

Introduction to Arduino use

What is Arduino? (1/4)

Arduino's Word means
3 things

What is Arduino? (2/4)

- Physical Object



What is Arduino? (3/4)

- **Integrated Development Environment**



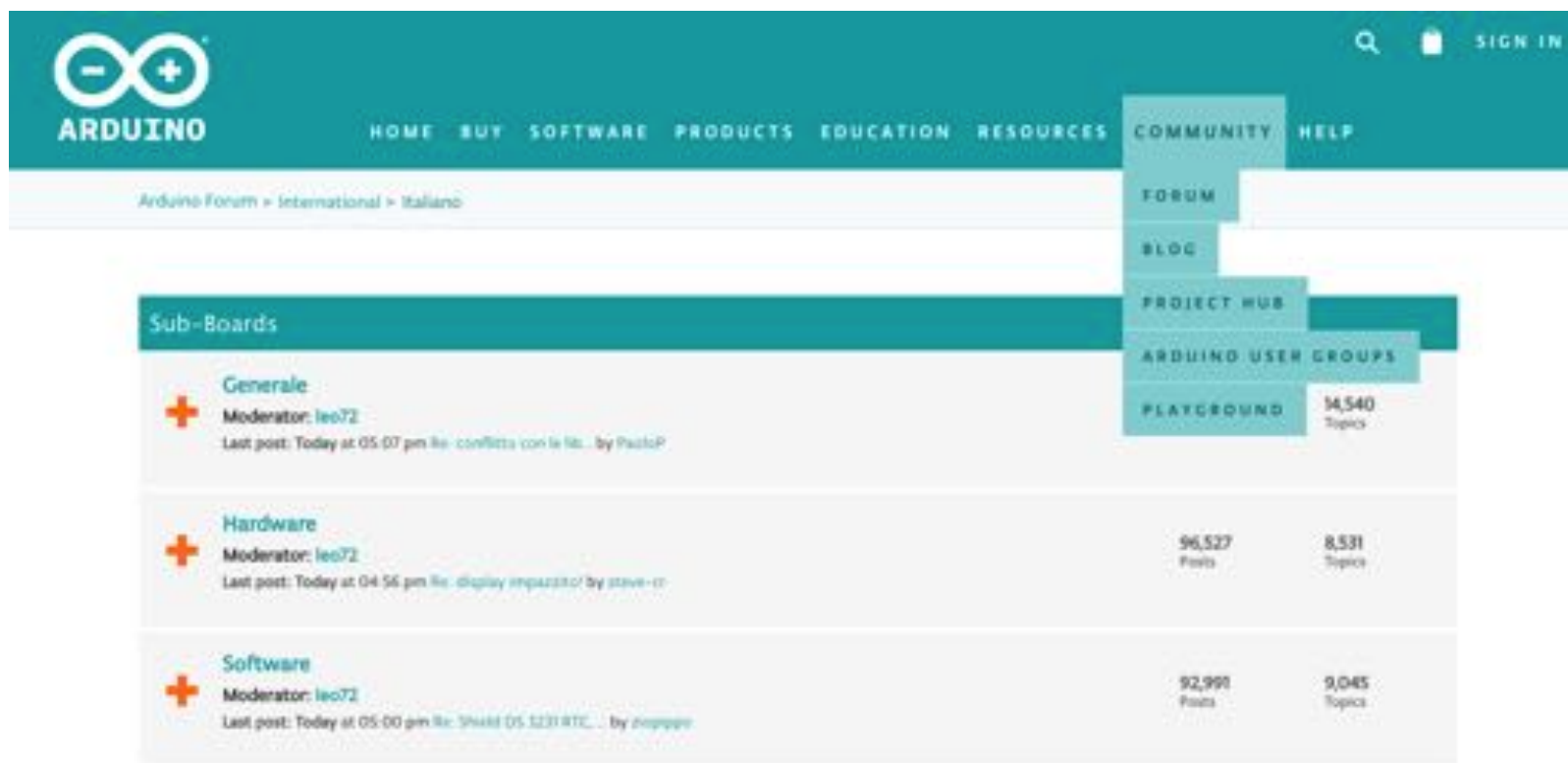
```
sketch_apr10a | Arduino 1.8.5
sketch_apr10a
void setup() {
  // put your setup code here, to run once:
}

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

Arduino Dev (Programming Port to /dev/tty.usbmodem1411)

What is Arduino? (4/4)

- A community and a development philosophy



The screenshot shows the Arduino website's community section. At the top, there is a teal navigation bar with the Arduino logo on the left and a search icon, a shopping cart icon, and a 'SIGN IN' link on the right. Below the navigation bar, the breadcrumb path reads 'Arduino Forum > International > Italiano'. The main content area is divided into two columns. The left column is titled 'Sub-Boards' and lists three categories: 'Generale', 'Hardware', and 'Software'. Each category includes a moderator name (leo72) and a 'Last post' entry. The right column is titled 'COMMUNITY' and lists 'FORUM', 'BLOG', 'PROJECT HUB', 'ARDUINO USER GROUPS', and 'PLAYGROUND'. The 'PLAYGROUND' section shows '14,540 Topics'. The 'Hardware' and 'Software' sections also show 'Posts' and 'Topics' counts.

