

第十五届全国大学生 智能汽车竞赛



声音信标组 技术报告

学 校： 南昌大学

队伍名称： 红黑红黑的小黄鸡二队

参赛队员： 胡紫灿

江力楚

张风娟

带队教师： 余运俊

李春泉

关于技术报告和学术论文使用授权的说明

本人完全了解第十五届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，比赛组委会和恩智浦半
导体公司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：江力楚 胡紫灿 张凤娟
带队老师签名：余运俊 李春泉
日 期：2020.8.30

目录

摘要	3
第一章 引言	4
第二章 系统框架	5
2.1 系统总体设计方案	5
2.2系统工作流程	5
2.3整车布局	6
第三章 智能车机械结构优化	7
3.1麦克风模块的安装	7
3.2碳纤维板的安装	7
3.3天线的安装	8
3.4激光雷达的安装	8
3.5编码器的安装和齿轮啮合	8
3.6 磁标的安装	9
3.7 小结	9
第四章 电路设计	10
4.1主板设计	10
4.2 驱动板设计	11
4.3 转接板设计	12
4.4 陀螺仪设计	13
第五章 软件设计	14
5.1 程序流程图	14
5.2 声音信号采集及处理	15
5.2.1 信号采集	15
5.2.1 信号处理	15
5.3 小车运动控制	17
5.3.1 PID算法介绍	17
5.3.1 方向控制	18
5.3.2 速度控制	18
5.4 避障策略	19
5.4.1 障碍为目标灯	20
5.4.2 障碍为非目标灯	20
第六章 智能车调试	21
6.1 嵌入式平台	21
6.2 仿真软件	22
6.3 上位机调试工具	23
6.4小结	23
第七章 总结	24
参考文献	25
附录A 信号处理核心代码	26
附录B 控制部分核心代码	35
附录C主板原理图和PCB	49
附录D 驱动原理图和PCB	50

摘要

本文详细介绍了我们为了第十五届全国大学生智能汽车竞赛声音信标组准备的智能小车。小车采用恩智浦公司的RT1064芯片作为控制器；通过MAX9814和RDA5807模块采集声音信号和射频信号实现信标定位；通过北醒公司的Tfmini plus采集障碍信息；利用编码器测量车模速度，构成速度闭环控制；通过陀螺仪角速度反馈，使小车运动更加平稳；利用蓝牙、上位机、OLED屏幕等对程序进行调试，获得更好的运行效果。

关键词： 智能车 单片机 PID RT1064 麦克纳姆轮 声音定位

第一章 引言

全国大学生智能车竞赛是从 2006 开始，由教育部高等教育司委托高等学校自动化类教学指导委员会举办的旨在加强学生实践、创新能力和培养团队精神的一项创意性科技竞赛，至今已经成功举办了十四届。在继承和总结前十四届比赛实践的基础上，竞赛组委会努力拓展新的竞赛内涵，设计新的竞赛内容，创造新的比赛模式，使得围绕该比赛所产生的竞赛生态环境得到进一步的发展。

为了实现竞赛的"立足培养、重在参与、鼓励探索、追求卓越"的指导思想，竞赛内容设置需要能够面向大学本科阶段的学生和教学内容，同时又能够兼顾当今时代科技发展的新趋势。比赛形式包括有竞速比赛与创意比赛两大类。竞速比赛中包含不同的组别，难度适合本科不同年级本科生学生参赛。在竞速赛基础上，适当增加挑战性，形成创意比赛的内容，适合部分有条件、能力强的本科生和研究参加创意比赛。

声音信标组须使用竞赛秘书处统一指定的竞赛 H 车模套件，指定使用英飞凌系列单片机，因南昌大学被列入美国实体制裁清单，可采用 STC 和 NXP 系列芯片。在本次比赛中，我们采用了恩智浦公司的微控制器 i.MX RT1064DVL6A 作为核心控制单元，自主构思控制方案及系统设计，融合声音信号采集处理、各类传感器的运用、电机驱动输出，最终实现快速平稳地完成寻找信标灯的任务。

在十一届“恩智浦”杯全国大学生智能汽车竞赛中，首次以常规组的形式出现了信标组。该组抛弃了常规的赛道，利用信标灯作为引导方式，选手需要制作智能车实现自主寻灯和路径规划，后面引进双车对抗，双方选手需要在同一场地实现寻灯任务，在原有的基础上，增加了对方车辆的不确定因素，使得难度大大提升，后来在十四届又更换了 H 车模，H 车模的麦克纳姆轮控制给信标寻灯方案带来了更多的可能。今年信标灯采用的是声音和无线射频作为引导信号，同时信标灯的形状也发生较大变化，使得声音信标组具有较大的挑战。

第二章 系统框架

2.1 系统总体设计方案

根据本次比赛要求，我们的小车采用了恩智浦的 i.MX RT1064DVL6A 芯片作为我们的控制核心。CPU 的选择一是要满足系统对运算速度的要求，二是要满足系统硬件设计的需求。由于声音定位需要大量复杂运算，并且使用 H 车模，需要四路正交解码读取编码器的脉冲和八路电机控制，声音定位需要 5 个 ADC 电压采集，避障需要 4 个 UART。RT1064 具有 600Mhz 算力，16 个 PWM 通道，16 个正交解码，32 个 ADC 采集，充分符合我们系统设计要求。

在今年的比赛中需要通过声音寻找响着的信标灯同时还要兼顾未响的障碍灯，因此我们使用了 4 个 MAX9814 和 1 个 RDA5807 模块实现声音定位，采用 4 个 TFmini plus 实现避障。

在声音定位方面，为了麦克纳姆轮噪声干扰，我们在小车上架起一个碳纤维板，并将麦克风转接板放入消音海绵中以减少噪声影响。

小车控制方面，我们采用 PD 控制小车的方向环，并利用陀螺仪反馈小车旋转角速度，使得小车运动更平稳。然后通过 PI 控制四个轮子运动，使小车达到预想的运动轨迹。

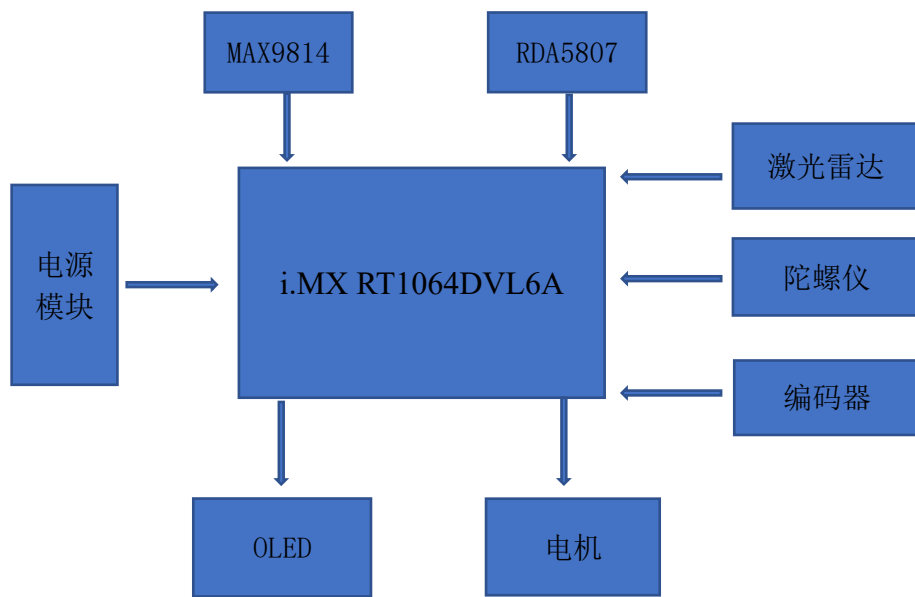
停车方案，我们前期通过分别大量采集在有信标灯响和没有信标灯响时麦克风和 FM 接收的信号，并利用 matlab 仿真对比发现有无信标灯互相关后结果峰值大小有明显差异，因此我们通过设定阈值来筛选小车是否检测到信标灯，从而实现快速准确停车和发车。

驱动设计方面，主要使用四片 mos 构成 H 电桥，利用单片机输出 PWM 输出来控制电机的正反转。同时利用单片机 QTIME 模块配合 1024 线的 AD 相编码器对小车速度实时采集，实现速度反馈，配合上陀螺仪，实现姿态控制。

除此之外，为了方便调试，我们在系统设计过程中，使用 OLED 显示计算结果实现对小车状态进行实时监测。

2.2 系统工作流程

根据上节的方案设计，我们小车上的系统大致可分为以下几大模块：RT1064 主控；声音采集模块；电源管理模块；电机驱动模块；速度检测模块；避障模块；姿态检测模块；辅助调试模块。它们之间相互关系及信息流如下图所示：



2.3 整车布局

为了实现稳定，简洁的要求，整车布局主要依据以下几点：

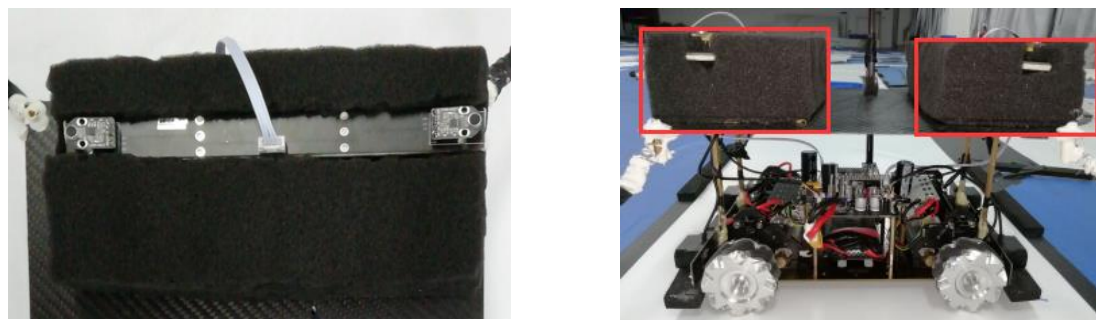
- (1) 车辆底盘在能顺利通过信标灯底盘的前提下，尽量降低重心；
- (2) 麦克风的安装位置尽量远离轮子，减少噪声干扰；
- (3) 电池安装尽量贴近底盘，降低整车重心；
- (4) 齿轮咬合要精密但又无过大阻力；
- (5) 整车在确保性能前提下，应减少重量，确保加减速性能；

第三章 智能车机械结构优化

根据本届大赛的竞赛规则，声音信标组必须使用 H 车模。在车模的搭建过程中，在规则允许的范围内，我们对车模进行了优化改装，具体改装方案如下。

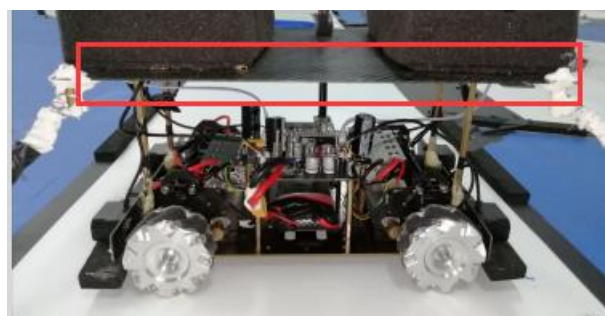
3.1 麦克风模块的安装

我们声音定位方案采用了四个麦克风模块，四个麦克风之间的距离以及安装位置对声音定位极其重要。首先，关于麦克风之间的距离，理论上来说四个麦克风之间距离越大越好，但是由于小车长度和宽度的限制，我们制作了 20cm*2cm 的转接板固定麦克风，然后通过铜柱将转接板举高固定在小车两头。后期调试发现轮子噪声对声音定位有较大干扰，因此我们将转接板嵌入隔音海绵中，并将其整体粘贴在碳纤维板上，一方面可以减少麦克纳姆轮和齿轮的噪声影响，另一方面可以减小麦克纳姆轮运动过程中颠簸带来麦克风的震荡。具体安装效果见下图：



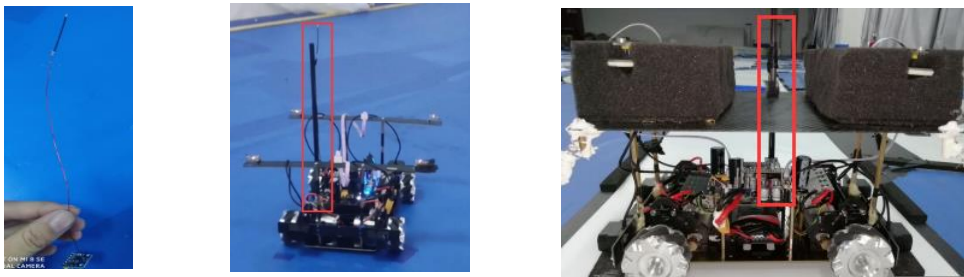
3.2 碳纤维板的安装

通过在没有信标灯响起时，猛地推动小车发现麦克风和 FM 互相关结果峰值能够达到我们设定的阈值，说明轮子噪声对于麦克风定位具有一定的干扰，因此我们通过不断购买不同材料的隔音板架在麦克风下面，最终发现碳纤维板的隔音效果最佳。为了固定碳纤维板，我们先用铜板做了一个三角形框架，然后将碳纤维板用 AB 胶粘在框架上，框架打孔连接在铜柱上，最后铜柱固定在车模底盘上。为了防止螺丝松动，我们将螺丝固定处用热熔胶封住，确保其稳定性。安装效果如图：



