12. Control motor forward reversal

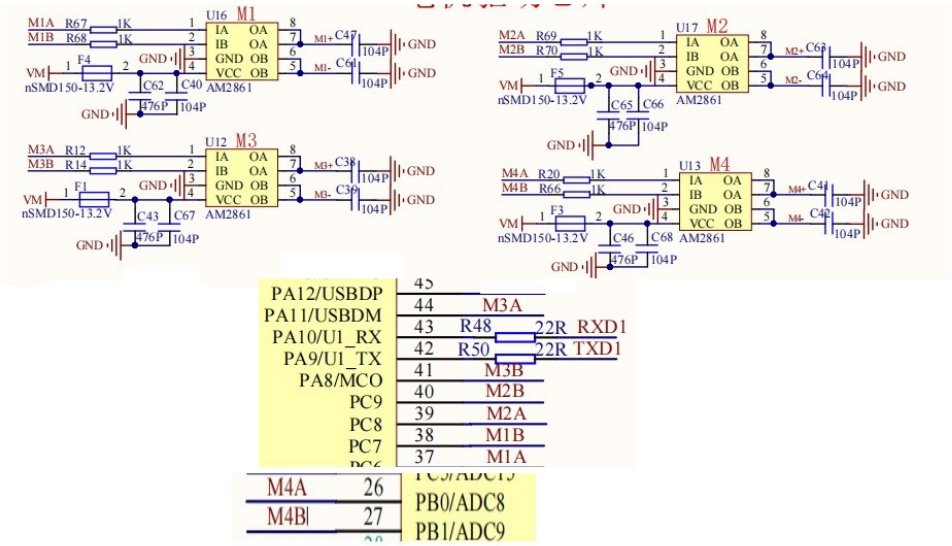
12.1. Experimental purpose

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

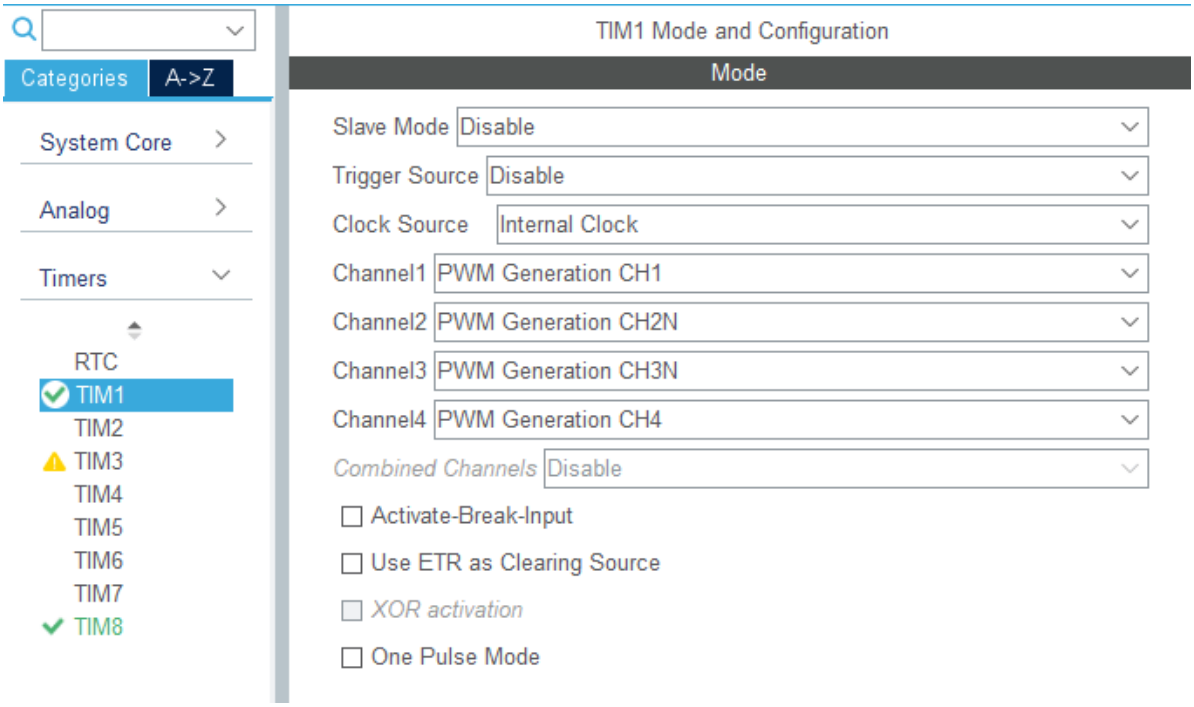
12.2. Configure pin information

