



华中科技大学

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY



## 《硬件课程设计》报告

课题名称：微型无人飞行器系统设计与应用实现

院系：电子信息与通信学院

专业：电子信息工程

班级：电信 1204 班

姓名：黄衍

学号：U201213468

## 1. 立项说明书摘要

本项目为华中科技大学电子信息与通信学院 2012 级硬件课程设计项目，项目名称为微型无人飞行系统的设计和应用。项目通过设计实现微型无人机飞行器系统和与之相关的信息处理模块，实现飞行器姿态控制起飞与降落、按手动控制通过无线通信模块实现无人飞行器计模块实现定点旋高。

四轴飞行器是一种利用四个旋翼作为飞行引擎来进行空中飞行的飞行器。进入 20 世纪以来，电子技术飞速发展四轴飞行器开始走向小型化，并融入了人工智能，使其发展趋于无人机，智能机器人。

四轴飞行器不但实现了直升机的垂直升降的飞行性能，同时也在一定程度上降低了飞行器机械结构的设计难度。四轴飞行器的平衡控制系统由各类惯性传感器组成。在制作过程中，对整体机身的中心、对称性以及电机性能要求较低，这也正是制作四轴飞行器的优势所在，而且相较于固定翼飞机，四轴也有着可垂直起降，机动性好，易维护等优点。

**关键词：**四轴飞行器、PID 控制算法、姿态数据解算。

### 1.1. 项目概述

项目初步内容为设计并实现一个微型无人飞行器系统以及与之相关的信息处理模块。

该微型无人飞行器能够根据外部传感器与控制器输入的信息，实现姿态自稳控制、稳定起飞与降落、按手动控制、利用超声波探头与气压计模块实现定高悬停控制、利用摄像模块实现飞行航拍图片与视频。

系统硬件方面采用 STM32F407DISCOVERY 开发板为核心处理器。该处理器内核架构为 ARM-Cortex-M4, 具有高性能、低成本、低功耗等特点。其他硬件模块用有、无线蓝牙模块、MPU6050 运动数据获取模块、超声波模块、无刷电机调节器（4 个）、无刷电机（4 个）。

软件控制方面。1. 利用获取到的 XYZ 三轴的加速度，以及系统与 XYZ 三轴之间的夹角，采用四元素解算算法，获得系统当前的运动姿态，即 ROL(翻滚角)、PIT(俯仰角)、YAW（方向角）。2. 采用 PID 自稳控制算法，使系统稳定的达到相对稳定的平衡状态。

### 1.2. 项目功能指标

1. 根据外部传感器与控制器输入的信息，实现姿态自稳控制、稳定起飞与降落；
2. 按手动控制或利用超声波模块实现低空的定高悬停控制。

## 2. 四旋翼系统基本介绍

### 2.1. 四旋翼飞行器简介

小型无人飞行器可分为固定翼和旋翼机两种。固定翼飞行器出现的时间较早，技术相对成熟，但是相对于某些需要在较小空间内执行任务的场合，固定翼飞行器机动性差的缺陷往往使其无法胜任，而旋翼飞行器则弥补了这一缺陷，由于体积小，轻便灵活，旋翼飞行器可以在很小的空间实现垂直起降、全方向飞行及目标上空盘旋，故在执行监视和侦查类任务时，旋翼式飞行器更占优势。

四旋翼飞行器属于旋翼机的一种，相比于其他种类的飞行器，四旋翼飞行器采用四个独立电机驱动，螺旋桨数目多，能产生更大的升力，且其机架倾角固定，在飞行过程中不需要通过调节倾角来改变飞行速度及姿态，简化了飞行器的结构，减少了自身重量。飞行器四个旋翼产生的推力能更好的实现飞行器的静态盘旋，可以在悬空静止的状态下瞬时改变其姿态，有高度的机动性和有效承载力。四旋翼飞行器的诸多优势，使得其成为目前人们研究的热点。

### 2.2. 四旋翼飞行器基本结构

四旋翼飞行器采用了轴对称的基本布局，其机械结构简单，四个旋翼均匀分布，处于同一高度平面，垂直装在机体轴上，中间装载必要的机用设备，如图 2-1 所示为某四旋翼飞行器的基本结构图。四旋翼飞行器的基本结构一般保护如下几部分：

#### 1) 升力系统

四旋翼飞行器的升力系统一般由四个完全相同的子系统构成，轴对称分布于四旋翼飞行器的四个角上。每个子系统包括一个直流无刷电机、一个旋翼。全部升力有四个旋翼提供，通过改变四个旋翼的转速，就可实现飞行器俯仰、滚转、偏航等运动。

#### 2) 动力系统

动力系统即能量供应系统，由于动力锂电池具有重量轻、体积小和能力高等特点，一般四旋翼飞行器采用 3S 的动力锂电池供电，并通过无刷电机电子调速器来驱动四个直流无刷电机。

#### 3) 飞行控制系统

一般四旋翼飞行器的控制系统安装在机体中心部位，主要有飞行控制的控制器和包括惯性导航元件在内的各个传感器器件。鉴于四旋翼飞行器体积小特点，选择飞行控制器和传感器时需要充分考虑各器件的体积和重量，并且为增加飞行系统的续航时间，器件功耗也必须考虑。

#### 4) 机体

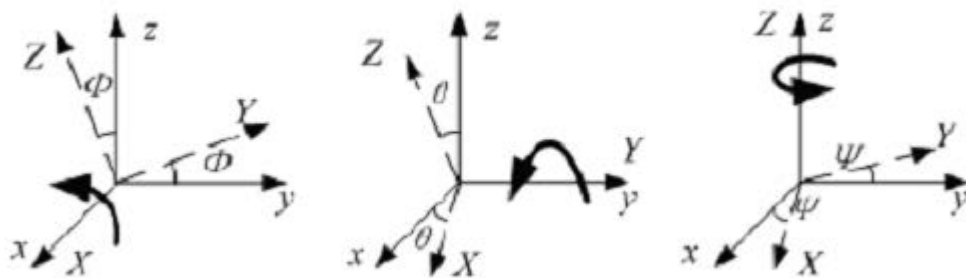
四旋翼飞行器的机体主要由两部份组成：○1 位于机体的中心平台，用于搭载控制器和相关传感器设备；○2 用于固定电机和螺旋桨的机体轴。此外还可包括起落架、桨保护架等。

### 2.3. 四旋翼飞行器的飞行原理

传统的直升机相比，四旋翼飞行器有着自己独特的地方，其呈十字平均分布的旋翼取代了传统的单独的旋翼，四个旋翼分别对机身产生单独的力和力矩，通过改变四个旋翼转速的大小，即可实现四旋翼飞行器各个方向的运动（俯仰、翻滚、偏航等）。

### 2.4. 四旋翼飞行器的动力学分析

在空中描述飞行器的俯仰、偏航、横滚等姿态信息时，需要引入空间的三维坐标系，便于使用空间矢量变换对飞行器的航行姿态进行数学描述。在空间中定义三个不同的三维坐标系，分别为惯性参考坐标系（导航坐标系）、地理坐标系和机体坐标系，其中惯性参考坐标系表示在绝对空间中禁止不动或匀速直线运动的参考坐标系，地理坐标系是用于确定物体在地球上位置的坐标系，机体坐标系为与飞行器固连的坐标系。



其中  $\phi$  ——沿  $x$  轴方向的滚动角（rad）；

$\theta$  ——沿  $y$  轴方向俯仰角（rad）；

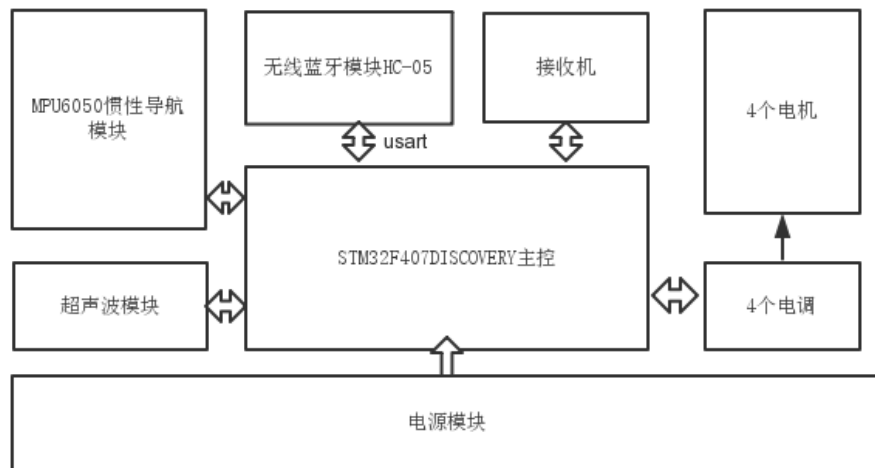
$\psi$  ——沿  $z$  轴方向偏航角（rad）。

## 3. 四旋翼飞行器的硬件系统设计

四旋翼飞行器具有载荷轻、体积小、完全自主飞行、控制复杂等特点，根据这些特点得出飞行控制的系统硬件设计的总体要求包括：材料轻型化、遥控设备可靠性高、传感器性能好、处理器速度快并具有良好的可靠展性和抗干扰性。在具体的方案设计过程中主要从以下方面考虑：可靠性、可行性、实时性和高集成度。为此在保证可行性的基础上尽量采用比较成熟的、可继承性的和可借鉴的技术和元器件。

下面首先给出系统硬件总体结构，然后介绍各个分模块硬件的选型和电路设计，主要包括飞行器微处理器、传感器、电机及电机驱动、无线数据传输模块及电源等。

### 3.1. 飞行器总体硬件结构



本系统的硬件总体框图如上：飞行器主控板上有：主处理器采用STM32F407DISCOVERY开发板、无线蓝牙模块（用于串口调试）、电机启动模块、电源管理模块等；遥控使用商品遥控及接收机。

