



华南理工大学
South China University of Technology

华南理工大学十四届室内创意组

华工木灵队

队员

苏锴南

王贤枋

邱强清

蔡铖陇

王敏杰

指导老师

陈安

邓晓燕

前言

这是华南理工大学室外创意组华工木灵队的审核文字资料，为了准备本次的恩智浦智能大赛创意组，我们队伍经过了长时间的研究以及思考，设计出了自己的一套解决的方案，这份文档是对于这次参赛的我们队伍的车模的一些资料介绍。本次的车模使用了 nxp 的 MK66FN2M0VLQ18 芯片作为主控，并围绕其进行了用一系列的硬件以及软件的设计，完成了基本的任务方案。在设计过程之中我们遇到也解决了许许多多的困难，让我们从实践之中学习到了许多课本之外的创新内容。对于本次的大赛我们有信心能都完成任务，取得佳绩。

目录

前言	1
整体框架以及功能	1
一、 车模整体架构说明	2
1.1 整体架构以及方案选择	2
1.1.1 主控芯片的选型	3
1.1.2 控制周期的选择	3
1.1.3 系统时间的设置	3
1.1.4 通信部分	3
1.1.5 动作组控制	4
1.1.6 上位机	4
1.1.7 驱动方案	6
1.2 基本功能说明	8
1.2.1 图像处理以及定位	8
1.2.2. 路径规划	8
1.2.3. 多方向的麦克纳姆轮运动	8
1.2.4. 棋子拾取拜摆放以及障碍摆放的操作	8
二、 机械设计方案	9
2.1 机械设计	9
2.1.1. 麦克纳姆轮运动装置	10
三、 硬件设计方案	12
3.1 主控芯片及主控电路	12
3.2 隔离模块	14

3.3	电源设计.....	14
3.4	电磁驱动器方案设计	16
3.5	步进电机驱动.....	17
3.6	光流传感器.....	18
3.7	单片机显示控制台	19
四、	软件设计方案.....	21
4.1	速度控制.....	21
4.2	可编程视觉模块 i.MX RT1062	22
4.2.1	i.MX RT1062 模块特点.....	23
4.3	定位部分.....	23
4.3.1	AprilTag 三维定位技术.....	24
4.3.2	线检测获取车体旋转角度	25
4.3.3	低分辨率摄像头精确校位	27
4.4	障碍棋子识别方案.....	27
4.4.1	圆形棋子的识别定位	27
4.4.2	红色障碍的识别定位	28
五、	算法设计方案.....	29
5.1	关于八皇后问题的解决方案.....	29
5.1.1.	解决思路	29
5.1.2.	实现方式	30
5.1.3.	运算实例	30
5.2	决赛对抗方案.....	30
5.2.1.	决策算法分析	30
5.2.2.	上位机设计	32
六、	作品特色描述.....	34

6.1	机械部分.....	34
6.1.1.	自行设计低重心车体	34
6.1.2.	导轨式障碍放置结构	34
6.2	硬件部分.....	35
6.2.1.	可拓展的驱动电路方案	35
6.3	机器视觉部分	35
6.3.1.	光流传感器	35
6.3.2.	双机器视觉模块定位	36

整体框架以及功能

第十四届恩智浦智能车大赛的创意组比赛要求参赛队伍设计一辆车模能够通过搭载了恩智浦的主控芯片的 openmv 完成赛场（8x8 的方格棋盘地图）之中的线条、April tag 以及障碍棋子的检测与辨认，并在此基础之上完成几何物的搬运，完成指定的如八皇后以及步步为营的下棋任务。我们的车模可以通过自己的设计的机械结构以及电子和控制算法完成预赛和决赛两个阶段的八皇后排序任务和步步为营的对抗挑战任务。

在研究比赛规则后，在其基础上我们在车模平台上完成了以下的功能：

1. 棋盘的位置定位

车模可以通过 openmv 自带的 April tag 的识别获取在棋盘之中的大致位置，然后辅助使用总钻风摄像头，光流传感器和环境光传感器对棋盘之中的准确位置进行定位，方便进行棋子障碍的摆放。

2. 棋子与障碍物的识别和摆放功能

车模的 openmv 能够识别定位棋子和障碍所在位置，并将信息传输给 k66 主控芯片，最后芯片会通过控制电磁铁对于障碍进行吸放，达到抓取和摆放障碍的目的，且摆放障碍时能够通过灰度传感器识别边线，做到沿边摆放。

3. 路径规划以及运动

车模能够通过 openmv 获得的图像处理，利用 openmv 自身的计算出当前平台所在棋盘格上的位置；同时图像处理还可以识别出棋盘上的棋子和障碍分布，并且按照预设算法，规划出一条快速且符合规则的摆放路线。车模使用的麦克纳姆轮，可以实现多方向的运动，更加的灵活。

一、车模整体架构说明

1.1 整体架构以及方案选择

车模上方装的摄像头是搭载了恩智浦芯片的高速 OpenMV 可以进行机器视觉以及图像处理，april tag 的图标可以给我们一个小车当前的位置的信息以及角度，多个 April tag 可以帮助更好的矫正返回的信息，加强精准度。将图像处理获得的一定信息通过串口发送到主控的 k66 芯片上。K66 会通过 pid 算法算出合适的 pwm 的占空比，之后通过将四路 pwm 输入到四个 h 桥控制电机进行运动。棋子以及障碍的吸放使用的是电磁铁。障碍和棋子之间埋了一个铁片。棋子由于需要拾取以及放置，所以使用的是推杆，分布在车中间可以吸起棋子。障碍由于不用拾起所以直接放在一个步进电机控制的四向杆子上面，可以收放在车底盘下面使得挂载障碍的时候不超出 50x50 的距离。同时车上使用了另一个 k66，对于车底下的 16 个光电对管进行联络控制并向主控的 k66 传输信息，使得其可以精准的对准。光流传感器以及总钻风可以起到辅助定位的作用

电路的主控制器采用了一款飞思卡尔公司的“[K66P144M180SF5V2](#)”单片机。主控电路的主要工作是接收两个全局摄像头返回的平台坐标以及棋子与障碍的位置，还有微控制器发送下来的动作组命令，先通过摄像头发下来的平台坐标信息及光电式编码器检测轮子速度，通过 PWM 控制驱动电路，调整电机的功率，实现闭环控制平台的移动，位置接近定点位置后，进行精确定位，最后执行相应的动作组，比如捡起棋子，放下障碍等等，完成之后等待下一条指令。

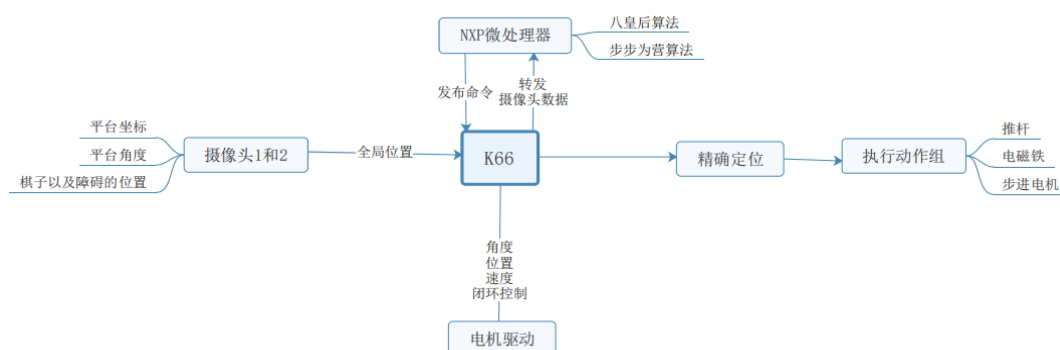


图 1-1 系统总体框架构图

1.1.1 主控芯片的选型

由下位机的软件架构可以知道，下位机需要与两个全局摄像头、微控制器通信，另外，在调试的过程中还需要与自己的上位机通信，所以需要至少 4 个串口。车体底盘的四个电机需要一个 ftm 模块，最好配有四路 ftm 正交解码，电磁铁、推杆和步进电机的驱动也需要 ftm 模块，所以需要两个以上的 ftm 模块。

上面是对于特殊接口的要求，实现其他功能的接口都不会太过苛刻。

综上，我们选择了飞思卡尔公司的“[K66P144M180SF5V2](#)”单片机，K66 具有五个串口，通信方面已经达到要求，而它只有四路 ftm 模块，除去两个 pwm 输出，只剩下两路 ftm，所以我们电机的闭环控制使用了两路正交解码、一路 lmptr 和一路 DMA 计数。

但是，DMA 计数存在一个缺点，有时候会读到一个很大的数，此时，在软件上做低通滤波就可以解决了。

1.1.2 控制周期的选择

通过测试运动周期为 20ms，15ms，10ms 和 5ms 的车轮运动状态，发现在 5ms 的时候电机可能由于电机编码器的缺陷，电机反而容易出现抖动。而 20ms 的控制周期过长，电机反应速度不够快，最后，我们选择了 15ms 的控制周期。

1.1.3 系统时间的设置

我们通信重传、步进电机、电磁铁的开关时间、系统指示灯等等都需要不断地查询时间，如果只是简单的使用 delay 函数，很容易就造成了控制时序乱套。于是，我们增加了系统时间，其他需要用到计时的部分只需要在内部设置一个静态变量计算时间差便可以实现计时了。

1.1.4 通信部分

车体在运动过程中需要与摄像头、微控制器长时间的短数据包通信，由于是短数据，按照惯常的做法即可，在串口中断中插入数据包的解析函数即可；另一方面，由于考虑到是长时间的通信，极有可能出现某一帧传输错误导致解析函数一直卡在接收的某个阶段，于是，在解析函数里面加上帧头帧尾时间差的检测，

当超过时间差阈值，便回退到帧头接收，经测试，这个方法可行。

下位机使用 DMA 串口传输，相比于使用普通串口传输，可以快速地传输大量数据，且基本不占用 CPU 的处理时间，这样才能发挥 CPU 应该有的性能。

1.1.5 动作组控制

在动作组的控制方面，由于有推杆上下、电磁铁开关、步进电机等等的控制，命令繁多，而且很多都需要使用到计时方案。所以，我们使用了类似于总线控制的思想，把全部的动作组都放在一个线程里面循环，如果需要使用到哪个动作组，只需要设定动作命令为相应命令，该动作将会自动接收并作出相应。当动作完成后，会把总线放空，这时才可以进行下一个动作。这样，可以很方便的作出动作，并且时序不会乱。

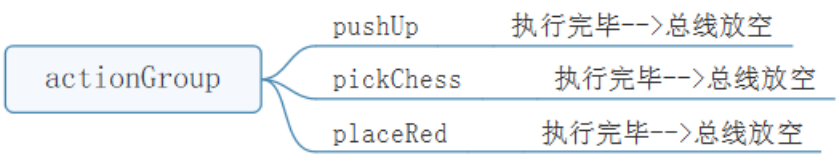


图 1-2 系统总线控制时序

1.1.6 上位机

我们通过上位机对车模运行时的速度，特殊路段的判断都进行了实时监控，通过上位机可以直观地反映出各类参数和阈值的准确与否，相对于“盲调”，这种调试方法可以更加合理客观方便。

上位机的界面使用 tabControl 控件进行控制，可以很方便地进行多个子界面的管理和切换。界面 1 是车体的状态显示、车体的数据可视化和字符串的接收。该界面在车体的摄像头调试和前期调试发挥了巨大的作用。

界面 3 是总钻风图像的读取、显示和保存，可以在上位机写图像处理函数，再下载进 K66，很方便直观。

我们的上位机还有一个比较有创新性的功能。由于我们上位机开发的波形显示界面一直不是很友好，而匿名上位机的功能更加丰富，不过使用匿名上位机，其中的协议不可更改，且往下位机发送数据时较为繁琐，需在程序中作较多修改。于是，我们取长补短，用虚拟串口软件，帮电脑虚拟了连接在一起的 COM1 和

如图界面 2 是参数整定界面，包括速度环、位置环和角度环 PID 参数的整定，速度、位置、运行角度的设定。在近期还加入了模糊表参数的整定，相对于“盲调”，这种调试方法可以更加合理客观方便。



