12. Control the forward and reverse rotation of the motor

12. Control the forward and reverse rotation of the motor
    12.1. Experimental purpose
    12.2. Configuration pin information
    12.3. Analysis of the experimental flow chart
    12.4. core code explanation
    12.5. hardware connection
    12.6. Experimental effect

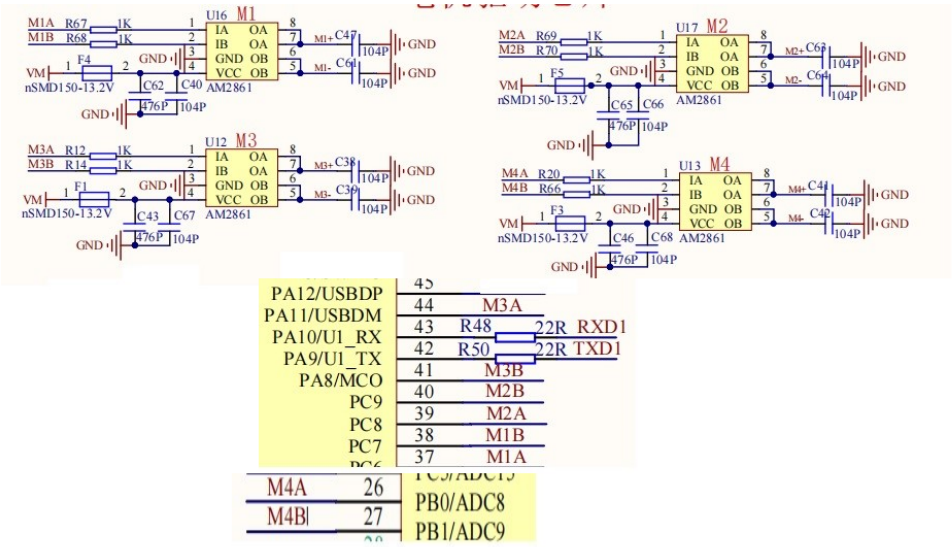
12.1. Experimental purpose

Using the timer function of STM32, the motor driver chip AM2861 is driven to control the forward rotation, reverse rotation and stop of the motor.

