

clase-05

martes 12 abril 2022, presencial

repaso clase anterior y programa hoy (10 min)

la clase pasada aprendimos:

- fundamentos de programación en Arduino
- comunicación serial entre Arduino y computador
- programar semáforo en Arduino
- construir circuito para semáforo

hoy aprenderemos:

- botón pulsador
- delay() y maneras de evitarlo
- potenciómetro
- PWM

repaso botón pulsador (60 min)

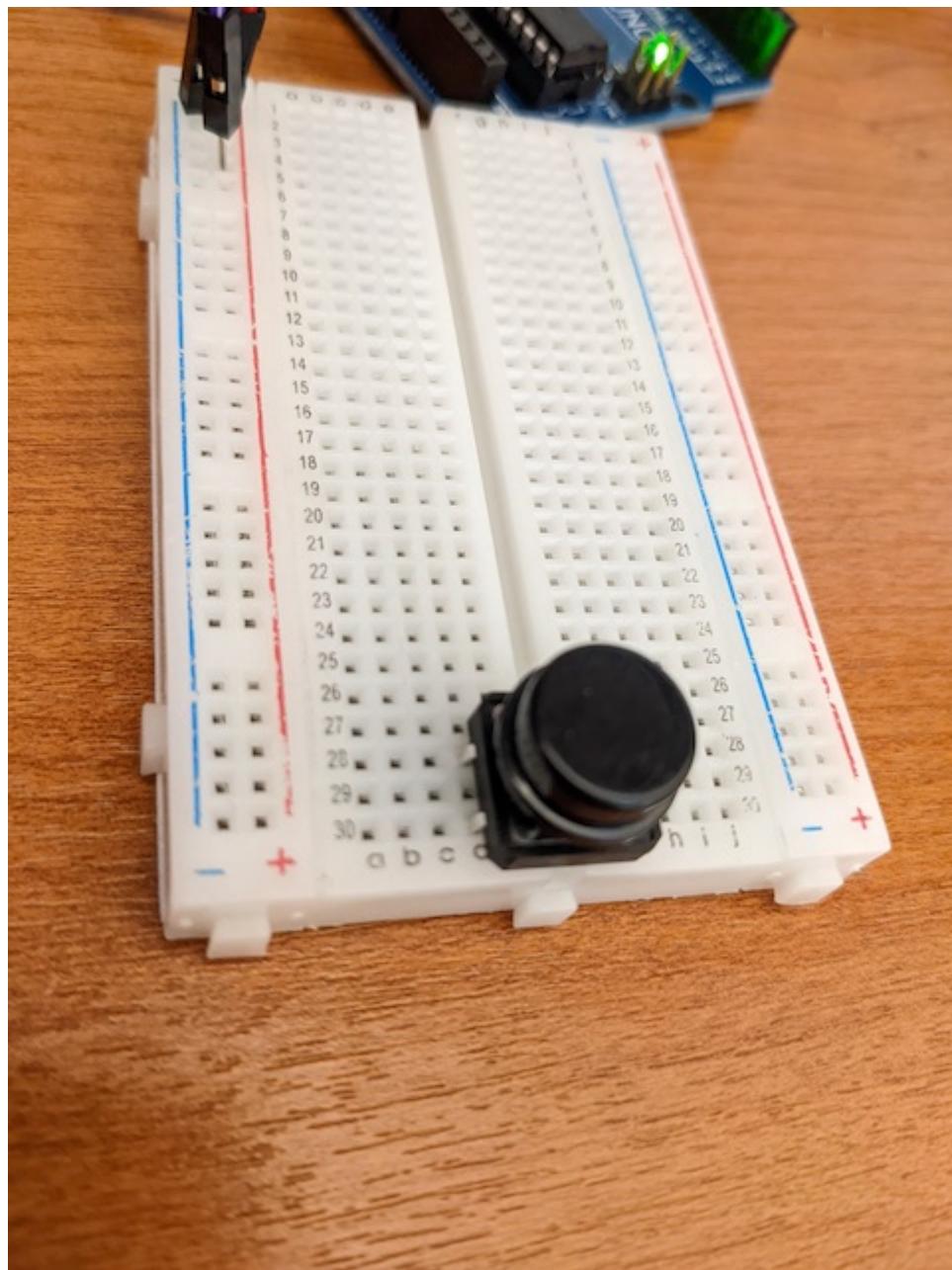
repasemos cómo conectar un botón pulsador como entrada digital al microcontrolador Arduino.

este es un pulsador.

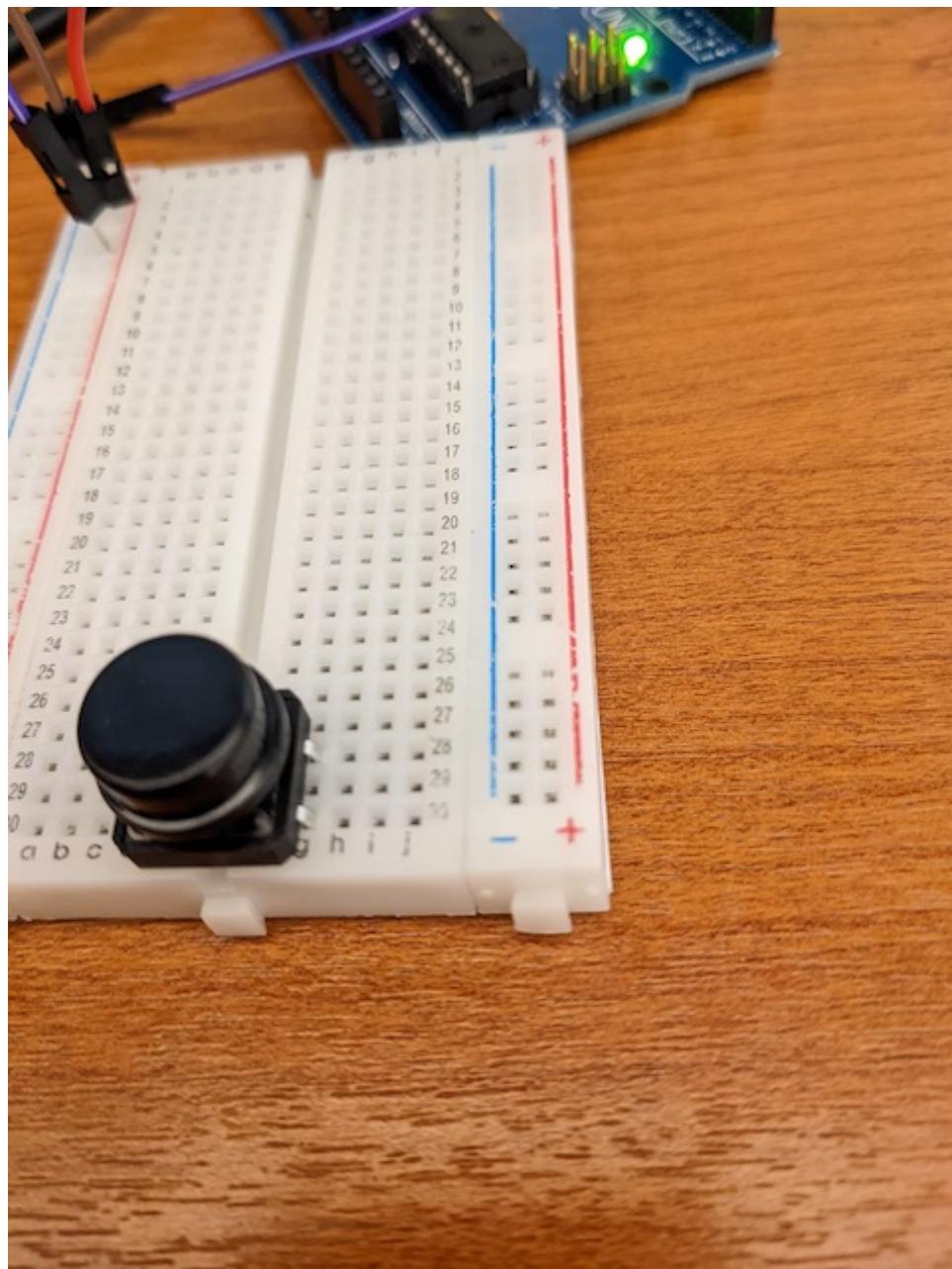


lo conectamos al extremo inferior de la protoboard, con pins repartidos entre columnas d y g.