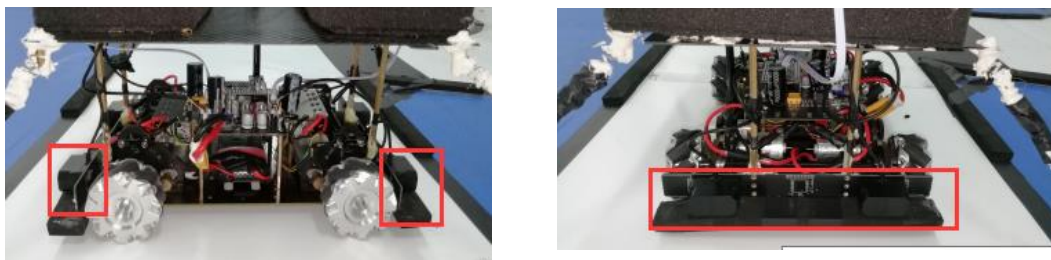
3.3 天线的安装

由于我们的方案是采用 FM 和麦克风做互相关运算，因此无线射频信号的接收尤为重要。我们利用 OLED 显示 FM 信号的 RSSI 值，然后通过对比使用不同天线和天线放置不同高度时 FM 信号的 RSSI 值，发现天线高度对 FM 信号强度影响很大，当然不同天线之间也有一定的差异。最终我们决定用长杜邦线接在模块上，然后在杜邦线上焊一根全向增益 WIFI 铜管天线，一起通过碳素杆举高，碳素杆通过法兰盘固定在底盘并用 AB 胶粘死确保小车运动时天线不晃动。具体连接方式及安装效果如下图：



3.4 激光雷达的安装

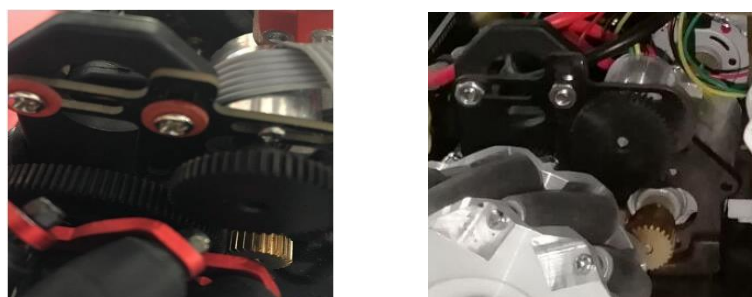
因为我们使用的激光雷达的检测范围与安装的高度有关，并且今年的信标灯更扁平，因此激光雷达的安装角度和高度显得非常重要，高了可能测不到信标灯，低了可能会测到不平的地面。为了便于调整该高度，我们先激光雷达用螺丝固定在铜板上，然后将铜板通过连接键固定在铜柱上，测试好合适的角度和高度后立即用 AB 胶粘好防止小车运动撞击使角度发生变化。为了防止激光雷达被撞坏，我们在前面垫了一些海绵略比其靠前。具体安装效果如下图所示：



3.5 编码器的安装和齿轮啮合

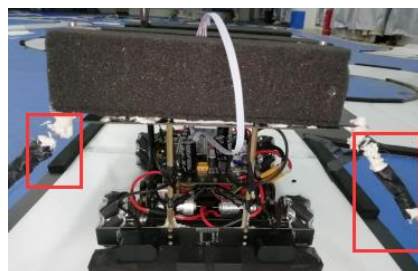
因为无法利用 H 车模自带的编码器安装孔，我们增加了编码器支架安装 1024 线的 AD 相迷你编码器。在此过程中，作为传动机构的三个齿轮之间的啮合对

车模动力有着重大的影响。这里，我们凭借经验和安装效果，通过实验的方法来确定合适的啮合程度。由于固定在车轴上的齿轮无法移动，我们只能通过调节电机和编码器的固定螺丝来调整齿轮和齿轮之间的距离。经过多次的试验，我们总结了一种啮合齿轮的标准，即在齿轮调整完后，将一张 A4 纸送入两个齿轮之间，若 A4 纸恰好能进入且取出后没有残破，则为合适的齿距。我们最终的安装效果如下图：



3.6 磁标的安装

安装磁标的目的只是为了触发信标灯，规则中说明磁标可以伸出小车 10cm，因此我们将磁标通过弹簧伸出去，之所以选择弹簧是因为我们发现当小车运动速度较慢时磁标不容易触发线圈，但是改用弹簧可利用弹簧自身弹性势能晃动磁标触发信标灯以准确快速达到熄灭信标的目的。整个使用弹簧晃动太大，因此我们将弹簧一部分用热熔胶粘住不让他晃动。具体安装如下图：



3.7 小结

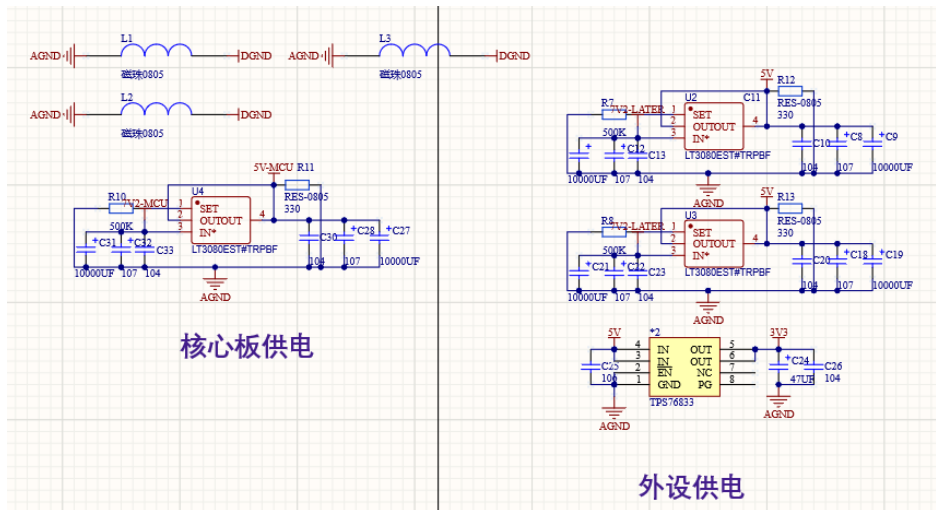
智能车的制作过程中，机械结构的好坏决定了这辆车的速度上限，拥有好的机械结构是对智能车后期提升的一个重要保障。我们在制作的过程中，参考了很多汽车构造的知识，让我们对智能车结构的了解有了更深的层次。在后续的调试中我们不断的改进，逐渐的去适合信标竞赛的要求，达到了一个相对满意的程度。

第四章 电路设计

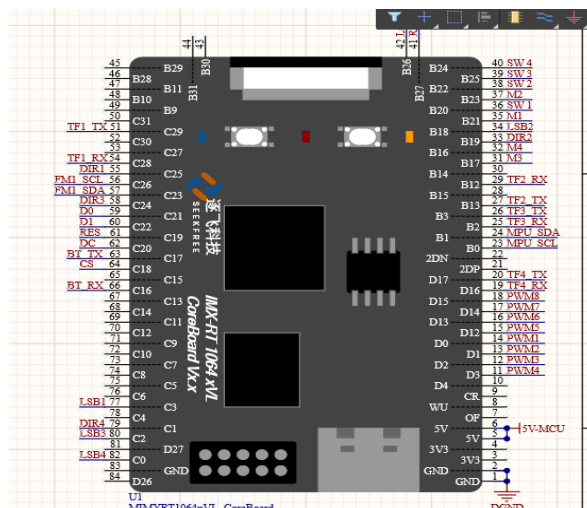
智能车的电路设计的好坏是智能车具有良好稳定性的基础，因此在前期的工作中我们花费了大量时间对电路板进行设计优化，从布局到走线都是精益求精，同时完全满足了智能车运行和调试的需要。

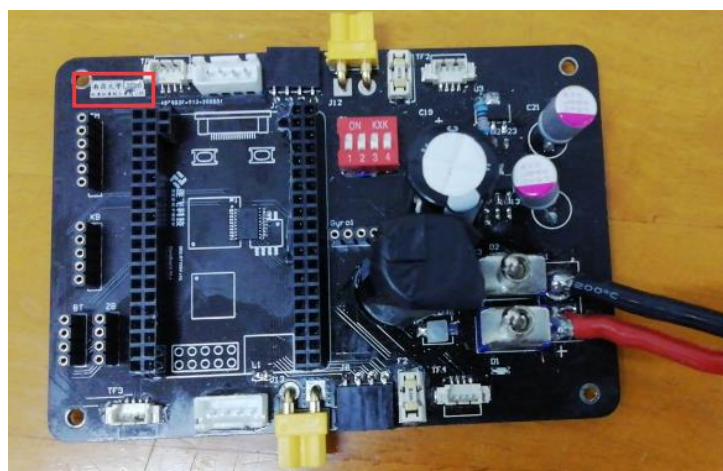
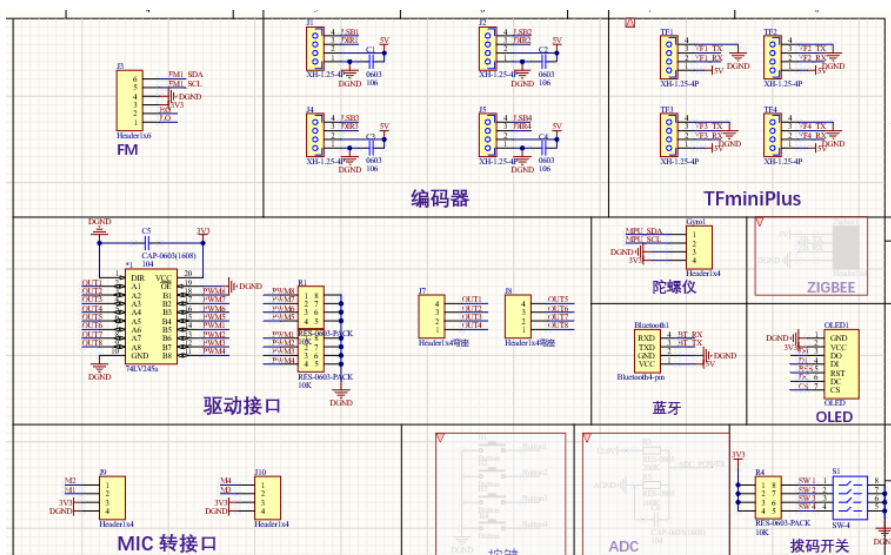
4.1 主板设计

该主板主要包括供电模块、核心板模块以及各项外设。在整体的设计中，我们首先确保电源模块的稳定，以此确保各个模块的正常工作。所以供电模块主要用到输出电流较大的 LT3080 稳压，需要配合散热片使用，防止芯片由于过热而产生不可逆的损耗，将核心板及外设分开供电解决 RT1064 的上电时序问题；



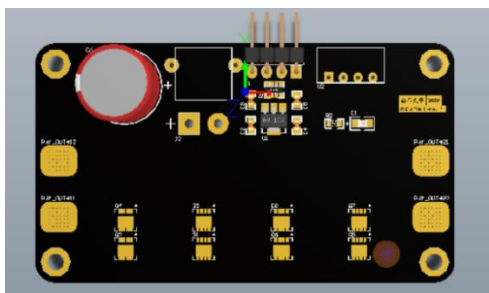
核心板用到的是逐飞科技旗下的 RT1064，该核心板的算力足够智能车快速高效地完成所有运算：外设主要包括陀螺仪、麦克风、编码器以及激光雷达等必要的模块，设计时应预留出足够接口。主板在设计时，既要满足到软件的需求，又必须兼顾空间资源的合理利用.原理图和实物图如下

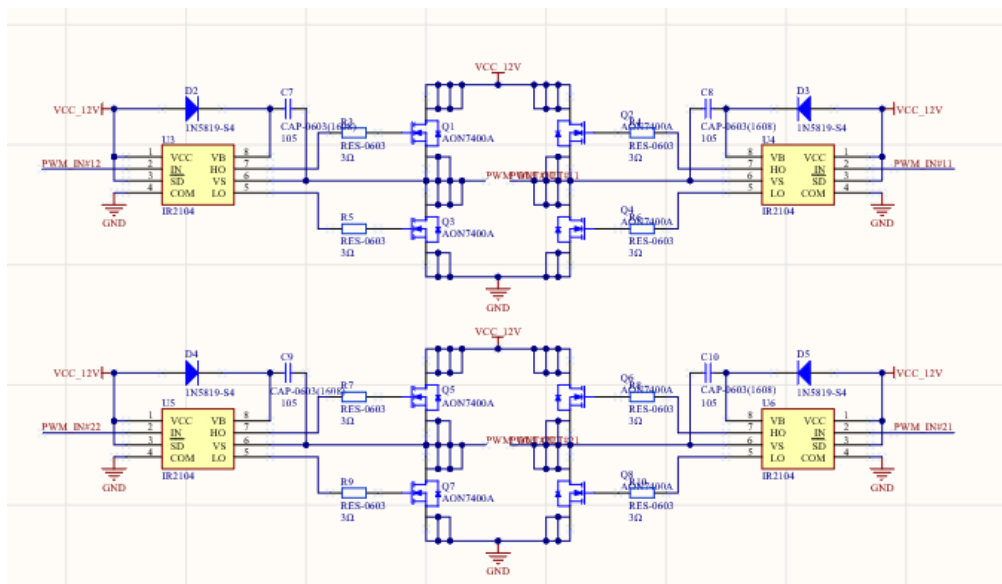




4.2 驱动板设计

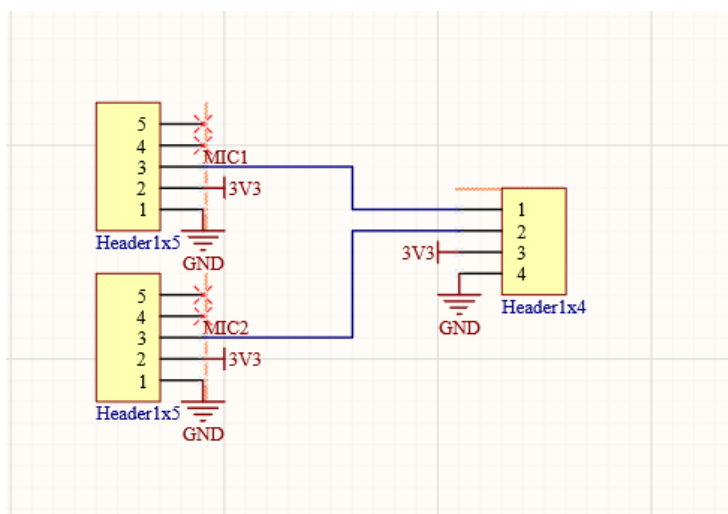
从主控板传过来的 PWM 信号首先通过 74HC245A 缓冲芯片，至 IR2104S 芯片。缓冲芯片的作用在于把主控引脚与驱动电路隔离开来，防止驱动板上的电压电流造成 PWM 口电流倒灌进而烧毁引脚。驱动信号通过 IR2104S 驱动芯片实现了电平的转换，由原来的 3.3V 电平转换成了 12V 的电平，保证了 AON7400 芯片能快速稳定地被驱动。为了保证驱动能通过较大电流，对主要的电机驱动回路的导线进行了加粗、开窗以及加锡处理。原理图和实物图如下所示





