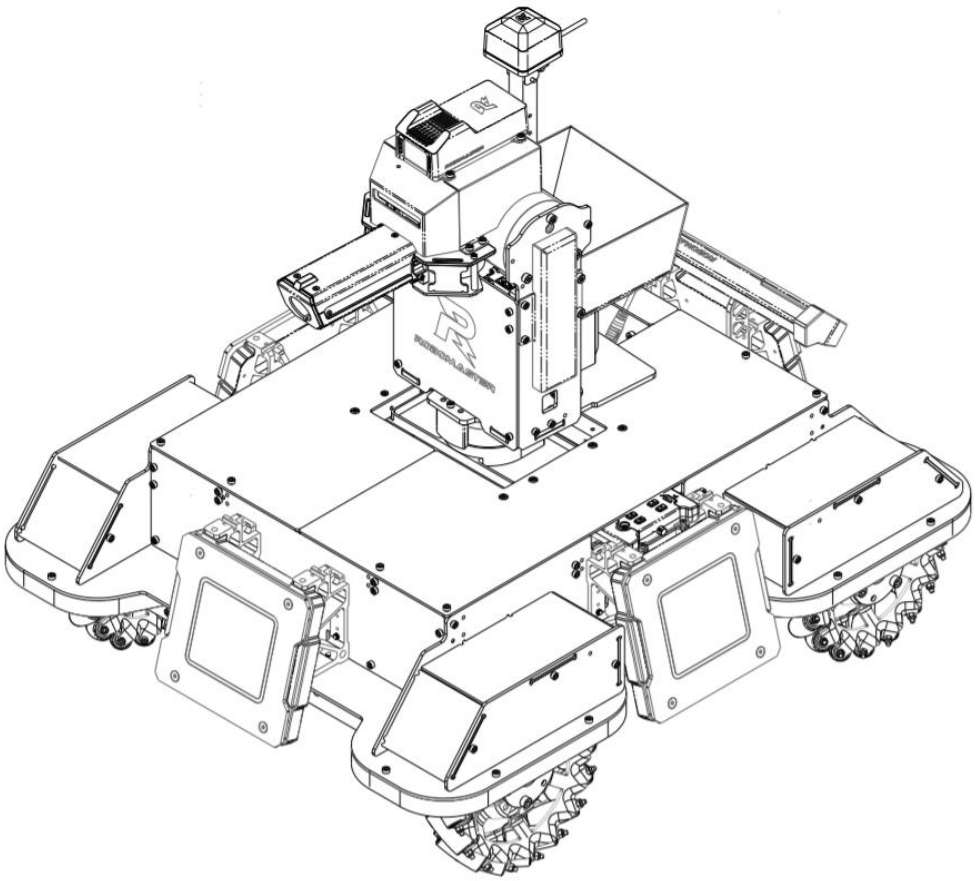
DJI RoboMaster

竞赛机器人 2020 自组装版
开发文档手册

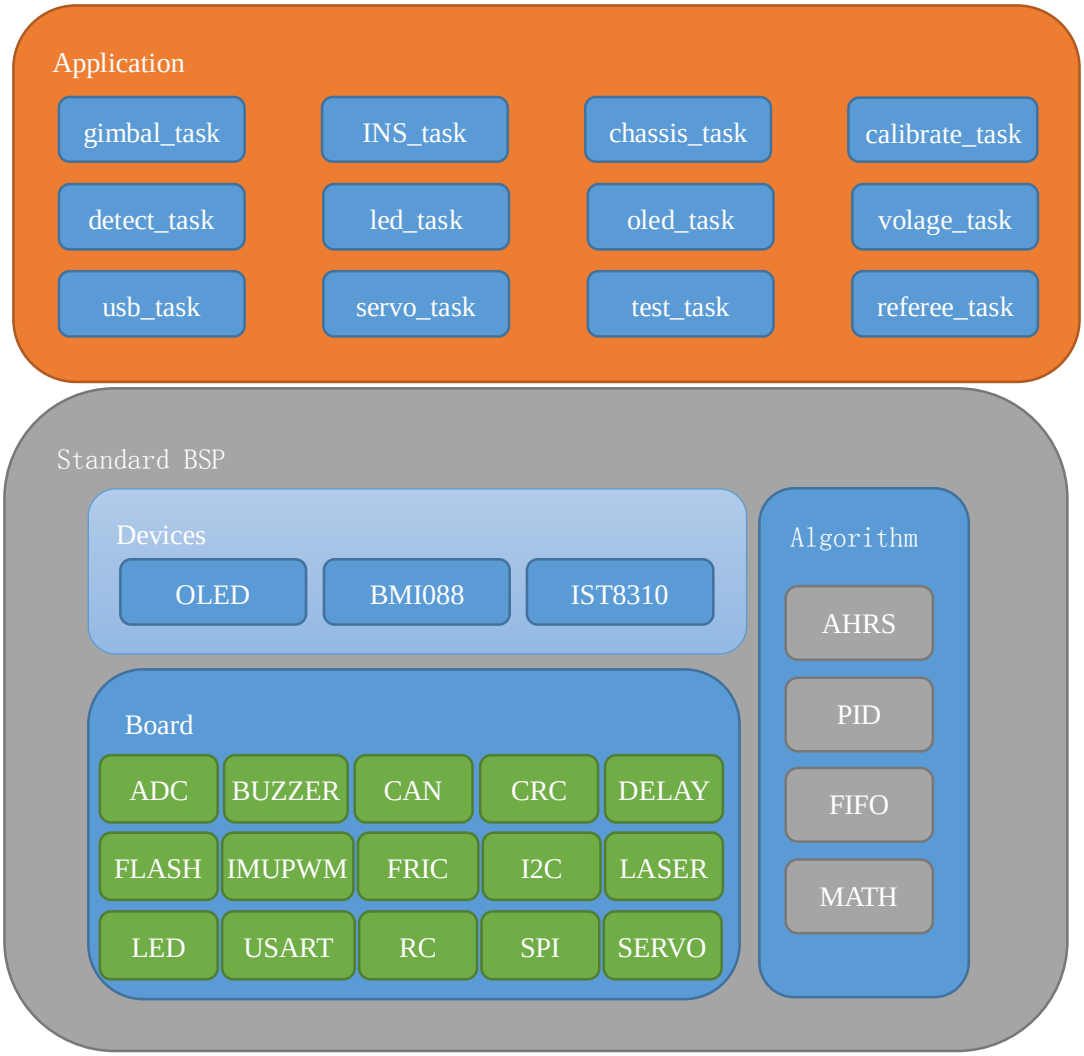
V1.0 2019.12



ROBOMASTER

目录

软件框图



文件目录

1. application：任务函数以及中断函数
2. bsp：实现对底层封装，移植标准库需要重新实现这一层级
3. components\algorithm：姿态解算算法以及 DSP 库
4. components\controller：PID 相关函数实现

5. components\devices：陀螺仪 BMI088 和 IST8310 和 OLED 的驱动
6. components\support：CRC8,CRC16 的校验函数以及 fifo 相关函数
7. doc：文档相关
8. Drivers：cubeMX 自动生成的驱动库
9. Inc：cubeMX 自动生成的 h 文件
10. MDK-ARM：cubeMX 生成的工程文件
11. Middlewares：cubeMX 生成的中间件
12. Src：cubeMX 自动生成的 c 文件

软件环境

Toolchain/IDE	MDK-ARM V5
STM32F4xx_DFP Packs	2.13.0
STM32CubeMx	5.2.1
package version	STM32Cube FW_F4 V1.21.1
FreeRTOS version	10.0.1
CMSIS-RTOS version	1.02