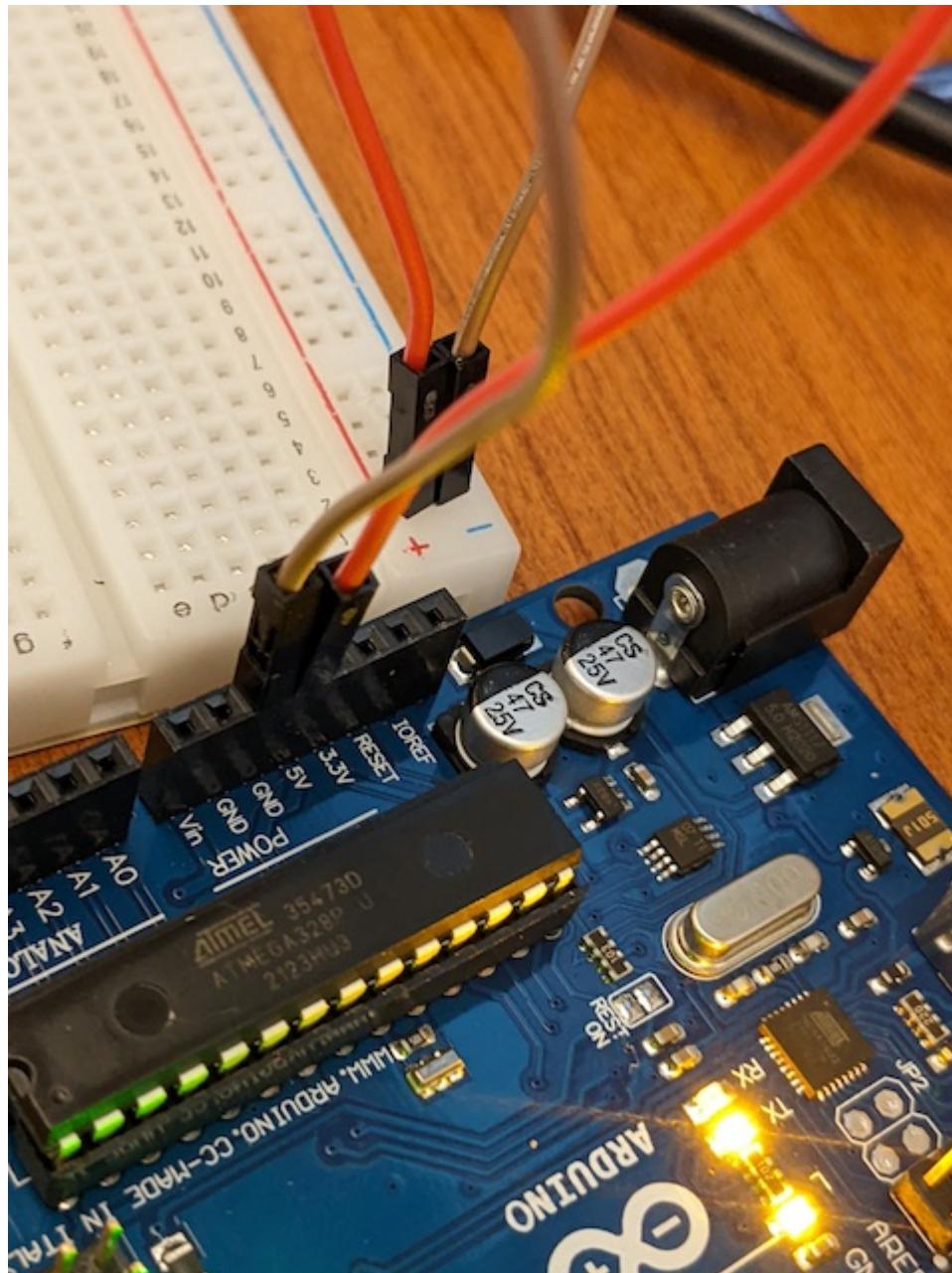
esta es la vista desde la izquierda.



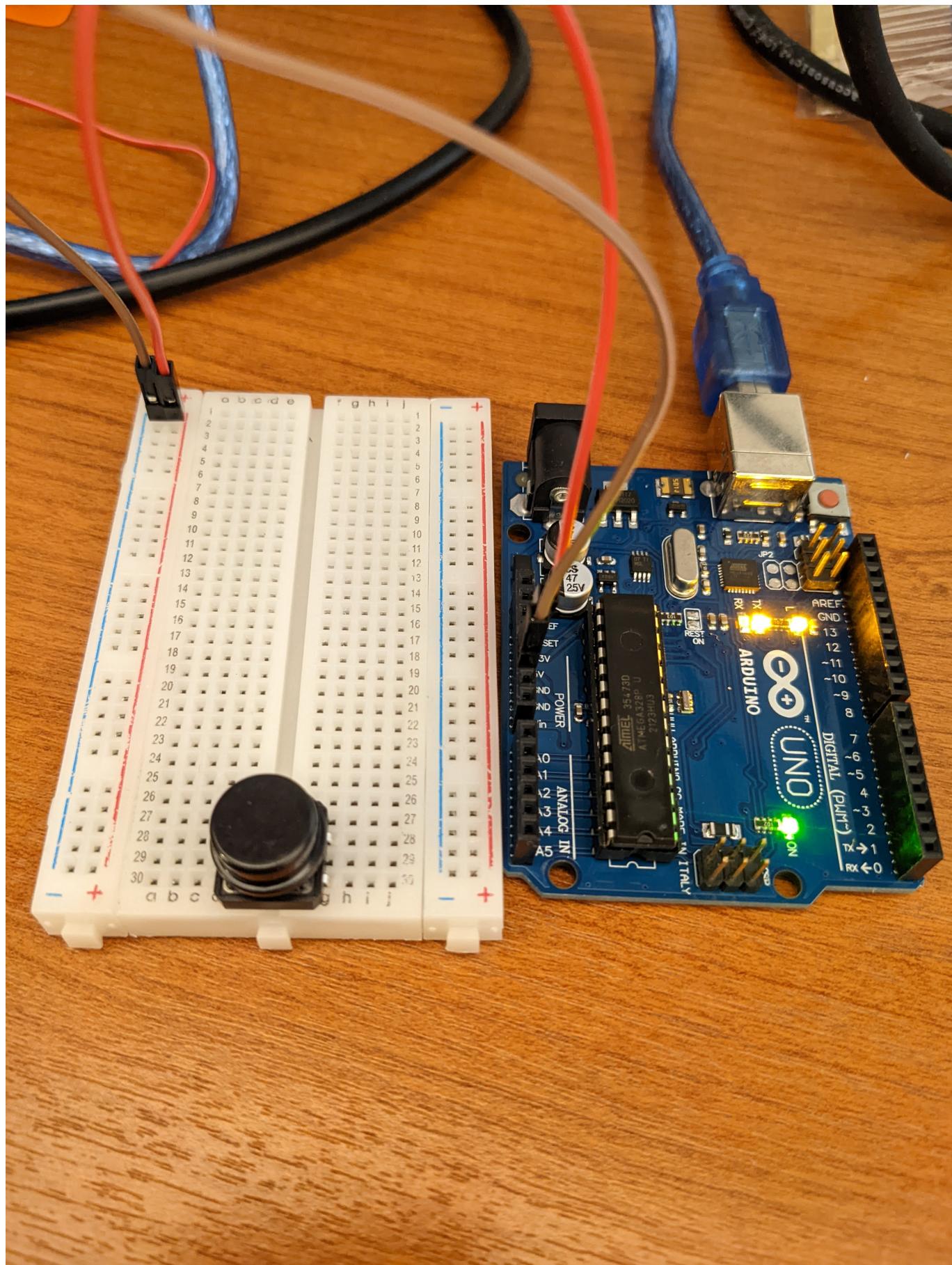
esta es la vista desde la derecha.



después conectamos cables a los pins 5V y GND de nuestro Arduino, convención es colores rojo o naranjo para voltaje de alimentación (5V) y colores café o negro para tierra (GND).

