

竞赛无人机搭积木式编程

——以 2021 年电赛国奖标准完整复现为例学习

首先我们需要了解下自动飞行任务执行过程几组关键变量的用法与实际作用效果：

```
5 uint16_t flight_subtask_cnt[FLIGHT_SUBTASK_NUM]={0}; //飞行任务子线程计数器，可以用于控制每个航点子线程的执行
6 uint32_t flight_global_cnt[FLIGHT_SUBTASK_NUM]={0}; //飞行任务子线程全局计数器，可以结合位置偏差用于判断判断航点是否到达
7 uint32_t execute_time_ms[FLIGHT_SUBTASK_NUM]={0}; //飞行任务子线程执行时间，可以用于设置某个子线程的执行时间
```

`flight_subtask_cnt` 用于控制飞行任务中具体执行哪一个子线程，比如我们需要对偏航角度进行控制顺时针旋转 90 度，我们可以将转动过程拆解成两个子线程，第一个线程为设置目标偏航角度，第二个任务为执行偏航控制并实时检测偏航控制完成与否，**每一个子线程完成后都会对 `flight_subtask_cnt` 计数器自加**，下次运行任务时会**自动进入下一线程中继续执行未完成的子线程**。

```
33 //*****
34 //顺时针转动90度，完成后降落
35 void flight_subtask_1(void)
36 {
37     static uint8_t n=0;
38     if(flight_subtask_cnt[n]==0) ①
39     {
40         Flight.yaw_ctrl_mode=CLOCKWISE;
41         Flight.yaw_ctrl_start=1;
42         Flight.yaw_outer_control_output =90; //顺时针90度
43
44         OpticalFlow_Control_Pure(0);
45         Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL); //高度控制
46         flight_subtask_cnt[n]=1;
47     }
48     else if(flight_subtask_cnt[n]==1) ②
49     {
50         Flight.yaw_ctrl_mode=CLOCKWISE;
51         Flight.yaw_outer_control_output =0;
52
53         OpticalFlow_Control_Pure(0);
54         Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL); //高度控制
55
56         if(Flight.yaw_ctrl_end==1) flight_subtask_cnt[n]=2; //执行完毕后，切换到下一阶段
57     }
58     else if(flight_subtask_cnt[n]==2) ③
59     {
60         Flight.yaw_ctrl_mode=ROTATE;
61         Flight.yaw_outer_control_output =RC_Data.rc_rpyt[RC_YAW];
62
63         OpticalFlow_Control_Pure(0);
64         Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-30); //高度控制
65     }
66     else
67     {
68         basic_auto_flight_support(); //基本飞行支持软件
69     }
70 }
71
```

新手对任务进行拆分时，实现某一控制任务可能存在着**多种实现方法**，比如在本案例中为了实现偏航逆时针旋转 90 度，我们完成可以不用让 `yaw_ctrl_mode` 工作在 `CLOCKWISE` 模式，直接运用绝对偏航角模式 `AZIMUTH`，在第一步给定期望时，用当前偏航角度 `WP_AHRS.Yaw` 减去 90 作为偏航期望同样可以实现上述控制效果，具体过程大家可以根据自己掌握的熟练程度来针对性的设计。

`flight_global_cnt` 用于子线程中判断是否满足某些条件的计数器，比如当判断条件为位置偏差时，通过 `flight_global_cnt` 的值可以用于判断航点位置是否抵达。

零基础学习竞赛无人机搭积木式编程指南

```
691 else if(flight_subtask_cnt[n]==1) //检测起飞点悬停完毕后,飞向目标A—第21号作业区域
692 {
693     basic_auto_flight_support(); //基本飞行支持软件
694
695     //判断是否到达目标航点位置
696     if(flight_global_cnt[n]<Waypoint_Fix_Cnt1) //持续4*5ms=0.05s满足
697     {
698         float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
699         if(dis_cm<=Waypoint_Fix_CM1) flight_global_cnt[n]++;
700         else flight_global_cnt[n]=2;
701     }
702     else //持续4*5ms满足,表示到达目标航点位置
703     {
704         target_position.x=base_position.x+work_waypoints[0][1];
705         target_position.y=base_position.y+work_waypoints[1][1];
706         target_position.z=base_position.z;
707         Horizontal_Navigation(target_position.x,
708                             target_position.y,
709                             target_position.z,
710                             GLOBAL_MODE,
711                             MAP_FRAME);
712         flight_subtask_cnt[n]++;
713         flight_global_cnt[n]=0;
714     }
715 }
716 else if(flight_subtask_cnt[n]<31) //到达A点所在作业点后,遍历所有航点并检测是否打点
717 {
```

`execute_time_ms` 用于子线程设置子线程的执行时间,比如在上一子线程执行完毕后,对下一子线程执行时间进行设置,在接下来执行下一子线程的过程中,会对 `execute_time_ms` 进行自减, `execute_time_ms` 减到 0 时表示执行时间完毕,用户可以通过判断 `execute_time_ms` 是否为 0 来决策进入下一子任务。

```
206 //本demo适用于激光雷达SLAM定位条件下,普通光流(LC307、LC302)定位条件下无效
207 //机体坐标系下相对位移
208 //右前上分别对应xyz正方向
209 void flight_subtask_5(void)
210 {
211     static uint8_t n=4;
212     if(flight_subtask_cnt[n]==0) ❶
213     {
214         basic_auto_flight_support(); //基本飞行支持软件
215         flight_subtask_cnt[n]=1;
216         execute_time_ms[n]=10000/flight_subtask_delta; //子任务执行时间
217         //向前移动100cm
218         Horizontal_Navigation(0,100,0,RELATIVE_MODE,BODY_FRAME);
219     }
220     else if(flight_subtask_cnt[n]==1) ❷
221     {
222         basic_auto_flight_support(); //基本飞行支持软件
223         if(execute_time_ms[n]>0) execute_time_ms[n]--;
224         if(execute_time_ms[n]==0)
225         {
226             flight_subtask_cnt[n]=2;
227             execute_time_ms[n]=10000/flight_subtask_delta; //子任务执行时间
228             //向右移动100cm
229             Horizontal_Navigation(100,0,0,RELATIVE_MODE,BODY_FRAME);
230         }
231     }
232     else if(flight_subtask_cnt[n]==2) ❸
233     {
234         basic_auto_flight_support(); //基本飞行支持软件
235         if(execute_time_ms[n]>0) execute_time_ms[n]--;
236         if(execute_time_ms[n]==0)
237         {
238             flight_subtask_cnt[n]=3;
239             execute_time_ms[n]=10000/flight_subtask_delta; //子任务执行时间
240             //向后移动100cm
241             Horizontal_Navigation(0,-100,0,RELATIVE_MODE,BODY_FRAME);
242         }
243     }
```

