2025

南京宇微电子科技有限公司

DRV8871 H 桥驱动器 V1.0

2025 年 9 月 4 日修订

FU KUN

目录

版本修订 2
客户须知 2
一、概览 3
二、板卡分区介绍 3
2.1 DRV8871 主芯片 3
2.2 电源供电 4
2.3 控制输入和电机输出 4
2.4 输出电流调节 5
三、应用 6
3.1 单直流电机驱动 6
3.2 双直流电机驱动 6
3.3 四电机并联驱动 7
四、原理图 8
五、联系我们 9

版本修订

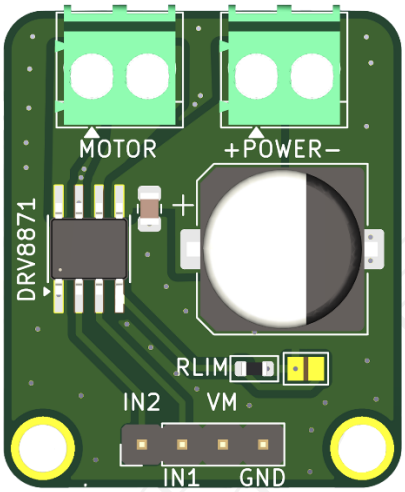
时间	版本号	修订内容
2025 年 9 月 4 日	V1.0	初版

客户须知

本文档为产品使用参考所编写，文档版本可能随时更新，恕不另行通知。本文中提供的所有使用方法、说明及建议仅供参考，不构成任何承诺或保证。使用本产品及本文档内容所产生的结果，由用户自行承担风险。本公司对因使用本文档或产品而导致的任何直接或间接损失，不承担任何责任。

一、概览

DRV8871 模块是一款功能强大的直流电机驱动器，基于德州仪器 DRV8871 芯片，专为控制有刷直流电机设计，适用于机器人、自动化设备等项目。它支持 6.5V 至 45V 宽电压输入，最大电流可达 3.6A，内置过流、过温及欠压保护，确保高可靠性。模块通过简单的电阻设置电流限制，默认 30K 电阻提供约 2A 限流，用户可通过更换电阻调整限流值。支持 PWM 信号控制，IN1 和 IN2 引脚可连接微控制器（如 Arduino），实现电机正反转及速度调节，使用便捷。模块尺寸紧凑（24.4mm x 20.4mm x 9.7mm），配有 2 引脚插针端子接头和杜邦接头，需简单焊接即可使用。我们提供详细教程、Kicad 文件及示例代码，方便用户快速上手。

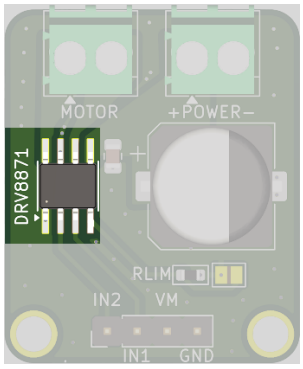


参数	
驱动芯片	DRV8871
供电	6.5~45V
尺寸	约 23mm×53.5mm×1.6mm（具体以实物为准）
重量	约 150g
配件	• 板卡×1 • 用户手册×1（电子版）

二、板卡分区介绍

2.1 DRV8871 主芯片

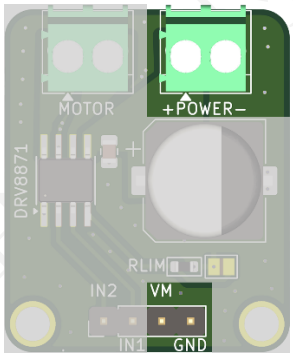
数据手册：DRV8871



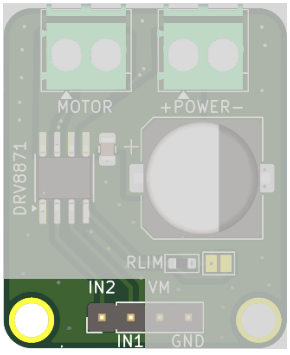
DRV8871 本质上是一颗 H 桥芯片，通过控制 H 桥四个 MOS 开合，达到控制电机正反转和停止的目的。通过控制开关频率，达到控制电机转速的目的。也就是说通过这样一颗单芯片，实现对电机的转动方向控制和转动速度控制。

