- El proyecto Arduino:

Recordando…

- **Pines de entrada**
- **Pines de salida**
- **Microcontrolador**
- **Conexión USB**
- **IDE de arduino**

Escuchas y respuestas con dos valores:

Estado digital	5 voltios	0 voltios
Opción 1	HIGH	LOW
Opción 2	1	0
Opción 3	TRUE	FALSE

pinMode – modos del pin - (OUTPUT, INPUT);

Pines de entrada: escuchan y capturan información del exterior; pulsadores, sensores…

INPUT – el pin se usa para leer si tiene 5V ó 0V

Pines de salida: envían información desde la tarjeta de arduino al exterior.

OUTPUT – activa aplicando 5 voltios o 0V al pin

Microprocesador: procesa el programa cargado a la placa.

digitalWrite(pin,valor) Se usa para activar o desactivar un pin digital.
Entre paréntesis se debe indicar qué pin modificar, y qué valor darle.

Ejemplo: digitalWrite(pin, HIGH);

***Tengamos en cuenta que hasta que se define el estado del pin como HIGH su valor por defecto será LOW.

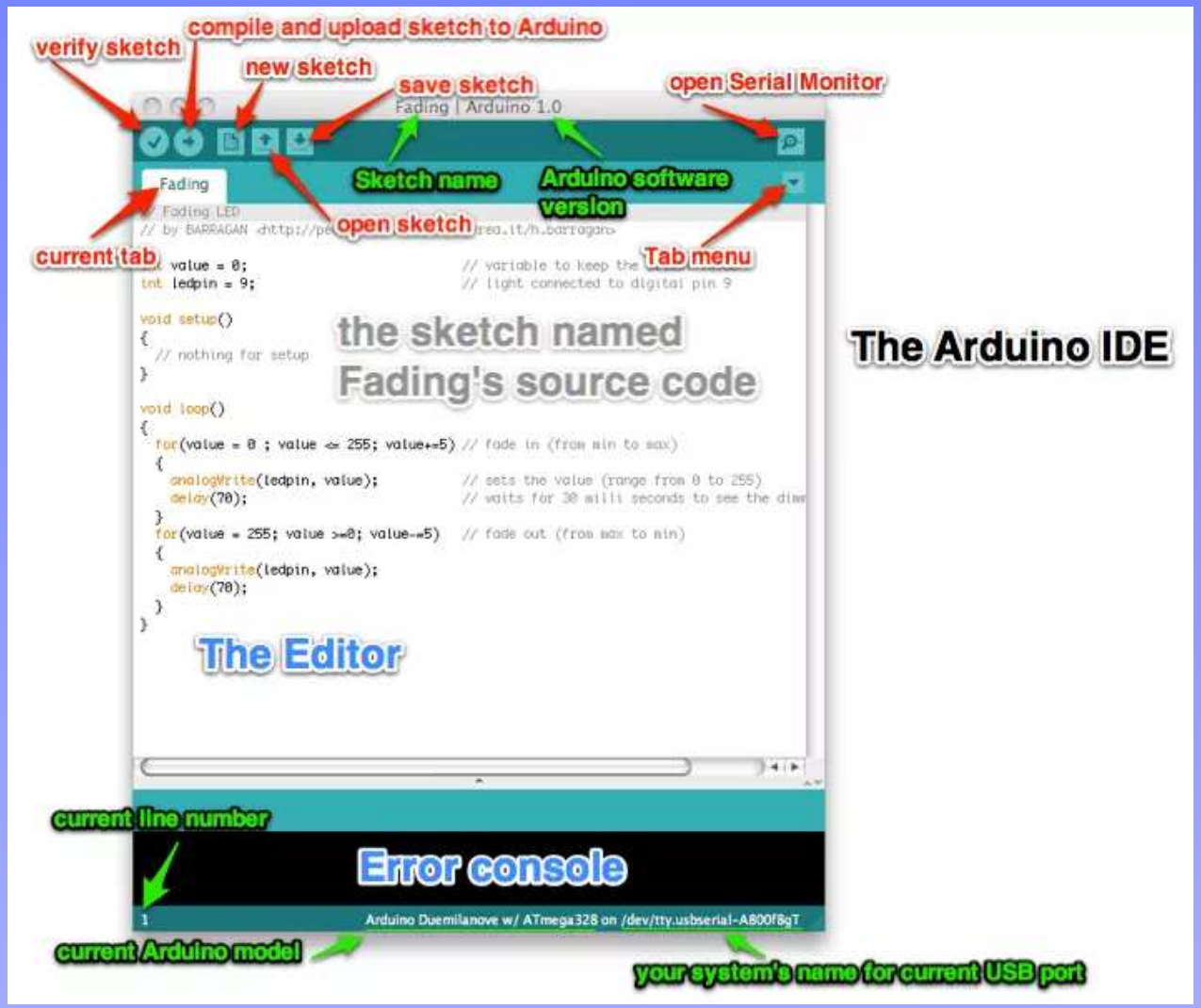
digitalRead(pin); esta instrucción lee el estado o valor de un pin dando HIGH si está a 5V o LOW si hay 0V.

