摘 要

本文是对于基于STM32嵌入式开发平台，设计并实现一个四轴飞行器步骤的描述。本文首先介绍了关于转接板的绘制问题，由于需要一个转接板连接开发板和四轴木块，故利用Altium Designer软件进行pcb板的绘制。并主要完成了使用遥控器实现电机旋转以及转速的调整。由于无刷直流电机的调速需要PWM波输出合适的数字模型，并且在利用STM32工作原理输出呼吸灯来初步实现PWM波的时候，需要通过占空比来控制灯亮灭的情况。在进行转速调整的时需要利用PWM输出电机实验，通过手动代码设置PWM波输出，从而达到控制四个电机常启动并保持在一定转速的目标。并且进行PWM输入捕获到输出驱动电机综合实验，完成捕获遥控器发出的PWM波，通过算法得出遥控器PWM的占空比，然后将占空比赋给四个电调，从而达到一个遥控器遥控四个电机转速的目的。最后使用遥控实现电机旋转以及转速调整。

**关键词：STM32，四轴飞行器，PCB，PWM**

# 目 录

第一章 课程设计概述 1

1.1 课程设计目的 1

1.2 课程设计任务 1

第二章 四轴飞行器的设计及实现 2

2.1 四轴飞行器的设计 2

2.1.1 前言 2

2.1.2 项目概述 2

2.1.3 需求规定 3

2.2 四轴飞行器的实现 4

2.2.1 转接板的实现 4

2.2.2 电机旋转以及转速调整的实现 6

第三章 存在问题与解决方案 18

3.1 绘制转接板中遇到的问题及解决方案 18

3.2 电机旋转以及转速调整的实现中遇到的问题及解决方案 18

第四章 总结及任务分工情况 20

4.1 总结 20

4.2 任务分工情况 20

参考文献 21

致谢 22

# 第一章 课程设计概述

## 1.1 课程设计目的

基于STM32嵌入式开发平台，以2-5个同学为单位，设计并实现一个四轴飞行器。

## 1.2 课程设计任务

本学期的主要任务分为两部分，四轴飞行器的机械部件搭建和硬件系统设计。机械部件搭建要求购买器材并搭建飞行器骨架，安装stm32核心板，自制转接板将各模块接入stm32核心板。硬件系统设计则要求通过C语言编写程序，驱动各外设，包括接受遥控器的遥控信号，然后通过这个信号操作四个马达同步旋转，并控制马达转速。

# 第二章 四轴飞行器的设计及实现

## 2.1 四轴飞行器的设计

2.1.1 前言

本章将根据实际情况对现场控制系统综合设计1的需求进行分析与阐述，说明相关功能点的具体内容。

1、目的

（1）需求分析是一个项目必不可少的组成部分，有了正确的需求分析，才能保证软件开发的质量、需求的完整与可追溯性。通过编写此需求分析说明书，旨在对目标项目提出完整、清晰、准确、具体的要求，以保证需求分析人员、开发人员、测试人员及相关利益人对需求达成共识。

（2）本说明书主要面向四轴飞行器课程设计的开发人员、测试人员以及验收人员。

2、范围

（1）名称：四轴飞行器综合设计1

（2）应用：基于windows平台上的AD19软件，CubeMX，KeiluVision5

3、 定义、缩写词、略语

GY86:该模块集成了加速度传感器，气压传感器，磁场强度传感器以及陀螺仪。

PWM：脉冲宽度调制，它是通过对一系列脉冲的宽度进行调制，等效出所需要的波形（包含形状以及幅值），对模拟信号电平进行数字编码，也就是说通过调节占空比的变化来调节信号、能量等的变化。

占空比：在一个周期内，信号处于高电平的时间占据整个信号周期的百分比。

2.1.2 项目概述

1、产品描述

产品的最终形态是一个基本的四轴飞行器骨架，该骨架包括基本的控制板stm32，自己设计的线路转接板，四个马达。

2、产品功能

实现对四轴飞行器骨架的搭建，并且能够独立运行各个部件的功能。

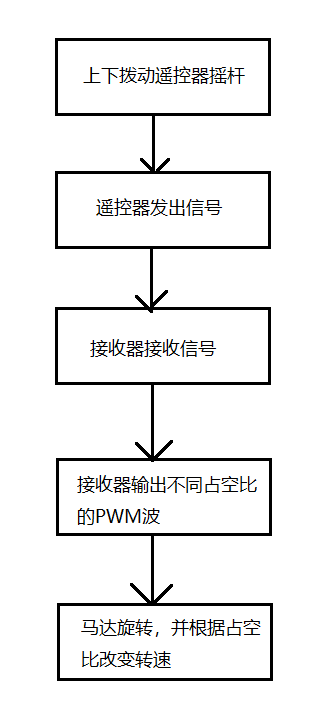
功能模块图如图2-1所示：

图2-1 功能模块示意图

3、用户特点

本系统只对具有相关知识的该项目的开发人员开放。

2.1.3 需求规定

1、功能需求

功能需要实现的目标：1.搭建四轴飞行器的骨架2.制作能够完整转接stm32的转接板3.能够使用遥控器操控电机的同步旋转以及转速调整。

2、设计约束

（1）开发及应用平台的限制：

开发平台为windows，通过keli进行编程，应用平台为stm32f401核心板。

（2）硬件的限制

硬件配置：stm32f401核心板

## 2.2 四轴飞行器的实现

2.2.1 转接板的实现

由于开发板的版型限制，制作四轴需要的模块不能很好的安装在开发版上，所以我们需要制作一块转接板。这块转接板主要用于模块的转接，这有利于我们后期的调试。同时，转接板对整个四轴的稳定性起到了很重要的作用[1]。

对于转接板，我们可以使用Altium Designer软件自行绘制，同时需要stm32f401re的长宽数据，并了解各个引脚对应的功能。绘制转接板分为绘制转接板原理图和绘制转接板pcb图。绘制流程见图2-2：

