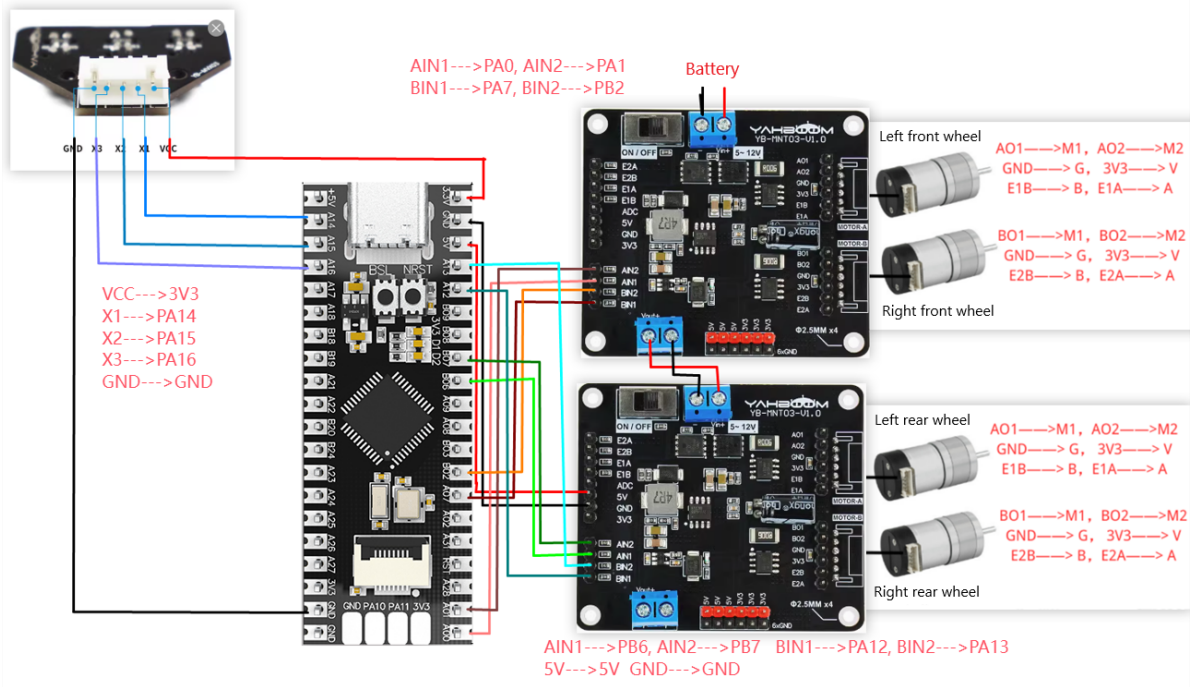
Tracking robot car

1. Learning Objectives

The three-way line patrol module enables the car to patrol the line

2. Hardware Connection

Wiring of three-way line patrol module, MSPM0G3507 and two-way motor driver board. The motor used is 310 motor.



Lower drive plate (left front wheel, right front wheel)	MSPM0G3507		Upper drive plate (left rear wheel, right rear wheel)	MSPM0G3507
AIN1	PA0		AIN1	PB6
AIN2	PA1		AIN2	PB7
BIN1	PA7		BIN1	PA12
BIN2	PB2		BIN2	PA13
Three-way line patrol module	**MSPM0G3507**		5V	5V
VCC	3V3		GND	GND
X1	PA14			
X2	PA15			
X3	PA16			
GND	GND			