el pin se puede especificar
como una variable
valor = digitalRead (pin)

http://playground.arduino.cc/Arduino NotebookTraduccion/DigitalIO

El siguiente ejemplo lee el estado de un pulsador conectado a una entrada digital y lo escribe en el pin de salida *led*:

```
int led = 13;          // asigna a LED el valor 13
int boton = 7;         // asigna a botón el valor 7
int valor = 0;         // define el valor y le asigna el
                       // valor 0

void setup()
{
pinMode(led, OUTPUT); // configura el led (pin13) como salida
pinMode(boton, INPUT); // configura botón (pin7) como entrada
}

void loop()
{
valor = digitalRead(boton); //lee el estado de la entrada botón
digitalWrite(led, valor);   // envía a la salida 'led' el valor leído
}
```



IDE

Archivos con extensión .ino

Instalación del software y configuración

Instalación del IDE Arduino en Windows:

http://arduino.cc/en/Guide/Windows

Librerías de Arduino:

Las librerías son colecciones de código que facilitan la interconexión de sensores, pantallas, módulos electrónicos, etc. El entorno de arduino incluye estas librerías de manera que se facilite el trabajo. (archivo – ejemplos).

***Existen cientos de librerías desarrolladas por terceros en internet, que nos ayudarán a conectar prácticamente cualquier dispositivo a nuestras tarjetas con arduino.

ENLACES DE INTERÉS:

http://playground.arduino.cc/Es/ArduinoNotebookTraduccion#

http://www.ardumania.es/

https://fabricadigital.org/2015/11/por-que-no-es-buena-idea-conectar-un-led-a-arduino-sin-resistencia/

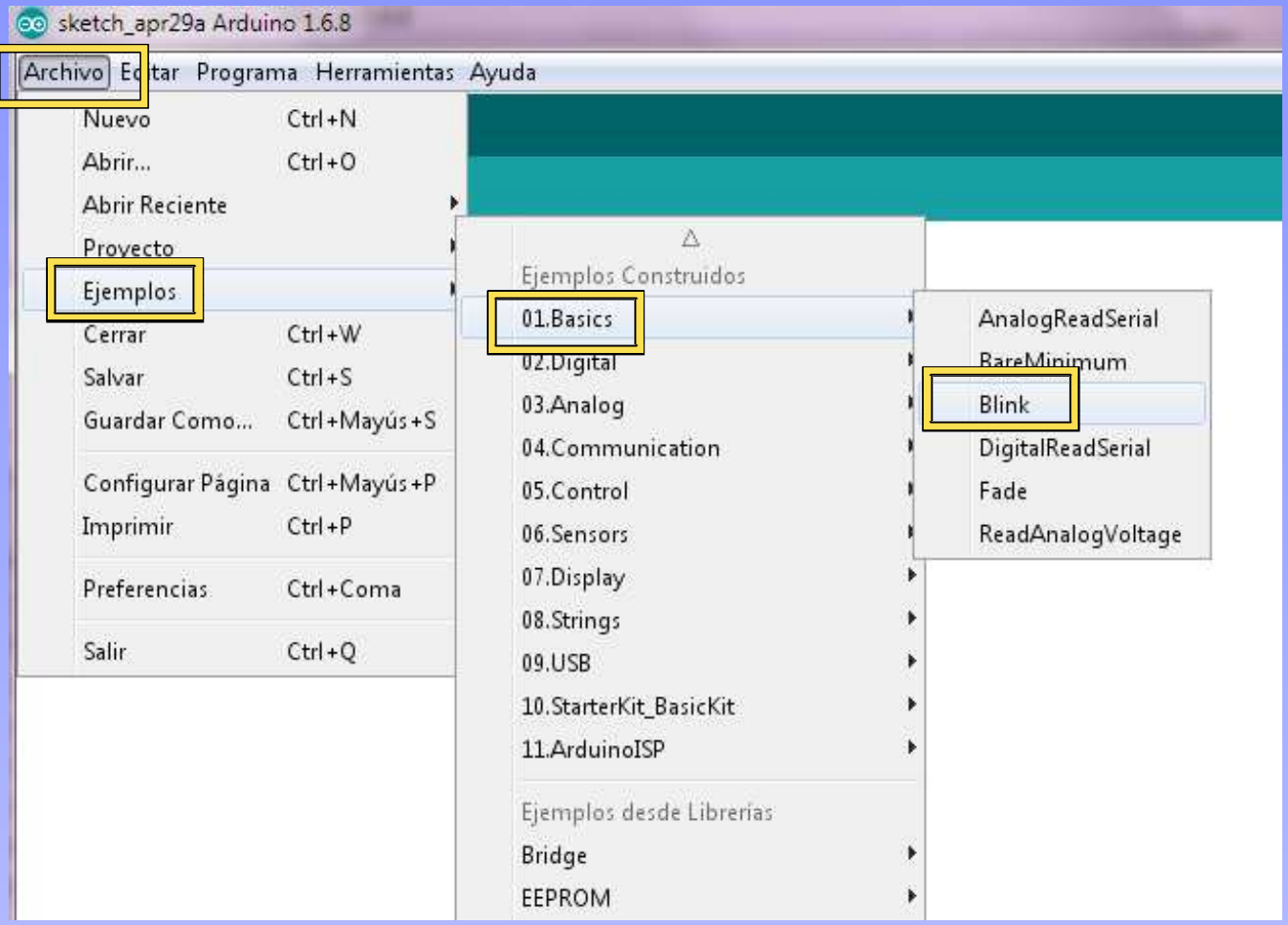
https://www.arduineando.com/tutoriales_arduino/

