DISEÑO DE ROBOTS CON *ARDUINO*

Revisado: 22/08/2010

Después del curso básico sobre Arduino nuestro objetivo es afrontar el diseño y construcción de un móvil robotizado con Arduino. Para ello vamos a conocer en primer lugar varios elementos fundamentales para la construcción del mismo.

MOTORES PARA NUESTRO ROBOT.

Servomotores de rotación no continua.

Los servomotores son motores que pueden girar dentro de un rango de 180º con bastante precisión, si no son muy potentes podemos alimentarlos directamente de la placa de arduino sin necesidad de alimentación externa. Existen gran variedad de potencia y tamaño, en función del proyecto podemos elegir unos u otros.

Cuando realicemos el montaje conectaremos el cable negro (o marrón) a tierra, el cable rojo a 5V y el cable blanco (o amarillo) al pin que haremos servir para enviarles los pulsos de configuración de la posición (en las placas arduino los pines 9 y 10).

Cable	Conexionar a ...
Negro o Marrón	Tierra (GND)
Rojo	5 V
Blanco o Amarillo	Pin que enviará los pulsos de configuración de la posición (en Arduino pines 9 o 10)

A la hora de programarlo utilizaremos la librería Servo de Arduino, que nos proporciona una instrucción con la cual podemos indicar los grados concretos a los que queremos que se posicione .

```
#include <Servo.h>

Servo myservo;

void setup()
{
  // Pin donde tenemos conectado el servo
  myservo.attach(9);
}

void loop()
{
  // Enviamos al servo la posición en grados
  // y esperamos a que se posicione
  myservo.write(30);
  delay(15);
}
```

Servomotor 6001 HB
#include <Servo.h> //se incluye la librería servo

Servo servo01; se define la variable servo01 como tipo Servo

void setup(){ // configuración
servo01.attach(13); // se conecta el servo01 al pin 13
}

void loop(){ // bucle principal
servo01.write (0); // pone el servo a 0º
delay (1000); // espera 1 seg.
servo01.write (90); // pone el servo a 90º
delay (1000); // espera 1 seg.
}

Servomotor de rotación continua.
Al construir nuestros robots casi siempre utilizamos este tipo de motores por dos razones: en primer lugar no necesitan circuitería extra ni alimentación externa, y en segundo lugar porque son bastante precisos. El tipo de servo continuo que utilizamos es casi siempre de la casa futuba, el cual podemos adquirir desde la web de parallax (www.parallax.com). Aunque también podemos adquirir servos estándar y intentar convertirlos en servos de rotación continua [ver página 108 del libro "Computación Física en Secundaria"].
Al igual que los servos no continuos conectaremos el cable negro a tierra (GND), y el cable rojo a 5V, pero en cable blanco en este caso lo podremos conectar a cualquier salida digital que tengamos libre.

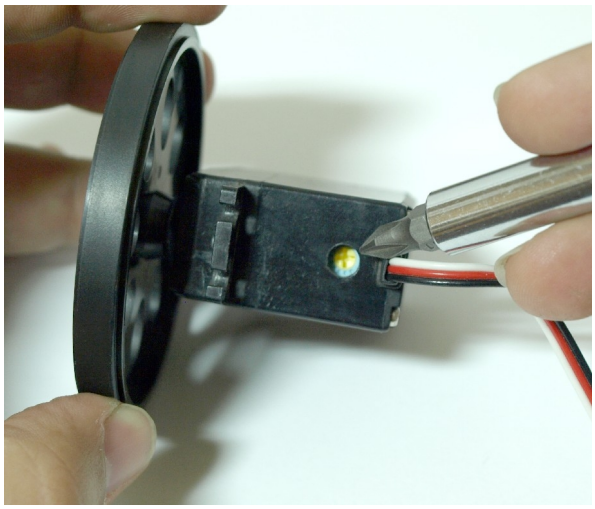
Cable	Conexionar a ...
Negro o Marrón	Tierra (GND)
Rojo	5 V
Blanco	Salida digital que tengamos libre

A la hora de programarlos tendremos que enviar un pulso de 20 milisegundos con una amplitud de

la cresta de 1 a 3 milisegundos, según la amplitud de estos pulsos el motor girará hacia un lado u otro a mayor o menor velocidad.

```
void setup()
{
  // Pin donde tenemos conectado el servo
  pinMode(2);
}

void loop()
{
  // Activamos la salia durante 1ms y la
  // mantenemos a baja 20ms para hacer el pulso.
  digitalWrite(2,HIGH);
  delayMicroseconds(1000);
  digitalWrite(2,LOW);
  delayMicroseconds(20000);
}
```

A veces nos podemos encontrar que los servos están descalibrados. Para calibrarlo haremos un programa que envíe un pulso de 1,5 ms de duración (para pararlo) y con un pequeño destornillador de estrella giraremos el tornillo blanco del servo hasta que se pare.



