竞赛无人机搭积木式编程（四）

---2023 年 TI 电赛 G 题空地协同智能消防系统（无人机部分）

无名小哥 2023 年 9 月 15 日

■ 赛题分析与解题思路综述

飞控用户在学习了 TI 电赛往届真题开源方案以及用户自定义航点自动飞行功能方案讲解后，能基于二次开发模式中已有的飞行控制 API 函数，即自动飞行支持函数和导航控制函数去实现特定赛题飞行动作、轨迹、航点等任务，上述内容是学生在备赛阶段学习无人机二次开发时的必备技能，对这部分不熟悉的可以回顾下前几讲的教程。

空地协同智能消防系统（G 题）

【本科组】

一、任务

设计一个由四旋翼无人机及消防车构成的空地协同智能消防系统。无人机上安装垂直向下的激光笔，用于指示巡逻航迹。巡防区域为 40dm×48dm。无人机巡逻时可覆盖地面 8dm 宽度区域。以缩短完成全覆盖巡逻时间为原则，无人机按照规划航线巡逻。发现火情后立即采取初步消防措施，并将火源地点位置信息发给消防车，使其前往熄灭火源。空地协同巡逻及消防工作完成时间越短越好。

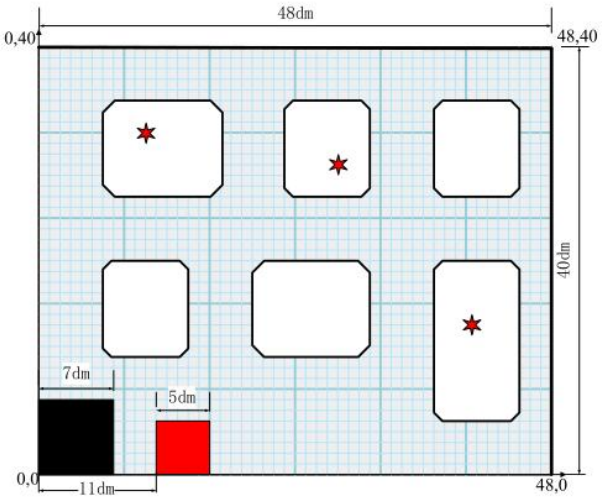


图 1 消防区域示意图

二、要求

1. 基本要求

（1）参赛队需自制模拟火源。模拟火源是用电池供电的红色光源，如 LED 等，用激光笔持续照射可控制开启或关闭：持续照射 2 秒左右开启，再持续照射 2 秒左右关闭。

（2）展示规划的巡逻航线图，在消防车上按键启动无人机垂直起飞后，无人机以 18dm 左右高度，在巡防区域按规划的航线完成全覆盖巡逻。

（3）无人机与消防车之间采用无线通信；巡逻期间无人机每秒向消防车发送 1 次位置坐标信息，消防车上显示器实时更新显示无人机位置坐标信息。

（4）巡逻中，消防车显示器显示巡逻航迹曲线，计算并显示累计巡逻航程。

（5）完成巡逻后，无人机返回，准确降落在起飞区域内。

2. 发挥部分

（1）手动操作激光笔点亮一个火源。在消防车上启动无人机巡逻。无人机按规划航线巡逻，发现火情后，前往接近火源（水平距离≤5dm）识别确认，再在无人机上用 LED 指示灯示警。

（2）无人机飞至火源地点上方，降低至 10dm 左右高度，悬停 3s 后抛洒灭火包，灭火包落在以火源点为中心、半径 3dm 圆形区域内；再将火源地点位置坐标发送给消防车，然后继续巡逻，完成后返航回到起飞点。

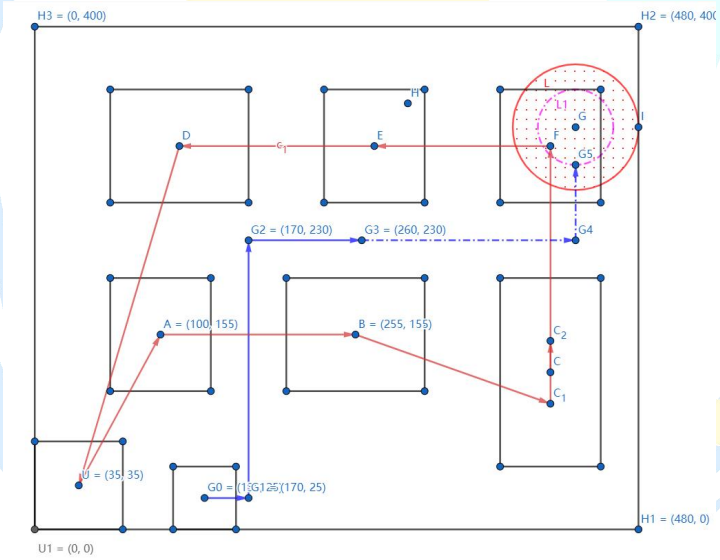
（3）消防车接收到火情信息，显示火源地点位置坐标后，从消防站出发前往火源地点，途中不得碾压街区及其边界线，在 5dm 距离内以激光笔光束照射模拟火源将其熄灭。

（4）熄灭模拟火源后消防车返回到出发区域内。发挥部分限时 360s 内完成。

（5）其他。

针对 2023 年 TI 电赛的无人机赛题“空地协同智能消防系统”这一具体任务，简单可以分为以下三个部分的设计：

- 无人机自主飞行任务设计
- 消防车自动运行与灭火任务设计
- 模拟火源的任务设计



- 无人机自主飞行任务设计概述

在无人机自主飞行任务设计中，关于实现基础部分飞行功能，用户可以无需编写自主飞行任务代码，直接利用程序中已有的用户自定义航点飞行功能，去通过按键或者地面站设置航点坐标即可完成基础部分飞行任务，有且仅需要做的工作是写 IO 控制去驱动激光笔点亮、串口通讯用数传去实时发送位置等信息给小车平台就可以，基础部分的无人机飞行任务设计只需要做很少的工作量就可以把分数拿满。

发挥部分无人机自主飞行任务设计分为航点遍历、底部视觉色块跟踪对准、驱动舵机实现投放动作，同样这三个元素在往届的赛题中均有涉及，躺赢者 PRO 飞控有与之配套的相关代码接口可以直接调用，如果用户在平常的训练中有跟着教程一路扎扎实实学习下来，自己编程去实现特定任务组合的任务代码应该也是驾轻就熟，对于熟练掌握的学生来讲就是用模块化的代码去搭积木的工作。