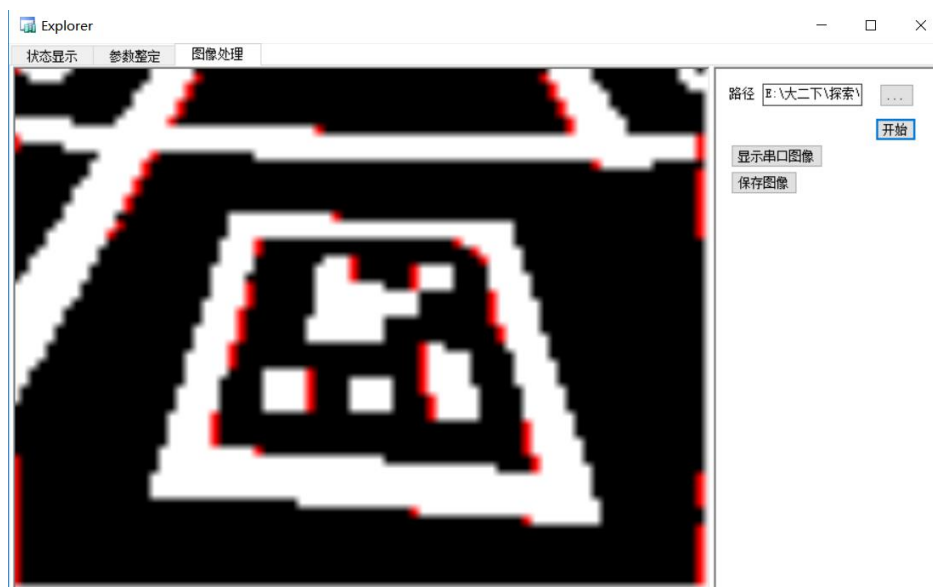


图 1-3 (C) 上位机界面

1.1.7 驱动方案

我们队伍选用 IRLR7843 型 MOSFET 来搭建 H 桥电路,通过查询芯片的数据手册可知道,IRLR7843 的门驱动电压需要达到 10V 左右,所以我们 MOS 驱动芯片选用 IR2103S,该驱动芯片可以驱动漏极电压高达 600V 的电压,IO 输出可以达到 12V,足够满足我们的 MOS 驱动了。C19 与 D6 构成自举电路。C3 去耦电容应尽量接近 IR2103S 的电源输入。这里 C19 需要特别注意,耐压要大于 30V。

考虑到我们有四个电机,一个推杆,多个电磁铁,如果把驱动电路集成在一块板子上,如果一个 H 桥坏了,那整个驱动都得拆下来重新调试,极其影响效率,于是,我们做成“插卡模式”,把一个个 H 桥做成一个个小板,这样更换驱动特别方便,也提高了调车的效率。

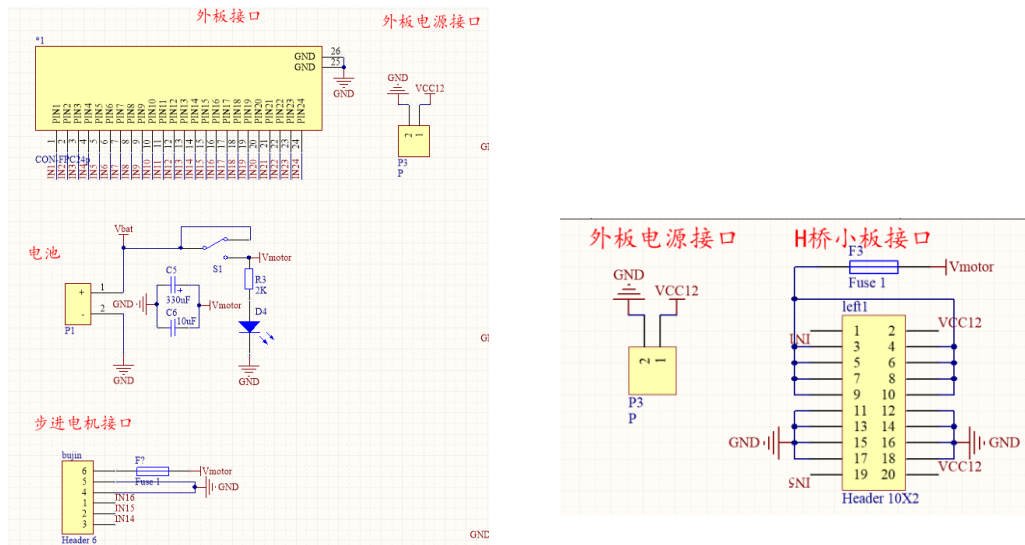


图 1-4 (A) 驱动板原理图

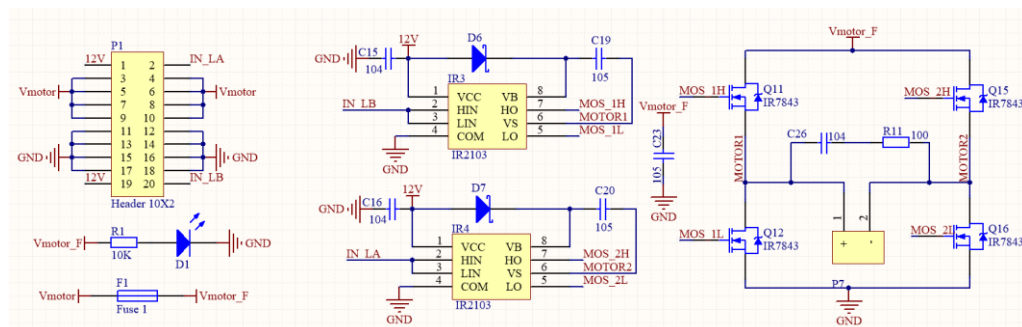


图 1-4 (B) 驱动板原理图

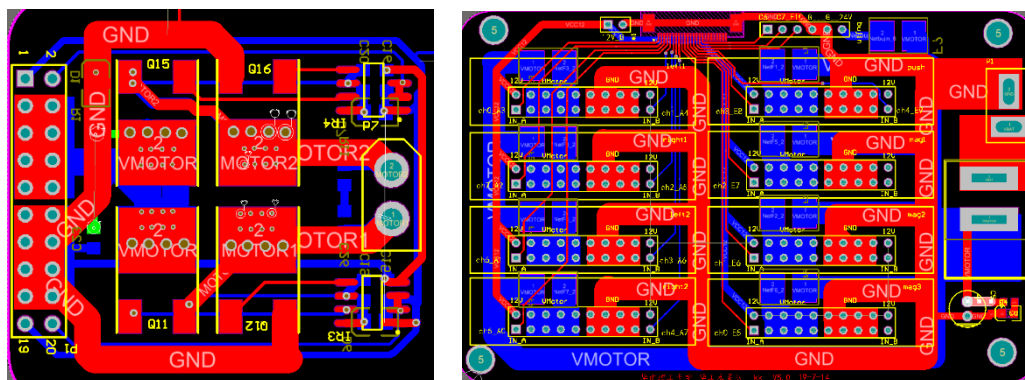


图 1-4 (A) 驱动板原理图

1.2 基本功能说明

1.2.1 图像处理以及定位

通过搭载了恩智浦芯片的高速进行机器视觉处理，可以获取感知棋盘 `april tag`，也可以感知棋盘之中线的位置以及棋子和障碍。从而获取当前车的位置和棋盘的状况，从而可以反馈信息给主控芯片。

1.2.2. 路径规划

搭载了算法功能，在从图像视觉处理之后可以了解全局状况，并且算出最优的解法，即移动步数最少，搬动棋子最少的路径，因此可以减少入场后所用的时间，对于八皇后问题和步步为营的解决有着莫大的帮助。

1.2.3. 多方向的麦克纳姆轮运动

主控芯片从摄像头之中获取的坐标角度，并且规划好路径之后，可以结合麦克纳姆轮的特性，做出向任意方向的平行移动以及顺时针与逆时针的旋转，并且其中的速度可以控制，以达到最短的移动距离，从而减少消耗的时间。

1.2.4. 棋子拾取并摆放以及障碍摆放的操作

棋子以及障碍的摆放均是靠电磁铁完成，棋子由中间的推杆控制上下，可以实现吸放，障碍由步进电机控制的齿条收放在车模的底部，平时由电磁铁吸附，当需要放置的时候可以从底部伸出，并且一次可以放置两枚障碍，减少了移动的时间。

二、 机械设计方案

2.1 机械设计

初始阶段，我们使用 solidworks 对车体进行建模，包括底盘、运动装置、摄像头支架、棋子拾取装置以及障碍放置装置。

底盘设计我们选用单层 4mm 厚度的板材，并使用工型轻质铝材作为支撑，保证底盘的强度，在承担一定负载的情况下不变形。底盘上有若干安装孔，用于固定螺丝，安装电路板及其它装置。中心留有圆孔，使拾取棋子的推杆通过。

运动装置选用四个带编码器的 24V 直流减速电机带动麦克纳姆轮，可以实现平面上的全向运动。麦克纳姆轮选用直径 152mm，配合直流减速电机，使我们的车能够以较快的速度到达指定位置完成任务。

摄像头支架最初使用工型轻质铝材搭建，但在实际测试中，由于支架较高，末端抖动较大，不利于视觉处理。后采取了三角支架，效果较为稳定。棋子拾取装置使用 24V 电动推杆。

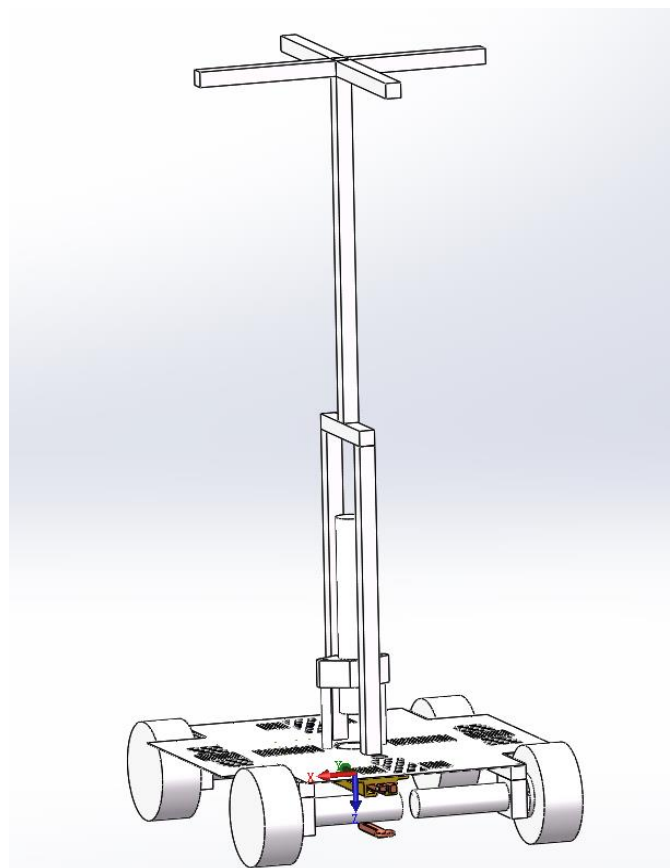


图 2-1 solidworks 车体建模

障碍放置装置我们使用步进电机带动齿轮齿条机构，车进场后展开齿条使得带着障碍的电磁铁到达预定位置。步进电机选用 42 步进电机，输出扭矩 0.28Nm，足够带动齿轮齿条机构。齿轮为 1 模 17 齿，保证尺寸较小的同时能使机构以较快的速度展开。



图 2-1 齿轮齿条传动示意图

2.1.1. 麦克纳姆轮运动装置

特点：通过对比全向轮，麦克纳姆轮的方案，选择了麦克纳姆轮的全向运动方案。根据麦克纳姆轮的全向运动原理，可以通过四个轮子不同转向产生的斜向合力的组合，实现车子不同的运动状态。

