竞赛无人机搭积木式编程（三）

---用户自定义航点自动飞行功能（全局定位，指哪打哪）

无名小哥 2023 年 6 月 10 日

用户通过对前面两讲中全国大学生电子设计竞赛真题植保无人机（2021）、送货无人机（2022）完整方案的学习。细心一点的客户可以发现：在激光雷达 SLAM/T265 双目相机提供全局定位数据的情况下，无人机的自主飞行部分的程序设计，基本都是通过飞控代码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实现。

同时需要结合底部/前向的机器视觉传感器以及激光雷达传感器对目标特征进行视觉/距离定位。比如送货无人机赛题中，达到目标航点上方后，通过视觉实现无人机的精准目标定位。通过视觉实现无人机位置的二次对准，其实就是用到的前些年赛题中的追踪移动色块的功能；

另外在植保无人机发挥部分要求中，需要识别到塔杆和条形码信息。通过激光雷达传感器识别道塔杆的水平位置、相对无人机机头方向的角度，对无人机偏航方向和机头与杆之间的距离进行控制，进而实现机头对准塔杆并调整与杆之间的间距并通过视觉识别特征这一复合的运动，这里用的就是基础飞行控制函数中的偏航控制 API 和速度控制 API 函数予以实现。

为了方便萌新用户针对具体竞赛内容实现快速的二次开发，我们新增加了用户自定义航点飞行功能，用户可以不需要自己编程去改飞控代码，就可以实现航点参数的录入，无人机能按照用户录入的航点动作进行自主飞行。这里需要注意的是默认提供的自定义航点自动飞行函数中飞行动作只是对航点的依次遍历，尚不涉及中间动作，比如需要结合视觉/激光雷达进行视觉定位、距离定位。

用户自定义航点自动飞行功能提供的是一个精简化的自主飞行框架，用户可以快速实现飞行动作的修改和编排。针对某一比赛任务设计中航点飞行任务部分用户可以无需重复设计，**直接参考本框架就可以轻松实现多航点目标自动飞行任务，留足更多的时间去着手机器视觉部分和更为精细化的任务设计**。另外电赛**国赛测评中要求参赛者去现场编程，快速实现某一飞行动作要求变得不再有任何压力**。

12_用户通过 ADC 按键录入自定义航点飞行功能——支持现场设置坐标参数（全局定位，指哪打哪）

演示视频：https://www.bilibili.com/video/BV1wP411z7jo/

1 五向按键的检测设计与航点参数调节界面设计

根据用户需要现场高效率的录入航点参数的这一具体要求，我们将上一版本ADC 安全绳按键进行了升级，在保留原有解锁、开发、上锁、降落四个独立按键的功能的同时，新增加了一路独立的五向按键采集 ADC 通道，一个 IO 口实现了 5 个方向按键的检测，并可配备有短按、长按、持续按、多次点动触发等，进一步丰富了按键键值。五向按键配合扩展版上 OLED 显示屏可方便实现多个参数的修改与保存。

1.1ADC 按键检测原理

ADC 按键的检测原理为每个按键串入不同的电阻值，按键按下后偏置电压经过电阻分压作用后，由单片机的 ADC 端口采集，飞控程序可以通过采集到的 ADC 值在某个区间范围来判断是哪个按键按下，并对按下的持续时间、次数灯进行逻辑处理，实现按键事件的响应。这部分原理以往教程有详细的讲解和分析，不属于本文重点，这里不再展开，用户可以参照以下链接自行学习。

