

Programming with Arduino

DOTT. GUIDO BORGHI

A.A. 2020/21

INNOVATION DESIGN

A project with Arduino

In order to create a project with Arduino there are two part to consider:

1. Hardware part

- The «real» **components** of the project:
 - Circuit
 - Sensors
 - Actuators
 - ...

2. Software part

- The **logic** of the project:
 - Setup function
 - Loop function

Arduino simulator

We use a **simulator** of the Arduino embedded board:



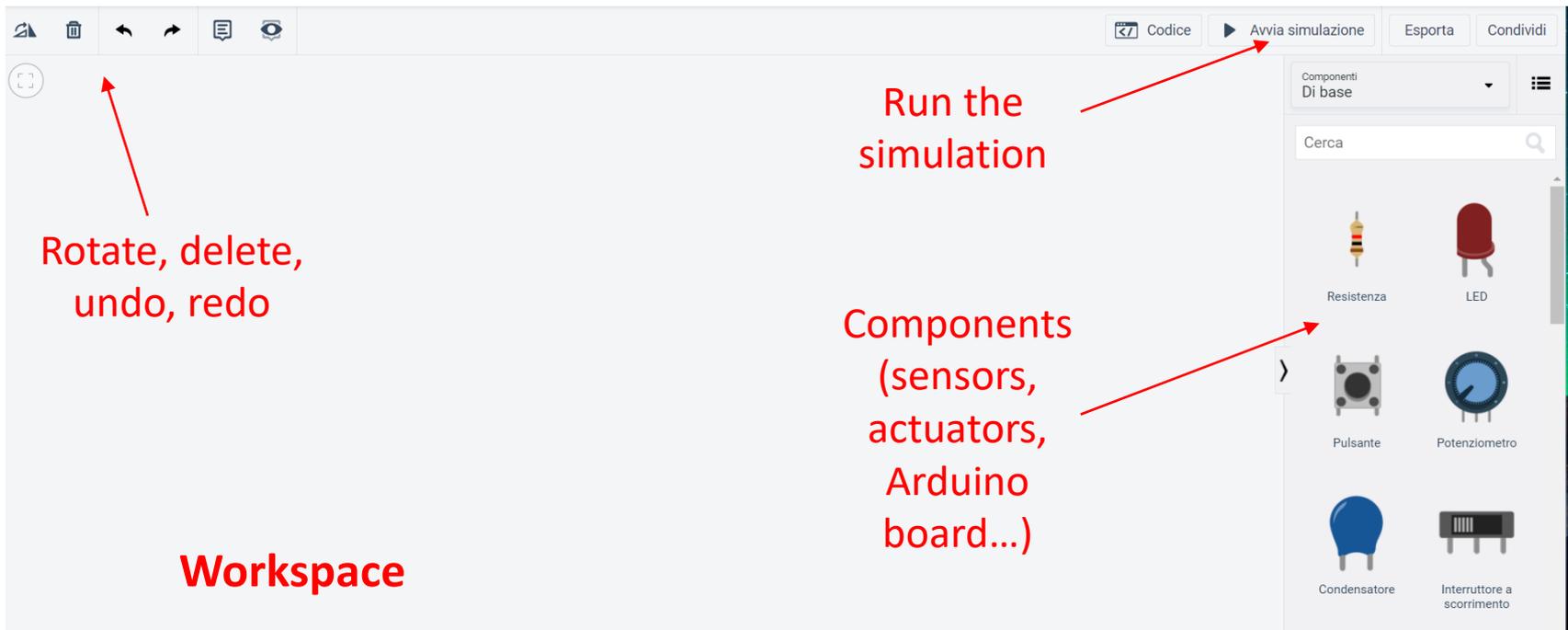
<https://www.tinkercad.com/>

Tinkercad

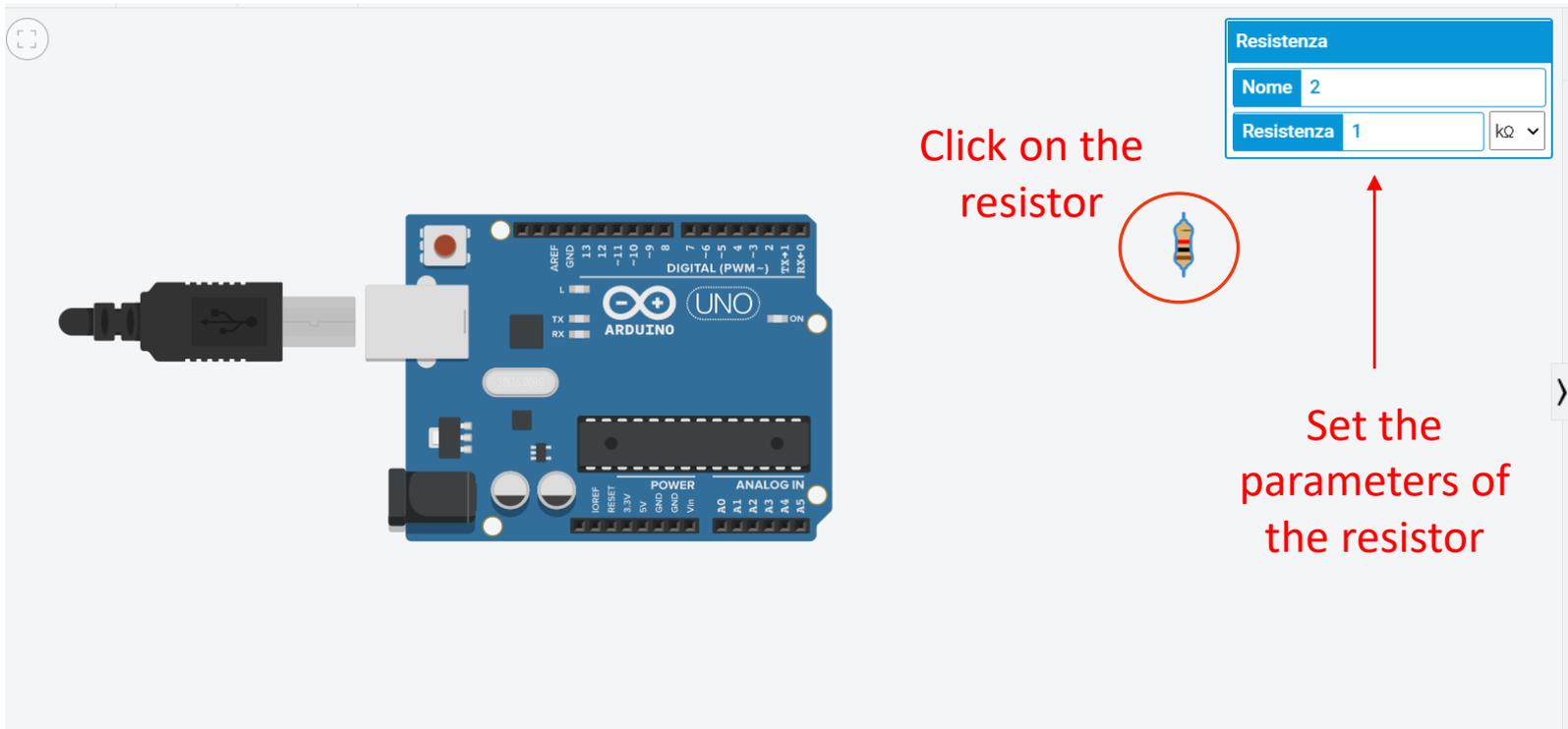
The image shows the Tinkercad web interface. On the left is a sidebar with a user profile icon at the top. Below it is a search bar labeled 'Cerca progetti...'. The sidebar lists several categories: 'Progetti 3D', 'Circuits' (highlighted in blue with an orange arrow pointing to it), 'Codeblocks' (with a 'NUOVO' badge), and 'Lezioni'. Below these is a section 'Partecipa alla lezione' and a 'Progetti' section with a 'Project 1' card and a '+ Crea un progetto' button. The main content area is titled 'Circuits' and features a green button 'Crea nuovo circuito' with