Instruction Manual

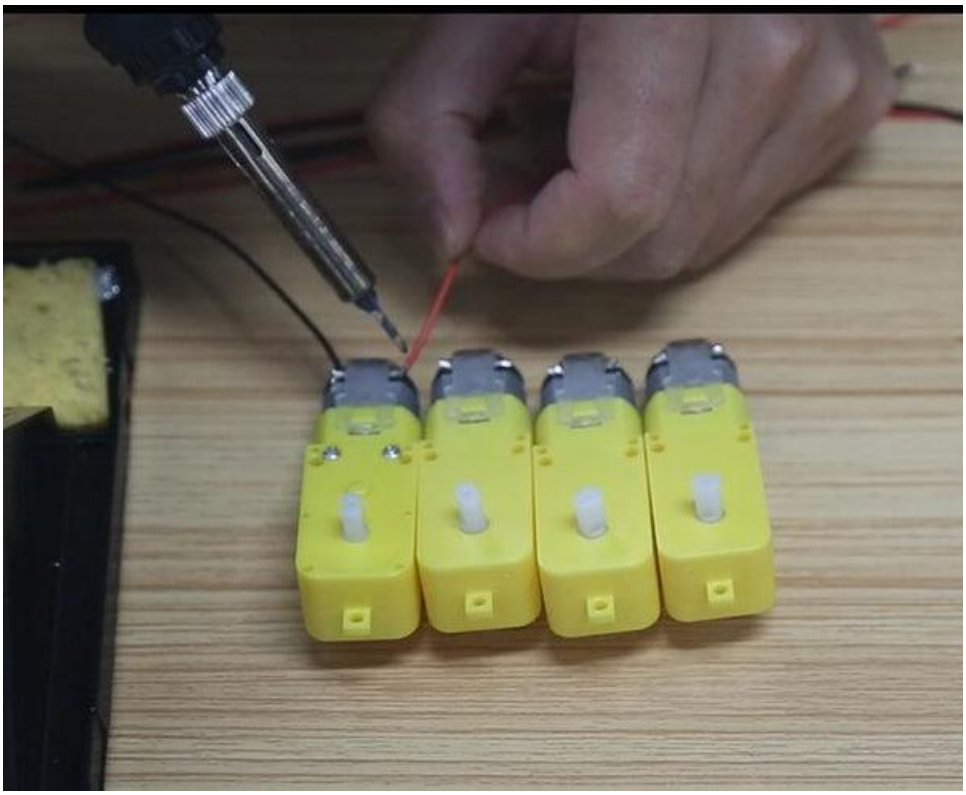
Multi-Functional 4WD Robot Car Chassis Kits UNO R3 For Robot Car Assembly
SKU: 30678

==========

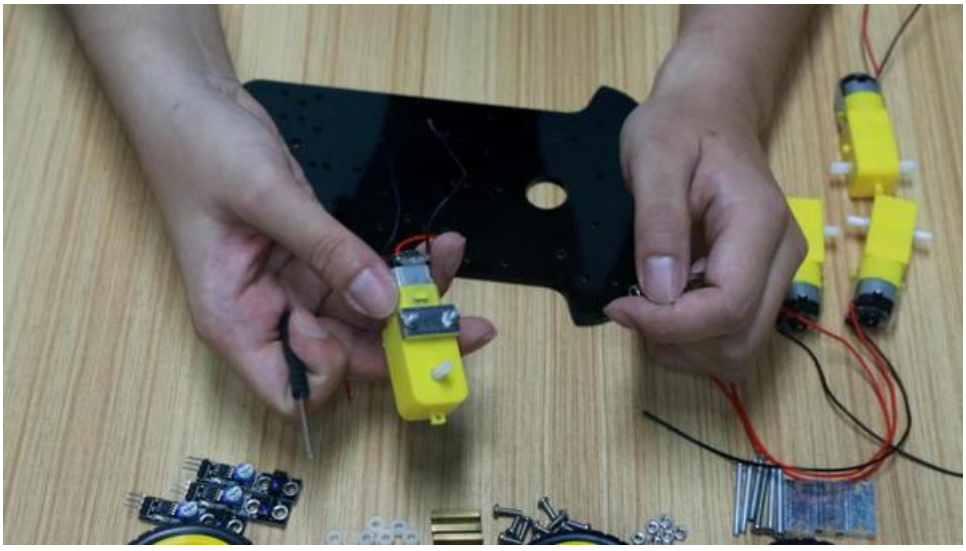
This is all the parts of a car. Below I will be divided into four parts to assemble the car.