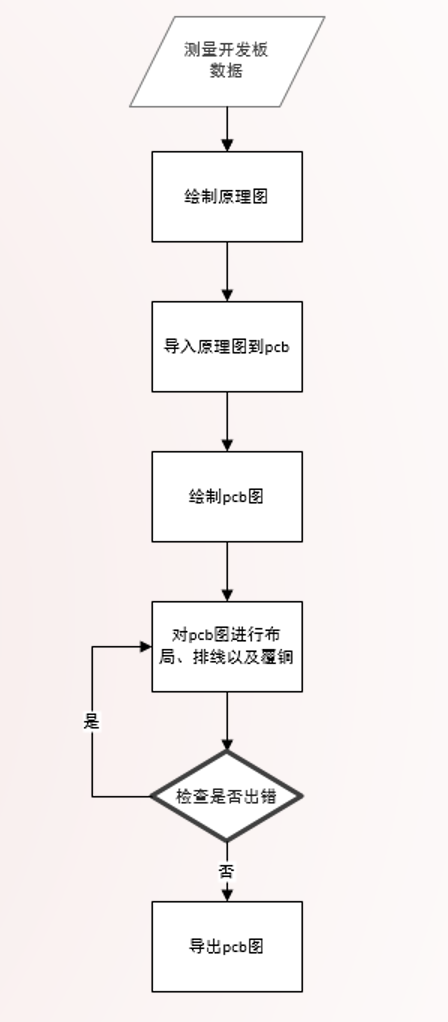


图 2‑2 转接板实现流程图

具体步骤如下[2]：

1. 先绘制原理图，将各个模块的引脚接到管脚上，完成电气连接，绘制好的原理图见图2-3。

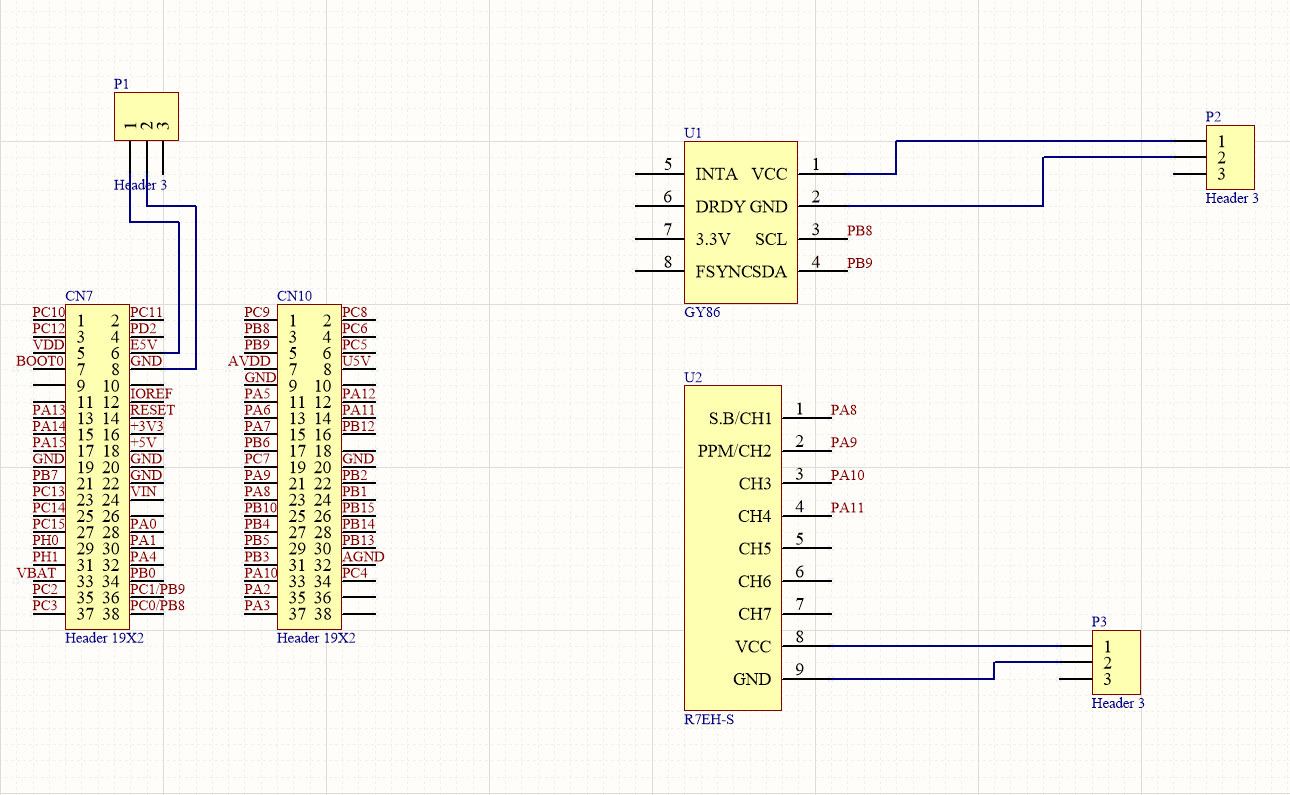


图 2‑3 原理图

1. 将原理图导入到pcb中，此时得到了具有电气连接的pcb模块。
2. 绘制转接板的边界，即机械层，这会决定转接板的具体形状与大小。
3. 将pcb模块放入机械层内，并对其进行布局，连线，覆铜。
4. 检查pcb图中是否存在错误，如果没有错误。导出pcb图，联系淘宝店铺进行打板。pcd图见图2-4(a)、图2-4(b)。

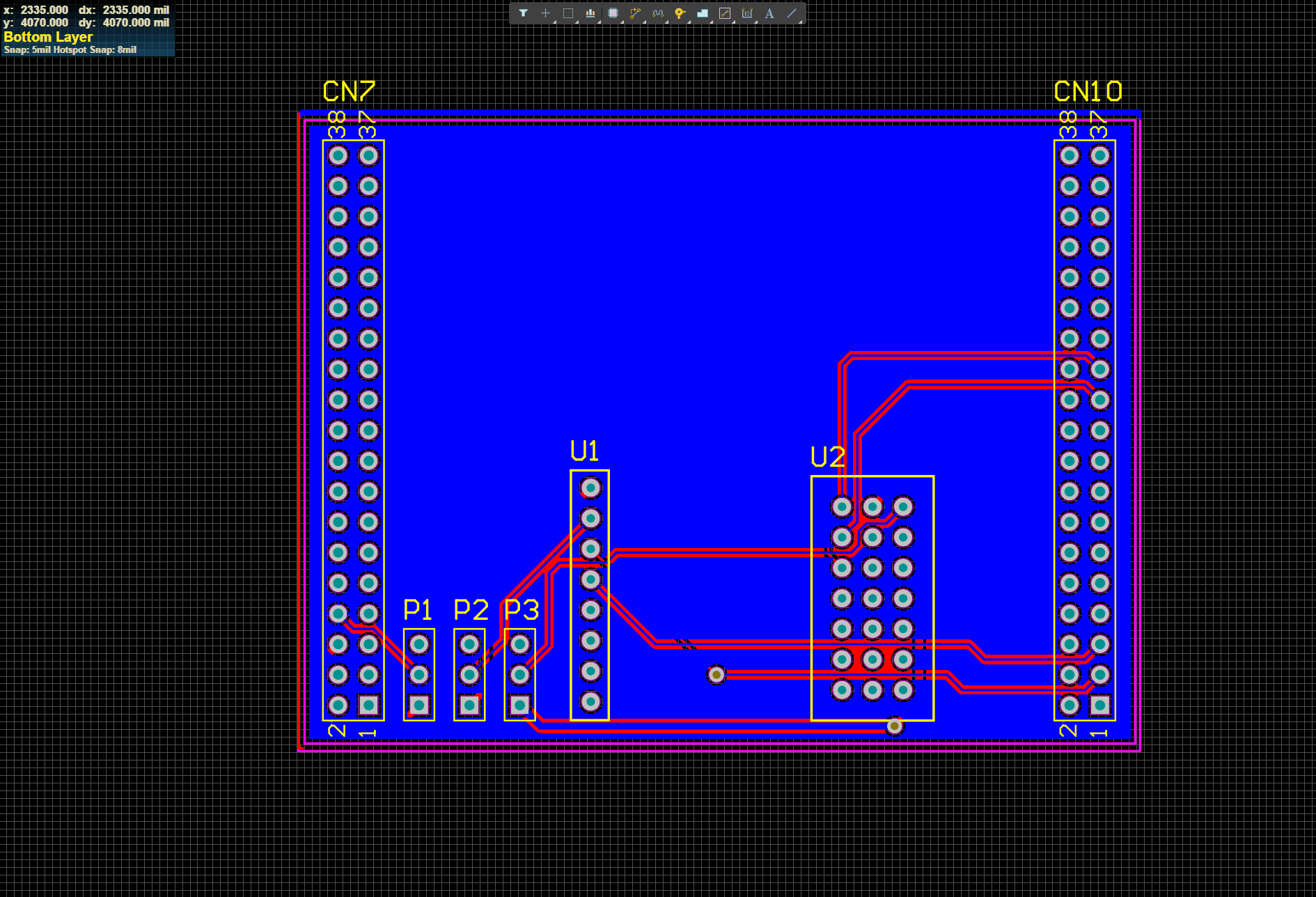


图 2‑4 pcb图(a)

