



# ACFLY EDU 飞控用户手册

V2.0



广州博睿创新科技有限公司

[www.acfly.cn](http://www.acfly.cn)

## 0 阅读提示

### 0.1 使用建议

建议用户在 ACFLY 飞行控制的 QQ 交流群（购买后由官方发送邀请）下载或者 B 站（\*）观看教学视频和阅读《ACFLY EDU Ti 飞控代码》了解产品，使用《ACFLY EDU 飞控用户手册》可快速了解使用过程。

\*B 站：<https://space.bilibili.com/250212993>

产品对应资料有两种下载方式：

- (1) ACFLY 售后群-群文件
- (2) 百度网盘：[https://pan.baidu.com/s/1jh9w8LYkVDMt\\_XuVKbfJyg](https://pan.baidu.com/s/1jh9w8LYkVDMt_XuVKbfJyg) 提取码：acac

### 0.2 安全提示

1. 严禁电机上电带桨进行程序烧录，以免发生意外；
2. 严禁电机上电带桨插拔电调信号线，以免发生意外；
3. 严禁飞行器上电后用表笔直接对板子进行测量，容易导致单片机烧坏；
4. 产品一切源码和资料仅限于学习交流，禁止用于商业用途！违者必究！

### 0.3 正版注册

无人机飞行控制系统主要由电路板、固件和其他软件组成，ACFLY 主要产品已经具备自主知识产权，为了防止被直接抄板和拷贝代码，对飞控板均采用正版注册的防抄袭方式，**出厂默认已经注册**。

将飞控连接到电脑，飞控正常情况下会在**滴滴两声后，RGB 三色灯交替闪烁**，此种状态证明该飞控证明已经注册，可正常使用，并可在我的电脑—>右键—>管理—>设备管理器—>端口中查看飞控的 COM 口号。若飞控灯不亮或者亮其他颜色，请联系客服。

### 0.4 关注微信公众号

请使用移动设备的微信软件扫描以下二维码，关注【博睿开源飞控】微信公众号，实时获取产品更新信息以及商务合作信息。



博睿开源飞控

# 目录

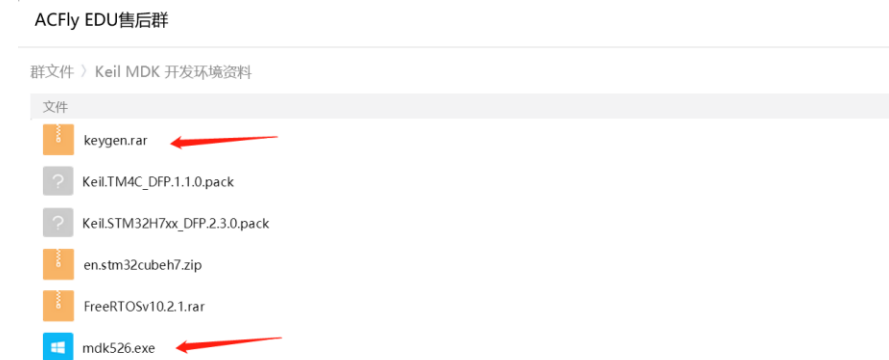
<b>0 阅读提示</b>	<b>1</b>
0.1 使用建议	1
0.2 安全提示	1
0.3 正版注册	1
0.4 关注微信公众号	1
<b>1 开发环境搭建</b>	<b>3</b>
1.1 编译环境（MDK）安装	3
1.2 编译环境设置	4
1.3 固件烧录	4
<b>2 硬件接口定义与规格</b>	<b>5</b>
<b>3 飞控供电</b>	<b>7</b>
<b>4 屏幕显示说明</b>	<b>8</b>
<b>5 飞控安装</b>	<b>9</b>
5.1 飞控固定	9
5.2 电机转向及顺序	9
5.3 连接接收机	9
<b>6 飞控初始设置及校准</b>	<b>10</b>
6.1 飞控上电初始化	10
6.2 遥控器校准	10
6.3 加速度计校准	11
6.4 磁罗盘校准	11
6.5 机体水平校准	11
<b>7 电调电机校准</b>	<b>12</b>
<b>8 飞控调参</b>	<b>13</b>
8.1 调参软件	13
8.2 参数分类	13
8.3 电池参数	14
8.4 姿态控制参数	14
<b>9 代码框架</b>	<b>15</b>
9.1 代码总体布局	15
9.2 代码执行流程	17
9.3 模式的执行	18
<b>10 代码接口手册</b>	<b>19</b>
10.1 传感器接口	19
10.2 接收机接口	21
<b>11 代码二次开发教程</b>	<b>22</b>
<b>12 版本更新日志</b>	<b>22</b>

# 1 开发环境搭建

## 1.1 编译环境（MDK）安装

### （1）MDK 开发软件安装

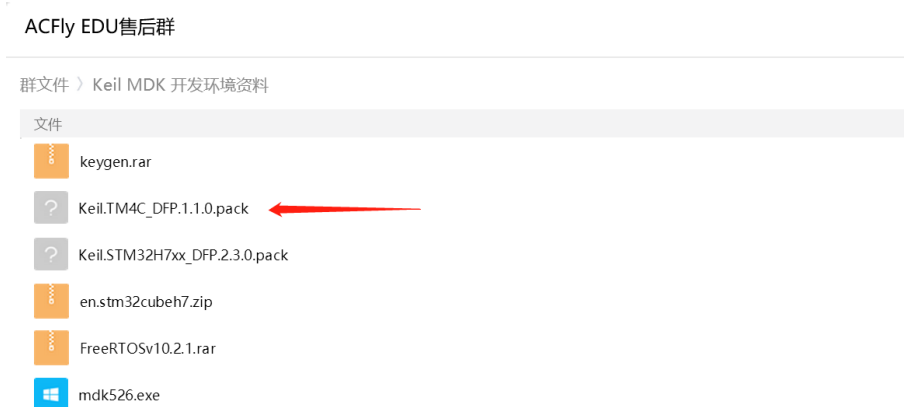
到售后群下载或者在 keil 官网 <http://www.keil.com/> 的 Download 页面自行下载双击安装 MDK5.26:



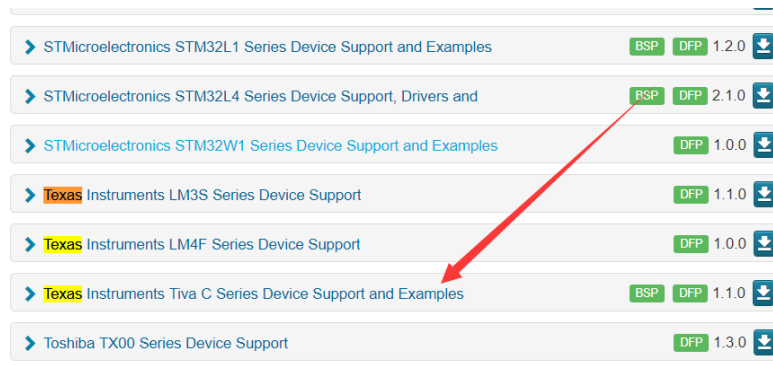
选自己喜欢的目录安装完成后，使用 keygen 注册机进行破解（File->License Manger 中输入序列号，具体怎么破解可以查百度）。

### （2）芯片支持包安装

在售后群中下载支持包进行安装：

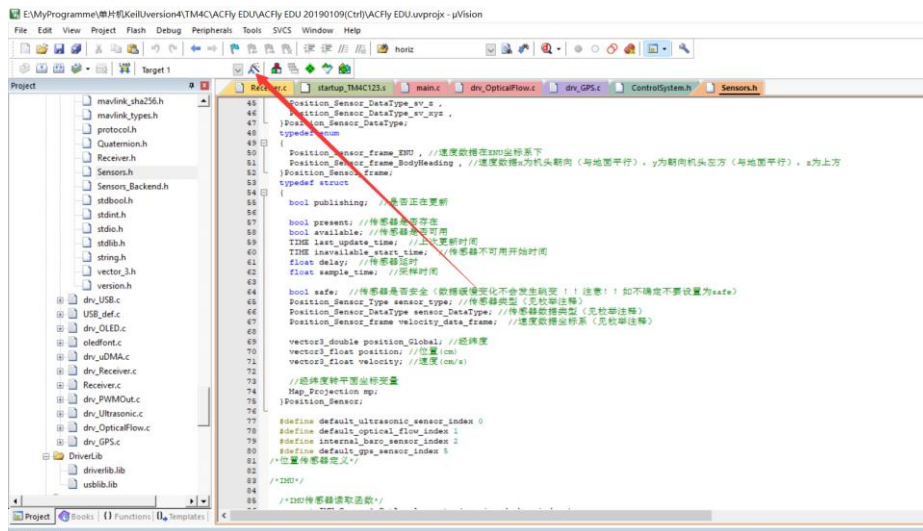


或在 keil 官网 <http://www.keil.com/pack> 下载下图中的最新版本软件支持包并安装

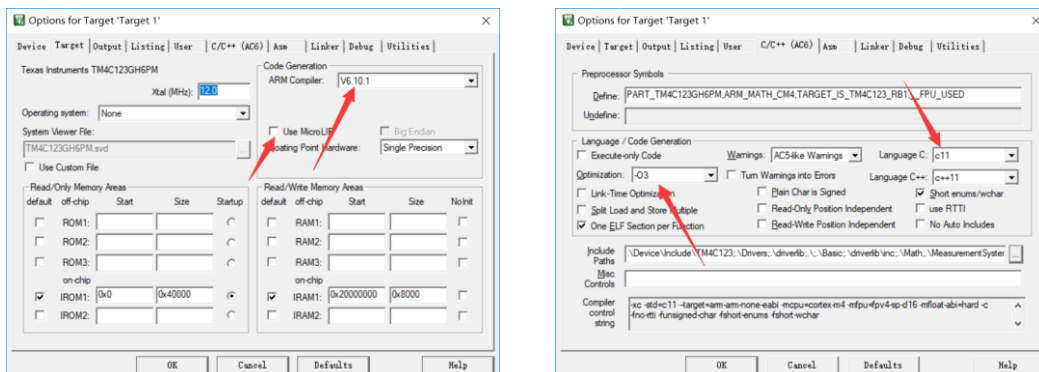


## 1.2 编译环境设置

在 MDK 界面下点击下列按钮进入工程配置选项：



在 Target 选项卡中选择 V6 开头的编译器（如果没有，可以安装 MDK 较新的版本或者前往 <https://developer.arm.com/products/software-development-tools/compilers/arm-compiler/downloads/version-6> 网页单独下载编译器并安装进 MDK），并取消 Use MicroLib 复选框；在 C/C++(AC6)选项卡中进行下图中的配置（由于这款单片机内存较小，不选择编译优化可能导致栈越界）：

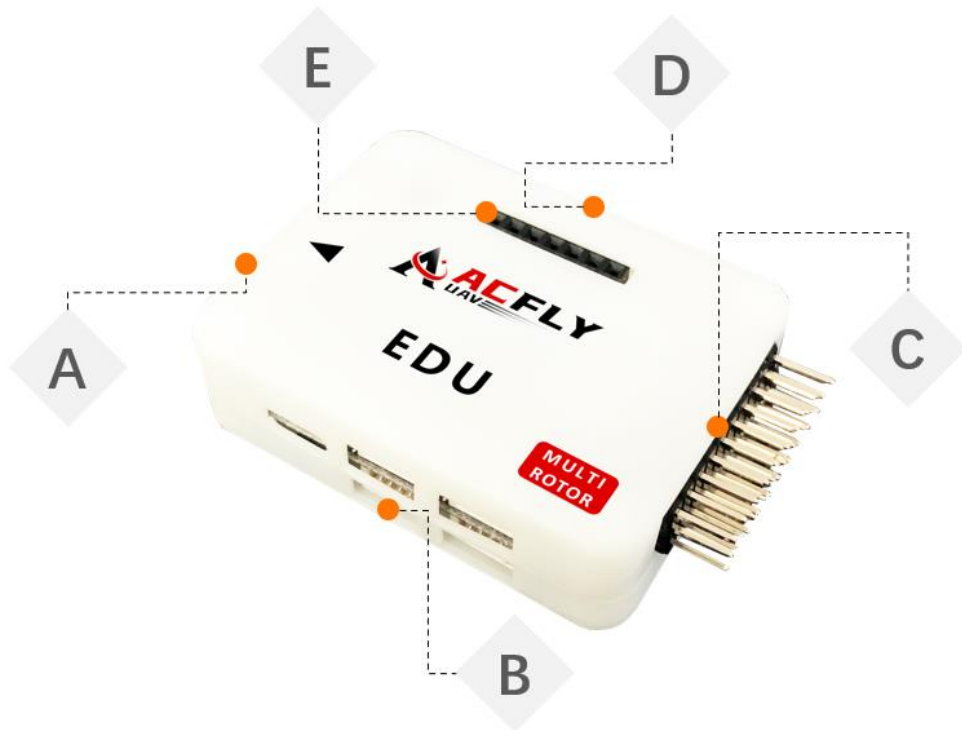


## 1.3 固件烧录






代码烧录需要连接飞控 SWD 接口，接口定义如下图所示。将此接口与 ST-Link（推荐）、JLink 等下载器的对应接口连接，然后接通电源即可下载。若接上时电源灯不亮，则将所有外设包括接收机接口都拔掉，只保留 SWD 接口的接线。

