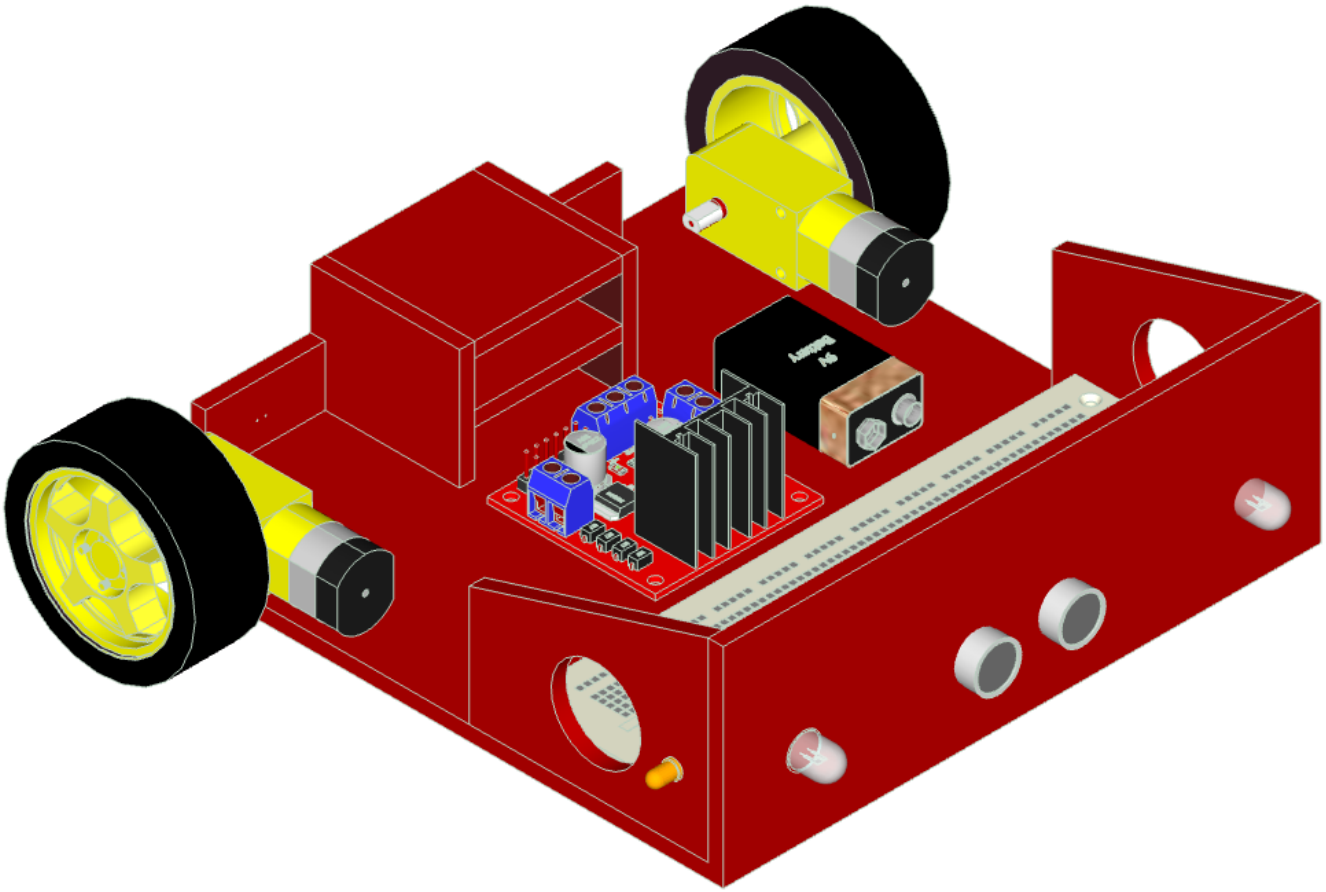
COCHE AUTÓNOMO EVITA-OBSTÁCULOS

(CAE)



PROYECTO REALIZADO POR: RODRIGO PUIA, GUILLEM MONTAÑANA, JORGE MARTÍNEZ

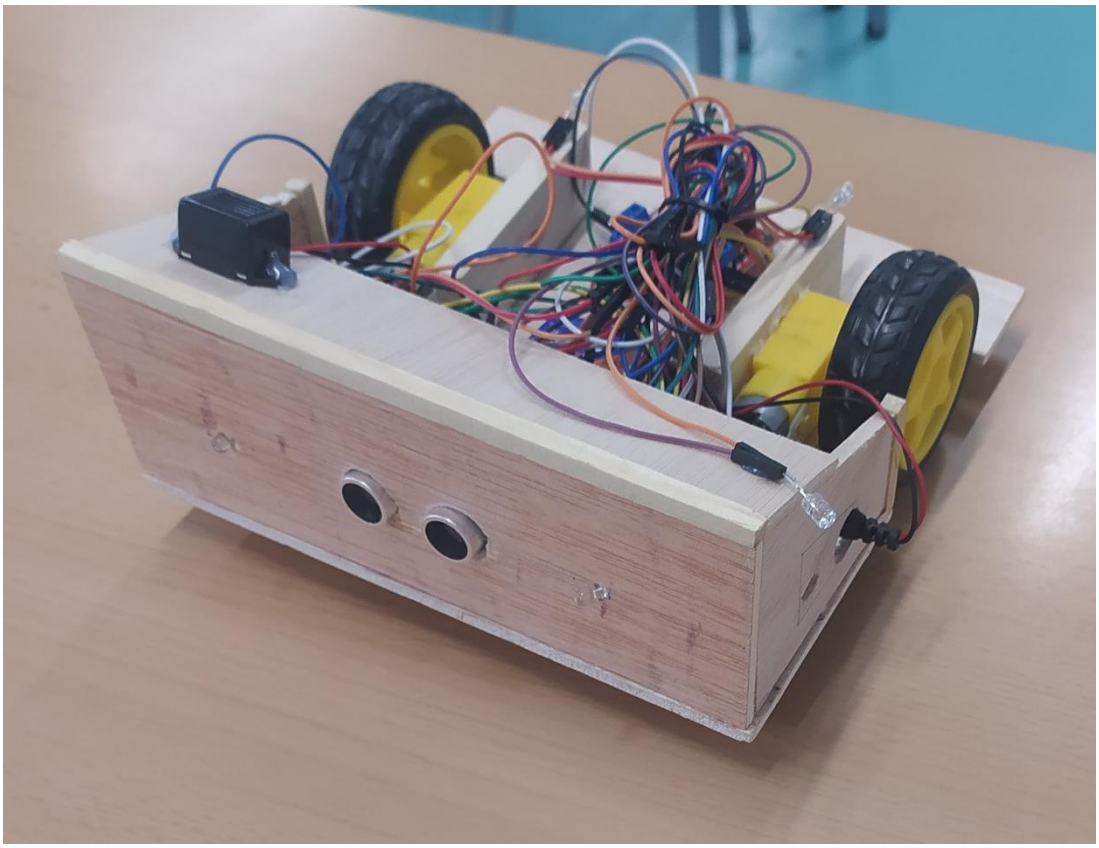
TUTORIZADO POR: ELENA GIL y CAROLINA CABALLERO

IES LA PATACONA (ALBORAYA-VALENCIA)

ÍNDICE

1. Resumen del proyecto

Nuestro coche autónomo evita-obstáculos (CAE) consiste en un vehículo guiado por un sensor ultrasonidos HC-SR04. Al detectar un obstáculo lo esquivará con unos determinados movimientos ya programados. El coche, al no detectar ningún obstáculo, tiene sus LED's delanteros (blancos) encendidos y ambos motores avanzan. En cambio, si detecta un obstáculo, detendrá sus motores y encenderá sus LED's traseros (rojos) mientras inicia la marcha hacia atrás. Pasado un tiempo, un motor hará avanzar su rueda y el otro, al contrario, la hará retroceder, haciendo así que el vehículo gire mientras enciende su LED lateral (naranja). Por último, al esquivar el obstáculo se repetirá esta sucesión de variables hasta que nosotros decidamos lo contrario.



1. Foto de nuestro Coche Autónomo Evita-Obstáculos

2. Introducción

A lo largo de la etapa de la ESO hemos ido aumentando nuestros conocimientos acerca de la programación por bloques, haciendo entre otros proyectos como un robot saca-latas, un vúmetro o un cubo de leds, proyectos que diseñábamos con SketchUp, simulábamos con Tinkercad y después programábamos con mBlock, por bloques, para finalmente subirlos a la placa Arduino mediante ArduinoIDE. Este curso (1º de bachillerato), sin embargo, hemos empezado a programar por código, lo que nos da más libertad y nos ofrece una infinidad de posibilidades para crear y mejorar nuevos proyectos más complejos.

En primer lugar, creamos una casa domótica que se controlaba a través de un mando por infrarrojos. Al ver que concluimos el proyecto con éxito, pensamos que podíamos llegar más lejos y crear un proyecto que realmente tuviera una utilidad aplicable a nuestro día a día. En ese momento vimos un prototipo que empezaron a crear los compañeros del curso pasado y nos llamó la atención, por lo que nos decantamos por la opción de crear un coche autónomo con un sensor de ultrasonidos.