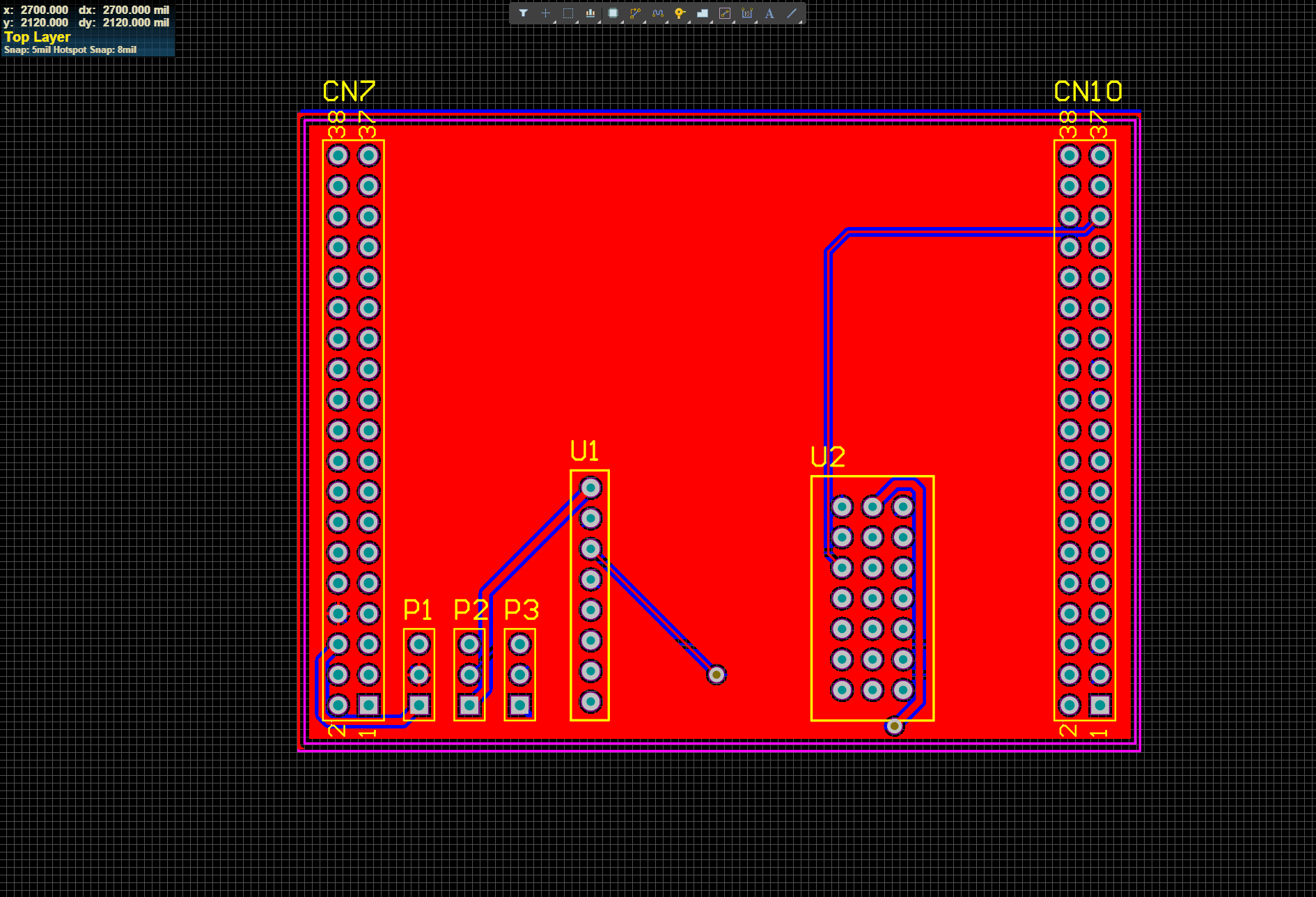


图 2‑4 pcb图(b)

1. 在转接板上进行模块的焊接与测试。

2.2.2 电机旋转以及转速调整的实现

2.2.2.1 关于跑马灯的实现

在初步学习STM32后，首先进行跑马灯实验，跑马灯即是在LED小灯亮后，完成使其指示不同的工作状态的行为。因为STM32F401RET6的PA5脚连接了小灯，所以当PA5复位时，灯灭，PA5置位时，灯亮，不断置复位则达到跑马灯的效果。跑马灯实现的核心代码见代码2-1。

代码2-1 跑马灯实现的核心代码

|  |
| --- |
| //初始化GPIO  RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);//使能GPIO时钟使得GPIO口可以正常工作  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5;//选取PA5  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT;//设置为普通输出模式  GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;//选择普通输出中的推挽输出  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz;//设置GPIO口输出频率为100M赫兹  GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;//设置为上拉  GPIO\_Init(GPIOA, &GPIO\_InitStructure);//初始化GPIO使以上设置生效  //操作IO口  //反复调用GPIO\_ResetBits()和GPIO\_SetBits()函数来不断将PA5脚复位（置低电平）和置位（置高电平）  GPIO\_ResetBits(GPIOA,GPIO\_Pin\_5);  GPIO\_SetBits(GPIOA,GPIO\_Pin\_5); |

2.2.2.2 关于呼吸灯的实现

关于无刷直流电机的调速需要PWM波形输出合适的数字模型，首先了解PWM，即脉冲宽度调制。

我们选择借助在STM32输出呼吸灯来初步实现PWM波，并且需要通过占空比来控制的灯亮灭情况，看起来就像人在呼吸的样子，即为呼吸灯。

定时器会循环计数，其中ARR为计数的上限值，称为自动重装载值，而位于0和ARR之间的CCRx可以取[0,ARR)中的任意整数，称为比较值，当脉冲宽度调制模式为1时，计数值<CCRx会产生有效电平，计数值>CCRx会产生无效电平，而设置输出比较极性低时，有效电平为低电平，无效电平为高电平，则说明：当0≤计数值<CCRx时，通道会输出低电平，当CCRx<计数值<ARR时通道会输出高电平（如图2-5所示），而占空比就是一次循环中，高电平持续时间除以整个周期，即下图的（t2-t1）/t2。时钟来源为APB1，经过两倍的倍频之后为84M，除以（预分频系数psc+1）后即为计数频率（1s计数多少次）。

