TA0138 Robot Kit Manual

Line tracking
Obstacle avoidance
Remote control
Fall prevention

1. Introduction

Multi-functional smart car is a learning application development system of microcontroller, a four-wheel car system. It's based on ARDUINO microcontroller series atmega-328p. It has functions such as line tracking, obstacle avoidance, infrared remote control. This kit contains many interesting programs. It can also be expanded to have external circuit module to have other functions. This kit is designed to help you interestingly learn Arduino. You can learn Arduino MCU development ability while having fun.

2. Functions of this smart car

Ultrasonic obstacle avoidance: automatically avoid obstacle.
Line tracking: go along a black line.
Infrared remote control: go forward, backward and circle around.

3. Parameters

1. Motor parameters: Voltage: 6V, reduction ratio: 1 to 48
2.Use L298N driver module for motor control, true separation from microcontroller.
3. Three line tracking modules, detecting black and white line, high precision, can also be used in fall prevention.
4. Infrared remote communication module makes up the remote control system.

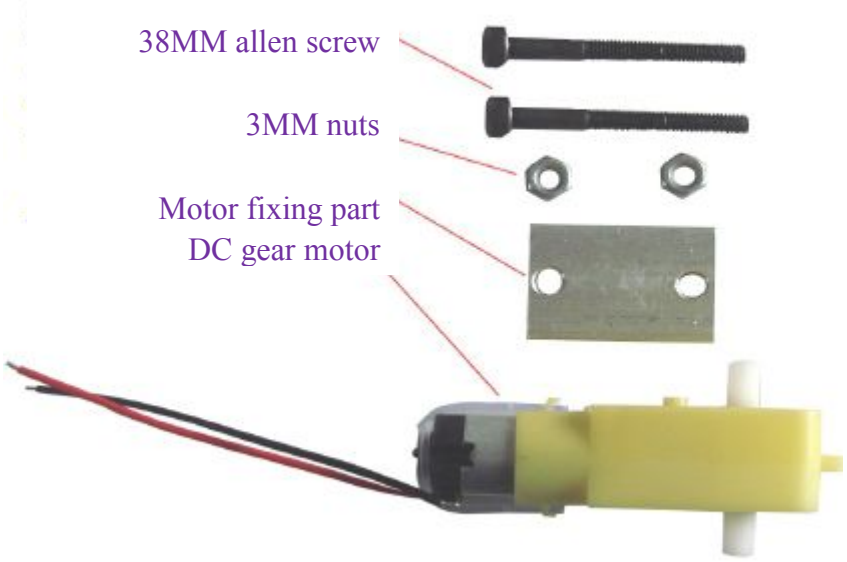
Car length: 23cm; width: 12cm. Can be used under external 7 ~ 12V voltage, also with various sensor modules to realize various functions.

4. Component list

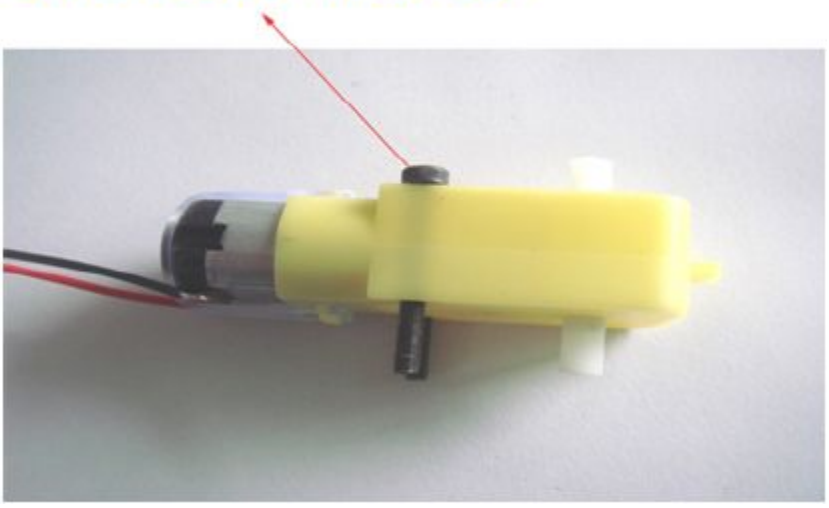
1. Gear motor *4
2. High-quality tires *4
3. Motor fixing part *4
3. 100 * 213 * 5MM acrylic board *1
4. 100 * 213 * 5MM acrylic board *1
5. L298N motor driver board *1
6. ARDUINO UNO328 controller board *1
7. ARDUINO sensor shield *1
8. Holder*1
9. Servo motor *1
10. Ultrasonic sensor Module *1
11. Three-channel line tracking module
12. IR receiver sensor
13. MCU remote control
14. Mini breadboard *1
15. 6-cell AA battery case*1
16. Dupont wire *30
17. 18650 charger *1
18. 1M USB cable
19. 35MM Copper bush *3
20. 20MM Copper bush *2
21. 6MM Copper bush *6
22. 3MM screws & nuts *several

5. Installation instructions

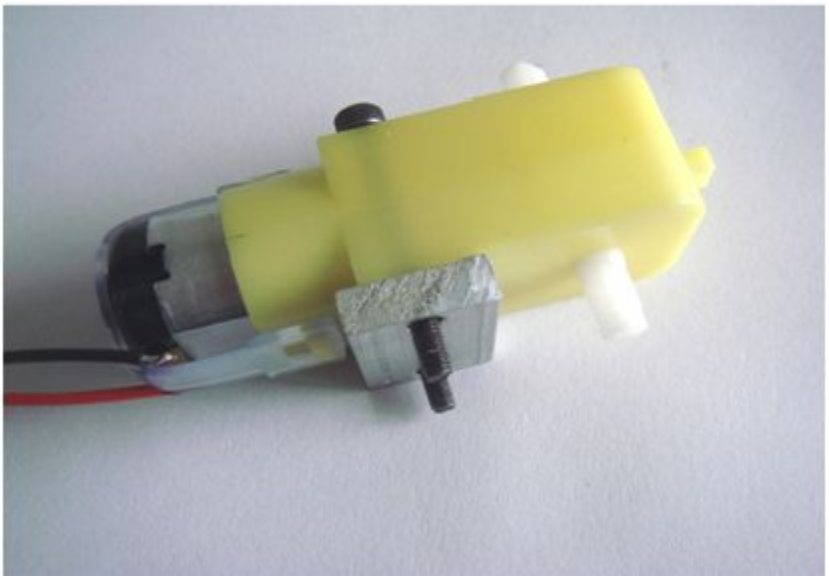
1. Motor installation



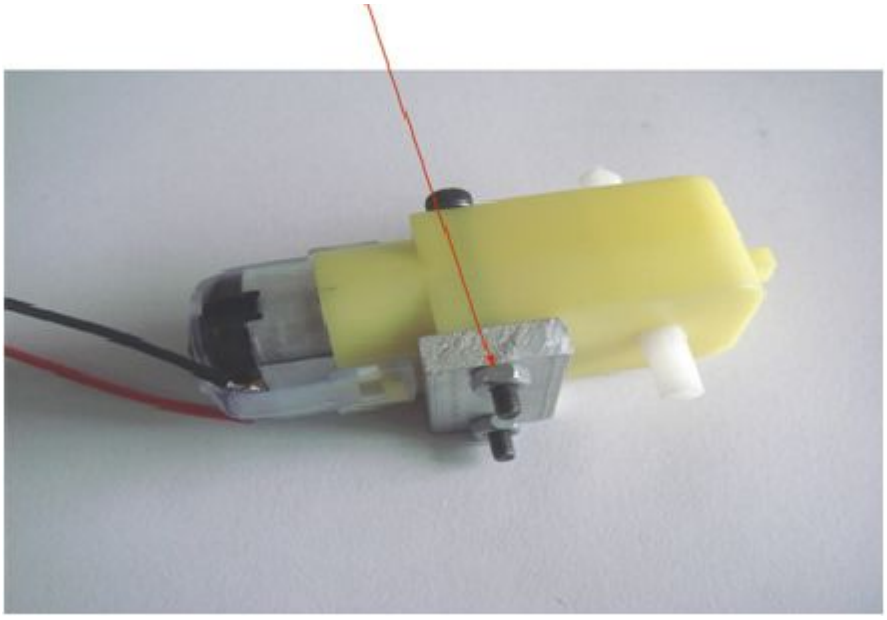
1) place the 38MM screw into the mounting hole of DC gear motor



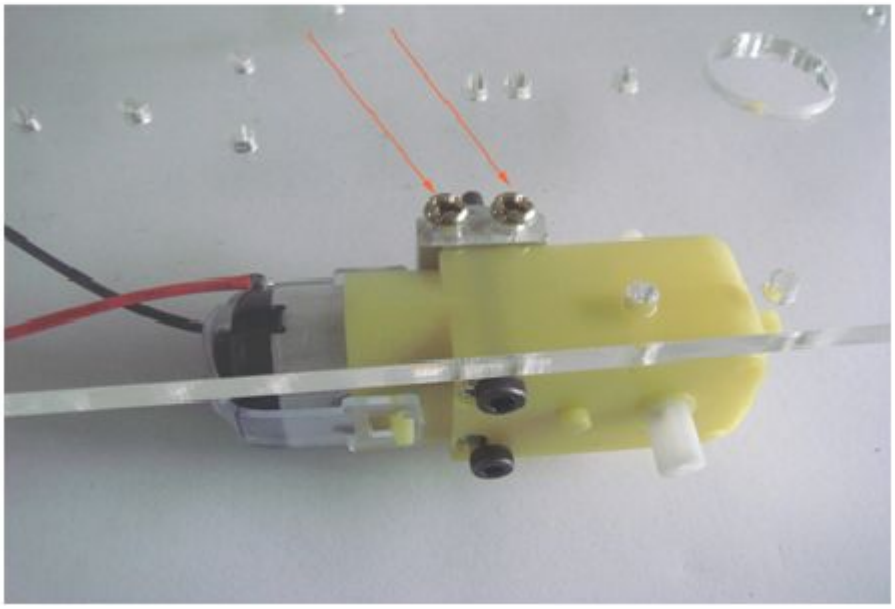
2) place the motor fixing part into the screws



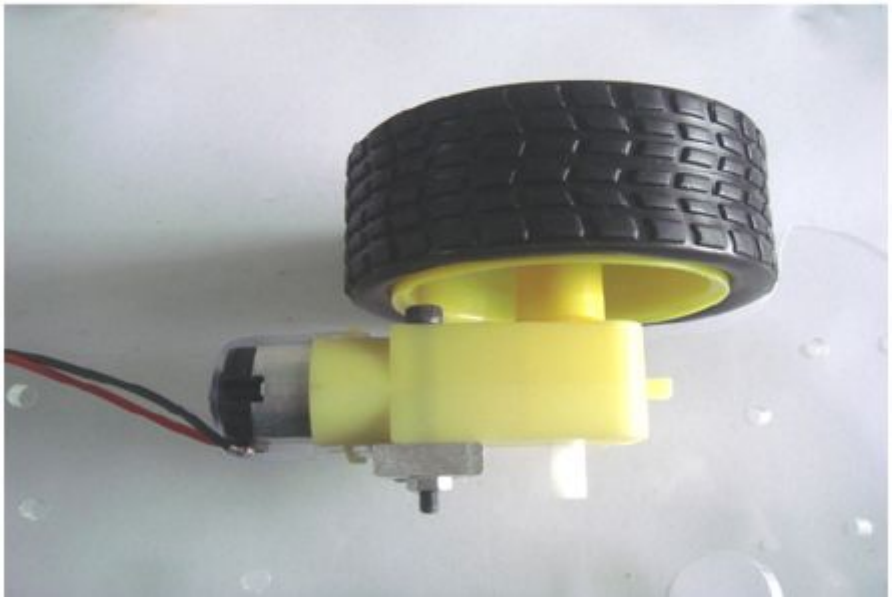
3) mount the nuts to fasten the screws



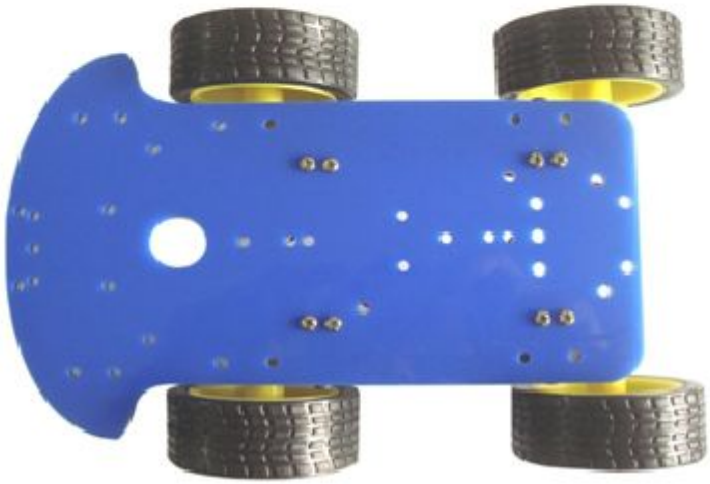
4) use two 3*10 screws to install the motor part into the motor mounting hole of the car chassis



5) install the wheel

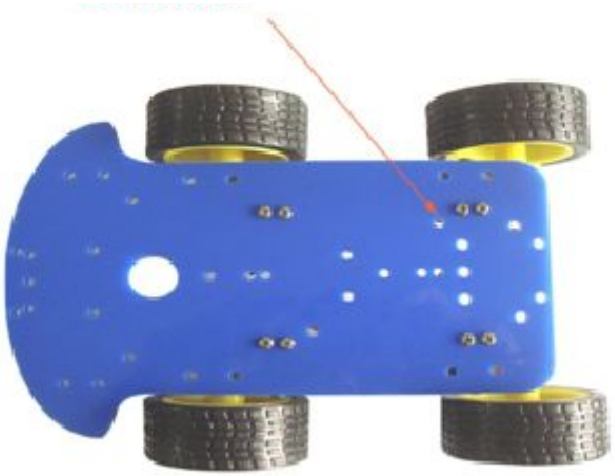


Repeat the above process and complete below assembly.

