

编号: _____



桂林电子科技大学
GUILIN UNIVERSITY OF ELECTRONIC TECHNOLOGY

毕业设计(论文)说明书

题 目: 四轴飞行器的控制

学 院: _____

专 业: _____

学生姓名: _____

学 号: _____

指导教师: _____

职 称: _____

题目类型: 理论研究 实验研究 工程设计 工程技术研究 软件开发

2014 年 5 月 30 日

摘要

四轴飞行器作为低成本的实验平台，在各个领域都有发挥作用的潜力。对比于其他类型的飞行器，四轴飞行器拥有结构简单紧凑，飞行效率高，机动灵活的优点。本文介绍四轴飞行器的设计过程，主要包括结构设计，硬件设计，软件设计三个部分。四轴飞行器的构件设计采用纯板件设计，全机仅使用三种不同的板件、一种型号的尼龙柱和一种型号的螺丝，即可组装成完整的机架实体。为了降低重量和信号连接线的复杂程度，减少 PCB 数量，设计中将导线设计在机架中，所以机架本身同时作为结构部件与 PCB 两种功能。

本设计为了解决传统飞行控制器接线复杂，模块较多的问题，采用了两个新的设计理念：硬件上使用单总线设计理念，软件上为虚拟智能终端。单总线设计理念是指使用一条贯穿全机的 CAN Bus 总线来代替绝大部分的信号线。虚拟智能终端使得终端虚拟化，实现单个硬件实体上运行多个终端的能力。增加了使用灵活度，减少了硬件实体的总数，减轻了编程负担。

本设计使用的传感器包括三轴陀螺仪、三轴加速度计、三轴磁阻传感器（电子罗盘）和大气压力传感器。定位系统包括 GPS，超声波测距模块。无线通讯方面，使用航模 9 通道 RC 遥控器，串口无线透传模块。地面站使用开源项目 QGC(QGroundControl 项目)，并使用开源协议 Mavlink（Micro Air Vehicle Communication Protocol）。姿态算法使用四元数算法，姿态误差修正算法使用互补滤波算法，控制算法为 PID 算法。

关键词：四轴飞行器；姿态计算；四元数；远程控制；Mavlink

Abstract

As a low cost experimental platform, quadcopters are very potential in many application areas. Comparing to other UAVs, quadcoters have many advantages like simple structure, high efficiency and good maneuverability. This essay introduces the processing of designing a quadcopter, including structure designing, hardware designing and software designing. The quadcoter's structure is constituted by 3 different parts, and all these parts are flats. In this design, 3 different parts, 1 type of nylon pillar and 1 type of nylon spike can make up a completed quadcopter. For reducing weight, wire complexity and the number of PCBs of the quadcoter, this designing put the wires in to the structures. So the structures also serve as PCBs.

In order to eliminate the complexity of wiring, two new design philosophies, Single Bus and Virtual Smart Terminal, are used in this design. The Single Bus uses a CAN Bus to replace all of the traditional wire connections. And several Virtual Smart Terminals can share a single hardware terminal. These two philosophies improve the flexibility of the design, reduce the number of hardware terminal, and reduce the difficulty of programming.

A serial of sensors are used in this design, including 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetoresistive sensor and air pressure sensor. Positioning methods are GPS and ultrasonic ranging. Wireless communication methods are a model RC controller and a wireless serial port transceiver. Ground control station uses an open source project QGC and Mavlink protocol. Attitude calculation is based on quaternion and complementary filter. Control algorithm is PID.

Key words: Quadcopter; Attitude Calculation; Quaternion; Remote-Control; Mavlink

目 录

引言.....	1
1 序论	2
1.1 选题背景、目的、意义	2
1.2 国内外发展现状	2
1.2.1 国外发展现状	2
1.2.2 国内发展现状	3
2 系统综述	3
2.1 四轴飞行器的飞行原理	3
2.2 四轴飞行器的设计目标	4
2.3 四轴飞行器的基本工作过程	4
2.4 四轴飞行器的总体构成	4
2.4.1 飞控构成.....	5
2.4.2 电调构成.....	5
2.4.3 电源板构成.....	5
2.4.4 其他飞行支持器件.....	5
2.5 编程语言、编程工具和 RTOS.....	5
2.5.1 编程语言 — C 语言	5
2.5.2 编程环境 — RealView MDK.....	6
2.5.3 实时操作系统 — RT-Thread.....	6
3 可行性参数及结构设计	6
3.1 可行性参数设计	6
3.1.1 指定基本参数	6
3.1.2 电机与螺旋桨选定	7
3.1.3 电池的选型	9
3.2 结构设计	10
3.2.1 电源板与底板设计	11
3.2.2 支臂设计	12
3.2.3 电调板与电机座结构设计	13
3.2.4 主体结构质量估算	14
3.2.5 实物装配图	14
4 硬件电路设计	15
4.1 四轴飞行器单总线（CAN Bus）设计理念	16

4.2 飞行控制器电路设计	17
4.2.1 陀螺仪选型	18
4.2.2 气压计选型	19
4.2.3 其他传感器选型	20
4.2.4 飞控主要元器件列表	21
4.2.5 飞控原理图、PCB 及其说明	21
4.3 无感无刷电子调速器硬件设计	23
4.3.1 无刷电调器件选型及设计原理	23
4.3.2 无刷电调器原理图、PCB 图与实物图	24
4.4 电源底板硬件设计	26
4.4.1 电源底板的设计描述	26
4.4.2 电源底板原理图、PCB 图与实物图	26
4.5 其余电路硬件设计	28
4.5.1 通用拓展板——Universal Expansion Board	28
4.5.2 专用调试器	30
5 软件设计	31
5.1 实时操作系统 RT-Thread 简介	31
5.1.1 飞控程序建立在 RTOS 上的必要性	31
5.1.2 选择 RT-Thread	32
5.2 虚拟智能终端的设计理念	33
5.3 飞控程序设计	35
5.3.1 线程总览与程序结构	35
5.3.2 提取传感器数据	37
5.3.3 姿态算法	39
5.3.4 控制算法	41
5.3.5 输出映射	42
5.3.6 导航算法	43
5.4 电调程序设计	44
5.5 其他模块程序设计	45
5.6 上位机	45
5.6.1 Mavlink 协议简介	45
5.6.2 QGroundControl 上位机简介	45
5.6.3 Deep Blue V0.3 上位机简介	46
6 调试与试飞	47
6.1 传感器的标定与校准	47

6.2 振动测试与滤波器参数选择	47
6.2.1 模拟振动测试平台	47
6.2.2 加速度计的振动测试和滤波器参数选择	48
6.2.3 陀螺仪的振动测试和滤波器参数选择	50
6.3 PID 参数调整	51
6.4 试飞	51
7 总结与展望	52
7.1 总结	52
7.2 展望	52
谢 辞	53
参考文献:	54
附 录	55
1. 飞控原理图	55
2. 无刷电调原理图	56
3. 电源底板原理图	57
4. UBE 原理图	58

引言

小型多轴飞行器因其结构简单，组装方便，已经越来越接近大家的生活。随着 MEMS 传感器、微控制器、电机和电池技术的发展和普及，多轴飞行器已经成为微小型无人机中的新锐力量。直到今天，多轴飞行器已经应用到各个领域，如军事应用、公安追捕、灾害搜救、地理调查、输电线巡查、电影拍摄等。

以汶川地震救灾为例子，传统的调查方式为有人直升机或者固定翼拍摄。限于飞机的数量和飞行高度，当地的能见度，地形等因素，有人飞机不能很好的完成现场拍摄等任务，救灾人员无法实时地分析现场状况。此时需要的是能低空拍摄，灵活调度的微小型无人机，多轴飞行器则是现场最适合的无人应用平台。

目前广泛应用的无人飞行器有固定翼飞机，单轴直升机等。与固定翼相比，多轴飞行器对场地要求低，能实现垂直起降，定点悬停等优点，与直升机相比，有结构简单，提高升力效率的潜力大，成本低等优点。

四轴飞行器是多轴飞行器中最典型的案例，相比其他多轴飞行器，四轴飞行器有简单的结构，易于制造，而在飞行原理与控制方式上，四轴飞行器与其他多轴飞行器没有太大的区别，仅仅在于输出控制量时对单个电机的映射不同。所以，本文就四轴飞行器的设计过程进行阐述，并说明其飞行控制原理。

1 序论

1.1 选题背景、目的、意义

资料显示，随着信息战和网络中心战系统的迅速发展，无人机日益成为战争体系对抗中关键装备之一，在需求推动下，预计今后 10 年内全球对无人机需求总数约达 23000 架，其市场增加到原来的 3 倍，年均开支将由 27 亿美元增加到 83 亿元。未来 10 年共计花费近 550 亿美元。

垂直（VTOL）和短距起降无人机的销售高峰将出现在 2010~2013 年，据悉，法、德、西班牙和英国将会化费 5 亿欧元为本国新型战舰配备这类无人机。美制“火力侦察兵”无人旋翼机预约订货在不断增加。

随着无人机技术的成熟和使用经验的丰富，今后无人机民用领域将不断扩展，防灾救灾、农林作业、缉私缉毒、大地测绘、资源探测、治安反恐、管线监视等各种民用无人机市场需求已出现了明显增长。

四轴飞行器是一种飞行器，同时也是一个飞行的试验平台。当智能设备搭载在四轴飞行器这个灵活的试验平台上时，就拥有了其他形式的智能机器人所没有的优势。例如，困扰地面机器人最大的难题在于如何去适应全部的地形，又例如水上机器人需要面对起伏的波浪和海水的腐蚀，这些问题在飞行平台上，都可以得到完美的解决。但是四轴飞行器作为飞行平台，也有本身的一些缺陷，例如飞行时间不长，容易受风的影响，平台不稳定等问题。

目前在工业发达的国家中，四轴飞行器已经作为一个通用的平台来进行试验和应用了，但国内对四轴飞行器的研究起步较晚，研究力度较少。在国内还没有成熟技术的背景下，对四轴飞行器的控制进行研究，具有重要意义和应用前景。

1.2 国内外发展现状

1.2.1 国外发展现状

国外有不少民间微小型飞控（无人机飞行控制器）项目，包括 PIXHAWK 项目、AutoQuad 项目、APM（ArduPilotMega）项目等，他们都可以控制包括四轴飞行器在内的多种飞行器。不少项目甚至是开源的，为爱好者们学习提供了很多方便。

来自 ETH Zurich 大学 Institute Dynamic Systems and Control 实验室的 Raffaello D'Andrea 与他的团队将四轴飞行器与运动算法相结合，完成了如打球、顶倒立杆、集群飞行和抓取等非常精确的动作和准确的控制效果。

1.2.2 国内发展现状

2008年，在随着德国著名的四轴飞行器飞控 Mikrokopter 的开源，在国内的电子工程师中掀起了一阵 DIY 四轴飞行器的热潮。但因当时 MEMS 传感器较贵，姿态算法有很大的缺陷等一些原因，DIY 热潮很快在 2009 年消散。

2012 年，随着锂电池性能提升，传感器和微控制器价格下降，航空模型得到较大的发展。这时，一些辅助性航模设备，包括固定翼陀螺仪和其他飞行增稳系统等出现并有一定的用户群。DIY 四轴飞行器的门槛变得较低，国内也重新出现了一次 DIY 飞控的热潮。

目前国内对四轴飞行器的研究还处于比较初步的状态，开源的项目较少。比较出名开源项目有匿名四轴飞行器，起始于 2012 年底，现在已经发展出多个版本飞行器硬件和完善的上位机，是一款非常有潜力的四轴飞行器项目。

2 系统综述

2.1 四轴飞行器的飞行原理

四轴飞行器几乎是结构最简单的飞行器，控制上也很容易对其进行分析。四个旋翼分别产生四个垂直方向的力和四个反扭力，当这 8 个力处于平衡状态时，四轴飞行器可以在静止的空气中平稳悬停，当控制其中一个或者多个力共同改变时，四轴飞行器将可以离开平稳状态向所需要的方向进行改变。

但其本身是不稳定的，需要一套飞行控制系统对每一个电机的输出量进行实时调整。这套系统需要做的工作是检测回来姿态，并计算出控制量，并控制四个电机，即可使飞机的受力发生改变。图 2-1 四轴飞行器飞行原理 展现了其悬停时四个电机的转速一致。图 2-2 四轴飞行器逆时针旋转 为四轴飞行器逆时针旋转的例子，1、3 号电机减速，2、4 号电机加速。其余控制情况类似。

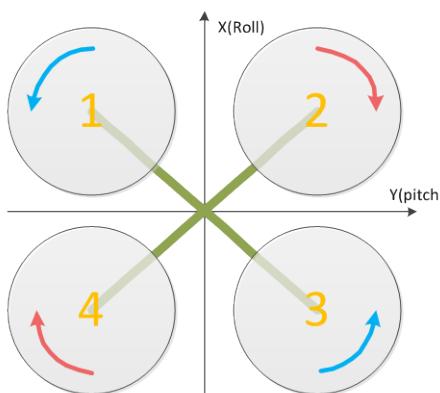


图 2-1 四轴飞行器飞行原理

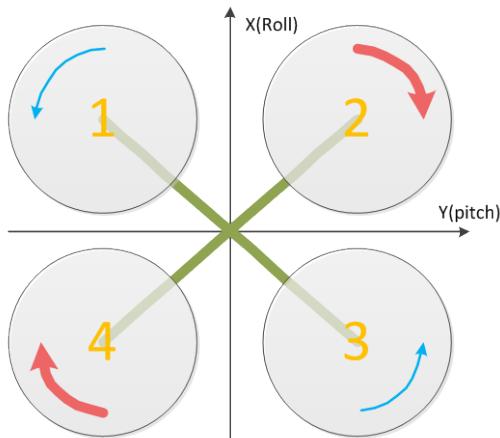


图 2-2 四轴飞行器逆时针旋转

2.2 四轴飞行器的设计目标

设计目标为实现四轴飞行器的控制。系统以 STM32F407VG 系列微控制器作为飞控系统的核心控制器件，采用惯性传感器和绝对位置定位方式，对飞行器的姿态进行检测，并相应地实现自动控制飞行器平衡，定点飞行等功能。

2.3 四轴飞行器的基本工作过程

实现自动控制的主要工作过程分为两部分。第一部分为姿态算法部分，MCU 通过 SPI 总线提取多个不同的惯性传感器件和电子罗盘数据，通过四元数算法和互补滤波器进行融合后进行姿态积分，而后通过四元数转欧拉角矩阵，得出当前的飞行器姿态角度。第二部分为控制算法部分，MCU 利用当前的姿态角度，与期望姿态角度做对比，得到偏差角度，将偏差角输入 PID 控制算法，即可输出三个方向上的修正量。最后，利用三个方向的修正量，通过映射关系，映射到四个电机后输出，即可实现飞行器自动控制飞行。

其他工作，包括飞行器与上位机通信，飞行器的失控保护系统，工作记录等。

2.4 四轴飞行器的总体构成

四轴飞行器硬件主要由三种电路板组成，分别为一块飞控板（无人机飞行控制器），一块电源管理板，四块无刷电子调速器。此外还有一个串口数传模块和若干个通过通用拓展板拓展的传感器与接收机模块。

2.4.1 飞控构成

飞控（飞行控制器）是整个控制系统的中心，它负责计算当前飞行器的姿态，并输出控制量。MCU 使用 Cortex-M4F 内核的 STM32F407VG 微控制器，操作系统使用 RT-Thread 1.2。飞控板拥有较多的感知能力，传感器包括三轴陀螺仪、三轴加速度计、三轴磁阻传感器（电子罗盘）和大气压力传感器。另外还有 TF 卡插槽，mini-USB 接口，2 路 UART 接口（其中一路专用于调试），1 路 PPM 信号输入端口与 12 路舵机兼容 PWM 输出接口。此外它内置了 GPS 模块，可以独立实现单板导航控制功能。

2.4.2 电调构成

电调（无感无刷电子调速器）主要功能是负责驱动三相无感无刷电机，并通过 CAN Bus 实现电压监测，电流反馈功能。MCU 使用 Cortex-M3 内核的 STM32F103C8T6 微控制器，与飞控相同地，软件上基于 RT-Thread 1.2 操作系统。电调包括 1 路 UART 接口，1 路 SWD 调试口，1 路 CAN Bus 总线接口，1 路 PWM 输入接口。飞控可以选择通过 PWM 输入引脚控制电调板，或者 CAN Bus 总线控制电调板，仅取决于接线方式。

2.4.3 电源板构成

电源板主要功能是负责电池电压（11.1V~12.6V）到 5V 的转化，内置一个 3A 的 DC/DC 转化电路。电源板与电调相同，内置 STM32F103C8T6 微控制器并基于 RT-Thread 1.2 操作系统，使电源管理独自形成一个智能的模块。电源管理板负责监控最多 5 路单节锂电池电压(3.7~4.2V)与 1 路整块锂电池电压(最高 21V)监控，同时提供电池电流监控。当电流或者电压接近危险值时，通过自身的 CAN Bus 接口，向飞控和四个电调发出电源危险警告，并通过蜂鸣器发出声音警报，使四轴飞行器尽快着陆以保护锂电池和飞行器。

2.4.4 其他飞行支持器件

包括动力锂电池、电机、螺旋桨等。

2.5 编程语言、编程工具和 RTOS

2.5.1 编程语言 — C 语言

系统硬件使用的各个微控制器均为 32 位的 Cortex-M 内核的芯片，计算性能与硬件资源远远超过 8 位，16 位单片机。为了保证算法与代码兼容性和可读性，本设计完全使用 C 语言进行编程。现代编译器中，已经能很好地优化代码结构，使高级语言编译出来的代码接近汇编语言的效率，同时 C 语言的可读性是汇编所不及的。C 语言是除了汇编

语言以外最接近底层的编程语言，所以用 C 语言也能很好地解决硬件问题，并提高程序的运行效率，可以在贴近硬件的同时，也能使用更高级的编程思维。

2.5.2 编程环境 — RealView MDK

编写代码及编译过程，均是使用 RealView MDK 5 来进行的。MDK 5 是一个相当优秀的 IDE (Integrated Development Environment)。Keil 公司开发的 ARM 开发工具 MDK，是用来开发基于 ARM 核的系列微控制器的嵌入式应用程序。它适合不同层次的开发者使用，包括专业的应用程序开发工程师和嵌入式软件开发的入门者。MDK 包含了工业标准的 C 编译器、宏汇编器、调试器、实时内核等组件，支持所有基于 ARM 的设备，能帮助工程师按照计划完成项目。

2.5.3 实时操作系统 — RT-Thread

RT-Thread 是一款由中国开源社区主导开发的开源嵌入式实时操作系统（遵循 GPLv2 许可协议），它包含实时嵌入式系统相关的各个组件：实时操作系统内核，TCP/IP 协议栈、文件系统、libc 接口、图形界面等。

四轴飞行器上的每一个微控制器中的软件，实际上都是基于 RT-Thread 1.2 的一个或者多个线程。RT-Thread 1.2 是一个国产开源实时操作系统(RTOS: Real Time Operating System)，采用 C 语言面向对象的编程思维，非常巧妙地兼顾了 C 语言的底层编程特点和高级语言的面向对象的编程思想。它优秀的可裁剪性和丰富的组件，使四轴飞行器的软件设计和硬件设计过程变得简单且利于维护，并提高了代码的复用性。

3 可行性参数及结构设计

四轴飞行器是一个复杂实体，包含非常多的结构零件及参数设计。本文将从简入手计算主要的关键参数，包括几个主要部分，电机选择、螺旋桨选择、电池选择、机架尺寸设计等几个主要参数。

3.1 可行性参数设计

3.1.1 指定基本参数

根据资料与一般经验，作者确定出以下几个部分的参数，利于个人独立制作与调试，并能兼顾一定载荷与续航时间。

四轴飞行器基本参数如表 3-1：

表 3-1 四轴飞行器基本参数

参数名称	值	单位
空机质量（不含电池）	350	g
机架对角线长度	350	mm
续航时间	>10	分钟
载荷	>200	g

根据表 3-1 中的几个基本设计参数，可以完成大部分的参数选定。

3.1.2 电机与螺旋桨选定

航模无刷电机通常是指三相无感无刷交流电机。相比于有刷电机，无刷电机少了电刷，磨损主要是在轴承上了。从机械角度看，无刷电机几乎是不需要进行维护的。无感无刷电机的缺点是它的启动性能相比与有刷电机较差，启动时扭矩不足，驱动器在低转速时不能很好定位转子的极性。但当其用于航空模型时，这种缺点显得不是很严重，因为螺旋桨在启动时是气流阻力较小，并且螺旋桨质量较轻转动惯量较小，电机启动负载非常小。此外无感无刷电机拥有非常高的能量转换效率，同时体积较小，这大大延长了飞机的留空时间。所以无感无刷电机在航模上的应用已经代替了有刷电机，成为航模中最常见的电动机。

本设计中的四轴飞行器，它所需求的升力、留空时间等参数和使用条件，与航模使用的条件是类似的，所以作者选择使用航模无感无刷电机作为四轴飞行器的电机。

航模无感无刷电机分为两类，一类是内转子电机，一类是外转子电机。通常内转子电机转子有 2 个磁极，定子有 3 个或者更多磁极。相比之下，外转子电机转子磁极通常为 10 个以上，定子磁极也通常为 10 个以上。所以内转子的相对于外转子电机更容易达到高的转速，而外转子电机通常有比较高的扭矩。

对于固定桨距螺旋桨在桨尖线速度不超过 0.7 倍音速时，有如下关系式：

$$T = Ct \cdot \rho \cdot n^2 \cdot D^4$$

$$P = Cp \cdot \rho \cdot n^3 \cdot D^5$$

式中：T—拉力；Ct—拉力系数；Cp—功率系数；ρ—空气密度；n—螺旋桨转速；D—螺旋桨直径；

根据关系式，可得出，一般情况下，桨尖线速度不超过音速的 0.7 倍时，桨直径增大拉力随之增大，效率随之提高，需求的扭矩也越大。所以，本设计中选用外转子电机，使其驱动大直径低螺距的螺旋桨，以提高力效率（推力/功率），使四轴飞行器拥有较长的续航时间。

航模无感无刷电机型号一般使用 4 位数字表示，例如 2212，前二位表示电机定子的直径,22mm，后二位表示电机定子的厚度，12mm。一般而言，定子直径越大扭力越大，KV 值越低（KV 值定义：输入电压增加 1 伏特，无刷电机空转转速增加的转速值），厚度越厚，额定功率越大。

