基于MSP432的四旋翼飞行器软件设计

DY-FC-432&DY-R-Drone V1.0.0

无人机软件设计

软件任务：

- 悬停
- 移动

遥控器输入指令

- 一键起飞
- 一键降落
- 油门控制
- 偏航控制
- 俯仰控制
- 横滚控制

两大核心任务：悬停和飞行移动，涉及四个目标控制状态量，高度、俯仰角、横滚角、偏航角。

悬停：停在一定高度（以室内飞行为主，无GPS）、俯仰角和横滚角为零，偏航角可能存在一定角度。

飞行移动：通过遥控器摇杆控制无人机沿着不同方向移动。

油门控制：改变无人机（质点）电机转速；
偏航控制、俯仰控制、横滚控制：改变无人机（质点）运动方向。

无人机软件设计

一键起飞：

启动一键起飞功能，延时14s，无人机起飞悬停至约1.2m高度，悬停，开始降落。

```
/*一键起飞*/
void one_key_take_off()
{
        if(flag.unlock_en)
        {
                one_key_taof_start = 1;
                flag.fly_ready = 1;
        }
}
/*一键降落*/
void one_key_land()
{
        flag.auto_take_off_land = AUTO_LAND;
}
```

flag.unlock_en：用作解锁使能

one_key_taof_start：用作14秒计时

flag.fly_ready：用作起飞标识

flag.auto_take_off_land：用作降落标识

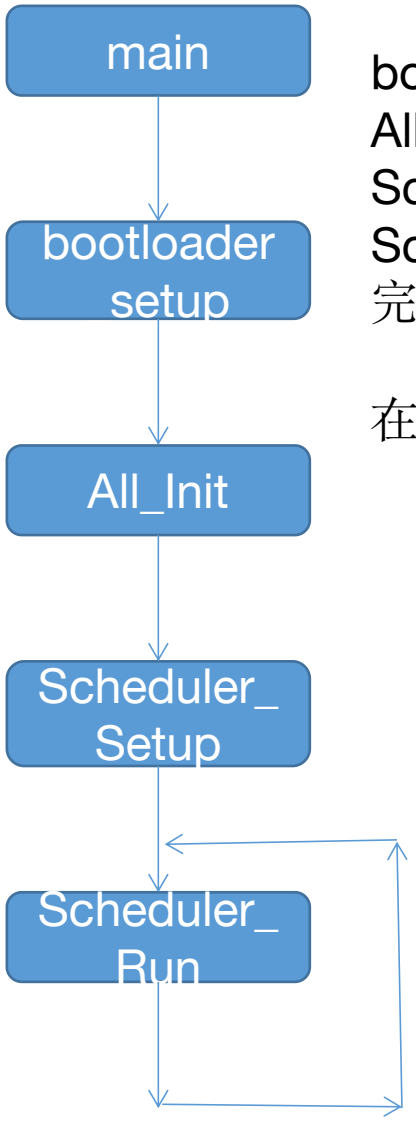
```
/*一键起飞任务（主要功能为延迟14s）*/
void one_key_take_off_task(u16 dt_ms)
```

。 。 。

```
if(one_key_taof_start > 1400 && flag.motor_preparation == 1)    //1400*10=14000ms=14s
{
  one_key_taof_start = 0;
    if(flag.auto_take_off_land == AUTO_TAKE_OFF_NULL)
    {
      flag.auto_take_off_land = AUTO_TAKE_OFF;
      //解锁、起飞

      flag.taking_off = 1;
    }

}
```

。 。 。

无人机软件设计



从主函数开始

bootloader setup:设置程序启动入口
All_Init:所有设备初始化（包括连接传感器设备的MCU外设以及软件程序设计用的外设）
Scheduler_Setup:在没有RTOS下，设置了一个任务调度器
Scheduler_Run:运行任务调度器，所有系统功能，除了中断服务函数外，都在任务调度器内完成

在All_Init中，函数返回值是一个数据结构。

Q：这里为什么要设置bootloader?
A：通过串口给程序升级

Q:为什么使用任务调度器，是RTOS吗?
A：飞控是具有复杂的任务控制，为了便于理解，使用软件分时模拟了一个任务调度器，没有使用RTOS

无人机软件设计

从任务调度器开始

时间片转换计算：

时间片有：1000Hz=1ms、500Hz=2ms、200Hz=5ms、100Hz=10ms、50Hz=20ms、20Hz=50ms、2Hz=500ms

500ms是10个50ms， 500ms是25个20ms， 500ms是50个10ms， 500ms是100个5ms， 500ms是250个2ms,500ms是500个1ms,

任务时间片频率越高任务执行的频率越高，即单位时间任务执行的次数越多。

1ms任务：传感器数据读取，姿态解析，飞行状态任务，开关状态任务，姿态角速度环控制，电机输出控制，数传数据交换

2ms任务：飞行控制DY_Flight_Control（使用时，外接OPENMV3模块）

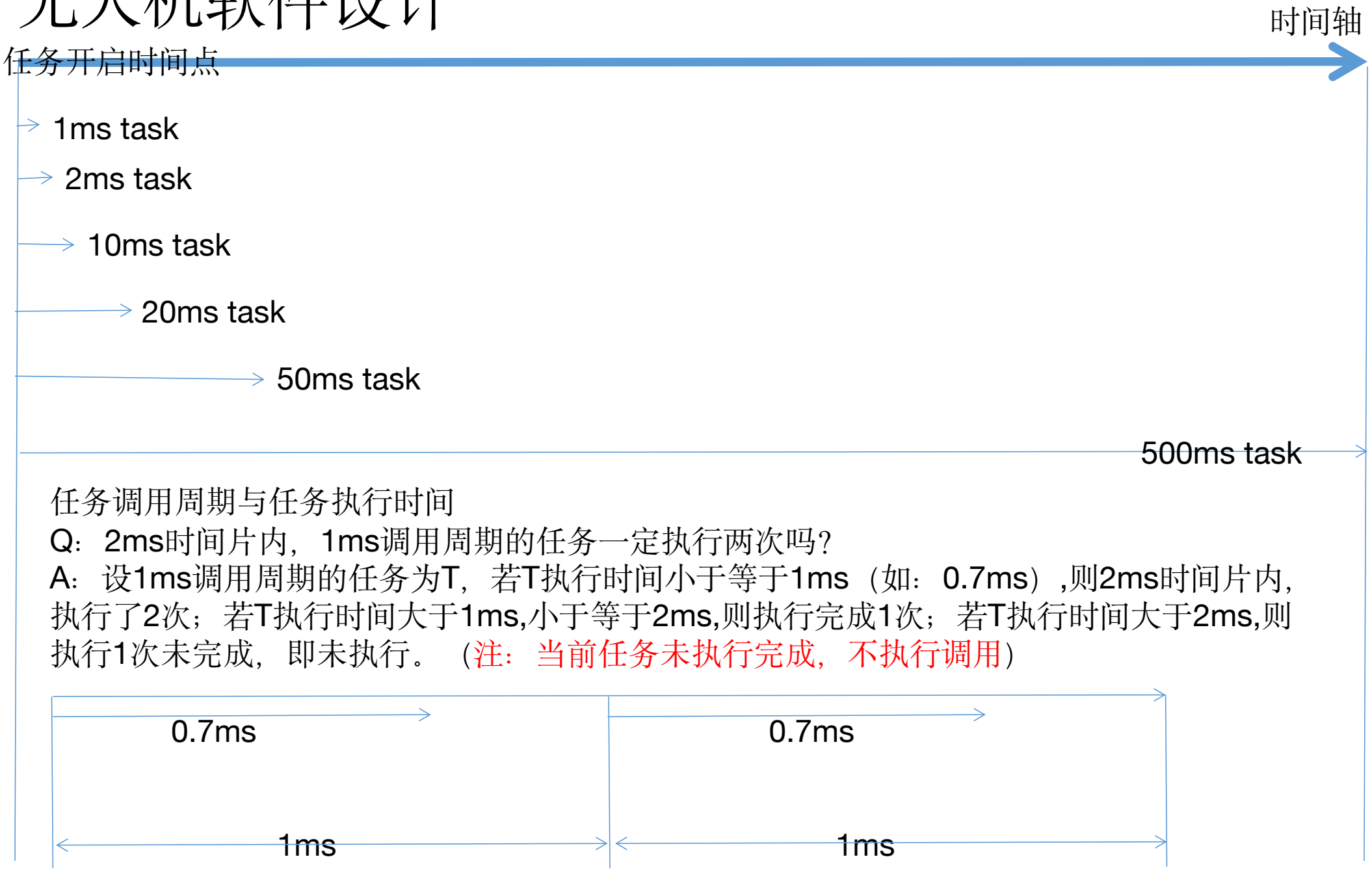
10ms任务：遥控器处理，飞行模式设置，PID控制，灯光控制

20ms任务：电子罗盘数据处理任务

50ms任务：TOF激光任务，电池电压相关任务

500ms任务：延时存储相关任务

无人机软件设计



任务调用周期与任务执行时间

Q：2ms时间片内，1ms调用周期的任务一定执行两次吗？

A：设1ms调用周期的任务为T，若T执行时间小于等于1ms（如：0.7ms），则2ms时间片内，执行了2次；若T执行时间大于1ms,小于等于2ms,则执行完成1次；若T执行时间大于2ms,则执行1次未完成，即未执行。（注：当前任务未执行完成，不执行调用）

无人机软件设计

```c
void Scheduler_Run(void)
{
    uint8_t index = 0;
    //循环判断所有线程，是否应该执行
    for(index=0;index < TASK_NUM;index++)
    {
        //获取系统当前时间，单位ms
        uint32_t tnow = SysTick_GetTick();
        //进行判断，如果当前时间减去上一次执行的时间，大于等于该线程的执行周期，则执行线程
        if(tnow - sched_tasks[index].last_run >= sched_tasks[index].interval_ticks)
        {
            //更新线程的执行时间，用于下一次判断
            sched_tasks[index].last_run = tnow;
            //执行线程函数，使用的是函数指针
            sched_tasks[index].task_func();
        }
    }
}
```

综上，依次执行任务：

```c
static sched_task_t sched_tasks[] =
{
    {Loop_1000Hz,1000,  0, 0},
    {Loop_500Hz , 500,  0, 0},
    {Loop_200Hz , 200,  0, 0},
    {Loop_100Hz , 100,  0, 0},
    {Loop_50Hz  ,  50,  0, 0},
    {Loop_20Hz  ,  20,  0, 0},
    {Loop_2Hz   ,   2,  0, 0},
};
```

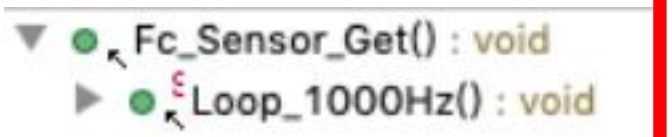
即对于要执行的任务，确认前一个一定已经执行完成，在执行新的。

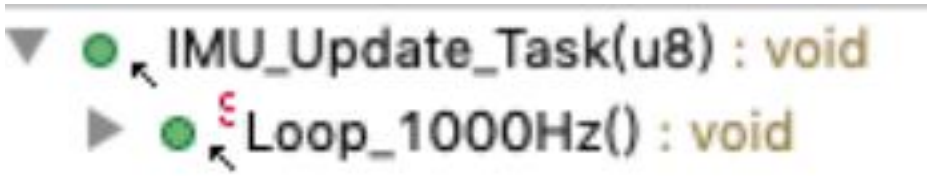
无人机软件设计

1ms任务：

```
/*传感器数据读取*/
Fc_Sensor_Get();          //读取ICM20602（1ms）、AK8975+SPL0601（20ms）原始数据
/*惯性传感器数据准备*/
Sensor_Data_Prepare(1);    //ICM20602数据处理（滤波、转换）==>
                           //陀螺仪（度每秒、弧度每秒）+加速度计（厘米每平方秒）
/*姿态解算更新*/
IMU_Update_Task(1);        //使用四元数法进行姿态更新
/*获取WC_Z加速度*/
WCZ_Acc_Get_Task();        //地理坐标系下Z轴的运动加速度==>wcz_acc_use
/*飞行状态任务*/
Flight_State_Task(1,CH_N);
/*开关状态任务*/
Swtich_State_Task(1);      //判断PMW3901光流模块、第三方光流、TOF数据是否有效
/*姿态角速度环控制*/
Att_1level_Ctrl(1e-3f);
/*电机输出控制*/
Motor_Ctrl_Task(1);
/*数传数据交换*/
DY_DT_Data_Exchange();
```

无人机软件设计





1ms执行一次IMU_update

Loop_1000Hz

Fc_Sensor_Get

---->1ms读取一次加速度计和陀螺仪

---->20ms读取一次电子罗盘和气压计

数据计算

flag.start_ok==1---->1ms执行一次传感器数据获取。

Fc_Sensor_Get.IMU--->数据处理--->IMU_update

无人机软件设计

Fc_Sensor_Get

mpu_buffer[14]
icm20602_readbuf(MPUREG_ACCEL_XOUT_H,14,mpu_buffer);

1ms

for(i = 0; i < 6; i++)
ak8975_buf[i] = Drv_SPI2_RW(0xff);

20ms

Sensor_Data_Prepare

```
/*读取buffer原始数据*/
sensor.Acc_Original[X] = (s16)((((u16)mpu_buffer[0]) << 8) | mpu_buffer[1]);
sensor.Acc_Original[Y] = (s16)((((u16)mpu_buffer[2]) << 8) | mpu_buffer[3]);
sensor.Acc_Original[Z] = (s16)((((u16)mpu_buffer[4]) << 8) | mpu_buffer[5]);

sensor.Gyro_Original[X] = (s16)((((u16)mpu_buffer[ 8]) << 8) | mpu_buffer[ 9]) ;
sensor.Gyro_Original[Y] = (s16)((((u16)mpu_buffer[10]) << 8) | mpu_buffer[11]) ;
sensor.Gyro_Original[Z] = (s16)((((u16)mpu_buffer[12]) << 8) | mpu_buffer[13]) ;
```

```
/*赋值*/
for(i = 0;i<6;i++)
{
    sensor_val_rot[i] = sensor_val[i];
}

/*数据坐标转90度*/
sensor_val_ref[G_X] =  sensor_val_rot[G_Y] ;
sensor_val_ref[G_Y] = -sensor_val_rot[G_X] ;
sensor_val_ref[G_Z] =  sensor_val_rot[G_Z];


sensor_val_ref[A_X] =  (sensor_val_rot[A_Y] - save.acc_offset[Y] ) ;
sensor_val_ref[A_Y] = -(sensor_val_rot[A_X] - save.acc_offset[X] ) ;
sensor_val_ref[A_Z] =  (sensor_val_rot[A_Z] - save.acc_offset[Z] ) ;
```

```
/*得出校准后的数据*/
for(i = 0; i < 3; i++)
{

    sensor_val[A_X+i] = sensor.Acc_Original[i] ;

    sensor_val[G_X+i] =(s32)(sensor.Gyro_Original[i] - save.gyro_offset[i]) ;
}
```