安装方案：采用“0-正方形”四个轮子位于正方形的四个顶点，平移和旋转都没有任何问题。

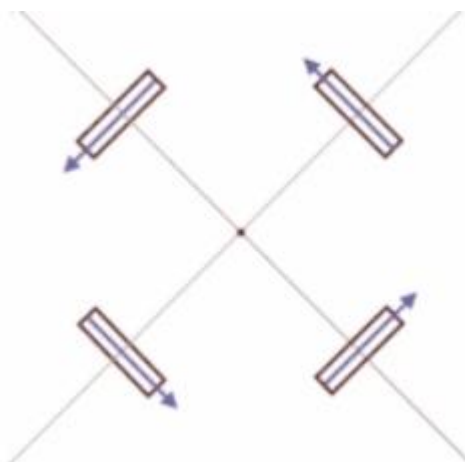


图 2-2 麦克纳姆轮的全向运动原理

通过上述几种运动不同比例的叠加，可以实现小车不同方位不同姿态的运动，从而实现小车的全向运动。

速度分解计算：采用较为简单的逆运动学计算方式进行运动分解，以得到各个轮子的转速。由于全向移动底盘是一个纯线性系统，而刚体运动又可以线性分解为三个分量。那么只需要计算出麦轮底盘在「沿 X 轴平移」、「沿 Y 轴平移」、

「绕几何中心自转」时，四个轮子的速度，就可以通过简单的加法，计算出这三种简单运动所合成的「平动+旋转」运动时所需要的四个轮子的转速。最快速由推动底盘观察现象就能得出。

通过观察可知，Y 轴的运动为四个轮子都同时正转或同时反转。X 轴的运动为车体前方的两个轮子与后方的两个轮子为相反转向就可实现。所以对于需求的速度以及角度，可以建立一个 X-Y 坐标系，通过正交分解得到 X 轴与 Y 轴的方向的速度，然后再叠加到各个轮子上即可。

计算式子：

```
set_right1=y_speed-x_speed - spine;  
set_left1=y_speed+x_speed - spine;  
set_right2=y_speed+x_speed + spine;  
set_left2=y_speed-x_speed + spine;
```


三、 硬件设计方案

3.1 主控芯片及主控电路

主控电路的作用是接受各个模块的反馈信息并提供相应的控制信号，主控芯片采用恩智浦公司提供的“K66”单片机，主控芯片通过核心板的方式接入主控电路中，控制电路连接的外部设备有摄像头，电机驱动板，电机编码器，步进电机驱动板，小型显示控制台，光流传感器，蓝牙。主控制芯片通过蓝牙模块与上位机进行通讯，通过串口与两个 OPENMV 进行通讯以获取在棋盘中的具体位置，通过 spi 与光流传感器进行通讯，光流传感器的作用是进行高速运动时的精确定位，以便进行位置环的控制。

主控芯片通过插入核心板接入到电路板中，接入原理图如图：

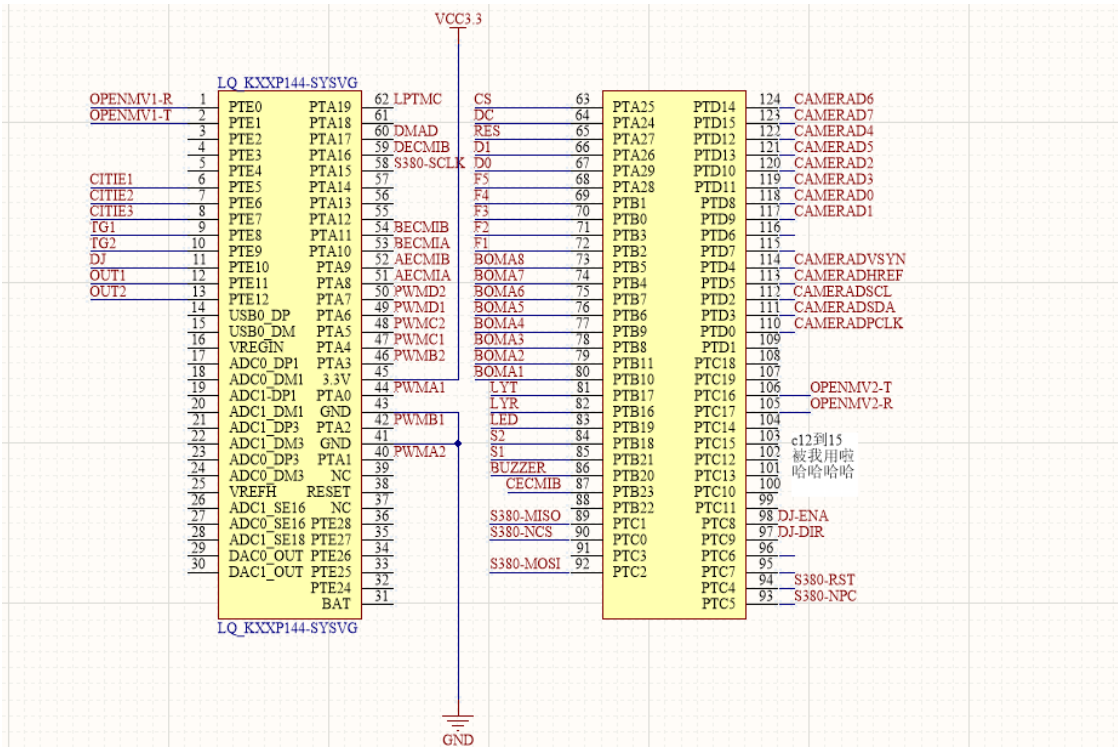


图 3-1 K66 核心板接口原理图

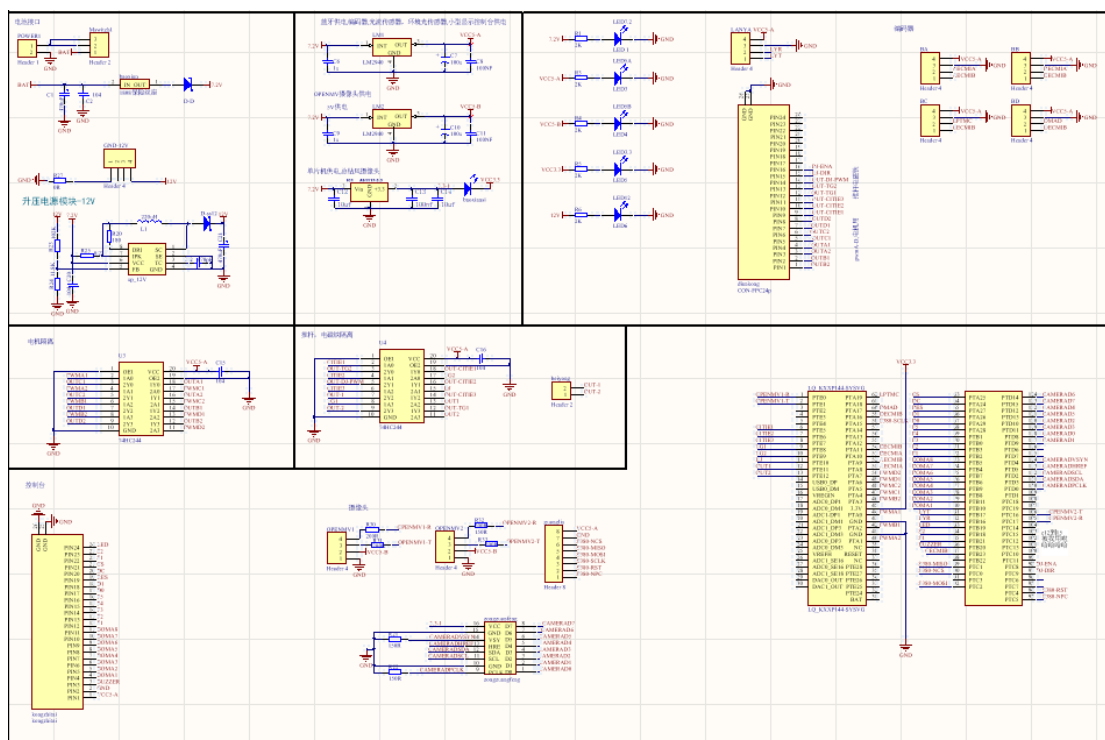


图 3-2 主控制板的电路原理图

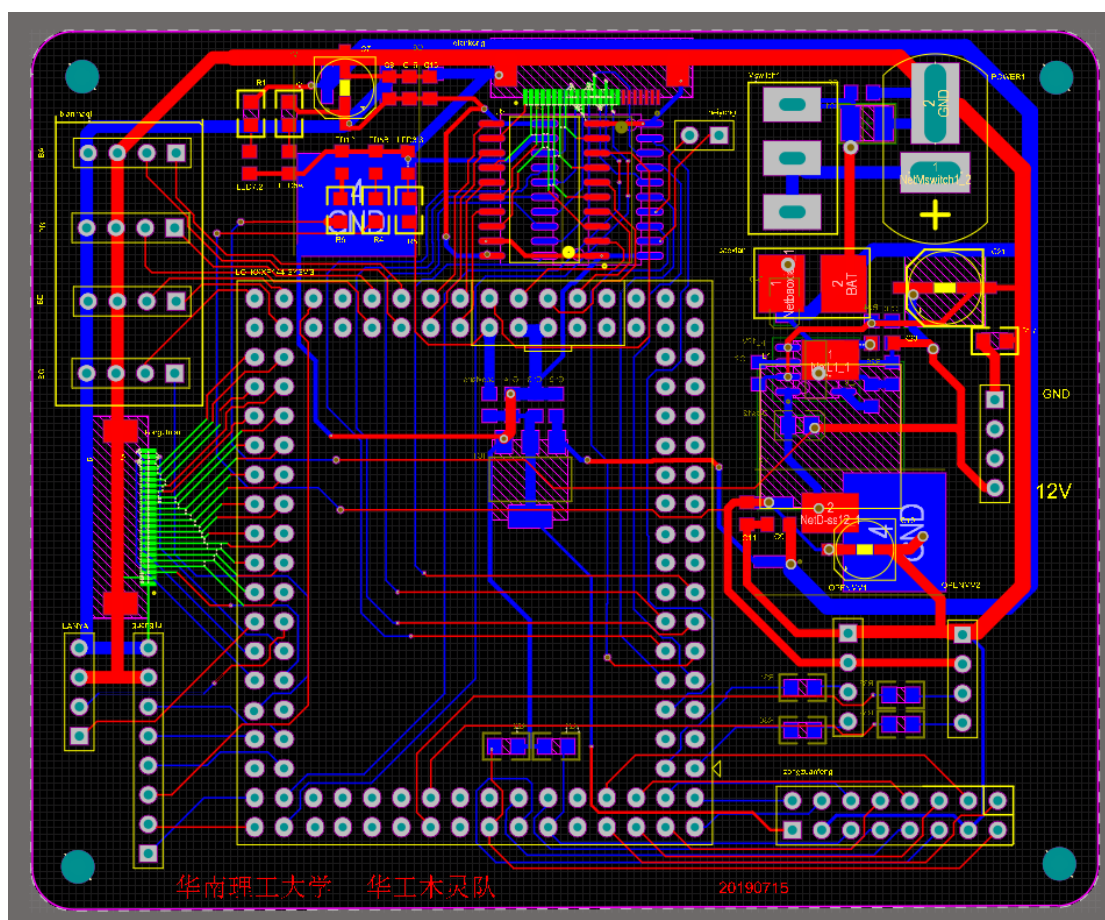


图 3-3 主控制板的电路 PCB 布局图

3.2 隔离模块

在嵌入式系统的设计中，一般采用 74 系列的三态门缓冲器来作为单片机的 I/O 口缓冲，以提高单片机接口的驱动能力，同时可以提供一定程度上的短路或反接保护。我们队伍采用的是 74HC244 芯片，主要作用在提高 I/O 口驱动能力的同时，防止驱动电路失效或者故障导致输出短路，将单片机的 I/O 口烧坏。由于需要驱动的器件较多，因此需采用两个 74HC244。通过 74HC244 进行驱动控制的模块有四个运动电机，一个步进电机，一个电推杆，五个电磁铁。