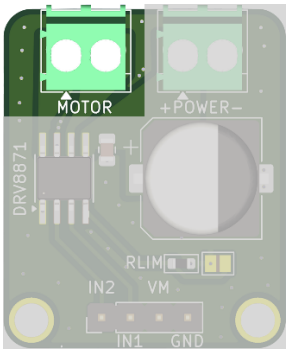
2.2 电源供电

电源输入接口有两个，第一个是螺钉式接线端子，通过线缆连接至 6.4V 至 45V 的供电器或电池。另一个使用杜邦线，连接 VM 端和 GND 端。因为杜邦线通过电流比较小，大电流发热严重，建议优先使用螺钉式接线端子供电。



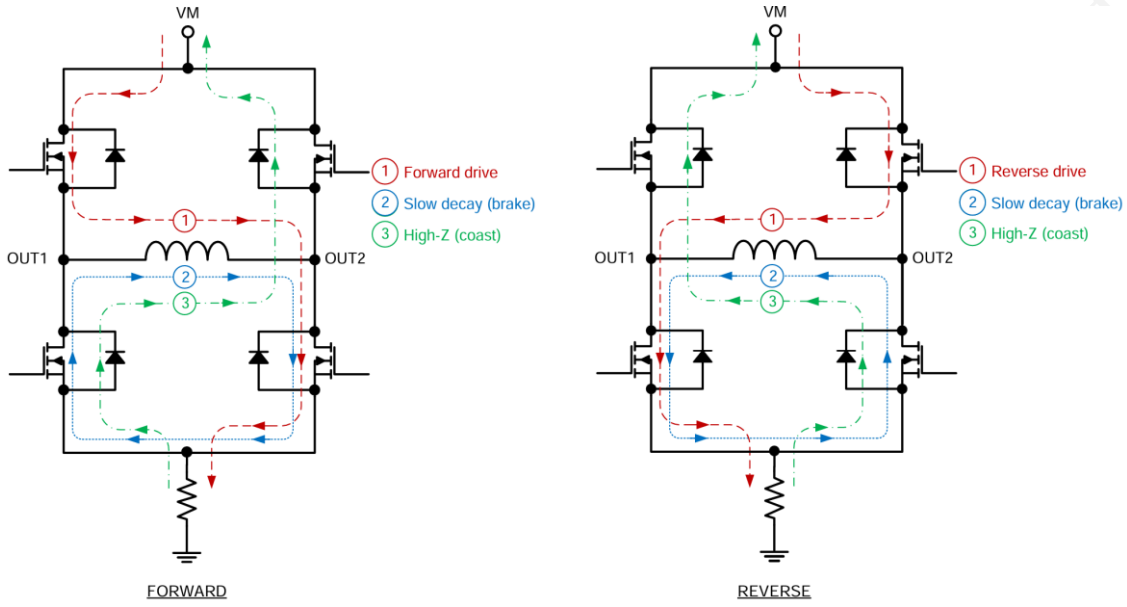
2.3 控制输入和电机输出





IN1	IN2	OUT1	OUT2	DESCRIPTION
0	0	High-Z	High-Z	Coast; H-bridge disabled to High-Z (sleep entered after 1 ms)
0	1	L	H	Reverse (Current OUT2 → OUT1)
1	0	H	L	Forward (Current OUT1 → OUT2)
1	1	L	L	Brake; low-side slow decay

数据手册中给到了一张 H 桥的工作原理



一个 H 桥有四个 NMOS 组成，这里简单将其看作开关管，就是说栅极输入高电平 MOS 打开，栅极输入低电平 MOS 关闭。值得注意的是，模块的两个输入端 IN1 和 IN2 并非直接控制 H 桥的四个栅极，而是通过逻辑控制器和一对 MOS 控制器间接连接。

IN1	IN2	左上	左下	右上	右下	OUT1	OUT2
0	0	关	开	关	开	缓慢停止	缓慢停止
0	1	关	开	开	关	电流输出	电流输入
1	0	开	关	关	开	电流输入	电流输出
1	1	关	关	关	关	迅速停止	迅速停止

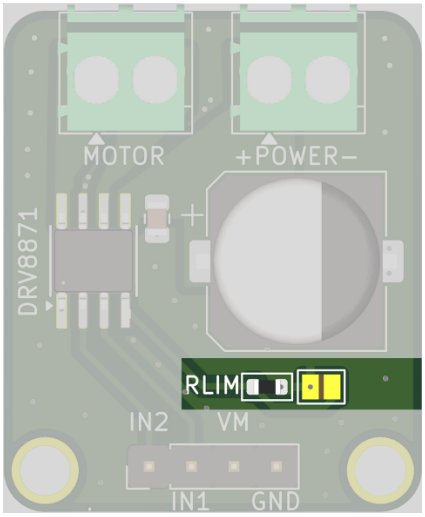
缓慢停止，是指在下桥臂形成一个续流环路，当 MOS 管均打开时，因为有体二极管存在，是双向导通的，这时电机会依靠惯性缓慢停止。迅速停止也就是刹车，电机感应电动势产生的感应电流方向从供电电源的负极流向正极，感应电动势与供电电压方向相反，所以感应电流会迅速衰减，并储存在附近的电容中。

