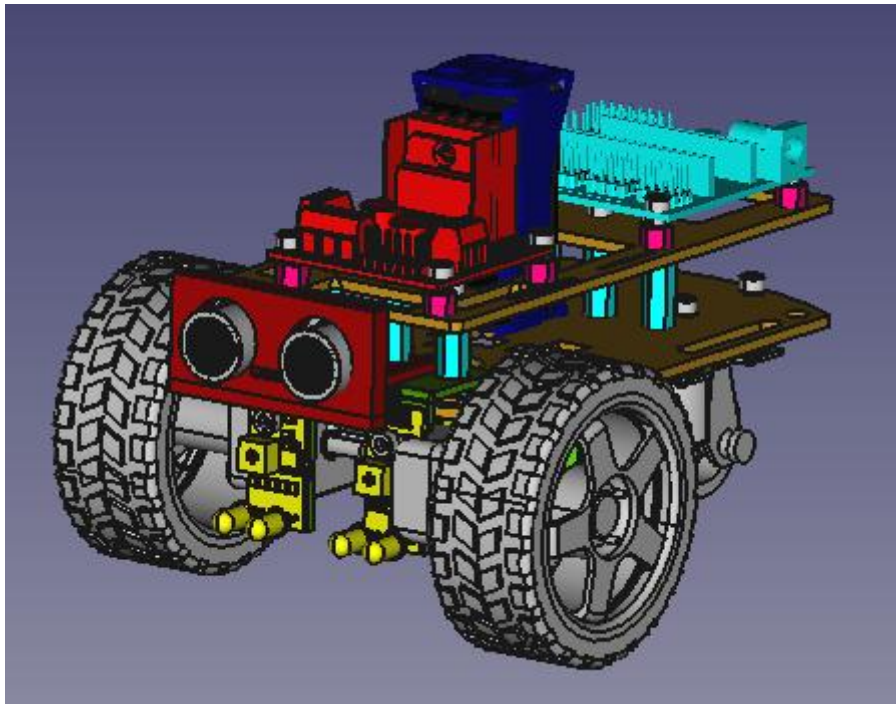


MASAYLO

Robot educativo Open Source

Manual básico de montaje y programación



Antonio Gómez García
M.^a Dolores Noguerras Atance



ÍNDICE

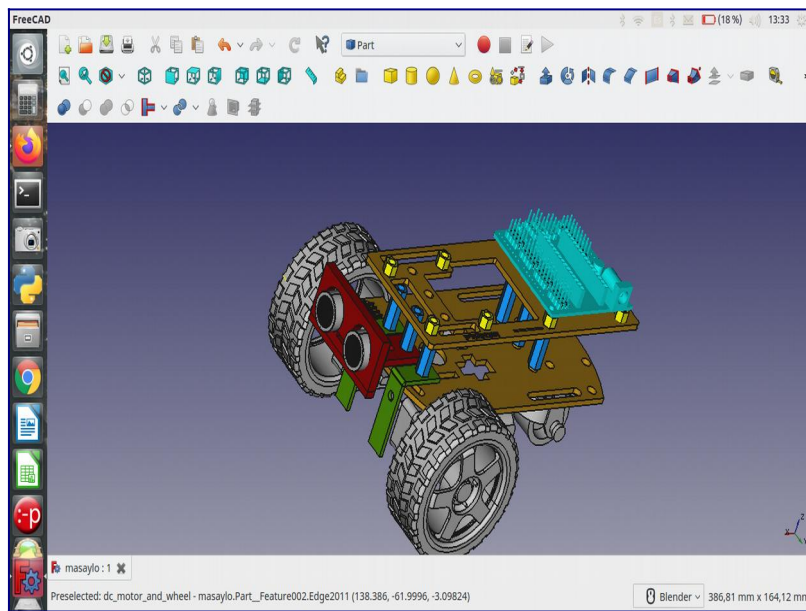
Sumario

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.

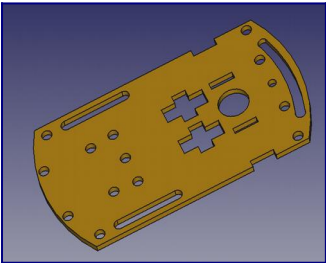
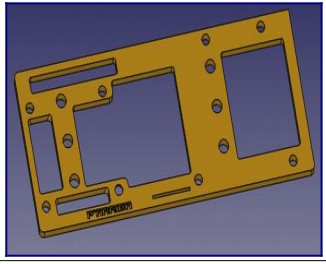
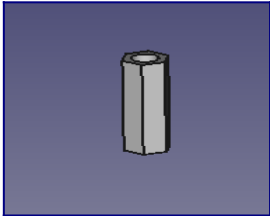


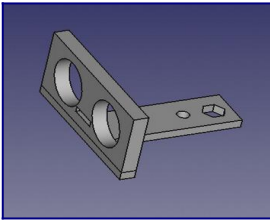
Apariencia general

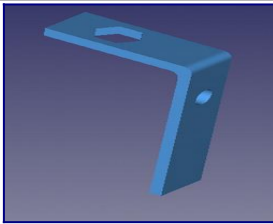
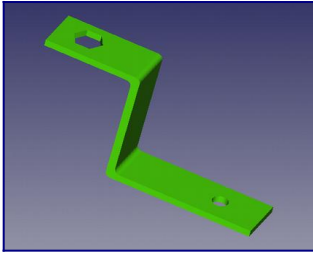


Utilizaremos: un Arduino nano con su correspondiente shield de expansión, un módulo L298N para gobernar dos motores DC, una rueda loca de 25x13 cm (o similar), un módulo sensor de ultrasonidos de tipo HC-SR04, dos sensores de infrarrojos de tipo FC-51 y un módulo BT modelo HC-05 o HC-06 que posiblemente 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).

