



许昌学院
XUCHANG UNIVERSITY

本科生毕业设计

基于 SLAM 激光雷达的智能送餐机器人

学	院	<u>信息工程学院</u>
专	业	<u>物联网工程</u>
班	级	<u>2017 级本科 1 班</u>
学	号	<u>5006170044</u>
学 生 姓 名		<u>马豪勇</u>
联 系 方 式		<u>17665201525</u>
指 导 教 师		<u>张向群</u> 职称: <u>副教授</u>

2021 年 5 月

独 创 性 声 明

本人郑重声明：所呈交的毕业论文（设计）是本人在指导老师指导下取得的研究成果。除了文中特别加以注释和致谢的地方外，论文（设计）中不包含其他人已经发表的研究成果。与本研究成果相关的所有人所做出的任何贡献均已在论文（设计）中作了明确的说明并表示了谢意。

签名： _____
_____年____月____日

授权声明

本人完全了解许昌学院有关保留；使用本科生毕业论文（设计）的规定，即：有权保留并向国家有关部门或机构送交毕业论文（设计）的复印件和磁盘，允许毕业论文（设计）被查阅和借阅。本人授权许昌学院可以将毕业论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用影印；缩印或扫描等复制手段保存；汇编论文（设计）。

本人论文（设计）中有原创性数据需要保密的部分为（如没有，请填写“无”）：

学生签名： _____
_____年____月____日
指导教师签名： _____
_____年____月____日

基于 SLAM 激光雷达的智能送餐机器人

摘 要

在我国餐饮业中，顾客点餐和送餐等服务是件繁琐且耗时的工作。人工点餐的时候比较繁忙且容易出错，使得餐厅工作人员工作量增大。自新型冠状病毒爆发以来，为避免交叉感染，众多餐饮行业宣布暂停营业，餐饮市场损失严重。为提高送餐效率和降低其服务费用的成本，同时也降低交叉感染的风险，可以把物联网技术引入到餐饮市场服务行业，本论文设计了一款基于 SLAM 激光雷达的智能送餐机器人系统。

按照送餐机器人系统的设计要求，本论文设计了智能点餐系统、室内环境检测装置以及送餐机器人。能够实现在无服务员情况下完成点餐、送餐及支付整个流程，较为有效地提高了餐厅在人员较多情况下的服务效率。硬件系统方面，本文完成了基于 Raspberry Pi 4B 的室内环境监测装置以及基于 SLAM 激光雷达的送餐机器人的安装调试；软件系统方面，完成了客人点餐 APP 以及商家管理平台的设计。送餐机器人系统采用目前比较先进的 SLAM 算法+激光雷达，完成了送餐机器人的控制系统的开发，实现了送餐机器人的送餐功能，包括构建模拟餐厅地图、自身定位、路径规划、导航、实时避障等，达到设计目标。

关键词： 点餐 APP；商家管理平台；激光雷达；SLAM 算法

Intelligent Food Delivery Robot Based on SLAM Lidar

ABSTRACT

In China's catering industry, customer ordering and delivery is a tedious and time-consuming work. Manual ordering is busy and error-prone, which increases the workload of restaurant staff. Since the outbreak of novel coronavirus, in order to avoid cross-infection, a number of food and beverage industry announced the suspension of business, food and beverage market losses. In order to improve the efficiency of food delivery and reduce the cost of its service fees, as well as reduce the risk of cross-infection, the Internet of Things technology can be introduced into the catering market service industry. In this paper, a SLAM lidar based intelligent food delivery robot system is designed.

According to the design requirements of the food delivery robot system, this paper designed an intelligent food ordering system, indoor environment detection device and food delivery robot. It can complete the whole process of ordering food, delivering food and paying without a waiter, which effectively improves the service efficiency of the restaurant in the case of more people. In terms of hardware system, this paper completed the installation and debugging of indoor environment monitoring device based on Raspberry Pi 4B and food delivery robot based on SLAM Lidar. In terms of software system, the design of customer ordering APP and merchant management platform is completed. The food delivery robot system adopts the currently advanced SLAM algorithm + laser radar to complete the development of the control system of the food delivery robot and realize the food delivery function of the food delivery robot, including the construction of simulated restaurant map, self-positioning, path planning, navigation, real-time obstacle avoidance, etc., to achieve the design goal.

Key words: Order the APP; Merchant Management Platform; Laser radar; SLAM algorithm

目 录

1. 绪论	1
1.1 研究背景及意义	1
1.1.1 研究背景	1
1.1.2 研究意义	1
1.2 国内外研究现状	2
1.2.1 国外现状	2
1.2.2 国内现状	2
1.3 关键技术难点的研究及解决方法	2
1.4 送餐机器人发展趋势	3
2. 系统需求分析	4
2.1 硬件需求	4
2.1.1 室内环境检测装置	4
2.1.2 送餐机器人	4
2.2 软件需求	4
2.2.1 开发工具	4
2.2.2 开发语言	5
2.2.3 开发框架	5
2.2.4 开发环境	6
2.3 性能需求	6
3. 硬件设计	7
3.1 室内环境检测装置设计	7
3.1.1 装置组成	7
3.1.2 核心控制模块	7
3.1.3 网络传输模块	8

3.1.4 RGB-LED 灯模块.....	8
3.1.5 烟雾报警传感器模块.....	9
3.1.6 火焰报警传感器模块.....	9
3.2 送餐机器人硬件设计.....	10
3.2.1 HiBot 底盘驱动板.....	10
3.2.2 RPLIDAR A1 激光雷达.....	10
3.2.3 核心控制模块.....	11
3.2.4 运动模组.....	12
3.2.5 IMU 姿态传感器.....	12
4. 软件设计.....	13
4.1 智能点餐系统设计.....	13
4.1.1 系统功能及组成.....	13
4.1.2 智能点餐系统的总体流程设计.....	14
4.2 点餐 APP 的实现.....	15
4.2.1 登录注册功能.....	15
4.2.2 选座功能.....	17
4.2.3 订餐管理功能.....	18
4.2.4 呼叫服务员功能.....	19
4.2.5 一键支付功能.....	20
4.3 商家管理平台的实现.....	21
4.3.1 登录注册功能.....	22
4.3.2 查看当前订单功能.....	23
4.3.3 查看历史订单功能.....	24
4.3.4 智能分析功能.....	24
4.3.5 查看室内环境状态功能.....	25
4.4 服务器端设计.....	26
4.4.1 点餐 APP 服务器设计.....	26
4.4.2 室内环境检测装置服务器设计.....	27
4.5 数据库设计.....	27