an orange arrow pointing to it. Below the button are three project cards, each showing a circuit diagram with an Arduino Uno and a breadboard. The first card is 'Brilliant Hango' (3 ore fa, Privato), the second is 'Terrific Stantia-Albar' (3 ore fa, Privato), and the third is 'Fantastic Bruticus' (16 giorni fa, Privato). Each card has a heart icon and a comment icon.

Tinkercad

Open the code section



Tinkercad



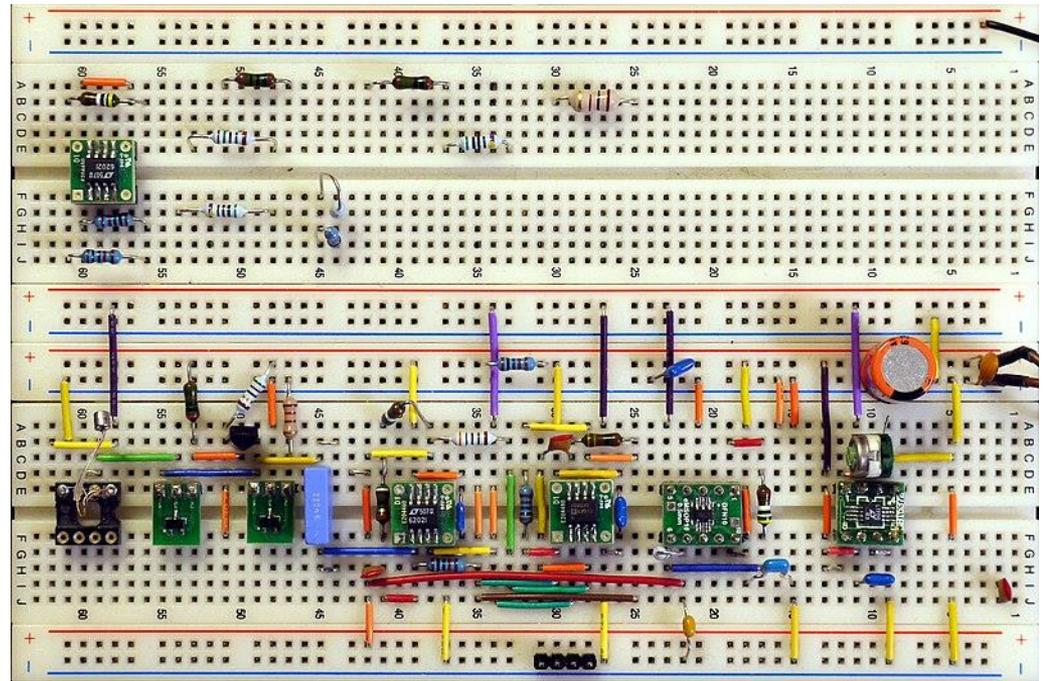
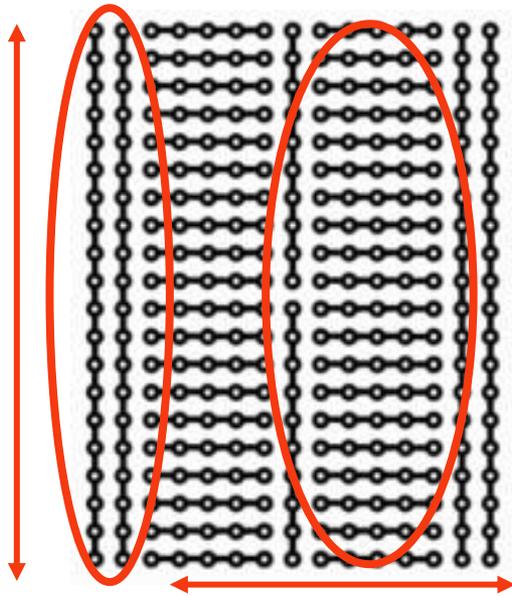
The image shows a screenshot of the Tinkercad software interface. On the left, there is a 3D model of an Arduino Uno board with a USB Type-C cable plugged into its port. On the right, a resistor component is highlighted with a red circle. To the right of the resistor is a configuration panel titled "Resistenza". The panel has three fields: "Nome" with the value "2", "Resistenza" with the value "1", and a unit dropdown menu set to "kΩ". A red arrow points from the text "Set the parameters of the resistor" to the "Resistenza" field. Another red arrow points from the text "Click on the resistor" to the resistor component.

