竞赛无人机基础飞行控制函数

无人机的在空中的基本飞行包括姿态控制（角度、角速度）与高度控制（位置、速度、加速度）两个部分。

① 姿态控制是无人机按照飞行控制指令姿态保持水平或者以某一倾斜角度飞行，同时结合偏航方向旋转控制可以实现自转、刷锅等飞行动作。需要注意的是姿态自稳只是对无人机的姿态角度进行了控制，并不能使无人机水平位置不发生偏移，水平位置依赖于定点控制（如 SLAM 定位、GPS 定位、光流定位等）。

② 高度控制是控制无人机在某一高度位置上悬停或者以一定速度控制无人机上升和下降。

- 姿态控制函数

  void Angle_Control_Target(Controller_Output *_flight_output)

  输入参数 Controller_Output 结构体定义如下

```
typedef struct
{
    float throttle_control_output;        //油门控制器最终输出，变量暂未使用
    float roll_control_output;            //横滚姿态控制器最终输出，变量暂未使用
    float pitch_control_output;           //俯仰姿态控制器最终输出，变量暂未使用
    float yaw_control_output;             //偏航姿态控制器最终输出，变量暂未使用
    float roll_outer_control_output;      //横滚姿态控制器输入
    float pitch_outer_control_output;     //俯仰姿态控制器输入
    float yaw_outer_control_output;       //偏航姿态控制器输入
    uint16_t motor_output[MOTOR_NUM];     //电机映射输出值，变量暂未使用
    uint16_t temperature_control_output;//温度控制器输出值，变量暂未使用
    uint16_t yaw_ctrl_cnt;                //偏航控制计数器
    uint8_t yaw_ctrl_mode;                //偏航控制模式
    uint8_t yaw_ctrl_start;               //偏航控制开始标志位
    uint8_t yaw_ctrl_end;                 //偏航控制结束标志位
    uint32_t start_time_ms;               //偏航控制开始时间
    uint32_t execution_time_ms;           //偏航控制执行时间
    uint8_t init;                         //偏航控制初始化标志位
}Controller_Output;
```

其中用户只需要关注圈红处的变量，在姿态自稳控制中，以下三组期望输入来源于遥控器输入信号解析后的直接给定，偏航角控制模式为 ROTATE 表示偏航手动控制模式，此手动模式也是飞控初始化后默认偏航控制模式。

```
Flight.roll_outer_control_output =RC_Data.rc_rpyt[RC_ROLL];
Flight.pitch_outer_control_output=RC_Data.rc_rpyt[RC_PITCH];
Flight.yaw_outer_control_output   =RC_Data.rc_rpyt[RC_YAW];
Flight.yaw_ctrl_mode=ROTATE;
```

```
90  enum YAW_CTRL_MODE
91  {
92      ROTATE=0,                   //手动偏航控制模式
93      AZIMUTH=1,                  //绝对偏航角度控制模式
94      CLOCKWISE=2,                //相对偏航角度顺时针控制模式
95      ANTI_CLOCKWISE=3,           //相对偏航角度逆时针控制模式
96      CLOCKWISE_TURN=4,           //角速度控制顺时针模式
97      ANTI_CLOCKWISE_TURN=5,//角速度控制逆时针模式
98  };
99
```

除了 ROTATE 偏航控制模式，在用户二次开发无人机自主飞行控制中，主要有使用的还有如下几种:

① AZIMUTH 绝对偏航角度控制模式

其中偏航模式参数 AZIMUTH 表示直接对偏航角度进行控制，此时 yaw_outer_control_output 给定的是期望偏航角度。

② CLOCKWISE 相对偏航角度顺时针控制模式

其中偏航模式参数 CLOCKWISE 表示直接对偏航角度进行控制，此时 yaw_outer_control_output 给定的是以当前偏航角度为基准，需要顺时针旋转的偏航角度。

③ ANTI_CLOCKWISE 相对偏航角度逆时针控制模式

其中偏航模式参数 ANTI_CLOCKWISE 表示直接对偏航角度进行控制，此时 yaw_outer_control_output 给定的是以当前偏航角度为基准，需要逆时针旋转的偏航角度。