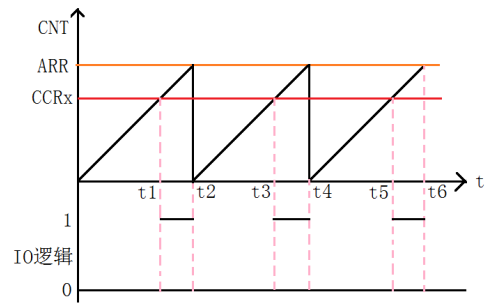


图2-5 占空比示意图

通过编程调节寄存器内部的数据可以实现控制占空比的输出，即PWM波的输出，从而实现呼吸灯。呼吸灯的关键代码见代码2-2。

代码2-2 呼吸灯实现的关键代码

|  |
| --- |
| //使能定时器2时钟和GPIOA时钟  RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2,ENABLE);  RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA,ENABLE);  //初始化IO为复用功能输出  GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5;//选取PA5  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;//设置为复用模式  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz;//设置GPIO口输出频率为100M赫兹  GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;//设置为推挽复用输出  GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;//设置为上拉  GPIO\_Init(GPIOA, &GPIO\_InitStructure);//初始化GPIO使以上设置生效  // PA5复用映射到定时器2  GPIO\_PinAFConfig(GPIOA,GPIO\_PinSource5,GPIO\_AF\_TIM2);//PA5复用为定时器2  TIM\_TimeBaseStructure.TIM\_Prescaler=psc;//定时器分频//其中，psc为传入参数，由自己设置  TIM\_TimeBaseStructure.TIM\_CounterMode=TIM\_CounterMode\_Up;//设置定时器为向上计数模式  TIM\_TimeBaseStructure.TIM\_Period=arr;//设置自动重装载值//其中，arr为传入参数，由自己设置  TIM\_TimeBaseStructure.TIM\_ClockDivision=TIM\_CKD\_DIV1;//设置时钟分割  TIM\_TimeBaseInit(TIM2,&TIM\_TimeBaseStructure);//初始化定时器2使以上设置生效  //初始化输出比较参数  TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM1;//选择定时器模式：TIM脉冲宽度调制模式1  TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable;//比较输出使能  TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_Low;//TIM输出比较极性低  TIM\_OC1Init(TIM2, &TIM\_OCInitStructure);//初始化输出通道  TIM\_OC1PreloadConfig(TIM2, TIM\_OCPreload\_Enable);//使能预装载寄存器  TIM\_ARRPreloadConfig(TIM2,ENABLE);//使能自动冲装载的预装载寄存器允许位  TIM\_Cmd(TIM2, ENABLE);//使能定时器  TIM\_SetCompare1(TIM2,led0pwmval);//不断修改比较值来动态修改占空比 |

2.2.2.3 PWM输出驱动电机实验

此实验的目的在于通过手动代码设置PWM波输出，从而达到控制四个电机正常启动并保持在一定转速的目标。由于涉及到PWM输出，所以我们选择其中一个定时器TIM3，并配置PA6、PA7、PB0、PB1引脚为复用推挽输出模式并复用为TIM3 的四个PWM输出通道。其次，电调事先已经和T4EU-6无线遥控器匹配，所以电调接受的频率为54.27Hz，占空比范围为5.4%~10.9%。而对于STM32F401RE板子的定时器来说，与PWM输出相关的频率fpwm、系统时钟频率fclk、占空比duty、自动装载值arr、分频系数psc、脉冲pulse的关系[3]为（见公式2-1和公式2-2）：

 (2-1)

 (2-2)

系统时钟频率fclk采用系统默认值16MHz，自动装载值arr设置为999方便直观计算，因为PWM输出频率fpwm固定为54.27Hz，所以得出分频系数psc=293。STM32CubeMX软件配置如图2-6、图2-7、图2-8：

