CONTROL DE VELOCIDAD DE MOTOR DC CON ARDUINO USANDO TRANSISTORES

¿Alguna vez has intentado encender un motor sólo a través de Arduino? Probablemente sí, y en el mejor de los casos tu motor no se habrá movido. En el peor, tu Arduino ha muerto tras una nube de humo negruzco.

Así que seguramente te estarás preguntando: "¿Por qué no puedo controlar un motor DC directamente con Arduino? ¡Si funciona con 5V!" En este ejemplo voy a explicarte cómo construir un controlador de motores mediante un transistor NPN, que podrás utilizar con Arduino u otro tipo de controlador.

Por qué no puedes conectar un motor directamente a Arduino

El problema no es el voltaje. Es la **Intensidad** (Amperios) que proporciona Arduino. La Intensidad/Corriente es la **cantidad de electrones que circulan por un conductor** cada unidad de tiempo, y Arduino a duras penas puede dar **50mA** (miliamperios). Un motor DC estándar consume unos 1000mA. Uno verá que los números no cuadran.

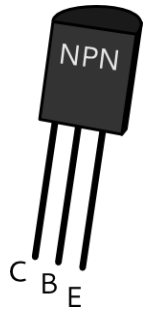
La solución es utilizar un **controlador de motores**. En cierto modo se comporta como un interruptor, cuando se cierra deja circular corriente a través de sí. Sólo que, en vez de activarse manualmente, **es activado mediante un microcontrolador** y además **es capaz de regular la velocidad del motor**.

Simplemente conectando estos controladores y con un poco de código es posible hacer funcionar un motor o cualquier otro artefacto que necesite mucha energía para funcionar.

Con un transistor NPN y un diodo y una resistencia es posible crear un controlador de motores casero fácilmente. Y por una pequeña fracción de lo que costaría prefabricado.

¿Cómo funciona un Transistor NPN?

El transistor es un **componente capaz de amplificar o disminuir la corriente que circula por él**. Es como un interruptor, pero se abre y cierra mediante corrientes eléctricos. El transistor NPN tiene tres patas, **Colector**, **Base** y **Emisor**:



NOTA IMPORTANTE: *La mayoría de transistores NPN tienen las patas en este orden (CBE). Sin embargo,* ***algunos modelos tienen las patas al revés de como está la imagen*** *(EBC). Antes de usar un transistor, mira siempre la documentación para saber cuál es su orden. Si tienes dudas, dime el modelo en los comentarios y te ayudaré.*

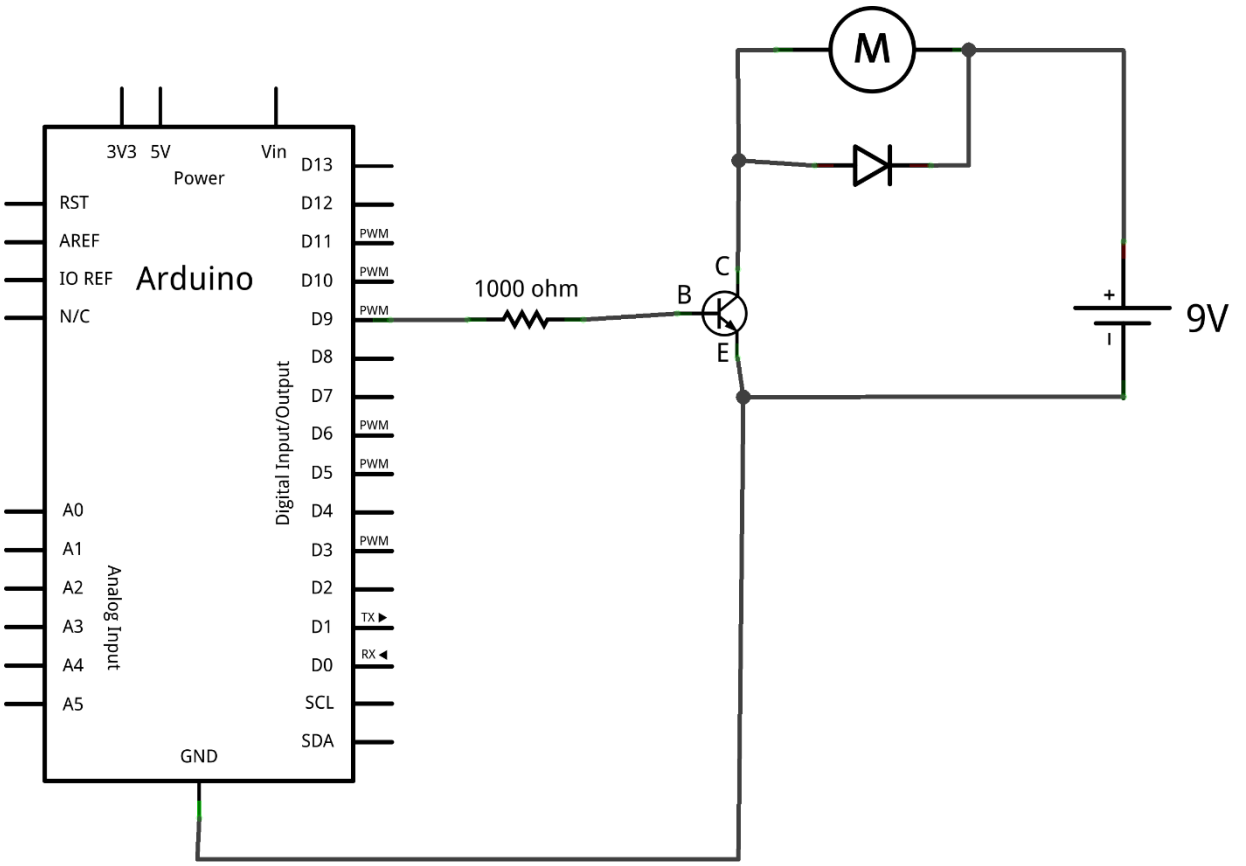
Según la corriente que pase por la Base, el transistor dejará pasar más o menos intensidad del Colector al Emisor, de tal manera que el transistor actúa como un interruptor. Si no hay corriente en la base se dice que el transistor está en **Corte**. Si la corriente en la Base es muy alta, se dice que el transistor está en **Saturación** y actuará como un interruptor cerrado.

