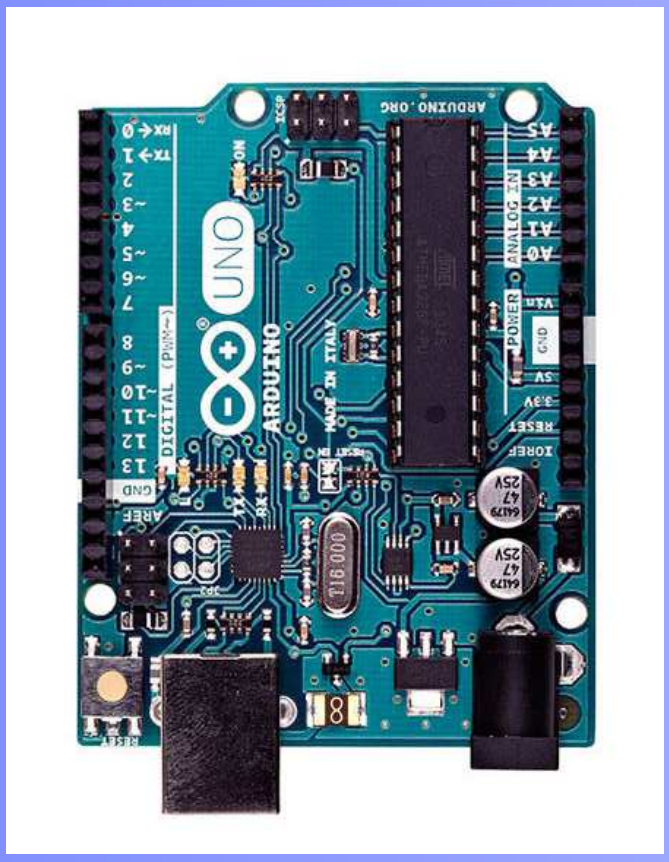
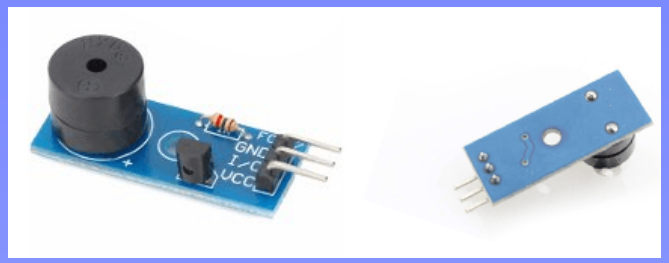

ARDUINO.ORG
UNO
ARDUINO
DIGITAL (PWM~)
ANALOG IN
POWER
16.000

BUZZER O ALTAVOZ



Convierten una señal eléctrica en una onda de sonido.

Buzzer pasivo: no disponen de corriente interna por lo que hay que proporcionar una señal eléctrica para conseguir el sonido deseado. Podemos modificar la señal y con ello variar el tono y generar melodías.

Buzzer activo: disponen de un oscilador interno, con lo que sólo es necesario suministrar corriente al dispositivo.

FUNCIONES DE SONIDO ARDUINO

```
1  tone(pin, frecuencia); //activa un tono de frecuencia determinada en un pin dado
2  noTone(pin);           //detiene el tono en el pin
```

La función tone() también permite especificar la duración del sonido generado.

```
1  tone(pin, frequencia, duracion);  //activa un tono de frecuencia y duracion determ
```

La función tone(); lo que hace es intercambiar valores HIGH/LOW a la frecuencia deseada en el pin seleccionado hasta que detengamos la orden.

Frecuencia de audio

La frecuencia del sonido hace referencia a la cantidad de veces que vibra el aire que transmite ese sonido en un segundo. La unidad de medida de la frecuencia son los Hertzios (Hz). La medición de la onda puede comenzarse en cualquier punto de la misma.

