User Manual

ATK-MiniFly 遥控器开发指南

MiniFly

用户手册

ALIENTEK

广州市星翼电子科技有限公司

目 录

1. MiniFly 遥控器硬件资源

1.1 系统框架

MiniFly 遥控器系统框架如图 1.1.1 所示：

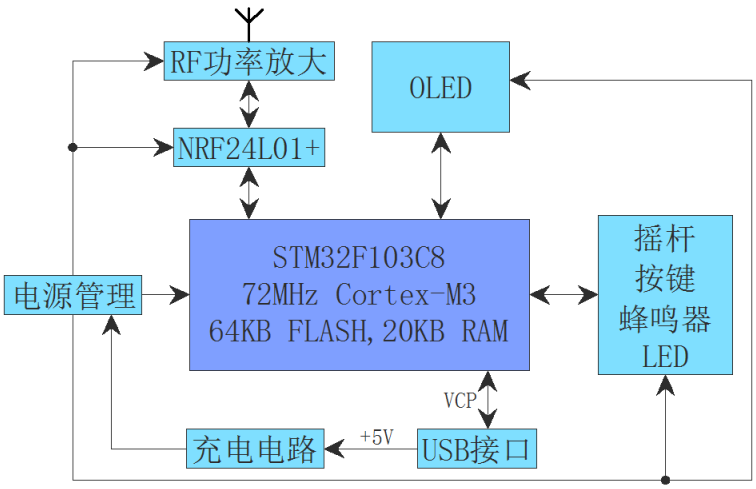

图 1.1.1 系统框架

MiniFly 遥控器采用了 STM32F103C8T6 作为控制 MCU，外围硬件主要有蓝色 0.96 寸 OLED、NRF24L01+、RF 功率放大、2.4G 天线、摇杆、按键、蜂鸣器、LED 等。各接口电路原理下面详细分析。

1.2 MCU 接口

MiniFly 遥控器 MCU STM32F103C8T6 电路图如图 1.2.1 所示：

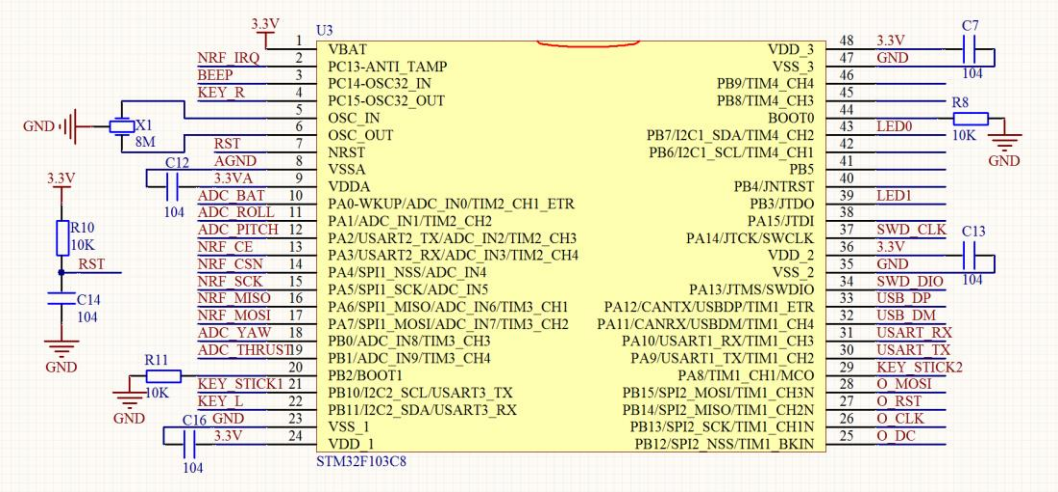

图 1.2.1 遥控器 MCU 接口

MiniFly 遥控器 MCU 采用 STM32F103C8T6，是 32-bit Cortex-M3 内核芯片，具有 20KB SRAM、64KB FLASH、7 个定时器、7 通道 DMA 控制器、2 个 SPI、2 个 IIC、3 个串口、1 个 USB 全速接口、1 个 CAN 接口、2 个 12 位 ADC、以及 35 个通用 IO 口等。该芯片外设丰富、功能强大，足以满足开发要求。我们 MiniFly 遥控器主要使用了 SPI、USB、ADC、DMA 等外设，其中 SPI 主要用于与 OLED 和 NRF24L01 通信；USB 主要用于与上位机通信和固件升级；ADC 和 DMA 主要用于采集摇杆电压。

1.3 OLED 接口

遥控器使用了蓝色 0.96 寸 OLED 做人机交互界面，接口电路如图 1.3.1 所示：

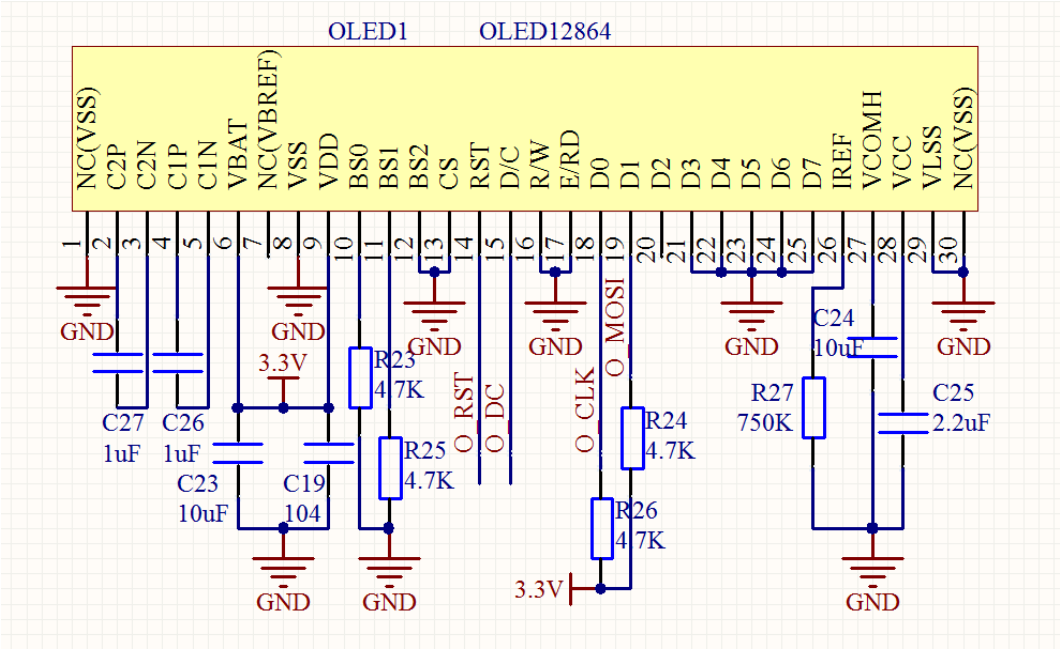

图 1.3.1 OLED 接口电路图

该蓝色 0.96 寸 OLED 分辨率为 128*64，内部驱动芯片为 SSD1306。该芯片内部集成 DCDC 升压，只需 3.3V 供电即可正常工作，支持 8 位 6800 并口、8 位 8080 并口、IIC 以及 4 线 SPI 等通信方式。这里我们使用了 4 线 SPI 方式通信，O_RST 为复位控制脚、O_DC 为数据命令控制脚、O_CLK 为 SPI 时钟控制脚、O_MOSI 为 SPI 数据输入脚。OLED 的控制使用的是 STM32 的 SP2。

1.4NRF24L01+和 PA 接口

MiniFly 遥控器使用了 NRF24L01+和四轴主机通信，接口电路图如图 1.4.1 所示：

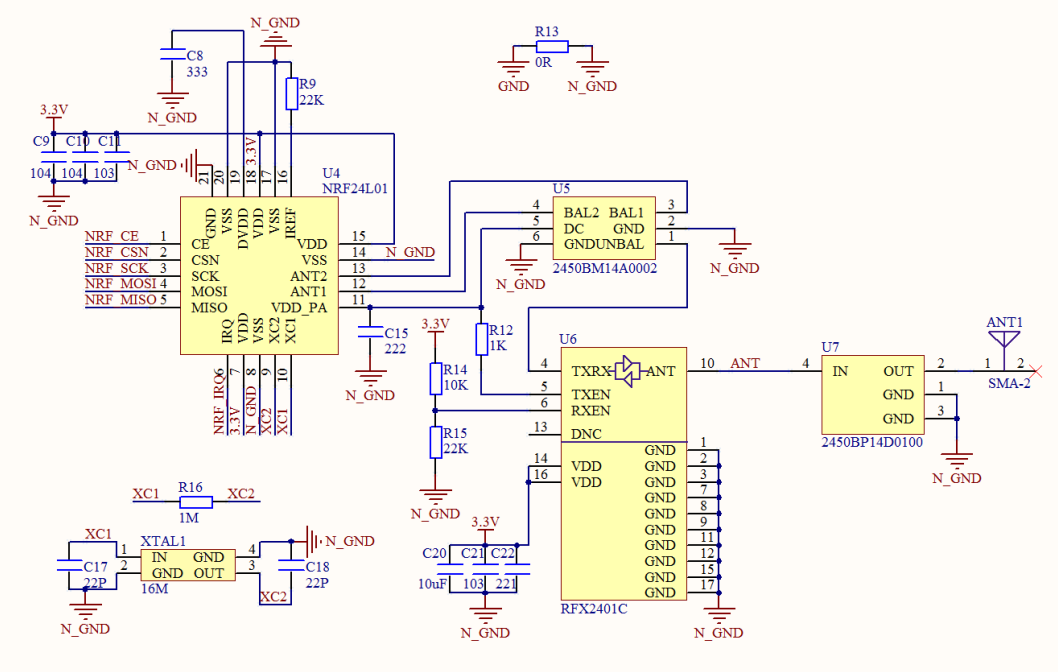

图 1.4.1 NRF24L01⁺和 PA 接口

NRF24L01⁺是 NORDIC 公司出产的 2.4GHz 无线通信芯片，早期版本是 NRF24L01，后期版本带了+。早期版本不能实时双向通信，同一时刻只能是发送模式或接收模式，实现收发功能需要来回切换模式。后期带+的版本嵌入了 Enhanced ShockBurst 通信机制，这个机制可以实现实时双向通信。下面我们简单介绍一下这个机制。

在 Enhanced ShockBurst 通信机制下，当配置为 TX 模式时，发送端可以发送带 ACK 标志的数据包和不带 ACK 标志的数据包。带 ACK 标志数据包意思就是接收端需要返回一个应答包，相反，不带 ACK 数据包即接收端就不必返回应答包。当发送不带 ACK 标志数据包时，通信机制自动打包数据并启动发送，发送后立即进入低功耗模式等待下一次发送。当发送带 ACK 标志数据包时，通信机制启动发送后立即转为接收模式等待应答包，如果接收到应答包则进入低功耗模式等待下一次发送并产生发送完成中断。如果等待接收超时则会自动重发，当重发次数达到设定的值时会结束当前发送并产生重发失败中断。当配置为 RX 模式时，模块一直处于接收状态，如果接收到数据包则会根据 ACK 的标志来选择要不要返回应答包。这样，两端就可以实现实时双向通信了，不需要手动切换发送接收模式。关于 Enhanced ShockBurst 通信机制在 nRF24L01P_Product_Specification_1_0.pdf 有详细的说明，大家可以看一下。

