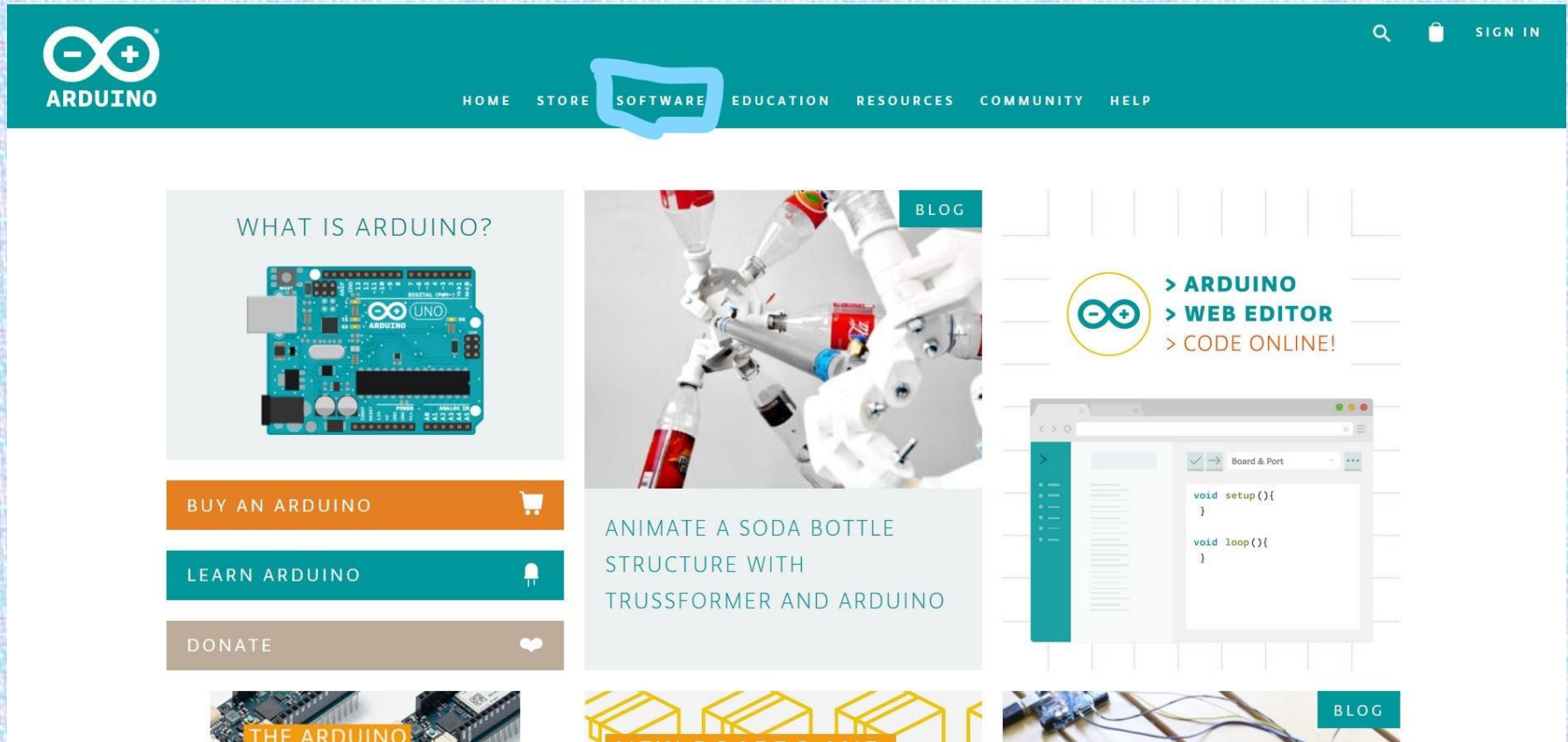
THE ARDUINO

NEW BOARDS AND



SAPIENZA
UNIVERSITÀ DI ROMA

Download Arduino IDE Software.



The screenshot shows the Arduino website homepage. The top navigation bar is teal with the Arduino logo on the left and links for HOME, STORE, SOFTWARE (highlighted with a blue hand-drawn box), EDUCATION, RESOURCES, COMMUNITY, and HELP. On the right of the bar are search, cart, and SIGN IN icons. The main content area features several promotional tiles: 'WHAT IS ARDUINO?' with an image of an Arduino Uno board and buttons for 'BUY AN ARDUINO', 'LEARN ARDUINO', and 'DONATE'; a 'BLOG' tile with an image of a robotic arm and the text 'ANIMATE A SODA BOTTLE STRUCTURE WITH TRUSSFORMER AND ARDUINO'; a 'BLOG' tile with the Arduino logo and text '> ARDUINO > WEB EDITOR > CODE ONLINE!'; and a tile showing the Arduino IDE web editor interface with code for setup() and loop(). At the bottom, there are partial views of 'THE ARDUINO' and 'NEW BOARDS' tiles.