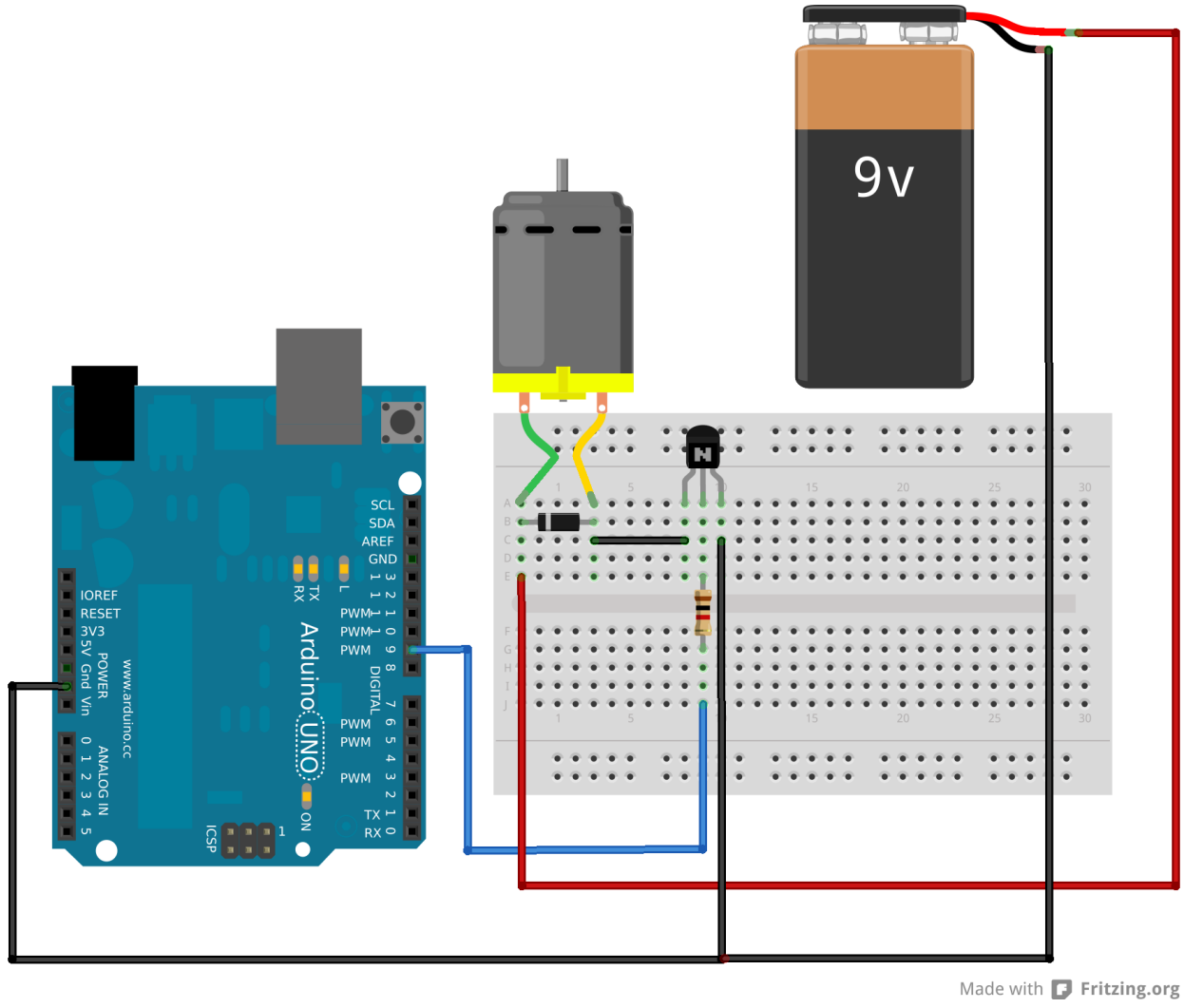
Material

Transistor NPN, Díodo, Resistencia (de 1 kΩ aproximadamente), Hilo conductor, Arduino, Motor y baterías. Las baterías deben ser suficientes para alimentar el motor

Circuito

Alimentaremos Arduino y el motor independientemente. El microcontrolador enviará un pulso a la Base del transistor y éste dejará pasar corriente entre su Colector y Emisor, cerrando el circuito y permitiendo que el motor se mueva. Aquí presento dos circuitos: ambos son iguales, pero uno es esquemático y el otro está en fritzing.