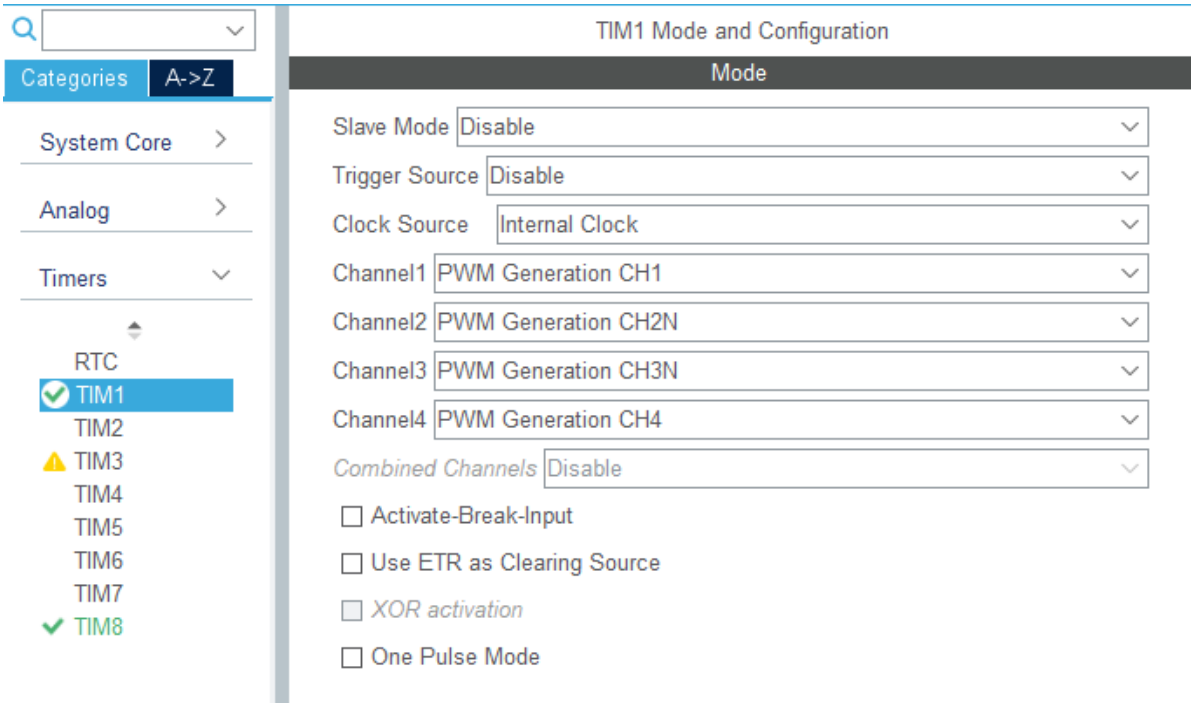
12.2. Configuration pin information

1. Import the ioc file from the Beep project and name it Motor.

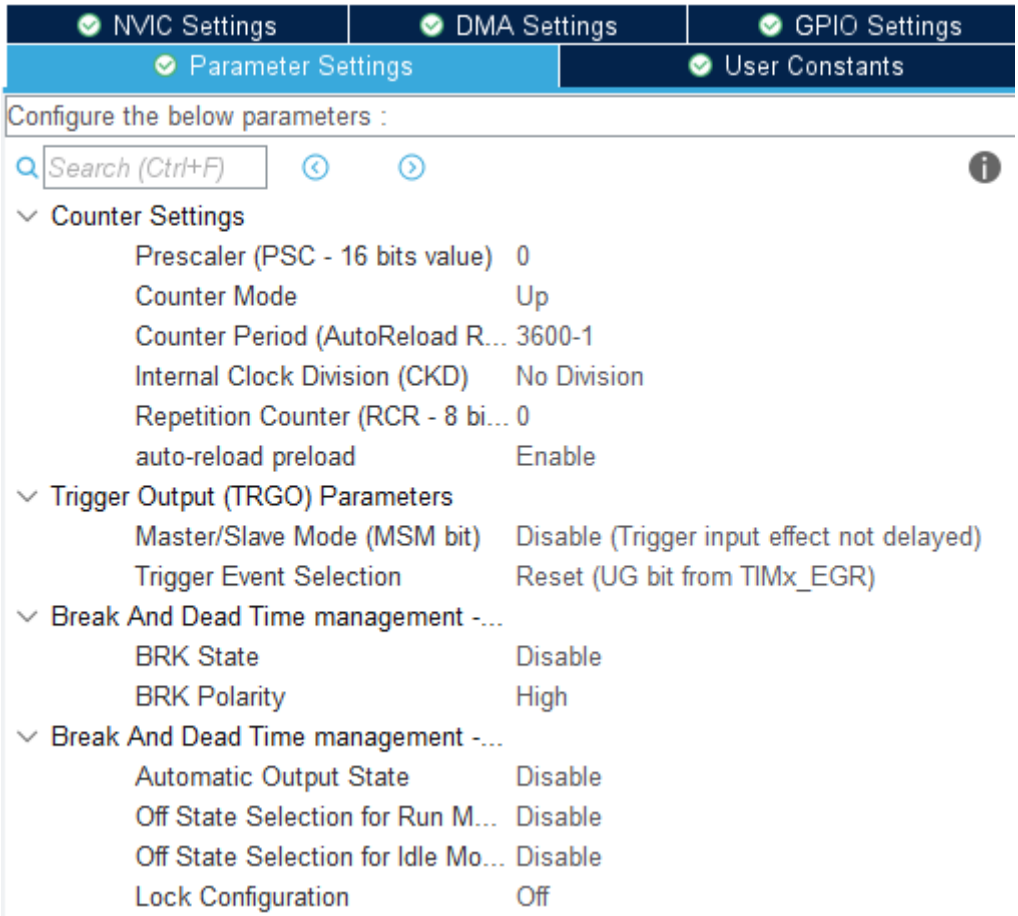
According to the schematic diagram, there are a total of four AM2861 motor driver modules, one motor driver module controls one motor, and the pin configuration is shown in the figure below.



