第十七届全国大学生
智能汽车竞赛

技 术 报 告

学　　校：北京信息科技大学

队伍名称：FAI

参赛队员：杨昌儒、杨雅子琪、庞可、方向、赵得佐

带队老师：张晓青

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于保留、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：

带队教师签名：

日　　期：2022.8.16

目录

第一章 引言

随着社会科技的不断发展，电子科技的影响逐渐深入到各行各业之中，嵌入式的迅猛发展也为智能研究提供了更为广阔的平台。并且随着 5G 网络的到来，无人驾驶汽车的研究与发展步入了一个新的快车道。通过联合各种传感器和控制器组成的控制方式就成为了新的研究热点，控制算法和控制策略也显得更加重要。

全国大学生智能汽车竞赛是国家教学质量与教学改革工程资助项目，通过使用德国英飞凌半导体公司生产的单片机作为车载核心控制模块，增加道路传感器、摄像头、电机驱动电路以及编写相应的程序，制作出一个可以自主识别道路的汽车模型。因而该竞赛是涵盖了智能控制、模式识别、传感技术、汽车电子、电气、计算机、机械等多个学科的比赛，对学生的知识融合和实践能力的提高，具有良好的推动作用。

本篇技术报告将从智能车的机械结构、硬件电路、算法设计等方面全面详细的介绍整个准备过程。分为整体设计思路、上下位机、开发环境、调试过程 5 个章节进行细节性的阐述。

第二章 正文

2.1 整体设计思路

根据竞赛规则要求，上位机选择基于赛灵思芯片的百度 Edge Board，使用 Paddle-Lite 进行识别相关处理。下位机选择逐飞科技提供的 Infineon（英飞凌）TC212 微控制器。摄像头架设于整体车模之上，型号为 S320，镜头为 28mm/110°，水平视角 62°，垂直视角 46°，通过 USB2.0 与 Edge Board 连接。动力单元为 RS-555 电机，配合 CSPE5-500 光电编码盘，对电机实现闭环控制。车模转向动力来源选择 CS-3120 舵机，通过四杆机构对前轮进行角度控制。

软件层面，我们将摄像头直接接驳在 Edge Board 上，通过 OpenCV 进行图像初步处理与传输，使用 Paddle Lite 进行对图片的特征识别。识别特征后，通过算法计算预测轨迹，并将动作指令发送给下位机，下位机能实现对上位机动作指令的处理，并将动作指令发送给直流电机（驱动轮）和舵机（转向轮）。

2.2 下位机电路设计

由于使用的 Infineon（英飞凌）TC212 微控制器，需要对其配套电路进行设计。电路板包含了电压转换电路、电机驱动电路、电机控制电路、舵机驱动电路、舵机控制电路、按键、电池电压检测电路等。

2.2.1 电池电源稳压滤波电路

系统供电为直流供电，需要设置一些防反接的设计。在电源接口上，我们使用了正负极有极强区分度的 T 型接口，并且在电路设计上，通过 AO3400A 型（AO3400 改进型）MOS 管进行防反接设计。AO3400A 原理图等如下图所示：

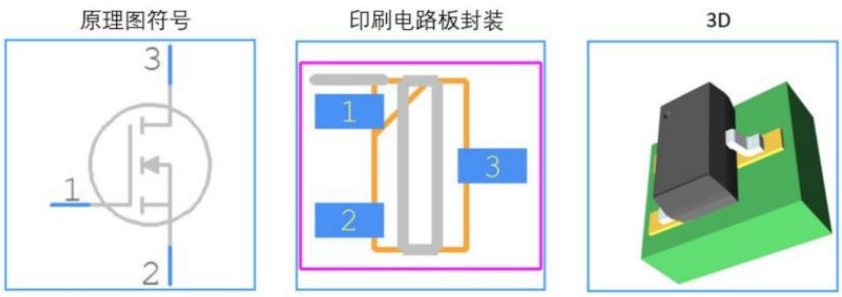

图 2.1 AO3400A 示意图

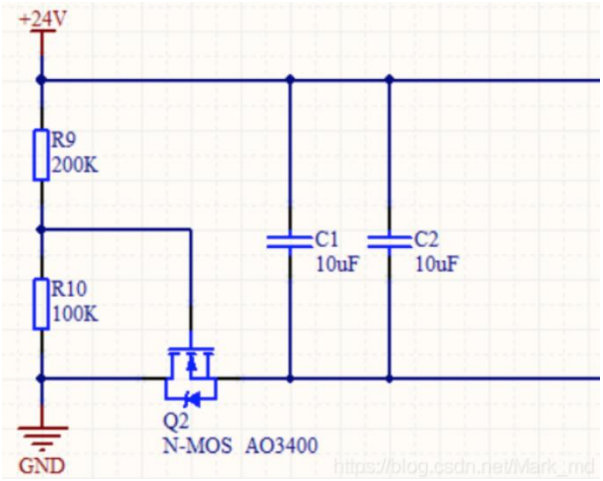

图 2.2 N-MOS(AO3400A)防反接电路原理图

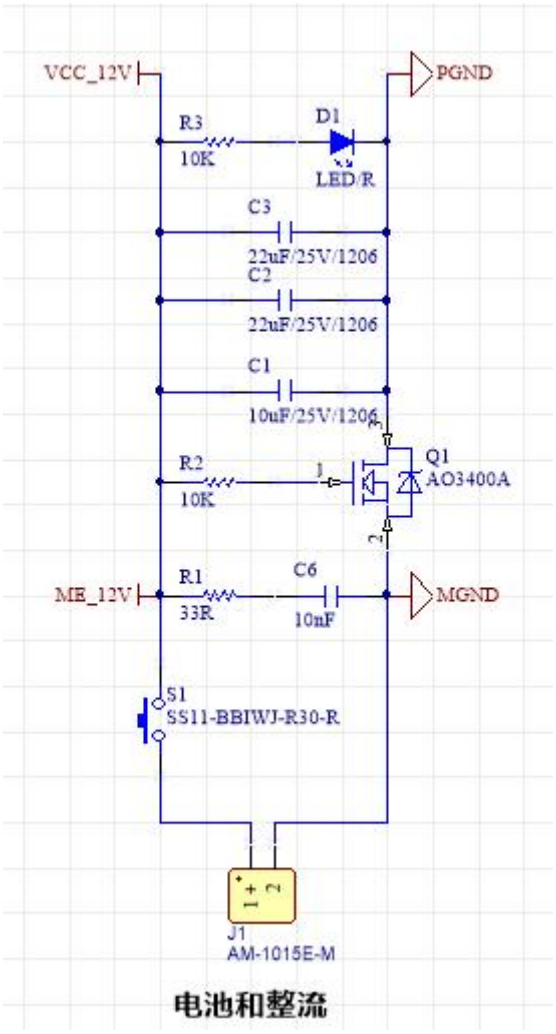

图 2.3 电池供电稳压电路

当电池正确接入电路后，打开开关，电荷经过电容器后被整形，且点亮 LED 灯表示已经通电，并输出稳定的 12V。

2.2.2 Edge Board 供电电路

将有整流稳压功能的 Edge board 供电电路接入电池供电电路产生的 12V 稳定电压，直接供给 Edge board 的电源接口，即可正常驱动上位机。

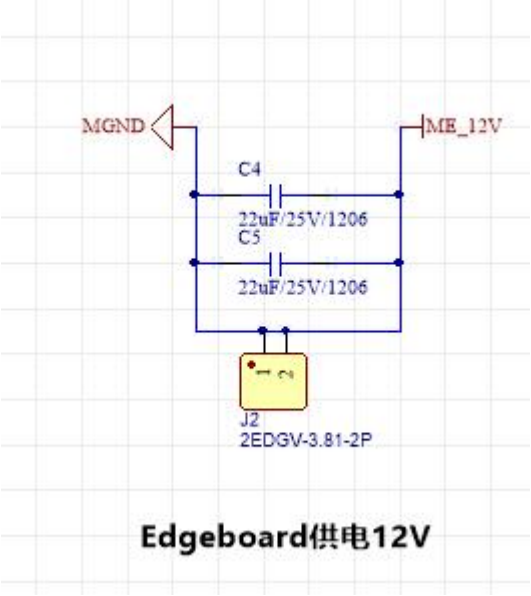

图 2.4 Edge Board 供电电路

2.2.3 USB2.0-HUB 和 12V 转 5V-3A 外设供电电路

由于 Edge Board 只有两个 USB 接口，所以我们设计了一个 USB 拓展坞，并且为缓解 Edge Board 对外设（摄像头、遥控手柄、USB 网卡等）供电的压力，设计了 12V—5V 的降压电路。本电路采用 TPS5430 降压稳压芯片，电压输入范围为 5.5V-36V，能够以 95%的转换效率输出稳定的 3A 电流。根据芯片数据手册，通过 TI 官网的“Power Designer”工具，能够很方便地生成所需电路。在设置好输入电压范围、输出电压、输出电流、使用环境温度等后，点击“Open Design”按钮，即可生成相关电路，再依据自身需求增加电路功能模块。我们增加了并联电容组以增强供电的稳定性。

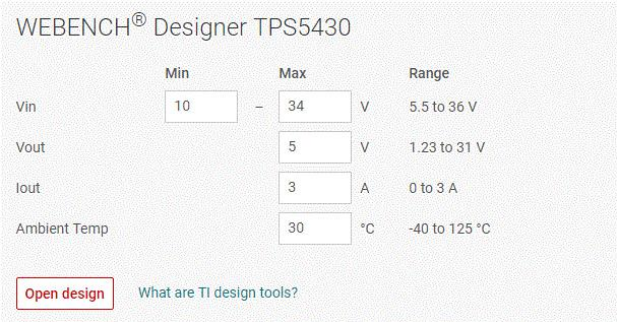

图 2.5 WEBENCH® Power Designer 工具

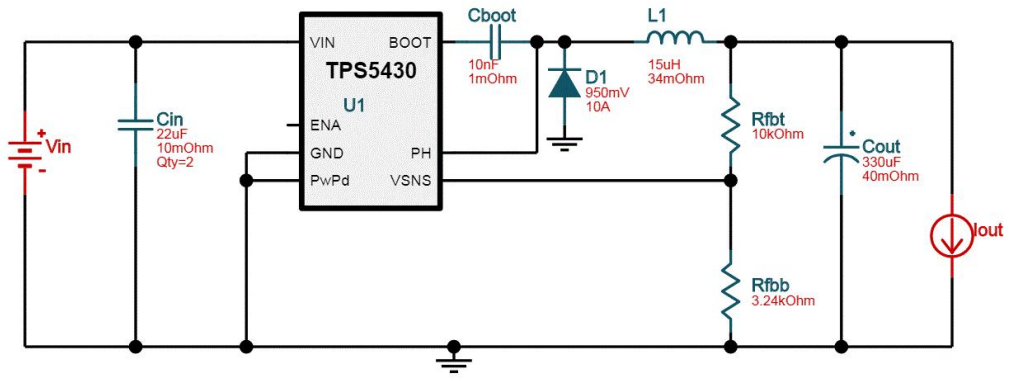

图 2.6 自动生成的电路

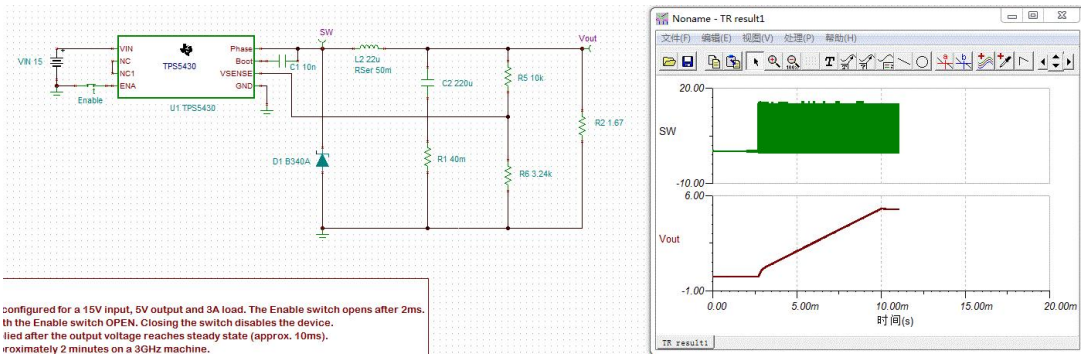

图 2.7 电路仿真

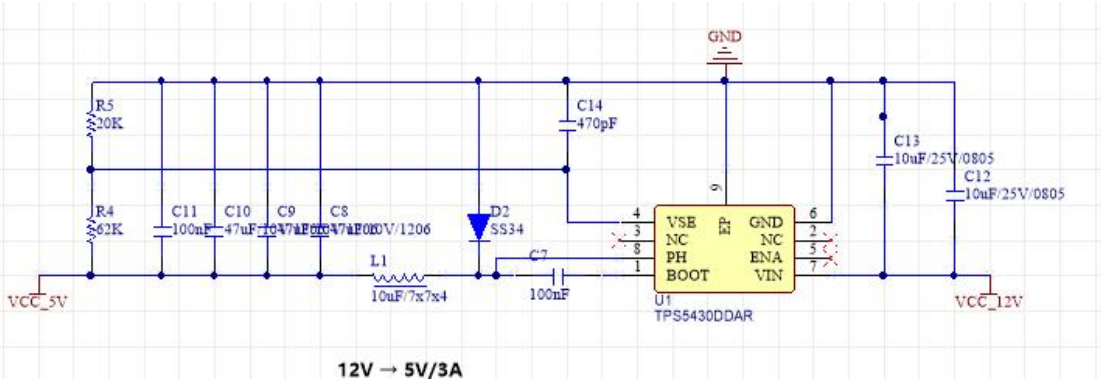

图 2.8 12V 转 5V3A 降压电路

USB2.0-HUB 电路结构采用 SL2.1s USB-HUB 控制芯片，该芯片为国产的四端口 USB2.0 芯片，工作需要 5V 供电，外部电路结构简单，仅需添加电容和滤波电路，芯片自带复位电路和 5V 转 3.3V 电路。USB2.0 接口引脚为 VBUS、D-、D+、

