

*Tutoriales*  
*Diseño en*  
***Medios***  
***Interactivos***

# 02

## *Conexión Arduino a p5.js por el puerto serial*

- a.*** Enviar datos desde Arduino
- b.*** Recibir y enviar datos al navegador
- c.*** Instalar librería en p5.js
- d.*** Leer y visualizar datos en p5.js

## ***Materiales***

Arduino\*

Cable de conexión Arduino

Potenciómetro o fotoresistencia

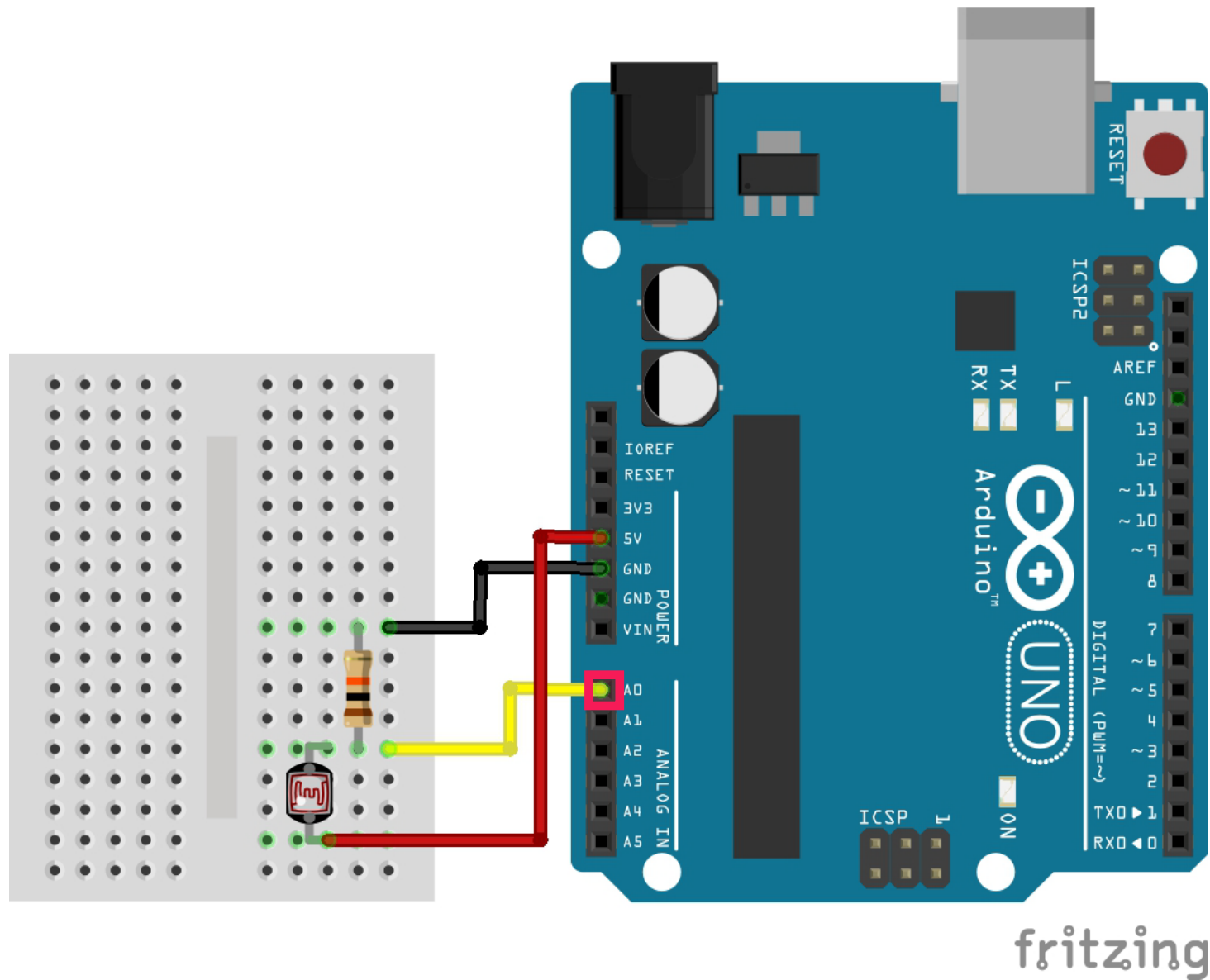
\*Tener el software de arduino instalado

([arduino.cc/en/main/software](https://arduino.cc/en/main/software))