MPU6050 惯性导航模块和主控间采用 IIC 总线连接，无线蓝牙模块和主控件采用串口 usart 连接，实现串口透传，便于系统的调试。

### 3.2. 控制器

控制器是控制系统的核心器件，起到协调和控制其他各模块的作用。它在每个控制周期内实时处理惯性测量装置（陀螺仪、加速度计和电子罗盘）输出的信息，完成一系列复杂的控制算法，得到四旋翼飞行器的姿态和位置信息，结合接收到的遥控器控制指令，计算出控制量，转化为相应的 PWM 信号通过电子调速器驱动四个无刷电机工作，保持四旋翼飞行器稳定飞行。

STM32 系列是意法半导体公司（ST）专为要求高性能、低成本、低功耗的嵌入式应用专门设计的基于 ARM Cortex-M4 内核的微处理器。本系统采用的正是STM32F407DISCOVERY开发板。其时钟频率达到 168MHz，具备丰富的外设资源，功耗较低，满足系统的所要求。

### 3.3. 惯性导航模块

惯性导航模块是飞机、航天器、舰艇和导弹采用的惯性导航系统的主要组成部分，利用惯性测量装置采集到的数据通过航位推算可以追踪飞行器的位置。四旋翼飞行器因

受发动机振动及外界扰动的影响大，难以建立精确的数学模型。由于任务载荷有限，四旋翼飞行器都以低精度 MEMS 惯性器件作为惯性导航模块，即姿态测量装置。飞行姿态感测模块是感知四旋翼飞行器的飞行状态的信息，如俯仰角、翻滚角、偏航角等。姿态感测部件是整个硬件系统的核心。本系统采用 MPU6050，感知飞行器的运动状态变化，主要使用 1 个 3 轴角速度计，1 个 3 轴陀螺仪。

### 3.4. 电机及电机驱动

电动四旋翼飞行器根据电机又可分为有刷电机和无刷电机类型。本系统选择电机为飞行器提供动力。

由于无刷直流电机与有刷直流电机及感应电机相比具有很多优势，如具有更好的转速和转矩特性、响应速度快、效率高、使用寿命长、噪声小、转速范围大、可靠性高等。基于以上优点，动力装置采用新西达 2212 的无刷直流电机。

本系统采用无刷电子调速器来驱动无刷直流电机。

电子调速器，简称电调。有无刷与有刷之分，本文由于驱动无刷直流电机，选用无刷电调。它根据控制信号调节电动机的转速，一般的连接情况如下：

- 1、电调的输入线与电池连接；
- 2、电调的输出线（无刷三根）与电机连接；
- 3、电调的信号线与控制器连接。

另外，电调一般有电源输出功能，有 5V 左右的电压输出，为舵机、微处理器等其他控制设备供电。本系统采用新西达无刷电调。

### 3.5. 通信模块

#### 3.5.1. 无线遥控器

通过遥控器可对飞行器进行手动的控制，向四旋翼飞行器传输飞行和任务控制指令，如控制飞行器的升降，前进后退，偏航等，同时也能作为一个无线的开关装置，使得调试更方便。本系统选用的为一四通道的无线遥控器。

#### 3.5.2. 无线蓝牙数传模块

无线通信模块是四旋翼飞行器的重要组成部分，它可以把四旋翼飞行器自身的状态信息下传给 PC 机，图获得偏航角，电机速度等，供调试分析使用，本系统选用一块蓝牙模块，用无线蓝牙转串口和电脑通讯。完成系统的调试工作。

### 3.6. 电源模块

电源作为飞行器的动力来源，其在系统中的地位是极其重要的，四旋翼无人直升机要稳定工作必须有稳定的电源供给作为保障，为系统的各个模块提供动力。稳定的电源可以使系统在各种环境下长时间稳定的工作，而如果电源模块设计得不够合理，那么就像在系统中埋下了一颗不定时炸弹，系统随时都可能因此而崩溃。本系统选用了一块 11.1V 锂电池，电池容量为 2200mah，可以满足系统需求。

## 4. 四旋翼飞行器软件设计

通过分析国内外一些研究成果发现：姿态控制是整个飞行控制的关键。因为小型四旋翼飞行器的姿态与位置存在直接耦合关系，如果能精确控制飞行器姿态，则很容易实现对其位置与速度的控制。虽然 PID 控制器有很多缺点，受到自身结构的限制，往往需要在稳态性能和动态性能之间进行取舍。但不可否认，PID 控制器是一种结构简单，稳定性好，工作可靠，调整方便的控制器，同时也是目前工业控制中应用最广，最为广大技术人员所熟悉的控制算法。

### 4.1. 软件开发环境

MDK 开发套件源自德国 Keil 公司，是 ARM 公司目前最新推出的针对各种嵌入式处理器的软件开发工具。MDK 集成了业内最领先的技术，包括  $\mu$  Vision5 集成开发环境与编译器。支持 ARM7、ARM9 和最新的 Cortex-M3 核处理器，自动配置启动代码，集成 Flash 烧写模块，强大的 Simulation 设备模拟，性能分析等功能，与 ARM 之前的工具包 ADS 等相比，RealView 编译器的最新版本可将性能改善超过 20%。本设计使用的 STM32 的程序便是基于 Keil-MDK 开发的。



### 4.2. PID 控制器简介

PID (Proportion-Integral-Differential) 控制器包含三种类型的控制，比例控制，积分控制和微分控制，图 4-5 PID 控制系统原理框图。

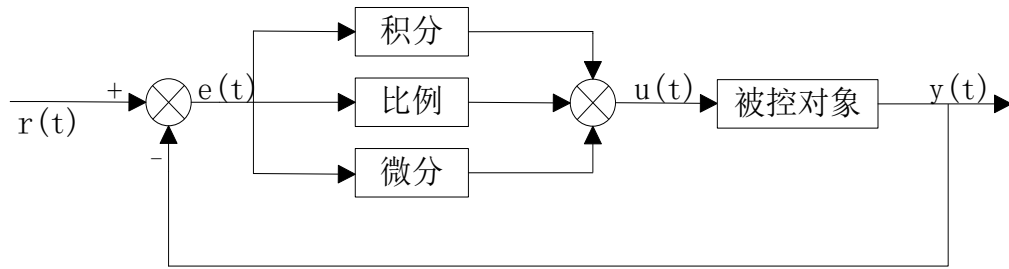
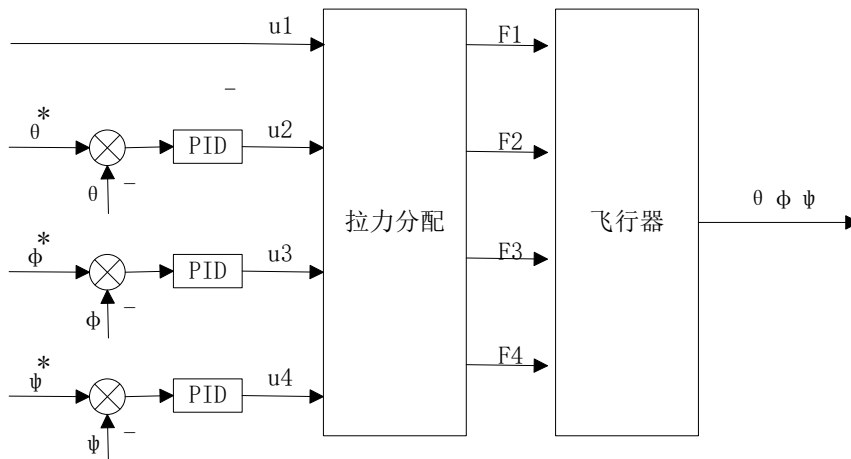


图 4-5PID 控制系统原理框图

图 4-5 中： $r$ —设定的期望值， $u$ —控制变量， $y$ —实际输出值， $e$ —控制偏差（ $e(t) = r(t) - y(t)$ ），是给定输出和实际输出之间的偏差。