## 2 硬件接口定义与规格

### (1) 接口定义



**注意：飞控朝上放置**

序号	图示	接口定义									
A		UART0: TX RX GND 5V					Ultrasonic: Echo Triger GND 5V				
		IIC1: 5V GND SCL SDA									
B		UART1: TX RX GND 5V					PowerADC: Bat Cur GND 5V				
		SWD: 3.3V GND SWCLK SWDIO					Power: 5V GND GND 5V				
C		PWM1	PWM2	PWM3	PWM4	PWM5	PWM6	PWM7	PWM8	PPM	SBUS
		S	S	S	S	S	S	S	S	S	S
		+	+	+	+	+	+	+	+	+	+
		GND	GND	GND	GND	GND	GND	GND	GND	GND	GND
D		UART2: TX RX GND 5V				UART7: TX RX GND 5V			UART3: TX RX GND 5V		
E		显示屏接口: GND 3.3V D0 D1 Rst DC GND									

## (2) 外设接口

- 超声波：Ultrasonic 接口
- 光流串口：Uart1 接口
- GPS 串口：Uart0 接口
- 外置罗盘：IIC1 接口
- OpenMV 串口：Uart3 接口
- 激光串口：Uart7 接口

## (3) 飞控尺寸重量

- 长 53mm，宽 40mm，高 15mm
- 重 23 克



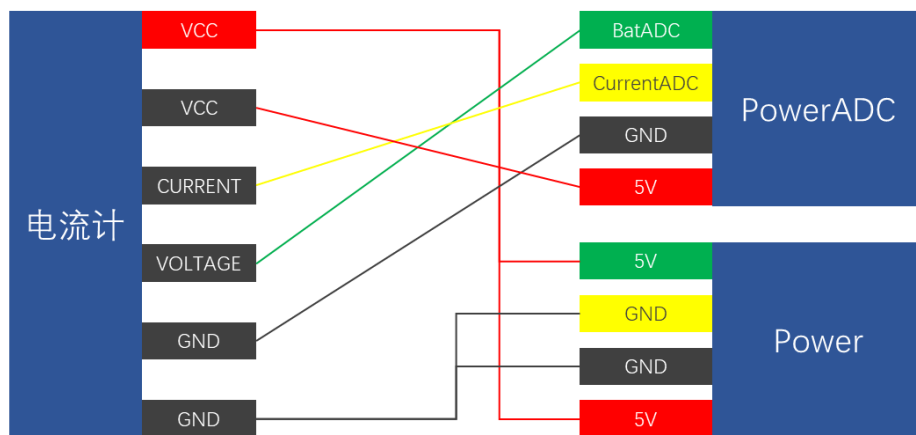
### 3 飞控供电

ACFLY EDU 飞控支持双冗余供电，供电电压 5V，分别是 powerADC 端口和 Power 端口。以下面的电流计为例，电流计共有 6 个引脚，可以将其一路 5V 和一路 GND 接到飞控的 Power 端口，而剩余的 5V、GND、CURRENT、VOLTAGE 引脚则对应接到飞控的 powerADC 接口。可直接购买使用 ACFLY 电流计，免焊线。



Pin	Signal	Volt
1 (red)	VCC	+5V
2 (blk)	VCC	+5V
3 (blk)	CURRENT	+3.3V
4 (blk)	VOLTAGE	+3.3V
5 (blk)	GND	GND
6 (blk)	GND	GND

电流计引脚说明



电流计与飞控的接口 GH1.25 4pin 接线方式



## 4 屏幕显示说明



参数	说明							
<b>Sensors:</b>	0	1	2	3	4	5	6	7
<b>Pos:</b>	超声波	未定义	内部 气压计	外部 气压计	未定义	GPS	光流	未定义
<b>Mag:</b>	外置罗 盘 1	外置罗 盘 2	内置罗 盘		<b>RC:</b>	接收机		<b>Bat:</b> 电压
				<b>FP:</b>	定位是否成功			
<b>Rol:</b>	Roll 的简写，横滚角度，单位度				<b>Pit:</b>	Pitch 的简写，俯仰角度，单位度		
<b>Yaw:</b>	偏航角度，单位度				<b>Vz:</b>	垂直方向速度，单位 cm/s		

\*符号说明：

✓：正常，已使用（GPS 表示定位成功）

·：已识别

×：没连接/没识别/不正常

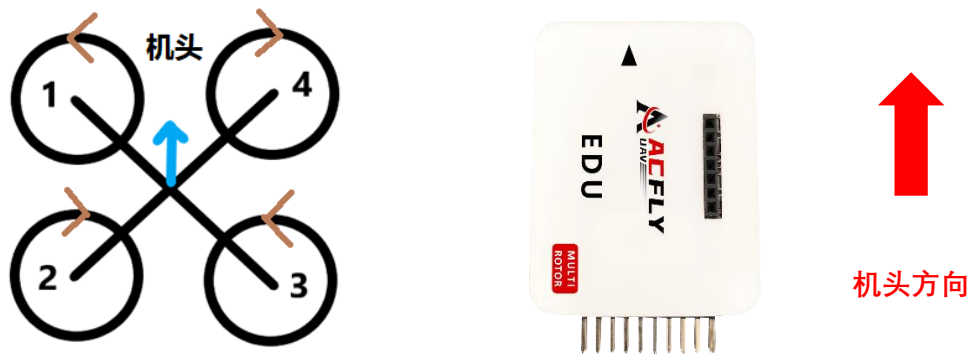
## 5 飞控安装

### 5.1 飞控固定

- 飞控支持免减震安装，需紧固在飞行器上，可使用 3M 胶固定飞控四个角即可，切勿只粘住飞控中间，飞控松动会影响飞行稳定性。

### 5.2 电机转向及顺序

- 以四旋翼为例，红色箭头为机头方向，飞控需按照下右图中的方向放置。飞控左上角电机为 1 号电机，六旋翼八旋翼类似，按逆时针顺序分别为 1、2、3、4。1 和 3 号电机逆时针旋转，2 和 4 号电机顺时针旋转。将飞控按上述方向紧固在飞行器上（需固定飞控四个角）。
- 分别将 1、2、3、4...号电机所连接的电调的信号线分别插在飞控板的 M1-M8 口。