MiniFly 遥控器就是使用了该通信机制与四轴的 NRF51822 实时双向通信，配置为 TX 模式。NRF24L01⁺的控制使用 STM32 的 SPI1。

RFX2401C 是集成电路 RF 单片机(射频前端集成电路)，包含所有内部 IEEE 802.15.4/ZIGBEE，提供服务所需的射频功能无线传感器网络，和其他无线系统在 2.4Ghz 的 ISM 波段。RFX2401C 架构集 PA、LNA、收发开关电路，相关的匹配网络,在 CMOS 单片机和谐波滤波器装置。该芯片在上面电路中的主要作用是增大无线信号的功率，增大通信距离。

如图 1.4.1 所示，NRF24L01⁺ 的天线引脚（ANT1\ANT2）后接了一颗巴伦芯片 2450BM15A0002，该芯片是一个平衡滤波器，可以提升无线通信的性能。发送时，NRF24L01⁺的发射信号经过巴伦芯片平衡滤波后接到功率放大芯片 RFX2401C 上，信号放大后经过带通滤波器 2450BP14D0100 及搭配的 2.4G 高增益天线发射出去；接收时，天线接收 2.4GHz 频段信号经过带通滤波器 2450BP14D0100 滤波后经过 RFX2401C 功率放大和巴伦芯片平衡

滤波后回到 NRF24L01+。这样就可以大大增加通信距离，在空旷环境下，MiniFly 遥控器和四轴主机的通信距离大于 100 米。

1.5 摇杆和按键接口

MiniFly 遥控器板载了两个带按键的摇杆和两个独立按键，电路图如图 1.5.1 所示：

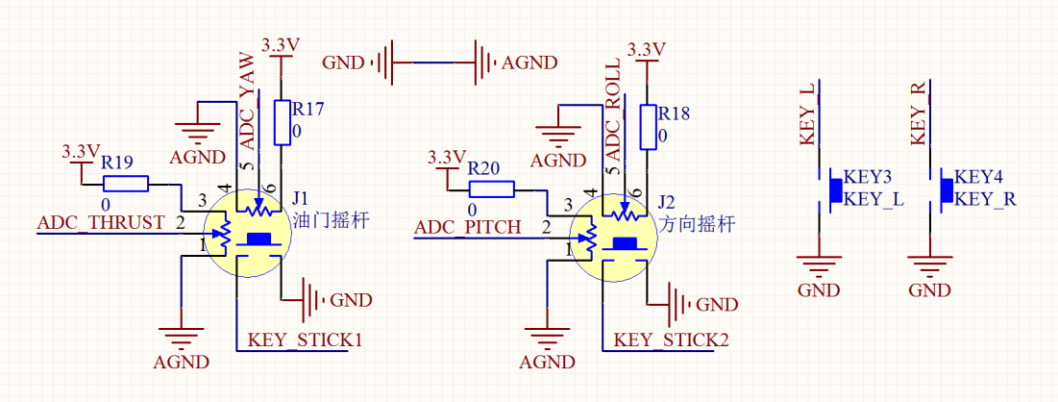

图 1.5.1 摇杆和按键接口

MiniFly 遥控器采用的摇杆是带按键和 360°自动回中类型的，方便定高飞行也可以手动飞行。一个摇杆有 2 个电位器，电位器中心抽头连接到 MCU 的 ADC 引脚。油门摇杆 2 个电位器的电压 AD 值转换为对应 THRUST（油门）和 YAW（航向角）的控制值；方向摇杆 2 个电位器的电压 AD 值转换为对应 PITCH（俯仰角）和 ROLL（横滚角）的控制值。所有电位器采集的 AD 值范围都在 0-4095，然后将 AD 值转换为对应控制值。

THRUST（油门）通道控制值的范围：0~100，当发送 0 表示没有油门输出，发送 100 表示油门满量程输出。

YAW（航向角）通道控制值范围：-200~200，当发送值 yaw 小于 0 时，四轴逆时针旋转，yaw 越小旋转快。相反，当发送值 yaw 大于 0 时，四轴顺时针旋转，yaw 越大旋转越快。

PITCH（俯仰角）通道控制值范围：-30~30，当发送值 pitch 小于 0 时，四轴向后运动，pitch 越小运动越快。相反，当发送值 pitch 大于 0 时，四轴向前运动，pitch 越大运动越快。

ROLL（横滚角）通道 AD 值换为控制值：-30~30，当发送值 roll 小于 0 时，四轴向左运动，roll 越小运动越快。相反，当发送值 roll 大于 0 时，四轴向右运动，roll 越大运动越快。

所有通道的控制值范围在程序中都可调节，飞行速度就是根据调节 PITCH 和 ROLL 通道的范围值实现。

1.6 LED 和蜂鸣器接口

MiniFly 遥控器板载了 2 个 LED 和 1 个蜂鸣器，电路图如图 1.6.1 所示：

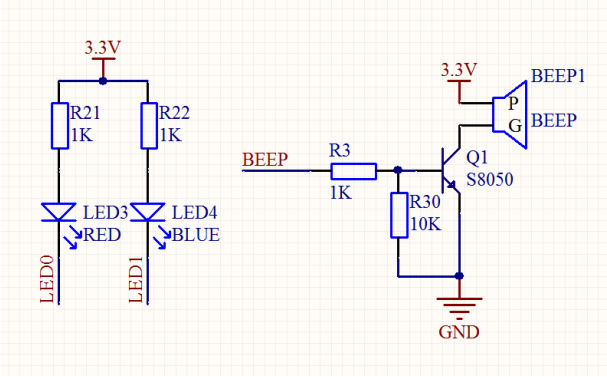

图 1.6.1 LED 和蜂鸣器接口

LED0（红）和 LED1（蓝）分别用于指示通信失败和通信成功，当通信失败时 LED0 常亮，当通信有干扰时 LED0、LED1 会交替闪烁，当通信良好时 LED1 常亮。蜂鸣器主要用于操作提示和低电量报警。

1.7 电源和 USB 接口

MiniFly 遥控器板载的电源接口部分，其电路图如图 1.7.1 所示：

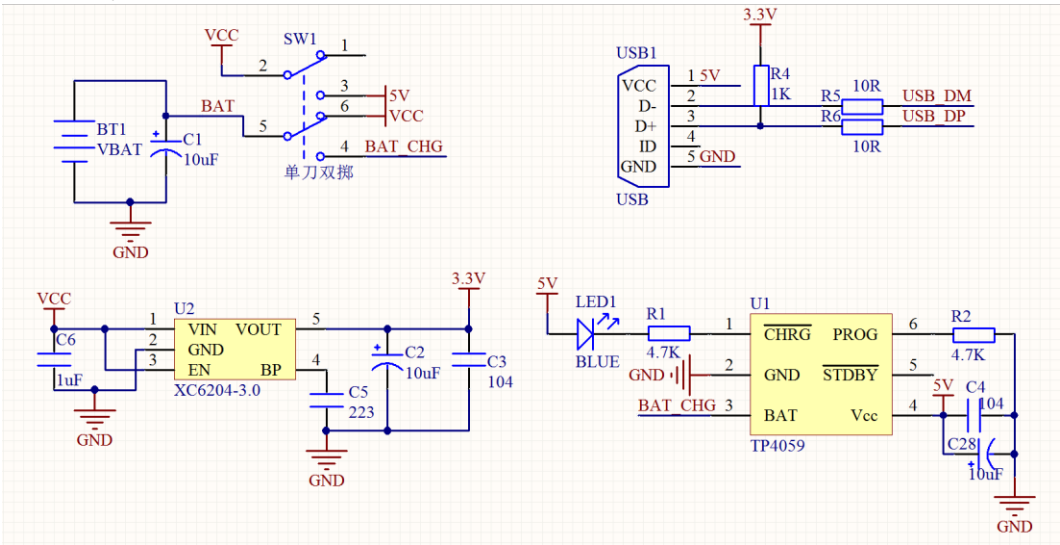

图 1.7.1 电源和 USB 接口

TP4059 是一款完整的单节锂离子电池充电器，带电池正负极反接保护反接，支持高达 600mA 的充电电流。该芯片的充电电流可以通过 PROG 脚的下拉电阻设定，我们的电路中设置的电阻 R2 为 4.7K，充电电流为 200mA。5V 电源（笔记本或者移动电源）可以通过 USB 线接上遥控器就可以给电池充电。遥控器电池为 200mAh 1C 的电池，充电电流小于等于 200ma。

XC6204B302 是一款低压差线性稳压器（LDO），输入电压高达 10.0V，输出 3.0V。因为是线性稳压器，所以输出纹波很小，同时低压差的功能就可以保证输入电压在很低的条件下输出电压也非常稳定。

如图 1.7.1 所示，当无外接 USB 时，单刀双掷开关往上，电池（BAT）连接到 VCC 给 XC6204B302 供电，开关往下打时则断开电池供电。当外接 USB 供电时，开关需要往下打，电池（BAT）连接到 BAT_CHG 这样才能给电池充电，VCC 连接到 5V，XC6204B302 的供电是通过到 USB 接入的 5V，这样就是可以一边玩一边充电。

1.8 下载接口

MiniFly 遥控器板载 SWD 仿真下载接口和 USART1 接口，电路图如图 1.8.1 所示：

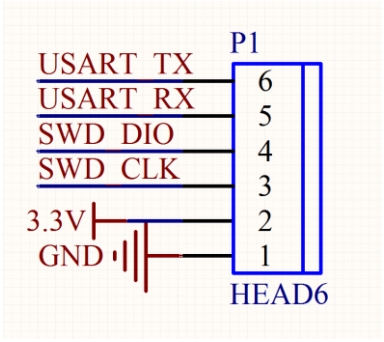

图 1.8.1 SWD 仿真下载接口

MiniFly 遥控器为客户预留了代码下载接口和 USART1 接口，方便客户二次开发。使用下载器下载程序时开关要拨到 STM32 标号处。

2. MiniFly 遥控器软件原理

2.1 通信链路

MiniFly 遥控器、四轴主机、上位机之间通信链路关系图如下图 2.1.1 所示：

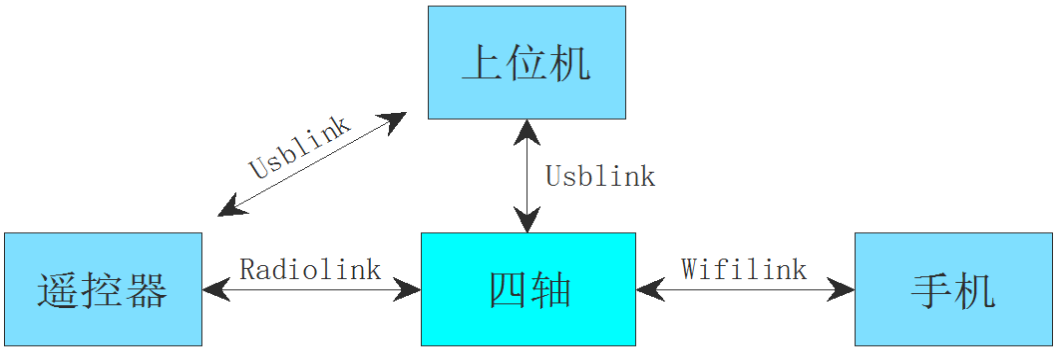

图 2.1.1 MiniFly 通信链路图

Radiolink 链路：无线通信方式，传输 ATKP 格式数据包。ATKP 格式如下：

MsgID（1 字节）	DataLen（1 字节）	Data（最多 30 字节）

MsgID：功能字；

DataLen：数据长度；

Data：有效数据（最多 30 字节，因为无线数据最大传输长度为 32 字节）。

Usblink 链路：STM32 USB 虚拟串口通信方式，传输带帧头和校验值的 ATKP 格式数据包。数据格式如下：

帧头（2 字节）	ATKP 数据包	校验（1 字节）

帧头为 0xAA、0xAA 时，传输方向：四轴→上位机、遥控器→上位机、四轴→遥控器。

帧头为 0xAA、0xAF 时，传输方向：四轴←上位机、遥控器←上位机、四轴←遥控器。

校验值：帧头至 ATKP 数据包所有字节之和。

WiFilink 链路：手机控制命令数据通过 WiFi 摄像头模块接收并转成串口方式输出与四轴通信，传输 WiFiPacket 数据包。WiFi 摄像头模块串口的输出只有 TX 引脚，所以 WiFilink 链路是单向通信，手机只能发送控制命令数据给四轴，四轴不能上传姿态数据给手机。WiFiPacket 格式如下：