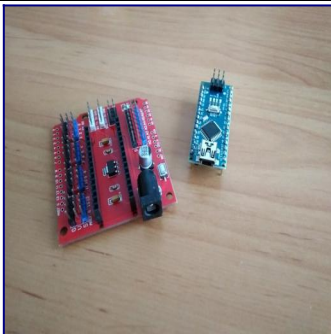

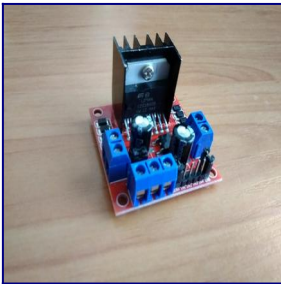
Piezas, tornillería y circuitos necesarios

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

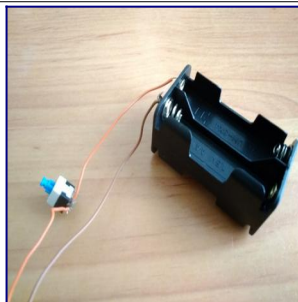
Electrónica imprescindible	
Arduino nano con shield de expansión v3.0	
Micromotores DC modelo TT con reductora y rueda(x2)	
Módulo L298N para control de motores DC	

Electrónica imprescindible

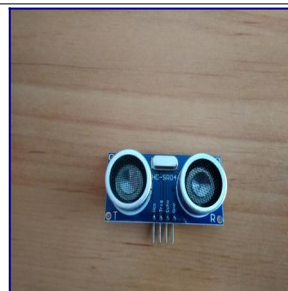
Interruptor de 8 mm o similar



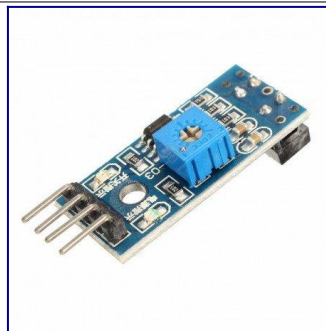
Portapilas 4xAA
(el portapilas 4xAAA también sirve)
Soldar el interruptor al polo positivo del portapilas

**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

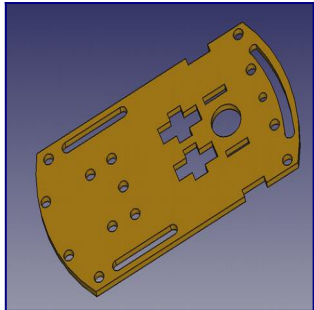

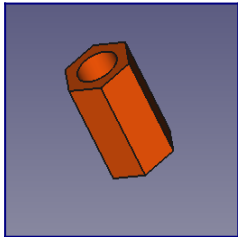

Descripción	Imagen
Rueda loca universal para coches, de nylon 23x15	
Tornillos y tuercas varios de tipo M3, de 12, 16 y 35 mm	


