

# clase-08

---

martes 03 mayo 2022, presencial

## repaso clase anterior y programa hoy (15 min)

la clase pasada aprendimos:

- sonido
- diseño de sintetizadores
- protocolo MIDI

hoy aprenderemos:

- comunicación serial con apps y Processing
- comunicación serial con navegador web y p5.js
- instrucciones proyecto mitad de semestre próxima semana

## comunicación serial con apps y Processing (60 mins)

gracias a la comunicación serial, podemos crear aplicaciones en Processing que son capaces de recibir información desde nuestro Arduino, y al revés también.

abramos el ejemplo [ej\\_00\\_arduino\\_envia\\_numeros](#) y subamos el código al microcontrolador Arduino.

este ejemplo envía por puerto serial una línea, con dos valores separados por coma. el primer valor es aleatorio y el segundo es un contador, que parte en 0 y aumenta en 1 por cada iteración del bucle `loop()` de Arduino.

podemos abrir el monitor serial del software Arduino IDE y ver los valores para confirmar que está funcionando.

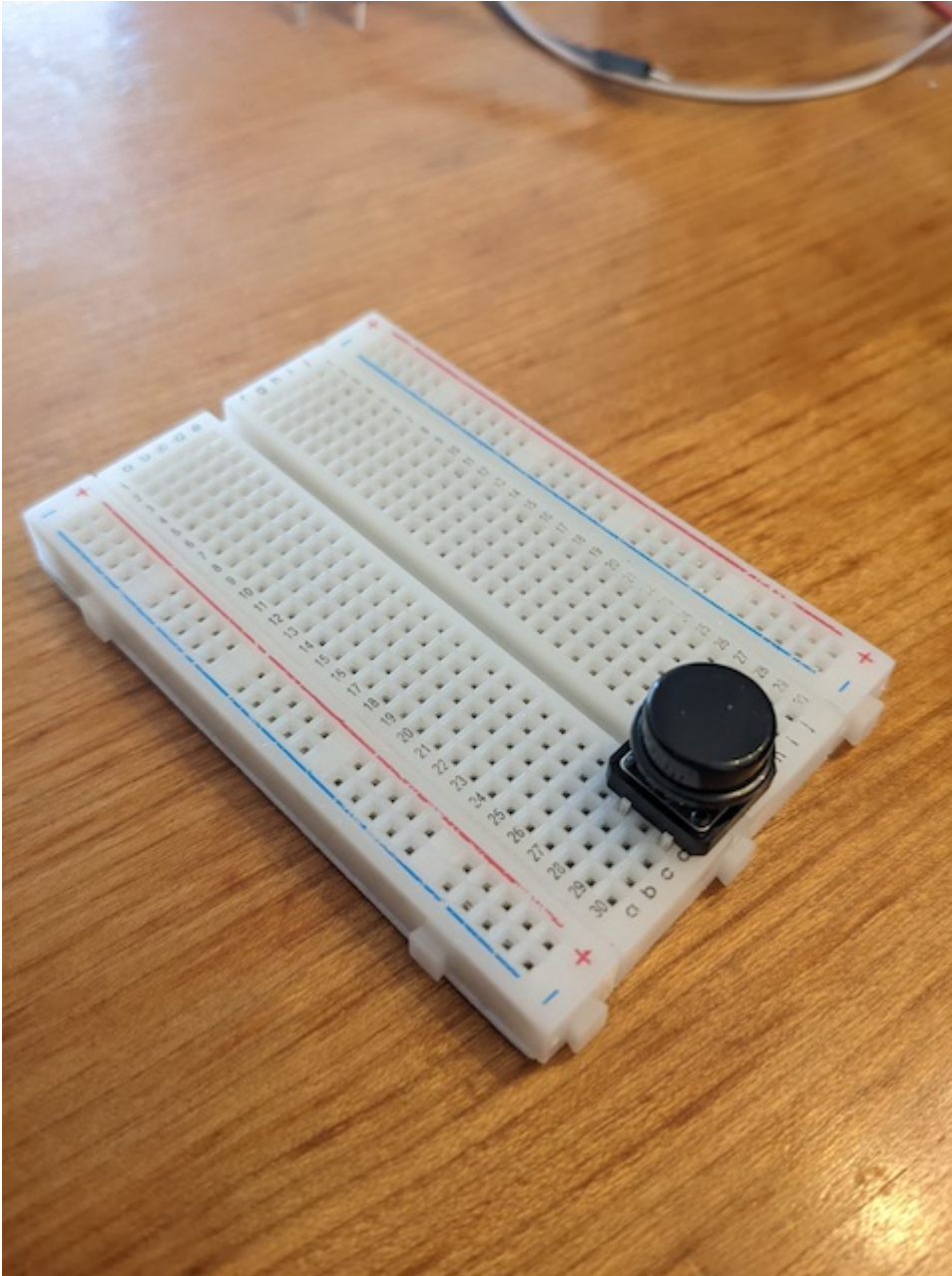
cerramos el monitor serial y luego abrimos el ejemplo de Processing [ej\\_00\\_processing\\_recibe\\_numeros](#) que recibe estos números, porque una comunicación serial solamente puede ir desde un lugar a otro, entonces si no cerramos el monitor serial de Arduino, el puerto serial estará ocupado y Processing no podrá acceder a esos valores.

el ejemplo en Processing dibuja una elipse usando los valores recibidos desde Arduino. para lograr hacer la conexión serial entre nuestro hardware microcontrolador Arduino y el software Processing, debemos correr el software en Processing y mirar la consola de Processing e identificar el número de puerto serial ocupado por Arduino, y actualizar ese valor en el código de Processing.

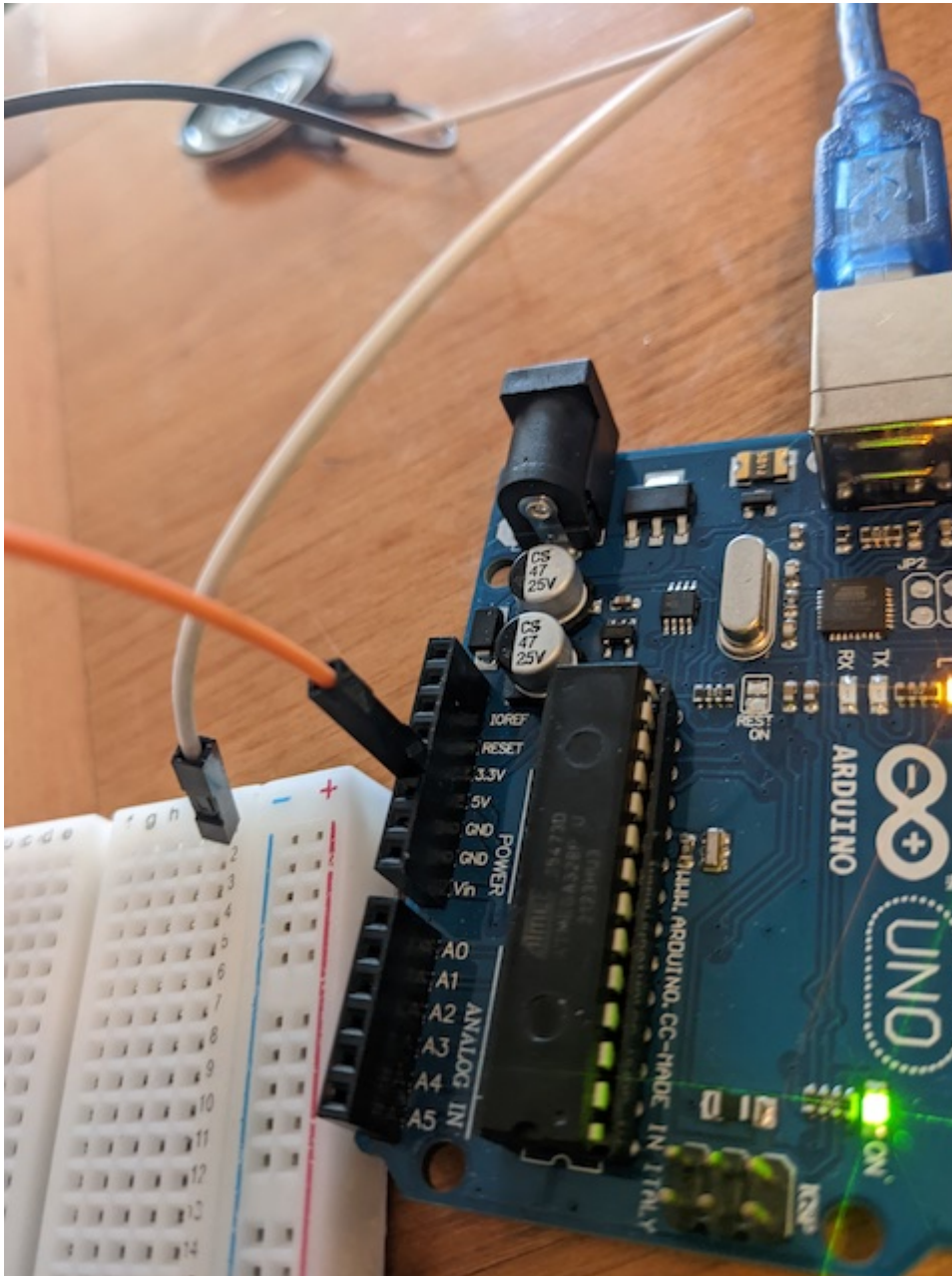
notar que esta configuración puede ser un problema, porque esto hace que nuestro software en Processing no sea "plug and play" o "conectar y listo", debemos configurar esto en cada computador o Arduino que usamos, para asegurar que la conexión serial funcione.

en el siguiente par de ejemplos [ej\\_01\\_arduino\\_envia\\_pulsador\\_potenciometro](#) y [ej\\_01\\_processing\\_recibe\\_pulsador\\_potenciometro](#), usaremos el mismo circuito con pulsador y potenciómetro de la clase anterior, acá está copiado a continuación.