- 消防车自动运行与灭火任务设计概述

消防车自动运行与灭火任务的设计中，要求小车在行驶达到火源的过程中不得碾压街区边界线，这点需要小车事先规划好从出发点 G0 前往火源 G 处的中间路径点。小车平台上可搭载机载计算机通过激光雷达/深度相机实现全局定位，需要做的工作是实现航点遍历+视觉识别后激光笔对准，可加以云台辅助控制激光笔，实现在某一范围内激光笔动作，目的是实现快速的灭灯，具体云台单轴还是双轴需要根据实际做的模拟火源光敏电阻分布情况来。也可以不需要云台直接让视觉传感器和激光笔同轴安装，小车在视觉对准火源的过程中照射到光敏电阻就可以使得火源熄灭。

- 模拟火源任务的设计

自制模拟火源中要求用电池供电的红色光源去作为模拟火源，模拟火源的亮灭可以用激光笔持续照射去控制，持续照射 2 秒左右开启，再持续照射 2 秒左右关闭，同时对模拟火源的尺寸和形状有一定的要求。

（4）参赛队需制作 3 只带电源开关的模拟火源，根据评委指示放置在某些白色街区中。模拟火源可用电池供电的红色 LED 等，需带向上的喇叭形遮光罩，遮光罩角度约 60° 左右，见图 2，高度不超过 10cm。可用激光笔控制其开启或

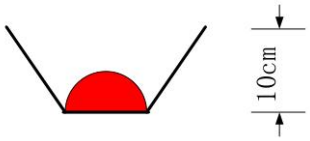


图 2 模拟火源遮光罩示意

关闭。竞赛结束一并封入作品箱。模拟火源发光部分直径不大于 2cm。

参赛学生可以通过自己熟练的任意一款单片机，简单设计基于灰度传感器、光敏传感器分压电路，加上 MOS 管驱动电路，编写 AD 电压采集、IO 驱动程序，去实现这一任务，这部分代码量非常小，只要是学过单片机、懂基础的电路硬件知识的同学都能实现。设计虽然简单但是也有一些设计细节要点需要注意：

- 红色光源的强度要足够，确保能满足无人机在不同环境光线下都能稳定识别。如果无人机检测火源不可靠，意味着发挥部分的无人机释放灭火包、消防车运行部分的得分会全部丢掉，即使你的无人机和小车运行的又快又稳。相当于仅可拿到基础部分的分数，从实际比赛来看，很有些队伍出现了现场火源检测不到，只拿到了基础分数。由于国测现场无人机有部分队伍现场出现炸机情况，国测现场基本只要你飞机能跑完基础部分就是国一，虽然在奖项上来看都是一等奖，但是为了不给自己留遗憾，完全可以通过合理设计光源信息避免。针对 23 年电赛无人机赛题中只需要检测单一火源信息的情况，可以采用红色光源+红外光源结合的设计，机器视觉模组加上红外截止滤光片能保证稳定检测识别。
- 遮光罩的材质与遮光罩的尺寸影响无人机和消防车对模拟火源的识别，为了增加红色 LED 灯作为光源时的发光面积和聚光效果，通常采用聚光环或者手工折叠锡箔纸去实现灯罩的设计，所以灯罩需要有一定高度并且不透光。同时发光灯珠是在灯罩底部，使得消防车上安装的摄像头从侧面上就难以看得到模拟光源灯珠发出来的光，故而消防车上的激光笔对模拟火源的照射也无法实现。实际比赛中看到有些学校采用的透光的灯罩或者将灯罩外侧单独涂红色便于小车去检测模拟火源。
- 模拟火源上光敏电阻容易对无人机上激光笔照射做出错误响应，题目中要求的是消防车照射火源时，火源才允许熄灭，但是实际赛题中在无人机识别到火源后，也需要用激光笔去照射模拟火源区并进行投放灭火包的这一动作。

为了防止无人机激光笔照射到光敏电阻时，模拟火源的提前熄灭，通常需要让灯罩截面足够大，俯视投影能覆盖底下整个电路板部分，防止无人机激光笔的直射造成的误触发。或者直接将光敏电阻朝下能有效避免无人机激光笔的影响，通过调整消防车激光笔的到同一安装高度去实现稳定照射并检测。

- 光敏电阻组成的分压电路稳态电压输出值受到环境光线影响需要现场标定，由于不同环境光线下光敏输出阻抗存在差异，因此程序里面检测阈值需要现场标定，实际可以采用按键或者电位器进行现场标定。当然如果在光敏电阻朝下并且底板为深色的情况下，可无需标定。
- 激光笔的光斑比较小，需要多个光敏电阻均匀分布才能确保激光笔照射到后实现灭火，由于没办法保证激光笔光斑能严格对准某一区域的光敏电阻，往往需要依靠灯罩、电路板的间接反射才能照射到光敏电阻，并且题目要求要持续不间断照射 2S 才行，实际小车很难做到这一点，故可以设计两组检测回路，比如单独用一路灰度传感器去作为 2S 持续照射的去点亮或者关闭，另外一路光敏回路作为消防车灭火时用，只要有激光笔照射到光敏上就可以触发光源熄灭条件。

综上所述，学生在实现 2023 年 TI 电赛空地协同智能消防系统中，针对无人机自主飞行任务设计，对于平时有积累并有掌握往届真题、案例 demo 的同学难度基本可以忽略，只需要把通过无线数传传输无人机状态给小车，小车发送 ADC 按键键值去无线操控无人机解锁、上锁、SDK、降落后，基本无人机部分的工作量就能完成。消防车如果搭载有机载计算机全局定位的情况下航点遍历、视觉对准、舵机云台控制等这些属于常规项，用通用的 SLAM 小车套件实现难度也不大，当然没有 SLAM 定位情况下，单靠 IMU+编码器组成的里程计系统，在选取摩擦力大的轮胎情况下也同样可以完成。最后单看模拟火源部分工作量也很少，除了上面说的一些细节要注意，对于普通电子信息相关专业学生能高效率完成。模拟火源熄灭的评分只占很少一部分，消防车能视觉对准火源照射，火源没有熄灭也不会影响学生去拿国奖。毕竟有些省份的无人机赛题只要发挥部分小车能动起来就能推优综测，综测过了，无人机国测基础部分能飞现场不炸机基本就是稳拿国一。

