基于STM32的四旋翼飞行器

作 者 姓 名： 张君豪

指 导 教 师： 陈刚 副教授

单 位 名 称： 计算机科学与工程学院

专 业 名 称： 通信工程

东 北 大 学

2017年6月

**The Quadrocoptor Based on STM32**

by ZHANG Junhao

Supervisor ：Associate Professor CHEN Gang

**Northeastern University**

**June 2017**

# **毕业设计（论文）任务书**

|  |
| --- |
| **毕业设计（论文）题目：**  **基于stm32的四旋翼飞行器** |
| **设计(论文)的基本内容：**  本毕业设计利用单片机STM32设计四旋翼飞行器，并完成整个系统的调试，并在该平台上移植并测试机器人操作系统ROS。  (1)掌握stm32的使用及程序调试方法  (2)阅读有关微型飞行器算法及控制理论的文献，设计飞行器的控制器  (3)设计整个系统的软硬件系统，并完成整个系统的调试  (4)并在平台上移植并测试机器人操作系统ROS。 |
| **毕业设计（论文）专题部分：**  **题目：**  **设计或论文专题的基本内容：** |
| **学生接受毕业设计（论文）题目日期**  **第1周**  **指导教师签字：**  **2017年3月5日** |

基于STM32的四旋翼飞行器

摘 要

四旋翼飞行器是一种结构新颖，体型较小，性能卓越的可垂直升降多旋翼飞行器。由于其具有其结构简单成本低，便于维护和制造等优势，因而在近几年军用和民用无人机领域占据着越来越重要的地位。机器人操作系统(Robot Operating System,ROS)是一种得到广泛应用的开源机器人操作与控制系统软件框架，是用于机器人的一种开源的后操作系统，能够实现在不同的机器人上复用代码，避免开发者重复劳动。本文旨在设计一款基于STM32的四旋翼飞行器并将机器人系统ROS移植到STM32平台，实现飞行器与ROS的通信。

本文首先介绍了四旋翼飞行器及机器人操作系统的发展过程及前景。然后根据四旋翼飞行器系统功能以及后续研究需要，设计了系统硬件原理图以及PCB，搭建了飞行器硬件平台。本文采用四元数姿态解算算法对姿态数据进行解算，同时采用串级PID控制算法对飞行器的姿态进行控制，从而实现飞行器的稳定飞行以及有效控制。在此基础上，rosserial功能包与原有程序相结合，将四旋翼飞行器作为机器人操作系统的节点，实现话题的发布与订阅，将飞行器姿态信息实时地传输给ROS。最后，对本文的不足和缺点进行了反思和总结，提出了下一步的研究方向。

**关键词**：四旋翼飞行器，机器人操作系统，四元数互补滤波，串级PID，rosserial

The Quadrocoptor Based on STM32

Abstract

**Key words**：

目 录

**毕业设计（论文）任务书** I

摘 要 II

Abstract III

第1章 引 言 1

1.1 课题研究背景 1

1.1.1 四旋翼飞行器介绍 1

1.1.2 机器人操作系统介绍 1

1.2 课题研究目的及意义 2

1.3 本文的内容及主要工作 3

第2章 飞行器硬件设计 5

2.1 四旋翼飞行器基本结构和工作原理 5

2.2 系统总体设计 6

2.3 飞控板硬件设计 7

2.3.1 微控制器选型及最小系统设计 7

2.3.2 传感器 8

2.3.3 无线及遥控通信 9

2.3.4 电源及扩展电路 10

2.3.5 飞控板PCB设计 14

2.4 遥控板硬件设计 15

2.4.1 电源电路设计 15

2.4.2 遥杆电路及微调电路 16

2.4.3 遥控板PCB设计 17

第3章 飞行器姿态解算算法 18

3.1 姿态解算的目的及意义 18

3.2 飞行器姿态表示方法 18

3.2.1 欧拉角 18

3.2.2 四元数表示法 19

3.3 姿态解算算法实现 21

3.3.1 互补滤波算法 21

3.3.2 四元数姿态解算算法实现 22

第4章 飞行器控制算法与参数整定 26

4.1 PID控制器 26

4.1.1 PID控制器介绍 26

4.1.2 单级PID控制算法 27

4.1.3 串级PID控制算法 27

4.2 PID参数整定 28

4.2.1 上位机软件介绍 29

4.2.2 PID参数整定过程及结果 30

第5章 飞行器与机器人操作系统通信 32

5.1 ROS的总体框架及基本概念 32

5.2 rosserial通信协议简介 34

5.2.1 rosserial协议包格式 34

5.2.2 主题协商与同步 34

5.3 基于rosserial功能包的节点通信 35

5.3.1 STM32与ROS通信的实现 35

第6章 结 论 38

参考文献 39

致谢 40

# 引 言

## 课题研究背景

### 四旋翼飞行器介绍

自主携带动力装置，能够实现自主飞行且不需搭载人员的飞行器称为无人飞行器（UAV）。随着微电子技术以及自动控制技术的不断发展，无人飞行器由于其无需载人、安全系数高等优点广泛应用于军用领域和民用领域[1]。

根据飞行平台结构的不同，无人飞行器可分为固定翼飞行器、直升机和多旋翼飞行器[2]。固定翼飞行器，顾名思义就是机翼形状固定，相对于机体不会产生相对运动的飞行器，其优点是飞行时间长、飞行效率高以及载重量大，缺点是起飞和降落都需要特定的条件——跑道。从1903年，莱特兄弟发明第一架飞机到现在，固定翼飞行器的技术已经相当成熟，并在民用航空领域以及军事领域展现出了其优越的性能。

直升机和多旋翼飞行器可以统称为旋翼飞行器。与固定翼飞行器不同的是，旋翼飞行器在空中飞行的升力由旋翼与空气进行相对运动产生反作用力获得，典型的旋翼飞行器即为我们熟知的直升机。旋翼飞行器具有空中悬停以及垂直起降等固定翼飞行器所无法比拟的优点，因此有着更加广阔的应用前景。

微型四旋翼飞行器是一种体型较小、重量较轻（尺寸在15cm以下，重量不足500g）的多旋翼飞行器，其动力装置对称分布在机体的前、后、左、右四个方向[3]。随着现代计算机技术及控制理论飞速发展，微型四旋翼飞行器也得到了极大地发展。凭借简单的结构以及灵活独特的飞行方式，微型四旋翼飞行器吸引了诸多科研工作者的注意，并逐渐成为无人飞行器领域的焦点。

### 机器人操作系统介绍

开源机器人操作系统(Robot Operating System，以下简称为ROS)最早是斯坦福大学人工智能实验室与谷歌合作的项目，2008年起由谷歌前员工Scott Hassan创立的机器人技术公司Willow Garage负责维护，其他20多家公司共同开发，自2013年起开始交由开源机器人基金会（OSRF）负责管理。

ROS是一种开源的用于机器人的后操作系统。它具有类似于操作系统的功能，包含硬件抽象描述、底层设备控制、共用功能的实现、进程间的消息传递、程序功能包管理，它也提供一些用于从获取到运行多机整合程序的工具程序和库，来帮助软件开发者创建机器人的应用程序，因此ROS也可以被称为次级操作系统[4]。

ROS建立的最初动机是提高代码的可重用率。ROS是一种分布式模块化的开源软件框架，该框架利用面向服务（SOA）的软件技术，通过标准的TCP(UDP)/IP协议将节点间数据通信解耦，可以使执行程序单独进行设计，松散地、实时地组装起来[5]。

ROS的主要优点总结起来有以下几条：

（1）功能化与模块化。ROS的功能化与模块化主要归功于其使用了点对点设计以及服务和节点管理器等方式，可以将计算机视觉和语音识别等需要庞大计算能力支撑的功能带来压力分散化，这种设计可以解决未来服务机器人发展遇到的困难。

（2）不依赖编程语言。ROS使用了一种语言中立的接口定义语言（language-neutral interface definition language，IDL）来实现各模块之间的信息传递，从而可以使得开发者利用C++、Python和Lisp等多种语言在ROS中进行点设计。

（3）精简与集成。ROS不修改用户的main()函数。所有代码可以被其他的机器人软件使用。其优点是ROS很容易和其他机器人软件平台集成。在计算机视觉算法方面，ROS已经与OpenCV实现集成。在驱动、导航和模拟器方面，ROS已经与Player系统实现集成。在规划算法方面，ROS已经与OpenRAVE系统实现集成。

（4）便于测试。ROS拥有一个名为rostest的内建单元/集成测试平台，它很容易集成调试和分解调试。

（5）规模化与标准化。ROS适用于大型运行系统和大型程序开发。在ROS中，在机器人系统设计过程中经常用到的数据结构和接口被定义成为了标准库的形式，方便节点的复用。

（6）开源。ROS遵从BSD（Berkeley Software Distribution）协议，对个人和商业应用以及及修改完全免费。

## 课题研究目的及意义

近年来，MEMS(Micro-Electro-Mechanical Systems，微型电子机械系统)及飞行控制理论不断发展，与高速低功耗嵌入式微处理器结合使用，使得微型四旋翼飞行器的开发和研究有了重大的突破。微型四旋翼飞行器潜力和使用价值逐渐被发掘，在近几年的时间里，引起了国内外众多研究人员、飞行器爱好者以及开发者的主意，而且逐渐的“飞入寻常百姓家”。国外典型的四旋翼飞行器有：由RC公司研发的由无线遥控器以及控制系统组成的Draganflyer四旋翼飞行器以及斯坦福大学在其基础上，添加了超声波测距传感器、GPS定位导航以及蓝牙通讯等模块并将原来的三轴陀螺仪替换为更专业的惯性测量单元(IMU)；除此之外，各种各样的网络开源四旋翼飞行器也层出不穷，如KK飞控模块、MK飞控模块和MWC飞控模块，随之出现的还有各种各样的用于飞行器调试的上位机软件[6][7]。而在国内，四旋翼飞行器吸引了国内高校及科研机构的目光并取得了一定的成果：国防科技大学自主设计了一款飞行器，并对其进行动力学建模，不仅使用Backstepping算法对其进行姿态控制器的设计，同时利用了Lyapunov方法对其进行稳定性分析，最后通过仿真实验验证了算法的有效性[8]。在商业领域，四旋翼飞行器产业蓬勃发展，如大疆公司的phantom系列、筋斗云系列多旋翼飞行器占据了国内多旋翼飞行器市场的半壁江山，被广泛应用于航拍监控等领域。

