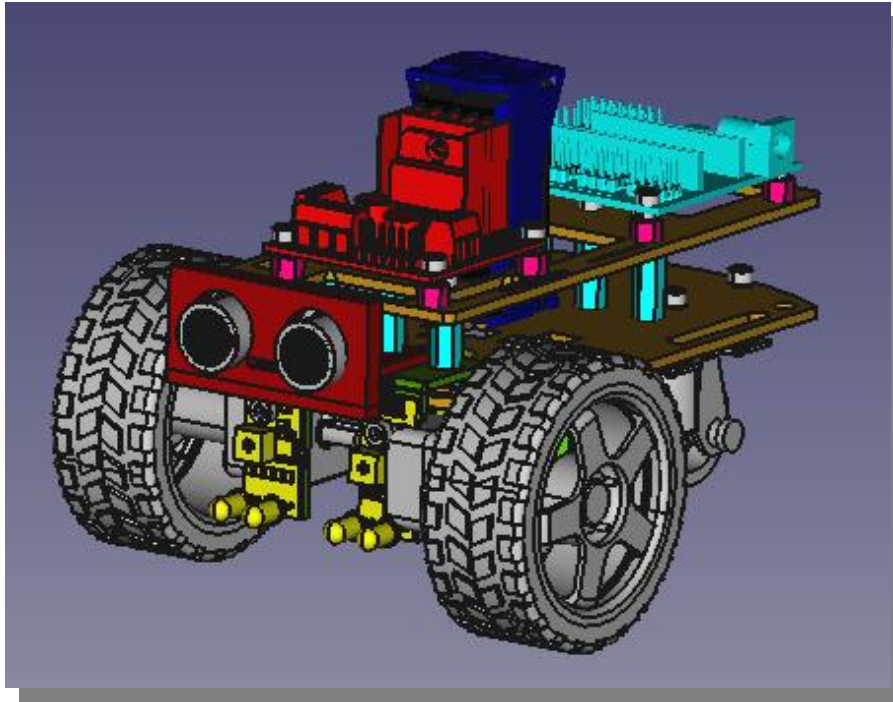


MASAYLO

Robot educativo Open Source

Manual básico de montaje y programación



Antonio Gómez García
M.^a Dolores Noguerras Atance



ÍNDICE

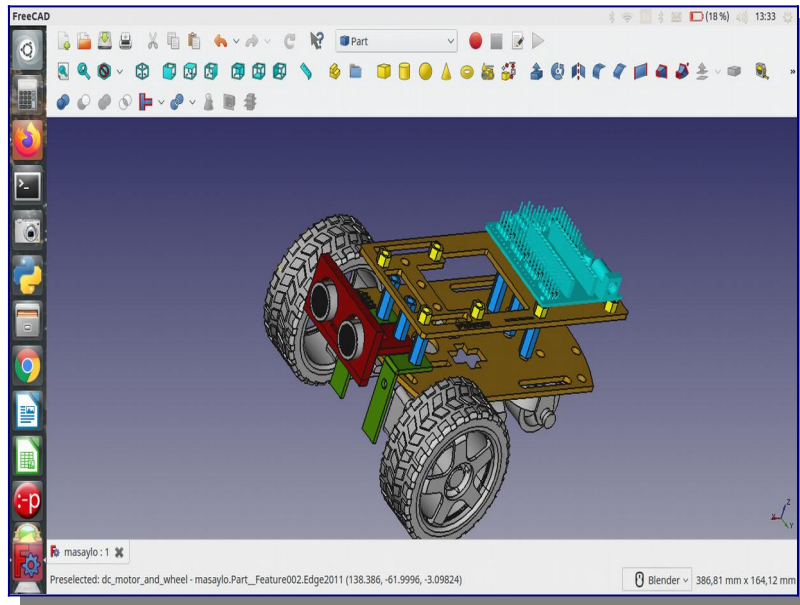
Sumario

Robot educativo Masaylo.....	7
Apariencia general.....	7
Piezas, tornillería y circuitos necesarios.....	8
Piezas impresas en 3D.....	8
Electrónica.....	9
Electrónica imprescindible.....	9
Sensores opcionales.....	10
Otros.....	11
Instrucciones de montaje.....	12
Fijar motores a la base.....	12
Piezas.....	12
Proceso.....	13
Montar sensores y segundo piso.....	14
Piezas.....	14
Proceso.....	15
Fijación de la electrónica.....	18
Piezas.....	18
Proceso.....	19
Fin del montaje.....	20
Conexionado (pinout) de Masaylo.....	22
Algunos matices a tener en cuenta.....	22
Electrónica imprescindible y circuitos opcionales.....	22
Pines de Arduino.....	25
Conexión de la alimentación.....	25
Alimentación de Masaylo. Uso del L298N.....	26
¿Por qué no podemos controlar motores DC desde Arduino?.....	26
El circuito L298N.....	27
Alimentación de Arduino a través del L298N.....	28
Movimiento básico del Masaylo.....	30
Consideraciones iniciales.....	30
Mi primer programa: mover a Masaylo hacia adelante.....	30
Uso de funciones.....	31
adelante().....	32
atras().....	32
izquierda().....	33
derecha().....	33
alto().....	34

Robot educativo Masaylo

Estoy intentando desarrollar un robot compatible con Arduino impreso en 3D, Masaylo, de tipo Low-Cost, propulsado por motores DC de los típicos que podemos encontrar en cualquier Aula-Taller de Tecnología. A medida que progreseemos intentaremos dotar al proyecto de una estructura algo más modular y coherente.

Apariencia general



Utilizaremos: un Arduino nano con su correspondiente shield de expansión, un módulo L298N para gobernar dos motores DC, una rueda loca de 25x13 cm (o similar), un módulo sensor de ultrasonidos de tipo HC-SR04, dos sensores de infrarrojos de tipo FC-51 y un módulo BT modelo HC-05 o HC-06 que posiblemente habrá que tunear mediante comandos AT (o no; lo fundamental es saber a qué velocidad vamos a comunicarnos con él, y el código original espera que trabajemos a 19200 baudios; si es más cómodo para vosotros, podéis modificar el código para que trabaje a 9600 baudios).

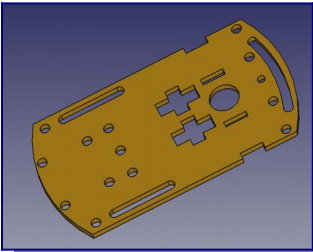
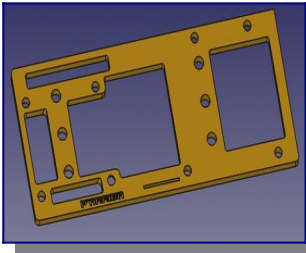



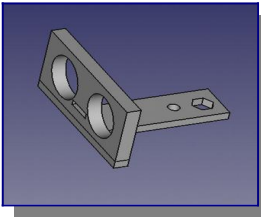
Sobre los códigos mostrados en el manual

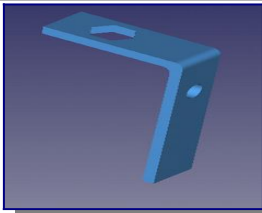
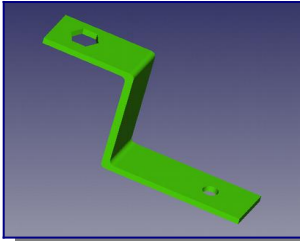
Todos los códigos e imágenes mostrados en el presente manual son de libre uso para el lector bajo la licencia Open Source correspondiente, y están en el repositorio de los autores:

<https://github.com/agomezgar/masaylo>

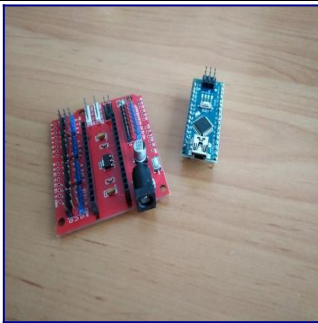

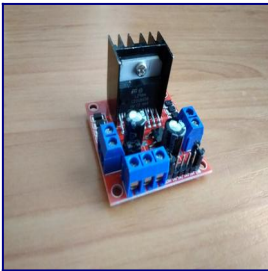
Piezas, tornillería y circuitos necesarios

Piezas impresas en 3D

Pieza impresa en 3D	Imagen
Base	
Segundo piso	
Separadores(pasatornillos) entre base y segundo piso (x6)	
Separadores (pasatornillos) para shield de Arduino y módulo L298 N (x8)	
Colocadores/fijadores de motores a base	
Soporte de sensor de ultrasonidos	

Pieza impresa en 3D	Imagen
Soporte de sensor de infrarrojos FC51 (si tiras de sensores baratos)	
Soporte de sensor de infrarrojos TCRT5000 (si los prefieres a los FC51)	

Electrónica

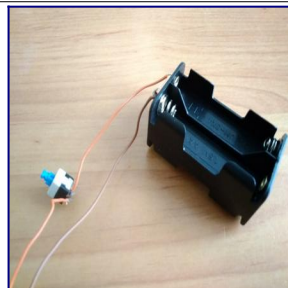
Electrónica imprescindible	
Arduino nano con shield de expansión v3.0	
Micromotores DC modelo TT con reductora y rueda(x2)	
Módulo L298N para control de motores DC	

Electrónica imprescindible

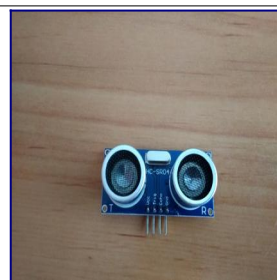
Interruptor de 8 mm o similar



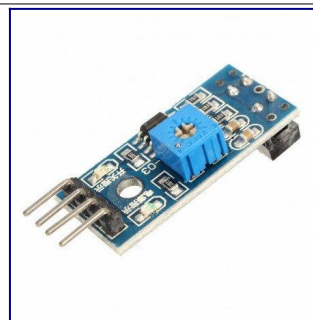
Portapilas 4xAA
(el portapilas 4xAAA también sirve)
Soldar el interruptor al polo positivo del portapilas

**Sensores opcionales**

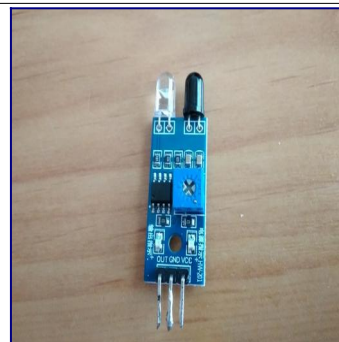
Sensor de ultrasonidos HC-SR04



Sensor de infrarrojos TCRT 5000
(Primera opción)

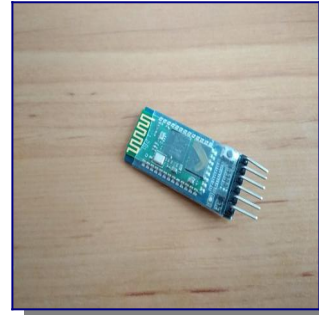


Sensor de infrarrojos FC51
(Más barato, es el que utilizaremos en los tutoriales)





Sensores opcionales

Módulo Bluetooth HC05 ó HC06
(En aquellos proyectos en que se utilicen, será MUY IMPORTANTE vigilar la velocidad a la que vamos a comunicarnos con dicho módulo, 9600 ó 19200 baudios)
No es exactamente un sensor, sino un módulo de comunicación inalámbrica, pero, ¡bueno!. No nos pondremos tiquismiquis, ¿no?.



Otros

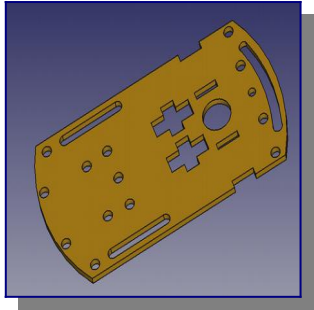

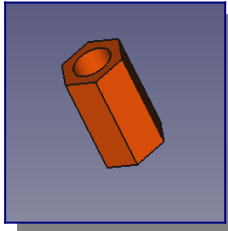

Descripción	Imagen
Rueda loca universal para coches, de nylon 23x15	
Tornillos y tuercas varios de tipo M3, de 12, 16 y 35 mm	


Instrucciones de montaje

Según nuestra experiencia, es muy común que a la hora de iniciar cualquier proyecto, por más esfuerzos que haga el estudiante, fallará algo, o no se dispondrá de alguna pieza o herramienta, o no es exactamente del modelo específica. ¡No pasa nada!. Murphy está siempre presente cuando hablamos de Robótica Educativa. No hay que preocuparse. Intentemos seguir las instrucciones, y cuando algo no salga como estaba previsto, haremos lo que hacemos siempre, ¡improvisar!. ¡Vamos allá!