电赛飞行器安全绳+无遥控器按键控制方案

https://www.bilibili.com/read/cv11399668

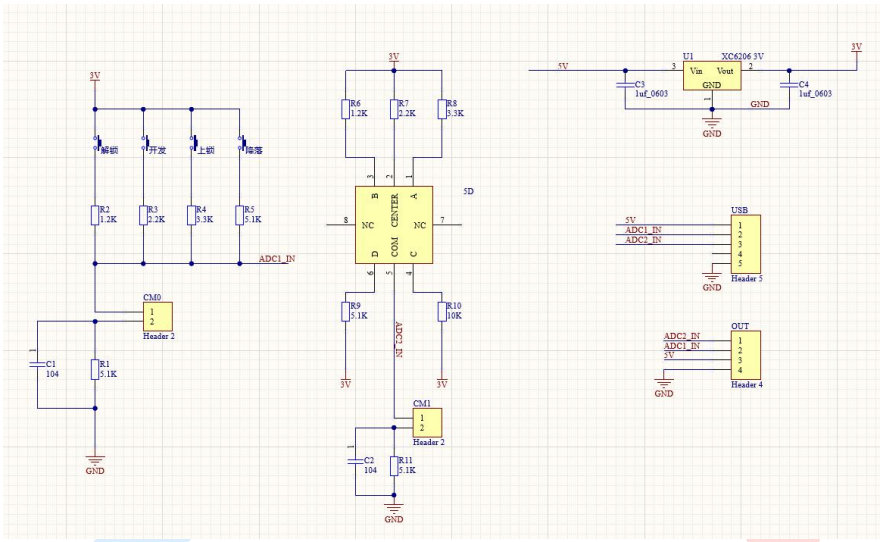
电赛禁止“无线通讯及遥控”解决方案

https://www.bilibili.com/read/cv7897817

电赛神器 ADC 安全绳按键无遥控器控制方案

https://www.bilibili.com/video/BV1Gq4y1E7XM

```
#if (ADC_KEY_VERSION==NEW_AFTER_20230501)
  if(Button_ADCResult[0]<=0.5f)        Key_Value[0]=0x00;//没有按键按下，下拉5.1K，输入0V
  else if(Button_ADCResult[0]<=1.70f)  Key_Value[0]=0x04;//降落按键按下，   上接5.1K，输入3.3V*5.1/10.2=1.65V
  else if(Button_ADCResult[0]<=2.10f)  Key_Value[0]=0x02;//上锁按键按下，   上接3.3K，输入3.3V*5.1/8.40=2.00V
  else if(Button_ADCResult[0]<=2.40f)  Key_Value[0]=0x03;//开发按键按下，   上接2.2K，输入3.3V*5.1/7.30=2.31V
  else if(Button_ADCResult[0]<=3.00f)  Key_Value[0]=0x01;//解锁按键按下，   上接1.2K，输入3.3V*5.1/6.30=2.67V
  else Key_Value[0]=0x00;
#endif
```