ROS自问世之初便由于其代码可重用率高等诸多优点而备受关注，被广泛应用于机器人开发以及人工智能之上。国内不少科研机构都基于ROS进行了研究，如左轩尘等在基于ROS平台的基础上，设计了操作者使用手臂姿态和语音控制的空间机器人人机交互系统，并采用双虚拟机械臂反馈的方法，从而避免视频反馈带来的延时感，实现了操作者在虚拟现实环境中对空间机器人的遥操作；南京大学佘元博基于ROS平台，提出了LAMDA-ROB-2智能交互型移动机器人架构方案以及自然人机交互系统设计方案，并且提出了自主导航系统设计方案，能够通过对环境地图上的坐标进行语义关联标注，使得自然人机交互系统中的语义命令能够与自主导航系统中的地点进行匹配[9][10]。

ROS的应用目前主要集中在性能较高的处理器之上，而基于广泛应用于各行各业的嵌入式微处理器的开发基本还是一个很大的空白。本文主要就是设计一款基于STM32嵌入式微处理器的微型四旋翼飞行器，采用四元数互补滤波算法及串级PID控制算法，使其能够平稳飞行。并以此为平台，实现STM32与ROS的通信，将飞行器的姿态数据实时地传输至ROS，为后续ROS在嵌入式微处理器上的应用提供参考。

## 本文的内容及主要工作

本文主要介绍微型四旋翼飞行器的硬件设计，飞行器姿态解算算法设计以及飞行器控制算法的设计，最后介绍了飞行器与ROS通信传输姿态信息的实现。

各章的内容组织如下：

第2章介绍四旋翼飞行器硬件平台设计，包括飞控板和遥控板硬件的选择以及原理图和PCB设计。

第3章主要介绍四旋翼飞行器姿态解算算法设计，首先介绍飞行器姿态结算的目的及意义，介绍了飞行器姿态的表示方法，包括欧拉角和四元数表示法，然后介绍了四元数姿态解算的算法实现。

第4章主要介绍四旋翼飞行器串级PID控制算法以及利用上位机对PID参数的调试。

第5章介绍了四旋翼飞行器与ROS的通信，并将姿态信息实时传输到ROS上的实现方法。

第6章对毕业设计及论文进行总结，以及下一步的工作方向，指出论文的不足及需要改进的地方。

# 飞行器硬件设计

## 四旋翼飞行器基本结构和工作原理

四旋翼无人机的结构主要由四个等长机臂组成的机架、电动机、螺旋桨等部分组成，按前进方向的位置将四旋翼无人机通常分为十字形、X形以及H形，论文采用最常用的X形布局，见示意图2.1，四个电机分别安装在机臂端点的安装座上，电机上安装螺旋桨，因电机在转动的过程中会产生转矩，因此相邻两个电机转向必须相反，而对角线上的两个电机的转动方向必须相同，对电机的转速进行控制便可以实现空间的6个基本运动（分别沿3个坐标轴的平动和转动），即偏航运动（绕Z轴旋转）、垂直运动（沿Z轴运动）、俯仰运动（绕X轴旋转）、横滚运动（绕Y轴旋转）、倾向运动（沿X轴运动）、前后运动（沿Y轴运动）。

电机2

电机4

电机1

电机3

x

y

z

前

θ

ψ

ᶲ

图 2.1 四旋翼飞行器X形示意图

偏航运动和垂直运动：当四旋翼无人机保持对角线电动机转速相同，相邻电动机转速存在差异的情况，电动机转动时产生转矩的差异使无人机实现偏航运动，如图2.1，当电机2和电机4的转速大于电机1和电机3时，使得无人机实现绕Z轴的顺时针方向偏航。使四个电机的输出功率发生相同的改变便可以使无人机实现Z轴方向的垂直运动。

俯仰运动和横滚运动：使电动机2、3的转速变化量相同，电动机1、4变化量相同，两变化量不同即可以实现无人机俯仰运动，横滚运动与俯仰运动原理相同。

前后运动和倾向运动：为了能够使四旋翼无人机能在水平面内运动，需在水平面内对无人机施加一定的外力，使无人机横滚和俯仰，无人机发生一定量的倾斜，从而使螺旋桨的升力在水平方向产生分量，实现无人机水平运动。

## 系统总体设计

系统硬件的总体设计主要包括遥控板和飞控板两个部分，遥控板采用常见羊角把游戏手柄的外形设计，控制输入采用四向摇杆，无线数据传输采用2.4G无线模块；飞控板采用控制处理核心和机架一体的设计。遥控板主要实现的功能包括ADC的采集和数据的无线发送以及利用串口与ROS进行通信。飞控板的功能主要包括无线数据的发送与接收、自身姿态数据的获取与实时解算、电机PID增量的计算和电机的驱动。整个四轴飞行器系统包括人员操作遥控端和飞行器控制端，遥控端主控制器STM32通过ADC外设对摇杆数据进行采集，把采集到的数据通过2.4G无线通信模块发送至飞控端，同时接收飞控板发送来的姿态信息，并通过串口将姿态信息实时地发送给ROS。飞控板的主要工作就是通过无线模块进行控制信号的接收，并且利用惯性测量单元获得系统实时地加速度和角速度原始数据，并且根据这些数据解算出当前的系统姿态，然后根据遥控板发送的目标姿态和当时的姿态差计算出PID电机增量，最后通过调节PWM占空比进而改变电机转速，使得飞行器能够稳定飞行[11]。系统的总体设计图如图2.2所示。

摇杆

ADC

STM32处理器

串口

机器人操作系统

中断

ADC

IO

按键输入

电位器

LED

SPI

无线模块

无线模块

SPI

STM32处理器

ADC

ADC

2.4G

ADC

IO

电池电压

LED

遥控板

飞控板

MPU6050

电机

图 2.2 系统总体设计框图

## 飞控板硬件设计

### 微控制器选型及最小系统设计

在四轴飞行器飞行控制器中，传感器电信号的采样、转换，信号数据的融合，以及控制器的运算都通过微控制器的CPU及其外设实现。在飞行控制器工作时需要大量的运算，尤其在IMU运算与控制算法运算时，会涉及到大量的矩阵与乘除法运算，因此在微控制器选型时应选择运算性能强、运算速度快的微控制器。同时，飞行控制器中的各种模块需要不同的接口与外设进行采样，如电池电压采样需要使用A/D转换单元，惯性测量单元IMU数据采集需要使用I2C总线进行数据通信，因此选择微控制器应具有丰富的外设以简化电路设计，提高硬件稳定性。在达到飞行控制系统控制性能要求的前提下，综合满足其重量、体积、功耗和成本的需求，微控制器的选型应考虑高性能、低功耗、外设丰富的高性能单片机。

STM32F103系列单片机是由世界知名的半导体公司意法半导体所开发的STM32系列单片机的增强型产品。STM32F103基于Cortex-M3内核设计，是新型的32位嵌入式微控制器。Cortex-M3是一个32位的ARM内核，采用ARMv7-M架构，支持16位Thumb指令集和基本的32位Thumb-2指令集架构，集成硬件除法器(2-12个周期)和单周期(32x32)硬件乘法器，支持饱和数学运算。

综合外设需求及性价比，选择STM32F103CBT6作为主控芯片，其稳定工作频率可达72 MHz，拥有128KB的闪存可选程序存储器，20KB的SRAM，2个12位模数转换器（多达16个输入通道）7通道DMA控制器，37个快速I/O端口，串行单线调试（SWD）和JTAG接口调试模式，7个定时器，2个I2C接口（支持SM-Bus/PMBus），3个USART接口（支持ISO7816接口、LIN、IrDA接口和调制解调控制），2个SPI接口（18Mb/s），CAN接口（2.0B主动），USB2.0全速接口[12]。单片机最小系统电路图如图 2.3所示。其中，Y1为16MHz单片机主晶振，C2、C4为起振电容，以保证其正常振荡；R1、C1以及K2组成复位电路，实现单片机的瞬间上电以及按键上电；R2、R3用于设置程序启动方式（R3在最后的焊接过程中不焊）。

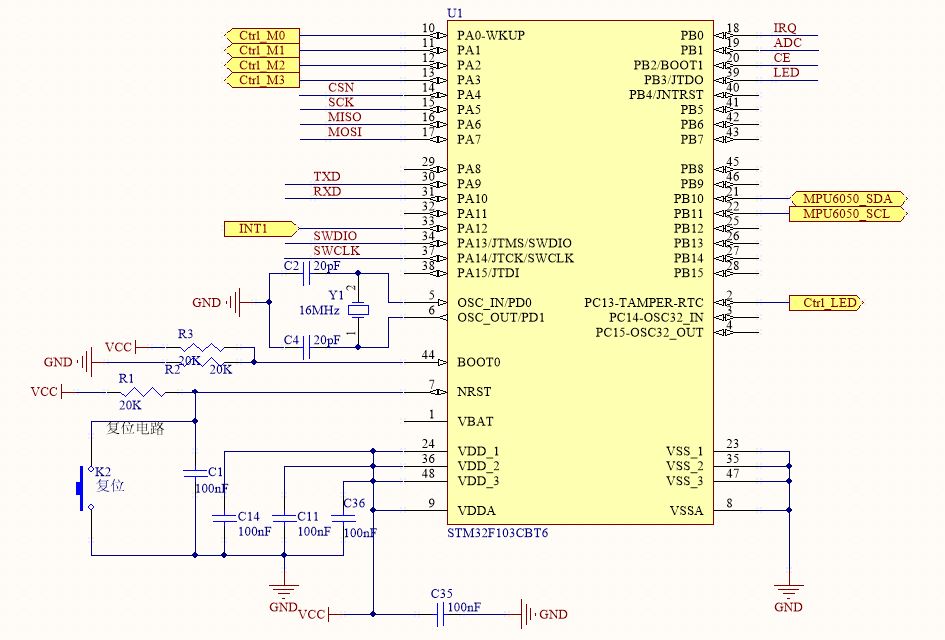


图 2.3 单片机最小系统电路图

### 传感器

四旋翼飞行器姿态数据的获取主要是通过传感器采集飞行器的空中姿态原始数据，飞行器空中姿态数据主要包括绕三个轴转动的角速度和和角度，三个角度分别为YAW（偏航角）、ROLL（横滚角）、PITCH（俯仰角）。