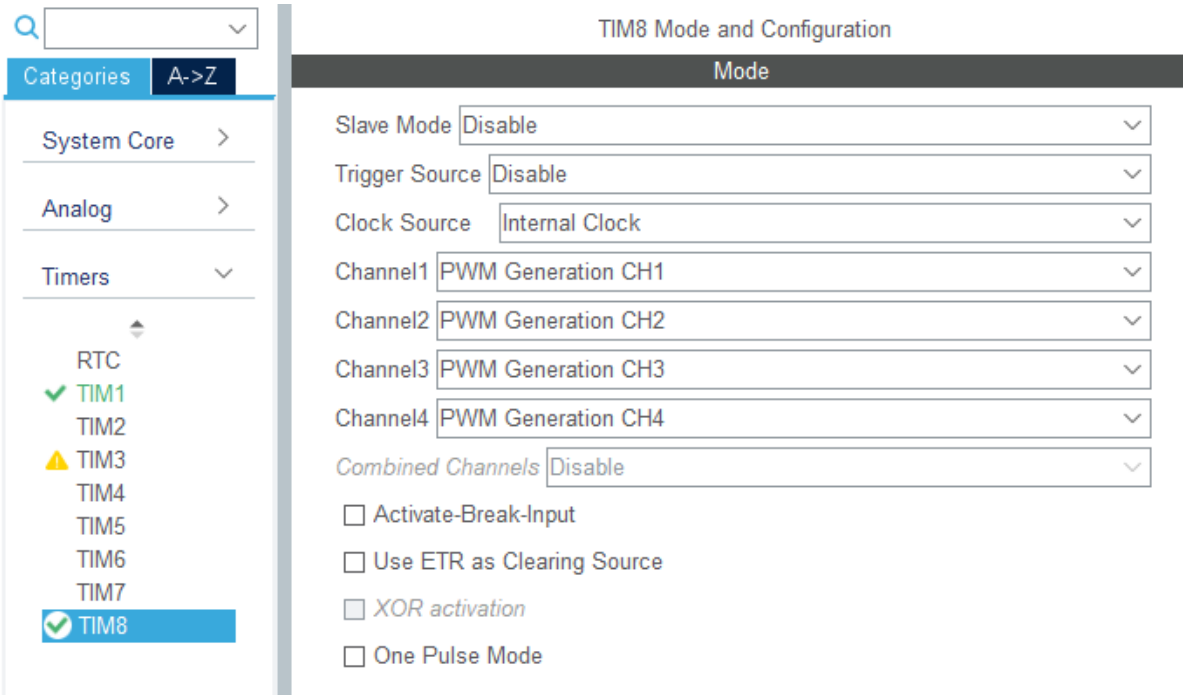
2. First set timer 1, select the internal clock as the clock source, and set the corresponding pins of the four channels to output PWM signals CH1 CH2N CH3N CH4 PA8 PB0 PB1 PA11.