Fijar motores a la base

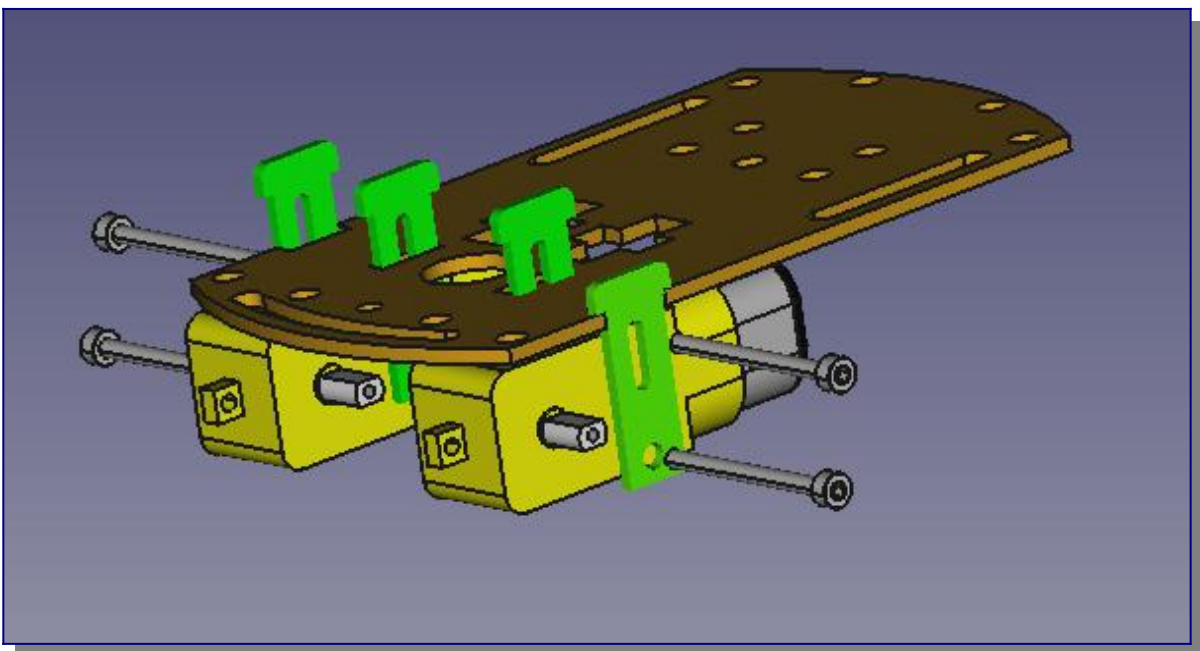
Piezas

Nombre	Imagen
Base impresa en 3D	
Colocadores (x4)	
Separadores (pasatornillos) para rueda loca (x4) (la pieza en STL se llama "separaRueda(x4)")	
Motor DC modelo TT con motoreductora y rueda	

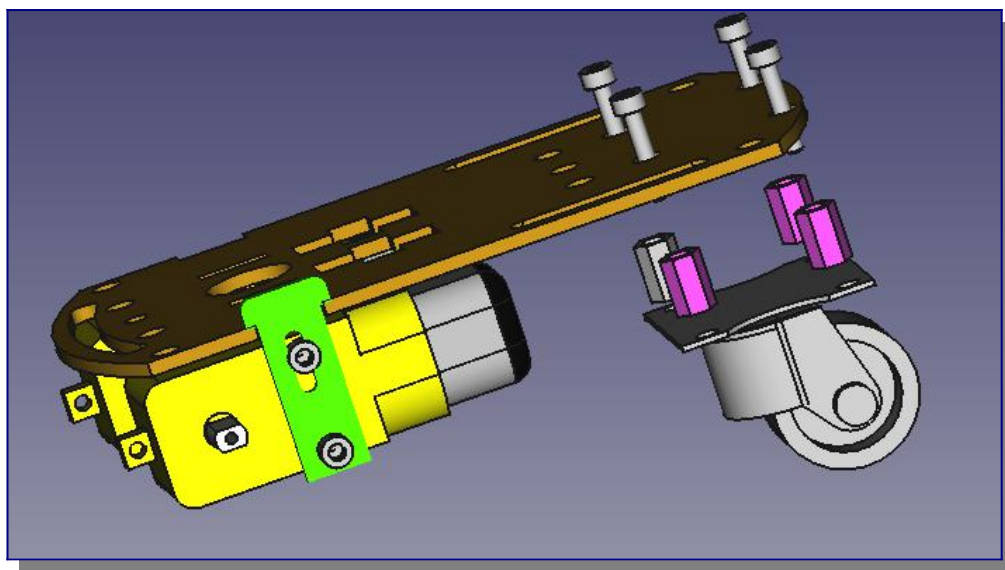
Nombre	Imagen
Rueda loca universal para coches	
4 tornillos M3x35 mm para atornillar los motores a la base	
4 tornillos M3x16 mm para fijar la rueda loca a la base	
8 tuercas M3	

Proceso

a) Fijar motores a la base y atornillar, asegurando que el eje mire hacia la parte delantera del robot y los terminales de conexión miren hacia fuera



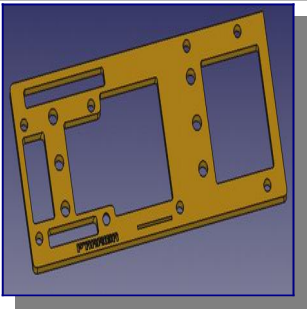

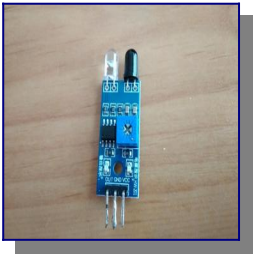
b) Fijar rueda loca a la base mediante tornillos M3x16 mm a través de los pasadores



Montar sensores y segundo piso

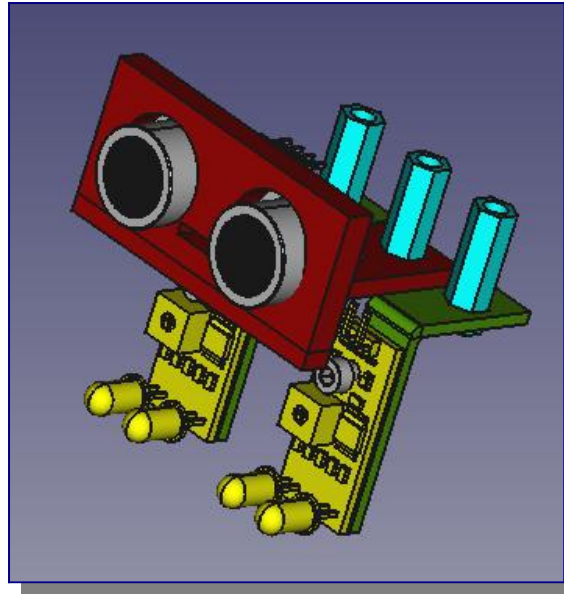
Piezas

Nombre	Imagen
Separadores(pasatornillos) entre base y segundo piso(x6) (la pieza en STL se llama "SEPARADOR(x6)")	
Soporte para sensor de ultrasonidos	
Soporte para sensor IR (modelo FC-51 ó TCRT5000)(x2)	

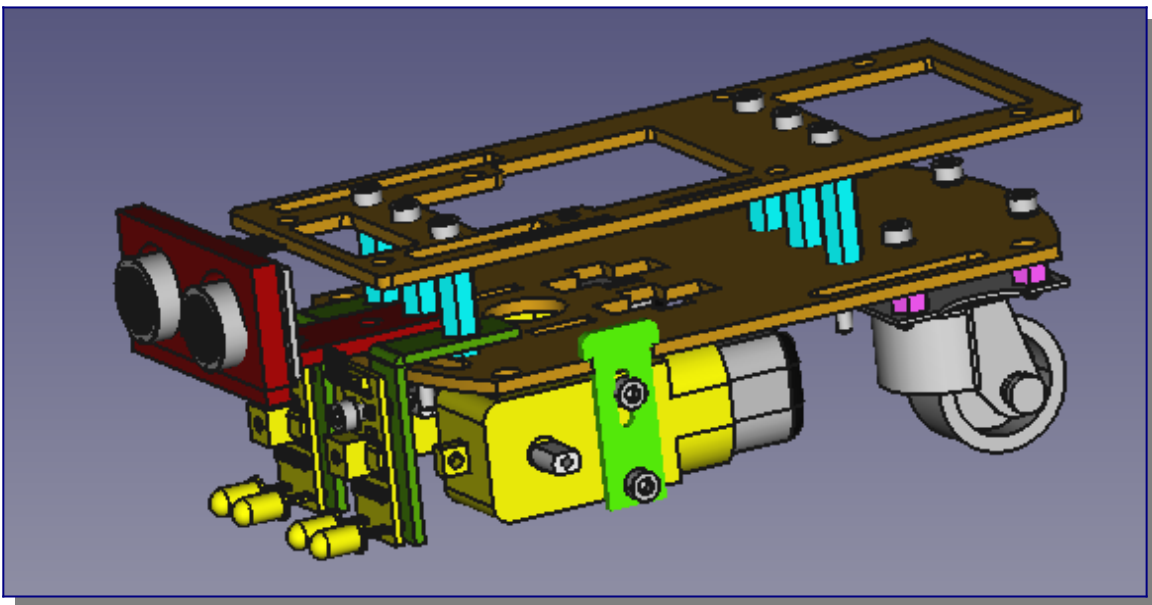
Nombre	Imagen
Segundo piso del robot	
Sensor de ultrasonidos HC-SR04	
Sensor de infrarrojos (x2)	
6 tornillos M3x35 mm para fijar el segundo piso a la base	
3 tornillos M3x12 mm para atornillar los sensores a sus soportes	
9 tuercas M3	

Proceso

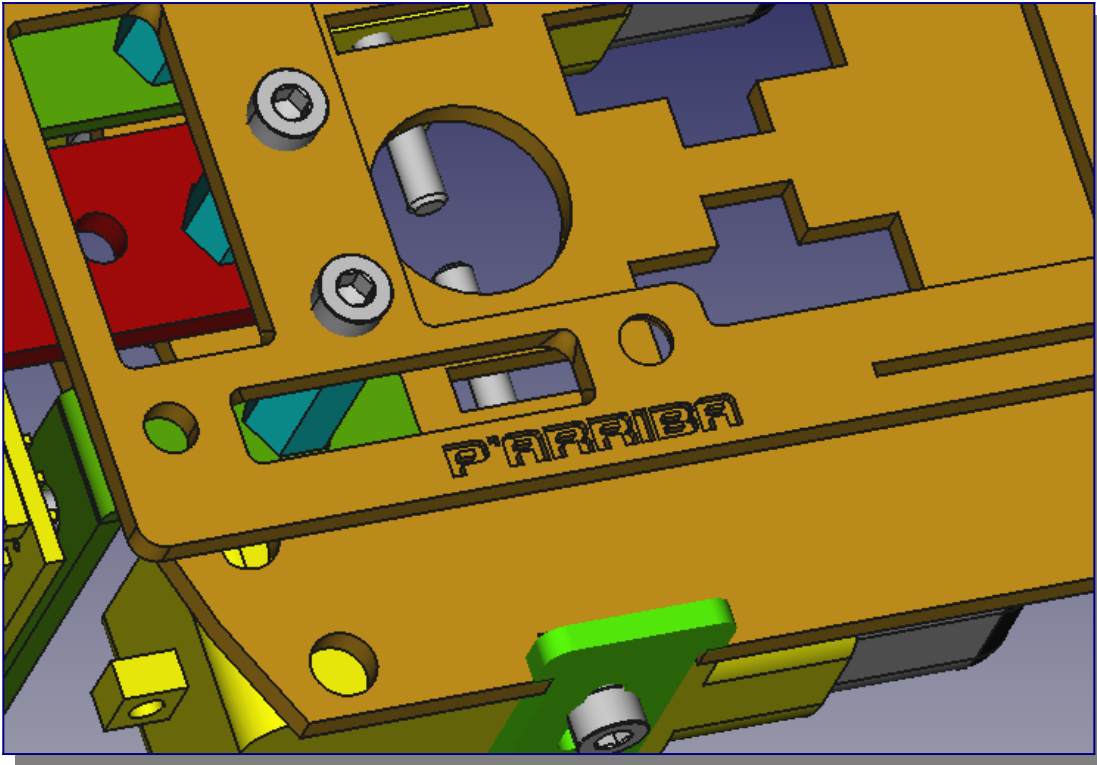
a) Montar los sensores en sus soportes. Cada soporte estará atravesado por un separador y se prepararán para colocarlos en la parte delantera del robot



b) Atornillar la pieza llamada "Segundo Piso" a la base utilizando los seis separadores



Nota: Al contrario que con la base, el segundo piso tiene orientación. Hay un pequeño serigrafiado, algo borroso, que pone: "P'ARRIBA". Imagínate para qué sirve.