Download Arduino IDE Software.

The screenshot shows the Arduino website's 'SOFTWARE' page. At the top, there's a navigation bar with links: HOME, STORE, SOFTWARE, EDUCATION, RESOURCES, COMMUNITY, and HELP. Below this, the main content area features a large banner for the 'ARDUINO WEB EDITOR' with a 'GETTING STARTED' button and a 'CODE ONLINE' button. Below the banner is a section titled 'THE ARDUINO MKR FAMILY JUST GOT EVEN BIGGER! SEE THEM ALL!'. Further down, the 'Download the Arduino IDE' section is visible, featuring the Arduino logo and text about the 'ARDUINO 1.8.7' release. To the right of this section, there are links for 'Windows installer', 'Windows app', 'Mac OS X 10.8 Mountain Lion or newer', 'Linux 32 bits', 'Linux 64 bits', 'Linux ARM', 'Release Notes', 'Source Code', and 'Checksums (sha512)'. At the bottom, there are sections for 'HOURLY BUILDS' and 'BETA BUILDS'.

ARDUINO 1.8.7
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, for Windows XP and up
Windows zip file for non-admin install
Windows app Requires Win 8.1 or 10
Get it
Mac OS X 10.8 Mountain Lion or newer
Linux 32 bits
Linux 64 bits
Linux ARM
Release Notes
Source Code
Checksums (sha512)

HOURLY BUILDS
LAST UPDATE: 18 October 2018 14:25:19 GMT
Download a preview of the incoming release with the most updated features and bugfixes.
Windows
Mac OS X (Mac OS X Mountain Lion or later)
Linux 32 bit, Linux 64 bit, Linux ARM, Linux ARM64 (experimental)

BETA BUILDS
Download the Beta Version of the Arduino IDE with experimental features. This version should NOT be used in production.
Windows
Mac OS (Mac OS X Mountain Lion or later)
Linux 32 bit, Linux 64 bit, Linux ARM, Linux ARM64 (experimental)



