

# PROGRAMMING WITH ARDUINO

# Software Arduino

## The Arduino Programming Environment (IDE)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

# Software - Download

## Download the Arduino IDE



### ARDUINO 1.8.2

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

**Windows** ZIP file for non admin install

**Windows app**

**Mac OS X** 10.7 Lion or newer

**Linux** 32 bits

**Linux** 64 bits

**Linux** ARM

[Release Notes](#)

[Source Code](#)

[Checksums \(sha512\)](#)

### ARDUINO SOFTWARE HOURLY BUILDS

Download a preview of the incoming release with the most updated features and bugfixes.

[Windows](#)

[Mac OS X](#) (Mac OS X Lion or later)

[Linux 32 bit](#) , [Linux 64 bit](#) , [Linux ARM](#)

ARDUINO 1.0.6 / 1.5.x / 1.6.x

### PREVIOUS RELEASES

Download the [previous version of the current release](#), the classic [Arduino 1.0.x](#), or the [Arduino 1.5.x Beta version](#).

All the [Arduino 00xx versions](#) are also available for download. The Arduino IDE can be used on Windows, Linux (both 32 and 64 bits), and Mac OS X.

# Software - Installation

- Windows

[arduino.cc/windows](https://arduino.cc/windows)

Installation for: Windows 7, Vista, e XP

- Mac OS X

[arduino.cc/mac](https://arduino.cc/mac)

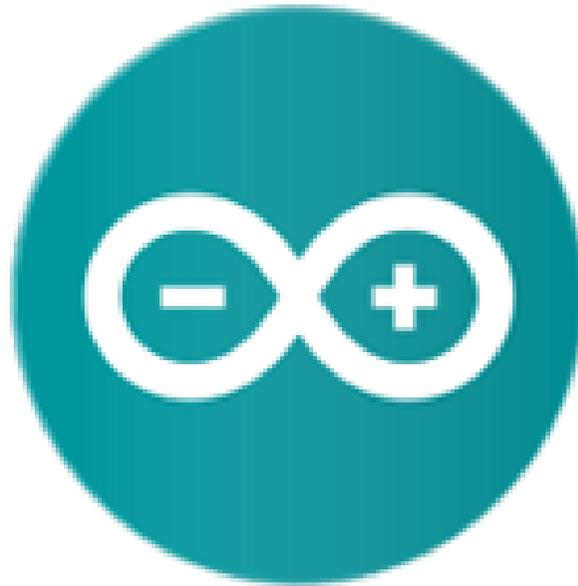
Installation for: OS X 10.7 or newer

- Linux

[arduino.cc/linux](https://arduino.cc/linux)

# Communication with Arduino

- Launch the Arduino IDE (double click)

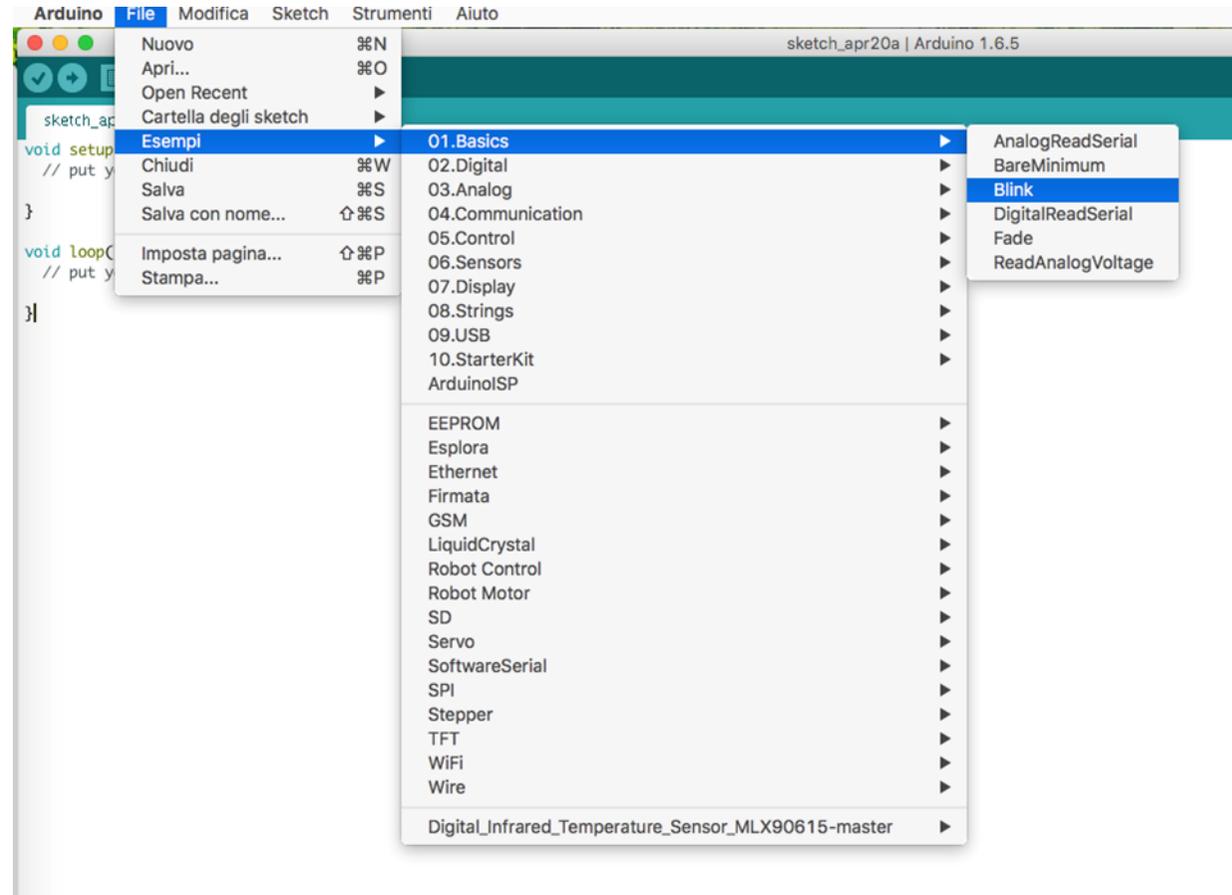


# Arduino Program Development

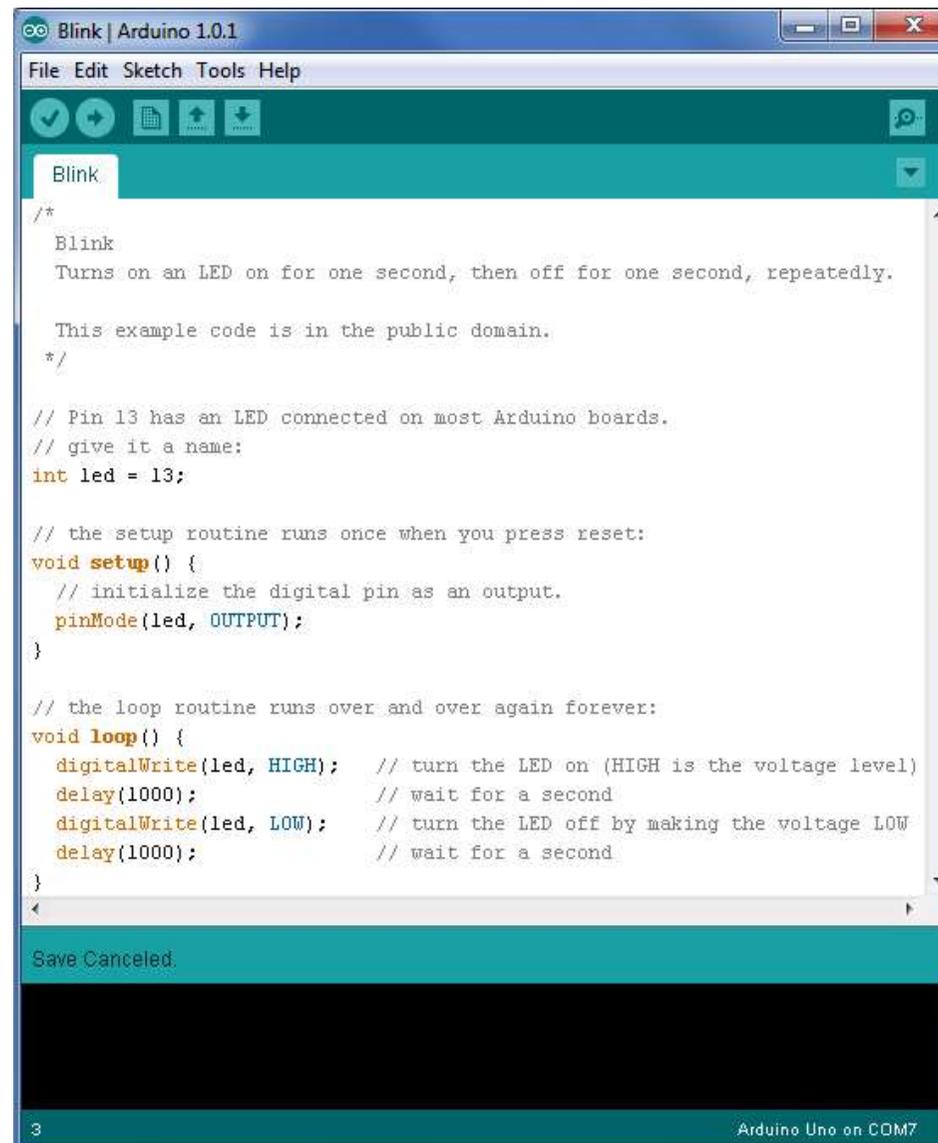
- Based on C++ without 80% of the instructions.
- A handful of new commands.
- Programs are called 'sketches'.
- Sketches need two functions:
  - void setup( )
  - void loop( )
- setup( ) runs first and once.
- loop( ) runs over and over, until power is lost or a new sketch is loaded.

