

## 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

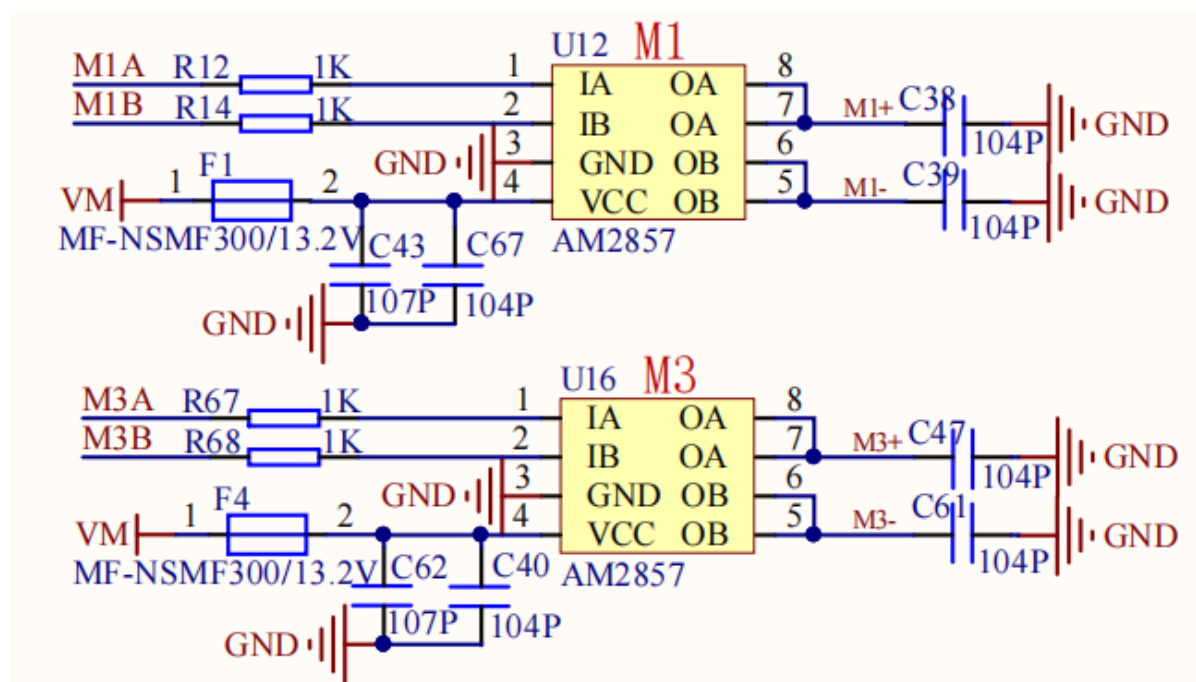
### 12.1. Experimental purpose

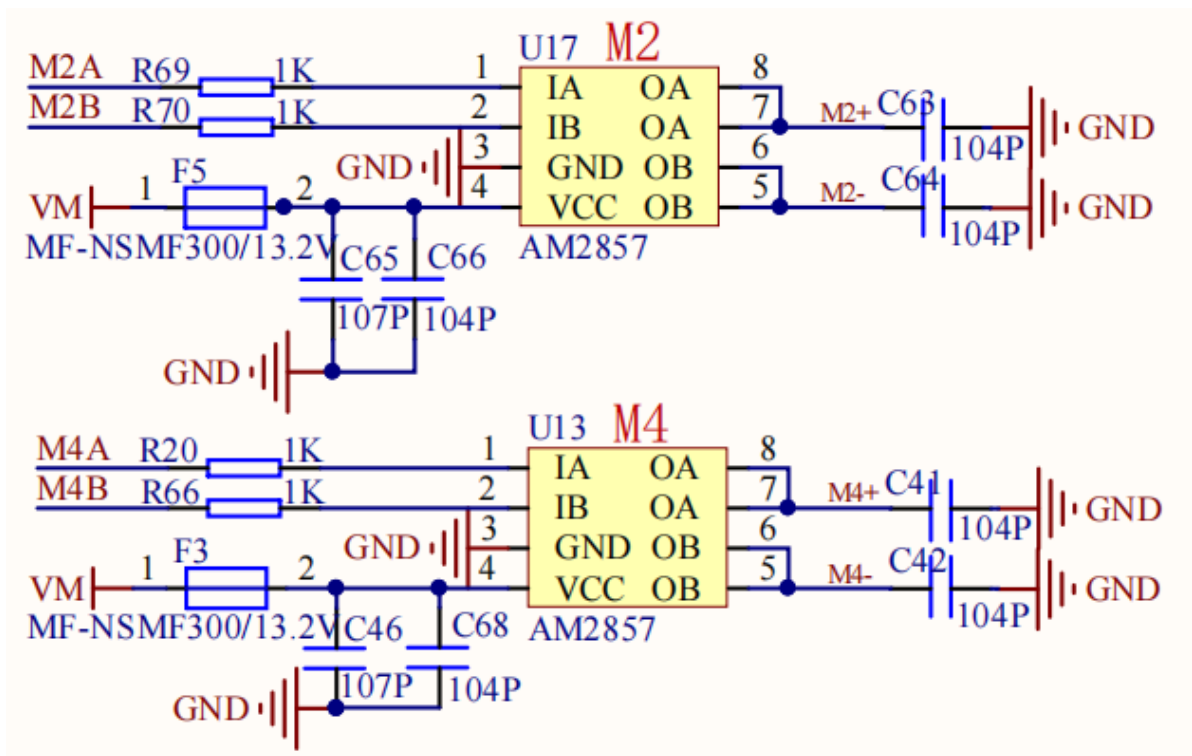
Using the timer function of STM32, the motor driver chip AM2857 is driven to control the forward rotation, reverse rotation and stop of the motor.

### 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 AM2857 motor driver modules, one motor driver module controls one motor, and the pin configuration is shown in the figure below.





PA12/USBDP	45
PA11/USBDM	44
PA10/U1_RX	43
PA9/U1_TX	42
PA8/MCO	41
PC9	40
PC8	39
PC7	38
PC6	37

M4A	26
M4B	27

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.

Categories

A-Z

System Core >

Analog >

Timers >

RTC

✓ TIM1

TIM2

⚠ TIM3

TIM4

TIM5

TIM6

TIM7

✓ TIM8

TIM1 Mode and Configuration

Mode

Slave Mode

Disable

Trigger Source

Disable

Clock Source

Internal Clock

Channel1

PWM Generation CH1

Channel2

PWM Generation CH2N

Channel3

PWM Generation CH3N

Channel4

PWM Generation CH4

Combined Channels

Disable

☐ Activate-Break-Input

☐ Use ETR as Clearing Source

☐ XOR activation

☐ One Pulse Mode

Other parameters are shown in the figure below:

✓ NVIC Settings	✓ DMA Settings	✓ GPIO Settings
✓ Parameter Settings	✓ User Constants	
Configure the below parameters :		
<input type="text" value="Search (Ctrl+F)"/> <input type="button" value="←"/> <input type="button" value="→"/> <input type="button" value="i"/>		
<b>Counter Settings</b>		
Prescaler (PSC - 16 bits value) 0		
Counter Mode Up		
Counter Period (AutoReload R... 3600-1		
Internal Clock Division (CKD) No Division		
Repetition Counter (RCR - 8 bi... 0		
auto-reload preload Enable		
<b>Trigger Output (TRGO) Parameters</b>		
Master/Slave Mode (MSM bit) Disable (Trigger input effect not delayed)		
Trigger Event Selection Reset (UG bit from TIMx_EGR)		
<b>Break And Dead Time management -...</b>		
BRK State Disable		
BRK Polarity High		
<b>Break And Dead Time management -...</b>		
Automatic Output State Disable		
Off State Selection for Run M... Disable		
Off State Selection for Idle Mo... Disable		
Lock Configuration Off		

