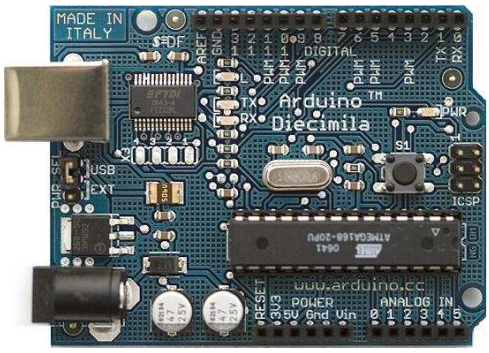
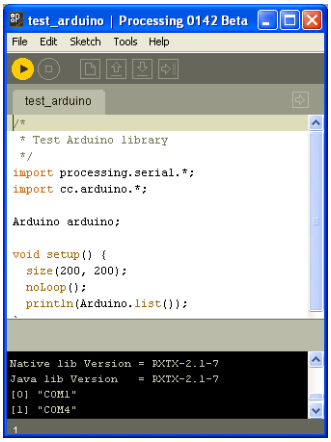
EJERCICIOS DE MANEJO DE ARDUINO DESDE PROCESSING



+



Arduino Processing

José Manuel Ruiz Gutiérrez

INDICE

16. Juego Básico de PinPong
17. Cambio del color de un círculo.
18. Pelota cambiando de tamaño y botando
19. Control de una salida de Arduino mediante el envío a la tarjeta de una letra
20. Mezclador virtual de colores
21. Trazado Grafico de una señal.
22. Enviar valor analógico a Arduino a través del puerto serie.

Nota: *Los materiales recogidos en este documento, listados de código y algunos textos explicativos han sido recogidos en la pagina Web oficial de Arduino (http://www.arduino.cc/es/ y http://www.arduino.cc), correspondiendo al autor de este documento la labor de compilación, traducción e incorporación de imágenes, y esquemas de funcionamiento.*

1. Introducción.

Se trata de realizar aplicaciones en las que el programa que controla y monitoriza los datos de Arduino este en el IDE Processig.

Metodos para controlar Arduino desde el IDE Processing:

Existen dos metodos para controlar Arduino dese processing:

1. Mediante la Librería Arduino para Processing
2. Mediante la lectura/escritura de datos a través del puerto serie.

1.1. Método 1: Librería Arduino para Processing

Antes de nada debemos realizar los siguientes pasos para acondicionar el entorno Processing:

a. CARGAR LA LIBRERIA ARDUINO EN EL IDE DE PROCESSING

No debemos olvidarnos antes de nada de cargar el firmare correspondiente en Arduino.

El fichero de la librería Arduino para Processing esta en el archivo processing-arduino o arduino-processing-e231 que se encuentra en la página de arduino. Dentro de ellos hay una carpeta que se llama Arduino y contiene la librería. *(\processing-arduino\arduino* o *\arduino-processing-e231\arduino*).

http://www.arduino.cc/playground/uploads/Interfacing/processing-arduino.zip

Para que Processing pueda trabajar con la librería de Arduino debemos incluir la carpeta **Arduino** dentro de la carpeta librerías del IDE Processing:

\processing-0138\libraries

b. CONFIGURAR ARDUINO PARA QUE PROCESSING PUEDA DIALOGAR CON EL.

Para cargar el firmware en Arduino nos vamos a la librería processign-arduino y en la carpeta

\arduino\firmware\Standard_Firmata
se encuentra el fichero
Standard_Firmata.pde

que debemos cargar con el IDE Arduino y lo debemos descargar sobre Arduino. En este fichero están las funciones que luego se utilizaran desde el ID Processing para poder dialogar con Arduino.

Instrucciones para instalarla

1. Descompactar la librería y copia la carpeta "arduino" en la carpeta "libraries" perteneciente al directorio de la aplicación Processing.
2. Abrir el firmware (en arduino/firmware) en Arduino y descargar este en la tarjeta Arduino.
3. Configurar Processing para serial: http://processing.org/reference/libraries/serial/
4. En Processing, abrir uno de los ejemplos que viene con la librería Arduino.
5. Modificar el código de ejemplo seleccionando el puerto serie correcto.
6. Ejecute el ejemplo.

Referencia de las funciones de la librería

Las siguientes funciones se encuentran en la Librería Arduino para Processing y comunican (a partir de Processing) con un Arduino, una vez que el Firmata se ha instalado en la tarjeta

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).

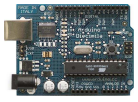
digitalRead(pin): devuelve el valor leído de una de las entradas digitales, Arduino.LOW o bien Arduino.HIGH (el pin debe estar configurado como entrada).

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).

1.2 Método 2: Mediante intercambio de datos a traves del pueerto serie



Se puede controlar Arduino desde Processing sin necesidad de incluir la libraria Arduino en processing, en este caso se trata de recoger datos del puerto que la tarjeta Arduino envia al puerto serie.

Procedimiento:

1.- Se carga en la tarjeta Arduino el programa que se encargue de escribir en el puerto el dato que después leera Procesing y lo incorporara en el programa que este ejecutando.

2.- Cargar y ejecutar el programa en el IDE Processing que recogerá los datos que Arduino le envía por el puerto serie.

Ejercicios prácticos utilizando la librería Arduino para processing (Método 1).

2. Test de Funcionamiento de Arduino

Con este sencillo ejemplo se trata de comprobar si Procesing se comunica con Arduino.

El programa lo que hace es imprimir en la ventana de datos los puertos detectados del ordenador y los numera dentro de un array

Los puertos detectados son:
[0] “COM1”
[1] “COM4”

En este caso la tarjeta Arduino estaba colocada en el puerto USB COM4

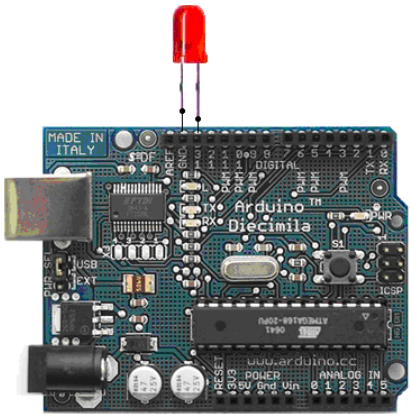
Solución

```
/*
 * Test de Puertos utilizados
 */
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
```

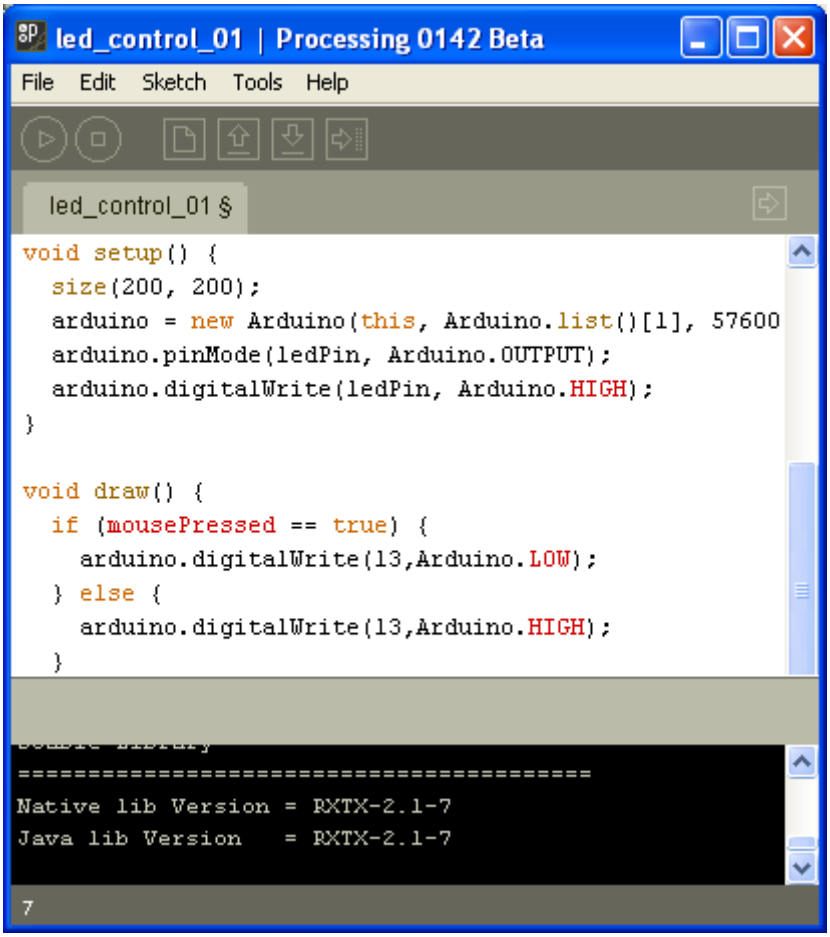
```
void setup() {
  size(200, 200);
  noLoop(); // Fuerza a no realizar el bucle del programa
  println(Arduino.list()); // Imprime la lista de puertos COM
}
void draw() {
}
```

3. Gobierno de una salida de Arduino desde Processing.

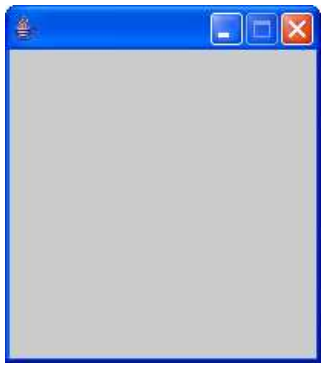
Con este ejemplo vamos a encender y apagar un diodo led desde el entorno de processing pulsando el botón izquierdo del ratón estando en la pantalla activa de Processing en ejecución.



Esquema de montaje



Entorno Processing con el programa cargado



Ventana de interacción sobre la que debemos hacer clic con el ratón para activar y desactivar el LED conectado en el PIN13

Tenemos varias formas de escribir el programa :

Solución 1

```
/*
 * enciende el led cuando se presiona el botón del ratón
 * apaga cuando levantamos el botón
 */

import processing.serial.*;  //Importamos las librerías necesarias
import cc.arduino.*;

Arduino arduino;  // Crea el objeto Arduino
int ledPin = 13;  // Designa el numero de PIN para el LED

void setup() {  //Configura el puerto y las señales con las que va a trabajar
  size(200, 200);
  arduino = new Arduino(this, Arduino.list()[1], 57600);  // Configura el puerto como [1]
  arduino.pinMode(ledPin, Arduino.OUTPUT); // Configura el PIN13 como salida
  arduino.digitalWrite(ledPin, Arduino.HIGH); //Enciende el LED
}

void draw() { //Dibuja una ventana de interacción
  if (mousePressed == true) {  //pregunta si se ha pulsado el botón del ratón
    arduino.digitalWrite(13,Arduino.LOW); // Si se ha pulsado apaga el LED
  } else {
    arduino.digitalWrite(13,Arduino.HIGH); // Si no esta pulsado enciende el LED
  }
}
```

Solución 2

```
/*
 * Raton presionado -> LED on
```

```
 * Raton sin presionar-> LED off
 */
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin = 13;

void setup() {
  size(200, 200);
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  arduino.pinMode(ledPin, Arduino.OUTPUT);
  arduino.digitalWrite(ledPin, Arduino.HIGH);
}

void draw() {}

void mousePressed() {  // Procedimiento para botón pulsado
  arduino.digitalWrite(ledPin, Arduino.LOW);
}
void mouseReleased() { //Procedimiento para botón levantado
  arduino.digitalWrite(ledPin, Arduino.HIGH);
}
```

