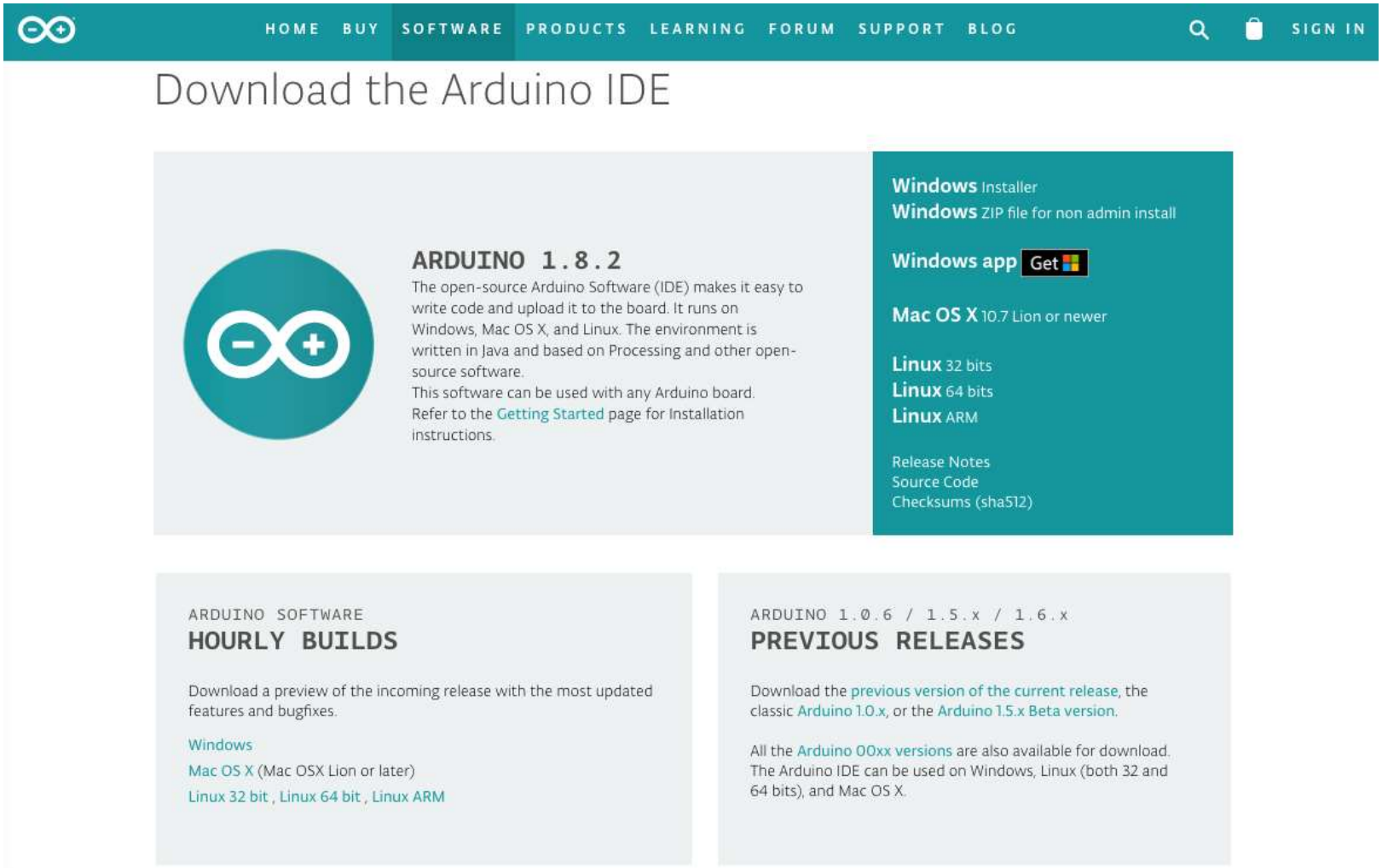
PROGRAMMING WITH ARDUINO

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



Software - Installation

- Windows

  arduino.cc/windows

  Installation for: Windows 7, Vista, e XP
- Mac OS X

  arduino.cc/mac

  Installation for: OS X 10.7 or newer
- Linux

  arduino.cc/linux

Communication with Arduino

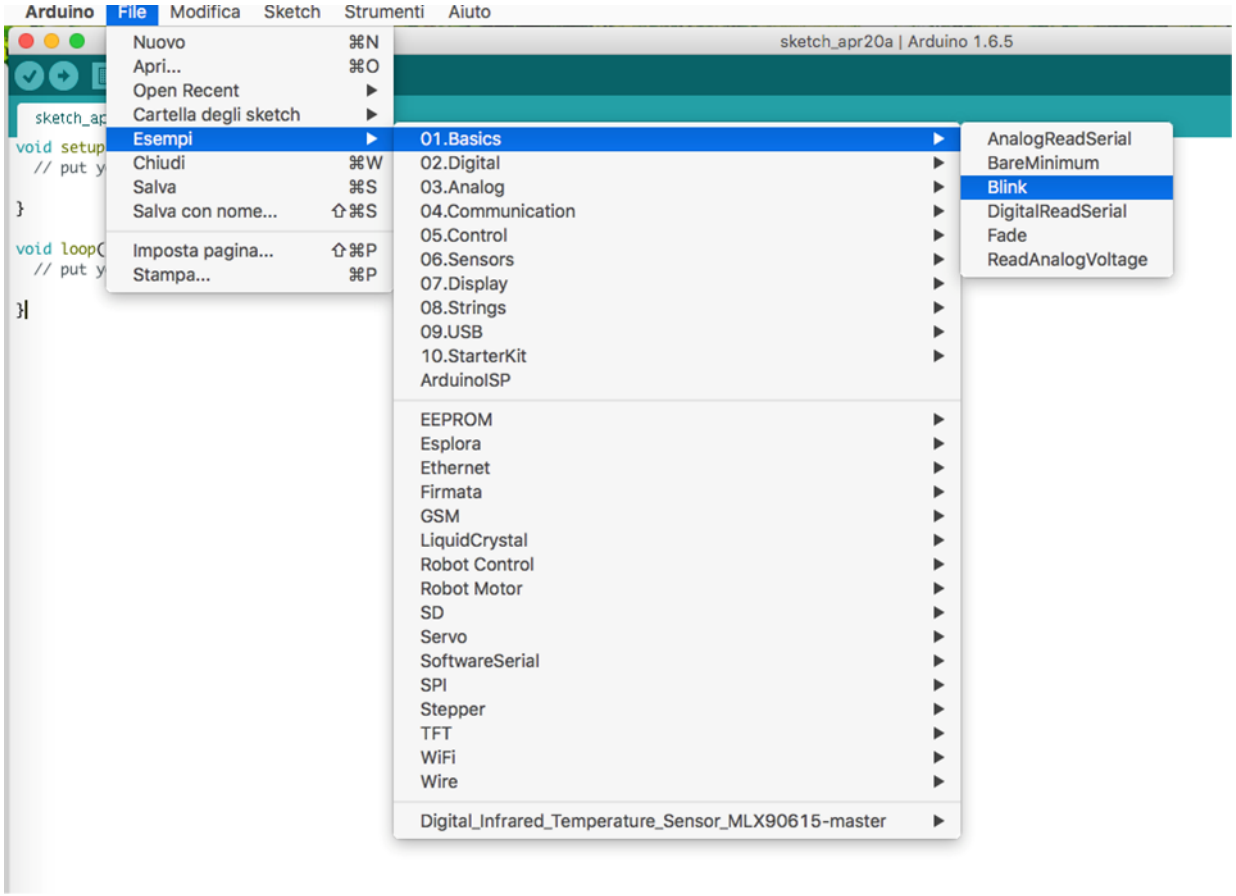
- Launch the Arduino IDE (double click)

