第十七届全国大学生
智能汽车竞赛

完全模型
技 术 报 告

学　　校：　厦门理工学院

队伍名称：　暗涌惊涛

参赛队员：　江力、程佳文、陈思铨、纪烽顺

带队教师：　周承仙、周薇

关于技术报告和研究论文使用授权的说明

本人完全了解全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：江力 程佳文 陈思铨 纪烽顺

带队教师签名：周玉仙

日期：2022.08.20

目录

第一章 引言

随着信息科学技术的迅速发展，电子技术，人工智能技术在汽车行业运用越来越广泛，如此一来，智能化的汽车领域就显得尤为重要了。智能化已经被作为衡量电子产业的重要标准。智能化的汽车也会在未来的市场占据主导地位。受教育部高等教育司委托(教高司函[2005]201 号文)，高等学校 自动化专业教学指导分委员会主办“恩智浦”杯全国大学生智能汽车竞赛，以 迅猛发展的汽车电子为背景，涵盖了控制、模式识别、传感技术、电子、电气、 计算机、 机械等多个学科交叉的科技创意性比赛。参赛选手须使用竞赛秘书处统一指定并负责采购竞赛车模，自行采用指定的微控制器作为核心控制单元，自主构思控制方案及系统设计，包括传感器信号采集处理、控制算法及执行、动力电机驱动、转向舵机控制等，完成智能汽车工程制作及调试，于指定日期与地点参加场地比赛。参赛队伍之名次（成绩）由赛车现场成功完成赛道比赛 时间为主，技术方案及制作工程质量评分为辅来决定。

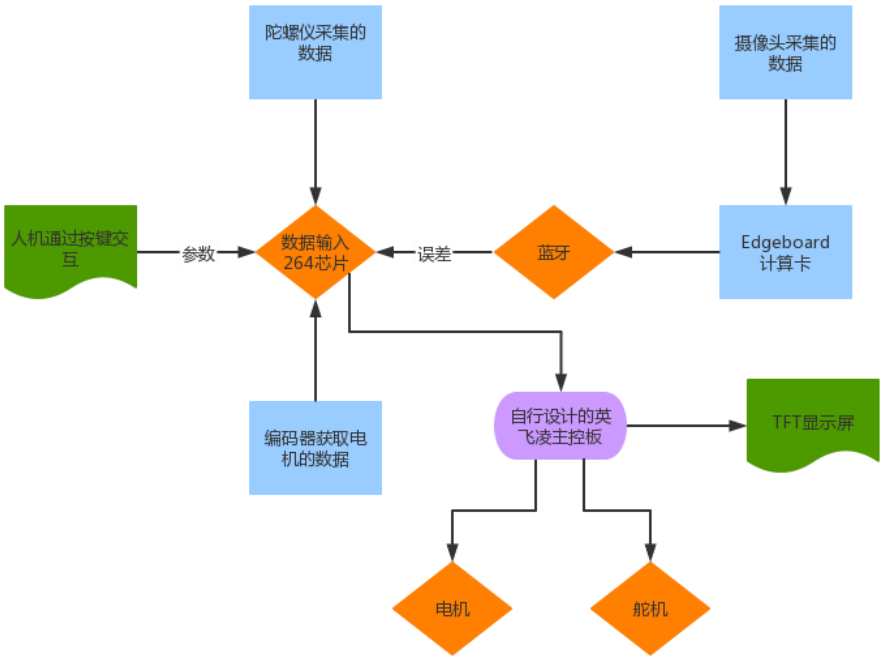
本篇技术报告所阐述内容主要包括了百度完全模型组的机械系统、硬件系统、软件系统等。具体表现在电路的创新设计，算法的独特控制，在竞赛准备的过程中，培养了我们对电路设计，程序设计编程，系统的调试等方方面面的能力，让我们的知识融会贯通，并且运用到了实际中，提升了我们的创新能力，对我们今后的学习及工作有莫大的积极影响。

第二章 智能车系统总体设计

2.1 整车设计思路

本次智能车竞赛本队使用 I 型四轮车模，主要由检测系统，控制决策系统和动力系统构成。其中检测系统采用摄像头，六轴姿态传感器，控制决策系统采用 TC264 英飞凌和 Edgeboard 计算卡作为主控芯片，动力系统主要有计算卡计算误差，通过 usb 转 ttl 传给 264 单片机控制 SASU 的数字舵机 CS-3120 大学生智能车竞赛专用舵机的转角和大学生智能车竞赛专用直流电机的转速。整体控制框架通过摄像头采集图片，然后将图片传给计算卡，在计算卡里面运行相应的图像处理代码，来达到控制的效果，同时由六轴姿态传感器获取自身姿态，编码器获取电机转速，将所有信息反馈给主控单片机，根据车辆运行状态，合理给出控制，实现车体路径的控制和元素的识别。

系统设计框图 2.1：



2.2 车模布局及参数

智能单车车模整体，包括车模重心，传感器支架固定方式、高度、长度，舵机固定形式，舵机传动方式，编码器安装位置，六轴姿态传感器安装位置等，都对整个智能车系统的稳定运行起着至关重要的作用。

整体布局方式（如图 2.2 所示），布局的目的以及优点：

（1）车模主控板放于车模正中央位置，以降低单车左右车身质量差，增强平衡性，让左右过弯的惯性基本一样。

（2）自购 SASU 的 3s 锂电池安装在车模本身的电池仓里面，这个电池仓位于车模的中心，可以很好的将重心集中在车模的中心。

（3）舵机固定在车模本身自带的舵机固定结构，固定在车头，以连接杆作为传动方式。

（4）六轴传感器安装与车身最低最后的部分，降低车身震动对其的影响。

（5）编码器安装是光电码盘，直接连接在电机上，精度很高。

（6）摄像头传感固定在碳素杆上，底座是自己 3D 打印的，为了防止它晃动，还增加了三个固定的支架。

图 2.2 车模示意图

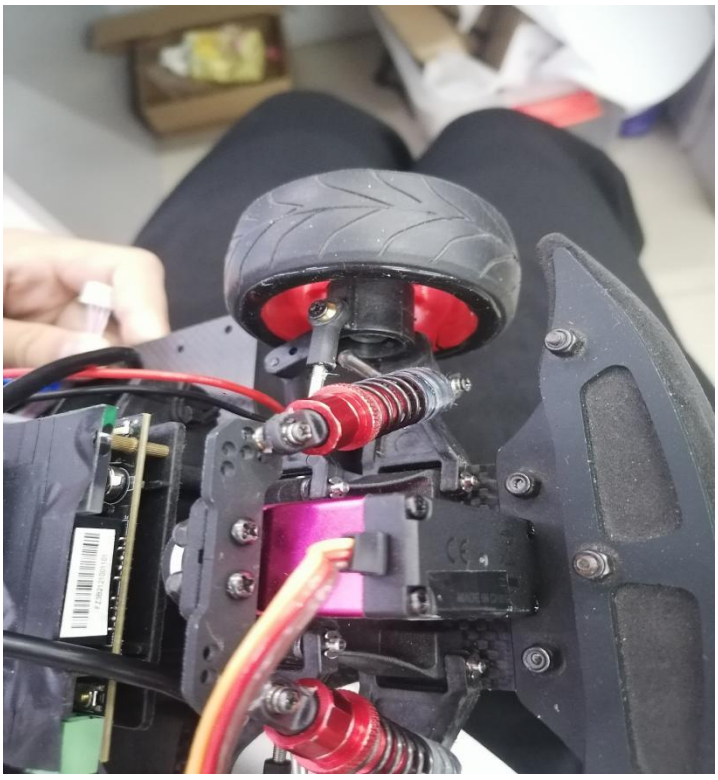
第三章 机械结构

采用本届智能车比赛新加入的 I 车模，通过对舵机安装，悬挂的改动，车壳的改装，传感器安装与选用，车模加固及重心降低等方面对 I 车模的改造进行介绍。

3.1 舵机的安装

其实 I 车模的很多结构都是在身缠时就做好的，也是做成了很适合这个车模本身的结构，就比如舵机，在安装上只需要直接安装在原有的结构上就好了，我们本身需要调节的时传动杆和前轮的灵活程度，先说传动杆吧，传动杆时在拐弯时最重要的结构，因为它决定车模的运行轨迹，必须把它们调到合适的程度时，在将它们调到一样的长度，此外舵机转矩=摆杆作用力*摆杆长度

通过公式可以得出：拉杆作用力越大，反应越灵敏，转向速度越快。而转矩一定时，摆杆越长，作用力反而就越小，故此得出摆杆不需要太长。经过实验，我们最终确定选择舵机连杆长度在 5cm 左右（需要毫米级微调）。



3.2 悬挂的改动

悬挂起到了一定减重的作用，但是在 I 车模中，悬挂却是让车更加的晃动，特别是在过弯时候，速度慢还好，速度一旦快了起来，就晃个不停，此外，I 车模的悬挂上的弹簧还没有什么力，即使将弹力调到最大，也没办法将车尾支撑起来，因此，我将悬挂进行了固定操作，用 3D 打印机打印了 4 个塑料圆柱，将其卡在弹簧下，防止弹簧激烈震动，以达到过弯不晃动的效果。

3.3 车壳的改装

今年的完全模型组要求每个参赛队伍的车必须带着车壳跑，可以用官方卖的车壳，也可以自己设计车壳，但是，因为自己的技术不到位，只能用官方卖的车壳，但是又有一个难点，那就是官方的车壳大小虽然和 I 车模合适，但是不方便安装，也不方便调试，车顶的摄像头孔也很小，每次安装车壳就必须把摄像头拆下来，但是对于智能车来说，摄像头就是命，就是灵魂，一但动了摄像头很多元素就重新开始调。

因此，必须将车壳的一部分做成活动的那种，不仅要方便安装和拆卸，也要方便带着车壳调试。

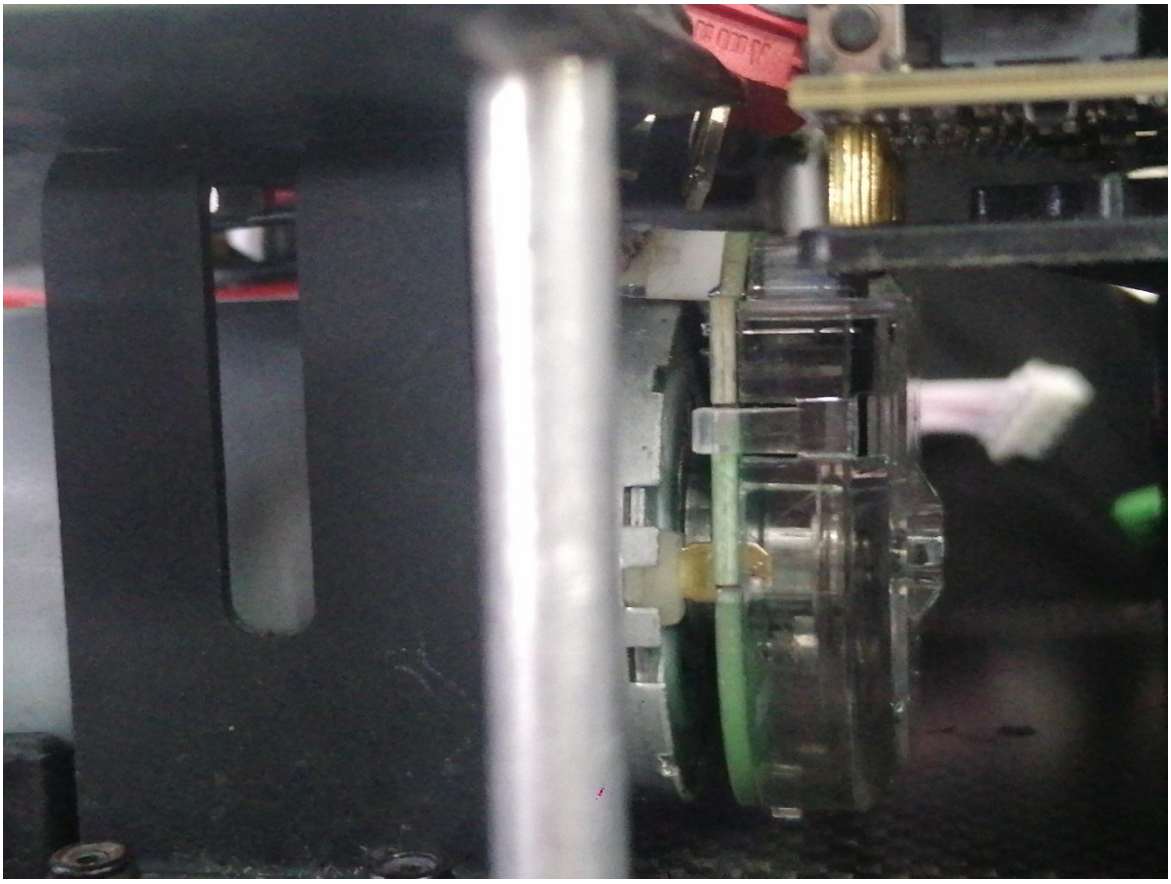
为此我做出如下图的改动：

d

如上图，将车壳的上，左，右都做成了活动的结构，最上面的活动部分，是为了让车壳方便安装，并且不用采写摄像头，左边的结构，是为了方便安装电池，右边的结构，是为了方便按键调参，插线下载以及插网线调试等操作，其中，活动的连接部分是用一种微型的活动页连接完成。

