



# Guía práctica Día 1


En esta guía vamos a aprender el manejo básico de la placa Arduino, cómo programarla cargando un programa (*sketch*) con el Arduino IDE (entorno de desarrollo integrado), y hacer uso de sus entradas/salidas analógicas y digitales. Durante la realización de los diferentes ejercicios propuestos, ahondaremos también en conceptos importantes a tener en cuenta al momento de diseñar un circuito a usar con Arduino y también al momento de registrar una señal analógica. Todo lo realizado aquí es extrapolable al ámbito de la experimentación del día a día en nuestros laboratorios.

<b>Usando el protoboard</b>	<b>1</b>
<b>Construir el riel de alimentación</b>	<b>2</b>
<b>Ejercicio A. Interactuando con los puertos digitales</b>	<b>3</b>
<b>Ejercicio B. Generando movimiento</b>	<b>6</b>
<b>Ejercicio C. Adquisición de datos a través del puerto analógico</b>	<b>9</b>
Rango de tensión del conversor A/D	11
Frecuencia de muestreo	12

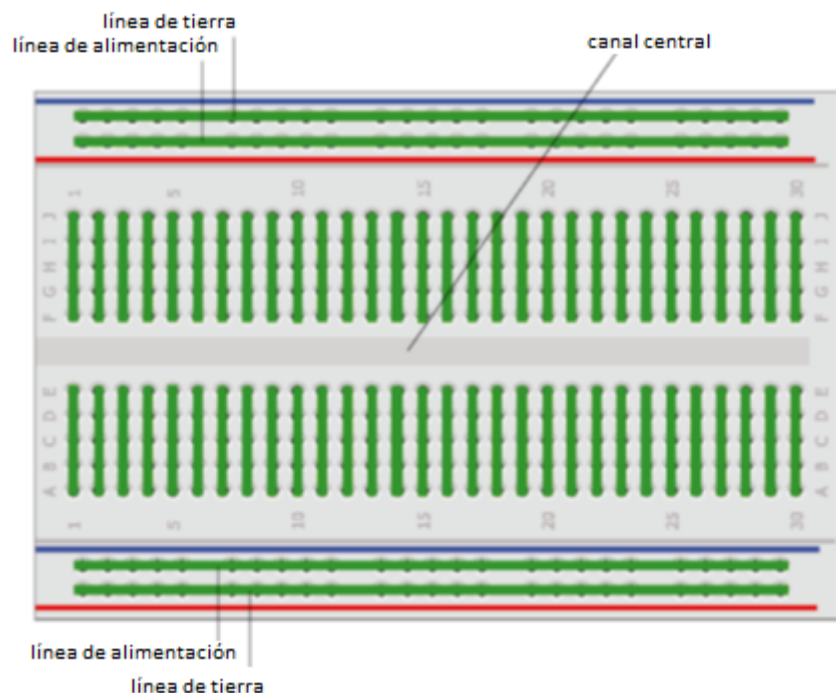
## Usando el *protoboard*

El *protoboard* nos va a permitir interconectar los diferentes componentes y entradas/salidas del Arduino mediante cables, sin tener que soldarlos entre sí.

Todos los pines adyacentes a la línea roja están conectados entre sí. Lo mismo sucede con los pines adyacentes a la línea azul. Estos pines están diseñados para ser usados como línea de alimentación o rieles de los circuitos a armar. Los pines “rojos” se usan para llevar la alimentación (generalmente 5V), mientras que los pines “azules” se usan para la conexión a

tierra (*ground*, GND, ).

Todo el resto de los pines están alineados verticalmente en columnas con una letra identificadora y unidos entre sí horizontalmente en filas numeradas. Además, entre estos últimos hay una división (canal central) que permite montar fácilmente los circuitos integrados.



## Construir el riel de alimentación

### ¡Advertencia!

*Asegurarse que el Arduino aún no esté conectado a nuestra computadora y que los cables que vamos a conectar entre la salida de tensión de 5V y GND de éste y el protoboard no se toquen entre sí. Si estuvieran en cortocircuito, fluiría una corriente muy grande, la suficiente como para freír nuestro Arduino.*

Dentro del kit provisto encontrarás una placa ARDUINO UNO junto con su respectivo cable USB para conectarla a la computadora. Además, encontrarás un pequeño gabinete plástico para proteger la placa, el cual recomendamos que uses.

