第十八届全国大学生智能汽车竞赛

技 术 报 告

学　　校：山东大学（威海）

队伍名称：山魂三队

参赛队员：赵俊民、唐靖、赵一鸣

带队教师：王小利

关于研究论文使用授权的说明

本人完全了解全国大学生智能汽车竞赛关保留、使用研究论文的规定，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：2023.8.6

摘 要

本文设计的智能车系统以英飞凌公司 Tricore 系列的 TC264 微控制器为核心控制单元，采用大赛组委会统一指定的 O 型独轮车模，基于 ADS 编译环境进行整车软件程序的编写，主要介绍独轮车系统的原理以及软硬件设计，具体包括机械结构的设计、电路设计与制作、车模平衡控制以及图像处理思路等。为完成指定的循迹任务，独轮车模通过编码器采集脉冲获取电机的实时转速，通过陀螺仪获取车身的姿态，基于串级 PID 通过动量轮实现车模的平衡和转向的闭环控制，控制参数的调试主要基于上位机和人机交互界面。车模上的红外传感器用来判断赛道上的横断路障，电磁杆用来进行断路的循迹。图像处理主要使用灰度摄像头检测赛道信息，以此来提取赛道信息，同时获取赛道的左右边界和中线，并对特殊元素进行补线操作得到实际中线与理论中线之差，以此作为转向控制的原始输入，完成整个赛道的循迹任务。

关键字：TC264；串级PID；上位机；灰度摄像头；图像处理

目录

第一章 引言

1.1 智能车研究背景

智能车作为当今科技领域的热门研究方向之一，已经在交通出行、工业自动化、智能物流等领域展现出巨大的潜力和应用前景。全国大学生智能汽车竞赛作为我国智能车领域最具影响力的赛事之一，不仅为大学生提供了一个锻炼自己技术和创新能力的平台，也推动了智能车领域的研究和发展。

全国大学生智能汽车竞赛是以“立足培养、重在参与、鼓励探索、追求卓越”为指导思想，鼓励创新的一项科技竞赛活动。竞赛要求在规定的汽车模型平台上，使用 Infineon 系列单片机作为核心控制模块，通过增加道路传感器、电机驱动模块以及编写相应控制程序，制作完成一个能够自主循迹的模型汽车。

对于独轮车车组别来说，这是一个更具挑战性的竞赛项目。独轮车的平衡和控制需要更高的技术要求，涉及到传感器数据的获取和处理、动态平衡控制、路径规划和决策等多个方面的技术。通过参加独轮车车组别的竞赛，可以更深入地研究和探索这些技术，提高参赛选手的技术水平和创新能力。

在本报告中我们将通过对小车设计制作的电路、结构、算法、调试等的介绍,尽力展现我们一年以来付出的心血与汗水。

1.2 本文的概况及结构安排

第一章，叙述智能车研究背景，给出本文的结构安排。

第二章，叙述本队独轮车的总体方案设计以及整体的布局情况。

第三章，介绍小车的硬件电路设计，包括主控芯片、电源模块、驱动电路、电磁信号检测板等。

第四章，介绍小车的机械结构。

第五章，重点介绍独轮车平衡与转向的原理。

第六章，重点介绍摄像头算法的原理。

第七章，介绍系统开发环境与调试。

第二章 系统总体设计

2.1 系统总体方案设计

智能车的核心单元采用英飞凌半导体公司设计生产的 TC264 芯片。该芯片采用双核 TriCore 架构，最高主频为 200MHz，高达 2.5MB 的闪存与 240KB 的 RAM，完全满足智能车控制的算力需求。系统主要包括信号采集系统、信号处理系统和控制系统。赛道信息主要通过 MT9V03X 总钻风摄像头进行采集，将采集到的各种数据经单片机系统处理之后通过对电机的控制实现速度闭环。此外增加了人机交互系统，通过按键与屏幕可以实现对系统参数的直接设置。系统总体方案设计框图如图 2.1。

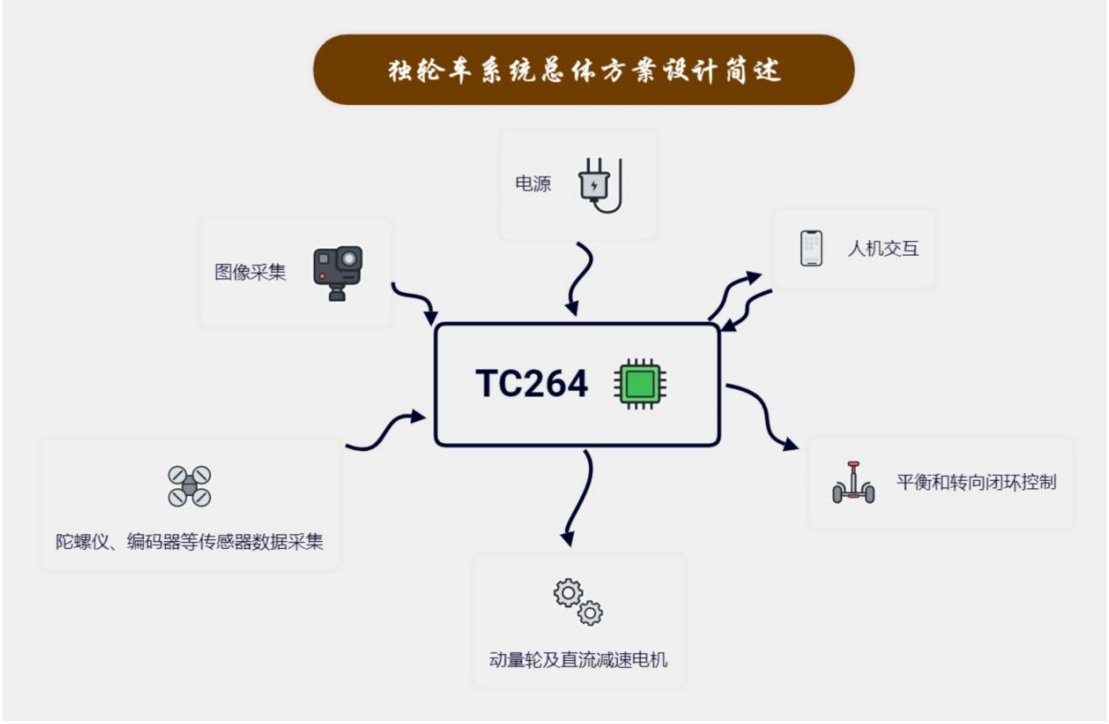


图 2.1 系统总体方案设计框图

2.2 智能车总体布局

我们使用的是竞赛指定的独轮 0 车模，该车模前后具有两个内置驱动电路的无刷电机，用于独轮车的平衡和转向控制；同时直流电机控制车模前后的平衡与行进。为了更好地控制平衡，降低调试难度，在车模搭建时具有以下特点：

(1) 主板和驱动板采用接插式，有效降低了电路板面积，更适应 0 车模结构。

(2) 采用两片 3S 电池串联组成的 6S 电池组，3S 电池分别放置在车模两端，在降低车模重心的同时具有良好的对称性。

(3)为使车模运行过程中更加平稳，我们将陀螺仪安装在双无刷电机的中心以更加精确地反映小车当前的姿态来进行小车运动时的控制，以及对特殊元素的辅助判断。

(4) 自行设计的太极形状飞轮保护壳，在保护赛道与队员的同时也象征了独轮车阴阳调和、平衡统一的特点。

(5) 摄像头支架与电磁前瞻板放置在车模靠前的位置，红外测距模块放置在电磁前瞻板下方，使用热熔胶进行固定，以便更加精确地判断赛道元素。

智能车的外形大致如下图所示。

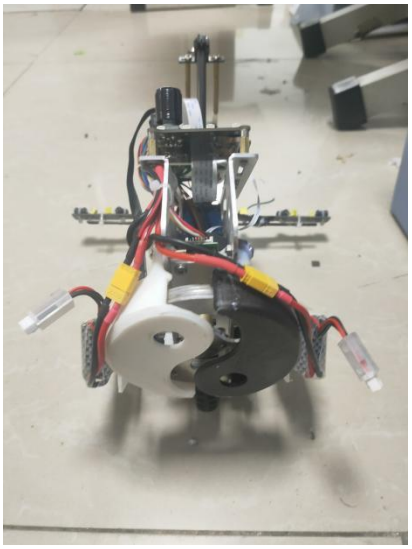


图 2.2.1 小车背面照

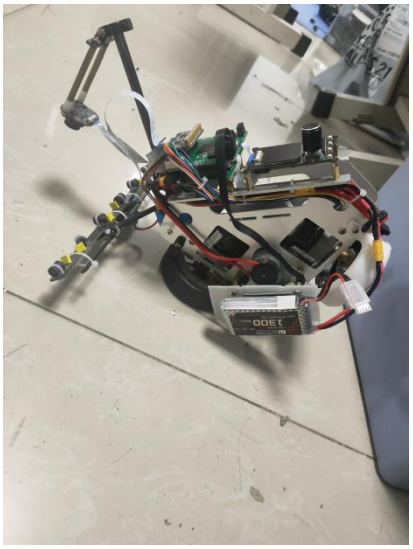


图 2.2.2 小车侧面照

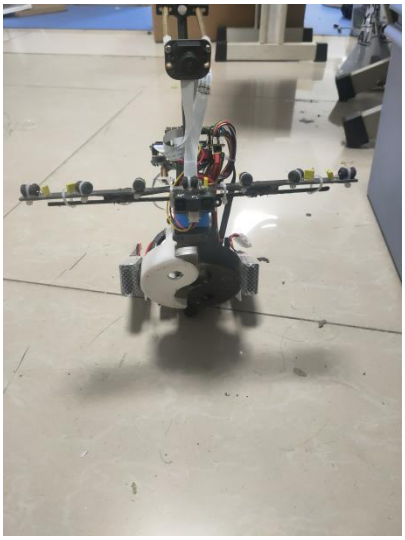


图 2.2.3 小车正面照

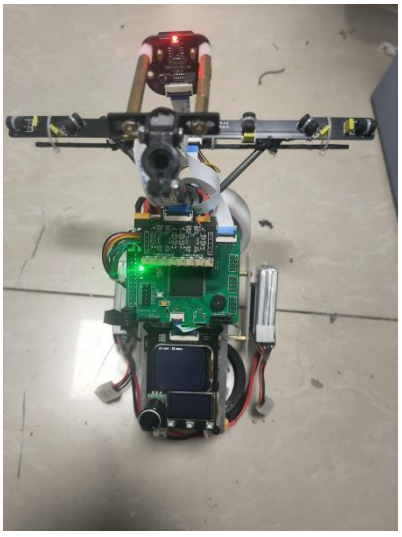


图 2.2.4 小车俯视照

第三章 硬件电路设计

3.1 主控芯片

核心单片机子系统采用英飞凌半导体公司设计生产的 TC264 芯片。该芯片采用 32 位 TriCore 架构，具有多个处理核心，支持高性能计算和多线程操作。此外 TC264 还有带纠错编码 ECC 保护的最大能到 2.5MB 闪存；48 个 DMA 通道；功能强大的 GTM 模块； Flexrey、CAN、CAN FD、LIN、SPI 接口；待机控制；单电源 5V 或 3.3V 供电等。为了减轻电路板重量，我们根据官方数据手册设计了单片机最小系统电路，具体原理图如图 3.1.1 所示：