Servomotor de rotación Continua SM-S4303R

El programa para controlarlo es:

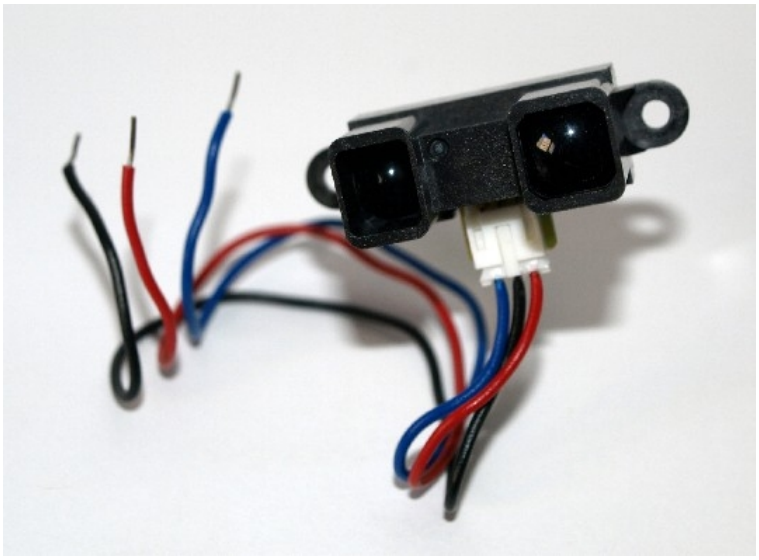
```
void setup() {
pinMode (13,OUTPUT); // se configura el pin 13 como de salida (se conecta cable blanco)
}

void loop() {// bucle principal
int x;
for(x=0;x<500;x=x+1) { //repite 500 veces la función izquierda
izquierda();}

for(x=0;x<500;x=x+1) { //repite 500 veces las función para
para();}

for(x=0;x<500;x=x+1) { //repite 500 veces la función derecha
derecha();}

for(x=0;x<500;x=x+1) { //repite 500 veces la función para
para();}
}

void izquierda() {
digitalWrite (13,HIGH);
delayMicroseconds (2100); //se da un pulso alto durante 2,1 ms
digitalWrite (13,LOW);
delayMicroseconds (20000); //se da pulso bajo durante 20 ms (esto está estandarizado)
}

void para() {
digitalWrite (13,HIGH);
delayMicroseconds (1500); //se da un pulso alto durante 1,5 ms
digitalWrite (13,LOW);
delayMicroseconds (20000);
}
```

```
void derecha() {
digitalWrite (13,HIGH);
delayMicroseconds (900); //se da un pulso alto durante 0,9 ms
digitalWrite (13,LOW);
delayMicroseconds (20000); //se da pulso bajo durante 20 ms (esto está estandarizado)
}
```

SENSORES PARA NUESTRO ROBOT.

Sensor de proximidad .

Los sensores de proximidad que haremos servir son los “sharp 2Y0A02” . Estos sensores utilizan tecnología de infrarrojos para detectar a que distancia esta el objeto, mediante un emisor y un receptor de luz infrarroja detectan la distancia a la que esta el objeto que tenemos delante y nos envía un voltaje mayor o menor en función de la distancia. Existen varios modelos de este tipo de sensores sharp, lo que varia de un modelo a otro es la precisión a grandes distancias o a cortas, según el proyecto utilizaremos unos u otros.



A la hora de conectarlos necesitaremos un conector JST de 3 terminales de donde saldrán 3 hilos de conexión, 2 para alimentar el sensor (rojo → 5V ; negro → GND) y un tercero que nos da el valor de distancia. Este tercer hilo (azul en nuestra fotografía) lo conectaremos a uno de los pines de entrada analógicos y con la función “AnalogRead” obtendremos el valor del sensor.

Cable	Conexionar a ...
Negro o Marrón	Tierra (GND)
Rojo	5 V
Azul	Pin de entrada analógica libre