El Arduino está conformado por un microcontrolador ATMEGA que posee entradas y salidas tanto analógicas como digitales que nos van a permitir interactuar con el mundo exterior a partir de las instrucciones que le carguemos (*sketch*). Podrás encontrar una guía con la disposición de los pines y las conexiones básicas de Arduino en el archivo [Arduino Pinout y Conexiones Basicas.pdf](#).

Lo primero que vamos a necesitar es construir el riel que proporcionará el voltaje necesario para los componentes electrónicos que vamos a usar y posteriormente para alimentar nuestro amplificador operacional en la próxima guía. Para ello, vamos a conectar nuestro Arduino al *protoboard* de la siguiente manera:

1. Conectar la salida de voltaje de 5V del Arduino mediante un cable macho-macho (rojo preferiblemente) a alguno de los orificios sobre la línea roja del *protoboard*.



2. Conectar el pin de GND del Arduino mediante otro cable macho-macho (negro preferiblemente) a alguno de los orificios sobre la línea azul del *protoboard*.

**¡Advertencia!**


*Tené cuidado al conectar a los pines del Arduino. Para no equivocarte de pin, mirá la placa desde arriba para ver bien las etiquetas, o contá los pines. Por ejemplo, AREF es el tercero y GND el cuarto desde el comienzo de la fila.*

## Ejercicio A. Interactuando con los puertos digitales

¡Comencemos a usar nuestro Arduino! En esta sección vamos a aprender a manejar los puertos digitales a partir de las funciones *pinMode()*, *digitalWrite()* y *digitalRead()*.

Vamos a empezar controlando el encendido de un diodo emisor de luz (LED) rojo mediante el uso de un pulsador.

Primero, vamos a identificar cuál es el LED rojo y cual es su terminal positivo (ánodo) y su terminal negativo (cátodo) mediante el multímetro.

1. Encender el multímetro en la función de medición de diodos  y colocar las puntas del multímetro entre los terminales del LED. Este se encenderá con un brillo mínimo si fue correctamente polarizado y la punta roja del multímetro indicará el terminal que corresponde al ánodo y la punta negra con el cátodo. En caso de que el LED no se encienda, intercambiar las puntas de medición. El cátodo también puede ser identificado ya que presenta un borde recto en la base del encapsulado plástico del LED.

Los diodos LED tienen un voltaje y una corriente de trabajo nominal para que se enciendan y brillen de forma óptima. En el caso de nuestro LED rojo, la corriente es de 20mA y su voltaje es de 1.8V. La corriente no debe superarse porque el dispositivo podría quemarse. Por lo tanto, es necesario conectarlo en serie con una resistencia que limite la corriente máxima que circula por el LED. Para calcular esta resistencia vamos a usar la ley de Ohm.



$$V - V_{LED} = I_{LED} * R$$

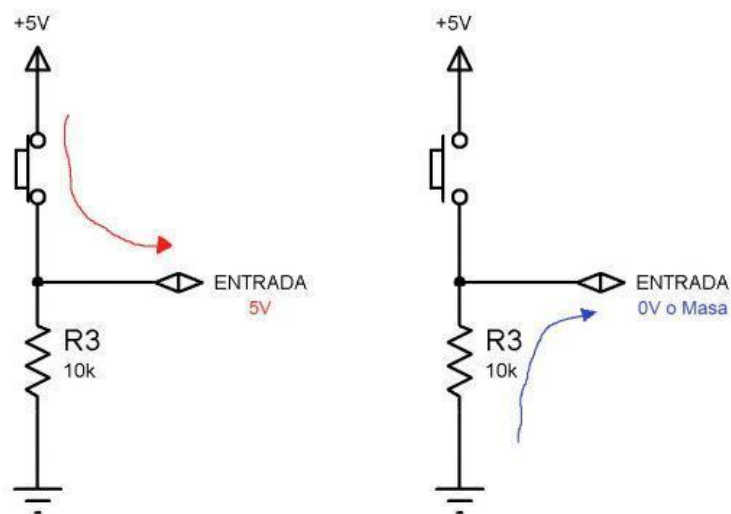
$$R = \frac{V - V_{LED}}{I_{LED}} = \frac{5V - 1.8V}{0.02A} = 160\Omega$$