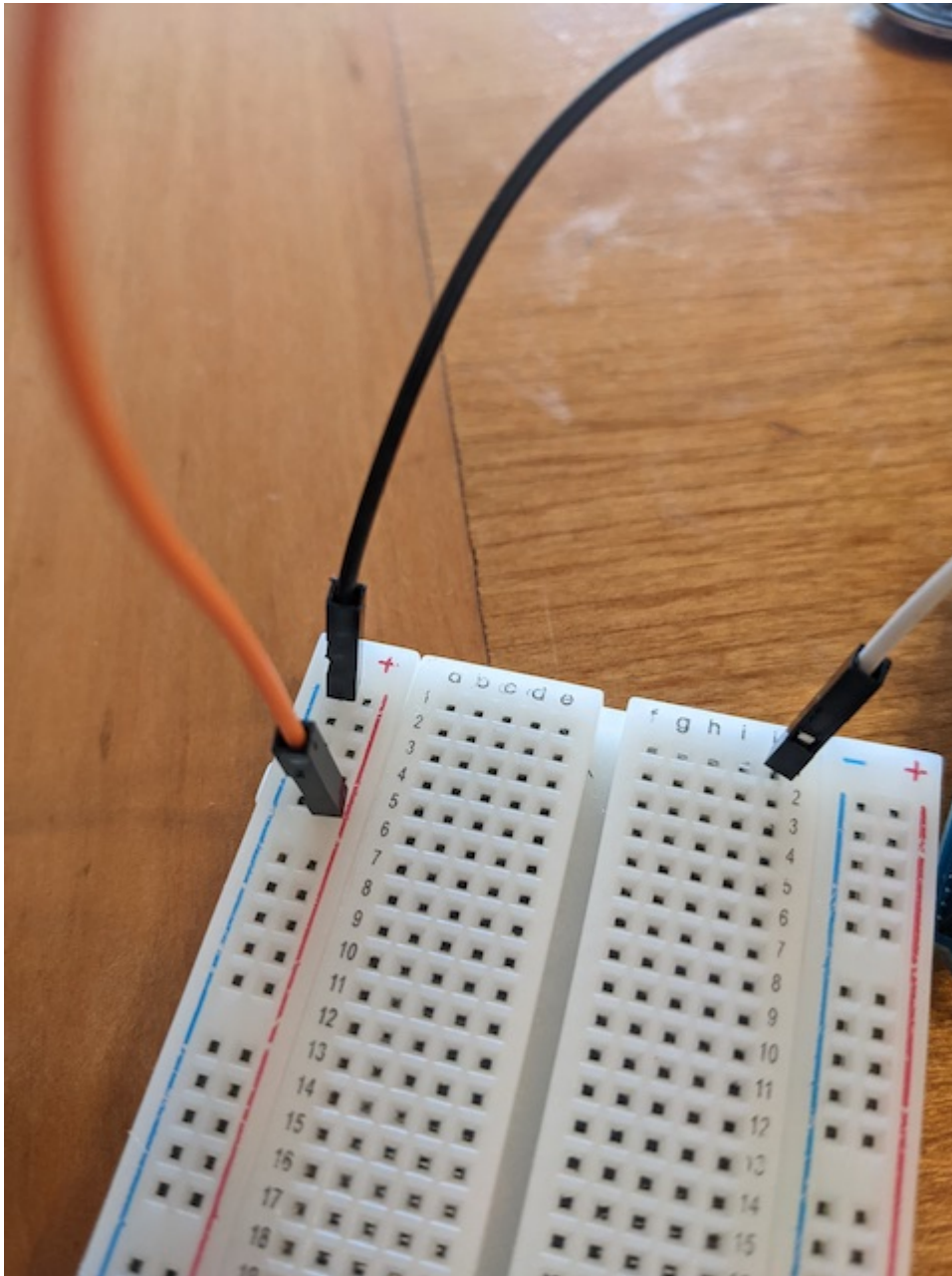
partamos de nuestra protoboard anterior, con un botón pulsador en el extremo inferior.



usemos un cable para acceder a los 5v desde nuestro Arduino.

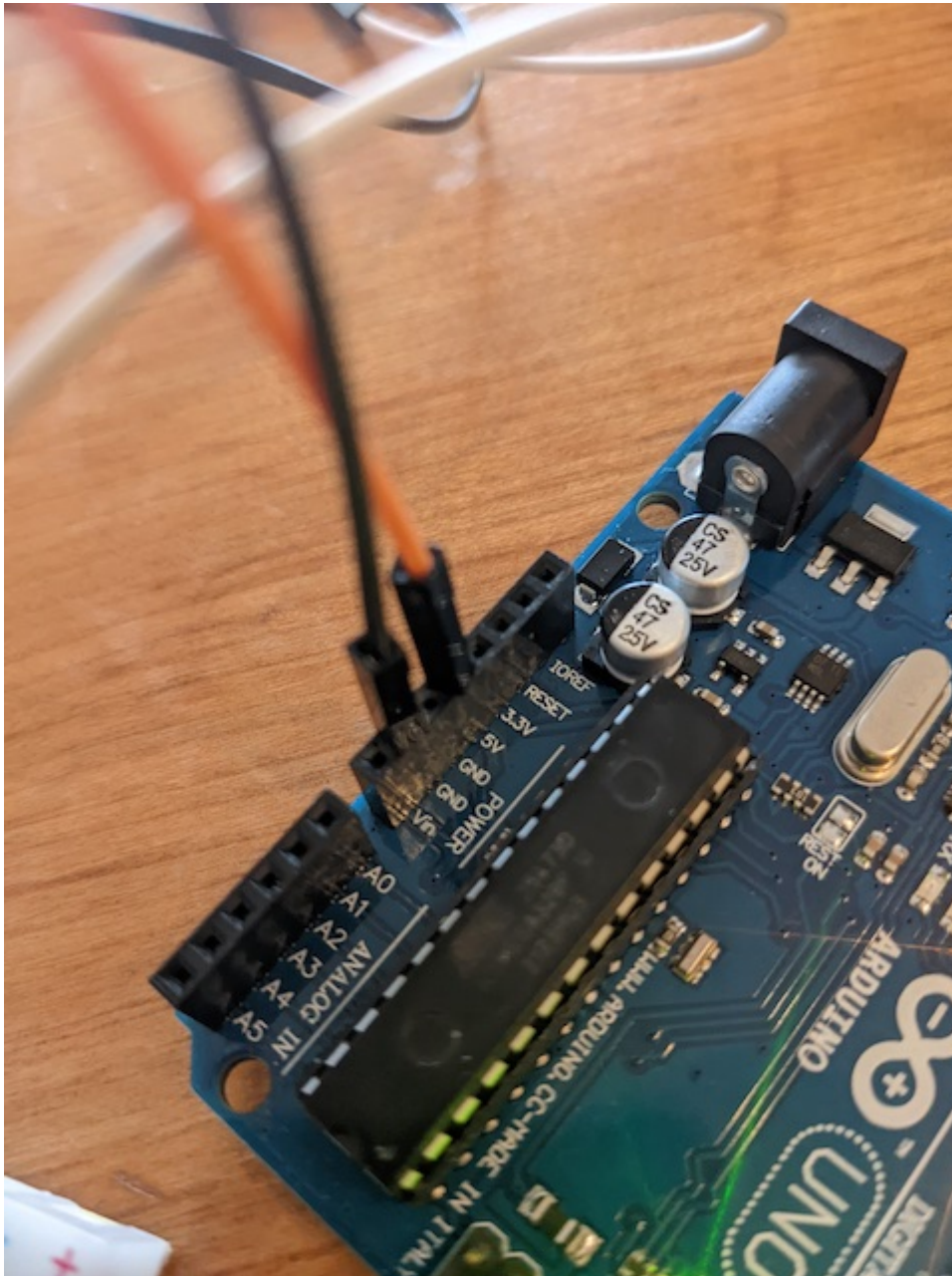


el otro extremo del cable se conecta al riel de alimentación de la protoboard.

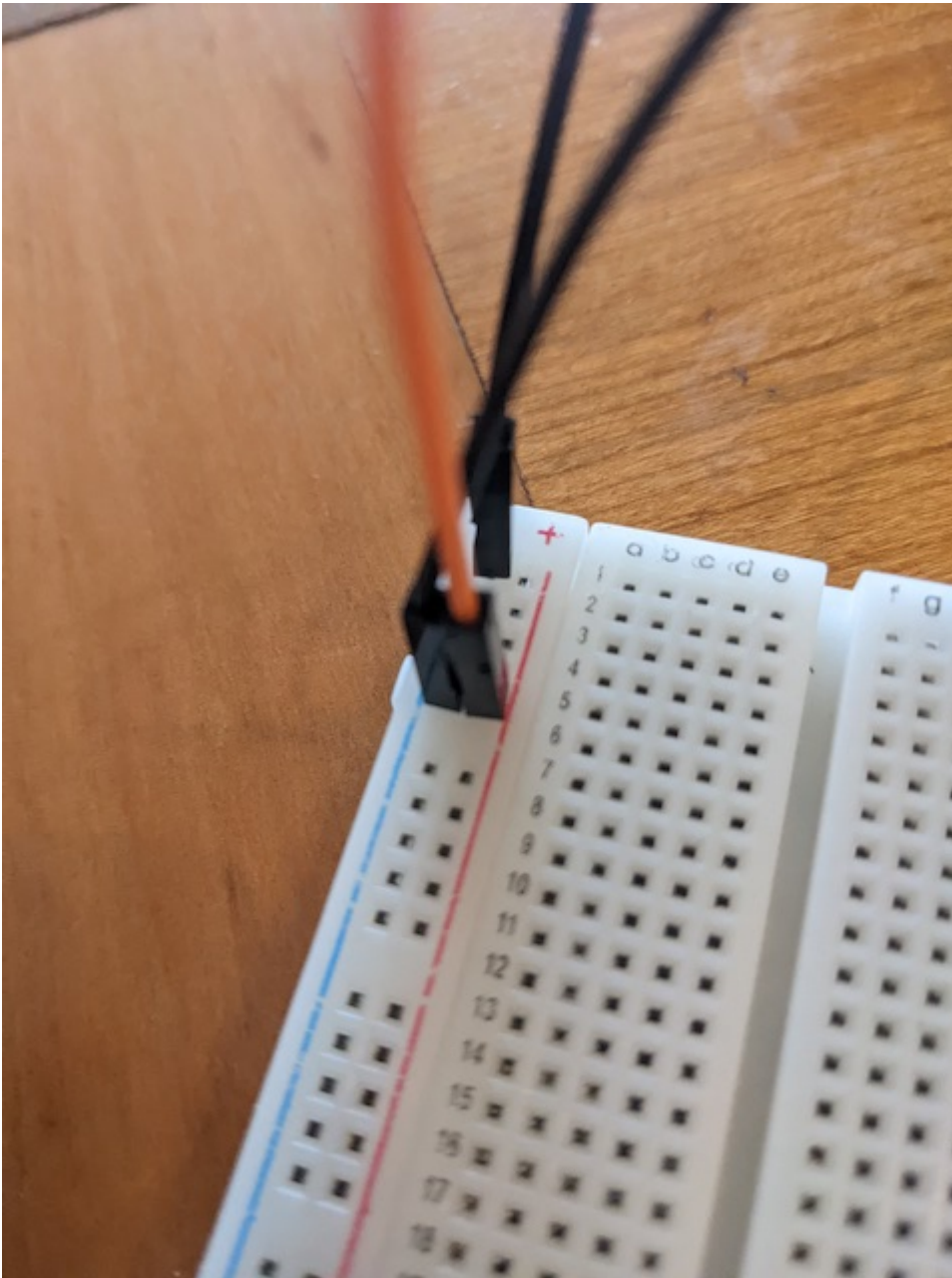


usamos otro cable para acceder a tierra en Arduino.