4.5.1 数据库概念结构设计	27
4.5.2 数据库逻辑结构设计	28
5. 送餐机器人的详细设计	30
5.1 激光雷达节点	30
5.1.1 激光雷达介绍	30
5.1.2 激光雷达消息格式	30
5.2 送餐机器人启动节点	31
5.3 送餐机器人开发环境配置	32
5.4 STM32 和 ROS 通信	32
5.4.1 通信内容和原理	32
5.4.2 STM32 的串口通信	32
5.5 Gmapping 建图算法	33
5.5.1 Gmapping 介绍	33
5.5.2 Gmapping 功能包实现的前提	34
5.6 送餐机器人局部路径规划算法	34
5.6.1 DWA 路径规划	34
5.6.2 送餐机器人的运动模型	34
5.6.3 速度采样	35
6. 系统调试	37
6.1 Gmapping 建图	37
6.2 DWA 导航	40
7. 总结与展望	43
7.1 总结	43
7.2 展望	43
参考文献	44
附 录	46
致 谢	53

1. 绪论

1.1 研究背景及意义

1.1.1 研究背景

近年来,随着我国现代人们物质生活水平的进一步提高,中国的餐饮服务行业也在快速地发展,在现在这个信息化的时代,消费者对信息化和智能化的需求和关注程度越来越高。目前,中国餐饮信息化的发展还处于起步阶段。餐饮企业对餐饮企业信息化存在误区以及对餐饮系统的认识不足等因素,使得餐饮信息化远远落后于其他行业。在用餐高峰期,服务员效率低、易出错、客人点菜速度慢等众多问题暴露出来,2020年,面对新型冠状病毒这场突如其来的危机,为防止病毒传播,相关部门提出的众多的限制要求使得客源流量突然下降,许多餐饮企业纷纷倒闭止损。

近年来,中国老龄化的现象越来越严重,在一定程度上这种现象促使了劳动力市场的持续增加,促使了智能化机器人业务在社会中得到普遍的开展^[1]。激光雷达机器人是目前市面上最新系列的智能机器人,它通过自身的激光雷达传感器对周围环境的观测,运用 SLAM 算法等优化算法,得出起点到终点之间的最优路径,提高工作效率。本文将借助激光雷达技术,设计一款基于 SLAM 激光雷达的送餐机器人系统,在“无人”情况下实现从点餐到送餐的全过程,非接触的送餐方式更能保证用餐安全,提高餐饮企业的工作效率。

1.1.2 研究意义

2020年,突如其来的疫情对各个行业影响很大。特别是餐饮行业,在本身的高人力成本、高材料成本的基础上又要面对资源短缺,房租上涨和人力成本不断增高等困难。采用人工的送餐方式,在用餐高峰期时,会出现排队拥堵,服务员服务不及时等情况,以及无法保证送餐过程中人与人交叉感染的风险。

本文提出的基于 SLAM 激光雷达的送餐机器人,基于非接触式、精确、智能高效的送餐方式,对餐饮行业复工复产有很大帮助。通过送餐机器人,采用非接触的方式,可以减少人与人的接触,使用餐更加安全。目前,送餐机器人的身影已经出现在许多餐饮行业门店中,吸引了大量的顾客前来观摩,用送餐机器人代替餐厅工作人员不仅可以吸引客流量,同时也节省了餐厅的投入成本。许多餐饮企业开始改革创新,用送餐机器人完成机械式的送餐和非技术的工作,而餐厅服务员可以投入到个性化的工作中。送餐机器人给增值服务提供了空间,无疑是给餐饮企业增加了新的卖点。

根据行业协会的统计数据显示,近年来,餐饮业的员工工资和餐厅的房租的总和将

占餐厅收入的三分之一。在人工成本、房租等成本的持续上涨的趋势下，餐饮行业将寻找新的解决方案，送餐机器人新技术的引用将解放部分餐饮工作人员的工作，与此同时，送餐机器人的投资成本相比于餐厅服务员的成本少很多，投资回报周期短。

1.2 国内外研究现状

1.2.1 国外现状

据专家预测，在 2020 年左右，海外市场将出现大量的送餐机器人。信息时代和“5G”时代到来，使我们的生活更加智能化，这更加促进了解放廉价劳动力的局面。相对于国内而言，国外像美国、法国等发达国家，送餐机器人将在国外市场会拥有广阔市场，因为多年来国外的人力成本一直很高，其次，人力成本的大幅度提高，导致利润空间一再压缩，发展空间收到限制，不少餐饮企业不得不进行裁员，这样会使服务质量大大降低。在送餐机器人的研究领域和应用范围，国外处于刚起步阶段，例如，美国的 **Bear Robotics** 公司的送餐机器人产品仍处于试验样机阶段，投入到餐饮服务业市场的道路还很漫长。使用场景少、没有量产、没有在大量的实验中得出数据、没有更新换代产品升级等一系列问题暴露出来。

1.2.2 国内现状

餐饮行业是我国较早开放的行业。经历多年的竞争与发展，我国餐饮业发展已经进入了多行业交叉发展的新阶段^[2]，行业发展非常迅速。随着人们生活消费水平、生活质量的提高以及信息化时代的到来，在竞争格局和餐饮消费行为上，餐饮行业时刻的发生巨大变化。越来越多的餐饮企业利用信息技术来提高管理水平。中国传统食品与现代信息管理的结合对于企业做大做强是非常重要的。在国内，已经有很多家科技创新公司在研发送餐机器人，并且已经在餐厅投入使用，并得到了很好的预期效果。

近年来，中国市场对智能机器人的需求越来越大，已经成为世界上最大的机器人消费市场。国内地区许多城市都出现了送餐机器人的身影，人们大多会慕名前来参观送餐机器人，从而使得客流量增大，同时也带动了餐饮行业的利润，使得送餐机器人具有很大的发展空间。

1.3 关键技术难点的研究及解决方法

本文设计了一款基于 SLAM 激光雷达的送餐机器人系统，包含智能点餐系统、室内环境检测装置和送餐机器人，实现了无人情况下完成点餐，无接触式的送餐功能。本文主要研究的内容如下：

- (1) 用餐高峰期时会存在订单数量过多，处理不及时的情况。对于用户过多、数

据过多的情况可以通过多线程多用户的服务器解决。基于这一难点，本文采用 Python 语言的 socket 模块通信，商家管理平台服务器端与点餐 APP 客户端之间相互数据传输，实现了多线程多用户同时发送数据给服务器，在同一时间能处理大量的订单数据。

(2) 送餐机器人在送餐过程中可能存在静止的障碍物或者是动态的障碍物，我们需要在构建好全局地图的基础上，采用局部路径规划的方法来躲避障碍物，目前有两种局部路径规划算法，DWA 和 TEB，根据实际情况综合考虑本论文设计中采用 DWA 算法，因为 DWA 导航相比于 TEB 导航在障碍物较多的情况下，安全性很高，不会碰到障碍物。