2.4 输出电流调节

外部电阻（R_LIM）连接在 ILIM 引脚与地之间。DRV8871 模块默认焊接一个 30kΩ 电阻，对应约 2A 的电流限制。电流计算公式为

$$I_{LIM} = \frac{60}{R_{LIM}(k\Omega)} = \frac{60}{30} = 2A$$

通过更换 R_LIM 电阻，用户可调整电流限制值，理想条件下最高允许提供 3.6A 电流。

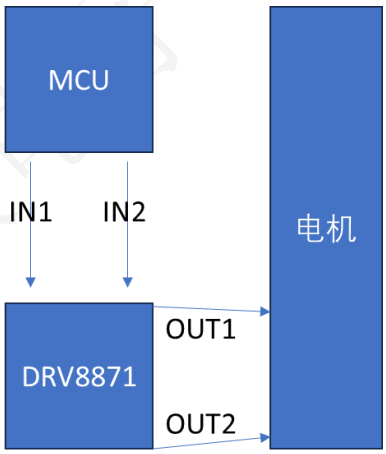


三、应用

3.1 单直流电机驱动

单电机可以控制电机转动方向和转动速度，电机的两条线分别接到 OUT1 和 OUT2。

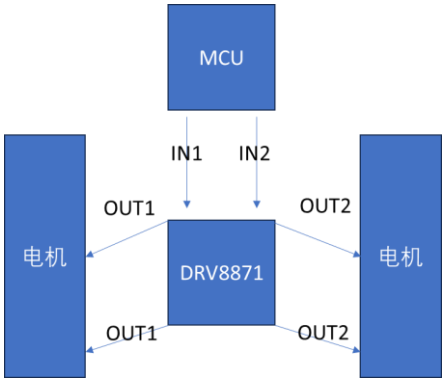
应用的场景是需要单电机灵活控制的产品，如电风扇、舵机等。



3.2 双直流电机驱动

双电机可以控制转动速度，每个电机的一条线接 GND，另一条线分别接 OUT1 和 OUT2。控制逻辑是用 IN1 高低电平间接控制一个电机的启动和停止，用 IN2 高低电平间接控制另一个电机的启动和停止。使用 IN1 和 IN2 输出的 PWM 波，间接控制电机的转速。