Para que el ser humano pueda oír un determinado sonido su frecuencia debe estar comprendida entre los 20 y los 20.000 Hz.

PRÁCTICA:

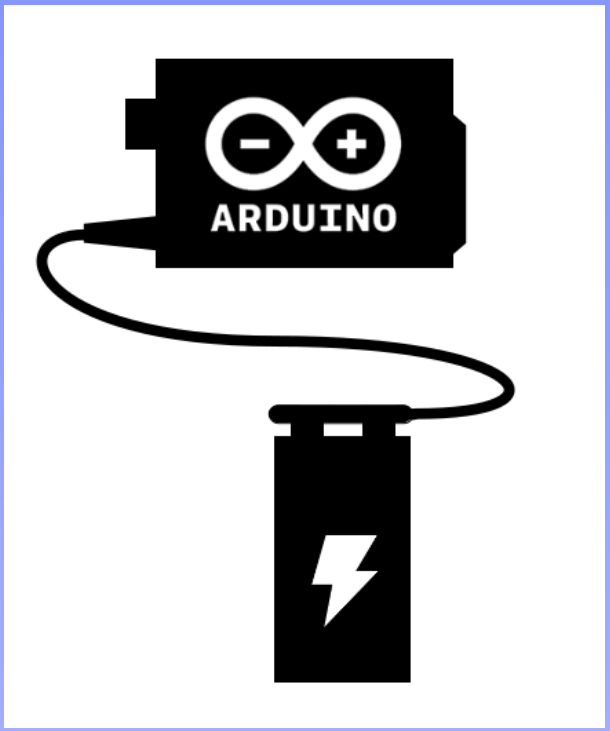
Conectamos un buzzer a arduino.