1.4 送餐机器人发展趋势

从目前国内的研究现状来看，送餐机器人具有很好的发展空间。从 2015 年起，国内的少部分的餐厅开始出现送餐机器人，但由于技术还不成熟，并没有给餐饮行业带来盈利。近年来，送餐机器人再次出现在大众的眼前。随着科技的发展，定位技术的不断创新和完善，送餐机器人的定位准确度越来越高，能够实现精准的路径规划，避障等功能。科技发展的背后，激光雷达等核心零件的成本也由原来的数万元控制到如今的几千元不等，同时，送餐机器人的使用寿命和性能也得到了显著的提高，性价比更高。

送餐机器人的投入使用，也解决了人力成本过高等一系列问题，越来越多的商业巨头开始进军智慧餐厅，那送餐机器人将会作为重要的组成部分。由此看出，送餐机器人的出现将会成为未来餐厅行业发展的一个趋势。

2. 系统需求分析

2.1 硬件需求

2.1.1 室内环境检测装置

室内环境检测装置的核心部分为核心控制模块，它需要完成接收数据信息，并处理信息功能的实现。经过对系统的流程以及实现功能的分析，选用 Raspberry Pi 4B 作为核心控制模块，并选用以下硬件实现：

- (1) Raspberry Pi 4B 主板，为该系统的核心控制模块，用于接收传感器数据信息，处理信息，信息转发和接收。
- (2) 远红外火焰传感器，检测外界红外光的强弱，本论文设计中用来检测是否发生了火灾。
- (3) MQ-2 烟雾传感器，检测外界烟雾浓度强弱，本论文设计中用来检测是否有烟雾。
- (4) RGB-LED 灯，模拟室内灯光。
- (5) 无源蜂鸣器，模拟警报器。

2.1.2 送餐机器人

在选择机器人平台时需要考虑的许多因素，比如价格，定位精确度和调试难易程度等，根据需求，我们选用 HiBot 机器人平台。送餐机器人的主控核心部分采用 Raspberry Pi 4B，并选用以下硬件实现：

- (1) Raspberry Pi 4B 主控核心模块，基于 ARM 的微型电脑主板，支持 Python、C、C++ 等编程语言。
- (2) RPLIDAR A1 激光雷达，测距传感器，很容易应用到 SLAM 算法中并且很容易实现较高的测量精度。
- (3) 基于 STM32F405 的底盘驱动板，在该驱动板上设置了丰富的接口，比如，激光雷达接口，树莓派通信接口，GPIO 功能口，充电管理模块等。
- (4) 有刷直流电机。
- (5) 增量式编码器。
- (6) MPU6050 姿态传感器。

2.2 软件需求

2.2.1 开发工具

- (1) Android Studio

Android Studio 是一个基于 IntelliJ IDEA 的 Android 开发环境。Android Studio 为开发和调试提供了集成的 Android 开发工具^[3]。Android Studio 生成了 Android 应用程序设计和组件。

(2) PyCharm

PyCharm 是一种 Python IDE，在开发 Python 语言时，它是一款高效率、功能强大的工具。版本分社区版和专业版，专业版提供了更加强大的功能，比如本论文设计中的商家管理平台就是用 Flask 框架开发的。

(3) VMware Workstation

VMware Workstation 是一款功能强大的桌面虚拟计算机软件^[4]。由于本论文设计中进行 ROS 分布式开发前，我们可以在 VMware Workstation 中加载出一台虚拟机，进行 Ubuntu 虚拟环境的搭建。

(4) Navicat

Navicat 是一种十分方便的数据库管理工具，在开发人员的需求中，满足数据库管理，用直观的图形化界面，方便我们进行数据库的操作。完全满足本设计开发要求。

2.2.2 开发语言

(1) 用户点餐 APP 主要使用 Java 语言，Java 语言使用可跨平台，开发资源分享，面向对象的设计使项目更安全。

(2) 商家管理平台系统和送餐机器人主要使用 Python 语言开发的。Python 可移植性强、可扩展性多并且有丰富的库。Python 提供了许多接口和库，方便我们进行扩展。

(3) 数据库选用 SQLite，SQLite 是一个进程内库，它实现了一个自给自足的、无服务器的、零配置的事务性 SQL 数据库引擎^[5]。我们在使用的时候，不需要配置它，直接访问。

2.2.3 开发框架

框架开发具有效率高、系统安全性高的优点。系统服务器与客人手机里的点餐 APP 和室内环境检测装置之间的数据交互使用 Python 的 Socket 通信进行开发，Python 提供了两个基本的 Socket 模块。一个是 Socket，另一个是 SocketServer，使用 socket 模块可以方便我们进行开发^[6]。此论文设计了服务器开发，其基础实现结构为以下文件：

- (1) appserver.py 接收点餐 APP 发来的数据流，解析并存入 sqlite 数据库。
- (2) pserver.py 接收室内环境检测装置中各种传感器检测的数据。
- (3) piclient.py 将室内环境检测装置检测的数据轮询的发给服务器端。

2.2.4 开发环境

点餐 APP 的开发使用的是 android studio 开发，需要配置 android studio 运行所需的环境，JDK 版本选择 jdk 11.0.11 版本。商家管理平台的开发使用的是 Python Flask 框架，需要配置 Python 运行所需的环境，选择 PyCharm 软件。送餐机器人的开发使用的是 Ros 系统，需要在 Linux 的环境下安装 Ros 系统。室内环境检测装置的数据通信模块与服务程序进行通信，服务器在后台运行才能进行正常的数据交互。

2.3 性能需求

智能点餐系统需要在餐厅营业期间保持后台运行状态，商家管理员可以登录到商家管理平台上查看客人的订餐情况，并做出一系列操作。综合实用性以及管理员的体验，对本系统做出以下性能需求：

- (1) 点餐 APP 操作方便，使用简单，后期可维护强，满足大部分用户的需求。
- (2) 系统稳定运行，用餐高峰期时间，系统接收大量订单，处理大量数据，需要系统保持稳定性，冗余量要求少。
- (3) 数据传输安全可靠，准确，不容易丢失，响应速度高。
- (4) 扩展性和灵活性好，方便以后产品升级换代和系统维护。

3. 硬件设计

3.1 室内环境检测装置设计

3.1.1 装置组成

本装置的主控的核心部分使用 Raspberry Pi 4B 主板，其上搭载核心控制模块，网络传输模块，外接 RGB_LED 灯、烟雾传感器、火焰传感器、蜂鸣器作为环境采集模块，各模块共同协助运行，完成室内环境检测工作。总体框架图如图 3-1 所示