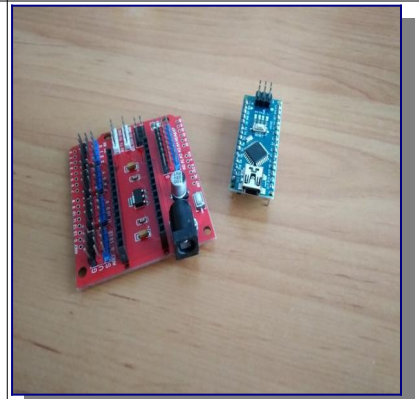
Fijación de la electrónica

Piezas

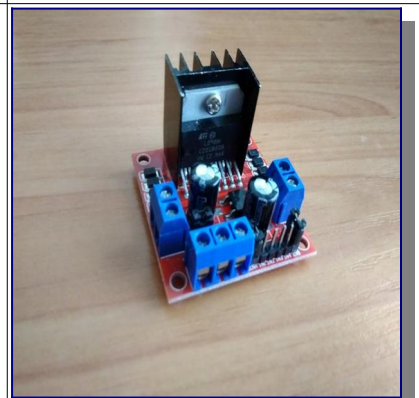
Separadores para la electrónica
(la pieza en STL se llama "separadorSec(x8)")


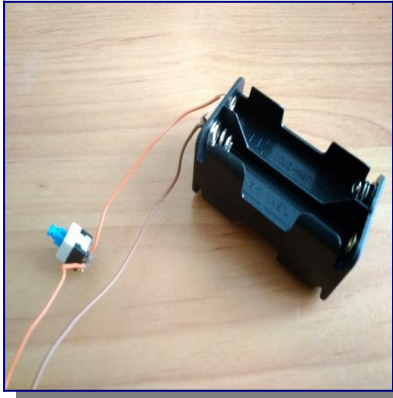


Tarjeta Arduino Nano montada sobre shield de expansión V3



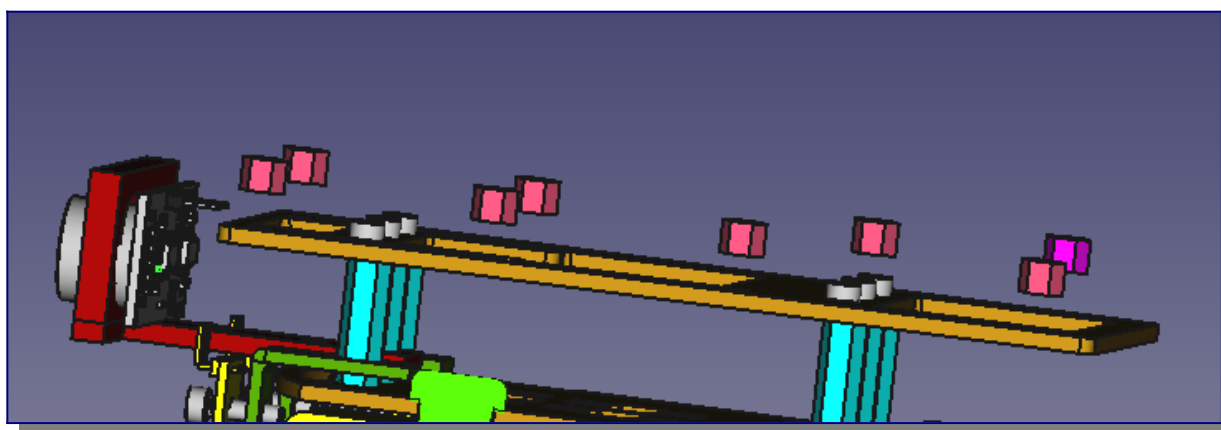
Módulo de control de motores DC L298N



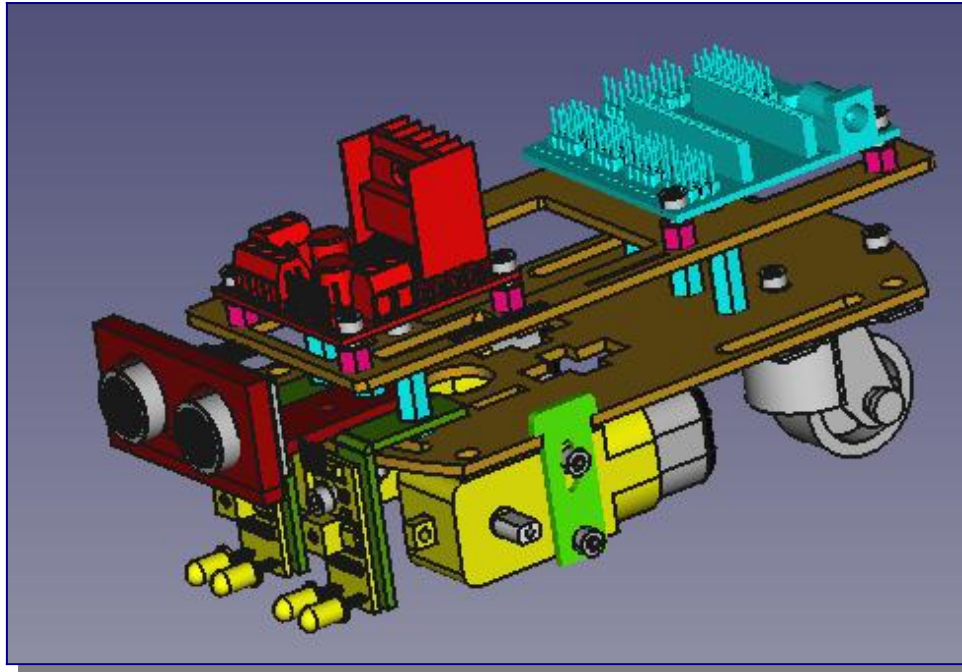
<p>Separadores para la electrónica (la pieza en STL se llama "separadorSec(x8)")</p>	
<p>Portapilas 4xAA con interruptor conectado a cable de positivo</p>	
<p>8 tornillos M3x12 mm para atornillar los módulos al segundo piso</p>	
<p>8 tuercas M3</p>	

Proceso

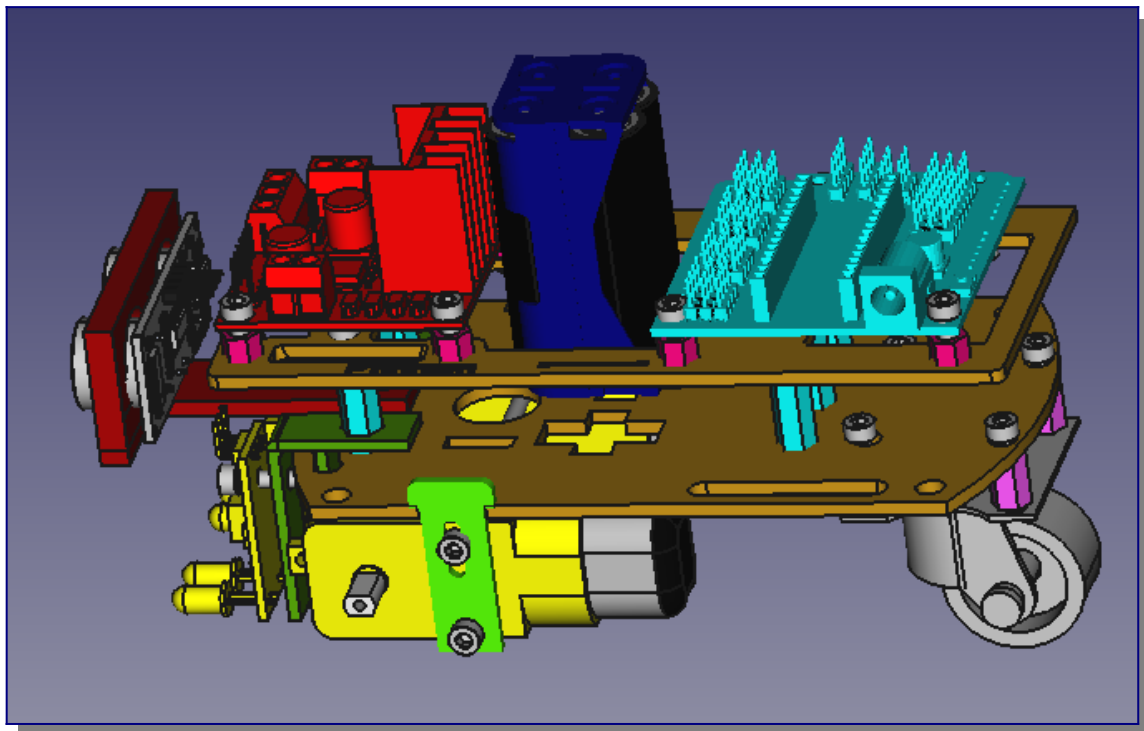
a) Colocar los separadores frente a sus respectivos taladros en el segundo piso



b) Fijar ambos circuitos al segundo piso con los tornillos M3x12 mm y sus respectivas tuercas

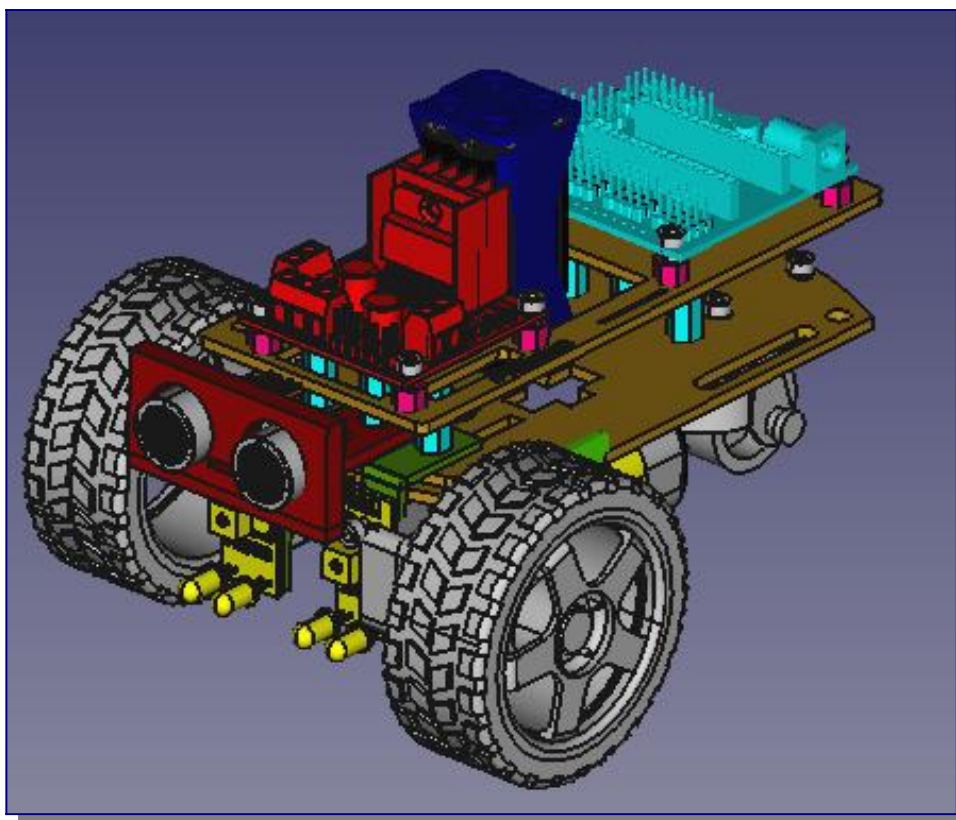


Poner el portapilas en el hueco que queda entre la shield de Arduino y el módulo L298 N



Fin del montaje

Colocamos las ruedas en los motores y, ¡TACHÁÁÁÁN!



Conexionado (pinout) de Masaylo

Algunos matices a tener en cuenta

Masaylo es un robot educativo cuyo único fin es ayudar a comprender al estudiante o aficionado algunos de los aspectos básicos de la Robótica y el Automatizado y Control Electrónico que hoy en día se trabajan en múltiples unidades didácticas dentro de materias relacionadas con la Tecnología tanto en Educación Secundaria como en Primaria.