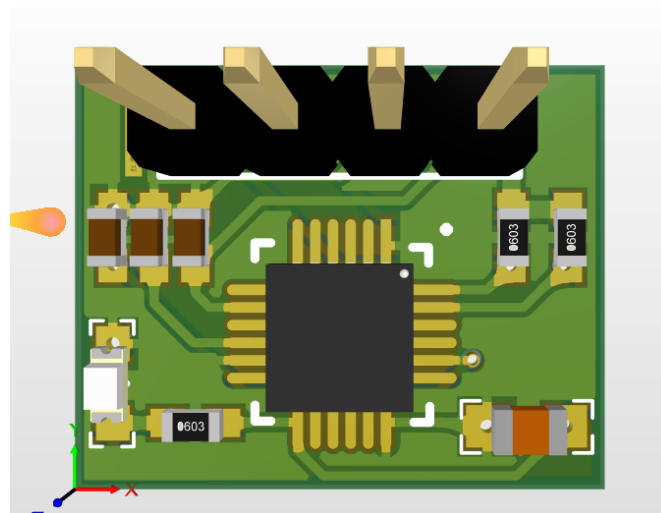
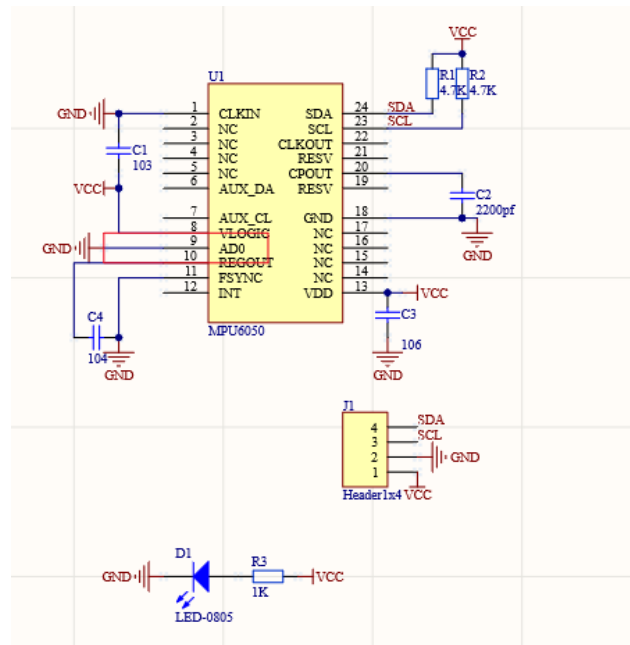
4.3 转接板设计

为了使接收到的声音信号更精准，我们将四个麦克风分别分立在 H 车模的四个角，制作转接板的目的是固定麦克风的位置以达到精准接收，下图为基于龙邱麦克风模块的转接板，规格为 200mm×20mm。原理图和实物图如下：



4.4 陀螺仪设计

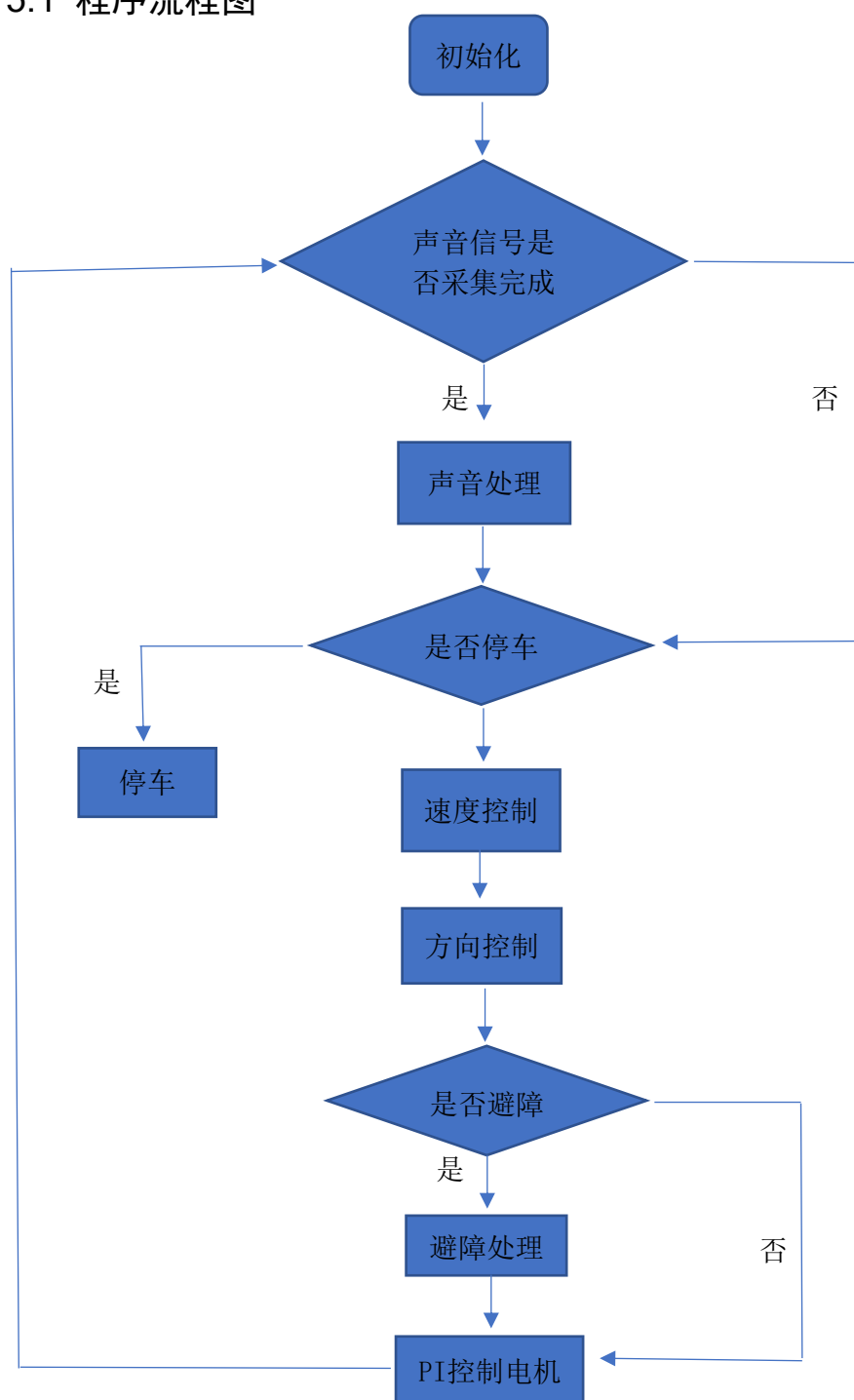
为了检测车模倾角，我们用了基于MPU6050的陀螺仪，该芯片内部整合了3轴陀螺仪和3轴加速度传感器，并可利用自带的数字运动处理器（DMP）硬件加速引擎，通过IIC接口，向应用端输出姿态解算后的数据。为了使测量尽量准确，我们选择将陀螺仪放在车体中间。原理图如下：



第五章 软件设计

高效稳定的控制程序和高适应性的算法是智能车快速平稳运行的基础。我们设计的智能车采用恩智浦公司生产的 i.MX RT1064DVL6A 芯片作为控制器，通过 MAX9814 模块和 RDA5807 模块采集信息，为使小车快速平稳运行，我们使用了编码器测速对小车进行速度闭环。速度和方向控制算法方面，我们采用工业控制广泛采用的经典 PID 算法，配合理论计算与实际参数调试过程使程序和小车充分契合，使智能车达到快速平稳运行的目的。

5.1 程序流程图



5.2 声音信号采集及处理

5.2.1 信号采集

今年的声音信标主要发出两种导航信号，第一种是 Chrip 声音信号，规格为起始频率 250HZ，终止频率 2000HZ 的线性调频信号，信号时间长度为 0.2048 s，DA 转换频率为 10KHZ，DA 输出为 12bit；第二种是射频信号，华东赛区比赛采用的是 95MHZ 频率。由于我们采用的方案是射频信号和麦克风声音信号做互相关运算，因此我们用 ADC 以 10KHZ 采样频率采集 MAX9814 的输出和 RDA5807 的 RO 或 LO 输出。首先，初始化 5 个 ADC 通道，然后初始化一个 PIT 通道，开启一个 100us 的 PIT 定时器，将声音采集放入 PIT 定时器中断，并设定采集完成标志，当采集完 2048 个数据时进行声音处理。之所以采集 2048 个数据，是因为信号长度为 0.2048s，恰好为 100us*2048，如果提高采样频率，采集信号个数也应随之变化，以确保采完一帧数据。

5.2.1 信号处理

我们的信号处理方案是采用四个麦克风信号分别和 FM 信号做互相关运算，得出四个麦克风到信标灯的距离，然后选择前面或者后面两个麦克风得出偏差对小车进行方向控制，通过解三角形得出两个麦克风中心距灯的距离对小车进行速度控制以及切灯控制。

声音测距基本原理是通过声音在空气中传播的速度和时间差来测量声源与接收器之间的距离 D ： $D = \Delta t * V_{\text{sound}}$ 。其中空气中的声音传播速度一般为： $c_{\omega} = c_0 + \sqrt{(1 + \alpha * \omega)}$ ，其中 ω 是空气温度， α 是空气膨胀系数，取 $1/273.15 = 3.661 * 10^{-3} (1/^{\circ}\text{C})$ ， c_0 是空气在零摄氏度是的传播速度，约为 331m/s。而需要得到距离还需要知道发送和接收信号之间的延时。一种比较简便的方法就是通过相关法来确定。由于射频信号传输速度接近光速，所以相当于是发送信号，所以只需要将射频信号与麦克风接收的声音信号做互相关运算即可。

对两个实数离散时间信号 $x[n]$, $y[n]$, $n \in \{0, \dots, N-1\}$, 它们之间的普通互相关运算的定义为 $R_{xy}[n] = \sum_{(m, m-n \in [0, N-1])} x[m] \cdot y[m-n]$, 从定义上看计算两个长度为 N 的相关运算，乘法、加法的计算复杂度与 N^2 成正比。我们方案需要计算 4 组 2048 和 2048 长度的互相关计算，计算量非常之大。在信号运算中，还有一个应用更广泛的运算：卷积运算。 $x(t)$, $y(t)$ 之间的卷积运算定义为： $x(t)*y(t)=\int_{-\infty}^{\infty} x(\tau) \cdot y(t-\tau)d\tau$ 。对比一下可发现相关运算和卷积运算的定义，

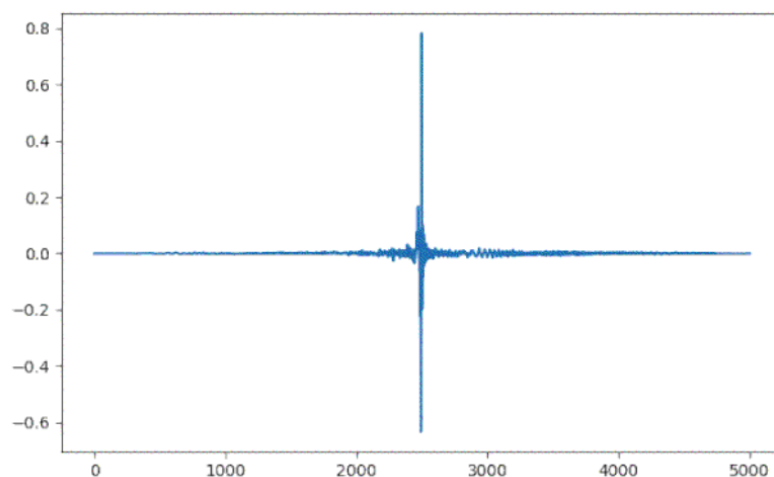
可以看出它们之间的关系： $R_{xy}(t) = x(t) * y^*(-t)$ 。之所以讨论相关运算与卷积运算之间的关系是为了寻找相关运算的快速算法。计算一个序列 $x[n], n \in \{0, \dots, N-1\}$ 的离散傅里叶变化 $DFT\{x[n]\}$ 有相应的快速算法——快速傅里叶变换 $FFT\{x[n]\}$ ，在 N 为 2 的整数次幂的情况下，计算 FFT 的乘法和加法的复杂度在 $\log_2 N$ 的数量级别。正变换和反变换的复杂度相同。根据傅里叶变换的卷积定理，序列的时域卷积运算，在频域是乘积运算。基于此，再利用前面讨论的相关与卷积运算之间的关系，可以得到连个序列的相关运算的快速算法： $R_{x,y}(t) = FFT^{-1}\{FFT[x(t)] \cdot FFT[y(t)]^*\}$ 。在声音定位中，利用上面公式计算的时候，还需要将两个信号通过补零，变成两个序列长度之和。