帧头（1 字节）	功能数据	帧尾（1 字节）

帧头：0x66，帧尾：0x99。

遥控器与四轴 NRF51822 无线通信，无线通信本身自带了 CRC16 校验，传输是可靠的，因此直接可以传输 ATKP 数据包。Usblink 链路通信是串口方式，为了区分一帧数据和保证传输可靠性，所以需要加上帧头和校验。WiFilink 链路同理，也需要加上帧头和帧尾。

当遥控器与四轴 NRF51822 通信成功后，遥控器会不断发送数据包给 NRF51822，NRF51822 接收并解析，然后再转发给 STM32F11，STM32F411 接收到后会立即向 NRF51822 返回一条数据包，一应一答模式。NRF51822 与 STM32F411 通信使用的串口方式，数据格式跟 usblink 链路数据格式一样。如果遥控器连接了上位机，遥控器会将接收到的应答数据包先解析，解析完成后转发给上位机。同样，上位机发下来的数据遥控器也会先解析再转发给四轴。Radiolink 链路中，遥控器定周期发送控制命令，四轴定周期返回姿态和其他数据。Usblink 链路中，如果四轴连接了上位机，四轴会定周期返回姿态和其他数据给上位机；如果遥控器连接了上位机，遥控器将四轴返回的数据转发给上位机。上位机同一时刻只能连接

一个 Usblink 链路。

当四轴外接 WiFi 摄像头模块时，摄像头图像数据通过 WiFi 传给手机 APP，手机 APP 可以控制四轴飞行。**注意：当手机和遥控器同时打开时，手机和遥控器不能同时控制四轴飞行。**因为四轴程序中设置了优先响应遥控器的控制命令，只有遥控器关闭或者加锁后，不再发送姿态控制命令，四轴才会响应手机的控制命令。

2.2 程序框架

打开 MiniFly 遥控器固件 Firmware_F103 工程，展开工程分组如下图 2.3.1 所示：

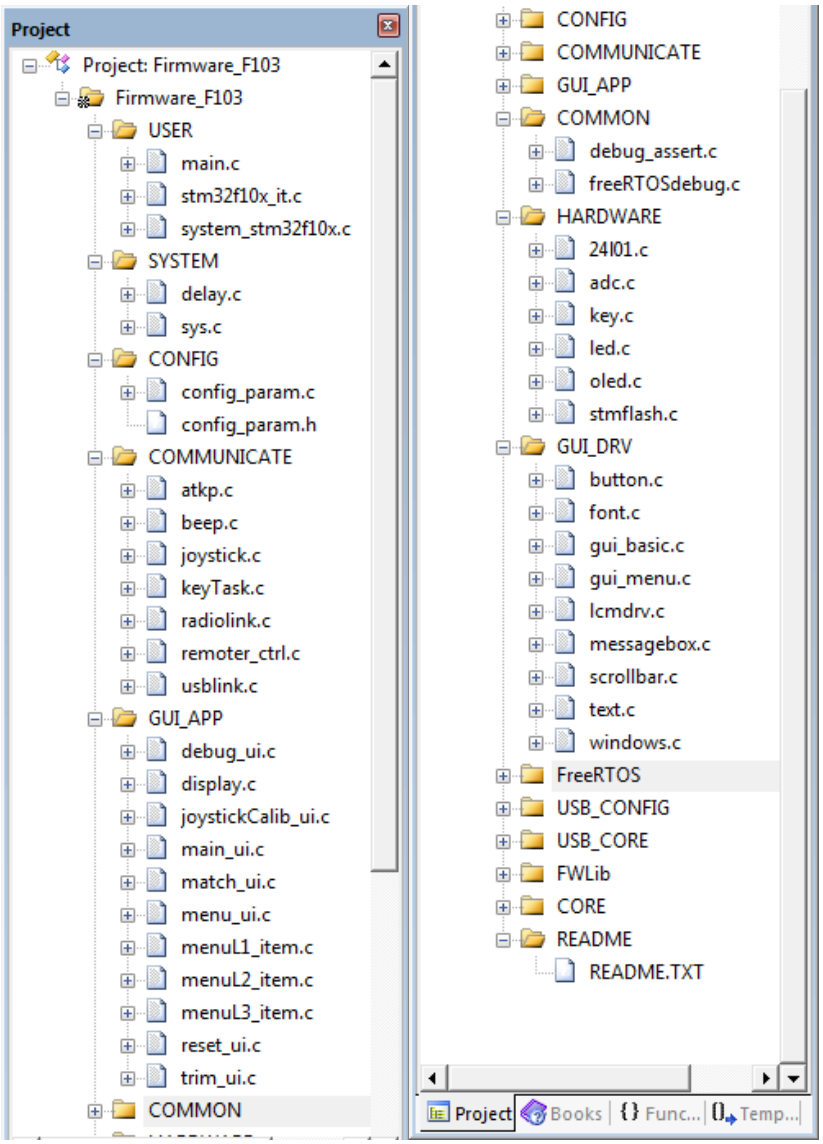

图 2.3.1 工程分组

如图 2.3.1 所示，可以看到工程分有 14 个分组，其中 CONFIG 分组主要是配置参数保存至 Flash 相关的代码。COMMUNICATE 分组主要是跟通信相关的代码。GUI_APP 分组主要是界面显示相关的代码。COMMON 分组主要是调试相关的代码。HARWARE 分组主要是硬件底层驱动相关代码。GUI_DRV 分组主要是界面驱动相关代码。

下面我们先讲解一下 USER 分组的 main.c。main.c 包含了所有硬件初始化和任务的创建的代码。初始化代码如下：

```
int main(void)
```

```
{
    NVIC_SetVectorTable(FIRMWARE_START_ADDR,0);
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_4);/*中断配置初始化*/
    delay_init();           /*delay 初始化*/
    configParamInit();      /*配置参数初始化*/
    ledInit();              /*led 初始化*/
    oledInit();             /*oled 初始化*/
    beepInit();             /*蜂鸣器初始化*/
    keyInit();              /*按键初始化*/
    joystickInit();         /*摇杆初始化*/
    usb_vcp_init();         /*usb 虚拟串口初始化*/
    radiolinkInit();        /*无线通信初始化*/
    usblinkInit();          /*usb 通信初始化*/
    displayInit();          /*显示初始化*/
    /*创建起始任务*/
    xTaskCreate(startTask, "START_TASK", 100, NULL, 2, &startTaskHandle);
    vTaskStartScheduler();/*开启任务调度*/
    while(1){};             /*  任务调度后不会执行到这  */
}
```

任务创建代码如下：

```
void startTask(void *param)
{
    /*进入临界区*/
    taskENTER_CRITICAL();
    /*创建无线连接任务*/
    xTaskCreate(radiolinkTask, "RADIOLINK", 100, NULL, 6, &radiolinkTaskHandle);
    /*创建 usb 发送任务*/
    xTaskCreate(usblinkTxTask, "USBLINK_TX", 100, NULL, 5, NULL);
    /*创建 usb 接收任务*/
    xTaskCreate(usblinkRxTask, "USBLINK_RX", 100, NULL, 5, NULL);
    /*创建飞控指令发送任务*/
    xTaskCreate(commanderTask, "COMMANDER", 100, NULL, 4, NULL);
    /*创建按键扫描任务*/
    xTaskCreate(keyTask, "BUTTON_SCAN", 100, NULL, 3, NULL);
    /*创建显示任务*/
    xTaskCreate(displayTask, "DISPLAY", 200, NULL, 1, NULL);
    /*创建参数配置任务*/
    xTaskCreate(configParamTask, "CONFIG_TASK", 100, NULL, 1, NULL);
    /*创建无线通信数据处理任务*/
    xTaskCreate(radiolinkDataProcessTask, "DATA_PROCESS", 100, NULL, 6, NULL);
    /*创建 USB 通信数据处理任务*/
    xTaskCreate(usblinkDataProcessTask, "DATA_PROCESS", 100, NULL, 6, NULL);
    /*删除开始任务*/
    vTaskDelete(startTaskHandle);
```

```
        /*退出临界区*/
        taskEXIT_CRITICAL();
    }
```

初始化代码主要是硬件底层驱动的初始化，初始化完毕先创建一个起始任务。任务创建代码主要是在起始任务里再创建系统功能任务，任务创建完毕之后删除起始任务并启动任务调度。下面我们来讲解一下功能任务的具体作用。

radiolinkTask 主要功能是发送 ATKP 数据包给四轴，并接收四轴返回的应答包。radiolinkTask 函数在 radiolink.c 中。

usblinkTxTask 主要功能是给 ATKP 数据包加上帧头和校验并发送给上位机。usblinkTxTask 函数在 usblink.c 中。

usblinkRxTask 主要功能是接收上位机发下来的串口数据，按照 ATKP 格式打包。usblinkRxTask 函数在 usblink.c 中。

commanderTask 主要功能是将采集摇杆电位器的 AD 值转换为姿态控制命令，并以 10ms 的周期通过 radiolink 链路发送给四轴。commanderTask 函数在 remoter_ctrl.c 中。

keyTask 主要功能是扫描按键，根据按键按下的时间来区分长按和短按。keyTask 函数在 keyTask.c 中。

displayTask 主要功能是显示界面，50ms 刷新一次界面。displayTask 函数在 display.c 中。

configParamTask 主要功能是保存参数。任务中 1000ms 判断一次配置参数有无改变，当有参数改变后 6S 内不再改变则将新参数写入 Flash。这样做的目的是避免在微调四轴时频繁写 Flash，Flash 擦写次数是有限的。configParamTask 函数在 config_param.c 中。

radiolinkDataProcessTask 主要功能是处理四轴返回的应答包数据，处理完之后再通过 usblink 链路转发给上位机。radiolinkDataProcessTask 函数在 atkp.c 中。

usblinkDataProcessTask 主要功能是处理上位机发下来的 ATKP 数据包，处理完之后通过 radiolink 链路转发给四轴。上位机发下来的 ATKP 数据包由 usblinkRxTask 打包。usblinkDataProcessTask 函数在 atkp.c 中。