3.4 光栅码盘的安装

我们选择的是 SASU 官方购买的和电机配套的光栅码盘，它安装方便，固定牢固，精度很高，并且，为了防止光线和灰尘的影响，官方还特地制作了一个保护壳，用来隔绝这些影响因素，安装效果图如下：



3.5 摄像头传感器的安装

完全模型组采用的是摄像头循迹，摄像头采集图片，然后用代码进行处理，因此，固定摄像头就成了一件非常重要的事情，因为要得到一个不倾斜的，稳定的图像，才能做好控制，车的姿态才好，元素识别才正常。

为此，我们自己 3D 打印了碳素杆的底座，底座的高度在 5cm 左右，可以很好的固定碳素杆，底座的四周有四个螺丝孔，可以进一步固定碳素杆，防止碳素干晃动，但是因为碳素杆比较高的缘故，在碳素杆的上面存在着惯性，这也导致摄像头还是会有一点轻微的晃动，为此，我们增加了三个专门固定摄像头的碳素杆的支架，效果图如下：

第四章 硬件电路的设计

硬件电路设计在可靠的基础上尽量简单化，满足稳定工作的前提下，保证主控板集中化，一体化，减小单车的负载。电源管理模块保证整个系统供电稳定，在保证传感器采集的信息精确有效的前提下，将电机驱动电路与主控板一体化处理。并且在电机驱动在正常工作的情况下对其他电路的影响和干扰。

4.1 硬件设计方案

整个智能车控制系统由单片机和外围电路构成。为了减小噪声信号对电路带来的干扰，我们将驱动电路的地与控制部分电路的地使用 0 欧姆电阻分隔开。主板的电路主要包含以下部分：电源稳压电路、舵机接口、电机驱动电路、编码器接口、蓝牙串口接口、TFT 屏幕接口、电源接口、指示灯、按键、电源开关、蜂鸣器、拨码开关、核心板接口等。

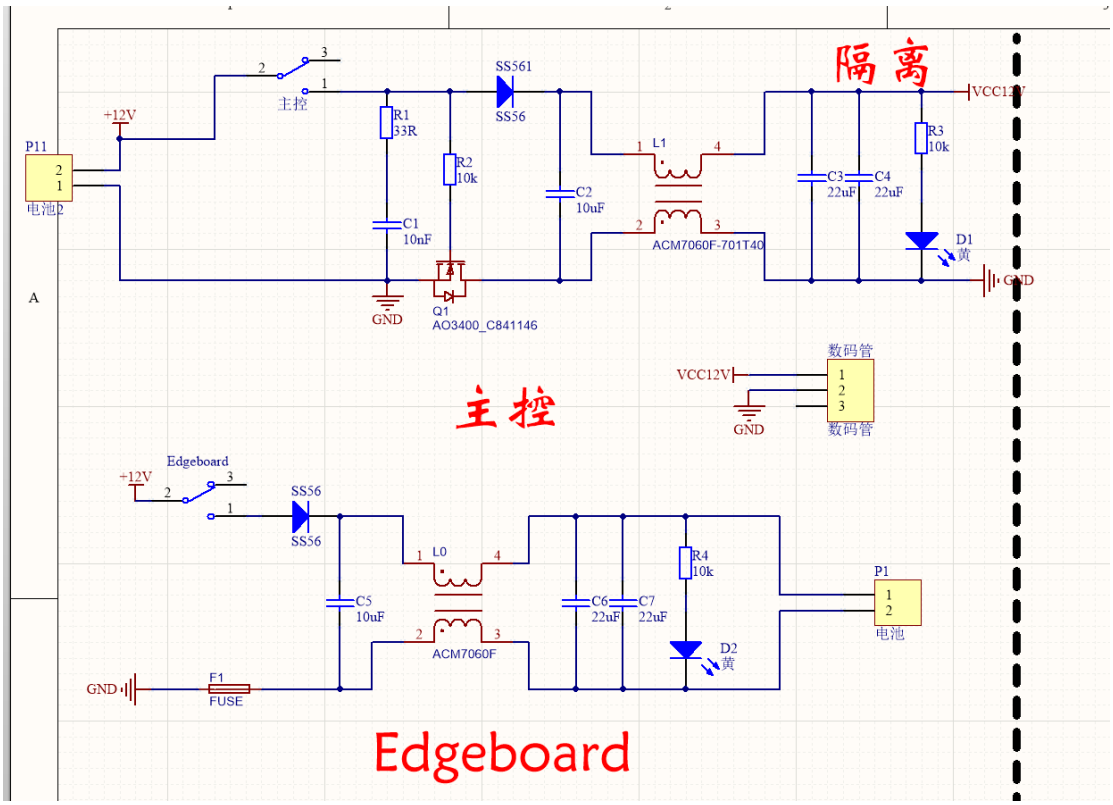
4.2 电源模块

电源模块为小车系统的其他各模块提供所需要的电源,在整个智能车中充当着单片机和外部硬件间信息传递的桥梁，既能把单片机中的程序命令传达给硬件，又能把从外部硬件读到的信号反馈给单片机，是整个智能车的关键。设计中，除了需要考虑电压的范围和电流容量等基本参数外，还要在电源转换效率，降低噪声，防止干扰和电路简洁方面进行优化。可靠的电源方案是整个硬件电路稳定可靠运行的基础。

由于整个小车工作系统的电源均来自一个 3S 锂电池,容量为 2200mA，各个系统中各模块所需要的工作要的工作电压和工作电流都有所不同,因此需要多个不同的稳压电路，将电池电压转换为各个模块所需要的电压，还需要更具模块的功能和用途选择不同类型的稳压模块。

整个车模由两节 18650 锂电池提供电能，配合开关、指示灯以及靠近总电路的滤波电容组成电源模块，下图为 2200mA 的 3S 锂电池和电源模块。

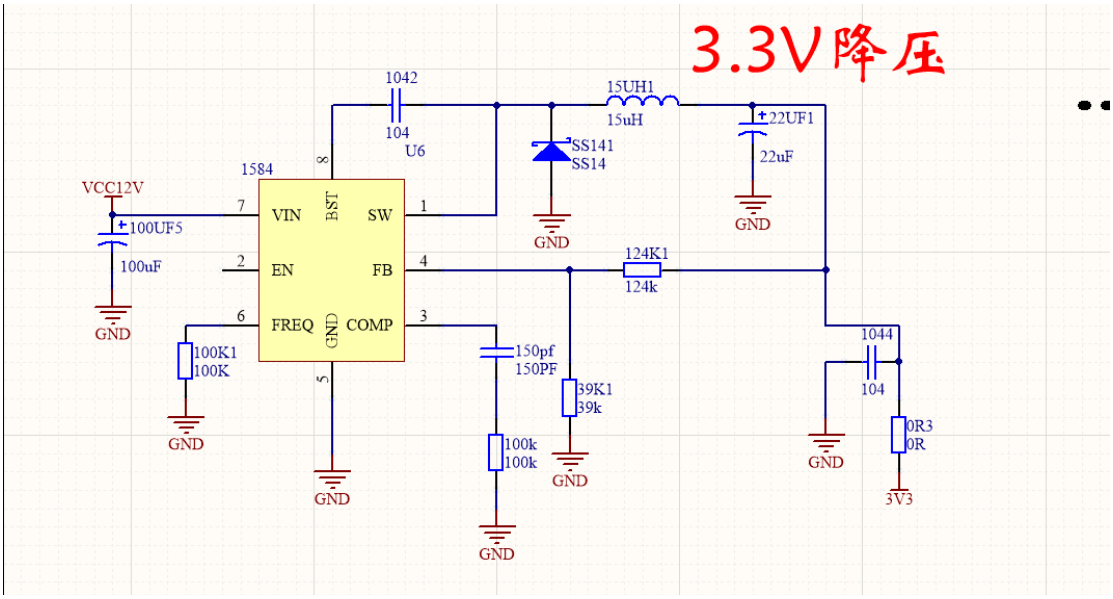

全国大学生智能汽车竞赛专用
高性能锂电池
2200 mAh
11.1V
3S1P 25C 24.42Wh
CB-22003
北京赛曙科技有限公司


隔离
主控
Edgeboard
+12V
P11
电池2
主控
SS561
SS56
R1
33R
R2
10k
C1
10nF
C2
10uF
L1
ACM7060F-701T40
C3
22uF
C4
22uF
R3
10k
D1
黄
GND
Q1
AO3400_C841146
VCC12V
数码管
SS56
L0
ACM7060F
C5
10uF
C6
22uF
C7
22uF
R4
10k
D2
黄
P1
电池
F1
FUSE
Edgeboard

4.3 3.3V 稳压模块

在整个主控模块中，用到 3.3V 电压的有按键，TFT 显示屏，陀螺仪。我们选择采用 MP1584B 低功耗低压降线性稳压芯片，耐压有 28V，标称 3A 输出电流，实测 2A 发热但是没有问题，再大了发热太严重，所以 PCB 设计要注意底部裸铜进行散热处理，估计 2A 以内放心用。空载时电流很小，只有 0.37mA，空载或轻负载时候，输出为锯齿波。29V 输入加上负载芯片就坏了。工业 24V 的环境使用估计有风险,但是用在这里完全够用，因为，这里的输入电压为 12v，输出 3.3v，外设所属要的供电电流也非常的小，完全不会超过 2A，此外还可以调节相关的比例电阻来达到你所需要的输出电压。

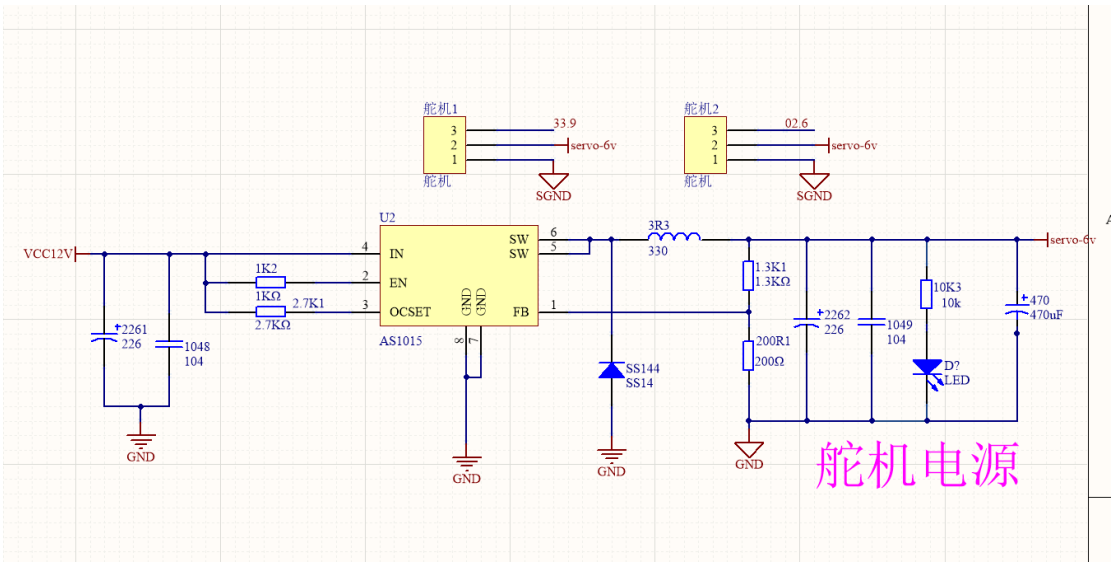
原理图如下：



4.4 舵机 6.7V 稳压模块

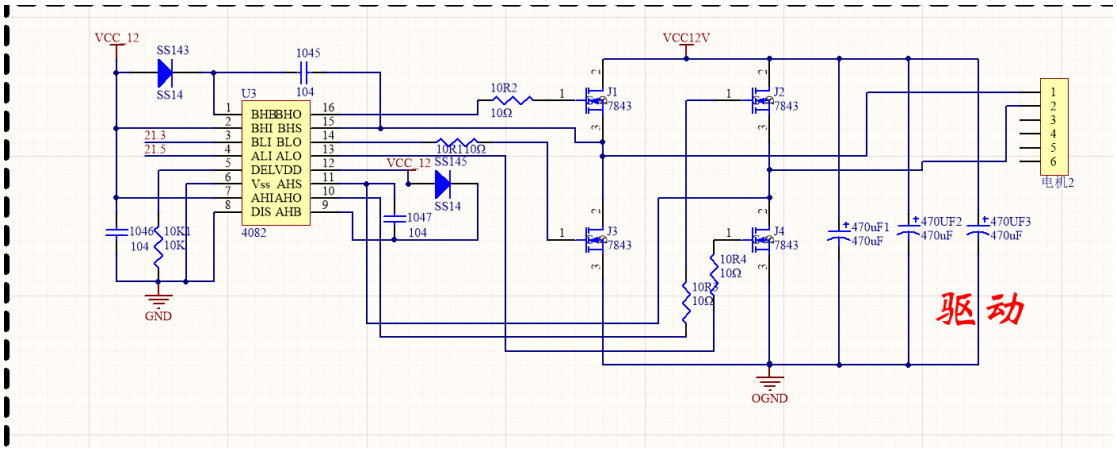
由于完全模型组的舵机是大力矩的舵机，所以所以需要较大的输出功率，较大的电流，所以我们选择独立出舵机的稳压电路模块。而 AS1015 这款电压模式为降压型 DC-DC 变换器，最大可以达到 5A 的输入电流，采用 300KHz 固定频率开关的 PWM 控制方案，输入电压范围从 3.6-23V，且有电压由 0.8V-VIN 的电压输出范围。且芯片带有电流限制和过热保护，短路保护。很适合作为大力矩舵机稳