Power up your board

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

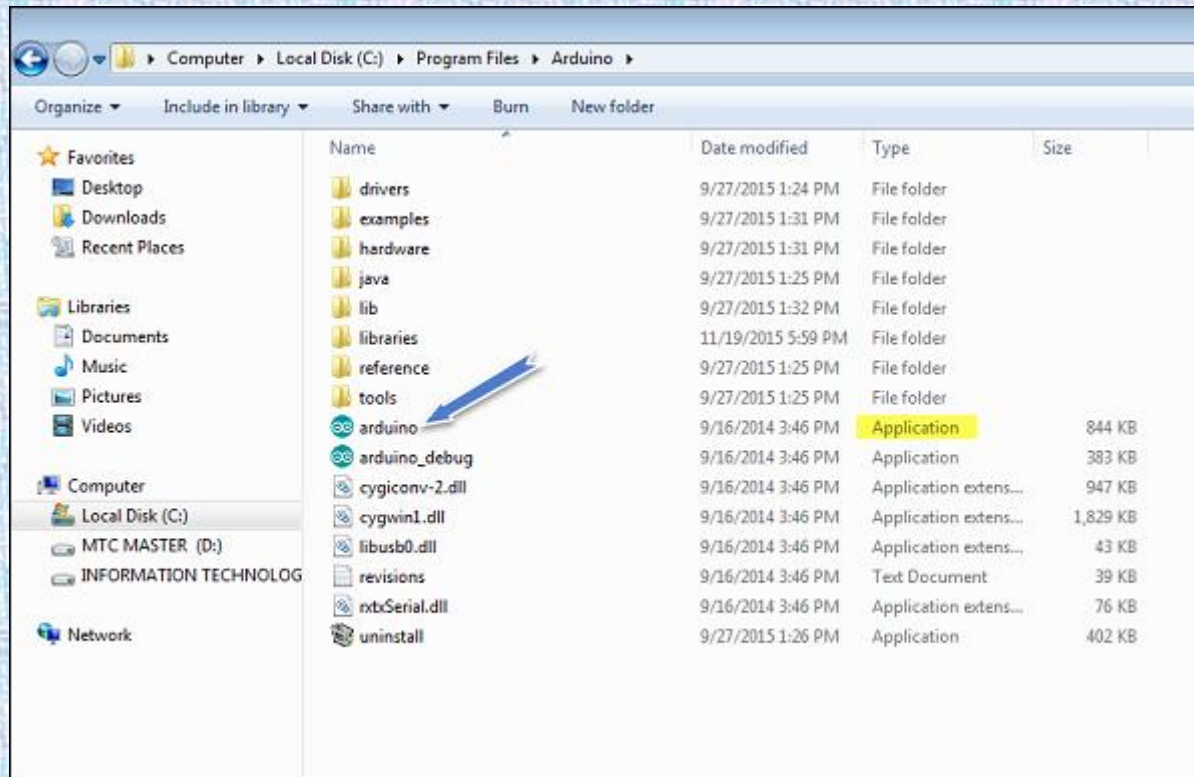


Launch Arduino IDE

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.

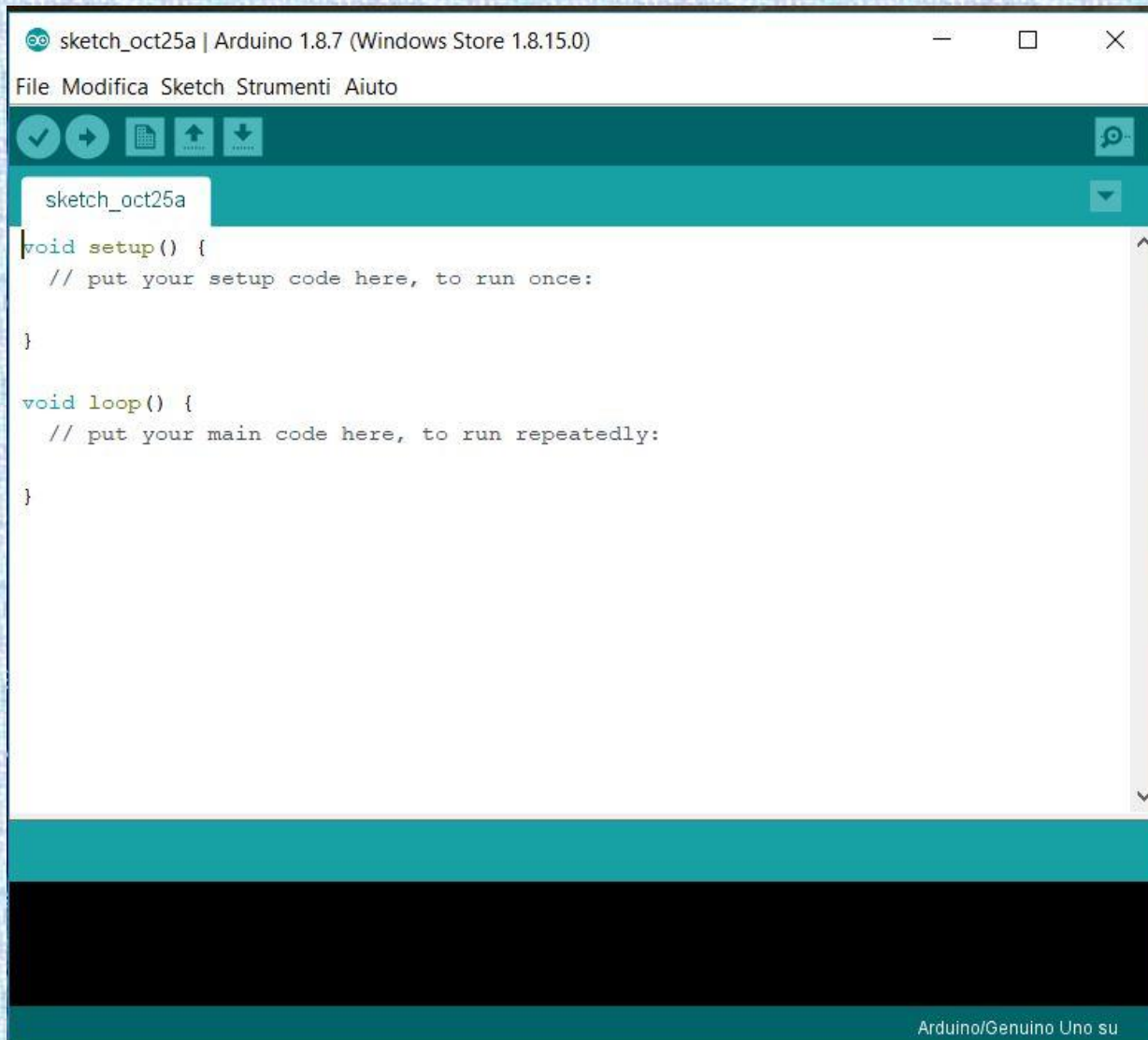


Launch Arduino IDE





Launch Arduino IDE





Arduino Sketch

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.