Sub-Board	Moderator	Last post	Posts	Topics
Generale	leo72	Today at 05:07 pm Re: conflicts con le lib... by PaoloP		
Hardware	leo72	Today at 04:56 pm Re: display incompatib... by stave-cr	96,527	8,531
Software	leo72	Today at 05:00 pm Re: Shield DS 1231 RTC... by ziopepp	92,990	9,045

Arduino Hardware (1/5)

- Entry Level



Arduino Hardware (2/5)

- Enhanced Features



Arduino Hardware (3/5)

- In this course we will use Arduino DUE



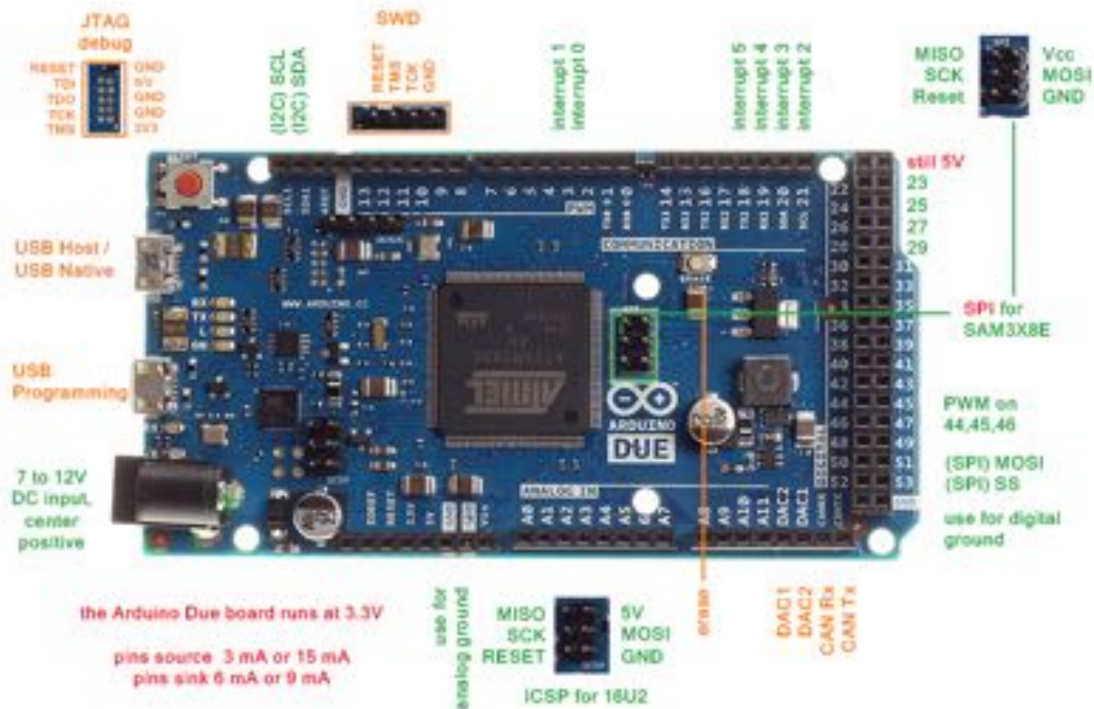
Arduino Hardware (4/5)

- **Technical Specification**

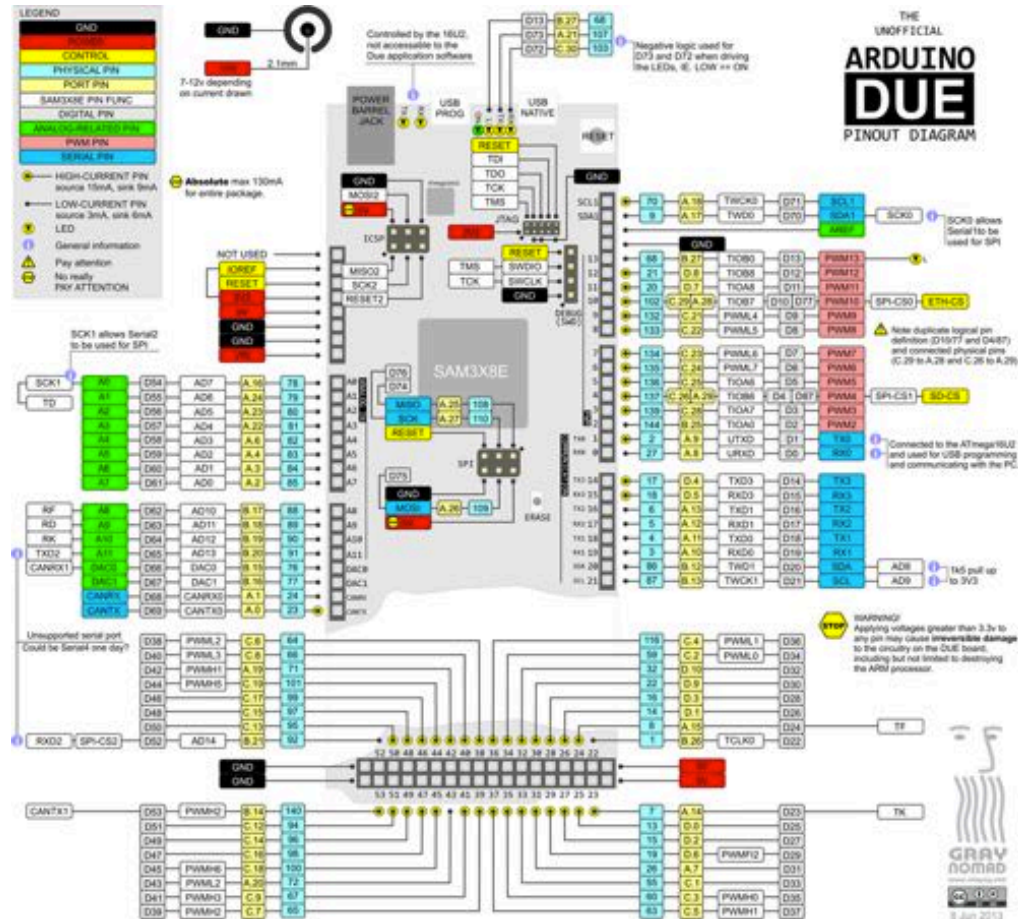
Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

Arduino Hardware (5/5)

- Fundamental Parts



Arduino DUE Pin Mapping



Terminology

- **sketch**

- Is the program that you write and run on the Arduino board

- **pin**

- Input and output connectors

- **digital**

- It means that it can only take two values: HIGH or LOW, in another way ON/OFF or 0/1

- **analog**

- When the values are continuous (infinite)
-

The Software (1/2)

- Similar to a text editor;
- You can write, visualize and verify the syntax;
- You can upload the sketch on your board.



The screenshot shows the Arduino IDE interface with the 'Blink' sketch loaded. The title bar reads 'Blink | Arduino 1.8.5'. The code editor contains the following text:

```
Blink
Created: 2014-05-08 10:00:00 AM
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
}
```

At the bottom right of the IDE, the board and port are identified as 'Arduino Due (Programming Port) 3u /dev/cu.usbmodem1411'.

The Software (2/2)

CLARIFICATION

During these lessons to indicate the software development environment, we will use:

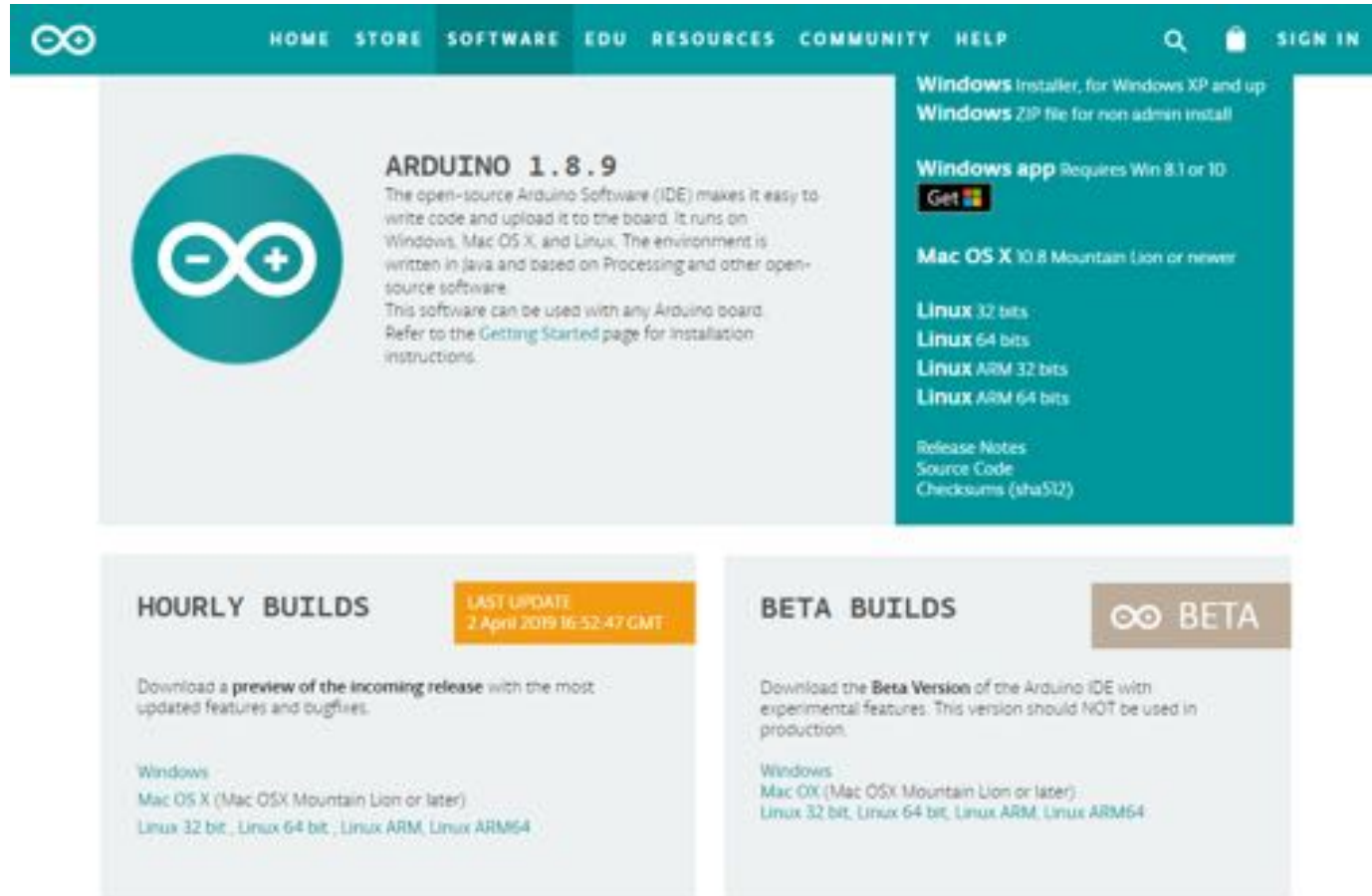
software Arduino

Or with the same means:

IDE

IDE means: *Integrated Development Enviroment*,
in italiano: *ambiente di sviluppo integrato per la
realizzazione di programmi.*

Software Installation



The screenshot shows the Arduino IDE download page. The navigation bar includes links for HOME, STORE, SOFTWARE, EDU, RESOURCES, COMMUNITY, and HELP, along with search, cart, and sign-in icons. The main content area features the Arduino logo and the title 'ARDUINO 1.8.9'. Below the title, there is a description of the IDE and its compatibility with various operating systems. To the right, there are links for downloading the Windows installer and ZIP file, the