Automatic Sailboard Control System

自动帆板控制系统

开源修订版

著作人信息								
姓名	性别	出生年月	学院	专业	年级	教学号	邮箱	电话
金翼飞	未定义	未定义	未定义	未定义	未定义	未定义	791056842@qq.com	未定义
丰德浪	未定义	未定义				未定义	917011853@qq.com	未定义
赵　琦	未定义	未定义				未定义	1323130151@qq.com	未定义

团队名称:未命名队

目录

一、引言

1.1 系统简介

本系统涉及一种自动帆板控制系统。本系统通过控制风扇电机转速，来调节风速大小，改变帆板(图 1.1-01)所受风压，从而改变帆板与竖直方向夹角 θ 。借助系统实现的反馈调节功能，使 θ 按照程序设定稳定在一定范围内(图 1.1-2)。

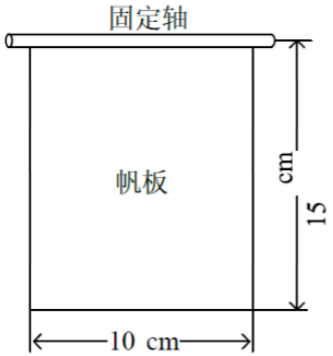

图 1.1-01

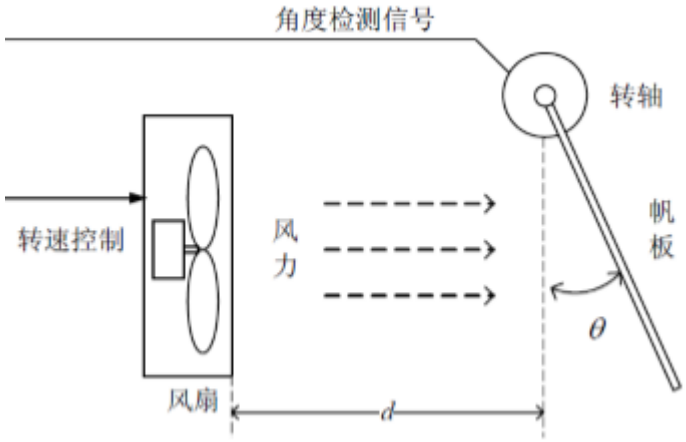

图 1.1-02

1.2 技术要求

1.调节系统未启动时人工转动帆板，能够实时显示帆板的转角θ。显示范围为0-90°，分辨率不低于5°，绝对误差≤10°；

2.当转轴与电机间距d=5cm时，通过上微机程序控制θ角大小，系统自动控制风扇转速并使帆板转角θ能够在0-45°范围内变化，并能实时显示θ大小；

3.控制调节过程在10秒内达到设定值并维持5秒以上，整个过程有提示；

4.帆板角度调节的步进值为5°，误差不超过10°；

5.转轴与风机水平间距d在5-15cm 范围内步进为1cm可调；

6.间距d在5-15cm 范围内任意选择，通过上位机设定帆板θ转角，范围为0-45°，在10秒内达到设定值并维持目标的θ值，维持5秒以上，过程中实时显示θ角大小。

1.3 发展概况

申报小组检索国内外众多文献，除作为2011年全国大学生电子设计大赛F题外，尚未检索到“帆板控制系统”的应用实例及发展历史。

二、方案设计

2.1 方案选择

经过对运行效率、响应速度、开发成本(含人力资源成本、经济成本和时间成本)及界面友好性等多方面考虑，我小组迅速得出一种可行方案，该方案如下(图 2.1-01)：

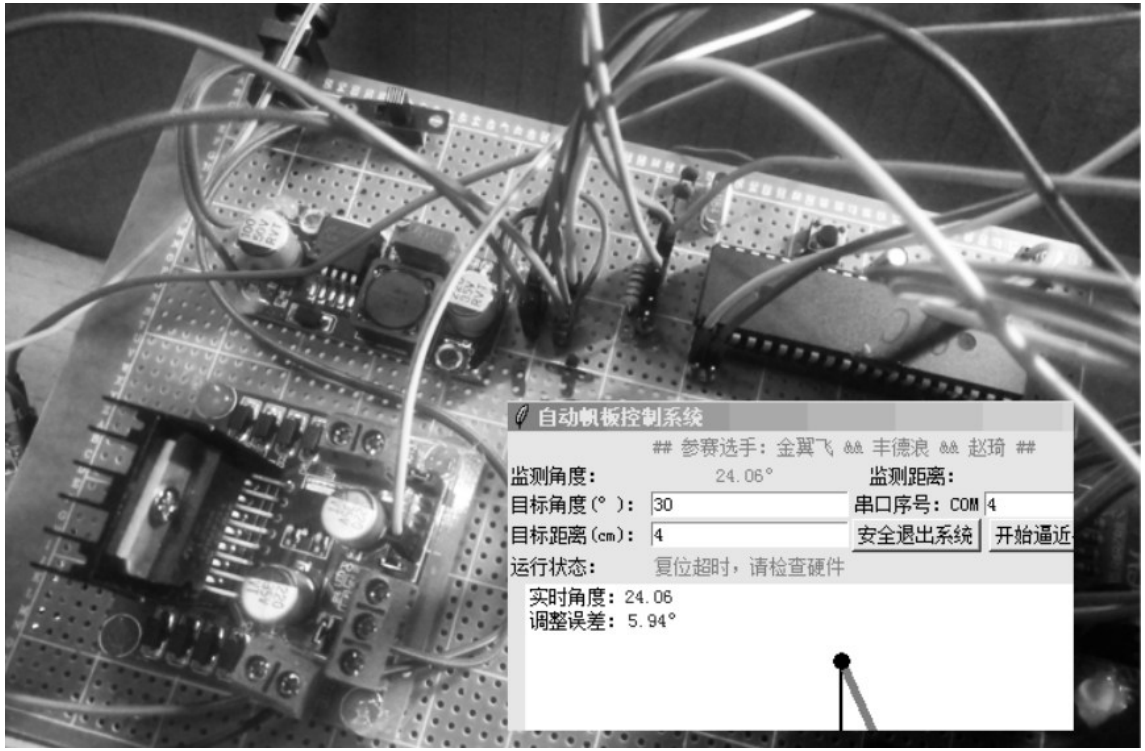

图 2.1-01

1.由含 MPU6050 六轴运动处理器的 GY-521 传感器采集帆板各个方向的加速度数据(舍弃角加速度数据)；

2.传感器将采集到的数据同过 IIC 总线传输到 Arduino Nano 进行格式转换；

3.Arduino Nano 将收集到的加速度数据按照一定的格式通过 UART 拓展总线传输到上位机的 PL2303(或 CH341)串口转 USB 模块；

4.上位机程序启动后初始化变量并通过 Python3.6 提供的 Tkinter 图形库绘制图形界面；

5.读取用户输入的参数，启动校验进程；

6.检查用户输入格式并执行系统自检，如发现错误由校验进程重新绘制图形界面的状态显示标记以输出错误信息；

7.核对串行端口号格式并检查对应端口是否处于闲置状态，如发现错误由校验进程重新绘制图形界面的状态显示标记以输出错误信息；

8.检查用户输入数据的有效性，如发现错误由校验进程重新绘制图形界面的状态显示标记以输出错误信息；

9.当确认用户输入信息正确后在图形类中打开串行端口(图形类使用单独线程独占串行端口，线程对象由 Python3.6 提供的 threading 类派生并使用 start 方法执行)；

10.开始对照风机轨道位置，并在用户接口上绘制风机位置信息，向串行数据解释微控制器(为优化中断运行效率，防止中断冲突和指令遗漏，本设计使用 2 片 STC89C52RC 微控制器)发送风机轨道位移指令(控制步进电机)，每条指令间有一指定时间间隔(在消除指令竞争前提下的最优时间间隔)；

11.等待风机轨道位置调整完成，打开串行数据接收线程，实时计算帆板的角度(基于重力加速度分量，如式)；

当 θ ≤120° 时：

$$\theta = 90° + \frac{180° \times \tan^{-1}\frac{a_y}{a_x}}{\pi} + 11.5$$ ，其中 11.5 为本系统的误差修正参数。

当 θ ≥120° 时：

$$\theta = \frac{180° \times \tan^{-1}\frac{a_y}{a_x}}{\pi} - 90° + 11.5$$ ，其中 11.5 为本系统的误差修正参数。

12.调用绘图子函数并自增计数器变量，然后在用户接口绘制实时角度数据；
13.当绘图计数器每达到一定范围时清空，并绘制帆板示意图和监测角度数据(使数据更有利于用户观察)；
14.将实时角度与用户输入数据进行比较(实时读取输入)，通过串口下传风机转速调整指令；
15.调整指令送入 PWM 控制微控制器(减少串行中断对 PWM 的影响，保持风机转速相对稳定)，并调整 PWM 占空比；
16.重复执行以上 11-15 步。

2.2 方案优势

1.一定程度上确保了数据处理和动作实时性；
2.避免传输数据发生竞争，
3.用户接口友好；
4.具有实时模拟功能，方便监控人员远程直观观察帆板状态；
5.设计简单，在效率和成本之间做出了较好的权衡。