在用 C 语言做信号处理之前，我们通过大量采集麦克风信号和 FM 射频信号并存入 CSV 文件中，然后使用 matlab 读取，然后仿真以确定我们的方案有效，经过不断寻找资料和尝试最终形成我们最后的方案。如下是我们仿真程序和结果：

```

1 - mic_sample = csvread('D:\signal_data\sound1.csv');
2 - fm_sample = csvread('D:\signal_data\fm.csv');
3
4 - mic_process = mic_sample(1:2048);
5 - fm_process = fm_sample(1:2048);
6
7 - mic_m = mean(mic_process); %求均值
8 - fm_m = mean(fm_process);
9
10 - mic_process(1,:) = mic_process(1,:) - mic_m; %去均值
11 - fm_process(1,:) = fm_process(1,:) - fm_m;
12
13 - mic_process(2049:4096) = 0; %补零
14 - fm_process(2049:4096) = 0;
15
16 - cor_result = ifft(fft(mic_process).*conj(fft(fm_process)));
17
18 - [max_pos]=find(cor_result == max(cor_result(:)));
19

```



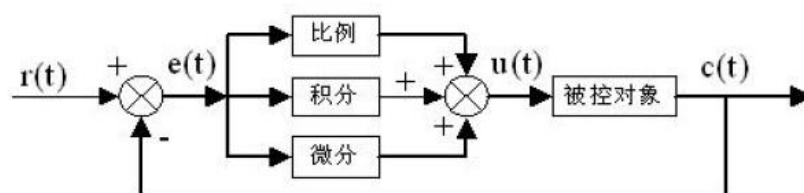
matlab 仿真达到预期效果后，之后就只需将 matlab 语言的代码转化成 C 语言写在 IAR 中，并通过蓝牙发送上位机观察采集信号和处理结果的波形，验证 C 代码的正确性，验证无误后，将声音处理封装成函数，后期调车只需调用即可，并将处理结果直接显示在 OLED 屏幕上，可以大大减少后期工作量。具体程序见附录里面源码。

5.3 小车运动控制

5.3.1 PID算法介绍

在过程控制中，PID 控制器是一直应用最为广泛的一种控制器，也是众多控制算法中应用最为普遍的控制算法当被控对象的结构和参数不能完全掌握，或得不到精确的数学模型时，控制理论的其它技术难以采用时，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用 PID 控制技术最为方便。即当我们不完全了解一个系统和被控对象，或不能通过有效的测量手段来获得系统参数时，最适合用 PID 控制技术。PID 控制，实际中也有 PI 和 PD 控制。PID 控制器是一种线性控制器，它根据给定值与实际输出值构成控制偏差。将偏差的比例(P)、积分(I)和微分(D)通过线性组合构成控制量，对被控对象进行控制，故称 PID 控制器。

由于单片机控制是采样控制，只能根据采样时刻的偏差控制输出量，不能计算连续输出量，所以要采用离散化的 PID 算法。离散 PID 分为位置式和增量式两种。位置式的输出量直接就是控制量，由于全量输出，所以每次输出均与过去的状态有关，计算时要对过去偏差进行累加，计算机工作量大；而且因为计算机输出量对应的是执行机构的实际输出，如计算机出现故障，输出大幅度变化会引起执行机构位置的大幅度变化，这种情况往往是生产实践中不允许的，在某些场合，还可能造成严重的生产事故。因而产生了增量式 PID 控制的控制算法，所谓增量式 PID 是指数字控制器的输出只是控制量的增量。由于计算机输出增量，所以误动作时影响小算式中不需要累加。控制增量的确定仅与最近 k 次的采样值有关，所以较容易通过加权处理而获得比较好的控制效果。原理框图如图所示：



5.3.1 方向控制

小车的方向控制我们采用的是 PD 控制器。由麦克纳姆轮的运动方程我们可以知道，H 车模运动速度可看成三个方向的运动的叠加，横移速度，竖直速度，Z 轴旋转速度。我们信号处理得到方向偏差的方案是通过选取的两个麦克风和 FM 相关结果峰值下标得到，经过尝试发现用偏差控制 Z 轴旋转速度比控制横移小车能更快抵达目标，运动轨迹也更加平稳。同时为了小车运动姿态更加平滑，我们采用陀螺仪测出来的角速度来取代 D 项的偏差之差项，并取得良好的效果。

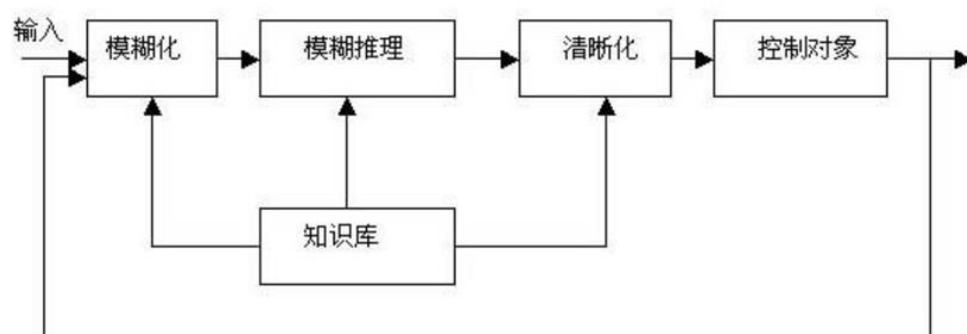
5.3.2 速度控制

小车前进的速度控制我们采用的模糊控制法。使得小车在距信标不同距离获得不同的速度，且比直接根据距离给定分段速度小车运动更加平滑。

5.3.2.1 模糊控制简介

利用模糊数学的基本思想和理论的控制方法。在传统的控制领域里，控制系统动态模式的精确与否是影响控制优劣的最主要关键，系统动态的信息越详细，则越能达到精确控制的目的。然而，对于复杂的系统，由于变量太多，往往难以正确的描述系统的动态，于是工程师便利用各种方法来简化系统动态，以达成控制的目的，但却不尽理想。换言之，传统的控制理论对于明确系统有强而有力的控制能力，但对于过于复杂或难以精确描述的系统，则显得无能为力了。因此便尝试着以模糊数学来处理这些控制问题。

一个实际的模糊控制系统实现时需要解决三个问题：知识表达、推理策略和知识获取。知识表示是指如何将语言规则用数值方式表示出来；推理策略是指如何根据当前输入条件产生一个合理的结果；知识获取解决如何获得一组恰当的规则。由此可见，模糊控制是以模糊集合论、模糊语言及模糊逻辑推理为基础的一种计算机控制。而模糊控制系统是由模糊控制器和控制对象组成的，其结构如下：



模糊控制的基本结构，它主要包括以下四部分。

1. 模糊化

将输入值以适当的比例转换到论域的数值，利用口语化变量来描述测量物理量的过程，根据适合的语言值(linguistic value)求该值相对的隶属度，此口语化变量称为模糊子集合(fuzzy subsets)。

2. 知识库

包括数据库(data base)与规则库(rule base)两部分，其中数据库提供处理模糊数据的相关定义;而规则库则藉由一群语言控制规则描述控制目标和策略。

3. 模糊推理

模仿人类下判断时的模糊概念，运用模糊逻辑和模糊推论法进行推论，得到模糊控制讯号。该部分是模糊控制器的精髓所在。

4. 清晰化

(1) 将模糊的控制量经清晰化变换，变成表示在论域范围内的清晰量。

(2) 将表示在论域范围内的清晰量经尺度变换，变成实际的控制量。

5.3.2.2 模糊控制在速度控制的应用

在小车前进速度控制中，我们将声音处理计算出来的小车与信标灯之间的距离作为模糊控制的输入量，建立模糊规则表如下：

```
uint16 rule[7][7] =
{
    /*Track complexity*/ // 0, 1, 2, 3, 4, 5, 6, Prospect See
    /* 0 */ {6, 5, 4, 3, 2, 1, 0}, //U语言数组里面的下标
    /* 1 */ {5, 5, 4, 3, 2, 1, 0},
    /* 2 */ {4, 4, 3, 2, 1, 1, 0},
    /* 3 */ {3, 3, 3, 2, 1, 1, 0},
    /* 4 */ {2, 2, 2, 1, 1, 0, 0},
    /* 5 */ {2, 1, 1, 0, 0, 0, 0},
    /* 6 */ {1, 1, 0, 0, 0, 0, 0},
}; //横D竖P
```

规则表的建立是通过多次的试验以及经验结果得到的，从某种程度上来说不具有严格的规则性，但相对于我们的实际情况来说，相对合适。

5.4 避障策略

由于今年的信标灯形状为扁平的圆台型，在小车行驶过程中若不能灵活避开

障碍，将对小车和信标灯带来不可估量的后果。况且声音信标灯的射频信号极其不稳定，小车若正对信标灯猛烈撞击将对射频信号发出造成巨大的毁坏，而且难以发现，是对本场地后面参赛队伍的极其不负责任的行为。因此强烈建议参赛队员做好避障措施，做一个负责任的信标人。强烈建议明年信标组规则撞灯罚时，并且修灯记录比赛时间，严重毁坏信标灯者应适当赔偿一定金额。

5.4.1 障碍为目标灯

由于我们声音处理结果已经得出了方向偏差和距离，只需距离小于一定阈值，我们就可以认为障碍为目标灯，此时小车开始做切灯动作。要想做切灯动作，只需通过实验得出小车在我们预想切线路径上的方向偏差和距离之间的关系即可对偏差做出相应的坐标偏移，使得小车按照我们预想路径运行。以下是我们实验得出需要偏移的坐标与距离之间的关系：

```
313     if(beacon_distance < TANGENT_TH) //切灯处理
314     {
315         if(motor_dir == 1)
316             beacon_error = (maxacor_index[1]+maxacor_index[2])-(maxacor_index[0]+maxacor_index[3]);
317         else if(motor_dir == -1)
318             beacon_error = (maxacor_index[0]+maxacor_index[3])-(maxacor_index[1]+maxacor_index[2]);
319
320         if(beacon_error >= 0) //灯在左边
321         {
322             if(beacon_distance >= 92)
323                 beacon_error -= 2;
324             else if(beacon_distance >= 86)
325                 beacon_error -= 3;
326             else if(beacon_distance >= 74)
327                 beacon_error -= 4;
328             else if(beacon_distance >= 59)
329                 beacon_error -= 5;
330             else if(beacon_distance >= 23)
331                 beacon_error -= 6;
332             else if(beacon_distance >= 6)
333                 beacon_error -= 7;
334             else
335                 beacon_error -= 8;
336         }
```

5.4.2 障碍为非目标灯

倘若目标灯的距离不在切灯距离内，并且此时运动方向前面两个激光雷达测量的最小距离小于安全避障距离，开始进行避障，安全避障距离我们选择的是 1m，所以只需在小车运动基础上加一点横移速度即可避开障碍。为了防止小车在运动过程中突然遇到信标灯，并且非常近有撞上信标的危险，我们还设置了一个紧急避障距离，如果小车进入紧急避障距离，此时小车其他速度都置零，只加一个很大的横移速度避开障碍，避免造成不可估量的后果。至于避障方向，则通过偏差方向来确定小车往哪边行驶，即灯在左边，则避障走左边；反之同理。

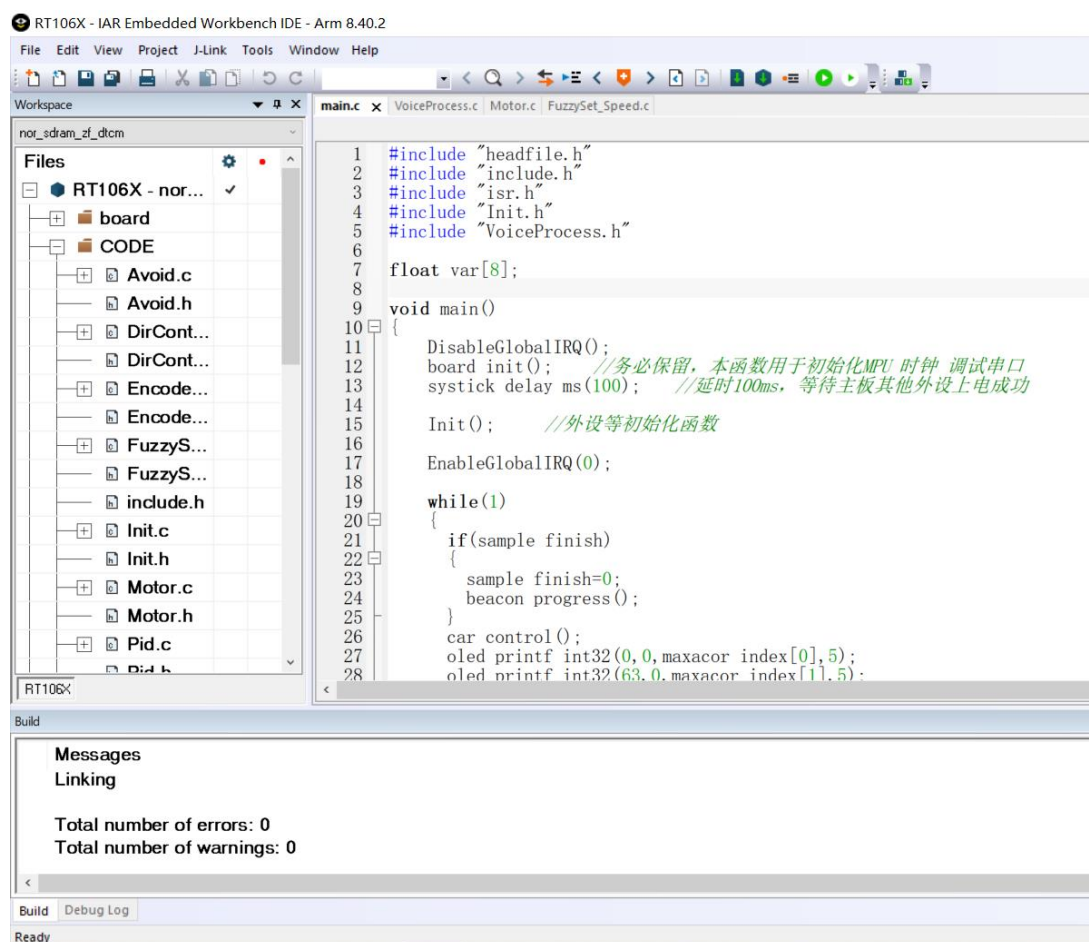
第六章 智能车调试

6.1 嵌入式平台

我们的软件程序编写任务主要在 IAR Embedded Workbench IDE 平台下进行的,该平台是一个非常有效的集成开发环境 (IDE),它使用户充分有效地开发并管理嵌入式应用工程。作为一个开发平台,他具备较为完善的特性。