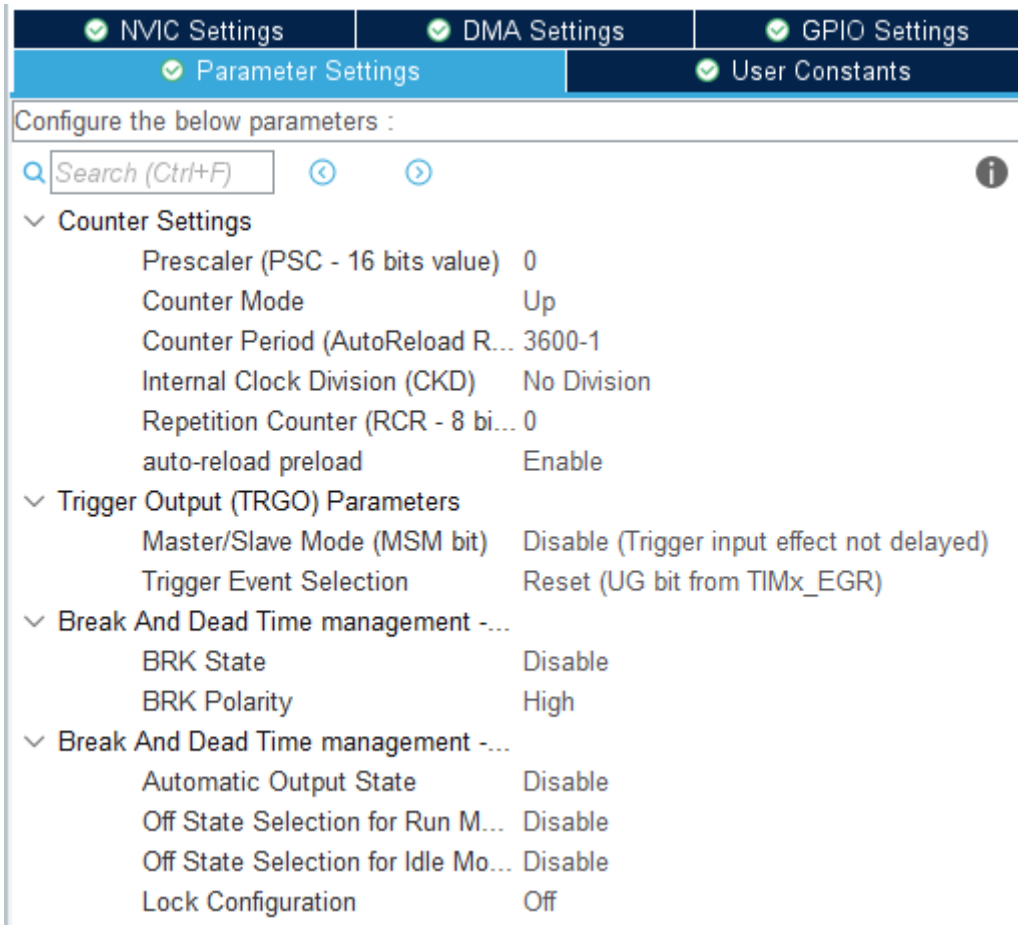
Other parameters are shown in the figure below:



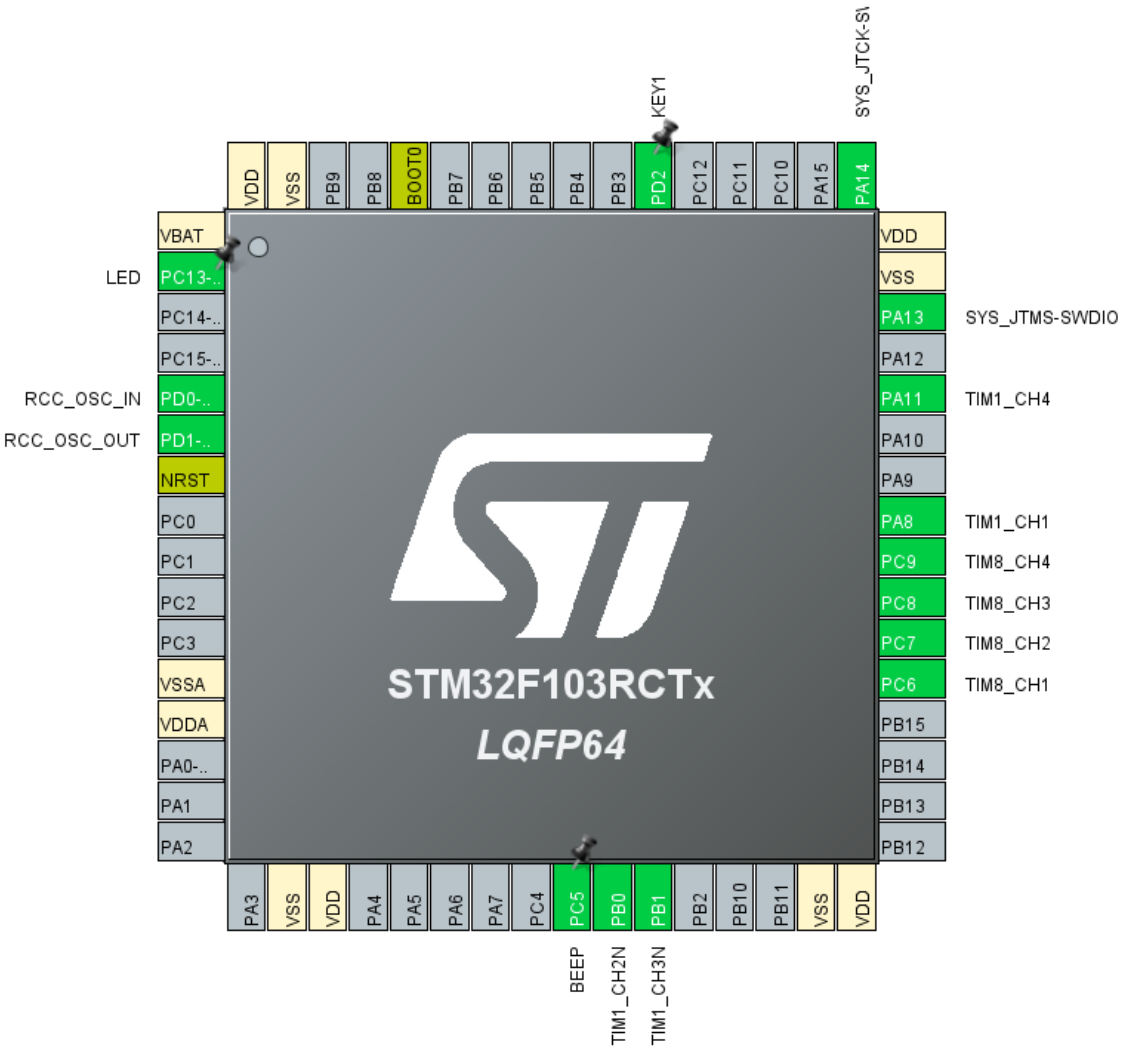
3. Next, set timer 8, select the internal clock as the clock source, and set the corresponding pins of the four channels to output PWM signals CH1 CH2 CH3 CH4 PC6 PC7 PC8 PC9.



