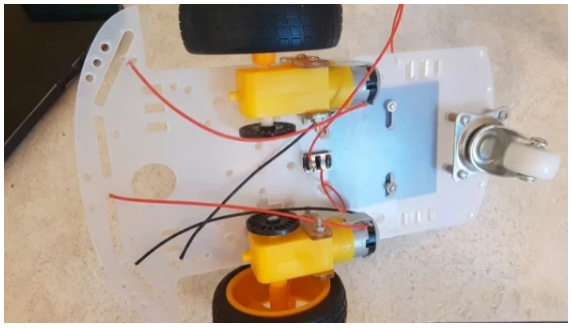
Ashar,Nahim
2smxd

Manual de instalación del RoboCar BT

RoboCar BT es un coche robot que se controla desde una aplicación móvil mediante Bluetooth. En este proyecto se ha realizado el montaje completo del coche, la conexión de sus componentes y la programación necesaria para que responda a las órdenes enviadas desde el móvil. Para ello se han utilizado un kit del coche con el driver L298N y los motores, una placa ZUM BT con Bluetooth integrado, cuatro pilas AA, cables jumpers, un interruptor y la aplicación creada con App Inventor.

Antes de empezar, hay que preparar todas las piezas necesarias: el chasis, los motores, las ruedas, la placa ZUM BT, el driver L298N, el portapilas, el interruptor y los cables. Tener todo preparado desde el principio ayuda a trabajar de forma más ordenada y a evitar errores durante el montaje. Lo primero que se hace es colocar el chasis como base del proyecto y fijar los motores en los laterales con sus soportes y tornillos.

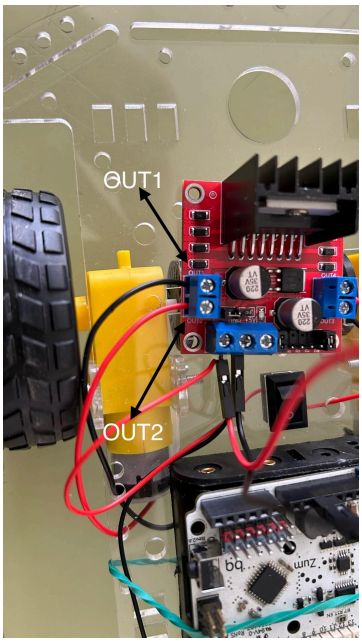
A continuación, se fijan los motores a los laterales del chasis y se colocan los cables de conexión. En cada motor se sueldan dos cables, uno negro y uno rojo. Primero se suelda el cable negro, que queda en la parte de arriba, y después el cable rojo, que queda en la parte de abajo. Así se dejan listos para conectarlos al driver L298N. Una vez puestos, se encajan las ruedas en los ejes de los motores y se comprueba que giren bien.

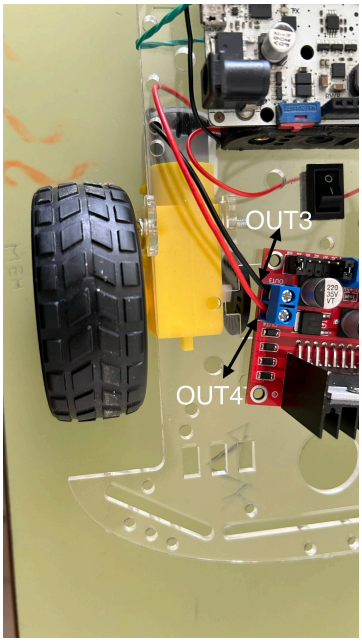


Más tarde, se colocan la placa ZUM BT y el driver de motores L298N en la parte superior del coche, dejando espacio suficiente para hacer las conexiones con comodidad. Los motores se conectan al módulo L298N: un motor va a las salidas OUT1 al cable negro y OUT2 al cable rojo y el otro a OUT3 al cable negro y OUT4 al cable rojo, para poder controlar cada lado del coche por separado. Si al probarlo un motor gira al revés, solo hay que cambiar el rojo y el negro de ese motor.