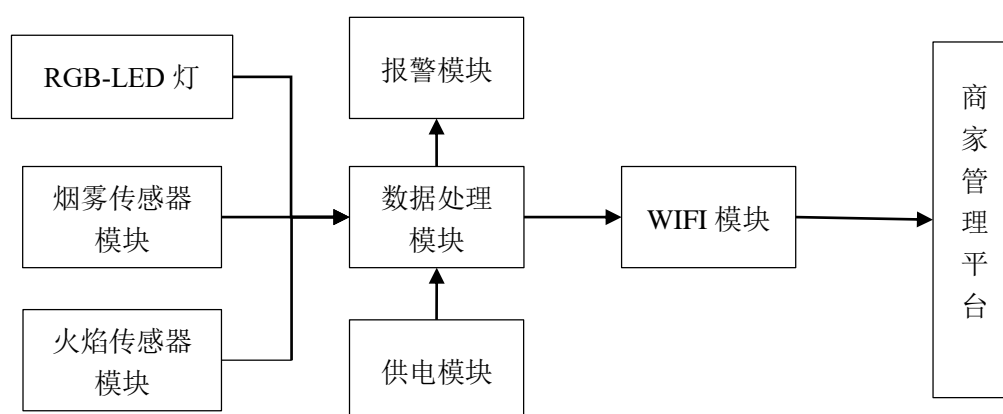


图 3-1 室内环境检测装置总体框架图

室内环境检测装置主要是对餐厅环境进行检测，并将检测结果通过 TCP/IP 协议发往服务器端，商家管理平台通过对接收的数据进行判断，有效的数据将显示在网页上。同时商家管理平台可以在网页上远程控制 RGB-LED 灯的开关，不用手动开关，方便了商家管理员对餐厅的管理。

3.1.2 核心控制模块

核心控制模块作为整个室内环境检测装置的核心组成部分，具有数据传输与处理、运行控制系统等核心功能^[7]。本论文设计中采用树莓派 4B 主板作为室内环境检测的核心控制组成部分，因为它可以通过 GPIO 口进行数据传输与处理，并且具有系统控制等核心功能，所以选树莓派 4B 作为主控制芯片。详细功能分布如图 3-2 所示。

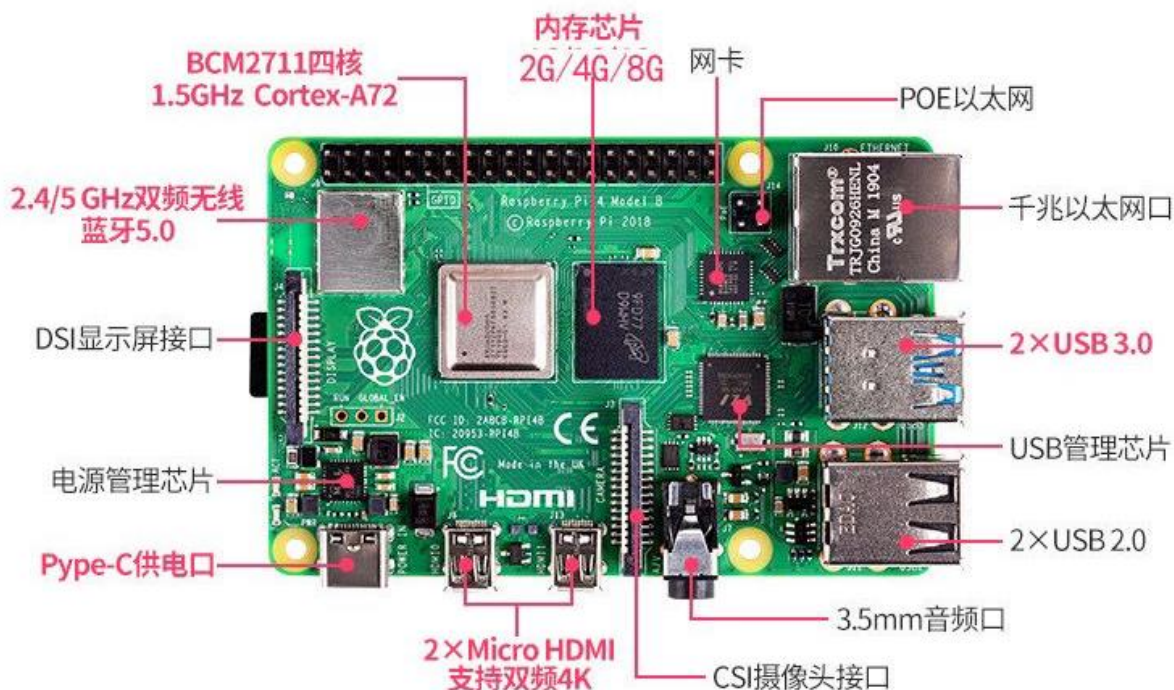


图 3-2 详细功能分布图

3.1.3 网络传输模块

Raspberry Pi 4B 主板本身带有 WIFI 模块, 通过 python socket 编程可完成树莓派与商家管理平台的互传工作。可实现本地数据上传至商家服务端。

3.1.4 RGB-LED 灯模块

RGB-LED 模块可以发出各种颜色的光。在本论文设计中, 其相当于餐厅的灯, 如果某片区域无客人, 我们可以通过商家管理平台远程控制灯的开关, 一方面无需手动开关, 另一方面节省用电, 减少成本支出。实物如图 3-3 所示。在该装置初始化时, 会循环检测 LED 灯引脚的高低电平信号, 并将信号发送给商家管理平台, 在商家管理平台的管理界面上可以控制 LED 灯的开关, 方便快捷。

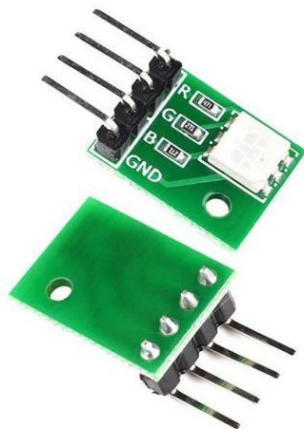


图 3-3 RGB-LED 灯实物图

3.1.5 烟雾报警传感器模块

MQ-2 气体传感器所使用的气敏材料是在清洁的空气中导电率较低的二氧化锡 (SnO_2)。实物图如图 3-4 所示。上电后需要预热 20s 左右,测量的数据才稳定。轮询读取引脚口的输入高低电平状态,当没有检测到烟雾时,DO 输出高电平,灯灭;当检测到烟雾时,输出低电平,指示灯亮,同时蜂鸣器发出警报声音。



图 3-4 烟雾传感器实物图

3.1.6 火焰报警传感器模块

火焰传感器可以检测火焰或者波长在 800 纳米~1000 纳米范围的光源,传感器与火焰要保持一定距离,以免高温损失传感器,用打火机模拟时,需要保持 60 厘米~90 厘米的距离,以免损坏传感器。同时还可以调节灵敏度。实物图如图 3-5 所示。将火焰传感器模块的引脚 DO 连接到 Raspberry Pi 4B 的 25 号引脚,通过检测引脚的高低电平状态判断是否存在火焰。当检测到火焰的同时,蜂鸣器会发出警报声音。



