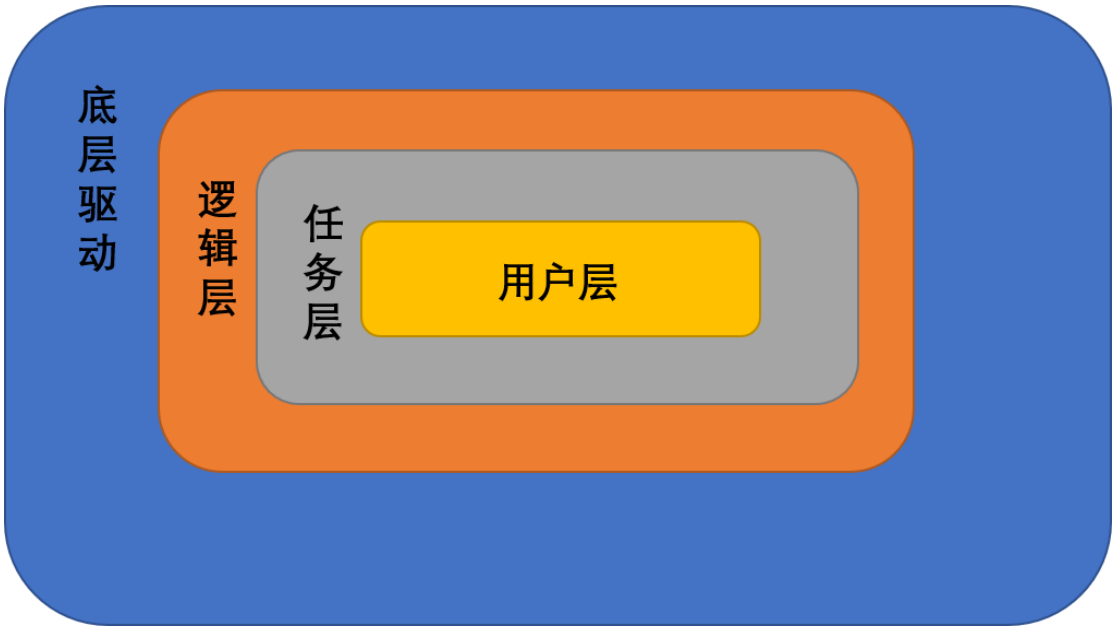
底盘代码详解

李心涛 2022/6/28

目录

1. 整体框架

1.1 整体框架图



1.2 框架简单解释

1. 底层驱动：包含 Algorithm 和 Bsp 两部分
2. 逻辑层：包含 Application
3. 任务层：包含 Task
4. 用户层：包含 UI 以及未实现的用户交互部分

2. 底层驱动

2.1 Bsp 文件

BspMotor.c/h

功能：实现电机控制指令的发送
函数：

1. void ChassisCMD(int16_t motor1, int16_t motor2, int16_t motor3, int16_t motor4)
motor:赋予底盘 3508 电机电流值，注意，此值范围未[-16383,16384]。
备注：所占 CAN 总线 ID 为 0x200

CanReceiveDecom.c/h

功能：本代码主要负责底盘电机数据的接收以及云台下发数据的接收
函数：

1. void can_filter_init(void)
   功能：一、配置 CAN 的筛选器；二、使能 CAN
   备注：启用 FIFO_0，掩码模式，对 ID 全通
2. void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
   功能：CAN 的接收回调函数
   备注：根据 CAN 总线的 ID 选择相对应的解包函数
3. const motor_measure_t *GetChassisMeasure(uint8_t i)
   功能：返回底盘电机数据指针
   备注：充分确保底盘数据的安全，在使用过程中不会被修改
4. const motor_measure_t *GetYawMeasure(void)
   功能：返回 YAW 轴电机数据的指针
   备注：同 3

重要的数据结构体：

```
//底盘电机数据指针
motor_measure_t ChassisMotor[4];
//Yaw 轴电机数据指针
motor_measure_t YawMotor;
//射击状态与射击目标的信息
Aim_t Aim;
typedef __PACKED_STRUCT
{
    uint8_t AimStatus;
    uint8_t AimTarget;
}Aim_t;
//云台向底盘发送的指令
PTZ_t PTZ;
typedef __PACKED_STRUCT
{
    uint8_t AimTargetRequest;
    int16_t FBSpeed;
    int16_t LRSpeed;
    uint8_t ChassisStatueRequest;
    uint8_t PTZStatusInformation;
}PTZ_t;
```