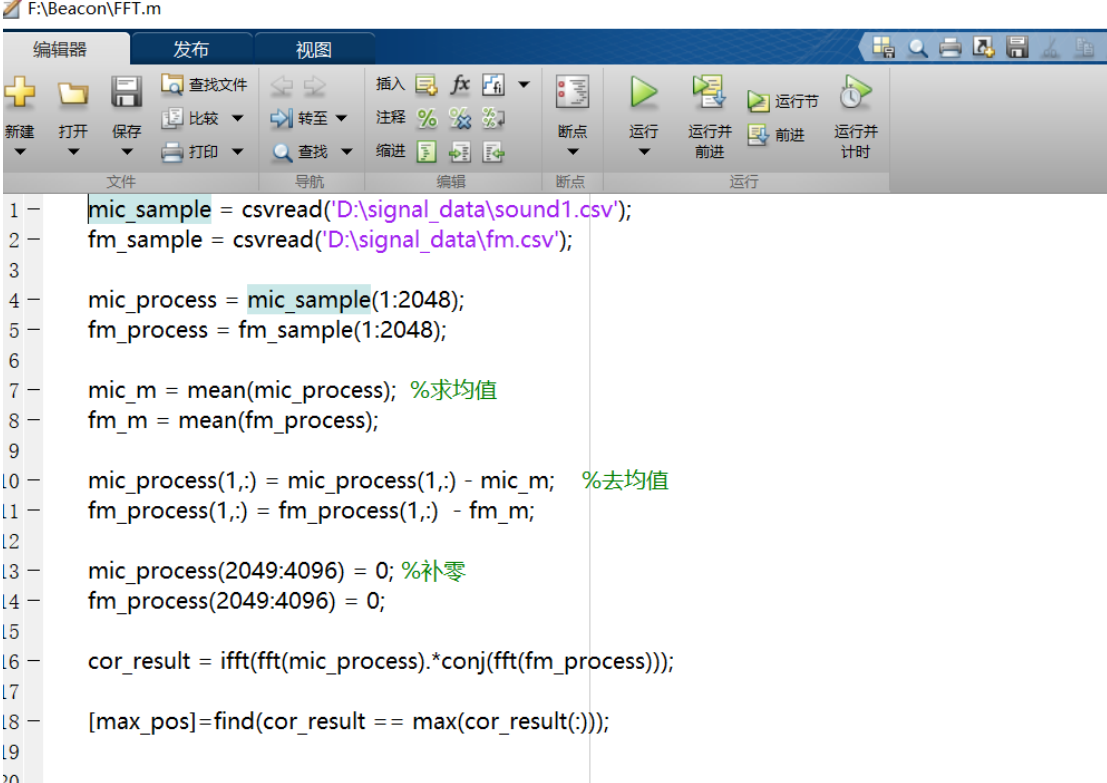
嵌入式 IAR Embedded Workbench 使用于大量 8 位、16 位以及 32 位的微处理器和微控制器,使用户在开发新的项目时也能在所熟悉的开发环境中进行。它为用户提供一个易学和具有最大量代码继承能力的开发环境,以及对大多数和特殊目标的支持。嵌入式 IAR Embedded Workbench 有效提高用户的工作效率,通过 IAR 工具,用户可以大大节省工作时间。

由于我们使用的芯片 RT1064 较为新,所以使用的是较高版本的 IAR。因为 IAR 只能向下兼容,新版本建立的工程无法在低版本软件中打开,因此我们组商量好选择统一使用 8.40.2 版的 IAR。程序开发和调试界面如下图:

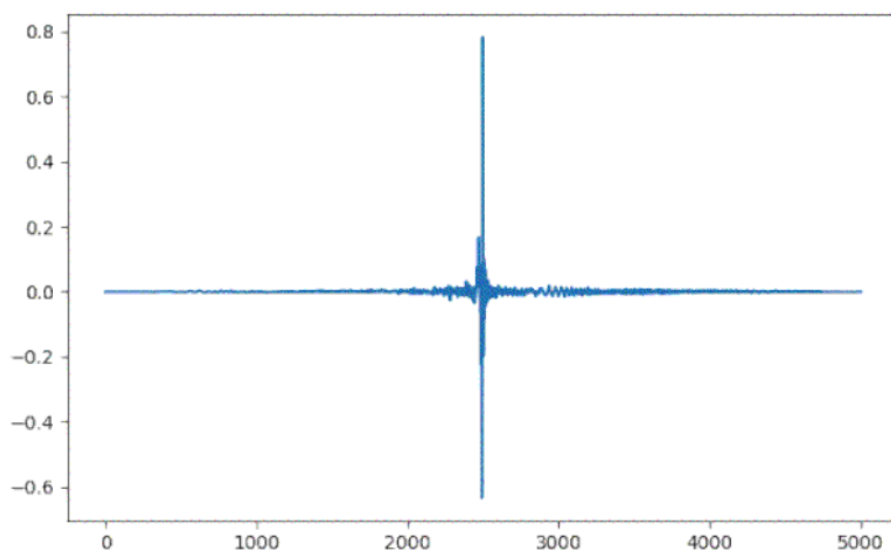


6.2 仿真软件

我们在做声音信号处理时主要采用的是 matlab 观察波形和处理结果，以验证方案的正确性。Matlab 是美国 The MathWorks 公司出品的一款强大的数学软件，里面有我们需要的 FFT 数学函数，大大减少我们自己研究算法的时间。并且具有丰富的矩阵运算、绘图函数、图像处理功能，使得信号处理更加直观清晰。一下是我们的仿真界面：

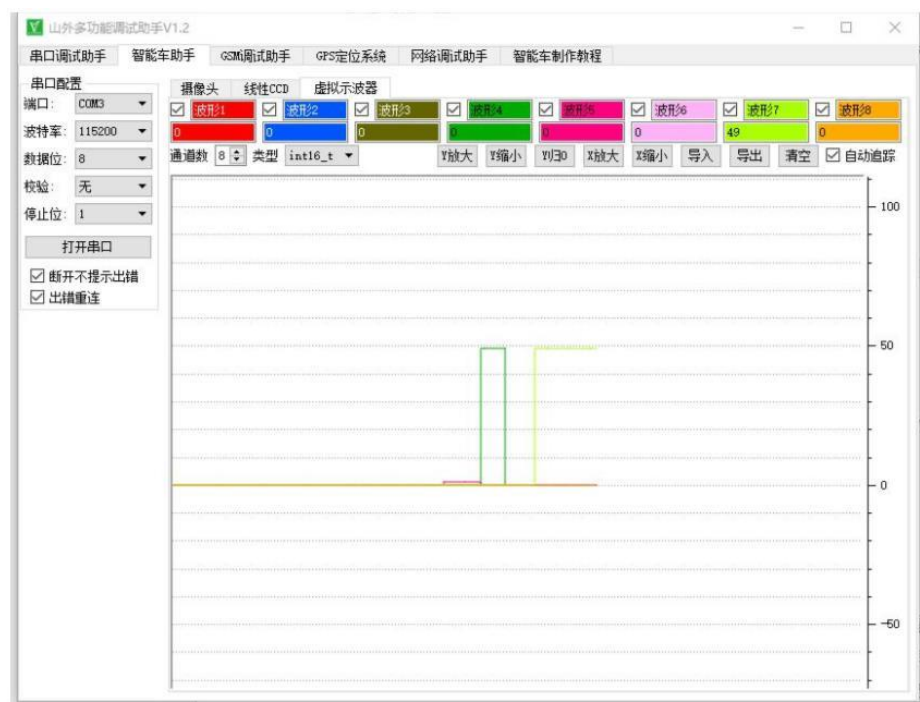


```
F:\Beacon\FFT.m
编辑器 发布 视图
+ 新建 打开 保存 比较 打印 查找文件 转至 查找 插入 注释 缩进 断点 运行 运行并前进 运行并计时
1 - mic_sample = csvread('D:\signal_data\sound1.csv');
2 - fm_sample = csvread('D:\signal_data\fm.csv');
3
4 - mic_process = mic_sample(1:2048);
5 - fm_process = fm_sample(1:2048);
6
7 - mic_m = mean(mic_process); %求均值
8 - fm_m = mean(fm_process);
9
10 - mic_process(1,:) = mic_process(1,:) - mic_m; %去均值
11 - fm_process(1,:) = fm_process(1,:) - fm_m;
12
13 - mic_process(2049:4096) = 0; %补零
14 - fm_process(2049:4096) = 0;
15
16 - cor_result = ifft(fft(mic_process).*conj(fft(fm_process)));
17
18 - [max_pos]=find(cor_result == max(cor_result(:)));
19
20
```



6.3 上位机调试工具

通过蓝牙串口将数据传送到上位机，可以在车模运行过程中对车模的参数进行实时观测，根据传回的数据合理的整定参数，有利于对汽车整体性能的提高，以下即为山外的上位机，能传输图像和波形：



6.4 小结

通过多种调试方式的组合，大大减少了我们的工作量，尤其是山外上位机的应用，很好的检验了我们程序的可用性，避免了很多的弯路。而其他几种对车模数据的显示，帮助我们更好的整定参数。所谓磨刀不误砍柴工，我们花了一定的时间去开发有用的方式，在我们的调试后期，确实为我们提供了很大的便利。

第七章 总结

本文从机械、硬件、软件等方面详细介绍了车模的设计和制作方案。从机械和硬件方面，我们力求电路的稳定可靠，尽量将车模做的轻快灵活，尽可能降低重心；在软件方面，我们努力追求最有效的行驶路径，不断优化各方面策略，软件主动防翻、有效的避障策略等是我们的创新点，尽可能提高车模的稳定性。

在车模制作过程中，我们学到很多知识，逐渐对单片机控制、电路设计等有了比较好的认识，很感谢智能车基地能够给我们这样的平台，感谢各位指导老师和学长学姐们的指导。智能车比赛永远不是孤军奋战，更体现的是整个团队的力量，也正是在备战比赛中，我们基地成员表现得比以往更加团结，也让每一位基地成员感受到智能车基地大家庭的温暖。经过近一年的辛苦准备，我们也积累了很多经验，深刻认识到坚持不懈的重要性，从开始招新的近百人到最后的十几人，考验的不是谁更聪明，而是谁可以坚持努力，不断地发现问题解决问题，坚持为比赛付出时间和精力。

最后，再次感谢南昌大学智能车基地这个大家庭以及一直帮助我们信标组的学长学姐们，感谢一直支持和关注智能车比赛的学校和学院领导以及各位老师，也感谢组委会能坚持组织这样一项很有意义的比赛。也祝愿红黑红黑的小黄鸡二队的队员们学业有成，未来可期！终有一天你们的小车将不在赛道上驰骋，而未来在前方只有序曲没有终章，信标人永远前进，不曾彷徨，向着星辰大海，发车！

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] 蔡述庭. “飞思卡尔”杯智能汽车竞赛设计与实践. 北京: 北京航空航天大学出版社. 2012.
- [3] 王淑娟, 蔡惟铮, 齐明. 模拟电子技术基础. 北京: 高等教育出版社. 2009
- [4] John G. Proakis. 数字信号处理. 北京: 电子工业出版社. 2019.
- [5] 郑君里, 应启珩, 杨为理. 信号与系统. 北京: 高等教育出版社. 2011.
- [6] 邹霞. 麦克风阵列信号处理. 北京: 国防工业出版社. 2016.
- [7] 余志生主编. 汽车理论. 北京: 机械工业出版社. 2009.
- [8] 王望予主编. 汽车设计. 北京: 机械工业出版社. 2004.
- [9] 白志刚. 自动调节系统解析与 PID 整定. 化学工业出版社, 2012.