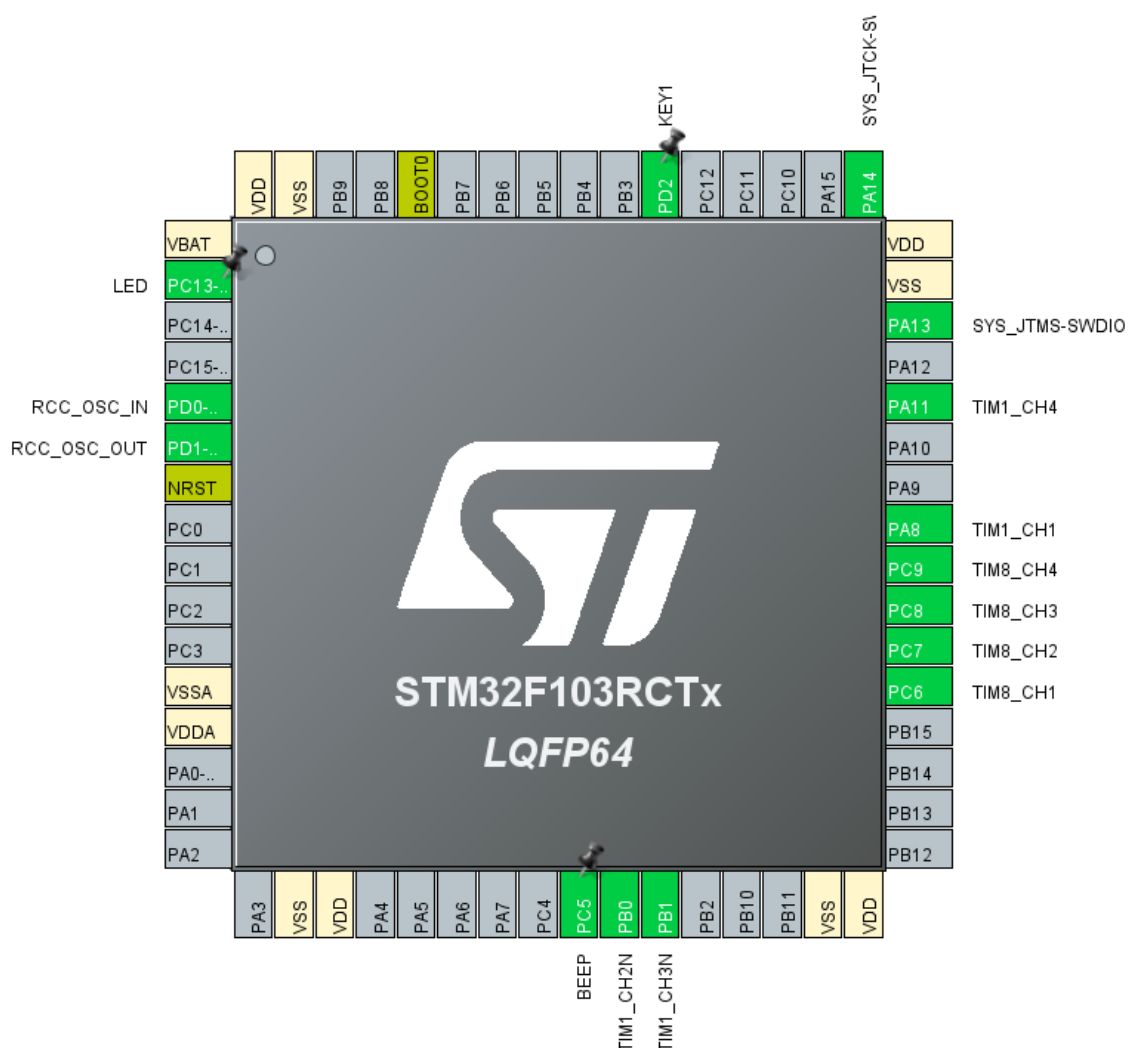
3. Next, set timer 8, select the internal clock as the clock source, and set the corresponding pins of the four channels to output PWM signals CH1 CH2 CH3 CH4 PC6 PC7 PC8 PC9.