无人机软件设计

1ms任务----数据处理得到：

sensor.Gyro_deg[0]、sensor.Gyro_rad[0]、sensor.Acc_cmss[0]
sensor.Gyro_deg[1]、sensor.Gyro_rad[1]、sensor.Acc_cmss[1]
sensor.Gyro_deg[2]、sensor.Gyro_rad[2]、sensor.Acc_cmss[2]

```
/*转换单位*/
for(i =0 ;i<3;i++)
{
    /*陀螺仪转换到度每秒，量程+-2000度*/
    sensor.Gyro_deg[i] = sensor.Gyro[i] *0.06103f ;

    /*陀螺仪转换到弧度度每秒，量程+-2000度*/
    sensor.Gyro_rad[i] = sensor.Gyro[i] *RANGE_PN2000_TO_RAD ;

    /*加速度计转换到厘米每平方秒，量程+-8G*/
    sensor.Acc_cmss[i] = (sensor.Acc[i] *RANGE_PN8G_TO_CMSS);
}
```

输入：
mpu_buffer

数据处理

输出：
sensor.Gyro_deg
sensor.Gyro_rad
sensor.Acc_cmss

IMU_Update_Task

```
imu_data =  {1,0,0,0,

            {0,0,0},
            {0,0,0},
            {0,0,0},
            {0,0,0},
            {0,0,0},
            {0,0,0},
             0,0,0};
```

```
/*姿态计算，更新，融合*/
IMU_update(dT_ms
*1e-3f, &imu_state,
sensor.Gyro_rad,
sensor.Acc_cmss,
mag.val,    ?
&imu_data);
```

输入：
sensor.Gyro_rad
sensor.Acc_cmss

数据处理

输出：
imu_data

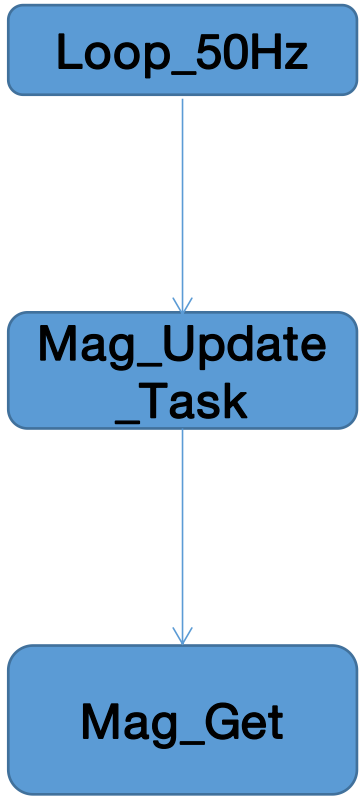
无人机软件设计

mag_val ?

```
void Mag_Get(s16 mag_val[3])
{
    s16 t[3];

    t[0] = ((((int16_t)ak8975_buf[1]) << 8) | ak8975_buf[0]) ;
    t[1] = ((((int16_t)ak8975_buf[3]) << 8) | ak8975_buf[2]) ;
    t[2] = ((((int16_t)ak8975_buf[5]) << 8) | ak8975_buf[4]) ;

    /*转换坐标轴为ANO坐标*/
    mag_val[0] = +t[0];
    mag_val[1] = -t[1];
    mag_val[2] = -t[2];
}
```

```
static s16 mag_val[3];
void Mag_Update_Task(u8 dT_ms)
{
    Mag_Get(mag_val);

    Mag_Data_Deal_Task(dT_ms,mag_val,imu_data.z_vec[Z],sensor.Gyro_deg[X],sensor.Gyro_deg[Z]);
}
```

```
static void Loop_50Hz(void) //20ms执行一次
{
    /*罗盘数据处理任务*/
    Mag_Update_Task(20);
}
```



20ms执行一次磁力计数据计算任务。

无人机软件设计

Flight_State_Task:

```
/*设置油门摇杆量*/
thr_deadzone = (flag.wifi_ch_en != 0) ? 0 : 50;
fs.speed_set_h_norm[Z] = my_deadzone(CH_N[CH_THR],0,thr_deadzone) *0.0023f;     //-1.035~1.035  油门归一值
fs.speed_set_h_norm_lpf[Z] += 0.2f *(fs.speed_set_h_norm[Z] - fs.speed_set_h_norm_lpf[Z]);
```

0:油门摇杆中间位置

```
/*推油门起飞*/
if(flag.fly_ready)
{
  if(fs.speed_set_h_norm[Z]>0.01f && flag.motor_preparation == 1)
  {
    flag.taking_off = 1;
  }
}
```

1

```
if(switchs.of_flow_on || switchs.dy_pmw3901_on)    //使用光流模块
{
  max_speed_lim = 1.5f *wcz_hei_fus.out;
  max_speed_lim = LIMIT(max_speed_lim,50,150);
}
```

```
/***************OpenMv控制模式***************/
  if(DY_Debug_Mode == 1)
  {
    fs.speed_set_h[X] = dy_pit;
    fs.speed_set_h[Y] = dy_rol;
  }
  else
  {
    fs.speed_set_h[X] = fs.speed_set_h_cms[X];
    fs.speed_set_h[Y] = fs.speed_set_h_cms[Y];
  }
```

3

```
if(flag.taking_off)
{
  if(flying_cnt<1000)     //1s
  {
    flying_cnt += dT_ms;
  }
  else
  {
    /*起飞后1秒，认为已经在飞行*/
    flag.flying = 1;
  }

  if(fs.speed_set_h_norm[Z]>0)
  {
    /*设置上升速度*/
    fs.speed_set_h[Z] = fs.speed_set_h_norm_lpf[Z] *MAX_Z_SPEED_UP;
  }
  else
  {
    /*设置下降速度*/
    fs.speed_set_h[Z]  = fs.speed_set_h_norm_lpf[Z] *MAX_Z_SPEED_DW;
  }
}
else
{
  fs.speed_set_h[Z] = 0 ;
}
```

2

油门摇杆向上

油门摇杆向下

DY 德研電科

无人机软件设计

land_discriminat

```
/*调用检测着陆的函数*/
land_discriminat(dT_ms);

/*倾斜过大上锁*/
if(rolling_flag.rolling_step == ROLL_END)
{
  if(imu_data.z_vec[Z]<0.25f)//75度  //////
  {

    flag.fly_ready = 0;
  }
}
```

```
/*油门归一值小于0.1并且垂直方向加速度小于阈值  或者启动自动降落*/
if( fs.speed_set_h_norm[Z] < 0.1f && imu_data.w_acc[Z]<200) || flag.auto_take_off_land == AUTO_LAND
{
  if(ld_delay_cnt>0)    //200ms
  {
    ld_delay_cnt -= dT_ms;
  }
}
else
{
  ld_delay_cnt = 200;
}
```

摇杆低于中间位置且加速度方向向下即升力小于重力

使用一键降落

```
/*意义是：如果向上推了油门，就需要等垂直方向加速度小于200cm/s2 保持200ms才开始检测*/
if(ld_delay_cnt <= 0 && (flag.thr_low || flag.auto_take_off_land == AUTO_LAND) )
{
  /*油门最终输出量小于250并且没有在手动解锁上锁过程中，持续1.5秒，认为着陆，然后上锁*/
  if(mc.ct_val_thr<250 && flag.fly_ready == 1 && flag.locking != 2)//ABS(wz_spe_f1.out <20 ) //还应当 与上速度条件，速度小于正20厘米每秒。
  {
    if(landing_cnt<1500)
    {
      landing_cnt += dT_ms;
    }
    else
    {

      flying_cnt = 0;
      flag.taking_off = 0;

      landing_cnt =0;
      flag.fly_ready =0;

      flag.flying = 0;
    }
  }
  else
  {
    landing_cnt = 0;
  }
}
else
{
  landing_cnt  = 0;
```

如果向上推了油门，就需要等垂直方向加速度小于200cm/s2 保持200ms才开始检测

无人机软件设计

```
/*校准中，复位重力方向*/
if(sensor.gyr_CALIBRATE != 0 || sensor.acc_CALIBRATE != 0 ||sensor.acc_z_auto_CALIBRATE)
{
  imu_state.G_reset = 1;
}

/*复位重力方向时，认为传感器失效*/
if(imu_state.G_reset == 1)
{
  flag.sensor_ok = 0;
  WCZ_Data_Reset();
}
else if(imu_state.G_reset == 0)
{
  if(flag.sensor_ok == 0)
  {
    flag.sensor_ok = 1;
    DY_DT_SendString("IMU OK!",sizeof("IMU OK!"));
  }
}
```

传感器校准

```
if(flag.fly_ready == 0)
{
  flag.flying = 0;
  landing_cnt = 0;
  flag.taking_off = 0;
  flying_cnt = 0;

  //复位融合
  if(flag.taking_off == 0)
  {

  }
}
```

未使用一键起飞

无人机软件设计

输入：
油门摇杆量
CH_N[CH_THR]

数据处理

输出：
fs.speed_set_h_norm[Z]
fs.speed_set_h_norm_lpf[Z]

输入：
fs.speed_set_h_norm_lpf[Z]

数据处理

输出：
fs.speed_set_h[Z]

输入：
摇杆量CH_N[CH_PIT]、
CH_N[CH_ROL]

数据处理

输出：
speed_set_h_cms

DY
德研電科

无人机软件设计

无openmv3

输入：

speed_set_h_cms

数据处理

输出：

fs.speed_set_h[X]
fs.speed_set_h[Y]

```
/***************OpenMv控制模式***************/
if(DY_Debug_Mode == 1)
{
  fs.speed_set_h[X] = dy_pit;
  fs.speed_set_h[Y] = dy_rol;
}
else
{
  fs.speed_set_h[X] = fs.speed_set_h_cms[X];
  fs.speed_set_h[Y] = fs.speed_set_h_cms[Y];
}
```

综上所得

输入

油门摇杆
CH_N[CH_THR]

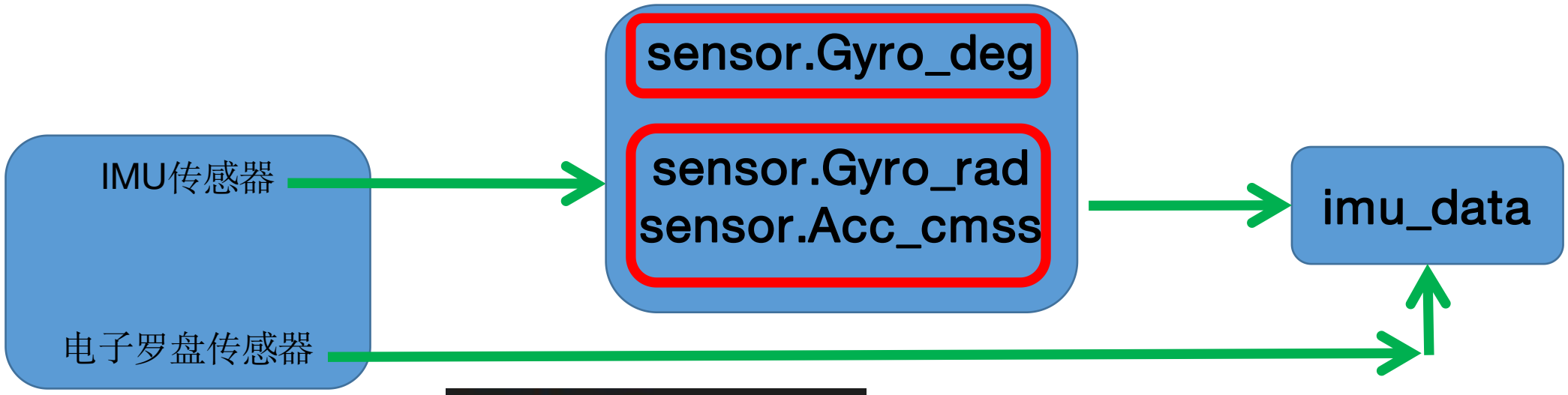
俯仰摇杆
CH_N[CH_PIT]

横滚摇杆
CH_N[CH_ROL]

输出

fs.speed_set_h[Z]
fs.speed_set_h[X]
fs.speed_set_h[Y]

无人机软件设计



sensor.Gyro_deg

imu_data

```
typedef struct
{
    float w;//q0;
    float x;//q1;
    float y;//q2;
    float z;//q3;

    float x_vec[VEC_XYZ];
    float y_vec[VEC_XYZ];
    float z_vec[VEC_XYZ];
    float hx_vec[VEC_XYZ];

    float a_acc[VEC_XYZ];
    float w_acc[VEC_XYZ];
    float h_acc[VEC_XYZ];
```

```
    float h_mag[VEC_XYZ];

    float gacc_deadzone[VEC_XYZ];

    float obs_acc_w[VEC_XYZ];
    float obs_acc_a[VEC_XYZ];
    float gra_acc[VEC_XYZ];

    float est_acc_a[VEC_XYZ];
    float est_acc_h[VEC_XYZ];
    float est_acc_w[VEC_XYZ];
```

```
    float est_speed_h[VEC_XYZ];
    float est_speed_w[VEC_XYZ];


    float rol;
    float pit;
    float yaw;
} _imu_st ;
extern _imu_st imu_data;
```

