第5章

# 四旋翼飞行器的飞行控制

## 5.1 飞行控制板控制系统总体框架

本章的系统硬件主要由三部分构成：Android 手机客户端、WR703N 路由器的图像采集传输和飞行控制板及传感器，其总体构架如图 5.1 所示。

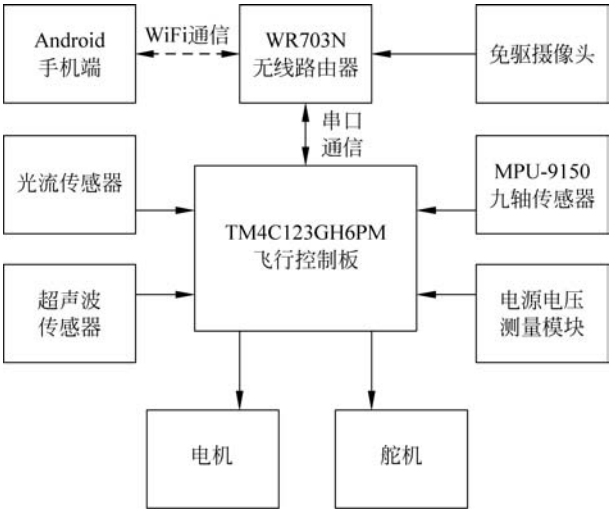


图 5.1 系统总体框架结构图

## 5.2 所用器件

飞行器的各种零部件及开发调试器件如图 5.2 所示。

电源选用 1200mA·h 锂电池为整个系统供电，惯性器件选用 MPU-6050，高度传感器选用 US-100 超声测距模块（自带温度传感器对测距结果进行校正，同时具有 GPIO、串口等多种通信方式，内带看门狗，工作稳定可靠，使用 GPIO 通信模式）。NRF 无线通信模块，电

调，无刷电机（飞行器采用配套的无刷电机电子调速器驱动），螺旋桨选用 6045 桨，处理器选用 TI 公司的 TM4C123G，光流传感器 PX4Flow，9g 舵机，相影 HD720P 高清网络摄像头，WR703N 无线路由器。

图 5.2　飞行器零部件示意图

摄像头安装在舵机上，手机端发送指令，通过路由器接收控制舵机的运动，摄像头就可以从不同方向拍摄画面，拍摄到的图像由路由器发送到手机端。

摄像头和路由器的重量越小越好，所以选型上挑选重量较小的 HD720P 高清网络摄像头和 WR703N 无线路由器，分别如图 5.3 和图 5.4 所示。


图 5.3　相影 HD720P 高清网络摄像头

图 5.4　WR703N 无线路由器

图 5.5 和图 5.6 列出了摄像头和路由器的参数。

技术参数

| | |
|---|---|
| 即插即用免驱(UVC) | 成像距离：30cm～∞ |
| 免驱(UVC)硬件动态像素：HD720(1280×720) | 白平衡：自动 |
| 系统默认：640×480 | 曝光调节：自动 |
| 软件最大动态影像像素：1600×1200 | 光照要求：10Lux |
| 软件最大照片影像像素：4000×3000 | 电压频率：50Hz/s |
| 色彩压缩格式：YUY2(UVC默认), RGB24(Driver) | 影像静态存储格式：BMP/JPEG |
| 适应接口：USB2.0, USB1.1 | 影像动态存储格式：AVI |
| 显示帧率：QVGA/JPEG下最快60帧/s | HD720P时整机最大功耗：≤120MA |

图 5.5　相影 HD720P 高清网络摄像头的性能参数

主要参数

| | | | |
|---|---|---|---|
| 产品类型 | 3G无线路由器 | 网络标准 | 无线标准：IEEE 802.11n、IEEE 802.11g、IEEE 802.11b，有线标准：IEEE 802.3、IEEE 802.3u |
| 最大传输速率(Mbps) | 150 | WAN接口 | 1个10/100Mbps LAN/WAN复用接口 |

主要参数　3G功能　VPN支持　WDS无线网桥

| | | | |
|---|---|---|---|
| 产品类型 | 3G无线路由器 | 网络标准 | 无线标准：IEEE 802.11n、IEEE 802.11g、IEEE 802.11b，有线标准：IEEE 802.3、IEEE 802.3u |
| 最大传输速率(Mbps) | 150 | WAN接口 | 1个10/100Mbps LAN/WAN复用接口 |
| 天线类型 | 内置天线 | 安全系统 | 无线MAC地址过滤 无线安全功能开关 64/128/152位WEP加密 WPA-PSK/WPA2-PSK、WPA/WPA2安全机制 |
| 管理系统 | 远程Web管理，配置文件导入与导出，Web软件升级 | 其他性能 | 频段带宽可选：20MHz、40MHz，自动 |
| 信道数 | 1～13 | | |

**图 5.6　WR703N 无线路由器的主要参数**

## 5.3　四旋翼飞行器的组装

### 5.3.1　飞行器的整体组成

飞行器有以下部件：上下两块碳纤维机架，4 个无刷电机，4 个电调，1 个超声波模块，1 个 PX4Flow 光流传感器，9g 舵机和相影 HD720P 高清网络摄像头，1 个路由器，1 对正桨，1 对反桨，1 块飞控板，1 块电池，1 条固定电池的魔术贴，连接机架之间的零件若干，固定机架的螺丝若干。

### 5.3.2　飞行器的组装步骤

(1) 首先将 4 个无刷电机通过螺丝固定在底板的机架上。注意：必须是一条对角线装正丝，另一条对角线装反丝。

(2) 在飞行器正方向的一边装两个相同且醒目颜色的桨，以此在操控飞行器时作为飞行器正方向的参照物。

(3) 适当剪短无刷电机 3 根导线与配套电调的 3 根导线的长度，套上适当长度热缩管。

(4) 将 3 对香蕉头分别与一对电机和电调对接的导线焊接。

(5) 将热缩管套入无刷电机的导线内和电调的导线内，对齐后用热风机吹缩固定。

(6) 按照同一步骤完成另外 3 个无刷电机与电调之间的连接。

(7) 将 4 个电调连电池的正极导线选出，适当剥去外层胶皮后，一起套入较大热缩套内，然后与一根短粗导线焊接，焊锡完全包裹住导线后，移动热缩套至焊接处遮挡，用热风机吹缩后完成。

(8) 对于 4 个电调的 4 根负极导线采取相同的方法连接电池的负极。

(9) 将以上的电调总正负极线焊接到分压板的电池电压输出脚上,注意红色导线对应正极,黑色对应负极。

(10) 将分压板上电池电压输入脚上焊接上 XT60 插座,将分压板安装于机架中心置于机架层中部。

(11) 电池的正负极线焊接上 XT60 插头,同样注意正负极对应的颜色。

(12) 将螺柱通过螺丝固定在机架上方,然后将另一块机架用螺丝盖在螺柱上固定。

(13) 在上机架的中心通过 4 组短螺柱固定飞控板,此时注意飞控板的摆放的正方向。

(14) 将 4 根电调上连接飞控板的双口导线按照顺序插在飞控板上。

(15) 用尺子和划刀裁剪出 4 块大小合适并且相同的泡沫塑料,通过黑胶布固定在机架上,作为飞行器的起落架。

(16) 装好飞行器之后,先对电机第一次运转进行初始化的过程。用事先编好的电调初始化程序下载入飞控板运行一次,设置 4 个电机最大最小转速。

(17) 调整电机的旋转方向。电机对角的转向相同,相邻的转向相反,这样才能抵消飞行器的自旋。0 号电机和 2 号电机是顺时针方向,1 号电机和 3 号电机是逆时针方向。注意调整转向时,一定要拆下桨。把遥控油门开到最低,接通电源,遥控器解锁,电机以最低转速开始工作,观察其顺逆方向,从 0 号(正方向的左上位置)电机开始,关注一个标志点来判断方向,开关操作两至三次验证。若与预期方向相反,则把电机和电调对接的三个信号线中任意两根信号线交叉对接即可。

## 5.4 地面站

地面站的主要功能如下。

(1) 实时采集遥控器数据,采集遥控器的 PPM 信号并解析成真正的遥控数据,实现遥控器对四旋翼飞行器的控制。

(2) 实现飞控板之间的交互通信,包括传送遥控数据和控制命令数据,接收飞行器的状态数据,例如超声波传送的高度数据。

(3) 在 LCD 屏幕上动态显示飞行器的飞行状态数据。地面站实物如图 5.7 所示。

图 5.7 地面站实物图

## 5.5　飞行控制板的功能

飞行控制板的主要功能如下。

(1) 通过电机驱动程序控制飞控板输出四路 PWM 波，调节 4 个电机的转速来实现对四旋翼飞行器运动状态的控制。

(2) 传感器数据的采集与处理。包括超声波模块的高度数据、光流传感器的位移数据等。

(3) 与地面站的交互通信。接收地面站发送来的遥控数据以及其他控制命令，同时向地面站反馈自身各项状态数据。

飞行器正反面见图 5.8 和图 5.9。

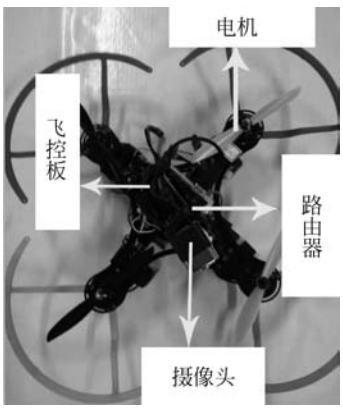


图 5.8　四旋翼飞行器的正面图

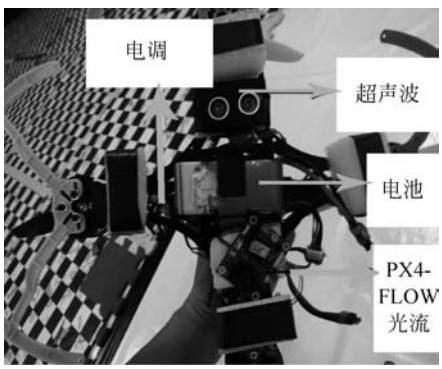


图 5.9　四旋翼飞行器的反面图

## 5.6　软件结构

### 5.6.1　飞行控制板软件结构

飞控板控制软件对于实时性的要求非常高，通常采用 C 语言进行系统软件的编程，在主程序中先执行各类硬件和软件的初始化函数，初始化完成后进入主循环。除了数据通信和超声波定高以外，其余飞控板所要执行的任务均在 400Hz 的定时中断中依次进行。数据通信模块利用端口触发接收中断处理，超声波模块则在循环数据采集到之后触发中断将数据存入变量。采用一个主定时中断，免去了对于中断优先级复杂的设置，重要的处理程序不会被其他中断所打断，易于确认每一次中断是否在下一次中断到来前执行完毕，方便了系统的设计，也确保了飞控系统的实时处理的稳定性。

### 5.6.2　通信数据帧格式

飞行器与地面站之间采用 NRF24L01 进行无线通信。在发送数据时，自动加上字头和 CRC 校验码。在接收数据时，又自动把字头和 CRC 校验码移去。NRF24L01 还有自动应

答及自动重发机制，故大大提高了数据通信的可靠性。可简化自定义的通信协议，帧长度固定，数据以先入先出的方式读取，高字节在前，低字节在后，空数据以 00H 代替。通信协议见表 5.1。

**表 5.1　飞行器与地面站通信协议**

| 飞行器与地面站双方通信数据包格式 | | | | |
|---|---|---|---|---|
| | 字节数 | 数据内容 | | |
| 字头 | 1 | 01010101 和 10101010 交替发送 | | |
| 接收方地址 | 5 | 0x11，0x23，0x58，0x13，0x58 | | |
| 数据 | 32 | | | |
| crc 校验 | 2 | 0xFFFF | | |
| 飞行器发送地面站数据内容(飞行状态数据) | | | | |
| int16[0] | int16[1] | int16[2] | int[3] | int[4] |
| Pitch | Roll | Yaw | 高度(cm) | SumX(cm) |
| int[5] | int[6] | (地面站通过 16 来判断是否为飞行器端发送的数据，是则显示数据在 LED 上，否则不执行) | | |
| SumY(cm) | 16 | | | |
| 地面站发送飞行器数据内容(控制飞行器飞行数据) | | | | |
| int16[0] | int16[1] | int16[2] | int16[3] | int16[4] |
| Pitch | Roll | Throttle | Yaw | Mod |
| int16[5] | (飞行器收到数据，判断是否为 13，是则执行飞行控制命令，Mod 为解锁闭锁指令) | | | |
| 13 | | | | |

上位机通过串口与地面站连接，再通过地面站发送指令给飞行器。上位机发送的数据低字节在前，高字节在后。上位机与地面站通信协议如表 5.2 所示。

**表 5.2　上位机与地面站通信协议**

| | 帧头 | 数据类型 | 数据内容 | 校验位 | 说明 |
|---|---|---|---|---|---|
| | 2 字节 | 1 字节 | 28 字节 | 1 字节 | |
| 数据长度 | 一帧数据总共 32 字节 | | | | |
| 帧头 | 0x9c 0x5f | | | | 帧头 |
| 数据内型 | | 0x23 | | | 让飞行器写入上位机发送的 PID1 值，roll/pitch/yaw |
| | | 0x24 | | | 让飞行器写入上位机发送的 PID2 值，high/POS X/POS Y |
| 数据内容 | | | | | 未写入为 00H |
| 校验位 | | | | | 第 0～30 位相加 |