Arduino Program Development

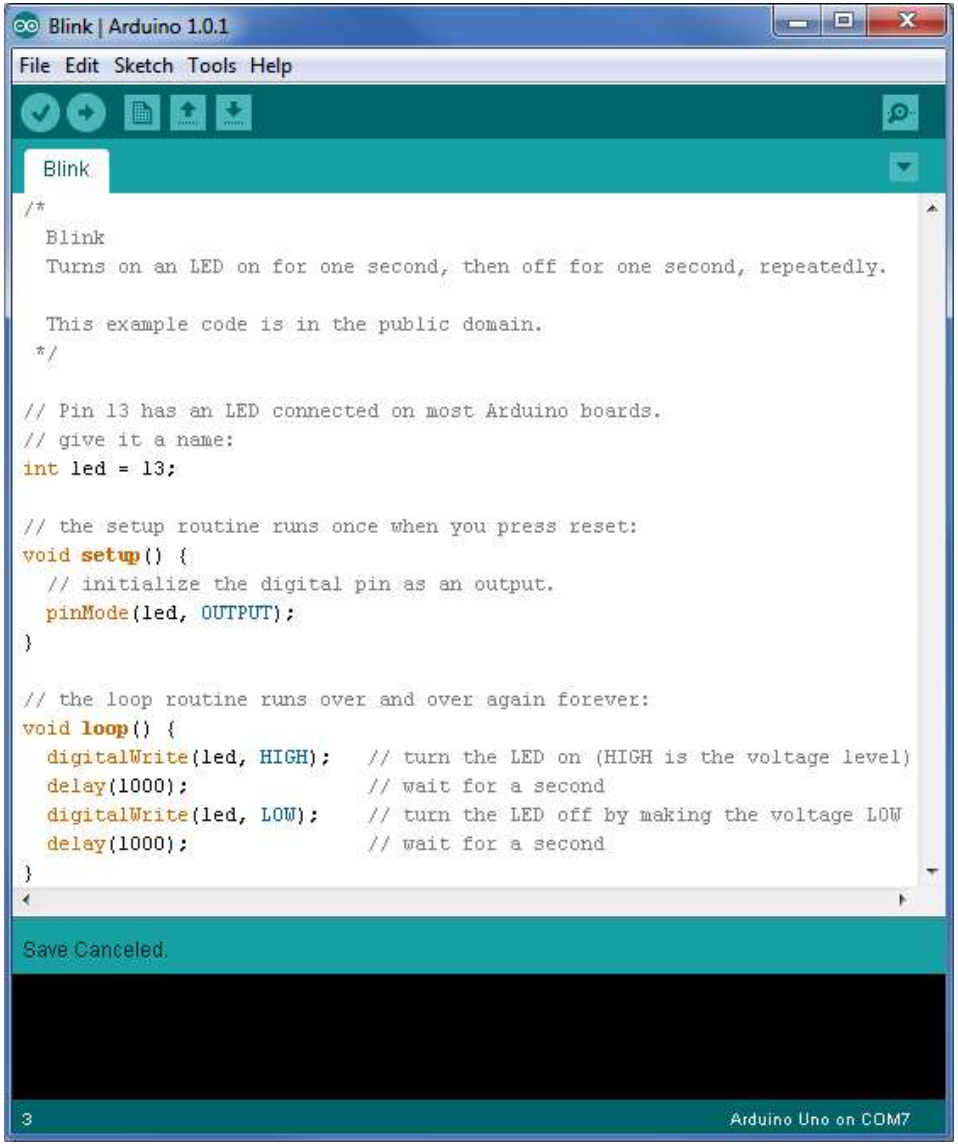
- Based on C++ without 80% of the instructions.
- A handful of new commands.
- Programs are called 'sketches'.
- Sketches need two functions:
  - void setup( )
  - void loop( )
- setup( ) runs first and once.
- loop( ) runs over and over, until power is lost or a new sketch is loaded.

- Open the sketch

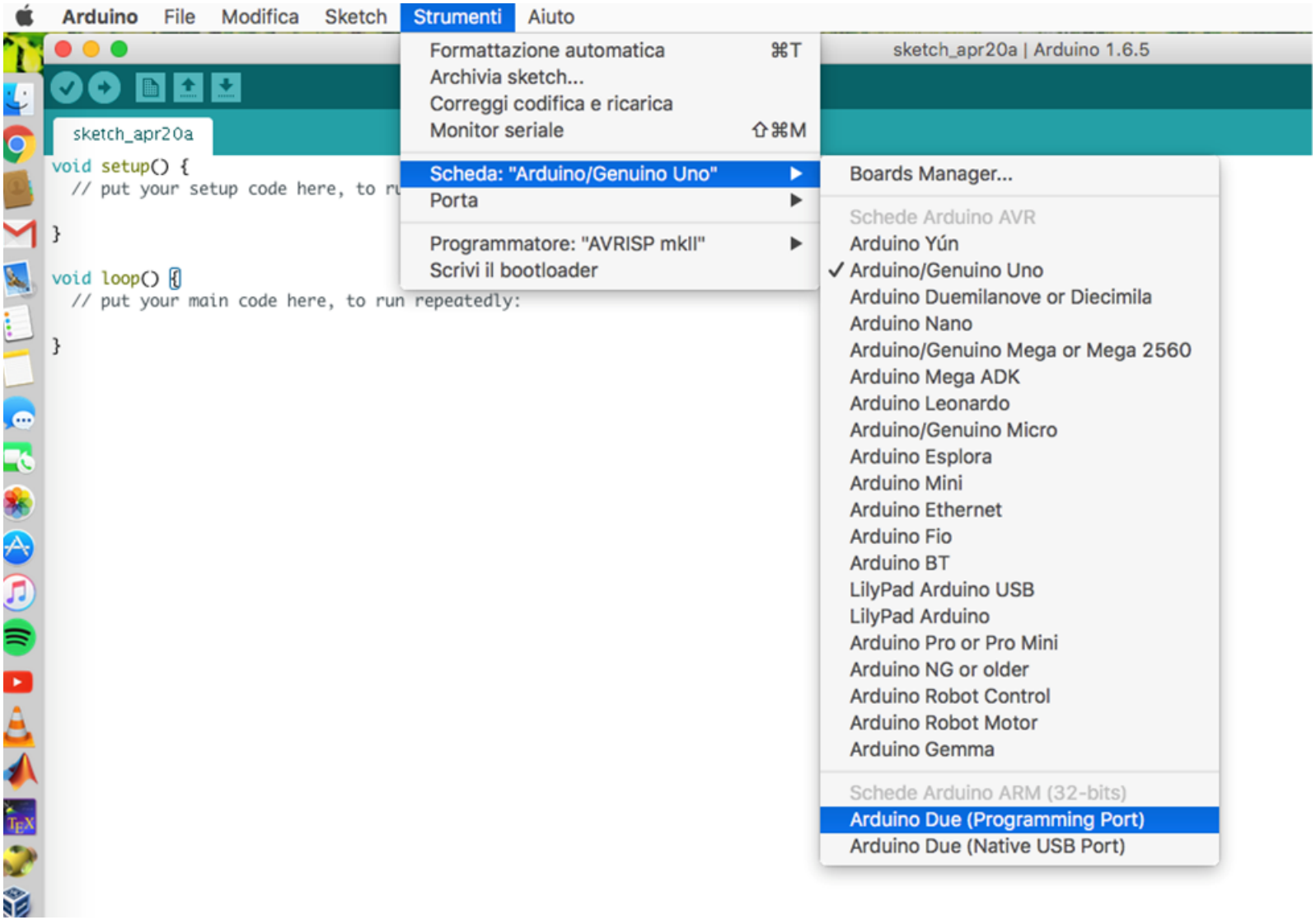
- Numerous sample sketches are included in the compiler
- Located under File, Examples
- Once a sketch is written, it is uploaded by clicking on File, Upload, or by pressing <Ctrl> U