# ***a.***

## ***Enviar datos desde Arduino***

# 1. Preparar el circuito: conectar potenciómetro o fotoresistencia al arduino



El sensor debe estar conectado al pin **A0** de entrada analógica

## 2. En Arduino escribir un programa que lee los datos del sensor y los imprime en el monitor serie

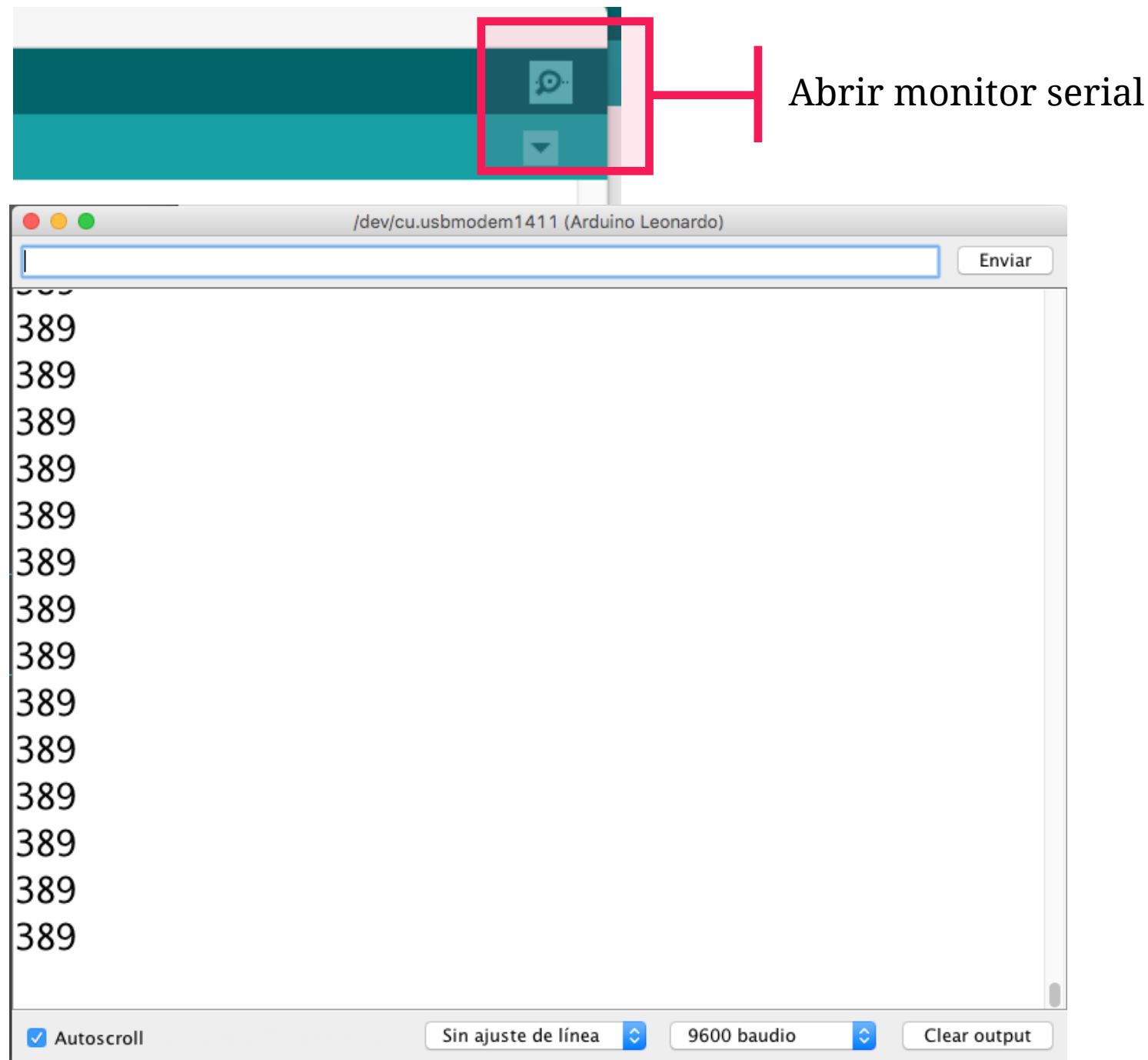
```
const int analogIn = A0; //Pin que lee los datos del sensor
int sensorValue = 0;      //datos recibidos del sensor

void setup() {
  //Inicia la comunicación serial en 9600 bps:
  Serial.begin(9600);
}
void loop() {
  //lee los datos recibidos del sensor y los guarda en la variable sensorValue
  sensorValue = analogRead(analogIn);
  //Imprime los resultados en el monitor serial
  Serial.println(sensorValue);
  delay(2);
}
```

Comando que envía los datos recibidos en formato **ASCII** al puerto serial y permite verlos en el monitor.  
Más sobre esto acá: [ascii.cl/es](https://ascii.cl/es)

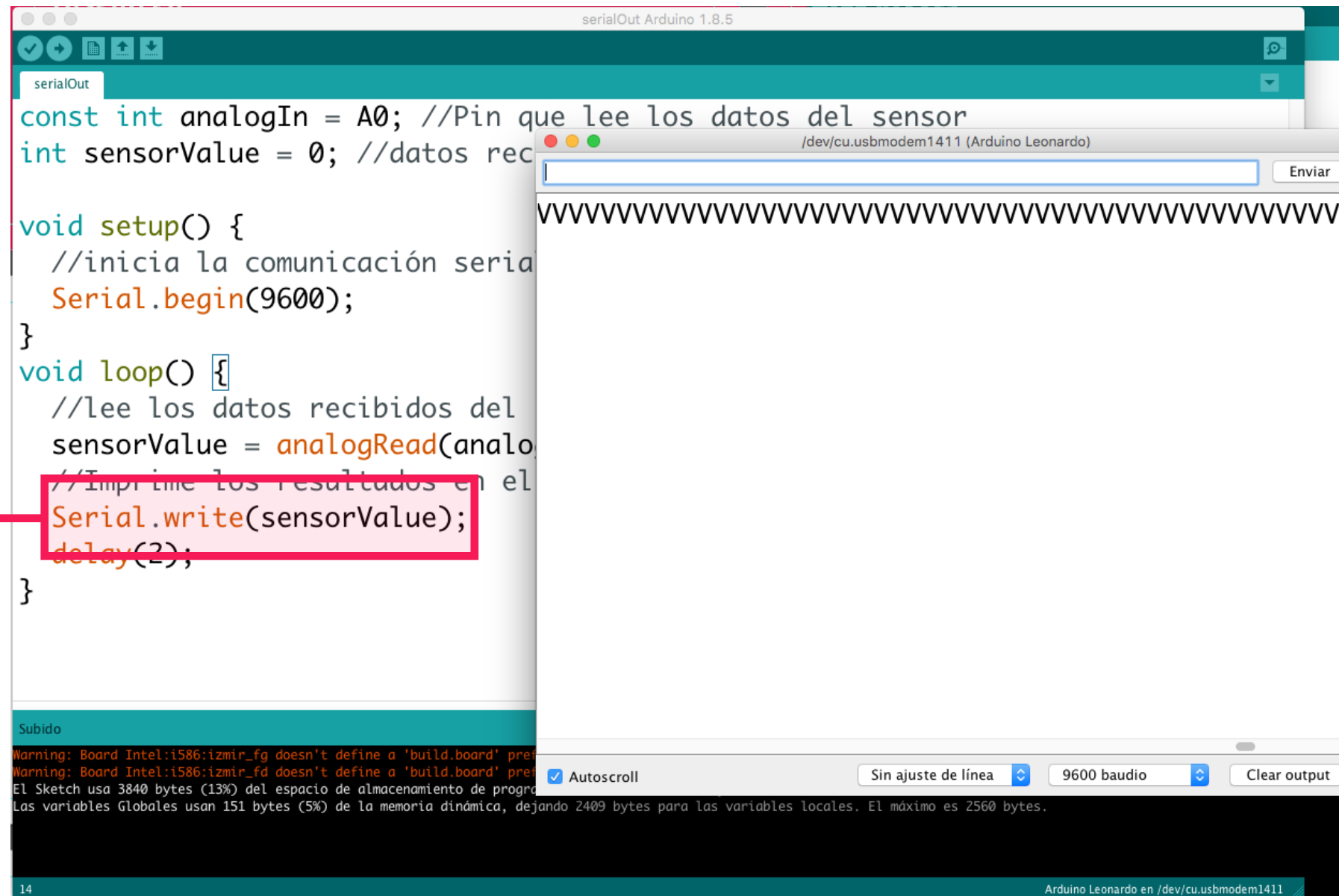
Este código se puede ver y descargar en: [laurajunco.github.io/medios/serialOut/serialOut.ino](https://laurajunco.github.io/medios/serialOut/serialOut.ino)

- Al correr el programa y abrir el monitor serie se ven los valores enviados por el sensor



Un **puerto serie** es una interfaz de comunicaciones de datos digitales, utilizado por computadoras donde la información es transmitida enviando **un solo bit a la vez**.