注意：Aim_t 、PTZ_t 这两个结构体的使用，参见《CAN 总线数据内容及数据格式规定》

ChassisSendProtocol.c/h

功能：本代码主要负责底盘向上位机发送数据整理，负责将数据打包

```
//底盘功率和枪口热量
uint8_t *send_power_heat_data(void);
//枪口射速
uint8_t *send_bullet_speed(void);
//发射机构上限
uint8_t *send_bullet_limit(void);
//底盘功率上限
uint8_t *send_power_limit(void);
//机器人自身信息
uint8_t *send_robot_information(void);
//发送敌方血量
uint8_t *send_enemy_information(void);
```

bsp_referee.c/h

功能：裁判系统串口 6 的空闲中断接收初始化与 DMA 发送使能
函数：

1. void usart6_init(uint8_t *rx1_buf, uint8_t *rx2_buf, uint16_t dma_buf_num)
   功能：空闲中断初始化
2. void usart6_tx_dma_enable(uint8_t *data, uint16_t len)
   功能：发送函数 DMA 使能

vofa.c/h

功能：串口 1，用于串口重定向发送到 VOFA+上位机，用于数据观察
备注：理论上强调不使用半主机（no semihosting）模式就可以不使用微库，但是实际操作发现必须要使用，不使用微库会导致代码卡死，这里比较未知。

2.2 Algorithm 文件

这部分属于基础代码，没有多余内容需要强调。
注意：pid.c/h 文件多添加了 PID_ANGLE 模式

3. 逻辑层

Application

由于电容任在开发中，超级电容部分相关函数被暂时不做解释。

ChassisPowerBehaviour.c/h

功能：实现底盘的功率闭环控制
函数
1. void ChassisPowerloop()
   功能：刷新底盘控制的相关数据
2. void ChassisReduceRate()
   功能：计算电流削减比例

这一部分包含底盘功率闭环的重要核心，后面会单独详细介绍

RefereeBehaviour.c/h

功能：这一部分是实现和裁判系统的数据接收与上云台及视觉所需数据包形成的代码，主要使用 FIFO 完成。
函数：
1. void init_referee_struct_data(void)
   功能：分配变量空间
2. void referee_data_solve(uint8_t *frame)
   功能：裁判系统数据的获取以及上云台与视觉所需数据包的合成
3. ……【FIFO 相关数据包】
   功能：实现双 FIFO 功能，注意，索引为 0 的 FIFO【如果放置于 buffer1】弹出优先级高，需要等到 buffer1 数据弹完，buffer2 的数据才开始弹出。

RM_Cilent_UI.c/h

功能：UI 绘制的基本函数，这份 BSP 库由“山东理工大学 齐奇战队“提供，注释完整，不做说明。
注意：结构体的定义必须按照字节对齐的方式，即使用#pragma pack(1)、#pragma pack()函数进行对结构体定义的包裹。

4. 任务层

Task

DVTask.c/h

功能：用于数据可视化(data view),主要是向 VOFA+发送数据用于图像绘制。

RefereeTask.c/h

功能：解包由 RefereeBehaviour.c/h 形成的裁判系统原始数据包，得到裁判系统接报后的最终数据。
函数：

1. void referee_unpack_fifo_data(void)
   功能：FIFO 中单包数据的解包，因此需要放在任务里一直循环。
2. void USART6_IRQHandler(void)
   功能：串口中断不定长接收

CanSendTask.c/h

功能：向云台与视觉发送存储在 FIFO 中的数据包
函数：

1. static void Can_Send(CAN_HandleTypeDef *hcan,uint32_t id,uint8_t lenth,uint8_t *buffer)
   功能：发送指定字长与 ID 的 CAN 数据包

UI.c

功能：UI 绘制

ChassisTask.c/h

功能：控制底盘的运动
函数：

1. void ChassisInit()
   功能：将数据指针赋给结构体、初始化 PID
2. void ChassisSetmode()
   功能：设置模式
3. void ChassisContolSet()
   功能：速度参数、相位补偿参数、小陀螺速度参数的设置
4. void ChassisControlLoop()
   功能：串级 PID 双环控制得到电流
5. void BufferFunctionInit(BufferFunction_t *BufferFunction,fp32 frame_period)
   功能：缓冲函数