GND 原理图如下：

图 2.9 USB2.0 原理图

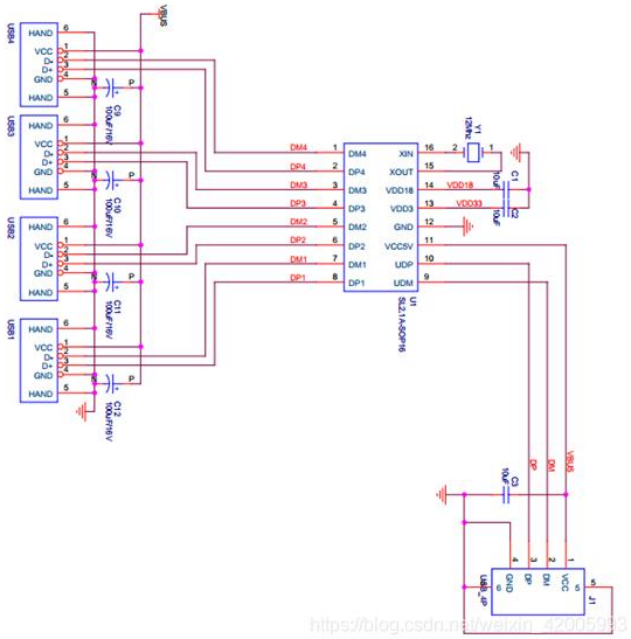

图 2.10 USB2.0-HUB 电路原理图

USB 转 UART（U 转串）芯片使用当下最常用的 CH340 芯片。CH340G 是全速 USB2. 0 芯片，外围电路简单（只需要提供晶振和电容），对于 Linux 和 Windows 系统完全兼容。上下位机通过它进行通信。

2.2.4 下位机供电

下位机采用英飞凌 TC212，根据芯片手册，TC212 需要 3.3V 供电，所以采用 ME62113.3 芯片为核心进行电路设计。ME62113.3 是低功耗的压差 LDO 芯片，工作电压为 2-6V，最大输出电流为 0.5A，能够将 5V 电压转化为 3.3V 并接给单

片机。电路设计为：

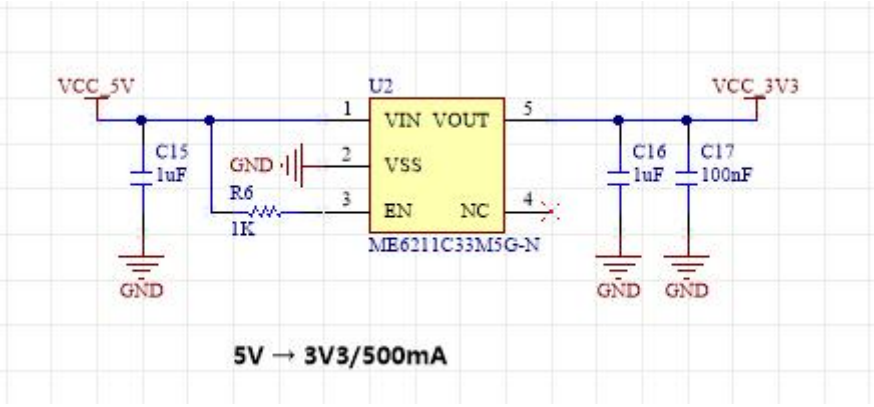

图 2.11 下位机供电（5V 转 3.3V）电路设计

2.2.5 直流电机驱动电路

直流电机控制方式为 PWM（脉冲宽度调制），采用全桥整流电路进行驱动。核心原理示意图如图所示：

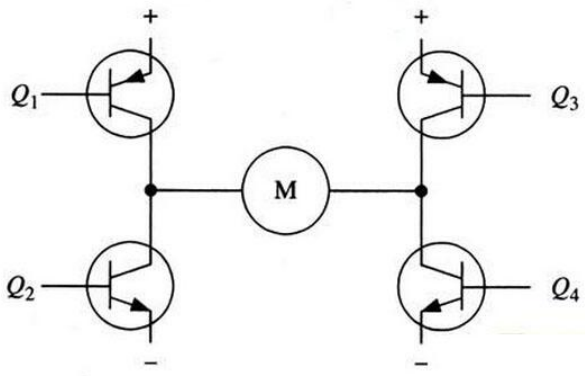

图 2.12 全桥整流电路的原理图

上图所示为 H 桥式驱动电路，4 个三极管组成 H 的 4 个边界，M 在 H 中间一横上，要使电机运转必须将对角线上一对三极管导通，根据三极管导通方向的不同，电流可能会从右向左或从左向右流过电机，进而使电机正转或反转。

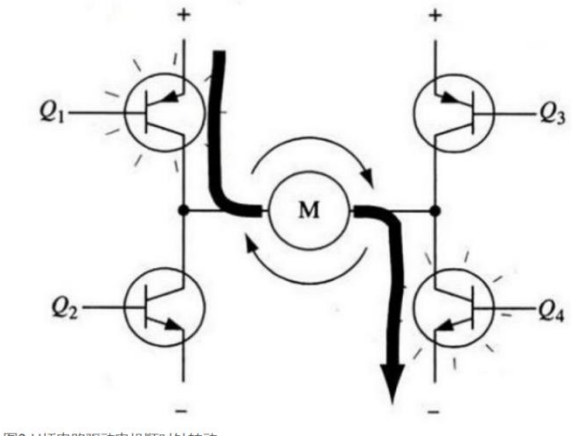

图 2.13 H 桥电路驱动电机顺时针旋转

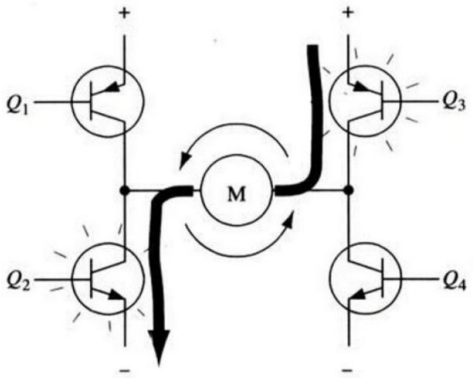

图 2.14 H 桥电路驱动电机逆时针旋转

H 桥驱动电路有集成好的电机驱动芯片，集成好的芯片使用方法简单，但是所能承受的电流较小，对于我们所选用的直流电机来说，有烧毁芯片的风险。所以我们用 IRLR7843 芯片（MOS 管）构造了 H 桥，处理 PWM 信号。PWM 信号经过 SN74LVC1G04 芯片（非门）处理成能够放大且能够正常驱动电机的电流，再经过由 IR2184 功率放大芯片足证的 H 桥，从而实现对直流电机的控制。

IR2184 芯片为功率放大芯片，IR2184 型半桥驱动芯片能够驱动高端和低端两个 N 沟道 MOSFET，能提供较大的栅极驱动电流，并具有硬件死区、硬件防同臂导通等功用。运用两片 IR2184 型半桥驱动芯片能够组成完好的直流电机 H 桥式驱动电路，而且 IR2184 价格低廉，功用完善，输出功率能够满足小车要求。

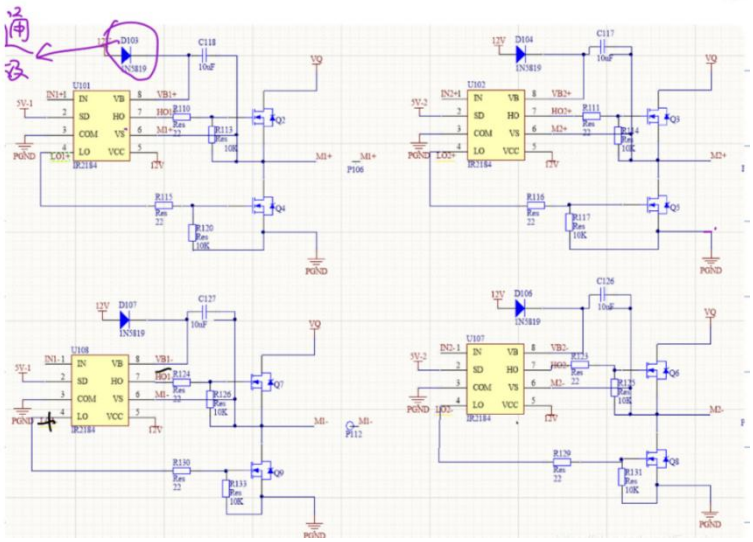

图 2.15　IR2184 常用电路

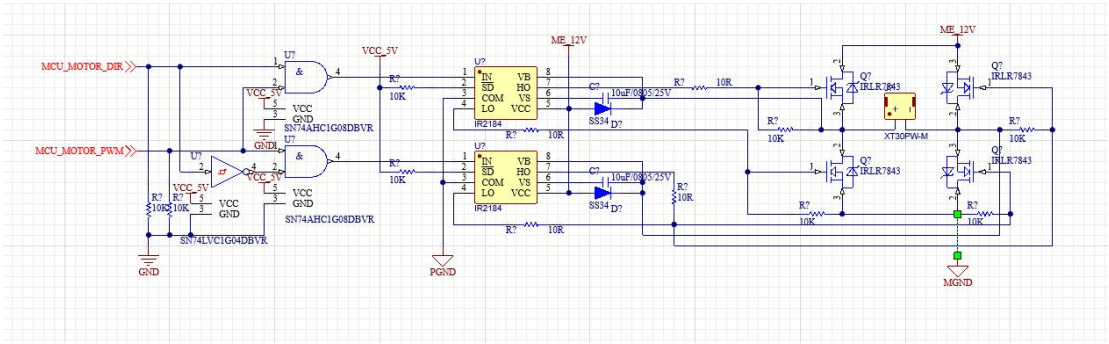

图 2.16 H 桥电机控制电路

2.2.6 舵机驱动电路

舵机为 6.3V 供电，我们采用 TPS5430 降压稳压芯片对 12V 进行降压操作。因为舵机都是以 PWM 方式控制角度，所以不需要额外驱动芯片。经过 TI Power Designer 工具辅助设计，最终电路如图：