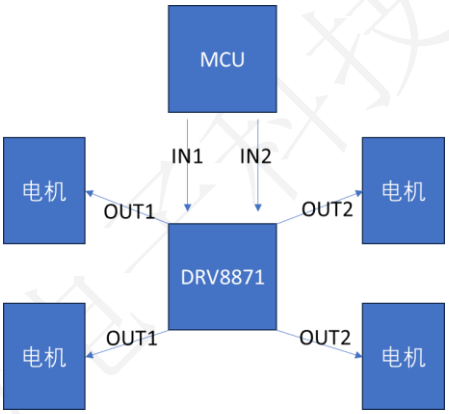
可应用在两个独立后驱轮、一个前转向轮的小车运动控制中。



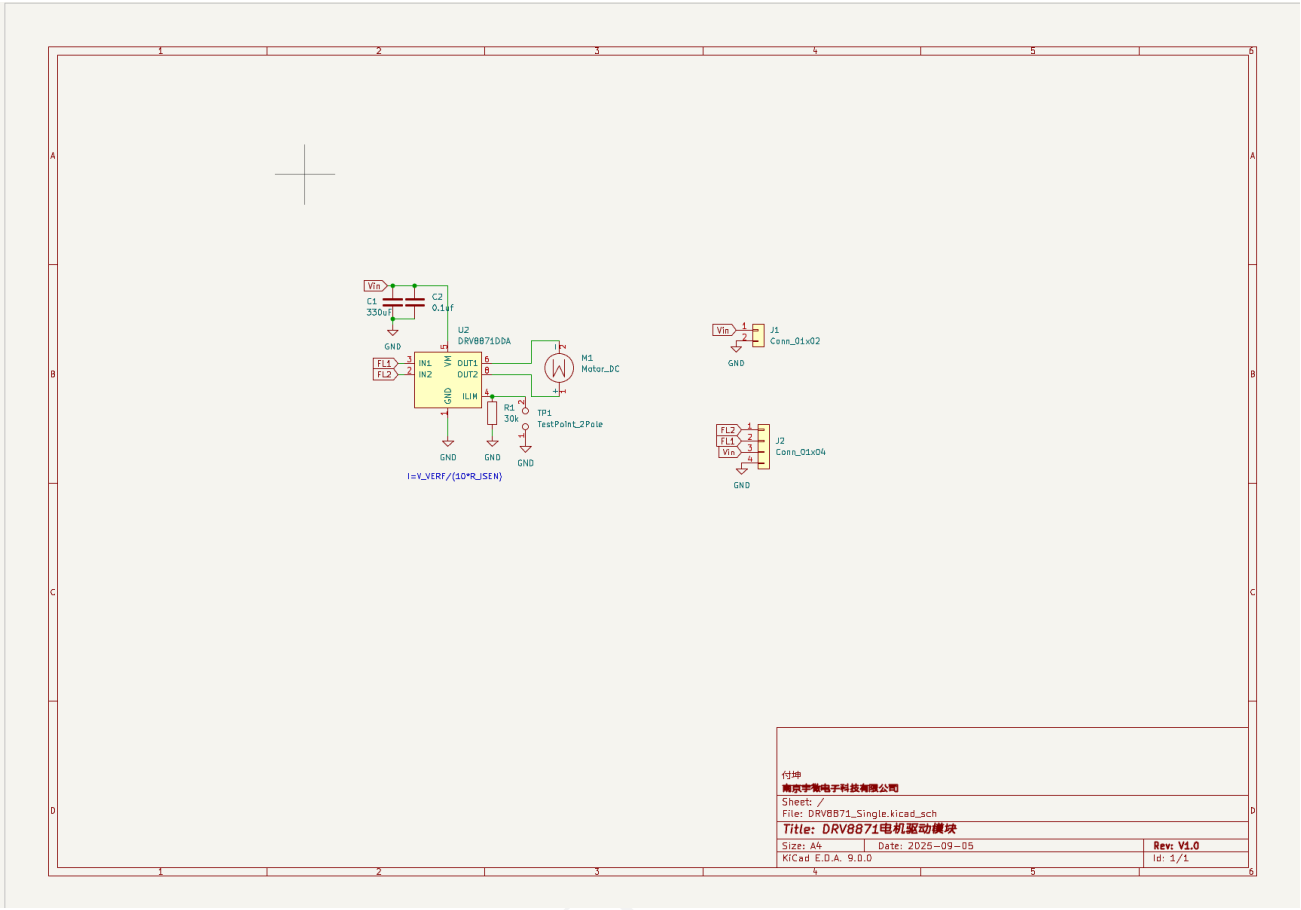
3.3 四直流电机驱动

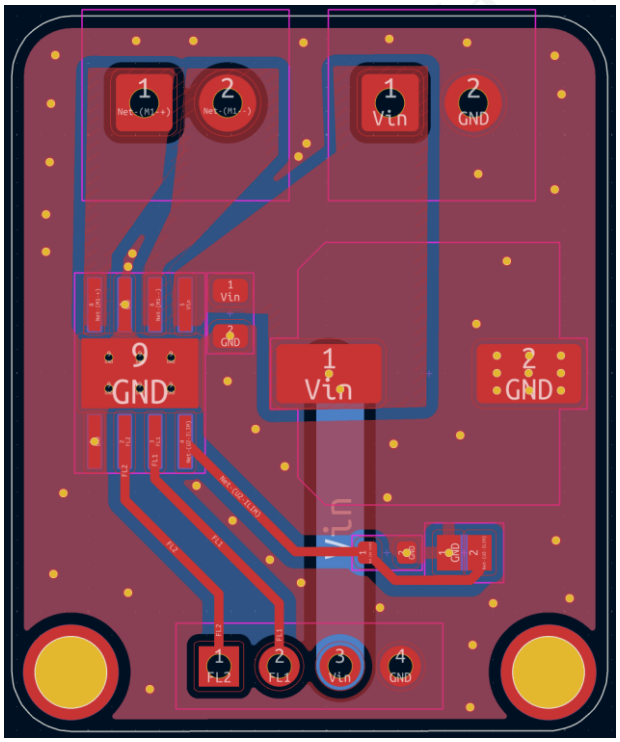
四电机是双电机方案的延申，也只能控制电机转动的速度。由同一个 OUT 口控制的所有电机，均响应相同的运动指令。