无人机软件设计

WCZ_Acc_Get_Task

输入：
imu_data.w_acc[Z]

数据处理

输出：
wcz_acc_use

Flight_State_Task
飞行任务状态

设置一键起飞

未一键起飞

WCZ_Data_Calc

WCZ_Data_Calc

Att_1level_Ctrl
姿态控制---角速度环

1ms-->Att_1level_Ctrl

目标角速度：att_1l_ct.exp_angular_velocity[ROL]、
att_1l_ct.exp_angular_velocity[PIT]

反馈角速度：att_1l_ct.fb_angular_velocity[ROL]
att_1l_ct.fb_angular_velocity[PIT]
att_1l_ct.fb_angular_velocity[YAW]

PID计算（前馈值 0、目标值、反馈值、PID参数、输出量）

输出量：mc.ct_val_rol
mc.ct_val_pit
mc.ct_val_yaw

无人机软件设计

切换状态任务
Swtich_State_Task

```
void Swtich_State_Task(u8 dT_ms)
{
  switchs.baro_on = 1;

  if(sens_hd_check.dy_pmw3901_ok)      //PMW3901光流模块
  {
    if(flag.flight_mode == LOC_HOLD)
    {
      if(switchs.dy_pmw3901_on == 0)
      {
        LED_state = 14 ;       //切换指示触发2下（2闪蓝）
      }
      switchs.dy_pmw3901_on = 1;
    }
    else
    {
      if(switchs.dy_pmw3901_on)
      {
        LED_state =  24 ;      //切换指示触发1下（2闪红）
      }
      switchs.dy_pmw3901_on = 0;
    }
  }
  else
  {
    switchs.dy_pmw3901_on = 0;
  }
```

光流传感器和TOF的状态指示

```
if(sens_hd_check.tof_ok)          //TOF模块
{
  if(tof_height_mm<1900)
  {
    if(switchs.tof_on == 0)
    {
      LED_state = 14 ;          //切换指示触发1下（2闪蓝）
    }
    switchs.tof_on = 1;
  }
  else
  {
    if(switchs.tof_on )
    {
      LED_state = 24 ;          //切换指示触发1下（2闪红）
    }
    switchs.tof_on = 0;
  }
}
else
{
  switchs.tof_on = 0;
}
```

无人机软件设计

1ms任务----数据处理得到：

sensor.Gyro_deg[0]、sensor.Gyro_rad[0]、sensor.Acc_cmss[0]
sensor.Gyro_deg[1]、sensor.Gyro_rad[1]、sensor.Acc_cmss[1]
sensor.Gyro_deg[2]、sensor.Gyro_rad[2]、sensor.Acc_cmss[2]

```
/*转换单位*/
for(i =0 ;i<3;i++)
{
    /*陀螺仪转换到度每秒，量程+-2000度*/
    sensor.Gyro_deg[i] = sensor.Gyro[i] *0.06103f ;

    /*陀螺仪转换到弧度度每秒，量程+-2000度*/
    sensor.Gyro_rad[i] = sensor.Gyro[i] *RANGE_PN2000_TO_RAD ;

    /*加速度计转换到厘米每平方秒，量程+-8G*/
    sensor.Acc_cmss[i] = (sensor.Acc[i] *RANGE_PN8G_TO_CMSS);
}
```

输入：
mpu_buffer

数据处理

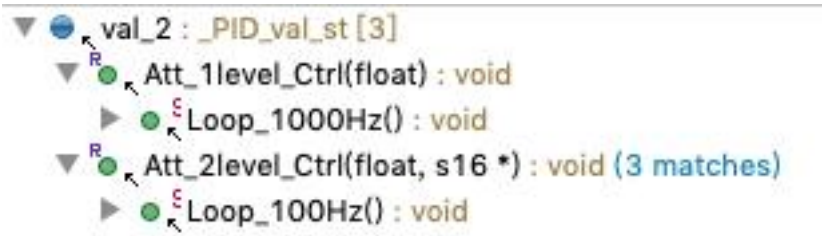
输出：
sensor.Gyro_deg
sensor.Gyro_rad
sensor.Acc_cmss

```
/*反馈角速度赋值*/
att_1l_ct.fb_angular_velocity[ROL] = sensor.Gyro_deg[X];
att_1l_ct.fb_angular_velocity[PIT] = sensor.Gyro_deg[Y];
att_1l_ct.fb_angular_velocity[YAW] = sensor.Gyro_deg[Z];
```

反馈值是经过传感器测量读取，通过数据处理后得到

```
/*目标角速度赋值*/
for(u8 i = 0;i<3;i++)
{
    att_1l_ct.exp_angular_velocity[i] = val_2[i].out;
}
```

val_2 ？



目标值：
att_1l_ct.exp_angular_velocity

PID

Motor_Ctrl_Task

反馈值：
att_1l_ct.fb_angular_velocity

Att_1level_Ctrl

无人机软件设计

Att_1level_Ctrl的PID输出val_1

```
        ct_val[i] = (val_1[i].out);
    }

/*赋值，最终比例调节*/
mc.ct_val_rol =                          FINAL_P *ct_val[ROL];
mc.ct_val_pit = X_PROPORTION_X_Y *FINAL_P *ct_val[PIT];
mc.ct_val_yaw =                          FINAL_P *ct_val[YAW];
/*输出量限幅*/
mc.ct_val_rol = LIMIT(mc.ct_val_rol,-1000,1000);
mc.ct_val_pit = LIMIT(mc.ct_val_pit,-1000,1000);
mc.ct_val_yaw = LIMIT(mc.ct_val_yaw,-400,400);
```

```
motor_step[m1] = mc.ct_val_thr +mc.ct_val_yaw -mc.ct_val_rol +mc.ct_val_pit;
motor_step[m2] = mc.ct_val_thr -mc.ct_val_yaw +mc.ct_val_rol +mc.ct_val_pit;
motor_step[m3] = mc.ct_val_thr +mc.ct_val_yaw +mc.ct_val_rol -mc.ct_val_pit;
motor_step[m4] = mc.ct_val_thr -mc.ct_val_yaw -mc.ct_val_rol -mc.ct_val_pit;
```

电机控制任务

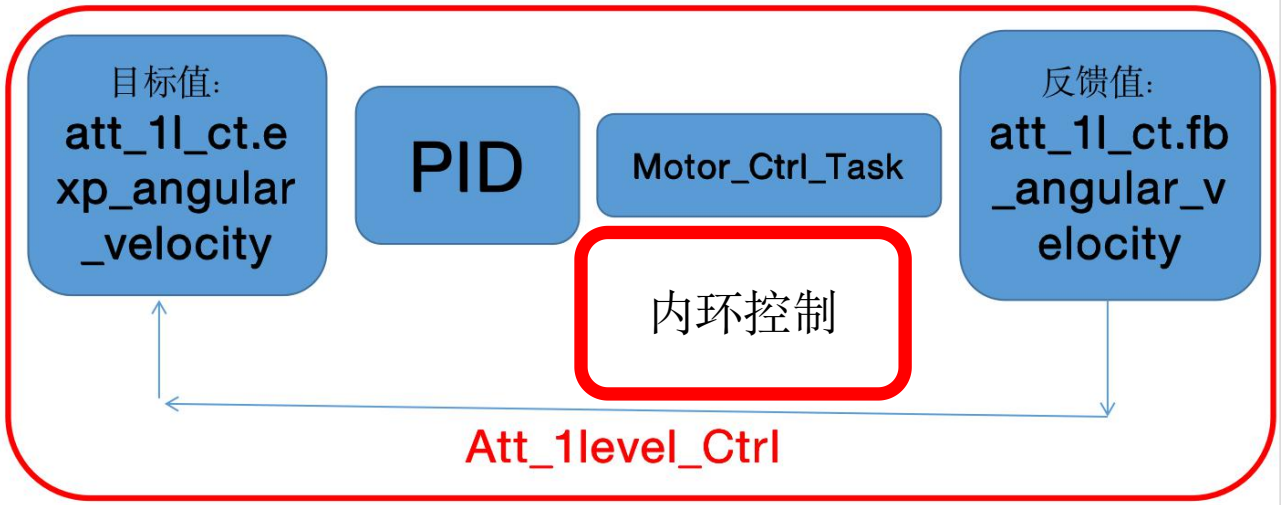
Loop_1000Hz

Motor_Ctrl_Task

1ms执行一次电机控制任务。

mc.ct_val_thr ？

无人机软件设计



1ms任务

姿态数据获取及处理

FC_Sensor_Get
Sensor_Data_Prepare
IMU_Update_Task
WCZ_Acc_Get_Task

飞行状态任务

Flight_State_Task

开关状态任务

Swtich_State_task

姿态角速度环PID控制

Att_1level_Ctrl

设置电机输出

Motor_Ctrl_Task

不接数传模块，省去DY_DT_Data_Exchange

至此，还有两个未知量

val_2 ？

mc.ct_val_thr ？

无人机软件设计

未挂载OPENMV模块，可暂时忽略500Hz的任务。

```
static void Loop_500Hz(void)     //2ms执行一次
{
    /*OpenMv控制*/
    DY_Flight_Control();
}
```

```
static void Loop_200Hz(void)     //5ms执行一次
{

}
```

保留待扩充

10ms任务：

```
        /*遥控器数据处理*/
        RC_duty_task(10);
        /*飞行模式设置任务*/
        Flight_Mode_Set(10);
        /*获取姿态角 （ROLL PITCH YAW） */
        calculate_RPY();
        /*姿态角度环控制*/
        Att_2level_Ctrl(10e-3f,CH_N);
        /*位置速度环控制*/
        Loc_1level_Ctrl(10,CH_N);
        /*高度数据融合任务*/
        WCZ_Fus_Task(10);
        /*高度速度环(内环)控制*/
        Alt_1level_Ctrl(10e-3f);
        /*高度环(外环)控制*/
        Alt_2level_Ctrl(10e-3f);
/*光流数据融合*/
if(sens_hd_check.dy_pmw3901_ok)
{
  OpticalFlow_DataFusion_Task();
}
        /*灯光控制*/
        LED_Task(10);
```

DY 德研電科

无人机软件设计

```
enum
{
    ROL = 0,
    PIT,
    YAW,
    VEC_RPY,
};
```

```
enum
{
    CH_ROL = 0,
    CH_PIT ,
    CH_THR ,
    CH_YAW ,
    AUX1 ,
    AUX2 ,
    AUX3 ,
    AUX4 ,
    CH_NUM,
};
```

```
s16 CH_N[CH_NUM] = {0,0,0,0};
```

RC_duty_task
遥控器数据处理

```
void RC_duty_task(u8 dT_ms) //建议2ms调用一次
{
    if(flag.start_ok)
    {
        ////////////获得通道数据/////////////////////
        if(DY_Parame.set.pwmInMode == PWM)
        {
            for(u8 i=0;i<CH_NUM;i++)
            {
                if(Rc_Pwm_In[i]!=0)//该通道有值，=0说明该通道未插线
                {
                    CH_N[i] = 1.2f *((s16)Rc_Pwm_In[i] - 1500); //1100 -- 1900us,处理成+-500摇杆量
                }
                else
                {
                    CH_N[i] = 0;
                }
                CH_N[i] = LIMIT(CH_N[i],-500,500);//限制到+-500
            }
        }

        ////////////////////////////////////////////
        //解锁监测
        unlock(dT_ms);
        //摇杆触发功能监测
        stick_function(dT_ms);
        //通道看门狗
        ch_watch_dog(dT_ms);
        //失控保护检查
        fail_safe_check(dT_ms);
    }
}
```

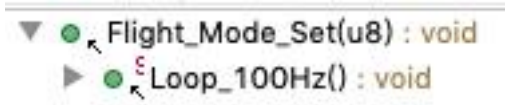
输入
油门摇杆
CH_N[CH_THR]
俯仰摇杆
CH_N[CH_PIT]
横滚摇杆
CH_N[CH_ROL]

输出
fs.speed_set_h[Z]
fs.speed_set_h[X]
fs.speed_set_h[Y]

无人机软件设计

Flight_Mode_Set

遥控器数据处理



用于判别飞行模式，一种一键起飞，悬停，一键降落。

另一种是姿态控制模式，即遥控飞行。

姿态:

flag.flight_mode = LOC_HOLD

悬停:

flag.flight_mode = LOC_HOLD

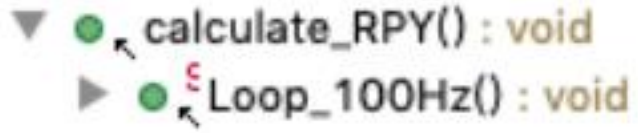
```
if(CH_N[AUX2]<-200)
{
    if(DY_Debug_Height_Mode==0)
    {
      DY_Debug_Height_Mode = 1;
      one_key_take_off();
      dy_height = 30;
    }
    else
    {
      if(tof_height_mm>=1200 && DY_CountTime_Flag==0)
      {
        dy_height = 0;
        DY_CountTime_Flag = 1;
      }
      if(DY_CountTime_Flag)
      {
        DY_Task_ExeTime++;
        if(DY_Task_ExeTime>=1500 && DY_Land_Flag==0)
        {
          DY_Land_Flag = 1;
          one_key_land();      //一键降落
        }
      }
    }
}
```

```
if(speed_mode_old != flag.speed_mode)
{
  speed_mode_old = flag.speed_mode;

  if(flag.speed_mode == 1)
  {
    LED_state = 13;
  }
  else if(flag.speed_mode == 2)
  {
    LED_state = 14;
  }
  else
  {
    LED_state = 15;
  }
}
```

```
if(CH_N[AUX1]<-200)              //-500`-200
{
  flag.flight_mode = ATT_STAB;     //姿态模式
}
else if(CH_N[AUX1]<200)    //-200~200
{
  flag.flight_mode = LOC_HOLD;     //定高悬停
}
else                               //200~500
{
  flag.flight_mode = LOC_HOLD;     //定高悬停
      one_key_land();                        //一键降落
}
```