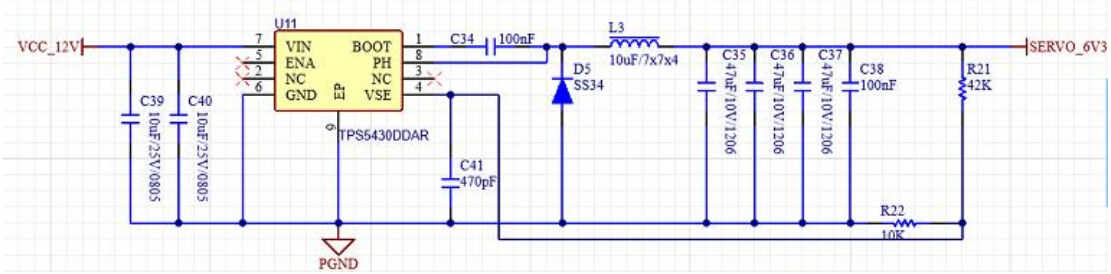

图 2.17 12V 转 6.3V 电路

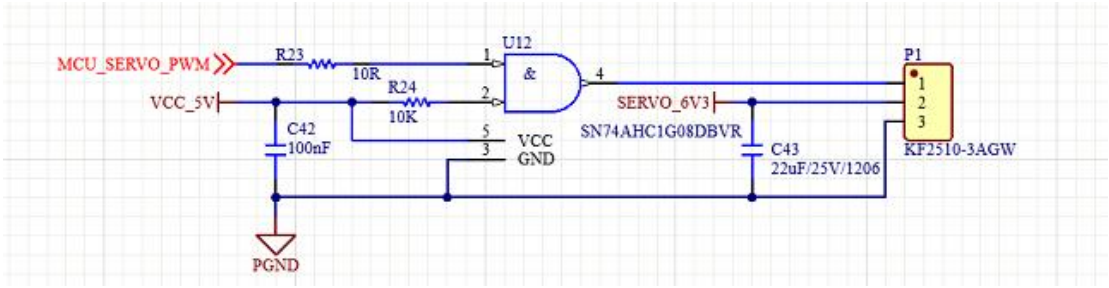

图 2.18 PWM 舵机控制电路

注：KF2510 为 3Pin 接线端子

2.2.7 陀螺仪与电量计设计电路

陀螺仪芯片选择 MPU6050 六轴陀螺仪芯片。与下位机通信方式为I^2C，但程序设计中未使用陀螺仪，所以在电路板上未实际贴片。

电量计采用 CW2015CHBD 电量计芯片。采用I^2C通信。实际使用中电量监测使用在电池平衡端连接的 BB 响。

2.2.8 按键电路

为启动程序方便，我们增加了两个按键。连接到下位机上，实际使用中，按下按键后，像主程序发送“start”信号。

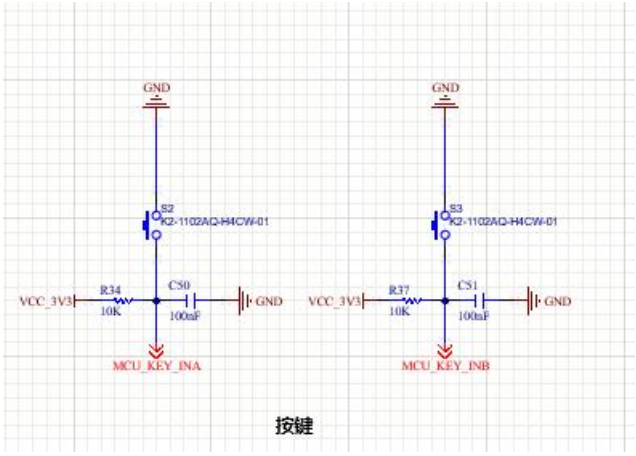

图 2.19 按键电路

2.2.9 蜂鸣器电路

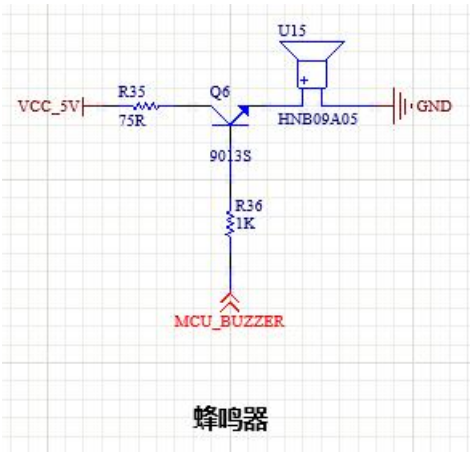

图 2.20 蜂鸣器电路

2.2.10 电路板设计思路

因为电路板为自行设计，I 型车模的空闲位置为左右两侧位置（侧箱位置），所以电路板形状在最开始就定为契合空闲位置形状而设计的。

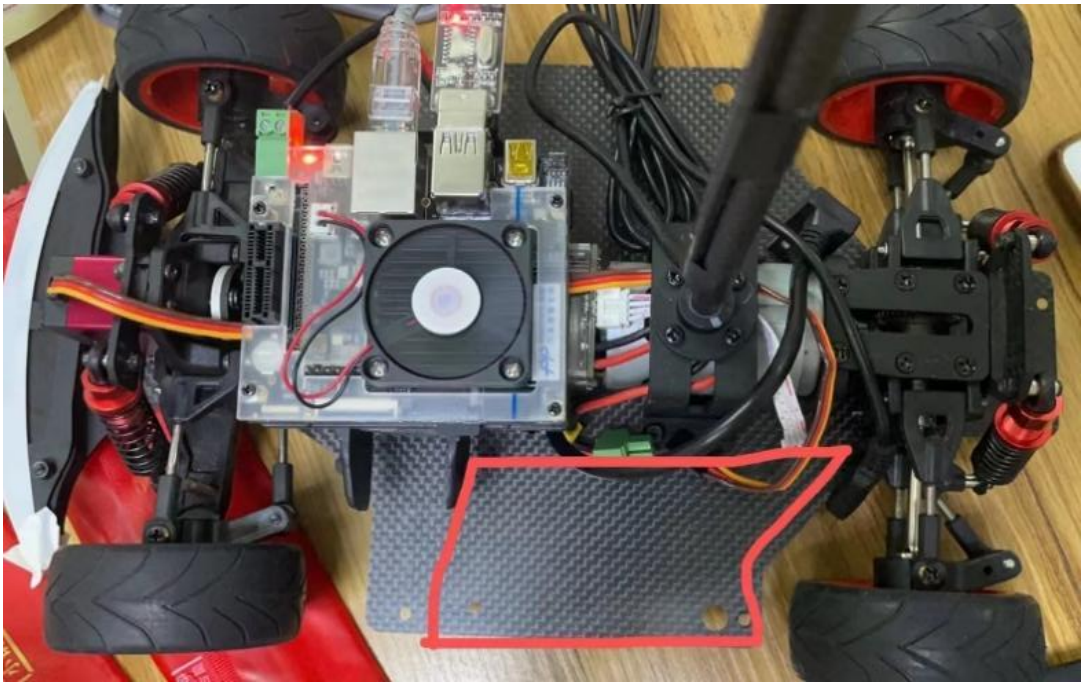

图 2.21 I 型车模空闲位置

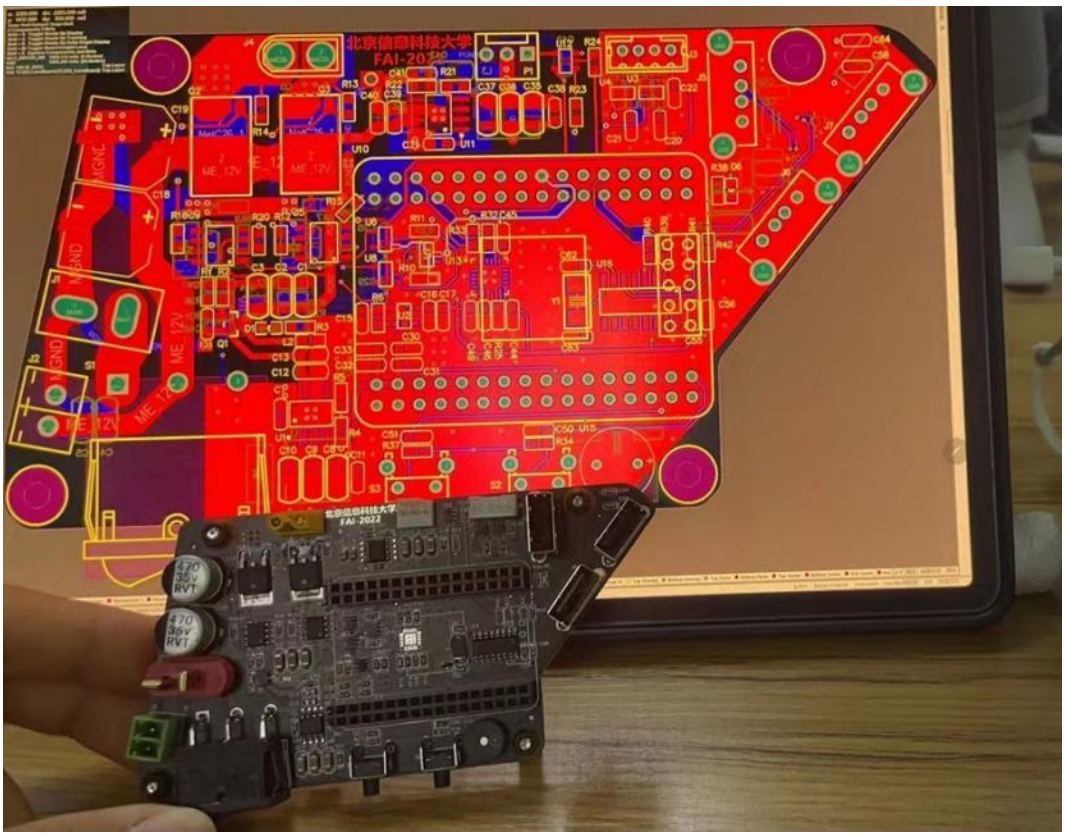

图 2.22 电路板设计与实物打板

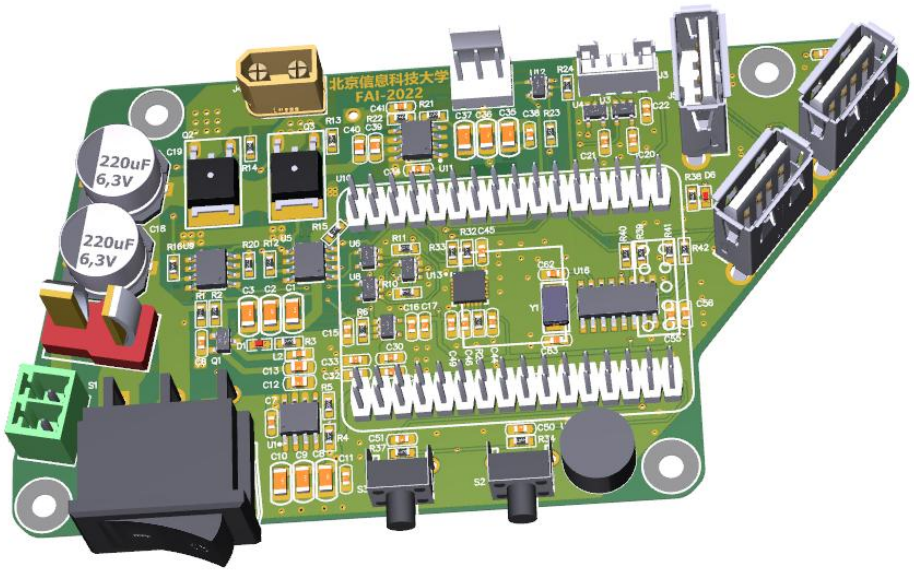

图 2.23 电路板 3D 模型

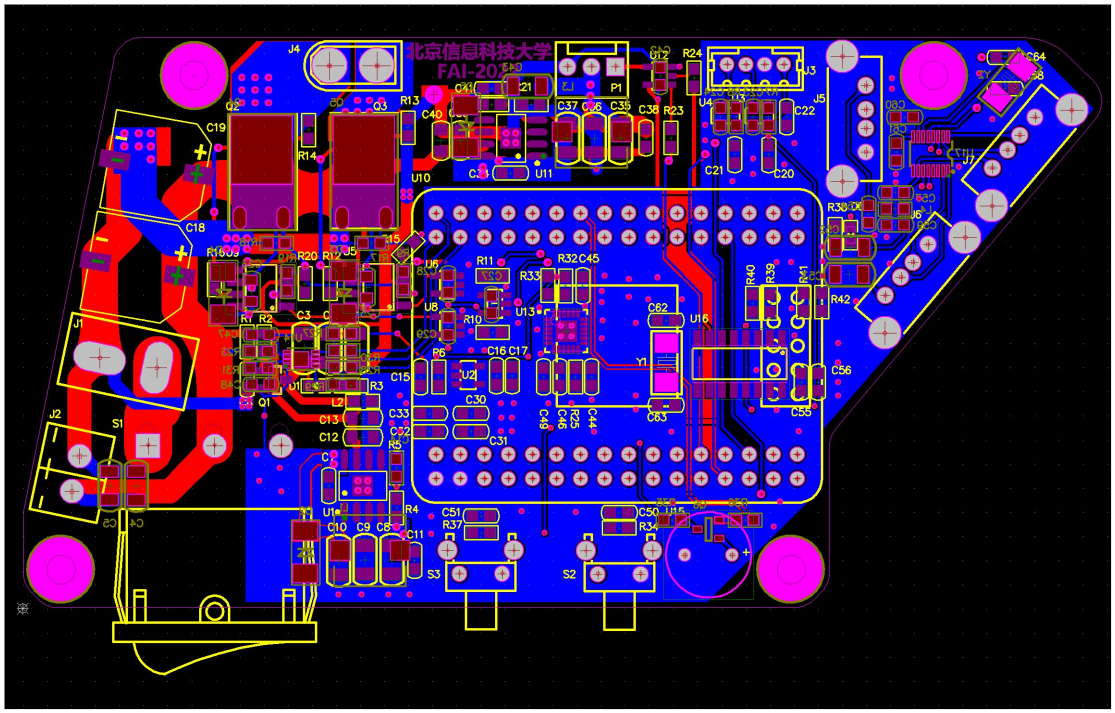

图 2.24 非机械层光绘图

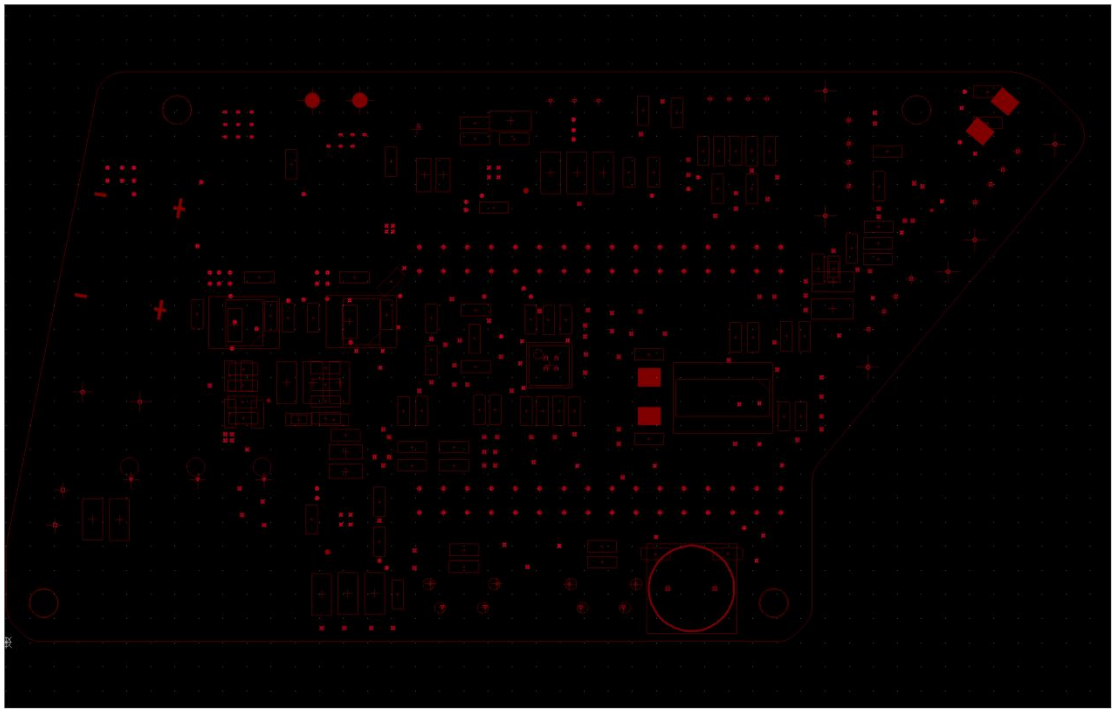

图 2.25 机械层光绘图

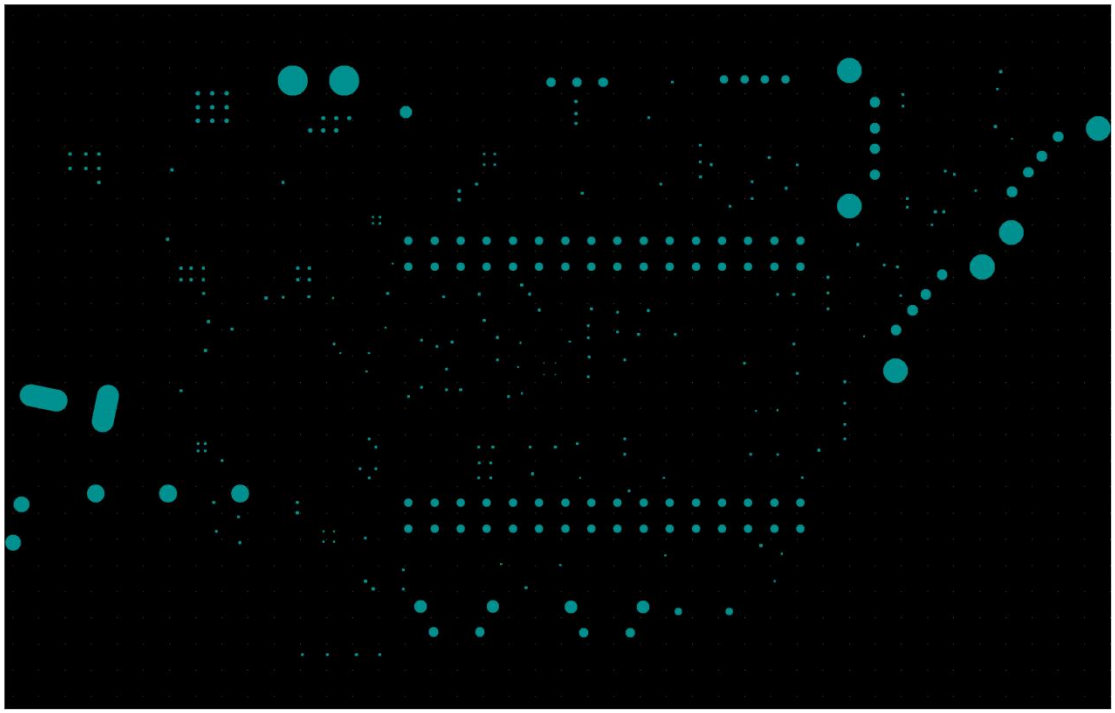

图 2.26 钻孔层光绘图

2.3 上位机程序设计

上位机是整个系统的大脑，我们采用百度大脑出品的 Edge Board 计算卡作为上位机。Edge Board 有着更高的算力，并且有 FPGA 芯片计算加速功能，对于智能车来说，高算力能够带来更高的车速和更精准的路径预测。我们通过对驱动车辆后轮用的直流电机添加光电编码器，以实现对电机的闭环控制，通过摄像头获取赛道信息，进而实现系统的闭环。在速度和转向控制上，我们采用 PD 算法和多 P 算法，尽可能精准地控制车辆的走线。

2.3.1 赛道图像特点与程序设计基本思路

经过标定的摄像头向上位机传输的图像具有明显特征。上位机算力评估后，我们决定采用 320*240 分辨率，在尽可能保证清晰的前提下，减小对算力的要求。

图 2.27 三岔路口图像演示

经过对摄像头采集的赛道进行分析，视野中颜色区分度极大。能够十分容易地进行二值化以减小对算力的依赖。红色的赛道特殊元素标志在白色赛道上，且十分明显，将相机获取的数据传入到训练好的模型中，就能得出该图像中是否存在特殊元素。

程序设计思路基本分为串行和并行两种方法。考虑到往年比赛的赛道是未知的，所以我们采用并行结构设计识别结构。将不同识别功能的程序单独放在 src 文件夹下，编写 Main 函数进行循环判断，条件符合就进入到当下条件的识别程序。

为发车方便，我们编写了上下位机通信程序，当下位机按钮按下时，向主程序发送 start 命令，程序开始运行。

```
    cout << "准备出发!!!" << endl;
    detection = Detection::DetectionInstance("/dev/video0", "../res/model/mobilenet-ssd-v2"); //
    printAiEnable = false;                                                                      //

    while (!driver->receiveStartSignal()) //串口接收下位机按键按下信号
    {
        ;
    }
    cout << "------------- Program start!!! -------------" << endl;

    for (int i = 0; i < 30; i++) // 3秒后发车
    {
        driver->carControl(0, PWMSERVOMID);
        waitKey(100);
    }
}
```

图 2.28 按键启动程序

2.3.2 赛道识别

由于所采用的相机为彩色相机，直接处理数据算力消耗过高，所以使用 OpenCV 库对图像进行预处理。将 RGB 图像转到灰度空间，再根据感兴趣的灰度范围对图像进行二值化处理（OTSU）。二值化处理后的图像只有 0 和 255 两个颜色（白和黑），算力消耗极低。