OUT1, OUT2: OUT3, OUT4:

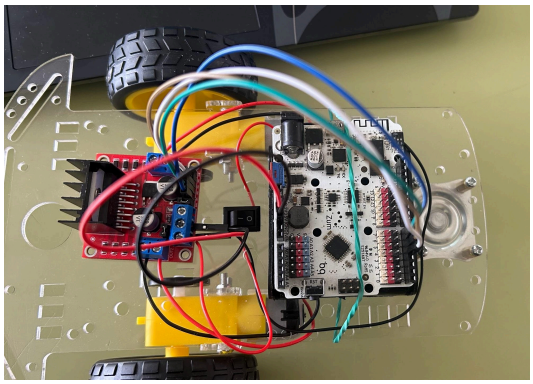
Ashar,Nahim
2smxd



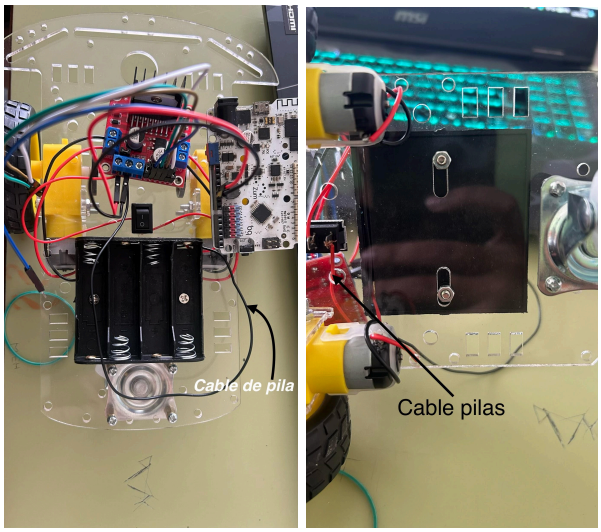


Ashar,Nahim
2smxd

Después se conecta la placa ZUM BT al driver L298N mediante los pines de control. En el programa se usan los pines 7 (Azul), 6 (Verde), 5 (Marrón) y 4 (Blanco) de la placa para enviar las órdenes a las entradas IN1(Azul), IN2 (Verde), IN3 (Marrón) y el IN4 (Blanco) del driver. Esta conexión es la que permite controlar el sentido de los motores y hacer que el coche avance, retroceda, gire o se detenga.

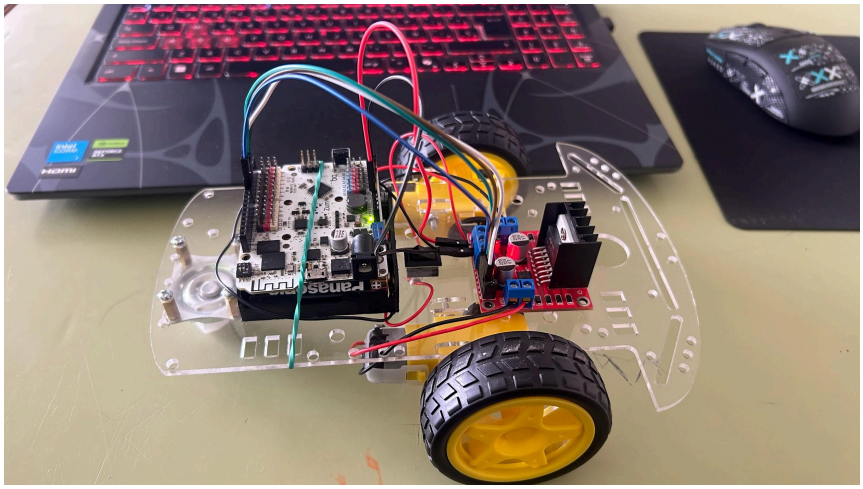


Seguidamente, se coloca el portapilas con las cuatro pilas AA y se conecta la alimentación al sistema. El cable rojo del portapilas va al interruptor y desde el interruptor sale otro cable rojo hacia la alimentación del circuito. El cable negro del portapilas va directamente a GND. El interruptor se coloca en medio del cable rojo para poder encender y apagar el coche sin tener que desconectar las pilas.



Después de eso, se unen las masas de los componentes para que todo el circuito funcione correctamente. Es importante que la placa ZUM BT, el driver L298N y la alimentación compartan la misma masa, porque si no, el coche puede fallar al recibir las órdenes. Antes de seguir, hay que revisar que todos los cables estén bien puestos y