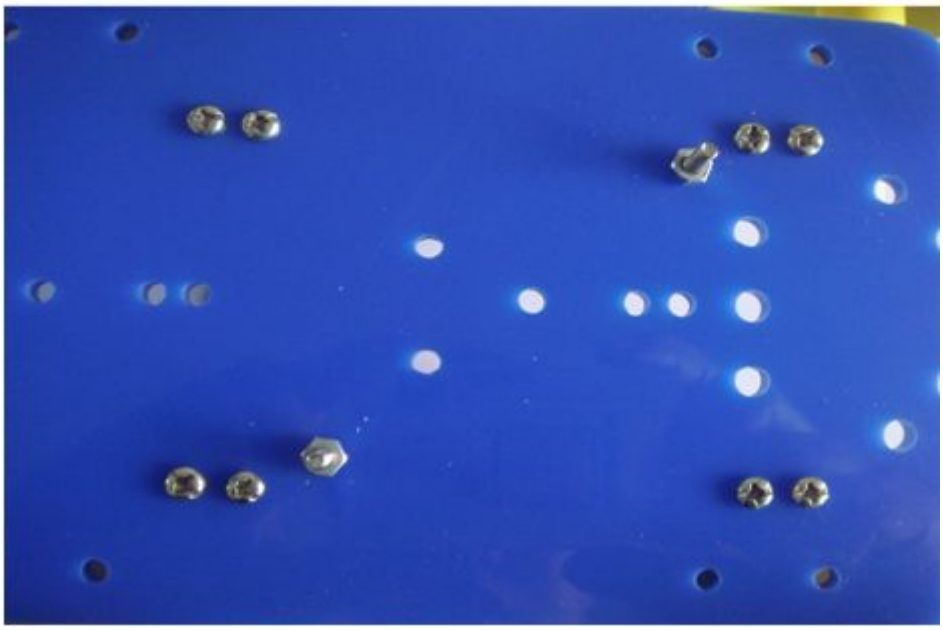


2. Installation of motor driver board

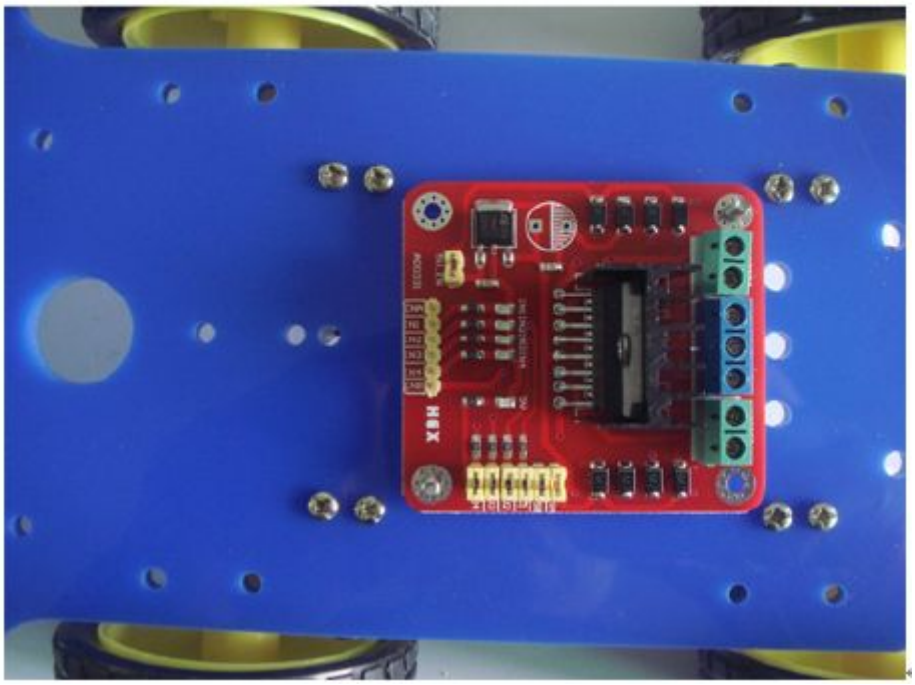
Installation location of L298N motor driver board



Place the 3*20MM screws into the mounting holes of motor driver board, place with nuts



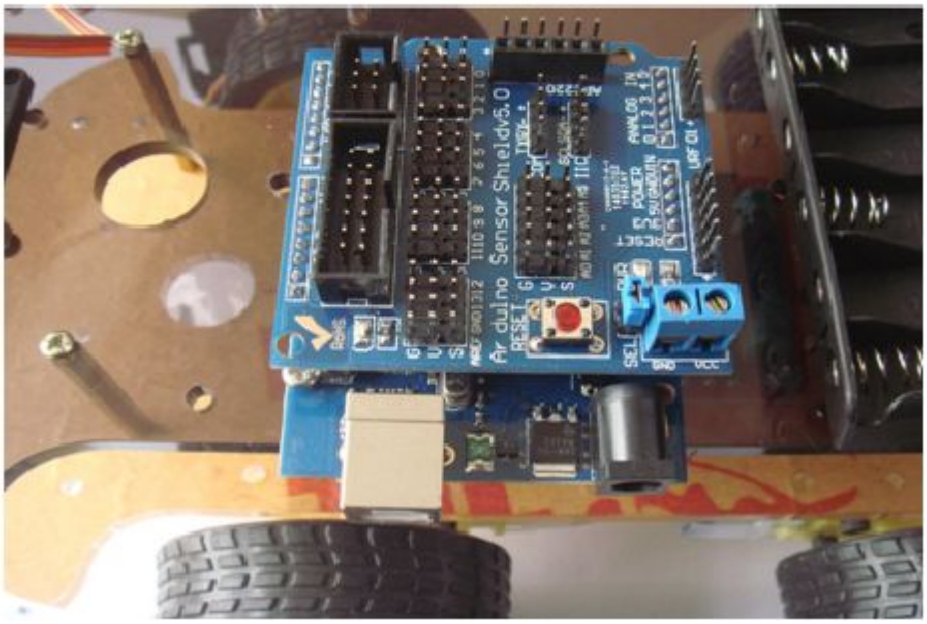
Install the L298N motor driver board



3. Installation of the main controller board

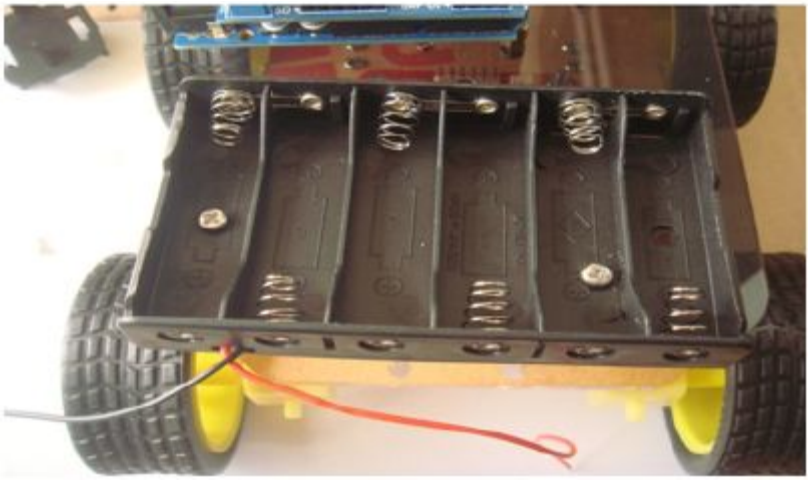
Install the copper bush of the controller board

Install the controller board to correspond mounting holes and stack the sensor shield on top of it



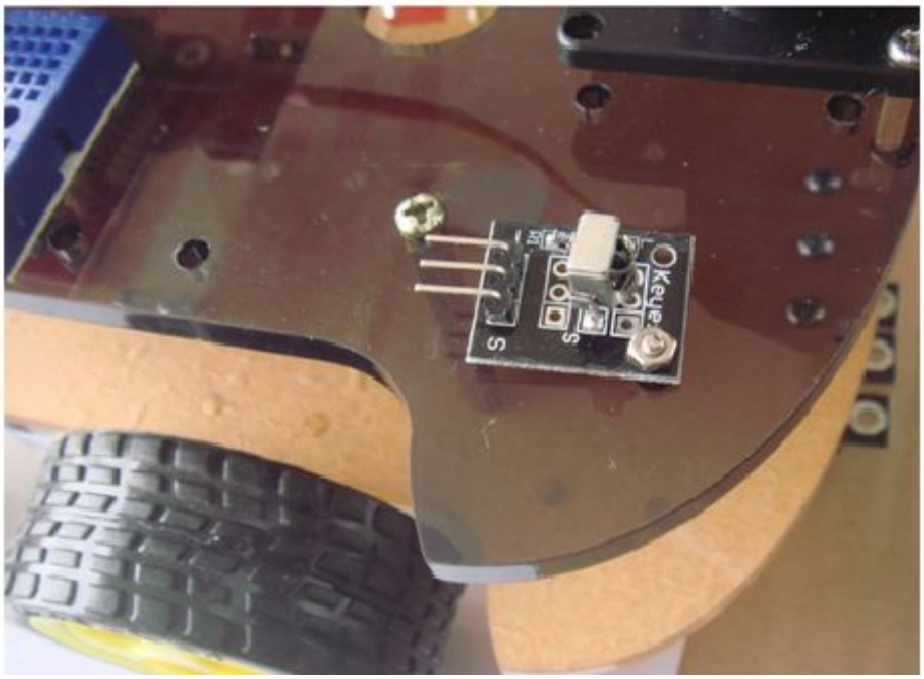
4. Install the battery case

Install the battery case



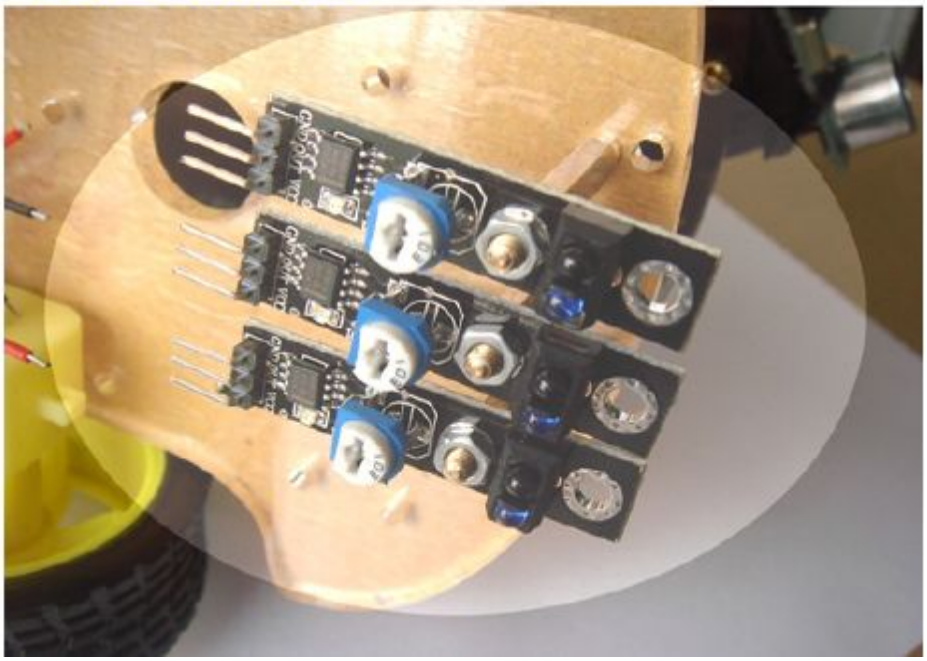
5. Install the IR receiver

Fix the IR receiver module to the mounting hole of the front part



6. Installation of line tracking module

Use three 20MM copper bush to fix the three line tracking modules to the bottom chassis



7. Installation of holder and servo motor

Holder, servo motor & ultrasonic module installation

Take out the cross-shape plastic part from the servo

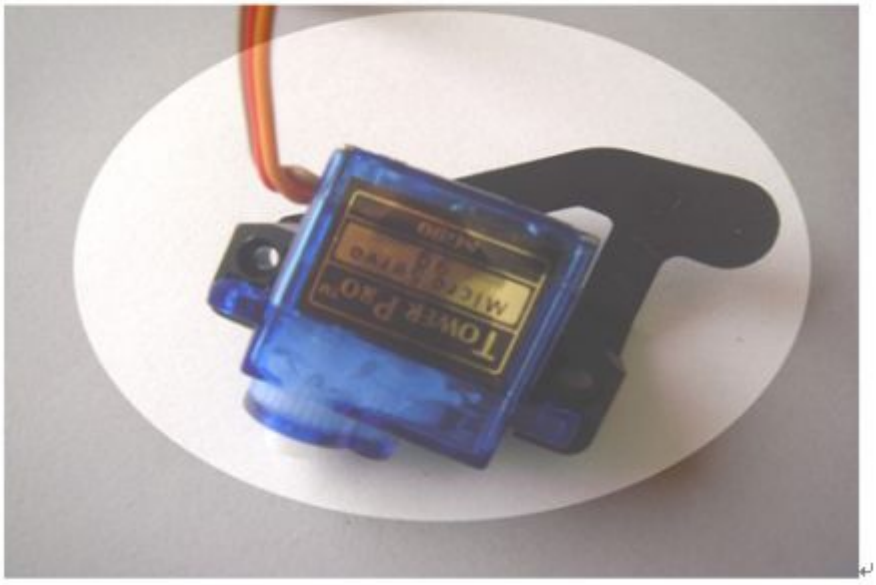
Cut the four arms into equal length

Fix the 2*8MM and 1.2*5MM in the cross-shape part and install it to the holder bottom, as picture shows.

In the holder bottom, install nuts to the screw

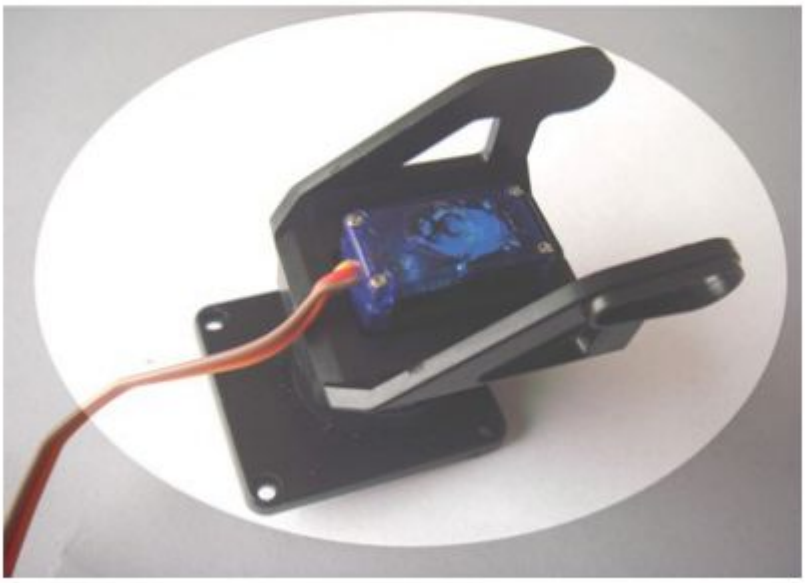
Apply heat staking glue to fix the screw

Install the servo motor to the two sides of the holder



After the above installation, fix it by screws

Place the installed Servo motor into the cross-shape plastic part and adjust the direction

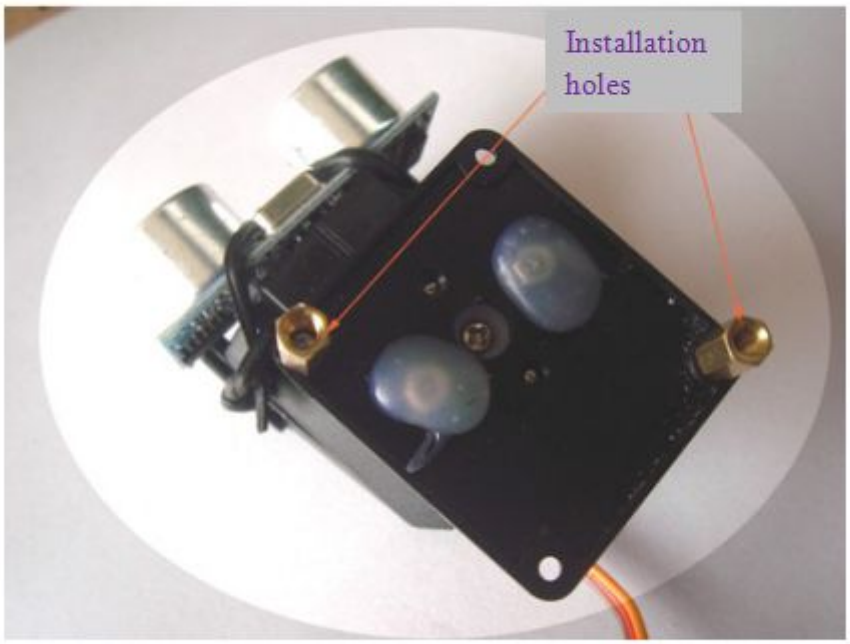


Take out the 2*6MM screw from the servo accessory bag and install it to the below indicated hole