2.3 程序流程图

本系统的整体程序设计如图 2.3-01(初始化)、图 2.3-02(数据实时处理)、图 2.3-03(调速)。

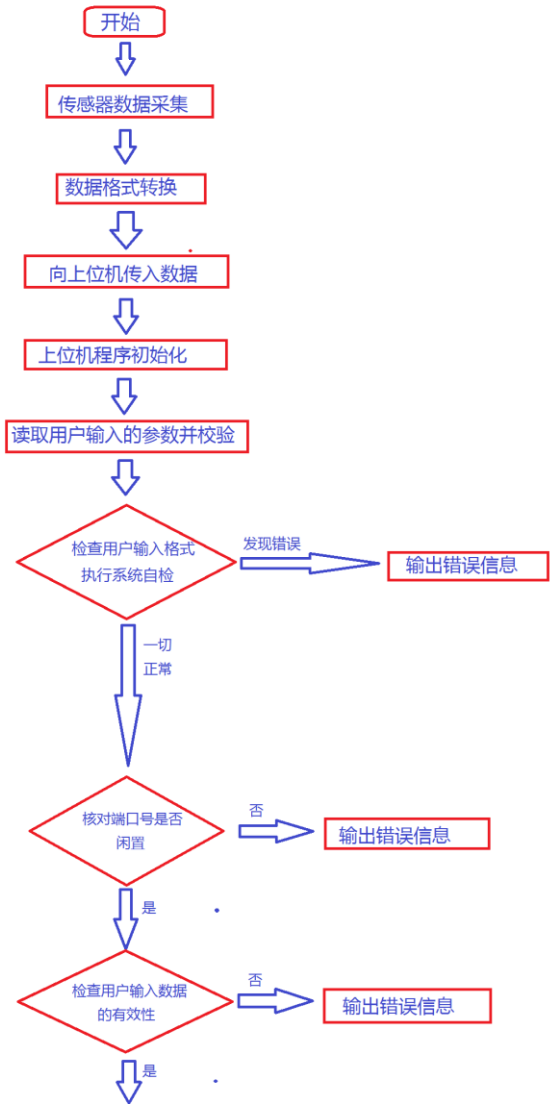

图 2.3-01

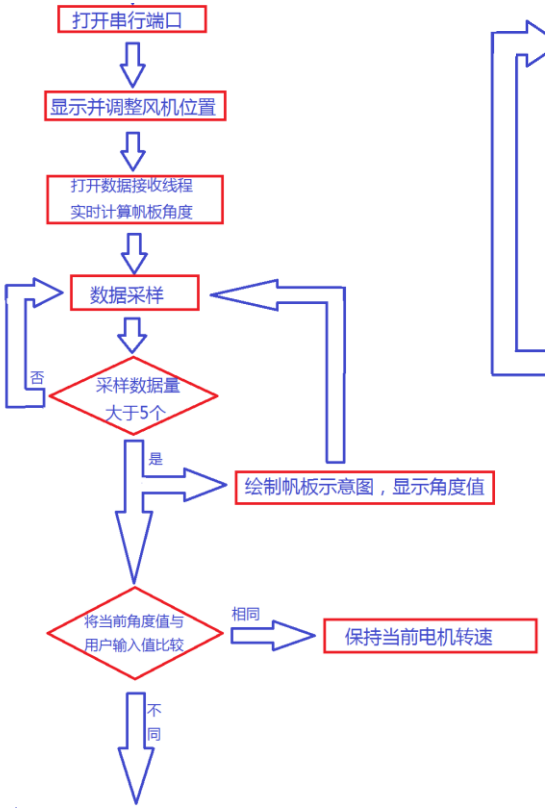

图 2.3-01

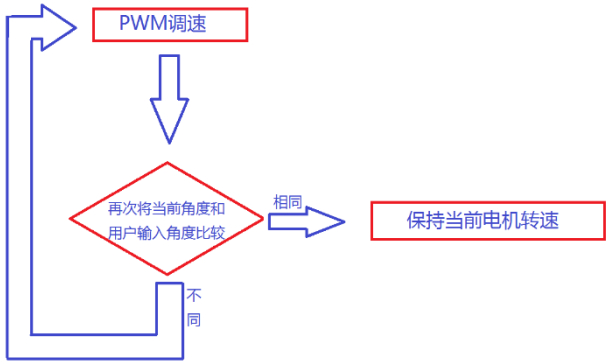

图 2.3-03

2.3 电路图

如图 2.3-01。

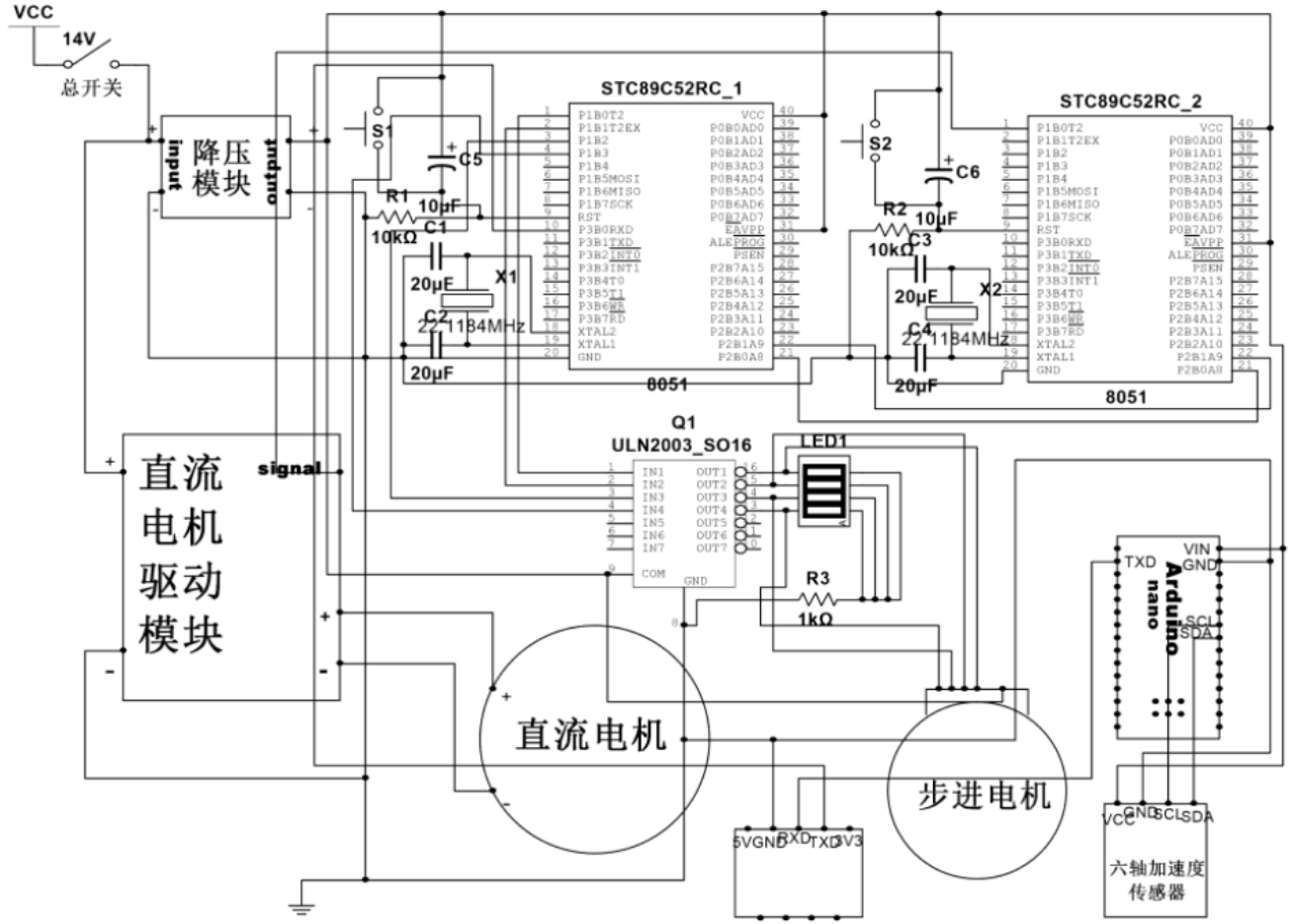

图 2.3-01

三、系统内部通信协议

3.1 协议设计

本系统上位机与下位机采用 UART 通信，传感器与数据采样转换器采用 IIC 协议通信，这两种协议均为公共协议，这里不在赘述。

本系统的串行指令执行子系统与脉冲宽度调制子系统、高速数据处理子系统与串行指令执行子系统间采用如下协议：

(1).串行指令执行系统向脉冲宽度调制子系统以带宽为 2 宽度为 10ms 脉冲方式发送 PWM 占空比变化方向控制脉冲；

(2).高速数据处理子系统向串行指令执行子系统以 19200 波特率的 UART 串行方式发送 8bit 控制信号，并放置一个停止位，其中‘A’对应的 ASCII 码表示风机 PWM 占空比增加，‘S’对应的 ASCII 码表示风机 PWM 占空比减少，‘F’对应的 ASCII 码控制轨道电机向前运动，‘B’对应的 ASCII 码控制轨道电机向后运动。此部分设计一个监测部分，‘R’对应的 ASCII 码用作测试信号，当此部分 TX 与 RX 端口同时接入上位机时，下发此指令后上位机收到‘O’表示指令解析系统正常。

四、串行指令解析与执行子系统

4.1 子系统概述

本子系统由一片 STC89C52RC 微控制器及一片 ULN2003 逻辑门电路构成，用于下位机对上位机下发的指令进行解析，并完成轨道控制电机的初始化和复位工作。

系统使用内部时钟源，时钟源频率为 22.1184MHz(测试误差 0.04%)，串行端口波特率为 19200。这部分内建串行接收处理中断和指令解析程序。另外，步进电机位置初始化由此部分完成，ULN2003 用于驱动步进电机。

此子系统设计初计划加入 PWM 功能来调节风机转速以减少微控制器数量，但考虑到中断优先级、PWM 连续性和系统反应时间等问题，最终决定分离 PWM 和串行指令解析系统。

此子系统实物如图 4.1-01。

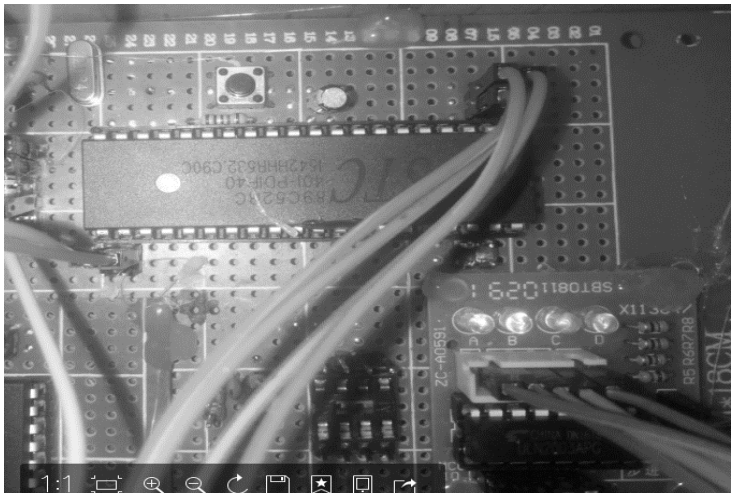

图 4.1-01

4.2 串行中断实现

此子系统程序为串行中断部分，用于实现串行数据的收发(发送部分用于调试)。

实现该功能的 C 语言函数如下(部分代码)：

```
//设置波特率、时钟源、串行状态忙标记、指令标志、循环变量和指令接收缓冲区(此部分程序源于本小组以往的开源代码)
#define FOSC 22118400L   //晶振频率
#define BAUD 19200   //波特率
#define P1_init ~0xFF   //步进电机驱动初始值，防止组绕过热
bit busy; //状态忙标志
int flag=0;   //指令待执行标志
int iloop=0;  //循环读取标志
char ram[10]=0;   //指令缓冲区
char command=0;   //指令读取字节

//初始化串行端口工作状态，设置中断类型等
SCON=0xda;    //串口 11 位异步收发，打开串口接收允许位，初始化标志位和校验位
TMOD=0x20;    //定时器设置为 8 位自动重装计数初值计数器
ISP_CMD=0x00;
ISP_CONTR|=0x81;
ISP_TRIG=0x46;
ISP_TRIG=0xb9;    //设置 STC 内部寄存器
TH1=TL1=-(FOSC/12/32/BAUD);     //设置计时器初值
TR1=1;      //启动定时器 1
ES=1;       //打开串行中断允许位
EA=1;       //开总中断允许位

//接收终中断服务函数，用于接收 1 字节串行数据
void Uart_Isr() interrupt4 using 1
{
    if(RI)      //当串行接收中断标志置位时取出缓冲区的字节
    {
        RI=0;       //中断标志软件置零
        command=SBUF;       //取出缓冲区中的数据
        flag=1;         //指令执行标记置位
    }
    if(TI)      //串行发送中断置位时的处理部分
    {
        TI=0;       //软件置零中断标志
        busy=0;     //清除串口忙标志
    }
}

//单字节发送函数，用于调试
void SendData(unsigned char dat)
{
    while(busy)    //当串口忙时
        ACC=dat;        //寄存等待发送的字节
    if(P)  //奇偶校验检查
```

```
{
    TB8=1;      //发送奇偶校验位
}
else
{
    TB8=0;      //发送奇偶校验位
}
busy=1;    //置忙串行状态标志
SBUF=ACC;      //将 ACC 中缓存的数据发出
}

//字符串发送函数，用于调试
void SendString(char* s)
{
    while(*s)
    {
        SendData(*s++);   //通过指针遍历逐位发送字符串
    }
```