Hoy en día hay muchos y muy bien diseñados robots educativos relacionados con Arduino y con características Open Source (Escornabot y OttoDIYs son dos de los primeros ejemplos que vendrán a la mente de muchos lectores). Si nos hemos atrevido a presentar nuestro propio proyecto a este respecto es porque creemos que hay, entre otros, dos puntos a su favor:

- Es un robot de muy bajo coste, a la par que sólido y de construcción muy simple.
- Se propulsa mediante motores DC de los más comúnmente utilizados en cualquier taller de Tecnología de cualquier instituto de Educación Secundaria.

El uso de motores DC, comúnmente más fáciles de encontrar que los stepper y los servomotores de aeromodelismos como los SG-90, son además muy baratos y mucho más sólidos que sus antagonistas. Pero su principal ventaja es que el adolescente o niño que empieza a aprender a programar un robot como Masaylo comprende mucho mejor el control de un motor DC que el de un motor paso a paso o un servomotor, que exigen el uso de pulsos de tipo PWM o enviar secuencias de control a través de varias patillas del microcontrolador. Un motor DC, en cambio, gira con mucha potencia en un sentido o en otro dependiendo de la polaridad de la alimentación que se le proporcione.

Electrónica imprescindible y circuitos opcionales

Como ya se ha comentado, Masaylo necesita únicamente una tarjeta de control de tipo Arduino Nano insertada en una shield de expansión y un módulo de control de tipo puente en H como el L298N para controlar los motores de corriente continua (además, por supuesto, de dichos motores). Los sensores de ultrasonidos y de infrarrojos, además del módulo Bluetooth que se proponen, son no sólo opcionales, sino incluso sustituibles por otros sensores o actuadores a gusto del programador.

Empezaremos aprendiendo a programar las salidas digitales de Arduino mediante el uso de funciones para introducir en el robot la secuencia de movimientos que deseamos. A partir de ahí, todo lo que venga después dependerá exclusivamente de nuestra imaginación.

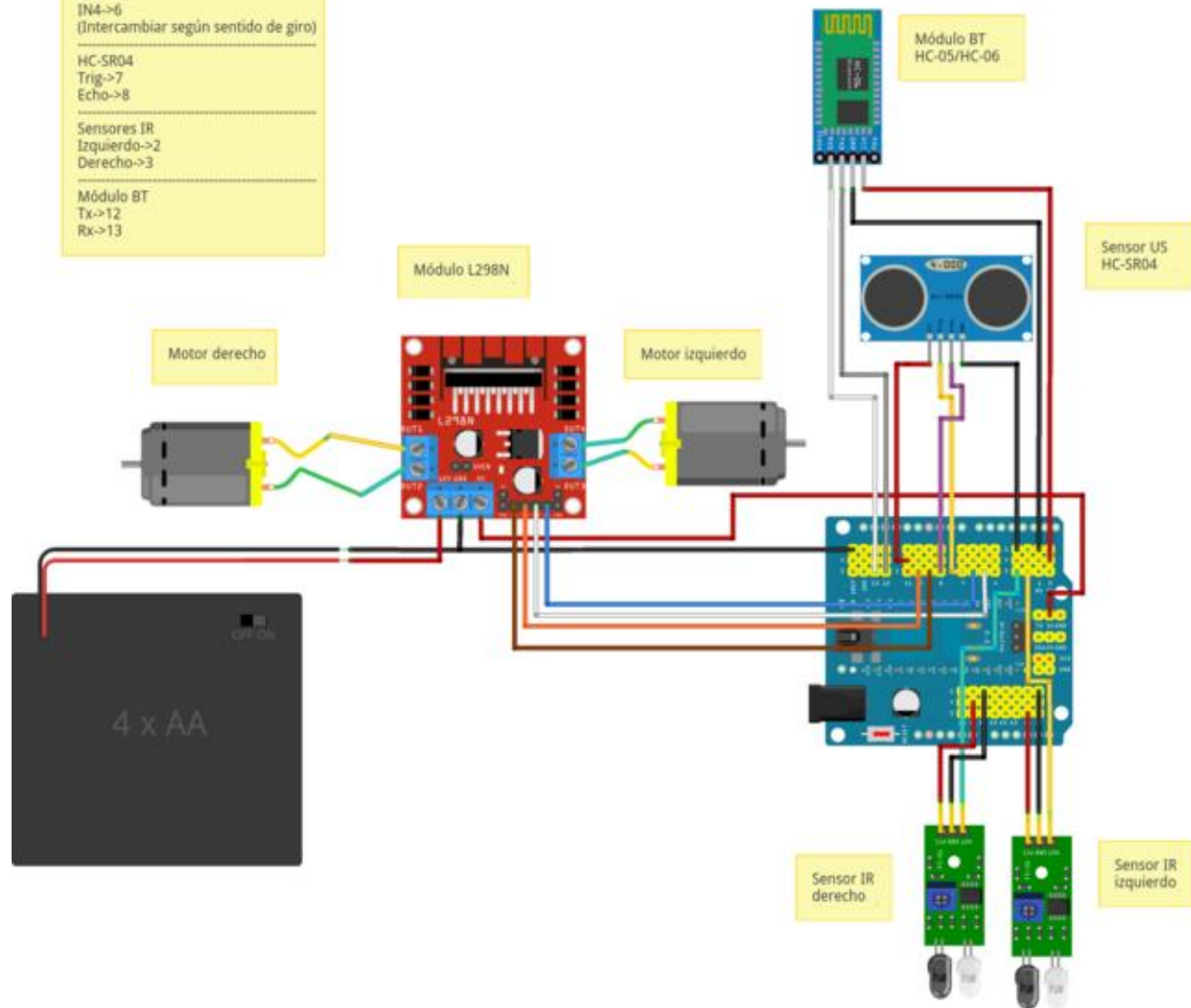
El Masaylo está diseñado para un portapilas de 4 pilas tipo AA, pero incluso este aspecto constructivo queda a disposición de la imaginación y la capacidad de improvisación del estudiante que se disponga a llevar a cabo el proyecto.

L298N
 IN1->9
 IN2->10
 IN3->5
 IN4->6
 (Intercambiar según sentido de giro)

 HC-SR04
 Trig->7
 Echo->8

 Sensores IR
 Izquierdo->2
 Derecho->3

 Módulo BT
 Tx->12
 Rx->13



Pines de Arduino

El programador algo más experto puede optar por su propia elección de pines, sea por comodidad o por que le parezca más acertada. Al utilizar, en el planteamiento de esta primera versión del robot, únicamente pines digitales, esto no debería plantear ningún problema. No obstante, en los próximos capítulos, los códigos que incluiremos harán referencia a este pinout:

ELEMENTO DE CONTROL	PIN/PINES DIGITALES
Motor izquierdo	5 y 6
Motor derecho	9 y 10
Sensor infrarrojo izquierdo	2
Sensor infrarrojo derecho	3
Sensor de ultrasonidos HC-SR04	Trigger → 7 Echo → 8
Módulo de comunicaciones Bluetooth	Tx → 12 Rx → 13

Quedan algunos pines sin usar (0 y 1 normalmente se obvian porque son los que utiliza el ordenador para comunicarse con Arduino). Queda a la imaginación del lector posibles usos para ellos.

Conexión de la alimentación

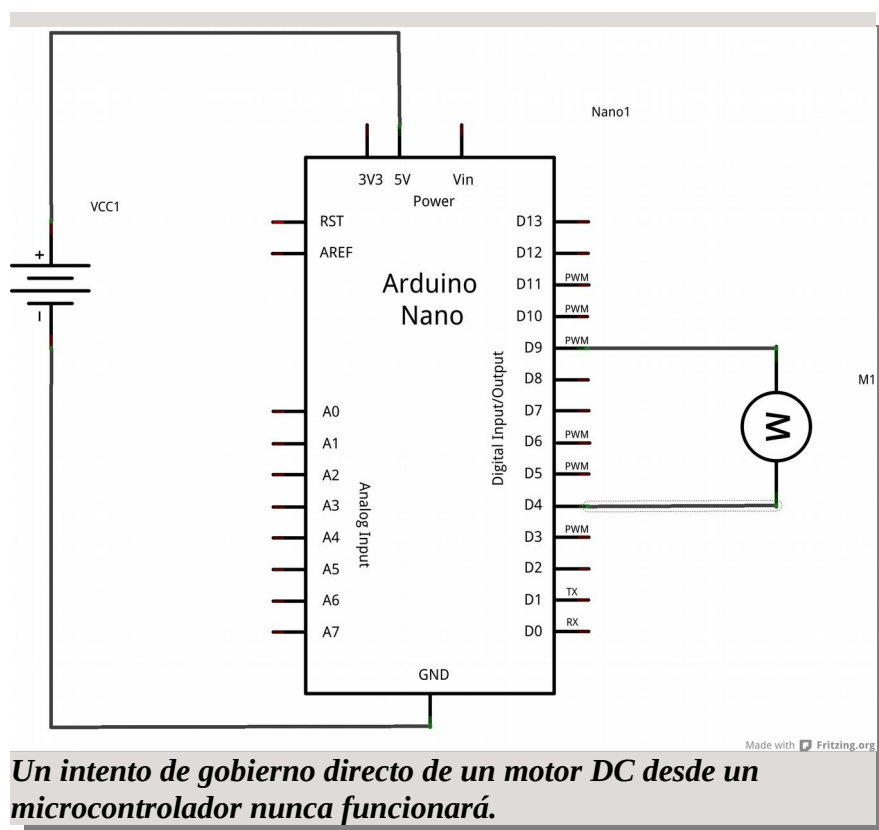
Quizás sorprenda un poco al inexperto que, a diferencia de los montajes a que estamos acostumbrados, el polo positivo de la alimentación no se conecta a +5V ni al pin Vin de la Arduino, sino a la patilla +12 V del L298N. Esto se explicará en el próximo capítulo. Quede también **MUY CLARO QUE LAS TIERRAS (PINES GND) DE AMBOS CIRCUITOS, ARDUINO Y L298N, DEBEN ESTAR INTERCONECTADAS ENTRE SÍ.**

Alimentación de Masaylo. Uso del L298N

¿Por qué no podemos controlar motores DC desde Arduino?

Uno de los mayores escollos que nos encontramos en Robótica Educativa, cuando trabajamos en centros educativos, es el tema de la movilidad del robot. Por diversas razones, el gobierno de un motor desde un microcontrolador se hace normalmente muy difícil, dado que la propia naturaleza de estos dispositivos, basados en la excitación de bobinas mediante electricidad, hace que absorban suficiente corriente eléctrica como para “aturdir” al microchip que intenta manejarlos.

El circuito de la siguiente imagen, por ejemplo, no es funcional. La primera intención del principiante es conectar el motor a dos pines digitales de Arduino (4 y 9 en nuestro ejemplo), con intención de controlarlo alternando “1” y “0” lógicos (esto es, conectando los bornes del motor a +5 V y tierra para lograr el giro del motor en el sentido correspondiente). Pronto se desengañará y comprobará que el motor no obedece.

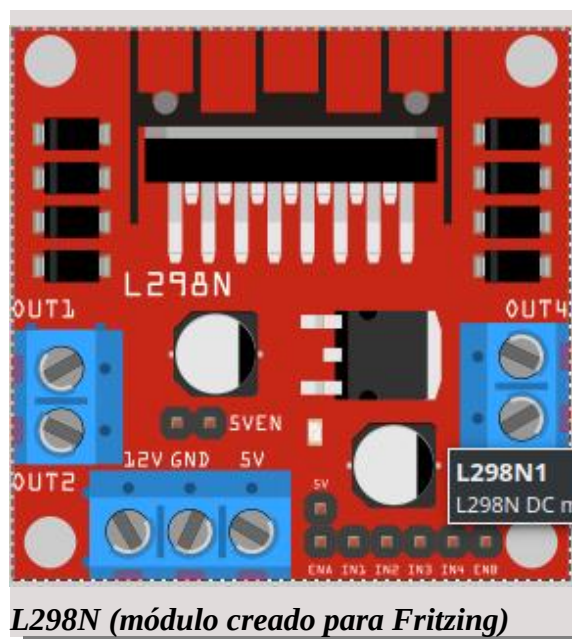


¿Por qué?. Si nuestra tarjeta asocia los estados “HIGH” y “LOW” a 5 V y 0 V, respectivamente, el motor debería moverse. Sin embargo, olvidamos que lo que mueve al motor no es la tensión, sino la intensidad. Y los microcontroladores como los que contiene Arduino pueden mantener las tensiones establecidas siempre que no se les pida una intensidad de corriente por encima de unos pocos miliamperios, muy por debajo de lo que necesita un motor de corriente continua para empezar a funcionar. ¿El resultado? No hay movimiento, y muy probablemente la tarjeta se bloqueará.

