Curso de:

Programación
ARDUINO

(Publico en general)

Nombre:

MakerBolivia

Calle sucre Nº 749 entre 16 de Julio y Antezana, 1er piso Cochabamba – Bolivia, teléfono 4258813, celulares 65486014 - 68450386

PRESENTACION

El presente material didáctico educativo, se ha preparado pensando en la aquellos aficionados a la ciencia, para desarrollar su creatividad y habilidades en la electrónica, mecánica, programación, robótica, domótica, etc. Con este material el aficionado podrá seguir el desarrollo del curso con tranquilidad, concentrándose más en el trabajo práctico.

Este material consta de una parte teórica-práctica donde el alumno tendrá la posibilidad de repasar en casa y reproducir los experimentos desarrollados en clase ya que se cuenta con imágenes virtuales fáciles de reproducir, además, una parte para el desarrollo en clase, en la cual el alumno escribirá sus observaciones y datos hallados en los experimentos en clase.

Confiando y agradeciendo siempre a Dios padre por las bendiciones que derrama en nuestras vidas.

Prof. Alanoca M. Luis Fernando

CONTENIDO ANALÍTICO
CURSO ARDUINO:

(Básico - Intermedio)

INTRODUCCIÓN

Instalación y descripción del Arduino IDE.

BÁSICO

1) Diodo Led.
2) Pulsador.
3) Potenciómetro.
4) LDR.

INTERMEDIO

5) Ultrasónico (Sensor).
6) Infrarojo (Sensor).
7) Motores DC.
8) Servomotores
9) LCD (Display).
10) Keypad (Teclado).

AVANZADO

11) Ultrasónico + Motores DC.
12) Servomotor + Ultrasónico + Motores DC.
13) Módulo de Bluetooth + Android.
14) Relé + Android + Bluetooth + Luces.

Introducción

INSTALACIÓN Y DESCRIPCIÓN DEL ARDUINO IDE

(Integrated development environment) IDE.

¿Qué es el IDE de Arduino?

Es un espacio de desarrollo integrado basado en Processing. Processing da soporte a lenguajes de programación como C, C++, el nuevo C# y Java.

En otras palabras, el IDE de Arduino es un entorno que nos permite comunicar nuestro microcontrolador Arduino con nuestro ordenador y así poder integrarle cualquier programa escrito en C o cualquier lenguaje de programación que derive de C. Ese programa podrá interacturar con cada salida o entrada de nuestro Arduino.

¿Cómo me descargo e instalo el IDE de Arduino?

En primer lugar accederemos al sitio web: www.arduino.cc

Una vez se haya cargado la página debemos navegar por el menú hasta encontrar el apartado *Download*.

Una vez cargada la pestaña de descargas, a la derecha, podemos observar los enlaces de descarga en función de tu equipo. Lo más común para el usuario de Windows es el enlace de **Windows Installer.** Además Arduino nos ofrece un instalador que no necesita de permisos de administrador (**Windows ZIP file for non admin install**) capaz de instalarse en ordenadores de centros públicos por cualquier usuario.

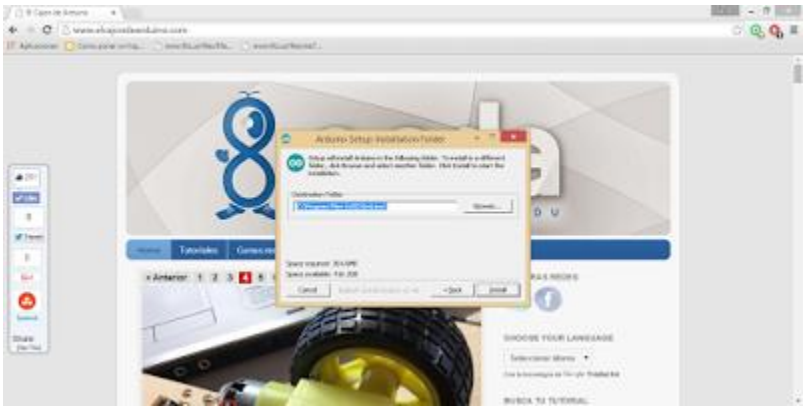
Por último si nuestro caso es que nuestro sistema operativo es **Mac OS X** o **Linux** también disponemos de nuestro instalador.

Nosotros nos centraremos en la instalación del primer enlace comentado anteriormente.

Una vez se haya descargado el IDE de Arduino debemos ejecutar el archivo .exe. *Nota: Probablemente para iniciar el instalador habrá que aceptar que el programa quiera hacer cambios en nuestro ordenador.*

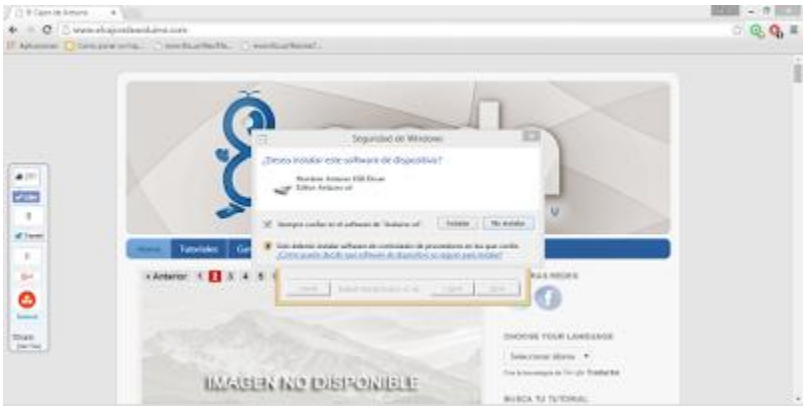
Cuando empiece el asistente de instalación tendremos que aceptar los términos y condiciones del software. Después de leeros (o no) el texto pulsaremos en **I Agree**.

Por último nos quedará establecer la ruta de instalación.



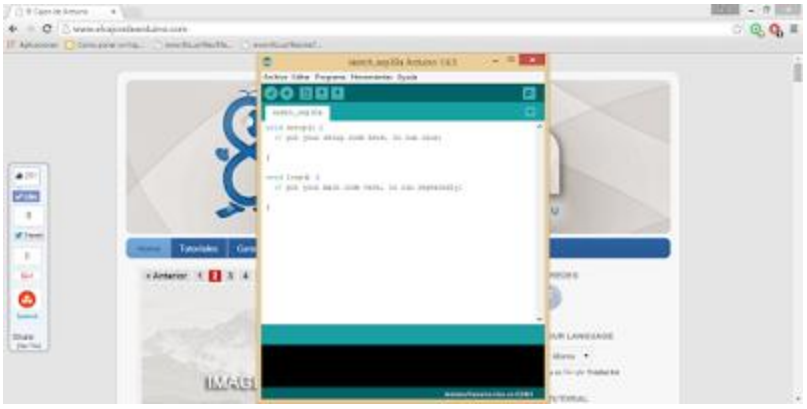
Cuando hayamos seleccionado dónde lo queremos instalar el asistente comenzará su instalación. Habitualmente, los programas por defecto suelen instalarse en el directorio "Archivos de programa" o "Program Files" que es dónde estarán instaladas el resto de aplicaciones de tu ordenador.

Por último el software querrá instalar un controlador para que los puertos USB de tu ordenador reconozcan que has conectado un Arduino.



Cuando éste termine tendremos instalado de manera satisfactoria nuestro IDE de Arduino.

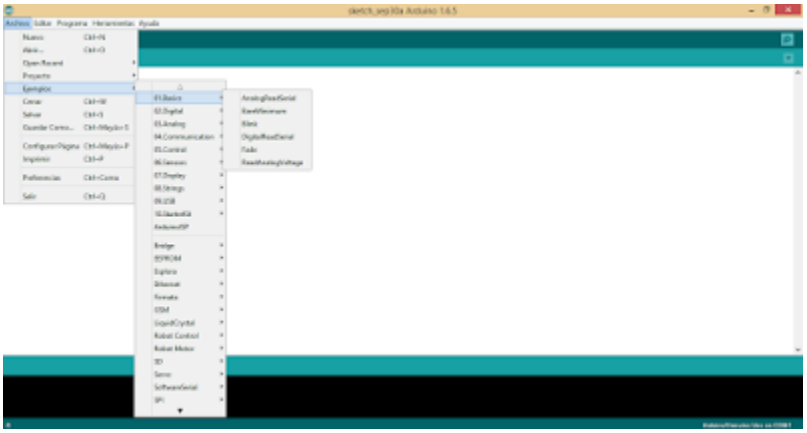
Una vez salgamos del instalador abriremos la aplicación "Arduino" la cual abrirá nuestro espacio de trabajo.



En nuestra pantalla principal encontramos algunas cosas significativas:

Menú de Archivo: En el menú de archivo podemos encontrar las opciones de Nuevo, Abrir…, Abrir reciente, Guardar Como… En definitiva lo que usualmente reside en el menú archivo de cualquier software en base Windows. La diferencia radica en los

apartados Proyectos y Ejemplos. El primero nos dará acceso a proyectos completos desarrollados con el IDE. Creados por nosotros dónde no sólo tendremos un programa si no que podemos tener unos programas asociados a otros. La pestaña de Ejemplos es bastante útil ya que disponemos de un montón de micro proyectos con anotaciones que nos ayudará a avanzar por nuestra cuenta.



Título del documento: En el entorno de Arduino a los programas se les llama Sketch y por defecto nuestro IDE nos asignará un nombre a nuestro programa bajo la premisa Sketch.

Lupa: Este botón se encuentra arriba a la derecha. El monitor Serial (la lupa) es un botón que nos abrirá un cuadro di diálogo para interactuar con nuestro Arduino. En él seremos capaces de enviar y recibir información e incluso trabajar con la información que recibimos a posterior.

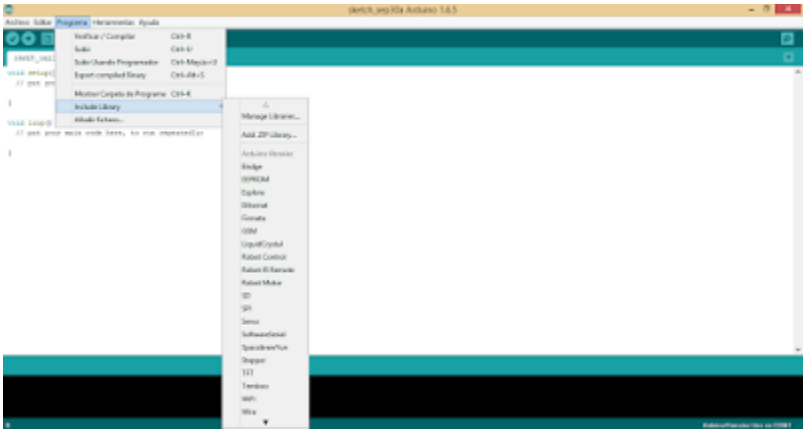
Check: El botón lo podremos encontrar a la izquierda. Este botón nos verificará el programa en busca de errores sintácticos, variables mal declaradas u otro tipo de errores para

evitar que compilemos un programa que falla y podamos dañar nuestro controlador.

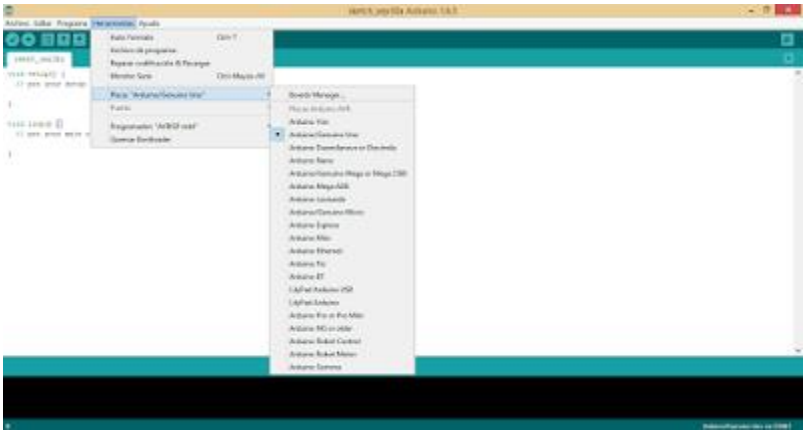
La flecha: Se encuentra al lado del Check. Ese es el botón de compilar el que se encargará de comprimir la información una vez verificada y transformarla a lenguaje máquina para que lo entienda nuestro controlador para finalmente transferirlo al Arduino.

Pie de ventana: En el pie a la derecha podemos observar que aparece "Arduino/Genuino UNO on COM1". Este mensaje nos indicará en todo momento qué tipo de microcontrolador espera el IDE que le conectes y en qué puerto USB hay que conectarlo. Habitualmente el ordenador asignará el puerto de forma automática y el IDE lo reconocerá sin hacer ninguna configuración.

En la pestaña Programa tenemos una pestaña fundamental para el trabajo con Arduino. El IDE contiene un montón de librerías para ahorrarnos tiempo de programación. Estas librerías harán la parte fea de la programación de tu proyecto ya que se encargarán de la interpretación y la comunicación entre nuestro programa y nuestro controlador. Si alguna vez usamos una librería externa para un determinado proyecto disponemos de la opción de añadir la librería en un .ZIP.



En la pestaña Herramientas tenemos otros dos apartados a tener en cuenta. El tipo de placa con el que vamos a trabajar y el puerto (COM) con el que vamos a conectar. Para evitar errores siempre que empecemos un nuevo proyecto debemos seleccionar el controlador con el que vamos a trabajar de la lista. El apartado Puerto lo visitaremos bajo unas condiciones específicas. Si tu Sketch no muestra ningún error de conexión, no tendremos por que pasarnos por aquí pero si nos aparece un error y tiene más puertos USB ocupados puede que nuestro IDE no haya reconocido bien dónde se encuentra nuestro controlador. Entraremos en la pestaña y cambiaremos el puerto COM al que nos ofrezca el IDE.