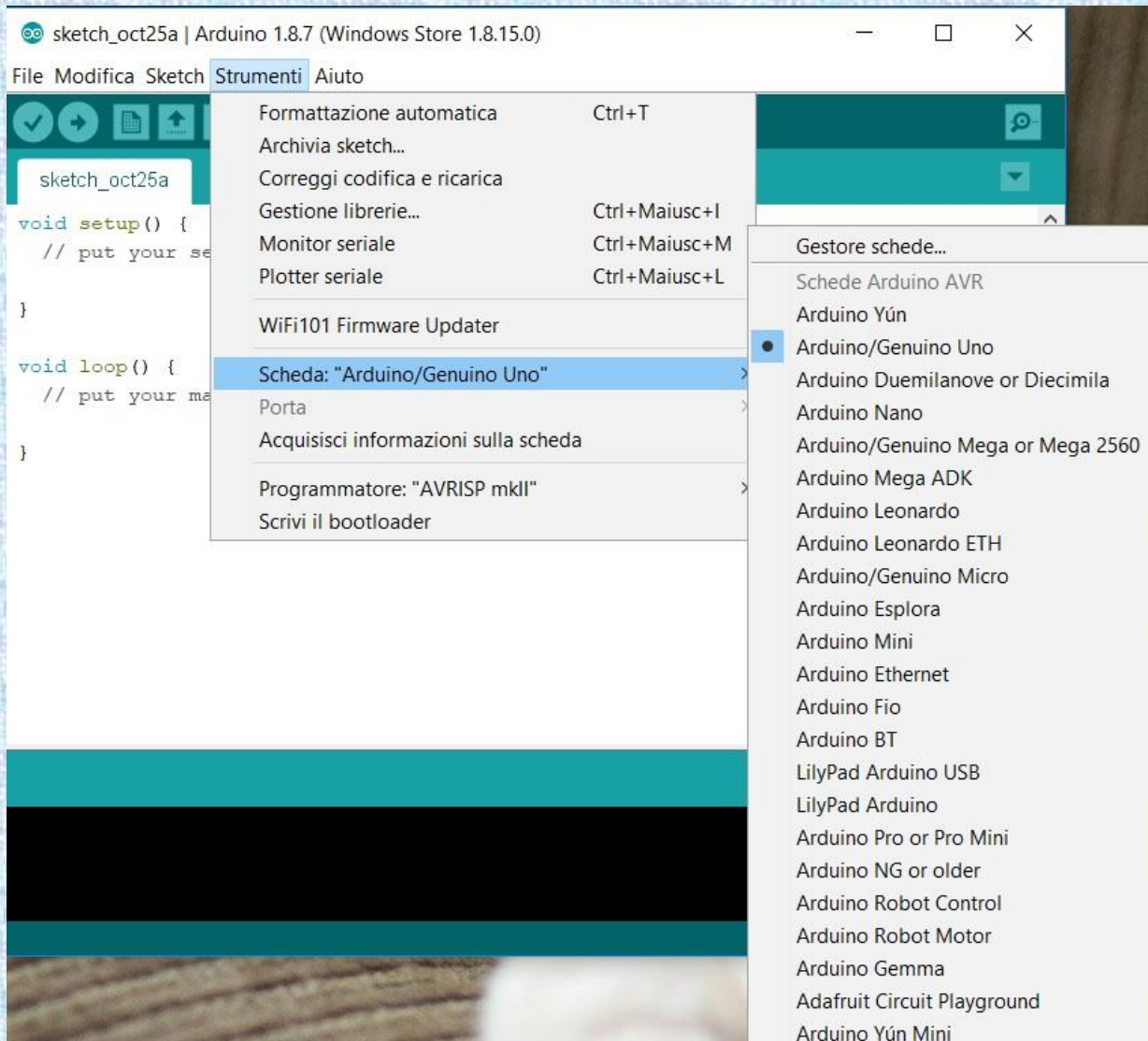


Arduino Sketch

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

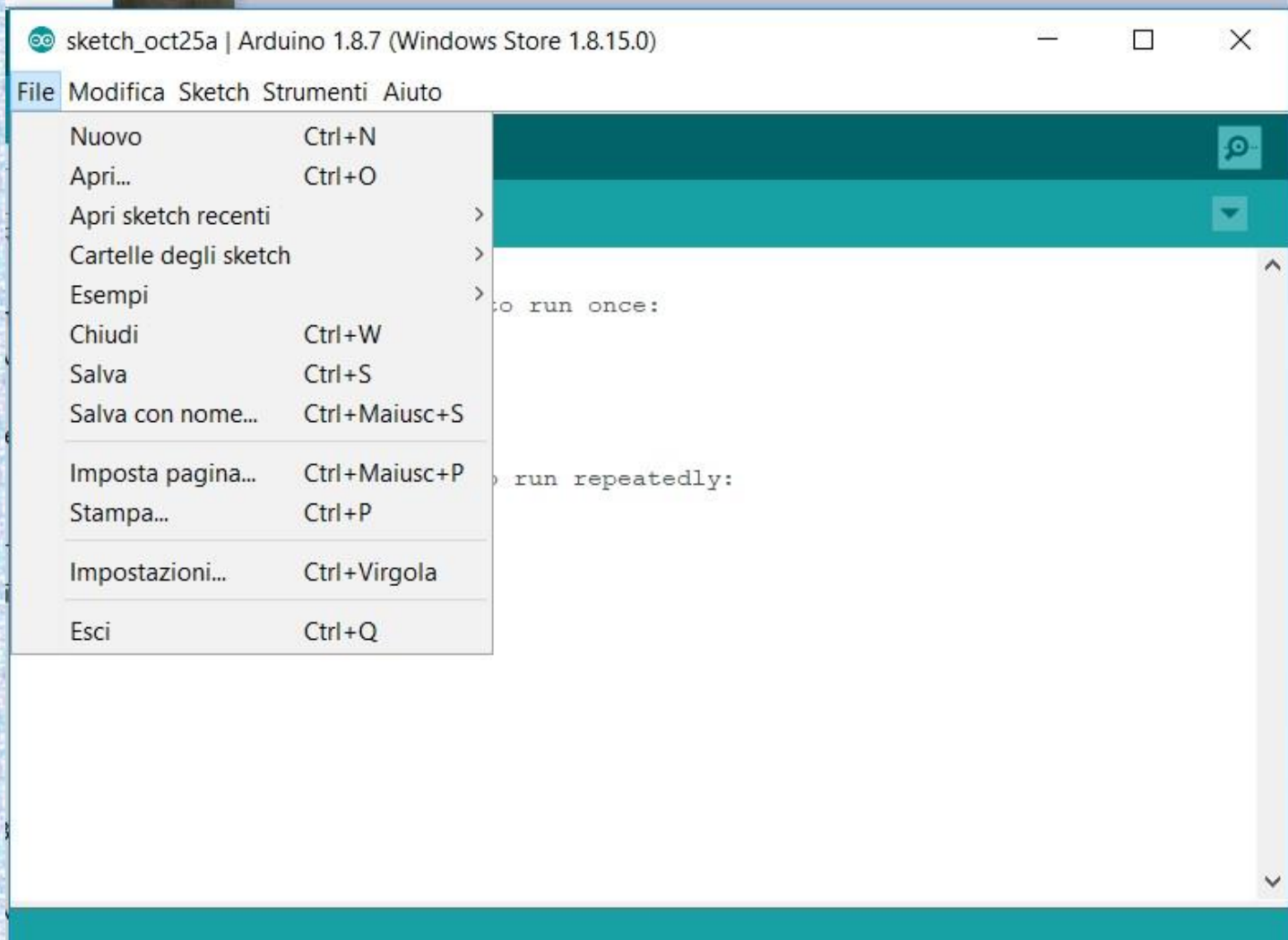


Choose Arduino Board





Choose Arduino Sketch





Open your first project

Once the software starts, you have two options:

Create a new project.

Open an existing project example.