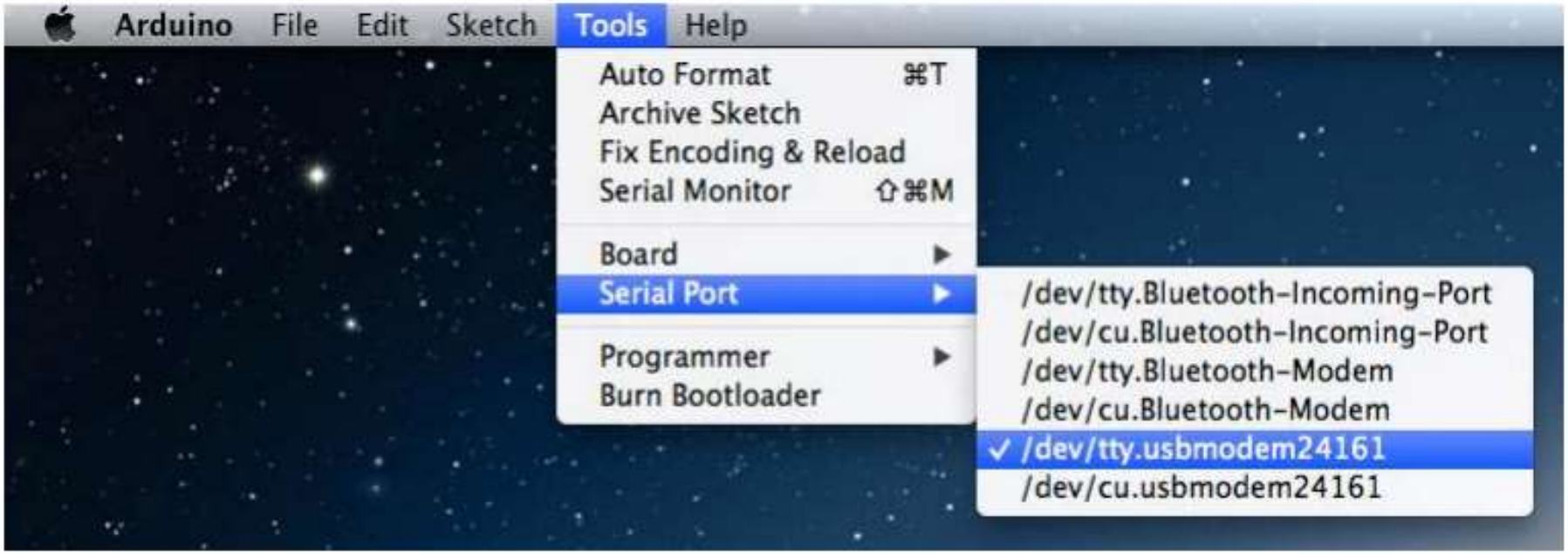
- Open the Blink sketch



- Select the Board

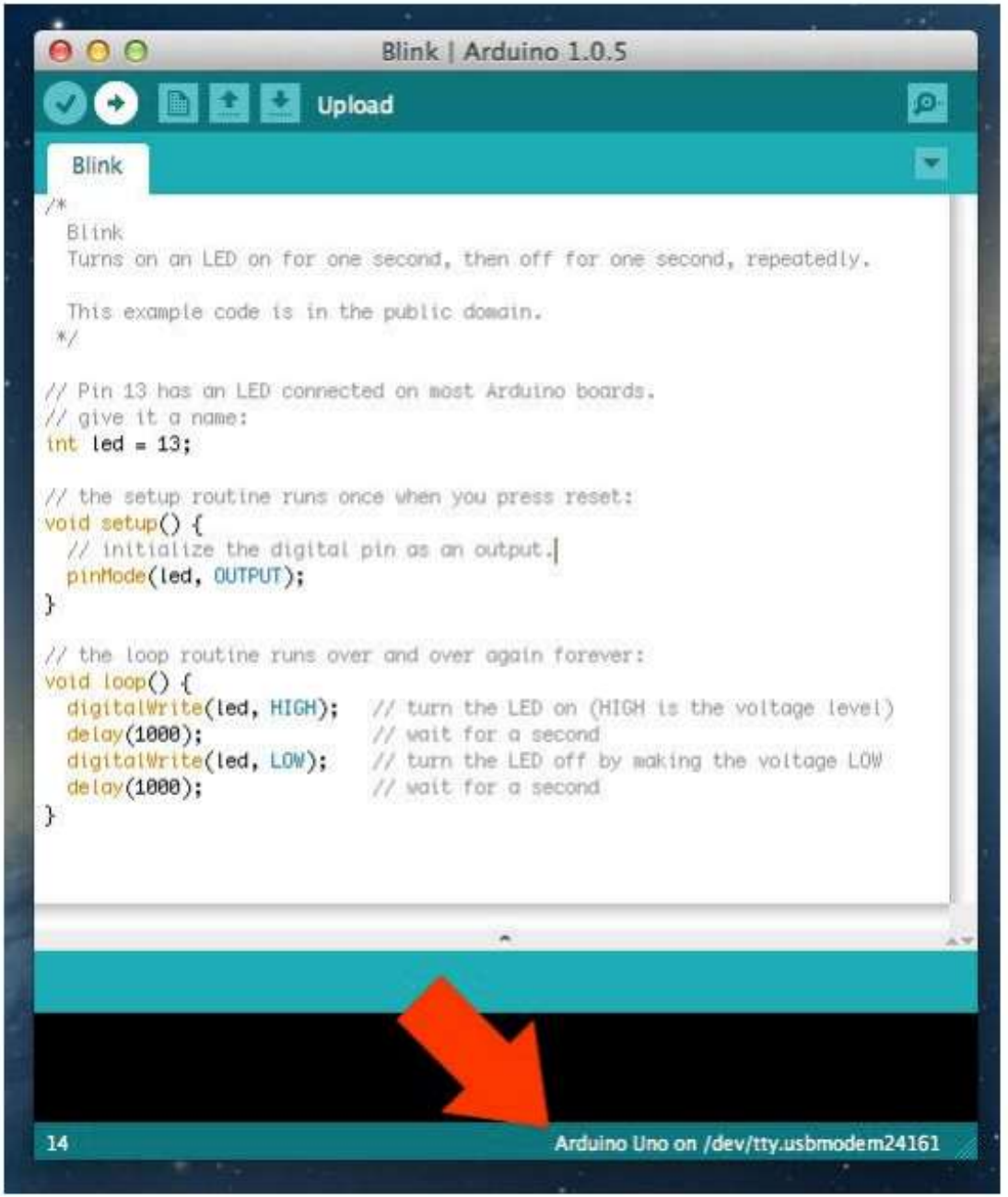


- Select the Serial Port

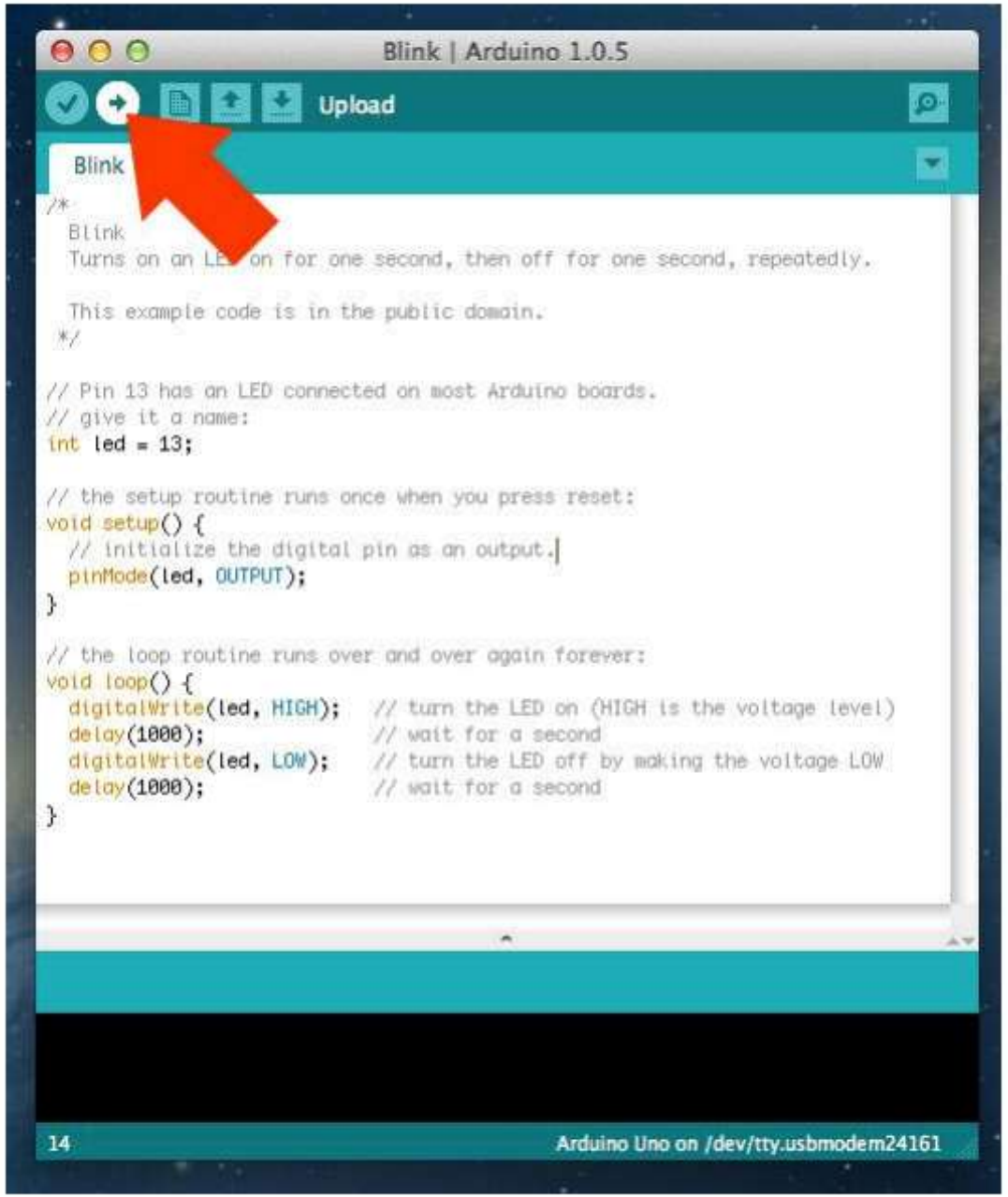


- **Mac**:
  You can indifferently choose between
  /dev/tty.usbmodemXXXXX or /dev/cu.usbmodemXXXXX
- **Windows**:
  There are one or more COM ports:
  choose the one with the higher number if it does not work try with the other proposals.

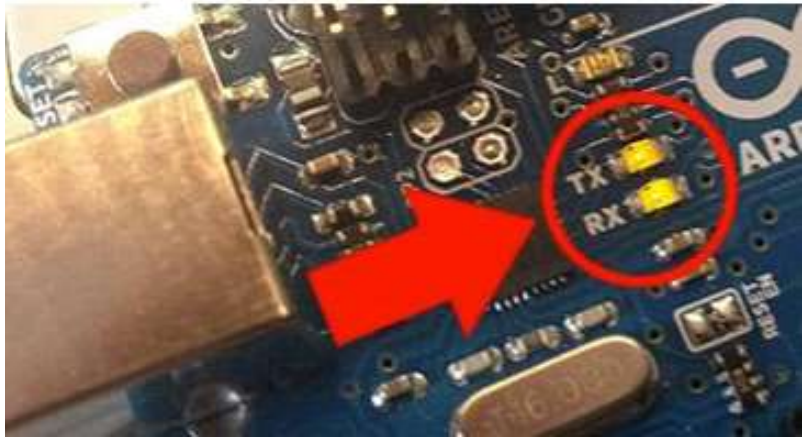
- The connection to the serial port is reported in the code window in bottom right



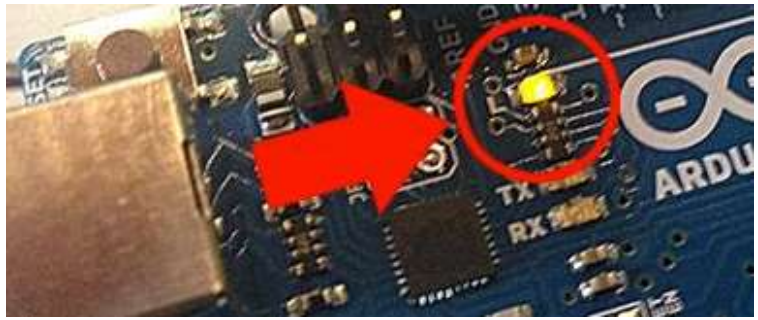
- Loading the Blink sketch on the board through the Upload button