Click on the resistor

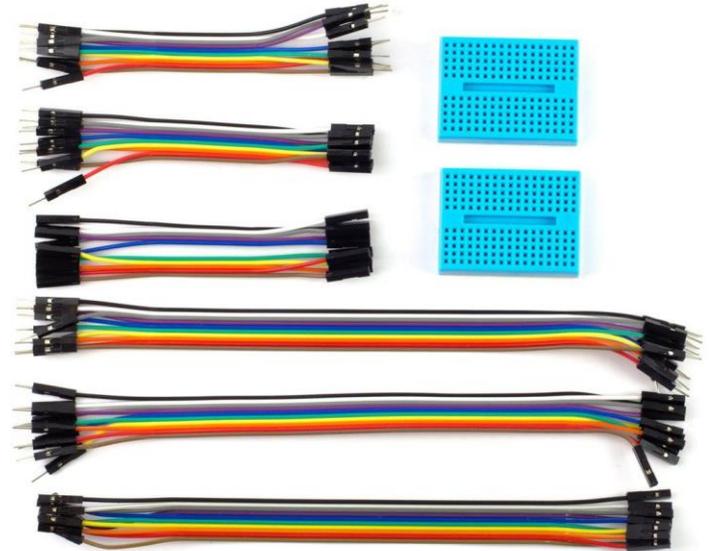
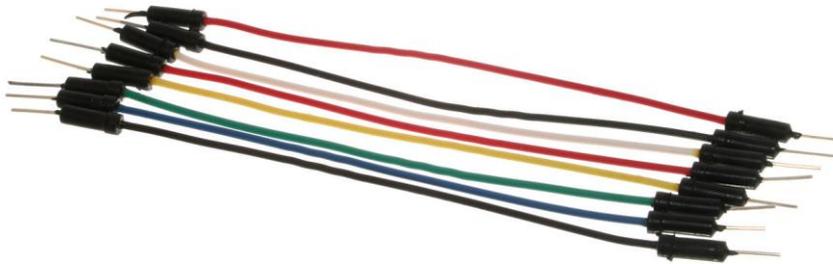
Set the parameters of the resistor

