2.4G handle control

The tutorial mainly demonstrates the remote control function of the balance car combined with the 2.4G wireless handle.

```
The tutorial only introduces the standard library project code
```

Hardware connection

Peripherals	Development board
PS2: CS	PB12
PS2: SCK	PB13
PS2: MISO	PB14
PS2: MOSI	PB15

Control principle

Control the movement state of the balance car by reading the buttons of the handle.

- 2.4G wireless handle

The 2.4G wireless handle is mainly composed of a handle and a receiver (the receiver adapter board is convenient for pin connection with the development board).

> **Handle**

Send key information to the receiver.

> **Receiver**

Receive the data sent by the handle and pass it to the development board; the development board can also configure the handle's sending mode by sending data through the handle.

Handle Button Description

Handle Button	Program Function Description	Program Definition (Part)
Direction Key (①): △	Forward	PSB_PAD_UP
Direction Key (①): ▽	Backward	PSB_PAD_DOWN
Direction Key (①): ◁	Turn Left	PSB_PAD_LEFT
Direction key (①): ▷	Turn right	PSB_PAD_RIGHT

Handle Button	Program Function Description	Program Definition (Part)
Function key (②): ▲	Forward	PSB_TRIANGLE/PSB_GREEN
Function key (②): ×	Backward	PSB_CROSS/PSB_BLUE
Function key (②): □	Turn left	PSB_SQUARE/PSB_PINK
Function key (②): ○	Turn right	PSB_CIRCLE/PSB_RED
Joystick: ③	Control direction (valid in dual light mode)	PSB_L3/PSS_LX/PSS_LY
Joystick: ④	Control direction (valid in dual light mode)	PSB_R3/PSS_RX/PSS_RY
Button (⑤): L1/L2	Accelerate	PSB_L1/PSB_L2
Button (⑥): R1/R2	Deceleration	PSB_R1/PSB_R2
Button (⑦): START	Turn off power saving mode	PSB_START
Button (⑧): MODE	Switch mode (indicator light display: single light and dual light mode)	
Button (⑨): SELECT	Not used	PSB_SELECT

Main code

The tutorial mainly explains the code for the 2.4G wireless controller remote control function. For detailed code, please refer to the corresponding project file.

> **PS2_Contorl_Car**

Control the movement state of the car by reading the data of the corresponding key value.

```
void PS2_Contorl_Car(void)
{
    //Analog value control
    g_PS2_LX = PS2_AnologData(PSS_LX);
    g_PS2_LY = PS2_AnologData(PSS_LY);
    g_PS2_RX = PS2_AnologData(PSS_RX);
    g_PS2_RY = PS2_AnologData(PSS_RY);
    g_PS2_KEY = PS2_DataKey();

    // The handle is not communicating
    if ((g_PS2_LX == 255) && (g_PS2_LY == 255) && (g_PS2_RX == 255) && (g_PS2_RY
== 255))
    {
```