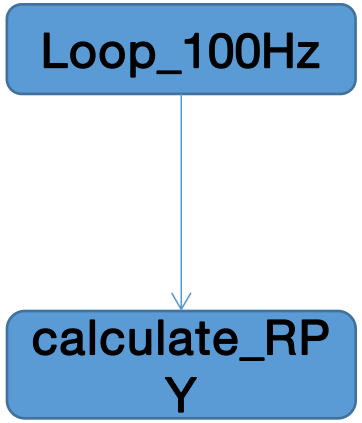
无人机软件设计

计算姿态角



R---->Roll
P---->Pitch
Y---->Yaw
10ms执行一次

```
void calculate_RPY()
{
    ////////////////////////输出姿态角////////////////////////////

    t_temp = LIMIT(1 - my_pow(t[2][0]),0,1);

    //imu_data.pit = asin(2*q1q3 - 2*q0q2)*57.30f;

    if(ABS(imu_data.z_vec[Z])>0.05f)//避免奇点的运算
    {
        imu_data.pit =  fast_atan2(t[2][0],my_sqrt(t_temp))*57.30f;
        imu_data.rol =  fast_atan2(t[2][1], t[2][2])*57.30f;
        imu_data.yaw = -fast_atan2(t[1][0], t[0][0])*57.30f;
    }
}
```



calculate_RPY
获取姿态角

无人机软件设计

Att_2level_Ctrl
姿态控制---角度环

10ms-->Att_2level_Ctrl
目标角度值：att_2l_ct.exp_rol、att_2l_ct.exp_pit、 att_2l_ct.yaw_err
反馈角度值：att_2l_ct.fb_rol、att_2l_ct.fb_pit
分别对角度RPY进行PID计算（前馈值 0、目标值、反馈值、PID参数、输出量）
输出量：val_2

loc_ctrl_1.out ？

输入

输出

```
/*角度环控制*/
void Att_2level_Ctrl(float dT_s,s16 *CH_N)
{
  /*积分微调*/
  exp_rol_tmp = - loc_ctrl_1.out[Y];
  exp_pit_tmp = - loc_ctrl_1.out[X];

  if(ABS(exp_rol_tmp + att_2l_ct.exp_rol_adj) < 5)
  {
    att_2l_ct.exp_rol_adj += 0.1f *exp_rol_tmp *dT_s;
    att_2l_ct.exp_rol_adj = LIMIT(att_2l_ct.exp_rol_adj,-1,1);
  }

  if(ABS(exp_pit_tmp + att_2l_ct.exp_pit_adj) < 5)
  {
    att_2l_ct.exp_pit_adj += 0.1f *exp_pit_tmp *dT_s;
    att_2l_ct.exp_pit_adj = LIMIT(att_2l_ct.exp_pit_adj,-1,1);
  }

  /*正负参考ANO坐标参考方向*/
  att_2l_ct.exp_rol = exp_rol_tmp + att_2l_ct.exp_rol_adj + POS_V_DAMPING *imu_data.h_acc[Y];    //exp_rol_tmp起主要作用
  att_2l_ct.exp_pit = exp_pit_tmp + att_2l_ct.exp_pit_adj + POS_V_DAMPING *imu_data.h_acc[X];

  /*期望角度限幅*/
  att_2l_ct.exp_rol = LIMIT(att_2l_ct.exp_rol,-MAX_ANGLE,MAX_ANGLE);
  att_2l_ct.exp_pit = LIMIT(att_2l_ct.exp_pit,-MAX_ANGLE,MAX_ANGLE);
```

无人机软件设计

```
/*摇杆量转换为YAW期望角速度*/
att_1l_ct.set_yaw_speed = (s32)(0.0023f *my_deadzone CH_N[CH_YAW] 0,65) *max_yaw_speed);
/*最大YAW角速度限幅*/
set_yaw_av_tmp = LIMIT(att_1l_ct.set_yaw_speed ,-max_yaw_speed,max_yaw_speed);

/*没有起飞，复位*/
if(flag.taking_off==0)//if(flag.locking)
{
  att_2l_ct.exp_rol = att_2l_ct.exp_pit = set_yaw_av_tmp = 0;
  att_2l_ct.exp_yaw = att_2l_ct.fb_yaw;
}
```

CH_N[CH_YAW]
摇杆航向控制

att_2l_ct.exp_yaw

```
/*赋值反馈角度值*/
att_2l_ct.fb_yaw = imu_data.yaw ;

att_2l_ct.fb_rol = (imu_data.rol ) ;
att_2l_ct.fb_pit = (imu_data.pit ) ;
```

IMU_Update_Task

```
imu_data = {1,0,0,0,
           {0,0,0},
           {0,0,0},
           {0,0,0},
           {0,0,0},
           {0,0,0},
            0,0,0};
```

```
/*姿态计算，更新，融合*/
IMU_update(dT_ms
*1e-3f, &imu_state,
sensor.Gyro_rad,
sensor.Acc_cmss,
mag.val,
&imu_data);
```

输入：
sensor.Gyro_rad
sensor.Acc_cmss

数据处理

输出：
imu_data

```
/*限制为+-180度*/
if(att_2l_ct.exp_yaw<-180) att_2l_ct.exp_yaw += 360;
else if(att_2l_ct.exp_yaw>180) att_2l_ct.exp_yaw -= 360;

/*计算YAW角度误差*/
att_2l_ct.yaw_err = (att_2l_ct.exp_yaw - att_2l_ct.fb_yaw);
/*限制为+-180度*/
if(att_2l_ct.yaw_err<-180) att_2l_ct.yaw_err += 360;
else if(att_2l_ct.yaw_err>180) att_2l_ct.yaw_err -= 360;

/*限制误差增大*/
if(att_2l_ct.yaw_err>90)
{
  if(set_yaw_av_tmp>0)
  {
    set_yaw_av_tmp = 0;
  }
}
else if(att_2l_ct.yaw_err<-90)
{
  if(set_yaw_av_tmp<0)
  {
    set_yaw_av_tmp = 0;
  }
}
```

无人机软件设计

目标值:
att_2l_ct.exp_rol
att_2l_ct.exp_pit
att_2l_ct.exp_ya
w_err

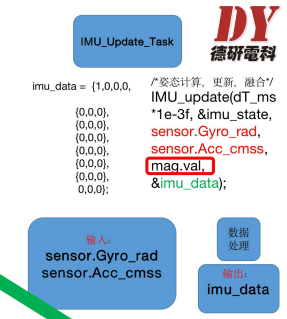
PID

反馈值:
att_2l_ct.fb_rol
att_2l_ct.fb_pit

外环控制

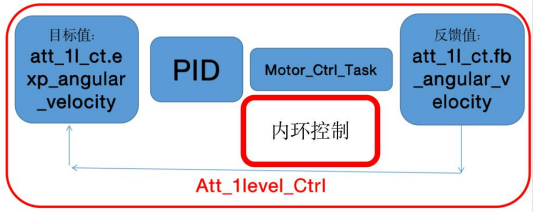
Att_2level_Ctrl

imu_data



loc_ctrl_1.out[Y]?
loc_ctrl_1.out[X]?
CH_N[CH_YAW]

所以还有三个未知量
mc.ct_val_thr ?loc_ctrl_1.out[Y]?
loc_ctrl_1.out[X]?



Att_2level_Ctrl的PID输出val_2

Att_1level_Ctrl的目标值

至此，还有两个未知量
val_2 ？解决了来源
mc.ct_val_thr ？

无人机软件设计

10ms-->Loc_1level_Ctrl

目标光流值：loc_ctrl_1.exp[X]、loc_ctrl_1.exp[Y]
修正数据量：fb_speed_fix[0]、fb_speed_fix[1]（也通过PID计算）
反馈光流值：loc_ctrl_1.fb[X]、
PID计算（前馈值 同目标值、目标值、反馈值、PID参数、输出量）
输出量：loc_ctrl_1.out

```
if(switchs.of_flow_on || switchs.dy_opticalflow_on || switchs.dy_pmw3901_on)     //光流数据有效
{

  loc_ctrl_1.exp[X] = fs.speed_set_h[X];
  loc_ctrl_1.exp[Y] = fs.speed_set_h[Y];

    if(switchs.dy_pmw3901_on)                    //PMW3901光流模块
    {
      loc_ctrl_1.fb[X] = DY_PMW_OF_DX2;
      loc_ctrl_1.fb[Y] = DY_PMW_OF_DY2;

      fb_speed_fix[0] = DY_PMW_OF_DX2FIX;
      fb_speed_fix[1] = DY_PMW_OF_DY2FIX;
    }
```

输入
油门摇杆
CH_N[CH_THR]
俯仰摇杆
CH_N[CH_PIT]
横滚摇杆
CH_N[CH_ROL]

输出
fs.speed_set_h[Z]
fs.speed_set_h[X]
fs.speed_set_h[Y]

10ms任务
OpticalFlow_DataFusion_Task

```
DY_PMW_OF_DX2 = (int16_t)pmw_pixel_flow.Vx;
DY_PMW_OF_DY2 = (int16_t)pmw_pixel_flow.Vy;
DY_PMW_OF_DX2FIX = (int16_t)pmw_pixel_flow.VxFix;
DY_PMW_OF_DY2FIX = (int16_t)pmw_pixel_flow.VyFix;
```

无人机软件设计

目标值:
loc_ctrl_1.exp[X]
loc_ctrl_1.exp[Y]

PID

反馈值:
loc_ctrl_1.fb[X]
loc_ctrl_1.fb[Y]
fb_speed_fix[X]
fb_speed_fix[Y]

内环控制

Loc_1level_Ctrl

光流传感器值

```
DY_PMW_OF_DX2 = (
DY_PMW_OF_DY2 = (
DY_PMW_OF_DX2FIX =
DY_PMW_OF_DY2FIX =
```

输入
油门摇杆
CH_N[CH_THR]
俯仰摇杆
CH_N[CH_PIT]
横滚摇杆
CH_N[CH_ROL]

输出
fs.speed_set_h[Z]
fs.speed_set_h[X]
fs.speed_set_h[Y]