DESARROLLANDO EJEMPLO BLINK

https://www.arduino.cc/en/Tutorial/Blink

1º Después de indicar en la IDE de Arduino el puerto con el que trabajamos así como el tipo de placa:

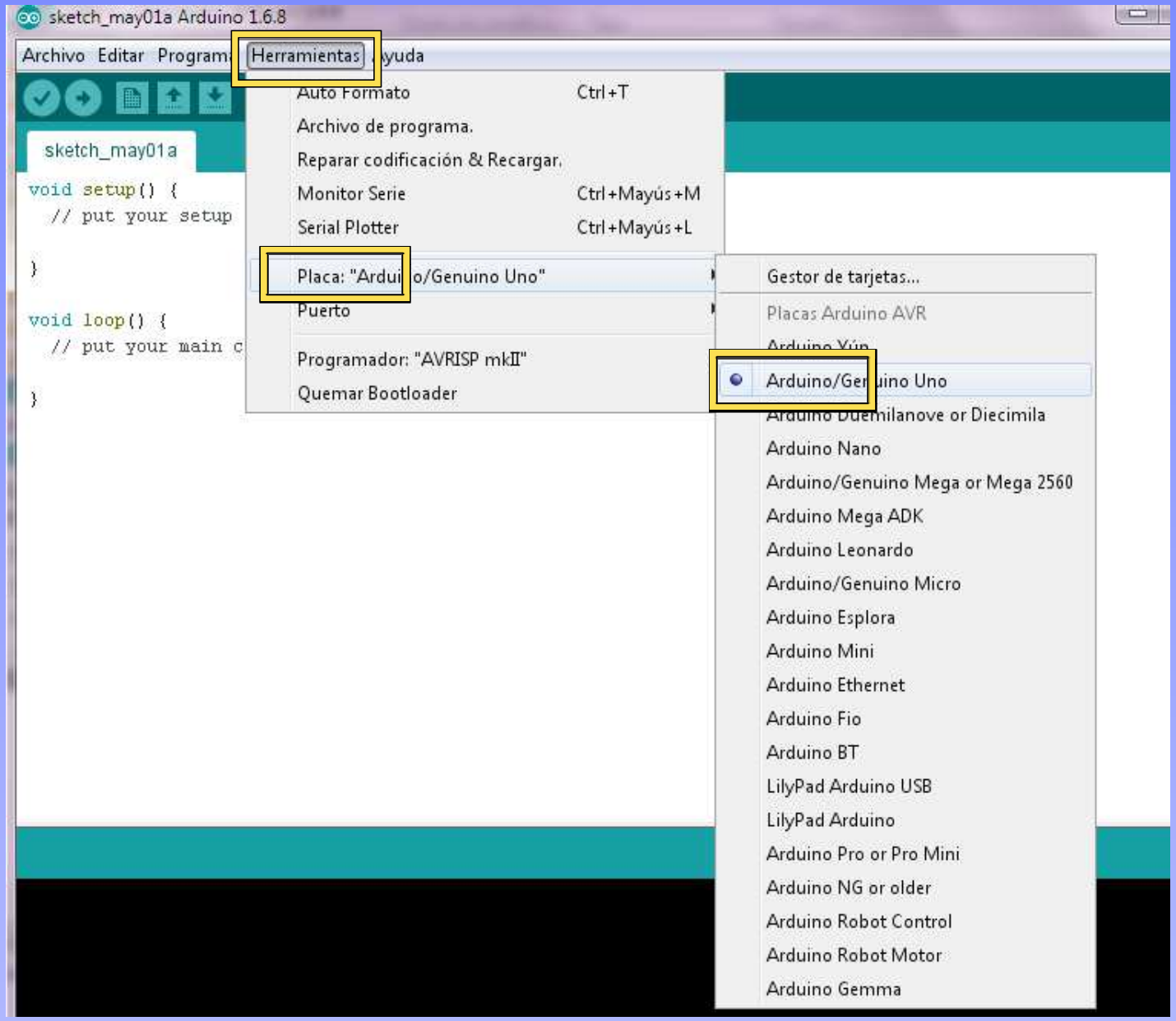
2º



El programa.

```
void setup() {
  // Ponemos el pin 13 en modo salida.
  pinMode(13, OUTPUT);

}

void loop() {
  // Lanzamos 5 Voltios por el pin 13.
  digitalWrite(13, HIGH);
  // Esperamos 1 segundo.
  delay(1000);
  // Dejamos el puerto 13 a 0 V.
  digitalWrite(13, LOW);
  // Esperamos 1 segundo.
  delay(1000);
}
```