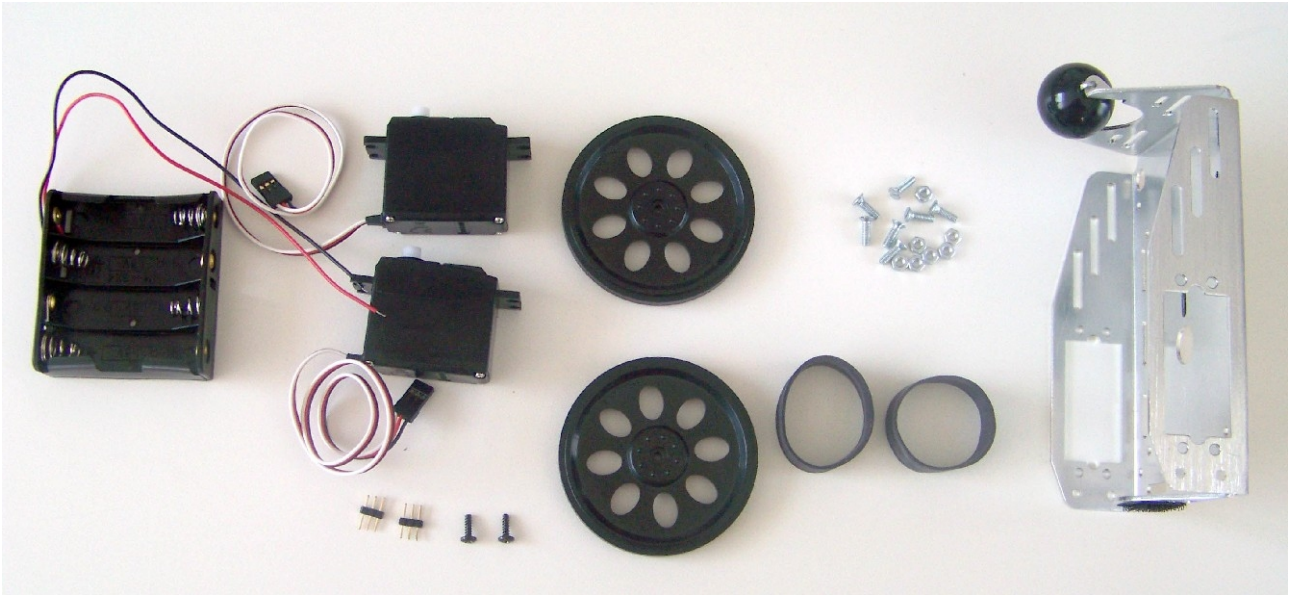
```
void setup(){
pinMode(13,OUTPUT);
Serial.begin(9600);
}
void loop()
```

```
int a;
pinMode(13,OUTPUT); /*disparo del medidor de longitud */
digitalWrite(13,HIGH); /se le da un pulso de 2 microsegundos para indicar que se quiere medir
delayMicroseconds(2);
digitalWrite(13,LOW);
pinMode(13,INPUT); /* modo lectura de retardo de señal */
a = pulseIn(13, HIGH)/37; /*lo pasamos a centimetros para eso dividimos por 37 ya va incluido la división entre 2 para la distancia*/
Serial.println (a);
delay(5000); /* espero 5 segundos para la siguiente lectura */
}
```

PROGRAMACIÓN DEL ROBOT.

Robot con rutas pre-programadas.

Para esta práctica montaremos dos servomotores de rotación continua a un chasis metálico, tal y como se explica en el apéndice de la página 107 del libro “Computación Física en Secundaria”. Y desde arduino programaremos una ruta para que el robot siga un camino concreto. El chasis metálico que utilizamos es el de los robots boebot de parallax, pero podemos fabricarnos el nuestro propio con piezas de mecano, madera, o con cualquier otro material que tengamos a mano.



Para hacer más simple la programación crearemos 4 funciones “izquierda”, “derecha”, “adelante” y “atras”. Como los servos están colocados simétricamente, para hacer mover el robot hacia adelante tendremos que girar un motor al lado contrario de su motor simétrico.

```
void adelante() {
digitalWrite(2,HIGH);
delayMicroseconds(1000);
digitalWrite(2,LOW);
digitalWrite(3,HIGH);
delayMicroseconds(2000);
digitalWrite(3,LOW);
```

```
void atras() {
digitalWrite(3,HIGH);
delayMicroseconds(1000);
digitalWrite(3,LOW);
digitalWrite(2,HIGH);
delayMicroseconds(2000);
digitalWrite(2,LOW);
```

delayMicroseconds(20000); *}*	*delayMicroseconds(20000);* *}*
void derecha() { *digitalWrite(2,HIGH);* *delayMicroseconds(2000);* *digitalWrite(2,LOW);* *digitalWrite(3,HIGH);* *delayMicroseconds(2000);* *digitalWrite(3,LOW);* *delayMicroseconds(20000);* *}*	*void izquierda() {* *digitalWrite(2,HIGH);* *delayMicroseconds(1000);* *digitalWrite(2,LOW);* *digitalWrite(3,HIGH);* *delayMicroseconds(1000);* *digitalWrite(3,LOW);* *delayMicroseconds(20000);* *}*

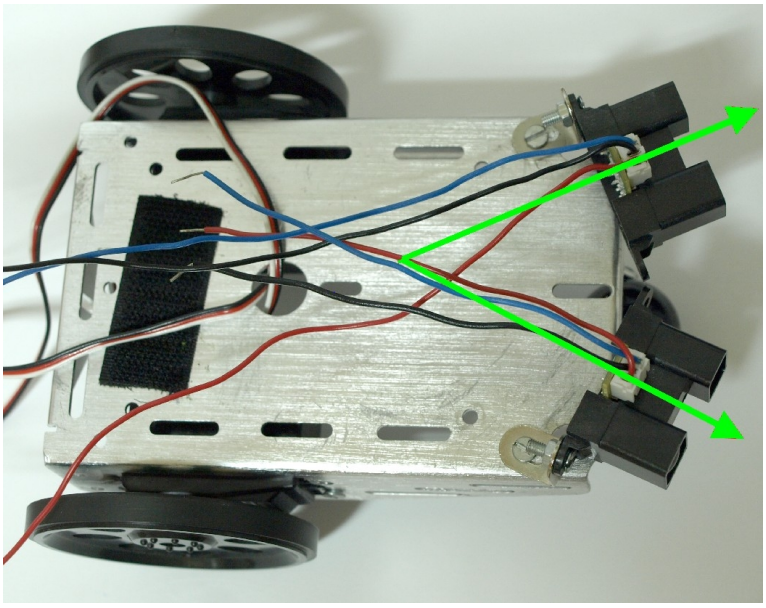
Como podéis observar cada una de estas funciones realiza un único pulso, con lo cual si queremos que el robot ande unos centímetros hacia adelante tendremos que implementar un bucle for que llama a la función adelante tantas veces como tiempo queremos que ande hacia adelante el robot.

```
void setup() {
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
}
void loop() {
int x;
for(x=0;x<100;x=x+1) {
adelante();
}
for(x=0;x<80;x=x+1) {
derecha();
}
for(x=0;x<200;x=x+1) {
adelante();
}
delay(3000);
}
```

Robot que nunca choca.