航模定距螺旋桨型号一般用 4 位数字表示，前两位代表直径，后两位为螺距，但因直径不同略有区别。10 英寸以下的螺旋桨，例如 8043，80 代表直径为 8 英寸，43 代表桨距为 4.3 英寸。10 寸或以上螺旋桨例如 1045，10 代表 10 直径为 10 英寸，45 代表桨距为 4.5 英寸。

一般而言，四轴飞行器的所有电机最大静推力，应该为其悬停时的 1.5 倍以上，才能保证单个电机有余量来维持飞行器稳定飞行。根据设计空机质量、载荷等参数，并假设电池质量为 200g 以内时，可以方便地计算出所需的推力为 1125g 左右，所以单个电机最大升力需要在 281g 左右。

综合电机重量，功率等因素考虑，最终选择一款 D2206 电机，它的 KV 值为 1200，转子磁极数为 14，定子极数为 12，与其配套的螺旋桨为 1045 螺旋桨。测试条件为 3 节锂电池串联，电压 12.6V 至 11.1V。因厂家未给出 KV 值为 1200 这个型号的测试数据，所以作者对其进行推与电流测试。推力测试方法如下：使用电子称将电机垫高，并保持电机水平，然后将电子称校准，螺旋桨向上吹风，将电子称往下压，螺旋桨的推力即是电子称显示的压力。电流测试使用一个可以测直流电的钳形电流表，因为其他电路所需电流较小，可以忽略，所以直接测试电池的输出电流。如图 3-1 电机电流与螺旋桨推力测试。

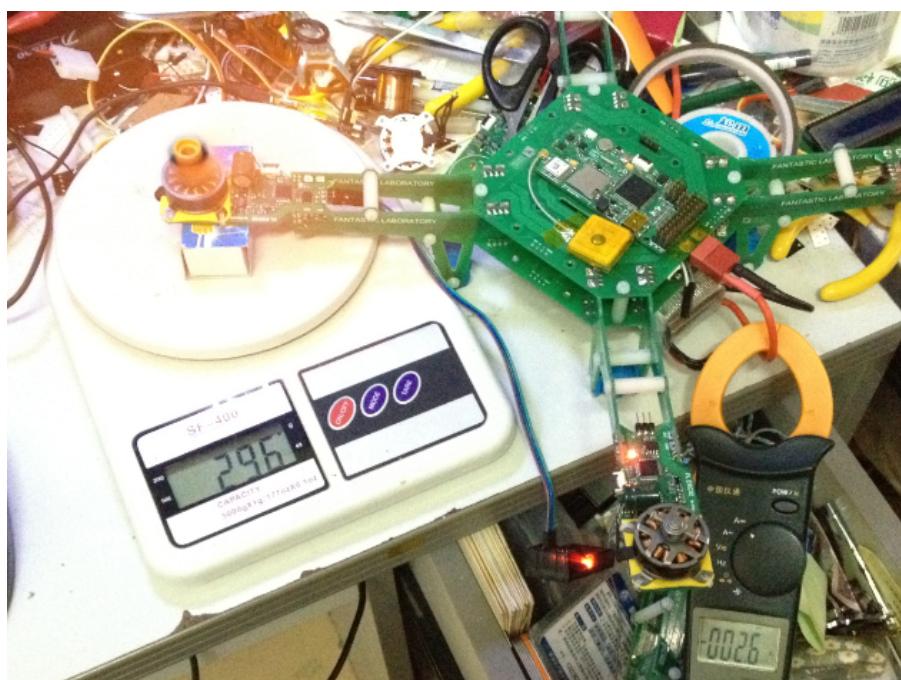


图 3-1 电机电流与螺旋桨推力测试

根据测试结果图 3-1，得知，电机的最大推力为 296g，此时电流为 2.6A。可以轻易算出，最大功率时力效率（推力除以输入功率的比值，单位 g/w）为 10.25g/w。

3.1.3 电池的选型

对于锂电池来说，一般应用关心两个参数，电压与容量。但在这里锂电池需要再关注另一个参数，放电倍数，即电池放电 C 率，1C,2C,0.2C 是电池放电速率：表示放电快慢的一种量度。C 率等于电池的最大放电电流除以电池容量。无疑 C 率越高，锂电池的放电性能越好，电量也可以更快地从电池中放出。C 率一般还与电池的内阻相关，内阻越小，C 率越大。C 数在航模动力锂电池里面是一个重要的指标，普通应用中，放电电流较小，电池内阻引起的电压掉落较少，但用于航模中的锂电池，一般要求在 6 至 12 分钟内放完电量，此时 C 率约为 10 至 5。由电池内阻引起的电压下降将会比较大，电池本身因为放电引起的热损耗会增加，然而锂电池的特性使得它本身在高温下工作将严重减少它的寿命。

所以，在对锂电池选型时，应当优先考虑电池的 C 率。一般而言，航模使用的动力锂电池，都是聚合物锂离子电池，而且均经过特殊的改进，比如使用三元材料制作，增加电极面积等，使得航模用的动力锂电池拥有非常高的 C 率，根据不同需求，可达 10 至 40C。

根据电机与螺旋桨的测试结果，需要 3 个单体串联的锂电池（11.1~12.6V）才能达到所需升力的功率需求。3 串联锂电池中，常见的型号有 2200mAh, 3300mAh, 4000mAh 等。更大的容量意味着更长的工作时间、更大的放电电流（C 率相同时）与更重的重量。

当四轴飞行器工作于悬停状态时，以总重量 750g(350g 机架+200g 电池+200g 载荷) 来计算，根据前面计算出来的力效率，很容易就可以算出，四个电机的悬停时总工作电流不大于 7.5A。如果选用 2200mAh 的电池，留空时间为 $2.2/7.5 = 0.26$ 小时，约为 16 分钟，大于设计要求的 10 分钟。此时需求的平均 C 率为 $7.5A/2.2Ah = 3.4C$ 。最大瞬时电流为 $2.6A * 4 / 2.2Ah = 4.7C$ 。

所以，只需要选择 11.1V 2200mAh 10C 或更高 C 率的锂电池。经过筛选，设计使用口碑较好的花牌 11.1V 2200mAh 25C 锂聚合物电池，电池照片如图 3-2。



图 3-2 电池选型与重量

3.2 结构设计

结构设计是四轴飞行器的设计中的重要环节。本设计中的四轴飞行器，相对于其他产品在结构设计中，有下列几个创新点。

1. 主要构件即是 PCB 电路板，除了飞控本身独立一块 PCB 外，其他构件充当智终端的作用，包括传感与伺服终端。
2. 纯板件零件设计，利于加工和减少加工成本。
3. 零件数量少，只有 3 个不同的平板构件和 1 个型号的尼龙柱，1 个型号的尼龙螺丝钉。
4. 模块化设计，可以灵活更换部件。
5. 机身结构形成自锁状态，减少零件间的最大接触应力。

所有的板件采用 1.6mm 的 PCB 板件，由 PCB 打样公司制板并加工成品。组装成晶图 3-3 四轴飞行器主体组装图所示：



图 3-3 四轴飞行器主体组装图

3.2.1 电源板与底板设计

底板需要兼顾锁紧支臂，提供电池捆绑点，固定飞控，固定拓展模块等功能，所以将其外形设计如下。设计中尽量采用对称设计。四轴主体结构的固定方式为上下两层板夹紧支臂模块。四个对角线上设计四对插槽，用于固定四轴的四个支臂模块。一个四轴飞行器中，共使用两块底板，其中一块上焊接电子元件与微控制器，兼做电源智能终端，用于电源监控，电压变换等功能（此部分功能在硬件设计章节描述，这里只阐述外形设计），另一块底板不焊接电子元件，单纯作为结构部件。因为插槽使用对称设计，所以使用同一块底板设计即可使用两次。图 3-4 底板构件设计图显示了底板的外形设计。

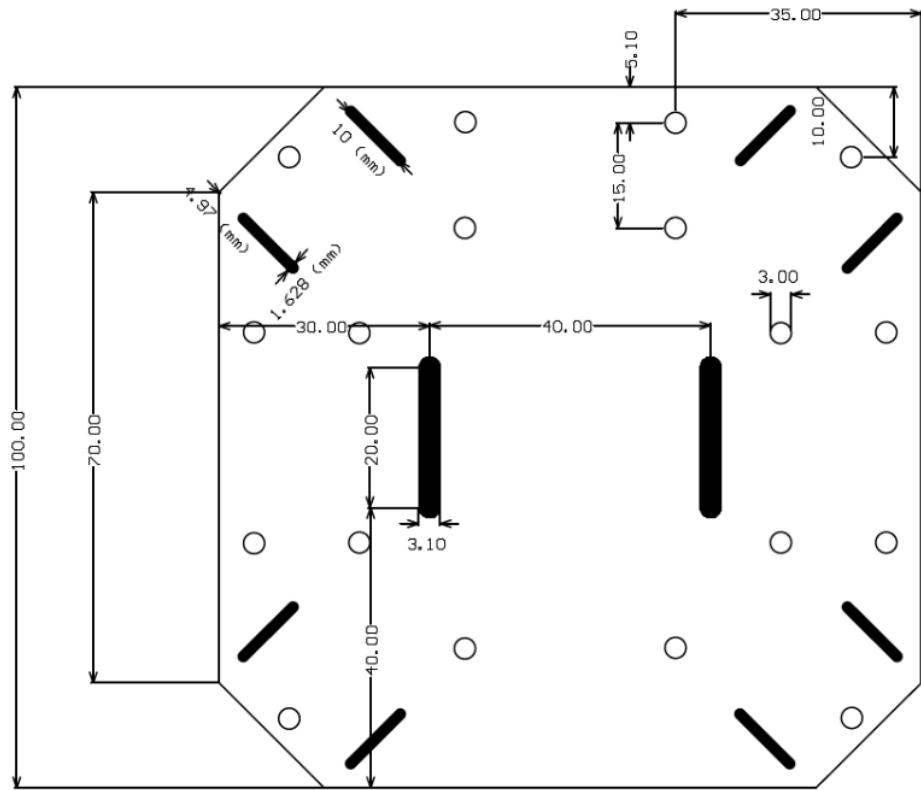


图 3-4 底板构件设计图

3.2.2 支臂设计

支臂的设计如：图 3-5 支臂构件外形设计图。

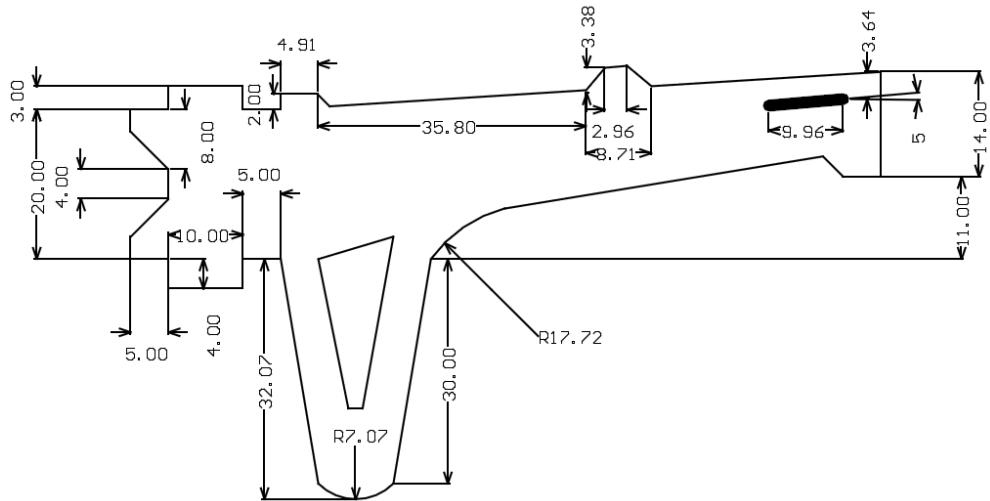


图 3-5 支臂构件外形设计图

支臂的设计要求轻便稳固，因为在四轴飞行器中，支臂零件将需要同时使用八个（每一个电机座需要两个支臂来固定）。如果能在支臂上做优化设计，将重量与强度设计到

最优，就能使重量大大减轻，这意味着四轴飞行器可以搭载更重的载荷，或是在载荷相同的情况下，延迟续航时间。

支臂同时兼顾起落架的功能，下方伸出 3.2 厘米的长度，这样可以把挂载在底板上的锂电池支撑离开地面，使其得到保护。起落架还能使螺旋桨离地面的高度增加，减少起飞着陆过程中，螺旋桨的地面效应对四轴飞行器稳定性的负面影响。

支臂设计的电机座安装插槽设计为有 5 度的斜度，其作用类似固定翼飞机机翼的上反角，使螺旋桨产生的力方向微微向四轴飞行器中心轴正上方靠拢，这样可以产生一个趋于稳定的自稳效果。

如图 3-6 上反角（Dihedral）示意图，上反角（Dihedral）可以让固定翼飞机产生自稳效果。

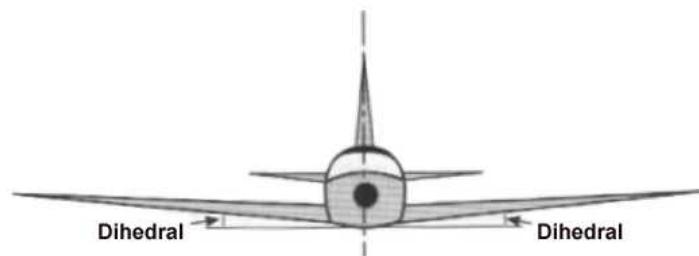


图 3-6 上反角（Dihedral）示意图

3.2.3 电调板与电机座结构设计

电调板如下图：

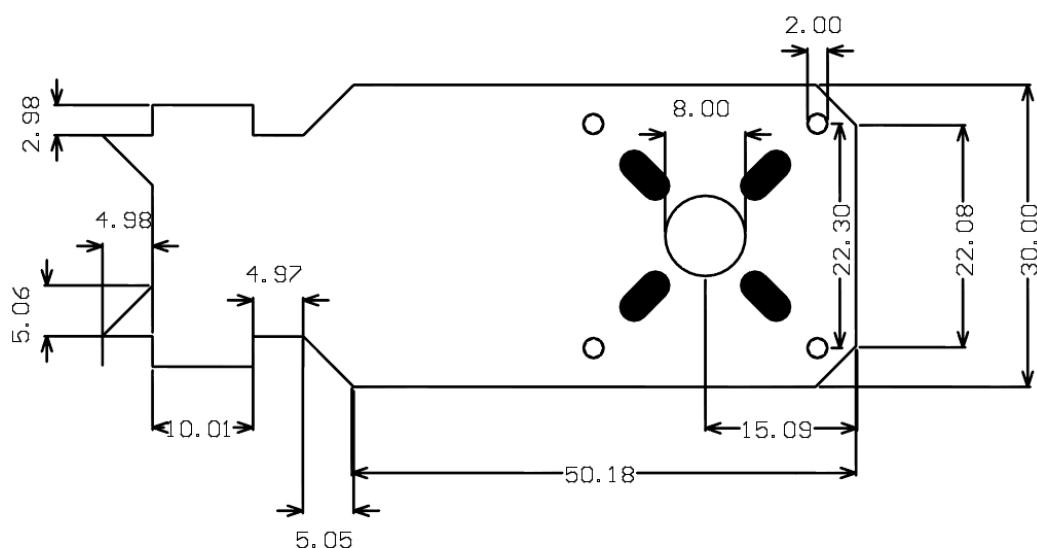


图 3-7 电调板与电机座外形设计图

电机座与电源底板类似，兼做电机驱动与反馈智能终端，电子部分将在硬件设计章节详述。外形上，同样需要遵循简单对称的设计思路，两块支臂各通过一个插槽固定电机座，电机座上设计了多种电机安装插槽。可以兼容 2204 电机至 2216 电机。

3.2.4 主体结构质量估算

通过以上设计，可以估计各个板件的面积。并与制板厂家咨询后，得知 1 平方分米 1.6mm 厚度的 PCB，质量为 30~35g，实际值取决于 PCB 上的铜面积。我们以 32 克/平方分米估算。如表 3-2 的估计，可以得到结构部分的重量估计值约为 135g。

表 3-2 机架质量统计

名称	估计面积（平方分米）	数量（个）	质量（克）
电源板与底板	0.95	2	60.8
支架	0.2	8	51.2
电调板与电机座	0.18	4	23.04
总计			135.04

可得构件部分设计重量为 135g 左右，根据图 3-3 实际测量为 130g。

3.2.5 实物装配图

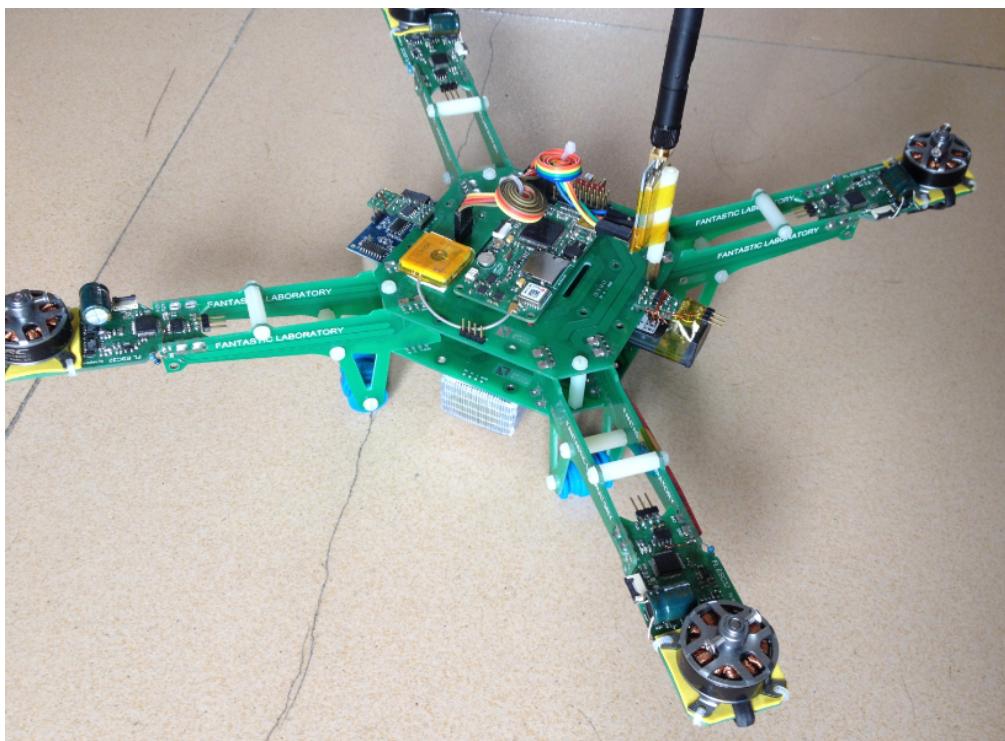


图 3-8 四轴飞行器实物装配图

图 3-8 四轴飞行器实物装配图为四轴飞行器装配完成的照片，可以看到机械部分整个飞行器只使用了 3 个不同的结构构件。实际使用中，四轴飞行器的机架坚固轻便，大大延长了留空时间。

4 硬件电路设计

四轴飞行器的硬件电路分为几大模块：

1. 飞行控制器模块。
2. 无刷电子调速器模块。
3. 电源底板模块。
4. 通用拓展板模块。
5. 专用调试器模块。

以上各个模块组成四轴飞行器的所有电路功能。各个独立的模块都拥有自己的职能和地位，只有它们相互协调工作，才能保证四轴飞行器的稳定飞行。

最重要的是飞行控制器模块，它负责管理所有飞行相关的主要的工作，是四轴飞行器的神经中枢和大脑。无刷电子调速器模块负责驱动独立的无感无刷电机，并反馈电机相关数据，是四轴飞行器的四肢和神经末梢。电源底板模块，负责将电池输出的高电压转化为 5V 电压，供给四轴飞行器上的所有电子设备，并完成电源监控和管理相关工作，是四轴飞行器的心脏。

拓展板则作为四轴飞行器的通用感知、伺服转化板，提供多种丰富的接口与非常小的体积。而专用调试器，则是由一个 USB-HUB 芯片挂载了 Jlink-OB 与 USB 转 UART 芯片组成的联合调试器，通过自定的 8Pin 调试口，可以调试四轴飞行器上各个模块中的 STM32 微控制器。

四轴飞行器电路模块结构如图 4-1。

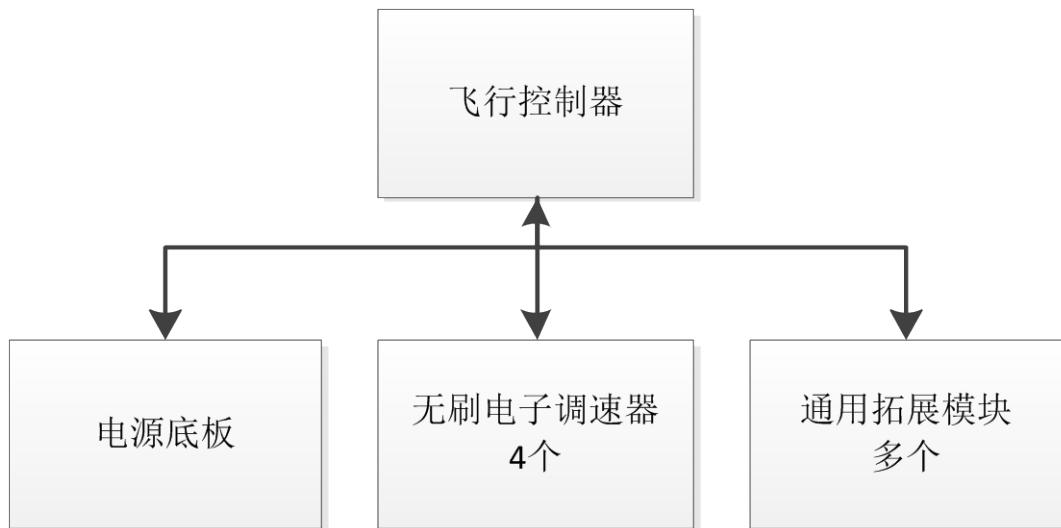


图 4-1 四轴飞行器电路结构图

4.1 四轴飞行器单总线（CAN Bus）设计理念

值得注意的是所有的模块之间的数据交换，是通过一条 CAN Bus 总线来进行的。虽然对于某些模块来说，数据流经过转换，看起来像是走了弯路，但单一的总线代替原来的各种各样的信号线是利大于弊的，下面将具体分析传统接线方式与 CAN Bus 的区别。