En principio esto son unas pinceladas básicas de cómo funciona el IDE de Arduino.

Programación

¿QUÉ ES LA PROGRAMACIÓN?

Es el proceso de diseñar, codificar, depurar y mantener el código fuente de programas computacionales. El código fuente es escrito en un lenguaje de programación. El propósito de la programación es crear programas que exhiban un comportamiento deseado. El proceso de escribir código requiere frecuentemente conocimientos en varias áreas distintas, además del dominio del lenguaje a utilizar, algoritmos especializados y lógica formal. Programar no involucra necesariamente otras tareas tales como el análisis y diseño de la aplicación (pero sí el diseño del código), aunque sí suelen estar fusionadas en el desarrollo de pequeñas aplicaciones.

ESTRUCTURAS Y LENGUAJE DE PROGRAMACIÓN EN IDE ARDUINO

Cuando se empieza a programar en un nuevo lenguaje, el primer programa que por convenio de manera no escrita se realiza es el denominado "Hola Mundo", se trata de que en la pantalla del ordenador apareciese **"Hola Mundo"**, pero para realizar esto en arduino sería un poco complicado como primera experiencia.
Nuestro "Hola Mundo" será realizar un programa para conseguir que un led parpadee.
Antes de comenzar es necesario establecer unos conceptos básicos que posteriormente serán ampliados.
En el lenguaje de programación Arduino existe dos funciones básicas que al menos siempre deben estar para que el programa funcione:

```
void setup() {
declaraciones;
}
void loop() {
declaraciones;
}
```

La función **setup** se utiliza para declarar variable, modos de trabajo de los pinMode, inicializar las comunicaciones serie… Se ejecuta solo una vez y aunque no tengamos que realizar nada de lo anterior debe de estar aunque sea vacía.
La función **loop** se ejecuta a continuación y como su propio nombre indica es un bucle que se ejecuta indefinidamente, con lo que posibilita que el programa sea capaz de dar respuesta a los eventos exteriores. Incluye el código de lectura de entradas, activación de salidas, esperas, comunicaciones serie, etc.
Es posible declarar una **función de usuario** para realizar tareas repetitivas y para realizar una programación más ordenar, clara e intuitiva.
Una función es un conjunto de códigos que tiene un nombre y es ejecutada al ser llamada. Este tipo de funciones no son

imprescindibles como las dos anteriores, pero si convenientes para obtener una programación estructurada y clara.
En primer lugar se declara el tipo de la función, que será el valor retornado por la función (int, byte, long, flota, void). A continuación del tipo, se declara el nombre de la función y, entre paréntesis, los parámetros que se pasan a la función. Si la función no devuelve ningún valor entonces se colocara delante la palabra "void".

```
tipo nombreDeLaFuncion(parametro) {
código;}
```

Para definir el principio y el final de un bloque de instrucciones, declaraciones y sentencias, se utiliza las llaves "**{}**". Una llave de apertura "{" siempre debe ir seguida de una llave de cierre "}". Las llaves no balanceadas provocan errores de compilación.
El punto y coma "**;**" se utiliza al final de cada declaración para separar los elementos del programa o también para separar los elementos en un bucle for (ya explicaremos las distintas estructuras de control de flujos). El olvido del punto y coma producirá un error de compilación, además es un error de difícil detección por lo que si obtenéis un error raro, esto es de lo primero que hay que comprobar.
Existen dos tipos de comentarios que se pueden realizar al código:
Comentario a una línea de código y se realiza con una doble barra inclinada **//**

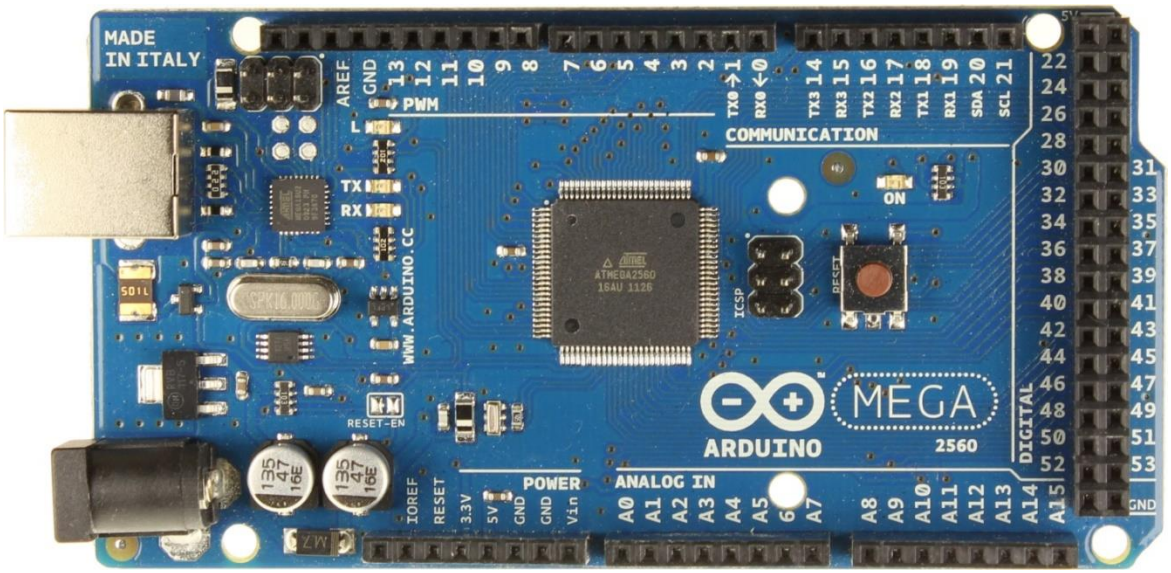
```
código // Comentario
```

Bloque de comentario o comentario multilínea y se realiza empezando con una barra inclinada y un asterisco "**/***", luego el comentario y se termina con un asterisco y una barra inclinada "***/**"

```
Código /* Comentario multilínea
en el que ocupa más de una línea */
código
```

Con estos conceptos básicos que más adelante seguiremos ampliando, ya podemos realizar nuestro primer programa.

TIPOS DE PLACAS PROGRAMABLES (PLATAFORMA ARDUINO)



Arduino MEGA

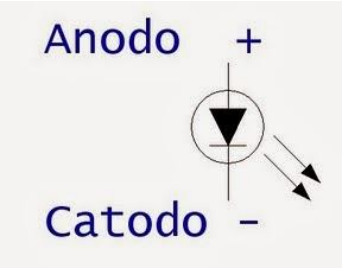
Arduino UNO

Arduino NANO

LABORATORIO 1

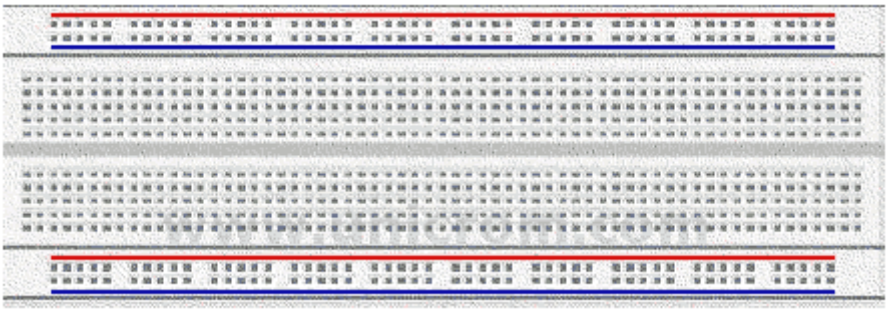
DIODO LED

Un diodo LED es un componente electrónico capaz de transmitir luz. Emite dicha luz cuando la corriente eléctrica lo atraviesa, y lo hace, además, de forma proporcional. A más intensidad de corriente atravesándolo, más luz.
Puede ser conectado ya sea polarización directa como inversa, aunque sólo funcionará si se encuentra conectado de forma directa.



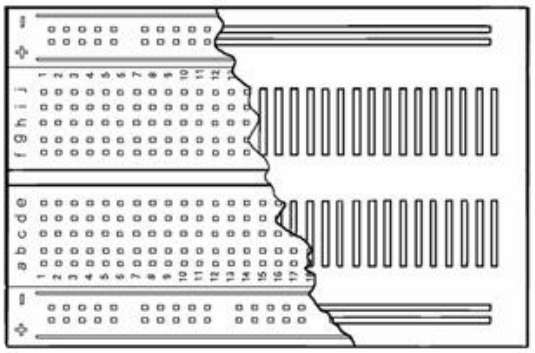
La manera de distinguir el ánodo (terminal positivo) del cátodo (terminal negativo) es fijándonos en la longitud de las patillas del diodo. El ánodo es más largo que el cátodo. Se dibujan las flechas para indicar que se trata de un diodo LED.
Hay que tener en cuenta que, cuando vayamos a conectarlo para diseñar un circuito con nuestro **Arduino UNO**, es necesario conectar una resistencia en serie al diodo, así limitamos la intensidad que circula y evitamos que pueda ser dañado.
¿Qué valor mínimo de resistencia es recomendable?
Según la ley de Ohm, cuanto mayor sea el valor de la resistencia, menos intensidad circulará por el diodo, y menos luz emitirá. La intensidad con la que funciona correctamente un diodo LED es de unos 15mA, por lo que lo podemos resolver conectando una resistencia de 220 ohmios en serie.

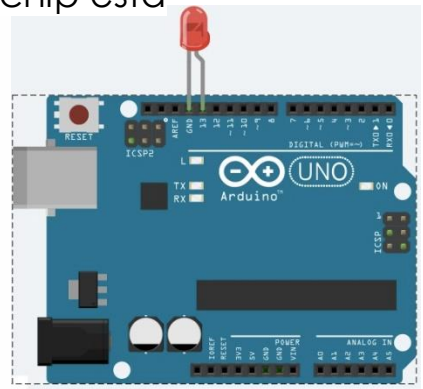
Primero de todo, vamos a ver cómo funciona la placa **breadboard o protoboard.**
Vemos que arriba y abajo posee dos líneas (azul y roja, respectivamente). Estas líneas con agujeritos, se llaman buses. La línea roja indica el bus sometido al voltaje de entrada, y la línea azul será la conectada a tierra. Todos los puntos marcados con la línea roja serán equivalentes porque están conectados entre sí, y lo mismo ocurre con los puntos marcados con la línea azul. Además, buses están aislados eléctricamente uno del otro.



Placa de prototipado (breadboard o protoboard)

Por otro lado, en la parte central aparecen un gran cantidad de agujeros. Se usan para colocar y conectar los componentes. Las conexiones internas están dispuestas de forma vertical, de forma que los agujeros son completamente equivalentes si pertenecen a la misma vertical. Al conjunto de todos los agujeros equivalentes conectados entre sí se les denomina nodo. Finalmente, detectamos una zona central, la que separa la parte superior de la inferior. Se suele utilizar para colocar los circuitos integrados, de forma que la mitad de un chip está aislada de la otra.



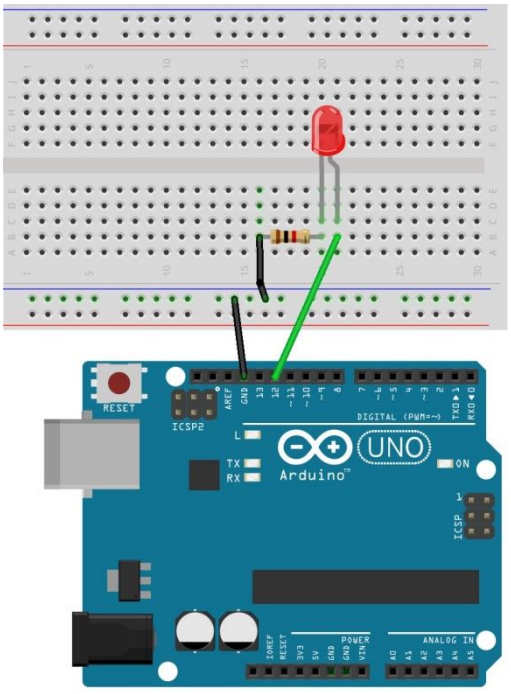


EJEMPLO
Conectar un LED y ponerlo en funcionamiento de forma intermitente.

Materiales:

- Placa protoboard
- 1 Resistencia 220
- LED
- Cables
- Microcontrolador ARDUINO UNO

Puede realizarse de varias formas. La más fácil sin duda es, teniendo en cuenta que el pin 13 ya lleva incorporado la resistencia del valor que necesitamos, conectar el el LED directamente en dicho pin y en la GND (tierra), quedando de la siguiente forma.
Haciéndolo un poquito más complicado y así entender las conexiones, vamos a conectarlo al pin 12. Se puede observar en la imagen siguiente que hemos utilizado una resistencia que se encuentra en serie con el LED. Hay que tener en cuenta que el circuito debe cerrarse, sino no funcionará.
Tal y como hemos explicado antes en cuanto al funcionamiento del LED y de la placa protoboard, entendemos pues el hecho de que aparezca una resistencia conectada en serie al LED. Siguiendo el orden de: cable rojo (normalmente usado para la parte positiva), cable verde, resistencia, LED, cable blanco, cable negro (usado para la parte negativa), tenemos el circuito cerrado, garantizando que, al menos, circule intensidad por él.