根据上面的讲解,我们知道 `flight_global_cnt` 与 `execute_time_ms` 都可以用于决策子线程是否执行完毕,通常简单一点的子线程只需要用二者之一就可以,特殊一点的情况比如在某子任务内需要结合视觉处理,但是由于自己写的视觉代码识别失败、现场某些不可控因素或者自己设计的搜索路径太过冗杂,导致在有限的时间内,程序不能及时完成本阶段子线程,但是竞赛比赛赛题时间上是有要求的,所以为了保险起见,我们可以在判断位置、视觉标记是否完成的基础上,加上合理的最大执行时间限定,当子线程处理超时会强制进入下一线程或者直接返航降落。

IO 控制函数

`void Laser_Light_Work(_laser_light *light)`

```
4 typedef struct
5 {
6     uint16_t times;           //预设闪烁总次数
7     uint8_t reset;           //闪烁进程复位标志
8     uint16_t cnt;            //闪烁控制计数器
9     uint16_t times_cnt;      //记录已闪烁次数
10    uint8_t end;              //闪烁完成标志位
11    uint32_t port;            //闪烁所在的端口
12    uint8_t pin;              //闪烁所在的GPIO
13    uint32_t period;          //闪烁周期
14    float light_on_percent;    //单个周期内点亮时间百分比
15 } _laser_light;
```

IO 控制初始化过程和赋值用法如下：

```
16 void Reserved_IO_Init(void)
17 {
18     #if RESERVED_IO_ENABLE
19         SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
20         GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2);
21
22         GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_0, 0);
23         GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_1, 0);
24         GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2, 0);
25     #endif
26
27     laser_light_1.port=GPIO_PORTD_BASE;
28     laser_light_1.pin=GPIO_PIN_0;
29     laser_light_1.period=200; //200*5ms
30     laser_light_1.light_on_percent=0.2f;
31     laser_light_1.reset=0;
32
33     laser_light_2.port=GPIO_PORTD_BASE;
34     laser_light_2.pin=GPIO_PIN_1;
35     laser_light_2.period=200; //200*5ms
36     laser_light_2.light_on_percent=0.2f;
37     laser_light_2.reset=0;
38 }
39
1540 else if(flight_subtask_cnt[n]==37) //间隔5s后再次闪烁
1541 {
1542     basic_auto_flight_support(); //基本飞行支持软件
1543     if(execute_time_ms[n]>0) execute_time_ms[n]--;
1544     if(execute_time_ms[n]==0)
1545     {
1546         laser_light_1.reset=1;
1547         laser_light_1.times=barcode_id; //闪烁barcode_id次
1548         flight_subtask_cnt[n]++;
1549         flight_global_cnt[n]=0;
1550     }
1551 }
```

了解了上述自主飞行任务设计关键要点后，我们以 2021 年全国大学生电子设计竞赛中 G 题植保飞行器题目要求为例，编写自动飞行任务函数完成比赛内容。

无名创新

植保飞行器 (G 题)

【本科组】

一 任务

设计一基于四旋翼飞行器的模拟植保飞行器，能够对指定田块完成“撒药”作业。如图1所示作业区中，灰色部分是**非播撒区域**，绿色部分是待“播撒农药”的区域，分成多个50cm×50cm虚线格区块，用1~28数字标识，以全覆盖飞行方式完成播撒作业。作业中播撒区域不得漏撒、重复播撒，非播撒区域不得播撒，否则将扣分；播撒作业完成时间越短越好。

图 1 中,黑底白字的“十”字是飞行器起降点标识;“21”是播撒作业起点区块,用“A”标识;飞行器用启闭可控、垂直向下安装

的激光笔的闪烁光点表示播撒动作,光点在每个区块闪烁 1~3 次视为正常播撒;同一区块光点闪烁次数大于 3 次,将被认定为重复播撒。激光笔光点闪烁周期 1~2s。




图 1 播撒作业区示意图

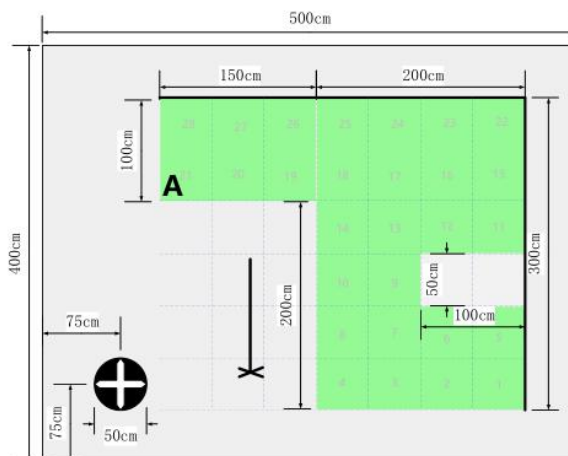


图 1 播撒作业区示意图

二 要求

1. 基本要求

- (1) 飞行器在“十”字起降点垂直起飞, 升空至 $150 \pm 10\text{cm}$ 巡航高度。
- (2) 寻找播撒作业起点, 从“A”所在区块开始“撒药”作业。
- (3) 必须在 360 秒内完成对图 1 中所有绿色区块进行全覆盖播撒。
- (4) 作业完成后稳定准确降落在起降点; 飞行器几何中心点与起降点中心距离的偏差不大于 $\pm 10\text{cm}$ 。

