12. Control motor forward and reverse

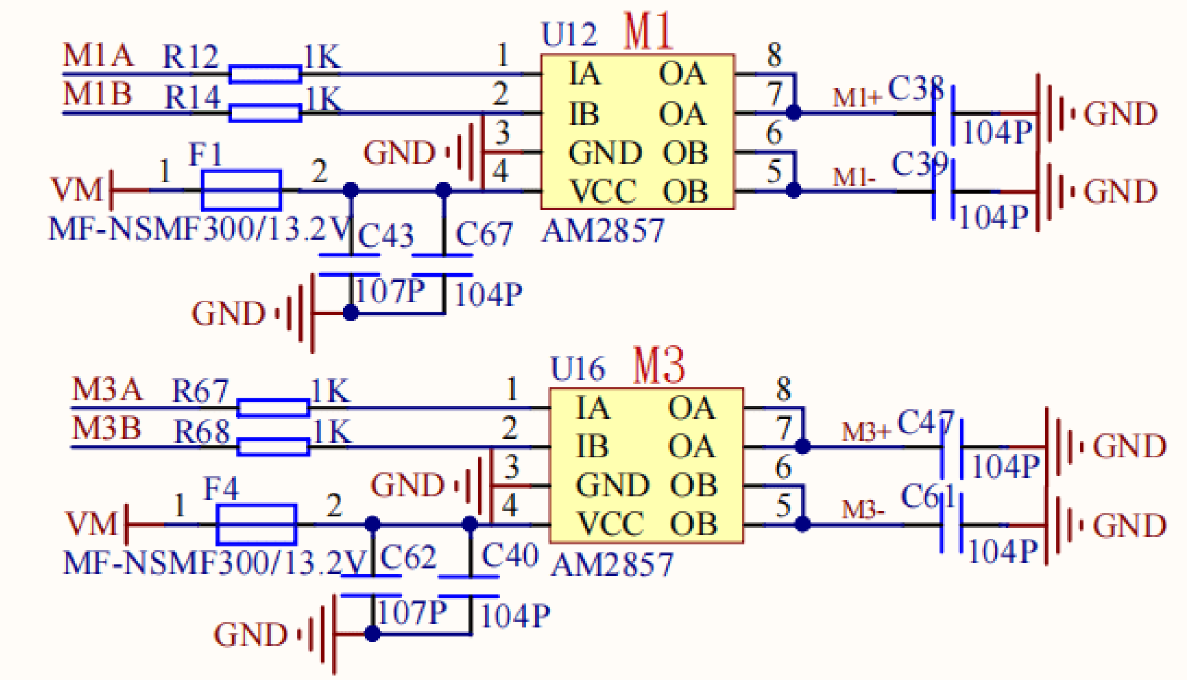
12.1. Purpose of the experiment

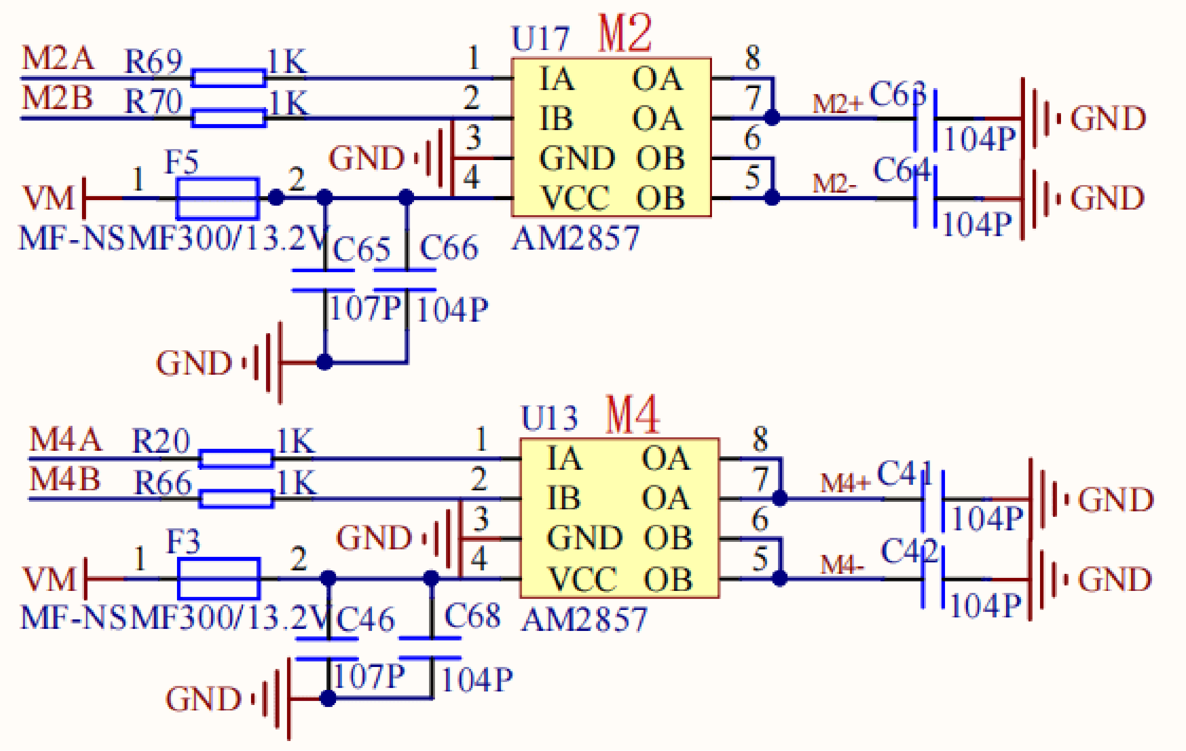
Use the timer function of STM32 to drive the motor driver chip AM2857, so as to control the motor forward, reverse and stop.