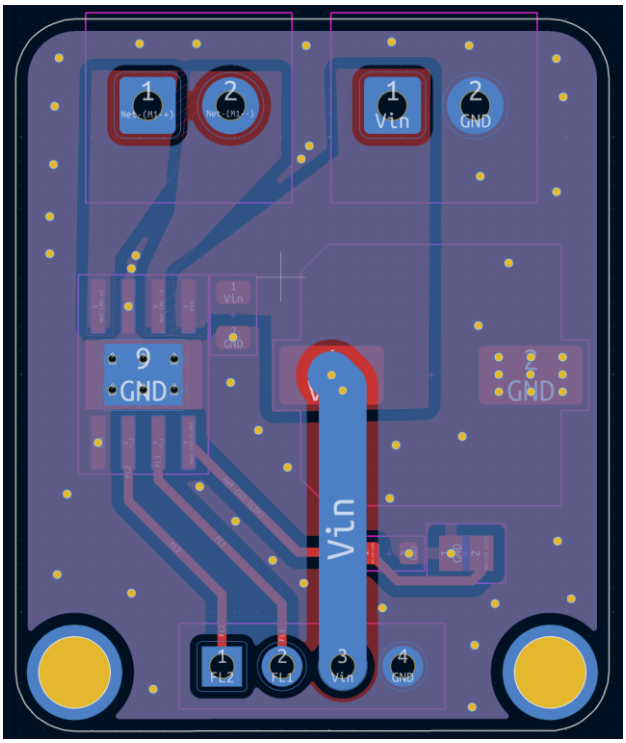
可应用在四个独立驱动轮的小车运动控制中。



四、原理图与 PCB 布线


Vin
C1
330uF
C2
0.1uF
GND
U2
DRV8871DDA
FL1
FL2
IN1
IN2
VM
OUT1
OUT2
GND
ILIM
M1
Motor_DC
R1
30k
TP1
TestPoint_2Pole
I=V_VERF/(10*R_ISEN)
J1
Conn_01x02
J2
Conn_01x04
付坤
南京宇鲨电子科技有限公司
Sheet: /
File: DRV8871_Single.kicad_sch
Title: DRV8871电机驱动模块
Size: A4
Date: 2025-09-05
KiCad E.D.A. 9.0.0
Rev: V1.0
Id: 1/1


GND
Vin
FL2
FL1


GND
Vin
FL2
FL1

五、编程指南

5.1 基于 STM32CubeMX IDE

下载和安装这里不再赘述，请直接查看这篇文章：

5.1.1 基于 GPIO 输出 PWM 控制

以下是为 STM32F103C8T6 微控制器设计的基于 HAL 库 GPIO 控制的加减速函数，用于控制 DRV8871 模块驱动电机，实现速度变化。此方案使用 GPIO 模拟 PWM 信号，通过软件定时（基于 HAL_Delay）控制 DRV8871 的 IN1 和 IN2 引脚，以实现正反转和速度调节。

5.1.1.1 常量定义

```
volatile int16_t duty = 50; // Current speed
const uint16_t PWM_CYCLE = 10;          // PWM period (10ms, ~100Hz)
```

5.1.1.3 generatePWM 函数

```
// Generate software PWM for IN1/IN2 based on duty (-100 to 100)
void generatePWM(int16_t duty) {
    uint32_t onTime, offTime;
    duty = duty> 100 ? 100 : (duty < -100 ? -100 : duty);
    if (duty >= 0) {
        onTime = (duty * PWM_CYCLE) / 100; // On-time in ms
        offTime = PWM_CYCLE - onTime;
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);     // IN1 HIGH
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET); // IN2 LOW
        HAL_Delay(onTime);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET); // IN1 LOW
        HAL_Delay(offTime);
    } else {
        onTime = ((-duty) * PWM_CYCLE) / 100; // On-time for reverse
        offTime = PWM_CYCLE - onTime;
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET); // IN1 LOW
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);     // IN2 HIGH
        HAL_Delay(onTime);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET); // IN2 LOW
        HAL_Delay(offTime);
    }
}
```