2. Conectar el LED rojo en serie con dos resistencias de 100Ω entre el pin 7 del Arduino y la línea de GND del riel de alimentación del *protoboard*. El cátodo del LED (borde recto) debe ir conectado a la tierra GND. Recomendamos que ubiques todos los



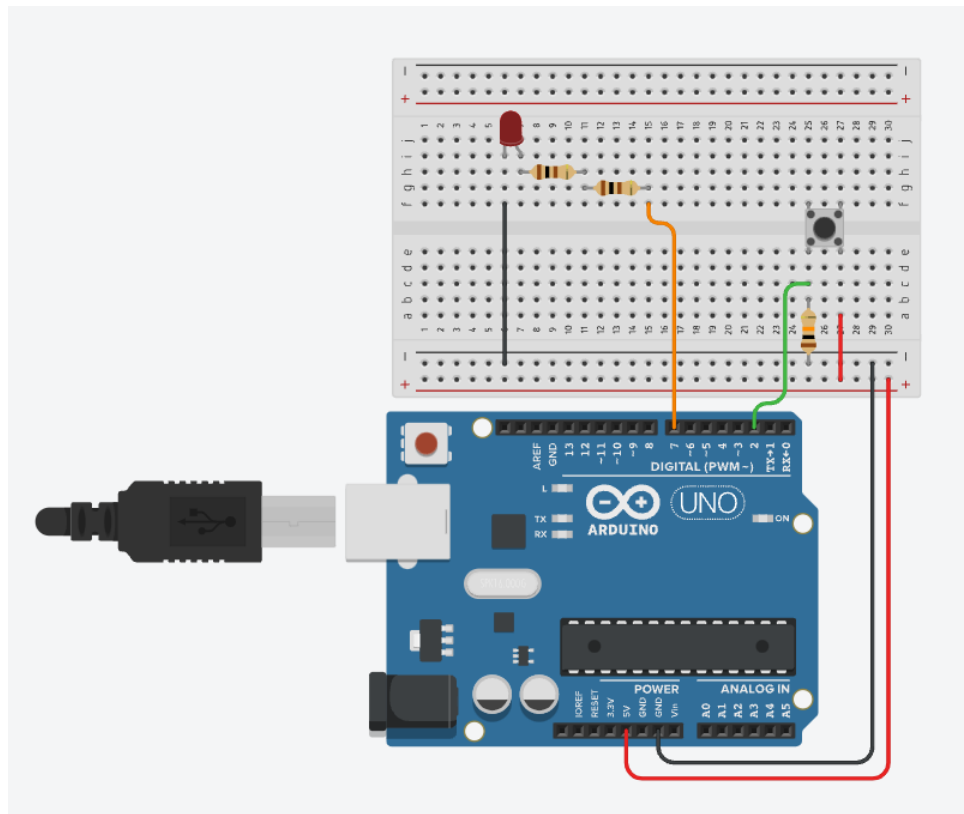
componentes en un extremo del *protoboard* para dejar espacio para los componentes del siguiente ejercicio.

Ahora, vamos a conectar el pulsador . Queremos que, al presionarlo, nuestro Arduino lo detecte a través de su entrada digital, procese esa información y encienda el LED. Para que la entrada digital detecte la presión del pulsador, vamos a enviar un nivel lógico alto: 5V. Esto lo vamos a lograr con ayuda de una resistencia  en la configuración *pull-down* que se esquematiza a continuación.



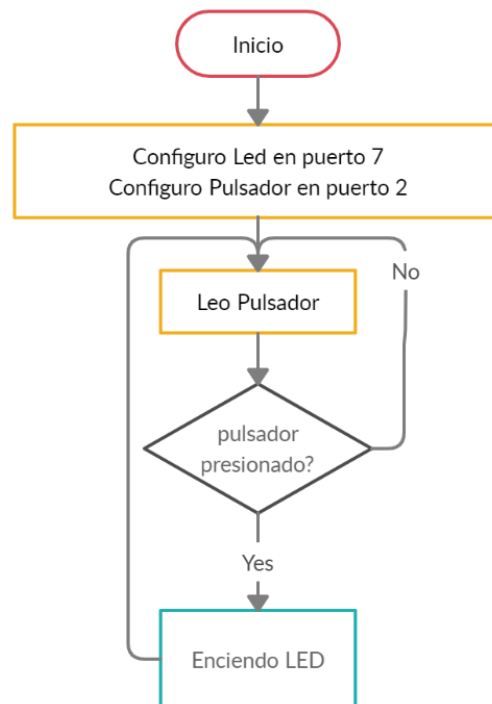
La finalidad de esta resistencia es asegurar que se mantenga un nivel lógico bajo (0V) en la entrada del Arduino siempre que el pulsador no esté presionado, y un nivel lógico alto (5V) únicamente cuando se lo presiona.