图 3-5 火焰传感器实物图

3.2 送餐机器人硬件设计

HiBot 具有高精度的定位性能,定位性能评价指标主要包含定位的准确度、精密度、定位分辨率等。送餐机器人的定位精度非常高,我们也可以采用相应的算法对其定位性能优化提升。

3.2.1 HiBot 底盘驱动板

HiBot 底盘驱动板采用 STM32F405 底盘控制器,STM32F405/415 提供了工作频率为 168MHz 的 Cortex™-M4 内核的性能。当 CPU 工作于所有允许的频率时,在闪存中运行的程序,可以达到相当于零等待周期的性能。STM32F4 系列微控制器集成了单周期 DSP 指令和 FPU,提升了计算能力,可以进行一些复杂的计算和控制^[8]。

如图 3-6 所示,送餐机器人底盘驱动电路中各个功能模块。

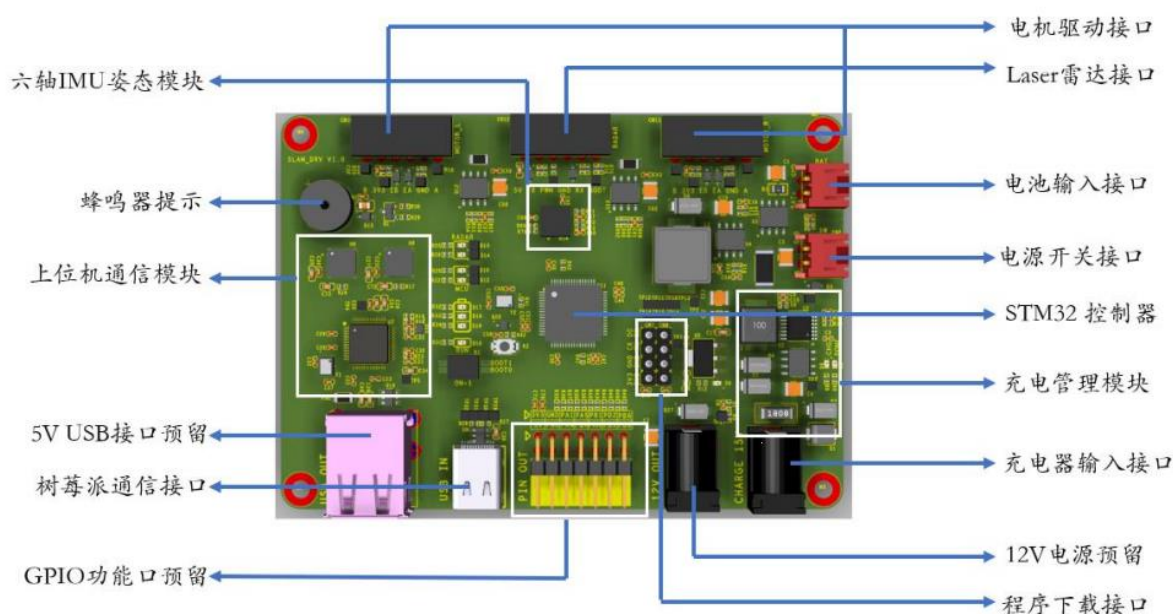


图 3-6 电盘驱动板

3.2.2 RPLIDAR A1 激光雷达

激光测距法主要用于距离测量。激光测距单元比较精确,其输出只需做比较简单的处理就可以实现,比较容易应用于 SLAM。采用 RPLIDAR A1 激光雷达完全满足功能需求,详细性能如表 3-1 所示:

表 3-1 激光雷达详细性能

项目	单位	最小值	典型值	最大值	备注
测距范围	米	待定	0.15-12	待定	基于白色反光物体测得
扫描范围	度	不适用	0-360	不适用	
测距分辨率	毫米	不适用	<0.5	不适用	测量物体在 1.5 米以内
角度分辨率	度	不适用	≤ 1	不适用	5.5hz 扫描时
单次测距时间	毫秒	不适用	0.5	不适用	
测量频率	赫兹	2000	≥ 4000	8000	
扫描频率	赫兹	5	5.5	10	扫描一周的频率

3.2.3 核心控制模块

HiBot 机器人核心控制模块采用 Raspberry Pi 4B 主板。它相当于一台微型电脑，树莓派以低功耗、小巧、方便携带、有丰富的 GPIO 口等优点，以及成本低，非常适合用于学习人工智能。详细参数如表 3-2 所示。

表 3-2 树莓派详细参数

网卡	BroadcomBCM2835(CPU,GPU,DSP 和 SDRAM,USB)
CPU	ARM1176JZF-S 核心(ARM11 系列)700MHZ
GPU	Broadcom VideoCroe IV,OpenGL ES 2.0
内存	512Mbyte
USB2.0 接口个数	2
音频输出	3.5mm 插孔, HDMI
板载存储	SD/MMC/SDIO 卡插槽
网络接口	10/100 以太网接口
外设	8 个 GPIO、UART、I2C、SPI 总线
额定功率	700mA(3.5W)
电源输入	5V
操作系统	Debian GNU/Linux

3.2.4 运动模组

HiBot 配备有刷直流电机，有刷直流电机是直流电机的一种，在使用时需要配备电机驱动器，驱动器主要对电机的电流速度做高精度的控制。HiBot 使用增量式编码器，如图 3-7 所示，对于机器人从开始运动时刻累计里程，累计的里程我们可以转化为统一的单位，如 m/s 、 rad/s 等单位使用^[9]。

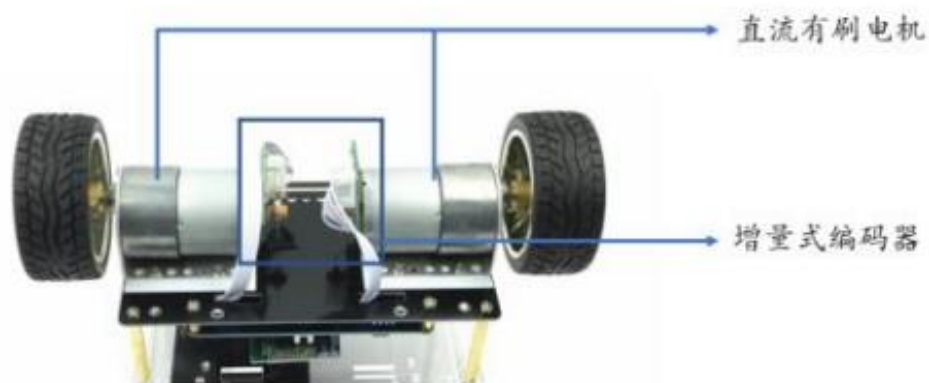


图 3-7 运动模组

3.2.5 IMU 姿态传感器

在 HiBot 上我们所采用的是 MPU6050，芯片内部主要包含三个单轴的加速度测量设备和三个独立的单轴陀螺仪，对于最初的加速度和旋转角速度我们可以直接通过 I2C 总线进行读取，然后利用 AHRS 算法将系统中的姿态数据进行了融合，可以获得比较高精度的姿态数据。