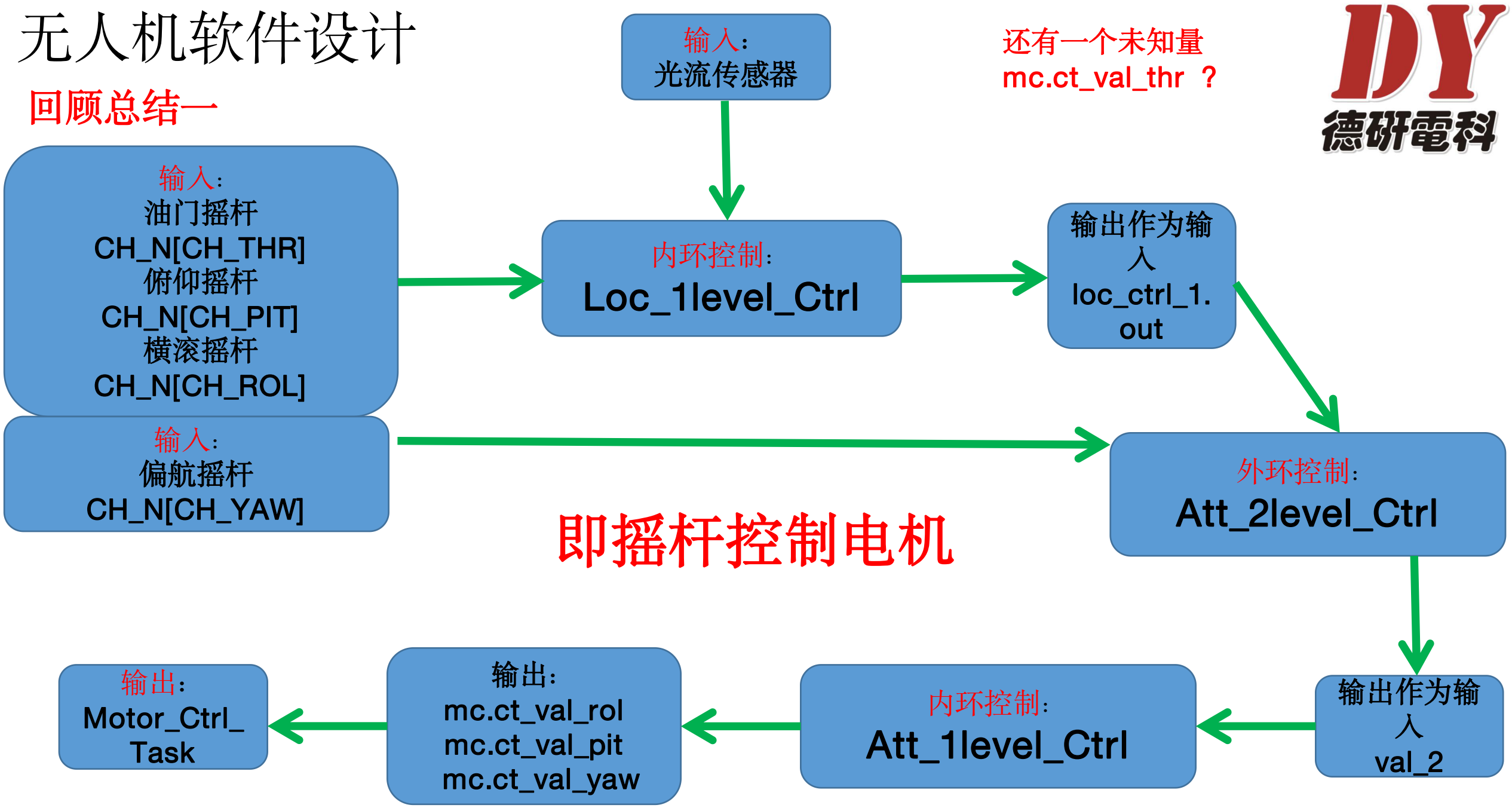
Loc_1level_Ctrl的PID输出loc_ctrl_1.out

所以还有一个未知量mc.ct_val_thr ？

loc_ctrl_1.out[Y]?loc_ctrl_1.out[X]? 解决了来源

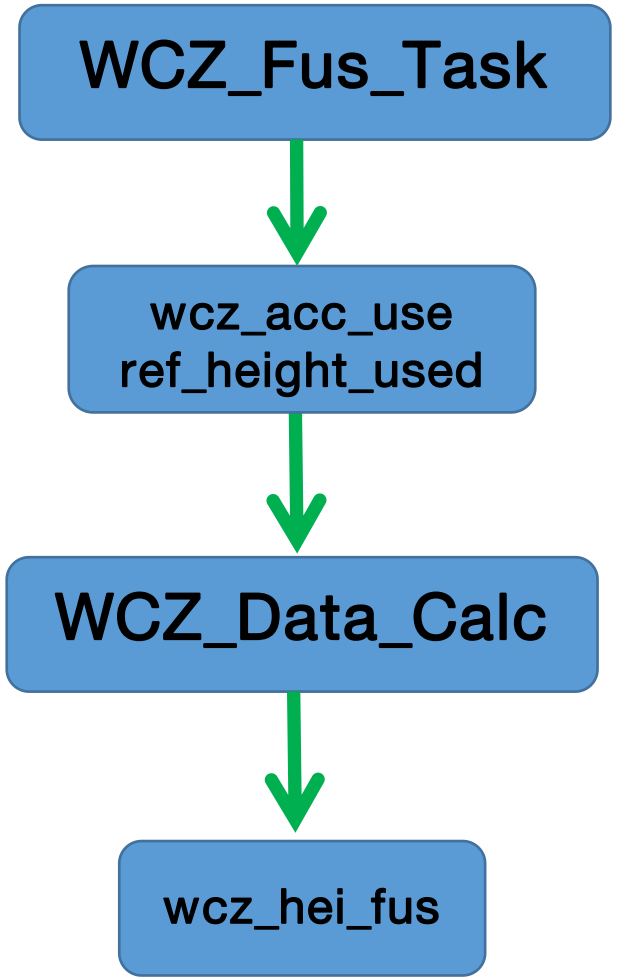
无人机软件设计

回顾总结一



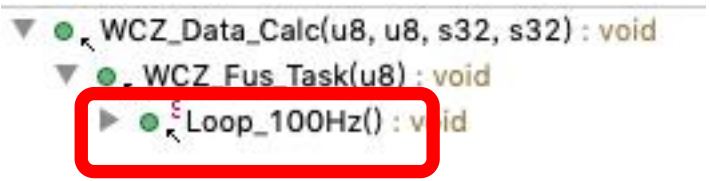
无人机软件设计

WCZ_Fus_Task




```
if(switchs.tof_on || switchs.of_tof_on) //TOF数据有效
{
    if(switchs.of_tof_on) //光流带TOF，光流优先
    {
        ref_tof_height = OF_ALT;
    }
    else
    {
        ref_tof_height = tof_height_mm/10;
    }

    if(tof_offset_ok == 1)
    {
        ref_height_get_2 = ref_tof_height + baro2tof_offset;//TOF参考高度，切换点跟随气压计

        ref_height_used = ref_height_get_2;

        tof2baro_offset += 0.5f *((ref_height_get_2 - ref_height_get_1) - tof2baro_offset);//记录气压计切换点，气压计波动大，稍微滤波一下
        //tof2baro_offset = ref_height_get_2 - ref_height_get_1;
    }
    else
    {
        baro2tof_offset = ref_height_get_1 - ref_tof_height ; //记录TOF切换点

        tof_offset_ok = 1;
    }
}
else
{
    tof_offset_ok = 0;

    ref_height_used = ref_height_get_1 ;
}
```

```
WCZ_Data_Calc(dT_ms,wcz_f_pause,(s32)wcz_acc_use, s32)(ref_height_used))
```



无人机软件设计

WCZ_Fus_Task任务：

baro_height =(s32)user_spl0601_get();一直读取

初始状况：baro_h_offset=baro_height;baro_offset_ok=1;

ref_tof_height=tof_height_mm/10;

baro2tof_offset = ref_height_get_1 - ref_tof_height ; //记录TOF切换点

tof_offset_ok = 1;

若没有一键起飞或推油门则一直执行：

ref_height_get_1 = baro_height - baro_h_offset + baro_fix + tof2baro_offset;

ref_tof_height=tof_height_mm/10;

ref_height_get_2 = ref_tof_height + baro2tof_offset;

ref_height_used = ref_height_get_2;

tof2baro_offset += 0.5f *((ref_height_get_2 - ref_height_get_1) - tof2baro_offset);

若一键起飞和推油门向上都会使：flag.taking_off置1，baro_offset_ok = 2;

ref_tof_height=tof_height_mm/10;

ref_height_get_2 = ref_tof_height + baro2tof_offset;

ref_height_used = ref_height_get_2;

tof2baro_offset += 0.5f *((ref_height_get_2 - ref_height_get_1) - tof2baro_offset);

在1ms任务中，读取气压计数据

Fc_Sensor_Get---->

baro_height = (s32)Drv_Spl0601_Read();

无人机软件设计

```
void Alt_2level_Ctrl(float dT_s)
{
    Auto_Take_Off_Land_Task(1000*dT_s);

/***************OpenMv控制***************/
    if(DY_Debug_Height_Mode == 1)
    {
        fs.speed_set_h[Z] = dy_height;
    }
/*****************************************/

    fs.alt_ctrl_speed_set = fs.speed_set_h[Z] + auto_taking_off_speed;
```

fs.alt_ctrl_speed_set ？
alt_val_2.out ？

Alt_1level_Ctrl

wcz_spe_fus.out

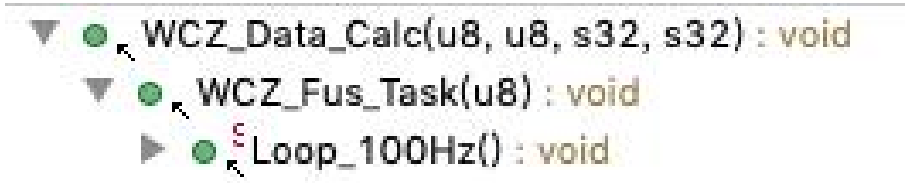
```
wcz_acc_fus.fix_ki = 0.1f;
wcz_acc_fus.in_est = wcz_acc;
wcz_acc_fus.in_obs = wcz_ref_acc;
wcz_acc_fus.ei_limit = 200;
inte_fix_filter(dT_ms*1e-3f,&wcz_acc_fus);


wcz_spe_fus.fix_kp = 0.4f;
wcz_spe_fus.in_est_d = my_deadzone(wcz_acc_fus.out,0,wcz_acc_deadzone);
wcz_spe_fus.in_obs = wcz_ref_speed;
wcz_spe_fus.e_limit = 200;
fix_inte_filter(dT_ms*1e-3f,&wcz_spe_fus);


wcz_hei_fus.fix_kp = 0.8f;
wcz_hei_fus.in_est_d = wcz_spe_fus.out;
wcz_hei_fus.in_obs = ref_height;
wcz_hei_fus.e_limit = 100;
fix_inte_filter(dT_ms*1e-3f,&wcz_hei_fus);
```

Loop_100Hz

WCZ_Fus_Task

WCZ_Data_Calc

/*高度数据融合任务*/
WCZ_Fus_Task



无人机软件设计

Alt_1level_Ctrl
输入是
fs.alt_ctrl_speed_set
alt_val_2.out

来源
fs.alt_ctrl_speed_set
alt_val_2.out

来源
Alt_2level_Ctrl

来源
Alt_2level_Ctrl的输出
alt_val_2.out

Alt_1level_Ctrl
反馈是
wcz_spe_fus.out

来源
WCZ_Fus_Task的
WCZ_Data_Calc

Alt_1level_Ctrl
输出是
alt_val_1

最终转化为
mc.ct_val_thr

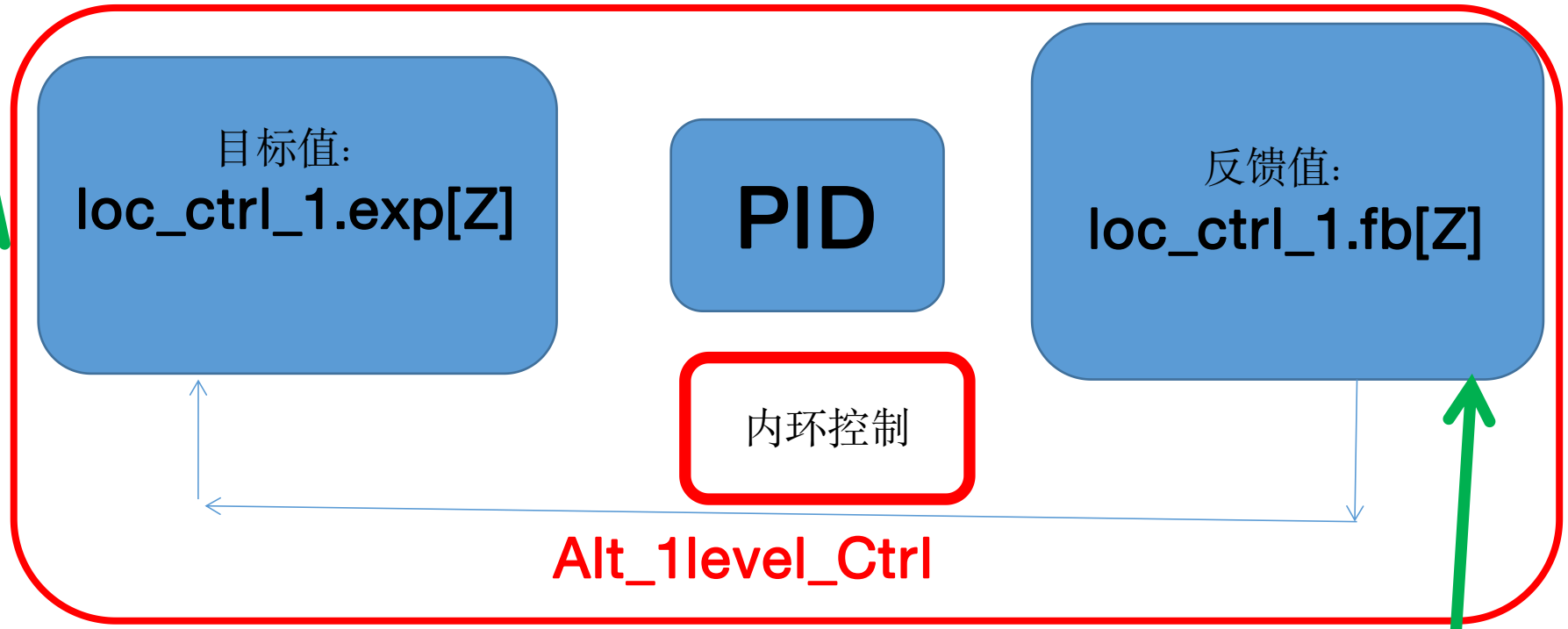
所以到目前所述
理清Alt_2level_Ctrl
的PID控制，则所以
环路即分析完成。

无人机软件设计

Alt_1level_Ctrl的PID输出mc.ct_val_thr

10ms-->Alt_1level_Ctrl

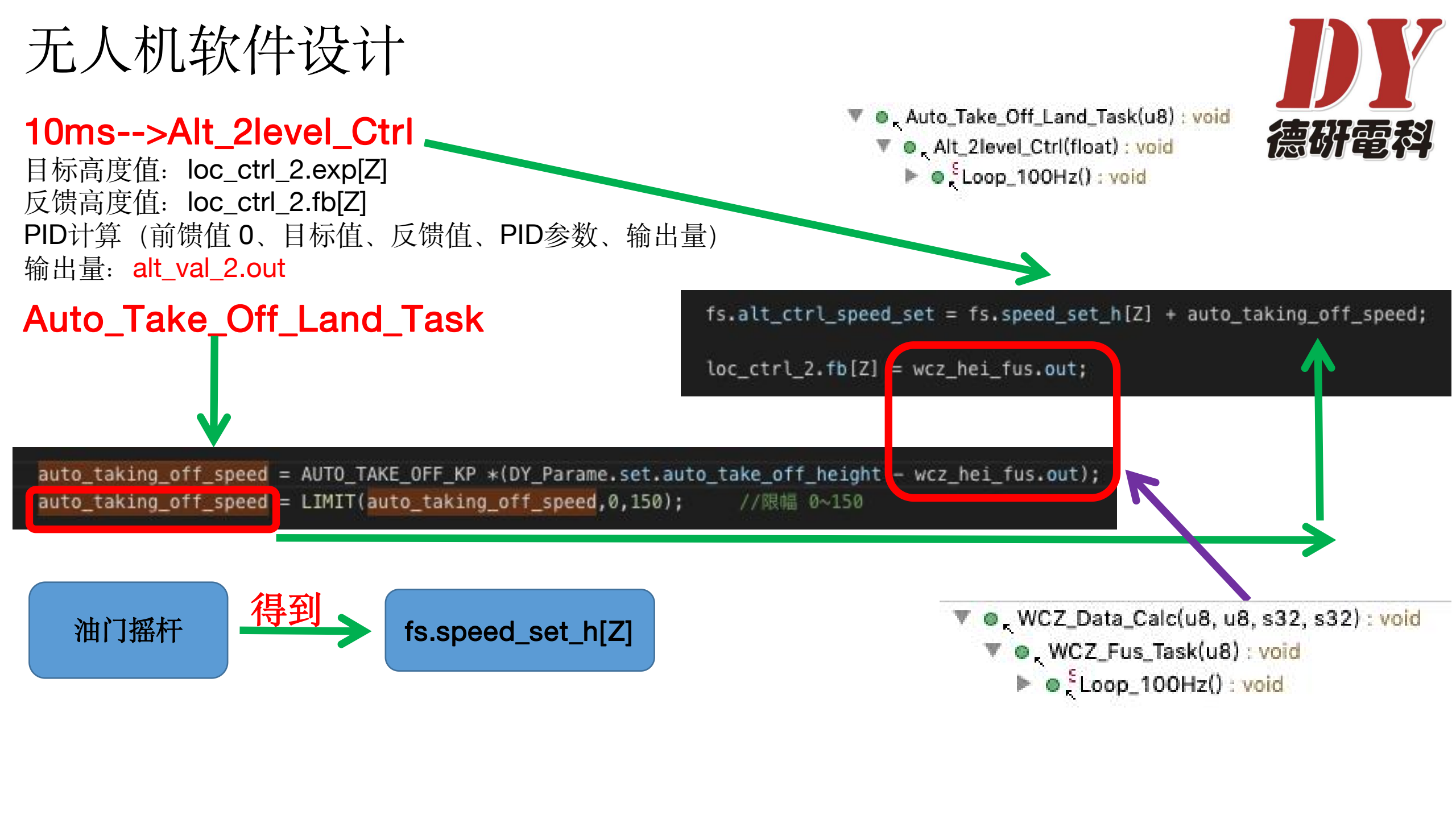
反馈值：loc_ctrl_1.exp[Z]
目标值：loc_ctrl_1.fb[Z]
PID计算（前馈值 0、目标值、反馈值、PID参数、输出量）
输出量：mc.ct_val_thr




```
loc_ctrl_1.exp[Z] = fs.alt_ctrl_speed_set + alt_val_2.out;
```