Solución 3

```
/*
 * El led se enciende y apaga al cambiar el estado del ratón (conmutador on/off)
 */
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin = 13;
boolean isLedOn = false;
void setup() {
  size(200, 200);
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  arduino.pinMode(ledPin, Arduino.OUTPUT);
  arduino.digitalWrite(ledPin, Arduino.HIGH);
}

void draw() {}

void mousePressed() {  //Procedimiento para testear el estado del ratón
  if(isLedOn) {
    arduino.digitalWrite(ledPin, Arduino.HIGH);
```

```
  } else {
    arduino.digitalWrite(ledPin, Arduino.LOW);
  }
  isLedOn = !isLedOn;
}
```

Solución 4

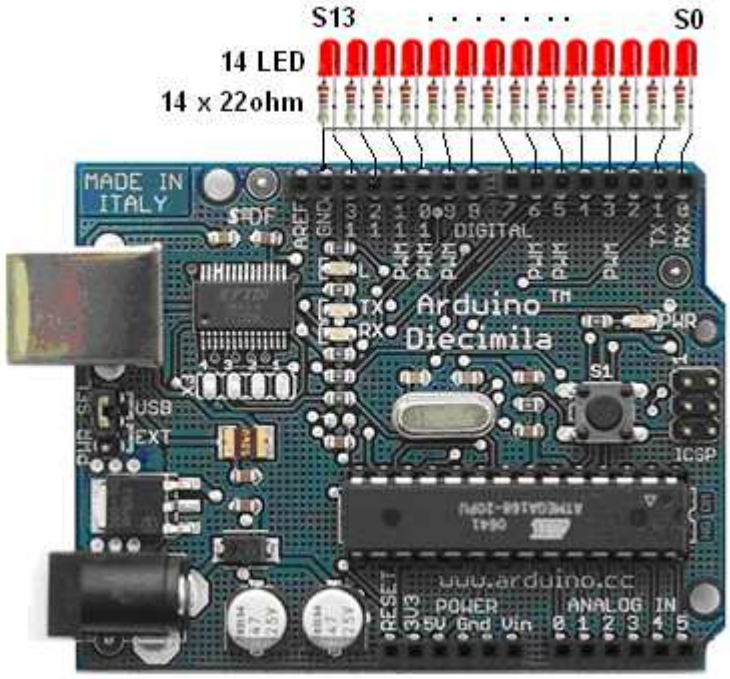
Al pulsar con el ratón el led se pone intermitente y al pulsar de nuevo se apaga (conmutador)

```
/*
 * Al pulsar con el ratón de forma conmutada el LED parpadea
 */
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin = 13;
boolean blinkLed = false; //Variable que indica si el LED esta parpadeando

void setup() {
  size(200, 200);
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  arduino.pinMode(ledPin, Arduino.OUTPUT);
  arduino.digitalWrite(ledPin, Arduino.LOW);
}

void draw() {
  if(blinkLed) {
    arduino.digitalWrite(ledPin, Arduino.LOW);
    delay(50);
    arduino.digitalWrite(ledPin, Arduino.HIGH);
    delay(50);
  } else {
    arduino.digitalWrite(ledPin, Arduino.LOW);
  }
}
void mousePressed() {  // Detecta si el ratón esta pulsado
  blinkLed = !blinkLed;
}
```

4. Activación de múltiples salidas.

En este ejemplo se trata de establecer una pantalla con tantos cuadrados(que representan pulsadores) como salidas tiene Arduino (0 a 13) y bastará con pulsar con el ratón en el botón correspondiente para activar la salida correspondiente.

Esquema de conexionado.



Aspecto de la Pantalla de ejecución

En la figura vemos que esta activado un botón. Se trata de la salida digital numero 13

Solución

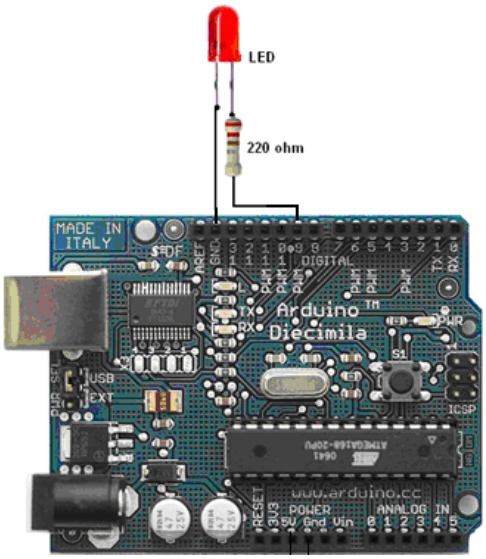
```
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;

color off = color(4, 79, 111); // Definimos los colores del botón en estado off
color on = color(84, 145, 158); // Definimos los colores del botón en estado on
// designamos en un array de tamaño 13 el estado de cada una de las entradas
int[] values = { Arduino.LOW, Arduino.LOW, Arduino.LOW, Arduino.LOW,
 Arduino.LOW, Arduino.LOW, Arduino.LOW, Arduino.LOW, Arduino.LOW,
 Arduino.LOW, Arduino.LOW, Arduino.LOW, Arduino.LOW, Arduino.LOW };

void setup() {
  size(470, 200);
   println(Arduino.list()); // Mostramos los puertos detectados
  arduino = new Arduino(this, Arduino.list()[1], 57600);
   for (int i = 0; i <= 13; i++)   // Configuramos los pines 0 al 13 como salidas
  arduino.pinMode(i, Arduino.OUTPUT);
}

void draw() {
  background(off);
  stroke(on);
  for (int i = 0; i <= 13; i++) {  // se testea el estado del array de estados de salidas
    if (values[i] == Arduino.HIGH)
      fill(on); // se pone el color de on si esta en estado alto
    else
      fill(off);  // se pone el color de off si esta en estado bajo
      rect(420 - i * 30, 30, 20, 20);
  }
}

// Procedimiento para detectar si el ratón se ha pulsado sobe uno de los botones
void mousePressed()
{
  int pin = (450 - mouseX) / 30;
  if (values[pin] == Arduino.LOW) {
    arduino.digitalWrite(pin, Arduino.HIGH);
    values[pin] = Arduino.HIGH;
  } else {
    arduino.digitalWrite(pin, Arduino.LOW);
    values[pin] = Arduino.LOW;
  }
}
```

5. Encendido y apagado de un led de manera paulatina (efecto fading).

Con este ejemplo se trata de realizar el encendido y apagado gradual de un LED que conectamos al PIN9 que en este caso actuará como salida analógica enviando a el un valor que oscilará entre 0-255 (encendido) y entre 255-0 (apagado) de manera cíclica.

En la ventana de datos de Processig se mostrara el valor que se este enviando

Esquema de conexionado



Conexión de un LED al pin 9

Aspecto de la ventana de terminal del IDE de Processing

Solución

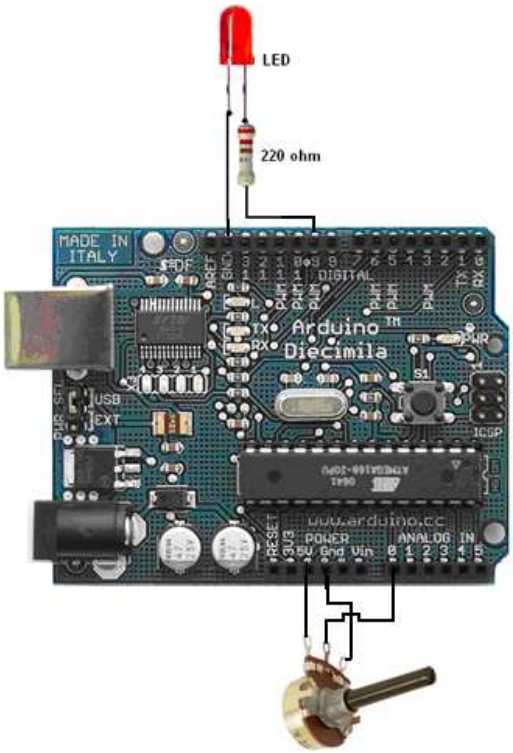
```
/*
 * encendido y apagado gradual de un LED
 */
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int pwm=0;  //Variable nivel de iluminación del LED
int ledPin=9;
boolean rising=true;  // indica si el encendido es hacia arriba de 0-255

void setup() {
  size(200, 200);
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  arduino.pinMode(ledPin, Arduino.OUTPUT);
  arduino.analogWrite(ledPin, pwm);
}

void draw() {
  arduino.analogWrite(ledPin, pwm); // Escribe el valor pwm en la salida PIN9 del LED
  println(pwm);  // Escribe en la ventana de datos el valor de la variable pwm
   if(rising) {  // contador ascendente hasta llegar a 255
    pwm+=2;
    if(pwm>=255) {
      rising=false;
    }
   }else {
     pwm-=2; // contador descendente hasta llegar a 0
     if(pwm<=0) {
      rising=true;
    }
   }
}
```

6. Control del nivel de iluminación de un LED mediante un potenciómetro.

En este caso se trata de conseguir variar el grado de intensidad de un LED haciendo uso de un potenciómetro conectado en un canal de entrada analógica de Arduino. De la misma manera queremos monitorizar el nivel de iluminación en la ventana activa de Procesing haciendo que su color cambie en función del nivel de la señal de entrada que generemos con el potenciómetro.

LED colocado en el PIN9
Potenciómetro colocado en el (entrada analógica 0)

Esquema de conexionado



Solución

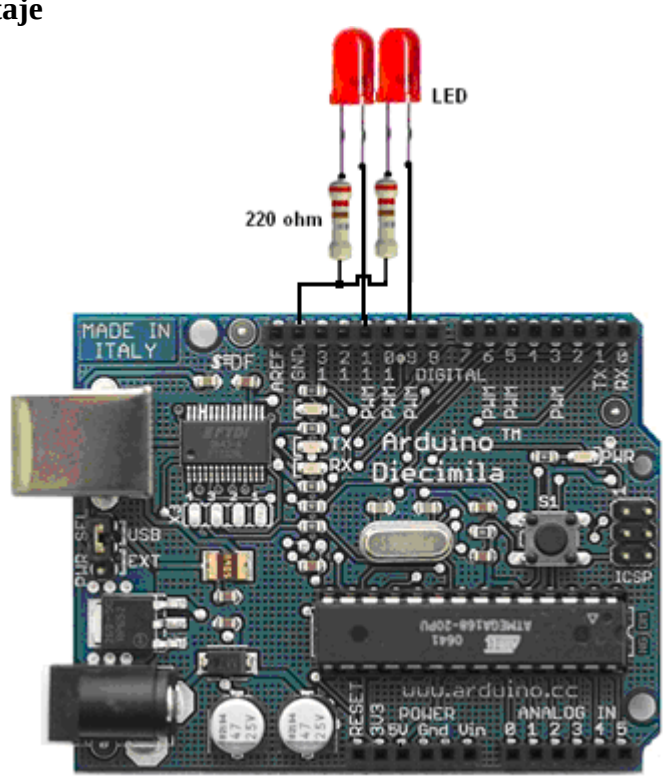
```
/*
 * Gobierno del nivel de encendido de un led mediante un potenciómetro
 */
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin=9;
int potPin=0;
int val;
```

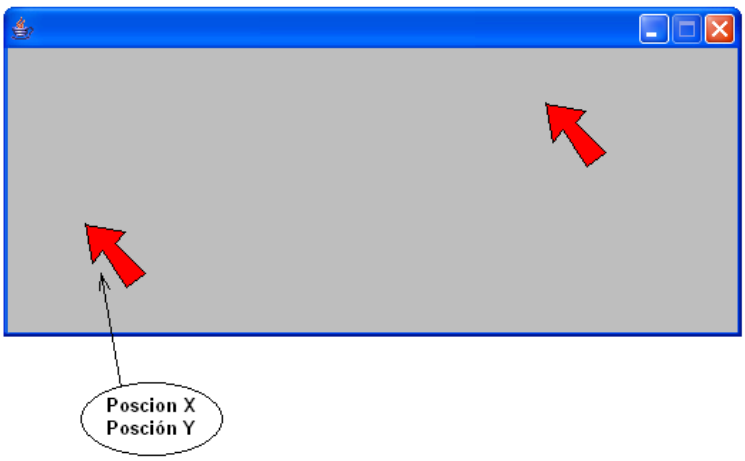
```
void setup() {
  size(200, 200);
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  arduino.pinMode(ledPin, Arduino.OUTPUT);
}

void draw() {
  //lee la señal del potenciometro (0..1024), divide por cuatro (0..255)
  val = arduino.analogRead(potPin)/4;
  /envia al LED el valor leido y ajustado (0..255) señal PWM
  arduino.analogWrite(ledPin, val);
  //varia la intensidad del color de la ventana de processing
  background(255-val,0,0);
}
```

7. Control del valor de una salida analógica de Arduino mediante la posición X e Y del ratón sobre la pantalla de ejecución de Processing.