CAN 是 Controller Area Network 的缩写（以下称为 CAN），是 ISO 国际标准化的串行通信协议。在汽车产业中，出于对安全性、舒适性、方便性、低公害、低成本的要求，各种各样的电子控制系统被开发了出来。由于这些系统之间通信所用的数据类型及对可靠性的要求不尽相同，由多条总线构成的情况很多，线束的数量也随之增加。为适应“减少线束的数量”、“通过多个 LAN，进行大量数据的高速通信”的需要，1986 年德国电气商博世公司开发出面向汽车的 CAN 通信协议。此后，CAN 通过 ISO11898 及 ISO11519 进行了标准化，在欧洲已是汽车网络的标准协议。

通过下面的表 4-1，可以清楚看到 CAN Bus 与传统多根信号线在四轴飞行器的应用中的区别。

表 4-1 多重信号线与 CAN Bus 的对比

对比项目	多重信号线	CAN Bus
线材数量	随模块数量增加而增加	2 条信号线+2 条电源线
线材种类	各种各样	双绞线
抗干扰能力	参差不齐	好
传送距离	近	远（达 40 米）
模块之间数据交换	需要交换机	模块直接互相通信无障碍
数据传输实时性	参差不齐	非常好（报文长度固定，有优先级）
增加模块难度	需要修改硬件增加接口	直接挂载在总线
编程难度	不同接口难度不同	可以使用统一解码函数
布线难度	大	小
传输速度	不同	最高 1M bit/s
校验	参差不齐	硬件 CRC 校验及自动重发机制

通过表 4-1，容易看出 CAN Bus 总线在四轴飞行器的便利。在硬件设计时，应用了 CAN Bus 相对而言减少了布线难度，而软件上，使用了 CAN Bus 后，令解码函数统一了，从而减少了软件的复杂程度。

每一个 CAN Bus 报文，都是明文传送，所以每一个模块，都可以不通过主机，而直接通过报文 ID 接收其他模块发出的报文，间接减少了主机的处理压力。例如当电源板发送低电压警报到总线时，电调板可以通过 CAN Bus 总线中的报文 ID，确定出此次报文是属于电源模板发出的紧急信息，并做相应的处理，例如闪动电调板上的 LED 等发

出警告，或者在电量处于危险程度时，强制降低电机功率使四轴飞行器缓降避免更大的损失。相反地，电源板可以监听 CAN Bus 中电调板发送的电压电流反馈信息，如果发现超负荷运转等危机情况，利用电源板内置的蜂鸣器发出声音报警。以上两个例子，均不需要飞行控制器本身参与，其反应过程，类似于脊椎动物的脊柱产生的本能反应。

4.2 飞行控制器电路设计

飞行控制器（以下简称飞控）作为本设计中核心的模块。为了简化整机设计，对飞控有几个最基本的要求：第一，集成度高，以尽量减少外置模块的数量；第二，能够独立完成控制及导航任务；第三，体积小质量轻，对于四轴飞行器来说，每少一点重量，都是对续航时间或者载荷能力的提升。

飞控需要非常强大的处理能力，在飞行时，滤波、姿态与控制算法需要非常多的浮点计算。多路输出、输入接口则需要控制器拥有多种多样的外设。另外，飞控程序本身的多任务特性，则要求控制器有足够大的内存和最小的中断时间。如使用传统 8 位、16 位单片机，很难完成如上任务，在后续章节中将对飞控中重要线程与其占用资源做描述。

综合上述因素，本设计中采用 ST 公司生产的 Cortex-M4F 内核控制器 STM32F407VGT6。它拥有最高 168MHz 的运行频率，FPU 浮点运算单元，192K Byte RAM，1M Byte Flash，其中包括 64k 单周期访问的 CCM 内核专用储存区。此外还有各种丰富的外设，包括多个 ADC，SPI，SDIO，UART，CAN Bus，DMA，USB 等本设计必须的外设。经过与其他型号的 Cortex-M 内核控制器进行比较，该型号控制器非常适合本设计所需的要求，第一，相对较高的运行频率、额外的 DSP 指令集、FPU 浮点单元和 CCM 内存等可以提供足够的计算能力，是各个复杂算法的基础。第二，丰富的外设使得飞控 PCB 设计变得简单，减少外部 IC 的数量，并能一定程度上减轻 CPU 的工作压力。第三，Cortex-M 内核特有的 NVIC，相比其他内核而言有更快地中断相应速度和更丰富的配置能力，能尽快地完成切换任务工作，。

传感方面，飞控中集成了 10 个自由度的传感器，包括三轴陀螺仪（MAX21000）、三轴加速度传感器（ADXL362）、三轴磁阻传感器/电子罗盘（HMC5983）和大气压力传感器（MS5611）。有了这 10 个自由度的传感器，即可计算出当前四轴飞行器的姿态角度、飞行器朝向和海拔高度等。

飞控中集成了超小封装的 GPS 模块（Ublox Max 7Q），在提供 GPS 定位能力的情况下，尽量减轻了模块本身的质量。这个 GPS 模块提供高达 10Hz 的定位信息输出，相比普通 GPS 模块仅有的 1Hz 输出而言，拥有更好的动态定位性能。

电源部分，为了提高效率，降低能耗和发热量，采用了 DC/DC 芯片（LT1616）从外部输入的 5V 电压降压到 3.3V 给微控制器、GPS 模块和传感器供电。

飞控有不错的输入输出能力。特别地，由于 CAN Bus 总线的引入，减少了四轴飞行器的接线负担，使飞控的在减少实际引线的情况下，提高了输入输出能力。飞控包含以下接口：1 路 PPM 信号输入，12 路 PWM 舵机控制信号输出，1 路 UART 接口，1 路 CAN Bus 总线接口，1 路 USB 接口，1 路专用 8 Pin 调试口(包括 1 路 UART 与一路 SWD 调试接口)。

飞控板设计有一个 microSD 卡插槽，此卡通过 STM32F407VGT6 的 SPI 总线访问，用于记录详细的调试数据与飞行数据。

4.2.1 陀螺仪选型

陀螺仪是所有传感器中的重中之重，陀螺仪的精度与带宽直接决定了飞控能否准确地检测出飞机当前的姿态。飞控上使用的陀螺仪是 MEMS 工艺制造的，这类陀螺仪利用科里奥利力来检测旋转的角速度。飞控上只需要将旋转的角速度带入姿态算法，即可积分出当前的姿态。

经过与多款 MEMS 陀螺仪对比，本设计中最终选择 Maxim 生产的 MAX21000 陀螺仪。MAX2100 的性能指标如表 4-2：

表 4-2 MAX21000 陀螺仪参数

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Gyro Full-Scale Range	G _{FSR}	User selectable	±500			dps
			±1000			
			±2000			
Gyro Rate Noise Density	G _{RND}	For all the f _g and over the whole V _{DD} including 1.8V	0.009			dps/√Hz
Gyro Rate Noise Density in Eco Mode	G _{SPRND}	For all the FS and over the whole V _{DD} including 1.8V at 200Hz ODR	0.025			dps/√Hz
Gyro Bandwidth (Lowpass) (Note 5)	G _{BWL}		2	400		Hz
Gyro Bandwidth (Highpass) (Note 6)	G _{BWH}		0.1	100		Hz
Phase Delay	G _{PDL}	At 10Hz, 400Hz bandwidth, 10kHz ODR	2.9			deg
Output Data Rate (Note 7)	G _{ODR}		5	10k		Hz
Sensitivity Error	G _{SE}		±2			%
Sensitivity	G _{SO}	G _{FSR} = 31.25	960			digit/ dps
		G _{FSR} = 62.5	480			
		G _{FSR} = 125	240			
		G _{FSR} = 250	120			
		G _{FSR} = 500	60			
		G _{FSR} = 1000	30			
		G _{FSR} = 2000	15			
Sensitivity Drift Over Temperature	G _{SD}	Maximum delta from T _A = +25°C	±2			%
Zero Rate Level Error	G _{ZRLE}		±0.5			dps
Zero Rate Level Drift Over Temperature	G _{ZRLD}	Maximum delta from T _A = +25°C	±2			dps
Startup Time from Power Down	G _{TUPL}		45			ms
Startup Time from Standby Mode	G _{TUPS}	G _{ODR} = 10kHz, G _{BWL} = 400Hz	2			ms
Nonlinearity	G _{NLN}		0.2			%f _g
Angular Random Walk (ARW)	G _{ARW}		0.45			°/hr
In-Run Bias Stability	G _{IBS}	At 1000s	4			%/hr
Cross Axis	G _{XX}		1			%

MAX21000 拥有非常大的量程（2000 度/秒），同时可配置成非常高的灵敏度（最高 960digit/dps），除此之外，还有很小的随机游走误差（ARW）和非常短的相位延迟。在以上优点下，它也非常小，封装尺寸仅有 3mm*3mm 的大小，所以 MAX21000 是飞控比较好的选择。

4.2.2 气压计选型

大气压力传感器作为非惯性器件，虽然不直接参与姿态计算，但它却在自动控制时起了非常重要的作用，所以气压计的选型也是很重要的。而气压计通常在四轴飞行器上，只用作高度计算，所以在进行气压计选型时，只需要重点考虑气压精度与温度测量精度。

在经过对多个型号的大气压力传感器的对比后，设计中采用 MS5611 这款 MEMS 气压计。这款气压计拥有 24bit 内置 ADC，可以通过气温与大气压力计算出海拔高度精度为 10CM，是性能相当不错的传感器。表 4-3 与表 4-4 分别列举了 MS5611 的气压与温度测量参数。

表 4-3 MS5611 压力测量参数

PRESSURE OUTPUT CHARACTERISTICS ($V_{DD} = 3$ V, $T = 25^\circ\text{C}$ UNLESS OTHERWISE NOTED)

Parameter	Conditions		Min.	Typ.	Max	Unit
Operating Pressure Range	P _{range}	Full Accuracy	450		1100	mbar
Extended Pressure Range	P _{ext}	Linear Range of ADC	10		1200	mbar
Total Error Band, no autozero		at 25°C, 700..1100 mbar at 0..50°C, 450..1100 mbar at -20..85°C, 450..1100 mbar at -40..85°C, 450..1100 mbar	-1.5 -2.0 -3.5 -6.0		+1.5 +2.0 +3.5 +6.0	mbar
Total Error Band, autozero at one pressure point		at 25°C, 700..1100 mbar at 10..50°C, 450..1100 mbar at -20..85°C, 450..1100 mbar at -40..85°C, 450..1100 mbar	-0.5 -1.0 -2.5 -5.0		+0.5 +1.0 +2.5 +5.0	mbar
Maximum error with supply voltage	$V_{DD} = 1.8$ V ... 3.6 V			±2.5		mbar
Long-term stability				±1		mbar/yr
Recovering time after reflow (1)				7		days
Resolution RMS	OSR	4096 2048 1024 512 256		0.012 0.018 0.027 0.042 0.065		mbar

表 4-4 MS5611 气温测量参数

TEMPERATURE OUTPUT CHARACTERISTICS ($V_{DD} = 3$ V, $T = 25^\circ\text{C}$ UNLESS OTHERWISE NOTED)

Parameter	Conditions		Min.	Typ.	Max	Unit
Absolute Accuracy	at 25°C -20..85°C -40..85°C		-0.8 -2.0 -4.0		+0.8 +2.0 +4.0	°C
Maximum error with supply voltage	$V_{DD} = 1.8$ V ... 3.6 V			±0.5		°C
Resolution RMS	OSR	4096 2048 1024 512 256		0.002 0.003 0.005 0.008 0.012		°C

4.2.3 其他传感器选型

除了陀螺仪与气压计以外，还有三轴加速度计与三轴磁阻传感器是飞控上必备的传感器。三轴加速度计为飞控提供航位推算与校准陀螺仪的功能，而三轴磁阻传感器可以通过检测地球磁感线方向，为飞控提供机头航向角度，即电子指南针功能。

经过一些对比和筛选后，选定三轴加速度及型号为 ADI 公司生产的 ADXL362，磁阻传感器为霍尼韦尔公司生产的 HMC5983。

ADXL362 提供三轴 12bit 分辨率的加速度测量，提供最大 8G 的量程，已经能满足四轴飞行器在飞行过程中发生大过载时还能保证传感器不超过量程。详细数据在表 4-5 有详细列举。

表 4-5 ADXL362 参数

Table 1.

Parameter	Test Conditions/Comments	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range	User selectable		$\pm 2, \pm 4, \pm 8$		g
Nonlinearity	Percentage of full scale		± 0.5		%
Sensor Resonant Frequency			3500		Hz
Cross Axis Sensitivity ²			± 1.5		%
OUTPUT RESOLUTION	Each axis				
All g Ranges			12		Bits
SENSITIVITY	Each axis			± 10	%
Sensitivity Calibration Error			1		mg/LSB
Sensitivity at X _{out} , Y _{out} , Z _{out}			2		mg/LSB
Scale Factor at X _{out} , Y _{out} , Z _{out}			4		mg/LSB
2 g range			1000		LSB/g
4 g range			500		LSB/g
8 g range			250		LSB/g
Sensitivity Change Due to Temperature ³			0.05		%/°C
0 g OFFSET	Each axis				
0 g Output	X _{out} , Y _{out}	-150	± 35	+150	mg
Z _{out}		-250	± 50	+250	mg
0 g Offset vs. Temperature ³					
Normal Operation	X _{out} , Y _{out}		± 0.5		mg/°C
Z _{out}			± 0.6		mg/°C
Low Noise Mode and Ultralow Noise Mode	X _{out} , Y _{out} , Z _{out}		± 0.35		mg/°C

HMC5983 是一款三轴磁阻传感器，内置 12bit 分辨率 ADC，最大测量范围可达 8 高斯。同时提供高达 220HZ 的数据更新率，是飞控的电子罗盘理想选择。详细参数如表 4-6 所示。

表 4-6 HMC5983 参数

Performance

Field Range	Full scale (FS)	-8		+8	gauss
Mag Dynamic Range	3-bit gain control	± 1		± 8	gauss
Sensitivity (Gain)	VDD=3.0V, GN=0 to 7, 12-bit ADC	230		1370	LSb/gauss
Digital Resolution	VDD=3.0V, GN=0 to 7, 1-LSb, 12-bit ADC	0.73		4.35	milli-gauss
Noise Floor (Field Resolution)	VDD=3.0V, GN=0, No measurement average, Standard Deviation 100 samples (See typical performance graphs below)		2		milli-gauss
Linearity	± 2.0 gauss input range			0.1	\pm % FS
Hysteresis	± 2.0 gauss input range		± 25		ppm
Cross-Axis Sensitivity	Test Conditions: Cross field = 0.5 gauss, Happlied = ± 3 gauss		$\pm 0.2\%$		%FS/gauss
Output Rate (ODR)	Continuous Measurement Mode Single Measurement Mode	0.75		220 160	Hz

4.2.4 飞控主要元器件列表

表 4-7 飞控主要元器件

型号	器件类型
STM32F407VGT6	微控制器
LT1616	DC/DC 电源芯片
U-blox Max 7	微型 GPS 模块
MAX21000	MEMS 三轴陀螺仪
HMC5983	MEMS 三轴磁阻传感器
ADXL362	MEMS 三轴加速度传感器
MS5611	MEMS 气压计
SN65HVD230	CAN 总线接口芯片

表 4-7 列出了飞控中的主要电子元件，不包括电阻、电容、电感和插接件等辅助器件。

4.2.5 飞控原理图、PCB 及其说明

图 4-2 飞控硬件连接简图 描述了飞控板内各个模块与控制器之间的联系。所有的传感器通过 SPI 总线与 STM32F407VGT6（以下简称 STM32）的 SPI3 连接，microSD 卡使用 SPI 模式，与 STM32 的 SPI1 连接。GPS 模块使用 STM32 的 UART4 通信。STM32 的 TIM1，TIM3 和 TIM4 各输出 4 通道 PWM 信号，一共组合成 12 路 PWM 输出。PPM 信号输入使用 STM32 的中断引脚进行捕获。CAN Bus 总线连接到收发器芯片后，通过外部端子进行输出。UART2 接口设计为无线串口数传专用的连接口，通过数传电台与地面上位机通信，所以用外部件接线端子引出。

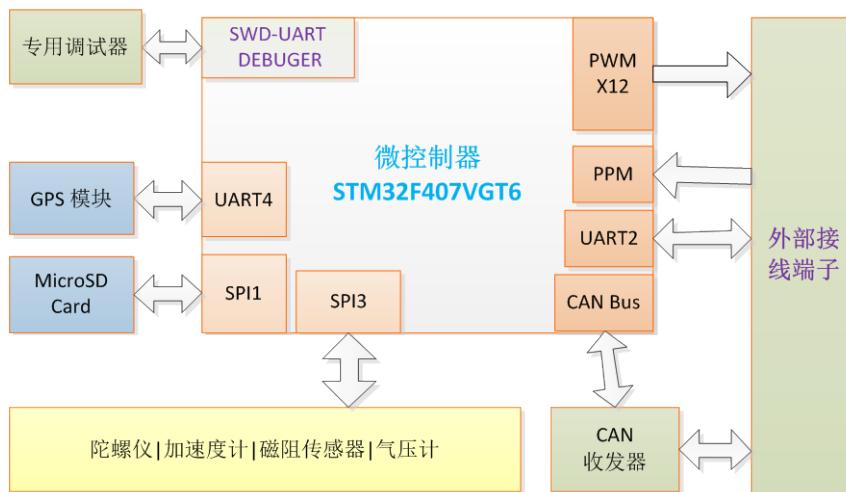


图 4-2 飞控硬件连接简图

图 4-3 飞控主要部分原理图 展现了飞控上主要的元器件与电气连接。完整原理图在附录 1. 飞控原理图。

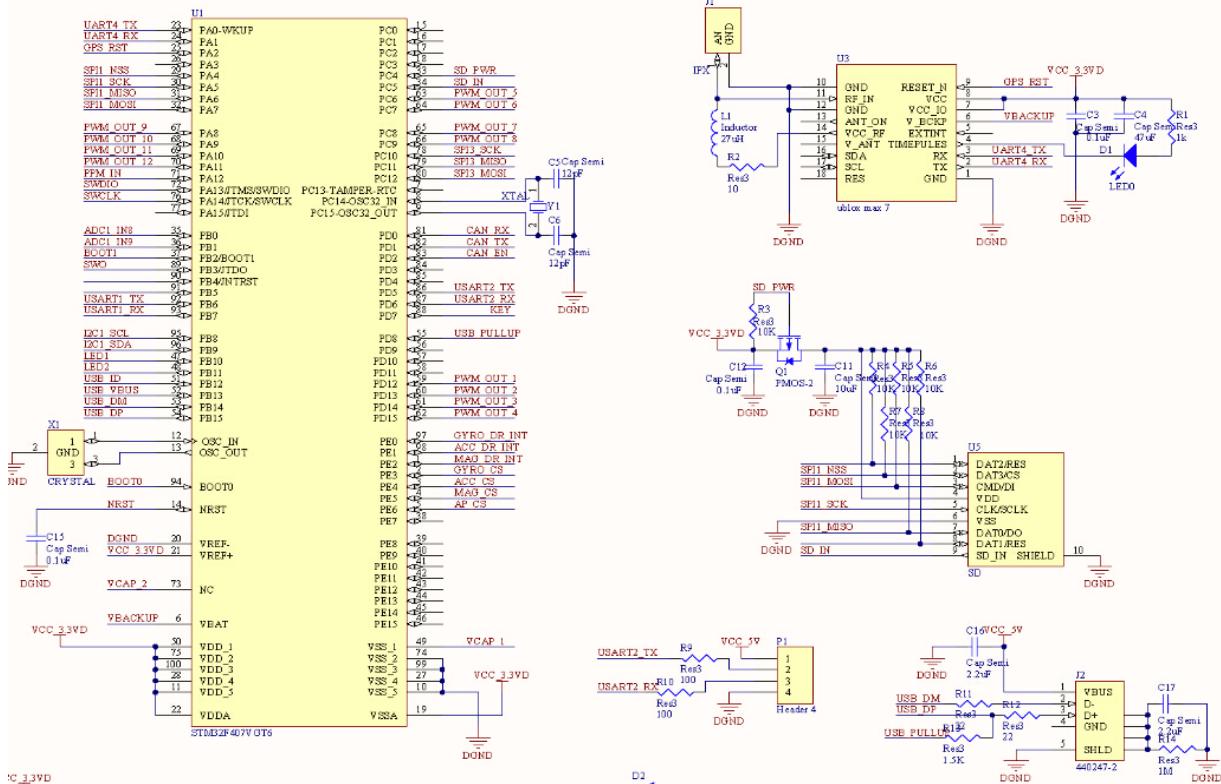


图 4-3 飞控主要部分原理图

图 4-4 飞控设计效果图为 PCB 效果图，展示了元件位置，PCB 尺寸等信息。在 PCB 设计中，应将 PCB 设计得尽量小，以保证其通用性。飞控在设计之初，就将其设计为单板（不需要外接模块）可完成基本的自稳与导航功能，并且其宽度较小，除了可以应用在四轴飞行器中，更可以运用在小型固定翼飞机中。通常小型固定翼飞机需要尽量减少机身的横截面积以达到更小的空气阻力。飞控板仅有 35mm 的宽度，可以运用于大部分的微小型固定翼飞机中。而其单板的自稳与导航能力，更是降低了飞控系统的重量。

PCB 使用 2 层板，可以实现较低的生产难度和降低生产成本，全部元件为正面分布，底部平坦整洁，并设有安装孔，降低飞控板的安装难度。

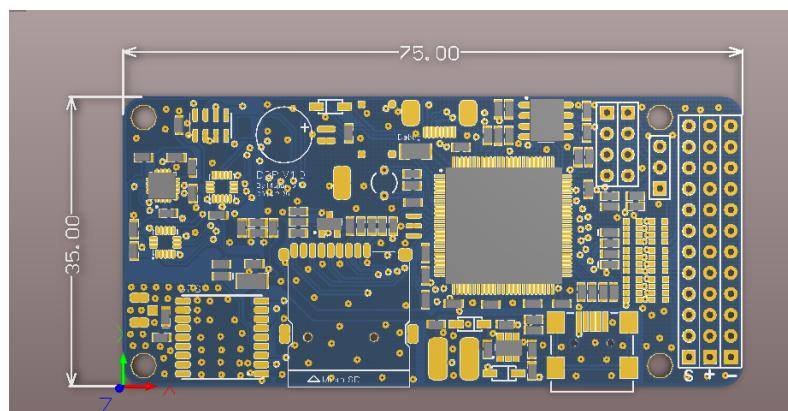


图 4-4 飞控设计效果图

4.3 无感无刷电子调速器硬件设计

四轴飞行器的四个无刷电机，是四轴飞行器唯一的动力输出方式，飞控发出的每一个动作，都必须通过这四个电机直接的相互协调工作才能完成。而电调则作为电机的驱动电路，是控制指令的直接执行。如果无感无刷电子调速器（以下简称电调）设计得不好，将使四轴飞行器变得不稳定或者难以控制，严重情况下甚至无法起飞。