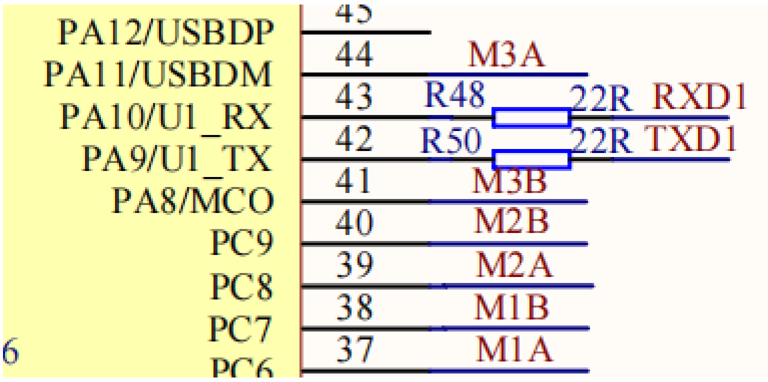
12.2 Configuring Pin Information

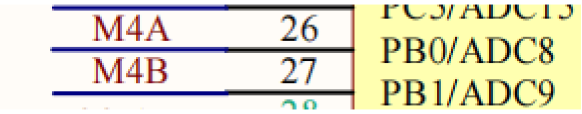
1. Import the ioc file from Beep's project and name it Motor.

According to the schematic diagram, we can see that there are four AM2857 motor driver modules in total, one motor driver module controls one motor, and the pin configuration is shown below.

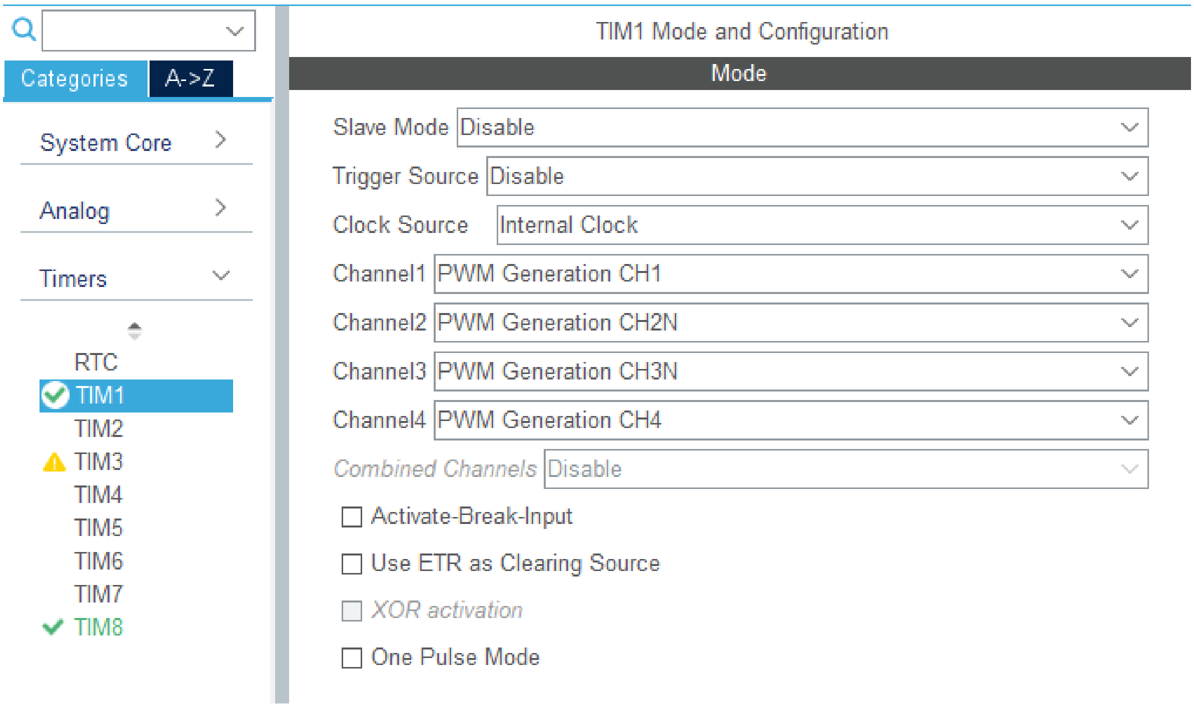








2. First set timer 1, clock source select internal clock, set the four channel output PWM signal CH1 CH2N CH3N CH4 corresponding pin PA8 PB0 PB1 PA11.