2. 发挥部分

- (1) 将作业区中任意位置的 3~4 个连续播撒区块改为用非播撒区域颜色覆盖, 重复基本要求 (1)~(3) 的作业。
- (2) 在作业区中放置一只高度为 150cm、直径 $3.5 \pm 0.5\text{cm}$ 的黑色杆塔, 杆塔上套有圆环形条形码 (放条码的高度为 120~140cm); 作业中或返航途中, 飞行器识别条形码所表征的数字, 用 LED 闪烁次数显示数字, 间隔数秒后再次闪烁显示。
- (3) 以起降点“十”字中心为圆心, 以上述 (2) 中识别的数字乘 10cm 为半径, 飞行器在该圆周上稳定降落; 飞行器几何中心点与该圆周最近距离的偏差不大于 $\pm 10\text{cm}$ 。
- (4) 在测试现场随机抽取一个项目, 30 分钟内现场完成一组飞行动作任务的编程调试, 并完成飞行动作。
- (5) 其他。

● 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=1;
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位置。第二步为实时检测高度偏差值，连续 2S 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，用法参照 Developer_Mode.c 开发者模式中 case 11 用法，自主起飞任务完成后会进入 case 12 执行航点遍历作业任务。

```
Developer Mode
115 break;
116 case 8:
117 {
118     flight_subtask_5();//机体坐标系下相对位移,正方形轨迹
119 }
120 break;
121 case 9:
122 {
123     flight_subtask_7();//导航坐标系下,基于初始点的绝对坐标位移,正方形轨迹
124 }
125 break;
126 case 10:
127 {
128     flight_subtask_8();//航点飞行轨迹圆,半径、航点数量可设置
129 }
130 break;
131 case 11://自动起飞到某一高度
132 {
133     if(Auto_Takeoff(WORK_HEIGHT_CM)==1)
134     {
135         *mode+=1;//到达目标高度后,切换到下一SDK任务
136         SDK_DT_Send_Check(0x07,UART3_SDK);//起飞完毕之后,将底部OPENMV设置成检测农作物模式
137     }
138 }
139 break;
140 case 12:
141 {
142     //2021年电子设计竞赛g题植保无人机
143     Agriculture_UAV_Closetloop();//基础部分
144     //Agriculture_UAV_Innovation();//发挥部分
145 }
146 break;
147 case 13:
148 {
```

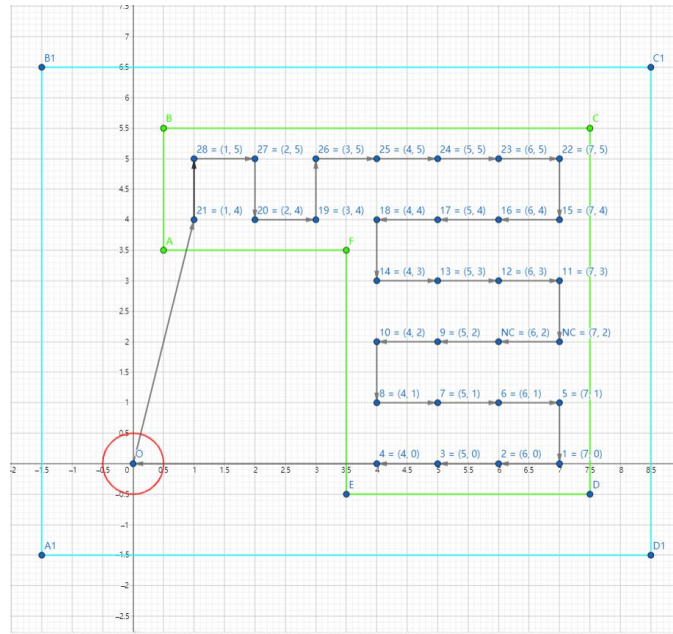
● 第二阶段——航点遍历作业任务

void Agriculture_UAV_Closetloop(void)

第一步生成航点轨迹数组,记录当前高度 **height**,并发布第一个目标航点位置 **0,0,height**),接着会在第二线程中判断水平位置误差,来确定完成水平航点任务与否。检测悬停目标完成后会发布 **21** 号区域位置航点。

```
521 /*****
522 const int16_t work_waypoints_table[2][32]={
523 {0 1,1,2,2,3,3,4,5,6,7,7,6,5,4,4,5,6,7,7,6,5,4,4,5,6,7,7,6,5,4,0},
524 {0 4,5,5,4,4,5,5,5,5,5,4,4,4,4,3,3,3,3,2,2,2,2,1,1,1,1,0,0,0,0}
525 };
526 float work_waypoints[2][32]={0};
527 uint16_t work_time_gap[32]={
528 3000,
529 10000,
530 3000,3000,3000,3000,3000,3000,3000,3000,3000,
531 3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,
532 3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,
533 10000
534 };
535 #define Scale_Param 50.0f
536 void work_waypoint_generate(void)
537 {
538     for(uint16_t i=0;i<32;i++)
539     {
540         work_waypoints[0][i]=Scale_Param*work_waypoints_table[0][i];
541         work_waypoints[1][i]=Scale_Param*work_waypoints_table[1][i];
542     }
543 }
544 }
545
```