Installation hole

Use cable to fix the ultrasonic module to the holder front

Place the 6MM copper nut into the installation holes on the holder bottom

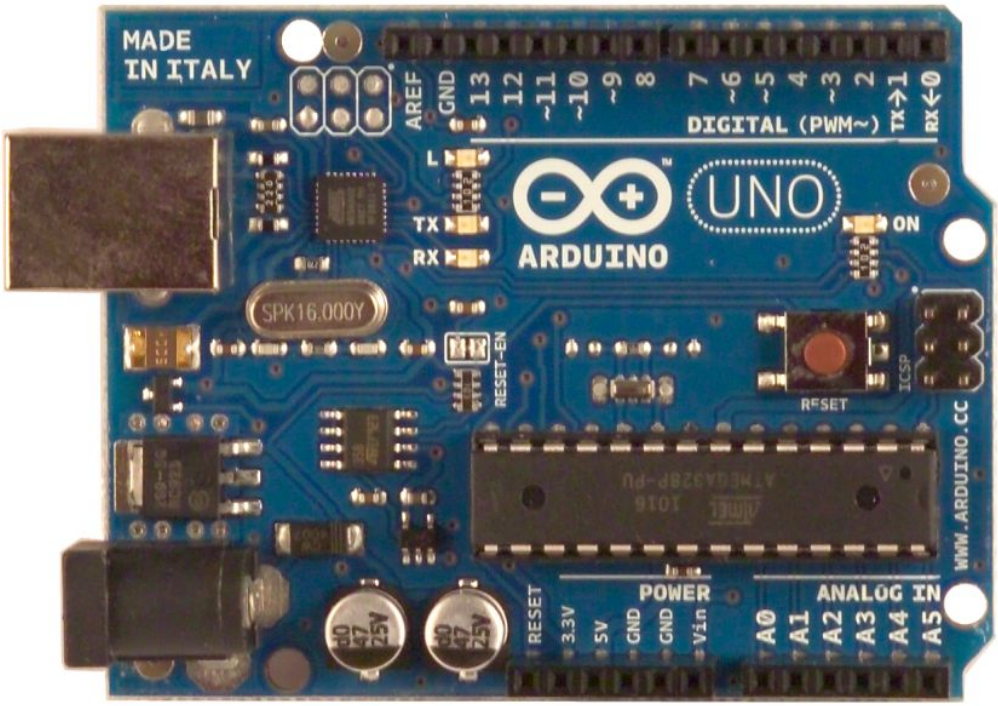


Install the assembled holder part to the car chassis

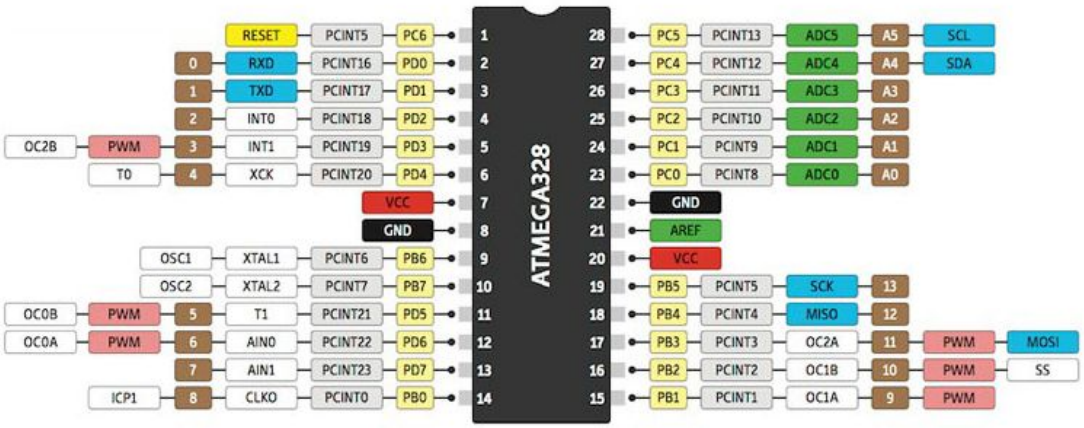
Installation is complete, thank you!

6. Application of Arduino

1. Introduction



Arduino is an open-source hardware project platform from Italy. This platform includes a circuit board with simple I/O function and program development environment software. It can be used to develop interactive products. For example, it can read signals of multiple switches and sensors, and control light, servo motor and other various physical devices. It can also be developed into PC peripherals to communicate with the software in PC.



Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. To properly define Arduino, we can use some examples.

When the coffee is ready, do you want the coffee pot to give you a "squeak" sound for reminder?

When there is a new e-mail, will your cell inform you with a mail alert?

Do you want a shiny fluffy toy?

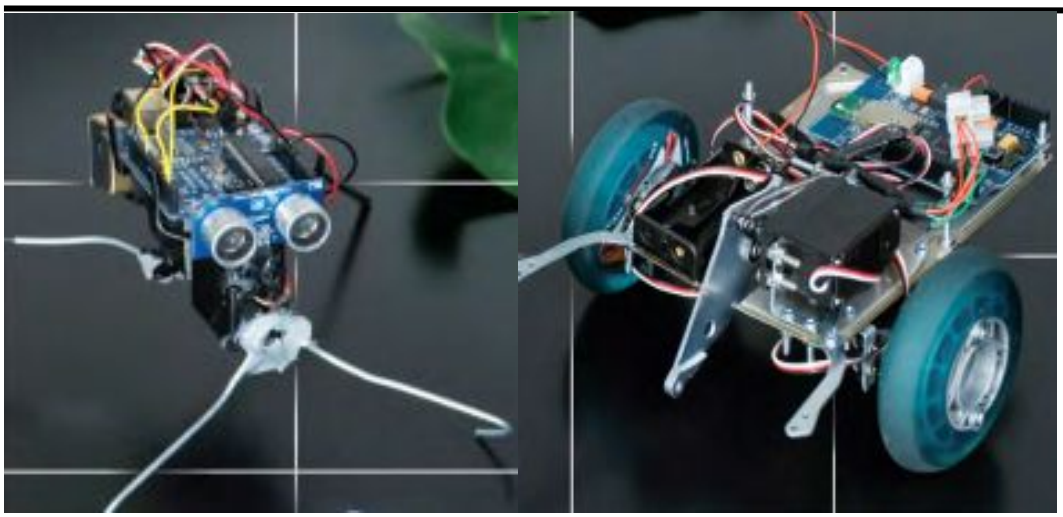
Professor X's wheelchair with voice function and beverage distribution function?

A buzzer that can do experimental test with a quick button?

Do you want to build your son an arm-cannon from Galaxy Warrior?

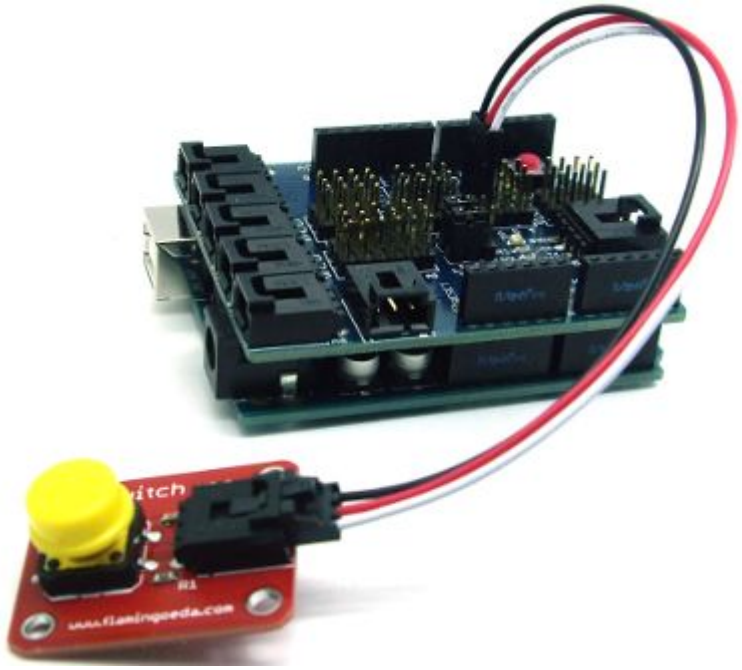
DIY heart rates monitor or record cycling data into memory card?

Have you ever thought about making a robot that can draw and can wildly run in the snow?



Well, Arduino can help you do all that.

Arduino takes off in that it can convert analog input into digital input. That is to say, Arduino can identify signal input of sensors such as photo sensor, temperature, sound or any low-cost sensor in the market. For digital sensors, Arduino supports SPI (high-speed synchronous serial port) and I2C bus. This function makes it suitable for 99% sensors in the market. Other development platform cannot easily achieve this. Just think about how odd it is to bring together a Beagleboard and an Arduino just to aquire sensor data.



Simple, but not too simple

Traditional development boards are often complicated and with a lot of accessories such as LCD screen, button, LEDs, seven segment LED display etc. Development

board can have all these functions. Number of functions on an Arduino is an absolute minimum. If you want to expend its function, you can simply add an shield. There are hundreds of thousands of Arduino shield, vary from LCD screens to wireless Internet technology. You can decide how many shields to be used. It's also easy to expend functions on a shield, also profitable for people who make it.

Not made by chip makers

Arduino development board is not designed by chip makers. Why do we emphasize this? Chip makers add strange things to their products to make them special. While Arduino stresses commonality between micro controller units rather than differences. This means Arduino is a great platform for starters. Things you can do on an Arduino board can be done on micro controllers.

2. Arduino driver installation and program download:

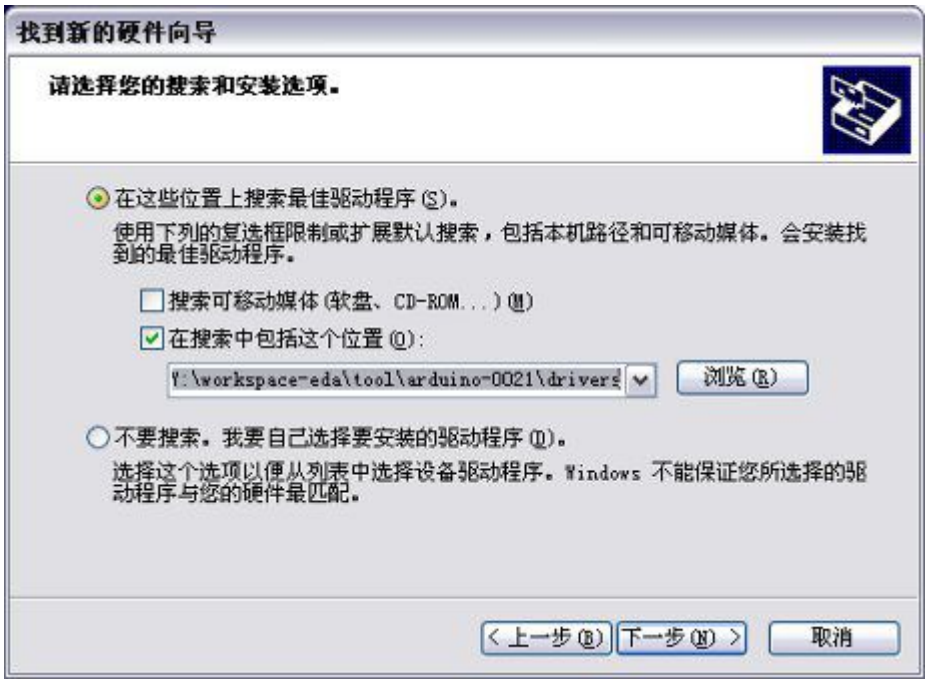
First, download the Arduino development software, web address:
http://arduino.cc/en/Main/Software
Downloaded file is a arduino-0023.zip compressed folder, unzip it to your hard drive.
When the Arduino UNO is connected to Windows via USB cable, there will be pop up says new USB device named "Arduino UNO" is found,

Then Windows will guide us into the "Found New Hardware" window, select the "No, not now" option, click "Next" button:

The next step you need to install the necessary drivers Arduino UNO, select one of the

"Install from a list of specific location (Advanced)" option, click "Next" button:

Arduino UNO USB driver is placed under “drivers” directory of Arduino 0021 installation directory, we need to specify the directory as driver searching directory for driver installation:

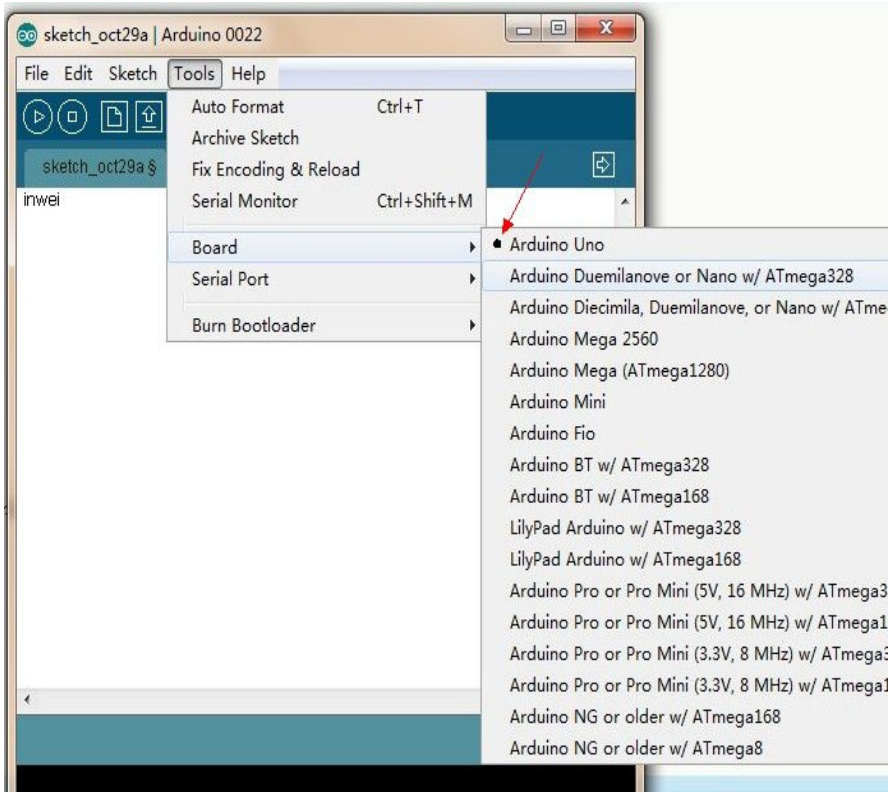


Click the "Next" button, Windows will begin to find and install Arduino UNO USB drivers:

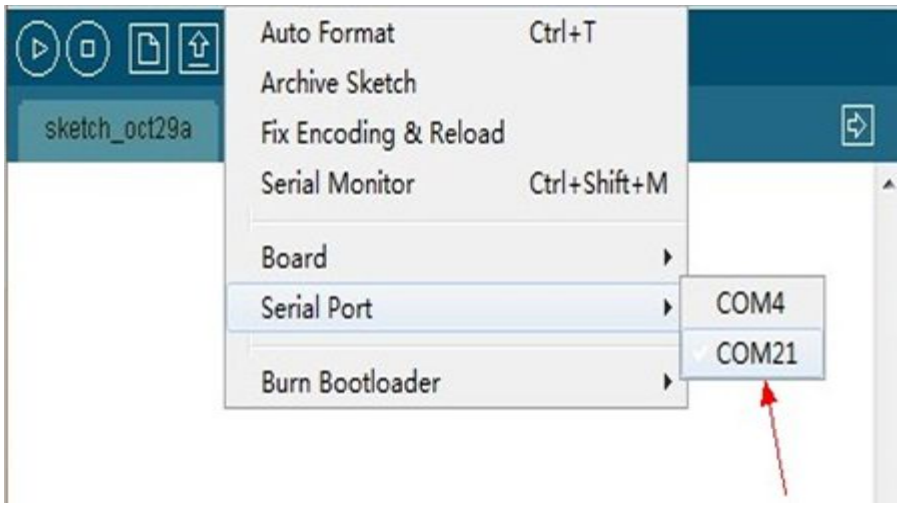
If everything goes well, we will see the following screen:

After the USB driver installation is successful, we can find the appropriate Arduino UNO serial ports in Windows Device Manager:

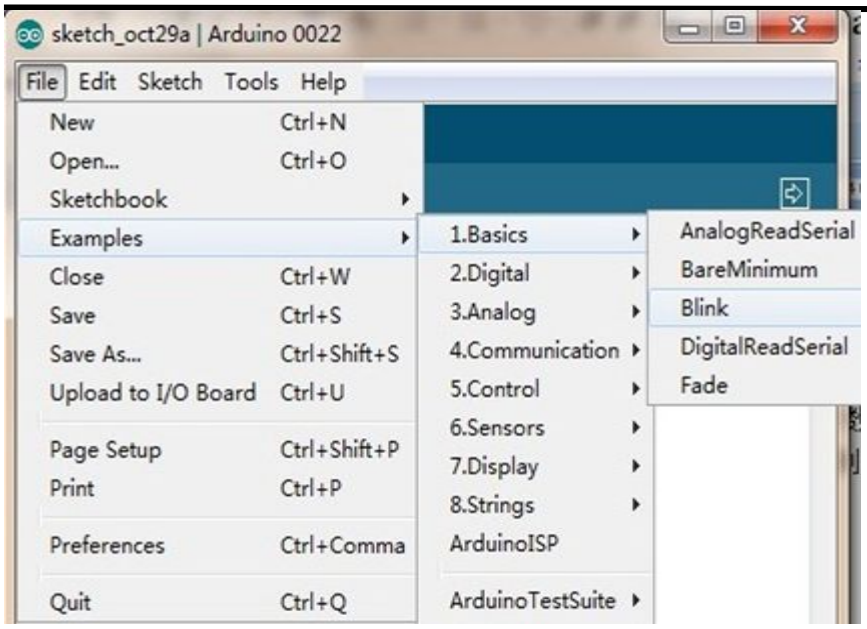
Following is an example of program uploading, called light up the LED "L" on the main board.
In Arduino-0023 programming interface, click [Tools], move the mouse to the drop-down menu [Board] option in the pop-up submenus select [Arduino UNO].



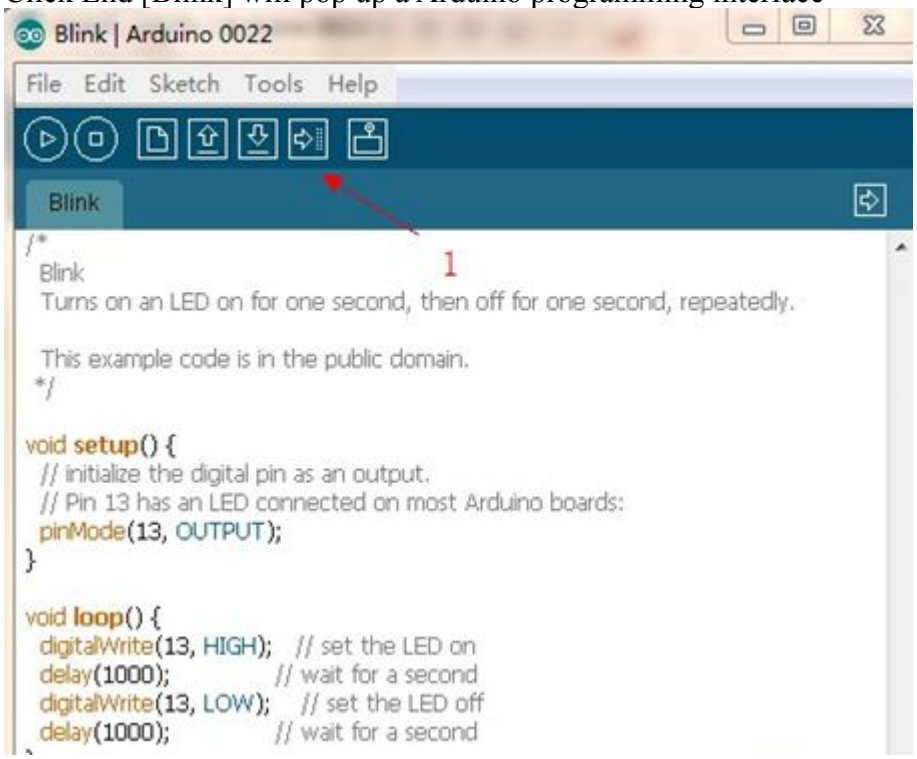
Next is to select com port. In device manager, you can see the com port number. If Arduino UNO port is 21, here click 21.



Let’s upload a "L" flashing sample program, move mouse button to [File]
Inside the pop-up menu, move the mouse to pull down [Examples], the menu extend to the right to [1.Basics]
Move the mouse to [1.Basics] After the menu continues to expand, find the [Blink], left-click [Blink]



Click End [Blink] will pop up a Arduino programming interface



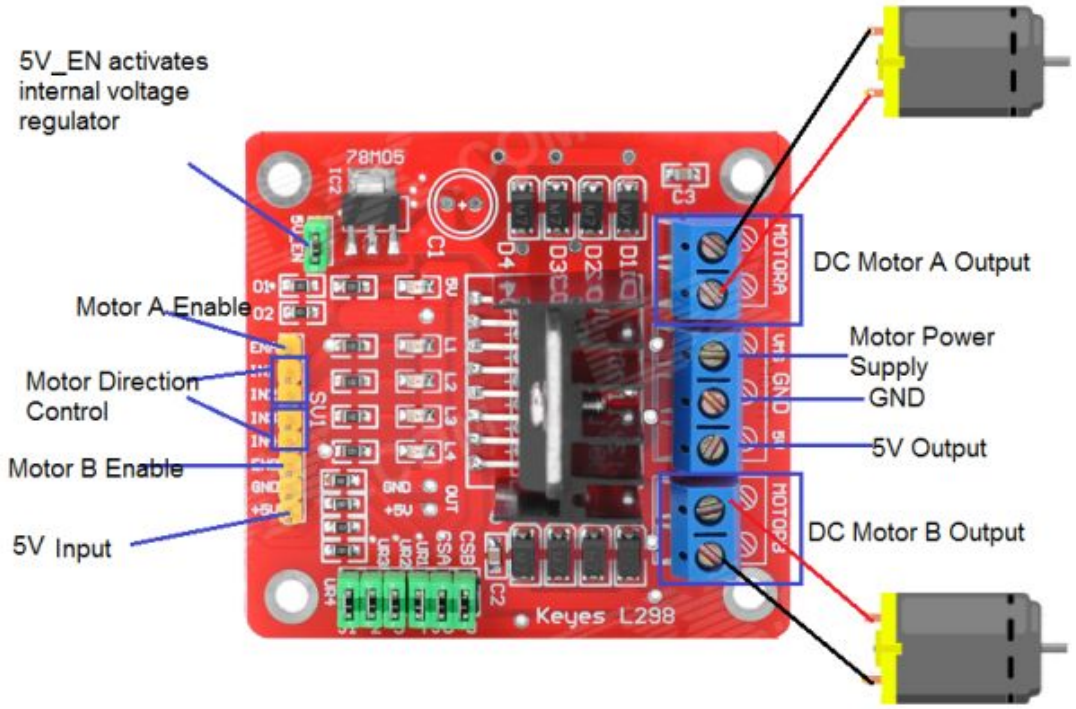
Directly left click red arrow indicated icon, you can see the Arduino UNO motherboards have two yellow lights flash for a while, along with two yellow lights that flashes much quicker. In the compiling page below, you can see it says “Done uploading”. The “L” LED on the main board will be on for 1 second and off for 1 second.

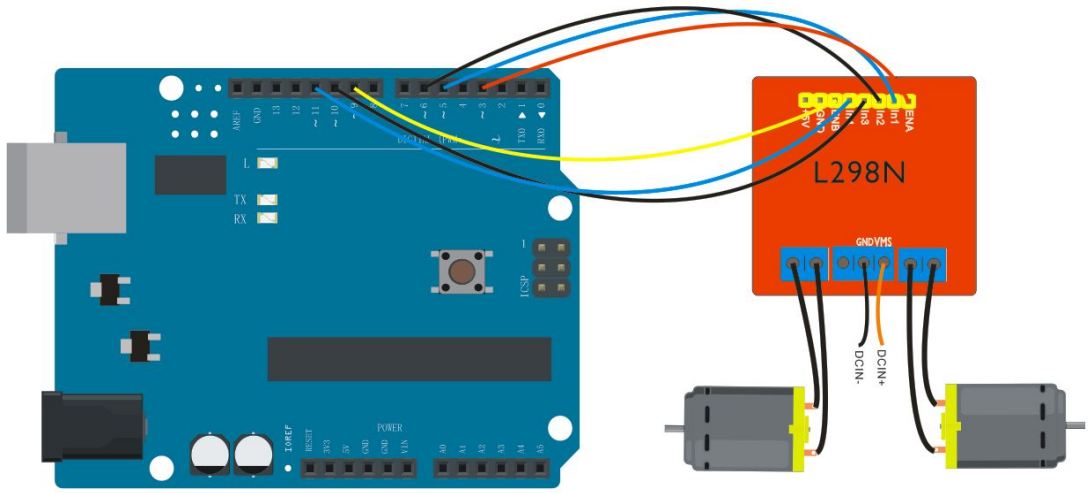
So congratulations, your first program has been a success! ! !

7. Project details

1. The application of L298N motor driver board

For the instruction for L298N driver board please refer to (L298N dual-H bridge DC motor driver board manual). Some of you still don't know how to control dual DC motor. Here are the details.





For the VMS driver part, power supply can be external power source, generally about 9V. For the logic part, power supply can be from the board internally; with terminals in suspense state, they can also be connected to +5V to +7V power. The three pins between each terminal are used to control the dual DC motor. EA and EB are connected to Arduino PWM interface for motor speed regulation. I1, I2, I3, I2 interface, connected to Arduino digital interfaces, are used for controlling the motor going forward, backward, steering and braking. Up until now, the preparatory work is completed. You can begin writing the program now. Here, the program for your reference

Program as follows:

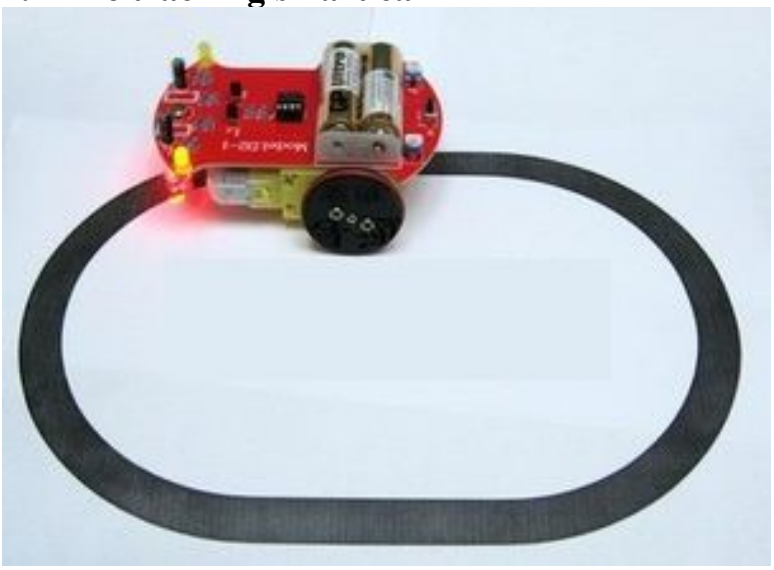
int pinI1 = 8 ;/ / define interfaces I1

```
int pinI2 = 9 ;/ / define I2 interfaces
int speedpin = 11 ;/ / define EA (PWM control) Interface
int pinI3 = 6 ;/ / define I3 Interface
int pinI4 = 7 ;/ / define I4 Interface
int speedpin1 = 10 ;/ / define EB (PWM control) Interface
void setup ()
{
pinMode (pinI1, OUTPUT);
pinMode (pinI2, OUTPUT);
pinMode (speedpin, OUTPUT);
pinMode (pinI3, OUTPUT);
pinMode (pinI4, OUTPUT);
pinMode (speedpin1, OUTPUT);
}
void loop ()
{
/ / Straight
analogWrite (speedpin, 100) ;/ / set the speed of the input analog value
analogWrite (speedpin1, 100);
digitalWrite (pinI4, LOW) ;/ / make DC motor (right) turn counterclockwise
digitalWrite (pinI3, HIGH);
digitalWrite (pinI1, LOW) ;/ / make DC motor (left) clockwise
digitalWrite (pinI2, HIGH);
delay (2000);
/ / Back
analogWrite (speedpin, 100) ;/ / set the speed of the input analog value
analogWrite (speedpin1, 100);
digitalWrite (pinI4, HIGH) ;/ / make DC motor (right) clockwise
digitalWrite (pinI3, LOW);
digitalWrite (pinI1, HIGH) ;/ / make DC motor (left) turn counterclockwise
digitalWrite (pinI2, LOW);
delay (2000);
/ / Left
analogWrite (speedpin, 60) ;/ / set the speed of the input analog value
analogWrite (speedpin1, 60);
digitalWrite (pinI4, LOW) ;/ / make DC motor (right) turn counterclockwise
digitalWrite (pinI3, HIGH);
digitalWrite (pinI1, HIGH) ;/ / make DC motor (left) turn counterclockwise
digitalWrite (pinI2, LOW);
delay (2000);
/ / Right
analogWrite (speedpin, 60) ;/ / set the speed of the input analog value
analogWrite (speedpin1, 60);
digitalWrite (pinI4, HIGH) ;/ / make DC motor (right) clockwise
digitalWrite (pinI3, LOW);
digitalWrite (pinI1, LOW) ;/ / make DC motor (left) clockwise
digitalWrite (pinI2, HIGH);
delay (2000);
/ / Brake
digitalWrite (pinI4, HIGH) ;/ / make DC motor (right) brake
digitalWrite (pinI3, HIGH);
digitalWrite (pinI1, HIGH) ;/ / make DC motor (left) brake
digitalWrite (pinI2, HIGH);
delay (2000);
}
```