附录 A 信号处理核心代码

```
#include "arm_const_structs.h"
#include "arm_math.h"
#include "arm_common_tables.h"

#include "VoiceProcess.h"
#include "SEEKFREE_RDA5807.h"
#include "zf_adc.h"
#include "zf_pit.h"
#include "math.h"
#include "zf_systick.h"
#include "Motor.h"

#define PEAK_TH      10000           //峰值阈值
#define MAXACOR_TH  230             //峰值最大下标，场地最大8m，最大下标
230

#define C0 331.6                  //空气在0摄氏度速度
#define DILATATION 0.6           //系数
#define TEMPERATURE 28           //空气温度

#define SOUND_VELOCITY 346.6      //声速为331.6+0.6*T    T取25° C

int sample_finish =0;             //数据采集完成标志
uint8 mic_flag=0;                 //adc采集完成标志位

/*
 * ADC数据采集麦克风信号
 *
 *      MIC1      MIC2
 *
 *      FM
 *
 *      MIC4      MIC3
 */
float mic1_data[MIC_NUM]={0};     //保存麦克风1数据的缓存
float mic2_data[MIC_NUM]={0};     //保存麦克风2数据的缓存
float mic3_data[MIC_NUM]={0};     //保存麦克风3数据的缓存
float mic4_data[MIC_NUM]={0};     //保存麦克风4数据的缓存
float fm_data[MIC_NUM]={0};       //保存FM数据的缓存

int16 maxacor_index[4]={0}; //相关峰值下标
```

```

float beacon_distance=0;//距离
float beacon_angle = 0;//角度
float beacon_error = 0;//偏差
//uint8 quadrant=0;

void voice_init()                //ADC采集初始化，FM初始化
{
    //麦克风采集
    adc_init(MIC1_ADC, MIC1_CH, ADC_12BIT);
    adc_init(MIC2_ADC, MIC2_CH, ADC_12BIT);
    adc_init(MIC3_ADC, MIC3_CH, ADC_12BIT);
    adc_init(MIC4_ADC, MIC4_CH, ADC_12BIT);
    adc_init(FM_ADC, FM_CH, ADC_12BIT);

    //FM初始化
    rda5807_init(FM_FREQUENCY);

    // pit_interrupt_us(PIT_CH1, SAMPLE_FRE); //麦克风信号采集
}

void voice_sample()              //声音信号采集
{
    static int16 mic_sample_num = 0;

    if(mic_sample_num < MIC_NUM)
    {
        //麦克风数据采集
        //在FFT运算中需要减去直流偏量
        //模块直流偏量1.25V     $1.25/3.3 * 2^{12}=1551$      $2^{12}=4096$ 
        mic1_data[mic_sample_num]=0.05*(adc_convert(MIC1_ADC, MIC1_CH)-1551);
        mic2_data[mic_sample_num]=0.05*(adc_convert(MIC2_ADC, MIC2_CH)-1551);
        mic3_data[mic_sample_num]=0.05*(adc_convert(MIC3_ADC, MIC3_CH)-1551);
        mic4_data[mic_sample_num]=0.05*(adc_convert(MIC4_ADC, MIC4_CH)-1551);

        fm_data[mic_sample_num]=0.05*(adc_convert(FM_ADC, FM_CH)-1100);

        mic_sample_num++; //采集点数加一
    }
    else
    {
        mic_sample_num = 0;
        sample_finish=1;
    }
}

```

```

    }
}

void normal(float *x, uint16 len)      //去掉均值
{
    float sum = 0;
    uint16 i = 0;

    for(i = 0; i < len; i++)
    {
        sum += x[i];
    }

    sum = sum / len;

    for(i = 0; i < len; i++)
    {
        x[i] = (x[i] - sum);
    }
}

}

/*
 *   acor1:      MIC1和FM 互相关结果
 *   acor2:      MIC2和FM 互相关结果
 *   acor3:      MIC3和FM 互相关结果
 *   acor4:      MIC4和FM 互相关结果
 *   len   :   数据长度
 *   index:   存放相关结果最大值下标
 */
void seek_max(float *acor1, float *acor2, float *acor3, float *acor4, int16
len, int16 *index)      //寻找幅值最大下标
{
    int16 i = 0;

    index[0]=0;
    index[1]=0;
    index[2]=0;
    index[3]=0;

    for(i=1;i<len;i++)
    {
        if(acor1[i] > acor1[index[0]])

```

```

    {
        index[0] = i;
    }
    if(acor2[i] > acor2[index[1]])
    {
        index[1] = i;
    }
    if(acor3[i] > acor3[index[2]])
    {
        index[2] = i;
    }
    if(acor4[i] > acor4[index[3]])
    {
        index[3] = i;
    }
}
}

```

//声音处理获取峰值下标

```

void voice_progress()
{
    arm_rfft_fast_instance_f32 s;//傅里叶变换结构体参数

    float mic1_fft_in[MIC_NUM*2]={0};    //补零结果
    float mic2_fft_in[MIC_NUM*2]={0};
    float mic3_fft_in[MIC_NUM*2]={0};
    float mic4_fft_in[MIC_NUM*2]={0};
    float fm_fft_in[MIC_NUM*2]={0};

    float mic1_fft_out[MIC_NUM*2]={0};    //fft结果
    float mic2_fft_out[MIC_NUM*2]={0};
    float mic3_fft_out[MIC_NUM*2]={0};
    float mic4_fft_out[MIC_NUM*2]={0};
    float fm_fft_out[MIC_NUM*2]={0};

    float acor1[MIC_NUM*2]={0};            //卷积结果
    float acor2[MIC_NUM*2]={0};            //信号1乘以信号2共轭
    float acor3[MIC_NUM*2]={0};
    float acor4[MIC_NUM*2]={0};

    float acor1_ifft[MIC_NUM*2]={0};      //ifft结果
}

```

```

float acor2_ifft[MIC_NUM*2]={0};
float acor3_ifft[MIC_NUM*2]={0};
float acor4_ifft[MIC_NUM*2]={0};
//去均值
normal(mic1_data,MIC_NUM);
normal(mic2_data,MIC_NUM);
normal(mic3_data,MIC_NUM);
normal(mic4_data,MIC_NUM);
normal(fm_data,MIC_NUM);

//补零
for(int16 i=0;i<MIC_NUM;i++)
{
    mic1_fft_in[i] = mic1_data[i];
    mic1_fft_in[i+MIC_NUM] = 0;

    mic2_fft_in[i] = mic2_data[i];
    mic2_fft_in[i+MIC_NUM] = 0;

    mic3_fft_in[i] = mic3_data[i];
    mic3_fft_in[i+MIC_NUM] = 0;

    mic4_fft_in[i] = mic4_data[i];
    mic4_fft_in[i+MIC_NUM] = 0;

    fm_fft_in[i] = fm_data[i];
    fm_fft_in[i+MIC_NUM] = 0;
}

//信号傅里叶变换
arm_rfft_fast_init_f32(&s,MIC_NUM*2);
arm_rfft_fast_f32(&s, mic1_fft_in, mic1_fft_out, 0);
arm_rfft_fast_f32(&s, mic2_fft_in, mic2_fft_out, 0);
arm_rfft_fast_f32(&s, mic3_fft_in, mic3_fft_out, 0);
arm_rfft_fast_f32(&s, mic4_fft_in, mic4_fft_out, 0);
arm_rfft_fast_f32(&s, fm_fft_in, fm_fft_out, 0);

//对信号带通滤波
for(int16 i = 0;i < 500;i++)
{
    mic1_fft_out[i] = 0;
    mic2_fft_out[i] = 0;
    mic3_fft_out[i] = 0;
    mic4_fft_out[i] = 0;
}

```

```

    fm_fft_out[i] = 0;
}
for(int16 i = 4000; i < 4096;i++)
{
    mic1_fft_out[i] = 0;
    mic2_fft_out[i] = 0;
    mic3_fft_out[i] = 0;
    mic4_fft_out[i] = 0;
    fm_fft_out[i] = 0;
}

// (信号1)*(信号2的共轭)
// (a+bi)(c-di)=(ac+bd)+(bc-ad)i
for(int16 i=0;i<MIC_NUM*2;i+=2)
{
    acor1[i]=mic1_fft_out[i]*fm_fft_out[i]+mic1_fft_out[i+1]*fm_fft_out[i+1];
    acor1[i+1]=mic1_fft_out[i+1]*fm_fft_out[i]-mic1_fft_out[i]*fm_fft_out[i+1];

    acor2[i]=mic2_fft_out[i]*fm_fft_out[i]+mic2_fft_out[i+1]*fm_fft_out[i+1];
    acor2[i+1]=mic2_fft_out[i+1]*fm_fft_out[i]-mic2_fft_out[i]*fm_fft_out[i+1];

    acor3[i]=mic3_fft_out[i]*fm_fft_out[i]+mic3_fft_out[i+1]*fm_fft_out[i+1];
    acor3[i+1]=mic3_fft_out[i+1]*fm_fft_out[i]-mic3_fft_out[i]*fm_fft_out[i+1];

    acor4[i]=mic4_fft_out[i]*fm_fft_out[i]+mic4_fft_out[i+1]*fm_fft_out[i+1];
    acor4[i+1]=mic4_fft_out[i+1]*fm_fft_out[i]-mic4_fft_out[i]*fm_fft_out[i+1];
}

// 傅里叶反变换
arm_rfft_fast_f32(&s, acor1, acor1_ifft, 1);
arm_rfft_fast_f32(&s, acor2, acor2_ifft, 1);
arm_rfft_fast_f32(&s, acor3, acor3_ifft, 1);
arm_rfft_fast_f32(&s, acor4, acor4_ifft, 1);

// 寻找ifft变换结果最大幅值下标

seek_max(acor1_ifft, acor2_ifft, acor3_ifft, acor4_ifft, MAXACOR_TH, maxacor_index);

if(acor1_ifft[maxacor_index[0]] < PEAK_TH)
    maxacor_index[0] = 0;
if(acor2_ifft[maxacor_index[1]] < PEAK_TH)
    maxacor_index[1] = 0;
if(acor3_ifft[maxacor_index[2]] < PEAK_TH)

```



```

        maxacor_index[2] = 0;
        if(acor4_ifft[maxacor_index[3]] < PEAK_TH)
            maxacor_index[3] = 0;
    }

//计算距离和角度
//d1_in:左麦克到灯距离, d2_in:右麦克到灯距离, d_out , error_out两麦克中心到灯距离与偏差
//偏差左+右-
void dis_error_cal(float d1_in, float d2_in, float *d_out, float *error_out)
{
    *error_out = (d2_in - d1_in);
    d1_in*=3.4;
    d2_in*=3.4;
    *d_out = sqrtf((2*d1_in*d1_in + 2*d2_in*d2_in - MIC2MIC*MIC2MIC)/4);
}

//angle是beacon到mic中点的连线与x轴正向的夹角余弦值
//左- 右+
void dis_angle_cal(float d1_in, float d2_in, float *d_out, float *angle_out)
{
    d1_in*=3.4;
    d2_in*=3.4;
    *d_out = sqrtf((2*d1_in*d1_in + 2*d2_in*d2_in - MIC2MIC*MIC2MIC)/4);
    *angle_out = ((*d_out)*(*d_out)+0.25*MIC2MIC*MIC2MIC-
d2_in*d2_in)/(2*(*d_out)*MIC2MIC);
}

//计算距离和角度信息
void cal_beacon()
{
    if(!(maxacor_index[0]||maxacor_index[1]||maxacor_index[2]||maxacor_index[3]))
    {
        stop_flag=1;
    }
    else
    {
        stop_flag=0;
        if((maxacor_index[0]+maxacor_index[1]) <=
(maxacor_index[2]+maxacor_index[3]))
        {
            motor_dir = 1; //灯在前面

dis_error_cal(maxacor_index[0],maxacor_index[1],&beacon_distance,&beacon_error)

```

```

;

    if(beacon_error > ERROR_MAX)        beacon_error = ERROR_MAX;    //偏差限幅
    else if(beacon_error < -ERROR_MAX)  beacon_error = -ERROR_MAX;
}
else if((maxacor_index[0]+maxacor_index[1]) >
(maxacor_index[2]+maxacor_index[3]))
{
    motor_dir = -1;

dis_error_cal(maxacor_index[2],maxacor_index[3],&beacon_distance,&beacon_error)
;

    if(beacon_error > ERROR_MAX)        beacon_error = ERROR_MAX;
    else if(beacon_error < -ERROR_MAX)  beacon_error = -ERROR_MAX;
}

if(beacon_distance < TANGENT_TH)//切灯处理
{
    if(motor_dir == 1)
        beacon_error = (maxacor_index[1]+maxacor_index[2])-(
(maxacor_index[0]+maxacor_index[3]));
    else if(motor_dir == -1)
        beacon_error = (maxacor_index[0]+maxacor_index[3])-(
(maxacor_index[1]+maxacor_index[2]));

    if(beacon_error >= 0)//灯在左边
    {
        if(beacon_distance >=92)
            beacon_error -= 2;
        else if(beacon_distance >= 86)
            beacon_error -= 3;
        else if(beacon_distance >= 74)
            beacon_error -= 4;
        else if(beacon_distance >= 59)
            beacon_error -= 5;
        else if(beacon_distance >= 23)
            beacon_error -= 6;
        else if(beacon_distance >= 6)
            beacon_error -= 7;
        else
            beacon_error -= 8;
    }
else

```

```

{
    if(beacon_distance >=92)
        beacon_error += 2;
    else if(beacon_distance >= 86)
        beacon_error += 3;
    else if(beacon_distance >= 74)
        beacon_error += 4;
    else if(beacon_distance >= 59)
        beacon_error += 5;
    else if(beacon_distance >= 23)
        beacon_error += 6;
    else if(beacon_distance >= 6)
        beacon_error += 7;
    else
        beacon_error += 8;
}
}
}

void beacon_progress()
{
    voice_progress();
    cal_beacon();
}