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

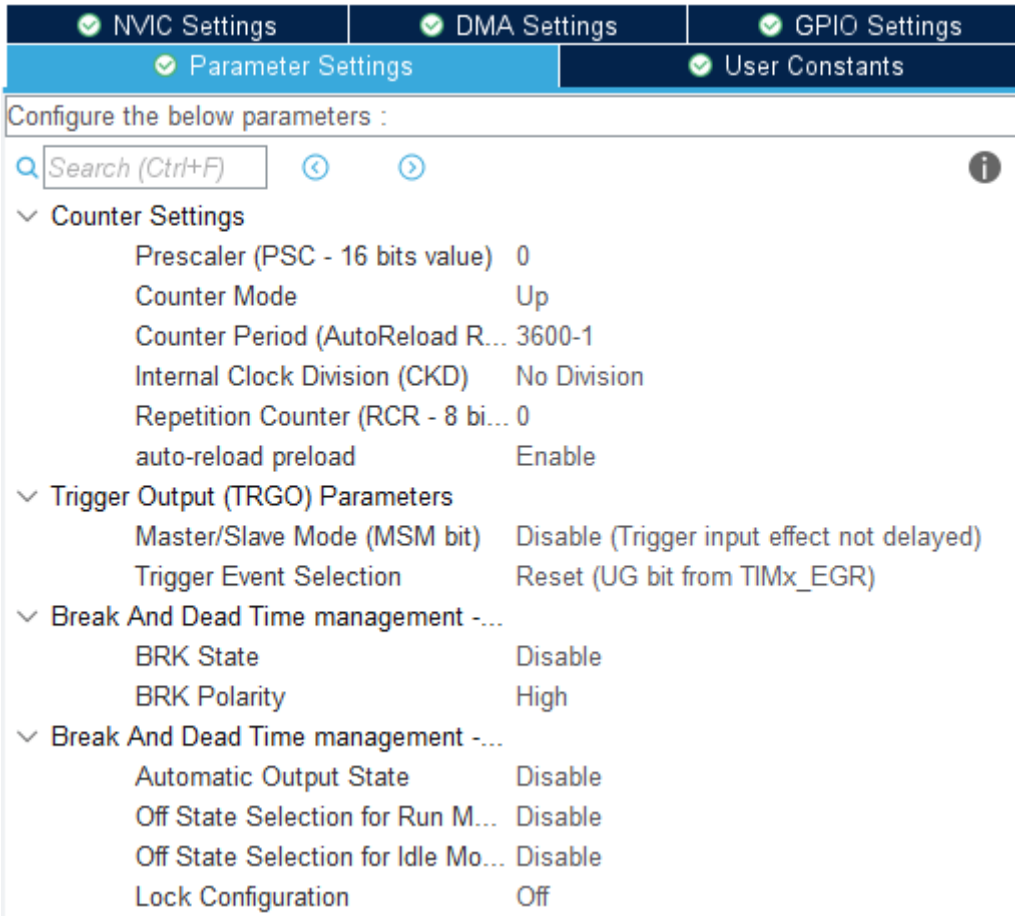
According to the schematic diagram, there are four AM2861 motor drive modules in total, one motor drive module controls one motor, and the pin configuration is shown in the figure below.