压电路的芯片。原理图如下：



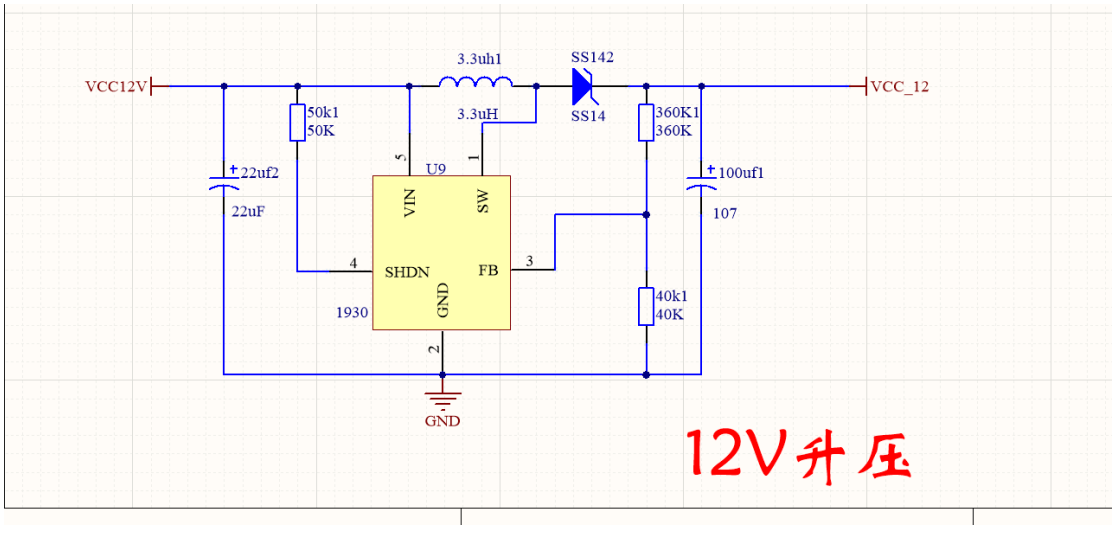
4.5 电机驱动电路模块

单片机能够输出直流信号，但是它的驱动才能也是有限的，所以单片机普通做驱动信号，驱动大的功率管如 Mos 管，来产生大电流从而驱动电机，且占空比大小能够经过驱动芯片控制加在电机上的均匀电压到达转速调理的目的。我们使用的电机驱动电路主要采用 N 沟道 MOSFET 构建 H 桥驱动电路，H 桥是一个典型的直流电机控制电路，由于它的电路外形酷似字母 H，故得名曰“H 桥”。4 个开关组成 H 的 4 条垂直腿，而电机就是 H 中的横杠。要使电机运转，必需使对角线上的一对开关导通，经过不同的电流方向来控制电机正反转。原理图如下图所示：



4.6 驱动电路所需的 12v 升压模块

由于 H 桥驱动电路所使用的 HIP4082 芯片需要 12 的开启电压，所以需要 12 升压电路。我们选择的是 LM1930 升压芯片，LT1930 是业界最高功率的 SOT-23 开关稳压器。它均包含内部 1A、36V 开关，从而允许在很小的电路板占位面积上产生大电流输出。LT1930 在 1.2MHz 频率下开关，允许使用小型、低成本和高度较低的电容器和电感器，性价比较高，原理图如下：

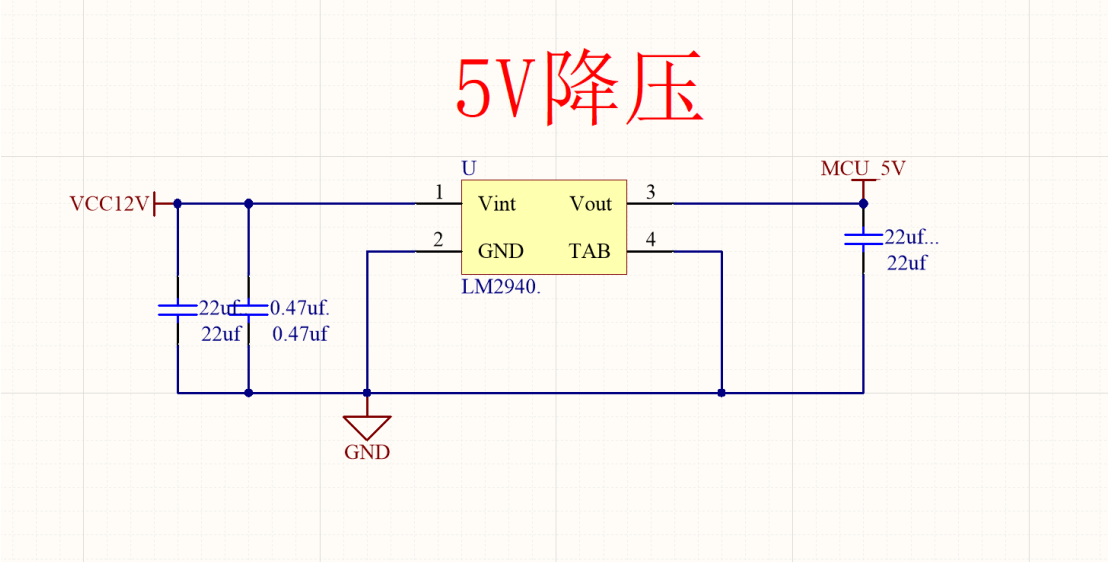


4.7 5V 将压模块

在英飞凌主控板上，使用 5v 模块的主要有 264 单片机、蓝牙模块，编码器模块，特别是 264 英飞凌单片机，所以 5v 模块最为重要，因此，我们选用的是

LM2940 芯片，此外，为了保护单片机，我们选择将单片机和其它外设分开，为此，我们准备了两个 5v 模块，分别给它们供电，排除了外设对单片机的干扰。

具体原理图如下：



4.8 按键与拨码开关模块

现场调试过程中，往往需要根据实际情况对各种参数进行调整和修改，因此在主板上加上按键和拨码开关是十分必要的。原理图如下图所示：

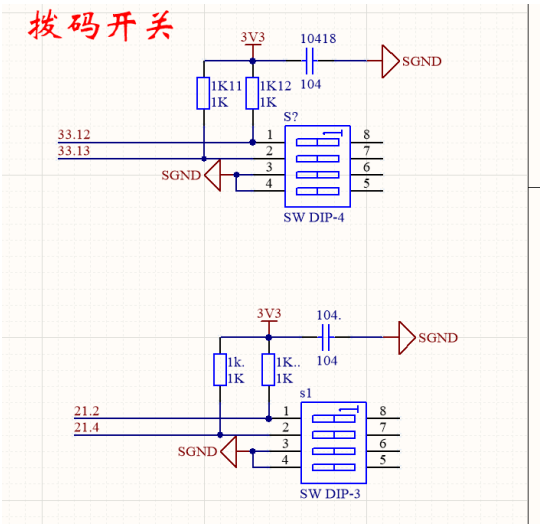


图 4.8.1 按键原理图

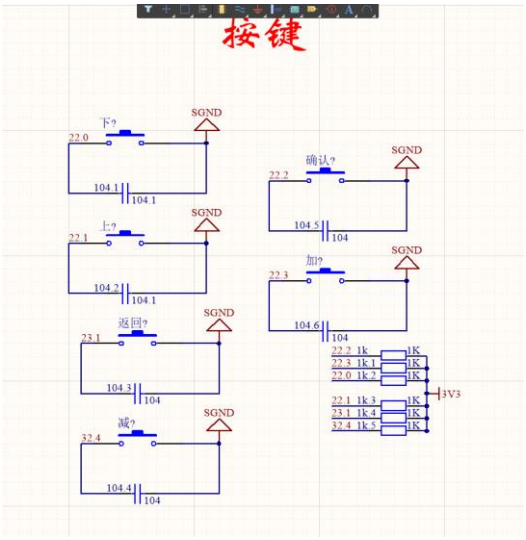

按键
下?
22.0
SGND
104.1
104.1
上?
22.1
SGND
104.2
104.1
返回?
23.1
SGND
104.3
104
减?
32.4
SGND
104.4
104
确认?
22.2
SGND
104.5
104
加?
22.3
SGND
104.6
104
22.2 1k
22.3 1k.1
22.0 1k.2
22.1 1k.3
23.1 1k.4
32.4 1k.5
1K
3V3

图 1

4.9 蜂鸣器

蜂鸣器和指示灯可以帮助调试过程中知道程序的运行状态。原理图如下图所示：

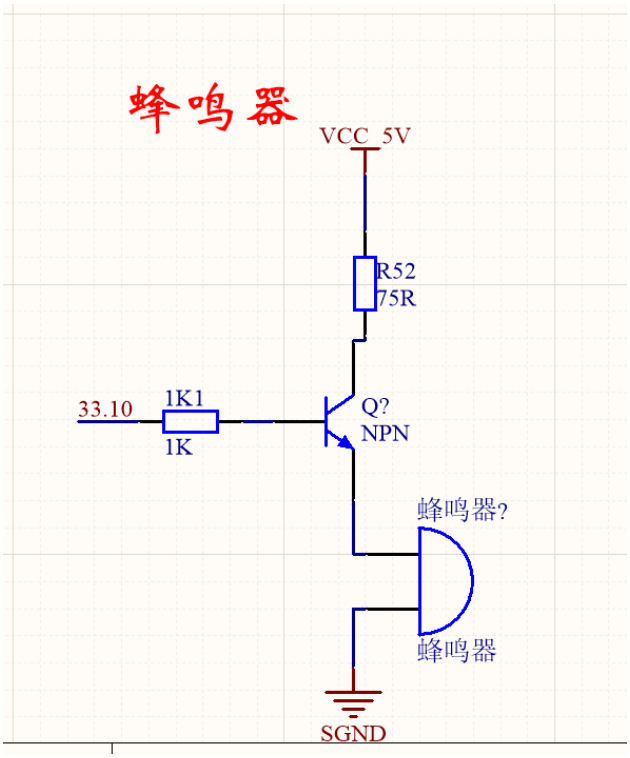


图 4.9 蜂鸣器原理图

4.10 单片机电路

根据需要引出适量引脚。如下图所示：

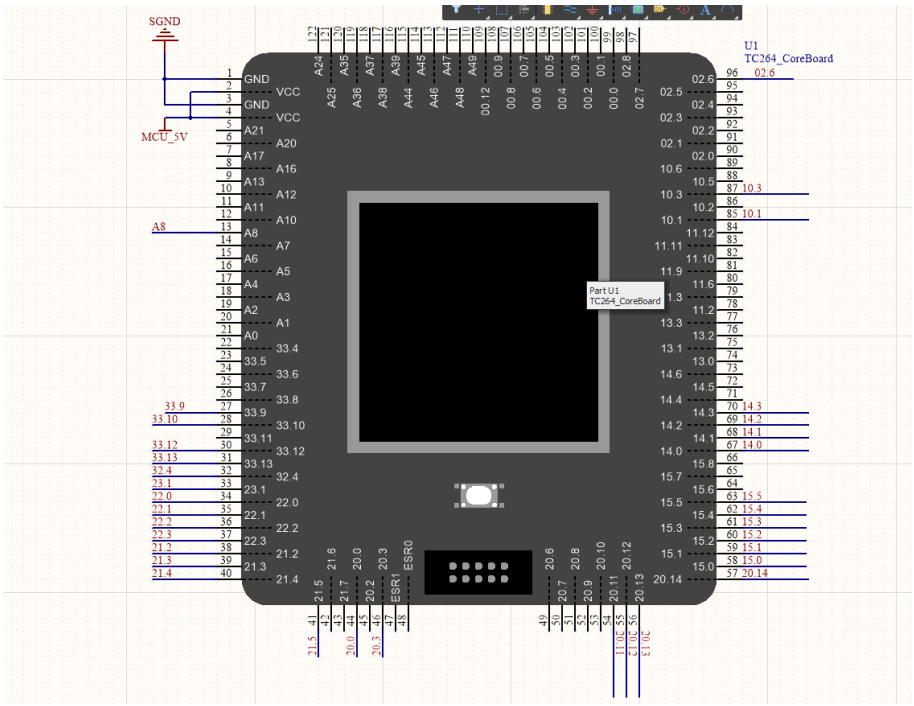


图 4.10 单片机电路

4.11 处理图像的计算卡模块

完全模型组的摄像头必须插在计算卡上，由计算卡处理图像和模型，计算卡外形图如下：

图 4.11 计算卡模块

4.12 usb 拓展坞模块

因为计算卡和外部硬件交互的主要接口是 usb 接口，但是它的 usb 接口只有两个，完全不够用，所以只能用 usb 拓展口来拓展 usb 口，然而用现成的的 usb 拓展口不太方便安装车壳，因此我们选择自己设计 usb 拓展口，原理图如下：

