****

**“水下机器人”**

**总结报告**

院 系：船舶与机电工程学院

队 伍：菜虚鲲

队 长：员来

队 员：周云雷

许明乐

指导老师：白兴兰 张兆德

2019年9月

**目录**

[**一：目的及要求** **3**](#_Toc31754_WPSOffice_Level1)

[**二：功能及原理** **3**](#_Toc16664_WPSOffice_Level1)

[**三：主要设备仪器** **4**](#_Toc29586_WPSOffice_Level1)

[**四：设计思路** **4**](#_Toc27285_WPSOffice_Level1)

[**4.1电机安装方向：** **5**](#_Toc16664_WPSOffice_Level2)

[**4.2设计思想：** **6**](#_Toc29586_WPSOffice_Level2)

[**五：具体问题 8**](#_Toc27285_WPSOffice_Level1)

[**六：问题分析 8**](#_Toc27285_WPSOffice_Level1)

[**6.1问题一分析 8**](#_Toc27285_WPSOffice_Level2)

[**6.2 问题二分析 9**](#_Toc22819_WPSOffice_Level2)

[**七：解决措施 10**](#_Toc27285_WPSOffice_Level1)

[**八：视觉部分 17**](#_Toc27285_WPSOffice_Level1)

[**8.1灯带检测及巡线 18**](#_Toc27285_WPSOffice_Level2)

[**8.2寻盘** **19**](#_Toc22819_WPSOffice_Level2)

[**8.3字母识别 21**](#_Toc21988_WPSOffice_Level2)

**[九：对于比赛的一些建议 21](#_Toc27285_WPSOffice_Level1)**

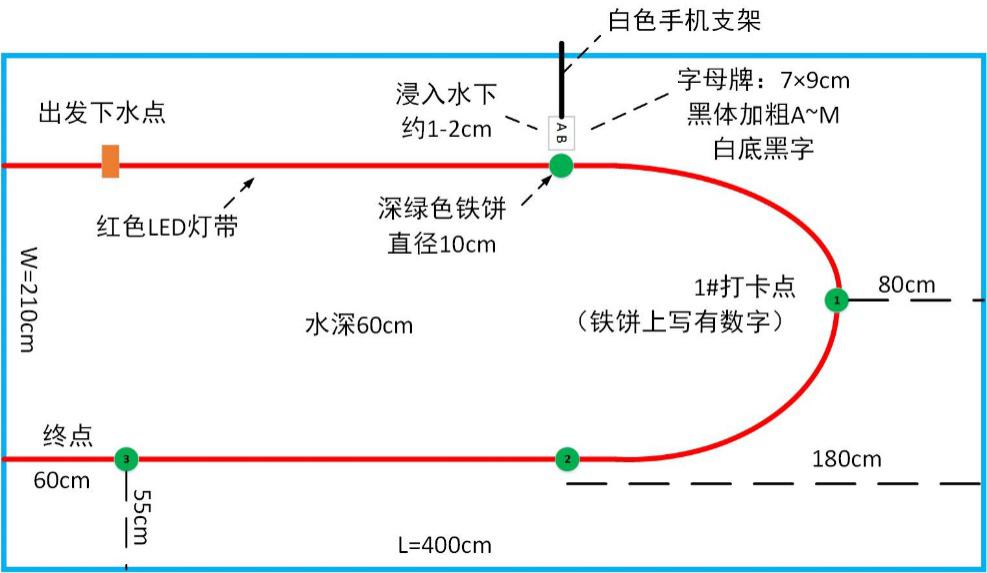
[**十：心得总结 22**](#_Toc27285_WPSOffice_Level1)

**一：目的及要求**

水下机器人是通过STM32和树莓派实现水下探索，相关要求如下：

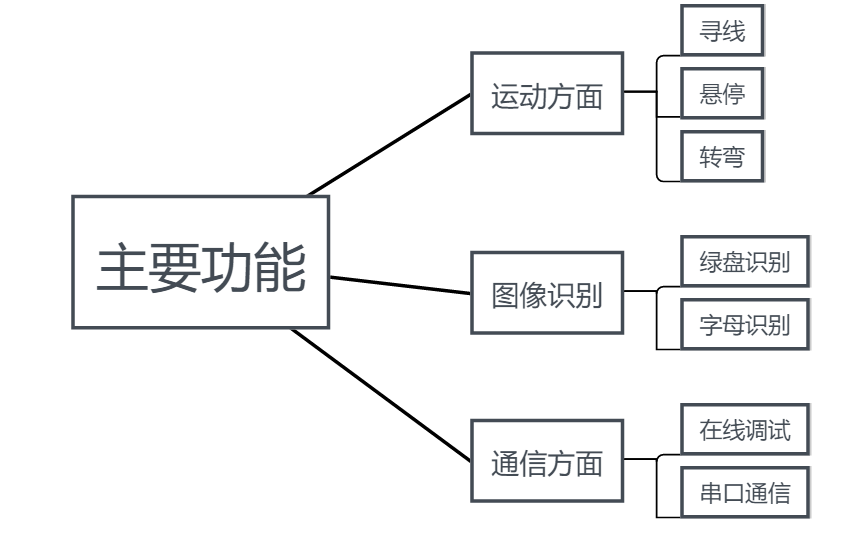
1. 能够通过灯带为引导使得水下机器人完成运动。
2. 依次通过打卡点并完成闪烁打卡和照片拍摄（保证数字清晰）。
3. 按顺序实现字母识别。
4. 在实现上述要求的同时尽快所有工作。

下图是比赛路径图：



**二：功能及原理**

根据水下机器人调试的基本目的和要求进行代码的编写和调试，因为基础代码是提供的，因此主要任务是使水下机器人的各种功能进行实现，主要侧重在以下几个方面：



其中在线调试是利用Lora模块实现STM32与PC的通信，串口通信是利用串口实现树莓派与STM32的通信。图像识别之后完成相关运动和信息传递。

**三：主要设备仪器**

16V电源、摄像头、STM32、树莓派、电机、各种感应模块。

**四：设计思路**

**4.1电机安装方向：**

使用了北京博雅工道有限公司设计的机器人外壳结构。因为我们以前玩过飞控的四旋翼无人机，有一定的经验，一开始考虑到这种方式可以实现全运动通道的控制，控制较简单，而且此次比赛的赛道弯道巡线要求较高，试想通过四旋翼式结构前后可以俯仰前进，可以控制左右旋翼的高低更好的通过弯道。