Windows app, and Mac OS X installer. Further down, there are links for Linux 32-bit, Linux 64-bit, Linux ARM 32-bit, and Linux ARM 64-bit. At the bottom, there are sections for 'HOURLY BUILDS' and 'BETA BUILDS'.

HOME STORE SOFTWARE EDU RESOURCES COMMUNITY HELP

ARDUINO 1.8.9

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

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

HOURLY BUILDS LAST UPDATE 2 April 2019 16:52:47 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

BETA BUILDS BETA

Download the Beta Version of the Arduino IDE with experimental features. This version should NOT be used in production.

Windows
Mac OS X (Mac OS X Mountain Lion or later)
Linux 32 bit, Linux 64 bit, Linux ARM, Linux ARM64

Communication with Arduino

- **Launch the Arduino IDE (double click)**



Arduino Program Development

- **Based on C++ without 80% of the instructions.**
- **A handful of new commands.**
- **Programs are called 'sketches'.**
- **Sketches need two functions:**
 - `void setup()`
 - `void loop()`
- **`setup()` runs first and once.**
- **`loop()` runs over and over, until power is lost or a new sketch is loaded.**

Parts of the IDE main screen



← Name of current sketch

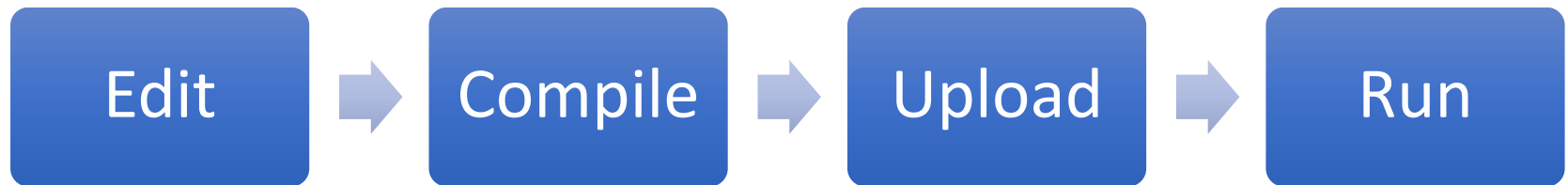
← Main menus

← Action buttons/icons

- ✓ Verify (AKA compile)
- ▶ Upload (send to Arduino)
- 📄 Start a new sketch
- 📄 Open a sketch (from a file)
- 📄 Save current sketch (to a file)
- 📄 Open Serial Monitor window

Programming

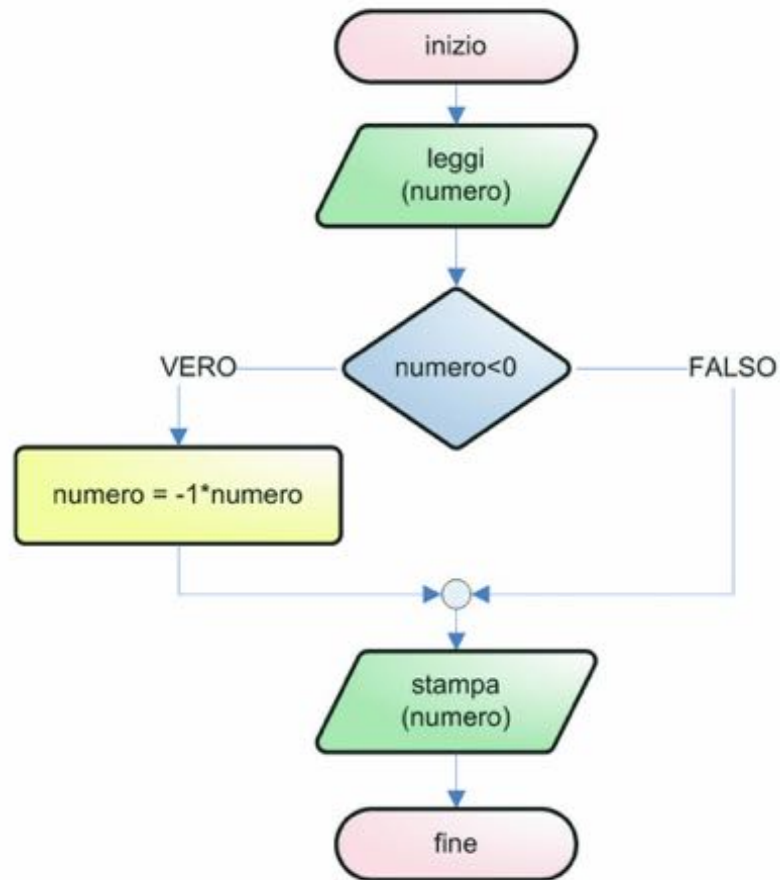
Development Cycle



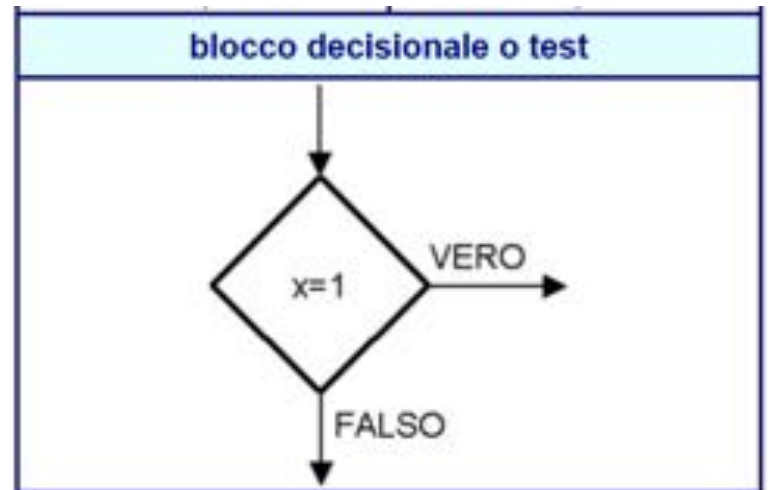
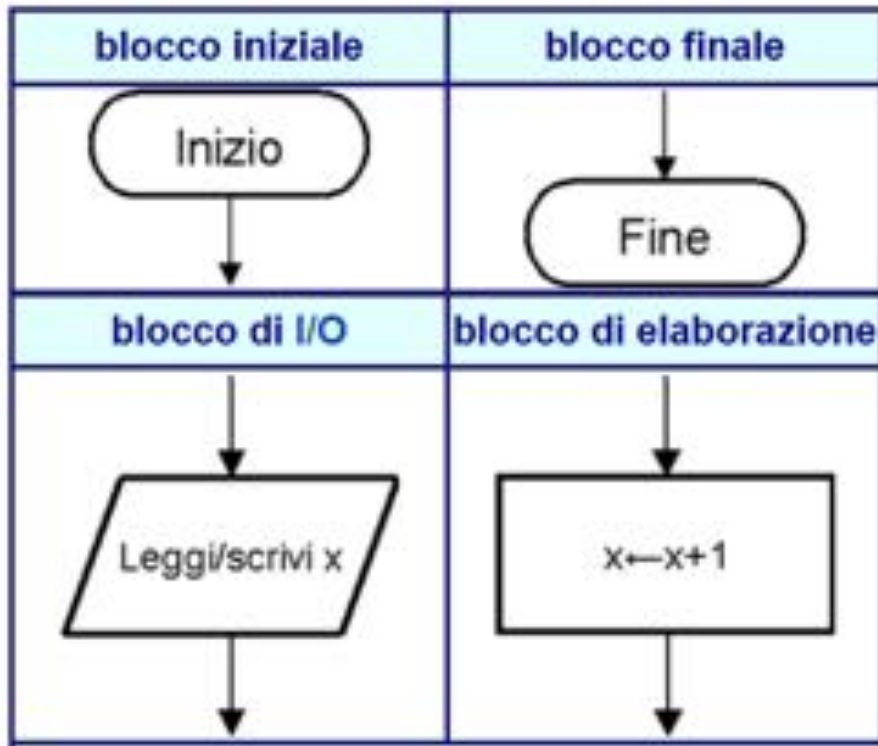
Compile: Compile means to translate the sketch into machine language, also known as object code

Run: Arduino sketch is executed as soon as terminates the step of uploading on the board

Flow diagram



Flow diagram



The structure of Arduino Sketch (1/2)



The screenshot shows the Arduino IDE interface with a window titled "BareMinimum | Arduino 1.0.5-r2". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for saving, opening, and other functions. The main text area displays the following code:

```
BareMinimum §  
  
void setup()  
{  
  // put your setup code here, to run once  
  // e.g. define variables; initialize pins; include libraries  
}  
  
void loop()  
{  
  // put your main code here, to run repeatedly  
  // e.g. read sensor, log data to SD card, pause 1 sec, repeat  
}
```

The structure of Arduino Sketch (2/2)

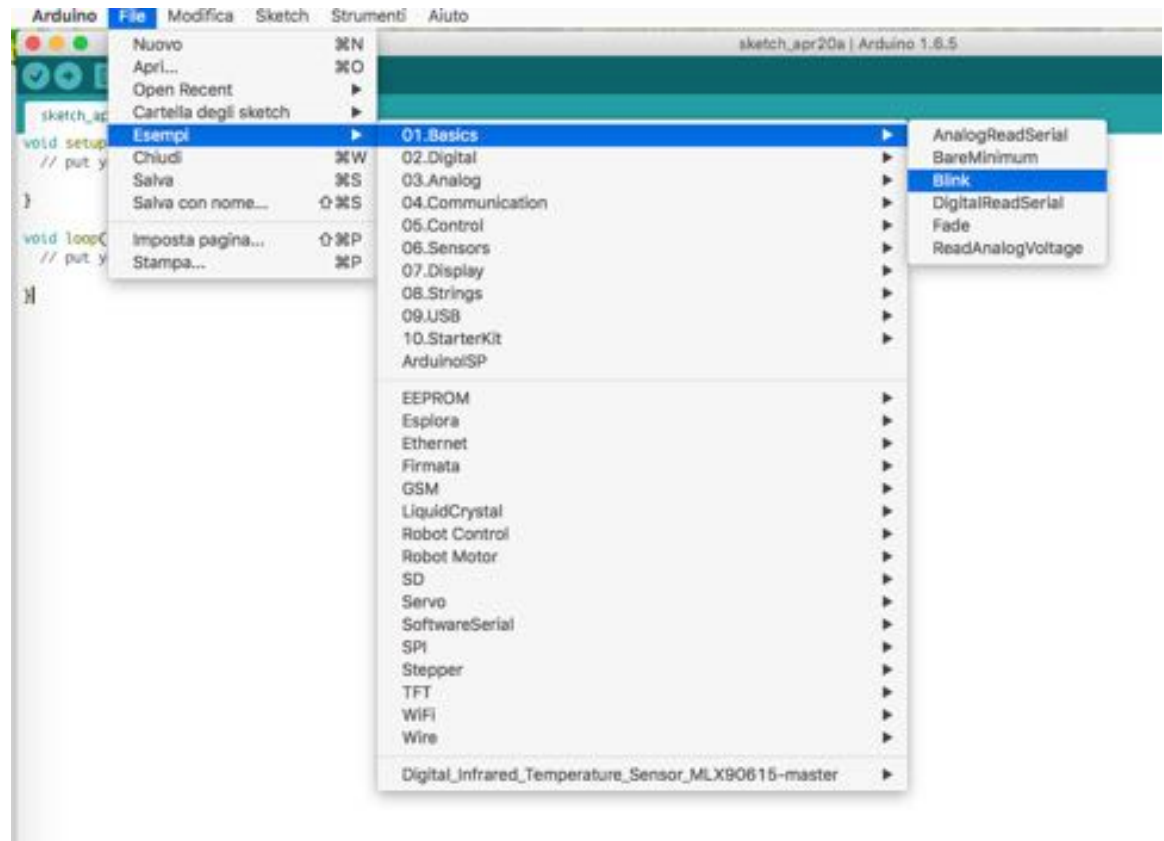
- The first one is “`setup()`”. Anything you put in this function will be executed by the Arduino just once when the program starts.
- The second one is “`loop()`”. Once the Arduino finishes with the code in the `setup()` function, it will move into `loop()`, and it will continue running it in a loop, again and again, until you reset it or cut off the power.

Arduino specific function

- **pinMode(*pin*, *mode*)**
 - Configures a digital pin to read (input) or write (output) a digital value
- **digitalWrite(*pin*, *value*)**
 - Writes the digital value (HIGH or LOW) to a pin set for output
- **digitalRead(*pin*)**
 - Reads a digital value (HIGH or LOW) on a pin set for input
- **analog versions of above**
 - **analogRead's** range is 0 to 1023 (for Arduino Uno)
