1.目前硬件调试所需要做的工作

（1）PID 参数整定

机器人接收到速度指令后，利用 PID 控制使自身速度达到给定值。关于 PID 控制，是一种自动控制算法，网上有很多教程，如果大家学过自动控制原理这门课应该也会了解。PID 控制通常可分为位置式和增量式，位置式是基础的。我们的机器人用到的是增量式，但是两者原理基本一致。

PID 控制中一共有 3 个参数，分别是 Kp、Ki、Kd，对应比例项、积分项、微分项。这 3 个参数会影响控制系统的性能和稳定性。所谓 PID 参数整定就是调整这 3 个参数的数值（有时可能还要调整 PID 频率），使系统表现达到最佳，但是一般不用微分项，也就是只调整 Kp 和 Ki 即可。PID 参数的整定比较玄学，需要凭经验和感觉不断试错。

现在的机器人之前应该都已经做过 PID 整定了，所以估计不太需要过多调整，但是新的机器人估计要调。大家到时候也可以到场地拿一台性能较差的机器人试试手。

PID 算法在 pid.c 中实现，控制电机的 PID 参数在 cfg.h 中定义。

（2）排查机器人性能并修复

28 日晚上几个同学已经做过了初步排查，有几台因为零件坏了而基本用不了，可以拆下一些好用的零件相互拼装。其他机器人也或多或少有一些问题，路线走偏的问题居多，可能因为每个轮子和电机的性能都不太一样，而且机器人各个方向的速度是通过编码器计算电机转速转换得到的，个人感觉也不太精确。后续可以尝试在程序上乘系数、加补偿等方式看能否优化这个问题。

（3）解决串频问题

以前比赛中与别的队伍之间出现过串频问题，接下来看看能不能复现这个问题，然后通过改变通信频点或其他方法来解决这个问题。

其中频点的设置在 misc.c 文件中的 void read_dip_sw(u8 *freq, u8 *num, u8 *mode) 函数。程序会读取用于设置频点的拨码盘的数值 0~15，然后分别对应到 1~125，因为所用的无线模块 nrf24l01 有 125 个可选通道。

（4）其他改进工作

现在调整 PID 参数比较麻烦，需要在程序里面先改，然后编译、把程序烧进控制板，再测试机器人，如此反复。目前有一个思路是类似 crazy 控制机器人那样，在上位机可以直接输入 PID 参数，然后随控制指令一起发送到小车执行，最后把确定的参数刷一遍到控制板上就行，这样就可以实现在线调整参数，不用重复劳动。

另外，大家如果觉得有什么需要改进的也可以发到群里一起讨论交流。

2.控制指令格式

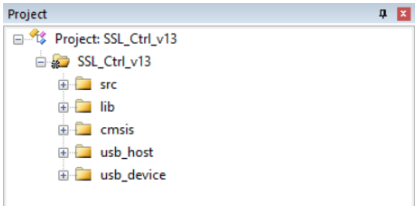
Github 上 yisi_robot 目录下的 ssl_user_manual.pdf 介绍了机器人的硬件组成以及使用说明，其中也介绍了发射机发送给机器人的数据格式。发射机每发出一个数据包，最多可以包含对 3 个机器人的控制指令，无论控制的是哪 3 个机器人，所有机器人都可以接收到数据包，然后机器人会判断接收到的数据包是否包含控制自己的指令，若有，则执行，否则跳过。

也就是说，如果场上有 3 台机器人，那么发射机每秒只需发送 60 个数据包就可达到 60Hz 的控制频率；如果有 6 台机器人，那么发射机每秒需要发送 120 个数据包，其中每个数据包只控制 3 台机器人，这样针对每一台机器人才能达到 60Hz 的控制频率。