En este curso hemos estudiado los diferentes componentes electrónicos que existen y hemos tratado con ellos haciendo simulaciones y programas simples para ver su funcionamiento. Es por ello por lo que, a la hora de empezar el proyecto, ya teníamos una idea de cómo comenzar a construirlo sin dudar mucho.

3. Objetivos e hipótesis de la investigación

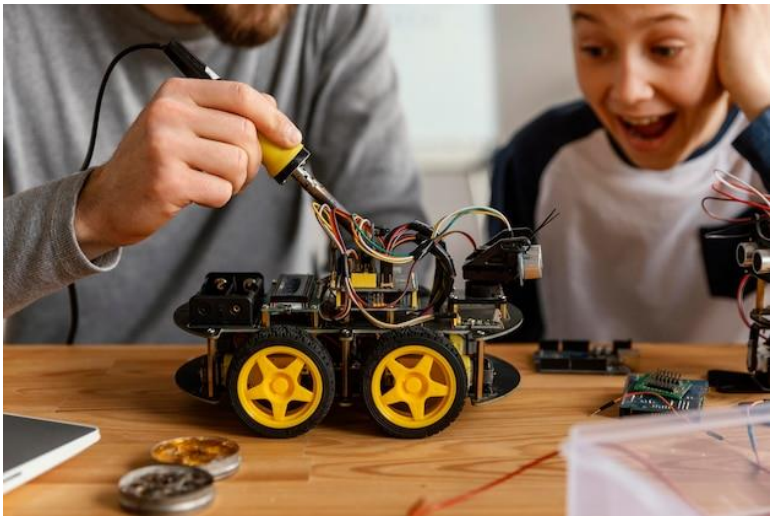
ODS 3: Salud y Bienestar:

Muchas ocupaciones implican movimientos pesados y repetitivos, que suponen para los trabajadores un mayor riesgo de lesiones por este tipo de movimientos. En la realización de las tareas que requieren este tipo de movimientos se podría contar con la ayuda del robot evita obstáculos, que es autónomo, lo cual promueve el bienestar.

2. ODS3: Salud y Bienestar en el trabajo (fuente Freepik)

ODS 4: Educación de Calidad:

En la educación, como ocurre en nuestro caso, los coches autónomos pueden servir de herramienta didáctica para enseñar a estudiantes sobre robótica, ingeniería y programación.



3. ODS4: Educación de Calidad (fuente Freepik)

ODS 13: Acción por el Clima:

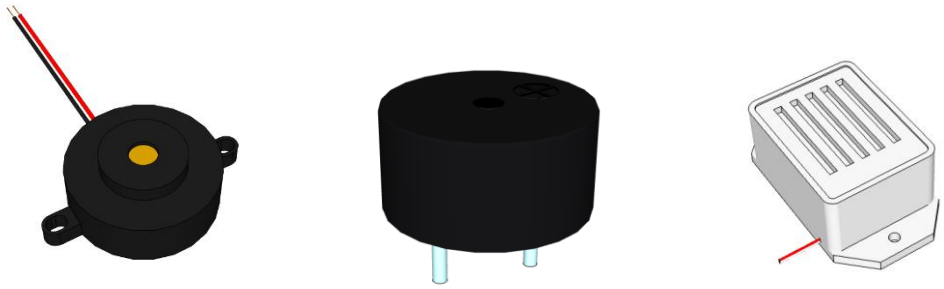
La maqueta de este coche autónomo es eco-friendly ya que, la única energía que necesita es la de una pila, que será siempre una pila recargable para reducir la contaminación.

4. ODS13: Acción por el Clima (fuente Freepik)

Este proyecto tiene como objetivo englobar todos los conocimientos aprendidos durante nuestro progreso como alumnado de la asignatura de Tecnología e Ingeniería I. Aprendimos, los dos primeros trimestres, a programar por separado los diferentes componentes de los que consta el proyecto: un motor, un zumbador, un LED, un sensor ultrasonidos y a trabajar con el módulo controlador de motores, que permite encender y controlar dos motores de corriente continua desde Arduino, variando tanto la dirección como la velocidad de giro. Tras dos trimestres enteros aprendiendo programación por código era la hora de demostrar todo lo que sabíamos hacer en tan solo un proyecto de un mes. Además, la idea de este proyecto es diversa ya que se podría utilizar para la limpieza del hogar o como juguete de entretenimiento.

Esta idea no es nueva y nace a partir de una metodología de trabajo llevada a cabo desde que entramos a nuestro instituto. En la clase de tecnología del IES La Patacona no se utiliza mucho el libro y el curso se enfoca en uno dos o tres proyectos dependiendo del curso del que hablemos. Como se ha mencionado anteriormente, todos los alumnos aprendimos individualmente a programar por código los diferentes elementos que íbamos a necesitar, por separado.

Primero, tenemos el zumbador, es un dispositivo que permite convertir una señal eléctrica en sonido. Emite un sonido o zumbido en un mismo tono. Está polarizado, lo cual quiere decir que hay una patilla positiva (cable rojo, señalada con + o Vcc) y otra negativa (cable negro, sin señal o Gnd). Si se conecta mal no sonará y habrá que cambiarlo de posición.



5. Diferentes modelos de zumbadores (fuente SketchUp)

Podemos distinguir dos tipos de zumbadores:

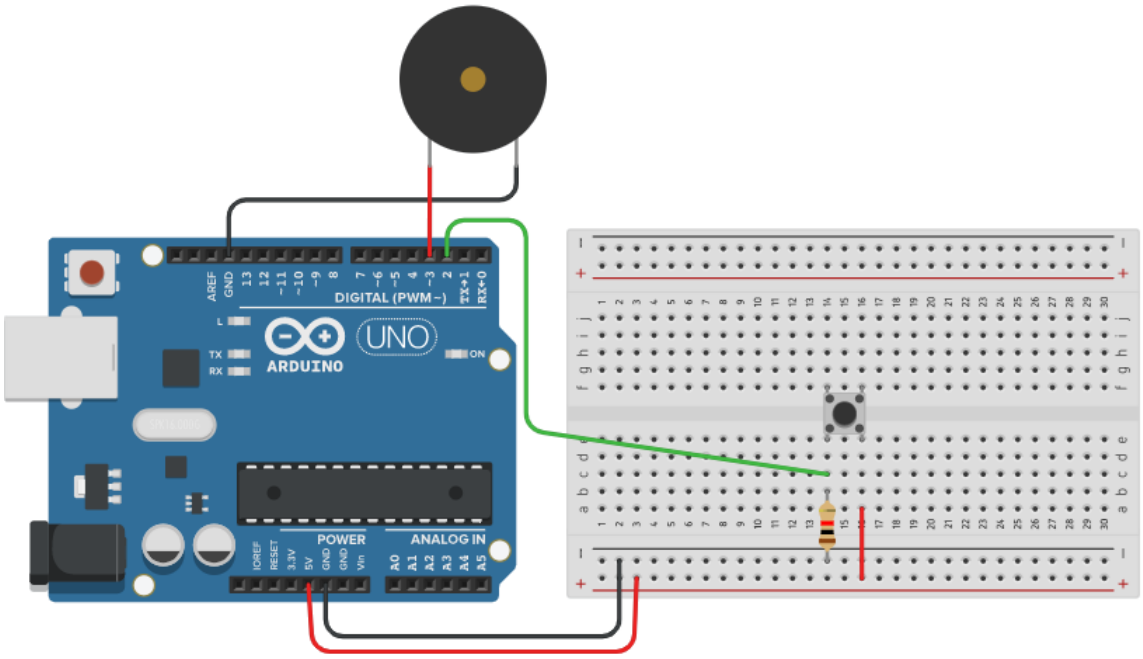
- Activos: Generan un tono de una frecuencia determinada al ser atravesadso por una corriente continua. Si se conectan directamente a Vcc y GND emiten un sonido. Si la patilla negativa (o cable negro) se conecta a GND de la placa Arduino, y la patilla positiva o cable rojo a un pin de salida (cualquier pin), al poner esa salida en alto se genera sonido y al ponerla en bajo no emite sonido.
- Pasivos: Para que emita un tono, se le debe enviar una señal analógica, de una frecuencia determinada. Si no se indica la frecuencia, el zumbador hará un click al activarse y desactivarse, pero no sonará.

Puede llegar a emitir melodías si usamos la función tone(), que viene incorporada en el IDE de Arduino, y que permite cambiar tanto la frecuencia, como su duración. Al usar la función tone se puede conectar a cualquier pin de Arduino, sin necesidad de que sea PWM.