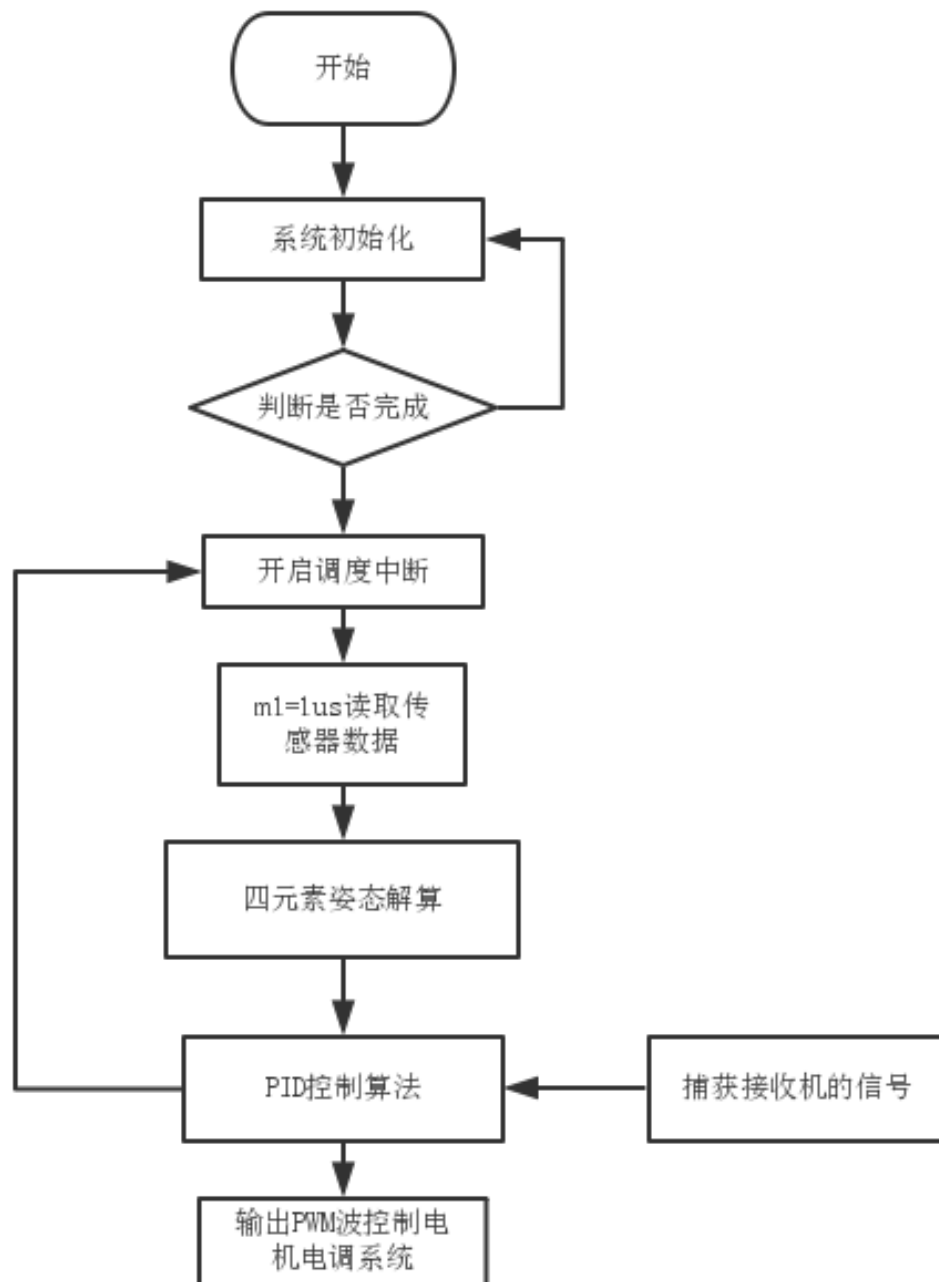
#### 4.3 基于 PID 控制器设计

在四旋翼飞行器的简化模型的基础上，设计了如下图的四旋翼飞行器的姿态控制的基本结构框图。





### 4.3. 程序整体运行流程

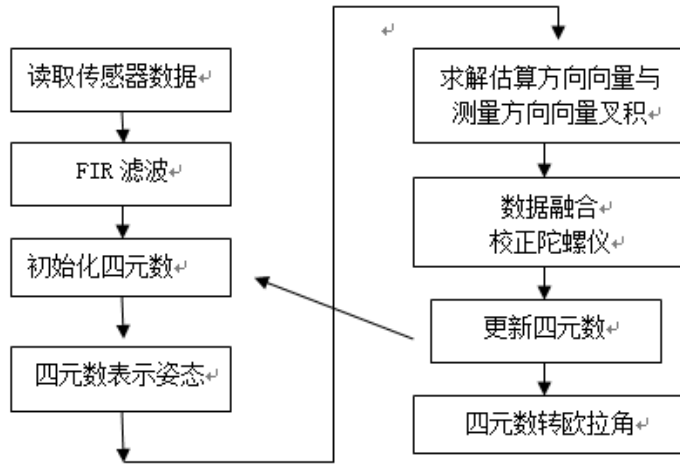


系统流程总体介绍：系统进入先进行外设的各个模块的初始化，当初始化完成后，可以运行 TIM3 调度中断，TIM3 为 0.5ms 一次，当计数器 M1 到达 2 及 1MS 时，获取一次传感器的数据，当 M2 计数达到 4 及 2MS 时，进行一次姿态解算和 PID 控制算法，控制 PWM 波的输出

占空比。

#### 4.3.1. 四元素计算姿态角的实现

根据前面给出的姿态解算方程与四元数，即可得到姿态计算系统的计算原理：



毕卡二阶算法为：

其中  $\Delta\theta$  为角增量  $q(n+1) = \left(1 - \frac{\Delta\theta_0^2}{8}\right)I + \frac{1}{2}[\Delta\theta]q(n)$  本设计基于互补滤波的思想上

完成的四元素算法，其核心思路为利用加速度测得的重力向量与估计姿态得到重力向量的误差来矫正陀螺仪积分误差，然后利用矫正后的陀螺仪积分得到姿态角。

首先不妨设处理后的加速度数据为： $ax, ay, az$ ，单位  $m/s^2$ 。加速度计的向量为  $(ax, ay, az)$  陀螺仪数据为： $gx, gy, gz$ ，单位  $rad/s$ 。陀螺仪向量  $(gx, gy, gz)$ ，可得由载体到导航坐标系的四元数形式转换矩阵为： $q_b^s$ 。根据余弦矩阵和欧拉角的定义，地理坐标系的重力向量，转到机体坐标系，是  $q_b^s$  中的第三列的三个元素，即  $q_b^s \cdot [0 \ 0 \ 1]^T$ 。

所以加速的向量与估计重力向量叉积： $\vec{e} = \frac{\vec{a}}{|\vec{a}|} \times q_b^s \cdot [0 \ 0 \ 1]^T$ 。

然后利用向量的叉积， $\vec{e}$  可视为误差向量，这个叉积向量仍旧是位于机体坐标系上的，而陀螺积分误差也是在机体坐标系，而且叉积的大小与陀螺积分误差成正比，正好拿来纠正陀螺。由于陀螺是对机体直接积分，所以对陀螺的纠正量会直接体现在对机体坐标系的纠正。用上面得到的结果校正陀螺仪： $\vec{\theta} = \vec{g} \cdot t + k \cdot \vec{e}$

此处  $k$  为一个常量系数。

再利用二阶毕卡法解四元数微分方程（4-6），更新四元数为下一

$$[\Delta\theta] = \begin{bmatrix} 0 & -\Delta\theta_x & -\Delta\theta_y & -\Delta\theta_z \\ \Delta\theta_x & 0 & \Delta\theta_z & -\Delta\theta_y \\ \Delta\theta_y & -\Delta\theta_z & 0 & \Delta\theta_x \\ \Delta\theta_z & \Delta\theta_y & -\Delta\theta_x & 0 \end{bmatrix} \quad \Delta\theta_0^2 = \Delta\theta_x^2 + \Delta\theta_y^2 + \Delta\theta_z^2$$

最后将四元数转变为欧拉角：

$$Q\_ANGLE.Pitch = \arcsin(-2 * q1 * q3 + 2 * q0 * q2)$$

$$Q\_ANGLE.Roll = \arctan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2 * q2 + 1)$$

$$Q\_ANGLE.Yaw = \arctan2(2 * q1 * q2 + 2 * q0 * q3, -2 * q2 * q2 - 2 * q3 * q3 + 1)$$

## 5. 项目完成情况

### 5.1. 项目初始预期

使飞行器能够平稳地飞行，通信使用我们自主开发的协议，并且能实现摄像头的摄像功能。

### 5.2. 项目初始分工

黄衍：专心 PID 控制算法和电机控制,不用考虑硬件底层

张哲：实现 NRF 射频模块数据的传输

沈钊：购买遥控器，并写好 stm32 代码以测量占空比，作为控制信号

武威：研究摄像头模块，写好驱动，获取图像，为航拍做准备

钱鹏志：写好 MPU6050（实际上只是调试和移植）的驱动，做好数据接口

任奕臻：一开始我不知道有这个组员，没有给她分工

### 5.3. 项目实际分工

钱鹏志：调好了 MPU6050，但是只得到了原始数据，而且没有做工程规范下的移植。帮忙焊好了主板，这算是不错的贡献，但是主板走线比较差。

张哲：放弃了 NRF，先被组长委任调试 stm32UART 接口，成功，后被安排做好工程移植，浪费了一整天的时间！失败。然后和组长一起调试 UART 蓝牙模块，在 win7 系统下调试成功，在组长的指导和督促下完成了数据的读和写，期间调试花了一晚上（通宵）的时间，然后放弃了课设，回家了。注：PWM 输出模块并不是他做的。

沈钊：购买了遥控器，并用示波器测量了占空比变化范围，调试了别人的代码，没有成功，没有做好相关驱动的移植，没有遵守工程规范。然后让他做存储器（实际上是调试现有代码），发现了存储器的误存现象，并认真记录了存储 0~255 时所有可能的错值

情况。和组长（我）交流，组长认为可以设计映射算法，用两个字节存一个字节，实现无差错存取，他认为太难，没有做。

武威：一直处于打酱油状态，因为忙于艺术团（貌似），所以我当没有这个人。项目火烧眉毛的时候，加入团队，协助张哲的调试，但是并没有起太大作用，倒是性格还行，嘻嘻哈哈，缓解团队压力。

任奕臻：不得不说是我最佩服的女生之一，也是我见过的女生中最会写代码的，毕竟团队（联创）出身。也是项目最后几天加入，被安排做摄像头和 LCD，以及接任沈钊的存储器纠错映射算法工作。非常有耐心，善于查阅文档，虽然最后摄像头没有完全调好，但是也获取到了图像，并且调试好了 LCD，能够显示图片（之前我做了只能显示文字）。存储器算法实现情况组长未知（各忙各的，没太多时间交流）。

黄衍：定义了工程规范，建议项目源文件和 stm32 驱动库分开，分文件夹。尝试教大家使用 github，但是全部没有学会，导致代码传递不便，以及工程管理混乱，各自用各自的模板，几乎没人遵守我一开始定下的规范，导致模块整合难度巨大。做了大量的工作（本来很多本该组员做的，都做了）。项目成果中，除非指明，则都是我做的。