El código para este ejemplo (con el pin12) será el siguiente:

Código:

```
int led = 12;
void setup() {
   pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Práctica 1

- Realice un código para que 5 luces de distintos colores se enciendan y se apaguen en ¼ de segundos.

Solución:

- Realice la secuencia de luces, un led se enciende seguido de otro led cada 1/5 de segundos.

Solución:

- Realice la secuencia de un semáforo.

Solución:

LABORATORIO 2

PULSADOR

Partiendo de la definición de interruptor como un dispositivo que cuando se encuentra "cerrado" deja pasar la corriente y cuando está "abierto" impide el paso de la misma, un pulsador no es más que un tipo de interruptor, el cual se mantendrá en posición de "cerrado" tanto tiempo como pulsado lo mantengamos, gracias a la lámina conductora que produce el contacto.

El circuito estará abierto cuando el pulsador no esté presionado, de forma que al apretarlo, se cerrará el circuito, haciendo que pase la corriente a través de él.

De acuerdo con las imágenes, puede parecer que el pulsador tiene 4 terminales, pero lo cierto es que las dos patillas que se encuentran enfrentadas de cada lado están unidas internamente entre sí, lo que supone que en realidad tengamos solamente dos.

Ejemplos

Para controlar un LED mediante un pulsador, no es necesario programar nada, ya que conectamos 5V de entrada con masa (GND), como si estuviera conectado a una pila.

Los cables rojo y negro indican el ánodo y el cátodo, respectivamente, haciendo el comportamiento como el de un circuito eléctrico corriente.

Caso 1

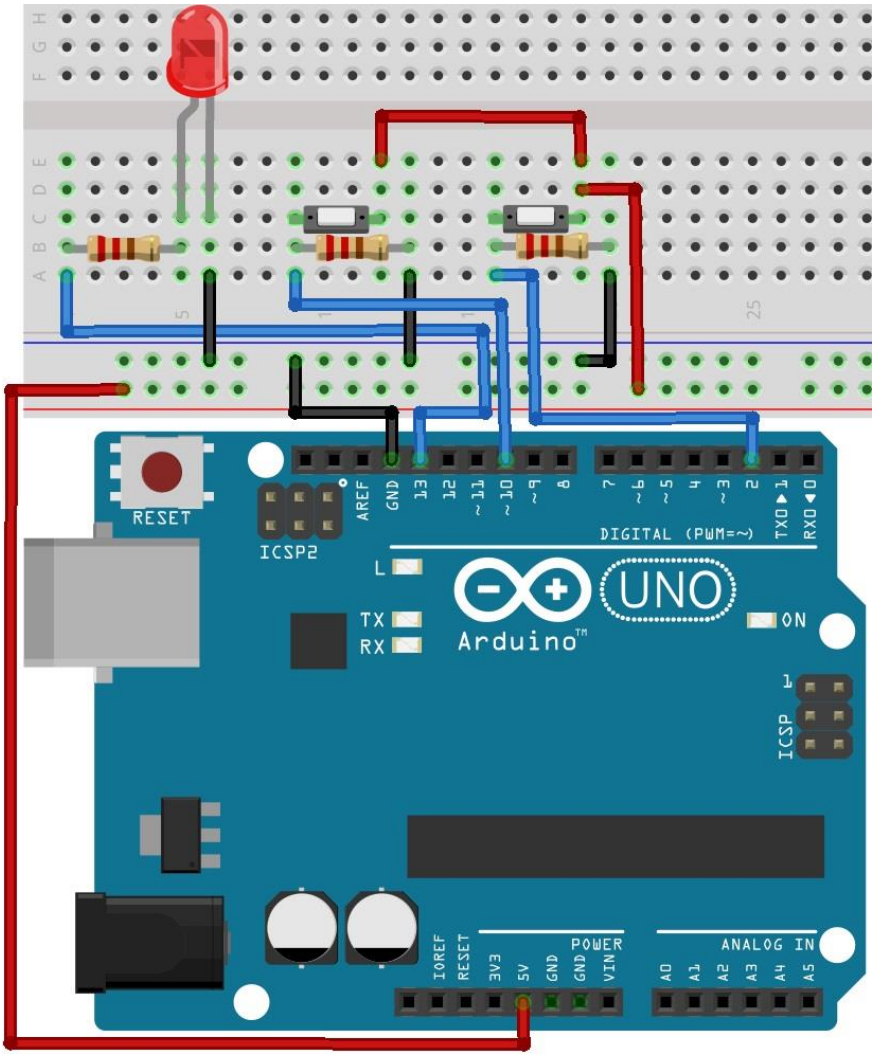
Unimos el terminal de entrada y el de salida de la parte de la izquierda (o de la derecha), de forma que se queda unido

mediante la lámina conductora.
Con esto conseguimos que el circuito esté siempre cerrado (encendiéndose el LED), comportándose el pulsador como un cable, es decir, sin poder actuar de ninguna forma.

Caso 2
Cuando unimos el terminal de la izquierda con el de la derecha (o al revés), independientemente de si sea el de "arriba" o "abajo" (se verá más claramente en el vídeo), entonces el pulsador sí que actúa como un interruptor. Mientras no esté pulsado, el circuito estará abierto y por tanto el LED permanecerá apagado. El momento en el que pulsemos, entonces se cerrará el circuito encendiéndose finalmente el LED.

Control del encendido y apagado de un LED con dos pulsadores
Para poder realizar este circuito, es necesario que dispongamos de 2 resistencias de 1 kΩ cada una, para utilizar en el pulsador.

Vamos a ver cómo sería el montaje:
Primero conectaríamos los dos pulsadores, cada uno de ellos unido a un pin (2 y 10 en nuestro caso). En la misma patilla donde hemos puesto los cables (rojos) colocamos las resistencias unidas a masa (horizontal de arriba de color azul), para evitar que se nos quemen los pulsadores. Y, con el cable negro, conectamos dicha horizontal con la masa de nuestro Arduino (GND).
Unimos ahora el cable de alimentación, que distribuirá la tensión en ambos pulsadores, a la otra patilla que queda libre.
Del Pin de 5V se puede observar en la imagen siguiente cómo une ambos pulsadores (cables naranja).
Finalmente, faltará colocar el LED como hemos explicado en una entrada anterior (Diodo LED. Introducción y ejemplo). Lo haremos mediante el pin 13, quedando finalmente el montaje que se muestra a continuación:



Una vez tenemos el montaje, bastará con introducir un sencillo sketch para hacer que funcione correctamente.

Código:

```
int LED = 13;
int pulsador1 = 2;
int pulsador2 = 10;

void setup () {
  pinMode (LED, OUTPUT);
  pinMode (pulsador1,INPUT);
  pinMode (pulsador2, INPUT);
}
```

```
void loop () {
  if (digitalRead(pulsador1) == HIGH)
  {
    digitalWrite (LED, HIGH);
  }
  else if (digitalRead(pulsador2) == HIGH)
  {
    digitalWrite (LED, LOW);
  }
}
```

Práctica 3

- Realice:
-

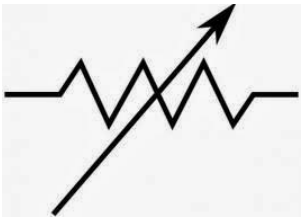
- Realice:

- Realice:

LABORATORIO 3

POTENCIÓMETRO

Un potenciómetro es una resistencia variable, podemos elegir el valor que puede tomar. De esta forma, controlamos la intensidad de corriente que fluye por un circuito si éste está conectado en paralelo, así como la diferencia de potencial si está conectado en serie.
El potenciómetro dispone de tres patillas: entre las dos de sus extremos existe siempre un valor fijo de resistencia, y entre cualquiera de los dos extremos y la patilla central tenemos una parte de ese valor. Es decir, la resistencia máxima que ofrece el potenciómetro entre sus dos extremos no es más que la suma de las resistencias entre un extremo y la patilla central.



Ejemplo
Potenciómetro para encender 5 LEDs
En este ejemplo observaremos que a medida que variamos la resistencia interna del potenciómetro, se irán encendiendo más o menos LEDs.

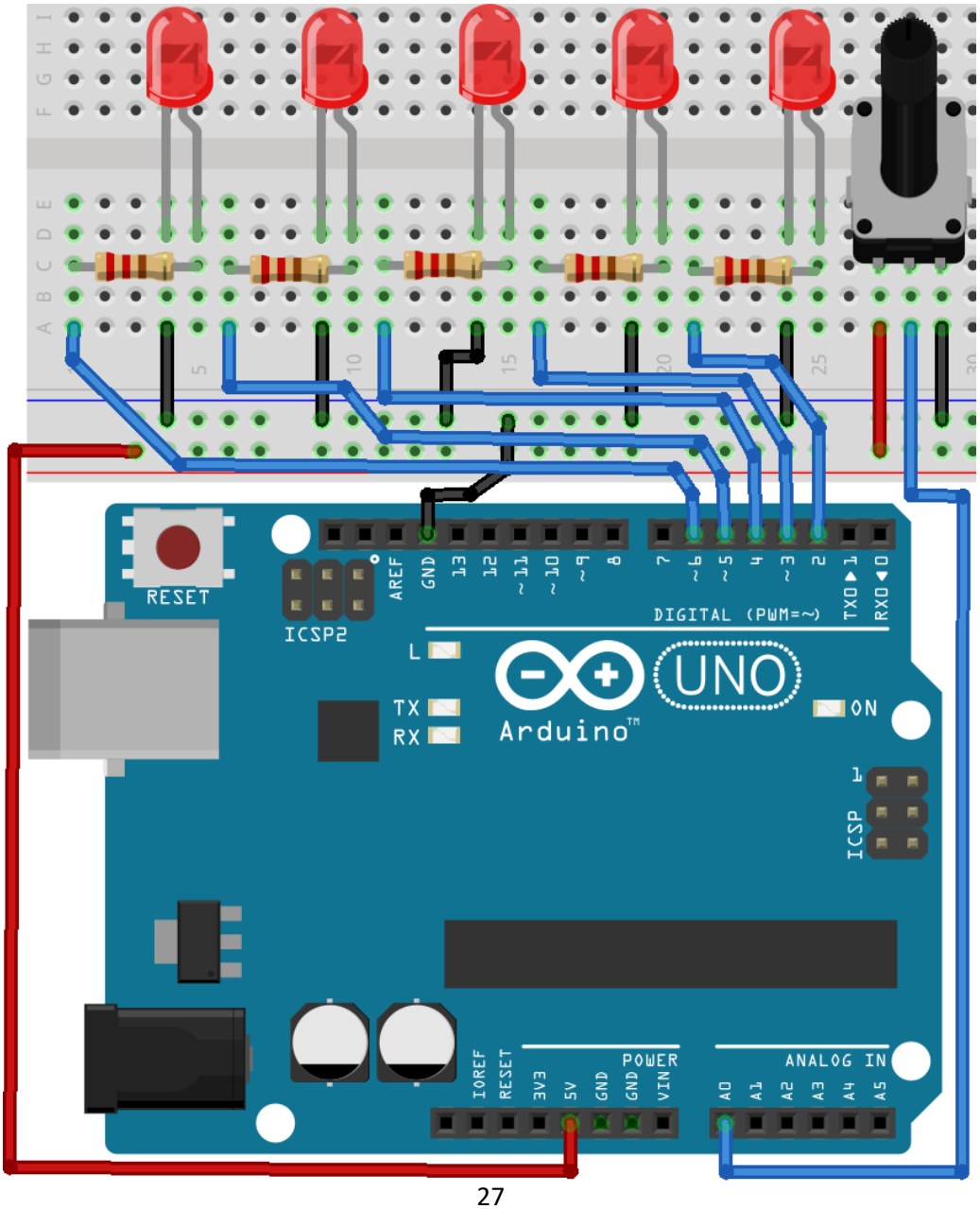
Materiales:

- 5 LEDs (el color no es significativo)
- 5 resistencias de 220Ω (una para cada LED)
- 1 potenciómetro 50kΩ

Primero, procederemos a montar el circuito en la placa protoboard. Antes que nada, conectaremos el potenciómetro: la patilla de la derecha, la conectaremos a la alimentación de 5V (cable rojo), la patilla central la conectaremos al pin analógico A0 (cable amarillo), y finalmente, el cable blanco irá conectado a masa, quedando el siguiente resultado:

conexiones del potenciómetro

Una vez tenemos esta parte hecha, procedemos a conectar los LEDs. Hay que tener en cuenta, como se ha explicado en una entrada anterior (Diodo LED. Introducción y ejemplo), que para cada diodo LED, es necesario conectar una resistencia de 220Ω para evitar que éste pueda ser dañado.
Así, como tenemso 5 LEDs, utilizaremos 5 pines distintos (2, 3, 4, 5 y 6). Seguidamente irá conectada una resistencia (cables azules) y a continuacion el LED. Para terminar, la salida del LED debemos conectarla a masa, para así garantizar que el circuito esté cerrado (cable naranja).
El resultado final sería el siguiente:



Una vez realizado el montaje, pasamos a escribir el código para poder mandarle la información a nuestro arduino.

Código:

```
int leds[]={2,3,4,5,6};
int pot;
int n=0;

void setup()
{
   for(n=0;n<5;n++)
   {
     pinMode(leds[n],OUTPUT);
   }
   Serial.begin(9600);
}

void loop()
{
  pot = analogRead(0);
  Serial.print("Valor del potenciometro: ");
  Serial.print(pot);
  Serial.print("\n");
  delay(2000);

  if(pot >= 0 && pot <= 150)
  {
    for(n=0;n<1;n++)
    {
        digitalWrite(leds[n],HIGH);
        n = 1;
    }
    for(n=1;n<5;n++)
    {
        digitalWrite(leds[n],LOW);
    }
  }
```