4. 软件设计

4.1 智能点餐系统设计

4.1.1 系统功能及组成

智能点餐系统由点餐 APP 和商家管理平台系统组成。智能点餐系统采用多方式就餐，主要为移动终端的网上订餐方式。客人可以在所使用的移动设备终端智能手机上下载点餐 APP 软件，连接餐厅的 WIFI，进行网上选餐订餐，商家管理员登录商家管理平台可以实时的接收客人的订单详情，并及时的处理。鉴于以上功能，设计了如下的智能点餐系统。

智能点餐系统主要包括五大功能模块，分别是：点餐 APP、商家管理平台、TCP/IP 服务器、智能分析、无线传输。如图 4-1 显示，

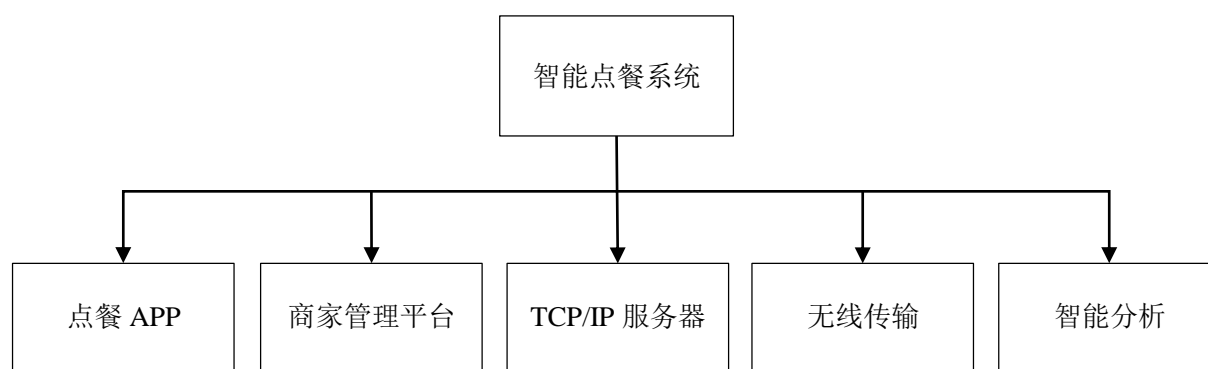


图 4-1 智能点餐系统构架图

下面对每一个功能模块进行详细的介绍。

(1) 点餐 APP：在智能点餐系统中，客人所使用的点餐工具就是智能手机，在连接餐厅无线路由器的网络后，客人可以通过智能手机下载点餐 APP 客户端进行登录、注册、点餐、呼叫服务员以及支付等操作，该 APP 界面设计简单明了，方便了客人的使用，可以在无服务员的情况下，完成点餐流程，有效地提高了点餐效率。

(2) 商家管理平台：在智能点餐系统中，商家管理员可以通过登录商家管理平台实时的查看客人的订单信息，比如座位号、人数、菜名、点餐时间等信息。同时，还可以通过该平台，查看室内环境监测装置反馈的信息。方便商家管理员进行查看。

(3) TCP/IP 服务器：在智能点餐系统中，点餐 APP、室内环境检测装置与商家管理平台之间的交互都是通过 TCP/IP 服务器作为中介来完成交互的。

(4) 无线传输：点餐 APP、室内环境检测装置和 TCP/IP 服务器之间的通信是通过无线传输技术来完成的。

(5) 智能分析：餐饮行业如果想取得长远的发展，离不开客人的点餐行为，知道客人喜欢那些菜，知己知彼百战百胜。我们将客人的点餐信息存储到 SQLite 数据库里，然后将数据进行分析比对，得出客人喜欢哪些菜品。通过分析出的这些数据信息，餐厅领导可以做出科学的决策^[10]。

点餐 APP 客户端采用 TCP/IP 局域网通信协议，通过连接餐厅的无线路由器，将数据发送到商家管理平台的服务器终端。SQLite 数据库服务器用于储存信息，并可由管理人员在服务端登录管理平台查看订单信息、智能分析结果以及一系列操作。本系统的物理架构图如图 4-2 所示。

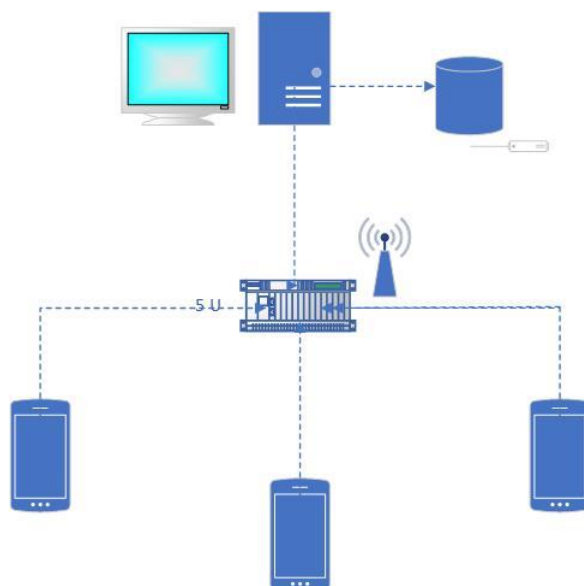


图 4-2 物理构架图

4.1.2 智能点餐系统的总体流程设计

根据以上图例对点餐流程进行梳理。

1. 客人在手机上下载点餐 APP，注册登录账号。
2. 客人填写座位号，用餐人数，点餐，并确定餐单。
3. 商家管理平台接收客人的订单。
4. 送餐机器人开始送餐。
5. 客人用餐结束，付款结账，结束。