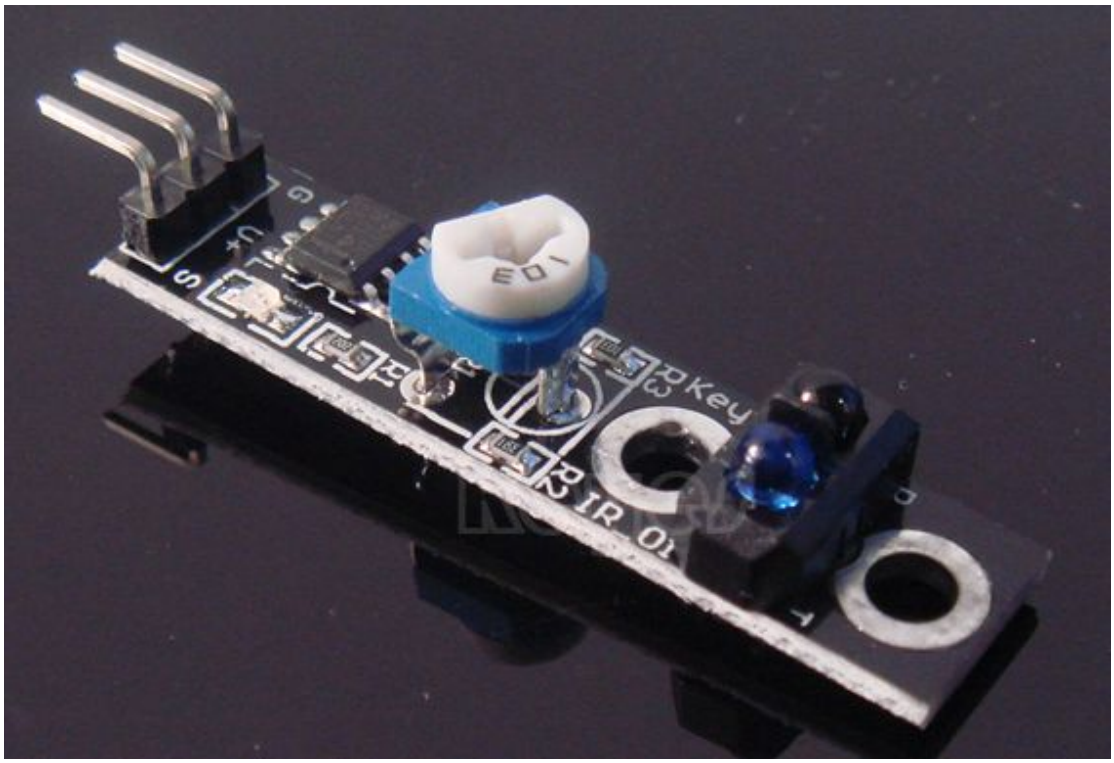
Note: in the program, there can be other ways to make the motor turning left or right. You can try

it out yourself.

2. Line tracking smart car



Line tracking principle: the working principle of the TCRT5000 infrared double tube is to use the infrared's different reflectivity of different color, and convert the strength of the reflected signal into current signal. For black and white line tracking module, it's high level signal when detecting black line, low level when detecting white line. Detect height is 0-3cm.
Note: you can use a knob potentiometer in the circuit to adjust the sensitivity of this tracking module.
TCRT5000 infrared double tube is widely applied in robot design, and industrial manufacture. It can be used for black and white line tracking robot and industrial counting sensors.



Usage:
1. For the sensor, there are 3 pins, namely GND, VCC, OUT. VCC and GND are for power

supply. OUT is for signal output.
2. When it detects an object, the signal end outputs low level; when there is no object detected, the signal end outputs high level.
3. By judging whether the signal output is 0 or 1, it can know whether the object exists.
Performance parameter:
1. Detection distance: about 2cm for white paper floor. Different distance depends on the color difference, farthest for white color.
2. Power supply voltage: 2.5V to 12V, no more than 12V. (note: it is better to use low voltage power supply, high power supply voltage will result in shorter service life of the sensor. 5V power supply is preferred.)
Working current: 18~20ma when the supply voltage is 5V. After many tests, when the working current is 18~20ma, the sensor is at its best performance in anti-jamming capability.
When it detects an object, the signal end outputs low level; when there is no object detected, the signal end outputs high level.
5. The sensor outputs TTL level; can be directly connected to the IO port of 3.3V or 5V MCU.

Black or white line detecting principle:

1.For color black, it has characteristic of low reflectivity to light. When the surface color is not black, most of the red light the sensor sends will be reflected. The sensor outputs low level 0.

2. If there is black line on a surface, when the sensor is on it, color black has low reflectivity; reflected infrared light is not enough to reach the sensor action level, so the sensor output is still l.

3.To detect the black line, we only need a MCU to determine whether the sensor output is 0 or 1.

4. Detecting principle of white line is the same with black line. When white line is detected, the color surrounding the white line is close to black. We then adjust the adjustable resistor of the infrared sensor; lower the sensitivity until the surrounding color cannot be detected. The sensor can then detect the white line.

Sample program:

```
int pin = 7 ;/ / Defines interface Arduino digital pin7
int val ;/ / define variables
void setup ()
{
pinMode (ledPin, OUTPUT) ;/ / set the digital interface output interface 7
Serial.begin (9600) ;/ / set the serial port baud rate to 9600kbps
}
void loop ()
{
value val = digitalRead (pin) ;/ / read digital interface
Serial.println (val) ;/ value / output digital interface
}
```

Line tracking smart car

After some basic knowledge of line tracking module, let's make our own line tracking smart car.
This design of line tracking car system is base on Arduino . It includes software and hardware design. The controller unit of the car is Arduino UNO, using infrared photoelectric sensor to detect the black track and feedback the information to the Arduino board. Arduino board will have analysis of the acquired information, timely control the driver motor to adjust the moving direction of the car. So the car can drive automatically along the black track, realizing the line tracking function.
the system design is as below figure 1-1.

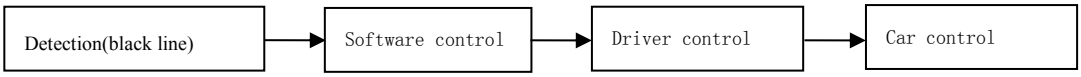


Figure 1-1

Car tracking principle: here, tracking means the car will move along the black line on the white floor. Because the black line and the white floor has different reflection coefficient, the track can be determine by the strength of the reflected light. Usually, it adopts the method of infrared detection. That is to use infrared ray's feature of reflection over different color. When the car is driving, it will continually emit infrared ray to the ground. When the infrared ray encounters the floor made of white paper. It will be reflected. The reflected ray will be received by the receiver of the car. If it encounters the black track, infrared ray will be absorbed so the receiver won't receive infrared ray. The microcontroller will determine the location of the black line and the moving direction of the car. Note that the detection rage of the infrared detector is limited.
The control system of the line tracking car consists of a main control circuit module, power supply module, infrared detecting module, motor, and a driver module.
The control system design is as below figure 2-1.

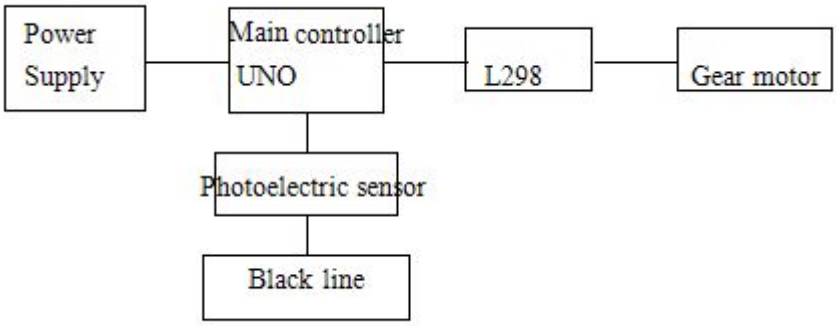


Figure2-1

Flow chart of line tracking car

When the car enters the tracking mode, it begins constantly scanning I/O port of the MCU. When the I/O port picks up a signal, it will enter the processing; firstly determine which detector detects the black line.

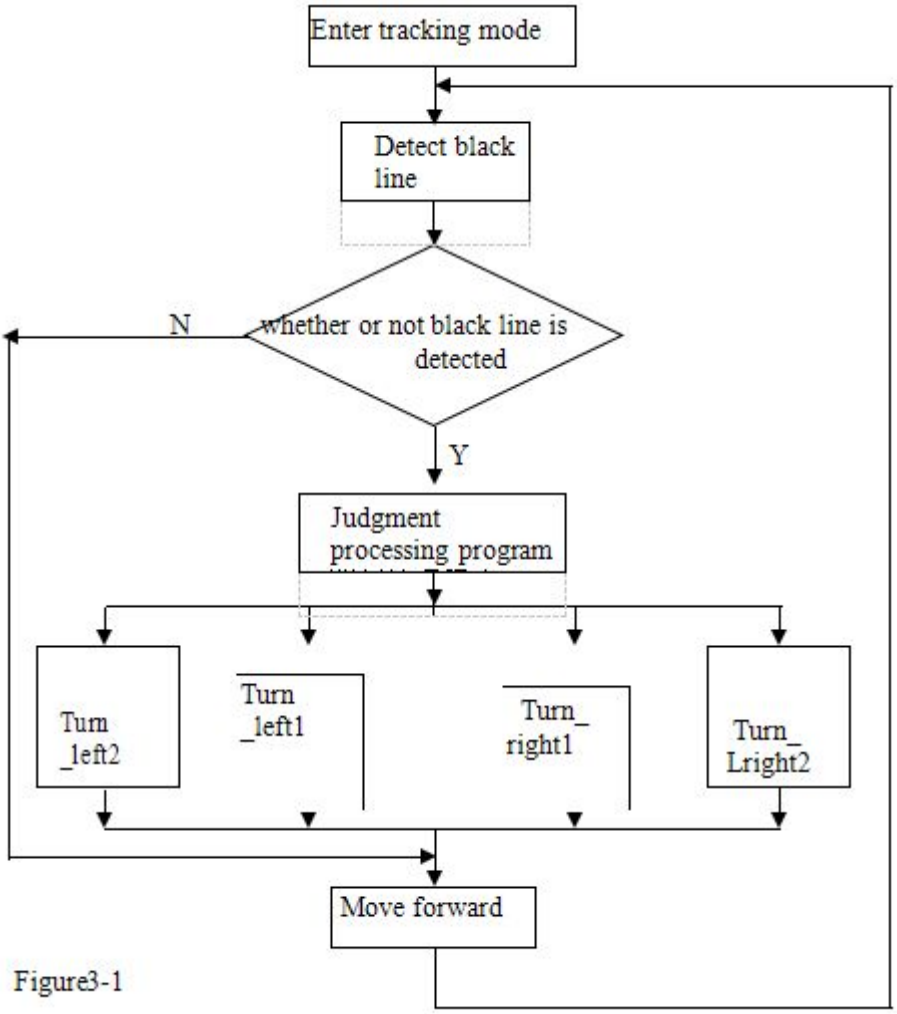
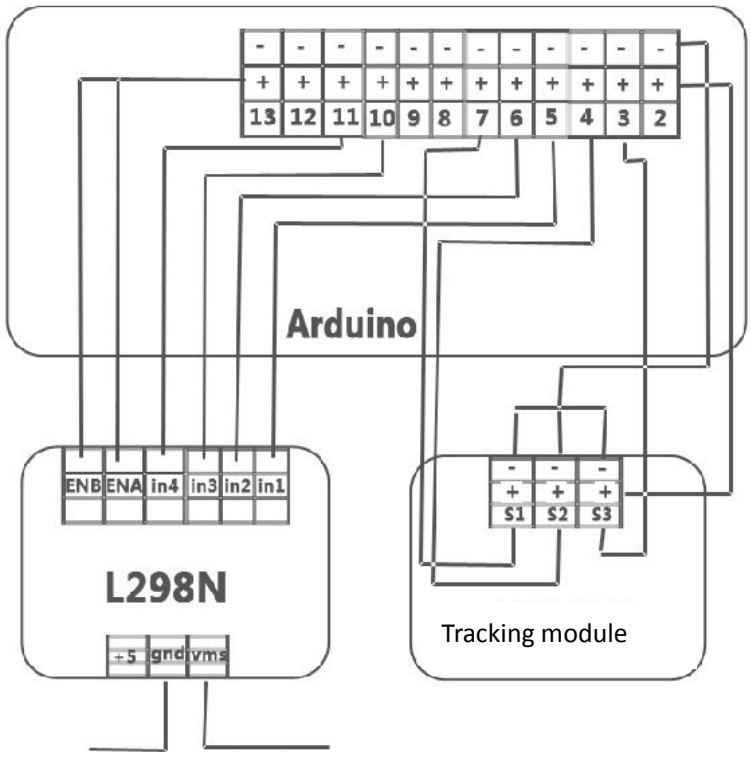


Figure3-1

Arduino tracking car wiring diagram

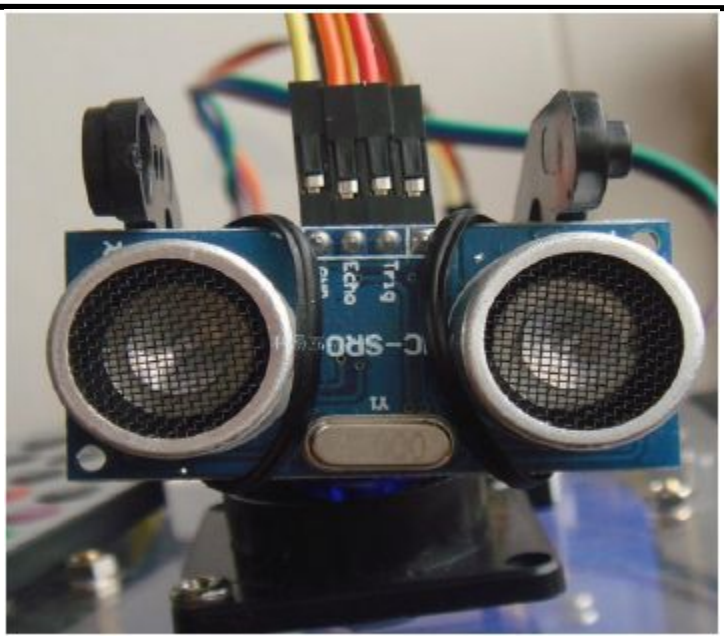


Tracking car Ardu :