```
  if(pot >= 150 && pot <= 300)
  {
    for(n=0;n<2;n++)
    digitalWrite(leds[n],HIGH);
    for(n=2;n<5;n++)
    digitalWrite(leds[n],LOW);
  }
  if(pot >= 300 && pot <= 450)
  {
    for(n=0;n<3;n++)
    digitalWrite(leds[n],HIGH);
    for(n=3;n<5;n++)
    digitalWrite(leds[n],LOW);
  }
  if(pot >= 450 && pot <= 600)
  {
    for(n=0;n<4;n++)
    digitalWrite(leds[n],HIGH);
    for(n=4;n<5;n++)
    digitalWrite(leds[n],LOW);
  }
  if(pot >= 600)
  {
    for(n=0;n<5;n++)
    digitalWrite(leds[n],HIGH);
  }
}
```

Práctica 2

- Realice:

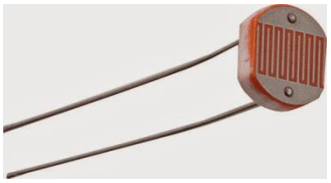
- Realice:

- Realice:

LABORATORIO 4

SENSOR LDR

Antes de empezar con el tutorial, es necesario saber que un sensor LDR es un componente electrónico pasivo cuyo valor de la resistencia varía en función de la luz que recibe. Cuanta más luz reciba, el valor de su resistencia será menor.



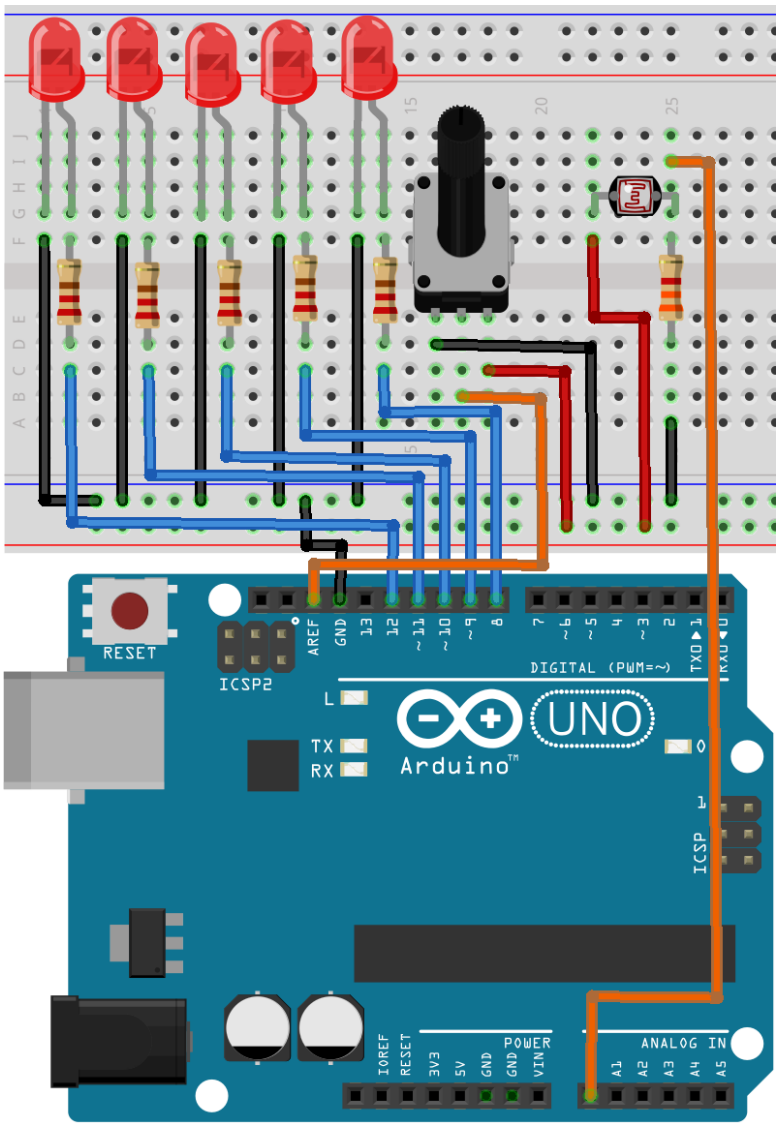
Materiales:

- Placa breadboard.
- 5 Diodos LED.
- 5 resistencias de 220 Ω.
- 1 LDR (resistencia dependiente de luz).
- 1 potenciómetro 50kΩ (uno de 10kΩ también podría ser útil).
- 1 resistencia de 1kΩ.
- Cables.

Procedimiento a realizar:

Se conectarán 5 LED que irán encendiéndose dependiendo del dicho valor de resistencia, ligado inversamente con la cantidad de luz, de forma que conforme vaya disminuyendo la cantidad de luz, se irán encendiendo los LED de forma progresiva.
Un ejemplo típico podría ser la utilización en farolas urbanas que van encendiéndose conforme va anocheciendo.
El circuito distribuido en la protoboard podría tener esta forma:

Vayamos por partes.
Primero concretamos los pines que vamos a utilizar para los LED. Por orden, utilizaremos el 12, 11, 10, 9 y 8. Unidos mediante un cable a la breadboard, colocamos las resistencias de 220 Ω puesto que podríamos quemar los diodos por un exceso de calor.
El potenciómetro, por otro lado, es una resistencia variable que dispone de tres patillas: entre las dos de sus extremos existe siempre un valor fijo de resistencia, y entre cualquiera de esos extremos y la patilla central tenemos una parte de ese valor máximo. Es decir, la resistencia máxima que ofrece el potenciómetro entre sus dos extremos no es más que la suma de las resistencias entre un extremo y la patilla central, y entre la patilla central y el otro extremo.
Lo utilizamos para determinar el valor mínimo de la luz que es capaz de detectar el sensor LDR, de forma que cuando vayamos a ponerlo en marcha debemos regularlo para un correcto funcionamiento. Si ponemos el valor de la referencia muy baja, empezarán a funcionar los LED con menos luz ambiente que si ponemos una referencia elevada.
De forma práctica, conectaremos la patilla central al pin AREF (ofrece un voltaje de referencia externo para poder aumentar la precisión de las entradas analógicas) que luego le diremos al programa que lo vamos a usar como referencia externa (moverlo manualmente según el valor mínimo de luz que queramos detectar). Mientras que cada patilla la conectaremos al ánodo y al cátodo (la patilla de la izquierda al cátodo, y la de la derecha al ánodo) de la breadboard para unirlo con el sensor y los LED.
Para el sensor, la señal que recibe es una señal analógica que obtenemos del exterior para transformarla en digital, por lo que colocaremos el cable de entrada en un pin analógico, A0 en nuestro caso. Al colocar la resistencia de 1kΩ en la parte de arriba del sensor, estamos creando un divisor de tensiones, de manera que cuanta más luz haya, más tensión tendremos a la entrada de nuestra entrada analógica.
Quedando finalmente el montaje de la siguiente manera:



Nos queda escribir el código que le pasaremos al Arduino.

Código:

```
int valorLDR = 0;

int pinLed1 = 12;
int pinLed2 = 11;
int pinLed3 = 10;
int pinLed4 = 9;
int pinLed5 = 8;

int pinLDR = 0;
```

```
void setup()
{
  pinMode(pinLed1, OUTPUT);
  pinMode(pinLed2, OUTPUT);
  pinMode(pinLed3, OUTPUT);
  pinMode(pinLed4, OUTPUT);
  pinMode(pinLed5, OUTPUT);

  analogReference(EXTERNAL);
}
void loop()
{
  valorLDR = analogRead(pinLDR);

  if(valorLDR >= 1023)
  {
    digitalWrite(pinLed1, LOW);
    digitalWrite(pinLed2, LOW);
    digitalWrite(pinLed3, LOW);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
  else if((valorLDR >= 823) & (valorLDR < 1023))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, LOW);
    digitalWrite(pinLed3, LOW);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
  else if((valorLDR >= 623) & (valorLDR < 823))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, LOW);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
```

```
else if((valorLDR >= 423) & (valorLDR < 623))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
  else  if((valorLDR >= 223) & (valorLDR < 423))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, HIGH);
    digitalWrite(pinLed5, LOW);
  }
  else
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, HIGH);
    digitalWrite(pinLed5, HIGH);
  }
}
```

Práctica 4

- Realice:

- Realice :

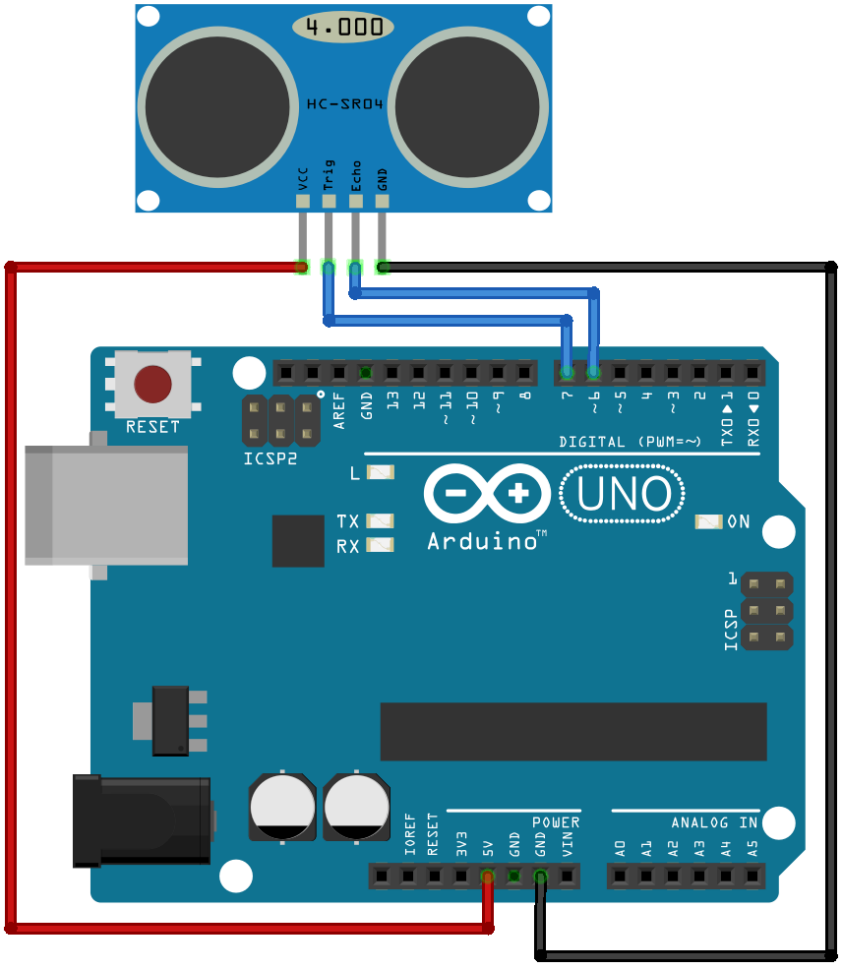
- Realice:

LABORATORIO 5

SENSOR ULTRASONIDOS HC-SR04

El sensor de ultrasonidos se enmarca dentro de los sensores para medir distancias o superar obstáculos, entre otras posibles funciones.
En este caso vamos a utilizarlo para la medición de distancias. Esto lo consigue enviando un ultrasonido (inaudible para el oído humano por su alta frecuencia) a través de uno de la pareja de cilindros que compone el sensor (un transductor) y espera a que dicho sonido rebote sobre un objeto y vuelva, retorno captado por el otro cilindro.
Este sensor en concreto tiene un rango de distancias sensible entre 3cm y 3m con una precisión de 3mm.

¿Qué recibimos en el sensor?
El tiempo que transcurre entre el envío y la recepción del ultrasonido.
¿Cómo vamos a traducir dicho tiempo en distancia?
Aprovechando que la velocidad de dicho ultrasonido en el aire es de valor 340 m/s, o 0,034 cm/microseg (ya que trabajaremos con centímetros y microsegundos). Para calcular la distancia, recordaremos que v=d/t (definición de velocidad: distancia recorrida en un



determinado tiempo).
De la fórmula anterior despejamos d, obteniendo d=v·t, siendo v la constante anteriormente citada y t el valor devuelto por el sensor a la placa Arduino.
También habrá que dividir el resultado entre 2 dado que el tiempo recibido es el tiempo de ida y vuelta.

Material:

- Sensor ultrasonidos HC-SR04 de Electrohobby
- Placa Arduino UNO
- Cables
- Cable USB
- Protoboard

Conexiones:
El sensor consta de 4 pines: "VCC" conectado a la salida de 5V de la placa, "Trig" conectado al pin digital de la placa encargado
de enviar el pulso ultrasónico, "Echo" al pin de entrada digital que
recibirá el eco de dicho pulso y "GND" a tierra.

Código:

```
long distancia;
long tiempo;
void setup(){
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(6, INPUT);
}
void loop(){
  digitalWrite(7,LOW);
  delayMicroseconds(5);
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  tiempo=pulseIn(6, HIGH);
  distancia= int(0.017*tiempo);
```