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

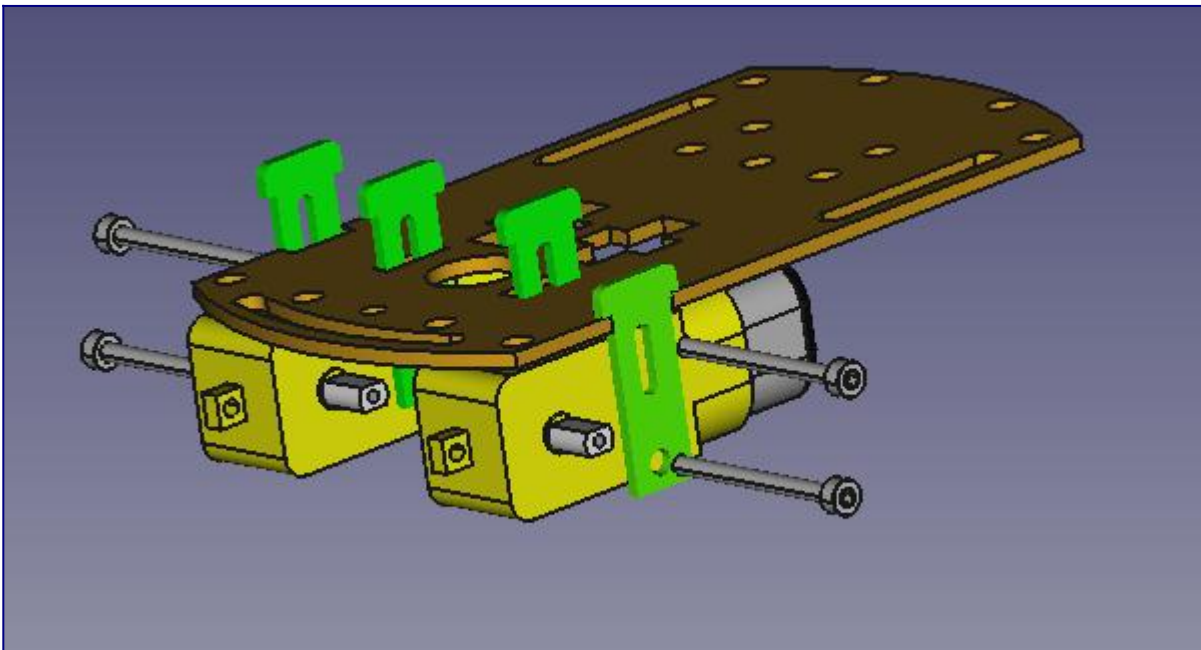
Piezas

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	

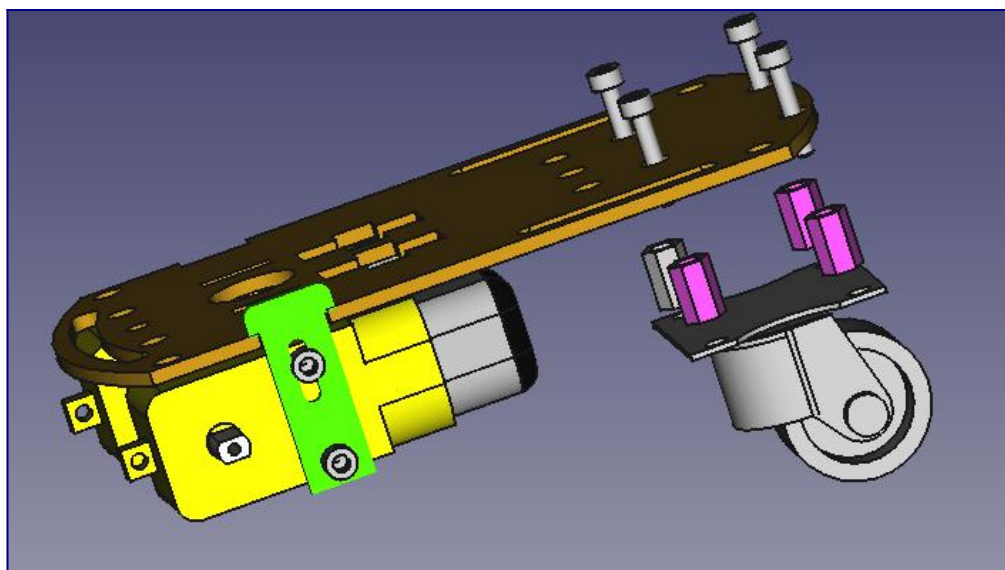
Nombre	Imagen
Rueda loca universal para coches	
4 tornillos M3x35 mm para atornillar los motores a la base	
4 tornillos M3x16 mm para fijar la rueda loca a la base	
8 tuercas M3	

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


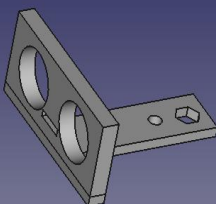
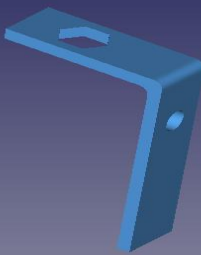



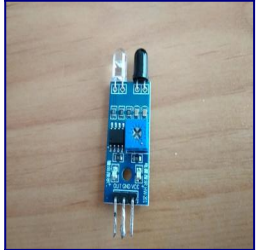
b) Fijar rueda loca a la base mediante tornillos M3x16 mm a través de los pasadores



Montar sensores y segundo piso

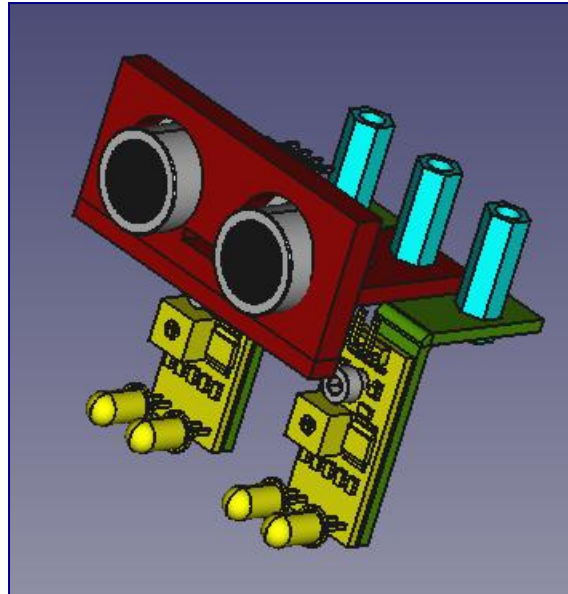
Piezas

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)	

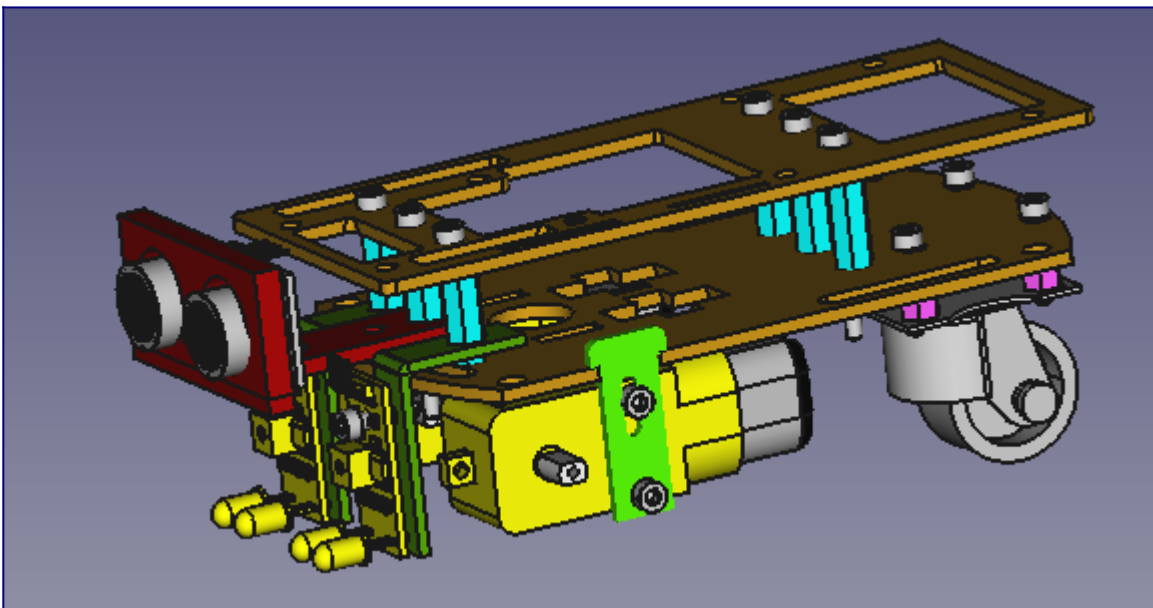
Nombre	Imagen
Segundo piso del robot	
Sensor de ultrasonidos HC-SR04	
Sensor de infrarrojos (x2)	
6 tornillos M3x35 mm para fijar el segundo piso a la base	
3 tornillos M3x12 mm para atornillar los sensores a sus soportes	
9 tuercas M3	