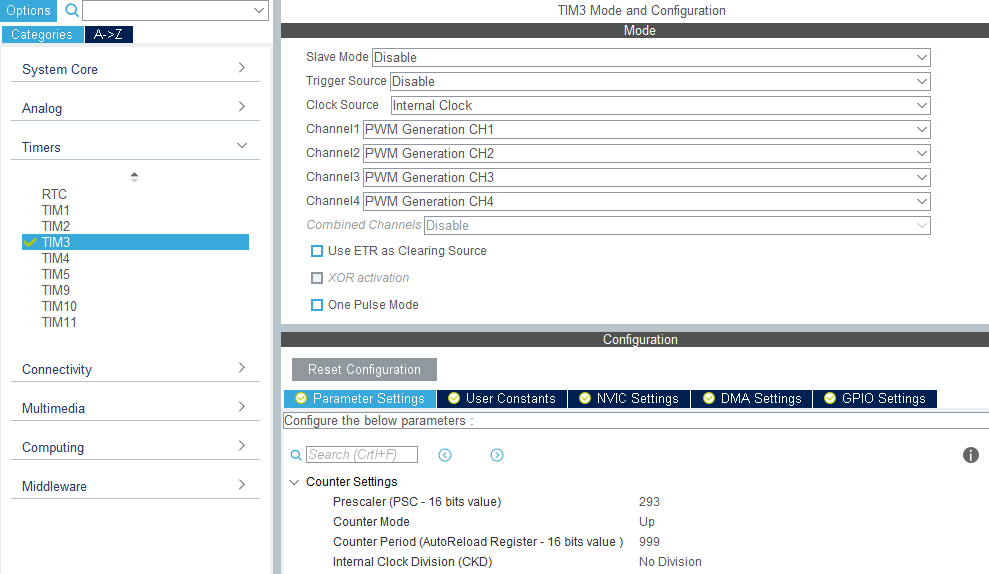


图 2-6 配置界面

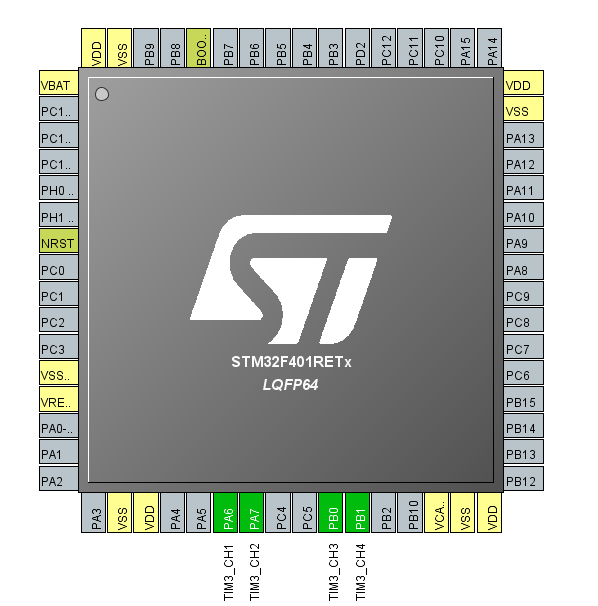


图 2-7 引脚信息

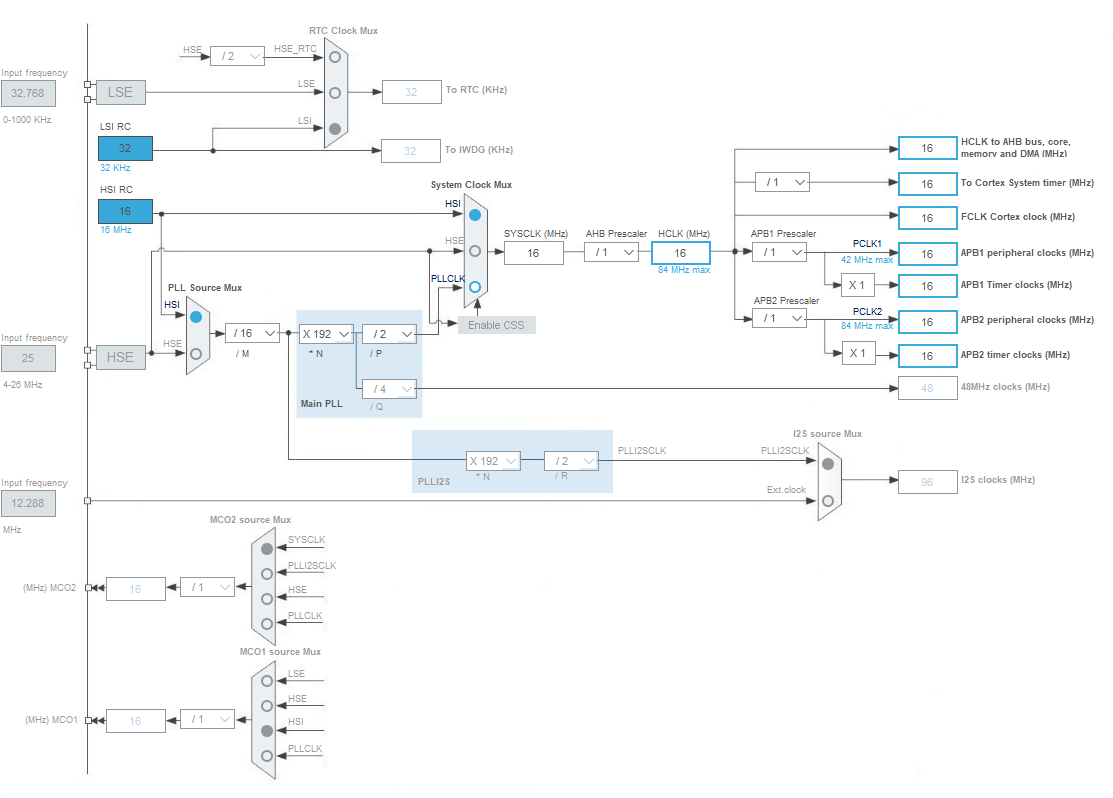


图 2‑8 系统时钟信息

点击生成MDK-ARM V5工程文件。打开工程后便可清晰看出目录树结构，如图2-9所示。

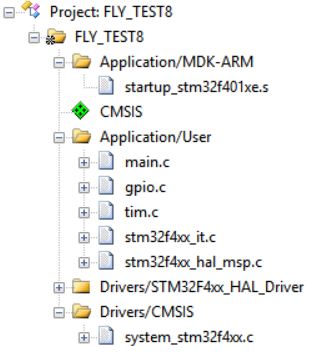


图 2‑9 目录

其中Application/MDK-ARM文件夹包含STM32F401RE的启动文件，Application/User则包含用户的主函数main.c、PA6等引脚的GPIO配置信息gpio.c、TIM3的配置文件tim.c等，Drivers/STM32F4xx\_HAL\_Driver文件夹包含的是HAL库里面对STM32F4系列的驱动文件，Drivers/CMSIS文件夹则包含CMSIS标准的驱动文件。

STM32CubeMX这款软件的优势之处就在于十分直观化的图形操作界面，对STM32板子的各种功能的配置无需自己从无到有一个个地去编写，只需要在图形化操作界面上对个功能模块的基本设置进行选择就好，可以说减轻了很多工程师的负担，从而提高了工作效率。

接下来就是连接硬件电路，将四个电调的信号线分别连接在STM32F401RE的PA6、PA7、PB0、PB1引脚处。

然后就是主函数的编写。

主函数要实现的功能是先开启TIM3，然后设置TIM的输出PWM波的初始占空比为5.4%，紧接着延时5秒左右。这样做的目的就在于电调启动是有一个机制的，而不是说直接通电后就能接受PWM波。电调首先要接受最低占空比的PWM波（这里指5.4%），一直持续大概3秒左右后，电调启动成功，便可以接受任何占空比的PWM波（这里限制占空比范围为5.4%~10.9%）。在主函数的while(1)语句前，先设置电调的占空比为较大值，让电机转起来。然后再while(1)循环内部不做任何事情，保持电机运转转态。主函数的流程图如图2-10。

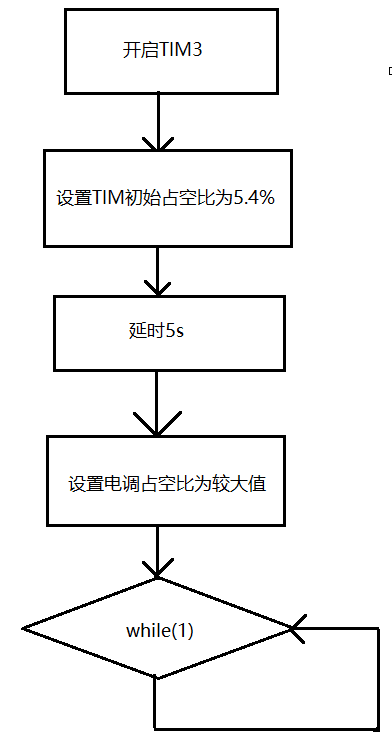


图2-10 主函数的流程图

具体主函数核心代码见代码2-3

4-3 主函数核心代码

|  |
| --- |
| HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_1);//开启TIM3的PWM输出通道1  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_2);//开启TIM3的PWM输出通道2  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_3);//开启TIM3的PWM输出通道3  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_4);//开启TIM3的PWM输出通道4  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_1, lowPulse);//设置通道1初始占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_2, lowPulse);//设置通道2初始占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_3, lowPulse);//设置通道3初始占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_4, lowPulse);//设置通道4初始占空比  HAL\_Delay(5000);//延时5秒来满足电调启动过程  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_1, topPulse);//将电机转速提高  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_2, topPulse);//将电机转速提高  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_3, topPulse);//将电机转速提高  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_4, topPulse);//将电机转速提高  while (1){}//保持电机转速 |

2.2.2.4 PWM输入捕获到输出驱动电机综合实验

此实验的目的在于捕获遥控器发出的PWM波，通过算法得出遥控器PWM的占空比，然后将此占空比赋给四个电调，从而达到一个遥控器遥控四个电机转速的目的。PWM波输出的具体细节前面的PWM波输出驱动电机实验已经做过详细介绍，这里不再赘述。这里主要介绍如何捕获遥控器发出的PWM波并通过算法得到此PWM波的占空比。

首先介绍下遥控器的具体信息。如图2-11、图2-12所示，遥控器红框处的摇杆上推和下拉时遥控器所发出的PWM波将会通过接收器的红框处最上方的引脚输出。



图 2‑11 遥控器摇杆



图 2‑12 遥控器引脚

我们要做的就是将图2-12中接收器红框最上方的引脚连接至STM32F401RE板子上，然后通过PWM输入捕获捕获此PWM信号并通过算法从中得出PWM波的占空比。这里我们选择TIM2作为PWM输入捕获的定时器，关于TIM定时器输入捕获的时钟图如图2-13：