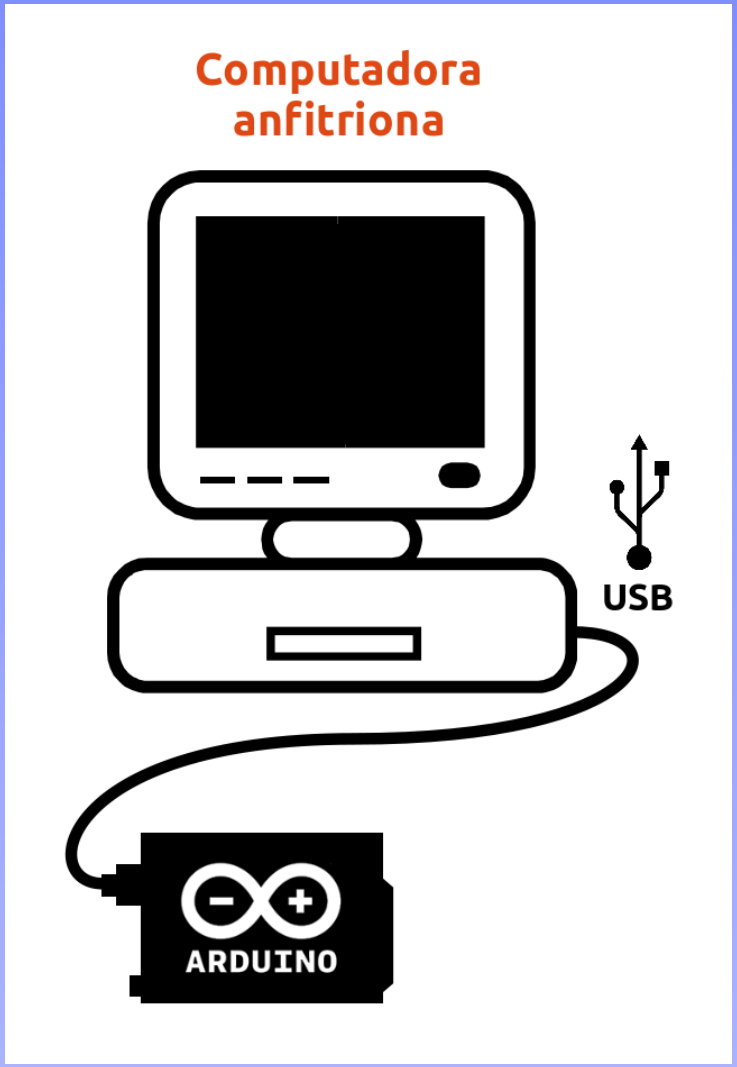
Figura 16: buzzer

- Código y circuito

vemos ejemplo con modulación
en el tono

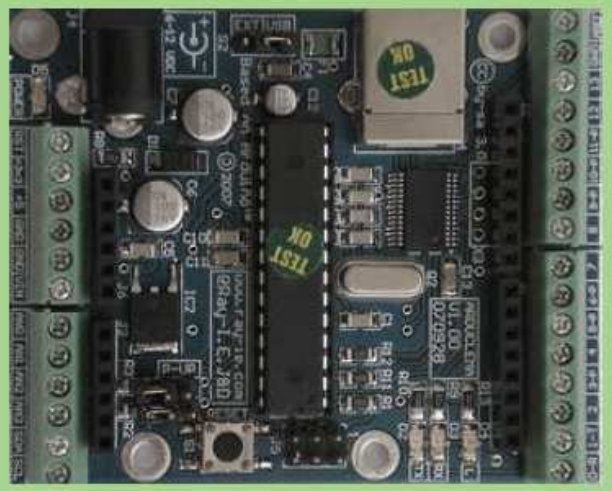
COMUNICACIONES CON EL EXTERIOR



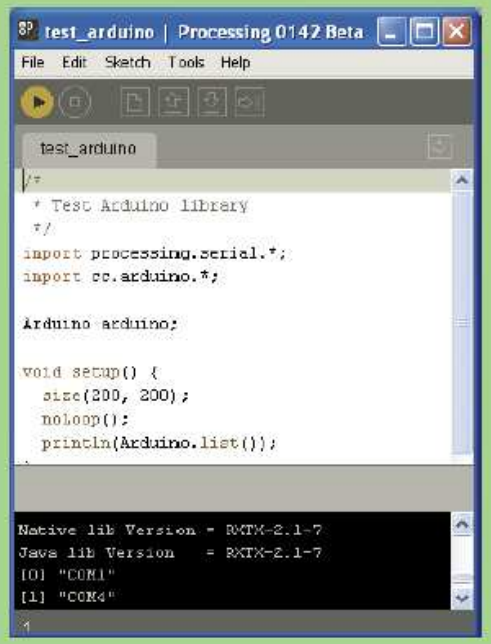


COMUNICACIÓN DE ARDUINO CON PC O DISPOSITIVO.

(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.

COMUNICACIÓN CON FIRMATA:

Firmata es un protocolo genérico para la comunicación con microcontroladores desde un software instalado en un ordenador. Su objetivo es controlar completamente Arduino desde software instalado en un ordenador, sin escribir una sola línea de código de Arduino.