Part I Assembly Of Wheels, Motors And Tracking Sensors

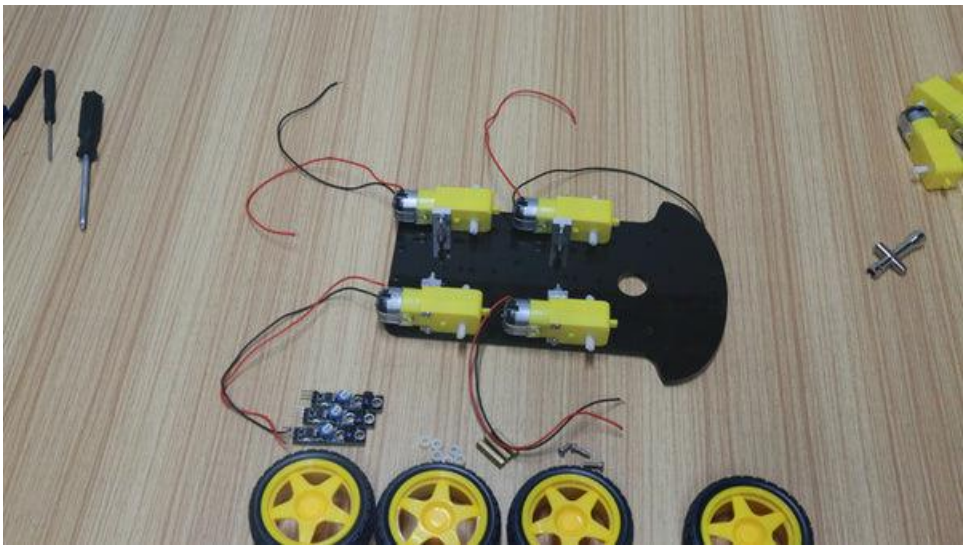
First, we solder the red and black power wires to the motor, and the soldering position remains uniform.



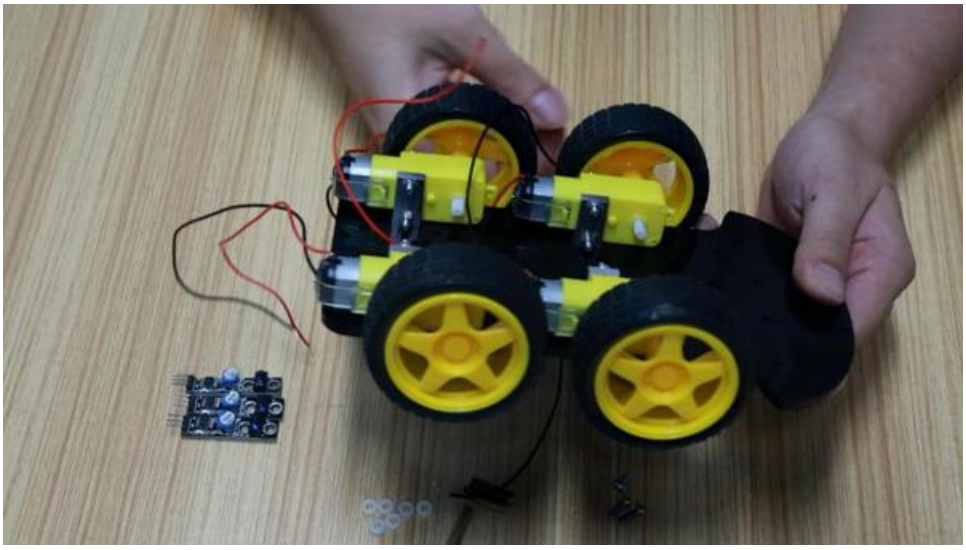
After soldering, connect the fixing post of the motor to the motor with long screws and nuts.



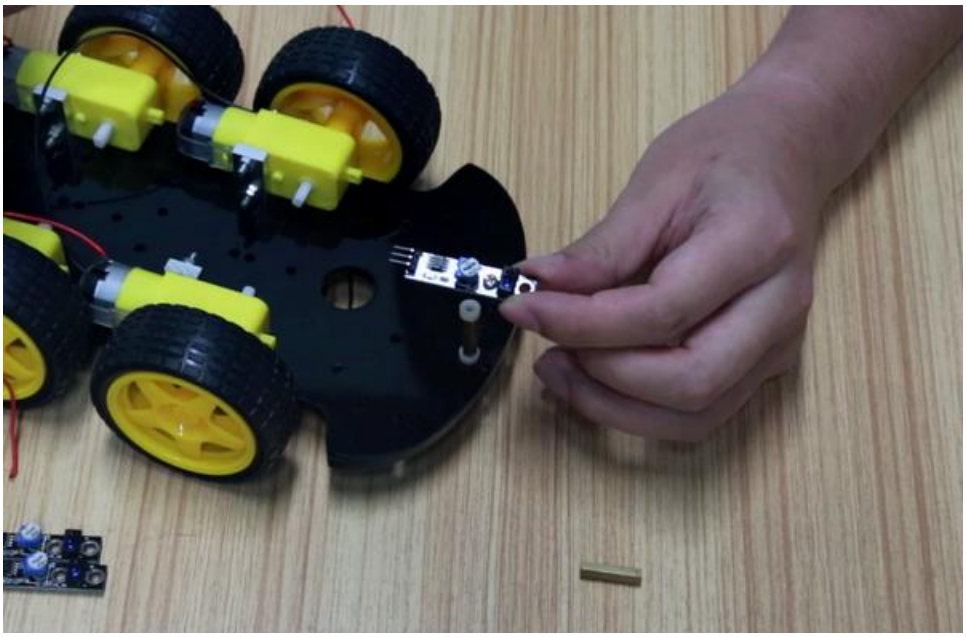
Note that the motor should be installed with the same direction as shown in the figure below, fixed with screws.