## 4. Cambiar la línea **Serial.println()** por **Serial.write()** Correr el programa de nuevo.



The screenshot shows the Arduino IDE interface. The main editor window displays the following code:

```
const int analogIn = A0; //Pin que lee los datos del sensor
int sensorValue = 0; //datos recibidos

void setup() {
  //inicia la comunicación serial
  Serial.begin(9600);
}

void loop() {
  //lee los datos recibidos del sensor
  sensorValue = analogRead(analogIn);
  //Imprime los resultados en el monitor de serie
  Serial.write(sensorValue);
  delay(2);
}
```

The line `Serial.write(sensorValue);` is highlighted with a red box. A red line connects this box to the explanatory text below. The serial monitor window, titled `/dev/cu.usbmodem1411 (Arduino Leonardo)`, shows a continuous stream of 'V' characters, representing the raw byte output of the sensor data.

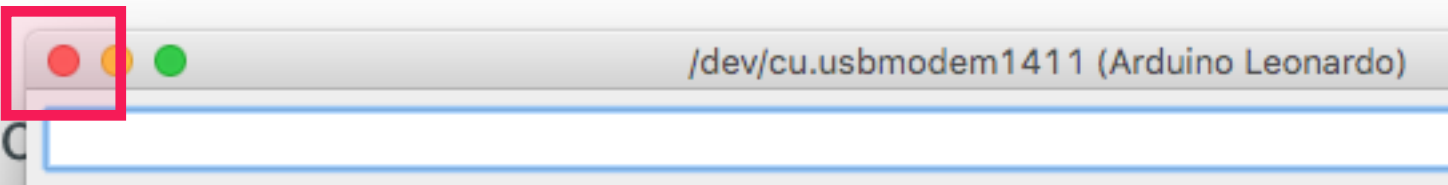
At the bottom of the IDE, the status bar indicates: "Subido", "Warning: Board Intel:i586:izmir\_fg doesn't define a 'build.board' prefix", "Warning: Board Intel:i586:izmir\_fd doesn't define a 'build.board' prefix", "El Sketch usa 3840 bytes (13%) del espacio de almacenamiento de programa", "Las variables Globales usan 151 bytes (5%) de la memoria dinámica, dejando 2409 bytes para las variables locales. El máximo es 2560 bytes.", "14", and "Arduino Leonardo en /dev/cu.usbmodem1411".

**Serial.write()** envía los datos al puerto serie en forma de **bytes**.  
No están traducidos al formato ASCII (el que leemos normalmente)

5. Una vez el programa se haya subido cerrar el monitor serie. Esto debido a que solo una aplicación a la vez puede estar leyendo los datos del puerto serial

```
t int analogIn = A0; //Pin que lee los datos de  
sensorValue = 0; //datos recibidos
```

```
setup() {  
  inicia la co  
  rial.begin(9  
  
  loop() {  
    lee los dato  
    nsorValue =  
    Tmprime los
```



Una vez el sketch se haya subido a la tarjeta, ésta empezará a emitir los datos del sensor. No es necesario mantener la aplicación de Arduino abierta.



## *¿Por qué?*

Usualmente los navegadores no pueden acceder al puerto serial del computador.

Es necesario un programa que reciba los datos del puerto y los envíe al navegador.

# ***b.***

*Recibir y enviar datos al navegador*

1. Ir a la dirección [github.com/vanevery/p5.serialcontrol/releases](https://github.com/vanevery/p5.serialcontrol/releases) y descargar la última versión de **p5.serialcontrol**

The screenshot shows the GitHub release page for 'p5.serialcontrol' version 'Alpha 6'. On the left, there's a sidebar with a 'Pre-release' label, the version '0.0.6', and the user 'bd101ed'. The main content area has the title 'Alpha 6' and a note that 'vanevery released this 16 days ago'. Below this, it says 'Latest updates from p5.serialport, Latest updates from @kaganjd'. The 'Downloads' section lists three items: 'p5.serialcontrol-darwin-x64.zip' (highlighted with a red box and labeled 'Para Mac'), 'p5.serialcontrol-win32-x64.zip' (highlighted with a red box and labeled 'Para Windows'), and 'Source code (zip)'. Below that is 'Source code (tar.gz)'.