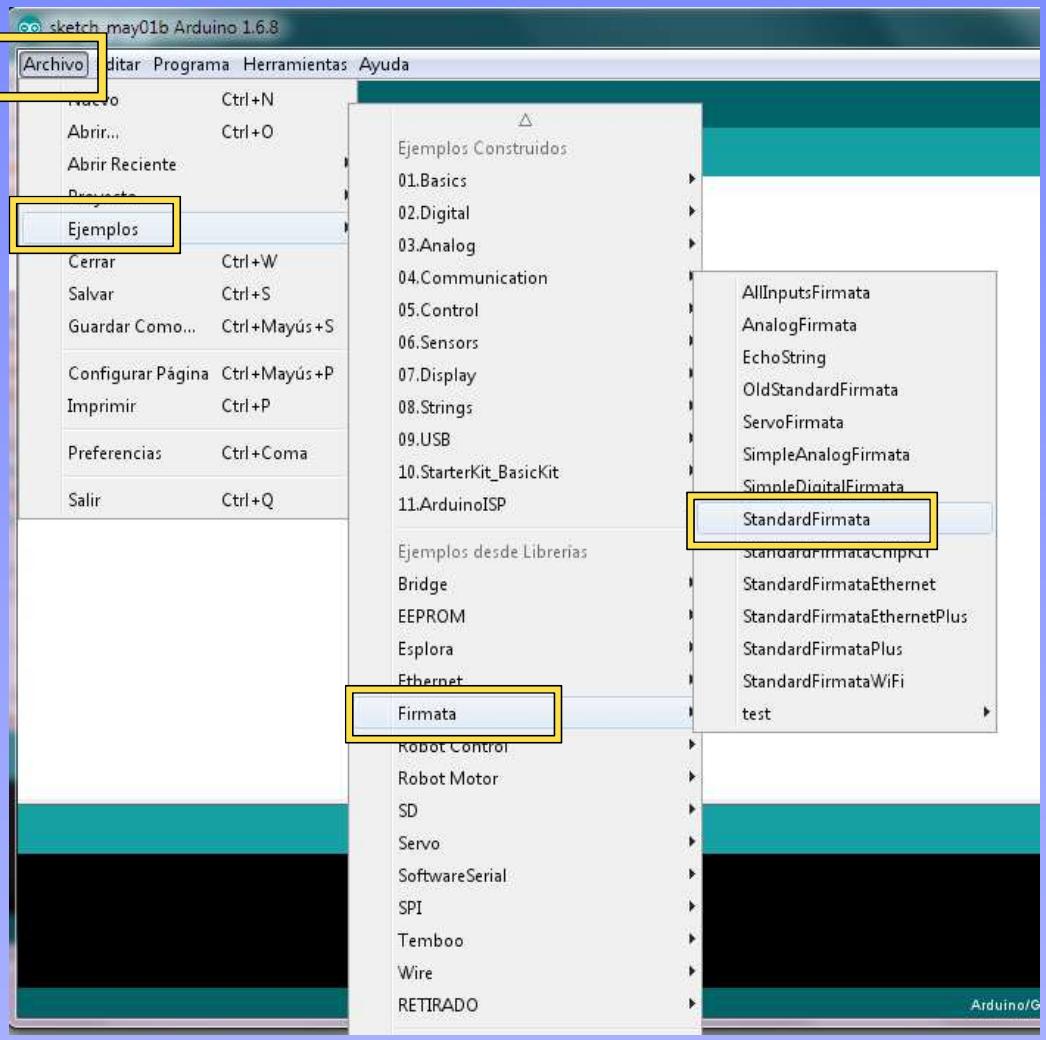
EL MICROCONTROLADOR SE CONVIERTE EN UNA EXTENSIÓN DE NUESTRO ENTORNO DE DESARROLLO.

```
import processing.serial.*;

import cc.arduino.*;

Arduino arduino;
```