3. Program Description

- three_linewalking.h

```
/*

    从车身后面往前看：  左侧到右边巡线传感器顺序为  L  M  R

*/

#define LOW     (0)
#define HIGH    (1)


#define LineWalk_L_IN       ( ( ( DL_GPIO_readPins(Sensor_PORT,Sensor_X3_PIN) &
Sensor_X3_PIN ) > 0 ) ? 1 : 0 )
#define LineWalk_M_IN       ( ( ( DL_GPIO_readPins(Sensor_PORT,Sensor_X2_PIN) &
Sensor_X2_PIN ) > 0 ) ? 1 : 0 )
#define LineWalk_R_IN       ( ( ( DL_GPIO_readPins(Sensor_PORT,Sensor_X1_PIN) &
Sensor_X1_PIN ) > 0 ) ? 1 : 0 )
```

The macro definition reads the electrical levels of the three pins of the three-way line patrol module. When looking from the back of the vehicle body, that is, facing away from the module's lamp beads, the order from left to right is L M R.

- three_linewalking.c

```
/**
* @brief         获取巡线状态 Get line patrol status
* @param[in]     int *p_iL, int *p_iM,  int *p_iR  三路巡线位指针  Three-way line
patrol pointer
* @param[out]    void
* @retval        void
*/
void three_GetLineWalking(int *p_iL, int *p_iM, int *p_iR)
{
    *p_iL = LineWalk_L_IN;
    *p_iM = LineWalk_M_IN;
    *p_iR = LineWalk_R_IN;
    printf("L = %d M = %d R = %d\r\n",*p_iL,*p_iM,*p_iR);
}

/**
* @brief         巡线模式运动 Line following mode movement
* @param[in]     void
* @param[out]    void
* @retval        void
*/
void three_LineWalking(void)
{
    int LineL = 0, LineM = 0, LineR = 0;
```

```
    three_GetLineWalking(&LineL, &LineM, &LineR);   //获取黑线检测状态  Get black
line detection status
    Motion_Set_Pwm(0,0,0,0);
  if( LineL == HIGH &&  LineM == HIGH) //继续直行   Continue straight
    {
        Motion_Set_Pwm(800,800,800,800);
    }
    else if ( LineM == HIGH && LineR == HIGH) //继续直行    Continue straight
    {
        Motion_Set_Pwm(800,800,800,800);
    }
    else if (LineL == HIGH ) //左转   Turn left
    {
        Motion_Set_Pwm(-800,1100,-800,1100);
    }
    else if (LineR == HIGH) //右转    Turn right
    {
        Motion_Set_Pwm(1100,-800,1100,-800);
    }
    else //if(LineM == HIGH ) //前进  go ahead
    {
        Motion_Set_Pwm(900,900,900,900);
    }

}
```

- three_GetLineWalking: Read the levels of the three pins on the module.
- three_LineWalking: Directly determine the position of the car based on the obtained level value and change the rotation of the motor.

Note: The control logic of this code is only applicable to oval tracks. If it is other tracks, you need to modify the code yourself to adapt.

- bsp_motor.h

```
#define PWM_M1_A(value)
DL_TimerG_setCaptureCompareValue(PWM_0_INST,value,GPIO_PWM_0_C0_IDX);
#define PWM_M1_B(value)
DL_TimerG_setCaptureCompareValue(PWM_0_INST,value,GPIO_PWM_0_C1_IDX);

#define PWM_M2_A(value)
DL_TimerG_setCaptureCompareValue(PWM_0_INST,value,GPIO_PWM_0_C2_IDX);
#define PWM_M2_B(value)
DL_TimerG_setCaptureCompareValue(PWM_0_INST,value,GPIO_PWM_0_C3_IDX);

#define PWM_M3_A(value)
DL_TimerG_setCaptureCompareValue(PWM_1_INST,value,GPIO_PWM_1_C0_IDX);
#define PWM_M3_B(value)
DL_TimerG_setCaptureCompareValue(PWM_1_INST,value,GPIO_PWM_1_C1_IDX);

#define PWM_M4_A(value)
DL_TimerG_setCaptureCompareValue(PWM_2_INST,value,GPIO_PWM_2_C0_IDX);
#define PWM_M4_B(value)
DL_TimerG_setCaptureCompareValue(PWM_2_INST,value,GPIO_PWM_2_C1_IDX);
```

Define the PWM duty cycle function for the four motors.