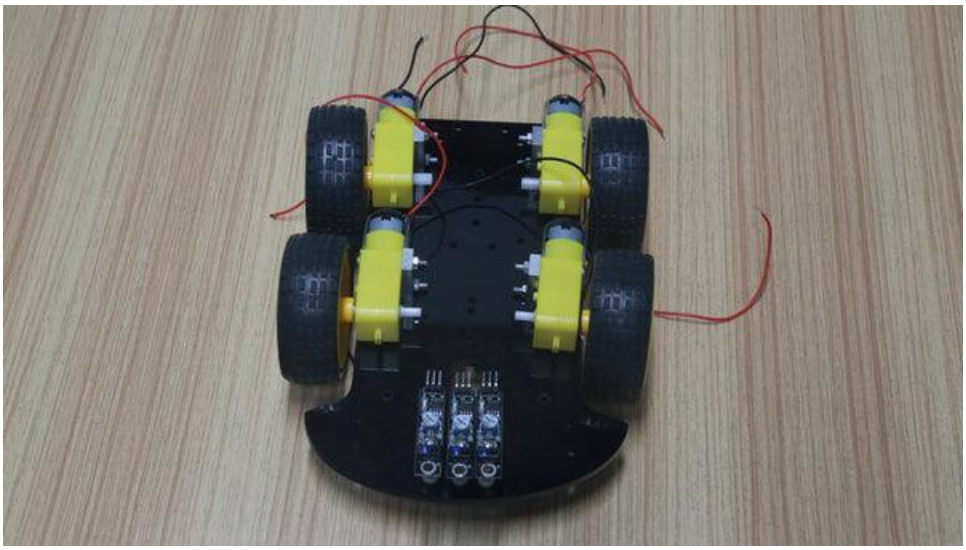
Mount the wheel to the motor.



Next, install 3 tracing modules in front of the car.
Each sensor requires a short copper post and two plastic spacers.
This is to shorten the distance between the sensor and the ground, and the data obtained is more accurate.
The two ends of the copper column are fixed with screws.

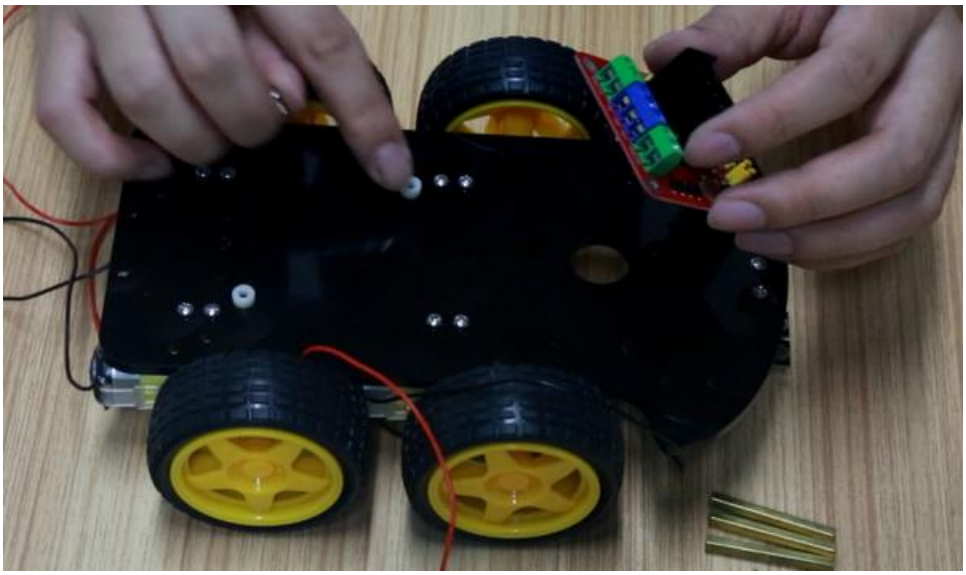


Complete the first part.