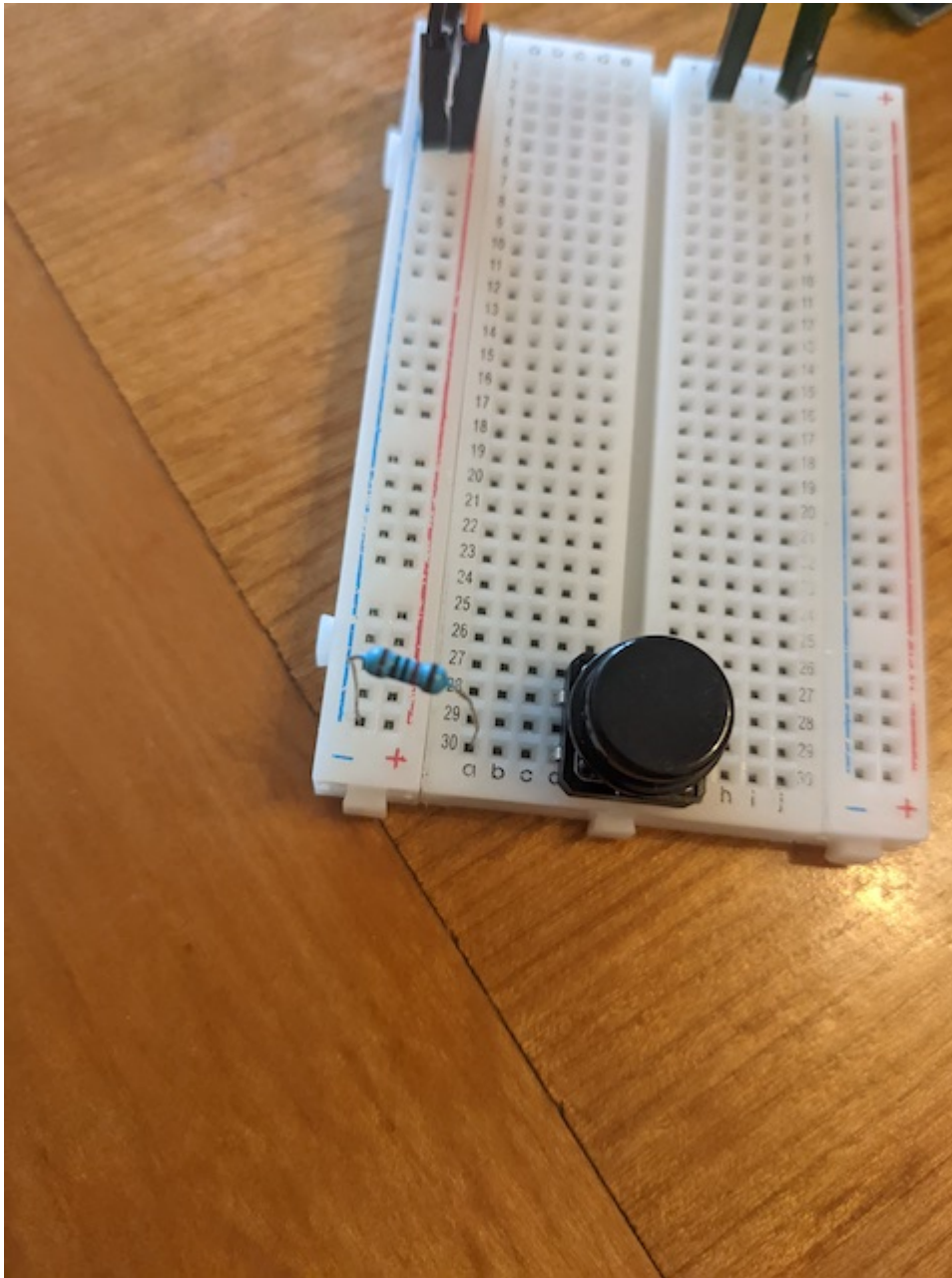




y el otro extremo lo usamos para conectar al riel de tierra de la protoboard.

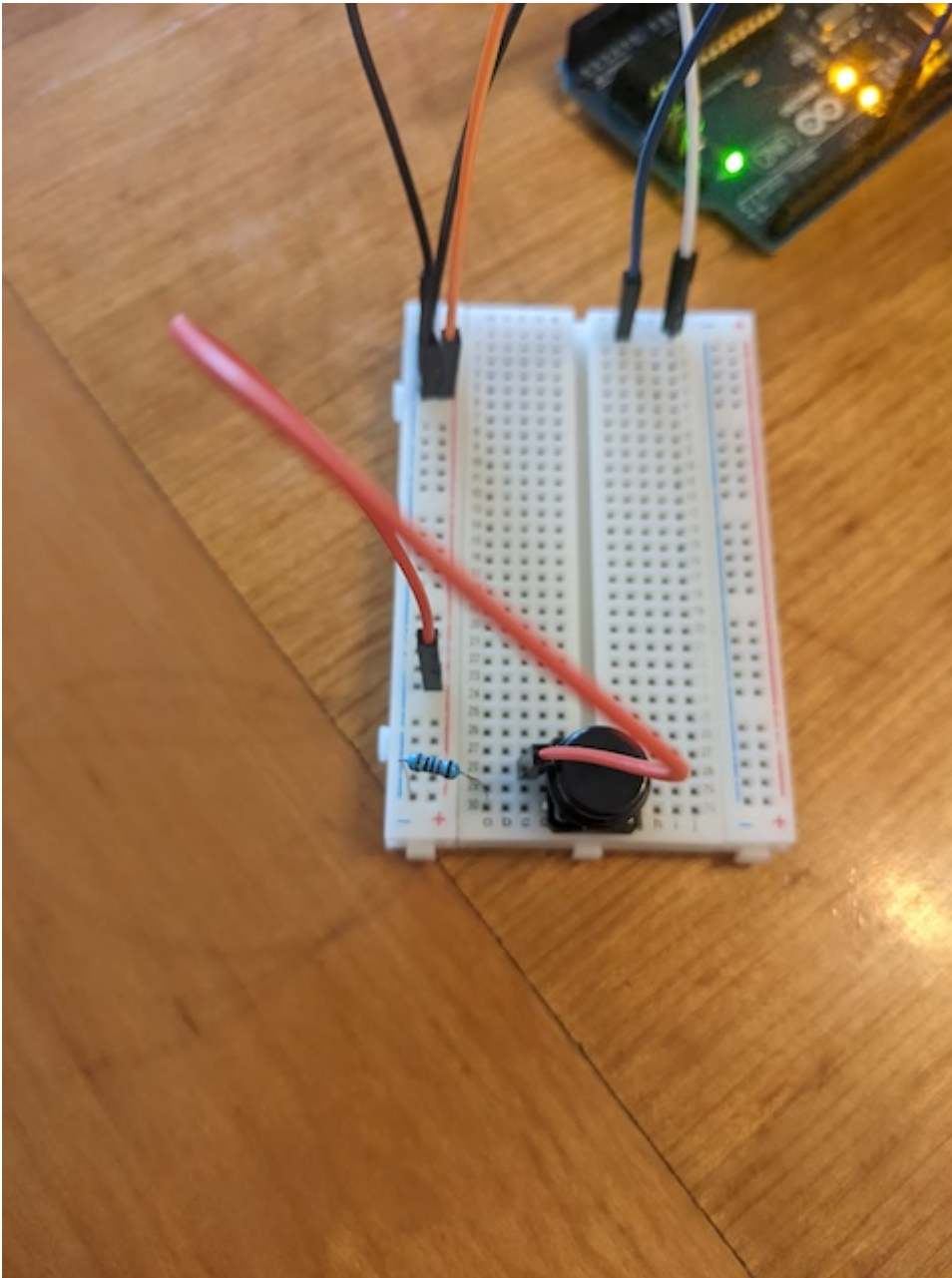


conectamos un resistor entre tierra y el terminal inferior del pulsador.