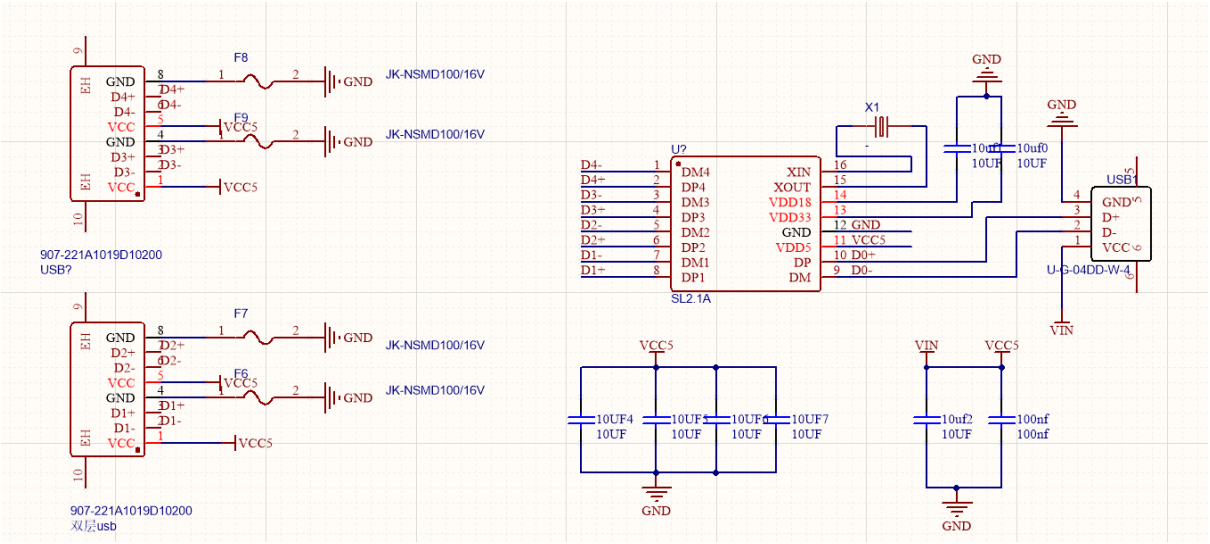


图 4.12usb 拓展坞原理图

4.13 调试工具

在整个电路系统中，除上述几个模块外，为了方便调试，我们还加入 TFT 屏、按键、蓝牙等模块。TFT 屏和键盘使调车人员能对小车参数直接观察和修改；蓝牙模块则可将小车行进状况有关参数发送到上位机中，使调车人员可以实时监测小车的情况，方便调试。且只需要用核心板上的串口模块，方便快捷。


1.8'128x160 RGB TFT LCD

第五章　任务标志检测

本届新引入的组别完全模型组相对于传统的竞速组别而言具有更加复杂的机器视觉处理和模型 车运动控制任务。完全模型组比赛赛道以室内循环赛道为基础，赛道材质，赛道规格均保持一致。在导引方式上完全保留室内循环赛道的导引方式，并在此基础上添加完全模型组任务导引标志和锥桶，引导车模完成完全模型组赛道任务。

5.1 数据采集

为了引导比赛任务的完成，在比赛赛道的任务元素和特殊元素区域的前方指定的区域贴有固定的地面标志。标志的整体外框尺寸为 16cm×16cm 的正方形，标志颜色为红色，正方形内多余的灰色区域裁减掉，按照赛道任务和元素贴在赛道的指定位置。

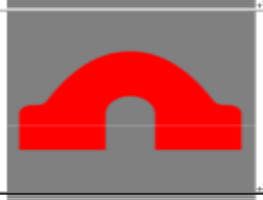
序号	名称	说明	图示
1	泛行区标志	表示前方三岔路口围成的泛行区域，内部区域包括蓝色底布均可行驶	
2	禁止通行标志	放置在泛行区域进出口连线上，车辆需要绕过此标志进行通行。（此标志高出距离地面有 2cm 高度其余均紧贴）	
3	施工区标志	表示前方为施工区，需要绕行赛道外障碍桩围成的临时路段	
4	坡道标志	表示道路前方有坡道	
5	加油站标志	表示前方为加油站，车辆需要驶入加油站并按照指定的出口驶出加油站	
6	加油站出口数字标志	加油站设置有“1”和“2”两个出口，并在出口地面贴有对应的“1”和“2”数字标志。 比赛时加油站的入口处会随机放置车辆需要驶出时的出口数字。	

图 5.1.1 任务标识

为了能够完成对应的标志任务，采用开源深度学习平台飞桨完成模型的训练、推理和部署。为保证模型的精度需要提供更加真实的训练数据，所以在进行数据采集的时候应当尽可能的模拟车子行进过程的轨迹姿态。

图 5.1.2 部分数据集图片

采集完数据后，通过 labeling 软件对拍摄的数据集设置标签(例如图 5.1.4 中的标签是 shigong)然后再根据标签依次对每一张图片进行图片标注，生成对应的 xml 文件。

图 5.1.3 数据集制作软件 labelImg

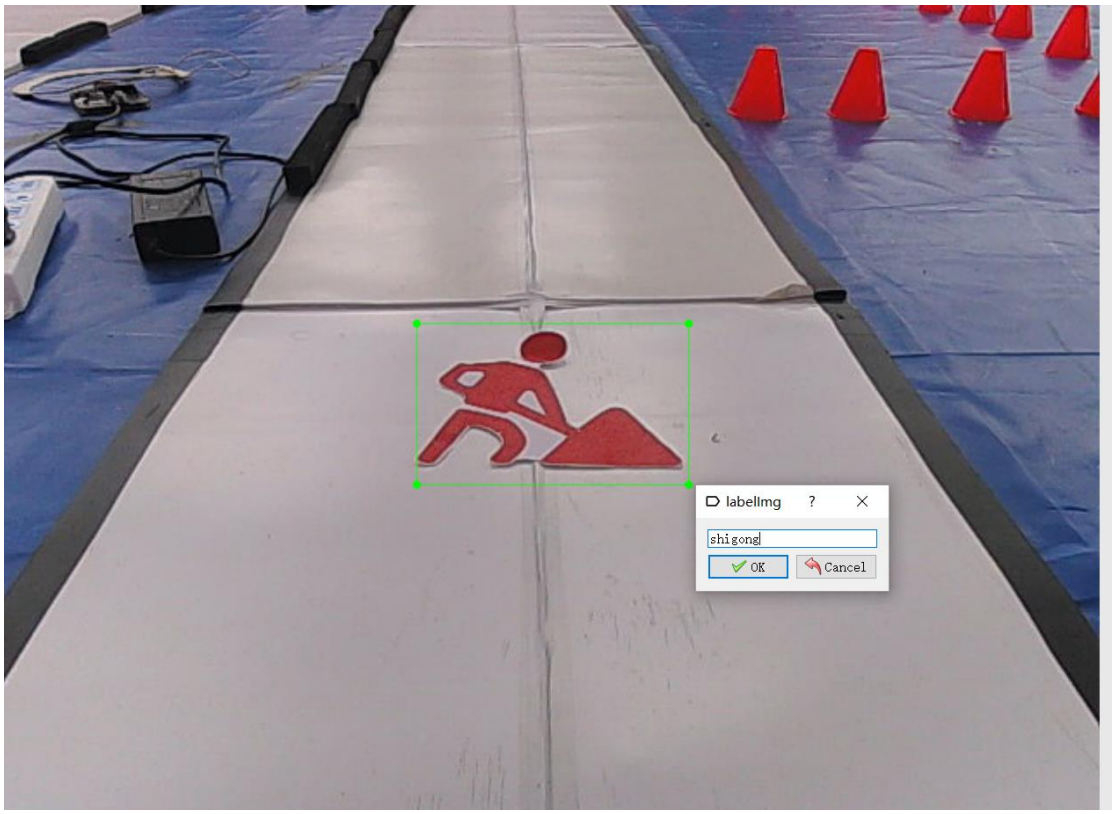


图 5.1.4 图片标签的设定

最后对其进行整合完成对相应标志的数据集的制作，然后将整合完之后的数据集上传到飞桨平台上，通过更改模型的训练集和测试集进行调用。

5.2 模型框架与训练

对于模型的训练是采用开源深度学习平台飞桨完成的。整体的模型框架是基于 PaddleDetection。

PaddleDetection 为基于飞桨 PaddlePaddle 的端到端目标检测套件，内置 30+模型算法及 250+预训练模型，覆盖目标检测、实例分割、跟踪、关键点检测等方向，其中包括服务器端和移动端高精度、轻量级产业级 SOTA 模型、冠军方案和学术前沿算法，并提供配置化的网络模块组件、十余种数据增强策略和损失函数等高阶优化支持和多种部署方案。

图 5.2.1 飞桨平台

在整体的模型框架搭建完成后，导入之前制作好的数据集进行训练，然后根据实际的情况来对模型的参数进行调整（就学习率而言，作为优化算法中更新网络权重的幅度大小，当学习率过大则可能导致模型不收敛，损失 loss 不断上下震荡；学习率过小则导致模型收敛速度偏慢，需要更长的时间训练。对于批次大小 batch_size 和迭代次数，应当考虑到内存资源的限制和测试错误率，尽可能的避免出现内存崩溃，过拟合等情况的出现。），以达到更高的精度。

5.3 模型部署

模型的部署首先要将要将训练好的模型从平台上导出，在 PaddleDetection 文档中就有相关的模型导出方法，主要是通过调用 export_model.py 文件实现将模型导出分成三个文件_model_ ， _params_,infer_cfg.yml，如图 5.3.1

图 5.3.1 模型导出文件

然后根据训练模型所使用的 label_list.txt 文件，修改板卡上所对应的 label_list.txt 文件中的标签，再运行模型进行验证。

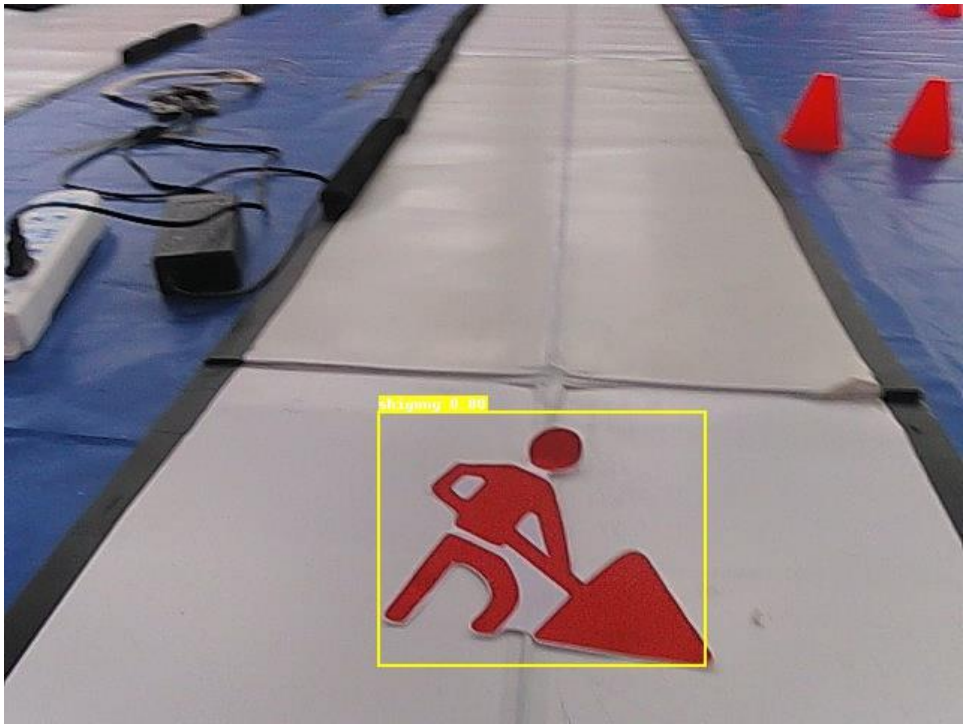
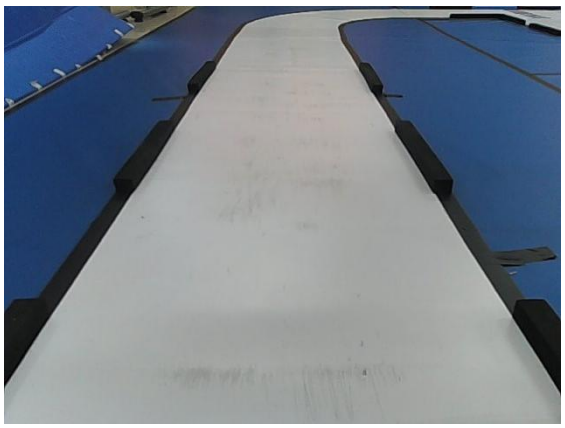