零基础学习竞赛无人机搭积木式编程指南



```
666 void Agriculture_UAV_Closetloop(void)
667 {
668     static uint8_t n=10;
669     Vector3f target_position;
670     if(flight_subtask_cnt[n]==0)//起飞点作为第一个悬停点
671     {
672         basic_auto_flight_support();//基本飞行支持软件
673         work_waypoint_generate();
674         //记录下初始起点位置，实际项目中可设置为某一基准原点
675         //base_position.x=VIO_SINS.Position[EAST];
676         //base_position.y=VIO_SINS.Position[NORTH];
677         base_position.z=NamelessQuad.Position[UP];
678
679         execute_time_ms[n]=work_time_gap[0]/flight_subtask_delta;//子任务执行时间
680         target_position.x=base_position.x+work_waypoints[0][0];
681         target_position.y=base_position.y+work_waypoints[1][0];
682         target_position.z=base_position.z;
683         Horizontal_Navigation(target_position.x,
684                             target_position.y,
685                             target_position.z,
686                             GLOBAL_MODE,
687                             MAP_FRAME);
688         flight_subtask_cnt[n]=1;
689         flight_global_cnt[n]=0;
690     }
691     else if(flight_subtask_cnt[n]==1)//检测起飞点悬停完毕后，飞向目标A——第21号作业区域
692     {
693         basic_auto_flight_support();//基本飞行支持软件
694
695         //判断是否到达目标航点位置
696         if(flight_global_cnt[n]<Waypoint_Fix_Cnt1)//持续4*5ms=0.05s满足
697         {
698             float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
699             if(dis_cm<Waypoint_Fix_Cm1) flight_global_cnt[n]++;
700             else flight_global_cnt[n]=2;
701         }
702         else//持续4*5ms满足，表示到达目标航点位置
703         {
704             target_position.x=base_position.x+work_waypoints[0][1];
705             target_position.y=base_position.y+work_waypoints[1][1];
706             target_position.z=base_position.z;
707             Horizontal_Navigation(target_position.x,
708                                 target_position.y,
709                                 target_position.z,
710                                 GLOBAL_MODE,
711                                 MAP_FRAME);
712             flight_subtask_cnt[n]++;
713             flight_global_cnt[n]=0;
714         }
715     }
```

第二步无人机飞向 21 号作业点，同时实时检测无人机是否到达 21 号航点，达到后判断底部视觉 OPENMV 识别底部颜色情况，当无人机正下方为农作物时，会设置激光笔闪烁打点。随即会依次发布 28、27、20、19、26、25、...、1、2、3、4 航点任务，飞机会依次飞到对用航点上见结合 OPENMV 识别情况决策激光笔打点与否。

无名创新

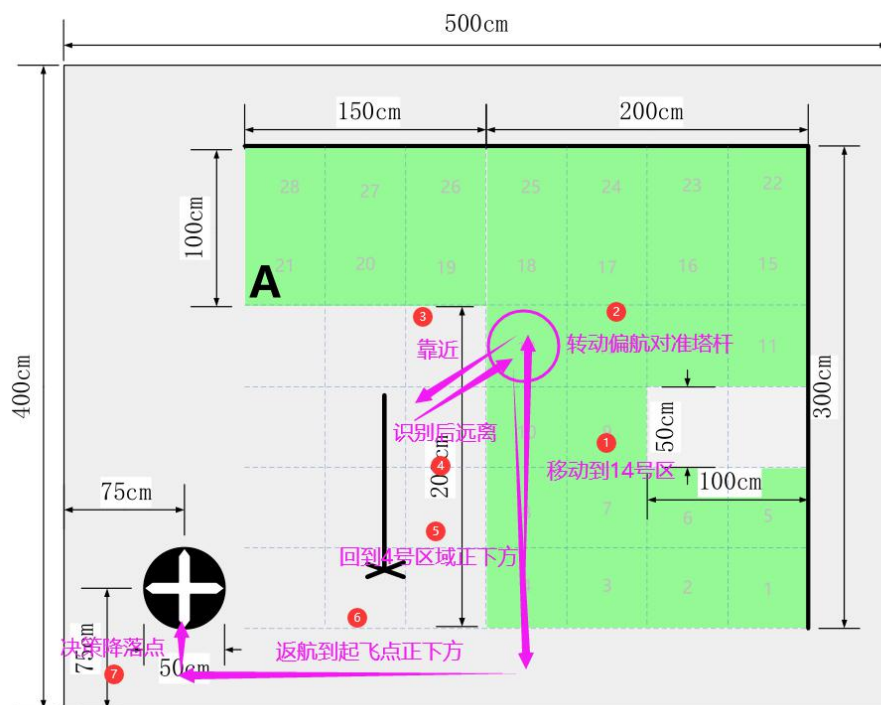
零基础学习竞赛无人机搭积木式编程指南

```
716 else if(flight_subtask_cnt[n]<31)//到达a点所在作业点后，遍历所有航点并检测是否打点
717 {
718     basic_auto_flight_support();//基本飞行支持软件
719     //判断是否到达目标航点位置
720     if(flight_global_cnt[n]<Waypoint_Fix_Cnt1)//持续4*5ms=0.05s满足
721     {
722         float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
723         if(dis_cm<=Waypoint_Fix_CM1) flight_global_cnt[n]++;
724         else flight_global_cnt[n]/=2;
725     }
726     else//持续4*5ms满足，表示到达目标航点位置
727     {
728         execute_time_ms[n]=work_time_gap[flight_subtask_cnt[n]]/flight_subtask_delta;//子任务执行时间
729         target_position.x=base_position.x+work_waypoints[0][flight_subtask_cnt[n]];
730         target_position.y=base_position.y+work_waypoints[1][flight_subtask_cnt[n]];
731         target_position.z=base_position.z;
732         Horizontal_Navigation(target_position.x,
733                               target_position.y,
734                               target_position.z,
735                               GLOBAL_MODE,
736                               MAP_FRAME);
737         flight_subtask_cnt[n]++;
738         flight_global_cnt[n]=0;
739         if(Opv_Top_View_Target.trust_flag==1)//到达目标航点后，判断底部是否为农作物
740         {
741             laser_light_1.reset=1;
742             laser_light_1.times=1;//闪烁1次
743         }
744     }
745 }
746 else if(flight_subtask_cnt[n]==31)//到达最后一个航点单独处理
747 {
748     basic_auto_flight_support();//基本飞行支持软件
749     //判断是否到达目标航点位置
750     if(flight_global_cnt[n]<Waypoint_Fix_Cnt1)//持续4*5ms=0.05s满足
751     {
752         float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
753         if(dis_cm<=Waypoint_Fix_CM1) flight_global_cnt[n]++;
754         else flight_global_cnt[n]/=2;
755     }
756     else//持续4*5ms满足，表示到达目标航点位置
757     {
758         execute_time_ms[n]=1000/flight_subtask_delta;//子任务执行时间
759         flight_subtask_cnt[n]++;
760         flight_global_cnt[n]=0;
761         if(Opv_Top_View_Target.trust_flag==1)//到达目标航点后，判断底部是否为农作物
762         {
763             laser_light_1.reset=1;
764             laser_light_1.times=1;//闪烁1次
765         }
766     }
767 }
```