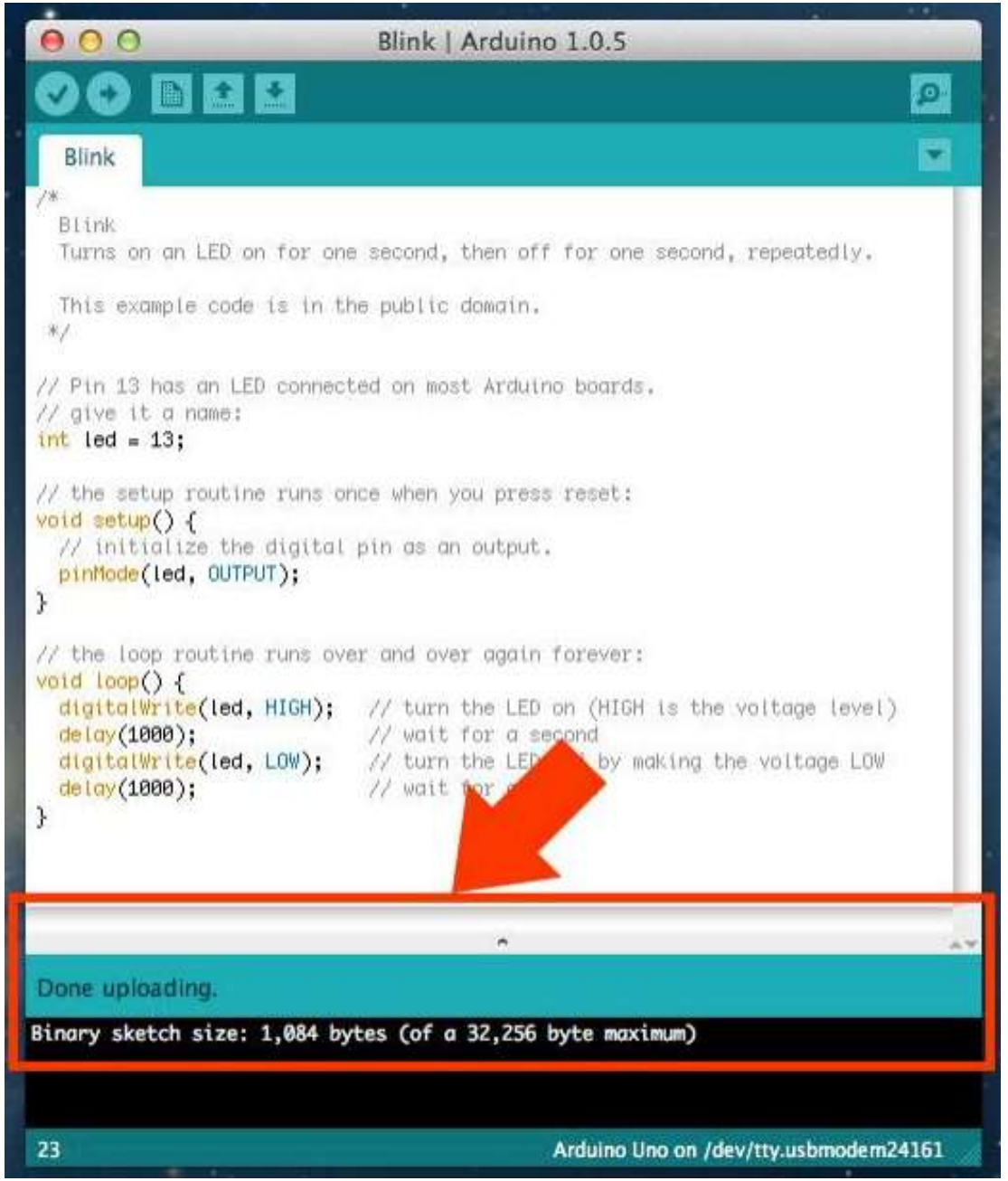


- It will take a few seconds, during this operation you will see that the LEDs RX and TX (receive and transmit)flash.



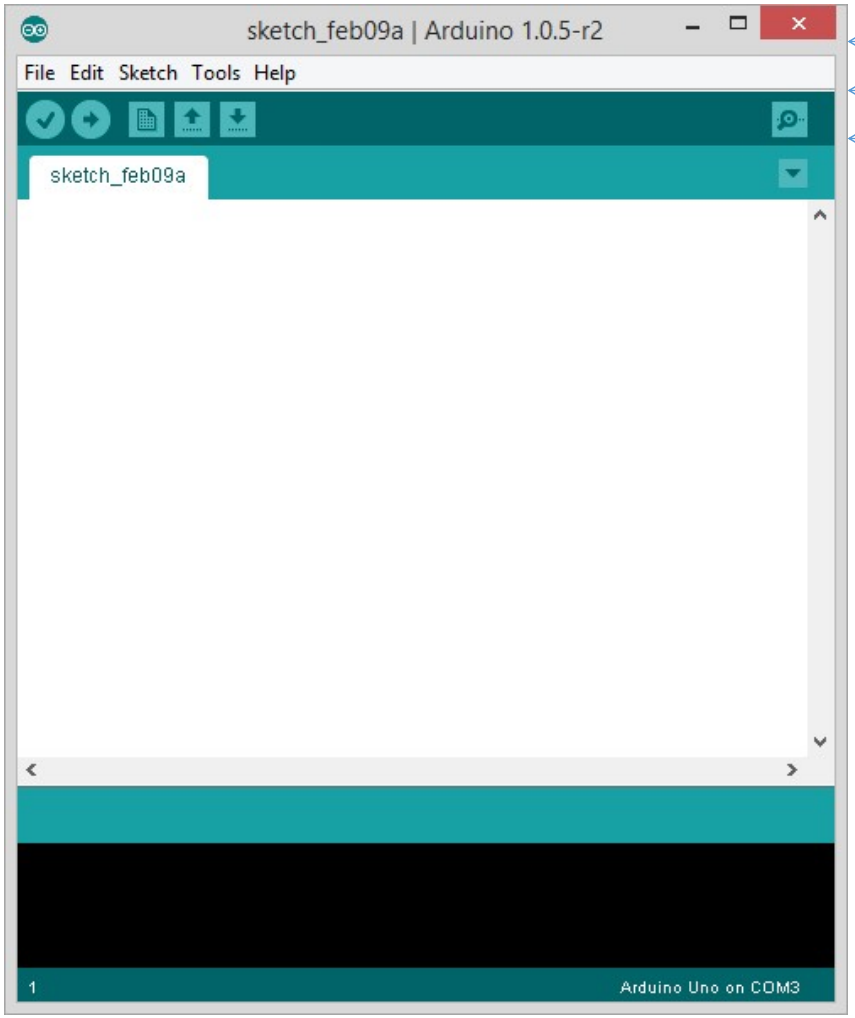
If everything will be succesfull you will be returned message "Done uploading.” in the staus bar, and the LED L starts flashing