编程规范

变量和函数命名方式遵循 Unix/Linux 风格

不需要精确计时的任务，采用自行实现的软件定时器实现，定时精度受任务调度影响

功能介绍

1. 校准功能：提供云台校准，陀螺仪零漂校准，底盘重设 ID 的功能
2. 底盘控制功能：完成底盘的麦轮运动控制，底盘功率控制，提供 4 种控制模式： 跟随云台角度闭环控制，跟随底盘角度闭环控制，底盘旋转无角度闭环控制，原生 CAN 控制。
3. 离线判断功能：根据数据反馈的时间戳来判断设备是否离线。
4. 云台控制功能：完成云台的角度控制。提供 3 种控制模式，陀螺仪角度控制，电机码盘角度控制，原生 CAN 控制。
5. 姿态解算功能：完成陀螺仪加速度计的角度融合，解算欧拉角。
6. LED 的 RGB 切换：使用三色 LED 完成 RGB 显示，呼吸灯效。用于显示程序是否死机。
7. OLED 显示功能：将电池电量，设备错误信息显示出来，方便使用者定位问题。
8. 裁判系统数据解析：使用单字节解析裁判系统数据，适用于 2019 年裁判系统，裁判系统需要升级总决赛版本。
9. 遥控器数据解析：使用串口空闲中断函数，解析接收机发送的数据。
10. 舵机控制：将 4 个空闲的 PWM 输出舵机信号，通过按键进行控制，方便之后添加弹仓控制或者简易的机械装置。
11. 射击控制：控制下供弹装置，完成发射逻辑。

12. 电源采样：采样电源电压，并估计当前电池电量，作为简单电量判断，用于电池在机器人内部，不方便观测电量的场合。

功能实现框架

全局变量通过指针传递的方式进行，减少 extern 使用，例如遥控器指针提供 const RC_ctl_t *get_remote_control_point(void) 返回遥控器常指针，在需要遥控器参数的任务中新建一个 local_rc_ctrl 的结构体指针，通过获取指针方式传递全局变量。

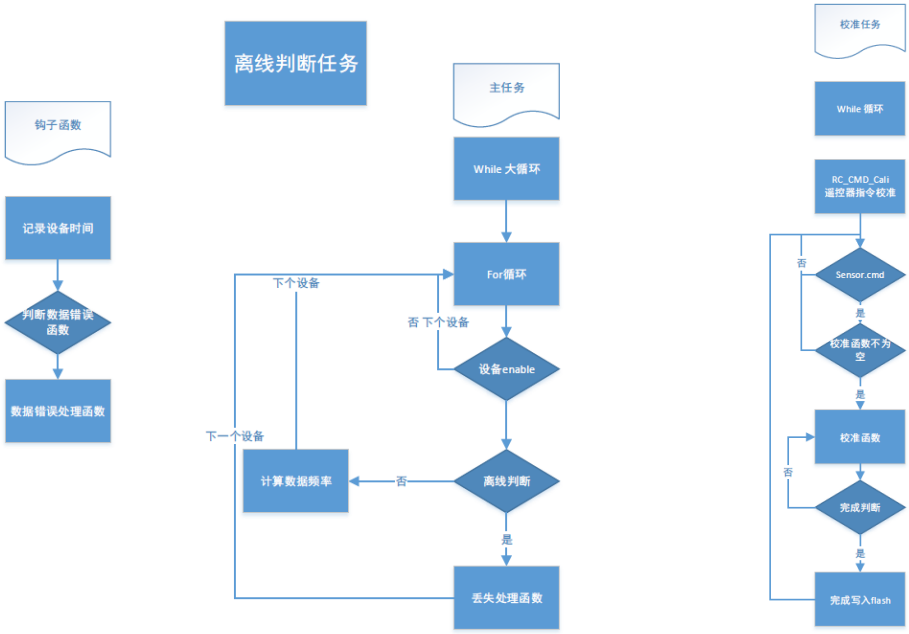
所有的全局变量均在 app 层，方便人员查看参数，不需要跳转到其他层去寻找来源，驱动层和硬件层只提供处理函数以及硬件初始化功能函数。

对于云台控制任务，底盘控制任务等，有对应的结构体控制变量集成所需要的变量，方便使用者查看。

以下介绍一些常用任务实现流程图。

a. 校准任务以及模块离线判断任务

校准任务主要完成陀螺仪零漂校准，云台中值校准，底盘进入快速设置 ID 模式。模块离线判断任务主要通过判断模块的数据发送时间，与当前系统时间的差值来判断是否掉线。这两个任务都是主要通过指针函数来完成。



离线任务中新增设备

如果需要在离线判断任务中添加一个设备，可以按照下列步骤操作。

1.第一步在 detect_task.h，添加设备名字在 errorList 的最后，像

```
enum errorList
{
    ...
    XXX_TOE,       //新设备
    ERROR_LIST_LENGHT,
};
```

2.在 detect_init 函数,添加 offlineTime（离线时间）, onlinetime（上线时间）, priority（优先级）参数

```
 uint16_t set_item[ERROR_LIST_LENGHT][3] =
 {
        ...
     {n,n,n}, //XX_TOE
};
```

3.如果有 data_is_error_fun ,solve_lost_fun,solve_data_error_fun 函数，赋值到函数指针

4.在 XXX_TOE 设备数据来的时候，添加函数 detect_hook(XXX_TOE).