Este problema, como todos los demás, tiene diferentes soluciones. Una consiste en buscar otro tipo de motores, como los stepper o “paso a paso”, con menos exigencias a este respecto. Otra consiste en utilizar relés (tampoco es muy funcional por diversas razones). La que nos ocupa en este caso consiste en utilizar una alimentación separada de Arduino (esto es, otra pila o batería), eso sí, con una tierra común (ambos polos negativos deben estar conectados entre sí) y un circuito que hará de intermediario entre la tarjeta y dicha alimentación. Este tipo de circuitos utiliza transistores según el modelo denominado “puente en H”, como es el ejemplo del L293DNE, [sobre el que ya escribimos hace algunos años sobre su posible uso con Arduino](#). Dicho circuito en el caso de Masaylo es el L298N.

El circuito L298N

Elegimos este modelo por su accesibilidad comercial (puede conseguirse muy fácilmente), muy bajo precio y por su adaptabilidad a múltiples condiciones de trabajo en tensiones y modulación de velocidad.



Hay mucha bibliografía en español sobre las características de este módulo ([nosotros recomendamos en particular un excelente artículo sobre el particular en la web de Luis Llamas, www.luisllamas.es](#)), así que no creemos posible añadir nada útil al respecto. Sólo resaltaremos estos puntos:

- El propósito general de este tipo de circuitos es recibir en sus entradas señales lógicas de 5 V que repetirán en sus respectivas salidas con la alimentación que se les da (hasta 35 V), solucionando así el problema de alimentar correctamente a los motores.
- Admite alimentaciones de 6 V hasta 12 V con el *jumper regulador* (nuestro caso), y hasta 35 V si quitamos dicho jumper.
- El uso de *ENA* y *ENB* si quitamos sus jumper permitiría la modulación de la velocidad del motor mediante impulsos *PWM*, *previa desinstalación del mismo jumper regulador*(no lo necesitaremos).
- El borne serigrafiado como +5V funciona como una salida de tensión con los jumper puestos, pero si los quitáramos tendríamos que proporcionar dicha tensión para alimentar la lógica del circuito.
- Por el propio diseño del circuito, entre la tensión que proporcionemos y la que llegue a los motores DC se producirá una pérdida de alrededor de 1,5 V.

Por otro lado, la lógica de conexionado para gobernar hasta dos motores de corriente continua es extremadamente intuitiva:

ENTRADA	SALIDA
IN1	OUT1
IN2	OUT2
IN3	OUT3
IN4	OUT4

Como ya se ha comentado en apartados anteriores, es posible que alguno o ambos motores funcionen al revés de como se les indica (esto es, que giren en sentido antihorario cuando se les pide sentido horario, o al revés). Bastará con intercambiar las correspondientes patillas INx o OUTx (lo que resulte más sencillo).

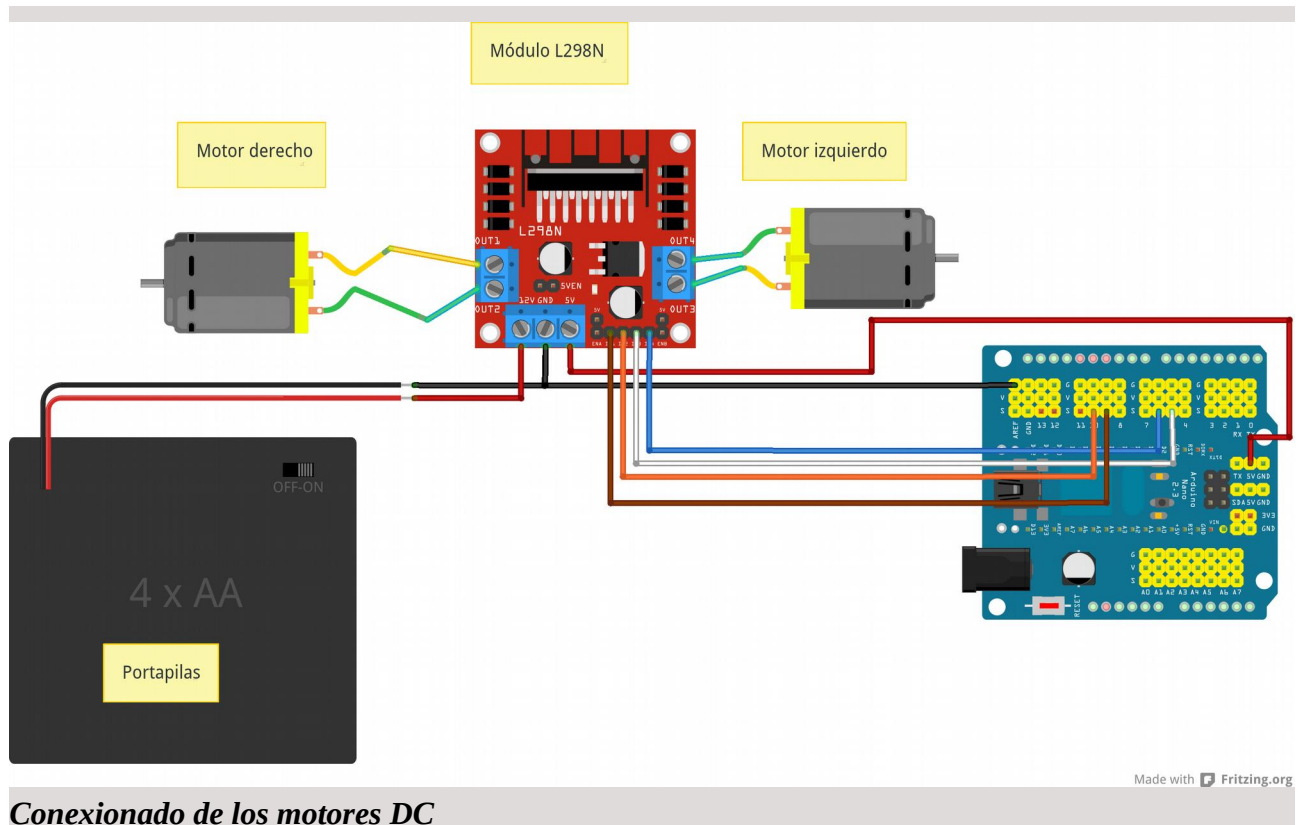
Alimentación de Arduino a través del L298N

Por las razones esgrimidas antes, la alimentación del Masaylo se producirá en base a las siguientes premisas:

- En ningún caso quitaremos el jumper del regulador.

- Las tierras (polo negativo, GND) de Arduino y L298N deben estar interconectadas entre sí.
- Alimentaremos el L298 N desde el portapilas, y a nuestra Arduino desde el borne +5V del L298 N

En suma, el diagrama básico de conexión portapilas → L298n → Arduino y los motores DC quedaría así:



Conexionado de los motores DC

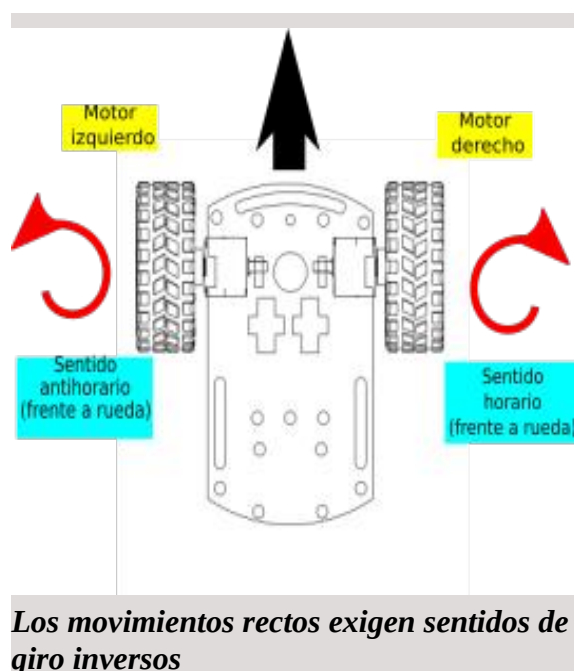
En el siguiente capítulo iniciaremos al lector en el movimiento simple (control todo-nada) del Masaylo mediante funciones sencillas.

Movimiento básico del Masaylo

Consideraciones iniciales

Ya se ha insistido varias veces en que los ejemplos contemplados en este libro están adaptados al pinout exhibido en el apartado correspondiente, y que es posible que alguno o ambos motores funcionen inversamente a lo esperado. En este caso, la solución es tan simple como intercambiar, en el L298N, sus correspondientes conexiones INx o OUTx, a gusto del lector.

Debe tenerse en cuenta también que ambos motores DC se encuentran en posición enfrentada, por lo que si queremos, por ejemplo, que el robot vaya hacia adelante, el motor izquierdo girará en sentido antihorario (visto desde enfrente de la rueda), y el derecho en sentido horario. En el diagrama de conexiones proporcionado en este libro ya se ha tenido en cuenta, lo que no es óbice para que se pueda haber cometido algún error fácilmente solucionable por el lector mediante el intercambio de patillas explicado.



Mi primer programa: mover a Masaylo hacia adelante.

Recapitulemos: el motor izquierdo está asociado a las patillas 5 y 6 de nuestra Arduino, y el derecho a las 9 y 10. Visto así, y con la previsión de que ambos motores deben tener sus sentidos intercambiados, sólo tenemos que escribir un “1” lógico en las patillas 5 y 9 y un “0” en las patillas 6 y 10.

Empecemos por un ejemplo muy sencillo: hagamos que en `setup()`, Masaylo se mueva hacia adelante cinco segundos, para a continuación pararse:

```
void setup() {  
  //Designamos los pines correspondientes como salidas  
  pinMode(5,OUTPUT);  
  pinMode(6,OUTPUT);  
  pinMode(9,OUTPUT);  
  pinMode(10,OUTPUT);  
  //Ponemos "1" y "0" según corresponda  
  digitalWrite(5,HIGH);  
  digitalWrite(6,LOW);  
  digitalWrite(9,HIGH);  
  digitalWrite(10,LOW);  
  //Esperamos un tiempo  
  delay(5000);  
  //Ponemos un "1" en las dos patillas a nivel bajo  
  digitalWrite(6,HIGH);  
  digitalWrite(10,HIGH);  
  
}  
  
void loop() {  
  //No queremos funcionamiento en bucle  
}
```

Mi primer programa con Masaylo: adelante y paro

Para detener un motor, bastará con poner a “0” o a “1” ambas patillas. Nosotros hemos optado por la segunda solución.

Uso de funciones

Para agilizar la programación, y presuponiendo que el estudiante ya tiene cierto conocimiento del entorno de programación de Arduino, basado en C++, crearemos cinco funciones básicas de control del motor:

adelante()

Código	Movimiento
<pre>void adelante(){ digitalWrite(5,HIGH); digitalWrite(6,LOW); digitalWrite(9,HIGH); digitalWrite(10,LOW); }</pre>	

atras()

Código	Movimiento
<pre>void atras(){ digitalWrite(5,LOW); digitalWrite(6,HIGH); digitalWrite(9,LOW); digitalWrite(10,HIGH); }</pre>	

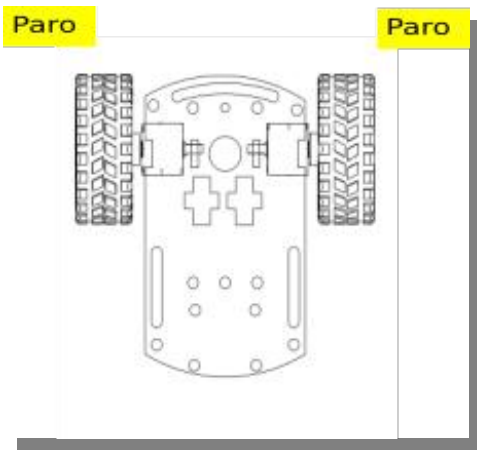
izquierda()

Código	Movimiento
<pre>void izquierda(){ digitalWrite(5,LOW); digitalWrite(6,LOW); digitalWrite(9,LOW); digitalWrite(10,HIGH); }</pre>	

derecha()

Código	Movimiento
<pre>void derecha(){ digitalWrite(5,HIGH); digitalWrite(6,LOW); digitalWrite(9,LOW); digitalWrite(10,LOW); }</pre>	

alto()

Código	Movimiento
<pre>void alto(){ digitalWrite(5,LOW); digitalWrite(6,LOW); digitalWrite(9,LOW); digitalWrite(10,LOW); }</pre>	

Programando Masaylo con funciones: izquierda, adelante, derecha, adelante, atras

El siguiente programa pretende proporcionar la base más simple posible al no iniciado para que comprenda el uso de funciones en programación: vamos a proporcionar a Arduino una secuencia de movimientos izquierda → adelante → derecha → adelante → alto → atras, de duración aleatoria entre 0,5 y 2 segundos cada movimiento.