TIM1 Mode and Configuration

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

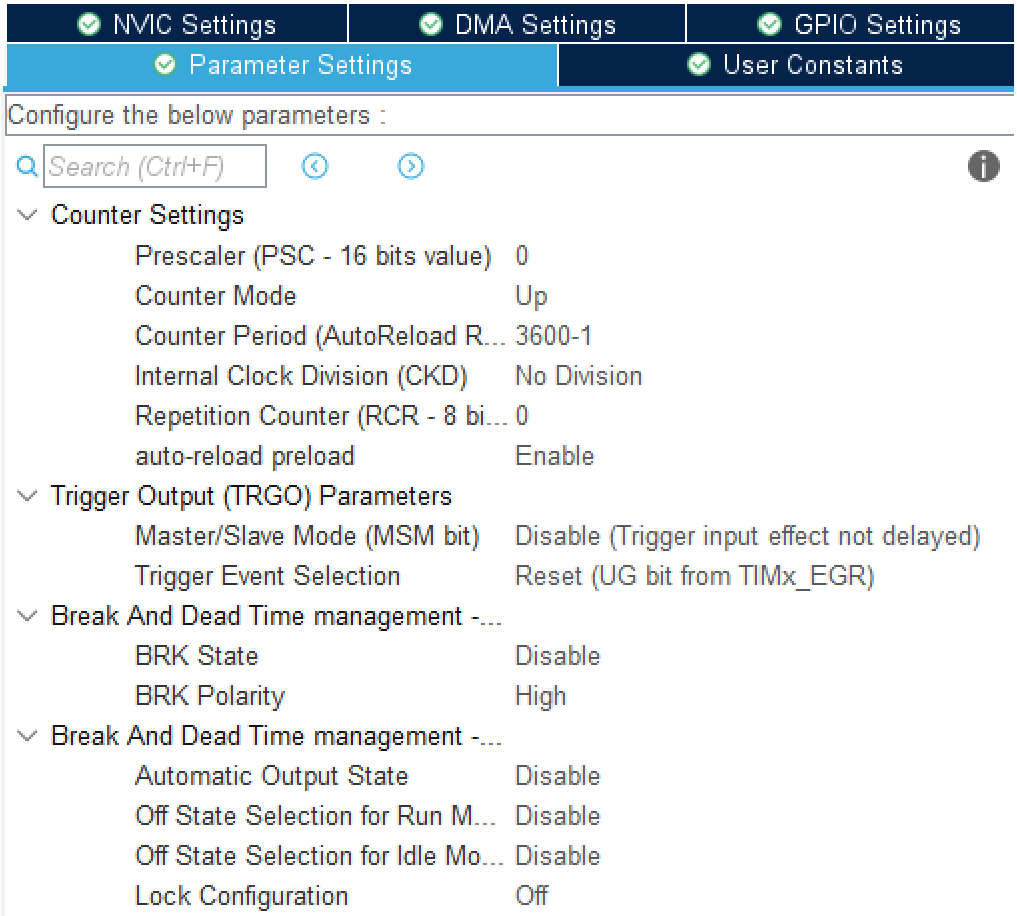
□ Activate-Break-Input

□ Use ETR as Clearing Source

□ *XOR activation*

□ One Pulse Mode

Other parameters are shown below:



Configure the below parameters :

Counter Settings

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

Trigger Output (TRGO) Parameters

Master/Slave Mode (MSM bit) Disable (Trigger input effect not delayed)

Trigger Event Selection Reset (UG bit from TIMx_EGR)

Break And Dead Time management -...

BRK State Disable

BRK Polarity High

Break And Dead Time management -...

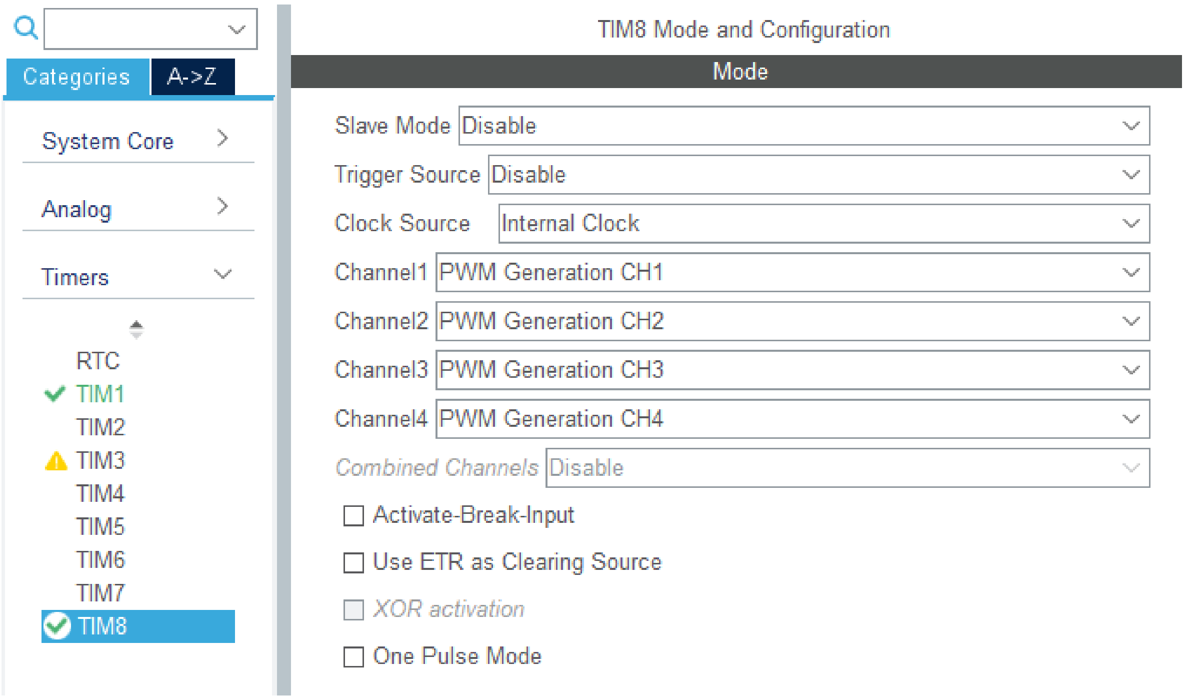
Automatic Output State Disable

Off State Selection for Run M... Disable

Off State Selection for Idle Mo... Disable

Lock Configuration Off

3. Next, set timer 8, clock source selection of internal clock, set the four channel output PWM signal CH1 CH2 CH3 CH4 corresponding to the pin PC6 PC7 PC8 PC9.