M
3V3
5V
Vin
Power
RST
AREF
IO REF
N/C
Arduino
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
SCL
SDA
PWM
TX
RX
Digital Input/Output
A0
A1
A2
A3
A4
A5
Analog Input
GND
1000 ohm
C
B
E
9V


9v
Arduino UNO
www.arduino.cc
Made with Fritzing.org

¿Para qué sirve el díodo del motor? Los motores DC, cuando se reduce rápida y drásticamente el voltaje que se les proporciona, **provocan un pico de voltaje elevado** durante un momento que puede dañar la placa. El diodo evitará este efecto (aquí hay una explicación más detallada) y te ahorrará fortunas en reemplazos. Este diodo se conoce como **Flyback Diode.**

La resistencia es por seguridad, por si se produce algun fallo y parte del corriente que circula por el transistor de desvía a Arduino. Con 9V no pasará nada. Con 60 voltios, sí.

Si utilizas Arduino ¿qué código hay que cargar? El ejemplo oficial de Fade bastará. Este código envia un pulso analógico a través del Pin Digital 9, justo lo que se necesita para controlar el transistor.

```
1
2
3     /*
4      Fade
5
6      This example shows how to fade an LED on pin 9
7      using the analogWrite() function.
8
9      This example code is in the public domain.
10     */
11
12    int led = 9;           // the pin that the LED is attached to
13    int brightness = 0;    // how bright the LED is
14    int fadeAmount = 5;    // how many points to fade the LED by
15
16    // the setup routine runs once when you press reset:
17    void setup()  {
18      // declare pin 9 to be an output:
19      pinMode(led, OUTPUT);
20    }
21
22    // the loop routine runs over and over again forever:
23    void loop()  {
24      // set the brightness of pin 9:
25      analogWrite(led, brightness);
26
27      // change the brightness for next time through the loop:
28      brightness = brightness + fadeAmount;
29
30      // reverse the direction of the fading at the ends of the fade:
31      if (brightness == 0 || brightness == 255) {
32        fadeAmount = -fadeAmount ;
33      }
34      // wait for 30 milliseconds to see the dimming effect
      delay(30);
    }
```

Recuerda revisar el esquema antes de probarlo. Los humanos tenéis un don para causar todo tipo de problemas en los circuitos. Si todo ha funcionado como debía verás que el motor se enciende, va aumentando de velocidad y se para.

Y ya está, tienes tu primer controlador de motores caseros. Este es el controlador más sencillo que puedes fabricarte, y como has visto **sólo puede controlar la velocidad de giro**. Para controlar también la dirección, se necesitan circuitos un poco más complejos, como **el Puente-H**.

CONTROL DE VELOCIDAD DE MOTOR DC CON ARDUINO CON TRANSISTOR 2N2222 (de https://www.prometec.net/motorcc/)

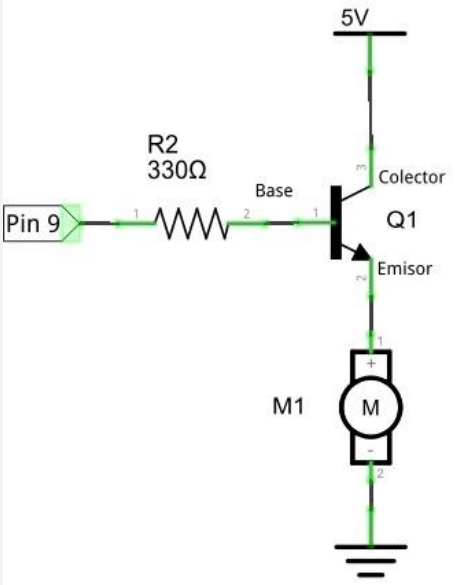
En la sesión transistores, vimos que era muy fácil hacer girar un pequeño motor mediante un transistor que regulase la intensidad de corriente que le entregamos.

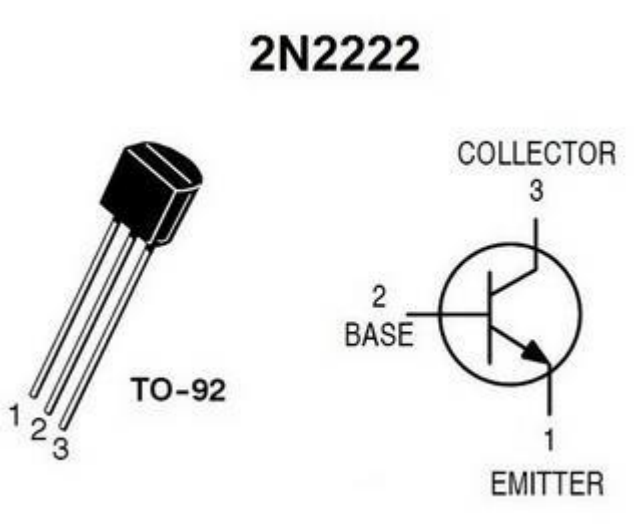
En aquella sesión nos centramos casi en exclusiva en el **transistor**, para presentar este componente, que vamos a utilizar una y otra vez, y prácticamente usamos el motor como excusa sin entrar en el tema. Pero en esta ocasión queremos poner el foco en el motorcito y como lo controlamos y por eso vamos a volver a este ejemplo, pero complicándolo un poco más (Que le vamos a hacer), añadiendo un **potenciómetro** que nos permita **variar la velocidad** de giro el motor.

