TP: entrada analógica en Arduino

Aquí continuaremos trabajando con la plataforma Arduino, en particular nos volcaremos a elementos que servirán (y mucho!) a nuestros proyectos: Las entradas analógicas.

Digital vs. analógico:

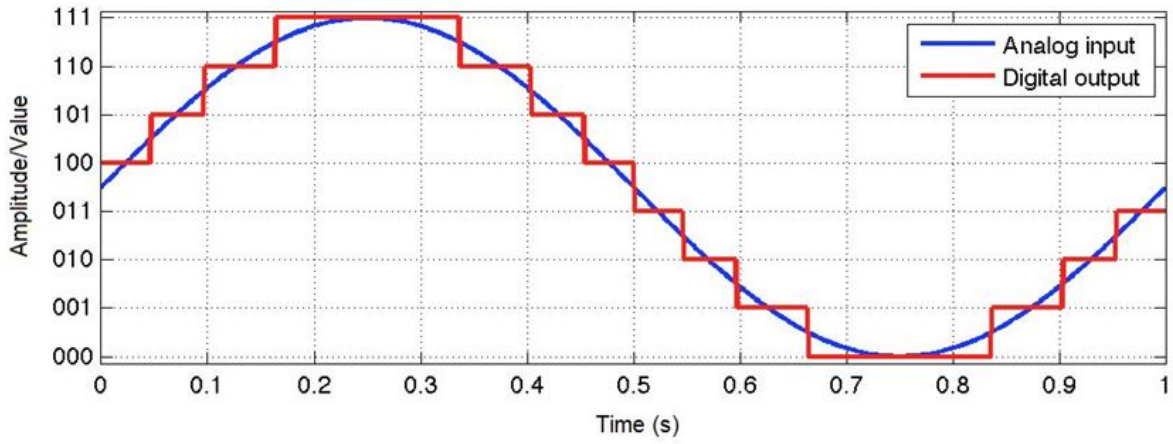
Muchas veces hemos escuchado hablar de señales o valores digitales y analógicos para representar magnitudes físicas, como por ejemplo la temperatura. Ahora bien, si nos preguntamos si la temperatura es una variable física digital o analógica, puede que no nos sea tan fácil distinguirlo en lo inmediato.

Si pensamos un poco como medimos esta variable podríamos decir que es digital, ya que usamos termómetros digitales para medirla. Pero hace un tiempo no muy lejano se usaban termómetros de mercurio los cuales eran analógicos. Entonces esto mete más ruido en nuestra duda si esta variable es digital o analógica.

La realidad es que la temperatura es una variable analógica, ahora como nos damos cuenta de esto. Si tomamos dos valores de temperatura cualquiera por ejemplo 20°C y 30°C, y buscamos un valor entre ellos existen varios por ejemplo el 21°C. Si nuevamente volvemos a buscar un valor entre el 20°C y el 21°C encontraremos un valor por ejemplo el 21,5°C y si continuamos así podremos encontrar infinitos valores. Bueno esto es la naturaleza de una variable analógica, esta esta representada por valores continuos.

Termómetro digital y uno de mercurio

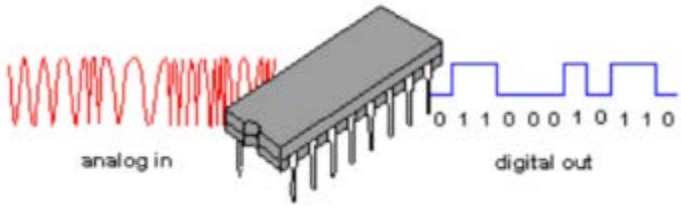
Bien volviendo a la idea de los termómetros, en particular el digital esto se cumple parcialmente, suponiendo un termómetro económico tendrá a lo sumo dos dígitos luego de la coma, por ejemplo, con una temperatura de 35,28°C. ¿Y? Bueno este termómetro solo podrá representar como valores continuos a este valor 35,27°C por debajo y 35,29°C. Este termómetro tiene una representación digital de la temperatura, por lo tanto discretizada (limitada a) una cierta cantidad de valores. Esto implica que la lectura del termómetro puede no ser exacta ya que si la misma es 35,284°C estaría indicando de menos (esto es lo que se conoce como error de discretización).



Ejemplo de una misma señal en forma analógica y en forma discreta

Muchas veces se asocia a que algo digital es algo representado en forma binaria es decir con números 1 y 0. Esto efectivamente es algo digital (solo hay dos estados posibles), pero como vimos con el caso del termómetro digital no es la única forma de tener una representación digital (solo necesitamos limitar la cantidad de posibles casos).