图 5.3.2 部分图片验证

第六章　图像巡线处理及元素思路

一、 图像巡线处理

在正式处理图像之前，需要特别注意的是摄像头的对正处理，摄像头是车子的眼睛，如果摄像头歪了，车子也不会跑正，更不可能跑好，所以我们在摄像头的对正方面下了些功夫，经过不断的拆、拧，最终将摄像头对正，达到了如下图的效果：



我们的摄像头采集到的是一幅320＊240的RGB图像，初次接触彩图的处理，让我们显得有些不知所措，Opencv 库的加持更让我们对繁多处理方式产生了疑惑。在尝试、比较了一些方法（比如将 RGB 图转成 HSV 图提取颜色）后，我们还是选择了比较传统的处理方式：将图像转成灰度图或者二值图处理，并在此基础上寻找边界，主要有以下两个思路：

1、大津法二值化：

大津法(OTSU)，又叫最大类间方差法，基本原理是寻找某个阈值，将图像分割为前景和背景，使得前景和背景的类间方差最大。

大津法的优点是简单、阈值动态，能较好适应各种场地，我们一开始为了快速上手，使用了一段时间的大津法，二值化后的图像边界很好找，白黑跳变就是边界，但是大津法的缺点是不能很好适应光线不均的情况，也就是不抗反光，为此我们还是选择在灰度图上来寻找边界。

2、Sobel 算子

Sobel 算子是一种一阶微分算子，常用于边缘检测领域，传统的 Sobel 算子使用水平（0 度）和垂直（90 度）两个方向的卷积模板来对图像进行卷积运算。

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \qquad \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

0°边缘方向　　90°边缘方向

Sobel 算子的优点是它是依靠周围八个像素点的灰度值乘上模板对应的权重算出一个梯度值，可信度高，不容易找错边界且抗光性好，但是缺点是如果对全图进行 Sobel 卷积运算，时间开销大，且得出的梯度值没有方向，如果没有处理得当，可能会出现把左边界找成右边界的情况。所以在找边界时，先比较灰度值，在靠近边界时再使用 Sobel 算子进行卷积运算，将算出来的梯度值与提前设定好的梯度阈值比较，就能又快又准的判断边界。

在查找相关文献时，我们得知 Sobel 算子实际上有 8 个方向的模板。

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

0°　　45°　　90°　　135°

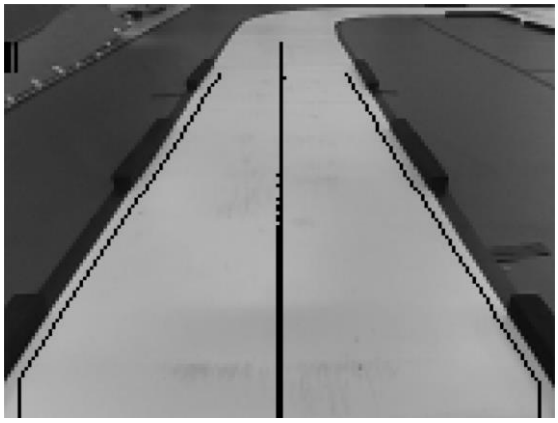
$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}$$

180°　　225°　　270°　　315°

同时我们也注意到由于摄像头的特性，我们的边界是斜向上延伸的，所以我们使用了非传统的 Sobel 模板：一个 45 度和一个 135 度的模板，取得了比传统

Sobel 模板更好的效果。

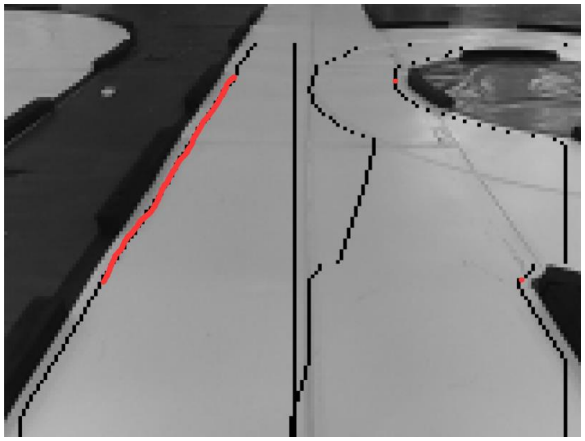
最后寻得的边界如下图所示：



二、元素思路

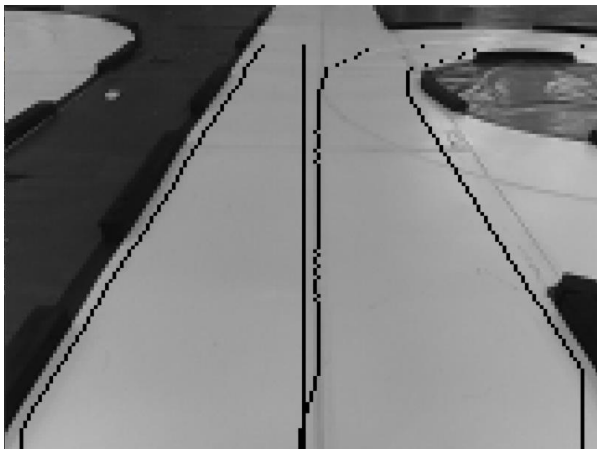
1、圆环处理

圆环这个元素年年都有，年年都是很难的元素，我们也在圆环处理上花费了不少功夫。以右圆环为例，核心的识别思路是找到右边界第一个拐点，第一个拐点上方丢线，丢线上方存在第二个拐点，第一个拐点到第二拐点间的左边界都没丢线。

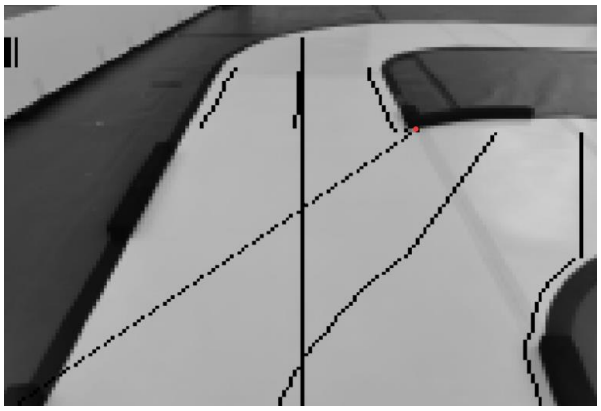


我们把圆环大致分成了 5 个阶段处理，分别为入环走直、入环、环内循迹、出环、出环走直。

入环走直：在入环前，由于有一大段丢线的边界，所以如果不做处理的话入环前会扭，为保证较好的路径，需要将右边界拐点相连。

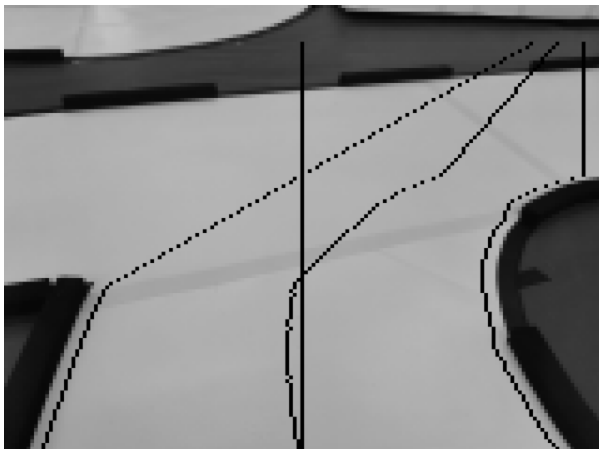


1）入环：入环时靠近环口，将图像最左下角和右上方拐点相连作为左边界即可，待右上方拐点消失，进入环内循迹。



2）环内循迹：环内只需正常循迹就行。

3）出环：出环将左下拐点和右边界上方点相连作为左边界，待左下拐点消失就出环成功。



4）出环走直：出环后，右边界还是有一大段丢线，做入环走直阶段类似处理即可。

2、十字处理

1 第一种情况：正入十字，并且下拐点很清楚而且在该拐点的下方边界很清晰，能够准确的计算出斜率等的信息

2 第二种情况：找不到下拐点，并且存在丢线的情况，意味着已经进入到十字里面了

3 第三种情况：斜入十字比较严重（这个部分放在判断里面）

特征 1：一边丢线，从连续的丢线的地方一直向上直到这个地方丢线结束的这一段区域内，另外的边线一定会出现先变大后边小的情况，可以对变化的范围进行一个限制，在这个限制范围内一定会出现拐点,这个拐点就是十字的下拐点，这个拐点的特征理论上可以将圆环和十字的特征区别开来，并且这个拐点必须是连续的不能存在非常大的跳变。特征 2：拐点的该行算出来的中线一定是白色（防止误判的条件）

4 备注：补线就要两边一起补起来一定不要存在只补一边的情况，遇到这种情况可以用单边循迹拟合出另外一条边界信息（这个部分放在最后的合理性检测里面）

十字的全部流程：

观察发现：十字主要分成三种情况：

第一种：是下拐点十分清晰的可以找到，并且在下拐点找到的时候，下拐点往下存在的边界的点非常清晰和准确，那么可以根据这些点计算出这条边界的表达式。

判断方法：这种情况下一定会出现边界两边同时丢线的情况，可以根据这一点判断是不是有可能是十字（这一部分放在 CrossPreJudge 里面）

补线方法：1、找到单边第一个丢线的地方（一定要在某一个行往上在开始寻找，这路取的行数是第 100 行，因为最后一行的赛道宽度比较大，有可能在转弯的时候就会出现第一个丢线的情况）

2、在找到的单边第一个丢线的地方相向下寻找最值（左边线是最大值，右边线

是最小值），在这个最大值的地方再写个循环再向下寻找有效的 10 个点，7 个点以上是连续的（相邻的差值不超过 3）可以认为找到的这些点是精准的，通过这些点利用最小二乘法计算直线并拟合出直线。

3、备注：目前寻找有效的点数的有一点 bug：判断是否是连续的情况，可能会出现这一部分边线没有准确的寻找出来，一部分是丢线并且是连续的，

虽然跟上一个点不是连续的，但是跟下一点是连续的，那么采集到的数组就是不精准的。遇到这种情况有下面的三种办法进行处理：

（1）更改基础巡线的方法，让边界拟合的时候更加精准一点，写一些反光的算法

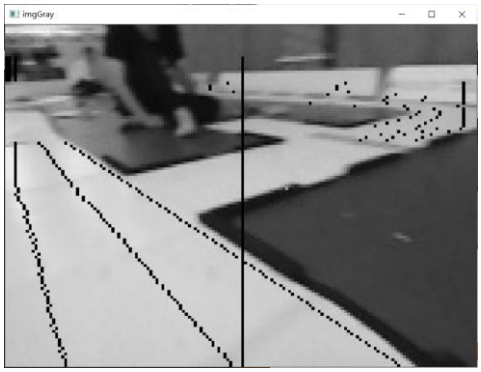
（2）对寻找数组的方法进行更改

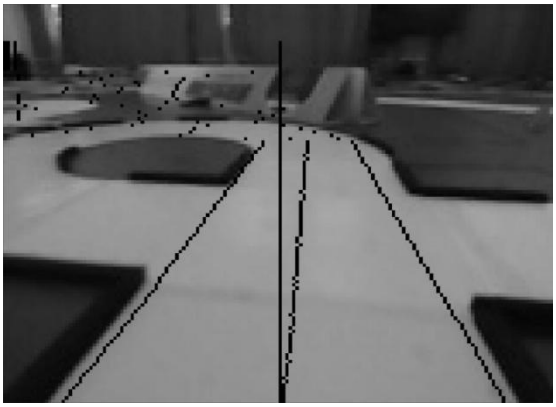
（3）对计算出来的边界的斜率进行限制幅度，保证斜率是在一个正常的范围里面。

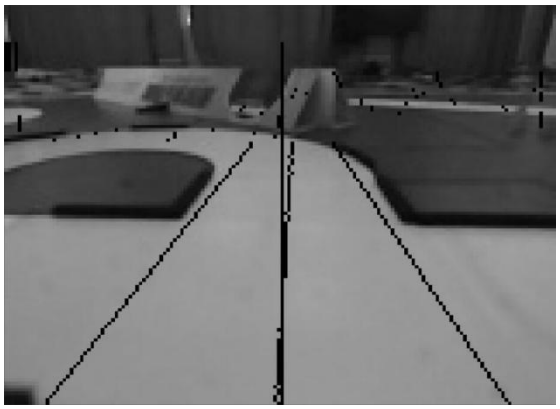
第二种：下拐点找不到，上拐点找的很清楚，一般出现再进入十字里面，下拐点消失的情况