- Open the sketch

- Numerous sample sketches are included in the compiler
- Located under File, Examples
- Once a sketch is written, it is uploaded by clicking on File, Upload, or by pressing <Ctrl> U



- Open the Blink sketch

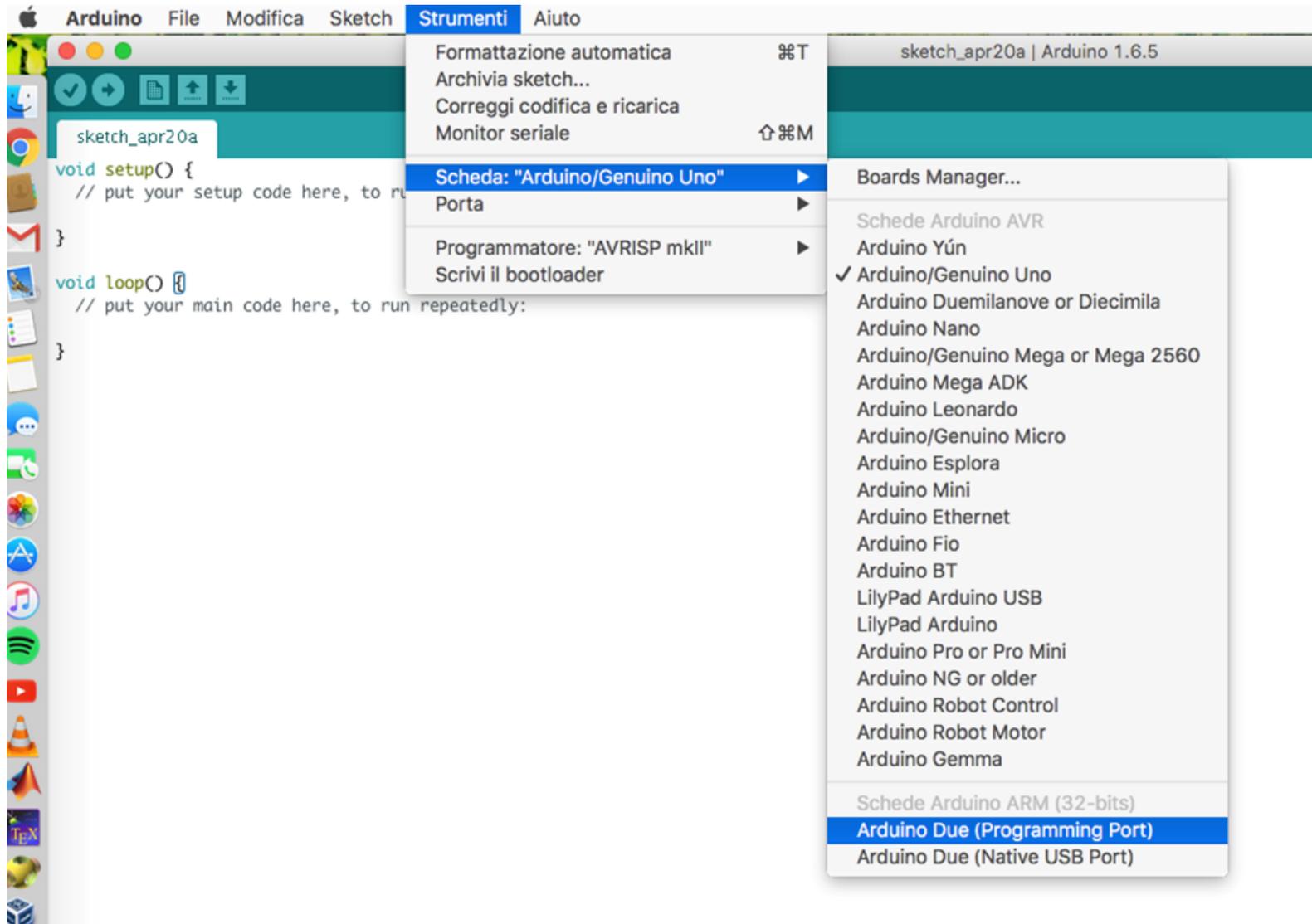


The screenshot shows the Arduino IDE interface with the Blink sketch open. The window title is "Blink | Arduino 1.0.1". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for checking, saving, opening, and uploading. The sketch name "Blink" is displayed in a teal bar. The code editor contains the following text:

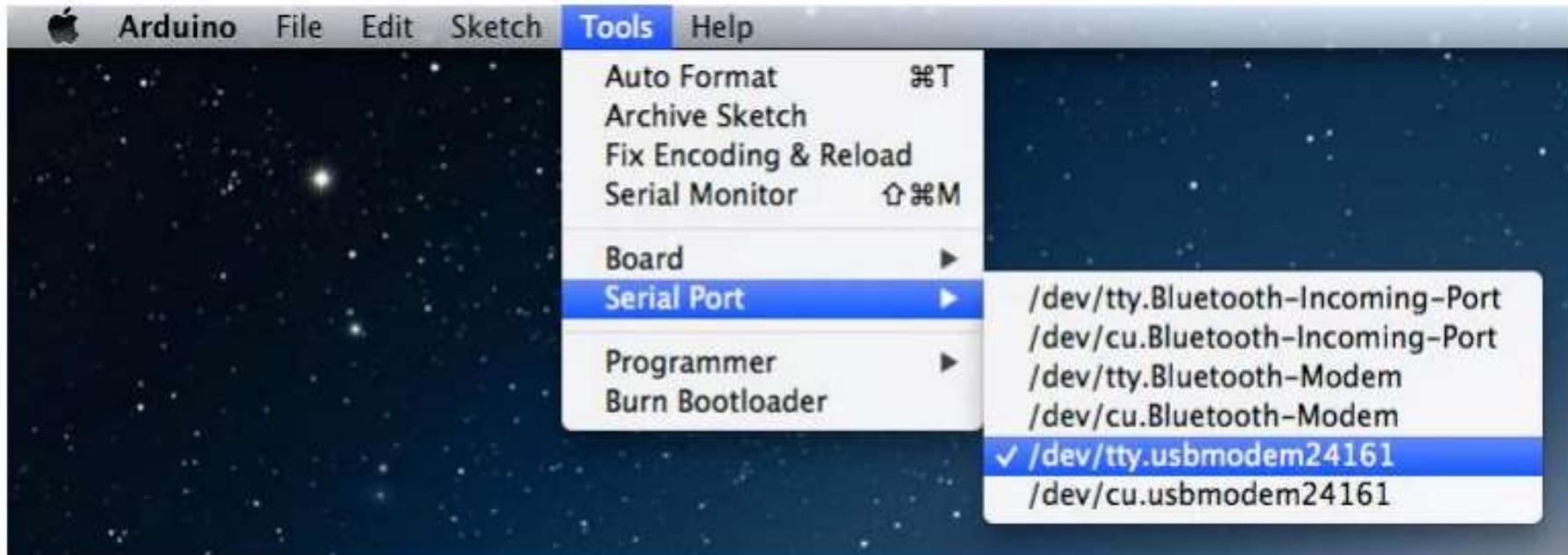
```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
  */  
  
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize the digital pin as an output.  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```

At the bottom of the IDE, a teal status bar displays "Save Canceled." and "3". The bottom right corner of the window shows "Arduino Uno on COM7".

- Select the Board

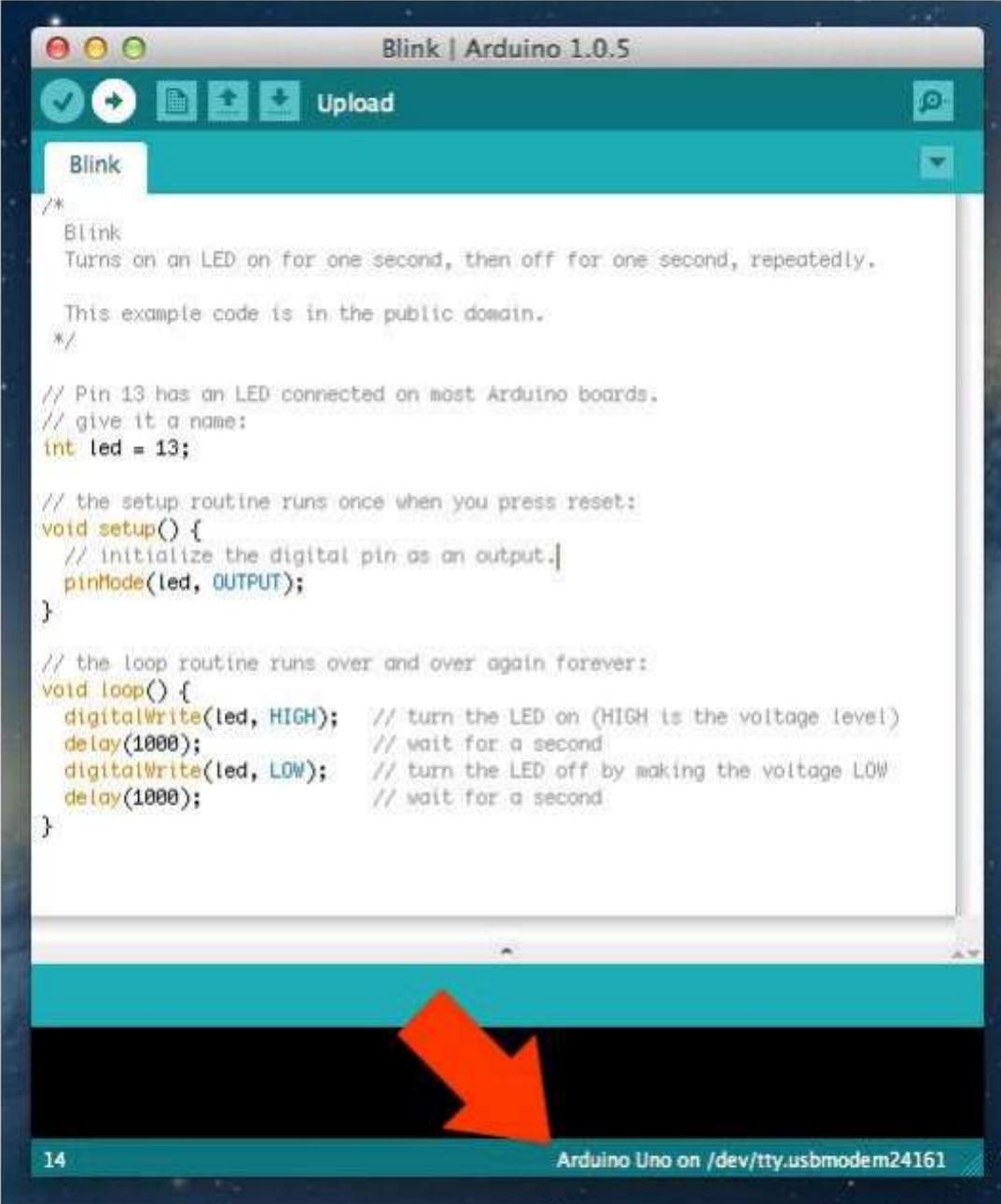


- Select the Serial Port



- **Mac:**  
You can indifferently choose between  
`/dev/tty.usbmodemXXXXX` or `/dev/cu.usbmodemXXXXX`
- **Windows:**  
There are one or more COM ports:  
choose the one with the higher number if it does not work try with the  
other proposals.

- The connection to the serial port is reported in the code window in bottom right



The screenshot shows the Arduino IDE interface. The title bar reads "Blink | Arduino 1.0.5". The top toolbar includes a checkmark, a refresh icon, a document icon, an upload icon, and a "Upload" button. Below the toolbar is a tab labeled "Blink". The main code window contains the following code:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
  */

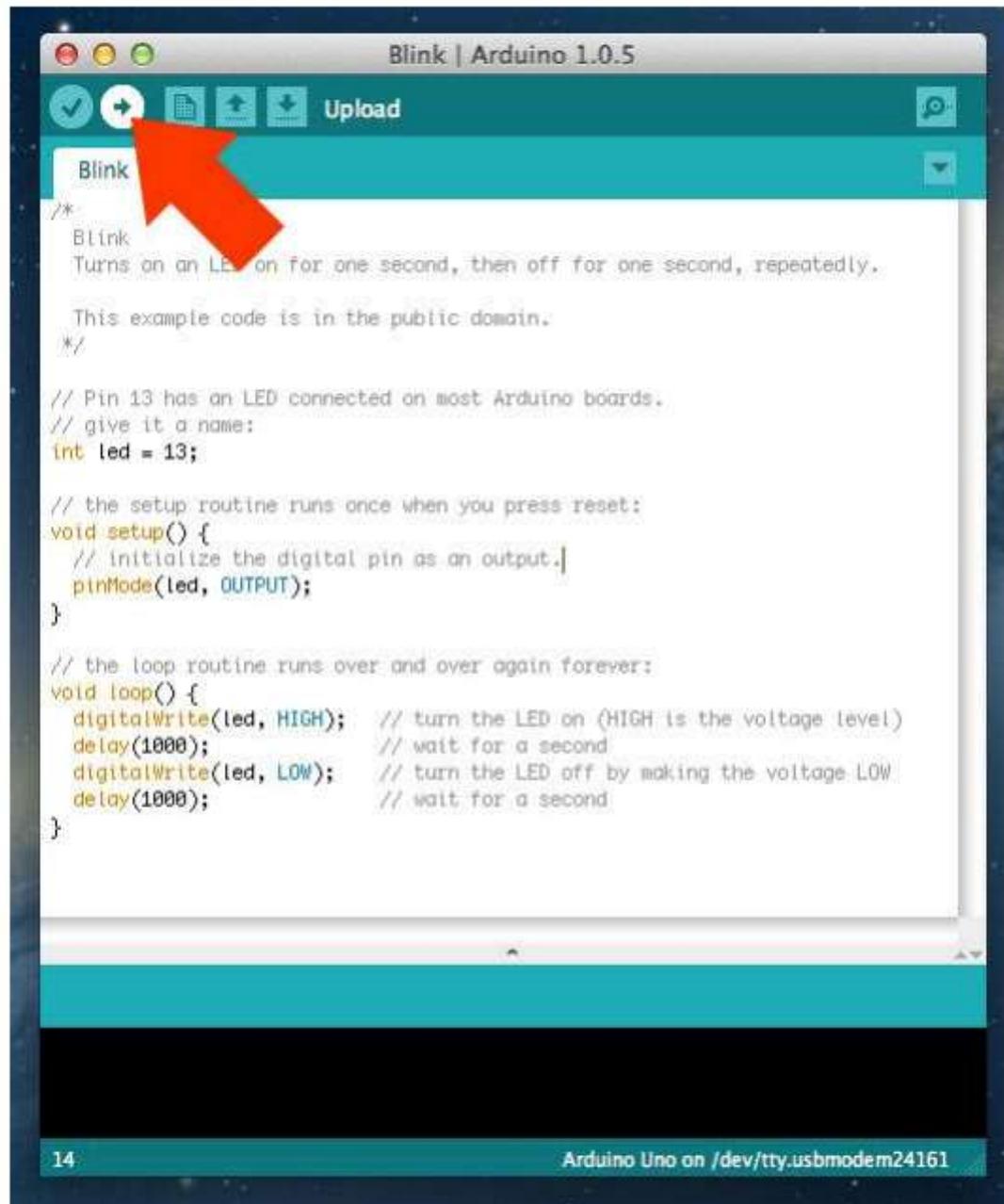
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

At the bottom of the IDE, a status bar shows "14" on the left and "Arduino Uno on /dev/tty.usbmodem24161" on the right. A large red arrow points from the code window down to the serial port connection text in the status bar.

- Loading the Blink sketch on the board through the Upload button



- It will take a few seconds, during this operation you will see that the LEDs RX and TX (receive and transmit) flash.

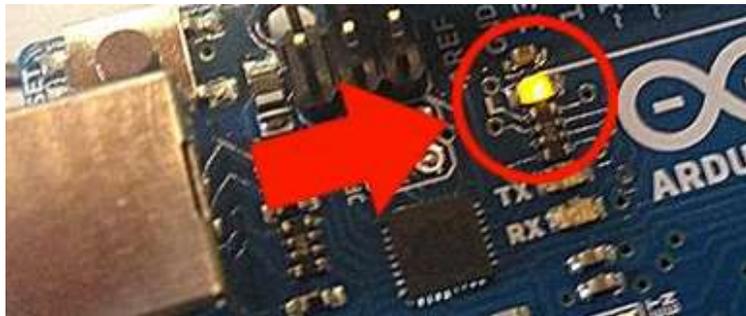