En esta práctica partiremos del montaje anterior donde teníamos 2 servos conectados a nuestro arduino en los pines 2 y 3. También reutilizaremos las funciones “adelante”, “atras”, “izquierda” y “derecha” para controlar su desplazamiento.

A todo el montaje de la práctica anterior añadiremos un par de sensores de distancia para realizar la navegación autónoma y detectar posibles colisiones con los objetos cercanos. El montaje de los sensores lo haremos en un ángulo de unos 45 grados para asegurarnos que cubrimos posibles puntos muertos.



Respecto al software, nuestra primera implementación será relativamente simple. Escribiremos 4 condicionales y en función de si detectan algo alguno de los sensores o no, se tomará una decisión u otra. En este caso las lecturas de los sensores no se harán ponderadas para acortar el tiempo de reacción. Los valores de "400" no son funcionales para todo tipo de montajes, pueden cambiar en función del sensor y de como esta conectado, dependerá de cada montaje y tendremos que hacer las pruebas pertinentes para detectar el valor correcto.

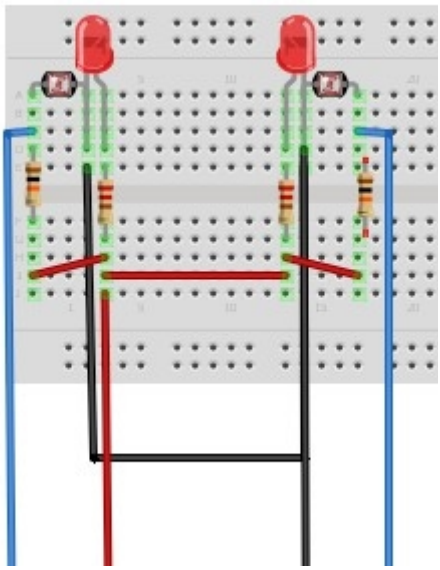
```
void loop() {
valizq = analogRead(0);
valder = analogRead(1);
if (valder>400 && valizq>400) {
atras();
} else {
if(valder>400) {
izquierda();
} else {
if (valizq>400) {
derecha();
} else {
adelante();
}
}
}
}
```

Utilizando esta primera versión del software tenemos un pequeño error, cuando el robot detecta una esquina realiza un pequeño movimiento hacia atrás y acto seguido vuelve a tirar hacia adelante. Una de las posible soluciones es ajustar el valor de uno de los sensores más alto que el otro, de esta manera nuestro robot en caso de duda siempre girará hacia un lado. Otra forma más elegante de solucionarlo sería substituir la llamada a la función "atras" por un bucle que llame a la función "atrás" un número determinado de veces, de esta forma hará un recorrido mayor hacia atrás y después podrá corregir correctamente su rumbo. Otra posible solución es guardar en variables un histórico de hacia donde ha girado las últimas ejecuciones e intentar detectar esos puntos muertos.

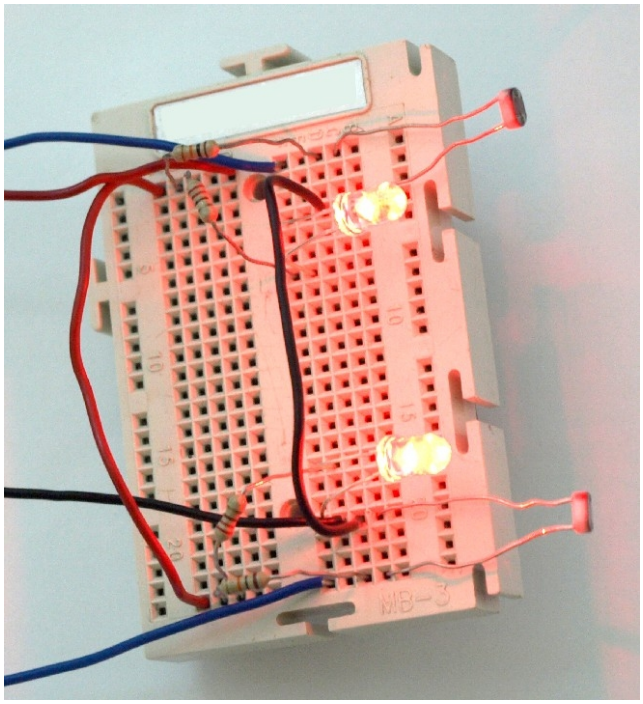
Robot perseguidor.
La idea es realizar un robot que sea capaz de perseguir un objeto en movimiento. Para ello aprovecharemos el montaje de la práctica de la página 81 del libro "Computación Física en Secundaria", y modificaremos el software.
Respecto al movimiento hacia adelante controlaremos la distancia del objeto a tres rangos: lejos, cerca y muy cerca. Si el objeto esta muy cerca iremos hacia atrás para no chocar, si lo tenemos cerca nos aproximaremos él y si lo tenemos lejos nos mantendremos en reposo. Respecto a izquierda y derecha haremos todo lo contrario a la práctica anterior, de forma que si detectamos algo a la izquierda giraremos a la izquierda para encararnos al objeto, y de forma similar si detectamos algo a la derecha giraremos a la derecha para encararnos.