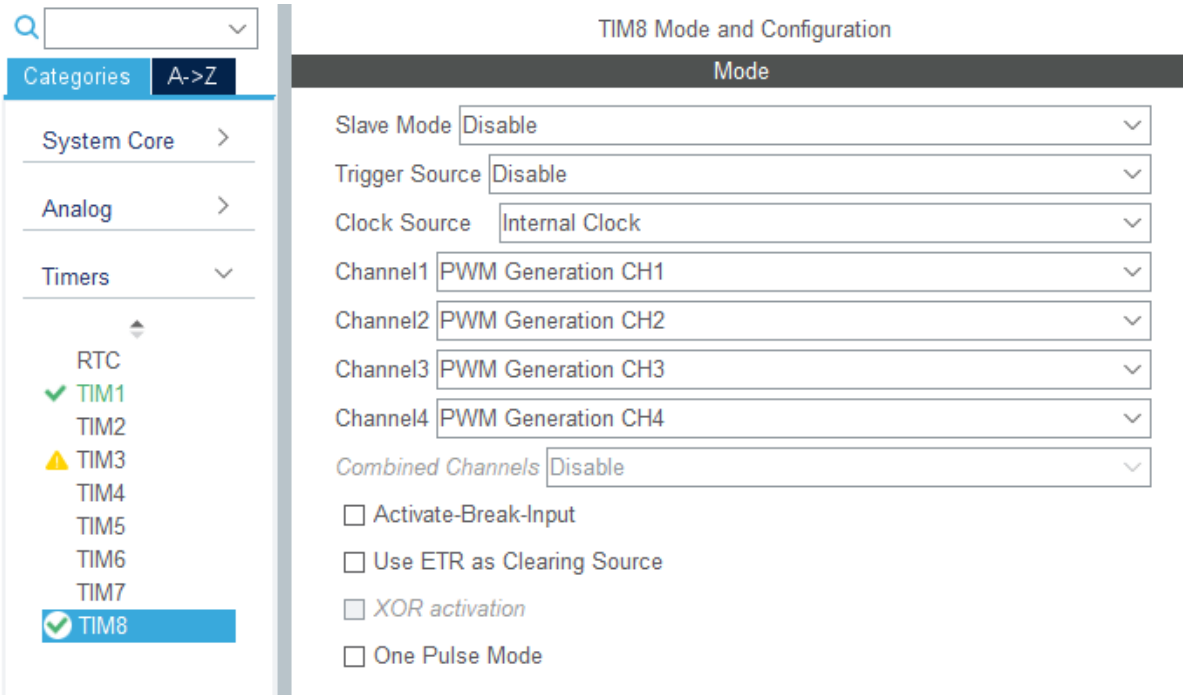
2. Set timer 1, clock source internal clock, and set four channels to output PWM signal CH1 CH2N CH3N CH4 corresponding to pin PA8 PB0 PB1 PA11.



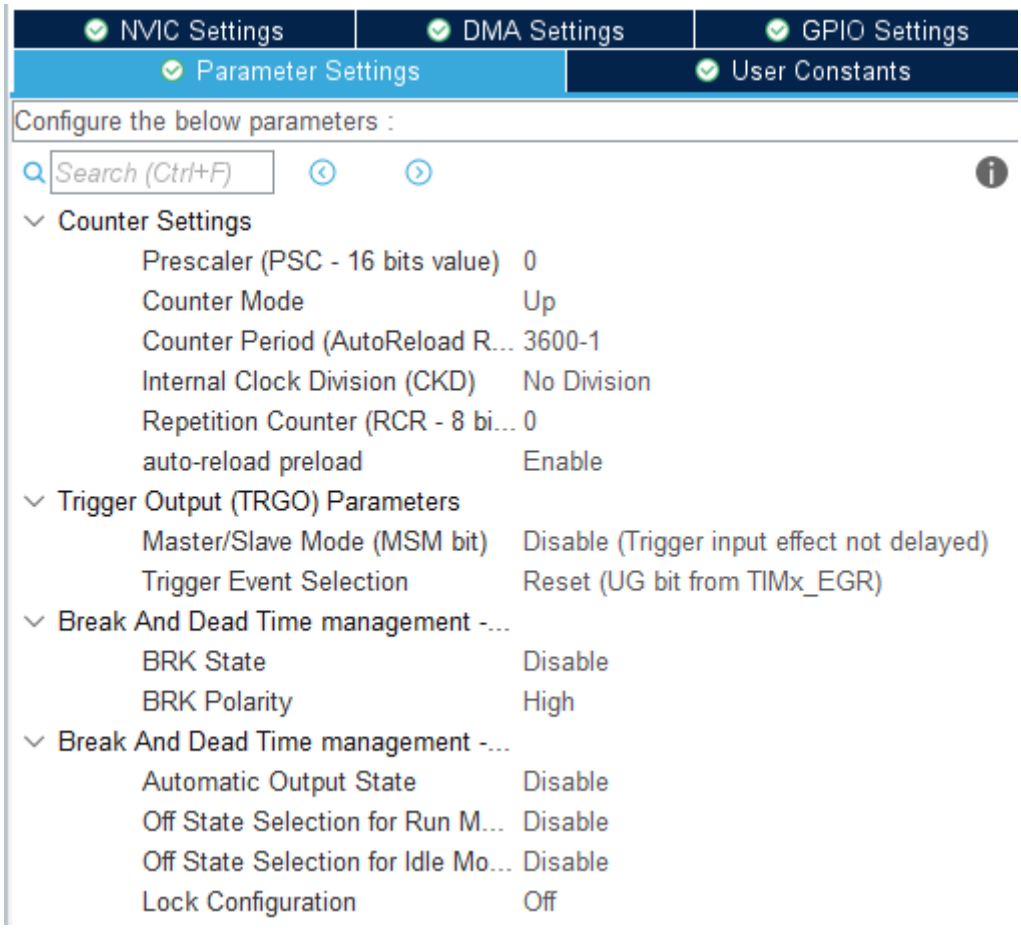
Other parameters are shown in the figure below:



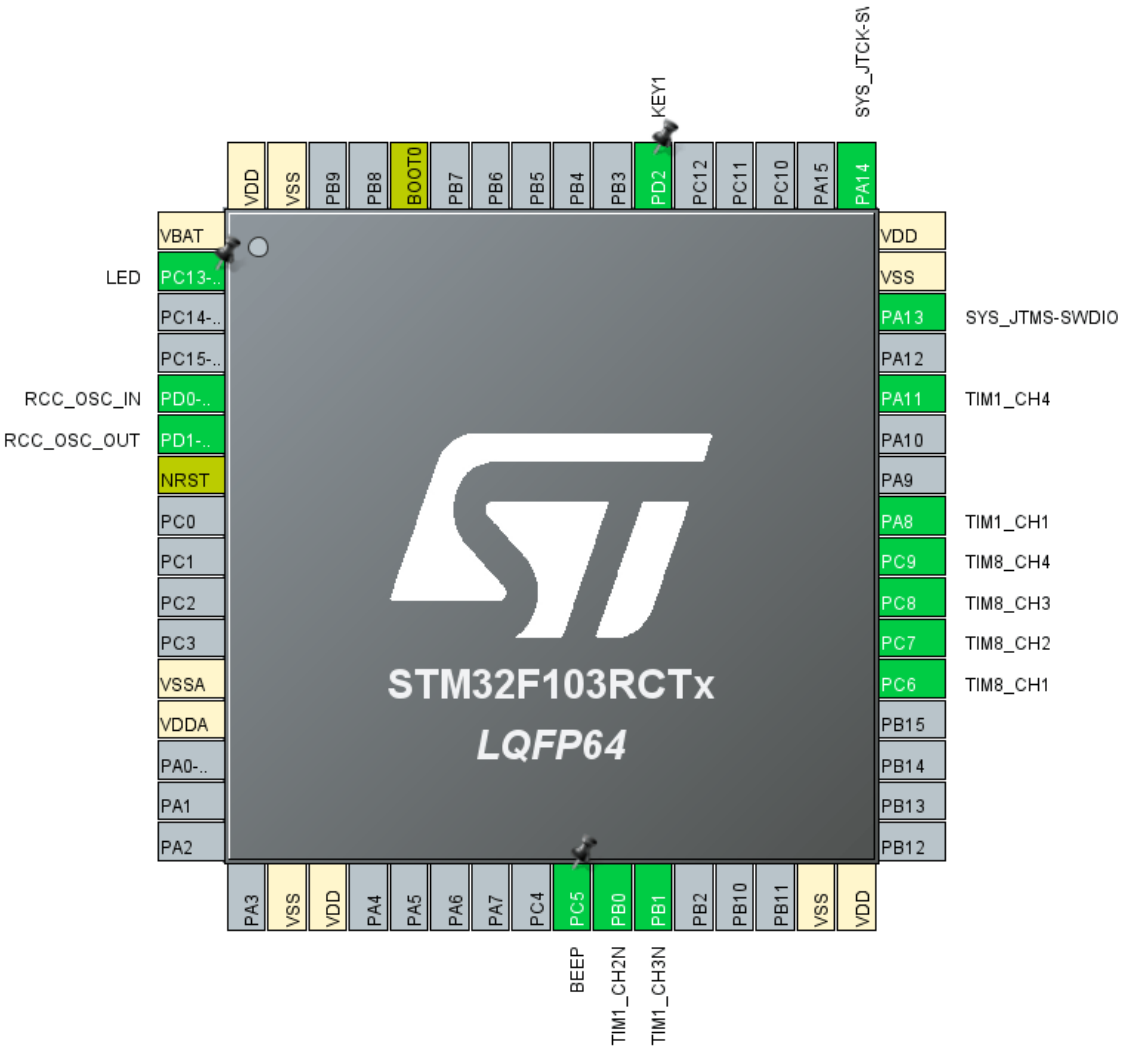
3. Then set timer 8, select the internal clock as the clock source, and set four channels to output PWM signals. CH1 CH2 CH3 CH4 corresponds to pins PC6 PC7 PC8 PC9.