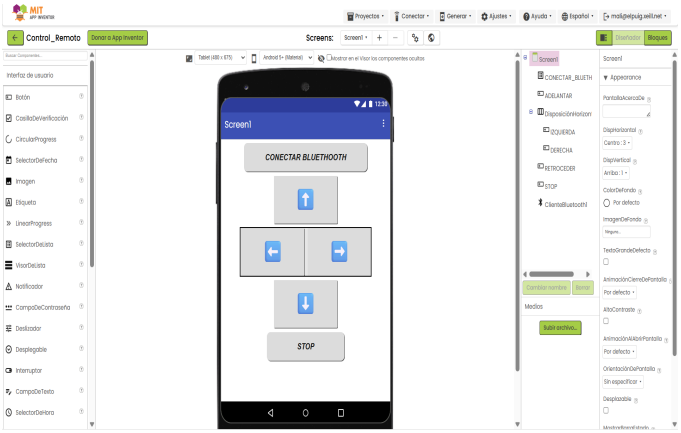
que no haya ninguna conexión suelta, si esta todo bien se encenderà y parpadearà si no esta conectado al bluethooth.

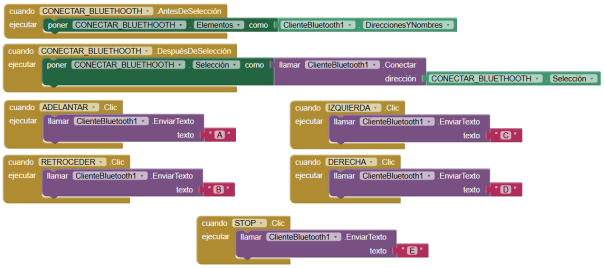


En cuanto a la programación, se hizo para que la placa recibiera comandos por Bluetooth y activara los pines del motor según la orden enviada desde la aplicación. El programa utiliza las letras A, B, C, D y E para indicar las acciones del coche: avanzar, retroceder, girar a la izquierda, girar a la derecha y detenerse. De esta forma, cada botón de la aplicación envía una letra y la placa la interpreta directamente.

La aplicación se creó con MIT App Inventor desde cero. Primero se diseñó la pantalla con un botón o lista para conectar el Bluetooth y varios botones para controlar el coche. Después se programaron los bloques para que cada botón enviara una orden distinta a la placa. Los botones principales son Adelante, Atrás, Izquierda, Derecha y Stop. Cuando el usuario pulsa uno de los botones, la app envía el comando correspondiente y el coche realiza la acción programada.

Ashar,Nahim
2smxd





Al principio se pensó en usar un Arduino Uno con un módulo HC-05 externo. Sin embargo, como surgieron varias dificultades con esa opción, se decidió cambiar a una placa ZUM BT con Bluetooth integrado. Este cambio simplificó el montaje y permitió terminar el proyecto de una forma más estable. Por eso, aunque la idea inicial fue otra, la solución final resultó más práctica para el funcionamiento del coche.

Una vez terminado el montaje, se realizaron pruebas para comprobar que el coche respondía bien a las órdenes de la aplicación. Al verificar que todo funcionaba correctamente, el proyecto quedó listo para usarse. El resultado final es un coche robot controlado desde el móvil mediante Bluetooth, con una estructura sencilla y un funcionamiento práctico, dejarè aqui una demo de como funcionara el coche.

Ashar,Nahim
2smxd

Codigo Arduino:

```
#include <BitbloqSoftwareSerial.h>

bqSoftwareSerial board_bluetooth(0, 1, 19200);

char comando;

int IN1 = 7;
int IN2 = 6;
int IN3 = 5;
int IN4 = 4;

void setup() {
  board_bluetooth.begin(19200);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop() {

  if (board_bluetooth.available()) {

    comando = board_bluetooth.read();

    switch (comando) {

      case 'A': // Adelante
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        digitalWrite(IN3, HIGH);
        digitalWrite(IN4, LOW);
        break;

      case 'B': // Atrás
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH);
```

Ashar,Nahim
2smxd

```
            digitalWrite(IN3, LOW);
            digitalWrite(IN4, HIGH);
            break;

        case 'C': // Izquierda
            digitalWrite(IN1, HIGH);
            digitalWrite(IN2, LOW);
            digitalWrite(IN3, LOW);
            digitalWrite(IN4, HIGH);
            break;

        case 'D': // Derecha
            digitalWrite(IN1, LOW);
            digitalWrite(IN2, HIGH);
            digitalWrite(IN3, HIGH);
            digitalWrite(IN4, LOW);
            break;


        case 'E': // Stop
            digitalWrite(IN1, LOW);
            digitalWrite(IN2, LOW);
            digitalWrite(IN3, LOW);
            digitalWrite(IN4, LOW);
            break;
      }
   }
}
```