Para facilitar el cambio de conexiones entre pines, a su vez, nos valdremos de la instrucción `#DEFINE` para utilizar referencias a los pines en lugar de escribir directamente los números.

```
//Asignamos una referencia a cada pin
```

```
#define MIA 5
```

```
#define MIB 6
```

```
#define MDA 9
```

```
#define MDB 10
```

```
//FUNCIONES DE MOVIMIENTO
```

```
void adelante (){
```

```
digitalWrite(MIA,HIGH);  
digitalWrite(MIB,LOW);  
digitalWrite(MDA,HIGH);  
digitalWrite(MDB,LOW);  
}
```

```
void izquierda (){  
  digitalWrite(MIA,LOW);  
  digitalWrite(MIB,LOW);  
  digitalWrite(MDA,HIGH);  
  digitalWrite(MDB,LOW);  
}
```

```
void derecha(){  
  digitalWrite(MIA,HIGH);  
  digitalWrite(MIB,LOW);  
  digitalWrite(MDA,LOW);  
  digitalWrite(MDB,LOW);  
}
```

```
void atras(){  
  digitalWrite(MIA,LOW);  
  digitalWrite(MIB,HIGH);  
  digitalWrite(MDA,LOW);  
  digitalWrite(MDB,HIGH);  
}
```

```
void alto(){  
  digitalWrite(MIA,LOW);  
  digitalWrite(MIB,LOW);
```

```
digitalWrite(MDA,LOW);
digitalWrite(MDB,LOW);
}

void setup() {
//Declaramos todos los pines como salidas digitales
pinMode(MIA,OUTPUT);
pinMode(MIB,OUTPUT);
pinMode(MDA,OUTPUT);
pinMode(MDB,OUTPUT);
}

void loop() {
//Se llama a cada función en el orden deseado y duración especificada
izquierda();
delay((int)random(500,2000));
adelante();
delay((int)random(500,2000));
derecha();
delay((int)random(500,2000));
adelante();
delay((int)random(500,2000));
alto();
delay((int)random(500,2000));
atras();
delay((int)random(500,2000));
}
```

Control de velocidad mediante PWM

Regulación de velocidad en corriente continua

Como sabrán ya aquellos lectores con una cierta experiencia en el trabajo con tarjetas microcontroladoras como Arduino, no suele ser habitual la posibilidad de reproducir en estos dispositivos señales de tipo analógico, pero sí que es posible *EMULAR* dichas señales mediante la técnica denominada *Pulse Width Modulation*, o modulación por ancho de pulso, que en Arduino se resume en el uso de la función ***analogWrite(pin, velocidad)***.

A grandes rasgos, dicha técnica consiste en emitir “1” y “0” lógicos a través de la patilla elegida, alternándolos con una cierta frecuencia (pulsos), y cuya respectiva duración con respecto a la señal contraria supondrá un peso específico que marcará una media a lo largo del tiempo que oscilará entre 0 y 5 V.

Por ejemplo, si a lo largo de un segundo, alternamos 25 ciclos de 20 milisegundos con un “0” lógico (0 V en el pin) con otros 25 ciclos de igual duración pero con un “1” (5 V), el valor medio de dicha señal a lo largo de un segundo será exactamente de la mitad (2,5 V).

Si repartimos de manera distinta la duración de dichos pulsos, por ejemplo, 25 ciclos de 10 milisegundos para el “0”, alternándolos con con 25 ciclos de 30 milisegundos para el “1”, puede calcularse que el valor medio “emulado” (si seguimos trabajando con valores de 0 V y 5 V) es de 3,75 V.

Uso de la función ***analogWrite(pin, velocidad)***

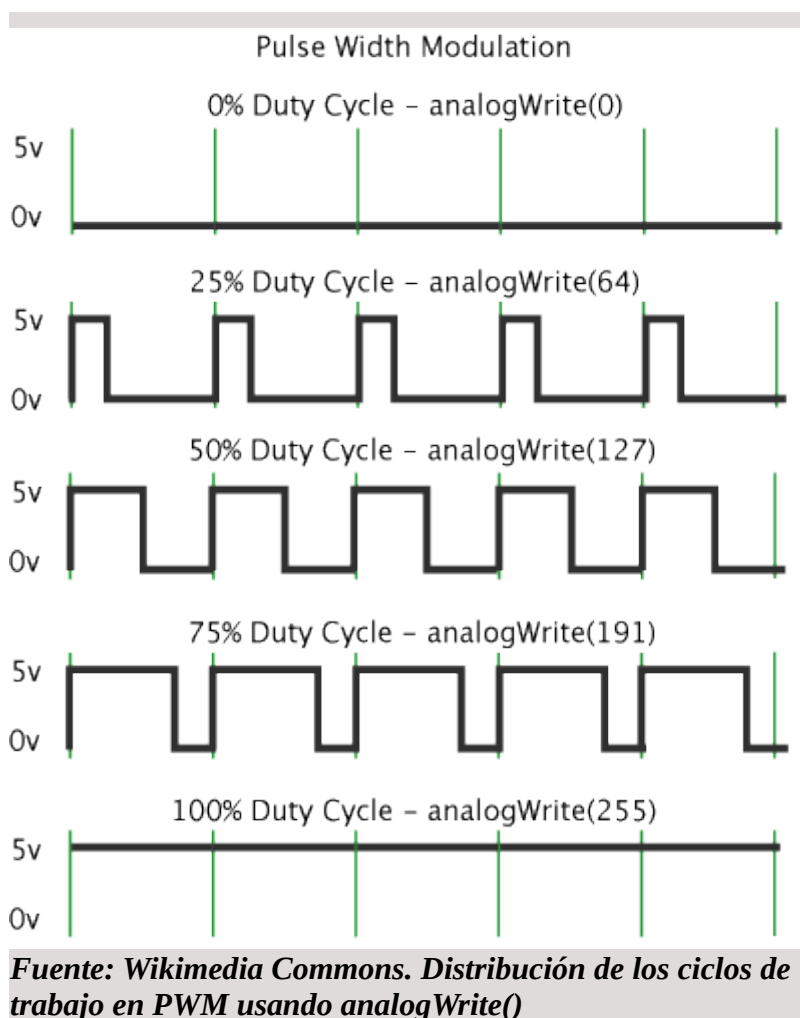
Como ya se ha explicado, y sabrá el lector algo más avezado en Robótica Educativa, la tarjeta Arduino puede utilizar la técnica PWM en muchas de sus patillas mediante la función ***analogWrite(pin, velocidad)***. En la tarjeta Nano, concretamente, dicha función está disponible en los pines 3,5,6,9,10 y 11.

La función ***analogWrite()*** es muy fácil de utilizar . Sólo precisa de dos parámetros entre paréntesis, separados por una coma, a saber: el pin en que queremos el pulso, y el valor que queremos establecer, válido entre 0 y 255. Así, 0, por ejemplo supondría 0 V, 127 emularía un valor de 2,5 V y 255 el valor completo de 5 V.

Influencia del L298N sobre nuestra función ***analogWrite***

Ya hemos explicado que el L298N es un circuito de transistores montados como puente en H que emiten a la salida una señal equivalente a la de la alimentación, con una cierta caída de tensión de algo más de 1 V (hasta 1,5 V) en ocasiones. Si alimentamos el Masaylo con tensiones inferiores a los 6 V que dará nuestro portapilas, podemos encontrarnos con errores en el funcionamiento del robot usando esta técnica que redunden incluso en su imposibilidad de moverse. Si alimentáramos, por el contrario, al robot con tensiones superiores, por ejemplo, 9 V, habrá variaciones en su

comportamiento que permitirán un control más afinado de las velocidades, pero ello obligará a su vez al programador a estar pendiente de no sobrepasar valores que supongan tensiones perjudiciales para el motor DC (no aconsejamos superar los 6 V en bornes), si bien son elementos muy estables y sólidos (por eso aguantan tantos años en talleres de Tecnología en Educación Secundaria).



Uso de funciones para control de velocidad en el Masaylo

En el capítulo anterior, utilizábamos la función *digitalWrite(pin,estado)* en las patillas que controlaban cada motor en sus bornes escribiendo “1” (5V) y “0” (0V) en cada una de las patillas. Para regular la velocidad, seguiremos utilizando *digitalWrite()* para escribir “0” en uno de los contactos del motor, por un lado, y *analogWrite()*, por otro, para establecer un pulso que a lo largo del tiempo produzca el efecto de la señal que queremos en dicho contacto.

Hay que tener claro, por otro lado, que estos motores DC necesitan un valor mínimo de entre 3 y 3,5 V para empezar a moverse,. Por ello, aunque el valor máximo de *analogWrite* es 255, (equivalente,

en un principio, a 5 V), el valor mínimo no debería bajar de un número dictado por la experiencia en cada robot, y que para nosotros no debería bajar de 120.

Igual que hemos hecho en el capítulo anterior, proponemos una serie de funciones que simplifiquen la programación del robot en orden a disponer movimientos controlados.

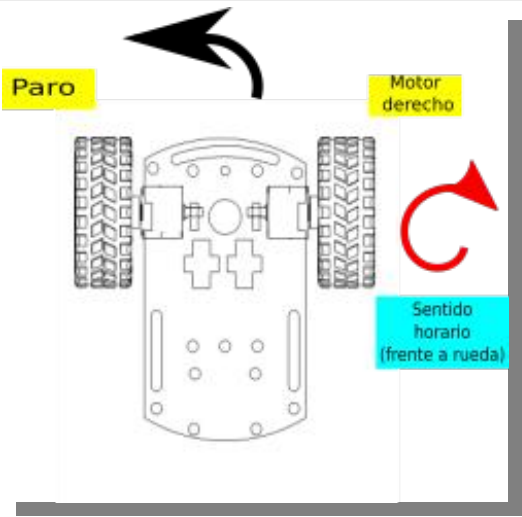
adelante(valor)

Código	Movimiento
<pre>void adelante(int valor){ analogWrite(5,valor); digitalWrite(6,LOW); analogWrite(9,valor); digitalWrite(10,LOW); }</pre>	

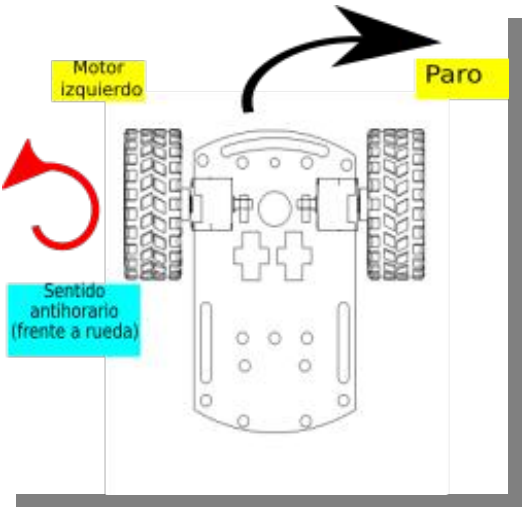
atras(valor)

Código	Movimiento
<pre>void atras(int valor){ digitalWrite(5,LOW); analogWrite(6,valor); digitalWrite(9,LOW); analogWrite(10,valor); }</pre>	

izquierda(valor)

Código	Movimiento
<pre>void izquierda(int valor){ digitalWrite(5,LOW); digitalWrite(6,LOW); digitalWrite(9,LOW); analogWrite(10,valor); }</pre>	

derecha(valor)

Código	Movimiento
<pre>void derecha(int valor){ analogWrite(5,valor); digitalWrite(6,LOW); digitalWrite(9,LOW); digitalWrite(10,LOW); }</pre>	

No debe preocupar al lector la coincidencia en los nombres de dichas funciones con las del capítulo anterior; ambos tipos pueden convivir en el mismo programa de Arduino gracias a lo que se conoce en programación como ***SOBRECARGA DE FUNCIONES***

La función **alto()**, definida en el capítulo anterior, sigue siendo lógicamente válida dado que no es posible modular la velocidad en un vehículo parado.