```
  Serial.println("Distancia ");
  Serial.println(distancia);
  Serial.println(" cm");
  delay(1000);
}
```

Práctica 5

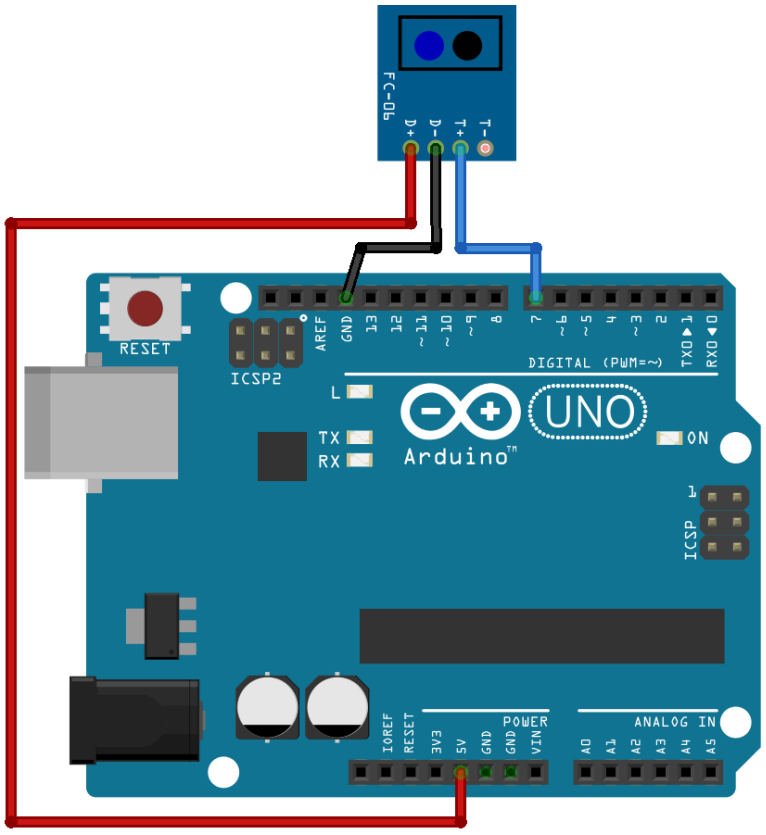
- Realice:

- Realice:

- Realice:

LABORATORIO 6

INFRAROJO



Código:

```
int infrarojo = 7;

void setup ()
{
     Serial.begin(9600);
     pinMode (infrarojo, INPUT);
}

void loop ()
{
  if (digitalRead(infrarojo) == HIGH)
  {
    Serial.println(”Color NEGRO“);
```

```
    delay(1000);
  }
  else
  {
   Serial.println(”Color BLANCO“);
   delay(1000);
  }
}
```

Practica 6

- Realice:

- Realice:

LABORATORIO 7

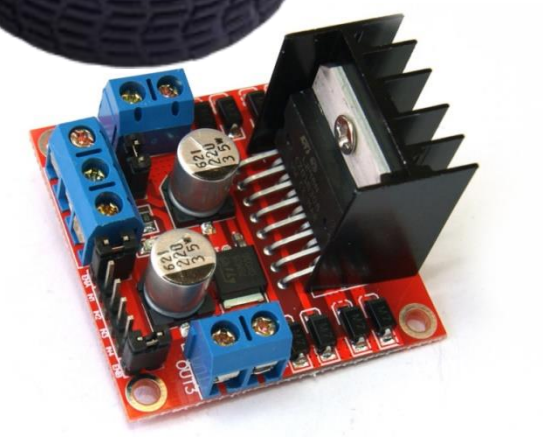
MOTORES DC

Este tutorial es el primero de una serie de tutoriales que tienen por objetivo la realización de un robot tipo coche, controlado por un Arduino UNO.
En este tutorial, se va a hacer una pequeña introducción de la tracción del vehículo, el módulo de potencia necesario y un programa sencillo para testear los motores.

Motores

El vehículo de este tutorial va a consistir en dos motores DC que trabajan desde 3V hasta los 6V. Simplificando, su funcionamiento se basa en aplicarle corriente al motor para que el eje gire. Se puede aplicar corriente en ambos sentidos, lo que provocará que dicho motor gire también en ambos sentidos. Estos motores tienen dos pestañas (bornas) en las que se conectarán los cables. Normalmente hay que soldar los cables para que queden fijos.

Módulo de potencia

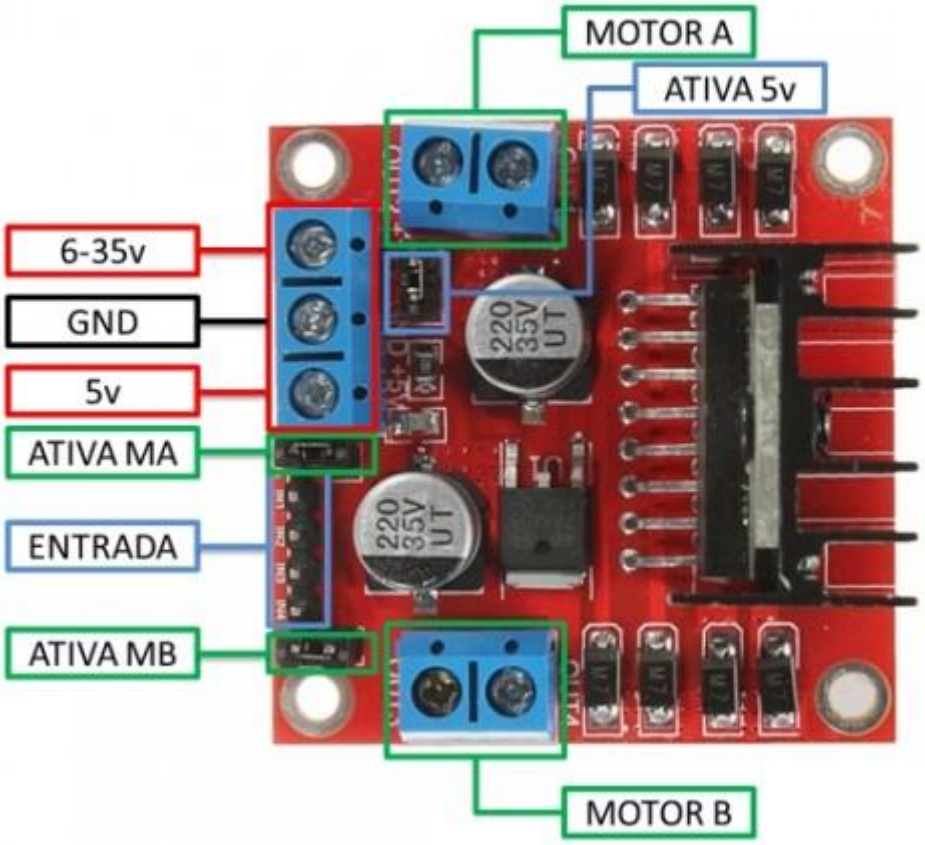


Se necesitará un módulo de potencia, es decir, electrónica extra para hacer funcionar estos motores. ¿Por qué? Porque estos motores tienen un consumo de 250 mA y la

placa Arduino UNO (con la que vamos a trabajar), sólo puede sacar por los pines digitales 40 mA como intensidad máxima.

(Detalle de tornillería de Nylon para levantar el módulo para no dañar la mesa, cortesía de Electrohobby.es)

El módulo elegido es el puente H basado en el encapsulado L298N (fácil de encontrar por "motor driver L298N" en muchas tiendas).



Este dispositivo puede ser alimentado, con el jumper de la salida de 5 V activo (ver imagen abajo), desde 6V hasta 12V para un funcionamiento mínimo de los motores (existe una caída de tensión de hasta 3V dentro del módulo, entonces necesitaremos un mínimo de 6V para poder tener a la salida de los motores los 3V mínimos).

Explicación inputs/outputs del módulo

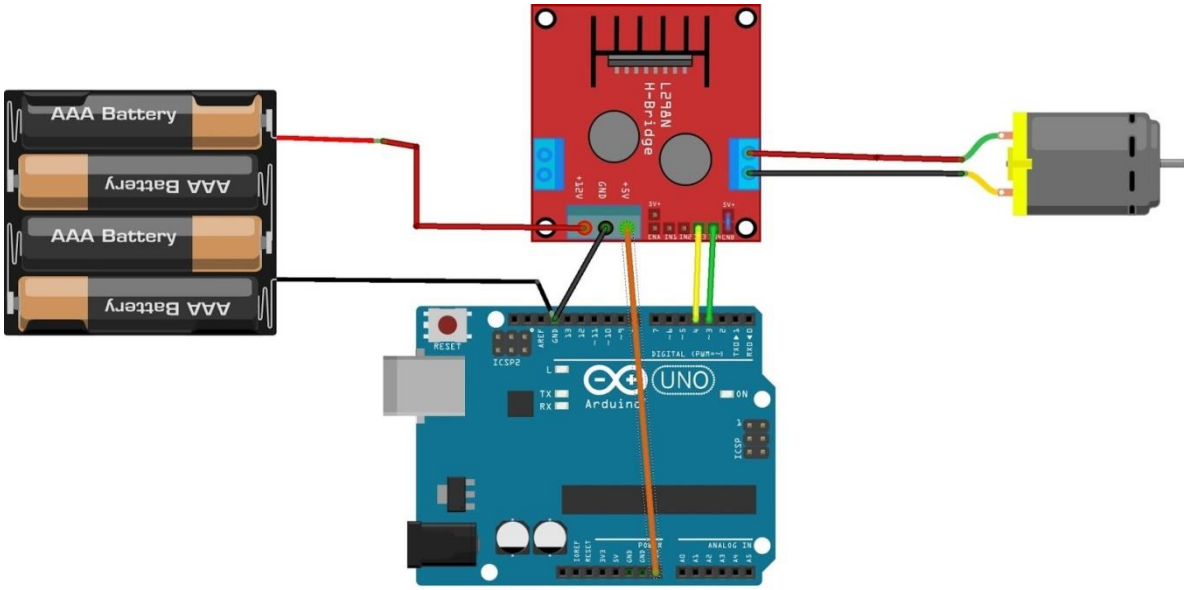
La alimentación del módulo se hace por las entradas de **+12V Input** y **GND**, conectando el primero al positivo de la batería (si es un portapilas, cable rojo ahí) y el negativo al GND (cable negro del portapilas ahí).
La salida de los motores será por las bornas de **Output A** y **Output B**. IMPORTANTE conectar positivo y negativo de forma simétrica de los motores para que los motores giren en la misma dirección.
El módulo tiene una salida de 5V para alimentar la placa Arduino desde el mismo, la salida **+5V Output**. Recordar que se necesita el jumper puesto entre los pines de **Enable 5V output**. Dejando el jumper de los pines de **Enable A** y **Enable B**, activaremos el Output A y el Output B respectivamente.
Los **Digital Input** son para excitar los motores. Son cuatro pines que irán conectados a salidas digitales del Arduino. El IN1 e IN2 para el motor conectado al Output A y el IN3 y el IN4 para el motor conectado en el Output B. Para que el coche vaya hacia delante o hacia atrás, girando los motores de forma correcta, tendremos que activar siempre los IN2+IN4 en HIGH o IN1+IN3 en HIGH (siempre números pares juntos e impares juntos para ir rectos, tanto hacia delante como hacia atrás).

Material:

- Placa Arduino UNO
- 2x motores DC
- Cables
- Modulo puente H L298N
- Soldador y alambre de estaño (en caso de no venir soldados los cables a las bornas de los motores)

Conexión:

El siguiente esquema muestra la conexión de los cables entre los diferentes componentes.



Para este programa, con 6V en el portapilas es suficiente. Para un sólo motor y con el ENABLE B activo y el ENABLE A desactivo (como en la imagen), recomendable no exceder de los 9V.
El siguiente programa sirve para probar los motores y como primera toma de contacto con el módulo L298N.

Código:

```
int IN3 = 4;
int IN4 = 3;

void setup()
{
  pinMode (IN3, OUTPUT);
  pinMode (IN4, OUTPUT);