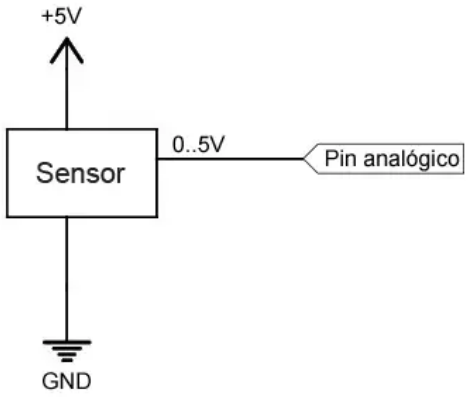
Leyendo una señal analógica desde un microcontrolador: El ADC



La mayor parte de las variables físicas de nuestro ambiente son analógicas (temperatura, humedad, distancia, etc.), pero nuestras computadoras y en particular nuestro microcontrolador (Arduino) es digital. Entonces como podemos conectar estos dos mundos. Para esto en el caso particular que nos interese medir alguna variable analógica con nuestro microcontrolador existen dentro de él, un modulo llamado **Conversor analógico digital** (conocido como CAD, o mas comúnmente por sus siglas en ingles como Analog Digital Converter -ADC).

Este dispositivo lo que realizará será tomar un rango de tensión, en general el que trabaja nuestro Arduino es 0 – 5V (ver que es una variable analógica), y la convertirá a un número digital que podremos trabajar con la programación.

Como nuestro Arduino trabaja con valores de tensión, cualquier otra variable que deseemos medir la convertiremos a tensión, esto es básicamente el trabajo de los sensores.



Los ADC se diferencian por la cantidad de bits que tienen, por ejemplo 6, 8 10, etc. Estos nos dicen en cuantas partes pueden dividir el rango de entrada de la señal analógica. Por ejemplo, para un conversor de 10 bits que es el caso del Arduino tendremos 1024 niveles que podremos detectar. La fórmula genérica para saber el número de niveles es, donde n es el número de bits:

$$niveles = 2^n$$

Ahora bien, en el caso más simple el número de niveles se mapea directamente a los valores leídos de la variable analógica. Por ejemplo, si la variable analógica vale 0 V, el valor digital será el 0. Y si la variable es 5V el valor digital será 1023 (¿pensar, que paso con el 1024?). ¿Y si tenemos algún número intermedio? Bueno esto es una relación lineal por lo que podremos aplicar la regla de tres simples:

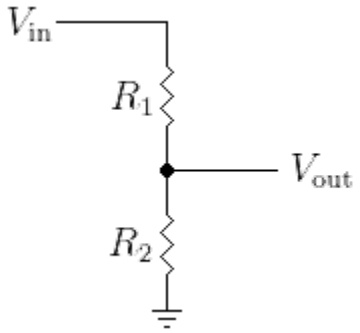
Siendo X el valor leído de la variable analógica e Y el valor leído en el Arduino:

$$Y = \frac{X * 1023}{5}$$

¿Y cómo hacemos esta lectura en el código de Arduino? Para esto utilizaremos una función llamada **analogRead()**. Esta función podrá leer valores analógicos de los pines analógicos de nuestro Arduino, aquelles indicados con una letra **A (A0,A1,etc.)**

Ejemplo lectura del potenciómetro

En este ejemplo leeremos la tensión de un pin analógico de nuestro Arduino, el cual estará conectado a un circuito conocido como divisor resistivo (ver dibujo).

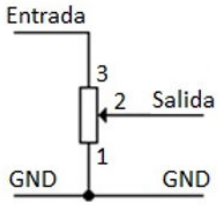


El divisor resistivo es un circuito que tiene dos resistencias en serie (R1 y R2) y la carga o elemento de interés se conecta en el lugar de la segunda resistencia. La serie de dos resistencias baja la tensión vista por la segunda de acuerdo con la siguiente formula (no es más que la aplicación de la ley de Ohm 😊):

$$V_{out} = \left(\frac{R_2}{R_1 + R_2}\right) V_{in}$$

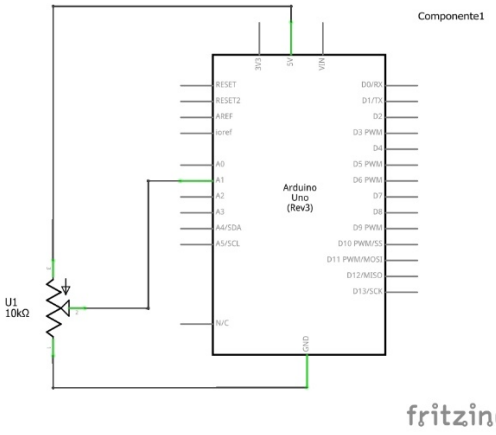
El divisor resistivo es el circuito mas simple para adaptar niveles de tension.