1.2 航点参数界面设计

OLED 显示屏参数显示分为 8 行，第 1 行显示内容为页面和页码提示，第 2 行至第 8 行数据为 1-7 个航点的 ENU（等效东北天）方向的坐标信息，其中 EN 方向的输入参数表示相对初始基准点的位置增/减量。它是一个相对位置坐标参数，并非实际无人机内部实时位置，二者在水平位置上相差一个初始基准点的坐标(base_position.x,base_position.y)。U 方向输入的是绝对坐标信息，即无人机离地面的高度值。


```
case 25:
{
  uint16_t base_item=50;
  LCD_clear_L(0,0);   display_6_8_string(0,0,"nav_point:1/7 e-n-u");write_6_8_number(115,0,Page_Number+1);
  LCD_clear_L(0,1);   display_6_8_string(0,1,"p1");               write_6_8_number(25,1,param_value[base_item+0]);  write_6_8_number(60,1,param_value[base_item+1]);  write_6_8_number(95,1,param_value[base_item+2]);
  LCD_clear_L(0,2);   display_6_8_string(0,2,"p2");               write_6_8_number(25,2,param_value[base_item+3]);  write_6_8_number(60,2,param_value[base_item+4]);  write_6_8_number(95,2,param_value[base_item+5]);
  LCD_clear_L(0,3);   display_6_8_string(0,3,"p3");               write_6_8_number(25,3,param_value[base_item+6]);  write_6_8_number(60,3,param_value[base_item+7]);  write_6_8_number(95,3,param_value[base_item+8]);
  LCD_clear_L(0,4);   display_6_8_string(0,4,"p4");               write_6_8_number(25,4,param_value[base_item+9]);  write_6_8_number(60,4,param_value[base_item+10]); write_6_8_number(95,4,param_value[base_item+11]);
  LCD_clear_L(0,5);   display_6_8_string(0,5,"p5");               write_6_8_number(25,5,param_value[base_item+12]); write_6_8_number(60,5,param_value[base_item+13]); write_6_8_number(95,5,param_value[base_item+14]);
  LCD_clear_L(0,6);   display_6_8_string(0,6,"p6");               write_6_8_number(25,6,param_value[base_item+15]); write_6_8_number(60,6,param_value[base_item+16]); write_6_8_number(95,6,param_value[base_item+17]);
  LCD_clear_L(0,7);   display_6_8_string(0,7,"p7");               write_6_8_number(25,7,param_value[base_item+18]); write_6_8_number(60,7,param_value[base_item+19]); write_6_8_number(95,7,param_value[base_item+20]);

  static uint16_t ver_item=1;
  if(ver_item==1)   display_6_8_string(18,1,"*"); else if(ver_item==2)  display_6_8_string(53,1,"*"); else if(ver_item==3)  display_6_8_string(88,1,"*");
  if(ver_item==4)   display_6_8_string(18,2,"*"); else if(ver_item==5)  display_6_8_string(53,2,"*"); else if(ver_item==6)  display_6_8_string(88,2,"*");
  if(ver_item==7)   display_6_8_string(18,3,"*"); else if(ver_item==8)  display_6_8_string(53,3,"*"); else if(ver_item==9)  display_6_8_string(88,3,"*");
  if(ver_item==10)  display_6_8_string(18,4,"*"); else if(ver_item==11) display_6_8_string(53,4,"*"); else if(ver_item==12) display_6_8_string(88,4,"*");
  if(ver_item==13)  display_6_8_string(18,5,"*"); else if(ver_item==14) display_6_8_string(53,5,"*"); else if(ver_item==15) display_6_8_string(88,5,"*");
  if(ver_item==16)  display_6_8_string(18,6,"*"); else if(ver_item==17) display_6_8_string(53,6,"*"); else if(ver_item==18) display_6_8_string(88,6,"*");
  if(ver_item==19)  display_6_8_string(18,7,"*"); else if(ver_item==20) display_6_8_string(53,7,"*"); else if(ver_item==21) display_6_8_string(88,7,"*");
```

```
        execute_time_ms[n]=0;
        //航点计数器清0
        flight_global_cnt2[n]=0;
        return ;
      }

      uint16_t current_num=constrain_int32(flight_global_cnt2[n],0,user_setpoint_max-1);//限幅防溢出
      if(user_setpoint_valid_flag[current_num]==true)//如果当前的航点有效，就设置目标航点进入下一线程
      {
        x=nav_setpoint[current_num][0];
        y=nav_setpoint[current_num][1];
        z=nav_setpoint[current_num][2];
        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;
      }
```

五向按键的中的上下按键短按可以实现某一参数的选中，*提示光标会移动到待调整参数的前面。

```
//通过3D按键来实现换行选中待修改参数
if(_button.state[UP_3D].press==SHORT_PRESS)
{
    _button.state[UP_3D].press=NO_PRESS;
    ver_item--;
    if(ver_item<1) ver_item=21;
    Bling_Set(&Light_2,500,50,0.2,0,GPIO_PORTF_BASE,GPIO_PIN_2,0);//蓝色
}
if(_button.state[DN_3D].press==SHORT_PRESS)
{
    _button.state[DN_3D].press=NO_PRESS;
    ver_item++;
    if(ver_item>21) ver_item=1;
    Bling_Set(&Light_2,500,50,0.2,0,GPIO_PORTF_BASE,GPIO_PIN_2,0);//蓝色
}
```