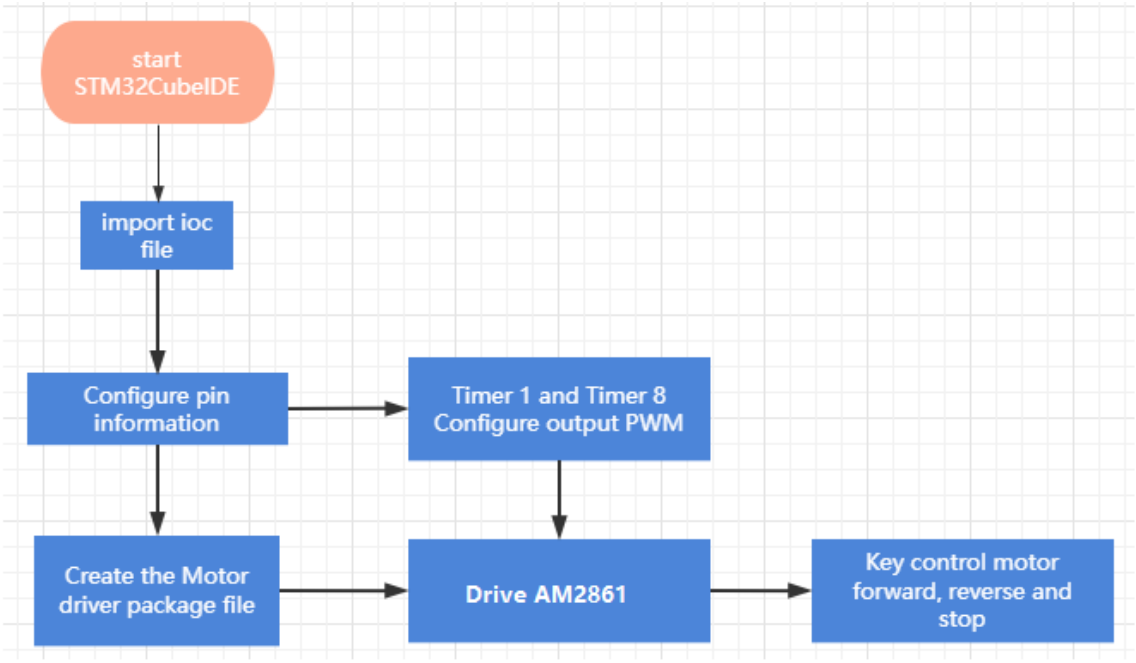
Other parameters are the same as timer 1.