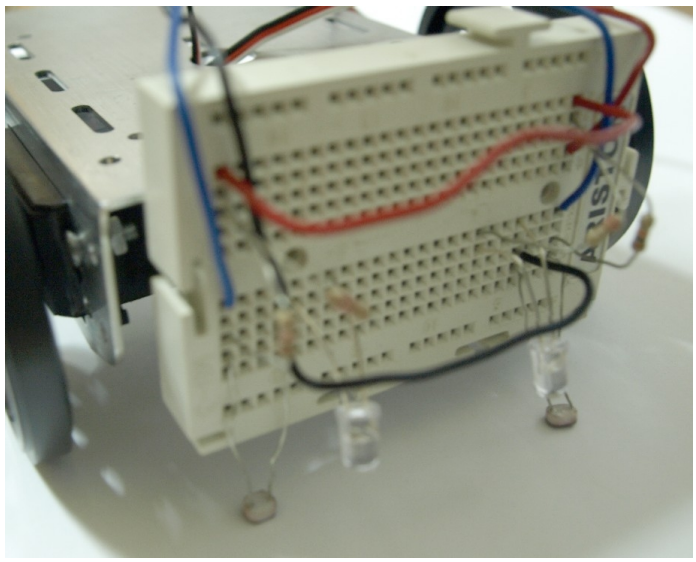
```
void loop() {
valizq = analogRead(0);
valder = analogRead(1);
if (valder>700 && valizq>700) {
atras();
} else {
if(valder>400 && valizq>400) {
adelante();
} else {
if (valizq>400) {
izquierda();
} else {
if(valder>400) {
derecha();
} else {
//No hacemos nada
}
}
}
}
}
```

Robot seguidor de lineas .
En este caso modificaremos el robot de la práctica anterior y en vez de sensores de distancia utilizaremos dos fotorresistencias para determinar el color del suelo por donde pasa el robot. Este montaje es extremadamente sensible a las condiciones lumínicas de donde se realiza, por esta razón utilizaremos un rollo de cintas aislante negra (a ser posible mate) para dibujar la línea en el suelo, e intentaremos que la habitación este bien iluminada. En caso de que tengamos problemas en diferenciar el suelo blanco del negro de la cinta aislante colocaremos un par de leds blancos al lado de las fotorresistencias para mejorar las condiciones de iluminación.



Esquema para el circuito de detección de lineas negras. Los cables Negros se conectan a tierra, los rojos a 5V, y los azules (que nos devuelven el valor de las LDRs) van conectados a los pines AnalogIn.
Recordemos que las fotorresistencias no se conectarán directamente a los pines analógicos 0 y 1, sino que realizaremos un montaje similar al de la página 21 del libro “Computación Física en Secundaria”.





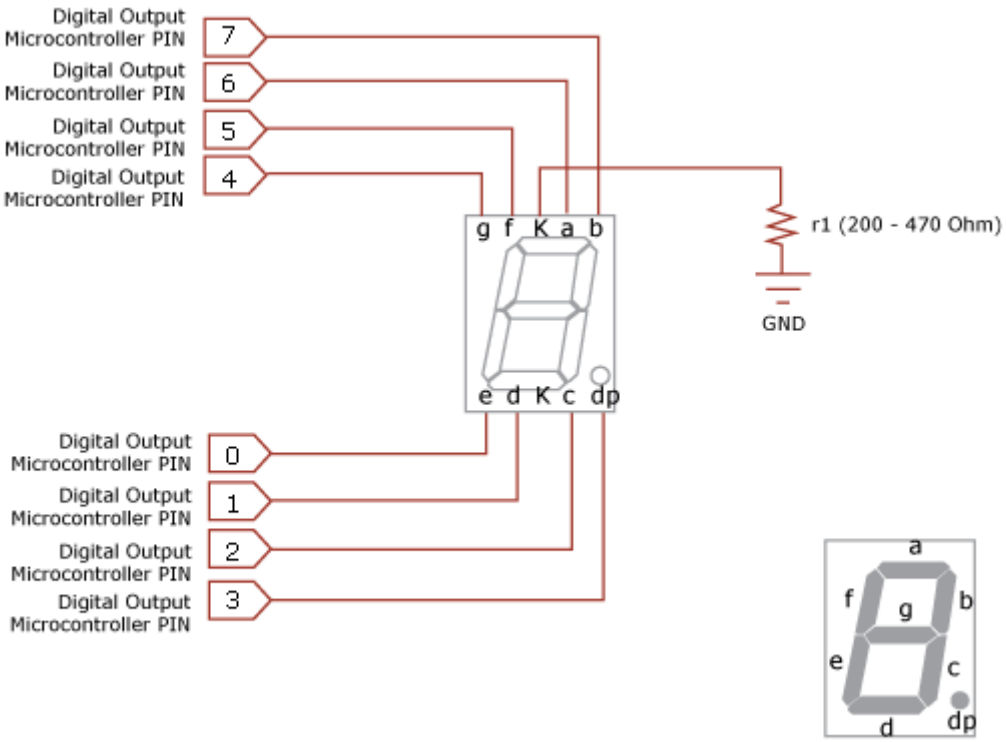
Respecto al software, este vuelve a ser un montaje bastante sencillo. Si las dos fotorresistencias detectan color blanco, podemos deducir que vamos por buen camino y seguimos hacia adelante. Si en caso contrario solo una de las fotorresistencias detecta blanco, significa que nos hemos desviado y debemos corregir el rumbo a derecha o izquierda. Y por último si topamos con negro en los dos sensores deberíamos retroceder un poco para retomar la ruta.

```
void loop() {
valizq = analogRead(0);
valder = analogRead(1);
if (valder>400 && valizq>400) {
atras();
} else {
if(valder>400) {
derecha();
} else {
if (valizq>400) {
Izquierda();
} else {
adelante();
}
}
}
}
```