Para ello usaremos un motor de continua, de 5V, con rueda, que podéis encontrar por muy poco dinero y que nos viene de perlas para apreciar el de giro.

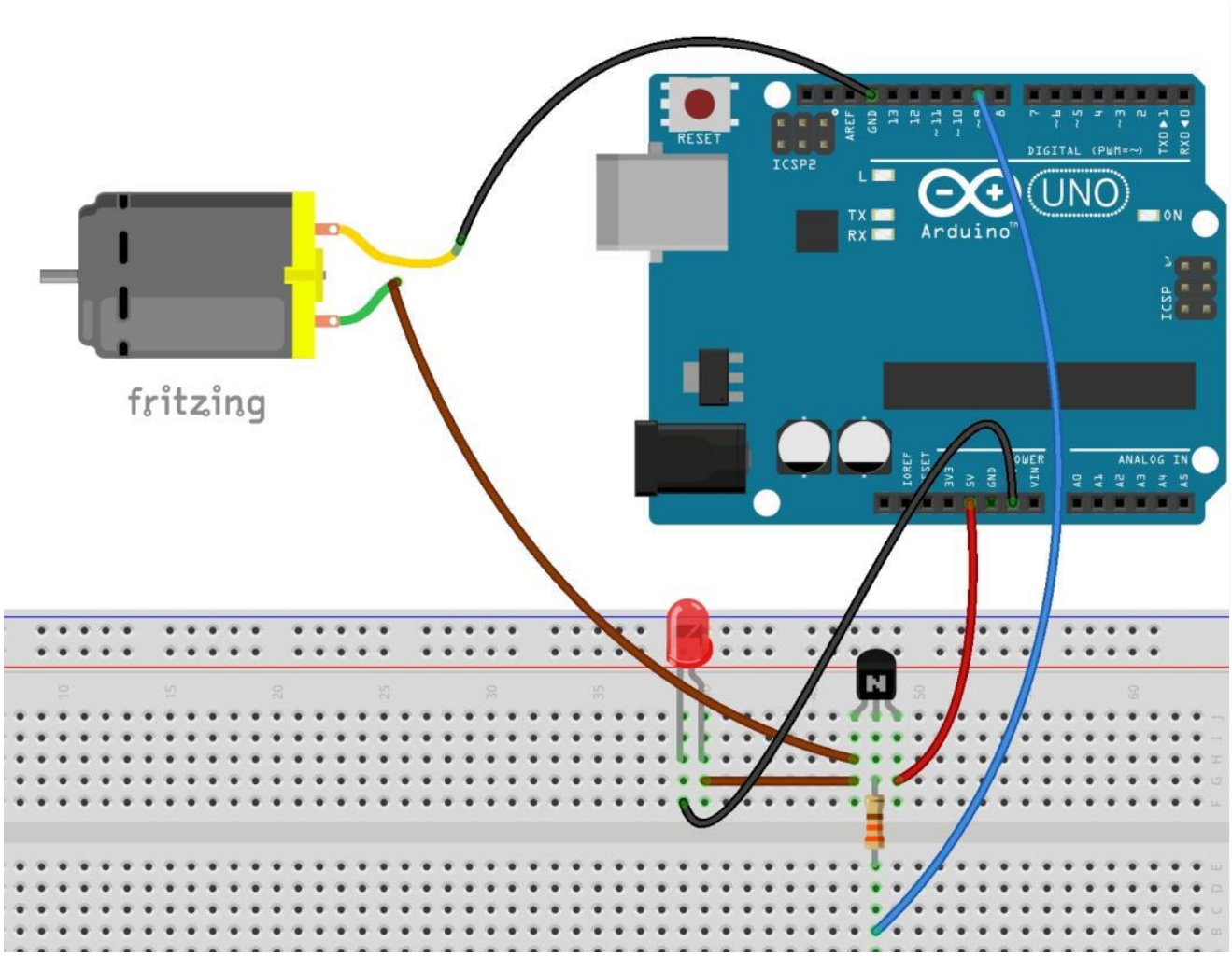
Como cualquier motor, por pequeño que sea consume más de lo que uno de los pines de nuestros Duinos pueden proporcionar, necesitamos usar un **transistor** como el 2N2222 para alimentarlo. Y usaremos uno de los pines de **Arduino** para gobernar este transistor.

Vamos a repetir el montaje que vimos en la sesión **transistores** con este motorcito. Os incluyo copia del esquema eléctrico que usamos allí y del diagrama de pines del transistor.





Aquí tenéis el esquema inicial sin potenciómetro para que comprobéis las conexiones del motor y que funciona correctamente, antes de seguir.



