4-channel tracking

The tutorial mainly demonstrates the patrol function of the balance car combined with the four-way tracking module.

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

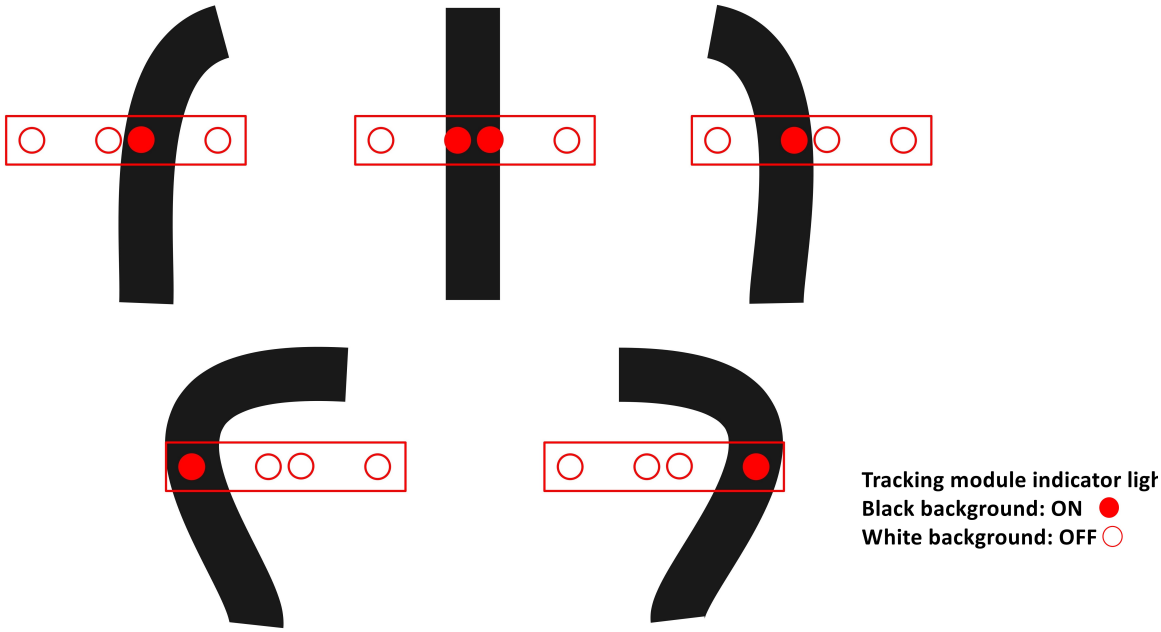
Hardware connection

LED corresponding to the four-way patrol module	Four-way patrol module	Development board
	VCC	5V/3.3V
L1	X1	PC4
L2	X2	PC5
L3	X3	PB0
L4	X4	PB1
	GND	GND

Control principle

By reading the X1, X2, X3, and X4 interface levels of the four-way tracking module, it is determined whether the black line is detected and where the black line is located.

```
The program uses PID to implement the line patrol function. If the PID parameters
are not used, the effect is poor.
```



> **Black line detected**

The light is on → The corresponding interface of the four-way line patrol module outputs a low level;

> **White line detected**

The light is off → The corresponding interface of the four-way line patrol module outputs a high level.

```
Note: The corresponding relationship between the interface, LED, and adjustment knob shall be based on the silkscreen numbers, for example: X1 corresponds to L1, SW1
```

Main code

The tutorial mainly explains the code for the four-way patrol function. For detailed code, refer to the corresponding project file.

