5. Draw a circle to make a prison

1. Learning objectives

Take the drive of the Mailun car as an example. We used the four-way patrol module to patrol the line before. Now we use this feature to surround the car with black lines, so that the car always moves in the circle we surrounded.

2. Experimental preparation

1. The car wiring has been installed and installed correctly
2. Debug the four-way patrol module. The indicator light is on when encountering a black line. The indicator light is off when it is not a black line.

3. Implementation principle

The basic principle of infrared sensor patrol is to use the reflective properties of objects. Our experiment is to patrol the black line on a white background. When the infrared ray is emitted to the black line, it will be absorbed by the black line. When it is emitted to materials of other colors, it will be reflected back to the infrared receiving tube.

Through the four-way patrol module, we let the car turn left or right when it encounters a black line. It can only go straight when it is all white. It can also be imagined as a kind of obstacle avoidance, with the obstacle being a black line.

4. Code analysis

Source code path:

/home/pi/project_demo/05.Comprehensive_gameplay/5.car_geofencing.ipynb

```python
import sys
import time

sys.path.append('/home/pi/project_demo/lib')
# 导入麦克纳姆小车驱动库 Import Mecanum Car Driver Library
from McLumk_Wheel_Sports import *

speed = 25#25

try:
    while True:
        # 从I2C读取巡线传感器数据 Read line sensor data from I2C
        track_data = bot.read_data_array(0x0a, 1)
        track = int(track_data[0])
```

```python
        # 解析巡线传感器的状态 Analyze the status of the line patrol sensor
        x1 = (track >> 3) & 0x01
        x2 = (track >> 2) & 0x01
        x3 = (track >> 1) & 0x01
        x4 = track & 0x01
        """
        X2 X1 X3 X4
        |  |  |  |
        L1 L2 R1 R2
        """
        lineL1 = x2
        lineL2 = x1
        lineR1 = x3
        lineR2 = x4

        # 0000
        if not lineL1 and not lineL2 and not lineR1 and not lineR2:
            rotate_right(speed)
            time.sleep(1)
        # 1X00
        elif lineL1 and not lineR1 and not lineR2:
            rotate_left(speed-10)
            time.sleep(1)
        # 00X1
        elif not lineL1 and not lineL2 and lineR2:
            rotate_right(speed-10)
            time.sleep(1)
        # 10X1
        elif lineL1 and not lineL2 and lineR2:
            rotate_right(speed-10)
            time.sleep(1)
        # 1X01
        elif lineL1 and not lineR1 and lineR2:
            rotate_right(speed-10)
            time.sleep(1)
        # 0110
        elif not lineL1 and lineL2 and lineR1 and not lineR2:
            rotate_right(speed-10)
            time.sleep(1)
        # 0111
        elif not lineL1 and lineL2 and lineR1 and lineR2:
            rotate_right(speed-20)
            time.sleep(1)
        # 1110
        elif lineL1 and lineL2 and lineR1 and not lineR2:
            rotate_left(speed-20)
            time.sleep(1)
        # 1111
        elif lineL1 and lineL2 and lineR1 and lineR2:
            move_forward(speed)

        # 等待一段时间再进行下一次检测 Wait for a while before the next test
        time.sleep(0.01)

except KeyboardInterrupt:
    # 当用户中断程序时，确保所有电机停止 Ensure that all motors stop when the user
interrupts the program
    stop_robot()
```

```
    print("Ending")
```

```
stop_robot()
# 清理资源 Cleaning up resources
del bot
```

5. Experimental results

We put the car on a map surrounded by black lines, make sure the car's four-way patrol module has been set to light up when encountering a black line, and turn off when not encountering a black line, and then run the program. After running the program, we can see that when the car does not encounter a black line on the white map, the car goes straight; when it encounters a black line on the edge of the map, the car will turn right or left, always moving within the black circle.