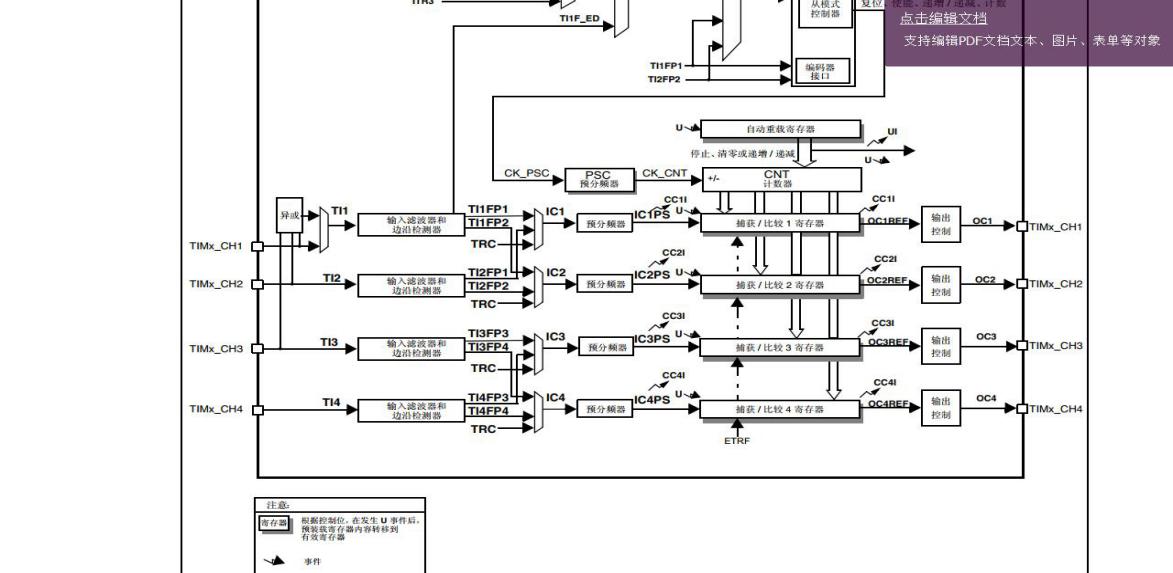


图 2‑13 输入捕获时钟图

如图2-13中TIMx\_CH1为对应引脚，将此引脚连接接收器的信号引脚便可以输入遥控器的PWM波到STM32F401RE板子上。从图4-13中可以看出，PWM信号经过输入滤波器后，可以分出两个一模一样的PWM信号TI1FP1和TI1FP2，这两个信号随后分别经过一个或门后变成IC1和IC2。如果再经过或门时没有其他信号干扰，那么IC1=TI1FP1，IC2=TI1FP2。接下来就是关于如何通过这两个PWM信号IC1、IC2设计输入捕获算法。原理图如图2-14[4]：

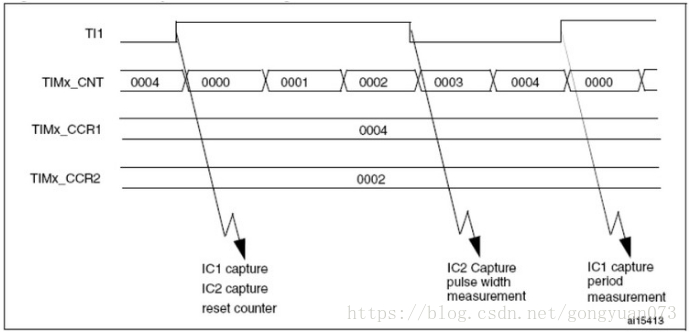


图 2‑14 原理图

在PWM输入信号波出现上升沿的时候，立即重置TIM2定时器的计数器值为0。在紧接着的下降沿时，通过IC2捕获这个信号并将此时的计数器值记录下来（这里记此值为tmp2）。在接下来的上升沿到来时，通过IC1捕获这个信号并将此时的计数器值记录下来（这里记此值为tmp1）。于是很容易得出，tmp2/tmp1的值就是遥控器输入的PWM波的占空比。

知道原理后，接下来就是相关定时器的配置与代码编写问题了。打开STM32CubeMX，图2-15为TIM2配置界面

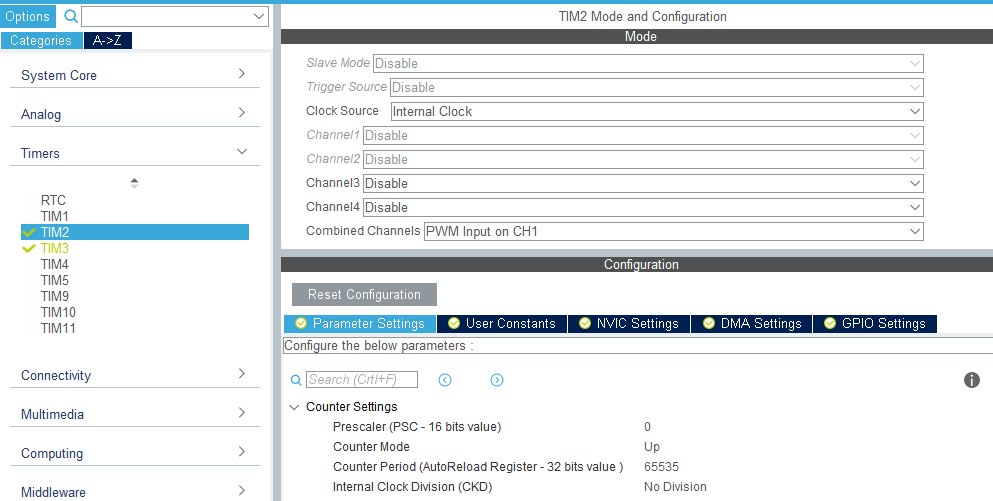


图 2‑15 TIM2配置界面

需要注意的是，TIM2作为PWM捕获用的定时器，需要打开TIM2 global interrupt，因为每遇到一次PWM的上升沿，就将计数器的值重置为0，所以这里必须有一个中断更新事件。具体配置如图2-16、图2-17、图2-18、图2-19：

