

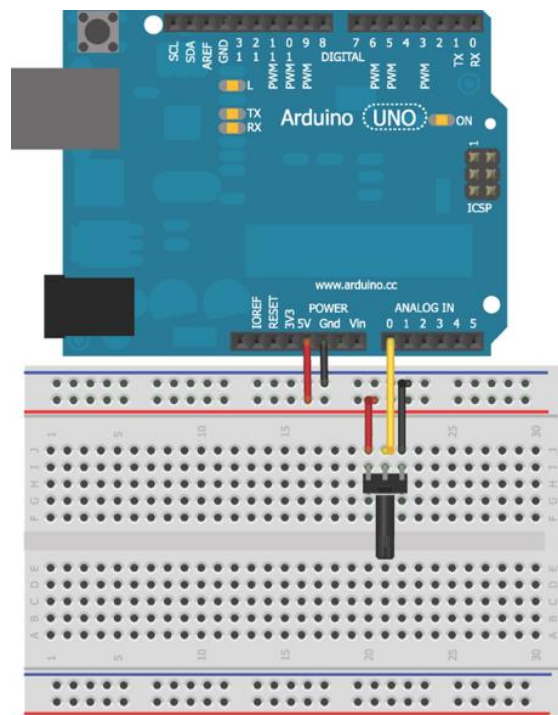
Practica 04 – USB. Comunicaciones Serie

David Subires Parra

Montajes

Ejercicio 1 – Escuchando a arduino

La primera tarea serie que podemos hacer con Arduino es abrir el terminal serie y mostrar la comunicación entre placa y PC. Esto es algo que ya se ha debido hacer en la práctica 3. En esta práctica vamos a ir más lejos.



He realizado este montaje con el potenciómetro digital proporcionado, MCP 4231. A continuación se detallan los pines de conexión de dicho amplificador:

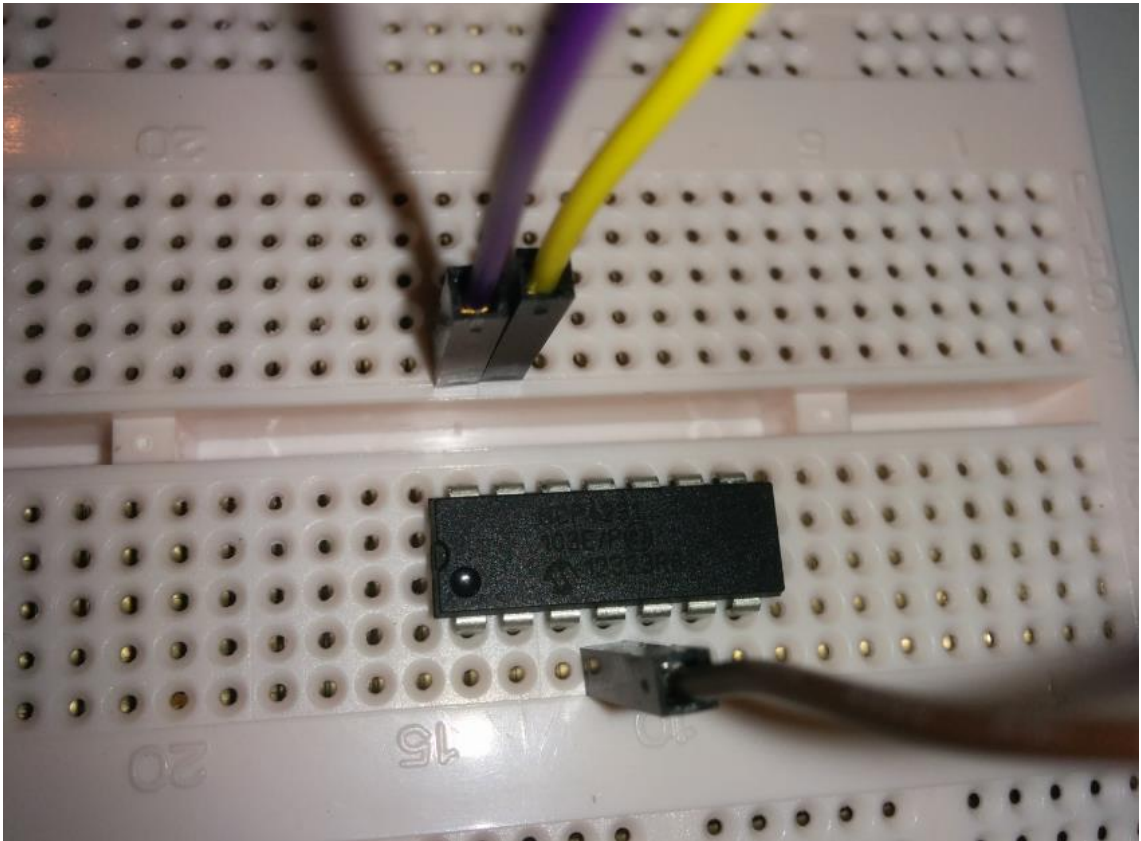


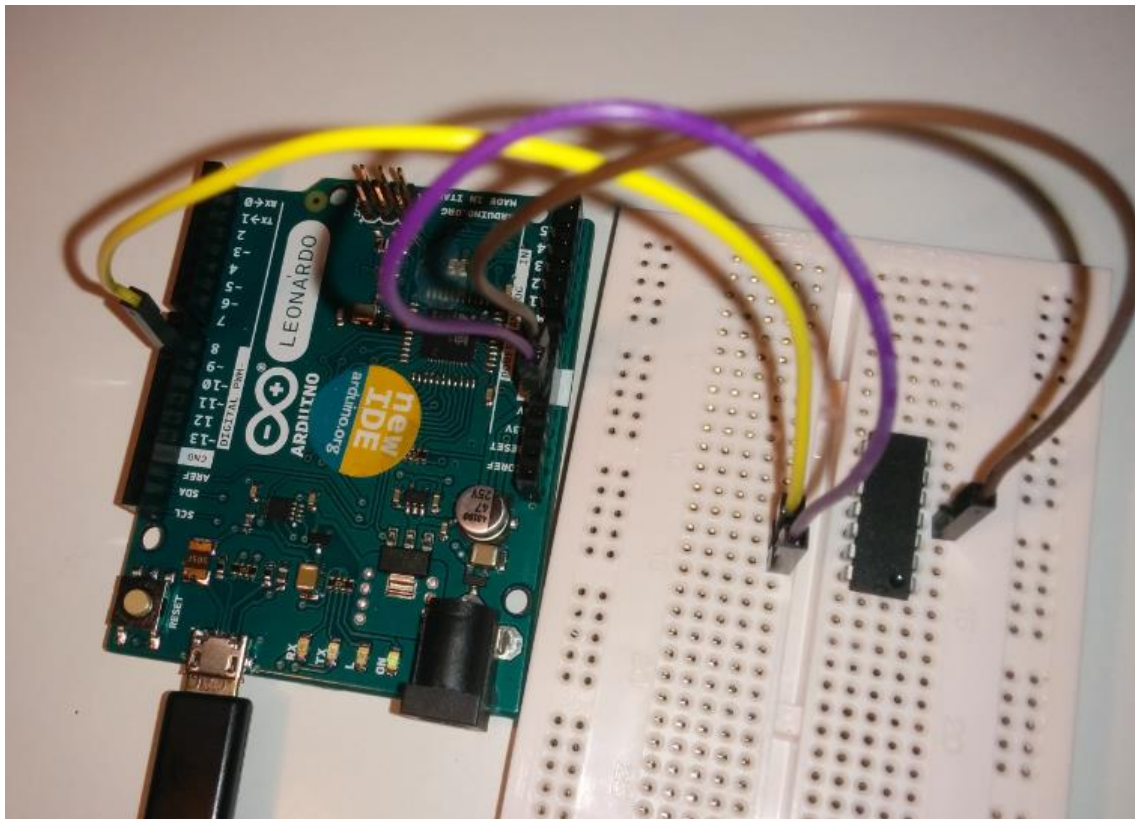
MCP42X1 Dual Potentiometers

\overline{CS}	1	14	V_{DD}
SCK	2	13	\underline{SDO}
SDI	3	12	\underline{SHDN}
V_{SS}	4	11	WP
P1B	5	10	P0B
P1W	6	9	P0W
P1A	7	8	P0A

MCP4231 Digital Potentiometer		
Pin #	Pin Name	Description
1	CS	The CS, or Chip Select, pin is the SS (slave select) pin for the SPI interface. It is active low. 0V means the chip is selected and 5V means it is not selected.
2	SCLK	SCLK is the Shared/Serial Clock. It is the SPI clock line.
3	SDI	These pins are the serial data in, also known as the MOSI.
4	Vss	This is where ground is connected to.
5	P1B	This is one terminal of potentiometer 1.
6	P1W	This is the wiper terminal of potentiometer 1.
7	P1A	This is one terminal of potentiometer 1.
8	PA0	This is one terminal of potentiometer 0.
9	PW0	This is the wiper terminal of potentiometer 0.
10	PB0	This is one terminal of potentiometer 0.
11	WP	This is Write Protect. You can ignore this pin. It's left not connected (NC).
12	SHDN	This is the Shutdown pin. It is active low. When held low, the hardware disconnects the wiper from the internal resistor network. Since we always want the potentiometer to be active, we keep this pin connected to +5V.
13	SDO	This is the serial data output pin. This pin is also known as MISO.
14	VDD	This is where the positive voltage source connects to.