因为航模运动的发展，市面上有不少成熟商品无感无刷电机电子调速器（以下简称商品电调），它们拥有较高的性能和不俗的兼容性，并且价格不贵。商品电调通常由通用的航模遥控接收机（以下简称接收机）控制，其控制信号与舵机控制信号一样。接收机通常发出 TTL 电平的 PWM 方波对舵机或者电调进行控制。PWM 的频率为 50Hz，高电平宽度为连续变化的 1~2 毫秒，对应着舵机运动 -45 度 ~ +45 度。对于商品电调而言，当其接收到的 PWM 信号高电平宽度为 1 毫秒时，认为油门为 0%；高电平宽度为 2 毫秒时，认为油门为 100%。

虽然商品电调有着兼容性好，稳定性有保证的优点，但当其用在四轴飞行器来驱动电机方面，商品电调会显得略微不足。例如就传输信号的方式而言，PWM 波形本身实时性能并不强。当飞控计算出控制量后，通过 PWM 接口传输给商品电调，因为其信号关系，会有 1 毫秒到 20 毫秒不等的延迟，对于四轴飞行器的反馈控制非常不利的。其次，PWM 信号传输的方向是单向的，这其实是一个开环的控制系统。飞控只能命令电调执行动作，而不能知道电调是否已经执行了动作，这将会对降低四轴飞行器的稳定性。

对于本系统而言，设计专用电调是有优势的。第一，本设计中，需尽量减轻重量，如果使用自行设计的电调，可以与机架结合为整体，减少额外的电路板造成的重量增加。第二，自制电调可以挂载在系统的 CAN Bus 总线上，CAN Bus 优秀的稳定性和实时性能将弥补传统 PWM 接口带来的不稳定性和延迟性。第三，自行设计电调可以增加一些传感元件进行反馈，飞控可以从 CAN Bus 上读取电机状态，也使其它虚拟智能终端能够实时地检测电机的运行情况。

对于电调设计，有以下基本要求：

1. 稳定运行。
2. 使用 CAN Bus 和传统 PWM 通信方式。
3. 可以通过 CAN Bus 进行电压、电流反馈。
4. 持续电流 10A 以上，耐压为 3 个串联锂电池电压范围。

4.3.1 无刷电调器件选型及设计原理

因篇幅有限，表 4-8 仅列出所选用的主要元器件：

表 4-8 电调主要元件

名称	类型	典型值
STM32F103C8T6	微控制器	低成本 32bit ARM 控制器, 运行频率 72MHz,
LMV321	运算放大器	轨到轨放大, 3.3V 工作电压, SOT 封装
SN65HVD230	CAN 接口芯片	SOP8 封装, 3.3V 工作电压
IRF7832	N-MOS	30Vds, 16A 平均电流, SOP8 封装
IR2301S	N-MOS 驱动	两路 N-MOS 驱动
XC6206-3.3	LDO	3.3V 线性稳压输出

一个电调的好坏，取决于是否能精确地检测当前电机转子的位置，通常在大型电机上，这个工作是由霍尔元件或是编码器来完成的。但航模的无刷电机为了减少重量和零件复杂度，一般没有内置霍尔元件与编码器，这也是“无感无刷电机”名称的来源。航模无刷电机，一般为三相交流电机，运行中，因为有感应电动势存在，所以电调可以在运转时，检测几个线圈的电压差，即可检测转子的电角位置，从而执行换相工作。

本设计中，MCU 通过内置 ADC 检测线圈电压差，通过与内置的表格对比后，可以知道转子的电角，从而可以自动实现准确的换相工作。同时 MCU 会监听 CAN Bus 总线上的报文，当检测到飞控发来的控制信息时，立即更新数据油门数据，使电机的输出功率发生变化。每间隔一段时间，电调就会发送包含电流，电压，转速的 CAN 报文到总线上，实现反馈机制。

电机驱动部分，是由 3 个半桥组成，每一个半桥由两个 N-MOS 组成，并有一个 N-MOS 驱动器驱动。电调一共使用 6 个 N-MOS 进行电机的驱动。此外 MOS 管内置反向保护二极管，所以电路中不需要设计额外的保护二极管。

电流检测部分为一个轨到轨运算放大器和一个串联在输入电路中的采样电阻（50 毫欧）组成。当 MCU 检测到电流过大时，会立即切断电机供电，实现软件过流保护功能。

4.3.2 无刷电调器原理图、PCB 图与实物图

如图 4-5 图 4-6 图 4-7 所示，分别为电调板的主要部分原理图文件，PCB 文件，实物图。

完整的电子调速器原理图在附录 2.无刷电调原理图。

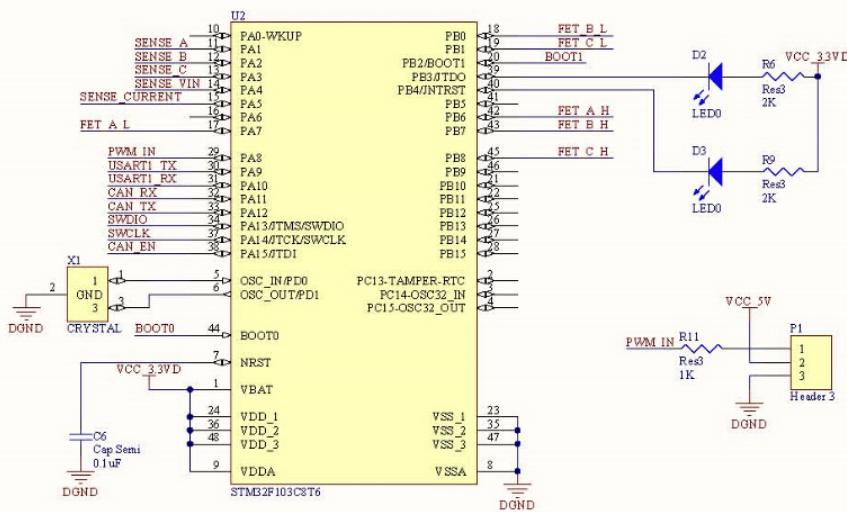


图 4-5 无刷电子调速器主要部分原理图

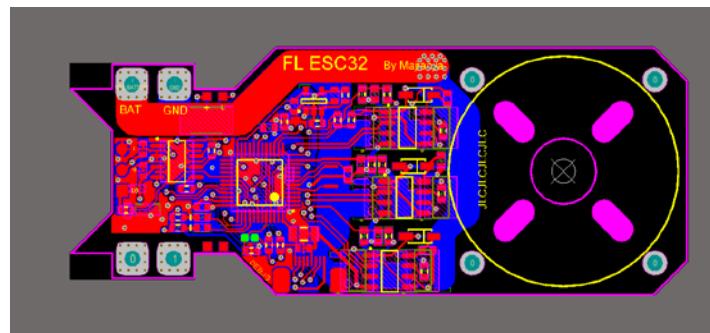


图 4-6 无刷电子调速器 PCB 图

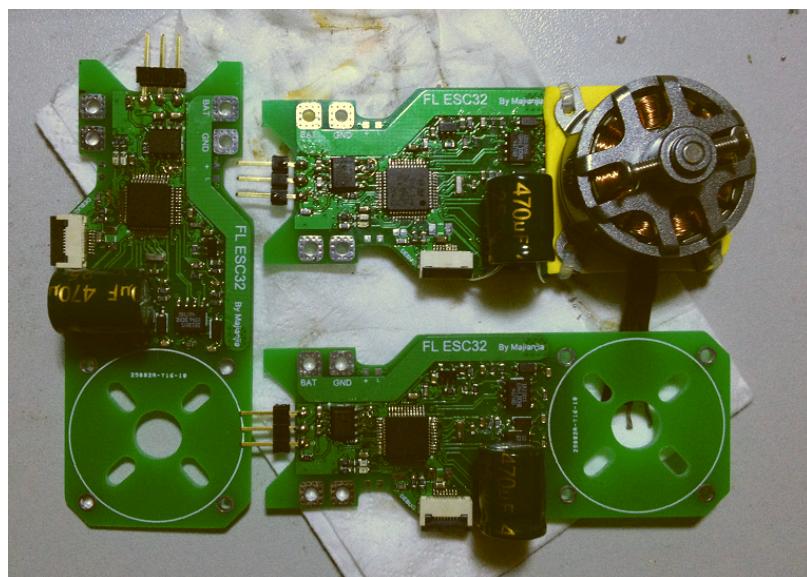


图 4-7 无刷电子调速器成品板

4.4 电源底板硬件设计

4.4.1 电源底板的设计描述

电源是整个四轴飞行器中的基层部分，在一般系统中电源通常只起到提供固定电压，保护电路的作用。在本设计中，尝试地加入一些智能的功能在里面，如多路电压、电流监控，电池电量管理等。

电源变换采用的是 AP1510 DC/DC 降压芯片，将电池电源电压（11.1~12.6V）降压到 5V，给所有挂载在 CAN Bus 总线上的终端供电（包括飞控、电调和经过 UEB 拓展的模块）。另外通过 LDO 将 5V 降压到 3.3V 给电源板上的 MCU 和其他芯片供电。

电源板与电调一样，使用 STM32F103C8T6 作为 MCU，同样具备 CAN Bus 通信能力。电路方面非常简单，电压与电流检测均使用 MCU 内置的 12bit ADC 来完成。

电流监控使用 LMV321 运算放大器和一个采样电阻进行采集。原理为欧姆定律，电流流过电阻产生的降压。运放将串联在电路中的采样电阻的电压差放大后，直接输出给 MCU 的 ADC 模拟接口。通过简单的计算后即可得到电路中的电流。电源板同时监控 5V 与电池电源两路电流。

电压监控使用两个电阻组成分压电路进行分压。分压产生的电流较小，MCU 的 I/O 口内部已有 5.1V 的保护二极管，所以电路上没有外接保护二极管。电源板一共提供最高 5 节锂电池的单节电压监测和总电压监测（最高 21V），可以最大程度地保证锂电池。

4.4.2 电源底板原理图、PCB 图与实物图

如图 4-8、图 4-9 所示，分别为电源底板的原理图和 PCB 图。完整原理图在 **3.电源底板原理图**。

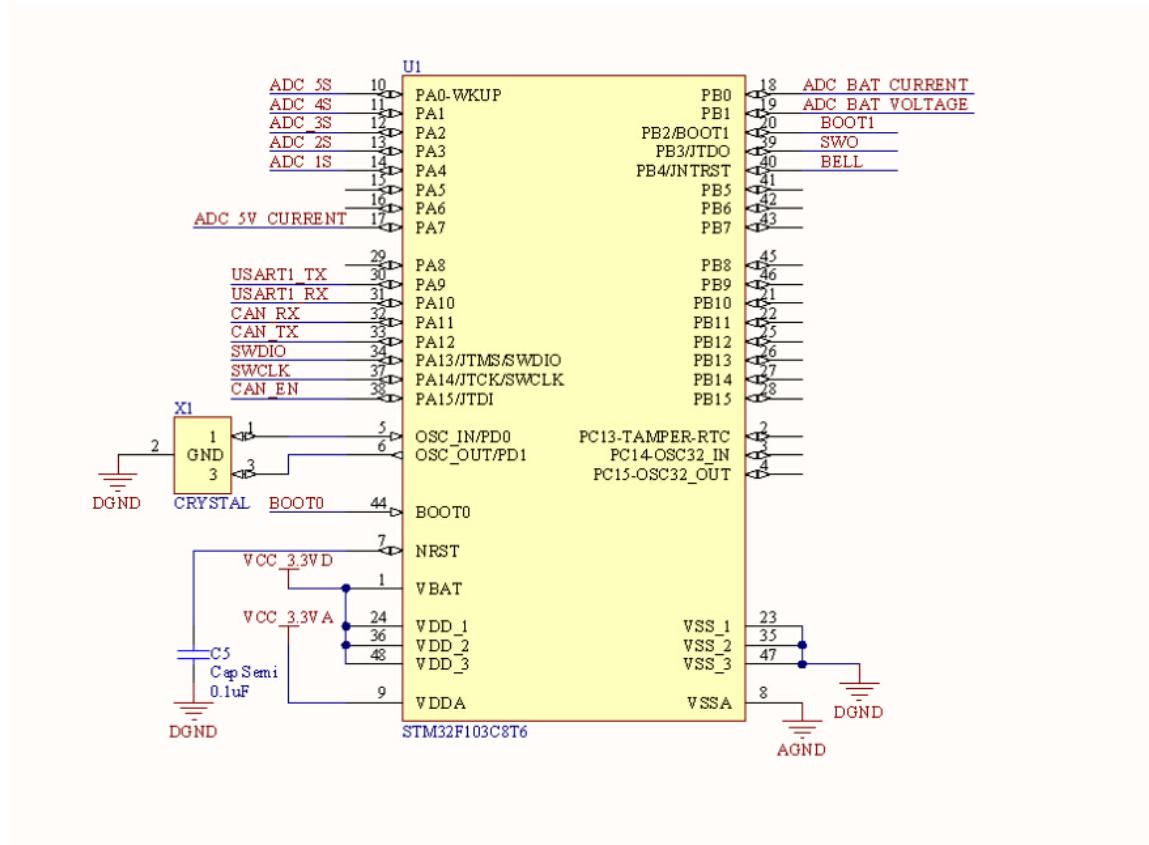


图 4-8 电源底板部分原理图

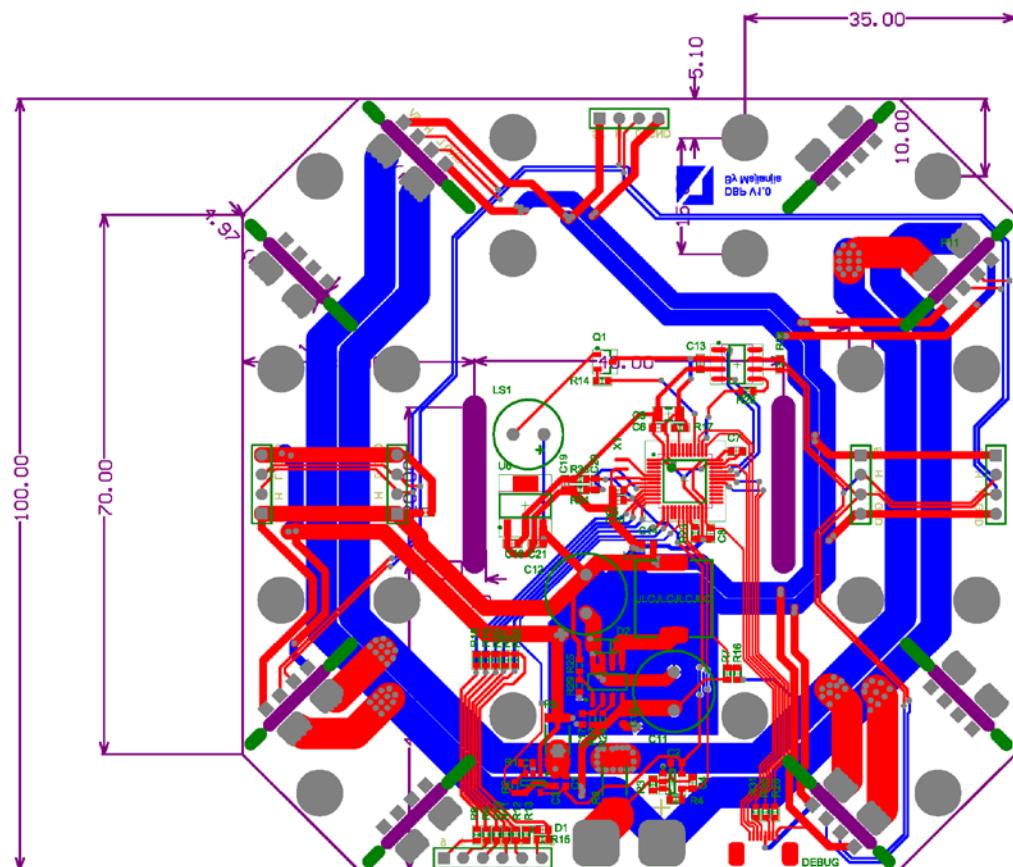


图 4-9 电源底板 PCB 图

4.5 其余电路硬件设计

除了飞控、电调和电源板以外，还需要制作一些配套的电路。以下列出其中两个，UEB 和 专用调试器的电路板的简介，原理图和 PCB。

4.5.1 通用拓展板——Universal Expansion Board

UEB （Universal Expansion Board）的设计目标，是作为一个通用的微小型多种接口与 CAN Bus 之间的转换器，并同时实现初步的数据处理与计算能力。丰富的拓展能力和微小的尺寸是它最大的特点。

所以 UEB 进行如下设计：为了实现“小”的特点，MCU 选择为 STM32F103T8，其封装为 6*6 毫米的 QFN-48；为了实现多种接口的转换，在板子上引出了 5V, 3.3V 电源接口与 MCU 的 PA0~PA7 这几个 GPIO 口。这 8 个 I/O 虽然数量较少，但通过配置复用功能，可以实现多种接口的配置能力，几乎囊括了所有常用的接口。表 4-9 展示了 UEB 支持的外扩功能类型。

表 4-9 MCU 引脚与功能支持对照

I/O 口名称	模拟输入	UART	SPI	INT	PWM	输入捕获
PA.0	●	●		●	●	●
PA.1	●	●		●	●	
PA.2	●	●		●	●	●
PA.3	●	●		●	●	
PA.4	●	●		●		
PA.5	●		●	●		
PA.6	●		●	●	●	●
PA.7	●		●	●	●	

注：“●”代表引脚支持该功能

已成功使用 UEB 来拓展的模块有许多，它们包括：超声波测距模块，接收机解码模块，光流传感器模块，空速计等功能模块等。使用 UEB 来拓展外部模块，为四轴飞行器的拓展增加了灵活性与稳定性。