五向按键的中的左右按键短按/长按可以实现某一参数的选中的自加、自减，其中短按是自加/减 1，长按是自加/减 50。

```
//通过左按键短按可以实现选中的参数行自减小1调整
if(_button.state[LT_3D].press==SHORT_PRESS)
{
  _button.state[LT_3D].press=NO_PRESS;
   param_value[base_item+ver_item-1]-=1;
}

//通过右按键短按可以实现选中的参数行自增加1调整
if(_button.state[RT_3D].press==SHORT_PRESS)
{
  _button.state[RT_3D].press=NO_PRESS;
   param_value[base_item+ver_item-1]+=1;
}

//通过左按键短按可以实现选中的参数行自减小50调整
if(_button.state[LT_3D].press==LONG_PRESS)
{
  _button.state[LT_3D].press=NO_PRESS;
   param_value[base_item+ver_item-1]-=50;
}

//通过右按键短按可以实现选中的参数行自增加50调整
if(_button.state[RT_3D].press==LONG_PRESS)
{
  _button.state[RT_3D].press=NO_PRESS;
   param_value[base_item+ver_item-1]+=50;
}
```

五向按键里面的**中按键长按**可以实现当前页面所有参数写入到 EEPROM 实现掉电存储。

```
//通过中间按键长按可以实现此页设置的所有参数保存
if(_button.state[ME_3D].press==LONG_PRESS)
{
  _button.state[ME_3D].press=NO_PRESS;
  //按下后对参数进行保存
  for(uint16_t i=0;i<21;i++)
  {
    WriteFlashParameter(RESERVED_PARAM+base_item+i,param_value[base_item+i]);
  }
}
```

2 航点自动飞行功能软件实现

用户录入的航点参数存储在飞控 EEPROM 内，飞控每次上电时会从 EEPROM 中读取航点参数并保存在 param_value 参数数组内，第一个航点的参数存储在该变量的第 51 维开始的三个变量内，航点生成函数的作用是判断 param_value 数组航点字节段区间的数据是否全为 0，来判断航点数据是否有效。当前能录入的最大航点数量为 28 个，针对不同的赛题任务，客户可以自己灵活调整。

```
2431 /*************************************************************************************************************************/
2432 #define nav_param_base (51-1)       //预留参数数组用于存放航点信息的起始量
2433 #define user_setpoint_max 28   //最大航点数，可根据实际项目需要自己定义，显示屏航点设置页面为4页，每页有7个航点，共28个
2434 #define user_setpoint_fixed_2d_cm  5.0f //5cm
2435 #define user_setpoint_fixed_3d_cm  10.0f//10cm
2436 #define user_setpoint_fixed_times  5//满足次数
2437 int32_t nav_setpoint[user_setpoint_max][3]={0,0,0};//航点坐标数组 0x200066D8
2438 uint8_t user_setpoint_valid_flag[user_setpoint_max]={0};//航点有效标志位
2439 void user_setpoint_generate(void)
2440 {
2441   memset(user_setpoint_valid_flag,0,sizeof(char)*user_setpoint_max);
2442   for(uint16_t i=0;i<user_setpoint_max;i++)
2443   {
2444     if((param_value[nav_param_base+3*i+0]==0
2445       &&param_value[nav_param_base+3*i+1]==0
2446       &&param_value[nav_param_base+3*i+2]==0)!=1)//通过判断参数组内数据均非0，来判定航点数据是否有效
2447     {
2448       user_setpoint_valid_flag[i]=true;
2449       nav_setpoint[i][0]=param_value[nav_param_base+3*i+0];
2450       nav_setpoint[i][1]=param_value[nav_param_base+3*i+1];
2451       nav_setpoint[i][2]=param_value[nav_param_base+3*i+2];
2452     }
2453     else user_setpoint_valid_flag[i]=false;
2454   }
2455 }
```