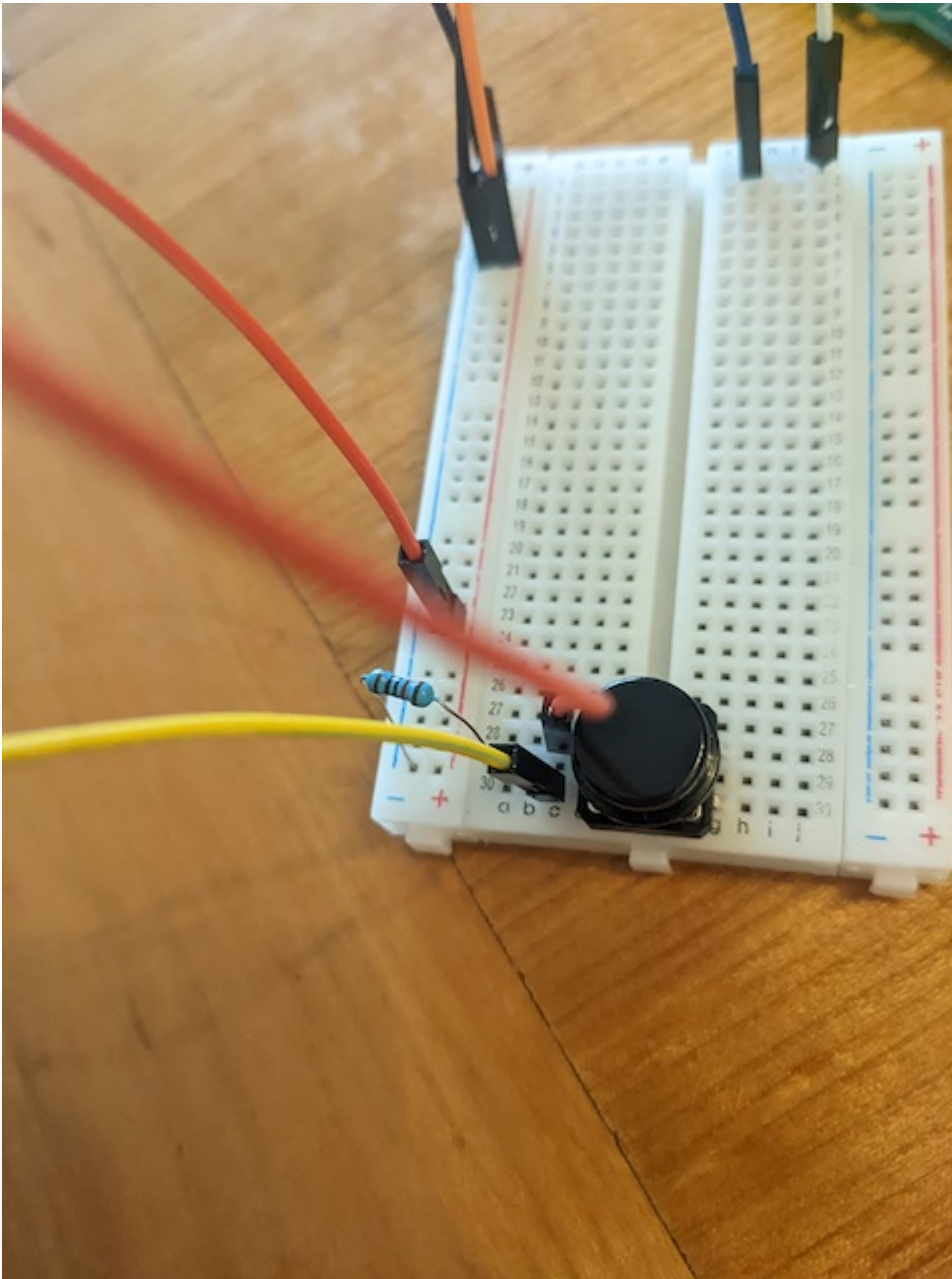
el terminal superior del pulsador lo conectamos al riel de alimentación 5V de la protoboard.



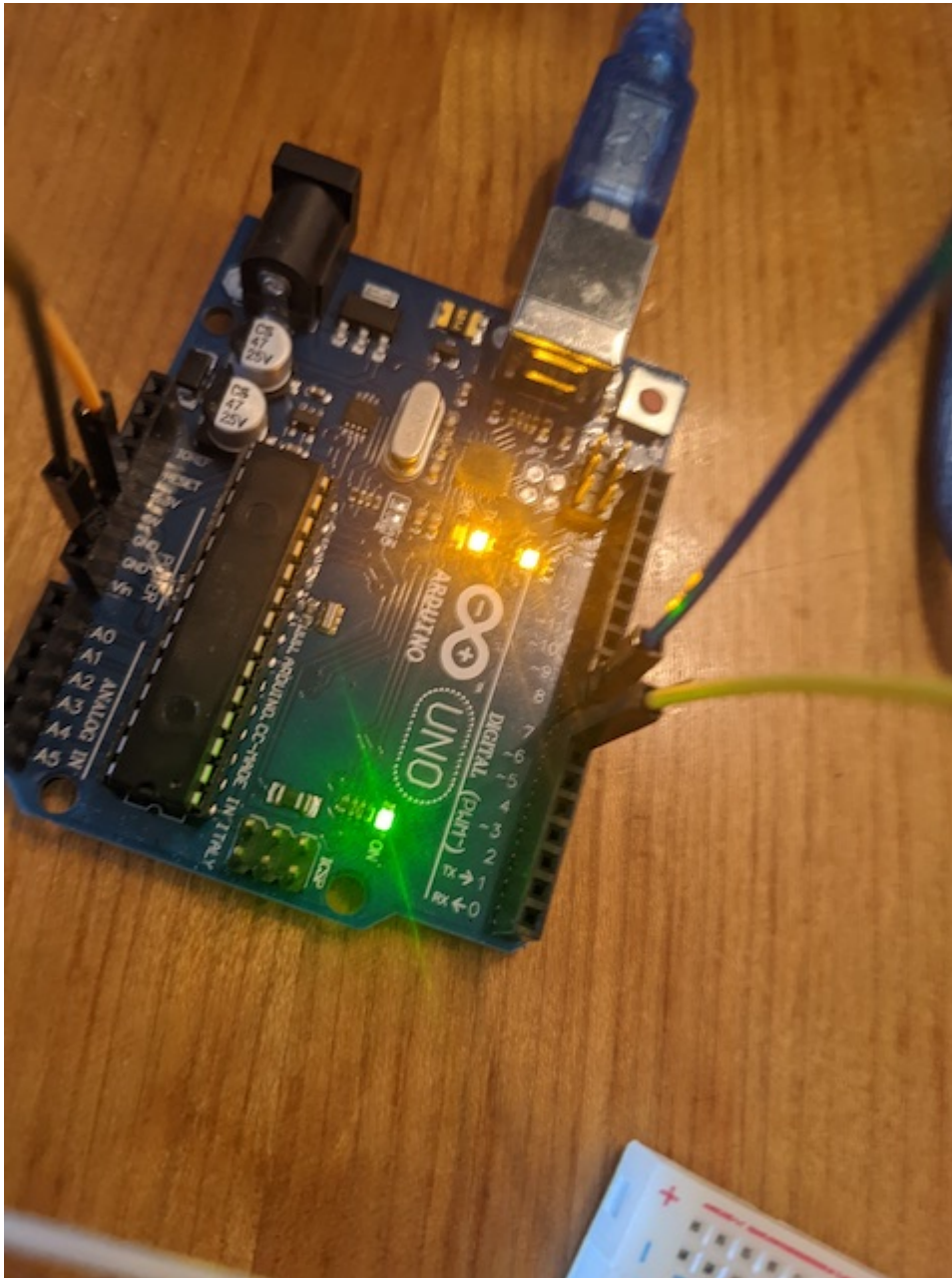


en el mismo terminal donde está el resistor, conectamos un cable para medir el voltaje.

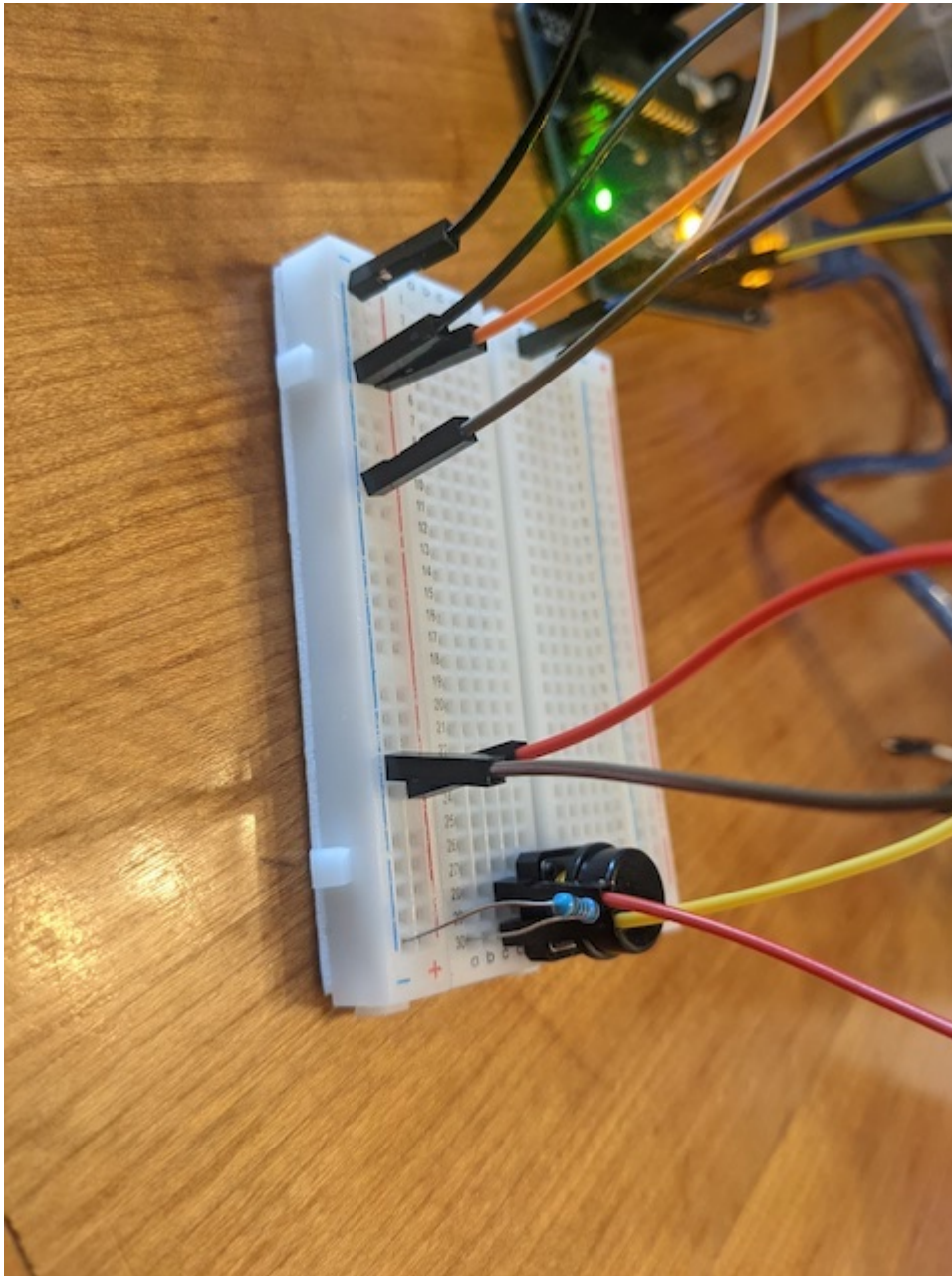




el otro extremo de ese cable lo conectamos al pin digital 7 de nuestro Arduino.