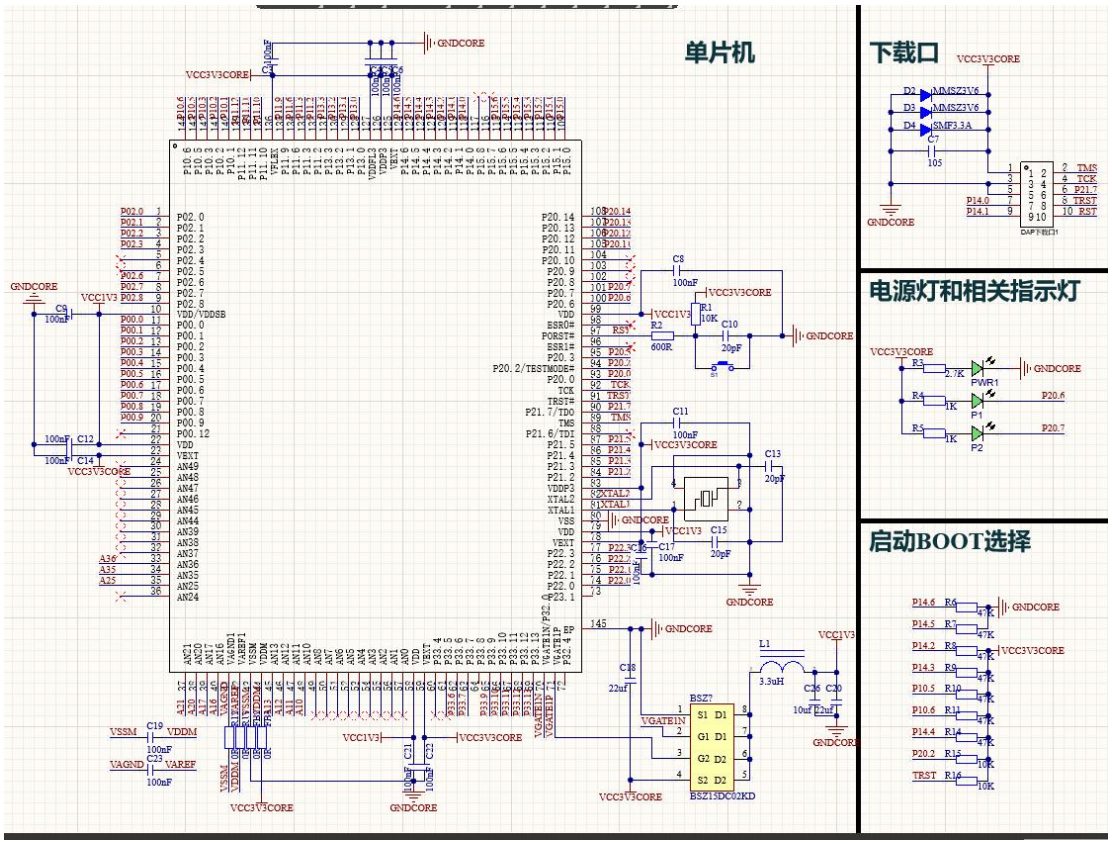


图 3.1.1 TC264 最小系统电路原理图

3.2 稳压电路

为了更好地降低独轮车重心，便于平衡控制，电路电源采用两片 3s 电池串联组成的 6s 电池组供电。由于各个模块需要的电压不同，需要将电源电压进行降压处理，主要分为以下两个部分：

1.采用 LM2596S-5.0 稳压芯片将电源电压降压至 5V 供电,可以为无线串口、测距模块进行供电，还可以用于后续降压 3.3V。原理图如图 3.2.1 所示。

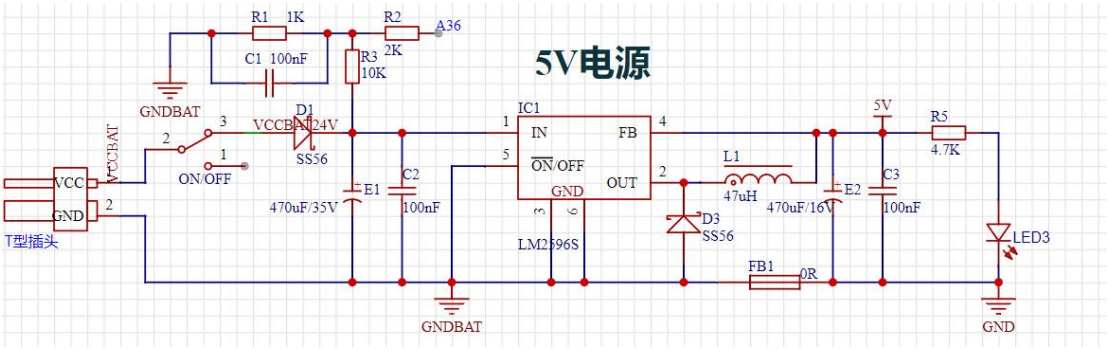


图 3.2.1　5V 稳压电路原理图

2.采用多路的 RT9013-33 稳压芯片将 5V 电压降压至 3.3V 供电，包括单片机、OLED 接口、ICM20602 等。原理图如图 3.2.2 所示。

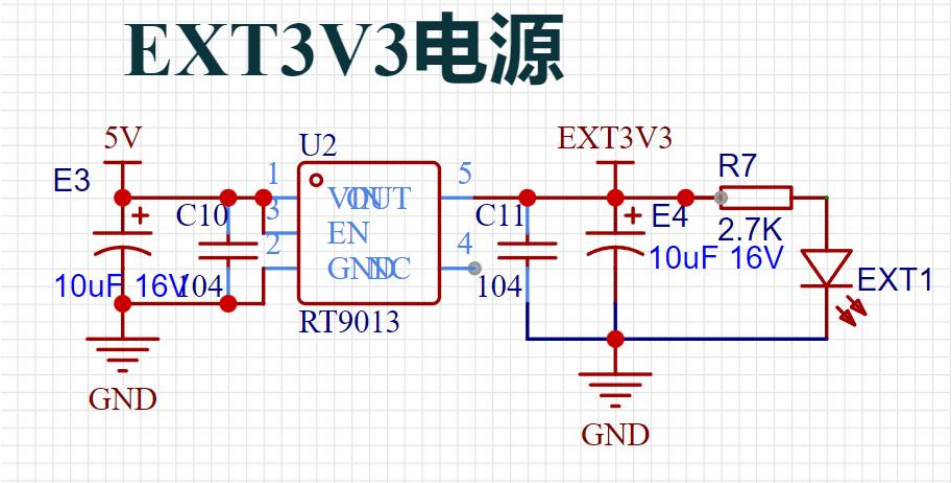


图 3.2.2　3.3V 稳压电路原理图

3.3 驱动电路

驱动芯片我们选择 TI 公司生产的 drv8701e，该芯片可通过 PH /EN 接口轻松连接控制器电路。内置的感测放大器能够实现可调节的电流控制。同时该芯片内置以下保护功能：欠压锁定，电荷泵故障，过流关断，短路保护，前置驱动器故障以及过热保护。外部 H 桥电路的 N 通道 MOSFET 选择东芝半导体公司的 TPH1R403NL，其内部 DS 导通内阻更小，只有 1.7 毫欧，瞬间峰值电流可达 200A，持续电流可达 60A，具有非常良好的性能。此外为了防止电机堵转等情况电流回流单片机造成单片机损坏，采用 74HC125 隔离芯片进行单片机与驱动板的隔离。单路驱动电路如图 3.3.1 所示。

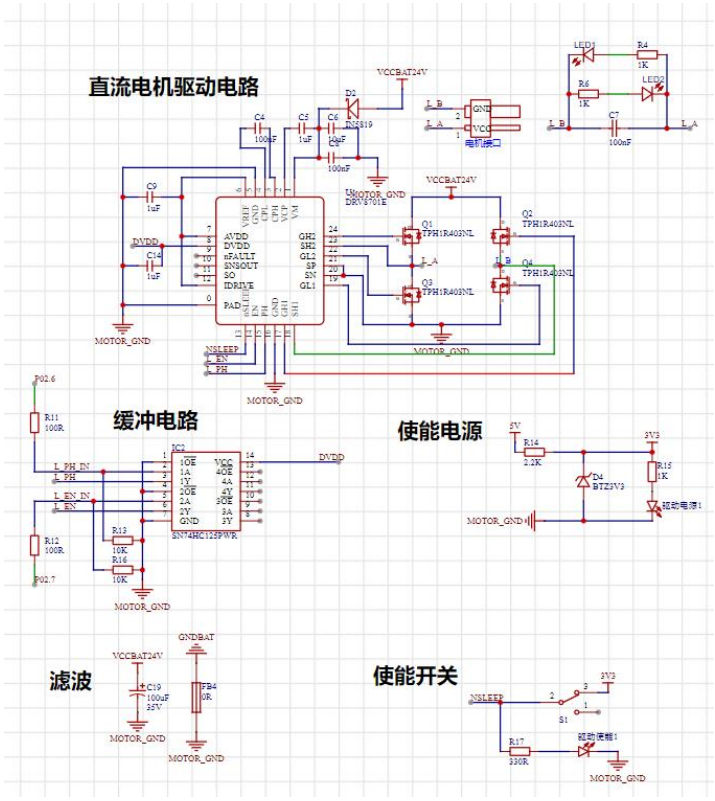


图 3.3.1 单路驱动电路原理图

3.4 电磁信号检测与放大电路

由于电磁断路的存在，尽管在独轮车组别有可以直接回库的规则，为了保险起见还是增加了电磁信号检测与放大模块，通过总共七路的 LC 谐振回路检测赛道场地的电磁信息，使用 OPA4377 运放芯片进行信号的放大并连接到单片机上的 ADC 接口。具体电路图如图 3.4.1 与 3.4.2 所示。

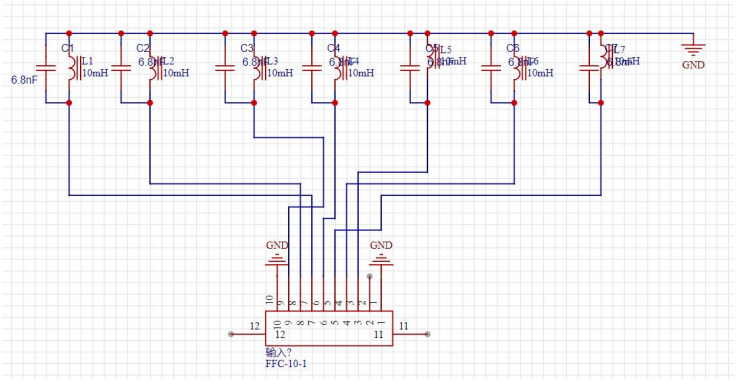


图 3.4.1 信号检测电路原理图

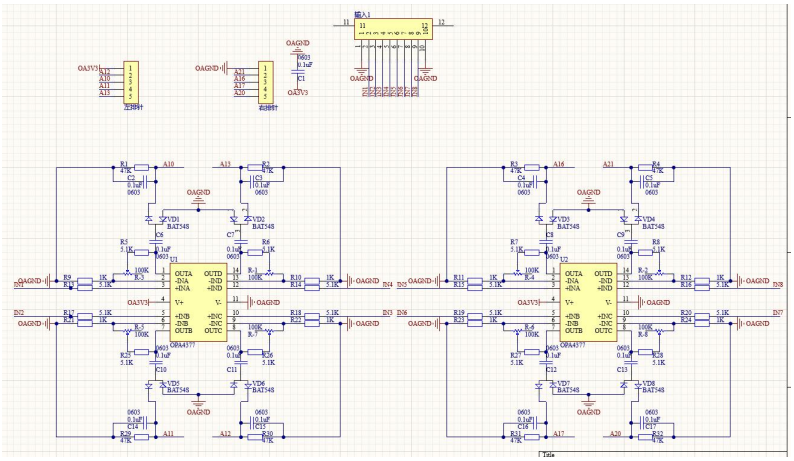


图 3.4.2 八路运放电路原理图

3.5 人机交互

由于赛道元素的复杂性和独轮车特殊回库规则的存在，设计出良好的人机交互模块是非常有必要的，可以大大节省调试的时间。我们设计的人机交互模块具有两块 IPS 屏幕、若干按键和一个旋转编码器，具体原理图如图 3.5.1 所示。