2.1 第一阶段——自动起飞到航巡高度方法 uint8_t Auto_Takeoff(float target)

```
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 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 18 用法，自主起飞任务完成后会进入 case 19 执行航点自动飞行功能。

```
190    case 18://自动起飞到某一高度
191    {
192      if(Auto_Takeoff(Target_Height)==1)
193      {
194        *mode+=1;//到达目标高度后，切换到下一SDK任务
195      }
196    }
197    break;
198    case 19://用户自定义航点飞行-无需二次编程，就可以实现3维空间内的若干航点遍历飞行
199    {
200      //用户通过地面站自定义或者按键手动输入三维的航点位置，无人机依次遍历各个航点，当前最多支持28个航点，可以自由扩展
201      Navigation_User_Setpoint();
202    }
203    break;
```

case 18执行自动起飞，达到期望高度后

继续执行case 19用户自定义航点飞行功能

```
129     case 11://自动起飞到某一高度
130     {
131       if(Auto_Takeoff(Target_Height)==1)//WORK_HEIGHT_CM
132       {
133         //*mode+=1;//到达目标高度后，切换到下一SDK任务
134         *mode+=4;//到达目标高度后，切换2022年7月省赛第一部分任务
135         //*mode+=5;//到达目标高度后，切换2022年7月省赛第二部分任务
136         //*mode+=6;//到达目标高度后，切换2022年7月省赛第三部分任务
137       }
138     }
139     break;
```

2.2 第二阶段——自定义航点飞行 Navigation_User_Setpoint(void)

第一步将高度期望设置成第一作业高度 150cm，水平位置期望为初始起飞时候的水平位置，在起飞点上方悬停时间设置为 1S，悬停时间可以根据实际需要调整。

```
2459  //用户通过按键自定义输入三维的航点位置，无人机依次遍历各个航点，最多支持28个航点
2460  void Navigation_User_Setpoint(void)
2461  {
2462    static uint8_t n=13;
2463    Vector3f target_position;
2464    float x=0,y=0,z=0;
2465    if(flight_subtask_cnt[n]==0)//起飞点作为第一个悬停点
2466    {
2467      basic_auto_flight_support();//基本飞行支持软件
2468      user_setpoint_generate();//生成航点
2469
2470      //记录下初始起点位置，实际项目中可设置为某一基准原点
2471      //base_position.x=VIO_SINS.Position[_EAST];
2472      //base_position.y=VIO_SINS.Position[_NORTH];
2473      base_position.z=First_Working_Height;//第一作业高度
2474
2475      x=base_position.x;
2476      y=base_position.y;
2477      z=First_Working_Height;
2478      target_position.x=x;
2479      target_position.y=y;
2480      target_position.z=z;
2481      Horizontal_Navigation(target_position.x,
2482                            target_position.y,
2483                            target_position.z,
2484                            GLOBAL_MODE,
2485                            MAP_FRAME);
2486      flight_subtask_cnt[n]=1;
2487      flight_global_cnt[n]=0;
2488      flight_global_cnt2[n]=0;
2489      execute_time_ms[n]=1000/flight_subtask_delta;//子任务执行时间
2490    }
```

起飞点上方悬停 1S 后，会根据按键录入航点数据进行判断，如果航点数据有效则将此航点设置成目标航点，并且线程计数器 flight_subtask_cnt[n]会赋值为 2，随后进入航点任务执行线程。