}
void loop()
{
```

```
  digitalWrite (IN3, HIGH);
  digitalWrite (IN4, LOW);
  delay(5000);

  digitalWrite (IN3, LOW);
  delay(500);

  digitalWrite (IN4, HIGH);
  delay(5000);

  digitalWrite (IN4, LOW);
  delay(500);
}
```

Práctica 7

- Realice:

- Realice:

- Realice:

LABORATORIO 8

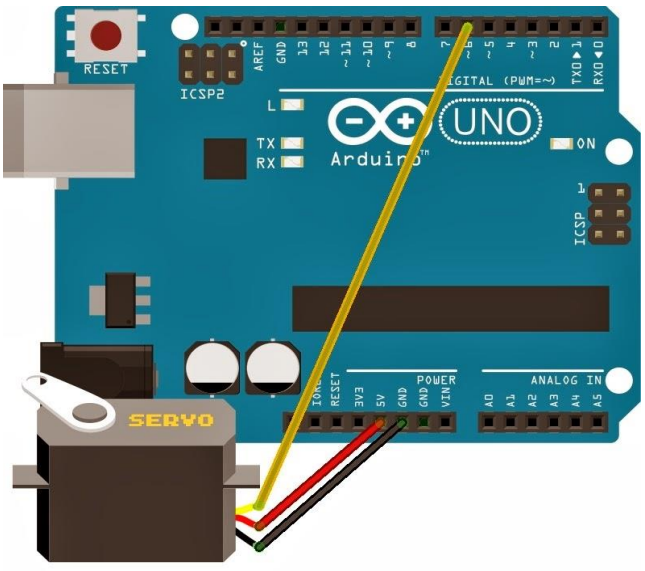
SERVOMOTOR MINI

Un servomotor (también conocido como servo), es un motor de corriente continua compuesto por engranajes que limitan la velocidad. Está limitado, teniendo un rango de movimiento de 0 a 180 grados.

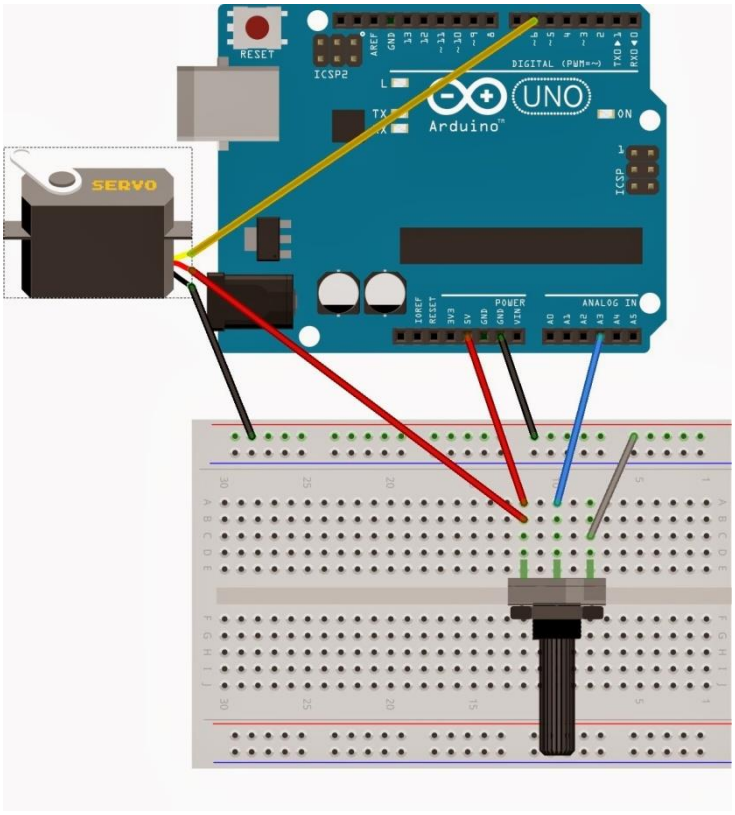
Como podemos observar en la imagen, el servo dispone d tres cables distintos. Cada uno de ellos se conecta de la siguiente manera: el cable de color rojo (normalmente) se utiliza para recibir la corriente eléctrica. Tenemos que tener en cuenta que debe recibir entre 5 y 7V para conseguir que funcione; el cable de color negro o marrón sirve para conectarlo a tierra; finalmente el cable de color amarillo, blanco o naranja es el que va conectado al pin de nuestro Arduino.

Código:

```
#include <Servo.h> //Incluimos la librería para
servos
Servo servo1;
void setup() {
  servo1.attach(6);
}
void loop() {
  servo1.write(0);
  delay(1000);
  servo1.write(45);
  delay(1000);
  servo1.write(90);
  delay(1000);
  servo1.write(135);
  delay(1000);
  servo1.write(180);
  delay(1000);
}
```



A partir del montaje que hemos hecho en el ejemplo anterior y, utilizando una placa breadboard, nos disponemos a realizar un nuevo montaje utilizando un potenciómetro. Recordamos que la patilla central de este componente es la que va conectada al pin que le asignemos. De las otras dos patillas, una será la que se conecta a tierra, y la otra a la alimentación (compartiendo los mismos valores que el servo). Actuando de esta forma, obtenemos el siguiente montaje:



Código:

```
#include <Servo.h>
Servo servo;
int pot=3;
void setup() {
  Serial.begin(9600);
  servo.attach(6);
}
void loop() {
  int valorpot=analogRead(pot);
  int val=map (valorpot,0,1023,0,180);
  Serial.println(val);
  servo.write(val);
  delay(100);

}
```

Práctica 8

- Realice:

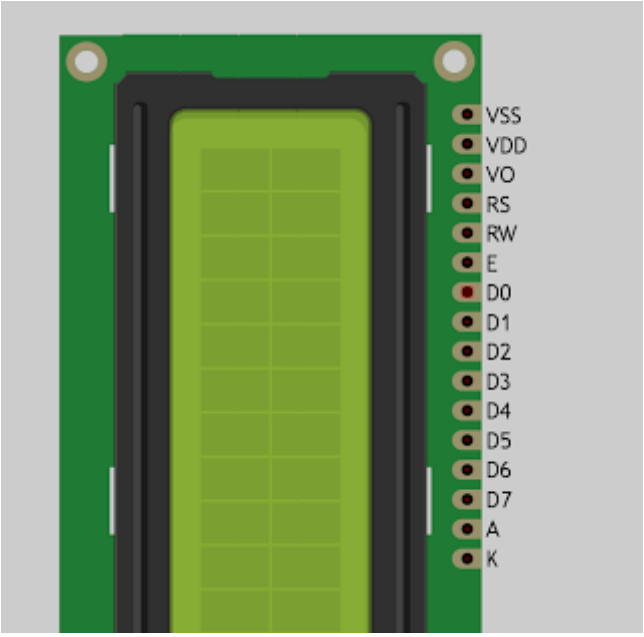
- Realice:

- Realice:

LABORATORIO 9

PANTALLA LCD

Primero de todo observaremos la serigrafía escrita en los pines de nuestra pantalla, siendo la siguiente:



VSS que es el pin de negativo o masa o 0 volts o GND.
VDD es la alimentación principal de la pantalla y el chip, lleva 5 voltios (recomendable ponerle en serie una resistencia para evitar daños, con una de 220 ohmnios es suficiente).
VO es el contraste de la pantalla, debe conectarse con un potenciometro de unos 10k ohms o una resistencia fija una vez que encontremos el valor deseado de contraste. Tengan en cuenta que si no conectan esto, no verán nada.
RS es el selector de registro (el microcontrolador le comunica a la LCD si quiere mostrar caracteres o si lo que quiere es enviar comandos de control, como cambiar posición del cursor o borrar la pantalla, por ejemplo).
RW es el pin que comanda la lectura/escritura. En nuestro caso siempre estará en 0 (conectado a GND) para que escriba en todo momento.
E es enable, habilita la pantalla para recibir información.

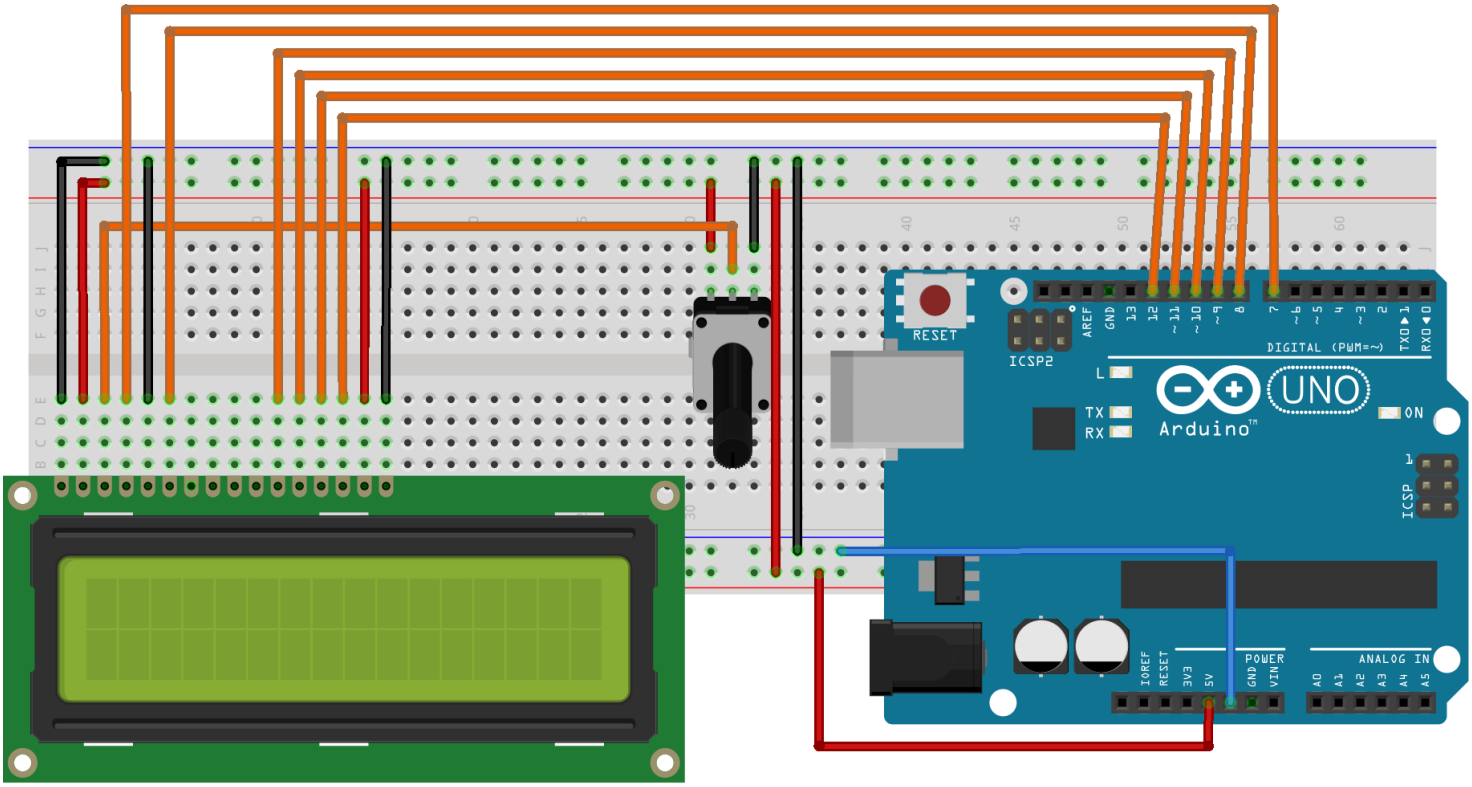
D0~D3 no los vamos a utilizar. Como pueden ver la pantalla tiene un bus de datos de 8 bits, de D0 a D7. Nosotros solamente utilizaremos 4 bits, de D4 a D7, que nos servirán para establecer las líneas de comunicación por donde se transfieren los datos. A y K son los pines del led de la luz de fondo de la pantalla. A se conectará a 4 o 5 volts y K a gnd.

Preparación de las conexiones de la pantalla LCD 1602A

La pantalla LCD viene sin conectores por defecto. Hay dos soluciones para este problema: soldar cables o soldar pines macho de 2,54mm. He optado por la segunda opción por la comodidad que representa (menos cable y acoplan perfectamente con la breadboard).

Procederemos a la soldadura de los mismos, siendo el resultado el siguiente:

Circuito:



Primero que todo, la pantalla necesitará ser alimentada. Conectaremos dos cables, uno al pin de la placa Arduino UNO +5V y otro al GND para conectarlos a las filas "+" y "-" de la breadboard.

Conexión: Arduino 5V --> fila +
Conexión: Arduino GND --> fila -

Ahora procederemos a la preparación del contraste de la pantalla LCD. Para ello haremos las siguientes conexiones mediante cables:

Conexión: fila GND (fila -) de la breadboard --> pin 1 de la LCD (VSS)
Conexión: fila 5V (fila +) de la breadboard--> pin 2 de la LCD (VDD)
Conexión: fila 5V (fila +) de la breadboard--> pin 15 de la LCD (A)
Conexión: fila GND (fila -) de la breadboard --> pin 16 de la LCD (K)

Para probar la correcta conexión, encenderemos la placa Arduino UNO mediante el cable USB al ordenador y veremos que la pantalla LCD se ilumina.
El siguiente paso es la introducción del potenciómetro, para ajustar el contraste de la pantalla. En mi caso he utilizado un potenciómetro de 50Kohmnios, pero uno de 10k también es válido. Lo conectaremos a la izquierda de la pantalla LCD sobre la breadboard y procederemos al cableado para la conexión de sus tres pines.

Conexión: primer pin del potenciómetro---> GND de la breadboard (fila -)
Conexión: pin de en medio potenciómetro --> pin 3 de la pantalla LCD (VO)
Conexión: tercer pin del potenciómetro---> 5V de la breadboard (fila -).

Cuando la placa Arduino esté alimentada (conexión USB-PC), se verá por pantalla caracteres en forma de cuadrado en la fila de arriba. Prueba a ajustar con el potenciómetro y verificar que todo funciona correctamente.
En el próximo paso, vamos a conectar la pantalla LCD a la placa Arduino UNO para que se pueda mostrar el mensaje de texto que queramos.

Conexión: pin 4 de la LCD (RS)---> pin 7 del arduino (salida digital, PWM)

Conexión: pin 5 de la LCD (RW) --> GND de la breadboard (fila -)
Conexión: pin 6 de la LCD (E)--> pin 8 de la placa Arduino UNO (PWM)
Conexión: pin 11 de la LCD (D4)--> pin 9 de la placa Arduino UNO (PWM)
Conexión: pin 12 de la LCD (D5)--> pin 10 de la placa Arduino UNO (PWM)
Conexión: pin 13 de la LCD (D6)--> pin 11 de la placa Arduino UNO (PWM)
Conexión: pin 14 de la LCD (D7)--> pin 12 de la placa Arduino UNO (PWM)

Código:

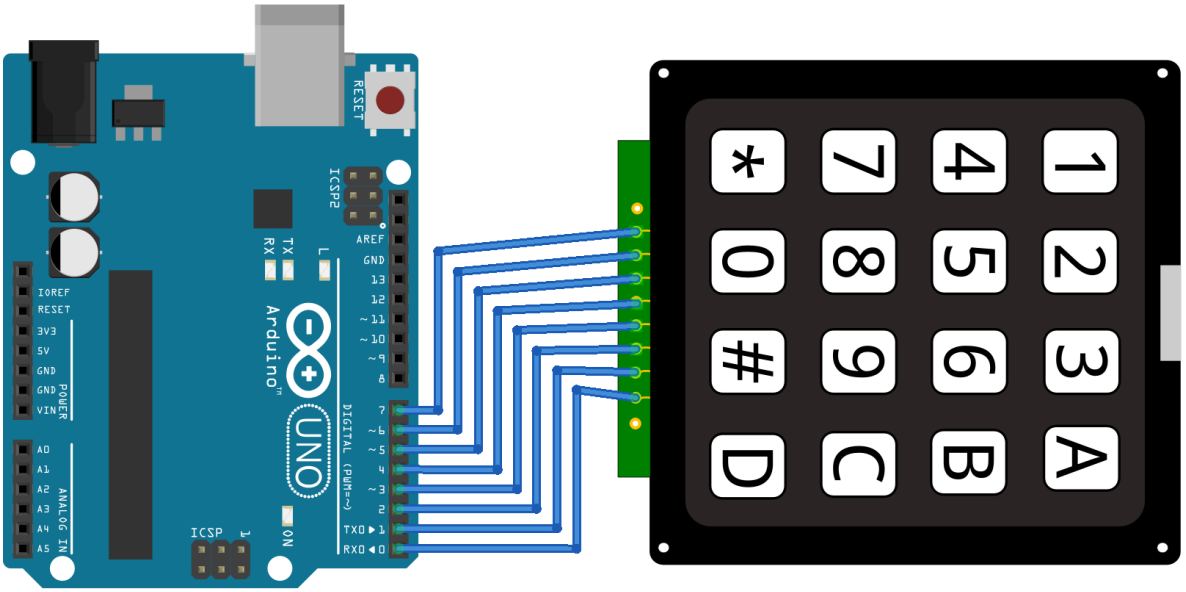
```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7, 8, 9, 10, 11 , 12);