在二值化后的图像上进行赛道识别，我们的做法是搜索图像中颜色突变的点，然后将两个点之间做差求斜率，将这一张图片内的所有突变点斜率都计算出来，进行三阶的贝塞尔拟合，进而得到精确的赛道类型（左转、右转等）。

在精确拟合赛道左右边线后，对对应点求中点，得出点的集合即为赛道中线，赛道中线与相机视场中线的差值的绝对值就是修正值。

特殊区域的赛道预测原理相同。AI 能够识别锥桶并返回锥桶中心位置，锥桶中心位置形成的点集拟合曲线后即为赛道边界。

2.3.3 舵机角度控制和电机速度控制

我们采用 PD 算法和多 P 算法。真实修正量依靠两个参数同时作用于赛道中线与相机视场中线的差值的绝对值，再通过 PWM 转换将像素距离转化成舵机应该调整的角度值所对应的 PWM 值。

速度控制原理为，将现有真实速度（光电编码器反馈数据）与理论速度进行对比，如果真实速度高，则将真实速度减去真实速度与期望速度的差值的绝对值；如果真实速度低，则将真实速度加上真实速度与期望速度的差值的绝对值。实现闭环控制。

2.3.4 岔路口识别原理

根据赛道在相机中的呈现形式易知，所有的岔路口的共同特征是赛道白色区域沿 X 轴变化，且通常是变宽。当经过一定时间的持续变宽后，即可认为该位置有岔路口，再辅以其他视觉识别，即可进行后续动作。

2.4 开发环境介绍

由于使用英飞凌生产的 TC212 单片机芯片，开发环境选择 AURIX，基于逐飞科技提供第底层代码进行开发。主要包括核心 PID 控制算法、上下位机通信、电机的闭环控制算法、USB 控制、IO 口控制、PWM 控制等。

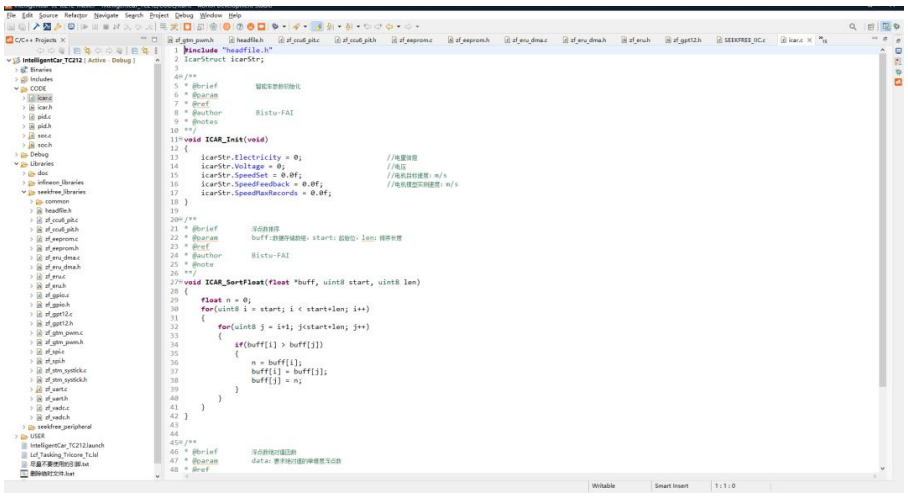

图 2.29 AURIX 软件示意图

识别程序使用 C++语言进行编写，辅助工具使用 Python 语言进行编写。使用 OpenCV 库进行图像处理。使用 Paddle__Lite 进行图片的 AI 识别。程序编写团队使用 VSCODE。VSCODE 有丰富的插件，能够提供自动补全、颜色区分等功能。

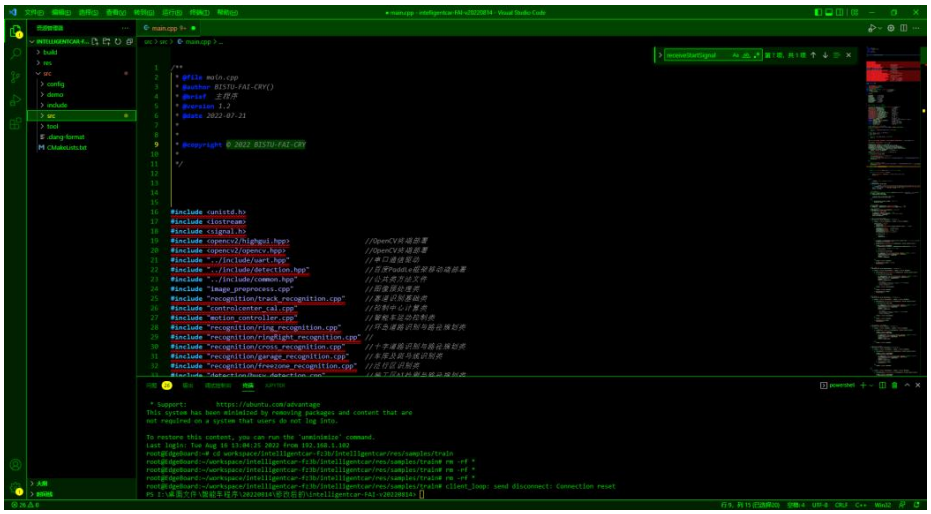

图 2.30 VSCODE 开发示意

AI 模型的训练我们采用赛事提供的 AI STUDIO，在例程的基础上进行开发与训练。

2.5 调试过程介绍

2.5.1 软件调试

我们将关键参数都定义到了一个 Config 文件中，在仅调整参数的情况下十分方便。共有 25 个参数，包括速度调节、pid 调节、项目使能、视场调节、圈数调节等内容，详情见下：

```
"#speedLow": "智能车最低速",
"#speedHigh": "智能车最高速",
"#speedDown": "特殊元素<加油站>|<施工区>减速速度",
"#speedFreezone": "泛行区速度",
"#speedGasBusy": "加油站|施工区速度",
"#speedSlop": "坡道（桥）速度",
"#speedGarage": "出入库速度",
"#runP1": "一阶比例系数：直线控制量",
"#runP2": "二阶比例系数：弯道控制量",
"#runP3": "三阶比例系数：弯道控制量",
"#turnP": "一阶比例系数：转弯控制量",
"#turnD": "一阶微分系数：转弯控制量",
"#debug": "调试模式使能（存图|看图）",
"#saveImage": "存储原始图像使能（非调试模式下）",
"#rowCutUp": "图像顶部切行（前瞻距离）",
"#rowCutBottom": "图像底部切行（盲区距离）",
"#disGarageEntry": "车库入库距离(斑马线Image占比%)[
"#rateTurnFreezone": "泛行区躲避禁行标志的转弯曲率%
"#GarageEnable": "车库使能",
"#GasStationEnable": "加油站使能",
"#BusyAreaEnable": "施工区使能",
"#SlopEnable": "坡道使能",
"#FreezoneEnable": "泛行区使能",
"#RingEnable": "环岛使能",
"#CrossEnable": "十字使能",
"#circles": "智能车运行圈数"
```

图 2.31 参数调试

所有数据都是依据现实情况进行调整。

2.5.2 硬件调试

a)硬件选型

底盘部分的主要部件包括有轮胎、悬架、电池、转向舵机、驱动电机、差速器、底盘碳板七部分构成。其中电池、转向多级、驱动电机因比赛规定无法修改。轮胎采用车模原装设计，为橡胶+海绵的设计。效果符合预期，未做更改。

在调试过程中，车模频繁出现驱动电机锥齿轮“打齿”现象。初步判断为驱动电机锥齿轮与差速器齿轮配合不严格导致。尝试将驱动电机锥齿轮沿轴向差速器侧移动无法解决问题。后进一步发现造成该现象的原因是锥齿轮与轴在径向配合不牢导致。因备赛时间不足，在初赛时仅通过拧死螺纹销的暴力方式暂时解决问题。此外、锥齿轮在于驱动电机轴固定时，轴向固定并不牢靠，后续将会更改这一部分结构设计。