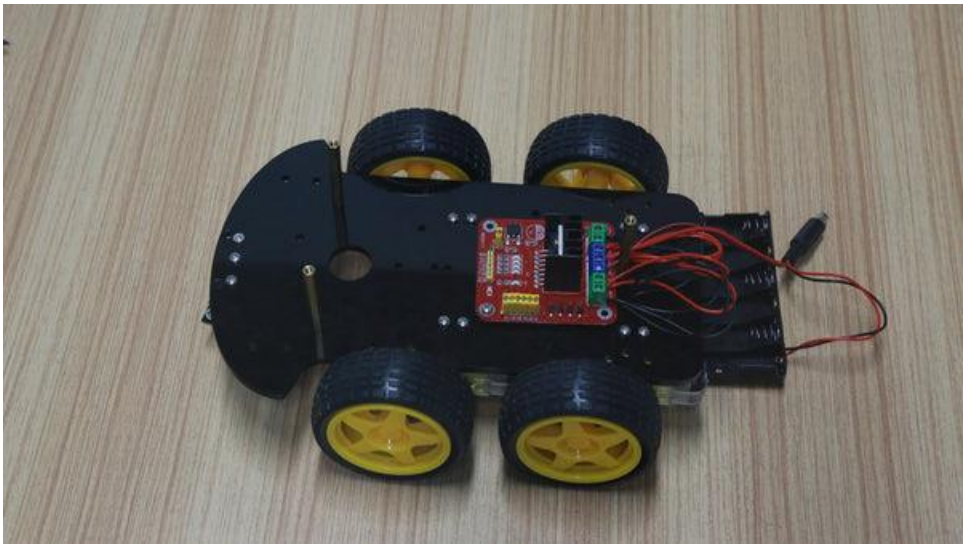


Part II Installing The Motor Drive Module

The motor drive module should be mounted on the opposite side of the motor.
Find two holes, use two plastic spacers under the board, and then fix them with screws and nuts.

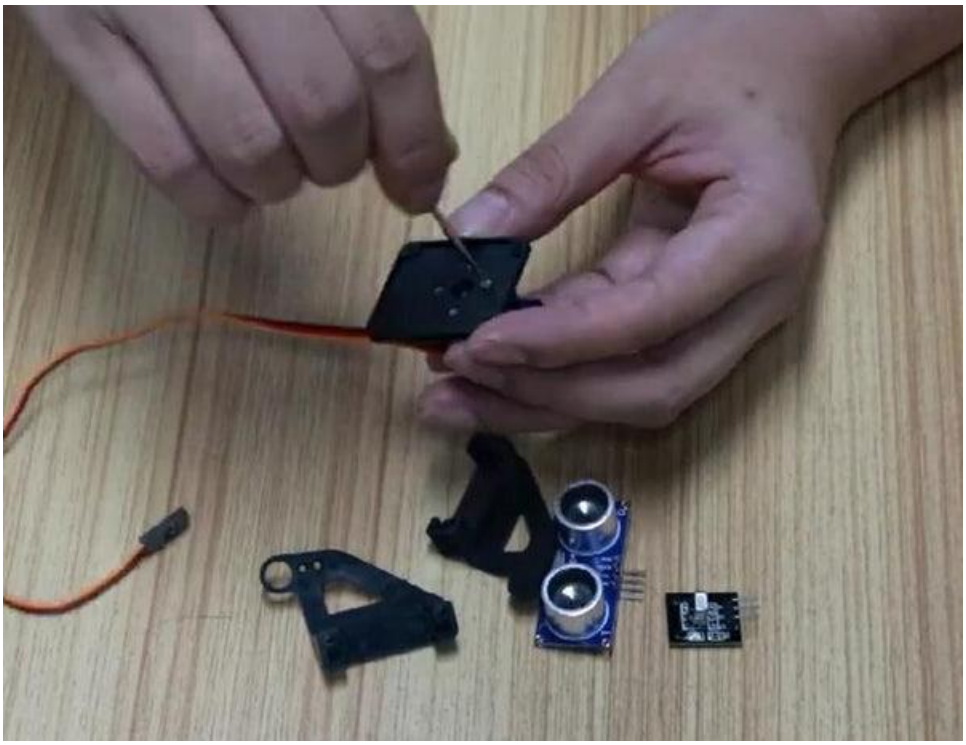


Complete the second part.