Programming

Parts of the IDE main screen



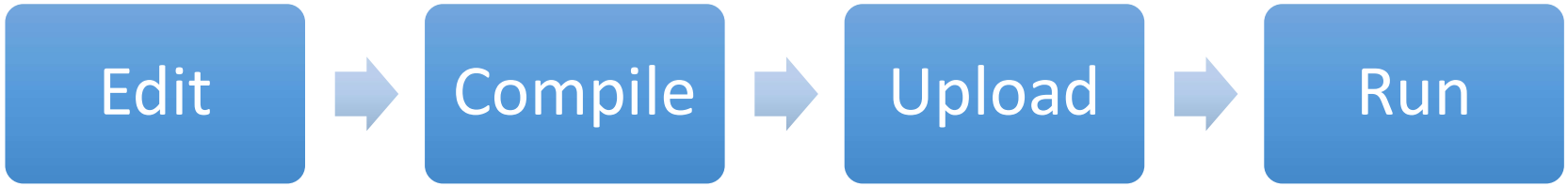
Name of current sketch

Main menus

Action buttons/icons

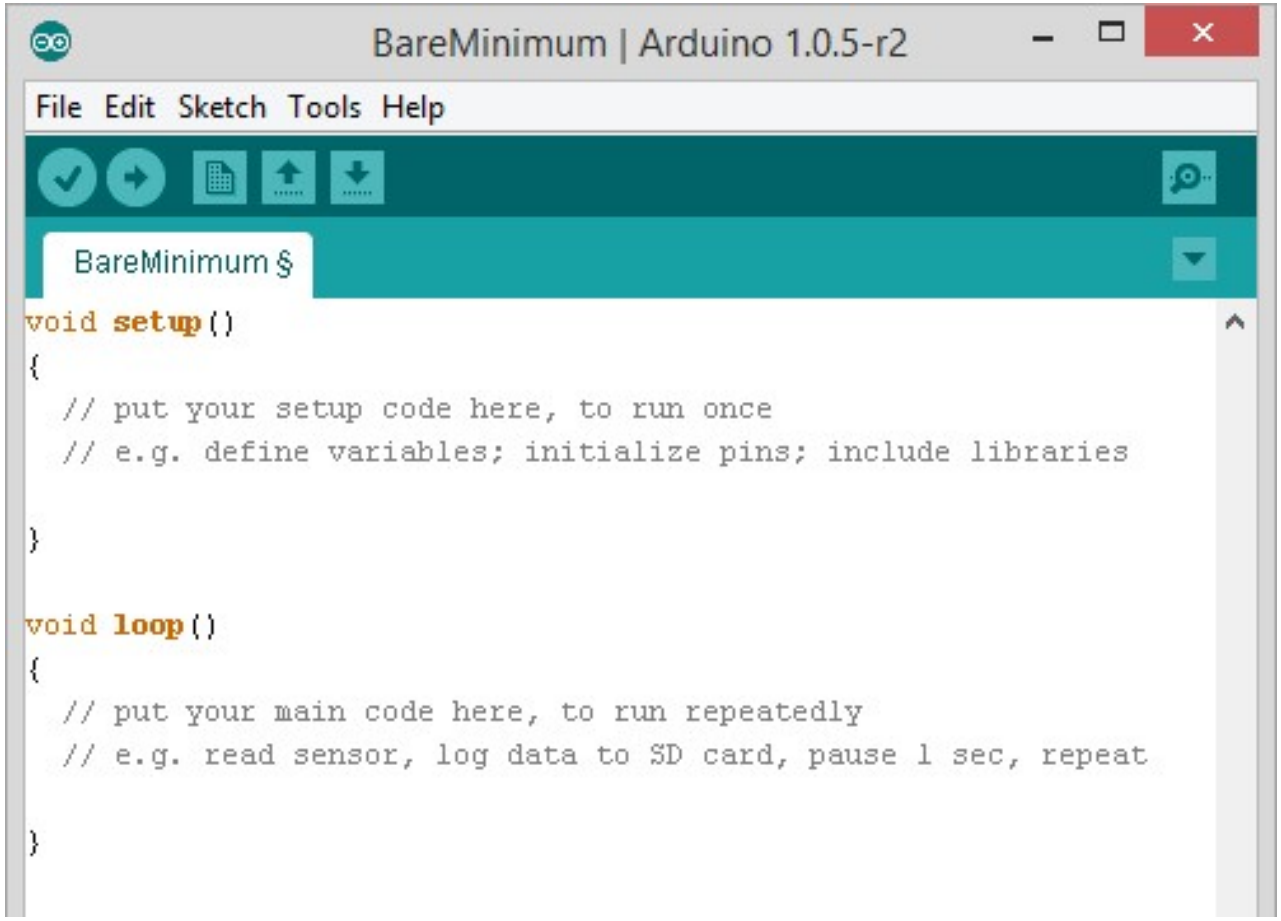
- Verify (AKA compile)
- Upload (send to Arduino)
- Start a new sketch
- Open a sketch (from a file)
- Save current sketch (to a file)
- Open Serial Monitor window

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



- 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

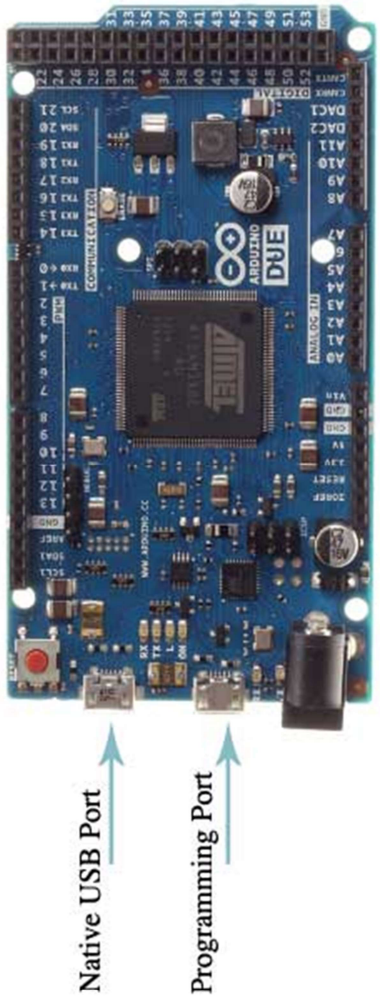
- **pinMode(*pin*, *mode*)**
  - Configures a digital pin to read (input) or write (output) a digital value
- **digitalWrite(*pin*, *value*)**
  - Writes the digital value (HIGH or LOW) to a pin set for output
- **digitalRead(*pin*)**
  - Reads a digital value (HIGH or LOW) on a pin set for input
- **analog 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

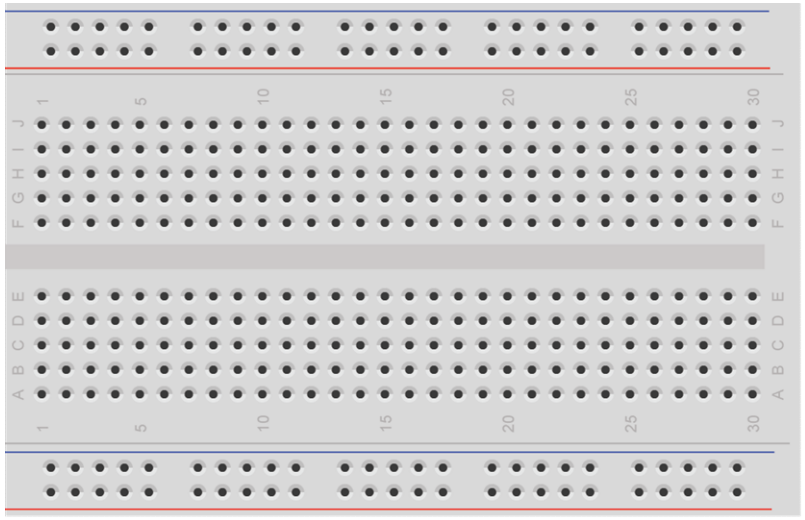
Blinking LED

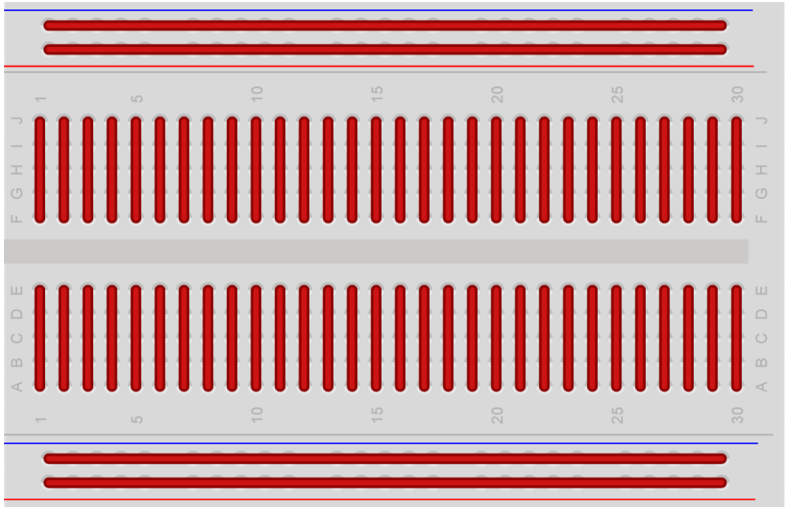
Arduino DUE



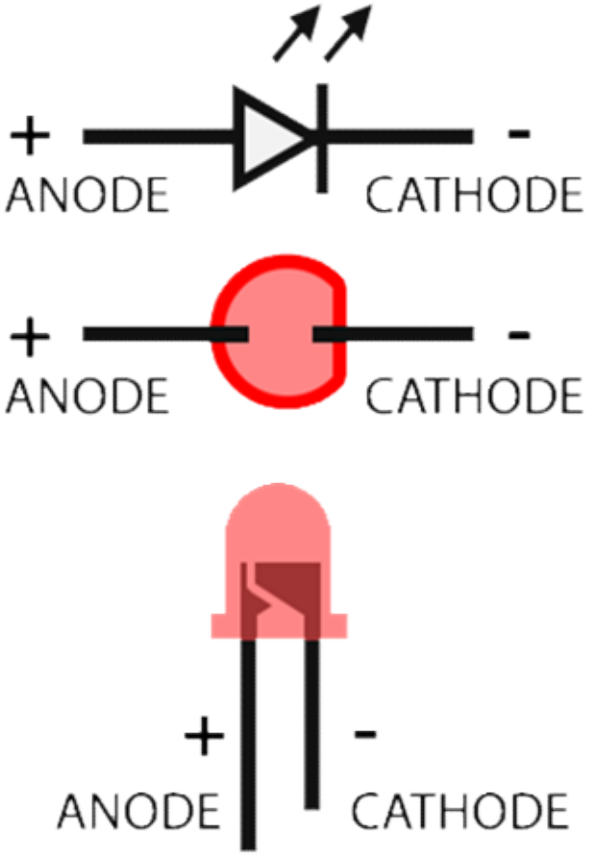
Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current 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