虽然赛题实现拆分来看每一项的工作量都不算大，单独某一项的难度可能还比不上大三大四学年的一个专业相关课程设计，但是这些都是建立在训练备赛阶段熟练掌握的已有资源的前提下。由于涉及的三个部分需要独立设计方案综合考量，并且彼此之间需要联合调试，整体上的工作量会很大，比赛阶段时间上会非常赶，特别是在自己实验室只有一组无人机队伍参赛的情况下，三个人能做好分工的情况下时间都显得捉襟见肘。

来年备战比赛的学生可以看看实际每个省无人机国赛出线或者省奖排名靠

前的队伍，能取得好成绩的很多都是该校有安排多组无人机参赛队伍，像西交、上大(SHU)、东大（SEU）、重大、北邮、桂电、山大（SDU）、哈工大、南工程、杭电等这样的无人机赛题方向传统强队，多年实力霸榜垄断该省份的无人机出线名额。同赛题多组参赛队伍的好处是彼此硬件方案接近，学校资源一致，虽然电赛要求每组队伍独立参赛，但是实际很难避免实验室为了确保出线名额，其它队伍参考种子队伍的方案去比赛。这种情况下就有更多的人手去做类似消防车、模拟火源基础服务的工作，比如一个实验室有三组队伍，这就相当 9 个人做了 3 个人的活，碰到今年每项都比较基础只是总工作量大的赛题，多组队伍参赛的学校想不出线都难。无人机赛题多组队伍参赛的结果好处就是能分担工作量，学生有更多的时间去做优化，做的好的情况下就是集体出线，完全不给同省其它高校无人机参赛队伍任何机会。

单看控制方向题目，由于无人机赛题相对小车或其它控制赛题经费开销巨大，一台搭载机载计算机支持激光雷达 SLAM/T265 定位的无人机外加上备用件耗材基本都要到 1w 往上走，外加当下经费吃紧、报销流程复杂，综合来看有实力组织多组参赛队伍去做无人机赛题的高校并不多。往往需要持续多年的人员经费投入才能取得好的成绩，当然也有一些首年使用 NC360 竞赛无人机开发平台就获得国奖、省一的同学，这部分相对比较少，每年都只有典型的几组，更多的是需要做好传承帮带，积累资源、持续投入迭代，这样自己的学校才会在电赛无人机赛题上有一席之地。

另外纵观这么多年的电赛无人机赛题得奖情况，还有一个现象就是有些省份出线和排名靠前的队伍并不一定是该省的头部 985、211 高校，有些省份双非、普通二本、民办独立高校的无人机获得国奖、省一的也很多，特别是在西北、西南和部分南方省份更为明显，经过以往调查来看这些学校往往有着较为充裕的经费支持，持续的供应能让这些学校的学生能用上更为可靠的硬件平台、定位方案、机载计算机平台，平时能做多套方案验证去高效率的学习，不会因为一时的炸机而畏手畏脚进展缓慢，在备战电赛无人机也更有底气，平常处理的问题多，比赛阶段团队协作运用综合能力，选择合适方法去高效率的解决实际问题就会更得心应手。

学生的能力固然重要，但是没有学校、实验室、指导老师的持续的支持和经费投入，单凭学生的一腔热血很难持久。没有学校支持顶多就是某届某个学生能力出众凭个人能力拿个一等奖，没有持续的经费支持，后面没人接手，青黄不接后就很难再拿好名次了，特别是在前几年口罩期间这种传承断代、设备长期吃灰的情况就更为明显。

- 无人机自动飞行任务的软件设计
- 第一阶段——自动起飞到航巡高度

uint8_t Auto_Takeoff(float target)//自动起飞到某一高度

```
1643  uint8_t Auto_Takeoff(float target)
1644  {
1645    static uint8_t n=11;
1646    Vector3f target_position;
1647    basic_auto_flight_support();//基本飞行支持软件
1648    if(flight_subtask_cnt[n]==0)
1649    {
1650      //不加此行代码，当后续全程无油门上下动作后，飞机最后自动降落到地面不会自动上锁
1651      Unwanted_Lock_Flag=0;//允许飞机自动上锁，原理和手动推油起飞类似
1652
1653      //记录下初始起点位置，实际项目中可设置为某一基准原点
1654      base_position.x=VIO_SINS.Position[_EAST];
1655      base_position.y=VIO_SINS.Position[_NORTH];
1656      base_position.z=NamelessQuad.Position[_UP];
1657
1658      //execute_time_ms[n]=10000/flight_subtask_delta;//子任务执行时间
1659      target_position.x=base_position.x;
1660      target_position.y=base_position.y;
1661      target_position.z=base_position.z+target;
1662      Horizontal_Navigation(target_position.x,
1663                            target_position.y,
1664                            target_position.z,
1665                            GLOBAL_MODE,
1666                            MAP_FRAME);
1667      flight_subtask_cnt[n]=1;
1668      return  0;
1669    }
1670    else if(flight_subtask_cnt[n]==1)
1671    {
1672      //判断是否起飞到目标高度
1673      if(flight_global_cnt[n]<400)//持续400*5ms满足
1674      {
1675        if(ABS(Total_Controller.High_Position_Control.Err)<=10.0f)  flight_global_cnt[n]++;
1676        else flight_global_cnt[n]/=2;
1677        return  0;
1678      }
1679      else//持续200*5ms满足，表示到达目标高度
1680      {
1681        return  1;
1682      }
1683    }
1684    return  0;
1685  }
```