MPU6050为全球首例集成六轴传感器的运动处理组件，内置了运动融合引擎，用于游戏控制器、体感遥控、手持和桌面的应用程序以及其他消费电子设备。它内置一个三轴MEMS陀螺仪、一个三轴MEMS加速度计、一个数字运动处理引擎（DMP）以及用于第三方数字传感器接口的辅助I2C端口（常用于扩展磁力计）。当辅助I2C端口连接到一个三轴磁力计，MPU6050能提供一个完整的九轴融合输出到其主I2C端口。MPU6050拥有16位模/数转换器（ADC），将三轴陀螺仪及加速度计数据转化为数字量输出。为了精确跟踪快速和慢速运动，MPU6050支持用户可编程的陀螺仪满量程范围有：±250、±500、±1000与±2000（单位为(°/s)或dps），支持用户可编程的加速度计满量程范围由±2G、±4G、±8G与±16G[13]。同时MPU6050内置一个可编程的低通滤波器，可用于传感器的滤波以及内置温度传感器和在整个工作温度范围内的振荡频率仅有±1%波动的振荡器，并具有高达10 000GHz的碰撞容忍度。MPU6050封装尺寸仅为4mm×4mm×0.9mm（QFN），因此综合其尺寸及性能，本文中选取MPU6050作为姿态获取传感器，并使用模拟I2C总线对其进行驱动，MPU6050原理图如图 2.4所示。

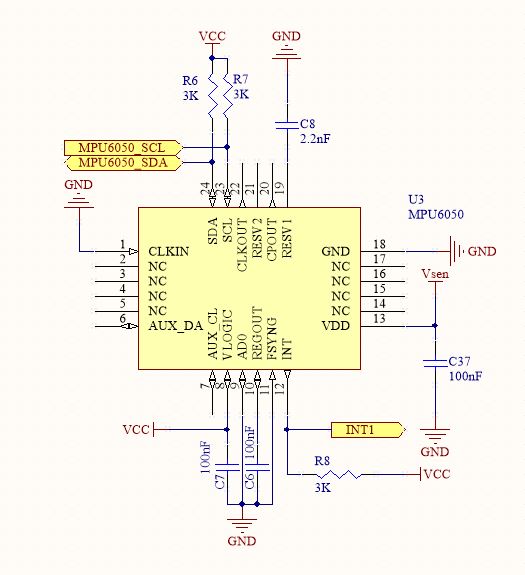


图 2.4 MPU6050原理图

### 无线及遥控通信

为了能够控制飞行器飞行，需要使用无线通信模块在遥控器和飞行器之间建立无线连接。常用的无线通信模块可分为两类：1）建立专用无线数据传输系统包括433MHz频段和2.4G频段；2）借助于现有通信网络GSM、CDMA等公用网信息平台，实现2G/3G通信。使用公用信息平台相比来说能够使数据通信更加稳定和可靠，而且还能够实现远程无线控制，但是它的使用相对来说较为复杂，而且通信有相对较大的时延，并且单位带宽的成本较高。而对于微型四旋翼飞行器来说，并不需要太远的传输距离，同时专用无线数据传输系统具有开发难度低，通信时延小，成本低的优点，因而建立专用无线数据传输系统相对而言更为实际。

本文中选用NRF24L01模块进行无线通信系统的搭建，该模块核心为NRF24L01芯片。NRF24L01是由Nordic公司生产的一款单片无线收发器芯片，其工作频段为2.4GHz~2.5GHz的ISM频段，最大传输速率可达2Mbps，多达125个信道可满足跳频通信和多点通信的需要，其还具有体积小巧（QFN封装）、高效GFSK调制，抗干扰能力强的特点，在无线音视频传输、工业控制等需要较大无线通讯传输速率的领域应用尤为广泛。因此本文中采用NRF2401模块实现无线通信，其接口电路图如图 2.5所示。

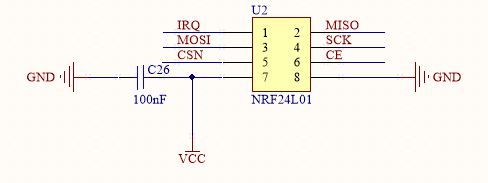


图 2.5 NRF24L01模块接口电路图

### 电源及扩展电路

#### 电源电路设计

飞行器的正常工作离不开电源，本文中选用电压为3.7V容量为650mAh锂电池作为供电电源，其质量大约为19g，一方面能够满足续航的需要，同时能够在最大程度上降低飞行器自身的重量。

对于各模块的供电，微控制器、传感器及无线通信模块的标准工作电压均在3.3V，而为了能够在后续研究中搭载其他5V设备，首先使用LTC3200进行对3.7V电源进行升压。LTC3200具有噪声小、输出电流大、封装小、开关频率高、输出电压稳定和关断后电流小等优点，其电路图如图2.6所示。

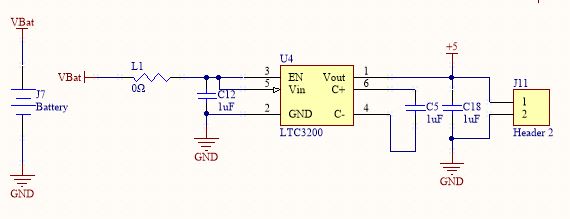


图 2.6 LTC3200升压电路图

由5V稳压到3.3V的芯片则选择XC6206，其最大的特点是超小封装（SOT-23），能够节省飞行器的板上空间，同时降低飞行器自重。结合模电知识，数字信号其实是高低电平，而快速的高低电平转换（如各类通信接口和单片机内部运行）都会使电源内部产生同步开关噪声。由于一般数字信号在1.8V以下为低电平，1.8V以上为高电平，较小的纹波对数字信号本身影响不大，而模拟信号对这样的纹波却相当敏感。例如单片机的A/D是8位的，那么电源波动1/256就会对采样结果产生影响，A/D位数越高，对电源的要求也就越高，因此采用数字信号电源和模拟信号电源分离的方式，同时利用电阻的感性，采用0Ω电阻隔离数字地和模拟地，模拟信号电源电路如图2.7所示和数字信号电源电路如图 2.8所示。

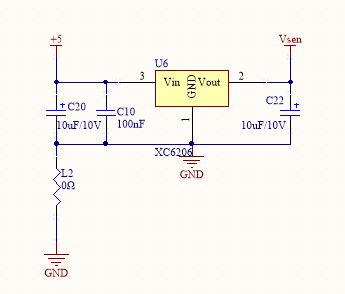


图 2.7 模拟信号电源电路

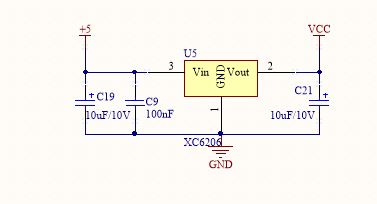


图 2.8 模拟信号电源电路

#### 扩展电路

扩展电路包括电机驱动电路、接口电路、电池电压检测电路以及LED指示电路。

STM32F103CBT6共有8个定时器，其中TIM1和TIM8为高级控制定时器，TIM6和TIM7为基本定时器，其余四个均为通用定时器。基本定时器主要作用是产生DAC触发信号，也可被用作通用的16位时基计数器。通用定时器和高级定时器的区别是通用定时器可以输出4路独立的PWM信号，高级定时器可以输出3对互补PWM信号外加4个独立通道共7路。飞行器在飞行过程中是通过调节PWM信号的占空比来调节电机转速的大小从而使飞行器不断调整姿态，因此选择通用定时器，对应的GPIO口为PA0、PA1、PA2、PA3，即TIM2的四个通道。简单依靠STM32的PWM信号是无法驱动电机的，需要通过MOS管（AO3401）将电流放大，才能驱动电机，为飞行器提供动力。由于电机相当于一个大电感，当断开回路时会产生很高的电压，为了防止反电动势烧坏电机，在这里加上了二极管作为泄流通道，电机驱动电路如图 2.9所示。

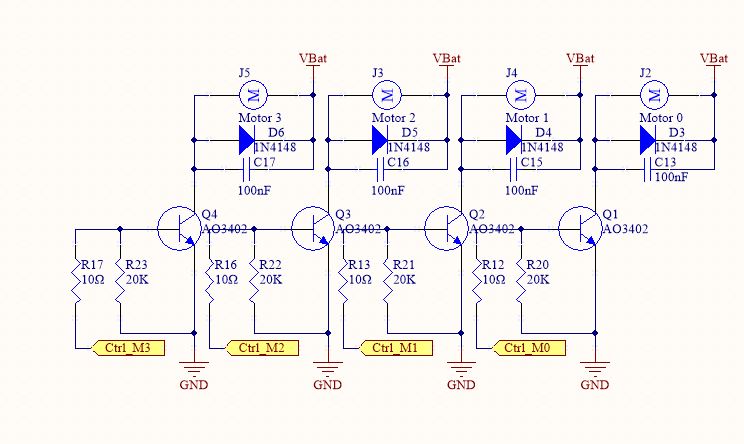


图 2.9 电机驱动电路

STM32支持串行单线调试（SWD）和JTAG接口，为了节省板上空间，采用SWD接口，如图 2.10所示。同时为了测试各模块功能是否正常，预留出了USART接口，如图2.11所示。

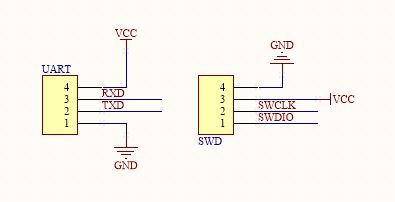


图 2.10 SWD接口电路图

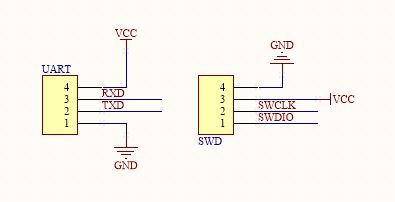


图 2.11 USART接口电路图

为了防止电池电量不足而导致电压过低使飞行器无法正常飞行而坠落，设计了添加了电池电压检测电路，由于STM32的IO口可承受最大电压为5V，为防止端口烧坏，利用电阻进行分压，即AD所测电压为电池电压的一半，其电路图如图 2.12所示。