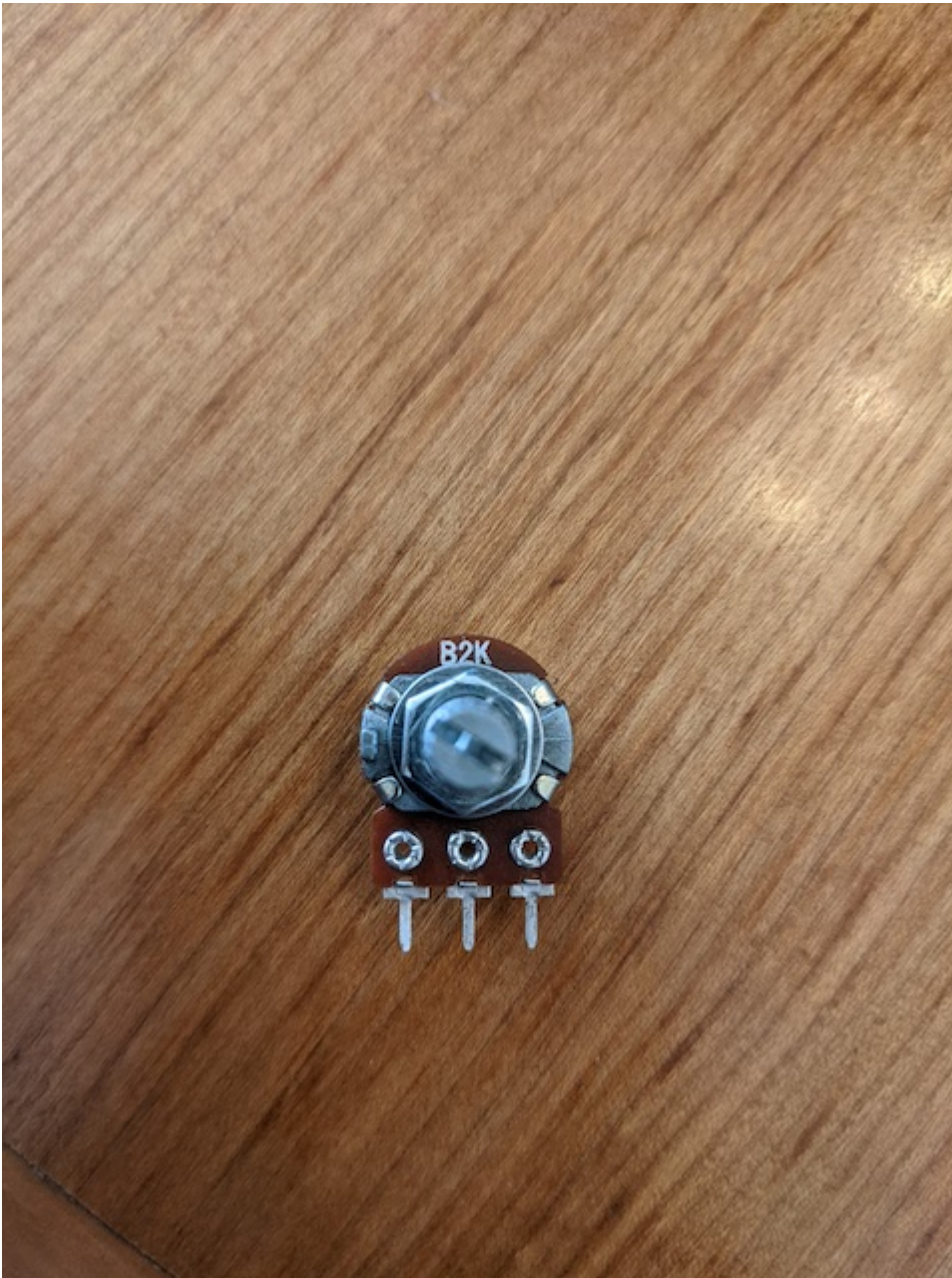


en mi protoboard noté que la tierra no se transmite bien a lo largo del riel, lo que hace que la lectura del pulsador sea ruidosa y no funcione bien, por lo que conecté un cable entre la parte superior del riel y la parte inferior, lo que solucionó el problema.



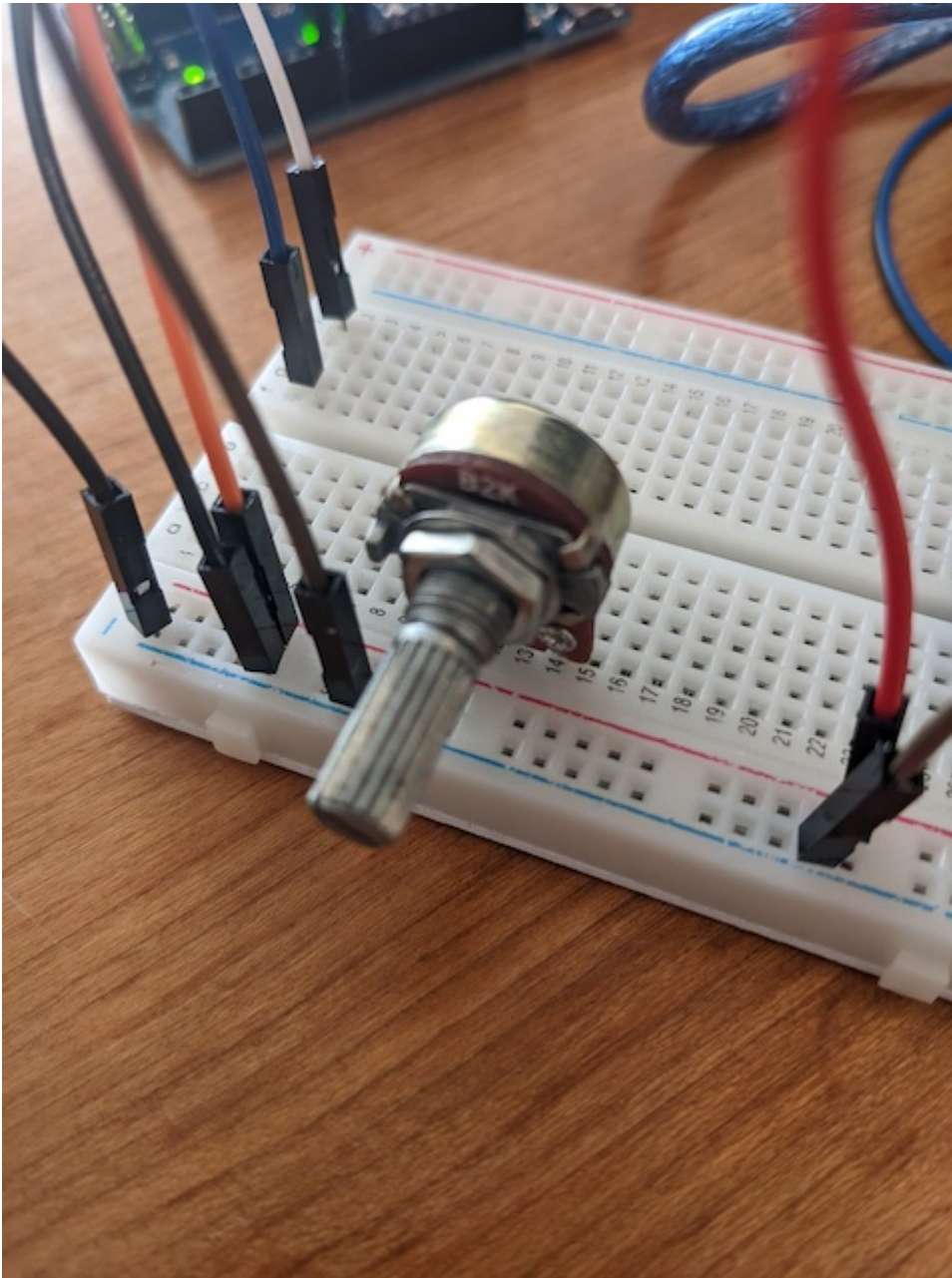
ahora conectaremos un potenciómetro de una sola vuelta.