Para comprobar que el motor funciona, podéis cargar este pequeño programa, que simplemente varia la tensión que ponemos en el pin 9 para modificar la velocidad de giro del motor:

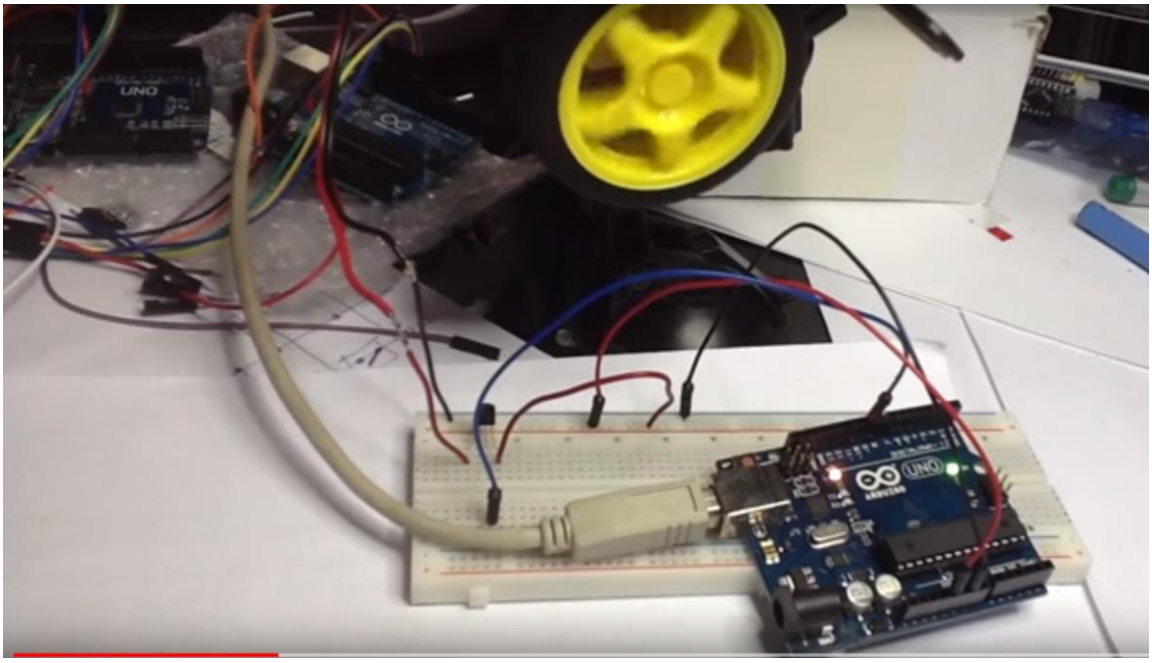
```
const int control = 9 ;

void setup()

   {    pinMode(control,  OUTPUT) ; }

void loop()

   {

        for ( int n = 0 ; n < 255 ; n++)

          {

               analogWrite (control,  n) ;

               delay(15) ;

          }

   }
```

Que como veréis, simplemente pone un valor analógico en el pin 9. Podemos controlar la velocidad del motor variando la intensidad de tensión que ponemos en la base del transistor, ya que esta regula la resistencia que el transistor presenta entre emisor y colector.

A 0V el transistor entra en **corte** (Resistencia infinita) y a 5V está en **saturación**, con lo que presenta resistencia nula.

- *No supongáis que podemos hacer que el motor gire de forma continua para todos los valores de tensión en la base. Por debajo de un cierto umbral se parará y solo girará cuando este se supere.*
- *Al hacer crecer el valor de tensión que le damos al motor, la velocidad de giro, irá aumentando progresivamente.*

Aquí os pongo un pequeño video con el resultado:



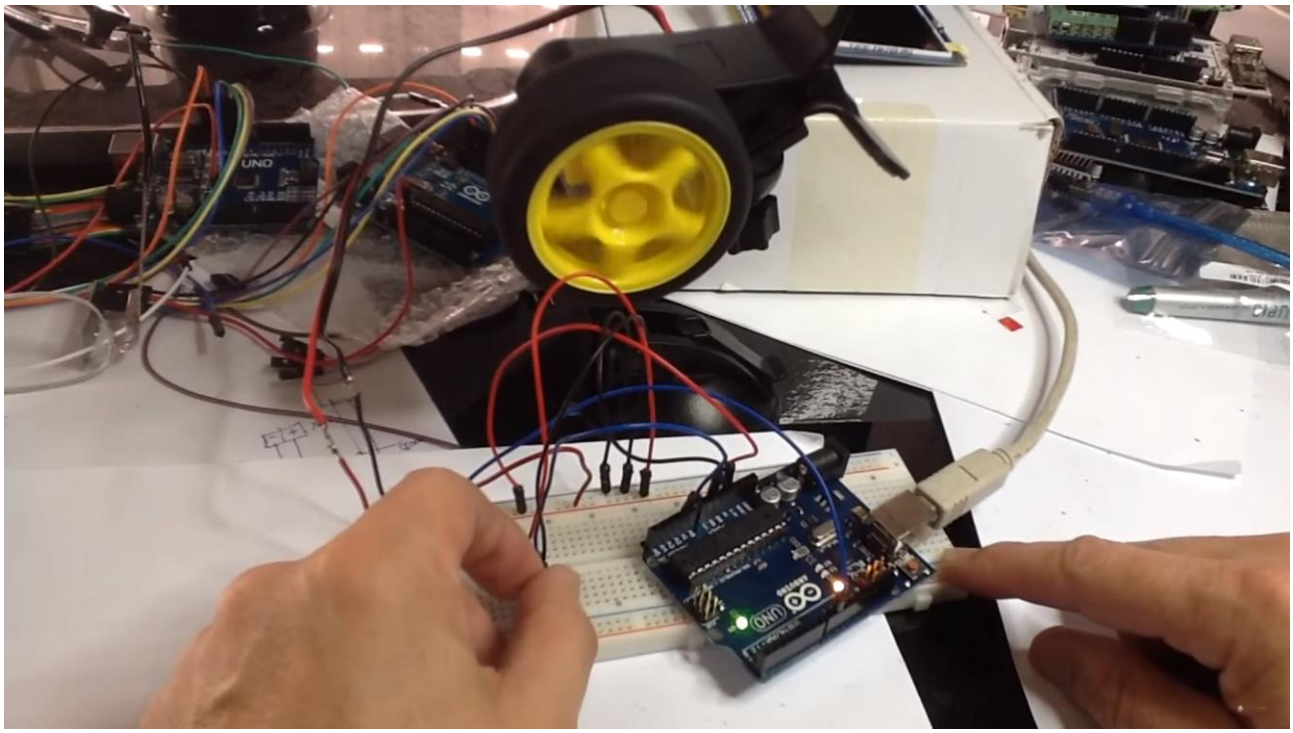
Vamos a probar ahora a colocar un potenciómetro conectado a la puerta analógica A1, cuya lectura usaremos para variar la señal que ponemos en la base del transistor. Con ella variaremos la velocidad de giro.