第三步 4 号区域作业点遍历完毕之后，会发布导航原点坐标，飞到起飞点正上方后，判断是否对准，位置对准后飞机原地降落，直至落到地面后触发地面检测条件后会自动给无人机上锁停桨。

```
769 else if(flight_subtask_cnt[n]==32)//最后返回起飞点
770 {
771     basic_auto_flight_support();//基本飞行支持软件
772
773     if(execute_time_ms[n]>0) execute_time_ms[n]--;
774     if(execute_time_ms[n]==0)
775     {
776         execute_time_ms[n]=work_time_gap[31]/flight_subtask_delta;//子任务执行时间
777         target_position.x=base_position.x+work_waypoints[0][31]; //导航原点
778         target_position.y=base_position.y+work_waypoints[1][31];
779         target_position.z=base_position.z;
780         Horizontal_Navigation(target_position.x,
781                               target_position.y,
782                               target_position.z,
783                               GLOBAL_MODE,
784                               MAP_FRAME);
785         flight_subtask_cnt[n]++;
786     }
787 }
788 else if(flight_subtask_cnt[n]==33)//到达起飞点
789 {
790     basic_auto_flight_support();//基本飞行支持软件
791     //判断是否到达目标航点位置
792     if(flight_global_cnt[n]<Waypoint_Fix_Cnt1)//持续4*5ms=0.05s满足
793     {
794         float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
795         if(dis_cm<=Waypoint_Fix_CM1) flight_global_cnt[n]++;
796         else flight_global_cnt[n]/=2;
797     }
798     else
799     {
800         flight_subtask_cnt[n]++;
801         flight_global_cnt[n]=0;
802     }
803 }
804 else if(flight_subtask_cnt[n]==34)//原地降落
805 {
806     Flight.yaw_ctrl_mode=ROTATE;
807     Flight.yaw_outer_control_output =RC_Data.rc_rpyt[RC_YAW];
808     OpticalFlow_Control_Pure(0);
809     Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-30);//高度控制
810 }
811 else
812 {
813     basic_auto_flight_support();//基本飞行支持软件
814 }
815 }
```


● 发挥部分



```
void Agriculture UAV Innovation(void)//发挥部分
```

发挥题目部分的前三个阶段和基础部分一致，故不再赘述，我们直接从4号区域播撒完毕开始解析，首先飞机会降低巡航高度，到达巡航高度后会进入下一子进程。

```

1448     else if(flight_subtask_cnt[n]==32)//首先降低飞行高度到130cm
1449     {
1450         basic_auto_flight_support();//基本飞行支持软件
1451         target_position.x=base_position.x+work_waypoints[0][30];
1452         target_position.y=base_position.y+work_waypoints[1][30];
1453         target_position.z=Barcode_Height_CM;//条形码所在高度
1454         Horizontal_Navigation(target_position.x,
1455                               target_position.y,
1456                               target_position.z,
1457                               GLOBAL_MODE,
1458                               MAP_FRAME);
1459         flight_subtask_cnt[n]++;
1460         flight_global_cnt[n]=0;
1461     }
1462     else if(flight_subtask_cnt[n]==33)//判断是否到达目标高度
1463     {
1464         basic_auto_flight_support();//基本飞行支持软件
1465         //判断是否到达目标高度
1466         if(flight_global_cnt[n]<100)//持续100*5ms满足
1467         {
1468             if(fabs(Total_Controller.High_Position_Control.Err)<=10.0f) flight_global_cnt[n]++;
1469             else flight_global_cnt[n]/=2;
1470         }
1471         else//持续100*5ms满足，表示到达目标高度
1472         {
1473             flight_subtask_cnt[n]++;
1474             flight_global_cnt[n]=0;
1475         }
1476     }

```

到达目标高度后，飞机会飞到 14 号作业区域，达到后无人机会采用激光雷达去识别塔杆的距离和角度方位。这里飞向 14 号区域的目的是使得无人机飞到赛场中心位置，避免比赛现场人员靠得太近而造成塔杆误检或者额外增加不必要的逻辑处理过程。