如图 4-10、图 4-11 所示，分别为 UEB 的主要部分原理图与 PCB 图。完整原理图在附录 4.UBE 原理图。

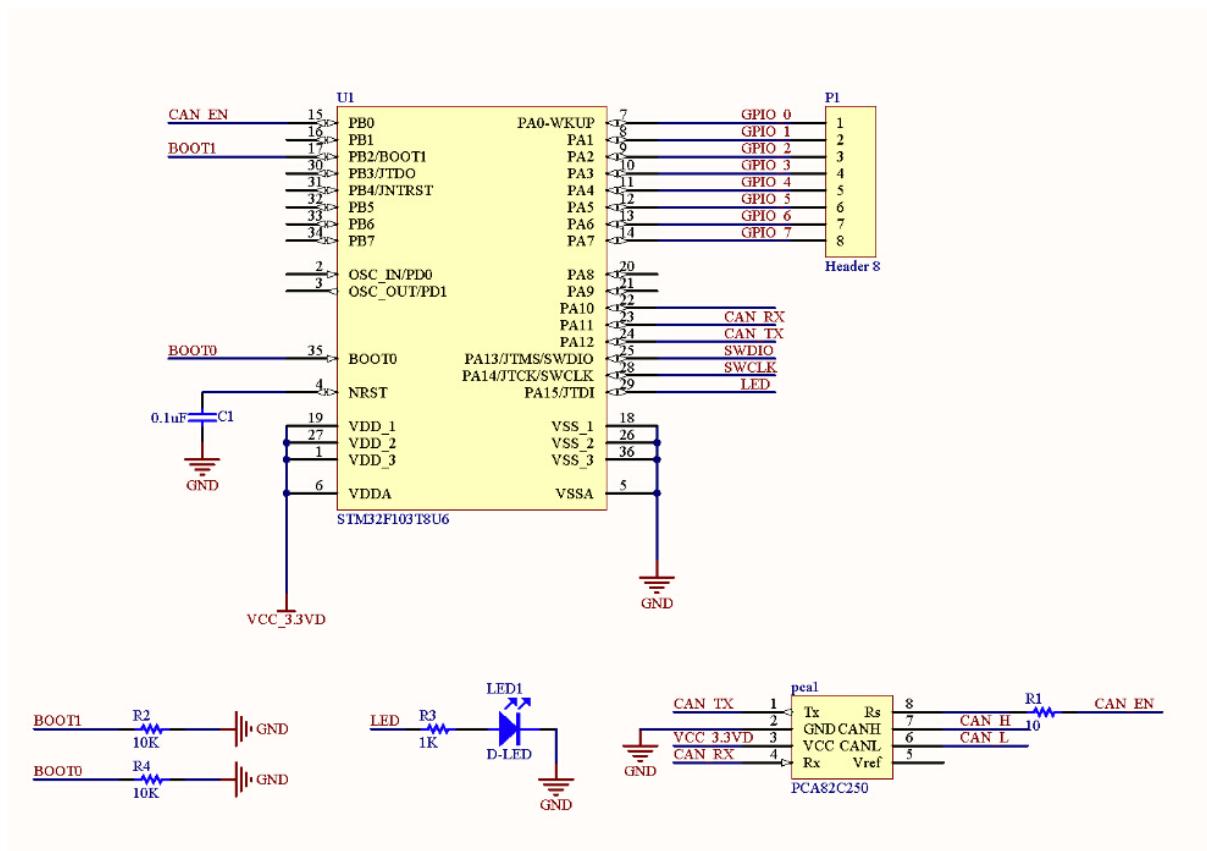


图 4-10 UEB 主要部分原理图

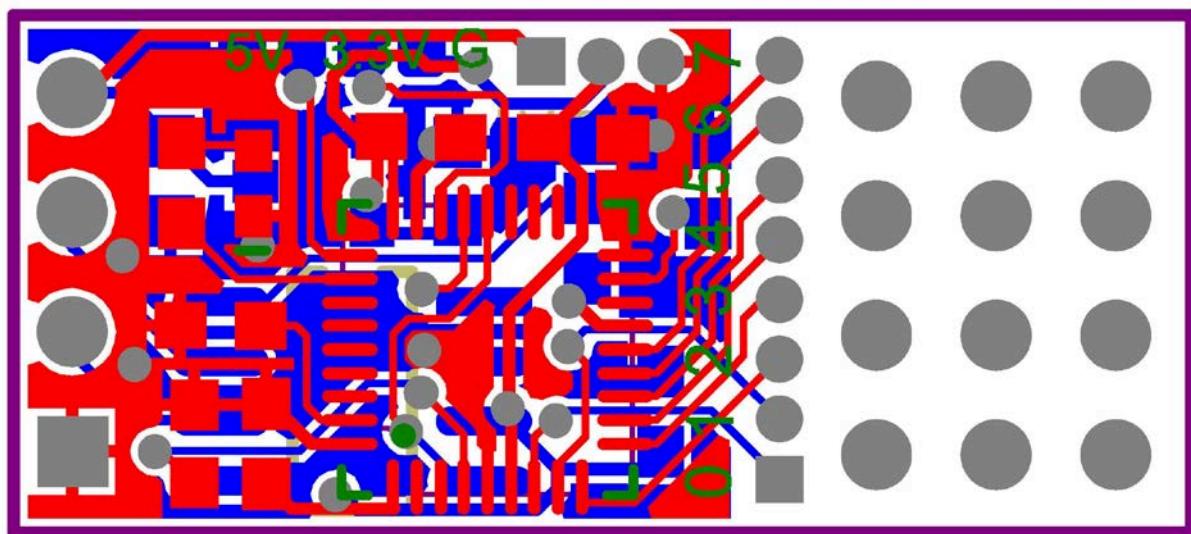


图 4-11 UEB PCB

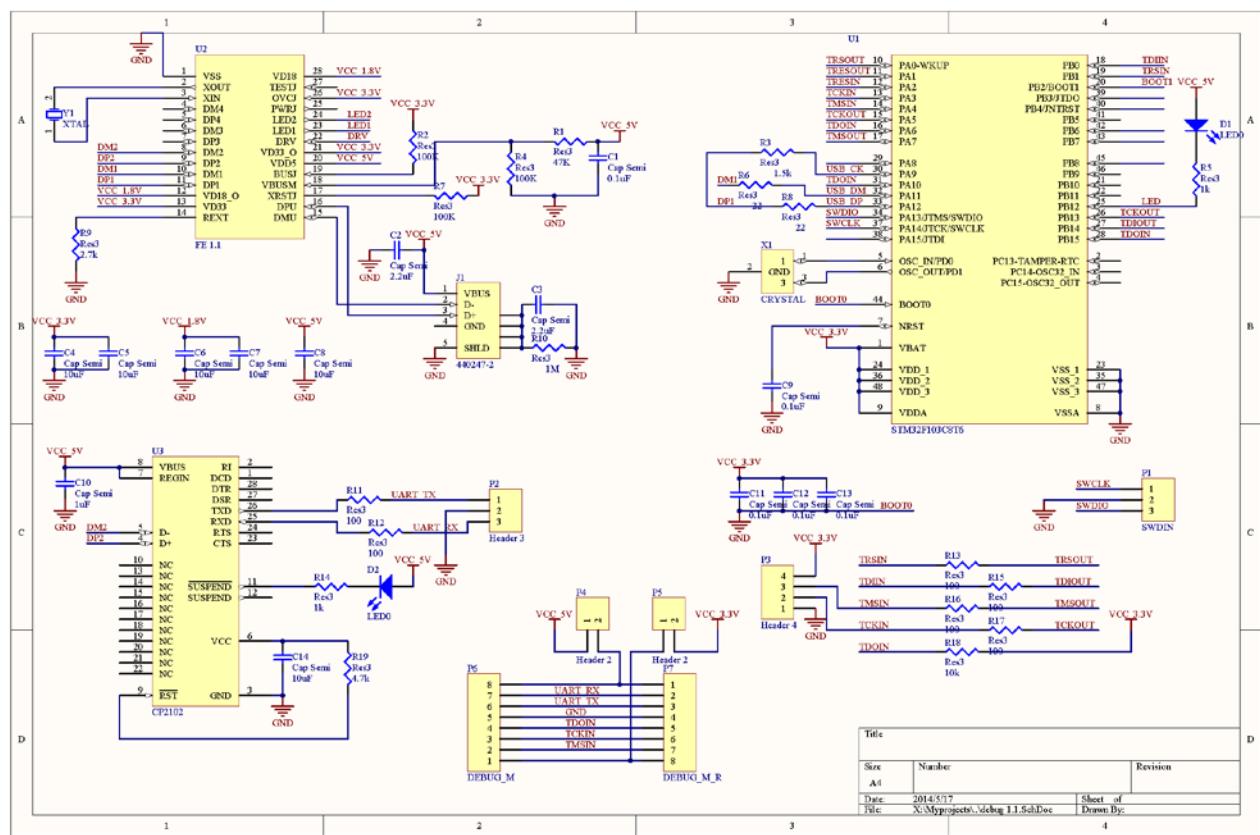
4.5.2 专用调试器

设计的所有的板子都采用了 STM32 系列的 MCU。STM32 属于 Cortex-M 系列内核的微控制器，通常有两种硬件调试接口 JTGA 与 SWD 接口。

大多数市场上的调试器，如 JLINK 与 ULINK 都是通过 JTGA 调试接口调试的，JTGA 接口尺寸非常大，不适合本设计中的电路板。通过查找资料后，决定设计一个 SWD 接口与 UART 接口的联合调试器。

原理如下，在板子上集成一个 USB-HUB 芯片（FE1.1）将 1 路 USB 2.0 接口通过集线器分为 2 路 USB 2.0 接口。将其中一路连接到 USB-UART Bridge 芯片 CP2102，另一路连接到 Jlink ARM-OB 上。ARM-OB 是一个基于 STM32F103C8 的 Jlink-V7 调试器，支持 SWD 接口。UART 与 SWD 通过一个自定义的 8Pin FPC 插座引出，这样一来，可以非常方便地插拔调试线，非常方便进行不同的电路板进行调试。

如图 4-12、图 4-13 所示，分别为专用调试器的原理图和实物图。



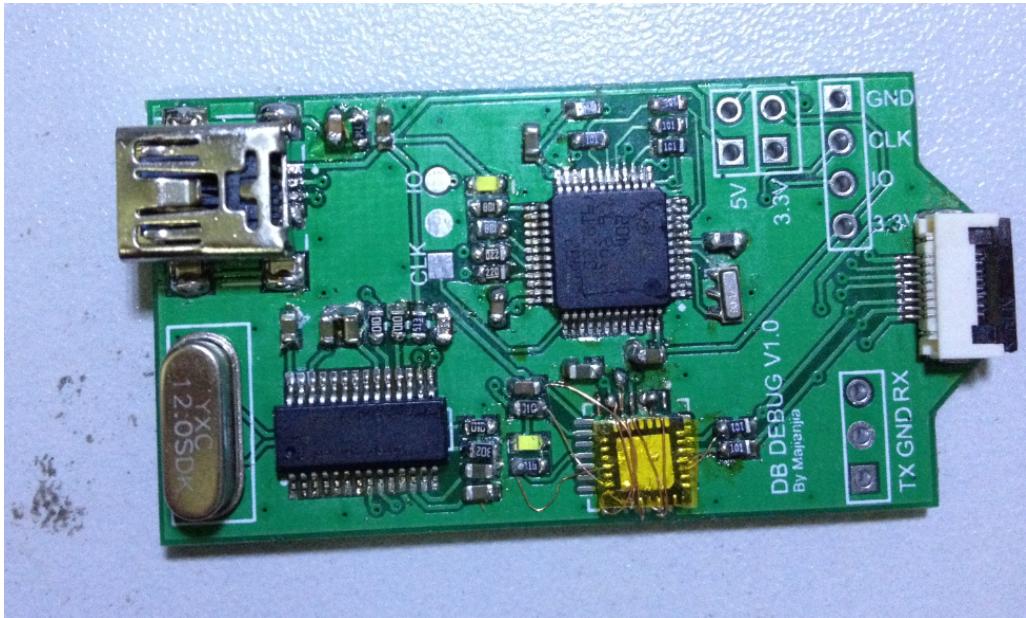


图 4-13 专用调试器实物图

5 软件设计

四轴飞行器的软件涉及多方面的知识，主要包括滤波算法，姿态算法与控制算法等几个方面。

前文说过，在本系统里，飞控相当于四轴飞行器的大脑，需要执行多个较为复杂的算法计算和同时执行多个线程。所以本章节主要将飞控部分程序的实现过程和原理进行详细阐述。对于其他拓展板上的软件设计，会进行相对简单的叙述。

在正式开始阐述飞控程序之前，将首先介绍 RT-Thread 和虚拟智能终端设计理念。

5.1 实时操作系统 RT-Thread 简介

5.1.1 飞控程序建立在 RTOS 上的必要性

前面曾经多次间接描述过飞控的程序的工作，实际上，本设计中的飞控除了需要运行很多算法以外，更需要执行很多不同的工作。在简单的嵌入式系统中，是由一个大的死循环来重复进行相同的工作，当需要拓展功能时，新的程序需要按照状态机的编程思维，以非堵塞的编程风格去编写程序。状态机是一个简单而且高效的多任务编程方式，但会不可避免地引入很多的程序状态标志变量。当系统变得很大时，这些状态标志得不到很好的管理，容易形成 Bug 或者使系统的执行效率变得很低。由于状态机本身也是一个大的死循环，当任务变多时，状态机的执行周期变化频繁，单个任务不能保证实时性。而严格的实时性对于飞控系统来说，是非常关键的。

飞控系统的工作非常多，而且大部分工作是并行完成的，关键工作（姿态计算与传感器数据提取等）需要相对较高的执行周期与实时性，而普通工作（温度检测，GPS 解码等）对实时性要求不是那么高。最理想的情况是每一项工作都能够并行地运行，并且互不打扰，这样一来可以保证各自工作的实时性，并且当其中一项工作陷入死循环后，其他工作能不受它的影响，这样一来飞控的鲁棒性会更强。

但是，不可能每一项工作都是单独运行的，它们之间需要沟通，需要共享一些数据。在传统嵌入式程序中，不同的函数想要共享同样的数据，通常需要定义很多的全局变量，当任务足够多时，不同的工作之间的同步显得非常麻烦。如果程序中引入了中断，同时将会埋下著名“生产者消费者问题”的隐患。

为了避免这些问题，将飞控的程序运行于 RTOS 上是非常有必要的。RTOS (Real Time Operating System) 是一类嵌入式系统的统称，它们通常有着非常不错的实时性能，比如线程调度时间一定，支持线程通信，支持抢占式调度，占用的储存空间和内存较少。常见的 RTOS 有 RT-Thread、Free RTOS、eCos、VxWorks 等等。

5.1.2 选择 RT-Thread

RT-Thread (<http://www.rt-thread.org/>) 是由中国开源社区主导开发的 RTOS 项目（遵循 GPLv2 许可协议），它包含实时嵌入式系统相关的各个组件：实时操作系统内核，TCP/IP 协议栈、文件系统、libc 接口、图形界面等。

如图 5-1 所示，为 RT-Thread 的软件结构图。

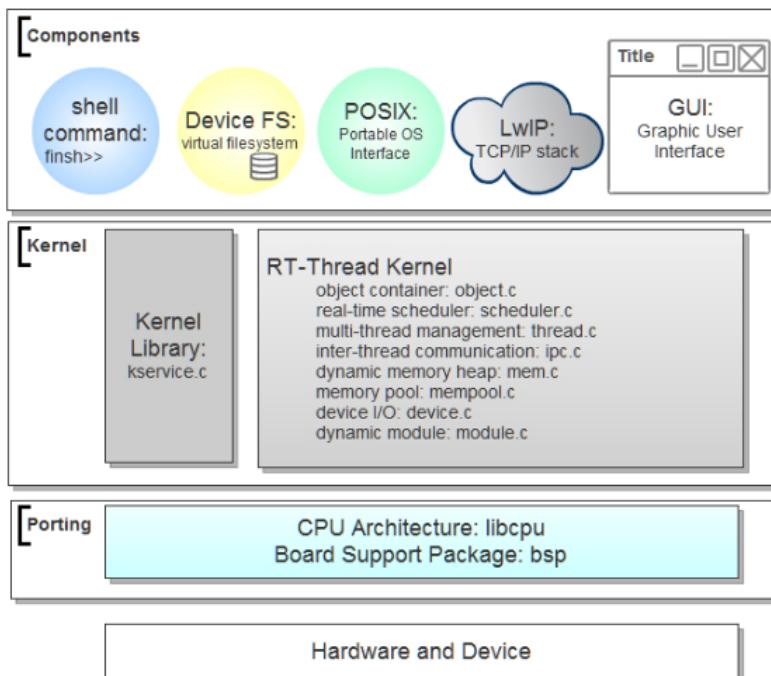


图 5-1 RT-Thread 软件结构图

编写本文时，RT-Thread 稳定版本为 1.2，所有模块上运行的 RT-Thread 的版本也为 1.2。在 RT-Thread 上编写程序，是非常简单的。对于用户来说，只需要关心上层应用的编写和驱动程序的编写即可，而驱动程序部分，RT-Thread 提供了大量的 BSP 包，可以直接查找与所使用的控制器对应的驱动程序，通常只需要小幅修改后即可使用。

RT-Thread 的内核是一个非常优秀且稳定的内核。提供了常规 RTOS 提供的各种线程通信方式，包括邮箱，消息队列，互斥量，信号量，事件等同步方式。此外还提供了多种内存管理方式和丰富的内核库。在资源占用上，RT-Thread 是非常灵活的，对 RT-Thread 进行裁剪非常方便，只需修改一个头文件，即可任意删除或添加所需组件，后面章节将要描述的电调程序是一个非常好的例子。

RT-Thread 虽然采用 C 语言进行编写，但其使用面向对象的编程思维，将系统中对象进行统一管理。它的设备管理器拥有强大的硬件管理能力，微控制器上的所有的 I/O 接口，都可以虚拟成统一的驱动，这个方式类似 Linux 中对驱动的操作形式。这对飞控本身有非常重要的影响，例如，设计中 4 个传感器共用 MCU 的一个 SPI3 接口，为了保证实时性，其中 3 个传感器采用它本身的 Data Ready 中断通知 MCU 来读取数据，如果没有做好互斥，实际中会因为其中两个传感器同时产生中断而导致 SPI3 被复用。如果在 RT-Thread 的设备管理器中操作，每一个 SPI 总线被看成一个 SPI 总线设备，每一个传感器被看成一个挂载在 SPI 总线设备上的 SPI 设备。当 SPI 设备需要使用总线设备时，必须先申请，此时设备管理器会自动进行互斥和优先级处理，这样一来可以完全避免这种问题的发生。当然这只是使用 RT-Thread 的其中一个好处。

5.2 虚拟智能终端的设计理念

本文中，虚拟智能终端（以下简称终端）的概念，是专门为了配合本设计中的飞控和 CAN Bus 而产生的概念。在 4.1 四轴飞行器单总线（CAN Bus）设计理念 中详细说明了使用单总线的好处，而硬件必须由软件来配合，才能发挥出其相应的作用。为了发挥 CAN Bus 带来的优势，这里提出虚拟智能终端的设计理念。

终端本身是虚拟的，不依赖固定的硬件，在本设计中，特指能接受或发送 CAN 报文的线程。这些线程可以检测总线上的报文，或者向总线上发送报文。终端的数量不受限于硬件数量，也就是说一个能收发 CAN 报文的硬件中，可以运行多个终端。

所有的四轴飞行器上的 MCU（包括飞控和其他模块），均运行着 RT-Thread，包括电源板，电调板，飞控板和多个拓展了传感器的 UEB。终端的实体，是一个 CAN 报文收发线程，该线程运行于 RT-Thread 之上。于是，可以得出终端运行的两个必要条件：其一，硬件拥有 CAN Bus 总线接口，其二，操作系统为 RT-Thread。

5.1.2 章中描述过，RT-Thread 拥有一个优秀的设备管理器来管理驱动。这个设备管理器为多个线程共通访问一个硬件外设提供了便利，意味着多个终端可以不冲突地访问

CAN Bus 总线。也正是因为有了统一的底层和操作系统，不同硬件之间移植终端会变得非常简单，仅仅需要修改 MCU 的 CAN 外设中报文过滤器相关的配置。

引入终端这个概念，对飞行器而言是有重要的意义的。飞行器需要尽量减少重量，而过多的拓展显然会增加不必要的重量。既然所有的模块都使用了 Cortex-M 内核的控制器，硬件拥有丰富的处理能力和外设资源，为什么不充分利用他们呢？如果拓展模块带有一个 32 位的 ARM 控制器，而工作单纯是为了在电量低的时候发出报警的声音，这是一种多么浪费的行为。有了虚拟终端的概念，可以在一个 MCU 上运行多个虚拟终端，如图 5-2 典型虚拟智能终端（电源板例子），MCU 内部一共有 3 个终端，其中一个负责控制灯光，一个负责控制蜂鸣器发出声音，另一个负责反馈电池信息，当他们接收到飞控或是其他终端发出的特定报文后，可以独立执行自己的工作。虽然硬件上这些终端位于同一个 MCU 内，但实际上他们已经能控制 MCU 的不同外设进行各自的工作了。对终端进行编程的时候，只需要保证同一个 MCU 内不同的终端所占用的 MCU 资源不冲突，这是非常容易的事。

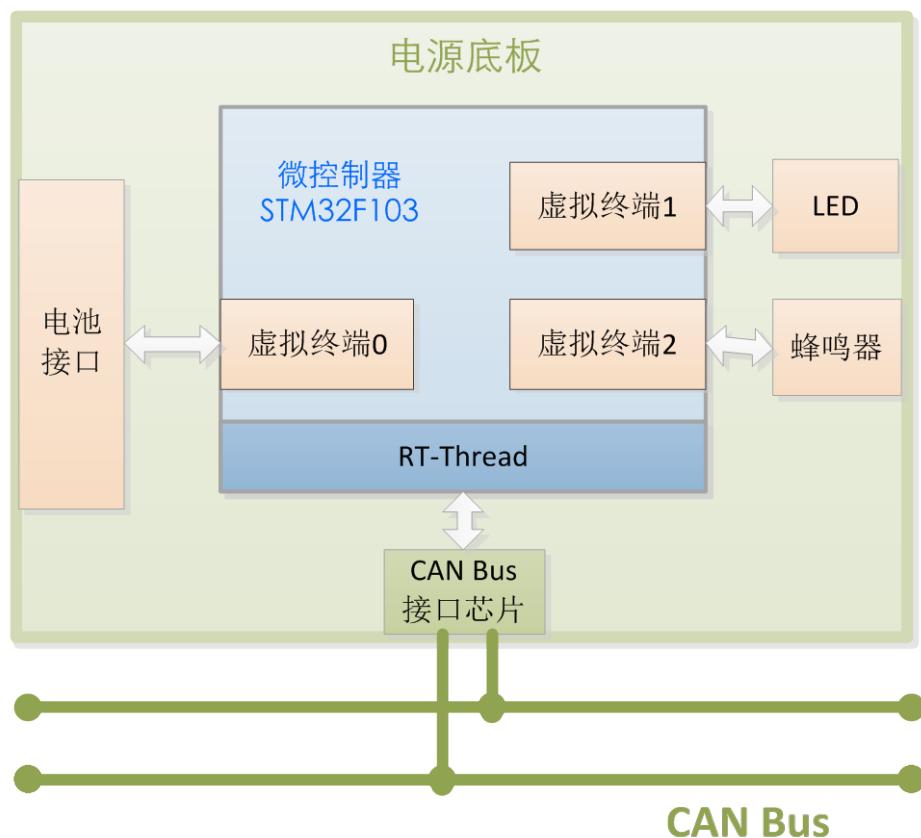


图 5-2 典型虚拟智能终端（电源板例子）

5.3 飞控程序设计

设计到这里，四轴飞行器有了硬件和软件平台，但如果飞控上运行的各个线程，飞行器是不可能离开地面的。

主要程序主要包括三大部分，姿态检测，控制算法，导航算法。接下来将介绍四轴飞行器的这三大部分算法。

5.3.1 线程总览与程序结构

在飞控程序中，各个线程执行着各自的工作。在 RT-Thread 下，可以通过其内置的 Finsh shell 将各个线程的状态打印出来。

Finsh 是 RT-Thread 内置的一个类似 C 语言语法的 shell 组件，也是一个小型的虚拟机，可以直接运行终端输入的小型 C 程序。这里只用到 Finsh 的命令功能，将专用调试器与飞控板连接，Finsh 通过调试器中的 USB 转 UART 功能，即可连接到电脑上的终端软件。这里使用的终端软件是 SecureCRT。飞控启动完成后，在 SecureCRT 中输入内置的 list_thread() 指令，可以打印出当前的线程状态如图 5-3 正在运行的线程。

thread	pri	status	sp	stack	size	max used	left tick	error
can_tx	0x0a	suspend	0x0000009c	0x000000400	0x00000009c	0x000000001	000	
can_rx	0x0a	suspend	0x000000114	0x000000400	0x000000114	0x000000001	000	
tshell	0x14	ready	0x00000019c	0x000000800	0x0000001b0	0x000000006	000	
tidle	0x1f	ready	0x000000054	0x000000400	0x000000074	0x00000001c	000	
mavlink	0x0c	suspend	0x00000025c	0x000000400	0x0000002b4	0x000000001	000	
expand	0x0f	suspend	0x000000094	0x000000400	0x000000094	0x000000001	000	
navigato	0x0d	suspend	0x00000014c	0x000000400	0x0000002b4	0x000000001	000	
ppm	0x0c	suspend	0x00000008c	0x000000400	0x00000008c	0x000000001	000	
canbus	0x0b	suspend	0x000000084	0x000000400	0x0000000e0	0x000000001	000	
ctrl	0x07	suspend	0x000000114	0x000000800	0x000000134	0x000000001	000	
log	0x16	ready	0x0000000a4	0x000000800	0x00000011c	0x000000001	-02	
calendar	0x08	suspend	0x00000009c	0x000000400	0x0000000a4	0x000000001	000	
position	0x06	suspend	0x000000184	0x000000800	0x000000294	0x000000001	000	
hbeat	0x08	suspend	0x000000124	0x000000400	0x000000124	0x000000001	000	
airpress	0x05	suspend	0x00000014c	0x000000400	0x0000001e4	0x000000001	000	
compass	0x05	suspend	0x000000174	0x000000400	0x0000001cc	0x000000001	000	
acc	0x05	suspend	0x00000012c	0x000000400	0x0000001c4	0x000000001	000	
gyro	0x04	suspend	0x000000134	0x000000400	0x0000001b4	0x000000001	000	
sd	0x14	suspend	0x000000094	0x000000400	0x000000094	0x000000001	000	
gps	0x0a	suspend	0x00000010c	0x000000400	0x000000274	0x000000002	000	
second	0x03	suspend	0x00000007c	0x000000100	0x00000007c	0x000000005	000	
led1	0x14	suspend	0x000000094	0x000000400	0x000000094	0x000000001	000	
	0,		0x000000000					