Proceso

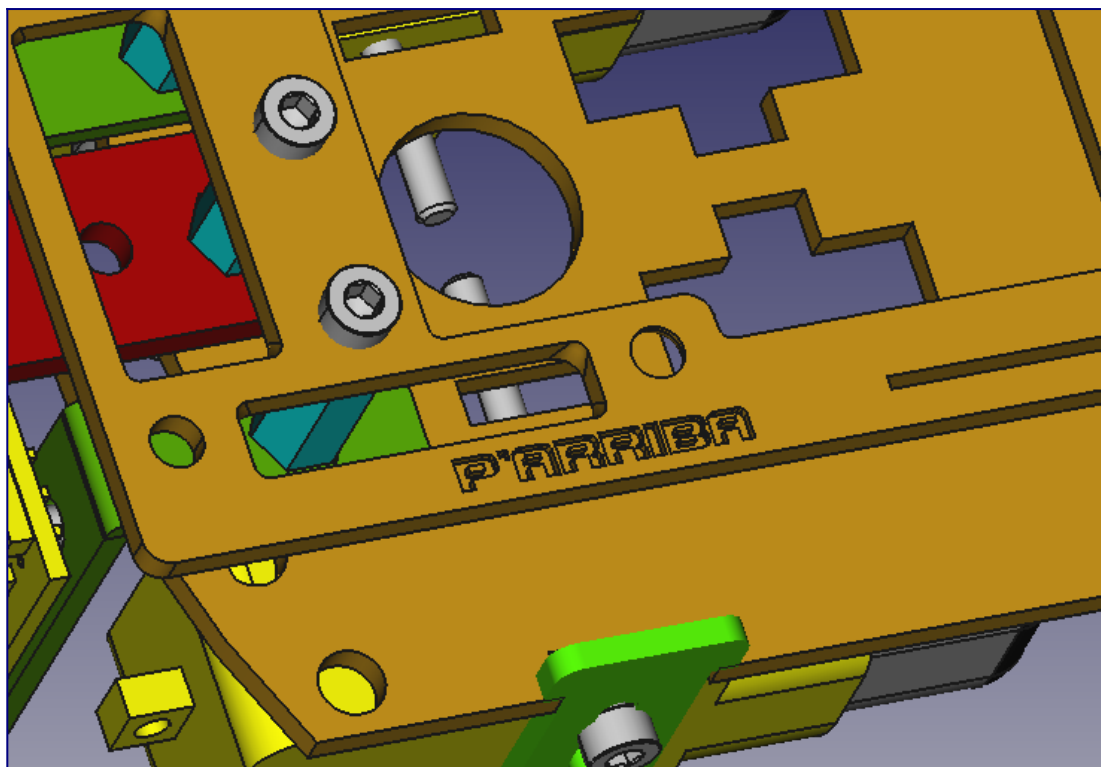
a) Montar los sensores en sus soportes. Cada soporte estará atravesado por un separador y se prepararán para colocarlos en la parte delantera del robot



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.



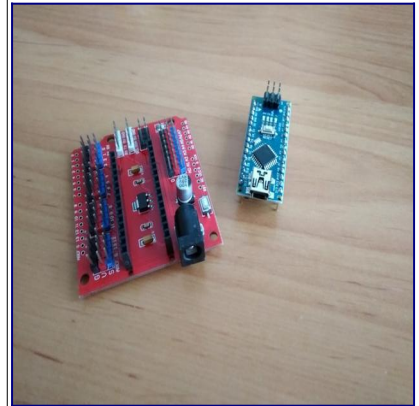
Fijación de la electrónica

Piezas

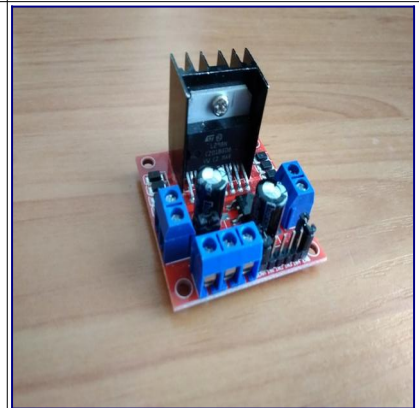
Separadores para la electrónica
(la pieza en STL se llama "separadorSec(x8)")