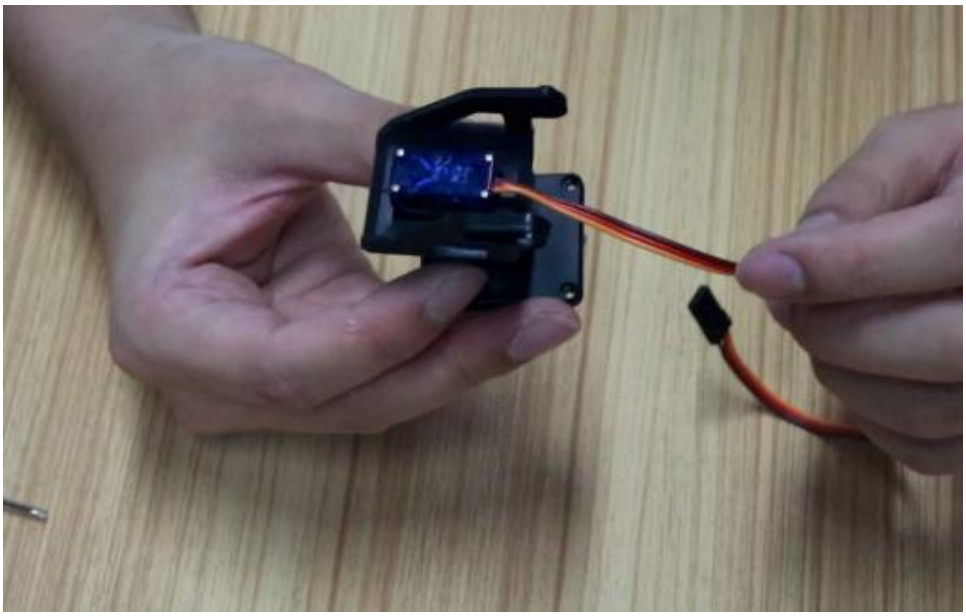


Part III Installation Of Steering Gear And UNO Board

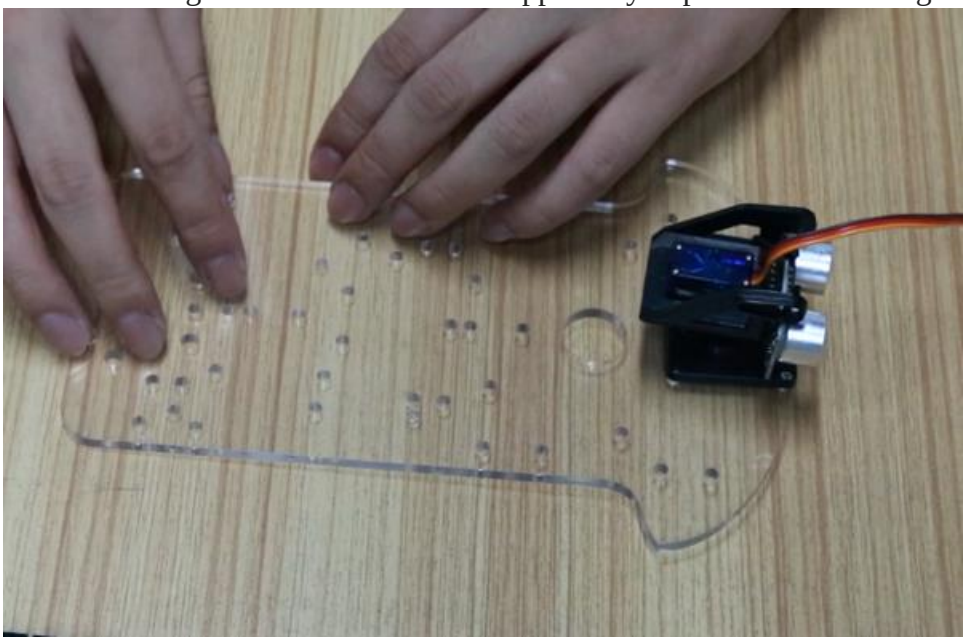
Install the rudder turntable on the steering gear, then fix it to the black bracket and tighten with 5 small screws.
Note: Turn the servos by hand so that the 90° position is facing forward and then fixed.



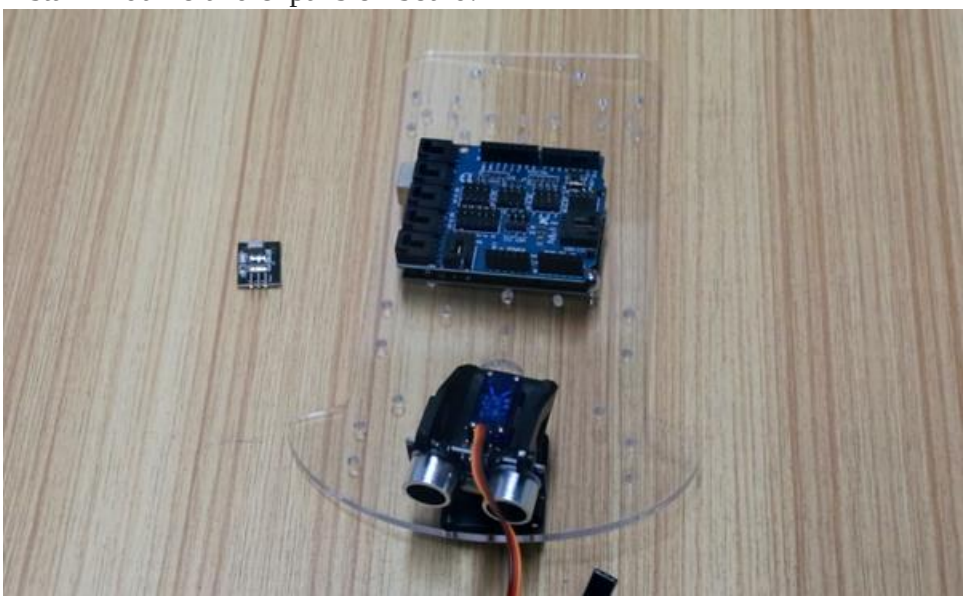
Install the brackets on the left and right sides of the servo, align the raised positions of the servo, and use two screws to fix the sides of the bracket.