3. Conectar el pulsador al pin 2 del Arduino (la “Entrada” en el esquema anterior) usando una resistencia de *pull-down* de 10K, como se muestra a continuación:



4. Editar y cargar al Arduino el siguiente [A pulsador y led.ino](#), configurando el pin 7 como salida y el pin 2 como entrada. ¡Atención! Deberás corroborar que el Arduino IDE esté configurado en el puerto USB correspondiente.

A continuación, puedes ver un diagrama de flujo resumiendo el programa del sketch que subimos al Arduino. Realizar estos diagramas de flujo para organizar cómo se comportarán nuestros programas es una buena práctica cuando comenzamos a programar. ¡Intenta usarlos!

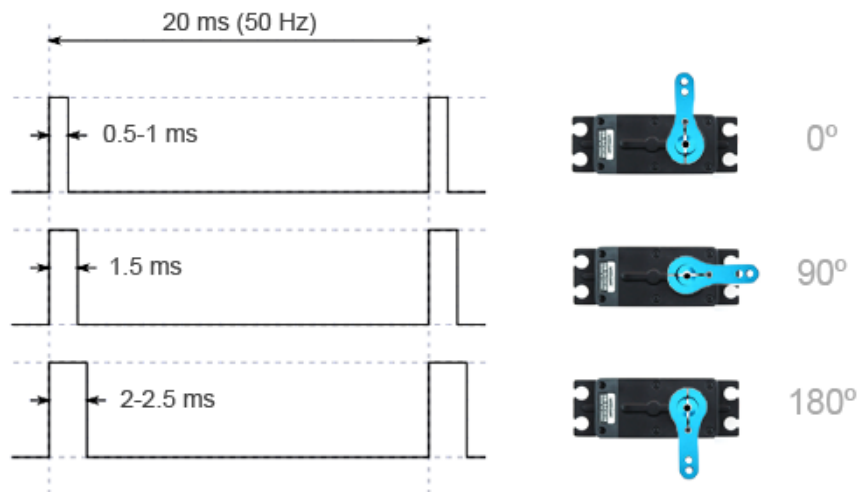


**EXTRA:** modificar el programa para que el LED quede encendido al presionar el pulsador y se apague al presionarlo de nuevo. ¿Te animás a intentarlo? La solución la podés encontrar en [A\\_pulsador\\_y\\_led.ino](#) y el diagrama de flujo en [03\\_pulsador\\_y\\_led\\_con\\_retencion\\_flowchart.pdf](#).

## Ejercicio B. Generando movimiento

Ahora que ya sabemos cómo utilizar los puertos digitales de nuestro Arduino, vamos a dar un paso más, interactuando con el mundo externo utilizando un servomotor para generar movimiento. Un servomotor, o más comúnmente servo, es un motor al que le indicamos directamente el ángulo deseado y éste se encarga de posicionarse en ese ángulo. Por lo tanto, en todo momento podemos saber la ubicación exacta del servo. Típicamente disponen de un rango de movimiento de entre 0 a 180°. Es decir, no son capaces de dar una vuelta por completo.

La información del ángulo de giro deseado se realiza mediante la transmisión de una señal pulsada (PWM) con un periodo de 20 ms. El ancho del pulso va a determinar la posición del servo. Para manejar nuestro servo vamos a hacer uso de la librería de Arduino, *Servo.h*.