校准任务中新增设备

如果需要在校准任务中添加一个设备，可以按照下列步骤操作。

1.添加设备名在 calibrate_task.h 的 cali_id_e，像

```
typedef enum
{
    ...
    //add more...
    CALI_XXX,
    CALI_LIST_LENGHT,
} cali_id_e;
```

2. 添加数据结构在 calibrate_task.h，大小必须是 4 字节倍数，像

```
typedef struct
{
    uint16_t xxx;
    uint16_t yyy;
    fp32 zzz;
} xxx_cali_t; //长度:8 字节 8 bytes，必须是 4, 8, 12, 16...
```

3.在 "FLASH_WRITE_BUF_LENGHT",添加"sizeof(xxx_cali_t)",

实现新函数 bool_t cali_xxx_hook(uint32_t *cali, bool_t cmd),

添加新名字在 "cali_name[CALI_LIST_LENGHT][3]"

申明变量 xxx_cali_t xxx_cail,

添加变量地址在 cali_sensor_buf[CALI_LIST_LENGHT]

在 cali_sensor_size[CALI_LIST_LENGHT]添加数据长度,

最后在 cali_hook_fun[CALI_LIST_LENGHT]添加函数。

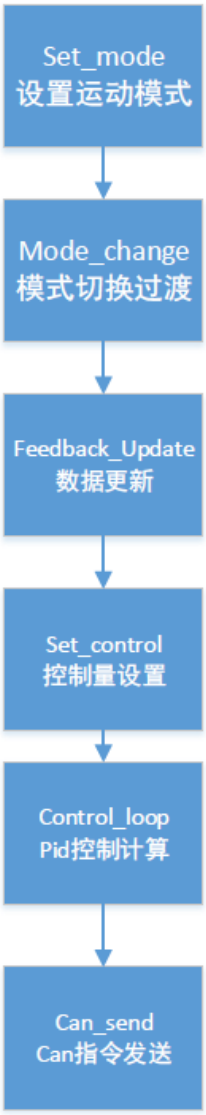
b. 云台，底盘，射击任务框架

对于云台和底盘采取两层分层结构，第一层是控制层，完成对不同控制目标的实现，第二层是行为层，对于不同的功能去实现对应的控制目标的设定。

对于云台来讲具有三种控制模式：陀螺仪角度控制，编码器角度控制，原始 CAN 控制；具有六种行为模式：无力行为，初始化行为，校准行为，绝对角度控制行为，相对角度控制行为，静止位置行为。

对于底盘来讲具有四种控制模式：跟随云台角度控制，跟随底盘角度控制，不跟随角度旋转控制，原始 CAN 控制；具有六种行为模式：无力行为，不移动行为，类步兵跟随云台行为，类工程跟随底盘角度行为，不跟随云台行为，开环行为。

射击，底盘以及云台整体流程类似，故而框架流程如下图。



过程	功能	备注
set_mode	通过遥控器设置状态机	
mode_change	状态机改变后更新的控制值	例如云台从陀螺仪切换编码器的过程，需

		要重新设置控制目标值
feedback_update	反馈数据更新	
set_control	设置控制量	
control_loop	控制器计算	
can_send	Can 发送指令	

云台控制任务中新增模式

对于云台来讲，如果要添加一个新的行为模式，可以参考如下步骤

1.首先在 gimbal_behaviour.h 文件中， 添加一个新行为名字在 gimbal_behaviour_e 中，

```
erum
{
    ...
    ...
    GIMBAL_XXX_XXX, // 新添加的
}gimbal_behaviour_e,
```