不要遗漏了基本自动飞行支持函数

1

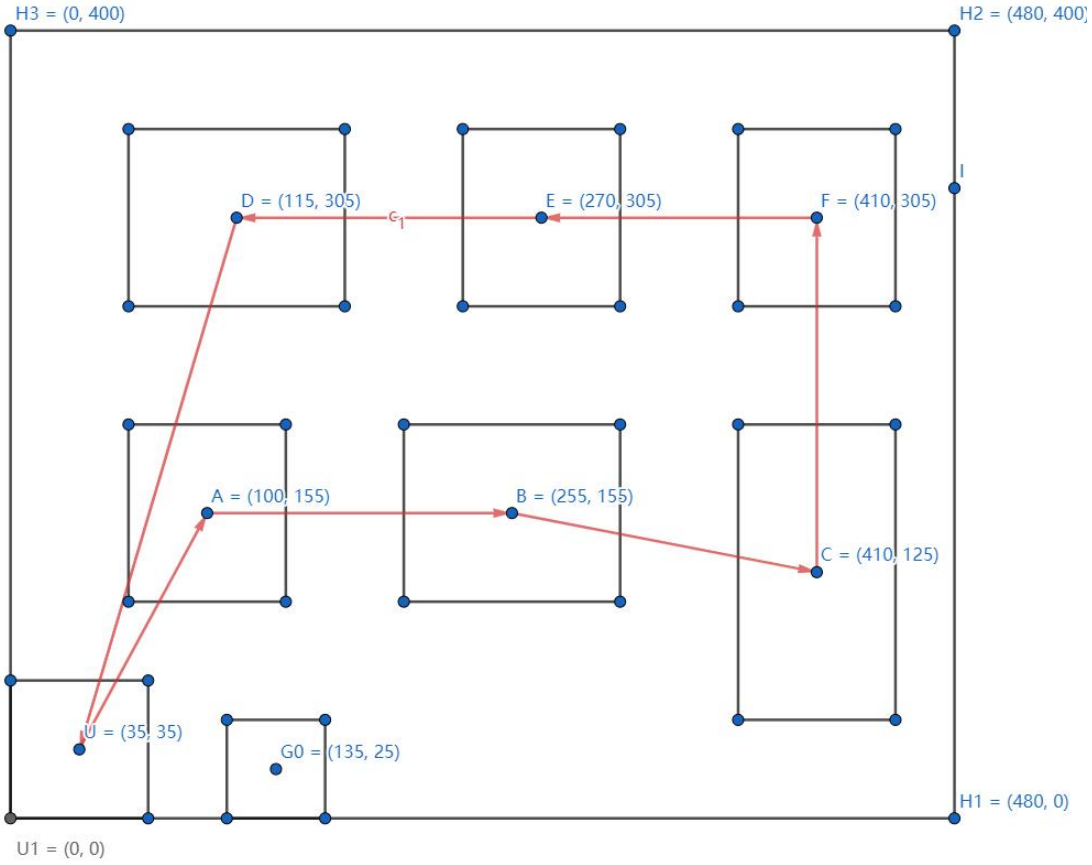
2

函数输入参数 target 为目标高度，自动起飞任务分为两个线程，第一步为记录当前 3 维位置信息，作为导航初始原点位置。并且通过导航控制函数设置期望目标高度位置。第二步为实时检测高度偏差值，连续 1S 满足位置偏差在 10cm 以内后，函数返回值置 1 后，自动起飞到目标高度任务完成，当前的 mode 会加上 task_select_cnt 进而实现自主起飞任务完成后，会进入 case 21/22 执行航点遍历作业任务，其中 task_select_cnt 可以通过遥控器调节，方便现场执行不同的 sdk 任务。

```
203     case 20://自动起飞到180cm高度，完成后根据task_select_cnt值来决策执行基础任务还是发挥任务
204     {
205       if(Auto_Takeoff(180)==1)//期望高度180cm
206       {
207         *mode+=task_select_cnt;//task_select_cnt设置为1:到达目标高度后，切换到基础部分任务
208                                //task_select_cnt设置为2:到达目标高度后，切换到发挥部分任务
209       }
210     }
211     break;
212     case 21://2023年TI国赛G题—基础部分
213     {
214       Air_Ground_Extinguish_Fire_System_Basic();//无人机航点遍历后返航降落
215     }
216     break;
217     case 22://2023年TI国赛G题—发挥部分
218     {
219       Air_Ground_Extinguish_Fire_System_Innovation();//无人机航点遍历后返航降落+灭火动作+检测火源信息
220     }
221     break;
```

- 基础部分——航点遍历作业任务

  void Air_Ground_Extinguish_Fire_System_Basic(void)



第一步将高度期望设置成第巡逻高度 180cm，水平位置期望为初始起飞时候的水平位置，并设置激光笔为持续闪烁，便于裁判判断飞机机身中心在地面上的投影位置，起飞点上方悬停时间设置为 1S。

```
2594  const int16_t uav_home_center_coordinate[2]={35,35};//U
2595  #define Patrol_Height          180//巡逻高度180cm
2596  #define patrol_fixed_3d_30cm   30.0f//30cm
2597  #define patrol_fixed_3d_20cm   20.0f//20cm
2598  #define patrol_fixed_3d_10cm   10.0f//10cm
2599  #define patrol_fixed_3d_5cm    5.0f//5cm
2600  #define patrol_fixed_times     3//满足次数
2601  #define patrol_fixed_2d_5cm    5.0f//5cm
2602  #define patrol_fixed_2d_times 5
2603  void Air_Ground_Extinguish_Fire_System_Basic(void)
2604  {
2605    static uint8_t n=14;
2606    Vector3f target_position;
2607    float x=0,y=0,z=0;
2608    if(flight_subtask_cnt[n]==0)//起飞点作为第一个悬停点
2609    {
2610      basic_auto_flight_support();//基本飞行支持软件
2611      //激光笔点亮
2612      laser_light_1.period=100;//200*5ms
2613      laser_light_1.light_on_percent=1.0f;
2614      laser_light_1.reset=1;
2615      laser_light_1.times=10000;
2616
2617      //记录下初始起点位置，实际项目中可设置为某一基准原点
2618      //base_position.x=VIO_SINS.Position[_EAST];
2619      //base_position.y=VIO_SINS.Position[_NORTH];
2620      base_position.z=Patrol_Height;//巡逻高度
2621
2622      x=base_position.x;
2623      y=base_position.y;
2624      z=Patrol_Height;
2625      target_position.x=x;
2626      target_position.y=y;
2627      target_position.z=z;
2628      Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
2629
2630      flight_subtask_cnt[n]=1;
2631      flight_global_cnt[n]=0;
2632      flight_global_cnt2[n]=0;
2633      execute_time_ms[n]=1000/flight_subtask_delta;//子任务执行时间
2634    }
```