### ¡Advertencia!

*En el caso de utilizar servomotores, motores paso a paso, o cualquier otro dispositivo que pueda necesitar un consumo considerable de corriente, es muy importante utilizar una fuente de alimentación externa. Caso contrario podríamos exigirle demasiado corriente a nuestro Arduino, destruyéndolo e incluso dañando en el puerto USB de la computadora. En este ejercicio, al tratarse de un servo pequeño y no ejercer fuerza sobre el mismo (consumo mínimo de corriente) y por cuestiones didácticas, la corriente que nos brinda la placa Arduino es suficiente.*

Vamos a realizar un montaje empleando el servomotor SG90, dos pulsadores y dos LED. Los pulsadores harán girar el servomotor en uno u otro sentido y los LED indicarán el sentido del servo. Debés colocar el accesorio (palanca de un brazo) en el eje del servomotor para poder observar el ángulo de giro.

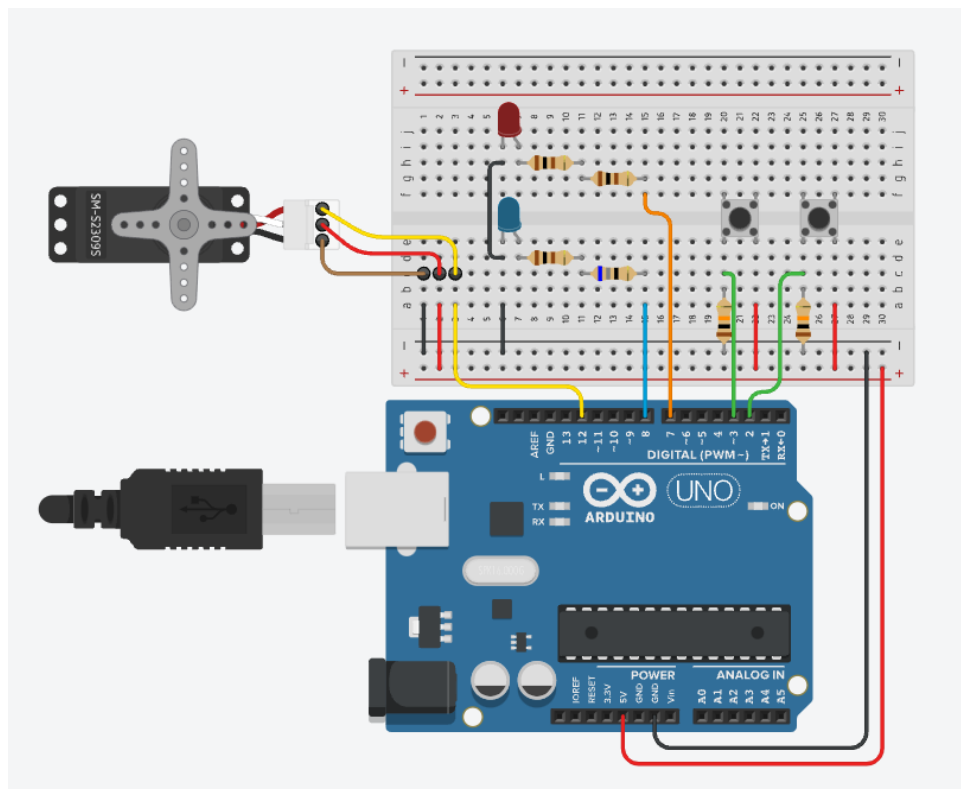
En este montaje vamos a utilizar 3 pines de medición. Debés separarlos mediante la ayuda de la pinza de punta y que el separador plástico quede en la mitad de lo mismos como se muestra en la figura:



1. Colocar el servomotor en el *protoboard* por medio de los 3 pines de medición. Conectar el cable rojo y marrón del servo a las líneas de 5V y GND del riel de alimentación, respectivamente.
2. Conectar el cable amarillo del servo al pin 12 del Arduino.



3. Conectar el LED rojo en serie con dos resistencias de  $100\Omega$  entre el pin 7 del Arduino y la línea de GND del riel de alimentación del *protoboard* (pueden usar la configuración del LED rojo del ejercicio anterior).
4. Conectar el LED azul en serie con una resistencia de  $100\Omega$  y otra de  $68\Omega$  entre el pin 8 del Arduino y la línea de GND del riel de alimentación del *protoboard* (¿Cuál sería la corriente que circularía por este LED azul?).
5. Conectar un pulsador al pin 2 del Arduino mediante una resistencia de *pull-down* de 10K (pueden usar la configuración del pulsador del ejercicio anterior).
6. Conectar otro pulsador al pin 3 del Arduino mediante una resistencia de *pull-down* de 10K.
7. Editar y cargar al Arduino el sketch [B control servo pulsadores.ino](#), configurando como entrada o salida los pines de los LED y pulsadores según corresponda.