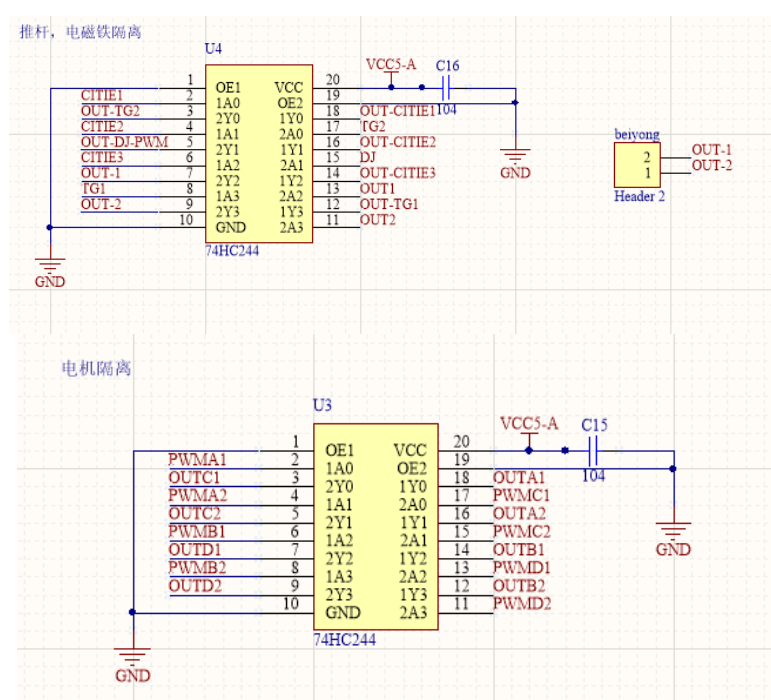


图 3-4 隔离模块原理图

3.3 电源设计

在电路设计中，需要首先分析的是电路上各元件的电压与功率需求。在本设计中，主要包括：

1. 单片机、编码器、OLED 显示屏，摄像头，光流传感器，环境光传感器等元件的 3.3V 与 5V 供电。

由于需要提供 3.3V，5V，12V，24V 的供电，经过稳定性的考虑，确定采取双电源方案，分别提供 7.2V 与 24V。然后通过稳压电路进行调整。

在对上述的各个模块的供电需求，我们进行了分类，3.3V 供电的有总钻风摄

像头，单片机，5V 供电的有电机编码器，蓝牙，两个 OPENMV，两个光流传感器，光电隔离模块，小型显示控制台。3.3V 与 5V 的供电都是小电流的供电需求，因此通过 7.2V 的电源进行稳压。

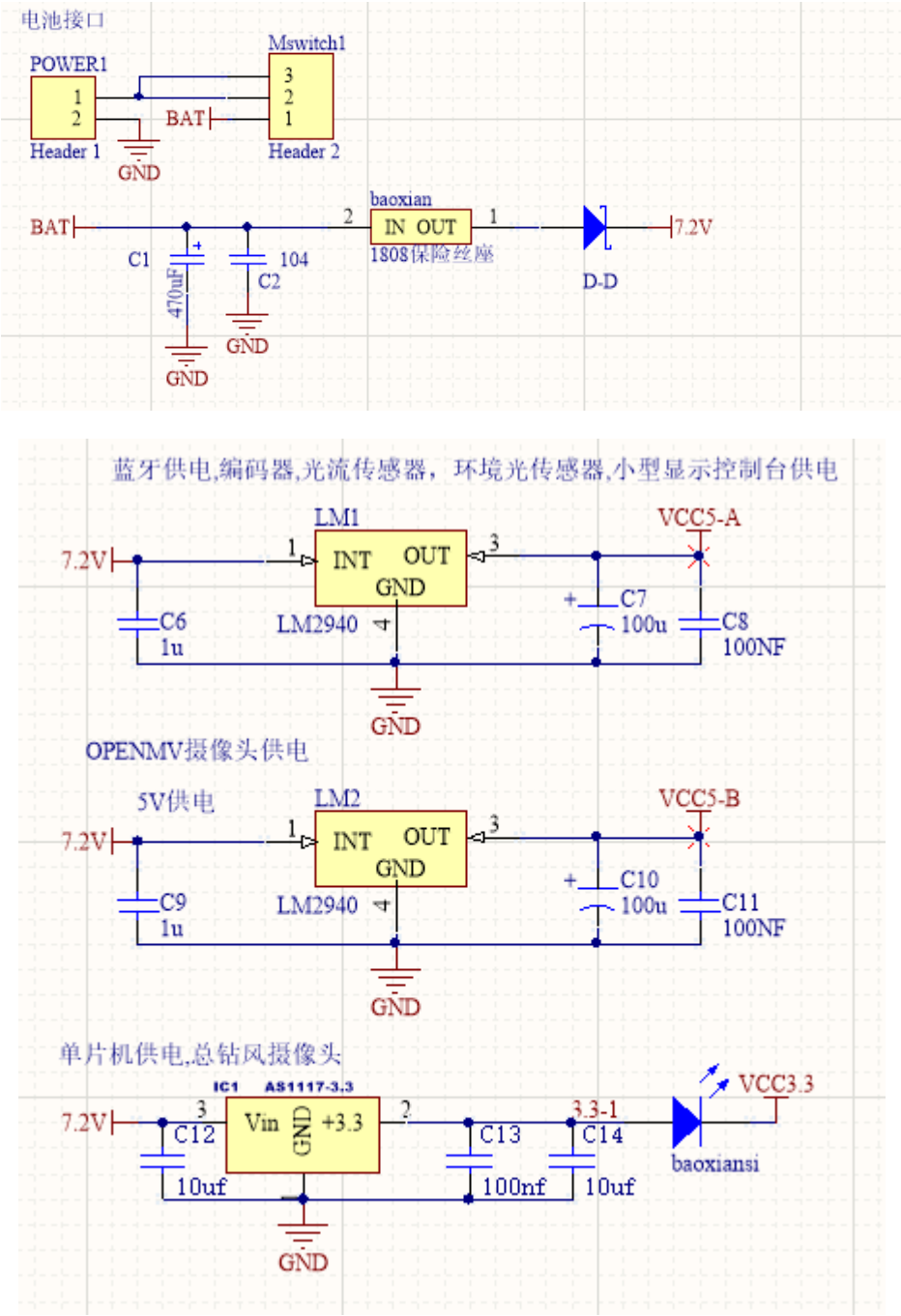


图 3-5 电路板电源部分方案

3.4 电磁驱动器方案设计

由于气泵在断电后在一定时间内，管道内会有一定的负压。可能会出现即使气泵完全断电，棋子也不能从吸盘上掉下的情况，导致搬运棋子或障碍失败。因此，我们希望能够控制气泵运行的方向（即产生负压与正压）。在放下棋子的过程中，使气泵吸盘短暂反向加压，使棋子能够得到释放。

要使气泵能够反接，经过队员的讨论总结出了如下的方法：

1. 采用双电源供电，使用+12V 与-12V 电源，通过 MOS 管或 IGBT 控制气泵两端的电压反向。
2. 采用全 H 桥进行反接控制
3. 采用双继电器，使两个继电器全部吸合或全部断开时，电流方向反相。

以上三种方法中，方法 1 需要设计负电源装置，电路比较复杂。在最初的设计中我们选择了电路相对简单的方法 3，但是由于两个电磁继电器很难做到断开与吸合的完全同步，在一个继电器吸合，另一个继电器断开的情况下，相当于电源短路。会在继电器的衔铁与电源变换装置上产生很大的短路电流，甚至会产生火花，使继电器寿命下降。如图，如果在切换瞬间，K1 已经吸合，但是 K3 还来不及吸合时，短路电流会如图红色箭头流过。长时间持续运行时，继电器可能会发热严重。

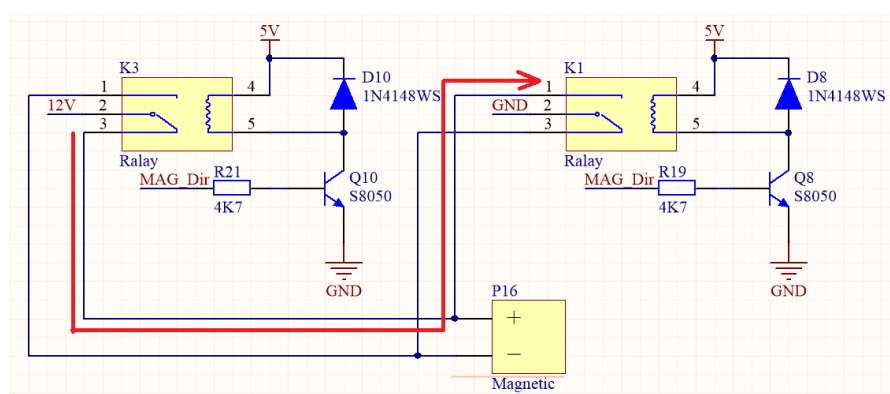


图 3-6 双继电器方案存在的问题

在新版本的硬件中，我们采用了性能和电路复杂度更加平衡的方法 2，即采用 MOS 管构成 H 桥。这样的方式来驱动可以反接运行的直流电动机等，是一种比较成熟的驱动方案。下面将详细地介绍经过改进的 MOS 气泵驱动电路的实现方案。

3.5 步进电机驱动

采用 lv8729 驱动芯片进行驱动，lv8729 驱动芯片驱动电流可达 1.8A，可以进行 128 细分，满足高驱动，高细分需求，支持的细分有：1、1/2、1/4、1/8、1/16、1/32、1/64、1/128； 为了提供足够大的扭矩和速度，选择驱动电源为 24V，lv8729 的驱动信号要求可以简化为 3 个信号输入即可，其他的引脚可以接入固定的电平，接入的三个信号为使能，脉冲，方向控制。由于 lv8729 工作时发热量大，因此必需安装散热片进行散热。

MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full Step
High	Low	Low	1/2 Step
Low	High	Low	1/4 Step
High	High	Low	1/8 Step
Low	Low	High	1/16 Step
High	Low	High	1/32 Step
Low	High	High	1/64 Step
High	High	High	1/128 Step

图 3-7 Lv8729 细分表

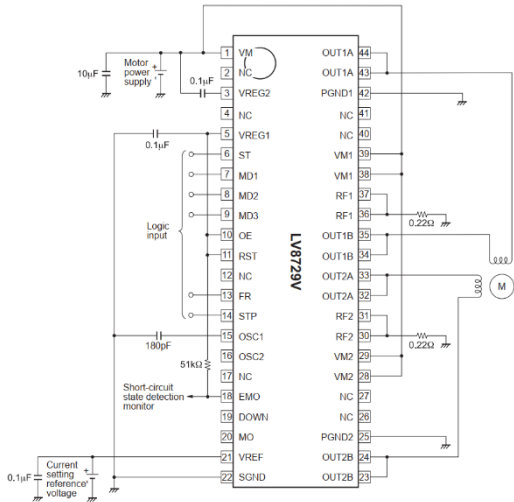


图 3-8 Lv8729 外围电路接线图

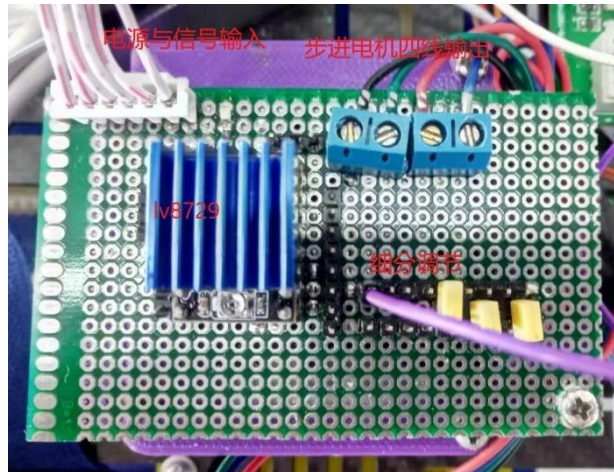


图 3-9 步进电机驱动接线图

3.6 光流传感器

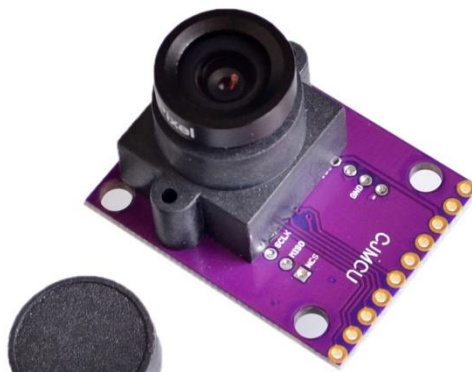


图 3-10 光流传感器

特点：光流传感器的主要特点是测距定位，常用于飞控的高空悬停。因此能用于车体移动过程的定位，以弥补摄像头的延迟问题。

工作原理：3080ADNS 光流传感器主要是通过一个低像数的镜头获取环境信息，通过自身的硬件对每一次的图像进行处理，得出自身的距离偏移量。

通讯方式：光流传感器的通讯方式主要是通过 spi 通讯。3080ADNS 光流传感器的处理芯片自带处理能力，单片机的主要任务是通过 spi 通讯方式向光流传感器进行命令传输与数据接收，通过简单的换算就能得到相应的车体运动偏移量。由于采取了两个光流传感器以防止死区的产生，由于 io 口的数量有限，所以要通过 spi 的片选进行通讯。spi 的片选功能解决了主机与多机进行通讯的问题。同时由于采用的是硬件级的 spi 通讯，所以信息交换的时间是非常短的，满足了通