## 

我们试验了两种四旋翼结构，分别是电机往上和往下，对此我们做了大量调式。

我们发现电机往上时，由于机器离水面太近，效率并不是很高，难以实现定深；电机往下时，我们在机器上加了足够的配重是机器重力大于浮力，然而是用对桨差速实现转弯，必须是转速增大，由于电机具有过流保护，转速不可过快，依然无法实现，而且由于其距水底太近，导致水流直接将灯带吹歪，破坏了场地。

所以我们采用两平两立的传统方式，而两平两立的传统方式比起四旋翼结构能够更好行进的同时，减少环境的破坏性。下图所示的为两平两立结构的水下机器人行进时的拍摄图。



**4.2设计思想：**

STM32的开发和调试采用Keilu5集成开发环境，在调试程序时需要使用调试电缆将调试用上位机和水下飞行器连接到一起。

STM32主控板主要硬件组成：

主控芯片：STM32F103VCT6

压传：HP206F

温湿度传感器：SHT35

惯导：JY901

LORA数传：E32-170T30D

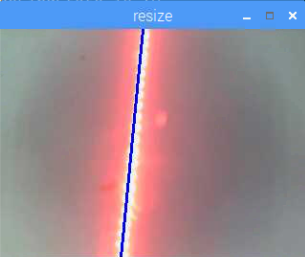
电流传感器：INA282

**由于机器自带例程搭载了操作系统，而单片机性能过于孱弱，机器识别运算速度 慢了很多，所以我们根据底层代码编写了裸机程序，大大简化了程序，使得STM32计算压力大大减轻，另外加入了计时器功能，实现各种运动控制。**

树莓派的主要功能是获取上下两个摄像头的图像，并进行实时处理，得到导航基线的位置，通过串口发送给STM32主控板实现自主巡线行使；还可以识别打卡标记和字母标记。在完成拍摄功能之后，同样还能通过串口发送给STM32主控板，再通过LORA数传发送给上位机，这样就可以在上位机中实时看到水下飞行器的打卡标记和字母标记的识别结果。

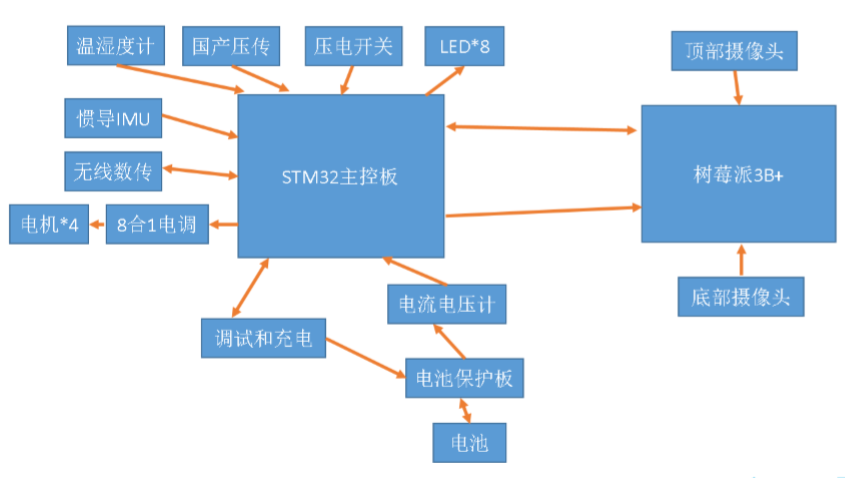


下图是调试时上下摄像头的拍摄截图，前者是下摄像头寻线图，后者是上摄像头字母拍摄图。

STM32主控板的主要功能包括根据压力开关状态来控制整机的供电；通过串口接收导航基线检测数据，并生成相应的电机控制指令，使水下飞行器沿着导航基线行驶。

同时获取搭载的多种传感器数据，实现飞行器的稳态控制，并将电量、温湿度、压传等数据通过LORA数传发送给上位机进行显示。



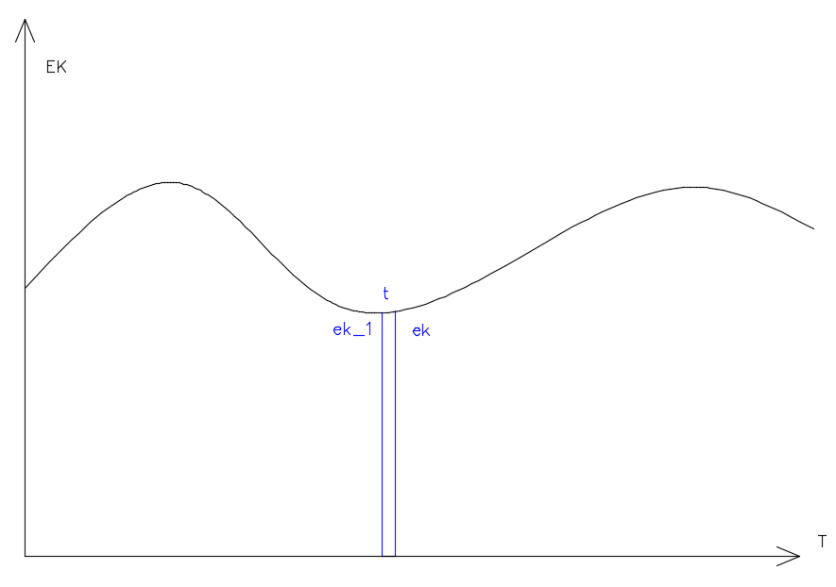
**五：具体问题**

1.无论怎么调pid，定深，俯仰，和循迹都调不好，控制不够硬，或者非常震荡。

2.转弯绕大弯，无法拍摄到1#打卡点，或者根本转不了弯。

**六：问题分析**

**6.1问题一分析**

针对问题1，很简单，就是控制周期太长，而且控制周期波动太大，控制周期太长延时就会太长，响应就慢，其次pid控制是个很讲究频率的东西，我相信大家有很多人不太懂pid，或者说就懂如何调pid，其实pid是个非常简单而且基础的控制方式，首先pid中有一个微分环节，因为我们用的是单片机，所以我们用的是数字pid，而数字pid公式里的微分环节是这么写的 kd\*(ek-ek\_1) 其中kd是微分系数，ek是本次误差，而ek\_1是上次误差，然后我们再来复习一下微分，微分就是求斜率，也就是求变化率，如果有一条EK关于时间的曲线，我们想要求某一时刻的变化率，那就是这一时刻的值（例如为ek）,减去在距离在当前时刻前一段时间（例如为t）时的值（例如为ek\_1），再除上这段时间差（即t）就是变化率（即（ek-ek\_1）/t），其中t如果越趋近于0，那这个变化率就越接近真实值，而这个时间差（t）就是我们pid的控制周期，我们的数字pid的微分环节真正应该写作kd\*（ek-ek\_1）/t,但是因为我们的控制周期都是固定的，所以t其实就是个定值，就被整合到前面的kd里面，也就不写出来了，从这里相信大家也能看出，控制周期不能过大不然求出来的变化率不准确，其次控制周期要固定，不然你的微分环节必定有问题。