OTROS ADITAMENTOS

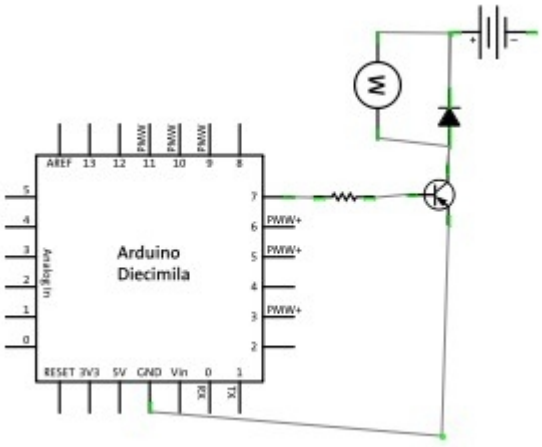
Display Siete Segmentos .

El display siete segmentos es un dispositivo que nos permite mostrar un dígito en una pantalla con 7 leds de forma alargada. Tal y como se puede apreciar en la ilustración hemos de conectar el pin “K” a una resistencia de 200-400 Ohms que va conectada a tierra, el resto de pins los conectaremos a las salidas que queramos utilizar.



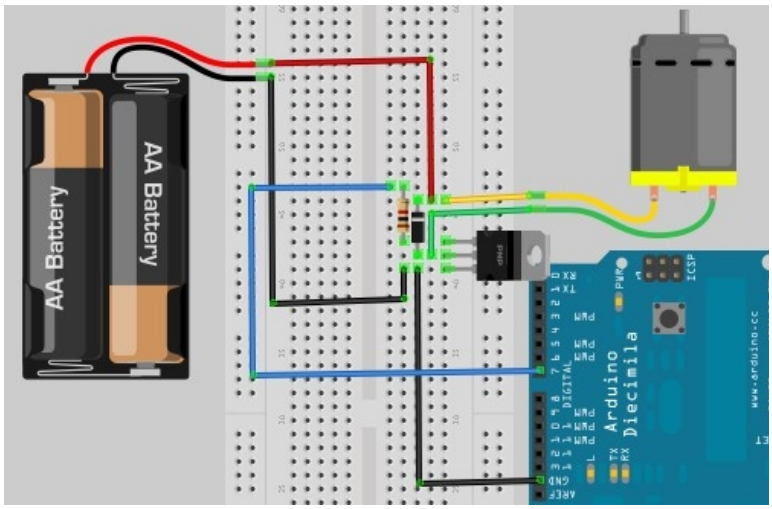
Motor DC con TIP120 .

Al conectar un motor sencillo de corriente continua (motor DC) a nuestra placa necesitaremos añadir algo de circuitería extra y una fuente de alimentación externa.

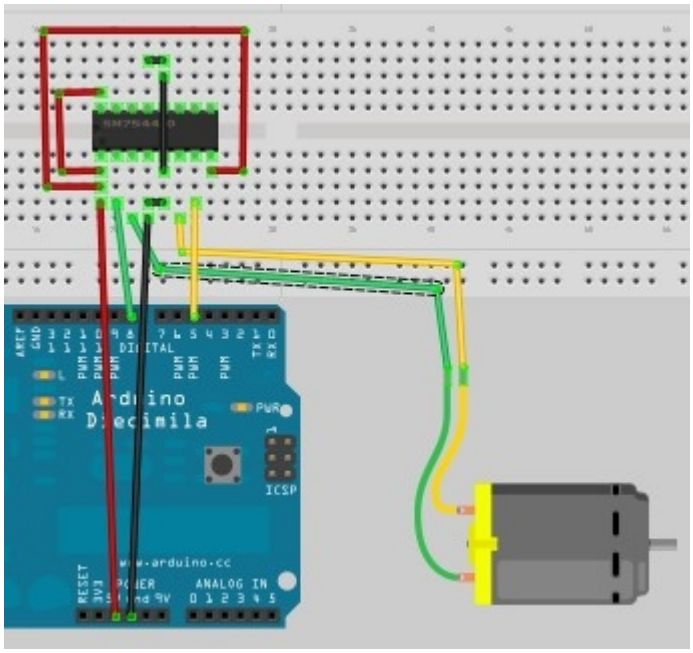


En este caso añadiendo una resistencia de 1kOhmios, un diodo y un TIP120, y un par de pilas AA podemos controlar un motor de continua conectado a cualquier salida digital, los inconvenientes de este montaje es que solo podremos hacer rodar el motor hacia un lado.

```
void setup()
{
  pinMode(2);
}

void loop()
{
  digitalWrite(2,HIGH);
  delay(150);
  digitalWrite(2,LOW);
  delay(150);
}
```