②③两种相对运动方式，在给定期望时同时需要设置 yaw_ctrl_start 参数，姿态控制函数在执行过程中会实时对角度偏差进行判断，看是否到达对应偏航角度，完成后会将 yaw_ctrl_end 标志位置 1，用户可以判断 yaw_ctrl_end 的数据来确定偏航控制是否完成，进而处理下一阶段子任务。初次接触的用户可以结合 Subtask_Demo.c 提供的 flight_subtask_1、flight_subtask_2 来练习掌握用法。

```
/*************************************************************************************************/
//顺时针转动90度，完成后降落
void flight_subtask_1(void)
{
  static uint8_t n=0;
  if(flight_subtask_cnt[n]==0)
  {
    Flight.yaw_ctrl_mode=CLOCKWISE;
    Flight.yaw_ctrl_start=1;
    Flight.yaw_outer_control_output  =90;//顺时针90度

    OpticalFlow_Control_Pure(0);
    Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制
    flight_subtask_cnt[n]=1;
  }
  else if(flight_subtask_cnt[n]==1)
  {
    Flight.yaw_ctrl_mode=CLOCKWISE;
    Flight.yaw_outer_control_output  =0;

    OpticalFlow_Control_Pure(0);
    Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制

    if(Flight.yaw_ctrl_end==1)  flight_subtask_cnt[n]=2;//执行完毕后，切换到下一阶段
  }
  else if(flight_subtask_cnt[n]==2)
  {
    Flight.yaw_ctrl_mode=ROTATE;
    Flight.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];

    OpticalFlow_Control_Pure(0);
    Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-30);//高度控制
  }
  else
  {
    basic_flight_support();//基本飞行支持软件
  }
}
```

```
case CLOCKWISE://顺时针——相对给定时刻的航向角度
{
  if(_flight_output->yaw_ctrl_start==1)//更新偏航角度期望
  {
    float yaw_tmp=WP_AHRS.Yaw- _flight_output->yaw_outer_control_output;
    if(yaw_tmp<0)    yaw_tmp+=360;
    if(yaw_tmp>360) yaw_tmp-=360;
    Total_Controller.Yaw_Angle_Control.Expect=yaw_tmp;
    _flight_output->yaw_ctrl_start=0;
    _flight_output->yaw_ctrl_cnt=0;
    _flight_output->yaw_ctrl_end=0;
  }

  if(_flight_output->yaw_ctrl_end==0)//判断偏航角是否控制完毕
  {
    if(ABS(Total_Controller.Yaw_Angle_Control.Err)<3.0f) _flight_output->yaw_ctrl_cnt++;
    else _flight_output->yaw_ctrl_cnt/=2;

    if(_flight_output->yaw_ctrl_cnt>=200) _flight_output->yaw_ctrl_end=1;
  }

  Total_Controller.Yaw_Angle_Control.FeedBack=WP_AHRS.Yaw;//偏航角反馈
  PID_Control_Yaw(&Total_Controller.Yaw_Angle_Control);//偏航角度控制
  //对最大偏航角速度进行限制
  float tmp=constrain_float(Total_Controller.Yaw_Angle_Control.Control_OutPut,-YAW_GYRO_CTRL_MAX,YAW_GYRO_CTRL_MAX);
  Total_Controller.Yaw_Gyro_Control.Expect=tmp;//偏航角速度环期望，来源于偏航角度控制器输出
}
break;
```