TIM8 Mode and Configuration

Mode

Slave Mode Disable

Trigger Source Disable

Clock Source Internal Clock

Channel1 PWM Generation CH1

Channel2 PWM Generation CH2

Channel3 PWM Generation CH3

Channel4 PWM Generation CH4

Combined Channels Disable

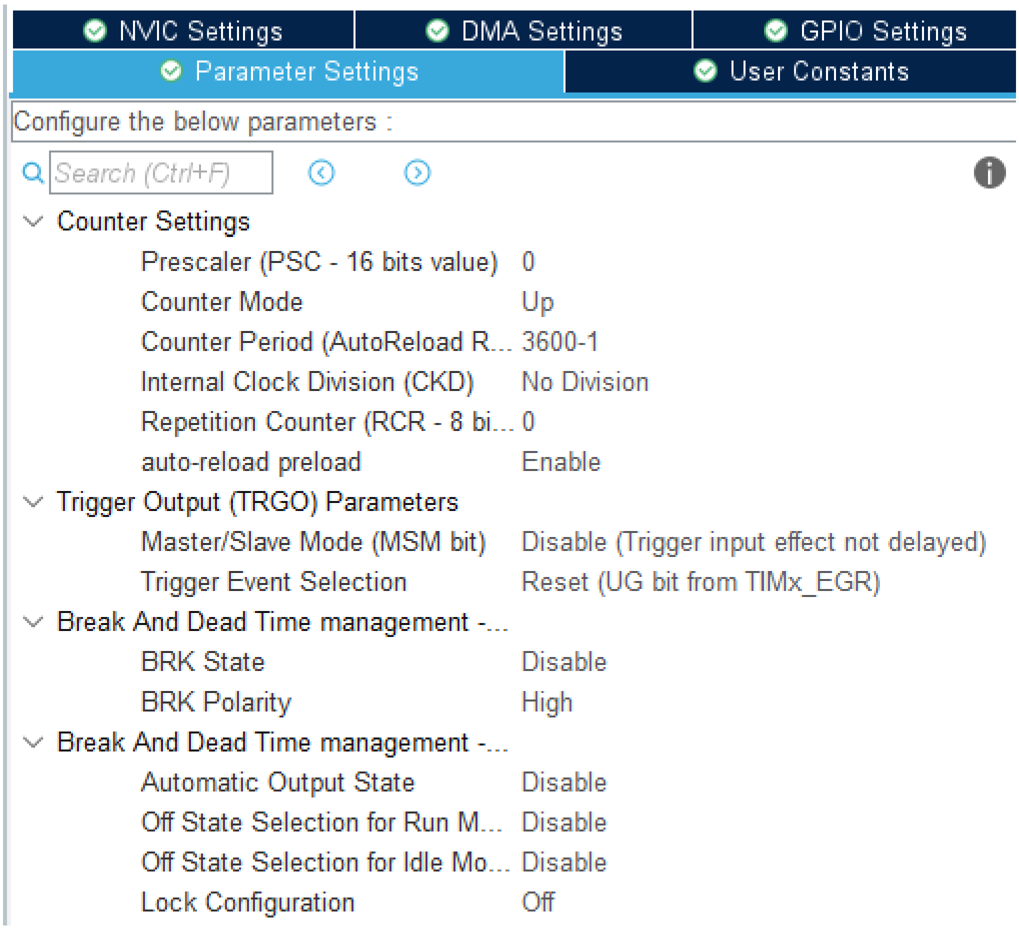
□ Activate-Break-Input

□ Use ETR as Clearing Source

□ *XOR activation*

□ One Pulse Mode

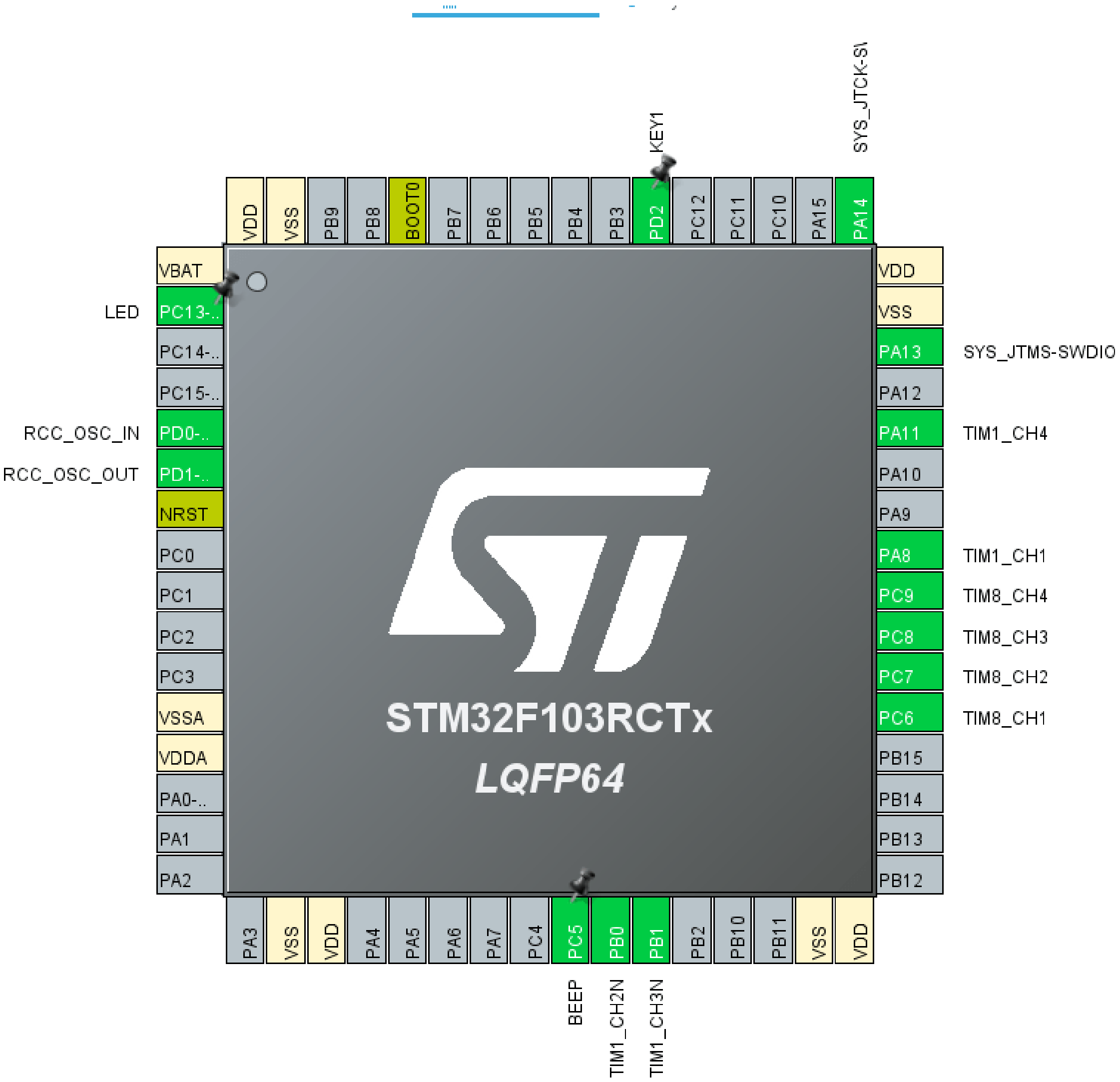
Other parameters are the same as for Timer 1.