PRÁCTICA: Alarma con zumbador activo

La alarma funciona al oprimir un pulsador.

a) CIRCUITO: Rbotón: 1 kΩ (rojo, negro, marrón)



6. Simulación en Tinkercad de alarma con zumbador activo

b) PROGRAMA:

```
int BOTON_pin = 2;     // pin del botón
int BUZZER_pin = 3;    // pin del buzzer
int Valor_BOTON;
void setup() {
   pinMode(BOTON_pin, INPUT);   // pin del botón como entrada
   pinMode(BUZZER_pin, OUTPUT);  // pin del buzzer como salida
   }
void loop() {
   Valor_BOTON = digitalRead (BOTON_pin)
   if (Valor_BOTON == HIGH) { // si es HIGH, el botón esta oprimido
     for(int i=0 ; i<5 ; i++)
       digitalWrite (BUZZER_pin, HIGH);
```

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

El elemento estrella del proyecto es el **sensor ultrasonidos HC-SR04**, que permite detectar obstáculos a cierta distancia. Un sensor de ultrasonidos es un dispositivo para medir distancias. Su funcionamiento se base en el envío de un pulso de alta frecuencia, no audible por el ser humano. Este pulso rebota en los objetos cercanos y es reflejado hacia el sensor, que dispone de un micrófono adecuado para esa frecuencia. Midiendo el tiempo entre pulsos, conociendo la velocidad del sonido, podemos estimar la distancia del objeto contra cuya superficie impacto el impulso de ultrasonidos. Los sensores de ultrasonidos son sensores baratos, y sencillos de usar. El rango de medición teórico del sensor HC-SR04 es de 2cm a 400 cm, con una resolución de 0.3cm. En la práctica, sin embargo, el rango de medición real es mucho más limitado, en torno a 20cm a 2 metros. Los sensores de ultrasonidos son sensores de baja precisión. La orientación de la superficie a medir puede provocar que la onda se refleje, falseando la medición. Además, no resultan adecuados en entornos con gran número de objetos, dado que el sonido rebota en las superficies generando ecos y falsas mediciones. Tampoco son apropiados para el funcionamiento en el exterior y al aire libre. Pese a esta baja precisión, que impide conocer con precisión la distancia a un objeto, los sensores de ultrasonidos son ampliamente empleados. En robótica es habitual montar uno o varios de estos sensores, por ejemplo, para detección de obstáculos, determinar la posición del robot o resolver laberintos.

- Pines del sensor de ultrasonidos

Es importante fijarse en las patillas que tiene el sensor. Por un lado, las típicas GND (tierra) y Vcc (alimentación 5 V) y por otro lado tiene dos patillas que son el Trigger (disparo) y Echo (eco). Estas dos patillas son las importantes. Estas las tenemos que conectar a las entradas/salidas digitales. Por un lado, Trigger va a funcionar en modo salida y por otro lado Echo va a funcionar en modo entrada.

- Funcionamiento del sensor de ultrasonidos

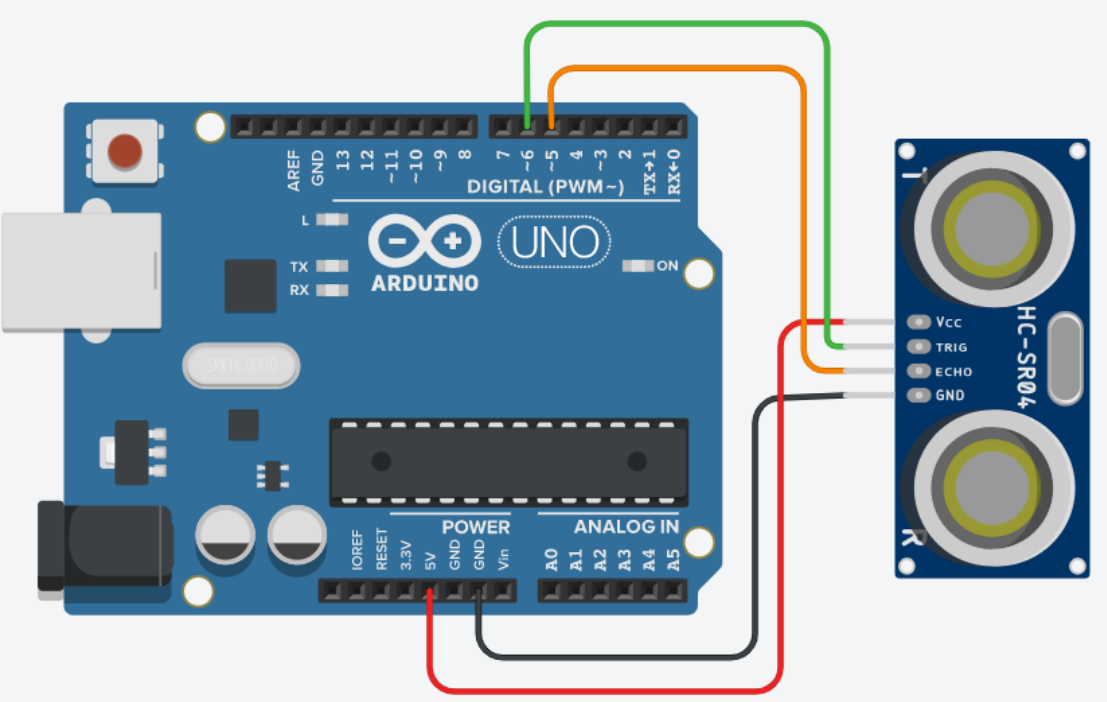
El Trigger manda un pulso de ultrasonidos y el Echo se encarga en recibirlo. Por lo tanto, tenemos el tiempo que tarda en recibirlo y la velocidad, la del sonido, así de fácil. Como hemos tenido que ir hasta el objeto y volver lo estamos calculando por dos así que hay que dividir por dos. Esta fórmula la tenemos también en la ficha técnica. Distancia = (Tiempo en estado HIGH * Velocidad del sonido) / 2En el caso de este sensor, hay que iniciar la comunicación entre el Trigger y el Echo para ello debemos mandar un pulso de 10 ms (milisegundos). Resumen de las características:

- ✓ Voltaje de trabajo 5 V
- ✓ Corriente de trabajo 15 mA
- ✓ Rango de precisión entre 3 cm y 400 cm (200 cm en realidad)
- ✓ La precisión puede llegar a 3 mm en su máximo valor
- ✓ Angulo de medición 15º.

7. Sensor ultrasonidos HC-SR04 (fuente SketchUp)

PRÁCTICA: Imprimir la distancia en el monitor serie

a) CIRCUITO:



8. Simulación en Tinkercad del sensor ultrasonidos HC-SR04 (imprimir mediciones en el monitor serie)

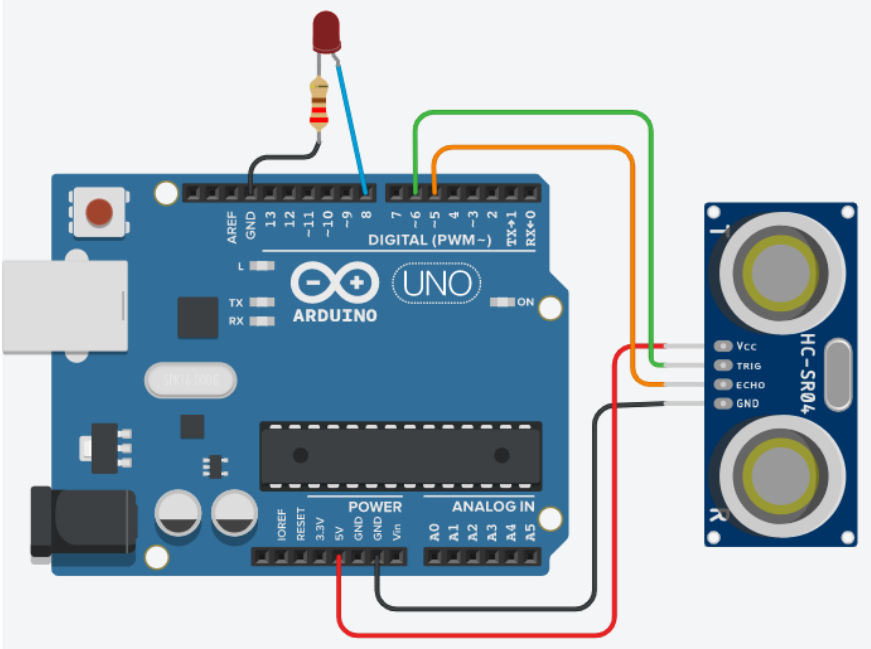
b) PROGRAMA: Escribe la distancia en el monitor serie.

Para activar el sensor necesitamos generar un pulso eléctrico en el pin Trigger (disparador) de al menos 10µs. Después pondremos el pin a Low. Posteriormente usamos la función "pulseIn" para obtener el tiempo requerido por el pulso para volver al sensor. Finalmente, convertirmos el tiempo en distancia mediante la ecuación correspondiente.
Observar que intentamos emplear siempre aritmética de enteros, evitando usar números en coma flotante. Esto es debido a que las operaciones en coma flotante ralentizan mucho el procesador, y suponen cargar un gran número de librerías en memoria.

```
  /* Programa que envía mediante el Monitor Serial el valor de distancia leído por el sensor ultrasónico HC-SR04
*/
  int TRIG = 6;                    // trigger en pin 6
  int ECO = 5;                     // echo en pin 5
  int DURACION;
  int DISTANCIA;
void setup(){
  pinMode(TRIG, OUTPUT);           // trigger como salida
  pinMode(ECO, INPUT);                    // echo como entrada
  Serial.begin(9600);              // inicializacion de comunicacion serial a 9600 bps
}
void loop(){
 digitalWrite(TRIG, HIGH);                // generacion del pulso a enviar
 delay(1);
 digitalWrite(TRIG, LOW);
 DURACION = pulseIn(ECO, HIGH);           // con funcion pulseIn se espera recibir un pulso alto en Echo
 DISTANCIA = DURACION / 58.2;             // distancia medida en centimetros
 Serial.print("Distancia: ");
 Serial.println(DISTANCIA);               // envio de valor de distancia por monitor serial
 delay(200);                              // demora entre datos
 }
```