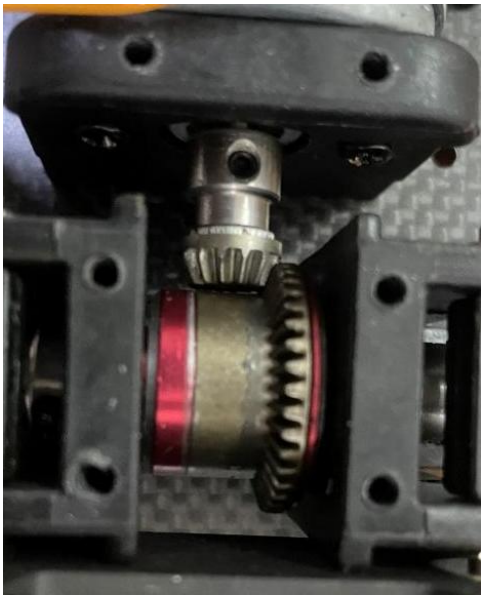

图 2.32 后轴传动装置

受加工周期影响，底盘碳板部分目前采用原装碳板。后续设计会开出更多减重孔。

悬架部分具体设计见悬挂修改和四轮定位部分。

b)悬挂修改

悬挂部分原装弹簧强度低，在车模上只能体现出很好的阻尼特性，其回弹效果接近于零。这种特征的车模，在放置车模时因力度不同，其摄像头俯角无法确定，对车模采集图像有极大影响。更换更大线径弹簧暂时无法加工，因此我们将悬挂在与车体链接一侧安装在更靠近车轮的预留孔，试图用这种方式增大回弹。在初赛时这种设计勉强可以使用。

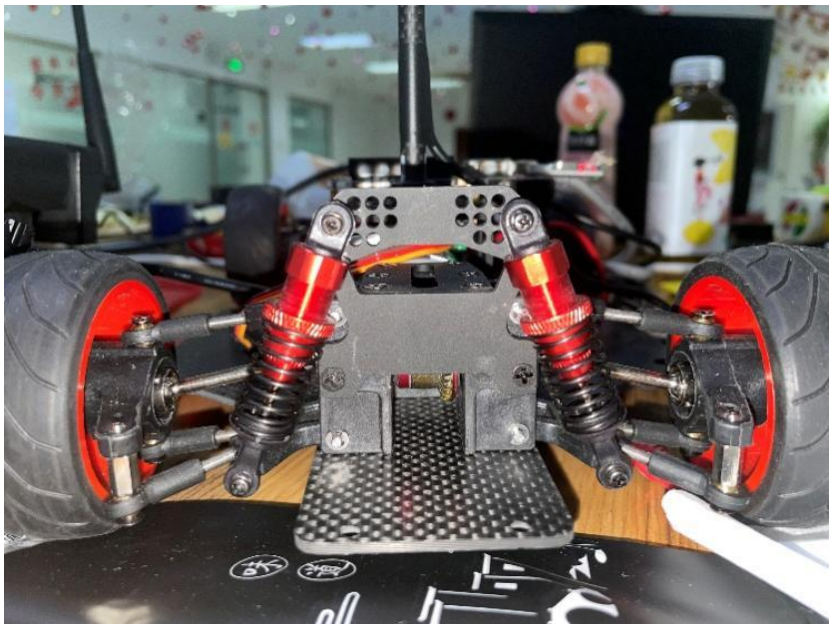

图 2.33 悬架硬度调节方式（暂行）

c)四轮定位

调节与车轮相连的两个拉杆，使前轮具有一定的负倾角，使其具有更好过弯稳定性，后轮采用 0° 倾角，使其尽可能不浪费抓地力。调整完成后，将车放在地上，轻轻推动小车，检查是否能够直线行驶。如果是，则四轮定位完成。

第三章 结论

在这份报告中，我们主要从智能车的机械结构、硬件电路、控制算法、调试方法等介绍了我们准备比赛的整个过程。作为全新的战队首次参加智能车竞赛完全模型组别，对于我们有太多的未知，但是我们不仅没有因此而退缩，反而迎难而上，付出大量的时间和精力，从一点一滴做起，刻苦钻研各项技术，队员之间密切配合、互相勉励，始终不言放弃，我们通过此次比赛学到了团队合作的重要性以及众多相关的技术知识。

智能车是培养大学生综合动手能力的一个很好的、很成熟的平台，在比赛里我们不仅需要编程能力，还需要对车辆机械结构有着较深的理解，也需要熟悉各种传感器和电路的性能。智能车比赛中最重要的是需要足够多的创新能力和想法，国家已经把创新放在了核心位置，如今的竞赛一方面是在考验大学生的基本知识，更是在考验每一个参赛大学生的创新能力，所以只有创新才可以从全国大学生中脱颖而出。

我们团队五个人并没有一帆风顺，但我们不惧艰难，苦难与折磨只会让我们更加的强大。

希望这篇技术报告会对未来参与智能车这项比赛的同学有所帮助，也祝愿他们可以在比赛中收获快乐。

参考文献

[1] 童诗白，华成英. 模拟电子技术基础[M]. 北京. 高等教育出版社. 2000

[2] 邵贝贝. 嵌入式实时操作系统[LC / OS-Ⅱ(第 2 版)[M]. 北京. 清华大学出版社. 2004

[3] 张文春. 汽车理论[M]. 北京. 机械工业出版社. 2005

[4] 雷霏霖,梁志毅.基于 CMOS 传感器 OV7620 采集系统设计[J].电子测量技术,2008,12(31):110-112.

[5] 王晓华. TensorFlow 2.0 深度学习应用实践. 第 2 版 ed. Print.

[6] 郭芳,曹桂琴.数据结构基础[M].大连：大连理工大学出版社，1994.

[7] 邵贝贝.单片机嵌入式应用的在线开发方法[M].北京:清华大学出版社,2004.

[8] 胡寿松.自动控制原理(第六版)[M].科学出版社,2014.

[9] 阎石.数字电子技术基础[M].北京:高等教育出版社,1998.

[10] 谭浩强.C 程序设计[M].北京:清华大学出版社,2005.

[11] 黄友锐. PID 控制器参数整定与实现. Print.