> **Turn_IRTrack_PD**

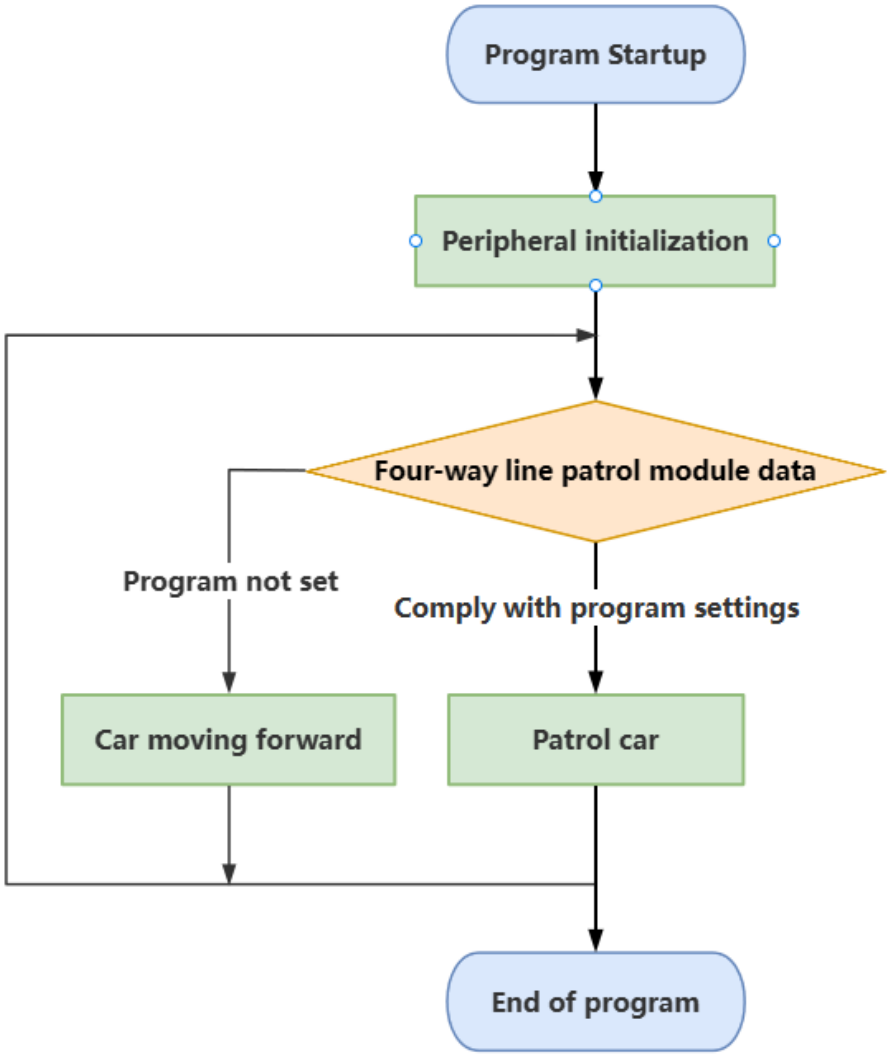
Four-way patrol module patrol PID implementation function. If the patrol effect is not good, modify the PID parameters of the app_tracking.c file. It is not recommended to modify the PID parameters of the pid_control.c file (the PID parameters of the pid_control.c file are subject to the parameters finally confirmed in the balance car parameter adjustment tutorial).

```
int Turn_IRTrack_PD(float gyro)
{
    int IRTrackTurn = 0;
    float err = 0;
    static float IRTrack_Integral;

    PID_track_get();  // Obtain deviation

    err=error-IRTrack_Minddle;

    IRTrack_Integral +=err;
```

```
IRTrackTurn=err*IRTrack_Trun_KP+IRTrack_Trun_KI*IRTrack_Integral+gyro*IRTrack_Tru
n_KD;

    return IRTrackTurn;
}
```

Program flow chart

Briefly introduce the process of function implementation:



Experimental phenomenon

> **Software code**

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

```
Product supporting data source code path: Attachment → Source code summary →
5.Balanced_Car_Extended → 01.BalancedCar_tracking_PID
```

> **Experimental phenomenon**

After the program is started, put the car on the patrol map, and press KEY1 according to the OLED prompt to start the balance car patrol function: OLED will display the deviation setting value and the level of the four-way patrol module in real time!

The balance car will balance first and then patrol the line. When the inclination angle is greater than a certain value, the car will stop moving.

```
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
```

Extended case: Four-way patrol (high-difficulty map)

This case is used with our smart car patrol track. It is difficult for novices to adjust, so only code reference is provided.

```
Product supporting information source code path: Attachment → Source code summary
→ 5.Balanced_Car_Extended → 01.BalancedCar_tracking_PID →
BalancedCar_tracking_yahboom_map
```