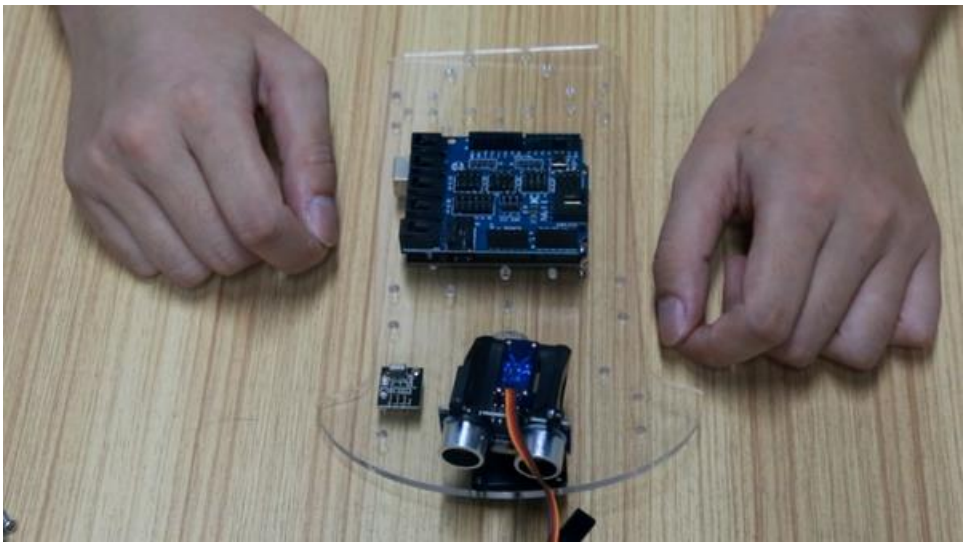
Attach the ultrasonic sensor to the servo holder using two zip ties.
Fix the steering base to the front of the upper acrylic plate with thin long screws and nuts.



Install Arduino and expansion board.

