辽宁石油化工大学毕业设计（论文）

Graduation Project (Thesis) for Undergraduate of LNPU

题　　目	捡球机器人控制系统设计
TITLE	Control system design of ball picking robot

学　　院	创新创业学院
School	Innovation and Entrepreneurship Academy
专业班级	实验 1806
Major**&Class**	Experimental class 1806
姓　　名	张宇航
Name	Yuhang Zhang
指导教师	王越
Supervisor	Yue Wang

2022 年 5 月 23 日

论文独创性声明

本人所呈交的论文，是在指导教师指导下，独立进行研究和开发工作所取得的成果。除文中已特别加以注明引用的内容外，论文中不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的工作做出重要贡献的个人和集体，均已在文中以明确方式标明并致谢。本声明的法律结果由本人承担。

特此声明。

论文作者（签名）：张宇航

2022 年 5 月 23 日

摘　要

随着乒乓球日益普及，越来越多的人成为乒乓球爱好者。但在乒乓球运动普及过程中，也出现了一个不容忽视的问题——捡球。特别是对职业乒乓球手来说，一天击发乒乓球上千次，捡球任务之繁重可想而知。本论文针对这一问题，结合机器视觉技术研发一款能自动捡球的服务机器人，该机器人能够胜任捡球任务。此外，根据已有的服务机器人市场调查报告，服务机器人市场仍是一片蓝海，因此该服务机器人市场应用前景广阔。

为提高捡球小车系统的自动化程度和可靠性，把捡球小车系统划分为硬件部分和软件部分。其中，硬件部分主要包括车体结构，树莓派 4B 控制模块，电机驱动模块，传感器检测模块，云台舵机模块和摄像头模块。利用 SolidWorks 对捡球小车进行 3D 建模。可以直观地看出捡球小车的总体硬件结构，最后在 Proteus 软件里搭建捡球小车硬件电路；软件部分则是通过对捡球小车需实现的功能以及工作原理进行分析，确定了捡球小车控制方案。捡球小车采用 Python 作为开发语言，搭配开源 OpenCV 视觉库，完成了乒乓球识别、避障、换点搜索和串口信息发送等模块的编程与调试。在乒乓球识别程序的编写上实现了创新，具有自己的独创性，能够突破现有 OpenCV 提供方法的局限性。捡球小车操作模式设计了自动模式和手动模式，并且针对手动模式设计了手机端 APP 进行操控。

在完成上述研究内容后，将所搭建好的硬件电路与软件相结合，制作出了捡球机器人实物，并进行试验验证。结果表明：该系统运行稳定可靠，能够满足实际需求，捡球小车和手机端 APP 图传时延较低，具有一定的推广价值。

关键词：捡球机器人；树莓派；OpenCV；目标识别

Abstract

With the increasing popularity of table tennis, more and more people become table tennis lovers. However, in the process of the popularization of table tennis, there is also a problem that can not be ignored - picking up the ball. Especially for professional table tennis players, it is conceivable that the task of picking up the ball is heavy when they hit and serve table tennis thousands of times a day. Aiming at this problem, combined with machine vision technology, this thesis develops a service robot that can automatically pick up the ball. The robot can be competent for the task of picking up the ball. In addition, according to the existing service robot market survey report, the service robot market is still a blue ocean, so the application prospect of the service robot market is broad.

In order to improve the automation and reliability of the ball picking car system, the ball picking car system is divided into hardware part and software part. Among them, the hardware part mainly includes body structure, raspberry pie 4B control module, motor drive module, sensor detection module, pan tilt steering gear module and camera module. 3D modeling of the ball picking car is carried out by SolidWorks. The overall hardware structure of the ball picking car can be seen intuitively. Finally, the hardware circuit of the ball picking car is built in Proteus Software; The software part analyzes the functions and working principle of the ball picking car, and determines the control scheme of the ball picking car. Using Python as the development language and opencv visual library, the ball picking car has completed the programming and debugging of table tennis recognition, obstacle avoidance, point change search and serial port information transmission. It realizes innovation in the compilation of table tennis recognition program, has its own originality, and can break through the limitations of existing opencv methods. The automatic mode and manual mode are designed for the operation mode of the ball picking trolley, and the mobile app is designed for the manual mode to operate.

After completing the above research contents, the built hardware circuit and software are combined to make the real object of the ball picking robot, which is verified by experiments. The results show that the system is stable and reliable, can meet the actual needs, and the image transmission delay of the ball picking car and the app on the mobile phone is low, which has a certain popularization value.

Key words: Ball pickup robot; Raspberry pie; OpenCV; Target recognition

目录

1 绪论

1.1 论文研究背景和意义

近年来“健康生活”的观念越来越深入人心，越来越多的人走出房间参加文体活动。我国国球乒乓球非常受欢迎，尤其是受到中国乒乓球队多次在国际大赛中获得冠军，为国家争得荣誉的积极影响，乒乓球成为很多人的首选项目。近年来，随着芯片算力的不断提升，人工智能发展十分迅速，并且已经在很多领域实现了深度融合，所以我们有理由相信针对乒乓球训练开发的智能乒乓球训练场将会代替传统乒乓球馆成为未来的一大趋势。因此，在这种大背景下，如何利用好这一巨大机遇来推动我国乒乓球运动水平的提高就显得尤为重要。据了解，目前全球已有超过 100 家公司研发出了相关产品和系统，其中大部分来自日本、德国等发达国家，而我国的相关企业也开始崭露头角，并取得了不俗的成绩。不仅如此，根据前瞻产业研究院发布的《2022-2027 年中国服务机器人行业市场前瞻与投资战略规划分析报告》（以下简称《分析报告》）数据显示:从中国市场来看，服务机器人产业是我国机器人产业增长最快的领域之一。同时《分析报告》显示，2014-2019 年中国服务机器人市场规模持续扩大，2019 年上升至 22 亿美元，较上年同期上升 33.1%。智能乒乓球发球机如"OMNI 型智能乒乓球发球机"已出现于市场上，可由 APP 控制，使用者可选择上下旋，长短球发球方式或根据使用者的要求任意自定义发球方式。与以往相比，用户能在同一时间内能接发更多乒乓球，球技也会有所提高。但是这样大量的接发球会给使用者带来繁重的捡球工作，如果完全的依靠人力，肯定会降低使用者的训练效率。虽然目前市场上存在着许多人工捡球的设备，但是这些设备需要耗费大量的人力物力进行手动操作，而且效率低下，无法满足人们日益增长的需求。此外，由于场地的限制，捡球的质量并没有得到良好的保障。因此设计一种能够针对乒乓球捡球工作的服务机器人并且应用到训练中，不仅可以帮助使用者减少大量的捡球时间，还可以提高使用者的训练效率。图 1-1 展示的是 OMNI 发球机。

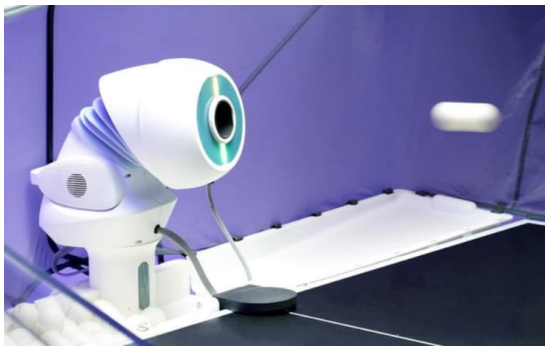

图 1-1 OMNI 发球机

1.2 捡球小车国内外研究现状

1.2.1 乒乓球捡球机器人研究现状

人工智能（AI）领域是当前最为活跃和最具应用前景的研究方向之一。高性能计算技术，云计算技术，大数据技术，物联网技术的进步与发展，都在推动人工智能技术走向更加成熟的阶段。然而，人工智能在文体领域中的应用才刚刚开始，学术界与工程界关于捡球机器人方面的研究很少，而我国对于乒乓球捡球机器人领域的研究相比更是少之又少，尚未有成熟系统的成果。

张行[1]等人开发设计了一款便携式乒乓球自动收集机器人，该机器人采用 ARM9 作为主控板，通过 CMOS 图像传感器识别场馆内随机散落的乒乓球，并采用红外传感器进行避障。此机器人在设计时更多的是从理论上实现的，其实际价值大大降低。

赵雪川[2]等设计了涵道式乒乓球捡球机，利用涵道中风扇叶片高速转动所产生的吸力把乒乓球吸到壳体中。这种捡球机器人的机械结构设计十分简单，容易实现产品落地，但是工作效率受到环境风速以及涵道密闭性等因素的影响，并且不具备避障功能。

安丹阳[3]设计了一种新型自主式乒乓球捡球机器人，详细阐述了其构形及关键技术研究，但未进行捡球机器人的实体设计。

可以看出，目前我国对于乒乓球捡球机器人设计还不是很多，而研究与设计的方向主要集中于捡球机构设计[4]、机器视觉[5]以及路径规划这三大部分。在市场需求上方面，存在着

较大的市场缺口，对乒乓球捡球机器人的研究还需进一步深入。

在国外，针对乒乓球捡球机器人方面的研究更加捉襟见肘，且多侧重于商业使用。迄今为止，已问世且评价良好的相关产品仅有高尔夫捡球机器人与网球机器人 Tennibot，但是与乒乓球相比，高尔夫与网球场馆面积较大，避障与路径规划重要性大打折扣，设计难度低。图 1-2 展示的是在售的 Tennibot 网球捡球机器人。

图 1-2 Tennibot 捡球机器人

1.2.2 乒乓球识别研究现状

机器视觉以计算机视觉理论工程化为基础，涉及光学成像技术[6]，视觉信息处理技术，人工智能技术和机电一体化技术。其在工业生产过程中具有广泛的应用前景。近年来，机器视觉在工业检测、军事及医学等多个领域都取得了重大进展，并逐渐形成一个新兴的产业——机器视觉技术。它以其非接触、实时性好、造价低等优点成为目前研究的热点之一。其中基于深度神经网络（DNN）的图像处理也是当前最热门的研究领域之一[7]。机器视觉主要包括两个方面：一是特征提取；二是模式识别。随着深度学习技术的不断完善，传统机器视觉得到了前所未有的发展，图像信息提取能力大大增强。伴随着机器视觉技术的飞速发展，图像处理技术被广泛应用于文体领域，通过图像处理技术能够更快解决乒乓球的识别问题。对于乒乓球识别方法可分为传统图像识别方法与机器学习识别方法。

机器学习对训练样本的数量和训练样本的纯净度都有很大的要求。对于一个给定的数据

集而言，要获得足够多的训练样本就必须花费很长时间去筛选，才能选出其中最适合建模的训练样本。但是在实际应用中，往往需要从成千上万个甚至上千万种的样本当中挑选出最优的训练样本，这就使得很多时候不能保证所选训练样本的质量。这样做不仅费时费力，而且会降低系统性能。因此，如何选择合适的训练样本来构建模型成为了一项非常重要而又困难的任务。传统的训练方法都是采用人工选取的方式对其进行测试并得到结果。由于人工挑选的样本量太少，且每个人可能具有自己特定的特征，所以无法完全适应机器学习算法的需求。这种做法存在很大局限性。利用大量高纯净度的训练样本可以训练出一个针对乒乓球识别的通用模型，再将待检测的图像使用训练好的模型进行识别。机器学习方法实施过于复杂，而且训练十分耗时，不方便对模型进行快速布署。训练所需训练样本较多，特别是在大数据时代，训练样本不仅结构不同，而且分布也不同，很难保证训练数据和测试数据能够满足机器学习模型中关于训练的独立同分布的假设。

这种情况下，传统图像识别方法相对于机器学习识别方法的优势得到了体现。乒乓球识别所需的轮廓与颜色这两个特征可利用 OpenCV 库迅速得到。OpenCV 专注于在现实中进行实时应用，其执行速度得到了可观的提升，并且通过编写优化的 C 代码实现了较好的实时性。

1.2.3 捡球路径规划方法研究现状

捡球路径规划方法总是希望能够找到一条最优的捡球路径。捡球路径规划方法可按已知信息完整度划分为全局遍历法，最邻近法以及已知乒乓球落点规划法。

全局遍历法（Global Sensing）是移动机器人领域最常用的方法之一，其算法简单，但是对于大区域来说会十分费时。

最邻近法（Multiple Projection）,顾名思义就是选取与移动机器人最接近的球为目标，但这样一来，若环境中存在较大障碍物阻挡球时，移动机器人就无法辨识全部区域并达到全局最优解。

已知乒乓球落点规划法实质上是一个多目标遍历问题，目前主流的解决方法分为深度优先和广度优先遍历算法以及动态规划算法。对于一般的单目标函数优化来说，深度优先算法能够很好地求解出全局最优点；而对复杂多峰性的多目标问题而言，广度优先算法则需要进行多次迭代才能得到满意解。动态规划算法虽然能在多目标的情况下找到最优路径，但是在实际工程应用中的效果并不理想。

蒲兴成团队[8]提出了一种基于改进粒子群算法（PSO）的移动机器人多目标点路径规划方法，为了解决粒子群算法早熟[9]问题，在粒子群算法中引入反向学习策略，对粒子群算法惯性权重与学习因子进行了改进，性能测试表明改进粒子群算法能够有效地避免粒子早熟，提高粒子群算法寻优能力与稳定性。

1.2.4 捡球小车避障研究现状

在机器人领域有关于避障的研究一直是热点研究方向。传统的方法主要依靠视觉导航和超声波测距技术实现。但是这两种方法都存在一定缺陷。随着学术界众多学者对它的深入研究，国内外学者们提出了很多新理论[10][11][12]。

1.人工势场法(Attention Point Field)。人工势场法在 1994 年被提出。在此算法中，目标与障碍物分别被认为是对机器人具有引力和斥力作用的对象，机器人沿着引力和斥力共同作用的方向运动。

2.向量场直方图法。该方法根据环境详细栅格地图构建机器人坐标系下障碍物概率直方图，根据概率直方图评估最优运动方向。以机器人当前位置为中心，构建栅格地图(Active Window)，对 Active Window 中的单元，根据实时传感器的检测信息建立障碍物的可信度概率，评估机器人每个方向上障碍物的密度，构成直方图，比较选取最优路径。

1.3 论文结构

根据捡球小车控制系统硬件和软件设计，本论文的内容安排如下：

第一章：绪论。在这一章中主要阐述了捡球小车研究的背景与意义，并从市场前景与实

用价值两方面给出捡球小车研究与设计的必要性。另外，对捡球小车在国内外的研究现状进行了描述。围绕乒乓球捡球机器人、乒乓球识别研究和捡球路径规划三个角度进行讨论。在本章最后对本文各章内容安排作了简要说明。

第二章：本章以捡球小车机械结构设计为目标，通过对捡球小车车身上各部件设计及安放位置进行合理设计，使得捡球小车能够平稳顺畅地运行。捡球小车机械结构设计。首先分析了捡球小车的设计需求，然后根据这些需求提出设计方案，并利用 SolidWorks 软件建立其三维模型，最后将其装配到一起，得到捡球小车样机实物。捡球小车捡球机构采用耙轮捡球机构来收集乒乓球，操作简单，效率高。

第三章：捡球小车控制方案设计。在这一章中，首先对捡球车控制系统的整体要求进行介绍，并对捡球小车运动控制方案，乒乓球识别控制方案以及自由避障控制方案进行详细设计。

第四章：捡球小车系统硬件设计。根据捡球小车的机械结构设计和控制方案设计，在本章节对捡球小车的硬件进行了选型。选用树莓派 4B 为主控制器，配以云台，舵机，马达，双路碰撞传感器及超声波传感器等组合使用。此外，还设计了捡球小车的硬件电路图。

第五章：系统软件设计和仿真分析。本章介绍了整个系统的软件设计，包括总体结构、各模块功能以及软件实现等方面的内容，在此基础上给出了部分程序代码并对其进行了验证。捡球小车的程序包括与手机通信的 socket 程序、摄像头识别效果调参程序、双路碰撞传感器程序、超声波测距程序、捡球小车前进后退程序以及串口信息发送程序。在多线程的控制下，这些程序运行得井然有序。

2 捡球小车的机械结构

本章节旨在设计捡球小车的机械结构。设计的机械结构既要保证捡球小车各个结构的合理性，也要保证捡球小车的平稳运行。捡球机构应当能够连续工作，在满足结构强度的前提下，尽可能的降低结构质量和单次拾取乒乓球的容纳量，提高续航能力。

2.1 捡球小车设计需求分析

伴随着社会经济的高速发展和人们对健康生活的追求，人们逐渐走出室内，参与到全民健身当中。乒乓球项目凭借其占地面积小、所需器材携带方便、入门门槛低等诸多优点，成为了许多人的首选健身项目。尤其是中国乒乓球队在今年的东京奥运会上斩获 5 金 3 银的傲人成绩，让越来越多青少年甚至中年人愿意选择这项既有益于身心健康又可以强身健体的运动。结合当前国家对体育的重视程度和全民健身目标的提出，越来越多的乒乓球手在国际大赛上不断为国争光，整个乒乓球项目的未来前景一片向好。

在练习乒乓球发球技术的时候需要大量的击发乒乓球，次数可达几千次甚至上万次。乒乓球正规比赛中对乒乓球场馆要求场地长不小于 14 米，宽不小于 7 米。对于乒乓球手来说，乒乓球各项技术训练就十分的消耗体力，还需要去捡起随机掉落在场馆各个地方的乒乓球无疑加重了乒乓球手体力消耗。对于乒乓球运动的初学者来说，在某种程度上也会磨灭他们对这项运动的激情。

为解决这个问题，使乒乓球手摆脱捡球这种重复性的体力劳动，急需为乒乓球研制和设计一种新型捡球机器人，遗憾的是市场上尚无类似产品。如今，随着机器人相关技术如视觉技术，感知技术，材料技术，PCB 技术等的发展，为同捡球机器人一样的服务型机器人提供了有力的技术支撑，使乒乓球捡球小车的实现有了客观条件。

2.2 捡球小车整体机械结构

2.2.1 捡球小车整体概览

本论文设计一种针对乒乓球的自动捡球智能小车。其能够在无人参与的情况下，实现自

主捡球、避障、前往指定地点等功能，同时也可以实现手动操作捡球小车实现对乒乓球场馆内随机散落的乒乓球做到无死角的搜索和拾取。捡球小车的整体机械结构示意图如 2-1 所示。