Pre-release

0.0.6

bd101ed

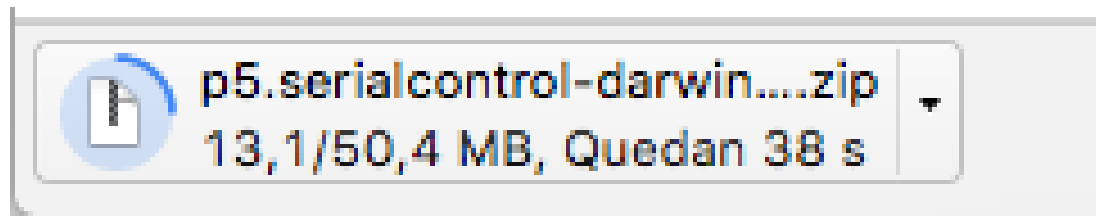
## Alpha 6

vanevery released this 16 days ago

Latest updates from p5.serialport, Latest updates from @kaganjd

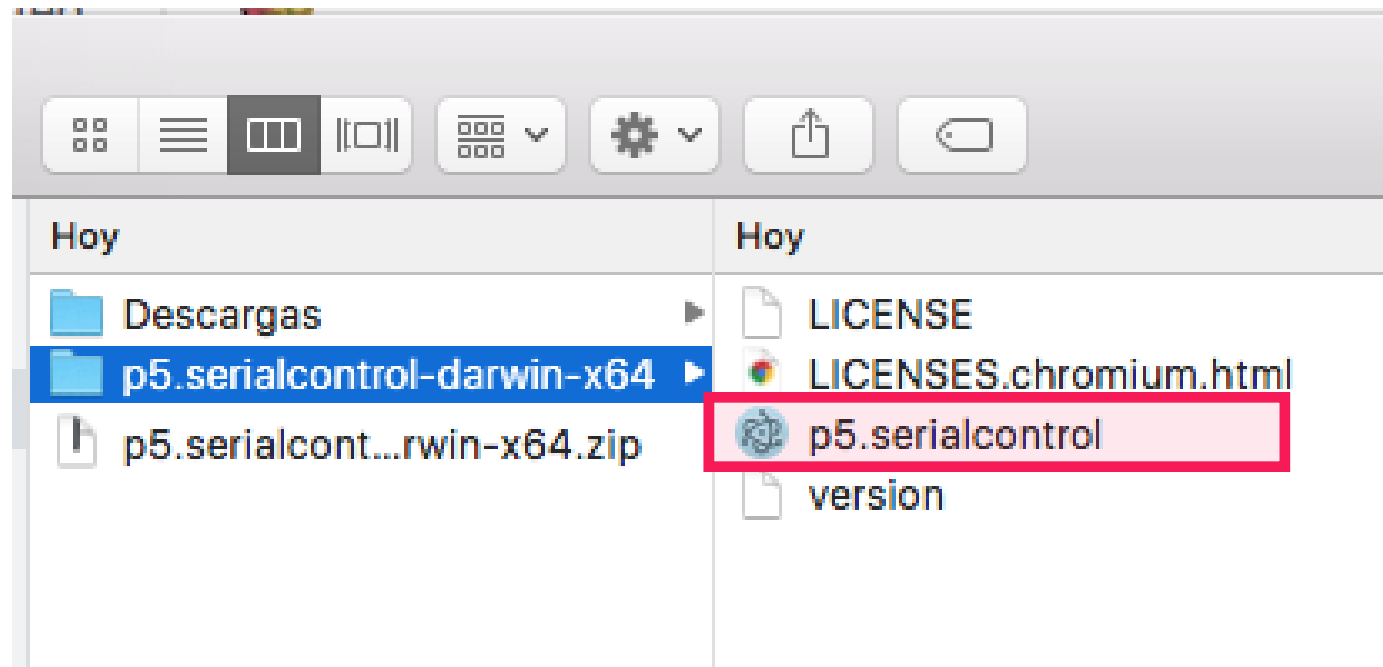
### Downloads

- [p5.serialcontrol-darwin-x64.zip](#) Para Mac
- [p5.serialcontrol-win32-x64.zip](#) Para Windows
- [Source code \(zip\)](#)
- [Source code \(tar.gz\)](#)



Se recomienda utilizar el navegador web **Google Chrome** para seguir estos pasos.

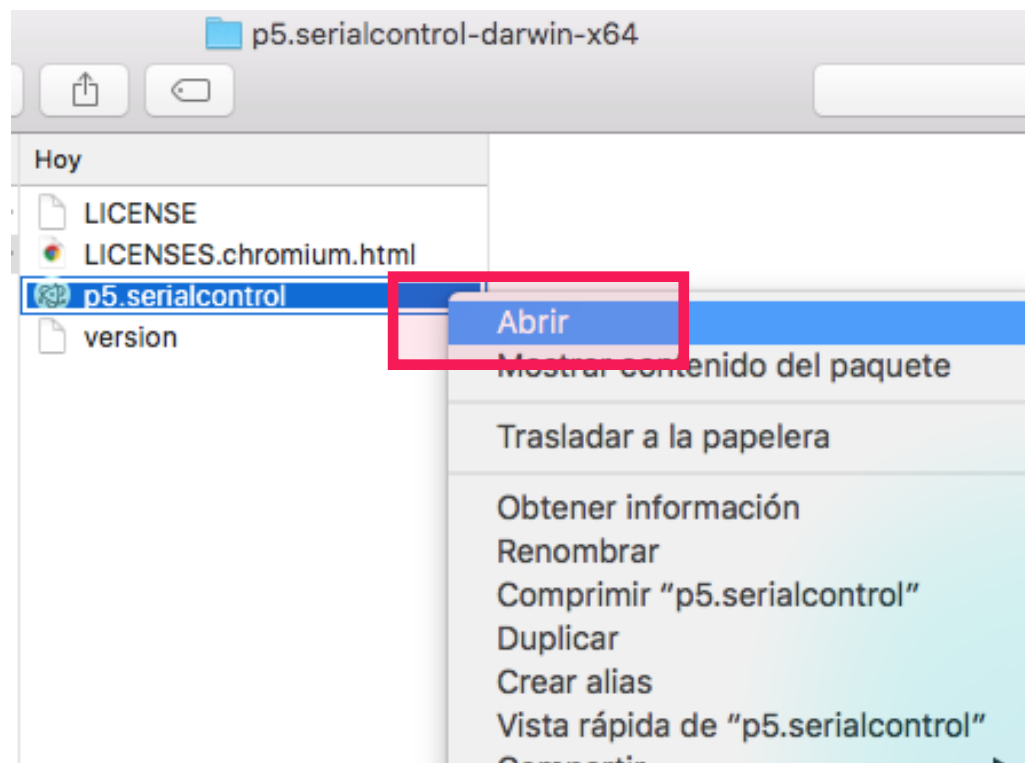
## 2. Descomprimir la carpeta y hacer clic en la aplicación p5.serialcontrol



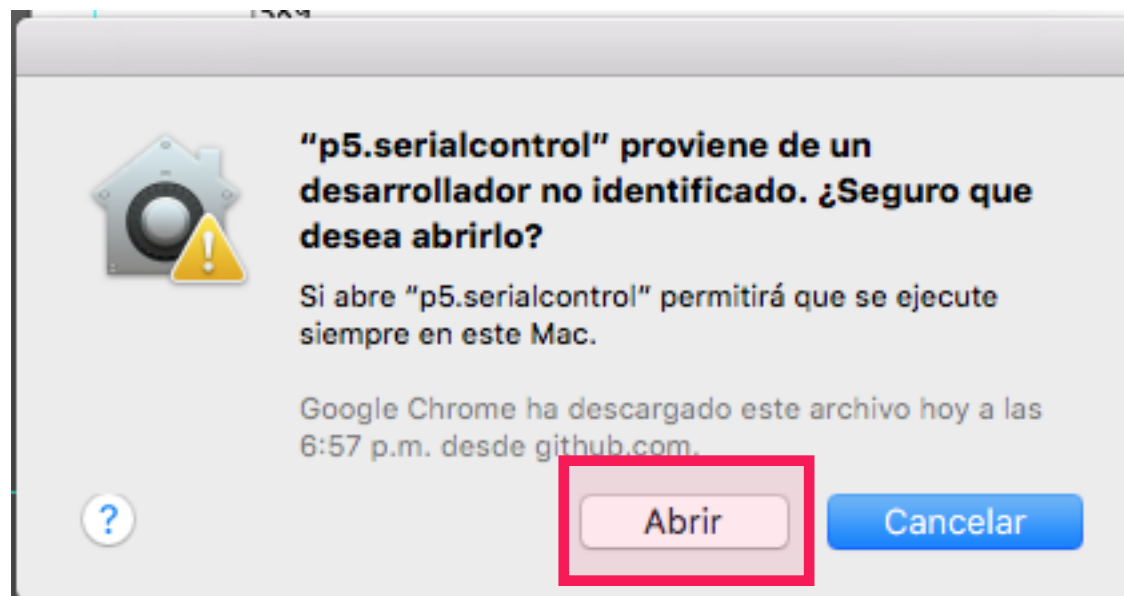
### *a.* Es posible recibir una alerta y no poder abrir la aplicación



**b.** En este caso hacer clic secundario sobre el archivo y elegir abrir



**c.** En la alerta que aparece hacer clic de nuevo en abrir



### 3. Abrir la aplicación y asegurarse de que el puerto seleccionado sea el correcto

#### Available Ports

/dev/cu.Bluetooth-Incoming-Port  
/dev/cu.usbmodem1411

List Ports

Select Serial Port:

/dev/cu.usbmodem1411

Close

Por lo general Arduino utiliza el puerto cu.usbmodem1411\*

Asegurarse de que el puerto esté abierto y en caso contrario abrirlo.