续表

| 地面站收到数据内型 0x23 时发送给飞行器数据 | | | | | |
|---|---|---|---|---|---|
| int16 | int16 | int16 | int16 | int16 | int16 |
| Roll 方向平衡 PID 参数值 P 值 | Roll 方向平衡 PID 参数值 I 值 | Roll 方向平衡 PID 参数值 D 值 | Pitch 方向平衡 PID 参数值 P 值 | Pitch 方向平衡 PID 参数值 I 值 | Pitch 方向平衡 PID 参数值 D 值 |
| int16 | int16 | int16 | int16 | 飞行器判别 12 执行写入 PID1 数据 | |
| Yaw 方向平衡 PID 参数值 P 值 | Yaw 方向平衡 PID 参数值 I 值 | Yaw 方向平衡 PID 参数值 D 值 | 12 | | |
| 地面站收到数据内型 0x24 时发送给飞行器数据 | | | | | |
| int16 | int16 | int16 | int16 | int16 | int16 |
| 高度平衡 PID 参数值 P 值 | 高度平衡 PID 参数值 I 值 | 高度平衡 PID 参数值 D 值 | X 方向悬停 PID 参数 P 值 | X 方向悬停 PID 参数 I 值 | X 方向悬停 PID 参数 D 值 |
| int16 | int16 | int16 | int16 | 飞行器判别 14 执行写入 PID2 数据 | |
| Y 方向悬停 PID 参数 P 值 | Y 方向悬停 PID 参数 I 值 | Y 方向悬停 PID 参数 D 值 | 14 | | |

## 5.7　光流传感器

### 5.7.1　PX4Flow 光流传感器

PX4Flow 是一款智能光学流动传感器。传感器拥有原生 752×480 像素分辨率，计算光流的过程中采用了 4 倍分级和剪裁算法(4×4 分级图像算法)，光流运算速度为 120Hz(室内)～250Hz(室外)，高感光度，有 24×24 高像素。通过 UART 口与飞行控制板连接，通过 USB 口连接计算机调试。

PX4Flow 调试平台为 Qgroundcontrol，安装好驱动后便可以与计算机连接。Qgroundcontrol 是专门为 PX4Flow 设计的地面站软件，通过该软件可以直观地看到修改 PX4Flow 参数的效果。

PX4Flow 可调参数参见第 2 章。

PX4Flow 通过 USB 和串口输出 MAVLink 数据包。通过读取这个数据包，可以得到 x、y 相对位移数据，用来控制飞行器的飞行。其中 MAVLink 通信协议是一个为微型飞行器设计的信息编组库。它通过串口高效地封装 C 结构数据，并将数据包发送至地面站。

### 5.7.2　Qgroundcontrol 软件的使用

主要使用 Qgroundcontrol 软件的 Analyze 功能，对光流输出数据进行观察，把光流调节到最合适的工作状态，然后再安装到飞行器上使用。

通过 video downlink 功能，能接收到摄像头传输的实时图像。调试显示界面如图 5.10 所示，控制参数设置如图 5.11 所示，修改完参数后，通过 set 键使修改的参数作用在波形上。调试平台能观察许多参数的波形，如光流到地面的高度，x 与 y 方向的光流值，其中通

过观察 integrated_x、integrated_y、integrated_xgyro 和 integrated_ygyro 来确认目前光流是否已经达到了可以正常使用的状态。integrated_x、integrated_y 分别表示 x 和 y 方向上的位移量，integrated_xgyro 和 integrated_ygyro 表示 x、y 在三轴陀螺仪补偿后的位移量。

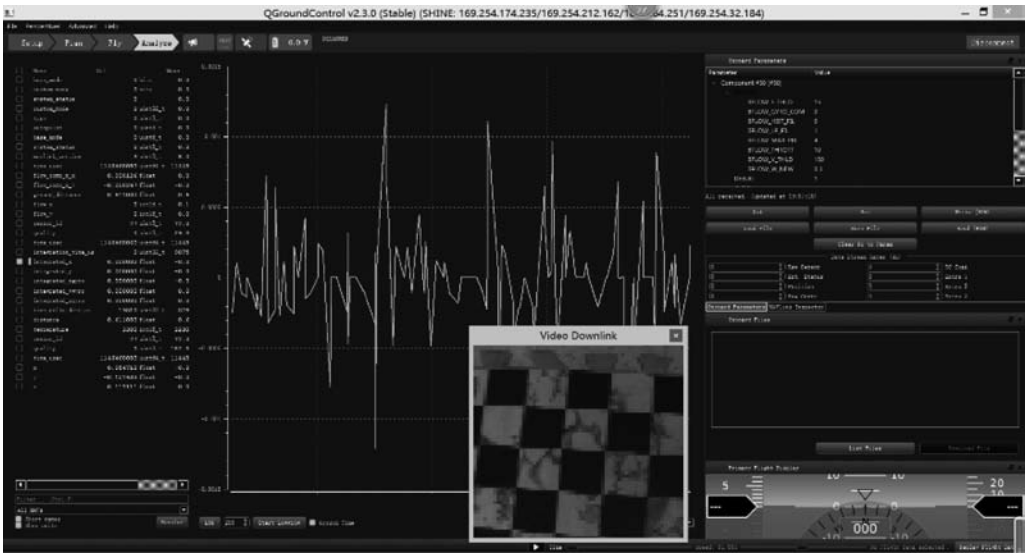


图 5.10　调试显示界面

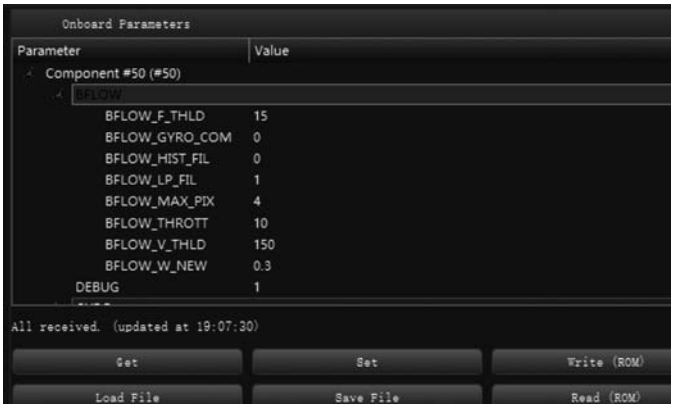


图 5.11　控制参数设置

### 5.7.3　PX4Flow 光流调试

调试 PX4Flow 主要通过修改参数完成，可修改参数已经在第 2 章列出，这里简述调试方法。

进入 Qgroundcontrol 软件，单击 Pixhawk on comX（根据实际串口选择）后，再单击 Connect 按钮，进入操作界面。

为了调试有好的效果，准备一张 60cm×60cm 大小的马赛克图案纸铺在地面。首先调整镜头焦距。单击 Video Downlink 按钮，可以显示出摄像头拍摄画面，画面清晰则说明焦距正确。摄像头得到的图像以及调试用的马赛克地面如图 5.12 所示。

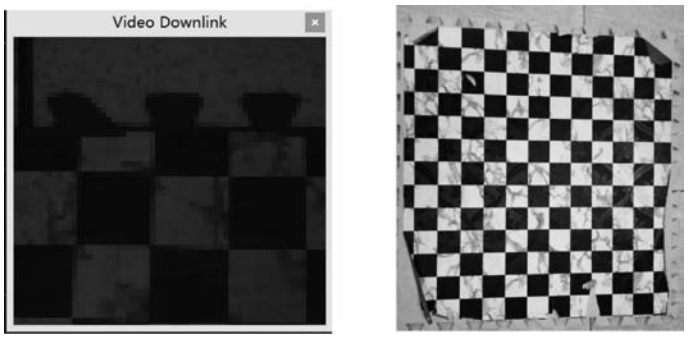


图 5.12　清晰的图像及调试用的马赛克地面

在马赛克地面上能更加准确地看到光流波形的变化，更便于调试，在普通地面可能效果并不理想，这与地面的材质、花纹、反光程度都有关系。调节参数是希望在普通地面也能达到与马赛克地面一样的效果，两种地面的实测效果图见图 5.13 和图 5.14。图中是 integrated_x(表示 x 方向上的位移量)在传感器上下移动时的波形。

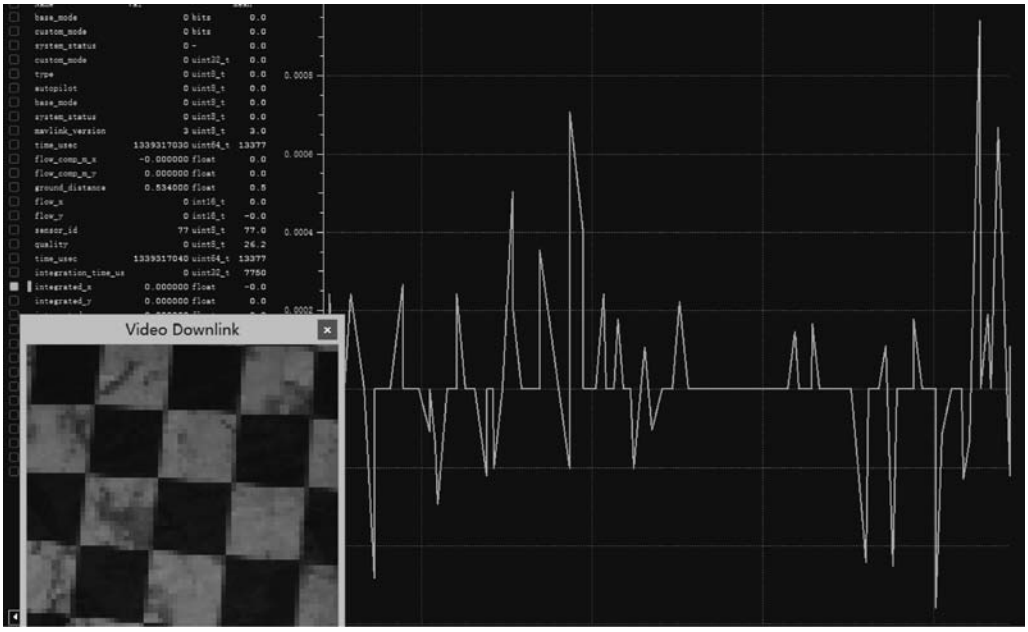


图 5.13　马赛克地面波形

在马赛克地面只有少量毛刺，普通地面抖动严重，并不理想。

经过测试发现，BFLOW_F_THLD、BFLOW_V_THLD 对波形影响较大，BFLOW_W_NEW 对波形影响较小，LENS_FOCAL_LEN 对陀螺仪输出波形有影响，其他参数对输出没有影响。

BFLOW_F_THLD 参数是一个特征阈值，定义了底层光流计算图形的质量。原值为 30，测试发现，把值调小至 15 后波形更为敏感，微小的移动偏量也会被探测到。

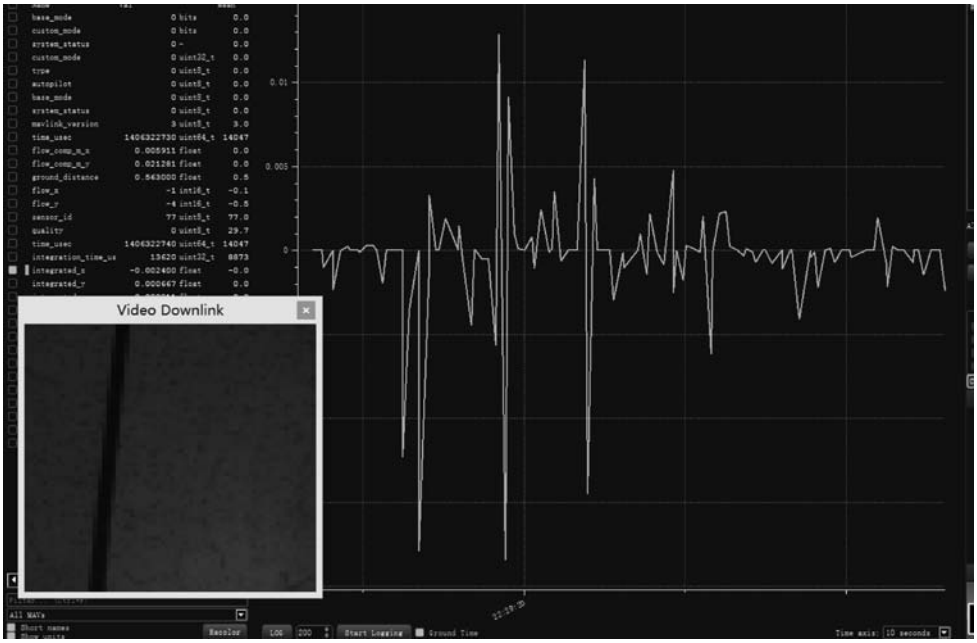


**图 5.14　普通地面波形**

BFLOW_V_THLD 参数是图形相关阈值，用来过滤较差的图形匹配。原值为 5000，基本所有图像都被用来计算，所以误差会很大，调低至 150，效果较为理想。

BFLOW_W_NEW 位光流低通滤波器增益，原值为 0.3，调至 0.1 有微小变化，基本影响不大。

LENS_FOCAL_LEN 为透镜焦距(mm)，此数值要与所使用镜头相对应。焦距对应时则陀螺仪输出波形在偏移时与 x 拟合理想，如图 5.15 所示。图中是 integrated_x 与 integrated_xgyro(表示 x 在三轴陀螺仪补偿后的位移量)，两条曲线的重合程度越好，表示光流参数越理想。