起飞点上方悬停 1S 后，会将键盘输入的第一个航点 A 点的水平坐标填入到期望的目标位置，随后飞机会执行从起飞点正上方飞向第一个航点 A 点的动作，在执行本任务中对水平位置误差进行实时检测，连续 N 次水平位置误差小于某一阈值，即可以认为到达 A 点正上方附近，满足达到 A 点条件后会将航点计数器 flight_global_cnt2[n]自加，flight_subtask_cnt[n]置 1，飞机会继续执行 flight_subtask_cnt[n]==1 中的任务，即刷新下一航点位置后再进入 flight_subtask_cnt[n]==2 中实时检测水平位置偏差，依次判断是否到达 B、C、F、E、D 航点。

```
else if(flight_subtask_cnt[n]==1)//起飞之后原定悬停1s后再执行航点任务
{
  basic_auto_flight_support();//基本飞行支持软件

  //判断所有航点均执行完毕，执行下一任务
  if(flight_global_cnt2[n]>=block_navpoint_num_basic)
  {
    flight_subtask_cnt[n]=3;//结束航点遍历
    flight_global_cnt[n]=0;
    execute_time_ms[n]=0;
    //航点计数器清0
    flight_global_cnt2[n]=0;

          //继续执行返航任务航点任务
    target_position.x=base_position.x;
    target_position.y=base_position.y;
    target_position.z=Patrol_Height;
    Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);

    return ;
  }

  uint16_t current_num=constrain_int32(flight_global_cnt2[n],0,block_navpoint_num_basic-1);//限幅防溢出
  if(execute_time_ms[n]>0) execute_time_ms[n]--;
  if(execute_time_ms[n]==0)//悬停时间计数器归零，悬停任务执行完毕
  {
    //设置A航点任务
    x=block_center_coordinate[current_num][0]-uav_home_center_coordinate[0];
    y=block_center_coordinate[current_num][1]-uav_home_center_coordinate[1];
    z=Patrol_Height;
    target_position.x=base_position.x+x;//水平位置期望为起飞后基准位置+位置x偏移
    target_position.y=base_position.y+y;//水平位置期望为起飞后基准位置+位置y偏移
    target_position.z=z;
    Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);

    flight_subtask_cnt[n]=2;
    flight_global_cnt[n]=0;
    execute_time_ms[n]=0;
  }
}
```

```
else if(flight_subtask_cnt[n]==2)//检测起飞点悬停完毕后，飞向下一个目标点
{
  basic_auto_flight_support();//基本飞行支持软件
  //判断是否到达目标航点位置
  if(flight_global_cnt[n]<patrol_fixed_times)//持续10*5ms=0.05s满足
  {
    float dis_cm=pythagorous3(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y,Total_Controller.Height_Position_Control.Err);
    //距离误差约束20cm，针对基础赛题可以加大此处阈值，从而实现更快速遍历
    if(dis_cm<=patrol_fixed_3d_20cm)  flight_global_cnt[n]++;
     else flight_global_cnt[n]/=2;
  }
  else//持续10*5ms满足，表示到达目标航点位置
  {
    flight_subtask_cnt[n]=1;
    flight_global_cnt[n]=0;
    execute_time_ms[n]=500/flight_subtask_delta;;//更改此值可以对每个航点的悬停时间进行设置
    //航点计数器自加
    flight_global_cnt2[n]++;
  }

  //判断所有航点均执行完毕，执行下一任务
  if(flight_global_cnt2[n]>=block_navpoint_num_basic)
  {
    flight_subtask_cnt[n]=3;//结束航点遍历
    flight_global_cnt[n]=0;
    execute_time_ms[n]=0;
    //航点计数器清0
    flight_global_cnt2[n]=0;

    //继续执行返航任务航点任务
    target_position.x=base_position.x;
    target_position.y=base_position.y;
    target_position.z=Patrol_Height;
    Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
    return ;
  }
}
```

当满足航点计数器 flight_global_cnt2[n]>=block_navpoint_num_basic 时表示所有区块都已经巡逻完毕，结束航点遍历后无人机继续执行返航降落。

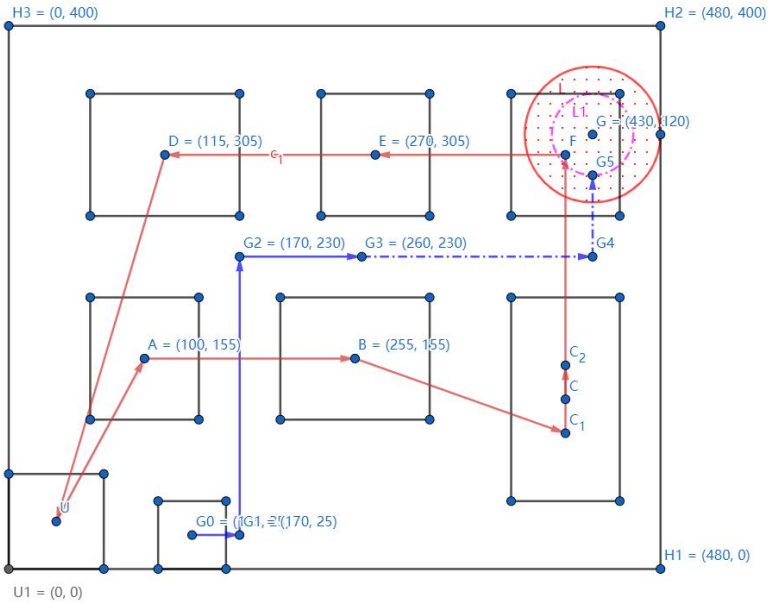
```
else if(flight_subtask_cnt[n]==1)//起飞之后原定悬停1s后再执行航点任务
{
  basic_auto_flight_support();//基本飞行支持软件

  //判断所有航点均执行完毕，执行下一任务
  if(flight_global_cnt2[n]>=block_navpoint_num_basic)
  {
    flight_subtask_cnt[n]=3;//结束航点遍历
    flight_global_cnt[n]=0;
    execute_time_ms[n]=0;
    //航点计数器清0
    flight_global_cnt2[n]=0;

         //继续执行返航任务航点任务
    target_position.x=base_position.x;
    target_position.y=base_position.y;
    target_position.z=Patrol_Height;
    Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);

    return ;
  }
```

无人机会执行返航动作，首先无人机飞到起飞点正上方，连续 N 次水平位置误差小于某一阈值即可认为达到起飞点正上方附近，满足水平抵达之后会执行原地降落至地面的任务，到达地面后无人机会满足地面检测条件自动上锁。

```
2712    else if(flight_subtask_cnt[n]==1)//飞向起飞点正上方
2713    {
2714      basic_auto_flight_support();//基本飞行支持软件
2715      //判断是否到达目标航点位置
2716      if(flight_global_cnt[n]<patrol_fixed_2d_times)//持续10*5ms=0.05s满足
2717      {
2718        float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
2719        if(dis_cm<=patrol_fixed_2d_5cm) flight_global_cnt[n]++;
2720        else flight_global_cnt[n]/=2;
2721      }
2722      else//持续10*5ms满足，表示到达目标航点位置
2723      {
2724        flight_subtask_cnt[n]++;
2725        flight_global_cnt[n]=0;
2726        execute_time_ms[n]=0;
2727
2728        //以下复位操作的作用：空中到地面不同高度，环境分布陈设变化造成的误差
2729        //不同高度梯度上陈设变化不大时，以下特殊处理部分可以不去掉
2730        //特殊处理开始
2731        /*
2732        send_check_back=4;//重置slam
2733        VIO_SINS.Position[_EAST] = 0;
2734        VIO_SINS.Position[_NORTH]= 0;
2735        OpticalFlow_Pos_Ctrl_Expect.x=0;
2736        OpticalFlow_Pos_Ctrl_Expect.y=0;
2737        */
2738        //特殊处理结束
2739      }
2740    }
2741    else if(flight_subtask_cnt[n]==4)//原地下降
2742    {
2743      Flight.yaw_ctrl_mode=ROTATE;
2744      Flight.yaw_outer_control_output=RC_Data.rc_rpyt[RC_YAW];
2745      OpticalFlow_Control_Pure(0);
2746      Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-30);//高度控制
2747    }
2748    else
```