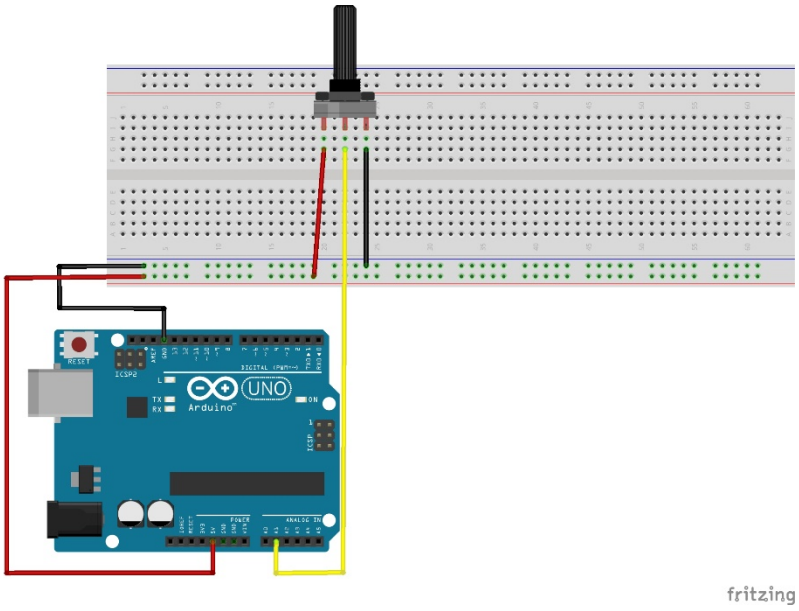
Volviendo a nuestro ejemplo en lugar de R1 y R2 utilizaremos un potenciómetro el cual es una resistencia variable:



Este componente tiene 3 patas, las de los extremos conectan la resistencia total del potenciómetro y la del medio será un curso móvil que podrá variar el valor de su resistencia entre cero y el valor máximo del potenciómetro.

Montaremos el siguiente circuito:





Programa:

```
// Ejemplo: Lectura analogica
int lectura=0;
void setup()
{
Serial.begin(9600);  //habilito comunicacion serie Arduino - PC
}
void loop()
{
lectura=analogRead(A1);  //leo valor del pin analogico
Serial.println(lectura); // envio a la PC el valor leido para verlo por puerto serie
delay(100); //delay para no saturar el puerto serie
}
```

Explicación: El objetivo del programa es ver en la PC el valor analógico de la señal convertido a digital por nuestro Arduino.

Si analizamos el código lo primero que encontramos es la definición de una variable de tipo int (solo guardara números enteros), que la llamamos lectura. Esta variable la usaremos para guardar el dato leído por el pin analógico.

Luego dentro del setup, encontramos la instrucción **Serial.begin(9600);** esta habilita la comunicación entre la PC y el Arduino, el 9600 es la velocidad de la comunicación medida en Baudios (no entraremos en mas detalles aquí, pero es suficiente con saber que esto es una comunicación de tipo serie que se hace atreves de la conexión USB entre la pc y el Arduino), para nuestro caso particular nos servirá para poder ver datos envidos desde el Arduino a la PC.