判断方法：和第一种方法的 CrossPrejudge 的方案是一样的。

如下图，是十字的图片：







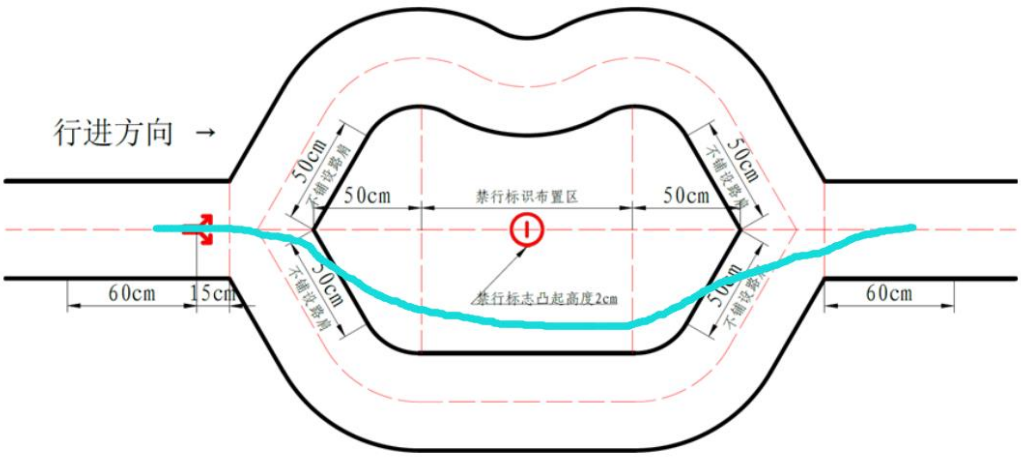
3、坡道处理

当目标检测模型检测到坡道标志后，进入坡道处理阶段。我们的处理方式较为简单，识别到坡道后，降速、降低有效行一段距离，恢复正常循迹即可，需要注意在坡上图像很乱，最好把元素的识别都关掉。

4、泛行区处理

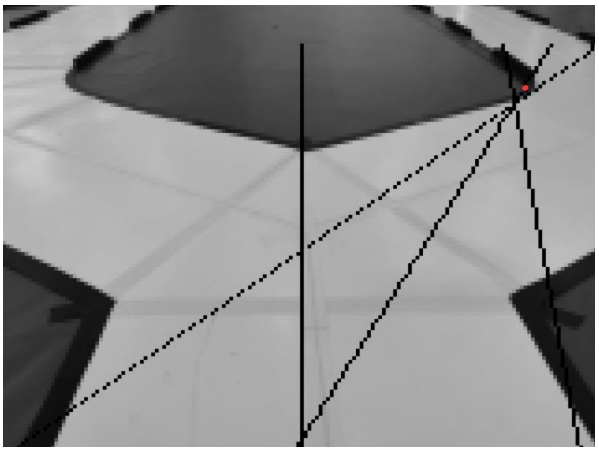
完全模型组的三岔和别的组别有很大差别，可以选择从中间的蓝布区域走也可以选择正常从赛道走，为让车子走最优路径，我们在泛行区的处理上花费了非常非常多的时间，可以这么说，我们在泛行区上花的时间是最多的，起码有花在圆环上的时间的两倍，在处理泛行区的那段时间，是我们做智能车最痛苦的时间，前前后后换了十几种方案，想各种各样的“歪门邪道”，最终确定了方案。

我们的思路是由于禁行标志放置位置未知，所以我们没有选择识别到然后绕开，而是直接“贴着”一旁的赛道走，这样风险小，路径好。



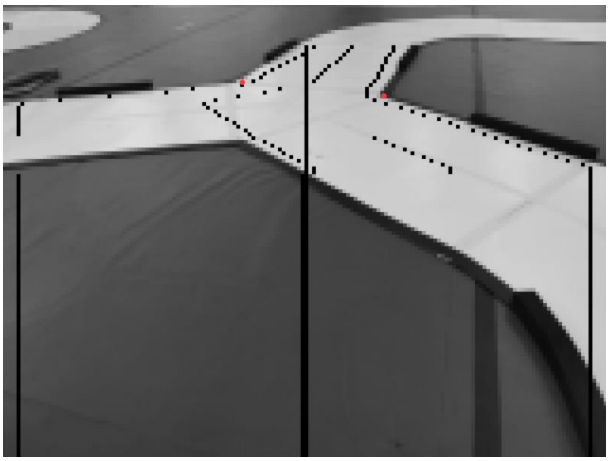
我们将泛行区大致分为了 3 个阶段：进泛行区、泛行区内寻线、出泛行区。

1）进泛行区：找三岔路右侧上方拐点，直接补线。待图像下方黑点数很多时就判定进入了蓝布，进入寻线阶段。

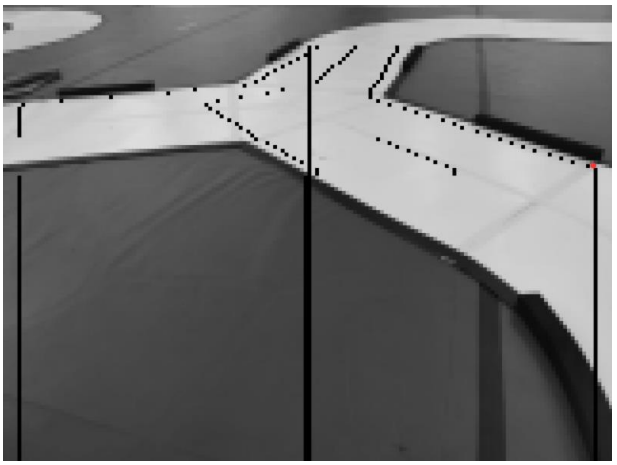


2）泛行区内寻线：在蓝布内寻线需要注意的是判断边界时，是从黑跳变到白，把右边界找到后，以此一边的边界来算出误差，达到实现循迹效果。

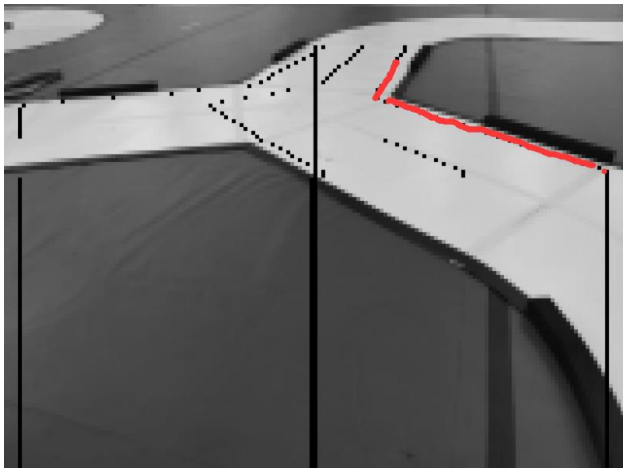
3）出泛行区：待到时机合适，开启正常循迹，可以看到现在的边界还是比较乱的，左右大部分都丢线了，为保证顺利出去，我们需要找到出口处的两个拐点。



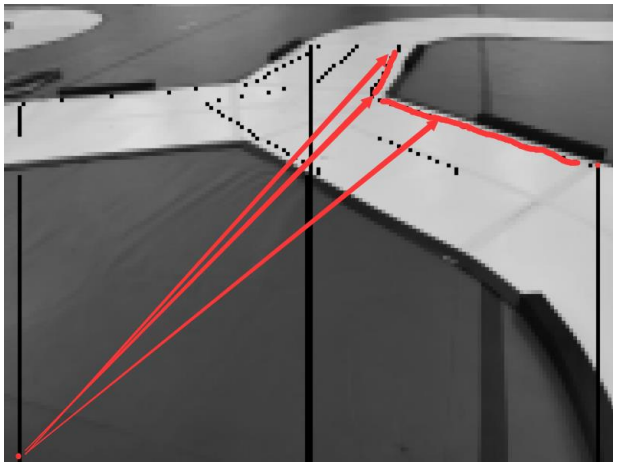
以找右上方拐点为例，沿着右边界找，找到第一个白黑跳变点，记为Start_Point。



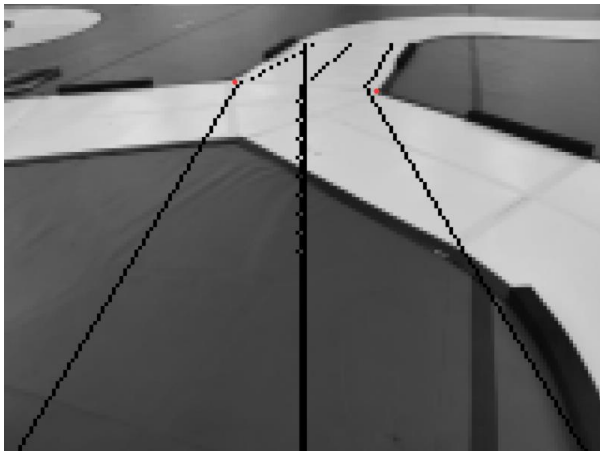
遍历 Start_Point 上方所有的边界点，求每个点到最左下角点的距离，存入数组中。



再对数组进行一次排序，距离最近对应的点就是我们要找的拐点。



我们找到两边的拐点后，直接补线，当图像下方白点数很多时，就算成功出去了。



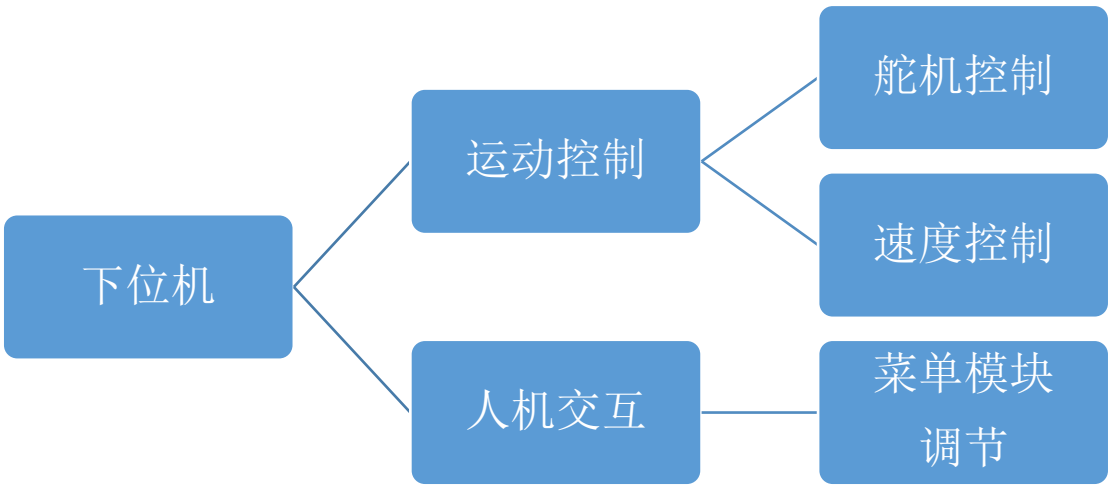
5、施工区和加油站处理

施工区和加油站这两个元素很像，可以说加油站是施工区的 Plus 版。同样，我们在施工区的处理上也花了大把大把的时间，但是我们不得不承认，虽然花了很多时间也想了很多方法，但是到最后也没有找到一种稳定的处理方式，不是进去容易撞锥桶就是出去容易撞锥桶，即便降速也无法很好避免。所以很遗憾，受限于自身的水平，不对这两个元素的方案做赘述。

第七章　车模下位机控制

7.1 车模下位机控制思路

承接第六章的内容可以知道，完全模型的下位机（英飞凌主控）的功能模块比较单一，主要是接受来自上位机的一些标志位和误差部分，根据接收到的数据进行相对应得控制处理。



7.2 人机交互：

下位机的最为常见的人机交互的模块就是通过 TFT 显示屏幕，显示出你要调节的一些参数，我们一般称之为菜单模块。菜单的思路不做详细的介绍。菜单一般需要以下的几个功能，显示出变量的名字和数值，选择变量，更改变量的参数，一般是加减。

菜单的全部的参数一般在 TFT 显示屏幕中是没有办法全部显示出来的，因此我们需要创建二级页和三级页，但是考虑到一般参数不会很多，因此一般到二级页就够了。下位机的参数相对比较少一些，我们就放置了舵机 pid 的参数，速度环控制的参数，车库控制的参数，以及两个 ai 元素（加油站和施工区）的参数。

说到人机交互这一部分，在这里顺便把上位机的人机交互也写出来。我们上位机的人机交互主要有三个部分，一个部分是利用 edgeboard 的优越的性能，我们可以通过无线 WiFi 模块进行图传，并且为了后续方便调试，我们可以直接将运行过程中的一些图片直接保存在 edgeboard 上面。第二个部分就是我们可以通过一些软件，输入指令进行上位机程序的运行。比如我们采用的是 SSH 和 WiFi 连接的方法，通过软件 MobaXterm_Personal_22.0 建立电脑和板卡之间的通信。第三个部分就是我们可以直接在 edgeboard 上面连接显示屏。通过显示屏也可以看到可视化的图形界面。

7.3 上下位机通信模块：

我们采用的是 USB 转串口的方法进行上下位机之间的通讯，上下位机之间的通讯也是很简单的。能够传输 0-255 的数据，然后自己写一个简单的通信协议，保证上下位机能够传输简单的数字，并对这些数字进行划分，特定区域的数据有特定的含义基本能实现上下位机的通信要求。

最近交流过程发现办卡上面有自带串口模块，也可以直接通过杜邦线直连进行通信。

7.4 运动控制：

我们采用的是方案是传统的智能车竞赛用的方案，就是位置式的 pid，舵机的 pwm = 舵机中值 + P*err + D*(err_last － err_now)，或者可以进行一些改进采用动态 pid,由于我们这个组别的速度并没有那么快，我们目前不采用动态 pid 进行控制。