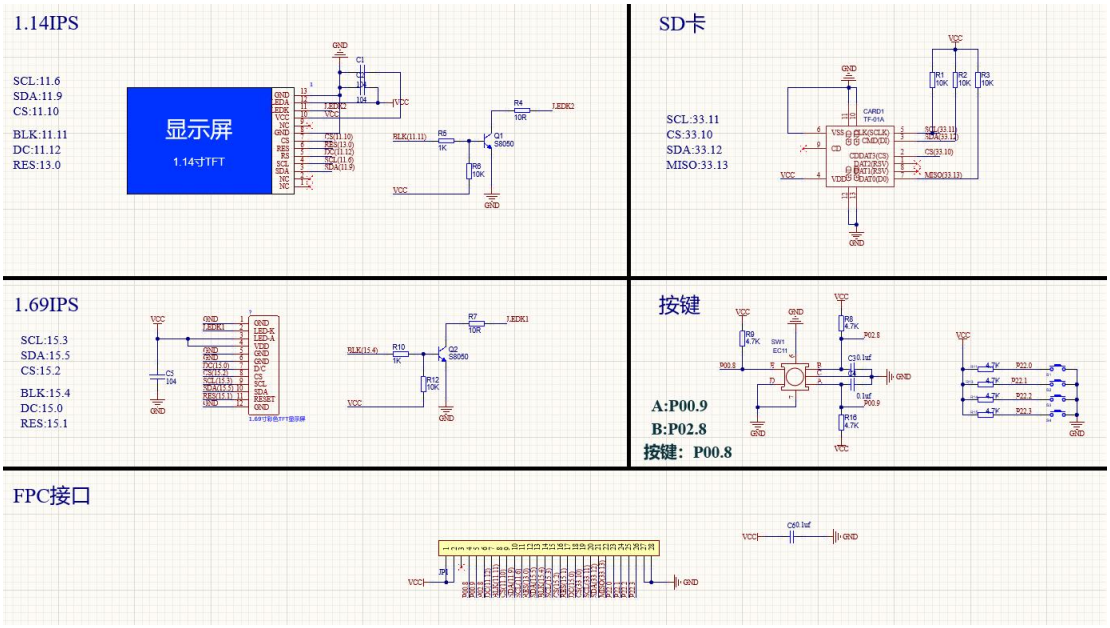


图 3.5.1 屏幕按键电路原理图

第四章 机械结构安装

4.1 主板、驱动板与人机交互模块的安装

由于独轮车 0 车模空间的局限性，我们采用接插式电路设计，这样可以节省空间，提升有效利用面积。使用点焊柱将底部的电源和单路驱动模块固定在独轮车上方，主板及其外设模块通过排针插在底部模块上方，再通过 fpc 软排线连接人机交互模块与主板，具体布局如图 4.1 所示。

4.2 陀螺仪位置

为了使陀螺仪位置更加靠近车模重心附近，降低平衡的调试难度，我们设计了陀螺仪转接板，通过 fpc 排线从主板引出到转接板上，再将逐飞的 icm20602 模块焊接在转接板上，通过热熔胶固定在 0 车模内部的两个无刷电机正中间。陀螺仪的安装位置如图 4.2 所示。

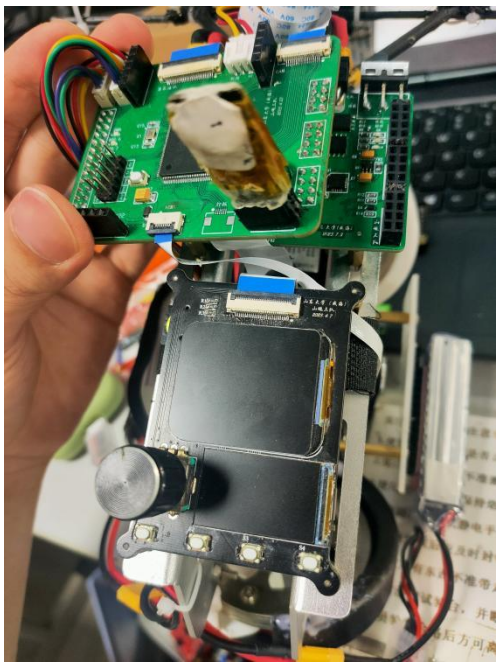


图 4.1 主板、驱动板与人机交互模块的布局

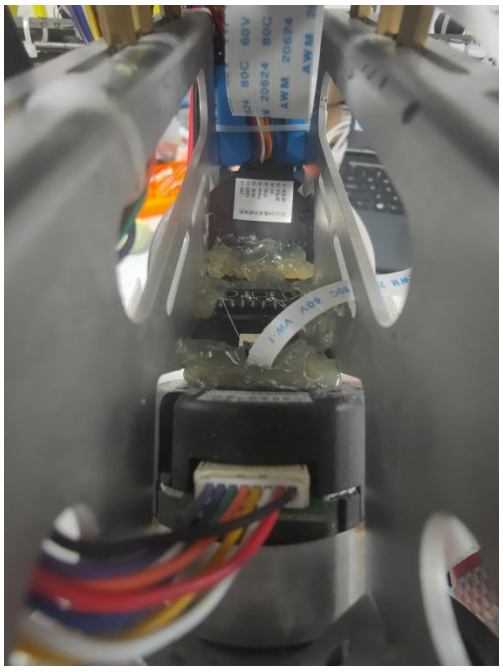


图 4.2 陀螺仪安装位置

4.3 摄像头与电磁前瞻的安装

我们使用一块大小正好的积木通过钻孔固定在独轮车前部，在积木上通过打孔和使用热熔胶的方式同时安装了高约14cm的摄像头单杆支架和电磁前瞻模块。如图4.3所示。

图 4.3 摄像头与电磁前瞻的安装

4.4 电池组的安装

0 车模的无刷电机额定电压为 24V，因此需要使用升压电路或者直接使用 6S 电池为其供电，我们选择直接使用 6S 电池。一开始我们使用的是如图 4.4 所示的单个 6S 电池塞在车模中心，后来发现这样会造成重心偏高，为平衡带来了难度。因此我们后续采用两块 3S 电池串联组成的电池组，分别固定在车模两侧。

实践证明这样的结构具有更低的重心和和更稳定的平衡效果，相应的布局如图4.5所示。

图 4.4 单个 6S 电池

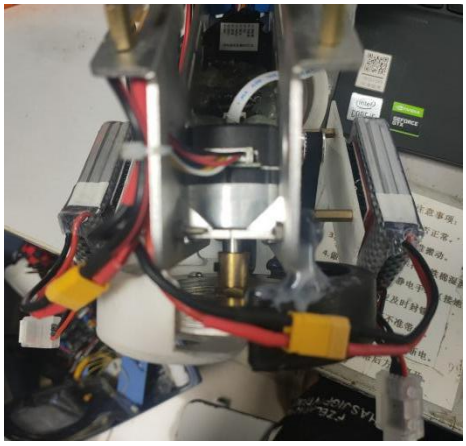


图 4.5 两片 3S 电池串联的电池

第五章 独轮车原理分析

5.1 总述

独轮车作为第十八届全国大学生智能汽车竞赛中的一个全新的组别，有着与往年所有车模都不一样的结构，因此我们需要在竞赛要求的基础上，搭建一个可以自主保持两个自由度的平衡并且完成转向的系统，同时独轮车系统要具有相应的环境感知能力，并能在运行过程中能够具有一定的抗干扰能力，由此稳定的完成比赛的任务。本章节主要介绍独轮车模的平衡和转向原理以及相应的控制方案，独轮车模实现平衡和转向的硬件支撑主要是 45°对称放置的动量轮、直流减速电机以及型号为 icm20602 的六轴陀螺仪，软件实现主要基于串级 PID、四元数以及巴特沃斯滤波。

5.2 独轮平衡原理

如果对独轮车平衡控制系统进行建模，可以发现独轮车模的结构虽然不同于自行车这一类自平衡系统，但是理论上来说跟单车的控制系统拥有相同的运动学模型，我们都可以化用倒立摆模型对其平衡控制系统进行建模，相较于单车，独轮车的平衡控制可以看作是两个倒立摆模型，如下图 5.1 所示：

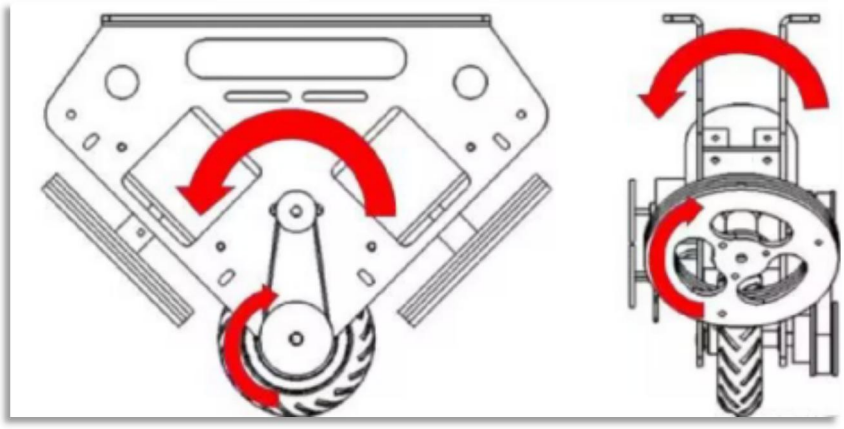


图 5.1 独轮车等效倒立摆模型

倒立摆是控制系统领域中一个很基本但又很实用的问题，很多基本的控制逻辑和实现都可以通过倒立摆模型表示出来，下面采用拉格朗日方程的方法对

倒立摆进行简单的运动学模型建立，为方便起见，此处以独轮车模翻滚角的平衡控制为例，可将其等效为如下倒立摆模型：

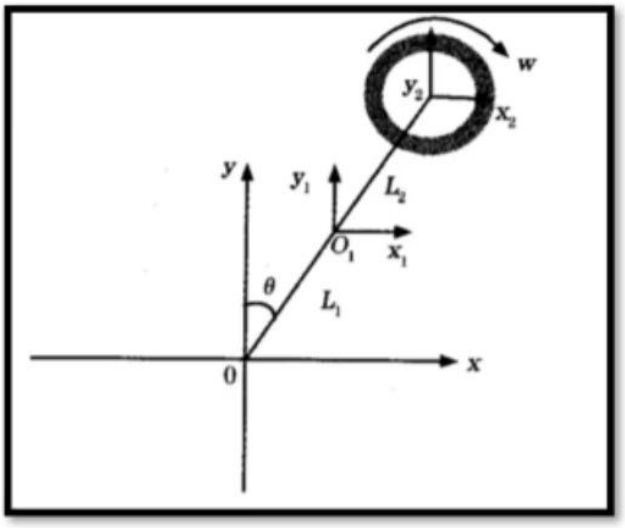


图 5.2 典型倒立摆模型

利用拉格朗日方程：$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}})-\frac{\partial L}{\partial q}-\frac{\partial D}{\partial \dot{q}}=f_i$ ，首先分析系统中的广义坐标有两个： θ 和 ϕ，因此拉格朗日方程可写成如下形式：

$$\begin{cases}\dfrac{d}{dt}\left(\dfrac{\partial L}{\partial \dot{\theta}}\right)-\dfrac{\partial L}{\partial \theta}-\dfrac{\partial D}{\partial \dot{\theta}}=f_1\\ \dfrac{d}{dt}\left(\dfrac{\partial L}{\partial \dot{\varphi}}\right)-\dfrac{\partial L}{\partial \varphi}-\dfrac{\partial D}{\partial \dot{\varphi}}=f_2\end{cases}$$