The final chip configuration pins are shown in the figure below:



12.3. Analysis of the experimental flow chart



12.4. core code explanation

1. Create new bsp_motor.h and bsp_motor.c, and add the following content to bsp_motor.h:

```
#define PWM_M1_A  TIM8->CCR1
#define PWM_M1_B  TIM8->CCR2

#define PWM_M2_A  TIM8->CCR3
#define PWM_M2_B  TIM8->CCR4

#define PWM_M3_A  TIM1->CCR4
#define PWM_M3_B  TIM1->CCR1

#define PWM_M4_A  TIM1->CCR2
#define PWM_M4_B  TIM1->CCR3


typedef enum {
    MOTOR_ID_M1 = 0,
    MOTOR_ID_M2,
    MOTOR_ID_M3,
    MOTOR_ID_M4,
    MAX_MOTOR
} Motor_ID;


void Motor_Init(void);
void Motor_Set_Pwm(uint8_t id, int16_t speed);
void Motor_Stop(uint8_t brake);
```

Among them, M1 corresponds to the motor in the upper left corner of the body, M2 corresponds to the motor in the lower left corner, M3 corresponds to the motor in the upper right corner, and M4 corresponds to the motor in the lower right corner.

2. Create the following content in the bsp_motor.c file:

The motor timer PWM output starts initialization.

```
// The PWM port of the motor is initialized  电机PWM口初始化
void Motor_Init(void)
{
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
    HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_3);
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);

    HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_3);
    HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_4);
}
```

3. Motor stop function, parameter brake=1 means brake stop, brake=0 means free stop.

```
// All motors stopped  所有电机停止
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;
}
```

4. Since the motor has a certain control dead zone, the dead zone can be filtered. If you choose not to filter, please define the MOTOR_IGNORE_PULSE parameter to 0.

```
// Ignore PWM dead band  忽略PWM信号死区
static int16_t Motor_Ignore_Dead_Zone(int16_t pulse)
{
    if (pulse > 0) return pulse + MOTOR_IGNORE_PULSE;
    if (pulse < 0) return pulse - MOTOR_IGNORE_PULSE;
    return 0;
}
```

5. The next step is to set the motor speed, where id is the motor ID, speed speed value range: ±(3600-MOTOR_IGNORE_PULSE), 0 is stop.

```
// 设置电机速度，speed:±（3600-MOTOR_IGNORE_PULSE），0为停止
// Set motor speed, speed:± (3600-MOTOR_IGNORE_PULSE), 0 indicates stop
void Motor_Set_Pwm(uint8_t id, int16_t speed)
{
    int16_t pulse = Motor_Ignore_Dead_Zone(speed);
    // Limit input  限制输入
    if (pulse >= MOTOR_MAX_PULSE)
        pulse = MOTOR_MAX_PULSE;
    if (pulse <= -MOTOR_MAX_PULSE)
        pulse = -MOTOR_MAX_PULSE;

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

6. Add the content of motor initialization in the Bsp_Init() function.