4.3 导轨控制

导轨控制函数由两个子函数和一个全局数组组成，用来实现导轨电机的精确控制。

其实现的 C 语言函数如下：

```
//定义全局变量和控制引脚
int
step_motor[8]={0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09};
   //步进电机控制数组
sbit ad=P2^1; //定义 P2.1 端口为 PWM 占空比增加信号入口
sbit su=P2^0; //定义 P2.0 端口为 PWM 占空比减少信号入口

//导轨前进函数
void forward()
{
    int i,j;
    for(j=0;j<10;j++)     //循环十次，确保运行速率
        for(i=0;i<8;i++)      //遍历步进电机控制数组
        {
            P1=~step_motor[i];   //输出步进电机控制信号数
        组
            delay();       //等待电机动作完成
        }
    P1=P1_init;   //步进电机控制电平恢复初值，防止组绕过
热
}

//导轨后退函数，基本同前进函数
void back()
{
```

```
inti,j;
for(j=0;j<10;j++)
    for(i=7;i>=0;i--)     //反向遍历步进电机控制信号数组
    {
        P1=~step_motor[i];
        delay();
    }
    P1=P1_init;
}
```

4.4 指令解析

此子系统的串行译码模块用于解析上位机传入的指令，并具体执行或下发执行。

其实现的 C 语言函数如下：

```
//指令解析处理函数
void terminal()
{
    char temp[2]=0;   //测试变量缓冲区，用于改进的内存管理机制，此程序中弃用
    if(flag)   //当指令待执行标记置位时执行
    {
        if(command=='R')      //测试指令 R
        {
            SendString("0");      //回显 0 表示连接正常
        }
        elseif(command=='B')      //步进电机返回指令 B
        {
            EA=0;       //关总中断允许位，防止此步骤被打断
            forward();     //步进电机前进一次(此处有逻辑错误，但不影响执行)
            EA=1;       //开总中断允许位
        }
    elseif(command=='F')      //步进电机前进指令 F
    {
        EA=0;       //关总中断允许位，防止此步骤被打断
        back();     //步进电机回退一次(此处有逻辑错误，但不影响执行)
        EA=1;       //开总中断允许位
    }
    elseif(command=='A')      //PWM 占空比增加指令
    {
        SendString("add\n");      //上位机测试反馈
        EA=0;       //关总中断允许位，防止此步骤被打断
        ad=0;       //增加指令下发
        wait();     //延时，防止 PWM 部分无法接收到此控制信号
        ad=1;       //停止下发
```

```
    EA=1;       //开总中断允许位
}
elseif(command=='S')      //PWM 占空比减少指令
{
    SendString("sub\n");      //上位机测试反馈
    EA=0;       //关总中断允许位，防止此步骤被打断
    su=0;       //减少指令下发
    wait();     //延时，防止 PWM 部分无法接收到此控制信
号
    su=1;       //停止下发
    EA=1;       //开总中断允许位
}
else        //容错选项
    {
        ;       //测试用，此处留空指令
    }
    flag=0;         //指令执行结束，执行标记置零
}
}
```

五、串行加速度数据采样子系统

5.1 子系统概述

此子系统的用于完成帆板各个方向加速度值的采样与编码转换工作，将由 GY-521 加速度和陀螺仪模块读取的数值由 IIC 协议编码转换为 UART 协议的自定编码，送由上位机处理。

自定编码为三段数据，分别表示 x、y、z 轴向的加速度分量(用传感器分辨率表示)，数据间用垂直制表符分隔，每组数据后由换行符结束。

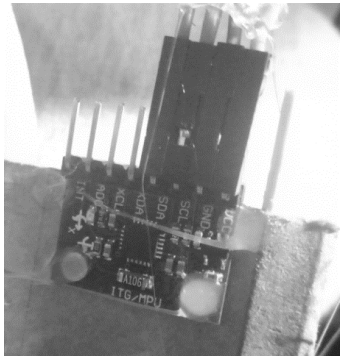

图 5.1-01

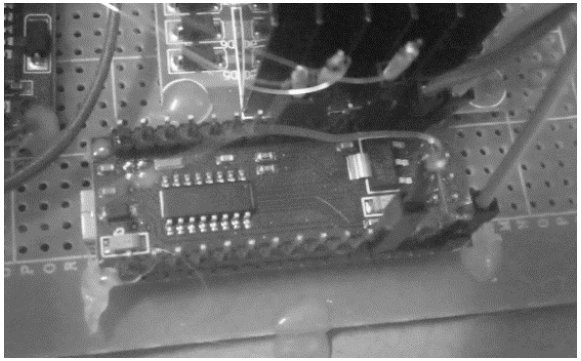

图 5.1-02

此部分由 GY-521 传感器(含 MPU6050 六轴运动数据处理器，图 5.1-01)和一片 Arduino Nano(图 5.1-02)组成。Arduino Nano 可以减少软件编程开销，复用已有的库函数能保证系统运行稳定性和效率。传感器与 Arduino

之间通过漆包线连接，以减少导线对帆板运动的阻碍。

5.2 IIC 协议简介

I2C 串行总线一般有两根信号线，一根是双向的数据线 SDA，另一根是时钟线 SCL。所有接到 I2C 总线设备上的串行数据 SDA 都接到总线的 SDA 上，各设备的时钟线 SCL 接到总线的 SCL 上。

为了避免总线信号的混乱，要求各设备连接到总线的输出端时必须是漏极开路（OD）输出或集电极开路（OC）输出。设备上的串行数据线 SDA 接口电路应该是双向的，输出电路用于向总线上发送数据，输入电路用于接收总线上的数据。而串行时钟线也应是双向的，作为控制总线数据传送的主机，一方面要通过 SCL 输出电路发送时钟信号，另一方面还要检测总线上的 SCL 电平，以决定什么时候发送下一个时钟脉冲电平；作为接受主机命令的从机，要按总线上的 SCL 信号发出或接收 SDA 上的信号，也可以向 SCL 线发出低电平信号以延长总线时钟信号周期。总线空闲时，因各设备都是开漏输出，上拉电阻 Rp 使 SDA 和 SCL 线都保持高电平。任一设备输出的低电平都将使相应的总线信号线变低，也就是说：各设备的 SDA 是“与”关系，SCL 也是“与”关系。

总线的运行（数据传输）由主机控制。所谓主机是指启动数据的传送（发出启动信号）、发出时钟信号以及传送结束时发出停止信号的设备，通常主机都是微处理器。被主机寻访的设备称为从机。为了进行通讯，每个接到 I2C 总线的设备都有一个唯一的地址，以便于主机寻访。主机和从机的数据传送，可以由主机发送数据到从机，也

可以由从机发到主机。凡是发送数据到总线的设备称为发送器，从总线上接收数据的设备被称为接受器。

I2C 总线上允许连接多个微处理器以及各种外围设备，如存储器、LED 及 LCD 驱动器、A/D 及 D/A 转换器等。为了保证数据可靠地传送，任一时刻总线只能由某一台主机控制，各微处理器应该在总线空闲时发送启动数据，为了妥善解决多台微处理器同时发送启动数据的传送（总线控制权）冲突，以及决定由哪一台微处理器控制总线的问题，I2C 总线允许连接不同传送速率的设备。多台设备之间时钟信号的同步过程称为同步化。

在 I2C 总线传输过程中，将两种特定的情况定义为开始和停止条件：当 SCL 保持“高”时，SDA 由“高”变为“低”为开始条件；当 SCL 保持“高”且 SDA 由“低”变为“高”时为停止条件。开始和停止条件均由主控制器产生。使用硬件接口可以很容易地检测到开始和停止条件，没有这种接口的微机必须以每时钟周期至少两次对 SDA 取样，以检测这种变化。

SDA 线上的数据在时钟“高”期间必须是稳定的，只有当 SCL 线上的时钟信号为低时，数据线上的“高”或“低”状态才可以改变。输出到 SDA 线上的每个字节必须是 8 位，每次传输的字节不受限制，但每个字节必须要有一个应答 ACK。如果一接收器件在完成其他功能（如一内部中断）前不能接收另一数据的完整字节时，它可以保持时钟线 SCL 为低，以促使发送器进入等待状态；当接收器准备好接受数据的其它字节并释放时钟 SCL 后，数据传输继续进行。

数据传送具有应答是必须的。与应答对应的时钟脉冲由主控制器产生，发送器在应答期间必须下拉 SDA 线。

当寻址的被控器件不能应答时，数据保持为高并使主控器产生停止条件而终止传输。在传输的过程中，在用到主控接收器的情况下，主控接收器必须发出一数据结束信号给被控发送器，从而使被控发送器释放数据线，以允许主控器产生停止条件。

I2C 总线在开始条件后的首字节决定哪个被控器将被主控器选择，例外的是“通用访问”地址，它可以在所有期间寻址。当主控器输出一地址时，系统中的每一器件都将开始条件后的前 7 位地址和自己的地址进行比较。如果相同，该器件即认为自己被主控器寻址，而作为被控接收器或被控发送器则取决于 R/W 位。

5.3 MPU6050 六轴运动处理器

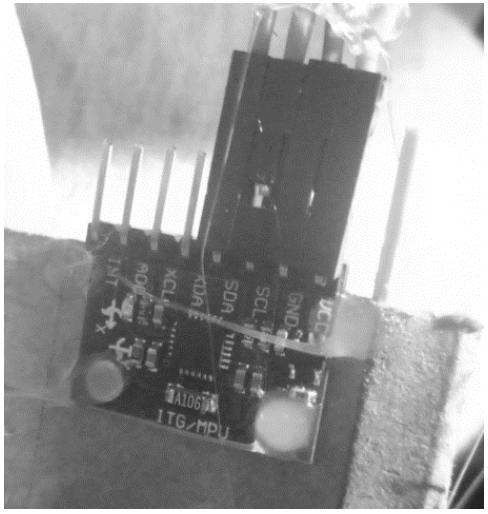

图 5.3-02

MPU-6050 的角速度传感器(图 5.3-01)感测范围为±250、±500、±1000 与±2000°/sec(dps)，可准确追踪快速与慢速动作，并且，用户可程式控制的加速器全格感测范围为±2g、±4g±8g 与±16g。产品传输可透过最高至 400kHz 的 IIC 或最高达 20MHz 的 SPI。MPU-6050 可在不同电压下工作，VDD 供电电压介为 2.5V±5%、3.0V±5%或 3.3V±5%，逻辑接口 VDDIO 供电为 1.8V±5%。MPU-6050 的包装尺寸 4x4x0.9mm(QFN)。其他的特征包含内建的温度感测器，包含在运作环境中仅有±1%变动的振荡器。

以数字输出 6 轴或 9 轴的旋转矩阵、四元数(Quaternion)、欧拉角格式(Euler Angle Format)的融合演算数据。具有 131LSBs/°/sec 敏感度与全格感测范围为±250、±500、±1000 与±2000°/sec 的 3 轴角速度感测器(陀螺仪)。可程式控制，且程式控制范围为±2g、±4g、±8g 和±16g 的 3 轴加速器。移除加速器与陀螺仪轴间敏感度，降低设定给予的影响与感测器的飘移。数字运动处理(DMP: Digital Motion Processing)引擎可减少复杂的融合演算数据、感测器同步化、姿势感应等的负荷。运动处理数据库支持 Android、Linux 与 Windows 内建之运作时间偏差与磁力感测器校正演算技术，免除了另外进行校正的需求。

5.4 IIC 转串口实现

IIC 转串口部分采用 Arduino Nano 实现，其时钟频率为 16MHz。

实现如下：

```
//加载头文件及初始化变量
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"      //引用 I2C 协议头文件和 MPU6050 支持头文件
MPU6050 accelgyro;         //创建一个 MPU6050 对象
int16_t ax,ay,az;      //声明 16 位加速度变量

//初始化函数
void setup()
{
    Wire.begin();      //启动 IIC 总线
    Serial.begin(19200);      //打开串行端口并设置波特率
    accelgyro.initialize();      //加速度计初始化
}

//读取及转换函数
voidloop()
{
```

```
    accelgyro.getAcceleration(&ax,&ay,&az);   //读取加速度计数据，存入 ax，ay，az 的指针指向的位置
    Serial.print(ax);
    Serial.print("\t");
    Serial.print(ay);Serial.print("\t");
    Serial.println(az);      //依照指定的格式上传加速度数据
    delay(50);     //延时以等待传输完成
}
```