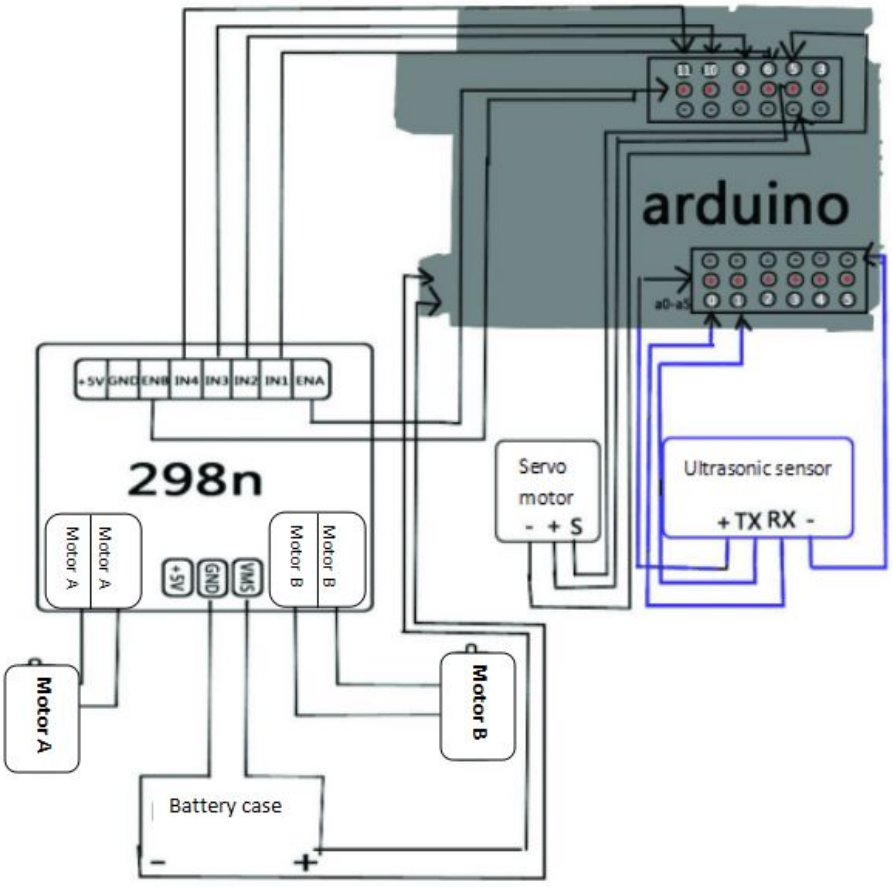
```
int MotorRight1 = 5;
int MotorRight2 = 6;
int MotorLeft1 = 10;
int MotorLeft2 = 11;
const int SensorLeft = 7;              / / Left sensor input pin
const int SensorMiddle = 4;            / / The sensor input pin
const int SensorRight = 3;             / / Right sensor input pin
int SL;           / / Left sensor status
int SM;           / / The sensor status
int SR;           / / Right sensor status

void setup ()
{
   Serial.begin (9600);
   pinMode (MotorRight1, OUTPUT);       / / Pin 8 (PWM)
   pinMode (MotorRight2, OUTPUT);       / / Pin 9 (PWM)
   pinMode (MotorLeft1,       OUTPUT);       / / Pin 10 (PWM)
   pinMode (MotorLeft2,       OUTPUT);       / / Pin 11 (PWM)
   pinMode (SensorLeft, INPUT); / / define left Sensors
   pinMode (SensorMiddle, INPUT) ;/ / definition sensors
   pinMode (SensorRight, INPUT); / / definition of the right sensor
}

void loop ()
   {
   SL = digitalRead (SensorLeft);
   SM = digitalRead (SensorMiddle);
   SR = digitalRead (SensorRight);

            if (SM == HIGH) / / in sensors in black areas
           {
```

```
        if (SL == LOW & SR == HIGH) / / left and right black white, turn left
        {
            digitalWrite (MotorRight1, LOW);
            digitalWrite (MotorRight2, HIGH);
            analogWrite (MotorLeft1, 0);
            analogWrite (MotorLeft2, 80);
        }
        else if (SR == LOW & SL == HIGH) / / left and right black white, turn right
        {
            analogWrite (MotorRight1, 0) ;/ / right turn
            analogWrite (MotorRight2, 80);
            digitalWrite (MotorLeft1, LOW);
            digitalWrite (MotorLeft2, HIGH);
        }
       else       / / Both sides white, straight
        {
            digitalWrite (MotorRight1, LOW);
            digitalWrite (MotorRight2, HIGH);
            digitalWrite (MotorLeft1, LOW);
            digitalWrite (MotorLeft2, HIGH);
              analogWrite (MotorLeft1, 200);
            analogWrite (MotorLeft2, 200);
            analogWrite (MotorRight1, 200);
            analogWrite (MotorRight2, 200);
      }
   }
  else / / the sensors in the white area
 {
     if (SL == LOW & SR == HIGH) / / left and right black white, fast turn left
    {
          digitalWrite (MotorRight1, LOW);
          digitalWrite (MotorRight2, HIGH);
          digitalWrite (MotorLeft1, LOW);
          digitalWrite (MotorLeft2, LOW);
    }
     else if (SR == LOW & SL == HIGH) / / left and right black white, quick right turn
    {
        digitalWrite (MotorRight1, LOW);
        digitalWrite (MotorRight2, LOW);
        digitalWrite (MotorLeft1, LOW);
        digitalWrite (MotorLeft2, HIGH);
    }
     else / / are white, stop
    {
    digitalWrite (MotorRight1, HIGH);
    digitalWrite (MotorRight2, LOW);
    digitalWrite (MotorLeft1, HIGH);
    digitalWrite (MotorLeft2, LOW);;
    }}}
```

3. Ultrasonic obstacle avoidance Smart Car



Ultrasonic obstacle avoidance is easy to realize, simple in calculation. It is easy to control it in real time with practical measuring accuracy. Therefore, it has become a common method for obstacle avoidance. For the application method of ultrasonic, please refer to “Arduino ultrasonic ranging instruction”. Below is the connection diagram for ultrasonic obstacle avoidance:

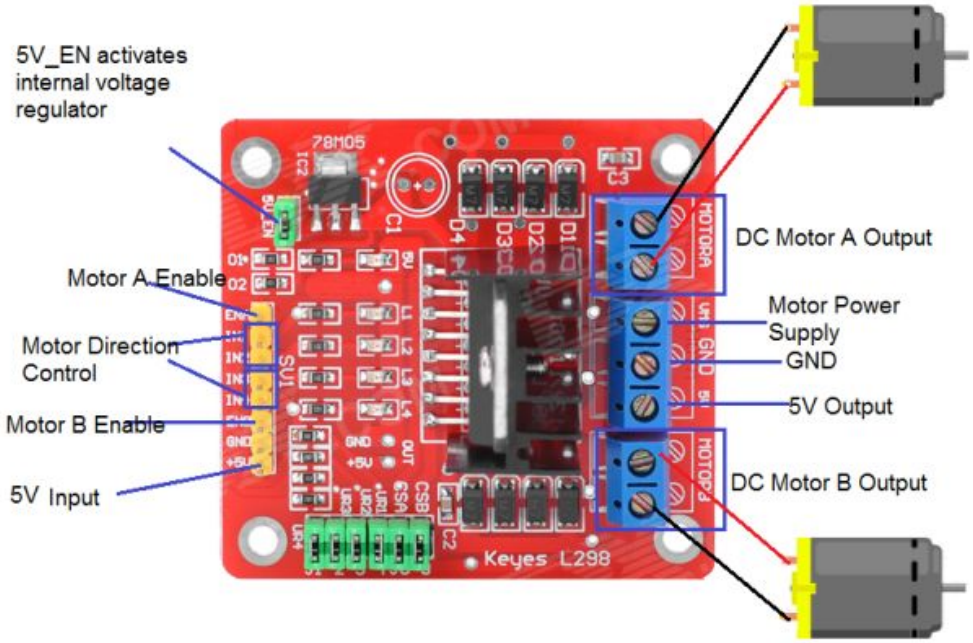


A: Motor connection

Motor 1 to L298N MOTOA

Motor 2 to L298N MOTOB

Use 1 contact of battery case of 6 cells of AA batteries to supply power for L298N motor driver module, another contact for Arduino main board. The + of the power supply for L298N motor driver module is connected to the VMS of L298N; the - to the GND. + 5V interface of L298N is not connected to anything.



Three: the motor and steering enabled (with the program)

int pinLB = 6; / / After defining the 6 pin left, then to the foot force plate PWM6

int pinLF = 9; / / Define the 9 pin left, then to the foot force plate PWM9

int pinRB = 10; / / Define pin 10 right rear, then to force the foot plate PWM10

int pinRF = 11; / / Define the 11-pin front right, then to the foot force plate PWM11

Four: Servo connections

myservo.attach (5); / / Define servo motor output section 5 pin (PWM)

Five: ultrasonic sensor connection

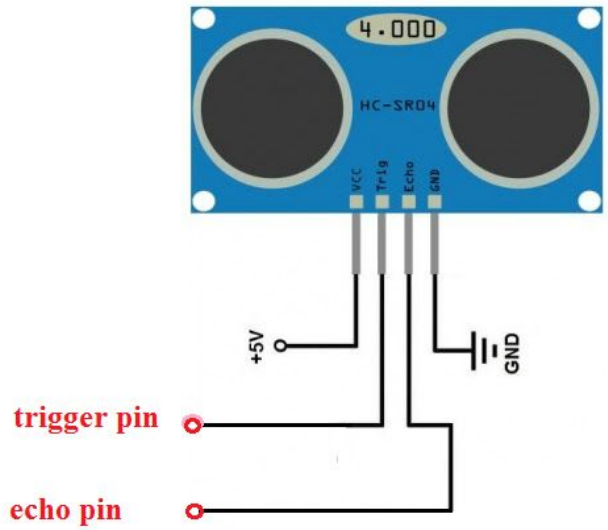
Ultrasonic sensors have four legs

VCC +5 V connection

TRIQ signal input

ECHO signal output

GND Ground




```
int inputPin = A0;       / / Define pin ultrasonic signal reception
int outputPin = A1;      / / Define pin ultrasonic signal transmitter

Ultrasonic Smart car obstacle avoidance procedures (ARDUINO)
      L = Left
      R = Right
      F = front
      B = after
* /
# Include <Servo.h>
int pinLB = 6;           / / Define pin left after 6
int pinLF = 9;           / / Define the 9-pin front left

int pinRB = 10;          / 10 pin definitions right rear /
int pinRF = 11;          / / Define the 11-pin front right

int inputPin = A0;       / / Define pin ultrasonic signal reception
int outputPin = A1;      / / Define pin ultrasonic signal transmitter

int Fspeedd = 0;             / /-Speed
int Rspeedd = 0;             / / Right speed
int Lspeedd = 0;             / / Left-speed
int directionn = 0;        / / Front Left = 8 after = 2 = 4 Right = 6
Servo myservo;                 / / Set myservo
int delay_time = 250; / / settling time after steering servo motors

int Fgo = 8;                 / / Forward
int Rgo = 6;                 / / Right
int Lgo = 4;                 / / Left
int Bgo = 2;                 / / Reverse

void setup ()
   {
   Serial.begin (9600);           / / Define motor output pin
   pinMode (pinLB, OUTPUT); / / pin 8 (PWM)
   pinMode (pinLF, OUTPUT); / / pin 9 (PWM)
```

```
    pinMode (pinRB, OUTPUT); / / pin 10 (PWM)
    pinMode (pinRF, OUTPUT); / / pin 11 (PWM)

    pinMode (inputPin, INPUT);           / / Define ultrasound input pin
    pinMode (outputPin, OUTPUT);        / / Define ultrasonic output pin

    myservo.attach (5);           / / Define servo motor output section 5 pin (PWM)
    }
void advance (int a)             / / Forward
       {
        digitalWrite (pinRB, LOW);       / / The motor (rear right) action
        digitalWrite (pinRF, HIGH);
        digitalWrite (pinLB, LOW);       / / The motor (left rear) action
        digitalWrite (pinLF, HIGH);
        delay (a * 100);
       }

void right (int b)                / / Turn right (single wheel)
       {
        digitalWrite (pinRB, LOW);        / / The motor (rear right) action
        digitalWrite (pinRF, HIGH);
        digitalWrite (pinLB, HIGH);
        digitalWrite (pinLF, HIGH);
        delay (b * 100);
       }
void left (int c)                 / / Turn left (single wheel)
       {
        digitalWrite (pinRB, HIGH);
        digitalWrite (pinRF, HIGH);
        digitalWrite (pinLB, LOW);        / / The motor (left rear) action
        digitalWrite (pinLF, HIGH);
        delay (c * 100);
       }
void turnR (int d)                / / Turn right (wheel)
       {
        digitalWrite (pinRB, LOW);       / / The motor (rear right) action
        digitalWrite (pinRF, HIGH);
        digitalWrite (pinLB, HIGH);
        digitalWrite (pinLF, LOW);       / / The motor (front left) action
        delay (d * 100);
       }
void turnL (int e)                / / Turn left (wheel)
       {
        digitalWrite (pinRB, HIGH);
        digitalWrite (pinRF, LOW);        / / The motor (front right) action
        digitalWrite (pinLB, LOW);        / / The motor (left rear) action
        digitalWrite (pinLF, HIGH);
        delay (e * 100);
       }
void stopp (int f)                / / Stop
       {
        digitalWrite (pinRB, HIGH);
        digitalWrite (pinRF, HIGH);
        digitalWrite (pinLB, HIGH);
        digitalWrite (pinLF, HIGH);
        delay (f * 100);
       }
```

```
void back (int g)                    / / Check out
      {

       digitalWrite (pinRB, HIGH);       / / The motor (rear right) action
        digitalWrite (pinRF, LOW);
       digitalWrite (pinLB, HIGH);       / / The motor (left rear) action
       digitalWrite (pinLF, LOW);
       delay (g * 100);
      }

void detection ()                / / Measure three angles (0.90.179)
      {
        int delay_time = 250;         Settling time / / servo motor after turning
        ask_pin_F ();                    / / Read from front

       if (Fspeedd <10)                / / If the distance is less than 10 cm in front
        {
        stopp (1);                       / / Clear the output data
        back (2);                        / / Check out 0.2 seconds
        }

        if (Fspeedd <25)                / / If the distance is less than 25 cm in front
        {
          stopp (1);                     / / Clear the output data
          ask_pin_L ();                  / / Read from left
          delay (delay_time);            / / Wait for a stable servo motor
          ask_pin_R ();                  / / Read from the right
          delay (delay_time);            / / Wait for a stable servo motor

          if (Lspeedd> Rspeedd)          / / If the distance is greater than the right from the left
          {
           directionn = Rgo;             / / Right away
          }

          if (Lspeedd <= Rspeedd)          / / If the left is less than or equal to the distance from
the right
          {
           directionn = Lgo;             / / Turn Left
          }

          if (Lspeedd <10 && Rspeedd <10)          / / If the distance to the left and right are less
than 10 cm distance
          {
           directionn = Bgo;             / / To go after
          }
        }
        else                               / / Add as front not less than (greater than) 25 cm
        {
          directionn = Fgo;                / / Move forward
        }

      }
void ask_pin_F ()         / / Measure the distance from the front
      {
         myservo.write (90);
        digitalWrite (outputPin, LOW);          / / Let ultrasonic transmitter low voltage 2 μ s
        delayMicroseconds (2);
```

```
          digitalWrite (outputPin, HIGH);         / / Let ultrasonic transmitter high voltage 10 μ s, where at least 10 μ s
          delayMicroseconds (10);
          digitalWrite (outputPin, LOW);          / / Maintain low voltage ultrasonic transmitter
          float Fdistance = pulseIn (inputPin, HIGH);       / / Read worse time difference
          Fdistance = Fdistance/5.8/10;             / / Time to turn to the distance (unit: cm)
          Serial.print ("F distance:");             / / Output distance (unit: cm)
          Serial.println (Fdistance);               / / Display the distance
          Fspeedd = Fdistance;                              / / Read into the distance Fspeedd (former speed)
       }
    void ask_pin_L ()        / / Measure the distance from the left
       {
          myservo.write (5);
          delay (delay_time);
          digitalWrite (outputPin, LOW);         / / Let ultrasonic transmitter low voltage 2 μ s
          delayMicroseconds (2);
          digitalWrite (outputPin, HIGH);        / / Let ultrasonic transmitter high voltage 10 μ s, where at least 10 μ s
          delayMicroseconds (10);
          digitalWrite (outputPin, LOW);          / / Maintain low voltage ultrasonic transmitter
          float Ldistance = pulseIn (inputPin, HIGH);       / / Read worse time difference
          Ldistance = Ldistance/5.8/10;             / / Time to turn to the distance (unit: cm)
          Serial.print ("L distance:");             / / Output distance (unit: cm)
          Serial.println (Ldistance);               / / Display the distance
          Lspeedd = Ldistance;                      / / Read into the distance Lspeedd (left-speed)
       }
 void ask_pin_R ()        / / Measure the distance from the right
       {
          myservo.write (177);
          delay (delay_time);
          digitalWrite (outputPin, LOW);         / / Let ultrasonic transmitter low voltage 2 μ s
          delayMicroseconds (2);
          digitalWrite (outputPin, HIGH);        / / Let ultrasonic transmitter high voltage 10 μ s, where at least 10 μ s
          delayMicroseconds (10);
          digitalWrite (outputPin, LOW);          / / Maintain low voltage ultrasonic transmitter
          float Rdistance = pulseIn (inputPin, HIGH);       / / Read worse time difference
          Rdistance = Rdistance/5.8/10;             / / Time to turn to the distance (unit: cm)
          Serial.print ("R distance:");             / / Output distance (unit: cm)
          Serial.println (Rdistance);               / / Display the distance
          Rspeedd = Rdistance;                              / / Will read into the distance Rspeedd (Right-speed)
       }

 void loop ()
    {
       myservo.write (90);       / / Let servo motor position ready to return to the pre-prepared next time measurement
       detection ();              / / Measure the angle and direction of judgment to where to move

      if (directionn == 2)       / / If directionn (direction) = 2 (reverse)
      {
         back (8);                              / /       Retrogression (car)
         turnL (2);                             / / Move slightly to the left (to prevent stuck in dead alley)
         Serial.print ("Reverse");         / / Display direction (backwards)
```

```
    }
    if (directionn == 6)                    / / If directionn (direction) = 6 (right turn)
    {
      back (1);
      turnR (6);                              / / Right
      Serial.print ("Right");           / / Display direction (turn left)
    }
    if (directionn == 4)                  / / If directionn (direction) = 4 (turn left)
    {
      back (1);
      turnL (6);                              / / Left
      Serial.print ("Left");            / / Display direction (turn right)
    }
    if (directionn == 8)                  / / If directionn (direction) = 8 (forward)
    {
     advance (1);                             / / Normal Forward
     Serial.print ("Advance");         / / Display direction (forward)
     Serial.print ("          ");
    }
  }
```