附录

程序源代码

上位机

```
/* main.cpp BISTU-FAI-CRY v2.0 */

#include <unistd.h>
#include <iostream>
#include <signal.h>
#include <opencv2/highgui.hpp>                    //窗口管理库
#include <opencv2/opencv.hpp>                     //OpenCV
#include "../include/uart.hpp"                    //串口通信
#include "../include/detection.hpp"               //Paddle 框架
#include "../include/common.hpp"                  //全局调用
#include "image_preprocess.cpp"                   //图像预处理
#include "recognition/track_recognition.cpp"      //赛道基础识别
#include "controlcenter_cal.cpp"                  //控制参数计算(视觉)
#include "motion_controller.cpp"                  //运动控制
#include "recognition/ring_recognition.cpp"       //左入环岛 识别和路径规划
#include "recognition/ringRight_recognition.cpp" //右入环岛 识别和路径规划
#include "recognition/cross_recognition.cpp"      //Cross 识别和路径规划
#include "recognition/garage_recognition.cpp"     //车库 识别与出入库路径规
划
#include "recognition/freezone_recognition.cpp"  //FreeZone 识别与路径规划
#include "detection/busy_detection.cpp"           //施工区 AI 识别与路径规划
#include "detection/gasstation__detection.cpp"   //加油站 AI 识别与路径规划
#include "detection/freezone_detection.cpp"       //FreeZone AI 识别与路径规
划
#include "detection/slope_detection.cpp"          //坡道 AI 识别与路径规划

using namespace std;
using namespace cv;
```

```
void callbackSignal(int signum);
void displayWindowDetailInit(void);
void displayWindowImageInit(void);
void slowDownEnable(void);
std::shared_ptr<Driver> driver = nullptr; //初始化串口
bool slowDown = false;                    //减速标志
uint16_t counterSlowDown = 0;             //减速计数

enum RoadType
        {
                BaseHandle = 0,   //基础赛道处理
                RingHandle,       //环岛赛道处理
                CrossHandle,      //十字道路处理
                FreezoneHandle,   //泛行区处理
                GarageHandle,     //车库处理
                GasstationHandle, //加油站处理
                BusyareaHandle,   //施工区处理
                SlopeHandle       //坡道处理
        };

int main(int argc, char const *argv[])
{
    std::shared_ptr<Detection> detection = nullptr; //初始化 AI 预测模型
    ImagePreprocess imagePreprocess;                //图像预处理类
    TrackRecognition trackRecognition;              //赛道识别
    ControlCenterCal controlCenterCal;              //控制参数计算
    MotionController motionController;              //运动控制
    RingRightRecognition ringRecognition;           //环岛识别
    CrossroadRecognition crossroadRecognition;      //十字道路处理
    GarageRecognition garageRecognition;            //车库识别
    FreezoneRecognition freezoneRecognition;        //泛型区识别类
    BusyareaDetection busyareaDetection;            //施工区检测
```

```
    GasStationDetection gasStationDetection;          //加油站检测
    FreezoneDetection freezoneDetection;              //泛行区检测类
    SlopeDetection slopeDetection;                    //坡道（桥）检测类
    uint16_t counterRunBegin = 1;                     //智能车启动计数器：等待
摄像头图像帧稳定
    RoadType roadType = RoadType::BaseHandle;         //初始化赛道类型
    uint16_t circlesThis = 1;                         //智能车当前运行的圈数
    uint16_t countercircles = 0;                      //圈数计数器

    // USB 转串口设备名 dev/ttyUSB0
    driver = std::make_shared<Driver>("/dev/ttyUSB0", BaudRate::BAUD_115200);
//初始化串口名
                if (driver == nullptr)  // 如果串口打开失败
                    {
                        std::cout << "Create Uart-Driver Error!" << std::endl;
//输出"串口创建失败"
                        return -1;
                    }

    //串口初始化，打开串口设备及配置串口数据格式
    int ret = driver->open();   //打开串口
                 if (ret != 0)
                    {
                        std::cout << "请检查 USB 接口是否松动" << std::endl;
//未打开串口，输出  "请检查 USB 接口是否松动" “uart open failed”
                        return -1;
                    }

    ipm.init(Size(COLSIMAGE, ROWSIMAGE), Size(COLSIMAGEIPM, ROWSIMAGEIPM));
// IPM（inverse perspective mapping）逆透视变换初始化

    signal(SIGINT, callbackSignal);     //程序退出
```

```
    motionController.loadParams();       //读 config.json 中的参数
    trackRecognition.rowCutUp = motionController.params.rowCutUp;   //切除图像中从上向下的行数
    trackRecognition.rowCutBottom  =  motionController.params.rowCutBottom; //切除图像中从下向上的行数
    garageRecognition.disGarageEntry                                    = motionController.params.disGarageEntry;  //车入库距离设定
    freezoneDetection.rateTurnFreezone                                  = motionController.params.rateTurnFreezone;  //FreeZone 转弯比率（转弯程度）大小控制【-1~1】
        if (motionController.params.GarageEnable)   //车出入库使能为真
            roadType = RoadType::GarageHandle;     //赛道类型为出库（最初状态）

        imagePreprocess.imageCorrecteInit();      // 镜头畸变矫正初始化

        if (motionController.params.debug)  //行为模式:debug
            {
                displayWindowDetailInit(); //显示窗口初始化
                detection                                               = Detection::DetectionInstance("../res/samples/sample.mp4", "../res/model/mobilenet-ssd-v2"); // debug 视频输入源：sample.mp4 预测用的模型：mobilenet-ssd-v2
                printAiEnable = true;   // AI 检测结果是否绘制：是
            }
            else    //行为模式：正常启动
    {
        cout << "准备发车!!!" << endl;
        detection       =       Detection::DetectionInstance("/dev/video0", "../res/model/mobilenet-ssd-v2");// 程序输入的视频源；摄像头  预测用的模型：mobilenet-ssd-v2
        printAiEnable = false;  // AI 检测结果是否绘制：否
```

```
        while (!driver->receiveStartSignal())   //下位机按键信号（start）信号接收
        {
            ;
        }
        cout << "--------- 系统启动成功!!! -------" << endl;    //成功接收，在终端输出："系统启动成功!!! "

        for (int i = 0; i < 20; i++) // 倒计时 2s 发车
            driver->carControl(0, PWMSERVOMID); //小车无动作，上位机开始接收下位机当前舵机 PWM 值（中值）
            waitKey(100);
        }
    }

    while (1)
    {
        bool imshowRec = false;     //AI 预测图像标注显示

        // #1: 视频源
        std::shared_ptr<DetectionResult>             resultAI             = detection->getLastFrame();      //获取 Paddle 多线程模型预测数据
        Mat frame = resultAI->rgb_frame;    //获取原始摄像头图像
                if (motionController.params.debug)
                {
                    savePicture(resultAI->det_render_frame);     //将处理后图像保存
                }
                else
                    {
                        if (motionController.params.saveImage) //保存原始图像
```

```
                savePicture(frame);
            }

        // #2: 图像预处理
        Mat imgaeCorrect = imagePreprocess.imageCorrection(frame);       //
将图像进放缩
        Mat imageBinary = imagePreprocess.imageBinaryzation(imgaeCorrect);
// 将 RGB 图像转二值图像

        // #3: 赛道识别
        trackRecognition.trackRecognition(imageBinary);     //车道线识别
                if (motionController.params.debug)      //如果打开 Debug 模式
                {
                    Mat imageTrack = imgaeCorrect.clone();      // 读取彩色
图像
                    trackRecognition.drawImage(imageTrack);      //在彩色图
像上显示车道线
                    savePicture(imageTrack);    //将图像保存
                }

        // 4： 加油站检测
        if (motionController.params.GasStationEnable) //检查是否使能加油站检
测
        {
         if (roadType == RoadType::GasstationHandle || roadType ==
RoadType::BaseHandle) //如果加油站检测与赛道检测共存
            {
         if      (gasStationDetection.gasStationDetection(trackRecognition,
resultAI->predictor_results))
                {
                 if (motionController.params.debug)  //如果行为设置为 Debug
                    {
                        Mat imageGasStation = Mat::zeros(Size(COLSIMAGE,
```

```
ROWSIMAGE), CV_8UC3); //图像初始化，以 8 位无符号整型存储的 RGB 图像
                        gasStationDetection.drawImage(trackRecognition);
//画出加油站和赛道边缘
                        imshow("imageRecognition", trackRecognition);    //
显示识别的图像
                        imshowRec = true;
                        savePicture(trackRecognition);   //存图
                    }
                    roadType = RoadType::GasstationHandle;  //状态切出
                }
                else
                    roadType = RoadType::BaseHandle;    //若不是加油站识别状
态，则为基本赛道识别状态

                if (gasStationDetection.slowDown) //如果是加油站状态，则减速
                    slowDownEnable();
            }
        }
        // 5：车库识别
        if (motionController.params.GarageEnable) //检查车库是否使能
        {
            if (roadType == RoadType::GarageHandle || roadType ==
RoadType::BaseHandle)     //如果车库是使能的
            {
                countercircles++;      //车辆运行圈数计数

                if
(garageRecognition.startingCheck(resultAI->predictor_results)) //如果检测到
车库起点
                {
                    ringRecognition.reset();     //初始化环岛识别
                    busyareaDetection.reset();   //初始化施工区识别
```

```
                    freezoneDetection.reset();   //初始化泛行区识别
                    gasStationDetection.reset(); //初始化加油站识别
                    freezoneRecognition.reset(); //初始化泛行区识别

                    if (countercircles >= 100)
                    {
                        circlesThis++;
                        countercircles = 0;
                    }
                }

                if  (circlesThis  >=  motionController.params.circles  &&
countercircles > 100) //如果达到预设圈数（config 里设置），则进入入库程序
                    garageRecognition.entryEnable = true;   //入库程序使能

                if   (garageRecognition.garageRecognition(trackRecognition,
resultAI->predictor_results))
                {
                    roadType = RoadType::GarageHandle;
                    if           (garageRecognition.garageStep           ==
garageRecognition.GarageEntryFinish) //入库动作正常结束
                    {
                        cout << ">>>>   成功入库    <<<<" << endl;
                        callbackSignal(0);  //回调 0
                    }
                    if (motionController.params.debug)  //如果是 Debug 模式
                    {
                        Mat   imageGarage   =   Mat::zeros(Size(COLSIMAGE,
ROWSIMAGE), CV_8UC3); //初始化图像
                        garageRecognition.drawImage(trackRecognition); //画
出赛道和车库边缘
                        imshow("imageRecognition", trackRecognition);    //
```

```
将车库图像显示在屏幕上
                        imshowRec = true;
                        savePicture(trackRecognition);   //存图
                    }
                }
                else
                    roadType = RoadType::BaseHandle;    //若不是车库识别状态，则为基本赛道识别状态

                if (garageRecognition.slowDown) //入库减速
                    slowDownEnable();   //减速
            }
        }


        // 6： 坡道检测
        if (motionController.params.SlopEnable) //检查是否使能坡道检测
        {
            if (roadType == RoadType::SlopeHandle || roadType == RoadType::BaseHandle)  //如果坡道检测与赛道检测共存
            {
                if (slopeDetection.slopeDetection(trackRecognition, resultAI->predictor_results))
                {
                    roadType = RoadType::SlopeHandle;   //状态切换
                    if (motionController.params.debug)  //如果是 Debug 模式
                    {
                        Mat imageFreezone = Mat::zeros(Size(COLSIMAGE, ROWSIMAGE), CV_8UC3);    //图像初始化，以 8 位无符号整型存储的 RGB 图像
                        slopeDetection.drawImage(trackRecognition);    //画出坡道和赛道边缘
                        imshow("imageRecognition", trackRecognition);  //显
```

```
示图像
                        imshowRec = true;
                        savePicture(trackRecognition); //存图
                    }
                }
                else
                    roadType = RoadType::BaseHandle;    //若不是坡道识别状态，则为基本赛道识别状态
            }
        }
        // 7： 施工区检测
        if (motionController.params.BusyAreaEnable) //检查是否使能施工区
        {
            if (roadType == RoadType::BusyareaHandle || roadType == RoadType::BaseHandle)  //如果施工区检测与赛道检测共存
            {
                if (busyareaDetection.busyareaDetection(trackRecognition, resultAI->predictor_results))
                {
                    if (motionController.params.debug)  //如果行为设置为Debug
                    {
                        Mat imageBusyarea = Mat::zeros(Size(COLSIMAGE, ROWSIMAGE), CV_8UC3); //图像初始化，以8位无符号整型存储的RGB图像
                        busyareaDetection.drawImage(trackRecognition);  //画出施工区和赛道边缘
                        imshow("imageRecognition", trackRecognition);  //显示图像
                        imshowRec = true;
                        savePicture(trackRecognition); //存图
                    }