Con este ejemplo se trata de conseguir variar el valor de la señal de salida (grado de iluminación del LED) de dos salidas analógicas de Arduino (salida 9 y salida 11)

Esquema de montaje





Aspecto de la pantalla de ejecución del IDE Processing

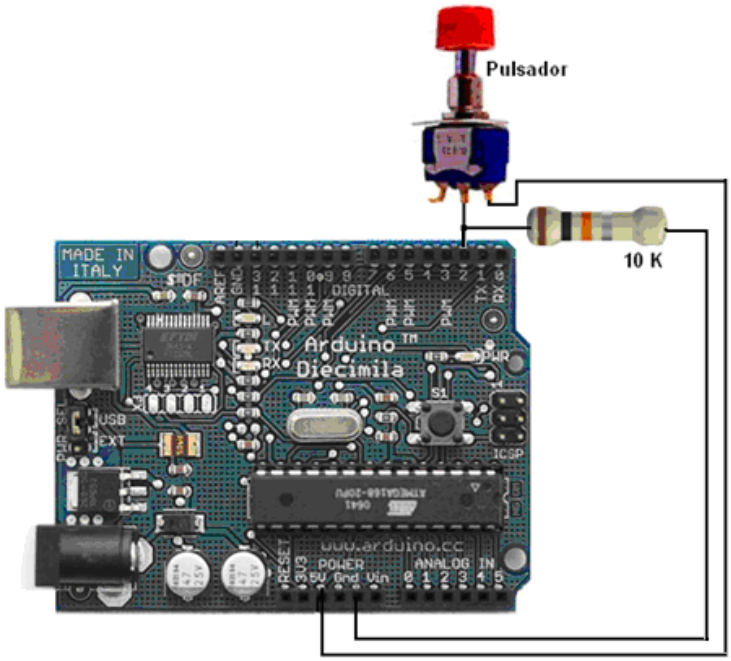
Solución

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

Arduino arduino;

void setup() {
  size(512, 512); // Definimos la ventana sobre la que se moverá el ratón
  arduino = new Arduino(this, Arduino.list()[1], 57600);
}

void draw() {
//Cambia el clor de la venta en función de la posición X del ratón
  background(constrain(mouseX / 2, 0, 255));
// Recoge el valor de la posición del ratón de la coordenada X y lo envía a la salida
// analógica 11
  arduino.analogWrite(11, constrain(mouseX / 2, 0, 255));
// Recoge el valor de la posición del ratón de la coordenada Y y lo envía a la salida
// analógica 9
  arduino.analogWrite(9, constrain(255 - mouseY / 2, 0, 255));
}
```

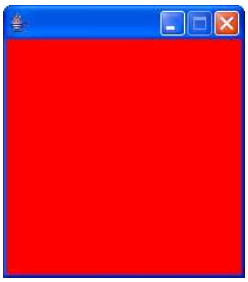
8. Lectura de pulsador

Se trata de realizar la lectura de una señal de entrada digital de Arduino y visualizar su estado mediante el color de fondo de la pantalla de ejecución de processing.

Esquema de montaje.



Aspecto de la pantalla de ejecución del IDE Processing



Botón sin pulsar

Botón pulsado

Solución

```
/*
 * cambiar el color de la ventana de ejecución dependiendo del estado de un pulsador
 * colocado en la entrada 2
 */
import processing.serial.*;
import cc.arduino.*;
```

```
Arduino arduino;
int switchPin = 2;

void setup() {
  size(200, 200);
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  arduino.pinMode(switchPin, Arduino.INPUT);
}

void draw() {
  if(arduino.digitalRead(switchPin)==Arduino.LOW) {  // Testea el estado de entrada 2
    background(255, 0, 0); // Si el estado es bajo, sin pulsar pone el fondo rojo
  } else {
    background(0, 0, 0); // Si el estado es alto, pulsado pone fondo negro
  }
}
```

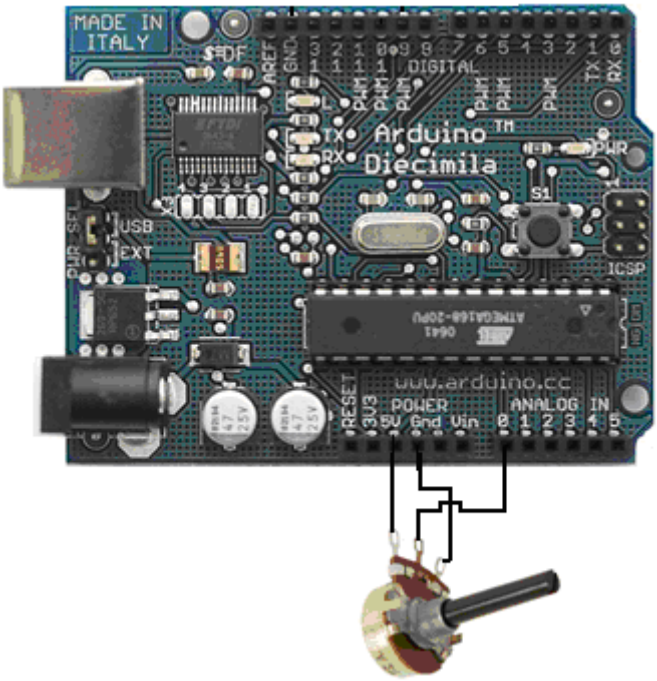
9. Detector de un nivel de señal analógica de entrada

Con este ejemplo se trata de ver como es posible leer una señal analógica a través de una de las entradas de Arduino y en función de su valor activar un texto y cambiar el color de la ventana de trabajo de Processing.

Señal de entrada analógica = Entrada analógica 0

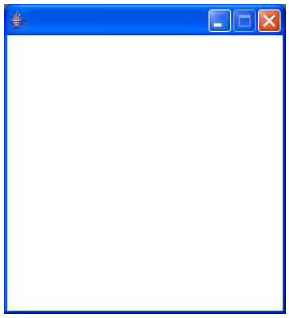
Valor de comparación = 512

Esquema de conexionado con la tarjeta Arduino



Aspecto de la pantalla de ejecución del IDE Processing

Valor >512



Valor <512

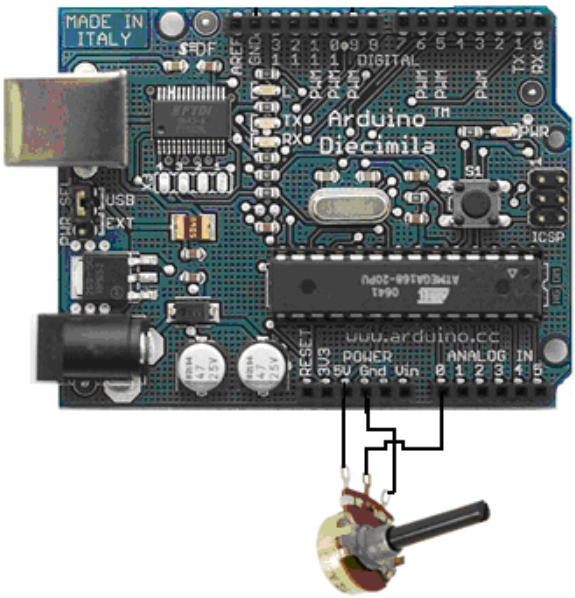
Solución

```
/**
 * Detector de Nivel.
 *
 */
 import processing.serial.*;
import cc.arduino.*;

int irpin=0; // Pin de entrada analógica
int val;  // valor leido de la entrada analógica
static final int IDLE =0; // estado normal si alarma
static final int ALARM =1; // estado de alarma
int mode;
int threshold=512; // Valor de comparación

Arduino arduino;
void setup()
{
  size(256,256);
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  PFont font;
  font = loadFont("sansserif-24.vlw");
  textFont(font);
  mode = IDLE;

}
void draw()
{

  val = arduino.analogRead(irpin); // lee valor del canal analógico
  switch(mode) { // Estructura switch
  case IDLE:  // caso IDLE=1 entonces
    if (val>threshold) { // si val es mayor que el valor de consigna (512)
      mode = ALARM; // se pasa al estado Alarma
    }
    background(255); // Color de fondo rojo
    break;

   case ALARM:  // caso alarma esta activo
     if(val<threshold) {  // Si val es menor que el valor de consigna (512)
       mode=IDLE; // Se pasa a modo IDLE
     }
     background(255,0,0); // fondo rojo
     fill(255);
     textAlign(CENTER); // muestra texto Alarma centrado
     text("Alarma!",128,128);