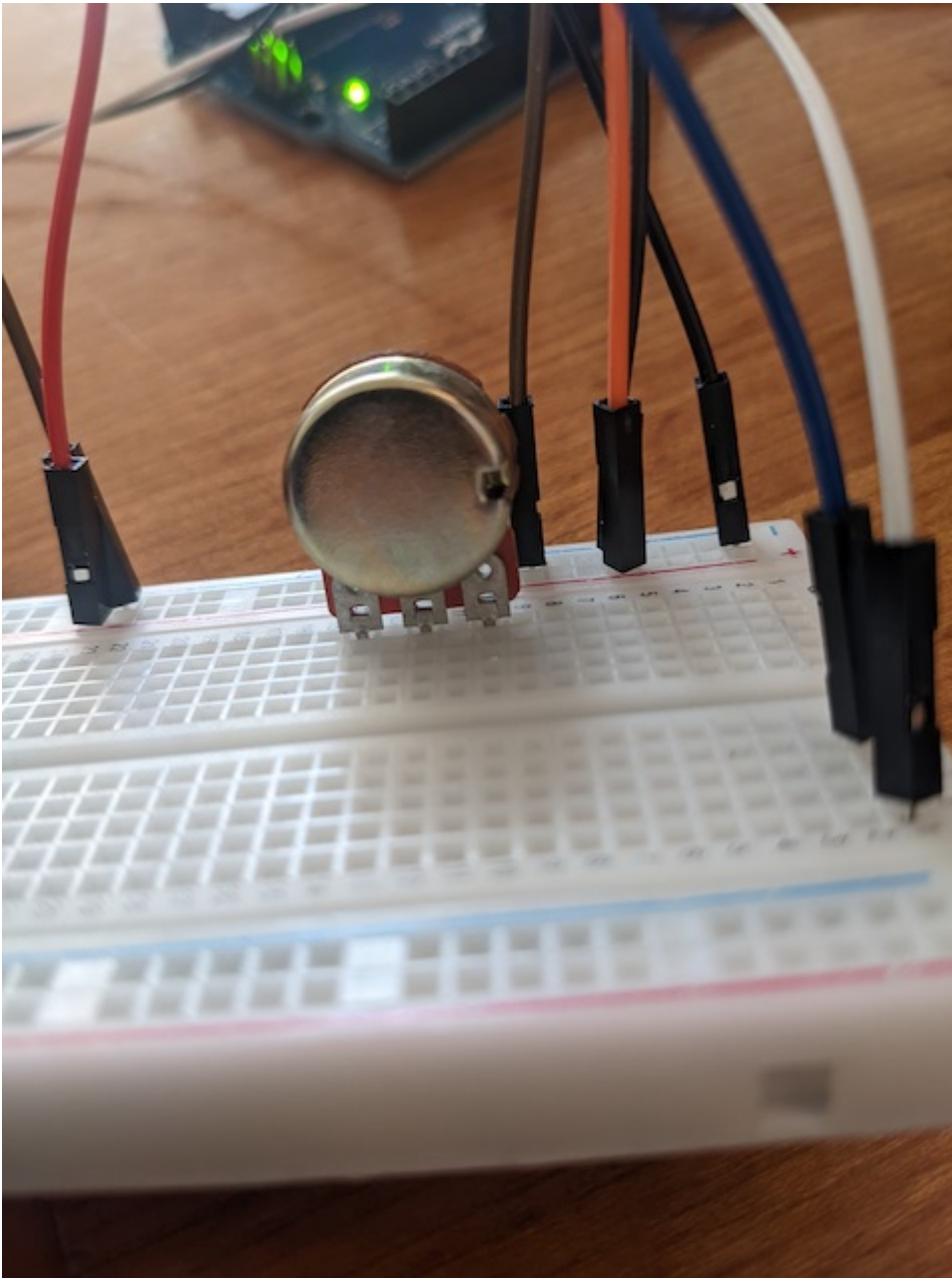


lo conectamos en la columna a de la protoboard, con el terminal izquierdo en la fila 10, el central en la 12, y el derecho en la 14.

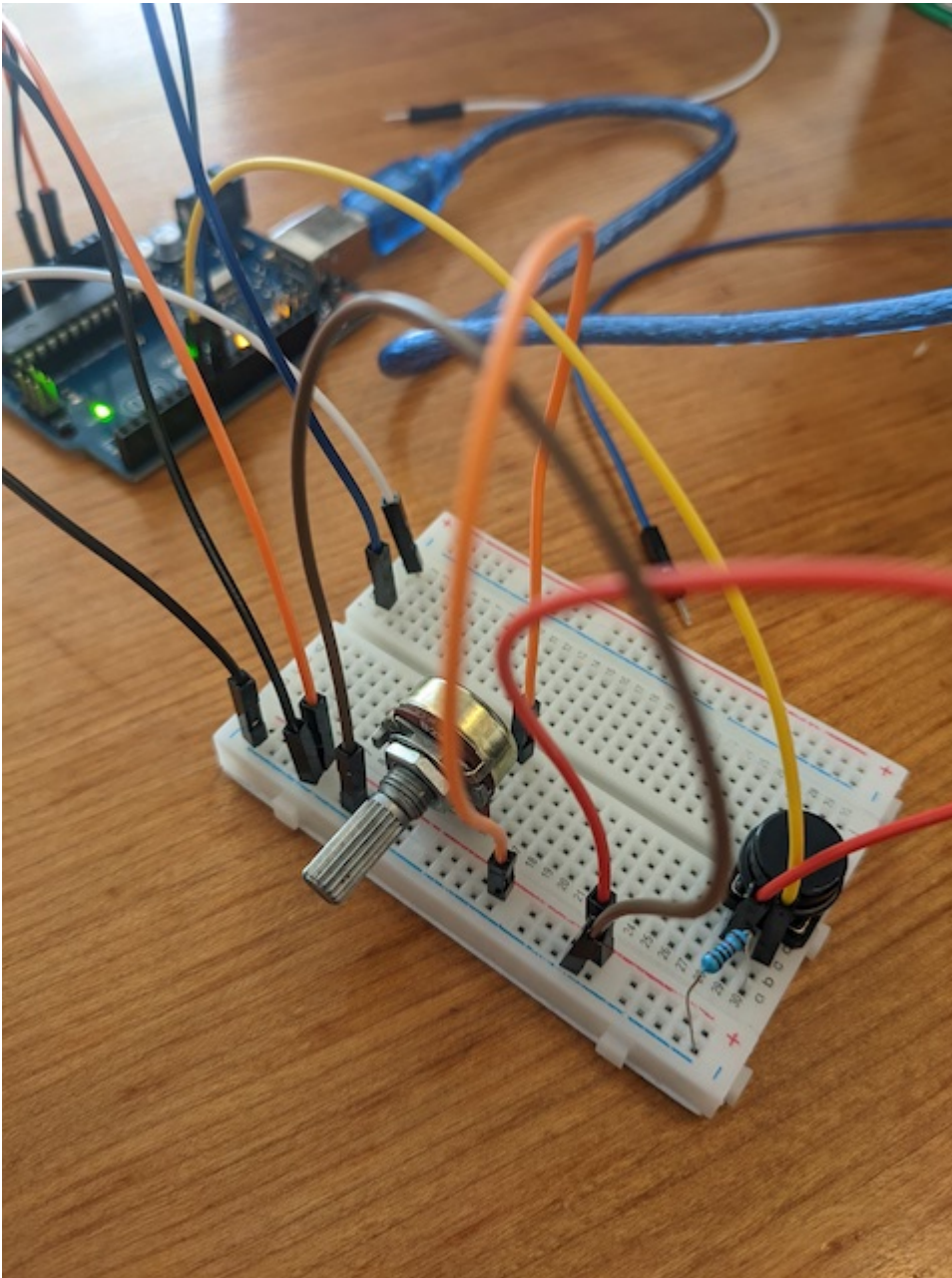




así se ve atrás donde realizaremos las conexiones.

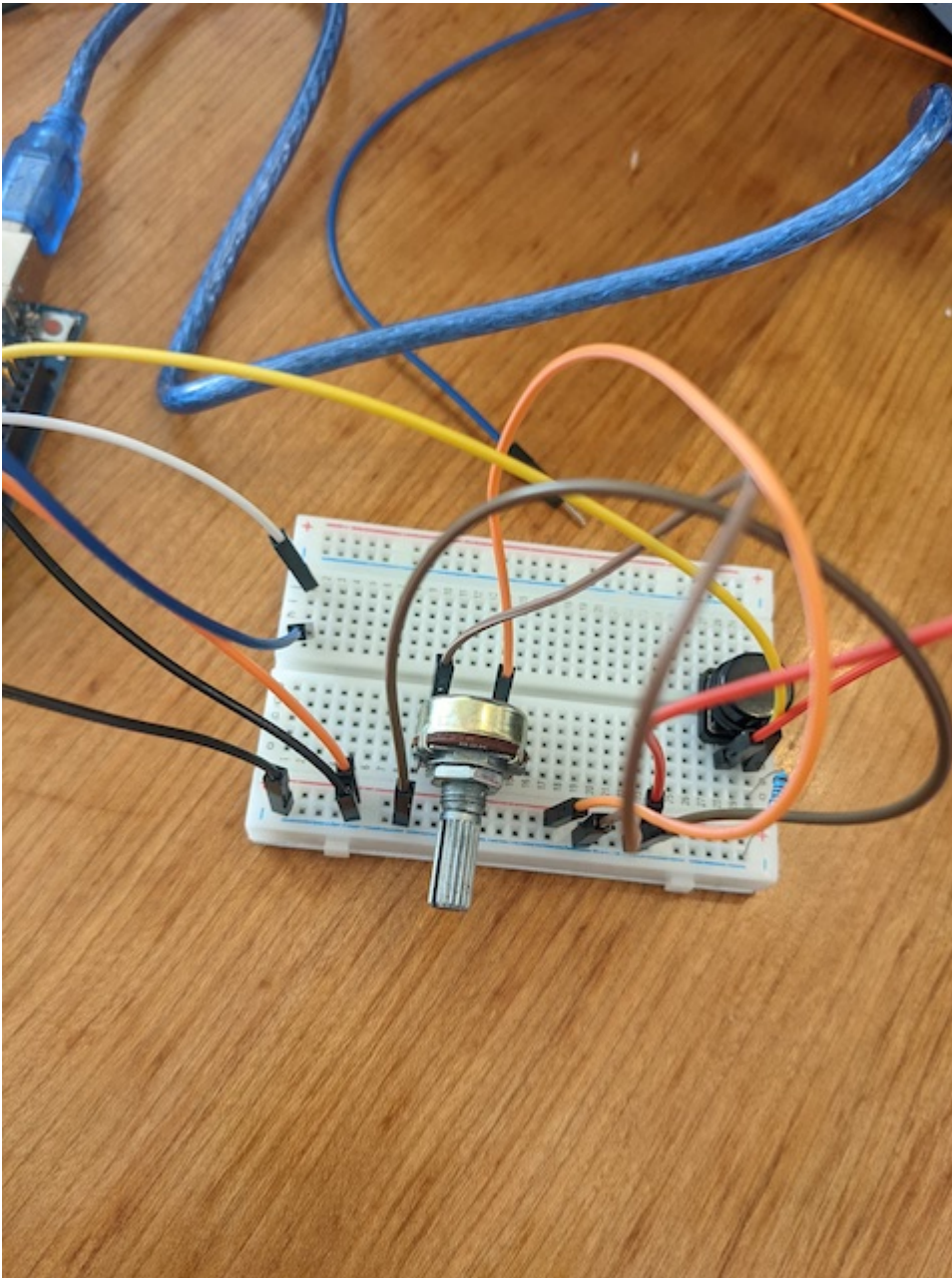


el terminal derecho en la fila 14 lo conectamos a 5V con un cable.