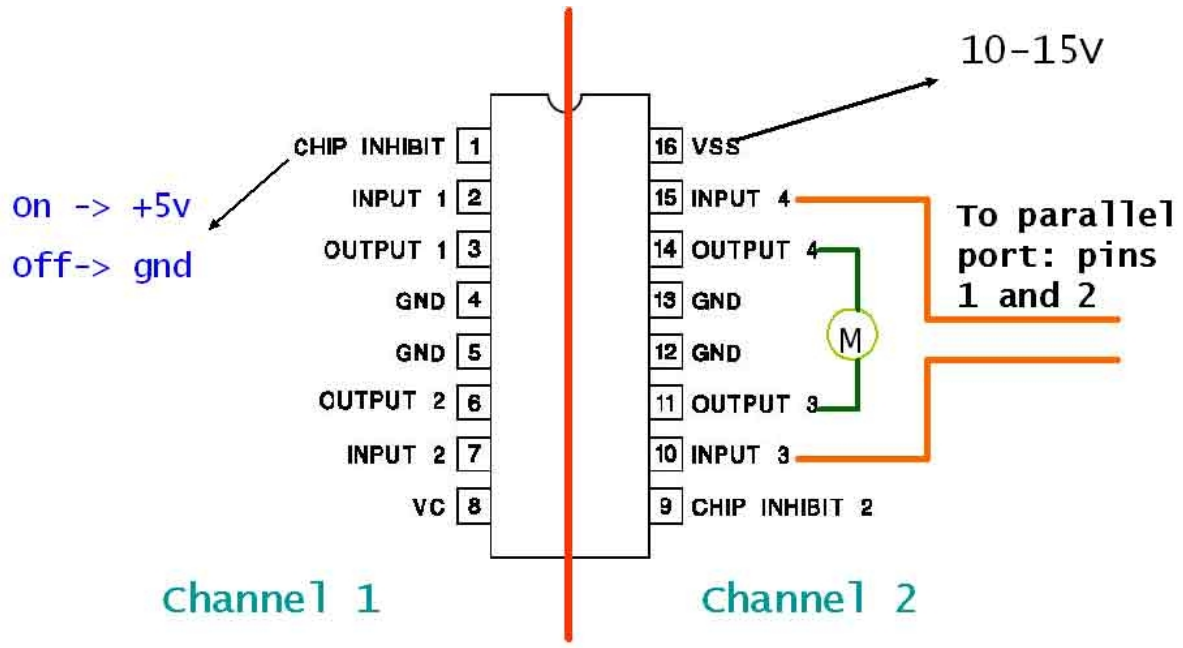


Motor DC con driver L293.
Si invertimos la polaridad de un motor de continua (motor DC) conseguimos que gire hacia el lado contrario. Para poder hacer esto y para poder suministrar la alimentación adecuada a este tipo de motores, la solución ideal es utilizar lo que se conoce como un driver de motores DC. Este driver es un chip que alimenta los motores a un voltaje diferente (3V en nuestro caso), y mediante 2 entradas digitales controlaremos hacia donde gira el motor.
El chip que utilizaremos (L293D o L293NE) nos permite controlar 2 motores de continua conectando 4 salidas digitales de nuestra placa (2 para cada motor). De estas dos salidas de cada motor si ponemos la primera a HIGH y la segunda a LOW girará hacia un lado y si lo hacemos a la inversa girará hacia el lado contrario.



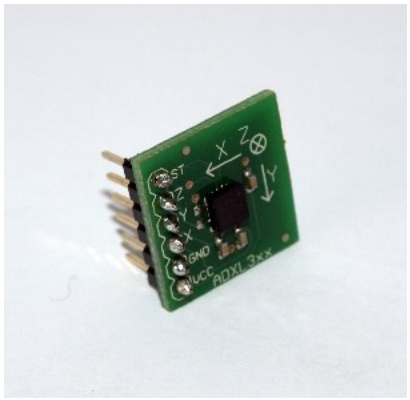
```
void setup() {
  pinMode(5,OUTPUT);
  pinMode(8,OUTPUT);
}
void loop() {
  // Gira hacia la derecha
  digitalWrite(5,HIGH);
  digitalWrite(8,LOW);
  delay(300);
  // Gira hacia la izquiera
  digitalWrite(8,HIGH);
  digitalWrite(5,LOW);
  delay(300);

}
```

Chip L293D/B(puente H):
Es un circuito integrado o chip, que puede ser utilizado para controlar simultáneamente la velocidad y dirección de dos motores de continua (contiene dos puentes H). La diferencia entre el modelo L393D y L293B, es que el primero viene con diodos de protección que evita los daños producidos por los picos de voltaje que puede producir el motor.