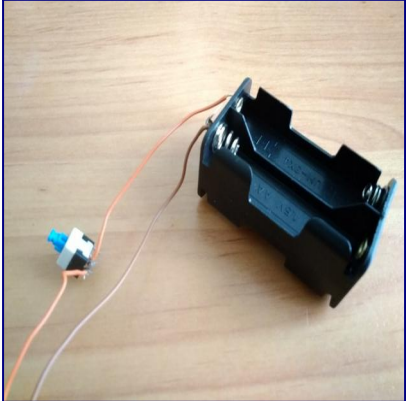


Tarjeta Arduino Nano montada sobre shield de expansión V3



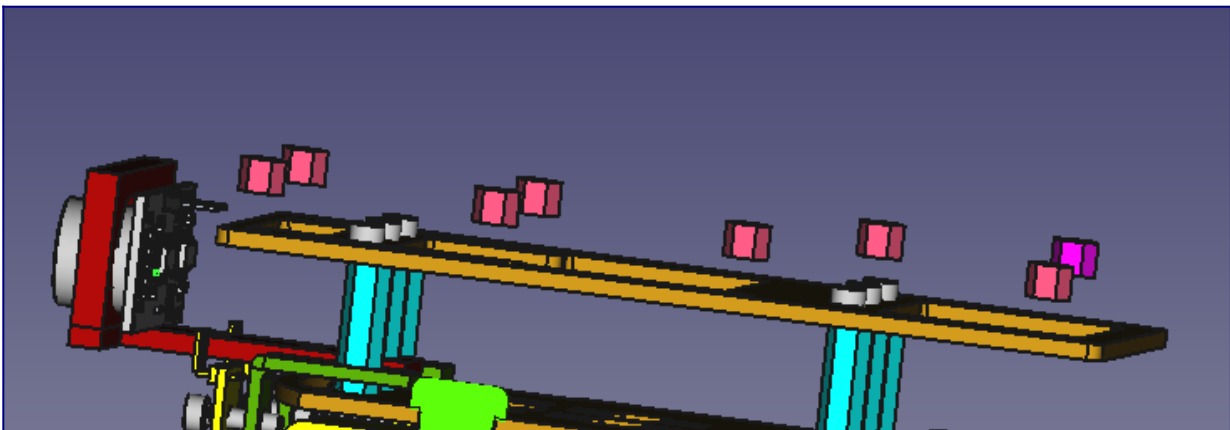
Módulo de control de motores DC L298N



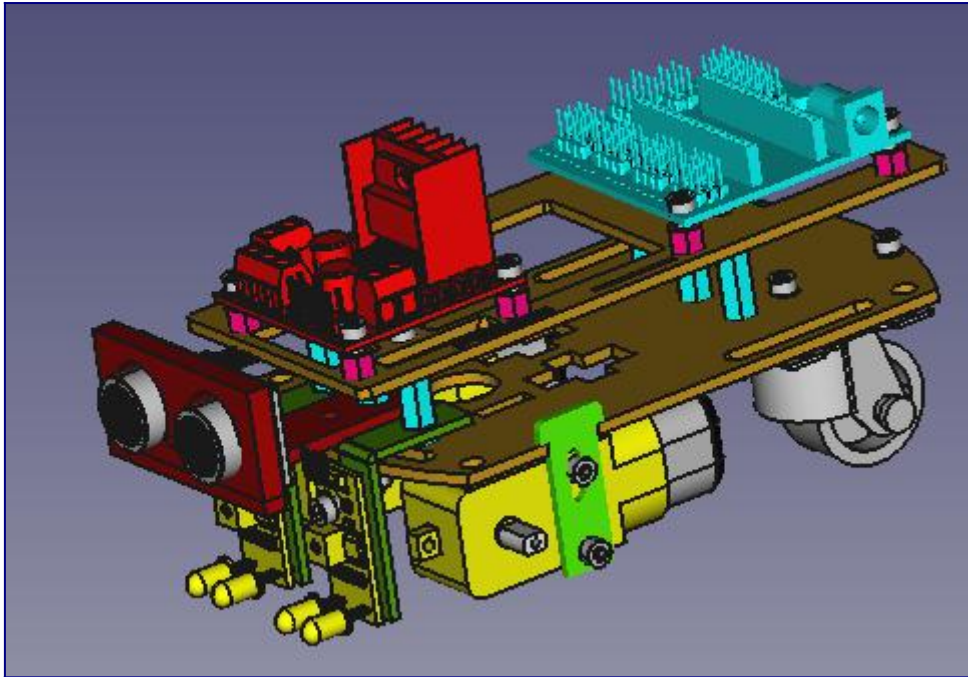
<p>Separadores para la electrónica (la pieza en STL se llama "separadorSec(x8)")</p>	
<p>Portapilas 4xAA con interruptor conectado a cable de positivo</p>	
<p>8 tornillos M3x12 mm para atornillar los módulos al segundo piso</p>	
<p>8 tuercas M3</p>	

Proceso

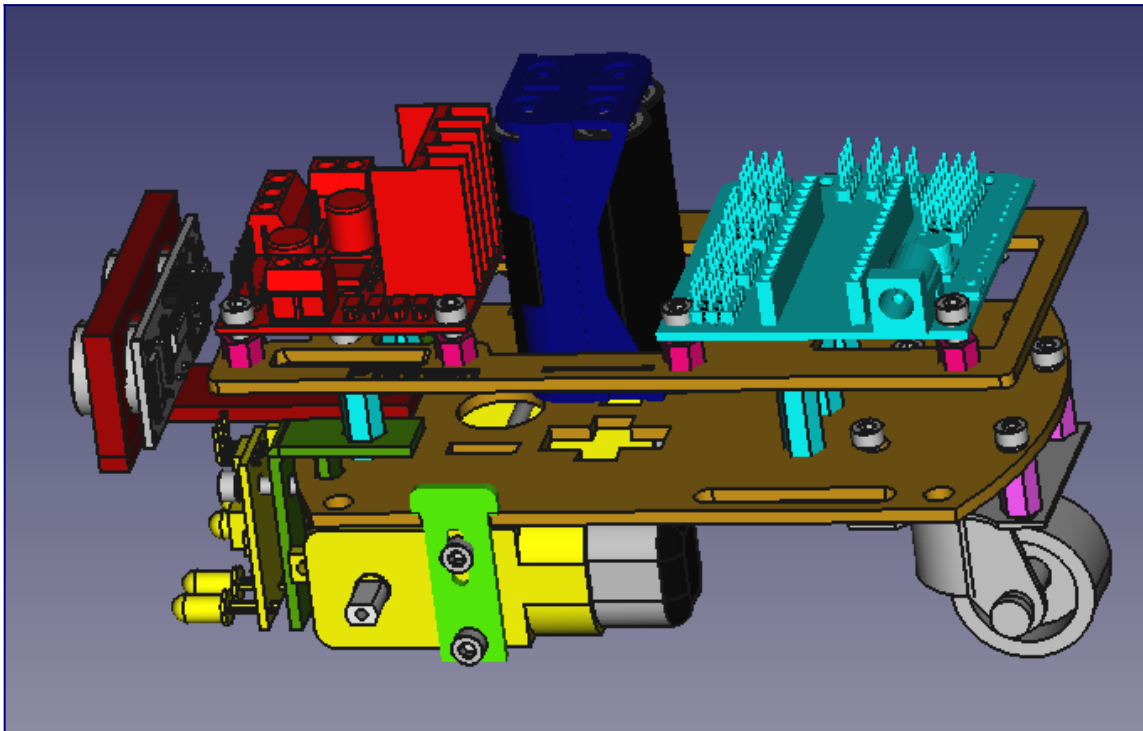
a) Colocar los separadores frente a sus respectivos taladros en el segundo piso