```
2491    else if(flight_subtask_cnt[n]==1)//起飞之后原定悬停1s后再执行航点任务
2492    {
2493      basic_auto_flight_support();//基本飞行支持软件
2494      if(execute_time_ms[n]>0) execute_time_ms[n]--;
2495      if(execute_time_ms[n]==0)//悬停时间计数器归零，悬停任务执行完毕
2496      {
2497        //判断所有航点均执行完毕，执行下一任务
2498        if(flight_global_cnt2[n]>=user_setpoint_max)
2499        {
2500          flight_subtask_cnt[n]=3;//结束航点遍历
2501          flight_global_cnt[n]=0;
2502          execute_time_ms[n]=0;
2503          //航点计数器清0
2504          flight_global_cnt2[n]=0;
2505          return ;
2506        }
2507
2508        uint16_t current_num=constrain_int32(flight_global_cnt2[n],0,user_setpoint_max-1);//限幅防溢出
2509        if(user_setpoint_valid_flag[current_num]==true)//如果当前的航点有效，就设置目标航点进入下一线程
2510        {
2511          x=nav_setpoint[current_num][0];
2512          y=nav_setpoint[current_num][1];
2513          z=nav_setpoint[current_num][2];
2514          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
2515          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
2516          target_position.z=z;
2517          Horizontal_Navigation(target_position.x,
2518                                target_position.y,
2519                                target_position.z,
2520                                GLOBAL_MODE,
2521                                MAP_FRAME);
2522          flight_subtask_cnt[n]=2;
2523          flight_global_cnt[n]=0;
2524          execute_time_ms[n]=0;
2525        }
2526        else//如果当前的航点无效，跳过当前航点，继续设置下一航点
2527        {
2528          //航点计数器自加
2529          flight_global_cnt2[n]++;
2530        }
2531      }
2532    }
```

若航点有效，则设置此坐标为目标位置

若航点数据无效，计数器自加，继续判断下一航点

在线程 2 中执行航点飞行任务，并实时判断无人机的三维位置偏差，连续 N 次满足偏差小于某一阈值时，可以认为当前航点已抵达，继续刷新下一航点信息。当所有航点遍历完毕后，跳到第 3 线程中，结束整个航点遍历过程。

```
2533    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
2534    {
2535      basic_auto_flight_support();//基本飞行支持软件
2536      //判断是否到达目标航点位置
2537      if(flight_global_cnt[n]<user_setpoint_fixed_times)//持续10*5ms=0.05s满足
2538      {
2539        float dis_cm=pythagorous3(OpticalFlow_Pos_Ctrl_Err.x,OpticalFlow_Pos_Ctrl_Err.y,Total_Controller.Height_Position_C
2540        if(dis_cm<=user_setpoint_fixed_3d_cm) flight_global_cnt[n]++;
2541        else flight_global_cnt[n]/=2;
2542      }
2543      else//持续10*5ms满足，表示到达目标航点位置，后降低目标高度
2544      {
2545        flight_subtask_cnt[n]=1;
2546        flight_global_cnt[n]=0;
2547        execute_time_ms[n]=0;
2548        //航点计数器自加
2549        flight_global_cnt2[n]++;
2550      }
2551      //判断所有航点均执行完毕，执行下一任务
2552      if(flight_global_cnt2[n]>=user_setpoint_max)
2553      {
2554        flight_subtask_cnt[n]=3;
2555        flight_global_cnt[n]=0;
2556        execute_time_ms[n]=0;
2557        //航点计数器清0
2558        flight_global_cnt2[n]=0;
2559      }
```

通过判断三维位置偏差值，来确定航点是否已经抵达

达到目标航点位置后，线程计数器置1，继续刷新下一航点

航点计数器自加

所有航点执行完毕后，线程计数器赋值为3，跳出航点执行过程

最后执行线程 3 中的原地下降任务，无人机降落到地面后会触发地面检测条件无人机会自动上锁，结束整个飞行过程。

```
2561    else if(flight_subtask_cnt[n]==3)//执行航点完毕后，原地下降，可以根据实际需要，自写更多的飞行任务
2562    {
2563      Flight.yaw_ctrl_mode=ROTATE;
2564      Flight.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];
2565      OpticalFlow_Control_Pure(0);
2566      Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-30);//高度控制
2567    }
2568    else
2569    {
2570      basic_auto_flight_support();//基本飞行支持软件
2571    }
```

原地降落——水平位置保持，竖直期望速度为-30cm/s