```
// The peripheral device is initialized  外设设备初始化
void Bsp_Init(void)
{
    Beep_On_Time(50);
    Motor_Init();
}
```

7. Add the function of button to control the motor in the Bsp_Loop() function, press the first time to go forward, the second time to free stop, the third time to retreat, and the fourth time to brake to stop.

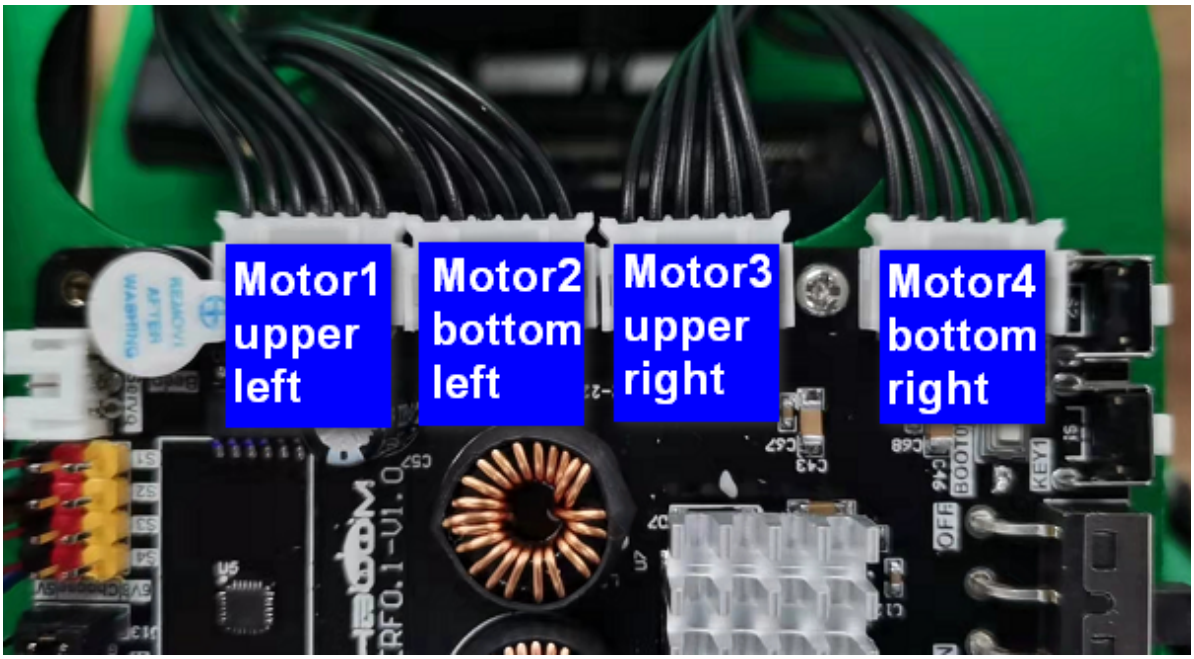
```
// main.c中循环调用此函数，避免多次修改main.c文件。
// This function is called in a loop in main.c to avoid
void Bsp_Loop(void)
{
    // Detect button down events    检测按键按下事件
    if (Keyl_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        static int state = 0;
        state++;
        int speed = 0;
        if (state == 1)
        {
            speed = 2000;
            Motor_Set_Pwm(MOTOR_ID_M1, speed);
            Motor_Set_Pwm(MOTOR_ID_M2, speed);
            Motor_Set_Pwm(MOTOR_ID_M3, speed);
            Motor_Set_Pwm(MOTOR_ID_M4, speed);
        }
        if (state == 2)
        {
            Motor_Stop(0);
        }
        if (state == 3)
        {
            speed = -2000;
            Motor_Set_Pwm(MOTOR_ID_M1, speed);
            Motor_Set_Pwm(MOTOR_ID_M2, speed);
            Motor_Set_Pwm(MOTOR_ID_M3, speed);
            Motor_Set_Pwm(MOTOR_ID_M4, speed);
        }
        if (state == 4)
        {
            state = 0;
            Motor_Stop(1);
        }

    }

    Bsp_Led_Show_State_Handle();
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}
```

12.5. hardware connection

The motor connecting line needs to be connected to the corresponding motor as shown in the figure below, otherwise it may cause the problem that the program does not match the phenomenon. Motor 1 corresponds to the motor in the upper left corner of the body, Motor 2 corresponds to the motor in the lower left corner, Motor 3 corresponds to the motor in the upper right corner, and Motor 4 corresponds to the motor in the lower right corner.



Since the power of the motor is relatively large, the expansion board should not be powered by USB 5V directly, but must be powered by DC 12V.

12.6. Experimental effect

Since the motor will turn when started, please stand up the trolley before the experiment, and the motor wheels are suspended in the air to avoid rampage.

After the program is programmed, the LED light flashes every 200 milliseconds. Press the first forward, the second free stop, the third backward, and the fourth brake to stop.