本函数中使用 PA0 和 PA1 两个 GPIO 引脚输出任意 PWM 占空比的函数，PWM 周期由 PWM_CYCLE 常量决定，通过不同的占空比调节速度。使用 if-else 语句，通过输入正负占空比（±duty）值调节电机转动速度，通过对不同输入“±”号的判断调节电机转动的方向。这里的 duty 是一个限定±100 之间的数，单位是百分比%。

5.1.1.5 main 主函数

```
while (1)
{
  generatePWM(50);
}
```

这里我们仅展示 while 循环的内容，设定一个速度“50”前向转动占空比。

5.1.2 基于 Timer 输出 PWM 控制

```
volatile int16_t targetSpeed = 50;   // Target speed (-1000 to 1000, maps to 0-100% PWM)
volatile int16_t currentSpeed = 0; // Current speed
```

5.1.2.1 定时器中断回调

```
// Timer interrupt callback
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
  if (htim->Instance == TIM2) {
      currentSpeed= targetSpeed > 100 ? 100 : (targetSpeed < -100 ? -100 : targetSpeed);
      updateMotor();
  }
}
```

此函数是 STM32 HAL 库中的定时器中断回调，配合前述代码，用于 TIM2 定时器，触发周期为 50ms（由 htim2.Init.Prescaler = 7199 和 htim2.Init.Period = 99 确定：72MHz / (7199+1) / (99+1) = 100Hz）。这里借助定时器中断，每隔 10ms 进入一次这个函数，用来更新速度，与 5.1.1 中使用 for 循环和延迟操作的逻辑一致。

5.1.2.2 updateMotor 函数

```
// Update motor PWM based on currentSpeed
void updateMotor(void) {
  if (currentSpeed >= 0) {
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, currentSpeed); // IN1 PWM
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, 0);                  // IN2 LOW
  } else {
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 0);                  // IN1 LOW
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, -currentSpeed); // IN2 PWM
  }
}
```

通过 TIM2 定时器的 PWM 输出（PA0 为 TIM2_CH1，PA1 为 TIM2_CH2）控制 DRV8871 模块的 IN1 和 IN2 引脚，实现电机正反转和速度调节。在 currentSpeed≥0 时，设置 TIM2 通道 1 (PA0, IN1) 的 PWM 占空比为 currentSpeed。在 currentSpeed≤0 时，设置 TIM2 通道 2 (PA0, IN1) 的 PWM 占空比为 -currentSpeed。其余两种占空比为 0 的情况，意味着持续的低电平。

5.1.2.3 MX_TIM2_Init 函数

```
/* TIM2 init function */
void MX_TIM2_Init(void)
```

```
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 7199;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 999;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  {
```

```
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1); // 启动通道 1 的 PWM
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2); // 启动通道 2 的 PWM

  HAL_TIM_Base_Start_IT(&htim2);              // 启动 TIM2 并启用更新中断
  /* USER CODE END TIM2_Init 2 */
  HAL_TIM_MspPostInit(&htim2);

}
```

这段代码在 STM32CubeMX 配置时就已经存在了，无需额外编写代码。

单片机主频设置为 72MHz，预分频是 7199，即进入定时器的主频是 10KHz，周期是 499，所以计数一个周期是 50ms。然后开启计数器以及中断回调，实现每 50ms 进入一次回调函数更新速度值。

紧接着配置 PWM 模式 1，初始占空比输出为零，应用并启用 CH1 和 CH2。

5.1.2.3 MX_TIM2_Init 函数

```
        targetSpeed = -100;
        HAL_Delay(1000);
        targetSpeed = -75;
        HAL_Delay(1000);
        targetSpeed = -50;
        HAL_Delay(1000);
        targetSpeed = -25;
        HAL_Delay(1000);
        targetSpeed = 0;
        HAL_Delay(1000);
        targetSpeed = 25;
        HAL_Delay(1000);
        targetSpeed = 50;
        HAL_Delay(1000);
        targetSpeed = 75;
        HAL_Delay(1000);
        targetSpeed = 100;
        HAL_Delay(1000);
        targetSpeed = 75;
        HAL_Delay(1000);
        targetSpeed = 50;
```

```
        HAL_Delay(1000);
        targetSpeed = 25;
        HAL_Delay(1000);
        targetSpeed = 0;
        HAL_Delay(1000);
        targetSpeed = -25;
        HAL_Delay(1000);
        targetSpeed = -50;
        HAL_Delay(1000);
        targetSpeed = -75;
        HAL_Delay(1000);
```

六、联系我们

若需任何帮助，请邮件联系我们：info@fukunlab.com

样品购买：淘宝店铺-宇微电子