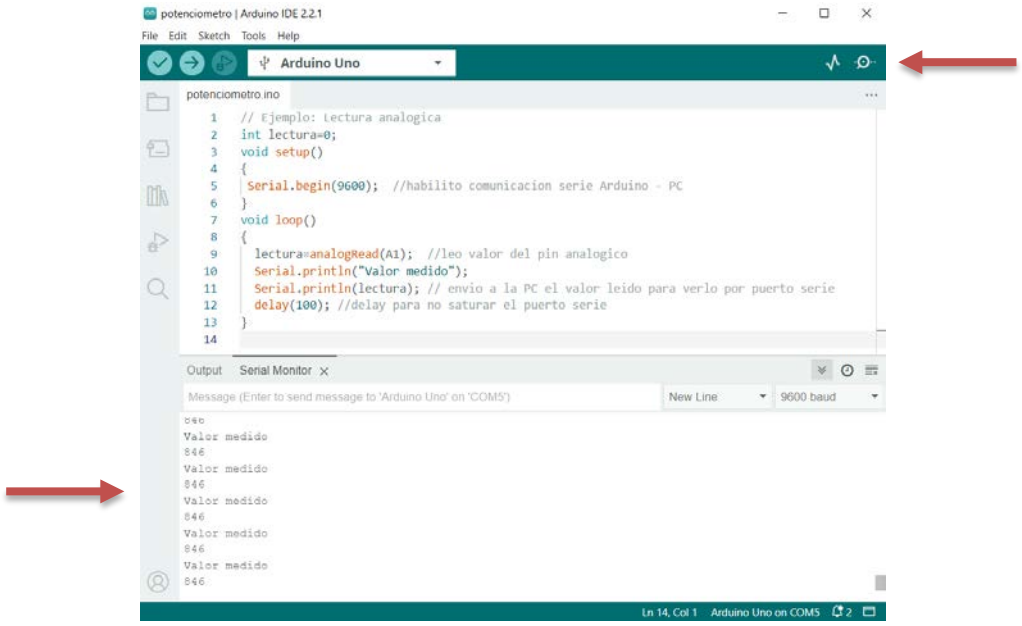
Luego nos encontramos con el bucle principal (el loop), en este lo primero que encontramos es la función **analogRead(A1)**, esta específicamente realiza una lectura analógica sobre la patita A1 (podría haber sido cualquier otra como A2, A3, etc.) y convierte el valor leído a digital para en nuestro caso guardarlo en la variable lectura.

Finalmente encontramos otra función nueva: **Serial.println(lectura);** esta enviara a la PC el dato que este entre paréntesis, aquí la variable lectura.

Serial.printl sirve tambien para enviar texto al PC siempre que el mismo este encerrado por comillas dobles, por ejemplo :

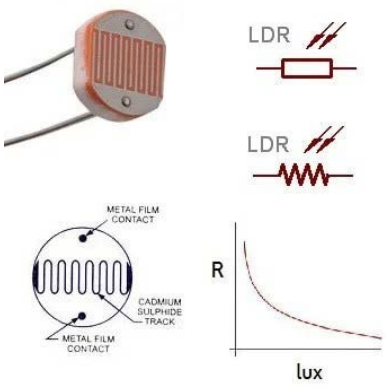
Serial.println("Hola mundo");

Probando: Para ver el funcionamiento de este programa una vez que lo hayamos descargado en el Arduino, abriremos el monitor serial (botón en forma de lupa arriba a la derecha). Este abre una consola en la parte inferior donde chequearemos que la velocidad de comunicación sea la deseada (para nuestro caso 9600) y luego podremos ver los datos recibidos. Si variamos el potenciómetro veremos que los números mostrados van del 0 al 1023 como esperábamos. ¿Que valor deberíamos tener justo a la mitad del recorrido del potenciómetro?