```

附录 B 控制部分核心代码

```
#include "Motor.h"
#include "DirControl.h"
#include "Pid.h"
#include "zf_pwm.h"
#include "FuzzySet_Speed.h"
#include "VoiceProcess.h"
#include "Avoid.h"

int8 motor_dir=1;          //电机运动方向
int8 stop_flag=1;          //停车标志
int8 speed_level;
motor_status motor;

void motor_init()
{
    pwm_init(F_R_GO_PWMCH, MOTOR_FRE, INIT_DUTY);
    pwm_init(F_R_BACK_PWMCH, MOTOR_FRE, INIT_DUTY);

    pwm_init(F_L_GO_PWMCH, MOTOR_FRE, INIT_DUTY);
    pwm_init(F_L_BACK_PWMCH, MOTOR_FRE, INIT_DUTY);

    pwm_init(B_L_GO_PWMCH, MOTOR_FRE, INIT_DUTY);
    pwm_init(B_L_BACK_PWMCH, MOTOR_FRE, INIT_DUTY);

    pwm_init(B_R_GO_PWMCH, MOTOR_FRE, INIT_DUTY);
    pwm_init(B_R_BACK_PWMCH, MOTOR_FRE, INIT_DUTY);
}

void speed_control()
{
    int16 basic_speed = 5000; //限制速度，速度大了影响声音定位
    motor.speed_set = 10 * motor_dir * FuzzySet_Speed(0, (int16)beacon_distance);
    if(motor.speed_set>basic_speed)
        motor.speed_set = basic_speed;
    else if(motor.speed_set < (-basic_speed))
        motor.speed_set = (-basic_speed);

    // int16 basic_speed = 2500;
    // motor.speed_set = basic_speed * motor_dir;
```

```

    motor.speed_set_F_L = motor.speed_set;
    motor.speed_set_F_R = motor.speed_set;
    motor.speed_set_B_L = motor.speed_set;
    motor.speed_set_B_R = motor.speed_set;
}

//一阶滞后滤波法
//方法：取a = 0~1，本次滤波结果 = a*本次采样值 + (1-a)*上次滤波结果
//优点：对周期性干扰具有良好的抑制作用，适用于波动频率较高的场合
//缺点：相位滞后，灵敏度低，滞后程度取决于a的大小，不能消除滤波频率高于采样频率
1/2的干扰信号
void filter_speed()
{
}

void filter_speed_F_L()
{
    static float a = 0;
    float speed_error;
    speed_error = motor.speed_set_F_L - motor.speed_F_L_save[0];
    a = speed_error/30;//分母如何选定 每个周期减下来的速度*2+一些 具体需要调试
    if(a < 0)        a = -a;
    if(a > 0.85)     a = 0.85;
    motor.speed_set_F_L = (int16)(motor.speed_set_F_L*(1-a) +
(motor.speed_F_L_save[0]+motor.speed_F_L_save[1]+motor.speed_F_L_save[2])*a/3);
    motor.speed_F_L_save[2] = motor.speed_F_L_save[1];
    motor.speed_F_L_save[1] = motor.speed_F_L_save[0];
    motor.speed_F_L_save[0] = motor.speed_set_F_L;
}

void filter_speed_F_R()
{
    static float a = 0;
    float speed_error;
    speed_error = motor.speed_set_F_R - motor.speed_F_R_save[0];
    a = speed_error/30;//分母如何选定 每个周期减下来的速度*2+一些 具体需要调试
    if(a < 0)        a = -a;
    if(a > 0.85)     a = 0.85;
    motor.speed_set_F_R = (int16)(motor.speed_set_F_R*(1-a) +
(motor.speed_F_R_save[0]+motor.speed_F_R_save[1]+motor.speed_F_R_save[2])*a/3);
    motor.speed_F_R_save[2] = motor.speed_F_R_save[1];
    motor.speed_F_R_save[1] = motor.speed_F_R_save[0];
    motor.speed_F_R_save[0] = motor.speed_set_F_R;
}

```

```

void filter_speed_B_L()
{
    static float a = 0;
    float speed_error;
    speed_error = motor.speed_set_B_L - motor.speed_B_L_save[0];
    a = speed_error/30;//分母如何选定 每个周期减下来的速度*2+一些 具体需要调试
    if(a < 0)          a = -a;
    if(a > 0.85)       a = 0.85;
    motor.speed_set_B_L = (int16)(motor.speed_set_B_L*(1-a) +
(motor.speed_B_L_save[0]+motor.speed_B_L_save[1]+motor.speed_B_L_save[2])*a/3);
    motor.speed_B_L_save[2] = motor.speed_B_L_save[1];
    motor.speed_B_L_save[1] = motor.speed_B_L_save[0];
    motor.speed_B_L_save[0] = motor.speed_set_B_L;
}

void filter_speed_B_R()
{
    static float a = 0;
    float speed_error;
    speed_error = motor.speed_set_B_R - motor.speed_B_R_save[0];
    a = speed_error/30;//分母如何选定 每个周期减下来的速度*2+一些 具体需要调试
    if(a < 0)          a = -a;
    if(a > 0.85)       a = 0.85;
    motor.speed_set_B_R = (int16)(motor.speed_set_B_R*(1-a) +
(motor.speed_B_R_save[0]+motor.speed_B_R_save[1]+motor.speed_B_R_save[2])*a/3);
    motor.speed_B_R_save[2] = motor.speed_B_R_save[1];
    motor.speed_B_R_save[1] = motor.speed_B_R_save[0];
    motor.speed_B_R_save[0] = motor.speed_set_B_R;
}

void judge_speed_max()
{
    if(motor.speed_set_F_L > SPEED_MAX)
        motor.speed_set_F_L = SPEED_MAX;
    else if(motor.speed_set_F_L < SPEED_MIN)
        motor.speed_set_F_L = SPEED_MIN;

    if(motor.speed_set_F_R > SPEED_MAX)
        motor.speed_set_F_R = SPEED_MAX;
    else if(motor.speed_set_F_R < SPEED_MIN)
        motor.speed_set_F_R = SPEED_MIN;

    if(motor.speed_set_B_L > SPEED_MAX)
        motor.speed_set_B_L = SPEED_MAX;

```

```

else if(motor.speed_set_B_L < SPEED_MIN)
    motor.speed_set_B_L = SPEED_MIN;

if(motor.speed_set_B_R > SPEED_MAX)
    motor.speed_set_B_R = SPEED_MAX;
else if(motor.speed_set_B_R < SPEED_MIN)
    motor.speed_set_B_R = SPEED_MIN;
}

void break_down()
{
    pwm_duty(F_R_GO_PWMCH, 0);
    pwm_duty(F_R_BACK_PWMCH, 0);

    pwm_duty(F_L_GO_PWMCH, 0);
    pwm_duty(F_L_BACK_PWMCH, 0);

    pwm_duty(B_L_GO_PWMCH, 0);
    pwm_duty(B_L_BACK_PWMCH, 0);

    pwm_duty(B_R_GO_PWMCH, 0);
    pwm_duty(B_R_BACK_PWMCH, 0);
}

void pull_up()
{
    // motor.speed_set = 0;
    motor.speed_set_F_L = 0;
    motor.speed_set_F_R = 0;
    motor.speed_set_B_L = 0;
    motor.speed_set_B_R = 0;
}

void roll_wheels()
{
    judge_speed_max();
    filter_speed_F_L();
    filter_speed_F_R();
    filter_speed_B_L();
    filter_speed_B_R();
    pid_control(&pid_F_L, motor.speed_F_L, motor.speed_set_F_L);
    pid_control(&pid_F_R, motor.speed_F_R, motor.speed_set_F_R);
    pid_control(&pid_B_L, motor.speed_B_L, motor.speed_set_B_L);
    pid_control(&pid_B_R, motor.speed_B_R, motor.speed_set_B_R);
}

```

```

if(pid_F_L.result > 0)
{
    if(pid_F_L.result > SPEED_MAX)
        pid_F_L.result = SPEED_MAX;
    pwm_duty(F_L_GO_PWMCH, (uint32)pid_F_L.result);
    pwm_duty(F_L_BACK_PWMCH, 0);
}
else
{
    if(pid_F_L.result < SPEED_MIN)
        pid_F_L.result = SPEED_MIN;
    pwm_duty(F_L_BACK_PWMCH, (uint32)(-pid_F_L.result));
    pwm_duty(F_L_GO_PWMCH, 0);
}

if(pid_F_R.result > 0)
{
    if(pid_F_R.result > SPEED_MAX)
        pid_F_R.result = SPEED_MAX;
    pwm_duty(F_R_GO_PWMCH, (uint32)pid_F_R.result);
    pwm_duty(F_R_BACK_PWMCH, 0);
}
else
{
    if(pid_F_R.result < SPEED_MIN)
        pid_F_R.result = SPEED_MIN;
    pwm_duty(F_R_BACK_PWMCH, (uint32)(-pid_F_R.result));
    pwm_duty(F_R_GO_PWMCH, 0);
}

if(pid_B_L.result > 0)
{
    if(pid_B_L.result > SPEED_MAX)
        pid_B_L.result = SPEED_MAX;
    pwm_duty(B_L_GO_PWMCH, (uint32)pid_B_L.result);
    pwm_duty(B_L_BACK_PWMCH, 0);
}
else
{
    if(pid_B_L.result < SPEED_MIN)
        pid_B_L.result = SPEED_MIN;
    pwm_duty(B_L_BACK_PWMCH, (uint32)(-pid_B_L.result));
    pwm_duty(B_L_GO_PWMCH, 0);
}

```



```

}

if(pid_B_R.result > 0)
{
    if(pid_B_R.result > SPEED_MAX)
        pid_B_R.result = SPEED_MAX;
    pwm_duty(B_R_GO_PWMCH, (uint32)pid_B_R.result);
    pwm_duty(B_R_BACK_PWMCH, 0);
}
else
{
    if(pid_B_R.result < SPEED_MIN)
        pid_B_R.result = SPEED_MIN;
    pwm_duty(B_R_BACK_PWMCH, (uint32)(-pid_B_R.result));
    pwm_duty(B_R_GO_PWMCH, 0);
}
}

```

```

void mecanum_control()
{
    if(motor.speed_set_F_L >= 0) //左前
    {
        if(motor.speed_set_F_L > SPEED_MAX)
            motor.speed_set_F_L = SPEED_MAX;

        pwm_duty(F_L_GO_PWMCH, motor.speed_set_F_L);
        pwm_duty(F_L_BACK_PWMCH, 0);
    }
    else
    {
        if(motor.speed_set_F_L < SPEED_MIN)
            motor.speed_set_F_L = SPEED_MIN;

        pwm_duty(F_L_BACK_PWMCH, (-motor.speed_set_F_L));
        pwm_duty(F_L_GO_PWMCH, 0);
    }

    if(motor.speed_set_F_R >= 0) //右前
    {
        if(motor.speed_set_F_R > SPEED_MAX)
            motor.speed_set_F_R = SPEED_MAX;

```

```

    pwm_duty(F_R_GO_PWMCH, motor.speed_set_F_R);
    pwm_duty(F_R_BACK_PWMCH, 0);
}
else
{
    if(motor.speed_set_F_R < SPEED_MIN)
        motor.speed_set_F_R = SPEED_MIN;

    pwm_duty(F_R_BACK_PWMCH, (-motor.speed_set_F_R));
    pwm_duty(F_R_GO_PWMCH, 0);
}

if(motor.speed_set_B_L >= 0)
{
    if(motor.speed_set_B_L > SPEED_MAX)
        motor.speed_set_B_L = SPEED_MAX;

    pwm_duty(B_L_GO_PWMCH, motor.speed_set_B_L);
    pwm_duty(B_L_BACK_PWMCH, 0);
}
else
{
    if(motor.speed_set_B_L < SPEED_MIN)
        motor.speed_set_B_L = SPEED_MIN;

    pwm_duty(B_L_BACK_PWMCH, (-motor.speed_set_B_L));
    pwm_duty(B_L_GO_PWMCH, 0);
}

if(motor.speed_set_B_R >= 0)
{
    if(motor.speed_set_B_R > SPEED_MAX)
        motor.speed_set_B_R = SPEED_MAX;

    pwm_duty(B_R_GO_PWMCH, motor.speed_set_B_R);
    pwm_duty(B_R_BACK_PWMCH, 0);
}
else
{
    if(motor.speed_set_B_R < SPEED_MIN)
        motor.speed_set_B_R = SPEED_MIN;

    pwm_duty(B_R_BACK_PWMCH, (-motor.speed_set_B_R));
    pwm_duty(B_R_GO_PWMCH, 0);
}

```

```

    }
}

void car_control()
{
    if(stop_flag)
    {
        break_down();
    }
    else
    {
        speed_control();
        direction_control();
        avoid_obstacle();
        //      mecanum_control(); //开环控制
        roll_wheels();      //PI闭环控制
    }
}

#include "DirControl.h"
#include "Motor.h"
#include "VoiceProcess.h"
#include "Pid.h"
#include "SEEKFREE_MPU6050.h"
#include "stdlib.h"

#define DIR_MAX 2000
#define DIR_MIN -2000

#define TRANSLATE_TH 60 //方向环变成横移档

dir_pid_t pid_dir;
float dir_error[5];

//陀螺仪辅助
void direction_control()
{
    //  filter_error();

    if((abs(maxacor_index[0]-maxacor_index[3])<= 1) || (abs(maxacor_index[1]-
maxacor_index[2])<= 1))
    {
        pid_dir.kp = 380;
    }
}