PRÁCTICA: Detector de obstáculos

a) CIRCUITO:



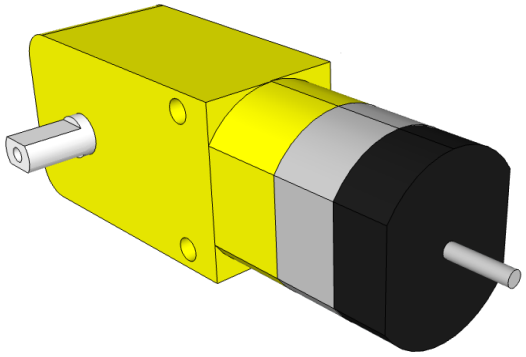
9. Simulación en Tinkercad del detector de obstáculos

b) PROGRAMA: Detector de obstáculos.

```
/*  Programa que envía mediante el Monitor Serial el valor de distancia leído por el sensor ultrasónico HC-SR04 y enciende y apaga LED dentro del rango de 0 a 20 cms */
  int TRIG = 6;                    // trigger en pin 6
  int ECO = 5;                     // echo en pin 5
  int LED = 8;                     // LED en pin 8
  int DURACION;
  int DISTANCIA;
void setup()  {
  pinMode(TRIG, OUTPUT);         // trigger como salida
  pinMode(ECO, INPUT); // echo como entrada
  pinMode(LED, OUTPUT);          // LED como salida
  Serial.begin(9600);            // inicializacion de comunicacion serial a 9600 bps
 }
void loop() {
  digitalWrite(TRIG, HIGH);              // generacion del pulso a enviar
  delay(1);                              // al pin conectado al trigger
  digitalWrite(TRIG, LOW);               // del sensor
  DURACION = pulseIn(ECO, HIGH);         // con funcion pulseIn se espera un pulso alto en Echo
  DISTANCIA = DURACION / 58.2;           // distancia medida en centimetros
  Serial.println(DISTANCIA);             // envio de valor de distancia por monitor serial
```

```
  delay(200);                                  // demora entre datos
  if (DISTANCIA <= 20 && DISTANCIA >= 0){      // si distancia entre 0 y 20 cms.
    digitalWrite(LED, HIGH);                   // enciende LED
    delay(DISTANCIA * 10);                     // demora proporcional a la distancia
    digitalWrite(LED, LOW);                    // apaga LED
    }
}
```

Por otro lado, es necesario conocer el funcionamiento de los motores DC y el módulo controlador de motores, empecemos por el primer mencionado. Un motor DC, o motor de corriente continua, es el tipo más común de motor. Los motores de corriente continua normalmente tienen dos bornes que se conectan, uno al polo positivo y otro al negativo de una pila o batería. Si conectas estos dos cables directamente a una batería, el motor girará. Si cambias los cables, el motor girará en la dirección opuesta. Un motor DC no se puede accionar desde una placa Arduino, pues ésta no saca suficiente potencia para ello. Se debe utilizar una fuente de alimentación externa para alimentar el motor y la placa Arduino para enviar las señales, además se usan controladores para cambiar el sentido del motor y su velocidad.



10. Motor DC (fuente SketchUp)

Por consiguiente, es necesario conoce el controlador de motores L298N puente-H. Es un controlador (driver) de motores, que permite encender y controlar dos motores de corriente continua desde Arduino, variando tanto la dirección como la velocidad de giro.

Conexiones del módulo:

a) Empecemos explicando la forma de alimentar el módulo: Usaremos una fuente de alimentación externa cuyo positivo lo conectaremos a V motor (que debe ser entre 5 y 12 V) y cuyo negativo lo conectaremos a GND (IMPORTANTE: este GND también debe ir conectado a GND de Arduino). A V lógico no se conecta nada (sería una salida de 5V y 500 mA si lo quisiéramos utilizar).

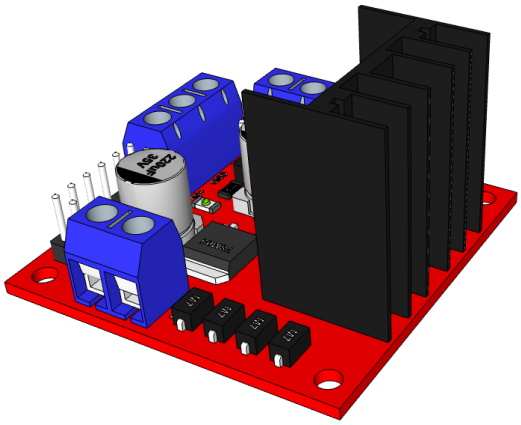
b) Conexión de los motores: Al módulo se pueden conectar 1 o dos motores, que se conectarán a los bornes motor A (OUT 1 y OUT 2), o motor B (OUT 3 y OUT 4). Según cómo conectemos los dos cables que salen de cada motor, estos girarán en sentido horario o antihorario.

c) Para el control del módulo: Los pines ENA, IN1, IN2correspondes a las entradas para controlar el MOTOR A:

- ✓ Con IN1 y IN2controlamos el sentido de giro del motor A (IN1 controla a OUT1 y IN2 controla a OUT2):
- ✓ Si a IN1 le llega una señal HIGH (valor 1) y a IN2 le llega una señal LOW (valor 0) el motor A gira en un sentido.
- ✓ Si a IN1 le llega una señal LOW (0) y a IN2 le llega una señal HIGH (1) el motor A gira en el sentido contrario. Esto es así porque el 1 pone al pin al voltaje de la fuente de alimentación y el 0 lo pone a GND.
- ✓ El pin ENA lleva un jumper (plástico de color negro), y sólo usaremos este pin si queremos controlar la velocidad del motor A. En este caso quitaríamos el jumper y conectaríamos el pin a una salida PWM de Arduino, que nos permitirá regular la velocidad según la señal enviada.

Si quitamos el jumper y no conectamos el pin ENA a una salida PWM de Arduino el motor no funcionará (lo hemos deshabilitado). Luego, si no vamos a regular la velocidad del motor el jumper del pin ENA debe estar puesto, para que el motor está habilitado y funcione.

De igual manera ENB, IN3, IN4permiten controlar el MOTOR B: Con IN3 y IN4 controlamos el sentido de giro del motor B (IN3 controla a OUT3 y IN4 controla a OUT4). ENB controla la velocidad del motor B, de modo similar a ENA.



11. Controlador de motores L298N puente-H (fuente SketchUp)

Así bien, nuestro objetivo, a partir de todo esto, era construir un robot con un sensor de ultrasonidos, que detectara y evitara los obstáculos. Al detectar un obstáculo girará hacia un lado. A continuación, irá hacia delante hasta detectar un nuevo obstáculo. Se utilizará contrachapado.