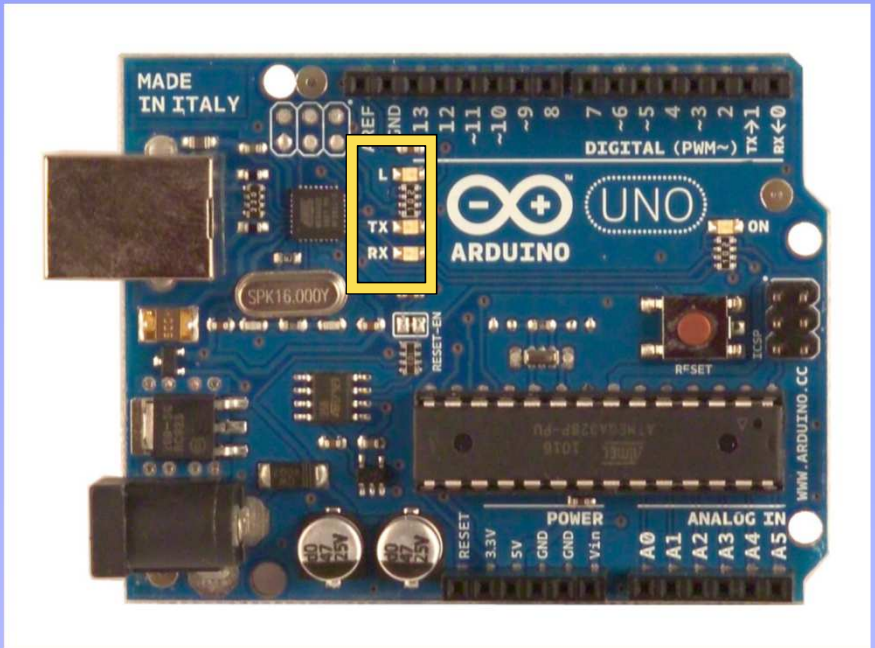
Indicamos el tipo de placa que tenemos para cargarle el código.

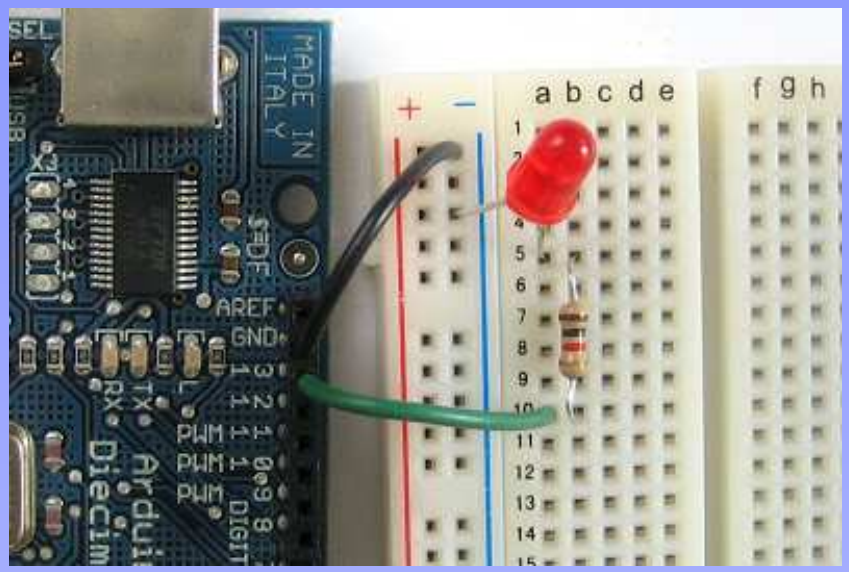


CONECTADO:

Hay 4 LEDs de estado :

ON [verde] indica que la placa está operativa.
L [amarillo] conectado directamente al microcontrolador, accesible a través del pin 13.
RX y TX [amarillo] sirven para indicar que la placa está recibiendo o transmitiendo datos a través de comunicación por el puerto USB.





http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=570

PRÁCTICA EN CLASE: PRIMER EJEMPLO CON BLINK.
EXPERIMENTAMOS CON EL TIEMPO DE ENCENDIDO Y APAGADO DEL LED. - ¿configuramos otro pin?

PRÁCTICA SEMÁFORO CON LEDS

Elementos necesarios

3 LEDS (ROJO, AMARILLO, VERDE)

Los leds son un tipo especial de diodos que emiten luz cuando la corriente eléctrica circula por ellos.

3 RESISTENCIAS 220

Las resistencias son un componente electrónico que actúa como una barrera y deja pasar solo cierta cantidad de electricidad (medida en Ohmios).

Por grupos (los formados para el proyecto final, plantead el circuito en la protoboard y conectad a arduino. Código en la IDE de arduino.

Hemos configurado hasta ahora pines como salida (OUTPUT),

Practicamos con pines como entrada (INPUT);

UN LED SE ENCIENDE CUANDO
CUANDO PULSAMOS UN BOTÓN

-pulsador
-led
-2resistencias

***a tener en cuenta:
¿digitalWrite ó digitalRead?

OTROS MATERIALES:

SENSORES: (para proyectos finales)

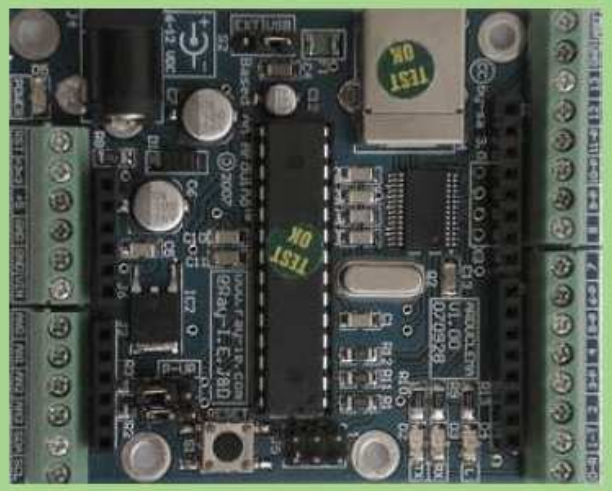
Los sensores convierten las medidas del mundo real en señales electrónicas que podemos utilizar en nuestras placas Arduino.