六、脉冲宽度调制子系统

6.1 子系统简介

此子系统用于实现对风机的(几乎)无间断 PWM 信号输出，并且实现占空比连续可调。

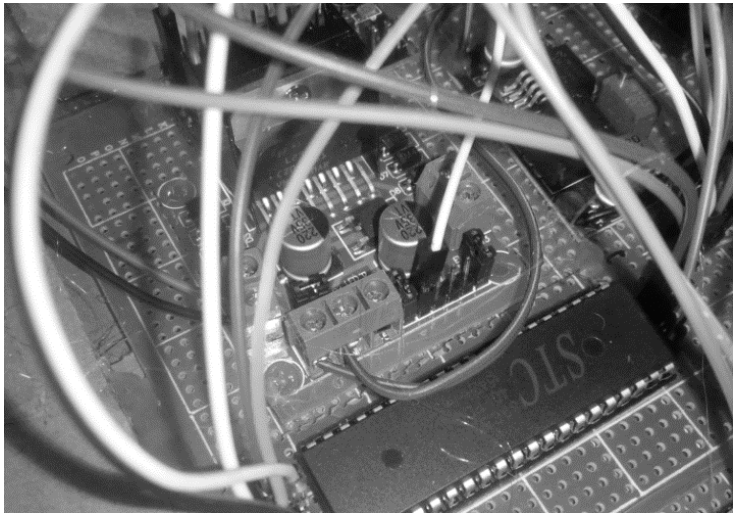

图 6.1-01

此部分由一片 STC89C52RC 微控制器、一块直流电机驱动板和一个 12V 风机组成。为了达到较为理想的效果(0-60° 连续可调)，我们采用过压 16.7%的 14V 输入。系统如图 6.1-01。

6.2 脉冲宽度调制子系统

此子系统提供开机 1s 满功率电机自检，并在控制信号到来前以 50%占空比运行。可在控制信号到来后实时调整占空比，完成闭环控制。

此部分的程序 C 语言实现如下：

```
//设置头文件及宏定义，同时设置信号输入引脚和 pwm 全局变量
#include <reg51.h>        //8051 寄存器定义头文件
#include <intrins.h>      //用于提供基于汇编的 nop 空指令
#define step 1    //定义 PWM 调节步长
#define all 333       //最大(100%)PWM 占空比对应的数值
int pwm;       //PWM 全局变量
sbit out=P1^0;     //电机 PWM 信号输出引脚
sbit ad=P2^1;      //占空比增加信号，低电平有效
sbit su=P2^0;      //占空比减少信号，低电平有效

//声明延时函数参数及变量
void delay();

//此部分主函数
int main()
{
    int i;      //声明循环变量空间
    pwm=all/2;      //设置 pwm 为满功率占空比的一半
    P2=0xFF;   //置位输入引脚为高
    P1=0x00;   //电机控制引脚初始化
    out=1;      //满功率输出测试自检
    delay();        //延时 1s
    out=0;      //自检结束
    while(1)        //进入循环控制阶段
    {
        if(~ad&&(pwm<all-step))       //当 ad 有低电平输入并且 pwm 值小于最大可增加值时执行
        pwm+=step;      //以步长增加占空比
        out=1;      //pwm 高电平阶段
        for(i=0;i<pwm;i++)    //高电平周期控制
            ;
        if(~su&&(pwm>=step))       //当 su 有低电平输入时并且 pwm 值大于最小可减数值时执行(穿插设计有利于减少控制信号等待时间，提高系统执行效率)
            pwm-=step;      //以步长减小占空比
        out=0;      //pwm 低电平阶段
        for(i=0;i<(all-pwm);i++)      //低电平周期控制
            ;
    }
}
```

```
//自检延时程序
void delay()  //延时 1s，由 STC-ISP 基于 22.1184MHz 时钟源生成
{
    unsignedchari,j,k;
    _nop_();
    i=15;
    j=2;
    k=235;
    do
    {
        do
        {
            while(--k);
        }while(--j);
    }while(--i);
}
```

七、高速数据处理子系统

7.1 子系统简介

此子系统提供了一个图形化用户接口，如图 7.1-01。

图 7.1-01

此部分收集传感器数据，用于反馈调整，并根据用户输入的参数自动逼近。图形化用户接口提供一个可视化

容器，用于模拟帆板状态。用户接口后台提供此系统所有的自动控制功能。

7.2 用户交互界面设计

此部分使用 Python3.6 的 Tkinter 模块提供的方法绘制图形化用户接口，设置全局变量等，实现该部分的 python 程序源码如下：

```
def window(self):     #定义窗体方法
    win=Tk()        #产生窗体对象
    self.exitflag=0   #初始化线程阻塞标志(threading 模块没有提供线程杀死方法)
    self.ang=0      #初始化角度变量
    self.flinesub=0        #初始化误差值
    self.lo=0       #初始化循环
    self.adjust=0       #初始化调整变量
    self.disbit=0       #初始化步进电机计数器
    self.readbuf="a".encode("ascii")         #初始化串行缓冲区，并从 ascii 转换为数据流格式
    self.nowangle=StringVar()         #初始化当前角度字符串变量
    self.nowdistance=StringVar()         #初始化当前距离字符串变量
    self.aimangle=StringVar()         #初始化目标角度字符串变量
    self.aimdistance=StringVar()         #初始化目标距离字符串变量
    self.sernumber=StringVar()        #初始化串行端口序号字符串变量
    self.gra=Canvas(win,width=400,height=200,bg="white")
    #初始化画布图形容器，并设置大小和底色
    self.tip=StringVar()      #初始化标记字符串
    self.ser=""         #初始化串口
    self.info=" "       #初始化提示信息

    win.title("自动帆板控制系统")        #设置窗体标题
    fra=Frame(win)          #从窗体产生帧
    text=Label(win,text="##参赛选手：金翼飞&&丰德浪&&赵琦##",fg="blue")       #产生标签
    lab1=Label(fra,text="监测角度：")       #检测角度标签
    nowang=Label(fra,textvariable=self.nowangle,fg="red")       #当前角度标签位置
    lab2=Label(fra,text="监测距离：")       #监测距离标签
    nowdis=Label(fra,textvariable=self.nowdistance,fg="green")      #当前距离标签位置
    lab3=Label(fra,text="目标角度(°)：")       #目标角度标签
```

```
    aimang=Entry(fra,textvariable=self.aimangle)      #目
标角度输入框
    lab4=Label(fra,text="目标距离(cm)：")      #目标距离
标签
    aimdis=Entry(fra,textvariable=self.aimdistance)
    #目标距离输入框
    lab5=Label(fra,text="串口序号：COM")      #串口序号
标签
    sernum=Entry(fra,textvariable=self.sernumber)     #串
口序号输入框
    exitbutton=Button(fra,text="安全退出系统
",command=self.safeexit)     #安全退出按钮
    runbutton=Button(fra,text="开始逼近并监测目标
",command=self.run)      #逼近按钮
    lab6=Label(fra,text="运行状态：")      #运行状态标签
    labtip=Label(fra,textvariable=self.tip,fg="red")
    #状态提示标签位置
    self.gra.create_line(200,50,200,175,width=2,tags="tl
ine")      #创建基准线
    self.gra.create_line(200,50,200,175,width=5,fill="re
d",tags="fline")      #创建动态线用于表示帆板
    self.gra.create_oval(195,45,205,55,tags="orig",fill=
"black")      #创建实心圆表示转轴
    self.gra.create_text(35,10,text="实时角度：
",tags="arganglab")      #在容器中创建实时角度字段
    self.gra.create_text(35,25,text="加速误差：
",tags="adjsub")      #在容器中创建加速误差字段
    self.gra.create_text(35,40,text="实时距离：
",tags="dislab")      #在容器中创建实时距离字段
    self.gra.create_text(35,55,text="距离误差：
",tags="dissub")      #在容器中创建距离误差字段

    text.pack()      #放置文本
    lab1.grid(row=1,column=1,sticky=W)     #表格格式放置
元素
    nowang.grid(row=1,column=2)     #表格格式放置元素
    lab2.grid(row=1,column=3)      #表格格式放置元素
    nowdis.grid(row=1,column=4)     #表格格式放置元素
    lab3.grid(row=2,column=1,sticky=W)     #表格格式放置
元素
    aimang.grid(row=2,column=2)     #表格格式放置元素
    lab4.grid(row=3,column=1,sticky=W)     #表格格式放置
元素
    aimdis.grid(row=3,column=2)     #表格格式放置元素
    lab5.grid(row=2,column=3)      #表格格式放置元素
    sernum.grid(row=2,column=4)     #表格格式放置元素
    exitbutton.grid(row=3,column=3)     #表格格式放置元素
    runbutton.grid(row=3,column=4)     #表格格式放置元素
    lab6.grid(row=4,column=1,sticky=W)     #表格格式放置
元素
    labtip.grid(row=4,column=2)     #表格格式放置元素
    fra.pack()     #放置帧
    self.gra.pack()      #放置图形容器

    win.mainloop()      #启动窗体线程
```

7.3 多线程设计

此部分完成程序部分的多线程布置(便于多个任务同时执行，保证系统的实时响应。程序中的延时参数是多次实验测定得到的稳定值)，及输入错误检测，其实现 Python 源码如下：

```
#主程序调用
go=autogo()        #实例化类
global exitflag        #退出标志生命为全局变量
exitflay=0     #置零全局退出标志
t1=threading.Thread(target=go.window)       #使用
threading 类提供的 Thread 方法创建主窗口的线程类
t1.start()     #启动主线程

#线程布置及输入检测函数
def run(self):
    try:
        self.ser.close()       #尝试关闭曾经打开的串口实例
    except:
        check=1         #尝试失败时做无用功
    check=1         #检查通过标记默认为 1
    self.adjust=0       #调整量初始化
    if(not self.sernumber.get().isdigit()):         #串行口
输入格式错误
        check=0         #检查通过标志置零
        self.sernumber.set("串行端口错误!")         #输出错
    误类型
        self.tip.set("你需要检查你的输入")      #输出错误
提示
    if(not self.aimangle.get().isdigit()):      #目标角
度输入格式错误
        check=0         #检查通过标志置零
        self.aimangle.set("角度错误!")       #输出错误
    类型
        self.tip.set("你需要检查你的输入")      #输出错
    误信息
    if(not self.aimdistance.get().isdigit()):       #目
标距离输入格式错误
        check=0         #检查通过标志置零
        self.aimdistance.set("距离错误!")       #输出错
    误类型
        self.tip.set("你需要检查你的输入")      #输出错
    误信息
    if(check):      #如果检查通过
        try:
            self.ser=serial.Serial('com'+self.sernumbe
        r.get(),19200)          #尝试开启对应串口
            t2=threading.Thread(target=self.rec)
            #尝试创建串口读取线程
            t2.start()      #尝试启动串口读取线程
```

```
        except WindowsError:      #如果尝试失败
            check=0          #检查通过标志置零
            self.sernumber.set("串行端口错误!")
            #输出错误类型
            self.tip.set("串口被占用或不存在")      #输
        出错误信息
    if(check):     #如果检查依然通过
        if(not eval(self.aimdistance.get())):      #如
    果目标距离输入为0
            aim=0       #设置目标距离为0
        else:       #否则
            aim=eval(self.aimdistance.get())       #设
        置目标距离为输入字符串对应的数值
            iloop=int(aim*34)-int(self.disbit)      #计
        算步进电机初始化所需循环次数
            if(iloop>0):       #当循环变量为正时
            for i in range(1,iloop+1):       #执行循环
                self.ser.write("F".encode("ascii"))
                #发送F指令(指令由ascii码转换为数据
            流)
                self.disbit+=1          #步进电机位置计数
                器加一
                time.sleep(0.15)      #等待步进电机动作
        if(iloop<0):       #当循环变量为负
            for i in range(1,abs(iloop+1)):     #循环执
        行
                self.ser.write("B".encode("ascii"))
                #发送B指令(指令由ascii码转换为数据
            流)
                self.disbit-=1          #步进电机位置计数
            器减1
                time.sleep(0.15)      #等待步进电机动作
        try:
            t3=threading.Thread(target=self.adj)
            #尝试调用threading类中的Thread方法创建调
        整函数的线程类
            t3.start()          #尝试启动调整线程
        except:
            self.tip.set("监控调整进程启动失败")
            #如果尝试失败则显示错误信息
```