- 发挥部分——航点遍历+色块对准+投放动作任务

  void Air_Ground_Extinguish_Fire_System_Innovation(void)



第一步将高度期望设置成第巡逻高度 180cm，水平位置期望为初始起飞时候的水平位置，并设置激光笔为持续闪烁，便于裁判判断飞机机身中心在地面上的投影位置，起飞点上方悬停时间设置为 1S，同时驱动排针 PWM 的预留 PWM2 对舵机进行控制实现夹住灭火包。在起飞点上方悬停完毕后会设置 A 点为目标航点，无人机会飞向 A 点。

```
2792  void Air_Ground_Extinguish_Fire_System_Innovation(void)
2793  {
2794      static uint8_t n=15;
2795      static float fx=0,fy=0;
2796      static uint16_t current_nav_cnt=0;
2797      Vector3f target_position;
2798      float x=0,y=0,z=0;
2799      if(flight_subtask_cnt[n]==0)//起飞点作为第一个悬停点
2800      {
2801          basic_auto_flight_support();//基本飞行支持软件
2802          //激光笔点亮
2803          laser_light_1.period=100;//200*5ms
2804          laser_light_1.light_on_percent=1.0f;
2805          laser_light_1.reset=1;
2806          laser_light_1.times=10000;
2807
2808          //记录下初始起点位置，实际项目中可设置为某一基准原点
2809          //base_position.x=VIO_SINS.Position[_EAST];
2810          //base_position.y=VIO_SINS.Position[_NORTH];
2811          base_position.z=Patrol_Height;//巡逻高度
2812
2813          x=base_position.x;
2814          y=base_position.y;
2815          z=Patrol_Height;
2816          target_position.x=x;
2817          target_position.y=y;
2818          target_position.z=z;
2819          Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
2820
2821          flight_subtask_cnt[n]=1;
2822          flight_global_cnt[n]=0;
2823          flight_global_cnt2[n]=0;
2824          execute_time_ms[n]=1000/flight_subtask_delta;//子任务执行时间
2825
2826          //利用预留PWM通道2对舵机控制
2827          Reserved_PWM2_Output(pinch_pwm_us);//夹住动作,具体值需要根据实际机械手自己调整
2828      }
```

```
2829    else if(flight_subtask_cnt[n]==1)//起飞之后原定悬停1S后再执行航点任务
2830    {
2831      basic_auto_flight_support();//基本飞行支持软件
2832      //判断所有航点均执行完毕，执行下一任务
2833      if(flight_global_cnt2[n]>=block_navpoint_num_innovation)
2834      {
2835        flight_subtask_cnt[n]=3;//结束航点遍历,准备返航降落
2836        flight_global_cnt[n]=0;
2837        execute_time_ms[n]=0;
2838        //航点计数器清0
2839        flight_global_cnt2[n]=0;
2840
2841        //将目标航点设置为home点,继续执行返航任务航点任务
2842        target_position.x=base_position.x;
2843        target_position.y=base_position.y;
2844        target_position.z=Patrol_Height;
2845        Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
2846        return ;
2847      }
2848
2849      uint16_t current_num=constrain_int32(flight_global_cnt2[n],0,block_navpoint_num_innovation-1);//限幅防溢出
2850      if(execute_time_ms[n]>0) execute_time_ms[n]--;
2851      if(execute_time_ms[n]==0)//悬停时间计数器归零，悬停任务执行完毕
2852      {
2853        //设置A航点任务
2854        x=block_innovation_coordinate[current_num][0]-uav_home_center_coordinate[0];
2855        y=block_innovation_coordinate[current_num][1]-uav_home_center_coordinate[1];
2856        z=Patrol_Height;
2857        target_position.x=base_position.x+x;//水平位置期望为起飞后基准位置+位置x偏移
2858        target_position.y=base_position.y+y;//水平位置期望为起飞后基准位置+位置y偏移
2859        target_position.z=z;
2860        Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
2861        flight_subtask_cnt[n]=2;
2862        flight_global_cnt[n]=0;
2863        execute_time_ms[n]=0;
2864      }
2865    }
```

达到 A 点之后的航点遍历过程，无人机会实时判断底部 OPENMV 视觉模拟火源情况，如果识别到了模拟火源，无人机会中止航点遍历的过程，进入色块对准模式。

```
2879        flight_global_cnt[n]=0;
2880        execute_time_ms[n]=2000/flight_subtask_delta;//更改此值可以对每个航点的悬停时间进行设置
2881        //航点计数器自加
2882        flight_global_cnt2[n]++;
2883      }
2884
2885      //判断所有航点均执行完毕，执行下一任务
2886      if(flight_global_cnt2[n]>=block_navpoint_num_innovation)
2887      {
2888        flight_subtask_cnt[n]=3;//结束航点遍历
2889        flight_global_cnt[n]=0;
2890        execute_time_ms[n]=0;
2891        //航点计数器清0
2892        flight_global_cnt2[n]=0;
2893        //继续执行返航任务航点任务
2894        target_position.x=base_position.x;
2895        target_position.y=base_position.y;
2896        target_position.z=Patrol_Height;
2897        Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
2898        return ;
2899      }
2900
2901
2902      if(fire_flag==1) return;//如果火源信息已经产生，后续巡逻不再对火情进行判断，仅仅是航点遍历飞行
2903      //判断底部火源情况
2904      if(flight_global_cnt2[n]>0)//飞到第一个街区A之后,才允许判断火情
2905      {
2906        if(Opv_Top_View_Target.trust_flag==1)//如果视觉检测到了火情
2907        {
2908          flight_subtask_cnt[n]=5;//允许提前结束航点遍历，执行色块对准
2909          flight_global_cnt[n]=0;
2910          execute_time_ms[n]=10000/flight_subtask_delta;//允许色块对准动作执行时间为10000ms
2911          //记录当前航点计数器,
2912          current_nav_cnt=flight_global_cnt2[n];//记录执行返航任务航点任务,便于抵近火源后重新恢复巡逻
2913          //发现火情后LED指示灯示警
2914          laser_light_2.period=200;
2915          laser_light_2.light_on_percent=0.5f;
2916          laser_light_2.reset=1;
2917          laser_light_2.times=10;
2918        }
2919      }
2920      /*****************************************************************************/
2921    }
```