TIM8 Mode and Configuration	
Categories	Mode
System Core >	Slave Mode Disable
Analog >	Trigger Source Disable
Timers >	Clock Source Internal Clock
<ul style="list-style-type: none"> <li>RTC</li> <li>✓ TIM1</li> <li>TIM2</li> <li>⚠ TIM3</li> <li>TIM4</li> <li>TIM5</li> <li>TIM6</li> <li>TIM7</li> <li>✓ TIM8</li> </ul>	Channel1 PWM Generation CH1
	Channel2 PWM Generation CH2
	Channel3 PWM Generation CH3
	Channel4 PWM Generation CH4
	Combined Channels Disable
	<input type="checkbox"/> Activate-Break-Input
	<input type="checkbox"/> Use ETR as Clearing Source
	<input type="checkbox"/> XOR activation
	<input type="checkbox"/> One Pulse Mode

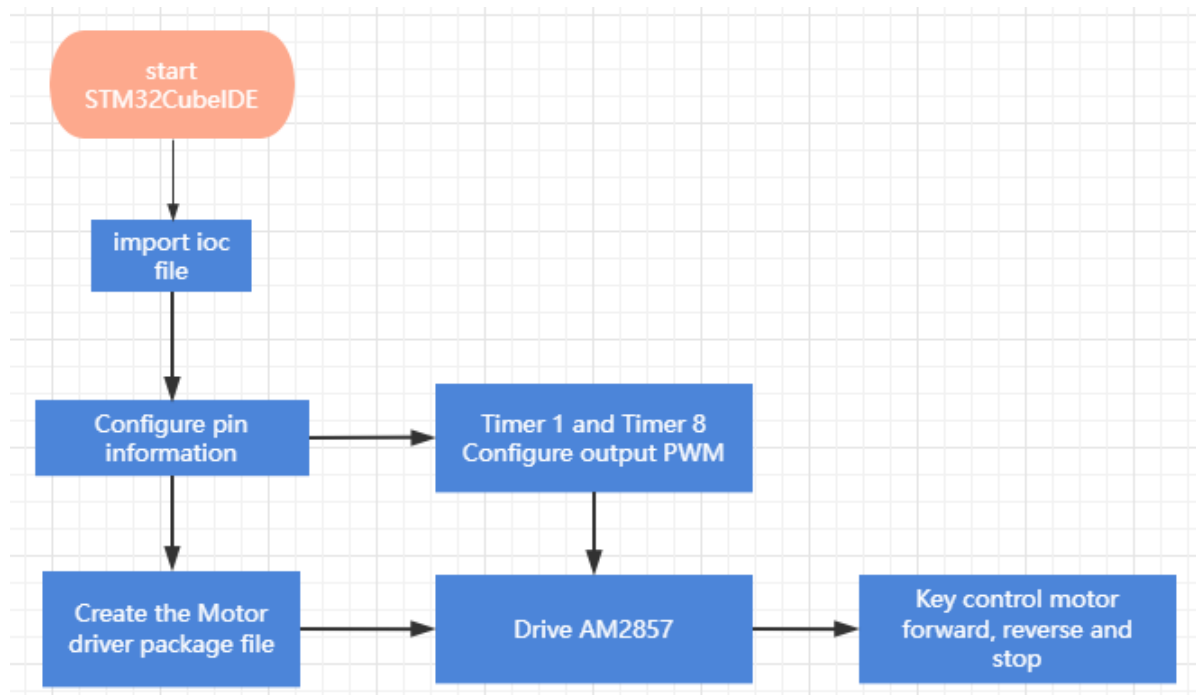
Other parameters are the same as timer 1.