4.Infrared remote control smart car

Before the experiment:

1. Place the function library IRremote into the Arduino "libraries" directory.

2. Open IrReceive.pde to acquire the code for your infrared remote control (IRcode will be displayed in Serial Monitor), write down IRcode and modify it in your IR code in the program.

```
 / *
    * IRRemote infrared remote control code test
    *          Examples 1.2: Show infrared protocol type, such as NEC, Sony SIRC, Philips RC5, Philips RC6 and other agreements
    * /
 # Include <IRremote.h>                          / / Function library references IRRemote

 const int irReceiverPin = 2;                 / / OUTPUT signals IR receiver connected to pin 2

 IRrecv irrecv (irReceiverPin);                 / / Define an object to receive infrared signals IRrecv
 decode_results results;                          / / Decoding results will result in structural variables in decode_results

 void setup ()
 {
    Serial.begin (9600);                          / / Open Serial port, the communication speed is 9600 bps
    irrecv.enableIRIn ();                        / / Start infrared decoding
 }

 / / Display the type of infrared protocol
 void showIRProtocol (decode_results * results)
 {
    Serial.print ("Protocol:");

    / / Judgment infrared protocol types
    switch (results-> decode_type) {
```

```
      case NEC:
         Serial.print ("NEC");
         break;
      case SONY:
         Serial.print ("SONY");
         break;
      case RC5:
         Serial.print ("RC5");
         break;
      case RC6:
         Serial.print ("RC6");
         break;
      default:
         Serial.print ("Unknown encoding");
    }

    / / Print the infrared codes to Serial port
    Serial.print (", irCode:");
    Serial.print (results-> value, HEX);           / / Infrared code
    Serial.print (",        bits: ");
    Serial.println (results-> bits);               / / Number of bits coded infrared
}

void loop ()
{
    if (irrecv.decode (& results)) {                  / / Decoding is successful, you receive a set of
infrared signals
       showIRProtocol (& results);                    / / Display the type of infrared protocol
       irrecv.resume ();                              / / Continue to accept a set of infrared signals
    }
}
```

Replace the IR code in the IR control part program with the code from the test result

```
//   ****Infrared control part********
long advence = 0x00EF807F;
long back = 0x00EFA05F;
long stop = 0x00EF906F;
long left = 0x00EF00FF;
long right = 0x00EF40BF;
```

Infrared remote smart car program

```
/ / ****** Infrared remote smart car program *******
# Include <IRremote.h>
int RECV_PIN = A0;
int pinLB = 6 ;/ / define interfaces I1
int pinLF = 9 ;/ / define I2 interfaces
int pinRB = 3 ;/ / define I3 Interface
int pinRF = 5 ;/ / define I4 Interface
/ / ****** Infrared control section ********
long advence = 0x00EF    807F    ;
long back = 0x00EFA    05F    ;
long stop = 0x00EF    906F    ;
long left = 0x00EF00FF;
long right = 0x00EF40BF;

IRrecv irrecv (RECV_PIN);
decode_results results;
```

```
void dump (decode_results * results) {
   int count = results-> rawlen;
   if (results-> decode_type == UNKNOWN)
      {
       Serial.println ("Could not decode message");
      }
   else
     {
      if (results-> decode_type == NEC)
         {
          Serial.print ("Decoded NEC:");
         }
      else if (results-> decode_type == SONY)
         {
          Serial.print ("Decoded SONY:");
         }
      else if (results-> decode_type == RC5)
         {
          Serial.print ("Decoded RC5:");
         }
      else if (results-> decode_type == RC6)
         {
          Serial.print ("Decoded RC6:");
         }
        Serial.print (results-> value, HEX);
        Serial.print ("(");
        Serial.print (results-> bits, DEC);
        Serial.println ("bits)");
    }
        Serial.print ("Raw (");
        Serial.print (count, DEC);
        Serial.print ("):");

   for (int i = 0; i <count; i + +)
       {
        if ((i% 2) == 1) {
        Serial.print (results-> rawbuf [i] * USECPERTICK, DEC);
       }
      else
        {
        Serial.print (- (int) results-> rawbuf [i] * USECPERTICK, DEC);
       }
      Serial.print ("");
       }
        Serial.println ("");
       }

void setup ()
   {
   pinMode (RECV_PIN, INPUT);
   pinMode (pinLB, OUTPUT);
   pinMode (pinLF, OUTPUT);

   pinMode (pinRB, OUTPUT);
   pinMode (pinRF, OUTPUT);

Serial.begin (9600);
```

```
    irrecv.enableIRIn (); / / Start the receiver
    }

int on = 0;
unsigned long last = millis ();

void loop ()
{
     if (irrecv.decode (& results))
     {
      / / If it's been at least 1/4 second since the last
      / / IR received, toggle the relay
      if (millis () - last> 250)
          {
           on = on!;
/ /              digitalWrite (8, on HIGH:? LOW);
           digitalWrite (13, on HIGH:? LOW);
           dump (& results);
          }
      if (results.value == advence)
      {DigitalWrite (pinRB, LOW) ;/ / make DC motor (right) GO
      digitalWrite (pinRF, HIGH);
      digitalWrite (pinLB, LOW) ;/ / make DC motor (left) GO
      digitalWrite (pinLF, HIGH);}

      if (results.value == back)


        {DigitalWrite (pinRB, HIGH) ;/ / make DC motor (right) BACK
        digitalWrite (pinRF, LOW);}

      if (results.value == left)
      {DigitalWrite (pinRB, LOW) ;/ / make DC motor (right) STOP
        digitalWrite (pinRF, HIGH);
        digitalWrite (pinLB, HIGH) ;/ / make DC motor (left) GO
        digitalWrite (pinLF, LOW);}


      if (results.value == right)
      {DigitalWrite (pinRB, HIGH) ;/ / make DC motor (right) GO
        digitalWrite (pinRF, LOW);
        digitalWrite (pinLB, HIGH) ;/ / make DC motor (left) STOP
        digitalWrite (pinLF, HIGH);}

      if (results.value == stop)
          {
        digitalWrite (pinRB, HIGH) ;/ / make DC motor (right) STOP
        digitalWrite (pinRF, HIGH);
        digitalWrite (pinLB, HIGH) ;/ / make DC motor (left) STOP
        digitalWrite (pinLF, HIGH);

     }

      last = millis ();
      irrecv.resume (); / / Receive the next value
   }
}
```

5. Multi-function smart car

1. Before the experiment:

1) Place file “IRremote” into the directory “Libraries” of Arduino.

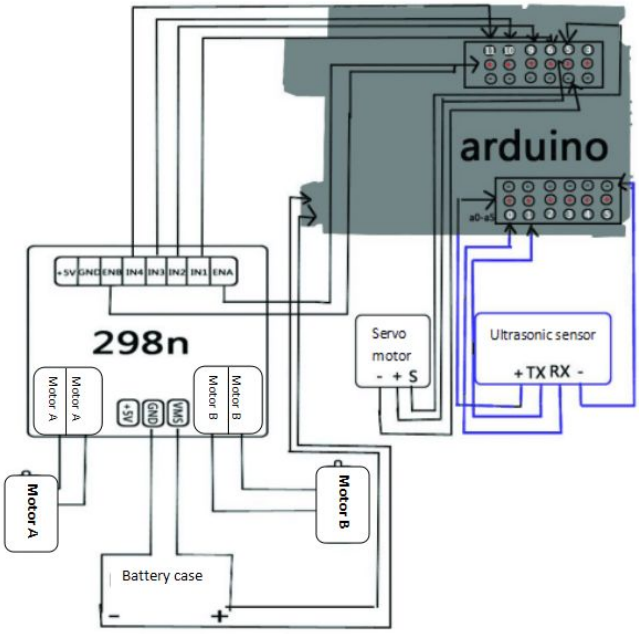
2) Open code IrReceive.pde to acquire the code for your infrared remote control (IRcode will be displayed in Serial Monitor), write down IRcode and modify it in your IR code in the multi-function program.

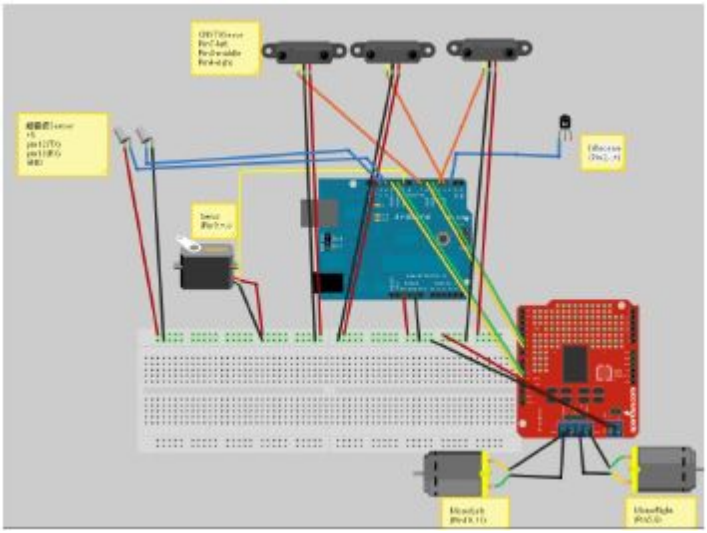
7 buttons will be needed:
<stop><line tracking mode><ultrasonic mode><go forward><go backward><turn left><turn right>

2. Operation:
Set your own IR code in the program and you are good yo go; you can adjust it to line tracking or ultrasonic mode. If you want to stop, just press the “stop” button and you can change the mode. For ultrasonic mode, press “stop” at the beginning of the detection direction.

3. Hardware required:
Arduino328 controller board *1
Arduino sensor shield *1
L298N motor driver board *1
IR remote controller + IR receiver sensor
Line tracking sensor *3
4-wheel smart car chassis *1
Servo motor *1
Holder *1
Ultrasonic sensor module *1

4. Wire connection





1) Motor connection

//**********define pin for motors

int MotorRight1=5;

int MotorRight2=6;

int MotorLeft1=10;

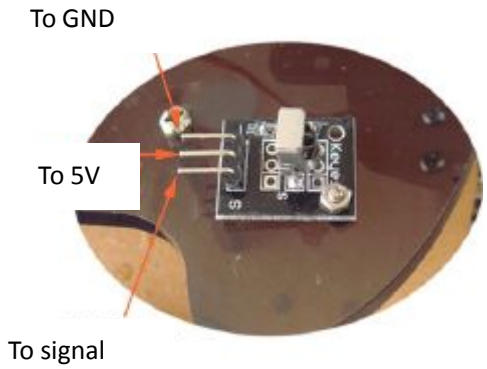
int MotorLeft2=11;

Above are pins on controller board for L298N, ENA and ENB of L298N connect to V (power +) ports of the shield; I1, I2, I3, I4 to D5, 6, 10, 11 of the shield (controlling the rotation direction of the motors).

(for the power supply and motor connection of L298N, pls refer to the introduction of L298N motor driver board)

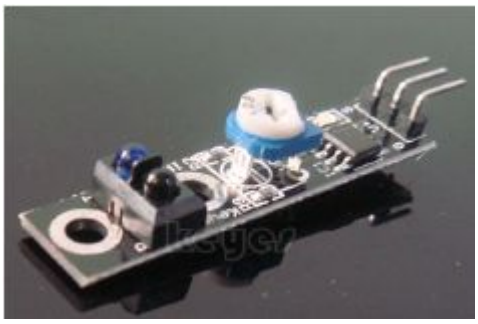
2) wire connection of IR receiver sensor

const int irReceiverPin = 2; // IR receiver, signal pin connect to pin 2



GND pin of IR receiver sensor connect to G port of the shield,
+5V pin of IR receiver sensor connect to +5V port of the shield,
S pin of IR receiver sensor connect to D2 port of the shield.

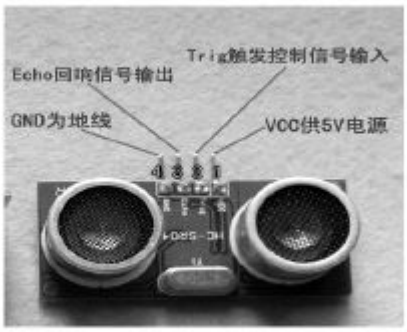
3) Connection of line tracking sensor



const int SensorLeft = 7; // left sensor input pin
const int SensorMiddle= 4 ; // middle sensor input pin
const int SensorRight = 3; // right sensor input pin

Three channels of line tracking sensor
First one signal pin to D7 of the shield, VCC to +5V of the shield, GND to G port of the shield.
Second one signal pin to D4 of the shield, VCC to +5V of the shield, GND to G port of the shield.
Third one signal pin to D3 of the shield, VCC to +5V of the shield, GND to G port of the shield.