线程计数器 flight_subtask_cnt[n]赋值为 5 进入色块对准模式，为了避免初始识别到火源时位置偏差太大，调整过激，这里有对未知色块位置-速度控制器输出进行了限幅度。

```
2958    else if(flight_subtask_cnt[n]==5)//执行色块对准，实现无人机飞至火源上方的动作
2959    {
2960      Color_Block_Control_Pilot();//俯视OPENMV视觉水平追踪
2961      //对色块对准控制的输出结果进行限幅，使姿态调节过程尽可能平滑
2962      Flight.roll_outer_control_output =constrain_float(Flight.roll_outer_control_output ,-10.0f,10.0f);
2963      Flight.pitch_outer_control_output=constrain_float(Flight.pitch_outer_control_output,-10.0f,10.0f);
2964      //偏航与高度控制
2965      Flight.yaw_ctrl_mode=ROTATE;
2966      Flight.yaw_outer_control_output =RC_Data.rc_rpyt[RC_YAW];
2967      Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制
2968
2969      //任务结束约束条件1
2970      //色块对准执行100000ms后，默认已完成对准
2971      if(execute_time_ms[n]>0) execute_time_ms[n]--;
2972      if(execute_time_ms[n]==0)//悬停时间计数器归零，悬停任务执行完毕
2973      {
2974        OpticalFlow_Control_Pure(1);//强制刷新悬停位置
2975        //
2976        x=VIO_SINS.Position[_EAST];
2977        y=VIO_SINS.Position[_NORTH];
2978        z=Patrol_Work_Height;
2979        target_position.x=x;
2980        target_position.y=y;
2981        target_position.z=z;
2982        Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
2983
2984        flight_subtask_cnt[n]++;
2985        flight_global_cnt[n]=0;
2986        execute_time_ms[n]=0;
2987
2988        //色块对准执行完毕后，将当前位置记录下来
2989        fx=VIO_SINS.Position[_EAST]; //火源位置cm
2990        fy=VIO_SINS.Position[_NORTH];//火源位置cm
2991      }
2992
2993      //任务结束约束条件2
2994      //判断是否到达目标正上方
2995      if(flight_global_cnt[n]<patrol_fixed_times)//持续10*5ms=0.05s满足
2996      {
2997        float dis_cm=pythagorous2(Opv_Top_View_Target.sdk_target.x,Opv_Top_View_Target.sdk_target.y);
2998        if(dis_cm<=patrol_fixed_3d_5cm) flight_global_cnt[n]++;
2999        else flight_global_cnt[n]/=2;
3000      }
```

输出限幅

时间约束

位置约束

色块对准的结束条件有两个，分别为时间约束和色块位置约束，无人机值执行色块对准的过程中，色块位置误差会逐渐收敛到 0，如果连续 N 次满足色块位置误差比较小会提前结束色块对准过程。

结束色块对准后，无人机会记录当前的水平位置供后续消防车使用，之后无人机会调节高度到 100cm。下降完毕后会原地悬停 3S 后驱动舵机投放灭火包。

```
3001      else//持续10*5ms满足，表示到达目标航点位置
3002      {
3003        OpticalFlow_Control_Pure(1);//强制刷新悬停位置
3004        //
3005        x=VIO_SINS.Position[_EAST];
3006        y=VIO_SINS.Position[_NORTH];
3007        z=Patrol_Work_Height;
3008        target_position.x=x;
3009        target_position.y=y;
3010        target_position.z=z;
3011        Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
3012
3013        flight_subtask_cnt[n]++;
3014        flight_global_cnt[n]=0;
3015        execute_time_ms[n]=0;
3016
3017        //色块对准执行完毕后，将当前位置记录下来
3018        fx=VIO_SINS.Position[_EAST]; //火源位置cm
3019        fy=VIO_SINS.Position[_NORTH];//火源位置cm
3020      }
3021    }
3022    else if(flight_subtask_cnt[n]==6)//下降高度至100cm
3023    {
3024      basic_auto_flight_support();//基本飞行支持软件
3025      //判断是否到达作业高度
3026      if(flight_global_cnt[n]<200)//持续200*5ms满足
3027      {
3028        if(ABS(Total_Controller.Height_Position_Control.Err)<=10.0f)  flight_global_cnt[n]++;
3029        else flight_global_cnt[n]/=2;
3030      }
3031      else//放出吊仓
3032      {
3033        flight_subtask_cnt[n]++;
3034        flight_global_cnt[n]=0;
3035        execute_time_ms[n]=3000/flight_subtask_delta;
3036      }
3037    }
```

投放完毕后更新火源信息，相关的坐标信息会通过串口数传发送给消防车。

Developer_Mode.c | Key.c | OLED.c | oled.h | Subtask_Demo.c

```
3033        flight_subtask_cnt[n]++;
3034        flight_global_cnt[n]=0;
3035        execute_time_ms[n]=3000/flight_subtask_delta;
3036      }
3037    }
3038    else if(flight_subtask_cnt[n]==7)//悬停3S后,抛洒灭火包
3039    {
3040      basic_auto_flight_support();//基本飞行支持软件
3041
3042      if(execute_time_ms[n]>0) execute_time_ms[n]--;
3043      if(execute_time_ms[n]==0)//悬停时间计数器归零，悬停任务执行完毕
3044      {
3045        //利用预留PWM通道2对舵机控制
3046        Reserved_PWM2_Output(release_pwm_us);//释放动作,具体值需要根据实际机械手自己调整
3047
3048        //舵机动作需要一定时间，这里给过渡时间2000ms
3049        flight_subtask_cnt[n]++;
3050        flight_global_cnt[n]=0;
3051        execute_time_ms[n]=2000/flight_subtask_delta;
3052      }
3053    }
3054    else if(flight_subtask_cnt[n]==8)//释放灭火包动作后2000ms
3055    {
3056      basic_auto_flight_support();//基本飞行支持软件
3057      if(execute_time_ms[n]>0) execute_time_ms[n]--;
3058      if(execute_time_ms[n]==0)//悬停时间计数器归零，悬停任务执行完毕
3059      {
3060        //更新火源信息,坐标数据通过串口发送到小车
3061        fire_flag=1;
3062        fire_x=(fx-base_position.x)+uav_home_center_coordinate[0];//去掉初始偏置后绝对坐标
3063        fire_y=(fy-base_position.y)+uav_home_center_coordinate[1];//去掉初始偏置后绝对坐标
3064
3065        target_position.x=VIO_SINS.Position[_EAST];
3066        target_position.y=VIO_SINS.Position[_NORTH];
3067        target_position.z=Patrol_Height;
3068        Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
3069
3070        flight_subtask_cnt[n]++;
3071        flight_global_cnt[n]=0;
3072        execute_time_ms[n]=0;
3073      }
3074    }
3075    else if(flight_subtask_cnt[n]==9)//恢复到180cm巡逻高度
3076    {
```