为了更加直观的了解代码框架，下面对部分代码做了连接关系，如下图 2.3.2 所示：

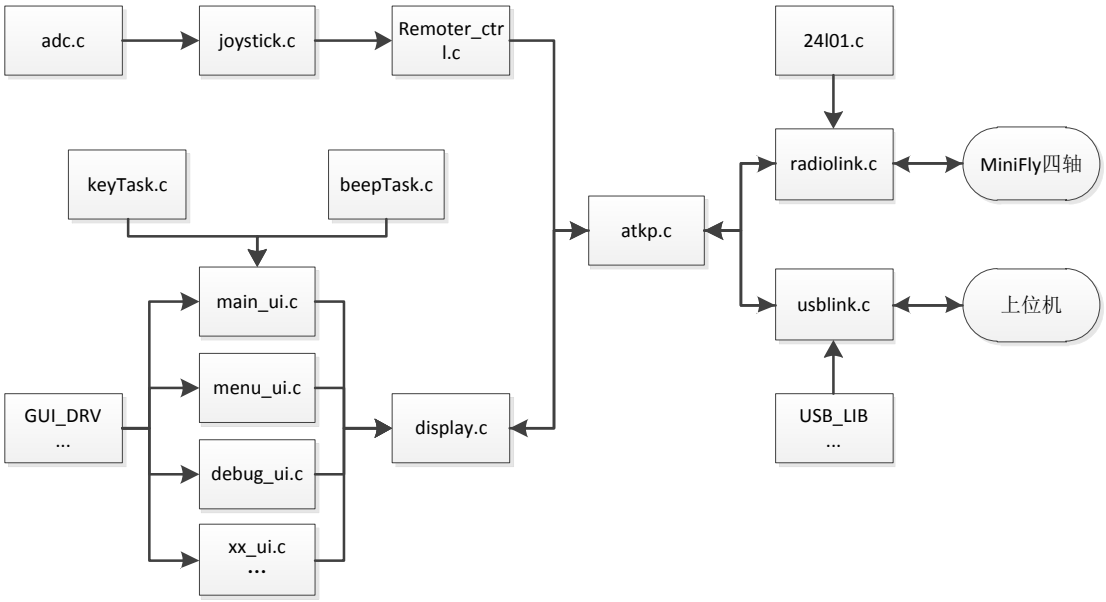

图 2.3.2 代码关系图

如图 2.3.2 所示，adc.c 主要实现采集摇杆电位器电压 AD 值。joystick.c 主要实现将 AD 值转为 THRUST、YAW、PITCH、ROLL 对应百分比。remoter.c 主要实现将百分比乘以设

定速度值并打包成 ATKP 包格式，然后以 10ms 周期性发送到 radiolink.c 的发送队列中，即上述的 commanderTask。radiolink.c 主要实现实时发送 ATKP 数据包给四轴，发送成功四轴会返回一个应答包，即上述的 radiolinkTask。usblink.c 主要实现发送 ATKP 格式数据包到上位机，接收上位机串口数据并打包为 ATKP 格式，即上述的 usblinkTxTask 和 usblinkRxTask。aktp.c 主要实现解析四轴通过 radiolink.c 返回的应答包并通过 usblink.c 转发给上位机，上位机通过 usblink.c 发下来的数据包将通过 radiolink 转发给四轴，即上述的 radiolinkDataProcessTask 和 usblinkDataProcessTask。usblink.c 是 USB_LIB 驱动代码上构建的，USB_LIB 即 USB_CONFIG 分组和 USB_CORE 分组的驱动代码。USB_LIB 主要实现了 USB 虚拟串口的功能。

main_ui.c、menu_ui.c、debug_ui.c 及其他 xx_ui.c 均是在 GUI_DRV 分组里的驱动代码上构建的，GUI_DRV 驱动代码主要是 oled 驱动代码、oled 显示字符串及汉字代码、oled 基本绘图代码等。mian_ui.c 主要实现主界面显示、低电量报警、解锁加锁、一键起飞降落、紧急停机、一键翻滚、切换至调试界面、切换至微调界面等功能。menu_ui.c 主要实现菜单显示功能。debug_ui.c 主要实现显示调试界面。界面切换是通过按键来实现，所以几乎所有的界面代码 xx_ui.c 都调用了 keyTask.c。部分界面需要蜂鸣器报警功能，所以调用了 beepTask.c。

2.3 通信协议

通信协议部分主要在 atkp.h 中，ATKP 数据包格式及 msgID 功能字定义代码如下：

```
/*上行帧头*/
#define UP_BYTE1 0xAA
#define UP_BYTE2 0xAA
/*下行帧头*/
#define DOWN_BYTE1 0xAA
#define DOWN_BYTE2 0xAF
#define ATKP_MAX_DATA_SIZE 30
/*ATKP 通讯数据结构*/
typedef struct
{
    u8 msgID;
    u8 dataLen;
    u8 data[ATKP_MAX_DATA_SIZE];
}atkp_t;

/*上行指令 ID*/
typedef enum
{
    UP_VERSION      = 0x00,
    UP_STATUS       = 0x01,
    UP_SENSER       = 0x02,
    UP_RCDATA       = 0x03,
    UP_GPSDATA      = 0x04,
    UP_POWER        = 0x05,
    UP_MOTOR        = 0x06,
```

```
    UP_SENSER2      = 0x07,
    UP_FLYMODE      = 0x0A,
    UP_SPEED        = 0x0B,
    UP_PID1         = 0x10,
    UP_PID2         = 0x11,
    UP_PID3         = 0x12,
    UP_PID4         = 0x13,
    UP_PID5         = 0x14,
    UP_PID6         = 0x15,
    UP_RADIO        = 0x40,
    UP_MSG          = 0xEE,
    UP_CHECK        = 0xEF,
    UP_REMOTOR      = 0x50,
    UP_PRINTF       = 0x51,
}upmsgID_e;

/*下行指令*/
#define  D_COMMAND_ACC_CALIB        0x01
#define  D_COMMAND_GYRO_CALIB       0x02
#define  D_COMMAND_MAG_CALIB        0x04
#define  D_COMMAND_BARO_CALIB       0x05
#define  D_COMMAND_FLIGHT_LOCK      0xA0
#define  D_COMMAND_FLIGHT_ULOCK     0xA1
#define  D_ACK_READ_PID             0x01
#define  D_ACK_READ_VERSION         0xA0
#define  D_ACK_RESET_PARAM          0xA1
/*下行指令 ID*/
typedef enum
{
    DOWN_COMMAND    = 0x01,
    DOWN_ACK        = 0x02,
    DOWN_RCDATA     = 0x03,
    DOWN_POWER      = 0x05,
    DOWN_FLYMODE    = 0x0A,
    DOWN_PID1       = 0x10,
    DOWN_PID2       = 0x11,
    DOWN_PID3       = 0x12,
    DOWN_PID4       = 0x13,
    DOWN_PID5       = 0x14,
    DOWN_PID6       = 0x15,
    DOWN_RADIO      = 0x40,
    DOWN_REMOTOR    = 0x50,
}downmsgID_e;
```

上行帧头、下行帧头、ATKP 数据包格式在上述 2.1 通信链路中的 usblink 链路和 radiolink

链路已经说明，这里不再赘述。上行指令 ID 和下行指 ID 令顾名思义即 ATKP 格式中的功能字 msgID，作用是在通信时用于区分该数据帧的作用。大部分指令 ID 主要用于上位机通信，指令 ID 的定义主要也是参考了匿名科创上位机的通信协议，匿名科创上位机的通信协议在可以在软件中查看，具体操作如下图 2.3.1 所示：

图 2.3.1 查看上位机通信协议

按照图 2.3.1 操作点击通信协议，将会自动打开通信协议 EXCEL 表格，如下图 2.3.2 所示：

图 2.3.2 匿名科创上位机通信协议

上位机通信指令 ID（功能字）及对应数据格式在这个通信协议 EXCEL 表格有详细的说明，这里就不详细说明。这个 EXCEL 对应的实现代码在四轴的 STM32F411_Firmware 工程的 atkp.c 中，大家可以结合程序理解。下面讲解一下遥控器和四轴通信的指令 ID。

首先是 DOWN_RADIO 指令。为了区分指令是发给谁的，我们使用 DOWN_RADIO 指令 ID 来指定是发送给 NRF51822 的指令。然后使用 Data[0]分为了 3 条下行功能指令，这 3 条下行指令和对应上行指令宏定义在 radiolink.h 中定义，代码如下：

```
/*上行指令*/
#define  U_RADIO_RSSI          0x01
#define  U_RADIO_CONFIG        0x02
```

```
/*下行指令*/
#define   D_RADIO_HEARTBEAT      0xFF
#define   D_RADIO_GET_CONFIG     0x01
#define   D_RADIO_SET_CONFIG     0x02
```

D_RADIO_HEARTBEAT 是遥控器实时发送给四轴 NRF51822 的心跳包。NRF51822 接收到心跳包后转发给 STM32F411，STM32F411 返回一条数据包给 NRF51822，NRF51822 则返回给遥控器。当 STM32F411 没有数据要返回时，NRF51822 会返回 U_RADIO_RSSI 数据包给遥控器。U_RADIO_RSSI 数据包的数据是无线信号强度值 rssi。发收数据格式如下：

	MsgID	Datalen	Data[0]	Data[1-29]
Send	0x40	1	0xFF	NULL
Back	0x40	1	0x01	rssi

如：

遥控器发送：0x40 0x01 0xFF

NRF51822 返回：0x40 0x01 0x01 0x2C

说明接收到的信号强度 rssi = 0x2C，即 44。

D_RADIO_GET_CONFIG 是执行匹配四轴时，遥控器发送给 NRF51822 获取无线配置的命令。NRF51822 接收此命令后会返回 U_RADIO_CONFIG 数据包。无线配置数据结构在 config_param.h 中定义，代码如下：

```
/*无线配置数据结构*/
typedef struct
{
    u8 channel;
    enum nrfRate dataRate;
    u32 addressHigh;    /*通信地址高 4 字节*/
    u32 addressLow;     /*通信地址低 4 字节*/
}radioConfig_t;
```

发收送数据格式如下：

	MsgID	Datalen	Data[0]	Data[1-29]
Send	0x40	1	0x01	NULL
Back	0x40	1+sizeof(radioConfig)	0x02	radioConfig

如：

遥控器发送：0x40 0x01 0x01

NRF51822 返回：0x40 0x0A 0x02 0x50 0x02 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

说明接收到的 radioConfig.channel = 0x50、radioConfig.dataRate = 0x02、radioConfig.addressHigh = 0x04030201、radioConfig.addressLow = 0x08070605。这里很多人会觉得奇怪，为什么不是 radioConfig.addressHigh = 0x01020304 呢？答：因为 STM32 是小端模式，即高位地址存储的是高位数据。例如，u32 temp = 0x12345678 的存储地址是 0x2000010，假如数组 u8 buf[4]的存储地址也是 0x2000010，那么 buf[0] = 0x78，buf[1] = 0x56，buf[2] = 0x34，buf[3] = 0x12。

D_RADIO_GET_CONFIG 是遥控器执行匹配四轴时，发送给 NRF51822 设置无线配置的命令。发收数据格式如下：

	MsgID	Datalen	Data[0]	Data[1-29]

Send	0x40	1+sizeof(radioConfig)	0x02	radioConfig
Back	无			

如：

遥控器发送：0x40 0x0A 0x02 0x50 0x02 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

NRF51822 返回：无返回

说明发送的无线配置 radioConfig.channel = 0x50、radioConfig.dataRate = 0x02、radioConfig.addressHigh = 0x04030201、radioConfig.addressLow = 0x08070605。

第二是 DOWN_REMOTOR 指令。DOWN_REMOTOR 指令 ID 是用来指定是遥控器下行给四轴的命令。然后使用 Data[0]分区分发送控制命令和控制数据发送。控制命令和控制数据枚举如下：

```
/*遥控数据类别*/
typedef enum
{
    REMOTOR_CMD,
    REMOTOR_DATA,
}remoterType_e;
```