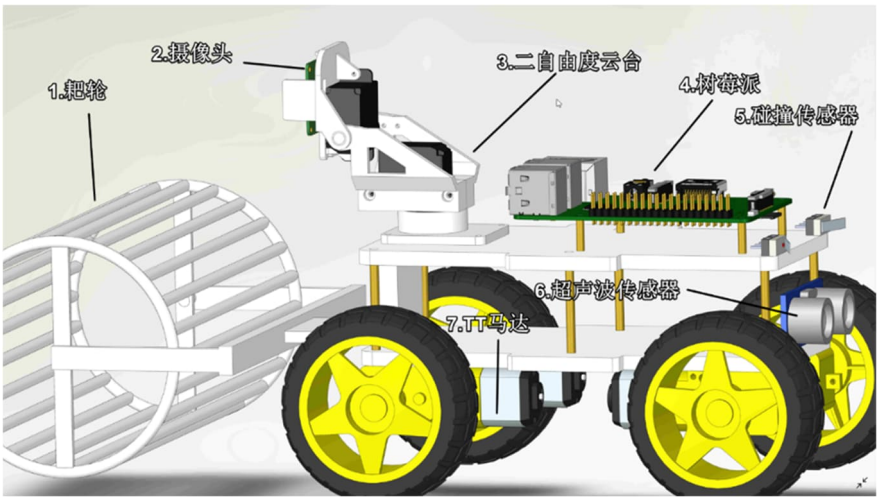

图 2-1 小车整体视图

从图 2-1 可以看出捡球小车的机械机构设计分为 7 部分。分别是负责捡球的耙轮、成像和搜索的摄像头和云台、主控制器、提供动力的马达和负责避障的碰撞传感器和超声波传感器。

2.2.2 具体部件作用描述

捡球小车最前端的耙轮结构用来拾取和容纳乒乓球。该机构是为了解决传统捡球小车中存在的结构复杂、成本高以及维修不便等问题而设计的。它通过旋转耙齿来抓取小球并将其运送到指定位置。耙轮结构由滚轮和支架拼接而成，采用 3D 打印一体成型技术，在保证了结构强度的基础上，也极大地减轻了重量。耙轮通过刚性连接牢固地卡入捡球小车前部，也便于后期维修更换。

支撑捡球小车的车身采用了两片 4WD 智能小车底盘，长约 190mm，宽约 140mm，厚度约 5mm。两片底盘使用 6 根铜柱作为支撑，便于安装捡球小车其他的硬件结构。4wd 智能小车底盘使用亚克力材料制作，5mm 的车身厚度使得底盘的结构强度大大提高，十分的抗摔，耐磨损。亚克力材料的重量很轻，两片底盘拼接在一起重量也只有 80g，大大减轻了电

机的负担。图 2-2 为捡球小车底盘实物图。

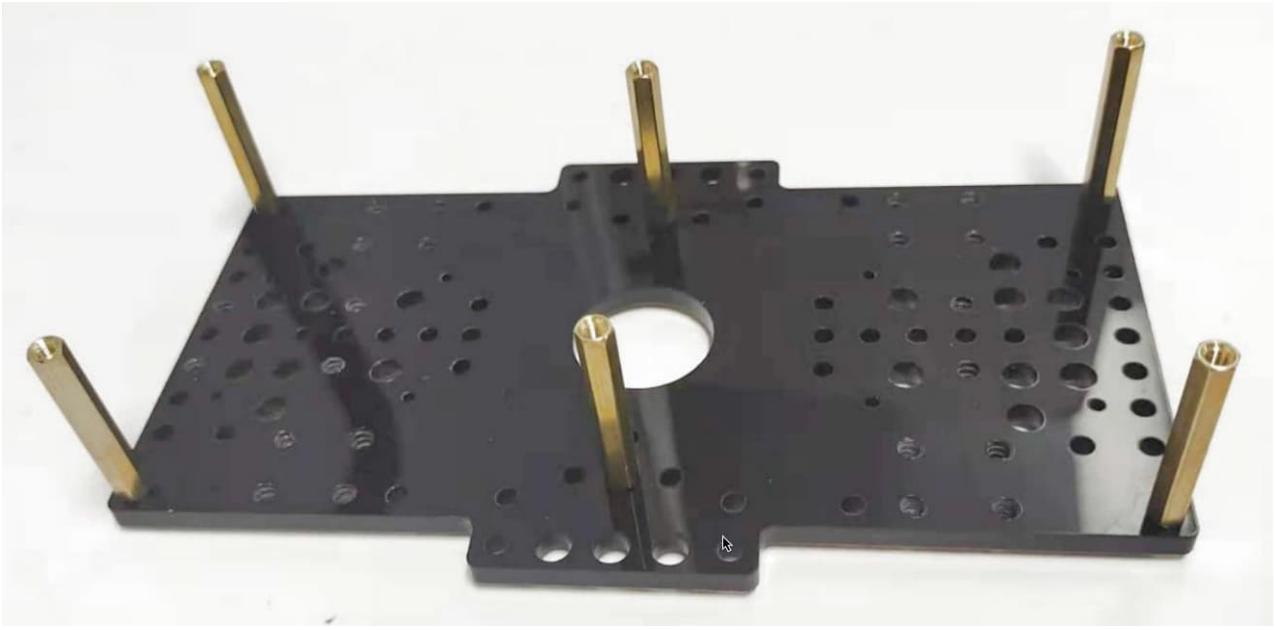

图 2-1 捡球小车底盘

TT 马达为整个捡球小车提供动力，可以接收来自主控制器的 PWM 信号调整旋转速度和旋转方向，从而达到控制捡球小车的前进速度和行进转向的目的。

云台和摄像头搭配使用起到搜索和识别环境中乒乓球的工作。云台的机械结构设计允许其可以左右各 90°和上下各 90°带动摄像头转动。云台的制作材料则是普通塑料，具有成本低，重量轻，易更换等优点。

双路碰撞传感器和超声波传感器则是起到自由避障的作用，确保捡球小车在行驶的过程中能够不与周围环境中的障碍物发生碰撞。

上述机械结构均采用螺丝或者榫卯结构安装在智能小车底盘。智能小车底盘有很多孔洞，方便调整变换已有的机械结构，也有利于后期升级加装其他的机械结构。

2.3 捡球小车工作原理

捡球小车的工作模式有手动和自动两种。自动模式依靠云台带动摄像头旋转，寻找视野范围内随机散落的乒乓球，主控制器会根据摄像头传输回来的图像信息判断找到的乒乓球个

数和规划最佳捡球路径。捡球小车在按照规划路径行驶的时候，自由避障程序作为中断程序也会同步在后台等待运行。当出现障碍物时，避障程序会获得捡球小车的主控制权，避障完成后再重新规划路径，最终将场馆内随机散落的乒乓球全部拾取。

3 捡球小车控制方案设计

本文旨在设计一种基于 OpenCV 视觉识别技术的乒乓球捡球小车，实现全自动或者手动参与下的捡球操作，能够通过二自由度云台的摄像头能够识别出其所处环境中随机散布的多个乒乓球，并且将结果传送至主控制器中进行数据处理，然后向捡球小车运动控制系统模块发出命令引导捡球小车平稳快速地向目标球所在位置运动，使得目标球能够被滚入乒乓球捡球模块耙轮中。

3.1 控制方案设计要求

设计的捡球小车要求其能够在无人参与的情况下，实现自主捡球、避障等功能；人工操作下能正确接接收理来自 APP 的指令，稳定的传输图像。设计目标的要求如下：

控制系统模块:模块化设计,便于后期升级改装,发生故障时进行排障;能确保捡球机器人平稳、快速、准确地完成捡球,并有一定抗干扰性。

视觉系统模块：能为乒乓球识别程序提供清晰的图像，并且能够对场馆内一定强度的光照具有鲁棒性。

避障模块：能够确保捡球小车无论是在前行还是后退的过程中不会和周围环境障碍发生碰撞，顺利完成捡球工作。

3.2 控制系统需要实现的功能

本文所设计的捡球小车分为手动操纵方式和自动操纵方式。手动模式时，捡球小车需成功与手机 APP 建立通讯，并实时向手机端传输摄像头成像内容，并接收来自手机端的“ 前进 ”、“ 后退 ”、“ 左转 ”、“ 右转 ” 等控制指令。捡球小车在接收手机端指令和实时图传时，应尽量减少传输延迟。捡球小车处于自动模式时，既要独立自主地完成以上手动模式的控制命令，又要精确地判断什么时候执行什么命令以及对场馆内乒乓球进行精确识别，并能抵御一定强度光线的干扰。

捡球小车在自动模式时，需把摄像头探测的环境信息传给主控制器进行处理，以便为捡

球小车发出什么样的控制指令。捡球小车行驶过程中自由避障程序亦应同步进行。当自由避障程序判断捡球小车周围存在障碍物时，捡球小车的中断被触发，自由避障程序获得捡球小车的主控制权，引导捡球小车避开障碍，完成避障后退出中断，再将捡球小车的主控制权交还给主控制程序。

在自动模式中，捡球小车除了要完成这些功能外，还会根据搜索点的不同而自动进行转移。搜索点是放置在乒乓球场馆四个角落的具有醒目颜色的方块。捡球小车判断出当前搜索区域的乒乓球已经拾取完毕后，会驶向下一个搜索点再次进行乒乓球的搜索与拾取工作，这样四个搜索点都完成搜索后即可认为乒乓球场馆内的乒乓球都已拾取完毕，完成局部解到全局解的过渡。

3.3 控制系统方案设计

捡球机器人的控制方案将紧密围绕运动控制、乒乓球识别和自由避障三个方面展开设计。

3.3.1 运动控制方案设计

捡球小车的运动控制方案设计是小车整体控制方案设计的重要一环，运动控制方案设计的优劣决定了捡球小车能否平稳顺畅的运动。图 3-1 是捡球小车的运动控制方案。

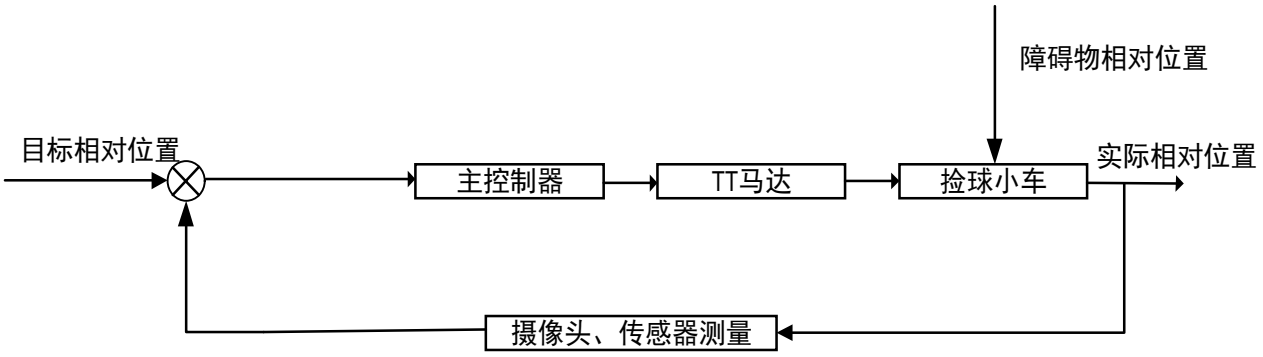

图 3-1 运动控制方案

（1）给定值：目标相对位置

（2）执行器：TT 马达

（3）被控对象：捡球小车

（4）测量变送器：摄像头和传感器

（5）扰动变量：障碍物相对位置

（6）被控变量：实际相对位置

捡球小车的运动控制方案里面提及的相对位置是针对捡球小车位置和目标球体位置而言。通过二者之间的相对位置调整捡球小车的行驶方向和行驶速度。

根据 3.2 小节提到的内容，当障碍物的相对位置影响到捡球小车的正常行驶的时候，捡球小车的运动控制权会由运动控制程序暂时移交给自由避障程序，等障碍物的相对位置不再影响捡球小车行驶时，运动控制程序再控制捡球小车的行驶。

3.3.2 乒乓球识别方案设计

乒乓球识别方案设计的高效合理与否决定了捡球小车最后执行捡球工作的质量高低。捡球小车通过搭载的云台带动摄像头扫描和识别场地里随机散落的乒乓球。搭载的云台可以实现左右和上下方向上各 180°的转动，确保摄像头能够完全扫描和识别小车前方可能存在的乒乓球。图 3-2 展示的是云台舵机控制方案。

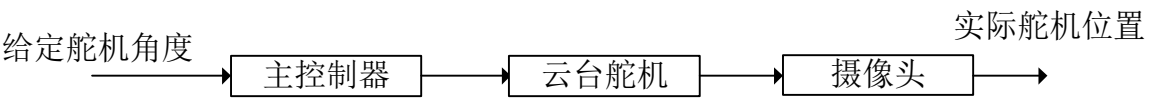

图 3-2 云台舵机控制方案

（1）设定值：给定舵机角度

（2）执行器：云台舵机

（3）被控对象：云台

（4）被控变量：实际舵机角度

由于云台舵机只需要带动摄像头转动扫描和识别场地里的乒乓球，所以这里的控制方案设计为开环控制，设计简单，容易控制。

3.3.3 自由避障方案设计

自由避障方案设计水准高低关乎到捡球小车能否顺利的行驶。关于捡球小车和场地里障碍物的距离测量和是否发生碰撞与否依靠的是超声波传感器和碰撞传感器。图 3-3 展示的是捡球小车的自由避障控制方案。

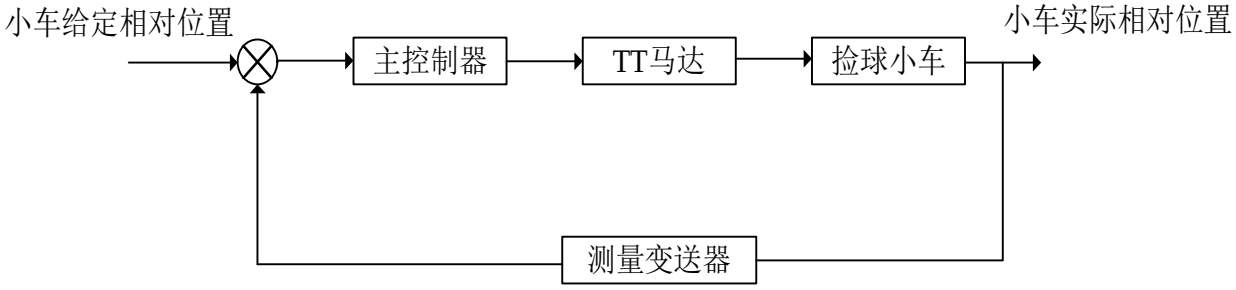

图 3-3 自由避障控制方案

（1）设定值：小车给定相对位置

（2）执行器：TT 马达

（3）被控对象：捡球小车

（4）被控变量：小车实际相对位置

（5）测量变送器：传感器

捡球小车和障碍物的相对位置通过摄像头和传感器测量获得，捡球小车的主控制器会根据已获得的相对位置对捡球小车的行驶方向做出修正，确保捡球小车不会和障碍物发生碰撞，顺利驶向目标球体。

3.4 捡球小车控制流程

捡球小车开机上电，如果是手动模式，用户控制捡球小车；如果选择自动模式，首先默认在当前位置搜索乒乓球，云台舵机带动摄像头转动搜寻目标，如果摄像头在成像范围内搜寻到乒乓球，主控制器会发出指令让捡球小车驶向目标球体，如果存在多个乒乓球，捡球小车会按照从左往右的顺序拾取。如果摄像头在当前搜索范围内没有搜寻到乒乓球，捡球小车会自行前往下一个目标搜索点，重复上述搜寻步骤，从而实现局部捡球到全局捡球的过渡。

在捡球小车行驶的过程中会设置中断，此中断提供给避障程序。如果自由避障程序判断出捡球小车和障碍物的相对位置影响到捡球小车的正常行驶，那么捡球小车会有自由避障程序控制，在绕开障碍物后再跳出中断，按照正常路径行驶。图 3-4 展示了捡球小车的整体工作流程。

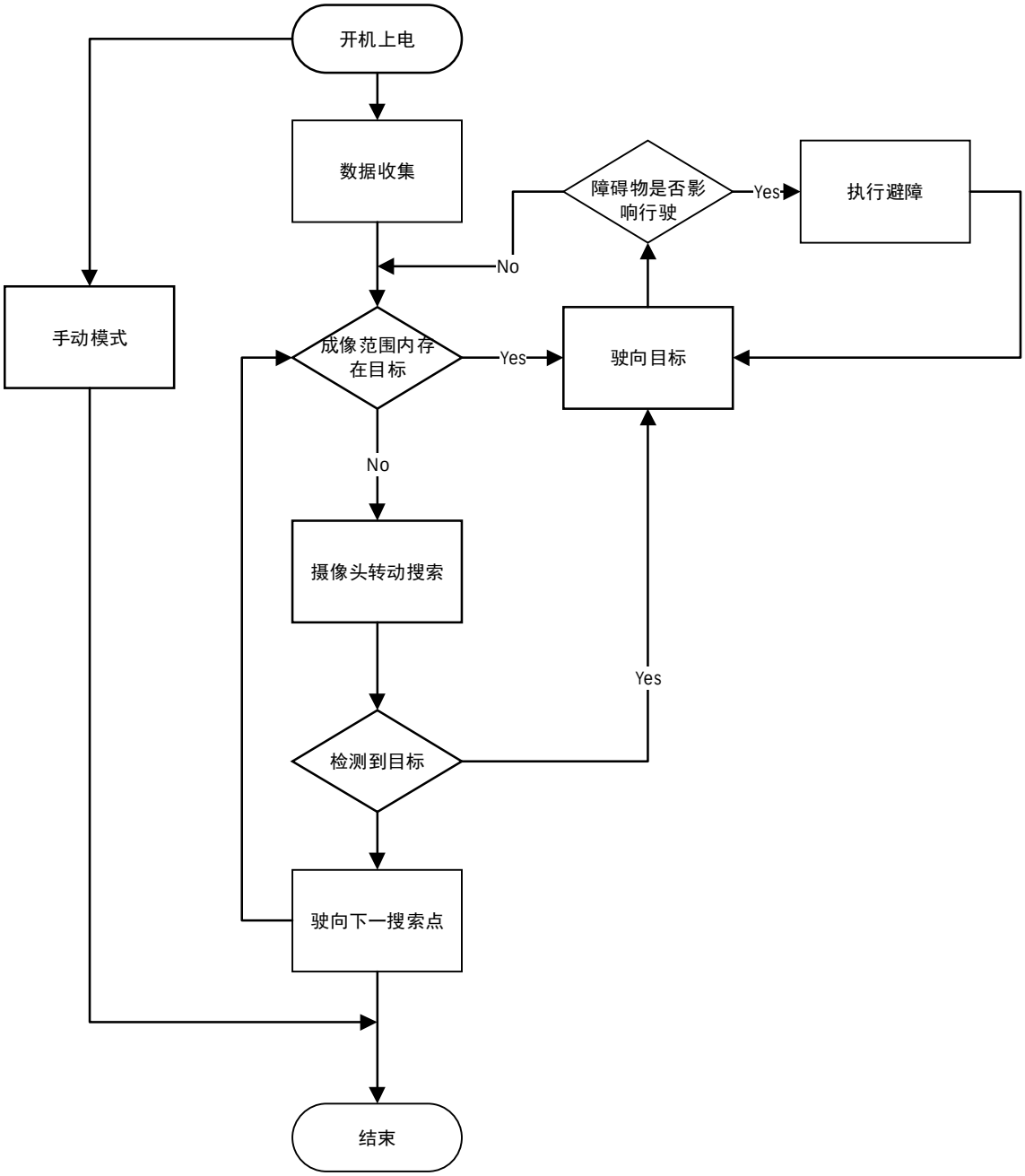

图 3-4 整体控制流程

4 捡球小车系统硬件设计

本论文设计的捡球小车通过各个模块之间的相互配合可以实现平稳顺利地拾取散落在场馆内的乒乓球。自动模式下，环境数据通过摄像头和各个传感器采集传输给树莓派 4B 进行分析处理，树莓派 4B 会依据得到的处理结果，对捡球小车发出最优的控制指令，确保捡球小车顺利完成捡球工作。手动模式下则是依靠人工控制捡球小车完成捡球工作。图 4-1 展示的是捡球小车硬件设计框图。