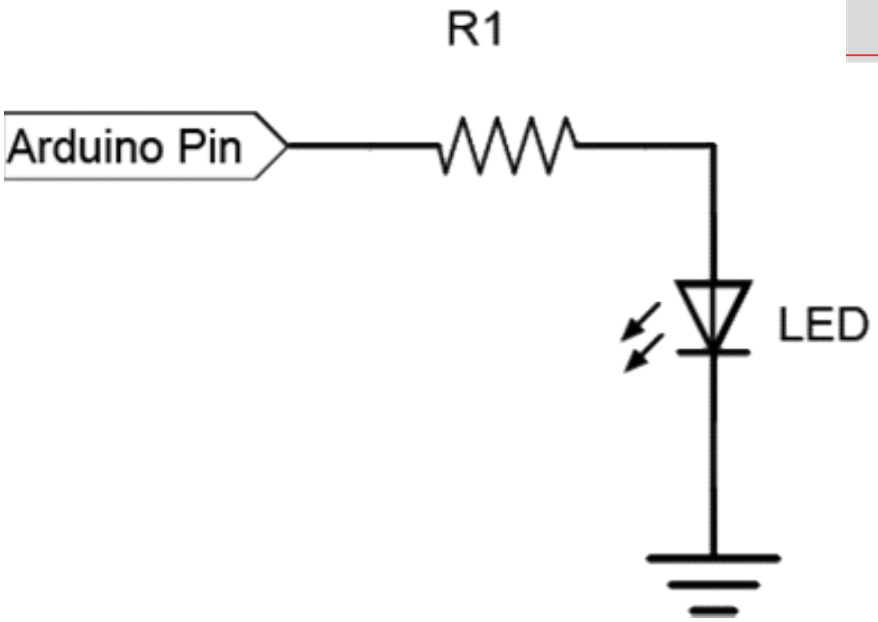


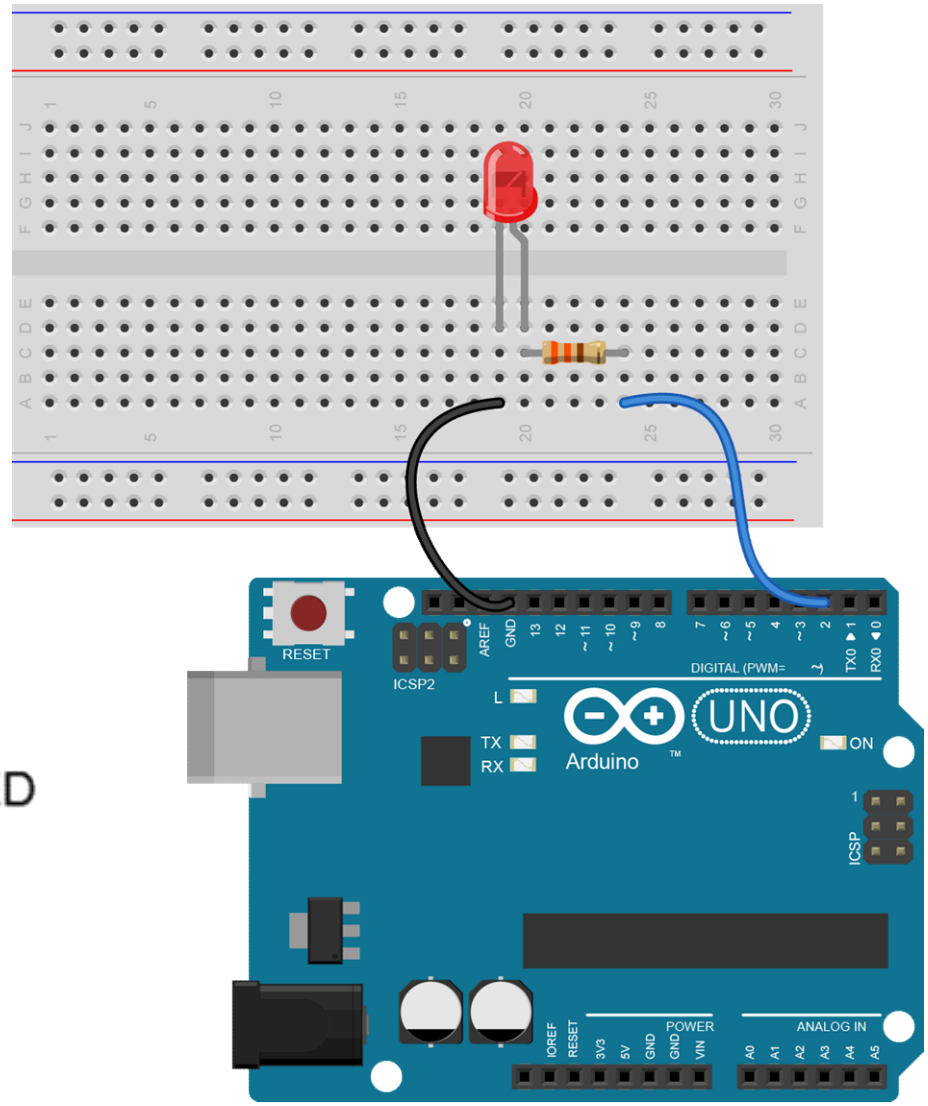
LED



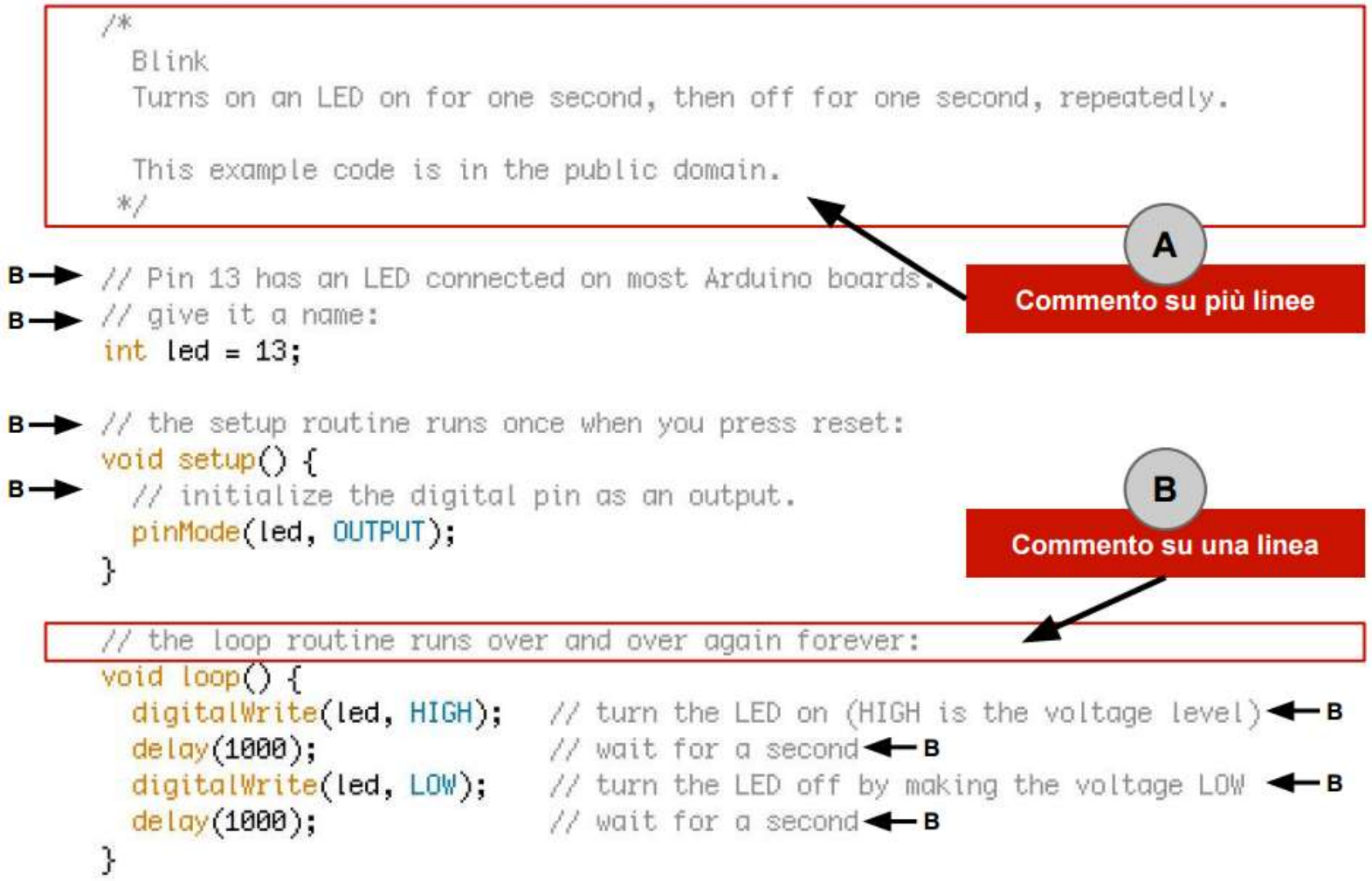
Circuit

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

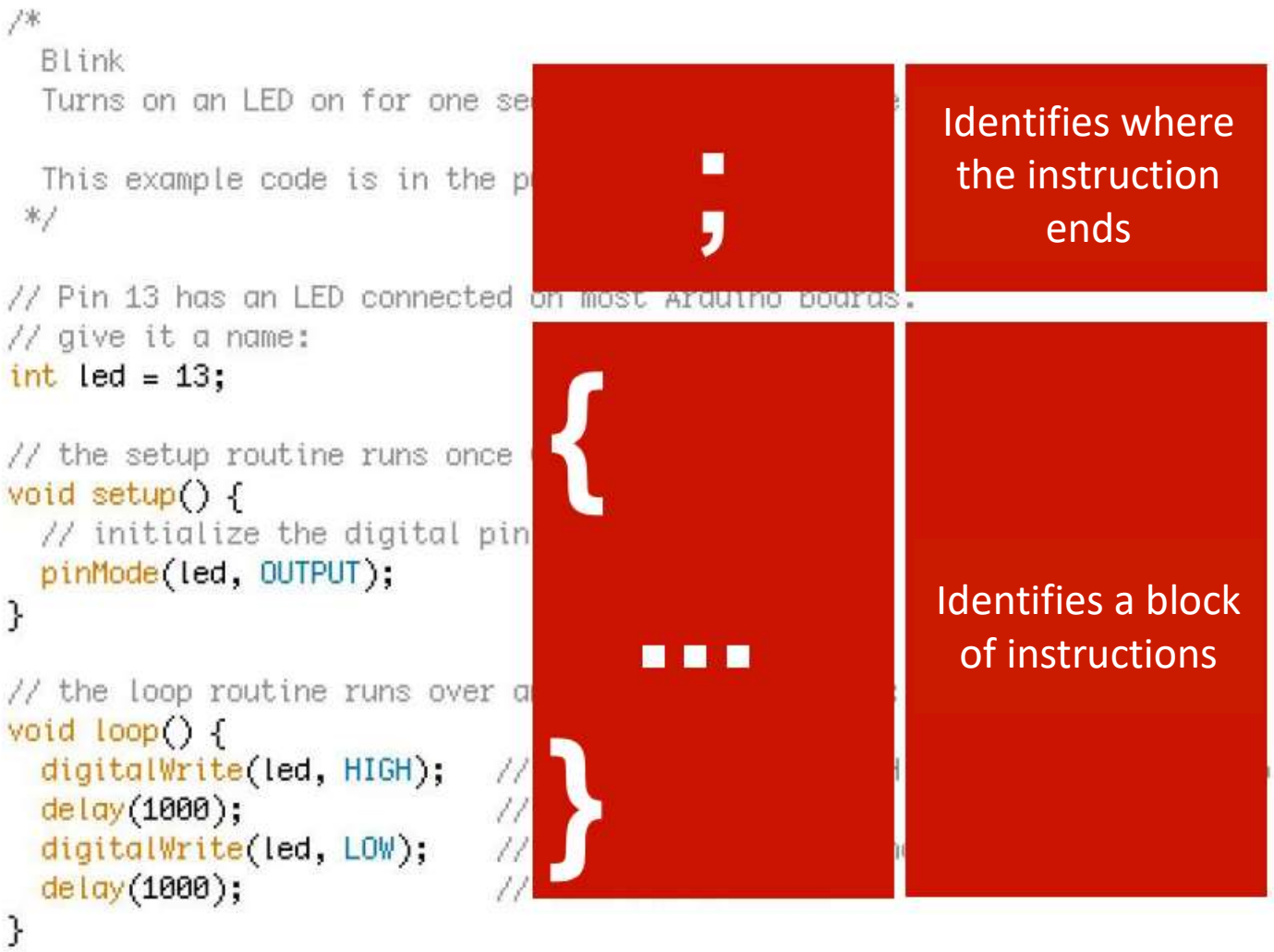




Code analysis 1/7



Code analysis 2/7



Code analysis 3/7

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

  This example code is in the p
 */

// Pin 13 has an LED connected
// give it a nam
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);               //
}
```

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 s

  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 seco

  This example code is in the pub
 */

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

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

// the loop routine runs over and
void loop() {
  digitalWrite(led, HIGH);   // t
  delay(1000);               // w
  digitalWrite(led, LOW);    // t
  delay(1000);               // w
}
```

pinMode (LED, OUTPUT);

pinMode is an instruction that specifyes 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 mos