讯的快速需求，通过实测可以得到同时与两个光流传感器进行通讯的时间可以达到 700 微秒的时间。

运作流程：单片机通过 spi 发送光流传感器的镜头参数命令对光流传感器进行初始化，然后两个光流传感器都会处于运行状态，等待接收下一个命令的指示。单片机需要执行的是定时发送读取指令对光流传感器的寄存器进行读数，然后通过二进制补码转换得出位置的信息。实际上光流传感器运行的独立性很高，单片机需要信息时只需发送查询命令并进行接收就行，所以对单片机的运算资源占用是非常低的。

3080ADNS 获取的图像：

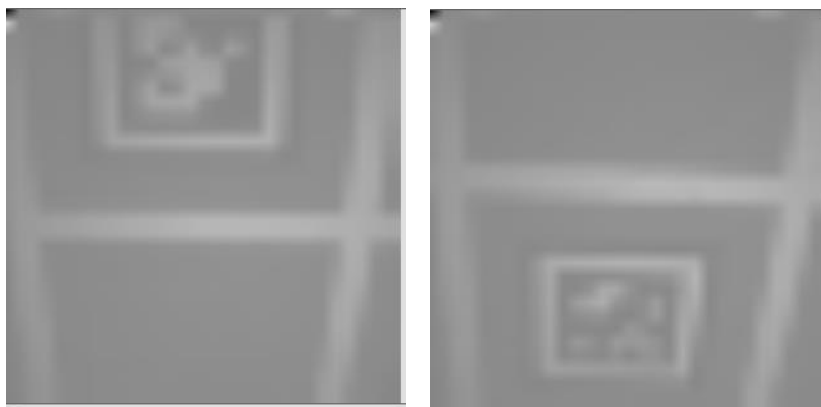


图 3-11 光流传感器获取的灰度图像

3.7 单片机显示控制台

小型显示控制台的主要作用是用于操作者与单片机的简单交互功能，电路中包含 OLED 显示，拨码开关，五项开关，轻触开关，指示灯和蜂鸣器，通过 FPC 排线与主控制板进行相连。

电路原理图：

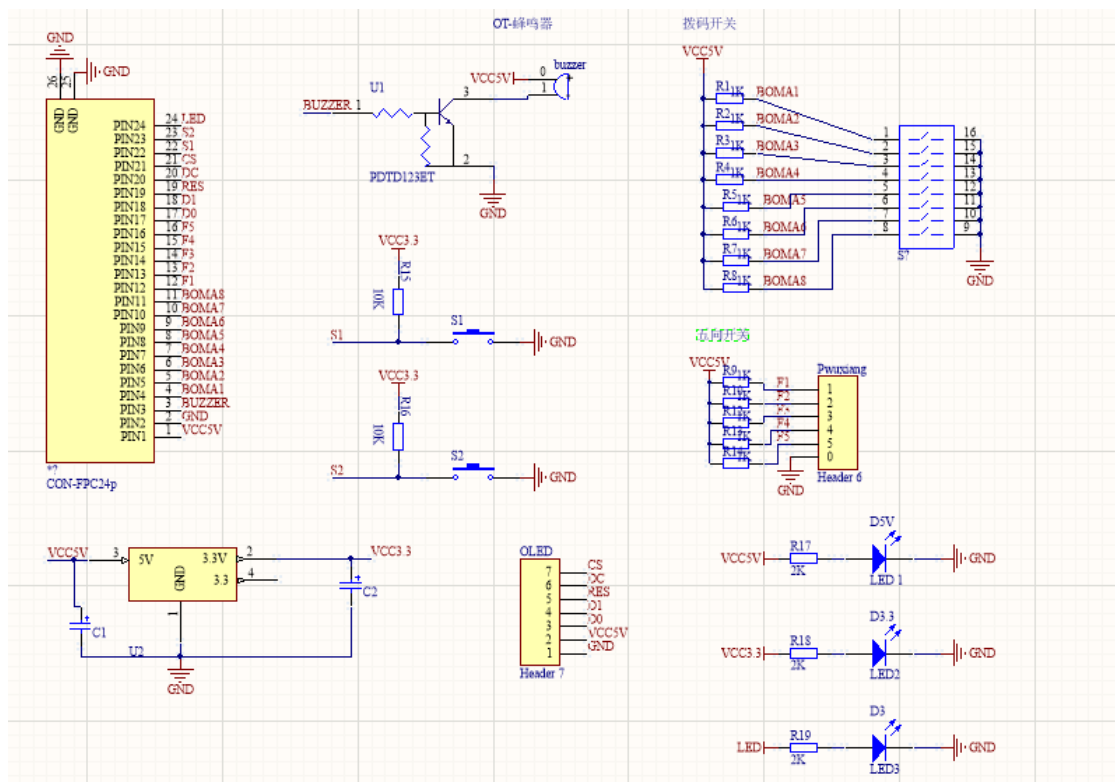


图 3-12 小型显示控制板电路原理图

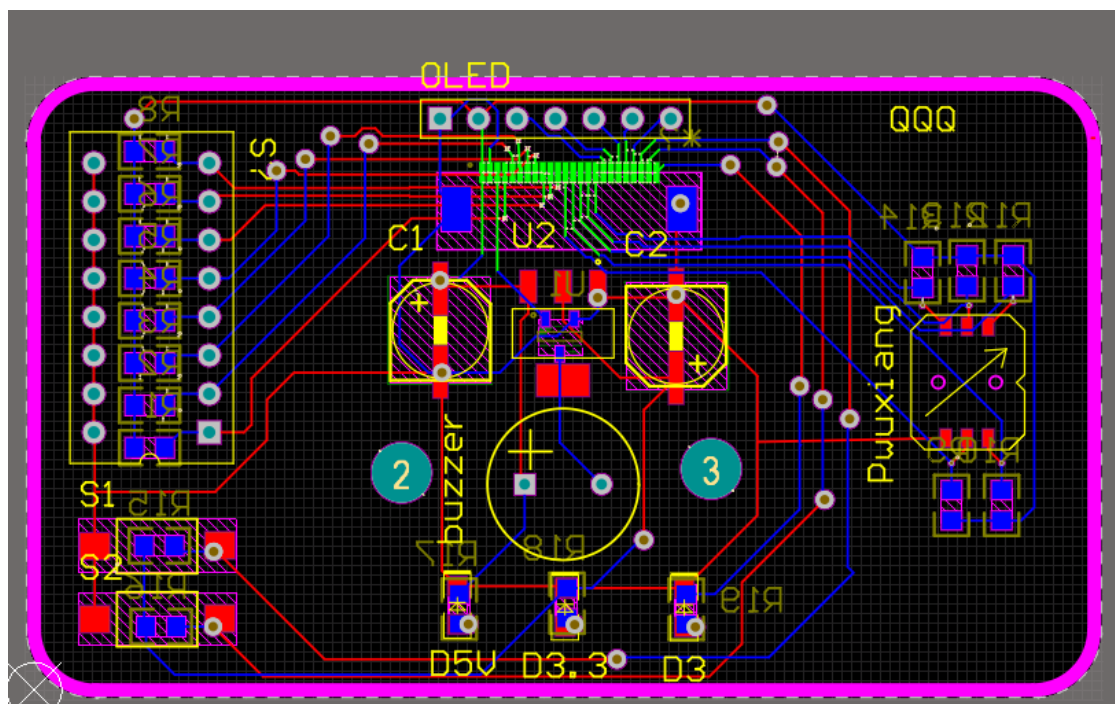


图 3-13 小型显示控制板电路 PCB 图

四、 软件设计方案

4.1 速度控制

电机的速度控制采用简单的 pid 控制。

pid 基本原理：

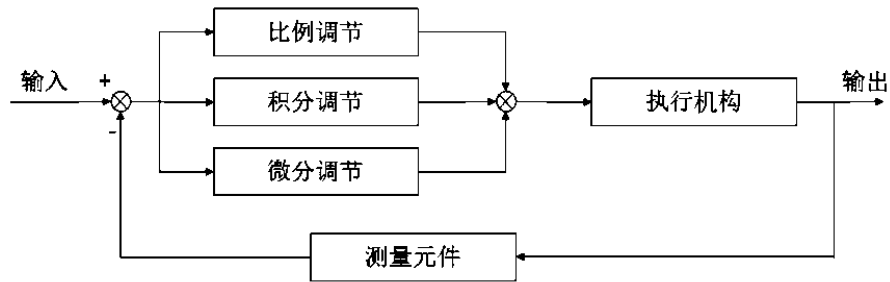


图 4-1 pid 基本原理

由于单片机运行的 pid 控制是离散系统的，因此采用下面的式子进行计算：

`pid->voltage = kp*pid->err + ki*pid->integral+ kd*pid->differential;`

pid 控制的效果与参数的整定有关，考虑到车启动时速度过大会导致车体的偏移过大，造成车体的运动极其不稳定，因此，在 pid 控速的基础上引入跟踪速度曲线的方法进行有效的调节。

速度曲线的目的是使车体在理想设定的速度中运行，包括加速，减速，恒速。速度曲线的实质是人为设定一条速度曲线图，通过 pid 进行跟踪，然后输出相应的 pwm 进行调速。速度曲线能最大限度的解决了四个电机速度变化过大时速度响应不一致的问题，同时可以解决追求快速性时容易导致过度超调的问题和车体启动偏移问题。

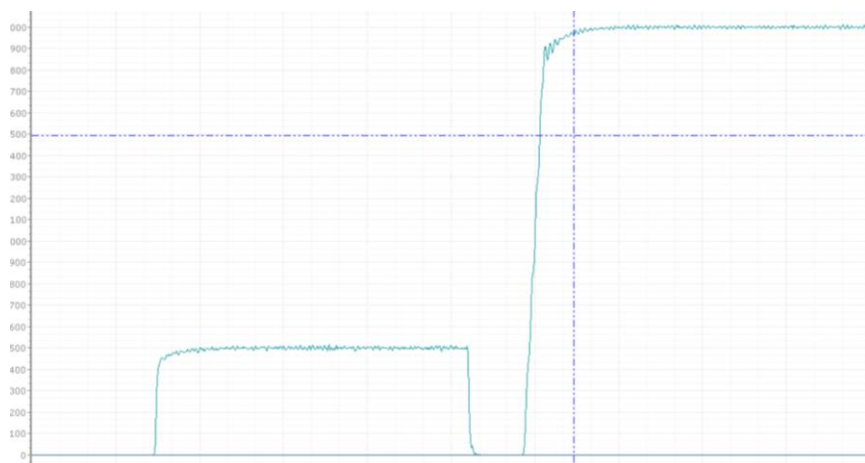


图 4-2 单电机速度跟踪曲线

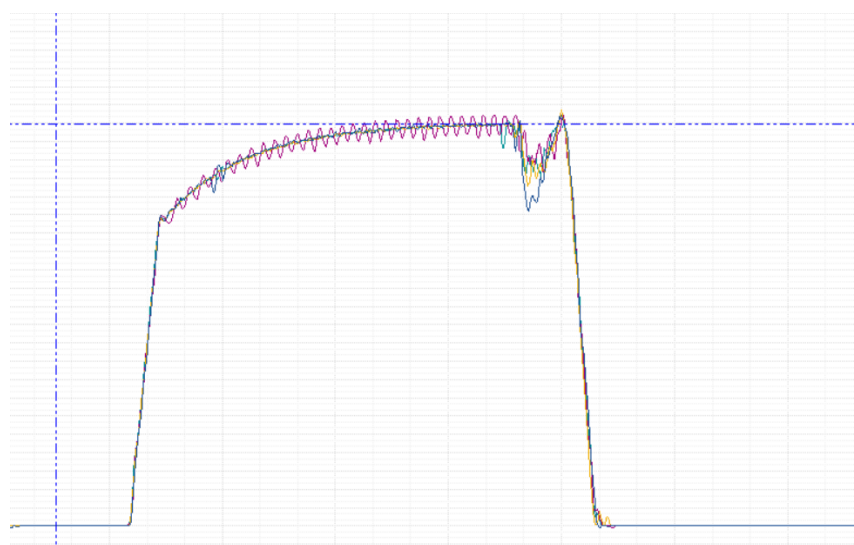


图 4-3 四个电机的速度调节曲线

在实际运动过程中，由于电机的缺陷和环境的影响，导致部分轮子未达到目标速度导致车身产生旋转，从而导致摄像头无法获取位置信息或者返回位置坐标不准确，对于电机的控制造成很大的影响。所以需要加入车身角度的闭环控制。通过摄像头返回的角度信息，进行角度 PID 计算出自旋的角速度，然后将该角速度与方向速度叠加，就可以使车身一直正向前方了。