系统的总动能可以表示为：$T=T_{\theta_1}+T_{\theta_2}$ ，式中，T_{θ_1} 和 T_{s_2} 分别为握杆和飞轮的动能。

摆杆的动能：$T_{v_1}=\left(m_1 v_{s_1}^{T} v_{o_1}+I_1\dot{\theta}^2\right)/2$

飞轮的动能：$T_{v_2}=\left(m_2 v_{c_2}^{\tau} v_{\theta_2}+I_2\dot{\theta}^2+I_2\dot{\phi}^2\right)/2$

系统的总动能为：$T=\left(m_1 v_{o_1}^{T} v_{q_2}+I_1\dot{\theta}^2\right)/2+\left(m_2 v_{o_2}^{\tau} v_{q_2}+I_2\dot{\theta}^2+I_2\dot{\phi}^2\right)/2$

摆杆的势能：$V_{at}=m_1 g L_1\cos\theta$

飞轮的势能：$V_{a2}=m_2 g(L_1+L_2)\cos\theta$

系统的总势能：$V=m_1 g L_1\cos\theta+m_2 g(L_1+L_2)\cos\theta$

综合以上各式，令 $L_1 = L_2$，从而得到拉格朗日算子:

$$L = \left[m_1 L_1^2 + 4m_2 L_1^2 + I_1 + I_2\right] \dot{\theta}^2/2 + I_2 \dot{\phi}^2/2 - (m_1 + 2m_2) g L_1 \cos\theta$$

系统的广义力矩：$f = \left[-I_2 \ddot{\phi} \quad u - I_2 \ddot{\theta}\right]^T$，式中，$u$ 为飞轮的驱动力矩 (即城速器输出的力矩)。系统耗散力包括:

摆杆耗散力: $\tau_1 = -c_1\dot{\theta}$ 和飞轮耗散力: $\tau_2 = -c_2 \dot{\phi}$ 式中，c_1为摆杆绕转轴 O 转动的摩擦阻力矩系数；c_2飞轮绕转轴 O_2 转动的摩擦阻力矩系数。终得出:

$$(a + I_2)\ddot{\theta} + I_2 \ddot{\phi} = b\sin\theta - c_1 \dot{\theta}$$
$$I_2(\ddot{\theta} + \ddot{\phi}) = u - c_2 \dot{\phi}$$

式中，$a = m_1 L_1^2 + 4m_2 L_1^2 + I_1; b = (m_1 + 2m_2) g L_1$。

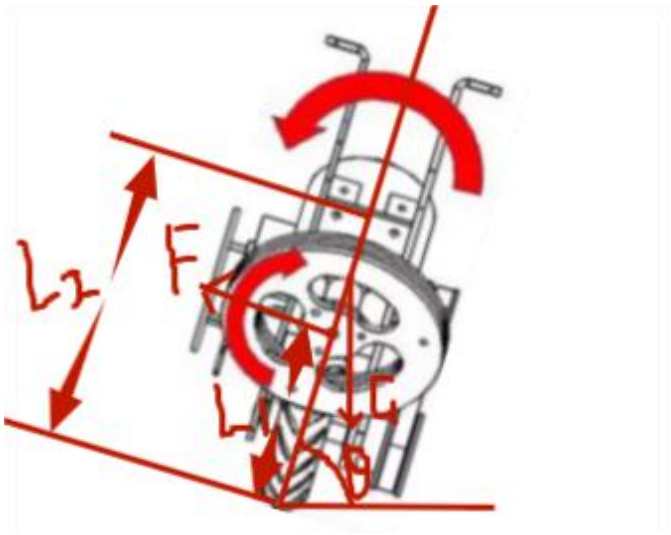
上式即为基于飞轮”单级倒立摆系统在忽略空气阻力的条件下的模型表达式。

5.3 平衡控制方案

在对独轮车的平衡控制方案的研究中，我们采用的最主要的控制方案是 PID 控制以及 LQR 控制，在动量轮的作用下，在静止状态下为了保持自行车的平衡，可以考虑如下几种方法：

5.3.1 并级 PID 控制

自行车在静止状态下姿态控制是通过负反馈来实现的。因自行车只有两个轮子着地，车体只会在轮子轴向方向上发生倾斜。假设控制动量轮转动时，在 每一个瞬时，提供一个冲量用于抵消重力的冲量，便可以保持车体平衡。



当自行车倾斜一点角度时，重力的作用会使自行车很快倒下，为了使自行车稳定在平衡位置，在车体加入动量轮，增加自行车额外的受力 F，使得恢复力 与倾斜角度相反。 控制动量轮，使得它做加速运动，这样，根据角动量守恒定律：$L = r * p$ 故角动量对时间的变化率为：

$$\frac{dL}{dt} = \frac{d(r * p)}{dx} = \frac{dr}{dt} * p + r * \frac{dp}{dt}$$

在上式中，右端第一项

$$\frac{dr}{dt} = v, p = m * v$$

因此矢量积：

$$\frac{dr}{dt} * \vec{p} = \vec{0}$$

由牛顿第二定律得,：

$$\frac{dp}{dt} = F$$

把上式改写为：

$$\frac{dL}{dt} = r * F$$

在 $0 \sim t1$ 时刻内应满足:

$$\Delta L_F = \int_0^{t1} L_1 * F_t dt \geq \int_0^{t1} L_2 * G dt$$

即对于瞬时而言：$L_1 * F_t > L_2 * G$

此外，为了使得自行车能够尽快地在垂直位置稳定下来，还需要增加阻尼

力。虽然存在着空气和摩擦力等阻尼力，相对阻尼力比较小。因此需要另外增加控制阻尼力。增加的阻尼力与翻滚角角速度成正比,方向相反。因此上式可变为：

$$F * L_1 = G * \cos\theta * L_2 - k_2 * \theta$$

通过假设简单的线性关系，可得控制动量轮速度的控制算法：

$$v = k_1 * \theta + k_2 * \theta'$$

但在实际运行时，车体保持静止平衡后会出现反复朝向一个方向倾斜并校正，会导致动 量轮转速不断上升，当转速达到驱动限制最高转速时，电机断电停转，车身失去平衡。为了 缓解这种问题出现，我们对期望角度进行修改。 设期望角度为θ ,机械平衡角度为θ_0 ,我们加入转速影响期望角度，使转速越 大，期望角度朝反向增大，则：$\theta = \theta_0 + k_1 * v * (\theta_{now} - \theta_0)$

则最终控制方案为:

$$v = k_1 * \left(\theta_0 + k_1 * v * (\theta_{now} - \theta_0)\right) + k_2 * \left(\theta_0 + k_1 * v * (\theta_{now} - \theta_0)\right)'$$

即通过 pid 控制:

$$\theta = \left(\theta_0 + k_1 * v * (\theta_{now} - \theta_0)\right) = 0$$

对于角度的 pid 控制，可以使用普通的 pid 函数来完成，也可以根据调试情 况对 pid 进行改进。 例如，对每次误差进行限幅，只有误差在某个范围内计为生效。

5.3.2 串级 PID 控制

使用串级 PID 进行控制，则需要将三个 PID 控制器进行串联，分别是角速度环、角度环和速度环，根据响应时间的快慢，将角速度环作为最内环，其次是角度环，速度环作为最外环，与直立车、平衡单车类似。

串级 PID 的代码实现比较简单，思路也比较容易实现，就是将三个普通的 PID 进行相应的串联，配合不同的控制周期实现串级 PID，只不过每一个环节的 PID 的误差输入的物理量都不一样，同时内环 PID 的输入要联系前一个环节 PID 的输出，相应的流程图如图 5.3 所示：

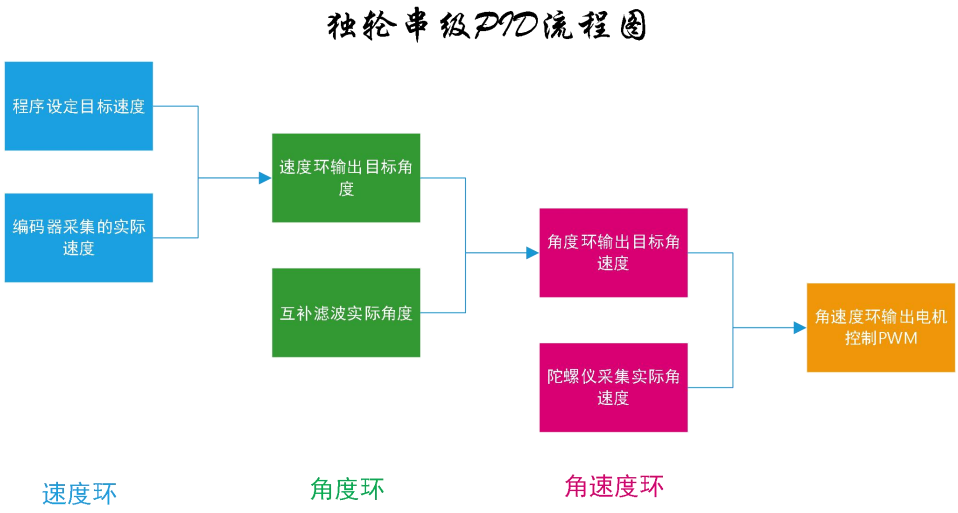


图 5.3 独轮车串级 PID 流程图

对于串级 PID 的参数调节，需要遵循相应的原则，首先要从内环开始往外环调节，依次是角速度环、角度环和速度环。

角速度环的调整是希望当车模朝一个方向发生倾倒时，控制动量轮的旋转产生一定的阻尼，抑制小车的角速度。

角度环的调整是希望小车根据角度的偏移而固定输出一个转速，抑制小车角度的倾斜。

速度环的调整是希望速度环是加快速度的增加，这样可以让电机输出更大的力矩以更快的达到平衡。

我们为了方便对独轮车的平衡参数进行调整，使用了 VOFA 上位机进行在线调参，避免了每次调整参数都需要进行按键的调参甚至是程序的烧录，使用上位机进行参数的整定可以方便的对相应曲线和相应的变量状态进行分析，以此来判断独轮车的平衡性能，上位机调参界面如图 5.4 所示：