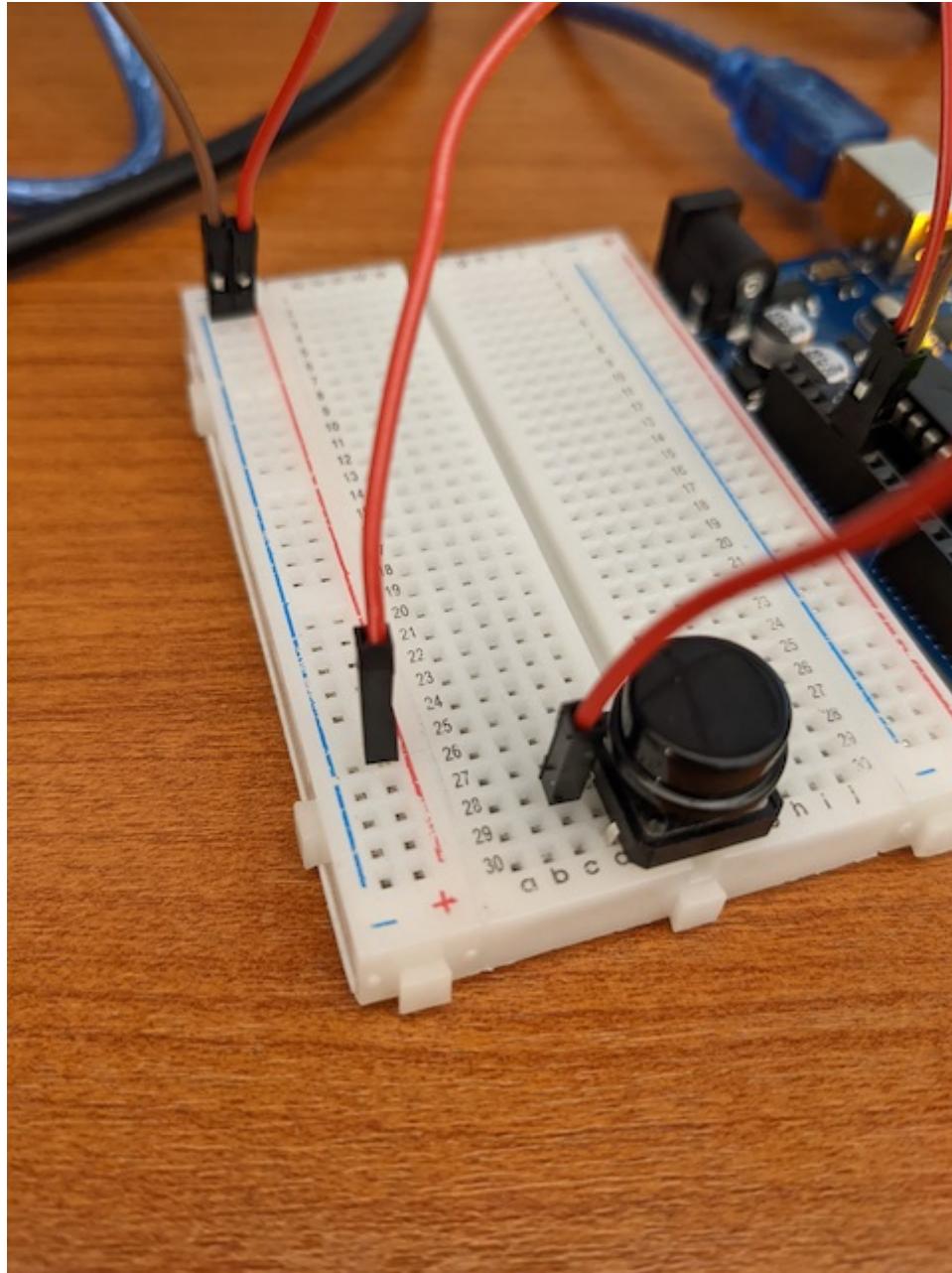


conectamos los otros extremos de los cables a los rieles de alimentación izquierdos de la protoboard. 5V desde Arduino se conecta a + rojo, y GND de Arduino se conecta a - azul.



hasta el momento el pulsador está solamente conectado mecánicamente a la protoboard, pero no tiene conexiones eléctricas.

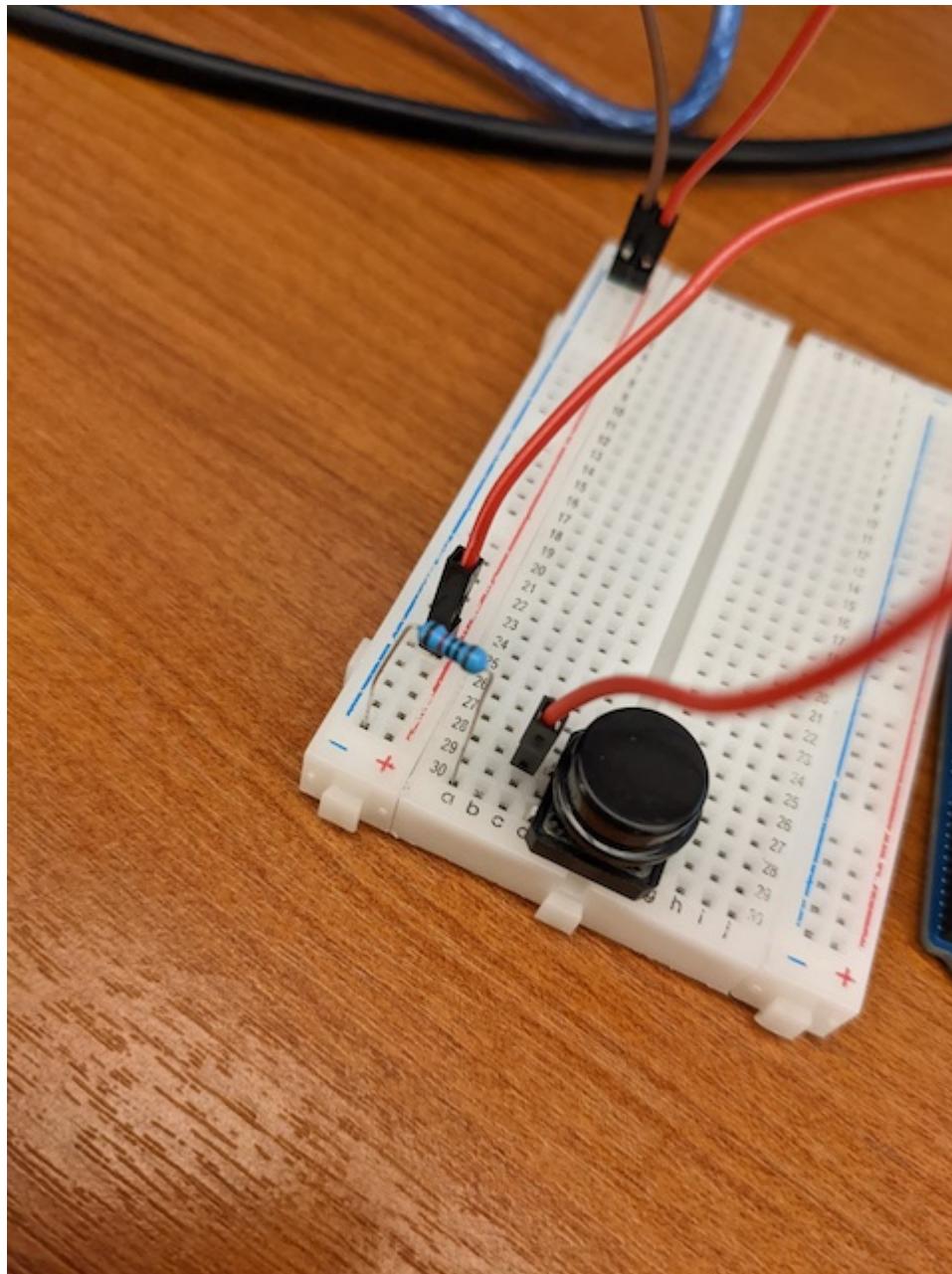
conectemos su pin superior a 5V con un cable rojo.



un circuito por donde circula corriente necesita de un camino entre los terminales de la fuente de poder, en nuestro caso los pins 5V y GND de Arduino.

queremos entonces conectar el otro pin del botón a tierra GND, pero no podemos hacer una conexión directamente con un cable, ya que eso haría un cortocircuito o conexión directa entre los terminales 5V y GND.