缓冲函数

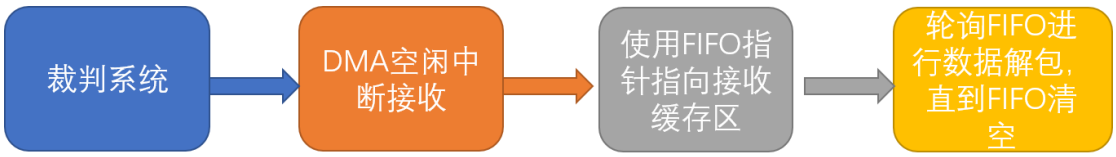
5. 通信分析

通信部分主要包含裁判系统通信以及云台、视觉通信两部分。

5.1 裁判系统通信

裁判系统通信使用串口 6 进行。

5.1.1 流程框图



5.1.2 步骤分析

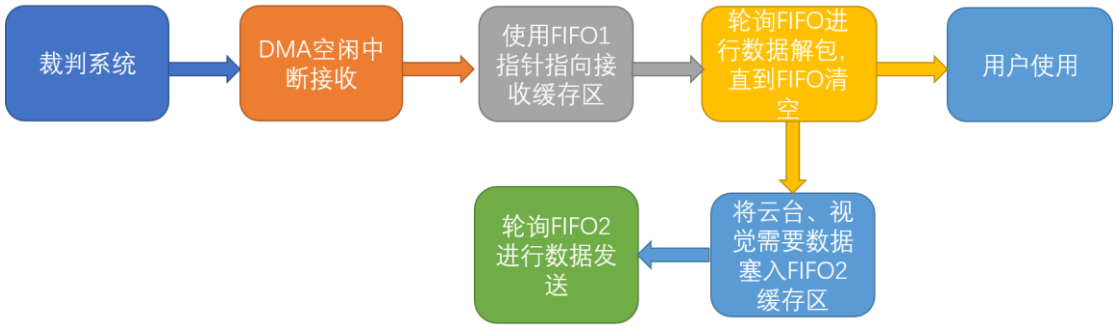
- DMA 空闲中断接收：使用双缓存区，详见 void usart6_init(uint8_t *rx1_buf, uint8_t *rx2_buf, uint16_t dma_buf_num)函数。注意这里 rx1_buf 和 rx2_buf 需要在地址空间中连续，最简单的方法解释二维数组。

- 使用 FIFO 指针指向接收缓存区：这里直接将缓存区作为 FIFO 数组缓存，由 int fifo_s_init(fifo_s_t *p_fifo, void *p_base_addr, int uint_cnt)定义，其中 uint_cnt 为双缓存区的总长度。
- 轮询 FIFO 进行数据解包，直到 FIFO 清空：这部分解包协议参考接收机/遥控器附带的大疆例程，官方并未提供具体的解包协议，只能通过代码反推。

5.2 云台、视觉通信

云台、视觉通信使用 CAN1 总线进行通信。

5.2.1 流程框图



5.2.2 步骤分析

- 将云台、视觉需要数据塞入 FIFO2 缓存区：这部分代码的实现需要先进行数据包的制作，详见"ChassisSendProtocol.c/h"文件，数据包格式为：