Puesto que la imagen del montaje anterior está realizada con un potenciómetro analógico, he realizado un montaje diferente, conectando los pines 4, 13 y 14 del potenciómetro a tierra, pin 9 y 5v respectivamente en la placa Arduino. Podemos observar el montaje en las siguientes imágenes:





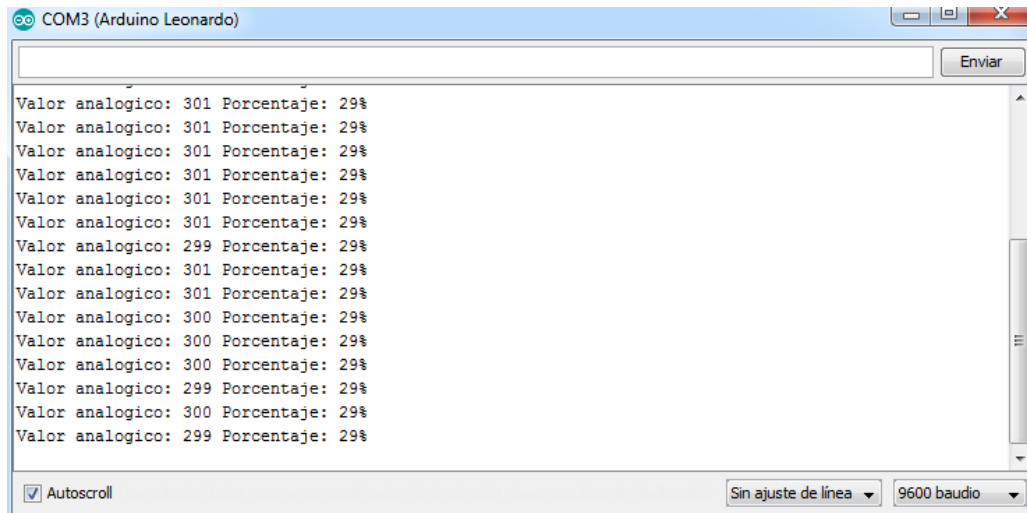
Y para finalizar este ejercicio, este es el código que he usado y las salidas que produce

```
ejercicio_01 Arduino 1.6.9
Archivo Editar Programa Herramientas Ayuda

ejercicio_01

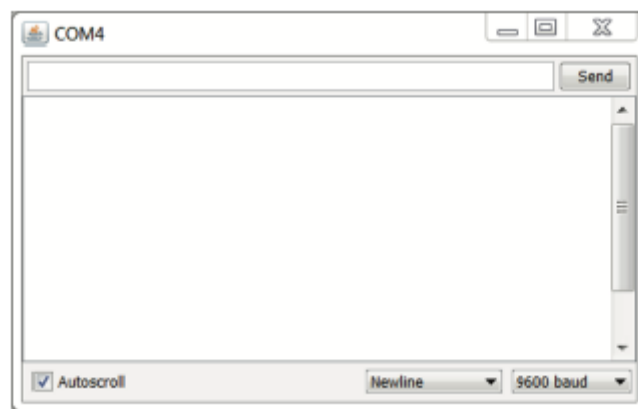
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  int val = analogRead(0);
  int por = map(val, 0, 1023, 0, 100);
  Serial.print("Valor analogico: ");
  Serial.print(val);
  Serial.print(" Porcentaje: ");
  Serial.print(por);
  Serial.println("");
  delay(1000);
}
```

Ejercicio 2 – Cómo “hablar” a Arduino

Para “hablar” con Arduino usaremos el campo de texto del terminal:



Primero, el desplegable ha de estar en “new line”. Esto determina qué se añade al final del comando que envías. A diferencia de otros terminales, éste manda todo el comando de una vez (a la velocidad especificada). Los demás van enviando carácter a carácter conforme los vas tecleando.

Cómo leer información desde un PC vía serie

El puerto serie de Arduino, tiene un buffer. Es decir, Arduino irá encolando todos los comandos que le mandemos. Y los irá procesando en orden. De modo que no es necesario que nos preocupemos por enviar los comandos antes de que termine el “loop” (la función que itera en Arduino).

Cómo decir a Arduino que muestre la información entrante.

Lo más sencillo es usar un “echo”. Para esto nos basta monitorizar el buffer de entrada serie a arduino e imprimir cualquier carácter que éste reciba. Para esto es preciso monitorizar la entrada y hacer que imprima cada carácter que reciba. Para esto:

- Serial.available() nos dice los caracteres que tenemos en el buffer de entrada, si es > 0 entonces tenemos que leer e imprimirlos por pantalla.
- Serial.read() nos da el siguiente caracter disponible en el buffer.

Cada llamada a Serial.read() solo devuelve 1 byte así que hay que leer mientras que Serial.available() devuelva valores mayores que cero. Cuando Serial.read() lee un byte, ese byte se borra del buffer.

```
//Programa de echo

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

void loop() {

  if (Serial.available() > 0)
  {
    data = Serial.read();
    Serial.print(data);
  }

}
```

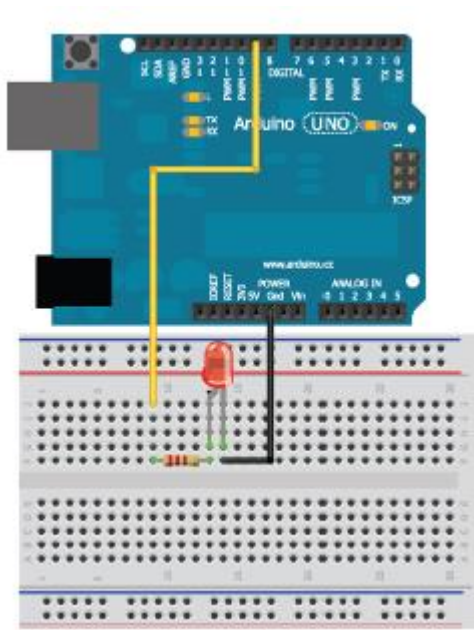
Abre el monitor serie y escribe cualquier cosa. Tan pronto como escribas y pulses “Send” se mostrará en el monitor serie.

Sin embargo, habrá ocasiones en las que quieras enviar valores numéricos (y no bytes) al Arduino.

Por ejemplo, si quieres encender un LED cuando envíes un 1, puedes if (Serial.read() == '1'). Las " indican que el valor debe ser tratado como un carácter.

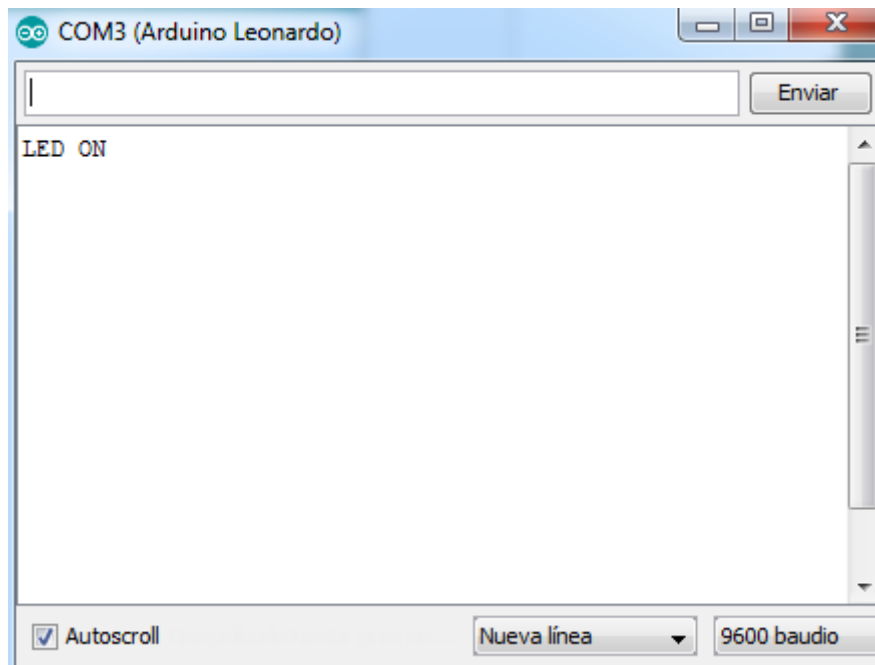
Otra opción es convertir cada character entrante en un entero int val = Serial.read() - '0'. Sin embargo esto no funciona si lo que pretendes es mandar números mayores que 9. Para esto tenemos la función parseInt() que extrae enteros de un canal de comunicaciones serie.