Resistenza	
Nome	2
Resistenza	1
	kΩ

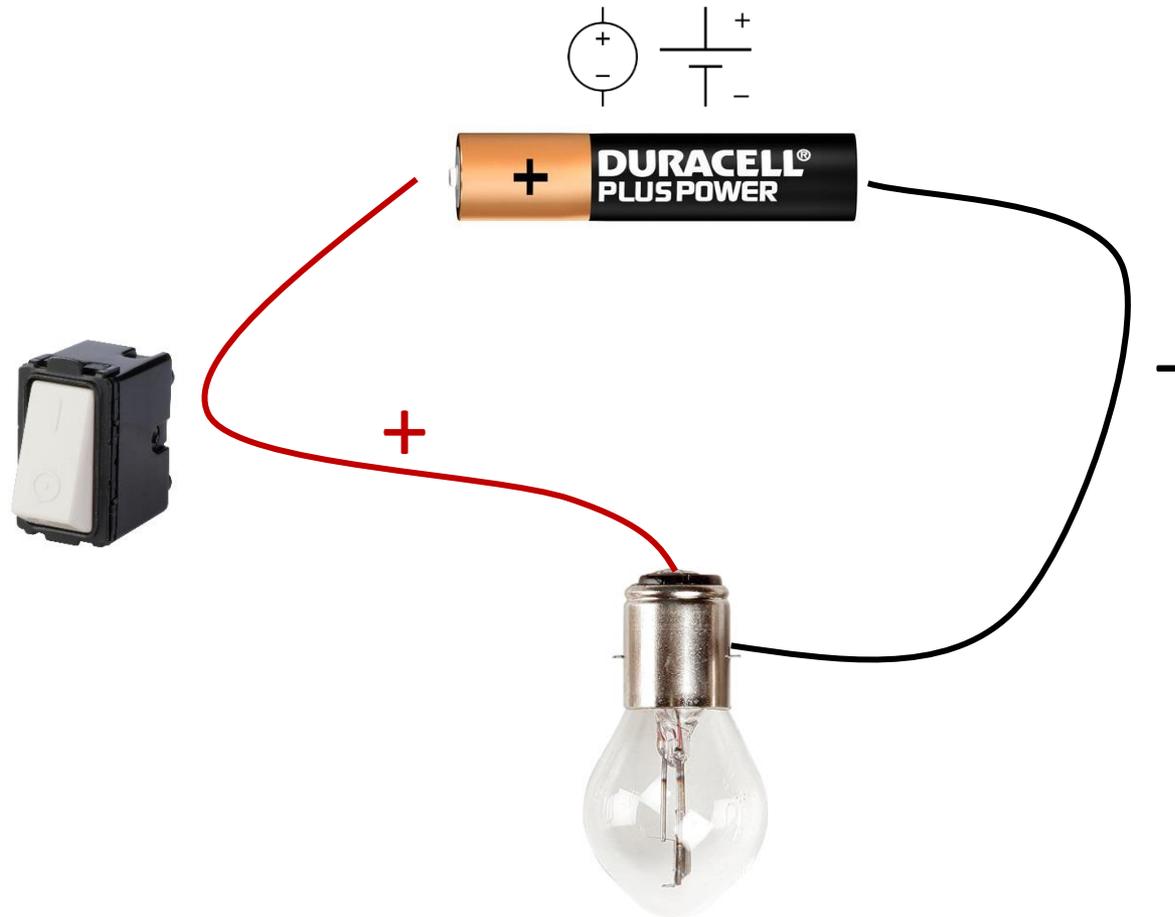
Breadboard



Breadboard wires



Circuit

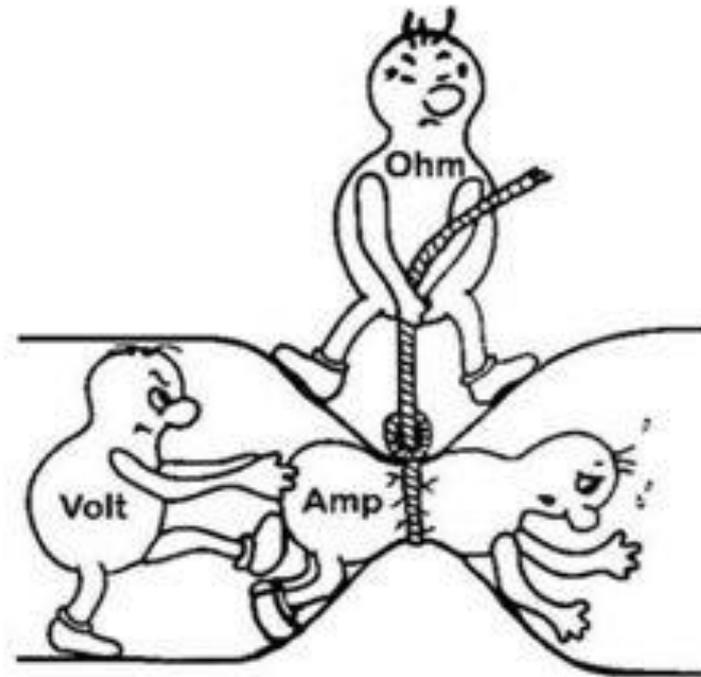


Ampere, Volt and Ohm

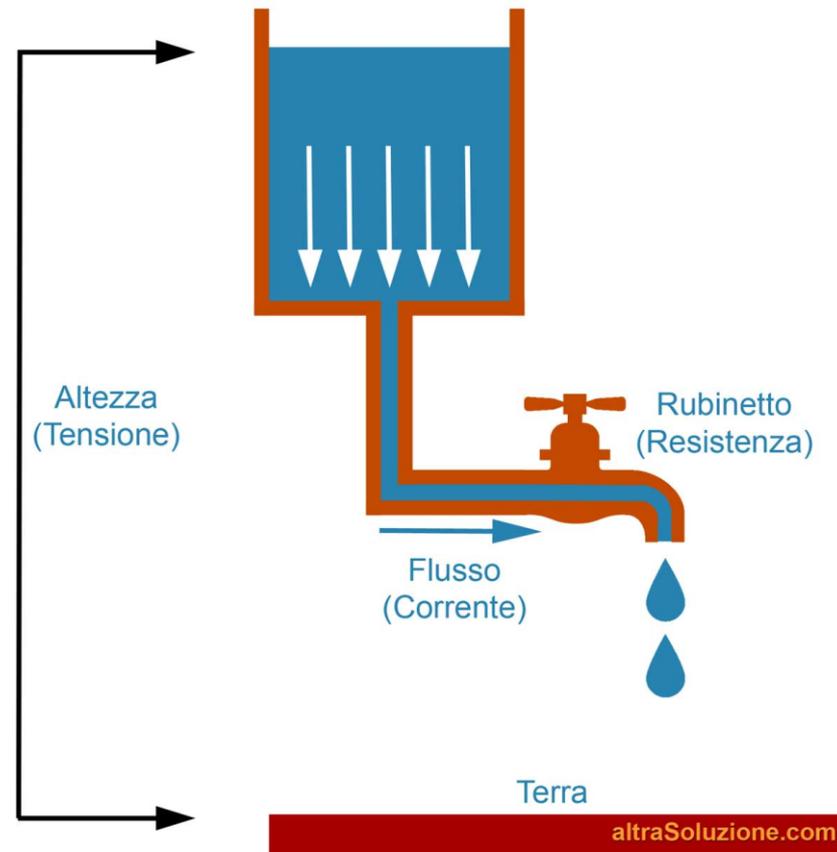
Voltage (Volt, V)

Current (Ampere, A)

Electrical resistance (Ohm, Ω)



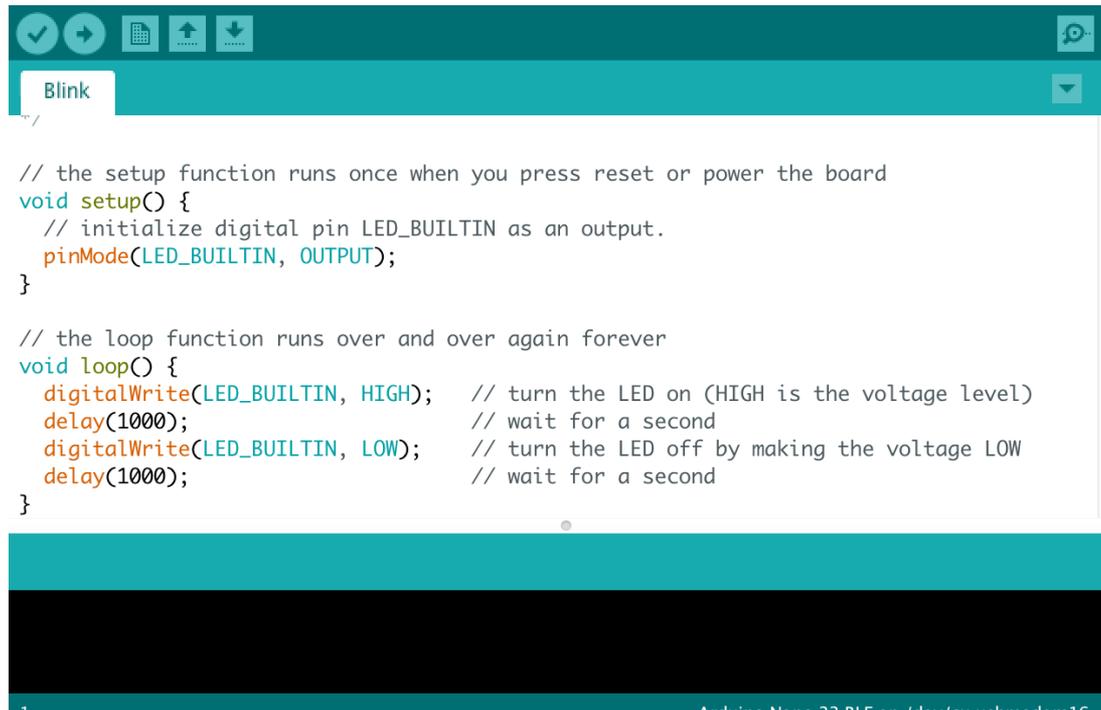
Comparison



Software

Arduino has its *Integrated Development Environment* (IDE)