```

```

    pid_dir.kd = 0.36;
}
else
{
    pid_dir.kp = 360.0;
    pid_dir.kd = 0.39;
}

get_gyro();
pid_dir.result = (int16)((pid_dir.kp)*(beacon_error)-pid_dir.kd *
mpu_gyro_z);

if(pid_dir.result > DIR_MAX)//防止车转太快
    pid_dir.result = DIR_MAX;
if(pid_dir.result < DIR_MIN)
    pid_dir.result = DIR_MIN;

motor.speed_set_F_L -= pid_dir.result;
motor.speed_set_F_R += pid_dir.result;
motor.speed_set_B_L -= pid_dir.result;
motor.speed_set_B_R += pid_dir.result;
}

////偏差之差
//void direction_control()
//{
//    pid_dir.kp = 120.0;
//    pid_dir.kd = 10;
//
//    pid_dir.err_last=pid_dir.err_current;
//    pid_dir.err_current=beacon_error;
//
//    pid_dir.result = (int16)((pid_dir.kp)*(beacon_error)+pid_dir.kd *
(pid_dir.err_current-pid_dir.err_last));
//
//    if(pid_dir.result > DIR_MAX)
//        pid_dir.result = DIR_MAX;
//    if(pid_dir.result < DIR_MIN)
//        pid_dir.result = DIR_MIN;
//
//    motor.speed_set_F_L -= pid_dir.result;
//    motor.speed_set_F_R += pid_dir.result;
//    motor.speed_set_B_L -= pid_dir.result;
//    motor.speed_set_B_R += pid_dir.result;

```

```

//}

void filter_error()
{
    dir_error[0]=beacon_error;
    static float a = 0;
    float sigal_error;
    sigal_error = dir_error[0] - dir_error[1];
    a = sigal_error/10;           //分母越小越滞后
    if(a < 0)
        a = -a;
    if(a > 0.85)
        a = 0.85;
    dir_error[0] = (1-a)*dir_error[0]+a*dir_error[1];
    dir_error[4] = dir_error[3];
    dir_error[3] = dir_error[2];
    dir_error[2] = dir_error[1];
    dir_error[1] = dir_error[0];
}

#include "Avoid.h"
#include "zf_pit.h"
#include "motor.h"
#include "VoiceProcess.h"
#include "TFmini.h"

#define SAFE_TH      100        //安全避障距离
#define DANGER_TH    35         //紧急避障距离

#define DRIFT_SPEED_S 800       //安全区侧移速度
#define DRIFT_SPEED_D 2000      //危险区侧移速度

uint8 avoid_flag=0;            //是否避障标志
int8 avoid_dir=LEFT;           //避障方向
uint8 merge_flag = 0;          //灯和障碍一个东西为1
AvoidMode_e avoid_mode;        //避障模式

/*
 * 激光雷达避障
 *
 *      TF1      TF2
 *
 */

```

```

*
*
*      TF3      TF4
*/

void avoid_check()
{
    if(motor_dir == 1)
    {
        if((block_distance1 <= DANGER_TH && block_distance1 > 0) ||
(block_distance2 <= DANGER_TH && block_distance2 > 0))//雷达有避障提示
        {
            avoid_flag = 1;
            avoid_mode = TRANSLATE_D;
            if(block_distance1 >= block_distance2
&&(block_distance1!=0)&&(block_distance2!=0))
                avoid_dir = LEFT;
            else if(block_distance1 <= block_distance2
&&(block_distance1!=0)&&(block_distance2!=0))
                avoid_dir = RIGHT;
        }
        else if((block_distance1 <= SAFE_TH && block_distance1 > DANGER_TH) ||
(block_distance2 <= SAFE_TH && block_distance2 > DANGER_TH))
        {
            if(beacon_distance > TANGENT_TH)
            {
                avoid_flag = 1;
                avoid_mode = TRANSLATE_S;
                if(beacon_error >= 0)//灯在左
                {
                    avoid_dir = LEFT;
                }
                else
                {
                    avoid_dir = RIGHT;
                }
            }
        }
    }
    else if(motor_dir == -1)
    {
        if((block_distance4 <= DANGER_TH && block_distance4 > 0) ||
(block_distance3 <= DANGER_TH && block_distance3 > 0))//雷达有避障提示
        {

```

```

        avoid_flag = 1;
        avoid_mode = TRANSLATE_D;
        if(block_distance4 >= block_distance3
        &&(block_distance4!=0)&&(block_distance3!=0))
            avoid_dir = LEFT;
        else if(block_distance4 <= block_distance3
        &&(block_distance4!=0)&&(block_distance3!=0))
            avoid_dir = RIGHT;
    }
    else if((block_distance4 <= SAFE_TH && block_distance4 > DANGER_TH) ||
    (block_distance3 <= SAFE_TH && block_distance3 > DANGER_TH))
    {
        if(beacon_distance > TANGENT_TH)
        {
            avoid_flag = 1;
            avoid_mode = TRANSLATE_S;
            if(beacon_error >= 0)//灯在左
            {
                avoid_dir = LEFT;
            }
            else
            {
                avoid_dir = RIGHT;
            }
        }
    }
}
}

```

```

void avoiding()
{
    switch(avoid_mode)
    {
        case TRANSLATE_S:
        {
            if(avoid_dir == LEFT)
            {
                motor.speed_set_F_L -= DRIFT_SPEED_S * motor_dir;
                motor.speed_set_F_R += DRIFT_SPEED_S * motor_dir;
                motor.speed_set_B_L += DRIFT_SPEED_S * motor_dir;
                motor.speed_set_B_R -= DRIFT_SPEED_S * motor_dir;
            }
            else if(avoid_dir == RIGHT)

```

```

        {
            motor.speed_set_F_L += DRIFT_SPEED_S * motor_dir;
            motor.speed_set_F_R -= DRIFT_SPEED_S * motor_dir;
            motor.speed_set_B_L -= DRIFT_SPEED_S * motor_dir;
            motor.speed_set_B_R += DRIFT_SPEED_S * motor_dir;
        }
    };break;
case TRANSLATE_D:
{
    if(avoid_dir == LEFT)
    {
        motor.speed_set_F_L = -DRIFT_SPEED_D * motor_dir;
        motor.speed_set_F_R = DRIFT_SPEED_D * motor_dir;
        motor.speed_set_B_L = DRIFT_SPEED_D * motor_dir;
        motor.speed_set_B_R = -DRIFT_SPEED_D * motor_dir;
    }
    else if(avoid_dir == RIGHT)
    {
        motor.speed_set_F_L = DRIFT_SPEED_D * motor_dir;
        motor.speed_set_F_R = -DRIFT_SPEED_D * motor_dir;
        motor.speed_set_B_L = -DRIFT_SPEED_D * motor_dir;
        motor.speed_set_B_R = DRIFT_SPEED_D * motor_dir;
    }
};break;

}
}

void avoid_obstacle()
{
    avoid_check();
    if(avoid_flag)
    {
        avoiding();
        avoid_flag = 0;
    }
}

#include "headfile.h"
#include "include.h"
#include "isr.h"
#include "Init.h"
#include "VoiceProcess.h"

```



```

float var[8];

void main()
{
    DisableGlobalIRQ();
    board_init();          //务必保留，本函数用于初始化MPU 时钟 调试串口
    systick_delay_ms(100); //延时100ms，等待主板其他外设上电成功

    Init();                //外设等初始化函数

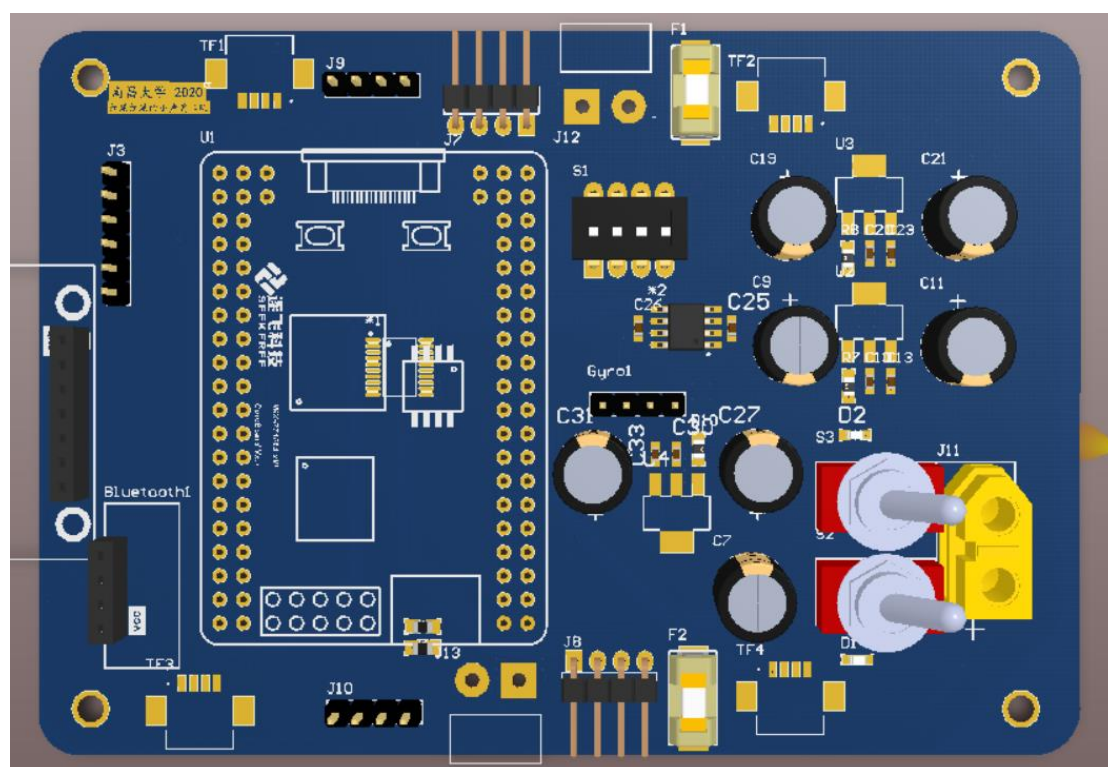
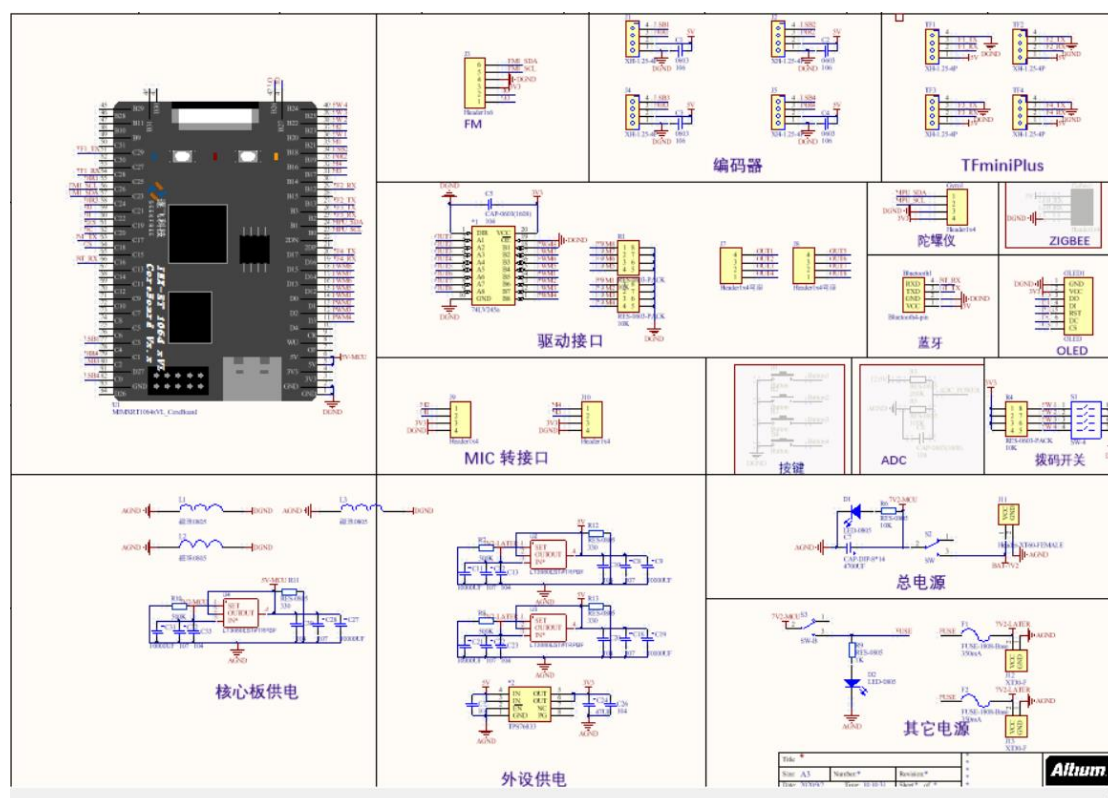
    EnableGlobalIRQ(0);

    while(1)
    {
        if(sample_finish)
        {
            sample_finish=0;
            beacon_progress();
        }
        car_control();
        oled_printf_int32(0,0,maxacor_index[0],5);
        oled_printf_int32(63,0,maxacor_index[1],5);
        oled_printf_int32(63,1,maxacor_index[2],5);
        oled_printf_int32(0,1,maxacor_index[3],5);
        oled_printf_float(0,2,beacon_error,5,5);
        oled_printf_float(63,2,beacon_distance,5,5);
        oled_printf_int32(0,3,mpu_gyro_z,10);

        oled_printf_int32(0,4,block_distance1,5);
        oled_printf_int32(63,4,block_distance2,5);
        oled_printf_int32(0,5,block_distance3,5);
        oled_printf_int32(63,5,block_distance4,5);
    }
}

```

附录 C 主板原理图和 PCB



附录 D 驱动原理图和 PCB