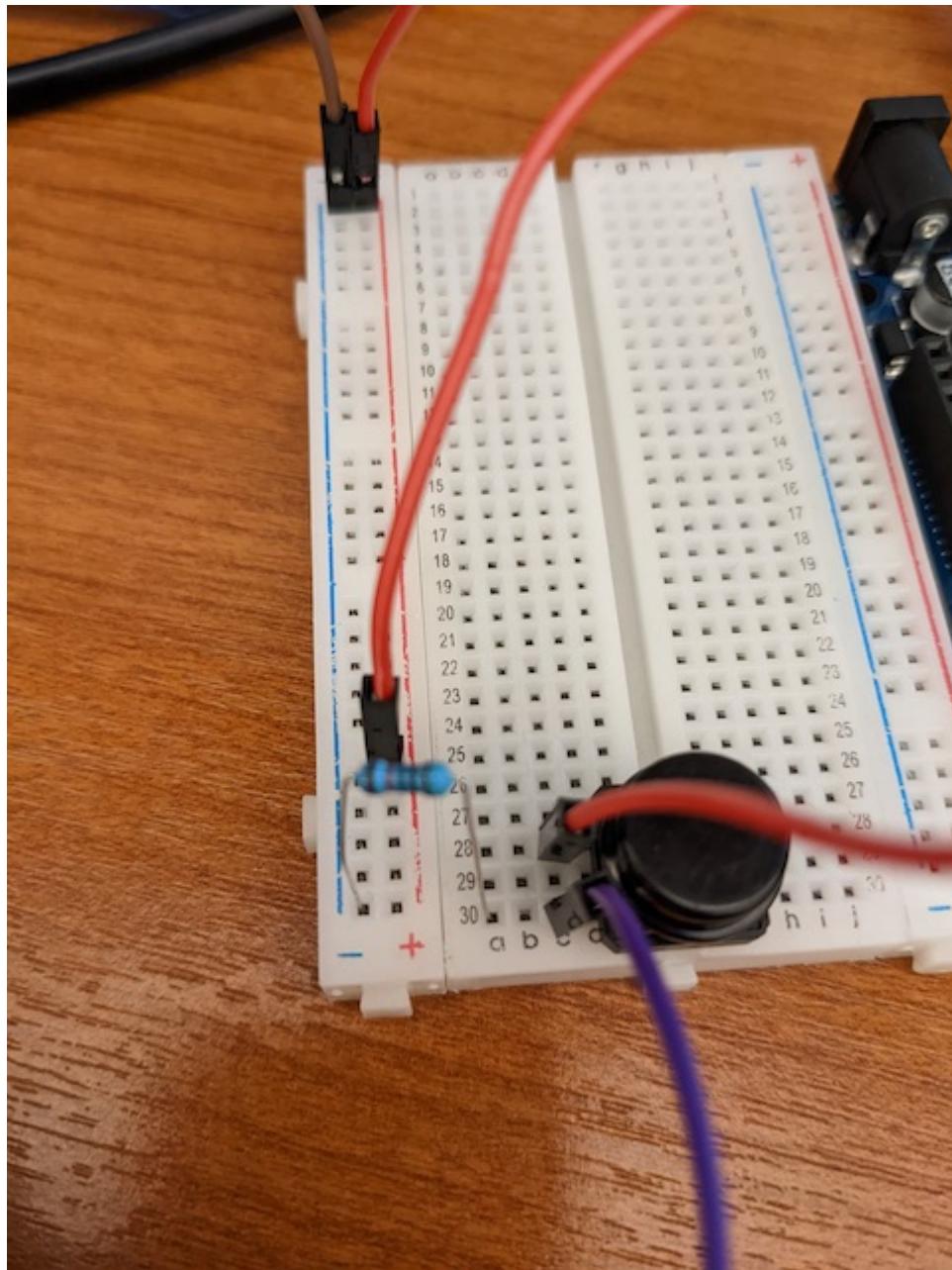
en vez de un cable, usamos un resistor para conectar el pin inferior a tierra.



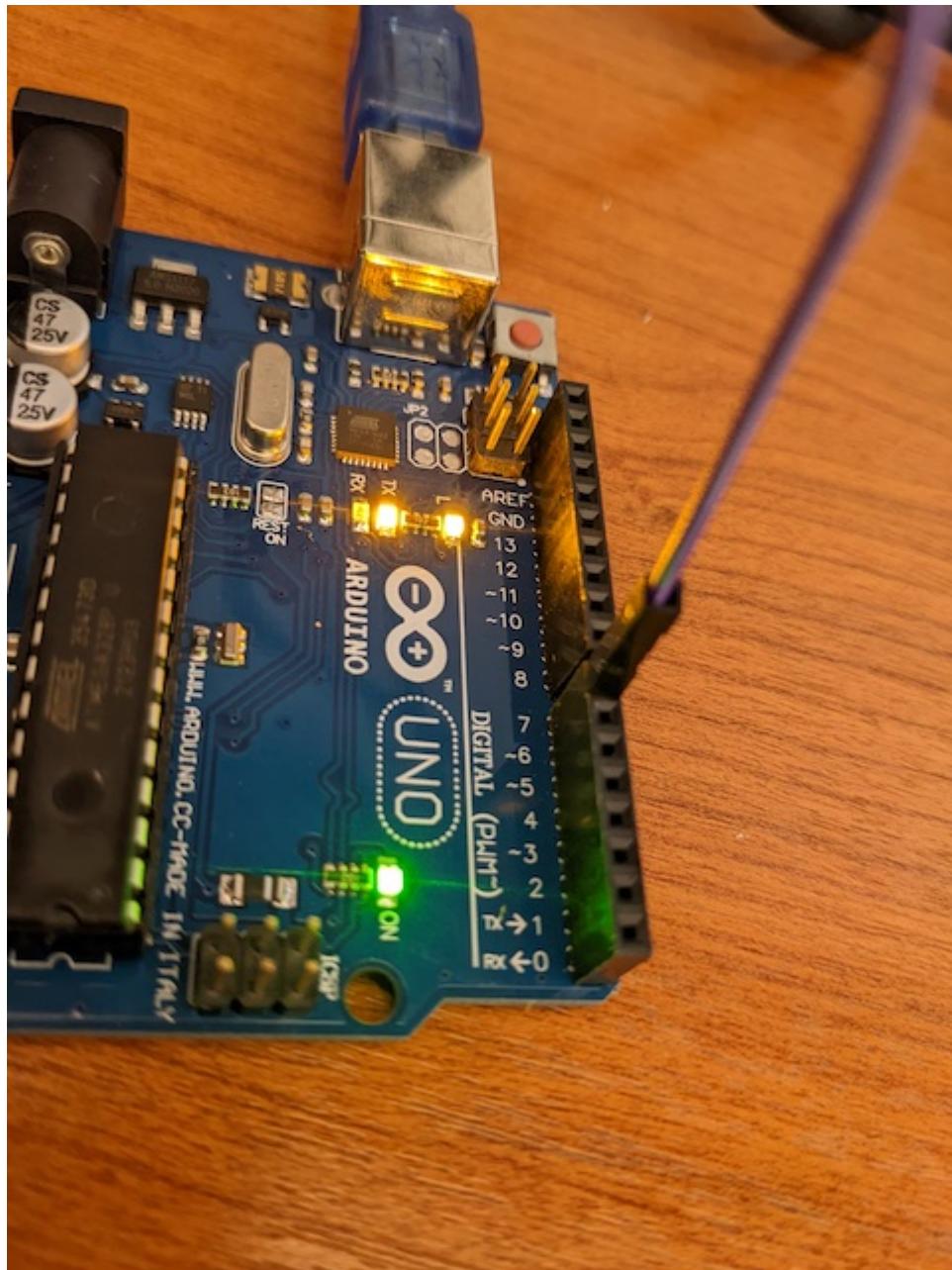
podemos leer el voltaje en los pines superior e inferior del pulsador, para obtener información digital, donde 1 es 5V y 0 es GND, en lógica digital.

el pin superior está conectado directamente a 5V, por lo que no aporta información, siempre es 1.