**EXTRA:** solo modificando el programa, haz que el LED rojo se encienda y quede encendido cuando el servomotor llegue al final de su recorrido. El led azul deberá encenderse mientras el servomotor esté funcionando normalmente y el led rojo deberá apagarse. TIP: verifica el estado de ambos pulsadores y asígnale la condición necesaria de salida para los LEDs. La solución podrás encontrarla [B EXTRA control servo pulsadores.ino](#).





## Ejercicio C. Adquisición de datos a través del puerto analógico

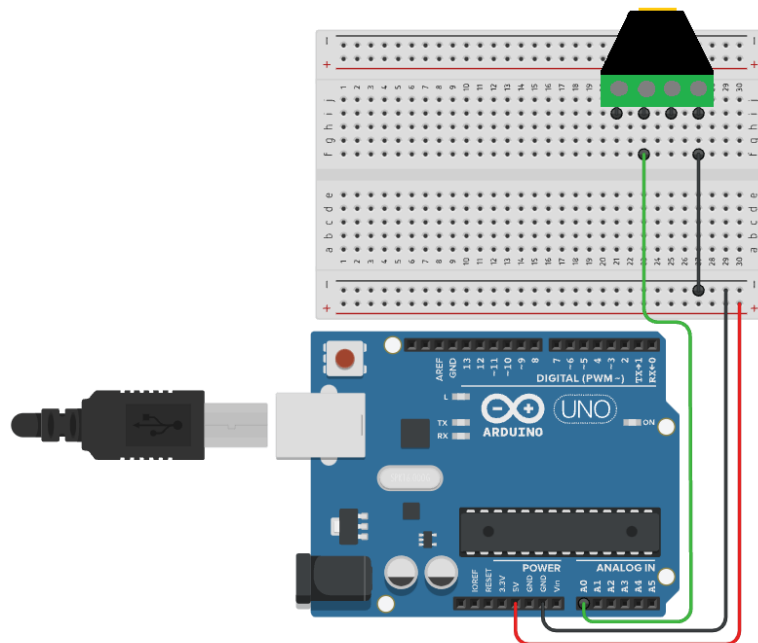
Ahora vamos a aprender a usar las entradas analógicas del Arduino para adquirir una señal analógica. Para esto vamos a hacer uso de las funciones `analogRead()` y `analogReference()`.

El microcontrolador de la placa Arduino solo entiende de números digitales (bits), por lo tanto, no entiende cómo una señal varía su amplitud en el tiempo. Para “traducir” las señales analógicas a digitales, hacemos uso de un conversor Analógico/Digital (ADC por sus siglas en inglés). Las entradas de los conversores A/D que dispone nuestra placa Arduino se corresponden con los pines marcados como A0 – A5. El valor máximo del rango de tensiones que utilizará el conversor para digitalizar la señal analógica estará determinado por el valor de tensión establecido en el pin de referencia (AREF) del Arduino.

Esta señal analógica puede provenir de diferentes fuentes como, por ejemplo, un sensor fotorresistivo o una señal biológica luego de ser amplificada. En este caso, vamos a usar un generador de señales online con el cual generaremos nuestra señal analógica.

1. Colocar en el *protoboard* el jack hembra de 3.5mm como se muestra en la imagen.

Este jack tiene 4 pines, de los cuales vamos a usar solo 3: el L, R y  $\text{GND}$  (canal izquierdo, canal derecho y tierra, respectivamente).





2. Conectar, mediante un cable macho-macho, el pin de tierra del jack al riel de tierra del *protoboard*.
3. Conectar, mediante un cable macho-macho, el pin del canal derecho (R) a la entrada analógica A0 de la placa Arduino.
4. Conectar el cable de audio con los plug de 3.5mm entre la salida de audio de la computadora y la entrada del jack.
5. Abrir el *sketch* [C\\_adquisicion\\_analogica.ino](#) y cargarlo en el Arduino.
6. Abrir el “Graficador del puerto serie” en el IDE de Arduino (Herramienta, Serial Plotter).
7. Ir al generador de señales online (<http://onlinetonegenerator.com?freq=4/>) o usar la aplicación para el móvil (Android) (<https://bit.ly/3C4DyNe>) y establecer los parámetros para generar una señal sinusoidal de 4Hz. Se puede variar la amplitud de la señal modificando el volumen en la página web del generador de señales o de la computadora o móvil.