在无人机到达 14 作业区域后，无人机会根据检测到的塔杆方位、距离信息，首先控制无人机机头对准塔杆，然后再靠近塔杆，控制无人机中心与塔杆的最短距离为 50cm。此过程过程中前向机器视觉 OPENMV 会实时检测是否有识别到条形码信息，对准和靠近过程中只要有识别到条码信息，就会提前结束当前子线程。

零基础学习竞赛无人机搭积木式编程指南

```
1480 target_position.x=base_position.x+4*Scale_Param;
1481 target_position.y=base_position.y+3*Scale_Param;
1482 target_position.z=Barcode_Height_CM;
1483 Horizontal_Navigation(target_position.x,
1484                       target_position.y,
1485                       target_position.z,
1486                       GLOBAL_MODE,
1487                       MAP_FRAME);
1488 //判断是否到达目标航点位置
1489 if(flight_global_cnt[n]<Waypoint_Fix_Cnt2)//持续10*5ms=0.05s满足
1490 {
1491     float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
1492     if(dis_cm<Waypoint_Fix_CM2) flight_global_cnt[n]++;
1493     else flight_global_cnt[n]/=2;
1494 }
1495 }
1496 else
1497 {
1498     flight_subtask_cnt[n]++;
1499     flight_global_cnt[n]=0;
1500 }
1501 else if(flight_subtask_cnt[n]==35)//首先控制偏航运动,使得飞机头部对准塔杆
1502 {
1503     float expect_yaw_gyro=Total_Controller.Yaw_Angle_Control.Kp*target_yaw_err;//期望偏航角速度
1504     flight_yaw_ctrl_mode=ROTATE;
1505     flight_yaw_outer_control_output=constrain_float(expect_yaw_gyro,-10,10);//以10deg/s的角度顺时针转动10000ms
1506     if(fabs(target_yaw_err)<5.0)//只有当偏航角度比较小时,才靠近杆
1507     {
1508         float dis_err=min_dis_cm=50;
1509         dis_err=constrain_float(dis_err,-20,20);
1510         OpticalFlow_Y_Vel_Control(Total_Controller.Optical_Position_Control.Kp*dis_err);
1511         OpticalFlow_X_Vel_Control(0);
1512         fix_flag=1;
1513     }
1514     else//否则原地悬停
1515     {
1516         OpticalFlow_Control_Pure(fix_flag);
1517         fix_flag=0;
1518     }
1519     flight_alt_hold_control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制
1520 }
1521 if(barcode_flag==1)//只要识别到了二维码,提前结束对准塔杆、靠近任务
1522 {
1523     flight_subtask_cnt[n]=36;
1524     flight_global_cnt[n]=0;
1525 }
1526 }
1527 }
```

机头对准+靠近塔杆 ①

只要有识别到条形码信息随时退出 ②

识别到条形码信息后,无人机会连续两次闪烁条形码表征的id信息,两次闪烁的间隔时间为3S。

```
1528 else if(flight_subtask_cnt[n]==36)
1529 {
1530     basic_auto_flight_support();//基本飞行支持软件
1531     laser_light_1.reset=1;
1532     laser_light_1.times=barcode_id;//闪烁barcode_id次
1533     execute_time_ms[n]=(barcode_id*1000+3000)/flight_subtask_delta;
1534     flight_subtask_cnt[n]++;
1535     flight_global_cnt[n]=0;
1536 }
1537 else if(flight_subtask_cnt[n]==37)//间隔5s后再次闪烁
1538 {
1539     basic_auto_flight_support();//基本飞行支持软件
1540     if(execute_time_ms[n]>0) execute_time_ms[n]--;
1541     if(execute_time_ms[n]==0)
1542     {
1543         laser_light_1.reset=1;
1544         laser_light_1.times=barcode_id;//闪烁barcode_id次
1545         flight_subtask_cnt[n]++;
1546         flight_global_cnt[n]=0;
1547     }
1548 }
```

间隔3S

首次闪烁激光笔的过程中,飞机会向后回退到14号作业区域,有的同学可能会好奇,为什么这里并没有再次发布14号区域航点,只有基本自动飞行支持函数,为什么会有自动退回到刚开始对准机头的位置的控制效果呢?这里还是需要回到上一子线程分析:

```
else if(flight_subtask_cnt[n]==34)//飞到地图中间14号位置,避免现场人员站立太近,激光雷达扫描误以为是距离最近障碍
{
    basic_auto_flight_support();//基本飞行支持软件
    target_position.x=base_position.x+4*Scale_Param;
    target_position.y=base_position.y+3*Scale_Param;
    target_position.z=Barcode_Height_CM;
    Horizontal_Navigation(target_position.x,
                        target_position.y,
                        target_position.z,
                        GLOBAL_MODE,
                        MAP_FRAME);
    //判断是否到达目标航点位置
    if(flight_global_cnt[n]<Waypoint_Fix_Cnt2)//持续10*5ms=0.05s满足
    {
        float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
        if(dis_cm<Waypoint_Fix_CM2) flight_global_cnt[n]++;
        else flight_global_cnt[n]/=2;
    }
    else
    {
        flight_subtask_cnt[n]++;
        flight_global_cnt[n]=0;
    }
}
```

```

1501 else if(flight_subtask_cnt[n]==35)//首先控制偏航运动,使得飞机头部对准塔杆
1502 {
1503     float expect_yaw_gyro=Total_Controller.Yaw_Angle_Control.Kp*target_yaw_err;//期望偏航角速度
1504     Flight.yaw_ctrl_mode=ROTATE;
1505     Flight.yaw_outer_control_output=constrain_float(expect_yaw_gyro,-10,10);//以10deg/s的角速度顺时针转动10000ms
1506     if (ABS(target_yaw_err)<=5.0f)//只有当偏航角度比较小时,才靠近杆
1507     {
1508         float dis_err=min_dis_cm-50;
1509         dis_err=constrain_float(dis_err,-20,20);
1510         OpticalFlow_Y_Vel_Control(Total_Controller.Optical_Position_Control.Kp*dis_err);
1511         OpticalFlow_X_Vel_Control(0);
1512         fix_flag=1;
1513     }
1514     else//否则原地悬停
1515     {
1516         OpticalFlow_Control_Pure(fix_flag);
1517         fix_flag=0;
1518     }
1519     Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);
1520 }
1521
1522 if (barcode_flag==1)//只要识别到了二维码,提前结束对准塔杆、靠近任务
1523 {
1524     flight_subtask_cnt[n]=36;
1525     flight_global_cnt[n]=0;
1526 }
1527 }
    
```

