

朱俊泽

8-1

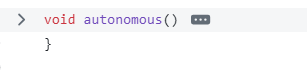
程序组

**竞赛模版：**

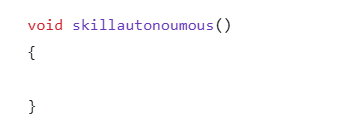
**Competition competition**

competition的实例化，写在文件头

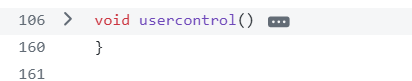
**自动阶段的函数内容**

****

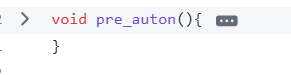
**技能赛的函数内容**

****

**手动阶段的函数内容**

****

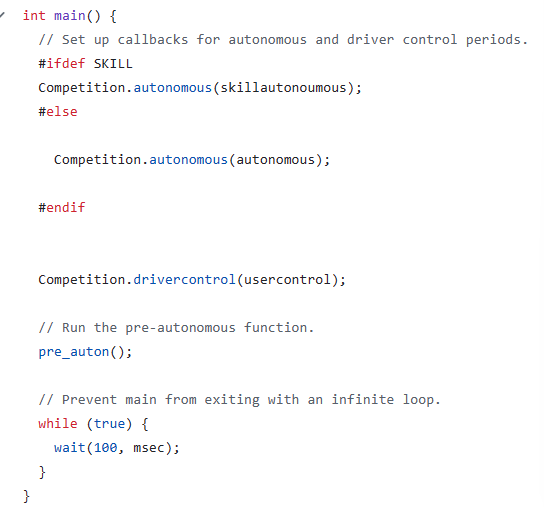
**比赛开始前的预处理（imu，雷达等）**

****

**代码框架：**

**main主函数中**

此时使用SKILL是否define来控制是技能还是自动，



**进入pre\_auton()：**

此时pre\_auton先由odom的define与否控制里程计还是雷达。

然后进行imu的安装检测，再设置imu的角度，这个角度init\_angle可在全局中调控

随后进入odom内容（轮式里程计的定位）



之后默认进入手动的线程

**Usercontrol（）**：这里进行了一系列关于电机的控制和状态的检测



**Pid：**

**麻学姐一句话太形象了**：pid是现在，过去，未来。p是比例代表现在，i是积分代表过去，d是导数代表趋势也就是未来。

**可以使用pid的场景**：位置上可以使用pid，速度上可以使用pid；

**不可以使用pid的场景**：多输入多输出的场景

**Pid调节的步骤：**

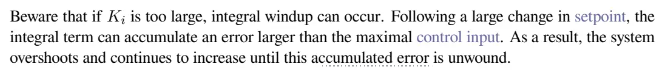
三个参数值kp，ki，kd；先都置零。

开始增加kp使得系统出现震荡。比如转向的pid增加kp使得系统出现摆头。

之后增加kd到抵消这个震荡

之后观察是否有稳态误差，如果有可以修改ki值。

！！！ki的值如果过大会过冲导致震荡、失控（也就是下文说的integral winduo）



**Pid调节的情况：**

在设定值内进行不可接受的震荡。是由于kp过大导致，可以减小kp，增大kd；

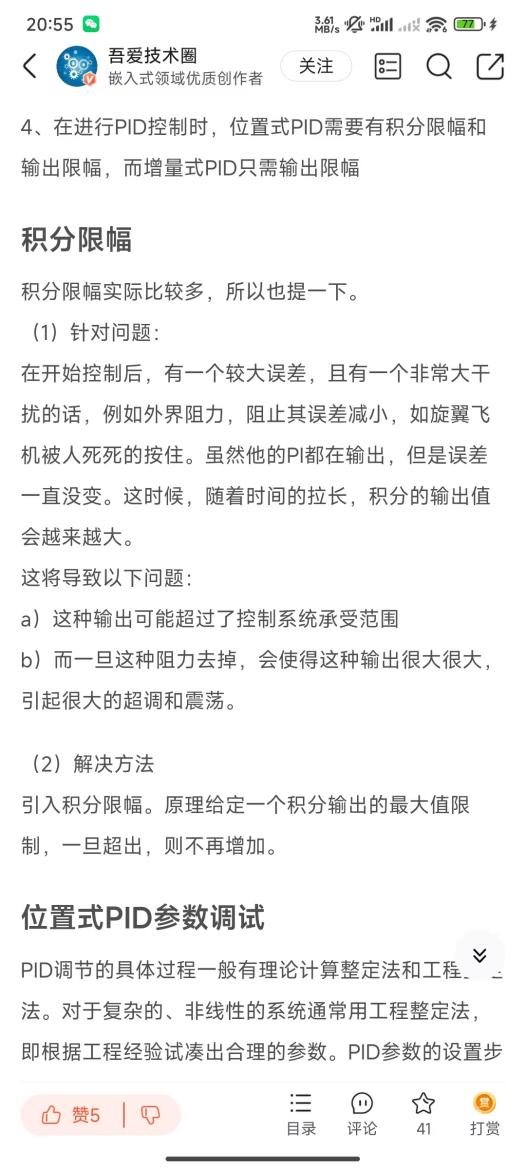
到达设定值过慢。是由于kd过大导致，可以减小kd；

**Pidcontrol中代码模板：**

****

在pidcontrol中

先设置了ki，然后实时的取值误差error，当误差大于一个一定的值时候，进行pid调控。先判断误差是不是在一个可接受范围内，在的话就积累误差，限制这个范围的原因是：



如果误差已经非常大了，就限额不增加。

之后根据这个积分在加上现在的误差后的值来控制范围。

然后计算误差的导数，变化斜率。然后计算应该输出的speed

根据speed是否超过上下限控制speed范围。还加了一个最小的输出单位确保能够克服固有的误差。

**Pidcontrol中代码问题：**

1722607212880

这样的话当误差达到某个值的时候就跳出了控制。这样失去了pid震荡回调的优势。因此可以考虑多判断几次，直到误差多次到达限制值的时候才结束控制。

if (fabs(error) > params->errorThreshold&&（params->cnt<0）)

设置一个params的cnt变量来控制误差达到目标值的次数。

或者以下思路：

加入死区，把小于误差范围设定为死区，如果这个误差在某时又出了死区，我再继续进行程序。  