Ejemplo de control de velocidad de Masaylo mediante funciones y PWM

El siguiente programa establece tres valores de velocidad discretos elegidos a voluntad como *lento*, *ligero* y *rápido*, asociados a los valores enteros 140, 200 y 255. Estableceremos una única secuencia de movimiento, por ejemplo, adelante rápido → adelante ligero → adelante lento → derecha lenta → atrás lenta → atrás media → izquierda rápida, separadas por intervalos de dos segundos para movimientos rectos y medio segundo para los giros. Veamos pues:

```
//Asignamos una referencia a cada pin

#define MIA 5
#define MIB 6
#define MDA 9
#define MDB 10

//FUNCIONES DE MOVIMIENTO DE VELOCIDAD REGULADA

void adelante(int valor){
  analogWrite(5,valor);
  digitalWrite(6,LOW);
  analogWrite(9,valor);
  digitalWrite(10,LOW);
}

void atras(int valor){
  digitalWrite(5,LOW);
  analogWrite(6,valor);
  digitalWrite(9,LOW);
  analogWrite(10,valor);
}

void izquierda(int valor){
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(9,LOW);
  analogWrite(10,valor);
}

void derecha(int valor){
  analogWrite(5,valor);
  digitalWrite(6,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
}

void alto(){
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
}
```



```
digitalWrite(9,LOW);
digitalWrite(10,LOW);
}
void setup() {
//Declaramos todos los pines como salidas digitales
pinMode(MIA,OUTPUT);
pinMode(MIB,OUTPUT);
pinMode(MDA,OUTPUT);
pinMode(MDB,OUTPUT);
//Se llama a cada función en el orden deseado y duración especificada
//adelante rápido, ligero y lento
adelante(255);
delay(2000);
adelante(200);
delay(2000);
adelante(150);
delay(2000);
//derecha lenta
derecha(150);
delay(500);
//atrás lenta y media
atras(150);
delay(2000);
atras(200);
delay(2000);
//izquierda rápida
izquierda(255);
delay(500);
//Paramos
alto();
```

```
}  
  
void loop() {  
  //No trabajamos en bucle en este ejemplo  
}
```

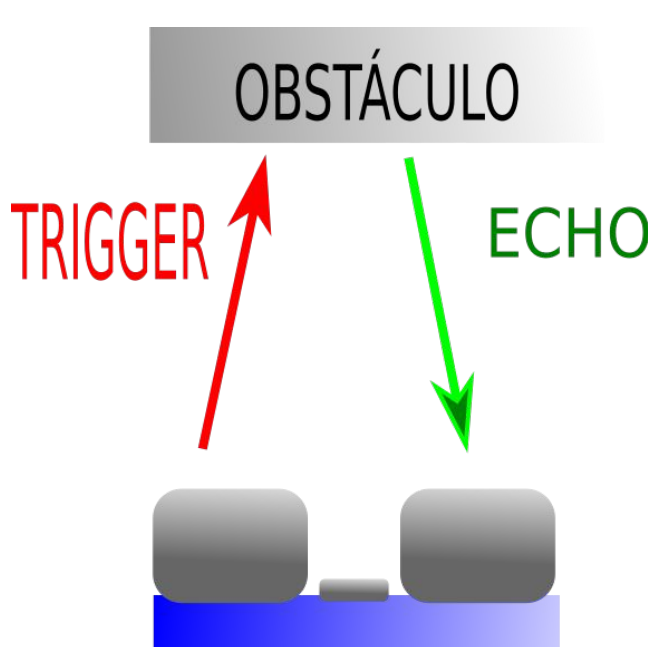
Trabajo con sensores: detección de distancias con el HC-SR04

Detección de distancias por ultrasonidos

Como casi todos los otros campos de conocimientos tratados a lo largo de este texto relacionados con la Robótica Educativa, existe ya una extensa bibliografía en la red sobre el uso de módulos como el HC-SR04 o el PING de Parallax en la red. No queremos dejar pasar la ocasión de llamar la atención del lector con respecto a un [genial artículo del ya mencionado Luis Llamas](#).

El principio básico de cualquier sensor de ultrasonidos es muy sencillo:

- Un circuito emisor, comúnmente denominado “*Trigger*”, lanza un pulso de ultrasonidos (para ello, Arduino debe emitir un “1” en la patilla correspondiente).
- Un temporizador se pone a 0 y empieza a contar.
- Un circuito receptor, de nombre común “*Echo*”, detecta el ultrasonido que ha rebotado (siempre que haya obstáculo) y envía un “1” lógico a una patilla de Arduino (puede ser o no la misma que hemos conectado al emisor, dependiendo del modelo).
- Detectado el “1” lógico en el pin conectado a “*Echo*” el temporizador se para.

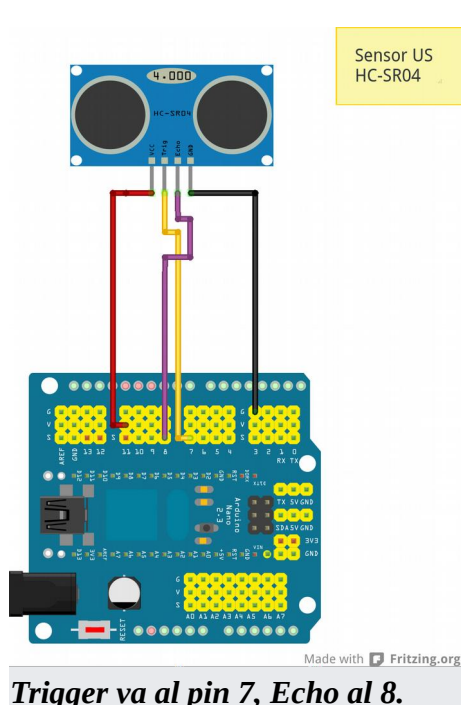


Principio de funcionamiento del sensor de ultrasonidos

- Conocida la velocidad del sonido (unos 343 m/sg), y que el sonido ha tenido que recorrer el espacio hasta el obstáculo dos veces, ida y vuelta, se realiza el cálculo correspondiente. En recorrer 1 cm, el tiempo que tarda el sonido es 0,000029155 sg. Teniendo en cuenta que la distancia es doble, un obstáculo localizado a 1 cm del sensor tardaría en detectarse 0,000058309 sg, es decir, 58,309 microsegundos, valor que utilizaremos para convertir el tiempo medido en distancia en cm.

Función básica para leer distancias con el HC-SR04

Ya indicamos en capítulos anteriores los pines que hemos previsto para el sensor de ultrasonidos: 7 para *Trigger* (habrá que declararlo como OUTPUT), y 8 para *Echo* (INPUT), tal como puede verse en la imagen. Como ocurre con todos los sensores de esta naturaleza, hay también siempre una patilla para conectar a +5 V (alimentación) y otra a tierra (GND).



A la función que utilizaremos para aprender a utilizar este sensor la llamaremos *dameDistancia()*, que devolverá un valor de tipo **long**:

```
long dameDistancia(){
long espacio;
long tiempo;
//Apagamos (para evitar falsas lecturas) y encendemos Trigger durante 10 us
digitalWrite(_7,LOW);
```

```
delayMicroseconds(4);
digitalWrite(8,HIGH);
delayMicroseconds(10);
digitalWrite(7,LOW);
//Medimos el tiempo en us que tarda en volver el pulso y realizamos el cálculo.
tiempo=pulseIn(8,HIGH);
espacio=tiempo/58.309;
//La función devuelve un valor de tipo long correspondiente a la distancia en cm.
return espacio;
}
```

Ejemplo 1: detección de distancias y comunicación vía puerto serie.

El presente programa utiliza la función *dameDistancia()* para medir el espacio entre el sensor y un obstáculo delante del Masaylo para pasarlo a continuación por el puerto serie:

```
//Definimos los pines echo y trigger
#define trig 7
#define echo 8
long dameDistancia(){
long espacio;
long tiempo;
//Apagamos (para evitar falsas lecturas) y encendemos Trigger durante 10 us
digitalWrite(trig,LOW);
delayMicroseconds(4);
digitalWrite(trig,HIGH);
delayMicroseconds(10);
digitalWrite(trig,LOW);
//Medimos el tiempo en us que tarda en volver el pulso y realizamos el cálculo.
tiempo=pulseIn(echo,HIGH);
espacio=tiempo/58.309;
```

```
//La función devuelve un valor de tipo long correspondiente a la distancia en cm.
return espacio;
}

void setup() {
//trigger es OUTPUT, echo INPUT
pinMode(trig, OUTPUT);
pinMode(echo, INPUT);
//Inicializamos el puerto serie
Serial.begin(9600);
}

void loop() {
//Utilizamos una variable local
long distancia=dameDistancia();
//Imprimimos la variable por el puerto serie
Serial.print("Distancia: ");
Serial.println(distancia);
//Tiempo mínimo para evitar desfases
delay(100);
}
```

Ejemplo 2: Masaylo salvaobstáculos

Apliquemos lo aprendido para que Masaylo se mueva libremente hacia adelante salvo que se encuentre un obstáculo entre 40 y 20 cm de distancia, en cuyo caso reducirá la marcha. Si el obstáculo se encuentra a menos de 20 cm, se irá hacia atrás y girará hacia la derecha para buscar otro camino.

```
//Asignamos una referencia a cada pin
#define MIA 5
#define MIB 6
#define MDA 9
```

```
#define MDB 10
#define trig 7
#define echo 8

//FUNCIÓN DE MEDICIÓN DE DISTANCIA
long dameDistancia(){
long espacio;
long tiempo;
//Apagamos (para evitar falsas lecturas) y encendemos Trigger durante 10 us
digitalWrite(trig,LOW);
delayMicroseconds(4);
digitalWrite(trig,HIGH);
delayMicroseconds(10);
digitalWrite(trig,LOW);
//Medimos el tiempo en us que tarda en volver el pulso y realizamos el cálculo.
tiempo=pulseIn(echo,HIGH);
espacio=tiempo/58.309;
//La función devuelve un valor de tipo long correspondiente a la distancia en cm.
return espacio;
}

//FUNCIONES DE MOVIMIENTO

void adelante (){
digitalWrite(MIA,HIGH);
digitalWrite(MIB,LOW);
digitalWrite(MDA,HIGH);
digitalWrite(MDB,LOW);
}

void izquierda (){
```

```
digitalWrite(MIA,LOW);
digitalWrite(MIB,LOW);
digitalWrite(MDA,HIGH);
digitalWrite(MDB,LOW);
}

void derecha(){
digitalWrite(MIA,HIGH);
digitalWrite(MIB,LOW);
digitalWrite(MDA,LOW);
digitalWrite(MDB,LOW);
}

void atras(){
digitalWrite(MIA,LOW);
digitalWrite(MIB,HIGH);
digitalWrite(MDA,LOW);
digitalWrite(MDB,HIGH);
}

void alto(){
digitalWrite(MIA,LOW);
digitalWrite(MIB,LOW);
digitalWrite(MDA,LOW);
digitalWrite(MDB,LOW);
}

//FUNCIONES DE MOVIMIENTO DE VELOCIDAD REGULADA

void adelante(int valor){
analogWrite(MIA,valor);
digitalWrite(MIB,LOW);
analogWrite(MDA,valor);
```



```
digitalWrite(MDB,LOW);
}

void atras(int valor){
digitalWrite(MIA,LOW);
analogWrite(MIB,valor);
digitalWrite(MDA,LOW);
analogWrite(MDB,valor);
}
void izquierda(int valor){
digitalWrite(MIA,LOW);
digitalWrite(MIB,LOW);
digitalWrite(MDA,LOW);
analogWrite(MDB,valor);
}
void derecha(int valor){
analogWrite(MIA,valor);
digitalWrite(MIB,LOW);
digitalWrite(MDA,LOW);
digitalWrite(MDB,LOW);
}

void setup() {
//Declaramos cada pin en el modo en que corresponda
pinMode(trig,OUTPUT);
pinMode(echo,INPUT);
pinMode(MIA,OUTPUT);
pinMode(MIB,OUTPUT);
pinMode(MDA,OUTPUT);
pinMode(MDB,OUTPUT);
}
void loop() {
//Medimos la distancia y la asignamos a una variable local
long distancia=dameDistancia();
if (distancia>30){
//No hay obstáculos a la vista
adelante();
```

```
}else if (distancia<20){  
  //Obstáculo ineludible, paro, atrás y derecha  
  alto();  
  delay(500);  
  atras();  
  delay(2000);  
  derecha();  
  delay(500);  
}else{  
  //Obstáculo entre 20 y 30 cm. Proceder con cautela  
  adelante(150);  
}  
//Damos tiempo para evitar problemas de desfases  
delay(100);  
}
```

Trabajo con sensores: siguelíneas con sensores de infrarrojos

Características de los sensores de infrarrojos

Un sensor de infrarrojos participa del mismo principio funcional que uno de ultrasonidos: se alimenta un diodo LED infrarrojo para que emita un pulso de luz, detectable por un fototransistor, en caso de haber rebotado en algún obstáculo. No obstante, este tipo de sensores presenta características propias que lo distingue de un sensor de ultrasonidos:

Ventajas

- Son sensiblemente más baratos
- Disponen de un potenciómetro que permite ajustar la sensibilidad del sensor
- La emisión de la luz infrarroja es continua, no se necesita pin de control. Así que sólo hacen falta tres conexiones: V, GND y OUT (pin de señal). Son más fáciles de instalar y de manejo más simple por parte del sistema de control.
- Pueden detectar cambios, si no de color, sí de luminosidad en la superficie sobre la que apuntan, lo que les hace ideales para el diseño de **robot siguelíneas**.