另一方面，我们发现角度的校正很不线性，只用传统的 PID 控制很难适应角度的矫正，所以我们加入了模糊 PID 控制，发现效果很好。

4.2 可编程视觉模块 i.MX RT1062

在本次竞赛中，机器视觉部分使用的是 NXP 恩智浦公司的基于 i.MX

RT1062 的可编程视觉模块；i.MX RT1062 视觉处理模块（以下简称 i.MX RT1062 模块）为高性能的处理器，主控芯片使用 NXP i.MX RT1062，其主频为 600MHz，使用 Cortex-M7 框架，能够胜任一般的机器视觉处理任务，故本队伍选用两块 i.MX RT1062 模块用于完成室内创意组中的定位、位姿矫正、棋子障碍位置识别以及下棋策略计算任务。

4.2.1 i.MX RT1062 模块特点

i.MX RT1062 模块具有以下特点：

- 1. 处理器性能优
主控芯片使用 NXP i.MX RT1062，600MHz Cortex-M7，1MB RAM，4MBFlash
- 2. 开发效率极高
i.MX RT1062 模块集成了 Micropython 运行环境，使用 Python 语言做应用程序的二次开发
- 3. 调试程序高效
兼容 OpenMV IDE 的调试监控协议，下载即可查看效果、监控帧缓冲、虚拟串口控制台

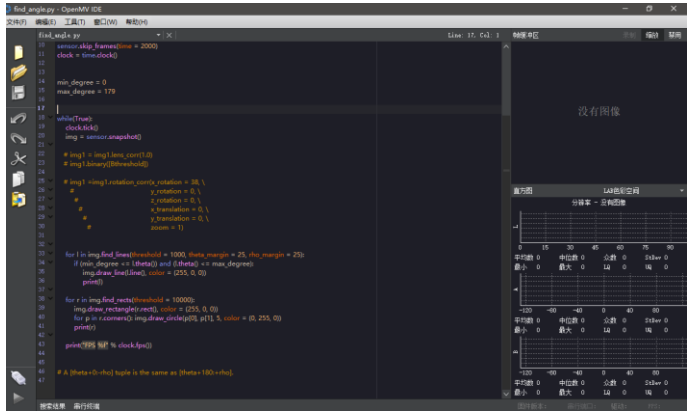


图 4-4 视觉处理调试界面

4.3 定位部分

本次比赛的场地为在八乘八的方形框内放置 AprilTag，运用 AprilTag 的 tag25h9 族的前 32 个码型作为位置定标的基础。故利用 AprilTag 能够使车体在场地中的任意一个位置获取其坐标。

4.3.1 AprilTag 三维定位技术

AprilTag 是一种先进的视觉定位技术，通过不同的 AprilTag 码型，通过视觉处理之后能够得到相机相对于 AprilTag 的相对位置，只要提前获知 AprilTag 码的位置，就能够快速获取到相机的相对位置，这对移动物体的定位有着广泛的应用。本此 i.MX RT1062 模块处理 AprilTag，通过测算转换后，通过获取 AprilTag 距离车体 XY 方向的距离，然后通过 AprilTag 在世界中的实际坐标，即可推算出车体的位置。

AprilTag 的 tag25h9 族有 35 个，取前 32 个摆放到棋盘中得到相应的定位地图。



图 4-5 AprilTag 的 tag25h9 族

AprilTag 的识别首先为通过高斯模糊等预处理，再作相应的像素梯度处理，计算 AprilTag 的每一个像素点的梯度方向和幅值，并且把相同的梯度方向和幅值得像素集群到一个部件中。进而用递归深度优先搜索算法得到四边形。

AprilTag 距相机距离与角度是通过单应性和外在评估 (Homography and extrinsics estimation) 的方法得到，由于是图像通过运用线性变换 (DLT) 算法，以齐次坐标表示，所以它的定义只有按比例。如果需要推算出实际标签的位置和方向，需要知道相机的焦距和标签的物理大小。

基于以上特点，AprilTag 可在更远距离、较差光线和更扭曲的图像环境下被检测到。AprilTag 可应对所有种类的图像失真问题。

通过计算实际距离和 i.MX RT1062 模块中的 `image.find_apriltags()` 函数返回的 `tx`, `ty`, `tz` 的比例, 就能够推导出 AprilTags 与相机的角度和距离。

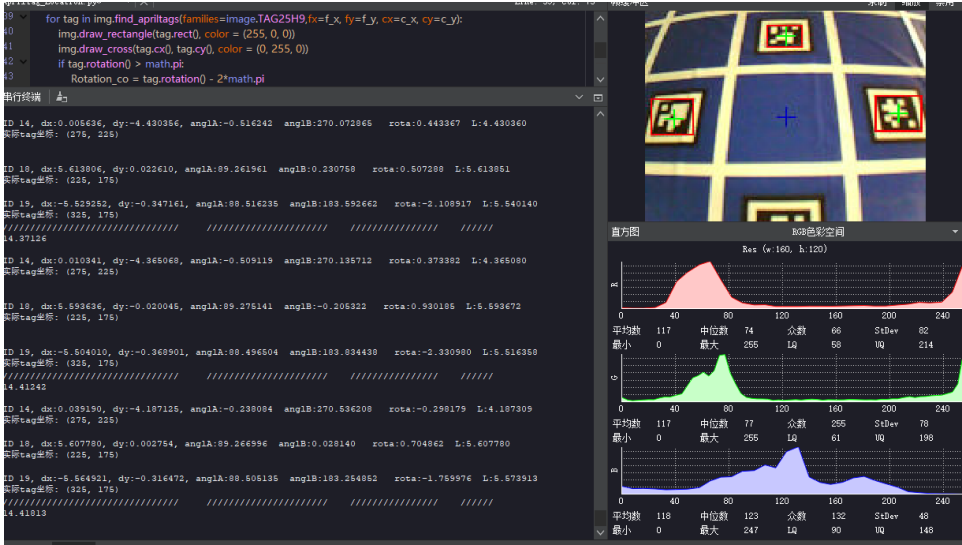


图 4-6 AprilTag 的识别和识别结果输出

4.3.2 线检测获取车体旋转角度

由于车体需要角度矫正才能完成车体移动和放置障碍等任务, 矫正车体角度显得尤为重要, 但是 AprilTag 计算得的角度无法满足实际的工程需要, 需要通过线检测能够识别出图像中存在的直线的角度, 基于摄像头安装的基准角, 即可计算出实际车的偏角。

霍夫线检测是通过参数化的几何特征来进行坐标系的转化, 在笛卡尔坐标系中表示一条直线有两个参数斜率 k 与截距 b , 而在霍夫空间中表示一条直线也有两个参数到原点的距离 d 与角度 θ , 在笛卡尔坐标系的点集反映在霍夫空间为一定曲线, 当不同曲线交与同一点时, 表明霍夫空间交于点的曲线所代表的笛卡尔坐标系点集为同一条直线。

对于图像上的任意一点 $p(x_0, y_0)$, 有 $R_\theta = x_0 \cos \theta + y_0 \sin \theta$.

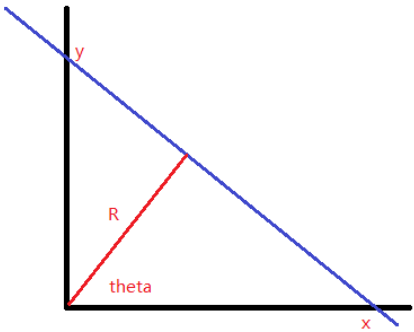


图 4-7 霍夫线检测原理

算法：

线检测步骤为，对图像进行畸变矫正和透视变换；由于 openmv 内存空间有限，对复杂的图像处理要求分辨率低，在低分辨率下图像中的粗直线的边界模糊，为此可以先对图像进行模糊处理，将两条白线边界结合成一条，去除同一条白线两边界模糊对结果的干扰，即先对图像进行均值滤波，然后进行线检测处理，该模糊处理适合快速的低分辨率图像。线检测的方式为图像进行算子模板处理之后，利用霍夫线检测原理，调用 openmv 中的 `image.find_line()` 函数设置一定的检测阈值即可得到图像中理想的线条的角度。

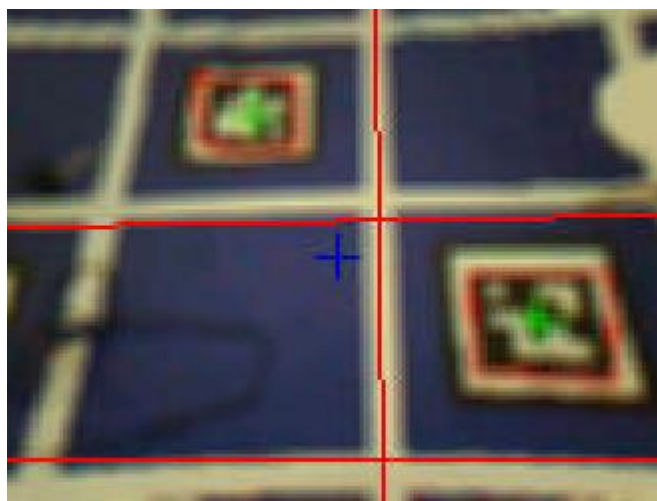


图 4-8 线检测处理结果

算法改进：

由于上述方法基于低分辨率的图像并做了模糊处理，识别的精度可以进一步提高。算法改进的思路为将 openmv 设置成高分辨率的模式后仅截取图像的中心部分进行角度的获取。设计的算法能够有效结合竞赛场地的特点快速计算；计算角度的算法思路为搜索局部二值图像中的黑白跳变的点，然后进行统计阈值筛选，最终的能够准确识别出线条并且准确度较高。该方法能够检测具有一定倾斜角度的直线，当设定一定的线段搜索范围后，算法效率较高。



图 4-10 改进后的线检测

4.3.3 低分辨率摄像头精确校位

由于 i.MX RT1062 模块的分辨率较低，将 openmv 架设在较高位置对于放置障碍尚有一定的难度，故需要使用低分辨率摄像头对车体到白线的精确校准定位，使得障碍能够准确落在白线上。

该摄像头运行在主控 NXP K66 上，获取到灰度图像之后进行大津法二值化处理后搜索边界白线位置，以精确定位。



图 4-11 低分辨率摄像头处理后图像

4.4 障碍棋子识别方案

4.4.1 圆形棋子的识别定位

由于八皇后任务以及步步为营任务均需要作品自行识别白色圆形棋子的位置在进行策略的计算和执行相应的动作，故需要作品具有自主识别棋子位置的功能，i.MX RT1062 模块的圆形检测能够获取到图像中圆形的相对于画面中心的位置，根据之前 AprilTag 中计算得到的车体位置，由于标定后的图像的像素点与实际平面坐标系的实际距离可以通过简单的比例即可换算，故能够结算得到出现在图像中的圆形棋子。识别棋子的结果如下图。