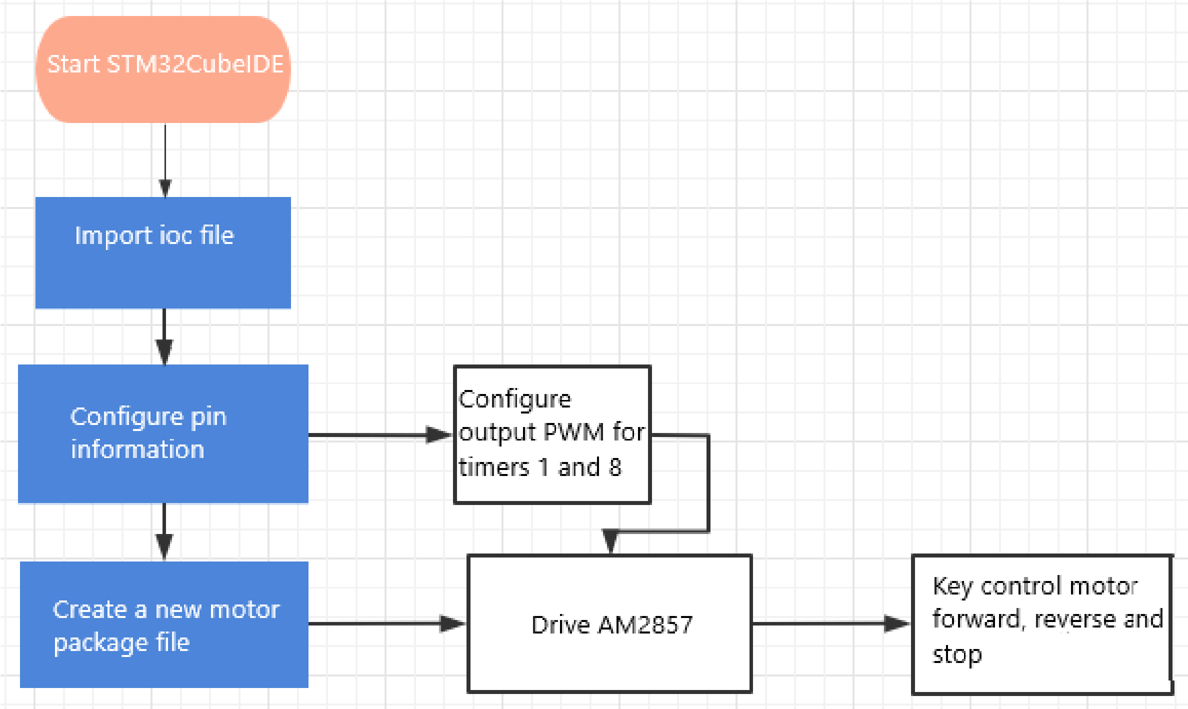
Configure the below parameters :

Counter Settings	
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
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Break And Dead Time management -...	
BRK State	Disable
BRK Polarity	High
Break And Dead Time management -...	
Automatic Output State	Disable
Off State Selection for Run M...	Disable
Off State Selection for Idle Mo...	Disable
Lock Configuration	Off

The final chip configuration pins are shown below:

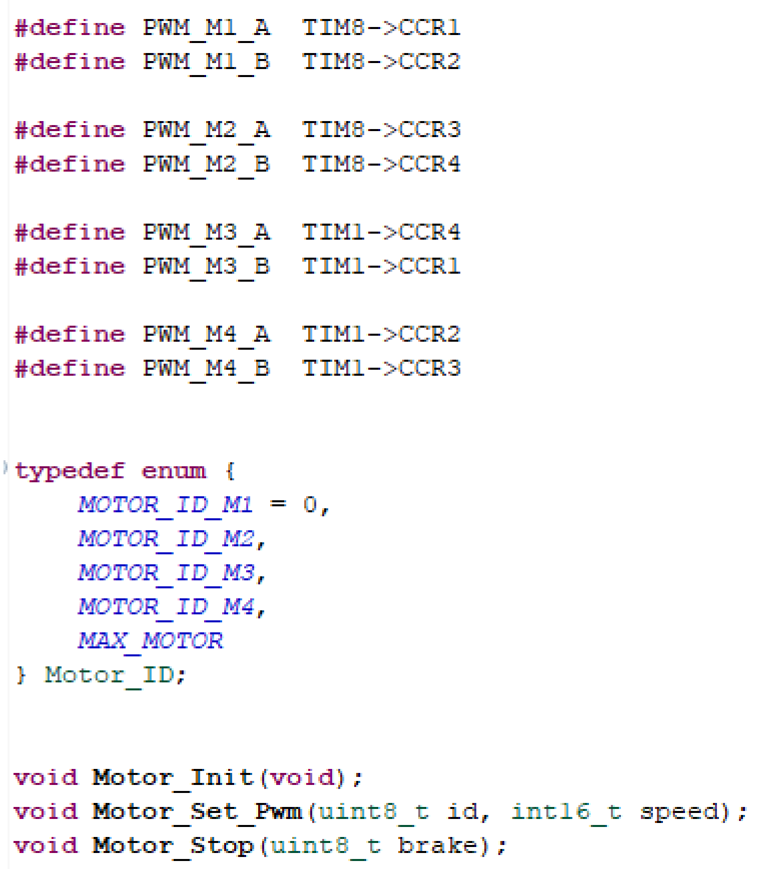


12.3. Experimental flowchart analysis


Start STM32CubeIDE
Import ioc file
Configure pin information
Configure output PWM for timers 1 and 8
Create a new motor package file
Drive AM2857
Key control motor forward, reverse and stop

12.4. Core code explanation

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

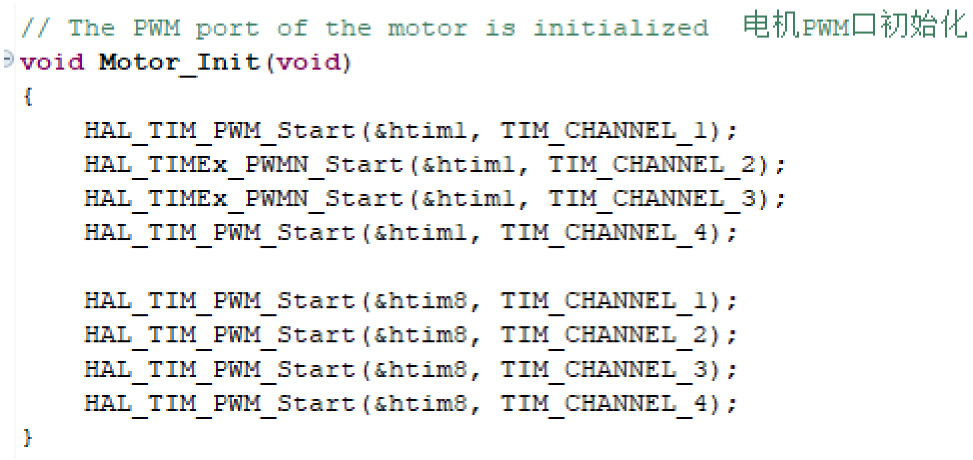


```
#define PWM_M1_A  TIM8->CCR1
#define PWM_M1_B  TIM8->CCR2

#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);
```

Where 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.

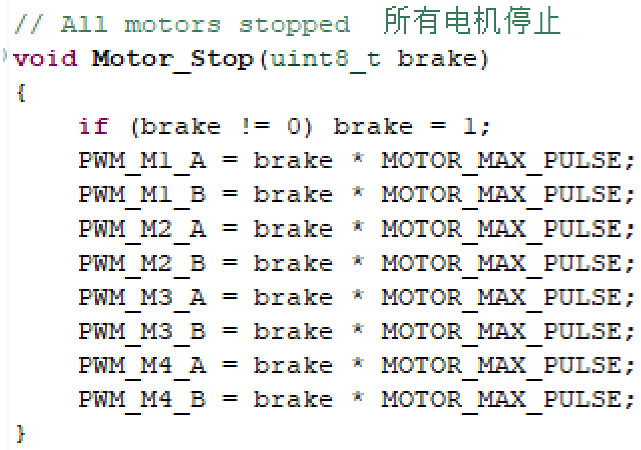
2. In the bsp_motor.c file, create the following new content:

Motor timer PWM output start initialization.