## 6. 项目成果

### 6.1. 工程规范

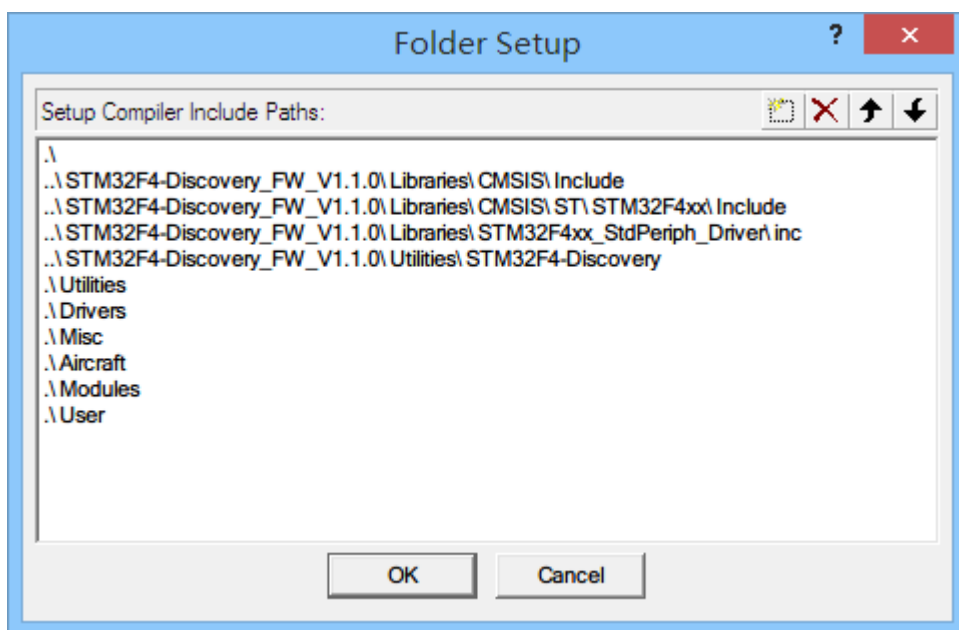
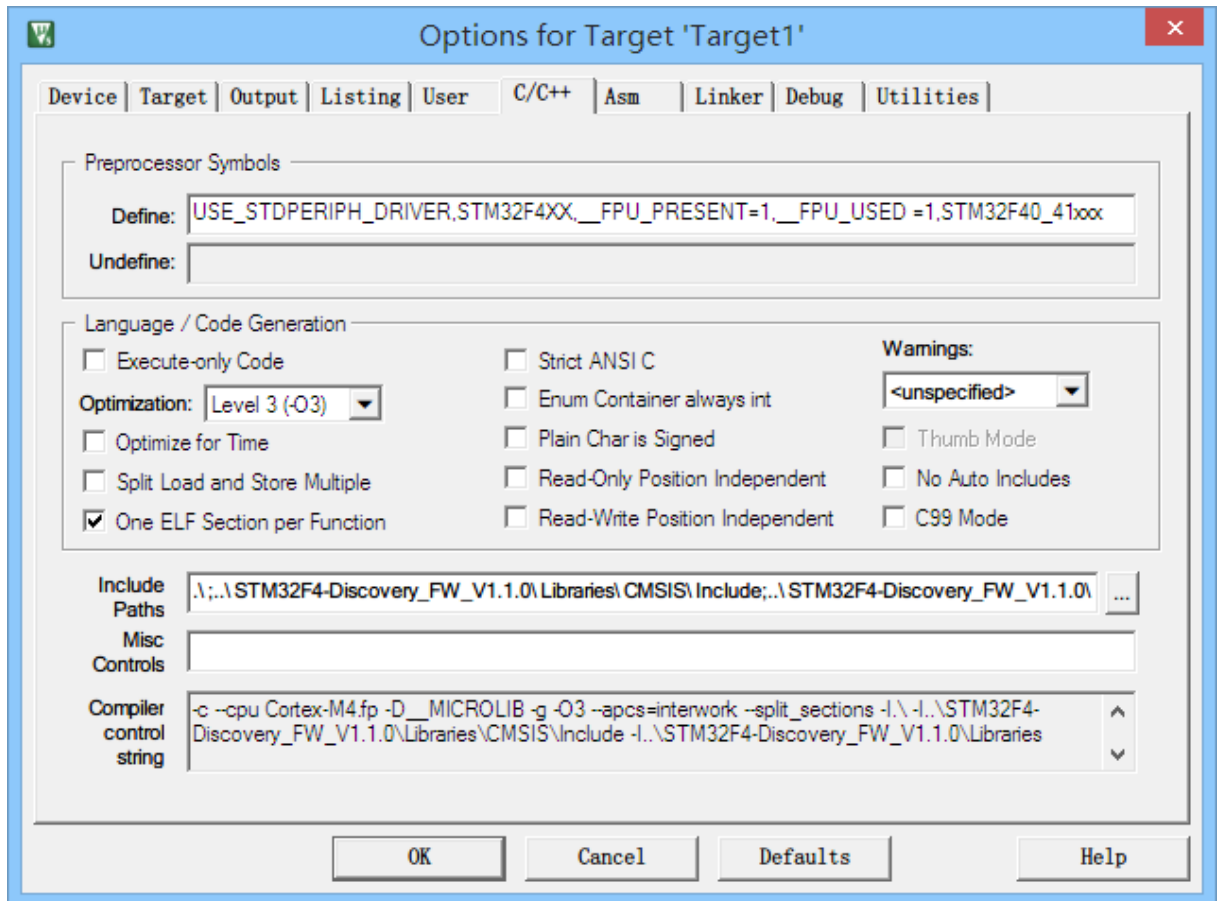
#### 6.1.1. 工程文件（Keil5 的.uvprojx 文件）规范

网上给的很多“小白”式模板，我个人评价是，用来很方便地玩玩可以，但是及其不适用于开发。首先，我们组中有人用这个模板，该模板把所有驱动库的源文件都导入了工程，严重违反了“只 include 必要的，只编译必要的”的开发原则，导致每次编译都要浪费大量的时间（编译了很多很多不必要的时间），张哲、钱鹏志、沈钊全都是这样，武威没有加入代码开发，只有任奕臻听了我的话，服从了我的建议，使用了我的模板。我发现大家用错了模板的时候，已为时晚矣（那是星期四晚上，就是老师打电话申请延期的前一天凌晨，快被气哭），也明白了为什么大家效率如此低下（单是编译浪费的时间就可想而知），可是我一开始就反复强调了规范，他们只顾着调试别人的代码，连工程移植全部都没做好，最后各个模块完全没有办法合，只有任奕臻同学做得还可以。

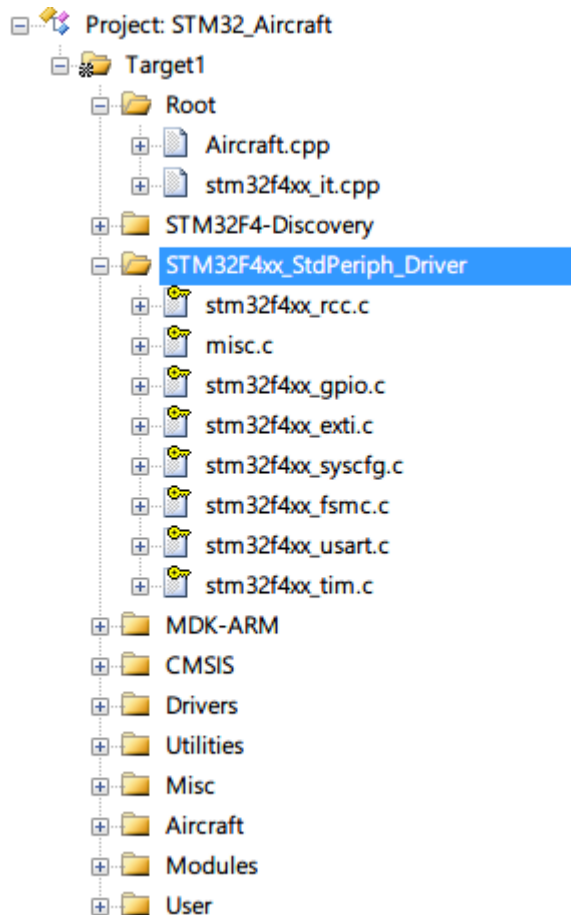
我定义的工程规范/模板如下（中途修改了很多，现在是最终版本）：

_Keil (15)		
D:\		
 SST51	2015/7/2 22:29	文件夹
 STM32_Aircraft	2015/7/6 1:50	文件夹
 STM32_JY901	2015/7/4 19:55	文件夹
 STM32_MPU6050_To_JY901	2015/7/5 16:28	文件夹
 STM32_Nandflash_Camera	2015/7/2 14:56	文件夹
 STM32_USART3	2015/7/3 23:06	文件夹
 STM32F4-Discovery_FW_V1.1.0	2011/10/28 10:38	文件夹

源码全部放在 D 盘根目录的 \_Keil 文件夹里面，方便别的队友在你的电脑上找代码。工程文件名和工程目录名保持一致，比如 STM32\_Aircraft，工程文件名叫 STM32\_Aircraft.uvprojx。驱动库 STM32F4-Discovery\_FW\_V1.1.0 与工程目录并列（而不是网上那样做的，把驱动库直接全部塞到工程里面去！），为了使工程能找到驱动库的头文件和源文件，在工程中设置 include paths:



驱动库源文件，只有在用到的时候加入工程（现在应该还是有冗余，但是至少比把所有源文件添加进去强!）：



编程规范，其实只有我一个人写原创代码，张哲在我的指导下写了一小段代码（用 USART 接收字符串，因为原生函数只能接收单个字符），遵守 Google 的 C++编程规范（<http://zh-google-styleguide.readthedocs.org/en/latest/>）。

以上规范，实际上只有我一个人遵守了。所以后面我要把所有人的模块都移植过来，工作量巨大。

### 6.1.2. 源代码规范

Keil 的工程源代码的文件结构如下表：

源文件组	Root
物理路径(相对于工程文件)	.\
包含的文件	Aircraft.cpp, stm32f4xx_it.cpp
描述	系统顶层源文件，stm32 的中断设置文件

源文件组	STM32F4-Discovery
物理路径(相对于工程文件)	..\STM32F4-Discovery_FW_V1.1.0\Utilities\STM32F4-Discovery\
包含的文件	stm32f4_discovery.c
描述	discovery 开发板应用源文件（实际没有多大用处）

源文件组	STM32F4xx_StdPeriph_Driver
物理路径(相对于工程文件)	..\STM32F4-Discovery_FW_V1.1.0\Libraries\STM32F4xx_StdPeriph_Driver\src\
包含的文件	stm32f4 外设函数库，包括 gpio、timer、usart 等，实际开发时用到哪些就把哪些加入工程（工程规范中提到的）
描述	stm32f4 外设函数库