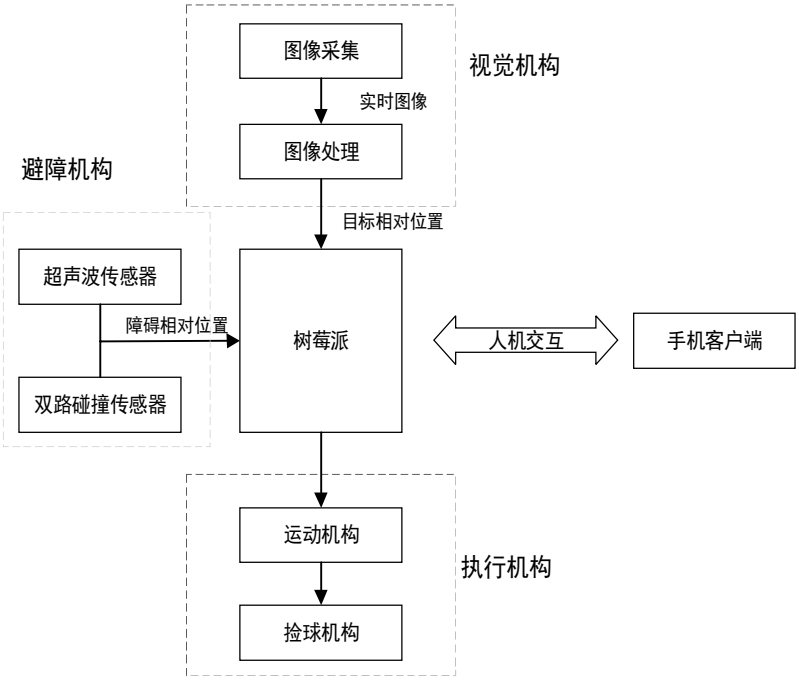

图 4-1 捡球小车硬件设计图

4.1 硬件选型

4.1.1 主控制器选型

主控制器相当于捡球小车的“大脑”，所以选择的主控制不仅要求其算力能够支持高效的运行捡球小车程序，还要求其能够支持捡球小车持续稳定工作。

市面上常见的开源控制器种类很多，用户的主流选择可以分为 C51、STM32、Arduino 和 Raspberry（以下简称树莓派）。本论文选择树莓派 4B 作为主控制器，树莓派是一款开源硬件，搭载着 1.5GHz 的 CPU、图形处理器、2GB 运行内存、USB 控制器等，构成一个片上系

统。树莓派 4B 提供 PC 级的性能，但功耗很低，使得捡球小车能够长时间连续工作。

树莓派提供了 40 个 GPIO（General PurposeInput/Output）引脚，同时为了接线方便还为树莓派配备了拓展板，使树莓派 4B 拥有了强大的拓展能力。图 4-2 和 4-3 展示了树莓派 4B 和扩展板实物图。

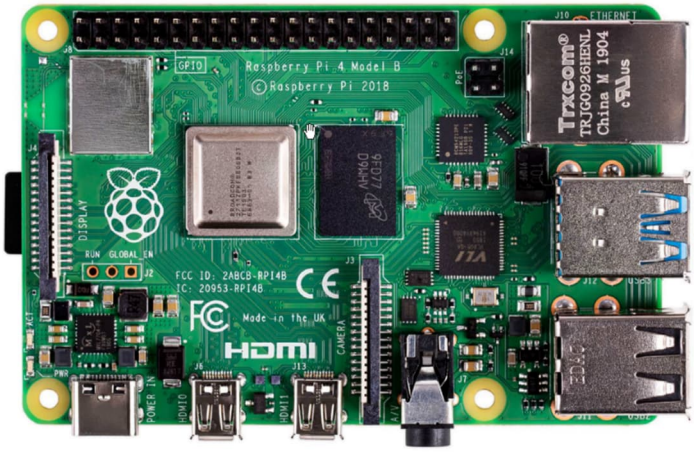

图 4-2 树莓派 4B 实物图

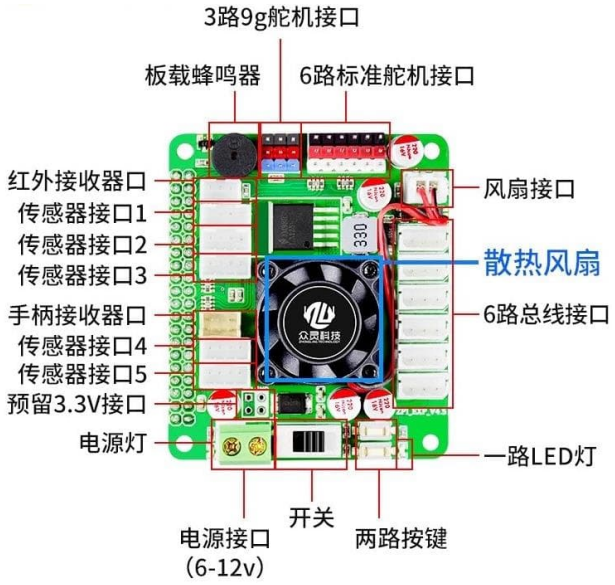

图 4-3 树莓派扩展板实物图

树莓派 4B 提供了 3 种方法可以对开发板上的 GPIO 引脚编号，分别是物理引脚编号、wringPi 编号和 BCM（Broadcom SOC）编号。本论文采用的是 BCM 编号，其基本信息可以

在树莓派 4B 终端输入 pinout 查看。图 4-4 展示树莓派的引脚分布。

图 4-4 BCM 基本配置信息

4.1.2 超声波传感器

超声波传感器顾名思义是一种能够发出超声波信号的传感器。超声波传感器发出的机械波信号不仅具有频率高、波长短、绕射现象小的优良特性，而且指向性十分优秀。在本论文中选择的超声波传感器型号是 HC-SR04，其测度精准、电气性能稳定、测度盲区小、体积小巧、工作功耗低、测量距离可达 400cm。图 4-5 和 4-6 分别展示了超声波传感器的实物图和引脚图

图 4-5 HC-SR04 实物图

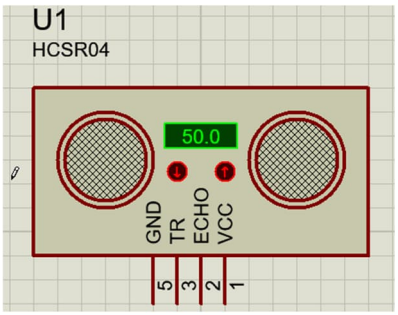

图 4-6 HC-SR04 接线图

表 4-1 引脚说明

引脚	说明
VCC	提供 5V 工作电压
Trig	控制端，用于出发超声波脉冲
Echo	接收端，接收反射信号
GND	接地

表 4-2 电气参数

名称	说明
工作电压	DC 5V
工作电流	15mA
工作频率	40Hz
输入触发信号	10us 的 TTL 脉冲

HC-SR04 的工作原理简单。在程序中最开头首先就将 Trig 和 Echo 端口都置为低电平，树莓派会给 Trig 端口发送持续时间至少 10 us 的高电平脉冲信号，超声波传感器也会自动向外发送 8 个 40KHZ 的声音脉冲。这 8 个 40KHz 的声音脉冲是超声波传感器特有的声音特征，能够与环境噪声区分开来。声音脉冲发出的同时，程序会将 Echo 端口置为高电平。

如果声音脉冲没有被环境中的物体反射回来，那么根据程序设计设计会认为 Echo 端口

在 38ms 后还没有收到回声信号就表示在超声波传感器的探测范围内没有障碍物，此时的工作时序图为图 4-7 所示。

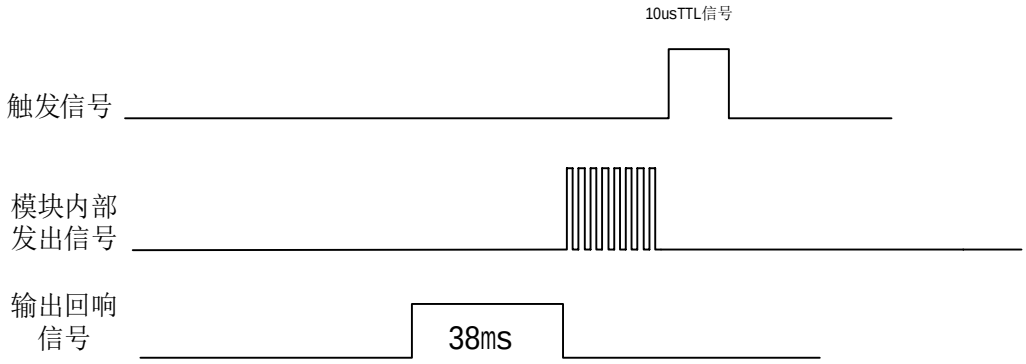

图 4-7 无障碍物时的工作时序图

如果声音脉冲被障碍物反射回来，那么 Echo 端口就会在收到被反射的声音脉冲后，会将引脚从高电平变置为低电平，同时程序会记录 Echo 高低电平变化时序的时间。高低电平变化的宽度取决于捡球小车与障碍物的距离在 150us 到 25ms 之间变化。此时超声波传感器的工作时序图如图 4-8 所示（假设脉冲宽度为 500us）。

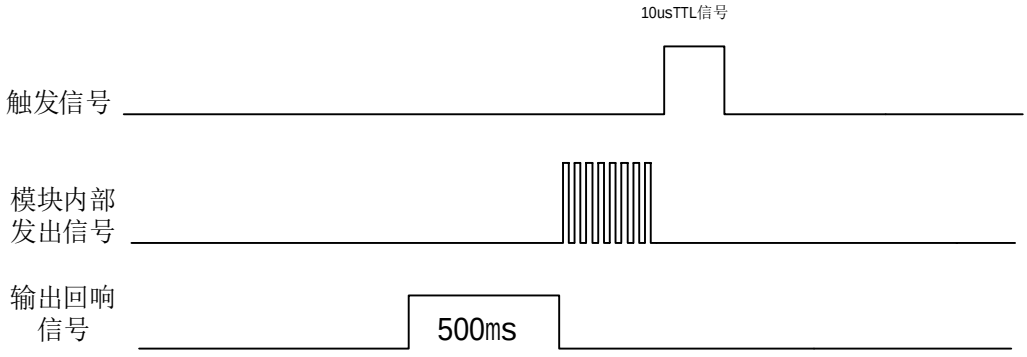

图 4-8 存在障碍物的工作时序图

有了脉冲宽度后，先将 us 的时间单位进行时间单位的转换，转换为 s 后根据测距公式就可以得到捡球小车距离障碍物的距离。

4.1.3 碰撞传感器

碰撞传感器又称微动开关，根据字面意思就可以知道微动开关只需很小的力就可以被触发。在默认状态下微动开关的 OUT 端口输出为低电平。在捡球小车碰到障碍物从而施加的

外力会瞬间使微动开关的动作簧片完成闭合，同时 OUT 端口输出高电平。当捡球小车和障碍物分离后动作簧片上的定动触点分离，OUT 端口又变为低电平。

微动开关的分类依据有很多种，比如按照体积分类、按照防护性能分类、按照分断形式分类等。微动开关在日常生活中使用的也很多，例如鼠标中就会安装。这也从侧面可以反映出微动开关的机械寿命很长，一般来说微动开关的使用次数在 3 万次到 1000 万次不等。

本论文选择的是小型微动开关，其尺寸仅为 28mm*21mm，重量只有 3g/个，方便安装在捡球小车尾部，也不会明显增加捡球小车的重量。同时小型微动开关的输出形式为数字开关量（0 和 1），所以无需单独加装模数转换器。图 4-9 和 4-10 展示了碰撞传感器的实物图和电路图。

图 4-9 碰撞传感器实物图

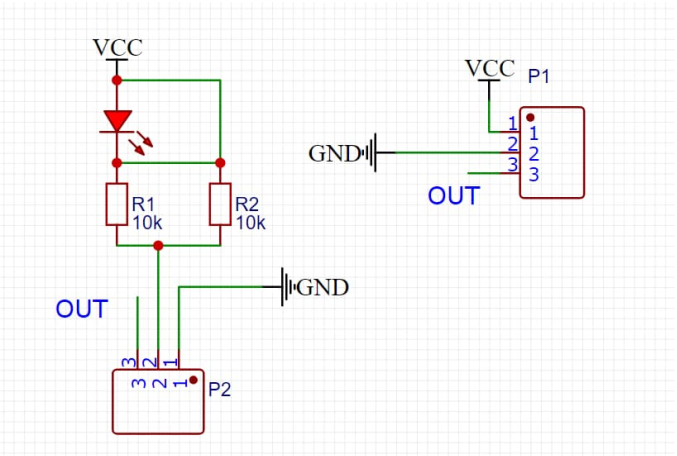

图 4-10 碰撞传感器接线图

表 4-3 引脚说明

引脚	说明
VCC	3V~5V 工作电压
GND	接地
OUT	高电平输出

4.1.4 电机选型

市面上流通的电机类别繁多，例如可以分为步进电机和直流减速电机。本论文选择的是 TT 双轴直流减速电机（或称 TT 马达），其凭借强磁、带抗干扰等特性被广泛应用于机器人和小车的动力模块。

TT 马达的减速比为 1:48，空载电流≤80mA，空载转速实测可达 180 转/分。捡球小车总共使用了四个 TT 马达为其提供动力，前后两组分别通过双路总线驱动模块连接。图 4-11 展示了 TT 马达的实物图。

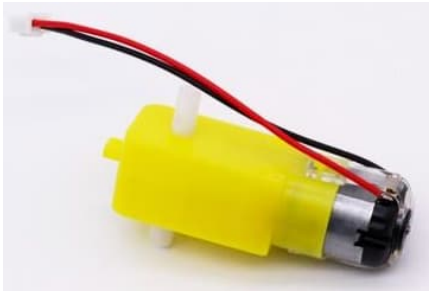

图 4-11 TT 马达实物图

4.1.5 电机驱动模块

L298N 电机驱动模块能够接受高压的的电机驱动模块，其不仅可以驱动步进电机而且由于内部有双 H 电桥，每个电桥可以提供 2A 电流，所以也可以驱动直流电机。一个 L298N 电机驱动模块可以同时驱动两个直流减速电机做不同动作，输出的电压在 6V 到 46V 之间，电流约为 2A，并且具有过热自断和反馈检测的功能。

但是在电机驱动模块的选择上本论文没有选择常用的 L298N 电机驱动模块，而是选择

了总线双路驱动模块，这样可以让捡球小车只需要 2 个电机驱动模块就可以驱动 4 个 TT 马达。

本论文选择的总线双路驱动模块内置了 BT6621 驱动模块和 STM32F030F4P6 控制芯片，该模块具备的一个 UART 串口信号即可完成电机速度的调速、转动时间以及转动方向的控制；模块可级联（串联到一起），方便拓展更多的模块；单总线通信协议，一根信号即可完成对所有设备的控制，支持修改波特率。图 4-12，4-13 展示了总线双路驱动模块实物图和 STM32F030F4P6 外围电路图。

