Aurora-Racing
Instruction Manual

Get the latest update from https://github.com/keywish/Aurora-Racing

Chang history

Date	Version	Description	Author
2018-01-20	**V.1.0**	**Create**	**Baron.li**
2018-5-10	**V.1.1**	**Modifly**	**Ken.chen**
2018-6-1	**V.1.2**	**Translation Modify**	**Richard.zhou**
2018-6-1	**V.1.3**	**Translation Modify**	**Ruby.zhou**

CONTENTS

Chapter 1 PREFACE

1.1 Purpose

Our purpose is to offer a learning platform for DIY lovers, makers and beginners, help to get a better understanding of Arduino, and its expansion system design methods and principles, as well as the corresponding hardware debugging methods. Further deepen the understanding of the design and application of Arduino and its extended system.

The instruction manual mainly introduces the installation, hardware and software for Aurora-Racing from the easy to the difficult and complicated. There are two parts for the instruction manual.
The first part mainly introduces how to use the common developing software and the method of downloading,debugging. And the second part mainly introduces about hardware and software. For the hardware part, it mainly introduces the functions, principles of every module, and for the software part, there are many examples for customers, it mainly introduces the applications of the Aurora.

There are lots of detailed schematic diagrams and example codes for each module, which is strictly tested to ensure the accuracy and precision. Moreover, you can easily find the library files in the corresponding file folder and download through the UART/simulator to the Arduino uno R3 board for the corresponding functions . You can debug each module according to the course or directly assemble the car to enjoy the fun of a maker.

1.2 Product Introduction

Aurora-Racing, as a 4-wheel multi-function smart racing car, we chosen Arduino uno r3 as its control board. It's different from the other 4-wheel's arduino cars, it is a rear-wheel-driving and front-steering. It shares the same principle with professional remote racing car, which the direction can be controlled by the two pull rods powered by the steering engine,the structure of the back wheel is a whole Aluminium body; We adopt all metal gear motor,two Flanged bearings are used in back wheels to increase the flexibility of the motor.

Chosen ATMEGA328P-PU as its main controller and TB6612FGN as its motor drive chip, Aurora-Racing is a multi-function smart racing car, compared to the traditional race, Aurora-Racing is equipped wireless control (Bluetooth, infrared remote control,etc.) and infrared trace, which can trace automatically. Of course, you can also add or cut off some functions if you like, for example, obstacle evading, adaptation for the PSP gamepad, WIFI controlling or mechanical arm.

Offer enough learning material,technical manual, detailed examples. Hand in hand teaches you from the beginning to the mastery. You can easily get started and achieve the functions you want.

Product features

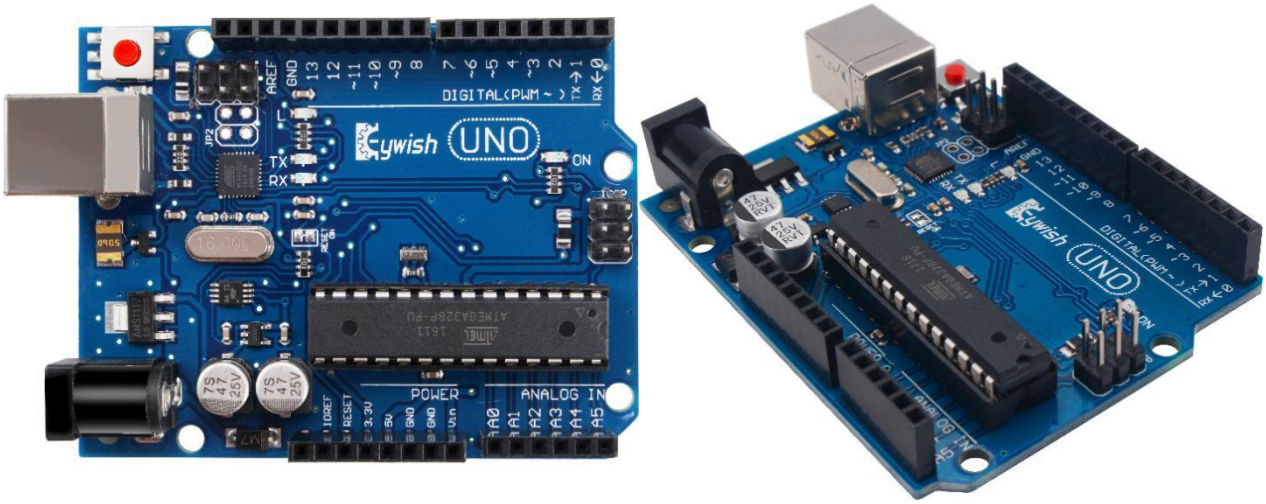
- 5 road black line tracing module
- High powerfull metal speed-reduction motor
- Flange cup type bearing
- Integral stamping molding kit,easier Installation,tighter
- 3000mAH,12v,rechargeable li-battery,longer battery life,and more dynamic
- 2 RGB turn lights
- Buzzer Turn around reminder
- Infrared remote control
- Android App control

1.3 Product package list:

 1 x Keywish UNO R3 Controller Board	 1 x Motor driver shield	 12V2000MA lithium battery	 Li-battery charger	 1 x Infrared remote	 1 x USB Cable	 4 x 65mm Blue Wheel
 1 x Aluminum alloy plate	 1 x Acrylic button plate	 1x Acrylic top plate	 1 x MG996R Servo Motor	 1 x Rudder angle 2x M2.5*12mm screw 1x M3*5 screw	 2 x Ball joints drag link	 2 x L-type holder
 2 x Front-wheel steering cup	 2 x Hexagonal adapter	 2 x M2*10 plug	 2 x M3 Flat-head screw	 2 x Front-wheel steering cup articulated shaft	 DC Motor/PH2.0mm x20cm Wire	 1xDC Motor holder
 2 x M4 Locknut	 2 x M4 Locknut	 2x8*12 bearing 2x4*8 bearing	 2x Motor bearing	 1x Motor drive linkage	 2 x Hexagonal adapter	 2 Pack Gear
 13 x M3 nut	 4x M3*10MM single copper pillar	 4x M3*10MM Double copper pillar	 8 x M3*22MM Double copper pillar	 4 x M3*35 single Copper pillar	 2x M3*12 screw	 2 x M3*5 Flat-head screw
 4 x M3*10 Flat-head screw	 45 x M3*8 screw	 2 x M2.5*22 screw	 1x M3*16screw	 4xwasher	5-line infrared tracking module	30cm Male to Female DuPont Line (7pin)
1x Cross sleeve	1 x 5*100screwdriver	5.5/2.1mm Female to Female DC Conversion Head Adapter	1x Allen wrench	Several cable ties	anticollision cotton	

2.1 About Arduino uno R3

In Aurora-Racing,we use Arduino uno r3 as main control board. Arduino uno r3 features 14 digitalinput/output (6 of them can be used as PWM output), six analog input inputs,one 16 MHz ceramic resonator,one USB connecter,one power adapter, one ICSP,one reset button.



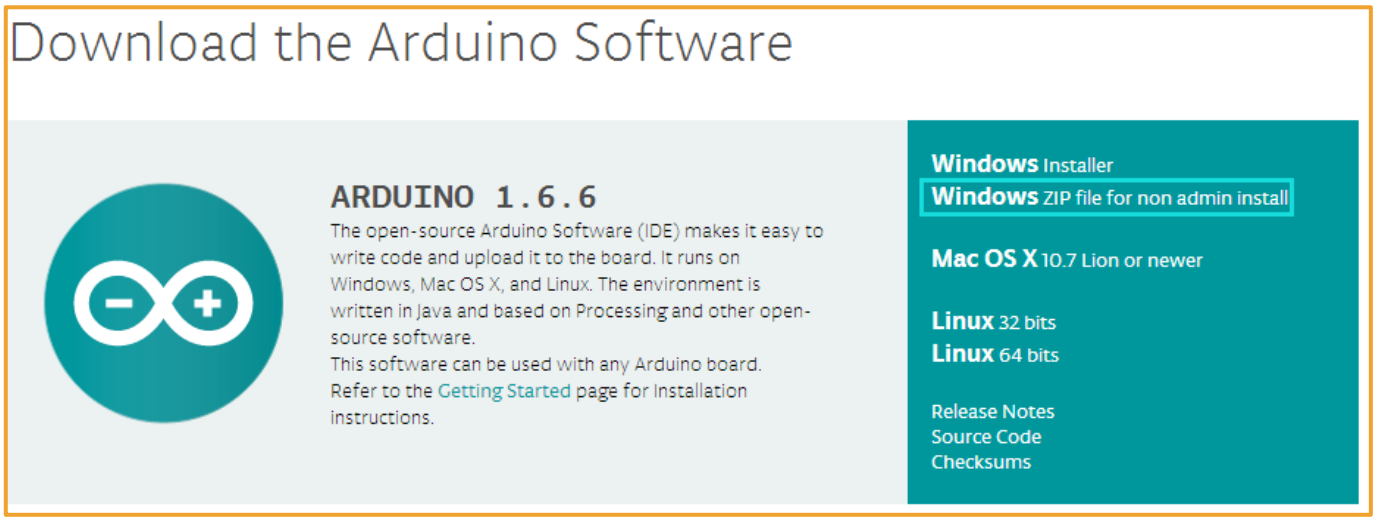
Specifications

- Working voltage: 5V
- Input voltage: 7V~12V DC or USB Power;
- Output voltage: 5V DC output, 3.3V DC output, extern power
- Microcontroller: ATmega328
- Bootloader：Arduino Uno
- Clock rate：16 MHz
- Support USB port agreement and USB charging(no other battery needed)
- Support ISP download
- Digital I/0 port: 14(4 PWM outputs)
- Analog input port: 6
- DC current I/0 port: 40 mA
- DC current 3.3V port: 50mA
- Flash momory: 32KB (ATmega328)(0.5 KB for boot program)
- SRAM：2 KB (ATmega328)
- EEPROM：1 KB (ATmega328)
- Dimensions：75*55*15mm

2.2 Development Software

Download link: https://www.arduino.cc/en/Main/Software

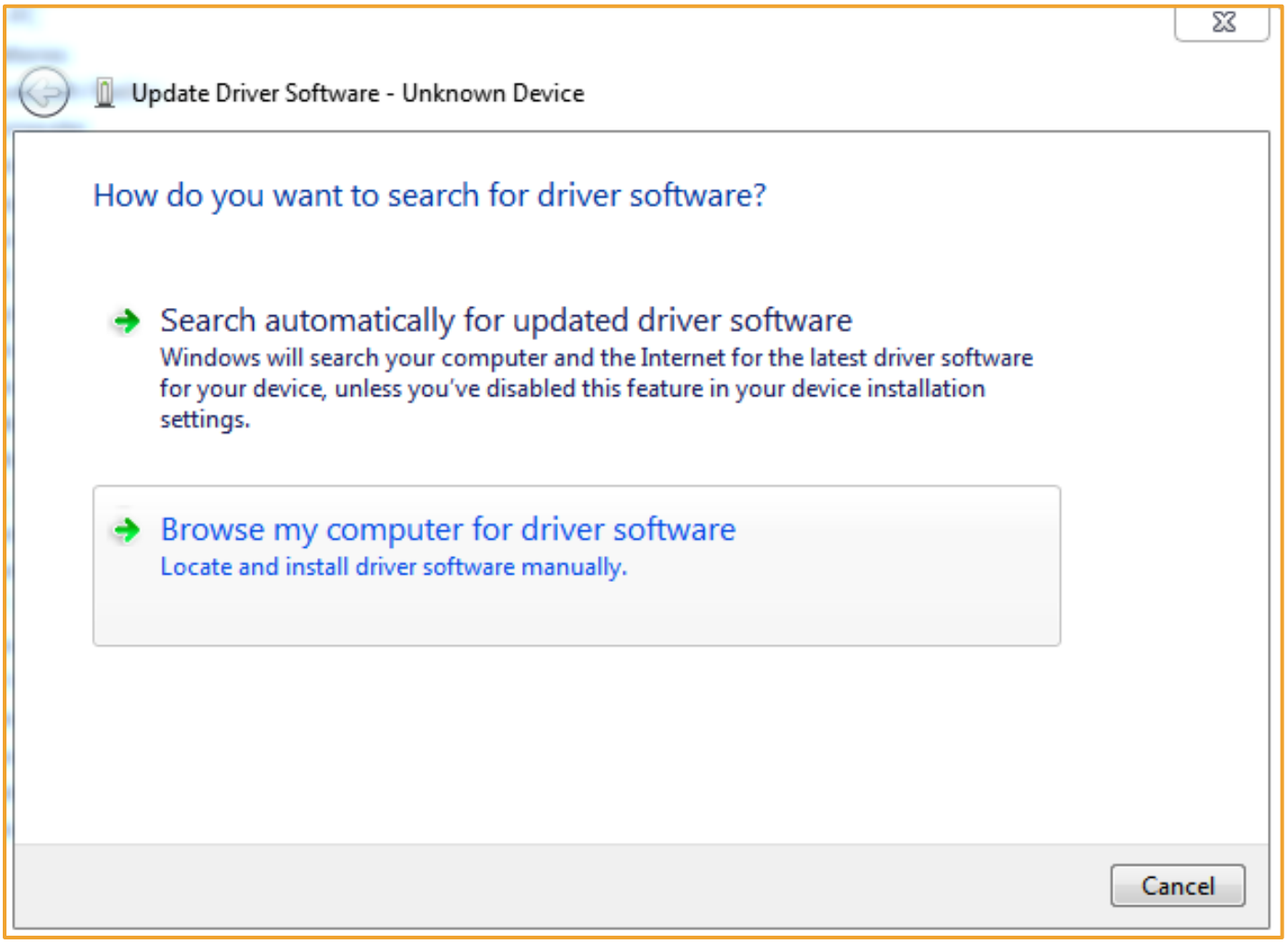
Windows, Linux, Mac are all available for downloading.



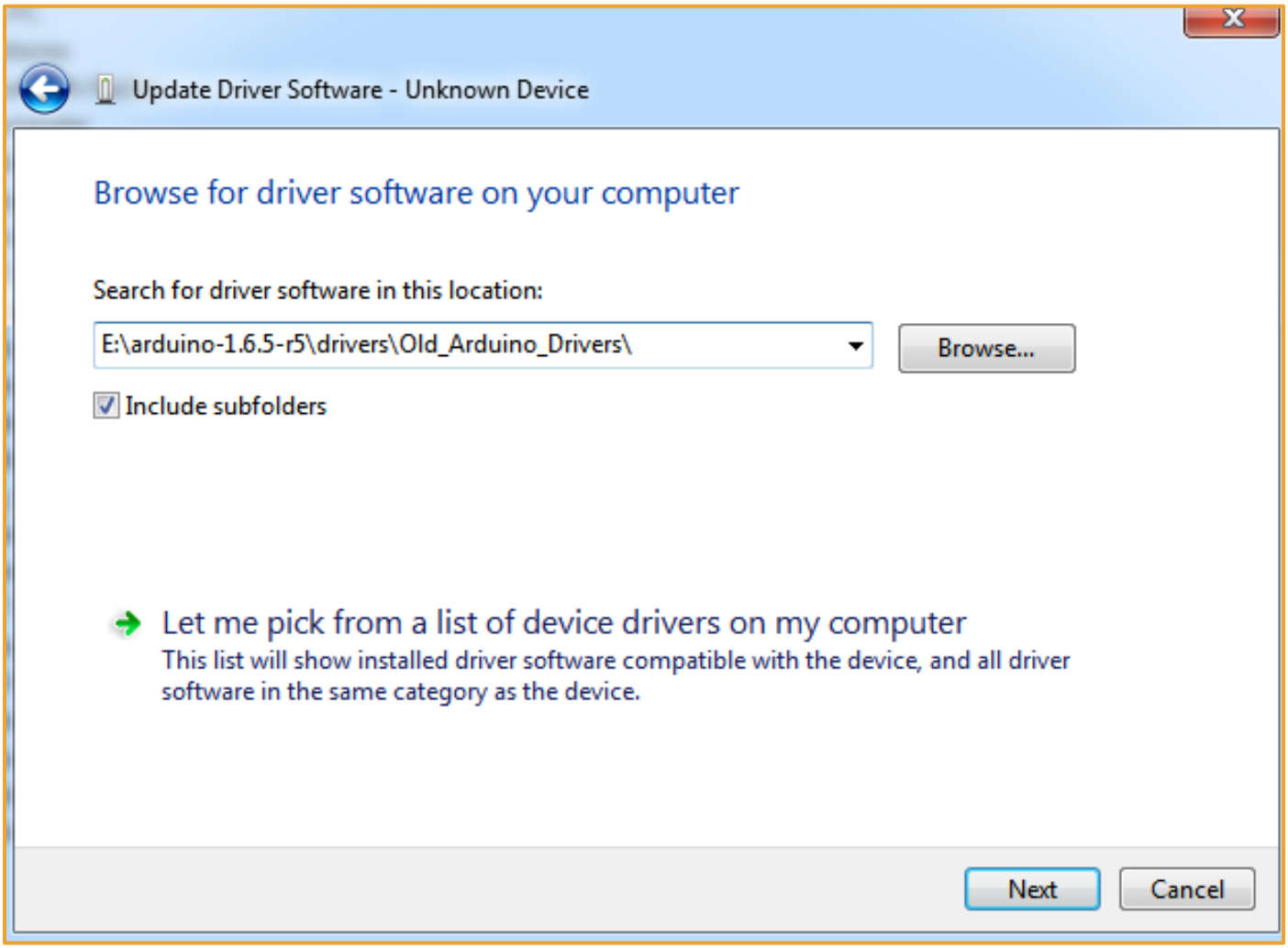
The interface of Arduino IDE is simple and the operation is rather convenient. If you want get more,please click https://www.arduino.cc/en/Guide/Environment

Install usb to serial port driver

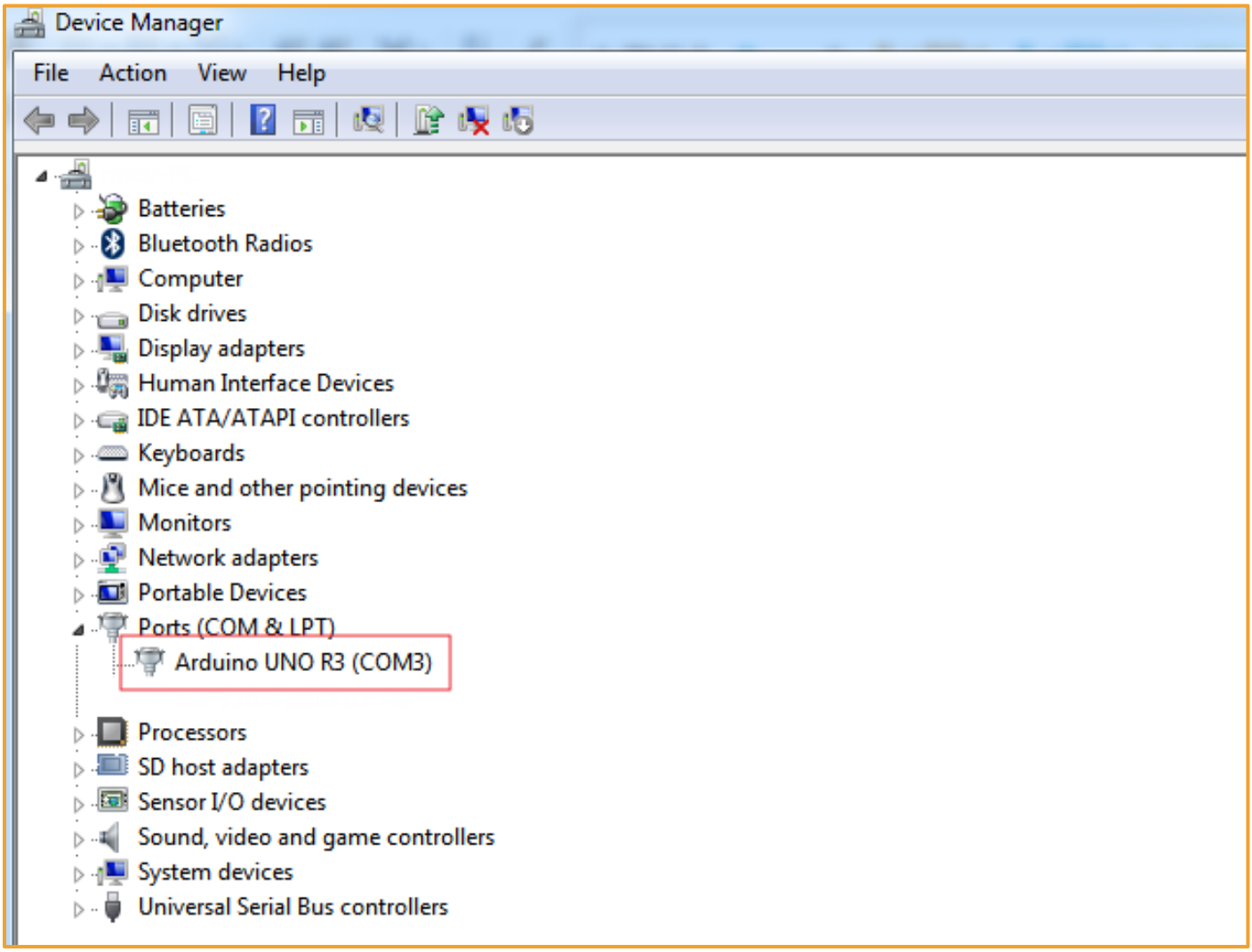
Inserting USB cable will prompt as follows, choosing the specified location to install.



Selecting download Arduino ide file “E:\arduino-1.6.5-r5\drivers\Old_Arduino_Drivers\” Checking the type of USB serial chip on the board, if it is Atmel, then choose the following path; if it is FTDI, you should choose the arduino\drivers\FTDI USB Drivers path.



Clicking the next step, you will be prompted with a successful installation message. Now you can change to equipment management to see Arduino UNO R3.

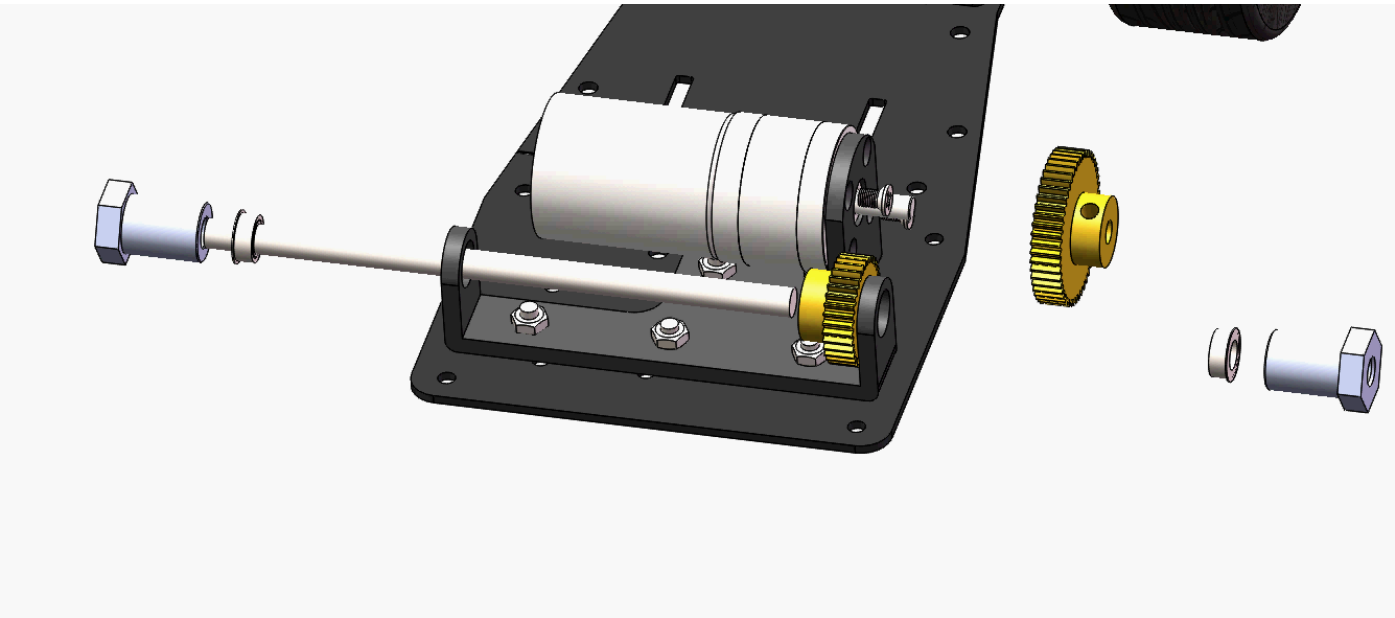

Device Manager
File
Action
View
Help
Batteries
Bluetooth Radios
Computer
Disk drives
Display adapters
Human Interface Devices
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Portable Devices
Ports (COM & LPT)
Arduino UNO R3 (COM3)
Processors
SD host adapters
Sensor I/O devices
Sound, video and game controllers
System devices
Universal Serial Bus controllers

3.1 Aurora-Racing installation

Firslty open the box and take out all parts and put them on the desk carefully. (Please carefully to avoid lose anything)

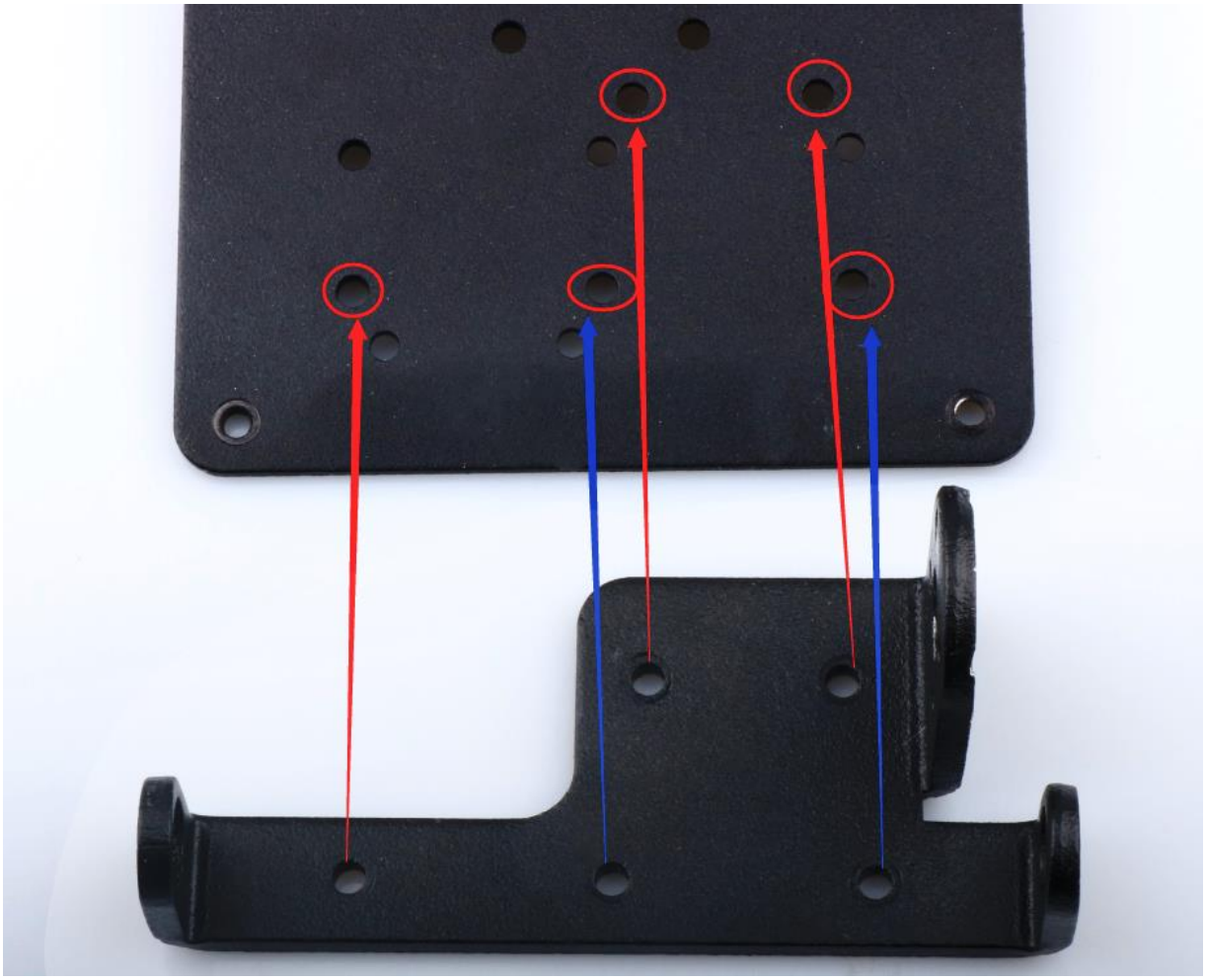
3.1.1 Rear wheel and motor installation

(Installation video tutorial\1.Rear Drive Assembly.avi)



Find the following parts: Aurora-Racing's baseplate,Wheels X2, motor, cables, motor stand, one big gear, one small gear, axle, coupling, flange bearing, two M3 flatheadscrews, 5 M3 nuts,7 M3*8 screws,2 M4*6 screws, set screws and screwdriver.

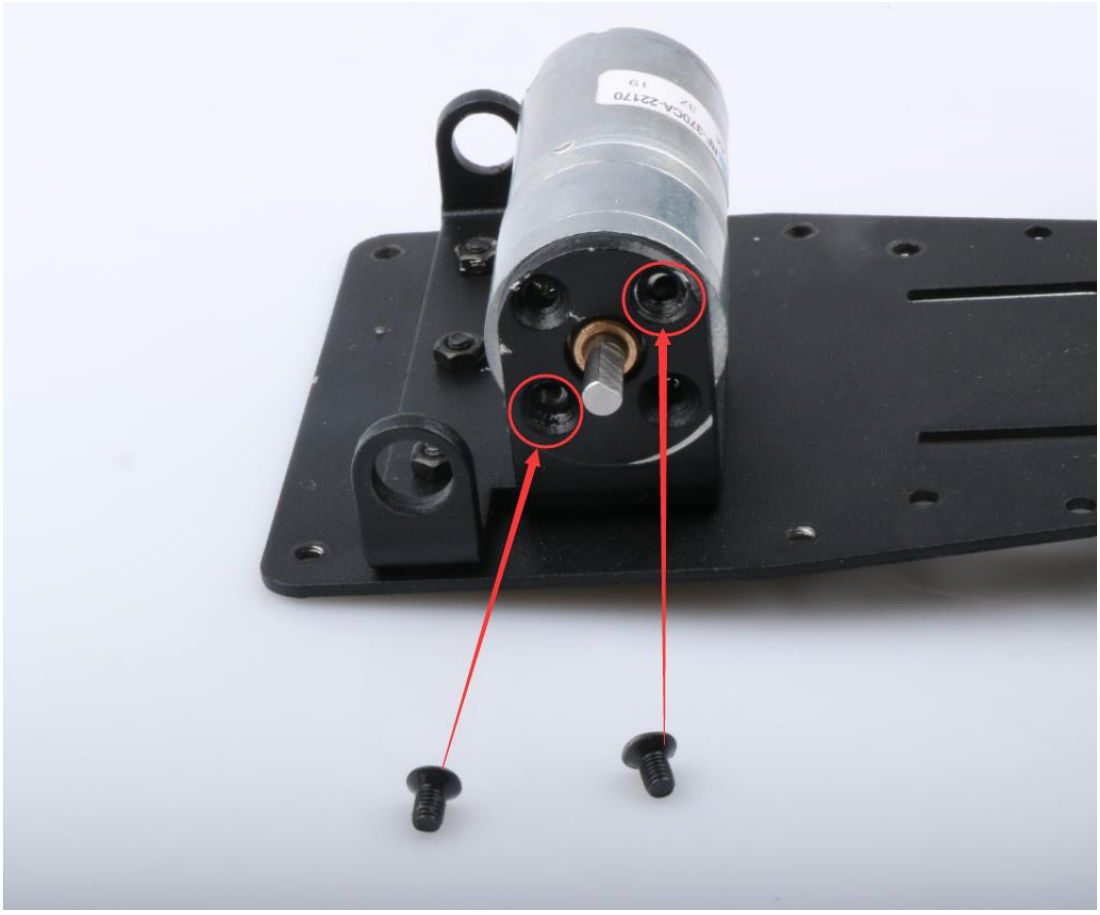
Step 1: Put the motor stand on the baseboard and fix it with M3*8 screws. As picture 3.1.2



Picture 3.1.1 Motor stand installation schematic diagram

Picture 3.1.2 Motor stand installation effect diagram

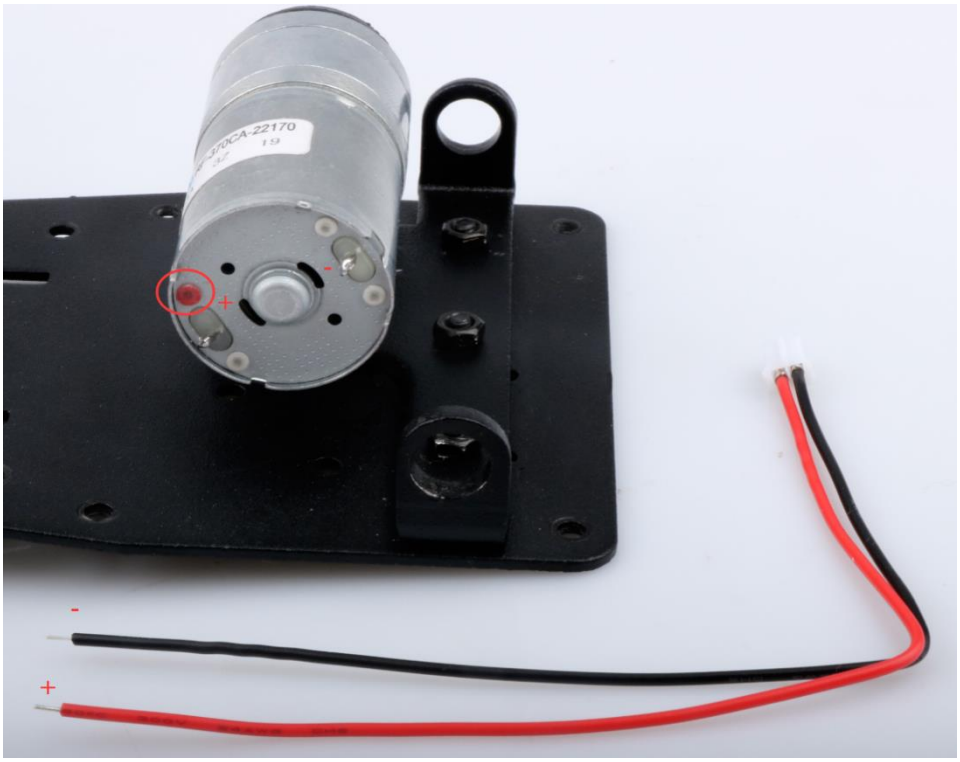
Step 2: Fix the motor on the stand according to the picture 3.1.3, and then tightening with M3 flathead screws as picture 3.1.4



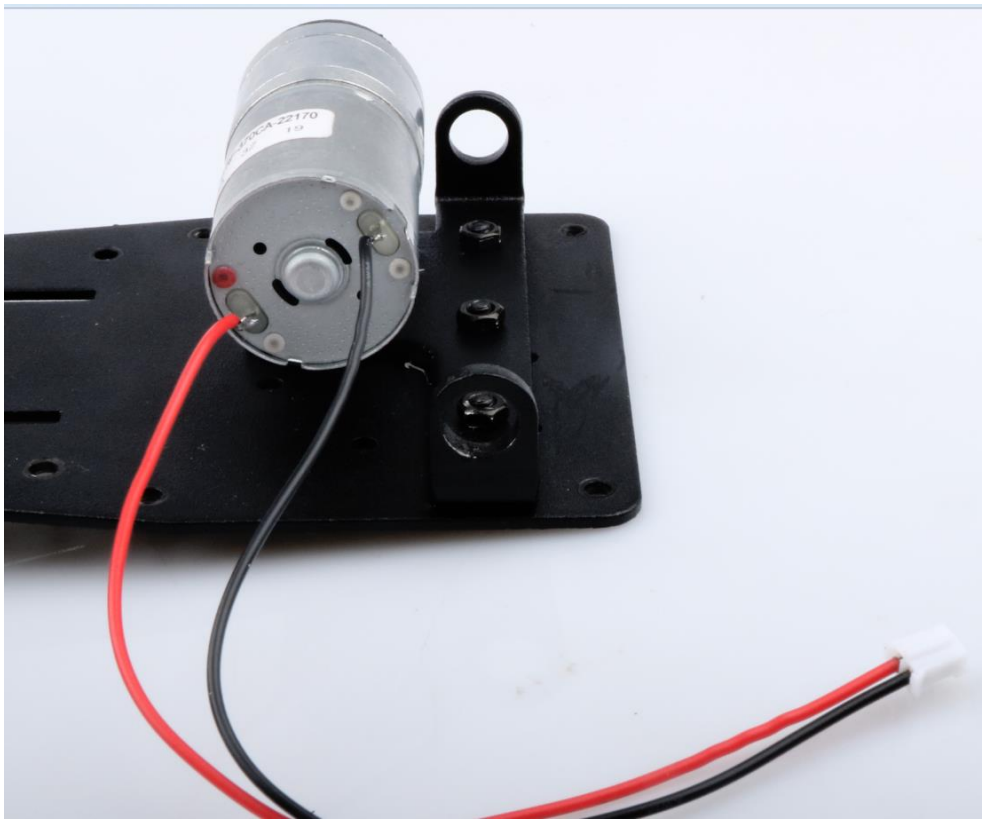
Picture 3.1.3 Motor installation schematic diagram

Picture 3.1.4 Motor installation effect diagram

Step3 :Weld the wire to the motor as picture 3.1.5. For a unified standard, the red dot on the motor is the positive pole and the other dot is the negative pole. Please check the picture 3.1.6 for more details.

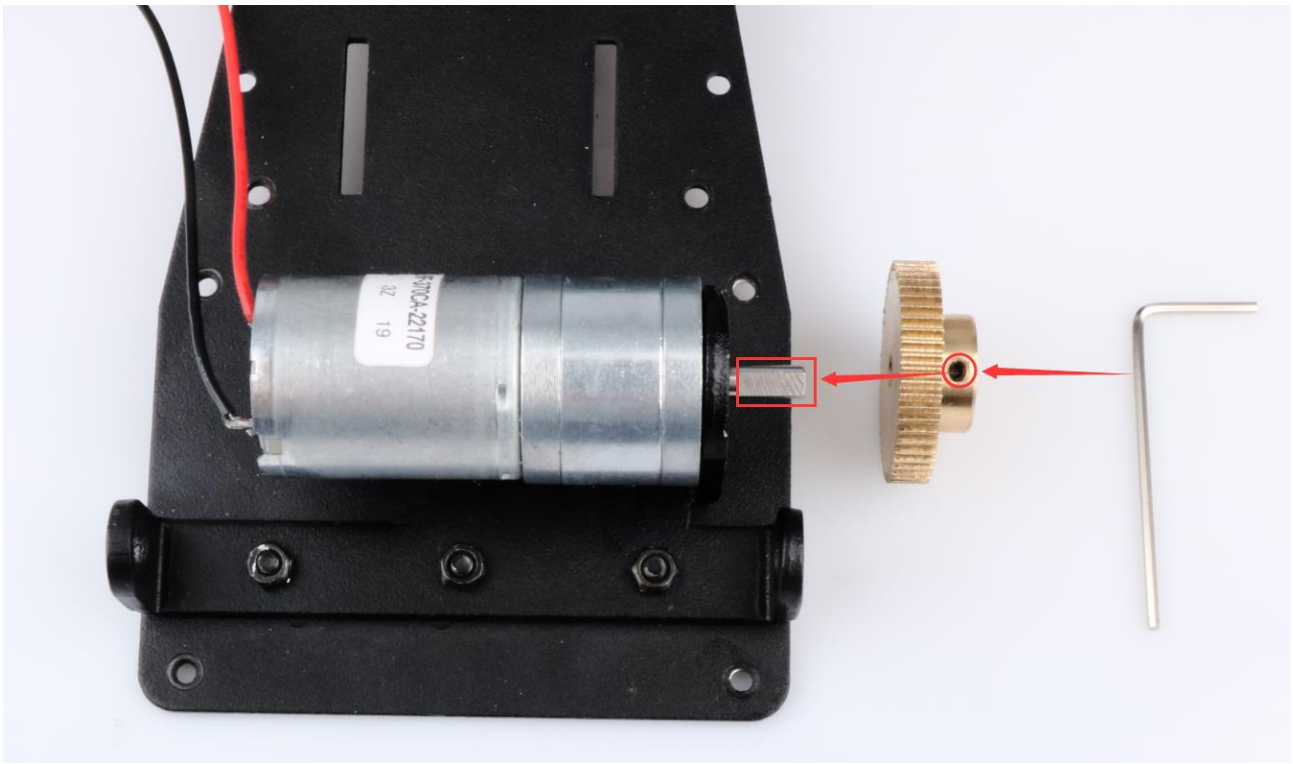


Picture 3.1.5 Motor wire welding schematic



Picture 3.1.6 Electrical wire welding effect diagram

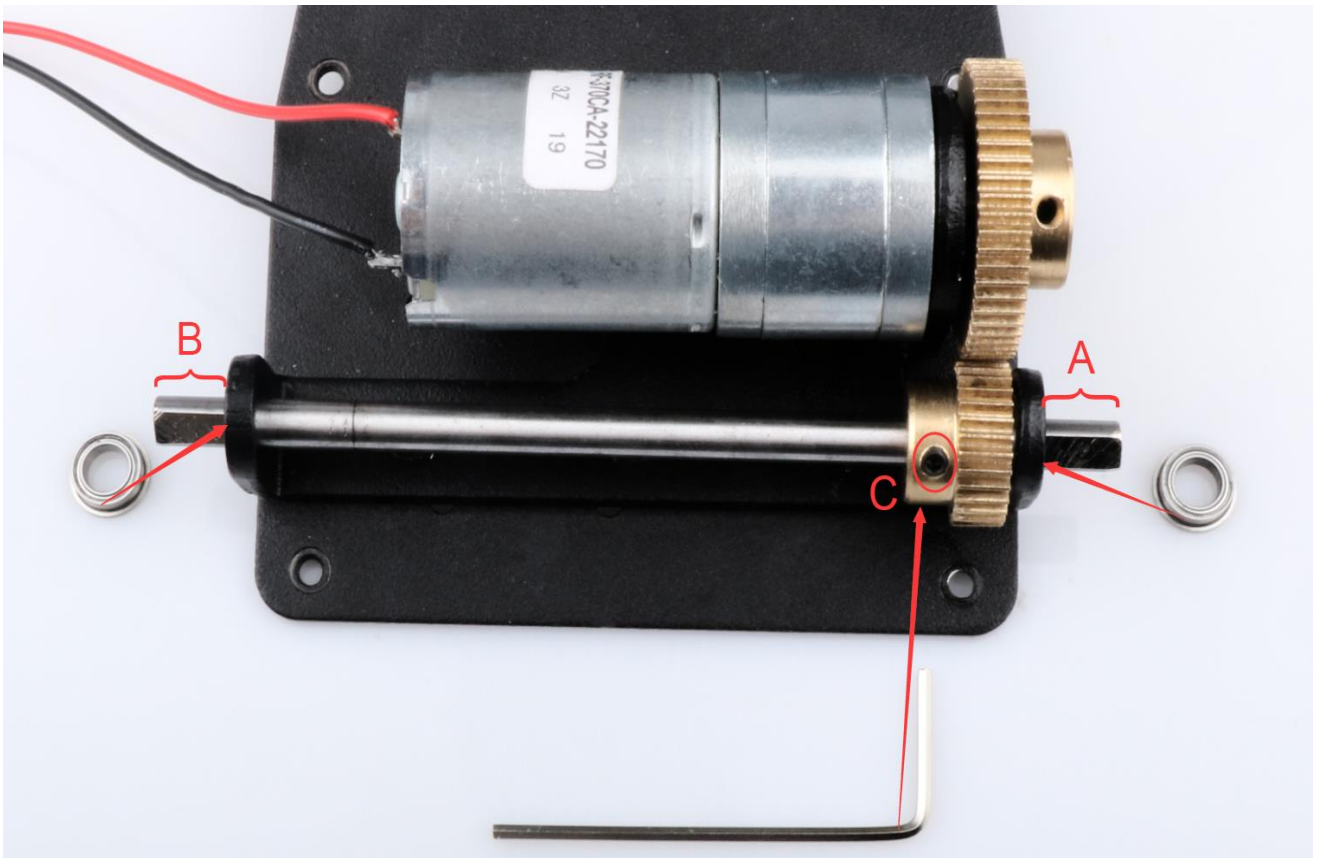
Step 4: Install motor gear(the big one). Install the motor gear on the motor's shaft. Make sure that hole of the set screw point to the shaft' side which was polished according to the picture 3.1.7, then tight the set screw,then you can see the effect as picture 3.1.8.



Picture 3.1.7 Motor gear installation

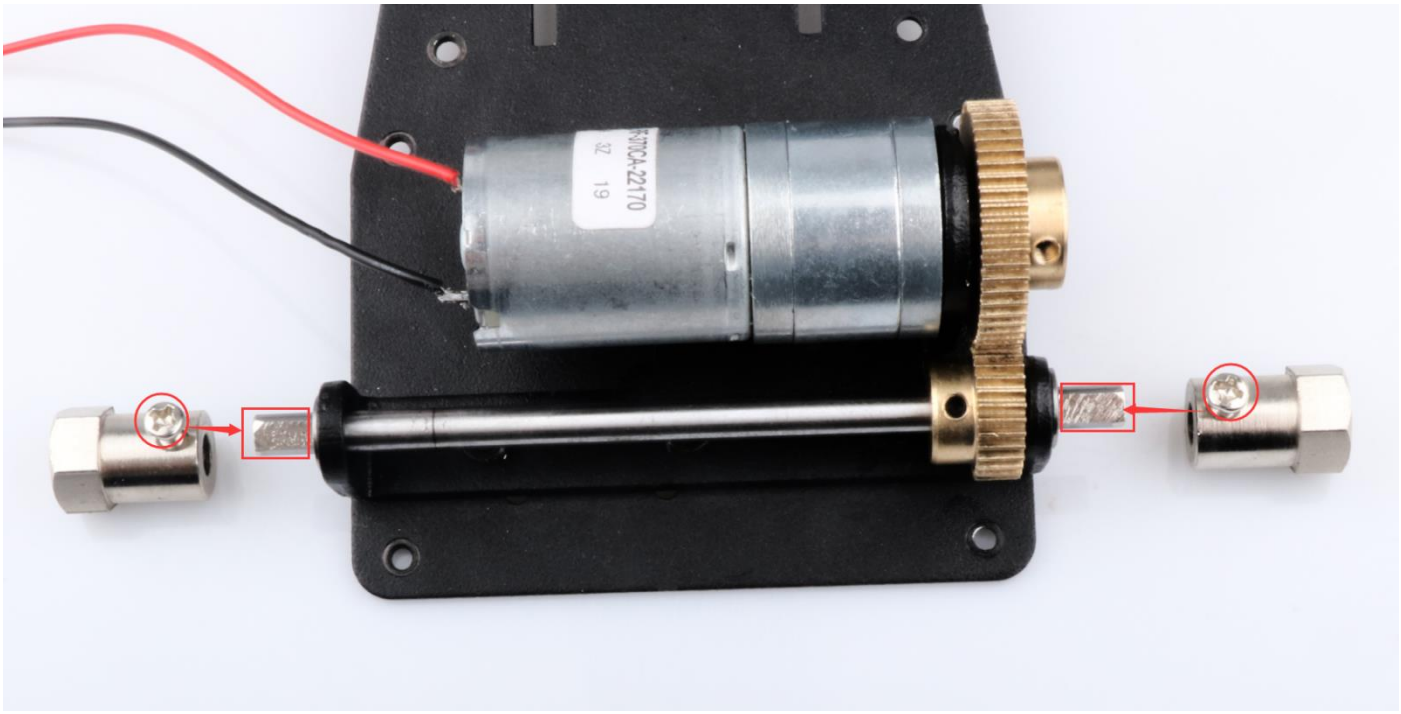
Picture 3.1.8 Motor gear installation effect

Step5 : Install axle, the small gear and the flange bearing according to the picture 3.1.9. Please note: Make sure the big gear and the small gear can be well interlocked. Distance A and Distance B should be approximately equal. Tight the screw of the part C.

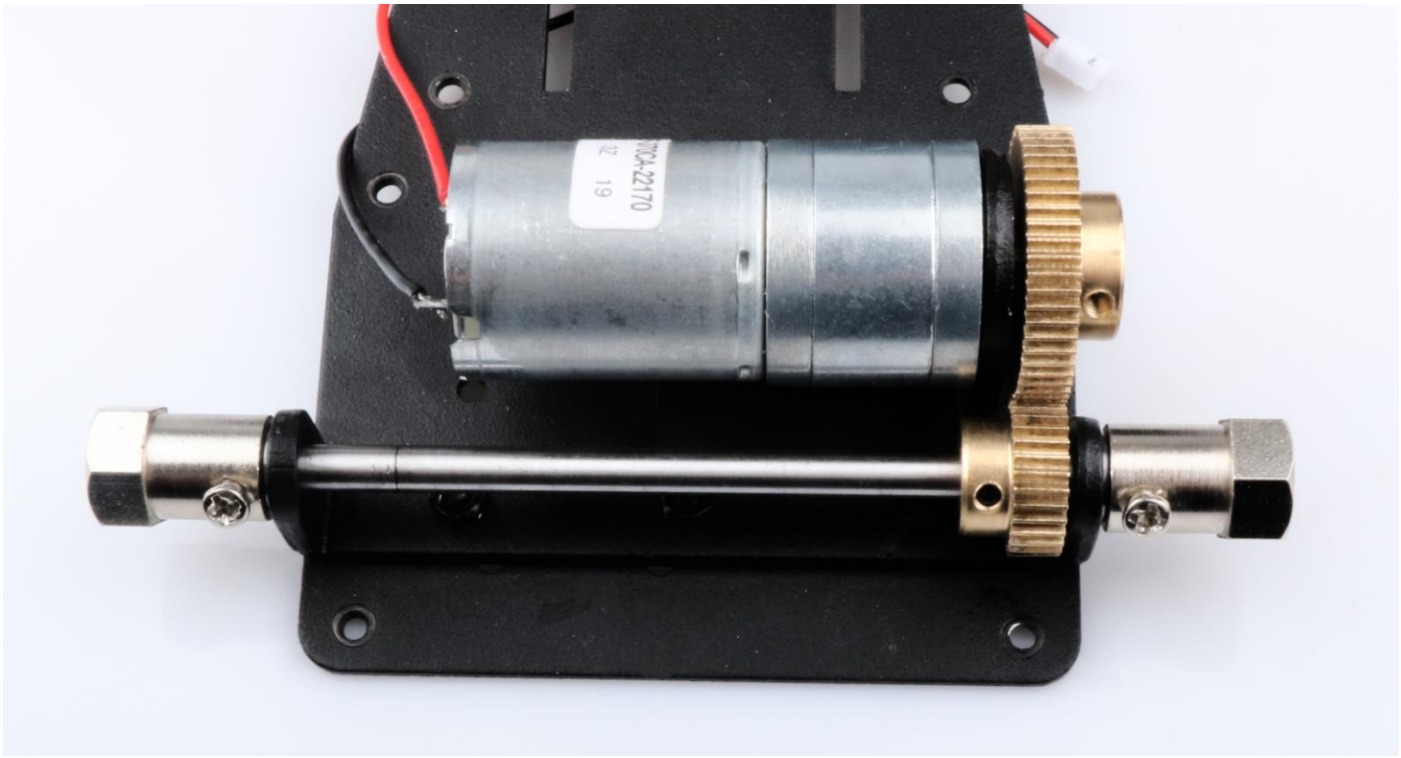


Picture 3.1.9 Axle and pinion installation

Step6: Install connectingshaft. As the power transmission part, connectingshaft connects axle and wheels. **The installation method is insert the smooth surface of the axle to the connectingshaft correctly like the picture 3.1.10, and then tight the screw. Please check the image 3.1.11 for more details.**

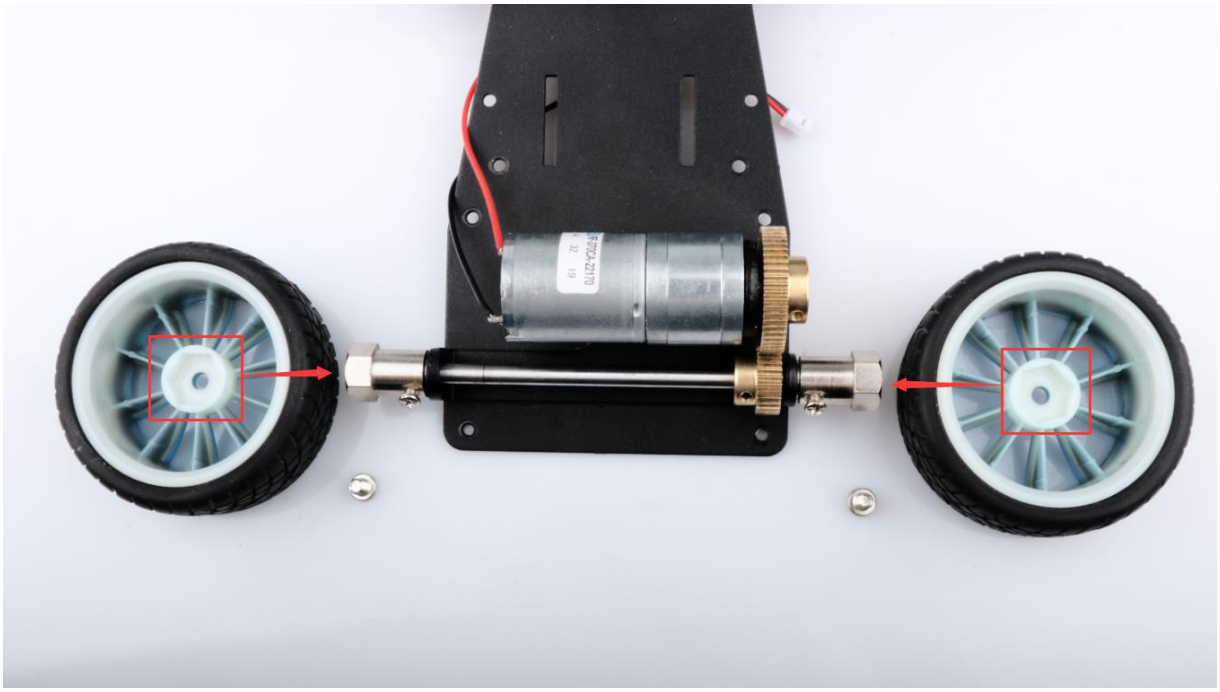


Picture 3.1.10 Connectingshaft installation

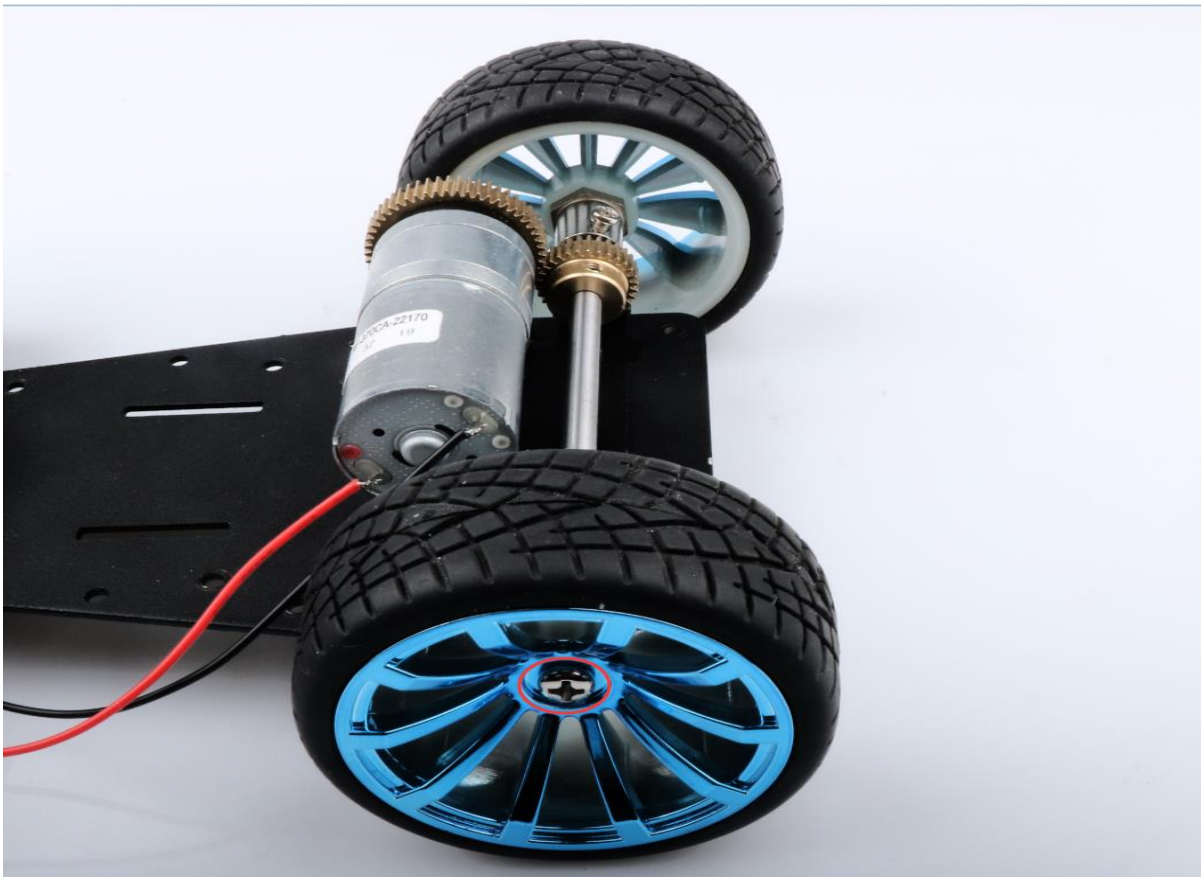


Picture 3.1.11 Connectingshaft installation effect

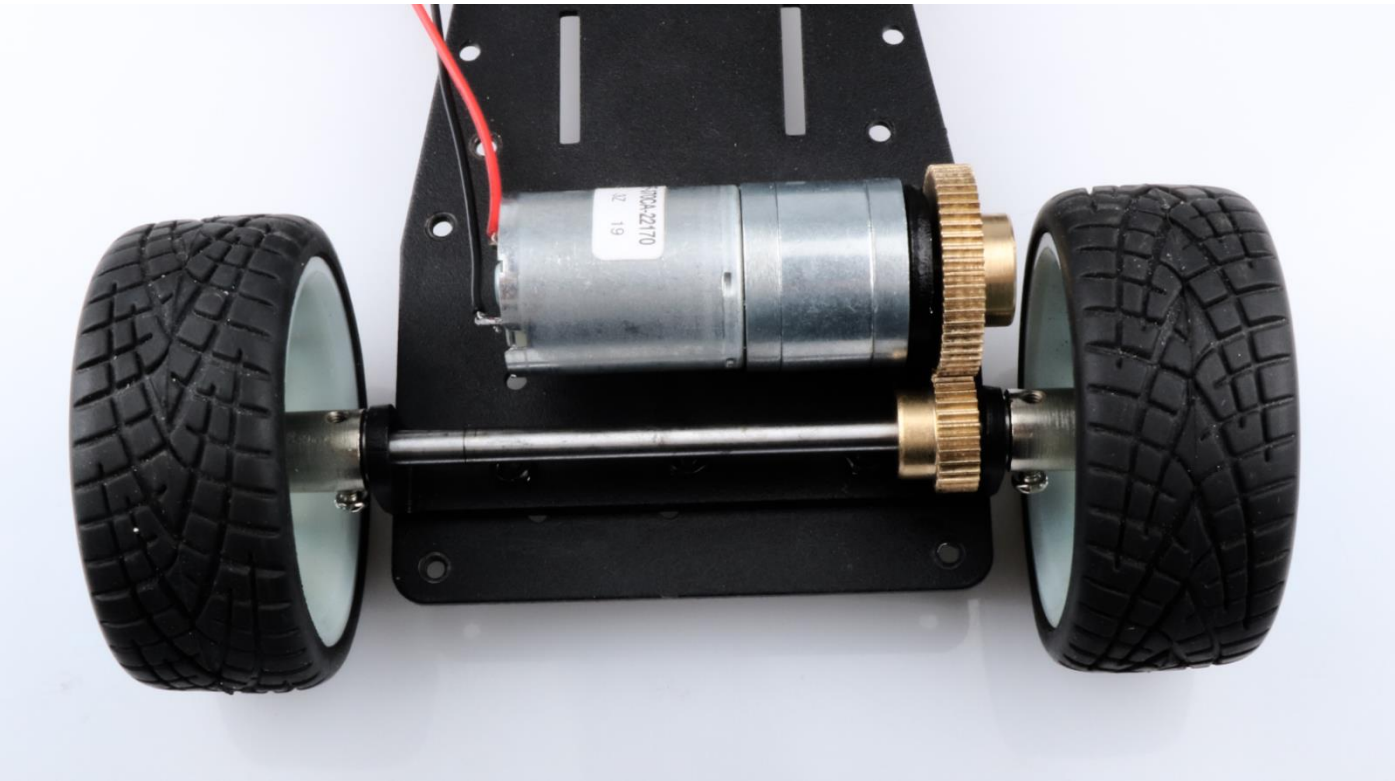
Step 7: Wheel installation. Aurora-Racing uses professional racing wheels, compared to the traditional wheels, it has stronger grip force, less friction, more stability. The installation of the wheels is quite simple. Do as the same in the picture 3.1.13, tight two M4X6 screws on the side of the wheels.



Picture 3.1.12 Wheels installation

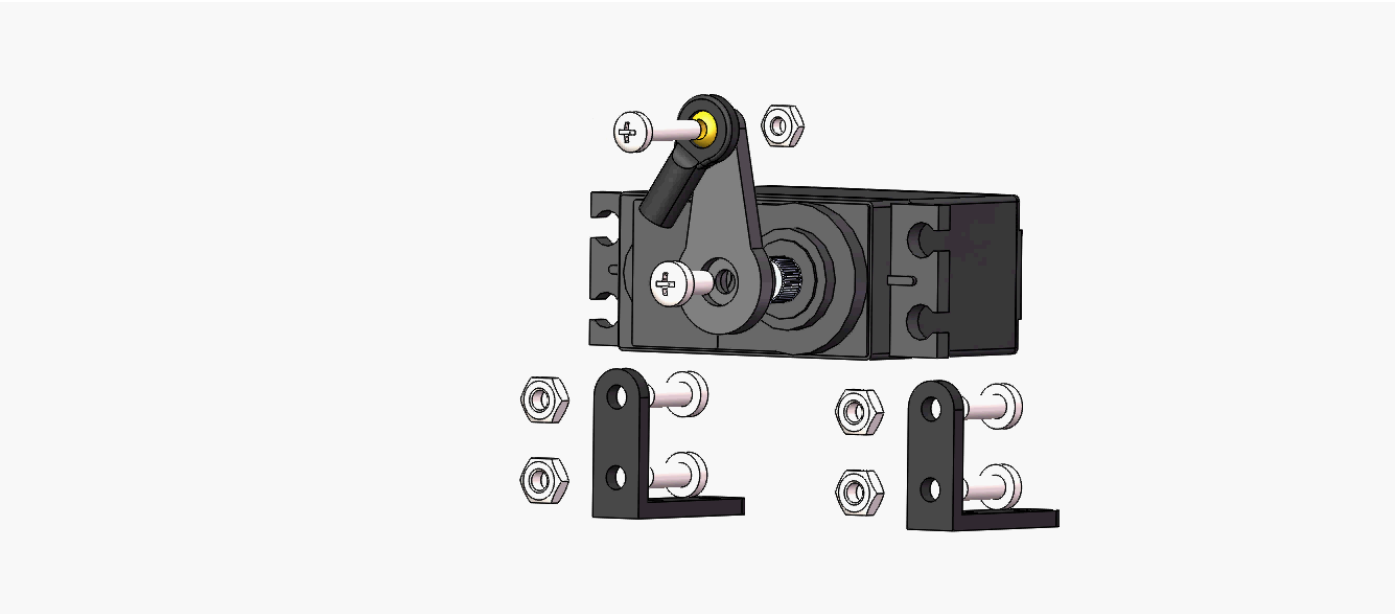


Picture 3.1.13 Tight wheels’ screw



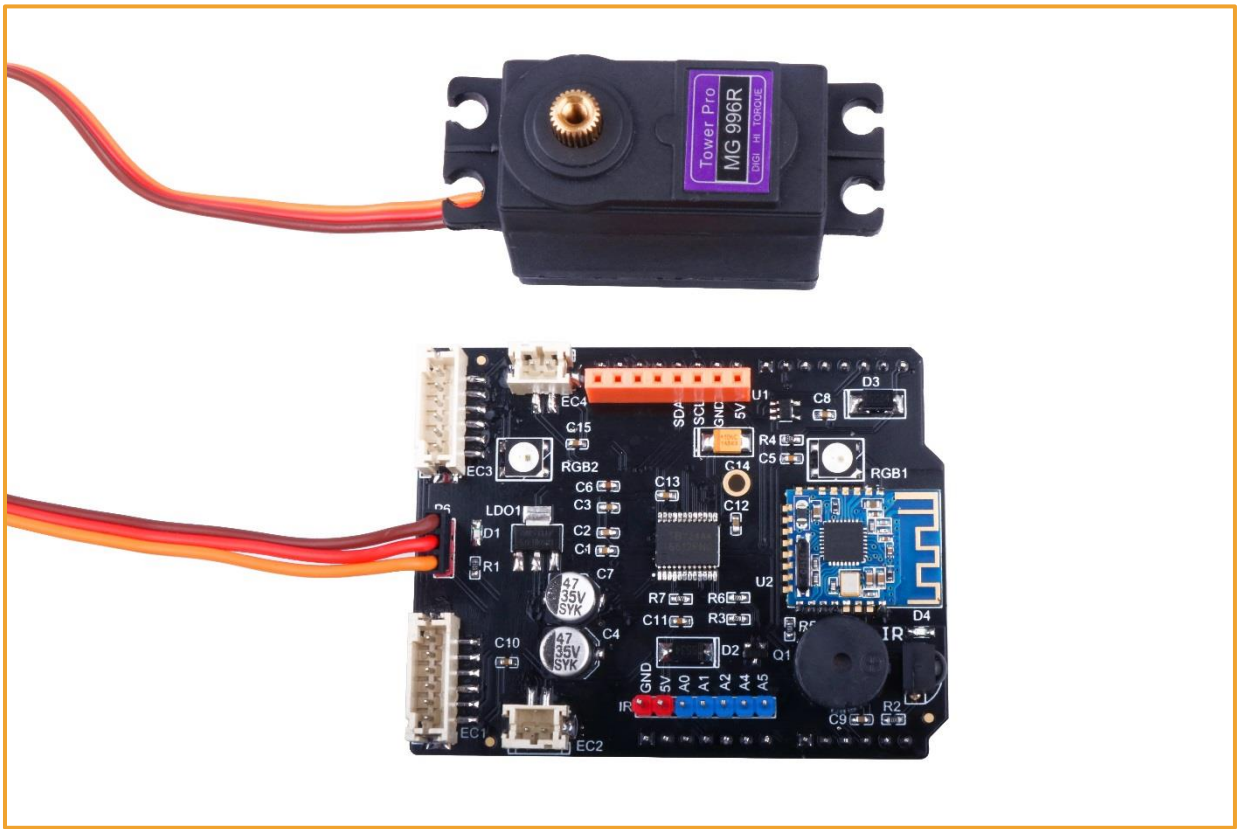
Picture 3.1.14 Wheels installation effect

3.1.2 Steering gear installation (Installation video tutorial\ 2.Servo Installation.avi)



Find the steering engine, two L-shaped steering engine stand, rudder horn+ fixing screw, M3*8,M3*12 screw and its nut.

Step 1: Connect the steering engine to its drive board, then insert the expansion board to the Arduino board. Copy the following application to the build environment (you can also open it on the CD-Rom **"Lesson\ModuleDemo\Servo\ServoCorrect\ServoCorrect.ino"**), and download the application to Arduino and make sure that steering engine spin 90°. Please check the picture 3.1.15 for more details about connecting of the motor and drive board.



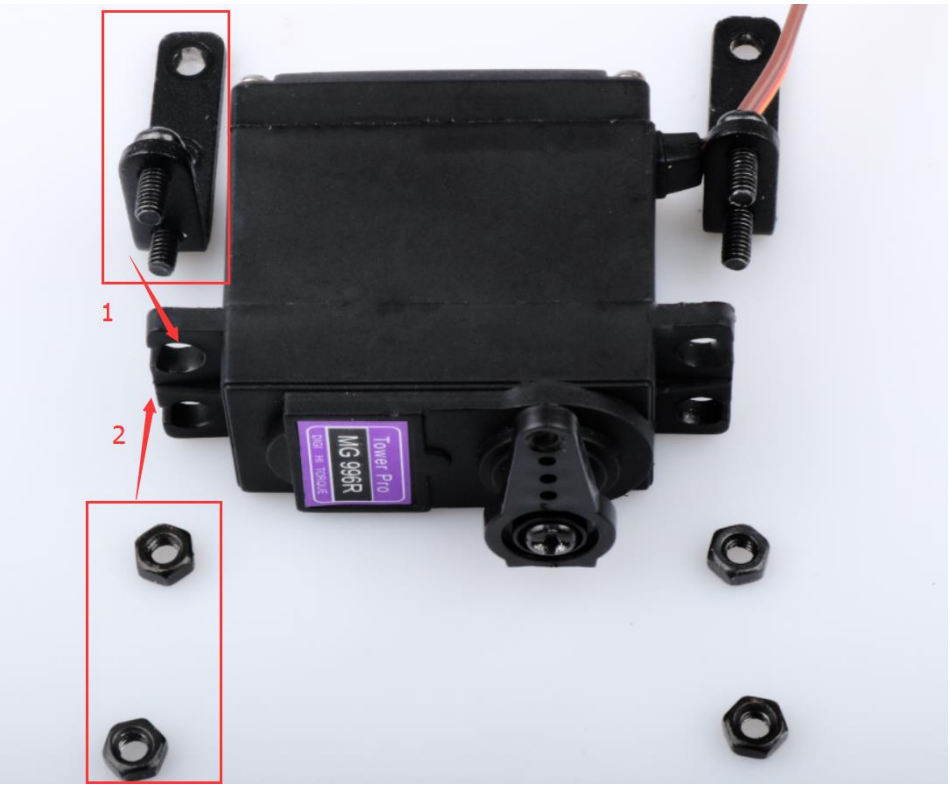
Picture 3.1.15 Connecting of MG996 and drive board

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 90;    // variable to store the servo position
void setup()
{
  myservo.attach(5);  // attaches the servo on pin 9 to the servo object
}
void loop()
 {
   myservo.write(pos);    // tell servo to go to position in variable 'pos'
   delay(15);             // waits 15ms for the servo to reach the position
}
```

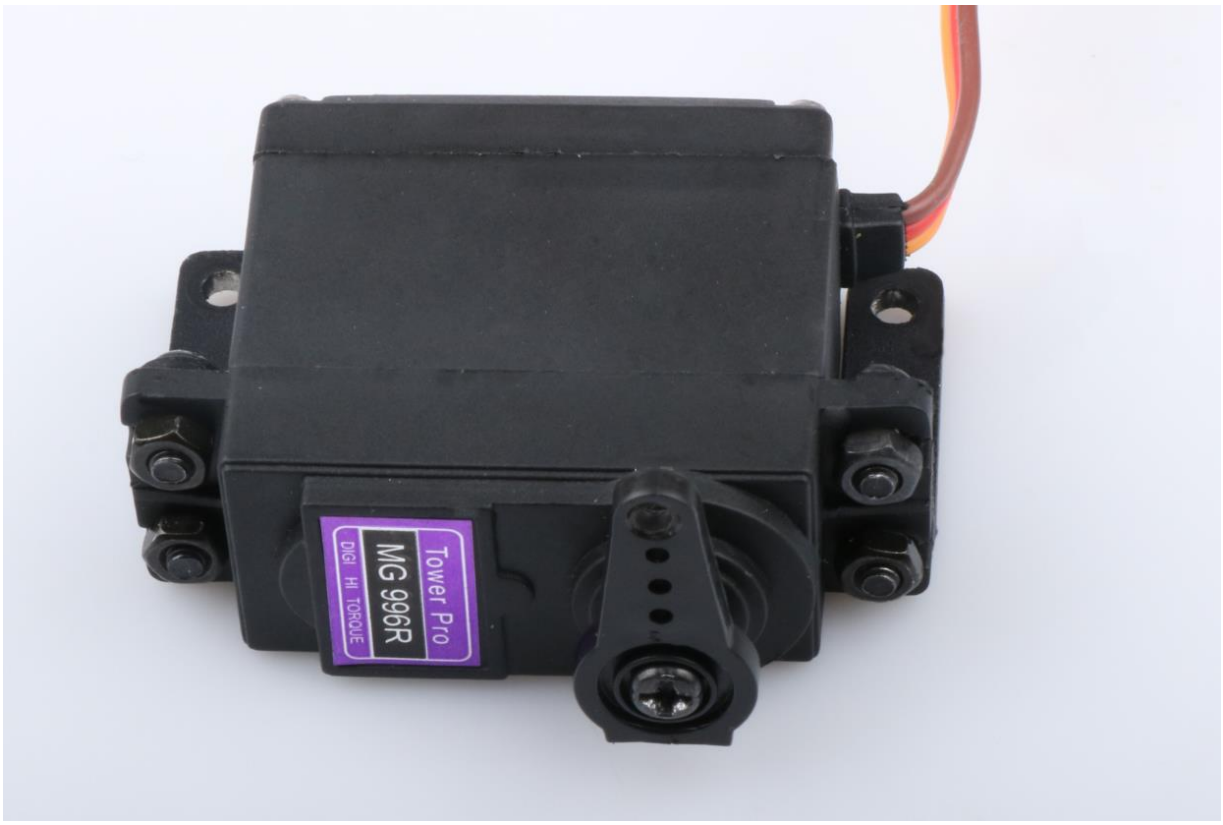
Step 2: Inserting rudder horn to steering engine vertically, and fix it. As the picture 3.1.16 .

Picture 3.1.16 Rudder horn installation

Step 3: Installation of L-shaped steering engine stand. The stands should be mounted the sides of the steering engine with screws.As picture 3.1.18.

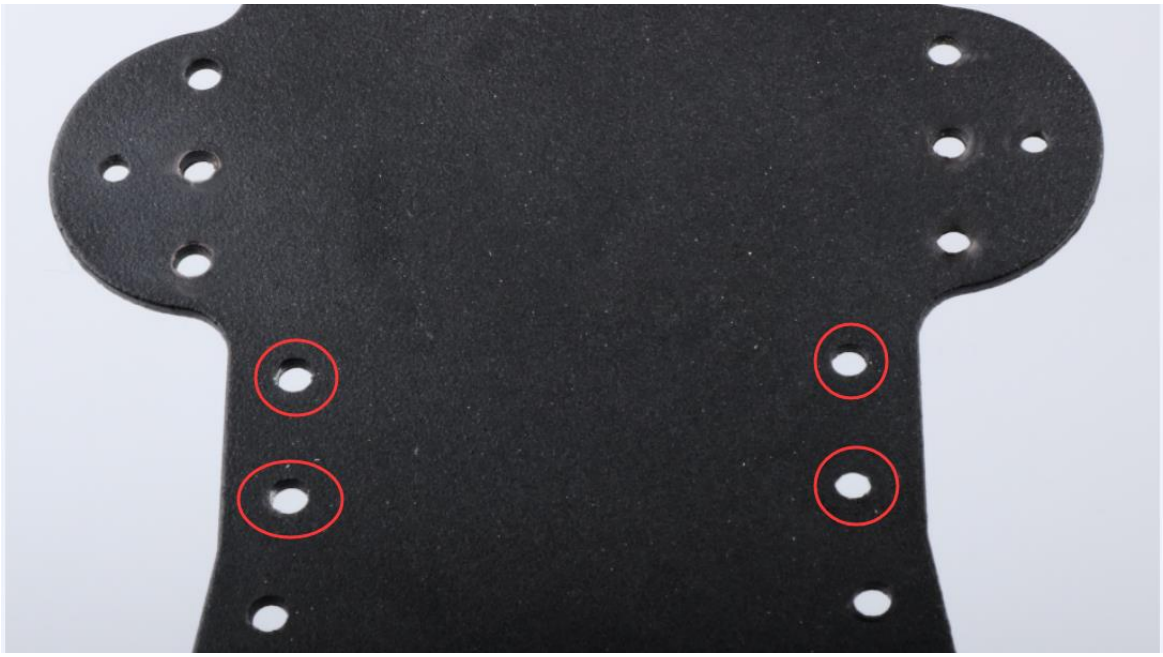


Picture 3.1.17 MG996 stand installation

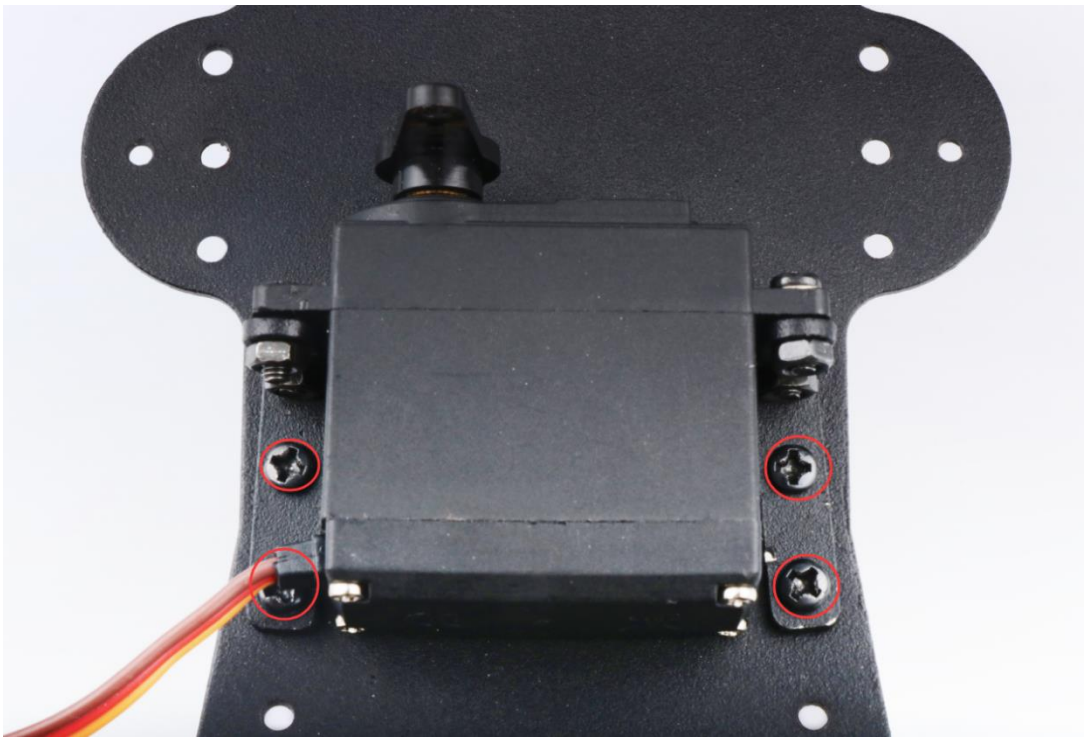


Picture 3.1.18 MG996 stand installation effect

Step 4: Fix the steering engine on the baseboard of the four marked holes in the picture 3.1.19 with screws. Please check the image 3.1.20 for more details.



Picture 3.1.19 Holes on baseboard



Picture 3.1.20 After Fixed

3.1.3 Front wheel and connecting rod Installation (Installation video tutorial\3.Wheels and connecting Rods.avi)

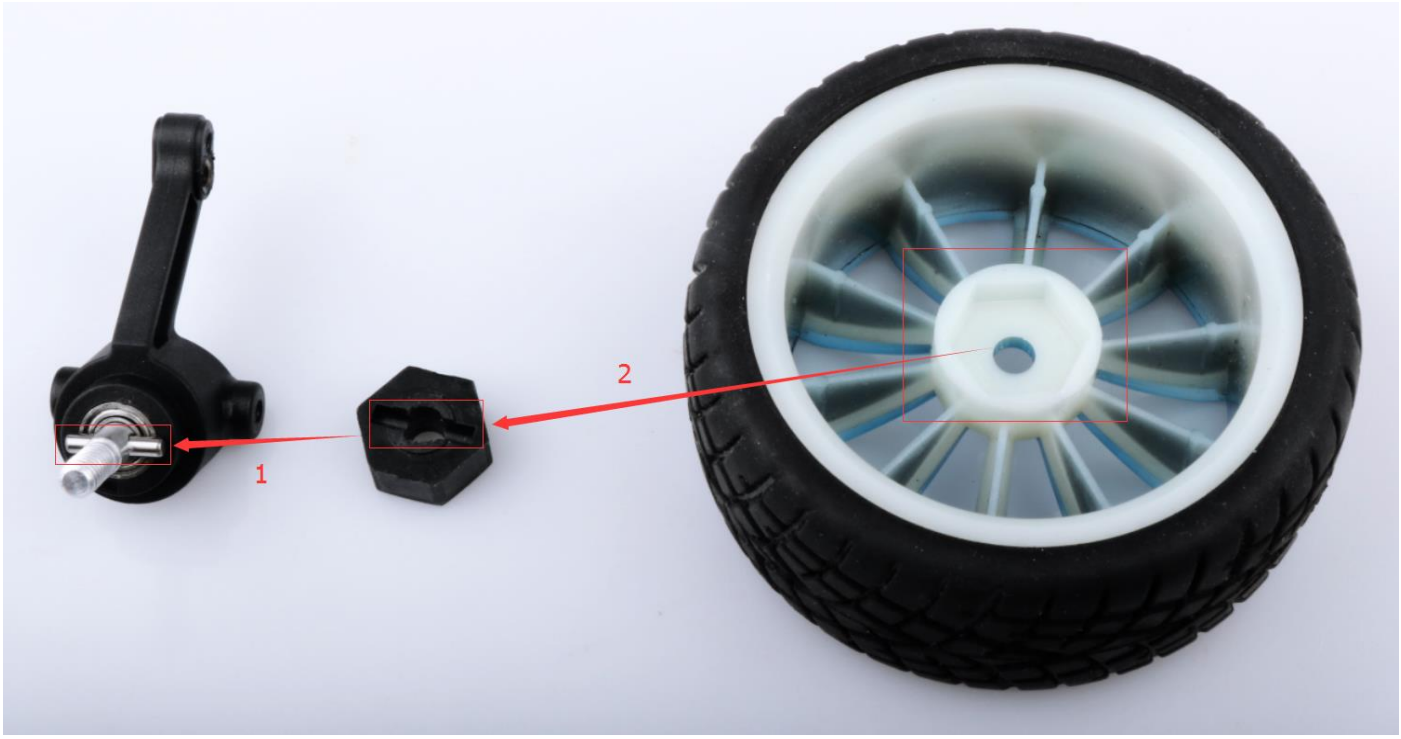
Step 1: Put the big bearing and the small one into steering knuckle separately. As picture 3.1.21.

Picture 3.1.21 Bearing installation

Step 2: Installation of the transmission shaft. Insert the transmission shaft to the steering knuckle. As picture 3.1.22.

Picture 3.1.22 Transmission shaft installation

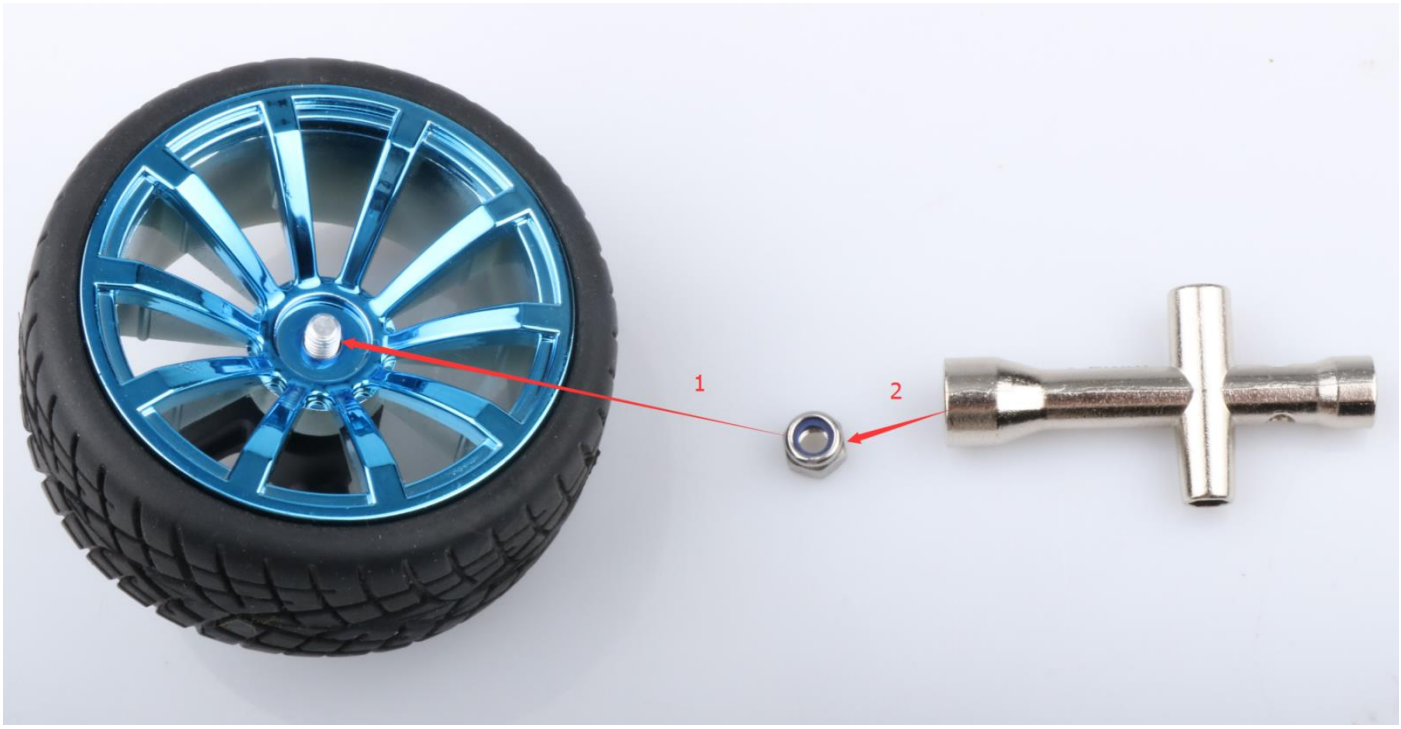
Step 3: Assembling of hexagonal connector and wheels. Insert the M2*10 bolt into the transmission shaft firstly, then insert hexagonal connector into the hexagonal hole on the wheels. As picture 3.1.24.



Picture 3.1.23 Front wheel installation

Picture 3.1.24 Front wheel installation effect

Step 4: Tight the screws. Assemble the other wheel with the same method. For more details please check the picture 3.1.25 and 3.1.26.

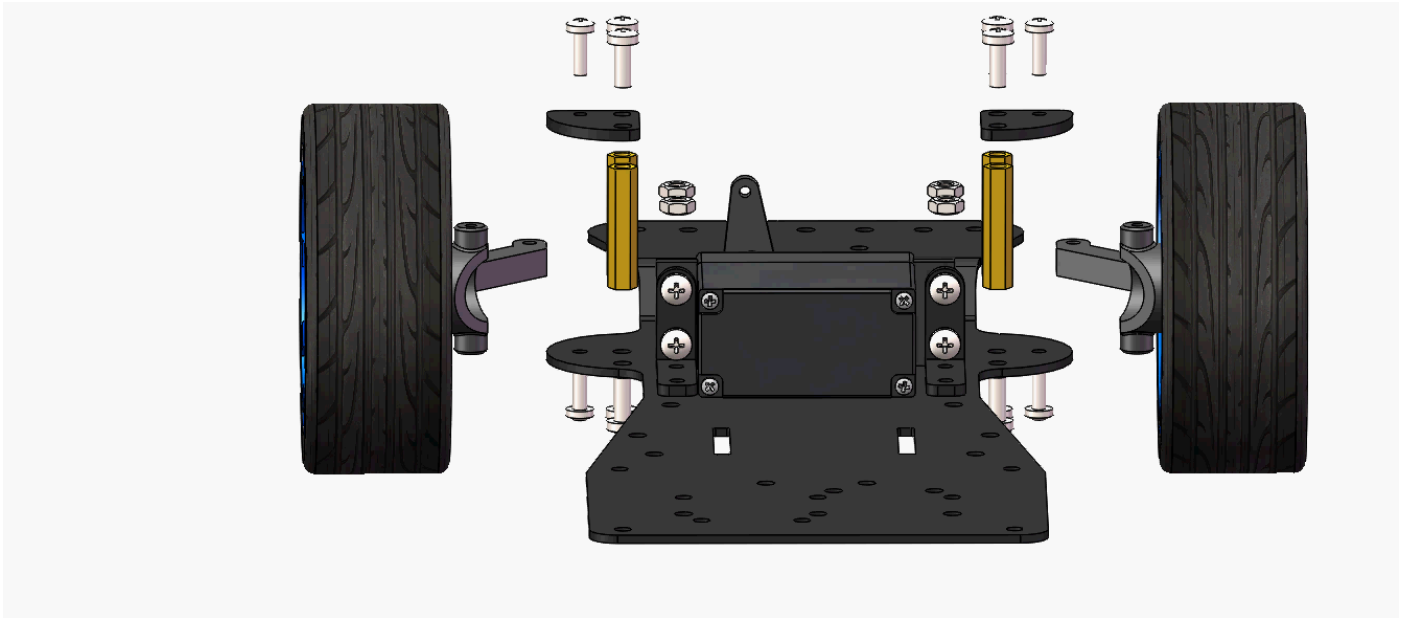


Picture 3.1.25 Wheel screw fixation

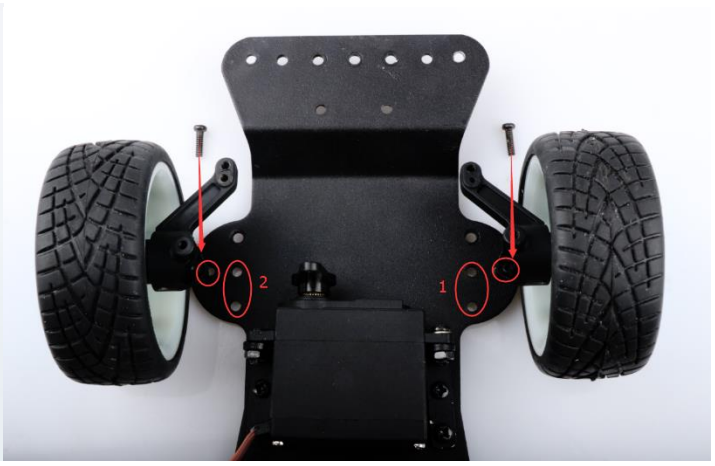
Picture 3.1.26 Complete wheel effect

Picture 3.1.27 Complete wheels effect

Step 5: Installation of wheels. Make the Steering Knuckle's screw hole point to the baseboard's screw hole and use M2.5*8 screw tight them. As picture 3.1.28, there are two spots marked 1 and 2, which should be fixed with two M3*22 copper cylinders on both sides.

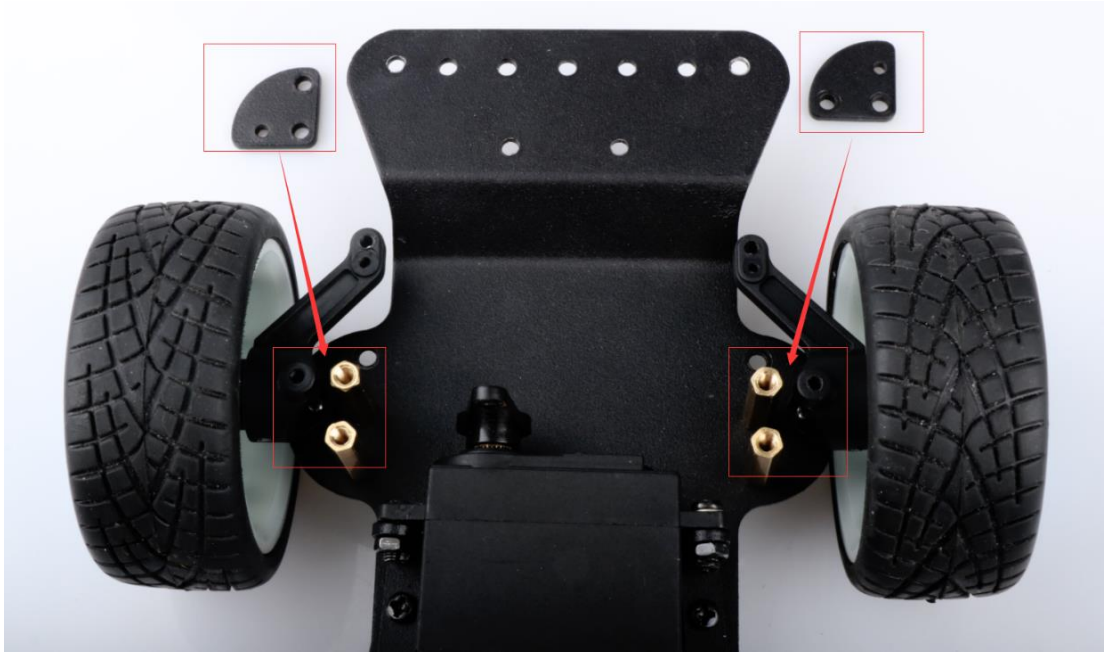


Picture 3.1.28-a Wheels installation



Picture 3.1.28-b Wheels installation

Step 6: Fix the wheels. At this step, we just need to fix the top of the Steering knuckle according to picture 3.1.29 then the wheels were fixed. The fan-shaped metal should be mounted on the top of the copper cylinders and Steering Knuckle with screws. For more details please check the picture 3.1.30.



Picture 3.1.29 Fan-shaped metal installation

Picture 3.1.30 Front wheels installation effect

Step 7: Installation of the connecting shaft. Find out 4 ball-shaped pull rod and 2 ball-shaped connecting rod (one is short, one is long). The two sides of ball-shaped connecting rods should be connected with pull rods. For more details please check the picture 3.1.31.

To install the connecting rod, we need to use M2.5*20mm, M2.5*16mm, M2.5*12mm three types of screws. As shown in picture 3.1.31.

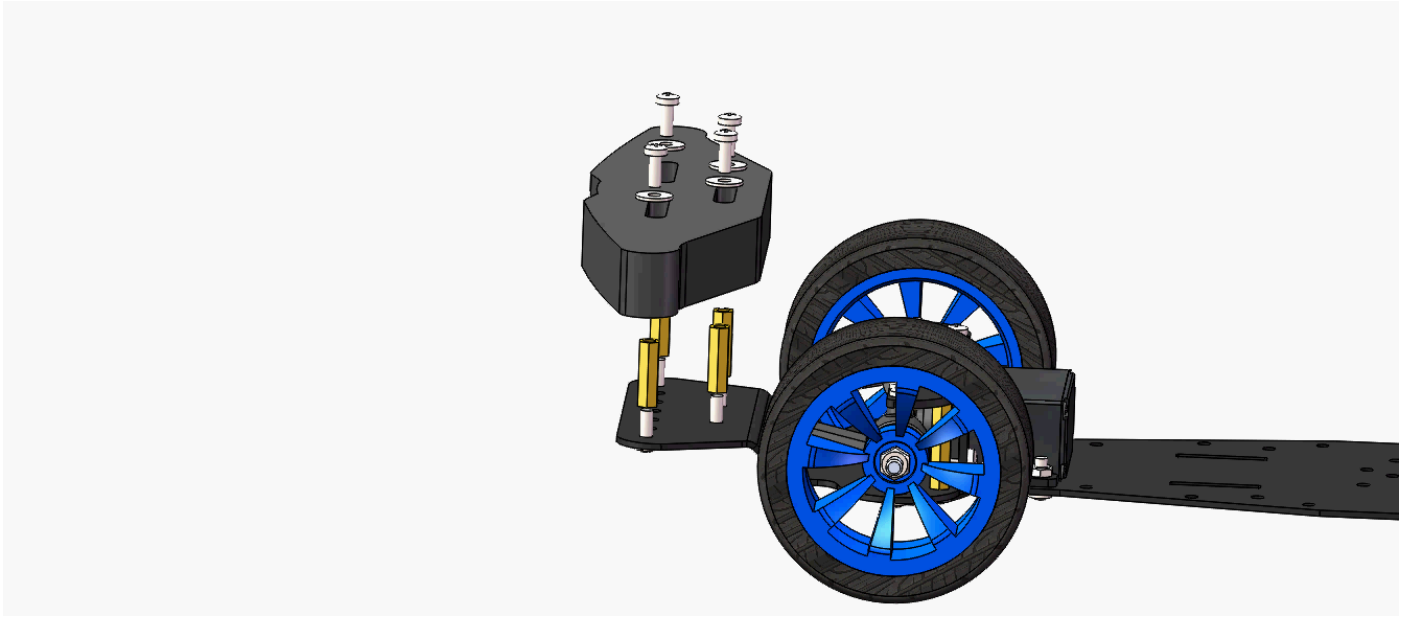
Picture 3.1.31 Connecting rods using

Finally, mount the connecting rods to the Aurora-Racing, the longer one should be connected with two wheels and under the Steering Knuckle, the short one should be connected with steering motor and right wheel,At the top of the steering cup,then screw it down.

Note: the length of connecting rod can be adjusted, that means that steering engine must be in 90° station when installing the connecting rod. The left and right wheels should be parallel with each other,the front wheel axle and the rear axle also should be parallel as picture 3.1.32. If not, please adjust the length of the connecting rod, otherwise, the Aurora-Racing cannot run straight.

Picture 3.1.32 Connecting rods installation

3.1.4 Anticollision and tracing module installation (Installation video tutorial\ 6.Anti-collision Cotton installation.avi)



Step 1: Four M3*16 copper cylinder should be fixed on the spot marked 1,2,3 on the baseboard according to the picture 3.1.33. For more details please check the picture 3.1.34.

Picture 3.1.33Fixed position of anticollision cotton

Picture 3.1.34 Copper cylinder installation

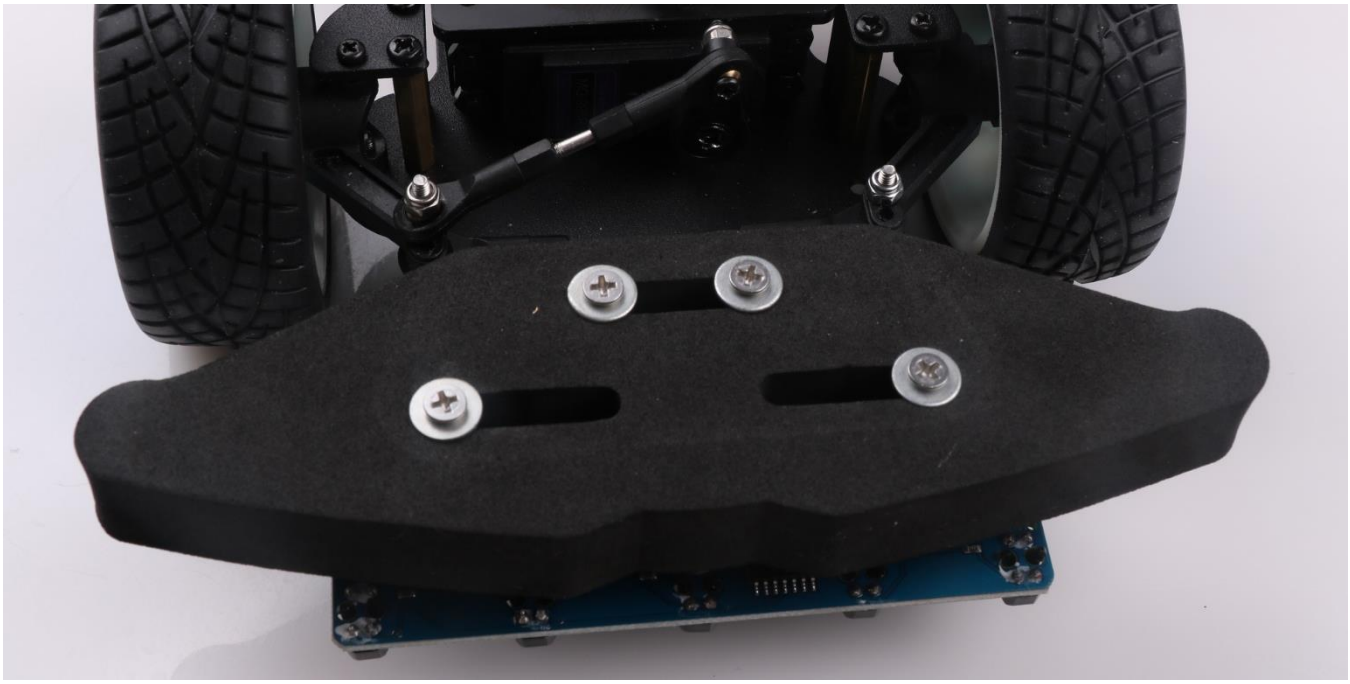
Step 2: Two Single M3*10 copper pillar should be fixed on the spot marked 2 and 3 on the baseboard according to the picture 3.1.34. For more details please check the picture 3.1.35.

Picture 3.1.35 Front and reverse graph

Step 3: Installation of tracing module. Make sure that tracing module is in the centre of the car. For more details please check the picture 3.1.36.

Picture 3.1.36 Tracking moudle installation

Step 4: Installation of anticollision pad. The anticollision pad should be fixed with spacers and screws. Please check the details in picture 3.1.37.

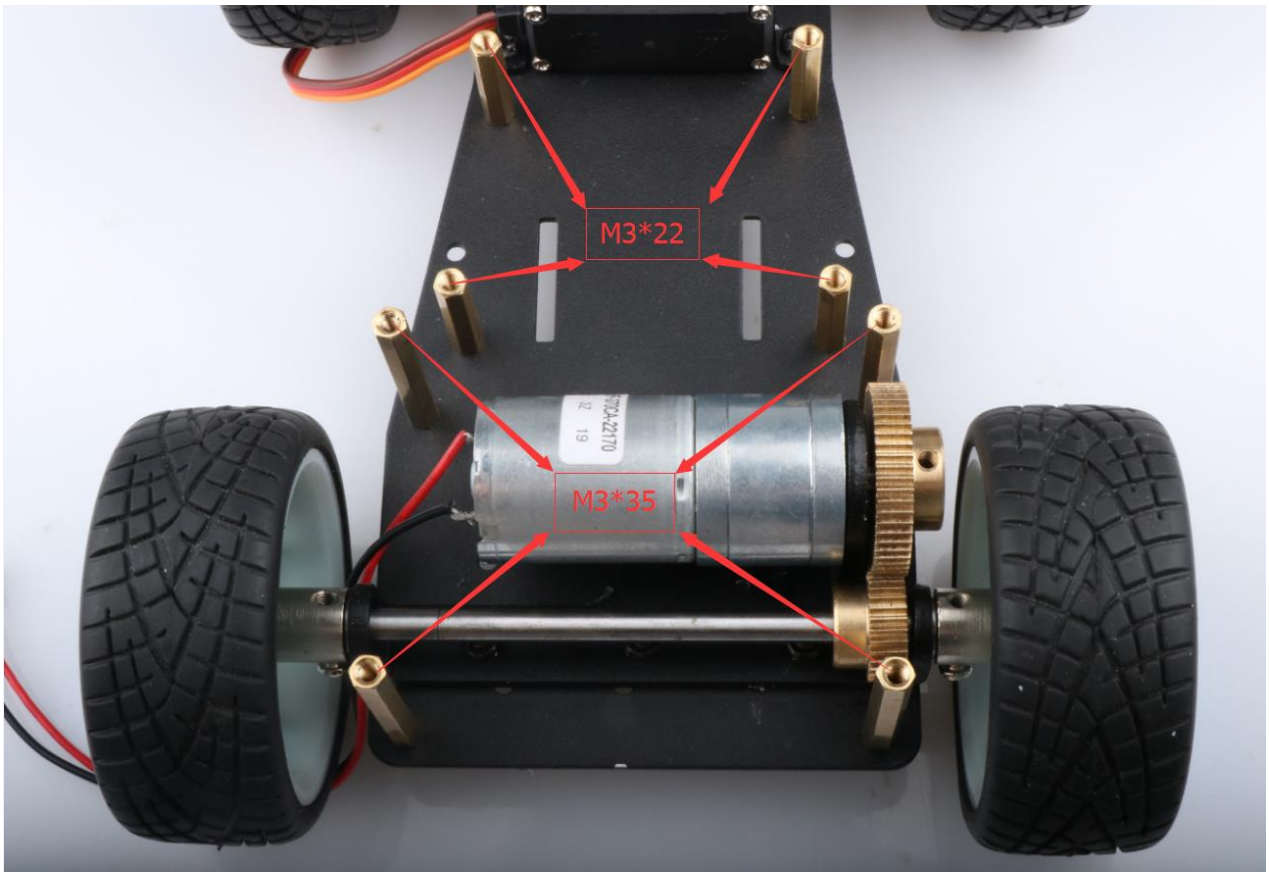


Picture 3.1.37 Anticollision pad installation

3.1.5 Installation of Acrylic baseboard

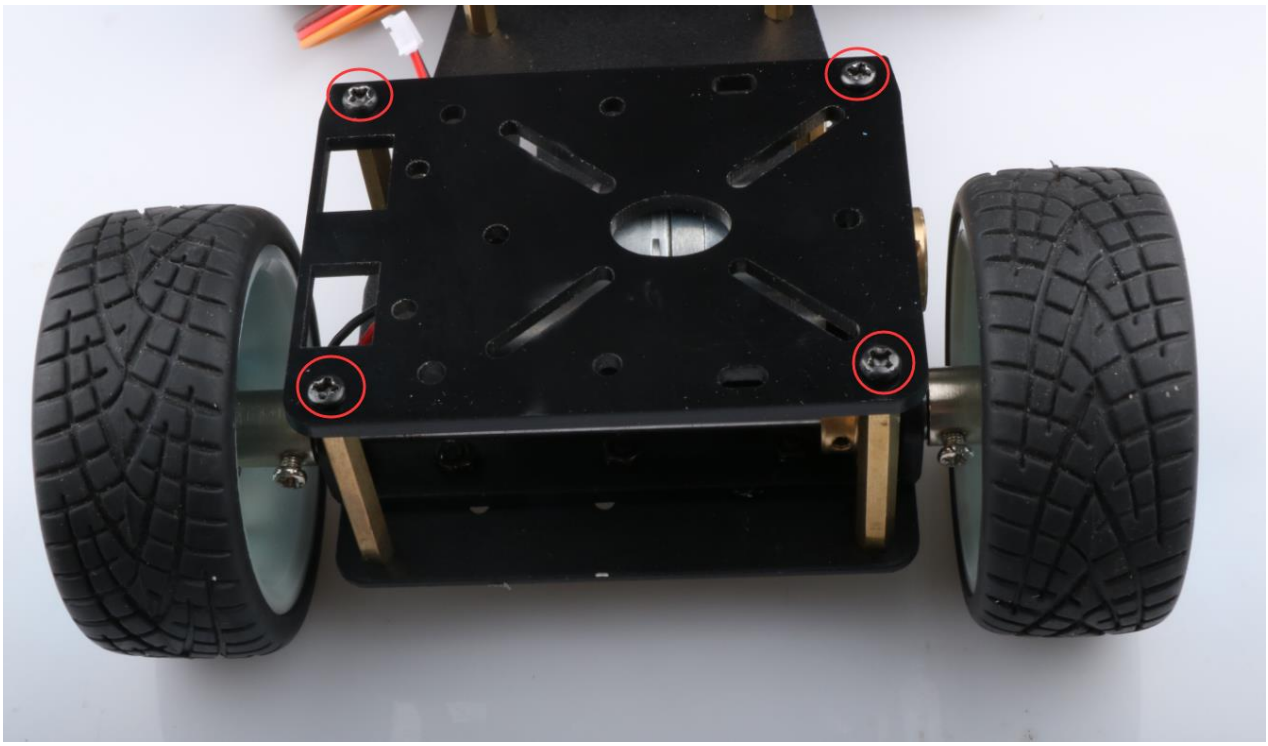
There are two acrylic baseboards in total, one is for battery supporting and the other one is for main control board supporting.

Step1 : Installation of copper cylinder. The copper cylinder can support the two acrylic baseboards. There need M3*22 and M3*35 copper cylinders,8pcs in total,should be fixed like the picture 3.1.38.



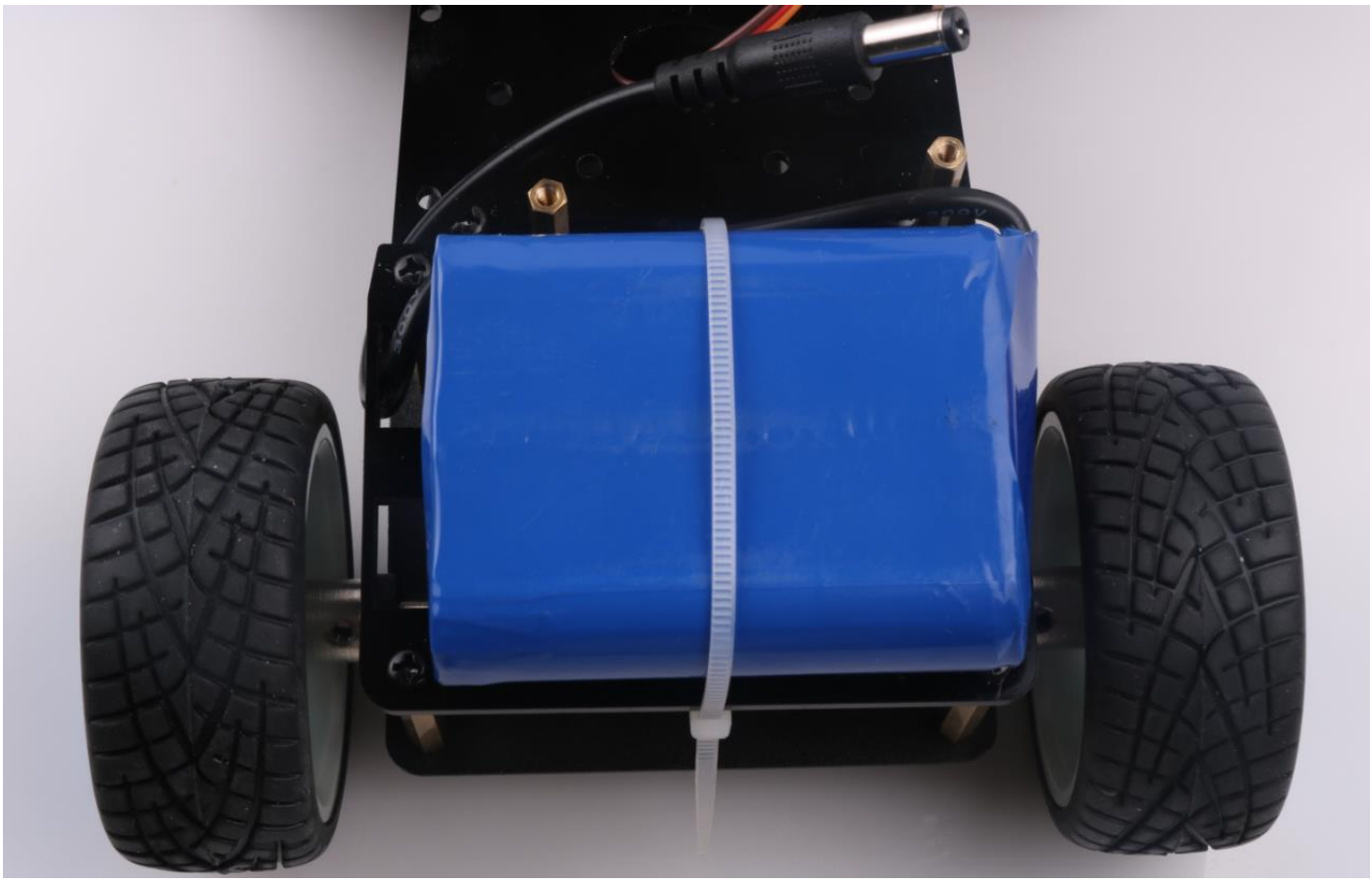
Picture 3.1.38 Copper cylinder installation

Step 2:Installation of acrylic boards. Please check the details in the picture 3.1.39.



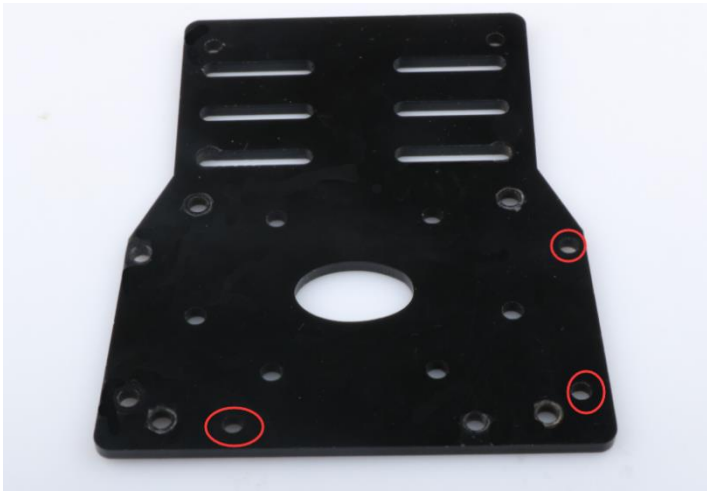
Picture 3.1.39 Acrylic installation

Step 3: Fix battery. Put the battery on the acrylic board and fix the battery with cable tie. As picture 3.1.40.



Picture 3.1.40 Battery fixed

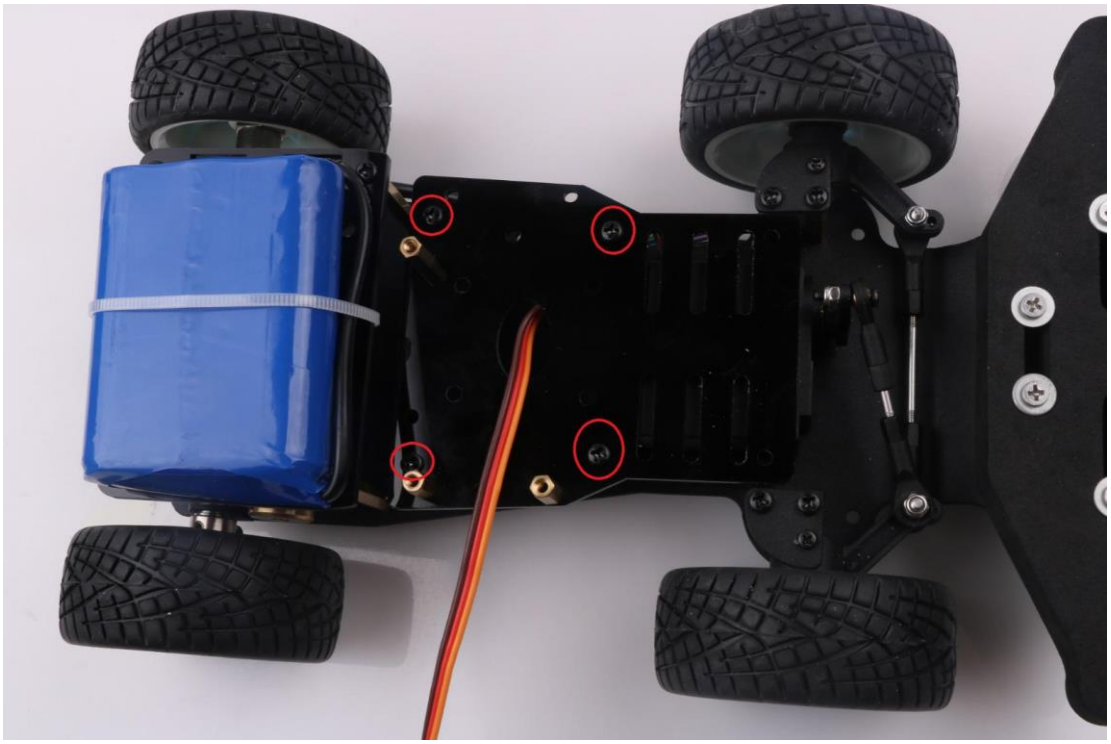
Step 4: Installation of the other acrylic board. This acrylic board is used to support the main control board. We don't recommend to fix main control board directly on the surface of the acrylic board for the convenience of change main board in future.Firstly we fixed 3 M3*10mm copper cylinders ,as picture 3.1.41 shown, for the acrylic board. Please check the details in picture 3.1.42.



Picture 3.1.41 Position to install copper cylinder

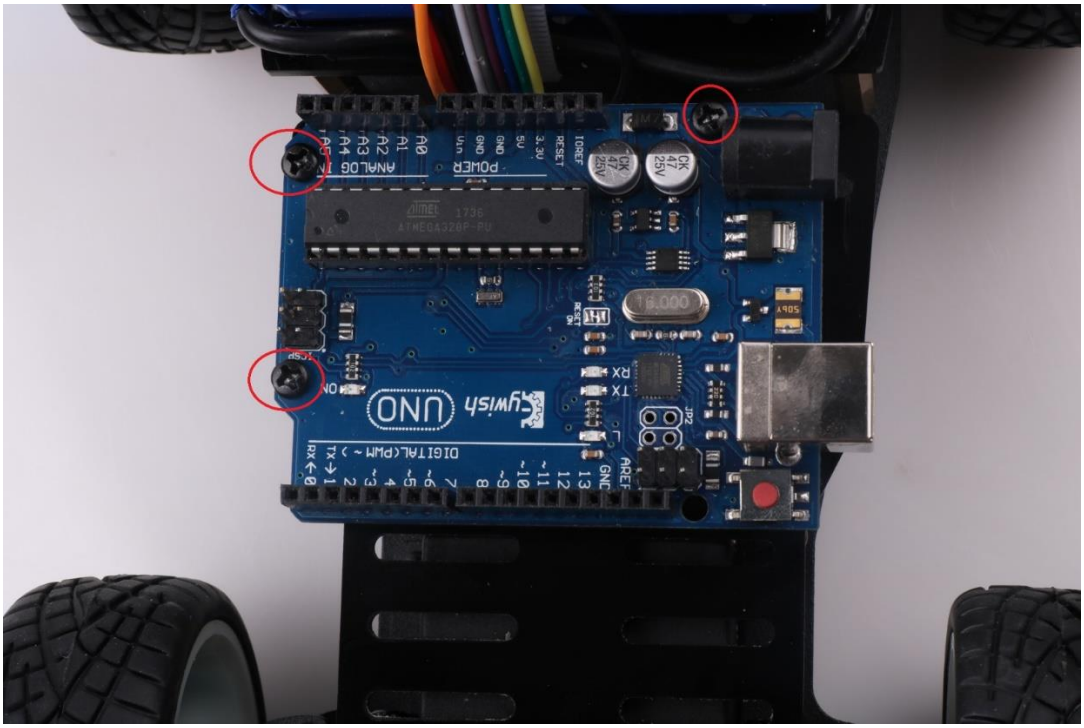
Picture 3.1.42 M3*10mm copper cylinder effect

Step 5: Installation of the acrylic board on the car with screws. Please check the details in picture 3.1.43.



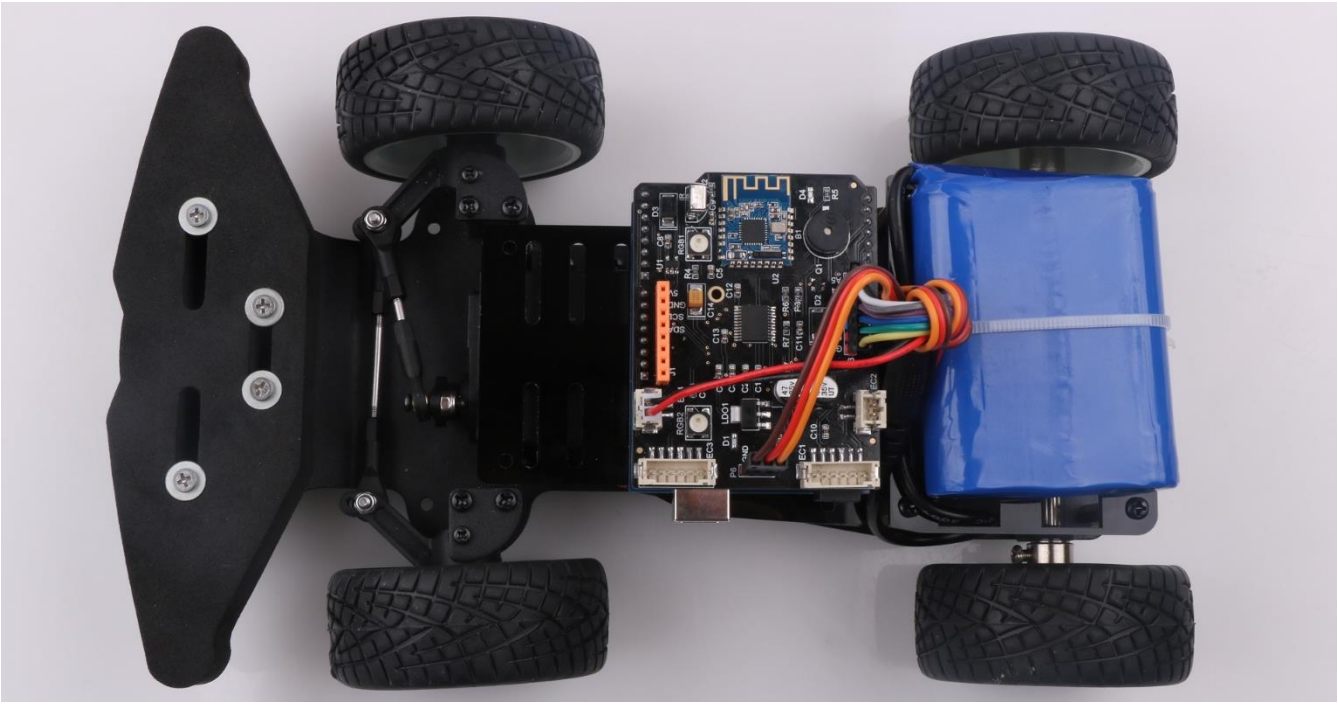
Picture 3.1.43 Acrylic installation effect

Step6 : Installation of main control board. Keywish UNO-R3 is chosen as the main control board for the “Aurora-Racing”, of course, you can also choose the Raspberry Pi, which can make your car more interesting, smarter. It is quite simple for the installation , just fix the main control board on the three copper cylinders with screws. Please check the details in the picture 3.1.44.



Picture 3.1.44 Arduino UNO installation effect

Step 7: Insert the expansion board to the main control board. Connect the the steering gear, motor and tracking module to the expansion board.As picture 3.1.45.

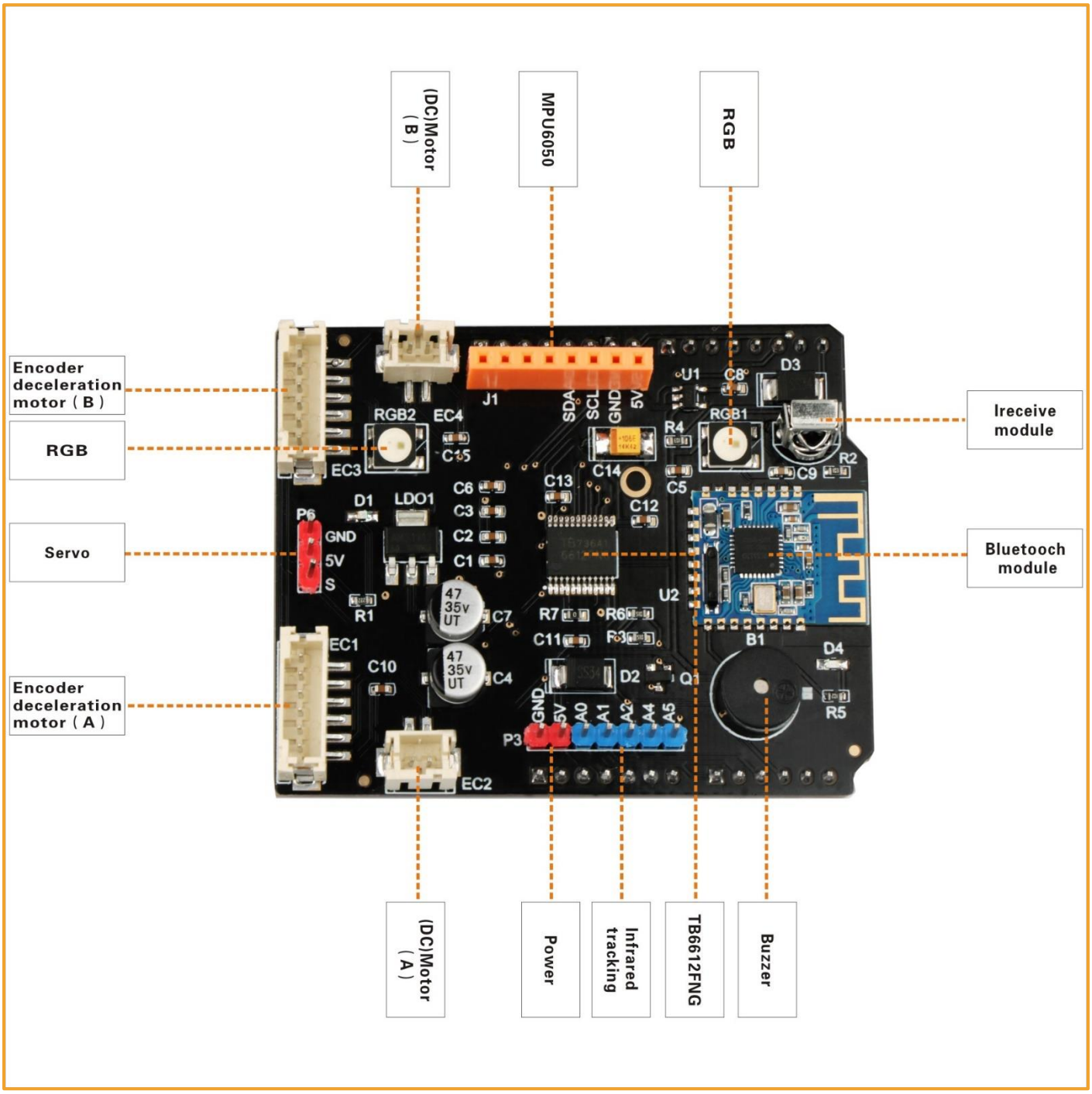


Picture 3.1.45 Extension board installation and connecting

So far, the installation of the car is basically finished. We believe that you have some basic understanding of the car in structure, function and modules after the short time of installation. The next thing you need to do is downloading the application to the development board to give it life. All applications can be found in the CD, enjoy now.However,it will be more fun if you can understand the program,write your own program.Let's learn the software part together.

3.2 Experiment

TB6612FNG motor drive board Frame diagram



Picture 3.2.1 TB6612 motor drive board Frame diagram

TB6612FNG is a motor driver extension board special for Arduino UNO R3, with Bluetooth 4.0 Moudle,MPU6050 Moudle Interface,IR,2 x6 pins encoder deceleration interfaces,2 x 2Pins PWM DC motor Interfaces,1 x servo Interface,5 extension Pins,2 x RGB LED ,Buzzer.The Shield will help you handle motor

diver and control issues ,give a easier and more intelligent solution when you build a Arduino Car,Robot,Balance car etc;

Features:

- 2 x 2pins PWM DC motor driver interface
- 2x 6pins motor encoder deceleration interface
- 1x 3pins servo interface
- 2x RGB LED light
- 1x mpu6050 module interface
- 1x On Board passive buzzer
- 1x On Board integrated infrared receiver
- 5x Extended interface A0 A1 A2 A4 A5

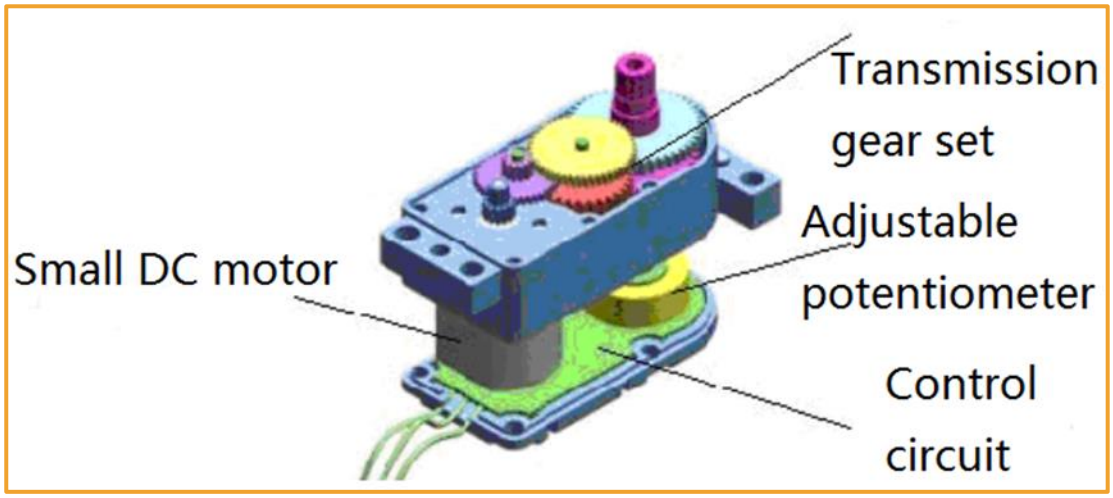
Specification:

Connect directly to Arduino UNO R3 and powed through UNO motherboard,Working Voltage:6-20V Output current: 1.2A single channel continuous drive current start / peak current: 2A (continuous pulse) / 3.2A (single pulse)

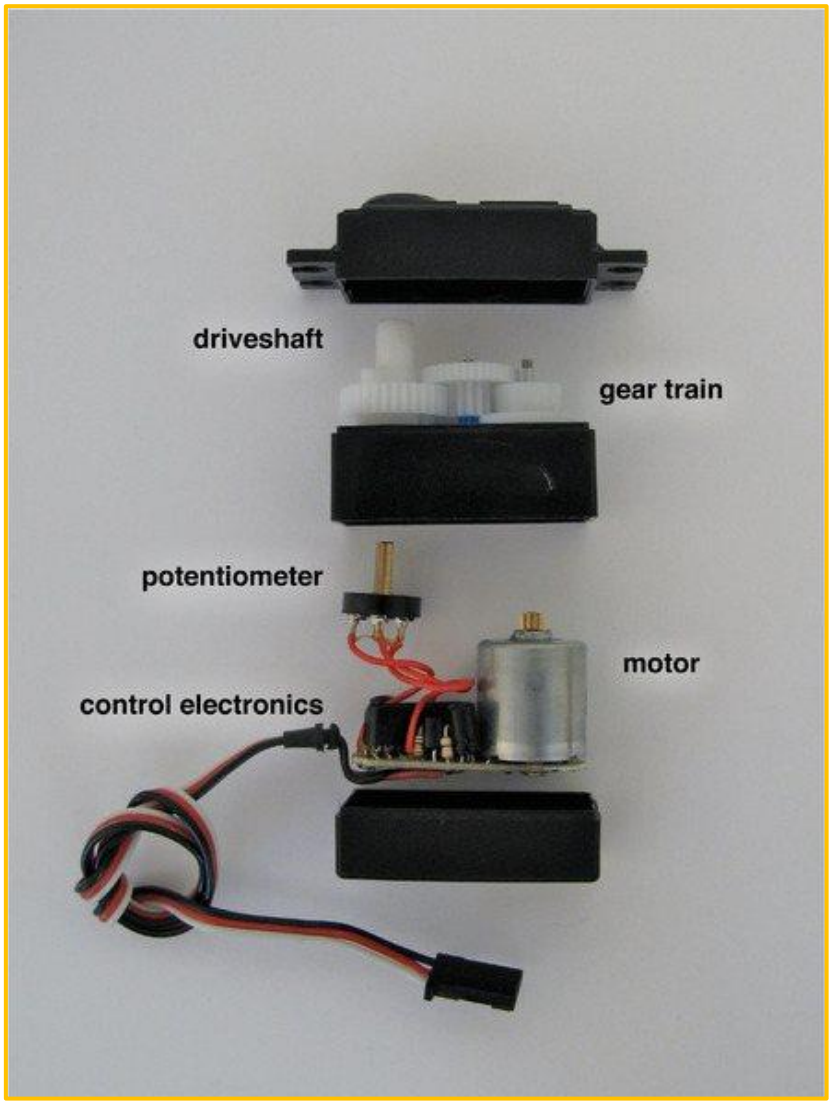
3.2.1 Servo Test Experiment

3.2.1.1 Description

Steering gear also called servo motor which originally used in ships. Since it can control the angle continuously through program, it has been widely used in intelligent steering robot to achieve all kinds of joint movement. It has characteristics of small volume, large torque, high stability, simple external mechanical design. Either in hardware or software design, the design of steering engine is an important part of car controlling. In general,a steering gear is mainly composed of the following parts, steering wheels, gear group, position feedback potentiometer, DC motor and control circuit.Figure 3.2.2 and figure 3.2.3 are the internal structure . Aurora-Racing used MG996R(180 degree) steering gear.



Picture 3.2.2 Internal structure A



Picture 3.2.3 Internal structure B

Picture 3.2.4 996R Physical map

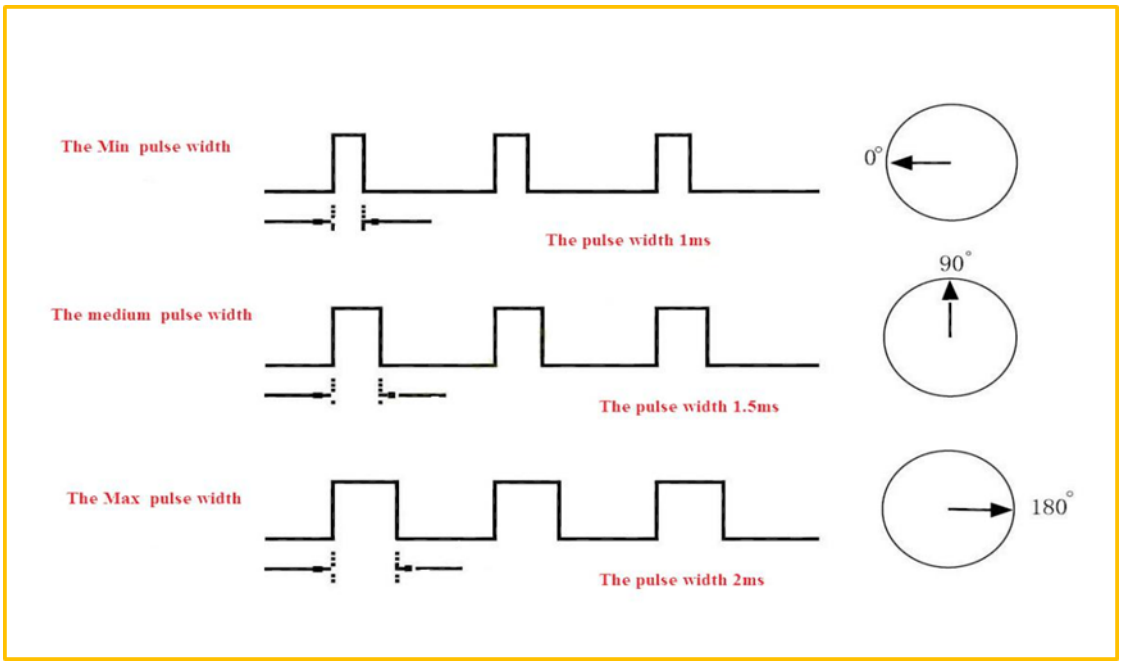
3.2.1.2 Working Principle

1. Steering gear

The control signal enters the signal modulation chip by the receiver channel, gets the DC bias voltage. The steering gear has a reference circuit which generates a reference signal with a period of 20ms and a width of 1.5ms. Comparing the obtained DC bias voltage with the voltage of the potentiometer,and obtaining the output voltage difference. Finally, the positive and negative output voltage difference in the motor driver chip decide the positive and negative rotation of motor. When the speed of motor is certain,the cascade reducer gear will drive potentiometer to rotate so that the voltage difference is reducing to 0,the motor will stop rotating.

When the control circuit receives the control signal, the motor will rotate and drive a series of gear sets, the signal will move to the output steering wheel when the motor decelerates. The output shaft of steering gear is connected with the position feedback potentiometer, the potentiometer will output a voltage signal to the control circuit board to feedback when the steering gear rotates, then the control circuit board decides the rotation direction and speed of the motor according to the position, so as to achieve the goal. The working process is as follows: control signal→control circuit board→motor rotation→gear sets deceleration→ steering wheel rotation→position feedback potentiometer→control circuit board feedback.

The control signal is 20MS pulse width modulation (PWM), in which the pulse width varies linearly from 0.5-2.5MS, the corresponding steering wheel position varies from 0-180 degrees, which means the output shaft will maintain a corresponding certain degrees if providing the steering gear with certain pulse width. No matter how the external torque changes, it only changes position until a signal with different is provided as shown in Picture.3.2.27. The steering gear has an internal reference circuit which can produce reference signal with 20MS period and 1.5MS width, there is a comparator which can detect the magnitude and direction of the external signal and the reference signal, thereby produce the motor rotation signal.



Picture 3.2.5 Relationship between output angle and the input pulse

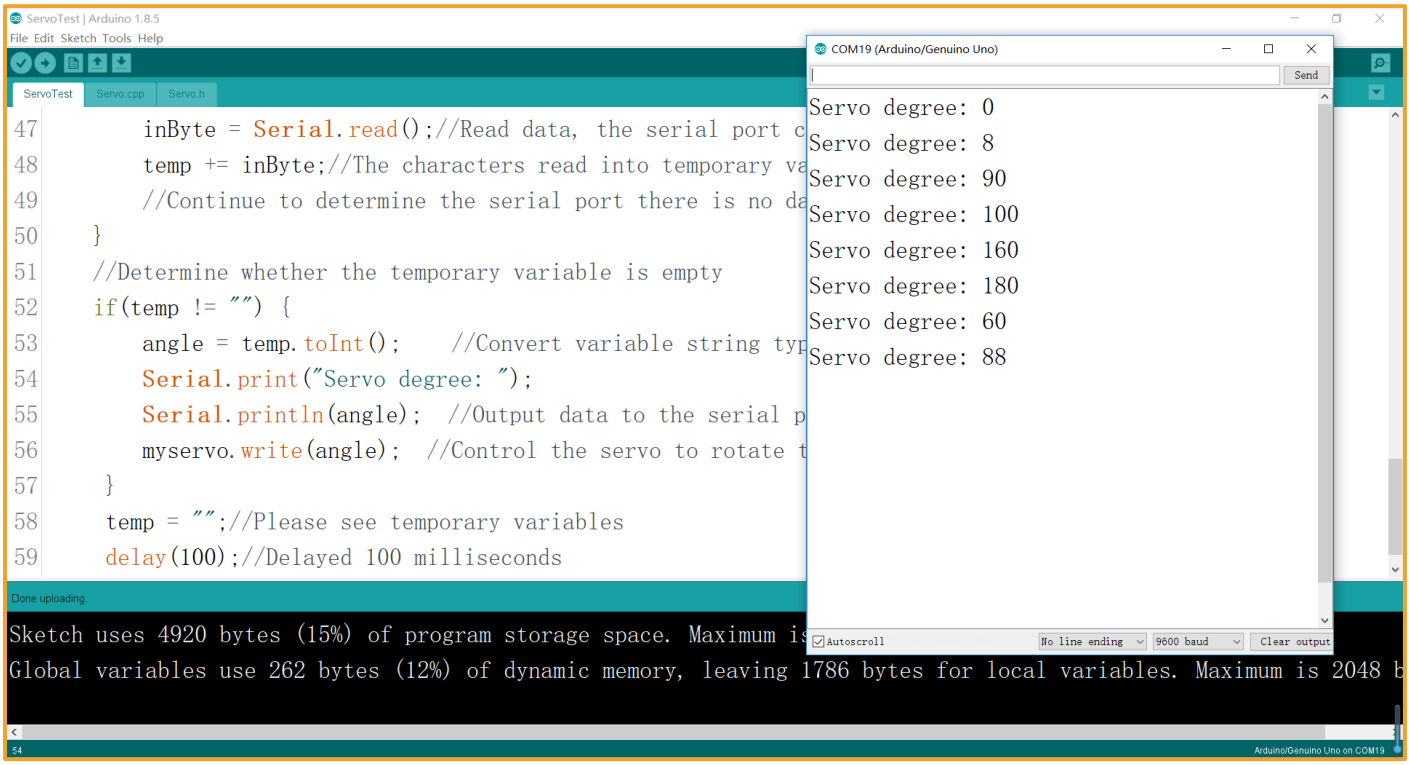
3.2.1.3 Servo Degree Test

The steering gear rotate to the specified angle according the input of the serial port

Program location: "Lesson\ModuleDemo\Servo\ServoTest\ServoTest.ino"

```
#include "Servo.h"

Servo myservo;  // Create a servo motor object

char inByte = 0; //Serial port to receive data
int angle = 0;  //Angle value
String temp = "";//Temporary character variables, or use it for the cache

void setup()
{
    myservo.attach(5);    //Define the steering gear pin to 13
    Serial.begin(9600);  //Set the baud rate
}

void loop()
{
    while (Serial.available() > 0) //Determine whether the serial data
    {
        inByte = Serial.read();//Read data, the serial port can only read 1 character
        temp += inByte;//The characters read into temporary variables inside the cache,
        //Continue to determine the serial port there is no data, know all the data read
out
    }
    //Determine whether the temporary variable is empty
    if(temp != "") {
        angle = temp.toInt();    //Convert variable string type to integer
        Serial.print("Servo degree: ");
        Serial.println(angle);  //Output data to the serial port for observation
        myservo.write(angle);  //Control the servo to rotate the corresponding angle.
     }
     temp = "";//Please see temporary variables
     delay(100);//Delayed 100 milliseconds
}
```

Result：



Picture 3.2.6 WS2811 Pin configuration

3.2.2 RGB WS2811 Experiment

3.2.2.1 RGB WS2811 Description

The WS2811 is 3 output channels special for LED driver circuit. Its internal includes intelligent digital port data latch and signal reshaping amplification drive circuit. Also includes a precision internal oscillator and a 12V voltage programmable constant current output drive. In the purpose of reduce power supply ripple, the 3 output channels designed to delay turn-on function.

Unlike the traditional RGB, WS2811 is integrated with a WS2811 LED drive control special chip, which requires a single signal line to control a LED lamp or multiple LED modules. Features as below:

- Output port compression 12V.
- Built-in voltage-regulator tube, only a resistance needed to add to IC VDD feet when under 24V power supply.
- 256 Gray-scale adjustable and scan frequency is more than 2KHz.
- Built in signal reshaping circuit, to ensure waveform distortion do not accumulate after wave reshaping to the next driver
- Built-in electrify reset circuit and power-down reset circuit.
- Cascading port transmission signal by single line

- Any two point the distance less than 5 Meters transmission signal without any increase circuit.
- When the refresh rate is 30fps, the cascade number is at least 1024 pixels.
- Send data at speed of 800Kbps.

For more parameters of WS2812,please check the file in CD:

“aurora-racing\Document\WS2811.pdf”

3.2.2.2 RGB WS2811 working principle

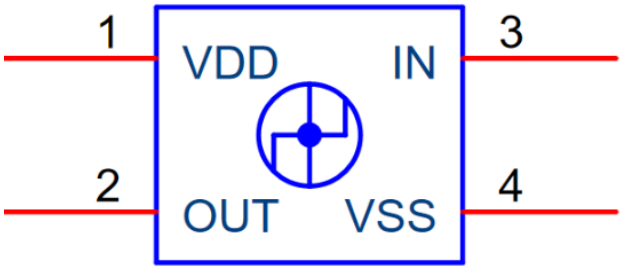
IC uses single NZR communication mode. After the chip power-on reset, the DIN port receive data from controlle,the first IC collect initial 24bit data then sent to the internal data latch,the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade IC through the DO port. After transmission for each chip, the signal to reduce 24bit. IC adopt auto reshaping transmit technology,making the chip cascade number is not limited the signal transmission,only depend on the speed of signal transmission.

The data latch of IC depend on the received 24bit data produce different duty ratio signal at OUTR,OUTG,OUTB port.All chip synchronous send the received data to each segment when the DIN port input a reset signal. It will receive new data again after the reset signal finished. Before a new reset signal received, the control signal of OUTR ,OUTG, OUTB port unchanged. IC sent PWM data that received justly to OUTR, OUTG, OUTB port, after receive a low voltage reset signal the time retain over **280μs**.

Pin function and Pin configuration as picture 3.2.7

NO	Symbol	Function description
1	VDD	Power supply voltage
2	OUT	Data signal cascade output
3	IN	Data signal input
4	VSS	Ground

WS2811 Pin function



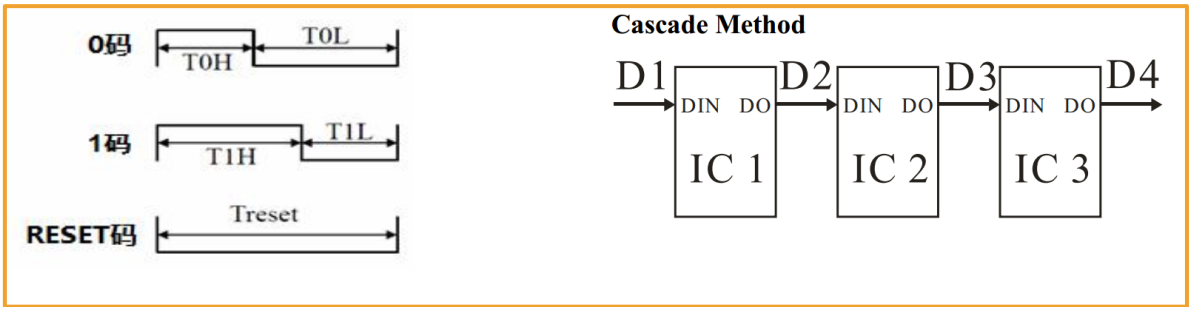
Picture 3.2.7 WS2811 Pin configuration

3.2.2.2 WS2811 drive principle

WS2811 The low level is represented by T0, consists of a 0.5μs high level and 2μs low level. The high level is represented by T1, consists of a 2μs high level 0.5μs low level. When low level last more than 50μs there will be a reset signal.

T0H	0 code, high voltage time	0.5μs
T1H	1 code, high voltage time	2μs
T0L	0 code, low voltage time	2μs
T1L	1 code, low voltage time	0.5μs
RES	Frame unit, low voltage time	>50μs

Sequence Chart



Picture 3.2.8 Waveform sequence diagram

Composition of 24bit Data

R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Data transmit in order of GRB, high bit data at first.

3.2.2.2 RGB WS2811 Test Program

Open the material,path： “Lesson\ModuleDemo\RGB\RGB_test2\RGB_test2.ino”

The purpose of this experiment is to show the double breath lamp effect that the RGB1 is gradually brightened, and the RGB2 gradually extinguished separately.Program as follows.

```
#include "Adafruit_NeoPixel.h"
#define RGB_PIN A3
#define MAX_LED 2
int RGB1_val = 0;
int RGB2_val = 255;
bool trig_flag = true;
Adafruit_NeoPixel strip = Adafruit_NeoPixel(MAX_LED, RGB_PIN, NEO_GRB + NEO_KHZ800);
void setup()
{
    strip.begin();  //default close all led
    strip.show();
}
```

```
void loop()
{
    // write 0 ~ 255  ~ 0 value to RGB1
    strip.setPixelColor(0, strip.Color(RGB1_val, 0, 0));
    // write 255 ~ 0 ~ 255 value to RGB2
    strip.setPixelColor(1, strip.Color(0, 0, RGB2_val));
    strip.show();
    if (trig_flag == 1) {
        RGB1_val--;
        RGB2_val++;
    } else {
        RGB1_val++;
        RGB2_val--;
    }
    if (RGB1_val >= 255 || RGB1_val <= 0 ) {
        trig_flag = !trig_flag;
    }
    delay(30);
 }
```

Adafruit_NeoPixel(MAX_LED, RGB_PIN, NEO_RGB + NEO_KHZ800);

The first parameter sets the number of RGB display, the second parameter sets the GPIO port that RGB use, and the third parameter sets the RGB data transmission speed mode.

NEO_KHZ800 800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)

NEO_KHZ400 400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)

NEO_GRB Pixels are wired for GRB bitstream (most NeoPixel products)

NEO_RGB Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)

strip.Color(0, 0, RGB2_val)

The three parameters are Red, Green, Blue, three color value components, the range 0~255, synthesis a 24bit value.

strip.setPixelColor(index, strip.Color(R, G, B))

The index parameter represents the serial number of the RGB to light.

strip.setPixelColor(0, strip.Color(RGB1_val, 0, 0))

Light RGB1 to red color.

strip.setPixelColor(1, strip.Color(0, 0, RGB2_val))

Light RGB2 to blue color.

3.2.3 Passive Buzzer

3.2.3.1 Description

The buzzer is an integrated electronic alarm device that uses DC voltage to power electronic products for sound devices.Buzzers are mainly divided into active buzzer and passive buzzer.The main difference between the two type device is as follow :

The ideal signal for active buzzer operation is direct current,usually labeled at VDC, VDD, etc. There is a simple oscillating circuit inside the buzzer. As long as it is energized, it can motivate the molybdenum plate to vibrate. But some active buzzer can work under specific AC signal, while it has high requires of AC signal voltage and frequency, this situation is very rarely .

The passive buzzer does not include internal oscillation circuit, the working signal is a certain frequency of pulse signal .If we give DC signal, the passive buzzer will not work, because the magnetic circuit is constant, the molybdenum sheet cannot vibrate. They are just as shown as follow picture 3.2.9 :

Active Buzzer Passive Buzzer

Picturec 3.2.9 Ative Buzzer and Passive Buzzer

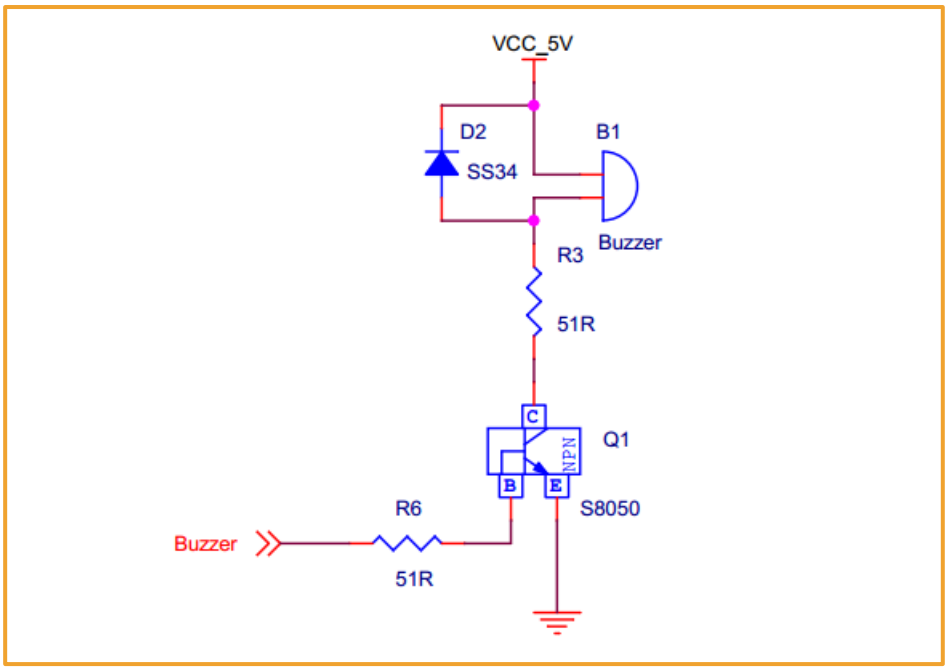
3.2.3.2 Buzzer working principle

The passive buzzer generates music mainly through the I/O port of the single-chip microcomputer to output different pulse signals of different levels to control the buzzer pronunciation.

For example, if Arduino use12MHzs crystal oscillator, to produce middle tune “Re” sound, it needs 587Hzs audio pulse frequency output .The audio signal pulse cycle is T=1/587=1703.5775us, half cycle time is 852us,the pulse timer needs always count at =852us/1us=852,when it count at 852,the I/O port will reverse the direction, then it get the “Re” sounds in C major scale.

In addition to that, the passive buzzer sound principle is the current through the electromagnetic coil, making the electromagnetic coil generated magnetic field to drive the vibration film audible. Therefore,a

certain amount of current is required to drive it, while the ARDUINOI/O pin outputs a lower voltage. Arduino output level can hardly drive the buzzer, so it needs to add an amplifier circuit. And transistor S8050 is used here as an amplification circuit. As shown in picture 3.2.10 is the onboard buzzer schematic diagram:



Picture 3.2.10 Schematic diagram of onboard buzzer

3.2.3.3 Experiment 1:Alarm Test

Experimental purposes:

Make the buzzer simulate the alarm sound

Experimental principle:

The sound starts with the frequency increasing from 200HZ to 800HZ, then stops for a period of time from 800HZ to 200HZ, and loops experimental code location.

“Lesson\ModuleDemo\Buzzer\AlarmSound\AlarmSound.ino”

```
void setup()
{
    pinMode(9,OUTPUT);
}
void loop()
{
    for(int i = 200; i <= 800; i++)   // 200HZ ~ 800HZ
    {
        tone(9,i);
    }
    delay(1000);                  //Max Frequency hold 1s
    for(int i= 800; i >= 200; i--)   // 800HZ ~ 200HZ
    {
        tone(9,i);
        delay(10);
    }
}
```

Firstly,we use a simple procedure to understand how to use the buzzer, and its sound principle. And to drive a buzzer like singing sound,we need make the buzzer issued frequency and duration of the different sound . Cycle is equal to the reciprocal of the frequency, so you can know the time by frequency, and then by calling the delay function or timer to achieve.Similarly the sound duration can also be achieved through the delay function. So the key factor to make the buzzer sing is to know how much time to prolong! Play music with Arduino, you just need to understand the two concepts of "tone" and "beat" .

The tone means the frequency at which a note should be sung.

The beat means how long a note should be sung.

The commonly used method is "look-up table method", this method is complex where you have to find the corresponding frequency of each note (according to the note, the frequency comparison), and then according to the formula converted to the corresponding time (take half cycle), and then through the delay function implementation. Finally by programming to achieve.

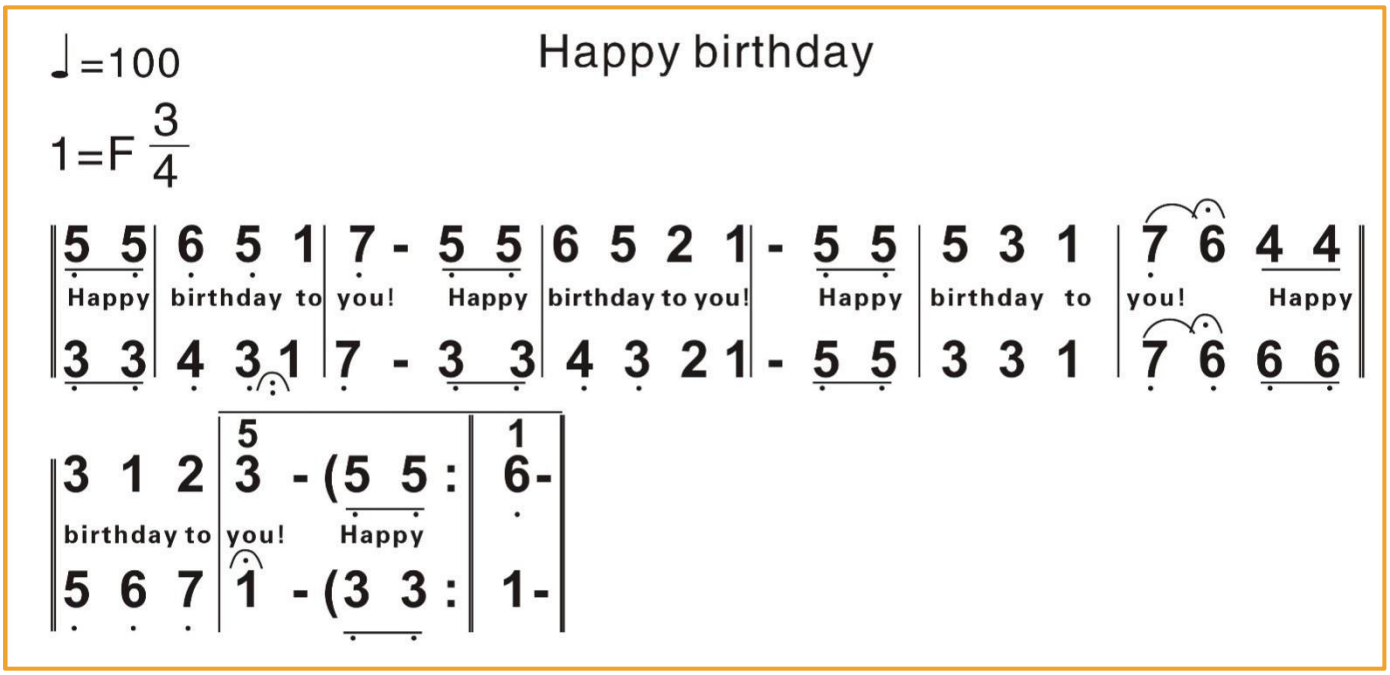
The whole process is like this:

Firstly, according to the score of Happy Birthday song, convert each tone to the corresponding frequency.

For example: picture 3.2.11 is the note frequency conversion table, picture 3.2.12 is the Happy Birthday song score.

	Musical notes	Corresponding frequency (Hz)	Half cycle(us)
Bass	1	261.63	1911.13
	1.5	277.18	1803.86
	2	293.66	1702.62
	2.5	311.13	1607.06
	3	329.63	1516.86
	4	349.23	1431.73
	4.5	369.99	1351.37
	5	392.00	1275.53
	5.5	415.30	1203.94
	6	440.00	1136.36
	6.5	446.16	1120.66
	7	493.88	1012.38
Alto	1	523.25	955.56
	1.5	554.37	901.93
	2	587.33	851.31
	2.5	622.25	803.53
	3	659.26	758.43
	4	698.46	715.86
	4.5	739.99	675.69
	5	783.99	637.76
	5.5	830.61	601.97
	6	880.00	568.18
	6.5	932.33	536.29
	7	987.77	506.19
Treble	1	1046.50	477.78
	1.5	1108.73	450.97
	2	1174.66	425.66
	2.5	1244.51	401.77
	3	1318.51	379.22
	4	1396.91	357.93
	4.5	1479.98	337.84
	5	1567.98	318.88
	5.5	1661.22	300.98
	6	1760.00	284.09
	6.5	1864.66	268.15
	7	1975.53	253.10

Picture 3.2.11 The note frequency conversion table



Picture 3.2.12 The score of Happy birthday song

Firstly , let's learn some knowledge about music score and look at the above music score ,the bass is the one which with point under the number, the normal tone without any point .The treble is the one which with point above the number. The bass of tone 5 is 4.5,the treble is 5.5. Other notes are the corresponding truth. There is a "1=F" on the upper left of music score, while the general music score is C ,it is "1=C".Note, the 1234567(do ,re,mi,fa,so,la,xi,duo) relative is CDEFGAB,not ABCDEFG.So, if the rule is F ,that means 2 is G tone,3 is A tone,……7 is E tone. So, in the situation, the bass 5 corresponds to bass 1.5,the tone need to move to the right or left. If you still don't understand, look at the following:

1 originally corresponding should be c,4 originally should correspond to f.

Then now 1 corresponds to F, which corresponds to 4, then 1.5 corresponds to 4.5, 2 corresponds to 5.

So, bass 5 is actually 4.5, so half cycle is 1803μs.

As to it is based on half-cycle calculations, because the single-chip microcomputer makes a sound by looping resetting the port connected to the buzzer, so it is a half-cycle. Because our product is passive buzzer, the active buzzer is full cycle.

Then according to the above reason, converse the tone one by one, achieve it by delay function. Because the frequency of the conversion of each note is different, you need to using multiple delay functions to achieve accurate tone frequencies one by one. But this is too complicated, and the microcontroller itself is not specifically to sing. The delay function has almost the same frequency in order to adapt to each tone, you need to calculate it by yourself, and different songs have different values, so this is the more troublesome issue.

After we know the frequency of the pitch, the next step is to control the playing time of the notes. Each note plays for a certain amount of time so that it can be a beautiful piece of music. The rhythm of notes is

divided into one beat, half beat, 1/4 beat, and 1/8 beat. We stipulate that the time of a clap of notes is 1, half a beat for the 0.5;1/4 Pat for the 0.25;1/8 0.125 ...

Firstly,ordinary notes occupy for 1 shots.

Secondly, underlined notes indicate 0.5 beats; two underlines are quarter beats (0.25)

Thirdly, the notes which followed by a point, which means more 0.5 beats, that is 1+0.5.

Fourthly, the notes followed by a "-", which means more than one beat, that is, 1 +1.

So we can give this beat to each note and play it out. As for the conversion of beats to frequency, it also has corresponding list, just follow table two:

Music beat	1/4 beat delay time	Music	1/8 beat delay time
4/4	125ms	4/4	62ms
3/4	187ms	3/4	94ms
2/4	250ms	2/4	125ms

Table 2: Beat and frequency correspondence table

It is also achieved through the delay function,of course there will be errors. The idea of programming is very simple, firstly convert the note frequency and the time you want to sing into the two arrays. Then in the main programming, through the delay function to reach the corresponding frequency . sing it over, stop for a while, and then sing it, all the conversion is complete, we get the following frequency (Table 3) and beat:

Do 262	Re 294	Mi 330	Fa 349	Sol 392	La 440	Si 494	Do_h 523
Si_h 988	Mi_h 659		La_h 880	Sol_h 784	Fa_h698	Re_h 587	

Table 3: Happy Birthday Song beat table

According to the music score, we can get the frequency of the birthday song.

Sol,Sol,La,Sol,Do_h,Si,Sol,Sol,La,Sol,Re_h,Do_h,Sol,Sol,Sol_h,Mi_h,Do_h,Si,La,Fa_h,Fa_h,Mi_h,Do_h,Re_h,Do_h float

The beat is as follows:

0.5,0.5,1,1,1,1+1,0.5,0.5,1,1,1,1+1,0.5,0.5,1,1,1,1,1,0.5,0.5,1,1,1,1+1,

Add beats and frequency to the program and download it to Arduino to play.

Happy Birthday music score beat, view table two rhythm and frequency corresponding table 1 beats time is 187*4 = 748ms

Note: The procedure is shown in the: "Lesson\Advanced Experiment\Happy_Birthday \Happy_Birthday.ino"

```
#define Do 262
#define Re 294
#define Mi 330
#define Fa 349
#define Sol 392
#define La 440
#define Si 494
#define Do_h 523
#define Re_h 587
#define Mi_h 659
#define Fa_h 698
#define Sol_h 784
#define La_h 880
#define Si_h 988
#include "RGBLed.h"

RGBLed rgbled_A3(7,A3);
int buzzer = 9;   // buzzer pin 9
int length;

// happy birthday Music score
int scale[] = {Sol, Sol, La, Sol, Do_h, Si, Sol, Sol,
               La, Sol, Re_h, Do_h, Sol, Sol, Sol_h, Mi_h,
               Do_h, Si, La, Fa_h, Fa_h, Mi_h, Do_h, Re_h, Do_h };
// Beats time

float durt[]=
{
    0.5, 0.5, 1, 1, 1, 1+1, 0.5, 0.5,
    1, 1, 1, 1+1, 0.5, 0.5, 1, 1,
    1, 1, 1, 0.5, 0.5, 1, 1, 1, 1+1
};
```

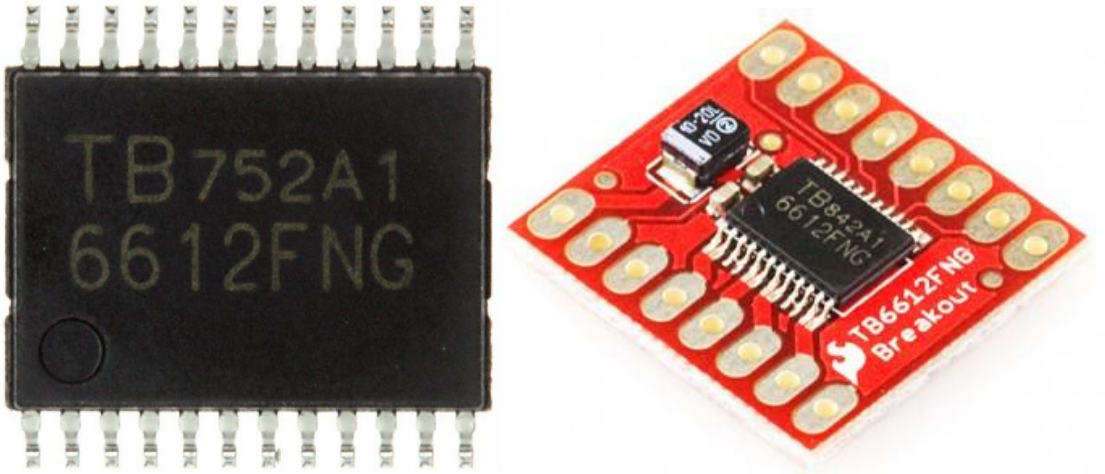
```
void setup()
{
    pinMode(buzzer, OUTPUT);
    // get scale length
    length = sizeof(scale) / sizeof(scale[0]);
    Serial.begin(9600);
}

void loop()
{
    for(int x = 0; x < length; x++) {
        // Serial.println(scale[x]);
        tone(buzzer, scale[x]);
        rgbled_A3.setColor(0, scale[x] - 425, scale[x] - 500, scale[x] - 95);
        rgbled_A3.show();
        // 1= 3/4F so one Beats is  187*4 = 748ms
        delay(748 * durt[x]);
        noTone(buzzer);
    }
    delay(3000);
}
```

3.2.4 TB6612FNG Drive Principle

3.2.4.1 TB6612FNG Introduction

TB6612FNG is a new type of DC motor drive device produced by Toshiba Semiconductor Corporation. It is much more efficient than the traditional L298N, and its volume is also greatly reduced. Within the rated range, the chip basically does not generate heat, and of course it becomes even more delicate . The most important is that has the very high integration level, Independent two-way control of 2 DC motors, so in the integrated, miniaturized motor control system, it can be used as an ideal motor drive components.

Tb6612FNG has a large current MOSFET-H bridge structure, dual channel circuit output, each channel output highest 1.2 A continuous drive current, start peak current up to 2A/3.2A (continuous pulse/single pulse); 4 motor control modes: forward/reverse/brake/stop; Standby state; PWM support frequency of up to kHz, in-chip low-voltage detection circuit and thermal shutdown protection circuit, a common package for SSOP24 small patches, as shown in picture 3.2.13



Picture 3.2.13 TB6612 Physical drawings and common module diagrams

Tb6612fng main pin function: ainl/ain2, bin1/bin2, PWMA/PWMB for the control signal input terminal; AO1/A02, B01/B02 for 2-way motor control output end; STBY control pins for normal work/standby state; The VM (4.5~15 V) and VCC (2.7~5.5 V) are motor-driven voltage inputs and logical level inputs respectively.

In addition, TB6612FNG is a MOSFET based H-Bridge integrated circuit, which is more efficient than the transistor H-bridge drive. The output load capacity of L293D is increased by one-fold compared to the drive current of the average 600MA in each channel and the pulse peak current of 1.2 A. Compared with the L298N heat consumption and the peripheral diode continuous circuit, it does not need to add a heatsink, the peripheral circuit is simple, only the external power filter capacitor can directly drive the motor, to reduce the system size.

For the PWM signal, it supports up to 100kHz frequency, relative to the above 2 chip 5 kHz and 40kHz also has a greater advantage.

3.2.4.2 Motor Control Unit Design

Unit Hardware Composition

Picture 3.2.14 shows the pulse and voltage diagram of the TB6612FNG. SCM Timer generates 4-channel PWM output as AIN1/AIN2 and BIN1/BIN2 control signals, A01 and A02, B01 and B02 control of Motor M1 and M2 in the picture. Using the timer output hardware PWM pulse, the SCM CPU only in the change of PWM duty ratio, the participation operation, greatly reducing the system operation burden and PWM software programming overhead. The input pins PWMA, PWMB and Stby are controlled by the I/0 level of the motor or the braking state as well as the device working status. The circuit uses voltage-V-10μf electrolytic capacitors and 0.1μf capacitors for power filtering, using power MOSFET to provide power reversal protection for VM and VCC.

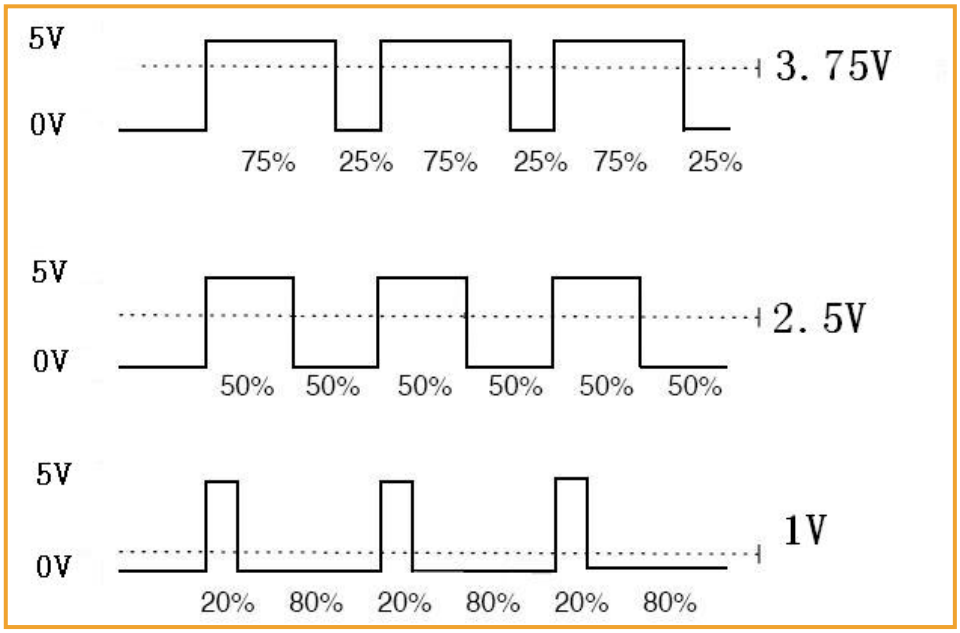
Software Control Software Implementation

Pulse width modulation generates the PWM signal of duty ratio change, and realizes the speed control of the motor by fast switching of the output state of the drive. The size of the PWM duty ratio determines the average output voltage, and then determines the speed of the motor. In general, single polarity and constant frequency PWM modulation method are adopted to ensure the stability of motor speed control. TB6612FNG logical truth-table is shown in tables three. When the device is working, the STBY pin is set to a high level, the IN1 and IN2 are unchanged, the input signal of the PWM pin can be controlled by the motor one-way speed, the PWM pin is a high level, and the input signal of IN1 and IN2 can be controlled by the bidirectional speed of the motor. The control logic of the A and B channels in the table is the same.

Input				Output		
IN1	IN2	PWM	STBY	01	02	Mode status
Hs	H	H/L	H	L	L	brake
L	H	H	H	L	H	Reverse
L	H	L	H	L	L	brake
H	L	H	H	H	L	Forward
H	L	L	H	L	L	brake
L	L	H	H	off		stop
H/L	H/L	H/L	L	off		Standby

Table three TB6612FNG logical truth tables

In Arduino, can not output analog voltage, can only output 0 or 5V of digital voltage value, we use the high resolution counter, using the square wave of the duty ratio is modulated to a specific analog signal level coding. The PWM signal is still digital, because at any given moment, the full amplitude of the DC power supply is either 5V (on) or 0V (off). The voltage or current source is added to the simulated load by a repetitive pulse sequence with a pass (on) or a break (off). When the pass is the DC power is added to the load, when the break is the power is disconnected. As long as the bandwidth is sufficient, any analog value can be encoded using PWM. The output voltage value is calculated through the time of the pass and break. Output voltage = (connect time/pulse time) * Maximum voltage value, picture 3.2.8 for pulse change corresponding voltage.



Picture 3.2.14 Pulse and Voltage diagram

The Arduino Uno has 6 PWM interfaces, namely the Digital Interface 3, 5, 6, 9, 10, 11. To ensure that the expansion board at the same time to support a variety of cars, here we choose 5, 6 (Timer 2) as the motor PWM control io, detailed IO definition as shown in Figure 3.2.15, can also refer to the supporting information in the **" Schematic\motor_driver.pdf "**.

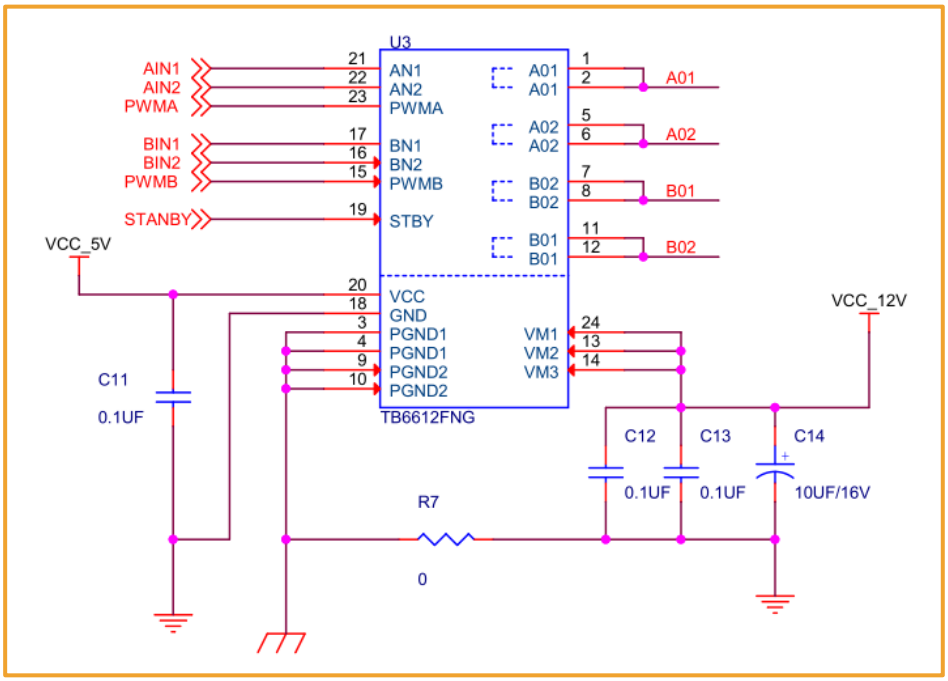


Fig. 3.2.15 TB6612FNG Application Circuit diagram

3.2.4.3 Motor Test Procedure

After knowing about these knowledge, we can control the motor, but before the control, we had better test whether the motor is working properly, the best way is to write a test program, download the program to Arduino, to observe whether the motor in accordance with our expectations set speed and direction of rotation. In the supporting information, we provide a small section of the Motor test program (file name **"Lesson\ModuleDemo\MotorTest\MotorTest.ino"**), of course, you can also write your own program. It's very simple,when we download the program to Arduino, we can see the motor: The story two seconds-----reverse two seconds-----stop two seconds.If it cycles like this all the time,indicating that the motor is working properly.

Then you can easily control the car by matching the other modules together.

The test procedure is very simple as follows

```
int motor_L2 = 4; //Bin1
int motor_L1 = 2; //Bin2
int PWMB = 6; //PWMB
int STBY = 7; //STBY
void setup()
{
    Serial.begin(9600);
    pinMode(STBY, OUTPUT);
    pinMode(motor_L1, OUTPUT);
    pinMode(motor_L2, OUTPUT);
}

void loop()
{
    digitalWrite(motor_L2,HIGH);
    digitalWrite(motor_L1, LOW);
    digitalWrite(STBY,HIGH);
    analogWrite(PWMB,255);
    delay(2000);
    digitalWrite(motor_L2,LOW);
    digitalWrite(motor_L1, HIGH);
    digitalWrite(STBY,HIGH);
    analogWrite(PWMB,255);
    delay(2000);
    digitalWrite(motor_L2,LOW);
    digitalWrite(motor_L1, HIGH);
    digitalWrite(STBY,HIGH);
    analogWrite(PWMB,0);
    delay(2000);
}
```

3.2.5 Infrared Tracing

3.2.5.1 Introduction of infrared tracing sensor

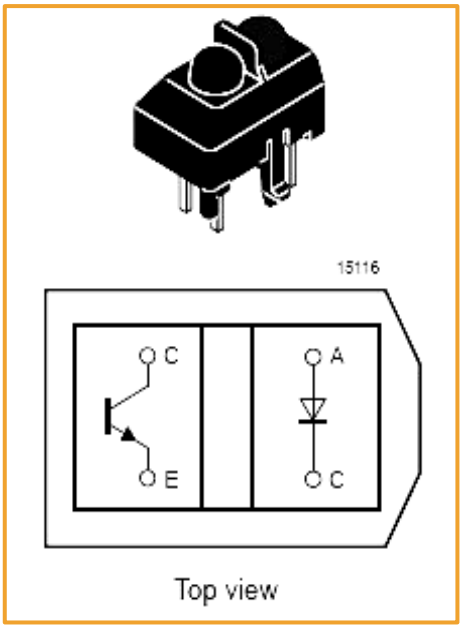
Note: When you are studying this section, you must pay attention to the black line or the white line you want to look for. If you experiment with the environment where the floor is black, then should seek white line (that is, put the white track on the floor); If you experiment with an environment where the floor is white, then you should look for black lines (on the floor) , in short, to make the circuit and other environment in a sharp difference.

Traces of the black line refers to the car on the white floor along the black lines, because the black and white floor of the light reflected coefficient is different, we can judge the "road."according to the intensity of reflected light was received.

White line tracking refers to the car on the black floor along the white line, because the white line and the black floor of the light reflection coefficient, you can receive the reflected light strength to judge the "road."

In the "aurora-racing" car, we use the 5-way TCRT5000 sensor as a tracing module, TCRT5000 infrared reflection sensor, is a set of transmitter and receiver in one photoelectric sensor, which consists of an infrared light-emitting diode and an n-IR receiver tube. Detection of reflective distance for 1mm-25mm, and the sensor sets up M3 as fixed installation hole, adjust the direction and fixed easy to use, using 74HC14 Schmitt trigger inverter, strong driving ability. And it can detect the black line in white background and detect white lines in the dark bottom. The module is shown in picture 3.2.16.

Picture 3.2.16 Physical part of the module



Picture 3.2.17 TCRT5000 Schematic diagram

3.2.5.2 Working Principle

In the above, we talk about seeking white lines or black lines of two trace patterns, in fact, whether it is to seek black or white line, we usually take the infrared detection method.

Infrared detection, that is using the infrared rays in different colors of the surface of the object has different characteristics of reflection, in the course of the car in the process of continuous emission of infrared light, when the emission of infrared radiation has not been reflected back or reflected back but not strong enough, the infrared receiver has been in the shutdown state, the module output end is low level , indicating that the diode has been in the off state; when the infrared light encounters the white paper floor when diffuse reflection, reflected light is installed in the car receiving tube received, and infrared is reflected back and strong enough, photosensitive transistor saturation, at this time the output of the module is high level, indicating diode is lit. Arduino is based on the reflected back of the infrared light to determine the location of the black line and the path of the car. However, infrared detection also has a lot of deficiencies, such as: The distance is limited, vulnerable to other light interference. The infrared detection module consists of a sending module and a receiving module, which can be used to detect the change of the state of the object by transmitting the infrared signal and judging the change of the receiving signal.

As shown in the schematic diagram 3.2.17 A, C, between the light-emitting diodes, C, E for the receiving diode, in the "aurora-racing" car, we used 5 tracks, the left and right sides of each of the two, the middle one, its installation as shown in picture 3.2.18,

Picture 3.2.18 Tracing module installation schematic

Among them Left1 and Right1 are the first-level directional control sensors, Left2 and Right2 are the second-level directional control sensors, and when the car advances, it always remains between the two first-level sensors of Left1 and Right1, when the car deviates from the black line:

If the Left1 detected black line, the Right1, Right2, Left2 and the Center sensor did not detect the black line, indicating that the car has a small offset to the left, then you need to adjust the car to the left a little rotation, keep the center sensor has been detecting black lines, and along the black line h as been walking

If the Right1 detect black line, the Left1, Left2, Right1 and the Center sensor did not detect the black line, indicating that the car has a small offset to the right, then you need to adjust the car slightly to the right rotation, keep the center sensor has been detecting black lines, and along the black line has been walking.

If the Left2 detected black line, the Right1, Right2, Left1 and the Center sensor did not detect the black line, indicating that the car has shifted significantly to the left, then you need to adjust the car to the left large rotation, keep the center sensor has been detecting black lines, and along the black line has been walking.

If the Right2 detected the black line, Left1, Left2, Right1 and the Center sensor did not detect the black line, indicating that the car has a generous shift to the right, then you need to adjust the car to the right large rotation, keep the center sensor has been detecting black lines, and along the black line has been walking;

If the car returns to the track, it will always drive along the black line. At this time, Left1, Left2 and Right1, Right2 all detect the white line and send a high level to the microcontroller.

3.2.5.3 Module parameters

- Adopt TCRT5000 infrared reflection sensor
- Detection distance: 1mm~25mm is suitable, focus distance is 2.5mm
- Comparator output, clean signal, good waveform, strong driving ability, more than 15mA
- Working voltage: 3.3V-5V
- Use wide voltage 74HC14D comparator, digital switch output(0 and 1)
- With a fixed bolt hole for easy installation.

Notice:

Ensure that the wiring is correct,the board might be destroyed if you have reverse connection between negative and positive poles.

For more parameters, please view the **"Document\TCRT5000.pdf"** file in the disc.

1、Install the sensor on the car and connect it to the expansion board with a wire (the assembly part has been completed)

Infrared Tracking Module	Arduino Uno
OUT1	A0
OUT2	A1
OUT3	A2
OUT4	A4
OUT5	A5

2、Make a track, if the floor of the experimental environment is white, paste a black tape that forms a circuit on the floor, otherwise paste white. As for the shape of the track, everyone can follow their own ideas. The best width of the tape is 31 to 35mm. In this tutorial, we are using black line track, as shown in Figure 3.2.13.

3、Module test, copy the following program to the IDE's compilation environment (or you can directly open the matching program on the CD), and download to the Development Board, open the "Serial monitor" (baud rate is 9600), observe no black lines (figure 3.2.20) and detect white line (Picture 3.2.21) when the data changes.

Instruction: Because the car using the steering gear as the steering drive, the curvature of the track should not be too large, otherwise the car can not complete the track turn.

```
int left1,centre,right1,left2,right2;
void setup()
{
    Serial.begin(9600);
    pinMode(A0, INPUT);
    pinMode(A1, INPUT);
    pinMode(A2, INPUT);
    pinMode(A4, INPUT);
    pinMode(A5, INPUT);
}

void loop()
{
  left1=analogRead(A0);
  left2=analogRead(A1);
  center=analogRead(A2);
  right1=analogRead(A4);
  right2=analogRead(A5);
  Serial.print("The left1 is:");
  Serial.print(left1);
  Serial.print("    ");
  Serial.print("The left2 is:");
  Serial.print(left2);
  Serial.print("    ");
  Serial.print("The center is:");
  Serial.print(centre);
  Serial.print("    ");
  Serial.print("The right1 is:");
  Serial.print(right1);
  Serial.print("    ");
  Serial.print("The right2 is:");
  Serial.println(right2);
  delay(100);
}
```

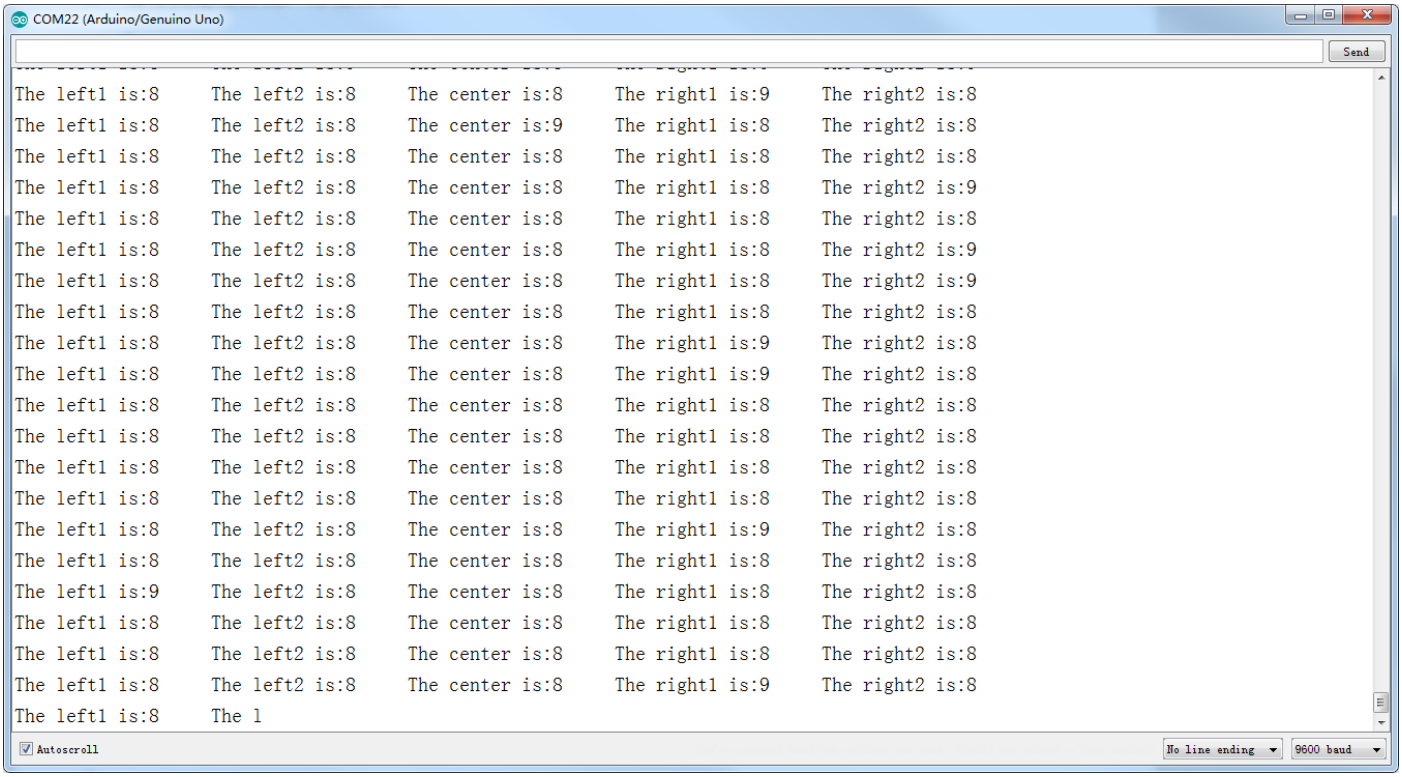
Picture.3.2.19 Black line circuit schematic

COM22 (Arduino/Genuino Uno)

```
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:973   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975   The center is:974   The right1 is:974   The right2 is:975
The left1 is:975   The left2 is:975
```

Autoscroll | No line ending | 9600 baud

Picture 3.2.20 Data when the black line is not detected by the sensor



Picture 3.2.21 Data when the sensor detects a black line

From picture 3.2.20 and 3.2.21 we can see that when the sensor does not detect the black line, the output is high,when it detects the black line,the output is low.Because using the analog port to collect sensor signals, the value of the serial printing is also analog. That is, high electricity at ordinary times to nearly 1024, low electricity at ordinary times for 0.After we having mastered the principle of the sensor,the trace of this section of the car can come to an ending.And then let's start the journey of the car together.

3.2.5.4 Software Design

1. Track principle of the racing car

After the car is powered on, it enters the trace mode, and the Arduino starts scanning the single-chip I/O port connected to the sensor. Once a signal change is detected on an I/O port, the Arduino will execute the corresponding judgment program:

When the second sensor on the left side of the car detects a black line, none of the other sensors detect a black line, indicating that the car has shifted slightly to the left. At this time, the front wheel of the car should rotate to a small angle to the left.When the middle sensor detects the black line, the front wheel returns positive and walks along the black line while continuing to scan the single-chip I/O port connected to the sensor.

When the second sensor on the right side of the car detects the black line, none of the other sensors detect the black line, indicating that the car has shifted slightly to the right. At this time, the front wheel of the car should rotate to the right with a small angle.When the middle sensor detects the black line, the front wheel

returns positive and walks along the black line while continuing to scan the single-chip I/O port connected to the sensor.

When the first sensor on the left side of the car detects the black line, none of the other sensors detect the black line, indicating that the car has been shifted to the left by a large amount. At this time, the front wheel of the car should be rotated to the left by a relatively large angle. During this process, the second sensor on the left will also detect the black line, so the car will reduce the angle of the front wheel to the left and the speed of the rear wheel. When the middle sensor detects the black line, the front wheel will returns positive and move forward along the black line. Walk while continuing to scan the single-chip I/O port connected to the sensor.

When the first sensor on the right side of the car detects the black line, no other sensor detects the black line, indicating that the car has shifted to the right by a large amount. At this time, the front wheel of the car should be rotated to the right by a large angle. During the process, the second sensor on the right will also detect the black line, so the car will reduce the angle of the front wheel to the right and the speed of the rear wheel. When the middle sensor detects the black line, the front wheel will returns positive and continue to walk forward, and continue to detect the black line to repeat the above action.

```
#include "Servo.h"// Because the steering wheel of the car is driven by a rudder, the program must contain the corresponding library file.
int E1 = 4; //Bin1
int M1 = 2; //Bin2****** The car only uses a DC motor drive, so the TB6612 drive chip can only be used all the way。
int E2 = 6; // Defines the PWMB interface as 6.
int M2 = 7; // The STBY interface is defined as 7.
int servo_pin = 5;// The steering gear is defined as No 5, and the rudder wire is inserted directly into the expansion board to the corresponding position.
Servo ForwardServo;// Create a control object for the steering wheel
void setup()
{
    Serial.begin(9600);
    pinMode(M1, OUTPUT);
    pinMode(M2, OUTPUT);
    pinMode(E1, OUTPUT); // Initialization of motor-driven IO for output mode
    ForwardServo.attach(servo_pin);
    ForwardServo.write(90);// The initial steering gear is 90°, and the center position.
}
void loop()
{
    int left1,centre,right1,left2,right2;// Defines the 5-way sensor variable name
    left1=analogRead(A0);
    left2=analogRead(A1);
    centre=analogRead(A2);
    right1=analogRead(A4);
    right2=analogRead(A5);//read five sensor
```

```
    if ((right1 >= 975)&&(right2 >= 975)&&(centre <= 8)&&(left2 >= 975)&&(left1 >= 975))
{
       //*******forward*******//
        ForwardServo.write(90);// The steering gear stays 90° when driving forward.
        int val=80;// PWM value, 2--255, the greater the value, the faster the car in
the track, so we recommend that the speed is not too fast
        digitalWrite(E1,HIGH);
        digitalWrite(M1, LOW);
        analogWrite(E2,val);
        digitalWrite(M2,HIGH);// High level normal work
    } else if ((right1 <= 8)&&(centre >= 975)&&(left1 >= 975)&&(left2 >= 975)&&(right2 >=
975)) {
        //***turn right1***//
        ForwardServo.write(125);// Rotate right from 90°to 120°and rotate the wheel 30 °
to the right.
        int val=70;
        digitalWrite(E1,HIGH);
        digitalWrite(M1, LOW);
        analogWrite(E2,val);
        digitalWrite(M2,HIGH); // High level normal work
   } else if((right1 >= 975)&&(right2 >= 975)&&(centre >= 975)&&(left1 <= 8)&&(left2 >=
975)) {
       //***Turn left1***//
        ForwardServo.write(45);// Rotate left from 90°to 40°and rotate the wheel 50 °
to the left.。
        int val=70;
        digitalWrite(E1,HIGH);
        digitalWrite(M1, LOW);
        analogWrite(E2,val);
        digitalWrite(M2,HIGH); // High level normal work
    }
```

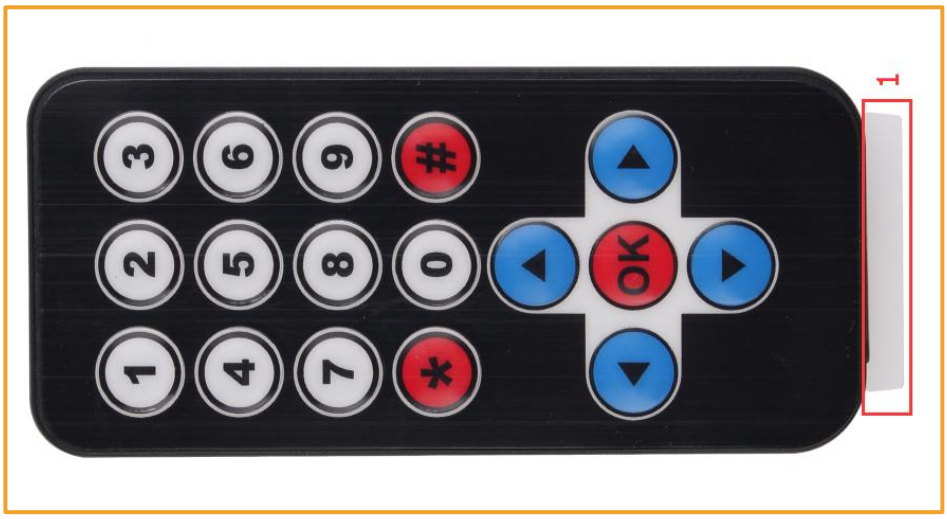
```
else if((right2 >= 975)&&(centre >= 975)&&(left2 <= 8)&&(right1= 975)&&(left1 >= 975))
{
        //***Turn left2***//
        ForwardServo.write(50);// Rotate left from 90°to 50°and rotate the wheel 40 °
to the left.。
        int val=70;
        digitalWrite(E1,HIGH);
        digitalWrite(M1, LOW);
    } else  if((right2 <= 8)&&(centre >= 975)&&(left2 >= 975)&&(left1 >= 975)&&(right1 >=
975)) {
     //***turn right2***//
        ForwardServo.write(120);// Rotate right from 90°to 105°and rotate the wheel 15 °
to the right.
        int val=70;
        digitalWrite(E1,HIGH);
        digitalWrite(M1, LOW);
        analogWrite(E2,val);
        digitalWrite(M2,HIGH); // High level normal work
   } else if((centre <= 8)&&(left2 <= 8)&&(right2 <= 8)&&(left1 <= 8)&&(right1 <= 8)){
         //*******Stop*******//
        ForwardServo.write(90);// When stopped, let the steering gear return to its
original position for easy execution of the next action.
        int val=0;
        digitalWrite(E1,HIGH);
        digitalWrite(M1, LOW);
        analogWrite(E2,val);
        digitalWrite(M2,LOW); // The low level is on standby, at which point the car
stops.
   }
}
```

Note: There may produce some errors when assembly the front wheels , this will result in a different rotation angle for each vehicle, so the program of the steering wheel rotation angle value just for reference, we may need to modify the angle value appropriately in the actual application.

3.2.6 Infrared Remote Control

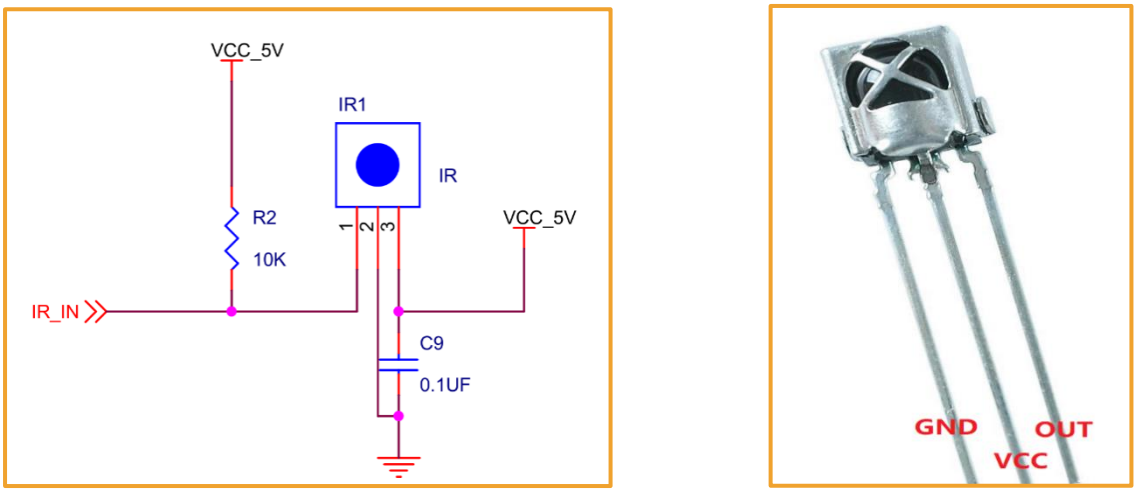
3.2.6.1 Introduction

Infrared remote control is widely used in every field at present.Infrared wireless remote control consists of Mini ultra-thin infrared remote controller (physical map shown in the picture 3.2.22) and integrated 38KHz infrared receiver. Mini ultra-thin infrared remote controller has 17 function keys, and the launch distance is up to 8 meters. Suitable for indoor control of various devices.



Picture 3.2.22 Infrared remote Control physical map

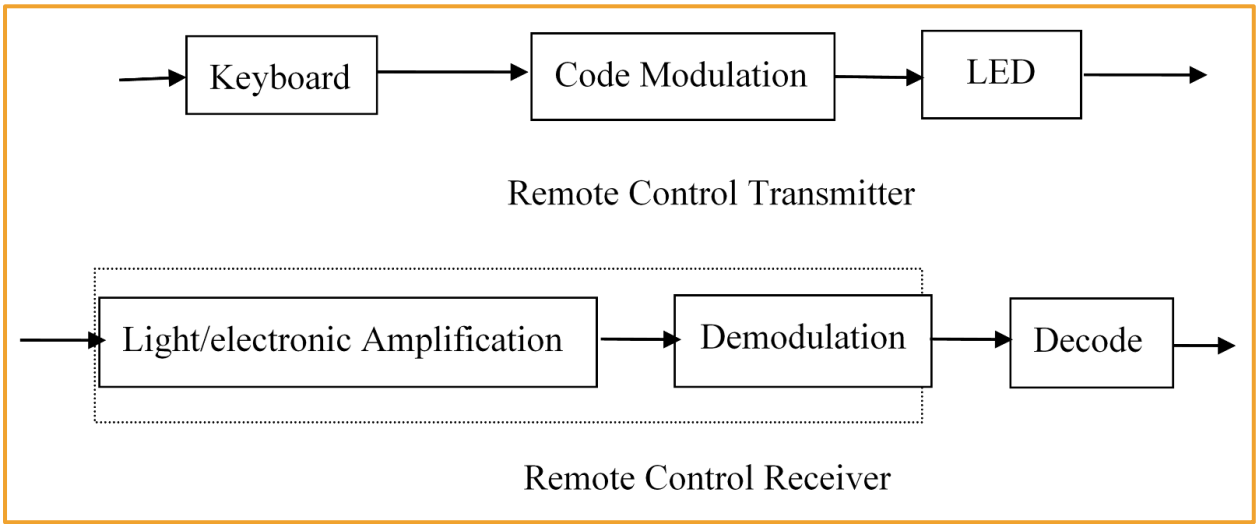
In the "Aurora-Racing" car, integrated IR receiver head has been added to the expansion board, just need to plug the expansion board into the Arduino, and in the program defined pins (8th number IO), the IR receiver head has three pins, including power supply feet, grounding and signal output feet. The circuit is shown in the picture 3.2.23. The ceramic capacitor 0.1uf is a decoupling capacitor, which filters out the interference from the output signal. The 1-terminal is the output of the demodulation signal, which is directly connected with the Arduino 8th of the single-chip microcomputer. When the infrared coded signal is emitted, the output of the square wave signal after the infrared joint is processed, and is provided directly to the Single-chip microcomputer, and the corresponding operation is carried out to achieve the purpose of controlling the motor.



Picture.3.2.23 Infrared receiver Head circuit diagram and physical map

3.2.6.2 Working Principle

Remote control system is generally composed of remote control (transmitter), receiver, when you press any button on the remote control, the remote will produce the corresponding coded pulse, output a variety of infrared as the medium of control pulse signal, these pulses are computer instruction code, infrared monitoring diode monitoring to infrared signals, The signal is then sent to the amplifier and the limiter, which controls the pulse amplitude at a certain level, regardless of the distance between the IR transmitter and the receiver. The AC signal enters the bandpass filter, the bandpass filter can pass the load wave of 30KHZ to 60KHZ, through the demodulation circuit and the integral circuit to enter the comparator, the comparator outputs the high and low level, restores the signal waveform of the transmitting end. As shown in the 3.2.24.

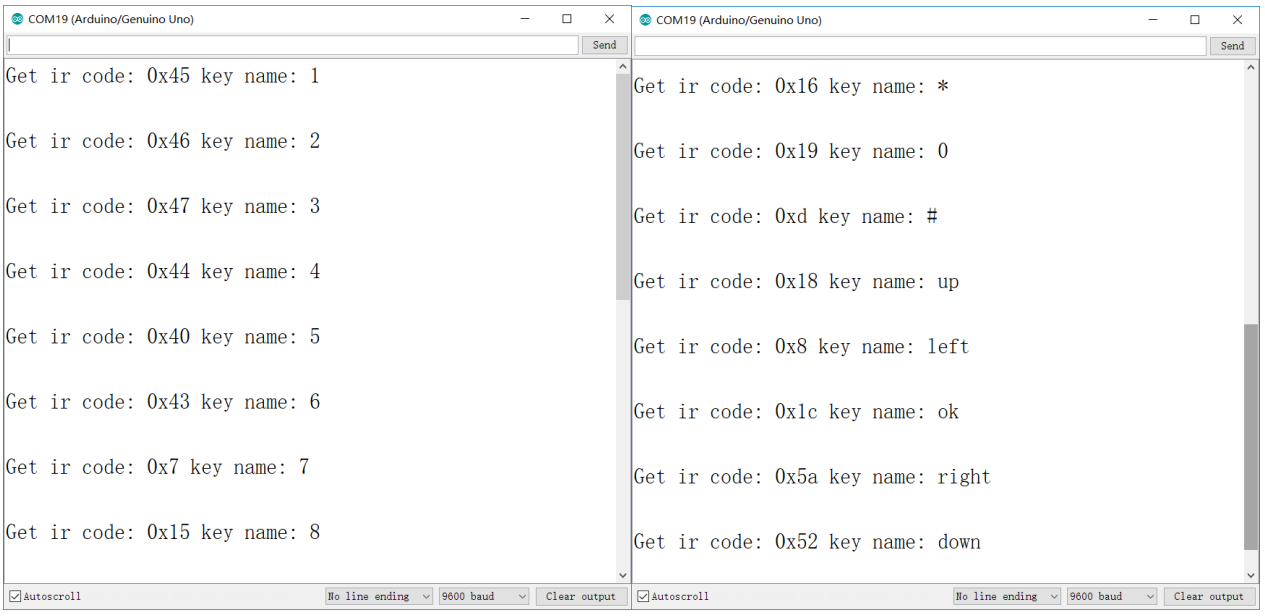


Picture.3.2.24 Infrared emitter and receiver system block diagram

3.2.6.3 Acquiring Infrared remote value

Open the program named “Lesson\ModuleDemo\IrkeyPressed\IrkeyPressed.ino” in the file and download it to the development board. Unplug the transparent plastic piece marked “1” in Figure 3.2.22. Then open the "serial monitor" and use the remote control to align the receiver head and press any key to observe the value displayed in the "serial monitor" and record it for later development as shown in the picture.

```
#include "IRremote.h"
IRremote ir(8);
unsigned char keycode;
char str[128];
void setup() {
    Serial.begin(9600);
    ir.begin();
}
void loop()
{
    if (keycode = ir.getCode()) {
        String key_name = ir.getKeyMap(keycode);
        sprintf(str, "Get ir code: 0x%x key name: %s \n", keycode, (char
*)key_name.c_str());
        Serial.println(str);
    } else {
      //   Serial.println("no key");
    }
    delay(110);
}
```



Picture 3.2.25 Remote coded query

In the picture 3.2.25, we can look at the IR code "0x45" and KeyName "1" two values, where "0x45" is a remote control key code, "1" is the Remote control button function name. Matching remote control all encoded values in Lesson\ModuleDemo\IrkeyPressed\Keymap.cpp.

3.2.6.4 Infrared Remote Control Procedure

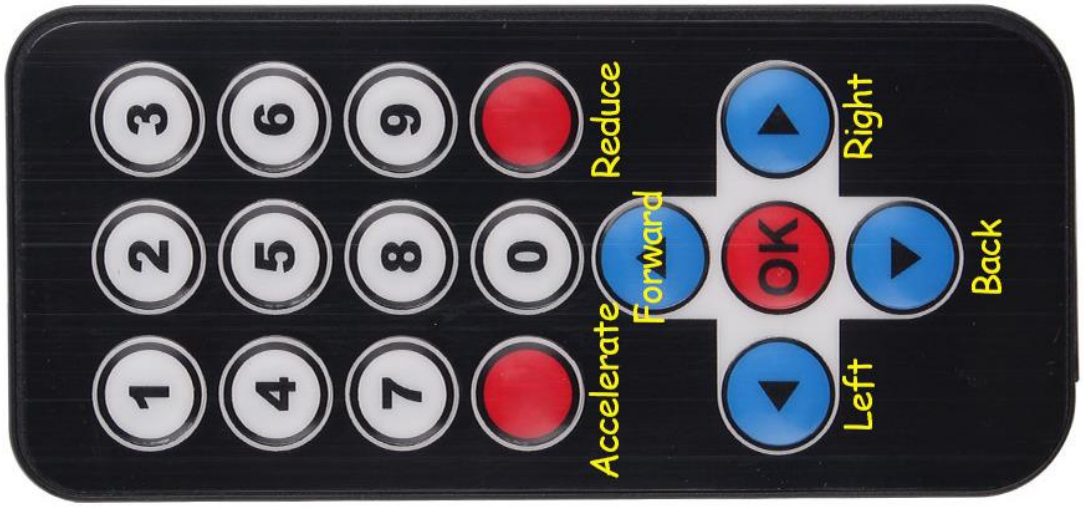
Program Location:Lesson\Advanced Experiment\IRremote\IRremote.ino

```
#include "Buzzer.h"
#include "RGBLed.h"
#include "Servo.h"
#include "IRremote.h"
#include "Keymap.h"
#define MOTOR_L2   4  //Bin1
#define MOTOR_L1   2  //Bin2
#define PWMB   6      //PWMB
#define STBY   7      //STBY
#define SERVO_PIN  5
Servo ForwardServo;
Buzzer buzzer(9);
IRremote ir(8);
RGBLed rgbled_A3(7,A3);
double Speed = 100;
int BaseDegree = 100;
int i = 40;
unsigned char keycode;
void setup()
{
    Serial.begin(9600);
    ir.begin();
    pinMode(STBY, OUTPUT);
    pinMode(MOTOR_L1, OUTPUT);
    pinMode(MOTOR_L2, OUTPUT);
    ForwardServo.attach(SERVO_PIN);
    ForwardServo.write(90);
}
void move(int direction )
{
     if (direction == 1) {
         digitalWrite(MOTOR_L2,HIGH);
         digitalWrite(MOTOR_L1, LOW); //GoForward
         digitalWrite(STBY,HIGH);
     }
```

```
    else if(direction == 2) {
        digitalWrite(MOTOR_L2,LOW);
        digitalWrite(MOTOR_L1,HIGH); // GoBack
        digitalWrite(STBY,HIGH);
    }
    else if(direction == 3) {
      digitalWrite(MOTOR_L2,LOW);
      digitalWrite(MOTOR_L1,HIGH); //stop
      digitalWrite(STBY,HIGH);
  }
}

void loop()
{
  if (keycode = ir.getCode()) {
    Serial.println(keycode, HEX);
      switch((E_IR_KEYCODE)ir.getIrKey(keycode)) {
       case  IR_KEYCODE_UP :
            Serial.println("up");
            ForwardServo.write(BaseDegree);
            move(1);
            analogWrite(PWMB,Speed);
            rgbled_A3.setColor(0,0,Speed-35,0);
            rgbled_A3.show();
            break;
       case IR_KEYCODE_DOWN :
            ForwardServo.write(BaseDegree);
            move(2);
            analogWrite(PWMB,Speed);
            rgbled_A3.setColor(0,Speed-35,Speed-35,Speed-35);
            rgbled_A3.show();
            break;
       case IR_KEYCODE_LEFT :
            ForwardServo.write(BaseDegree - 50);
            move(1);
            analogWrite(PWMB,Speed);
            rgbled_A3.setColor(2,25,15,0);
            rgbled_A3.setColor(1,0,0,0);
            rgbled_A3.show();
            break;
```

```
        case IR_KEYCODE_RIGHT :
            ForwardServo.write(BaseDegree + 40);
            move(1);
            analogWrite(PWMB,Speed);
            rgbled_A3.setColor(1,25,15,0);
            rgbled_A3.setColor(2,0,0,0);
            rgbled_A3.show();
            break;
        case IR_KEYCODE_STAR  :
            i = i + 10;
            if ( i >= 255) {
              i=255;
              buzzer.tone(2000, i);
            }
            Speed = i;
            Serial.println(Speed);//The hexadecimal line feed output code
            buzzer.tone(3000, i);
            break;
        case IR_KEYCODE_POUND :
             i = i - 10;
             if (i <= 40) {
                 i=40;
                 buzzer.tone(2000, i);
             }
             Speed = i;
             Serial.println(Speed);//The hexadecimal line feed output code
             buzzer.tone(3000, i);
             break;
        }
    } else {
        Serial.println("stop");
        ForwardServo.write(BaseDegree);
        move(3);
        analogWrite(PWMB,0);
        rgbled_A3.setColor(0,Speed-35,0,0);
        rgbled_A3.show();
    }
    delay(100);
}
```

The above programs are infrared remote control reference routines, we can open the supporting program and download to the Development Board, according to the picture 3.2.26 to use remote control.



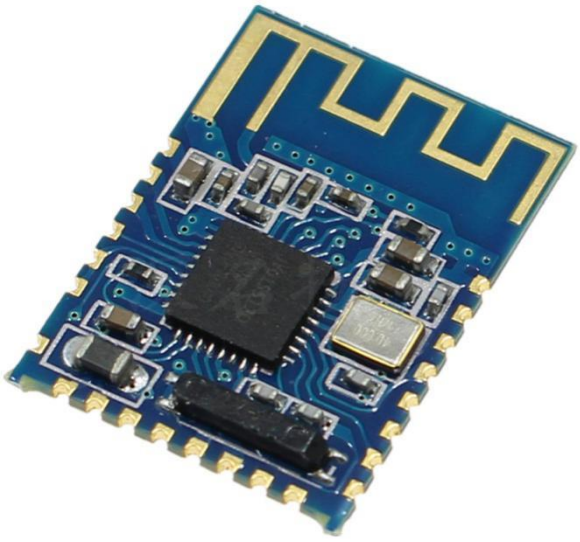
Picture 3.2.26 Infrared remote Control use instructions

3.2.7 Mobile phone Bluetooth control

3.2.7.1 Module Introduction

Aurora-racing support mobile phone Bluetooth app remote control function. The Bluetooth module used in the racing car is the JDY-16 ble module.

JDY-16 transmission module is based on Bluetooth 4.2 protocol standard, the working band for the 2.4GHZ range, modulation mode for the GFSK, the maximum emission power of 0db, the maximum emission distance of 80 meters, the use of imported original chip design, support users through the AT command to modify the device name, service UUID, transmit power, Matching password and other instructions, convenient and quick to use flexible. Module information see aurora-racingdocumentjdy-16-v1.2 (**“aurora-racing\Document\JDY-16-V1.2(English manual).pdf”**). PDF,the module physical map as shown in the picture 3.2.27.



Picture 3.2.27 JDY-16 Module

3.2.7.2 Function Introduction

- BLE high speed transmission, support 8K Bytes rate communication
- There is no limiton of bytes when sending and receiving datas, supporting 115200 baud rate continuous transceiver data .
- Support 3 working modes (please see At+starten instruction function description)
- Support (serial, IO, APP) sleep wake
- Support WeChat Airsync and WeChat applets、applications in micro-credit H5 or factory server communication and communication with APP
- Support 4-Way IO port control (applied to mobile phone control relays or LED lights Out)

- Support high precision RTC clock
- Support PWM function (via UART, IIC, APP, etc.)
- Support UART and IIC communication mode, the default is UART communication
- Ibeacon mode (support for micro-signal shake-roll protocol and Apple ibeacon Protocol)
- Host transmission mode (application of data transmission between modules, host and from machine communication)

JDY-16 module test method see **"aurora-Racing\JDY-16\JDY-16 Module Test.pdf"**

3.2.7.3 Bluetooth Control Aurora-Racing Principle

Use Bluetooth to control the car, in fact is using the Android app to send instructions to the Arduino serial port via Bluetooth to control the car. Since it involves wireless communication, one of the essential problems is the communication between the two devices. But there is no common "language" between them, so it is necessary to design communication protocols to ensure perfect interaction between Android and Arduino. The main process is: The Android recognizes the control command and package it into the corresponding packet, then sent to the Bluetooth module (JDY-16), JDY-16 received data and send to Arduino, then Arduino analysis the data then perform the corresponding action. The date format that the Android send as below, mainly contains 8 fields.

Protocol Header	Data Length	Device Type	Device Address	Function Code	Control Data	Check Sum	Protocol End Code

In the 8 fields above,we use a structural body to represent.

```
typedef struct
{
    unsigned char start_code;     // 8bit 0xAA
    unsigned char len;
    unsigned char type;
    unsigned char addr;
    unsigned short int function;   // 16 bit
    unsigned char *data;          // n bit
    unsigned short int sum;        // check sum
    unsigned char end_code;        // 8bit 0x55
}ST_protocol;
```

"Protocol Header" means the beginning of the packet, such as the uniform designation of 0xAA.
"Data length"means except the valid data length of the start and end codes of the data.
"Device type"means the type of device equipment
"Device address" means the address that is set for control

“Function code”means the type of equipment functions that need to be controlled,the function types we currently support as follows.

```
typedef enum
{
    E_BATTERY = 1,
    E_LED = 2,
    E_BUZZER = 3,
    E_INFO = 4,
    E_ROBOT_CONTROL = 5,
    E_ROBOT_CONTROL_SPEED = 6,
    E_TEMPERATURE = 7,
    E_IR_TRACKING = 8,
    E_ULTRASONIC = 9,
    E_VERSION = 10,
    E_UPGRADE = 11,
}E_CONTOROL_FUNC ;
```

“Data”is about the exact values that we control the racing car,such as speed and angle

“Checksum” is the result of different or calculated data bits of the control instruction.

“Protocol end code ”is the end part of the data bag,when receiving this data,it means that the data pack has been sent,and is ready for receiving the next data pack,here we specified it as 0x55.

For example: A complete data pack can be like this: “AA 07 01 01 06 50 00 5F 55”,

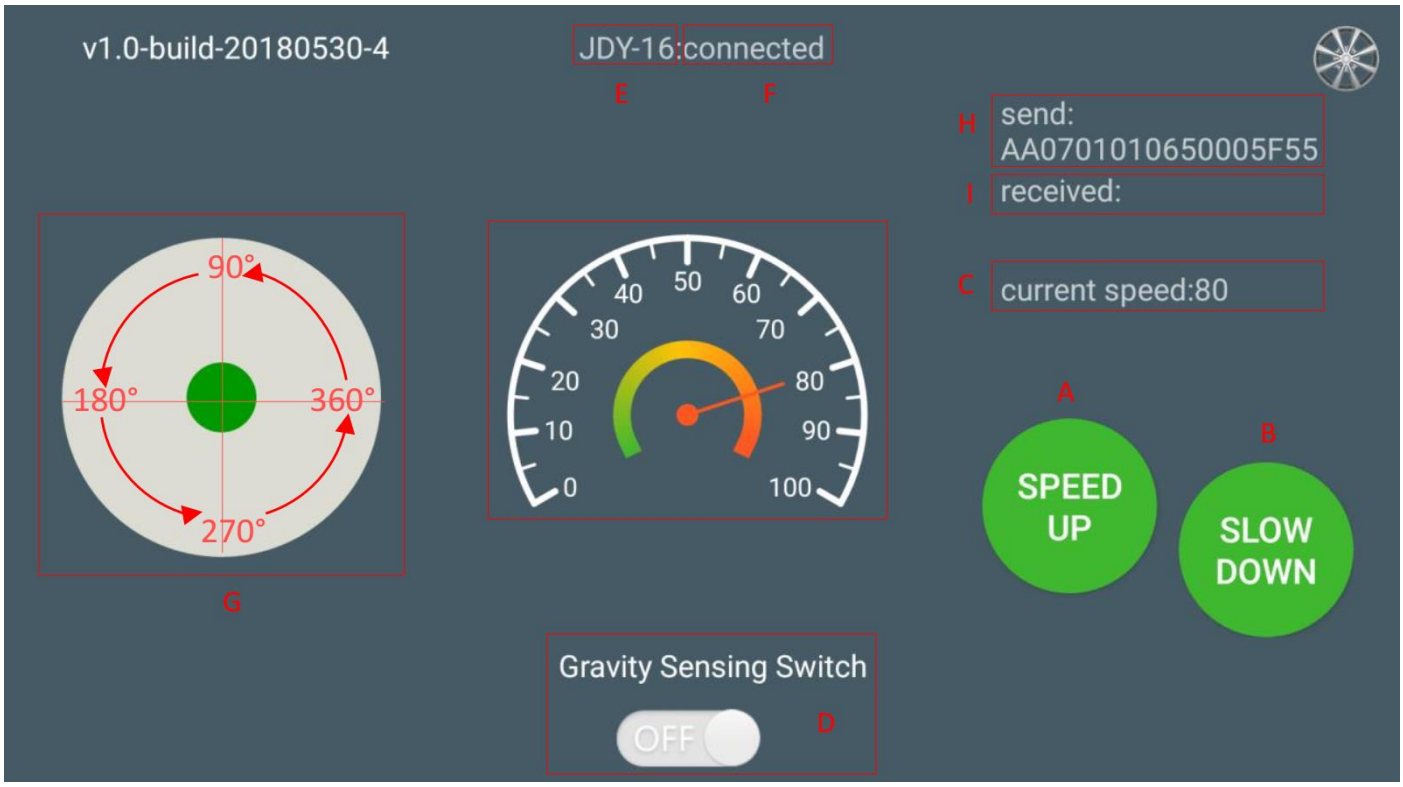
“07” Transmission Data Length 7 bytes

“06” is “Device type”， the device that we specified,such as LED、buzzer;here 06 means the “speed” of transportation,05 means the “direction” of transportation.

"50 (or 0050)" is the control data, where 0x50 is hexadecimal, converted to binary 80, if "Device type" transmits 06, then the data here is the speed value, that is, the speed is 80. If the transfer 05 o'clock is the control direction, that is, the 80° direction (forward).

“005F” is a checksum that is 0x07+0x01+0x01+0x06+0x50=0x5F.

“55” is the end code of the Protocol, indicating the end of data transfer.



Picture 3.2.29 Android Interface diagram

Above the picture 3.2.29

The "A, B" section is a plus deceleration button.

The "C" section includes the dashboard and the digital display area, two parts for synchronized display. Indicates the current speed, that is, the speed of the acceleration and deceleration.

The "D" section is a gravity remote sensing switch, which can be switched to the gravity remote sensing mode.

The "E" section indicates the Bluetooth name that is currently connected to.

The "F" section indicates the Bluetooth connection status, and if Bluetooth is not connected, the "disconnected" is displayed here.

The "G" section is the hand shakes the pole, the slippery sway pole may let the car rotate direction.

The "I" section is a data return area, such as the current driving state and speed of the trolley.

The "H" section is for packets sent, such as the data "AA 2B 55", which indicates that the speed is 35 (23 is 16 data, and conversion to 10 is the current speed of 35).

If the data sent is “AA 08 01 01 05 00 5B 00 6A 55”,it means that the car is driving forward(05 is a direction command instruction,91 is the value when 005B converted to binary.from picture 3.2.17, we can know that 91°also means moving forward)

3.2.7.4 Experimental procedure

1. Connect the bluetooth module to the Arduino serial port (this step we ignore directly, our expansion board has connected Bluetooth and Arduino serial port, more convenient, faster, stable).

2. Turn on the mobile bluetooth(**Note:** Do not connect the car's bluetooth in the phone settings, you can connect directly in our app) install "appbluetoothcontrolcom.keywish.robot.apk" to you phone and open the app (there is a software installation package in the accessory CD that currently only support Android phones, later will release the iOS version) ,it will automatically connect our car bluetooth.

3.2.7.5 Software Design

Bluetooth part of the program includes many library files, we do not annotate here, please open the file **"Lesson\AuroraRacing\ AuroraRacing.ino"**, download the program to the racing board, according to the above mobile phone APP operation to control racing car. We will continue to improve and increase the function of the app, please pay attention to our github updates.

3.2.8 PS2 Wireless Control (optional)

3.2.8.1 Introduction

Picture 3.2.30 PS2 wireless handle

PS2 handle is composed of two parts, the handle and receiver, the handle needs two section 7th 1.5V batteries, the receiver and arduino control board use the same power supply, the voltage is 3~5v, can not be reversed, can not exceed , overvoltage and reverse connection will cause the receiver to burn out. There is a power switch on the handle,turn handle switch to on, the lamp on the handle will flash if didn't search the receiver, the handle will enter Standby mode if this station last for some time,the light will go out. At this time, you need to press the "START" button to wake-up handle.

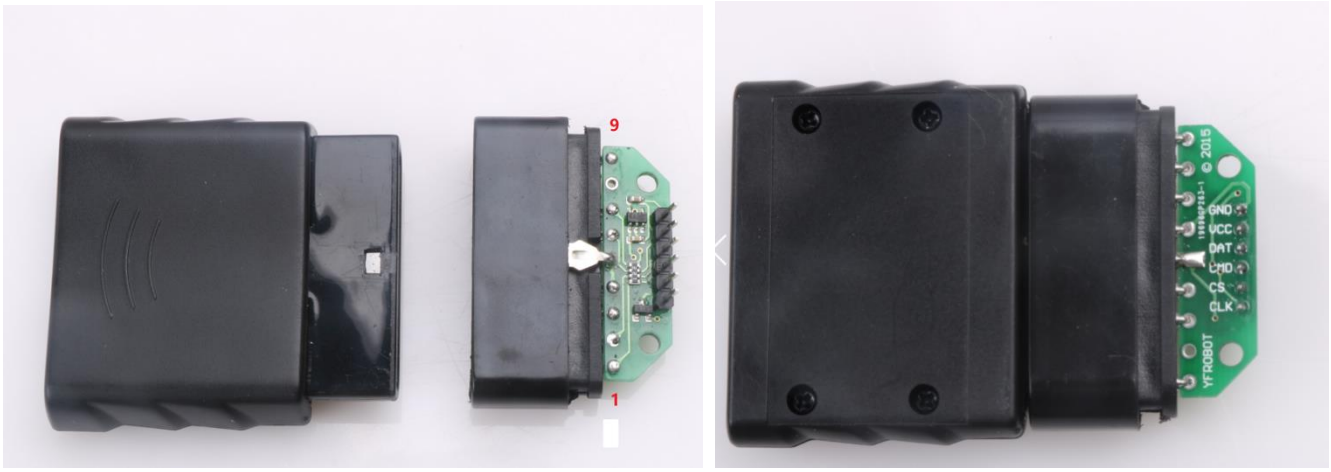
The receiver is connected to the Arduino, and powered by Arduino, such as the picture 3.2.40, in an unpaired condition,a green light will flash.If the handle is open, the receiver is powered, the handle and receiver will automatically pair, then the lamp is always bright, the handle pair succeeds. The key "mode" (handle batch is different, the identification may be "analog", but will not affect the use), you can choose "Red light Mode", "Green mode".

Some users reflect that the handle and receiver can not be paired properly!

Most of the problems are that the receiver wiring is incorrect or the program is incorrect.

Solution:The receiver only receives the power supply (the power cord must be connected correctly), without any data line and clock line, the handle is usually able to pair successfully. Paired success then lights will keep bright, indicating that the handle is good.

Then check if the wiring is correct, Is there any problem with program migration?



Picture 3.2.31 Remote receiver module

There are 9 interfaces on the the receiver, shown in the following table:

1	2	3	4	5	6	7	8	9
DI/DAT	DO/CMD	NC	GND	VDD	CS/SEL	CLK	NC	ACK

Note: Different batches, the appearance of the receiver will be some different, but the pin definition is the same,so don’t worry about the use.

Di/dat: Signal flow, from the handle to the host, this signal is a 8bit serial data, synchronous transmission in the clock down the edge. The reading of the signal is done in the process of changing the clock from high to low.

Do/CMD: Signal flow, from the host to the handle, this signal is relative to DI, the signal is a 8bit serial data, synchronously transmitted to the clock down the edge.

NC: Empty port; .

GND: Ground;

VDD: The 3~5V power supply;

CS/SEL: Providing trigger signals for handles, the level is low during communication;

CLK: The clock signal is sent by the host to maintain data synchronization;

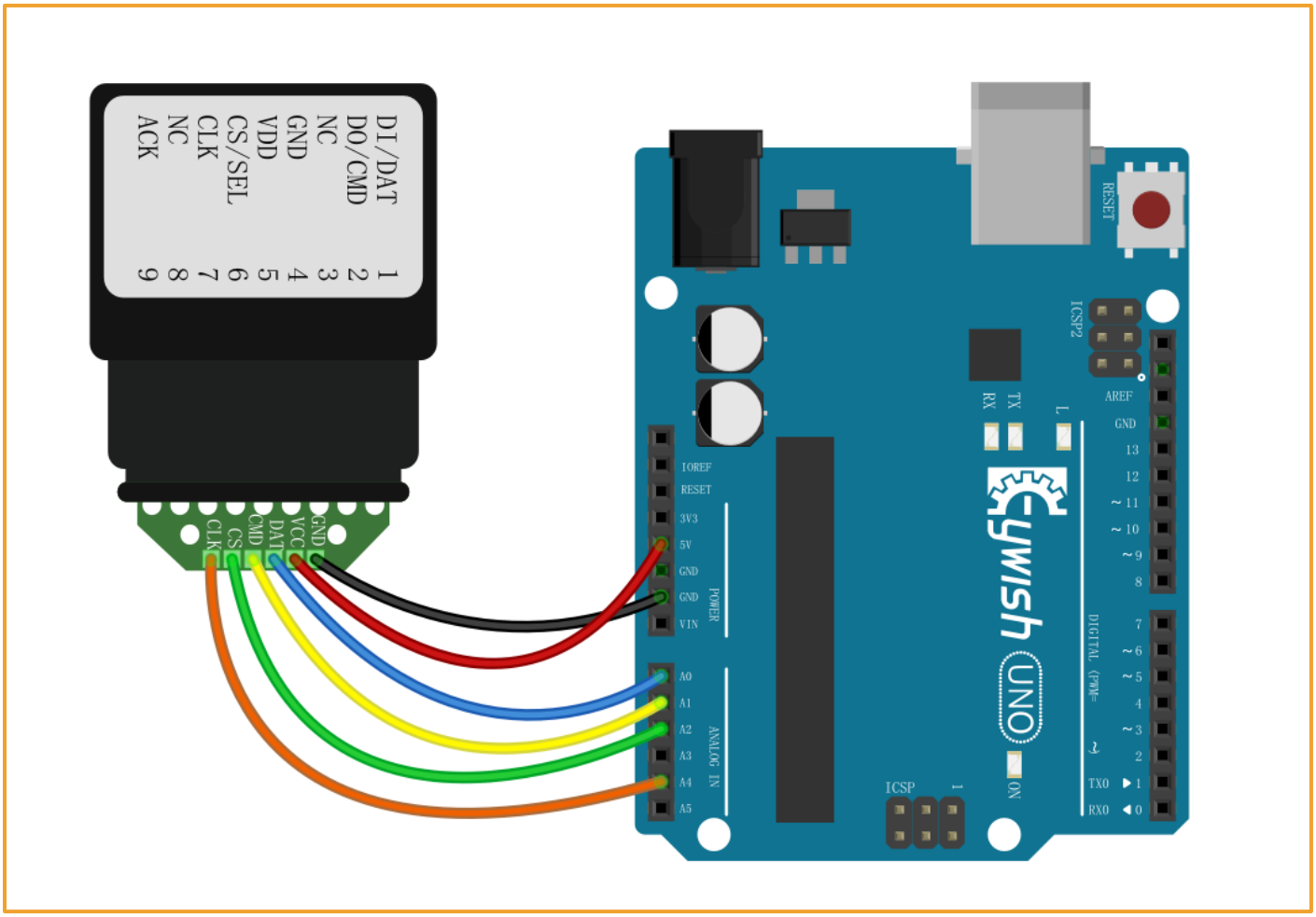
NC: Empty port;

ACK: the response signal from the handle to the host. This signal changes to low in the last cycle of each 8-bit data sending, and the CS remains low. If the CS signal do not remain low, the PS host will try another device in about 60 microseconds. The ACK port is not used in programming.

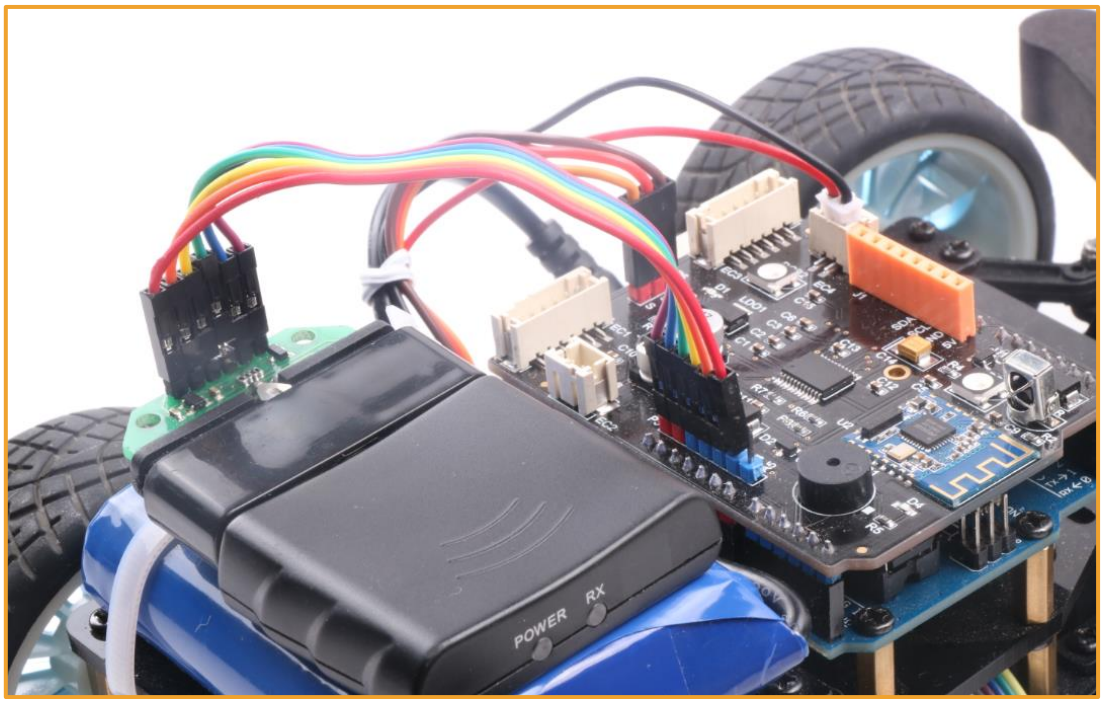
3.2.6.2 Experimental procedures

1, The wires are connected to the 1, 2, 4, 5, 6 and 7 pins of the receiver respectively as shown in the picture.3.2.41.

Ps2	Arduino Uno
DAT	A0
CMD	A1
CS	A2
CLK	A4



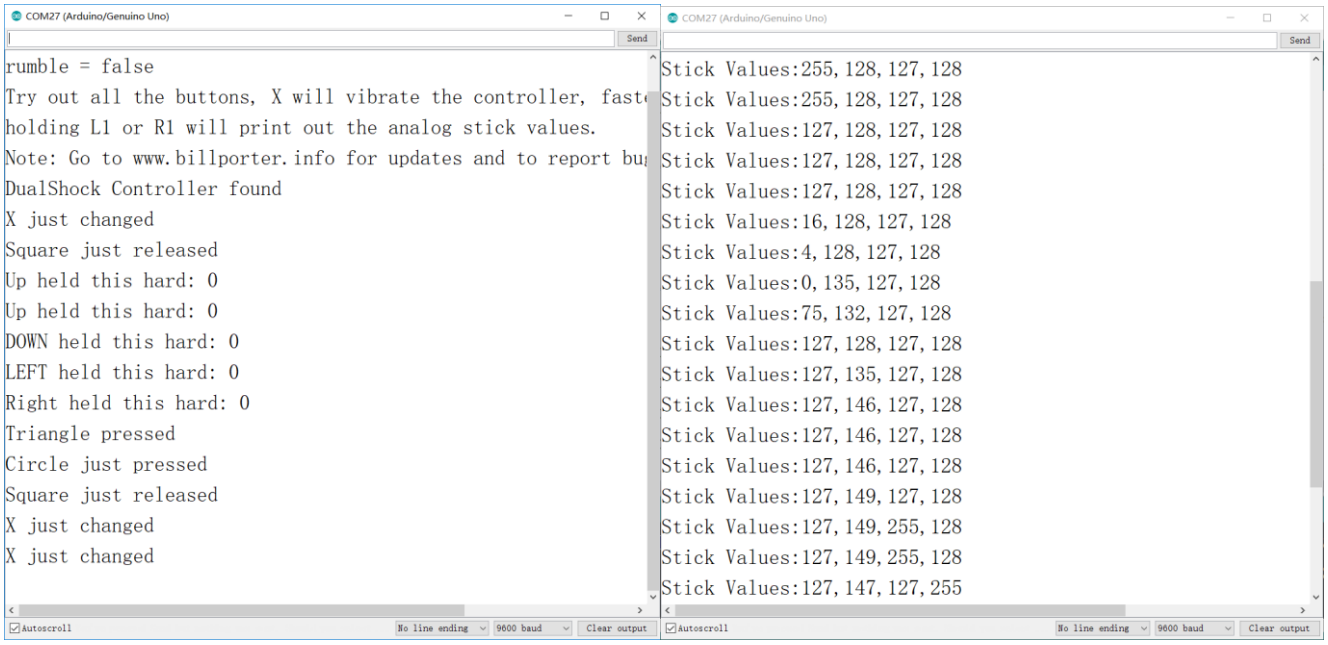
Picture 3.2.32 Arduino and receiver wire connection



Picture.3.2.33 Installation of Receiving Head

4、Open the CD"Lesson\ModuleDemo\PS2X\PS2X.ino ",download the program to the Arduino Board, open the PS2 remote control, if the receiver and the remote control has been connected (or the pairing is successful), the receiver's led will keep light, if not the LED light constantly flashing.

Then we open "serial monitor", press any button on the remote control, you can see the corresponding data in the "serial monitor" ,shown as the picture 3.2.34.



Picture 3.2.34 Serial monitor data display

3.2.6.3 Software Design

```
#include "PS2X_lib.h"  //for v1.6

/******************************************************************
 * set pins connected to PS2 controller:
 *   - 1e column: original
 *   - 2e colmun: Stef?
 * replace pin numbers by the ones you use
 ******************************************************************/
#define PS2_DAT     A0
#define PS2_CMD     A1
#define PS2_SEL     A2
#define PS2_CLK     A4

/******************************************************************
 * select modes of PS2 controller:
 *   - pressures = analog reading of push-butttons
 *   - rumble    = motor rumbling
 * uncomment 1 of the lines for each mode selection
 ******************************************************************/
//#define pressures   true
#define pressures   false
//#define rumble      true
#define rumble      false

PS2X ps2x; // create PS2 Controller Class

//right now, the library does NOT support hot pluggable controllers, meaning
//you must always either restart your Arduino after you connect the controller,
//or call config_gamepad(pins) again after connecting the controller.

int error = 0;
byte type = 0;
byte vibrate = 0;

// Reset func
void (* resetFunc) (void) = 0;
```

```
void setup() {
  Serial.begin(9600);
  delay(500);  //added delay to give wireless ps2 module some time to startup, before configuring it
  //CHANGES for v1.6 HERE!!! **************PAY ATTENTION*************
  //setup pins and settings: GamePad(clock, command, attention, data, Pressures?, Rumble?) check for error
  error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT, pressures, rumble);

  if(error == 0){
    Serial.print("Found Controller, configured successful ");
    Serial.print("pressures = ");
    if (pressures)
      Serial.println("true ");
    else
      Serial.println("false");
    Serial.print("rumble = ");
    if (rumble)
      Serial.println("true)");
    else
      Serial.println("false");
    Serial.println("Try out all the buttons, X will vibrate the controller, faster as you press harder;");
    Serial.println("holding L1 or R1 will print out the analog stick values.");
    Serial.println("Note: Go to www.billporter.info for updates and to report bugs.");
  }
  else if(error == 1)
    Serial.println("No controller found, check wiring, see readme.txt to enable debug. visit www.billporter.info for troubleshooting tips");

  else if(error == 2)
    Serial.println("Controller found but not accepting commands. see readme.txt to enable debug. Visit www.billporter.info for troubleshooting tips");

  else if(error == 3)
    Serial.println("Controller refusing to enter Pressures mode, may not support it. ");

  type = ps2x.readType();
```

```
  switch(type) {
    case 0:
      Serial.println("Unknown Controller type found ");
      break;
    case 1:
      Serial.println("DualShock Controller found ");
      break;
    case 2:
      Serial.println("GuitarHero Controller found ");
      break;
    case 3:
      Serial.println("Wireless Sony DualShock Controller found ");
      break;
   }
}

void loop() {
  /* You must Read Gamepad to get new values and set vibration values
     ps2x.read_gamepad(small motor on/off, larger motor strenght from 0-255)
     if you don't enable the rumble, use ps2x.read_gamepad(); with no values
     You should call this at least once a second
   */
  if(error == 1){ //skip loop if no controller found
    resetFunc();
  }
  if(type == 2){ //Guitar Hero Controller
    ps2x.read_gamepad();          //read controller

    if(ps2x.ButtonPressed(GREEN_FRET))
      Serial.println("Green Fret Pressed");
    if(ps2x.ButtonPressed(RED_FRET))
      Serial.println("Red Fret Pressed");
    if(ps2x.ButtonPressed(YELLOW_FRET))
      Serial.println("Yellow Fret Pressed");
    if(ps2x.ButtonPressed(BLUE_FRET))
      Serial.println("Blue Fret Pressed");
    if(ps2x.ButtonPressed(ORANGE_FRET))
      Serial.println("Orange Fret Pressed");
    if(ps2x.ButtonPressed(STAR_POWER))
      Serial.println("Star Power Command");
```

```
    if(ps2x.Button(UP_STRUM))         //will be TRUE as long as button is pressed
      Serial.println("Up Strum");
    if(ps2x.Button(DOWN_STRUM))
      Serial.println("DOWN Strum");
    if(ps2x.Button(PSB_START))        //will be TRUE as long as button is pressed
      Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))
      Serial.println("Select is being held");
    if(ps2x.Button(ORANGE_FRET)) {     // print stick value IF TRUE
      Serial.print("Wammy Bar Position:");
      Serial.println(ps2x.Analog(WHAMMY_BAR), DEC);
    }
  }
  else { //DualShock Controller
    ps2x.read_gamepad(false, vibrate); //read controller and set large motor to spin at
'vibrate' speed
    if(ps2x.Button(PSB_START))        //will be TRUE as long as button is pressed
      Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))




 Serial.println("Select is being held");

    if(ps2x.Button(PSB_PAD_UP)) {      //will be TRUE as long as button is pressed
      Serial.print("Up held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_UP), DEC);
    }
    if(ps2x.Button(PSB_PAD_RIGHT)){
      Serial.print("Right held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_RIGHT), DEC);
    }
    if(ps2x.Button(PSB_PAD_LEFT)){
      Serial.print("LEFT held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_LEFT), DEC);
    }
    if(ps2x.Button(PSB_PAD_DOWN)){
```

```
    vibrate = ps2x.Analog(PSAB_CROSS);  //this will set the large motor vibrate speed
based on how hard you press the blue (X) button
    if (ps2x.NewButtonState()) {       //will be TRUE if any button changes state (on to
off, or off to on)
      if(ps2x.Button(PSB_L3))
        Serial.println("L3 pressed");
      if(ps2x.Button(PSB_R3))
        Serial.println("R3 pressed");
      if(ps2x.Button(PSB_L2))
        Serial.println("L2 pressed");
      if(ps2x.Button(PSB_R2))
        Serial.println("R2 pressed");
      if(ps2x.Button(PSB_TRIANGLE))
        Serial.println("Triangle pressed");
    }

    if(ps2x.ButtonPressed(PSB_CIRCLE))              //will be TRUE if button was JUST
pressed
      Serial.println("Circle just pressed");
    if(ps2x.NewButtonState(PSB_CROSS))              //will be TRUE if button was JUST
pressed OR released
      Serial.println("X just changed");
    if(ps2x.ButtonReleased(PSB_SQUARE))             //will be TRUE if button was JUST
released
      Serial.println("Square just released");

    if(ps2x.Button(PSB_L1) || ps2x.Button(PSB_R1)) { //print stick values if either is
TRUE
      Serial.print("Stick Values:");
      Serial.print(ps2x.Analog(PSS_LY), DEC); //Left stick, Y axis. Other options: LX,
RY, RX
      Serial.print(",");
      Serial.print(ps2x.Analog(PSS_LX), DEC);
      Serial.print(",");
      Serial.print(ps2x.Analog(PSS_RY), DEC);
      Serial.print(",");
      Serial.println(ps2x.Analog(PSS_RX), DEC);
    }
  }
  delay(50);
}
```

In above programs,we just finished the experiment of testing the button. In this experiment we want to implement the PS2 remote control car function. We first define all the button functions as follows:

Picture 3.2.34 Functions of PS2 Handle Buttons

PS2 handle description:

Mark UP: move forward

Mark DOWN: move backward

Mark LEFT: turn left

Mark RIGHT turn right

Mark 3: right joystick (Mark 5) control key. When the R1 is pressed, the right joystick will work.

Mark 4: left joystick (Mark 6) control key. When L1 is pressed, the left joystick will work.

Joystick Right :adjust right rgb led

Joystick Left: adjust left rgb led

Start: power switch

Ps2 Control program in Lesson\Advanced Experiment\PS2XControl\PS2XControl.ino