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

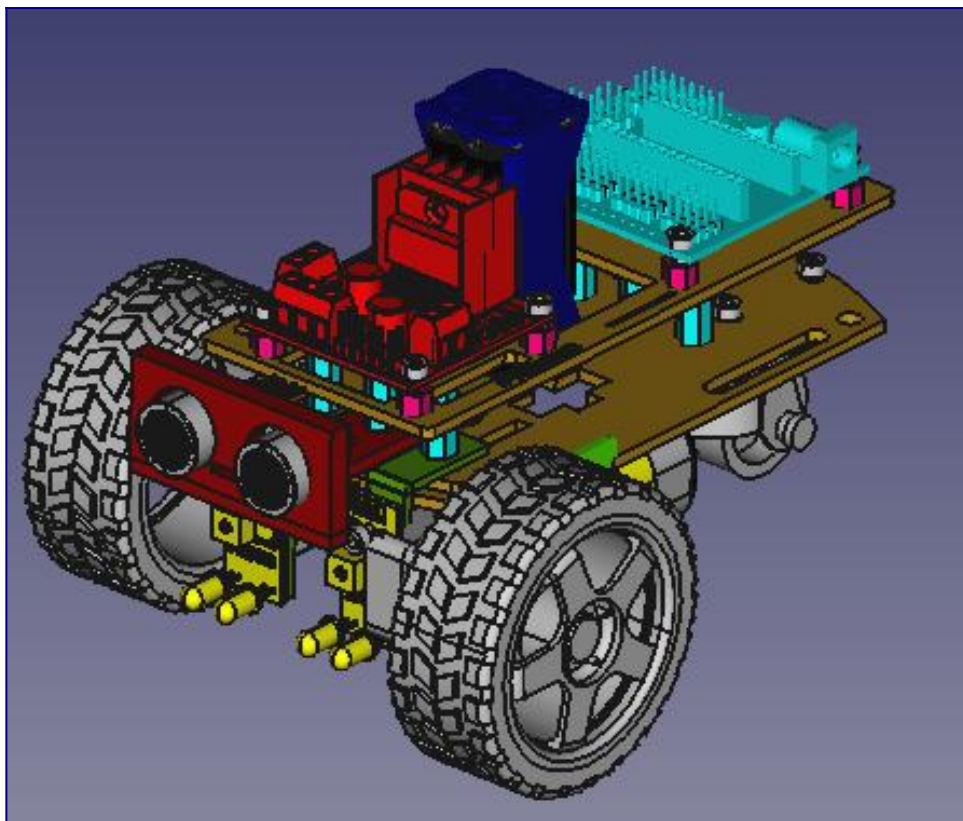


Poner el portapilas en el hueco que queda entre la shield de Arduino y el módulo L298 N



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.

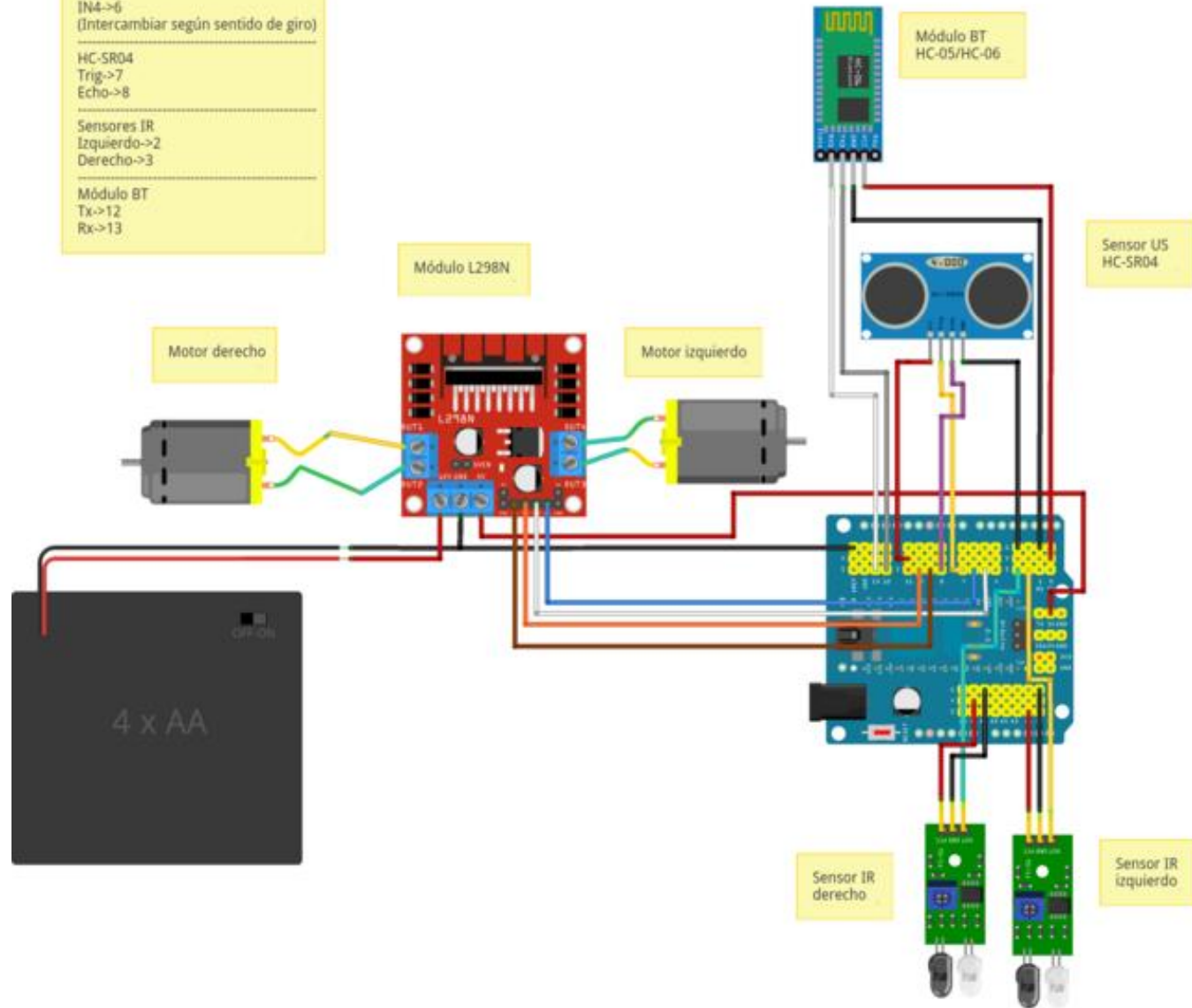
El Masaylo está diseñado para un portapilas de 4 pilas tipo AA, pero incluso este aspecto constructivo queda a disposición de la imaginación y la capacidad de improvisación del estudiante que se disponga a llevar a cabo el proyecto.