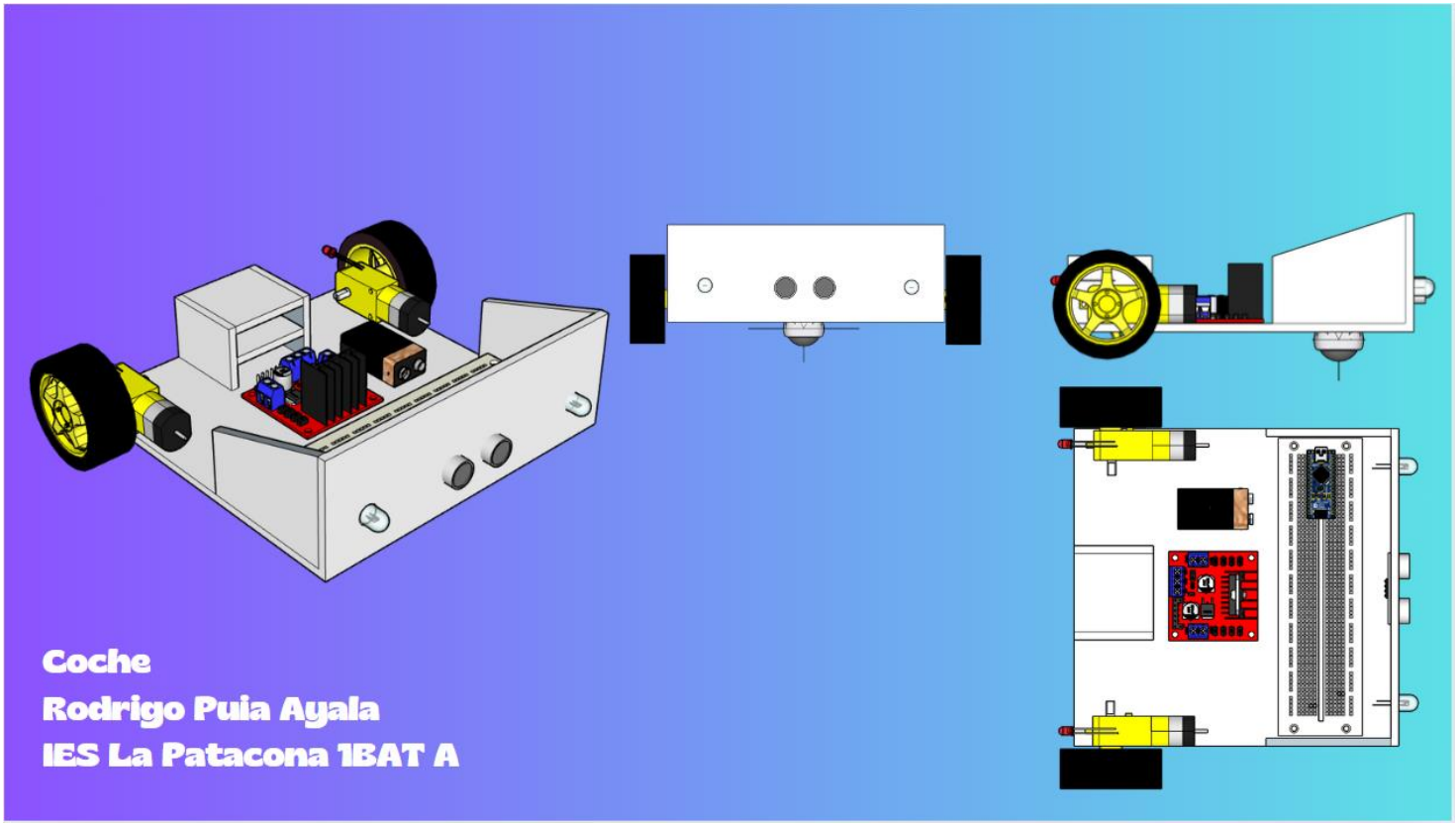
b) PROGRAMA:

```
    /* Inicialmente hacemos girar el motor en sentido horario, luego en antihorario y después lo detenemos, esto se repite constantemente. Cada estado lo enviamos por comunicación serial a la PC. Si el sentido de giro no corresponde al que se muestra en el monitor serial, simplemente invertir la polaridad de la conexión del motor, o también cambiar las funciones en el programa  */
    int PinIN1 = 7;
    int PinIN2 = 6;
void setup() {
      // inicializar la comunicación serial a 9600 bits por segundo:
      Serial.begin(9600);
      // configuramos los pines como salida
      pinMode(PinIN1, OUTPUT);
      pinMode(PinIN2, OUTPUT);
    }
void loop() {
      MotorHorario();
      Serial.println("Giro del Motor en sentido horario");
      delay(1000);
      MotorAntihorario();
      Serial.println("Giro del Motor en sentido antihorario");
      delay(1000);
      MotorStop();
      Serial.println("Motor Detenido");
      delay(1000);
      }
void MotorHorario()
    {
      digitalWrite (PinIN1, HIGH);
      digitalWrite (PinIN2, LOW);
    }
void MotorAntihorario()
    {
      digitalWrite (PinIN1, LOW);
      digitalWrite (PinIN2, HIGH);
    }
void MotorStop()
    {
      digitalWrite (PinIN1, LOW);
      digitalWrite (PinIN2, LOW);
    }
```

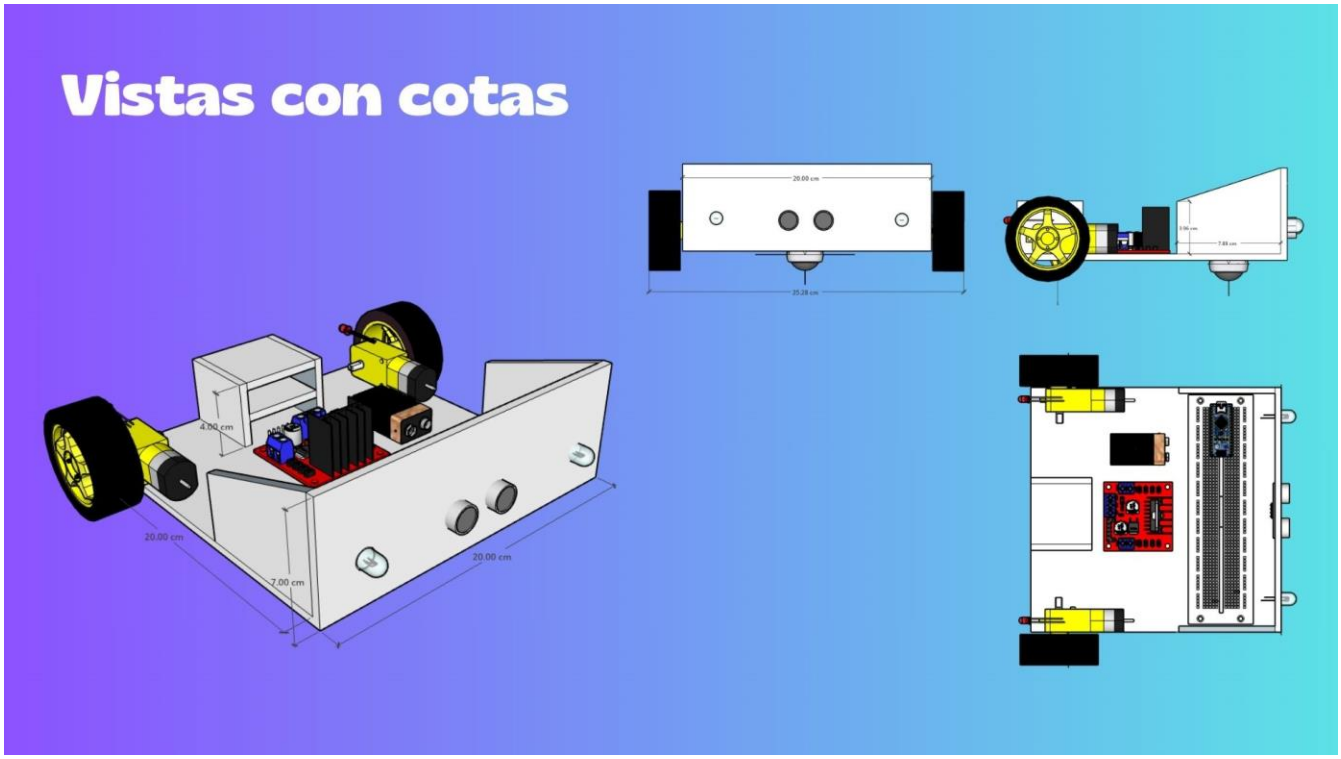
4. Materiales y métodos empleados

Diseño con Sketch Up

El primer paso del proyecto consistía en hacernos una idea de cómo sería en cuanto a tamaño y distribución de los componentes del proyecto. Lo mejor es que la forma de este era libre, por tanto, cada compañero tenía su propio diseño de coche. Entonces, como hicimos dos grupos de tres personas para hacer dos coches, cada grupo debía elegir un diseño de coche entre los tres que había. La primera idea de cómo será el proyecto no siempre coincide con el resultado final. Esto viene por una serie de inconvenientes mecánicos e informáticos de los que hablaremos después.