✓ NMC Settings	✓ DMA Settings	✓ GPIO Settings
✓ Parameter Settings	✓ User Constants	
Configure the below parameters :		
<input type="text" value="Search (Ctrl+F)"/> <input type="button" value="◀"/> <input type="button" value="▶"/> <input type="button" value="i"/>		
<div> <div>Counter Settings</div> <div> <div>Prescaler (PSC - 16 bits value)</div> <div>0</div> </div> <div> <div>Counter Mode</div> <div>Up</div> </div> <div> <div>Counter Period (AutoReload R...</div> <div>3600-1</div> </div> <div> <div>Internal Clock Division (CKD)</div> <div>No Division</div> </div> <div> <div>Repetition Counter (RCR - 8 bi...</div> <div>0</div> </div> <div> <div>auto-reload preload</div> <div>Enable</div> </div> </div>		
<div> <div>Trigger Output (TRGO) Parameters</div> <div> <div>Master/Slave Mode (MSM bit)</div> <div>Disable (Trigger input effect not delayed)</div> </div> <div> <div>Trigger Event Selection</div> <div>Reset (UG bit from TIMx_EGR)</div> </div> </div>		
<div> <div>Break And Dead Time management -...</div> <div> <div>BRK State</div> <div>Disable</div> </div> <div> <div>BRK Polarity</div> <div>High</div> </div> </div>		
<div> <div>Break And Dead Time management -...</div> <div> <div>Automatic Output State</div> <div>Disable</div> </div> <div> <div>Off State Selection for Run M...</div> <div>Disable</div> </div> <div> <div>Off State Selection for Idle Mo...</div> <div>Disable</div> </div> <div> <div>Lock Configuration</div> <div>Off</div> </div> </div>		

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:  $\pm(3600 - \text{MOTOR\_IGNORE\_PULSE})$ , 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 (Key1_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);
        }
    }

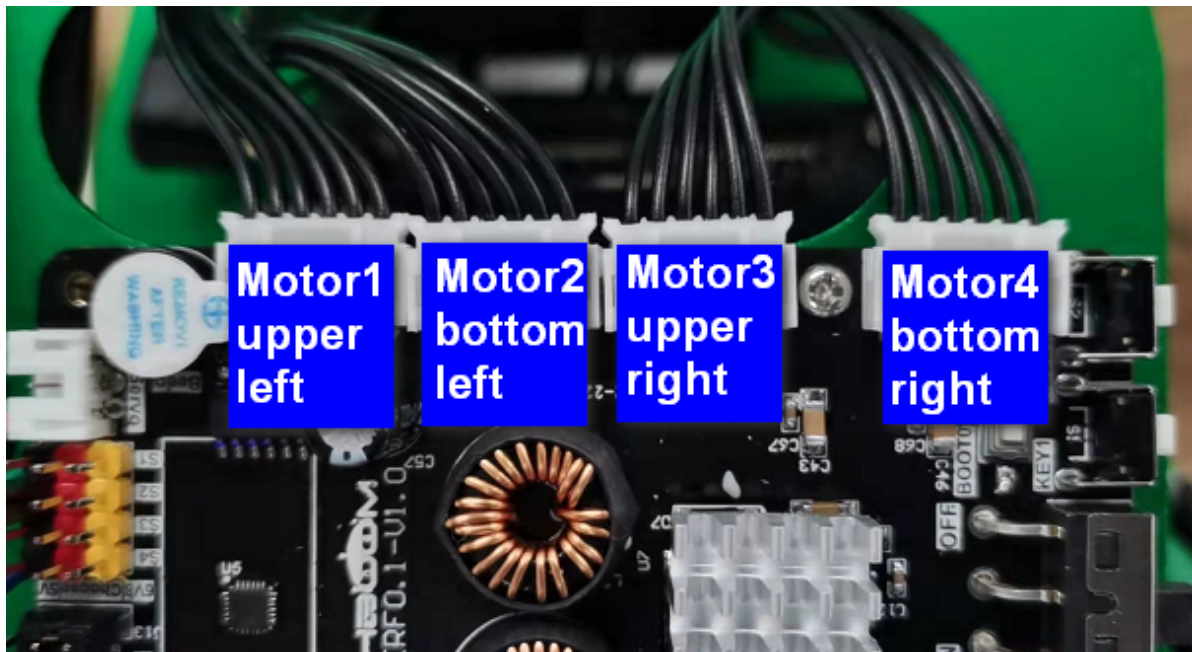
    Bsp_Led_Show_State_Handle();
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}

```

## 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.