                    roadType = RoadType::BusyareaHandle;    //状态切出
                }
```

```
                else
                    roadType = RoadType::BaseHandle;    //若不是施工区识别状
态，则为基本赛道识别状态

                if (busyareaDetection.slowDown) //如果是施工区状态，则减速
                    slowDownEnable();
            }
        }

        // 8: 环岛识别【废弃】
        if (motionController.params.RingEnable) //检查环岛是否使能
        {
            if (roadType == RoadType::RingHandle || roadType ==
RoadType::BaseHandle)
            {
                if (ringRecognition.ringRecognition(trackRecognition,
imageBinary))
                {
                    roadType = RoadType::RingHandle;
                    if (motionController.params.debug)
                    {
                        Mat imageRing = Mat::zeros(Size(COLSIMAGE, ROWSIMAGE),
CV_8UC3); //初始化
                        ringRecognition.drawImage(trackRecognition);
                        imshow("imageRecognition", trackRecognition);
                        imshowRec = true;
                        savePicture(trackRecognition);
                    }
                }
                else
                    roadType = RoadType::BaseHandle;
```

```
            }
        }

        // 9： 泛行区检测与识别
        if (motionController.params.FreezoneEnable) //检查泛行区是否使能
        {
            if (roadType == RoadType::FreezoneHandle || roadType == RoadType::BaseHandle)  //如果泛行区与塞东检测共存
            {
                if  (freezoneDetection.freezoneDetection(trackRecognition, resultAI->predictor_results))
                {
                    roadType = RoadType::FreezoneHandle;    //状态切换
                    if (motionController.params.debug)  //如果是 Debug 模式
                    {
                        Mat  imageFreezone  =  Mat::zeros(Size(COLSIMAGE, ROWSIMAGE), CV_8UC3);    //图像初始化，以 8 位无符号整型存储的 RGB 图像
                        freezoneDetection.drawImage(trackRecognition);   //画出泛行区和赛道边缘（禁行标志）
                        imshow("imageRecognition",       trackRecognition); //显示图像
                        imshowRec = true;
                        savePicture(trackRecognition); //存图
                    }
                    if (freezoneDetection.slowDown) //如果是泛行区，则减速
                        slowDownEnable();
                }
                else
                    roadType = RoadType::BaseHandle;    //若不是泛行区状态，则为基本赛道识别状态
            }
        }
```

```
        else    //如果没有使能，则进行不带 AI 的识别（等边三角形）【基本没用】
        {
            if (roadType == RoadType::FreezoneHandle || roadType == RoadType::BaseHandle)
            {
                if (freezoneRecognition.freezoneRecognition(trackRecognition, resultAI->predictor_results))
                {
                    roadType = RoadType::FreezoneHandle;
                    if (motionController.params.debug)
                    {
                        Mat imageFreezone = Mat::zeros(Size(COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化
                        freezoneRecognition.drawImage(trackRecognition);
                        imshow("imageRecognition", trackRecognition);
                        imshowRec = true;
                        savePicture(trackRecognition);
                    }
                }
                else
                    roadType = RoadType::BaseHandle;
            }
        }



        // 10： 控制参数计算
            if (trackRecognition.pointsEdgeLeft.size() < 30 && trackRecognition.pointsEdgeRight.size() < 30 && roadType != RoadType::FreezoneHandle && roadType != RoadType::SlopeHandle) //在车辆不在 Freezone 和 Slop 时检测不到赛道，则强行停止
            {
```

```
                counterOutTrackA++;
                counterOutTrackB = 0;
                if (counterOutTrackA > 20)
                    callbackSignal(0); //程序停止
            }
            else
            {
                counterOutTrackB++;
                if (counterOutTrackB > 50)
                {
                    counterOutTrackA = 0;
                    counterOutTrackB = 50;
                }
            }
        controlCenterCal.controlCenterCal(trackRecognition); //根据赛道边缘信息拟合运动控制计算
         // 11：十字路口识别
        if (motionController.params.CrossEnable) //检查十字路口是否使能
        {
            if (roadType == RoadType::CrossHandle || roadType == RoadType::BaseHandle)
            {
                if (crossroadRecognition.crossroadRecognition(trackRecognition, resultAI->predictor_results))
                {
                    roadType = RoadType::CrossHandle;

                    if (motionController.params.debug)
                    {
                        Mat imageCross = Mat::zeros(Size(COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化
```

```
                        crossroadRecognition.drawImage(trackRecognition, );
                        imshow("imageRecognition", trackRecognition);
                        imshowRec = true;
                        savePicture(trackRecognition);
                    }
                }
                else
                    roadType = RoadType::BaseHandle;
            }
        }


        // 12：运动控制
        if (counterRunBegin > 30)   //发车后前 30 帧图像不参与计算
        {
            //方向控制
            motionController.pdController(controlCenterCal.controlCenter);
// PD 控制器对车辆运动进行控制

            //速度控制状态机
                switch (roadType)
                {
                case RoadType::FreezoneHandle:      //泛行区处理速度
                    motionController.speedController(true,        slowDown,
controlCenterCal);  //变加速控制
                    break;
                case RoadType::GasstationHandle:    //加油站速度
                    motionController.motorSpeed                              =
motionController.params.speedGasBusy; //匀速控制
                    break;
                case RoadType::BusyareaHandle:      //施工区速度
                    motionController.motorSpeed                              =
```

```
motionController.params.speedGasBusy; //匀速控制
                    break;
                case RoadType::SlopeHandle:     //坡道速度
                    motionController.motorSpeed                                    =
motionController.params.speedSlop; //匀速控制
                    break;
                case RoadType::GarageHandle:    //出入库速度
                    motionController.motorSpeed                                    =
motionController.params.speedGarage; //匀速控制
                    break;
                default:    //赛道速度
                    motionController.speedController(true,         slowDown,
controlCenterCal); //变加速控制
                    break;
                }

            if (!motionController.params.debug) //debug 模式下车辆不启动
            {
                driver->carControl(motionController.motorSpeed,
motionController.servoPwm); //串口通信，姿态与速度控制
            }

            if (slowDown)//减速缓冲
            {
                counterSlowDown++;
                if (counterSlowDown > 50)
                {
                    slowDown = false;
                    counterSlowDown = 0;
                }
            }
        }
```

```
        else
            counterRunBegin++;

        // 13： Debug 下的存图与说明
     if (motionController.params.debug)
        {
            controlCenterCal.drawImage(trackRecognition, imgaeCorrect);
            switch (roadType)
            {
            case                                    RoadType::BaseHandle:
//赛道识别
                putText(imgaeCorrect, "[1] 赛 道 ", Point(10, 20),
cv::FONT_HERSHEY_TRIPLEX, 0.2, cv::Scalar(0, 150, 255), 1, CV_AA);      //
显示赛道类型
                break;
            case                                    RoadType::RingHandle:
//环岛识别
                putText(imgaeCorrect, "[1] 环 岛 ", Point(10, 20),
cv::FONT_HERSHEY_TRIPLEX, 0.2, cv::Scalar(0, 255, 150), 1, CV_AA);      //
显示赛道识别类型
                break;
            case                                   RoadType::CrossHandle:
//十字路口识别
                putText(imgaeCorrect, "[1] 十 字 路 口 ", Point(10, 20),
cv::FONT_HERSHEY_TRIPLEX, 0.2, cv::Scalar(0, 255, 150), 1, CV_AA);      //
显示赛道识别类型
                break;
            case                                RoadType::FreezoneHandle:
//泛行区识别
                putText(imgaeCorrect, "[1] 泛 行 区 ", Point(10, 20),
cv::FONT_HERSHEY_TRIPLEX, 0.2, cv::Scalar(0, 255, 150), 1, CV_AA);      //
显示赛道识别类型
                break;
```

```
            case                                    RoadType::GarageHandle:
//车库识别
                putText(imgaeCorrect,  "[1]  车 库 ",  Point(10,  20),
cv::FONT_HERSHEY_TRIPLEX, 0.2, cv::Scalar(0, 255, 150), 1, CV_AA);        //
显示赛道识别类型
                break
            case                                  RoadType::GasstationHandle:
// 加油站识别
                putText(imgaeCorrect,  "[1]  加 油 站 ",  Point(10,  20),
cv::FONT_HERSHEY_TRIPLEX, 0.2, cv::Scalar(0, 255, 150), 1, CV_AA);        //
显示赛道识别类型
                break;
            case                                    RoadType::BusyareaHandle:
//施工区处理
                putText(imgaeCorrect,  "[1]  施 工 区 ",  Point(10,  20),
cv::FONT_HERSHEY_TRIPLEX, 0.2, cv::Scalar(0, 255, 150), 1, CV_AA);        //
显示赛道识别类型
                break;
            case                                    RoadType::SlopeHandle:
//坡道处理[Useless]
                putText(imgaeCorrect,  "[1]  坡 道 ",  Point(10,  20),
cv::FONT_HERSHEY_TRIPLEX, 0.2, cv::Scalar(0, 255, 150), 1, CV_AA);        //
显示赛道识别类型
                break;
            }



            putText(imgaeCorrect, str, Point(COLSIMAGE - 50, ROWSIMAGE - 20),
cv::FONT_HERSHEY_TRIPLEX, 0.5, cv::Scalar(0, 255, 0), 1, CV_AA);      //圈数
显示
            if                                                  (!imshowRec)
//图像存储顺序与显示顺序一致
            {
                Mat  imageNone  =  Mat::zeros(Size(COLSIMAGE,  ROWSIMAGE),
```

```
CV_8UC3); //初始化
                imshow("imageRecognition", imageNone);
                savePicture(imageNone);
            }
            imshow("imageControl", imgaeCorrect);
            savePicture(imgaeCorrect);

            char c = waitKey(10);
        }
    }

    return 0;
}

/* 系统信号回调函数：系统退出 BISTU-FAI-CRY v2.0 */
void callbackSignal(int signum)
{
    driver->carControl(0, PWMSERVOMID); //智能车停止运动
    cout << "====程序结束 车辆停止！==== " << signum << endl;
    exit(signum);
}

/*OpenCV 图像显示窗口初始化（详细参数/Debug 模式） BISTU-FAI-CRY v2.0 */
void displayWindowDetailInit(void)
{
    //[1] 二值化图像：Gray
    string windowName = "imageTrack";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 320, 240);     //分辨率
    cv::moveWindow(windowName, 10, 10);         //布局位置

    //[2] 赛道边缘图像：RGB
```

```
    windowName = "imageRecognition";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 320, 240);     //分辨率
    cv::moveWindow(windowName, 10, 320);        //布局位置

    //[3] 原始图像/矫正后：RGB
    windowName = "imageControl";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 640, 480);     //分辨率
    cv::moveWindow(windowName, 350, 20);        //布局位置
}

/*OpenCV 图像显示窗口初始化（详细参数/看图模式） BISTU-FAI-CRY v2.0 */
void displayWindowImageInit(void)
{
    //[1] 原始图像/矫正后：RGB
    std::string windowName = "frame";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 320, 240);     //分辨率
    cv::moveWindow(windowName, 10, 10);         //布局位置

    //[2] 二值化图像：Gray
    windowName = "imageBinary";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 320, 240);     //分辨率
    cv::moveWindow(windowName, 350, 10);        //布局位置

    //[3] 有效路径图像：Gray
    windowName = "imagePath";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 320, 240);     //分辨率
    cv::moveWindow(windowName, 690, 10);        //布局位置
```

```
    //[4] 赛道边缘图像：RGB
    windowName = "imageTrack";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 320, 240);     //分辨率
    cv::moveWindow(windowName, 10, 320);        //布局位置

    //[5] 特殊元素处理图像：RGB
    windowName = "imageRecognition";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 320, 240);     //分辨率
    cv::moveWindow(windowName, 350, 320);       //布局位置

    //[6] 车辆控制图像：RGB
    windowName = "imageControl";
    cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
    cv::resizeWindow(windowName, 320, 240);     //分辨率
    cv::moveWindow(windowName, 690, 320);       //布局位置
}

/* 车辆减速使能 BISTU-FAI-CRY v2.0 */

void slowDownEnable(void)
{
    slowDown = true;
    counterSlowDown = 0;
}
```