图 4-12 总线双路驱动模块

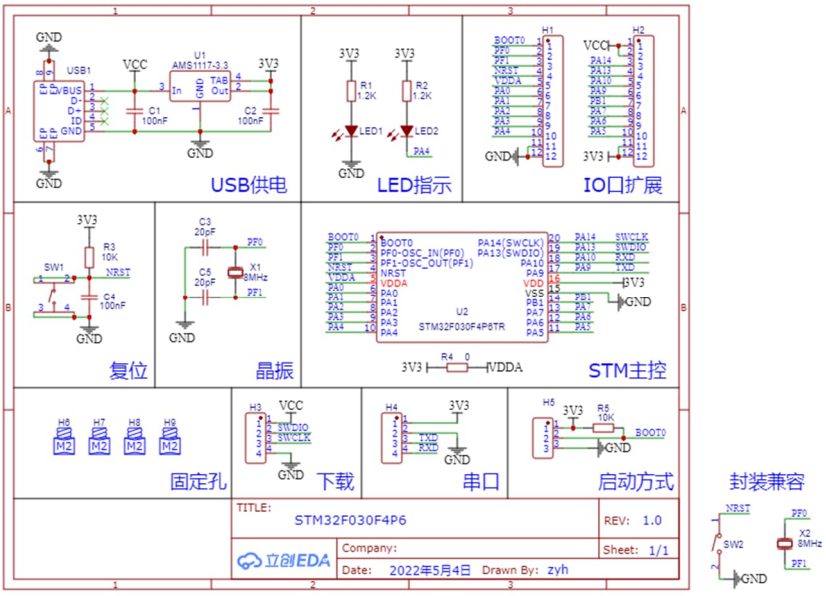

图 4-13 STM32F030F4P6 外围电路图

总线接口规格为 3Pin，分别是 TX/RX，VCC，GND。

表 4-4 引脚说明

引脚	说明
VCC	提供 6~12V 工作电压
GND	接地
TX/RX	通信接口

四个 TT 马达连接到两个双路总线驱动模块后，再由双路总线驱动电机引出一根总线对接线连接到树莓派的扩展板上，即可实现树莓派对 TT 马达的控制。

表 4-5 通讯协议

指令	注释
#idPpwmTtime!	电机控制指令：比如#000P1500T1000! 1：相当于每个总线电机的“名字”，其范围是 000 到 254，必须为三位数，不足的位数补 0。255 为广播 ID，所有设备都会响应这个指令； 2）pwm 的范围是 0500 至 2500 ，必须为四位数，不足的位数补 0，1500 表示停止，大于 1500 为正转，数值越大转速越快，小于 1500 为反转，数值越小反向转速越快； 3）Time 表示旋转时间(单位 s)，必须为四位数范围：0000 至 9999，特殊的 0 代表循环执行。
#000PID006! 双路 ID 修改回复：#007+008P!	更改 ID 指令：#000PID006!含义是把 ID 号为 000 的总线模块改成 ID 号为 006+007。此处只适用于双路驱动模块。
#000PID!	读取 ID 指令：读取当前电机的 ID 号#000PID!，一般读取指令#255PID!广播读取指令
#000PBD4!	波特率设置：设置舵机通信波特率，默认 115200。数字参数对应关系为：1-9600，2-19200，3-38400，4-57600，5-115200，6-128000，7-256000，8-1000000，该指令设置成功后返回#0K!。

4.1.6 云台舵机

市面上可供用户选择的舵机型号数量非常庞大，例如双轴总线舵机、PWM 单双轴舵机、SG90 舵机、SG92R 舵机等。本论文选择的是转动角度为 180 ° 的 SG90 舵机。将两个 SG90

舵机安装在云台上，从而带动摄像头转动。

舵机是一种角度伺服电机。舵机的转动角度由 PWM 脉冲控制，每通过一个 PWM 脉冲，舵机会转动一个特定的角度。得益于舵机的自身设计，舵机具备发出脉冲的功能。由此，舵机每需要转动一个特定的角度，其都会发出对应的脉冲数量。这样舵机转动的角度就和接收的脉冲数量构成闭环。使用这种舵机，树莓派可以准确统计出知道发了多少数量的脉冲给舵机，同时又收回了多少数量的脉冲，借此树莓派就能够很精确的控制舵机的转动角度，从而实现对舵机转动角度的精确定位。图 4-14 展示的是 SG90 舵机实物图，图 4-15 展示的是脉冲和角度的对应关系图。

图 4-14 SG90 舵机

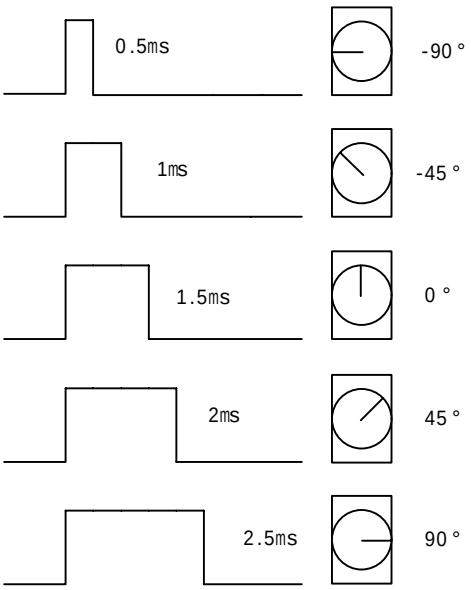

图 4-15 脉冲宽度与角度关系

4.1.7 摄像头

摄像头成像是否清晰关系到树莓派能否准确的识别到场馆里随机散落的乒乓球，所以摄像头的选择第一要义是尽可能的清晰地拍摄场馆图像。摄像头可供的选择也十分的众多，例如 ROS 深度相机，这种相机能够摄取环境的立体图像，让感知深度成为其优势，但是价格十分的昂贵，入门深度相机的均价为 1000 人民币左右。结合成像质量和成本的综合考虑，本论文选取的是树莓派官方高清夜视摄像头。图 4-16 展示的是树莓派官方摄像头实物图。

图 4-16 树莓派官方摄像头

此款摄像头的感光芯片使用的是 Sony IMX219、光圈可达 2.0、拥有定焦镜头、焦距为 3.04mm、拍摄照片分辨率为：3280×2464、拍摄视频规格支持 1080P30/720P60/480P90、畸变小于 1%。此外摄像头还安装了红外感光器，能够清晰地拍摄场馆里光线不好的地方。树莓派为官方摄像头提供了专门的接口（CSI 接口），能够轻松地插入，用户不用担心因为产品型号不支持带来的不兼容问题。

摄像头和二自由度云台通过螺母螺帽结构紧密拼接在一起，构成捡球小车的视觉系统，为树莓派稳快准地提供场馆图像。图 4-17 展示的是摄像头和二自由度云台安装在一起的实物图。

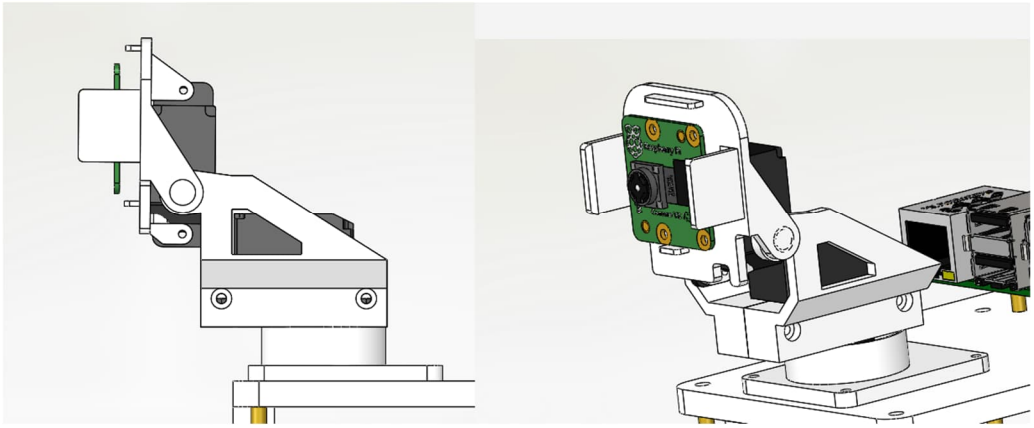

图 4-17 摄像头和二自由度云台示意图

4.1.8 供电模块

得益于树莓派的优异电路设计，使得用户只需要通过 Micro USB-C 口给树莓派提供 5V/3A 的电源即可。树莓派的 GPIO 引脚的高电压有 3.3V 和 5V 可供选择，输出的电流规定每个引脚最大输出电流为 16mA。为了避免捡球小车搭载的设备过多，导致出现功率不够，带不动负载的情况，这里直接选择给扩展板供电。扩展板支持 6~12V 的电压，最大工作电流为 10A，无需担心供电不稳导致的各种问题。同时扩展板也设计了开关，方便用户随时给树莓派供电或者断电。

图 4-18 捡球小车供电电池

本论文选择了航模 2s 高倍率充电电池给捡球小车供电。该款充电电池体积小，可以直

接塞入捡球小车；使用 SM 接口，能够防止反接；带有过充过放保护功能；能够输出 7.4V 的电压，电池容量有 2200mAh，实际测试中能够支持捡球小车工作四十分钟左右。使用对接线和扩展板连接，方便快速更换电池。多块电池搭配使用，能让捡球小车的续航时间大大提升，提高工作效率。

4.2 总体硬件电路设计

图 4-20 展示了捡球小车的总体硬件电路设计图。

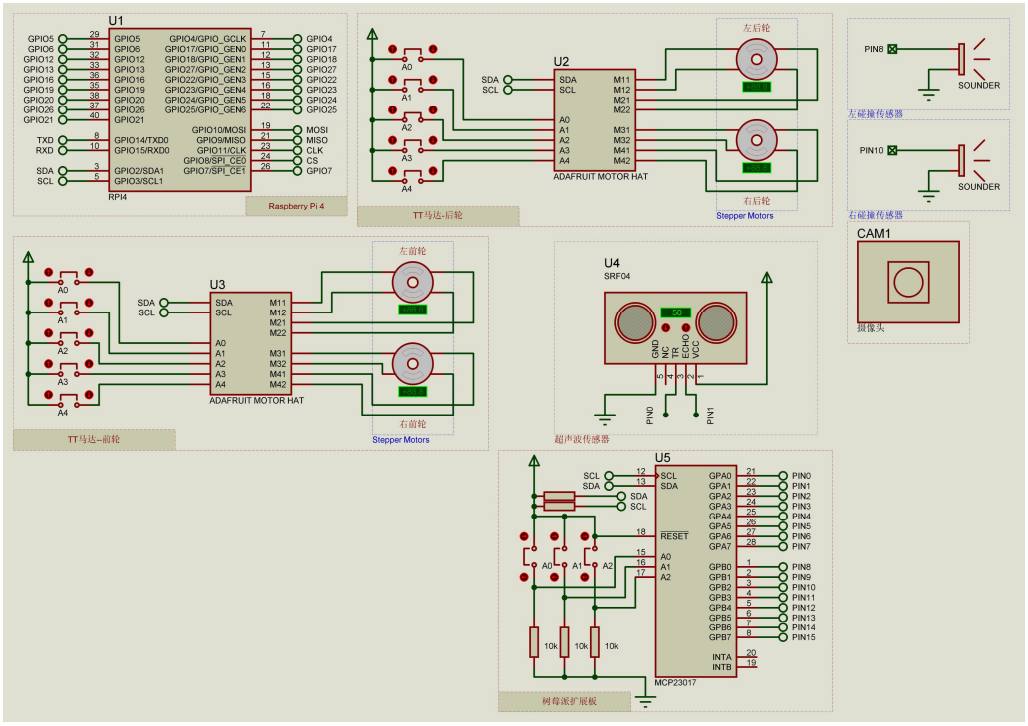

图 4-19 捡球小车总体硬件设计

图中左上角为主控制器树莓派 4B。在驱动模块位置设计了五个按钮，从上到下分别代表着：“ 前进 ”、“ 后退 ”、“ 左转 ”、“ 右转 ”、“ 停止 ”。由于 Proteus 没有碰撞传感器，所以使用蜂鸣器做了替代。图中最下方是树莓派扩展板。

5 系统软件设计和实物测试

前面的章节已经完成了捡球小车的控制方案设计、硬件选型和硬件电路的设计，本章节将在前面工作已经完成的基础上，开始设计捡球小车的运行程序。本章节将围绕以下 6 个主要功能进行程序的设计和讲解。代码篇幅适中的会直接在论文里展示，而代码篇幅较长的在论文中只会展示核心代码，全部代码会以附录的形式附在论文最后。

5.1 捡球小车程序主流程图

图 5-1 展示了捡球小车程序的主流程图，捡球小车的所有程序的设计都以此为参照进行开发设计。

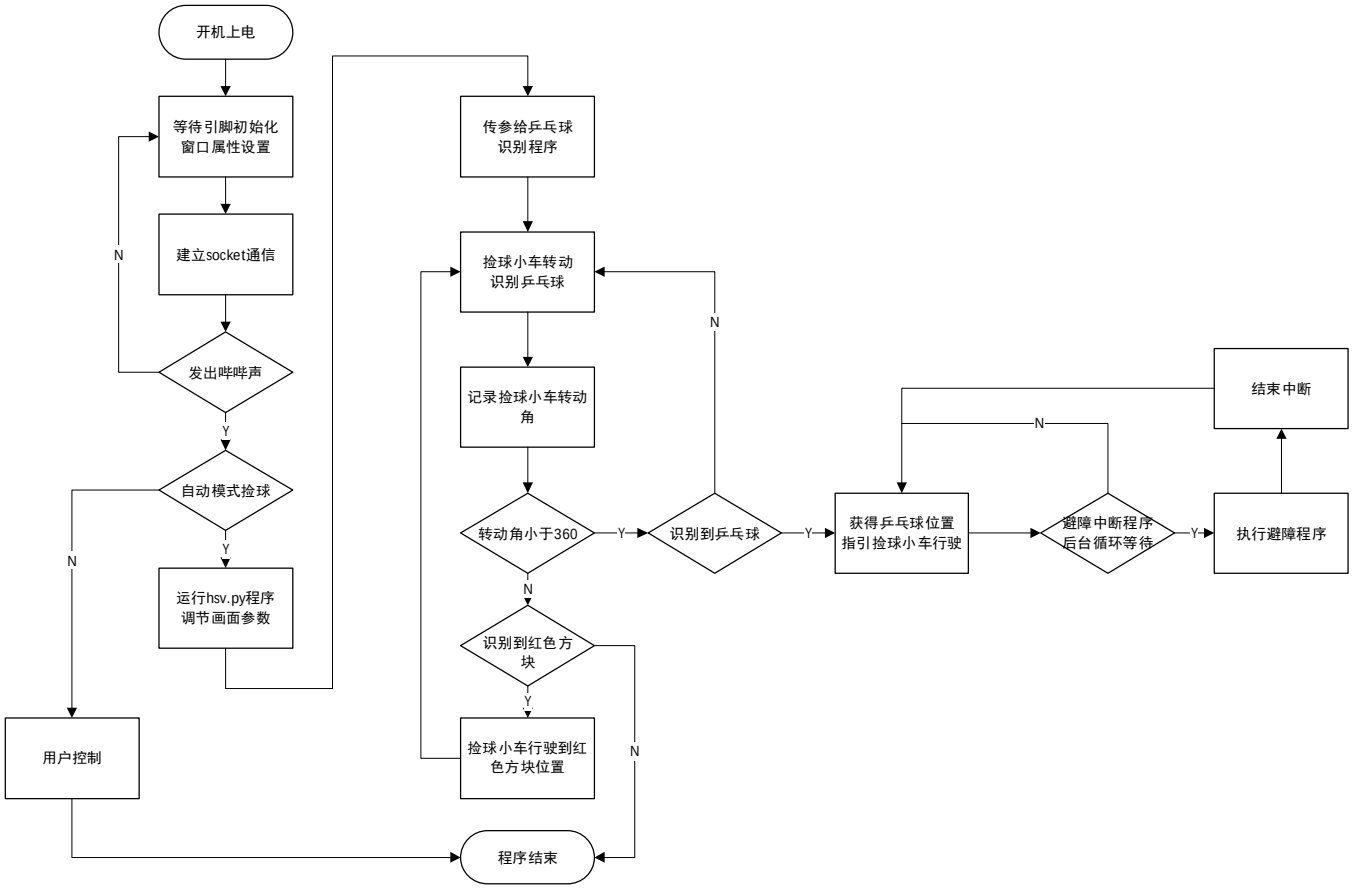

图 5-1 捡球小车总体程序流程图

捡球小车在上电和设置初始化完成后会发出哔哔声，提示用户可以选择工作模式。手动模式，用户需要正确在 APP 里输入 IP 地址。自动模式下，只需运行主程序，捡球小车就会

开始自动搜索场馆里散落的乒乓球，识别成功后自动驶向目标球体，在行驶的过程中，碰撞中断程序一直在后台循环等待。如果捡球小车旋转一圈后未发现乒乓球，就会自动识别并驶向固定位置的红色方块，在红色方块位置接着搜索。如果在红色方块的位置仍未发现乒乓球即可认为场馆内的乒乓球拾取完毕。

5.2 开发环境搭建

5.2.1 IDE 选择

IDE（Integrated Development Environment）选择的是 Pycharm 专业版。Pycharm 一个非常优秀的专业软件。它提供了许多功能强大的工具和丰富实用的内容，为开发者提供了很好的开发环境。同时也使开发者能够更加容易地使用工具包。

由于树莓派的 PRi.gpio 库不支持在树莓派之外的平台使用，所以需要远程开发或者树莓派外接屏幕开发。外接屏幕十分的不便，而 Pycharm 专业版提供了 SSH 远程连接，只需要树莓派和 Pycharm 处于同一网络下，配置好连接端口即可进行开发。

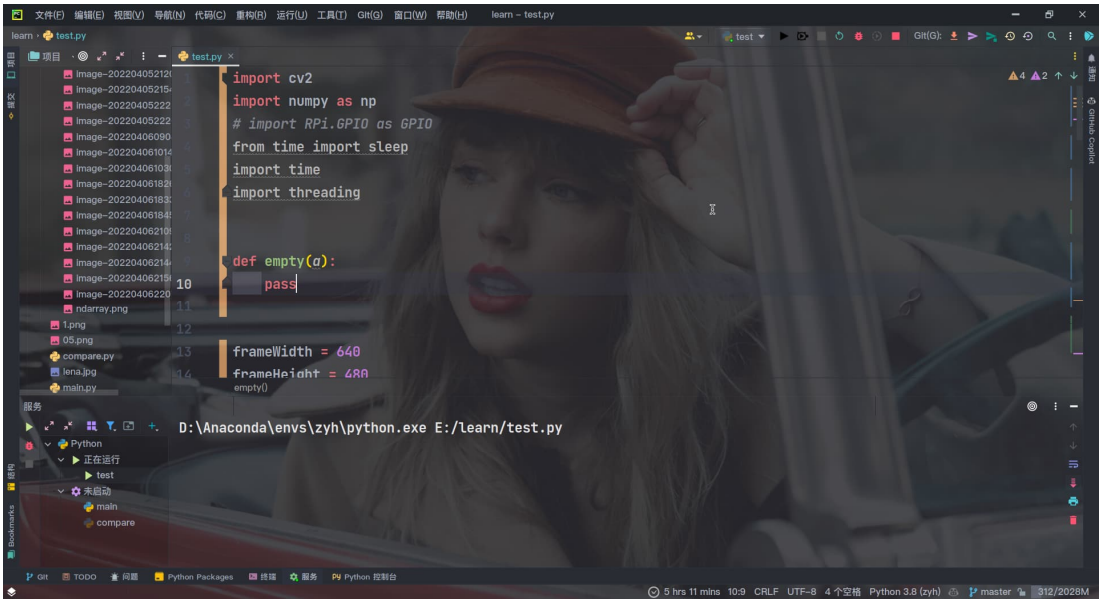

图 5-2 Pycharm 开发界面

5.2.2 树莓派操作系统选择

树莓派官方拥有强大的社区，社区里提供了针对各种使用环境的操作系统。例如针对音

乐播放开发的 Volumio 操作系统，针对家用模拟机开发的 RetroPie 操作系统。本论文选择的是 Raspberry Pi OS（Raspbian）的 64 位桌面版本。相较于其他操作系统，Raspbian 操作系统树莓派官方对硬件驱动和软件程序做了深度定制，而且配备了图形化操作界面，方便用户快速上手。使用官方提供的 Raspberry Pi Imager 镜像烧录工具将操作系统烧录到 SD 卡，然后插入树莓派即可。SD 卡容量建议选择 32GB 及以上。