7.4 串行接收线程

此部分用于串行信号接收，确保连续读取传采样系统上传的数据，便于及时处理并使系统做出反应。此程序由两个封装在同一个类中的方法组成，由于 python3 的 threading 类没有提供杀死进程的方法，标志位设计可以

使线程能够安全退出。其实现的 python 源码如下：

```
#接收循环线程
def rec(self):
    while(not self.exitflag):        #检查退出标志
        if(self.exitflag):        #再次确认退出标志
            break
        self.receive()        #执行一次接收处理函数

#接收处理函数
def receive(self):
    self.readbuf=self.ser.read()        #读取串口数据
    if(self.readbuf is b'\n'):        #监测行终止标志，由本
小组定义
        self.lo+=1    #监测标志自增
        info=self.info.split('\t')        #以垂直制表符分割
数据
        try:
            x=eval(info[0])
            y=eval(info[1])
            z=eval(info[2])        #尝试将三个加速度值转换
为数值型
        except:
            no=1        #如果失败则继续使用上一次的值
        self.adjust=1      #调整开关置位
        self.display(-y,z)        #送显当前数据
        self.info=""        #清空 info 缓冲区
    else:
        self.info+=self.readbuf.decode("ascii")      #如果未
检测到行结束，则向 info 缓冲区中插入新接收的数据转换来
的 ascii 码字符
```

7.5 脉冲宽度调制(PWM)线程

此部分用于实时根据传感器提供的数据调整 PWM 的占空比，其实现的 python 源码如下：

```
#占空比实时调整线程
def adj(self):
    while(notself.exitflag):    #检查退出标志
        if(self.exitflag):        #再次检查退出标志
            break
        if(self.adjust):        #检查调整允许标志
            ang=round(eval(self.aimangle.get())-
        self.ang,0)        #计算调整角
```

```
if(ang>0):      #如果调整角为正
    self.ser.write("A".encode("ascii"))
    #下发 A 指令(指令由 ascii 码转换为数据流)
if(ang<0):      #如果调整角为负
    self.ser.write("S".encode("ascii"))
    #下发 S 指令(指令由 ascii 码转换为数据流)
time.sleep(0.03)      #等待下位机完成动作
```

7.6 数据可视化模块

此部分用于显示 5 步间隔采样的加速度计算得出的角度数值并绘制示意图，同时计算输出一些必要信息。其实现的 python 源码如下：

```
#数据可视化函数
def display(self,y,z):
    at=math.atan(y/z)      #计算当前弧度
    ang=90+at*180/3.1415926+11.5          #计算当前角度
    if(ang>=120):      #如果角度大于等于 120°
        ang=-(180-ang)          #根据需要调整角度
        at=((ang-90)*3.1415926/180)      #根据需要调整弧度
    self.gra.delete("argang","orig","dis")      #删除上一次产生的图形
    self.flinesub+=ang          #计算偏角度积累
    self.gra.create_text(80,40,text=str(round(self.disbit/34,2)),tags="dis")      #创建距离显示数值
    self.gra.create_text(80,10,text=str(round(ang,2)),tags="argang",fill="green")    #创建实时角度实时数值
    self.gra.create_oval(195,45,205,55,tags="orig",fill="black")        #重新绘制原点
    if(self.lo==5):        #当五步采样完成
        self.flinesub=self.flinesub/5        #计算角度均值
        at=((self.flinesub-90)*3.1415926/180)      #计算弧度均值
        adjsub=abs(eval(self.aimangle.get())-self.flinesub)          #计算角度误差
        adjfill="red"      #如果误差大于 30 则显示红色字样
        if(adjsub<=30):          #当误差小于 30 但是大于 10 则显示橙色字样
            adjfill="orange"
        if(adjsub<=10):          如果误差小于 10 度显示绿色字样
            adjfill="green"
```

```
    self.gra.delete("fline","adj","win")        #清楚上一次产生的图形
    self.nowangle.set(str(round(self.flinesub,2))+"°")      #重绘角度数据观测值
    self.gra.create_text(85,25,text=str(round(adjsub,2))+"°",tags="adj",fill=adjfill)       #重绘角度误差观测值并附加误差颜色
    self.gra.create_line(200,50,200+125*math.sin(at+3.1415926/2),
50+125*math.cos(at+3.1415926/2),width=5,
fill="red",tags="fline")     #重绘帆板示意图
    if(adjsub<=10):       #当误差符合题目要求时
        self.gra.create_text(100,175,text="你说我们棒不棒！？",tags="win",fill="green")       #询问参赛小组是否棒
    self.lo=0       #清空采样计数变量
    self.flinesub=0       #清楚积累量
    self.ang=ang       #角度数据返回
```

7.7 安全退出模块

此部分用于实现退出时电机轨道归位和多线程终止标记功能。

其实现的 python 源码如下：

```
#安全退出函数
def safeexit(self):
if(self.disbit>0):       #如果步进电机计数器不为 0
for i in range(1,abs(self.disbit+1)):       #步进电机退避
self.ser.write("B".encode("ascii"))     #下发 B 指令(指令由 ascii 码转换为数据流)
time.sleep(0.15)       #等待步进电机动作
self.exitflag=1       #退出标志置位
time.sleep(1)       #等待线程阻塞
try:
self.ser.close()       #尝试关闭串口
except:
no=1       #如果失败则代表串口已经关闭
exit(0)     #退出程序
```

八、模型测试

8.1 模型测试设计

为了检验此设计的可靠性、稳定性和高效性，小组设计了如下测试方案：

1. 通电满功率测试，检测最大角度维持稳定性(60°)；
2. 分别在上位机输入 30°、45°、60°、15°、30°和 0°进行变换测试，并采样数据；
3. 数据分析与处理。

8.2 测试环境说明

测试采用上位机环境：

操作系统	处理器	随机存储器	软件环境	串口芯片
Windows Server 2008R2	Intel Core i7-3615QM	8GB DDR3 1600MHz	Python 3.6	PL2303

测试采用软件工作环境：

Python 版本	GUI 库	串口库	系统库	延时库	线程库	数据流	数学函数
3.6-64bit	Tkinter	Pyserials	sys、os	time	threading	Unicodedata	math

8.3 测试过程

1.运行 windows powershell(经过后期检查，上位机程序存在线程杀死 bug，使用 powershell 方便结束程

序)，使用下面的指令启动程序：

python ./自动帆板控制系统.py

2. 程序界面如图 8.3-01；
3. 容错测试：
   （1）、输入数据格式错误测试(图 8.3-02)；
   （2）、串口错误(图 8.3-03)；
4. 运行测试：
   （1）、启动满功率测试(图 8.3-04)；
   （2）、调整起始(图 8.3-05)；
   （3）、调整过程(图 8.3-06)；
   （4）、调整合格(图 8.3-07)；
5. 记录数据并进行处理。