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

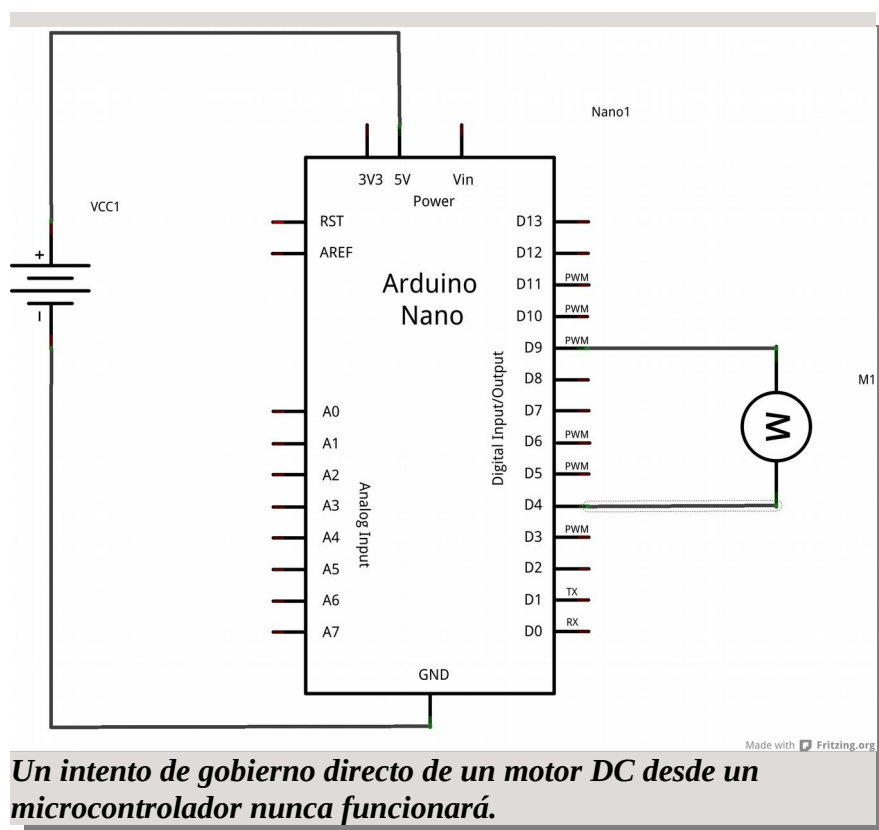
Quizás sorprenda un poco al inexperto que, a diferencia de los montajes a que estamos acostumbrados, el polo positivo de la alimentación no se conecta a +5V ni al pin Vin de la Arduino, sino a la patilla +12 V del L298N. Esto se explicará en el próximo capítulo. Quede también **MUY CLARO QUE LAS TIERRAS (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.

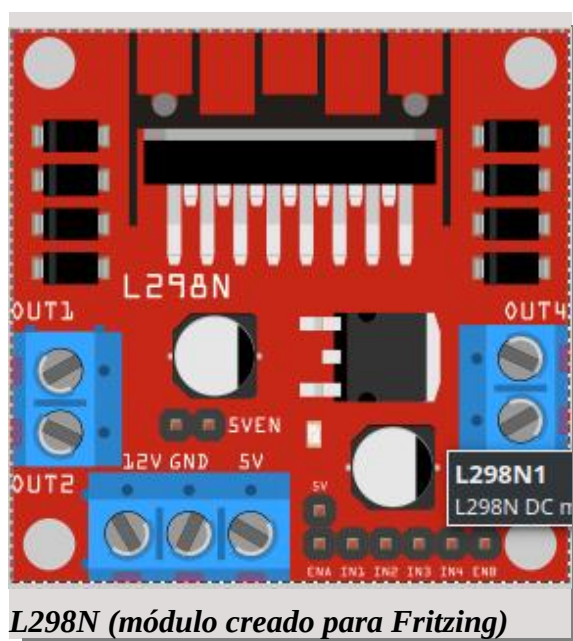


¿Por qué?. Si nuestra tarjeta asocia los estados “HIGH” y “LOW” a 5 V y 0 V, respectivamente, el motor debería moverse. Sin embargo, olvidamos que lo que mueve al motor no es la tensión, sino la intensidad. Y los microcontroladores como los que contiene Arduino pueden mantener las tensiones establecidas 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).