但是官方给的stm32例程里的pid控制周期不是固定的，而且非常长，我用在线仿真的方式粗测了一下大概是十几赫兹，这种频率对于平衡类控制来说有点偏慢了，自然调的就非常软，或者非常震荡,真正做法应该设定一个定时器，将pid控制放在定时器中断中执行。

总结一下原因，**机器控制周期过长，且控制周期不固定。**

**6.2问题二分析**

针对问题2，总结起来就一句话，缺少前瞻，什么意思呢，就是说摄像头要看的更远一点，为什么呢，首先传感器（即树莓派）处理循迹线需要时间，然后处理后的循迹线数据发送给stm32也需要时间，最后stm32收到数据经过计算再执行也需要时间，也就是说stm32寻的线其实是一段时间之前的线了，之间存在了一定的延时，等到stm32真正开始控制转弯的时候，机器其实早就已经跑过弯了，那么机器会绕大弯或者转不了弯也是很正常的事情，我在举个例子吧，人是低头走的快呢，还是抬头走的快呢，不用想肯定是抬头走的快吧，人低头走的话，遇到一个急弯，必定是反应不过来的，这是毋庸置疑的，所以如果机器的摄像头看的更远一点的话，处理远一点的循迹线，那等到stm32执行的时候，机器也就刚刚走到处理的那段线，那么中间的延时也就被消除了，或者如果摄像头看的更远一点的话甚至能切内弯走。

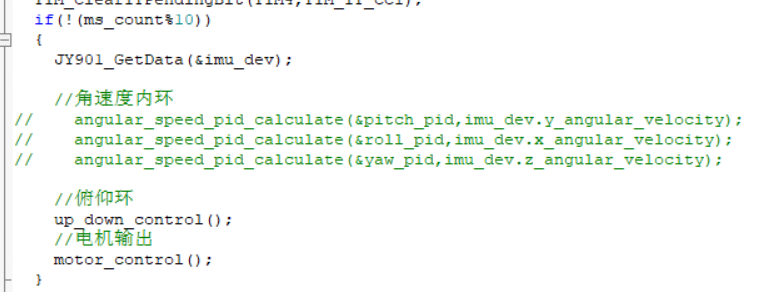
当然这只是其中一个问题，还是上面那个例子，人如果低头走的话，必定是走不快的，但是只要慢慢走，无论是多急的弯，都能转的过去，但是调试过这台机器的人一定知道，就算这台机器调的多慢，它也不一定能刚刚好踩着这条线过弯，那么根本问题是什么呢，那就是横向阻力不够，如果对象是车的话，就算这辆小车的前瞻很小，这辆小车也必定能完美过弯，因为地面会给轮胎一个很大的横向摩擦，但是对象是这台机器就不行，机器本身惯性就很大，而横向阻力又是水提供的，所以必定会飘出去，放在车子上讲，那就是它打滑了，他不是转过去的，他是飘过去的。除非机器能自己提供一个很大的横向阻力，或者前瞻再远一点，否则是很难转过去的。

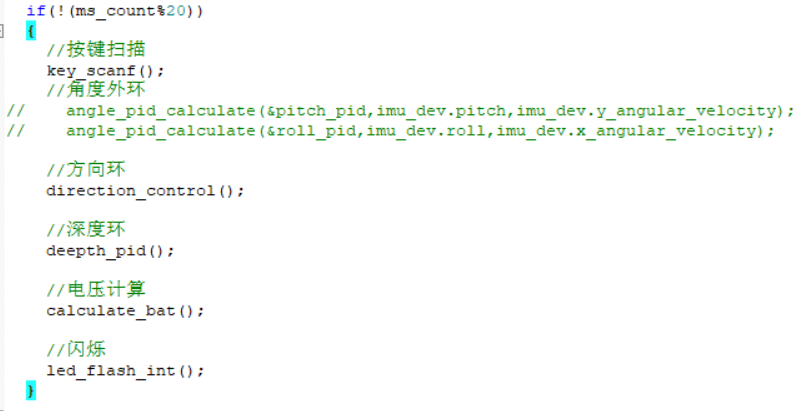
我们来总结一下原因吧，1.机器摄像头是垂直往下安装的，缺少前瞻，2.机器本身惯性很大，水无法提供足够的横向阻力。

**七：解决措施**

1.

因为例程里用的是操作系统，这个操作系统对于stm32f1来说运算有点勉强了，所以我们直接写了个裸机程序，这样就大大加快了运行速度，然后我们又设定了一个1ms的定时器，在中断中又划分了几个时间带来进行pid控制。





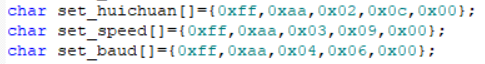
我们的俯仰环频率为100Hz,深度环频率为50Hz,方向环频率为50Hz不过方向环的频率是取决于传感器（即树莓派）的传输频率的，这里的50Hz并没有什么意义。

值得注意的是机器中的惯导模块默认输出频率非常低为10Hz,如果你传感器反馈的频率非常低，那么你就算在怎么提高pid的控制频率，你控制的响应也不会变快，所以我们按照jy901的数据手册对其进行设置，提高传输波特率为115200，减小需要传输的数据，只要欧拉角和三轴角速度，提高数据输出频率为100Hz，