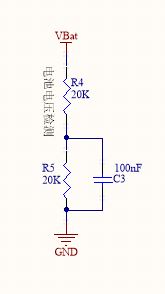


图 2.12 电池电压检测电路

LED指示灯在这里主要用于通信指示，当无线通信连接成功后，LED灯会长亮，电路图如图 2.13所示。

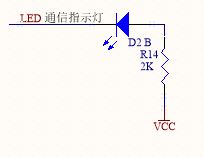


图 2.13 LED通信指示灯

### 飞控板PCB设计

在进行原理图设计过程中，需要自己画出一些Altium Designer官方库中所没有的器件的原理图，比如MPU6050、AO3402和XC6206等，并为这些原理图添加相应的封装库。本设计中飞控板与机架为一体，因此在原理图设计完成后，首先要画出飞行器的整体形状，之后将器件导入进行布局和布线。

器件布局时要综合考虑多方面的影响例如器件物理空间是否相互影响，器件的密度，以及整个飞控板的平衡等，同时在一定程度上还要考虑厂家的技术条件。简单布局完成后则要设置电气规则，之后进行布线，在布线过程中，要充分考虑线的宽度比如说主电源线要在30mile以上，确保其能够承受足够大的电流。除此之外还要考虑信号线之间的影响以及布线对高频信号的影响。最后则要进行电气规则检查，确保PCB生产完成后能够正常使用。

设计完成之后将PCB文件交由厂家进行生产，飞控PCB图如图2.14所示，飞控板最终焊接、装配后的硬件实物图如图 2.15所示。

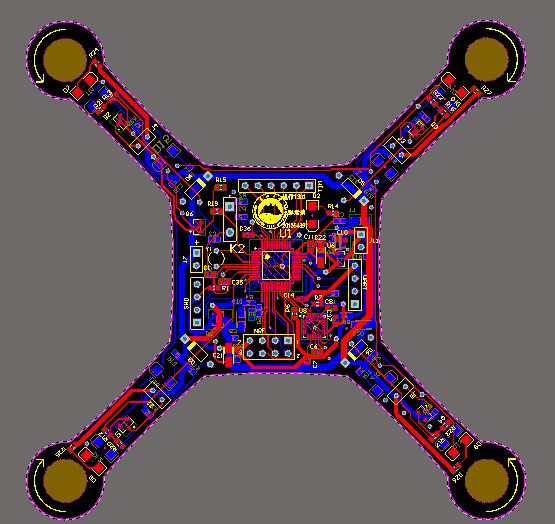


图 2.14 飞控板PCB图



图 2.15 飞控板实物图

## 遥控板硬件设计

相较于飞控板来讲，遥控板要实现的功能较为简单，运算量和代码量较小，因此选择闪存可选程序存储器为64KB的STM32F103C8T6单片机作为主控制器。遥控板单片机最小系统电路、NRF24L01模块接口电路、电池电压检测电路以及LED指示灯电路与飞控板相应电路基本相同，在此不再赘述，下面主要介绍遥控板电源电路以及摇杆电路及微调电路。

### 电源电路设计

遥控板电源电路与飞控板电源电路最大的不同是遥控板耗电量较小、空间大同时不需要3.7V升5V电源设计，直接使用LP2985AIM5-3.3电源稳压芯片将电池电压稳定在3.3V。

TP4057是一款采用恒定电流、恒定电压的线性充电方式且自带电池正负极反接保护的单节锂离子电池充电器。TP4057适合多种充电电源，包括USB电源和适配器电源。TP4057内部采用的是P-MOSFET架构，并且还增加了加上防反充电路，因此不需要额外添加外部检测电阻器和隔离二极管。其具有的热反馈功能可对充电电流大小进行调节，使得其在大功率操作下和高温环境中对芯片温度加以控制。TP4057最大充电电压固定于4.2V，但可以通过一个电阻器对充电电流进行外部设置。在达到最终浮充电压之后，充电电流会减小到设定值的十分之一，此时TP4057将自行终止充电循环。在充电电源（交流适配器或USB电源）断掉之后，TP4057则会自行进入到一个低电流的状态，将电池漏电流减小至2μA以下。此外，TP4057还具有充电电流监控、欠压闭锁、自动再充电的特点以及具有两个用于指示输入电压和充电结束的状态引脚。除遥控操作飞行器以及与ROS进行串口通信外，设计中在遥控板上加入了TP4057锂电池充电芯片，便于为飞控板以及自身电池充电。遥控板电源电路如图 2.16所示。

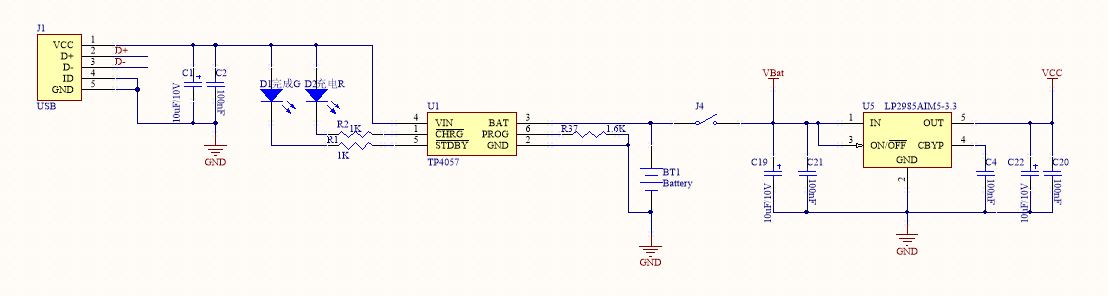


图 2.16 遥控板电源电路

### 遥杆电路及微调电路

摇杆电路如图 2.17所示。每个摇杆由2个电位器组成，因此摇杆电路只是简单的A/D采样电路。

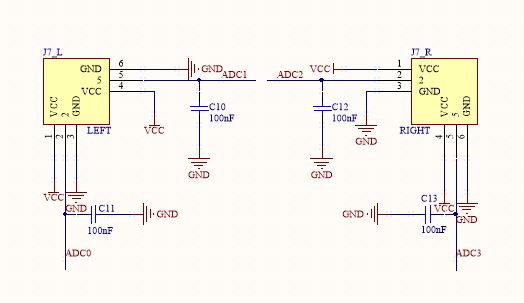


图 2.17 摇杆电路

在飞行器仅加油门飞离地面后不能垂直起飞是，可通过微调电路进行调节。如果飞行器朝前或朝后飞行，可通过前后微调旋钮进行调整；如果飞行器朝左或朝右飞行，则可通过左右微调旋钮进行调整；以此类推，自转微调用于飞行器顺时针或逆时针自转时进行校正。微调电路如图 2.18所示，从左至右依次为前后微调、左右微调和自转微调。

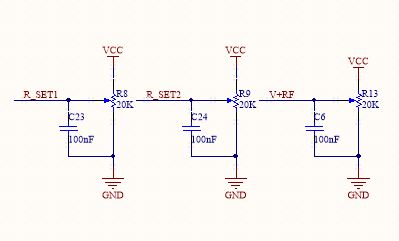


图 2.18 微调电路

### 遥控板PCB设计

遥控板PCB设计与飞控板相同，为了便于操控，遥控板PCB形状设计为常用的羊角形，遥控板PCB图如图 2.19所示，硬件实物图如图 2.20所示。

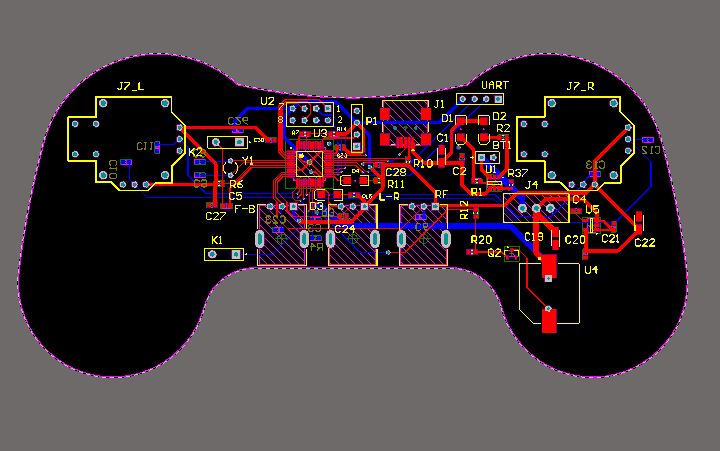


图 2.19 遥控板PCB图



图 20 遥控板实物图

# 飞行器姿态解算算法

## 姿态解算的目的及意义

姿态解算是指飞行器控制器通过传感器获取飞行姿态数据，之后对飞行器姿态角进行实时地计算，例如常见的姿态角有俯仰角（Pitch）、翻滚角（Roll）及偏航角（Yaw），控制器根据这些姿态角信息调节4个电机的输出大小，从而使飞行器能够平稳飞行并且按照操纵者的意愿做出相应的动作。通过三轴陀螺积分可直接获得飞行器姿态数据，陀螺仪动态响应特性较好，因此获得的数据短期内精度较高，但在进行姿态解算时会因积分产生累计误差而最终导致飞行器姿态发散；飞行器的倾角可通过加速度计测量，但它在对重力加速度敏感的同时也对运动加速度较为敏感，因此飞行器在做长时间机动运动时测量误差就会非常的大[14]。故而选择合适的姿态解算算法有着极为重要的意义。

目前有很多姿态滤波和解算算法，在飞行器中应用比较广泛的是扩展型卡尔曼滤波算法（EKF），其解算精度较高。然而EKF存在3大缺陷：（1）通常情况下雅可比矩阵的计算过程很难实现，而且其计算量相当的大，而微控制器的计算能力也非常有限；（2）在实际应用中，实际噪声很难符合白噪声的要求；（3）当线性化条件假设不成立时，线性化反而会导致滤波器极度的不稳定[15][16]。而互补滤波算法对传感器精度要求不高且计算量也不大，在微型四旋翼飞行器的姿态解算中应用广泛。

## 飞行器姿态表示方法

### 欧拉角

欧拉角表示方法由于可以直接读取出飞行器的姿态而被广泛应用。从一个坐标系转到另一个坐标系可以通过绕三个坐标轴连续旋转一次来实现，图 3.1描述了坐标系转换所进行的绕不同坐标轴的连续三次转动[17]。