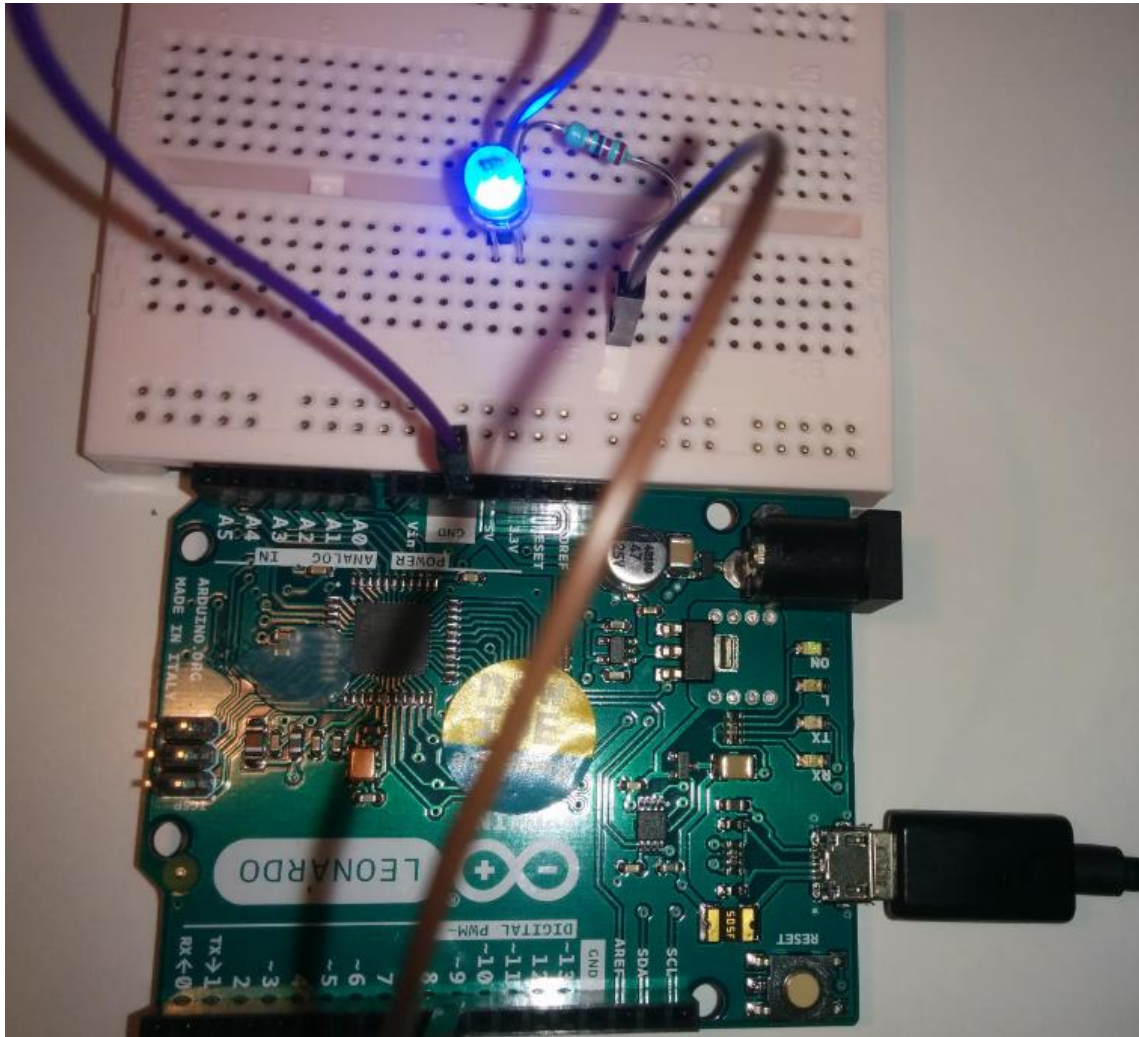
Haz este montaje:



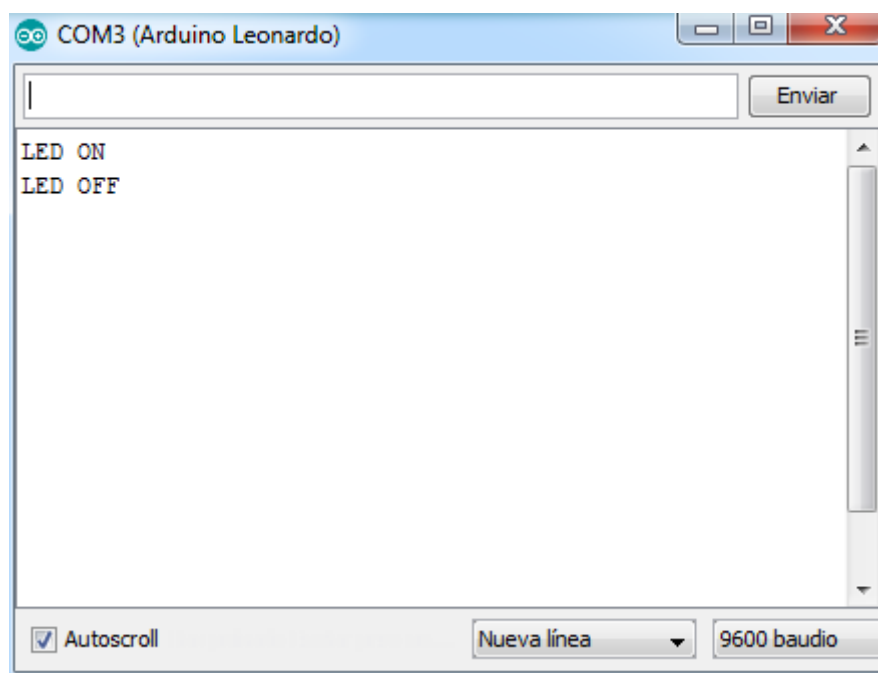
A continuación se muestra el montaje realizado con el led encendido y apagado, además del estado del monitor serie en cada ocasión:

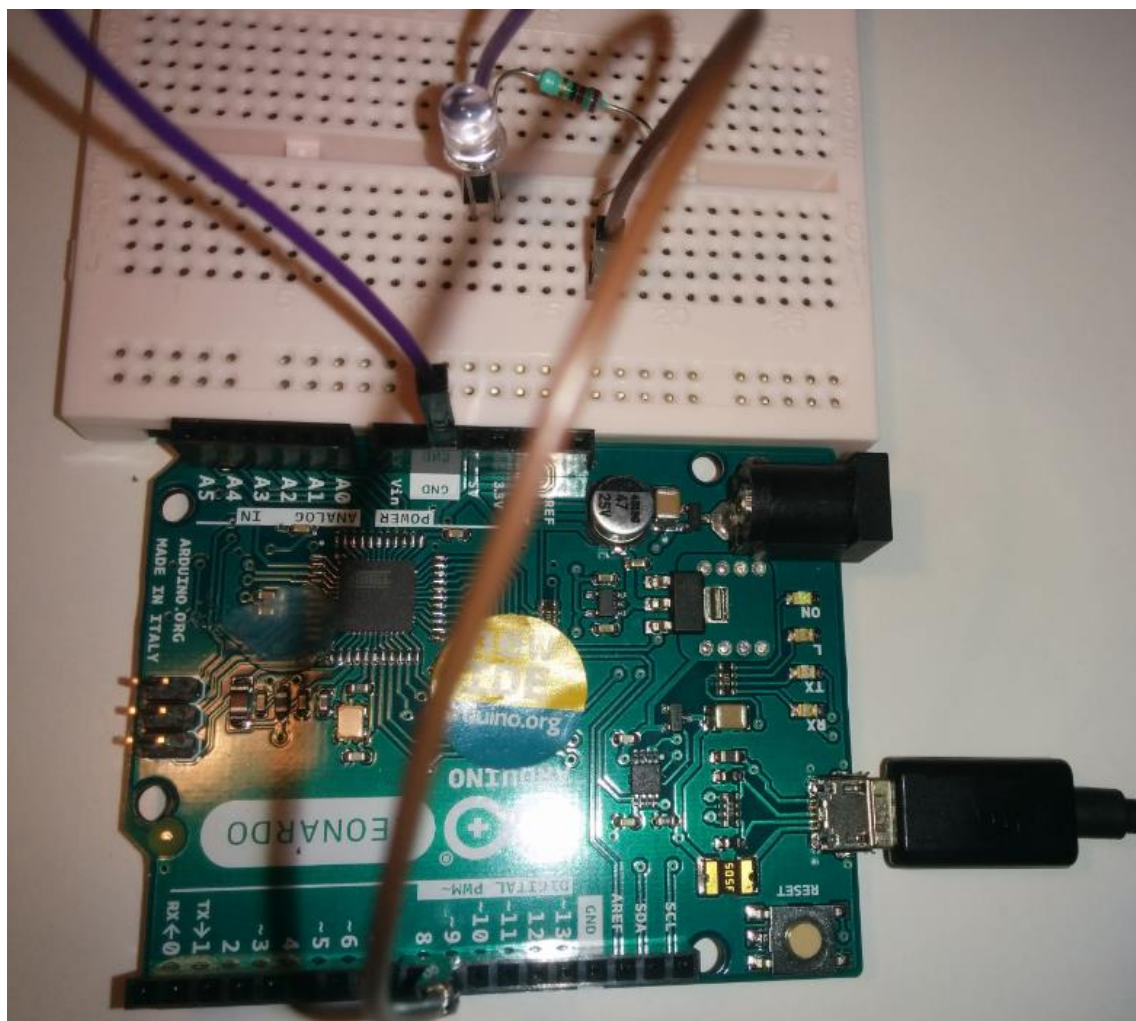
Enviamos un 1 mediante el campo de texto del monitor serie para encender el led





Enviamos un 0 mediante el campo de texto del monitor serie para apagar el led





El código que he utilizado es el siguiente:

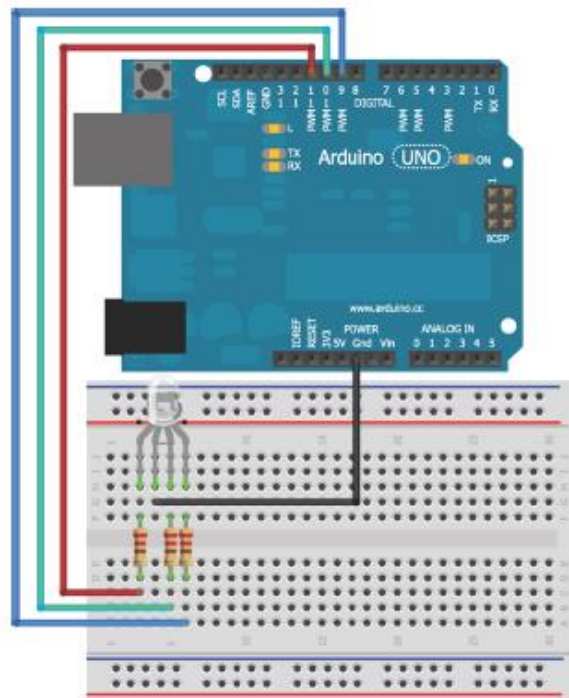
ejercicio_02

```
const int LED=9;
char data;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available() > 0){
    data = Serial.read();
    if (data == '1'){
      digitalWrite(LED, HIGH);
      Serial.println("LED ON");
    }
    else if (data == '0'){
      digitalWrite(LED, LOW);
      Serial.println("LED OFF");
    }
  }
}
```

Con el siguiente montaje vamos a explorar cómo se mandan comandos separados por comas para enviar comandos a diversos dispositivos a la vez.



Vamos a mandar tres bytes, valores (0–255) para configurar el brillo de cada LED del LED de color. Por ejemplo, para configurar todos los colores a su máximo brillo, enviaríamos “255,255,255”.