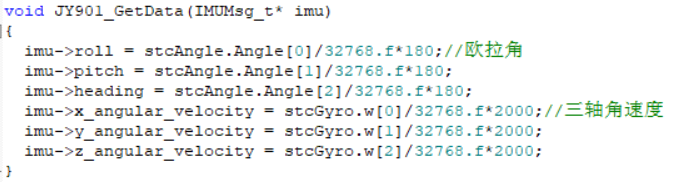
设置波特率

设置回传数据和输出频率

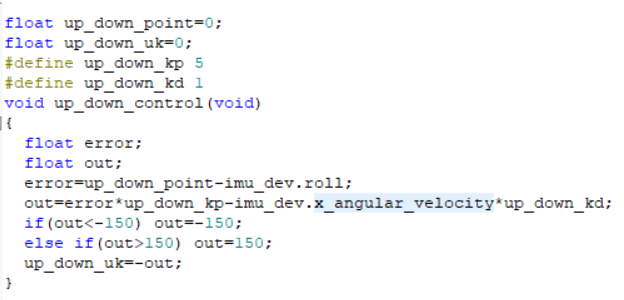
以下是查阅数据手册得到的命令语句：



另外回传的数据也要按照数据手册的公式进行转化，



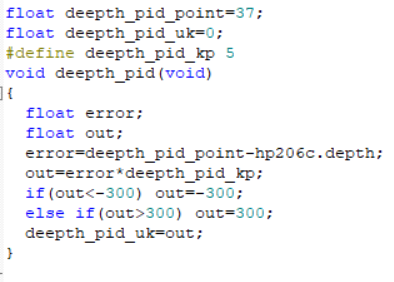
以下是俯仰控制pid程序：



俯仰控制我们用了比例微分控制，没加积分环节是因为不需要消除静差，这个机器的平衡并不是我们所熟知的倒立摆平衡，他放在水中，本身就会平衡，不会倒下去，所以只要比例微分控制就行了，加积分反而容易引起震荡，微分环节里我们并没有使用（ek-ek\_1）/t那个式子，而是直接用了惯导模块输出的x轴角速度，原因很简单，俯仰控制的就是俯仰角，而俯仰角的微分就是x轴角速度，虽然pid的控制周期t是0.01s，已经非常接近0了，但是论精确肯定是没有惯导直接测出来的准，不过用角速度替代微分项的时候一定要注意极性。

定深控制的话我们也做了一些优化，把软件i2c通信改成硬件i2c通信，加快通信速度（也可以用硬件spi通信，电路有连着，速度更快）。但我们发现定深响应还是很慢，主要是转换时间长要25ms，然后还加了滑动滤波，其实滑动滤波可以去掉，但是我们定深求的并不是快而是稳，所以这些都并不必要紧。

以下是定深控制pid程序：

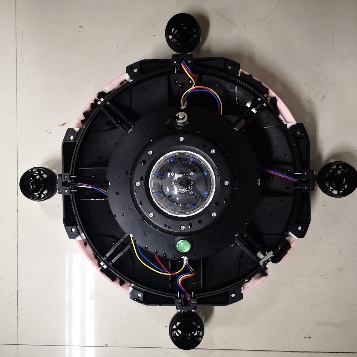


定深控制我们只用了比例控制，微分项的效果不大，所以没加，我们定深主要是稳，不求精确，所以没加积分项，同时将p值设的很小，这样的定深控制就非常稳，几乎不会波动，但p值设小会有一点就是静差特别大，所以我们又将目标深度设的更深一点，原本目标深度是20cm，而设定目标是37cm。

还有一点要说一下，大家一直抱怨电机死区太大，导致控制的不够好，但实际上无刷电调一般都有这种死区，这是无刷电机的特性，有一个怠速区，这台机器的真正缺陷就是电机性能过剩，只要一点点转速就已经达到了控制要求，可能大家会觉得奇怪为什么其他的无刷电调用在飞行器上就这么稳，很简单，一般的四轴飞行器光起飞就需要提供一定的占空比让它有个转速，然后在对它进行平衡控制的话，自然就不会进入电调的死区范围，所以我们定深也用了这个思想，稍微减轻了机器的重量，让它一开始就能漂浮在水面上，这也是为什么我们不盖盖子的理由，这样机器一开始就需要一定的转速才能沉进水中，之后在进行俯仰控制，就不会进入电调的死区范围，俯仰就控制的很稳，但重量也不能减的太轻，不然定深时水花太大，影响拍照。

*2.*

方向控制我们一开始也和大家一样，调的死去活来，但是当我们仔细研究发现了其中的原因后，我们就想出了两套方案。

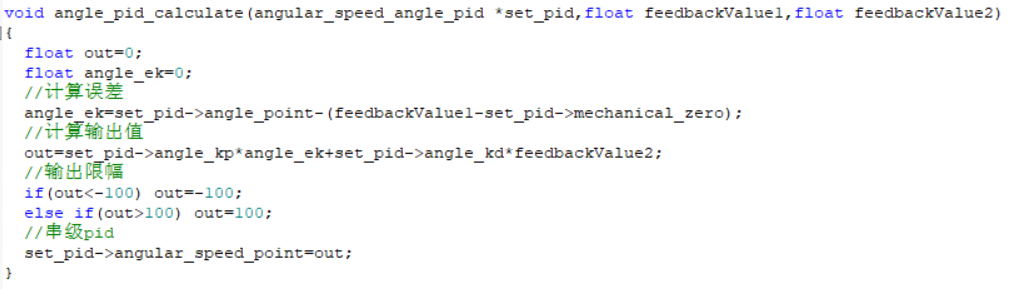
**方案一：**

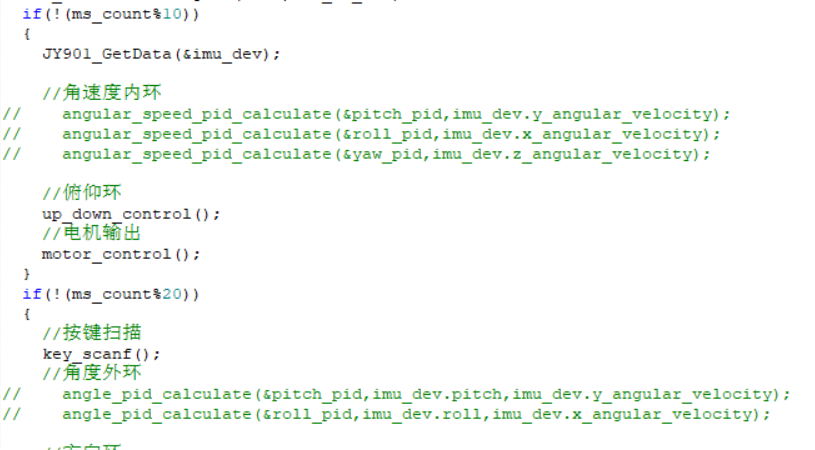
采用四旋翼结构，这也是个人最倾向的方案，前文已经提过了，由于水无法提供足够的横向阻力，所以即使速度很慢，仍然会产生横漂现象导致转弯失败，而四旋翼本身就可以全向运动，他自己就可以提供横向阻力，防止横漂，具体方法就是树莓派搜线得到两个参数，循迹线偏离中点的距离，和循迹线偏离前进方向的角度，他们分别与四旋翼横滚角和z轴角速度构建距离环和角度换，就能做到完美寻线，方向控制的程序没有，因为最后没用这种方案，其实程序也很简单，就两个单比例控制，以下是四旋翼平衡控制的程序：