源文件组	MDK-ARM
物理路径(相对于工程文件)	..\STM32F4-Discovery_FW_V1.1.0\Libraries\CMSIS\ST\STM32F4xx\Source\Templates\arm\
包含的文件	startup_stm32f4xx.s
描述	stm32f4 系列引导程序源文件（汇编）

源文件组	CMSIS
物理路径(相对于工程文件)	.\
包含的文件	system_stm32f4xx.cpp

描述	stm32f4 系列系统配置文件，包括时钟的设置，FPU 打开的选项等等
----	--------------------------------------

源文件组	Drivers
物理路径(相对于工程文件)	.\Drivers
包含的文件	Pin.cpp, Timer.cpp, PWM.cpp, IIC.c
描述	驱动层源文件，引脚类（用于配置引脚），定时器类（用于配置定时器工作状态），PWM 类（封装定时器类（PWM 输出状态），PWM 波各个属性），IIC 协议的软件实现

源文件组	Utilities
物理路径(相对于工程文件)	.\Utilities
包含的文件	LCD.cpp, AsciiLib.cpp, BluetoothCmd.cpp
描述	LCD 模块的驱动文件，ASCII 码字库，蓝牙串口命令解释器/执行器类  注：LCD 只是调试的时候用到，最终飞行器调试时已弃用

源文件组	Misc
物理路径(相对于工程文件)	.\Misc
包含的文件	Formatter.cpp, myDelay.cpp
描述	自己写的数据类型与字符串相互转换函数（也是只有在调试 LCD 时用到），自己定义的阻塞式延迟函数（用于电调的初始化）

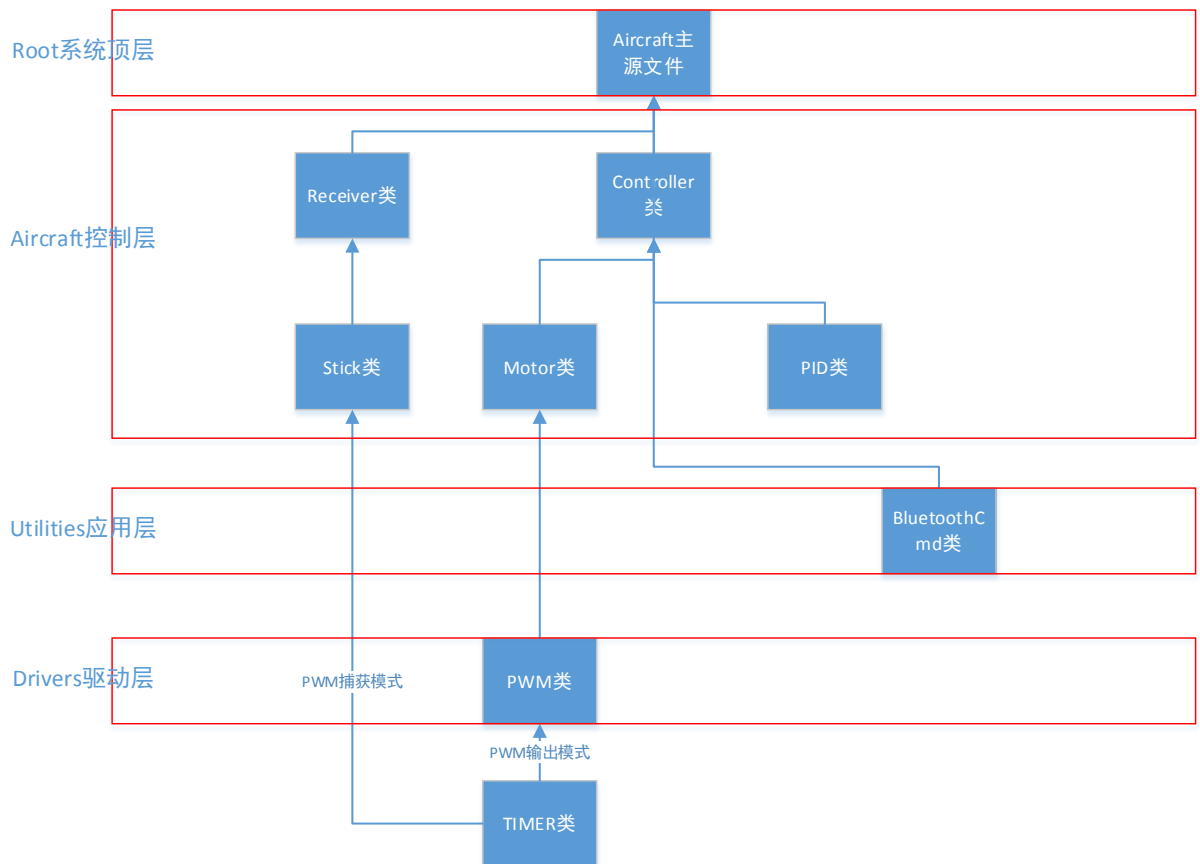
源文件组	Aircraft
------	----------



物理路径(相对于工程文件)	.\Aircraft
包含的文件	Receiver.cpp, Stick.cpp, Controller.cpp, PID.cpp, Motor.cpp
描述	<p>飞行器控制层源文件，包括：</p> <p>接收器类（四个 Stick 成员），单个通道遥控接收的 Stick 类，系统控制器 Controller 类（内置三个角度 PID 成员，并处理设定角度与 PWM 占空比的转换），电机类（内置 PWM 类，针对电机特性做 PWM 占空比限幅、启动检测等处理）</p>

## 6.2. 已经做好了

模块框架图（分层、分模块）：



注 1: 没有按照 UML 的规范来【10】(见《疯狂 JAVA 讲义》P31), 只是大致的示意

注 2: 严格意义上, PID 类也是跨平台通用的, 属于应用层, 但是本项目为了维护和调试方便, 放在了 Aircraft 控制层

### 6.2.1. PWM 波捕获, PWM 输出, 合作式调度器 (都和定时器有关)

本来 PWM 波的捕获是沈钊做的, 但是他用的别人的代码, 没有做好调试整合, 所以最后他做的部分我没有用到。

在星期二 (第一次验收) 回来, 到星期三, 我花了一天半时间熟悉了 stm32 的驱动库 (之前什么都不会, 因为我的预期是我只要写好控制层算法即可, 因为算法可以是跨平台的, 前提只要封装好底层驱动), 封装了一个 Timer 类, 可以一键设置定时器工作模式, PWN\_IN, PWN\_OUT 和单纯的中断合作式调度器 (scheduler) 模式。然后又花了大半天调试, 最终成功捕获到了遥控器的占空比。

### 6.2.2. 相关数据的接口和处理

测得了遥控器各个通道的占空比变化范围 (约 0.05~0.10), 并定义了一个 Stick (摇杆) 类, 进行了分类, 有的占空比数据有平衡点, 并且应该有正负 (比如 pitch、roll 和 yaw), 把平衡点校准和归一化到 0, 把其他值 (0.05~0.075, 0.075~0.10) 分别归一化到 -1~0 和 0~+1。

### 6.2.3. PID 控制器算法

分别实现 PID 类和 Controller 类。为什么要面向对象? 可能 C++ 的效率没有 C 高, 但是我认为 stm32 并不在乎这一点效率, 很多高级的板子 (TI 的 ARM——Tiva 系列, stm32 的 f4 系列) 都是 Cortex-M4 的, 使用的 ARMCC 编译器, 而 ARMCC 是支持 C++ 的, 只是有一点要注意, 在所有源文件为 cpp 的头文件中, 加入:

```
#ifdef __cplusplus
    extern "C" {
#endif
```

```
#ifdef __cplusplus
}
#endif
```

意思是把 C++ 混编成 C【2】。

使用 OO 之后, 虽然代码量多了点, 但是封装性很好, 可移植性和可维护性都大大提高, 又可以结合函数式编程风格, 使代码变得简洁、美观, 又不失逻辑性和可读性。(从上面的 UML 图也可以看出来)