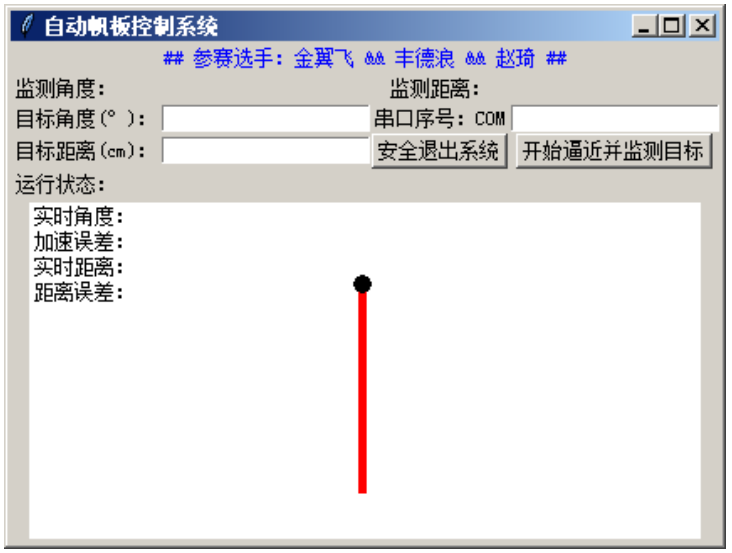

图 8.3-01

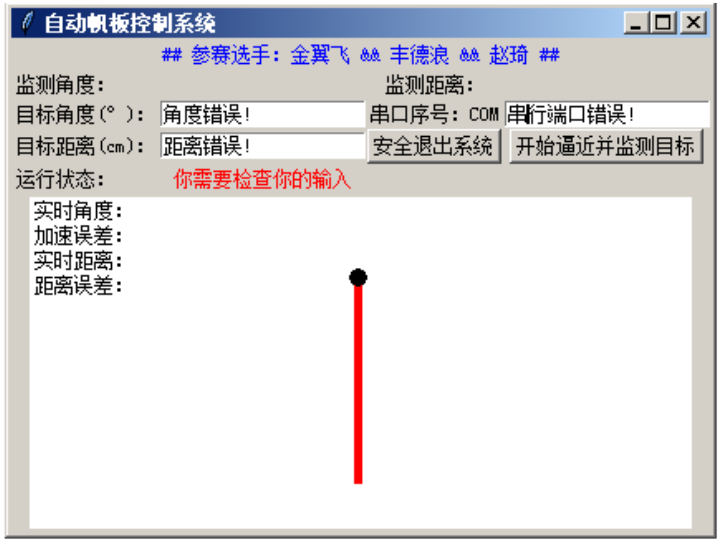

图 8.3-02

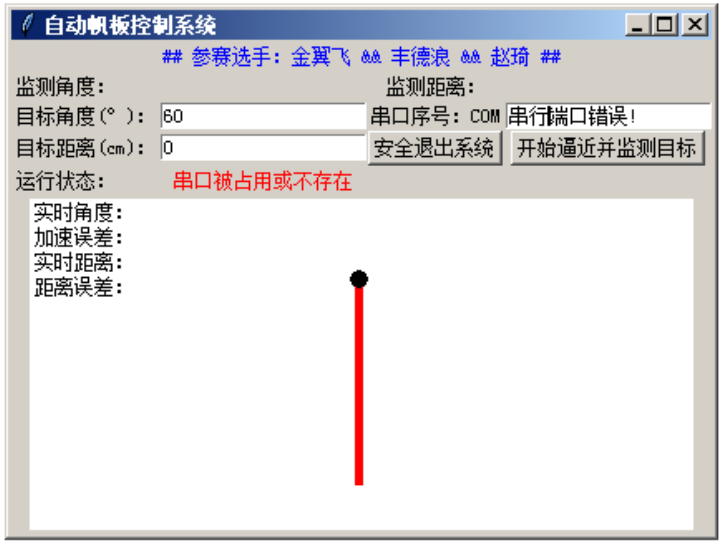

图 8.3-03

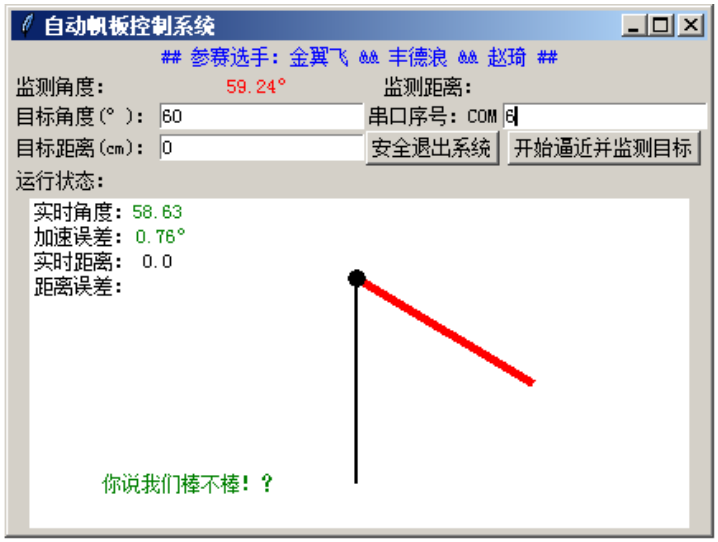

图 8.3-04

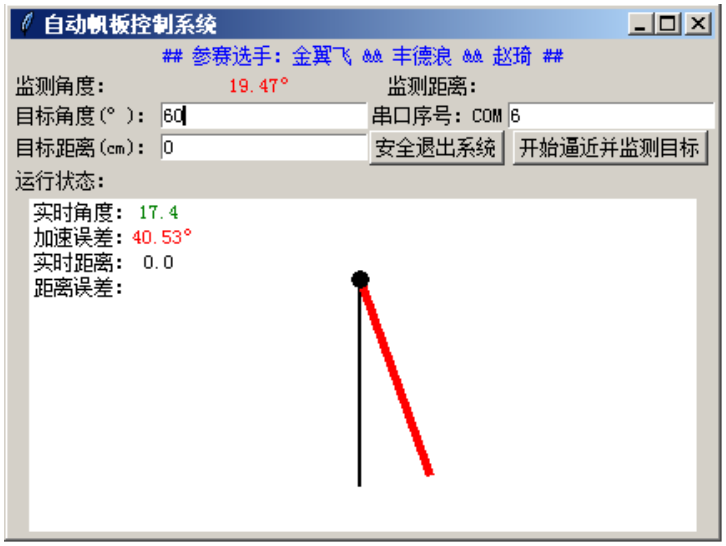

图 8.3-05

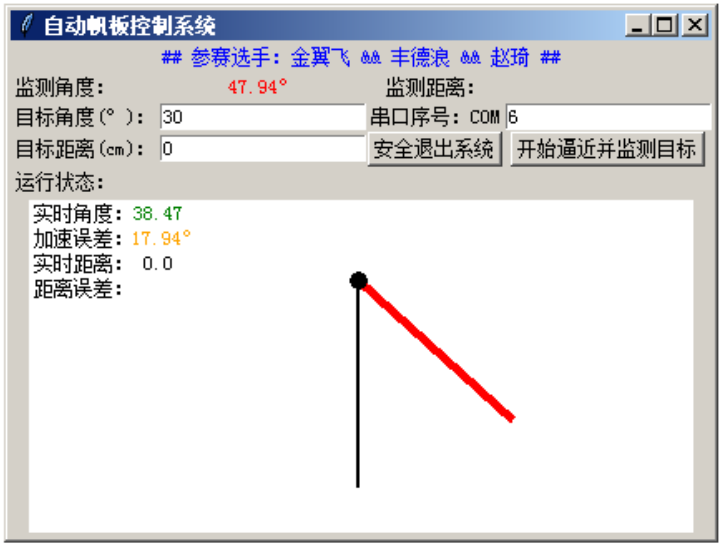

图 8.3-06

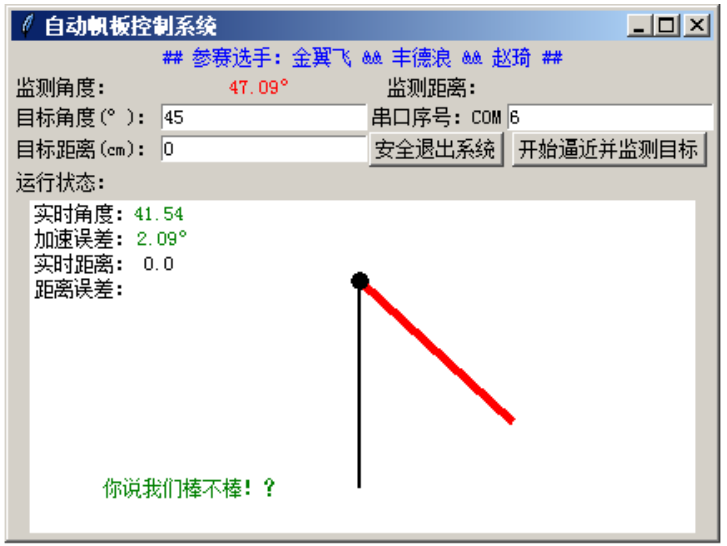

图 8.3-07

8.4 数据总处理及绘图(MATLAB 程序)

我们采用 MATLAB 对收集到的加速度分量数据进行处理，并绘制总体的散点图。

程序主要实现矩阵分割、角度计算和散点图绘制，为了确保数据真实可靠，我们没有对数据进行任何数字处理。

实现该功能的 MATLAB 程序如下：

```
load ser.txt ser      #载入测试数据
ser=ser(:,2:3);       #取 y，z 方向的加速度分量
ser=[[0:0.051641:87.73806]',ser];      #加入时间序列
x=[];      #角度容纳向量
for i=1:1700      #遍历矩阵
    if 90-180*atan(ser(i,2)/ser(i,3))/3.14159265358979+11.5>120;      #按照 python 中的函数进行处理
        x=[x;90+180*atan(ser(i,2)/ser(i,3))/3.14159265358979-11.5];
    else
        x=[x;90-180*atan(ser(i,2)/ser(i,3))/3.14159265358979+11.5];
    end
end
plot(ser(:,1),x,'.');        #绘制散点图
title('自动帆板控制装置测试数据散点图');      #绘制标题
xlabel('时间(s)');        #绘制横轴
ylabel('加速度角度(s)');     #绘制纵轴
```

最终取得的数据散点图如图 8.4-01.

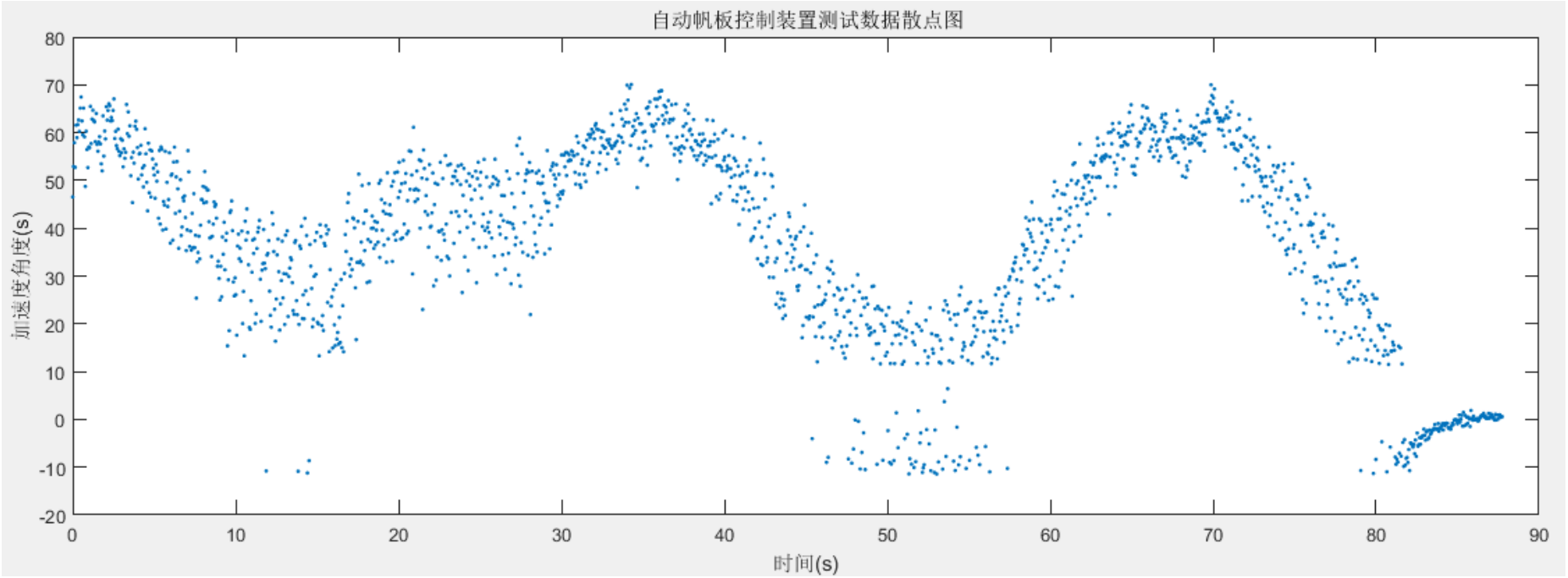

图 8.4-01

8.5 满功率测试

样本数(个)	时间(s)	方差	合理极大值	合理极小值	平均值
68	3.511588	20.1278	66.0467	55.8024	59.5108

如图 8.5-01.

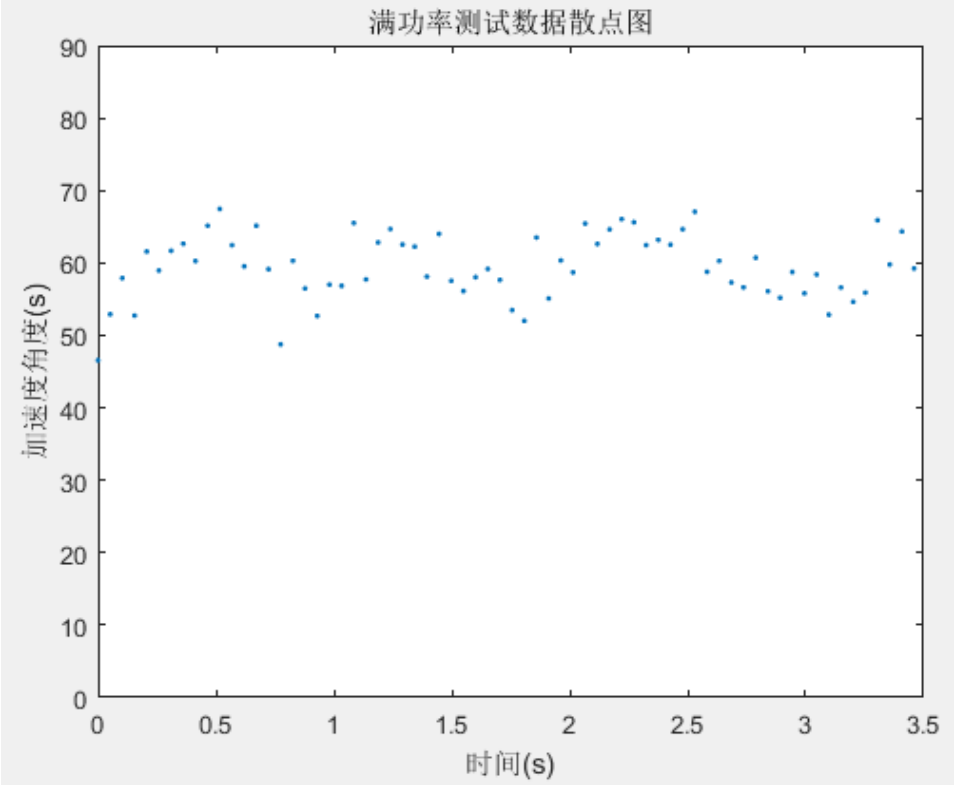

图 8.5-01

8.6 调节变化测试

测试项	样本数(个)	时间(s)	方差	均值
60°-30°	127	6.56	无意义	无意义
维持 30°	203	10.48	112.614	31.0822
30°-45°	136	7.02	无意义	无意义
维持 45°	272	14.05	88.753	42.0910
45°-60°	138	7.13	无意义	无意义
维持 60°	221	11.41	113.8133	53.8371
60°-15°	121	6.20	无意义	无意义
维持 15°	297	15.34	199.6460	17.6839
15°-60°	141	7.28	无意义	无意义
维持 60°	223	11.52	54.5250	56.1012
60°-0°	153	7.90	无意义	无意义