在 34 子线程中,我们有对 target_position.x, target_position.y 进行赋值,并发布航点位置。控制完毕后会进入 35 子线程中。无人机在机头对准塔杆的过程中,开始时 target_yaw_err 会大于 5° , 所以在下方位置-速度控制部分首先会执行②中位置控制部分内容,由于 fix_flag 初始值为 0,故 OpticalFlow_Control_Pure(fix_flag)控制效果就是原地悬停,悬停锁定的位置即为 34 子线程中发布的 14 号作业区。

接着无人机会继续转动直到机头与塔杆的角度偏差小于 5° , 此时无人机水平方向会进入③中的水平速度控制模式,无人机前后速度期望来源于前后方向的位置偏差计算出的位置比例控制的输出值;左右方向速度期望为 0,即希望无人机左右不发生调整。

在无人机各项参数稳定的前提下,一次完整的机头对准—靠近塔杆控制过程中,有且只会存在一次从②→①的过程,即不存在飞机在靠近的过程中,机头与塔杆之间的方向角不会再次出现大于 5° 的情况。当且只存在一次只对准塔杆(大角度偏差)→对准的同时靠近塔杆(小角度偏差)时,即不存在②→①→②这一过程,全程位置期望不会被刷新掉,即位置期望仍为 34 子线程中的位置刷新值,这种情况下当识别到条形码后,进入 36 子线程后,就可以实现自动回退到 14 号作业区的效果。当然此处完全可以用位置导航函数 Horizontal_Navigation 重新发布一个 14 号作业点坐标,实现主动回退到 14 作业区。

在第二次闪烁条形码 id 值后,无人机会飞到 4 号作业区正下方的区块(200,-50,height)上,然后飞到起飞点正下方的区块(0,-50,height)上。

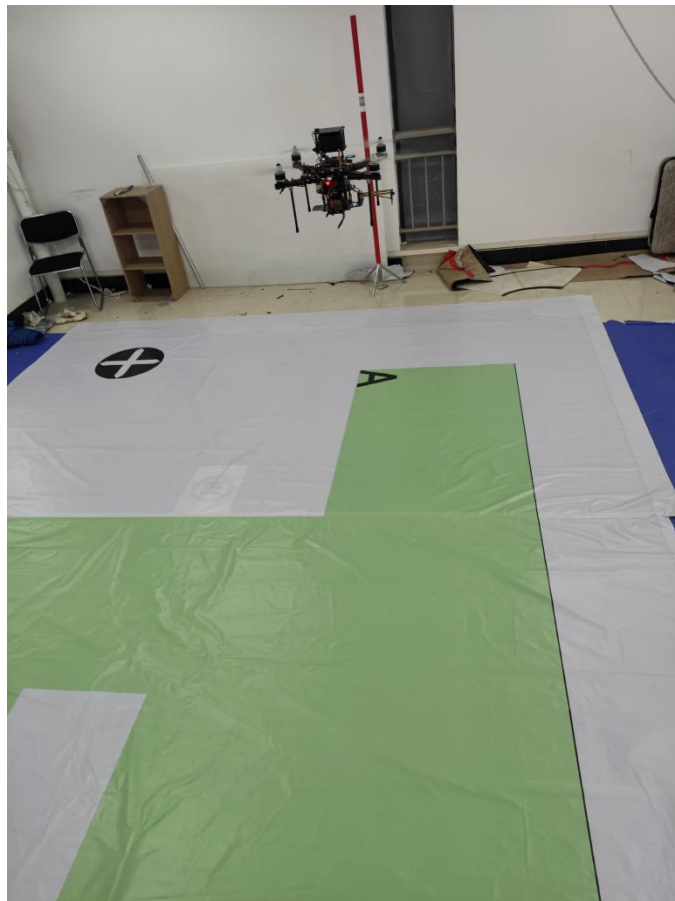
零基础学习竞赛无人机搭积木式编程指南

```
1552 else if(flight_subtask_cnt[n]==38) //退回到4号的下方格子
1553 {
1554     basic_auto_flight_support(); //基本飞行支持软件
1555     target_position.x=base_position.x+4*Scale_Param;
1556     target_position.y=base_position.y-1*Scale_Param;
1557     target_position.z=Barcode_Height_CM;
1558     Horizontal_Navigation(target_position.x,
1559                           target_position.y,
1560                           target_position.z,
1561                           GLOBAL_MODE,
1562                           MAP_FRAME);
1563     //判断是否到达目标航点位置
1564     if(flight_global_cnt[n]<Waypoint_Fix_Cnt2) //持续10*5ms=0.05s满足
1565     {
1566         float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
1567         if(dis_cm<=Waypoint_Fix_CM2) flight_global_cnt[n]++;
1568         else flight_global_cnt[n]/=2;
1569     }
1570     else
1571     {
1572         flight_subtask_cnt[n]++;
1573         flight_global_cnt[n]=0;
1574     }
1575 }
1576 else if(flight_subtask_cnt[n]==39) //退回到起飞点的下方格子
1577 {
1578     basic_auto_flight_support(); //基本飞行支持软件
1579     target_position.x=base_position.x;
1580     target_position.y=base_position.y-1*Scale_Param;
1581     target_position.z=Barcode_Height_CM;
1582     Horizontal_Navigation(target_position.x,
1583                           target_position.y,
1584                           target_position.z,
1585                           GLOBAL_MODE,
1586                           MAP_FRAME);
1587     //判断是否到达目标航点位置
1588     if(flight_global_cnt[n]<Waypoint_Fix_Cnt2) //持续10*5ms=0.05s满足
1589     {
1590         float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
1591         if(dis_cm<=Waypoint_Fix_CM2) flight_global_cnt[n]++;
1592         else flight_global_cnt[n]/=2;
1593     }
1594     else
1595     {
1596         flight_subtask_cnt[n]++;
1597         flight_global_cnt[n]=0;
1598     }
1599 }
```