La idea es que a medida que aumentamos o disminuimos la tensión en la base del transistor, la caída de tensión en este varia de forma acorde haciendo que el motor reciba más o menos tensión y esto se traduce en una cambo de su velocidad de giro.

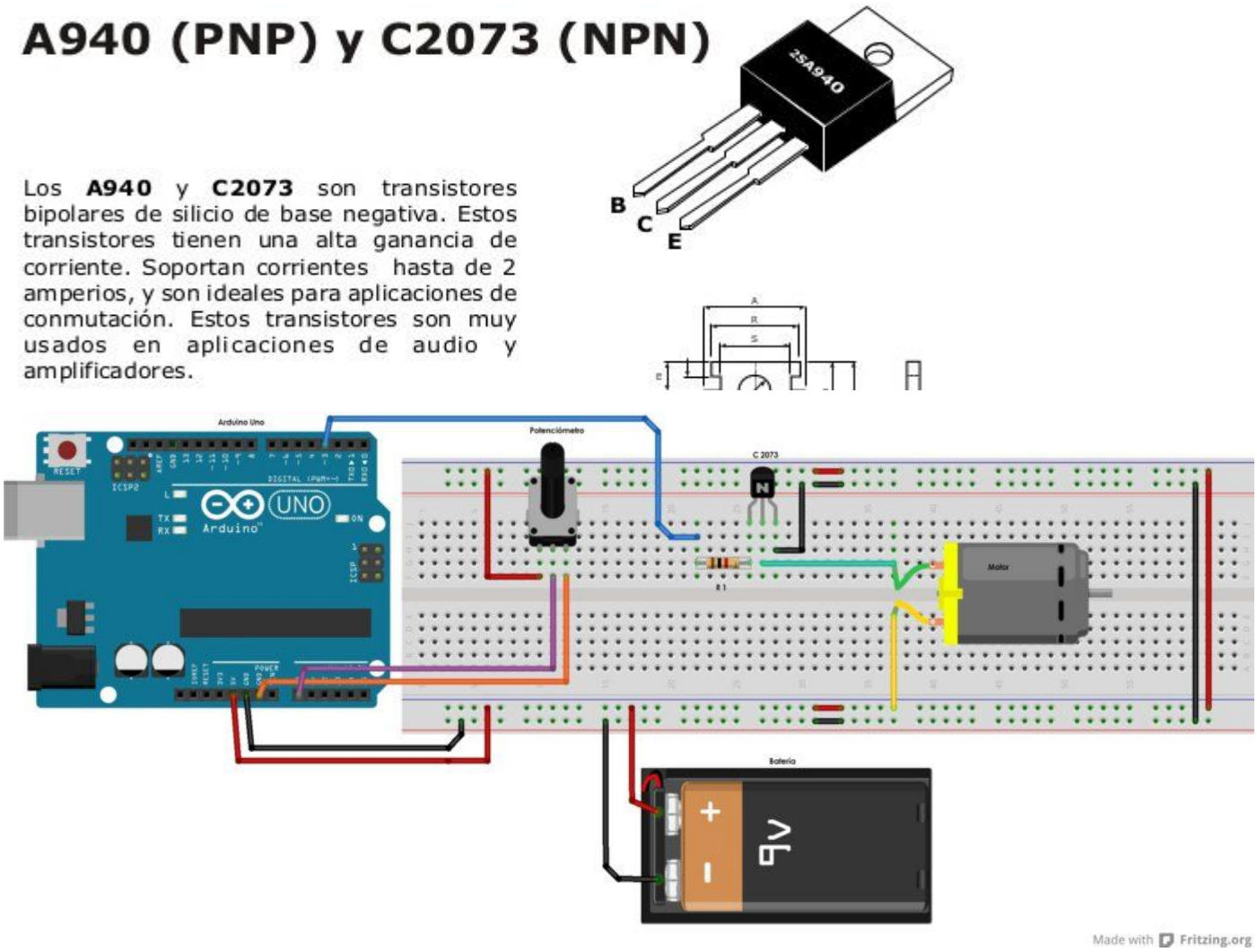
Y un programa que lee el potenciómetro y varía la velocidad de giro en consonancia puede ser algo como ésto:

```
const int control = 9 ;

void setup()

  {    pinMode(control,  OUTPUT) ;

       Serial.begin(9600);

  }



void loop()

  {

       int n = analogRead(A1)/ 4;     // Las lecturas analogicas van hasta 1024 y no hasta 255

       Serial.println(n);
```

```
      analogWrite (control,  n) ;

      delay(200) ;

  }
```