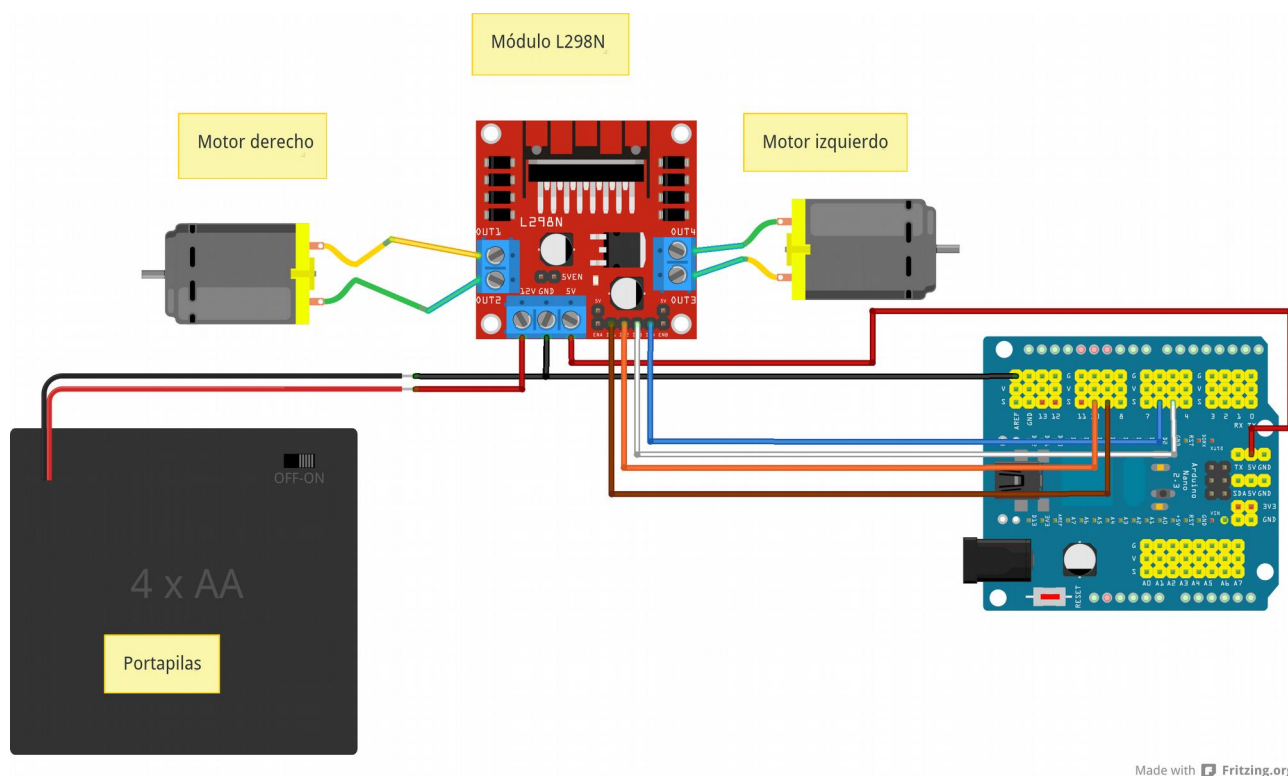
Alimentación de Arduino a través del L298N

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:



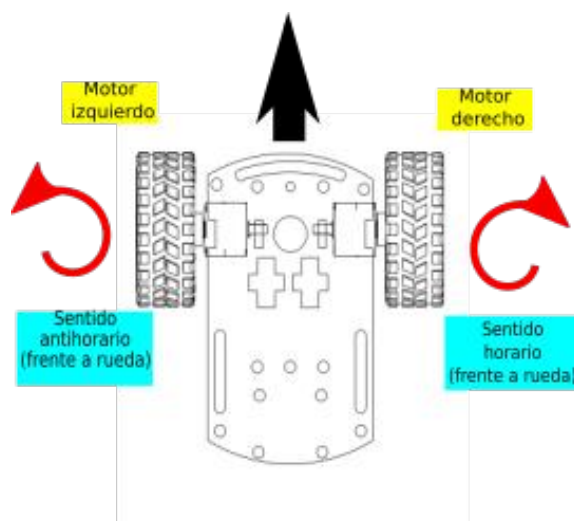
En el siguiente capítulo iniciaremos al lector en el movimiento simple (control todo-nada) del Masaylo mediante funciones sencillas.

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.

Resumamos: 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:

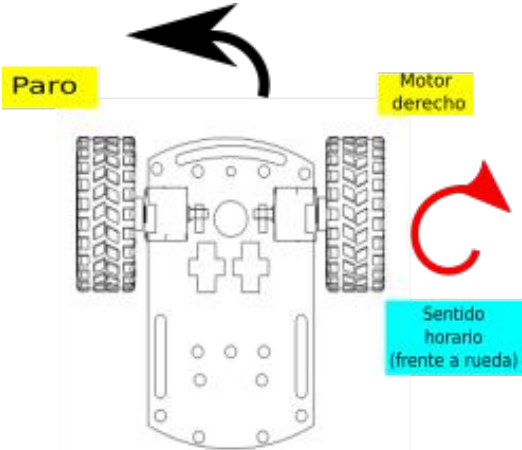
adelante()

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

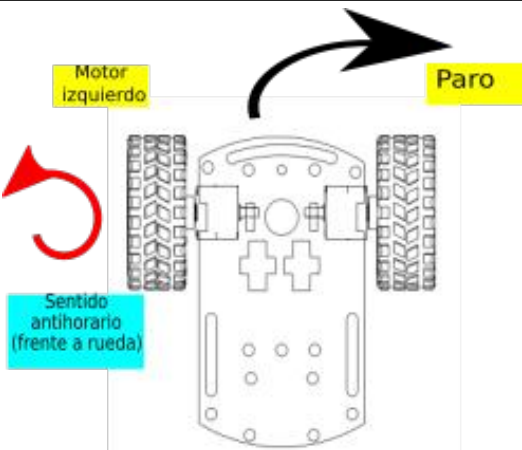
atras()

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

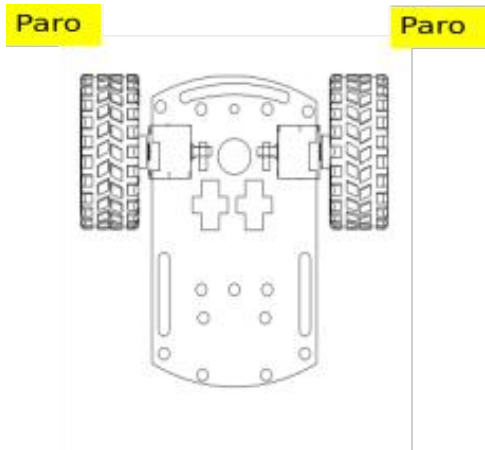
izquierda()

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);
atras();
}

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

Pendiente de redactar