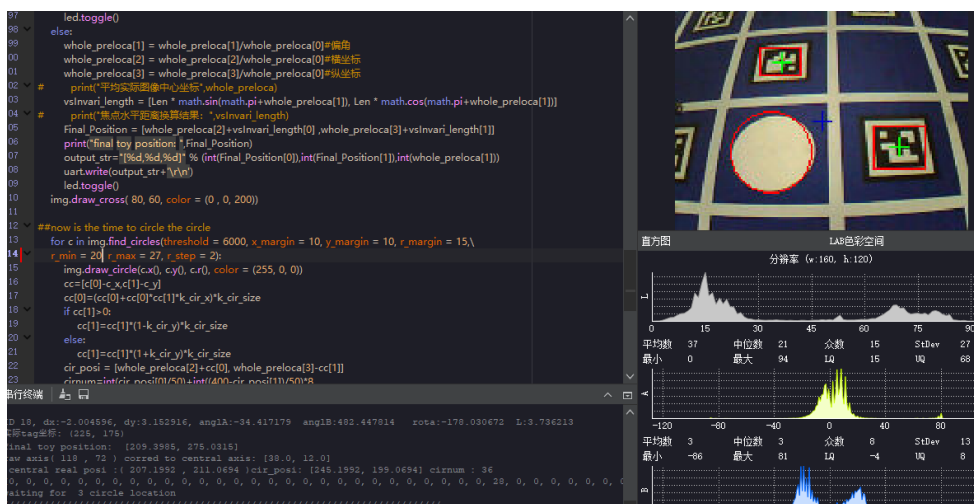


图 4-12 圆形检测及位置输出结果

4.4.2 红色障碍的识别定位

由于红色障碍在棋盘上显示明显的红色，通过色块的识别能够有效得到色块在图像的位置。对于 i.MX RT1062 模块的 `img.find_blobs()` 函数，需要先行得到图像中色块的颜色阈值信息，通过实际 lab 颜色空间的阈值处理得到其 roi，利用 openmv 的 lab 阈值编辑器处理的结果如下图所示。

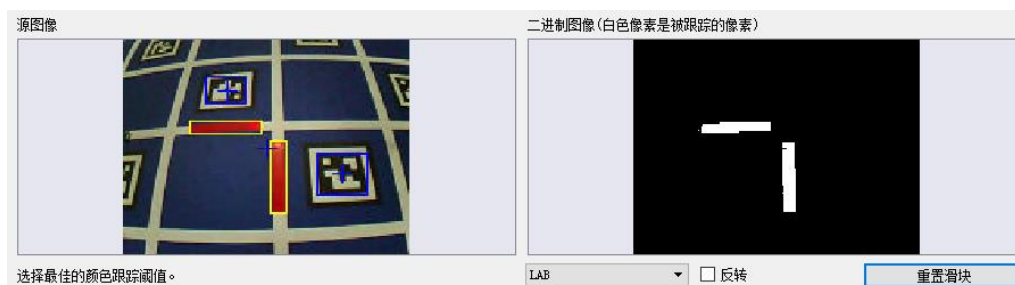


图 4-13 红色障碍 lab 空间阈值筛选结果

通过以上的 lab 空间筛选二值化后统计白点，继而能够得到该区域的位置信息，利用和圆识别推算位置的方式同样能够得到相应的障碍位置信息，特别的，根据障碍区域的长宽比值，可以进一步提高横向障碍和纵向障碍的区分。

五、 算法设计方案

5.1 关于八皇后问题的解决方案

八皇后的问题就是在 8x8 的棋盘上将八个棋子都放好之后,使其横竖斜线上均只有一个棋子。在实际的比赛之中八皇后问题解决得优劣是靠完成摆放的时间,因此除了车模运动时候的速度以及捡取棋子摆放障碍的速度之外还有的就是算法路径规划的优劣。对于八皇后问题来说最终可以完全符合八皇后摆放方法的一共有 92 种类的摆法,在已知这 92 种摆法的情况下处理现有的问题

5.1.1. 解决思路

为了使得完成八皇后问题的时间最短,需要考虑使得移动棋子数目最少,并且考虑场地之中车模移动棋子的顺序,使得其总的移动路径最短,这样就可以使得时间在算法上控制到最短。

所以首先解决移动棋子数目的问题。从现场获取到的棋子的分布位置之中,对于已有的 92 种解法进行对比,先找出移动棋子数目最少的若干种解法。将其单独取出。在再做下一步的计算。

其次是行走路程问题的最优解。行走的路程包括车子到达需要搬运棋子的距离,搬运棋子到目标位置的距离的总和,由于车子采用的是麦克纳母轮的全向运动原理,可以实现任意方向的直线运动,因此在行走路程上只需考虑每次运动起始点和目标点的直线距离的总和。因此最短行走距离的最优解求法是计算出实现摆放次数最少的解法中小车需要行走的总路程,取出结果最小的行走方式,即为行走路程问题的局部最优解。

将摆放次数最少和行走路程最短的局部最优解的结合,即可得出平台实现八皇后问题的用时最少的方式。

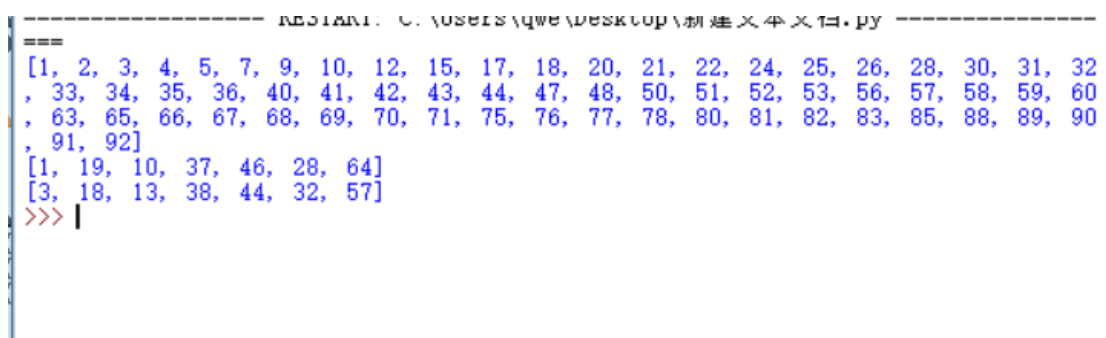
然而在实际运行过程中随着算法摆放次数最少的局部最优解的个数增加计算量呈指数增长,实际进行计算时间的估算时发现如果八个棋子分布呈同一直线或斜线出现需要 7 阶迭代的情况时,需要计算的情况高达 23 亿多种,计算机需要很长的计算时间,然而,出现这种情况的棋子分布只有 18 种,因此只要将这 18 种情况独立出来按照预设的方案执行,最大计算量将从 23 亿多种减少到 3 千多万种,可以快速处理出来,大大地减少了计算时间。

5.1.2. 实现方式

使用 python 语言进行编辑，可以在 openmv 上直接进行运算，将 92 种方法以列表的形式写入 python。之后在获取地图的棋子信息之后，同样存放为列表，然后与 92 种一一对比，找出移动棋子数最少的若干种方法。在找出的方法之中进行迭代，计算不同摆放顺序之下放置所需要走过的距离。这样就可以找出走出最短路径的方法。

5.1.3. 运算实例

前面讲到最多迭代的方法，最多需要移动七个棋子，此时迭代次数过多，利用 k66 或者 openmv 计算会很慢或者内存不够用，利用个人计算机先进行遇见你算，使用的 python 程序同 openmv，计算形如[1,10,19,28,37,46,55,64]列表字，算法比较出最短的运动距离，规划出的搬运路径如下：



```
===== RESTART: C:\Users\qwe\Desktop\新建文本文档.py =====  
[1, 2, 3, 4, 5, 7, 9, 10, 12, 15, 17, 18, 20, 21, 22, 24, 25, 26, 28, 30, 31, 32  
, 33, 34, 35, 36, 40, 41, 42, 43, 44, 47, 48, 50, 51, 52, 53, 56, 57, 58, 59, 60  
, 63, 65, 66, 67, 68, 69, 70, 71, 75, 76, 77, 78, 80, 81, 82, 83, 85, 88, 89, 90  
, 91, 92]  
[1, 19, 10, 37, 46, 28, 64]  
[3, 18, 13, 38, 44, 32, 57]  
>>> |
```

图 5-1 规划的搬运路径

5.2 决赛对抗方案

5.2.1. 决策算法分析

不管是围棋、国际象棋、五子棋，还是决赛中的步步为营，也称墙棋，它们本质上都能称作“博弈”。博弈，可以理解为有限参与者进行有限策略选择的竞争性活动，比如下棋、打牌、竞技、战争等。而大多数的对抗棋类游戏都属于零和博弈。

参与博弈的各方，在严格竞争下，一方的收益必然意味着另一方的损失，博弈各方的收益和损失相加总和永远为“零”，双方不存在合作的可能。博弈游戏，目的是寻找最优的方案使得自己能够利益最大化，对方利益最小化。

我们使用博弈树来记录棋局状态，通过遍历棋盘的位置推算，可以延伸出多层枝叶，理论上可以在博弈树上得到棋局的所有状态。当我们得到一个具有足够深度的博弈树时，就需要进行决策使得自己利益最大化。这就涉及到极大极小值搜索算法，从 Max 即 Player2 的角度，设置一个权值 w 给博弈树中的每个节点，用来表征这个节点(某一种状态)对 Max 的有利程度,这个权值越大，表示这种状态对 Max 更有利。在模拟对弈的过程中，Max 倾向于选择 w 极大的节点，而 Min 倾向于选择对自己最有利的棋局状态，即 w 极小的节点，根据这种思想，将博弈树拓展到叶节点，然后回溯回去即可得到应对各种情况的最有利策略。

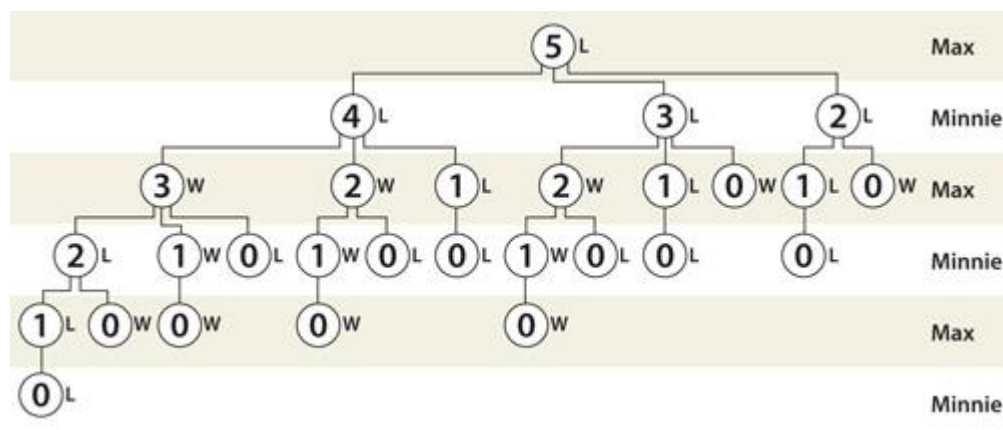


图 5-2 极大极小值搜索算法示意图

若博弈树的规模太过庞大，即使是计算能力超强的计算机也无从招架，因此在算法设计中，我们需要限制搜索深度和进行剪枝，降低搜索的规模，提高决策速度。

在极大极小值的基础上，使用 α - β 算法剪枝可以有效地提高运算效率，对一些不必要的估计值进行舍弃。其策略是进行深度优先搜索，当生成结点到达规定深度时，立即进行静态估计，一旦某一个非端点的节点可以确定倒推值的时候立即赋值，节省下其他分支拓展到节点的开销。

剪枝规则：

α 剪枝，任一极小层节点的 β 值不大于他任一前驱极大值层节点的 α 值，即 α (前驱层) $\geq \beta$ (后继层)，则可以终止该极小层中这个 MIN 节点以下的搜索过程。这个 MIN 节点的倒推值确定为这个 β 值。

β 剪枝，任一极大层节点的 α 值不小于它任一前驱极小值层节点的 β 值，即 β (前驱层) $\leq \alpha$ (后继层)，则可以终止该极大值层中这个 MAX 节点以下的搜索过程。这个 MAX 节点的倒推值确定为这个 α 值。