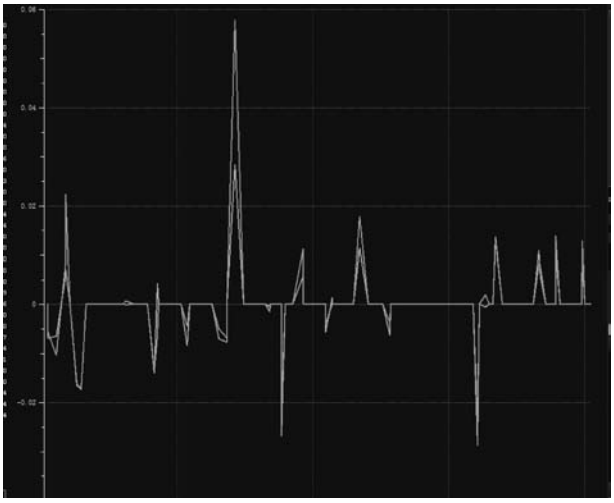

**图 5.15　xgyro 与 x 波形对比**

通过观察 x、y 的正负值关系，可以确定光流的 x 与 y 方向，见图 5.16。

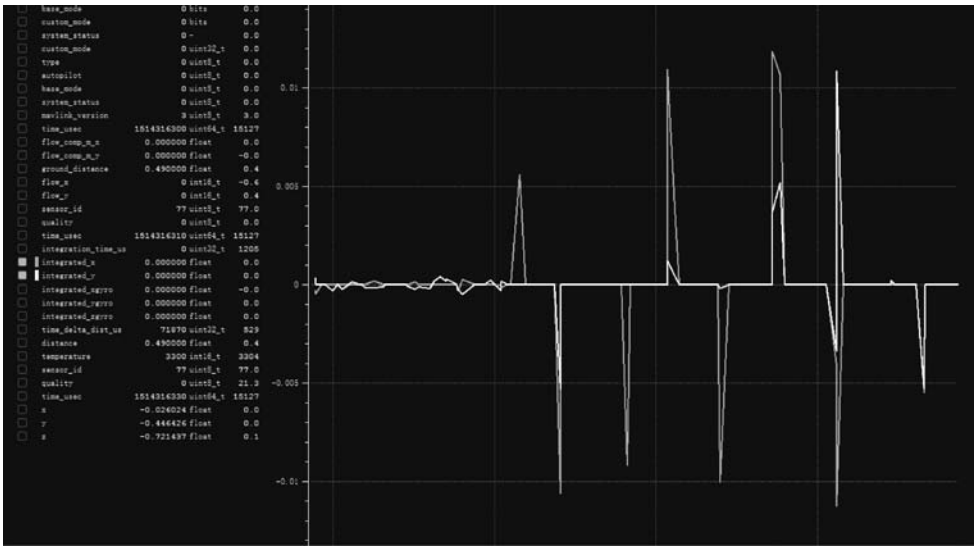

图 5.16　x 与 y 位移量波形图

值得注意的是，PX4Flow 对光照极为敏感。经测试发现，在夜晚、光照不好、日光灯照射下，光流工作状况并不理想，比较效果见图 5.17 和图 5.18。

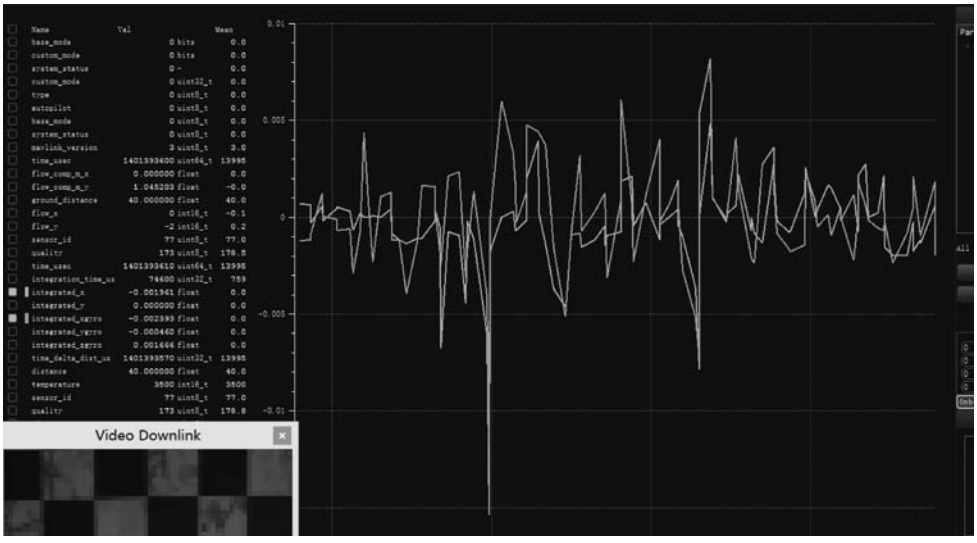


图 5.17　日光灯下光流波形有大量毛刺

光流在光照条件好的情况下，以 250Hz 速度处理图像，日光灯光照为 50Hz，对图像处理有极大干扰。所以光流应工作在明亮的室内环境，不可使用有频闪的日光灯。

试验表明，焦距相同镜头像素越高，x 与 y 的位移越精确，而不同焦距的镜头也对输出波形有影响。在不同高度，不同焦距的镜头在画面输出上也有很大的差别。通过观察输出波形，得到对应镜头焦距与高度的部分关系，如表 5.3 所示。

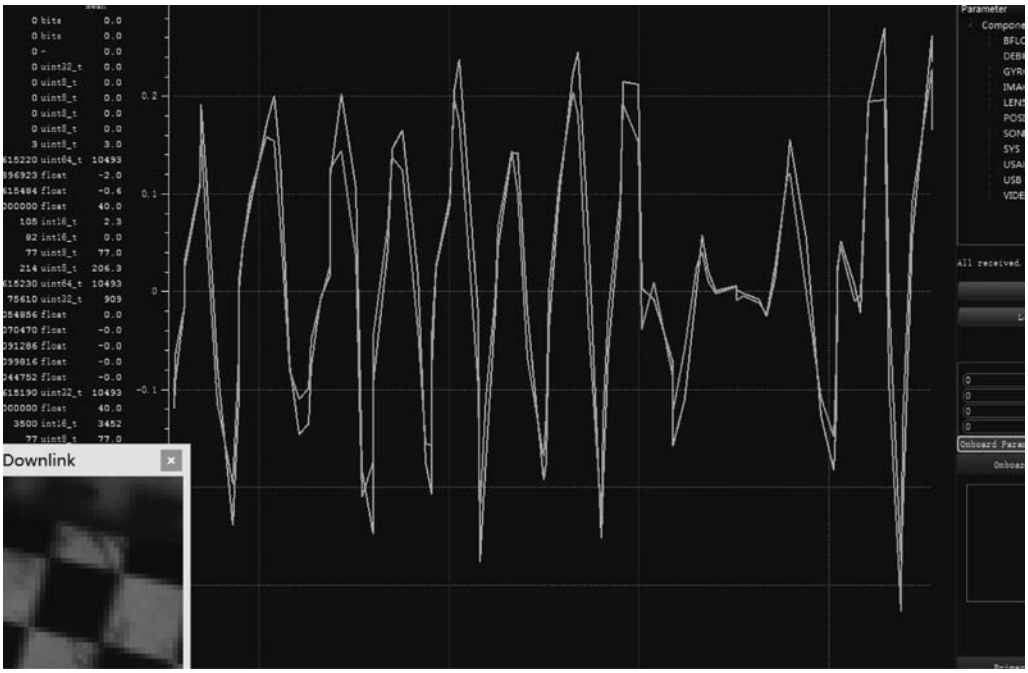


图 5.18 阳光下光流拟合波形

表 5.3 焦距与适用高度范围

| 焦　　距 | 适用高度范围 |
|---|---|
| 3.6mm | 50～60cm |
| 4.2mm | 80～90cm |
| 6mm | 100cm 以上 |

在调试完光流传感器后，就可以在地面站上读取光流数值来确认是否达到可以使用的程度。

在地面站上编写程序，通过串口可以读取到光流输出的数据包，然后将需要的数据显示到地面站的 LCD 屏上，地面站与光流连接示意图如图 5.19 所示。

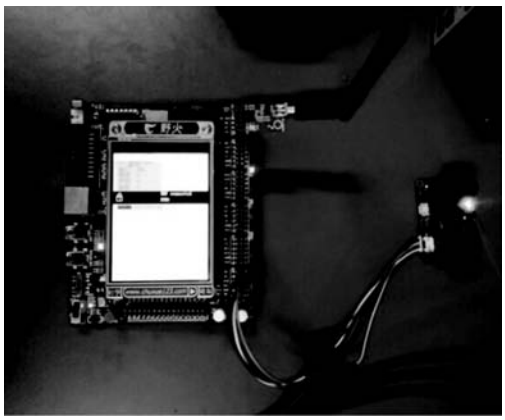

图 5.19 地面站与光流连接示意图

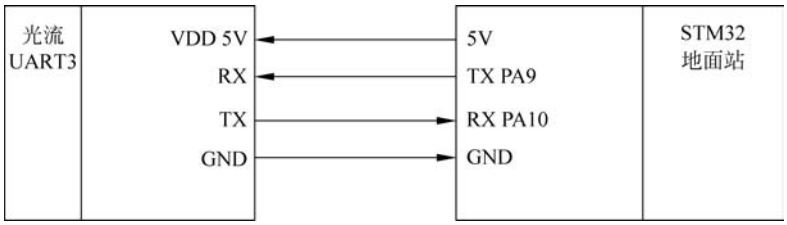


**图 5.19**　（续）

光流值显示数据值如图 5.20 所示。其中 YAW、height 对应光流的 x 位移量与 y 位移量，单位是像素。

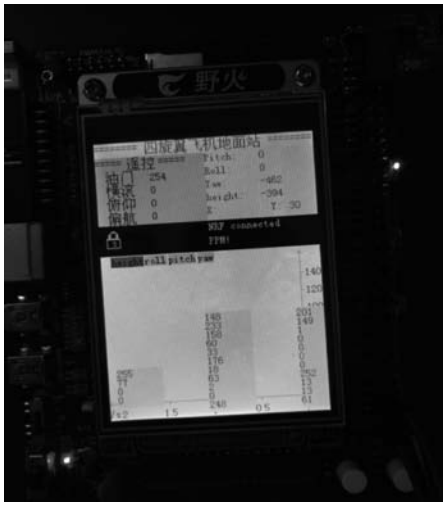

**图 5.20　光流显示数据值**

通过定义一个 26 元素的数组 flow_buf 来存放从光流发送的数据，数据对应的数组如表 5.4 所示。

**表 5.4　PX4 光流数据对应的数组**

| 数组编号 | 内　　容 | 数组编号 | 内　　容 |
|---|---|---|---|
| 0～7 | flow. time_sec | 20、21 | flow. flow_x. origin |
| 8～11 | flow. flow_comp_x. originf | 22、23 | flow. flow_y. origin |
| 12～15 | flow. flow_comp_y. originf | 24 | flow. id |
| 16～19 | flow. hight. originf | 25 | flow. quality |

同理，定义 flow_buf_rad 数组来取得陀螺仪矫正数据，只需要用到 x 与 y 的矫正数据。

flow_rad. integrated_x 和 flow_rad. integrated_y 指水平方向 x、y 位移的像素值，flow_rad. integrated_x 和 flow_rad. integrated_y 指陀螺仪偏转时 x、y 的位移像素值。在四旋翼飞行器悬停或位移时，输出 x、y 方向的位移值对飞行器进行修正。当四旋翼飞行器出现前倾、后仰等情况时，减去 flow_rad. integrated_x 和 flow_rad. integrated_y 可以使飞行器认为自己没有产生水平上的位移，以保证悬停的稳定。

通过地面站读取的数值是像素值，而不是实际的距离，所以还要对数据进行修正，使输

出到地面站的值为实际移动距离，同时修正后的位移值再通过 x 和 y 方向 PID 的调节，以实现飞行器的悬停。

在不同高度移动 20cm 像素输出差值如表 5.5 所示。

**表 5.5　不同高度移动 20cm 像素输出差值**

| 高度 60cm | 高度 80cm | 高度 1m |
|---|---|---|
| 280 | 200 | 156 |
| 275 | 200 | 152 |
| 278 | 226 | 150 |
| 260 | 225 | 170 |
| 290 | 210 | 170 |
| 270 | 220 | 151 |
| 270 | 200 | 190 |
| 取均值＝277 | 取均值＝208 | 取均值＝166 |

在不同高度移动 50cm 像素输出差值如表 5.6 所示。

**表 5.6　不同高度移动 50cm 像素输出差值**

| 高度 60cm | 高度 90cm |
|---|---|
| 703 | 463 |
| 704 | 460 |
| 710 | 455 |
| 670 | 460 |
| 700 | 463 |
| 710 | 461 |
| 690 | 464 |
| 取均值＝694 | 取均值＝462 |

由数据可知，高度越低，移动相同距离输出的值越大，这与实际情况相符合。

获得实验数据后，就可以开始进行高度与角度的修正了。

角度变化示意图见图 5.21，飞行器偏移会使其误认为自己产生了水平位移，需要矫正这一偏转误差。角度修正为：

```
x1 = flow_rad.integrated_x - flow_rad.integrated_xgyro;
y1 = flow_rad.integrated_y - flow_rad.integrated_ygyro;
```