5.2.3 树莓派配置

树莓派在默认情况下不开启 SSSH 连接，需要用户手动打开。为了操作方便，本论文选择使用 MOBAXterm 远程配置树莓派。

首先在本地电脑里新建 wpa_supplicant.conf 文件，用记事本输入以下代码：

```
network={
ssid="你的无线网络名称（ssid）"
key_mgmt=WPA-PSK
psk="你的 wifi 密码"
}
```

之后在新建名为 ssh 的无后缀文件。将这两个文件拖入到树莓派的 boot 盘根目录下即可开启 SSH 连接并接入到 WiFi。

打开 MobaXterm，选择 Session，新建 SSH 连接，输入树莓派连接到路由器的 IP 地址，用户名输入 Pi。Pi 用户的默认密码为 raspberry，输入后回车就进入了树莓派。在命令行里输入 sudo raspi-config 即可打开树莓派设置界面。一次操作： Interfacing Options ，VNC ，Yes。操作完成后即可打开 VNC 设置。图 5-3 展示了树莓派在 MobaXterm 程序里的远程终端界面。在图中可以看到在终端左边，MobaXterm 还支持实时查看终端文件十分方便。

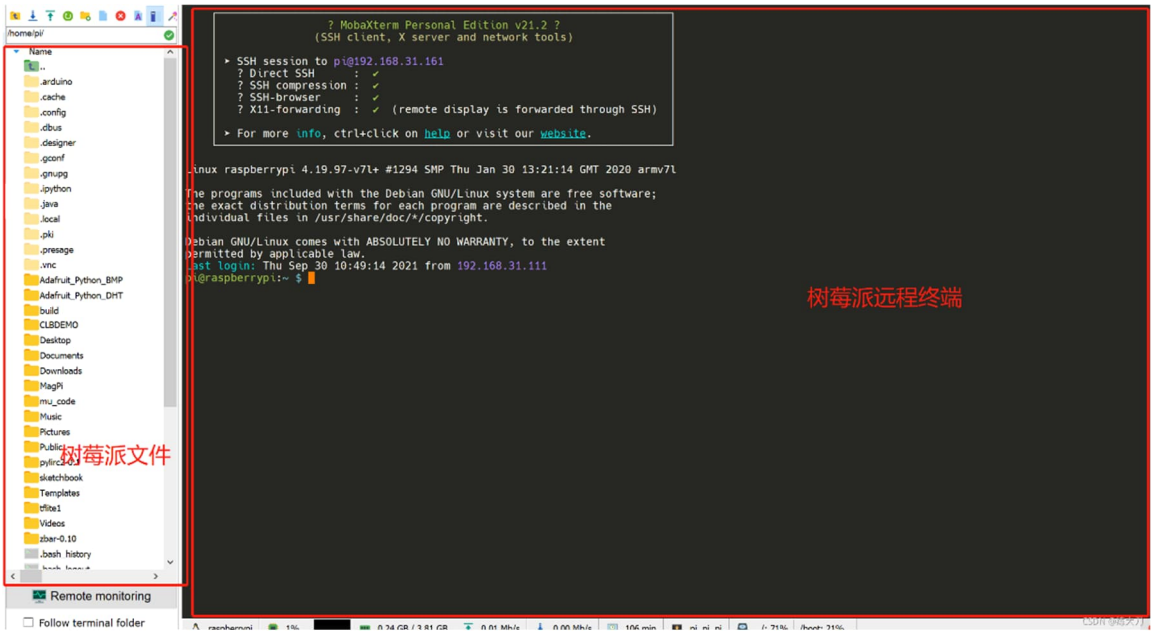

图 5-3 树莓派远程终端界面

之后再用 MobaXterm 新建 RDP 连接，配置同新建 SSH 连接。RDP 连接成功后即可成功进入树莓派图形操作界面

5.2.4 OpenCV 环境搭建

OpenCV 环境的搭建十分简单，只需简单几行 Linux 命令即可完成。

首先，在终端下输入 sudo nano /etc/apt/sources.list.d/raspi.list 命令将树莓派的下载源设置为清华源，提升下载速度。之后，在终端里输入 sudo apt-get update 将树莓派进行升级。升级完成后在输入 sudo apt-get install python-opencv 即可完成树莓派的安装。

5.3 捡球小车与客户端通信设计

捡球小车的这个功能是通过套接字（socket）实现的，能够让用户使用手机 APP 控制捡球小车的运动并且能够通过手机屏幕实时监控捡球小车前方车况[13]。socket 是计算机之间一种通信的方式，通过 socket 能够实现不同计算机之间的信息发送与接收。在 Python 中，socket 已经被集成为了库函数，方便开发者直接调用，大大节约程序的开发时间。要想实现两台设备之间的通信就需要分别为客户端和服务端设计程序。这里选择手机作为客户端，向捡球小

车发送指令；捡球小车作为服务端。

```
import threading
import socket
socket_port = 1314
socket_get_ok = 0
socket_receive_buf = ""
```

首先设计作为服务端的捡球小车程序。引入了 threading 和 socket 库。将 1314 端口设置为 socket 通信端口，并且将接收到的信息存放在 socket_receive_buf 变量里，变量 socket_get_ok 则是用于判断是否有数据接收。一般情况下，一个进程最少拥有一个线程，当存在多个线程的情况下，就需要设计一个主线程。同一个进程的所有线程共用系统资源，但是每个线程都拥有属于自己的栈和寄存器。捡球小车与手机 APP 的通信引入多线程可以大大提高系统的处理效率，减少服务端和客户端数据收发的时延。

```
def SocketEvent():
    global socket_receive_buf, socket_get_ok, socket_port
    serverSocket = socket.socket()
    hostIP = "192.168.12.1"
    print('hostIP:', hostIP)
    dataPort = socket_port
    address = (hostIP, dataPort)
    serverSocket.bind(address)
    serverSocket.listen(5)
    print('socket waitting.....')
    serverConnect, addr = serverSocket.accept()
```

```
    print('socket connected!')
```

这里定义了一个名为 SocketEvent 的函数。函数开头先创建了一个套接字，将网络地址设置为 192.168.121。serverSocket.listen(5)将超过限制数量的连接之前，捡球小车可以挂起的最大连接数量设置为 5，可以简单的理解为捡球小车的最多只可以连接 5 个手机 APP。

```
    while True:
        data = str(serverConnect.recv(1024))
        data = data.split('\"')[1]
        if len(data):
            print(data)
            socket_receive_buf = data
            if socket_get_ok == 0:
                uart_receive_buf = socket_receive_buf
                uart_get_ok = 0
                if mode == 0:
                    if uart_receive_buf.find('{') >= 0:
                        mode = 1
                    elif uart_receive_buf.find('$') >= 0:
                        mode = 2
                    elif uart_receive_buf.find('#') >= 0:
                        mode = 3
                if mode == 1:
                    if uart_receive_buf.find('}') >= 0:
                        uart_get_ok = 1
```

```
                mode = 0
            elif mode == 2:
                if uart_receive_buf.find('!') >= 0:
                    uart_get_ok = 2
                    mode = 0
            elif mode == 3:
                if uart_receive_buf.find('!') >= 0:
                    uart_get_ok = 3
                    mode = 0
            socket_get_ok = uart_get_ok
        else:
            print('socket disconnected.')
            serverSocket.listen(5)
            print('socket waitting.....')
            serverConnect, addr = serverSocket.accept()
            print('socket connected!')
    et_thread = threading.Thread(target=socketEvent)
```

这段程序是作用是将捡球小车收到的信息进行处理，用到的方法有 str、split 和 find。首先将捡球小车收到的信息使用 str 方法将其转换为字符串类型，在按照“\’”将其分割成多个字符串，这些字符串会保存在一个列表中，在传递给 socket_receive_buf 存储。之后会进行一系列的判断过程，通过 find 查找到的关键字来选择捡球小车的工作模式。

手机端 APP 设计则是使用了 Android 进行开发，通过 socket 与捡球小车通信,通过 surfaceview 读取捡球小车摄像头的视频流。由于工程文件太多，这里只展示手机端 APP 的

最后实现效果。图 5-4 展示了手机 APP 的连接配置页面。

图 5-4 手机端 APP 连接配置界面

首先将手机连接到捡球小车发出的热点，APP 默认配置好了能与捡球小车通信的正确网络地址、视频端口和数据端口。用户还可以通过程序右上角的安卓图标更改按键配置，方便用户设置为自己的习惯按键位置。同时还可以更改按键发出的串口指令。图 5-5 展示了程序运行界面。

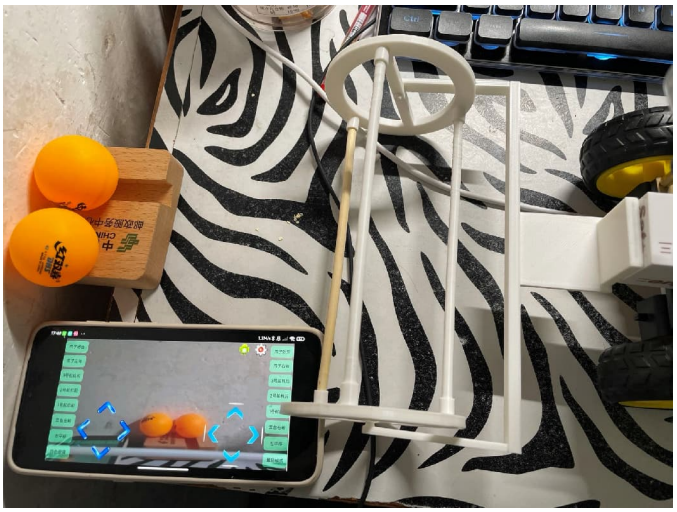

图 5-5 运行界面展示

图 5-6 用程序流程图演示手机 APP 如何通过 socket 与捡球小车进行通信，进行实时图

传与控制。

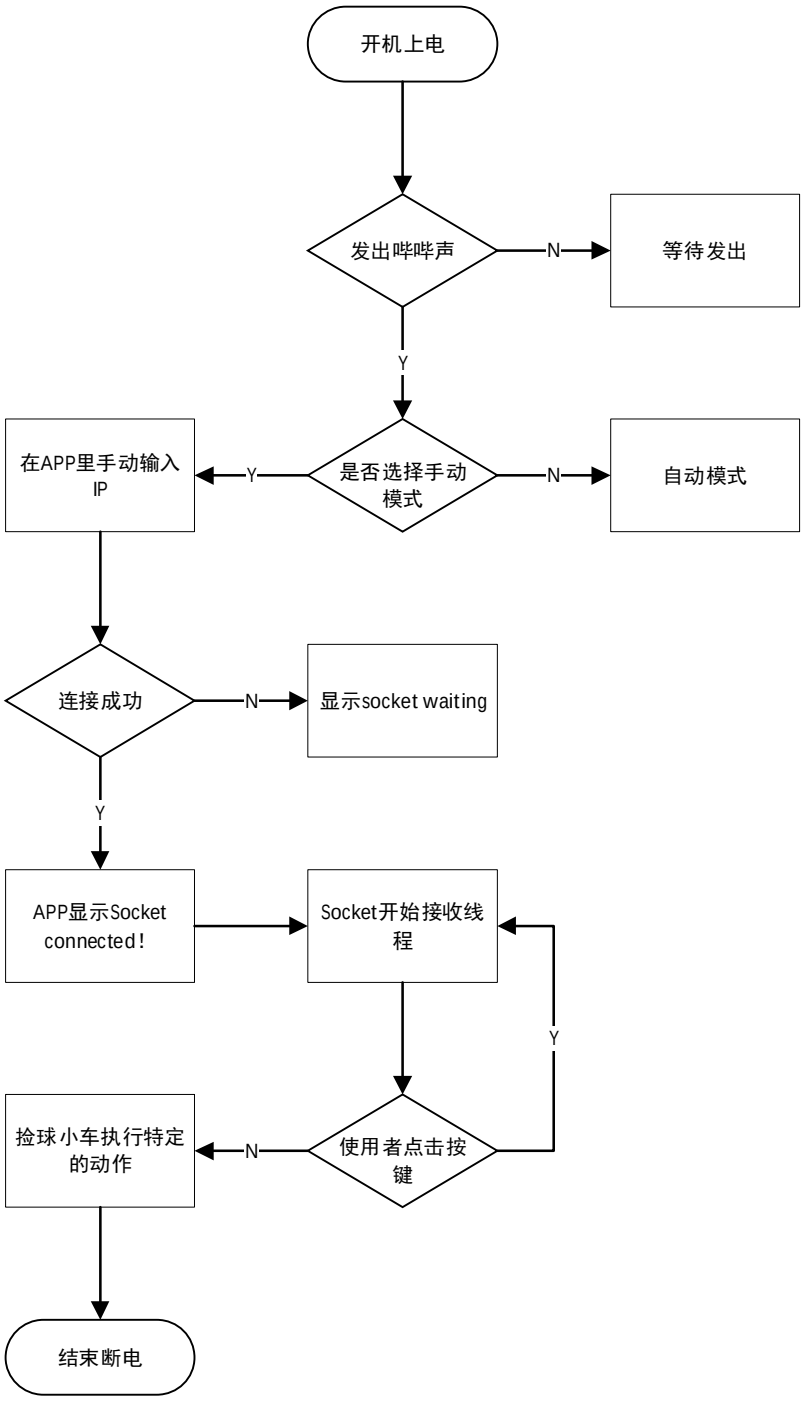

图 5-1 socket 通信流程图

5.4 乒乓球识别程序

乒乓球识别使用到了 NumPy 和 OpenCV 库。Numpy[14]为 Python 科学计算中的基本工

具，可用于矩阵的存储与处理，而在此程序中用于颜色阈值的设定;OpenCV 为计算机视觉中常用的库，并提供 Python 接口，可使开发者用 Python 语言来调用 C/C++,既能保证运行效率又能提高程序的阅读性。

OpenCV 为 HoughCircles()提供了一种检测图像圆形的方法，也称为霍夫圆环变换（Holf Transfer Transform）[15]。霍夫圆环变换在检测过程中会有两次检测，第一次检测发现可能为圆形的圆心坐标,第二次检测计算这些圆心坐标的半径长度,最后将两者储存于圆心坐标与半径长度所组成的浮点型数组中。霍夫圆形变换为开发者提供了大量的参数设置，尽管开发者使用起来会很方便，但随之而来的坏处就是局限性大，无法为捡球小车提供适合不同环境下场地的快速调参。而且很难迅速、准确地获得这些数据。如果要让捡球能够准确地识别到乒乓球并定位它就必须解决这个问题。

为此，本论文提出了一种新方法，不仅可以做到识别率高，而且能够让无任何编程基础的用户快速进行参数，达到最优的识别效果。

这个新方法的核心是通过先对图像预处理降低输入数据量，然后利用 HSV 颜色模型[16]及边缘检测技术[17]对特征进行匹配。

新方法里包括设计一个名为 hsv.py 程序，此程序的功能是对摄像头捕捉到的画面进行图像进行预处理，调试各种画面和窗口参数，达到最优的识别效果。此程序的优点在于用户只用拖动滑条即可完成调参，不需要用户需要专业的知识。

图 5-7 展示了乒乓球识别程序的流程图。在流程图中可以看到乒乓球识别程序有两大分支，一个是对摄像头捕捉到的画面进行预处理，窗口的设置，利用前一个程序得到的参数，在进行空间色彩变换、边缘检测等操作识别到乒乓球。另一个是创建滑条、设置每个滑条代表的参数和参数的变化范围和将设置好的滑条显示出来。

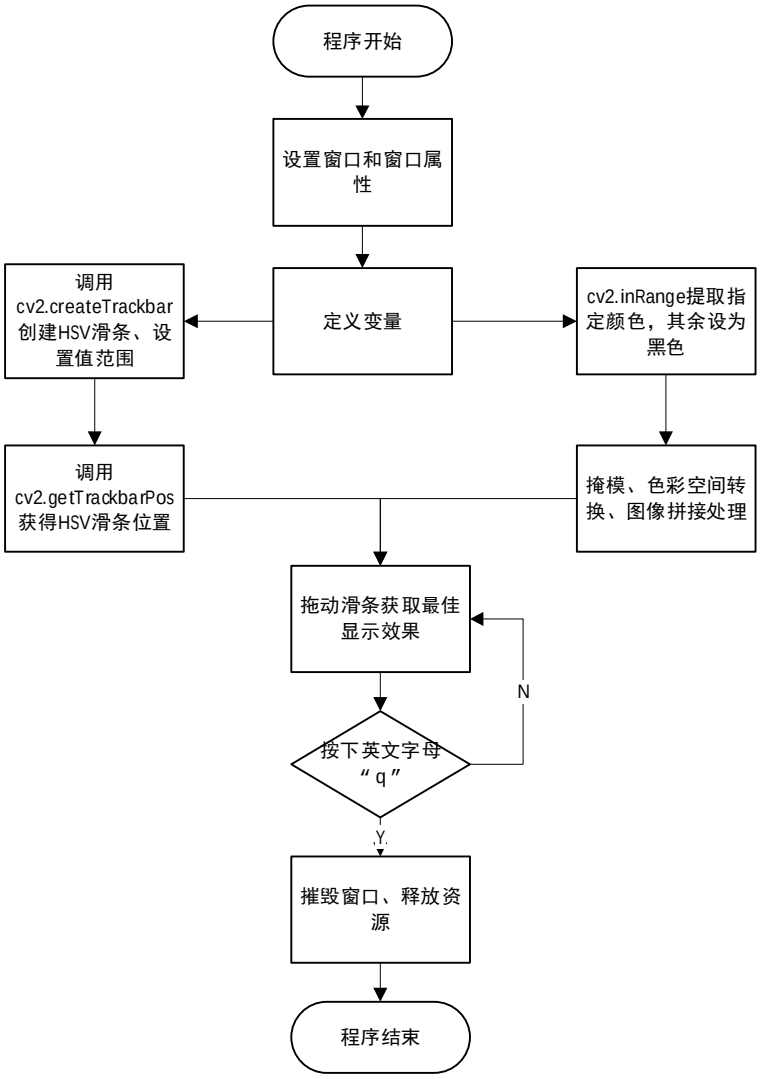

图 5-7 乒乓球识别程序流程图

```
import cv2
import numpy as np
frameWidth = 960
frameHeight = 540
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
```

```
cap.set(10, 50)
pulse_ms = 30
lower = np.array([13, 96, 119])
upper = np.array([23, 255, 255])
```