控制命令主要是控制四轴实现一些功能性操作的命令，比如一键起飞降落、一键翻滚、一键紧急停止等。控制数据主要是发送给四轴姿态控制数据。当 Data[0] == REMOTOR_CMD 时，Data[1]为控制命令；当 Data[0]== REMOTOR_DATA 时，Data[1]之后为控制数据。控制数据结构如下：

```
/*遥控控制数据结构*/
typedef __packed struct
{
    float roll;
    float pitch;
    float yaw;
    float thrust;
    float trimPitch;
    float trimRoll;
    bool ctrlMode;
    bool flightMode;
    bool RCLock;
} remoterData_t;
```

发送控制数据时，发收数据格式如下：

	MsgID	Datalen	Data[0]	Data[1-29]
Send	0x50	1+sizeof(remoterData)	0x01	remoterData
Back	无			

发送控制数据函数代码如下：

```
/*发送遥控控制数据*/
void sendRmotorData(u8 *data, u8 len)
{
    if(radioinkConnectStatus() == false)
        return;
```

```
        atkp_t p;
        p.msgID = DOWN_REMOTOR;
        p.dataLen = len + 1;
        p.data[0] = REMOTOR_DATA;
        memcpy(p.data+1, data, len);
        radiolinkSendPacket(&p);
    }
```

调用发送控制数据函数的具体代码如下：

```
    remoterData_t    send;
    send.roll = 0.0;
    ……/*给 send 结构体赋值*/
    sendRmotorData((u8*)&send, sizeof(send));
```

如以上代码，当需要控制数据时，先使用 remoterData_t 定义一个 send 结构体数据，然后调用 sendRmotorData((u8*)&send, sizeof(send));即可发送控制数据了。

遥控器功能操作指令共有 5 条，这 5 条功能指令宏定义在 remoter_ctrl.h 中定义，代码如下：

```
    /*下行指令*/
    #define   CMD_GET_MSG              0x01/*获取四轴信息（自检）*/
    #define   CMD_GET_CANFLY           0x02/*获取四轴是否能飞*/
    #define   CMD_FLIGHT_LAND          0x03/*起飞、降落*/
    #define   CMD_EMER_STOP            0x04/*紧急停机*/
    #define   CMD_FLIP                 0x05/*4D 翻滚*/
    /*上行指令*/
    #define   ACK_MSG                  0x01 /*四轴返回的信息*/
```

CMD_GET_MSG 是获取四轴状态信息的命令。该命令在 main_ui.c 中调用，500ms 定周期发送一次。四轴接收此命令后返回 ACK_MSG 数据包，数据包包含了 MiniFlyMsg 数据结构，该数据结构在 remoter_ctrl.h 中定义，代码如下：

```
    typedef __packed struct
    {
        u8 version;
        bool mpu_selfTest;      /*MPU9250 自检*/
        bool baro_slfTest;      /*气压计自检*/
        bool isCanFly;          /*四轴是否飞行*/
        bool isLowpower;        /*四轴是否低电压*/
    } MiniFlyMsg_t;
```

发收送数据格式如下：

	MsgID	Datalen	Data[0]	Data[1]	Data[2]	Data[3-29]
Send	0x50	3	0x00	0x01	0x00	NULL
Back	0x50	1+sizeof(MiniFlyMsg)	0x01	MiniFlyMsg		

如：

遥控器发送：0x50 0x03 0x00 0x01 0x00

STM32F411 返回：0x50 0x06 0x01 0x0A 0x01 0x01 0x01 0x00

说明返回的 MiniFlyMsg.version = 0x0A，MiniFlyMsg.mpu_selfTest = 0x01, MiniFlyMsg.baro_selfTest = 0x01, MiniFlyMsg.mpu_isCanFly = 0x01, MiniFlyMsg.isLowpower

= 0x00。

CMD_GET_CANFLY 命令未使用。

CMD_FLIGHT_LAND 是发送一键起飞和降落功能，无返回。数据格式如下：

	MsgID	Datalen	Data[0]	Data[1]	Data[2]	Data[3-29]
Send	0x50	3	0x00	0x03	0x00	NULL
Back	无					

如：

遥控器发送：0x50 0x03 0x00 0x03 0x00

STM32F411 返回：无返回

当四轴接收到此命令后先一键起飞，再次接收到此命令后则一键降落，如此循环即可实现一键起飞和一键降落的功能。

CMD_EMER_STOP 是发送紧急停机功能，无返回。数据格式如下：

	MsgID	Datalen	Data[0]	Data[1]	Data[2]	Data[3-29]
Send	0x50	3	0x00	0x04	0x00	NULL
Back	无					

如：

遥控器发送：0x50 0x03 0x00 0x04 0x00

STM32F411 返回：无返回

当四轴在定高飞行过程中接收到此命令则立即停止电机转动；手动模式下直接松开油门即可停止电机转动，此命令无效。

CMD_FLIP 是发送一键翻滚功能，flipDir 为翻滚方向，无返回。翻滚方向枚举如下：

```
enum dir_e
{
    CENTER=0,
    FORWARD,
    BACK,
    LEFT,
    RIGHT,
};
```

命令数据格式如下：

	MsgID	Datalen	Data[0]	Data[1]	Data[2]	Data[3-29]
Send	0x50	3	0x00	0x05	flipDir	NULL
Back	无					

如：

遥控器发送：0x50 0x03 0x00 0x05 0x01

STM32F411 返回：无返回

说明遥控发送的 flipDir = 0x01，即四轴接收到此命令后向前（FORWARD）翻滚。

至此，通信协议部分简单说明完毕。DOWN_RADIO 相关指令主要是在 radiolink.c 和 math_ui.c 中调用，DOWN_REMOTOR 指令主要在 remoter_ctrl.c 和 main_ui 中调用，其他 C 文件也有调用，大家可以结合程序源码解读。

2.4 关于 FreeRTOS

FreeRTOS 学习资料请移步：
http://www.openedv.com/forum.php?mod=viewthread&tid=88410&highlight=freertos

3. MiniFly 遥控器二次开发

3.1 开发环境搭建

我们 MiniFly 源码工程是基于 MDK5 开发，下面我们先安装这个软件。MDK5 安装源文件在 ATK-MiniFly 微型四轴资料中有提供，如下图 3.1.1 所示。

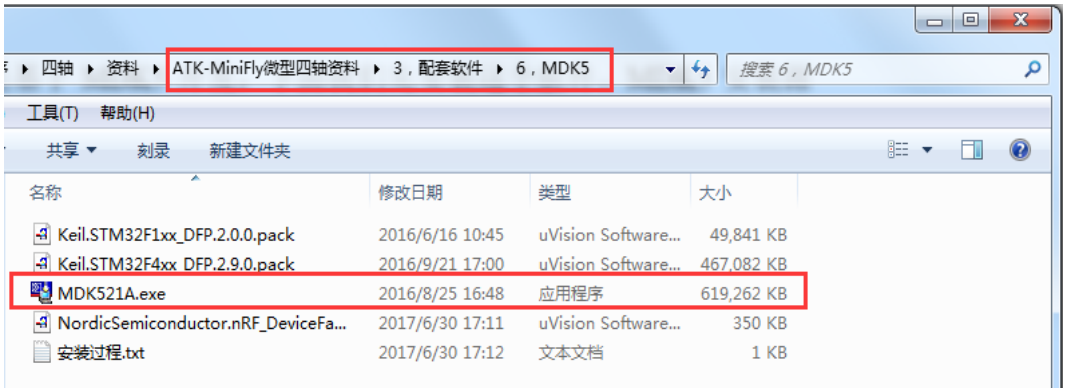

图 3.1.1 MDK 安装源文件目录

MDK 安装过程就不详细说明，安装前先看“安装过程.txt”。安装完 MDK 软件之后还需要安装芯片包（图 3.1.1 中.pack 文件），直接双击芯片包文件即可安装（先安装 MDK 软件）。所有都安装完成之后打开 MiniFly Remoter Firmware_F103 源码工程如下图 3.1.2 所示。

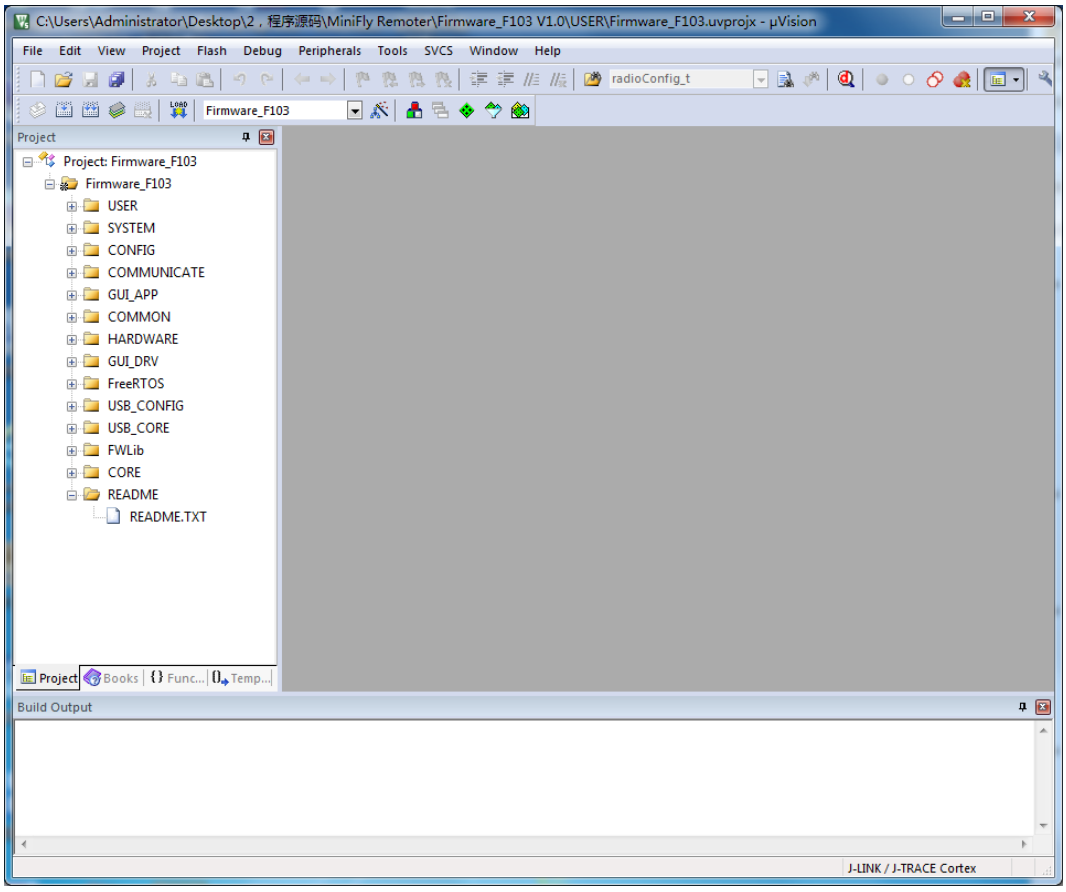

图 3.1.2 源码工程示图

3.2 固件下载和 USB 固件升级

3.2.1 固件下载

首先我们给下载器安装驱动。下载器驱动文件路径：“ATK-MiniFly 微型四轴资料\3，配套软件\4，下载器驱动”。驱动安装过程这里就不详细说明，安装完成后将 USB 线连接好下载器并插入电脑，然后打开设备管理器，在通用串行总线控制器中即可找到 J-Link driver，如下图 3.2.1 所示。