En Windows los nombres de los puertos suelen ser **COM3, COM4, COM5**. Una buena forma de saber cual es el puerto del Arduino es desconectarlo del equipo, mirar los puertos que aparecen y volver a conectarlo. De esta manera el puerto que cambia es el de la tarjeta Arduino.

## 4.

The screenshot shows the 'Serial Console' window. At the top, there is a 'List Ports' button and a 'Select Serial Port:' dropdown menu showing '/dev/cu.usbmodem1411'. Below this is a 'Close' button. The main section is titled 'Serial Console:' and contains an 'Enabled:' checkbox which is checked. To the right of the checkbox, a red vertical line points to the text 'Habilita el monitor'. Below the checkbox, a list of sensor values is displayed: 14, 14, 14, 14, 14, 14, 14, 15, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14. A red vertical line points to the value '15' with the text 'Se deben imprimir los valores del sensor'. At the bottom left, there is a 'Clear' button.

## 5. Por último, deshabilitar el monitor y dejar la aplicación corriendo



Para este punto se deben tener dos programas corriendo:

- El **sketch de arduino** que se subió a la tarjeta, lee los datos del sensor y los envía por el puerto serial. No es necesario que la aplicación de arduino esté abierta.
- La aplicación **p5.serialcontrol** que puede leer datos del puerto serial y enviarlos a un navegador

Es necesario utilizar la librería  
p5.serialport para poder acceder a los  
datos del puerto serial desde p5.js

C.

*Instalar librería en p5.js*



1. Ir a [github.com/vanevery/p5.serialport/releases/tag/v0.0.21](https://github.com/vanevery/p5.serialport/releases/tag/v0.0.21) y descargar la librería **p5.serialport**

v0.0.21  
9cd43dd

**v0.0.21**

vanevery tagged this on 14 Jul 2016 · 20 commits to master since this tag

Update version number

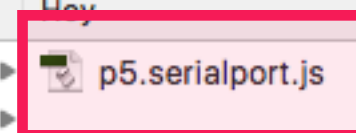
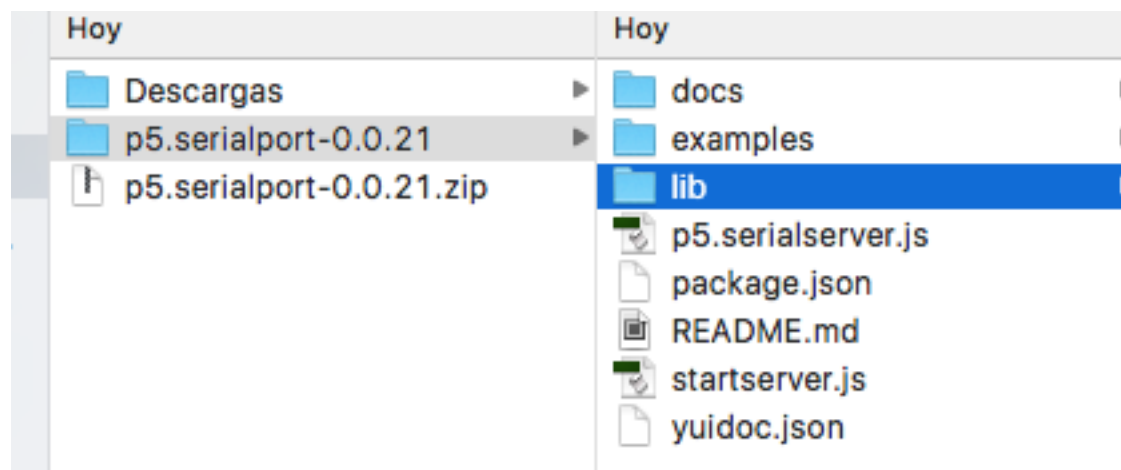
## Downloads

Source code (zip)

Source code (tar.gz)