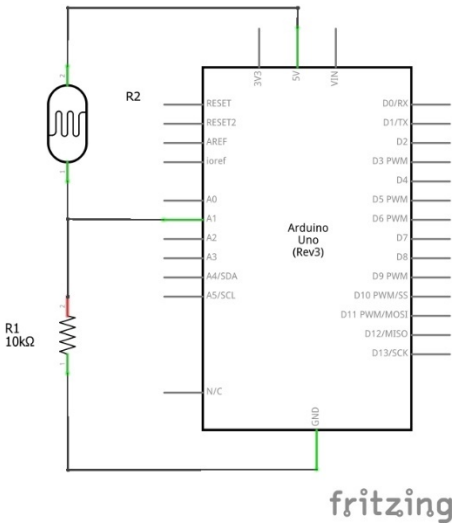
Actividad: Usando un LDR

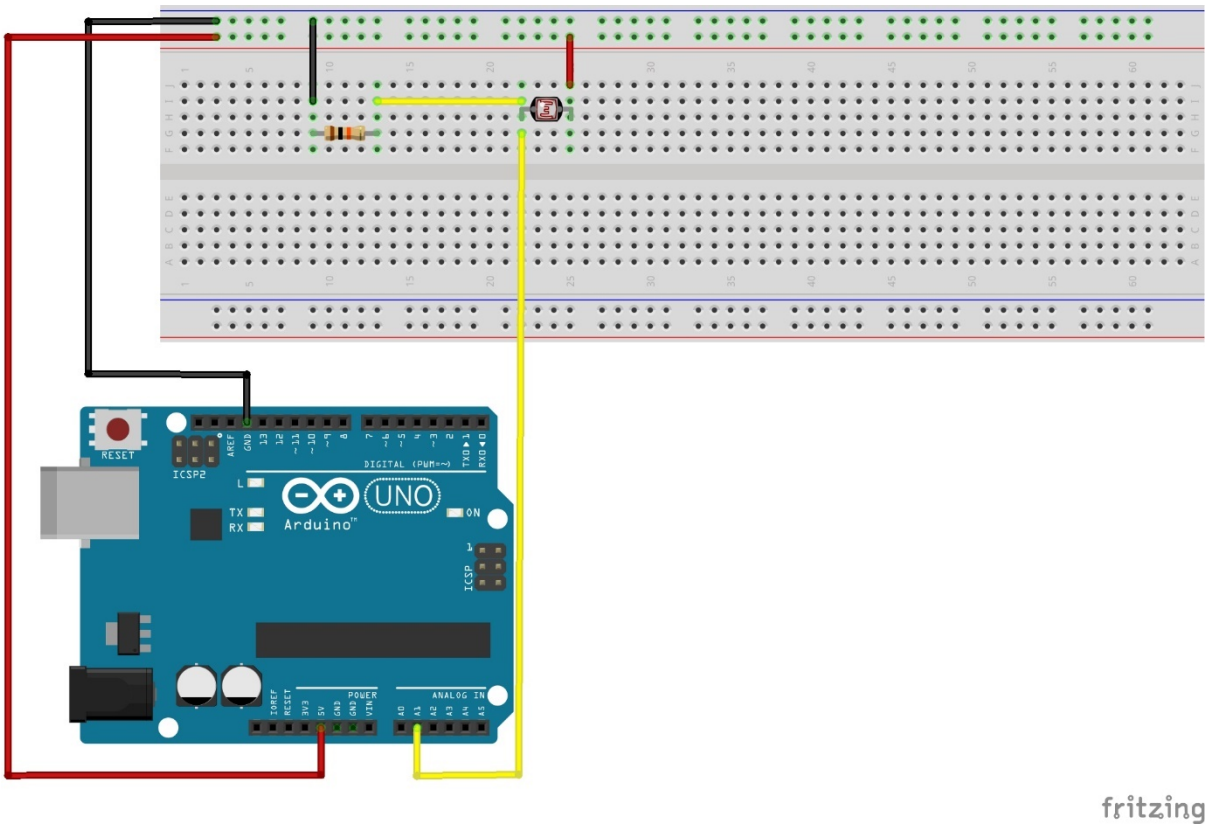
Un LDR (light-dependent resistor) o fotorresistor es un sensor resistivo que varía su resistencia al cambiar la cantidad de luz que incide sobre el mismo, a mayor luz menor resistencia.



Componte LDR, simbología, partes y curva de resistencia contra cantidad de luz recibida.

Podemos usar este sensor para medir con nuestro Arduino la cantidad de luz ambiente, lo más simple es conectar uno de estos dispositivos en forma similar a como hicimos con el ejemplo del potenciómetro, como se muestra en el siguiente esquema:





Ahora si para hacer …

1) **Lea el valor de la entrada analógica del LDR con luz ambiente, proyectando sombra sobre el sensor e iluminándolo con una linterna. Registre estos valores.**
2) **Mezclando el punto anterior y lo de clases previas, arme un circuito que active un led cuando no hay luz en el ambiente y que se apague al detectarla. Este es el comportamiento de las fotocédulas que apagan automáticamente las luces de las calles 😊. Ayuda: detectar justo un valor en la lectura del potenciómetro no es fácil, será mejor comparar contra rango de valores (por ejemplo, hay luz si la lectura es mayor un valor dado).**
3) **Monte un circuito donde se conecten a un Arduino 3 leds y una fotorresistencia. Con este circuito simularemos la secuencia de un semáforo que encienda y apague sus luces durante el día, y que por la noche solo quede parpadeando el led central.**
4) **Con el mismo circuito del punto 3, modifique el programa para que los leds se enciendan si:**
   1er led: hay luz ambiente
   2do led: hay una luz mayor a la reconocida como ambiente
   3er led: no hay luz

5) **Uno para calcular: Si tenemos un sensor que nos entrega tensión en el rango de 0 a 12V, no lo podemos conectar directamente a nuestro Arduino. Para poder utilizarlo una solución simple es utilizar un divisor resistivo. Se posee una resistencia de 10KΩ (=10000Ω) como se muestra en el siguiente circuito. Calcular la resistencia R.**