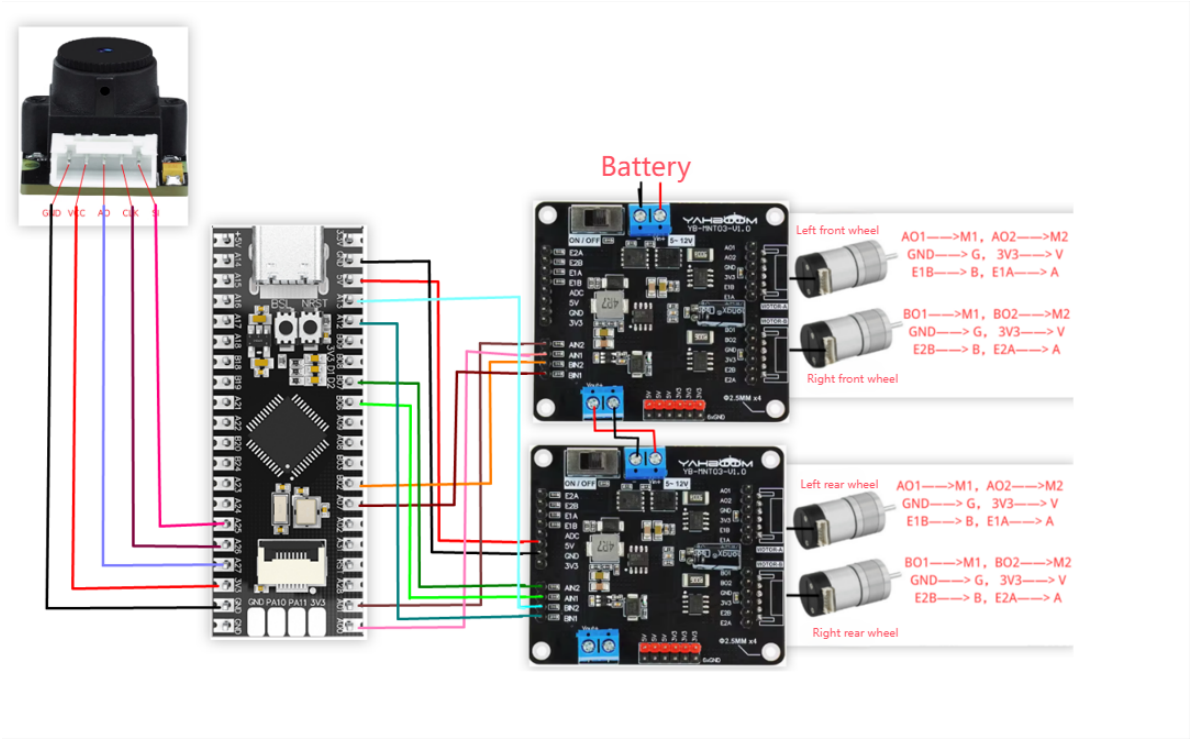
CCD tracking-Mini chassis (purchased separately)

1. Learning objectives

Realize the line patrol movement of the car through the linear CCD module combined with the car chassis

2. Hardware connection

The wiring of the linear CCD module, MSPM0G3507 and the dual-channel motor drive board



Lower driver board (left front wheel, right front wheel)	MSPM0G3507		Upper driver board (left rear wheel, right rear wheel)	MSPM0G3507
AIN1	PA0		AIN1	PB6
AIN2	PA1		AIN2	PB7
BIN1	PA7		BIN1	PA12
BIN2	PB2		BIN2	PA13
Linear CCD module	**MSPM0G3507**		5V	5V
GND	GND		GND	GND
VCC	3V3			
AO	PA27			
CLK	PA26			
SI	PA25			

3. Program Description

Tracking version without PID:

- ccd.c

```
void use_ccd_line_motion(void)
{
    if(CCD_Zhongzhi >80 && CCD_Zhongzhi<110) //假设黑线在摄像头的右边  Assume the black line is on the right side of the camera
    {
        Motion_Set_Pwm(700,-700,700,-700);//往右调整    Adjust right
    }
    else if(CCD_Zhongzhi >110 && CCD_Zhongzhi<128)
    {
        Motion_Set_Pwm(700,-700,700,-700);//往右调整    Adjust right
    }

    if(CCD_Zhongzhi >30 && CCD_Zhongzhi<=50) //假设黑线在摄像头的左边  Assume the black line is on the left side of the camera
    {
        Motion_Set_Pwm(-700,700,-700,700);//往左调整    Adjust left
    }
    else if(CCD_Zhongzhi<=30)
    {
        Motion_Set_Pwm(-700,700,-700,700);//往左调整Adjust left
    }

    if(CCD_Zhongzhi >50 && CCD_Zhongzhi<80) //直走    Go straight
    {
        Motion_Set_Pwm(500,500,500,500);
    }
}
```

Directly use the median value obtained after the algorithm to determine the position, and then directly drive the motor to move. If you burn directly according to the code provided in this case, in most cases you cannot directly run the perfect effect. The median value here needs to be modified to a suitable range according to your own ambient light conditions and track conditions.

- 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], if it exceeds the
range, it is invalid.
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);
    }
}
```

Limit the motor input value to not exceed the limit range.

- bsp_motor.h

```
// 设置电机速度，speed:±（3200-MOTOR_IGNORE_PULSE）, 0为停止
// Set motor speed, speed:± (3200-MOTOR_IGNORE_PULSE), 0 indicates stop
void Motor_Set_Pwm(uint8_t id, int16_t speed)
{
//  int16_t pulse = 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.

Note: The non-PID version is intended to provide the simplest CCD tracking control idea, and the effect may not be ideal. If you have higher requirements for the effect, you can use the code with PID version.

There are PID versions:

- ccd.c

```
void use_ccd_line_motion_PID(void)
{
    g_ccd_median=CCD_Zhongzhi - 55;
    pid_output_ele = (int)(APP_ELE_PID_Calc(g_ccd_median));//位置式PID Position
PID
    motion_car_control(CCD_SPEED, 0, pid_output_ele);//直接控制电机       Direct
motor control

}
```

In this version, the main function is replaced with a line patrol function with a PID version. The 55 here is just the median value used in the environment where I tested it. You may need to modify it according to your actual situation. The calculated error is calculated by the position PID algorithm to control the movement of the motor.