### 5.3 连接接收机

- 如使用 SBUS 接收机，将接收机的 SBUS 信号线（3 线）连接至飞控的 SBUS 口。
- 如使用 PPM 接收机，将接收机的 PPM 信号线（3 线）连接至飞控的 PPM 口。

## 6 飞控初始设置及校准

### 6.1 飞控上电初始化

飞控上电后会进行初始化（校准等），此时飞控处于 M00\_init 模式下等待初始化完成，状态灯三快一慢闪烁。待状态灯三色慢闪变化并发出滴滴后代表自检完成（进入 M01\_Ground）模式。

飞控每次上电后会先进行陀螺校准，此时飞控会检测飞控是否在运动，如果飞控处于运动状态将无法通过陀螺校准。因此上电后请将飞控先保持 10 秒左右静止。

陀螺校准过后板子可以缓慢移动，飞控将进行位置加速度的初始对准，此过程比较长可能需要半分钟至一分钟（校准加速度计后此过程可以缩短）。

### 6.2 遥控器校准

此飞控要求遥控器至少具有 6 个通道，包含 4 个摇杆和 2 个按钮，最多支持 8 个通道，校准的第一个按钮用来切换定位和自动飞行模式。校准的第二个按钮支持一键起飞和航点自动飞行功能。

首先等待 4.1 中的飞控初始化完成，飞控在 M01\_Ground 模式下，飞控状态灯切换至三色慢闪。

将飞控连接到电脑，打开 ACFLY EDU 调参软件，在 COM 选择中找到飞控端口号，点击 open。

(1) 打开遥控器（已将接收机连接至飞控，参考 3.3）

部分遥控器需要提前设置好遥控器的 PPM/SBUS 模式，根据需要设置 2 个按钮（即辅助通道）

(2) 将四个摇杆通道回中，所设置的 2 个按钮拨到 2 档或 3 档（数字最大档）位置。在调参软件中点击“校准遥控器”按钮。飞控进入 M10\_RCCalib 遥控器校准模式，此时红灯常亮。



(3) 拨动一下任意摇杆或档位然后迅速回中，飞控状态灯变为蓝色渐变

(4) 等待蓝灯闪烁两次（蜂鸣器哔），表示已记录所有摇杆初始位置

(5) 按如下顺序操作遥控器：油门最下（哔）—>偏航最左（哔）—>俯仰最下（哔）—>横滚最左（哔）—>油门最上（哔）—>偏航最右（哔）—>俯仰最上（哔）—>横滚最右（哔）—>杆回中—>按钮 1 拨向 1 挡位置（哔）—>按钮 1 拨回原位—>按钮 2 拨向 1 挡位置（哔）—>按钮 2 拨回原位—>按钮 3 拨向 1 挡位置（哔）（或把油门拉低完成较准）—>按钮 3 拨回原位—>按钮 4 拨向 1 挡位置（哔）（或把油门拉低完成较准）—>按钮 4 拨回原位—>完成（绿灯闪哔哔）

错误情况：

- 红灯闪哔—长叫并退出模式回到 M01：遥控器或接收机断开
- 完成校准时红灯闪哔—长叫：摇杆通道的最高油门、最低油门到摇杆中间的行程不一致。

## 6.3 加速度计校准

首先等待 4.1 中的飞控初始化完成，飞控在 M01\_Ground 模式下，飞控状态灯切换至三色慢闪。

遥控器油门最小，偏航最左，俯仰最下，横滚最左，两秒左右将进入 M12\_ACCCalib 加速度校准模式。

将装好飞控的无人机分别摆放六个面（不用按顺序，差不多水平即可）静止。校准时飞控蓝灯由暗变亮，完成一个面后会闪烁然后变红。

飞控灯红色表示当前面已经校准或者飞控在移动无法校准。

## 6.4 磁罗盘校准

首先等待 4.1 中的飞控初始化完成，飞控在 M01\_Ground 模式下，飞控状态灯切换至三色慢闪。

遥控器油门最小，偏航最左，俯仰最上，横滚最左。两秒左右将进入 M13\_MagCalib 磁罗盘校准模式。

旋转无人机尽量多的三维角度，旋转时飞控指示灯会由暗变亮指示当前进度。完成后飞控指示灯闪烁表示校准完成。

飞控灯红色表示飞控没有在旋转无法校准。

## 6.5 机体水平校准

**此步骤非常关键，务必进行校准！**

首先等待 4.1 中的飞控初始化完成，飞控在 M01\_Ground 模式下，飞控状态灯切换至三色慢闪。

将无人机机体放平（是机体不是飞控）并静止不动。飞控遥控器油门最小，偏航最左，俯仰回中，横滚最左，两秒左右将进入 M15\_HorizontalCalib 水平校准模式。

