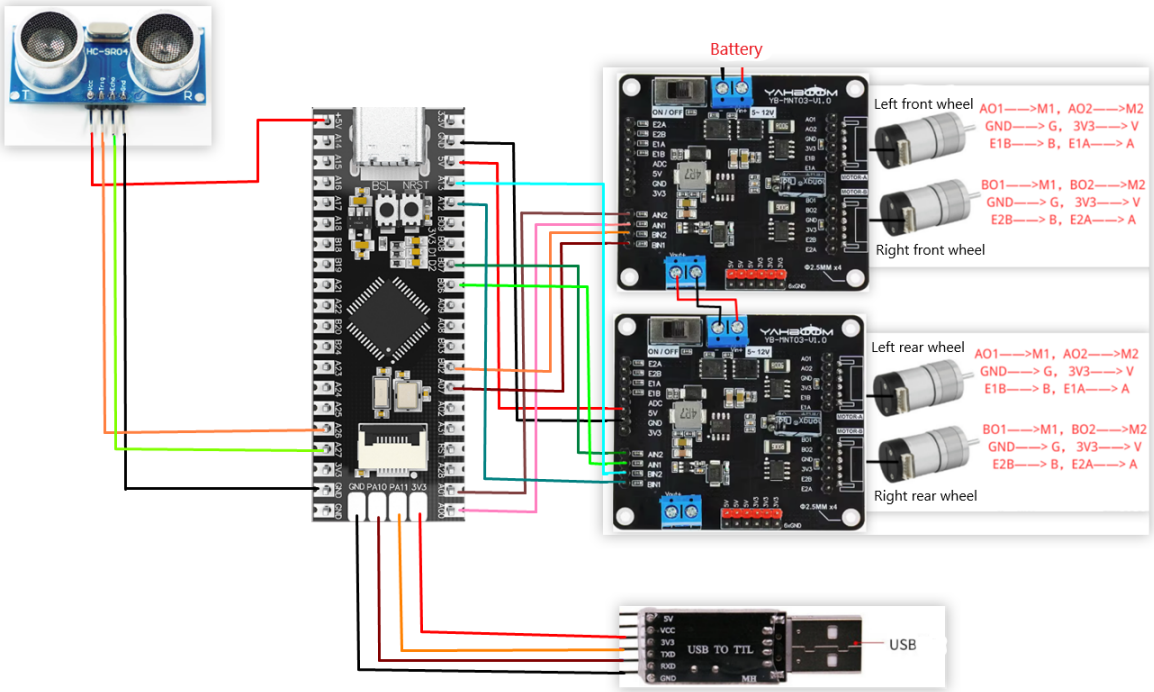
Ultrasonic obstacle avoidance

1. Learning Objectives

Obstacles are detected through ultrasonic modules and obstacle avoidance function is added.

2. Hardware Connection

The connection between the ultrasonic module, MSPM0G3507 and the dual-channel motor driver board.



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
Ultrasonic module	**MSPM0G3507**		5V	5V
VCC	5V		GND	GND
Trig	PA26			
Echo	PA27			
GND	GND			

3. Program Description

- empty.c

```
int main(void)
{
    //开发板初始化  Development board initialization
    USART_Init();
    Ultrasonic_Init();

    while(1)
    {
        uint32_t Value = (int)Hcsr04GetLength();
//      printf((const char *)"Distance = %dCM\r\n", Value);

        if(Value>30)//前方30CM以内无障碍物时 When there is no obstacle within
30CM in front
        {
            Motion_Set_Pwm(100,100,100,100);//前进    Forward
            delay_ms(100);
        }
        else//当检测到前方障碍物小于30CM   When the obstacle ahead is detected
to be less than 30CM
        {
            Motion_Set_Pwm(-100,-100,-100,-100);//先后退，后右转   Back up
first, then turn right
            delay_ms(500);
            Motion_Set_Pwm(700,-300,700,-300);
            delay_ms(500);
        }
    }
}
```

In the while loop of the main program, the distance detected by the current ultrasound is continuously obtained. When it is detected that there is no obstacle within 30CM, the car will continue to move forward. When an obstacle is detected within 30CM, it will first retreat for 500 milliseconds and then turn right for 500 milliseconds.

- 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

```
// 设置电机速度，speed:±（1000-MOTOR_IGNORE_PULSE）, 0为停止
// Set motor speed, speed:± (1000-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.

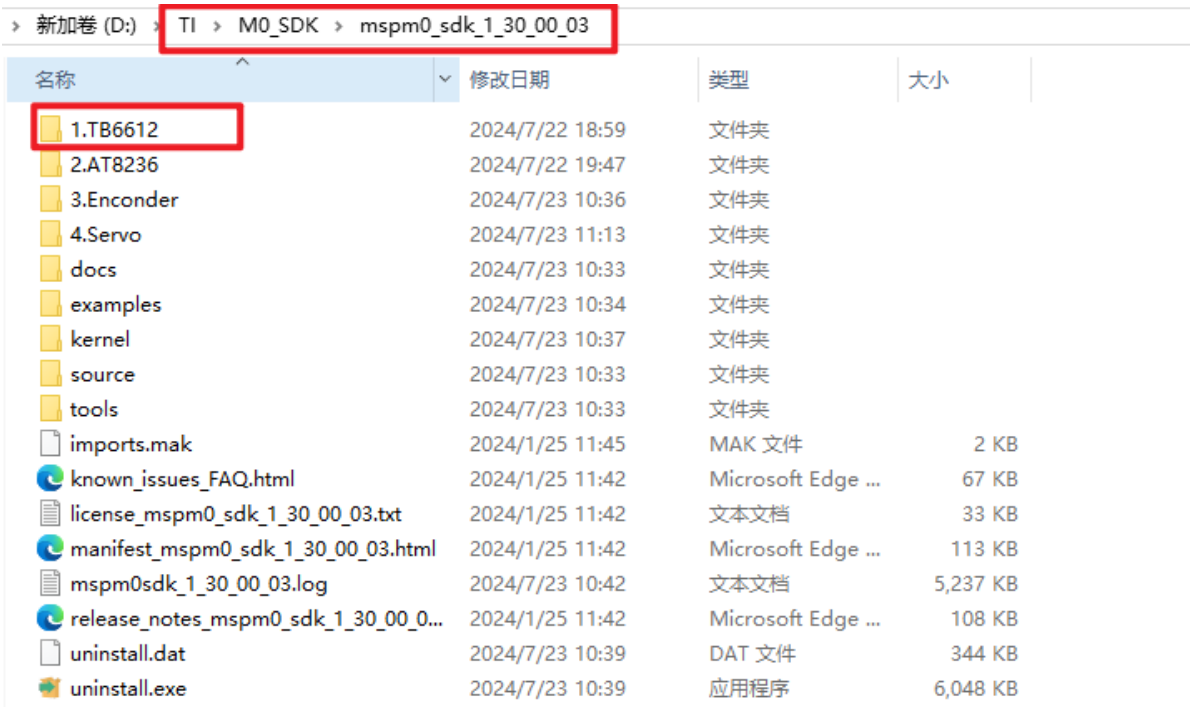
- app_motor.c

```
// 控制小车运动，Motor_X=[-1000, 1000]，超过范围则无效。
// Control the movement of the car, Motor_X=[-1000, 1000]. 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, put the car on the ground, and when it recognizes an obstacle 30cm ahead, the car will first retreat, then turn right to avoid the obstacle in front.