  }
}
```

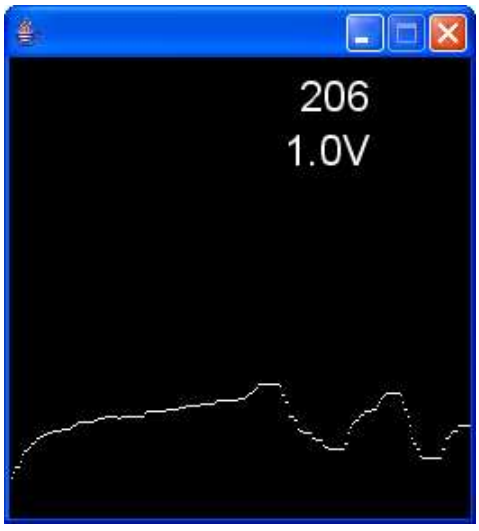
10. Lectura de un sensor y representación en modo grafico y texto del valor leído.

En este ejercicio se va a leer un valor de una señal analógica presente en la entrada analógica 0 y se va mostrar en la ventana de Processing su valor convertido (0-1024), su valor real en voltios (0-5) y su representación gráfica. En la figura se muestra lo que queremos conseguir.

Montaje con la tarjeta Arduino



Aspecto de la pantalla de ejecución del IDE Processing



Aspecto de la pantalla de ejecución de Processing

Solución

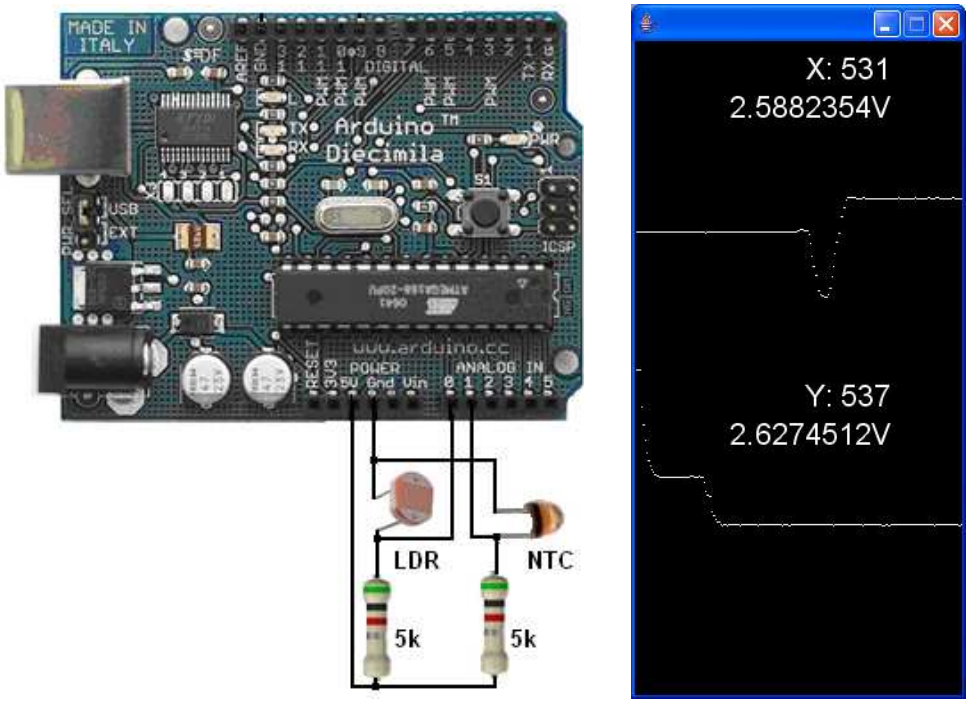
```
/**
 * Representación de valor procedente de un sensor.
 */
import processing.serial.*;
import cc.arduino.*;

int[] xvals; // Array que almacenara los valores leídos
int val; // Valor leído
int arrayindex = 0; // Puntero o índice del array
Arduino arduino;
int potPin=0;
void setup()
{
  size(256, 256);
  xvals = new int[width];
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  PFont font;
  font = loadFont("sansserif-24.vlw");
  textFont(font);

}
void draw()
{
  background(0);
   // shift array left by one
  for(int i=1; i<width; i++) {
    xvals[i-1] = xvals[i];
  }

  // añade un nuevo valor al final del array
  //lee la entrada analógica (0..1024), divide su valor por 4 (0..255)
  // to stay within canvas drawing limits
  val = arduino.analogRead(potPin);
  xvals[width-1] = val/4;

  // dibuja el array
  for(int i=1; i<width; i++) {
    stroke(255);
    point(i, 255-xvals[i]); //flip y coordinate so 0 is at bottom
  }
  textAlign(RIGHT);
  text(val, 200, 30);
  text(5.0*(xvals[width-1]/255.0)+"V",200,60);
}
```

11. Lectura y monitorización de dos sensores

Con esta aplicación se trata de monitorizar el valor de dos sensores colocados en los canales analógicos 0 y 1,

Esquema de montaje de la tarjeta Arduino

Se han colocado dos sensores uno de luz (LDR) y otro de temperatura (NTC). Las resistencias de 5K se deben adaptar en su valor al rango de variación de los sensores, por lo tanto pueden variar su valor.



Aspecto de la pantalla de ejecución de Processing y esquema de montaje

Solución

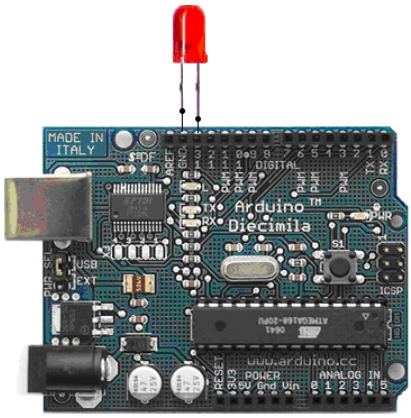
```
/**
 * Graph two sensor values simultaneously with text output
 */
import processing.serial.*;
import cc.arduino.*;

int[] xvals; // Definición de array para almacenar el canal X
int[] yvals; // Definición de array para almacenar el canal Y
int xval; // Valor leído en el canal X
int yval; Valor leído en el canal Y
int arrayindex = 0;
```

```
Arduino arduino;
int xpin=1; // Designación del PIN para canal X
int ypin=0; // Designación del PIN para canal Y
void setup()   // Configuración
{
  size(256,512);
  xvals = new int[width];
  yvals = new int[width];
  arduino = new Arduino(this, Arduino.list()[1], 57600);
  PFont font;
  font = loadFont("sansserif-24.vlw");
  textFont(font);
}

void draw()
{
  background(0);

  // desplaza la posición del array un lugar a la izquierda
  for(int i=1; i<width; i++) {
    xvals[i-1] = xvals[i];
    yvals[i-1] = yvals[i];
  }
  // añade el nuevo valor al final del array
  // lee el valor de la entrada (0..1024), divide por 4 (0..255)
  // para mantenerse dentro de límites lienzo de dibujo
  xval = arduino.analogRead(xpin);
  xvals[width-1] = xval/4;
  yval = arduino.analogRead(ypin);
  yvals[width-1] = yval/4;
  // dibuja el array
  for(int i=1; i<width; i++) {
    stroke(255);
    point(i, 255-xvals[i]); //flip y coordinate so 0 is at bottom
    point(i, 512-yvals[i]); //flip y coordinate so 0 is at bottom
  }
  textAlign(RIGHT);
  text("X: "+xval, 200, 30);
  text(5.0*(xvals[width-1]/255.0)+"V",200,60);
  text("Y: "+yval, 200, 256+30);
  text(5.0*(yvals[width-1]/255.0)+"V",200,256+60);

}
```

Ejemplos de control de Arduino mediante Processing utilizando el intercambio de datos a través del puerto serie

12. Enciende y apaga un LED pulsando cualquier tecla

En este ejercicio se trata de poder gobernar desde Processig una salida digital de Arduino (PIN13). La salida se activara y desactivara en modo biestable cuando pulsemos cualquier tecla del teclado



Programa para cargar en Arduino desde el IDE Arduino

ARDUINO

```
int ledPin= 13; // selecciona el PIN para el LED
int status =HIGH;
int val;

void setup() {
beginSerial(9600);
pinMode(ledPin, OUTPUT); // declara el  LED como salida
}

void loop(){
val= serialRead();// lee el valor del puerto serie
if(val!= -1) {
status = !status;
}
digitalWrite(ledPin, status);
delay(100);
}
```

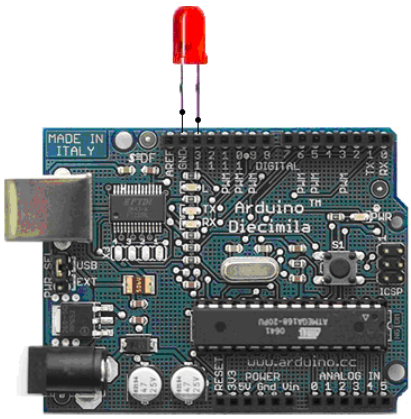
Programa para Cargar en el IDE Processing desde el que en modo ejecucion se gobierna la salida PIN13 pulsando cualquier tecla del teclado

PROCESSING

```
import processing.serial.*;

Serial port;
void setup() {
  size(255, 255);
  port = new Serial(this, Serial.list()[1], 9600);
}

void draw()
{
background(0);
}
void keyReleased() { //Manda al puerto la tecla pulsada
port.write(key);
}
```

13. Enciende y apaga un LED pulsando la letra “L”

En este ejemplo se trata de gobernar igualmente la salida digital del PIN13 pero esta vez se debe pulsar una tecla determinada, la “L”



Programa para Arduino

```
int ledPin= 13; // selecciona el pin para el LED
int status = HIGH;
int val;
void setup() {
beginSerial(9600);
pinMode(ledPin, OUTPUT); // declara LED como salida
}
void loop(){
val= serialRead();// lee el valor del puerto seri
if(val!= -1 && val== 'L') { // Pregunta si el valor es L y además el estado cambió
status = !status;
}
digitalWrite(ledPin, status);
delay(100);
}
```

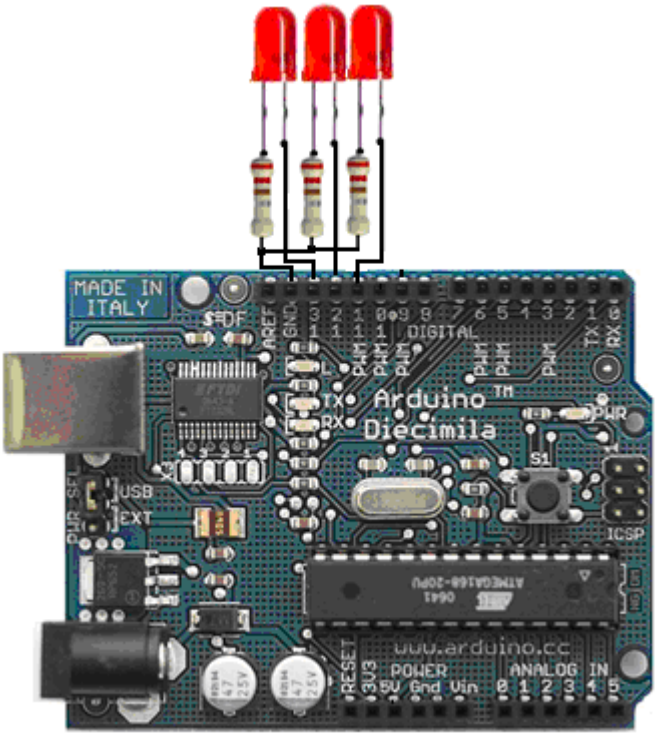
Programa para Processing (el mismo que el del anterior ejemplo)

```
import processing.serial.*;

Serial port;
void setup() {
  size(255, 255);
  port = new Serial(this, Serial.list()[1], 9600);
}

void draw()
```

```
{
background(0);
}
void keyReleased() { //Manda al puerto la tecla pulsada
port.write(key);
}
```

14. Control de tres LEDs desde las teclas 1,2 y3

En esta variante queremos controlar el encendido y apagado de cualquiera de los tres LEDs conectados a las salidas digitales PIN13, PIN12, PIN11 mediante las teclas “1”, “2” y “3” respectivamente actuando estas en modo biestable (una pulsación enciende la siguiente pulsación apaga)



Programa para Arduino

```
int ledPin1= 13, ledPin2= 12, ledPin3= 11;
int status1 = HIGH, status2 = HIGH, status3 = HIGH;
int val;
void setup() {
beginSerial(9600);
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
pinMode(ledPin3, OUTPUT);
}

void loop(){
val= serialRead();// lee el valor del puerto
if(val!= -1) {
switch(val) {
case'1': status1 = !status1; break;
case'2': status2 = !status2; break;
```