- bsp_motor.c

```
// 所有电机停止 All motors stopped
void Motor_Stop(uint8_t brake)
{
    if (brake != 0) brake = 1;
    PWM_M1_A(brake * MOTOR_MAX_PULSE);
    PWM_M1_B(brake * MOTOR_MAX_PULSE);
    PWM_M2_A(brake * MOTOR_MAX_PULSE);
    PWM_M2_B(brake * MOTOR_MAX_PULSE);

    PWM_M3_A(brake * MOTOR_MAX_PULSE);
    PWM_M3_B(brake * MOTOR_MAX_PULSE);
    PWM_M4_A(brake * MOTOR_MAX_PULSE);
    PWM_M4_B(brake * MOTOR_MAX_PULSE);
}

// 设置电机速度，speed:±3200, 0为停止   // Set the motor speed, speed: ±3200, 0
means stop
void Motor_Set_Pwm(uint8_t id, int16_t speed)
{
    int16_t pulse = Motor_Ignore_Dead_Zone(speed);
    // 限制输入 Limit Input
    if (pulse >= MOTOR_MAX_PULSE)
        pulse = MOTOR_MAX_PULSE;
    if (pulse <= -MOTOR_MAX_PULSE)
        pulse = -MOTOR_MAX_PULSE;

    switch (id)
    {
    case MOTOR_ID_M1:
    {
        if (pulse >= 0)
        {
            PWM_M1_A(pulse);
            PWM_M1_B(0);
        }
        else
        {
            PWM_M1_A(0);
            PWM_M1_B(-pulse);
        }
        break;
    }
    case MOTOR_ID_M2:
    {
        pulse = -pulse;
        if (pulse >= 0)
        {
            PWM_M2_A(pulse);
            PWM_M2_B(0);
        }
        else
        {
```

```
            PWM_M2_A(0);
            PWM_M2_B(-pulse);
        }
        break;
    }

    case MOTOR_ID_M3:
    {
        if (pulse >= 0)
        {
            PWM_M3_A(pulse);
            PWM_M3_B(0);
        }
        else
        {
            PWM_M3_A(0);
            PWM_M3_B(-pulse);
        }
        break;
    }
    case MOTOR_ID_M4:
    {
        pulse = -pulse;
        if (pulse >= 0)
        {
            PWM_M4_A(pulse);
            PWM_M4_B(0);
        }
        else
        {
            PWM_M4_A(0);
            PWM_M4_B(-pulse);
        }
        break;
    }

    default:
        break;
    }
}
```

Give the input value to the motor to control the forward and reverse rotation and speed of the motor. This control logic is not suitable for all motors and motor driver boards. If you are using the motor driver board and motor mentioned in this tutorial, if the direction of the car is not correct, you need to check the wiring again. If you are using a motor driver board and motor not mentioned in this tutorial, you need to modify it according to your own situation.