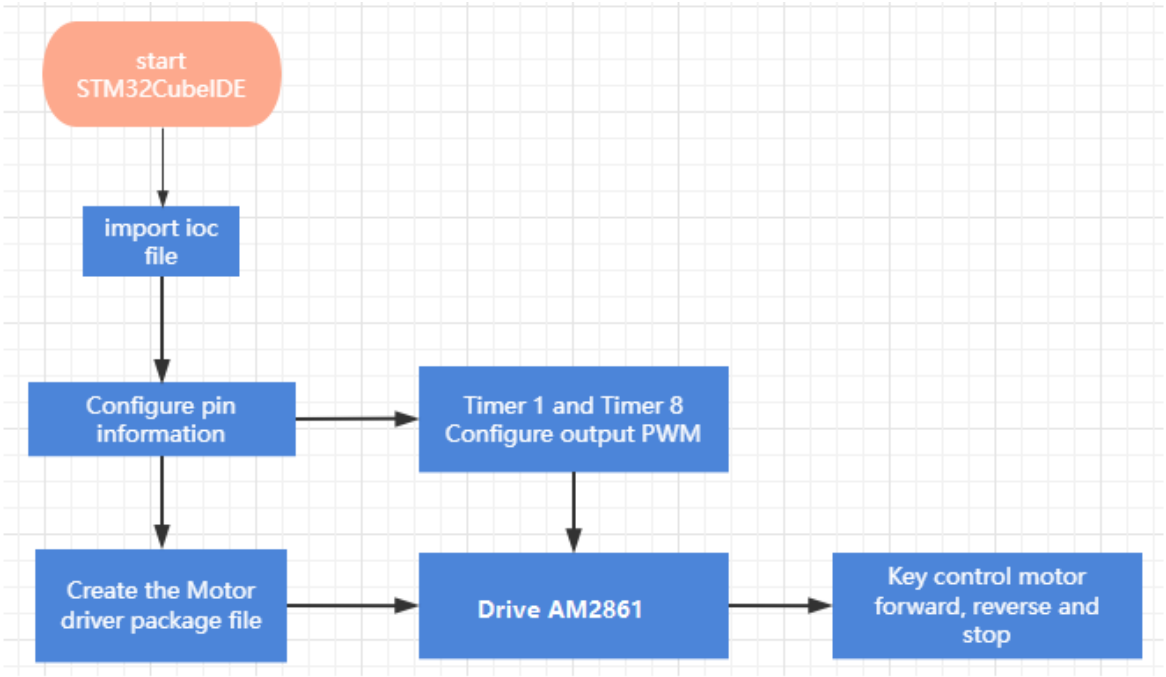
Other parameters are the same as those of timer 1.