总体流程图如图 4-3 所示：

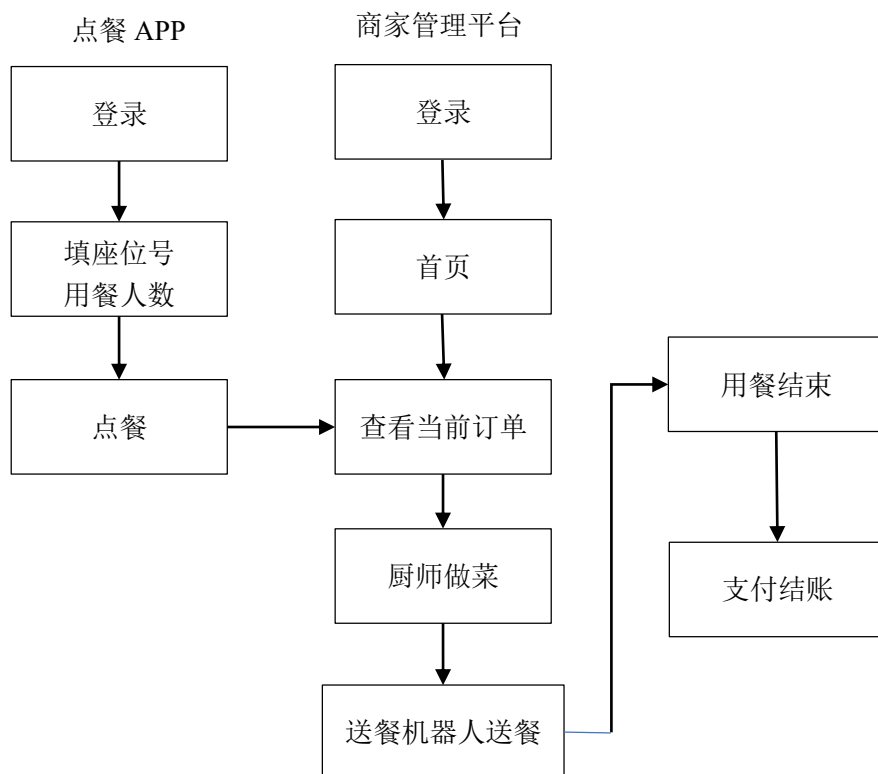


图 4-3 总体流程图

4.2 点餐 APP 的实现

智能点餐系统的点餐 APP 包含以下功能：登陆注册、选座、订餐管理、呼叫服务员以及一键支付等功能，如图 4-4 所示。

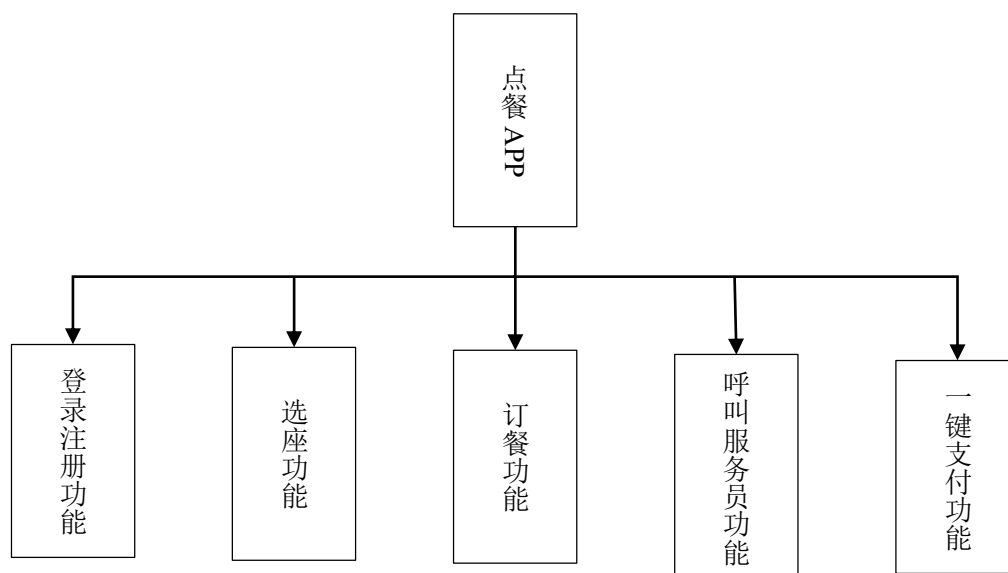


图 4-4 点餐 APP 功能图

4.2.1 登录注册功能

(1) 登录注册功能描述

客人打开点餐 APP 软件，首先出现登录界面，点击右上角注册按钮，即可进入注册界面，在注册界面，需要填写用户名，密码，确认密码以及验证码等信息，填写内容的格式准确无误即注册成功。返回登录界面，填写刚注册的账号和密码，填写准确无误即登录成功，进入点餐 APP 首页。

（2）登录注册功能界面设计

点餐 APP 的登录注册界面设计简单明了，易于操作。客人输入自己的账号和密码后，点击注册事件后，系统就会把数据提交到数据库里。返回登录界面，填写账号密码信息，验证无误，即可进入点餐首页。

点餐 APP 注册界面如图 4-5 所示

图 4-5 点餐 APP 注册界面

点餐 APP 登陆界面如图 4-6 所示



图 4-6 点餐 APP 登录界面

4.2.2 选座功能

(1) 选座功能描述

客人登录成功后跳转到选座功能界面，需要填写客人所坐的座位号以及用餐人数的信息。目的是方便商家管理员了解客人的用餐信息，为接下来的送餐做准备。

(2) 选座功能界面设计

点餐 APP 的选座功能界面设计比较简单明了，易于操作。客人输入座位号和人数信息后，点击确认按钮后，系统就会把数据暂时存到本地，等待选菜完毕一起发送给商家管理平台。

点餐 APP 的选座界面如图 4-7 所示



图 4-7 点餐 APP 选座界面

4.2.3 订餐管理功能

(1) 订餐管理功能描述

客人通过填写自己所在的座位号以及用餐人数后进入订餐首页，在订餐首页，点击右下角的加号，选择选餐选项，进入选餐界面。选餐完成后点击提交，即将选座信息，用餐人数信息，选餐信息一同发往商家管理平台，等待上菜。

(2) 订餐管理功能界面设计

点餐 APP 的订餐管理功能界面设计新颖。客人点击右下角的加号，会出现三个选项，选择选餐选项，跳转到选餐界面，选餐界面的左边是菜品分类，右面是该菜系的具体产品，分类明确。选餐完成后会跳转到订单界面，在订单界面上会显示选餐详情，如果选餐少了，点击添菜按钮，即可在原有的基础上添加新的菜品。