平均维持误差：3.3474°，平均调节时间：7.02s。

九、设计总结

9.1 设计缺陷

在设计及调试过程中，我们发现本设计存在如下缺陷：

1. 风机噪声过大；

2. 上位机程序没有对串行数据做初次检测，容易导致第一个数据错误；

3. 硬件部分结构脆弱，容易损坏；

4. 设备精确度不高；

5. 上位机程序存在无法正常退出或关闭串口。

9.2 设计改进

本设计可能进行如下改进：

1. 结构加固，外观美化；

2. 程序 bug 修复。

十、开放源代码许可

10.1 开放源代码许可

本设计所有程序及设计方案由设计人(金翼飞、丰德浪、赵琦)保留所有权利，但为了推动技术发展，便于广大爱好者研究交流、促进改进，设计人决定开放源代码。现声明开放源代码许可如下：

1.任何组织和个人可以复制、运行、修改和再发布这些程序，但所有由本源码复制、修改或再发布的副本必须注明原始著作人(设计人)，并采用与这份源码副本相同的许可开放源代码；

2.设计人在 GPL 公约前提下保留此副本及其衍生副本盈利的权利；

3.本许可授权任何组织和个人使用本手册涉及到的包括但不限于 C、Python 和 MATLAB 程序源代码；

4.对此份设计副本被复制、运行、修改和再发布可能引发的直接和间接损失由其复制、运行、修改和再发步主体人承担相应责任；

5.设计人放弃直接通过对本副本提供技术支持而盈利的权利，并不承担任何技术支持责任；

6.本许可未提及部分依照 GPL 公约执行，本设计手册完整性由吉林大学开放源技术协会和金翼飞个人 PGP 数字签名进行确认，但吉林大学开放源技术协会和金翼飞个人不承担副本引发的直接或间接损失。

10.2 吉林大学开放源技术协会数字签名

签名文件与本文件同步发布

吉林大学开放源技术协会 PGP 公钥：


```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2
mQGNBFioU+gBDACzOSGg7jRAO/MoGuLeLdURWBln0IjJaAlFHQ8NOTk6WFSti94Z
r1UXYIvHTc6mq4VP2z+jnWksES0q/rcDxLdoev6gbqQ8CsHdob0MVuoQnpEItfnu
UL+zW+s1tUG+qm0J4Rjj0q69eYYi4MEv7pR9mELr7Dkz68ZoIifdDafHK+Hi1l/H
ifb637oz50lo5zGYBL++5GtwTexvH+u+v77UI/KXwaZrVCDt8SaBzei9YIgG6xOR
MnxTAOLXmTRksDOMTy9/I449/c3KJVao0tpE43k608v5QOHMJJA5xHFWEwdxZUEw
0AIFI0oA40x1XtEv3w1YGpac0LcHQaGjAFqvARXBYhkQ+ibLncFmeq9dV9ODokxq
2klG5fiEP0jJJjgdajsengOZwub1iw/X9jfyYuId2tihe+0PqP10p6fYnVLiTa6c
KhpvWivMnaUGce9/hzcpcE39uPwCm2B6wiWFWoKEr0qu/V1DcdzAzuKvaw908rNB
kHvPagMs25LbEHsAEQEAAbQbSm1uWW1mZWkgPEppb1lpZmVpQGp5Zi5jb20+iQGw
BBMBCgAaBAsJCAcCFQoCFgECGQEFg1ioU+gCngECmwEACgkQ1zY2vUqqm9mhmQv/
WncQe8jmCK1U9ZK6zJssUHuPLzRzTLcn0rC+VKXHYMBrufncxGY0d1/jSA2Cjr30
bAWTH9eIO/fQZtAcXrk3ZYt3+151hNVURLjvW11un3E63tA1vv+o5u6M903uAHEi
XqY0rxzrbZg/jGfWEnRyeaEzs/Fqqqn4cwOVwx41kYELQEAo9ROU3/TEV8dGjyBW
VM6uC0114vb7LsNpYpW3qO0bVgPeQCHvbre1cF668ZGwZpUYeHKTAUdnN8+5w2VQ
hdBFCBmtZvrKr3MnrCPTFRI4zQ0Z00Ea+JDkJwg1BpOB+zfTq/ngem7/JHwOpQ0Y
ePefVPnio0YQKVA0YBWYzBdI7hKGh0y32MCKYZuEGgoc2+7vXxAeTjPkWV3M1FeL
IMUWBbu2ehL+2HNgP2JxnqxMBb5QeVfnwn10ZhPm2mB/EE4wK1F2M/DASJpsni4A
aTFkof8jPUMf4ZTPMRedba05W4g50aJPALJNBR0yYxvg2D3mauTtEQ9eDpn1CUm1
iQGcBBABCgAGBQJYq+SBAAoJENfs8nJCb9+7IIcL/3cEGaTUKCEiw1mDDN1SzBM1
Pw+0VB1+BhYHNwVom2Dq0/ICTMkq3C/pfWfN9GDwx5sLLrya7/djrFE1WnILxmuM
```

Cbp9r4Eb5+yhvaZZZu+Q6qckWiK21QfVLV13Iy0shbCTxHjSWbEbhTdw8QJyECoo
wROFuBauJKvxtRLCmk4g4jvU2+vMRj6DK6dK3IdEoG1d06tfTb6LriaCkoNkXtan
pt5oMUwWV+kqebDQ54KNq1GT4agsZd1S4k2AifvmcWMVgToW0jM/wg94IszjbQim
jUKEIjBU3/wknXA4nrte1cICGYxgS7fTf2egPD4hvWeC/JKRb0rhxkYQhpbE+1/4
uCbYa8JwsmoFPPVZ9WSA7v4ieANV1BW4XKRQepqWdx+2XZqXpqNirzPBf9DY0+0k
UpQ/1nJHmFo0g4QTAg1hn3ymf44iZup3Gxt3BmAAK0R++AGGPqio9YLj3a3aUNIm
d4hwp+sbIxyRnvMx5i0EhqyTyAfmdXhZQ/qGHvCQZIkCHAQQAQoABgUCWLPjvwAK
CRAEvo+fk1tMFsWHEACJBuNd/NnpibxqZTcRDqNHVS5nmfw47V6eSaFAJISOQAZy
G1B+SWwBJoS3/S/rXPdXM402r9G55GEcAVhfvU07EHUvWl//R6YJILG8NVFaJIbV
tMMUT19vfTJ7jJbc2MkUKYZDuMbU7wKfxHF4qx1sTka09+sC80V+hNaC6FrrVpL+
4tbDKqX3oRiTUWQGkg99cM7oI3p064BMErm0J7LoaxQjjznSvMTKfljHaqriwKL0
I87Ew5bCjZEECqof3wAvF8MLwQENjHQiUEIUVDK3Kx6CqZK0wcUtvCyXfvVXuT4z
UzBI/YbZSAeR1xR8vrwwGdQj22KR1wM7hBHmZU0sqItFuFs8hSV7mXEad0j3EABS
0zNMFjcAWtmrLq6NZq3IQYvJofv6zZtm8NZEp4V92ebg5DFJS51eH3QAYMFJBZPq
6JgFXWWU6/g/3zQsC4inh8jYMAFy3eLrv2f81uT1AFf9VP3EL4867ikLxkCNU9n1
rHKsgYyfRCBJMmrsbZvG0uF6AasNpYjKrHeZH+9Jm5F0fgZv2DH8D1HyeHy3wBtm
1L1TA8p63cHyL9Dvx30WhoYp1CnoQYX3vGv4ZglqsmZ0uBSUzXhRB7JSiZGoa80a
waesHjf0k6wAjUqre0suUJUPwNAR0ptTjToDvHWddHv7DkYj0x77JGCP8/MVo7kB
jQRYqFPoAQwAqenNhILAZ12TuWtnYw0EI0x6hADY8AXe/CxiIssKsCrvv9WZcw84
cAZczbiq4Yv10eR00Y+1yj61qAc/0J/tb11XTkYjxK/tSpiX5YqDiQ/usEMGjvd0
zjok9hCqvFvL5FEy1ErpZSAioRf9JEQxX9F+cHJTpjFiJm+xgeeTYJP7gC6tM161
UsoujPpM4D3Buw04zpYpsEEViHq/0qswPLGIQ+1CcF1CIr7ZVbsWkQfvc9Tx384B
J73sSLd2aMZxaZmRF4/f/nRq/q7RCZ+4uEACheag1MOE3S6tEzKAskY7aa0gY55x
MYdcbW7gyWnKwG34DAI1+LG5H/2ne6eN00/A0UA/Z6634dmCy6CbPbMoNcjNPq4U
mr+s3h2NpUagcFAriz7wp0AsXCIdv2I9uAPL3R8J0AB+Kb2v0Fu8ukCu9WntBuai
R1fJMI6yRR16iB6qVgSN9CSD+ZfZGX0Habo2QDAHiLFJcnTBc8p+U8ZcVSKNp916
1LU6Ato/iW1BABEBAAGJAz4EGAEKAAkFg1ioU+gCmwIBqQkQ1zY2vUqqm9nA3aAE
GQEKAAYFA1ioU+gACgkQoXza050HwDH19wwAoViaFv2i4Ri01ZszP33ZIfEQVwXU
7gJPNTKXjW4VgYG80zZm17IAupKh8MADM1a0uHuQA55K1+F/USLqpa1B43rxpyzt