- <https://www.arduino.cc/en/main/software>



```
// 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
}
```

Software

HARDWARE

SOFTWARE ▾

DOCUMENTATION ▾

COMMUNITY ▾

BLOG

ABOUT

Download the Arduino IDE



ARDUINO 1.8.13

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 source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.



Windows Installer, for Windows 7 and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10



Mac OS X 10.10 or newer

Linux 32 bits

Linux 64 bits

Linux ARM 32 bits

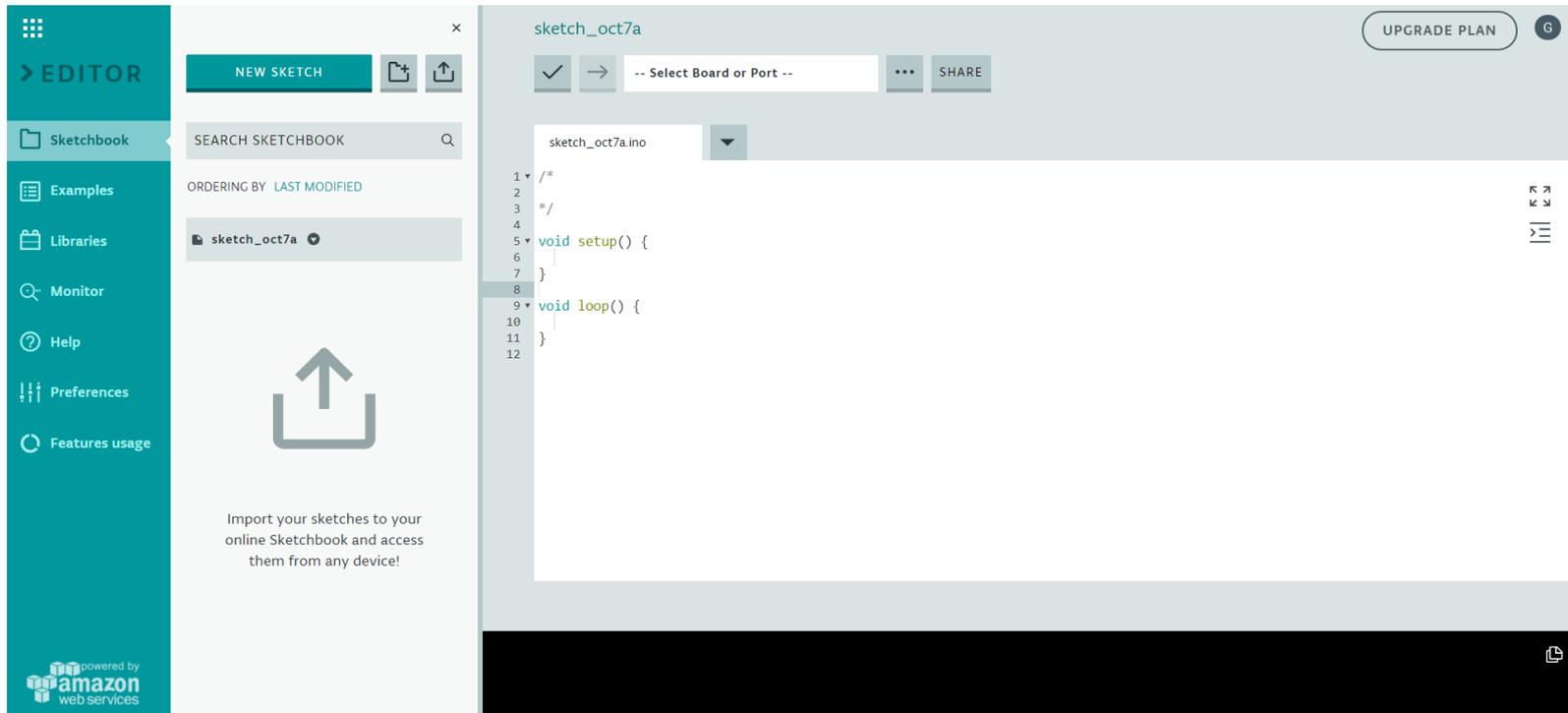
Linux ARM 64 bits

[Release Notes](#)

[Source Code](#)

[Checksums \(sha512\)](#)

IDE web editor



The screenshot displays the IDE web editor interface. On the left is a teal sidebar with navigation options: EDITOR, Sketchbook, Examples, Libraries, Monitor, Help, Preferences, and Features usage. At the bottom of the sidebar is the Amazon logo with the text "powered by amazon web services".

The main editor area is titled "sketch_oct7a" and includes a "NEW SKETCH" button, a search bar for "sketchbook", and a list of sketches ordered by "LAST MODIFIED". A large upload icon is present with the text: "Import your sketches to your online Sketchbook and access them from any device!".

The code editor shows the following code for "sketch_oct7a.ino":

```
1 /*
2
3 */
4
5 void setup() {
6
7 }
8
9 void loop() {
10
11 }
12
```

At the top right of the editor area, there is a "UPGRADE PLAN" button and a user profile icon. Below the code editor, there is a "SHARE" button and a "-- Select Board or Port --" dropdown menu.