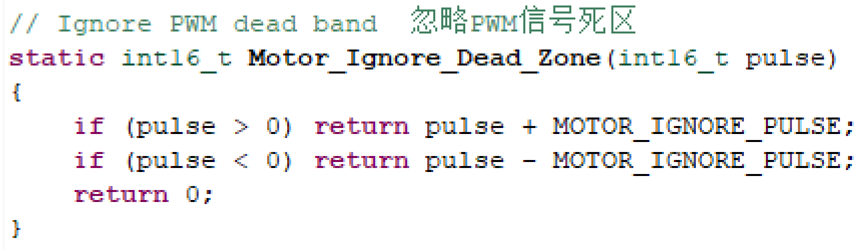
```
// The PWM port of the motor is initialized  电机PWM口初始化
void Motor_Init(void)
{
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
    HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_3);
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);

    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 for brake stop, brake=0 for free stop.



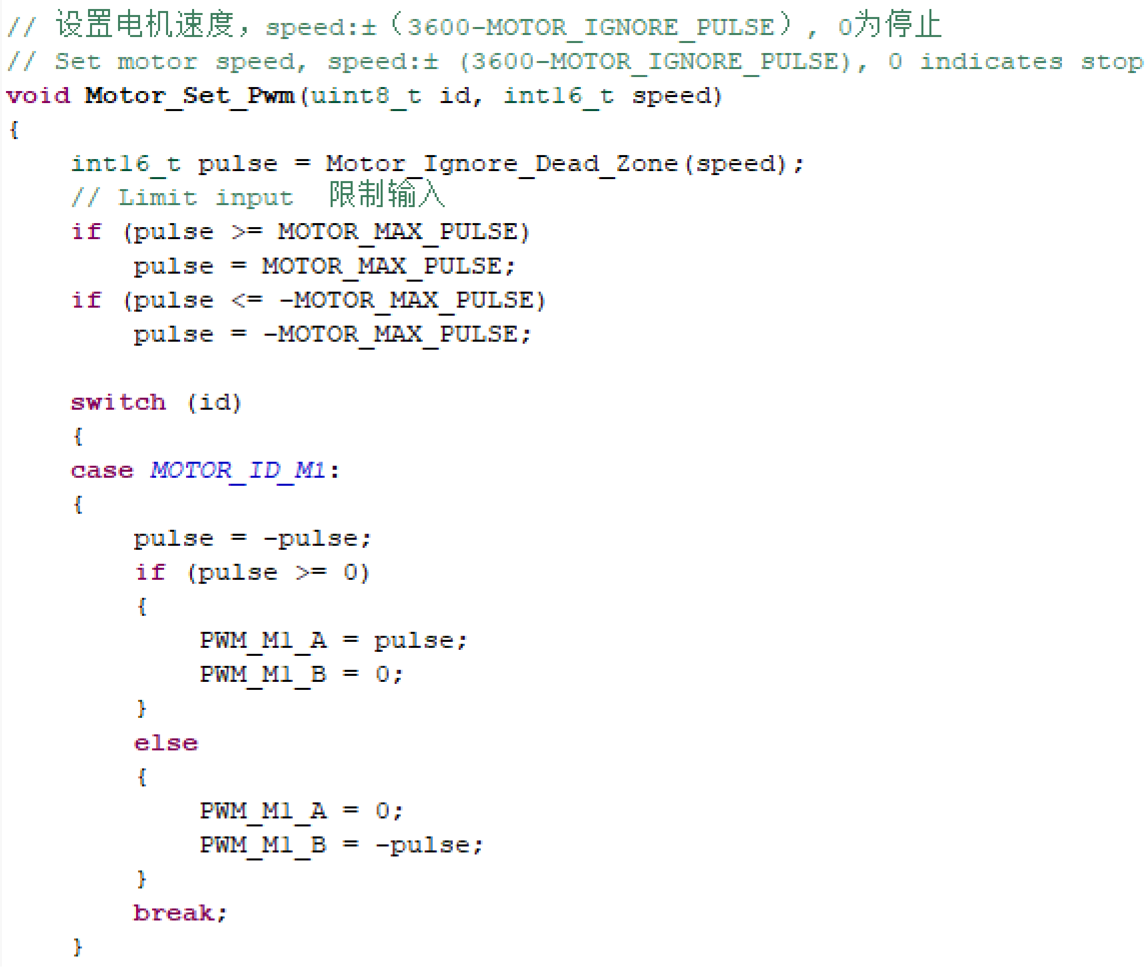
```
// All motors stopped  所有电机停止
void Motor_Stop(uint8_t brake)
{
    if (brake != 0) brake = 1;
    PWM_M1_A = brake * MOTOR_MAX_PULSE;
    PWM_M1_B = brake * MOTOR_MAX_PULSE;
    PWM_M2_A = brake * MOTOR_MAX_PULSE;
    PWM_M2_B = brake * MOTOR_MAX_PULSE;
    PWM_M3_A = brake * MOTOR_MAX_PULSE;
    PWM_M3_B = brake * MOTOR_MAX_PULSE;
    PWM_M4_A = brake * MOTOR_MAX_PULSE;
    PWM_M4_B = brake * MOTOR_MAX_PULSE;
}
```

4. 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 as 0.



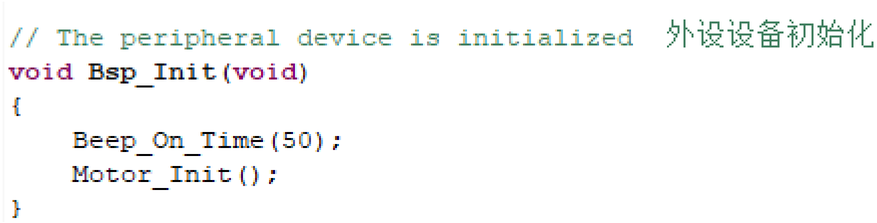
```
// Ignore PWM dead band  忽略PWM信号死区
static int16_t Motor_Ignore_Dead_Zone(int16_t pulse)
{
    if (pulse > 0) return pulse + MOTOR_IGNORE_PULSE;
    if (pulse < 0) return pulse - MOTOR_IGNORE_PULSE;
    return 0;
}
```

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 for stop.



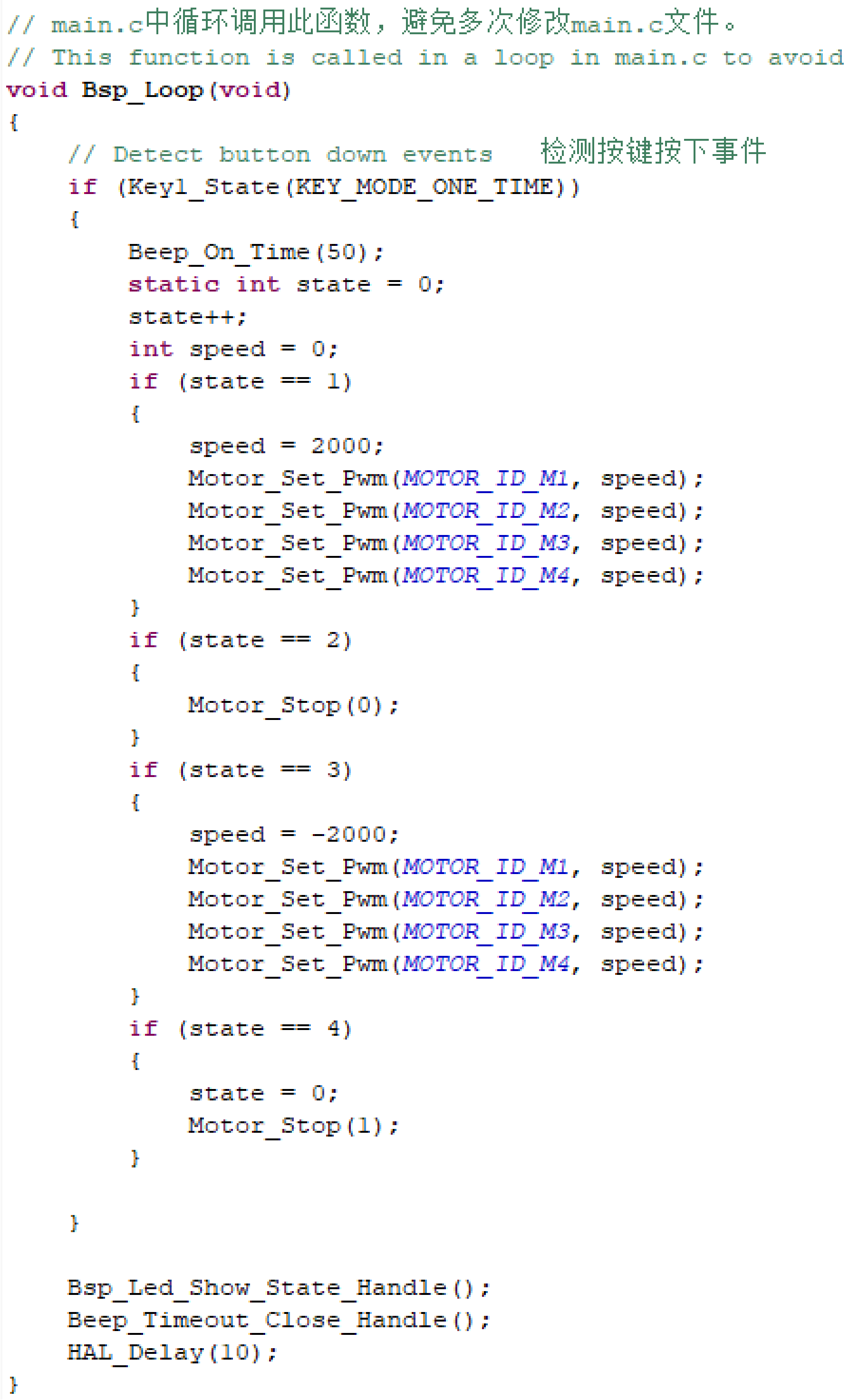
```
// 设置电机速度，speed:±（3600-MOTOR_IGNORE_PULSE），0为停止
// Set motor speed, speed:± (3600-MOTOR_IGNORE_PULSE), 0 indicates stop
void Motor_Set_Pwm(uint8_t id, int16_t speed)
{
    int16_t pulse = Motor_Ignore_Dead_Zone(speed);
    // Limit input  限制输入
    if (pulse >= MOTOR_MAX_PULSE)
        pulse = MOTOR_MAX_PULSE;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