**这是角速度内环，**



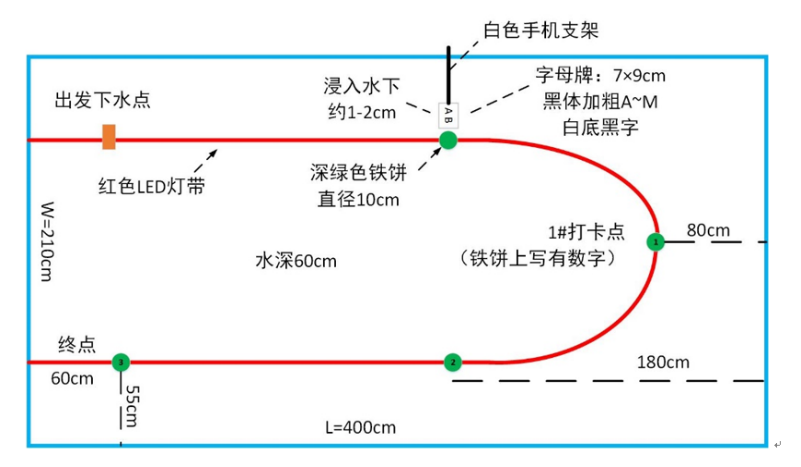
**这是角度外环，**



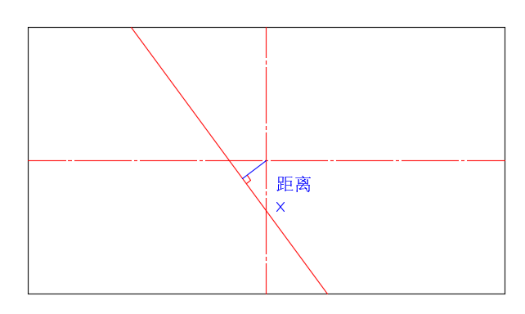
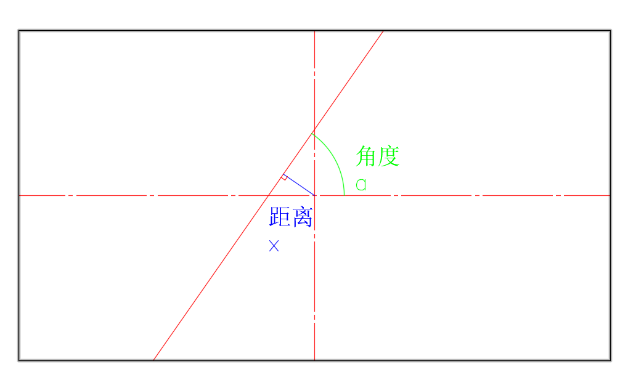


内环角速度环频率为100Hz，外环角度环频率为50Hz。其中航偏角不需要控制，只需要控制内环z轴角速度环。测试的发现机器控制非常稳，当然前提是机器一定要重，必须要让他有一定转速才能浮起来，原因前文已提，我们是又加了四个配重块，机器直接就会沉到底，寻线的时候做到几乎始终贴着线走的地步，但是我们后来放弃了这个方案，因为这种结构转向的时候会比较慢，道理大家应该都懂，阻力太大了，所以他转向的时候，电机的转速就要很高，但是设计这台机器的人考虑到电机转速太高可能会坏，所以加了个保护的电路，这就导致我们每次寻线刚转到一半弯的时候，电机就突然停转了，一开始不知道原因的时候我还以为没电了，后来发现就算充满电也会出现这种情况，查了资料以后发现是这个原因，气的都快吐血了，只好又换回原来的结构。

**方案二：**

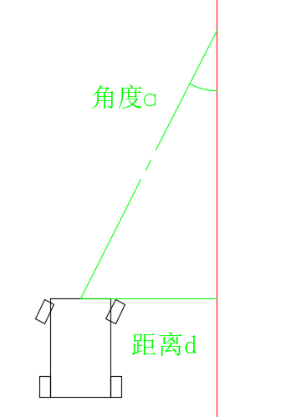
想办法提前转弯，提高前瞻肯定是没戏的，没人敢拆那台机子，并且还敢把摄像头往前扳一点的，那么只能从赛道入手，

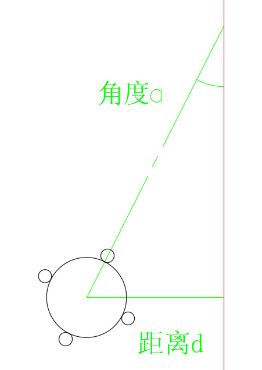
从赛道图片可以发现在大弯得进弯口和出弯口各有一个打开点，这是个很有用的信息，我们可以在进弯的时候改变方向环的目标值，使得它靠一边循迹，然后在出弯的时候改回来就行，但难就难在，反馈的数据有两个，一个是偏离中点的距离，另一个是偏离循迹线的角度，他们跟循迹有什么关系，这个方向环究竟怎么构建，我们一开始的想法就是直接将偏离中点的距离作为反馈的输入，但是当我调试的时候发现，机器简直晃到爆炸，我当时觉得非常的不可思议，后来仔细观察回传的图片终于发现了原因，