④ CLOCKWISE_TURN 角速度控制顺时针模式

其中偏航模式参数 CLOCKWISE_TURN 表示直接对偏航角速度进行控制，此时 yaw_ctrl_mode 给定的是顺时针旋转的期望偏航运动的角速度。

⑤ ANTI_CLOCKWISE_TURN 角速度控制逆时针模式

其中偏航模式参数 CLOCKWISE_TURN 表示直接对偏航角速度进行控制，此时 yaw_ctrl_mode 给定的是顺时针旋转的期望偏航运动的角速度。

④⑤两种相对运动方式，在给定期望时同时需要设置 yaw_ctrl_start 参数，以及需要执行的时间参数 execution_time_ms，姿态控制函数在执行过程中会实时对已执行的时间进行判断是否完成，完成后会将 yaw_ctrl_end 标志位置 1，用户可以判断 yaw_ctrl_end 的数据来确定偏航控制是否完成，进而处理下一阶段子任务。初次接触的用户可以结合 Subtask_Demo.c 提供的 flight_subtask_3、flight_subtask_4 来练习掌握用法。

```
/*************************************************************************************
//以10deg/s的角速度顺时针转动10000ms，完成后降落
void flight_subtask_3(void)
{
  static uint8_t n=2;
  if(flight_subtask_cnt[n]==0)
  {
    Flight.yaw_ctrl_mode=CLOCKWISE_TURN;
    Flight.yaw_ctrl_start=1;
    Flight.yaw_outer_control_output  =10;//以10deg/s的角速度顺时针转动10000ms
    Flight.execution_time_ms=10000;//执行时间

    OpticalFlow_Control_Pure(0);
    Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制
    flight_subtask_cnt[n]=1;
  }
  else if(flight_subtask_cnt[n]==1)
  {
    Flight.yaw_ctrl_mode=CLOCKWISE_TURN;
    Flight.yaw_outer_control_output  =0;

    OpticalFlow_Control_Pure(0);
    Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制

    if(Flight.yaw_ctrl_end==1)  flight_subtask_cnt[n]=2;//执行完毕后，切换到下一阶段
  }
  else if(flight_subtask_cnt[n]==2)
  {
    Flight.yaw_ctrl_mode=ROTATE;
    Flight.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];

    OpticalFlow_Control_Pure(0);
    Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-30);//高度控制
  }
  else
  {
    basic_flight_support();//基本飞行支持软件
  }
}
```

```
case CLOCKWISE_TURN://以某一角速度顺时针旋转多长时间
{
  uint32_t curr_time_ms=millis();

  if(_flight_output->yaw_ctrl_start==1)//更新偏航角度期望
  {
    //对最大偏航角速度进行限制
    float tmp=constrain_float(-_flight_output->yaw_outer_control_output,-YAW_GYRO_CTRL_MAX,YAW_GYRO_CTRL_MAX);
    Total_Controller.Yaw_Gyro_Control.Expect=tmp;//偏航角速度环期望，来源于偏航角度控制器输出
    _flight_output->yaw_ctrl_start=0;
    _flight_output->yaw_ctrl_cnt=0;
    _flight_output->yaw_ctrl_end=0;
    _flight_output->start_time_ms=curr_time_ms;//记录开始转动的时间
  }

  if(_flight_output->yaw_ctrl_end==0)//判断偏航角是否控制完毕
  {
    uint32_t tmp=curr_time_ms-_flight_output->start_time_ms;
    if(tmp>=_flight_output->execution_time_ms)
      _flight_output->yaw_ctrl_end=1;
  }
  else
  {
    //执行完毕后，
    //1、将偏航角速度期望给0，
    //2、停止旋转并锁定当前偏航角，需要退出CLOCKWISE_TURN模式，角度期望才会有效，因为此模式没有对偏航角度进行控制
    Total_Controller.Yaw_Gyro_Control.Expect=0;
    Total_Controller.Yaw_Angle_Control.Expect=WP_AHRS.Yaw;
  }
}
break;
```

- 高度控制函数

  void Flight_Alt_Hold_Control(uint8_t mode,float target_alt,float target_vel)

  输入参数 uint8_t mode 可选参数如下：

```
typedef enum
{
  ALTHOLD_MANUAL_CTRL=0,      //高度手动控制
  ALTHOLD_AUTO_POS_CTRL,      //高度直接位置控制
  ALTHOLD_AUTO_VEL_CTRL,      //高度直接速度控制
}ALTHOLD_CTRL_MODE;
```

ALTHOLD_MANUAL_CTRL、ALTHOLD_AUTO_POS_CTRL、ALTHOLD_AUTO_VEL_CTRL 分别表示高度手动控制、高度直接位置控制、高度直接速度控制，在 ALTHOLD_MANUAL_CTRL 模式中，高度可以通过遥控器油门杆来控制，油门杆偏离中位死区向上、向下分别表示给定无人机期望向上爬升速度和下降速度，油门回中后飞机会锁定当前高度作为悬停保持高度；ALTHOLD_AUTO_POS_CTRL 模式中，期望高度直接来源于 target_alt，无人机将高度位置控制到 target_alt 所在高度进行悬停；ALTHOLD_AUTO_VEL_CTRL 模式中，无人机不在对高度位置进行控制，高度方向上期望速度直接来源于 target_vel 给定，arget_vel>0 表示期望速度向上，反之表示期望速度向下，相关用法例子如下：

```
    case 15://前面预留case不满足情况下执行此情形
    {
      Flight.roll_outer_control_output =RC_Data.rc_rpyt[RC_ROLL];
      Flight.pitch_outer_control_output=RC_Data.rc_rpyt[RC_PITCH];
      Flight.yaw_ctrl_mode=ROTATE;
      Flight.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];
      Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制
    }
    break;
    case 16://SDK模式中原地降落至地面怠速后停桨,用于任务执行完成后降落
    {
      OpticalFlow_Control(0);
      Flight.yaw_ctrl_mode=ROTATE;
      Flight.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];
      Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-50);//高度控制
    }
    break;
    default:
    {
      Flight.roll_outer_control_output =RC_Data.rc_rpyt[RC_ROLL];
      Flight.pitch_outer_control_output=RC_Data.rc_rpyt[RC_PITCH];
      Flight.yaw_ctrl_mode=ROTATE;
      Flight.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];
      Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制
    }
  }
}
```