2. 实现一个新的函数 gimbal_xxx_xxx_control(fp32 *yaw, fp32 *pitch, gimbal_control_t *gimbal_control_set);

"yaw, pitch" 参数是云台运动控制输入量

第一个参数: 'yaw' 通常控制 yaw 轴移动,通常是角度增量,正值是逆时针运动,负值是顺时针

第二个参数: 'pitch' 通常控制 pitch 轴移动,通常是角度增量,正值是逆时针运动,负值是顺时针

在这个新的函数, 你能给 "yaw"和"pitch"赋值想要的参数

3. 在"gimbal_behavour_set"这个函数中，添加新的逻辑判断，给 gimbal_behaviour 赋值成 GIMBAL_XXX_XXX

在 gimbal_behaviour_mode_set 函数最后，添加"else if(gimbal_behaviour == GIMBAL_XXX_XXX)" ,然后选择一种云台控制模式

GIMBAL_MOTOR_RAW : 使用'yaw' and 'pitch' 作为电机电流设定值,直接发送到 CAN 总线上.

GIMBAL_MOTOR_ENCONDE : 'yaw' and 'pitch' 是角度增量, 控制编码相对角度.

GIMBAL_MOTOR_GYRO : 'yaw' and 'pitch' 是角度增量, 控制陀螺仪绝对角度.

4. 在"gimbal_behaviour_control_set" 函数的最后，添加

```
else if(gimbal_behaviour == GIMBAL_XXX_XXX)
{
    gimbal_xxx_xxx_control(&rc_add_yaw, &rc_add_pit, gimbal_control_set);
}
```

底盘控制任务中新增模式

如果要添加一个新的行为模式

1.首先，在 chassis_behaviour.h 文件中， 添加一个新行为名字在 chassis_behaviour_e 中

erum
{
    ...
    ...
    CHASSIS_XXX_XXX, // 新添加的
}chassis_behaviour_e,

2. 实现一个新的函数 chassis_xxx_xxx_control(fp32 *vx, fp32 *vy, fp32 *wz, chassis_move_t * chassis)

"vx,vy,wz" 参数是底盘运动控制输入量

第一个参数: 'vx' 通常控制纵向移动,正值 前进， 负值 后退

第二个参数: 'vy' 通常控制横向移动,正值 左移, 负值 右移

第三个参数: 'wz' 可能是角度控制或者旋转速度控制

在这个新的函数, 你能给 "vx","vy",and "wz" 赋值想要的速度参数

3. 在"chassis_behaviour_mode_set"这个函数中，添加新的逻辑判断，给 chassis_behaviour_mode 赋值成 CHASSIS_XXX_XXX

在函数最后，添加"else if(chassis_behaviour_mode == CHASSIS_XXX_XXX)" ,然后选择一种底盘控制模式

4 种:

CHASSIS_VECTOR_FOLLOW_GIMBAL_YAW : 'vx' and 'vy'是速度控制，'wz'是角度控制 云台和底盘的相对角度

你可以命名成"xxx_angle_set"而不是'wz'

CHASSIS_VECTOR_FOLLOW_CHASSIS_YAW : 'vx' and 'vy'是速度控制，'wz'是角度控制 底盘的陀螺仪计算出的绝对角度

你可以命名成"xxx_angle_set"