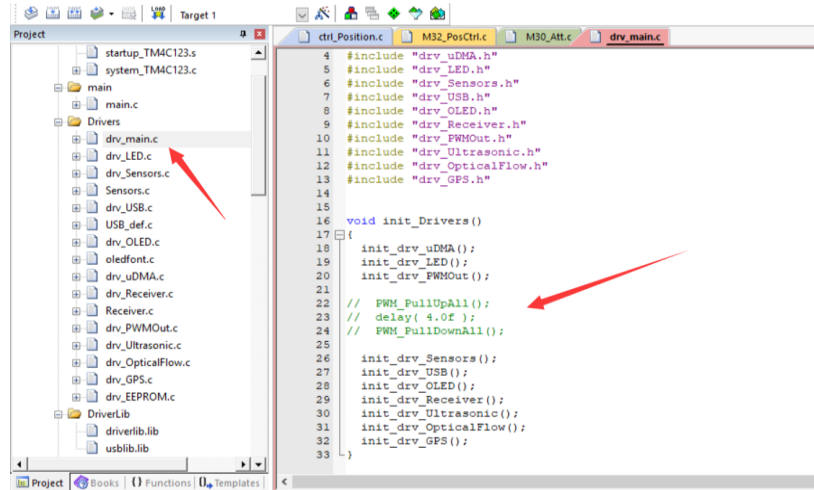
飞控指示灯会由暗变亮指示当前进度，完成后飞控指示灯闪烁表示校准完成

## 7 电调电机校准

### (1) 电调行程校准

**首先将桨拆下，以免发生意外！天行者电调必须校准，不然可能堵转！**

在 drv\_main 文件中，将下面几行取消注释：



编译后将代码烧进飞控，插电池给飞控和电调电机重新上电，之后会听到电调嘀嘀-嘀的声音，表示校准完成。

**校准完成后将上述代码重新注释、编译并烧进飞控，以免之后发生意外！**

### (2) 电机起转油门设置

#### ■ 参数名：MT\_STThr

**天行者电调必须认真设置，不然可能堵转！**

乐天电调为 10，天行者电调为 15 左右。

此值设置后，四个电机应当开始同时旋转（倾斜飞机不会出现有的在转有的不转的情况）

### (3) 电机推力非线性补偿系数设置

#### ■ 非线性补偿系数参数名：MT\_NonlinF

#### ■ 电机满油门推力参数名：MT\_FullThrR

补偿电机推力不是线性的系数。非线性补偿系数为 0 时表示不修正，为 1 为最大修正，默认为 0.65。电机满油门推力参数范围 0.65-1，1 时表示最大油门时就是最大推力，0.9 时表示 90%油门时已是最大推力（后面的 10%无效）

计算：

假设油门推力方程为： $F = kx^2 + (1 - a)x$ ，其中 F 为推力，x 为油门（0-1），a 为系数。

又假设满推力时油门为 m（油门范围 0-1）

即有： $km^2 + (1 - a)m = 1$

得： $k = \frac{1 - (1 - a)m}{m^2}$

解方程组  $kx^2 + (1 - a)x = out$ ，其中 out 为修正前的油门输出，得到的 x 即为线性修正后的油门输出。

## 8 飞控调参

### 8.1 调参软件

- 本飞控使用 Mavlink 协议，支持使用 Mavlink 的参数协议进行调参。

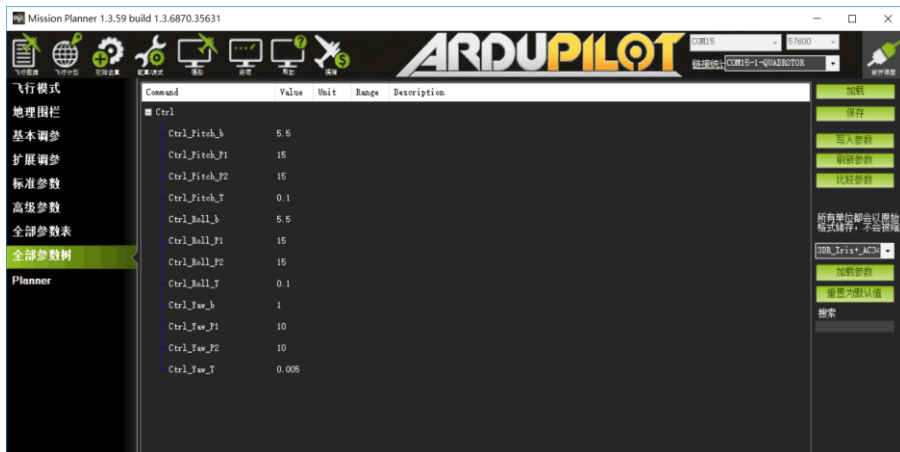
#### (1) ACFLY 地面站

- 打开 ACFLY 地面站 (可到售后群下载)，通过 USB 将飞控连接到电脑。
- 点击配置-参数表-可对相应参数进行修改。



#### (2) 支持 Mavlink 协议的开源地面站

以 Mission Planner 为例，将飞控连接至电脑，右上角选择 COM 口号，点击连接，地面站将获取飞控的参数列表。（注意：请在代码的 main 文件里将 init\_debug 注释掉或在 Debug 文件里不要往此端口写入非 Mavlink 的调试数据，不然将导致 Mavlink 误码过多地面站不识别）



在配置调试—>参数表中修改参数，然后点击右侧的写入参数即可将参数写入飞控。

### 8.2 参数分类

不同类别参数以开头分组，如 AC\_开头为姿态控制参数组，MT\_开头为电机参数组。

- AC---Attitude Control 姿态控制参数：包括 Roll、Pitch、Yaw 三轴感度，增益等。
- MT---Motor 电机参数：包括电机惯性时间常数，非线性系数等。
- BAT---Battery 电池参数：包括标准电池电压等。
- UAV---飞行器参数：包括飞行器类型等。

## 8.3 电池参数

### (1) 电池标准电压

参数名: BAT\_STVoltage

飞行器使用电池的标准电压, 如 3s 电池可设置为 11.6v。

此值用于在电压检测可用时, 动态调整姿态控制器 b 参数。

比如调参设置 b 参数为 10, 标准电压为 10v, 当测量到电池电压为 15v 时会动态改变 b 值为  $10 \times 15v / 10v = 15$ 。

### (2) 电池电压 ADC 采样放大倍数

参数名: BAT\_VADCMag

电压采集模块的分压倍数。

比如电压采集模块将电池电压分压到 1/10 给飞控采样, 此值应设置为 10。

## 8.4 姿态控制参数

### (1) 参数类型

姿态参数分为四类:

- b 参数 (参数名: AC\_Roll\_b、AC\_Pitch\_b、AC\_Yaw\_b): 飞行器控制对象的增益, 飞机力气越大此参数越大。参数过小飞行器会高频振荡发抖, 过大将控制不住 (打杆软绵绵没力)。**主要调此参数就行。**
- T 参数 (参数名: MT\_T): 飞行器电机的惯性时间常数。飞机桨加速至期望值的时间越长, 此参数越大。此参数过小飞行器会高频振荡发抖。**针对特定机型, 此参数需微调。**此参数越小 (电机加速快), 抗扰性能越好, 此参数太小会导致 b 怎么调都会有震荡现象。此参数大不会震荡但是抗扰性能会打折扣 (适中就行, 没必要追求太强抗扰)
- TD4P 参数 (参数名: AC\_Roll\_TD4Pn、AC\_Pitch\_TD4Pn、AC\_Yaw\_TD4Pn n=1...4): 前馈跟随速度 (打杆快慢), TD4 滤波器跟随打杆指令的增益 P1...P4。此参数越大打杆跟随越快。**可调节此参数来调节手感。**
- P 参数 (参数名: AC\_Roll\_Pn、AC\_Pitch\_Pn、AC\_Yaw\_Pn n=1...4): 第 1 到 4 层反馈环的反馈增益。此参数越大反馈增益越大。**不建议新手调节, 调了用处也不大。**

### (2) b、T 参数调节

以下参数标准电压: 3s=11.6v、4s=15.2v、6s=22.8v

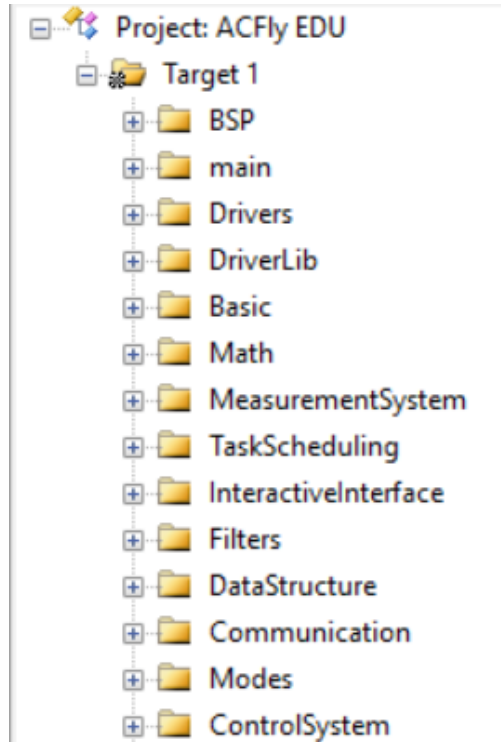
	F450+U22	F450+dji2	精灵 3 机架	dh600+疯狂	Tarot X6+	f550 六轴	f330+朗宇
配置	16 800kv 电机 +1147 桨 +4s	312 940kv 电机 +9450 桨 +4s	+dji2312 940kv 电机 +9450 自锁 桨+3s	4114 400kv 电机+飞越 1555 折叠桨 +6s	朗宇 4108 320kv 电 机+1855 碳桨 +6s	+dji 2312 电机 +9450 自 锁桨 +4s	x2212 1400kv 电 机+8038 桨 +3s
T	0.1	0.1	0.1	0.05	0.15	0.1	0.1
Roll Pitch	b=5.5	b=7.5	b=8.5	b=5.5	b=1.2	b=5.5	b=7.5
Yaw	b=1.0	b=1.0	b=2.0	b=1.0	b=0.8	b=1.0 T=0.005	b=1.0



## 9 代码框架

### 9.1 代码总体布局

本飞控代码已经分组在 14 个大类里，如图所示。建议看的部分：（3）驱动中的 Sensors 接口及 Receiver 接口；（5）Basic 中的时间实现；（7）解算系统中的解算系统接口；（8）任务调度器接口；（13）模式中的飞行模式；（14）控制系统中的控制系统接口。



#### （1）BSP（板级支持包）

板级支持文件，最底层的库，不用看基本不用修改（可以修改 startup.s 里面的堆栈设置）。

#### （2）Main（主函数文件）

主函数包括：

- 初始化所有需要用到外设；
- 进入 STS 任务调度器（while 循环执行任务）
- 错误中断拉低所有输出

#### （3）Driver（驱动）

驱动包含：

- 外设的初始化配置（drv\_开头文件）
- 传感器接口，包括（建议细看，二次开发必备）：
  - Sensors.c：传感器接口实现函数
  - Sensors.h：传感器读取接口函数声明，建议细看
  - Sensors\_Backend.h：传感器注册、更新接口函数声明，建议细看
- 接收机接口，包括：
  - Receiver.c：接收机接口实现函数



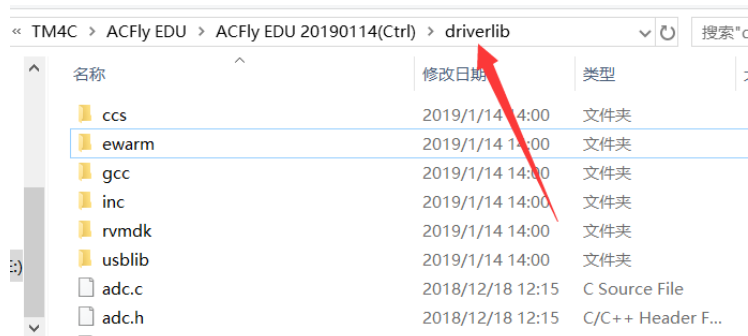
- Receiver.h: 接收机读取接口函数声明, **建议细看**

其中, drv\_Sensors.c 里面执行 IMU 传感器读取操作并写入 Sensors 接口, 然后通过挂起一个不用的中断的方式进入解算及控制中断。

```
598
599
600
}
//挂起解算控制任务中断
NVIC_SetPendingIRQ( I2C3_IRQn );
```

#### (4) DriverLib (TI 的库)

有需要自己写驱动的, 可以在下面目录下查看库函数的实现及说明:



#### (5) Basic (基本)

Basic.c 里面初始化 systic 定时器用于计时, 实现了 TIME 结构体用于时间计算, 其他部分程序所有时间相关操作都是基于 TIME, **建议细看**。