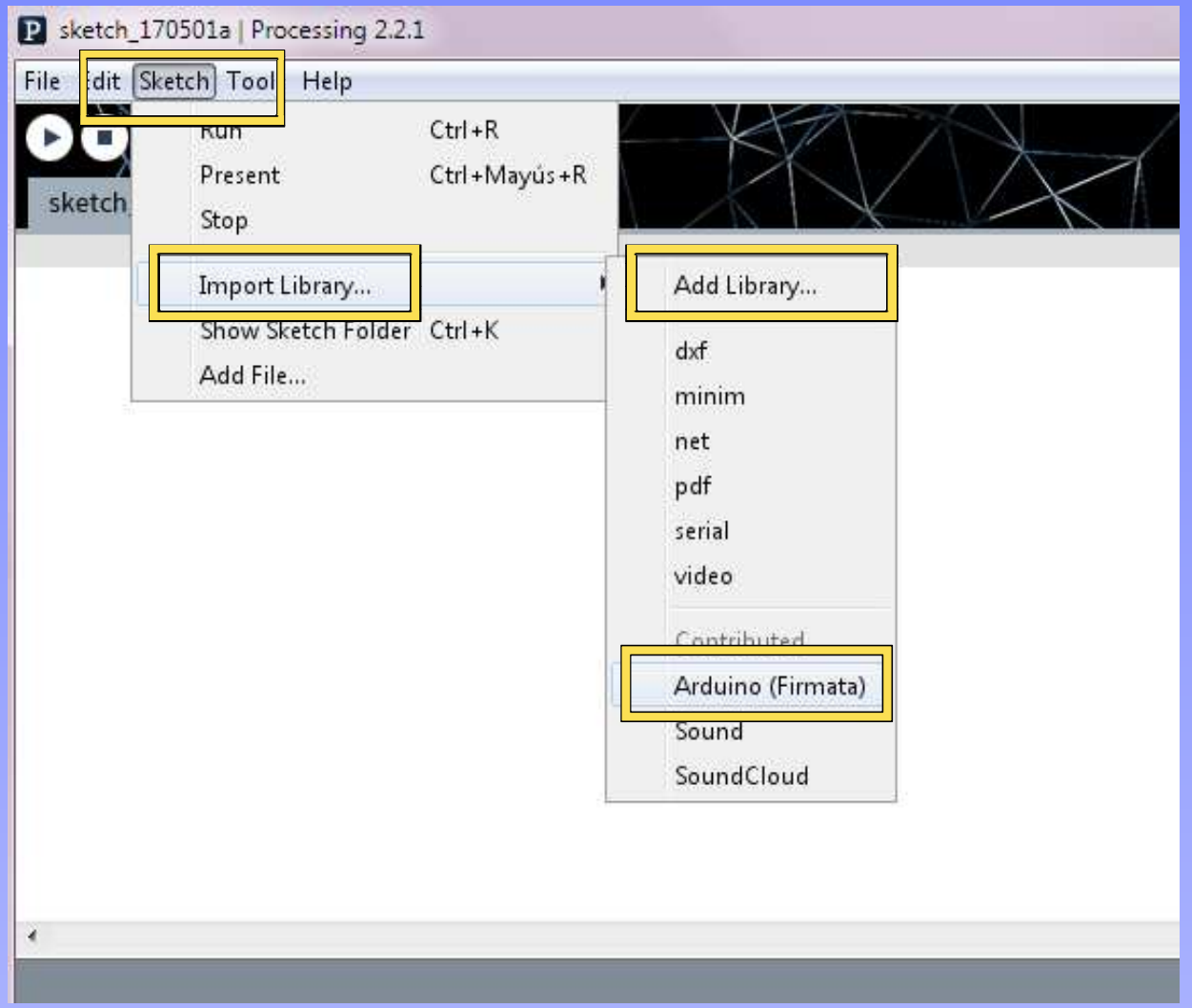
EN IDE DE ARDUINO: descargar la librería para processing