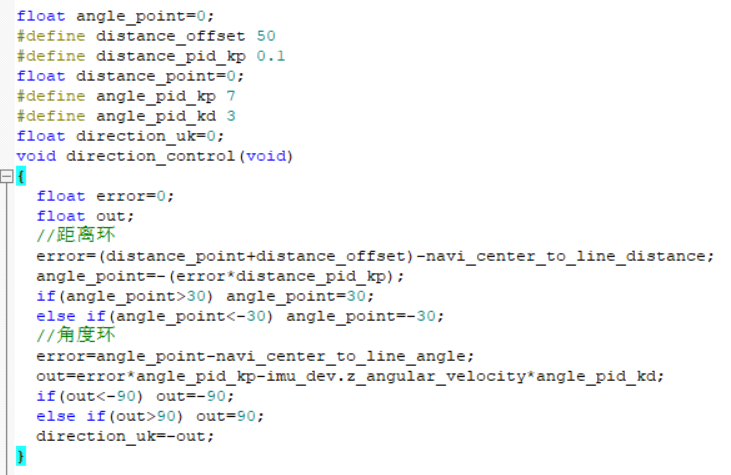
树莓派回传的距离是拟合的中线距离中点的垂直距离，也就是说如果机器没有在前进，那么机器就算再怎么旋转，他的距离都是始终不变的，换句话说这种方向环的构建本身就是错误的，中间还缺少了一个环节进行控制，那就是角度环，可能大家很难理解，其实这是仿照前轮转向的四轮车循迹的控制方式，

我们下面简单的分析，首先前轮转向四轮车的循迹。



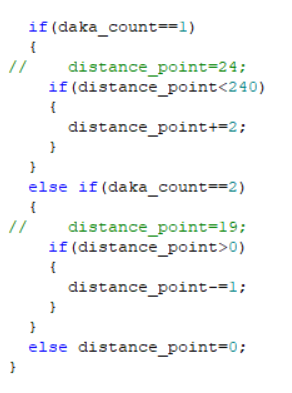
四轮车转向的话当它偏离中线一定的距离时，前轮只要打一个角度，那么只要小车一直往前行驶，它就必定能走到中线上，如果我们将前轮的打脚作为pid的输出，而偏离的距离作为pid的反馈，那么他只需要进行比例微分控制，就能做到完美寻线，

而我们的机器循迹也是同样的道理，当机器偏离中线一定的角度时，只要让机器与中线呈一个角度，那么机器只要一直往前走，它也能走到中线上，而我们之前的建立的方向环错在哪呢，那就是我们之前方向环的输出直接就作为电机的输出，而不是控制机器与中线的角度，而电机的输出其实控制的是机器的角速度，如果需要控制角度，那就需要额外一个角度环来进行串级pid控制，这样我们的方向环就可以构建了，以下是方向环控制程序。

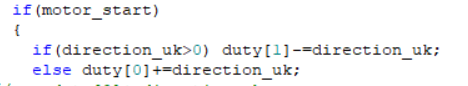


角度环用了比例微分控制，其中微分项还是用了惯导模块获取的角速度代替，因为树莓派数据回传的频率很低，大概将近20Hz，这样用数字pid的微分效果很差，而角度环控制的其实就是机器的航偏角，航偏角的微分就是z轴角速度，所以用角速度代替，效果显著，角度环的p值取到了7也没有震荡，回稳非常快，积分项由于很容易震荡，所以并没有用，当然也就会有静差，所以我们p值就尽量取大，使它产生过冲，而距离环只用了比例控制，因为数据回传频率低，微分的效果不大，而距离环本身就带有积分性质，所以积分项也不需要加，值得注意的一点是串级pid的内外环的控制频率一般要相差2到3倍以上，不然很容易引起共振，但是我们内环的频率取决于树莓派的回传频率也就是将近20Hz的样子，这样我们的外环频率就只能设到6~7Hz，这样的方向控制频率实在是太低了，响应很慢，所以我们还是将距离环和角度环放在一起控制了，取而代之的就是减小外环的p值，另外distance\_offset为寻线误差值，由于摄像头安装并不是很准确，并且在没有积分项的情况下左右电机输出的大小也不一致，所以每台机器并不是都踩着循迹线走，可能或多或少都偏一点，需要自行修正。

有了上面这个方向环，我们就可以回到一开始的那个方法那了，即遇到第一个打卡点后，改变距离环的目标值，使机器靠一边寻线，这样就能使机器踩着线过弯，然后在出弯识别到打卡点后，让目标值归0，使机器回到正常寻线就行。但是要注意改变距离环目标值的时候一定要平滑输出，不然数据突变会导致转弯有卡顿。这种方法类似于手动提高前瞻，但是只适用于固定赛道上，程序如下。



还有两点，第一点方向环最后输出的值不能采用左右电机一加一减的方式控制，前文提过，机器本身就有横漂现象，而且电机的输出不是呈线性的，如果采用一加一减控制，会加剧机器的横漂现象，所以我们采用一不变一减的方式控制，可能大家会觉得这样转弯岂不是速度很慢，但事实并不是这样，机器本身并没有速度环，是开环控制，给定一点速度后，由于阻力很小，全程其实都是加速行驶，再实际转弯的时候一不变一减速度也不会损失，顶多是止住了加速的趋势而已，程序如下。