- app_motor.c

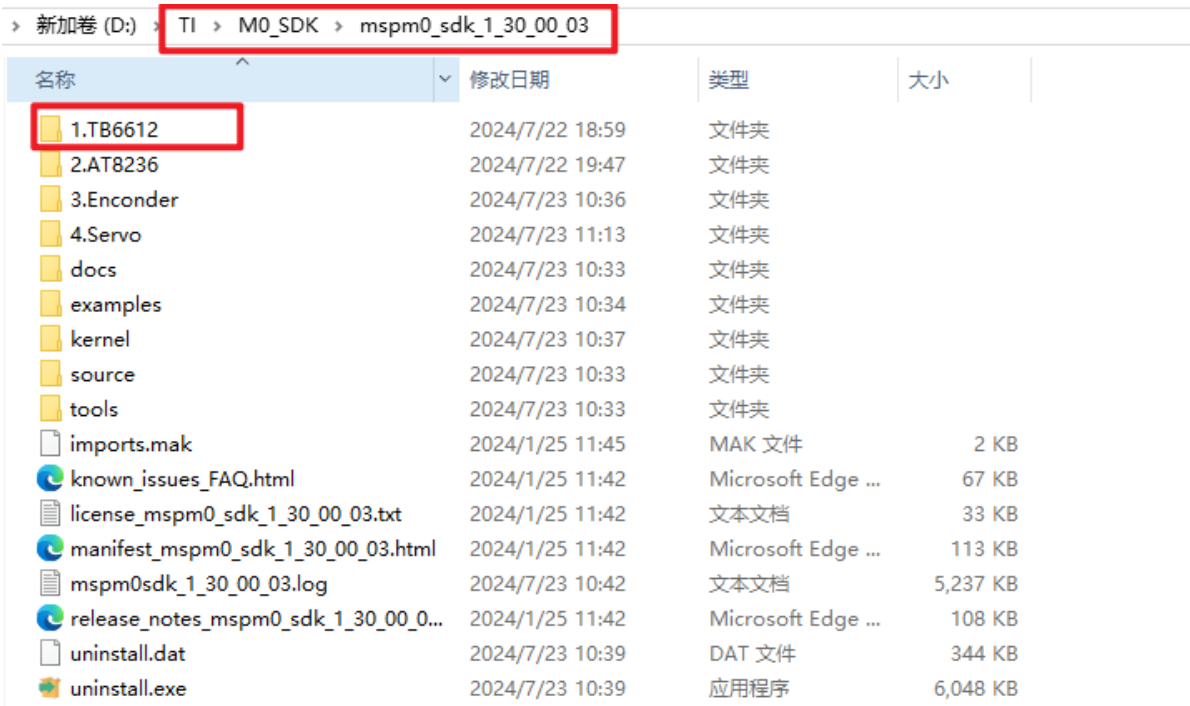
```
// 控制小车运动，Motor_X=[-3200, 3200]，超过范围则无效。     // Control the movement of
the car, Motor_X=[-3200, 3200]. It will be invalid if it exceeds the range.
void Motion_Set_Pwm(int16_t Motor_1, int16_t Motor_2, int16_t Motor_3, int16_t
Motor_4)
{
    if (Motor_1 >= -MOTOR_MAX_PULSE && Motor_1 <= MOTOR_MAX_PULSE)
```

```
    {
        Motor_Set_Pwm(MOTOR_ID_M1, Motor_1);
    }
    if (Motor_2 >= -MOTOR_MAX_PULSE && Motor_2 <= MOTOR_MAX_PULSE)
    {
        Motor_Set_Pwm(MOTOR_ID_M2, Motor_2);
    }
    if (Motor_3 >= -MOTOR_MAX_PULSE && Motor_3 <= MOTOR_MAX_PULSE)
    {
        Motor_Set_Pwm(MOTOR_ID_M3, Motor_3);
    }
    if (Motor_4 >= -MOTOR_MAX_PULSE && Motor_4 <= MOTOR_MAX_PULSE)
    {
        Motor_Set_Pwm(MOTOR_ID_M4, Motor_4);
    }
}
```

The input value of the motor cannot exceed the limit range.

Note: The project source code must be placed in the SDK path for compilation,

For example, the path:D:\TI\M0_SDK\mspm0_sdk_1_30_00_03\1.TB6612



4. Experimental Phenomena

Burn the line patrol program to MSPM0G3507. Patiently connect the wires according to the wiring diagram. After connecting the wires, patiently check whether the wires are connected correctly. If you do not check, the car may not move at best, or the board may be burned directly. After confirming that everything is correct, turn on the upper and lower drive board switches, and place the car on the black line ellipse map with a white background. Finally, you can see that the car moves along the black line. **Because the line patrol module will be affected by strong light, please make sure that there is no outdoor strong light interference when patrolling the line.**