 - The Due and the Zero have 12-bit ADC capabilities that can be accessed by changing the resolution to 12. This will return values from `analogRead()` between 0 and 4095.
- **serial commands**
 - `print`, `println`, `write`, `delay`
- **Other example**
 - <https://www.arduino.cc/en/Reference/HomePage>

Arduino Sketch Example

- Numerous sample sketches are included in the compiler
- Located under File, Examples

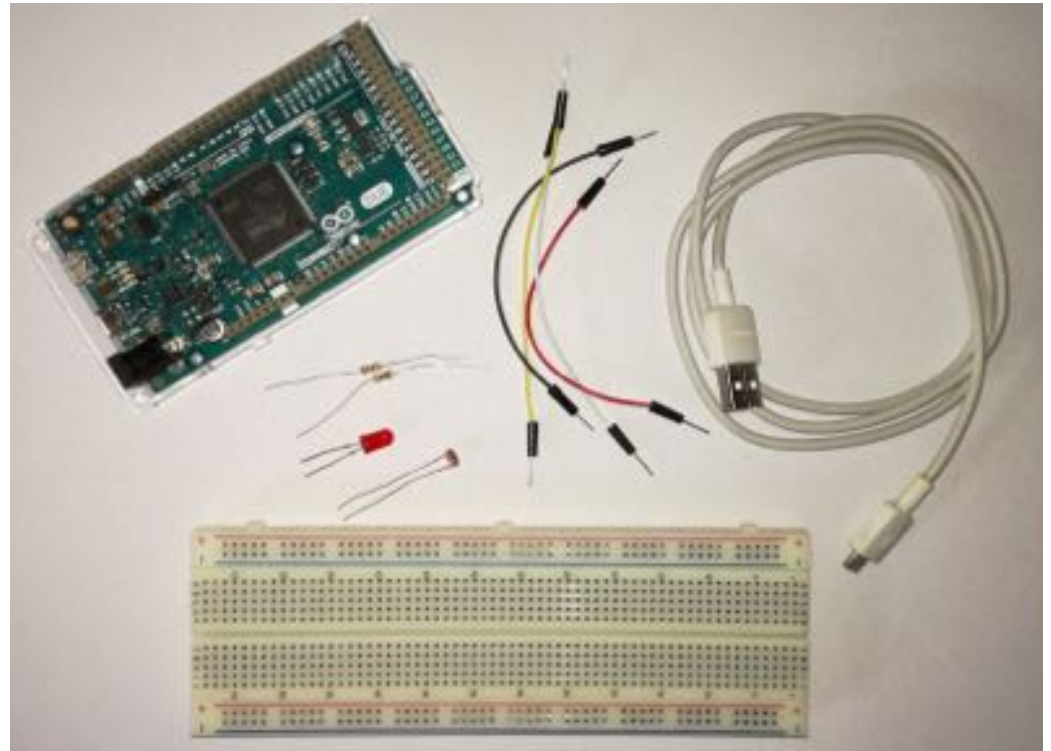


First exercise:
turn on a LED if there is
no light

Photoresistor with LED

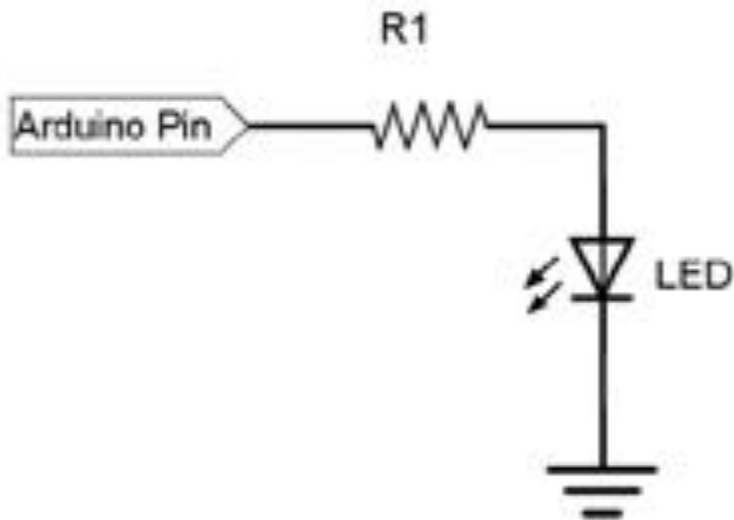
What are we going to use?

- Arduino DUE
- Breadboard
- USB Cable
- LED
- Photoresistor
- 10k Ω resistance
- 220 Ω resistance
- cables

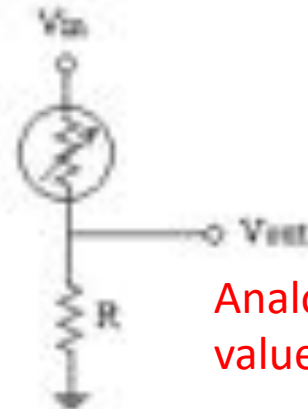
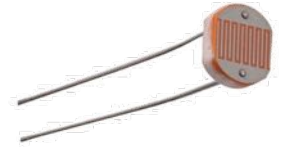


How to connect components

LED

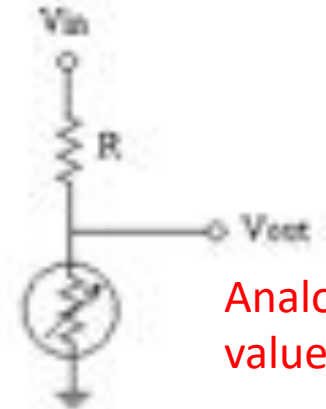


Photoresistor



Analog
value

This circuit gives an output voltage that increases with the light level.

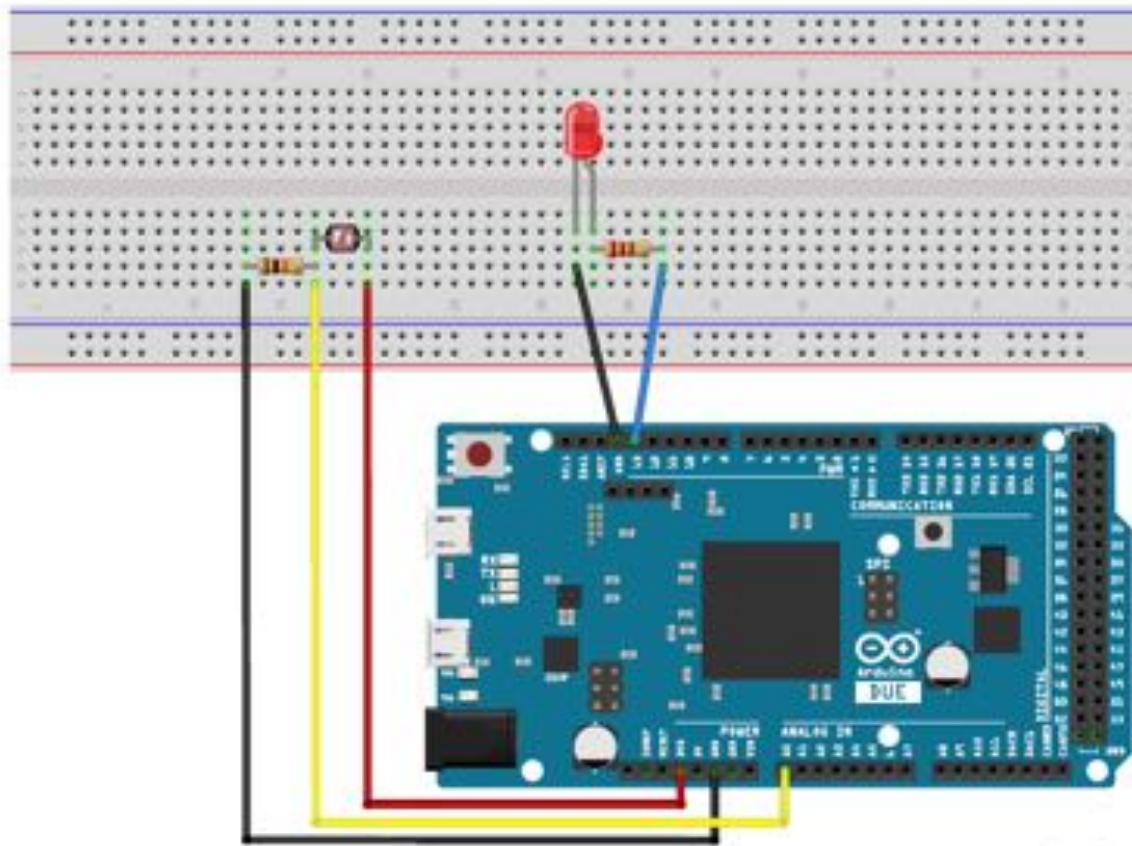


Analog
value

This circuit gives an output voltage that decreases with the light level.

[Find a datasheet](#)

How to connect components



Arduino sketch