假设全局坐标系固定于地面保持静止，而随飞行器转动的坐标系初始姿态与重合为，然后按以下顺序旋转至最终的姿态：

（1）绕着的Z轴按右手坐标系规则转动偏航角（yaw）*ψ*，得到式（3.1）中的旋转矩阵，转动后的新姿态为。

 (3.1)

（2）绕着的Y轴按右手坐标系规则转动俯仰角（pitch）*θ*，得到式（3.2）中的旋转矩阵，转动后的新姿态为。

 （3.2）

（3）绕着的X轴按右手坐标系规则转动翻滚角（roll），得到式（3.3）中的旋转矩阵，转动后得到最终坐标系。

 （3.3）

连续绕三个坐标轴的基本旋转运动，可以合成任意两个坐标系下的转换矩阵，基本旋转运动的先后顺序决定了合成的先后顺序。因此，机体坐标系相对于全局坐标系下的姿态可以看作是地理坐标系绕yaw、pitch和roll做基本旋转运动后的复合结果。从而，*Ab*在*Ae*中的姿态表示即为以上三次转动的合成矩阵：

 （3.4）

其中，，其他同理。

若已知旋转矩阵，则可得欧拉角：

 (3.5)

其中，，其他同理。

### 四元数表示法

四元数是现今表示姿态较为常用的一种方法，四元数表示法没有奇异点的问题并且计算量小、精度高，能够方便的给出旋转的转轴和旋转角[18]。

四元数是用超复数乘法来等效表示姿态变换的，它的定义类似于复数的定义，定义中用到了三个虚数，其具体表示如下：

 （3.6）

其中，为实数，既是相互正交的单位向量，又是虚单位，并且满足如下性质：

 （3.7）

式中，⊗表示四元数乘法。上述规则可叙述为：相同单位向量相乘时呈现虚单位特性；相异单位向量做乘积时呈现单位向量交叉特性。

四元数也可简单的表示为，在进行姿态解算是，用以表示旋转矩阵的四元数必须是单位四元数，因此*q*应当满足：

 （3.8）

给定两个四元数和，假定他们的乘积为***w***：

 （3.9）

此结果可以分解为两个矩阵的乘积，姿态解算过程中将会用到。分解如下：

 （3.10）

F

θ

图 3.1 旋转产生四元数示意图

通过绕一旋转轴F旋转δ，便可构造一个四元数，其示意图如图 3.1。假定旋转轴为单位向量，则四元数可表示为：

 （3.11）

由此可得：

 （3.12）

旋转矩阵表达式为：

 （3.13）

式中，

将式（3.12）带入到式（3-14）中可得：

 （3.14）

进而由式（3.5）可得欧拉角：

 （3.15）

## 姿态解算算法实现

### 互补滤波算法

互补滤波器是一种频域特性滤波器，来自不同传感器对同一信号测量得到的数据常用于其来融合。一般来说，互补滤波器所融合的信号至少有两种是频率特性互补的，对于飞行器来说，陀螺仪和加速度计两个输入量都能分别对姿态角进行解算，其中陀螺仪包含低频噪声，应采用高通滤波器对其进行滤波；加速度计输入量则包含了高频噪声，那么就应通过低通滤波器来滤除[16]。二者的频率特性互补，那么可以采用互补滤波对其进行姿态解算，最终输出较为稳定准确的信号。其工作原理图如图 3.2所示。

陀螺仪

滑动滤波

数值积分

高通滤波

加速度计

IIR滤波

角度解算

低通滤波

俯仰角翻滚角

图 3.2 互补滤波数据融合原理图

假设由陀螺仪和加速度计分别解算出的飞行器姿态角x的值为[19]：

 （3.16）

式中，为加速度计高频噪声和为陀螺仪低频噪声。

假设互补滤波器由低通滤波器和高通滤波器两部分构成，*f*为滤波器常数，则有：

 （3.17）

则姿态角*x*的估计值可表示为：

 （3.18）

对应的差分方程表达式为：

 （3.19）

式中，为k时刻陀螺仪角速度积分值，为加速度计解算的角度值。

### 四元数姿态解算算法实现

运用互补滤波的思想进行姿态融合实质上就是利用加速度计解算出的姿态角去修正陀螺仪积分的漂移误差。在姿态融合过程中，一般都是用都是欧拉角表示姿态角，但当飞行器角度为90°时，对加速度计进行姿态解算时会出现反三角函数无解的情况，为了避免该问题，可采用四元数进行解算姿态[20]。

四元数姿态解算算法流程如图 3.3所示：

初始化四元数

归一化加速度计值

四元数表示重力分量

计算误差向量

互补滤波修正角度

更新四元数

归一化更新后的四元数

四元数转欧拉角

图 3.3 四元数姿态解算算法流程图

四元数姿态解算具体步骤如下：

（1）初始化四元数

定义四元数为全局变量。

（2）将加速度计的三维向量转化为单位向量（归一化）

 （3.20）

（3）用四元数表示三轴的重力分量（为四元数表示的旋转矩阵的第三行）

 （3.21）

（4）求加速度计读取的方向与重力加速度方向的差值

 （3.22）

在飞行器坐标系中，加速度计测得的重力加速度分量为；陀螺仪积分后推算得到的重力向量是。陀螺仪积分误差是用向量叉乘表示因此该误差向量仍位于飞行器坐标系中。

（5）互补滤波修正角度

误差积分：

 （3.23）

式中，为误差积分常数，为误差积分后的结果。

通过式（3.23）中的PI修正陀螺零偏：

 （3.24）

式中，为误差比例系数，为陀螺仪测量的角速度。

（6）更新四元数

 （3.25）

式中，halfT为姿态更新周期的一半。

（7）归一化更新后的四元数

 （3.26）

（8）四元数转欧拉角

 （3.27）



# 飞行器控制算法与参数整定

## PID控制器

### PID控制器介绍

PID控制理论是通过观察舵手的操作发展而来的。PID控制器最早可以追溯到19世纪末的调速器设计，在船舶自动操作系统中逐渐发展起来的。最早期的PID控制器是1911年Elmer Sperry开发的，而第一篇PID控制器理论分析论文则是由俄裔美国工程师尼古拉斯∙米诺尔斯基发表的。

PID（比例（Proportion）、积分（Integral）、微分（Derivative））控制器是一个在工业控制领域中应用广泛的反馈回路控制器，由比例单元P、积分单元I和微分单元D组成。PID控制的基础是比例控制，可以减少调节时间和稳态误差，但过大的比例作用，则会使系统的稳定性降低；积分控制可消除稳态误差，但可能增加超调，造成系统的稳定性降低；微分控制能够预见偏差变化趋势从而增强系统响应性能以及减弱超调趋势。最基本的单级PID控制算法流程图如图 4.1，其对应的公式为式（4.1）

积分

积分

积分

受控对象

*r(t)*

+

*e(t)*

−

*u(t)*

*y(t)*

+

+

+

图 4.1 单级PID算法流程图

 （4.1）

在实际的应用中，只需要应用PID控制器的部分单元，因此只需将不需要的的单元的参数置为零即可。故而PID控制器可以变为PI控制器或PD控制器等。二者的特点分别为：在系统稳定的前提下，引入PI控制器可以使系统的的稳态控制质量明显提高，稳态误差也随之消除；PD控制则可将输入信号的变化趋势反应出来，在一定程度上具有预见性，通过早期有效的为系统引进一个修正信号，使系统的阻尼程度得以增加，从而提高系统的稳定性。由于微分控制对噪声十分敏感，所以如果没有积分控制的话，系统就会存在稳态误差，因此相对来说，PI控制器比较常用。

### 单级PID控制算法

单级PID控制属于一种线性控制器，这种控制器被广泛应用于四轴上。[21][22]通过前面的分析可以得出，要使得飞行器能够平稳飞行，那么显而易见的就是要控制前文中所述的翻滚角、俯仰角和偏航角。那么最简单的一种控制策略就是角度单环PID控制器，系统框图如图4.2所示：

姿态

PID控制

电机

IMU

四旋翼飞行器

期望角度

PWM×4

当前角度

−

图 4.2 角度单环PID控制器系统框图

其实现算法如下：

当前角度误差=期望角度−当前角度

单环PID\_P项=\*当前角度误差

当前角度误差积分及其积分限幅

单环PID\_I项=\*当前角度误差积分

当前角度的微分（角速度，由陀螺仪给出）

单环PID\_D项=\*当前角度的微分

单环PID输出=单环PID\_P+单环PID\_I+单环PID\_D

### 串级PID控制算法

单级PID控制器是一种线性控制器，但是四旋翼飞行器是一个非线性的系统。因此四旋翼飞行器采用单级PID控制算法，会使系统存在不同程度的超调和震荡。我们可以在姿态控制中加入角速度环，从而进一步的控制它的角速度，构成串级PID控制器，这样就可以增加飞行器的稳定性并提高它的控制品质[23][24][25]。

串级PID控制器简单来说就是两个PID控制器串在一起，分内环和外环，其中外环为角度环，内环为角速度环。四旋翼飞行器采用串级PID控制器进行控制，即使外环数据即角度发生剧烈变化，仍然可以实现一个非常好的控制效果。串级PID控制器系统框图如图 4.3所示[21]。

姿态

角速度环PID控制

电机

IMU

四旋翼飞行器

期望角度

PWM×4

当前角度

−

角度环PID控制

期望

角速度

当前角速度

图 4.3 串级PID控制器系统框图

串级PID控制实现算法如下：

当前角度误差=期望角度−当前角度

外环PID\_P项=外环\*当前角度误差

当前角度误差积分及其积分限幅

外环PID\_I项=外环\*当前角度误差积分

外环PID输出=外环PID\_P+外环PID\_I

当前角速度误差=外环PID输出−当前角速度

内环PID\_P项=内环\*当前角速度误差

当前角速度误差积分及其积分限幅

内环PID\_I项=内环\*当前角速度误差积分

当前角速度微分=（本次角速度误差-上次角速度误差）

内环PID\_D项=内环\*当前角速度的微分

内环PID输出=内环PID\_P+内环PID\_I+内环PID\_D

## PID参数整定

在实际的工业应用中，整定PID参数的方法通常有衰减曲线法、临界比例法和人工调试法等。衰减曲线法是根据系统的衰减频率特性来进行参数的整定，首先将系统调试到出现两个相邻的超调波形，两个波形要求一大一小，记录下两个波形出现的时间差以及此时的比例系数*Kp*，通过时间差和*Kp*即可整定出较理想的PID参数；临界比例法适用于系统阶数大于3阶的振荡系统；人工调试法即通过调试经验对控制器的各个参数分别进行调试，一般要按照先比例、再积分、后微分的顺序进行调节。人工调试法相对于其他两种方法实现起来较为简单，因此在这里采用人工调试的方法。