ACTIVIDAD DE BÚSQUEDA

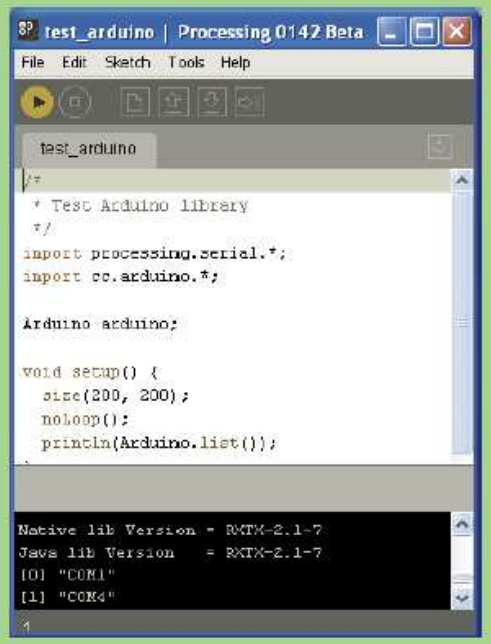
- Sensor de ultrasonidos
- Sensor PIR

COMUNICACIÓN DE ARDUINO CON PC O DISPOSITIVO.

(Interacción PC entorno físico)



+



COMUNICACIÓN

Métodos para controlar Arduino desde el IDE Processing:

Existen dos métodos:

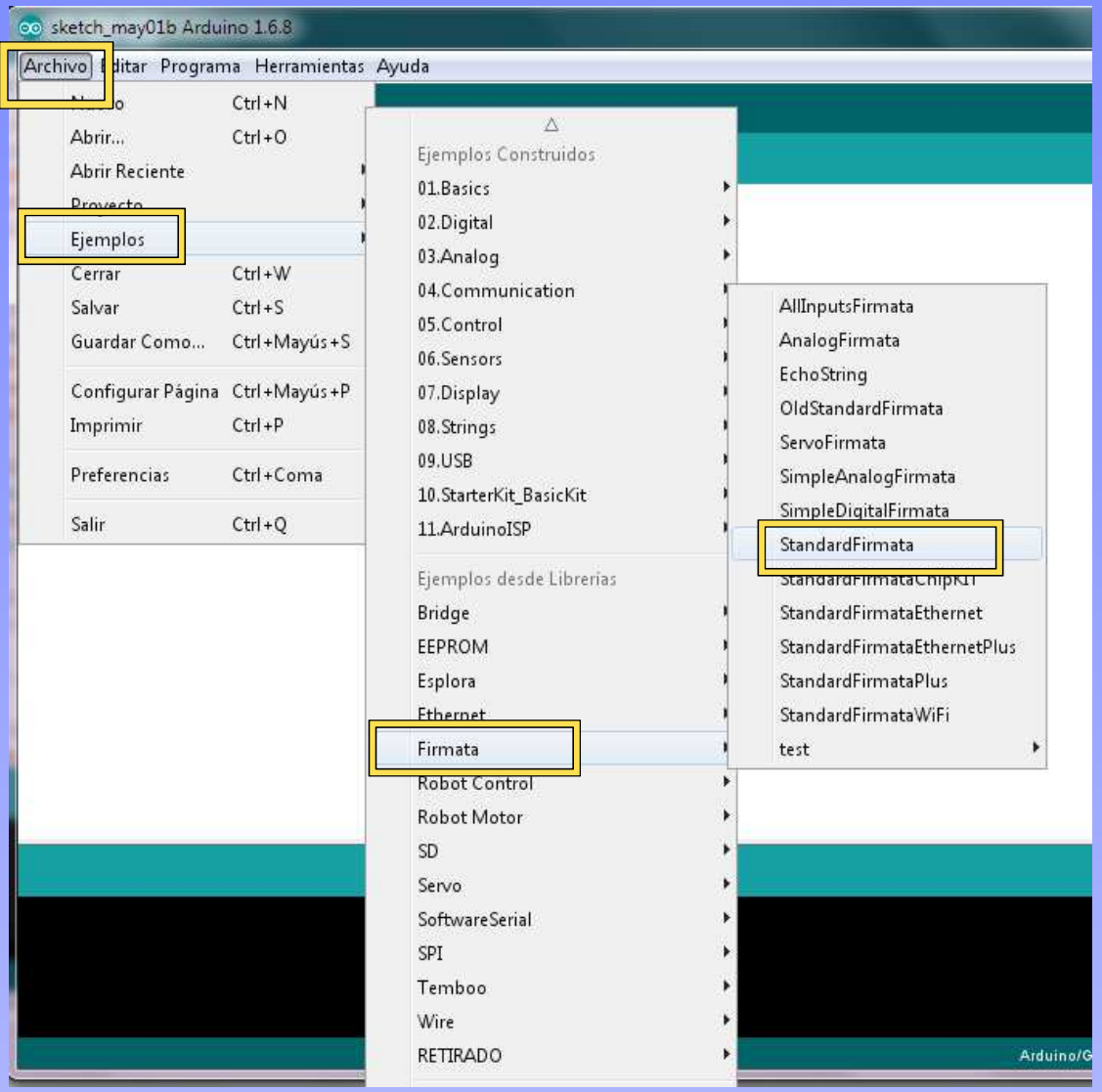
1. Mediante la Librería Arduino para Processing-FIRMATA (StandardFirmata).
2. Mediante la lectura/escritura de datos a través del PUERTO SERIE.