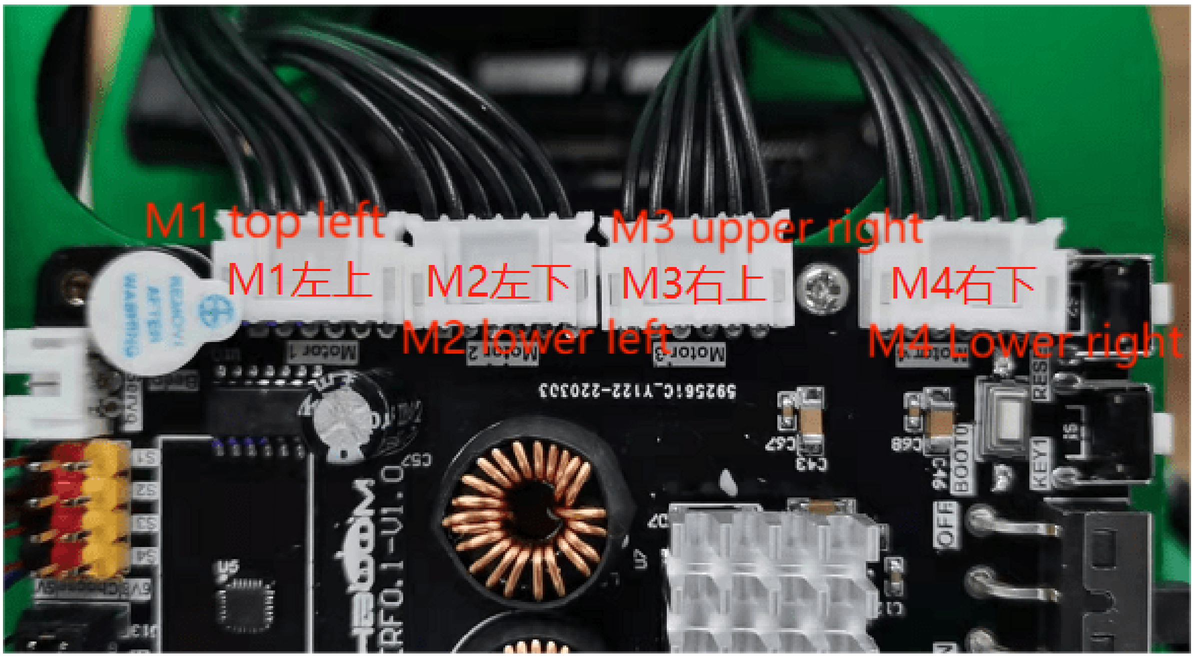
7. In the Bsp_Loop() function add the key to control the motor, press the first forward, the second free stop, the third backward, the fourth brake stop.



```c
// 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 wires need to be connected to the corresponding motors as shown in the following figure, 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.



Due to the relatively large power of the motor, the expansion board should not directly use USB 5V power supply, must use DC 12V power supply.

12.6. Experimental effect

Since the motor will spin when it starts, please set up the cart before the experiment, with the motor wheels hanging in the air, to avoid running across the road.

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