```
case'3': status3 = !status3; break;
}
}
digitalWrite(ledPin1, status1);
digitalWrite(ledPin2, status2);
digitalWrite(ledPin3, status3);
}
```

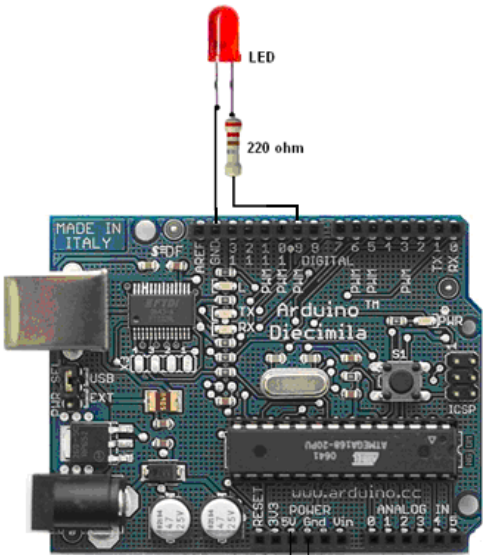
Programa para Processing (el mismo que el de los ejercicios anteriores)

```
import processing.serial.*;

Serial port;
void setup() {
  size(255, 255);
  port = new Serial(this, Serial.list()[1], 9600);
}

void draw()
{
background(0);
}
void keyReleased() { //Manda al puerto la tecla pulsada
port.write(key);
}
```

15. Controlar el brillo de un LED con el ratón

En este ejemplo se trata de controlar el brillo de un LED conectado en la salida PIN9 que actuará como salida Analógica PWM.

El valor de la cantidad de brillo lo envía Processing en función de la posición X del ratón sobre la ventana de ejecución de Processing.



Programa para Asrduino

```
int ledPin= 9; // selecciona el PIN para el LED
int val;
void setup() {
beginSerial(9600);
pinMode(ledPin, OUTPUT); // declara el LED como salida
}
void loop(){
val= serialRead();// lee el valor del puerto
if(val!= -1) {
analogWrite(ledPin, val);
}
}
```

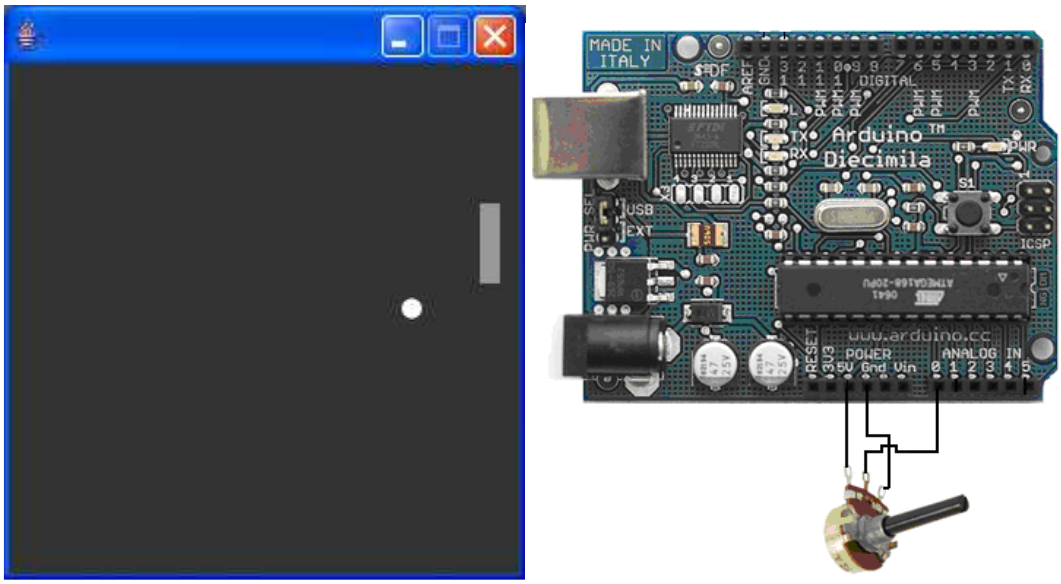
Programa para Processing

```
import processing.serial.*;
Serial port;

void setup() {
size(255, 255);
```

```
port = new Serial(this, Serial.list()[1], 9600);
frameRate(10);
}
void draw() {
background(0);
port.write(mouseX);
}
```

16. Juego Básico de PinPong

Vamos a realizar un ejemplo que consistirá en manejar una pala de pinpong mediante un potenciómetro colocado a la entrada analógica 0 de la tarjeta Arduino. El método que vamos utilizar es el método 2 es decir el de intercambio de datos entre la tarjeta Arduino y Processing a traves del puerto de comunicación serie en el que esta conectado Arduino



Pantalla de ejecución del juego de PinPong ejecutado desde Procesing

En la figura vemos la pantalla con la pelota y la paleta que se podrá desplazar mediante el potenciómetro colocado en la entrada “analog0”.

Programa para cargar en la tarjeta desde el IDE Arduino

```
// Envío de Entrada analógica 0 al puerto serie
int potPin=0; // Designa el numero de entrada analógica
void setup(){
  Serial.begin(19200); //Configura la velocidad de transmisión con el puerto
  }

void loop(){
  int val=analogRead(potPin); // Define val como el valor leído del puerto
  val=val/4; //Acondiciona el valor leído para ajustarse al tamaño de ventana
  Serial.print(val,BYTE); // envía val al puerto en formato BYTE
  delay(75); // espera 75 ms
  }
```

Programa para cargar y ejecutar desde el IDE Processing

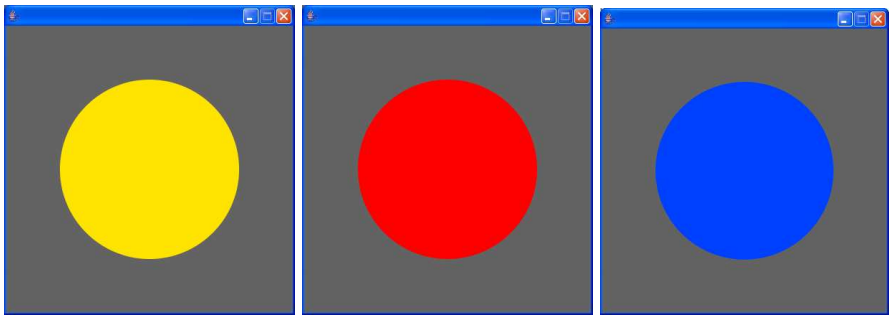
```
/**
 * Programa PinPong.
 *
 * Move the mouse up and down to move the paddle.
 *
 * Modified to use Serial port by Tod E. Kurt, 2007
 *
 * Updated 13 January 2003 by K Pfeiffer
 */

import processing.serial.*;

String portname = "COM4";  // aquí colocamos el puerto por el que recibimos el dato
Serial port;  // Creamos un objeto llamado port de la clase Serial

// Variables para definir la pelota
float ball_x;
float ball_y;
float ball_dir = 1;
float ball_size = 5;  // Radio
float dy = 0;  // Dirección

// variables para definir la pala
int paddle_width = 5;
int paddle_height = 20;
int paddle_pos;   // nueva posición
int paddle_ppos;  // última posición
int dist_wall = 15;

void setup()
{
  size(255, 255);
  rectMode(CENTER_RADIUS);
  ellipseMode(CENTER_RADIUS);
  noStroke();
  smooth();
  ball_y = height/2;
  ball_x = 1;

  // Abre el puerto al que esta conectada la tarjeta con una velocidad de (19200 bps)
  port = new Serial(this, portname, 19200);
}

void draw()
{
  background(51);

  ball_x += ball_dir * 1.0;
```

```
 ball_y += dy;
 if(ball_x > width+ball_size) {
   ball_x = -width/2 - ball_size;
   ball_y = random(0, height);
   dy = 0;
 }

if (port.available() > 0) {  // Si el dato está disponible,
   paddle_ppos = paddle_pos;      // guarda la ultima posición
   paddle_pos = port.read();      // lee y almacena la nueva posición
 }

 // Desplaza la pala verticalmente en la pantalla
 float paddle_y = constrain(paddle_pos, paddle_height, height-paddle_height);

 // Testea si la pelota toca la pala
 float py = width-dist_wall-paddle_width-ball_size;
 if(ball_x == py
    && ball_y > paddle_y - paddle_height - ball_size
    && ball_y < paddle_y + paddle_height + ball_size) {
   ball_dir *= -1;
   if(paddle_pos != paddle_ppos) {
     dy = (paddle_pos - paddle_ppos)/2.0;
     if(dy >  5) { dy =  5; }
     if(dy < -5) { dy = -5; }
   }
 }

 // Si la pelota toca la pala o la pared, cambia de dirección
 if(ball_x < ball_size && ball_dir == -1) {
   ball_dir *= -1;
 }
   // Si la pelota toca la parte superior o inferior del borde, cambia dirección
 if(ball_y > height-ball_size) {
   dy = dy * -1;
 }
 if(ball_y < ball_size) {
   dy = dy * -1;
 }

 // Dibuja la pelota
 fill(255);
 ellipse(ball_x, ball_y, ball_size, ball_size);

 // Dibuja la paleta
 fill(153);
 rect(width-dist_wall, paddle_y, paddle_width, paddle_height);
}
```

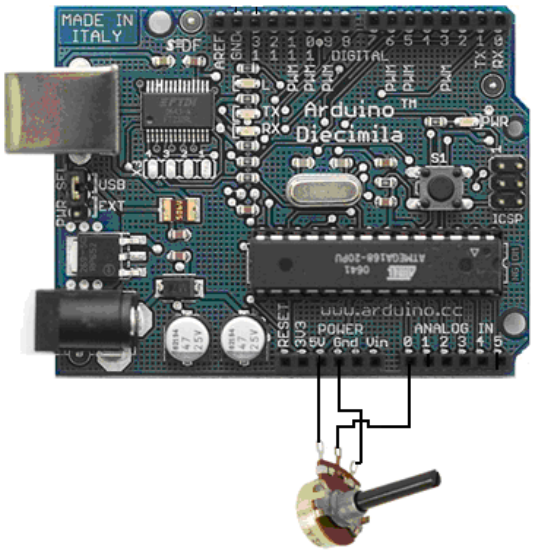
17. Cambio del color de un círculo.

Con esta aplicación se trata de cambiar el color de una circulo dibujado en la pantalla de ejecución e Processing en función de la señal analógica leída a través del puerto serie que la tarjeta Arduino lee de una de sus entradas analógicas (entrada0).

En la figura vemos el aspecto de la pantalla de ejecución de Processing.



Distinto color en función del valor leído en el puerto serie.



Programa para cargar en la tarjeta desde el IDE Arduino