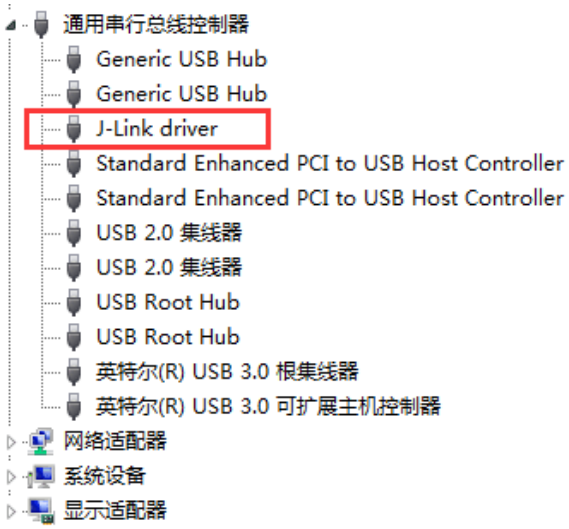

图 3.2.1 下载器驱动

然后将下载器连接到遥控器，下载器通过 6Pin 线连接到遥控器，如下图 3.2.2 所示。然后将下载器开关打到 STM32 处，如下图 3.2.3 所示。

注意：下载 STM32F411 和 STM32F103 芯片程序时将下载器开关打到 STM32 处；下载 NRF51822 芯片程序时下载器开关打到 NRF51xx 处。

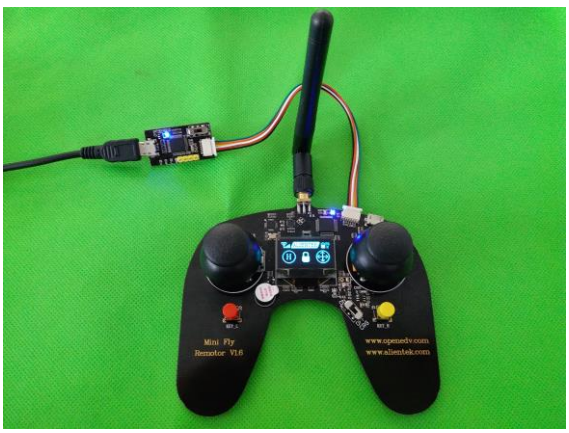

图 3.2.2 下载器连接图

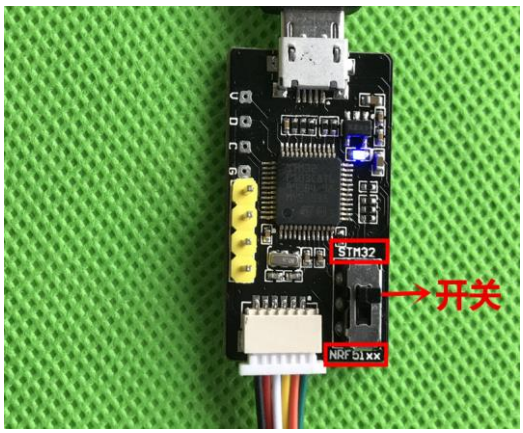

图 3.2.3 下载开关操作图

然后打开程序源码。驱动安装完成之后，我来看看程序源码文件。程序源码目录树如下图 3.2.4 所示。程序源码包括 MiniFly Master（主机）和 MiniFly Remoter（遥控器）两部分，其中 MiniFly Master 包含了 NRF51822 和 STM32F411 两个芯片的程序，MiniFly Remoter 包含了 STM32F103 的程序。三个芯片只有 NRF51822 没有 Bootloader 程序，Bootloader 程序主要用于固件跳转和 USB 固件升级。遥控器开机时，先运行 Bootloader 程序，判断当前 KEY_L 键是否按下，如果按下大于 3S 则留在 Bootloader 程序等待固件升级，否则跳转至固

件程序。

图 3.2.4 程序源码目录树

编译固件并下载。解压 Firmware_F103 V1.0.zip 并打开程序工程，点击编译，等待编译完成再点击下载。操作步骤如下图 3.2.5 所示。下载完成后如图 3.2.6 所示。所有程序源码下载都可以按照此方法执行，这里就不重复。

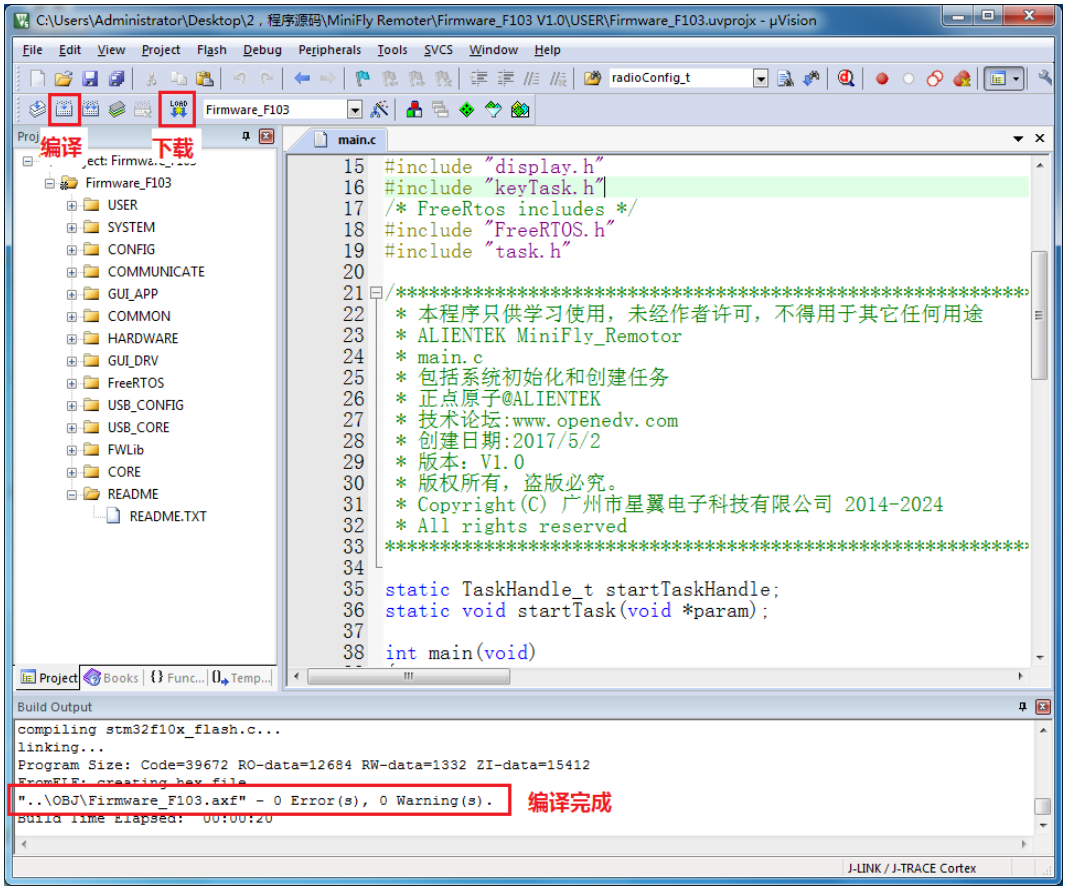

图 3.2.5 固件下载操作步骤示图

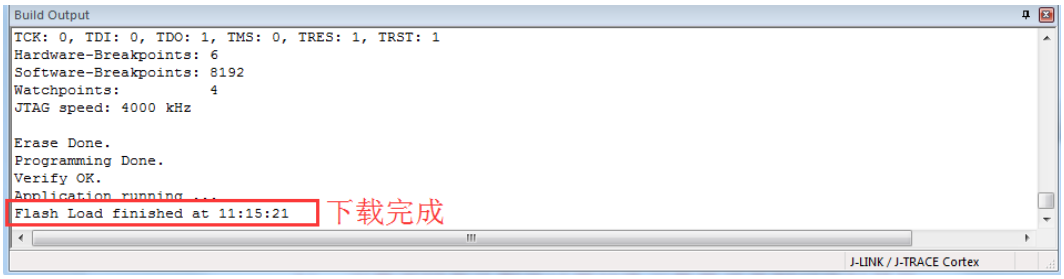

图 3.2.6 程序下载完成示图

提示：

a) 下载时，芯片必须是上电状态；

b) 下载不同芯片程序时要切换下载器开关到对应标号处；

c) 更多下载调试和 MDK 新建工程请参考我们 ALIENTEK 战舰开发板 A 光盘资料中的《STM32F1 开发指南》，其他开发板的开发指南均可。

3.2.2 USB 固件升级

第一步，安装 STM32 USB 虚拟串口驱动

STM32 USB 虚拟串口驱动路径：“ATK-MiniFly 微型四轴资料\3，配套软件\3，STM32 USB 虚拟串口驱动”。安装过程这里就不详细说明，安装完成后将 USB 线连接好四轴或遥控器插入电脑。电脑提示发现新硬件并安装驱动，如果安装成功了可以在设备管理器看到端口号，如下图 3.2.7 所示。如果安装失败了，请参考路径下的安装失败解决办法。

图 3.2.7 虚拟串口安装成功图示

第二步，进入 Bootloader 程序等待固件升级

安装好 STM32 USB 虚拟串口驱动后，将四轴或遥控器进入 Bootloader 程序模式等待固件升级。进入 Bootloader 后，将 USB 线连接好插入电脑，打开设备管理器找到端口号。

注意：Bootloader 程序等待固件升级超时时间为 60S，60S 内需执行升级否则会跳转至固件程序。

遥控器进入 Bootloader 方法：遥控器在关机状态长按 KEY_L 键开机，等待 LED1（蓝色）闪烁再松开按键即可。

四轴进入 Bootloader 方法：四轴在关机状态下长按电源键开机，等待 M2 灯先闪烁 M3 灯后闪烁再松开按键即可。

第三步，打开固件升级软件升级

固件升级软件（FirmwareUpgrade.exe）在 ATK-MiniFly 微型四轴资料\3，配套软件\2，固件升级路径下。打开软件，如下图 3.2.8 所示。按照图中步骤操作，step1:选择串口；step2:打开串口；step3:打开升级 bin 文件；step4:点击开始升级。升级完成后如下图 3.2.9 所示。