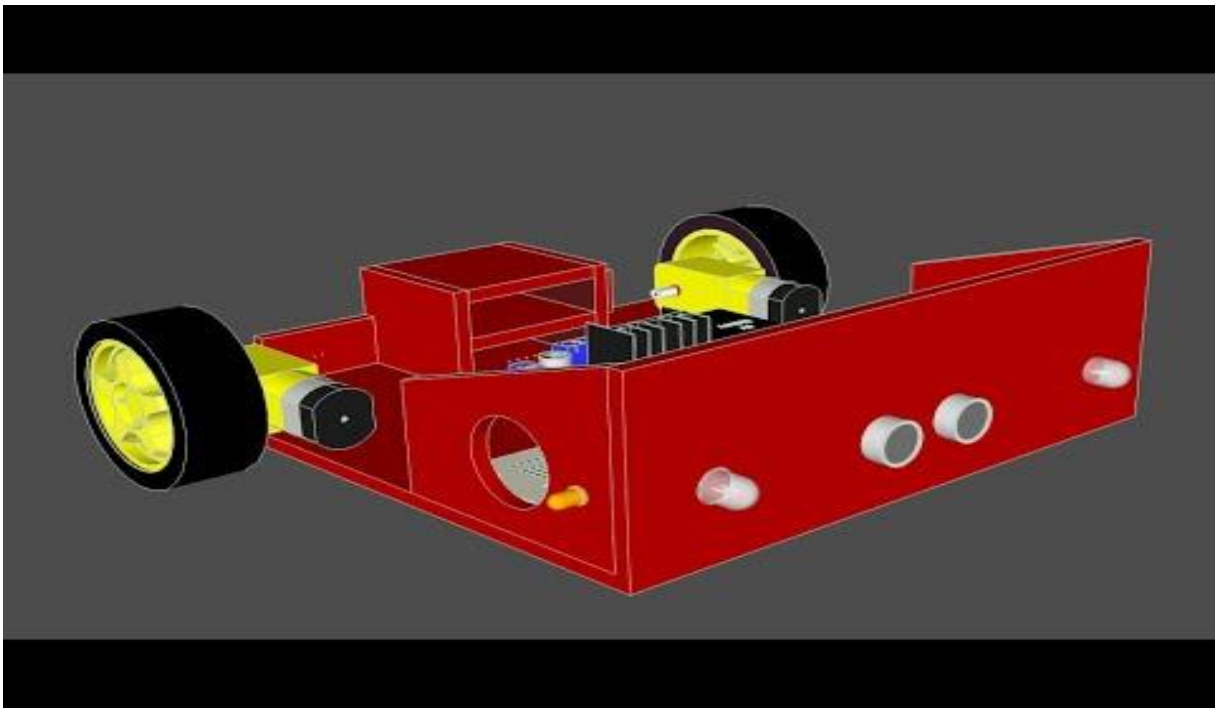


12. Diseño con SketchUp del Coche Evita Obstáculos



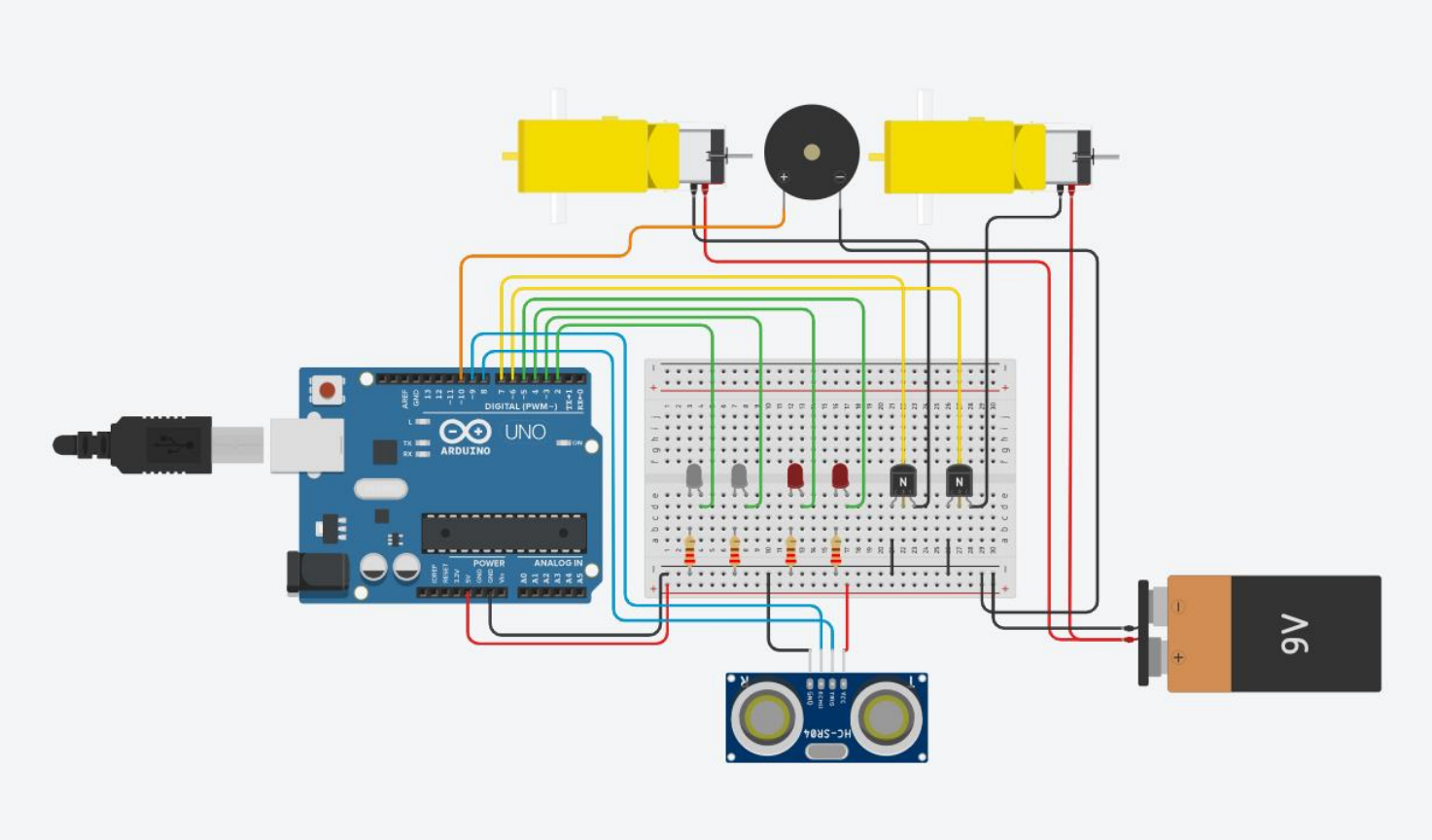
13. Diseño acotado del Coche Evita Obstáculos con SketchUp