```
Blink | Arduino 1.0.5  
Blink  
Turns on an LED on for one second, then off for one second, repeatedly.  
  
This example code is in the public domain.  
*/  
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize the digital pin as an output.  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```

Compiling sketch...

3 Arduino Uno on /dev/tty.usbmodem24161

If everything will be succesfull you will be returned message "Done uploading." in the staus bar, and the LED L starts flashing

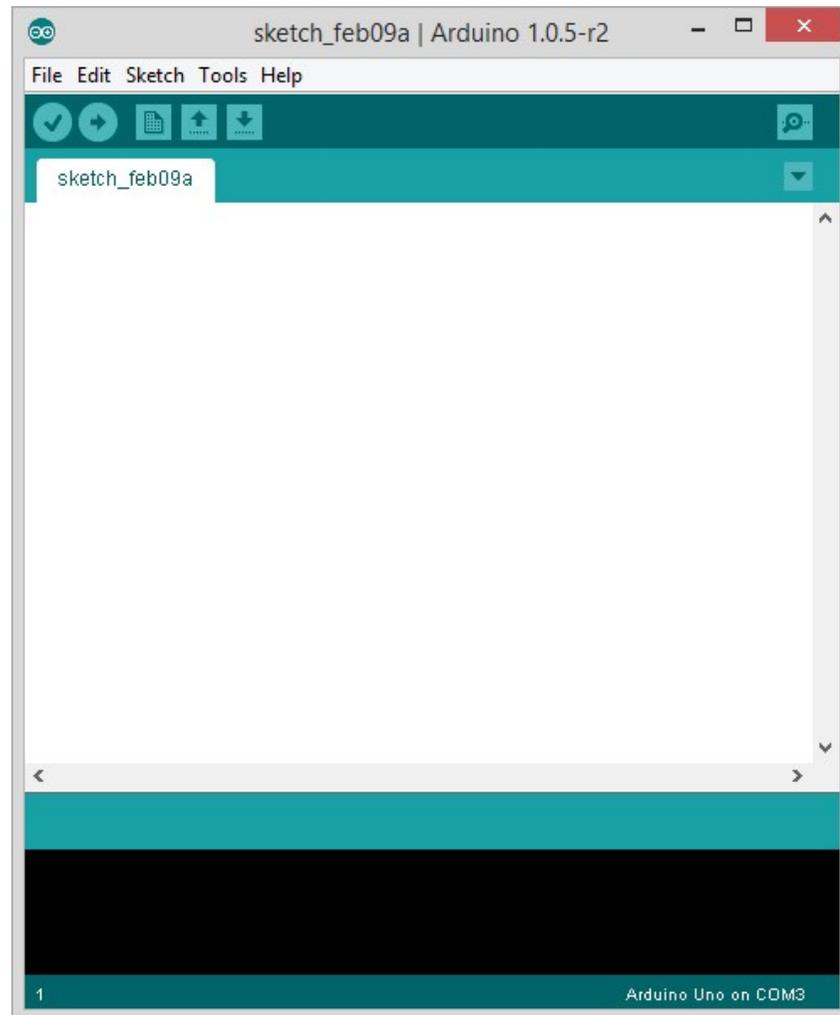
A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.0.5". The code editor shows the following C++ code:

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
  */  
  
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize the digital pin as an output.  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```

A red arrow points from the code to the status bar at the bottom. The status bar is highlighted with a red box and contains the text: "Done uploading." and "Binary sketch size: 1,084 bytes (of a 32,256 byte maximum)". At the very bottom, the status bar shows "23" and "Arduino Uno on /dev/tty.usbmodem24161".