FIRMATA:

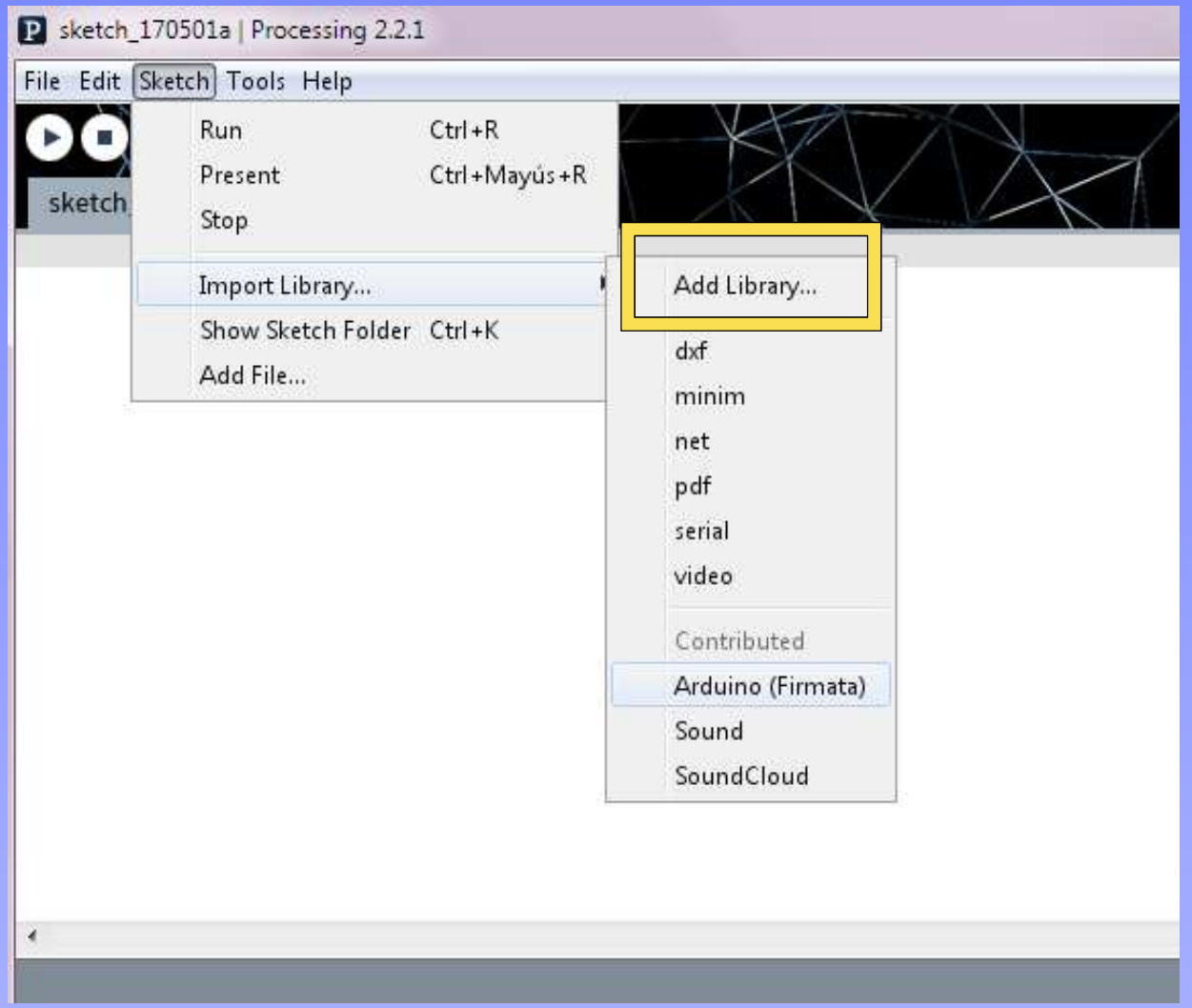
Firmata es un protocolo genérico para la comunicación con microcontroladores desde un software instalado en un ordenador. Este protocolo se puede implementar en cualquier arquitectura de microcontroladores, así como en cualquier paquete de software. Su objetivo es controlar completamente Arduino desde software instalado en un ordenador, sin escribir una sola línea de código de Arduino.