***¿Qué ves en el Serial Plotter? ¿Por qué crees que la señal se ve así?***

## Rango de tensión del conversor A/D

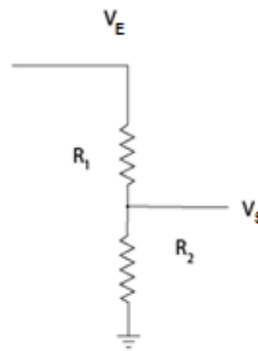
Nuestra señal se ve muy pequeña y además solo vemos una parte de la misma, los semiciclos positivos.

Al registrar una señal es importante conocer la excursión máxima de la misma, es decir el rango de voltaje en el que varía. Esto es importante para evitar introducir una señal de un voltaje mayor al que nuestro dispositivo puede adquirir, y también para asegurarnos que podremos digitalizarla en todo su rango. El Arduino puede adquirir señales analógicas que varían entre 0 y AREF (en nuestro caso 5V), sin embargo, la señal que estamos introduciendo tiene un rango que varía entre voltajes por debajo y por encima de 0V. Por tal motivo, la parte negativa de la misma no puede ser “vista” por el Arduino.

Aún no vimos cómo amplificar una señal, así que no podemos “hacerla más grande”, pero sí podemos modificar el rango de tensión que emplea el conversor A/D para convertir la señal analógica y “verla” más grande. Es decir, podemos cambiar la resolución de tensión del conversor del Arduino.

***¿Qué crees que significa esto?***

Vamos a usar un valor de referencia de 1V para el conversor A/D. ¿Pero, y de dónde sacamos esa tensión? Podemos sacarla a partir de los 5V del riel de alimentación usando un divisor resistivo.

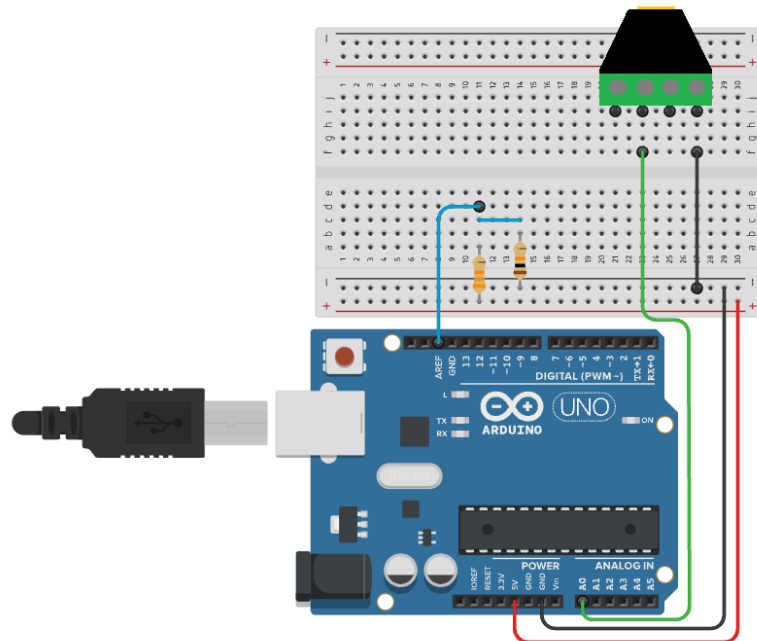


Un divisor resistivo es un arreglo de dos resistencias en serie que nos permite tener en el punto medio una fracción del voltaje aplicado a los extremos del arreglo. La tensión a la salida (punto medio) va a ser proporcional al valor de las resistencias y será:

$$V_S = V_E * \frac{R_1}{R_1 + R_2}$$

En este caso,  $V_E$  es el riel de alimentación y  $V_S$  es el pin R del jack. Entonces:

8. Conectar dos resistencias  $R_1 = 33K$  y  $R_2 = 10K$  en serie entre los puntos de 5V y GND del riel de alimentación (se corresponden con  $V_E$  y  $\text{GND}$  en el esquema del divisor resistivo).
9. Conectar un cable macho-macho entre el punto medio del divisor resistivo y el pin AREF del Arduino (correspondiente con  $V_S$  en el esquema de del divisor resistivo).