图 5-3 正在运行的线程

通过图 5-3 正在运行的线程 可以看到，飞控中运行了 22 个线程，其中“tshell”线程是 finsh shell 的执行线程，“tidle”线程则是系统内置的空闲线程，其余 20 个线程均为飞控相关的线程。

由于线程编程中，编写的对象为独立线程，一般情况下大部分线程为并行执行。只有一些特殊的线程之间是需要按顺序执行。由于这种特殊性，很难通过一张简单的程序流程图将飞控的各个线程表现出来，所以在这里将各个线程的功能做一个说明。值得注意的是，线程之间会有底层，应用层之分，区别在于底层更靠近硬件，例如读取传感器，操作 LED 等，而应用层基本与硬件无关，如姿态计算，导航计算等。线程之间也用优先级来区分其对实时性的要求，优先级数字越小则线程的优先级越高，相应线程的实时性越好。

表 5-1 线程注释

线程名称	所属层	优先级	功能简述
gyro	底层	0x04	读取陀螺仪的数据，滤波，动态改变量程
acc	底层	0x05	读取加速度计的数据，滤波，动态改变量程
compass	底层	0x05	读取磁阻传感器数据，滤波，计算航向
airpress	底层	0x05	读取气压计数据，滤波，计算海拔高度，爬升率等
position	应用层	0x06	使用各个传感器的值来计算姿态角度
ctrl	应用层	0x07	通过姿态角度与期望角度，计算出控制量并输出
navigato	应用层	0x0D	负责导航功能，可以接替手工操作输出期望角度
expand	应用层	0x0F	终端实体与解码函数
canbus	应用层	0x0B	监视 CAN 收发线程是否正常
can_tx	底层	0x0A	CAN 消息发送线程
can_rx	底层	0x0A	CAN 消息接收线程
hbeat	应用层	0x08	负责扫描系统中重要线程是否工作正常
second	应用层	0x03	负责更新系统时钟
calendar	底层	0x08	负责更新与校准 RTC 时钟
gps	底层	0x0A	解码 GPS 输出的 NEMA-183 字符串
mavlink	应用层	0x0C	Mavlink 协议守护线程，负责与上位机交换数据
ppm	底层	0x0C	解码接收机的 PPM 输入信号
log	应用层	0x16	负责将所需数据流保存在 microSD 卡中，用于调试
sd	底层	0x14	监视 microSD 插座，并完成热插拔的相关工作
led1	底层	0x14	LED 的闪动方式执行线程
tshell	应用层	0x14	Finsh shell 值守线程
tidle	应用层	0x1F	空闲线程，RT-Thread 内置线程，用于释放资源

在表 5-1 中可以看到各个线程的功能。其中有几个线程是比较重要的，而它们之间是有着先后执行关系的。图 5-4 中将这部分线程的顺序表示出来了。线程之间的同步方式使用的是 RT-Thread 提供的信号量。当上级线程完成工作，例如陀螺仪数据读取成功，并且新的计算周期开始后，就发送一个信号量给下级线程，下级线程在完成工作后，会一直等待信号量，除非达到设定的超时时间造成超时返回。超时返回后，会有相应的处理方案，比如发送错误信息到调试口，或者重启上级线程。一般情况下不会有超时返回，超时返回往往意味着最严重的错误，飞行已经无法再继续。

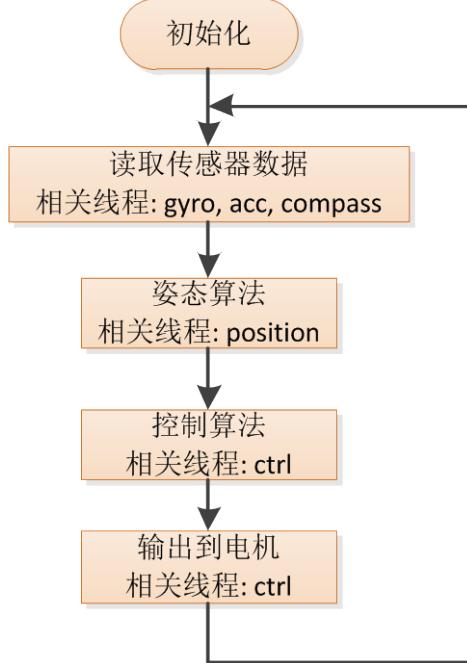


图 5-4 核心线程

5.3.2 提取传感器数据

提取传感器数据是四轴飞行器能检测自身姿态的基础。有了传感器数据，才能计算出实时的姿态与位置数据。

飞控上集成了 10 个自由度的传感器，其中包括三轴陀螺仪、三轴加速度计、三轴磁阻传感器和大气压力传感器。硬件设计上，这些传感器都是挂载在 MCU 的 SPI3 接口。前面提到过，飞控程序是运行在 RT-Thread 之上的，而 MCU 的 SPI 接口则由 RT-Thread 的设备驱动进行管理。这样一来，MCU 的 SPI3 接口在软件中，实际上是以 SPI 总线设备的形式存在。而在初始化中，所有的传感器都以 SPI 设备的形式，挂载在 SPI 总线设备上了。虽然传感器不同，但是对于它们之间的编程都是及其类似的。

这里以读取陀螺仪的测量值为例子，结合源代码并添加中文注释，简要说明飞控中读取传感器数据的过程。陀螺仪的读取，是由线程“gyro”负责的，其功能包括陀螺仪的初始化，数据滤波，动态量程处理等。

首先是接口初始化部分，这部分并不在 gyro 线程中，由系统硬件初始化线程完成。

```
//将 SPI3 总线注册到系统中，SPI 总线设备的名称为“spi3”。
stm32_spi_register(SPI3, &stm32_spi, "spi3");
//将 gyro 设备挂载在名为“spi3”总线上，设备名称是“spi30”。
rt_spi_bus_attach_device(&rt_spi_device_30, "spi30", "spi3", (void*)&stm32_spi_cs_30);
```

经过上面的工作，“gyro”的初始化函数可以通过查找“spi30”这个设备找到陀螺仪。在 RT-Thread 中系统的初始化使用类似 Linux 的方式，为了统一管理初始化函数，使用函数指针的形式依次调用系统中的各个初始化函数。这样做好处是在系统真正开始运行之前，已经完成了初始化，并且初始化的顺序是固定的。如果在不同的线程中调用自己的初始化，一旦他们之间有依赖关系，就需要其他手段来控制初始化顺序，这是非常复杂且容易出错的。

手动编写陀螺仪初始化函数，并把它输出到系统的初始化调用列表中。

```
int rt_max21000_init(void); //初始化并打印出相关的信息
INIT_APP_EXPORT(rt_max21000_init); // 将其输出到应用层初始化
```

通过以上两步，在“gyro”线程初始化完成之前，传感器就已经可以正常工作了，“gyro”线程中即可直接读取传感器的数据或者修改量程。“gyro”线程实际也是一个大循环，这也是最直观的编程方式。

每一个新的数据转换完成，传感器通过硬件 DataReady 中断驱动 MCU 的外部中断引脚。在中断程序中，它释放一个信号量给“gyro”线程以通知它读取新的数据。传感器的更新速度由传感器内的寄存器设置。在飞控程序中，传感器的 ODR（Output Data Rate）被设置为 2000Hz，所以“gyro”线程的运行周期也为 2000Hz。

“gyro”线程里面，死循环 while(1)依次执行下列代码。由于源码较多，下列代码是经过筛选后保留的关键部分程序，能说明“gyro”线程的基本的工作方式。

```
While(1)
{
    gyro_get_data(); // 使用“spi30”设备读取陀螺仪数据并转换，自动量程
    filter_in.x = max_raw.x; // 提取瞬时角速度值
    filter_in.y = max_raw.y;
    filter_in.z = max_raw.z;
```

```

filter_out = sliding_windows_filter_3(filter_in); //将其输入加权滑动窗口滤波器

gyro.pitch = filter_out.x; //提取滤波完成的值，并根据关系转换坐标轴
gyro.roll = filter_out.y;
gyro.yaw = filter_out.z;
gyro.temperature.temp = max_raw.temp; //传感器温度

position_release_start(); //释放信号量，通知 position 线程进行姿态计算
}

```

几乎所有的传感器读取，都经历这样的一个取值，变换单位，滤波，输出的过程。其不同之处在于对传感器的配置和对滤波参数的设置上有所不同，大体的程序结构是类似的，所以不再对其他的传感器进行说明。

5.3.3 姿态算法

在飞控系统中，采用捷联惯性导航作为其基本思想，结合其他定位手段对捷联惯性导航中的误差进行修正。姿态算法部分的职责，即是将上一时刻中的飞控状态（姿态角度，位置信息等）与这一时刻中的传感器输出进行结合，并计算出这一时刻的状态。因为姿态算法拥有很高的执行周期（1000Hz），所以姿态算法的结果直接被用于控制算法的计算，以修正瞬时误差，是四轴飞行器能保持稳定的基础。

将各个传感器的数据提取后，需要通过姿态算法的计算，才能得出实际的角度值。在飞控中姿态计算使用的是四元数算法。四元数（Quaternions）是超复数中最简单的一种形式，由爱尔兰数学家 William Rowan Hamilton 在 19 世纪中期发明的数学概念。从它被发明以来，一直都没有受到大部分人的重视，即使在今天，四元数的应用也并不是特别广泛。但当其用于描述空间的旋转时，显得相当自如。实际测试也表明，用四元数来计算空间旋转，相对于方向余弦矩阵来说可以减少很多的计算量，并且可以避免奇异点问题。四元数算法虽然在高动态时，有圆锥误差，但并不需要深究它的修正方式。因为在本系统中，动态并不算高，其次相比起陀螺仪的噪声和漂移，以及修正方式来说它足够小，已经可以忽略。

四元数的形式为式 5-1：

$$ai + bj + ck + d \quad (\text{式 5-1})$$

其中四元数的定义为式 5-2：

$$i^2 = j^2 = k^2 = ijk = -1 \quad (\text{式 5-2})$$

关于四元数在姿态算法中的推导，四元数有着不一样的运算法则，这里因篇幅有限不再阐述，直接使用其变换公式。

设四元数：

$$Q = [w, x, y, z]^T$$

$$|Q|^2 = w^2 + x^2 + y^2 + z^2 = 1$$

使用旋转轴和绕旋转轴旋转的角度构建四元数：

$$\begin{aligned} w &= \cos(\alpha / 2) \\ x &= \sin(\alpha / 2) \cos(\beta_x) \\ y &= \sin(\alpha / 2) \cos(\beta_y) \\ z &= \sin(\alpha / 2) \cos(\beta_z) \end{aligned}$$

设在笛卡尔坐标系中 φ , θ , ψ 分别为绕 x, y, z 的轴旋转量。相应地在 Tait-Bryan angles 坐标系中则分别对应 roll (滚转角), pitch (俯仰角), yaw (航向角)。

根据四元数的定义可以得到欧拉角转四元数的方法, 式 5-3:

$$Q = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(\varphi / 2) \cos(\theta / 2) \cos(\psi / 2) + \sin(\varphi / 2) \sin(\theta / 2) \sin(\psi / 2) \\ \sin(\varphi / 2) \cos(\theta / 2) \cos(\psi / 2) - \cos(\varphi / 2) \sin(\theta / 2) \sin(\psi / 2) \\ \cos(\varphi / 2) \sin(\theta / 2) \cos(\psi / 2) + \sin(\varphi / 2) \cos(\theta / 2) \sin(\psi / 2) \\ \cos(\varphi / 2) \cos(\theta / 2) \sin(\psi / 2) - \sin(\varphi / 2) \sin(\theta / 2) \cos(\psi / 2) \end{bmatrix} \quad (\text{式 5-3})$$

也可得到四元数与欧拉角的转化方法, 式 5-4:

$$\begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(wx + yz)}{1 - 2(x^2 + y^2)} \\ \arcsin[(wy - zx)] \\ \arctan \frac{2(wz - xy)}{1 - 2(y^2 + z^2)} \end{bmatrix} \quad (\text{式 5-4})$$

通过以上式 5-3 与式 5-4 两个转化公式, 可以在四元数与欧拉角之间建立关系。这样即可将陀螺仪的输出量转换为四元数, 并与上一时刻的四元数相乘, 进行姿态递推, 然后再从四元数中转换成较为直观的欧拉角进行输出。这个递推的过程, 即是姿态更新的过程, 也是姿态算法的核心。

理论上, 在四元数算法中, 只要求使用三轴陀螺仪的结果, 即可完成每一步的姿态递推。但这是建立在陀螺仪没有误差的基础上, 实际应用中每一个传感器都是有误差的, 而在 MEMS 工艺的传感器上, 误差显得更加明显。在实际使用时, 温度, 加工精度, 电压波动等, 都能引起陀螺仪输出的不稳定。

因为陀螺仪的输出有许多误差, 所以飞控需要更多的传感器来修正陀螺仪误差产生的姿态估计错误。通常来说, 如果不使用全局定位的方式 (例如使用多个摄像机拍摄并识别出飞行器的姿态), 对于板载传感器而已, 修正的能力是有限的, 有的方式只能修正一个轴的误差, 有的可以修正两个轴, 但他们并不是稳定的, 有时会因为一些干扰而失效。

其中最典型的两个器件，是三轴加速度计和三轴磁阻传感器。在对他们进行说明之前，先约定四轴飞行器处于微小加速度或匀速状态。大加速度状态下，加速度计的输出将会产生较大的误导性。

将加速度计和磁阻传感器的输出值假想成一个单位向量。那么加速度计测量出来的值，可以转换成一个指向地心的向量，大小为重力加速度，而磁阻传感器可以转换为一个顺着当前地磁场磁感线的向量，大小为地磁场强度。毫无疑问，这两个向量是交差的，两个交差的向量即可确定一个包含这两个向量的平面，利用这个平面，即可修正因为陀螺仪误差引起的姿态误差。

在本设计中，是将加速度计的输出转换为 roll 与 pitch 相对地平面的旋转量，磁阻传感器则计算出当前磁北的位置，即 yaw 的旋转量。将他们与当前输出的姿态角度进行对比后，将差值记录下来，并在下一次计算四元数之前，先将其与陀螺仪输出的角增量用变参数互补滤波器进行融合后，再计算四元数。这样一来增加了修正算法，使得计算出来的姿态不会因为陀螺仪误差而发散。Roll 与 pitch 修正程序里，可变参数互补滤波器的参数由加速度计的值的稳定性决定，yaw 修正程序里，互补滤波器滤波参数固定。

5.3.4 控制算法

5.3.3 姿态算法中提到过，控制算法需要使用姿态算法的结果，即当前姿态角度来修正四轴飞行器的误差。在描述控制算法的之前，需要先引入两个定义：期望姿态，姿态误差。

定义 1：期望姿态，即由导航算法或者遥控器指定的姿态。

定义 2：姿态误差，即期望姿态与当前瞬时姿态的差值。

有了这两个定义，很容易知道，实际上控制算法的职责，是在尽可能短的时间内，将姿态误差减少到可以接受的误差范围内，使四轴飞行器向期望的姿态靠近。这一部分的工作，是由“ctrl”线程负责执行的。由于姿态算法的结果（即当前飞控的姿态角度）更新速率非常高，达到 1000Hz，所以控制算法中，也是以 1000Hz 的执行周期进行运算，这么做是为了尽量减少整个系统对姿态误差做出反应的延迟。

在实际编写控制代码之前，应该先尝试推导一下，输入与输出的关系。在四轴飞行器中，输入量即是姿态误差，输出量一般为各个电机的油门。在四轴飞行器，油门大小与螺旋桨产生的力在四轴飞行器倾角较小时为近似线性的关系。

本设计中，使用的控制算法为经典的位置式 PID，输入量为姿态误差，输出量为三个轴对应的姿态角度的控制量。

5.3.5 输出映射

在 5.3.4 控制算法中，计算结果为三个轴（roll, pitch, yaw）对应的控制量输出，这个输出并不可以直接赋值给电机。需要一个转换过程，将各个轴的输出量转化成飞行器中的各个伺服器（例如四轴飞行器中的四个电机）需要的输入量。

这个转换的过程，本文中称为输出映射。输出映射同样是由“ctrl”线程执行的，但输出映射分为多种模式，这么做是为了让飞控能兼容多种不同种类的飞行器。不同的飞行器，包括固定翼、直升机和其他特殊形状的飞行器，虽然拥有不同的伺服系统，产生的伺服效果也各不一样，但他们都有一个共同的特点，独立或配合工作时，都能控制飞行器产生三个轴（roll, pitch, yaw）的旋转运动。

输出映射的好处，是使得飞控更容易兼容各种不同种类的飞行器，只需要新建一个飞行器相关的配置信息结构体，并编写相应的输出映射函数。事实上，本设计中的飞控可以无缝地兼容了 4 轴飞行器和飞翼式飞行器，并在 2013 年初已经成功地实现飞翼式飞行器自动控制。

四轴飞行器的输出映射非常简单。四轴飞行器拥有两对旋转方向相同的螺旋桨，两对之间旋转方向不同，每一对螺旋桨分对角线分布。如图 5-5 四轴飞行器的输出映射关系。

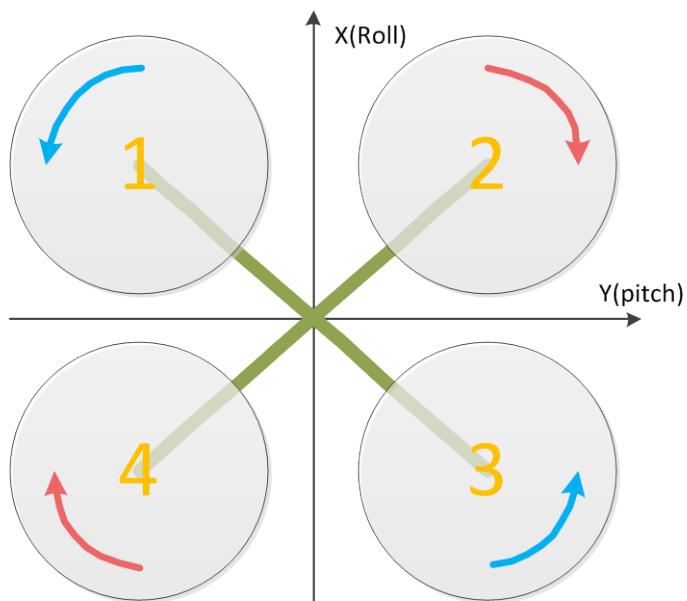


图 5-5 四轴飞行器的输出映射关系

得出三个轴的输出量后，直接按照对应关系，映射给 1、2、3、4 号电机，即可使四轴飞行器按控制算法的结果生成的修正量进行修正。

5.3.6 导航算法

四轴飞行器能自主导航的前提，是必须知道自己的实时位置。“gps”线程负责对 GPS 模块发送回来的 NEMA-183 格式的导航信息做解析，提取出有用的定位信息，比如经纬度和海拔高度。但 GPS 的定位效果较差，数米的精度对于四轴飞行器来说是比较难以处理的，所以飞控中结合加速度计对位置进行估算。在海拔高度方面，由超声波测距、气压计、加速度计以及 GPS 的海拔高度信息所共通推算，这部分计算是由“navigato”线程执行的。

5.3.5 输出映射中提出了期望姿态的定义，那么在飞行中，期望姿态的值是如何得到的呢？期望姿态的值来源有两个，它们分别对应不同的控制模式，手动模式和自动导航模式。这部分程序在“navigato”线程里。

在手动模式中，期望姿态的值直接来源于操控者手上的遥控器的控制量大小。遥控器输出多个比例通道，其中 4 个通道的值来控制飞机的姿态，分别对应油门，roll，pitch，yaw。

在自动模式中，期望姿态的值来源于飞控将要飞行的方向，通常为下一个航点的航向。特别地，在导航算法中，使用一个柱坐标来描述飞控中期望飞行的位置。柱坐标来自下一个航点与飞控的当前位置作差。相比于其他坐标系，柱坐标在描述航向和高度差方面有着先天的优势。柱坐标的夹角 (φ) 直接代表着航向 (yaw) 的修正量，高度 (z) 直接对应着油门的大小，半径 (ρ) 则用于描述距离差，使用距离差可以对速度进行控制，速度又对应着倾角 (roll 和 pitch) 的控制。于是，导航算法即可以通过柱坐标来输出类似于遥控遥控器输出（油门，roll，pitch，yaw）的值，这么一来即可在手动与自动导航模式之间无缝兼容。

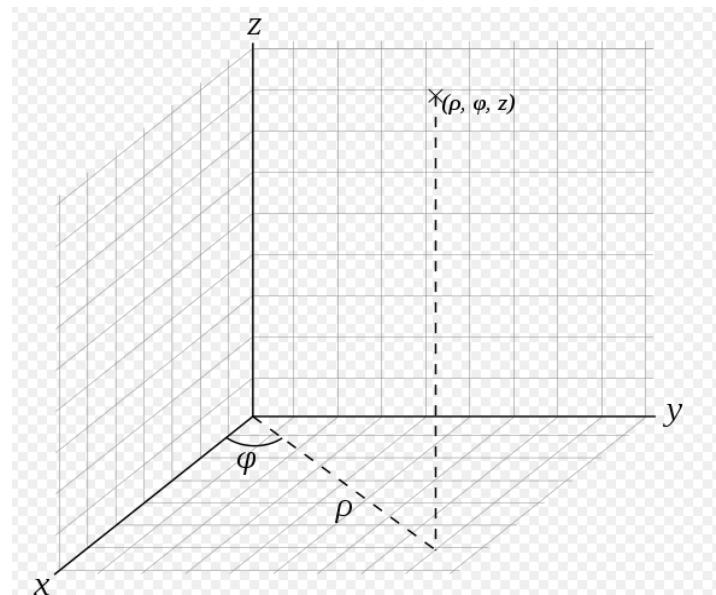


图 5-6 柱坐标

在导航部分中，同样使用了 PID 算法，输入是柱坐标表示的航点与当前坐标的差值，输量映射为 pitch, roll 轴的倾斜度和油门值，yaw 的值为切换自动飞行时的初始固定值。