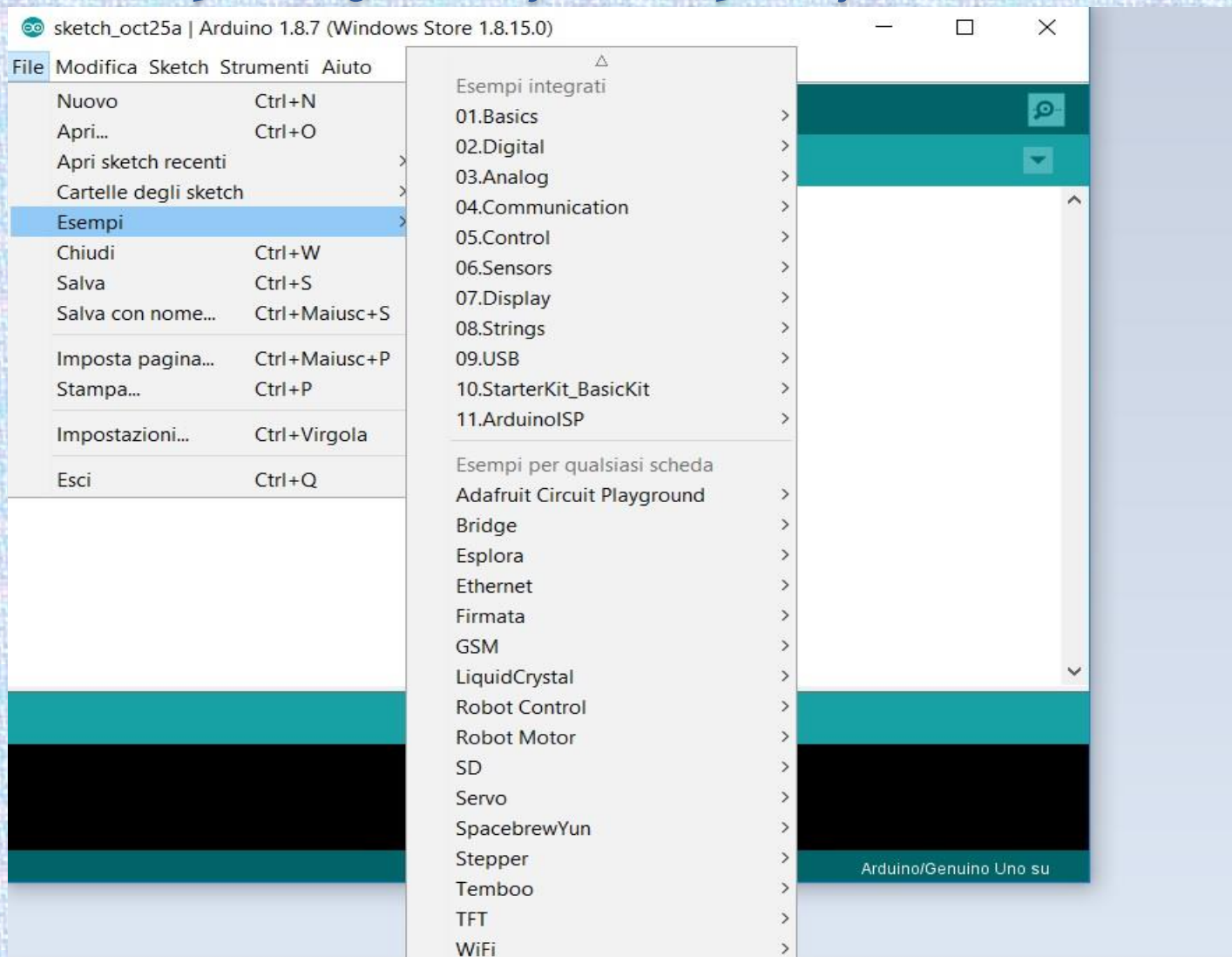
To create a new project, select File → New.

To open an existing project example, select File → Example → Basics → Blink.