CONTROL DE VELOCIDAD DE MOTOR DC CON ARDUINO CON TRANSISTOR C2073



La importancia del Control de velocidad de Motor DC por potenciómetro radica en que el usuario podrá regular directamente la velocidad del motor DC, ajustándolo así a sus necesidades.

En otro post habíamos hablado sobre cómo regular la velocidad del motor DC por luminosidad (consultar Control de Velocidad de un Motor DC por Luz).

Los componentes utilizados en esta experiencia fueron: 1 Batería de 9 Voltios, 1 Transistor NPN (C2073), Jumpers, 1 Resistencia de 1 kΩ, 1 Motor DC, 1 Arduino Uno, 1 Protoboard, 1 Potenciómetro.

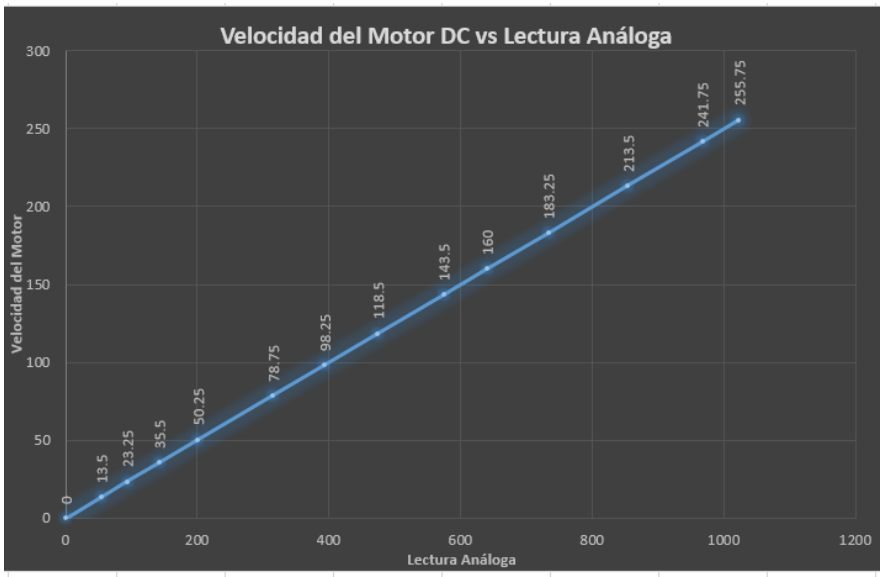
En esta experiencia se utilizó potenciómetro, debido a que es un circuito con baja corriente, y no un reóstato ya que este disipa más potencia y es utilizado para circuitos de mayor corriente.

Los extremos del potenciómetro se conectan al +5V y a GND. El punto central, se conecta a una entrada analógica, en este caso (A0).

Como recordamos los pines analógicos en Arduino son manejados por un convertidor analógico/digital de 10 bits, por lo que entregan a su salida, valores entre 0 y 1023. De esta manera la tensión que entrega el potenciómetro a la entrada analógica, en la función analogRead, variará entre 0 (cuando esté a 0V) y 1023 (cuando esté a 5V).

Abrimos Arduino IDE y escribimos el siguiente código:

```
void setup(){
  Serial.begin(9600);
}
void loop(){
float niv=analogRead(A0)/4;
analogWrite(3, niv);
Serial.println (niv);

delay(1000);
}
```




```
void setup(){
  Serial.begin(9600);
}
void loop(){
float niv=analogRead(A0)/4;
analogWrite(3, niv);
Serial.println (niv);

delay(1000);
}
```

Este código es igual al que vimos en el post Control de velocidad de un motor DC por luz.

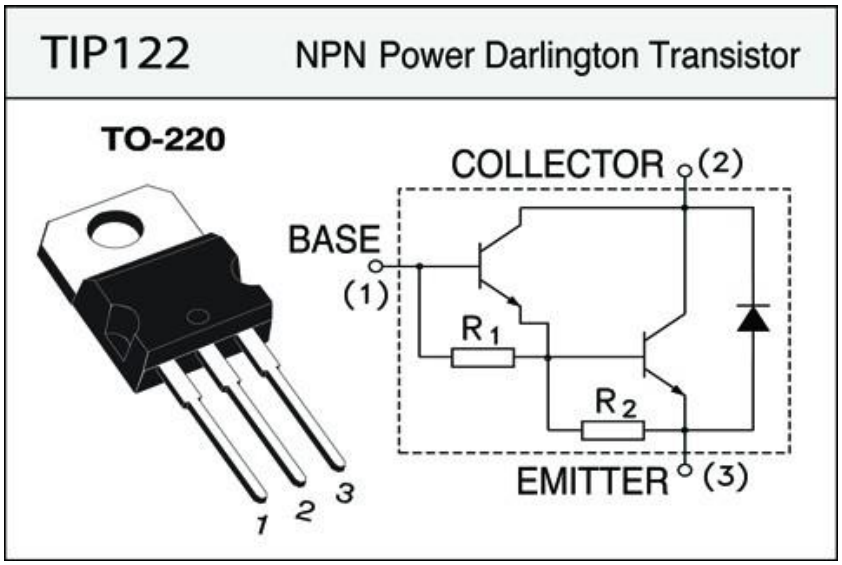
Para tener un concepto más claro veamos el comportamiento del siguiente gráfico:

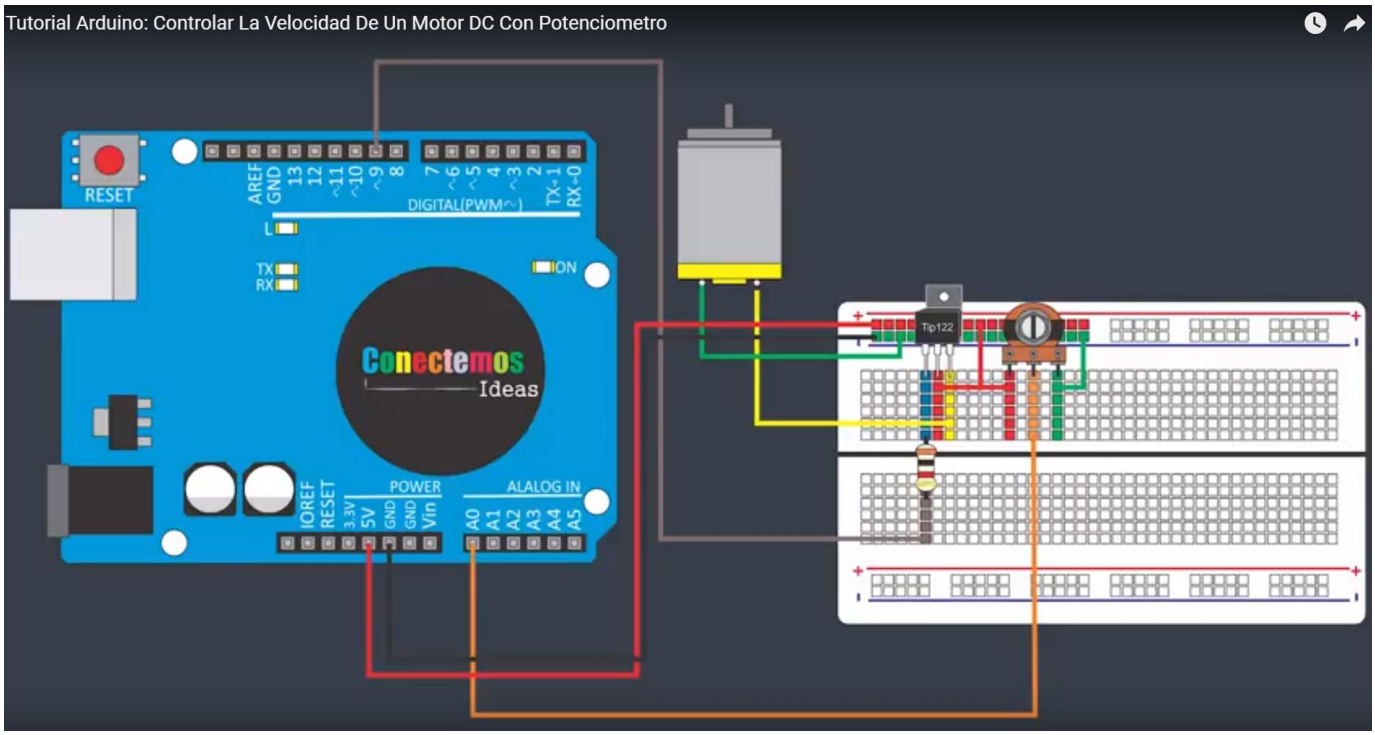
Valores de Lectura Análoga (analogRead(A0))

Gráfico de Velocidad del Motor vs Lectura Análoga.

La gráfica muestra una relación directamente proporcional de la lectura análoga con relación a la velocidad del motor, que es la pendiente positiva que podemos apreciar.

CONTROL DE VELOCIDAD DE MOTOR DC CON ARDUINO Y TIP 122(Par Darlington)





```
Control de velocidad de un  motor con potenciometro (PWM)

*/

int PotPin = A0; // Pin análogo A0 en donde está conectada la señal del potenciómetro

int Motor = 9;   // Pin 9 de nuestro arduino llamado Motor

int ValPot = 0;  //  lo utilizamos para guardar el valor leido de PotPin
int pwm1;         // lo utilizamos para guardar el valor convertido a pwm1
void setup() {

  pinMode(Motor, OUTPUT); // //Motor = Salida

             }

void loop() {

  ValPot = analogRead(PotPin); //Leemos el Pin análogo A0 (PotPin) en donde está conectada la señal del potenciómetro
                               //Lo guardamos en (ValPot)

  pwm1 = map(ValPot, 0, 1023, 0, 255);
                         //Convertimos a PWM
                         //pwm1 = (valor que recibo, de Mínimo, de Máximo, a Mínimo, a Máximo)

 analogWrite(Motor, pwm1);


          }
```