4) Connection of ultrasonic sensor



int inputPin =13 ; // define signal input pin rx for ultrasonic sensor
int outputPin =12; // define signal output pin tx for ultrasonic sensor
Ultrasonic sensor has 4 pins, namely VCC, TRIG, ECHO, GND.
VCC connect to +5V of the shield
TRIG connect to D13 of the shield as signal input
ECHO connect to D12 of the shield as signal output
GND connect to G of the shield

5) Connection of servo motor
myservo.attach(9); // set motor to pin D9 (PWM)

The servo motor has three wires, gray, red and yellow.
For gray---GND connect to G port of the shield
Red---+5V connect to +5V of the shield
Yellow ---signal pin connect to D9 of the shield

5. Full program of smart car

```
#include <IRremote.h>
#include <Servo.h>
//**********************define motor pin************************
int MotorRight1=5;
int MotorRight2=6;
int MotorLeft1=10;
```

```
int MotorLeft2=11;
int counter=0;
const int irReceiverPin = 2; // define IR receiver pin

//**********************set up IRcode************************
long IRfront= 0x00FF629D;              // going upward
long IRback=0x00FFA857;                // backward
long IRturnright=0x00FF22DD;       // turn right
long IRturnleft= 0x00FFC23D;         // turn left
long IRstop=0x00FF02FD;                // stop
long IRcny70=0x00FF6897;               // CNY70mode
long IRAutorun=0x00FF9867;           // ultrasonic mode
long IRturnsmallleft= 0x00FFB04F;
//**********************define pins for CNY7*********************************
const int SensorLeft = 7;           // left sensor pin
const int SensorMiddle= 4 ;        // middle sensor pin
const int SensorRight = 3;          // right sensor pi
int SL;        // left sensor status
int SM;        // middle sensor status
int SR;        // right sensor status
IRrecv irrecv(irReceiverPin);    // define IRrecv to receive IR signal
decode_results results;             // place decode result in decode_results val
//************************* define pin for ultrasonic ****************************
int inputPin =13 ; // define signal receiving pin rx
int outputPin =12; // define signal transmitting pin tx
int Fspeedd = 0; // distance upfront
int Rspeedd = 0; // distance on the right
int Lspeedd = 0; // distance on the left
int directionn = 0; // front=8 back=2 left=4 right=6
Servo myservo; // set myservo
int delay_time = 250; // stabling time of servo motor
int Fgo = 8; // go forward
int Rgo = 6; // go right
int Lgo = 4; // go left
int Bgo = 2; // go backward
//*********************************************************************(SETUP)
void setup()
{
  Serial.begin(9600);
  pinMode(MotorRight1, OUTPUT);    // pin 8 (PWM)
  pinMode(MotorRight2, OUTPUT);    // pin 9 (PWM)
  pinMode(MotorLeft1,    OUTPUT);    // pin 10 (PWM)
  pinMode(MotorLeft2,    OUTPUT);    // pin 11 (PWM)
  irrecv.enableIRIn();          // turn on IR decoding
```

```
    pinMode(SensorLeft, INPUT); // define left sensor
  pinMode(SensorMiddle, INPUT);// define middle sensor
  pinMode(SensorRight, INPUT); // define right sensor
  digitalWrite(2,HIGH);
  pinMode(inputPin, INPUT); // define input pin for ultrasonic
  pinMode(outputPin, OUTPUT); // define output pin for ultrasonic
  myservo.attach(9); // define servo motor output pin as pin5 (PWM)


 }
//**********************************************************************(Void)
void advance(int a) // go forward
{
          digitalWrite(MotorRight1,LOW);
          digitalWrite(MotorRight2,HIGH);
          digitalWrite(MotorLeft1,LOW);
          digitalWrite(MotorLeft2,HIGH);
          delay(a * 100);
}
void right(int b) // turn right (single wheel)
{
          digitalWrite(MotorLeft1,LOW);
          digitalWrite(MotorLeft2,HIGH);
          digitalWrite(MotorRight1,LOW);
          digitalWrite(MotorRight2,LOW);
          delay(b * 100);
}
void left(int c) // turn left (single wheel)
{
        digitalWrite(MotorRight1,LOW);
        digitalWrite(MotorRight2,HIGH);
        digitalWrite(MotorLeft1,LOW);
        digitalWrite(MotorLeft2,LOW);
        delay(c * 100);
}
void turnR(int d) // turn right (two wheels)
{
        digitalWrite(MotorRight1,HIGH);
        digitalWrite(MotorRight2,LOW);
        digitalWrite(MotorLeft1,LOW);
        digitalWrite(MotorLeft2,HIGH);
        delay(d * 100);
}
void turnL(int e) // turn left (two wheels)
```

```
{
        digitalWrite(MotorRight1,LOW);
        digitalWrite(MotorRight2,HIGH);
        digitalWrite(MotorLeft1,HIGH);
        digitalWrite(MotorLeft2,LOW);
        delay(e * 100);
}
void stopp(int f) // stop
{
       digitalWrite(MotorRight1,LOW);
       digitalWrite(MotorRight2,LOW);
       digitalWrite(MotorLeft1,LOW);
       digitalWrite(MotorLeft2,LOW);
       delay(f * 100);
}
void back(int g) // backward
{
           digitalWrite(MotorRight1,HIGH);
           digitalWrite(MotorRight2,LOW);
           digitalWrite(MotorLeft1,HIGH);
           digitalWrite(MotorLeft2,LOW);;
           delay(g * 100);
}
void detection() // measure 3 angles (front, left, right)
{
     int delay_time = 250; // stabling time of servo motor
     ask_pin_F(); // read the upfront distance

     if(Fspeedd < 10) // if distance less than 10CM
    {
        stopp(1); // clear output
        back(2); // backward 0.2S
    }
     if(Fspeedd < 25) // if distance less than 25CM
    {
        stopp(1); // clear output
        ask_pin_L(); // read distance on the left
        delay(delay_time); // wait for servo motor to stabilize
        ask_pin_R(); // read the distance on the right
        delay(delay_time); // wait for servo motor to stabilize

        if(Lspeedd > Rspeedd) // if distance on the left is greater than the distance on the right
       {
           directionn = Lgo; // turn left
```

```
        }

          if(Lspeedd <= Rspeedd) // if distance on the left is less than or equal to the distance on the
right
          {
             directionn = Rgo; // turn right
          }

          if (Lspeedd < 15 && Rspeedd < 15) // if distance both left and right is less than 10CM
         {
             directionn = Bgo; // go backward
          }
       }
       else // if distance upfront is greater than 25CM
     {
         directionn = Fgo; // go forward
     }
}
//*************************************************************************
****
void ask_pin_F() // measure the distance upfront
{
myservo.write(90);
digitalWrite(outputPin, LOW); // ultrasonic transmit low voltage 2μs
delayMicroseconds(2);
digitalWrite(outputPin, HIGH); // ultrasonic transmit high voltage 10μs, here should be at least
10μs
delayMicroseconds(10);
digitalWrite(outputPin, LOW); // maintain low voltage
float Fdistance = pulseIn(inputPin, HIGH); // read time in between
Fdistance= Fdistance/5.8/10; // convert time into distance (unit: cm)
Serial.print("F distance:"); // output distance (unit: cm)
Serial.println(Fdistance); // display distance
Fspeedd = Fdistance; // read distance into Fspeedd (forward speed)
}
//*************************************************************************
***
void ask_pin_L() // measure distance on the left
{
myservo.write(177);
delay(delay_time);
digitalWrite(outputPin, LOW); // ultrasonic transmit low voltage 2μs
delayMicroseconds(2);
digitalWrite(outputPin, HIGH); // ultrasonic transmit high voltage 10μs, here should be at least
```

```
10μs
delayMicroseconds(10);
digitalWrite(outputPin, LOW); // maintain low voltage
float Ldistance = pulseIn(inputPin, HIGH); //    read time in between
Ldistance= Ldistance/5.8/10; // convert time into distance (unit: cm)
Serial.print("L distance:"); // output distance (unit: cm)
Serial.println(Ldistance); // display distance
Lspeedd = Ldistance; // read distance into Lspeedd (left speed)
}
//*****************************************************************************
*
void ask_pin_R() // measure distance on the right
{
myservo.write(5);
delay(delay_time);
digitalWrite(outputPin, LOW); // ultrasonic transmit low voltage 2μs
delayMicroseconds(2);
digitalWrite(outputPin, HIGH); // ultrasonic transmit high voltage 10μs, here should be at least
10μs
delayMicroseconds(10);
digitalWrite(outputPin, LOW); // maintain low voltage
float Rdistance = pulseIn(inputPin, HIGH); // read time in between
Rdistance= Rdistance/5.8/10; // convert time into distance (unit: cm)
Serial.print("R distance:"); // output distance (unit: cm)
Serial.println(Rdistance); // display distance
Rspeedd = Rdistance; // read distance into Rspeedd (right speed)
}
//*****************************************************************************
*(LOOP)
void loop()
{
        SL = digitalRead(SensorLeft);
        SM = digitalRead(SensorMiddle);
        SR = digitalRead(SensorRight);
//*****************************************************************************nor
mal remote control mode
   if (irrecv.decode(&results))
      {             // decoding succeeded, acquire a set of IR signal
      /*************************************************************************/
        if (results.value == IRfront)// go forward
          {
           advance(10);// go forward
          }
/*************************************************************************/
```

```
        if (results.value ==    IRback)// backward
          {
           back(10);// backward
          }
/*********************************************************************/
        if (results.value == IRturnright)// turn right
        {
           right(6); // turn right
        }
/*********************************************************************/
       if (results.value == IRturnleft)// turn left
       {
          left(6); // turn left);
       }
/*********************************************************************/
      if (results.value == IRstop)// stop
    {
       digitalWrite(MotorRight1,LOW);
       digitalWrite(MotorRight2,LOW);
       digitalWrite(MotorLeft1,LOW);
       digitalWrite(MotorLeft2,LOW);
     }
//*********************************************************************cny70
mode black:LOW white: HIGH
      if (results.value == IRcny70)
    {
       while(IRcny70)
       {
          SL = digitalRead(SensorLeft);
          SM = digitalRead(SensorMiddle);
          SR = digitalRead(SensorRight);

          if (SM == HIGH)// middle sensor in black area
          {
              if (SL == LOW & SR == HIGH) // left sensor black, right sensor white, turn left
              {
                  digitalWrite(MotorRight1,LOW);
                  digitalWrite(MotorRight2,HIGH);
                  analogWrite(MotorLeft1,0);
                  analogWrite(MotorLeft2,80);
              }
              else if (SR == LOW & SL == HIGH) // left sensor white, right sensor black, turn
right
              {
```

```
                analogWrite(MotorRight1,0);// turn right
                analogWrite(MotorRight2,80);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,HIGH);
            }
            else    // both sides are white, go forward
            {
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,HIGH);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,HIGH);
                analogWrite(MotorLeft1,200);
                analogWrite(MotorLeft2,200);
                analogWrite(MotorRight1,200);
                analogWrite(MotorRight2,200);
            }
        }
        else // middle sensor in white area
       {
            if (SL == LOW & SR == HIGH)// left sensor black, right sensor white, turn left
{
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,HIGH);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,LOW);
            }
            else if (SR == LOW & SL == HIGH) // left sensor white, right sensor black, turn right
            {
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,LOW);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,HIGH);
            }
            else // both sides are white, stop
            {
            digitalWrite(MotorRight1,HIGH);
            digitalWrite(MotorRight2,LOW);
            digitalWrite(MotorLeft1,HIGH);
            digitalWrite(MotorLeft2,LOW);;
            }
        }
        if (irrecv.decode(&results))
        {
                irrecv.resume();
```

```
                    Serial.println(results.value,HEX);
                if(results.value ==IRstop)
                {
                   digitalWrite(MotorRight1,HIGH);
                   digitalWrite(MotorRight2,HIGH);
                   digitalWrite(MotorLeft1,HIGH);
                   digitalWrite(MotorLeft2,HIGH);
                   break;
                }
          }
       }
        results.value=0;
    }
//*********************************************************************
ultrasonic mode
 if (results.value ==IRAutorun )
        {
              while(IRAutorun)
           {
               myservo.write(90); // homeset servo motor, ready the position for next measurement
               detection(); // measure the angle and decide which way to move
                 if(directionn == 8) // when directionn= 8(forward)
               {
                  if (irrecv.decode(&results))
              {
                irrecv.resume();
                Serial.println(results.value,HEX);
                if(results.value ==IRstop)
                {
                   digitalWrite(MotorRight1,LOW);
                   digitalWrite(MotorRight2,LOW);
                   digitalWrite(MotorLeft1,LOW);
                   digitalWrite(MotorLeft2,LOW);
                   break;
                }
              }
                    results.value=0;
                    advance(1); // forward
                    Serial.print(" Advance "); // display direction(forward)
                    Serial.print(" ");
                 }
               if(directionn == 2) // when directionn= 2(backward)
             {
```

```
  if (irrecv.decode(&results))
 {
   irrecv.resume();
   Serial.println(results.value,HEX);
   if(results.value ==IRstop)
   {
     digitalWrite(MotorRight1,LOW);
     digitalWrite(MotorRight2,LOW);
     digitalWrite(MotorLeft1,LOW);
     digitalWrite(MotorLeft2,LOW);
     break;
   }
 }
    results.value=0;
    back(8); // backward
    turnL(3); // slightly turn left (prevent stuck in corner)
    Serial.print(" Reverse "); // display direction (backward)
}
  if(directionn == 6) // when directionn = 6(turn right)
{
 if (irrecv.decode(&results))
 {
    irrecv.resume();
    Serial.println(results.value,HEX);
   if(results.value ==IRstop)
   {
     digitalWrite(MotorRight1,LOW);
     digitalWrite(MotorRight2,LOW);
     digitalWrite(MotorLeft1,LOW);
     digitalWrite(MotorLeft2,LOW);
     break;
   }
 }
   results.value=0;
     back(1);
     turnR(6); // turn right
     Serial.print(" Right "); // display direction (right)
}
  if(directionn == 4) // when directionn= 4(turn right)
{
   if (irrecv.decode(&results))
 {
   irrecv.resume();
   Serial.println(results.value,HEX);
```

```
                if(results.value ==IRstop)
                {
                  digitalWrite(MotorRight1,LOW);
                  digitalWrite(MotorRight2,LOW);
                  digitalWrite(MotorLeft1,LOW);
                  digitalWrite(MotorLeft2,LOW);
                  break;
                }
              }
                  results.value=0;
                  back(1);
                  turnL(6); // turn left
                  Serial.print(" Left "); // display direction(left)
              }

                if (irrecv.decode(&results))
              {
                irrecv.resume();
                Serial.println(results.value,HEX);
                if(results.value ==IRstop)
                {
                  digitalWrite(MotorRight1,LOW);
                  digitalWrite(MotorRight2,LOW);
                  digitalWrite(MotorLeft1,LOW);
                  digitalWrite(MotorLeft2,LOW);
                  break;
                }
              }
            }
                  results.value=0;
          }
/*********************************************************************/
       else
      {
              digitalWrite(MotorRight1,LOW);
              digitalWrite(MotorRight2,LOW);
              digitalWrite(MotorLeft1,LOW);
              digitalWrite(MotorLeft2,LOW);
       }


          irrecv.resume();       // receive next set of IR signal
    }
}
```

Trademark statement:

Robotale and the logo are the registered trademark of Shenzhen KEYES DIY Robot Co., Ltd. For continuous improvement and upgrading of our products, Our company may change information or products mentioned in this manual without further notice. Without our written consent or authorization, anyone can not use, copy or publish parts or all the content in this product manual.

Disclaimer:

In Any application (such as experiment, competition, secondary development)that user do in using this product, user himself/herself shall take the risk. Our company is not liable for any direct, indirect or collateral damage (including personal safety damage, profit or credit losses etc.) during the use of our products. Children under the age of 14 must be accompanied by an adult to use the product to do related experiment.

Erratum statement:

In order to correctly convey the product information, we spend a great deal of time and effort in this manual. We hope users will read the contents carefully. And if there are omissions, you are more than welcome to contact us by email jmddz925@126.com.. For the improvement of the manual and to keep it up-to-date, we will continuously revise the content of the manual.