陀螺仪输出值为偏转误差，只需要位移量减去这个值就可以使飞行器在原地偏转时输出的位移值不变。

不同高度移动相同距离示意图见图 5.22。飞行器越低，移动同一距离输出的值就会越大，称为高度误差。

高度修正程序如下：

```
x_mm = px4_sumy * High_Now * conv_factor;
      y_mm = px4_sumx * High_Now * conv_factor;
```

可知这是一个与高度相关的线性关系，只需要得到高度因数 conv_factor 即可，High_Now 是通过超声波数据得到的当前高度值。通过表 5.5 和表 5.6，运用 Excel 软件运算进行拟合，即可得到参数值 conv_factor ＝0.0012f，拟合公式见图 5.23。

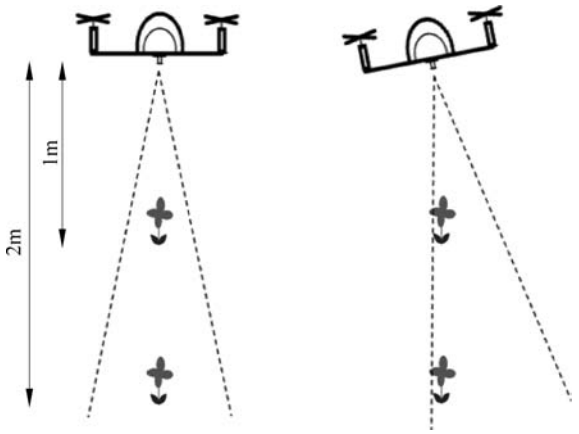


图 5.21　角度变化示意图

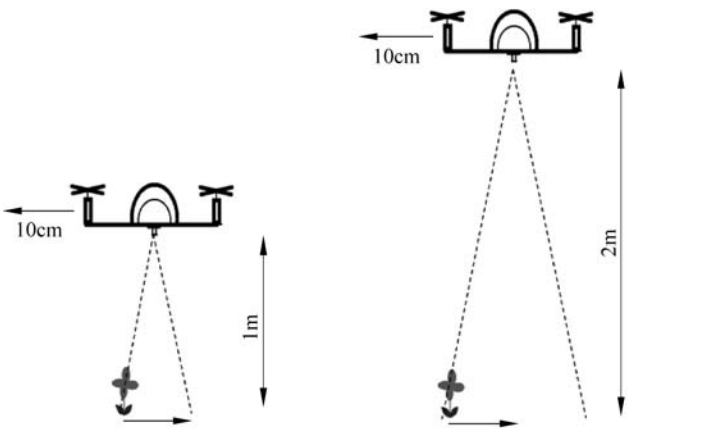


图 5.22　不同高度移动相同距离示意图

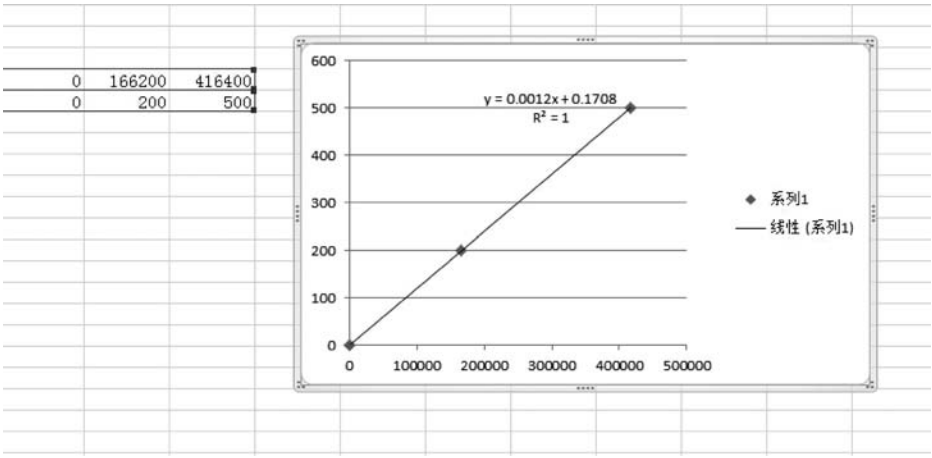


图 5.23　拟合公式

### 5.7.4 PX4Flow 光流与四旋翼飞行器的硬件连接

光流传感器有 4 个引脚需要连接到飞控板上，分别是 5V 电源输入、GND、TX、RX。光流 TX 与主控芯片 PD4（UART6-RX）相连，光流 RX 与主控芯片 PD5（UART6-TX）相连。

光流传感器与四旋翼飞行器安装方向示意图见图 5.24。通过地面站读取到的移动数值，可以确认飞行器向正方向飞行，光流输出值为负值，所以在进行飞行器悬停校正时，应该加上光流输出的位移量。光流传感器安装方向应与飞行器正方向平行，摄像头位置在飞行器尾部方向。

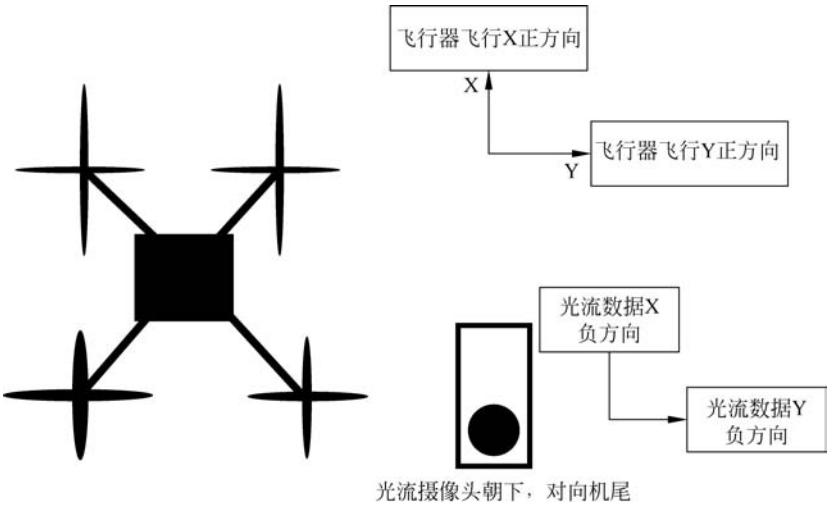


**图 5.24 光流传感器与四旋翼飞行器安装方向示意图**

### 5.7.5 PX4Flow 光流与四旋翼飞行器的软件连接

光流传感器能实现飞行器的定点悬停，软件流程图见图 5.25。光流传感器连接的是飞控板 UART6 串口，初始化时开启串口接收中断。

在地面站能够查看是否能正常接收到正确的光流数值，飞控程序根据姿态和高度将光流数值修正完毕得到实际的位移值后，通过 PID 调节就可以实现飞行器悬停。程序如下：

```
PID_POS_XOUT = (eeprom_readdate[3]/1000.0) * (tot_x_cm/10.0) +
(eeprom_readdate[5]/100.0) * (tot_x_cm - last_tot_x_cm)/10.0;
PID_POS_YOUT = (eeprom_readdate[6]/1000.0) * tot_y_cm/10.0 +
(eeprom_readdate[8]/100.0)  * (tot_y_cm - last_tot_y_cm)/10.0;
```

将其转换为数学公式：

控制输出量＝P×当前方向偏移量＋D×（当前方向偏移量－上一次偏移量）

eeprom_readdate[3]和 eeprom_readdate[6]是 x 方向上的 P 值和 D 值，eeprom_readdate[5]和 eeprom_readdate[8]是 y 方向上的 P 值和 D 值。由于四旋翼飞行器的对称关系，x 与 y 方向的 P 值是相同的，P＝0.035，D 值也是相同的，D＝3.8。

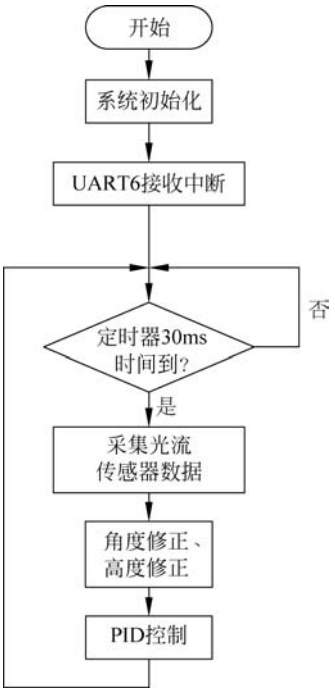


**图 5.25　定点悬浮流程图**

数值除以 1000、100 是因为上位机传输到飞控的 P、D 值是整形，这里要转换为小数。同样位移偏移量的单位也要转换。

定点悬停的输出量 PID_POS_X.OUT 和 PID_POS_Y.OUT 控制的是飞行器的 U2（翻滚输入控制量）、U3（俯仰控制量），使光流输出的位移值趋于 0，则飞行器能够保持飞行在原点附近，实现定点悬停的功能。

最后输出给电机的控制值为 PID 调节后的六轴传感器姿态值、光流数据、高度数据以及遥控器输入值之和。姿态、高度和光流的 PID 调节参数如图 5.26 所示。

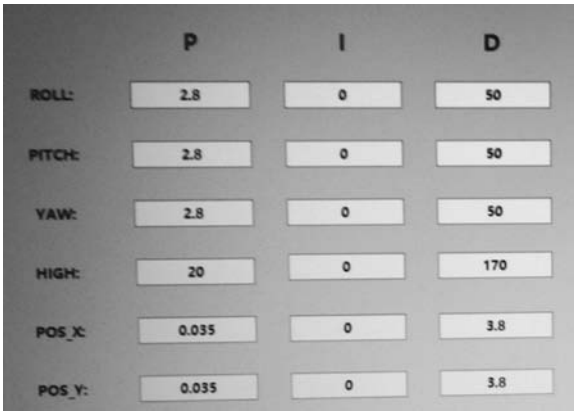


**图 5.26　姿态、高度和光流 PID 值**

## 5.8 飞行器及图像传感器调试

整个系统的调试需要软件部分与硬件部分联合进行调试。下面介绍几个在调试中主要出现的问题及对应的解决方法。

(1) 问题描述：在软件之前已调试成熟的情况下，新组装的四旋翼飞行器在下载了软件后进行试飞时产生自旋，或轻推微小油门飞行器会翻身，或偏航严重。

解决方案：

① 首先确定电机转向正确，螺旋桨安装正确。飞控板安装方向确定了飞行器的正方向。电机的编号 m1、m2、m3、m4 确定了其正反转方向。

② 飞控板拆卸后再次安装的位置是否对齐飞行器正方向。当改变飞控板安装方向时，就意味着对应的 m1、m2、m3、m4 电机改变位置，则其电机正反转就会发生错误。

③ 如果上电时电调持续鸣叫，证明电调损坏，需要更换。更换后需要重新进行初始化才能正常工作。

④ 电机无桨空转时声音异常，或手动旋转电机轴不灵活，则电机需要更换。

(2) 问题描述：飞行器在起飞时向左或者向右自旋一定角度。

解决方案：这可能是因为某些电机没有垂直安装或者电调没有初始化校准成功。

(3) 问题描述：飞行器起飞时总是朝着某一个方向偏移。

解决方案：如果只是起飞时歪，飞至空中后悬停能自动调整稳定，那应该是飞行器重心不对或起飞地面不平的原因。可通过电池的安装位置调整重心，还可以在飞控姿态程序中在倾斜方向上减去能使飞行器水平的偏移量。

### 5.8.1 图像传输

可以实现视频传输的方式很多，比如使用 ARM 板、DSP 平台进行编程，通过图传模块或 WiFi 模块来传输图像。考虑到要将航拍功能应用于载荷有限的四旋翼飞行器上，应选用体积较小的、空间利用率高的 WiFi 发射装置。

本方案采用 WR703N 无线路由器作为 WiFi 信号发射器。它不仅体积小，集成度高，利于安装固定，而且在软件部分采用了开源的 OpenWrt，程序编写灵活简单。实物图如图 5.27 所示。其 USB 接口连接摄像头，串口通信三根导线接头连接飞控板。

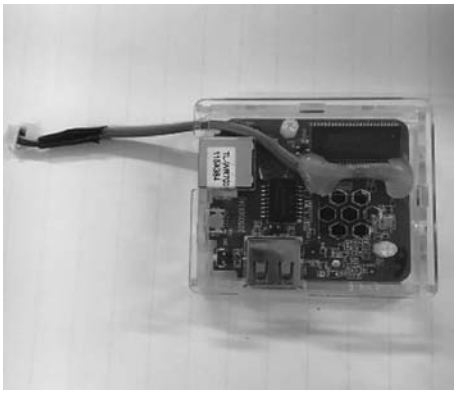

图 5.27 WR703N 路由器

### 5.8.2　OpenWrt 的使用

1. OpenWrt 固件刷写

刚买来的 WR703N 无线路由器并不能直接使用，要对其进行 OpenWrt 固件的刷写。

(1) WR703N 路由器连接计算机，在计算机上找到本地连接属性对话框，对“Internet 协议版本 4(TCP/IPv4)”进行正确的配置，如图 5.28 和图 5.29 所示。