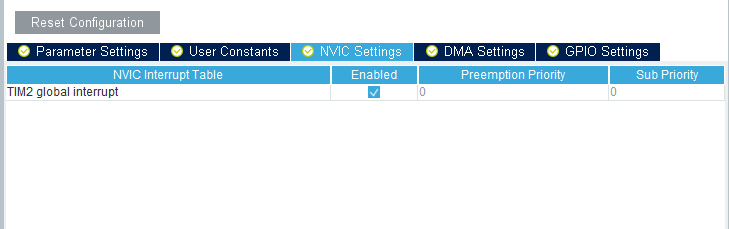


图 2‑16 具体配置界面

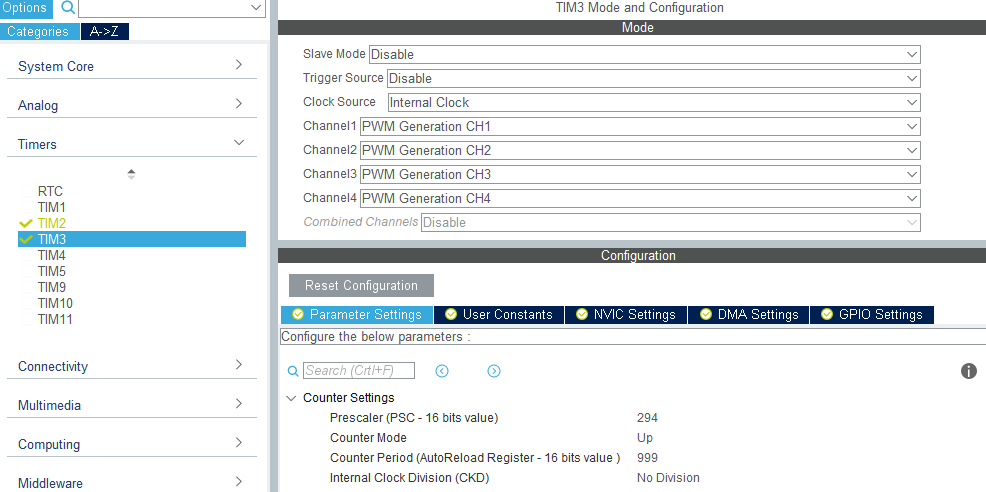


图 2‑17 TIM3配置界面

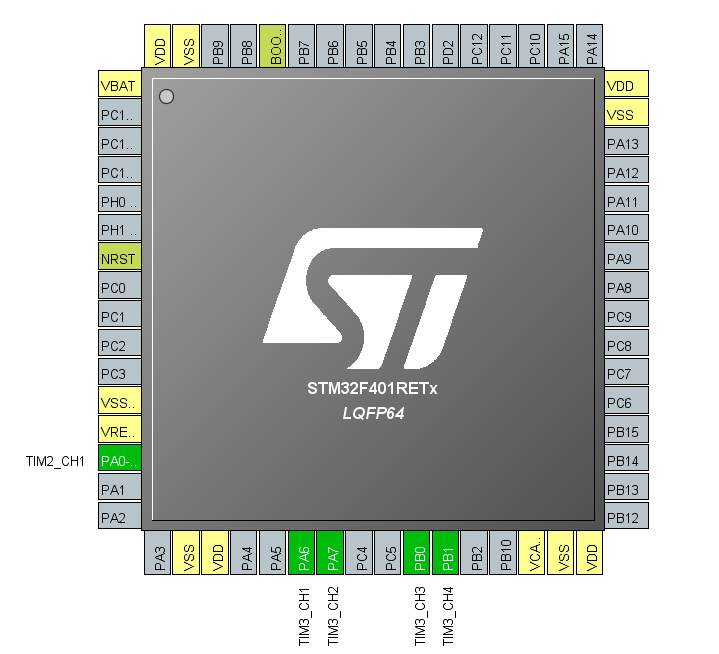


图 2‑18 引脚配置图

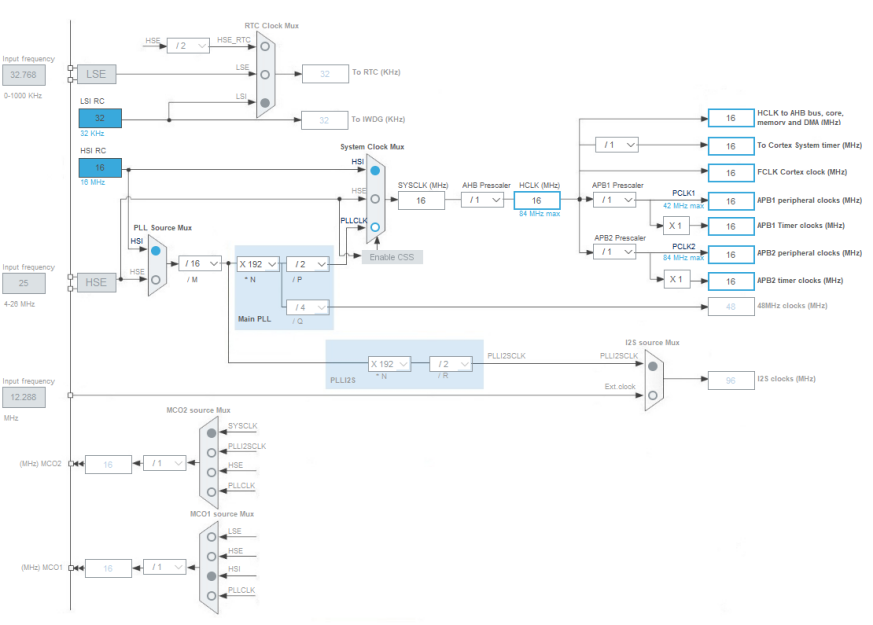


图 2‑19 时钟配置图

点击生成MDK-ARM V5工程文件。打开工程后便可清晰看出目录树结构如图2-20：

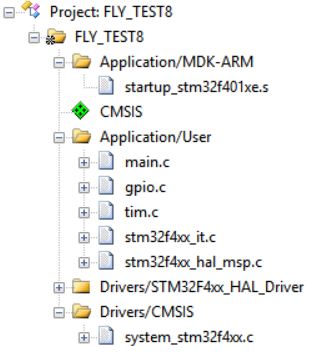


图 2‑20 目录

接下来就是连接硬件电路，将四个电调的信号线分别连接在STM32F401RE的PA6、PA7、PB0、PB1引脚处，并将接收器的信号引脚连接在PA0引脚处，最后将板子上的外部供电引脚连接在图二红框下面两个对应的VCC、GND引脚处从而给接收器供电。

然后就是主程序的编写。

首先要编写的就是一个回调函数用以实时获取捕获的PWM波的占空比（这里称之为duty）并将定时器TIM2的计数器值重置。然后就是主函数的编写。主程序要实现的功能是先开启TIM3、TIM2，然后设置TIM3的输出PWM波的初始占空比为5.4%，紧接着延时5秒左右。在while(1)循环里面，首先停止TIM3，然后将实时捕获到的占空比duty赋给TIM3的四个PWM输出通道，然后再打开TIM3。自此便可实现程序实时读取遥控器PWM波并将之输出到四个电调上从而达到遥控效果。具体核心代码见代码2-4：