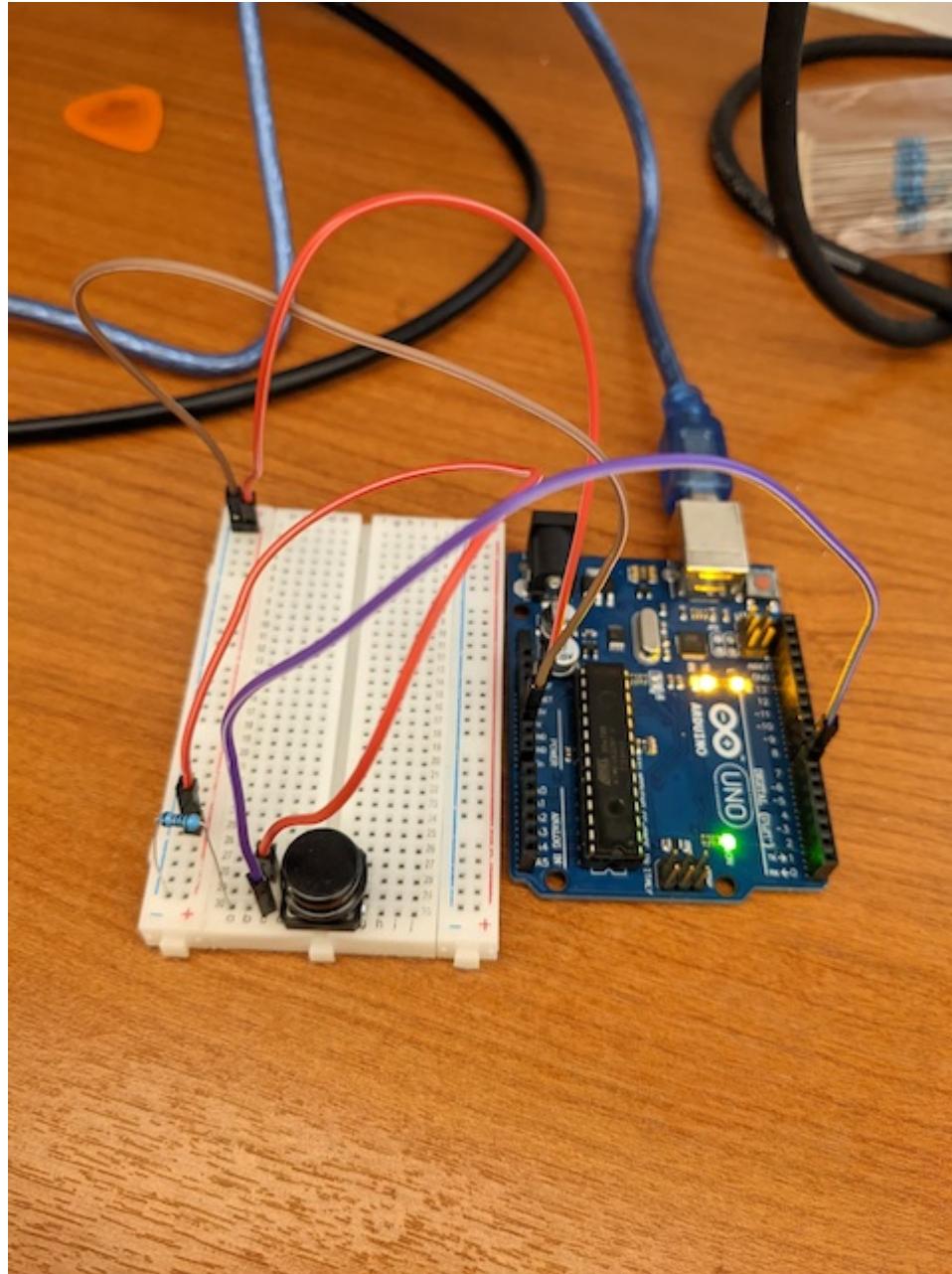
el pin inferior es el que alterna entre estados. está en 0 cuando no está pulsado, y cambia a 1 cuando está pulsado, por lo que conectamos un cable a esta fila en la protoboard.



el otro extremo del cable lo conectamos a la sección digital de Arduino, para este ejemplo al pin 7.



esta es la vista final del circuito.



subamos el código de [ej_00_lectura_pulsador](#) de lectura de pulsador a nuestro Arduino.

expliquemos con ley de Ohm la razón por la que el pin inferior alterna entre voltajes.

la ley de Ohm postula que $\Delta V = I * R$, lo que se puede interpretar así:

- $R = \Delta V / I$: resistencia R es la constante de proporcionalidad directa entre diferencia de voltaje ΔV y corriente I.
- cuando no hay corriente ($I = 0$) a través de una resistencia R, no hay caída de voltaje en una resistencia ($\Delta V = 0$), por lo que ambos terminales tienen el mismo voltaje.

cuando el pulsador NO ESTÁ PRESIONADO, no puede circular corriente ($I = 0$) a través de la resistencia R, por lo que no hay diferencia de voltaje ($\Delta V = 0$) entre los terminales de la resistencia R, por lo que ambos comparten voltaje. Como Arduino hace que un terminal sea GND ($V = 0$), el otro también es 0, y eso es lo que lee Arduino.

cuando el pulsador SÍ ESTÁ PRESIONADO, se convierte en un cable, que conecta 5V la resistencia y al pin de lectura de Arduino, y ese voltaje cae a GND a través de la resistencia, circulando corriente y disipando

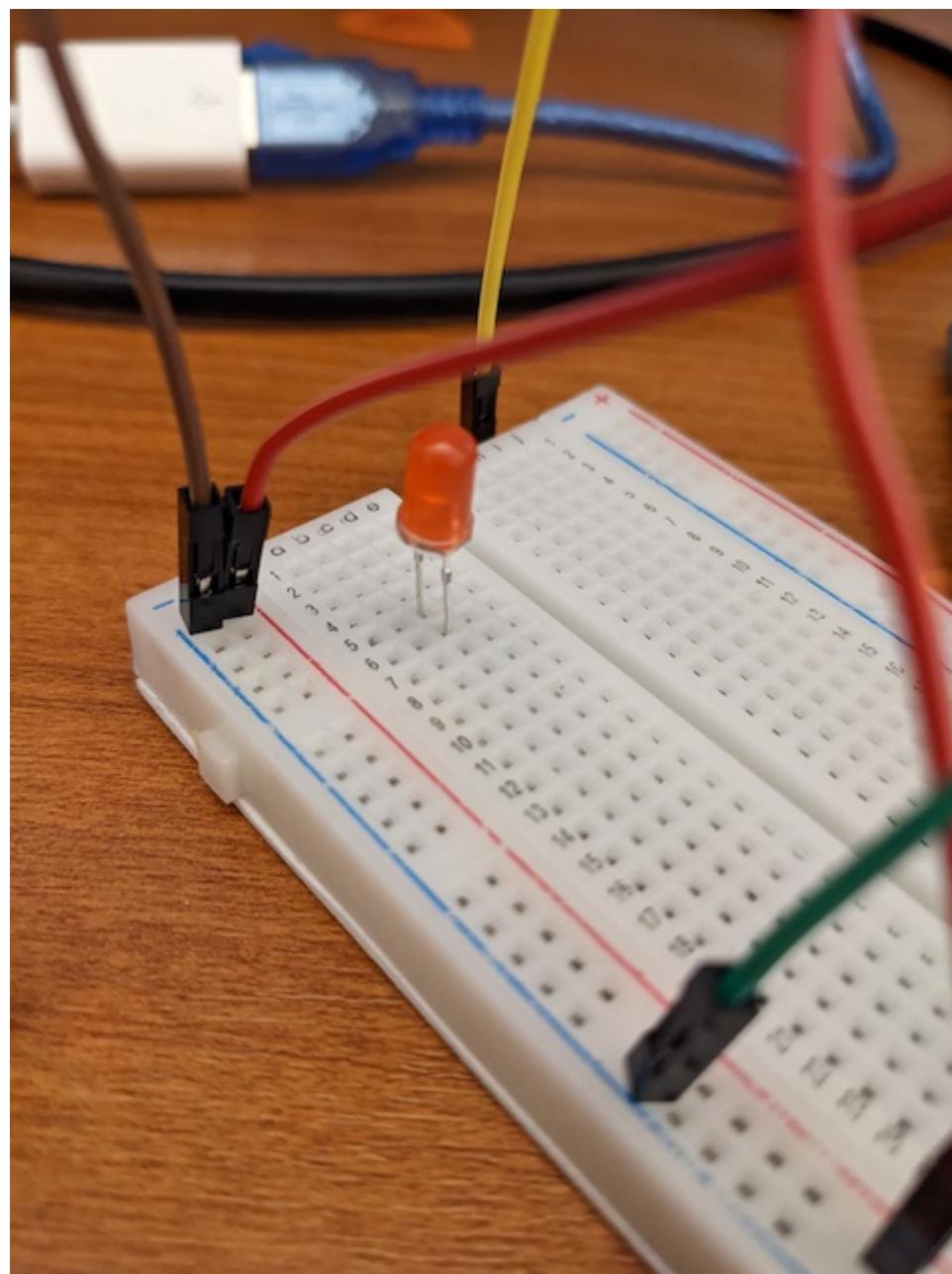
energía.

pulsador y luz (60 min)