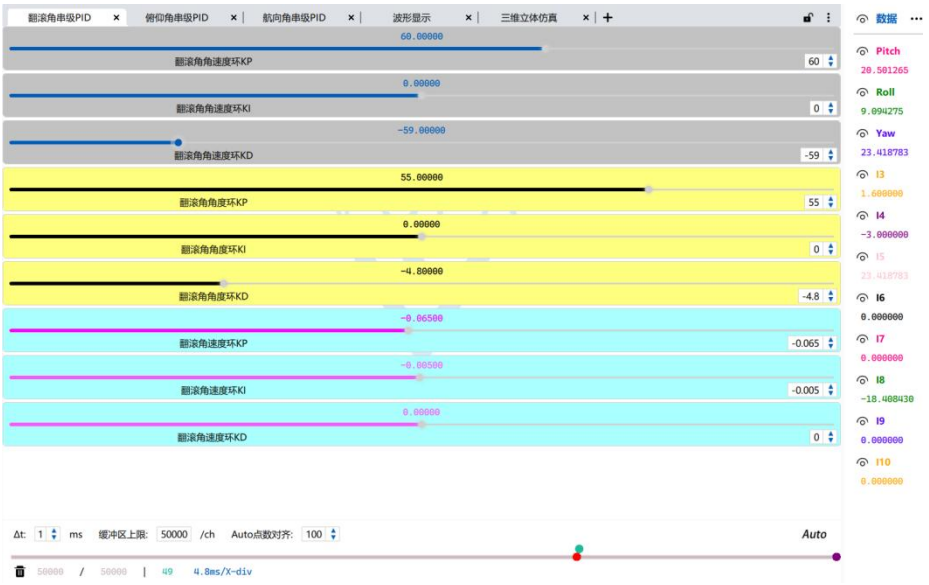


图 5.4 串级 PID 上位机调参界面

通过上位机我们可以方便的对各个环节的 PID 参数进行调节与可视化，可以实时的观察参数之间的关系，以此来进行分析和相应的调整。

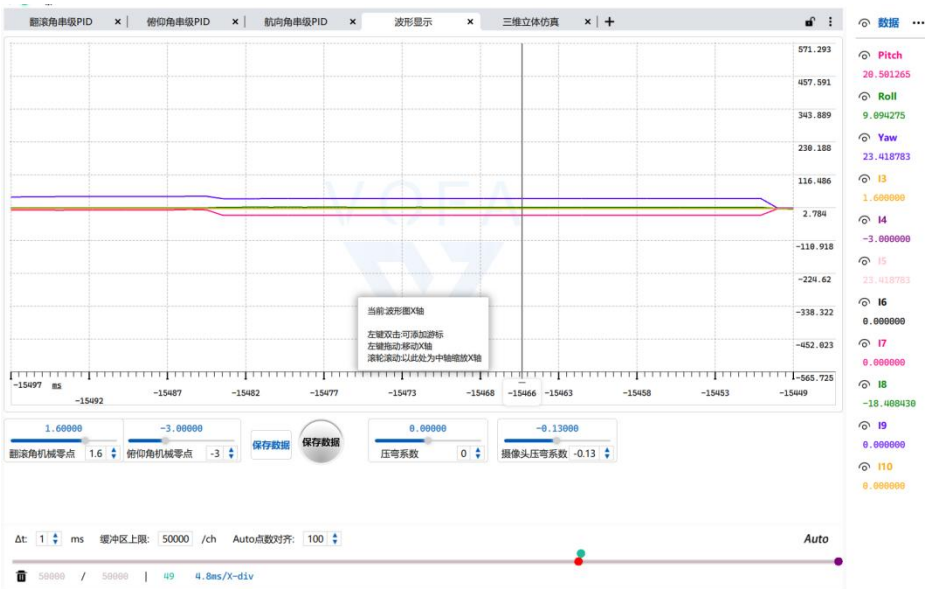


图 5.5 上位机波形图显示

通过对关心的变量进行波形图的显示，可以方便我们对控制过程中相应的控制量的变化进行分析，以此来判断独轮车的平衡性能，同时由于独轮车模的结构不同，重心位置不同，或者是陀螺仪安装的位置不同，会导致独轮车翻滚角和俯仰角的机械零点不同，因此需要对不同车模的机械零点进行微调，同时我们还设置了转向时压弯角度的调整，所有参数都设置完成之后就可以用上位

机进行数据的保存，此时车模收到保存数据的指令之后就会将当前调试完成的参数保存到 flash 当中掉电保存参数。

以上就是关于独轮车模翻滚角的平衡控制，对于俯仰角的平衡控制大体与之相同，但是由于独轮车模还需要进行前进，所以在直立的速度环上，需要在误差项加入你的期望速度，共同进行 PID 的闭环控制。

5.3.3 LQR 控制

根据 5.2 节独轮平衡原理中的介绍，将最终得到的公式进行泰勒级数的近似处理：

$$\begin{cases}(a+I_2)\ddot{\theta}+I_2\ddot{\phi}=b\theta-c_1\dot{\theta}\\ I_2(\ddot{\theta}+\ddot{\phi})=u-c_2\dot{\phi}\end{cases}$$

选定 $x=(x_1,x_2,x_3,x_4)^T=(\theta,\dot{\theta},\phi,\dot{\phi})^T$ 作为状态变量(倒立摆角度 θ 、角速度 $\dot{\theta}$ 飞轮角度 ϕ 、飞轮角速度 $\dot{\phi}$)，根据选定的状态变量做分离可以得到如下状态空间方程：

$$\begin{pmatrix}\dot{\theta}\\ \ddot{\theta}\\ \dot{\phi}\\ \ddot{\phi}\end{pmatrix}=\begin{pmatrix}0 & 1 & 0 & 0\\ \frac{b}{a} & -\frac{c_1}{a} & 0 & \frac{c_2}{a}\\ 0 & 0 & 0 & 1\\ -\frac{b}{a} & \frac{c_1}{a} & 0 & -\frac{(a+I_2)}{aI_2}c_2\end{pmatrix}\begin{pmatrix}\theta\\ \dot{\theta}\\ \phi\\ \dot{\phi}\end{pmatrix}+\begin{pmatrix}0\\ -\frac{1}{a}\\ 0\\ \frac{a+I_2}{aI_2}\end{pmatrix}u$$

由状态空间方程就得到了 LQR 中最重要的两个矩阵：

A 矩阵：

$$\begin{pmatrix}0 & 1 & 0 & 0\\ \frac{b}{a} & -\frac{c_1}{a} & 0 & \frac{c_2}{a}\\ 0 & 0 & 0 & 1\\ -\frac{b}{a} & \frac{c_1}{a} & 0 & -\frac{(a+I_2)}{aI_2}c_2\end{pmatrix}$$

B 矩阵：

$$\begin{pmatrix}0\\ -\frac{1}{a}\\ 0\\ \frac{a+I_2}{aI_2}\end{pmatrix}$$

由 LQR 的表达式：

$$J_{min} = \min\left\{\int_0^{\infty}(x^TQx + u^TRu)dt\right\}$$
$$= \int_0^{\infty}(x^TQx + x^TK^TRKx)dt$$
$$= \int_0^{\infty}x^T(Q + K^TRK)xdt$$

表达式中的 Q 矩阵决定你对应状态空间变量的收敛速度，R 矩阵决定输入的收敛速度，这两个是超参数，根据自己的需要进行相应的设置，因此求解该表达式中的 K 值是其核心问题，LQR 实际上提供了一个通过迭代的方法求解最优 K 的方法，matlab 也提供了直接求解 K 值的函数调用 K=lqr(A,B,Q,R)。

由此就得到了最终的 PWM 输出：

$$u=k*x \quad (x = (x_1, x_2, x_3, x_4)^T = (\theta, \dot{\theta}, \phi, \dot{\phi})^T)$$

相应的 C 语言代码如下：

```
1.  //LQR 倒立摆模型的相关参数
2.  #define M_PI 3.14159
3.  float adjust_ang= -10.16;
4.  float adjust_p_vel= -3.09;
5.  float adjust_motor_vel= -0.102;
6.  float xishu = 100;
7.  float adjust_total_ang= 0;   //微调角度
8.  float target_voltage;//输出
9.  // LQR 稳定控制器功能
10. // 计算需要设置电机的电压，以稳定摆
11. float controllerLQR(float p_angle, float p_vel, float m_vel){
12.   // 如果角度可控
13.   // 计算控制律
14.   // LQR controller u = k*x
15.   //  - k = [-10.1584, -3.0902, -0.1020]
16.   //  - x = [摆角, 摆速度, 电机速度]'
17.   float u = (adjust_ang*p_angle + adjust_p_vel*p_vel + adjust_motor_vel*m_vel)*xishu;
18.   //对输出进行限幅
19.   u =constrain_float(u, -9000.0, 9000.0);
20.   return u;
21. }
```

但是在调试过程中发现由于独轮车模的结构过于复杂，不能够精确的得到车模系统的重心、转动惯量和摩擦力矩系数等关键的参数，因此即便运动学模型得到成功的建立，但是平衡的效果仍然非常的不理想，对 LQR 的参数进行大

量的调整仍然得不到很好的稳定效果。

5.3 转向控制方案

转向控制使用双电机差速，同样使用串级 PID，在车模上产生一对等大反向的力，使得车模进行转向：

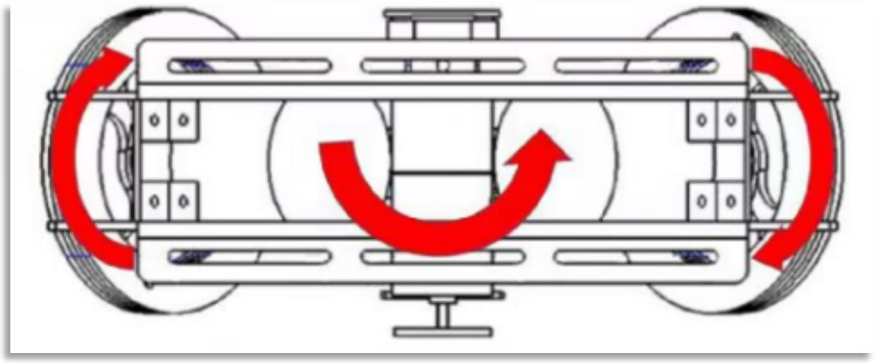


图 5.6 独轮转向示意图

由此可以看出来独轮转向的思路特别的简单，只需要将摄像头采集到的赛道的误差转换成对应的差速值即可像三轮这样的差速结构的小车一样完成转向控制，但是实际控制更加复杂，动量轮并不是依靠转速来提供相应的力矩完成转向的任务，而是需要动量轮进行加速运动来提供转向所需的力矩，加速度越大，力矩越大，转向越快，但是动量轮的转速是有上限的，因此不能时时刻刻持续的进行加速运动，导致独轮不能得到持续力矩来支持你连续同方向的转向，因此会出现转向饱和的问题，这里的饱和不是我们常见的转向环 PID 的积分饱和现象，而是指动量轮的转速已经达到了上限，无法再加速了，导致独轮车模无力转向，同时动量轮的高速旋转也让独轮车的平衡很难再继续得到保障，导致独轮车模失控倒地。

解决独轮转向饱和的一个重要的方法就是进行独轮车模的压弯处理，和许多单车转向的原理类似，通过动态的改变独轮车模的机械零点，让车模系统自身的重力的分量能够在转向的方向上提供一个向心力，辅助独轮车模进行转向，此时就能一定程度上缓解独轮车持续差速转向导致动量轮转速饱和的问题。

同时独轮压弯的思想与单车有着异曲同工之妙，但是在压弯的实现上仍然存在一定的区别，我们使用的方法是将车模行进轮的速度与当前赛道的误差结

合起来综合考虑小车的压弯角度，即：压弯角度=压弯系数*当前行进速度*赛道偏差，将此压弯角度叠加到翻滚角的机械零点上就能完成压弯转向。

5.4 滤波与姿态解算

5.4.1 巴特沃斯滤波

我们独轮车系统使用的 IMU 模块是基于 icm20602 的六轴陀螺仪，含陀螺仪和加速度计，在进行陀螺仪数据的获取过程中，通过对原始数据进行观察发现原始数据存在断续的变化，不够连续，导致解算得到的姿态角不够稳定，因此我们选择了巴特沃斯滤波器对陀螺仪采集到的原始数据进行相应的滤波处理，使得得到的数据更加的丝滑。