图 5.28　“本地连接 属性”界面

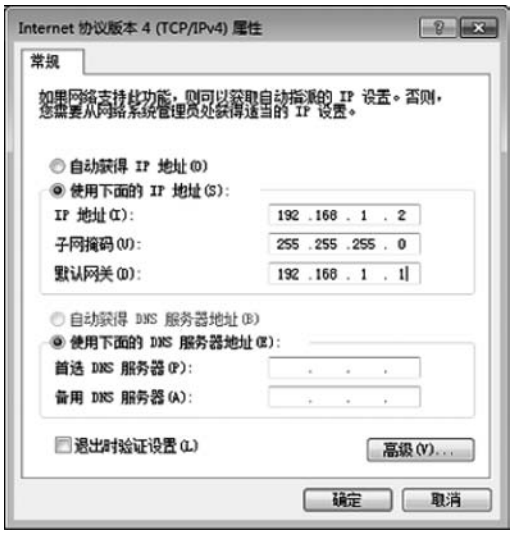


图 5.29　Internet 协议属性界面

(2) 将烧写的固件放入 tftp32 文件夹内，如图 5.30 所示。tftp32 相当于一个多功能的网络服务器包，主要用于不同服务器之间的资源传输。

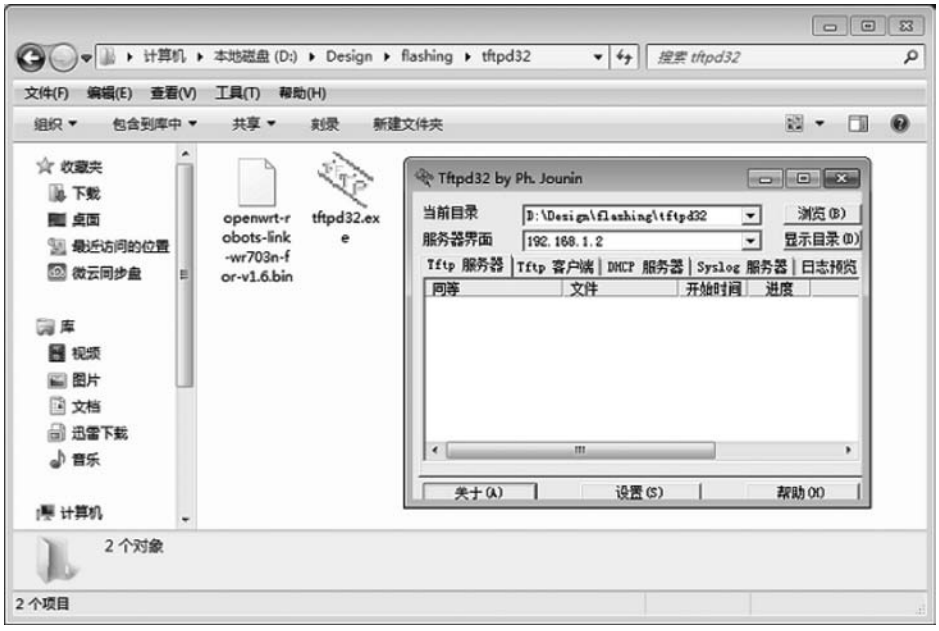


图 5.30　放置固件到目的路径

(3) 在计算机与路由器之间用网线和 TTL 线连接，登录 PuTTY，如图 5.31 所示。PuTTY 是一款在 Windows 平台下访问 Linux 系统的软件。

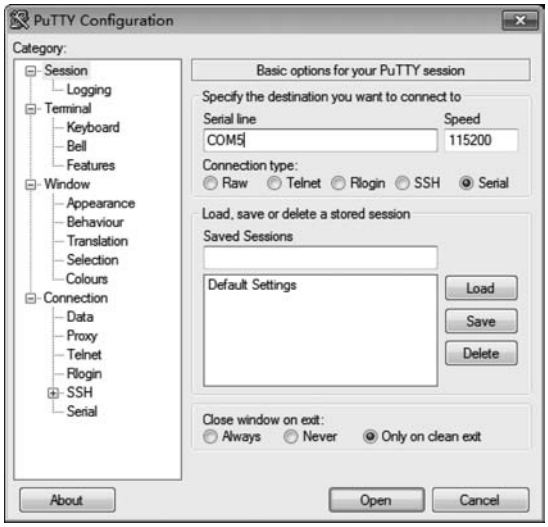


**图 5.31　使用 PuTTY 登录**

(4) 对系统进行适当的配置即可完成固件刷写，路由器的启动页面如图 5.32 所示。软件 SecureCRT 功能与 PuTTY 相似。

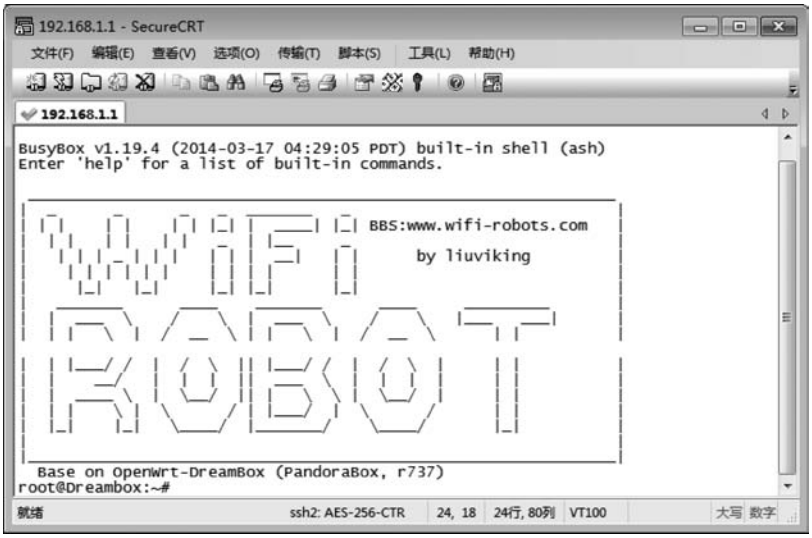


**图 5.32　OpenWrt 启动画面**

OpenWrt 是一个集成化的 Linux 系统，在功能上有很好的扩展性，可用于 U 盘、串口透传、智能家居、中继转发等。

2. 备份安装 bin 文件

(1) 用 PuTTY 进入路由以后，输入 cat /proc/mtd 查看分区状况。里面说明了各个分区的内容，如图 5.33 所示。

```
root@Dreambox:~# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00020000 00010000 "u-boot"
mtd1: 000e1bd4 00010000 "kernel"
mtd2: 002ee42c 00010000 "rootfs"
mtd3: 000b0000 00010000 "rootfs_data"
mtd4: 00010000 00010000 "art"
mtd5: 003d0000 00010000 "firmware"
```

**图 5.33　分区情况**

firmware 分区包含了所有的 kernel、rootfs、rootfs_data 和 art，如表 5.7 所示。

**表 5.7　分区情况说明**

| 分区名称 | 描　　述 |
|---|---|
| u-boot | 设备初始化程序＋引导程序代码本身 |
| kernel | 内核，顾名思义是一个系统的最核心部分，存放的都是对内核进行管理的核心信息，这些信息都是以二进制形式存放的 |
| rootfs | 完整的系统文件包含只读和可写 |
| rootfs_data | 在 rootfs 中的可写部分的位置，为配置文件所在区，rootfs_data 相当于/overlay |
| art | EEPROM 分区，在 Atheros 的方案中这个分区保存了无线的硬件参数 |
| firmware | 完整的固件位置包含了除 u-boot 和 art 之外全部的内容 |

（2）输入以下几行命令就可以备份整个.bin 格式的 back_firmware 文件了，如图 5.34 所示。

```
root@Dreambox:～# cd /tmp
root@Dreambox:/tmp# dd if=/dev/mtd5 of=/tmp/back_firmware.bin
7808+0 records in
7808+0 records out
```

```
root@Dreambox:/tmp# ls
TZ                  hosts           resolv.conf       sysinfo
back_firmware.bin   lock            resolv.conf.auto
dhcp.leases         log             run
etc                 overlay         state
```

**图 5.34　文件备份目录**

（3）通过 WinSCP 传到 Windows 系统，如图 5.35 所示。WinSCP 是不同系统之间文件传输处理的工具。


**图 5.35　备份文件显示**

3. 编译一个 Hello World 程序

（1）在 Ubuntu 下，新建名为 hello_world.c 文件，编写相应的程序。

```
vi hello_world.c
```

（2）输入以下内容：

```
#include<stdio.h>
int main(char argc,char *argv[])
{
```

```
        int i = 1;
        while(1){
                        …
                printf("Hello world!!! %d\n",i);
                if(i < 10)
                        i++;
                else
                        i = 1;
                sleep(1);
        }
        return 0;
}
```

(3) 用 mips 指令编译“hello_world. c”文件,并生成“hello_world”可执行文件 mipsopenwrt-linux-gcc hello_world. c-o hello_world。

(4) 用 WinSCP 将文件传输到/tmp 目录下。

(5) 在开发板的串口终端下执行 hello_world 文件。

```
/tmp/hello_world
```

4. 交叉编译并生成 IPK 文件

交叉编译通常用于两个有不同控制语言的平台之间,先在 A 机器上对程序进行编译,生成可执行文件,再将执行文件转移到 B 机器上来运行。起初要从官网上下载交叉编译工具链并进行安装,工具链通常包括编译器、连接器、解释器和调试器四部分。

(1) 切换到 openwrt 根目录,执行下列命令。

```
cd./package/utils          //进入 package/utils 目录
mkdir hello_world          //创建一个名为"hello_world"的文件夹,用于放置安装包源码
cd hello_world             //进入相应文件夹
mkdir src                  //新建一个名为"src"的文件夹
vi src/hello_world.c       //在 src 目录下新建一个名为 hello_world.c 文件
```

(2) 在 src 目录下新建一个 Makefile,输入相应的内容。

```
vi src/Makefile
```

(3) 在当前目录下新建一个 Makefile 文件并进行编写。

Makefile 如同脚本文件,其中定义编写了一系列系统需要执行的操作命令。

(4) 返回至 openwrt 文件夹,执行命令:

```
make menuconfig
```

选择已经加进去的安装包:

```
utilities ---->
<M> hello_world.....Hello world - prints a hello world message
```

保存并退出。

(5) make V=s。等待编译完成后,可以在 openwrt 目录下的 base 文件夹内找到生成的安装包/hello_world_1.0_ar71xx.ipk。

(6) 将 ipk 包传输与安装到开发板上即可运行。更改权限 root@Dreambox:/# chmod u+x /tmp/hello_world,运行结果如图 5.36 所示。

```
root@Dreambox:/#  /tmp/hello_world
Hello world!!!1
Hello world!!!2
Hello world!!!3
Hello world!!!4
Hello world!!!5
Hello world!!!6
```

**图 5.36　Hello World 运行结果**

5. OpenWrt 高级功能

除了基本的 Hello World 程序的运行外,OpenWrt 这个嵌入式 Linux 系统还有很多高级功能可以拓展,以下介绍了其中的 3 种模式。

(1) AP 模式。接入点模式,可以不用设置 SSID 和密码,关闭 DHCP Server,外设通过 WiFi 连接开发板。

(2) Router。路由器模式,输入 SSID 和密码才可以进入,开启 DHCP Server,外设通过 WiFi 连接开发板。

(3) 二级无线路由器模式设置。开启 DHCP Server,连接开发板的设备和上级路由器处于不同网络,一般不能互通。

6. OpenWrt 挂载摄像头

(1) 要实现视频传输必须安装相关功能的 IPK,如表 5.8 所示。

**表 5.8　视频传输相关 IPK 介绍**

| IPK 名称 | 描　　述 |
|---|---|
| kmod-video-core | 摄像头内核模块,UVC 驱动依赖包 |
| kmod-video-videobuf2 | UVC 驱动依赖包 |
| kmod-video-uvc | UVC 驱动 |
| libpthread | mjpeg-streamer 依赖包 |
| libjpeg | mjpeg-streamer 依赖包 |
| mjpg-streamer | mjpeg-streamer 功能安装包 |

(2) 插入摄像头。根据系统打印信息判断开发板是否支持该型号的摄像头,必须选择 UVC 摄像头。在安装完摄像头之后会出现/etc/config/mjpg-streamer 配置文件,如图 5.37 所示。

```
config mjpg-streamer core
        option enabled          "0"
        option device           "/dev/video0"
        option resolution       "640x480"
        option fps              "5"
        option www              "/www/webcam"
        option port             "8080"
```

**图 5.37　mjpg-streamer 文件内容**

文件中的内容分别配置了设备名称、摄像头的分辨率、帧率、访问目录与访问端口。

(3) 在连接上路由器发射的 WiFi 信号之后,用浏览器打开网址 http://192.168.1.1:8080/? action=stream 即可看到实时传输的视频,效果如图 5.38 所示。

图 5.38 视频传输效果

## 5.9 电源电压测量

由于锂电池连续使用或长期静置会产生电池过放的现象，这也就意味着电池报废，因而对电池电量的实时监控是十分有必要的。一般情况下，每节锂电池充满为 4.2V，而将其报警电压设置为 3.7V(即需要充电电压)。

如果采用带有报警功能的 1-8S 高精度电量显示器，飞行器飞行时插在电池上实时检测电池电量，其电压检测精度为±0.03V，电量显示于数码管上，需要几十毫安的驱动电流，低电量时会启动蜂鸣器进行报警，整个电量显示器的质量约为 9g。相比而言，采用飞控 AD 口设计的分压电路进行电池电压检测，锂电池的耗电量远小于电量显示器，质量小，而且电路简单，生产成本很低。