网上的飞行器的 PID 算法大多是基于特定硬件，封装性和可维护性极差！这也是我坚持自己造轮子的原因。

PID 类头文件如下：

```
/**
 * @Filename: PID.cpp
 * @Author: Kelly Hwong
 * @Update: 2015.6.27
 * @Description: 跨平台通用类，PID 反馈控制器
 */
#ifndef PID_H_
#define PID_H_

#ifdef __cplusplus
extern "C" {
#endif

class PID {
private:
    // 调度设置
    /* PID 类不负责调度设置，调度设置外外部 Control 类中实现，PID 只要负责自己的事情
    PID 自己的事情就几个，输入期望（比如角度值），输入测量值，Eval 误差
    ( error_pre_error_error_d_error_i_ )
    输出反馈值（还是角度）
    unsigned int interval_unit; // 调度时间单位，in mS，用来各个输出计算
    // 与调度器有关
    unsigned int interval_ticks; // 调度间隔，in ticks
    unsigned int interval_counter; // 调度计数
    */
    // 控制设置
    float Kp_;
    float Ki_;
    float Kd_;
    float dt_; // 时间间隔
    // 反馈设置
    float setpoint_;
    // 误差计算
    float pre_error_; // 上一个误差
    float error_; // 误差
    float error_d_; // 误差的差分
    float error_i_; // 误差的积分
    // 输出
```

```

float pout_; // P 路输出
float iout_; // I 路输出
// 积分限幅
float imax_;
float imin_;
float dout_; // D 路输出
float out_; // 总输出

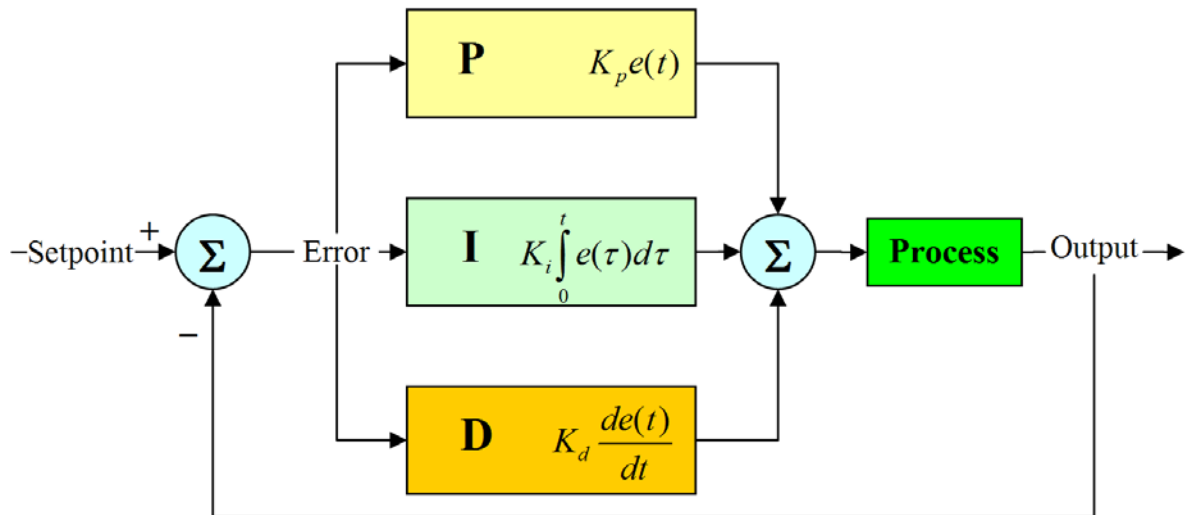
void EvalError(float measured_value);
void EvalOut(void);
inline void limit_i();
public:
    PID();
    PID(float Kp, float Ki, float Kd, float dt);
    PID(float Kp, float Ki, float Kd, float dt, float imin, float imax);
    void Routine(float measured_);
    void setpid(float Kp, float Ki, float Kd);
    void set_i_limit(float imin, float imax);
    void setpoint(float sp);
    // get 输出
    float setpoint(void);
    float pout(void);
    float iout(void);
    float dout(void);
    float out(void);
}; // class PID

#ifdef __cplusplus
}
#endif

#endif

```

参 考 了 维 基 百 科 -PID 控 制 器  
 (<https://zh.wikipedia.org/wiki/PID%E6%8E%A7%E5%88%B6%E5%99%A8>)【3】的定义:



PID 就应该只做 PID 的事情，接收 setpoint，计算误差和计算三个通道的输出和总输出，setpoint 接口和测量接口都应该交给外部模块去做（本工程中是 Controller），而不是像很多代码那样把误差计算和反馈输出写在一个函数里面，尤其是那种几个 PID 写在一起，可移植性和模块化程度及其糟糕。

PID 类实现后，由 Controller 类（基于特定控制对象，非通用）去实例化 PID，Controller 类头文件如下：

```
/**
 * @Filename: Controller.h
 * @Author: Kelly Hwong
 * @Update: 2015.7.2
 * @Description: 飞行器姿态控制器
 */

#ifndef CONTROLLER_H_
#define CONTROLLER_H_

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include "Controller.h"
#include "PID.h"

class Controller {
    unsigned int interval_unit; // 调度时间单位，in mS，用来各个输出计算
    // 与调度器有关
```

```

    unsigned int interval_ticks; // 调度间隔, in ticks
    unsigned int interval_counter; // 调度计数
private:
    // 输入的油门 0.05~0.10
    float throttle_;
    // 调度器的参数
    float routine_freq_;
    float scheduler_tick; // 调度定时器的调度时间间隔, 单位 S
    float routine_flag_;
    uint8_t routine_flag_int_;
    uint8_t routine_counter_;
    // 设置的三个角度
    float pitch_setpoint_;
    float roll_setpoint_;
    float yaw_setpoint_;
    // 测量的三个角度
    float measured_pitch_;
    float measured_roll_;
    float measured_yaw_;

    // 控制器输出的电机的占空比, 然后要接口给主函数中的 Motor 类 ( setduty )
    float motor1_duty_;
    float motor2_duty_;
    float motor3_duty_;
    float motor4_duty_;
public:
    /* 只能 public 了, 虽然封装性不好, 但是不这么做不能初始化 */
    // 实例化的 PID 类
    PID pid_pitch;
    PID pid_roll;
    PID pid_yaw;
    Controller();
    Controller(float routine_freq, float scheduler_tick);
    void SetPoints(float p, float r, float y);
    void SetMeasures(float measured_pitch, float measured_row, float measured_yaw);
    void Routine(void); // 控制器控制例程 ( 用调度器调度 )
    uint8_t IsExecuted(void); // 返回给调度器
    float motor1_duty(void);
    float motor2_duty(void);
    float motor3_duty(void);
    float motor4_duty(void);

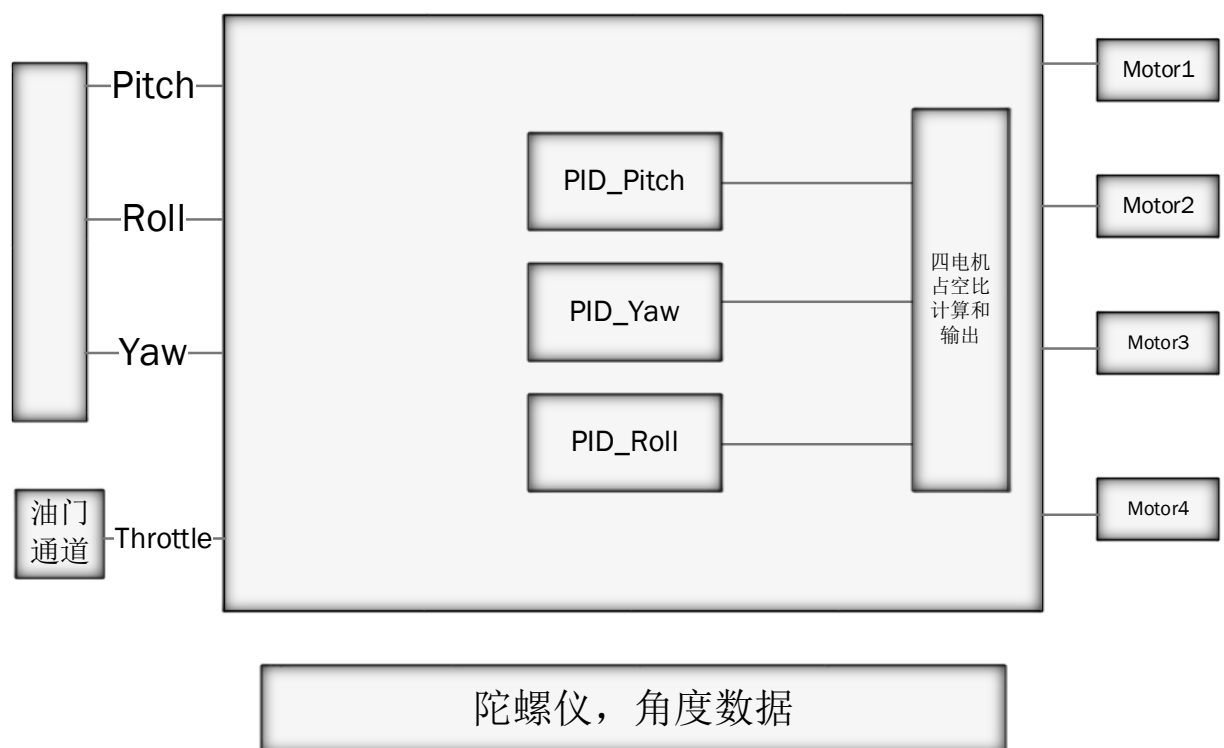
```

```
void throttle(float th);
}; // class Stick

#ifdef __cplusplus
}
#endif

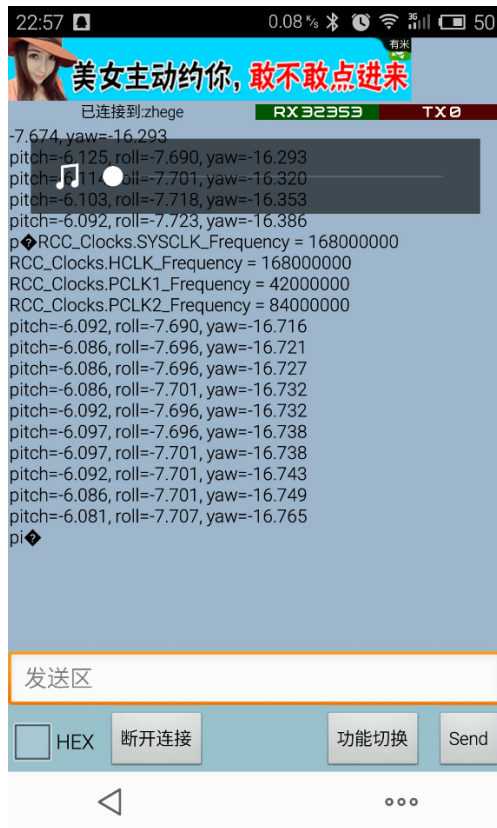
#endif
```

程序的统一对象模型（UML）如下图：



#### 6.2.4. 蓝牙模块

因为张哲只做了 win7 下的蓝牙串口（win8 兼容性问题，不支持这个蓝牙模块），所以我在我们班学委刘洋同学的帮助下，在星期六调好了安卓的接收机，可以当做无线串口助手，能够用来接受串口打印的数据（方便调试）。



同时, 我发现这个软件非常强大, 可以用来做上位机:



每个按钮定义了各自的帧格式, 点击按钮就可以把数据发送给串口。我们的 stm32 现在可以接受 USART 数据, 只是命令的解析还没有写。如果写好了就可以很快速地调节 PID 参数了。