最后会恢复到 180cm 巡逻高度，抵达后会继续识别到火源信息前的航点遍过程。

```
3075    else if(flight_subtask_cnt[n]==9)//恢复到180cm巡逻高度
3076    {
3077      basic_auto_flight_support();//基本飞行支持软件
3078      //判断是否到达作业高度
3079      if(flight_global_cnt[n]<200)//持续200*5ms满足
3080      {
3081        if(ABS(Total_Controller.Height_Position_Control.Err)<=10.0f)  flight_global_cnt[n]++;
3082        else flight_global_cnt[n]/=2;
3083      }
3084      else//准备返航降落
3085      {
3086        flight_subtask_cnt[n]=1;//1:抵近查看火情后,继续巡逻剩下的区块
3087                                //3:侦察到火情后,提前结束航点遍历,返回执行线程3即返航+原地降落
3088        flight_global_cnt[n]=0;
3089        execute_time_ms[n]=0;
3090        //航点计数器加1,继续执行剩余的航点
3091        flight_global_cnt2[n]=current_nav_cnt+1;
3092
3093        //继续执行返航任务航点任务
3094        target_position.x=base_position.x;
3095        target_position.y=base_position.y;
3096        target_position.z=Patrol_Height;
3097        Horizontal_Navigation(target_position.x,target_position.y,target_position.z,GLOBAL_MODE,MAP_FRAME);
3098      }
3099    }
3100    else
```

在巡逻完毕 A、B、C、F、E、D 后，会继续执行返航降落，这部分和基础部分代码一样，即 flight_subtask_cnt 为 1~3 的内容，故不再赘述，可参考基础部分解释。

最后给出小车无线发送 ADC 按键键盘 AD 值，消防车控制无人机解锁、上锁、SDK、自动降落的代码以及无人机发送三维位置、航程、模拟火源信

息的代码段。

```
void NCLink_Send_To_Firetruck(float x,float y,float z,float dis,int16_t xf,int16_t yf,uint8_t update)
{
  uint8_t sum=0,_cnt=0,i=0;
  int16_t _temp;

  nclink_databuf[_cnt++]=NCLink_Head[1];
  nclink_databuf[_cnt++]=NCLink_Head[0];
  nclink_databuf[_cnt++]=0xE1;
  nclink_databuf[_cnt++]=0;

  Float2Byte(&x,nclink_databuf,_cnt);//4
  _cnt+=4;
  Float2Byte(&y,nclink_databuf,_cnt);//8
  _cnt+=4;
  Float2Byte(&z,nclink_databuf,_cnt);//12
  _cnt+=4;
  Float2Byte(&dis,nclink_databuf,_cnt);//16
  _cnt+=4;

  _temp = (int)(xf);
  nclink_databuf[_cnt++]=BYTE1(_temp);
  nclink_databuf[_cnt++]=BYTE0(_temp);
  _temp = (int)(yf);
  nclink_databuf[_cnt++]=BYTE1(_temp);
  nclink_databuf[_cnt++]=BYTE0(_temp);

  nclink_databuf[_cnt++]=update;

  nclink_databuf[3] = _cnt-4;

  for(i=0;i<_cnt;i++) sum ^= nclink_databuf[i];
  nclink_databuf[_cnt++]=sum;

  nclink_databuf[_cnt++]=NCLink_End[0];
  nclink_databuf[_cnt++]=NCLink_End[1];
  USART4_Send(nclink_databuf,_cnt);
}
```

```
  {
    data_len--;
    buf[4+data_cnt++]=data;
    if(data_len==0)  state = 5;
  }
  else if(state==5)//异或校验
  {
    state = 6;
    buf[4+data_cnt++]=data;
  }
  else if(state==6&&data==NCLink_End[0])//帧尾0
  {
    state = 7;
    buf[4+data_cnt++]=data;
  }
  else if(state==7&&data==NCLink_End[1])//帧尾1
  {
    state = 0;
    buf[4+data_cnt]=data;

    //小车发送过来的数据解析
    uint16_t _cnt=data_cnt+5;
    uint8_t sum = 0;
    for(uint8_t i=0;i<(_cnt-3);i++)  sum ^= *(buf+i);
    if(!(sum==*(buf+_cnt-3)))                                          return;//判断sum
    if(!(*(buf)==NCLink_Head[1]&&*(buf+1)==NCLink_Head[0]))            return;//判断帧头
    if(!(*(buf+_cnt-2)==NCLink_End[0]&&*(buf+_cnt-1)==NCLink_End[1])) return;//帧尾校验
    float ad1,ad2;
    Byte2Float(buf, 4,&ad1);
    Byte2Float(buf, 8,&ad2);

    wireless_adc_value[0]=ad1;
    wireless_adc_value[1]=ad2;
  }
  else state = 0;
}
```

```
void ADC_Button_Read(void)
{
  //将原始的数字12位ADC数字亮转化成电压,单位V
  if(wireless_adc_key_ctrl_enable==1)//adc电压来源于无线数传传输，如小车
  {
    Button_ADCResult[0]=wireless_adc_value[0]*3.3f/4095.0f;//PD3
    Button_ADCResult[1]=wireless_adc_value[1]*3.3f/4095.0f;//PD2
  }
  else//adc电压来源于飞控自身端口采集
  {
    Button_ADCResult[0]=adc_value[1]*3.3f/4095.f;//PD3
    Button_ADCResult[1]=adc_value[2]*3.3f/4095.f;//PD2
  }
```

无线 ADC 按键使用相关操作教程方法见下方链接

https://www.bilibili.com/video/BV1vh4y1w7DP/

由于本节内容过多，基于盘古 TI MCU 的多功能控制器实现的消防车的教程和制作模拟火源的教程单独作为一节给出。