```
// Envío de Entrada analógica 0 al puerto serie
int potPin=0; // Designa el numero de entrada analógica
void setup(){
  Serial.begin(19200); //Configura la velocidad de transmisión con el puerto
  }

void loop(){
  int val=analogRead(potPin); // Define val como el valor leído del puerto
  val=val/4; //Acondiciona el valor leído para ajustarse al tamaño de ventana
```

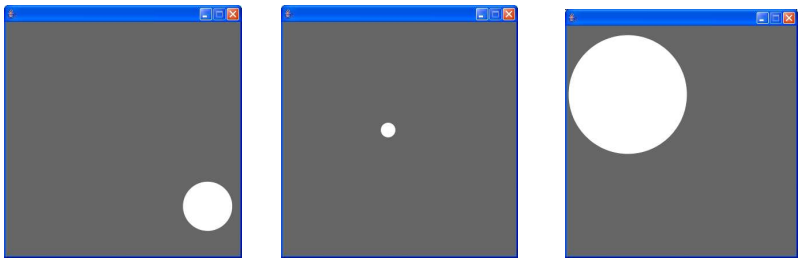
```
  Serial.print(val,BYTE); // envía val al puerto en formato BYTE
  delay(75); // espera 75 ms
  }
```

Programa para cargar y ejecutar desde el IDE Processing

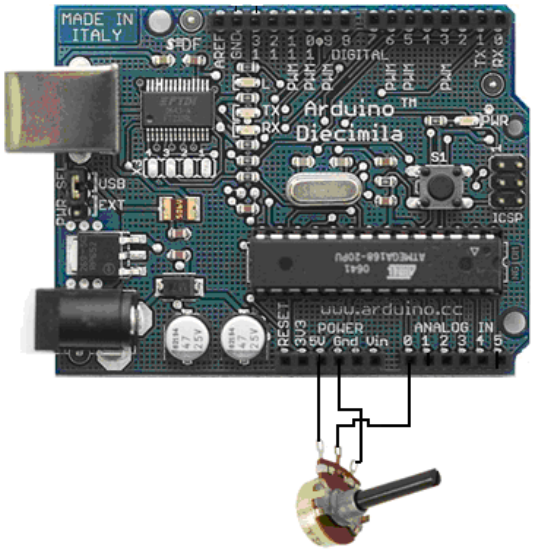
```
/**
 * Lectura Simple de una variable analógica
 *
 * Lee el valor a través del puerto serie y cambia el color del circulo
 *.
 */


import processing.serial.*;

String portname = "COM4";  // Se selecciona el puerto por el que se realizará la comunicación
Serial port;  // Crea un objeto de la clase Serial

int val=100;  // Valor recibido del puerto con un valor inicial

void setup()
{
  size(400, 400);

  colorMode(HSB, 255);
  ellipseMode(CENTER);   // dibuja el centro de la elipse
  noStroke();
  frameRate(30);
  smooth();

  // Abre el puerto conectado a la tarjeta
  port = new Serial(this, portname, 19200);
}
void draw()
{
  if (port.available() > 0) {  // Si el dato a leer está disponible
    val = port.read();         // lee y almacena el valor
  }
  background(99);
  // Dibuja la figura
  fill(val,255,255);  // este esta en formato HSB, el primer dato es el color
  ellipse(width/2, height/2, 250,250);
}
```

18. Pelota cambiando de tamaño y botando

Con este ejemplo se va cambiar el tamaño de una pelota que se dibujará como un círculo de color blanco en la pantalla y a la vez la pelota estará permanentemente moviéndose, rebotando en las paredes de la pantalla de ejecución de Processing



Tres vistas de la pantalla de ejecución de Processing en las que se aprecia que el tamaño y la posición de la pelota es distinto.



Programa para cargar en la tarjeta desde el IDE Arduino

```
// Envío de Entrada analógica 0 al puerto serie
int potPin=0; // Designa el numero de entrada analógica
void setup(){
  Serial.begin(19200); //Configura la velocidad de transmisión con el puerto
  }

void loop(){
  int val=analogRead(potPin); // Define val como el valor leído del puerto
  val=val/4; //Acondiciona el valor leído para ajustarse al tamaño de ventana
  Serial.print(val,BYTE); // envía val al puerto en formato BYTE
  delay(75); // espera 75 ms
  }
```

Programa para cargar en la tarjeta desde el IDE Arduino

```
/**
 * Pelota cambiando posición y tamaño.
 *
*/

import processing.serial.*;

String portname = "COM4";  // Selección de puerto de comunicación con Arduino
Serial port;  // Crea un objeto de la clase Serial
int size = 60;      // Anchura de la pelota
float xpos, ypos;    // Posición de inicio de la pelota

float xspeed = 3.9;  // Velocidad de la pelota en el eje X
float yspeed = 3.1;  // Velocidad de la pelota en el eje Y
int xdirection = 1;  // Izquierda derecha
int ydirection = 1;  // Adelante atrás


void setup()
{
  size(400, 400);
  colorMode(HSB, 255);
  noStroke();
  frameRate(30);
  ellipseMode(CENTER);   // dibuja el centro de la circunferencia (pelota)
  smooth();
  background(102);
  // Configura la posición de la pelota en el momento de comenzar
  xpos = width/2;
  ypos = height/2;

  // Abre el puerto de comunicación con Arduino y usa la misma velocidad (19200 bps)
  port = new Serial(this, portname, 19200);

}

void draw()
{
 if (port.available() > 0) {  // Si esta disponible el dato,
    size = port.read();        // lee y almacena el valor del nuevo tamaño
  }

  // Actualizar la posición del circulo
  xpos = xpos + ( xspeed * xdirection );
```

```
  ypos = ypos + ( yspeed * ydirection );

  // Test para ver si la forma es superior a los límites de la pantalla
  // Si es así, marcha atrás en su dirección de multiplicar por -1
  int halfsize = size/2;  // porque estamos dibujando el circulo desde el centro
  if (xpos + halfsize > width || xpos - halfsize < 0) {
    xdirection *= -1;
  }
  if (ypos + halfsize > height || ypos - halfsize < 0) {
    ydirection *= -1;
  }

  // Dibuja circulo
  background(102);  // se podría realizar la modificación
  //fill(size,255,255);  // Con la instrucción anterior varia también el color de la pelota
  ellipse(xpos, ypos, size, size);
}
```

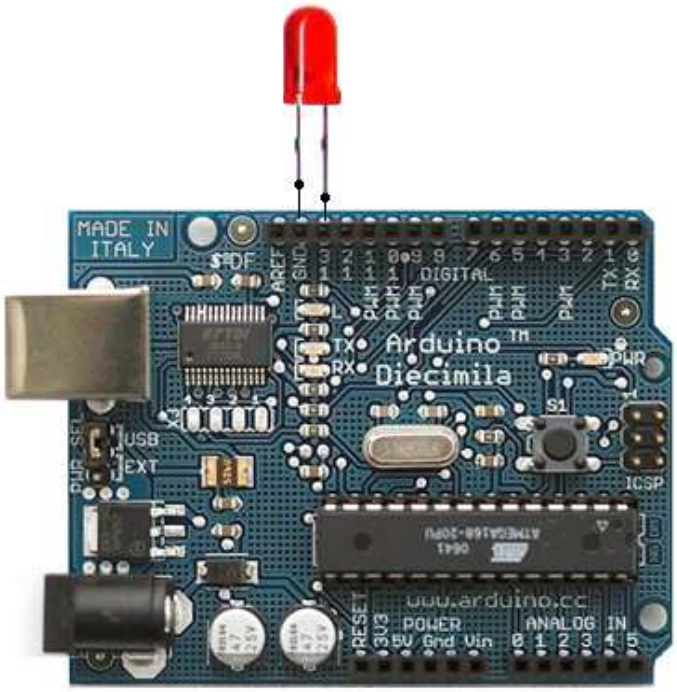
19. Control de una salida de Arduino mediante el envio a la tarjeta de una letra

Control desde el IDE de Arduino

Un ejemplo de la utilización de la tarjeta Arduino para recibir los datos de la computadora. En este caso en la salida 13 de Arduino se enciende un LED cuando se recibe el carácter 'H', y se apaga el LED cuando reciba el carácter «L».

Los datos se pueden enviar desde el monitor serie Arduino, o con otro programa de procesamiento (véase el código que aparece a continuación) como Processing pudiendo ser también Pure Data o MaX

Un LED en el pin13.

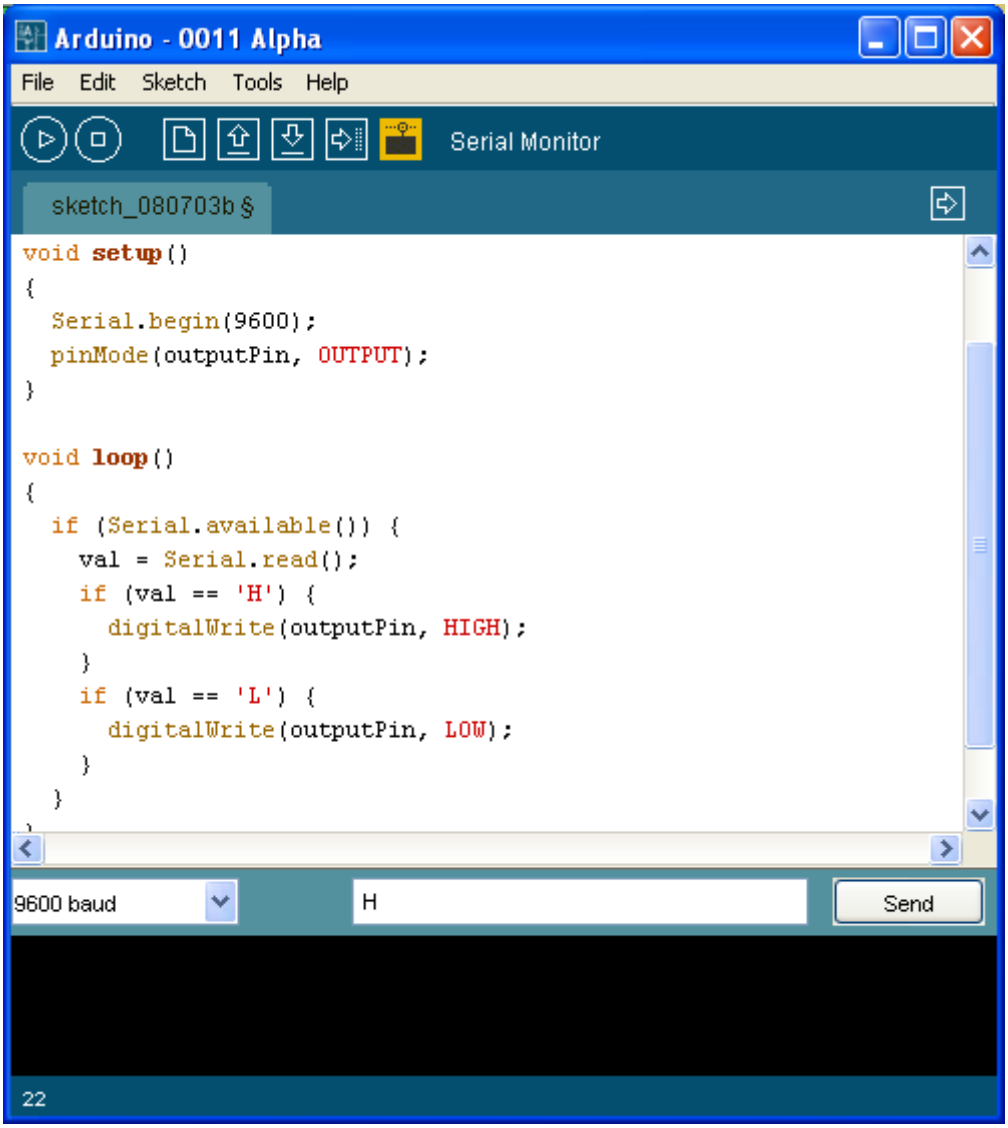


Esquema de montaje

Obsérvese que se ha colocado el diodo led sin resistencia en serie dado que el PIN13 de Arduino ya lleva incorporada una resistencia interior, en el caso de colocar el diodo Led en otra salida deberíamos colocar una resistencia de al entre 220 y 500 ohmios dependiendo del consumo de corriente del diodo

Código para descargar en Arduino desde el IDE Arduino