### 上位机软件介绍

为了方便调试，设计中使用MWC开源四轴飞控的上位机软件。MultiWii是一款通用的多旋翼飞行器控制软件，这款软件不仅能够控制三轴、四轴、六轴等系列的多旋翼飞行器，还可控制固定翼和直升机等航模。软件界面如图 4.4所示。

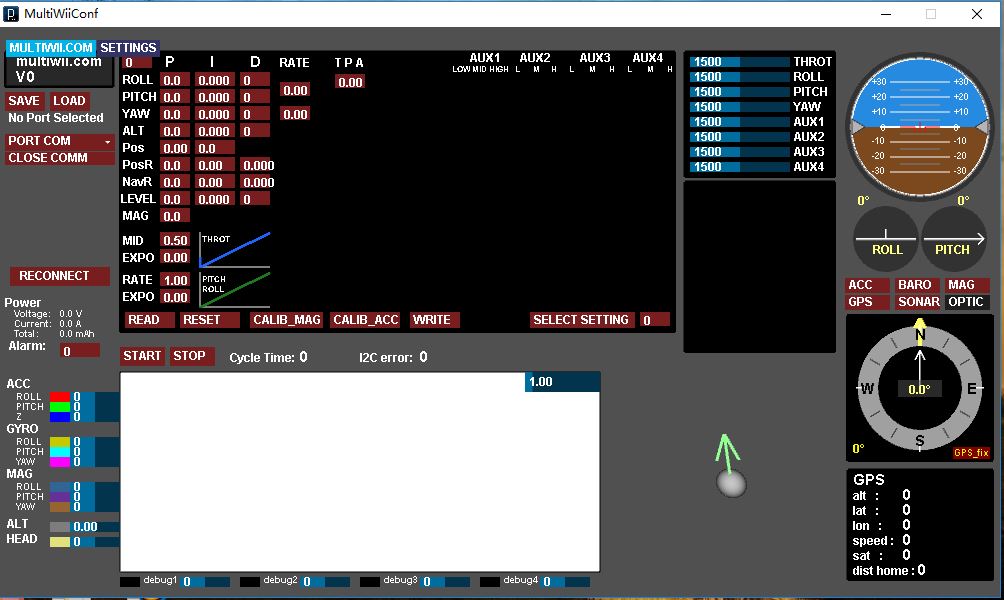


图 4.4 上位机软件界面

软件与遥控板使用串口进行协议通信，通信协议的一般格式如下：

发送给上位机：

‘$’‘M’‘>’[data length][code][data][checksum]

发送给飞行器：

‘$’‘M’‘<’[data length][code][data][checksum]

‘$’‘M’‘>’这三个字符为帧头，其中，‘>’表示向上位机发送数据，‘<’表示向飞行器发送数据；data length 为数据长度，即数据的个数；code为功能帧标志，本文中用到的功能帧标志及指令代码如表 4.1所示；data即为要发送的数据，低位在前，高位在后；checksum为数据校验位，由[data length][code][data]数据按字“异或”得到[26]。

表 4.1 功能帧标志及指令代码

|  |  |  |  |
| --- | --- | --- | --- |
| 功能帧标志 | 指令代码 | 方向 | 数据类型 |
| MSP\_RC\_DATA | 10 | < | UINT16 |
| MSP\_FLY\_DATA | 11 | > | UINT8&INT16&UINT16&UINT32 |
| MSP\_IDENT | 100 | > | UINT8 |
| MSP\_RAW\_IMU | 102 | > | INT16 |
| MSP\_MOTOR | 104 | > | UINT16 |
| MSP\_RC | 105 | > | UINT16 |
| MSP\_ATTITUDE | 108 | > | INT16 |
| MSP\_ANALOG | 110 | > | UINT8 |
| MSP\_PID | 112 | > | UINT8 |
| MSP\_MOTOR\_PINS | 115 | > | UINT8 |
| MSP\_SET\_PID | 202 | < | UINT8 |

### PID参数整定过程及结果

PID参数整定的具体过程如下：

（1）所有参数归零，先调Pitch和Roll的内环，不断增大，直到四轴能平稳起飞并记下这个值。然后不断增大，直到飞行器开始轻微振荡，把参数调小一些，然后加入一点点参数抑制振荡，内环暂时可以。此时四轴是一个三轴模式，只有陀螺仪起作用，飞行器可以保持当前角度，但不会自己修正角度。把四轴拿在手中摆动，可以明显感觉到抵抗的力。（如果此时四轴Yaw方向会自旋，就给Yaw内环加入参数，不断增大到四轴不会自旋）。

（2）调Pitch和Roll的外环，从0开始增大，会感到四轴回复力度越来越大，但是会振荡。加入一点参数，就能有很好的效果了。这个步骤如果怎么都调不好，那么就要返回上一步检查角速度环是否会振荡。

（3）Yaw内环从0开始增大，到飞行器转动平稳有力即可。

在调试过程中，为达到较好的调试效果，可以用两根绳子将飞行器对角线吊起来，这样在一定程度上也能保证安全，同时也能够防止由于参数不合适导致的侧翻和撞击对飞行器造成损坏。在吊着进行整定取得较好的效果后，将飞行器取下，并在现有的参数周围进行适当调整。之后则是在室外进行整定。经过反复进行整定，最终得到较为理想的PID控制参数见表 4.2。

表 4.2 四旋翼飞行器PID参数

|  |  |  |  |
| --- | --- | --- | --- |
| PID控制参数 | 比例（K） | 积分（I） | 微分（D） |
| 俯仰角内环 | 2.8 | 0 | 6 |
| 俯仰角外环 | 5.0 | 0.01 | 0 |
| 翻滚内环 | 2.8 | 0 | 6 |
| 翻滚角外环 | 5.0 | 0.01 | 0 |
| 偏航角 | 15.1 | 0.021 | 0 |

# 飞行器与机器人操作系统通信

## ROS的总体框架及基本概念

通常情况下，使用开源社区级、计算图级和文件系统级三个层级来划分ROS的基本概念[27]。如图5.1所示。

社区级

•代码仓库

计算图级

•节点、消息、主题、服务、节点管理器、参数服务器、消息记录包

文件系统级

•堆、包

图 5.1 ROS的层级

社区级主要用于开发者和用户发布和维护自己的软件仓库代码，是ROS系统的重要组成部分，得益于开源社区的大力支持，ROS才得以不断地发展壮大。

文件系统级主要是指存储在硬盘上可以查看的ROS源代码。在这一级中，两个概念重要的概念是包和集。功能包（Package）是ROS中软件组织的基本形式，可以包含ROS的节点、服务、消息以及配置文件等。集则是若干功能包的集合。

ROS系统组成的中坚力量是计算图级。计算图是ROS运行时各节点间点对点的网络形式，在这一级中，最重要的概念包括节点、主题、消息以及服务。除这些之外，ROS计算图级基本的概念还包括节点管理器、参数服务器和包。

节点（Node）是执行计算的过程，可以被看作为一个可执行文件。要想有一个进程可以与其他节点进行通信，则需要创建一个节点。每一个节点往往实现一种功能，因此实际应用中需要很多个节点之间密切配合。

消息（Message）用于节点间的相互通信。消息携带着一个节点发送到其他节点的数据信息。消息可以包含多种基本数据类型如整型、字符串甚至是时间。ROS中不仅包含了多种标准类型信息，而且用户还可以基于这些标准消息开发自定义类型的消息。

主题（Topic）有些类似于现实中的以太网交换机，提供了用于消息传递的通道，是对消息进行路由和管理的总线。发布者向主题发布消息，订阅者则通过主题订阅消息，主题起到的作用就是将发布者的消息收集起来，然后将这些消息分发给订阅者，这就保证了消息的发布者和订阅者无需知晓对方的存在，但在同时，也务必保证各个主题的名称是唯一的，否则就会像IP冲突一样无法将消息路由到目的地。

服务（Service）是一种请求-回复的通信方式，其经常被用到分布式系统中。这种通信模式下，一个节点提供某种服务，而另一个节点要想使用这种服务则需要先发送一个请求信息并等待响应。

节点管理器（Master）提供了节点名称的注册和查找功能。节点之间只有找到彼此之后，相互之间建立点对点的通信，才能进行节点订阅与发布以及服务的请求与相应。

参数服务器（Parameter Server）也是节点管理器的一部分，可以使数据通过关键词保存在一个系统的中心位置。

消息记录包（Bag）用于保存和回放ROS消息数据，是用于存储数据的重要机制。可以通过它收集各种各样的数据，并且这些数据能够回放，对日后的开发和测试起到巨大的帮助作用[4][5]。

ROS计算图级的概念及之间的相互关系如图 5.2 所示

主题

服务

发布

订阅

消息

回放

记录

请求

响应

消息记录包

图 5.2 计算图级概念及关系图

## rosserial通信协议简介

rosserial是一种ROS串行通信协议，通过串行传输介质实现ROS的P2P通信。这种协议通过简单的添加包头和包尾可以实现了多主题或者服务共享串行通信介质（如串口，网络socket）[28]。

### rosserial协议包格式

rosserial协议包格式如下：

第一字节：同步标志

第二字节：同步标志或者协议版本

第三字节：数据长度的低字节

第四字节：数据长度的高字节

第五字节：数据长度校验和

第六字节：主题号的低字节

第七字节：主题号的高字节

第N字节：数据流

N+8字节：主题ID和消息数据校验和

不同ROS发行版对应不同协议版本字段定义（0xff对应的版本为Groovy版本，0xfe对应的版本为：Hydro，Indigo和Jade）。

主题号0-100为系统功能专用主题使用，这些主题类似于消息rosserial\_msgs/TopicInfo中定义的那些特定主题。

长度和数据的校验和字段用于确保包的完整性，数据的校验和可以按照如下公式计算：

校验和=255-（（主题号低字节+主题号高字节+数据字节数）%256）

### 主题协商与同步

在数据传输之前，PC/平板一侧必须先向Arduino或者其它嵌入式设备发送主题查询请求，确定将要发送或者订阅的主题的名字和消息类型。

主题协商由主题查询请求，响应和主题定义组成。主题查询请求使用的topic ID为0。主题查询请求类似于如下所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0xff | 0xfe | 0x00 | 0x00 | 0xff | 0x00 | 0x00 | 0xff |