Retos:

- Hay que diferenciar entre números y comas.
- Hay que convertir los caracteres en números para usar las funciones `analogWrite()`.
- Los valores pueden tener uno, dos o tres caracteres.

`Serial.parseInt()` : Cada llamada a esta función lee hasta que encuentra un valor no numérico en el buffer. Y convierte lo leído en entero.

Para realizar este ejercicio he usado el siguiente código, proporcionado en el guión:

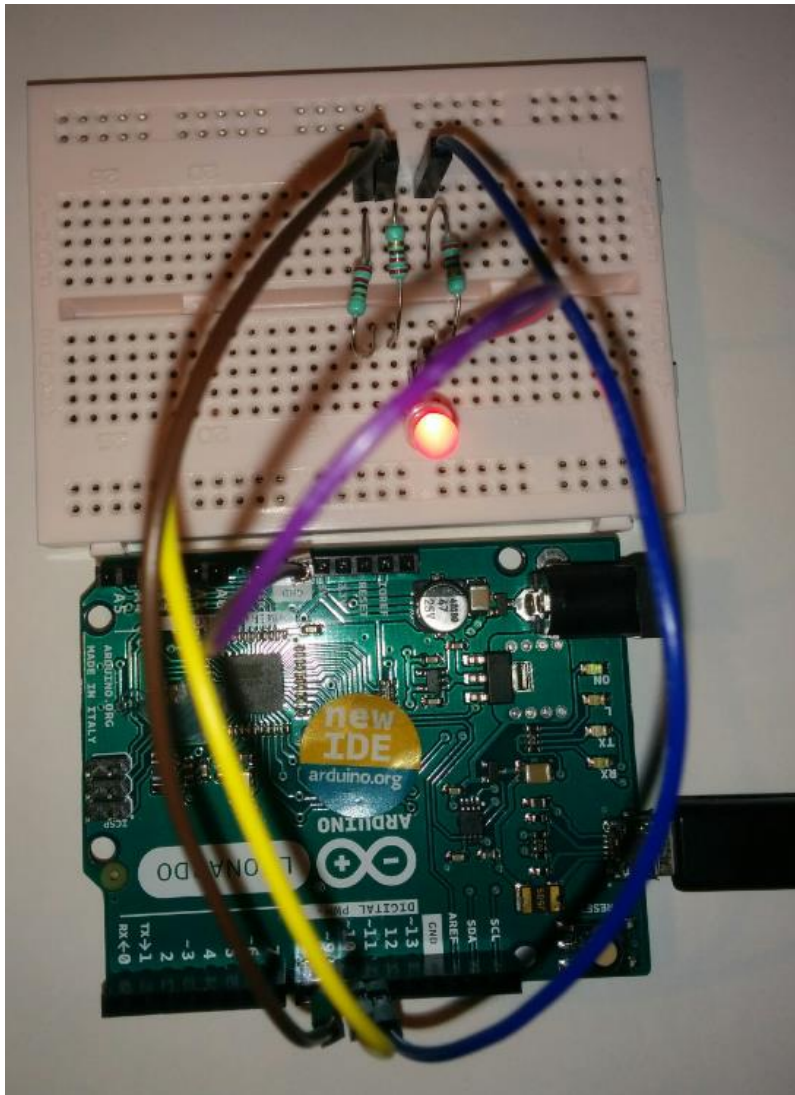
```
const int RED = 11;
const int GREEN = 9;
const int BLUE = 10;
// Valores para el nivel RGB
int rval = 0;
int gval = 0;
int bval = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);
}

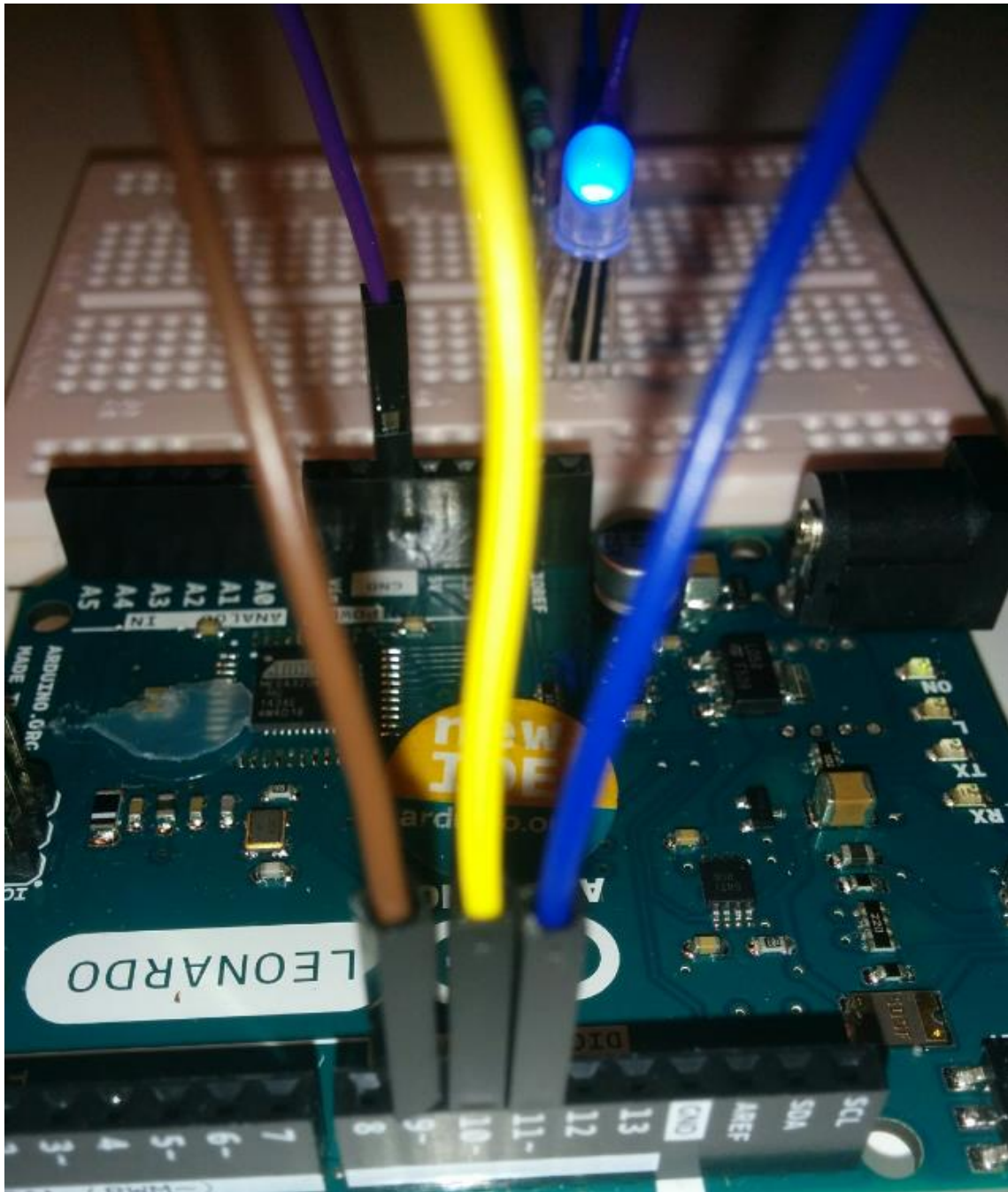
void loop() {
  // put your main code here, to run repeatedly:
  while (Serial.available() > 0) {
    rval = Serial.parseInt();
    gval = Serial.parseInt();
    bval = Serial.parseInt();
    if (Serial.read() == '\n') {
      analogWrite(RED, rval);
      analogWrite(GREEN, gval);
      analogWrite(BLUE, bval);
    }
  }
}
```

Para que coincidieran las variables **Red**, **Green** y **Blue** con sus respectivos colores he tenido que modificar el código, **intercambiando** entre **Green** y **Blue** los pines de conexión en la placa Arduino.

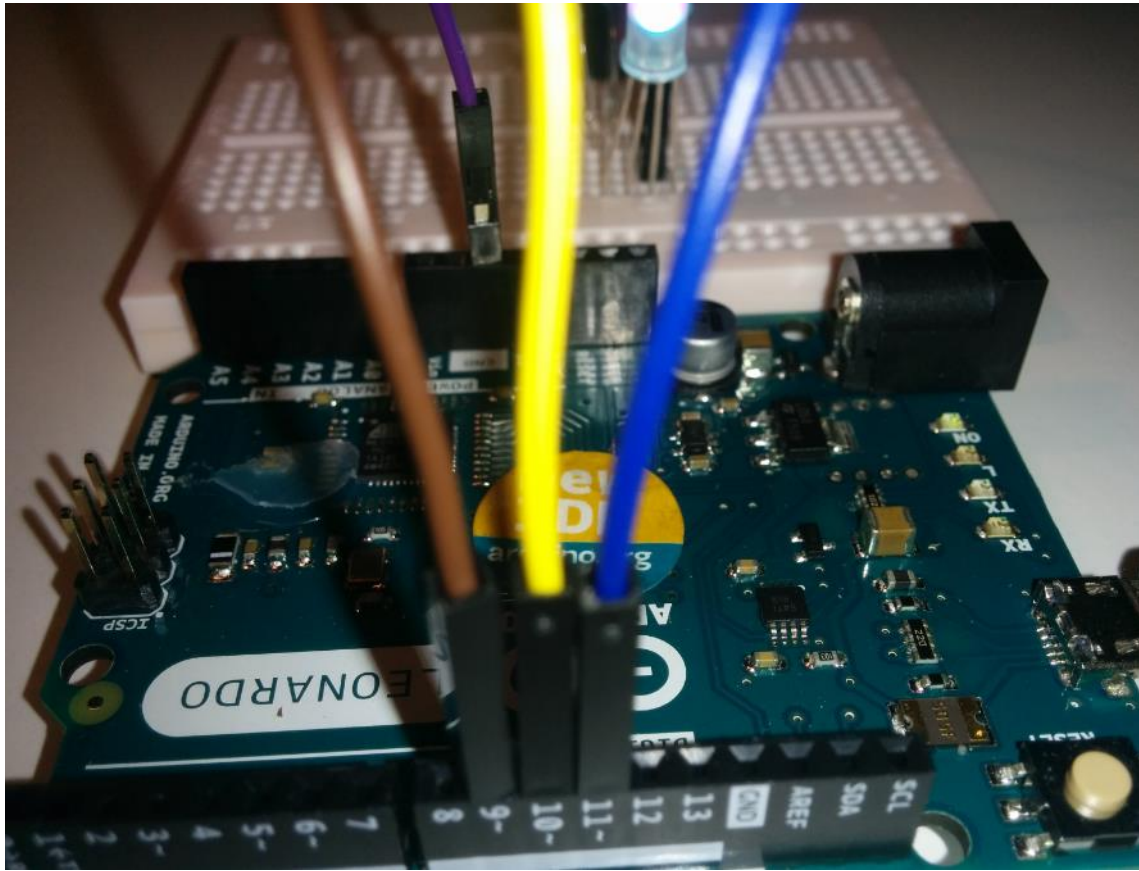
Introduciendo desde el monitor serie el comando 255,0,0, el led se activa en rojo:



Introduciendo desde el monitor serie el comando 0,0,255, el led se activa en azul:



Introduciendo desde el monitor serie el comando 255,255,255, el led se activa en blanco:



Ejercicio 3 – Cómo comunicarse con una aplicación de escritorio

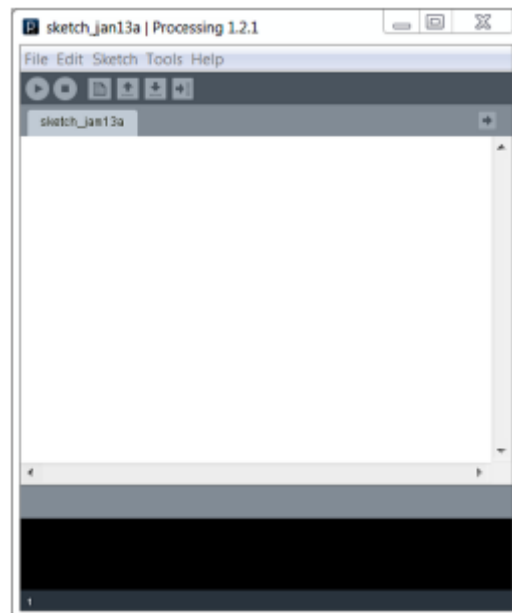
No siempre se usa el monitor serie, hay usos más interesantes. Para las pruebas que vamos a hacer para conectar arduino con aplicaciones vamos a usar Processing.

Cómo comunicarse con Processing

Processing tiene una interfaz de programación muy simple. Similar a la de arduino.

Cómo instalar processing

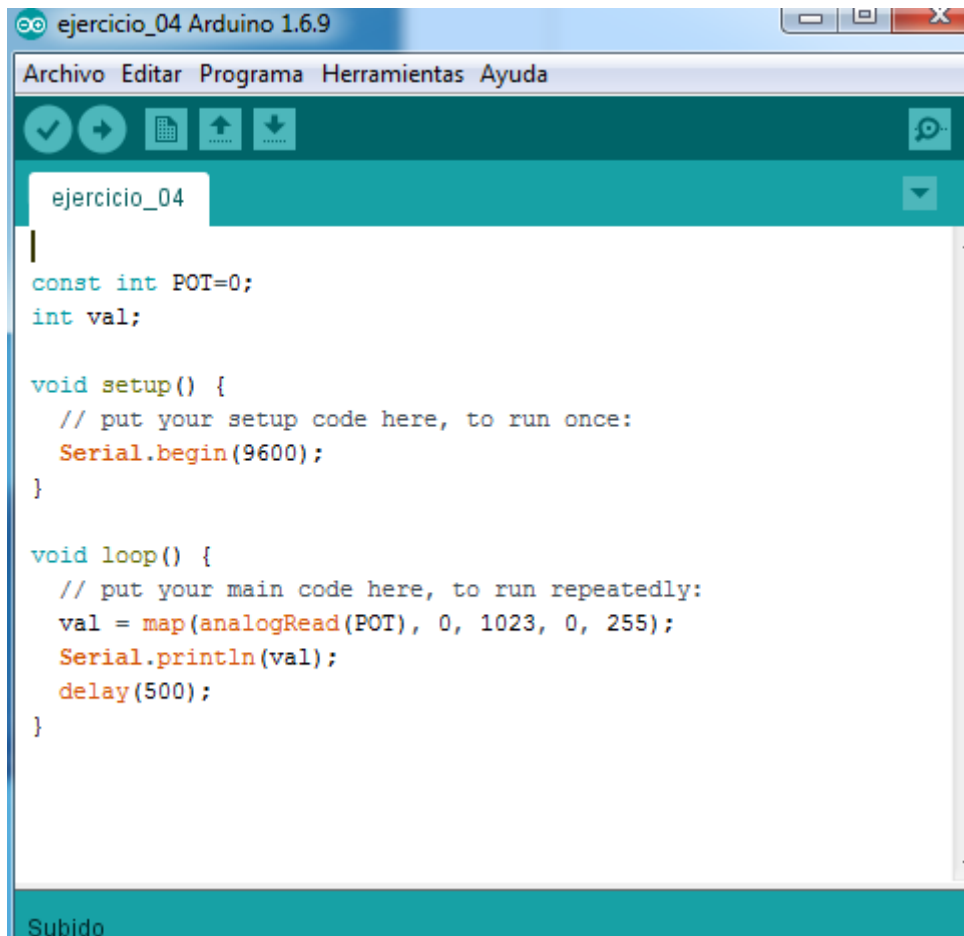
Hay que ir a <http://processing.org/download/> descargar y descomprimir.



Para el primer experimento con Processing, vamos a usar un potenciómetro conectado a Arduino. Y con él vamos a controlar el color de la ventana de una aplicación que se está ejecutando en el PC. Haz el montaje con el potenciómetro.

Los valores por el puerto serie (los valores del potenciómetro) hay que enviarlos con una cadencia de 50ms para que le dé tiempo a Processing a recuperarlo.

El código usado es el siguiente:



The screenshot shows the Arduino IDE interface with the file 'ejercicio_04' open. The code is as follows:

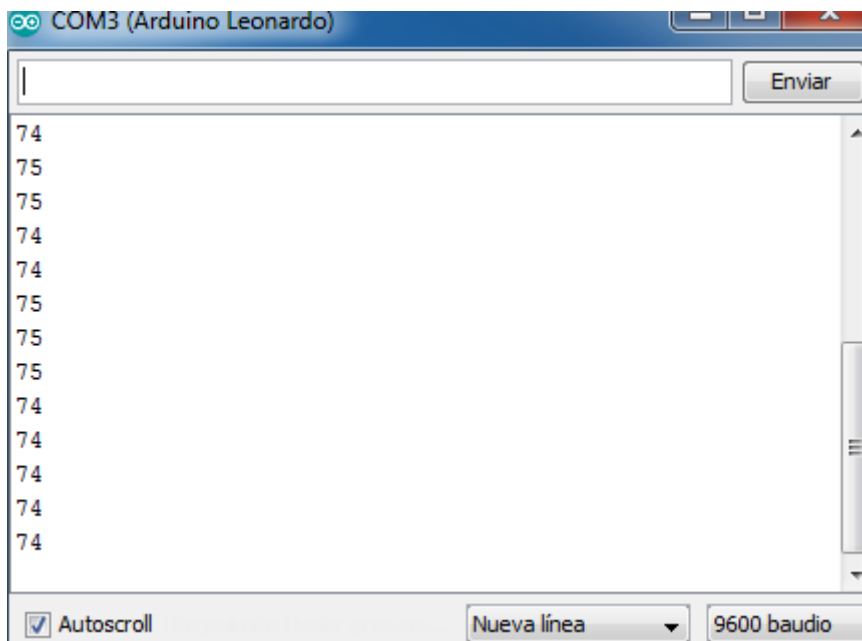
```
const int POT=0;
int val;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  val = map(analogRead(POT), 0, 1023, 0, 255);
  Serial.println(val);
  delay(500);
}
```