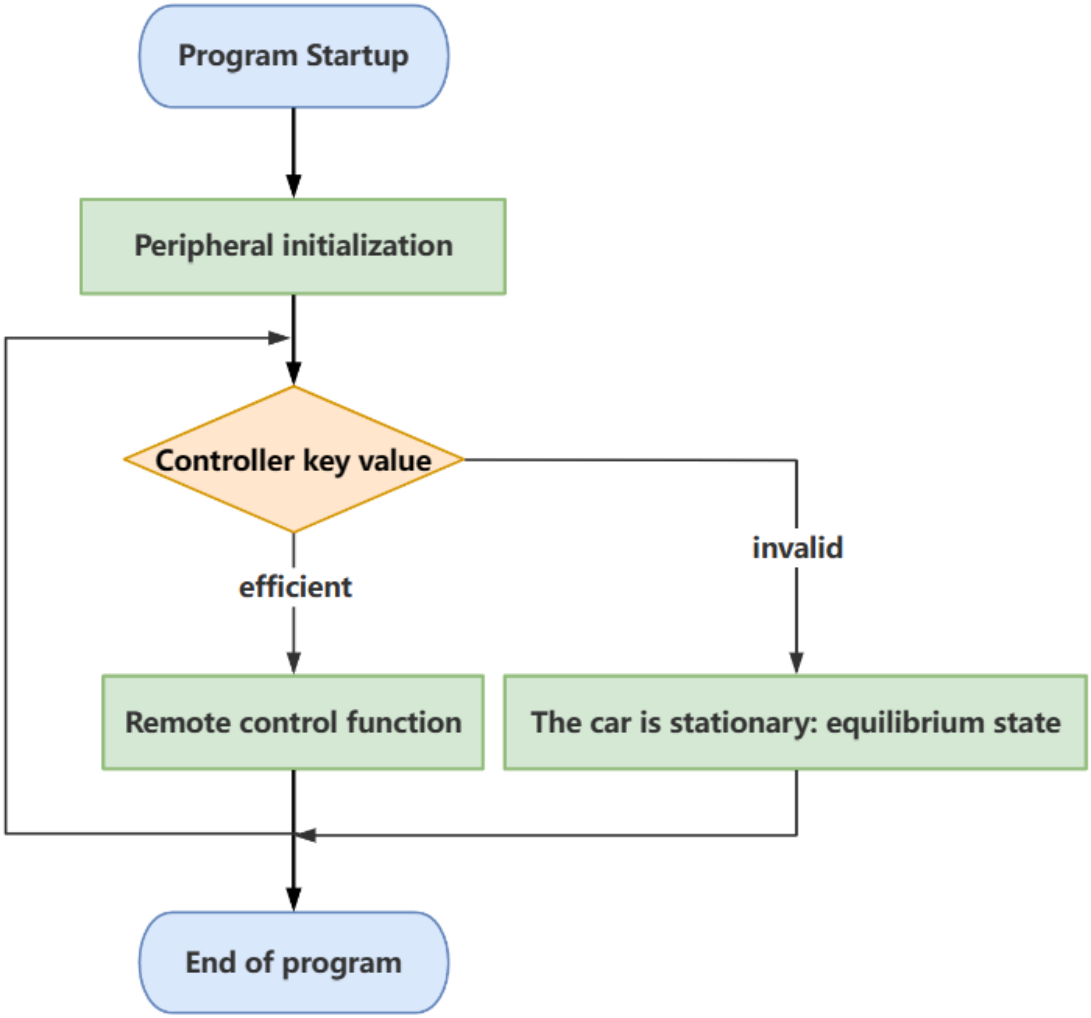
```
        if (g_flag == 1)
        {
            g_flag = 0;
        }
    }

    else
    {
        if (g_flag == 0)
        {
            g_flag = 1;
        }

        if((g_PS2_LX<50 && g_PS2_LY<50) || (g_PS2_RX<50 && g_PS2_RY<50))
        {
            g_newcarstate=enps2Fleft; // Front left
        }
        else if((g_PS2_LX>150 && g_PS2_LY<50) || (g_PS2_RX>150 && g_PS2_RY<50))
        {
            g_newcarstate=enps2Fright; // Front right turn
        }
        else if((g_PS2_LX<50 && g_PS2_LY>150) || (g_PS2_RX<50 && g_PS2_RY>150))
        {
            g_newcarstate=enps2Bleft; // Back left
        }
        else if((g_PS2_LX>150 && g_PS2_LY>150) || (g_PS2_RX>150 && g_PS2_RY>150))
        {
            g_newcarstate=enps2Bright; // Back right
        }
        else if((g_PS2_LY<90) || (g_PS2_RY<90))
        {
            g_newcarstate=enRUN; // go ahead
            return;
        }
            else if((g_PS2_LY>150) || (g_PS2_RY>150))
        {
            g_newcarstate=enBACK; // Back
            return;
        }
            else if((g_PS2_LX<90) || (g_PS2_RX<90))
        {
            g_newcarstate=enLEFT; // Turn left
            return;
        }
            else if((g_PS2_LX>150) || (g_PS2_RX>150))
        {
            g_newcarstate=enRIGHT; // Turn right
            return;
        }
        else
        {
            if (g_PS2_KEY == 0) // No buttons pressed
                g_newcarstate=enSTOP;
        }
    }
```

```
    // The following are key value controls
    switch (g_PS2_KEY)
    {
        case PSB_L1:
        case PSB_L2:
            speed_flag ++;
            if(speed_flag>5)
            {
                speed_flag = 5;
            }

            // real control
            Car_Target_Velocity +=10 ;
            Car_Turn_Amplitude_speed +=12 ;
            if(Car_Target_Velocity > 50)
            {
                Car_Target_Velocity = 50;
            }
            if(Car_Turn_Amplitude_speed>60)
            {
                Car_Turn_Amplitude_speed =60;
            }
            break;


        case PSB_R1:
        case PSB_R2:
            speed_flag --;
            if(speed_flag<1)
            {
                speed_flag = 1;
            }

            // real control
            Car_Target_Velocity -=10 ;
            Car_Turn_Amplitude_speed -=12 ;
            if(Car_Target_Velocity < 10)
            {
                Car_Target_Velocity = 10;
            }
            if(Car_Turn_Amplitude_speed<12)
            {
                Car_Turn_Amplitude_speed =12;
            }
            break;

        // forward
        case PSB_PAD_UP:
        case PSB_GREEN:
                g_newcarstate=enRUN;
                break;

        // back
        case PSB_PAD_RIGHT:
        case PSB_RED:
            g_newcarstate=enRIGHT;
```

```
            break;

        // left
        case PSB_PAD_DOWN:
        case PSB_BLUE:
            g_newcarstate=enBACK;

            break;

        // right
        case PSB_PAD_LEFT:
        case PSB_PINK:
            g_newcarstate=enLEFT;
            break;

        // stop
        default:
            if(g_flag == 1) // Red and green mode to judge
            {
                if(((g_PS2_LY >90 && g_PS2_LY < 140) && (g_PS2_LX >90 && g_PS2_LX
< 140)) && ((g_PS2_RY >90 && g_PS2_RY < 140) && (g_PS2_RX >90 && g_PS2_RX <
140)))
                {
                    g_newcarstate=enSTOP;
                    break;
                }
            }
            else
            {
                g_newcarstate=enSTOP;
            }
    }
}
```

Program flow chart

Brief introduction to the function implementation process:



Experimental phenomenon

> **Software code**

The Handle_Control_Car.hex file generated by the project compilation is located in the OBJ folder of the Handle_Control_Car project. Find the Handle_Control_Car.hex file corresponding to the project and use the FlyMcu software to download the program into the development board.

```
Product supporting information source code path: Attachment → Source code summary
→ 5.Balanced_Car_Extended → 03.Handle_Control_Car
```

> **Experimental phenomenon**

After the program is started, press the KEY1 button according to the OLED prompt to start the handle remote control function of the balance car: OLED will display the speed level in real time!

```
The program has voltage detection. If the voltage is less than 9.6V, the low
voltage alarm will be triggered and the buzzer will sound.
Common situations for triggering voltage alarms:
1. The power switch of the development board is not turned on, and only the Type-
C data cable is connected for power supply
2. The battery pack voltage is lower than 9.6V and needs to be charged in time
```