CHASSIS_VECTOR_NO_FOLLOW_YAW : 'vx' and 'vy'是速度控制， 'wz'是旋转速度控制

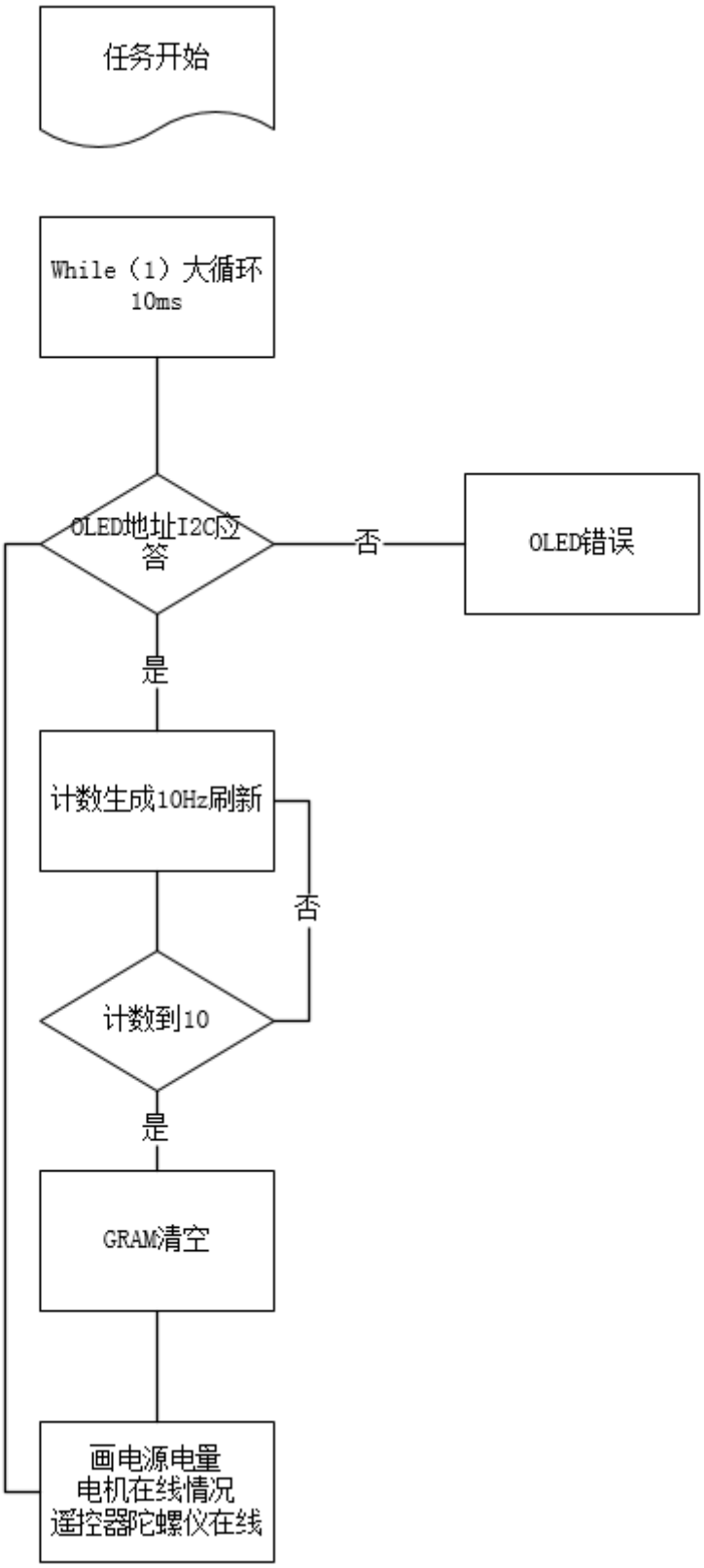
CHASSIS_VECTOR_RAW : 使用'vx' 'vy' and 'wz'直接线性计算出车轮的电流值，电流值将直接发送到 can 总线上

4. 在"chassis_behaviour_control_set" 函数的最后，添加

```
else if(chassis_behaviour_mode == CHASSIS_XXX_XXX)
{
    chassis_xxx_xxx_control(vx_set, vy_set, angle_set,
chassis_move_rc_to_vector);
}
```

c. OLED 任务框架

OLED 任务通过 100Hz 查询 oled 的 I2C 地址来确认连接情况， 10Hz 刷新 OLED 屏幕。OLED 支持 SSD1306 驱动的单色和双色模块.对于单色和双色模块，请在 OLED.C 文件中修改对应的宏定义即可。


任务开始
While（1）大循环
10ms
OLED地址I2C应答
否
OLED错误
是
计数生成10Hz刷新
否
计数到10
是
GRAM清空
画电源电量
电机在线情况
遥控器陀螺仪在线

邮箱：robomaster@dji.com
论坛：http://bbs.robomaster.com
官网：http://www.robomaster.com
电话：0755-36383255（周一至周五10:00-19:00）
地址：广东省深圳市南山区西丽镇茶光路1089号集成电路设计应用产业园2楼202