The final chip configuration pin is shown in the figure below:



12.3. Analysis of experimental flow chart



12.4. Core code interpretation

1. Create 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);
```

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:

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, brake=1 for brake stop, brake=0 for 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. Because the motor has a certain control dead zone, the dead zone can be filtered. If you choose not to filter, set 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. Next 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 motor initialization to 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 a button to control the motor function in the Bsp_Loop() function, press the first forward, the second free stop, the third backward, and the fourth brake stop.

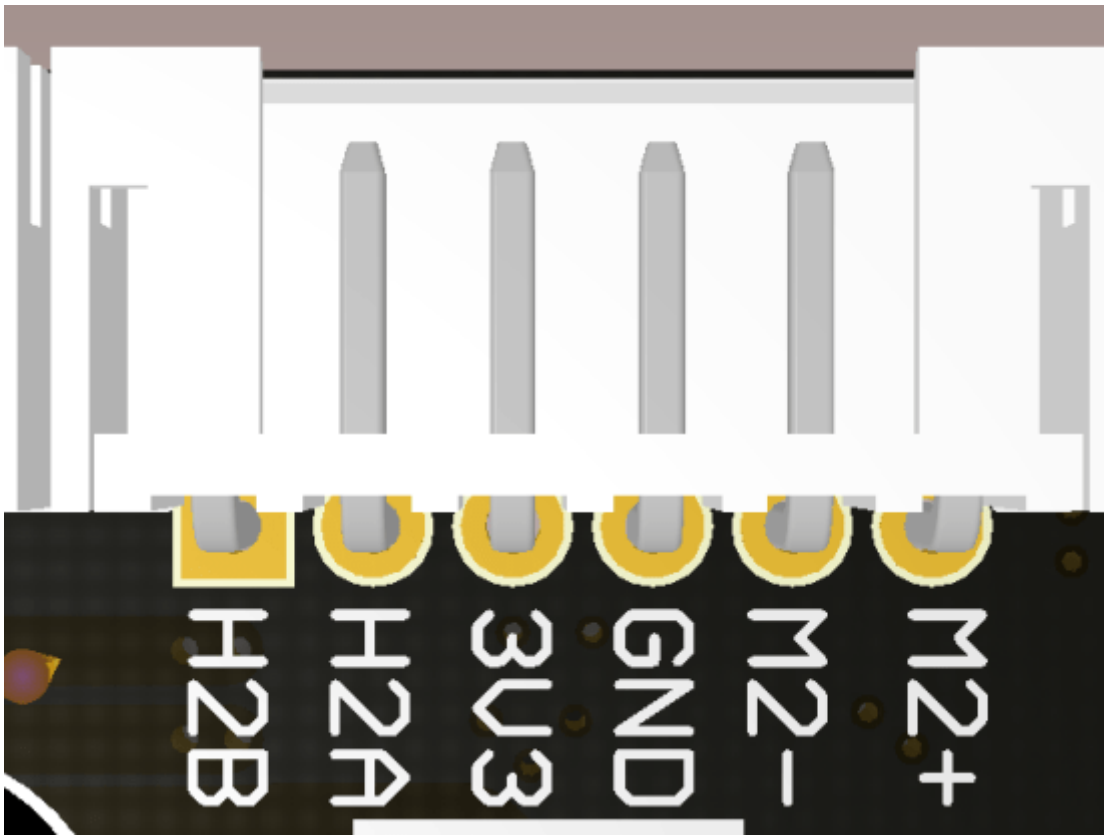
```
// main.c中循环调用此函数，避免多次修改main.c文件。
// This function is called in a loop in main.c to avoid
void Bsp_Loop(void)
{
    // Detect button down events    检测按键按下事件
    if (Keyl_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        static int state = 0;
        state++;
        int speed = 0;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

Because the power of the motor is relatively large, do not directly use USB 5V power supply for the expansion board, and must use battery power supply.

Motor interface sequence diagram:



12.6. Experimental effect

Since the motor will turn when started, please set up the car before the experiment, and the motor wheels are suspended to avoid rampaging.

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