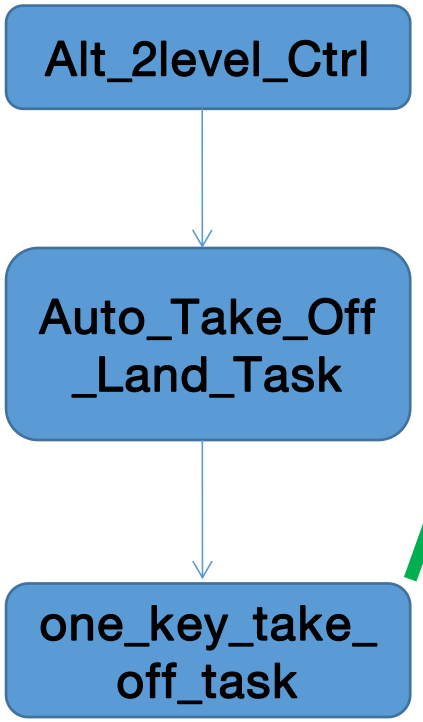
```
loc_ctrl_1.fb[Z] = wcz_spe_fus.out;
```

无人机软件设计


10ms-->Alt_2level_Ctrl
目标高度值：loc_ctrl_2.exp[Z]
反馈高度值：loc_ctrl_2.fb[Z]
PID计算（前馈值 0、目标值、反馈值、PID参数、输出量）
输出量：alt_val_2.out
Auto_Take_Off_Land_Task
Auto_Take_Off_Land_Task(u8) : void
Alt_2level_Ctrl(float) : void
Loop_100Hz() : void
fs.alt_ctrl_speed_set = fs.speed_set_h[Z] + auto_taking_off_speed;
loc_ctrl_2.fb[Z] = wcz_hei_fus.out;
auto_taking_off_speed = AUTO_TAKE_OFF_KP *(DY_Parame.set.auto_take_off_height - wcz_hei_fus.out);
auto_taking_off_speed = LIMIT(auto_taking_off_speed,0,150);
//限幅 0~150
油门摇杆
得到
fs.speed_set_h[Z]
WCZ_Data_Calc(u8, u8, s32, s32) : void
WCZ_Fus_Task(u8) : void
Loop_100Hz() : void

无人机软件设计

代码分析Alt_2level_Ctrl




```
one_key_take_off_task(dT_ms);
```

一键解锁起飞

```
if(flag.fly_ready)      //解锁
{
    if(flag.taking_off)     //起飞
    {
        if(flag.auto_take_off_land == AUTO_TAKE_OFF_NULL)
        {
            flag.auto_take_off_land = AUTO_TAKE_OFF;
        }
    }

}
else
{
    auto_taking_off_speed = 0;
    flag.auto_take_off_land = AUTO_TAKE_OFF_NULL;
}
```

auto_taking_off_speed =
AUTO_TAKE_OFF_KP
*(DY_Parame.set.auto_take_off_height -
wcz_hei_fus.out);

无人机软件设计

Alt_2level_Ctrl

```
if(fs.alt_ctrl_speed_set != 0)
{
    flag.ct_alt_hold = 0;
}
else
{
    if(ABS(loc_ctrl_1.exp[Z] - loc_ctrl_1.fb[Z])<20)
    {
        flag.ct_alt_hold = 1;
    }
}

if(flag.taking_off == 1)
{
    if(flag.ct_alt_hold == 1)       //定高悬停       flag.ct_alt_hold标志位由程序控制
    {
        PID_calculate( dT_s,            //周期（单位：秒）
                    0,              //前馈值
                    loc_ctrl_2.exp[Z],              //期望值（设定值）
                    loc_ctrl_2.fb[Z],           //反馈值（）
                    &alt_arg_2, //PID参数结构体
                    &alt_val_2, //PID数据结构体
                    100,//积分误差限幅
                    0           //integration limit，积分限幅
                    );     //输出==>alt_val2.out
    }
    else
    {
        loc_ctrl_2.exp[Z] = loc_ctrl_2.fb[Z] + alt_val_2.err;
    }
}
else
{
    loc_ctrl_2.exp[Z] = loc_ctrl_2.fb[Z];
    alt_val_2.out = 0;

}
```

没有设置起飞，则alt_val_2.out=0
设置起飞且设置了定高悬停，则进行Alt_2level_Ctrl的PID控制
若没有设置定高，则Alt_2level_Ctrl的输入主要取决于loc_ctrl_2.fb[Z]，即传感器数据融合后的高度值，高度值的改变通过控制油门摇杆实现。（建议需要将油门控制高度在TOF传感器测量范围内）

```
void Alt_2level_Ctrl(float dT_s)
{
    Auto_Take_Off_Land_Task(1000*dT_s);

/**************OpenMv控制**************/
    if(DY_Debug_Height_Mode == 1)
    {
        fs.speed_set_h[Z] = dy_height;
    }
/*****************************************/

    fs.alt_ctrl_speed_set = fs.speed_set_h[Z] + auto_taking_off_speed;

    loc_ctrl_2.fb[Z] = wcz_hei_fus.out;
```

无人机软件设计

Alt_2level_Ctrl的PID输出alt_val_2

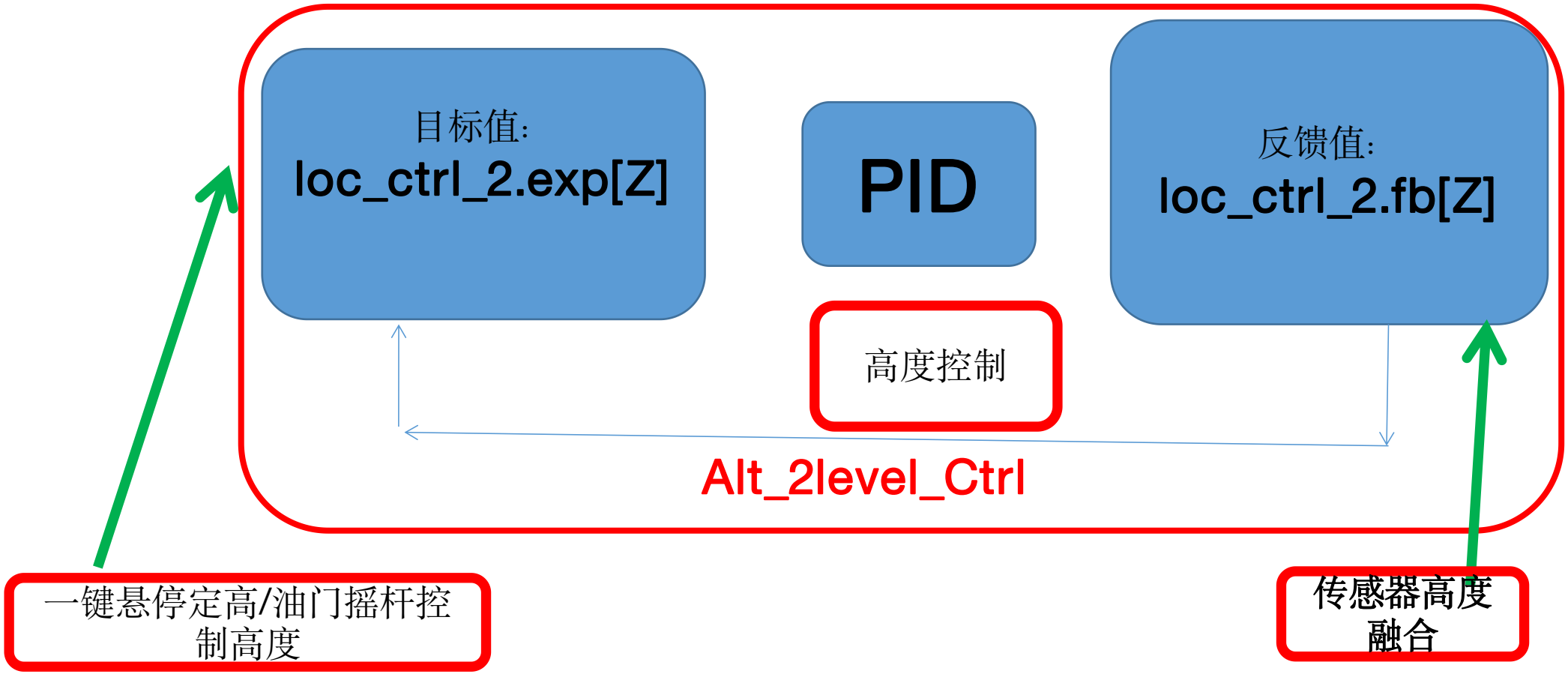
10ms-->Alt_2level_Ctrl

反馈值：loc_ctrl_2.exp[Z]

目标值：loc_ctrl_2.fb[Z]