The meaning of programming



programming

/ˈprɒɡræmɪŋ/

Vedi le definizioni in:

All

Computing

Broadcasting

Zoology

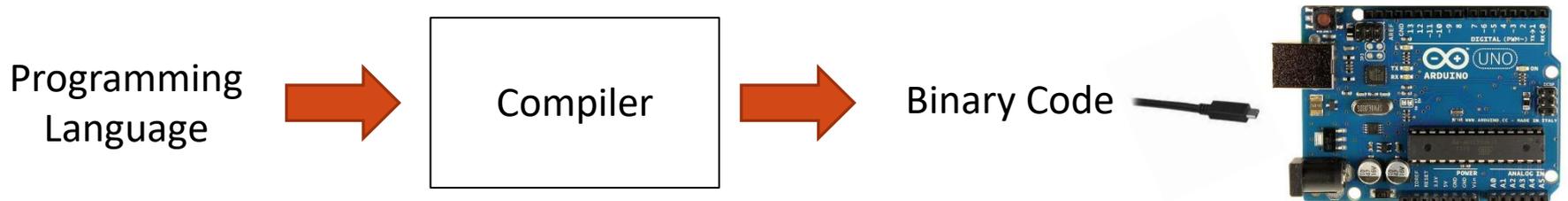
noun

1. the process of writing computer programs.
"programming languages"
2. the process of scheduling something, especially radio or television programmes.
"the programming of shows"

The meaning of programming

A programmer:

- Writes a **sequence of instructions** for the processor
- Uses a programming language «*understandable*» for humans
 - C/C++ for Arduino
- A compiler «transforms» this language in the binary representation

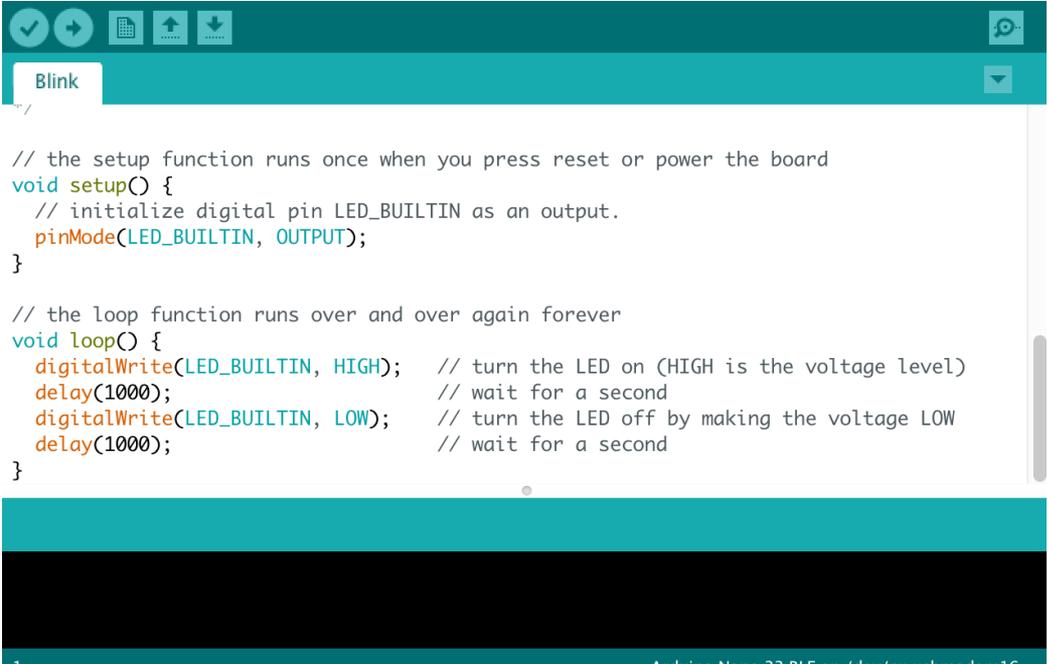


Programming with Arduino

There are two main **functions**:

- **Function setup**
 - *executed only once*

- **Function loop**
 - *continuously executed*



```
✓ ↻ 📄 ⬆ ⬇
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
}
```

Common functions

- `pinMode(<number_of_pin>, INPUT or OUTPUT);`
Set the pin in the input or output modality
- `val = digitalRead(<number_of_pin>);`
Reads the value from the specified digital pin and puts the value in val.
- `val = analogRead(<number_of_pin>);`
Reads the value from the specified analog pin and puts the value in val.
- `digitalWrite(<number_of_pin>, LOW/HIGH);`
Write a HIGH or a LOW value to a digital pin.

See the manual for available functions:

<https://www.arduino.cc/reference/en/>

Arduino functions

FUNCTIONS

For controlling the Arduino board and performing computations.

Digital operations



Digital I/O
digitalRead()
digitalWrite()
pinMode()

Analogic operations



Analog I/O
analogRead()
analogReference()
analogWrite()

Zero, Due & MKR Family
analogReadResolution()
analogWriteResolution()

Advanced I/O
noTone()
pulseIn()
pulseInLong()

Math
abs()
constrain()
map()
max()
min()
pow()
sq()
sqrt()

Trigonometry
cos()
sin()
tan()

Characters
isAlpha()
isAlphaNumeric()
isAscii()

Mathematical operations



Random Numbers
random()
randomSeed()

Bits and Bytes
bit()
bitClear()
bitRead()
bitSet()
bitWrite()
highByte()
lowByte()

External Interrupts
attachInterrupt()
detachInterrupt()

Interrupts

Arduino variables

In addition to functions, we can declare variables.

```
<type> <variable_name> = <value>
```

For example:

```
int buttonPin = 3;
int valueFromSensor;
valueFromSensor = digitalRead(buttonPin);
```

Remember to add a ; at the end of each instruction and variable definition.

Arduino data types

int

Integers are your primary data-type for number storage.

On the Arduino Uno an int stores a 16 bit value. This yields a range of -32,768 to 32,767 (minimum value of -2^{15} and a maximum value of $2^{15}- 1$).

Int's store **negative numbers** with the **2's complement technique**.

The Arduino takes care of dealing with negative numbers for you, so that arithmetic operations work transparently in the expected manner.

Arduino data types

char

A data type used to store a **character value**.