首先我们观察陀螺仪采集得到的原始数据以及对该原始数据进行 FFT 频域分析的波形图如下：

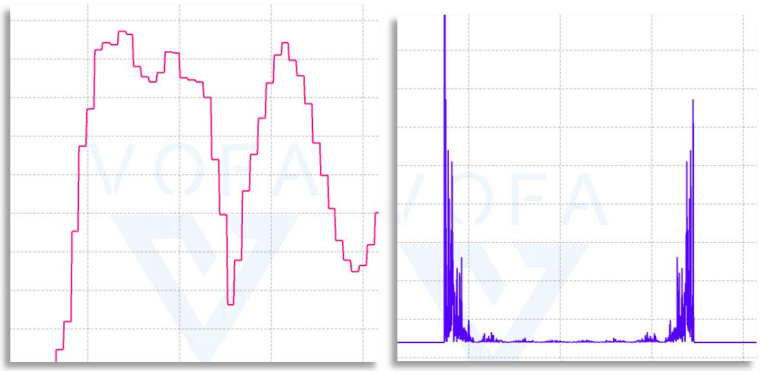


图 5.7 原始数据及其频域波形图

下面将使用巴特沃斯滤波得到的原始数据和 FFT 得到的频域数据叠加到原始数据上的波形图绘制如下：

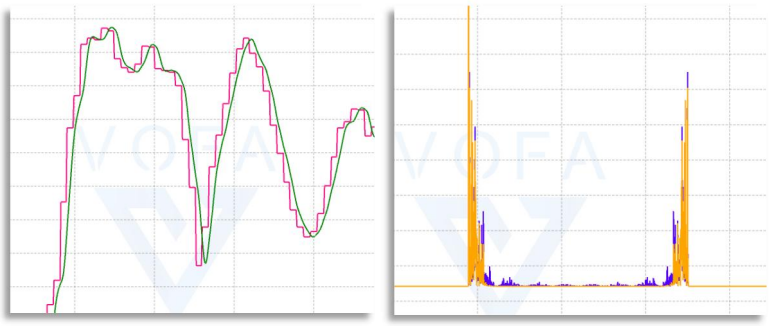


图 5.8 原始数据及滤波数据叠加的波形图

由此可以看出来，滤波之后的数据比原始数据更加的连续丝滑，且分析频域波形图可以发现，原始数据的高频成分基本上被滤除掉了，使得原始数据拥有更好的质量，为后期进行姿态解算奠定了基础。

5.4.2 姿态解算

对独轮车三个欧拉角的解算，我们使用了四元数，不同于常规的姿态解算方法，四元数解决了欧拉角存在万向死锁的问题，同时在四元数中对陀螺仪采集并进行巴特沃斯滤波得到数据进行翻滚角和俯仰角的解算时，结合了加速度计进行一阶互补滤波，得到了更加准确的角度，使得在平衡和转向的控制过程中独轮车系统有更加稳定的性能。

5.4 动量轮总体控制方案

得到平衡控制的 PWM 输出与转向控制的 PWM 输出之后，我们的控制思路就是简单的将两个 PWM 进行加和，得到最终左右动量轮的 PWM 输出，总体方案控制方框图如图 5.9 所示：

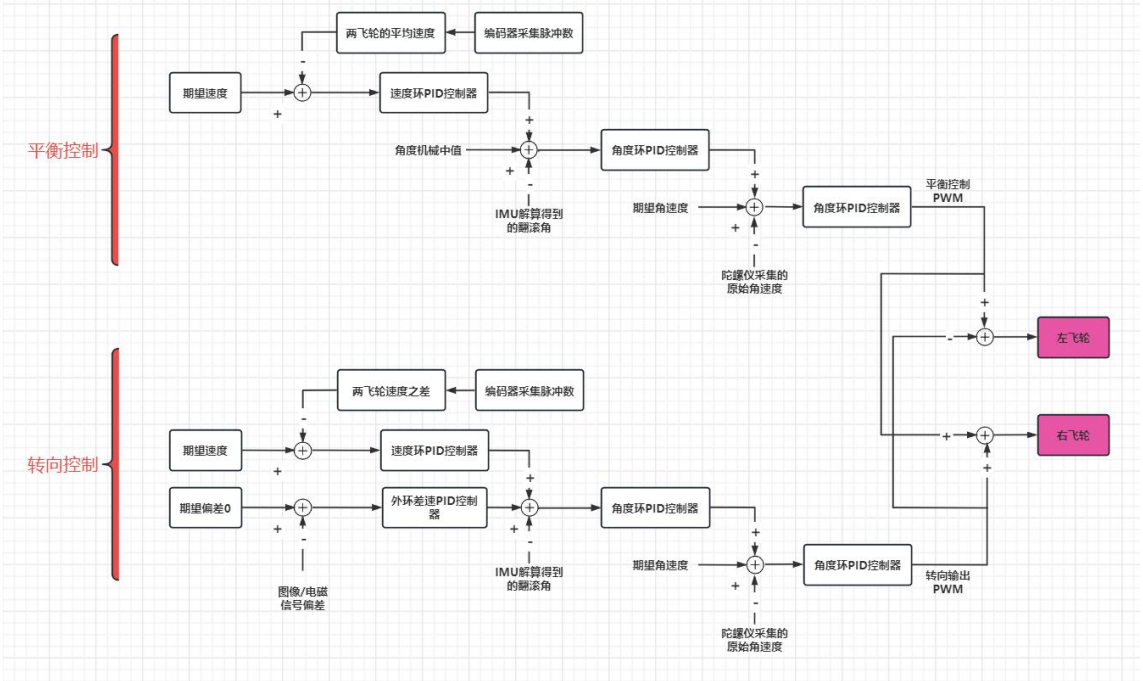


图 5.9 平衡和转向控制方框图

第六章 摄像头算法分析

6.1 软件设计流程

在实现小车运行功能时最主要在于图像的采集以及处理，首先我们经过 DMA 采集到一帧图像并将该图像进行二值化，通过对二值化的图像进行处理使处理会比直接处理原始图像速度更快，进而对二值化图像进行边线的提取以及各种特殊元素的判断与处理从而获得整个赛道的中线值，进行中线的滤波并算得其偏差值从而驱使小车进行转向闭环控制。具体流程图如下图 6.1

图 6.1 软件设计流程图

6.2 大津法(OSTU)二值化

大津法又名最大类间差方法，是日本学者大津于 1979 年提出的一种自适应阈值分割法。它通过统计整个图像的直方图特性，使得阈值能将图像分为相差最大的两类。

程序优化了遍历灰度值的循环，剪掉了灰度值小于 40 和大于 220 的部分，对于中间部分采用隔一行统计一行的办法降低复杂度。对于一灰度图像，其类间方差为山峰状，当搜索到峰值时，不必再向下继续遍历，进一步缩小时间开销。

图 6.2 二值化图像

6.3 赛道边界提取

摄像头处理采用八邻域算法。八邻域为迭代算法，迭代算法分三步，确定起始条件，迭代过程，退出条件。首先在底行左右搜索确定边界，之后进入迭代过程，对于当前迭代点而言，根据上两个点确定起始搜索方向，按照一定顺序，顺时针或逆时针遍历当前点的八个邻域，若判断到黑白跳变，则将该点作为新迭代点，并进入下一次迭代过程。算法的退出条件也很重要，需要在调试中慢慢确定。

图 6.3 边线提取补线图

6.4 元素识别

6.4.1 环岛

环岛最基础的特征主要是一边与长直道相同，另一边斜率变化会比较大且产生较多丢边行，第二阶段要环岛的黑环进行匹配，我们这里采用的是斜率匹配法。在准确匹配到黑环之后，便可根据黑环位置补出期望路径的中线位置，对所得中线矫正滤波之后，便可得到比较理想的入环路径。第三阶段为入环，找到上端拐点与对角连接进行补线。第四阶段环内正常寻迹。第五阶段为出环，一边边界会出现斜率突变的现象，这时就需要开始执行出环。第六阶段为出环末，寻找边界最值与对角连接进行补线。

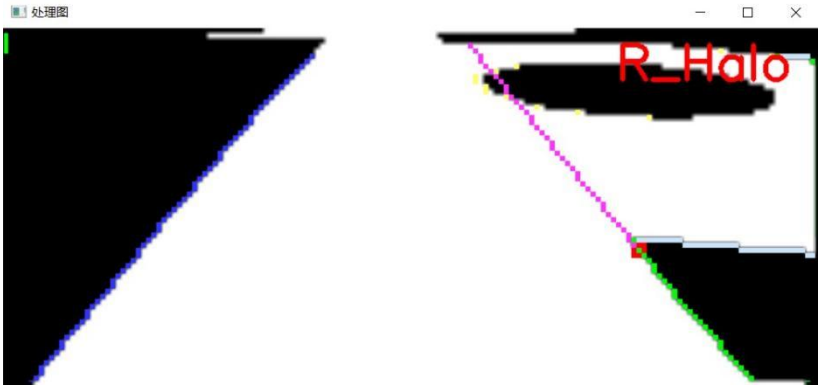


图 6.4 环岛

6.4.2 十字

十字的判断较为简单，判断到两个下拐点且判断为十字拐点即为十字。补线可以两点拉线，未判断到上拐点则从下往上拉线，下拐点消失则从上往下拉线。

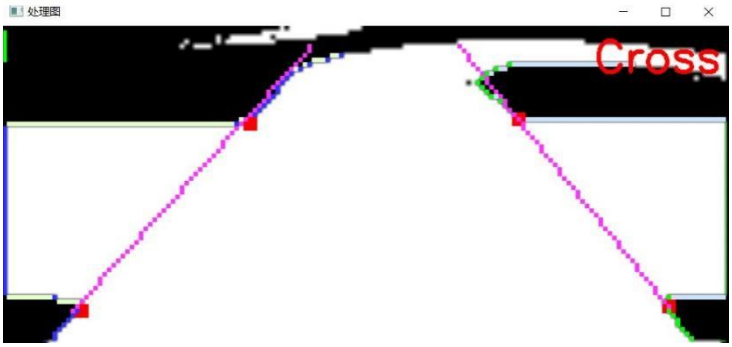


图 6.5 十字

6.4.3 坡道

坡道根据斜率进行判断，正常赛道无斜率跳变，但坡道图像前端向上突起，产生一个明显的斜率增加，当上下两段2斜率满足条件且二者之差大于阈值，则判断为坡道。

图 6.6 坡道

6.4.4 车库

识别车库的最主要任务就是寻找到车库旁边的斑马线，一旦某行黑白跳变点的数量大于设定阈值，即认为识别到了斑马线，根据计数到的斑马线个数选择补线路过或者入库。车库易与环岛产生误判，因为二者都满足斜率和拐点条件，处理的方法是一旦判断到斑马线，就进入车库状态，不再判断环岛；若判断为环岛，则继续进行车库判断，无斑马线则确定为环岛，有斑马线则修正为车库。