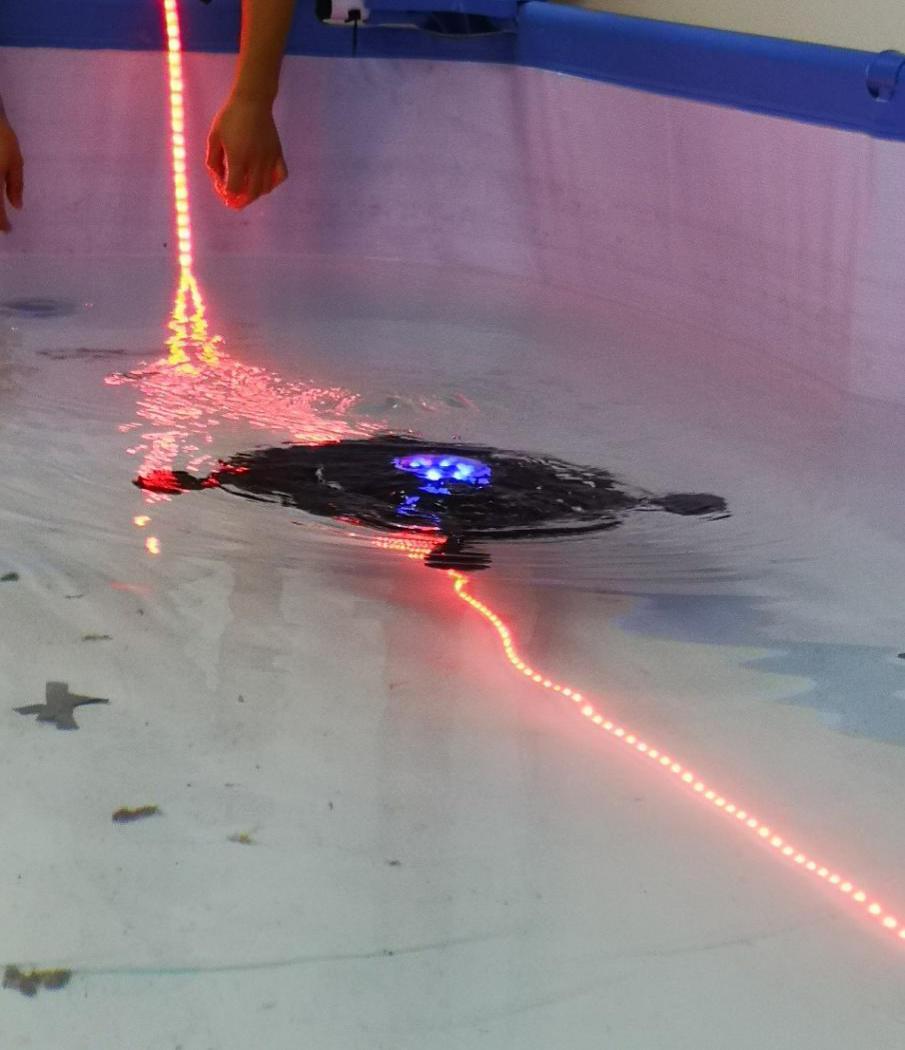
另一点是对树莓派程序的优化，树莓派的程序我都是直接用例程的，也就改改参数，调用调用那几个已经写好的算法，也没有什么自己的东西，大方面的就不讲了，稍微提几句好了，大家完全可以不需要执着于那些高深的算法，例如那个圆盘的检测，因为比赛场地的池子里不会有其他杂物，所以完全可以不需要霍夫梯度圆检测，只要判断到物体就可以认为是圆盘，这样可以大大提高树莓派回传速度，这样入弯的时候更容易提前，同样的，循迹灯也可以不用hsv识别，循迹灯本身是个发光源，所以它的rgb成分都非常高，而且池子里也没有其他发光体，不会误判，所有只要对这些成分进行阈值分割，就不容易丢线。

**八：视觉部分**

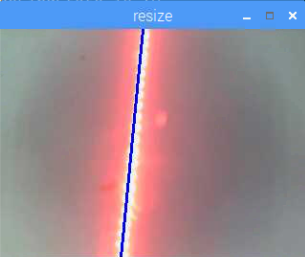
因为我们字母识别的训练集没有时间做，神经训练也没有，在实际比赛中也有诸多困难：字母牌拍不正，识别错误，误识别等，所以字母识别部分并不是很好，这里就简单介绍。

**4.1 灯带检测及巡线**

巡线是通过调用树莓派下方摄像头拍摄灯带照片，通过树莓派的处理得到拟合线循迹，结合偏航角控制算法完美实现灯带巡线功能。

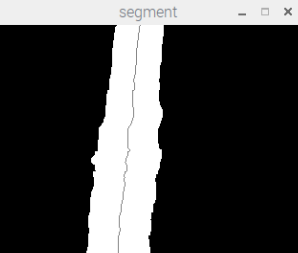


下方摄像头拍照反馈到电脑端的截图：



HSV分布分析：程序提取光带中心附近的HSV值，然后根据实际值来修改阈值范围。

通过腐蚀，闭运算之后得到连通域，连通域用最小二乘进行直线拟合。得到下图的处理图像。

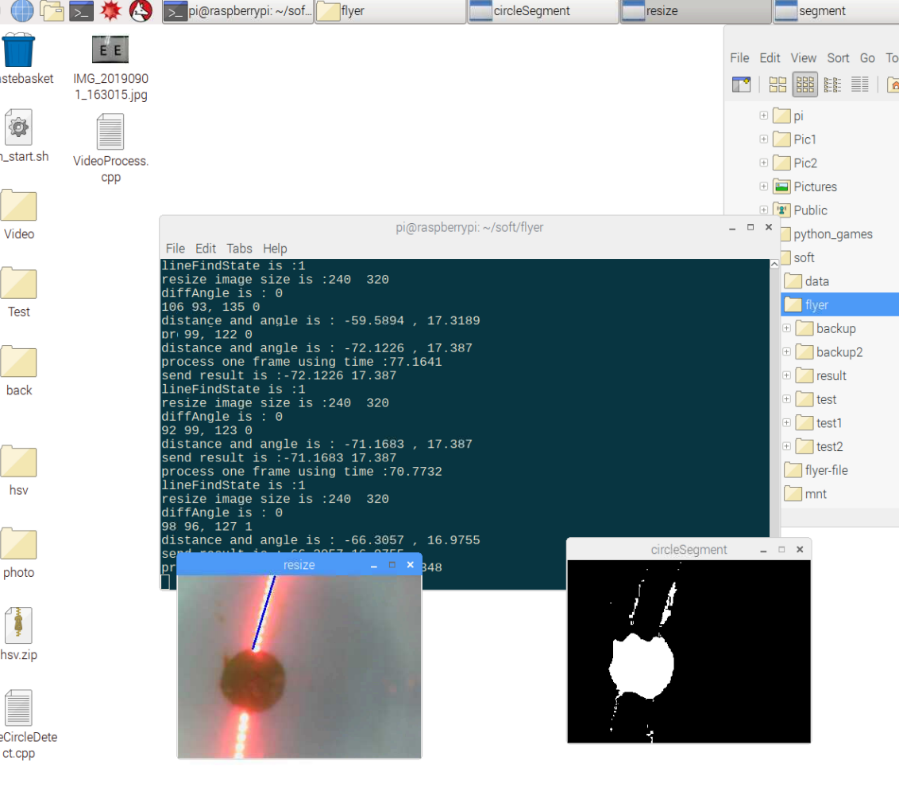


**4.2寻盘**

开始我们使用检测圆的方法来识别，但是具体调试时发现圆盘在灯带上，二值化以后圆在与灯带接触的地方由于红色太亮影响导致部分缺失，霍夫圆检测不能得到完整的圆，所以识别准确率很不稳定。

于是在考虑到各种情况之后，我们最后采用颜色识别的方法。

为了能够识别由于光线等环境因素造成的RGB提取区域不接近圆，而更改类圆参数又会导致误识别的概率加大，我们增加了像素占比的条件限制，这样的话使得识别的准确率增加了很多。