10. Modificar el sketch de Arduino en la línea donde define el tipo de referencia de entrada que va a usar el conversor A/D. En vez de decir `analogReference(DEFAULT)`, deberá decir `analogReference(EXTERNAL)`;

***¿Cómo se ve la señal ahora en el Serial Plotter? ¿Qué es lo que se modificó y qué consecuencias tiene para una señal analógica de entrada de mayor amplitud?***

***Extra: podés verificar con el multímetro que la tensión entre el punto medio del divisor resistivo y la tierra esté cerca de aproximadamente 1V.***

## Frecuencia de muestreo

Para poder digitalizar la señal analógica, el microcontrolador del Arduino primero tiene que muestrear la señal. Esto es, tomar muestras de tensión de la señal en diferentes puntos de la misma. Estos puntos de muestreo se realizan cada cierto intervalo de tiempo y esto es lo que denominamos frecuencia de muestreo. En el caso del sketch que venimos usando, la frecuencia de muestreo es de 100 Hz.

11. Modificar la frecuencia de la señal sinusoidal del generador de señales a 50Hz y observar qué pasa con la señal en el serial plotter.
12. Incrementar la frecuencia nuevamente, ahora a 60 Hz.

***¿Qué sucede con la señal en el serial plotter? ¿Por qué crees que sucede esto?***

***Pista: considerá qué frecuencia de muestreo estás utilizando.***

Para poder reconstruir con exactitud la forma de onda de una señal digitalizada, la frecuencia de muestreo utilizada tiene que ser superior a la máxima frecuencia presente en la señal analógica original. A esta frecuencia crítica se la llama frecuencia de Nyquist y se trata de la máxima frecuencia de la señal muestreada que es posible representar sin alteraciones.

En resumen, la frecuencia de muestreo ( $f_m$ ) tiene que ser:

$$f_m \geq 2 * f_{max}$$

Cuando no se cumple este criterio, es decir, cuando la frecuencia de muestreo es menor que la frecuencia de *Nyquist*, ésta no puede ser correctamente reconstruida. En este caso se produce un fenómeno que se denomina *aliasing*.

En esta guía aprendimos cómo usar el Arduino, haciendo uso de los puertos digitales y analógicos. Hicimos uso de conceptos como circuito en serie, resistencias *pull down*, ley de Ohm y divisor resistivo, entre otros. Además, aprendimos conceptos muy útiles al



momento de adquirir una señal analógica, como la resolución y la frecuencia de muestreo. Todas estas herramientas van a ser útiles para aprender cómo funciona un amplificador y posteriormente... ¡adquirir una señal biológica!



Este contenido fue creado por el Ing. Marcos Coletti para el Taller práctico abierto IBRO-LARC: Adquisición de señales neuronales, en colaboración con la Lic. Cecilia Herbert, la Lic. Azul Silva y la Dra. Verónica de la Fuente.

Este contenido está bajo licencia [Atribución/Reconocimiento-CompartirIgual 4.0 Internacional \(CC BY-SA 4.0\)](https://creativecommons.org/licenses/by-sa/4.0/). Esto significa que:

### Sos libre de...

- **Compartir** — copiar y redistribuir el material en cualquier medio o formato
- **Adaptar** — remezclar, transformar y construir a partir del material para cualquier propósito, incluso comercialmente.

### Bajo los siguientes términos:

- **Atribución** — Debés dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Podés hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que vos o tu uso tienen el apoyo de la licenciante.
- **CompartirIgual** — Si remezclás, transformás o creás a partir del material, debés distribuir tu contribución bajo la misma licencia del original.
- **No hay restricciones adicionales** — No podés aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier uso permitido por la licencia.

Este resumen legible por humanos no es un sustituto de [la licencia](https://creativecommons.org/licenses/by-sa/4.0/). Para ver una copia de esta licencia, visitá el siguiente vínculo: <https://creativecommons.org/licenses/by-sa/4.0/>.

Nos apoyan generosamente:

