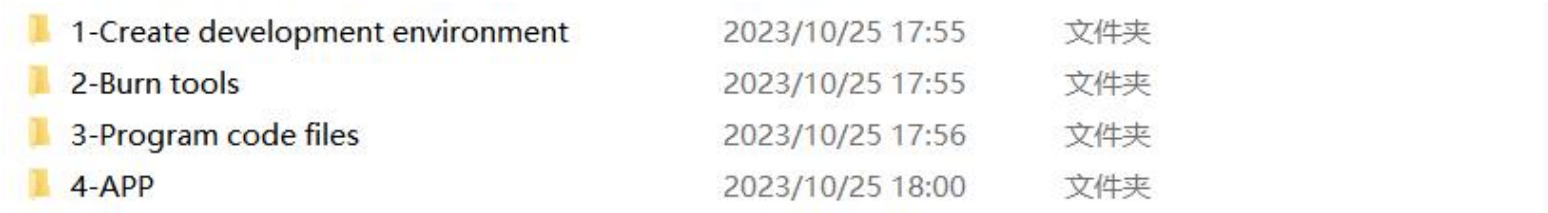
STM32 Ackerman smart remote control car

Tutorial introduction

This tutorial includes the file content as shown in the figure below to help you learn this kit and related programming knowledge from shallow to deep:



"1-Create development environment": Stores the package assembly guide and necessary software environment files, etc. In order to complete the assembly accurately and quickly, please be sure to read the manual in detail and assemble it according to the manual. At the same time, the prerequisite for realizing program functions is to correctly create the software environment. Please view the PDF file in this folder.

"2-Burning Tool": Stores the burning tool ST Link and operating instructions. It also includes the "Motor Test" project file. After burning, test whether the tool is normal.

"3-Program code file": stores code files, the codes are from simple to difficult, and the last code is the comprehensive remote control code;

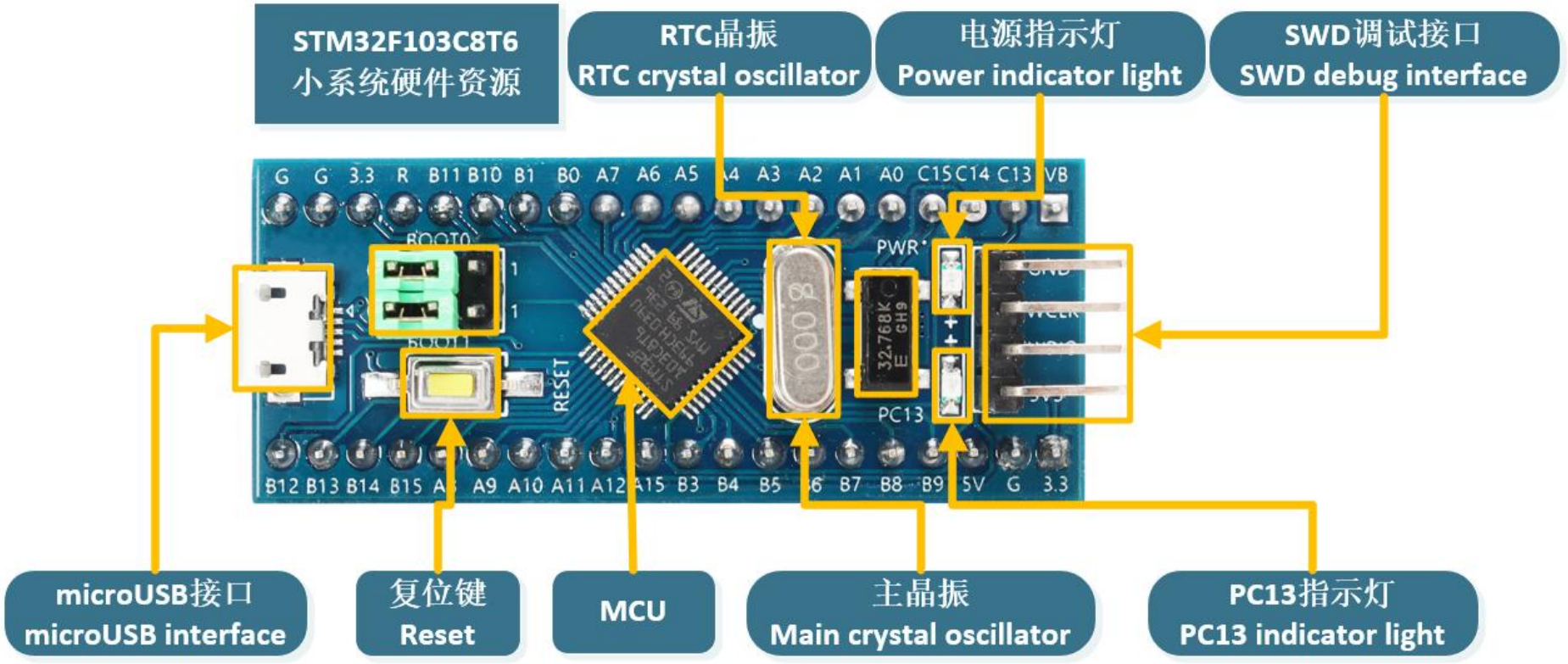
“4-APP”: stores the Android version APP installation package and APP usage guide for Bluetooth remote control.

After completing the assembly of the suit and the creation of the environment, follow this tutorial to gradually burn the smart car program and implement different functions!

Table of contents

1. Understanding the development kit and resetting the servo

1.1 Development board STM32F103C8T6 hardware resource distribution



01. Main chip model STM32F103C8T6 , core: ARM Cortex-M3, power supply: 2.0~3.6V (standard 3.3V)

02. Main frequency 72M

03. 3.3V voltage stabilizing chip, providing a maximum current of 800mA

04. One miniUSB interface can power the system version and reserve USB communication function

05. Reset button

06. One standard SWD port, supporting JLink, STLink, JLINKOB

07. BOOT select port

08. 1 power indicator light

09. One function indicator light, used to verify the basic functions of the IO port

1 0. Reserved serial port interface to facilitate connection with 5V development board

1 1. High performance Epson 32768Hz crystal oscillator, easy to start.

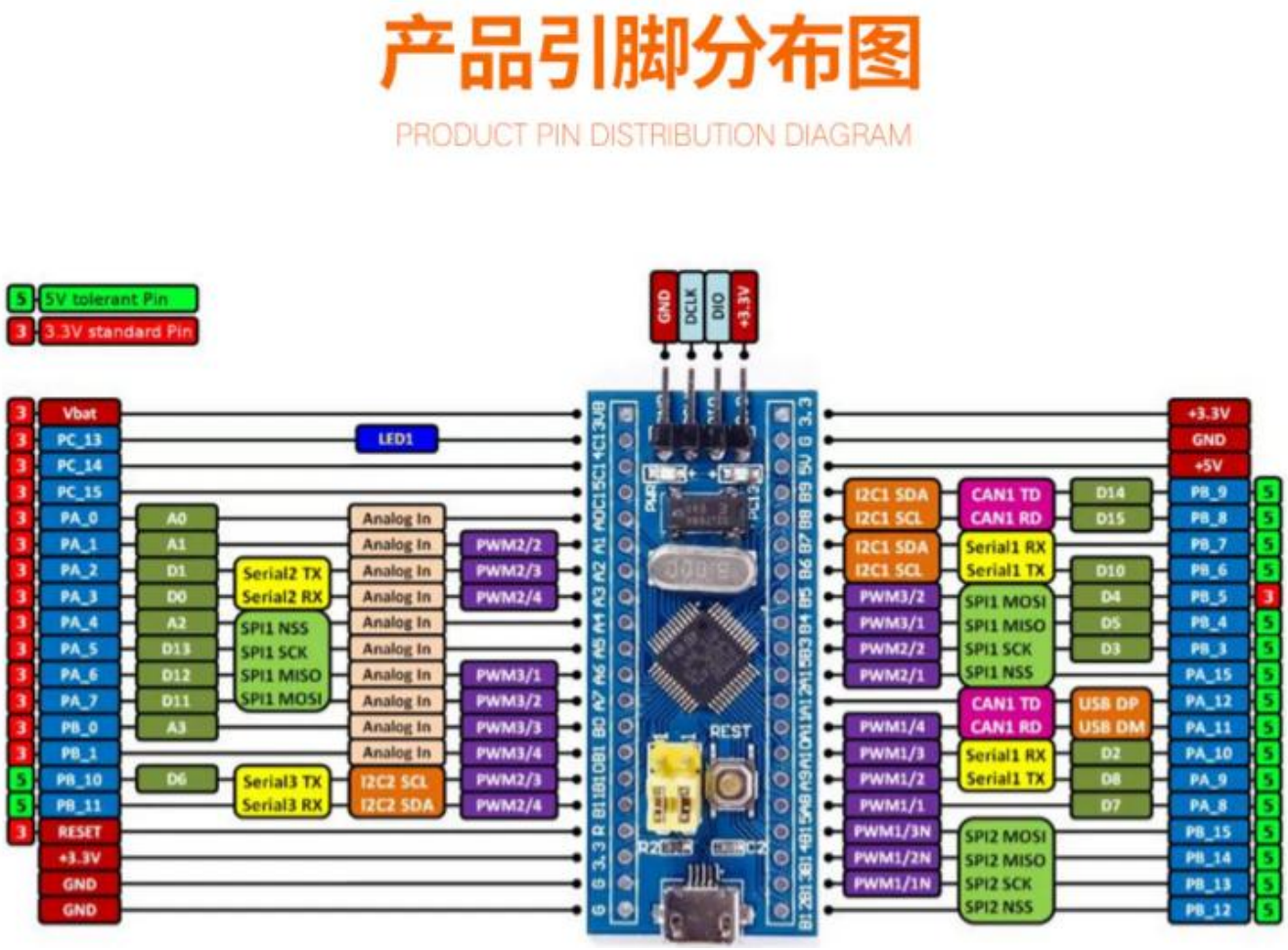
1 2. 20K RAM, 64K ROM, TQFP48 package

13. Timer resources: TIM1, TIM2, TIM3, TIM4

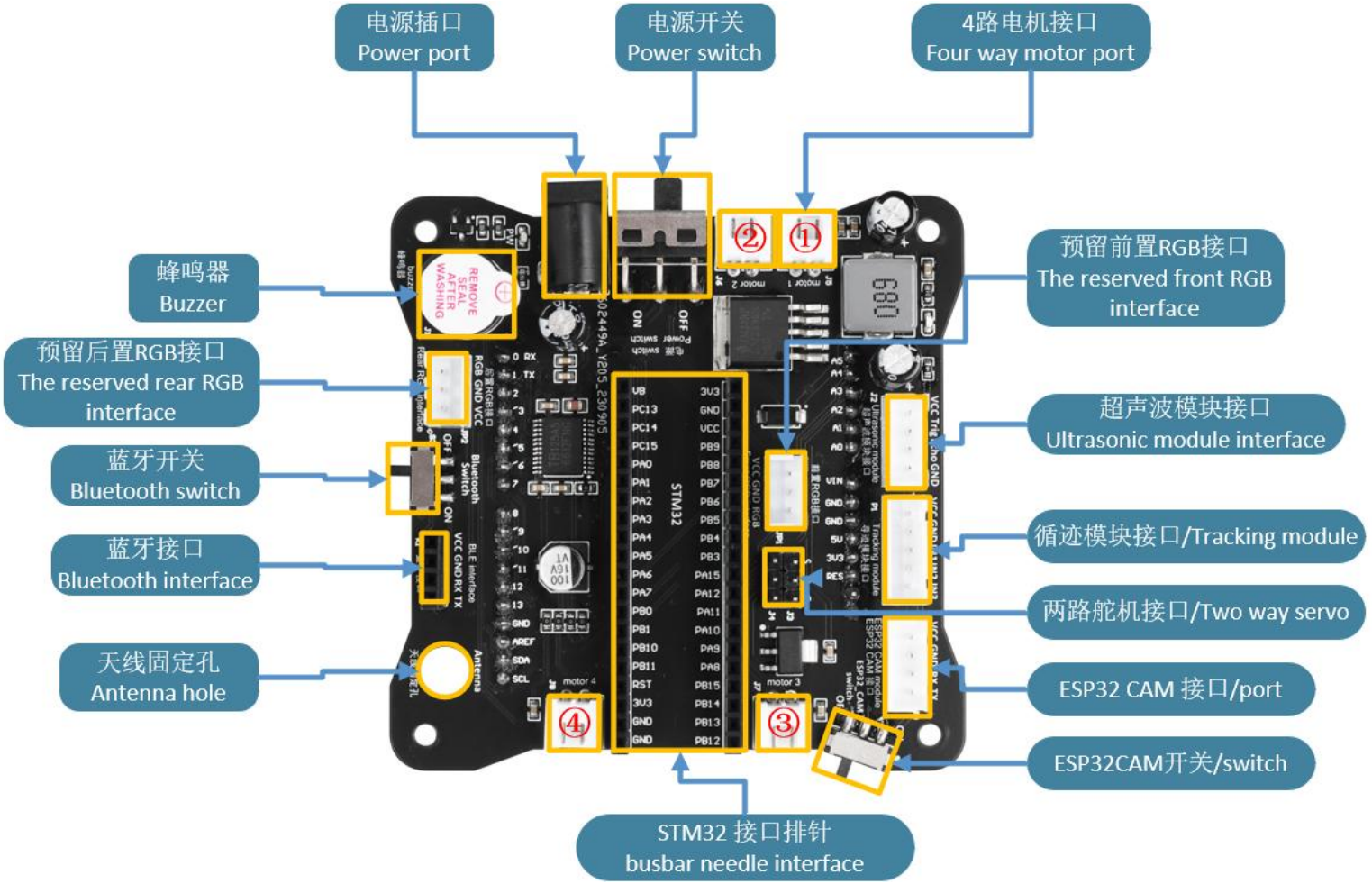
14. On-chip resources and peripherals

英文缩写	名称	英文缩写	名称
NVIC	嵌套向量中断控制器	CAN	CAN通信
SysTick	系统滴答定时器	USB	USB通信
RCC	复位和时钟控制	RTC	实时时钟
GPIO	通用IO口	CRC	CRC校验
AFIO	复用IO口	PWR	电源控制
EXTI	外部中断	BKP	备份寄存器
TIM	定时器	IWDG	独立看门狗
ADC	模数转换器	WWDG	窗口看门狗
DMA	直接内存访问	DAC	数模转换器
USART	同步/异步串口通信	SDIO	SD卡接口
I2C	I2C通信	FSMC	可变静态存储控制器
SPI	SPI通信	USB OTG	USB主机接口

1.2 Pin distribution diagram



1.3 Expansion board interface and resource distribution



① One power input DC hole (accepts 5V~12V)

② 4 motor interfaces

③ a buzzer

④ Two RGB interfaces

⑤ An ultrasonic sensor interface

⑥ A three-way tracking module interface

⑦ Two servo interfaces

⑧ An ESP32CAM expansion interface

⑨ A Bluetooth switch S3

⑩ An ESP32CAM switch S4

⑪ A reserved antenna hole

Note : When using ESP32 CAM, you need to turn on the S3 switch for power supply, and when using Bluetooth, you need to turn on the S4 switch. This expansion board is compatible with both STM32 and Arduino UNO main control boards.

1.4 Servo reset

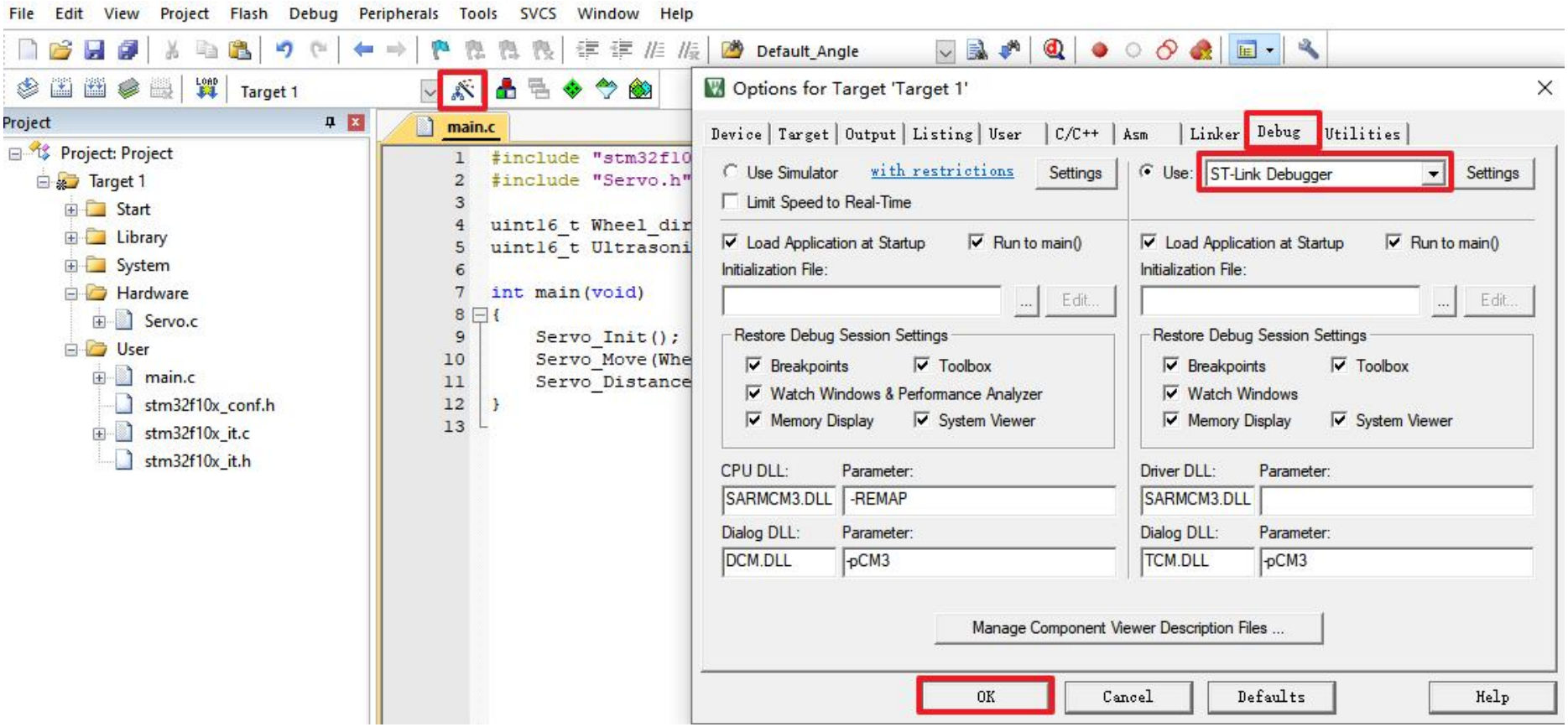
describe

The servo needs to be reset and corrected when assembling the kit. By burning the "servo reset" code, the servo can be reset to the initial "directly ahead" to facilitate correct assembly of the servo.

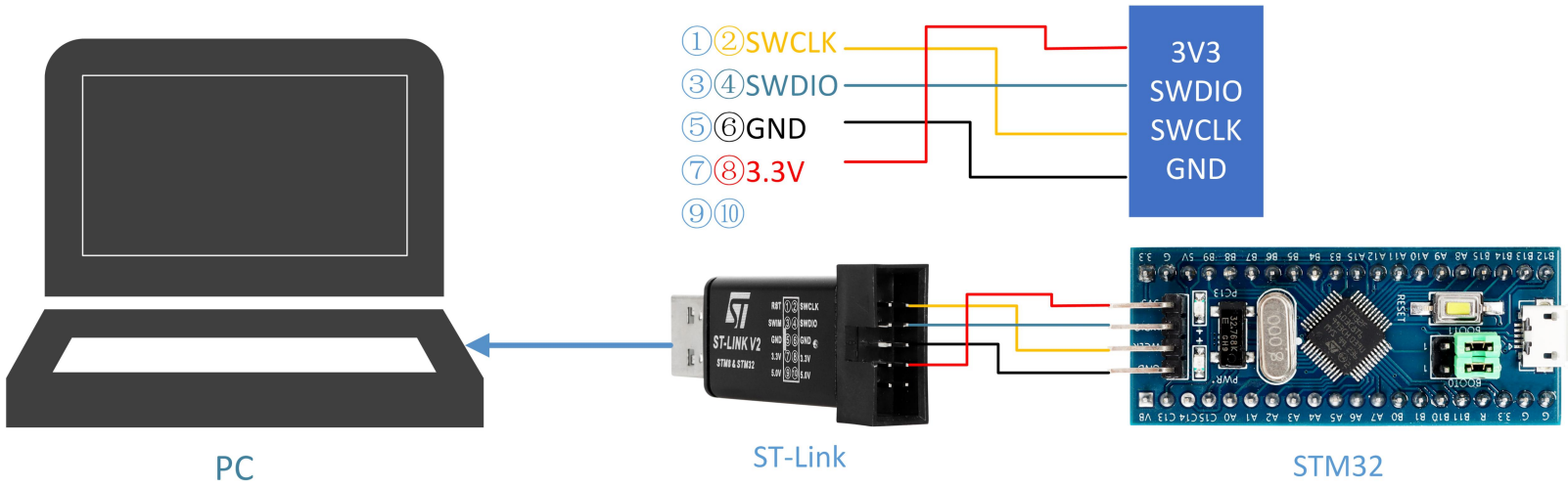
Burn servo reset code

Open the project file “0-Servo Reset” (folder path: 1-STM32\3-Program Code File\0-Servo Reset\Project.uvprojx)

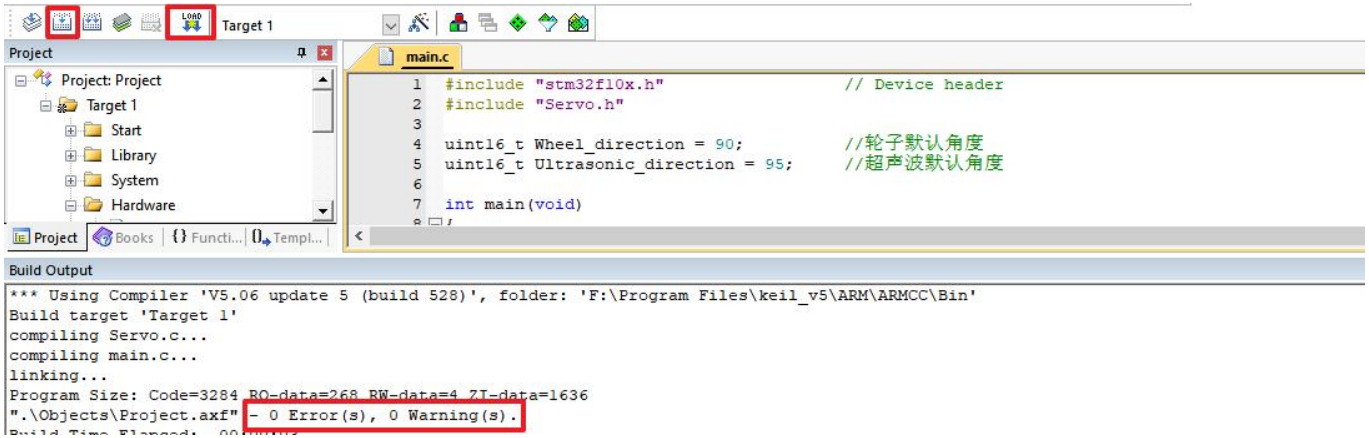
Check the Debug mode of "Target Options": ST-Link



One end of the USB of the ST-Link tool is plugged into the computer, and the other end is connected to the STM32 main control board with a 4-pin cable (note the pins to be connected)



Click "Compile", confirm it is correct, and then click "Download" to download the program to STM32

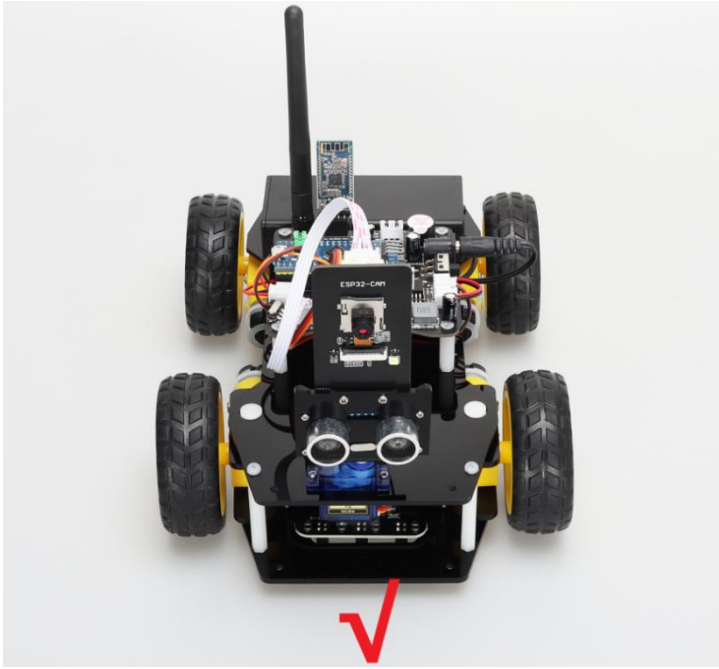


At this time, the servo begins to reset to the initial "directly ahead" position.

If you find that the ultrasonic module and front wheels are not facing straight ahead at this time, they need to be corrected again.

Correction method :

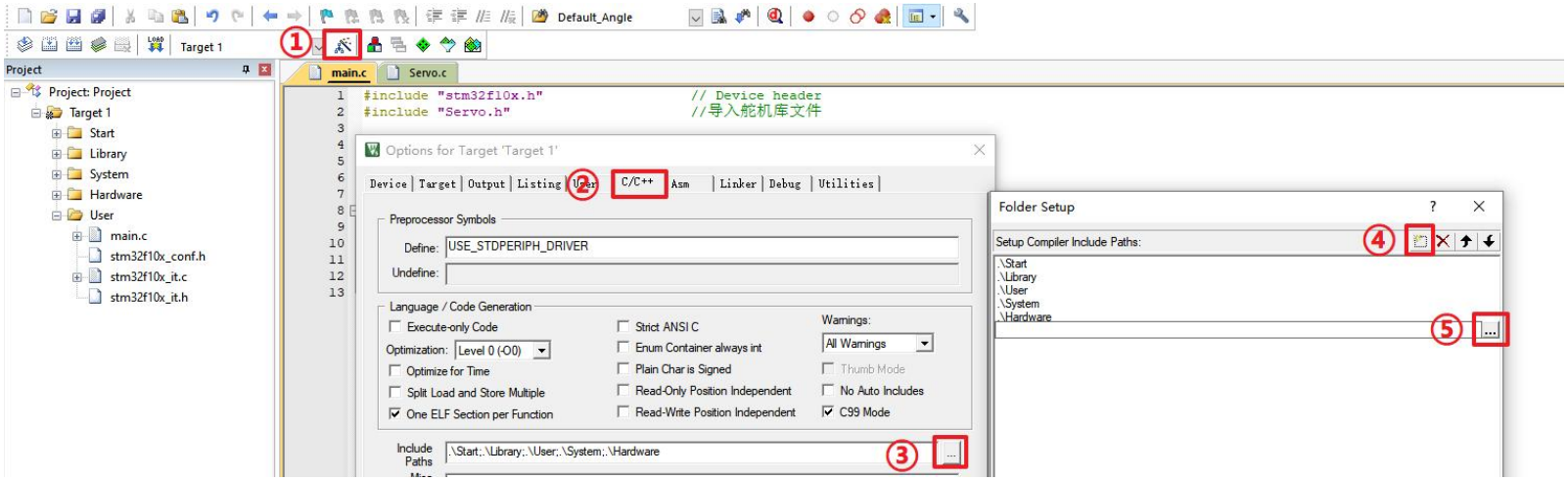
In the power-on reset state, remove the ultrasonic module or front wheel module, point it straight ahead (without turning the servo), and then re-fix it.

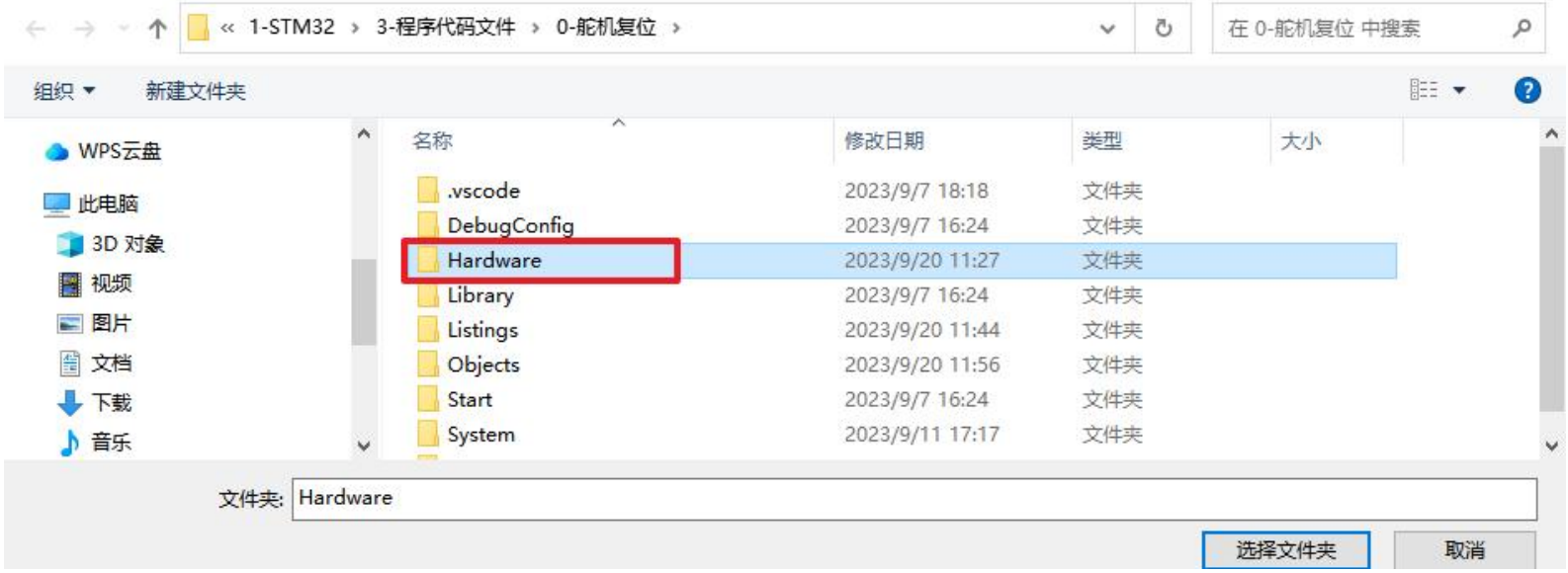


2. Custom function library

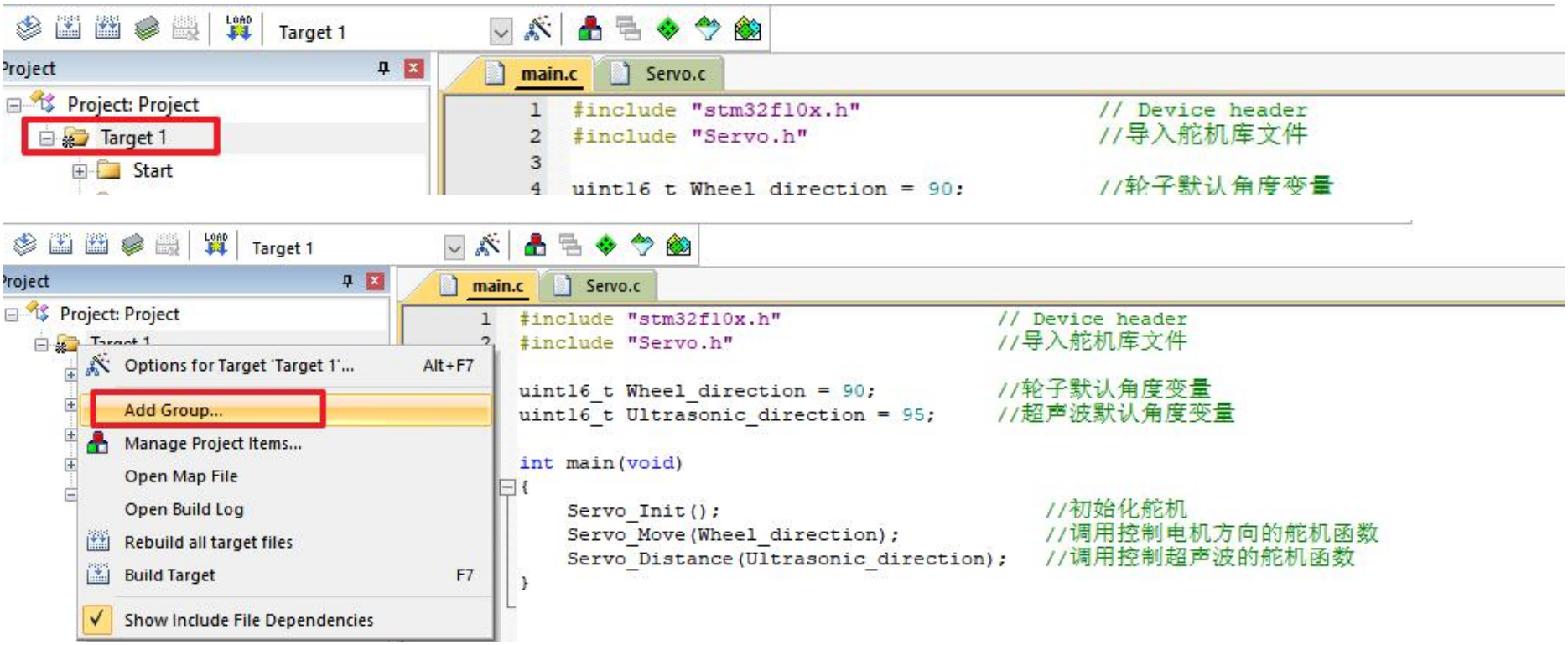
Each function library contains a file with the same name with .c and .h suffixes. The ".h" file contains encapsulated function declarations, and the ".c" file is the implementation of these functions. To use these function libraries, you must copy the function library file to the project directory and include it in the header of main.c in the project. At the same time, you must establish a path connection in the software settings. The specific operation is an example of adding the "STM32 Project Template" project of the servo library:

Copy the required library files Servo.c and Servo.h to the Hardware folder under the " STM32 Project Template " project directory (already placed in advance), open the "STM32 Project Template" project, as shown below, click according to the steps "Target Options" until the folder under the ⑤ selected item: Hardware

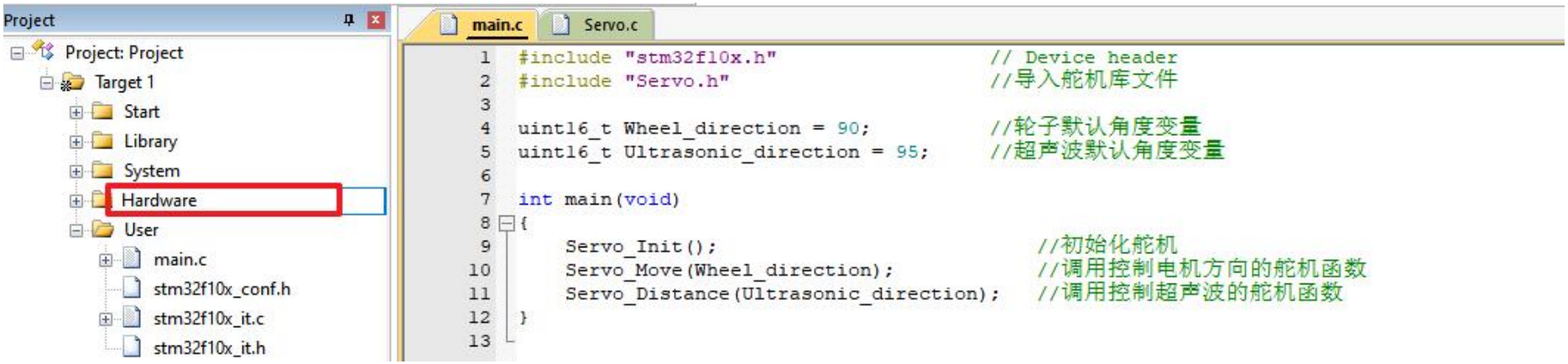

Default_Angle
Target 1
Project
Project: Project
Target 1
Start
Library
System
Hardware
User
main.c
stm32f10x_conf.h
stm32f10x_it.c
stm32f10x_it.h
main.c
Servo.c
#include "stm32f10x.h" // Device header
#include "Servo.h" //导入舵机库文件
Options for Target 'Target 1'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
Preprocessor Symbols
Define: USE_STDPERIPH_DRIVER
Undefine:
Language / Code Generation
Execute-only Code
Optimization: Level 0 (-O0)
Optimize for Time
Split Load and Store Multiple
One ELF Section per Function
Strict ANSI C
Enum Container always int
Plain Char is Signed
Read-Only Position Independent
Read-Write Position Independent
Warnings: All Warnings
Thumb Mode
No Auto Includes
C99 Mode
Include Paths .\Start;.\Library;.\User;.\System;.\Hardware
Folder Setup
Setup Compiler Include Paths:
.\Start
.\Library
.\User
.\System
.\Hardware
① ② ③ ④ ⑤


« 1-STM32 › 3-程序代码文件 › 0-舵机复位 ›
在 0-舵机复位 中搜索
组织
新建文件夹
WPS云盘
此电脑
3D 对象
视频
图片
文档
下载
音乐
名称
修改日期
类型
大小
.vscode 2023/9/7 18:18 文件夹
DebugConfig 2023/9/7 16:24 文件夹
Hardware 2023/9/20 11:27 文件夹
Library 2023/9/7 16:24 文件夹
Listings 2023/9/20 11:44 文件夹
Objects 2023/9/20 11:56 文件夹
Start 2023/9/7 16:24 文件夹
System 2023/9/11 17:17 文件夹
文件夹: Hardware
选择文件夹
取消

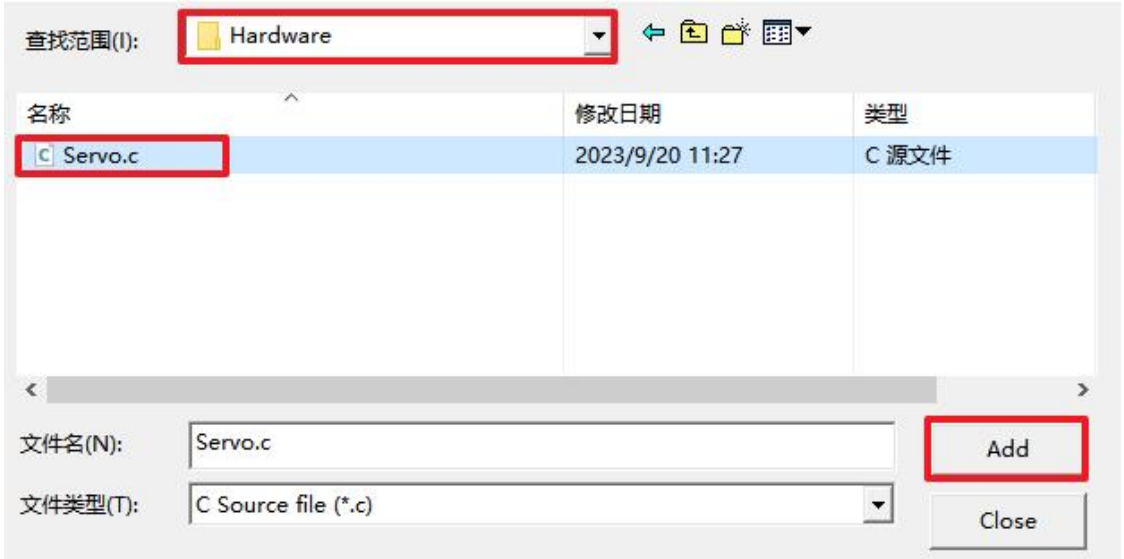
Click above to confirm. After binding the library path, return to the home page. Right-click on "Target 1" to add the "Group" folder.



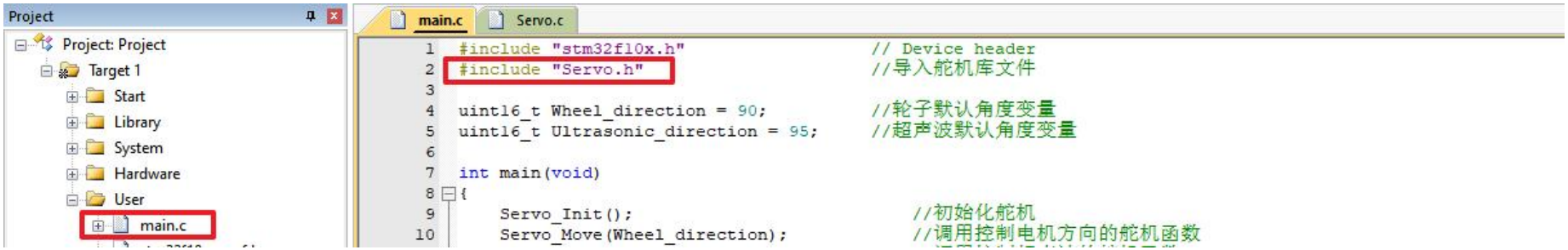
Press "F2" on the keyboard to change the group name to Hardware (the same name as the local folder just added), then double-click the "Hardware" group to add the library file



Select the library file to which you want to add the .c suffix



Return to the main.c file on the homepage, modify the code to control the servo reset and add an include statement. What is included here is the .h header file



Click "Compile" to complete adding binding

At this point, a Servo library has been successfully added from scratch. If you need to add another library file later, just double-click Hardware and select the .c file. There is no need to bind the link path.

It should be noted that the function codes (3~8) provided in this tutorial have already added libraries. If you add the same library repeatedly, an error will be reported. You need to try to add it in the "project template" that does not include the library. New Libraries to learn how to add libraries.

3.LED flashes

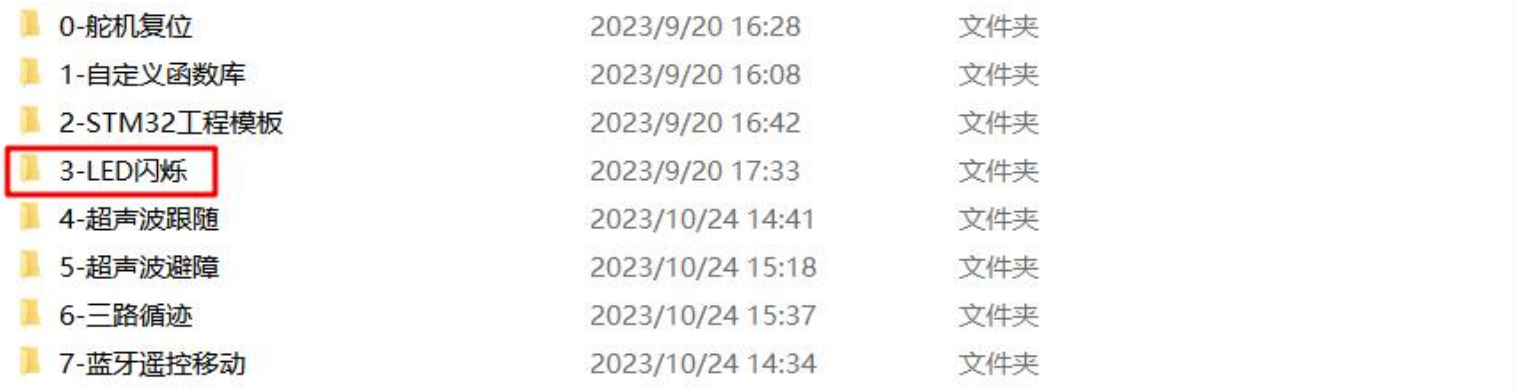
3.1 Description

This section will continue to familiarize you with software operations, program compilation, downloading and other operations, and modify the parameters of the delay function to make the lights flash slower.

3.2 Operation process review

① Connect the STM32 main control board to the computer through ST-Link;

②Open the "LED flashing" project program



③Confirm that the debug option is ST-Link;

④The compiler has no warnings or errors;

⑤Download the program to STM32;

⑥Modify the delay parameter to make the LED flash slower. Change 500 to 1000. You can see that the LED on the STM32 main control board flashes slower.

```
15      {   //三种方式实现LED闪烁
16          GPIO_ResetBits(GPIOC, GPIO_Pin_13);             //将PC13引脚置低，即将L
17          Delay_ms(500);
18          GPIO_SetBits(GPIOC, GPIO_Pin_13);
19          Delay_ms(500);
20
21          GPIO_WriteBit(GPIOC, GPIO_Pin_13, Bit_RESET);
22          Delay_ms(500);
23          GPIO_WriteBit(GPIOC, GPIO_Pin_13, Bit_SET);
24          Delay_ms(500);
25
26          GPIO_WriteBit(GPIOC, GPIO_Pin_13, (BitAction)0);//自带枚举：typedef enu
27          Delay_ms(500);
28          GPIO_WriteBit(GPIOC, GPIO_Pin_13, (BitAction)1);
29          Delay_ms(500);
```

4. Ultrasonic following car

4.1 Description

Ultrasonic ranging is a very useful and widely used measurement method. This section mainly focuses on understanding the working principle of the ultrasonic module and realizing the ultrasonic following function through programming.

4.2 Ultrasound

How does ultrasound work?

1. The transmitter (trig pin) sends a signal: high-frequency sound;

2. When the signal hits an object, it will be reflected;

3. Receiver (echo pin): Receives the signal reflected from it.

There are many methods of ultrasonic distance measurement. The principle of this system in ultrasonic measurement is: detect the transmission time of ultrasonic waves from the ultrasonic transmitter through the gas medium to the receiver,

multiply this time by the speed of sound in the air, and get the sound propagation distance.

The ultrasonic transmitter emits ultrasonic waves in a certain direction, and the MCU starts timing at the same time. The ultrasonic waves are launched in the air and return immediately when encountering obstacles on the way. The ultrasonic receiver stops timing immediately after receiving the reflected waves. According to the time T recorded by the timer , the distance (S) from the launch point to the obstacle can be calculated.

formula: **S = V * T/2**

Four factors limit the maximum measurable distance of an ultrasound system: the amplitude of the ultrasound wave, the texture of the reflector, the angle between the reflected and incident sound waves, and the sensitivity of the receiving transducer. The ability of the receiving transducer to directly receive the acoustic pulse will determine the minimum measurable distance.

4.3 Contains libraries and pin settings

①Delay function library Delay.h, microsecond Delay_us, millisecond Delay_ms, second Delay_s

②Ultrasonic sensor function library HCSR04.h, using pins Trig:PA0, Echo:PA1, timer TIM2

③Servo function library Servo.h, using pins for lower servo: PA6, upper servo: PA7, timer TIM3

④Motor function library PWM_Motor.h, using pins left motor pwm: PB6, right motor pwm: PB7, timer TIM4

Main program main.c:

Control the left motor direction: PB8, PB9;
Control the direction of the right motor: PA4, PB5;

Source code location	pin		timer	function	
HCSR04.c	Trig:PA0	Echo:PA1	TIM2	HCSR04_GetValue()	Get distance
Servo.c	Lower steering gear: PA6	Upper steering gear: PA7	TIM3	Servo_Move(Angle) Servo_Distance(Angle)	Accepts 0~180 parameters to control the servo
PWM_Motor.c	Left motor pwm: PB6	Right motor pwm: PB7	TIM4	PWM_Motor_L(Compare) PWM_Motor_R(Compare)	Accepts 0~100 integer parameters to control speed
Main program main.c	Left motor direction: PB8, PB9	Right motor direction: PA4, PB5		Forward () / Backward () TurnLeft() / TurnRight() Stop()	A function that encapsulates the movement of the car and accepts the parameters passed by the pwm function.

4.4 Follow the principle

- Get the distance in front of the ultrasonic measurement in real time and save it to the variable middleDistance
- If the distance is less than 15, the back function is executed. The car will stop when the distance is 15 to 25, the car will follow when the distance is 25 to 45, and the car will keep stopping at other distances.
- Add two flag variables: Fmod/flag

int Fmod; // forward and backward braking judgment, == 0: perform a forward motion when going backward to achieve an emergency stop effect, == 1: perform a backward motion when going forward to achieve an emergency stop effect .

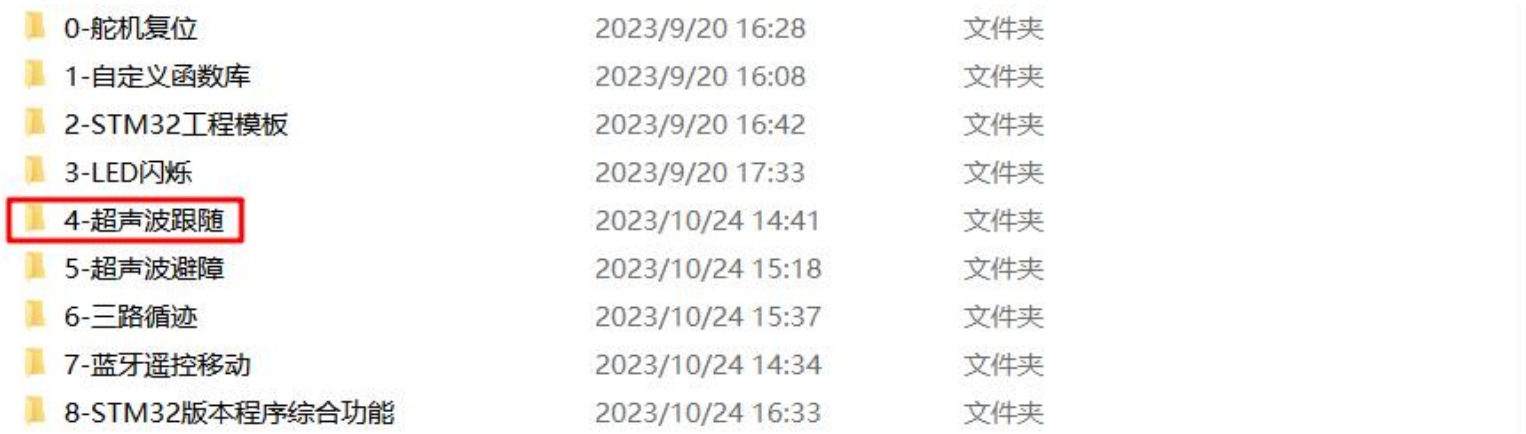
int flag;//Execution status, == 0: Braking has not been executed, == 1: Braking has been executed .

```
63 void Follow(){                            //跟随函数
64     middleDistance = HCSR04_GetValue(); //获取正前方距离值
65     int Fmod;                            //前后方向刹车判断,0：在后退情况下执行一下前进以达到急停效果，1：在前进情况下执行一下后退以达到急停效果
66     int flag;                            //执行状态，0:没有执行过刹车，1：已经执行过刹车
67     if(middleDistance <= 15){            //如果距离小于等于15cm，执行后退
68         Backward(speedL,speedR);         //后退
69         Fmod = 0;                        //标记目前属于后退
70         flag = 0;                        //标记目前没执行急刹车
71     }else if(middleDistance > 15 && middleDistance <= 25){  //如果距离大于15.且小于等于25，停车
72         if(Fmod == 0){
73             if(flag == 0){
74                 Forward(speedL,speedR); //执行50ms前进，急刹车
75                 Delay_ms(20);
76                 flag = 1;               //更改执行状态，防止再次执行
77             }
78         }else if(Fmod == 1){
```

4.5 Download the "ultrasonic follow" code

① Connect the STM32 main control board to the computer through ST-Link;

②Open the "Ultrasonic Follow" project program



③Confirm that the debug option is ST-Link;

④Compile the program without warning or error; download the program to STM32;

Program effect : The car follows the object in front to move forward or backward.

5. Ultrasonic obstacle avoidance car

5.1 Description

In the previous section, we have understood and learned the relevant knowledge of the ultrasonic module. This section mainly masters the principle of the obstacle avoidance car , and finally realizes the obstacle avoidance function through programming. The libraries, pins and functions used in this project are the same as those in the previous project, so no details will be given.

5.2 Steering gear (180°)

The steering gear (servo motor) control pulse signal period is a 20MS pulse width modulation signal (PWM), the pulse width is from 0.5ms to 2.5ms, and the corresponding steering position changes linearly from 0 to 180 degrees.

There is a reference circuit inside the steering gear, which generates a pulse signal with a period of 20ms and a width of 1.5ms. There is a comparator that compares the external signal with the reference signal to determine the direction and size, thereby generating a motor rotation signal.

5.3 Principle of obstacle avoidance

When it is judged that the distance between the ultrasonic wave and the obstacle in front is less than 35, it is an obstacle. The car stops suddenly. The servo drives the ultrasonic wave to turn left and right, and obtains the ultrasonic distance on both sides and saves it to the variables leftDistance and rightDistance . Compare the distances leftDistance and rightDistance between the left and right sides, and drive to the side with the larger distance . When you enter a dead end, that is, when both sides are less than 25, go back and obtain the distances between the left and right sides , and repeat the judgment in a loop . Otherwise, when the distance is greater than 35, the forward function forward() is executed, and the car drives forward by default.

```
103     if(middleDistance <= 35){                //超声波距离小于等于35即为遇到障碍
104         Backward(speedL,speedR);             //执行后退函数刹车
105         Delay_ms(50);
106         L_R_distance();                      //调用函数获取左、右侧距离
107         if(leftDistance > 25 && leftDistance >= rightDistance){
108             TurnLeft(speedL,speedR);
109             Delay_ms(500);
110         }else if(rightDistance >25 && rightDistance >= leftDistance){
111             TurnRight(speedL,speedR);
112             Delay_ms(500);
113         }else if(leftDistance <= 25 && rightDistance <= 25){     //进入死胡同。两边距离都小于25
114             while(leftDistance <= 25 && rightDistance <= 25){    //一直后退，直到至少有一边大于25
```

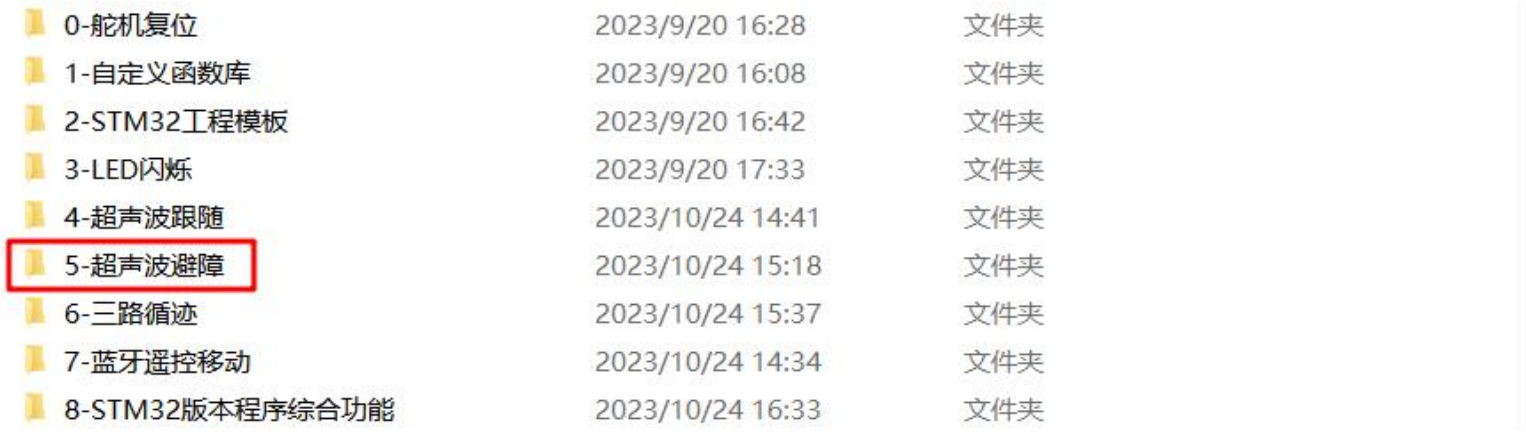
The size of the obstacle avoidance turn can be adjusted by modifying the parameter size in the delay function:

```
107         if(leftDistance > 25 && leftDistance >= rightDistance){
108             TurnLeft(speedL,speedR);
109             Delay_ms(500);
110         }else if(rightDistance >25 && rightDistance >= leftDistance){
111             TurnRight(speedL,speedR);
112             Delay_ms(500);
```

5.4 Download the "Ultrasonic Obstacle Avoidance" code

① Connect the STM32 main control board to the computer through ST-Link;

②Open the "Ultrasonic obstacle avoidance" project program



③Confirm that the debug option is ST-Link;

④Compile the program without warning or error; download the program to STM32;

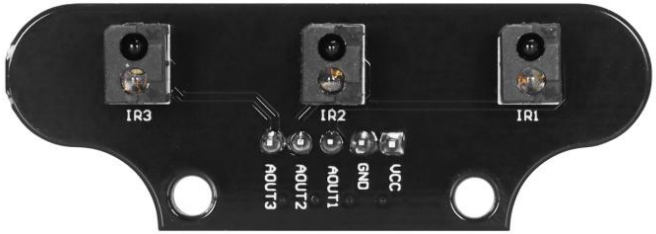
Program effect : When the car's ultrasonic wave detects obstacles ahead, it can automatically avoid obstacles and drive.

6. Three-way tracking car

6.1 Description

In this section , we will learn about the three-way tracking module, master the three-way tracking principle, and learn how to control the car to walk along the track through programming .

6.2 How does the four-channel tracking module work?

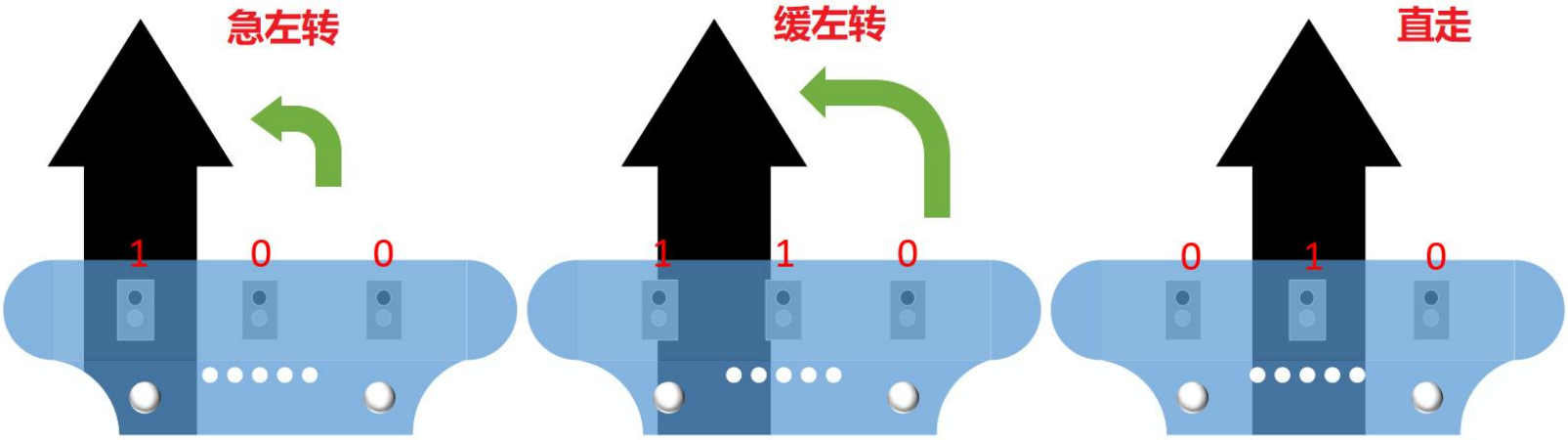


The entire module uses three ITR20001/T infrared reflection sensors. The infrared emitting diodes of the ITR2001/T sensors continuously emit infrared rays. When the emitted infrared rays are reflected by objects, they are received by the infrared receiver and the level value is output. By calculating the three The level value of the line output is used to determine the direction of the forward line.

Working voltage: 3.3V ~ 5V; Probe spacing: 16mm; Sensing distance: 1cm ~ 5cm

Notice! Since sunlight contains infrared radiation, sunlight will have a certain impact on the detected signal and should be avoided during use.

6.3 Tracking principle



When the sensor on the left side of the car detects a black line (left in the picture above), it means that the car is leaning too far to the right and needs to make a sharp left turn adjustment. Black lines are detected in the middle and left at the same time (in the picture above), which means that the car is slightly to the right and needs to be adjusted to slowly turn left. Only

the middle sensor detects the black line (right in the picture above), which means the car is in the middle of the black line and keeps going straight. The same goes for the sensor on the right.

main.c

```
    //检测到黑线为1，检测到红外反光为0
    if(T1 == 1 && T2 == 1 && T3 == 1){  //全部识别到黑线或拿起超出距离范围
        Stop();
    }else if(T1 == 0 && T2 == 1 && T3 == 0){    //中间识别到黑线
        Servo_Move(90);
        Forward(speedL,speedR);
    }else if(T1 == 1 && T2 == 0 && T3 == 0){
        Servo_Move(130);
        Forward(speedL-30,speedR+25);
    }else if(T1 == 1 && T2 == 1 && T3 == 0){
        Servo_Move(130);
        Forward(speedL-30,speedR+25);
    }else if(T1 == 0 && T2 == 1 && T3 == 1){
        Servo_Move(50);
        Forward(speedL+25,speedR-30);
    }else if(T1 == 0 && T2 == 0 && T3 == 1){
        Servo_Move(50);
        Forward(speedL+25,speedR-30);
    }else if(T1 == 0 && T2 == 0 && T3 == 0){    //全部识别到反光
        Forward(speedL,speedR);
    }
}
```

Adjust the turning size during tracking by controlling the speed of the left and right motors.

6.4 includes library and tracking module pin settings

① Three-way tracking module function library header file Tracking.h, using pins PC13/PC14/C15, three-way sensors on the left, middle and right

Source code location	pin			function	
Tracking.c	PC13	PC14	PC15	Get_State()	Get sensor level value

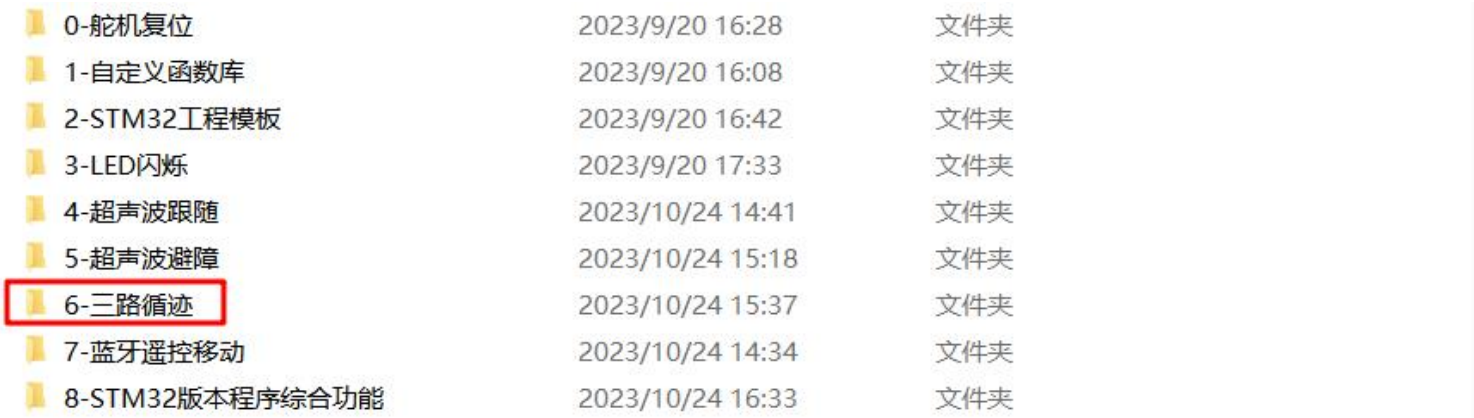
Tracking.c

```
//----------通过识别黑白线来输入高低电平的部分--------
int Get_State(uint16_t choice)
{
    uint16_t get=0;
    switch(choice)
    {
        case(1): get = GPIO_ReadInputDataBit(GPIOC,track_left);break;     //获取左边的电平
        case(2): get = GPIO_ReadInputDataBit(GPIOC,track_middle);break;   //获取中间的电平
        case(3): get = GPIO_ReadInputDataBit(GPIOC,track_right);break;    //获取右边的电平
    }
    return get;
}
```

6.5 Download the "three-way tracking" code

① Connect the STM32 main control board to the computer through ST-Link;

②Open the "Three-way Tracking" project program;



③Confirm that the debug option is ST-Link;

④Compile the program without warning or error; download the program to STM32;

Setting scene: Set up black lines on the flat ground and the turning angle should not be too large. The width of the black lines is about 1.5cm.

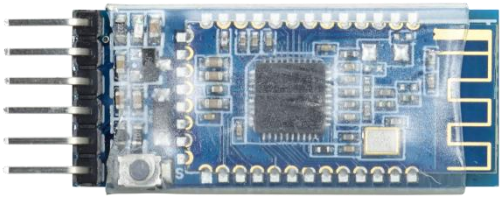
Program effect : The car follows the black line.

7. Bluetooth remote control mobile

7.1 Description

Realize basic movement or steering functions through APP remote control.

7.2 Bluetooth



Bluetooth is a wireless technology standard used to exchange data over short distances between different devices using short-wave ultra-high frequency radio waves in various fields such as industry, science and medicine. This set uses the AT-09 Bluetooth 4.0 module (BT05) as Bluetooth, which is low-power Bluetooth and has a fast and stable connection. With the APP, you can easily achieve rich effects, and the operation is simple and easy to use.

Bluetooth has 4 pins, which must be connected correctly to function, otherwise the Bluetooth will be damaged.

Note that one side of Bluetooth has been marked, and the pins are as follows:

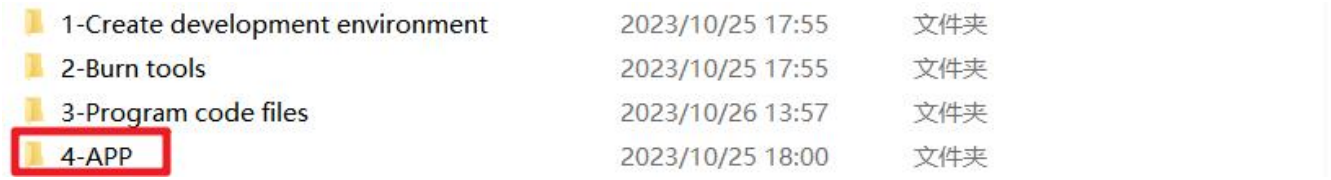
Bluetooth module	Main board
RX	TX
TX	RX
GND	G
VCC	V

7.3 Remote control car

① Burn the program to the STM32 main control board (only test the remote control "move" function, the comprehensive function is in the last code)

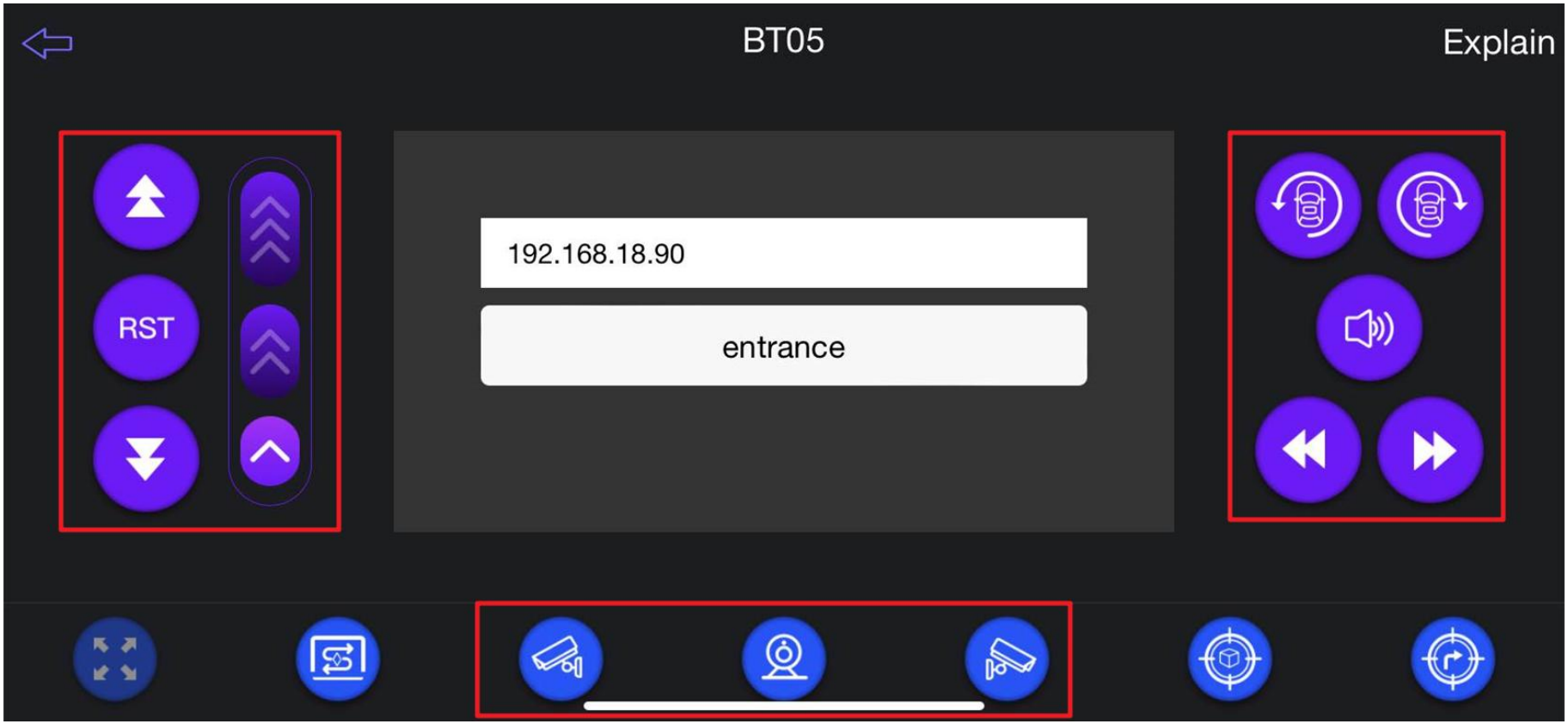
Open the code file (path: 3-Program Code File\7-Bluetooth Remote Control Mobile\Project.uvprojx)

②Go back to the previous directory and follow the "APP User Guide" in the 4-APP folder to install and connect the APP



③Start remote control car

This code program is only a basic Bluetooth remote control function and does not involve the camera and automatic mode switching functions, so only some buttons work. Please see the "Explain" description in the upper right corne.



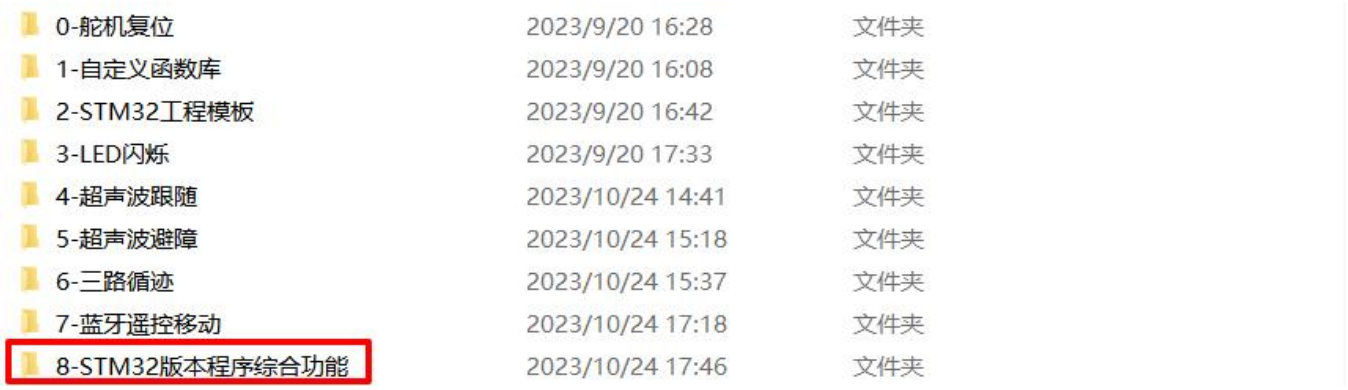
8.STM32 comprehensive functions

8.1 Description

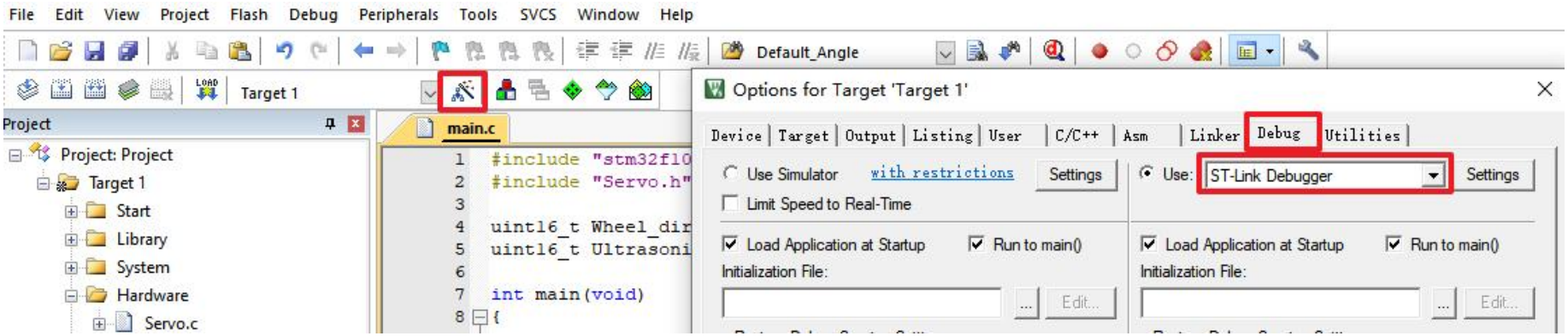
The comprehensive function program includes the functions of each mode, such as follow mode and tracking mode. Switch modes through APP remote control. By default, in manual mode, you can remotely control the car to move and control the camera direction.

8.2 Burn the program to the STM32 main control board

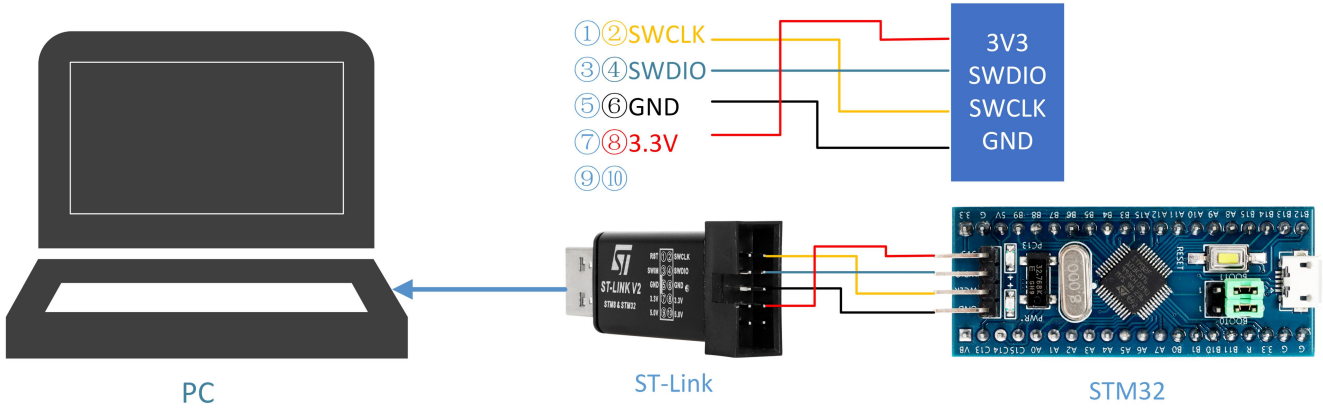
Open the code file (path: 3-Program code file\8-STM32 version program comprehensive function\Project.uvprojx)



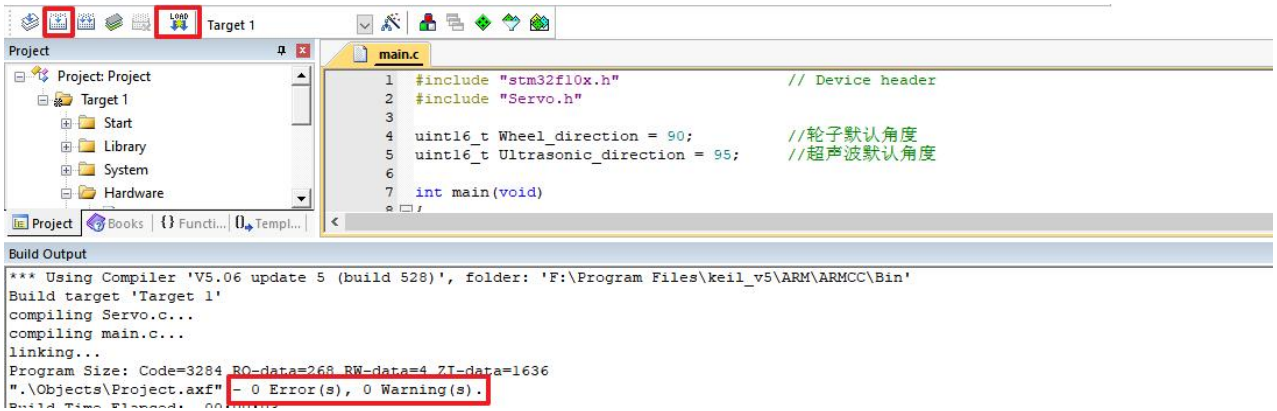
Check the Debug mode of "Target Options": ST-Link



One end of the USB of the ST-Link tool is plugged into the computer, and the other end is connected to the STM32 main control board with a 4-pin cable (note the pins to be connected)



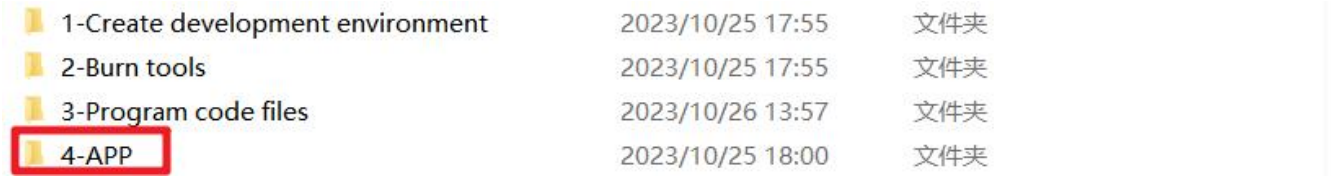
Click "Compile", confirm it is correct, and then click "Download" to download the program to STM32



8.3 Confirm that the download program is successfully connected to the ESP32 CAM and APP

① Regarding "downloading the program to ESP32 CAM", please return to the first-level directory "2-ESP32_CAM" to view the tutorial document;

② Follow the "APP User Guide" in the directory "1-STM32\4-APP" folder to install and connect the APP;



③ Check that the ESP32 CAM line is connected, the power switch is turned on, and the APP can obtain the camera image

normally;

8.4 Start remote control operation

Pay attention to the detailed operation instructions of the "Explain" button in the upper right corner.