导入需要用到的 opencv 和 numpy 库。frameWidth 和 frameHeight 代表着显示窗口的行列像素，通俗说就是窗口的大小。使用 VideoCapture 方法初始化摄像头，0 代表摄像头的索引。set 中，这里的 3，下面的 4 和 10 是类似于功能号的东西，数字的值没有实际意义。cap.set(10,50)则是设置了亮度。lower 和 upper 则是参照了 HSV 色彩空间内黄色的颜色范围，初始化了一个预设值。

```
def empty(a):
    pass


cv2.namedWindow("HSV")
cv2.resizeWindow("HSV", 640, 300)
cv2.createTrackbar("HUE Min", "HSV", 13, 179, empty)
cv2.createTrackbar("SAT Min", "HSV", 96, 255, empty)
cv2.createTrackbar("VALUE Min", "HSV", 119, 255, empty)
cv2.createTrackbar("HUE Max", "HSV", 23, 179, empty)
cv2.createTrackbar("SAT Max", "HSV", 255, 255, empty)
cv2.createTrackbar("VALUE Max", "HSV", 255, 255, empty)
```

这段代码的作用是产生控制滑条。使用 namedWindows 创建了一个叫 HSV 的窗口并将窗口尺寸设为 640*300。使用 createTrackbar 创建 6 个滑条分别代表着 HUE Min、SAT Min、VALUE Min、HUE Max、SAT Max 和 VALUE Max。HUE 代表着色调、SAT 代表着饱和度、

VALUE 代表着明亮度。这里使用 HSV 而不是 RGB 的原因是，HSV 是面向用户的，RGB 是面向硬件的。createTrackbar 的第五个参数是执行回调函数，回调函数始终具有默认参数，即轨迹栏位置。但是在捡球小车中，回调函数什么都不做，所以定义一个 empty 函数使用 pass 简单地通过即可。

```
h_min = cv2.getTrackbarPos("HUE Min", "HSV")
h_max = cv2.getTrackbarPos("HUE Max", "HSV")
s_min = cv2.getTrackbarPos("SAT Min", "HSV")
s_max = cv2.getTrackbarPos("SAT Max", "HSV")
v_min = cv2.getTrackbarPos("VALUE Min", "HSV")
v_max = cv2.getTrackbarPos("VALUE Max", "HSV")
lower = np.array([h_min, s_min, v_min])
upper = np.array([h_max, s_max, v_max])
```

这段程序的功能是获取滑条的位置，并且将这些值分别赋值给 h_min、h_max、s_min、s_max、v_min、v_max，再将这六个值传递给 lower 和 upper。这样就可以实现用过通过控制滑条来控制画面只显示乒乓球。

```
_, img = cap.read()
imgHsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
imgMask = cv2.inRange(imgHsv, lower, upper)
imgOutput = cv2.bitwise_and(img, img, mask=imgMask)
imgMask = cv2.cvtColor(imgMask, cv2.COLOR_GRAY2BGR)
imgStack = np.hstack([img, imgOutput])
cv2.namedWindow("imgMask", 0)
cv2.resizeWindow("imgMask", 800, 400)
```

```
    cv2.imshow('imgMask', imgStack)
    if cv2.waitKey(pulse_ms) & 0xFF == ord('q'):
        print("Quit\n")
    break
```

这段程序则是显示摄像头得到的画面进行处理和显示。cv2.inRange 方法可以提取用户想要的的颜色，这里的效果是只显示用户调试好的最佳乒乓球识别颜色，其余的位置显示为纯黑色。为了用户在调试完毕后可以直接退出，在程序的最后设置了按键检测，只要用户按下英文输入下的“ q ”键就可以退出程序。图 5-8 展示了程序运行结果。

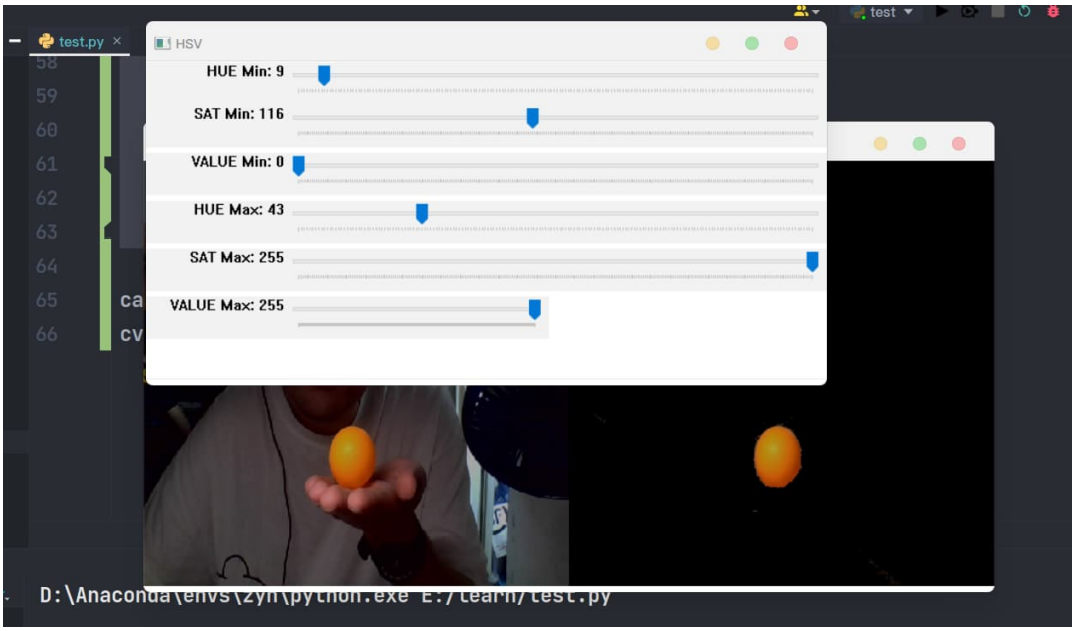

图 5-8 程序运行效果

图 5-6 展示了 hsv.py 程序运行的实时画面。从图中可以看出用户只需简单的拖动滑条，就可以让摄像头里只展示乒乓球。

```
    cnts = cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
    x_bias = 0
```

```
    y_bias = 0
    area = 0
    if len(cnts) >0 :
        cnt = max(cnts,key=cv2.contourArea)
        rect = cv2.minAreaRect(cnt)
        box = cv2.boxPoints(rect)
        c_x, c_y = rect[0]
        c_h, c_w = rect[1]
        c_angle = rect[2]
        if c_angle<-45:
            c_angle = -(90+c_angle)

        if 1 < c_h < 180 and 1 < c_w < 180:
            cv2.drawContours(frame, [np.int0(box)], -1, (0, 255, 255), 2)
            x_bias = c_x - width/2
            y_bias = c_y - hight/2
            area = c_h*c_w
    cv2.imshow('frame',frame)
```

这段程序在前面调试好参数的情况下，通过 cv2.findCountours()方法来计算图像梯度来判断出图像的边缘，再通过 cv2.drawContours()方法来绘制出轮廓。在这段程序之前，会对图像进行预处理。通过 cv2.erode()方法对图像进行腐蚀，强化图像的细节；cv2.GaussianBlur()方法进行高斯模糊，抑制图像中的噪声。

图 5-9 展示了乒乓球识别程序的最终识别效果。从图中可以明显的看到，乒乓球识别程

序不仅准确识别到了图像中的乒乓球，而且画出了乒乓球的轮廓和圆心。

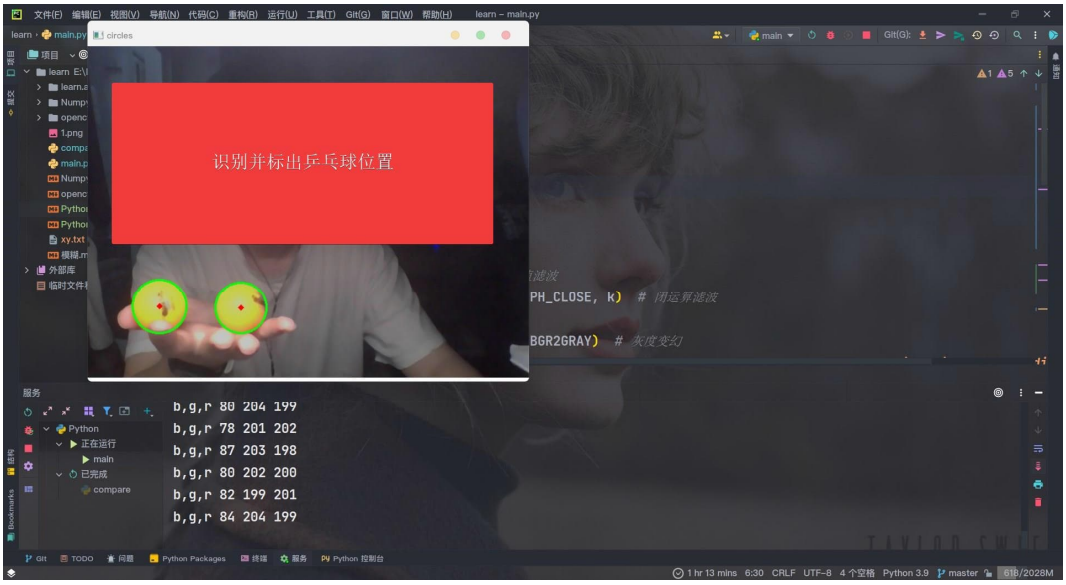

图 5-2 乒乓球识别最终效果

5.5 捡球小车跟踪程序

捡球小车跟踪程序需要前面乒乓球识别程序的配合，依靠准确识别出的乒乓球轮廓来指引捡球小车行驶。

```
    def car_run(speed_l1,speed_r1,speed_l2,speed_r2):
        textSrt = '#006P{:0>4d}T0000!#007P{:0>4d}T0000!#008P{:0>4d}T0000!#009P{:0>4d}T0000!'.format(speed_l1,speed_r1,speed_l2,speed_r2)
        print(textSrt)
        myUart.uart_send_str(textSrt)


    def car_go_back(speed):
        car_run(1500+speed,1500-speed,1500+speed,1500-speed)


    def car_left_turn(speed):
```

```
    speedl = 1500+speed*2//3
    speedr = 0
    car_run(speedl,speedr,speedl,speedr)

def car_right_turn(speed):
    speedl = 0
    speedr = 1500+speed*2//3
    car_run(speedl,speedr,speedl,speedr)

def car_stop():
    myUart.uart_send_str('#255P1500T1000!')
```

这段程序定义了四个函数，分别是 car_run、car_go_back、car_left_turn、car_rignt_turn、car_stop，可以实现捡球小车前进、后退、左转、右转和停止。变量 textSrt 代表的是向总线双路驱动模块发送的指令，接收到特定的指令，总线双路驱动模块会驱动捡球小车的 TT 马达执行特定的动作。总线双路驱动模块的通讯协议在第四章做了详细阐述。通过调用不同的函数，捡球小车会执行特定的动作。

图 5-10 展示了捡球小车追踪程序流程图，并按照流程图所给思路设计捡球小车追踪代码。

从流程图中可以直观的看出，捡球小车的追踪程序涉及到了很多的 if 条件判断，满足不同的条件，捡球小车会执行特定的动作。流程图里的固定线程运行时间有两个作用：一是发送串口数据要等待处理；二是防止一直树莓派往串口发数据到时捡球小车会出现抖动运行的情况。

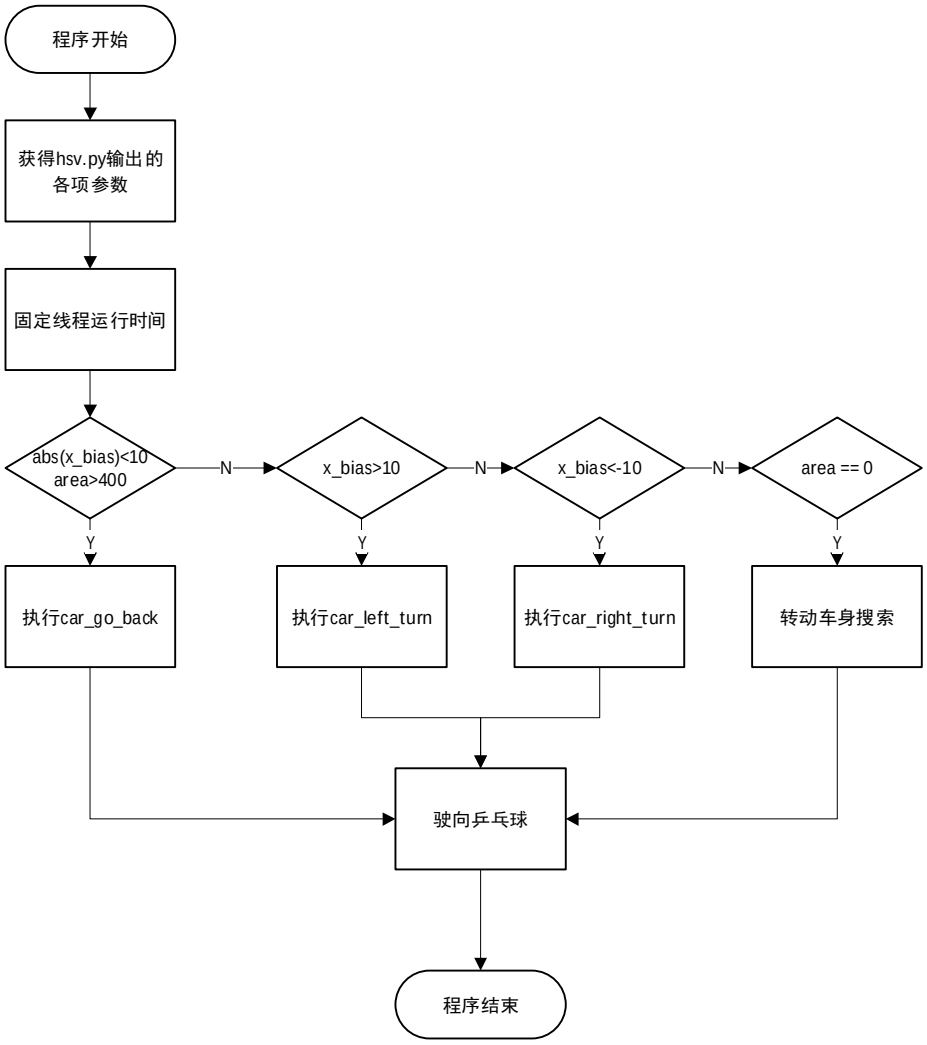

图 5-10 捡球小车跟踪程序工作流程

```
def car_follow():
    global systick_ms_bak,next_time,x_bias,y_bias,area
    while True:
        if int((time.time() * 1000))- systick_ms_bak >= int(next_time):
            systick_ms_bak = int((time.time() * 1000))
            if abs(x_bias) < 10 and area > 400:
                car_go_back(400)
                next_time = 0
```

```
        elif int(x_bias) > 10:
            next_time = 0
            interval_time = x_bias/100
            car_left_turn()
        elif int(x_bias) < -10:
            next_time = 0
            interval_time = -x_bias/100
            car_right_turn()
        elif area == 0:
            car_left_turn()
            time.sleep(0.5)
            car_stop()
            time.sleep(1)
            car_right_turn()
            time.sleep(0.5)
            car_stop()
```

这段程序是捡球小车的运行控制逻辑。程序会通过比较识别到的乒乓球的圆心和屏幕中点坐标的横坐标得到差值并且赋值给变量 x_bias。在满足 x_bias 绝对值小于 10 和 area 大于 400 的条件下，捡球小车才会驶向识别到的乒乓球；如果乒乓球圆心横坐标和屏幕中点横坐标超过 10，捡球小车分别会左转和右转；如果 area 的值为 0 则表示摄像头没有识别到乒乓球，那么捡球小车会左转再进行识别。其中的 time.sleep(n)则是代表程序等待 n 秒的时间再接着往下执行，等待的时间留给乒乓球识别程序进行识别。