void setup() {
  lcd.begin(16, 2);
  lcd.setCursor(0,1);
  lcd.write("El cajon de Ardu ");
}

void loop()
{

}
```

Practica 9

- Realice:

LABORATORIO 10

TECLADO MATRICIAL




```
#include<Keypad.h> //libreria para utilizar teclado
matrcial 4x4
#include<Servo.h>
Servo servomotor;
const byte filas = 4;
const byte columnas = 4;
byte pinesF[filas] = {9,8,7,6};
byte pinesC[columnas] = {5,4,3,2};
char teclas[filas][columnas] = {
 {'1','2','3','A'},
 {'4','5','6','B'},
 {'7','8','9','C'},
 {'*','0','#','D'}
};
Keypad teclado = Keypad(makeKeymap(teclas), pinesF,
pinesC, filas, columnas);
char tecla;
```

```
void setup() {
  servomotor.attach(10);
 Serial.begin(9600);
}
void loop() {
 tecla = teclado.getKey();
 if (tecla == '1'){
  servomotor.write(90);
 Serial.print(tecla);

 }
}
```

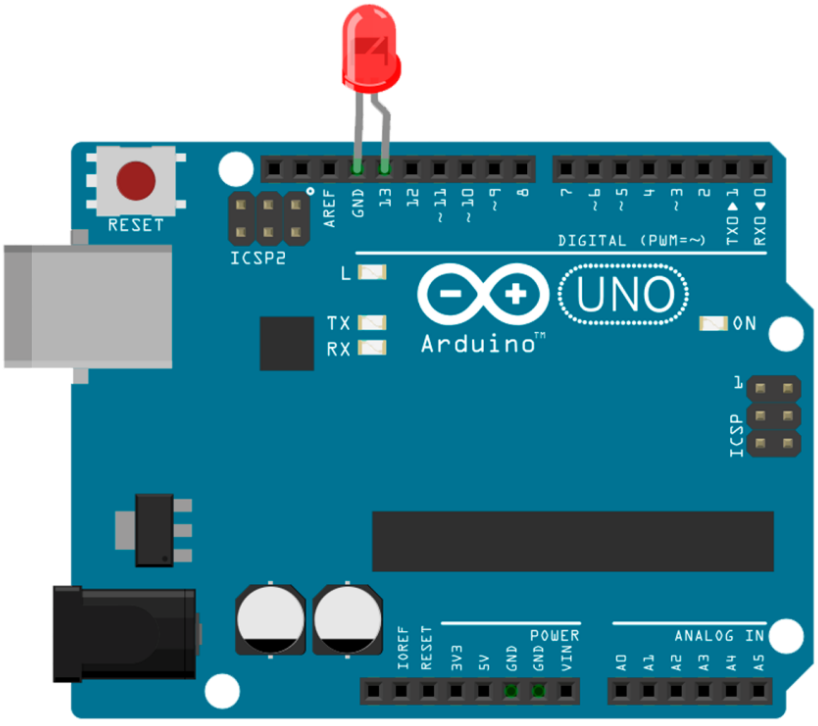
MAS EJEMPLOS

LED
(Diodo emisor de luz)

```
int led = 13;
void setup() {

  pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Materiales:

- Placa Arduino.
- Diodo LED.
- Cables de conexión.

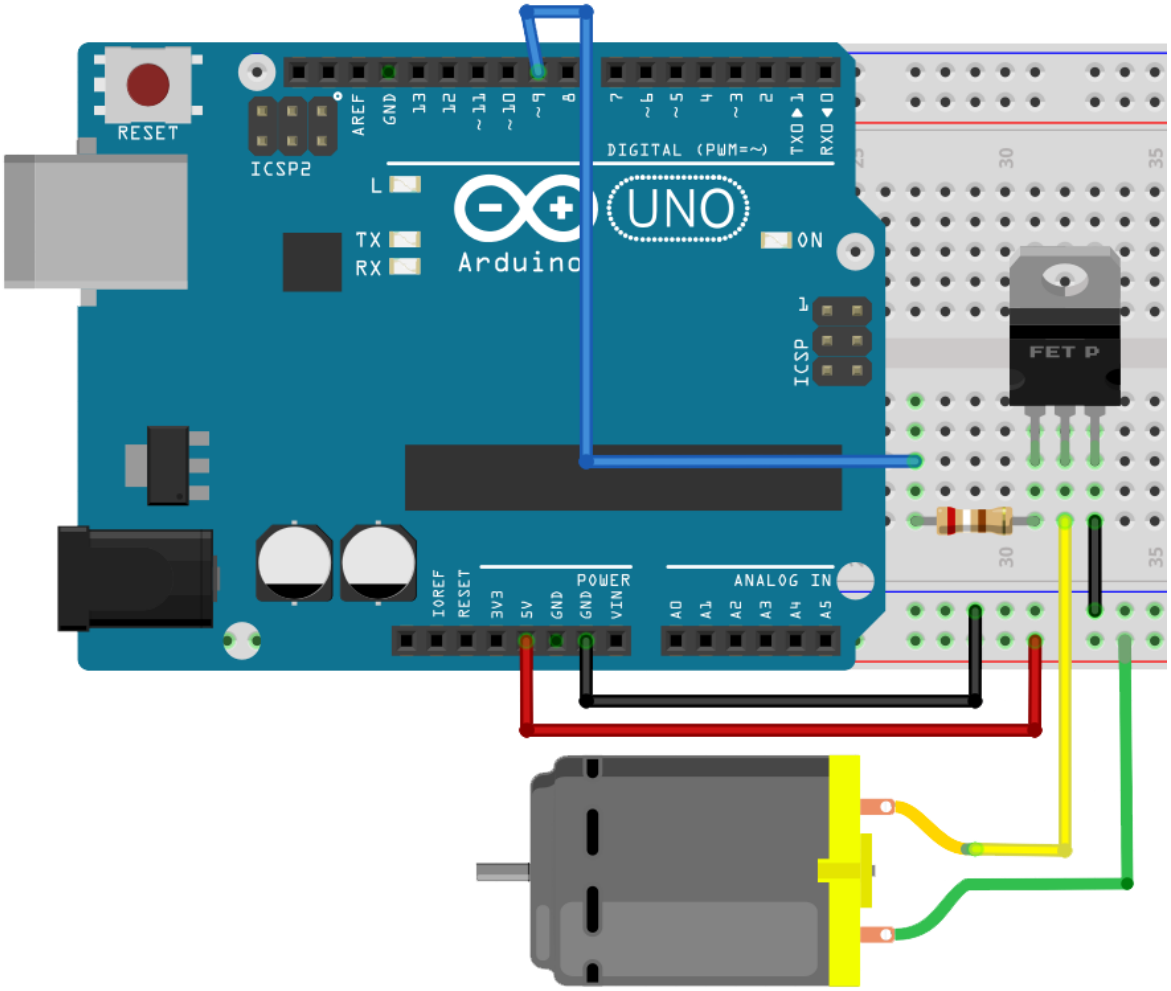


MOTORES DC
(Corriente continua)

```
int motor = 9;
void setup() {
  pinMode(motor, OUTPUT);
}
void loop() {
  digitalWrite(motor, HIGH);
  delay(1000);
  digitalWrite(motor, LOW);
  delay(1000);

}
```

Materiales:

- Placa Arduino.
- Transistor Tip122.
- Resistencia de 220 Ω.
- Cables de conexión.

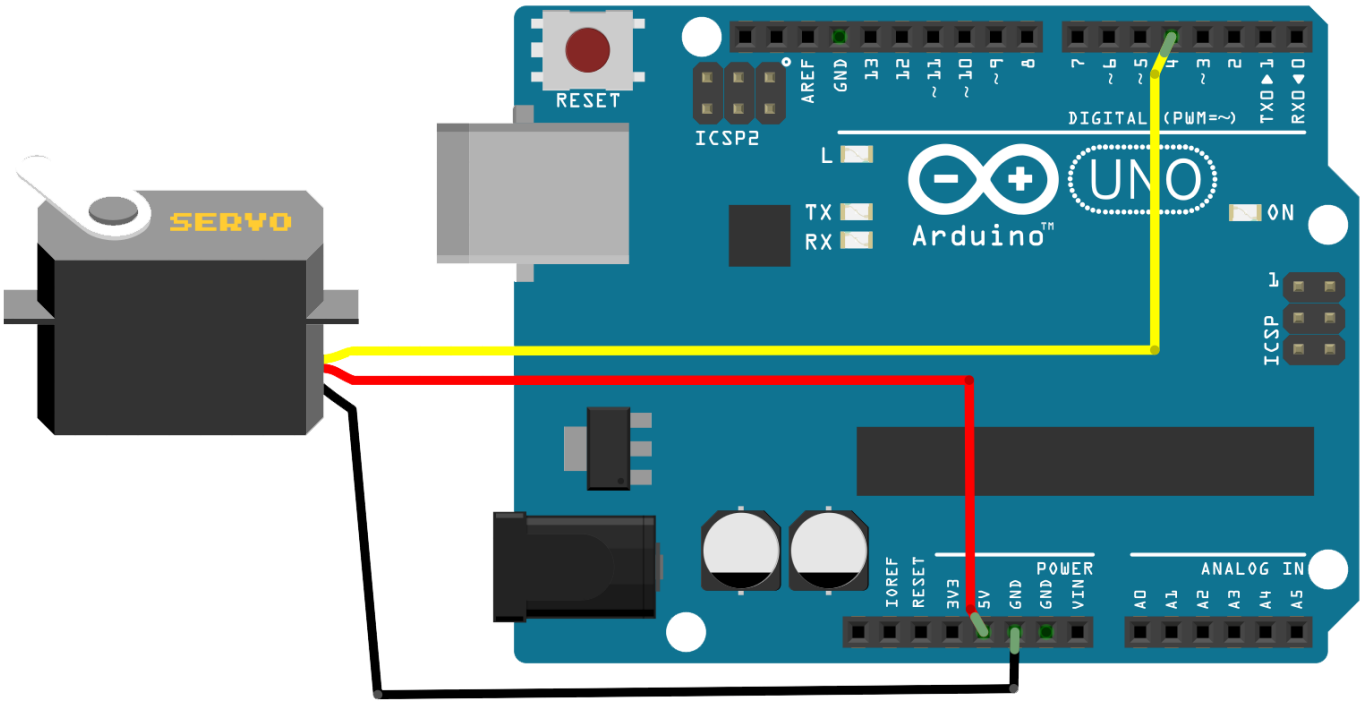


SERVOMOTOR

```
#include <Servo.h>
Servo servo1;
void setup() {
  servo1.attach(4);
}
void loop() {
  servo1.write(0);
  delay(1000);
  servo1.write(45);
  delay(1000);
  servo1.write(90);
  delay(1000);
  servo1.write(135);
  delay(1000);
  servo1.write(180);
  delay(1000);
}
```

Materiales:

- Placa Arduino.
- Servomotor.
- Cables de conexión.