5.4 电调程序设计

电调的程序设计，在本设计中使用的电调代码，并非自行设计，而是使用了与本硬件兼容的 ESC32 开源电子调速器项目的源码。经过查看 ESC32，发现其实际上是一个完全由中断驱动的调速器程序。主循环中只统计 CPU 占用率，systick 中断中负责解释和发送串口命令，ADC 中断则负责驱动电机。经过研究发现完全可以在其中加入虚拟智能终端。

于是在使用这部分开源代码时，在 ESC32 的底层插入了 RT-Thread 操作系统，将一些原先由 systick 驱动的函数改为由 RT-Thread 的线程驱动，在此之上劫持了 ESC32 的 PWM 信号更新函数，使得 ESC32 不再通过自己的硬件中断端口来读取 PWM 信号，而是由另外设计的软件来给予它新的油门的量。程序结构的包含关系在图 5-7 电调程序结构图。

这么做的好处是，可以将电调看成虚拟智能终端，挂载在 CAN Bus 总线上，使其实现更高的油门更新率和更低的延迟，此外虚拟终端可以向 CAN Bus 发送电流、电压和电机转速的值。这意味着飞控可以知道某一个电机是否正常，增加了四轴飞行器整体的安全性能。

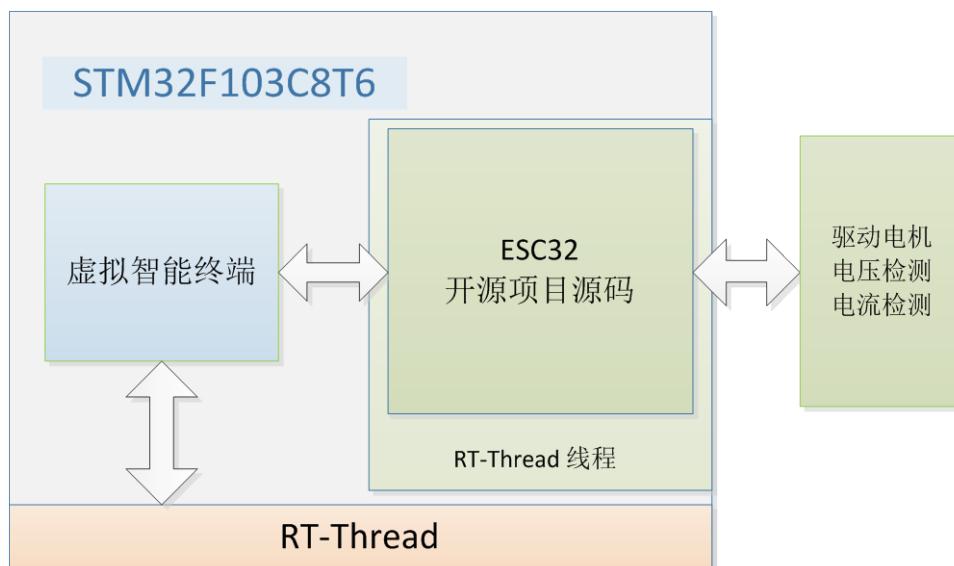


图 5-7 电调程序结构图

5.5 其他模块程序设计

由于运用了虚拟智能终端的设计理念，其他的模块包括构件模块（电调板，电源板）和通过 UEB 拓展的各个模块在程序设计上，并没有太大的区别。

这些模块有许多共同点，底层采用 STM32F103 系列微控制器，OS 层都为 RT-Thread，应用层都为终端。所以只需要为各自不同的硬件需求添加小部分硬件相关的代码，例如定时器的初始化（用于蜂鸣器），初始化 ADC(电压电流检测)，初始化各类型的总线（做接口转换）等功能。程序较为简单，所以不进行详细的叙述。

5.6 上位机

本次设计中，为了方便调试，使用了 2 种上位机进行调试。

5.6.1 Mavlink 协议简介

Mavlink 全称为 Micro Air Vehicle Communication Protocol，最初是 Lorenz Meier 为了 PIXHAWK 开源飞控项目编写的一套专门给微型空中机器人使用的通信协议。

Mavlink 协议的设计灵感来自于 CAN Bus，使用的是许可是 LGPL。这套协议是以头文件的形式存在，目前版本为 1.0.9，支持自动生成五种编程语言，分别为 C/C++语言，JAVA 脚本，python 语言，Wlua，C#。

在飞控上使用 Mavlink 协议，需要先生成 Mavlink 的头文件。使用源码中自带的基于 python 平台的 Mavlink Generator 即可很方便地生成相关头文件。Mavlink 是使用 xml 文件来描述协议中定义的参数的，所以生成头文件之前，需要编写 xml 文件，当然也可以使用自带的一些模板。本次设计中，直接使用 common.xml 进行生成，生成好后得到一个完整的 Mavlink 协议包，里面包括所有 Mavlink 支持的消息格式和类型定义。

5.6.2 QGroundControl 上位机简介

QGroundControl(QGC) 有着与 Mavlink 相似的起源，最早也是因 PIXHAWK 项目需要而设计的，现在为一个开源的微型飞行器专用的上位机。QGC 是对 Mavlink 协议支持最完整的上位机，同时也是许多开源飞控项目使用的地面控制软件。QGC 本身也是一个开源软件，遵循的开源协议时 GPL v3。

图 5-8 QGroundControl 界面为的软件截图。

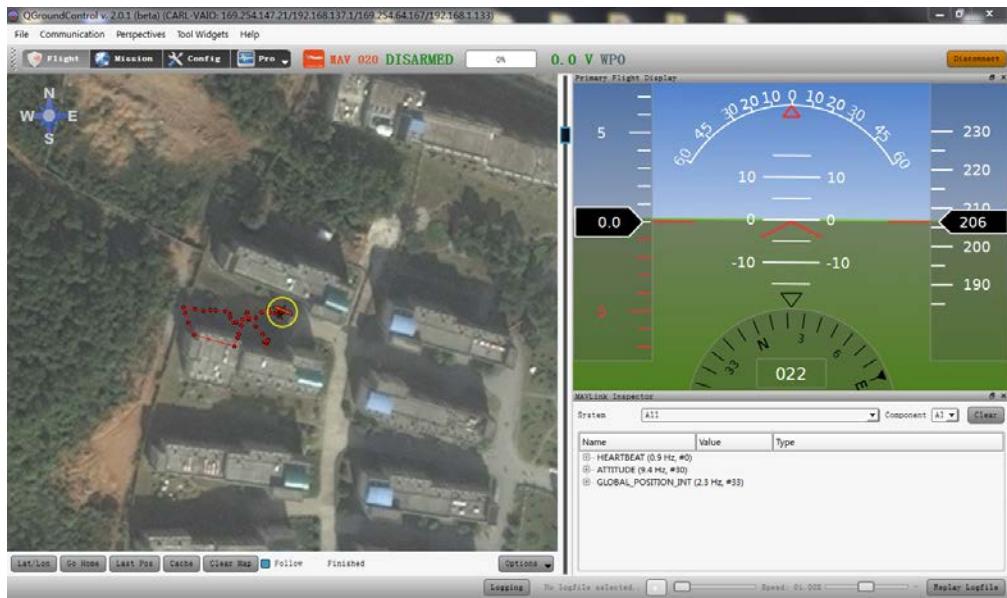


图 5-8 QGroundControl 界面

5.6.3 Deep Blue V0.3 上位机简介

Deep Blue V0.3 上位机是由作者使用 C++Builder 在 2011 年编写完成的飞控测试软件，包含一些参数的设置与显示功能。为了达到尽量接近实时的相应速度，使用一套非常简单的通信协议，降低了 MCU 的负担，同时能将许多变量的信息随着时间制成动态的图表，在调试初期起到了非常大的帮助，可以在系统的通信功能完善前，作为暂时的数据观察软件使用。但不能将所有的调试信息输出，说以在制作的后期使用较少。

显示效果如图 5-9 Deep Blue V0.3 上位机截图。

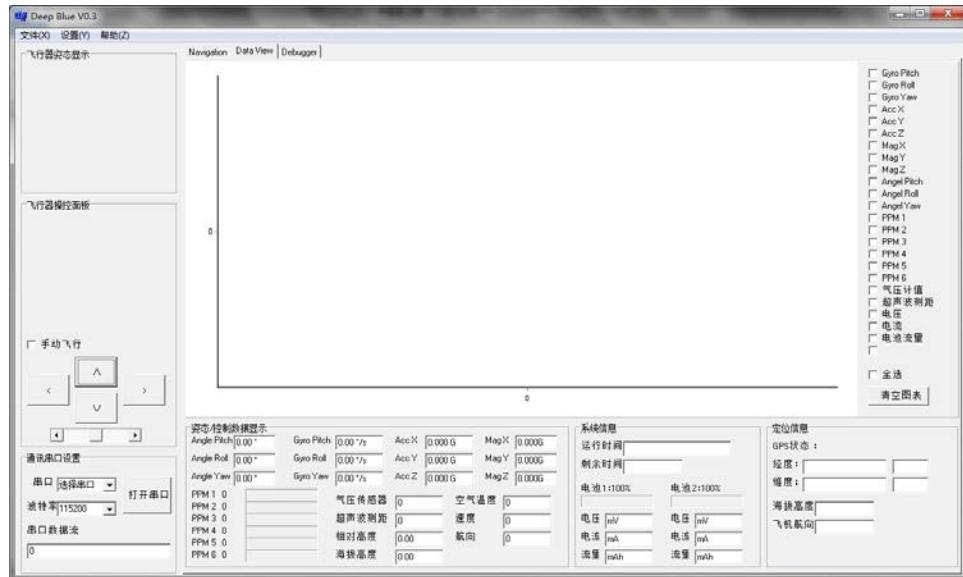


图 5-9 Deep Blue V0.3 上位机截图

6 调试与试飞

当四轴飞行器的硬件与程序都构建好后，距离起飞就不远了，剩下的最后一项工作需要完成是调参数。虽然在控制中，为了兼容不同的飞机而将控制输出统一成三个角度的控制量输出和一个油门量输出，但不同的飞机对同一个控制方式，有着不同的特性。即使是相同类型的四轴飞行器，所安装的螺旋桨不同，也会产生较大的差异，所以必须要针对每一个飞行器有一套自己固定的参数。

6.1 传感器的标定与校准

由于没有足够的实验条件，部分传感器无法完成标定的过程。

比如陀螺仪的标定需要高精度的转台，限于实验条件不足，对此不做标定，根据数据手册而言，陀螺仪本身精度较高，并且姿态计算过程中，有其他的校准方式，所以不做标定对于本系统来说影响不大。

(1) 对于加速度计来说，做简单的标定是较为容易实现的，只需在平放的情况下，测试各个轴的正反值即可算出 offset 和比例因子。

(2) 对于磁阻传感器，需要用手动全角度旋转，取得各个轴的最大值和最小值，然后进行椭球修正。

(3) 对于陀螺仪，因为实验条件限制，无法做到动态时的标定，这里只进行了静态标定。陀螺仪在静止时的各个量程中记录下输出值，即得出 offset。

(4) 对于气压计而言，生产厂家在生产时，已经做了标定，所以只需要直接取出标定的数据进行计算。

6.2 振动测试与滤波器参数选择

经过标定后的传感器已经变得较为精确，但在实际读取传感器参数的时候，读取回来的值并不能直接使用，因为此时的数据会夹杂噪声和较高频的机械振动，直接使用将会对系统的稳定性产生影响。

6.2.1 模拟振动测试平台

为了测试机架飞行时的振动对各个传感器的影响，设计中使用电源底板、尼龙柱和橡筋制作了一个临时测试支架。橡皮筋将飞控悬空，并在其下面固定一个带偏心锤的空心杯电机，当空心杯电机旋转起来后，因为偏心锤的作用，会带动飞控产生振动。通过控制空心杯电机的电压，即可调节其振动频率。

实物图如图 6-1 振动测试台实物图。

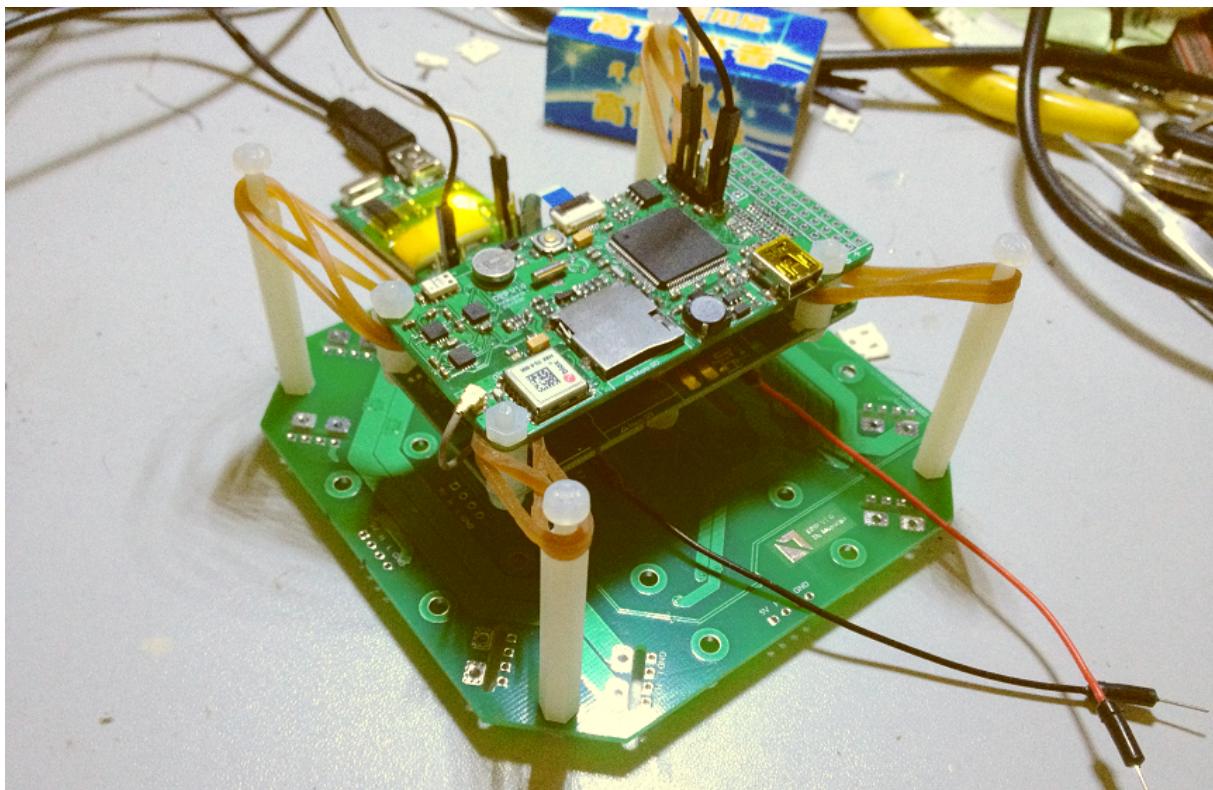


图 6-1 振动测试台实物图

6.2.2 加速度计的振动测试和滤波器参数选择

在所有传感器中，最容易产生噪声的是加速度计。加速度计的噪声来源于两个途径，第一，是由于加速度计本身的噪声，第二，是来自机体的振动。经过实际测试，来自机体振动占了主导地位，恶劣情况下，其峰峰值甚至可以超过 1G，计算出来的角度已经没有规则可言。

所以必须要对加速度计的原始数据进行滤波后，才能用于姿态计算和航位计算。关于滤波器的选用，这里选用的是移动窗口加权平均滤波器。不用卡尔曼滤波器，是因为它只关心上一个值，在高动态的噪音下，不能发挥较好的效果。对于本系统中的移动窗口加权平均滤波器而言，为了简化计算，权值为线性的，所以滤波的参数只需要调整滤波窗口的宽度即可。

使用自制振动台测试不能发出精确的振动效果，这里仅仅是作为一个测试手段进行使用，对于其发出的频率和振幅没有进行检测。

测试数据为飞控通过 microSD 卡记录下来后，电脑从 microSD 卡中读取数据并放入 Microsoft Excel 中进行观察的。这么做是为了保证不会漏掉任何一个数据，方便利用 Excel 强大的数据分析功能对滤波的效果进行测试。

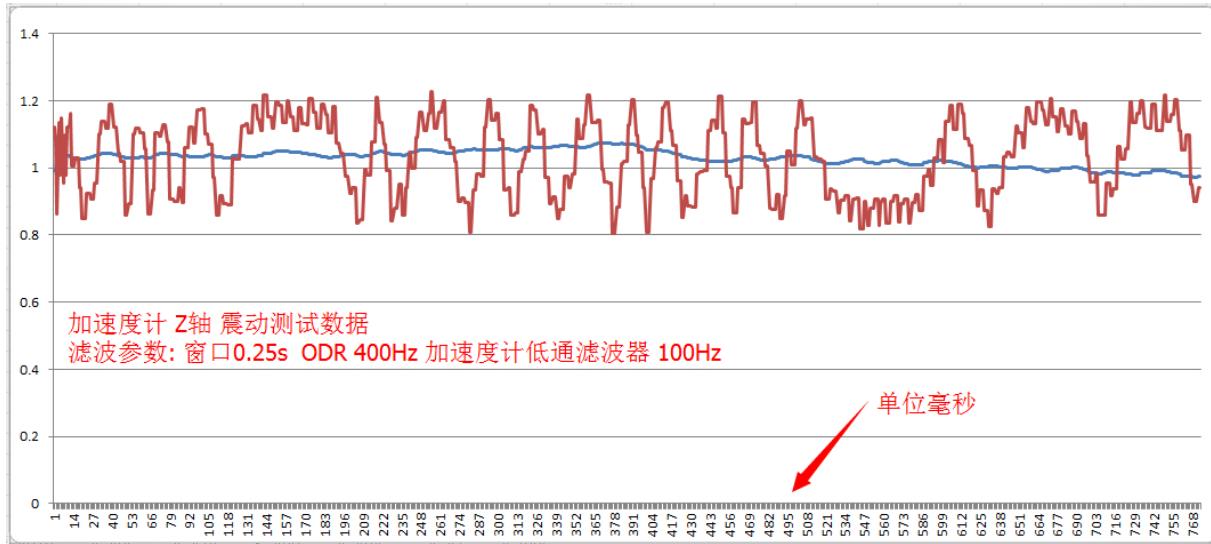


图 6-2 加速度计滤波效果测试

图 6-2 加速度计滤波效果测试 为滤波前与滤波后的图形对比。滤波窗口大小为 0.25 秒，滤波器输入量为加速度及 Z 轴的原始值，输出量为滤波后的值。

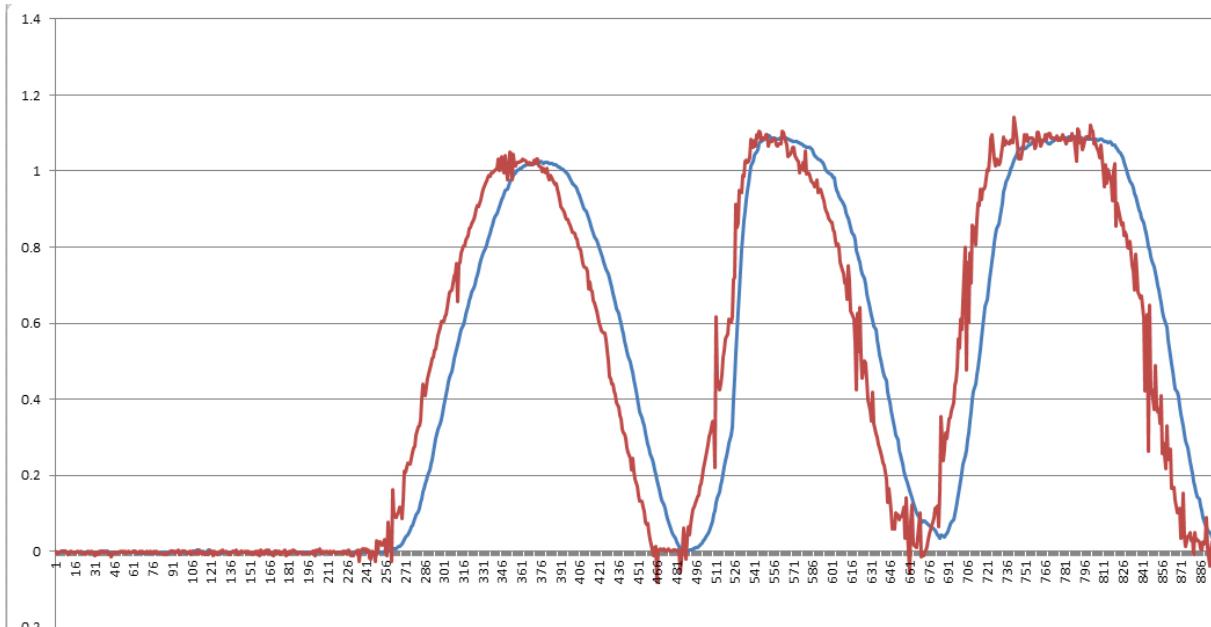


图 6-3 加速度计滤波延迟

图 6-3 加速度计滤波延迟为 0.25 秒的窗口下，横坐标的单位为 10ms。加速度计的实时值与滤波值之间的实际差，可以看出，大约在 150ms。

实际效果不错，最终决定用 0.25 秒的窗口宽度。

6.2.3 陀螺仪的振动测试和滤波器参数选择

虽然 MEMS 工艺的陀螺仪相对其他陀螺仪而言对振动较为敏感，但在实际使用过程中，陀螺仪的值较为平稳。同时，姿态算法其实是一个积分的过程，积分的过程本身会有滤波器的效果，所以陀螺仪的滤波看似并不必要。但在控制算法中，陀螺仪的输出直接用于 PID 控制算法中的 D 的计算，如果突变太快，有可能会影响系统的稳定性，所以保证一个平稳的变化率是非常有意义的。