Programming

# Parts of the IDE main screen



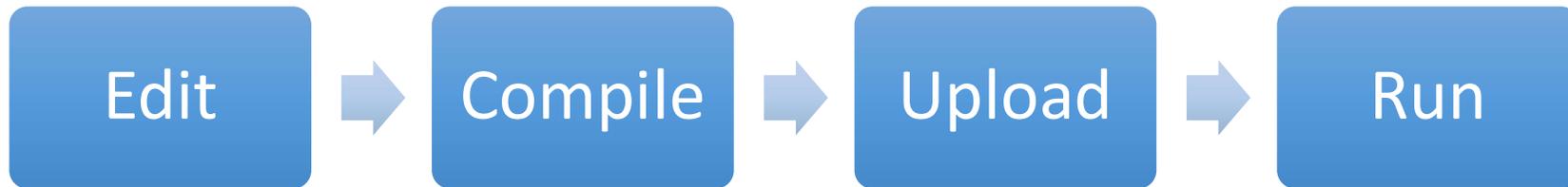
← Name of current sketch

← 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

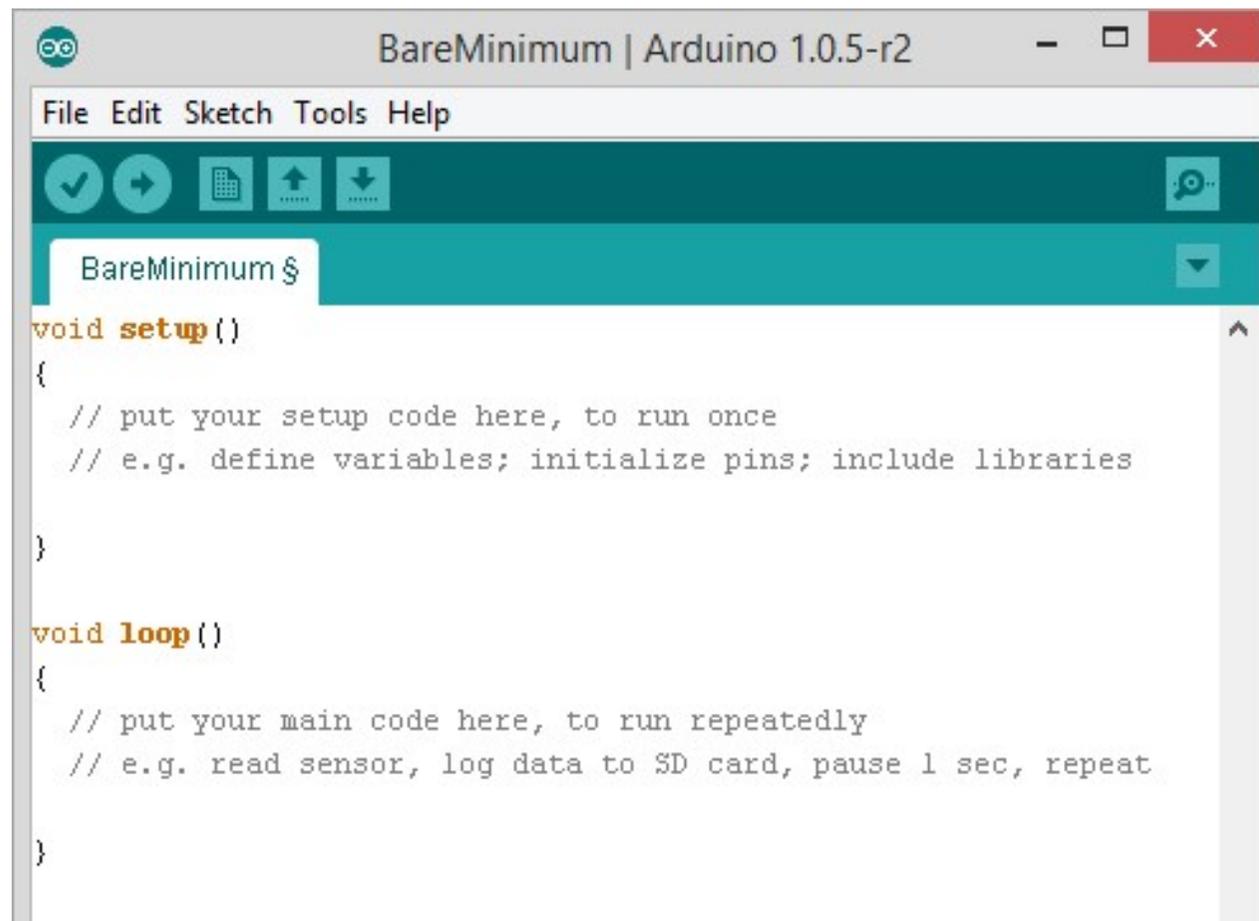
The development cycle is divided into 4 phases:



**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

# The structure of an Arduino Sketch

A screenshot of the Arduino IDE interface. The window title is "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 checkmark, right arrow, grid, up arrow, down arrow, and a search icon. A dropdown menu is open, showing "BareMinimum \$". The main text area contains the following code:

```
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 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.

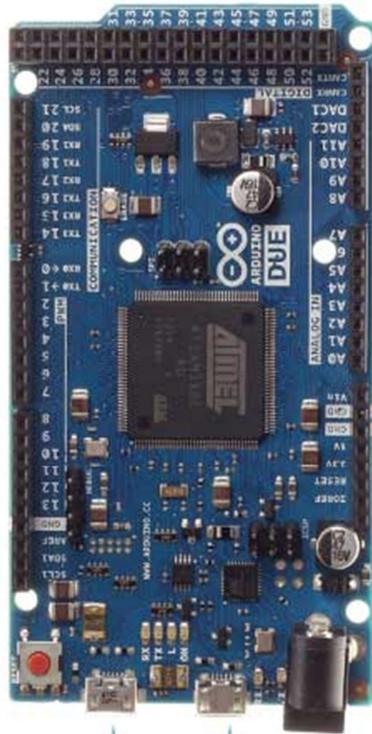
# Arduino Specific Functions

- **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>

First Program

Blinking LED

# Arduino DUE

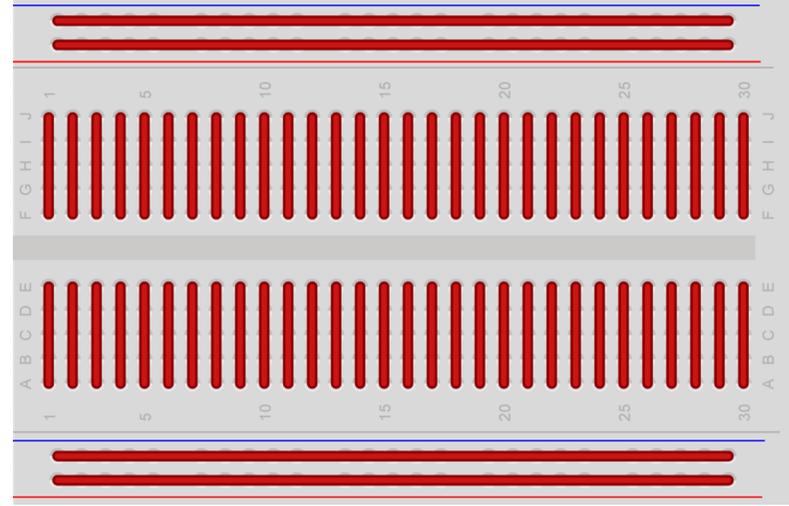
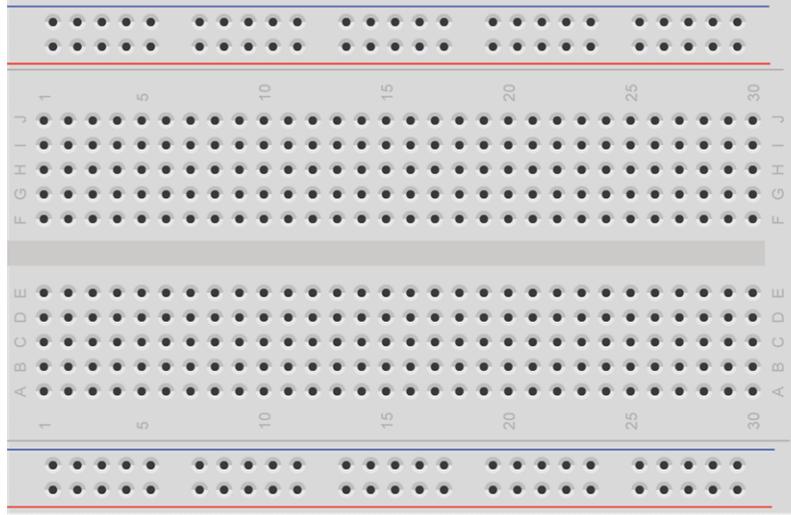


Native USB Port

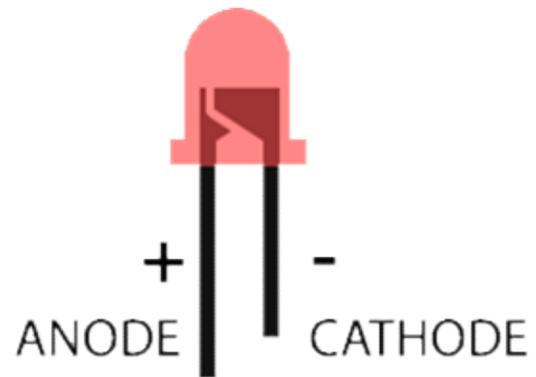
Programming Port

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

# Breadboard

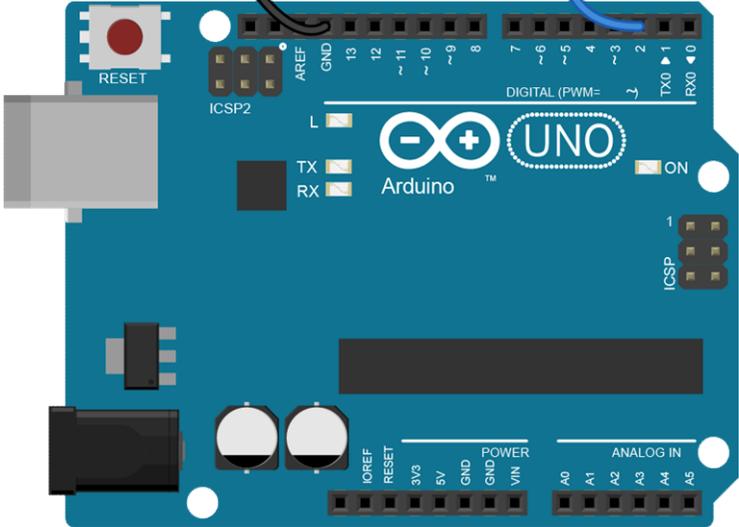
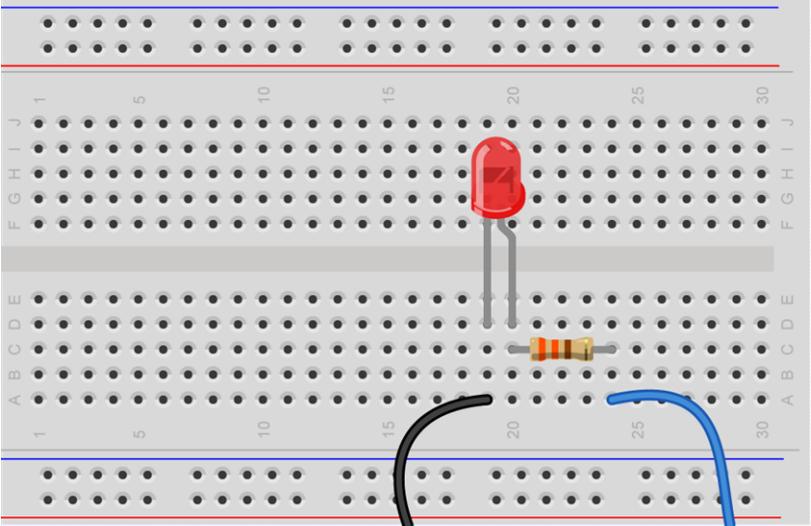
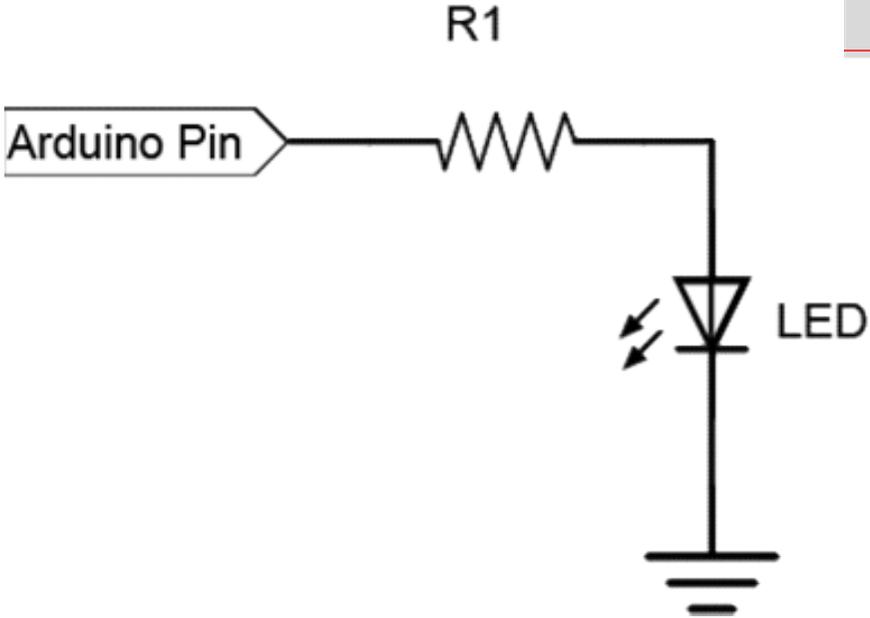


# LED



# Circuit

- Arduino board
- 1 breadboard
- 1 led
- 1 resistor of 150 ohm
- wires



# Code analysis 1/7

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
*/
```

```
B → // Pin 13 has an LED connected on most Arduino boards.  
B → // give it a name:  
    int led = 13;
```

```
B → // the setup routine runs once when you press reset:  
    void setup() {  
B → // initialize the digital pin as an output.  
        pinMode(led, OUTPUT);  
    }
```

```
// the loop routine runs over and over again forever:  
void loop() {  
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level) ← B  
    delay(1000); // wait for a second ← B  
    digitalWrite(led, LOW); // turn the LED off by making the voltage LOW ← B  
    delay(1000); // wait for a second ← B  
}
```