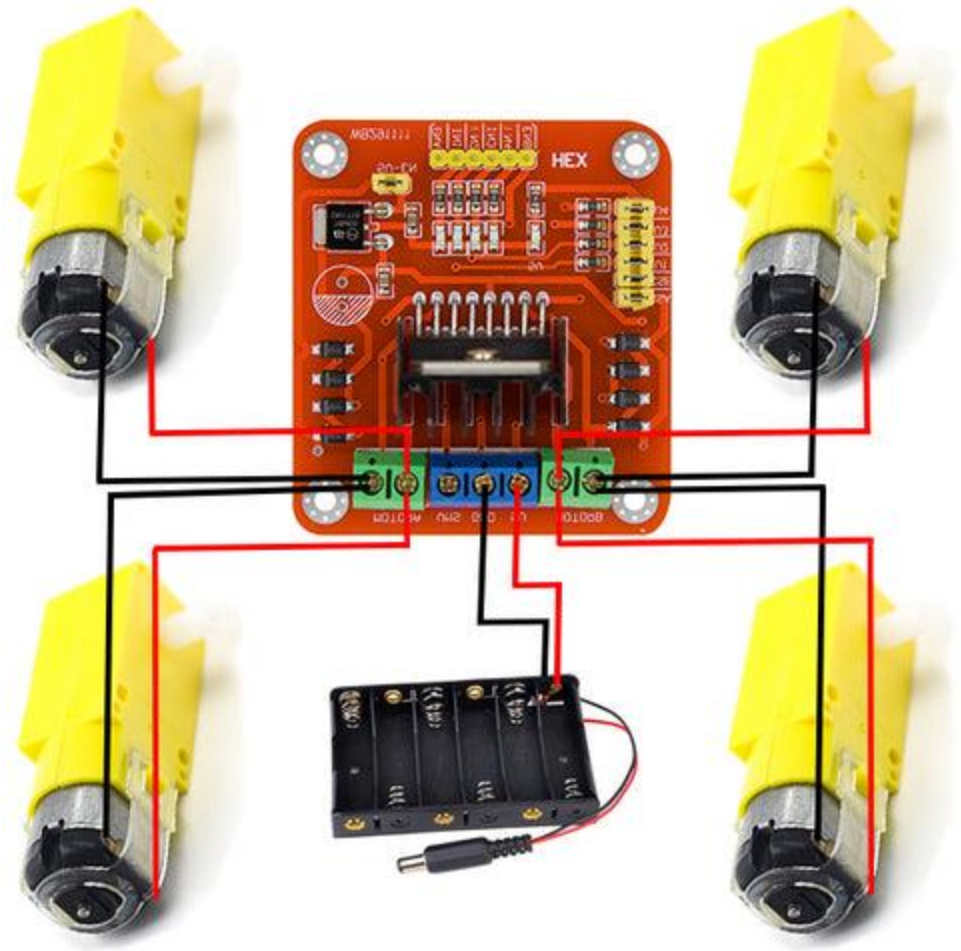


Install the receiver module of the infrared remote control next to the servo.



Part IV Connection

Motor drive module wiring diagram



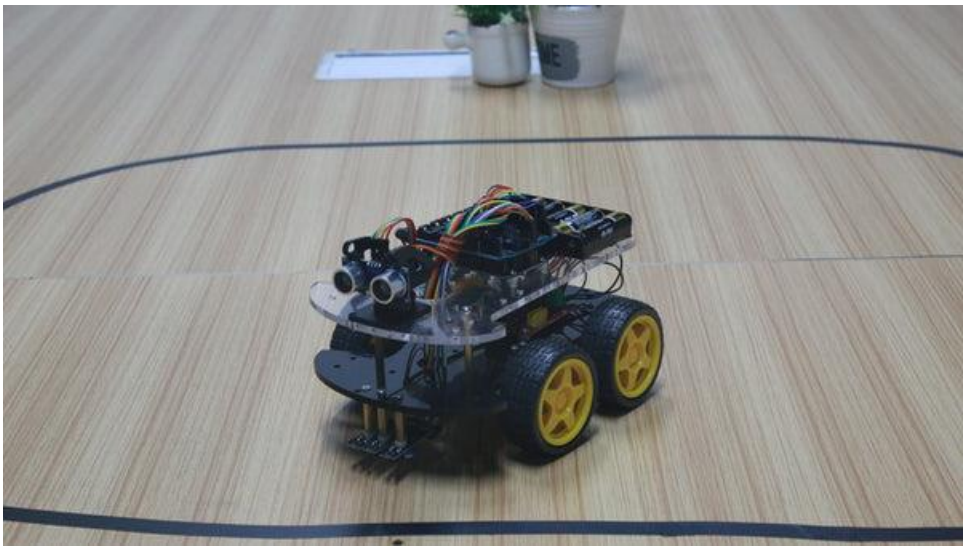
Arduino pin connection table

Ultrasonic sensor	ECHO	A0
	TRIG	A1
	GND	GND
	VCC	5V

Servo	Orange line	D12
	Red line	5V
	Brown line	GND
Infrared receiver module	G	GND
	R	5V
	Y	D13
Motor driven module	ENA	D6
	IN1	D7
	IN2	D8
	IN3	D9
	IN4	D10
	ENB	D5

Tracing module	G	GND
	V+	5V
	S-First	D4
	S-Second	D1
	S-Third	D2

Finished!