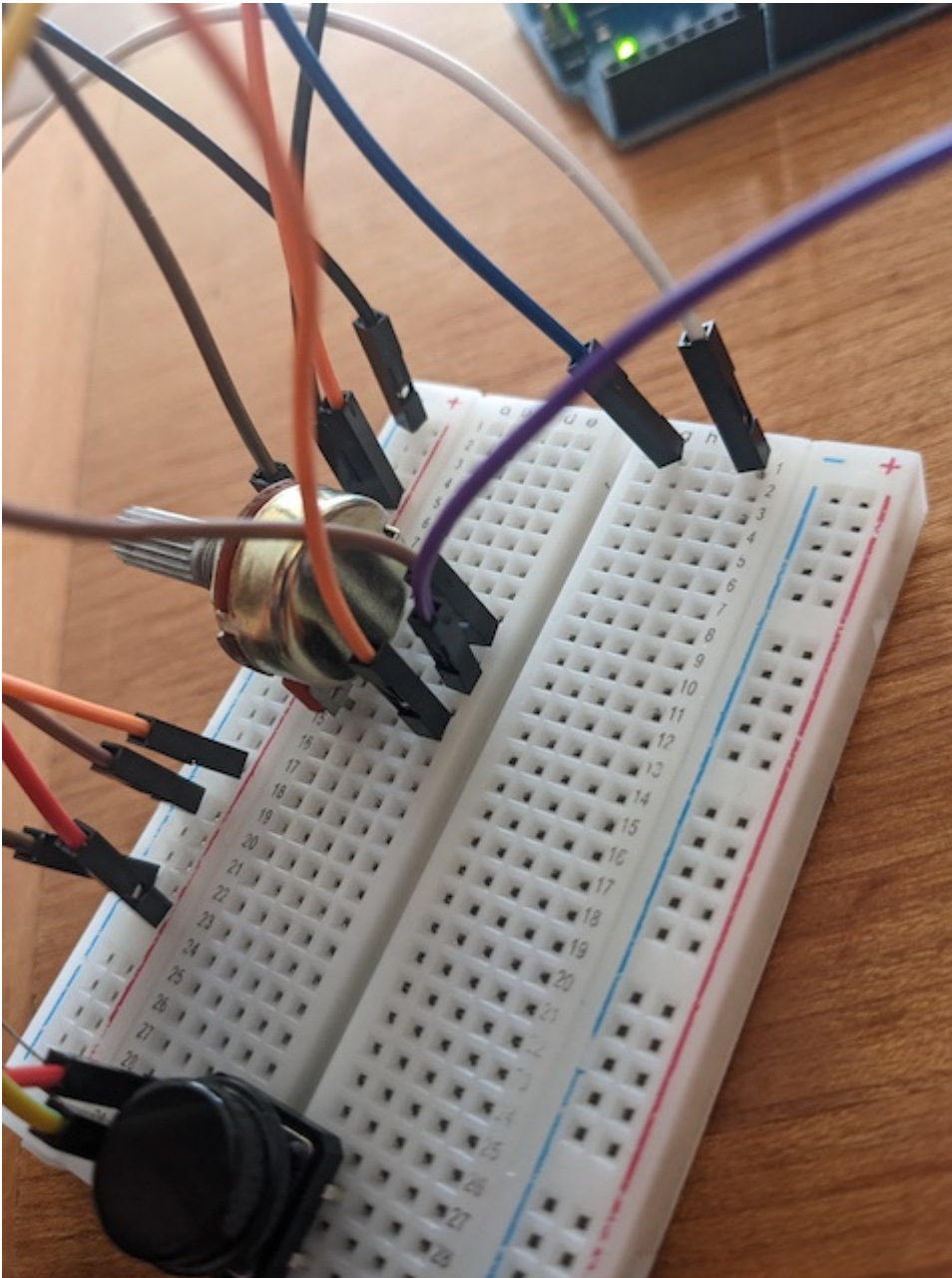
el terminal izquierdo en la fila 10 lo conectamos a tierra con un cable.



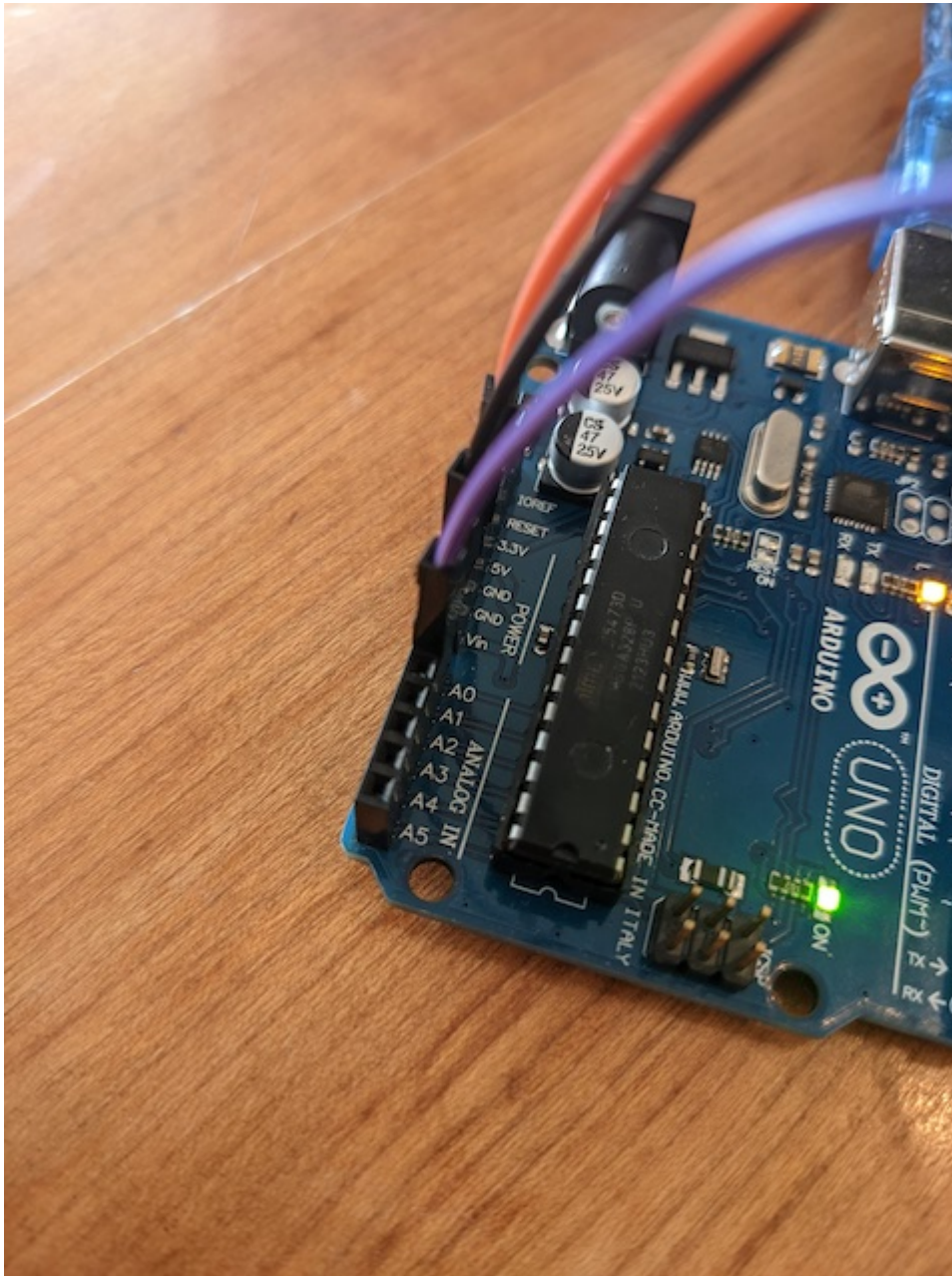


conectamos un cable al terminal central en la fila 12 para su lectura análoga.





el otro extremo del cable de señal lo conectamos al pin A0 de entrada analógica del Arduino.



## motivación de comunicación serial con Processing (15 min)

nuestro computador tiene distintas maneras de ingresar información, como ratón, teclado y cámara.

la gran gracia de nuestro microcontrolador Arduino es que tiene pines GPIO (general purpose input and output, en español pines de propósito general para entrada y salida).

un teclado de computador tiene muchas teclas que pueden detectar cuando son presionadas, y cada tecla envía una señal que distingue la tecla que fue presionada y en qué momento, lo que permite teclear sucesiones de caracteres en orden y rápidamente.

el ratón o trackpad permite mover el cursor en la pantalla en coordenadas X e Y. con los botones se pueden hacer acciones distintas, incluyendo click con botón izquierdo, central o derecho, doble click al pulsar el mismo botón dos veces consecutivas y rápido.

estos avances permitieron que los computadores pasaran de ser dispositivos controlados exclusivamente por comandos textuales, a lo que conocemos como **GUI** (graphical user interface, interfaz gráfica de usuario).

con nuestro microcontrolador Arduino, y el uso de puerto serial, podemos pensar en distintas interfaces humano - computador, que expanden o reemplazan los clásicos controladores ratón y teclado.

enlaces de interés:

- [Palo Alto Reserach Center \(PARC\)](#)
- Grupo de investigación [Tangible Media](#), liderado por profesor [Hiroshi Ishii](#) en MIT.