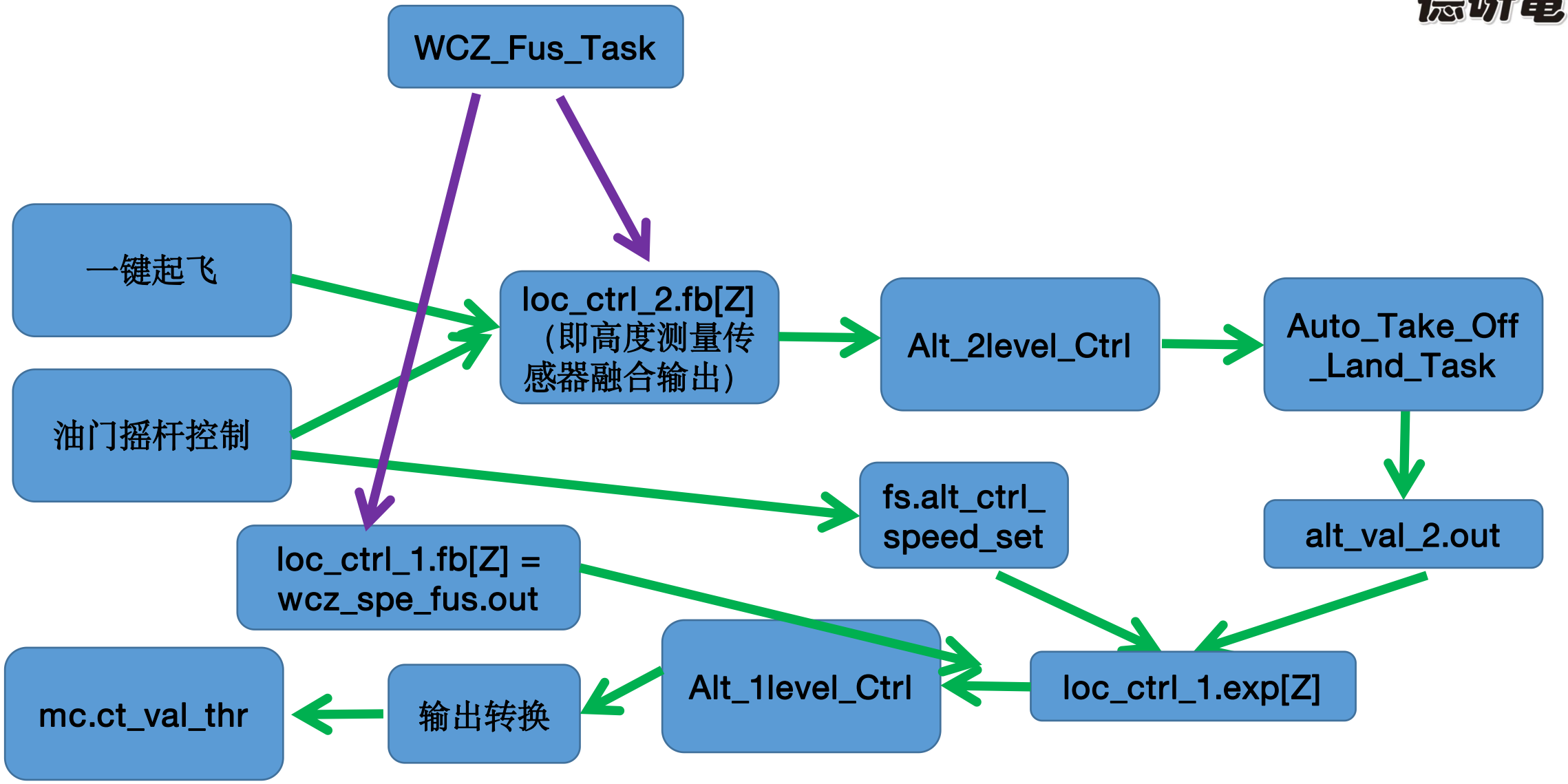
PID计算（前馈值 0、目标值、反馈值、PID参数、输出量）

输出量：alt_val_2



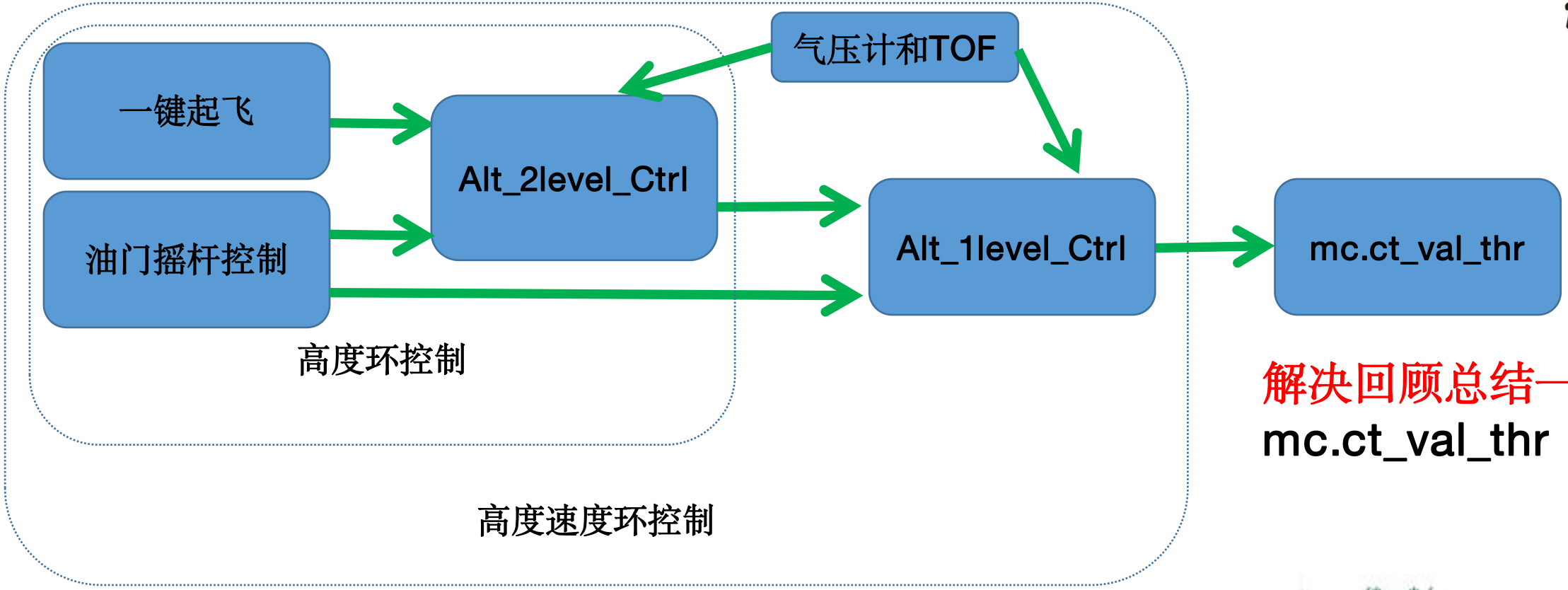
无人机软件设计

回顾总结二



无人机软件设计

进行抽象

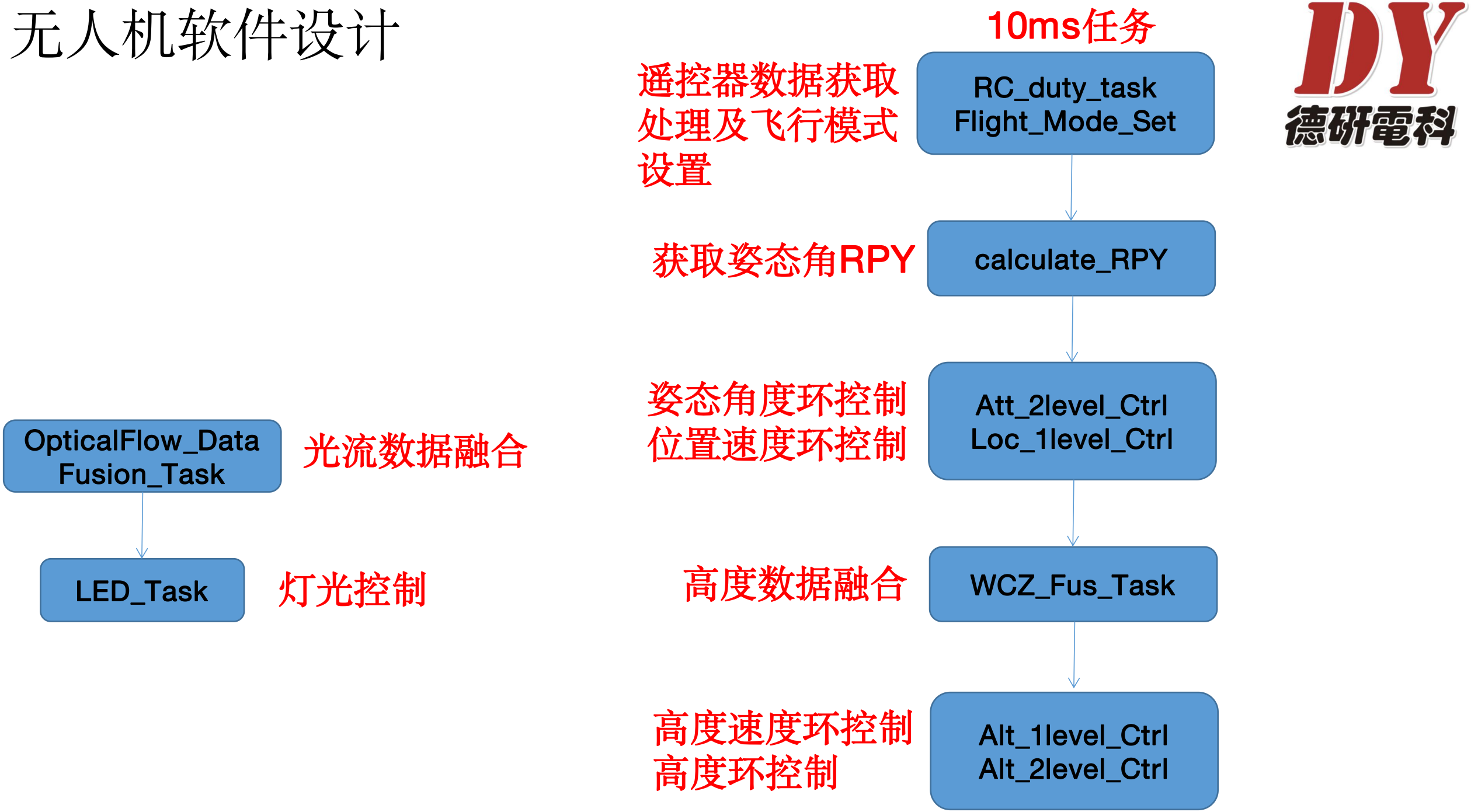


解决回顾总结一内容的
mc.ct_val_thr

第三方光流模块和标
配光流模块处理任务

```
/*--*/
DY_OF_DataAnl_Task(10);

/*光流数据融合*/
if(sens_hd_check.dy_pmw3901_ok)
{
  OpticalFlow_DataFusion_Task();
}
```

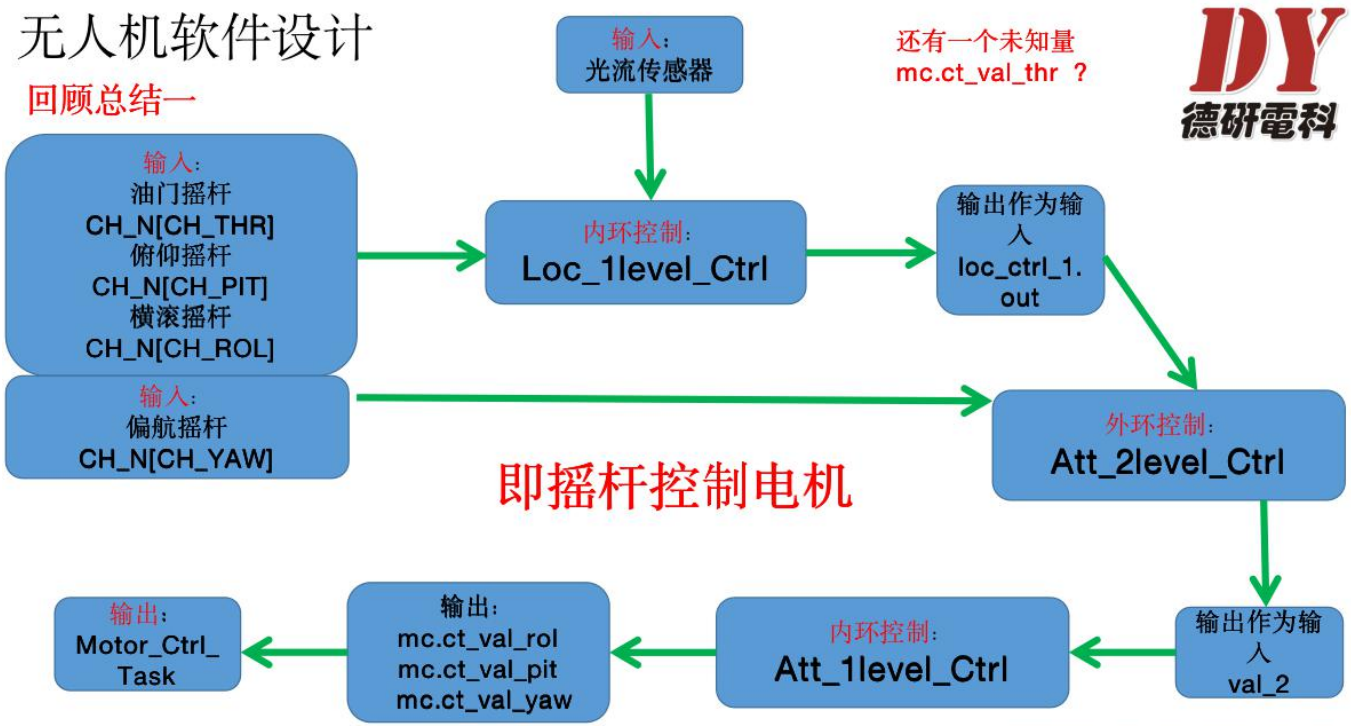
无人机软件设计

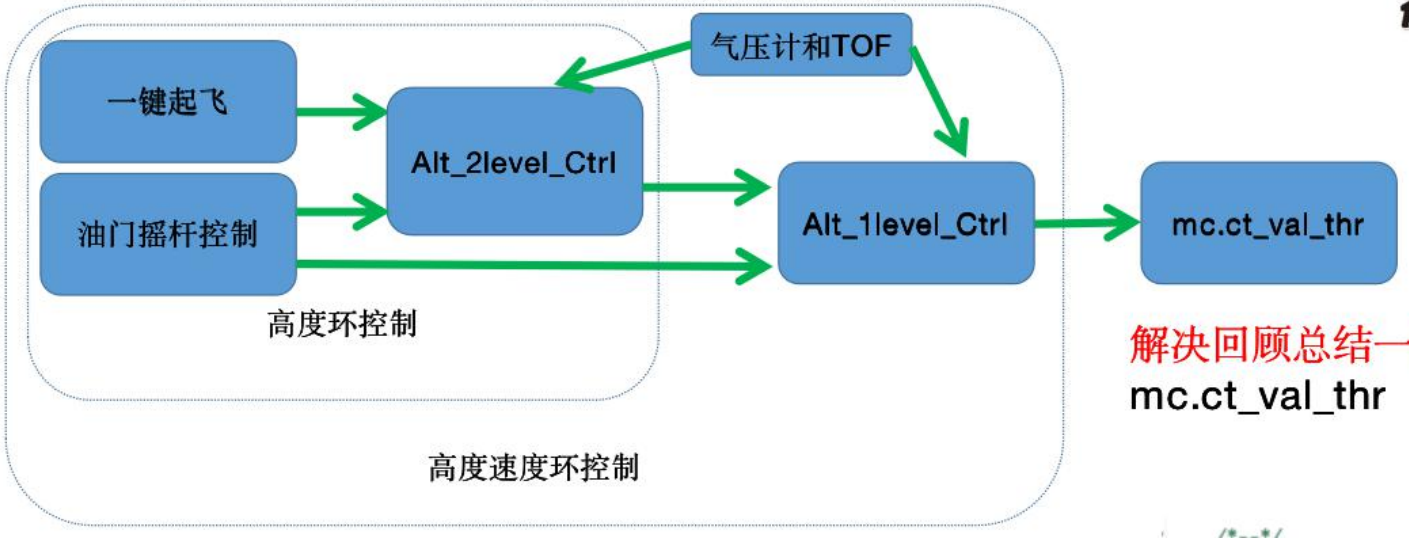

10ms任务
遥控器数据获取
处理及飞行模式
设置
RC_duty_task
Flight_Mode_Set
获取姿态角RPY
calculate_RPY
姿态角度环控制
位置速度环控制
Att_2level_Ctrl
Loc_1level_Ctrl
高度数据融合
WCZ_Fus_Task
高度速度环控制
高度环控制
Alt_1level_Ctrl
Alt_2level_Ctrl
OpticalFlow_Data
Fusion_Task
光流数据融合
LED_Task
灯光控制