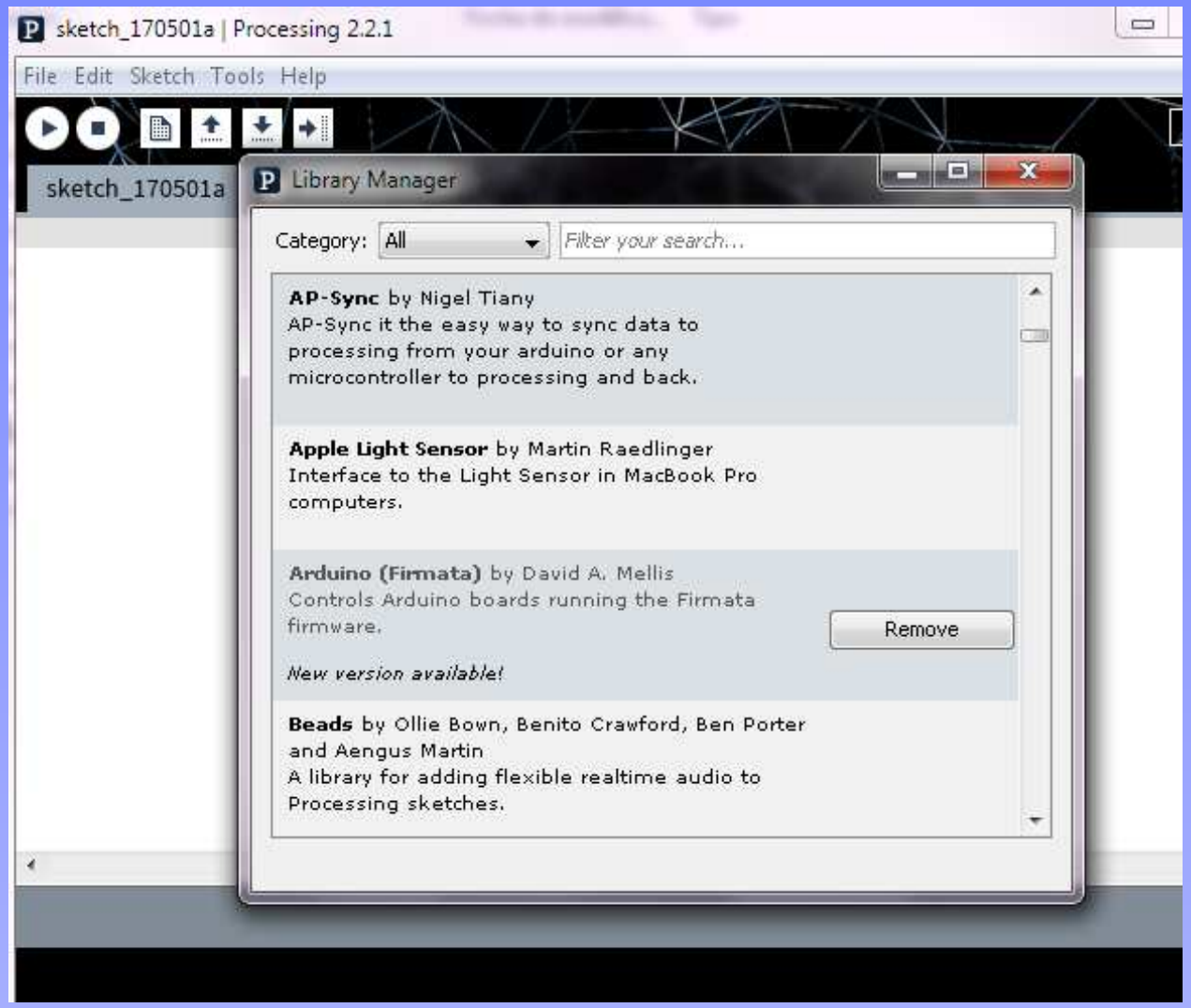
```
import processing.serial.*;

import cc.arduino.*;

Arduino arduino;
```

EN IDE DE ARDUINO: descargar la librería para processing



Abrimos sketch con el protocolo de firmata y lo subimos a nuestra placa. Con ello tenemos preparado el entorno de desarrollo y la placa.


sketch_170501a | Processing 2.2.1
File Edit Sketch Tools Help
sketch
Run Ctrl+R
Present Ctrl+Mayús+R
Stop
Import Library...
Show Sketch Folder Ctrl+K
Add File...
Add Library...
dxf
minim
net
pdf
serial
video
Contributed
Arduino (Firmata)
Sound
SoundCloud


sketch_170501a | Processing 2.2.1
File Edit Sketch Tools Help
sketch_170501a
Library Manager
Category: All
Filter your search...
AP-Sync by Nigel Tiany
AP-Sync it the easy way to sync data to processing from your arduino or any microcontroller to processing and back.
Apple Light Sensor by Martin Raedlinger
Interface to the Light Sensor in MacBook Pro computers.
Arduino (Firmata) by David A. Mellis
Controls Arduino boards running the Firmata firmware.
Remove
New version available!
Beads by Ollie Bown, Benito Crawford, Ben Porter and Aengus Martin
A library for adding flexible realtime audio to Processing sketches.

```
import processing.serial.*;
import cc.arduino.*;
Arduino arduino;

void setup() {
      size(200, 200);



    }
void draw() {
    }
```

```
1 import processing.serial.*;
2 // librería de la comunicación serial
3
4 import cc.arduino.*;
5 // librería de Arduino para Processing
```