代码2-4 主程序核心代码

|  |
| --- |
| int main(void){  ......  HAL\_TIM\_IC\_Start\_IT(&htim2, TIM\_CHANNEL\_1);//开启TIM2通道1  HAL\_TIM\_IC\_Start\_IT(&htim2, TIM\_CHANNEL\_2);//开启TIM2通道2  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_1);//开启TIM3PWM输出通道1  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_2);//开启TIM3PWM输出通道2  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_3);//开启TIM3PWM输出通道3  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_4);//开启TIM3PWM输出通道4  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_1, lowPulse);//设置TIM3通道1初始占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_2, lowPulse);//设置TIM3通道2初始占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_3, lowPulse);//设置TIM3通道3初始占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_4, lowPulse);//设置TIM3通道4初始占空比  HAL\_Delay(5000);//延时5秒来满足电调启动过程  while (1)  {  duty = tmp2 \* 100.0f / tmp1 + 0.5f;//实时得出遥控器PWM波的占空比  HAL\_TIM\_PWM\_Stop(&htim3,TIM\_CHANNEL\_1);//停止TIM3PWM输出通道1  HAL\_TIM\_PWM\_Stop(&htim3,TIM\_CHANNEL\_2);//停止TIM3PWM输出通道2  HAL\_TIM\_PWM\_Stop(&htim3,TIM\_CHANNEL\_3);//停止TIM3PWM输出通道3  HAL\_TIM\_PWM\_Stop(&htim3,TIM\_CHANNEL\_4);//停止TIM3PWM输出通道4  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_1, duty);//设置TIM3通道1PWM的占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_2, duty);//设置TIM3通道2PWM的占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_3, duty);//设置TIM3通道3PWM的占空比  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_4, duty);//设置TIM3通道4PWM的占空比  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_1);//开启TIM3PWM输出通道1  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_2);//开启TIM3PWM输出通道2  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_3);//开启TIM3PWM输出通道3  HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_4);//开启TIM3PWM输出通道4  HAL\_Delay(500);//延时500毫秒  }  ......  }  ........  void HAL\_TIM\_IC\_CaptureCallback(TIM\_HandleTypeDef \*htim)  {  if (htim->Channel == HAL\_TIM\_ACTIVE\_CHANNEL\_1)  {  tmp1 = HAL\_TIM\_ReadCapturedValue(&htim2, TIM\_CHANNEL\_1);//tmp1代表PWM波周期  }  else if (htim->Channel == HAL\_TIM\_ACTIVE\_CHANNEL\_2)  {  tmp2 = HAL\_TIM\_ReadCapturedValue(&htim2, TIM\_CHANNEL\_2);//tmp2代表高电平持续时间  }  } |

2.2.2.5 使用遥控器实现电机旋转以及转速调整

电机的三根杜邦线分别为PWM波接收、VCC、GND，顺序连接至接收器的第三排排针上，再将学生电源连接电机，等待小段时间后接收器和遥控器匹配成功，遥控器上下拨动左摇杆即会控制接收器输出不同占空比的PWM波，当占空比不同时，电机的旋转速度也会不同。

# 第三章 存在的问题与解决方案

## 3.1 绘制转接板中遇到的问题及解决方案

在画转接板时，可能会导致模块无法通过转接板与开发板连接的问题。

所以需要通过查阅开发板官方手册来解决：首先要明确各个引脚的功能，再结合模块的引脚定义，在Altium Designer中绘制出原理图。在绘制原理图时，需要注意模块间的电气连接是否有效。F401RE的官方引脚定义图见图3-1。

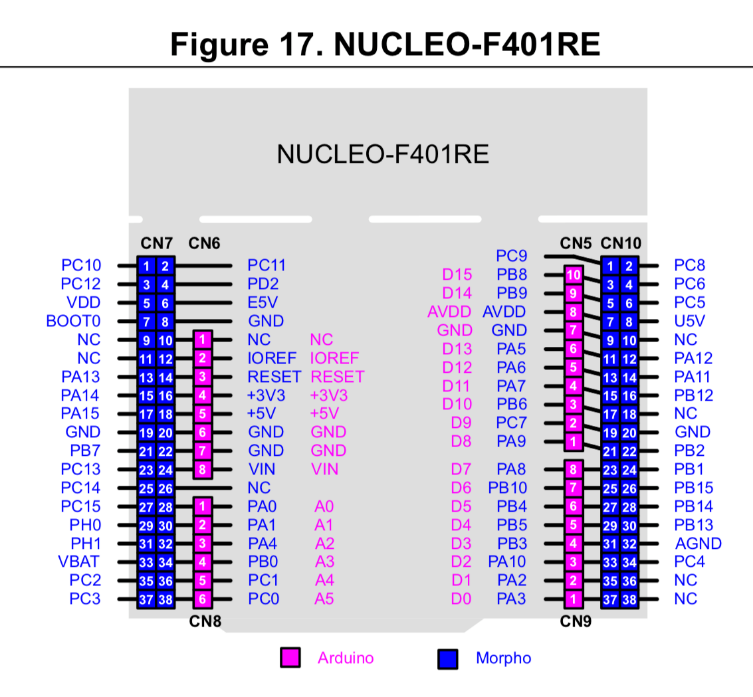


图 3‑1 F401RE官方引脚定义

## 3.2 电机旋转以及转速调整的实现中遇到的问题及解决方案

1、由于初步接触STM32F401，不了解其上的小灯连的是哪个GPIO脚。通过在网上下载STM32F401RET6原理图（如图3-2），查得小灯连的是PA5和PB13。

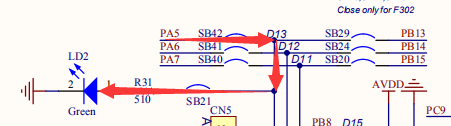


图 3‑2 STM32F401RET6原理图

2、在实现呼吸灯时需要有一定量的前置知识，故通过在网上下载讲解通用定时器基本原理的ppt，获知定时器的组成、工作原理、工作模式。

3、关于复用的理解：GPIO口功能强大，且连接了定时器，可以和定时器相映射。

4、定时器分频：查到网上的资料得出AHB时钟=APB1时钟\*APB1分频系数，溢出中断时间=（ARR（自动重装载值）+1）\*（PSC（预分频系数）+1）/Tclk（CK\_PSC时钟），CK\_CNT时钟=Tclk/（PSC+1）。

# 第四章 总结及任务分工情况

## 4.1 总结

通过一个学期的学习与实际操作，我们完成了四轴飞行器的机械部件搭建和硬件系统设计。按照机械部件搭建要求购买了器材并搭建飞行器骨架，并安装stm32核心板，自己绘制了转接板并将各模块接入stm32核心板。并且通过C语言编写程序，驱动各外设，包括接受遥控器的遥控信号，然后通过这个信号操作四个马达同步旋转，并控制马达转速，实现了使用遥控器实现电机旋转以及转速调整。

## 3.1 任务分工情况

**本小组成员任务分工情况**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 分工 | 完成情况 |
| 1 | 尧松 | 2017220301012 | 转接板绘制、报告撰写 | 完成 |
| 2 | 廖新语 | 2017220302012 | 测试、报告撰写 | 完成 |
| 3 | 杨嵘 | 2017220304011 | 代码编写、报告撰写 | 完成 |
| 4 | 卓子豪 | 2017220305017 | 代码编写、报告撰写 | 完成 |

# 参考文献

1. 刘丽,颜瑾,贾覃溶钊,杜雨欣,郝杰,王晓倩,韩楚赢.四轴旋翼飞行器系统的设计[J].电子世界,2018(22):198.
2. 本\_末.PCB绘制[N/OL].CSDN，2018-01-24.[2019-01-10].

<https://blog.csdn.net/hu_junhua/article/details/79155312>

1. 阿困.详解STM32的PWM输出及频率和脉宽(占空比)的计算[N/OL].CSDN，2014-04-22.[2019-01-10].

<https://blog.csdn.net/akunainiannian/article/details/24316143>

1. gongyuan073.使用stm32的HAL库完成pwm输入模式测量频率和占空比的详细教程[N/OL].CSDN.2018-07-17.[2019-01-10].

<https://blog.csdn.net/gongyuan073/article/details/81080011?tdsourcetag=s_pctim_aiomsg>

# 致谢

本报告的工作是在我们的指导教师廖勇老师的悉心指导下完成的，衷心感谢他在我们的课程设计中给予的悉心指导和鼓励，正是他悉心的指导、严谨的治学风格和孜孜不倦的教诲，给了我们无穷的启发和指引。