无人机软件设计

无人机软件设计

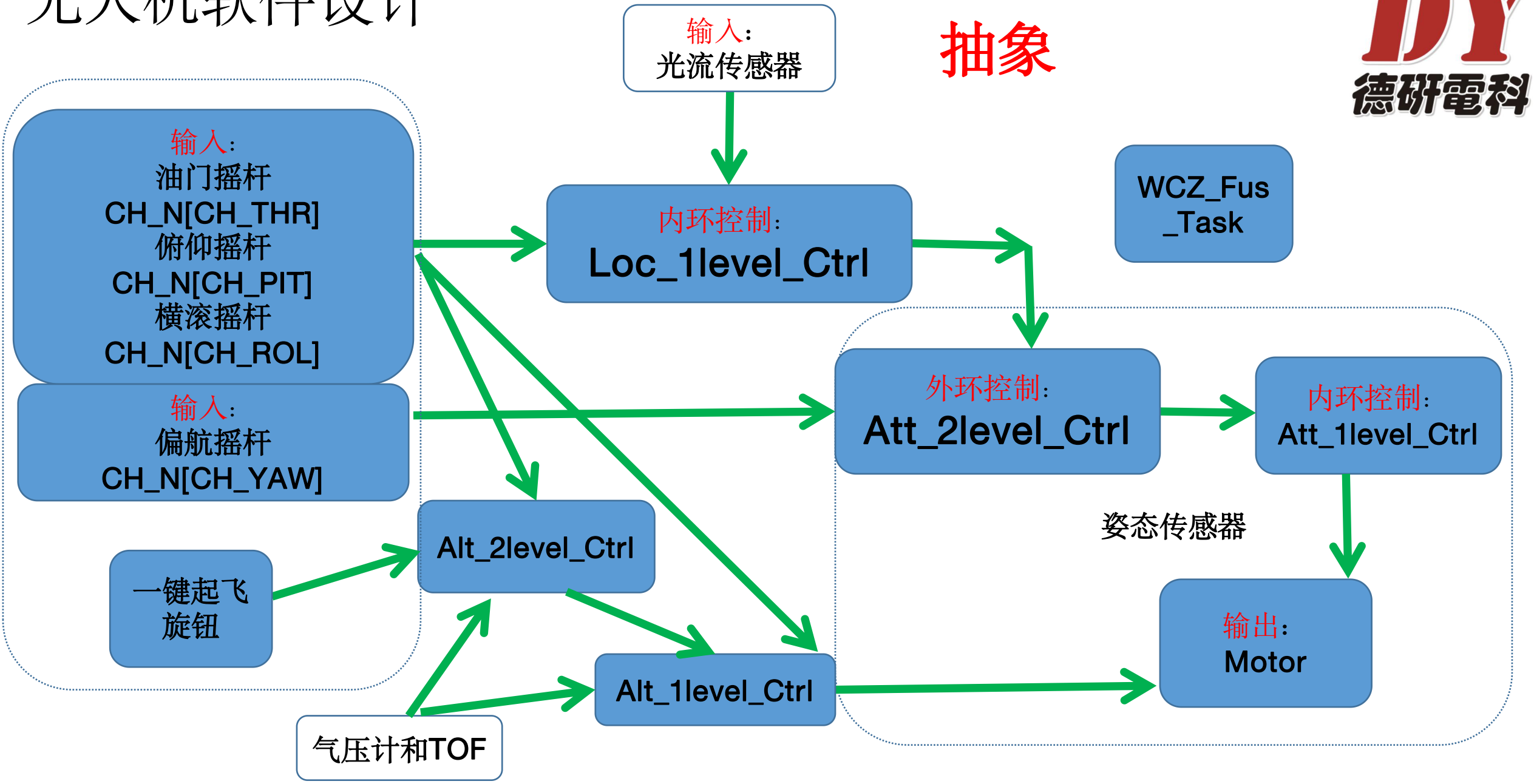
回顾总结一



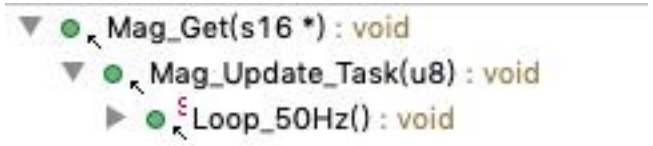


解决回顾总结一内容的
mc.ct_val_thr

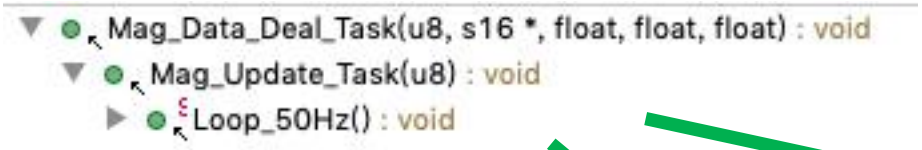
无人机软件设计


输入：
光流传感器
抽象
输入：
油门摇杆
CH_N[CH_THR]
俯仰摇杆
CH_N[CH_PIT]
横滚摇杆
CH_N[CH_ROL]
内环控制：
Loc_1level_Ctrl
WCZ_Fus
_Task
输入：
偏航摇杆
CH_N[CH_YAW]
外环控制：
Att_2level_Ctrl
内环控制：
Att_1level_Ctrl
姿态传感器
Alt_2level_Ctrl
一键起飞
旋钮
输出：
Motor
Alt_1level_Ctrl
气压计和TOF

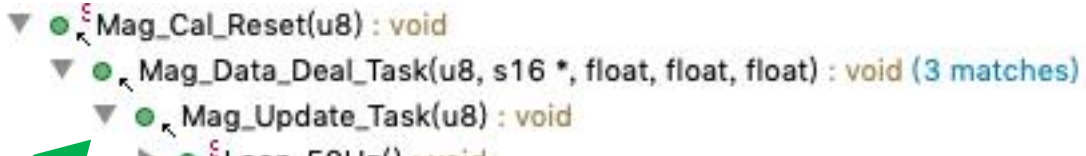
无人机软件设计



读出磁力计数据



磁力计数据处理任务



磁力计数据计算，数据复位

磁力计数据计算，XY方向数据

磁力计数据计算，Z方向数据

无人机软件设计

50ms任务

Drv_Vl53_RunTask

Tof激光任务

Power_UpdateTask

电池电压相关任务

```
void Drv_Vl53_RunTask(void)
{
    if(!VL53L0X_LINKOK)
        return;

    tof_height_mm = VL53L0X_FastRead();

    if(tof_height_mm == 0)
    {
        err_delay++;
        if(err_delay>10)
        {
            sens_hd_check.tof_ok = VL53L0X_LINKOK = 0;
        }
    }
    else
    {
        err_delay = 0;
    }
}
```

```
void Power_UpdateTask(u8 dT_ms)
{
    static s16 voltage_s16;
    float cut_off_freq;

    voltage_s16 = (s16)(adc0_value[0] *8.8653f);//1.128f

    /*****voltage_init_ok=1前后，cut_off_freq不同*****/
    if(voltage_init_ok == 0)
    {
        cut_off_freq = 2.0f;

        if(voltage_f >2000 && ABS(voltage_s16 - voltage_f) <200)
        {
            voltage_init_ok = 1;
        }
    }
    else
    {
        cut_off_freq = 0.05f;
    }
    /*********************************************/

    LPF_1_(cut_off_freq, dT_ms*1e-3f, voltage_s16,voltage_f);

    Plane_Votage = voltage_f *0.001f;

    if(Plane_Votag <DY_Parame.set.lowest_power_voltage)
    {
        flag.power_state = 3;
    }

    if(Plane_Votag <DY_Parame.set.warn_power_voltage)
    {
        if(LED_state>115)
        {
            LED_state = 1;
        }
    }

    if(Plane_Votage<DY_Parame.set.return_home_power_voltage)
    {

    }
}
```

读取TOF数据
读取电池电压

DY 德研電科

无人机软件设计

500ms任务

DY_Parame_Write_task

延时存储任务

DY_Parame_Write_task

DY_Parame_Write

```
static void DY_Parame_Write(void)
{
  All_PID_Init();                          //存储PID参数后，重新初始化PID
  DY_Parame.set.frist_init = SOFT_VER;   //初始化完毕

  Parame_Copy_Fc2para();

  Flash_SectorErase ( 0x000000, 1 );                  //擦除第一扇区
  Flash_SectorsWrite ( 0x000000, &DY_Parame.byte[0], 1 );      //将参数写入第一扇区
}
```

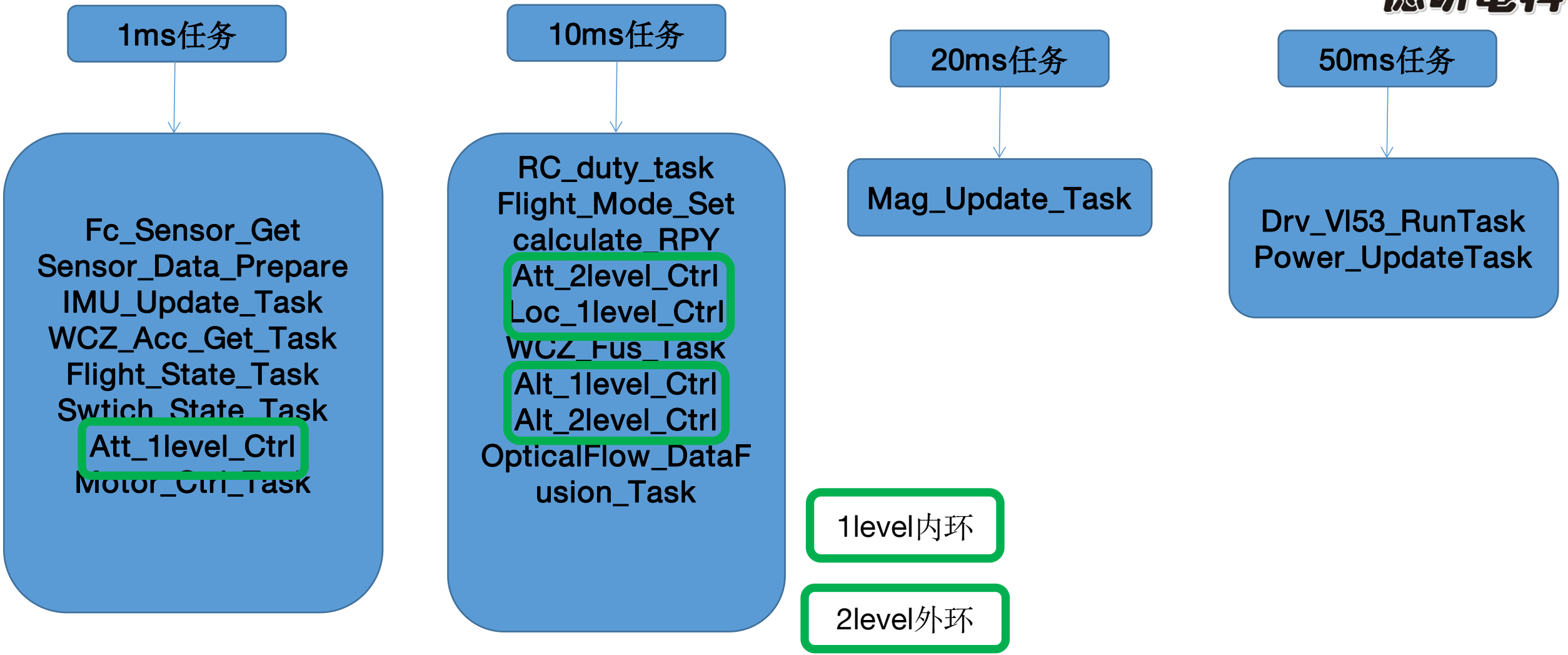
//因为写入flash耗时较长，飞控做了一个特殊逻辑，在解锁后，是不进行参数写入的，此时会置一个需要写入标志位，等飞机降落锁定后，再写入参数，提升飞行安全性

//为了避免连续更新两个参数，造成flash写入两次，我们飞控加入一个延时逻辑，参数改变后3秒，才进行写入操作，可以一次写入多项参数，降低flash擦写次数

```
__packed struct Parameter_s
{
    u16 frist_init; //飞控第一次初始化，需要做一些特殊工作，比如清空flash


    u8      pwmInMode;              //接收机模式，分别为PWM型PPM型

    float   acc_offset[VEC_XYZ];    //加速度计零偏  acc_offset[3]
    float   gyro_offset[VEC_XYZ];   //陀螺仪零偏

    float   surface_vec[VEC_XYZ];   //水平面向量
    float   center_pos_cm[VEC_XYZ]; //重心相对传感器位置偏移量

    float   mag_offset[VEC_XYZ];    //磁力计零偏
    float   mag_gain[VEC_XYZ];      //磁力计校正比例

    float   pid_att_1level[VEC_RPY][PID]; //姿态控制角速度环PID参数 pid_att_1level[3][3]
    float   pid_att_2level[VEC_RPY][PID]; //姿态控制角度环PID参数
    float   pid_alt_1level[PID];          //高度控制高度速度环PID参数
    float   pid_alt_2level[PID];          //高度控制高度环PID参数
    float   pid_loc_1level[PID];          //位置控制位置速度环PID参数
    float   pid_loc_2level[PID];          //位置控制位置环PID参数

    float   warn_power_voltage;     //警告电压
    float   return_home_power_voltage;      //返航电压
    float   lowest_power_voltage;       //最低电压

    float   auto_take_off_height;       //自动起飞高度
};
```

无人机软件设计

综上，抽象出



无人机软件设计

附件---控制状态

数据结构 --- flag	系统基本状态/传感器	系统初始化-----start_ok	
		传感器-----sensor_ok	
		静止标识-----motionless	
		电源状态-----power_state	
		无线通道使能-----wifi_ch_en	
		遥控信号丢失-----rc_loss	
		GPS*定位*-----GPS_OK,GPS_Signal_bad	未使用
	控制状态	*手动上锁* -----manual_locked	未使用
		解锁使能-----unlock_en	
		解锁标识-----fly_ready	
		低油门标识-----thr_low	
		锁定状态处理标识-----locking	
		起飞-----taking_off	
		设置偏航标识-----set_yaw	未使用
		定位悬停标识-----ct_loc_hold	未使用
		定高悬停标识-----ct_alt_hold	未使用
	飞行状态	是否处于飞行标识-----flying	
		自动降落表示-----auto_take_off_land	
		起飞点定位状态标识-----home_location_ok	未使用
		速度模式-----speed_mode	
		油门模式-----thr_mode	
		飞行模式-----flight_mode	未使用
		GPS*模式使能*-----gps_mode_en	未使用
		电机准备-----motor_preparation	
		电机锁定-----locked_rotor	未使用

软件系统主要使用的标识，未包含GPS模块，部分标识量未使用

无人机软件设计

控制状态

解锁使能-----unlock_en	flag.unlock_en=1，允许解锁标志位 flag.unlock_en=0，不允许解锁标志位
解锁标识-----fly_ready	flag.fly_ready=1，解锁 flag.fly_ready=0，不解锁
低油门标识-----thr_low	flag.thr_low=1，油门拉低 flag.thr_low=0，油门非低
锁定状态处理标识-----locking	flag.locking=0, flag.locking=1, flag.locking=2,
起飞-----taking_off	flag.taking_off=1，起飞 flag.taking_off=0，非起飞

谢谢观看！

官网

公众号

QQ群

淘宝店