```
int outputPin = 13;
int val;

void setup()
{
  Serial.begin(9600);
  pinMode(outputPin, OUTPUT);
}

void loop()
{
  if (Serial.available()) {
    val = Serial.read();
    if (val == 'H') {
      digitalWrite(outputPin, HIGH);
    }
    if (val == 'L') {
      digitalWrite(outputPin, LOW);
                }
        }
}
```



Control desde Processing

Si queremos gobernar la salida desde Processing debemos cargar el siguiente programa en processing y ejecutarlo (siempre que previamente este cargado en la tarjeta el programa anterior)

Codigo Processing

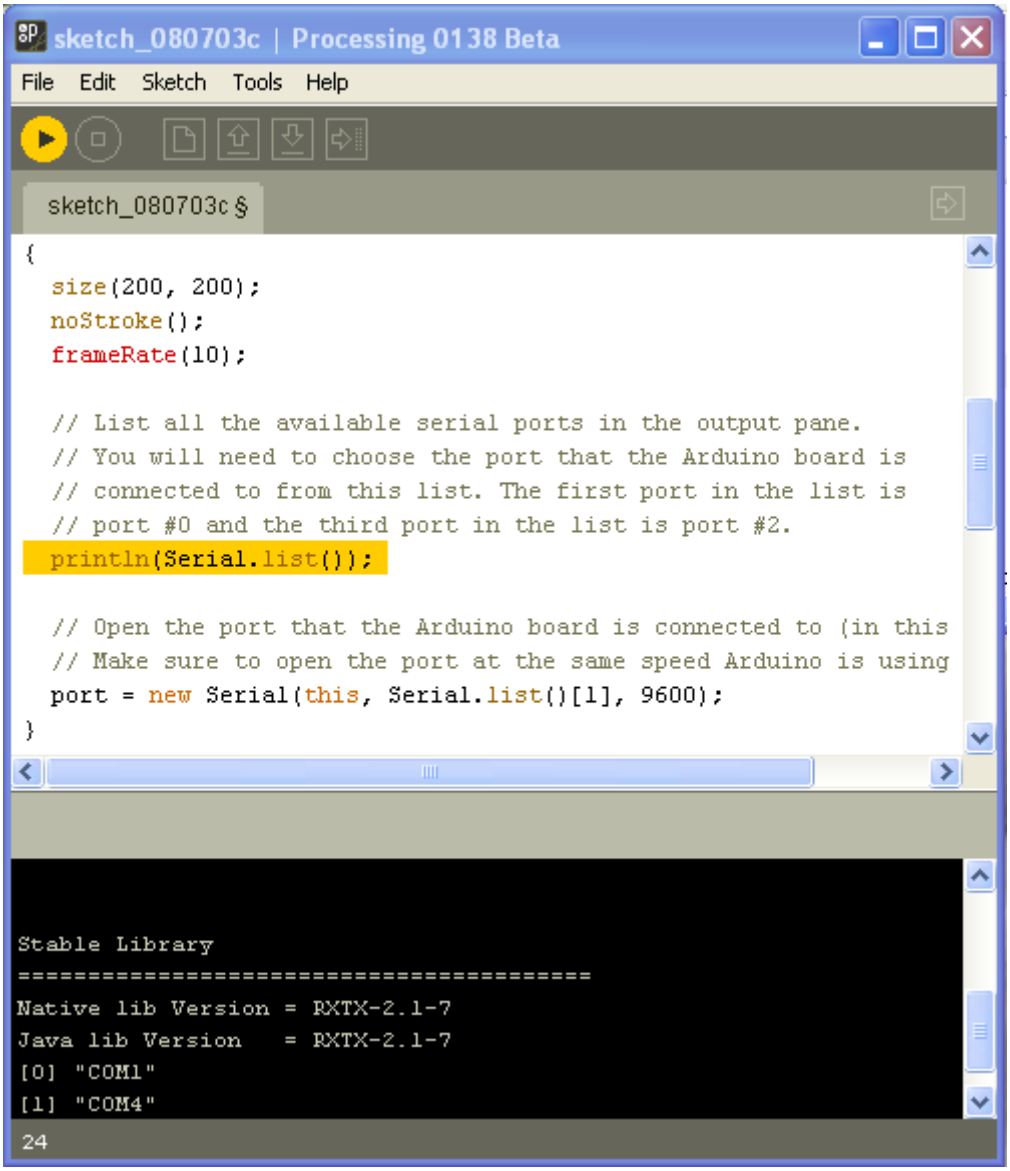
```
// mouseover serial
// by BARRAGAN <http://people.interaction-ivrea.it/h.barragan>

// Demonstrates how to send data to the Arduino I/O board, in order to
// turn ON a light if the mouse is over a rectangle and turn it off
// if the mouse is not.

// created 13 May 2004
```

```
import processing.serial.*;

Serial port;

void setup()
{
  size(200, 200);
  noStroke();
  frameRate(10);

  // List all the available serial ports in the output pane.
  // You will need to choose the port that the Arduino board is
  // connected to from this list. The first port in the list is
  // port #0 and the third port in the list is port #2.
  println(Serial.list());

  // Open the port that the Arduino board is connected to (in this case #0)
  // Make sure to open the port at the same speed Arduino is using (9600bps)
  port = new Serial(this, Serial.list()[0], 9600);
}

// function to test if mouse is over square
boolean mouseOverRect()
{
  return ((mouseX >= 50)&&(mouseX <= 150)&&(mouseY >= 50)&(mouseY <= 150));
}

void draw()
{
  background(#222222);
  if(mouseOverRect())      // if mouse is over square
  {
    fill(#BBBBB0);         // change color
    port.write('H');       // send an 'H' to indicate mouse is over square
  } else {
    fill(#666660);         // change color
    port.write('L');       // send an 'L' otherwise
  }
  rect(50, 50, 100, 100);  // draw square
}
```

La instrucción **println(Serial.list());** imprime en la ventana de salida de Processing los puertos detectados.

Téngase en cuenta que el puerto de comunicación debe consignarse adecuadamente y modificarlo en la instrucción:

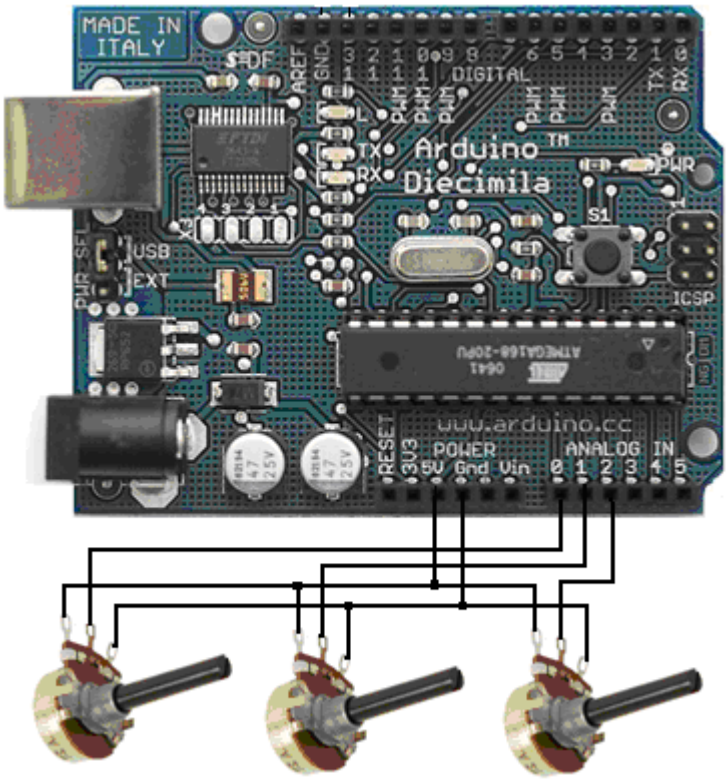
```
port = new Serial(this, Serial.list()[1], 9600);
```


sketch_080703c | Processing 0138 Beta
File Edit Sketch Tools Help
sketch_080703c §
{
size(200, 200);
noStroke();
frameRate(10);
// List all the available serial ports in the output pane.
// You will need to choose the port that the Arduino board is
// connected to from this list. The first port in the list is
// port #0 and the third port in the list is port #2.
println(Serial.list());
// Open the port that the Arduino board is connected to (in this
// Make sure to open the port at the same speed Arduino is using
port = new Serial(this, Serial.list()[1], 9600);
}
Stable Library
==========
Native lib Version = RXTX-2.1-7
Java lib Version = RXTX-2.1-7
[0] "COM1"
[1] "COM4"
24

20. Mezclador virtual de colores

Demuestra una técnica para el envío de múltiples valores desde la tarjeta Arduino al ordenador. En este caso se enviarán las lecturas de tres potenciómetros que se utilizan para fijar los colores rojo, verde y azul para el fondo de la pantalla de trabajo de Processing.

Circuito



Potenciómetros conectados a las entradas analógicas de los pins 0, 1, y 2.

Código para cargar a través del IDE Arduino en la tarjeta.

```
int redPin = 0;
int greenPin = 1;
int bluePin = 2;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print("R");
  Serial.println(analogRead(redPin));
  Serial.print("G");
  Serial.println(analogRead(greenPin));
  Serial.print("B");
  Serial.println(analogRead(bluePin));
```

```
  delay(100);
}
```

Código para cargar y ejecutar desde el IDE Processing una vez que ya hemos cargado el código anterior

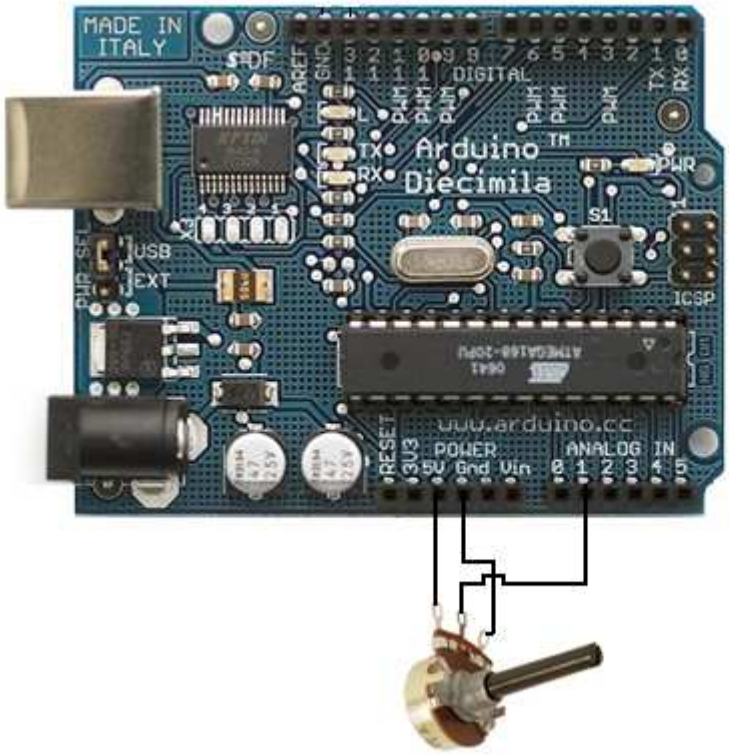
```
/**
 * Mezcla de colores
 * by David A. Mellis
 *
 * Created 2 December 2006
 *
 * based on Analog In
 * by <a href="http://itp.jtnimoy.com">Josh Nimoy</a>.
 *
 * Created 8 February 2003
 * Updated 2 April 2005
 */

import processing.serial.*;