Desventajas

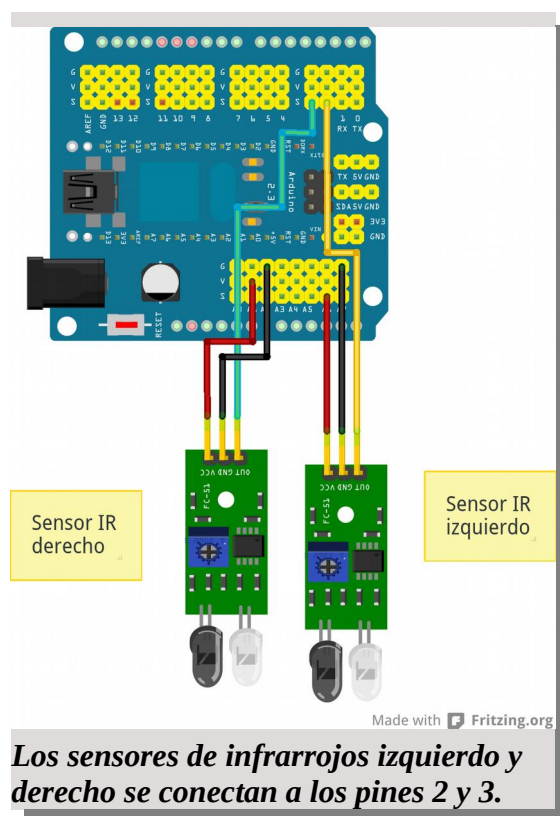
- Su rango de detección es sensiblemente más bajo que en el caso de los ultrasonidos. Raramente superan los 30 cm, en el mejor de los casos, en el FC-51, y menos de 20 mm en el TCRT5000
- Su sensibilidad es muy vulnerable a los cambios de luz en la estancia en la que se encuentre el robot, lo que obliga a continuos reajustes mediante el potenciómetro anexo. En ocasiones es recomendable rodear el par formado por el LED y el fototransistor por algún tipo de capuchón improvisado con cinta o cartón que reduzca al máximo el ruido ocasionado por dichos cambios de luz.

Conexionado y lectura de sensores infrarrojos

Masaylo incorpora soportes para dos tipos de sensores: el FC-51 y el TCRT5000. En estas líneas desarrollaremos el trabajo con el FC-51, que es el que teníamos a mano en el momento de redactarlas. El conexionado y proceso de lectura en ambos casos son los mismos. La diferencia entre ambos estriba en que tenemos la sensación de que el TCRT5000 puede ser más eficaz en el caso del siguelíneas, mientras que los FC-51 podrían ser más útiles si queremos hacerlos servir como sensores secundarios al de ultrasonidos en el caso del salvaobstáculos.

Este tipo de sensor, como ya se ha dicho, tiene tres patillas: dos de alimentación (5V y GND) y la de señal (OUT). En nuestro planteamiento de pinout, hemos establecido el patillaje:

- Sensor de infrarrojos izquierdo → Pin 2 de Arduino
- Sensor de infrarrojos derecho → Pin 3 de Arduino



Dado que son señales que Arduino debe leer, ambos pines (o los que elija el estudiante) deben ser puestos en modo INPUT.

El comportamiento de estos sensores sigue las siguientes pautas:

- Si la luz rebota en el fototransistor, significa que hay obstáculo o que se está detectando un color claro (se lee “blanco”). En la patilla OUT se leerá un “0” lógico
- Si la luz no vuelve al sensor, se entiende que no hay obstáculo, o que sí lo hay pero su superficie es lo bastante oscura como para absorberla (se lee “negro”). En la patilla OUT se leerá un “1” lógico.

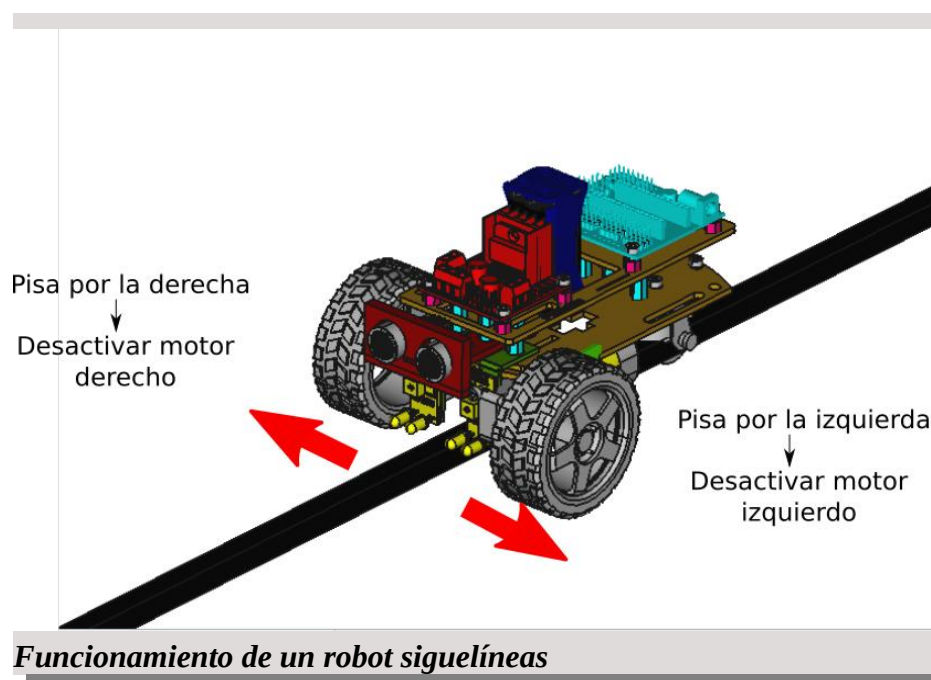
Programación de Masaylo como siguelíneas

Un clásico en robótica educativa es la programación de un siguelíneas. La idea es conseguir un sistema autónomo capaz de seguir un camino que tenga un tono claramente distinto de su entorno, normalmente una línea negra sobre fondo blanco. Para ello, existen múltiples métodos y de muy

diversa complejidad. En los robots de carreras se utiliza el cálculo PID (Proporcional-Integrativo-Derivativo), normalmente con muchos más de dos sensores. El uso, por otro lado, de motores de tipo paso a paso permite un control del movimiento muy necesario en este tipo de trabajos, aunque ello redunde, al menos en el caso de los modelos menos caros, en una clara pérdida de la velocidad.

Como Masaylo es un robot de muy bajo coste, pensado principalmente para la introducción del estudiante a los principios más básicos de este campo, optaremos por una solución intermedia que exigirá, eso sí, un riguroso calibrado de los sensores (potenciómetro mediante) antes de cada prueba práctica. Por otro lado, recalcar que no es conveniente utilizar funciones de movimiento de tipo todo/nada en este caso, dado que un motor DC proporciona excesiva potencia para dar tiempo al robot a reaccionar a los cambios en la trayectoria. Optaremos, en cambio, por el uso de movimientos de velocidad regulada mediante la función **analogWrite()** que comentábamos hace tres capítulos.

La idea es la siguiente: mientras ambos sensores envíen un “0” lógico al sistema de control, significa que nuestro vehículo sigue correctamente el camino, y enviaremos la orden de “adelante” a una velocidad adecuada (depende de la experiencia con cada robot). En cambio, si el sensor derecho, por ejemplo, envía un “1” (pisamos el camino por la derecha), significa que nos estaremos saliendo por la izquierda, así que habrá que girar hacia la derecha para compensar nuestra trayectoria. Lo mismo sucederá si el sensor izquierdo envía un “1” (nos salimos por la izquierda, hay que girar hacia la derecha).



¿Y qué sucederá si ambos sensores envían un “1” lógico?. Sucederá que, por algún error, el vehículo está mirando perpendicularmente a una parte del camino. El programador puede idear varias estrategias para que el robot retroceda y trate de retomar su camino. Por razones de simplicidad (es nuestro primer siguelíneas), nosotros lo pararemos.

Para simplificar la programación, hemos diseñado que llamaremos *leeSuelo()* y que devuelve un valor de tipo **int**, en base al cual decidiremos la estrategia a seguir: 0 → adelante, 1 → derecha, 2 → izquierda, 3 → alto.

```
//Recuerde el lector adaptar el programa a su propia elección de pines.
```

```
#define sIzquierda 2
```

```
#define sDerecha 3
```

```
int leeSuelo(){
```

```
//Los sensores dan 1 para negro, 0 para blanco
```

```
int valor=0;
```

```
if (digitalRead(sDerecha)==HIGH){
```

```
    valor=valor+1;
```

```
}
```

```
if (digitalRead(sIzquierda)==
```

```
valor=valor+2;
```

```
}
```

```
return valor;
```

```
}
```

Planteamiento: programar a Masaylo como siguelíneas

A continuación, aglutinando todo lo explicado hasta ahora en este libro, presentamos un programa de control de la velocidad en los motores DC de nuestro robot de acuerdo a la información enviada por los sensores infrarrojos. Como decíamos antes, la velocidad elegida (de valores que oscilarían entre 120 y 255) en cada motor dependerá de las propias circunstancias de cada robot en cada caso, previa calibración de dichos sensores. Nuestra elección ha sido de 130 para *adelante()* y de 150 para *izquierda()* y *derecha()*, aunque el estudiante puede experimentar con valores distintos.

```
//Definimos patillaje
```

```
#define MIA 5
```

```
#define MIB 6
```

```
#define MDA 9
```

```
#define MDB 10
```

```
#define sDerecha 3
```

```
#define sIzquierda 2
```

```
//Declaramos la variable global valorLeido que recogerá la lectura de los sensores
```

```
int valorLeido;
```

```
//Escribimos las funciones de movimiento de velocidad controlada
```

```
void alto(){
digitalWrite(MIA,LOW);

digitalWrite(MIB,LOW);
digitalWrite(MDA,LOW);

digitalWrite(MDB,LOW);
}

void adelante( int velocidad){
analogWrite(MIA,velocidad);
digitalWrite(MIB,LOW);
analogWrite(MDA,velocidad);
digitalWrite(MDB,LOW);
}

void izquierda(int velocidad){
digitalWrite(MIA,LOW);
digitalWrite(MIB,LOW);
analogWrite(MDA,velocidad);
digitalWrite(MDB,LOW);
}

void derecha(int velocidad){
analogWrite(MIA,velocidad);
digitalWrite(MIB,LOW);
digitalWrite(MDA,LOW);
digitalWrite(MDB,LOW);
}

//No la utilizamos en este caso, pero puede ser útil para modificaciones posteriores
void atras(int velocidad){
analogWrite(MIB,velocidad);
digitalWrite(MIA,LOW);
analogWrite(MDB,velocidad);
digitalWrite(MDA,LOW);
}

//Función de lectura de sensores siguelíneas leeSuelo()
int leeSuelo(){
//Los sensores dan 1 para negro, 0 para blanco. Valor recoge el total de las lecturas
int valor=0;
```

```
//Si pisamos por la derecha, sumamos 1
if (digitalRead(sDerecha)==HIGH){
  valor=valor+1;
}

//Si pisamos por la izquierda, sumamos 2
if (digitalRead(sIzquierda)==HIGH){
  valor=valor+2;
}
return valor;
}

//Configuración
void setup() {
  //Pines en modo lectura/escritura
  pinMode(MIA,OUTPUT);
  pinMode(MIB,OUTPUT);
  pinMode(MDA,OUTPUT);
  pinMode(MDB,OUTPUT);
  pinMode(sDerecha, INPUT);
  pinMode(sIzquierda, INPUT);
  valorLeido=0;
}

//Masaylo empieza su singladura
void loop() {
  //Llamamos a la función de lectura de los sensores
  valorLeido=leeSuelo();
  //Decidimos según lectura
  switch(valorLeido){
    case 3:
      alto();
      break;

    case 1:
      derecha(150);
      break;

    case 2:
      izquierda(150);
      break;

    case 0:
      adelante(130);
```



```
break;  
}  
}
```