Configurations.c 里有 EEPROM 的读取保存操作, 用于保存记录参数等, **可以不看**。

#### (6) Math (数学库)

包含四元数、三维向量运算, 以及一些简单的数学运算, 重力等常量的定义。

#### (7) MeasurementSystem (解算系统)

姿态解算及位置解算。

**建议细看解算系统接口 MeasurementSystem.h**, 包含解算结果的获取函数声明及使用说明。

#### (8) TaskScheduling (任务调度器)

简易任务调度器, 就是在主循环刷任务。

**建议细看 STS.h**, 包含任务调度器的函数声明及使用说明。

#### (9) InteractiveInterface (用户交互接口)

目前仅包含 LED 相关操作

#### (10) Filters (滤波器)

包含巴特沃斯低通滤波器、TD4 非线性滤波器、位置估计卡尔曼滤波器的实现。

#### (11) DataStructure (数据结构)

包含环形缓冲区的实现。

#### (12) Communic (通讯)

包含 Mavlink 库、调试通讯文件 Debug.c、通用端口交互文件 Commulink.c(驱动程序可通过 Commulink.h 里的函数注册端口成为通用端口用于 mavlink 等标准通讯)

### (13) Modes (模式)

建议细看飞行模式！二次开发必备

- 0-9 号为非飞行非校准的其他模式
- 10-19 号为校准模式
- 30-39 号为飞行模式
- M00 为初始化模式，等待解算系统初始化完成。然后进入 M01 地面模式。
- M01 下可通过遥控或上位机命令进入其他校准及飞行模式。

### (14) ControlSystem (控制系统)

建议细看 ControlSystem.h！二次开发必备

ControlSystem.h 包含控制系统的 API 接口。Ctrl\_Attitude 和 Ctrl\_Position 分别为姿态和位置控制器。

## 9.2 代码执行流程

### (1) 初始化及任务调度

Main 函数中首先调用 init 开头的函数进行初始化。

初始化函数中，会调用 STS 任务调度接口将需要在主循环里执行的函数加入到任务调度列表。

STS\_Run 函数判断任务调度列表中的任务是否需要被执行，是就执行。

```
40 ~
41 int main()
42 {
43     init_Basic();
44     init_drv_EEPROM();
45     init_MS();
46     init_ControlSystem();
47     init_Drivers();
48
49     init_Configurations();
50
51     init_Modes();
52     init_CommuLink();
53
54     init_Debug();
55     //while(1);
56     STS_Run();
57 }
```

### (2) 解算及控制任务

只有解算及控制是在中断中执行的。

在 drv\_Sensors.c 文件中，每次获取到传感器数据后，会通过挂起一个不用的中断的方式进入解算及控制中断。

```
434 //开启一个不用的外设中断
435 //用于在中断中运行解算及控制任务
436 IntPrioritySet(INT_I2C3, INT_PRIO_7);
437 I2CIntRegister( I2C3_BASE, MainMCHandler );
438 IntEnable( INT_I2C3 );

598 }
599 //挂起解算控制任务中断
600 NVIC_SetPendingIRQ( I2C3_IRQn );
```

```
784  
785 //主解算控制任务中断  
786 static void MainMCHandler()  
787 {  
788     NVIC_ClearPendingIRQ( I2C3_IRQn );  
789  
790     MS_main();  
791     static uint16_t Ctrl_Counter = 0;  
792     if( ++Ctrl_Counter >= 5 )  
793     {  
794         Ctrl_Counter = 0;  
795         //200hz运行控制  
796         ctrl_main();  
797     }  
798 }
```

该中断的优先级是 INT\_PRIO\_7，位置传感器更新中断优先级不要高于此优先级（也就是只能 INT\_PRIO\_7，否则会导致线程问题）。

### 9.3 模式的执行

模式是在任务调度器主循环中执行的。

Modes.c 里将模式任务加入到任务列表：

```
62 Mode_Task_ID = STS_Add_Task( STS_Task_Trigger_Mode_RoughTime ,  
63 Modes[ current_mode ].mode_enter();
```

模式任务触发时，进入模式的主函数：

```
23 static void Modes_Server( unsigned int Task_ID )  
24 {  
25     Modes[ current_mode ].mode_main_func();  
26 }
```

## 10 代码接口手册

- 本代码接口中所有位置、速度、加速度数据单位均为 **cm** 厘米
- 本代码接口中所有角度、角速度、角加速度数据单位均为 **rad** 弧度
- 本代码接口中所有磁场数据单位均为 **Gauss** 高斯
- 本飞控模块化编程，不同模块通过接口进行访问操作，此文档必看！！
- 此文档包括：传感器接口、接收机接口

### 10.1 传感器接口

- 位于 Drivers 目录下
- 传感器接口分为：传感器读取接口 和 传感器注册及更新接口。
- 函数声明分别位于：Sensors.h 和 Sensors\_Backend.h 里
- 函数定义位于：Sensors.c 里

#### (1) IMU 传感器定义 (Sensors.h)

IMU 传感器包括加速度计、陀螺仪、磁力计，每种各支持 3 个。定义如下：

```
11  /*IMU传感器定义*/
12  typedef struct
13  {
14      bool present; //传感器是否存在
15      TIME last_update_time; //上次更新时间
16      float sample_time; //采样时间
17
18      float sensitivity; //灵敏度 (原始数据->实际单位 陀螺: rad/s 加速度: cm/s^2 磁场: gauss)
19
20      vector3_int data_raw; //原始数据
21      vector3_float data; //实际单位数据
22  }IMU_Sensor;
23  /*IMU传感器定义*/
..
```

#### (2) 位置传感器定义 (Sensors.h)

位置传感器包括气压、光流、超声波等，本飞控最多支持同时存在 8 个定位传感器。定义如下：

```
53  typedef struct
54  {
55      bool publishing; //是否正在更新
56
57      bool present; //传感器是否存在
58      bool available; //传感器是否可用
59      TIME last_update_time; //上次更新时间
60      TIME inavailable_start_time; //传感器不可用开始时间
61      float delay; //传感器延时
62      float sample_time; //采样时间
63
64      bool safe; //传感器是否安全 (数据缓慢变化不会发生跳变 !! 注意!! 如不确定不要设置为safe)
65      Position_Sensor_Type sensor_type; //传感器类型 (见枚举注释)
66      Position_Sensor_DataType sensor_DataType; //传感器数据类型 (见枚举注释)
67      Position_Sensor_frame velocity_data_frame; //速度数据坐标系 (见枚举注释)
68
69      vector3_double position_Global; //经纬度
70      vector3_float position; //位置 (cm)
71      vector3_float velocity; //速度 (cm/s)
72
73      //经纬度转平面坐标变量
74      Map_Projection mp;
75  }Position_Sensor;
```