A

Commento su più linee

B

Commento su una linea

# Code analysis 2/7

```
/*  
  Blink  
  Turns on an LED on for one second  
  then off for one second repeating  
  This example code is in the p  
  */  
  
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once  
void setup() {  
  // initialize the digital pin  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over a  
void loop() {  
  digitalWrite(led, HIGH); //  
  delay(1000); //  
  digitalWrite(led, LOW); //  
  delay(1000); //  
}
```

;

Identifies where  
the instruction  
ends

{

...

}

Identifies a block  
of instructions

# Code analysis 3/7

```
/*
  Blink
  Turns on an LED on for one second
  ... then off for one second ...

  This example code is in the public domain
  and may be freely redistributed under the
  terms of the GNU public license.
  */

// Pin 13 has an LED connected to GND
// give it a name:
int led = 13;

// the setup routine runs once
void setup() {
  // initialize the digital pin
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over
void loop() {
  digitalWrite(led, HIGH); // turn the LED on
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off
  delay(1000);             // wait for a second
}
```

**int led = 13;**

A variable is a way for naming and storing a numerical value for later use by the program. 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.

These operations are carried out only once in program, but the value of the variable can be changed at any time using the arithmetic or using assignments. The following example stated that LED is an int, (Integer type) and that its initial value is equal to 13. This is called a **simple assignment**.

# Code analysis 4/7

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one  
  
  This example code is in the public domain.  
  */  
  
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize the digital pin as an output.  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```

## Struttura di base

```
void setup()  
{  
  istruzioni;  
}  
  
void loop()  
{  
  istruzioni;  
}
```

# Code analysis 5/7

```
/*
  Blink
  Turns on an LED on for one second

  This example code is in the public domain
  */

// Pin 13 has an LED connected on a Arduino
// give it a name:
int led = 13;

// the setup routine runs once when you
// reset the board
void setup() {
  // initialize the digital pin as
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over
// again forever
void loop() {
  digitalWrite(led, HIGH); // turn the LED on
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off
  delay(1000);             // wait for a second
}
```

## **pinMode (LED, OUTPUT);**

pinMode is an instruction that specifies how a particular pin is defined. In the parentheses, topics that can be numbers and letters are specified. Digital pins can be used as INPUT or OUTPUT. In this case, we want to flash the diode, for this reason LED must be define as OUTPUT pin. The INPUT and OUTPUT words are defined constants

# Code analysis 6/7

```
/*  
  Blink  
  Turns on an LED on for one second,  
  
  This example code is in the public  
  */  
  
// Pin 13 has an LED connected on most  
// boards; give it a name:  
int led = 13;  
  
// the setup routine runs once when you  
void setup() {  
  // initialize the digital pin as an  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over  
void loop() {  
  digitalWrite(led, HIGH); // turn  
  delay(1000);             // wait  
  digitalWrite(led, LOW);  // turn  
  delay(1000);             // wait  
}
```

## **digitalWrite (led, HIGH);**

The digitalWrite instruction has two arguments: the first one defines the pin, the second one indicates the status.

If the pin is configured as an OUTPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the output pin (it turns on or off an LED). The 'pin' can be specified as a variable or a constant. If the pin state is HIGH, it means that is applied a voltage of 3,3V (5V for Arduino Uno), while if the state is LOW the applied voltage is 0V.

# Code analysis 7/7

```
/*  
  Blink  
  Turns on an LED on for or  
  
  This example code is in the  
  */  
  
// Pin 13 has an LED connected  
// give it a name:  
int led = 13;  
  
// the setup routine runs once  
void setup() {  
  // initialize the digital  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over  
void loop() {  
  digitalWrite(led, HIGH);  
  delay(1000);  
  digitalWrite(led, LOW);  
  delay(1000);  
}
```



## **delay (1000);**

`delay ()` Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second). The instruction has a single numeric argument; It indicates the number of milliseconds to wait.