主题查询响应包(消息类型为rosserial\_msgs/TopicInfo）包含了特定主题信息，使用如表5.1所示的数据信息：

表 5.1 主题查询响应包数据信息

|  |  |  |
| --- | --- | --- |
| 数据类型 | 数据信息 | 说明 |
| uint 16 | topic\_id |  |
| string | topic\_name | 主题名称，如“mini\_Fly” |
| string | message\_type | 消息类型，如“geometry\_msgs/Twist” |
| string | md5sum |  |
| int32 | buffer\_size |  |

PC/平板与嵌入式设备相互之间的时间同步通过发送消息std\_msgs::Time实现 嵌入式设备可以向PC/平板发送空的时间消息获取当前时间，响应返回的时间可以用于时间同步（检查时钟偏差）。

## 基于rosserial功能包的节点通信

STM32与ROS的通信是通过rosserial功能包实现的，在ROS Wiki（用于记录有关系统信息的主要论坛）上，有基于rosserial的arduino与ROS的通信的教程，根据此教程，实现了STM32与ROS的通信。本文中STM32与ROS通信的实现中，使用的是Ubuntu 14.04 LTS版本，ROS版本为ROS indigo。

### 创建ros\_lib库

在arduino与ROS使用rosserial进行通信的教程中，首先就是通过arduino IDE创建ros\_lib库，而后将程序通过串口烧录到arduino中，并进行串口配置实现二者的通信[29]。利用arduino IDE生成ros\_lib库的具体步骤如下：

（1）安装arduino IDE。按Ctrl+alt+T打开终端，输入sudo apt-get install arduino。

（2）安装ROS包。在终端内输入sudo apt-get install ros-indigo-rosserial-arduino，之后继续输入sudo apt-get install ros-indigo-rosserial。

（3）设置arduino的SketchBook，将ros\_lib加入进去，关于SketchBook的文件目录，在File->Preferences，如图 5.3。

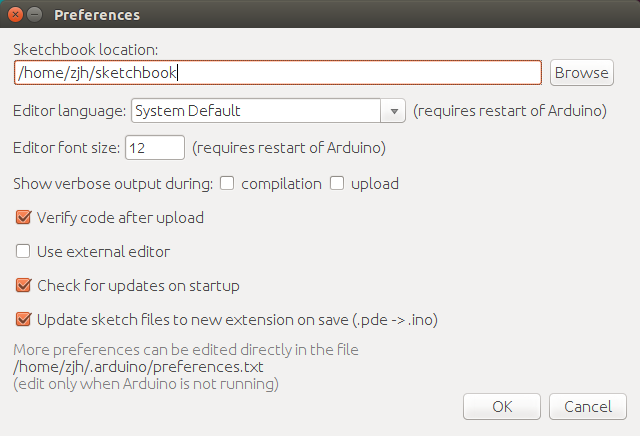


图 5.3 SketchBook目录

（4）生成ros\_lib库。在终端中输入rosrun rosserial\_arduino make\_libraries.py。

完成以上步骤后，在终端内输入arduino打开arduino IDE，选择File-Examples，便能够看到ros\_lib，说明ros\_lib库已经生成，界面如图 5.4所示。

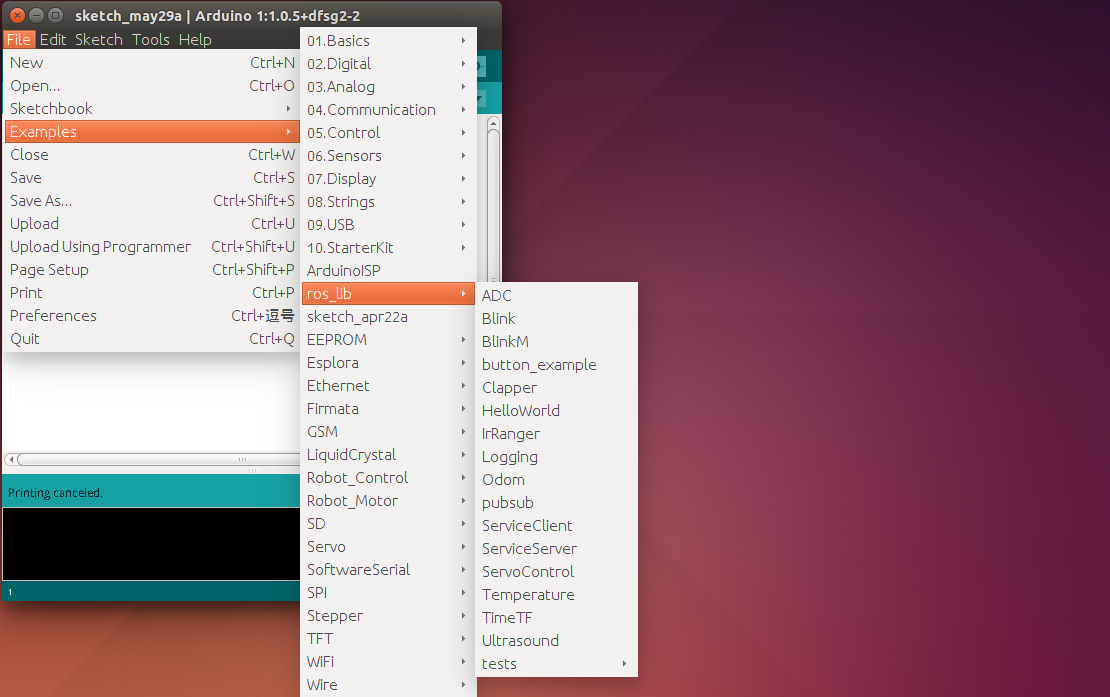


图 5.4 ros\_lib库生成界面

### MDK下的开发

本设计中，对STM32的开发是在MDK V5.12下完成的，该软件集成了μVision5 IDE集成开发环境，是目前针对ARM微控制器尤其是ARM Cortex-M内核最为便捷、功能最全的一款开发工具。下面是在飞行器源程序的基础上实现与ROS通信的过程。

（1）将上一小节中配置好的ros\_lib库复制到原有的工程下，同时将其添加到工程中。由于开发过程中，需要用到C++语言，因此在选项中的C/C++标签中的Misc Controls一栏写上-cpp，使得IDE能够支持C++编程。

（2）建立stm32Hardware.h。根据ROS Wiki提供的教程，需要建立stm32Hardware.h文件，并在其中写一个class stm32Hardware类，其中包括5个函数，分别是构造函数（一般为空）、初始化函数（init()，写串口初始化）、读串口函数（read）、写串口函数（write(data,length)）和返回时间函数（unsigned long time()）。下面主要介绍一下返回时间函数、读串口函数和写串口函数。

返回时间函数：相对来说较为简单，直接返回抵达定时器的数值即可。

读串口函数和写串口函数：此处读写串口用到是串口最基本的中断读写，利用模拟读写队列来完成，具体实现的过程如下：

① 串口接收中断。实现方法是把读出的数据存入队列中，之后在head处放入读出的数值，head自加，最后判断head是否等于tail，等于tail则关闭串口接收中断。

② 接收队列的读取。一方面要判断当前队列是否是满的，即head与tail不能相等；另一方面就是获取到队列里面要被读的数据。

③ 将数据放入发送队列中。首先要判断发送队列是否有空余地方，即是要放数据的地方和要发送数据的地方不重合），然后则是把要发送的数放入队列中，即将数据放在head处，之后head自加）。

④ 串口发送中断，与接收中断类似。

⑤ 串口中断处理函数。在所有的串口中断处理函数前面加上“extern C”，否则单片机将无法进入到终端中。

（3）将要飞行器姿态数据添加到数组中。首先遥控端不断接收飞行器发来的姿态数据，然后将姿态数据放进ROS的回调函数，这样订阅者在接收到ROS的消息后，就会将姿态数据发布出去。

### 结果测试

使用TTL转接口将遥控板串口与USB口连接起来，打开遥控器与飞行器电源，具体的测试步骤如下。

（1）打开终端，输入roscore启动ROS操作系统。

（2）打开一个终端，输入rosrun rosserial\_python serial\_node.py /dev/ttyUSB0运行rosserial客户端，出现如图 5.5所示的信息，说明连接成功。

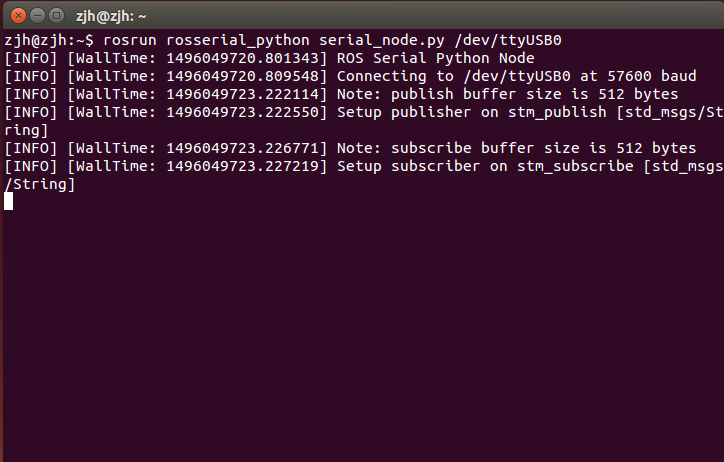


图 5.5 STM32与ROS连接成功信息

（3）打开一个终端，输入rostopic pub -r 5 /stm\_subscribe std\_msgs/String hello，则会以5 Hz的频率向话题stm\_subscribe发送hello。