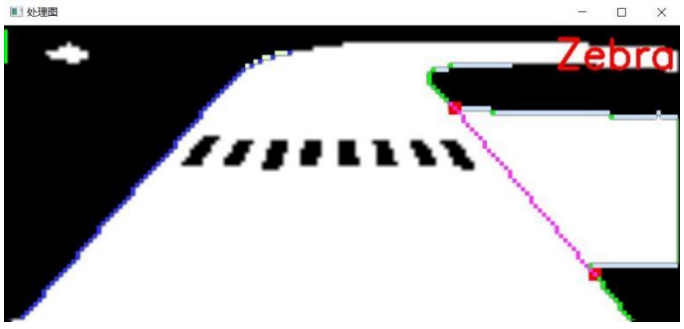


图 6.7 车库

6.4.5 断路/路障

对于图像本身来说，断路和路障的图像长相类似，都是远端产生一片黑，因此一同进行判断，记忆赛道或用红外的数据进行区分。若左右边界相交且红外无数据，继续利用黑拐点区分正入和斜入断路，若红外有数据，直接判定为路障。

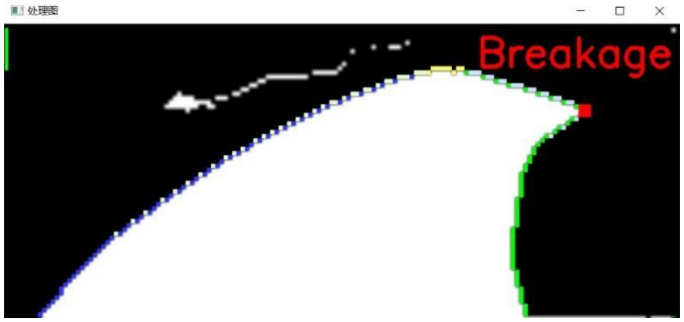


图 6.8 断路

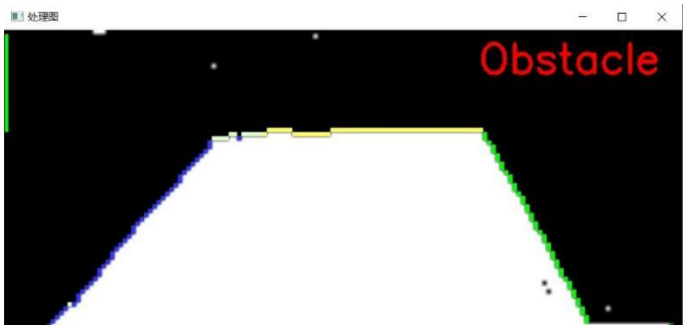


图 6.9 路障

6.5 边缘检测与直接灰度

18届恢复为线下比赛，大棚里的瓜现在要拉出来照阳光了，对图像处理算法的要求有所提高，因此尝试了几个抗光算法。

6.5.1 边缘检测

边缘检测可以检出边缘，而八邻域的目的是搜索边缘，二者可谓天造地设的一对。下面三个图可以直观看出边缘检测相比于二值化的优越性。

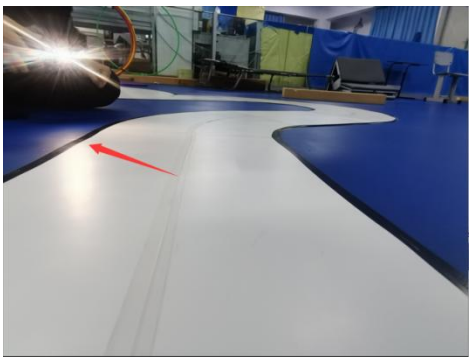


图 6.10 原图 手机拍摄

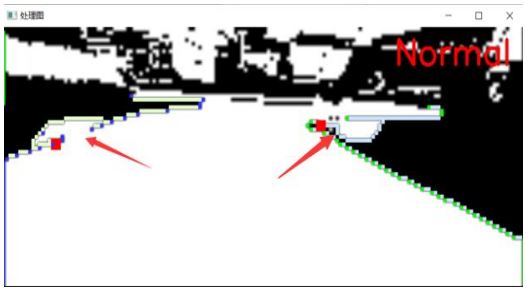


图 6.11 基于二值化

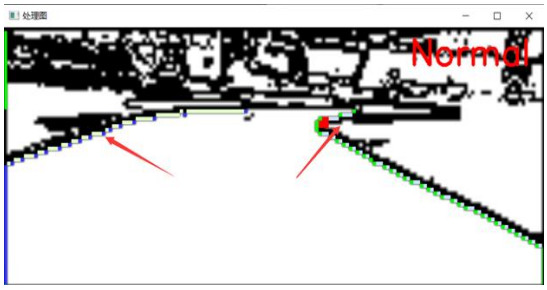


图 6.12 基于边缘检测

6.5.2 直接灰度

不管是直接二值化还是边缘二值化，都有将0~255赋值为0或1的操作，都属于有损。直接处理灰度图理论上是最优最能抗光的。

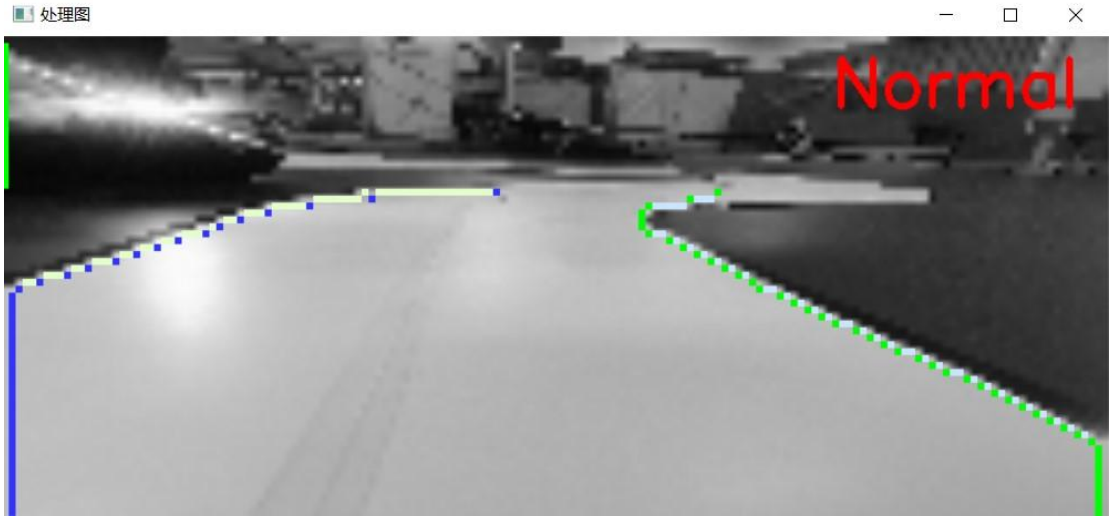


图 6.13 直接灰度

第七章 系统开发环境及调试

7.1 软件开发环境

程序开发在AURIX Development Studio（ADS）下进行，ADS是AURIX™开发工作室是一个免费的集成开发环境（IDE），专门用于开发AURIX微控制器的应用程序。

7.2 人机交互界面设计

应第十八届全国大学生智能汽车竞赛的要求，比赛过程中禁止使用电脑进行对车模进行程序的烧录，因此需要进行相应的人机交互界面的设计以满足一些关键参数离线进行调试，避免了由于赛道未知而导致程序过于固定存在的鲁棒性不足的问题，因此在人机交互界面的设计上，我们尽可能的将所需要调节的参数都写到了菜单中，通过按键、旋转编码器和屏幕来方便的对相关的参数变量进行显示与调试，具体菜单如下图所示：

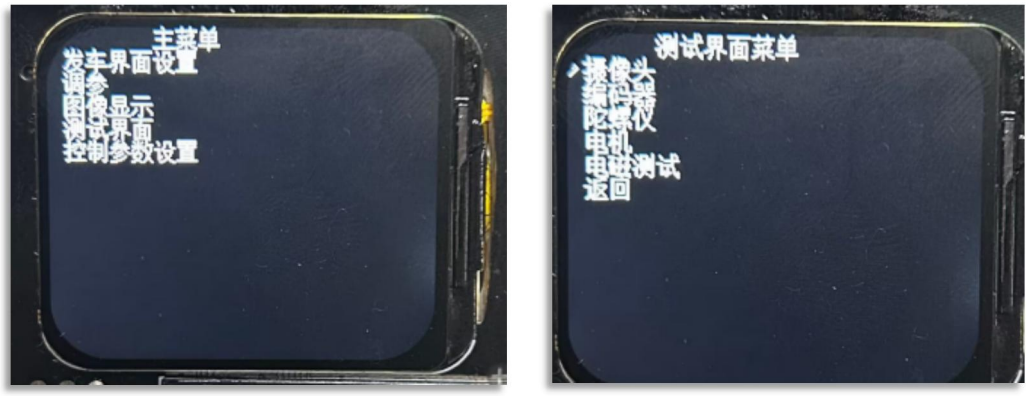


图 7.1 人机交互界面

7.2 手机蓝牙控制以及 PC 上位机调试

7.2.1 蓝牙控制

蓝牙控制采用了一款手机蓝牙调试软件，硬件使用了 HC-05 蓝牙模块，独轮车模通过主控板上插接的蓝牙模块与手机端蓝牙调试器进行通信，手机端蓝牙调试器发送相应的指令，独轮车通过编写的蓝牙包解析代码进行指令的解析，

实现了手机端参数的调试与保存，手机端通过波形显示独轮车的运行状态以及关键变量，具体如下图：

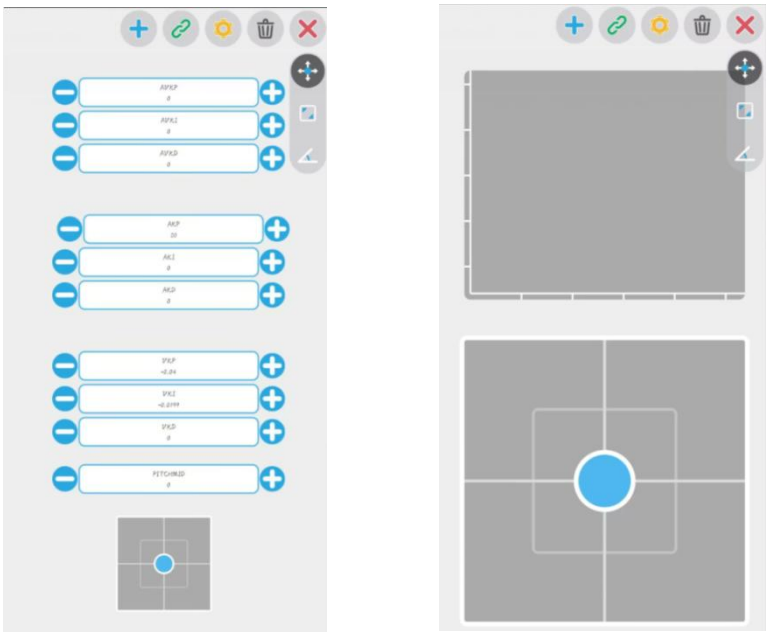


图 7.2 手机端蓝牙调试界面

7.3.2 上位机调试

对独轮车，电脑上位机我们主要 VOFA 上位机调试助手对我们的程序进行调试，比如参数的设置、相关变量的显示、独轮车当前状态的 3D 图显示以及相关变量的保存等。

参考文献

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

[2]徐世豪.智能车设计中增量式PID控制算法的运用[J].中国新通信,2017.

[3]王淑娟，蔡惟铮，齐明．模拟电子技术基础 [M]．北京：高等教育出版社．2009

[4]蔡述庭．“飞思卡尔”杯智能汽车竞赛设计与实践 [M]．北京：北京航空航天大学出版社．2012.

[5]俞道阳．“恩智浦”杯中基于视觉巡线的四轮小车控制系统设计[D].南昌大学,2021.