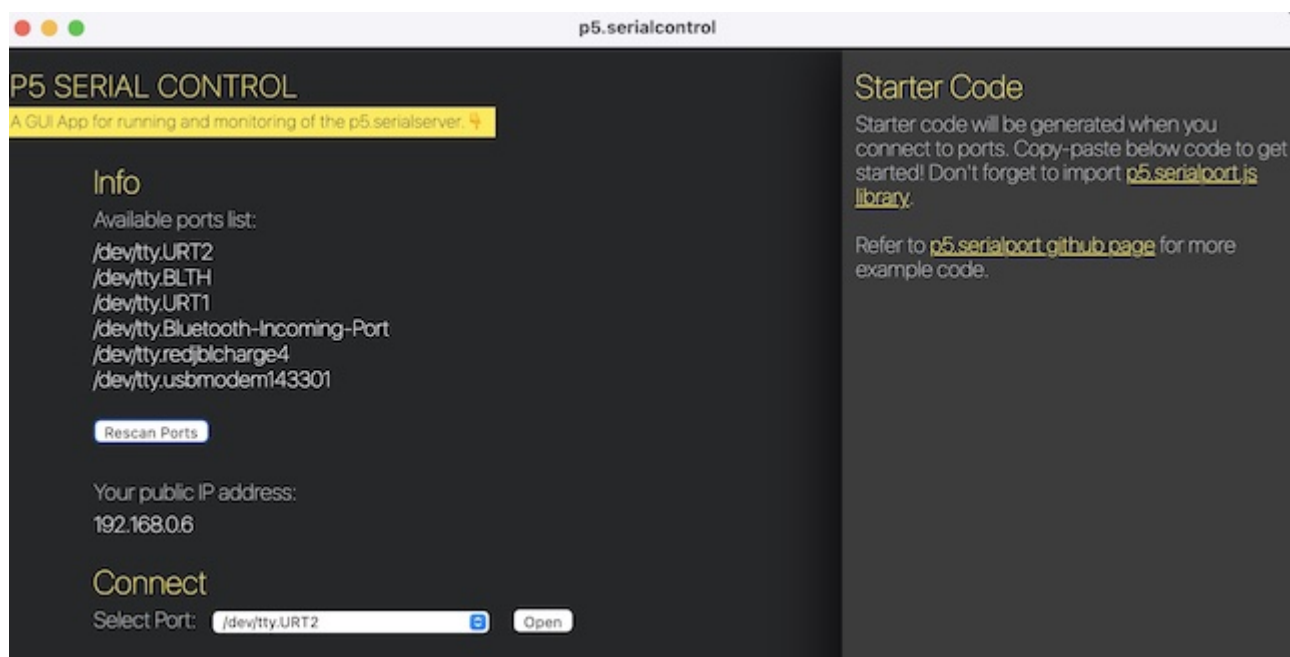
## comunicación serial con navegador web y p5.js (60 mins)

p5.js es una biblioteca de JavaScript y proyecto de la Processing Foundation, que nos permite crear webs interactivas.

p5.js tiene funciones y variables para detectar entrada con teclado y ratón, y hoy usaremos software adicional del proyecto [p5-serial](#) creado por [Shawn van Every](#), para añadir soporte para comunicación serial.

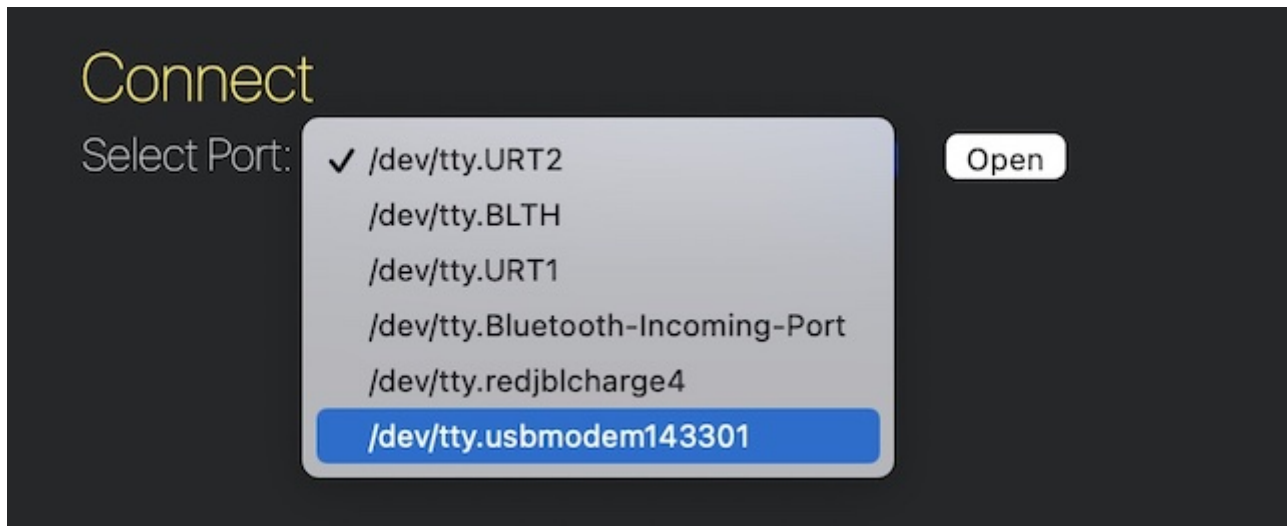
primero descargaremos el software [p5.serialcontrol](#) en nuestro computador. la versión más nueva en mayo 2022 es la "Beta 1.2" y se puede descargar desde la sección de [releases](#), al hacer click en "Assets" y escoger la versión necesaria para nuestro sistema operativo. Como en el laboratorio usamos computadores con Windows, descarguen la versión que dice "p5.serialcontrol-win32-x64.zip" con un tamaño de 67.6 MB.

tras descomprimir el archivo .zip pueden abrir la app p5.serialcontrol.

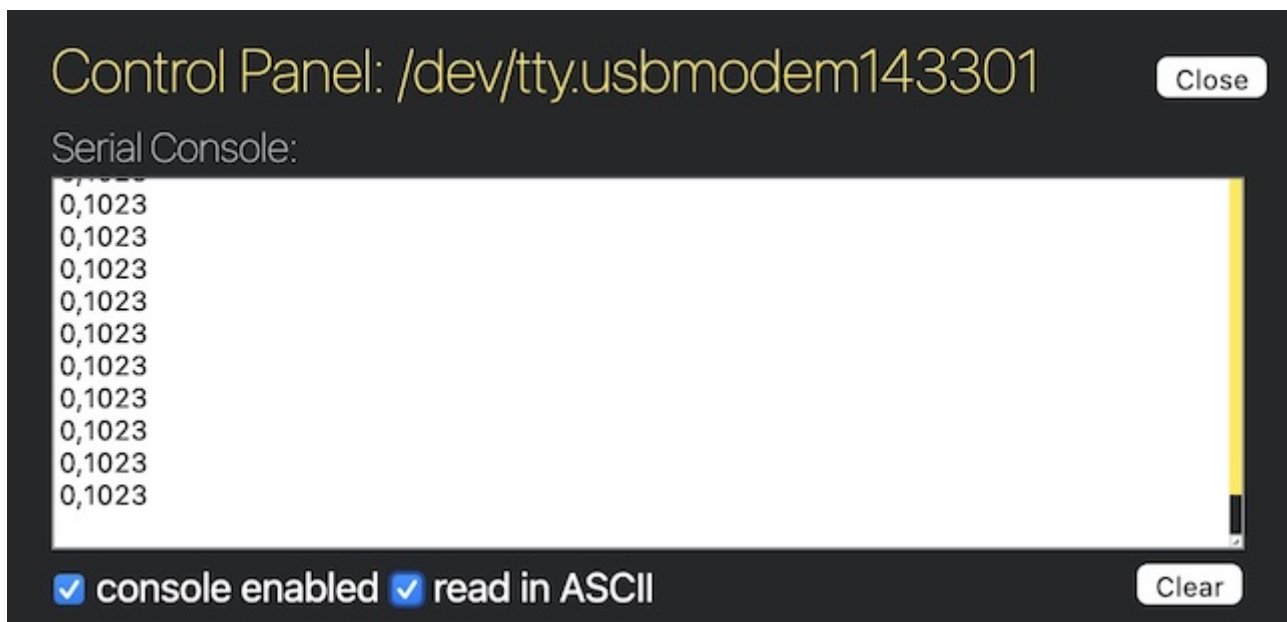


en la sección "Connect", elegir el puerto correspondiente a nuestro microcontrolador Arduino.





en la consola, abrir la consola con el botón "Open" y luego habilitar las dos cajitas que dicen "console enabled" y "read in ASCII", para poder leer los mensajes seriales desde Arduino, con las lecturas de estado de nuestro pulsador y potenciómetro.



si esto funciona, probaremos usando los ejemplos en los repositorios:

- <https://github.com/aud5i022-2022-1/clase-08-ejemplo-p5js-texto>
- <https://github.com/aud5i022-2022-1/clase-08-ejemplo-p5js-elipse>
- <https://github.com/aud5i022-2022-1/clase-08-ejemplo-p5js-camara>

estos ejemplos tienen en el archivo sketch.js, en su línea 13 el siguiente código

```
let nombrePuerto = "/dev/tty.usbmodem143301";
```

esta línea de código es la que me permitió que las páginas web reconocieran mi microcontrolador Arduino en el computador de mi casa, pero en el computador que están usando deberán reemplazar esta línea.

para esto, les pido que inicien sesión en su cuenta de GitHub, y hagan un "fork" del primer repositorio <https://github.com/aud5i022-2022-1/clase-08-ejemplo-p5js-texto>, con el botón Fork" en GitHub.



esto copiará los contenidos del repositorio original y lo agregará a su cuenta personal.

después de hacer eso, tenemos que pedirle a GitHub que publique nuestros archivos como una página web, eso se hace haciendo click en "Settings", luego en "Pages", y luego en el menú de "Source", eligiendo "main" y presionando el botón "Save".

si ahora entran a su web, que será del estilo <https://nombreUsuario.github.io/clase-08-ejemplo-p5js-texto/>, podrán ver el estado de su web.

abriendo la consola del navegador web podemos ver los nombres de los puertos de Arduino, y con eso actualizar el nombre del puerto de la línea 13 de sketch.js

tras grabar y esperar un poco, nuestra página funcionará y logrará leer datos desde nuestro microcontrolador Arduino!

## motivación de comunicación serial con p5.js (15 min)

los navegadores web y las páginas web nos brindan acceso a otras latitudes geográficas, archivos y bases de datos, y todo tipo de información de una manera inédita en la historia de la humanidad.

con nuestro microcontrolador Arduino podemos crear interfaces electrónicas y físicas de entrada y salida a la web.

enlaces de interés:

- [Sam Lavigne](#)
- [Stupid Hackathon](#)

## instrucciones proyecto mitad de semestre próxima semana (30 min)

la próxima semana haremos nuestro proyecto de mitad de semestre.

trabajen en grupos de 2 o 3 personas.

requisitos:

- deben enviarme un mail con los nombres de su grupo y el link de su repositorio
- el proyecto debe usar un microcontrolador Arduino
- deben crear un repositorio de GitHub
- el código de su Arduino debe estar en el repositorio
- deben agregar crédito sobre citas y referencias de código
- el archivo README.md del repositorio debe tener texto con influencias, instrucciones, resultados finales y aprendizajes

inspiración:

- [nicole he - grow slow](#)