### 6.2.5. 蓝牙串口命令

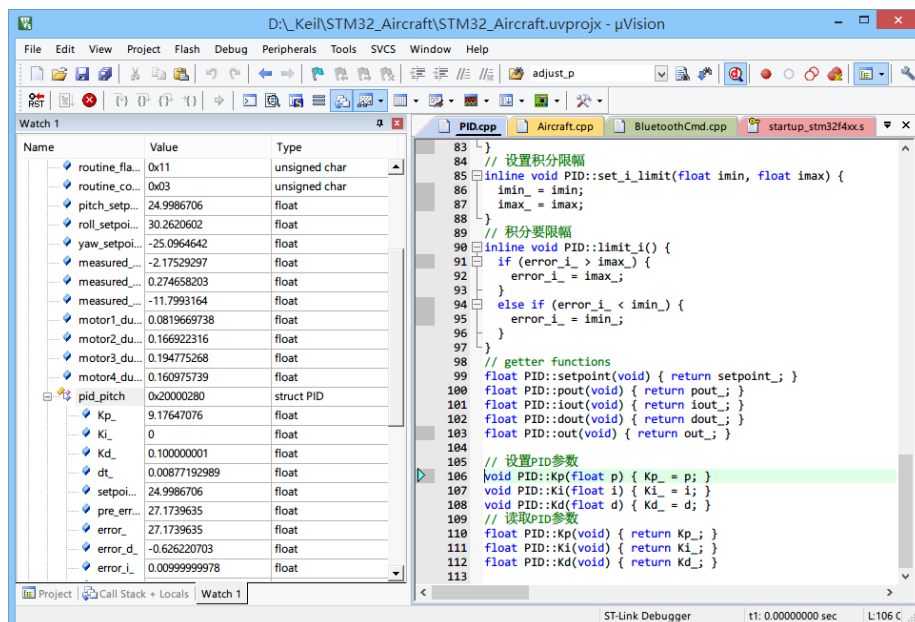
在撤回第一版本的文档后，又花了两三天时间做蓝牙串口命令的解释器&执行器（中途也是遇到了各种问题，看来什么事情都不是我想的那么简单）。

代码就不贴了，直接上效果图：

在安卓手机端滑动 PID.P 滑杆：



P 参数值设定在接近峰值，然后在 Keil 的在线调试端：



可以看到 pid\_pitch 的 Kp\_ 参数被设置为了 9.176，程序把 0~255（单字节无符号）线性变换到 0.0~10.0（浮点数），所以结果是正确的。

### 6.2.6. 陀螺仪

在星期六（实际上大家星期五全部都放弃了，接下来都是我一个人），把钱鹏志给的工程改成了我的工程规范，读出了数据，并做了坐标转换（各个轴正负设置），但是发现三个控制角度需要计算出来，比较麻烦。先向庄月清同学求助，他帮我把他自己的 JY901（带 MPU6050 并能硬件计算三个控制角度）移植到 stm32 上，但是失败了。

星期天，我找到电工基地的徐亚飞（机械学院，电工基地测控组）同学帮我焊上了他们以前用的模块（不知道叫什么，采样率 100Hz，能输出稳定的 pitch、roll、yaw 角度值）。数据见上文中蓝牙串口的接收区。

### 6.2.7. OV7670 摄像头模块

由任奕臻同学负责完成的，成果见她的报告【11】。

### 6.2.8. 1Gbit Nandflash 模块

由沈钊测试成功，并找到误写字符表（0~255 中的一小部分值），由黄衍提出纠错读写算法，沈钊同学放弃研究算法以及误写原因的查找和分析，由任奕臻同学接替，而任奕臻同学坚持先调试好摄像头模块（组长已反复强调先做简单的，字符映射处理而已，对于联创出身的人来说应该不难），最近几天又由于自己的私事，所以组长并不知道该模块调试如何。

## 7. 课设总结

注：小标题叫课设总结，只是课设结课了，但是我们的大创仍然会继续！

### 7.1. 个人经验总结

此次课设，我自觉受到了很多委屈，不过在老师的开导下，也想明白了很多事情。有的是开发经验和团队管理经验，而有的是人生经验和社会经验，我相信后者更为难得！

#### 7.1.1. 1+1 就一定>2？3 个臭皮匠就一定顶一个诸葛亮？从工程规范说起

我们知道 Keil 开发的是一个一个工程，每个工程有各自实现的功能模块，要把两个工程合二为一，必须保证工程的一致性（consistency），包括文件结构，include 目录设置。不得不说，虽然是组员们没有严格执行，也是我自己没有督促好，我应该从一开始就盯着他们写代码！不要用那些网上那些误导人的模板，而是用我自己的。可是我哪来的精力呢，自己这边都一大堆 bug 在调试，烦躁得不行，一边还要去管理他们。

#### 7.1.2. 如何开导思维未打开的同学？授人以鱼不如授人以渔

项目中期我打包了一个自己的最小工程模板（只有 Keil 项目文件，里面有 include 设置和文件组设置），交给组员，但是为什么大家没有执行？自我检讨，我只是逼着/强调他们去做，也许让他们看着我最后合模块痛苦的过程（可惜并不能，因为我星期五才开始合，那个时候他们已经不做了），他们就能理解。他们并不知道规范的重要性，我也没有花太多时间去解释，导致他们不能理解，而是固执于自己的习惯——别人都现成的

模板了，管他规范不规范，把我自己的模块调试好就完事儿了。

的确是交流不够。非团队（点、联创、冰岩，我本人呆过电工基地、点团队和数模基地）出身的人就很难理解多人合作应该遵守的规则。我应该多花一点时间，哪怕是半天，解释一下项目 A 和项目 B 怎么合起来的，他们就理解了，可是我没有。现在来看，与其自己后期填坑，不如在前期多花点精力，培养组员的工程意识。现在也不至于这么累。

### 7.1.3. 目标不同，怎么合作？

讲的是我择人眼光的问题，还是太天真。我系（院）大部分人都认为硬件课设很水，随便做做就完事儿了，看那些做小车的、做旋转 LED 的。但我不这么想，做硬件课设就是应该锻炼自己学习能力和动手能力的，做那种一两天就能学会的东西，最后能得到和收获什么呢？可是我们组中，相当一部分人，包括**比较优秀的张哲**同学，竟然持和普通人一样的态度，而我们的课题又非常难，抱着水一水的态度怎么可能做得好？而我自己也太傻，盲目信任队友，没有摸清他们的目的和做事的态度以及热情，到了星期五，队友都弃我而去的时候才追悔莫及。相反，庄月清同学就清楚自己组员的特长，做好了心里预期，自己一个模块一个模块地调，还免去了合多人工程的麻烦。多人合作确实应该做好分工（我理解错了老师说的“模块分工”），写代码的就一起写，而且写代码的人一定要**特别特别专业**，不是说懂 C 语言语法即可，而是要有强烈的工程规范意识。而擅长写文档的（比如张哲）就应该专门写文档，人脉广的（不过我们组人脉广的也就我自己）就应该专门去找人求助。

### 7.1.4. 能力越大，责任越大

我没有项目管理经验，特别是这样 6 个人的团队，我的规范也是我慢慢摸索出来的。由于我盲目信任别人，潜意识里面降低了自己的责任，到了后来担当起责任来就不堪重负。

如果我早一点认识到团队的能力（早在张哲在理应两三天调好却没有调好 NRF 模块的那几天，即两个多星期之前），就应该早早完成别的课程作业，加入团队，担当起抵住来。也许我能力不是最强的，但是在这个团队中，还只有我是最熟悉系统方案实现和 stm32 的开发的。

作为组员，我完成了大量的工作，对团队贡献也理应最大，但是作为 leader，在这次课设中，我是不合格的，没有尽到指导、引导组员学习的义务。

所以，一个优秀的 leader，不仅要自己学习能力强，接受新知识的能力快，还要善于引导和教育别人。而我，只做到了前者。

### 7.1.5. 不能闭门造车，善于寻求外援

我的个性是，遇到问题，不到自己解决之前，坚持自己查文档，自己学。我们团队也遇到了各种问题，比如一开始遇到的 STM32 没有与 PC 的串口接口的问题，张哲他们问了电工基地的人，于是买了 USB 转 TTL 模块，解决了这个问题，这一点就非常好。可是后续的开发，遇到了更多问题，而我们采取了“封闭开发”的方式（所有人关在寝室里，不准出去，因为我要盯着），反而失去了外援。

在最后的几天（交报告的上个周末），只有我一个人的情况下，我才意识到要去找外援，先后找了同级的庄月清同学，以及电工基地的徐亚飞同学。他们给了我很多帮助，很多问题也迎刃而解。

不得不说人是后验（马后炮）的生物，如果一开始我就多寻求一点支持和帮助，我们课设组不就会进行得更顺利么？

#### 7.1.6. 心胸宽阔，严于律己，宽以待人

我对自己要求很高，这一点要持续和保持，但是在合作过程中，我以对自己的要求要求别人，有时候非常不合适。有时候我批评组员这个不会那个不会，甚至发火，这是不对的，是我自己没有给他们安排好自己擅长的事情。此外，过程中我急躁了一些，弄得团队关系不融洽，也一定程度上影响了团队效率。我自己脾气和心理承受能力也差了一点，但经过此次课设，有了经验，也更好应对未来与他人合作中可能遇到的糟糕的情况了。

不得不说，此次课设，除了技术上的学习和进度（硬技能），还学到了很多**管理经验**和**社交经验**（软技能）！

### 7.2. 我对各个组员的评价（褒和贬）

#### 7.2.1. 张哲

典型的高 **GPA**，**低动手能力**，当初因为觉得他成绩好，学习能力**应该不差**和他组的队，结果却和我想的**完全相反**。不过还是我的问题，没有问清楚他的技能。他没做过嵌入式开发，我却一上来就给他一个“无线通信模块”的任务，他做不出来，也丧失了部分积极性。相反，他软件课设是安卓开发，我却拒绝了他“开发安卓上位机”的建议。

学习能力比较强，也有干劲，只不过遇到的挫折太多了，后期干劲不足。

很负责，一开始我不在的时候他坚持带组员去电工基地。中途缺材料的时候，也去了好几次广埠屯购买。比较擅长写文档，我们课设的开题报告、中期报告、结题整体报告都是他一个人写的。文档上，他为我们组做了**最大的贡献**。