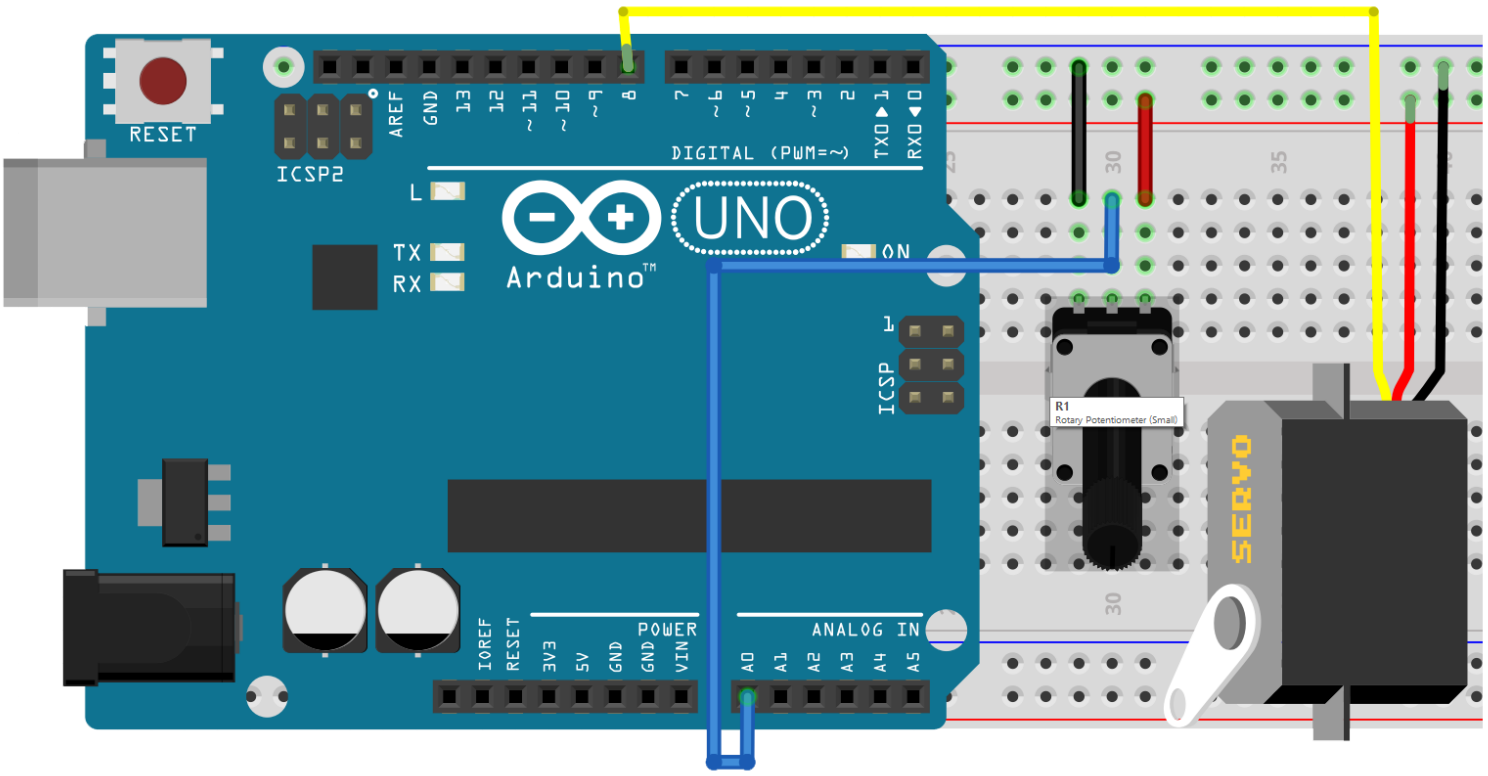


SERVOMOTOR Y POTENCIOMETRO

Materiales:

- Placa Arduino.
- Servomotor.
- Cables de conexión.
- Potenciometro (5KΩ o 10 KΩ).

```
#include <Servo.h>
Servo servo;
int pot=3;
void setup() {
  Serial.begin(9600);
  servo.attach(8);
}
void loop() {
  int valorpot=analogRead(pot);
  int val=map (valorpot,0,1023,0,180);
  Serial.println(val); ajustado
  servo.write(val);
  delay(100);
}
```



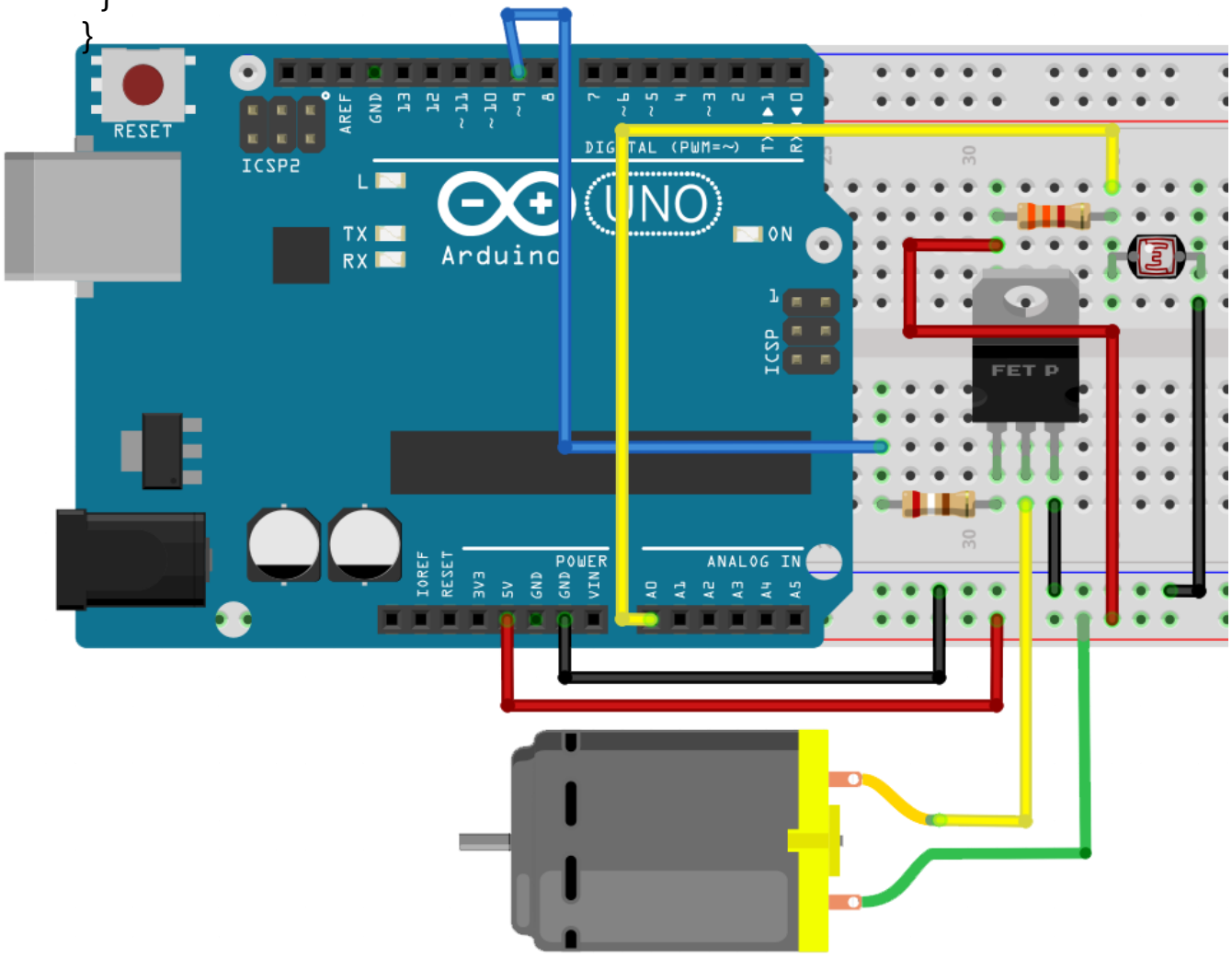
VELOCIDAD DE UN MOTOR DC CON UN LDR

Materiales:

- Placa Arduino.
- Transistor Tip122.
- Cables de conexión.
- Motor DC.
- 2 Resistencias (220 Ω y 3,3KΩ).
- Sensor LDR.

```
int valorLDR = 0;
int pinLDR = 0;
int motor = 9;
int vel=0;

void setup()
{
  pinMode(motor, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  valorLDR = analogRead(pinLDR);
  Serial.println(valorLDR);


  if(valorLDR >= 1023)
  {
    vel=20;
    analogWrite(motor, vel);
  }
  else if((valorLDR >= 823)&(valorLDR < 1023))
  {vel=50;
    analogWrite(motor, vel);

  }
  else if((valorLDR >= 623)&(valorLDR < 823))
  {
    vel=100;
    analogWrite(motor, vel);
  }
  else if((valorLDR >= 423)&(valorLDR < 623))
```

```
  {
    vel=150;
     analogWrite(motor, vel);

  }
  else  if((valorLDR >= 223)&(valorLDR < 423))
  {
    vel=200;
     analogWrite(motor, vel);
  }
  else
  {
    vel=250;
     analogWrite(motor, vel);
 }
}
```



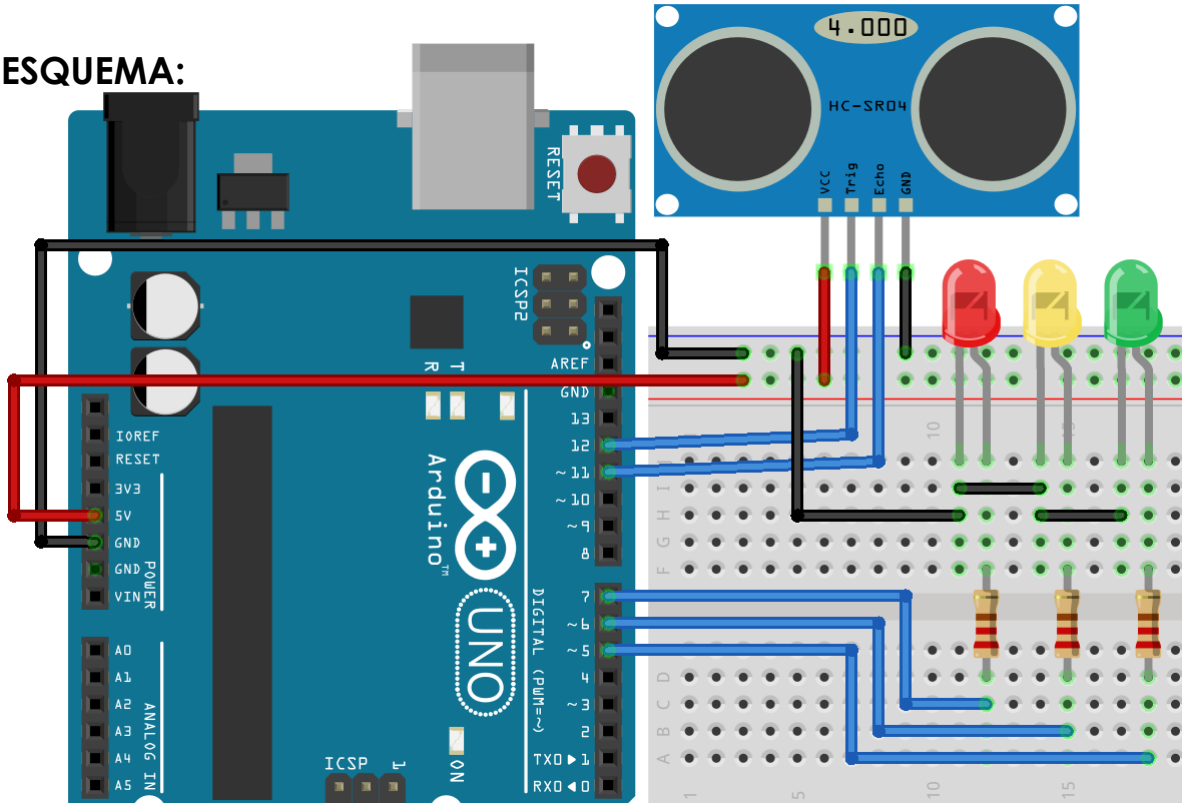
MEDIDA DISTANCIA CON ULTRASONIDOS Y LEDS

En esta entrada vamos a realizar medidas de forma cualitativa (con los colores de los leds) mediante el sensor de ultrasonidos ya utilizado en la entrada anterior. El conjunto tomará, de forma sencilla, una medida de la distancia a la cual está el objeto, encendiendo el LED rojo si dicho objeto está a menos de 20 centímetros, ámbar si está entre 20 y 40 centímetros, y verde si está a más de 40 centímetros.

Material:

- Sensor ultrasonidos HC-SR04
- Placa Arduino UNO
- LED rojo, verde y amarillo (se pueden usar los colores que tengáis, yo he usado los más familiares). 3x resistencias 220 ohmnios
- Cables
- Cable USB
- Protoboard

ESQUEMA:



Código:

```
int ledrojo=6;
int lednaranja=7;
int ledverde=8;
long distancia;
long tiempo;
int led;
void setup(){
  Serial.begin(9600);
  pinMode(12, OUTPUT);
  pinMode(11, INPUT);
  pinMode(ledrojo, OUTPUT);
  pinMode(lednaranja, OUTPUT);
  pinMode(ledverde, OUTPUT);
}
void loop(){
  digitalWrite(12 ,LOW);
  delayMicroseconds(10);
  digitalWrite(12, HIGH);
  delayMicroseconds(10);
  tiempo=pulseIn(11, HIGH);
  distancia= int(0.017*tiempo);
  if (distancia>40){
    led=1;
    digitalWrite(lednaranja,LOW);
    digitalWrite(ledrojo,LOW);
  }
  if (distancia<=40&&distancia>20){
    led=2;
    digitalWrite(ledverde,LOW);
    digitalWrite(ledrojo,LOW);
  }
  if (distancia<=20){
    led=3;
    digitalWrite(ledverde,LOW);
    digitalWrite(lednaranja,LOW);
  }
  switch(led){
    case 1:
```

```
    digitalWrite(ledverde,HIGH);
    break;
    case 2:
    digitalWrite(lednaranja,HIGH);
    break;
    case 3:
    digitalWrite(ledrojo,HIGH);
    break;
  }
  delay(50);
}
```

APUNTES