==========

Code:

```
#include <IRremote.h>
/* https://github.com/z3t0/Arduino-IRremote */
#include <Servo.h>
/* Define the pin */
#define Echo A0                  //ECHO pin of obstacle avoidance module
#define Trig A1                  //Trig pin of obstacle avoidance module
#define ENB 5                    //The pin of motor drive
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 10
#define ENA 6
#define RECV_PIN 13              //Infrared receiver pins
#define LineTeacking_Pin_Right  2        //The pins of the trace module
#define LineTeacking_Pin_Middle 1
#define LineTeacking_Pin_Left   4
#define SERVO 12                 //Servo motor pins

#define FORWARD 2
#define BACK 8
#define LEFT 4
#define RIGHT 6
#define STOP 5
#define MOVE_MODE 10
#define OBSTACLES_MODE 11
#define LINETRACKING_MODE 12
```

```
/* Infrared remote control coding  ,Each remote control is different, you need to test the code first */
#define KEY_2 16718055
#define KEY_4 16716015
#define KEY_6 16734885
#define KEY_8 16730805
#define KEY_5 16726215
#define KEY_CH1 16753245
#define KEY_CH2 16736925
#define KEY_CH3 16769565
/* Read the pins of the trace module */
#define LineTeacking_Read_Right   digitalRead(LineTeacking_Pin_Right)
#define LineTeacking_Read_Middle  digitalRead(LineTeacking_Pin_Middle)
#define LineTeacking_Read_Left    digitalRead(LineTeacking_Pin_Left)
int carSpeed = 250;
IRrecv irrecv(RECV_PIN);
decode_results results;
Servo myservo;
int rightDistance = 0, leftDistance = 0, middleDistance = 0;
int Mode = MOVE_MODE;
int Direction = 5;
unsigned long recvTime;
unsigned long lineTime;
void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(Echo, INPUT);
  pinMode(Trig, OUTPUT);
  pinMode(IN1, OUTPUT);
```

```
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  digitalWrite(ENA, HIGH);
  digitalWrite(ENB, HIGH);
  pinMode(LineTeacking_Pin_Right, INPUT);
  pinMode(LineTeacking_Pin_Middle, INPUT);
  pinMode(LineTeacking_Pin_Left, INPUT);
  myservo.attach(SERVO);
  myservo.write(90);
}

void loop() {
  recvData();
  moveMode();
  obstaclesMode();
  linetrackingMode();
}
void moveMode() {
  if (Mode == MOVE_MODE) {
    carSpeed = 250;
    switch (Direction) {
      case FORWARD: forward(); break;
      case BACK: back(); break;
      case LEFT: left(); break;
      case RIGHT: right(); break;
      case STOP: stop(); break;
```

```
      default: break;
    }
    if (millis() - recvTime >= 500) {
      Direction = STOP;
      recvTime = millis();
    }
  }
}
/* Adjust "carSpeed" if it hits the wall frequently  */
void obstaclesMode() {
  if (Mode == OBSTACLES_MODE) {

    carSpeed = 200;
    myservo.write(90);
    delay(500);
    middleDistance = getDistance();
    if (middleDistance <= 40) {
      stop();
      delay(500);
      myservo.write(10);
      delay(1000);
      rightDistance = getDistance();

      delay(500);
      myservo.write(90);
      delay(1000);
      myservo.write(180);
      delay(1000);
      leftDistance = getDistance();
```

```
      delay(500);
      myservo.write(90);
      delay(1000);
      if (rightDistance > leftDistance) {
        right();
        delay(360);
      }
      else if (rightDistance < leftDistance) {
        left();
        delay(360);
      }
      else if ((rightDistance <= 40) || (leftDistance <= 40)) {
        back();
        delay(180);
      }
      else {
        forward();
      }
    }
    else {
      forward();
    }
  }
}
int getDistance() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
```

```
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  return (int)pulseIn(Echo, HIGH) / 58;
}
/*The trace module needs to test whether the low level trigger or the high level trigger.
  It can be adjusted according to the actual situation*/
void linetrackingMode() {
  if (Mode == LINETRACKING_MODE) {
    carSpeed = 150;
    if (LineTeacking_Pin_Middle) {
      forward();
      //Serial.println("forward");
      while (LineTeacking_Pin_Middle);
    }
    if (LineTeacking_Pin_Left) {
      right();
      while (LineTeacking_Pin_Left);
    }
    else if (LineTeacking_Pin_Right) {
      left();
      while (LineTeacking_Pin_Right);
    }
    else if (LineTeacking_Pin_Left && LineTeacking_Pin_Middle) {
      right();
      while (LineTeacking_Pin_Left);
    }
    else if (LineTeacking_Pin_Right && LineTeacking_Pin_Middle) {
      left();
      while (LineTeacking_Pin_Left);
```

```
    }
    else {
      forward();
    }
  }
}
void recvData() {
  if (irrecv.decode(&results)) {
    recvTime = millis();
    switch (results.value) {
      case KEY_2:   Direction = FORWARD;  break;
      case KEY_4:   Direction = LEFT;     break;
      case KEY_6:   Direction = RIGHT;     break;
      case KEY_8:   Direction = BACK;    break;
      case KEY_5:   Direction = STOP;     break;
      case KEY_CH1:  Mode = MOVE_MODE; stop(); Serial.println("MOVE_MODE");
        delay(1000); break;
      case KEY_CH2:  Mode = OBSTACLES_MODE; stop();
Serial.println("OBSTACLES_MODE");
        delay(1000); break;
      case KEY_CH3:  Mode = LINETRACKING_MODE; stop();
Serial.println("LINETRACKING_MODE");
        delay(1000); break;
      default: break;
    }
    irrecv.resume();
  }
}
/*  The best thing to do is to input the HIGH and LOW of each pin once,
    look at the direction of each wheel of execution, and then define this function  */
```

```
void forward() {

  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("Forward");  //send message to serial monitor
}

void back() {


  digitalWrite(IN1, LOW);    //set IN1 hight level
  digitalWrite(IN2, HIGH);     //set IN2 low level
  digitalWrite(IN3, HIGH);     //set IN3 low level
  digitalWrite(IN4, LOW);    //set IN4 hight level
  Serial.println("Back");
}

void stop() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  //Serial.println("stop");
}

void right() {
```

```
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("right");
}

void left() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);

  Serial.println("left");
}
```