下位机

```
#include "soc.h"

SocStruct socStr;
```

```
void SOC_IIC_Init(void)
{
    gpio_init(P02_8, GPO, 1, PUSHPULL);
    gpio_init(P02_7, GPO, 1, PUSHPULL);
    SOC_IIC_SCL(1);
    SOC_IIC_SDA(1);
}

void SOC_IIC_Start(void)
{
        SOC_SDA_OUT();      //sda 线输出
        SOC_IIC_SDA(1);
        SOC_IIC_SCL(1);
        systick_delay_us(STM0,8);
        SOC_IIC_SDA(0);                 //START:when CLK is high,DATA change
form high to low
        systick_delay_us(STM0,8);
        SOC_IIC_SCL(0);              //准备发送或接收数据
}

void SOC_IIC_Stop(void)
{
        SOC_SDA_OUT();              //sda 线输出
        SOC_IIC_SCL(0);
        SOC_IIC_SDA(0);              //STOP:when CLK is high DATA change form
low to high
        systick_delay_us(STM0,8);
        SOC_IIC_SCL(1);
        SOC_IIC_SDA(1);              //发送结束信号
        systick_delay_us(STM0,8);
}
```

```
uint8 SOC_IIC_Wait_Ack(void) //应答等待
{
        uint8 ucErrTime=0;
        SOC_SDA_IN();                           //SDA 设置为输入
        SOC_IIC_SDA(1);systick_delay_us(STM0,2);
        SOC_IIC_SCL(1);systick_delay_us(STM0,2);

        while(SOC_READ_SDA)
        {
            ucErrTime++;
            if(ucErrTime>250)
            {
                SOC_IIC_Stop();
                return 1;
            }
        }

        SOC_IIC_SCL(0);                         //时钟输出 0

        return 0;
}

void SOC_IIC_Ack(void)
{
        SOC_IIC_SCL(0);
        SOC_SDA_OUT();
        SOC_IIC_SDA(0);
        systick_delay_us(STM0,4);
        SOC_IIC_SCL(1);
        systick_delay_us(STM0,4);
        SOC_IIC_SCL(0);
```

```
}

void SOC_IIC_NAck(void)
{
        SOC_IIC_SCL(0);
        SOC_SDA_OUT();
        SOC_IIC_SDA(1);
        systick_delay_us(STM0,4);
        SOC_IIC_SCL(1);
        systick_delay_us(STM0,4);
        SOC_IIC_SCL(0);
}

void SOC_IIC_Send_Byte(uint8 txd)
{
    uint8 t;
    SOC_SDA_OUT();
    SOC_IIC_SCL(0);                    //拉低时钟开始数据传输

    for(t=0;t<8;t++)
    {
            SOC_IIC_SDA((txd&0x80)>>7);
            txd<<=1;
            systick_delay_us(STM0,4);                  //对 TEA5767 这三个延时
都是必须的
            SOC_IIC_SCL(1);
            systick_delay_us(STM0,4);
            SOC_IIC_SCL(0);
            systick_delay_us(STM0,4);
    }
}
```

```
uint8 SOC_IIC_Read_Byte(unsigned char ack)
{
        unsigned char i,receive=0;
        SOC_SDA_IN();                                   //SDA 设置为输入

        for(i=0;i<8;i++ )
        {
            SOC_IIC_SCL(0);
            systick_delay_us(STM0,4);
            SOC_IIC_SCL(1);
            receive<<=1;

            if(SOC_READ_SDA)receive++;
            systick_delay_us(STM0,2);
        }

        if (!ack)
            SOC_IIC_NAck();//发送 nACK
        else
            SOC_IIC_Ack(); //发送 ACK

        return receive;
}

uint8 SOC_Write_Len(uint8 reg,uint8 len,uint8 *buff)
{
        uint8 i;
        SOC_IIC_Start();
        SOC_IIC_Send_Byte(WRITE_CW2015);
        if(SOC_IIC_Wait_Ack())                          //等待应答
        {
            SOC_IIC_Stop();
```

```
            return 1;
        }

        SOC_IIC_Send_Byte(reg);                         //写寄存器地址
        SOC_IIC_Wait_Ack();                             //等待应答

        for(i=0;i<len;i++)
        {
            SOC_IIC_Send_Byte(buff[i]);                 //发送数据
            if(SOC_IIC_Wait_Ack())                      //等待 ACK
            {
                SOC_IIC_Stop();
                return 1;
            }
        }
        SOC_IIC_Stop();

        return 0;
}

uint8 SOC_Write(uint8 reg,uint8 *buf)
{
        SOC_IIC_Start();
        SOC_IIC_Send_Byte(WRITE_CW2015);
        if(SOC_IIC_Wait_Ack())                          //等待应答
        {
            SOC_IIC_Stop();
            return 1;
        }

        SOC_IIC_Send_Byte(reg);                         //写寄存器地址
        SOC_IIC_Wait_Ack();                             //等待应答
```

```
        SOC_IIC_Send_Byte(*buf);                    //发送数据

        if(SOC_IIC_Wait_Ack())                      //等待 ACK
        {
            SOC_IIC_Stop();
            return 1;
        }

        SOC_IIC_Stop();

        return 0;
}

uint8 SOC_Read_Len(uint8 reg,uint8 len,uint8 *buff)
{
        SOC_IIC_Start();
        SOC_IIC_Send_Byte(WRITE_CW2015);
        if(SOC_IIC_Wait_Ack())                                  //等待应答
        {
            SOC_IIC_Stop();
            return 1;
        }
        SOC_IIC_Send_Byte(reg);                                 //写寄存器地址
        SOC_IIC_Wait_Ack();                                     //等待应答
        SOC_IIC_Start();
        SOC_IIC_Send_Byte(READ_CW2015);
        SOC_IIC_Wait_Ack();                                     //等待应答

        while(len)
        {
            if(len==1)*buff=SOC_IIC_Read_Byte(0);   //读数据,发送 nACK
```

```
            else *buff=SOC_IIC_Read_Byte(1);            //读数据,发送 ACK
            len--;
            buff++;
        }
        SOC_IIC_Stop();                                 //产生
一个停止条件

        return 0;
}

uint8 SOC_Read(uint8 reg,uint8 *buff)
{
        SOC_IIC_Start();
        SOC_IIC_Send_Byte(WRITE_CW2015);
        if(SOC_IIC_Wait_Ack())                          //等待应答
        {
            SOC_IIC_Stop();
            return 1;
        }
        SOC_IIC_Send_Byte(reg);                         //写寄存器地址
        SOC_IIC_Wait_Ack();                             //等待应答
        SOC_IIC_Start();
        SOC_IIC_Send_Byte(READ_CW2015);
        SOC_IIC_Wait_Ack();                             //等待应答
        *buff = SOC_IIC_Read_Byte(0);                   //读数据,发送 nACK
        SOC_IIC_Stop();                                 //产生
一个停止条件

        return 0;
}
//-----------------------------------------------[END]-------------------
-----------------------------------------
```

```
//-----------------------------------------------[UNIT-SOC-LOGIC]-----------
-----------------------------------------------
int8 no_charger_full_jump = 0;
unsigned int allow_no_charger_full =0;
unsigned int allow_charger_always_zero =0;
unsigned char if_quickstart =0;
unsigned char reset_loop =0;
unsigned char SOC_UpdataConfigInfo(void)
{
    int8 ret = 0;
    unsigned char i;
    unsigned char reset_val;
    unsigned char reg_val;
    /* make sure no in sleep mode */
    ret = SOC_Read(REG_MODE, &reg_val);
    if(ret)
    {
        return 1;
    }
    if((reg_val & MODE_SLEEP_MASK) == MODE_SLEEP)
    {
        return 2;
    }
    /* update new battery info */
    for(i = 0; i < SIZE_BATINFO; i++)
    {
        reg_val = cw_bat_config_info[i];
```

```
        ret = SOC_Write(REG_BATINFO+i, &reg_val);
        if(ret)
        {
            return 1;
        }
    }
    for(i = 0; i < SIZE_BATINFO; i++)
    {
        ret = SOC_Read(REG_BATINFO+i, &reg_val);
        if(ret)
        {
            return 1;
        }
        if(reg_val != cw_bat_config_info[i])
        {
            return 3;
        }
    }
    ret = SOC_Read(REG_CONFIG, &reg_val);
    if(ret)
    {
        return 1;
    }
    reg_val |= CONFIG_UPDATE_FLG;   /* set UPDATE_FLAG */
    reg_val &= 0x07;                /* clear ATHD */
    reg_val |= ATHD;                /* set ATHD */
    ret = SOC_Write(REG_CONFIG, &reg_val);
    if(ret)
    {
        return 1;
    }
    reset_val = MODE_NORMAL;
```

```
    reg_val = MODE_RESTART;
    ret = SOC_Write(REG_MODE, &reg_val);
    if(ret)
    {
        return 1;
    }
    systick_delay_us(STM0,100);  //delay 100us
    ret = SOC_Write(REG_MODE, &reset_val);
    if(ret)
    {
        return 1;
    }
    return 0;
}
unsigned char SOC_HardwareInit(void)
{
    unsigned char ret;
    unsigned char i;
    unsigned char reg_val = MODE_NORMAL;
    ret = SOC_Write(REG_MODE, &reg_val);
    if(ret)
    {
        return 1;
    }
    ret = SOC_Read(REG_CONFIG, &reg_val);
    if(ret)
    {
        return 1;
    }
    if((reg_val & 0xf8) != ATHD)
    {
        reg_val &= 0x07;
```

```
        reg_val |= ATHD;
        ret = SOC_Write(REG_CONFIG, &reg_val);
        if(ret)
        {
            return 1;
        }
    }
    ret = SOC_Read(REG_CONFIG, &reg_val);
    if(ret)
    {
        return 1;
    }
    if(!(reg_val & CONFIG_UPDATE_FLG))
    {
        ret = SOC_UpdataConfigInfo();
        if(ret)
        {
            return ret;
        }
    }
    else
    {
        for(i = 0; i < SIZE_BATINFO; i++)
        {
            ret = SOC_Read(REG_BATINFO +i, &reg_val);
            if(ret)
            {
                return 1;
            }
            if(cw_bat_config_info[i] != reg_val)
            {
                break;
```

```
            }
        }
        if(i != SIZE_BATINFO)
        {
            ret = SOC_UpdataConfigInfo();
            if(ret)
            {
                return ret;
            }
        }
    }
    for (i = 0; i < 30; i++) {
        ret = SOC_Read(REG_SOC, &reg_val);
        if (ret)
            return 1;
        else if (reg_val <= 100)
            break;
        systick_delay_ms(STM0,100);//delay 100ms
    }

    if (i >=30){
        reg_val = MODE_SLEEP;
        ret = SOC_Write(REG_MODE, &reg_val);
        return 4;
    }
    return 0;
}

#ifdef BAT_LOW_INTERRUPT
```

```
unsigned char SOC_ReleaseAlrtPin(void)
{
        signed char ret = 0;
        unsigned int reg_val;
        unsigned char alrt;

        ret = SOC_Read(REG_RRT_ALERT, &reg_val);
        if(ret)
        {
            return -1;
        }
        alrt = reg_val & 0x80;

        reg_val = reg_val & 0x7f;
        ret = SOC_Write(REG_RRT_ALERT, &reg_val);
        if(ret)
        {
            return -1;
        }

        return alrt;
}



int8_t SOC_UpdateAthd()
{
    int8_t ret = 0;
    unsigned char reg_val;
    char new_athd = 0;

    ret = SOC_Read(REG_CONFIG, &reg_val);
    if(ret)
```

```
    {
        return -1;
    }
    new_athd = (reg_val >> 3) - 1;
    if(new_athd <= 0){
        new_athd = 0;
    }
    new_athd = new_athd << 3;

    //"the new ATHD need set"
    reg_val &= 0x07;    /* clear ATHD */
    reg_val |= new_athd;    /* set new ATHD */
    ret = SOC_Write(REG_CONFIG, &reg_val);
    if(ret)
    {
            return -1;
    }
    return 0;
}

static void ALRT_ISR() //interrupt
{
    SOC_ReleaseAlrtPin();
        SOC_UpdateAthd();
    }
#endif

int8 SOC_Por(void)
{
    int8 ret = 0;
    unsigned char reset_val = 0;
    reset_val = MODE_SLEEP;
```

```
    ret = SOC_Write(REG_MODE, &reset_val);
    if (ret)
        return -1;
    systick_delay_us(STM0,100); //delay 100us

    reset_val = MODE_NORMAL;
    ret = SOC_Write(REG_MODE, &reset_val);
    if (ret)
        return -1;
    systick_delay_us(STM0,100); //delay 100us

    ret = SOC_HardwareInit();
    if (ret)
        return ret;
    return 0;
}



unsigned long SOC_GetVol(void)
{
    int8 ret = 0;
    unsigned char get_ad_times = 0;
    unsigned char reg_val[2] = {0 , 0};
    unsigned long ad_value = 0;
    unsigned int ad_buff = 0;
    unsigned int ad_value_min = 0;
    unsigned int ad_value_max = 0;

    for(get_ad_times = 0; get_ad_times < 3; get_ad_times++)
    {
        ret = SOC_Read(REG_VCELL, &reg_val[0]);
        if(ret)
```

```
        {
            return 1;
        }
        ret = SOC_Read(REG_VCELL + 1, &reg_val[1]);
        if(ret)
        {
            return 1;
        }
        ad_buff = (reg_val[0] << 8) + reg_val[1];

        if(get_ad_times == 0)
        {
            ad_value_min = ad_buff;
            ad_value_max = ad_buff;
        }
        if(ad_buff < ad_value_min)
        {
            ad_value_min = ad_buff;
        }
        if(ad_buff > ad_value_max)
        {
            ad_value_max = ad_buff;
        }
        ad_value += ad_buff;
    }
    ad_value -= ad_value_min;
    ad_value -= ad_value_max;
    ad_value = ad_value  * 305 / 1000;
    return(ad_value);       //14 位 ADC 转换值
}

void SOC_UpdateCapacity(void)
```

```
{
    int cw_capacity;
    cw_capacity = SOC_GetCapacity();
    if((cw_capacity >= -5) && (cw_capacity <= 100) && (socStr.Capacity != cw_capacity))
    {
        socStr.Capacity = cw_capacity;
        cw_capacity += 5;                //优化电池电量充不满的问题
        if(cw_capacity>100)
            icarStr.Electricity = 100;
        else if(cw_capacity<0)
            icarStr.Electricity = 0;
        else
            icarStr.Electricity = cw_capacity;
    }
}

uint8 Index = 0;
uint8 uSocErrorCnt= 0;
void SOC_UpdateVol(void)
{
    unsigned long cw_voltage;
    cw_voltage = SOC_GetVol();
    if(cw_voltage == 1)
    {

    }else if(socStr.voltage != cw_voltage)
    {
        socStr.voltage = cw_voltage;
        icarStr.Voltage = (float)cw_voltage*3.f/1000.f;
    }
```

```
    if(socStr.voltage < 11.1f)  //电量低于 5% || 电压<11.1v 开始报警
    {
        uSocErrorCnt ++;
        if(uSocErrorCnt > 2)
        {
            GPIO_BuzzerEnable(BuzzerWarnning);
        }
    }
    else
    {
        uSocErrorCnt = 0;
    }
}

void SOC_UpdateUsbOnline(void)
{
    if(0)
    {
        socStr.UsbOnline = 1;
    }else{
        socStr.UsbOnline = 0;
    }
}



void SOC_BatWork(void)
{
    SOC_UpdateUsbOnline();
    SOC_UpdateVol();
    SOC_UpdateCapacity();
}
```

```
unsigned char SOC_Init(void)
{
    unsigned char ret;
    unsigned char loop = 0;
    SOC_IIC_Init();

    ret = SOC_HardwareInit();

    while((loop++ < 200) && (ret != 0))
    {
        ret = SOC_HardwareInit();
    }

    if(loop >= 200 && ret != 0)
    {
        icarStr.errorCode |= (1<<9);
    }
    else
    {
        icarStr.errorCode &= ~(1<<9);
    }

    socStr.UsbOnline = 0;
    socStr.Capacity = 2;
    socStr.voltage = 0;

    return ret;
}


void SOC_Timer(void)
```

```
{
    socStr.Counter++;
    if(socStr.Counter>200000)
        socStr.Counter = 200000;
}

void SOC_Handle(void)
{
    if(socStr.Counter>1000)//1s
    {
        SOC_BatWork();
        socStr.Counter = 0;
    }
}
//-------------------------------------------------------          Bistu-FAI
------------------------------------------------------------




#ifndef __SOC_H__
#define __SOC_H__

/*--------------------------------------      I  N  C  L  U  D  E  S
---------------------------------------*/
#include "headfile.h"



/*-------------------------------------    D  E  F  I  N  I  T  I  O  N
---------------------------------------*/
//IIC
#define SOC_SDA_IN()                        {gpio_dir(P02_8, GPI, PULLUP  );}
//P02_8 输入模式
```

```
#define SOC_SDA_OUT()                          {gpio_dir(P02_8, GPO, PUSHPULL);}
//P02_8 输出模式
#define  SOC_IIC_SCL(N)                                    gpio_set(P02_7,N)
//SCL
#define  SOC_IIC_SDA(N)                                    gpio_set(P02_8,N)
//SDA
#define  SOC_READ_SDA                                       gpio_get(P02_8)
//输入 SDA



#define READ_CW2015                          0xc5
#define WRITE_CW2015                         0xc4

#define REG_VERSION             0x0
#define REG_VCELL               0x2
#define REG_SOC                 0x4
#define REG_RRT_ALERT           0x6
#define REG_CONFIG              0x8
#define REG_MODE                0xA
#define REG_BATINFO             0x10

#define MODE_SLEEP_MASK         (0x3<<6)
#define MODE_SLEEP              (0x3<<6)
#define MODE_NORMAL             (0x0<<6)
#define MODE_QUICK_START        (0x3<<4)
#define MODE_RESTART            (0xf<<0)
#define CONFIG_UPDATE_FLG       (0x1<<1)
#define ATHD                    (0x0<<3)        //ATHD = 0%

#define SIZE_BATINFO                64

#define BATTERY_UP_MAX_CHANGE   720             // the max time allow battery
```

```
change quantity
#define BATTERY_DOWN_MIN_CHANGE 60              // the min time allow battery
change quantity when run
#define BATTERY_DOWN_MIN_CHANGE_SLEEP 1800      // the min time allow battery
change quantity when run 30min
//#define BAT_LOW_INTERRUPT    1
typedef struct
{
    unsigned char UsbOnline;                                    //USB 插入状态
    unsigned int Capacity;                                      //电量
    unsigned int voltage;                                       //电压值
    uint32       Counter;                                       //计数器
}SocStruct;



extern SocStruct socStr;

void SOC_IIC_Init(void);                                        //IIC-IO 初始化
void SOC_IIC_Start(void);                                       //产生 IIC 起始信
号
void SOC_IIC_Stop(void);                                        //产生 IIC 停止信
号
uint8 SOC_IIC_Wait_Ack(void);                                    //等待应答信号
的到来
void SOC_IIC_Ack(void);                                         //产生 ACK 应答
void SOC_IIC_NAck(void);                                        //不产生 ACK 应答
void SOC_IIC_Send_Byte(uint8 txd);                              //IIC 发送一个字
节
uint8 SOC_IIC_Read_Byte(unsigned char ack);                     //读取一个字节
uint8 SOC_Write_Len(uint8 reg,uint8 len,uint8 *buf);            //IIC 连续写指定
长度数据
uint8 SOC_Write(uint8 reg,uint8 *buf);                          //IIC 写入数据
uint8 SOC_Read_Len(uint8 reg,uint8 len,uint8 *buf);             //IIC 连续读取指
```

```
定长度数据
uint8 SOC_Read(uint8 reg,uint8 *buf);                          //IIC 读取指定地
址的数据


unsigned char SOC_HardwareInit(void);                          //电量计底层初始
化
unsigned char SOC_UpdataConfigInfo(void);                      //更新电池信息
int8 SOC_Por(void);                                            //电量计上电复
位
int SOC_GetCapacity(void);                                     //获取电量
unsigned long SOC_GetVol(void);                                //获取电压
void SOC_UpdateCapacity(void);                                 //SOC 更新电池电
量
void SOC_UpdateVol(void);                                      //SOC 更新电池电
压
void SOC_UpdateUsbOnline(void);                                //USB 插入状态检
测
void SOC_BatWork(void);                                        //SOC 控制器


unsigned char SOC_Init(void);                                  //SOC 初始化
void SOC_Timer(void);                                          //电量计 IC 控制
时序
void SOC_Handle(void);                                         //电量计 IC 控制
逻辑
#endif



#include "headfile.h"
IcarStruct icarStr;



/* 智能车参数初始化 BISTU-FAI-CRY v2.0 */
void ICAR_Init(void)
```

```
{
    icarStr.Electricity = 0;
    icarStr.Voltage = 0;
    icarStr.SpeedSet = 0.0f;
    icarStr.SpeedFeedback = 0.0f;
    icarStr.SpeedMaxRecords = 0.0f;
}

/* 浮点数排序 BISTU-FAI-CRY v2.0 */

void ICAR_SortFloat(float *buff, uint8 start, uint8 len)
{
    float n = 0;
        for(uint8 i = start; i < start+len; i++)
        {
                for(uint8 j = i+1; j<start+len; j++)
                {
                        if(buff[i] > buff[j])
                        {
                            n = buff[i];
                            buff[i] = buff[j];
                            buff[j] = n;
                        }
                }
        }
}

/* 浮点数绝对值函数 BISTU-FAI-CRY v2.0 */
float CMATH_AbsFloat(float data)
{
    if(data < 0)
        return -data;
```

```
    else
        return data;
}



#ifndef __ICAR_H__
#define __ICAR_H__

/*----------------------------------------     I  N  C  L  U  D  E  S
----------------------------------------*/
#include "headfile.h"



/*---------------------------------------    D  E  F  I  N  I  T  I  O  N
---------------------------------------*/

/* 智能车自检步骤 BISTU-FAI-CRY v2.0 */
typedef enum
{
    Selfcheck_None = 0,             //开始测试
    Selfcheck_MotorA,               //电机正转启动
    Selfcheck_MotorB,               //电机正转采样
    Selfcheck_MotorC,               //电机反转启动
    Selfcheck_MotorD,               //电机反转采样
    Selfcheck_MotorE,               //电机闭环正转启动
    Selfcheck_MotorF,               //电机闭环正转采样
    Selfcheck_MotorG,               //电机闭环反转启动
    Selfcheck_MotorH,               //电机闭环反转采样
    Selfcheck_ServoA,               //舵机测试 A
    Selfcheck_Com,                  //通信测试
    Selfcheck_Buzzer,               //蜂鸣器测试
    Selfcheck_RgbLed,               //灯效测试
```

```
    Selfcheck_Key,                  //按键测试
    Selfcheck_Finish                //测试完成
}Selfcheck_Enum;

/* 智能车相关 BISTU-FAI-CRY v2.0 */
typedef struct                                  //[智能车驱动主板]
{
    float   Voltage;                            //电池电压
    uint8   Electricity;                        //电池电量百分比:  0~100
    float   SpeedSet;                           //电机目标速度:     m/s
    float   SpeedFeedback;                      //电机模型实测速度:  m/s
    float   SpeedMaxRecords;                    //测试记录最高速
    uint16  ServoPwmSet;                        //舵机 PWM 设置
    uint16  errorCode;                          //错误代码

    uint16     counterKeyA;                       //按键模式 A 计数器
    boolean    keyPressed;                        //按键按下
    boolean    motorEnable;                       //电机使能标志
    boolean    sprintEnable;                      //闭环冲刺使能
    uint16     counterSprint;                     //闭环冲刺时间

    boolean     selfcheckEnable;                  //智能车自检使能
    uint16      counterSelfcheck;                 //自检计数器
    uint8       timesSendStep;                    //发送超时数据次数
    uint16      counterModuleCheck;               //自检计数器
    Selfcheck_Enum  selfcheckStep;          //自检步骤
    uint8           speedSampleStep;        //速度采样步骤
}IcarStruct;



extern IcarStruct icarStr;
```

```
void ICAR_Init(void);
void ICAR_SortFloat(float *buff, uint8 start, uint8 len);
float CMATH_AbsFloat(float data);
#endif

//=========================================                    BISTU-FAI
=========================================//
```