Second Program

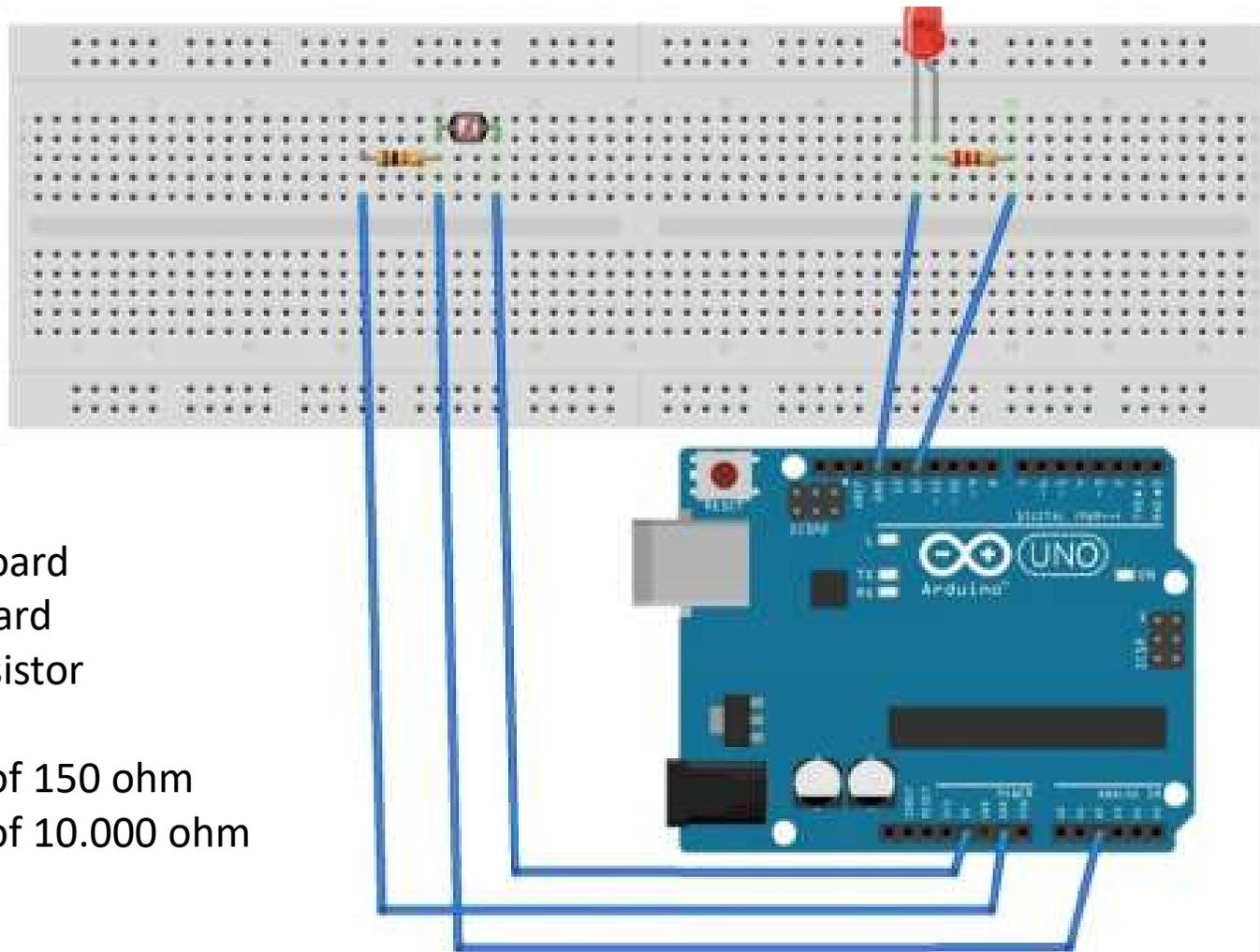
Photoresistor with LED

# Photoresistor

A photoresistor is a resistance whose impedance (and that means' whose capacity 'to circulate electricity') varies according to the light that strikes it. While the light increase, the resistance decreases, and vice versa. It is typically an analog type sensor. To use in necessary to connect a leg to an analog port and, in parallel, to a 10k ohm resistor connected to ground while connecting the other leg power from 5 volts. The analog port returns a value from 0 to 1023 (or from 0 to 4095 in 12 bits ADC) that varies according to the light that strikes the photoresistor.



# Circuit for photoresistor



- Arduino board
- 1 breadboard
- 1 photoresistor
- 1 led
- 1 resistor of 150 ohm
- 1 resistor of 10.000 ohm
- 5 wires

# analogReadResolution()

## Description

`analogReadResolution()` is an extension of the Analog API for the Arduino Due and Zero.

Sets the size (in bits) of the value returned by `analogRead()`. It defaults to 10 bits (returns values between 0-1023) for backward compatibility with AVR based boards.

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.

# Sketch

```
*/  
//  
int valorefotocellula;    // variabile in cui viene inserito il valore analogico (da 0 a 1023)  
                          // della tensione rilevata sulla fotocellula.  
//  
//  
void setup()  
{  
  pinMode(12, OUTPUT);   // definisce la porta 12 come porta di output  
}  
//  
//  
void loop()  
{  
  valorefotocellula = analogRead(2); // legge il valore fornito dalla fotoresistenza  
  if(valorefotocellula<=512) // 512 e' un valore intermedio (la scala analogica va  
                             // da 0 a 1023). Per rendere il sensore piu' o meno sensibile sara'  
                             // sufficiente aumentare o diminuire questo parametro.  
/* nota: in realta' sulla porta 2 arduino non legge il valore della luce ambientale, ma una  
tensione, che sara' bassa se l'impedenza della fotoresistenza (dipendente dalla luce ambientale)  
sara' alta e viceversa */  
  {  
    digitalWrite(12, HIGH); // accende il led se l'impedenza della fotoresistenza (impedenza  
                             // proporzionale alla luce rilevata) e' alta e quindi la luce ambientale e' bassa  
  }  
  else  
  {  
    digitalWrite(12, LOW); // in caso contrario lo spegne  
  }  
}
```

Libraries

# Arduino Libraries

- If there is a library that you need but is not included with the IDE, you can install it. Let's look at an example.
- Download the ZIP file on your computer. It doesn't matter what platform you are on; the libraries work the same regardless of whether you are on Windows, Mac or Linux.
- Also, don't worry about extracting the files from the ZIP archive. The newer versions of the Arduino IDE have an easy library installer that takes care of extracting the library from the ZIP file and copying the files to the right location.
- Assuming the library ZIP file is in your Downloads folder, start the Arduino IDE. Then click on "Sketch → Include Library → Add .ZIP Library...", like this:

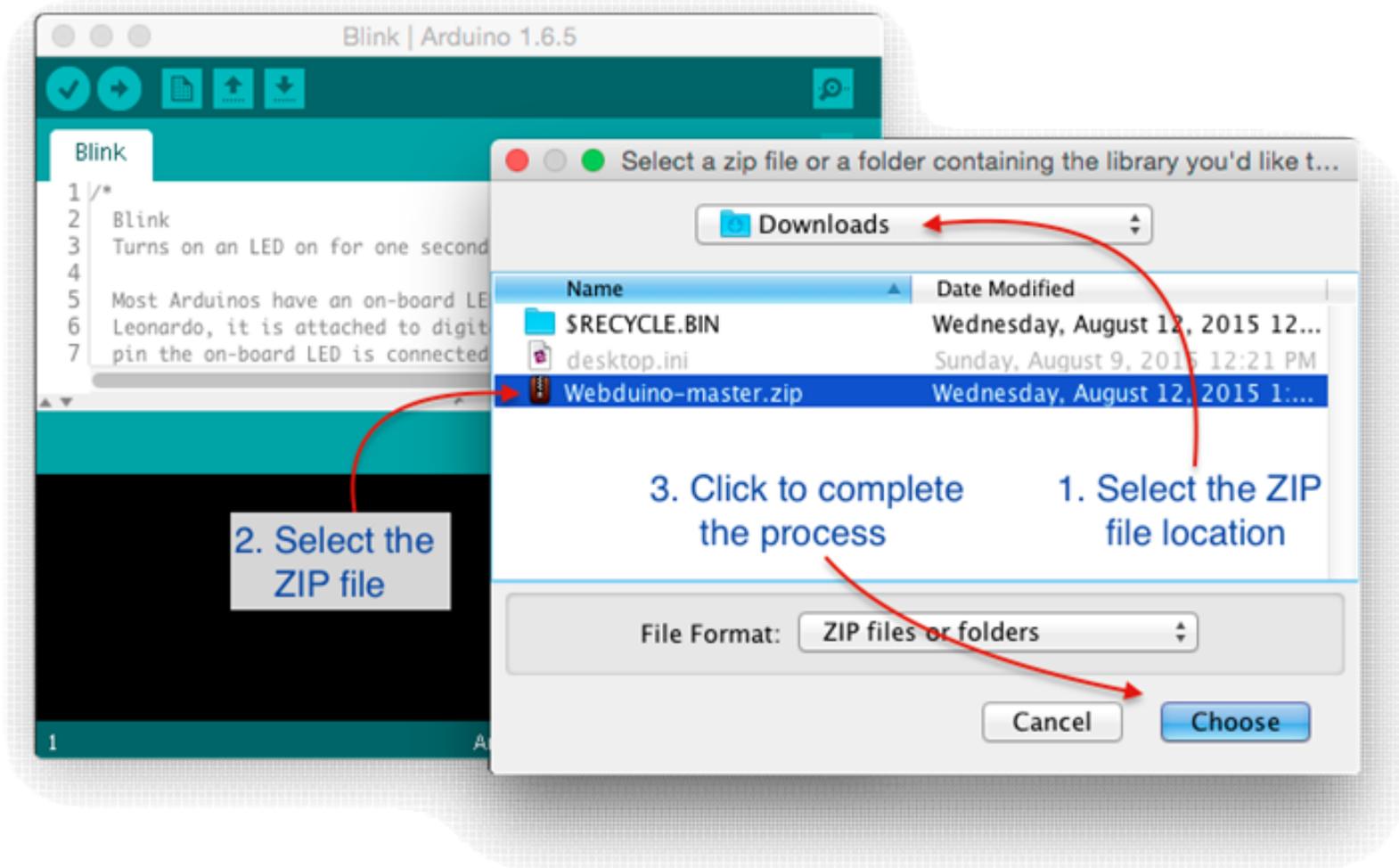
```
1 /*  
2  Blink  
3  Turns on an LED on for one second, then off for one second, repeating.  
4  
5  Most Arduinos have an on-board LED  
6  Leonardo, it is attached to digital pin 13. If you're unsure what  
7  pin the on-board LED is connected to on your Arduino model, check
```

- Verify / Compile ⌘R
- Upload ⌘U
- Upload Using Programmer ⇧⌘U
- Export compiled Binary ⌘S
- Show Sketch Folder ⌘K
- Include Library** ▶
- Add File...

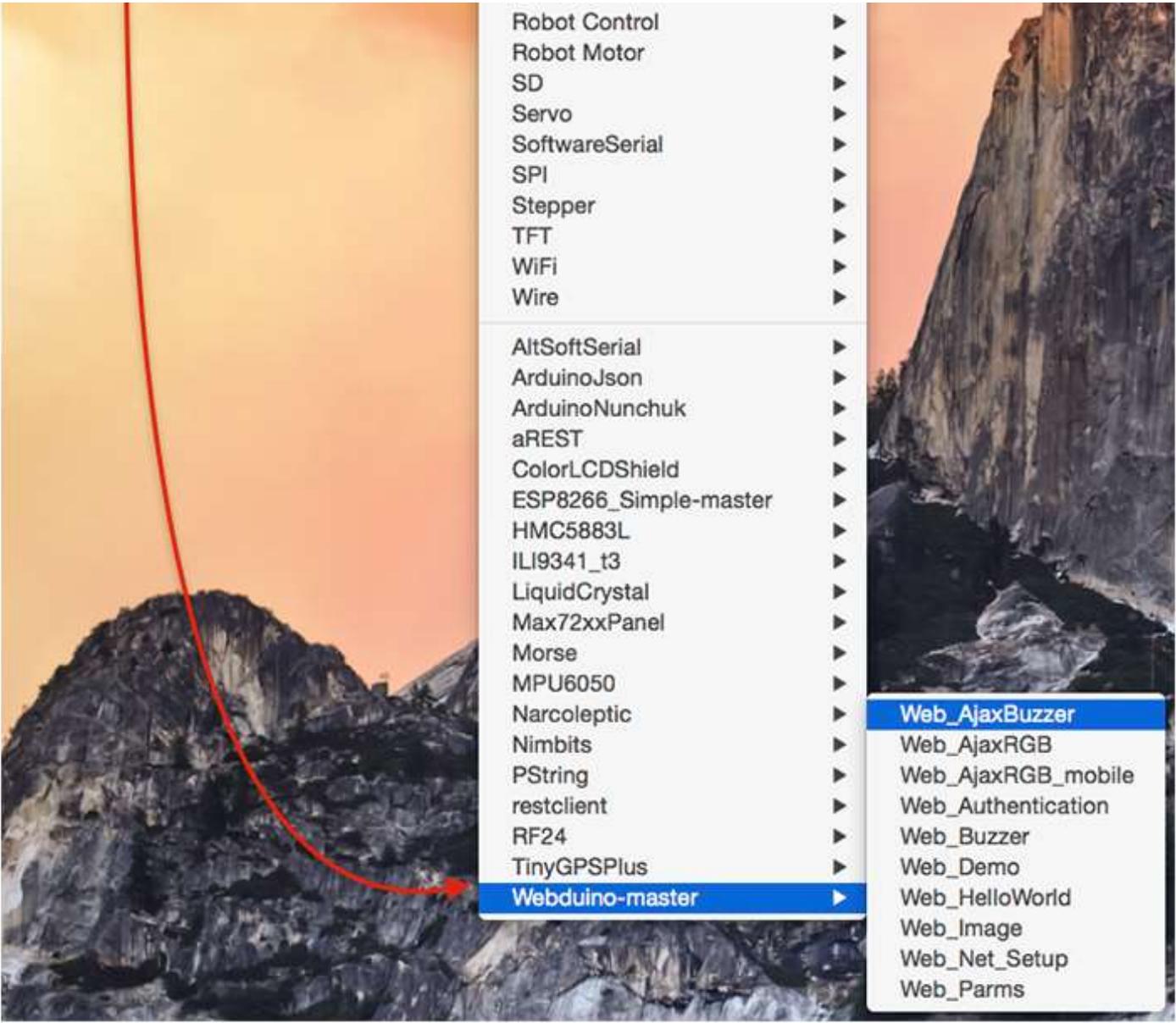
- Manage Libraries...
- Add .ZIP Library...**
- Arduino libraries
- Bridge
- EEPROM
- Esplora
- Ethernet
- Firmata
- GSM
- Robot Control
- Robot IR Remote
- Robot Motor
- SD
- SPI
- Servo
- SoftwareSerial
- SpacebrewYun
- Stepper
- TFT

Click to include a new library

A new dialogue box will pop up. Browse to the location of the ZIP file, select it, and click on Choose to complete the process:



- When you click on “Choose”, the dialogue box will disappear, but nothing else is going to happen. No confirmation, no sound... To make sure that the Webuino library was actually installed, you can look for the example sketches that most libraries include.
- Go to File → Examples, and look at the bottom of the list for your new library:



- Robot Control ▶
- Robot Motor ▶
- SD ▶
- Servo ▶
- SoftwareSerial ▶
- SPI ▶
- Stepper ▶
- TFT ▶
- WiFi ▶
- Wire ▶

- AltSoftSerial ▶
- ArduinoJson ▶
- ArduinoNunchuk ▶
- aREST ▶
- ColorLCDShield ▶
- ESP8266\_Simple-master ▶
- HMC5883L ▶
- ILI9341\_t3 ▶
- LiquidCrystal ▶
- Max72xxPanel ▶
- Morse ▶
- MPU6050 ▶
- Narcoleptic ▶
- Nimbits ▶
- PString ▶
- restclient ▶
- RF24 ▶
- TinyGPSPlus ▶

- Webduino-master ▶**

- Web\_AjaxBuzzer**
- Web\_AjaxRGB
- Web\_AjaxRGB\_mobile
- Web\_Authentication
- Web\_Buzzer
- Web\_Demo
- Web\_HelloWorld
- Web\_Image
- Web\_Net\_Setup
- Web\_Parms