Open Project

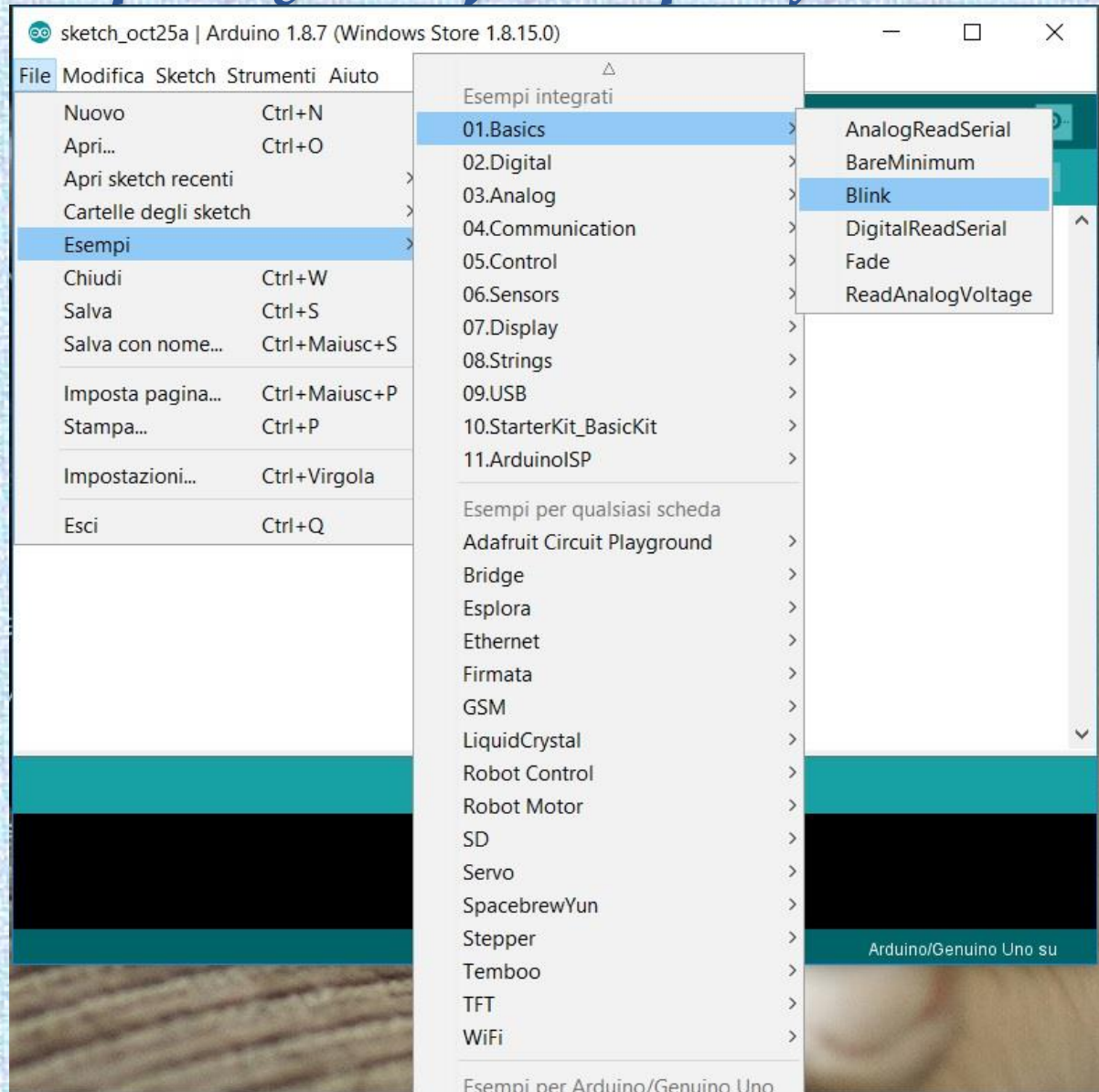


Open your first project



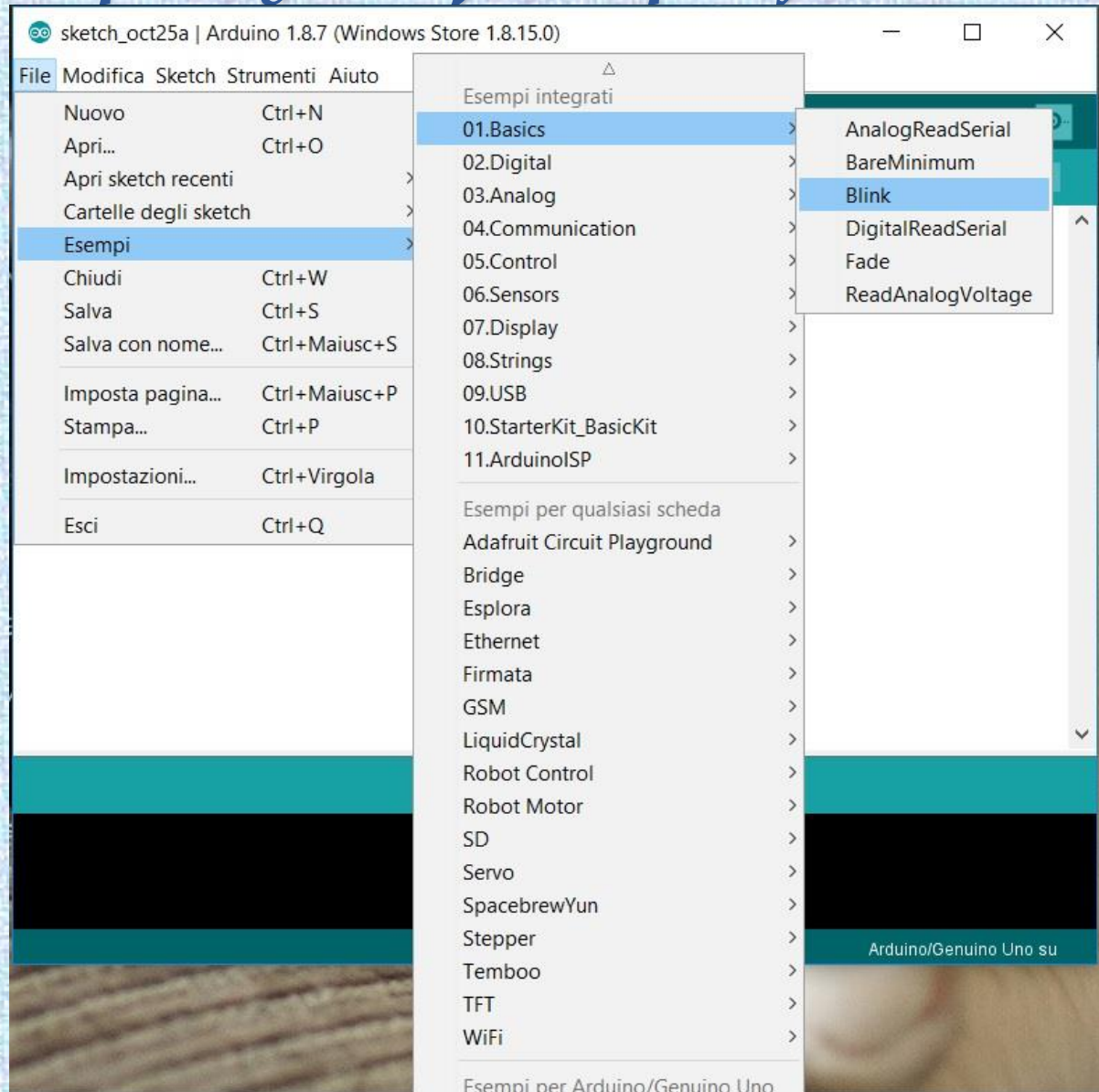


Open your first project





Open your first project: Blink





Blink

Blink

Turns 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>

modified 8 May 2014

by Scott Fitzgerald

modified 2 Sep 2016

by Arturo Guadalupi

modified 8 Sep 2016

by Colby Newman

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/Blink>

```
*/  
  
// 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  
}
```



RGB LED 1

// RGB LED

```
int LRed = 9;  
int LGreen = 10;  
int LBlu = 11;  
int del = 20; // attesa di 20  
ms per percepire il colore  
int valRed;  
int valGreen;  
int valBlu;
```



RGB LED 2

```
void setup() {  
  pinMode(LRed, OUTPUT);  
  pinMode(LGreen, OUTPUT);  
  pinMode(LBlu, OUTPUT);  
}
```



RGB LED 3

```
void loop() {  
  valBlu = 255;  
  valGreen = 255;  
  valRed = 0;  
  for(int i = 0 ; i < 255 ; i += 1){  
    valBlu -= 1;  
    valRed += 1;  
    analogWrite(LRed, valRed);  
    analogWrite(LBlu, valBlu);  
    analogWrite(LGreen, valGreen);  
    delay(del);  
  }  
}
```



RGB LED 4

```
for(int i = 0 ; i < 255 ; i += 1)
{
  valGreen -= 1;
  valBlu += 1;
  analogWrite(LRed, valRed);
  analogWrite(LBlu, valBlu);
  analogWrite(LGreen, valGreen);
  delay(del);
}
```



RGB LED 5

```
for(int i = 0 ; i < 255 ; i += 1)
{
  valGreen += 1;
  valRed -= 1;
  analogWrite(LRed, valRed);
  analogWrite(LBlu, valBlu);
  analogWrite(LGreen, valGreen);
  delay(del);
}
}
```