关于速度环方面我们采用的是增量式的 pid 进行速度闭环的控制，但是由于这个车模采用的是差速齿轮，并且差速基本上无法通过代码进行控制，因此速度环控制这一部分也是相对比较简单。

速度环调试参考资料：【转载】狂暴战车 直流电机转速闭环，增量式 pid 调试过程_cbirdfly.的博客-CSDN 博客_增量式 pid 调参

7.5 特殊元素的处理：

加油站和施工区都可以通过编码器计数和打角来进行姿态控制，并且相对稳定，但是实际运行的过程中发现需要调节的参数比较多和繁杂，需要花费相对长的一段时间进行调试。也可以采用图像处理发送误差来进行处理，缺点是图像相对复杂情况种类比较多。

第八章　edgeboard 开发工具

8.1 网口通讯：

1.1 调整电脑 IP

（1） 将电脑与 EdgeBoard 使用网线直连。（2） 进入“控制面板”→“网络和 Internet”→“查看网络状态和任务”。 （3） 点击“以太网”。 3 （4） 点击“属性”。 （5） 点击“Internet 协议版本 4”。 4 （6） 设置电脑 IP 地址为“静态 IP 地址”，点击“确定”。 5 （7） 关闭所有窗口。

1.2 连接通讯

1.2.1 Windows PowerShell

（1） 打开“Windows PowerShell”。

（2） 输入“ssh root@192.168.1.254”（修改成本机板卡 IP），按下回车键。

（3）输入“yes”，按下回车键（非第一次省略此步骤）。（4）输入“password：root”（password 为不可见字符），按下回车键即可进入 EdgeBoard 系统。

1.2.2 FinalShell 安装流程

（1） 下载 FinalShell 安装包。 （下载地址：http://www.hostbuf.com/downloads/finalshell_install.exe）

（2） 下载完成后双击运行“finalshell_install.exe”。

（3） 点击“我接受”。

（4） 点击“下一步”。

（5） 使用默认文件夹或自定义文件夹，点击“安装”。

（6） 若电脑没有安装过 Winpcap，点击“确定”（已安装忽略 6~11）。

（7） 点击“是”。

（8） 点击“Next”。

（9） 点击“I Agree”。

（10） 点击“Install”。

（11） 点击“Finish”。

（12） 点击“关闭”。

使用流程

（1） 打开“FinalShell”。

（2） 如果弹出询问接入网络对话框，点击“允许”。

（3） 点击左上角的文件夹图标。

（4） 点击最左侧的添加图标。

（5） 点击“SSH 连接”。

（6） 按图示输入信息。

（7） 点击“确定”。

（8） 双击“小车”。

（9） 根据使用需求点击“只接受本次”或“接受并保存”。

（10） 成功进入 EdgeBoard 系统。 功能区介绍 （1） 文件系统 （2） 命令系统

8.2 串口通讯：

2.1 环境搭建

（1） 将电脑与 EdgeBoard 使用 USB 线直连。

（2） 下载驱动 CP210x_Windows_Drivers。

下载地址：https://cn.silabs.com/developers/usb-to-uart-bridge-vcp/drivers

（3） 解压 CP210x_Windows_Drivers.zip。

（4） 双击运行“CP210xVCPInstaller_x64.exe”。

（5） 点击“是”。

（6） 点击“下一页”。

（7） 选中“接受协议”，点击“下一页”。

（8） 点击“完成”。

（9） 进入“控制面板”→“硬件和声音”→“设备管理器”。 记住端口号，此处为 COM8。

（10） 下载 MobaXterm。 （下载地址：https://mobaxterm.mobatek.net/download.html）

（11） 解压 MobaXterm_Portable_v21.5.zip。

2.2 连接通讯

（1） 双击运行“MobaXterm_Personal_21.5.exe“（如弹出防火墙，允许访问）。

（2） 点击“Session”。

（3） 点击“Serial”。

（4） Serial Port 选择 2.1 第（9）步记住的端口号，Speed 选择“115200”；点击“Advanced Serial settings”，Flow control 选择“None”，其余默认 不要修改； 点击“OK”。

（5） 成功连接板卡。

（6） 重启板卡，可收到板卡上报消息。

（7） 输入 root（用户名），按下回车键，再次输入 root（密码，字符不可见），按下回车键即可进入设备系统。

8.3 VNC 使用说明：

3.1 下载安装

（1） 下载 VNC Viewer。

（下载地址： https://www.realvnc.com/en/connect/download/viewer/windows/）

（2） 双击运行 VNC-Viewer-6.21.1109-Windows.exe。

（3） 选择“English”，点击“OK “。

（4） 点击“Next”。

（5） 接受协议，点击“Next”。

（6） 使用默认文件夹或自定义文件夹，点击“Next”。

（7） 点击“Install”。

（8） 点击“是”。

（9） 点击“Finish”。

3.2 连接使用

（1） 将电脑与 EdgeBoard 使用网线直连。

（2） 打开 VNC Viewer。

（3） 右键空白区域，点击“New connection”。

（4）输入 IP:端口(默认开放的端口号为 5902,此处为 192.168.1.12:5902)；输入自定义名称（此处为小车）；点击“OK”。

（5） 双击左上角的窗口（小车）。

（6） 若弹出提示框，点击“continue”。

（7） 输入密码：edgeboard，点击“OK”。

8.4 外接显示器使用说明：

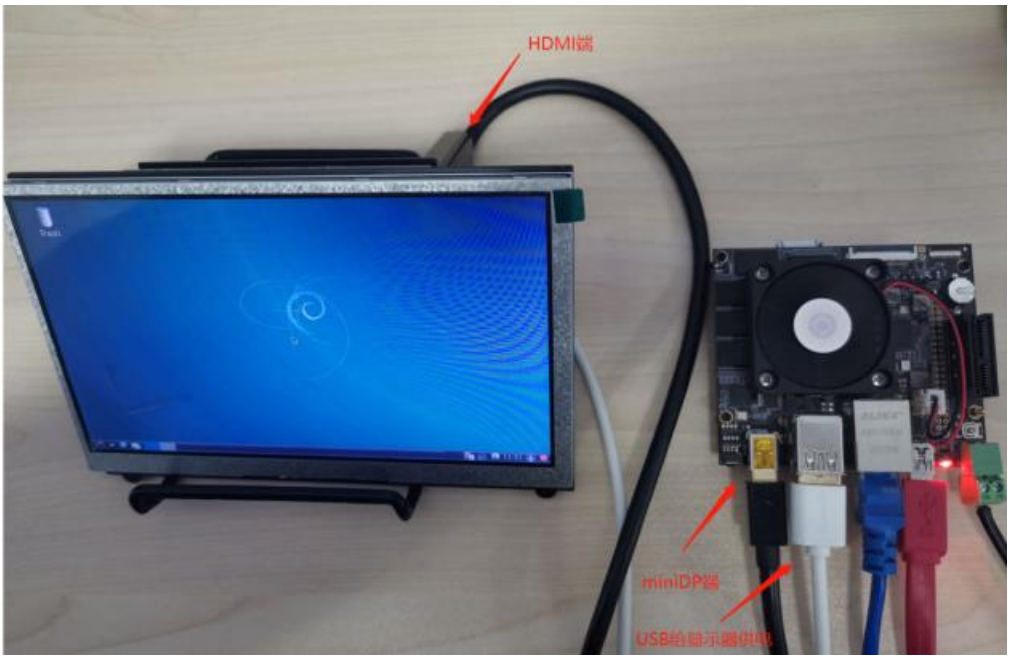
FZ3B 带有 1 路 MINI 型的 DisplayPort 输出显示接口，用于视频图像的显示， 最高支持 4K x 2K@30Fps 输出。 由于显示器的常见接口为标准 HDMI 接口,因此在使用此接口连接显示器的 时候就需要用到视频转接线，miniDP 转 HDMI 的转接线。经过测试并不是所有 的 miniDP 转 HDMI 线可以使用，必须用标有主动式的转接线才可以正常的视频 输出。

主动式 miniDP 转 HDMI 转接线

推荐链接：https://item.taobao.com/item.htm?ft=t&id=666788265034

连接显示器（以小显示器为例）

推进链接：https://item.taobao.com/item.htm?ft=t&id=666788929809 找到小显示器的 HDMI 接口，通过 miniDP 转 HDMI 线连接板卡即可。（注意 此类小显示器一般需要 USB 供电）。 开机启动后即可看到连接画面，有图形界面，有命令行界面，无论显示图形 界面还是命令行界面均表示视频输出成功。



8.5 Linux 常用指令介绍：

1、 ls list-列举当前目录下的文件

2、 cd Change Directory-切换文件路径

3、 mkdir Make Directory-新建一个新目录

4、 pwd Print Working Directory-显示当前目录的绝对路径

5、 rm Remove-删除给定的文件

6、 mv Move-移动文件或修改文件名称

7、 cp Copy-对文件进行复制

8、 cat concatenate and print files-查看文件内容

9、 date 显示和设定系统时间

10、 tar 用于建立和还原备份文件

11、 unzip 用于解压 zip 格式的压缩文件

12、 ping 检测是否与主机连通

13、 ifconfig 用于显示或设置网络设备

14、 vim 文本编辑器

vim 分为三种模式，分别是命令模式（Command mode），输入模式（Insert mode）和底线命令模式 （Last line mode） 例如：使用 vim 建立一个 test.txt 文件 vim

test.txt 进入命令模式 输入小写字母“i”，下方出现 INSERT,进入输入模式在此模式下可以写文件 文件编辑完后，按下 Esc 按键，退出输入模式，回到命令模式，然后输入 :q 离开文本编辑页面

15、 reboot 重启系统

16、 poweroff 关机

第九章 车模主要参数

9.1 车模外形参数

长：30.5cm

宽：19cm

高：37.5cm

9.2 传感器种类、规格(型号)数量。

摄像头（HF868-2 模组 2.8mm_130 度无畸变）一个，逐飞六轴陀螺仪一个，赛曙科技的光电码盘一个。

9.3 微处理器型号和个数

英飞凌 Tc264 单片机一块

Edgeboard 计算卡一块

9.4 电池的种类、规格和数量

种类：3S 高性能锂电池

规格：3S1P 25c 24.42wh

11.1v 2200mAh

数量：一块

参考文献

[1] 卓晴，黄开胜，邵贝贝．学做智能车．北京：北京航空航天大学出版社．2007.

[2] 白志刚．自动调节系统解析与 PID 整定．化学工业出版社，2012.

[3] 胡松涛.自动控制原理(第六版）[M].北京:科学出版社.2007

[4] 朱政合.飞思卡尔智能赛车及调试平台开发研究.大连理工大学.硕士学位论 文.2010.

[5] 谭浩强 C 程序设计[M] 北京：清华大学出版社，2001.

[6] 黄俊年，王立.浅析 PID 控制原理及应用[J]. 武汉：华中师范大学信息技术系.1671-7597(2010)0620109-01.

[7] 张哲,李玉丽.互补滤波在直立控制算法的应用探究[J].民营科技,2015(08):44.

[8] 何维康，田佳庆，陈育远.第十六届全国大学生智能汽车竞赛技术报告.技术报告,2021.

[9] 郑欢欢，白鱼秀，张雅琼.一种基于Ｓｏｂｅｌ算子的边缘检测算法.榆林学院信息工程学院.1007-757X(2020)10-0004-03

[10] 田甜，刘强，尹仕威，等.基于全变分去噪和八方向 sobel 算子的叶脉提取算法 [J] 浙江农业学报，2015，27(4) : 678 — 683.

.