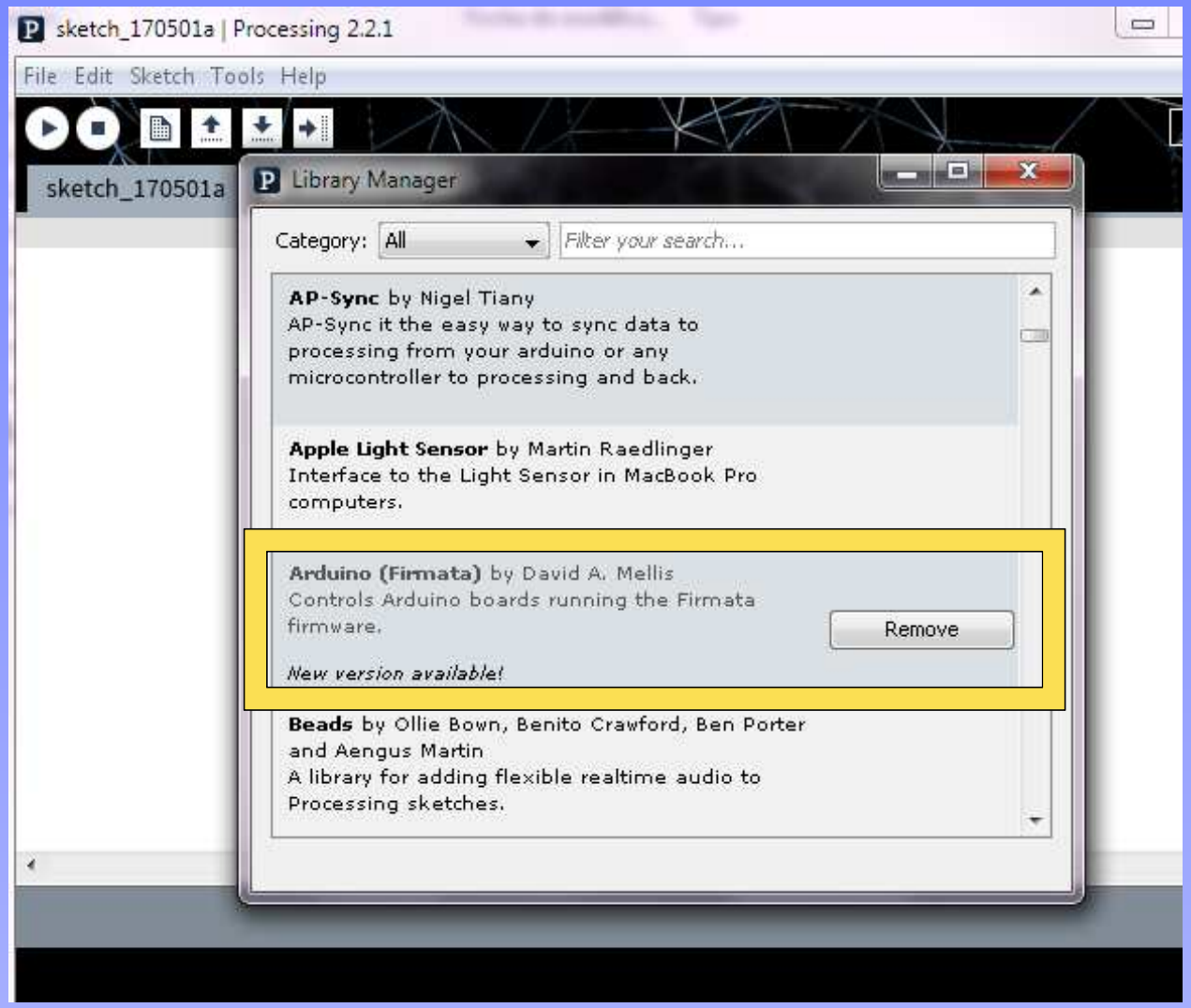
Abrimos sketch con el protocolo de firmata y lo subimos a nuestra placa. Con ello tenemos preparado el entorno de desarrollo y la placa.

http://www.instructables.com/id/Arduino-Installing-Standard-Firmata/

EN LA IDE DE PROCESSING:



- Instalamos la librería de arduino para la comunicación con processing.


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.

Cuando en la IDE de processing importamos librería para comenzar a trabajar con arduino

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

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



    }
void draw() {
    }
```

En este caso processing controla la tarjeta de arduino. Solamente una interfaz está abierta durante todo el proceso y se accede a arduino con el formato, Arduino.funcion();

COMUNICACIÓN POR PUERTO SERIE. SERIAL

- **Necesitamos código en los dos programas para que se comuniquen entre ellos.**

```
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.