5.2.2. 上位机设计

我们设计的上位机用于调试对抗决策的算法以及决策结果显示。可在 GUI 界面上移动棋子和摆放障碍模拟对局过程，由决策算法给出策略并显示。

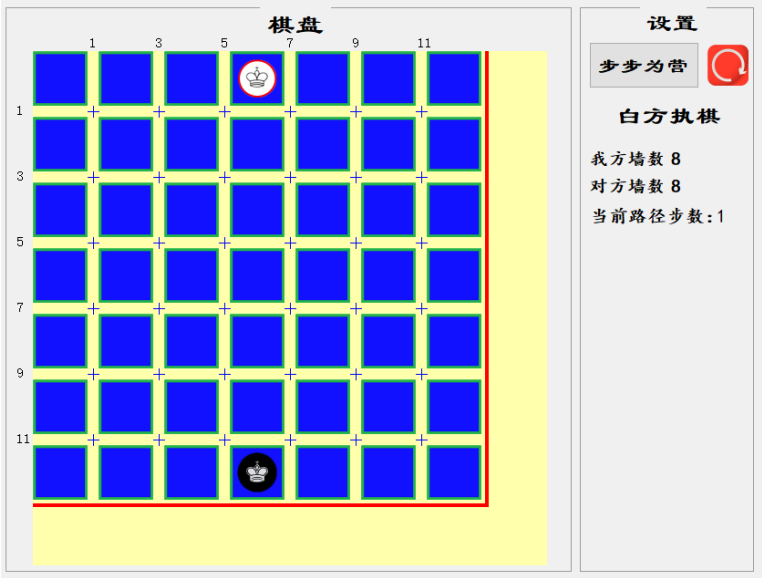


图 5-3 上位机界面

为了得到我方以及对方当前的获胜最短路径，我们使用了 A*寻路算法。A*算法吸取了 Dijkstra 算法中的 $cost_so_far$ ，为每个边长设置权值，不停的计算每个顶点到起始顶点的距离，以获得最短路线，同时也汲取贪婪最佳优先搜索算法中不断向目标前进优势，并持续计算每个顶点到目标顶点的距离，以引导搜索队列不断想目标逼近，从而搜索更少的顶点，保持寻路的高效。

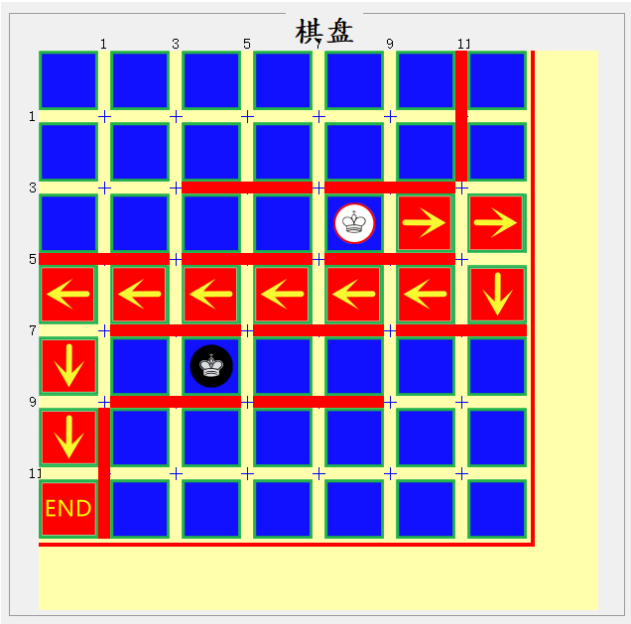


图 5-4 计算最佳路径

根据规则计算获得当前我方皇后可达的所有位置，使用评估算法结合寻路结果对每个选择进行评分，评分算法作为极大极小算法的选择的权值，完成最终决策。

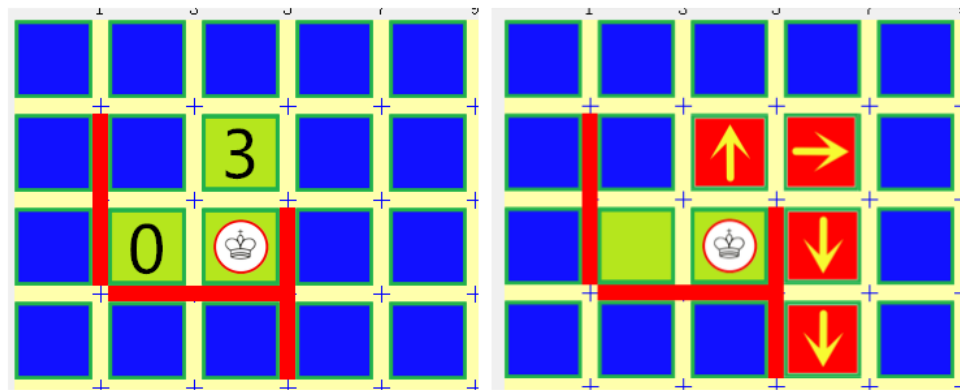


图 5-5 评分算法

附 A*寻路算法

```
while (!finish && !heap.isEmpty())
{
    Data *data = heap.getMinAndRemove(); // 取出g值最小的点
    //再找出其下一步可达的所有点，记录在reachableCell数组中
    findAllReachable(1, data->pos.x, data->pos.y, wOrb);

    for(int nx=0; nx<13; nx+=2)
    {
        for(int ny=0; ny<13; ny+=2)
        {
            if(reachableCell[nx][ny]==1) //该位置可达
            {
                Point pos = Point(nx, ny);
                if(pos.x==endPos[num].x&&pos.y==endPos[num].y) // 如果是终点,则跳出循环,不用再找
                {
                    finish = true;
                    lastData = data;
                    break;
                }
                auto nn = heap.find(pos);
                //已经在openlist中, 检查这条路径是否更优
                if (nn)
                {
                    if (nn->g>data->g + 1) //data->g + 1表示先经由当前格(data->g)再到nn(+1),比nn的g值更小
                    {
                        nn->g = data->g + 1; //重新计算nn的g值
                        nn->parent = data; //把nn的父亲设为当前格
                    }
                }
                else
                {
                    //返回空指针 则将当前格子加入openlist中 pos.distance(endPos) 到终点的直接距离
                    heap.add(new Data(pos, pos.distance(endPos[num]), data->g + 1, data));
                }
            }
        }
    }
}
```

图 5-6 附 A*寻路算法

六、 作品特色描述

6.1 机械部分

6.1.1. 自行设计低重心车体

为了满足车模在完成创意比赛过程中快速移动和对地面操作的要求，要求车体能够在车体在棋盘能够快速移动，故设计了低重心的车模，同时将捡起棋子和放下障碍的机构安装在车的底部，有效降低的车体重心，保证车体运动的快速和平稳。

6.1.2. 导轨式障碍放置结构

针对障碍放置任务，设计了四向快速的障碍放置方案，方案采用四向导轨，使用步进电机驱动齿条的方式，完成横向和纵向的障碍放置，使得车体在到达位置后迅速伸出导条，直接放下障碍即可退出棋盘外，比一般的机械臂方案要快速，同时控制简单快速，故障率低。该设计结合了车体底板底、车体重心低等特点，合理利用了车体底部的空间，使得车体的重心进一步降低，利于降低车模的运动控制。

四导轨的设计还能够直接让车体同时挂起四个障碍，减少了挂放障碍的次数，同时也使得车模能够同时带动四个障碍和一个棋子。能够很好地满足车模的步步为营的任务需求。



图 6-1 导轨装配结果图

6.2 硬件部分

6.2.1. 可拓展的驱动电路方案

由于车体的外设较多，包括五个电机、摄像头模块、若干电磁铁、电推杆等，需要在调试的过程中随时启用和暂置这些外设驱动的通断；同时在八皇后和步步为营的任务过程中，障碍放置导轨装置和电磁铁控制不同。采用自行设计的可拓展的驱动电路板，有利于随时启用的这些外设，使得车体的安全性更高，更重要的是使得车体的维修维护更加快速便捷，能够保证车体在比赛的过程中，由于可以备份易拔插的单个驱动板，不会出现驱动急需更换而缺少时间更换所有驱动的问题。

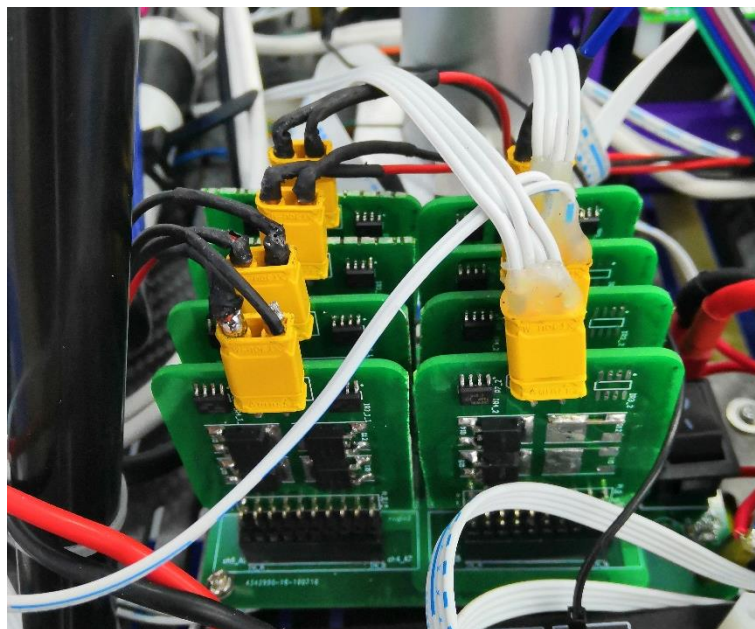


图 6-2 可拓展驱动板

6.3 机器视觉部分

6.3.1. 光流传感器

光流传感器是检测画面中移动的部分并输出相对位移，光流传感器快速测量自身的相对位移，继而能够得到车体的位移。改方案是为了解决车体在快速移动过程中由于机器视觉处理模块处理帧率过低的问题。在车体定位和物体识别的任务交给 openMV 模块，当车体快速的移动的时候，光流传感器将辅助 openMV 模块进行移动过程中的位置闭环。

6.3.2. 双机器视觉模块定位

为了将位置定位、棋子和障碍的识别过程速度提高，同时提高车体的定位精度。使用了双 i.MX RT1062 模块的视觉方案。双模块同时识别并通信，能够保证车体在棋盘任意位置的定位和快速移动，保证车体不会出现无法通过视觉定位的情况。

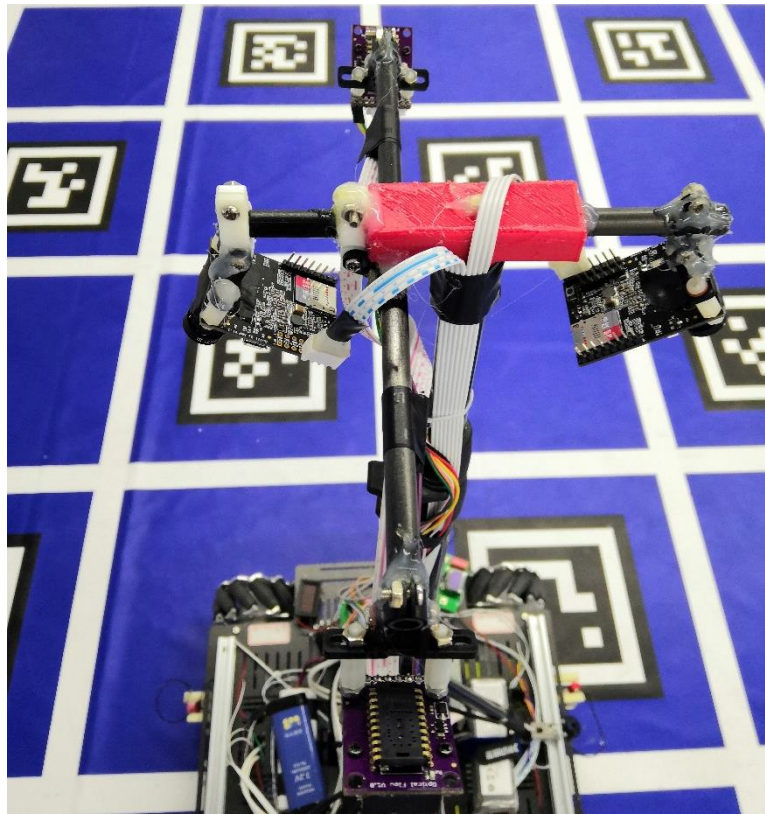


图 6-3 双 i.MX RT1062 模块及光流传感器