到达起飞点正下方的区块(0,-50,height)后,无人机会根据条形码 id 值计算出降落点并发布降落点坐标,无人机会飞到降落点上方后自动降落到底面完成整个飞行任务。

在熟练掌握本案例中的各个 API 用法前提下,用户进行二次开发可以把机载计算机端当做一个能提供高精度位置信息的传感器来使用就可以,新手用户可以实现不用在机载计算机下编写代码的情况下,就可以完成比赛中的所有内容。在有一定基础积累后,就可以考虑在机载计算机下用 ROS 系统做导航、避障、路径规划去实现比赛内容,解决问题的途径会更多,处理起来也会更加灵活,另外利用机载计算机端单目摄像头 OPENCV 处理视觉问题会比传统 OPENMV 更加强大、高效。

```
1600 else if(flight_subtask_cnt[n]==40) //退回到起飞点正下方barcode_id*10处悬停
1601 {
1602     basic_auto_flight_support(); //基本飞行支持软件
1603     target_position.x=base_position.x;
1604     target_position.y=base_position.y-10*barcode_id;
1605     target_position.z=Barcode_Height_CM;
1606     Horizontal_Navigation(target_position.x,
1607                           target_position.y,
1608                           target_position.z,
1609                           GLOBAL_MODE,
1610                           MAP_FRAME);
1611     //判断是否到达目标航点位置
1612     if(flight_global_cnt[n]<Waypoint_Fix_Cnt2) //持续10*5ms=0.05s满足
1613     {
1614         float dis_cm=pythagorous2(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y);
1615         if(dis_cm<=Waypoint_Fix_CM2) flight_global_cnt[n]++;
1616         else flight_global_cnt[n]/=2;
1617     }
1618     else
1619     {
1620         flight_subtask_cnt[n]++;
1621         flight_global_cnt[n]=0;
1622     }
1623 }
1624 else if(flight_subtask_cnt[n]==41) //原地降落
1625 {
1626     Flight.yaw_ctrl_mode=ROTATE;
1627     Flight.yaw_outer_control_output =RC_Data.rc_rpyt[RC_YAW];
1628     OpticalFlow_Control_Pure(0);
1629     Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-30); //高度控制
1630 }
1631 else
1632 {
1633     basic_auto_flight_support(); //基本飞行支持软件
1634 }
1635 }
1636 }
```



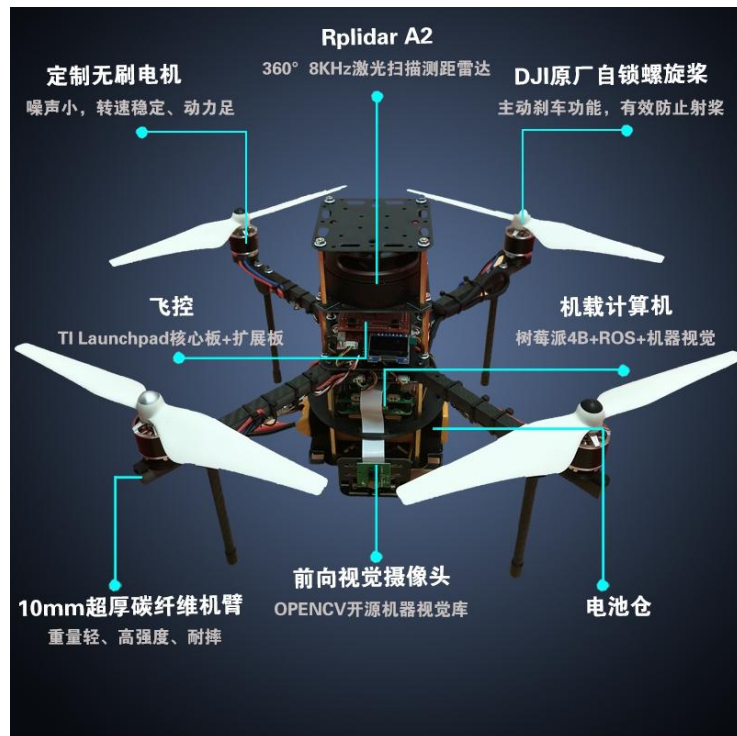
电赛 G 题植保无人机国奖标准方案学习样例

https://www.bilibili.com/video/BV11T4y1v7uW?spm_id_from=333.999.0.0

https://www.bilibili.com/video/BV1QR4y1F7vj?spm_id_from=333.999.0.0

无名创新

NC360 深度开源竞赛无人机开发平台



产品介绍详情页面

<https://item.taobao.com/item.htm?spm=a230r.1.14.1.48ee75f83ITHvO&id=669633894747&ns=1&abbucket=1#detail>