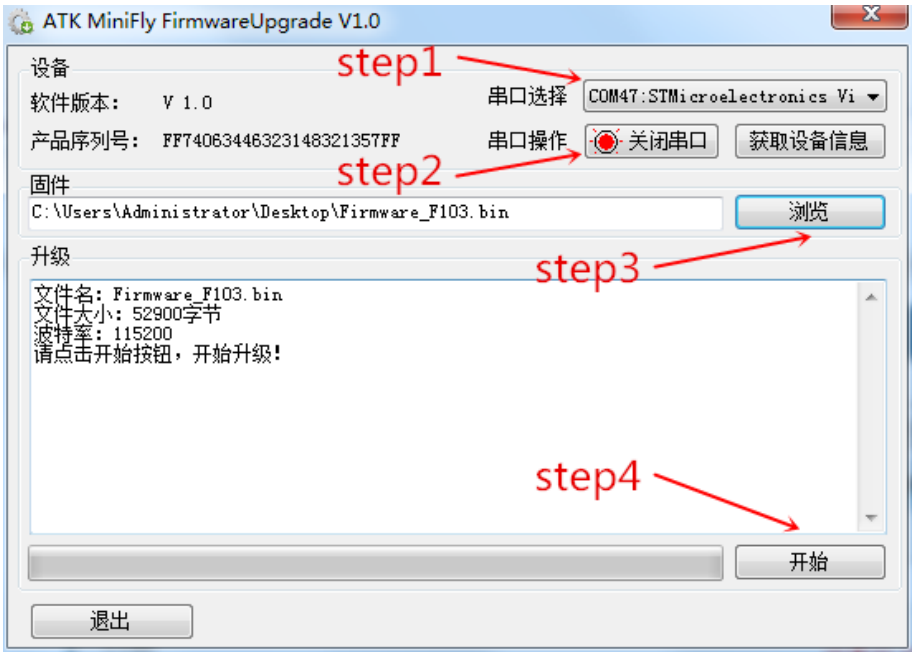

图 3.2.8 固件升级操作步骤示图

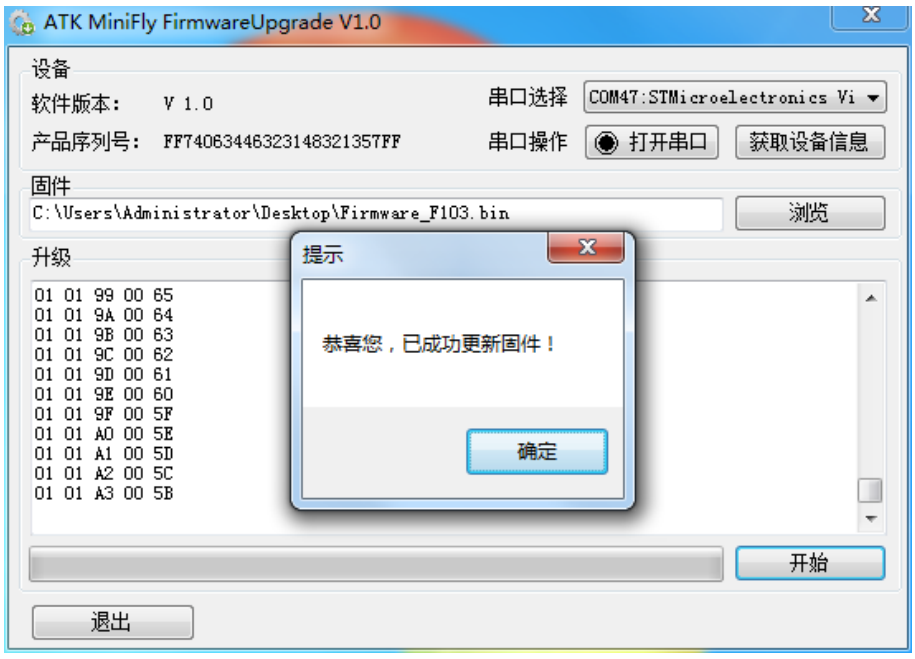

图 3.2.9 固件升级成功示图

提示：

1) Bootloader 程序等待固件升级超时时间为 60S，60S 内需点击开始升级否则跳转至固件；
2) 遥控器升级文件为 Firmware_F103.bin，四轴升级文件为 Firmware_F411.bin；
3) 升级前可先点击获取设备信息查看软件版本再决定要不要升级。

3.3 匿名科创地面站使用

我们 MiniFly 微型四轴兼容了功能强大的匿名科创地面站，该软件路径：“ATK-MiniFly 微型四轴资料\3，配套软件\1，匿名科创地面站”。匿名科创地面站支持多种通信方式，这里我们使用的是串口方式通信。下面介绍怎么使用匿名科创地面站查看飞控姿态，显示数据波形及 PID 调试。

第一步，将 USB 线连接四轴或遥控器

四轴和遥控器的固件都支持和匿名科创地面站通信、调试。使用 USB 线连接四轴或遥控器，在设备管理器中找到端口号。**注意**：使用遥控器和地面站通信时，只有四轴是开机状态才有数据上传，因为遥控器只做数据转发功能。

第二步，打开软件选择串口通信

图 3.3.1　串口通信设置图

打开软件后点击左边一列图标的程序设置，即可显示程序设置界面，如图 3.3.1 所示。我们选择通信连接方式为 COM 方式，然后选择端口号，设置波特率为 500000，最后点击右下角的打开连接，即可通信。当有数据上传时，RX:显示接收到数据的个数。

第三步，查看飞控状态

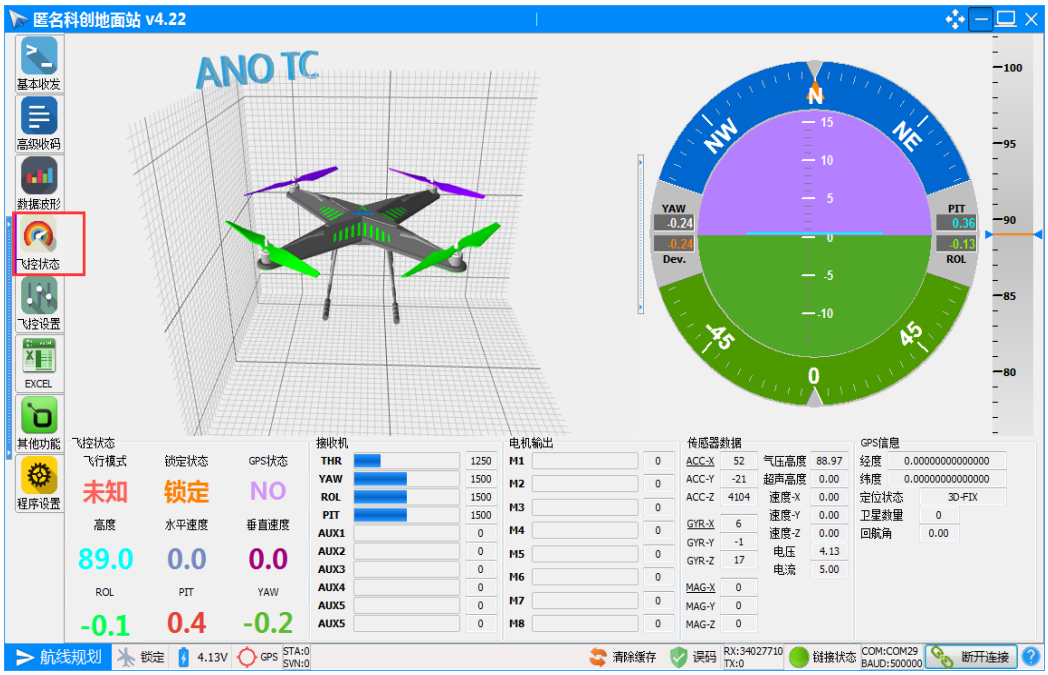

图 3.3.2　飞控状态图

点击左边一列图标的飞控状态即可显示飞控状态界面，如图 3.3.2 所示。飞控状态显示

主要包括当前四轴姿态（PIT\ROL\YAW）、接收机（遥控器控制数据）、电机输出（PWM）、传感器原始数据（ACC\GYR\MAG）、气压高度（单位 cm）、电池电压。飞行模式和 GPS 信息功能未使用。

第四步，查看数据波形

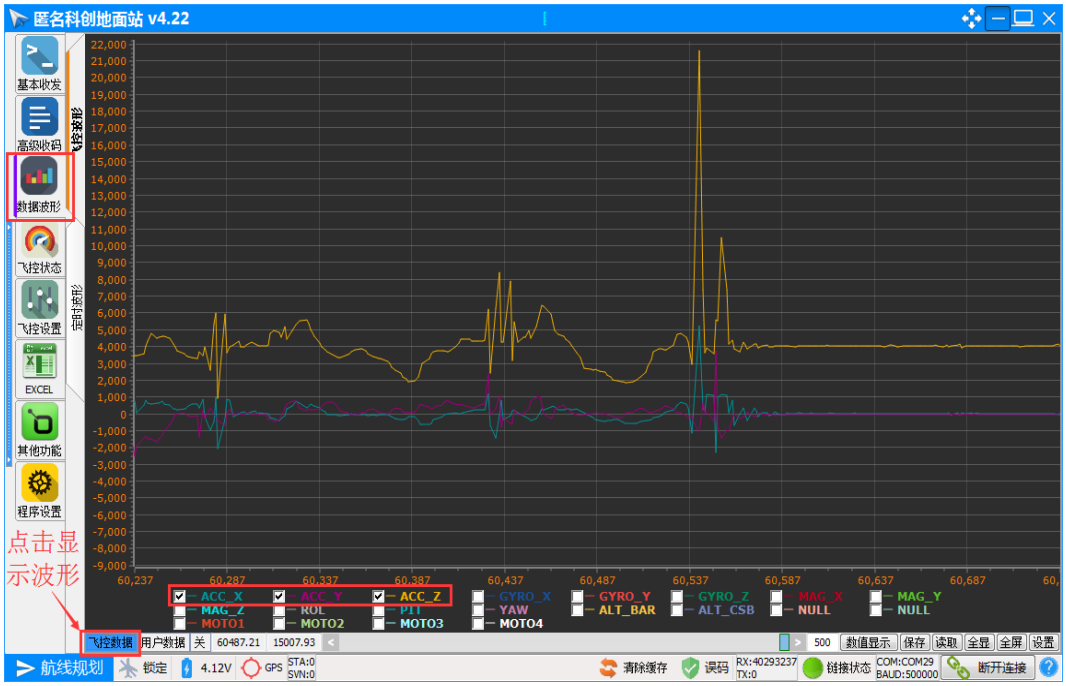

图 3.3.3　数据波形图

点击左边一列图标的数据波形即可显示数据波形界面，如图 3.3.3 所示。首先选中我们要显示的数据，再点击飞控数据即可显示数据波形。

第五步，PID 调试

图 3.3.4　PID 调试图

点击左边一列图标的飞控设置即可显示飞控设置界面，如图 3.3.4 所示。飞控设置包括 PID 设置、飞行模式设置、传感器校准，这里我们只使用了 PID 设置的功能。读取飞控：读

取四轴当前 PID 参数。写入飞控：将显示的所有 PID 参数写入到飞控。恢复默认值：将四轴 PID 参数恢复成出厂默认值。这里我们总共使用了 7 组 PID，分别是 roll 速率、pitch 速率、yaw 速率、roll 角度、pitch 角度、yaw 角度、高度位置。

注意：显示的 PID 数值是实际数值的 10 倍。由于地面站 PID 数据传输只能是 int16 类型不能浮点型，所以将实际数值乘以 10 再上传，同时写入时也除以了 10。

3.4 MiniFly 四轴 PID 调试

众所周知 PID 调试是四轴最难的一部分，也是最核心的一部分，PID 参数是否合理直接影响四轴飞行的效果。下面就介绍一下我们 MiniFly 四轴的 PID 调试方法。