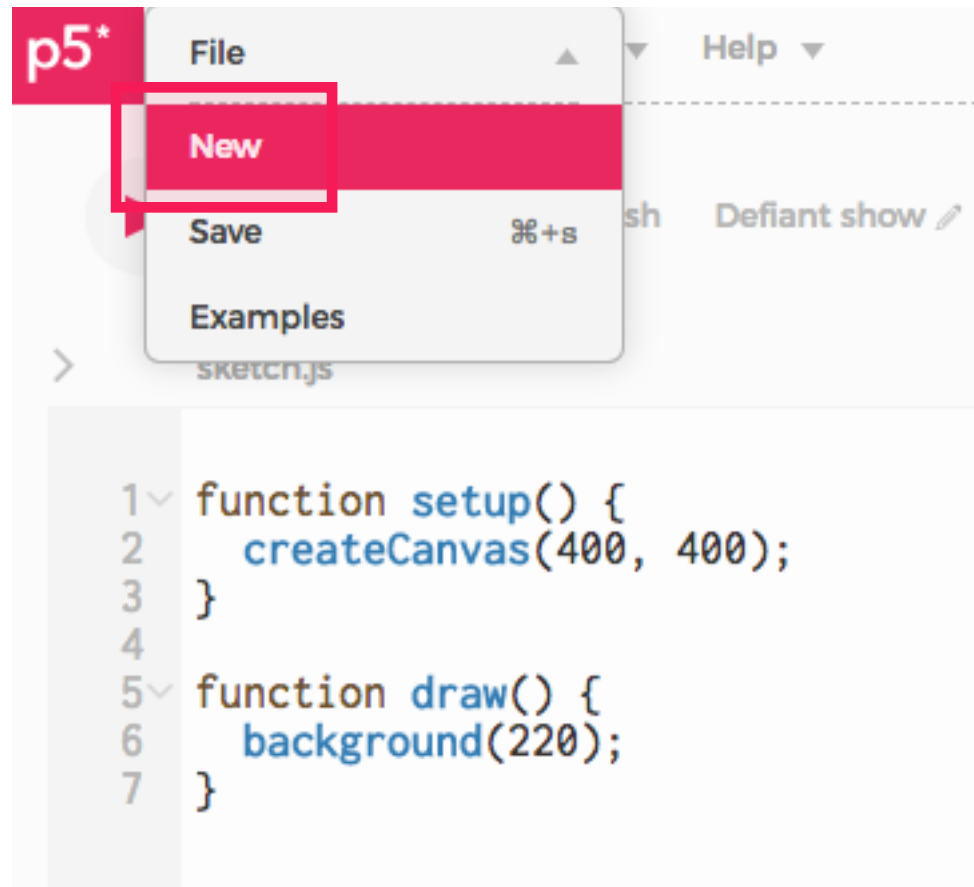
Descargar archivo .zip

2. Descomprimir carpeta en el equipo



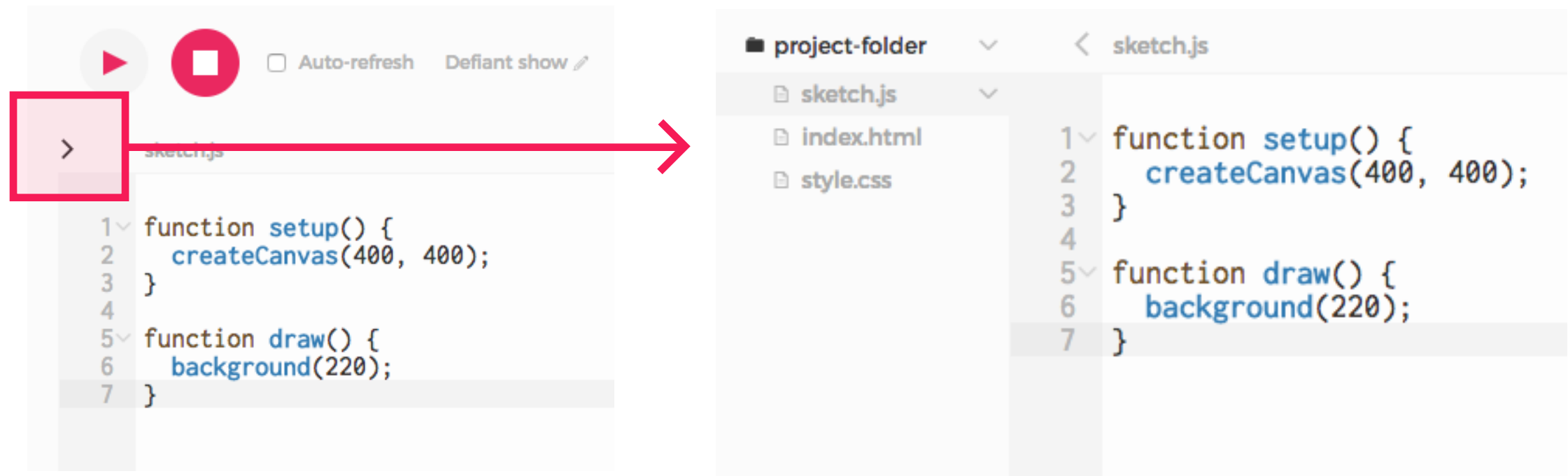
Este es el archivo  
que se va a utilizar

### 3. Crear un nuevo sketch en el editor web de p5.js [alpha.editor.p5js.org](https://alpha.editor.p5js.org)



**4.** Agregar el archivo descargado: **p5.serialport.js** al sketch

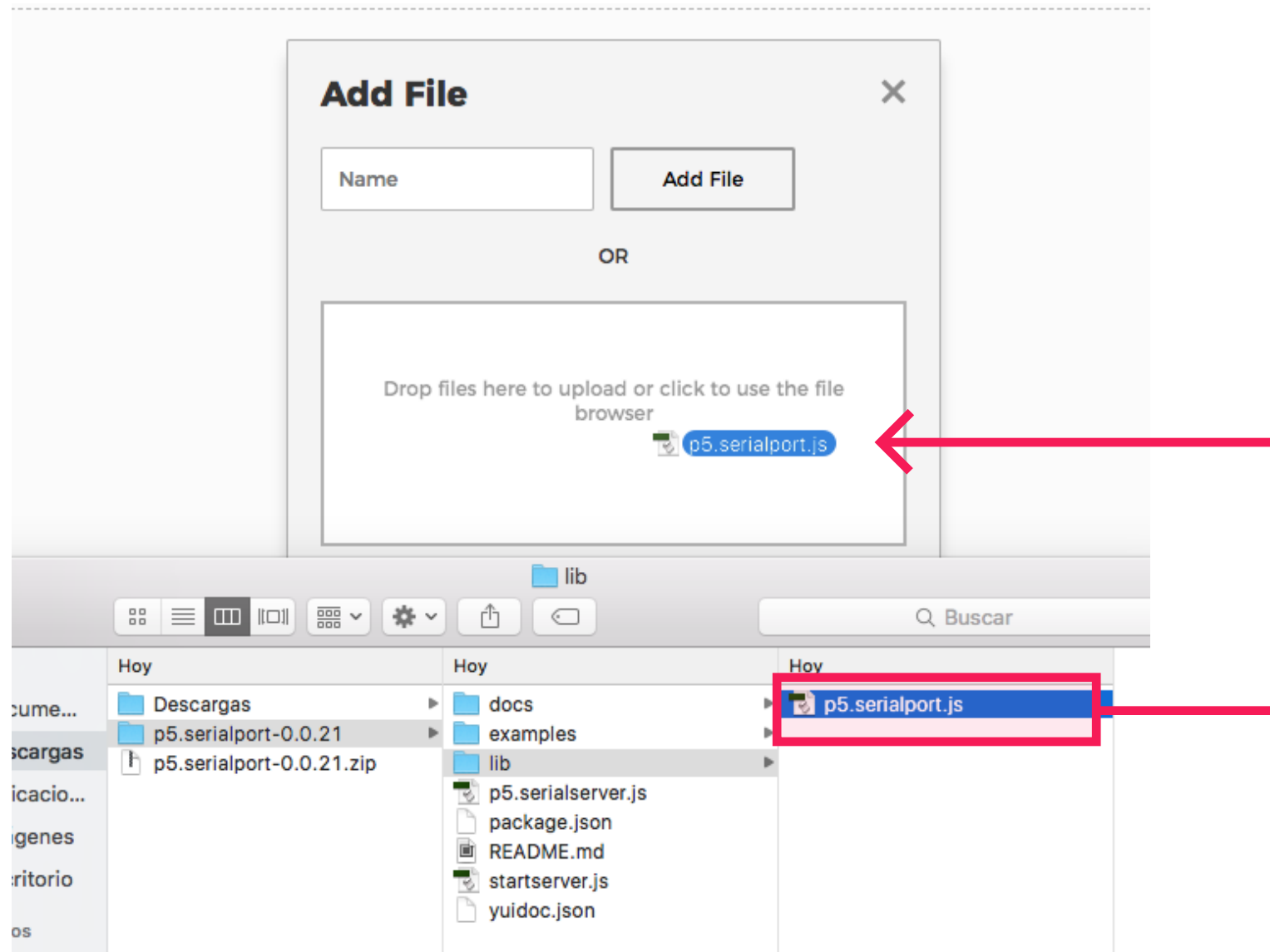
**a.** Desplegar el panel de archivos del sketch