2MZGPm0yG1tV+cbD0Nyc37wQKQR4hNL5J2v6kuSASJb4WsM5LLQEzTwtY3tliR/s
+9eky0uj9mmjsjfwhcyxW/0qnW7W41jYicZsBB0ffCzF6+qx1Hapapcim0cxK/YS
vjTYFSg7PqNNg+qYI1V/6kgm08KUsLMCQmkFY+t1toTbaI6XS8R5gaf26qrjinvD
W321RNCaUGQ5eJYY46DEE2xsmcp6hp1tPqkPWMBSpyUz78XrQ0W73z0Gv1DEQS8H
o+H/AwpVUb12Qw6+AVdV/VRF+BxB8XQ7Xm1/xrF05mEh4P2EGxk01U0YGkk/Khak
p5zsyUgL/RBfeu4t9+z+VFQm86aiCbWdtouzCrLTx9GEHMvkn9CDfGSNr/48ab3z
2PuJuGWJIMawNOmw1wEQETN5w1PMYQ3zuoQDEx8L/06I4w5Bs+kbMvpdueIGe1Rw
53LyRkEt6n+xM1+suttrHSVFZdpZVMkDjinFrpYxIoM0yZYxBnS0CBfvUoh8eYex
37wPTSRrKOZVegwRi6HACmLKo+nSCyjDA+Gf31A0BF3eX3ZtL1f/PU9cpSi8Y984
4fKEax0AGVwqFvTE4n4aIPdBvxj8JSDTgsSMuwfa7YWqxczbFhmg2NTpZherDUU+
a8bpx8borTjUKn0ZKBVwKtJqIpSazLTfBpWWaNTsg8vtnrAZD7U/p5uiNRHiPzT+
6tnKlRmJ2L1f4B3uWIENpmLTpfRW48dYDT+wsK3slyd3czwn4v/0AVFFS8cKya/g
qocCPvPZK+grRJCTX060Z3zNGWQ90eu66DE8qhpu04zqwaIhFHrTgs1TpxnVPyL1
8j/AS0JzBvqjUcTZ1KRlpZXU6r73VXk4OXb+03x8bwm17PmqVVvpEdLh4FzoUM2X
zPFFrtEtvvr0nw1IPSxN93Ru30HmVk4C9Zb163e/MbkBjQRYqFPoAQwA5/MSvEMz
iR2Cw+fgrBFTeuWtoygeHFWYw+7nCDizmRmwi5SyWbgPCBWwRhVoCfDMm1QfzFVs
9RT2GmE/AQgmKZnrqFVyxRs+L1lyGeXTsPD4eGPMtlm/YwIHrh9SnsUCnS5G0c14
ecYuur91bmUtVaafx312Jsv9CMRKHE71/aVsofDu37BxtzvhaEzJoYxiNb1HiZNU
03ALX7Ma1AI91ImPq9yb6ca8LzYaPTFj4VjfMOSZYo5740HXBPIpMyZRRBkdCZaJ
hGW2wW8XenX/ZXaJjg815ZVUaya0p8NwC0FbyaJ45vhYh7SgFKqGE6jsZfY/Z2+i
LqMv/s51pZaprKPC3U/WmaqWf7QTUYtCGHsnJ83ZA1v/DBCFv0s1xZuVKsoaNm4D
T9KDX6nDQ/IAcgQ3H6hraai/vhFHvyq0UkJq+5a12R0ju/qeG1VzAbZOFFF2e0e5
7A0RbDUenZrcE9Fs2ciantE9EiUGnhqSbn9LcTE0qp05MMZhhZii1srrABEBAAGJ
AZ8EGAEKAAkFglioU+gCmwwACgkQ1zY2vUqqm9mqSQv/Wyfug+nQuSYMA8xXqWx0
4ZtDccyhsOOY/Goe/li5CuA8QQdmohoTpbZQV//nbqw0uNy5b31prSNixjsEFhDn
aGw11KV5H1Jnijat4ccA8d9z4y7nMnasYvYC7ESBDh2ci770brHPPWHN0e0J7kZ5
k2XR60Bbu8hR4yU+rPgyfXWE/0kD7p4H1uYJCinChFHH4D1aFFIXGWYczmFroZ9m
M5kPAYq0BcShilyft+xMNZQdt5/0XRp90/LZuWpuHsLUcmU24hxC0trzB37F/B8m
wn0aQqVet0L9NM1qJYUtarD5XjcQtkjXnM1tv9bc4PvvWH29Un8b0Y4XhgTTIBgw

yATFZmW8Yu7HpAZL1kZrkoxi6hsXc9atFTS6gi7m0X0918X/AMtSqSFwn1MX1oYk
ucQZkPrRbpSwUyPiFd735ItKsImbjfT1/0+9wEFBM0RX4IRQTp48wct1szkx/Dqq
8g9SFJ1xYyCCWKpzf4j71PmcukGXQD2Vsq2a3hP+dz1a
=iPVN
-----END PGP PUBLIC KEY BLOCK-----

10.3 金翼飞 PGP 签名

与本文件同步发布

金翼飞私人 PGP 公钥：

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2
mQINBFivB4wBEADbrz+TtQMm13s/cctf8Yf0IgEkiZ6YnpqkfvP/mCL0KAJw79e+
64L9HoBtr2V4RIW5S/WRK1/pfYSiBbjBcSnrWwKA01xK0QH2Tm/sDEUa9AoLRCm6
z4mhqRWDAS9GN3DTMWTMG3Kr3yfvPi00y0j3DTFbTFtT3QRvYh/nm1tXwCC1tIR+
0v94q1dxR2RPNwiXRk101Dh1SLd36HIn0vTjpHo3J07Li8ki5RnWYMenfhTJ1wrM
My5H6FBaDG3zdxke1FQ7/1GOU4dvNG+WkFgDtf/55ePT8/0LghIaD8tau4pDyeYv
P/IgAGnojz2PiIw3arC8FPsPICGmenPnM6FhpkU0BnHvC3h3OKyQ5YSBmy1BxhKT
3IrPhKxormckEgyNCQWX+r291IrxocyCdN7SLmQhrgr5Q7MxYTp/wpzKWdAPXMBv
BbG6kaxZ2fwhEbgAi1odXKASW3K5cKVBY/9HD0xo1ZtYDUz1etFnF0fHymDrRjw8
rSh8kuCeZfBt/yaeUI+SQqwkK0ynt8oogt65cwIrWwdocdycH2v+tD/fMkITTYfp
194MyUkHb+d1MjLr5KU4DTgRgaYYgnKbpMqVfVFbS1+IPnVqUTz40dX/tRFCSS12
XLoU1vjPEGjNV4aCwFVNY0BvW0xa+RGj086ejwXkJC7wSsU2SELW1o8DVwARAQAB
tD/1kInmnpf1pKf1rab1vIDm1L7mupDmioDmnK/1jY/kvJoo5YWs5YWx6YOo5YiG

KSA8b3NOYUBvc3RhLm9yZz6JAjkEEwEIACMFA1ivB4wCGwMHCwkIBwMCAQYVCAIJ
CgsEFgIDAQIeAQIXgAAKCRAEvo+fk1tMFgU8EACLSfGNp7tmKiKsgsto9LhAuSsU
7g1//We2Z5Xefnzf/2tnyoiMXrMq107HsgjSiV293fmsRgeGVwRFFL1GUONQLP5S
W8SRIr+Z58jIf7ZZX5mRhGwfCsDhNGJVqStPttVv91GYaaUvcS1YkQa0r5MM/GcN
OtyUEmF41tJX9hH9BxK601PzPDMCQBZyA5HRvdDZeH82JYHjhMjptmfqpCf8zgjN
1srrFc6S02+83/j1RGzq6BuC4aRLv84IvrfaY49LRnZKjy5J+MxQDiWwkG/gm8dc
sKih/AAZsYj8FuoI2vyIuWtwghksehQQw0MVqpqFEP1bvK/HeaFihq3sKe0Tr0V9
ILhrSAhDR+HSh+nYS8bAfs2yJrIoU0UB6mtZE8a0bsQcsrZkTsKEgBjsqEmqcePO
+Ep3PAWqz1RZpGNy1sXy71IW9n91pdEEhB/K0Xi1NsjL5afFKqk6VPPYKf+yqe1t
+iLkX8/brDVEAzDMYitG/D2O9G2jFvs1H0ee8yk6KiiGFCrmtNWPrIbtjxN2uSYj
yGmN23DZWIIewu7CBbSWG6x48RQwp/EAWNkIbjXaMa4PwQvsAnAH0qRZN0DMk8Ye
Cm3Wu9c7LIPXdHBM0tWDv9KgLS6O6VQ2NZE9Gh30JY32zF7Zjhv2Yl3L513ZPUCH
LIrNr5DvR70aDuq1jrkCDQRYrweMARAA2ThIBWsT0AU9EGfN1qaSuzuMiKie1fqb
UpjXkpW+oeRhbDimCz9/J1FrFqcyRY7WuOhC5nf3b+H/Xx+1kgsXKy0mUpwxef3a
PyPbLpqU10eNFq1p0syi0hAnNFXoWioSTc2BAX3/rhqntGQCpg4XLkEkbPDA9Yiz
0aBJgTG4WMH10ANvN5dtGYGh+CHuztoQaF0z0xZed7rdStmdKbTwo7LdX/BGAoEz
owjZ1W/t8R1S/2ZrVAFXAs80uuRsy9G6hzKiQwb8b19MgFsq4hC1ce/BcQvM2gCu
wTjJFbUC9uwi0EnnFuSaLfMtR1dVYeE81fyuXZ5AGsoqpPqcWpaP56xYMsgWgjZ9
C5NxhkkeaE8wO6nUmAfYtvLgxoB2fWqDRsadLZQiYSBygjfBrs8JhwcBO0KsNh10
w4m+2dq/onE0bWDjwx09WDz5XmgEozWQUi1UXq0d4jKjikyfa6rfLuZbEtZDVCHY
2V5+0U17LzrnBAXiLQZmf0utc1panivBO04hwo7XUkRi4maQFfE832Vxe9gpMPwg
W3XPwsuT11dWoggF+pPY6EWovDRBHQaqJK39dgzB/kvs8UgGaoyqTnJH4svJ3icV
a0e6pQsv0+1YoYU34WutmR30391MqKni2ntc40eJxR1AXD0CuJzwm5+AZo9+gfSV
AJU2frHkyi8AEQEAAYkCHwQYAQgACQUCWK8HjAIbDAAKCRAEvo+fk1tMFi20EADX
083NDTFmLJvT7uPU9jlinCzYeLLraaBhMrQg/tDrUb8Lk6SYznmA2R3J9gWBGQn1
1YVL8ZZmKqW13rKhH2uVP3AEIX9+GQe2nuBIdow/IbUeOmW0Z3x5v6neyrnEffQq
SZG8K/c10AQ8a1+Sj9Emjp52NzveRtVjV8PNHx+21SC1F1cDIUDLX1Pvcq8KIrac
Dcfw609Yqev2xdMi9tIizpi6wn5JwFm7k85d77CaQrb4xBw3QikgHKZXPJ6Uy8Fw
uUaxfZJZZ5xRwnTRfA/sMgwEuiaveG94MCRpfxoiSqGhgtMmrkjZCc8BevBuV2SG

NU8cERk1X5P/v4SnXj2AVu69DhXSghN/bTh7jwHb5GcF2cHfkFwBZSKnEi8iX5nG
3haQnTpO2ngU/0y4IvGjUpTBe8ybFBCz7aepDe+U7m0HXAaDTn1809t4OB21BkBY
PC+z3C8+JMtrzq6aoCUx12Y9eT/cPwWoNyygoaMErFqg9aXdQcRLP/3zogw1urE0
jp8AtRLp8e9YRFbTLt8NbYUTimzs5SJLWkGyrEFM4Dw7ekyvwHvrP6JdFT3ZZa7y
f1Pd8efZnXr2QJdH1pGJDiBgghHAt6/pxGngx/xBUiGX+zrifF8/vQjqeLQ4sM0M
DSBoNedWU005DMaV60fpcH6P1nejqqMMIMoD0f0haw==
=M81q
-----END PGP PUBLIC KEY BLOCK-----

与本手册相关源代码和未尽事宜将在未来一段时间后发布在吉林大学开放源技术协会 GitHub 仓库中，仓库首页：http://www.github.com/JLU-OSTA

初稿：2017 年 04 月 04 日

第一次修订：2017 年 04 月 13 日

开源发布：2017 年 04 月 20 日