四轴飞行器是一个整体，相对而言本身转动惯量较大，不太可能有高频率大幅度的转动变化。

这里只是测试陀螺仪在大角速度变化时的滤波效果。

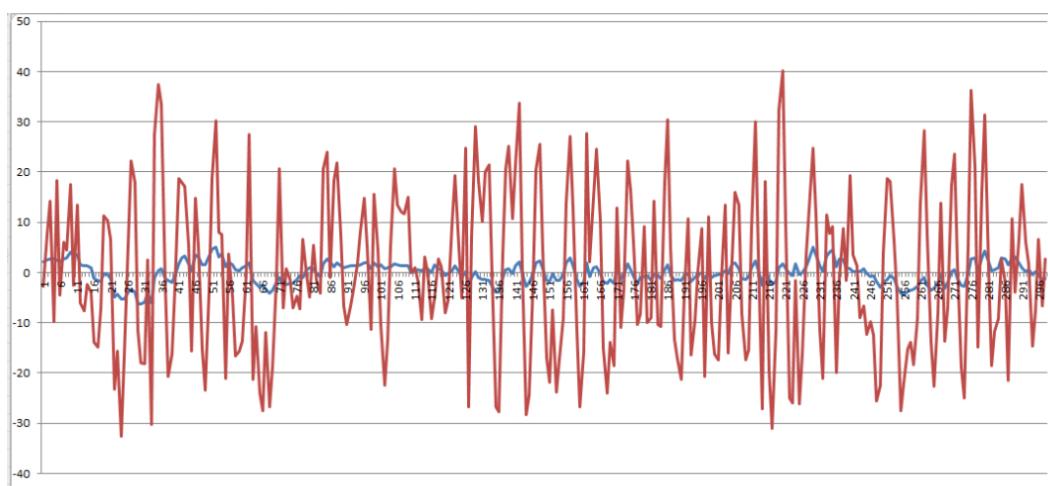


图 6-4 陀螺仪滤波器测试

图 6-4 陀螺仪滤波器测试中横轴的分度是 1ms，滤波器的窗口大小为 32 个采样值，相当于 16ms 的窗口宽度。红线为滤波之前 x 轴的输出值，蓝线为滤波后的值。在这种情况下，滤波器能有效滤除高频的转动变化的干扰。

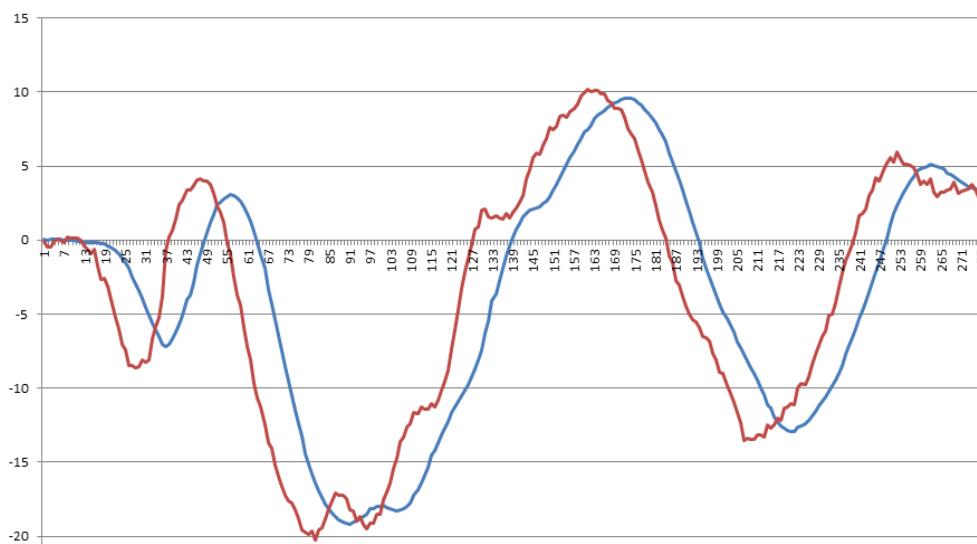


图 6-5 陀螺仪的滤波延迟

图 6-5 陀螺仪的滤波延迟 中可看出，在滤波窗口为 32 个采样值的情况下，陀螺仪的输出大约延迟 11ms。实际效果不会有太多负面影响，最终决定用 32 个采样值的窗口宽度。

6.3 PID 参数调整

PID 参数的调整方法，与传统方式是一样的。例如在控制算法中，调整俯仰的 PID 参数。用手握紧飞行器，但保持腕关节放松，先调整 P，从小到大，当飞行器接近振荡时，记录下当前的 P 值。然后再调整 D 参数，用手让四轴飞行器做快速俯仰运动，感到四轴飞行器有明显的抵抗时记录下 D 参数。依据这两个初步确定的参数，再结合实际飞行时的反应继续进行微调。最后加上 I 参数，当其出现缓慢振荡时，记录下当前的 I 参数并将其乘以 0.7 以后，得到的 I 参数即为所需要的 I 参数。这样就确定完成 PID 的三个参数了。

其他部分的程序如用到 PID 调参，均类似于以上例子。

6.4 试飞

试飞过程较为简单，当所有工作都准备好后，四轴飞行器第一次试飞就顺利离开地面，在经过几次调试 PID 参数后，已经能稳定飞行。实际飞行灵活轻便，可以挂上摄像机等负载，实现定高定点飞行等功能，达到设计要求。如图 6-6 为飞行器与地面控制站，图 6-7 为实际飞行时拍摄的照片。

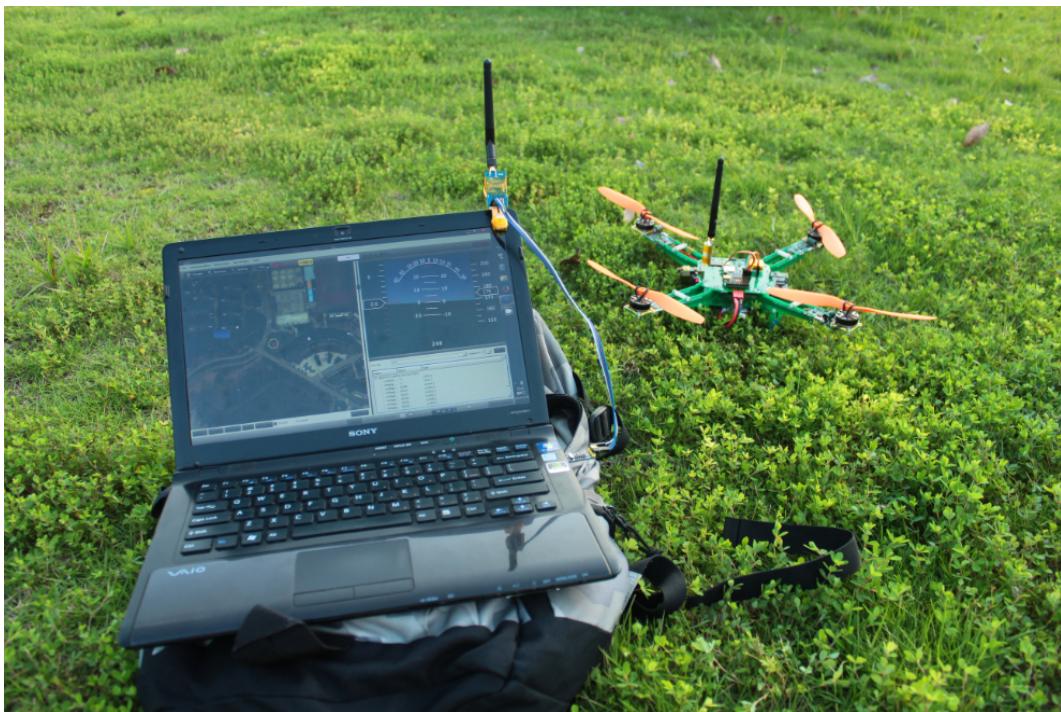


图 6-6 地面站与飞行器



图 6-7 实际试飞中

7 总结与展望

7.1 总结

在这次四轴飞行器的设计中，飞行器虽然能平稳飞行，但这是在小的动态情况下。在高动态中，本设计并不能使飞行器平稳飞行，甚至可能致坠机。另外，四轴飞行器不能在室内自动进行导航，因为缺乏足够的定位手段（例如，摄像头—激光地图探测器，对地摄像头定位等）和 SLAM 的能力。

如果能在后续的设计中，实现这两项功能，将使四轴飞行器可以进行室内导航，这意味着它的使用范围将大大增强，使用难度将显著降低，智能性也更进一步。

7.2 展望

四轴飞行器作为非常灵活实验平台，将有更多的发展空间。使用先进的算法和伺服机构，让飞行器本身能完成高难度的工作，并拥有非常强的鲁棒性，在自身损坏的情况下能安全返航。当四轴飞行器变得更完美时，将广泛应用与军事及民用领域，造福人类。

谢 辞

在完成学位论文之际，首先要感谢我的毕设指导老师张应红，他在我读本科期间，在生活和学习上都给予我很多的关心和帮助。我的进步和成长离不开导师的谆谆教诲和指导。

在论文写作期间，张老师严格要求论文质量，认真仔细地审阅论文，给出很多宝贵意见。他渊博的知识、严谨的治学态度、求实的工作作风都深深地影响着我，激励着我。在此，再次对张老师表示诚挚的感谢和崇高的敬意！

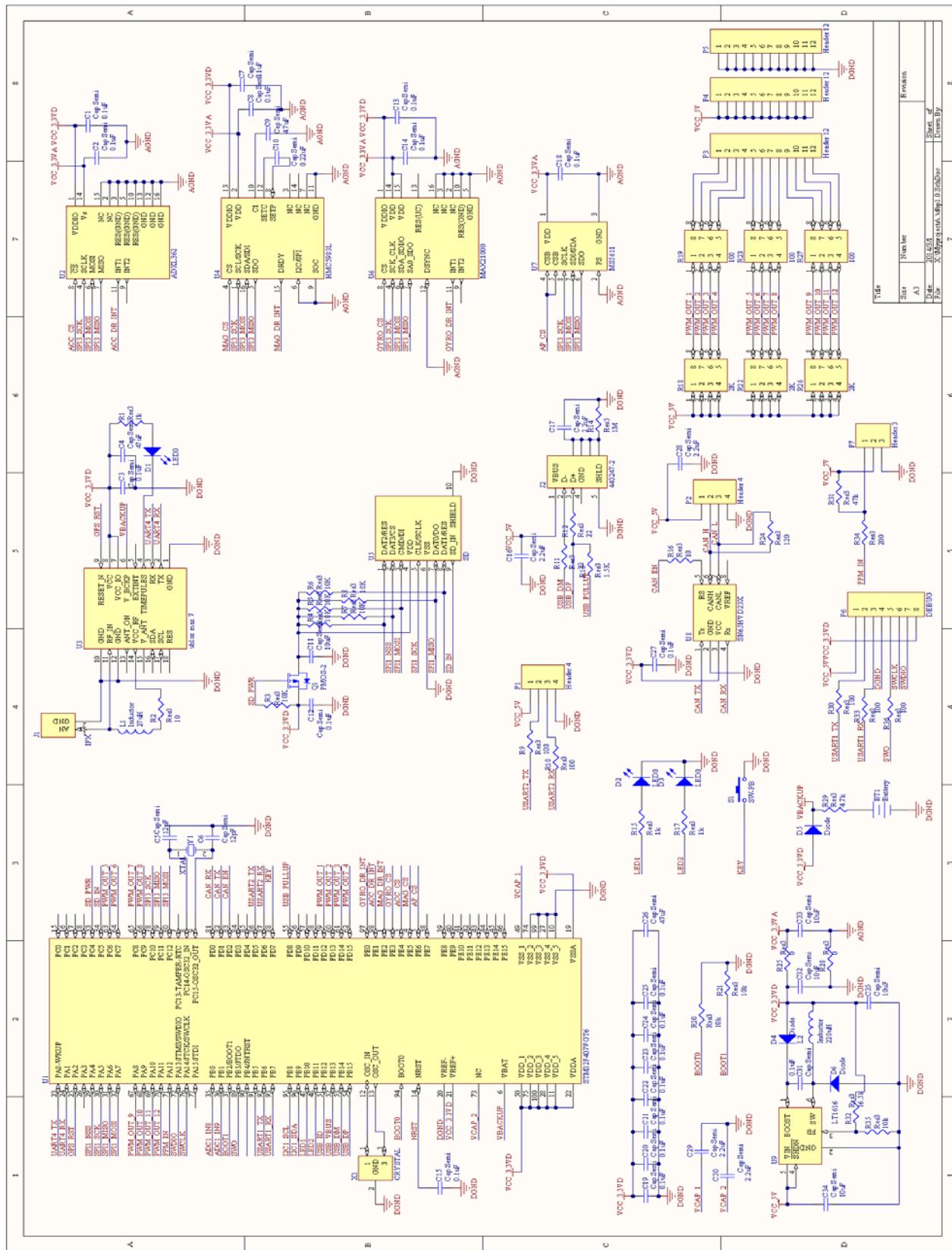
最后，还要感谢我的父母家人和朋友。感谢他们对我的支持和鼓励，让我顺利完成学业。

参考文献：

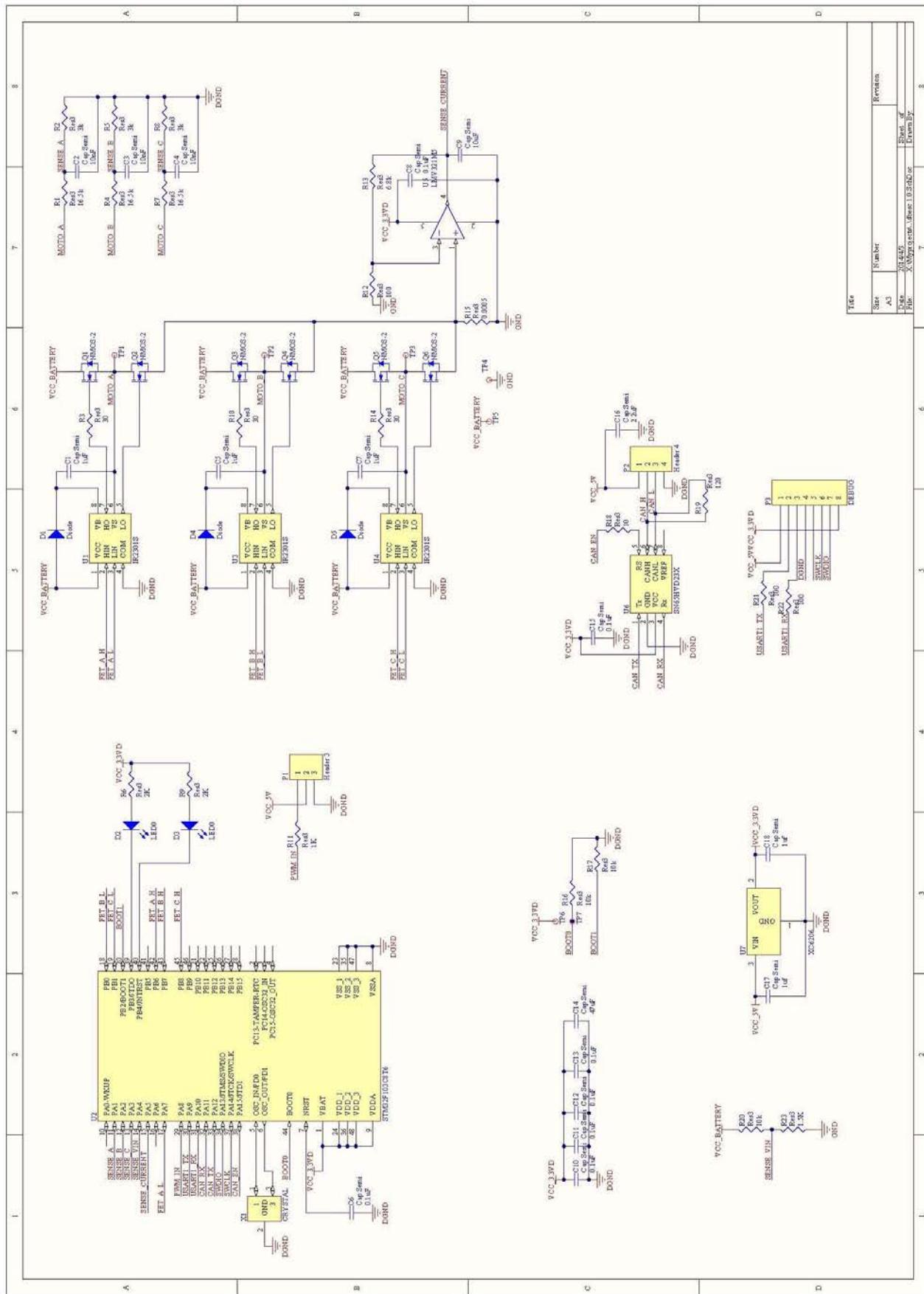
- [1] 胡仁喜, 孙明礼. ANSYS 13.0 电磁学有限元分析从入门到精通[M]. 机械工业出版社, 2011.12.
- [2] 周权, 黄向华. 四旋翼微型飞行平台姿态稳定控制试验研究[J]. 传感器与微系统, 2009, 28 (5) : 72-79
- [3] 刘刚, 彭荣群. Protel DXP2004SP2 原理与 PCB 设计[M]. 北京: 电子工业出版社, 2007: 283-320.
- [4] 周宾, 杨道业, 汤光华, 许传龙, 王式民. 圆环式静电传感器的动态灵敏度分析[J]. 中国电机工程学报, 2008, 28 (14) : 44~49.
- [5] 阿雷尼, 韦伯斯特. 传感器和信号调节(张伦) [M]. 北京: 清华大学出版社, 2003.
- [6] Salih A L. Modelling and PID controller design for a quadrotor unmanned air vehicle [J]. IEEE AQTR, 2010, 39(5):697-699.
- [7] Mian A, Wang Daobo. Nonlinear Flight control strategy for an underactuated quadrotor aerial robot [J]. IEEE AQTR, 2008 (1):8-16.
- [8] 周淑阁. FPGA/CPLD 系统设计与应用开发[M]. 北京: 电子工业出版社, 2011.
- [9] S. A. Cambone, K. J. Krieg, P. Pace, and L. Wells, ‘USA’s Unmanned Aircraft Roadmap, 2005-2030, National Defense, 2005.
- [10] B. L. Stevens and F. L. Lewis, Aircraft Control and Simulation, John Wiley & Sons, Second edition, 2003.
- [11] J. Pappalardo, “Unmanned aircraft roadmap reflects changing priorities,” National Defense, vol. 87, no. 392, pp. 30, 2003.

附录

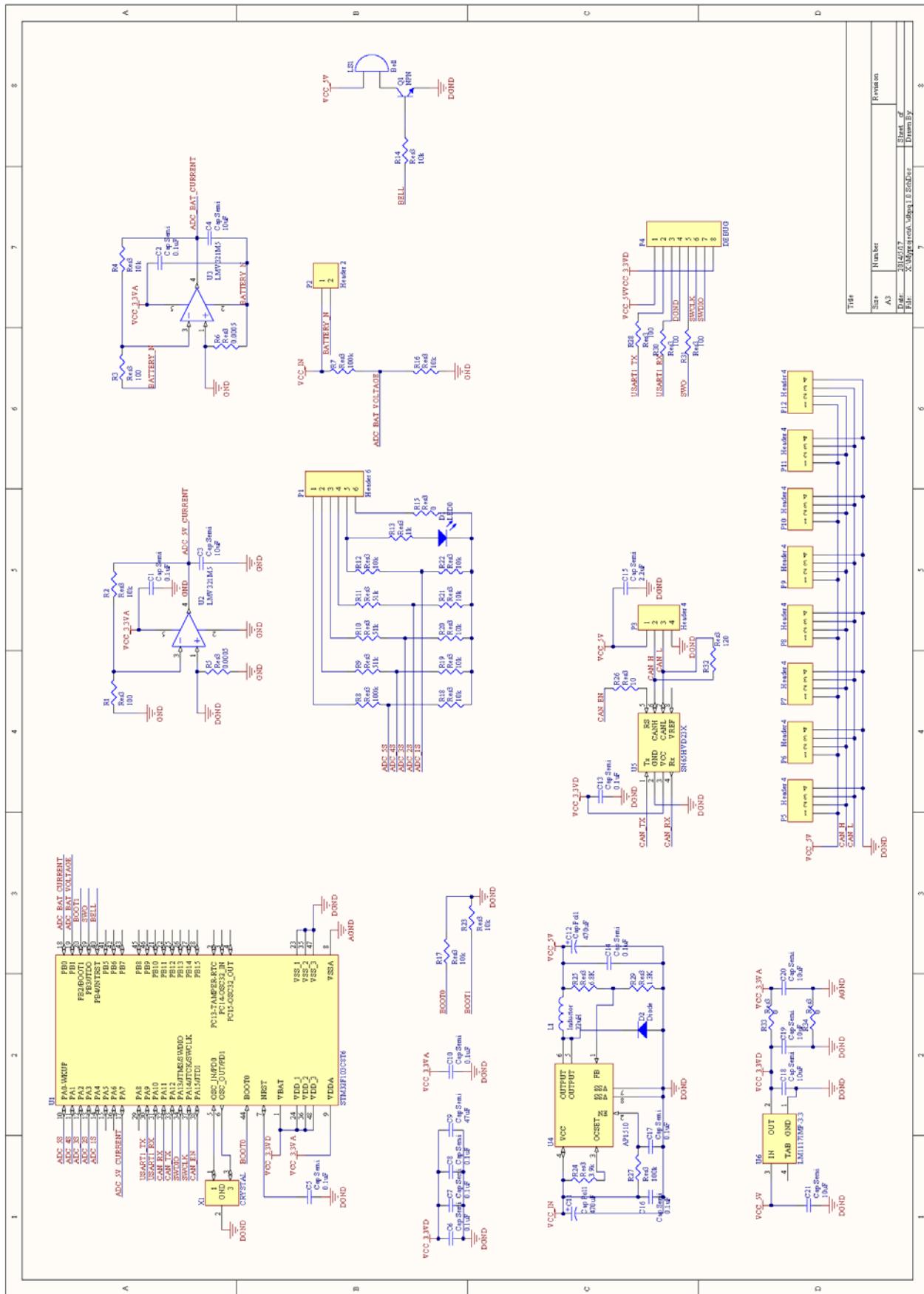
1. 飞控原理图



2.无刷电调原理图



3. 电源底板原理图



4. UBE 原理图