5.6 避障程序

捡球小车的避障程序分为两个部分，分别是超声波避障程序和碰撞传感器避障程序。二者相互搭配实现捡球小车避障功能。

图 5-11 展示了捡球小车的避障流程图。从图中可以看出避障程序是一个中断程序，在发生碰撞的时候触发中断。

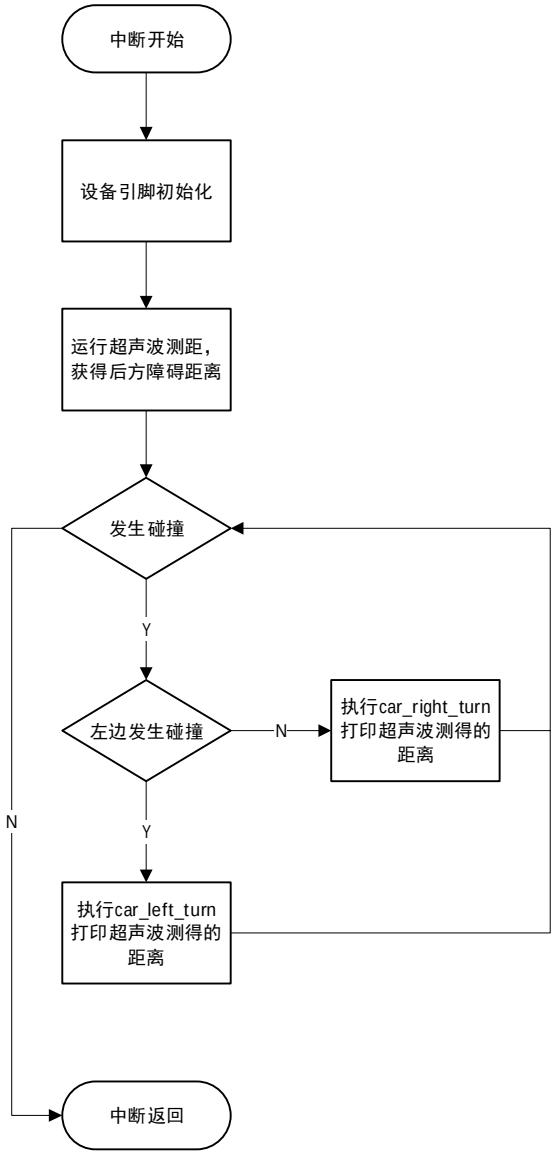

图 5-11 避障程序流程图

```
def setup():
```

```
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(TRIG, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN, pull_up_down = GPIO.PUD_UP)


def distance():
    err = 0
    GPIO.output(TRIG, 0)
    time.sleep(0.000002)

    GPIO.output(TRIG, 1)
    time.sleep(0.00001)
    GPIO.output(TRIG, 0)
    time0 = time.time()
    while GPIO.input(ECHO) == 0 and time.time()-time0<0.1:
        a = 0
    if time.time()-time0 >= 0.1:
        err = 1
        return 0

    time1 = time.time()
    while GPIO.input(ECHO) == 1  and time.time()-time1<0.1:
        a = 1
```

```
        if time.time()-time1 >= 0.1:
            err = 1
            return 0

        time2 = time.time()

        during = time2 - time1
    return during * 340 / 2 * 100
```

这段程序用来控制超声波传感器，获得小车与后方障碍物的距离。获得的距离用来辅助碰撞程序判断。

```
def collision_detection_left():
    if(devValue_left() == 0):
        time.sleep(0.02) #qu dou dong
        if(devValue_left() == 0):
            print(">Dev get Low")
            beep_on()

            while(devValue_left() == 0):
                car_left_turn()
                dis = distance()
                print ('dis:%d cm'%dis)
                time.sleep(0.1)
```

```
            beep_off()
            print(">>>Dev get High")

    def collision_detection_right():
        if(devValue_right() == 0):
            time.sleep(0.02) #qu dou dong
            if(devValue_right() == 0):
                print(">Dev get Low")
                beep_on()

                while(devValue_right() == 0):
                    car_right_turn()
                    dis = distance()
                    print ('dis:%d cm'%dis)
                    time.sleep(0.1)
                beep_off()
                print(">>>Dev get High")
    setup_dev()
```

这里定义了两个函数分别叫做 collision_detection_left 和 collision_detection_right，对应的是捡球小车左边和右边的碰撞传感器。在函数中 time.sleep()的功能是用用于来去抖动，防止碰撞传感器机械设计以及器件老化所带来的问题。如果碰撞传感器被触发（以左碰撞传感器为例），集成在拓展板上的蜂鸣器会发出哔哔声，同时捡球小车会逆时针旋转，确保不会

再次发生碰撞。此外，在程序终端还会打印出后方障碍物的距离。

5.7 换点搜索程序

换点搜索程序可以对捡球小车的结果进行全局解，能够保证场馆内所有区域都会被捡球小车摄像头扫描到，不会遗漏被大型障碍物遮挡的后方区域。

换点搜索程序的实现十分简单，如果捡球小车旋转一圈都没有找到乒乓球，那么捡球小车就会寻找摆在场馆内部固定位置的红色方块。

```
while(~red_block):
    collision_detection_left()
```

只要没搜索到红色方块就持续调用 collision_detection_left()，直到找到为止。红色方块的识别程序原理和乒乓球识别程序原理相同，这里不重复赘述。图 5-12 展示了捡球小车的换点搜索程序流程图。

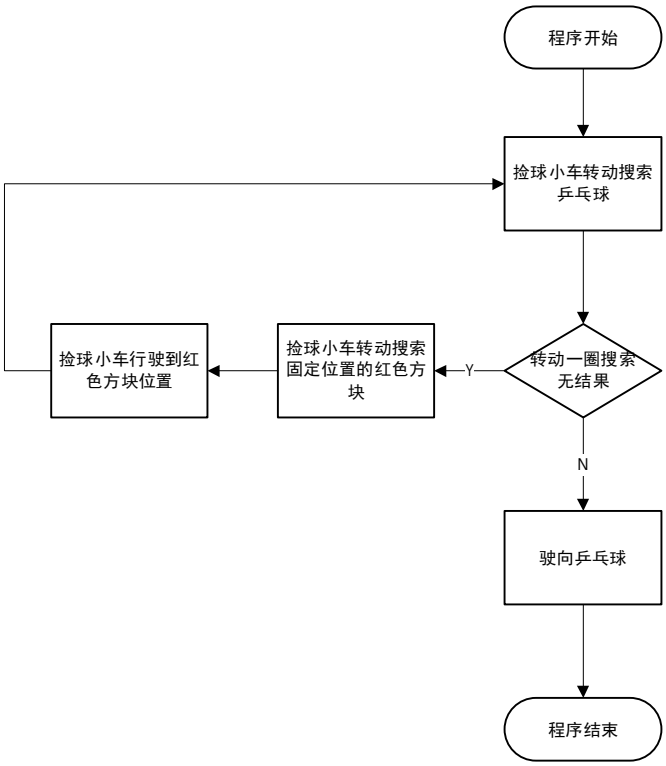

图 5-12 换点搜索程序流程图

5.8 串口程序

串口程序（z_uart）用来将前面程序产生的串口信息发送到双路总线驱动模块，只需要在前面的程序导包是导入 z_uart 即可。由于串口程序篇幅太长，这里只展示核心代码。

```
def loop_uart():
    global uart_get_ok, uart_receive_buf
    if(uart_get_ok):
        print(int(time.time()*1000))
        uart_send_str(uart_receive_buf)

        if uart_receive_buf == '$LEDON!':
            myLed.on(1)
        elif uart_receive_buf == '$LEDOFF!':
            myLed.off(1)
        elif uart_receive_buf == '$BEEPON!':
            myBeep.on()
        elif uart_receive_buf == '$BEEPOFF!':
            myBeep.off()

        uart_receive_buf = ''
        uart_get_ok = 0

if __name__ == '__main__':
    setup_uart(115200)
```

```
        myLed.setup_led()
        myBeep.setup_beep()
        myBeep.beep(0.1)
        myBeep.beep(0.1)
        myBeep.beep(0.1)
        try:
            while True:
                loop_uart()

        except KeyboardInterrupt:
            if ser != None:
                ser.close()
            myLed.off(1)
            myLed.off(2)
    myBeep.off()
```

因为捡球小车在运行时会持续产生串口信息，所以串口程序的核心要义是不断接收和发送串口信息，所以程序里有大循环确保串口信息可以传递给双路总线驱动模块。

5.9 实物测试

完成捡球小车的硬件设计和程序编写以及调试后，接下来对捡球小车进行实物测试。由于捡球小车的实物测试应由视频展示，论文只能放静态图片，所以本节只展示能以静态图片表达的测测样图，完整的测试过程将以视频的形式单独展示。

图 5-13 展示的是捡球小车的整体实物图。从图中可以清楚地看到捡球小车的整体构造和部件布局。

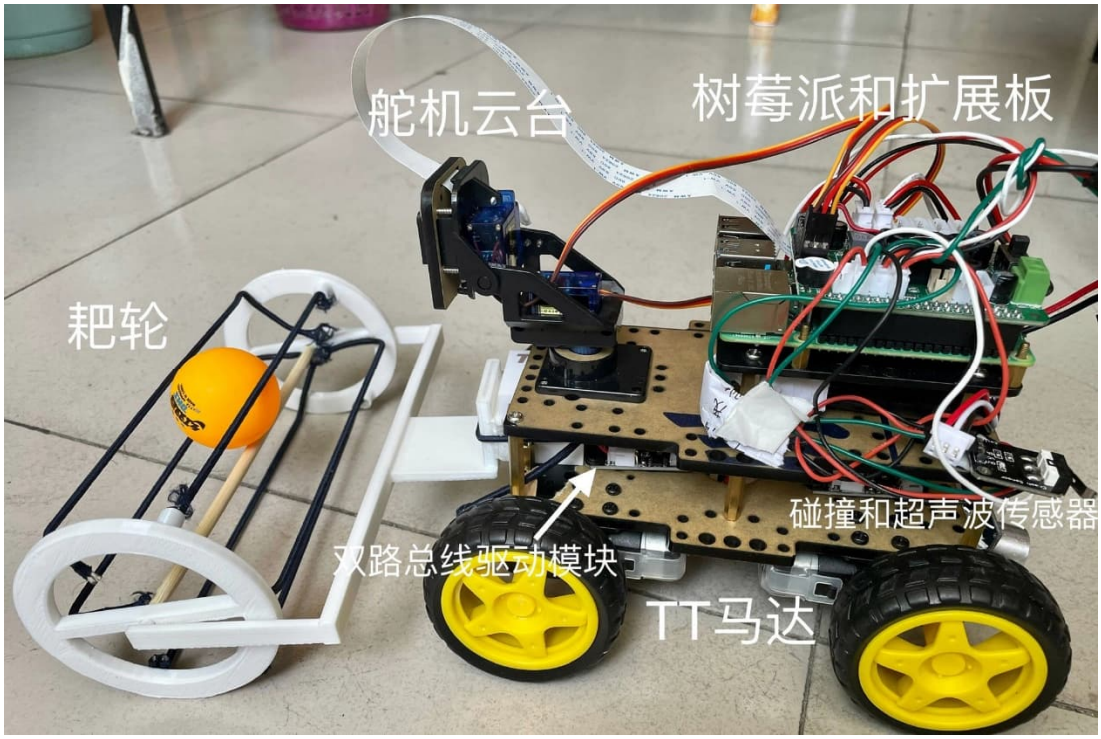

图 5-13 捡球小车实物图

为了方便乒乓球识别跟踪和避障程序的演示，下面展示的图片里将暂时耙轮拆除和将捡球小车托起。

图 5-14 展示的是捡球小车的乒乓球追踪程序的测试图。从图中的车轮残影可以看出捡球小车正在朝乒乓球的方向行驶。

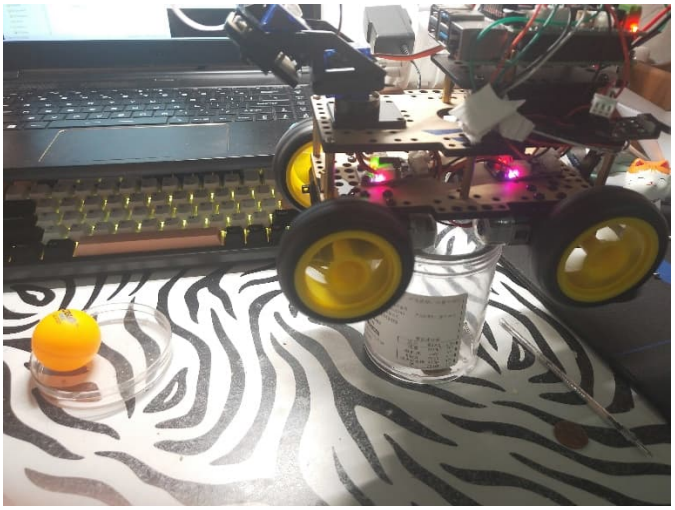

图 5-14 乒乓球追踪程序测试图

图 5-15 展示的是捡球小车避障程序，这里以右碰撞传感器被触发为例。从图中可以清楚地看到，当右碰撞传感器被触发后，捡球小车右边车轮将停止转动，左边车轮将持续转动，超声波传感器将在终端输出捡球小车与后方障碍的距离，进行辅助判断。图 5-16 展示的是超声波传感器的输出结果。

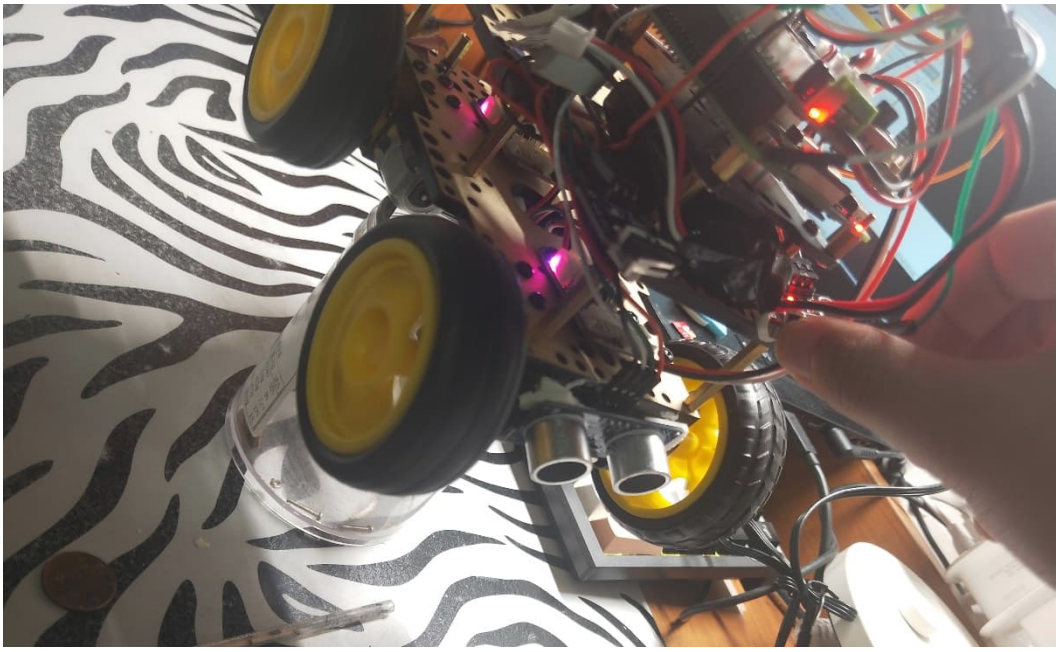

图 5-15 避障程序测试图

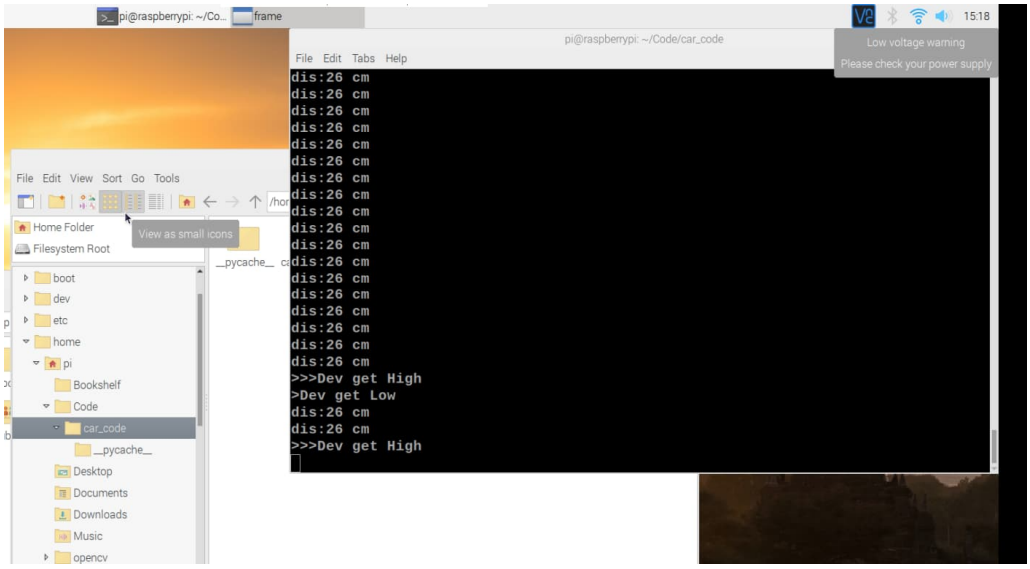

图 5-16 超声波程序测试图

图 5-16 展示的是手机 APP 的控制界面以及和捡球小车的实时图传画面。

图 5-16 手机 APP 画面

从图 5-16 里还可以看出，APP 预留了很多自定义按钮，方便用户自己设置，非常的人性化。

关于换点的程序图和完成程序的测试，将会以视频的形式进行展示。

6 结论

本次针对乒乓球的捡球小车设计参考了大量关于树莓派开发、硬件控制、机器视觉、图像处理的文献，研究了有关于目标识别、跟踪和定位技术的算法。其中重点研究了物体检测以及物体跟踪算法。舍弃 OpenCV 所直接提供的霍夫圆形检测方法，设计出一种可应用于多种情况且能零基础调参的乒乓球识别算法。该算法采用基于颜色信息的模板匹配法进行目标分割；根据不同场景下乒乓球与背景的差异，并通过阈值化处理图像提取目标区域，并通过边缘检测最终识别出环境里的乒乓球。经过实验验证，本文提出的算法具有很好的实时性和准确性。

本论文完成的工作如下：

1）完成了捡球小车的机械结构设计。通过捡球小车合理的机械结构保证了捡球小车不会出现部件连接不稳、车身摇晃、转动困难等不良现象。

2）完成了捡球小车的控制方案设计，并以此完成了硬件选型。捡球小车使用的硬件可以使它平稳顺畅地运行。使用是树莓派 4B 作为主控制器，凭借其可观的性能，保证了捡球小车不会出现宕机和算力不够的情况

3）对各功能进行程序编写和调试。通过不断的修改程序和进行样机验证，最终达到了捡球小车完美识别并捡球乒乓球的效果。其中，关于乒乓球识别程序做出了创新，提出了一套新的识别算法。

参考文献

[1] 张行，石晶，丁华锋，等. 新型自主乒乓球收集机器人设计研发[J]. 机械研究与应用，2019，32(05)：143-145.

[2] 赵雪川，李卫国，王利利. 涵道捡乒乓球机器人的设计与实现探究[J]. 科技创新与应用，2018，(17):83-84.

[3] 安丹阳. 新型自主式乒乓球捡球机器人的构形及关键技术研究[D]. 沈阳：东北大学，2008.

[4] 林广茂，王天雷，招展鹏，等. 基于视觉识别的全自动网球拾取机器人设计[J]. 机电工程技术，2017，46(3)：5.

[5] 程鹏,周朱德,陈章宝.视觉导航网球捡球机器人控制系统设计[J].工业控制计算机,2019,32(03):120-121.