// give it a name:
int led = 13;

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

// the loop routine runs over and ove
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  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 HIGHT, 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 t
 */

// Pin 13 has an LED connec
// give it a name:
int led = 13;

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

// the loop routine runs ov
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

delay (1000);

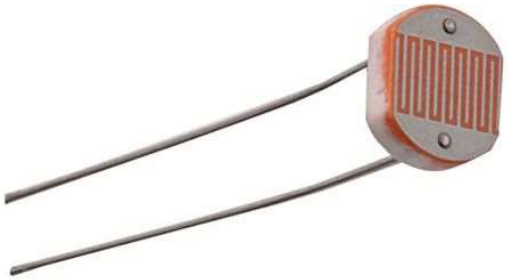
delay () Pauses the program for the amount of time (in miliseconds) specified as parameter. (There are 1000 milliseconds in a second). The instruction has a single numeric argument; It indicates the number of milliseconds to wait.

Second Program

Photoresistor with LED

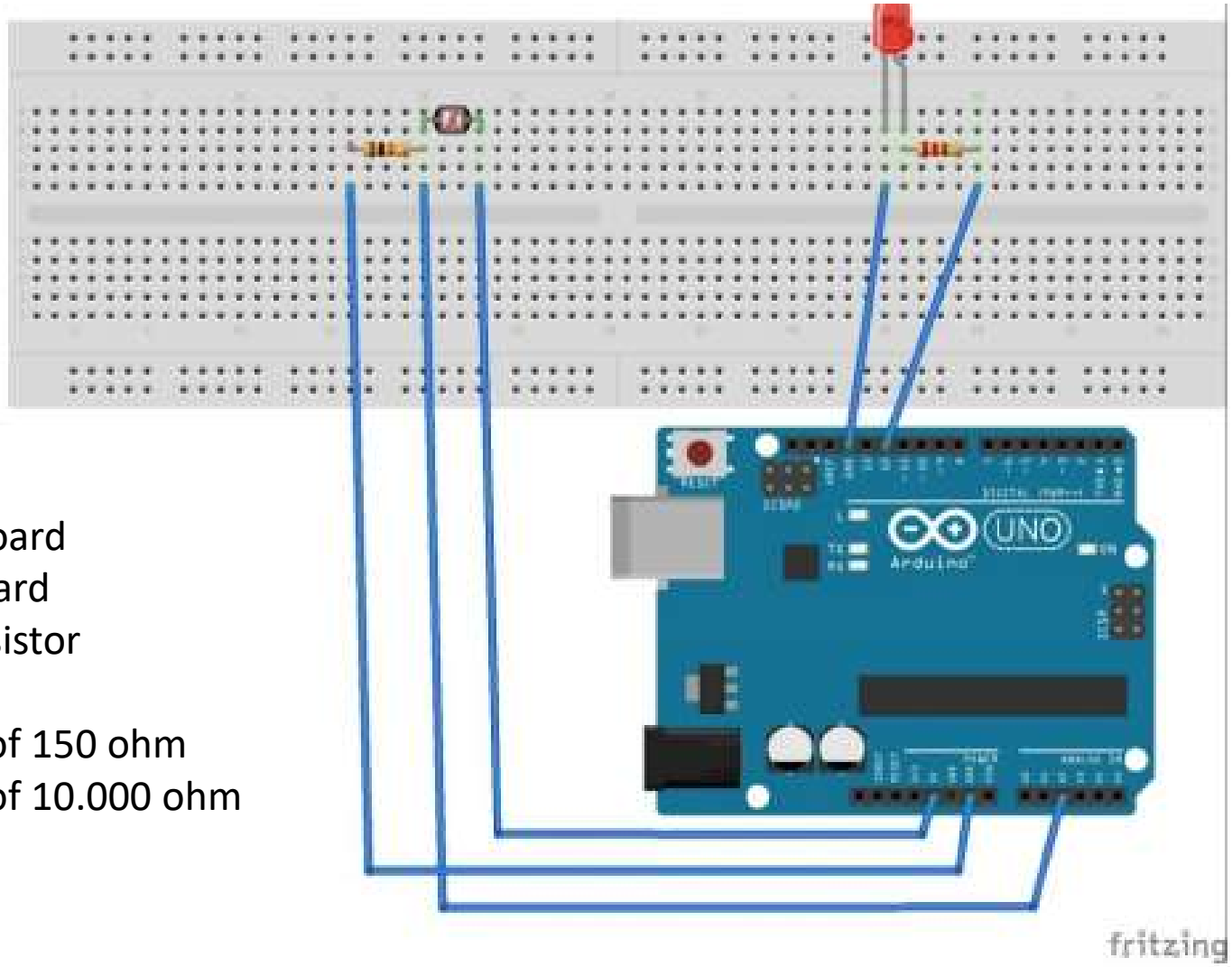
Photoresistor

A photoresistor and '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.

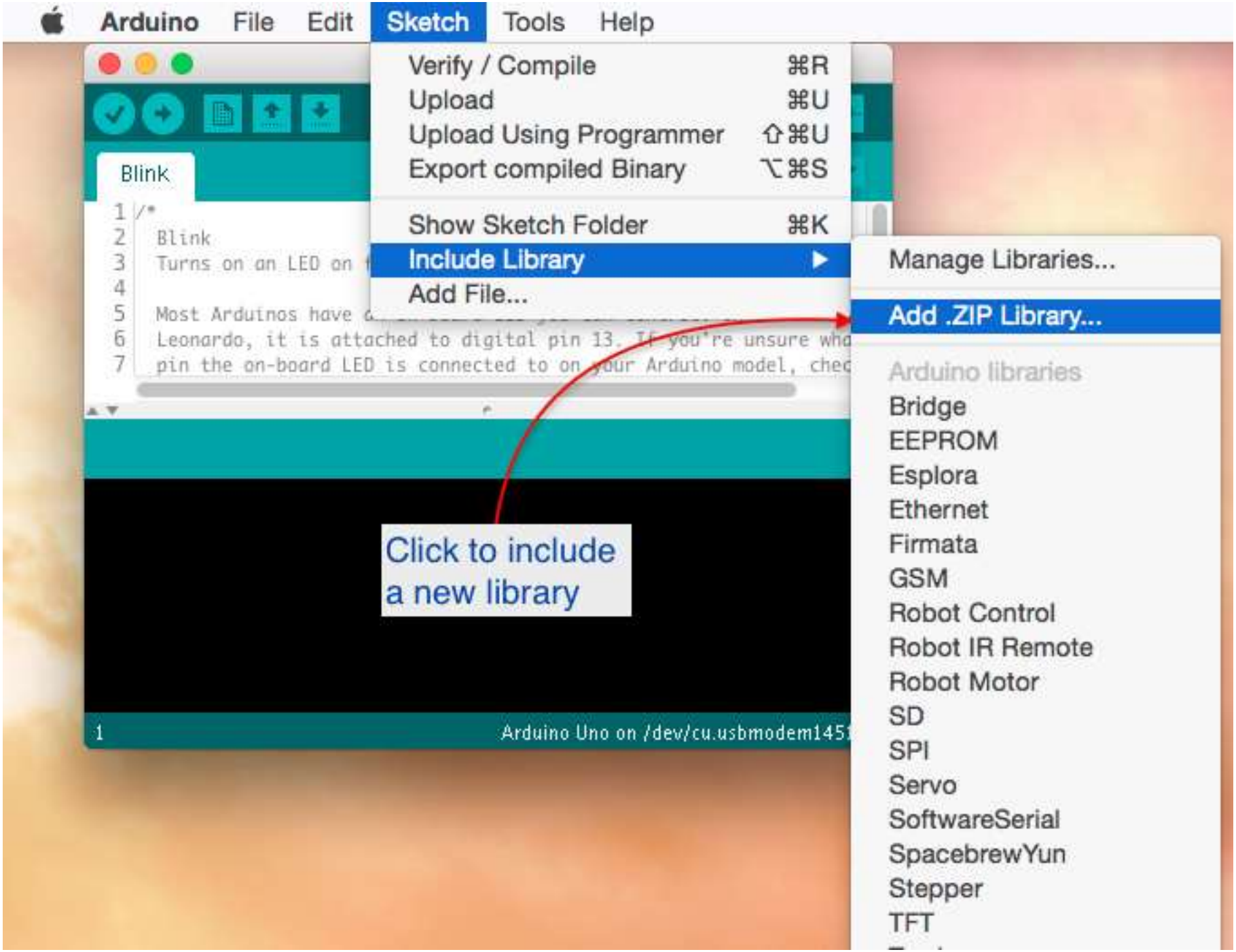
Sketch

```
 */
//
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
  }
}
```