**b.** Hacer clic en **project-folder** y luego en **Add file**

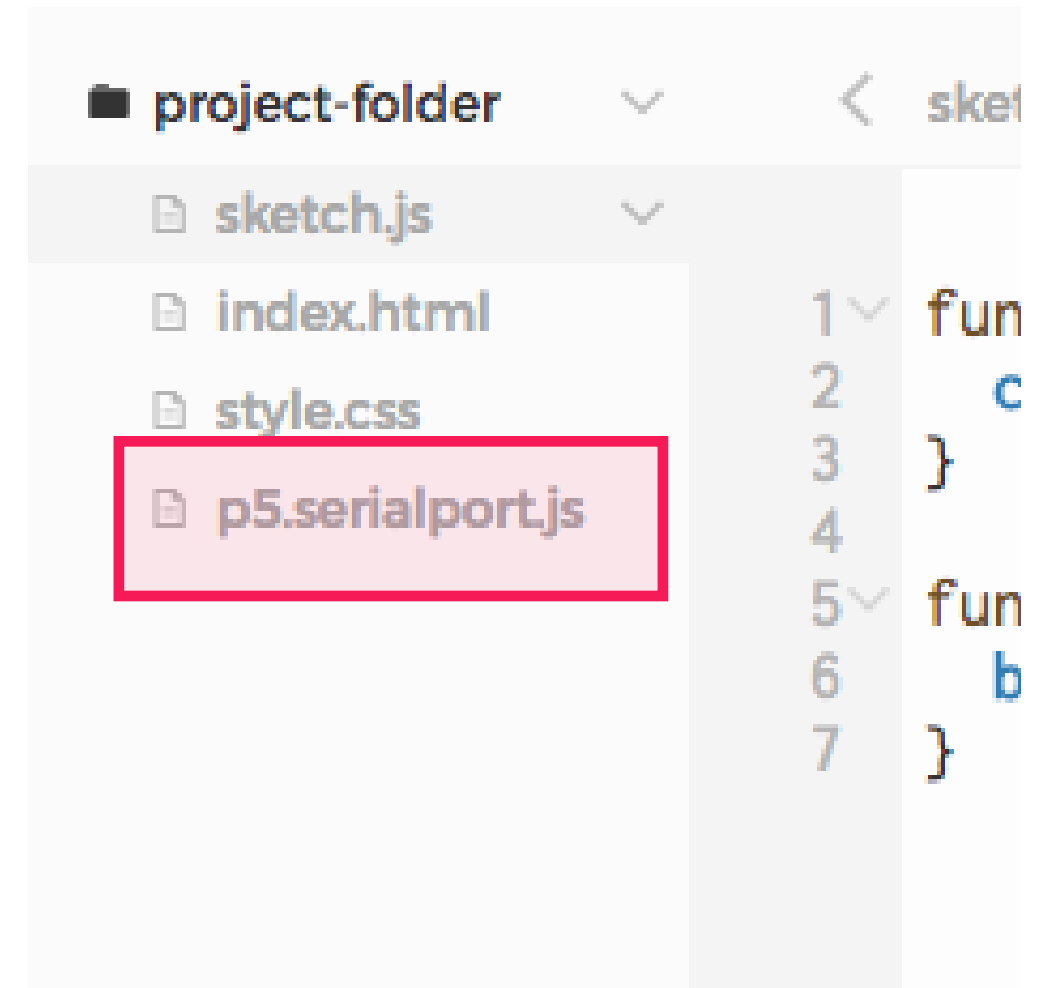
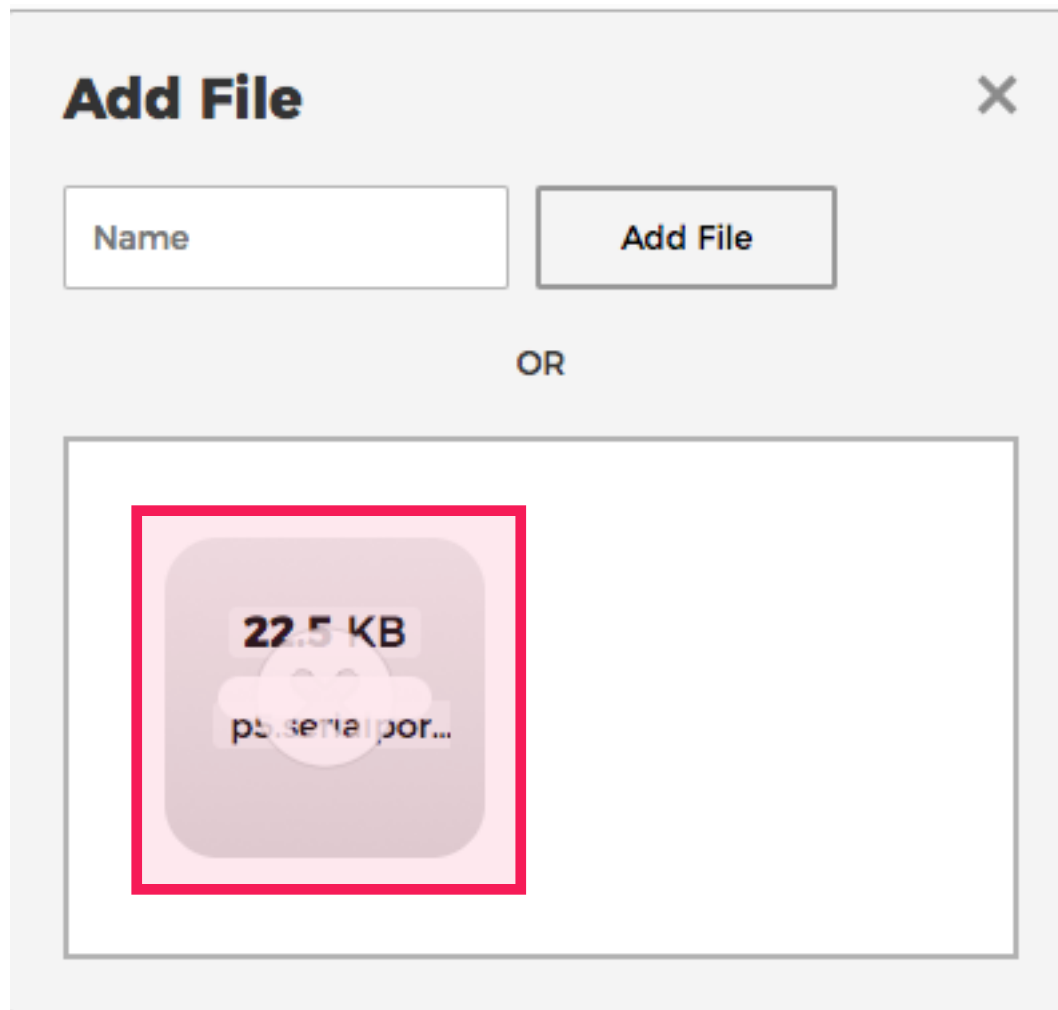