[6] 王志虎，沈小青，桂伟龙. 光学成像小目标检测技术综述[J]. 现代防御技术,2020，48(05)：67-73.

[7] 刘丽，赵凌君，郭承玉，王亮，汤俊. 图像纹理分类方法研究进展和展望[J]. 自动化学报，2018，44(04)：584-607.

[8] 蒲兴成，李俊杰，吴慧超，张毅. 基于改进粒子群算法的移动机器人多目标点路径规划[J]. 智能系统学报，2017,12(03)：301-309.

[9] 张贵阳，薛牧遥，朱子健，霍炬. 基于多适应值全参数自主变异粒子群的立体相机标定[J]. 系统仿真学报，2020，32(12)：2461-2468.

[10] ZAFAR Mohd N, MOHANTA J C. Methodology for Path Planning and Optimization of Mobile Robots: A Review[J]. Procedia Computer Science, 2018, 133: 141-152.

[11] TUNCER A, YILDIRIM M. Dynamic path planning of mobile robots with improved genetic algorithm[J]. Computers & Electrical Engineering, 2012, 38(6): 1564-1572.

[12] DEWANG H S, MOHANTY P K, KUNDU S. A Robust Path Planning For Mobile Robot Using Smart Particle Swarm Optimization[J]. Procedia Computer Science, 2018, 133: 290-297.

[13] 马睿，王荣，魏峻超. 一种基于树莓派和 Socket 通信的视频小车设计方案[J]. 电脑编程技巧与维护，

2020(12): 129-130.

[14] 王志强.基于霍夫变换的图像轮廓检测方法与优化[J].哈尔滨师范大学自然科学学报,2021,37(03):79-82.

[15] 王明迁, 李丹阳, 郝威凯, 李建国. 基于 HSV 颜色模型的图像识别技术研究[J]. 科技资讯, 2020, 18(35): 1-2+8.

[16] 徐武, 张强, 王欣达, 高寒, 秦浩然. 基于改进 Canny 算子的图像边缘检测方法[J]. 激光杂志, 2022, 43(04): 103-108.

致谢

时间总是在不经意间流过，大学四年的时光也非常短暂。来辽石化的四年，让我见识到了许多，也让我认识到了许多。感谢学校给我提供的平台，让我有机会接触到不同时代不同风格和不同类型的人，感受他们身上所蕴含的精神内涵，从而获得心灵上的启迪和情感上的满足。

在这里我要表达我对我的指导王越老师的感谢。桃李不言下自成蹊，春风化雨润物无声是我对老师的第一感觉。从毕设的开始到结束，老师一直都在亲力亲为，指导我完成各项工作，也让我从中学到了很多。希望我在读研期间也可以遇上像王越老师一样负责、春风化雨的导师。

月光还是少年的月光，九州一色还是李白的霜。月与雪色之间，你是第三种颜色。和她一起走过四年，四年的时光里，她让我的心也渐渐变了颜色，变得柔软而温暖。在那个夜晚我的心情像一个孩子一样快乐着，我的世界被一片雪花覆盖着，我的心里只有一片雪，我要把我所有的爱都给她,她是那么可爱！她的美丽就好像她的名字那样美丽！她是那么纯洁就像她名字那样纯洁！

希望在我以后的求学时光里，能够深耕于自己喜欢的领域，做自己热爱的事情，爱自己爱的人，呵护好我的她。

附录

carFollow.py

```
import cv2
import numpy as np
import time
import threading
import RPi.GPIO as GPIO
import pigpio
import os

import z_uart as myUart
import z_beep as myBeep
from math import *

TRIG = 17
ECHO = 18
os.system('./killmain.sh')
text_lower = np.array([13, 96, 119])
text_upper = np.array([23, 255, 255])
color_lower = text_lower
color_upper = text_upper

PIN_yuntai = 26
```

```
PIN_camera = 12


PIN_pengzhuang1 = 23
PIN_pengzhuang2 = 3

BEEP_PIN = 21


c_x = 160
c_y = 120
x_bias = 0
y_bias = 0
area = 0
next_time = 50
pwm_value1 = 1400
pwm_value2 = 1900

systick_ms_bak = 0
pi = pigpio.pi()

width = 320
hight = 240

cap = cv2.VideoCapture(0)
```

```
    cap.set(3,width)  #设置画面宽度
    cap.set(4,hight)  #设置画面高度

    myBeep.setup_beep()
    myUart.setup_uart(115200)

    #发出哔哔哔作为开机声音
    myBeep.beep(0.1)
    myBeep.beep(0.1)
    myBeep.beep(0.1)

    pi.set_servo_pulsewidth(PIN_yuntai, pwm_value1)
    pi.set_servo_pulsewidth(PIN_camera, pwm_value2)

    def setup():
        GPIO.setmode(GPIO.BCM)
        GPIO.setwarnings(False)
        GPIO.setup(TRIG, GPIO.OUT)
        GPIO.setup(ECHO, GPIO.IN, pull_up_down = GPIO.PUD_UP)


    def distance():
        err = 0
```

```
        GPIO.output(TRIG, 0)
        time.sleep(0.000002)

        GPIO.output(TRIG, 1)
        time.sleep(0.00001)
        GPIO.output(TRIG, 0)

        time0 = time.time()
        while GPIO.input(ECHO) == 0 and time.time()-time0<0.1:
            a = 0
        if time.time()-time0 >= 0.1:
            err = 1
            return 0

        time1 = time.time()
        while GPIO.input(ECHO) == 1  and time.time()-time1<0.1:
            a = 1

        if time.time()-time1 >= 0.1:
            err = 1
            return 0

        time2 = time.time()
```

```
    during = time2 - time1
    return during * 340 / 2 * 100


def destroy():
    GPIO.cleanup()
#csb chushihua
setup()    #csb setup
#初始化设备引脚
def setup_dev():
    GPIO.setup(PIN_pengzhuang1, GPIO.IN, pull_up_down = GPIO.PUD_UP)
    GPIO.setup(PIN_pengzhuang2, GPIO.IN, pull_up_down = GPIO.PUD_UP)

#读取 devValue 的值,触发为低电平
def devValue_left():
    return GPIO.input(PIN_pengzhuang1)

#读取 devValue 的值,触发为低电平
def devValue_right():
    return GPIO.input(PIN_pengzhuang2)

def destory():
    beep_off()
    GPIO.cleanup()
```

```
#蜂鸣器鸣叫
def beep_on():
    GPIO.output(BEEP_PIN, 1)
#蜂鸣器关闭
def beep_off():
    GPIO.output(BEEP_PIN, 0)

def collision_detection_left():
    if(devValue_left() == 0):
        time.sleep(0.02) #qu dou dong
        if(devValue_left() == 0):
            print(">Dev get Low")
            beep_on()

            while(devValue_left() == 0):
                car_left_turn()
                dis = distance()
                print ('dis:%d cm'%dis)
                time.sleep(0.1)

            beep_off()
            print(">>>Dev get High")
```

```
def collision_detection_right():
    if(devValue_right() == 0):
        time.sleep(0.02) #qu dou dong
        if(devValue_right() == 0):
            print(">Dev get Low")
            beep_on()

            while(devValue_right() == 0):
                car_right_turn()
                dis = distance()
                print ('dis:%d cm'%dis)
                time.sleep(0.1)

            beep_off()
            print(">>>Dev get High")
#pengzhuang chushihua
setup_dev()
D = threading.Thread(target = collision_detection_left)
D.setDaemon(False)
D.start()

E = threading.Thread(target = collision_detection_right)
```

```
E.setDaemon(False)
E.start()
#小车跟随
def car_follow():
    global systick_ms_bak,next_time,x_bias,y_bias,area
    while True:
        if int((time.time() * 1000))- systick_ms_bak >= int(next_time):
            systick_ms_bak = int((time.time() * 1000))

            #print(int(x_bias),int(y_bias),int(area))
            if abs(x_bias) < 10 and area > 400:
                car_go_back(400)
                next_time = 0
            elif int(x_bias) > 10:
                next_time = 0
                interval_time = x_bias/100
                car_left_turn()
            elif int(x_bias) < -10:
                next_time = 0
                interval_time = -x_bias/100
                car_right_turn()
            elif area == 0:
                car_left_turn()
```

```
            time.sleep(0.5)
            car_stop()
            time.sleep(1)
            car_right_turn()
            time.sleep(0.5)
            car_stop()
            time.sleep(1)

    """
    函数功能：串口发送指令控制电机转动
    范围：-1000～+1000
    '''
    def car_run(speed_l1,speed_r1,speed_l2,speed_r2):
        textSrt = '#006P{:0>4d}T0000!#007P{:0>4d}T0000!#008P{:0>4d}T0000!#009P{:0>4d}
T0000!'.format(speed_l1,speed_r1,speed_l2,speed_r2)
    #    print(textSrt)
        myUart.uart_send_str(textSrt)

    """
    函数功能：小车前进后退
    正值小车前进，负值小车后退
    范围：-1000～+1000
    '''
```

```
def car_go_back(speed):
    car_run(1500+speed,1500-speed,1500+speed,1500-speed)
"""
函数功能：小车左转
负值小车左转
范围：0～+1000
pwm 1500 == stop
'''
def car_left_turn():
    speed = 800
    speedl = 1500+speed*2//3
    speedr = 1500
    car_run(speedl,speedr,speedl,speedr)
"""
函数功能：小车右转
正值小车右转
范围：-1000～0
'''
def car_right_turn():
    speed = -800
    speedl = 1500
    speedr = 1500+speed*2//3
    car_run(speedl,speedr,speedl,speedr)
```

```
    """
    函数功能：小车停止
    '''
    def car_stop():
        myUart.uart_send_str('#255P1500T1000!')


    C = threading.Thread(target = car_follow)
    C.setDaemon(True)
    C.start()
    #无限循环
    while 1:
        collision_detection_left()
        #将摄像头拍摄到的画面作为 frame 的值
        ret,frame = cap.read()
        #高斯滤波 GaussianBlur() 让图片模糊
        frame = cv2.GaussianBlur(frame,(5,5),0)
        #将图片的色域转换为 HSV 的样式 以便检测
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv,color_lower,color_upper) #设置阈值，去除背景 保留所设置的
颜色
        #显示腐蚀后的图像
        mask = cv2.erode(mask,None,iterations=2)
```

```
        #高斯模糊
        mask = cv2.GaussianBlur(mask,(3,3),0)
        #图像合并
        res = cv2.bitwise_and(frame,frame,mask=mask)
        #边缘检测
        cnts = cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SI
MPLE)[-2]
        if len(cnts) >0 : #通过边缘检测来确定所识别物体的位置信息得到相对坐标
            cnt = max(cnts,key=cv2.contourArea)
            rect = cv2.minAreaRect(cnt)
            # 获取最小外接矩形的 4 个顶点
            box = cv2.boxPoints(rect)
            #获取坐标 长宽 角度
            c_x, c_y = rect[0]
            c_h, c_w = rect[1]
            c_angle = rect[2]
            if c_angle<-45:
                c_angle = -(90+c_angle)
            #为了防止背景的干扰，限定识别到木块的像素值范围
            if 3 < c_h < 180 and 3 < c_w < 180:
                #绘制轮廓
                cv2.drawContours(frame, [np.int0(box)], -1, (0, 255, 255), 2)
                x_bias = c_x - width/2
```

```
            y_bias = c_y - hight/2
            area = c_h*c_w
        else:
            c_h = 0
            c_w = 0
            x_bias = 0
            y_bias = 0
            area = 0
        cv2.imshow('frame',frame) #将具体的测试效果显示出来
        if cv2.waitKey(5) & 0xFF == 27: #如果按了 ESC 就退出 当然也可以自己设置
            break
    cap.release()
    cv2.destroyAllWindows() #后面两句是常规操作,每次使用摄像头都需要这样设置一波
```

hsv.py

```
import cv2
import numpy as np
import RPi.GPIO as GPIO
from time import sleep
import time
import threading


def empty(a):
    pass
```

```
frameWidth = 640
frameHeight = 480
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(10, 50)       # 设置亮度
pulse_ms = 30
# 调试用代码，用来产生控制滑条
cv2.namedWindow("HSV")
cv2.resizeWindow("HSV", 640, 300)
cv2.createTrackbar("HUE Min", "HSV", 13, 179, empty)
cv2.createTrackbar("SAT Min", "HSV", 96, 255, empty)
cv2.createTrackbar("VALUE Min", "HSV", 119, 255, empty)
cv2.createTrackbar("HUE Max", "HSV", 23, 179, empty)
cv2.createTrackbar("SAT Max", "HSV", 255, 255, empty)
cv2.createTrackbar("VALUE Max", "HSV", 255, 255, empty)

lower = np.array([13, 96, 119])
upper = np.array([23, 255, 255])
erorr=0
targetPos_x = 0
targetPos_y = 0
```

```
    lastPos_x = 0
    lastPos_y = 0


    while True:
        _, img = cap.read()
        imgHsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        h_min = cv2.getTrackbarPos("HUE Min", "HSV")
        h_max = cv2.getTrackbarPos("HUE Max", "HSV")
        s_min = cv2.getTrackbarPos("SAT Min", "HSV")
        s_max = cv2.getTrackbarPos("SAT Max", "HSV")
        v_min = cv2.getTrackbarPos("VALUE Min", "HSV")
        v_max = cv2.getTrackbarPos("VALUE Max", "HSV")


        lower = np.array([h_min, s_min, v_min])
        upper = np.array([h_max, s_max, v_max])
        imgMask = cv2.inRange(imgHsv, lower, upper)    # 获取遮罩
        imgOutput = cv2.bitwise_and(img, img, mask=imgMask)
        imgMask = cv2.cvtColor(imgMask, cv2.COLOR_GRAY2BGR)    # 转换后，后期才能够
与原画面拼接，否则与原图维数不同
        imgStack = np.hstack([img, imgOutput])
        cv2.namedWindow("imgMask",0)
        cv2.resizeWindow("imgMask",800,400)
        cv2.imshow('imgMask', imgStack)    # 显示
```

```
    if cv2.waitKey(pulse_ms) & 0xFF == ord('q'):        # 按下“q”推出（英文输入法）
        print("Quit\n")
        break

cap.release()
cv2.destroyAllWindows()
```

z_beep.py

```
import RPi.GPIO as GPIO
import time
#端口模式设置
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
#引脚定义
BEEP_PIN = 21
#蜂鸣器鸣叫
def on():
    GPIO.output(BEEP_PIN, 1)
     #蜂鸣器关闭
def off():
    GPIO.output(BEEP_PIN, 0)
#蜂鸣器端口反转
def flip():
    GPIO.output(BEEP_PIN, not GPIO.input(BEEP_PIN))
```

```
#间接发出响声，x 代表间接的时间，单位为秒
def beep(x):
    on()
    time.sleep(x)
    off()
    time.sleep(x)

#初始化蜂鸣器
def setup_beep():
    GPIO.setup(BEEP_PIN, GPIO.OUT, initial = 0)

#循环蜂鸣器，间隔 1 秒响一次
def loop_beep():
    beep(0.1)

#程序反复执行处
if __name__ == "__main__":
    setup_beep()
    beep(0.1)
    beep(0.1)
    beep(0.1)
    try:
```

```
        while True:
            loop_beep()
    except KeyboardInterrupt:
        off()
```

z_uart.py

```
import serial
import time
import threading
import z_led as myLed
import z_beep as myBeep
#全局变量定义
ser = ''
uart_baud = 115200
uart_get_ok = 0
uart_receive_buf = ""
uart_receive_buf_index = 0
#发送字符串 只需传入要发送的字符串即可
def uart_send_str(string):
    global ser
    ser.write(string.encode("utf-8"))
    time.sleep(0.01)
    ser.flushInput()
```

```
    #线程调用函数，主要处理数据接受格式，主要格式为 $...!  #...! {...}三种格式，...内容长度不限
    def serialEvent():
        global ser,uart_receive_buf_index,uart_receive_buf,uart_get_ok
        mode = 0
        try:
            while True:
                if uart_get_ok == 0:
                    uart_receive_buf_index = ser.inWaiting()
                    if uart_receive_buf_index > 0:
                        uart_receive_buf = uart_receive_buf+ser.read(uart_receive_buf_index).decode()
                        if mode == 0:
                            if uart_receive_buf.find('{') >= 0:
                                mode = 1
                            elif uart_receive_buf.find('$') >= 0:
                                mode = 2
                            elif uart_receive_buf.find('#') >= 0:
                                mode = 3
                        if mode == 1:
                            if uart_receive_buf.find('}') >= 0:
                                uart_get_ok = 1
                                mode = 0
                                ser.flushInput()
```

```
                elif mode == 2:
                    if uart_receive_buf.find('!') >= 0:
                        uart_get_ok = 2
                        mode = 0
                        ser.flushInput()
                elif mode == 3:
                    if uart_receive_buf.find('!') >= 0:
                        uart_get_ok = 3
                        mode = 0
                        ser.flushInput()
        except IOError:
            pass;


#串口接收线程
uart_thread = threading.Thread(target=serialEvent)


#串口初始化
def setup_uart(baud):
    global ser,uart_thread,uart_receive_buf
    uart_baud = baud
    ser = serial.Serial("/dev/ttyAMA0", uart_baud)
    ser.flushInput()
    uart_thread.start()
```

```
    uart_send_str("uart init ok!\r\n");
    uart_receive_buf = ''


#循环执行串口
def loop_uart():
    global uart_get_ok, uart_receive_buf
    if(uart_get_ok):
        print(int(time.time()*1000))
        uart_send_str(uart_receive_buf)


        if uart_receive_buf == '$LEDON!':
            myLed.on(1)
        elif uart_receive_buf == '$LEDOFF!':
            myLed.off(1)
        elif uart_receive_buf == '$BEEPON!':
            myBeep.on()
        elif uart_receive_buf == '$BEEPOFF!':
            myBeep.off()


        uart_receive_buf = ''
        uart_get_ok = 0
#大循环
if __name__ == '__main__':
```

```
setup_uart(115200)
myLed.setup_led()
myBeep.setup_beep()
#发出哔哔哔作为开机声音
myBeep.beep(0.1)
myBeep.beep(0.1)
myBeep.beep(0.1)
try:
    while True:
        loop_uart()
except KeyboardInterrupt:
    if ser != None:
        ser.close()
    myLed.off(1)
    myLed.off(2)
    myBeep.off()
```