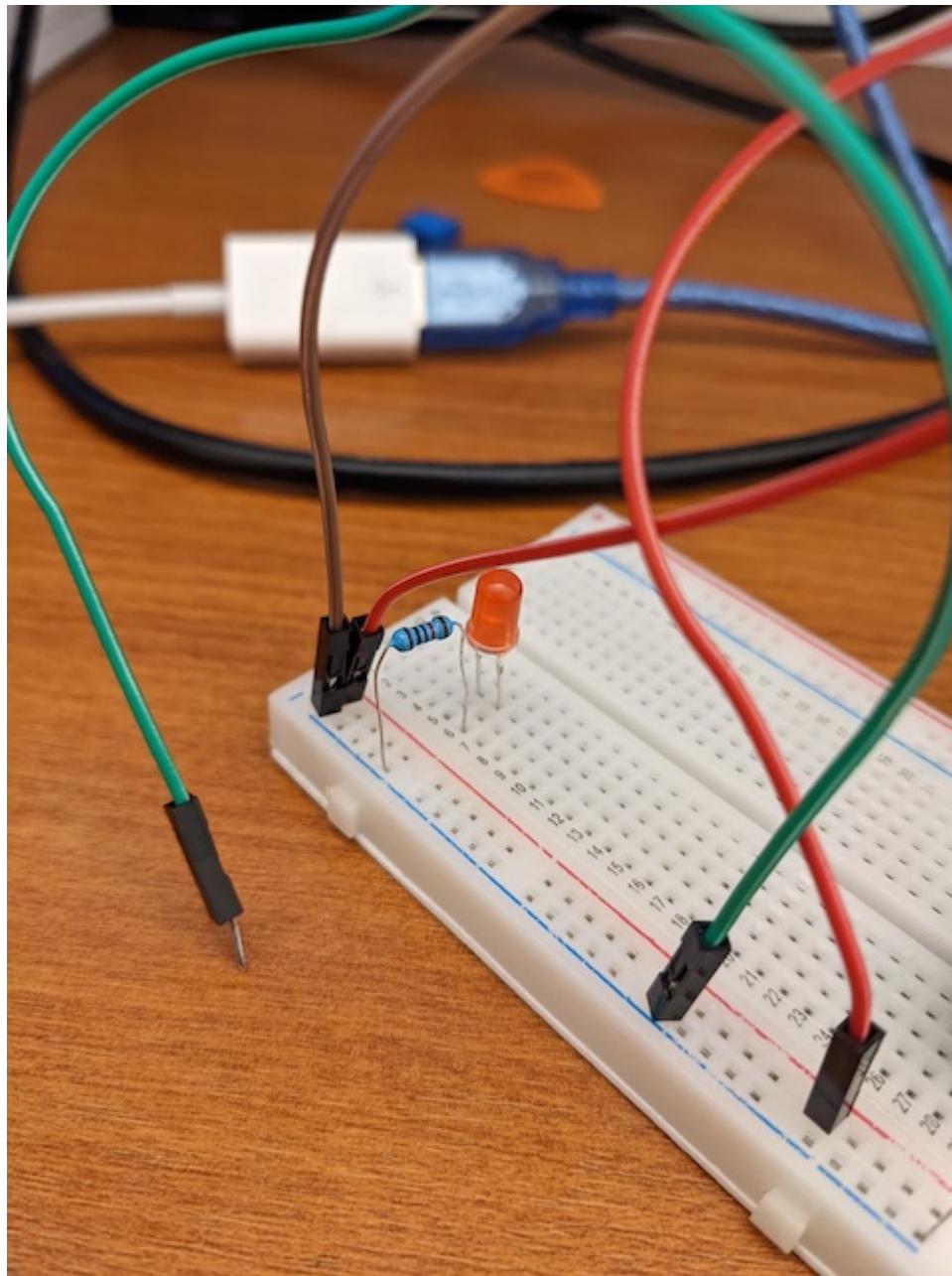
agregaremos a nuestro circuito un LED que pueda ser encendido o apagado según el pulsador.

seguiremos la convención de orientar terminales positivos arriba y terminales positivos abajo.

nuestro LED estará en la columna c de la protoboard, con el pin positivo en el terminal c5 y el pin positivo en el terminal c6. recordatorio: pin negativo (-) es el que tiene el lado plano.

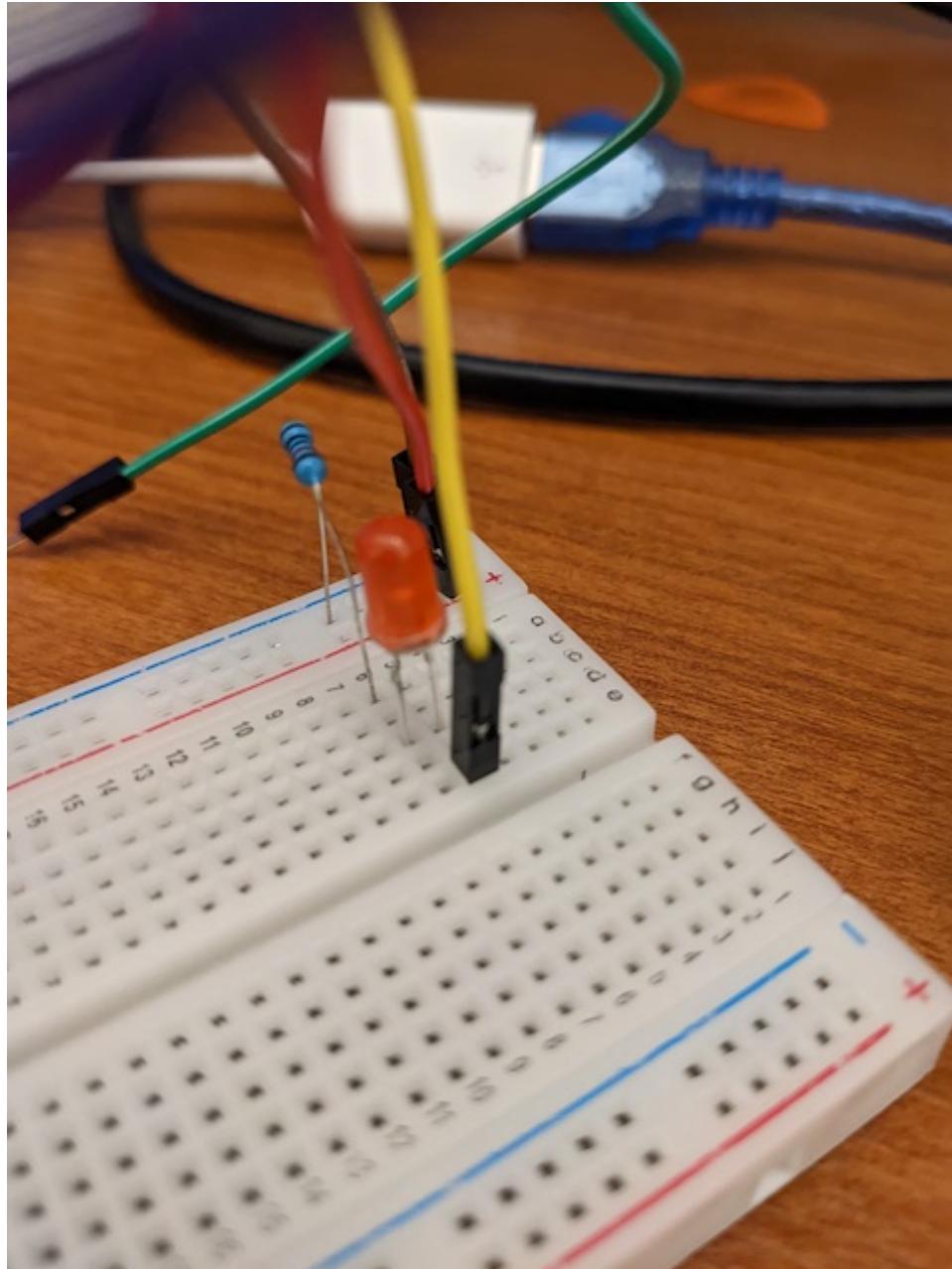


conectamos una resistencia entre tierra y la fila 6 para así conectar el pin negativo del LED a tierra a través de una resistencia, como lo hicimos con el pulsador.