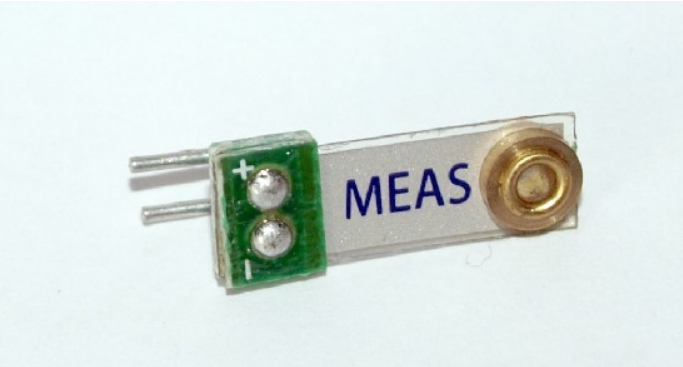
Contiene 4 pines digitales (2,7,10, 15) para controlar la dirección de los motores.
Los pines "enable" (1,9) admiten como entrada una señal PWM, y se utiliza para controlar la velocidad de los motores con la técnica de modulación de ancho de pulso.
Los motores van conectados entre uno de los pines 3, 6, 11, o 14.
La tensión Vss es la que alimentará o dará potencia al motor.

Acelerómetro.
Este tipo de sensores se ha popularizado últimamente por su extensivo uso en videoconsolas, teléfonos móviles y reproductores de música. Son sensores que nos envían mayor o menor voltaje en sus salidas en función de la aceleración (y por tanto también su inclinación) en 2 o 3 ejes (x-y o x-y-z esto depende del sensor). En el mercado podemos encontrar múltiples versiones de estos sensores, nosotros utilizamos los “adxl 3xxx”. Dado que estos sensores son algo más caros hemos de prestar atención a la hora de conectarlos para no dañarlos. En el caso del sensor “adxl 3xxx” conectaremos el pin llamado GND a tierra y el Vin o Vcc a 5V el resto de salidas las podemos conectar directamente a las entradas analógicas y leer los valores que nos dan con la función “analogRead” de arduino.



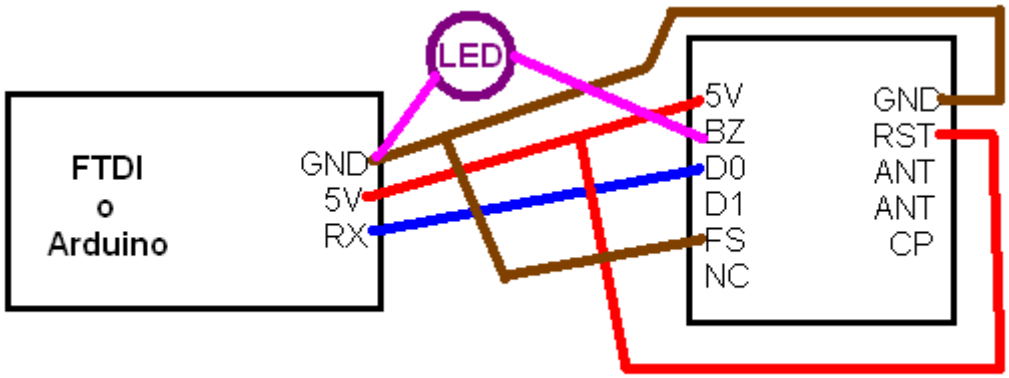
Estos sensores acostumbran a dar valores muy diferentes en función del modelo que utilizamos así que es recomendable estudiar los valores que nos envían antes de montar ninguna aplicación,

también es recomendable hacer servir la técnica de lectura de ponderación de entradas explicada en la página 46 del libro “Computación Física en Secundaria”.



Sensor de torsión y vibración.

Sensor RFID (IDentificación por RadioFrecuencia).
La identificación por radiofrecuencia es una tecnología muy utilizada en el mundo de la logística. Unos diminutos chips (a veces del tamaño de micras) tienen un código independiente cada uno y al pasarlos por delante de un sensor RFID estos chips envían un código único de 128 bits. Estos diminutos chips los tenemos en tarjetas de identificación, chapas, etiquetas o llaveros. Y sean del fabricante que sean, al pasar por delante de un sensor de RFID, este capta el código de identificación. Nosotros utilizaremos el sensor ID-12 que podemos adquirir en sparkfun por ejemplo.



En nuestro caso conectaremos el sensor ID-12 a un chip FTDI conectado directamente al puerto USB de nuestro PC para después recibir los datos directamente en processing.

```
import processing.serial.*;
Serial puerto;
String datos="";
void setup() {
size(600,200);
println(Serial.list());
puerto=new Serial(this, Serial.list()[1], 2400);
puerto.buffer(12);// El puerto [1] depende en función del PC
}
void draw() {
background(0);
```

```
println(datos);
}
void serialEvent(Serial puerto) {
datos=trim(puerto.readString());
}
```

Basado en el libro “Computación Física en Secundaria” de *Marco Antonio Rodríguez Fernández*

Basado en la wev http://arduino.cc/es_old