附录：部分核心代码及主控板原理图和电路图

主要代码：

```
#include "Cpu0_Main.h"
#include "headfile.h"
#pragma section all "cpu0_dsram"


void Data_Send(UARTN_enum uratn,unsigned
short int *pst);
void init_all();
void GetSwitchStatus();


unsigned short int data_send[8];
int Go = 0;
uint8 uart_buff = 0;
int switch1_status = 0;
int switch2_status = 0;

//脉冲计数
float Encoder_Pulse_Count = 0;
```

```
uint8 Start_Pulse_Count_Flag = 0;

//定距停车
float fDistanceStop_Value_Set = 500;
uint8 If_Start_fDistanceStop_Flag = 0;

//发车延时
int Go_DelayTime_Set = 2500;

//圆环环内速度
int Circle_Speed_Set = 65;
uint8 Circle_Speed_Control_Flag = 0;
int core0_main(void)
{
    get_clk();//获取时钟频率  务必保留
    //用户在此处调用各种初始化函数等
    init_all();

    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent,
0xFFFF);
    enableInterrupts();
```

```
    while (TRUE)
    {
        if (uart_query(UART_2, &uart_buff))
//当串口收到了一个数据的时候，把数据赋给了
uart_buff
        {
            if (uart_buff < 80 || uart_buff >
175)    //传过来的数据是误差
            {
                Error = uart_buff;
                if (Error > 100)
                {
                    Error = -(256 - Error);
                }
            }
            else
            {
                switch (uart_buff)
                {
                    case STOPSTOP: Go = 0;
```

```
Expect_Speed = 0; break;

                    case RUKUSTOP:
Ruku_Stage = 1; break;


                    case
CIRCLE_SPEED_CONTROL:
Circle_Speed_Control_Flag = 1; break;
                    case
CIRCLE_SPEED_CONTROL_OUT :
Circle_Speed_Control_Flag = 0; break;


                    case
INTO_SHIGONG_DISTANCE_START :
Into_SG_Stage = 1; break;
                    case
IN_SHIGONG_DISTANCE_START : In_SG_Stage = 1;
break;
                    case
OUT_SHIGONG_SPEED_CONTROL :
```

```
Out_SG_Speed_Control_Flag = 1; break;
                    case
OUT_SHIGONG_SPEED_CONTROL_OUT :
Out_SG_Speed_Control_Flag = 0; break;
                }
            }
        }

        //----------------------施工区
-------------------------//
        if (Into_SG_Stage == 1 &&
Into_SG_Last_Stage == 1)
        {
            IntoSGDistanceControl();
        }

        if (In_SG_Stage == 1 &&
In_SG_Last_Stage == 1)
        {
            InSGDistanceControl();
        }
```

```
        //----------------------入库
-------------------------//
        if (Ruku_Stage == 1 &&
Ruku_Last_Stage == 0)
        {
            systick_delay_ms(STM0,
Left_Ruku_DelayTime_Set);

            Ruku_Last_Stage = 1;

            Encoder_Pulse_Count = 0;
//清除脉冲计数
            Start_Pulse_Count_Flag = 1;
//开启脉冲计数

            CleanQuaternionData();
//清除四元数参数
        }

        if (Ruku_Stage == 1 &&
Ruku_Last_Stage == 1)           //入库
```

```
        {
            Error = 79;
            Ruku();
        }



        if (Go == 0)                //不发车的话就显示菜单
        {
            menu();
        }
        else
        {
            key_information();
            if (key_no_flag)    //检测停车
            {
                key_no_flag = 0;


                Expect_Speed = 0;
                Go = 0;
```

```
                systick_delay_ms(STM0,
10);//延时，按键程序应该保证调用时间不小于
10ms
                lcd_clear(WHITE);
            }
        }


        if(If_Start_fDistanceStop_Flag ==
1)
        {
            if (Encoder_Pulse_Count >=
fDistanceStop_Value_Set)
            {
                Expect_Speed = 0;
                Go = 0;
            }
        }
```

```
        //----------------------舵机
-------------------------//
        DuojiLoop();                    //舵机
闭环
//        AngleLoop();                    //角
度闭环
        DuojiControl();                 //舵机
控制
        DuojiLimit();                   //舵机
限幅
        pwm_duty(DUOJI_PORT, Duoji_Pwm);
//舵机
    }
}


//初始化
void init_all()
{
```

```
//
uart_init(UART_0,115200,UART0_TX_P14_0,UA
RT0_RX_P14_1);

uart_init(UART_2,115200,UART2_TX_P14_2,UA
RT2_RX_P14_3);


    lcd_init();      //初始化 TFT 屏幕
    gtm_pwm_init(DUOJI_PORT, 50,
Duoji_Init_Pwm);                    //舵机减小
往右   左 825   中 675 右 max520

    gtm_pwm_init(MOTOR_PORT1, 17000, 0);
//正转
    gtm_pwm_init(MOTOR_PORT2, 17000, 0);
    gpio_init(P22_2, GPI, 1, PUSHPULL);
    gpio_init(P22_3, GPI, 1, PUSHPULL);
    gpio_init(P21_1, GPI, 1, PUSHPULL);
    gpio_init(P32_4, GPI, 1, PUSHPULL);
```

```
gpt12_init(GPT12_T6,GPT12_T6INA_P20_3 ,GP
T12_T6EUDA_P20_0 );



//    simiic_init();                    //iic
通信
//    icm20602_init();
//    gyroOffsetInit();


    icm20602_init_spi();              //spi
通信
    gyroOffsetInit();


    pit_interrupt_ms(CCU6_0, PIT_CH0, 1);
    pit_start(CCU6_0, PIT_CH0);
    pit_enable_interrupt(CCU6_0, PIT_CH0);

    pit_interrupt_ms(CCU6_0, PIT_CH1, 5);
    pit_start(CCU6_0, PIT_CH1);
```

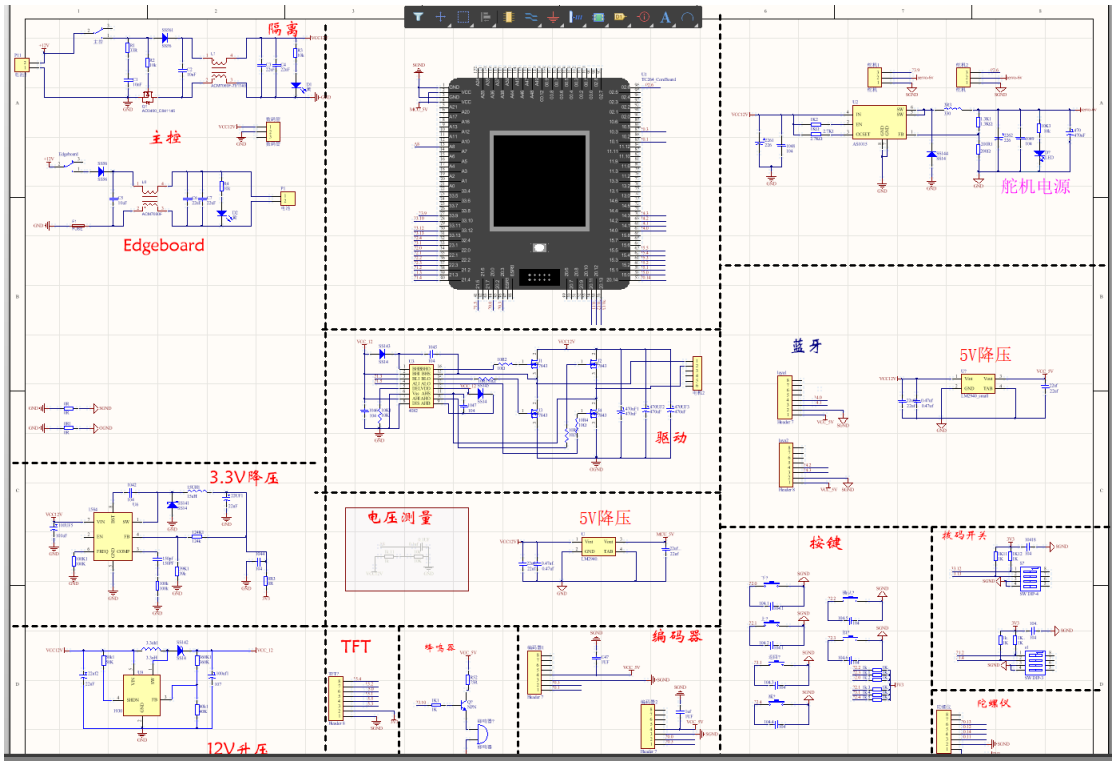
```
    pit_enable_interrupt(CCU6_0, PIT_CH1);
}


//获取拨码开关电平状态，往上拨是 1，下拨是 0
1 就是左三岔  0 就是右三岔
void GetSwitchStatus()
{
    switch1_status = gpio_get(switch1);
    switch2_status = gpio_get(switch2);
}



/*!
 *  @brief      匿名科创上位机
 *  @param
 *  @param
 *  @since      v5.0
*  Sample usage:
            需定义 unsigned short int
data_send[6];以及初始化 uart
```

```
                data_send[0]=bmq1;
                data_send[1]=bmq2;

Data_Send(UART4,data_send);
 */
void Data_Send(UARTN_enum uratn,unsigned
short int *pst)
{
        unsigned char _cnt=0;   unsigned
char sum = 0;
    unsigned char data_to_send[23];
//发送缓存
    data_to_send[_cnt++]=0xAA;
    data_to_send[_cnt++]=0xAA;
    data_to_send[_cnt++]=0x02;
    data_to_send[_cnt++]=0;
    data_to_send[_cnt++]=(unsigned
char)(pst[0]>>8);  //高 8 位
    data_to_send[_cnt++]=(unsigned
char)pst[0];  //低 8 位
    data_to_send[_cnt++]=(unsigned
char)(pst[1]>>8);
```

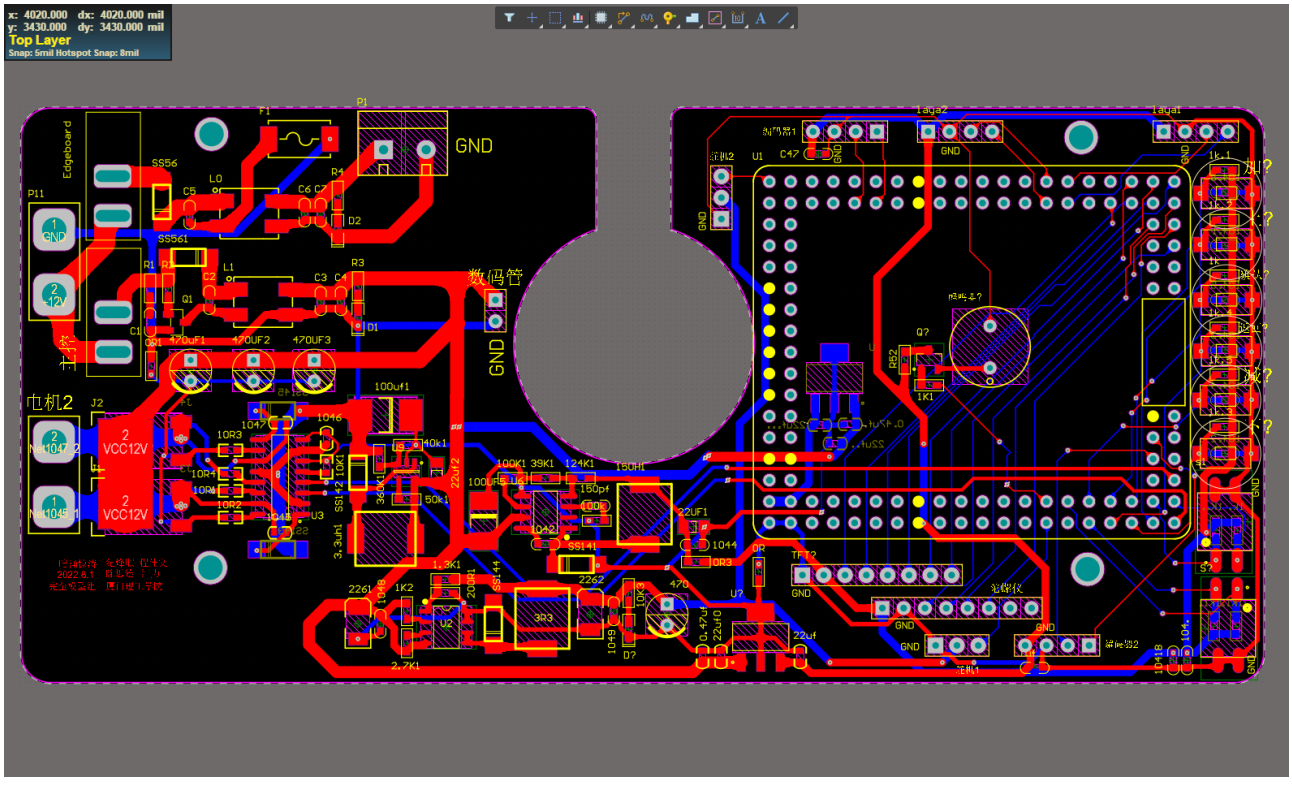
```
    data_to_send[_cnt++]=(unsigned
char)pst[1];
    data_to_send[_cnt++]=(unsigned
char)(pst[2]>>8);
    data_to_send[_cnt++]=(unsigned
char)pst[2];
    data_to_send[_cnt++]=(unsigned
char)(pst[3]>>8);
    data_to_send[_cnt++]=(unsigned
char)pst[3];
    data_to_send[_cnt++]=(unsigned
char)(pst[4]>>8);
    data_to_send[_cnt++]=(unsigned
char)pst[4];
    data_to_send[_cnt++]=(unsigned
char)(pst[5]>>8);
    data_to_send[_cnt++]=(unsigned
char)pst[5];
    data_to_send[_cnt++]=(unsigned
char)(pst[6]>>8);
    data_to_send[_cnt++]=(unsigned
char)pst[6];
```

```
    data_to_send[_cnt++]=(unsigned
char)(pst[7]>>8);
    data_to_send[_cnt++]=(unsigned
char)pst[7];
    data_to_send[_cnt++]=(unsigned
char)(pst[8]>>8);
    data_to_send[_cnt++]=(unsigned
char)pst[8];
    data_to_send[3] = _cnt-4;
    sum = 0;
    for(unsigned char i=0;i<_cnt;i++)
        sum += data_to_send[i];

    data_to_send[_cnt++] = sum;
        for(unsigned char i=0;i<_cnt;i++)

uart_putchar(uratn,data_to_send[i]);

}

#pragma section all restore
```

原理图：


隔离
主控
Edgeboard
舵机电源
蓝牙
5V降压
驱动
3.3V降压
电压测量
5V降压
按键
拨码开关
TFT
编码器
陀螺仪
12V升压

电路图：


Top Layer
GND
数码管
电机2
VCC12V