String buff = "";
int rval = 0, gval = 0, bval = 0;
int NEWLINE = 10;

Serial port;

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

  // Imprime una lista con los puertos disponibles de tipo COM
  println("Puertos disponibles:");
  println(Serial.list());

  //port = new Serial(this, "COM1", 9600);
  // Usa el primer puerto disponible
  port = new Serial(this, Serial.list()[0], 9600);
}

void draw()
{
  while (port.available() > 0) {
    serialEvent(port.read());
  }
  background(rval, gval, bval);
}

void serialEvent(int serial)
{
/ / Si la variable "serial" no es igual al valor para

   / / una nueva línea, añadir el valor a la variable "buff". Si el

   / / valor "serial" es igual al valor de una nueva línea,
```

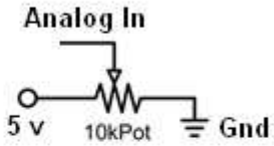
```
  / / guardar el valor en el buffer dentro de la variable "val".

  if(serial != NEWLINE) {
    buff += char(serial);
  } else {
/ / El primer carácter nos dice que tipo de color es el valor
    char c = buff.charAt(0);
    //lo quita de la cadena
    buff = buff.substring(1);
/ / Descartar el retorno de carro al final del buffer
    buff = buff.substring(0, buff.length()-1);
    // Convierte el string en un entero
    if (c == 'R')
      rval = Integer.parseInt(buff);
    else if (c == 'G')
      gval = Integer.parseInt(buff);
    else if (c == 'B')
      bval = Integer.parseInt(buff);
/ / Borrar el valor de "buff"
    buff = "";
  }

}
```

21. Trazado Grafico de una señal.

(Leída desde una entrada analógica de la tarjeta Arduino y mostrada mediante Processing)

Para realizar este ejemplo bastará conectar un potenciómetro a la entrada analógica 1 PIN 1 de la tarjeta Arduino tal como se muestra en la figura.





Circuito Simulacion Entrada Analógica

Se carga Arduino con el siguiente programa

PROGRAMA ARDUINO

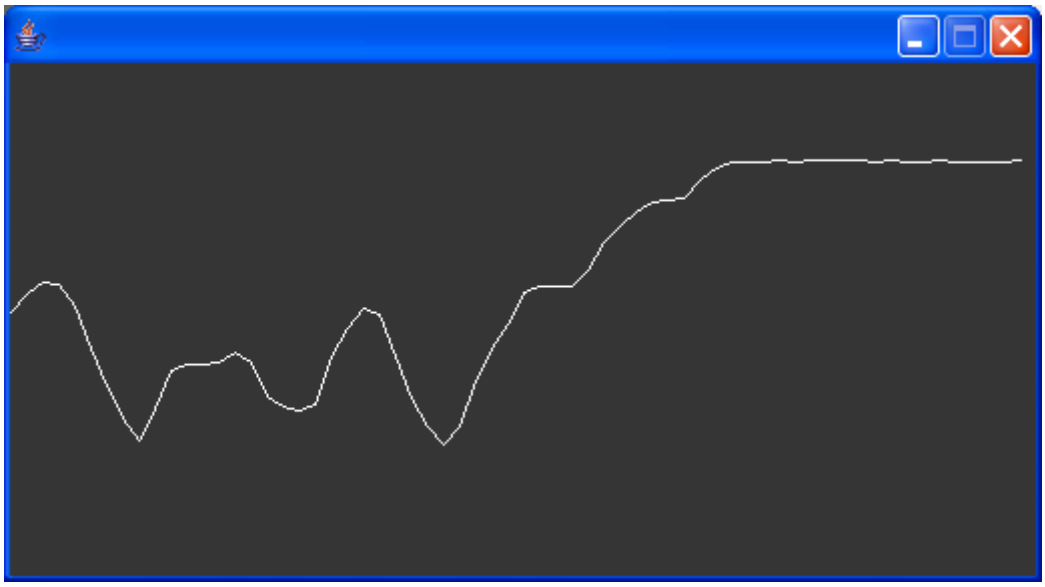
```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println(analogRead(1));
  delay(100);
}
```

A Continuación se carga en el entorno Processing el siguiente programa, sin olvidar configurar el puerto por el que leerá los datos que le llegaran de Arduino.

PROGRAMA PARA PROCESSING

```
// Graph
// by David A. Mellis
//
// Demonstrates reading data from the Arduino board by graphing the
// values received.
//
// based on Analog In
// by <a href="http://itp.jtnimoy.com">Josh Nimoy</a>.

import processing.serial.*;

Serial port;
String buff = "";
int NEWLINE = 10;

// Store the last 64 values received so we can graph them.
int[] values = new int[64];

void setup()
{
  size(512, 256);

  println("Available serial ports:");
  println(Serial.list());

  // Uses the first port in this list (number 0).  Change this to
  // select the port corresponding to your Arduino board.  The last
  // parameter (e.g. 9600) is the speed of the communication.  It
  // has to correspond to the value passed to Serial.begin() in your
  // Arduino sketch.
  port = new Serial(this, Serial.list()[1], 9600);

  // If you know the name of the port used by the Arduino board, you
  // can specify it directly like this.
  //port = new Serial(this, "COM1", 9600);
}

void draw()
{
  background(53);
  stroke(255);
```

```
  // Graph the stored values by drawing a lines between them.
  for (int i = 0; i < 63; i++)
    line(i * 8, 255 - values[i], (i + 1) * 8, 255 - values[i + 1]);

  while (port.available() > 0)
    serialEvent(port.read());
}

void serialEvent(int serial)
{
  if (serial != NEWLINE) {
    // Store all the characters on the line.
    buff += char(serial);
  } else {
    // The end of each line is marked by two characters, a carriage
    // return and a newline.  We're here because we've gotten a newline,
    // but we still need to strip off the carriage return.
    buff = buff.substring(0, buff.length()-1);

    // Parse the String into an integer.  We divide by 4 because
    // analog inputs go from 0 to 1023 while colors in Processing
    // only go from 0 to 255.
    int val = Integer.parseInt(buff)/4;

    // Clear the value of "buff"
    buff = "";

    // Shift over the existing values to make room for the new one.
    for (int i = 0; i < 63; i++)
      values[i] = values[i + 1];

    // Add the received value to the array.
    values[63] = val;
  }
}
```



Este sería el aspecto del gráfico sacado por Processing

22. Enviar valor analógico a Arduino a través del puerto serie.

Con esta practica se trata de realizar el envío de un valor numérico a través del puerto serie y que Arduino lo saque por una de sus salidas Analógicas, en este caso el PIN 9.

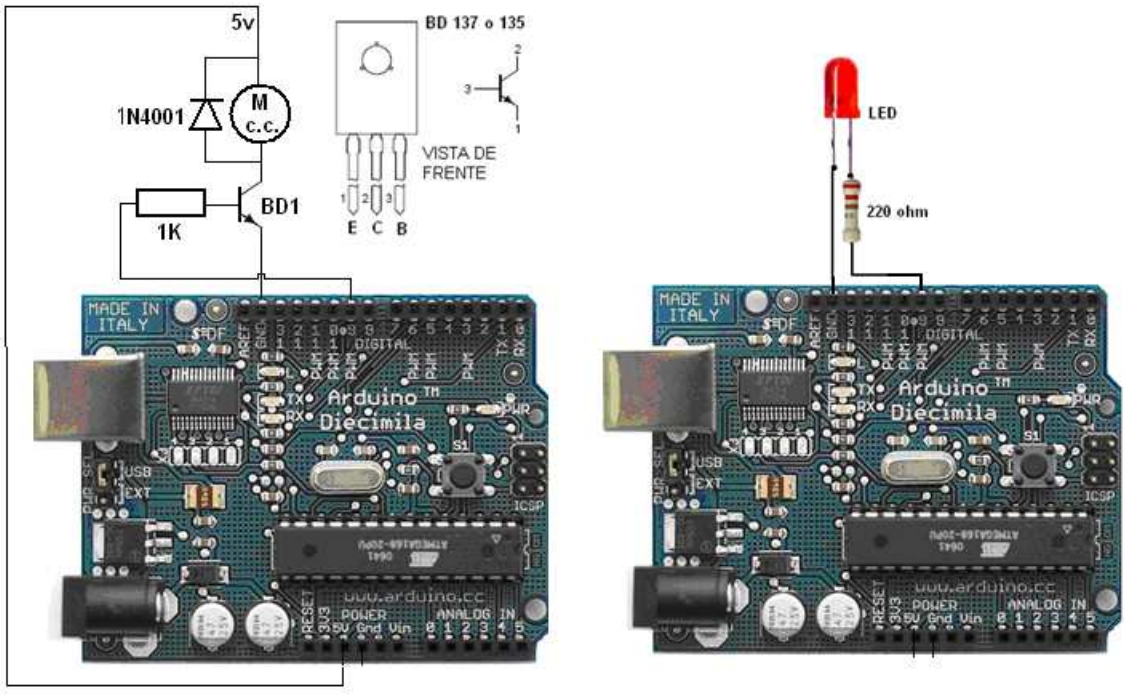
El valor lo vamos a obtener lanzando la aplicación Processing y desde este creando una pantalla sensible a la posición del ratón desplazamos este sobre ella y conseguimos capturar la coordenada x (posición horizontal del ratón) y la enviamos al puerto serie para que la pueda leer Arduino.

Tendremos que disponer de dos programas:

Programa para Arduino: Configurando el PIN9 como salida analógica de un valor

Programa para Processing: Que dibuja en pantalla la ventana de captura de ratón y envía el dato.

Esquema de aplicaciones para probar este ejercicio.



El valor que enviamos al puerto de salida configurado como analógico PWN nos puede servir para controlar la velocidad de un pequeño motor o simplemente la intensidad de encendido de un diodo LED.

Programa Arduino:

```
//Programa para configurar el PIN9 como salida analógica
// y lectura del puerto serie de un valor que será el que
// se entregue en la salida analógica configurada

int salida_analogica = 9;
byte valor_leido_puerto = 0;
int serByte = -1;

void setup() {
 pinMode(salida_analogica, OUTPUT);
 beginSerial(9600);
}

void loop() { // Bucle de lectura del dato por el puerto y su escritura en el PIN9

  serByte = serialRead(); //Byte leído en el puerto
  if (serByte != -1) {
    valor_leido_puerto = serByte;
  }
  analogWrite(salida_analogica,valor_leido_puerto*2);
}
```

Programa de Processing

```
// PROGRAMA PARA PROCESSING

//Captura del valor de la posición horizontal (coordenada x)
// del ratón y su envío al puerto a ARDUINO y su visualización

import processing.serial.*;

byte serialout = 0;
float T = 0.0;
float myfloat = 0.0;

Serial port;

void setup()
{
  size(1000, 100); //define el tamaño de la ventana sensible  al ratón
  background(0); //Color fondo
  port = new Serial(this, "COM4", 9600); // Definición del puerto

}

void draw()
{
```

```
  myfloat = float(mouseX);
T = (myfloat/width)*126;
serialout = byte(T);
port.write(serialout); //escribe en el puerto el valor de la coordenada x del ratón
println(serialout); //Imprime valor numérico de la posición del ratón

}
```

Modo de Operar:

Primero se carga el programa Arduino sobre la tarjeta y luego se carga y se ejecuta el programa Processing y se ejecuta.

Cuando aparezca la ventana movemos sobre la franja negra el ratón y observamos como el diodo Led colocado en la salida PIN9 se ilumina gradualmente.