（4）打开一个终端，输入rostopic echo /stm\_publish则会以5 Hz的频率返回飞行器的姿态数据，如图 5.6所示。

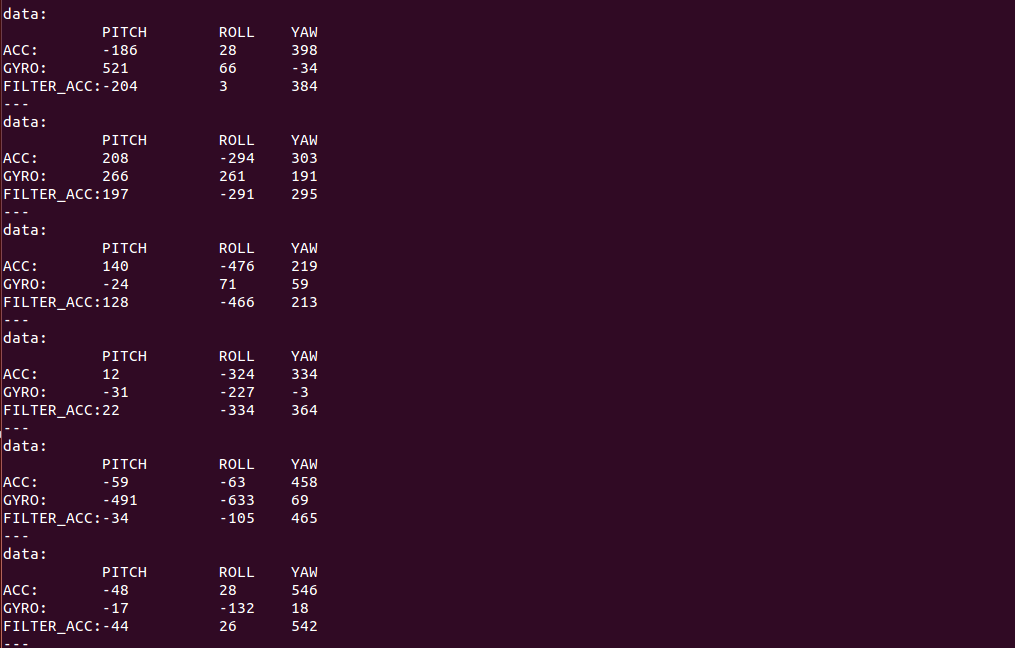


图 5.6 ROS返回的飞行器姿态数据

# 结 论

## 本文工作总结

本文以基于STM32的飞行器为平台，实现了STM32与ROS的通信，并将飞行器的姿态数据实时地展现了出来，为日后关于嵌入式微控制器与ROS的研究提供了简单的参考。本文主要的工作有以下几点：

（1）本文首先介绍关于四旋翼飞行器的基础知识，包括其发展历史、基本结构以及工作原理。同时简要介绍了机器人操作系统，说明其在机器人开发领域的作用以及在嵌入式微控制器应用的情况。

（2）根据日后研究和开发的需要，提出了四旋翼飞行器系统的总体需求，并根此需求，选择了合适的芯片并且使用Altium Designer对飞行器和遥控器进行了硬件设计和组装。

（3）为了使飞行器能够平稳飞行，在硬件平台的基础上，实现了飞行器姿态滤波和解算算法，并且使用了串级PID控制算法对飞行器进行有效的控制。

（4）在飞行器平稳飞行的基础上，本文介绍了使用rosserial功能包实现STM32与ROS通信的过程，并展现了飞行器的姿态数据。

## 展望

本论文的研究为四旋翼飞行器的设计以及ROS在嵌入式微控制器的应用提供了思路。但是，由于时间限制，主要有以下几个方面的工作还需进一步研究：

（1）由于本文设计的飞行器为微型四旋翼飞行器，仅仅考虑了最小的开发需求并且电机的转动会对磁力计产生干扰，因此四旋翼飞行器仅仅采用了MPU6050即陀螺仪和加速度计，而没有采用磁力计，因此姿态解算时只能确定飞行器的俯仰角和横滚角，而无法确定偏航角，所以只是在上电时刻，四旋翼机头指向的的方向为前方，无法准确的确定方向。在下一步的工作中，需要找到一种能够消除电机对磁力计产生干扰的方法，使飞行器可以确定偏航角。

（2）本文使用的PID控制器需要手动的进行调节，需要较丰富的调试经验，调试不当还会对飞行器造成一定的损坏，因此设计一种自适应的PID算法也是极有必要的。

（3）除了陀螺仪、加速度计以及磁力计，还可以在飞行器上搭载更多的传感器，如气压计、PX4FLOW光流传感器和GPS模块等。

（4）利用ROS中的rviz构建飞行器模型，并且利用飞行器的姿态数据实时展现飞行器的姿态。

（5）利用主题的发布与订阅，使PC作为上位机，对飞行器发出指令，对飞行器进行操控。

[参考文献](file://D:\Documents\毕设\桌面\论文模板\文献格式(篇数大于30篇且含近三年五篇文献)%201专著格式:序号.编著者.书名【M】,出版地:出版社,年代,起止页码%202期刊论文格式:序号.作者.论文名称【J】,期刊名称,年度,卷（期）:起止页码%203学位论文格式:序号.作者.学位论文名称【D】,发表地:学位授予单位,年度%20例:1.张艺.铸造工艺【M】,北京:机械工业出版社,1994,4-5%202.张颖伟.一类大组合系统的容错控制【J】,东北大学学报,2000,52(4):351-355%203.周丽.挖掘机的优化【D】:沈阳:沈阳东北大学:2000)

1. 甄红涛,齐晓慧,夏明旗等. 四旋翼无人直升机飞行控制技术综述[J]. 飞行力学,2012,(04):295-299.
2. P. G. Fahlstrom, T. J. Gleason. 无人机系统导论[M].北京: 电子工业出版社,2003.
3. Unmanned Aircraft Systems Roadmap,2005-2030[R].office Secretary of Defense (USA) Report August,2005.
4. 张建伟,张立伟,胡颖等.开源机器人操作系统——ROS[M].北京: 科学出版社, 2012.
5. Aaron Martinez Enrique Fernández. ROS 机器人程序设计[M].刘品杰,译.中国: 机械工业出版社, 2014.
6. Phillip Mc Kerrow. Modelling the Draganflyer four-rotor helicopter. In proceedings of the 2004 IEEE International Conference Robotics and Automation, 2004.
7. Gabe Hoffmann, Dev Gorur Rajnarayan, Steven L. Waslander, David Dostal, Jung Soon Jang, and Claire J. Tomlin. The Standard Testbed of Autonomous Rotorcraft for Multi Agent Control (STARMAC).In Proceedings of the 23rd Digital Avionics Systems Conference, 2004.
8. 聂博文. 微小型四旋翼无人直升机建模及控制方法研究[D].国防科学技术大学,2006.
9. 左轩尘,韩亮亮,庄杰,石琪琦,黄炜. 基于ROS的空间机器人人机交互系统设计[J]. 计算机工程与设计,2015,(12):3370-3374.
10. 佘元博. 基于ROS的交互型自主导航机器人平台的设计与实现[D].南京大学,2015.
11. 常国权,戴国强. 基于STM32的四轴飞行器飞控系统设计[J]. 单片机与嵌入式系统应用,2015,(02):29-32.
12. 蒙博宇.STM32自学笔记[M].北京: 北京航空航天大学出版社,2014.
13. 于雅莉,孙枫,王元昔. 基于多传感器的四旋翼飞行器硬件电路设计[J]. 传感器与微系统,2011,(08):113-115.
14. 万晓凤,康利平,余运俊等. 互补滤波算法在四旋翼飞行器姿态解算中的应用[J]. 测控技术,2015,(02):8-11.
15. M. Moghadam and F. Caliskan, "Actuator and sensor fault detection and diagnosis of quadrotor based on Two-Stage Kalman Filter," 2015 5th Australian Control Conference (AUCC), Gold Coast, QLD, 2015, pp. 182-187.
16. 吕印新,肖前贵,胡寿松. 基于四元数互补滤波的无人机姿态解算[J]. 燕山大学学报,2014,(02):175-180
17. 程学功. 四轴飞行器的设计与研究[D].杭州电子科技大学,2013.
18. 冀亮. 四旋翼小型无人飞行器[D].杭州电子科技大学,2015.
19. 何瑜. 四轴飞行器控制系统设计及其姿态解算和控制算法研究[D].电子科技大学,2015.
20. 乌仁别丽克. 基于串级PID控制算法的四旋翼无人机控制系统设计与实现[D].东华大学,2016.
21. 邹桐. 浅谈PID控制算法对四轴飞行器的重要性[J]. 电脑知识与技术,2015,(24):177-178.
22. A. L. Salih, M. Moghavvemi, H. A. F. Mohamed and K. S. Gaeid, "Modelling and PID controller design for a quadrotor unmanned air vehicle," 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, 2010, pp. 1-5.
23. 石朝霞,胡兴柳,姜思辰等. 四旋翼无人飞行器串级PID控制系统研究[J]. 滨州学院学报,2016,(04):10-16.
24. 陆伟男,蔡启仲,李刚等. 基于四轴飞行器的双闭环PID控制[J]. 科学技术与工程,2014,(33):127-131.
25. 李俊,李运堂. 四旋翼飞行器的动力学建模及PID控制[J]. 辽宁工程技术大学学报(自然科学版),2012,(01):114-117.
26. Mutiwii. Multiwii Serial Protocol [EB/OL]. http://www.multiwii.com/wiki/

index.php?title=Multiwii\_Serial\_Protocol

1. 鲁恩萌. 智能空中机器人系统的设计与实现[D].哈尔滨工业大学,2015.
2. Mike Purvis. rosserial/Overview/Protocol [EB/OL]. http://wiki.ros.org/ross

erial/Overview/Protocol

1. Shibhansh Dohare. Arduino IDE Setup [EB/OL]. http://wiki.ros.org/rosserial

\_arduino/Tutorials/Arduino%20IDE%20Setup

致谢

时光荏苒，岁月如梭，不知不觉在东北大学的四年学习生活即将结束，在此向一路走来陪伴我成长、学习、进步的各位亲人、老师、同学致以最真诚的感谢。

在此毕业设计（论文）完成之际，谨向我的指导教师陈刚副教授致以衷心的感谢。在上陈刚老师的课的时候，我就被陈老师严谨的治学态度和渊博的学识所折服。感谢陈刚老师在我毕业设计期间给予的悉心指导、热忱帮助和大力支持，感谢他不辞辛劳一次次帮助我发现问题、解决问题。

其次，还要对我的三位同学步鹤、胡野、刘东旭表示特别感谢，在四年的大学学习生活中，他们在很多方面给予我很大的帮助和宝贵的支持。另外，还要感谢东北大学“T-DT”机器人队的吴晨豪和倪学斌两位学长，感谢他们在这期间对我的无私帮助以及感谢在同一实验室共同学习、工作的学长学姐。