其中：

- sensor\_type 定义了传感器是经纬度定位、相对定位，还是测距定位传感器。
- sensor\_DataType 定义了传感器的数据类型：例如 z 轴位置数据，xy 速度数据等。
- velocity\_data\_frame 针对速度传感器，定义了速度传感器所测速度所在的坐标系。

### (3) IMU 传感器读取接口 (Sensors.h)

IMU 传感器读取接口会返回 const 的 IMU 传感器结构体：

```

84
85  /*IMU传感器读取函数*/
86  const IMU_Sensor* GetAccelerometer( unsigned char index );
87  const IMU_Sensor* GetGyroscope( unsigned char index );
88  const IMU_Sensor* GetMagnetometer( unsigned char index );
89  /*IMU传感器注册函数*/

```

### (4) IMU 传感器注册、更新接口 (Sensors\_Backend.h)

在 IMU 传感器更新前，首先调用注册函数进行注册，设置传感器的灵敏度。

注册完成后把 IMU 传感器编号及原始数据送入 update 接口即可完成更新。

```

11  /*IMU*/
12
13  /*IMU传感器注册函数*/
14  bool IMUAccelerometerRegister( unsigned char index , float sensitivity );
15  bool IMUGyroscopeRegister( unsigned char index , float sensitivity );
16  bool IMUMagnetometerRegister( unsigned char index , float sensitivity );
17  /*IMU传感器注册函数*/
18
19  /*IMU传感器更新函数*/
20  bool IMUAccelerometerUpdate( unsigned char index , vector3_int data );
21  bool IMUGyroscopeUpdate( unsigned char index , vector3_int data );
22  bool IMUMagnetometerUpdate( unsigned char index , vector3_int data );
23  /*IMU传感器更新函数*/

```

### (5) 位置传感器读取接口 (Sensors.h)

位置传感器读取接口会返回 const 的位置传感器结构体：

```

95  /*位置传感器*/
96
97  /*位置传感器读取函数*/
98  const Position_Sensor* GetPositionSensor( unsigned char index );
99  /*位置传感器读取函数*/

```

### (6) 位置传感器注册、更新接口 (Sensors\_Backend.h)

在位置传感器更新前，首先调用注册函数进行注册，设置传感器的类型等参数：

```

29
30  /*位置传感器注册函数*/
31  //safe: 传感器是否安全（数据缓慢变化不会发生跳变！！注意！！如不确定不要设置为safe）
32  bool PositionSensorRegister(
33      unsigned char index ,\
34      Position_Sensor_Type sensor_type ,\
35      Position_Sensor_DataType sensor_data_type ,\
36      Position_Sensor_frame sensor_vel_frame ,\
37      float delay ,\
38      bool safe \
39  );
40  //注销传感器
41  bool PositionSensorUnRegister( unsigned char index );
42  /*位置传感器注册函数*/

```

如果位置传感器很久没有更新，MS\_Main 解算任务中会自动把此传感器取消注册。

注册完成后把位置传感器编号及与传感器 sensor\_data\_type 对应的数据送入 update 接口即可完成更新：

```

49
50  /*位置传感器更新函数*/
51  //delay参数小于0则不会改变delay
52  bool PositionSensorUpdatePositionGlobal( unsigned char index , vector3_double p
53  bool PositionSensorUpdatePosition( unsigned char index , vector3_float position
54  bool PositionSensorUpdatePositionGlobalVel( unsigned char index , vector3_double
55  bool PositionSensorUpdatePositionVel( unsigned char index , vector3_float posit
56  bool PositionSensorUpdateVel( unsigned char index , vector3_float vel , bool av
57  /*IMU传感器更新函数*/
58

```



## 10.2 接收机接口

- 位于 Drivers 目录下
- 接收机接口分为：接收机读取接口 和 接收机更新接口。
- 函数声明分别位于：Receiver.h 和 Receiver\_Backend.h 里
- 函数定义位于：Receiver.c 里

### (1) 接收机定义 (Receiver.h)

接收机包含 SBUS、PPM 等协议的接收机（至少具有 6 个通道），定义如下：

```
6 //接收机定义
7 typedef struct
8 {
9     bool present; //是否存在
10    bool connected; //是否已连接
11    bool available; //是否可用
12    TIME last_update_time; //上次更新时间
13    float update_time; //更新时间间隔
14
15    float raw_data[16]; //原始数据
16    float data[8]; //校准后的数据
17 }Receiver;
```

### (2) 接收机读取接口 (Receiver.h)

- get\_Receiver 会返回指定接收机的 const 结构体
- get\_current\_Receiver 会返回当前接收机的 const 结构体（自动选择序号最低的可用接收机，无可用接收机是返回随机接收机）
- get\_current\_Receiver\_Type 返回当前接收机的类型（SBUS 接收机、PPM 接收机等）

```
26 //获取指定的接收机
27 const Receiver* get_Receiver( RC_Type rc );
28 //获取当前使用的接收机
29 const Receiver* get_current_Receiver();
30 //获取当前使用的接收机
31 RC_Type get_current_Receiver_Type();
```

### (3) 接收机更新接口 (Receiver\_Backend.h)

把接收机类型、原始数据、是否已连接等信息发送给接口即可完成接收机数据更新。

```
8 //更新接收机数据
9 void Receiver_Update( RC_Type _rc , bool connected
```

## 11 代码二次开发教程

本飞控二次开发采用视频教程的方式，到售后群下载链接说明：

ACFLY售后群

群文件 > ACFLY 视频教程 (A9 & EDU)

文件



ACFLY 视频教程链接0916.pdf

## 12 版本更新日志

日期	新版本	旧版本	更新内容
20200423	V1.1	——	——
20200721	V1.2	V1.1	<ul style="list-style-type: none"><li>更新第 8 章，去除分电板连接方式。</li></ul>
20200803	V1.3	V1.2	<ul style="list-style-type: none"><li>调整 4.3、4.4 校准顺序</li></ul>
20200925	V1.4	V1.3	<ul style="list-style-type: none"><li>优化配图说明</li><li>修改部分细节表述</li></ul>
20210331	V2.0	V1.4	<ul style="list-style-type: none"><li>修改硬件接口定义与规格</li><li>新增屏幕显示说明</li><li>修改全文 LOGO 与排版</li></ul>

ACFLY 开源飞控提供技术支持

内容如有更新，恕不另行通知

您可以在 ACFLY 开源飞控售后群或百度云链接查询最新版本《ACFLY EDU 飞控用户手册》