**C.** Aparece una ventana a la cual se debe arrastrar el archivo p5.serialport.js\*

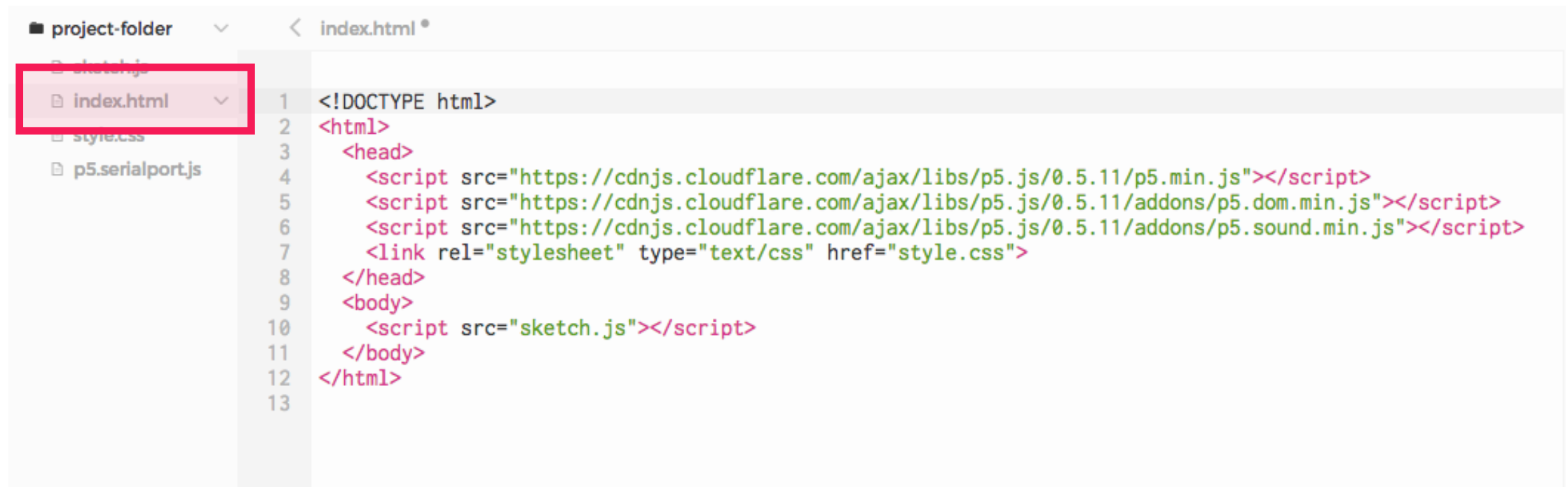


\*Para poder agregar el archivo en el editor web es necesario estar logueado en la página

*d.* El archivo debe aparecer en el panel de archivos del sketch



## 5. Abrir el archivo **index.html**



## 6. Hacer un enlace a la librería dentro del archivo



# Código index.html

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.11/p5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.11/addons/p5.dom.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.11/addons/p5.sound.min.js"></script>

    <script language="javascript" type="text/javascript" src="p5.serialport.js"></script>

    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <script src="sketch.js"></script>
  </body>
</html>
```

*d.*

*Leer y visualizar datos en p5.js*



# 1. Imprimir la lista de **puertos seriales** en p5.js

```
var serial; // Variable que guarda una instancia de la libreria p5.serialport

function setup() {
  serial = new p5.SerialPort(); // crea una nueva instancia de la libreria p5.serialport
  serial.on('list', printList); // cuando el puerto se abre llama a la función printList
}

//función que imprime una lista con los puertos seriales disponibles
function printList(portList) {
  // portList es una lista con los puertos
  for (var i = 0; i < portList.length; i++) {
    //imprime los puertos en la consola
    console.log(i + " " + portList[i]);
  }
}
```

Al correr este código se imprime en la consola del editor la lista de puertos disponibles.

```
Console
ws://localhost:8081
opened socket
0 /dev/cu.Bluetooth-Incoming-Port
1 /dev/cu.usbmodem1411
```

Este es el puerto que está utilizando el arduino

Este código se puede ver y descargar en: [laurajunco.github.io/medios/serial\\_read/sketch.js](https://laurajunco.github.io/medios/serial_read/sketch.js)

## 2. Agregar **eventos seriales**

```
var serial; // Variable que guarda una instancia de la libreria p5.serialport
var portName = '/dev/cu.usbmodem1421'; //variable con el nombre del puerto

function setup() {
  serial = new p5.SerialPort(); // crea una nueva instancia de la libreria p5.serialport

  serial.on('list', printList);           //llama a la función printList
  serial.on('connected', serverConnected); // llama a la función printList
  serial.on('open', portOpen);           // llama a la función portOpen
  serial.on('data', serialEvent);        // llama a la función serialEvent
  serial.on('error', serialError);       // llama a la función serialError
  serial.on('close', portClose);         // llama a la función portClose

  serial.open(portName);                 // open a serial port
}
```

Se están creando una serie de llamados que correrán unas funciones específicas cuando suceda alguno de los siguientes eventos:

**list** – El programa pide una lista de puertos

**connected** – cuando se conecto a la aplicación p5.serialcontrol

**open** – un puerto serial es abierto

**close** – un puerto serial es cerrado

**data** – se reciben datos de un puerto serial

**error** – algo sale mal

### 3. Crear las **funciones** que responden a cada **evento serial**

```
function serverConnected() {  
  println('conectado al servidor');  
}  
  
function portOpen() {  
  println('el puerto serial fue abierto')  
}  
  
function serialEvent() {  
  
}  
  
function serialError(err) {  
  
  println('algo salió mal' + err);  
}  
  
function portClose() {  
  
  println('el puerto serial se cerró');  
}
```

En esta función se reciben los datos del puerto serial

## 4. Crear una **variable global** para guardar los datos recibidos

```
var serial; // Variable que guarda una instancia de la libreria p5.serialport
var portName = '/dev/cu.usbmodem1421'; //variable con el nombre del puerto
var data = 0; //datos que llegan del puerto serial
```

## 5. Modificar la función **serialEvent()** para mostrar los datos en consola.

```
function serialEvent() { //llamada cuando se reciben datos al puerto serial
  data = serial.read(); //lee los datos
  console.log(data); //los imprime
}
```

Ahora que se tienen los datos recibidos del sensor en una variable de p5.js se pueden utilizar para dibujar en la función draw().

## 6. Pintar una elipse cuyo tamaño responda a los valores del sensor

```
function draw(){  
  background(0);  
  fill(255);  
  //dibuja una elipse con radio dictado por los datos recibidos  
  ellipse(width/2, height/2, data, data);  
}
```