ID高8位	ID低8位	数据长度	8位数据

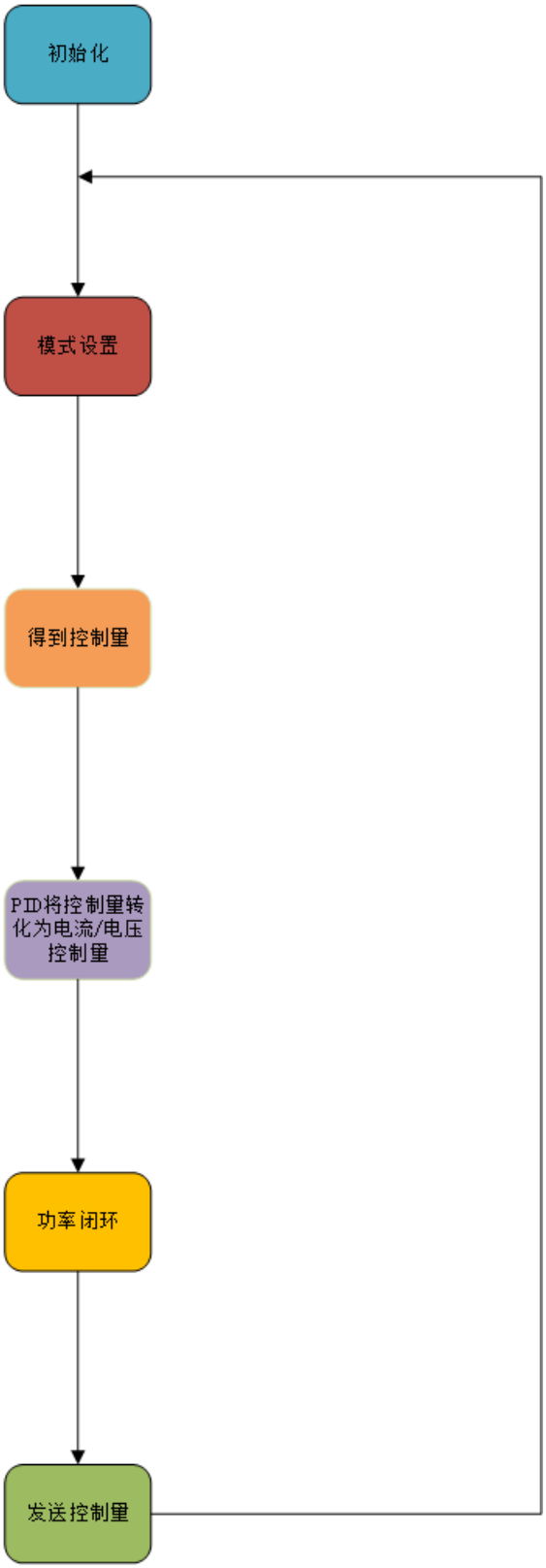
- 将云台、视觉需要数据塞入 FIFO2 缓存区：详见 void EnQueue(Queue_t *q, uint8_t *val,uint8_t lenth)函数，注意这里一共使用两个 FIFO，分为高低优先级。
- 轮询 FIFO2 进行数据发送：避免 CAN_FIFO 满，最多一次发送两包数据。这里使用 FIFO，一次发送一包数据，先轮询高优先级 FIFO0，发送完毕后发送低优先级 FIFO1。

5.2.3 综合分析

这种通信使用 FIFO 的方法，错开发送时间，可以维持个较高的数据发送效率，提高数据发送频率。

6. 功率闭环以及控制分析

6.1 框图分析


初始化
模式设置
得到控制量
PID将控制量转化为电流/电压控制量
功率闭环
发送控制量

6.2 步骤分析

- 初始化：引入数据指针以及初始化 PID 结构体
- 模式设置：根据云台发送指令确定模式。
- 设定值得到数据量：根据等级与底盘功率一对一进行对点调试，Erro_angle 用于补偿相位，rote_powkp 控制前后左右速度，roting_speed 控制小陀螺速度，调控底盘主要就是调控这三个参数。
- 根据设定值进行 PID
- 功率闭环。

6.3 理论分析

点击见链接

7. UI 分析

- 需要先添加 UI_Graph_ADD 执行一次，再执行更改 UI_Graph_Change
- Imagename 为 3 个字符，每个独立 UI 前 3 个字符必须不同，客户端才能区分
- 注意裁判系统对于带宽的需求

8. 调试效果

- 随动模式下功率 = 功率上限-5，允许开启或刹车时使用缓存功率，1 秒后缓冲功率回复，2 秒内不使用缓存功率。该模式下除了启动与刹车外，前后左右走直线，不包含启动后底盘斜着走，但是为直线的情形，必须以云台坐标系为准。
- 小陀螺不动模式下功率 = 功率上限 – 5，允许开启或刹车时使用缓存功率，1 秒后不使用缓存功率。该模式下不能产生底盘中心漂移。
- 小陀螺运动状态下功率 = 功率上限，允许使用缓存区，但离开 10-15 米的时候缓存功率处于 20W 及其以上，也不建议过多使用缓存功率
- 小陀螺下除了启动与刹车外，前后左右走直线，不包含启动后底盘斜着走，但是为直线的情形，必须以云台坐标系为准。
- 启动与刹车以及运动需要经过操作手确认，是否刹车缓慢、运动是飘的、跟随云台缓慢等等