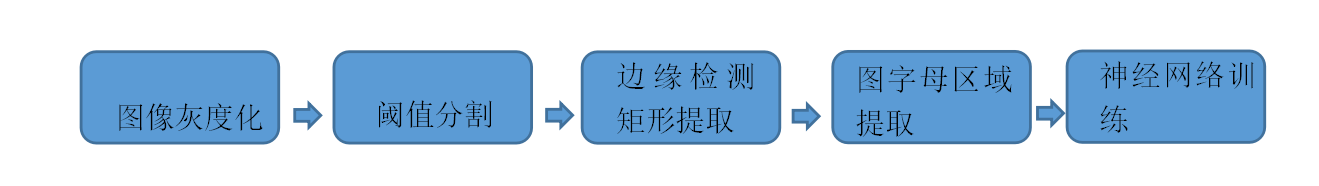
我们的方法相比于检测出绿色的圆形来说，虽然后者能够完整地取出整个绿色圆。但是就用途而言，我们只需要给单片机发送识别到绿色圆的信号，而不用整个绿色圆边界的完整信息。因此我们这种只用图像二值化并计算连通域中心的方法。能够非常快速并且准确地识别出绿饼，准确率高。

下图是完成寻盘之后的定点拍摄图，能够拍摄到相对清晰的数字“1”。

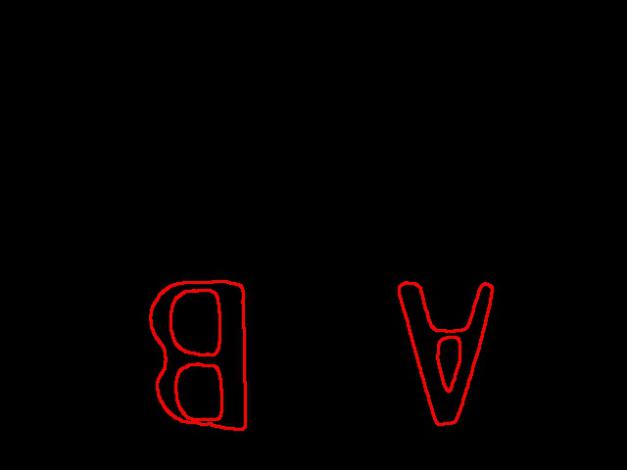
****

**4.3字母识别**

使用opencv进行神经网络训练，建立字母训练集，实现字母识别。



下图是字母识别的字母示例图展示：

** **

**九：对于比赛的一些建议**

1.螺旋桨需要做的更坚固，机器的螺旋桨虽然在水中足够坚固不会坏，但是一旦拿出水面，稍微空转的快一点就会坏。

2.电机需要更换为成对的有正牙和反牙螺纹的电机，虽然这台机子配备了正桨反桨，但是电机却都是反牙螺纹的电机，这样肯定会有一边电机旋转的时候子弹头容易松。

3.去掉电路中的电机保护电路，或者把电机保护的阈值提高，我其实非常希望用四旋翼结构，但无奈硬件限制了这点，虽然我知道这个设计是出于好意，但还是感觉有点遗憾。

4.测深可以不需要通过压传去检测，如果水深不深的话可以换成超声波测距或者激光测距，测量精度，检测频率都很高，而且价格也会便宜点。

5.如果赛道的图像处理不是很复杂的话可以考虑只用一个性能更高的单片机来处理，去掉树莓派，或者将树莓派更换成openmv。

6.循迹的摄像头可以稍微看的更远一点，或者干脆设计成角度可控的。

7.四旋翼结构很好，但是转向偏难，螺旋桨可能需要更换成大桨，电机更换成kv值更低的才行。

**十、心得总结**

这次比赛最大的收获就是加深了对pid的理解和掌握程度了，如果我们回过去再去看方向环控制的程序就会发现，其实算法很简单，就是距离乘上一个系数减去角度得到一个差值，然后将这个差值乘上另一个系数与角速度乘上一个系数相加后赋值给电机的占空比，但是算法虽然简单，但蕴含的思想却很深刻，虽然这个算法去凑有人可能凑的出来但是去调不一定能调的好，这也许就是pid控制的魅力所在，简单却精妙，不得不佩服前人的伟大。

当我们小组答辩完的时候记得有人说了一下不清楚串级pid，其实很简单，首先pid中有三个参量，目标值，反馈值，输出值，它的功能就是会帮我们让反馈值尽量接近目标值，例如方向环中的距离环，如果我们初始目标值为0，那么这个算法工作的时候，他就会让回传的距离接近0，而距离为0的时候就是机器刚好踩在线上，如果我们目标值为10，那么机器就会在循迹线右边，距离循迹线10个像素点的位置寻线，所以这就是我们为什么能够通过改变目标值来提前转弯的原因，说完目标值反馈值，接下来是输出值，输出值是计算以后的值，这个值可以直接输出给电机，例如我们的角度环，但也可以作为下一级pid的目标值，这样就成为了串级pid，其中输出值给下一级pid 的就是外环，另一个则是内环。

我们答辩结束的有同学问我为什么距离环是外环，角度环是内环，当时我没有讲清，我在这重新回答一下，虽然我不清楚这么讲正不正确，反馈控制是利用偏差消去偏差，而偏差就是离散数据的微分，我举个例子，不知道各位有没有听过伺服电机的经典三环控制，电流环，速度环，位置环，其中电流环是最内环，速度环其次，位置环为最外环，为什么是这样子分布呢，我简单分析一下，伺服电机的扭矩跟电流是呈线性关系的，所以我们控制电流的时候相当于控制的是扭矩，而如果电机的转动惯量不变的话，那么实际控制的就是角加速度，这里可以发现，我们改变电流就是改变加速度，改变加速度就会改变对应的速度，改变速度就会改变对应的位置，然后我们倒回去看，位置的微分就是速度，而速度的微分就是加速度，说到这不知道各位有没有明白一点串级pid的内外环，接下来我们在回到原来的方向环那里，看看为什么角度环是作为内环，首先我们要清楚一点，不能进行全向运动的机器都需要有一个前进的速度才能够通过改变方向来走回到线上，现在假设机器此时有一个前进的速度，如果我们改变与中线的角度，是不是相当于改变了回到中线时的速度，而且角度越大回到中线速度就会越快，如果各位想通了这点，应该就明白了为什么角度环作为内环的理由了吧。

最后再说一点，其实我们只要修改一下摄像头搜线的程序，让回传的数据等于某一行的x偏移值，那么只用单级pid也可以做到搜线，具体原因大家可以自己思考一下。