### 5.9.1 模数转换器概述

模数转换器的功能是将输入的连续模拟电压值转换成离散数字量。它的参考电平为 3.3V，也就是说当输入源输入电压为 3.3V 时，12 位模数采样值为 4095。另外，每一个 ADC 模块都包含了 4 个可以对输入源触发、捕获、中断等进行灵活配置编程的序列发生器。

对于 4 个采样序列发生器，第一个序列发生器能够捕获八路采样，第二、三个序列发生器能够进行四路采样，而第四个序列发生器则只能捕获一路采样。采样序列发生器还可以通过配置其相应的先后顺序来应对多个响应同时触发的情况，优先级最高的采样序列发生器首先被触发，优先级第二的采样序列发生器接着被触发，以此类推。

### 5.9.2 电源电压测量的实现

1. 电源分压电路的设计

由于 3S 锂电池充满时电压为 12.6V，而飞行控制板上 ADC 模块的参考电压为 3.3V，因此需要设计基本的分压电路进行电压转换。考虑到飞控板的安全问题与电路的耗电量问题，选用了 3.6kΩ、15kΩ 的电阻和正向管压降为 0.7V 的硅二极管各一个，如图 5.39 所示。

由图可得，电压输出值为

$$U_{out}=\frac{V_{CC}-U_{CE}}{R_1+R_2}\times R_2+U_{CE} \tag{5.1}$$

电路设计中，若 $R_1$、$R_2$ 分压电阻的阻值过大，则会导致电路中的电流过小，虽然功耗大大减小了，但这样微小的电流却不足以驱动 ADC 模块正常工作；若阻值选择太小，则使得电路中电流过大，功耗过大。

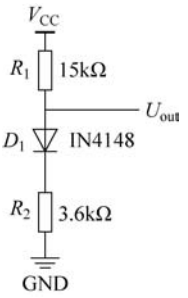


**图 5.39　电源电压测量电路**

2. ADC 采样的软件实现

ADC 的 API 封装了一系列处理 ADC 功能的函数，主要包含了 3 类：用于处理采样序列发生器；用于处理触发器和处理器；用于处理中断。

在本电路中，采用了 PE0 作为 AD 的采样引脚，其基本配置参数如表 5.9 所示。

**表 5.9　PE0 引脚参数**

| 引脚名称 | 引脚编号 | 引脚赋值 | 引脚类型 | 缓冲区类型 | 描　　述 |
|---|---|---|---|---|---|
| AIN3 | 9 | PE0 | I | 模拟 | 模数转换器输入 3 |

根据 PE0 引脚的参数来进行相应的初始化，具体程序如下：

```
void battery_init()
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC1);              //使能 ADC0
    SysCtlDelay(3);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);             //使能 PE0
    CPIOPinTypeADC(GPI0_PORTE_BASE,CPI0_PIN_0);              //使能 PE0 为 ADC 功能
    ROM_ADCReferenceSet(ADC1_BASE,ADC_REF_INT);
    //配置 ADC1,采样序列 3,ADC 处理器触发,优先级为 0
    ROM_ADCSequenceConfigure(ADC1_BASE,3,ADC_TRIGGER_PROCESSOR,0);
    //配置采样序列步进,ADC 通道 3,中断使能,对列结束选择
    ROM_ADCSequenceStepConfigure(ADC1_BASE,3,0,ADC_CTL_CH3|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC1_BASE,3);                          //使能 ADC 采样
    ADCIntClear(ADC1_BASE,3);                                //ADC1 中断清除
}
```

而对于读取数据，需要将 AD 数组中的值取出并赋给目标值，程序如下：

```
ADCIntClear(ADC1_BASE,3);                                    //ADC 中断清除
ADCProcessorTrigger(ADC1_BASE,3);                            //中断发生器触发 ADC
while(!ADCIntStatus(ADC1_BASE,3,false));                     //等待数据采样完成
ADCSequenceDataGet(ADC1_BASE,3,pui32ADC0Value);              //采集第一个像素点
SysCtlDelay(10);
battery_temp = pui32ADC0Value[0];                            //将 AD 数组中的值取出,赋值
```

### 5.9.3　电源分压模块的测试

测试阶段，进入 Debug 模式，通过单步调试将读取的 AD 值转换成理论电压值，公式为

$$U_{测量}=\frac{3.3\times(T_{emp}+1)}{4096} \tag{5.2}$$

其中 $T_{emp}$ 为单片机内读取的 12 位 AD 值，然后将其与实际值进行对比，结果如表 5.10 所示。

**表 5.10 采样值与实际值对比**

| $T_{emp}$ | $U_{测量}$ | $U_{out}$ |
|---|---|---|
| 2705 | 2.17 | 2.2 |
| 2785 | 2.24 | 2.3 |
| 2890 | 2.32 | 2.4 |
| 3045 | 2.45 | 2.5 |
| 3165 | 2.55 | 2.6 |
| 3282 | 2.64 | 2.7 |
| 3435 | 2.76 | 2.8 |
| 3560 | 2.86 | 2.9 |
| 3700 | 2.98 | 3.0 |

最后，由式(5.1)反推即可得到锂电池的实际电压值。

## 5.10 舵机

舵机是一个微型伺服功能的控制系统，可在 0°～180°内转动，具有安装灵活、结构稳固、便于控制、稳定性好等优点，适用于需要保持或变化角度值的驱动场合中。随着飞行器的日益发展，带有航拍功能的飞行器越来越受欢迎，而云台是用于置放免驱摄像头的平台。对于航拍这项功能，一个摄像头的拍摄范围是有限的，但通过云台使得镜头可以自由转动，从而拍摄观测到各个角度方向的影像。

### 5.10.1 舵机云台的硬件搭建

按照舵机说明书的介绍分步组装舵机云台，安装完成的实物见图 5.40。其中红色线代表 $V_{CC}$，咖啡色线代表 GND，橘色线代表信号输入端。

**图 5.40 舵机云台实物图**

### 5.10.2　舵机云台的软件编写

根据9G舵机的基本参数得知，要想实现单片机对舵机云台旋转位置的控制，首先需要产生周期为20ms的PWM信号；其次需要改变PWM信号的占空比，应使得脉冲宽度变化范围为0.5～2.5ms，对应了云台转动位置的0°～180°。由于飞行控制TM4C123GH6PM芯片自带PWM信号发生功能，因此只需对其进行基本配置即可。

在本方案中，采用了PB6、PB7作为上、下两个舵机的信号发生引脚，其基本配置参数如表5.11所示。

**表5.11　PB6、PB7引脚参数**

| 引脚名称 | 引脚编号 | 引脚赋值 | 引脚类型 | 缓冲区类型 | 描　　述 |
|---|---|---|---|---|---|
| MOPWM0 | 1 | PB6 | 输出 | TTL | 运行控制模式0，信号由PWM发生器0产生 |
| MOPWM1 | 4 | PB7 | 输出 | TTL | 运行控制模式1，信号由PWM发生器0产生 |

根据PB6、PB7引脚的参数进行相应的初始化，具体程序如下：

```
void camera_init()
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);            //使能 PWM0 模块
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);           //使能 PWM0 和 PWM1 输出所在 GPIO
    MAP_GPIOPinConfigure(GPIO_PB6_MOPWM0|GPIO_PB7_MOPWM1);
    MAP_GPIOPinTypePWM(GPIO_PORTB_BASE,GPIO_PIN_6|GPIO_PIN_7);
    MAP_GPIOPinTypePWM(GPIO_PORTB_BASE,);
    SysCtlPWMClockSet(SYSCTL_PWMDIV_32);                   //PWM 时钟配置：32 分频
    //配置 PWM 发生器 0：加减计数，不同步
    PWMGenConfigure(PWM0_BASE,PWM_GEN_0,PWM_GEN_MODE_UP_DOWN|PWM_GEN_MOOE_NO_SYNC);
    //设置 PWM 发生器 0 的频率，时钟频率/PWM 分频数/n，80M/32/50000 = 50Hz
    PWMGenPeriodSet(PWM0_BASE,PWM_GEN_0,50000);
    PWMPulseWidthSet(PWM0_BASE,PWM_OUT_0,step_top);
    PWMPulseWidthSet(PWM0_BASE,PWM_OUT_1,step_down);
    PWMOutputState(PWM0_BASE,(PWM_OUT_1_BIT|PWM_OUT_0_BIT),true);
    PWMGenEnable(PWM0_BASE,PWM_GEN_0);
}
```

由于系统的主频为80MHz，因而经过32分频和50000次时钟计数后，PWM方波的频率为80MHz/32/50000 ＝50Hz，即20ms。而变量step_top与step_down则用于控制PWM信号的脉冲宽度，取值为1250～6250，对应0.5～2.5ms。为了使云台在手机控制下有较明显的转动，将步进值取为250，即每次正向或反向旋转9°。

引脚输出波形如图5.41所示。


图 5.41 PWM 波形输出

## 5.11 四旋翼飞行器的遥控实现

### 5.11.1 手机终端控制需求分析

1. 系统功能需求

四旋翼飞行器的手机终端控制的目的就是用身边现有携带的智能设备与四旋翼飞行器建立稳定可靠的连接,控制飞行器的飞行并实现双向通信,所以功能需求包含以下内容。

(1) 控制四旋翼飞行器的飞行。

(2) 实时显示电压余量。

(3) 实时图像显示。

2. 系统性能需求

(1) 实时性。

(2) 可靠性。

(3) 准确性。

### 5.11.2 手机终端控制软件的设计方案

1. 设计原则

充分考虑到使用者的需求以及系统本身的特点,再结合软件的系统质量,该软件的设计应该遵循以下 5 个原则。

(1) 实用性。

(2) 容易使用。

(3) 界面的一致性。

(4) 可移植。

(5) 封装性。

手机 APP 软件编程环境采用 Eclipse,编程语言用 Java。开始编程前还需安装 JDK 以及配置环境变量。

2. 功能模块的设计

手机终端系统主要是利用飞行器上的 WiFi 路由设备对电池余量进行接收，操控飞行器平稳飞行，按照 Socket 通信将读取到的电压余量显示在界面上，并且当电量低于某一门限时进行报警。因此根据整个系统的需求分析，可以把系统分成 3 个功能模块，如图 5.42 所示。

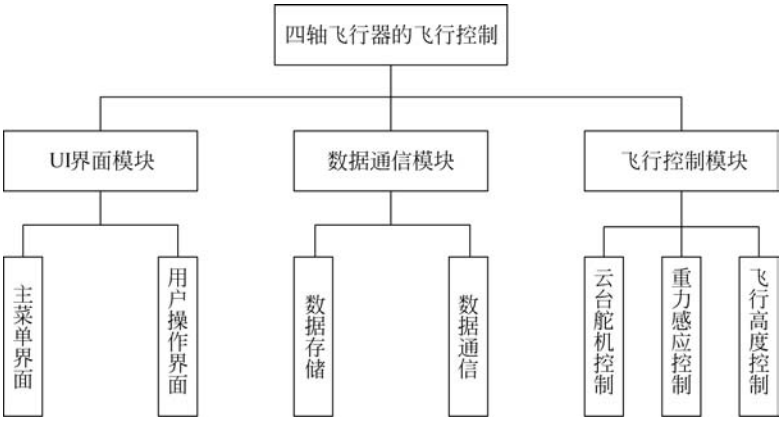


**图 5.42　手机功能模块结构图**

1）UI 界面模块

此模块是手机终端系统的人机接口界面，其中数据显示界面主要提供飞行高度、重力感应 3 轴数据以及云台拍摄的实时画面，用户操控界面主要有操控云台舵机上、下、左、右以及飞行高度控制。

2）数据通信模块

此模块分为数据存储与数据通信两部分。数据存储就是将上下行的数据保存在数据库里面，并实时地显示在界面上，包括飞行的高度、电池余量等。

（1）基于 TCP 协议的 Socket。

Socket 通信模型见图 5.43。

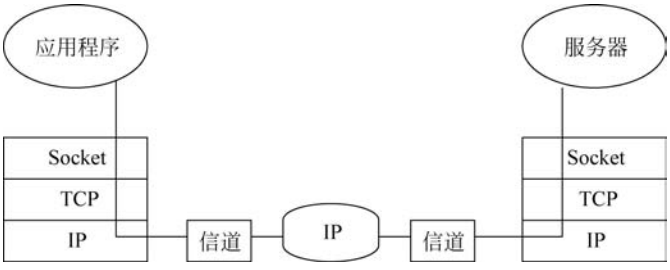


**图 5.43　Socket 通信模型**

（2）Socket 建立连接的过程。

Socket 又称套接字，在程序内部提供了与外界通信的端口，即端口通信。

Socket 建立连接主要分为以下 3 个步骤。

① 服务器(Server)监听。Server 创建 Socket，将它绑定到一个 IP 和 Port 上，并设置为

监听的模式，这时 Server 会实时地监控网络状态，等待 Client 的连接请求。TCP 协议通信见图 5.44。

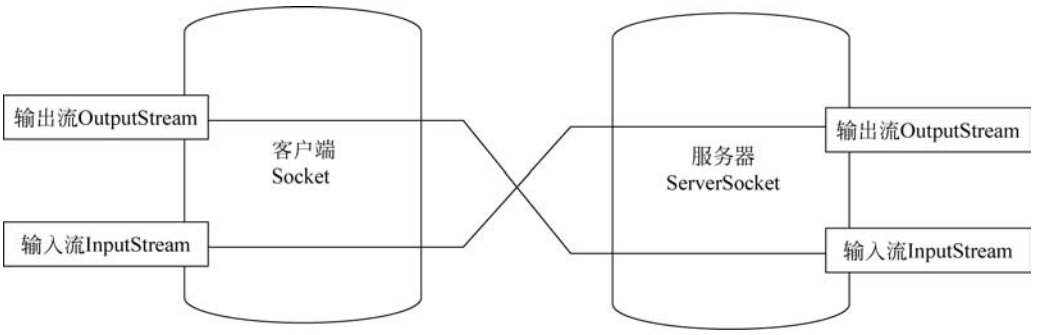


**图 5.44　TCP 协议通信**

② 客户端(Client)请求。Client 创建 Socket，并对 Server 发送连接请求，为了连接上 Server 端 Socket，Client 需要对连接的目标 Server 的 Socket 进行准确叙述，包括 Server 的 IP 和 Port，这是 Client 向 Server 发送请求的前提。