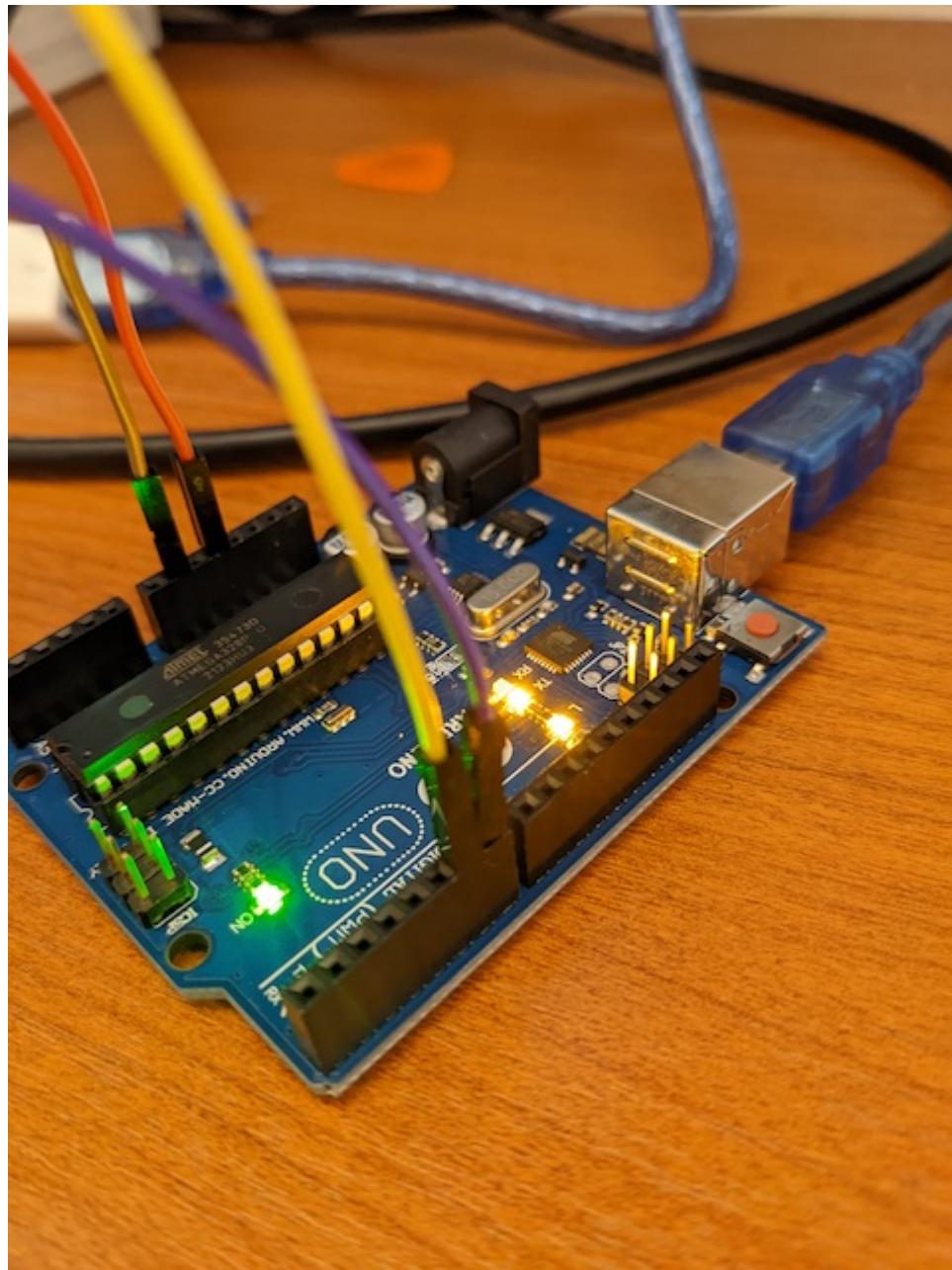


si conectamos el pin positivo del LED a 5V directamente, estará prendido sin interrupciones.

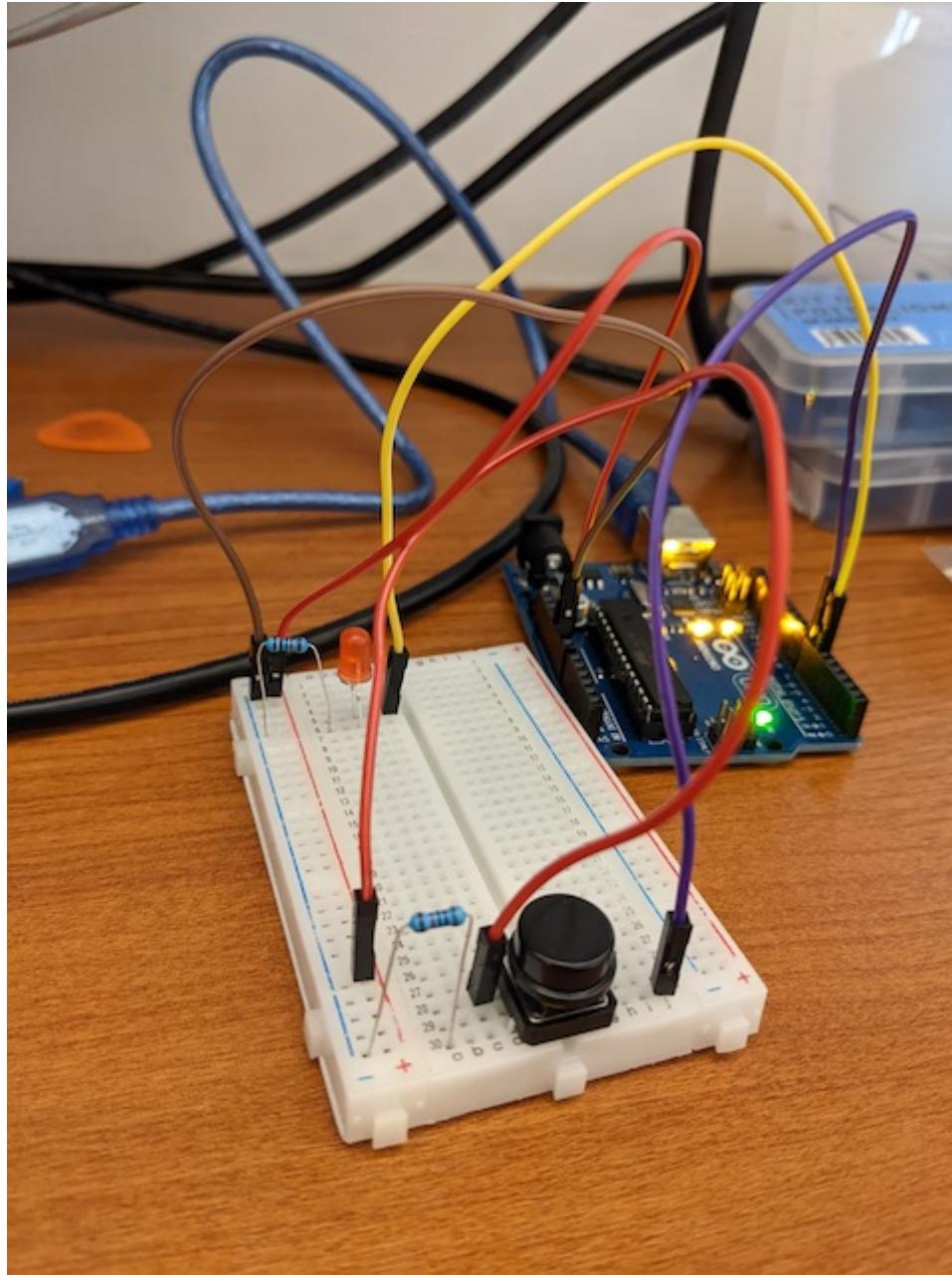
como queremos controlar el encendido y apagado del LED con el pulsador, conectaremos un cable a la fila 5 para conectarlo a un pin digital del microcontrolador Arduino.



usamos el pin 6 digital que está definido vía software como una salida digital, permitiendo así programar salidas de 5V o tierra para encender o apagar el LED respectivamente.



así se ve el circuito completo.



para probar este funcionamiento, subimos el código del ejemplo [ej_01_pulsador_luz](#).

ejemplos de luz intermitente (30 min)

usando el mismo circuito veremos 2 aproximaciones distintas a crear una luz intermitente.

en ambos, cuando el pulsador no está presionado, la luz está apagada, y cuando el pulsador sí está presionado, la luz alterna entre prendida y apagada de forma intermitente.

para la primera usaremos el ejemplo [ej_02_pulsador_luz_intermitente_delay](#), donde usamos la función delay() de Arduino para lograr la intermitencia.