[6]郑亚利,古训.基于CMOS摄像头循迹的智能车控制系统设计[J].贵阳学院学报(自然科学版),2019.

[7]吴杰,王小妮,刘鹏等.智能小车蓝牙通信模块设计与实现[J].北京信息科技大学学报(自然科学版),2019,34(06):64-69+75.DOI:10.16508/j.cnki.11-5866/n.2019.06.012.

[8]张坤．基于ARM单片机的智能小车循迹避障研究设计[D].齐鲁工业大学,2020.DOI:10.27278/d.cnki.gsdqc.2020.000415.

附录 A 车模技术检查表

队伍名称	山魂三队			
参赛学校	山东大学（威海）			
赛题组组别	独轮平衡组			
检查项目	规格 （选手自行填写）	符合 （√）	不符合 （×）	备注
1.车模类型是什么？	0 车模			如果是自制车模，请标明自制。
车模整体尺寸： 1.（包括传感器在内）长，宽，高(mm) 2. 摄像头组标明镜头距离地面高度。	长 255mm，宽 250mm，高 420mm			在填写是，请将所在组别规则对于车模尺寸限制同时进行填写。
1. 传感器种类、规格(型号)数量。 2. 是否使用自带MCU的成品传感器模块？型号是什么？	1. 逐飞总钻风摄像头、逐飞 ICM20602 陀螺仪、编码器、红外测距 2. 否			
1. 控制转向舵机型号是否自行改装舵机？ 2. 防伪易损标签是否完整？	1. 否 2. 是			
1. 是否增加伺服电机？ 2. 如果有那么种类、个数和作用？	否			
1. 电路中微处理器型号和个数？	TC264DA，1 个			
1. 是否具有其它可编程	无			

器件，个数与作用？				
1. 是否有无线通讯装置？ 2. 如果有，那么种类和个数？	无			
1. 电池的种类、规格和数量？	一个 6S 电池组（2 个 3S，1300mah 串联）			
1. 是否使用 GPS 导航？ 2. 是否没有使用 RTK？	1.否 2.否			
1. 后轮驱动电机是否是原车模电机？ 2. 是否具有防伪易损标签？	1. 是 2. 是			
1. 车模轮胎是否原有的纹理可辨析？ 2. 轮胎表面是否具有粘性物质？ 3. 对于麦克纳姆轮是否更换过小轮胶皮？	1. 是 2. 否			
1. 车模底盘是否是原车模底盘？ 2. 是否有大面积切割？	1. 是 2. 否			
1. 车轮轴距、轮距是否改装？ 2. 改装参数是什么？	否			
1. 车模驱动轮传动机构是否改装？ 2. 改装方式是什么？	否			

附录 A 车模技术检查表

1. 车模差速器是否改装？ 2. 改装方式是什么？	否			
1. 是否更换过原装车模中的机械元器件？更换后的规格 是什么？	否			
1. 车模电路板个数及功能。 2. 其中是否有购买成品、哪一些？	1.6 个，电磁前瞻，运放，主板，电源及单路驱动板，屏幕，陀螺仪转接板 2.否			
1. 自制电路板是否标记有学校名称、队伍名称、制作日期等信息？ 2. 标示信息在PCB的哪一层？	1. 是，山东大学（威海），山魂三队，2023 2. 敷铜层			请在表格中注明电路板队伍信息的内容。
其它待说明内容	无			
检查人员签名：	检查意见：			

附录 B 部分源代码

```
#include "Cpu0_Main.h"
#include "zf_common_headfile.h"
//#include "common.h"
#include "Init.h"
/*
 * Init.c
 *
 *  Created on: 2021 年 6 月 6 日
 *      Author: Asura丶
 */
int8 init_finish = 0;
void SYSTEM_INT(void)
{
    if(init_finish == 0)
    {    //按键初始化
//        key_init();
        //初始化蜂鸣器
        ips200_showstr(0 ,0 ,"SYSTEM init1");
        gpio_init(P33_9, GPO, 0, GPO_PUSH_PULL);
        gpio_init(P20_7, GPO, 1, GPO_PUSH_PULL);
        gpio_init(P20_6, GPO, 1, GPO_PUSH_PULL);
        ips200_showstr(0 ,0 ,"SYSTEM init2");
//        gpio_init(P20_2, GPO, 1, GPO_PUSH_PULL);//刹车引脚
        //红外 ADC 采集初始化
        adc_init(ADC2_CH3_A35, ADC_12BIT);
        ips200_showstr(0 ,0 ,"SYSTEM init3");
        //编码器初始化
        gpt12_init(GPT12_T4, GPT12_T4INA_P02_8, GPT12_T4EUDA_P00_9);//编码器初始化
        gpt12_init(GPT12_T2, GPT12_T2INB_P33_7, GPT12_T2EUDB_P33_6);//编码器初始化
        gpt12_init(GPT12_T5, GPT12_T5INB_P10_3, GPT12_T5EUDB_P10_1);//独轮动量轮编码器 1
        gpt12_init(GPT12_T6, GPT12_T6INA_P20_3, GPT12_T6EUDA_P20_0);//独轮动量轮编码器 2
        ips200_showstr(0 ,0 ,"SYSTEM init4");
        //2.0 寸并口屏幕初始化
//        ips200_init(IPS200_TYPE_SPI);
        ips114_init();
        ips200_showstr(0 ,0 ,"SYSTEM init5");
        //相关参数初始化
        setSpeed = 0;
```

```
39.          shacheflag=0;
40.
41.          //电机初始化
42.          motor_init();
43.          motor_init1();//无刷电机初始化
44.          ips200_showstr(0 ,0 ,"SYSTEM init6");
45.          //串口初始化，用于无线模块数据传输给上位机
46.          wireless_uart_init();
47.          ips200_showstr(0 ,0 ,"SYSTEM init7");
48.          //蓝牙串口初始化
49.  //        uart_init(UART_0, 9600, UART0_TX_P14_0, UART0_RX_P14_1);
50.
51.          //ips200_showstr(0 ,0 ,"Test start2...");//开始标志
52.          //摄像头初始化
53.          mt9v03x_init();
54.          ips200_showstr(0 ,0 ,"SYSTEM init8");
55.  //*********************陀螺仪初始化**********************//
56.          //陀螺仪初始
57.          ips200_showstr(0 ,0 ,"ICM20602 init");
58.
59.  #if IMU660RA_DEVICE == 1
60.      imu660ra_init();
61.  //    imu660ra_Offset();
62.  #else
63.      icm20602_init();
64.      system_delay_ms(100);
65.  //    icm20602_Offset();//传感器采集零漂
66.  #endif
67.      //获取陀螺仪的零漂
68.          GYRO_Offset.X = flash_read(1, 1, float);GYRO_Offset.Y = flash_read(1, 2, float);GYRO_Offse
    t.Z = flash_read(1, 3, float);
69.  //*********************陀螺仪初始化**********************//
70.
71.
72.          ips200_showstr(0 ,0 ,"ICM20602 init success");
73.
74.          //电感采集 ADC 初始化
75.          adc_init(ADC1_CH0_A16, ADC_10BIT);
76.          adc_init(ADC1_CH1_A17, ADC_10BIT);
77.          adc_init(ADC1_CH4_A20, ADC_10BIT);
```

```
78.         adc_init(ADC0_CH10_A10, ADC_10BIT);
79.         adc_init(ADC0_CH11_A11, ADC_10BIT);
80.         adc_init(ADC0_CH12_A12, ADC_10BIT);
81.         adc_init(ADC0_CH13_A13, ADC_10BIT);
82.         adc_init(ADC1_CH5_A21, ADC_10BIT);
83.
84.         adc_init(ADC2_CH4_A36, ADC_12BIT);//6S 电池电压采集初始化
85.         adc_init(ADC1_CH9_A25, ADC_12BIT);//红外测距
86.
87.         //初始化定时器,获取编码器值，用于计算占空比 ImageStatus.Det_True
88.         pit_init(CCU60_CH0, 1000);//单位是 us,1 毫秒的定时器中断
89.
90.         //给所有误差值置零
91.         Err = 0;
92.         ImageStatus.err = 0;
93.         ImageStatus.yerr = 0;
94.         ImageStatus.Det_True = MT9V03X_W*0.5;;
95.
96.     #if 0    //串级 PID 相关参数初始化
97.         //翻滚角相关参数初始化
98.         Cascade_Ang_Vel.kp = 200;
99.         Cascade_Ang_Vel.ki = 210;
100.        Cascade_Ang.kp = 20;
101.        Cascade_Ang.kd = -0.6;
102.        Cascade_Vel.kp = 0.05;
103.        Cascade_Vel.ki = 0.006;
104.        Cascade_Vel.kd = -0.3;
105.        //航向角串级 PID 参数初始化
106.        Cascade_Ang_Vel_Y.kp = 38;
107.        Cascade_Ang_Vel_Y.ki = 30;
108.        Cascade_Ang_Y.kp = 12;
109.        Cascade_Ang_Y.kd = -5;
110.        Cascade_Vel_Y.kp = -0.03;
111.        Cascade_Vel_Y.ki = 0.08;
112.        Cascade_Vel_Y.kd = -1;
113.        //定义串级 PID 相关控制变量以及初始化
114.        char wushua_flag=0;
115.        Roll_Mid = -0.5;
116.    #endif
117.    #if 0      // 旧车 翻滚角串级 PID 参数
```

```
118.                Cascade_Ang_Vel.kp = 200;
119.                Cascade_Ang_Vel.ki = 210;
120.                Cascade_Ang.kp = 20;
121.                Cascade_Ang.kd = -0.6;
122.                Cascade_Vel.kp = 0.05;
123.                Cascade_Vel.ki = 0.009;
124.                Cascade_Vel.kd = -0.3;
125.
126.                // 旧车 航向角串级 PID 参数
127.    //            Cascade_Ang_Vel_Y.kp = 20;
128.    //            Cascade_Ang_Vel_Y.ki = 2;
129.    //            Cascade_Ang_Y.kp = 12;
130.    //            Cascade_Ang_Y.kd = 0;
131.    //            Cascade_Vel_Y.kp = -0.18;
132.    //            Cascade_Vel_Y.ki = 0.1;
133.    //            Cascade_Vel_Y.kd = 0;
134.                Cascade_Ang_Vel_Y.kp = 38;
135.                Cascade_Ang_Vel_Y.ki = 30;
136.                Cascade_Ang_Y.kp = 12;
137.                Cascade_Ang_Y.kd = -5;
138.                Cascade_Vel_Y.kp = -0.03;
139.                Cascade_Vel_Y.ki = 0.08;
140.                Cascade_Vel_Y.kd = -1;
141.
142.
143.                // 旧车 俯仰角角串级 PID 参数
144.                Cascade_Ang_Vel_P.kp = -13;
145.                Cascade_Ang_Vel_P.ki = -1;
146.                Cascade_Ang_Vel_P.kd = -0.2;
147.                Cascade_Ang_P.kp = -30;
148.                Cascade_Ang_P.kd = 0.4;
149.                Cascade_Vel_P.kp = 7;
150.                Cascade_Vel_P.ki = 0.08 ;
151.                Cascade_Vel_P.kd = -6 ;
152.
153.                //定义串级 PID 相关控制变量以及初始化
154.                char wushua_flag=0;
155.                Roll_Mid = -0.5;
156.                Pitch_Mid = 4    ;
157.    #endif
```

```
158.                Roll_Mid = -0.5;
159.                Pitch_Mid = -6;
160.
161.                init_finish = 1;//相关外设初始化完成
162.    }
163. }
```