第一步，修改 F411 固件代码

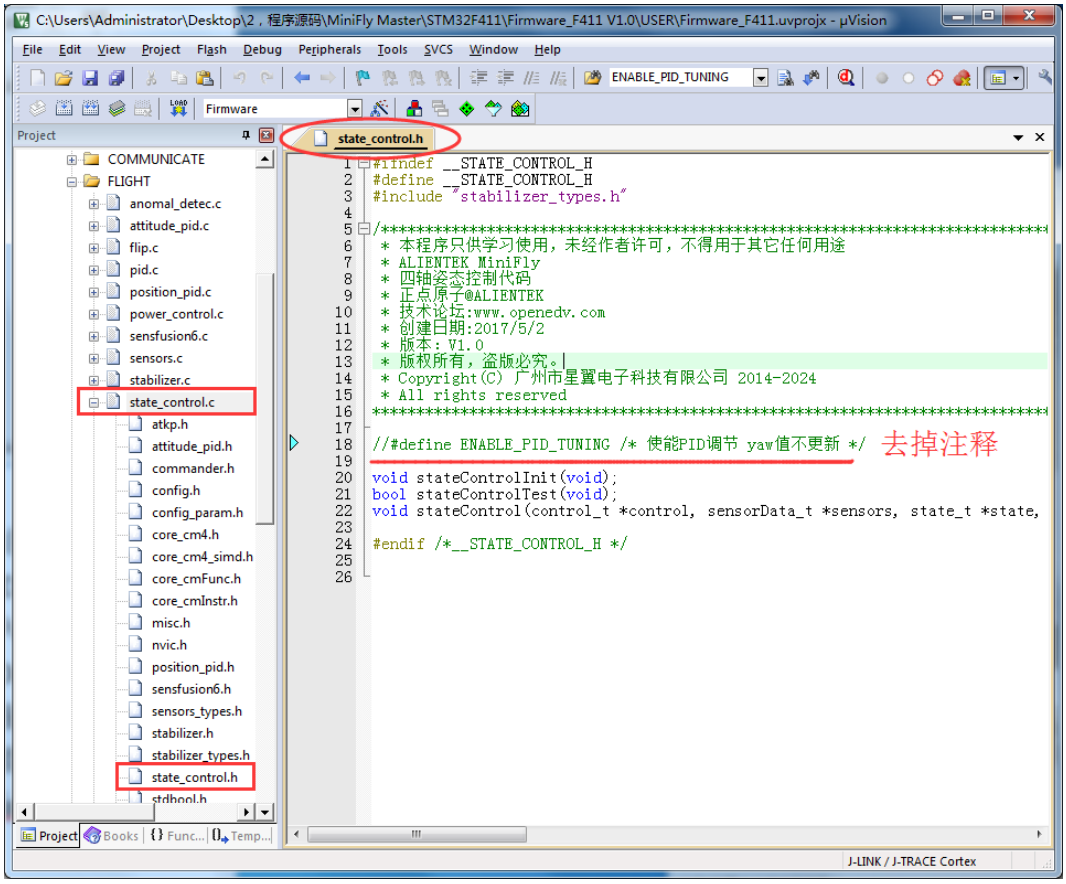

图 3.4.1 修改固件操作示图

为了方便 PID 调试，我们需要 state_control.h 中去掉 ENABLE_PID_TUNING 宏定义的注释，如上图 3.4.1 所示。这个宏定义的作用是使能后航向角(yaw)的值在姿态控制过程中一直是 0 不更新。因为 PID 调试一般都是先调 pitch 和 roll 方向，让四轴能自稳，自稳后再调节 yaw 方向的 PID，否则四轴会自旋不方便调试。按照图 4.4.1 操作，去掉注释然后编译固件并下载到四轴。

第二步，搭建四轴调试平台

图 4.4.2 MiniFly 四轴调试平台搭建示图

如上 7.4.2 所示，使用两条细绳一端拴住 MiniFly 四轴用于固定电池的排针，另一端拴住椅子的两端扶手，并保证 MiniFly 四轴可以 360 度旋转不会碰到椅子面。绳子拴住的排针最好是四轴中间位置的排针，排针不在机身中间，所以机身中间的排针不一定是排针中间的那一根。最好选择扶手较高的椅子，这样机身离椅子面足够的距离排风。注意：四轴开机后一直晃动陀螺仪是校准不通过的，需要使用物体卡住使其静止等待校准通过。

第三步，遥控器连接上位机

图 4.4.3 PID 初始设置示图

使用 USB 线将遥控器连接到上位机，点击右下角打开连接，然后并打开飞控设置，设

置自稳 ROL 和自稳 PIT 的 P 为 10，点击写入飞控，上方会显示“PID 数据 6”发送成功，如图 4.4.3 所示。串级 PID 调试一般先调内环（速率环），所以先将外环（角度环）PID 的 P 设置为 1，I 和 D 设置为 0。注意：上位机显示是放大了 10 倍，后面的调试值均是上位机显示的值。

第四步，调内环 pitch 和 roll 的 P

设置遥控器为手动模式，然后解锁长按 KEY_R 键进入调试界面。按照第三步操作后，我们给内环 ROL 速率和 PIT 速率的 P 同时设置一个值，然后推动油门至 50 左右观察并记录现象。下面是测试我们不同 P 的值和对应的现象：

P = 10 时，四轴乱晃无法固定于某一个角度，用手轻轻干扰四轴会绕绳子旋转，说明 P 值太小，力度不够。

P = 100 时，四轴晃动幅度减小但还是无法固定于某一个角度，用手轻轻干扰四轴还会绕绳子旋转，说明 P 值还是太小，力度不够。

P = 1000 时，四轴已经不晃动了，用手轻轻干扰明显感觉有回复力，说明 P 值已经慢慢趋向理想值。

P = 5000 时，四轴机身晃动严重已经震荡，说明 P 值已经过大，由此可确定理想的 P 值应该在 1000~5000 范围内。

P = 2500 时，四轴不晃动并且能固定在某一个角度了，机身也不震荡，用手去干扰能感觉到明显的回复力。说明 P 值已经很接近理想值了。

P = 3500 时，四轴机身轻微晃动说明已经轻微震荡，用手去干扰能感觉到很强的回复力。说明 P 值已经稍大于理想值了。

P = 3200 时，四轴没有晃动，用手去干扰能感觉到很强的回复力并有点晃动。说明 P 值就是我们的理想值了，晃动问题后期可以通过加点 D 抑制。

第五步，调内环 pitch 和 roll 的 D

根据第四步内环 P 已经调节好，但如果四轴受到干扰则会震荡，说明稳定性不好，这时我们就需要通过加 D 来增强稳定性。按照第四步的操作方法，设置 P 值为 3200，,测试不同的 D 值对应的现象：

D=10 时，用手去干扰四轴，能感觉到很强的回复力并有点晃动，说明 D 值有点小，抑制不了干扰。

D=100 时，用手去干扰四轴，能感觉没有晃动了但四轴需要长时间才能回复到水平，说明 D 值太大了，抑制了 P 的调整。

D=50 时，用手去干扰四轴，能感觉没有晃动了四轴也能较快回复到水平，说明 D 值接近理想值了。

就这样我们暂时把 D 值调好了，因为后面整机试飞时可能还会调节 D 值，这里先不用精确调节至某一个值。

第六步，调外环 pitch 和 roll 的 P

内环调好了，下面我们开始调外环。外环的主要作用是控制四轴姿态响应快慢，下面我们测试不同的 P 值，观察四轴的响应速度。按照前面操作方法，我们测试 P 值和现象如下：

P=10 时，方向摇杆往前打到最大，发现四轴慢慢倾斜，最终达到设定角度。响应速度不够快，说明 P 太小了。

P=100 时，方向摇杆往前打到最大，四轴瞬间倾斜了达到设定角度，而且力量很大。说明 P 值太大了。

P=50 时，方向摇杆往前打到最大，四轴较快的达到设定角度，不是很快也不是很慢。说明 P 值是我们理想值了。

第七步，调内环 yaw 的 PID

调节内环 yaw 的 PID 前需将第一步注释掉的 ENABLE_PID_TUNING 宏定义重新注释回来，并且编译下载至四轴。然后设置 yaw 外环 P 为 10，内环 P 设定为某一个值，慢慢推油门让四轴起飞至水平，推油门摇杆时不要向左或向右打到航向角。飞至水平后，推油门至 50，然后用手轻轻向左边或向右拨动四轴，感受四轴的回复力，并且听电机的声音。下面是我们设置不同内环 P 的值时对应的现象：

当 P=10 时，四轴几乎没有回复力，受干扰后左右摆晃，电机声音差异不大。说明 P 值太小了，没有修正的回复力。

当 P=100 时，四轴有一点点回复力，受干扰后左右摆晃，电机声音差异不大。说明 P 值还是太小。

当 P=1000 时，四轴有明显的回复力，受干扰后不会摆晃，电机声音差异很大。说明 P 值接近理想值了。

当 P=5000 时，四轴有明显的回复力，受干扰后两个电机停止转动，电机声音差异非常大。说明 P 值已经太大了。

跟以上调试现象和经验我们最终选定 P 值为 1200，然后再逐渐添加一点了积分 I，没有添加 D。内环的反馈是陀螺仪，航向角也是由陀螺仪积分而得，陀螺仪在四轴飞行过程有高频干扰，高频干扰使得 D 具有相反作用，所以我们没有添加 D。

第八步，调外环 yaw 的 PID

调节外环 yaw 的 P 和调节外环 pitch\roll 的 P 方法一样，主要是调节四轴的受控制的响应速度。给外环 yaw 的 P 设定一个值然后推油门和航向角，感受四轴旋转的快慢。这里我们选定 P 的值为 100。

第九步，手动试飞

经过以上步骤调试，四轴基本可以手动飞行了，松开绳子测试手动飞行。飞行过程中如果在无风条件下发现四轴晃动说明内环 pitch 和 roll 的 P 值大了，这时可以通过减小 P 或增加 D 来抑制晃动。如果发现航向角受外力干扰不能快速回复到原本状态，这可能是内环 yaw 的 P 太小了，需要增加 P。如果发现飞行过程会慢慢自旋，说明 yaw 的 I 太大了，需要减小 I 值。经过以上反复调试，我们最终确定我们 MiniFly 四轴 PID 参数如下图 4.4.4 所示：

图 4.4.4 MiniFly 四轴 PID 参数

第十步，调定高的 PID

定高 PID 调节是比较揪心的，调试过程我们按照正常的 PID 调试方法先调 P 再调 I 最后调节 D，反复调试发现根本行不通，四轴依旧很难定在一定高度范围内。后来经过分析，我们发现使用气压计定高，气压数据有很大的滞后性，并且四轴飞行过程垂直向下的风也会干扰气压读数。所以我们调试时不管怎么调节 P，出现的现象是要么慢慢的上升或慢慢的下降，要么快速上蹿下跳，根本不受控制。后来我们根据 PID 理论中 D 有抑制超调和干扰的作用，索性把 D 也先加上一点点，果然有效果，四轴上蹿下跳没那么厉害了。继续增加 D 四轴更加稳定了，当 D 增加到很大时，我们发现，推动油门四轴不怎么受控制了，说明 D 值太大了将 P 的作用都抑制了。反复调试后，我们最终选定 PID 的值为 210,0,600。

总结

PID 三项的的意义：P 是系统平衡的回复力； I 是消除误差，有辅助 P 的作用；D 是阻尼，抑制超调和干扰的作用。调试时一般先调节 P 找到临界震荡的 P 值，然后减小一点 P 值，增加 I 值消除静态误差，最后增加 D 值抑制干扰。要不要加 I 和 D 需根据实际情况而定。调试定高 PID 时，需要 P 和 D 同时调。PID 调试是比较繁琐的事情，需要耐心观察现象并分析原理。

4. 其他

1. 购买地址：

官方店铺：https://openedv.taobao.com/

2. 资料下载

下载地址：http://www.openedv.com/thread-105197-1-1.html

3. 技术支持

公司网址：www.alientek.com
技术论坛：www.openedv.com
传真：020-36773971
电话：020-38271790