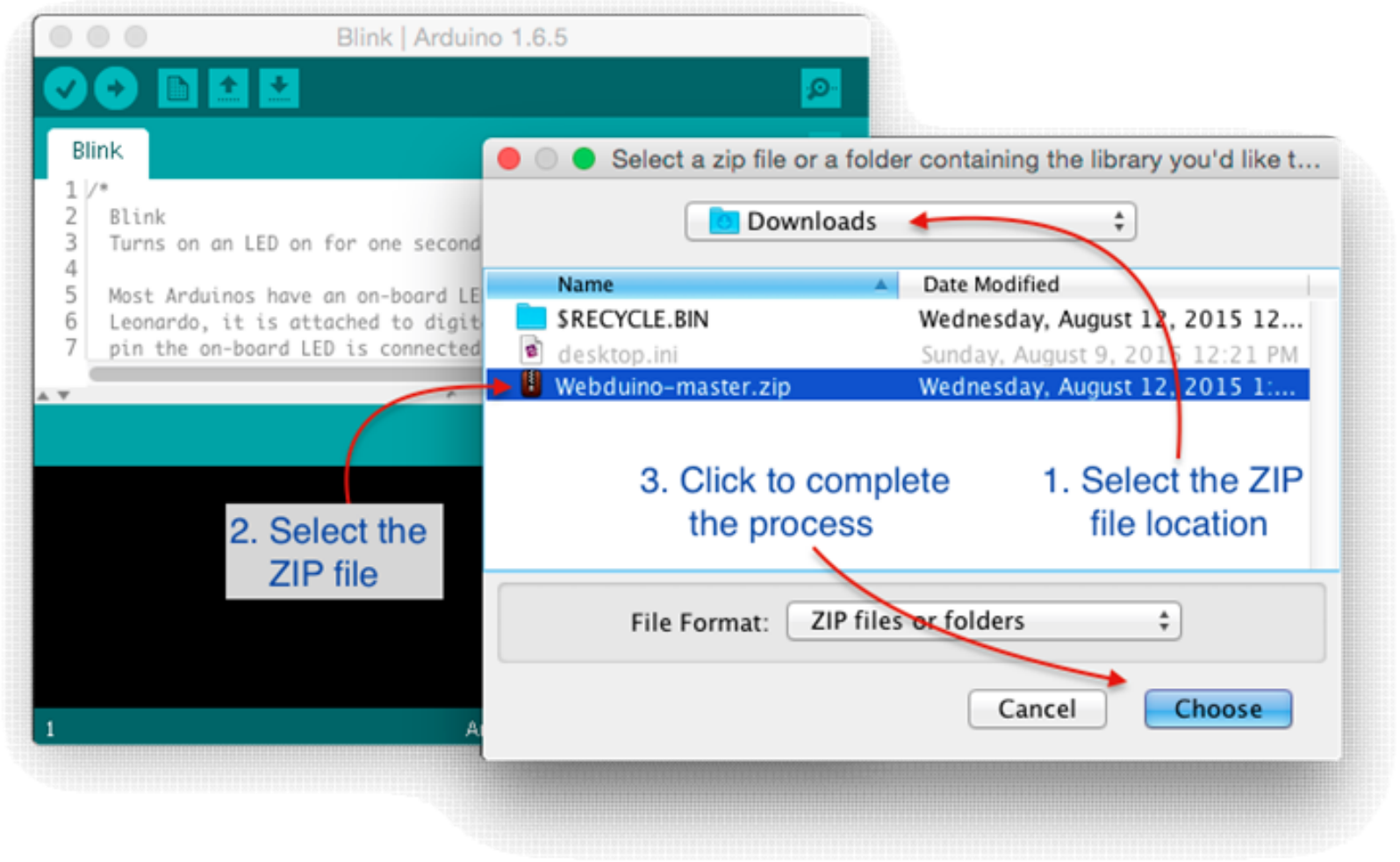
Libraries

Arduino Libraries

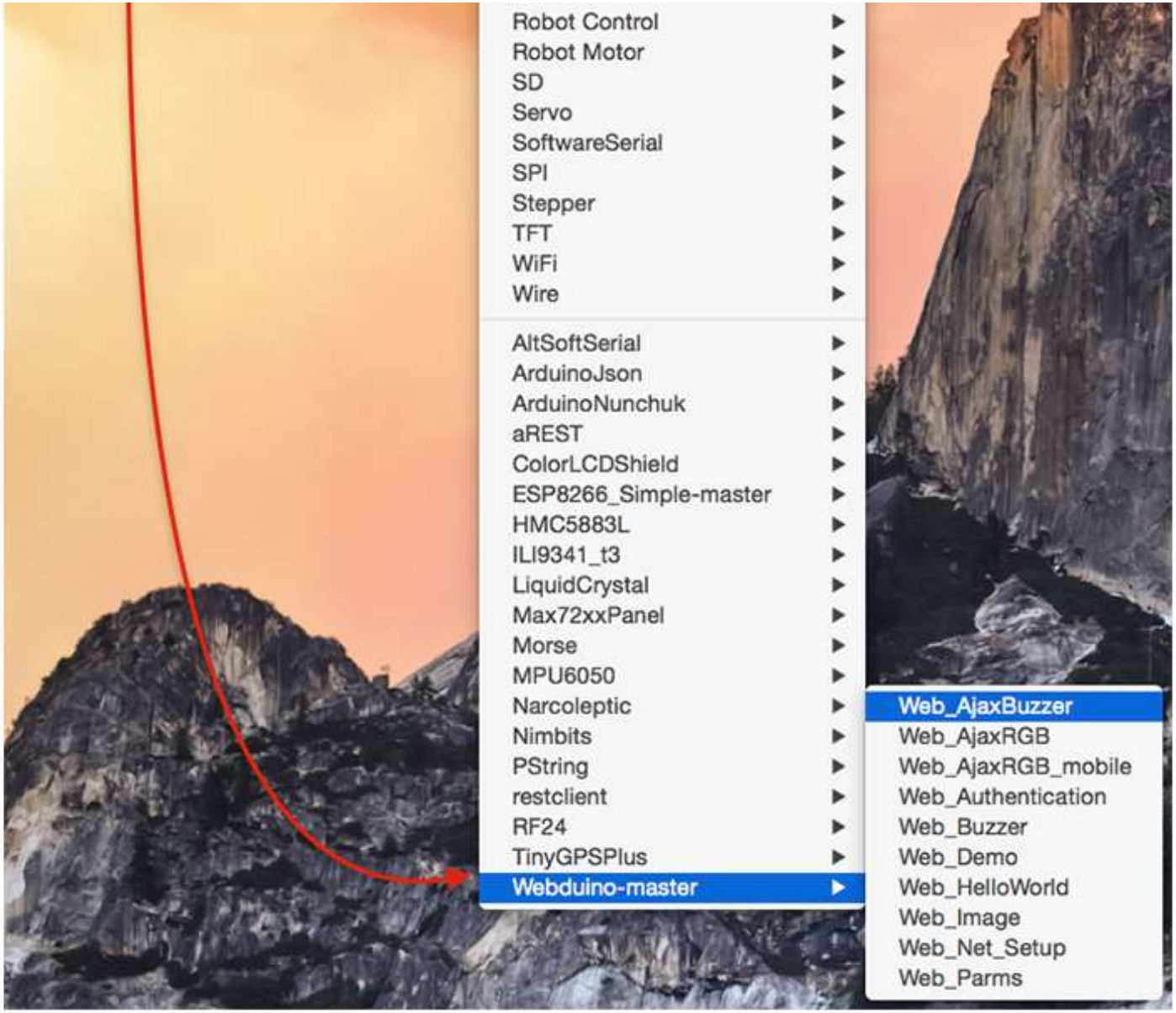
- If there is a library that you need but is not included with the IDE, you can install it. Let's look at an example.
- Download the ZIP file on your computer. It doesn't matter what platform you are on; the libraries work the same regardless of whether you are on Windows, Mac or Linux.
- Also, don't 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:


Arduino
File
Edit
Sketch
Tools
Help
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
Blink
Click to include a new library
Arduino Uno on /dev/cu.usbmodem145

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