```
// define variables
```

```
int led=13; // led connected to digital pin 13
int photoresistor;
```

```
void setup() {
```

```
  pinMode(led, OUTPUT); // initialize digital pin as an output
  Serial.begin(9600);
```

All variables must be declared before they can be used. Declaring a variable means:

- define the type of value that can assume: int, long, float, etc ...
 - assign a name
 - and optionally assign an initial value.
-

Arduino sketch

```
// define variables
```

```
int led=13; // led connected to digital pin 13
int photoresistor;
```

```
void setup() {
```

```
  pinMode(led,OUTPUT); // initialize digital pin as an output
  Serial.begin(9600);
```

```
}
```

Setup routine runs only ONCE when press reset

;

Identifies where the instruction ends

{

...

}

Identifies a block of instructions

Serial.begin();

Sets the data rate in bits per second (baud) for serial data transmission. So basically we are going to transfer 9600 bits per second to the computer.

Arduino sketch

```
void loop() {  
  
  photoresistor=analogRead(A0); // read the value given by photoresistor (analog pin A0)  
  Serial.println(photoresistor); // prints on the serial monitor analog values from the photoresistor  
  if (photoresistor<=400){ // threshold value below which I want the led to light up  
    digitalWrite(led,HIGH); // turn on led  
  }  
  else{  
    digitalWrite(led,LOW); // turn off led  
  }  
  delay(250); // repeat reading every 250 ms  
}
```

Loop routine runs over and over

If/else structure

Syntax

```
if (condition1) {  
  // do Thing A  
}  
else if (condition2) {  
  // do Thing B  
}  
else {  
  // do Thing C  
}
```

Arduino sketch