③ 两者确认连接并通信。处于监听状态的 Server 的 Socket 监听到 Client 的连接请求时，会对其做出响应并返回一个新的 Socket 给 Client 表示已接收请求，随着 Client 对此确认，两者至此就建立起连接进行通信。另外，Server 的 Socket 仍然处于监听的状态来响应其他 Client 的请求。

在四旋翼飞行器的飞控中，涉及四旋翼飞行器的飞控板与手机客户端之间的通信。总体来说，采用了 WiFi 的通信方式，将一个 WiFi 路由器作为一个站点，一方面可以接收飞控板串口发送过来的数据，并转发给到该路由器的手机客户端。另一方面，路由器还可以接收手机发送过来的数据并发送给四旋翼飞行器的飞控板，于是四旋翼飞行器的飞控板和手机客户端便建立了一个全双工通信的网络。

通信数据的发送和接收采用 TCP/IP 面向连接的传输协议。通信具体内容相互传送采用 ASCII 编码。

四旋翼飞行器的飞控板传送数据给手机客户端时：首先向手机端发送一个字符，这个字符可以让手机端知道传送过来的数值表示电池余量检测的数值。比如发送“D”，则表示发送的是电量的数值，最后需要发送一个换行符'\n'表示电压余量检测数值发送完毕。

四旋翼飞行器的飞控板收到手机端传送操控四旋翼飞行器摄像头的四种运动方向，数据传输协议为：当单片机串口接收到字符串'_CMDUa'时，表示云台向上；接收到'_CMDUb'时，表示云台向下；接收到'_CMDUc'时，表示云台向左；接收到'_CMDUd'时，表示云台向右。此外手机与网络摄像头之间的数据(视频)传输，采用的是 HTTP 协议。

(3) HTTP 视频传输协议。

采用 HTTP 协议作为视频传输的协议栈，主要分成网络层、传输层和应用层，如图 5.45 所示。

应用层是面向对象的协议，主要特点如下。

① 支持 Client/Service 模式。

② 通信的速度十分迅速。

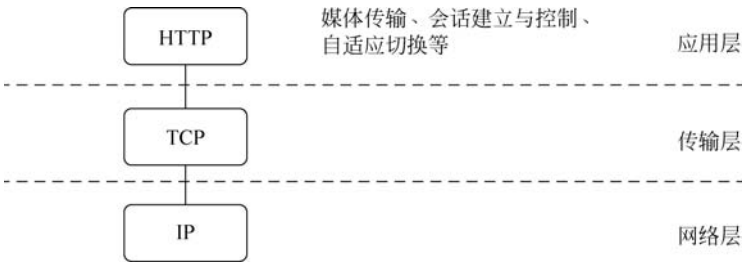


**图 5.45　HTTP 作为视频传输的协议栈**

③ HTTP 能够传输任意类型的数据对象，因此十分灵活。

④ 无须连接：HTTP 限制每一次的连接都只处理一个请求，利用这种方式可以节省传输时间。

3）飞行控制模块

利用手机终端向四旋翼飞行器发送云台的控制指令以及调整飞行高度等参数，及时调整四旋翼飞行器的飞行状态。另外，使用 Android 移动终端重力加速度进行重力感应遥控。

(1) Android 开发相关技术。

Android 是 Google 公司开发的基于 Linux 平台的开源手机操作系统，其包括操作系统、User 界面和 Applications，具有手机工作需要的全部功能。

Android 的系统架构与其操作系统一样，采用了分层的架构。Android 分成 4 层：应用程序层、应用程序框架层、系统运行库层和 Linux 核心层，如图 5.46 所示。

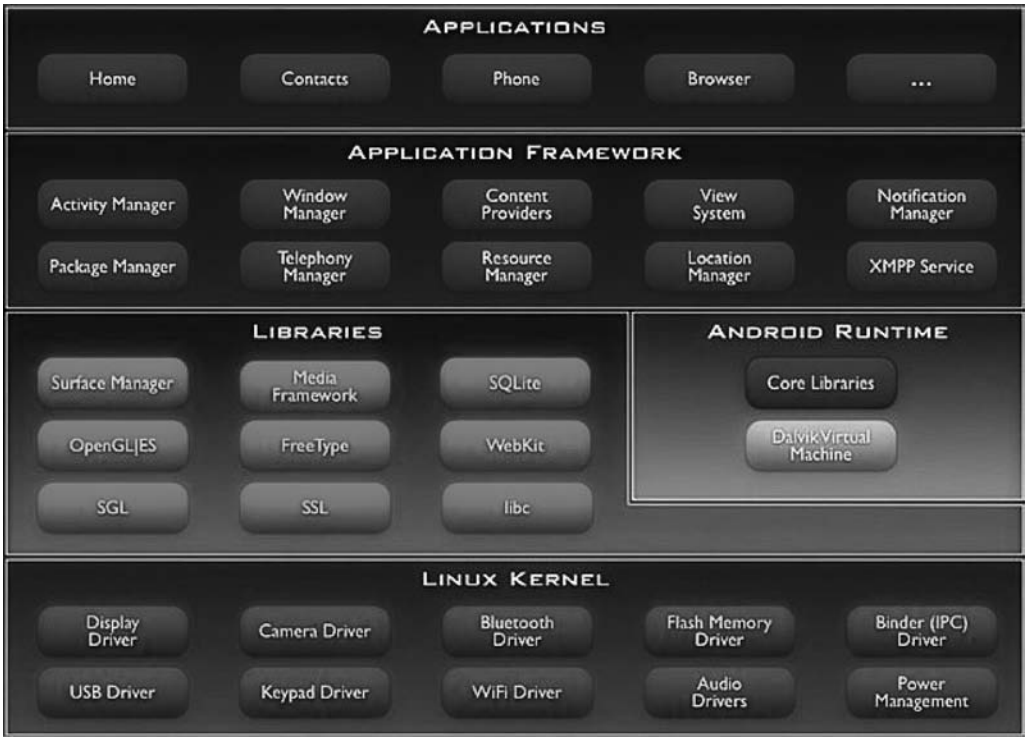


**图 5.46　Android 系统分层架构**

(2) Android 的四大组件。

Android 的四大组件分别为 Activity、Service、Content Provider、Broadcast Receiver。

① Activity。

- 一个 Activity 就是一个独立的界面。
- Activity 与 Activity 之间是利用 Internet 来实现通信的。
- 每一个 Activity 都要在 AndroidManifest. xml 中声明。

② Service。

- Service 用于在后台完成用户指定的操作。
- Service 需要在应用程序配置文件中进行声明。
- Service 组件没有图形 UI 界面。

③ Content Provider。

- 当需要在多个 Application 程序间进行数据共享时才需要 Content Provider。
- Content Provider 实现数据共享。

④ Broadcast Receiver。

- Broadcast Receiver 没有 User 界面,但可以通过开启一个 Activity 来响应收到的数据信息。
- Broadcast Receiver 的注册有两种方法:程序的动态注册和 AndroidManifest 文件中进行静态注册。

(3) Android 重力感应的开发。

重力加速度传感器(G-sensor),能够返回 X、Y、Z 三个方向的加速度。

该数值包含地心引力的影响,单位是 $m/s^2$。

将手机平放,X 轴为 0,Y 轴为 0,Z 轴为 9.81。

将手机朝下放在桌面上,Z 轴为−9.81。

将手机向左倾斜,X 轴为正值。

将手机向右倾斜,X 轴为负值。

将手机向上倾斜,Y 轴为负值。

将手机向下倾斜,Y 轴为正值。

### 5.11.3 软件工作的整个流程设计

设置手机终端的工作流程如下。

(1) 登录 iDrone,初始化系统参数。

(2) 检查 WiFi-OpenWrt 是否连接。

(3) 进入 App 操作界面,单击 Connect 按钮,再单击“一键起飞”按钮,利用重力感应进行飞行遥控。

(4) 微调飞行高度,调整云台舵机方向,观察周围环境,实时获取电压余量,若低于 11V 门限,则进行报警功能。

(5) 单击“一键降落”按钮,断开手机与四旋翼飞行器的连接,退出 App。

手机 App 工作流程如图 5.47 所示。

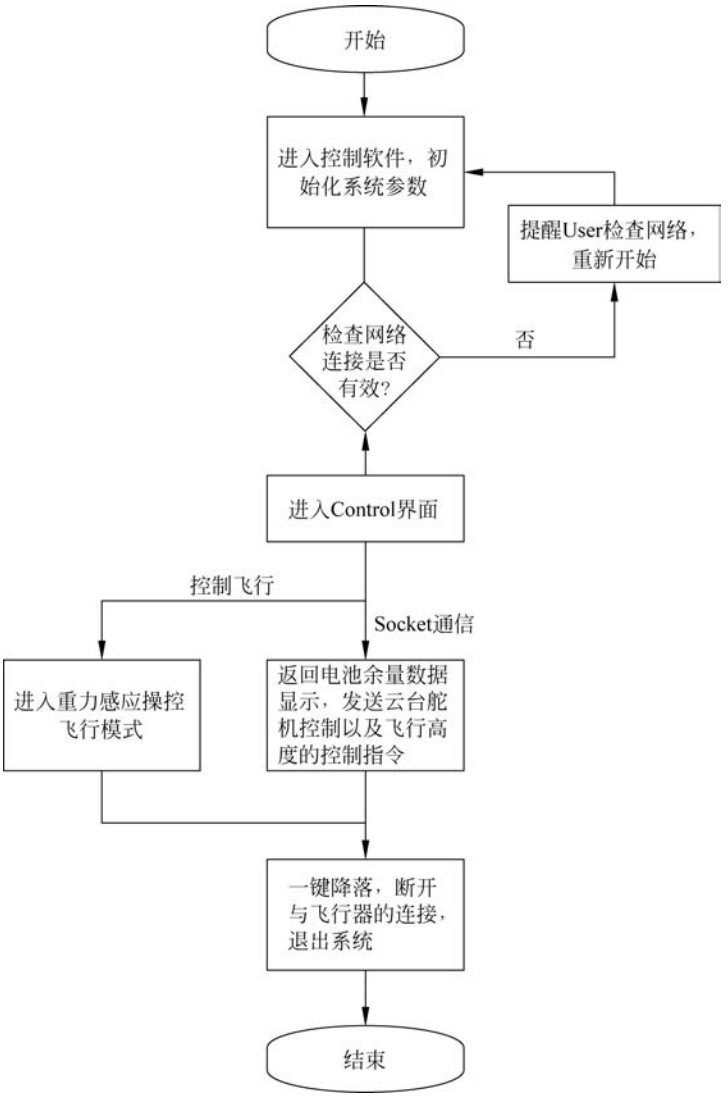


**图 5.47　手机 App 工作流程**

### 5.11.4　Android 传感器的种类

对于四旋翼飞行器的飞行控制，主要是利用手机终端的重力感应来实现的。Android 手持终端作为四旋翼飞行器的遥控系统的优势之一就是它含有多种传感器，如表 5.12 所示。

**表 5.12　Android 常用传感器种类**

| 传感器 | 说　　明 | 测量单位 | 常用场景 | 类　　型 |
|---|---|---|---|---|
| 重力加速度 | 测量应用于设备 X/Y/Z 三轴的加速度，包括重力 | $m/s^2$ | 运动检测(振动、倾斜等) | 硬件 |
| 陀螺仪 | 测量设备围绕设备 X/Y/Z 三轴的旋转率 | $m/s^2$ | 运动检测(旋转、翻转等) | 硬件 |

续表

| 传感器 | 说　明 | 测量单位 | 常用场景 | 类　型 |
|---|---|---|---|---|
| 线性加速度 | 测量应用于设备 X/Y/Z 三轴的加速度，重力除外 | $m/s^2$ | 检测一个单独的物理周的加速度 | 软件或硬件 |
| 旋转矢量 | 通过提供设备旋转矢量的三个要素测量设备的方向 | 无 | 运动检测和旋转检测 | 软件或硬件 |

四旋翼飞行器中选用重力加速度传感器来进行重力感应的遥控任务，利用加速度传感器能够十分灵敏地检测手机上下左右，能够通过倾斜手机来实现对飞行器的飞行控制，这种交互设计能够更直接轻松地对飞行器进行遥控。

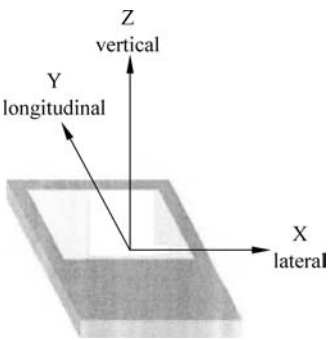


图 5.48　重力加速度传感器图

重力加速度传感器（G-sensor）主要用于获取 values[0]、values[1]、values[2]3 个参数。如图 5.48 所示，将手机平放，X 轴为 0，Y 轴为 0，Z 轴为 9.81。