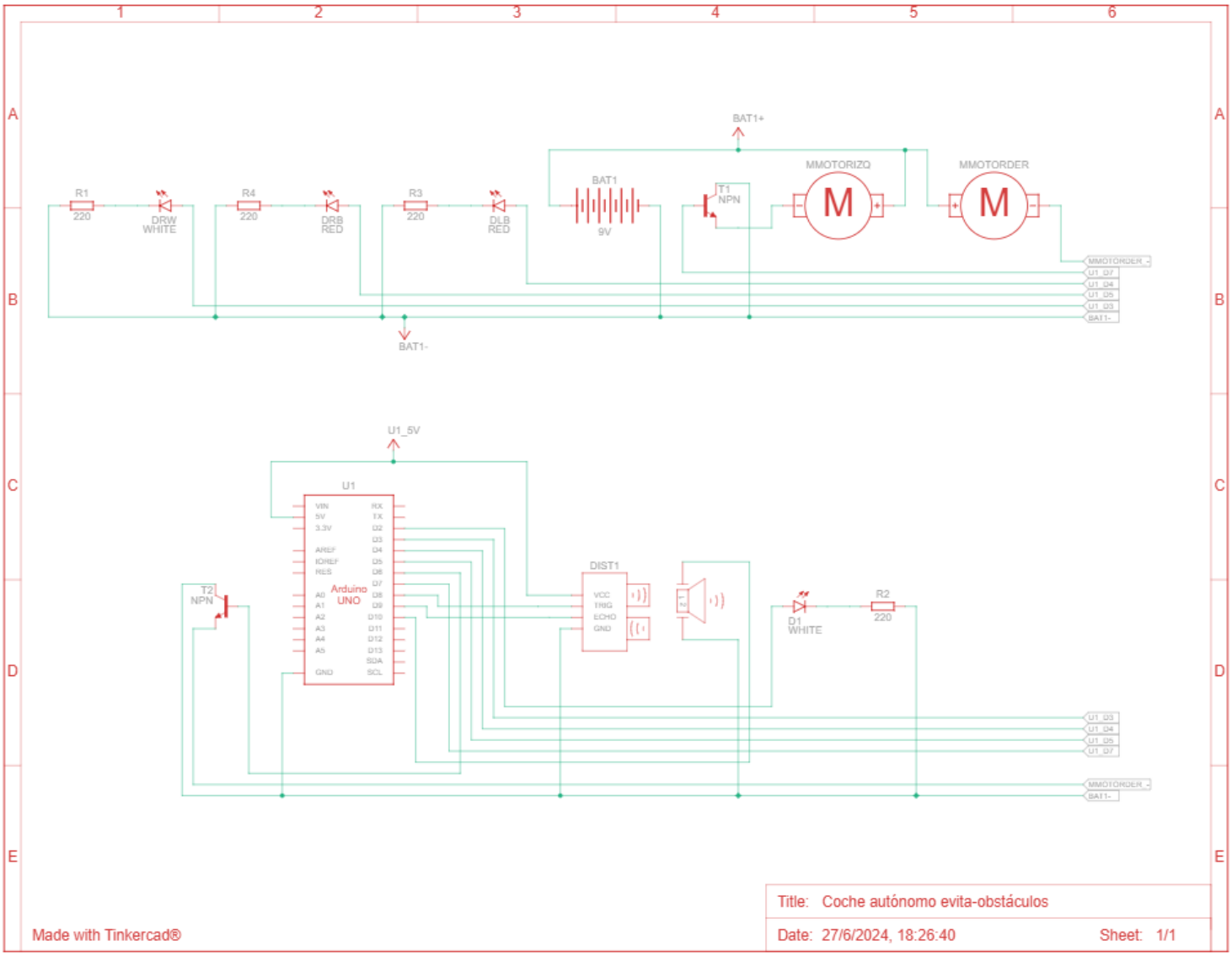
https://youtu.be/ylwNwn6kQ6w



Simulación con Tinkercad

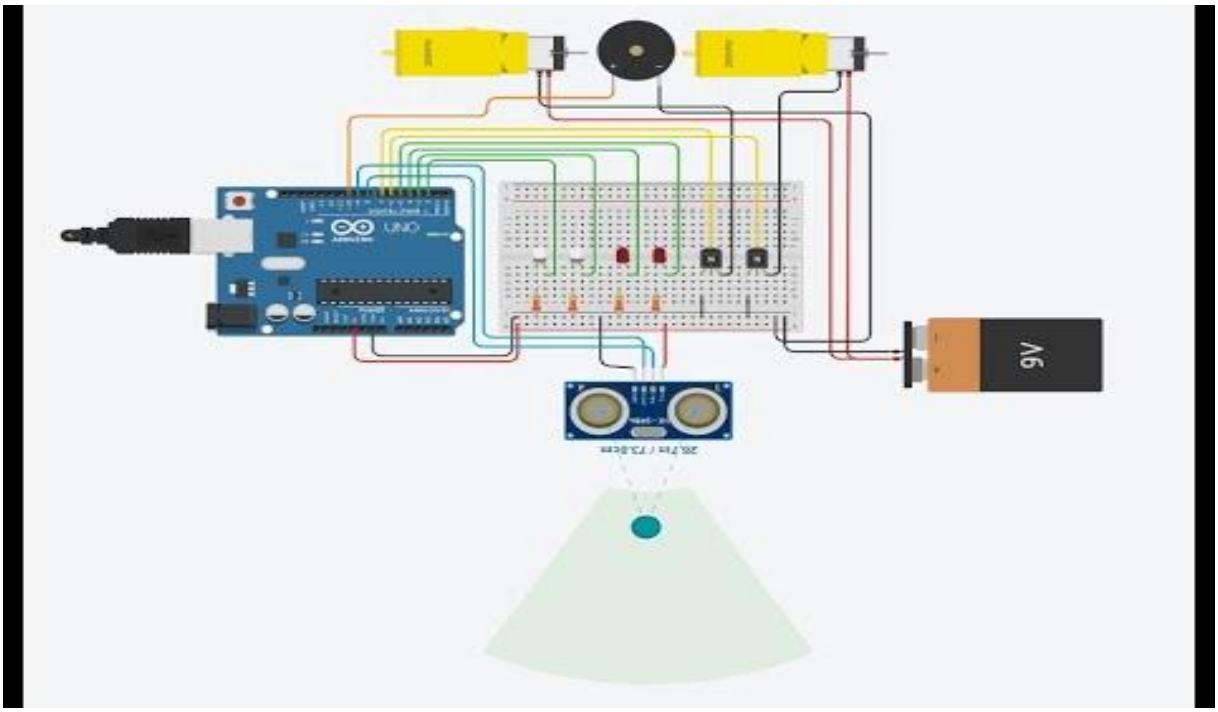
Este código de Tinkercad es para un robot que utiliza un sensor ultrasónico para medir la distancia a los obstáculos y reaccionar en consecuencia. Inicializa pines para LED's, motores, un sensor ultrasónico y un buzzer. En el ciclo principal, el sensor ultrasónico mide la distancia y la envía al monitor serial. Si la distancia es mayor a 20 cm, los LED's frontales blancos se encienden, los traseros rojos se apagan y ambos motores avanzan. Si la distancia es menor a 20 cm, los LED's frontales se apagan, los traseros se encienden, el buzzer emite un sonido intermitente y el robot gira deteniendo el motor derecho y activando el motor izquierdo durante 6 segundos para evitar el obstáculo.





14. Simulación en Tinkercad del Coche Evita Obstáculos

https://youtu.be/Yn7Xz3Nk6PA



Código de la simulación del Coche Evita Obstáculos en Tinkercad:

```
int LW = 2;

int RW = 3;

int LB = 4;

int RB = 5;0

int MotorDer = 6;

int MotorIzq = 7;

int trigPin = 8;

int echoPin = 9;

int buzzer = 10;

 void setup()

{
```

```
 Serial.begin(9600); //inicialización de comunicación serial a 9600 bps
 pinMode(2, OUTPUT); 0//se declaran todos los led's como salida
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
 pinMode(trigPin, OUTPUT); //trig pin como salida (manda un pulso)
 pinMode(echoPin, INPUT); //echo pin como entrada (recibe el pulso)
 pinMode(6, OUTPUT);
 pinMode(7, OUTPUT);
 pinMode(10, OUTPUT);
}
void loop()
{
 long duration, distance; //se activa el sensor ultrasonidos hc-sr04
 digitalWrite(trigPin,HIGH); //manda un pulso de ultrasonidos (pin 8)
 delayMicroseconds(1000);
 digitalWrite(trigPin, LOW);
 duration=pulseIn(echoPin, HIGH); //recibe el pulso (pin 9)
 distance =(duration/2)/29.1; //distancia medida en centímetros
 Serial.print(distance); //envío de valor de distancia por monitor serial
 Serial.println("CM");
 delay(10);
 if(distance>20) //obstáculo lejos (a más de 20 cm)
 {
  digitalWrite(2,HIGH); //led's blancos (delanteros) encendidos al no detectar obstáculo
```

```
  digitalWrite(3,HIGH);
  digitalWrite(4,LOW); //led's rojos (traseros) apagados al no detectar ostáculo
  digitalWrite(5,LOW);
  digitalWrite(6,HIGH); //rueda derecha avanza
  digitalWrite(7,HIGH); //rueda izquierda avanza
}
else if(distance<20)
{
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite (buzzer, HIGH);
  delay(500);
  digitalWrite (buzzer, LOW);
  delay(500);
  digitalWrite (buzzer, HIGH);
  delay(500);
  digitalWrite (buzzer, LOW);
  delay(500);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  delay(6000);
 }
}
```

Programa final en Arduino IDE:

Ahora, como este no era nuestro objetivo (tan solo empleamos Tinkercad por tradición), procedimos a remodelar nuestro código para adaptarlo a Arduino IDE. La gran diferencia fue la utilización de la placa controladora en vez de los transistores. Esto nos permitía que los motores DC junto a las ruedas pudiesen moverse en ambos sentidos, lo cual nos abría un abanico de posibilidades.

Este código de Arduino es para un coche robot que usa un sensor ultrasónico para medir la distancia a los obstáculos y reaccionar en consecuencia. Inicializa pines para LED's, motores, un sensor ultrasónico y un buzzer. En el ciclo principal, el sensor ultrasónico mide la distancia y la envía al monitor serial. Si la distancia es mayor a 10 cm, los LED's frontales blancos se encienden, los traseros rojos se apagan, el LED intermitente se apaga y ambos motores avanzan. Si la distancia es menor a 10 cm, los LED's frontales se apagan, los traseros se encienden, el LED intermitente se apaga, el coche retrocede, suena el buzzer de manera intermitente, y luego realiza un giro apagando los LED's rojos, encendiendo el LED intermitente y girando (motor derecho retrocede y motor izquierdo avanza) para evitar el obstáculo.

Aquí tenéis parte del programa del robot evita-obstáculos, si queréis construir y programar este proyecto tendréis que pensar un poquito, igual que han hecho nuestros alumnos durante este curso:

```
int MotorDer2 = 7; //IN2 (Motor derecha)
int MotorIzq1 = 12; //IN3 (Motor izquierda)
int MotorIzq2 = 13; //IN4 (Motor izquierda)
int trigPin = 8; //trigger (sensor ultrasonidos)
int echoPin = 9; //echo (sensor ultrasonidos)
int buzzer = 10;

void setup()
{
  Serial.begin(9600); //inicialización de comunicación serial a 9600 bps
  pinMode(2, OUTPUT); //se declaran todos los led's como salida
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(trigPin, OUTPUT); //trig pin como salida (manda un pulso)
  pinMode(echoPin, INPUT); //echo pin como entrada (recibe el pulso)
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}
void loop()
{
  long duration, distance; //se activa el sensor ultrasonidos hc-sr04
  digitalWrite(trigPin,HIGH); //manda un pulso de ultrasonidos (pin 8)
  delayMicroseconds(1000);
  digitalWrite(trigPin, LOW);
  duration=pulseIn(echoPin, HIGH); //recibe el pulso (pin 9)
```

Materiales:

Material	Cantidad	Precio unitario	Precio total
Tablón de madera contrachapada	3	3€	9€
Listones	2	2.41€	4.82
Arduino Uno R3	1	10.50€	10.50€
Motor de aficionado	2	4.30€	8.60€
LED blanco	2	1.20€	2.40€
Resistencia 220 Ω	5	0.11€	0.55€
LED rojo	2	1.20€	2.40€
LED naranja	1	1.20€	1.20€

Material	Cantidad	Precio unitario	Precio total
Módulo controlador de motores L298N PUENTE-H	1	8.06€	8.06€
Pila 9V	1	4.30€	4.30€
Zumbador	1	3.45€	3.45€
Sensor de ultrasonidos HC-SR04	1	2.58€	2.58€
TOTAL			57.86€

Construcción:

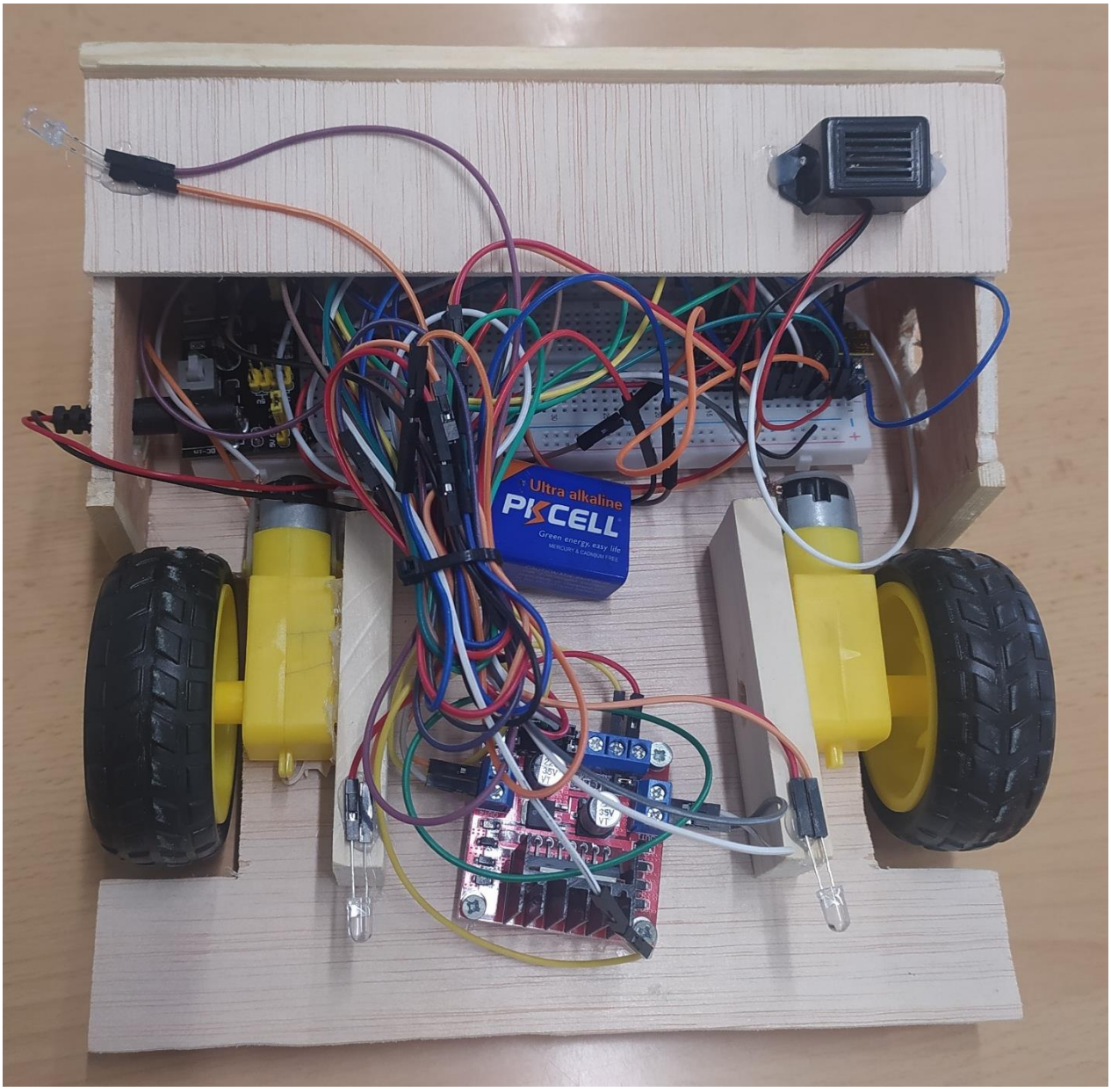
Primero de todo, cogimos las medidas del diseño en Sketch Up y las pasamos a nuestros tablones de madera contrachapada y listones. Hicimos las piezas con las sierras del taller y con limas y papel de lija quitamos las imperfecciones para finalmente unirlas mediante clavos y cola, y un poco de silicona caliente.

Una vez tenemos la estructura del coche, procedemos a colocar los motores, a los cuales les pusimos las ruedas. Después, en la parte inferior delantera del coche, pusimos una rueda de bola pequeña para que esté recto y pueda realizar todos los movimientos deseados.

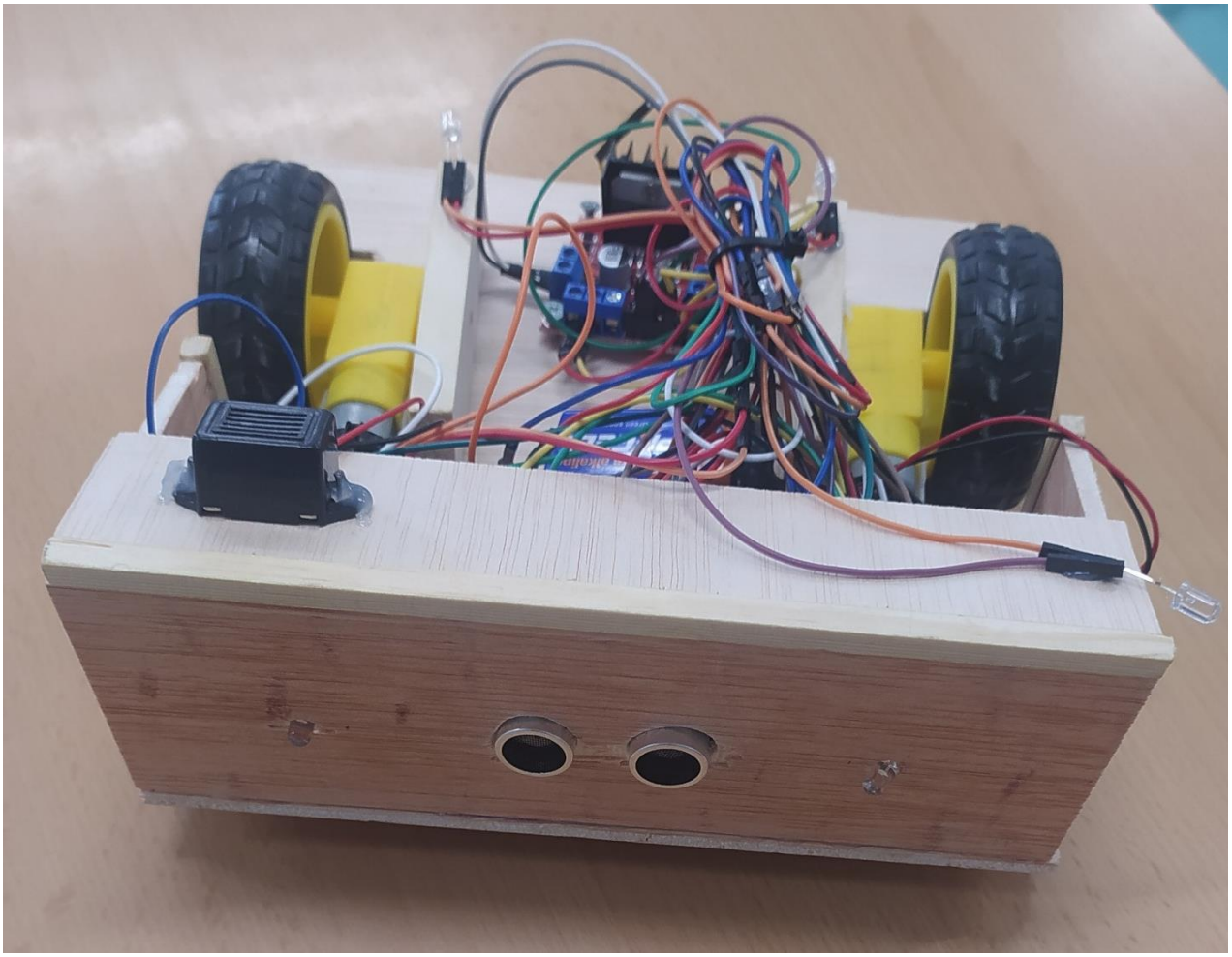
Por último, colocamos las dos placas, las luces LED y el zumbador que mediante cables los conectamos.

5. Resultados y análisis

Hemos tenido unos muy buenos resultados, el coche, tras un duro trabajo y muchas horas empleadas hemos conseguido que funcione. El funcionamiento es simple, el coche al no detectar ningún obstáculo a menos de 10 centímetros enfrente suya avanza, y al mismo tiempo tiene ambos LED blancos delanteros encendidos. Una vez detectado un objeto dentro de la distancia previamente comentada, ambas ruedas retroceden, los LED blancos delanteros se apagan, y se encienden las rojas traseras. Después de ir marcha atrás, se enciende el LED lateral naranja, el intermitente, la rueda izquierda gira a menor velocidad, mientras la derecha gira al sentido contrario para girar.



15. Resultado final del Coche Evita Obstáculos



16. Resultado final del Coche Evita Obstáculos

6. Conclusiones

En cuanto a la conclusión podemos afirmar el buen funcionamiento del coche, hemos podido solucionar todos los pequeños problemas que nos hacían retroceder un paso cuando queríamos avanzar dos, hemos conseguido solucionar los problemas que tuvimos al principio al programar mal los movimientos, al principio el coche funcionaba en sentido contrario, también la luz de los LED se encendían en el momento equivocado, de la misma manera que se encendían en el momento equivocado también se encendían los delanteros cuando se tenían que encender los traseros y al contrario. Por otro lado, teníamos un error en la programación para detectar la distancia establecida, fue un proceso bastante tedioso que nos llevó a invertir un par de horas para encontrar la solución.

Estamos muy contentos y orgullosos por nuestra capacidad de superar las adversidades a la que nos enfrentábamos cada día consiguiendo un resultado muy satisfactorio en el que pensamos que tiene un gran potencial para ganar la feria y llevarnos el primer premio.

7. Agradecimientos

Queremos darle las gracias al alumnado miembro de la asignatura de Tecnología e Ingeniería I, al igual que al profesorado de esta misma asignatura que nos ha ayudado a crear este maravilloso proyecto, además de los profesores que nos han permitido continuar en algunas de sus clases porque sabían lo importante que era este trabajo para nosotros, y por último también al director del centro I.E.S. La Patacona por brindarnos una materia única como lo es esta. Gracias a estos miembros del centro nombrados por ayudarnos todos para realizar el proyecto.

8. Bibliografía

- Material propio del departamento de Tecnología e Ingeniería.
- https://www.freepik.es/
- https://www.tinkercad.com/
- https://app.sketchup.com/