Character literals are written in single quotes, like this: 'A' (for multiple characters - strings - use double quotes: "ABC").

Characters are stores as numbers however through the **ASCII chart**.

The size of the char datatype is at least 8 bits.

Arduino data types

array

An array is a collection of variables that are accessed with an index number.

All of the methods below are valid ways to create (declare) an array.

- `int myInts[6];`
- `int myPins[] = {2, 4, 8, 3, 6};`
- `int mySensVals[6] = {2, 4, -8, 3, 2};`
- `char message[6] = "hello";`

Arduino data types

Arrays are zero indexed, that is, referring to the array initialization above, the first element of the array is at index 0, hence

```
mySensVals[0] == 2, mySensVals[1] == 4
```

It also means that in an array with ten elements, index nine is the last element. Hence:

```
int myArray[10]={9, 3, 2, 4, 3, 2, 7, 8, 9, 11};  
  
// myArray[9]    contains 11  
  
// myArray[10]   is invalid and contains random  
information (other memory address)
```

Arduino variables

VARIABLES

Arduino data types and constants.

Constants

HIGH | LOW

INPUT | OUTPUT | INPUT_PULLUP

LED_BUILTIN

true | false

Floating Point Constants

Integer Constants

Conversion

(unsigned int)

(unsigned long)

byte()

char()

float()

int()

long()

word()

Data Types

array

bool

boolean

byte

char

double

float

int

long

short

size_t

string

String()

unsigned char

unsigned int

unsigned long

void

Variable Scope & Qualifiers

const

scope

static

volatile

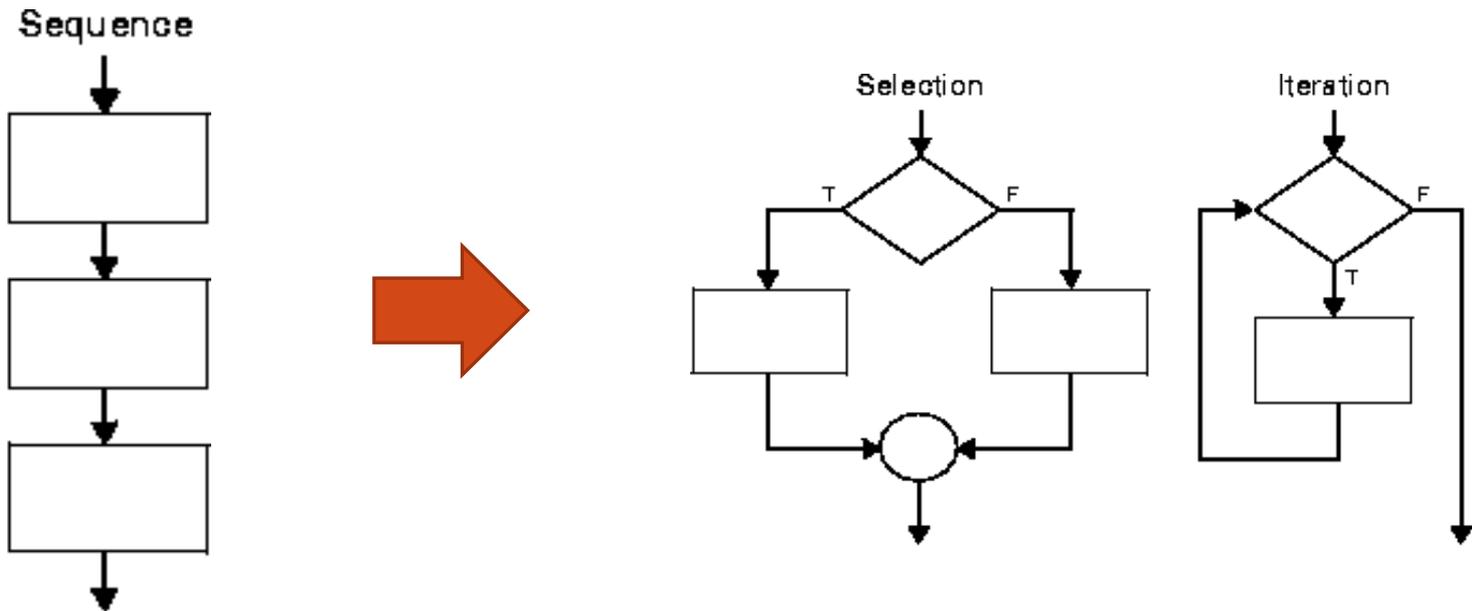
Utilities

PROGMEM

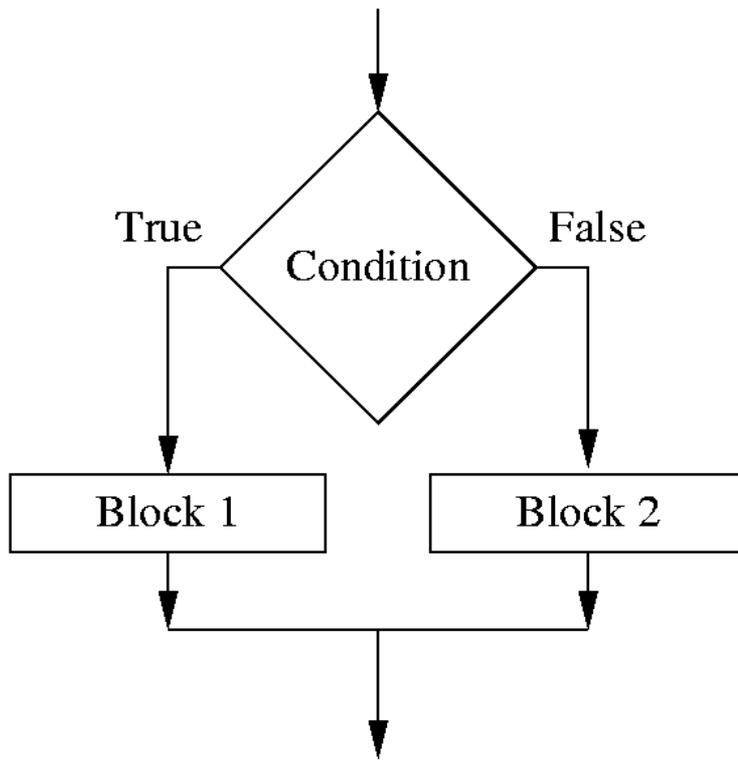
sizeof()

Arduino structures

We can change the flow of a sequence of instructions with control structures.