- app_motor.h

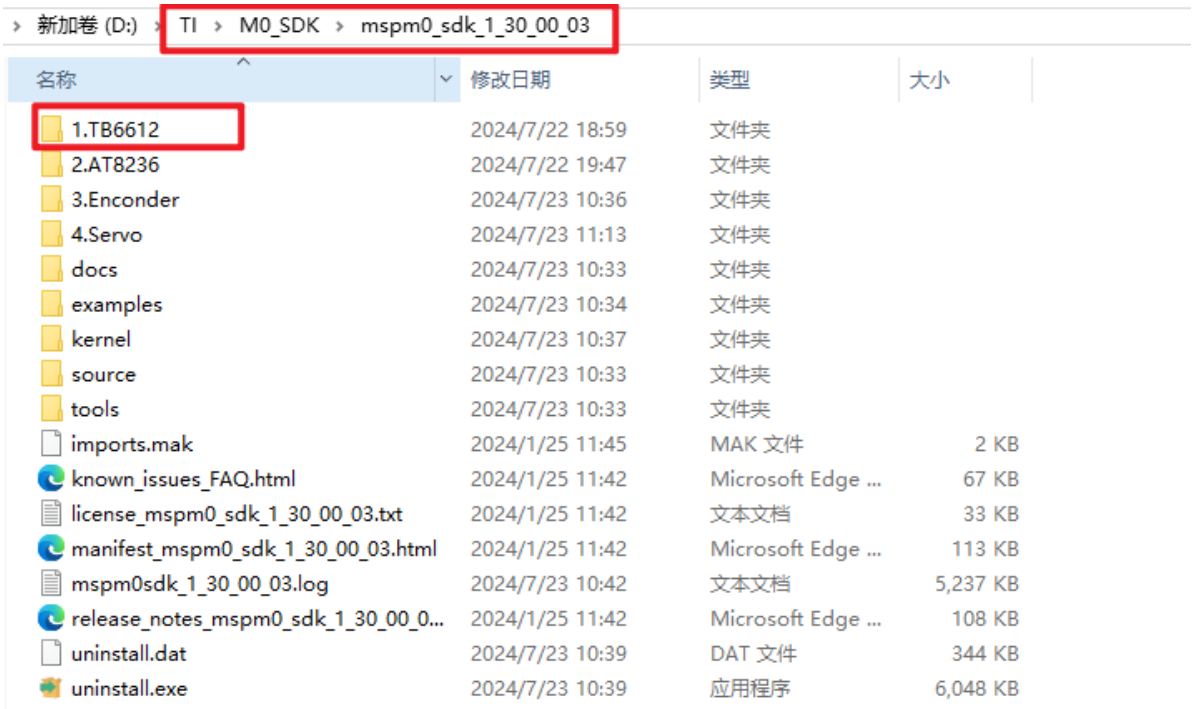
```
float APP_ELE_PID_Calc(int8_t actual_value)
{
    float pid_out = 0;
    int8_t error;
    static int8_t error_last=0;
    static float Integral;//积分    integral

    error=actual_value-MID_ERR;

    Integral +=error;

    //位置式pid Positional pid
    pid_out=error*CCD_PID_KP
                            +CCD_PID_KI*Integral
                            +(error - error_last)*CCD_PID_KD;

    return pid_out;
}
```

```
//直接控制pwm   Directly control pwm
void motion_car_control(int16_t V_x, int16_t V_y, int16_t V_z)
{
    float robot_APB = Motion_Get_APB();
    speed_lr = 0;
    speed_fb = V_x;
    speed_spin = (V_z / 1000.0f) * robot_APB;
    if (V_x == 0 && V_y == 0 && V_z == 0)
    {
        Motion_Set_Pwm(0,0,0,0);
        return;
    }


    speed_L1_setup = speed_fb + speed_spin;
    speed_R1_setup = speed_fb + speed_spin;
    speed_L2_setup = speed_fb  - speed_spin;
    speed_R2_setup = speed_fb  - speed_spin;

    if (speed_L1_setup > 1000) speed_L1_setup = 1000;
    if (speed_L1_setup < -1000) speed_L1_setup = -1000;
    if (speed_L2_setup > 1000) speed_L2_setup = 1000;
    if (speed_L2_setup < -1000) speed_L2_setup = -1000;
    if (speed_R1_setup > 1000) speed_R1_setup = 1000;
    if (speed_R1_setup < -1000) speed_R1_setup = -1000;
    if (speed_R2_setup > 1000) speed_R2_setup = 1000;
    if (speed_R2_setup < -1000) speed_R2_setup = -1000;

    Motion_Set_Pwm(speed_L1_setup, speed_L2_setup, speed_R1_setup,
speed_R2_setup);

}
```

- APP_ELE_PID_Calc：Position PID calculation function
- motion_car_control：According to the PWM output by the PID algorithm, the motor is driven to make the car follow the track.

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

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



IV. Experimental Phenomenon

Burn the line patrol program into MSPM0G3507. Patiently connect the wires according to the wiring diagram. After connecting the wires, please check whether the wires are connected correctly. If you do not check, the car may not move at best, or the board may burn 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 will move 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.**

Note: There are two folders in the source code. Nopid means that the pid algorithm is not used; pid means that the pid algorithm is used.