Estructura para abrir puerto de comunicación serial (para la transferencia de datos con Arduino)

```
1  Serial Puerto; // declaración del nombre del puerto
2
3  void setup()
4  {
5   println(Serial.list());
6   // Visualiza los puertos serie disponibles en la consola
7   Puerto = new Serial(this, Serial.list()[1],9600);
8   // Hay que cambiar el indice del Serial.list()[1] por el
9   // índice correspondiente al puerto serie que esta conectado
10  // al Arduino. Una vez iniciado el entorno gráfico, en la
11  // parte de abajo de la consola se puede apreciar el índice.
12  // 9600 corresponde a la velocidad de comunicación o BPS.
13 }
```

https://processing.org/tutorials/electronics/

Sintaxis específica para comunicación SERIE:

Serial.begin(rate)

Abre el puerto serie y asigna la tasa de baudios para la transmisión de datos serie. La típica tasa de baudios para comunicarse con el ordenador es 9600 aunque otras velocidades están soportadas.

```
void setup()
{
  Serial.begin(9600);  //abre el puerto serie
                       //ajusta la tasa de datos a 9600 bps
}
```

Nota: Cuando se usa la comunicación serie, los pines digitales 0 (Rx) y 1 (Tx) no pueden ser usados al mismo tiempo.

Serial.println(data)

Imprime datos al puerto serie, seguido de un retorno de carro y avance de línea automáticos. Este comando toma la misma forma que *Serial.print()*, pero es más fácil para leer datos en el *Serial Monitor*[1].

```
Serial.println(analogValue);   //envia el valor de 'analogValue'

//Ejemplo de aplicacion
void setup()
{
  Serial.begin(9600);              //ajusta al serie a 9600 bps
}

void loop()
{
  Serial.println(analogRead(0));  //envia valor analogico
  delay(1000);                    //pausa por 1 segundo
}
```

OS REMITO EL MANUAL DE SUPERVIVENCIA PARA ACLARAR ESTAS FUNCIONES

FUNCIONES SERIAL

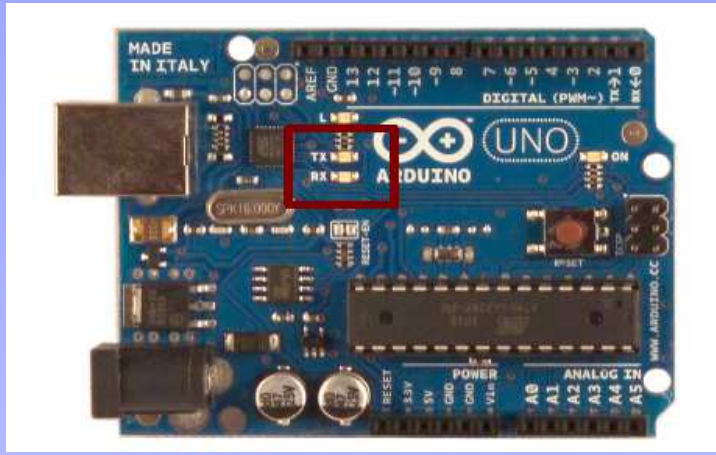
Las placas de Arduino poseen, en principio, un puerto serie para la comunicación con la computadora u otros dispositivos. Tal comunicación se produce en los pines TX y RX.

- La instrucción **Serial.begin()** inicializa el puerto serie y establece la velocidad de comunicación (especificada en baudios).

Fragmento de código

```
Serial.begin(9600); //abre el puerto serie y establece la velocidad de
comunicación en 9600bps
```

Inicializa la comunicación interna del microcontrolador, indicando qué velocidad vamos a trasmitir. Cuanto más grande sea el número, más bits por segundo pasarán por el cable.



SINTAXIS DEL PROGRAMA:

Arduino.list():devuelve una lista con los dispositivos serie (puertos serie) disponibles. Si su tarjeta Arduino está conectada a la computadora cuando usted llama a esta función, su dispositivo estará en la lista.

Arduino(parent, name, rate): crea un "objeto" Arduino (objeto a nivel de elemento de programación). *parent* debe aparecer sn comillas; *name* es el nombre del dispositivo serie (es decir, uno de los nombres devueltos por Arduino.list ()); *rate* es la velocidad de la conexión (57600 para la versión actual del de firmware).

pinMode(pin, mode): *pin* configura un pin digital como entrada (input) o como salida (output) *mode* (Arduino.INPUT o Arduino.OUTPUT).

digitalRead(pin): devuelve el valor leído de una de las entradas digitales, Arduino.LOW o bien Arduino.HIGH (el pin debe estar configurado como entrada).

digitalWrite(pin, value): escribe Arduino.LOW o Arduino.HIGH en un pin digital.

analogRead(pin): devuelve el valor de una entrada analógica leída (de 0 a 1023).

analogWrite(pin, value): escribe un valor analógico (señal tipo PWM) en un pin digital que soporta salida analógica (pines 3, 5, 6, 9, 10, y 11 para ATMEGA 168); valores debes estar comprendidos entre 0 (equivalente a off) y 255 (equivalente a on).

...

COMO CURIOSIDAD:

SIMULADORES ARDUINO

- Fritzing
- 123D circuit.io

https://aprendiendoarduino.wordpress.com/2015/03/24/simulador-arduino/