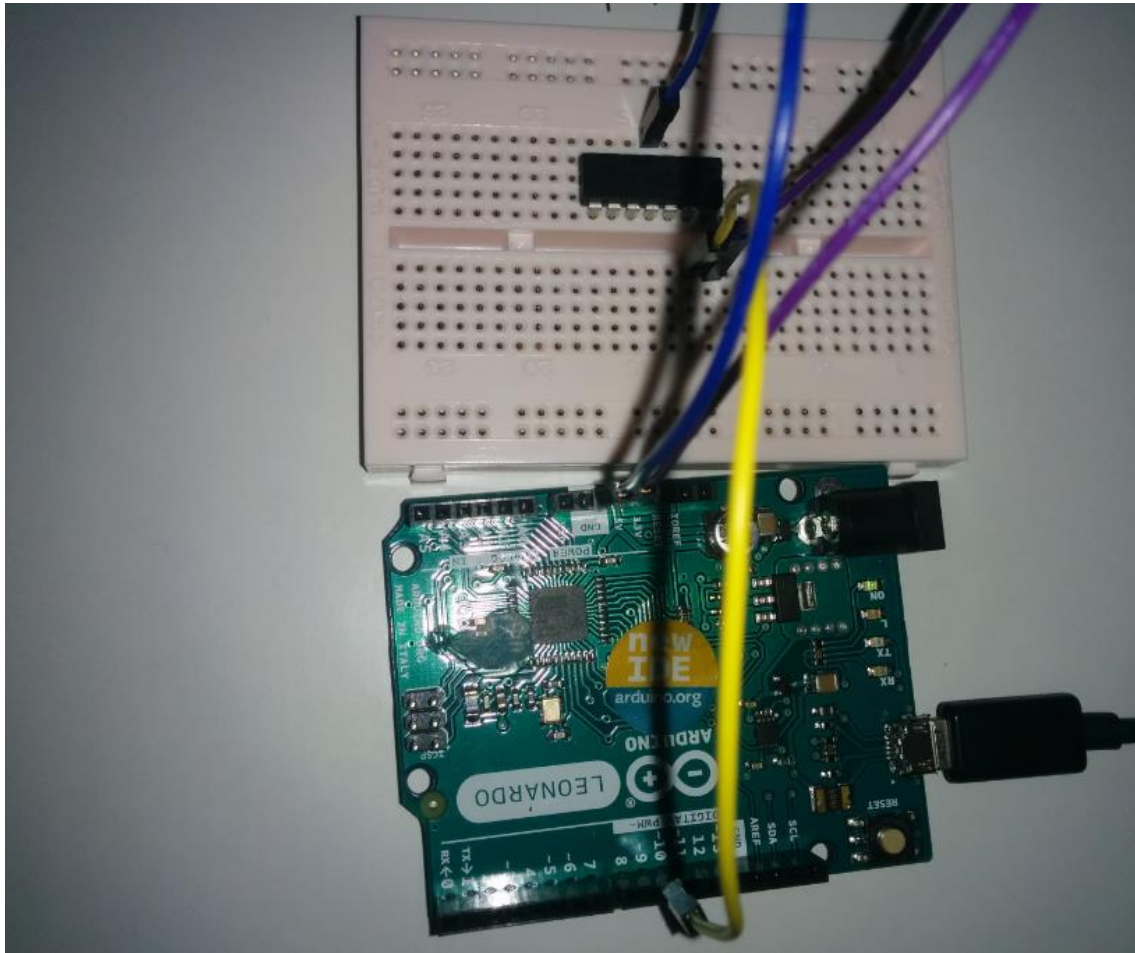
At the bottom of the IDE, the status bar indicates 'Subido' (Uploaded).

Y la salida que éste produce es la siguiente:



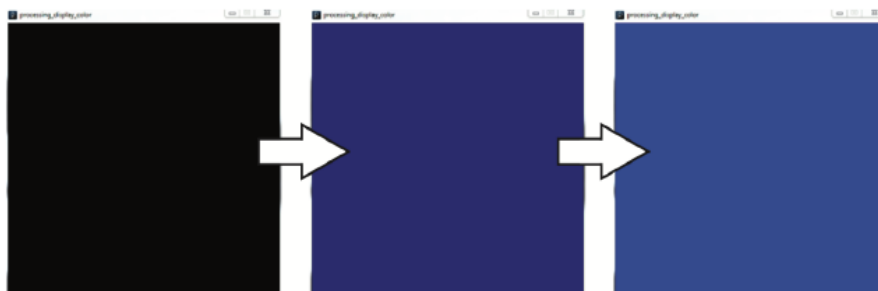
The screenshot shows the Arduino Serial Monitor window for 'COM3 (Arduino Leonardo)'. The output consists of a sequence of values: 74, 75, 75, 74, 74, 75, 75, 75, 74, 74, 74, 74, 74. The 'Enviar' (Send) button is visible at the top right. At the bottom, the 'Autoscroll' checkbox is checked, the 'Nueva línea' (New line) dropdown is set to 'Nueva línea', and the baud rate is set to '9600 baudio'.

A continuación podemos observar el montaje:



Ahora viene la parte interesante: escribir el programa usando Processing para hacer algo más a parte de mostrar valores, con los datos que nos lleguen.

Tras cargar el código en el IDE y tras haber configurado el puerto serie apropiadamente, asegúrate de que el monitor serie de arduino no está abierto. Sólo un programa puede tener acceso al puerto serie. Ejecuta la aplicación. Cuando lo hagas, aparecerá una ventana. Con el potenciómetro verás como cambia de color.

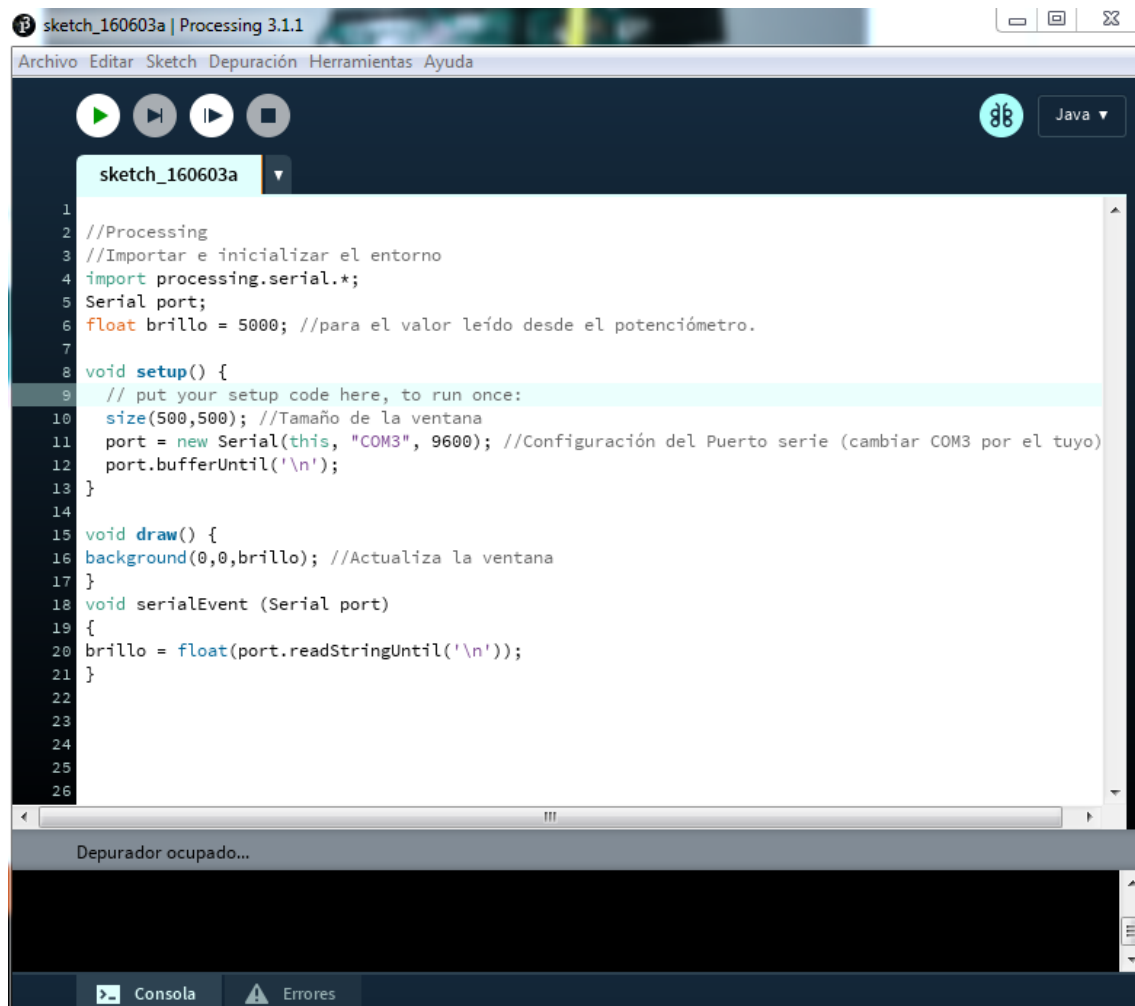


En lo que respecta al código, la librería de comunicación con el puerto serie no se carga, hay que importarla, para esto se usa "import processing.serial.*;" Y con Serial port, estamos creando un objeto para controlar el puerto serie.

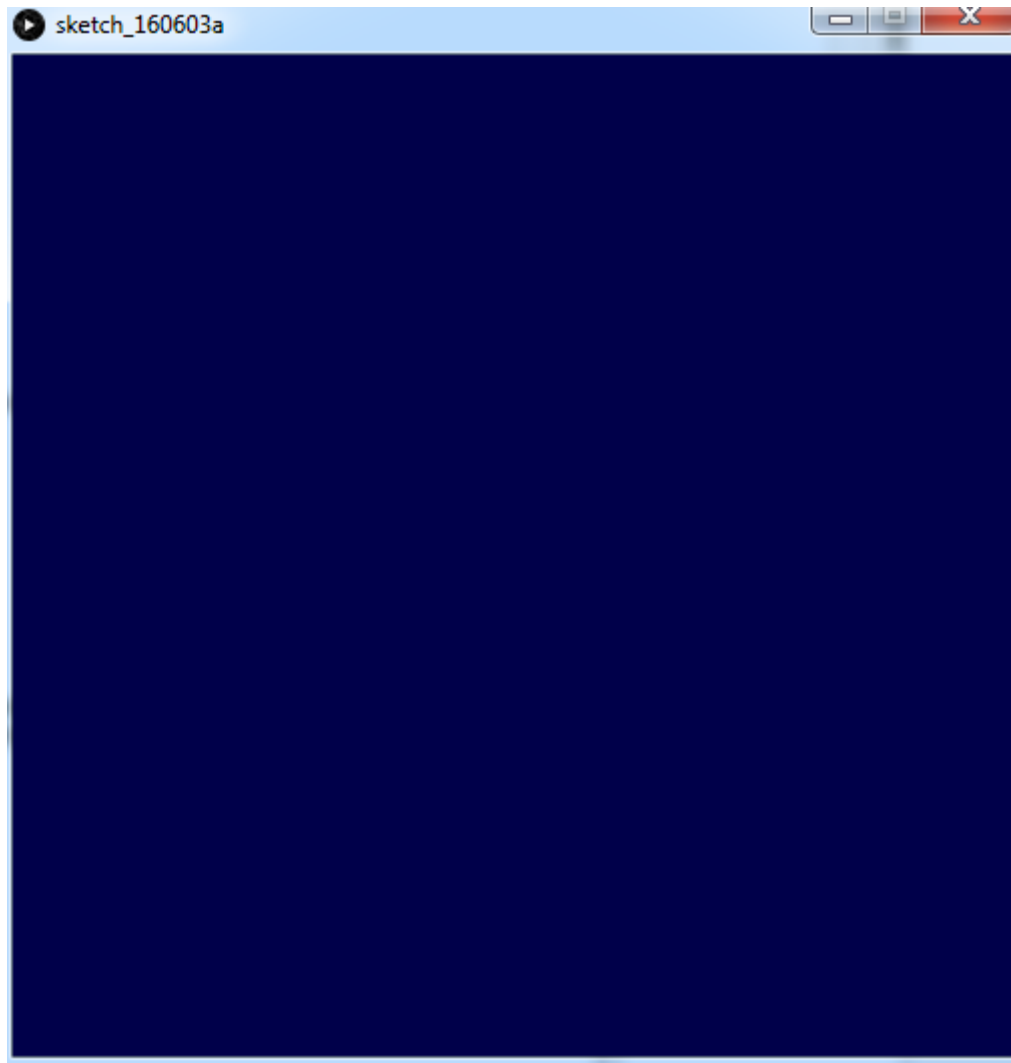
Processing, en lugar de tener una función "loop()" como Arduino, tiene otras funciones especiales. Este programa usa las funciones draw() y serialEvent(). La función draw() es similar a la función loop() de arduino. Se ejecuta constantemente y actualiza el display. La función background() establece el color de la ventana. En este caso, el potenciómetro controla el nivel de azul y el rojo y el verde están a cero.