如将手机旋转 90°放置，视频画面占满整个屏幕时，可以得到以下结论。

（1）把手机的正面朝下平放，Z 轴显示为－9.81；

（2）把手机的正面向上倾斜，X 轴显示为正值；

（3）把手机的正面向下倾斜，X 轴显示为负值；

（4）将手机的正面朝左倾斜，Y 轴显示为负值；

（5）将手机的正面朝右倾斜，Y 轴显示为正值。

### 5.11.5　重力感应的遥控方式

通过控制界面，进入重力感应遥控模式，接着就可以直接通过手机上下左右操控四旋翼飞行器的稳定飞行，具体使用方法如图 5.49 所示。

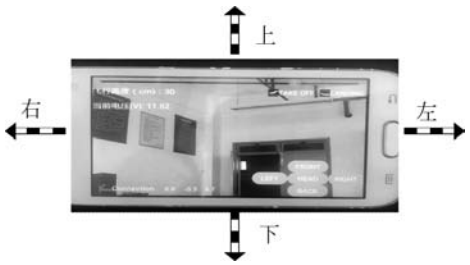


图 5.49　重力感应控制方式

### 5.11.6　重力感应遥控的实现

重力感应遥控的实现主要是通过 Android 感应检测管理——SensorManager。

1. 取得 SensorManager

使用感应检测 Sensor 首先要获取感应设备的检测信号，可以调用 Context.

getSysteService(SENSER_SERVICE)方法来取得感应检测的服务。

2. 实现取得感应检测 Sensor 状态的监听功能

通过以下两个 SensorEventListener 方法来监听,并取得感应检测 Sensor 状态:

```
public void onAccuracyChanged(Senso sensor,int accuracy);  //在感应检测到 Sensor 的精密度有
                                                           //变化时被调用到
public void onSensorChanged(SensorEvent event);            //在感应检测到 Sensor 的值有变化
                                                           //时会被调用到
```

3. 取得感应检测 Sensor 目标各类的值

通过下列 getSensorList()方法来取得感应检测 Sensor 的值:

```
List < Sensor > sensors = sm. getSensorList(Sensor.
TYPE_TEMPERATURE);
```

4. 注册 SensorListener

```
sm. regesterListener(SensorEventListener listener,
Sensor sensor, int rate);
```

第一个参数是监听 Sensor 事件;第二个参数是 Sensor 目标种类的值;第三个参数是延迟时间的精度密度。

5. 取消注册 SensorListener

```
sm.unregisterListener(SensorEventListener
listener)
```

6. 加速度感应检测——Accelerometer

Accelerometer Sensor 测量所有施加在设备上的力所产生的加速度的负值(包括重力加速度)。加速度所使用的单位是 $m/s^2$,数值是加速度的负值。

(1) SensorEvent. values[0]:加速度在 X 轴的负值。

(2) SensorEvent. values[1]:加速度在 Y 轴的负值。

(3) SensorEvent. values[2]:加速度在 Z 轴的负值。

重力感应的实现程序框图如图 5.50 所示。

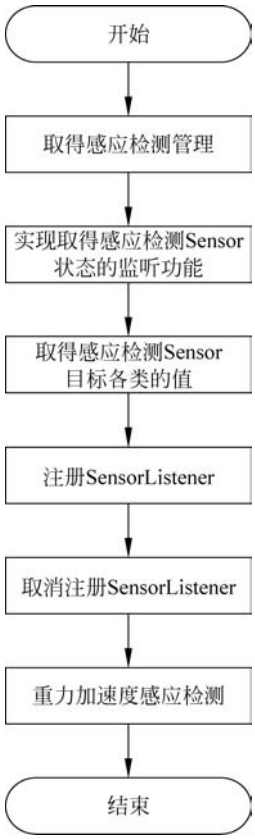


**图 5.50　重力感应的实现程序框图**

### 5.11.7　飞行控制高度的实现

SeekBar 可以用来当 Media 的进度指示和调整工具等,SeekBar 是 ProgressBar 的一个子类,飞行高度控制的实现框图如图 5.51 所示。

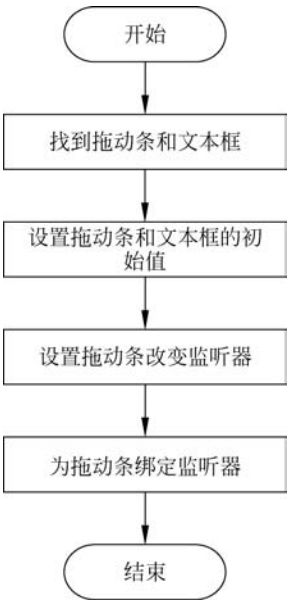


图 5.51 飞行高度控制的实现框图

## 5.12 手机终端与飞行控制板的通信

### 5.12.1 概述

在一般的设计中，人们用遥控器给飞行器发送指令，通常需要搭建一个地面站，由 NRF 进行无线传输，并且还需要对地面站的收发数据与液晶屏的显示进行程序上的编写，比较烦琐。若手机能够连接到路由器发射的 WiFi，那么就可以向飞行器发送控制命令。相比较而言，这种方法不需要遥控器、地面站、天线等大件实验器材的参与，实现起来简单方便，可扩展空间大。

### 5.12.2 数据通信模块

数据通信模块需要处理的数据，包含手机终端上传到四旋翼飞行器的飞行控制命令和四旋翼飞行器下载到手机终端的数据。

1. 上下行数据的传输

上下行数据的传输如表 5.13 和表 5.14 所示。

表 5.13 上行数据的传输

| 上行传输数据 | | 数据类型 | 数据值格式 |
|---|---|---|---|
| 1 | 一键起飞 | Int | _CMDUe $ |
| 2 | 一键降落 | Int | _CMDUf $ |
| 3 | 云台向上 | Int | _CMDUa $ |
| 4 | 云台向下 | Int | _CMDUb $ |
| 5 | 云台向左 | Int | _CMDUc $ |
| 6 | 云台向右 | Int | _CMDUd $ |
| 7 | 重力感应 X/Y/Z | Float | _CMDXx. x Yx. x Zx. x $ |
| 8 | 飞行高度 H | Int | _CMDHh $ |

注：_CMDXx. x Yx. x Zx. x $：当 X 轴或 Y 轴或 Z 轴在负轴方向时，x. x 表现为-x. x。

表 5.14　下行数据的传输

| 下行传输数据 | | 数据类型 | 数据值格式 |
|---|---|---|---|
| 9 | 电压余量 | Float | DYxx. x |

注：当 xx. x 小于等于 10.6V 时，产生警报。

2. 数据通信

四旋翼飞行器与手机终端系统通过基于 TCP/IP 的 Socket 实现连接，四旋翼飞行器是 Server，手机终端为 Client，Client 根据 Server 指定的 IP 和 Port 建立起 Socket 连接，然后从 Server 获取到输入流来读取并处理数据，最后把电压余量检测数值显示在 Control 界面。

Client 利用 Socket 连接到 Server，当 Client 和 Server 建立了 Socket 以后，程序就不会对 Server 和 Client 进行区分，而是直接通过两者的 Socket 进行通信。其中包括从 Socket 获取的输入流和输出流进行双向通信。

（1）实现四旋翼飞行器电压余量检测并报警的程序框图如图 5.52 所示。

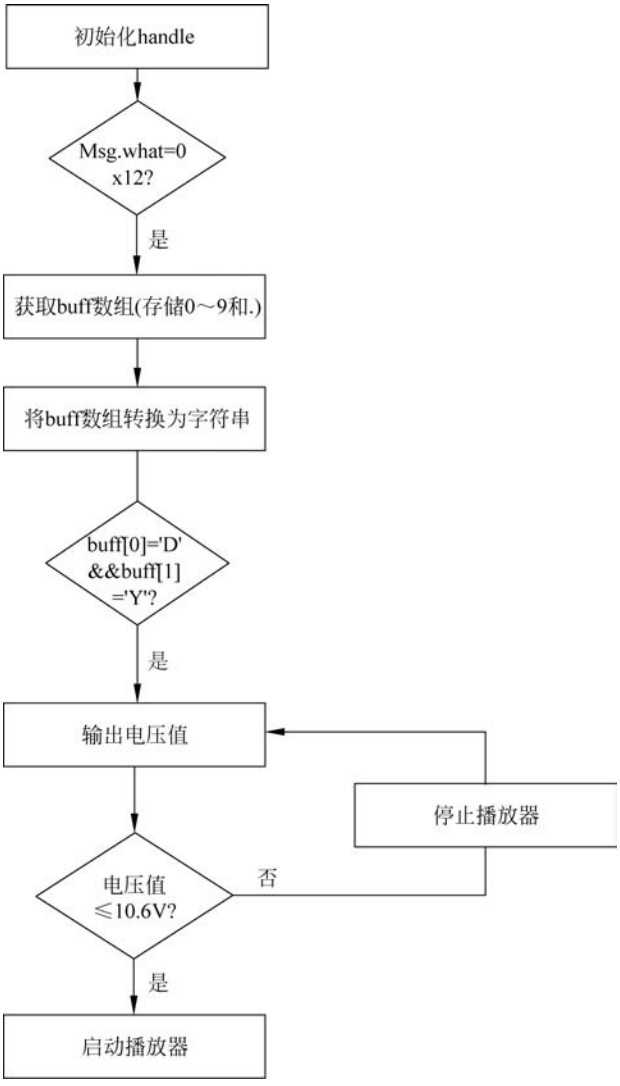


图 5.52　电压余量检测并报警的程序框图

(2) 控制云台方向以及一键起飞和降落的程序框图如图 5.53 所示。

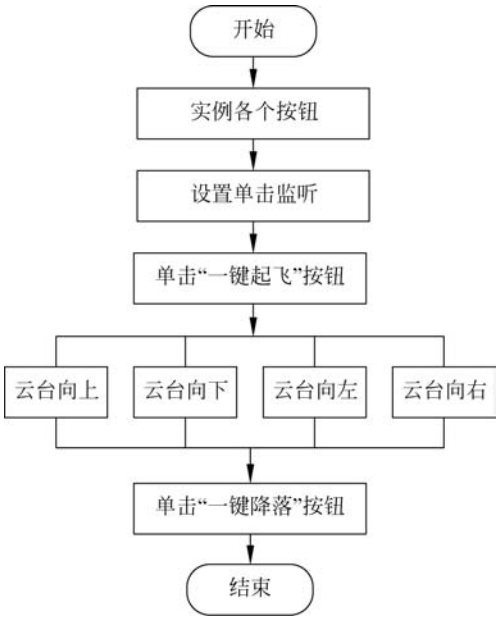


**图 5.53 实现云台方向控制以及一键起飞和降落程序框图**

### 5.12.3 手机终端界面设计

在 Windows 下搭建 Android 开发环境,利用 Eclipse 集成开发环境下设计 UI 界面模块。最关键的是控制界面的设计,这一界面用来显示四旋翼飞行器的飞行状态和四旋翼飞行器的状态信息,如图 5.54 所示。


**图 5.54 控制界面**

控制界面主要包括以下 6 部分。

(1) 重力感应数值。位于界面的左下方,可以通过重力感应控制飞行判断当前 X、Y、Z 的状态值。

(2) 云台舵机图像。整个界面的背景就是四旋翼飞行器上云台拍摄的实时视频。

(3) 云台舵机控制。位于界面的右下方,包括 FRONT、LEFT、BACK、RIGHT。

(4) 电压余量检测。位于界面的左上方,用于实时显示四旋翼飞行器的供电电压,若电压低于 11V 会启动自动报警功能。

（5）飞行高度控制。位于界面的最上方，用来拉动 Progress 亮标控制四旋翼飞行器的当前高度。

（6）一键起飞和降落。位于界面的右上方，单击按钮可以控制飞行器的一键起飞和一键降落。

### 5.12.4　WiFi 实时视频模块

WiFi 视频实时传输模块的实现就是将云台舵机传过来的数据进行解析，然后把每一帧的图片不断地绘图到手机屏幕上。WiFi 视频拍摄模块如图 5.55 所示。


**图 5.55　WiFi 视频拍摄模块**

利用 IP WiFi 云台，连接对应的 IP 和 Port，这里是 192.168.1.1:8080，就会在手机屏幕上获取到实时拍摄的画面，具体实现的程序框图如图 5.56 所示。

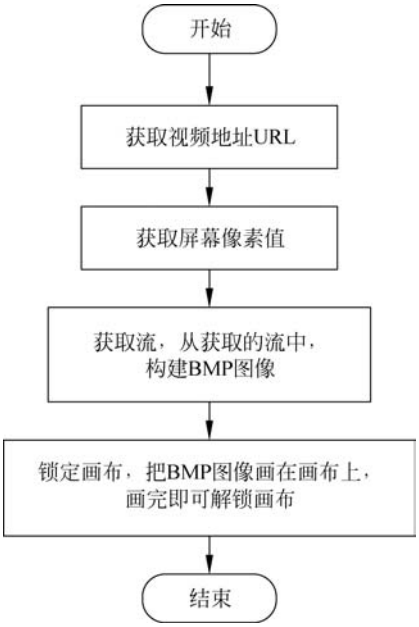


**图 5.56　WiFi 视频传输程序框图**