IF statement



```
if (condition) {  
    //statement(s)  
}  
else {  
}
```

Conditions

Comparison operators:

- `x == y` (x is equal to y)
- `x != y` (x is not equal to y)
- `x < y` (x is less than y)
- `x > y` (x is greater than y)
- `x <= y` (x is less than or equal to y)
- `x >= y` (x is greater than or equal to y)

Example

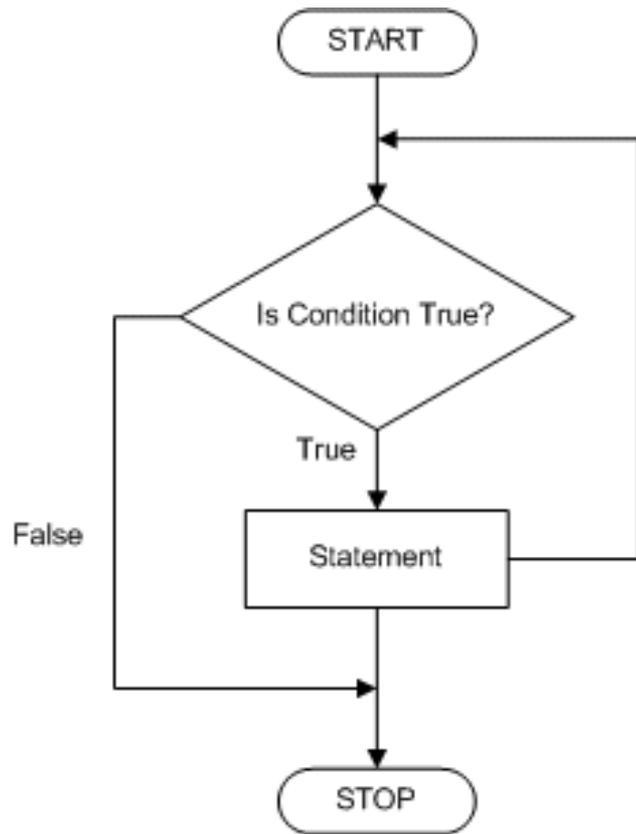
```
if (x > 120)
digitalWrite(LEDpin, HIGH);
```

```
if (x > 120)
digitalWrite(LEDpin, HIGH);
```

```
if (x > 120)
{digitalWrite(LEDpin, HIGH);}
```

```
if (x > 120) {
    digitalWrite(LEDpin1, HIGH);
    digitalWrite(LEDpin2, HIGH);
}
// all are correct
```

For statement



```
for (initialization; condition; increment) {  
    // statement(s);  
}
```

Check Again

For statement

```
for (initialization; condition; increment) {  
    // statement(s);  
}
```

Initialization: happens first and exactly once.

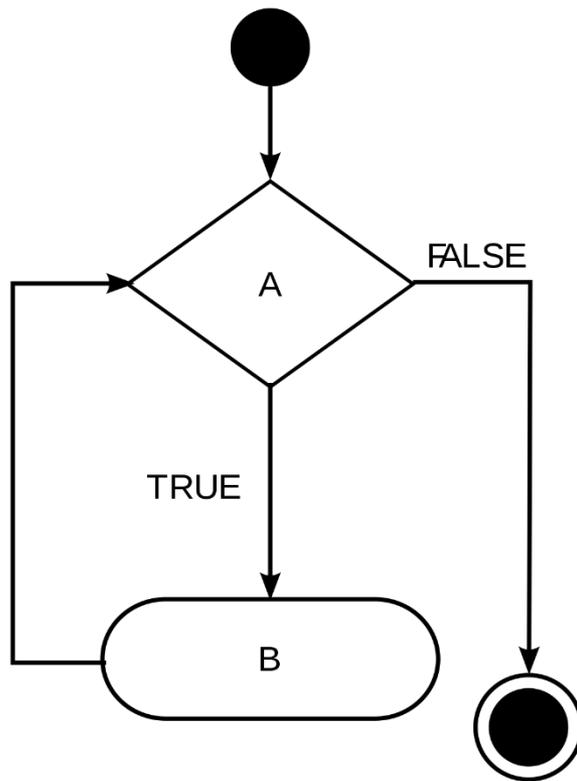
Condition: each time through the loop, condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.

Increment: executed each time through the loop when condition is true.

Example

```
for (int i = 0; i <= 255; i++) {  
    analogWrite(PWMPin, i);  
    delay(10);  
}
```

While statement



```
while (condition)
{
    // statement(s)
}
```

Example

```
int var = 0;
while (var < 200) {
    // do something repetitive 200 times
    var = var + 1; // same of v++;
}
```

Arduino structures

STRUCTURE

The elements of Arduino (C++) code.

Sketch

loop()
setup()

Control Structure

break
continue
do...while
else
for
goto
if
return
switch...case
while

Further Syntax

Arithmetic Operators

% (remainder)
* (multiplication)
+ (addition)
- (subtraction)
/ (division)
= (assignment operator)

Comparison Operators

!= (not equal to)
< (less than)
<= (less than or equal to)
== (equal to)
> (greater than)
>= (greater than or equal to)

Boolean Operators

Pointer Access Operators

& (reference operator)
* (dereference operator)

Bitwise Operators

& (bitwise and)
<< (bitshift left)
>> (bitshift right)
^ (bitwise xor)
| (bitwise or)
~ (bitwise not)

Compound Operators

%= (compound remainder)
&= (compound bitwise and)
*= (compound multiplication)
++ (increment)

Programming with blocks

In Tinkercad you can program also with logical blocks.

Click on «Code», then choose the version «blocks» (text and blocks + text are also available).