SerialEvent() se invoca cuando la condición bufferUntil() que se configura en el setup() se da.

Con el mismo montaje anterior, ejecutamos el siguiente código en processing :



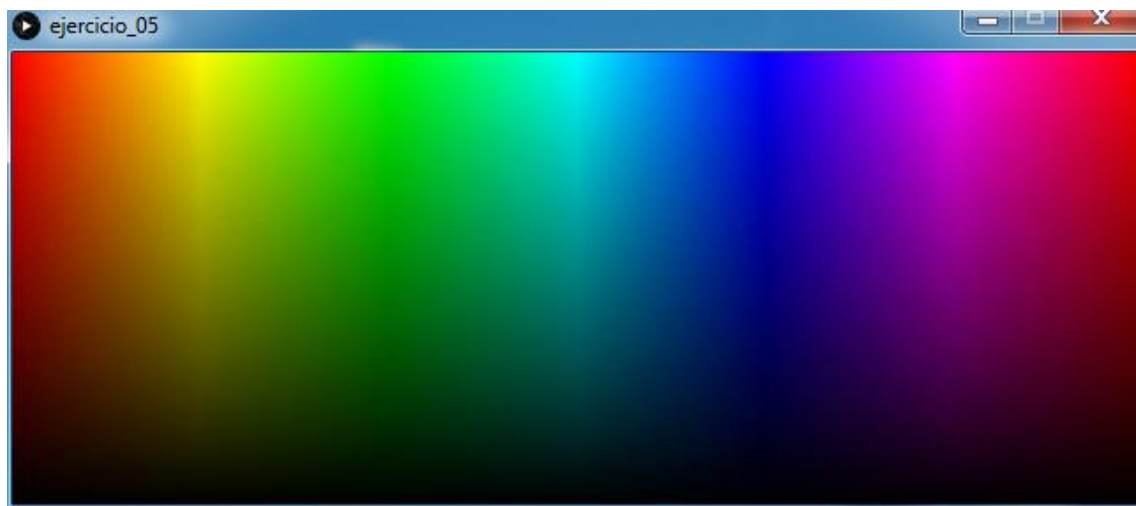
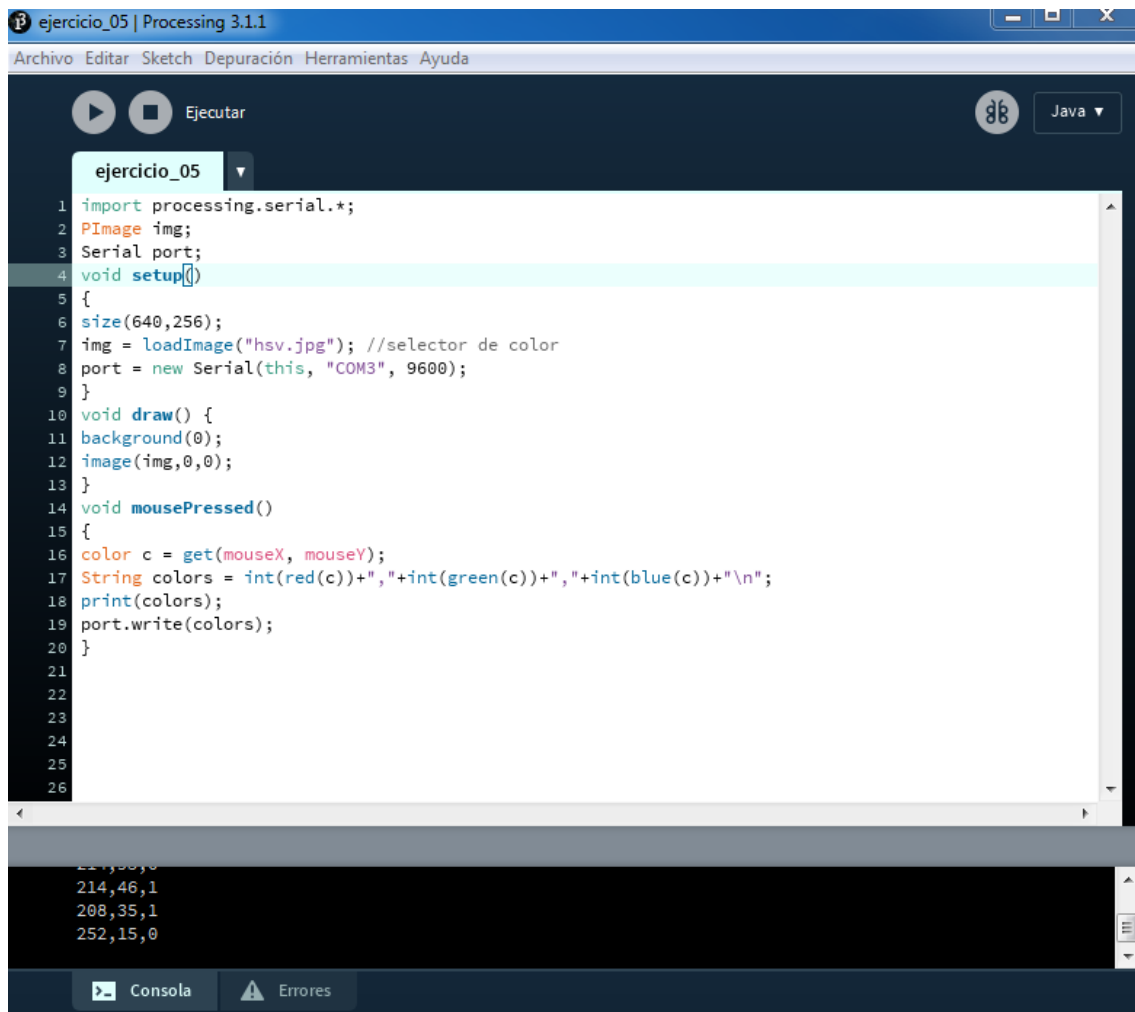
```
1 //Processing
2 //Importar e inicializar el entorno
3 import processing.serial.*;
4 Serial port;
5 float brillo = 5000; //para el valor leído desde el potenciómetro.
6
7
8 void setup() {
9   // put your setup code here, to run once:
10   size(500,500); //Tamaño de la ventana
11   port = new Serial(this, "COM3", 9600); //Configuración del Puerto serie (cambiar COM3 por el tuyo)
12   port.bufferUntil('\n');
13 }
14
15 void draw() {
16   background(0,0,brillo); //Actualiza la ventana
17 }
18 void serialEvent (Serial port)
19 {
20   brillo = float(port.readStringUntil('\n'));
21 }
22
23
24
25
26
```



Enviar datos desde processing al Arduino

Carga el programa que mandaba los tres valores de color separados por comas. Ahora vamos a hacer lo mismo pero con un selector de color. Carga el siguiente programa

Primero cargamos el código del ejercicio 2b en Arduino (ejercicio RGB), después cargamos el siguiente código en processing



Para ver este montaje en funcionamiento en realizado un pequeño vídeo:

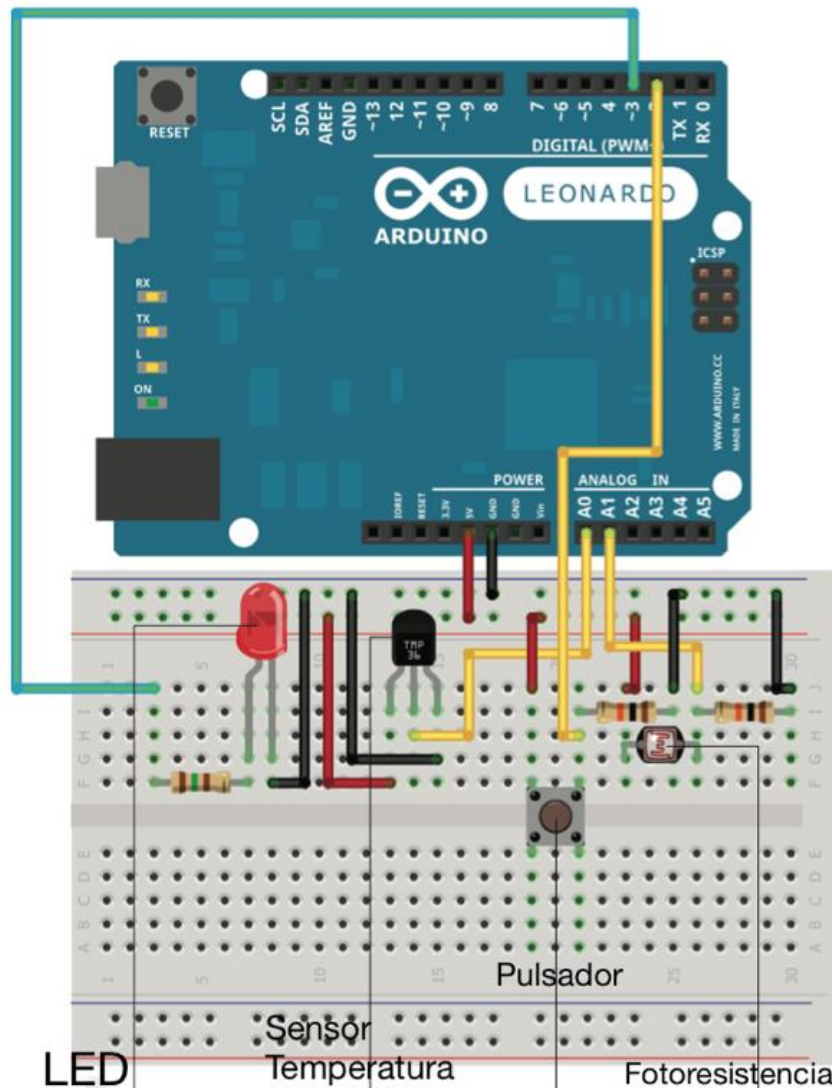
https://www.dropbox.com/s/dk75dqsw0w8r6f0/VID_20160603_162946.mp4?dl=0

Ejercicio 4 – Emulación de un teclado y de un ratón

Emulación de un teclado y de un ratón

Crea un documento con extensión “csv” usando el editor de documentos / textos que quieras. A continuación

realiza el siguiente montaje :



El botón sirve para activar o desactivar el logueado de información de los sensores. Si el botón activa el logueado de sensores, entonces los manda al PC, como si los estuvieras escribiendo desde el teclado, separando por comas sus valores. El LED debe encenderse para informar de que los valores se están leyendo. Como queremos que Arduino esté sondeando el estado del botón, aquí no podremos usar la función `delay()` para esperar 1000ms entre cada actualización. Para ello podemos usar la función `millis()` . Esta función permite crear un espacio no bloqueante de 1 ms entre transmisiones.


```

{
  digitalWrite(LED, HIGH);          //Turn the LED on
  if (millis() % 1000 == 0)         //If time is multiple
                                    //of 1000ms
  {
    int temperature = analogRead(TEMP); //Read the temperature
    int brightness = analogRead(LIGHT); //Read the light level
    Keyboard.print(counter);           //Print the index number
    Keyboard.print(",");               //Print a comma
    Keyboard.print(temperature);       //Print the temperature
    Keyboard.print(",");               //Print a comma
    Keyboard.println(brightness);      //Print brightness, newline
    counter++;                         //Increment the counter
  }
}
else
{
  digitalWrite(LED, LOW); //If logger not running, turn LED off
}
}

/*
 * Debouncing Function
 * Pass it the previous button state,
 * and get back the current debounced button state.
 */
boolean debounce(boolean last)
{
  boolean current = digitalRead(BUTTON); //Read the button state
  if (last != current)                   //If it's different...
  {
    delay(5);                           //Wait 5ms
    current = digitalRead(BUTTON);       //Read it again
  }
  return current;                       //Return the current
                                        //value
}

```

Fijaos en que la emulación de teclado comienza en el setup cuando ponemos `Keyboard.begin()`.

`Keyboard.print()` “escribe” los datos como si fuera un teclado, hacia tu PC.

Podemos, además, configurar Arduino para que lance combinaciones de teclas a nuestros PC. Trata de implementar un programa que bloquee tu ordenador cuando no haya luz. Para las combinaciones de teclas puedes entrar en :

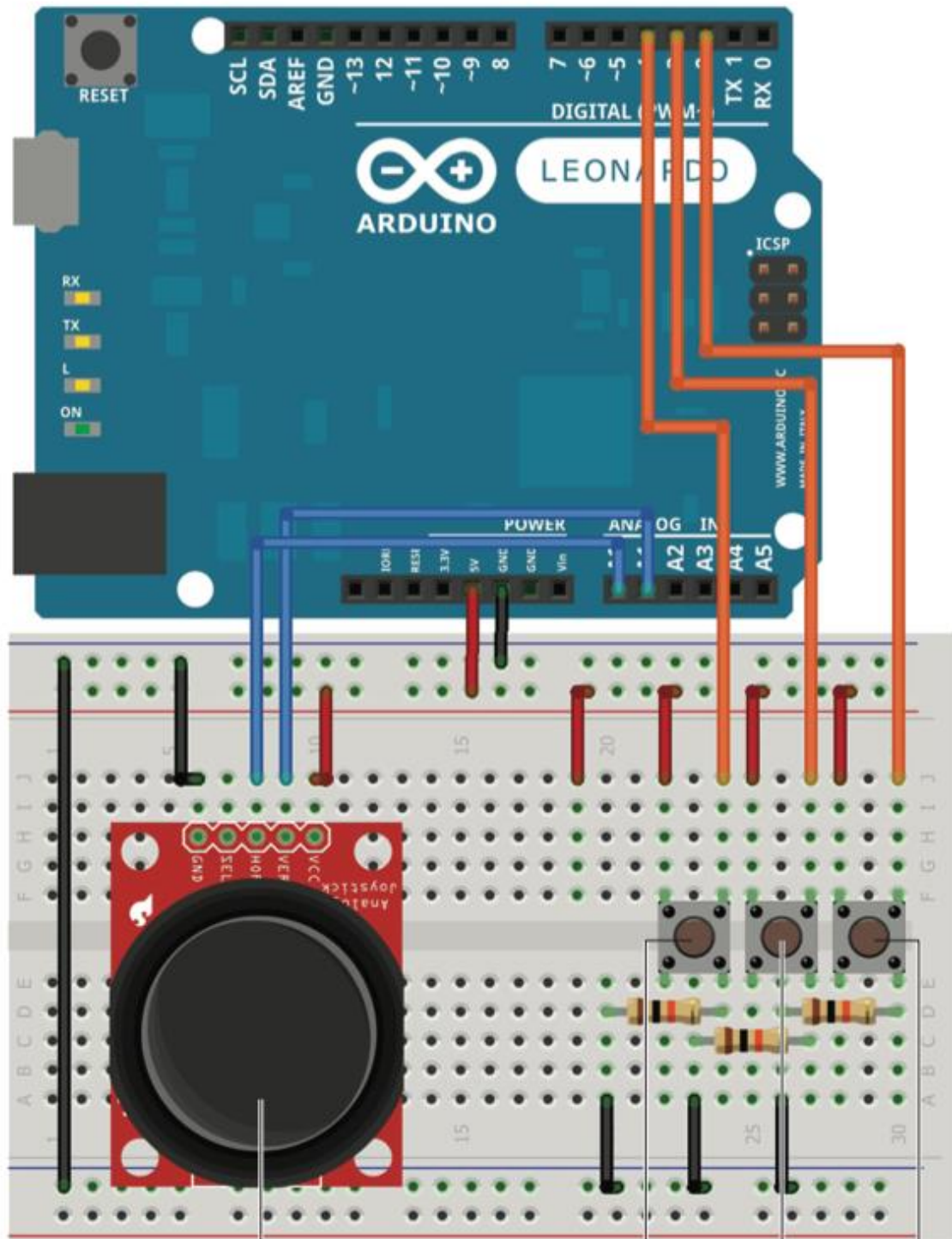
<http://arduino.cc/en/Reference/KeyboardModifiers> para ver las teclas especiales.

Como pista:

Tendrás que usar la función `keyboard.press` y la función `keyboard.releaseall`. Ya que si quieres pulsar varias teclas a la vez tendrás que:

`Keyboard.press('a');`

```
Keyboard.press('b');  
delay(100);
```



```

const int LEFT_BUTTON   =4; //Input pin for the left button
const int MIDDLE_BUTTON =3; //Input pin for the middle button
const int RIGHT_BUTTON  =2; //Input pin for the right button
const int X_AXIS        =0; //Joystick x-axis analog pin
const int Y_AXIS        =1; //Joystick y-axis analog pin

void setup()
{
    Mouse.begin();
}

void loop()
{
    int xVal = readJoystick(X_AXIS);           //Get x-axis movement
    int yVal = readJoystick(Y_AXIS);           //Get y-axis movement

    Mouse.move(xVal, yVal, 0);                  //Move the mouse

    readButton(LEFT_BUTTON, MOUSE_LEFT);       //Control left button
    readButton(MIDDLE_BUTTON, MOUSE_MIDDLE);    //Control middle button
    readButton(RIGHT_BUTTON, MOUSE_RIGHT);      //Control right button

    delay(5);                                  //This controls responsiveness
}

//Reads joystick value, scales it, and adds dead range in middle
int readJoystick(int axis)
{
    int val = analogRead(axis);                //Read analog value
    val = map(val, 0, 1023, -10, 10);          //Map the reading

    if (val <= 2 && val >= -2)                  //Create dead zone to stop mouse
drift
        return 0;

    else
        return val;                          //Return scaled value
}

```

```

}

//Read a button and issue a mouse command
void readButton(int pin, char mouseCommand)
{
    //If button is depressed, click if it hasn't already been clicked
    if (digitalRead(pin) == HIGH)
    {
        if (!Mouse.isPressed(mouseCommand))
        {
            Mouse.press(mouseCommand);
        }
    }
    //Release the mouse if it has been clicked.
    else
    {
        if (Mouse.isPressed(mouseCommand))
        {
            Mouse.release(mouseCommand);
        }
    }
}
}

```