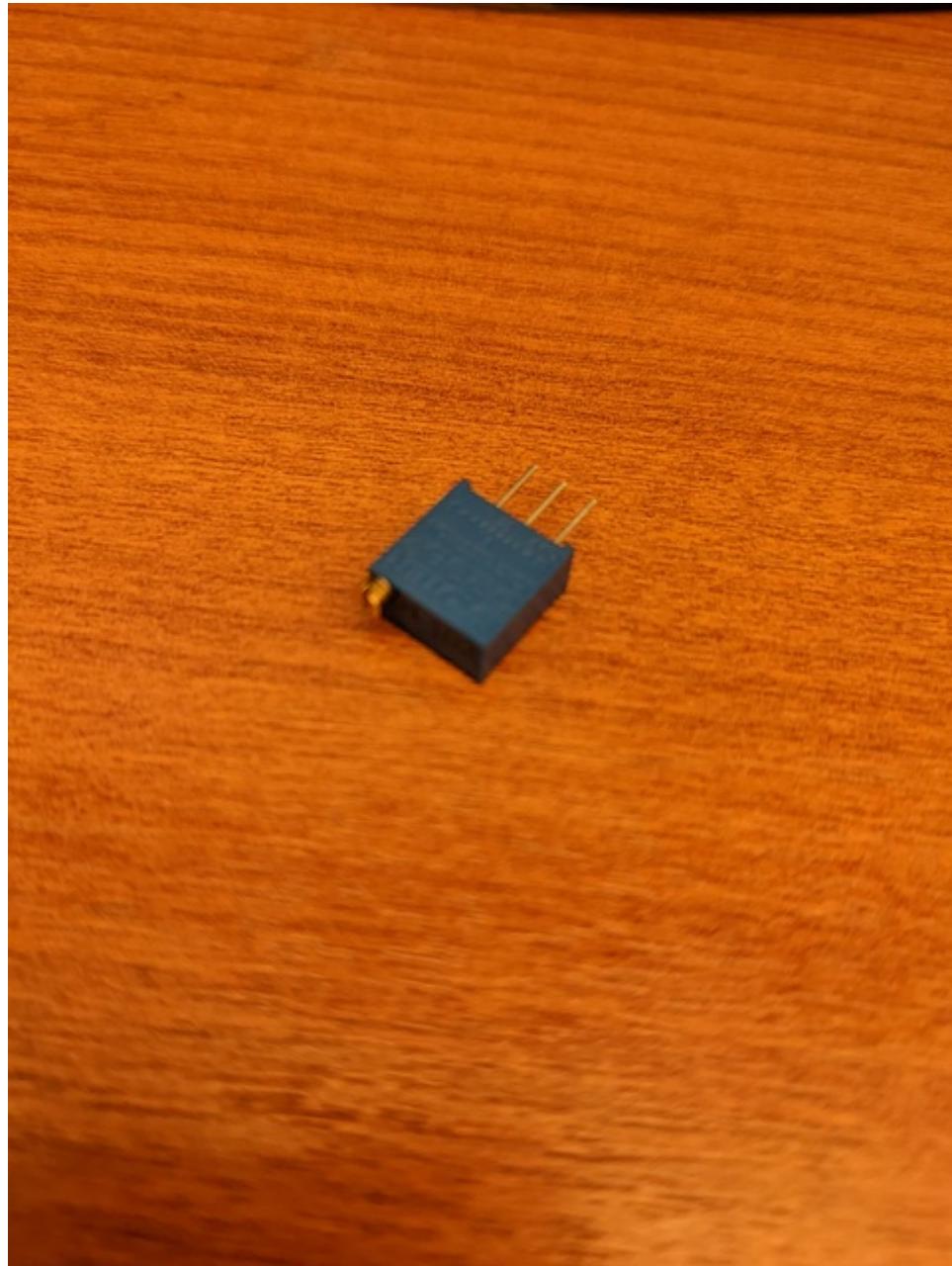
la función delay() introduce problemas de interacción, ya que mientras está corriendo esta función, Arduino no es capaz de leer ni escribir en sus otros pins.

por esto creamos el ejemplo [ej_03_pulsador_luz_intermitente](#), que es más complejo, y usa funciones de contador de milisegundos para detectar tiempo transcurrido, sin pausar el funcionamiento de Arduino al no usar funciones delay().

ejemplos de potenciómetro (30 min)

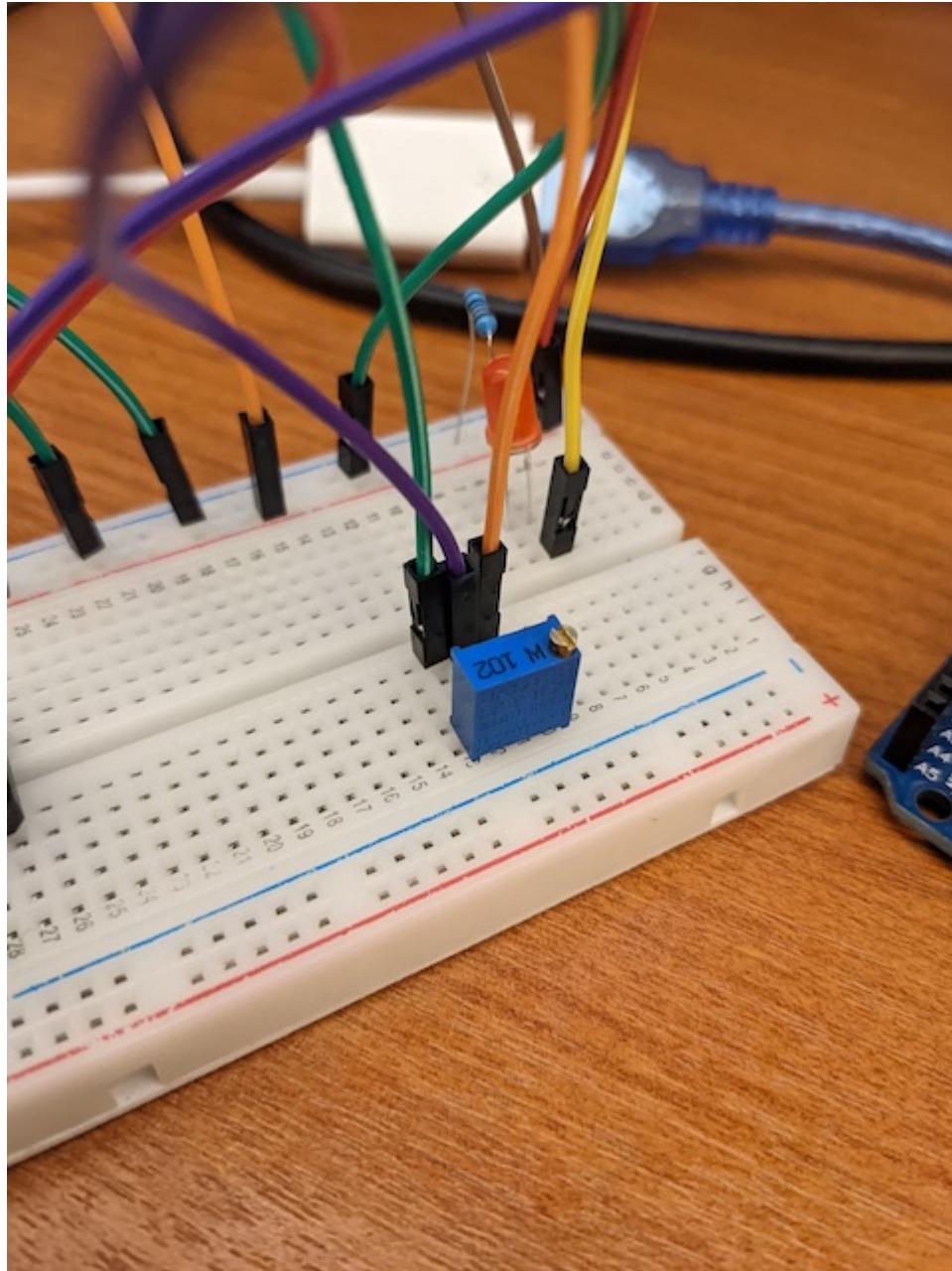
usaremos un potenciómetro de precisión, que posee 3 pines.

los pines de los extremos son conectados a poder (5V y tierra) y el del medio se usa como sensor de la perilla.



se recomienda esta manera de ubicarlo en protoboard para tener más espacio para operar su perilla.

al hacer girar la perilla en el sentido del reloj, el valor disminuye.



el ejemplo [ej_04_potenciometro/](#) que usaremos incluye las instrucciones como comentarios.

los ejemplos [ej_05_perilla_min_max](#) y [ej_06_perilla_mapeo](#) muestran maneras de detectar valores mínimo y máximo de forma automática y mapear a otro rango de operación.

el último ejemplo [ej_07_perilla_brillo_led/](#) permite cambiar brillo del LED con una salida PWM, que será la base de los ejemplos de sonido de la próxima semana.