3.硬件程序简介

通过 project 目录下的 SSL_Ctrl_v13.uvprojx 打开工程文件。打开之后会看到如下目

录，我们主要先关注 src 文件夹下的，其他大多是一些库。



先大概介绍各个文件的基本作用，有些文件非常复杂，但是大家并不需要都看懂，按需查看就行，有兴趣的同学可以深入了解。

（1）nrf24l01.c\nrf24l01_drv.c

这两个文件基本是用来驱动和控制 nrf24l01 芯片的。nrf24L01 是一个无线模块，发射机发送给机器人的指令就是通过这个模块来接收的。

（2）packet.c

主要处理接收到的数据包，包括解码数据包格式并把相关指令转换成对应的控制变量等。

（3）param.c

用于初始化机器人参数，例如 PID 参数、最大射门力度等。可以发现里面有几个宏 MOTOR_PID_KP、MOTOR_PID_KI、MOTOR_PID_KD 等，这就是 PID 的参数，定义在 cfg.h 中，我们后面改 PID 参数就在这里面修改。

（4）pid.c

实现了 PID 算法。

（5）robot.c

这个是控制机器人的关键程序，包含几个函数。

int init_robot(void)做一些初始化机器人的操作，主要还是调用底层的一些函数。

int do_power_monitor(void)每调用一次便检测一次电容的电压。

void do_timer()进行计数，计数达到一定值后便调用一次 PID 控制算法并更新电机控制量。

void do_run(void)是一个循环程序。板子上电并执行一些初始化程序之后，便进入一个循环程序。可以发现 do_run 里面会有一个 while(1)，里面会根据当前机器人的模式，不断接受外部指令，然后做出相应的操作。

void SysTick_Handler(void)是一个系统时钟中断服务程序。函数内部执行看门狗的喂狗操作、更新一个计时的变量 sys_tick（加 1）、并且执行一次 do_timer()。sys_tick 这个变量在其他很多地方都用到了，可以理解为用来获取当前系统的时间，这样方便每隔一段时间做相应的事情。

（6）system_stm32f4xx.c

比较底层的配置文件，可以不用管。

（7）timer.c

一些定时器的操作，包括延迟函数、获取当前系统时间的函数（其实就是获取 sys_tick 变量）的值。get_one_timer(u32 time)和 check_timer(timer_t timer)是一个软件实现的定时器。

（8）Usart_ble.c

与蓝牙操作相关的一些函数，目前没有用到蓝牙控制，可以先不用管。

（9）action.c

实现了机器人操作相关的一些函数，比如射门、运动、吸球等动作。最终是通过 int on_robot_command(packet_robot_t *packet)或者 int do_robot_command(packet_robot_t *packet)这两个函数来根据指令包的要求调用相关的动作的。

（10）comm.c

实现了通信数据处理，包括蓝牙模块、nrf24L01 模块、手柄等。

（11）gpio.c

主要是看门狗初始化、GPIO（通用输入输出接口）的初始化，涉及到比较底层的配置，按 F12 追踪后可以看到一些直接配置寄存器的操作，可能要结合芯片的数据手册、板子的电路图等才能看懂。

（12）Hm_bluetooth.c

实现了一些设置蓝牙模块的函数。

（13）i2c.c

实现了一些 i2c 操作的函数，主要是用于读写 EEPROM、控制陀螺仪 MPU6050 等。

（14）init.c

一些机器人的初始化功能。其中可以重点关注 int do_init(void)，里面调用了几个函数，分别负责初始化系统时钟、中断设置、GPIO、系统滴答时钟等，最后初始化整个系统。推荐从 int do_init(void)开始层层深入看代码。

（15）main.c

整个程序的入口。main 函数中只调用了两个函数，do_init()用于初始化，do_run()则不断循环，前面有说明。

（16）misc.c

一些比较偏向底层操作的函数，比较杂。基本是采样电池电压值、采样电容电压值、读取编码器、设置 PWM 这些。这些函数会被上层的踢球等操作调用。

（17）motor.c

与电机控制相关的一些操作，比如读取编码器、设置 PWM 等，也是比较偏向底层的。

（18）usb 开头的一些文件是来自 ST 官方提供的 usb 驱动库，手柄模式时会用到 usb 接口。