#### 7.2.2. 武威

一开始貌似忙于艺术团，比较不积极。但也是我的问题，没有做好督促工作。

性格很开朗，无论进展好不好，都乐呵呵的，一定程度上缓解团队紧张的氛围。

擅长焊东西，可是我一开始不知道，最后他帮我拆了一个模块（因为要换），还干得不错。

#### 7.2.3. 钱鹏志

比较内向木讷，呆萌呆萌的。比较有耐心，认真调试好了 **MPU6050**（虽然最后我没用上），但是**没有做工程移植**。不懂得站在组长角度上思考，但是是组长的沟通力度不够。

#### 7.2.4. 沈钊

学习能力差一点，但是懂得变通，我们 NFR 无线模块做不出来的时候，是他提议购买遥控器和接收机，暂时避开无线通信的问题，毕竟太复杂。也多亏这个建议，我们的课设才有后续模块，否则就会一直卡在这里了。

后期很配合，调试好了存储器，但是拒绝改进算法。可能还是怕困难，对自己不够有信心，需要有人引导和鼓励。

#### 7.2.5. 任奕臻

和武威一样，后期才加入团队。联创团队出生，有一定的工程规范意识，用 Keil 的时候上手很快，立即在我的模板下做好了移植。

遇到问题有个不撞破南墙不死心的态度，调试摄像头遇到了各种问题，她从没向组长抱怨，而是自己查阅了大量的文档和资料（详见她的报告）。

但是不太懂得“先易后难”，组长劝了好几次，先做存储器的映射算法，因为相比调试 OV7670 的一百多个寄存器，这个容易得多，但是直到最后她也没有做出来，可能是对摄像头模块有很大的执念。

## 8. 项目未来

今天（2015 年 7 月 6 日星期一）是课设结题的最后期限了，但是我们的大创——微型无人飞行器及其监控系统设计才刚刚开始

### 8.1. 还可以继续做的

#### 8.1.1. 弥补课设遗憾

沿着课设的主线走，把蓝牙控制协议的解析器写好，相应蓝牙的串口控制字符串，改变 PID 参数，来进行飞行中调试，使得飞行器达到平稳飞行的效果。我的课设结了，我也要去实验室参加科研实习训练，但我也会利用空闲的时间（比如睡觉前），把这当兴趣，一点一点去做。

然后使用任奕臻的摄像头模块和存储器模块进行航拍和数据存储。因为有管脚被复用。所以计划方案是两块 stm32 的板子，四旋翼上面一个用来主控动作，下面再固定一个专门做航拍（以及远程数据传输）。

接手张哲对 NFR 模块的研究，或者购买串口 Wifi 模块（最高波特率 115200，速率可能慢一些）实现远距离数据传输（大于 50 米），把图像传回地面站（PC/安卓手机）。

#### 8.1.2. 大创的拓展和创想

注：大脑风暴，可能并不能实现

大创就是天马行空了，实现自己有趣的 idea，比如我们可以购买树莓派（一种微型移动电脑，可搭载 Linux 系统，近几年非常火），做一个便携式地面站。

我们还可以去玩 3D 打印，易子钦同学是我高中校友，在材料学院，黄亮老师则是研究塑性材料的，我们可以自己去对飞行器建模，打印自己设计的外形的飞行器。

国家/学校给了我们经费，不好好利用，不多学点知识，不多锻炼点技能，不多实现一些创意，怎么能行呢！？

## 8.2. 更好的团队

之前说了句得罪人的气话“就是我带着那两个提高班的学弟做也比你们强”，这是客观事实。首先我个人，因为此次课设经历，学到了一些管理经验：

1. 组内多交流，无论有多么忙，无论有多少 BUG
2. 不要组内闭门造车，多问组外厉害及有经验的大牛，你结识的人决定了你的能力的上界
3. 规范订制宜早不宜迟，在星期六和庄月清同学调试 MPU6050 的时候，我让邵逸飞学弟把官方的 Example——IOToggle 按照我的工程模板做好移植，他做好了，并且也初步熟悉了 stm32f4 的开发环境（毕竟提高班高强度的培养，能力的确比普通班的非团队出身同学强很多）
4. 多人合作一定要用 github 这样的平台！不推荐甚至不允许用 U 盘拷来拷去，谁知道你拷得对不对。找队友，首先第一条，会用 Github（这不是玩笑）。所幸，我的曾晨学弟和邵逸飞学弟都非常熟悉 Github（曾晨比我更早开始使用）。

然后团队成员，我自己可能不会花太多时间做了，主要指导两个学弟做。并且刚结识了电工基地的同学——徐亚飞（自己以前在基地的人脉），他拟作为我们的技术指导，并且他是机械学院的，用他换掉挂名的陈越同学（计院特优生）也是可行的。

## 9. 附：调试和开发日志

注：可能有些时间我记不清了，还有些日期我取的午夜之前的日期，因为经常通宵。

2015 年 6 月 30 日星期二：

- 构建 stm32f4 驱动类 Timer，实现了 pwm\_in()初始化设置函数/方法

2015 年 7 月 1 日星期三：

- 改进 Timer 类，实现了 pwm\_out()和 mode\_sch()初始化设置函数/方法
- 调试 Timer 类的各个方法，调试了各个定时器作为输入，利用了 tim2~5 作为输入捕获器，tim8 作为合作式调度器，tim9,10,11 作为 pwm 发生器，均成功，捕获到了接收机在各个通道上的占空比输出

2015 年 7 月 2 日星期四：

- 调试好了 USB 转串口工具，成功打印字符串到计算机

- 监督张哲写好了字符串接收缓存算法，并不好用

2015 年 7 月 3 日星期五：

- 改进 USART 字符串接收缓存算法，使用多组测试数据测试，均已成功，发现问题所在——windows 系统发送的回车是 “\r\n” 不是 “\r” 或者 “\n”。
- PID 基类写好，组合到 Controller 类中

2015 年 7 月 4 日星期六：

- DEBUG Stick 类的占空比归一化方法，最终把油门占空比标准化到 0.05~0.10，把三个角度通道的占空比标准化、归一化到-1~+1
- 删除 Gyroscope，因为数学不好，课题紧急，没时间研究由重力分量获得 pitch 和 roll 角和用角速度积分得到 yaw 角的算法了，打算换模块
- 在刘洋的帮助下，调试好了串口蓝牙模块和 APP “蓝牙串口助手”，这样我电脑没有蓝牙，而手机有蓝牙也可以获得飞行器打印的数据了
- 组合 Stick 类为 Receiver 类，做好了与 Controller 类的数据接口，发现一个坑人的 BUG——系统应该先初始化/构造底层的定时器捕获器，再初始化/构造应用层/控制层类 Controller 否则 Controller 相应通道输入端始终得到 0（原始占空比值）

2015 年 7 月 5 日星期天：

- 在徐亚飞同学的帮助下调试好了新的模块，自带卡尔曼滤波，自带 yaw 角积分算法，得到了与飞行器本身参考系一致的三个欧拉角
- 实现动作控制逻辑（就是 pitch、roll、yaw 怎么输出给四个电机，正负关系），下午试飞飞行器（四个角绑在四个四个床头），崩了，但是逻辑应该没有出错（通宵后）
- Controller 类写入电机开启判断（只要 throttle 通道占空比小于阈值，关闭电机，不这么做有时候飞行器会失控，无法关闭）
- 对三个角度测量值，进行补偿，以系统初始化时测量到的第一组值为基准 0 点（因为寝室地面不是完全水平的），以第一个 yaw 角为基准 0 点（积分初值可能不为 0，保存了上次运行的结果）
- // PID 参数待调试

证据：本人 Github——[https://github.com/KellyHwong/STM32\\_Aircraft](https://github.com/KellyHwong/STM32_Aircraft)



21 次 commits !



连续 6 天的高强度编程！

注：调试蓝牙串口字符串命令的解释器/执行器又有新的 commits

## 10. 参考文献

作为组长，我个人查阅了大量的资料，包括但不限于以下：

- 【1】 Google 的 C++编程规范：<http://zh-google-styleguide.readthedocs.org/en/latest/>
- 【2】 StackOveflow-Combining C++ and C - how does #ifdef \_\_cplusplus work?:  
<http://stackoverflow.com/questions/3789340/combining-c-and-c-how-does-ifdef-cplusplus-work>
- 【3】 维 基 百 科 -PID 控 制 器 :  
<https://zh.wikipedia.org/wiki/PID%E6%8E%A7%E5%88%B6%E5%99%A8>
- 【4】 Microduino 四 轴 飞 行 器 教 程 :  
[https://www.microduino.cc/wiki/index.php?title=Microduino %E5%9B%9B%E8%B4%E9%A3%E8%A1%8C%E5%99%A8%E6%95%99%E7%A8%8B](https://www.microduino.cc/wiki/index.php?title=Microduino_%E5%9B%9B%E8%B4%E9%A3%E8%A1%8C%E5%99%A8%E6%95%99%E7%A8%8B)
- 【5】 我的四轴专用 PID 参数整定方法及原理 --- 超长文慎入：  
<http://blog.csdn.net/lynx2/article/details/8888063>
- 【6】 四旋翼飞行器 Quadrotor 飞控之 PID 调节（参考 APM 程序）：



[http://blog.csdn.net/super\\_mice/article/details/38436723](http://blog.csdn.net/super_mice/article/details/38436723)

- 【7】 MathWorks- Design and implement PID controllers :  
<http://cn.mathworks.com/discovery/pid-control.html>
- 【8】 新浪博客-增量式 PID 算法: [http://blog.sina.com.cn/s/blog\\_7c7e2d5a01011ta9.html](http://blog.sina.com.cn/s/blog_7c7e2d5a01011ta9.html)
- 【9】 Michael J. Pont 著, 周敏译, 《时间触发嵌入式系统设计模式》
- 【10】 李刚, 疯狂 JAVA 讲义, 第 2 版
- 【11】 任奕臻, 硬件课设课程报告