点餐 APP 的订餐管理界面如图 4-8 所示

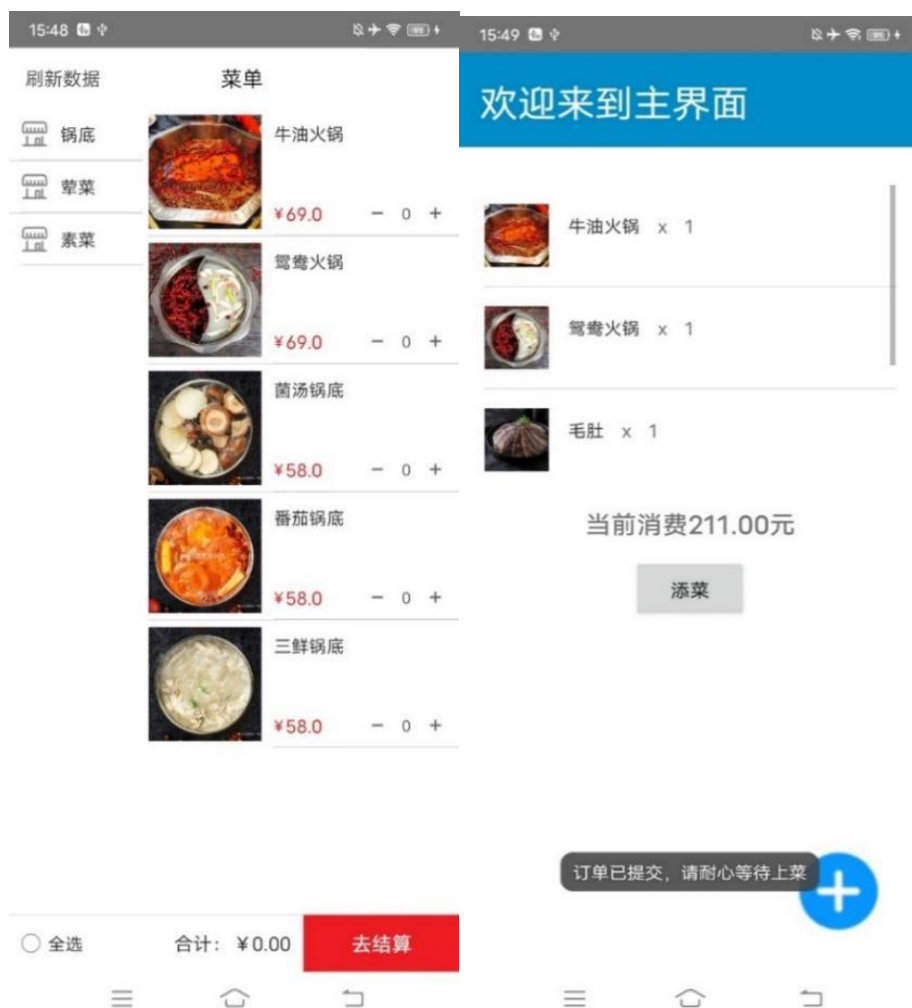


图 4-8 订餐管理界面

4.2.4 呼叫服务员功能

(1) 呼叫服务员功能描述

客人在订餐首页界面点击右下角的加号，点击呼叫服务员选项，即可实现呼叫服务员。

(2) 呼叫服务员功能界面设计

点餐 APP 的呼叫服务员界面设计通俗易懂，点击呼叫按钮，实现一键呼叫服务员，操作简单，无需繁琐的操作。

点餐 APP 呼叫服务员界面如图 4-9 所示

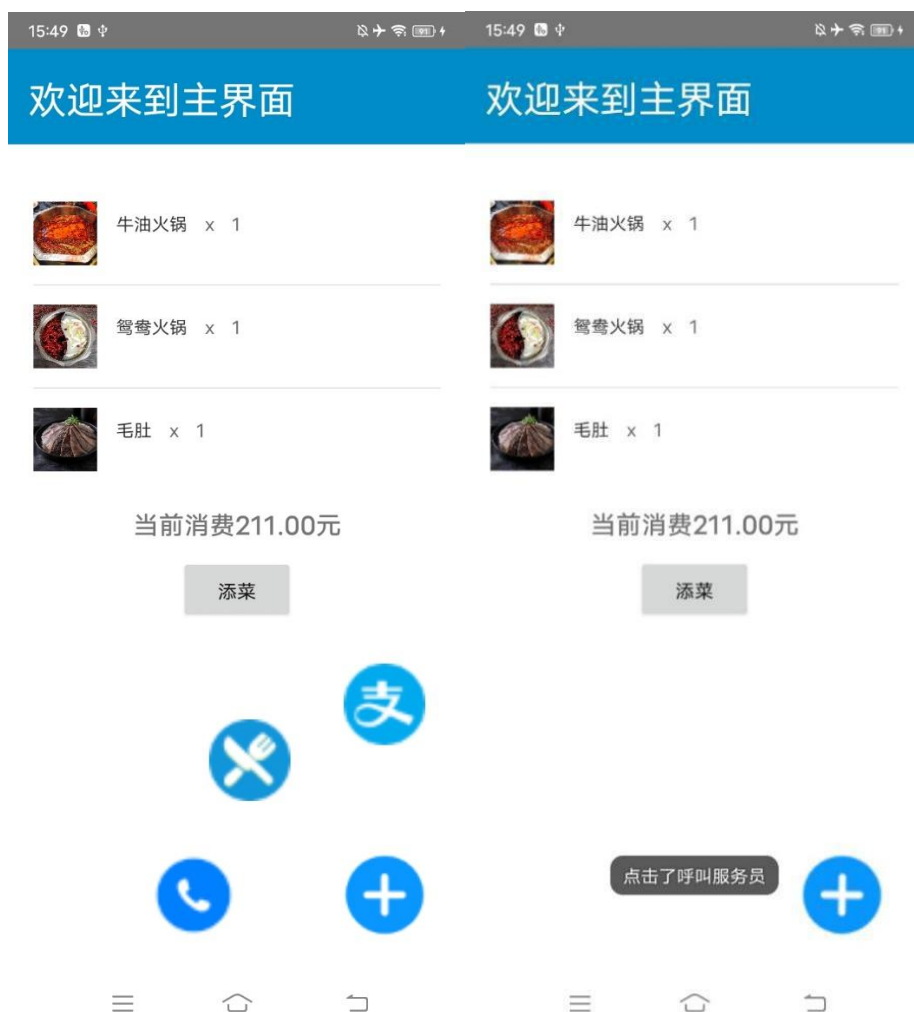


图 4-9 呼叫服务员界面

4.2.5 一键支付功能

(1) 一键支付功能描述

客人在订餐首页界面会显示订餐详情，以及总价格。点击右下角的加号，选择一键支付选项，即可实现自动跳转到支付宝里，完成支付，实现一键支付功能。

(2) 一键支付功能界面设计

点餐 APP 的一键支付功能界面非常简单明了，实现一键支付，无需繁琐的操作。点餐 APP 一键支付界面如图 4-10 所示

欢迎来到主界面

欢迎光临!



图 4-10 点餐 APP 支付界面

4.3 商家管理平台的实现

商家管理平台的服务器在后台多线程运行，一个是等待接收客人的点餐订单，一旦接收到数据，判断数据是否有缺失完整，有效的数据存到数据库里。商家管理平台便通过对数据库进行操作，查询，删除，增加等等。另一个是接收室内环境检测装置发出来的数据，首先还是判断数据的有效性。有效数据存储到数据库，通过对数据库操作的封装，实现对 RGB-LED 灯的远程控制。针对以上操作，设计了商家管理平台。

商家管理平台采用 Python 的 Flask 框架构建的,Flask 由单内核和两个外部库组成，其功能和配置非常简单。适用于微服务与轻应用。Flask 的特点为可定制性，灵活轻巧，通过扩展增加其功能。两个核心应用是 Werkzeug 和模板引擎 Jinja, Flask 工作模式如图 4-11 所示

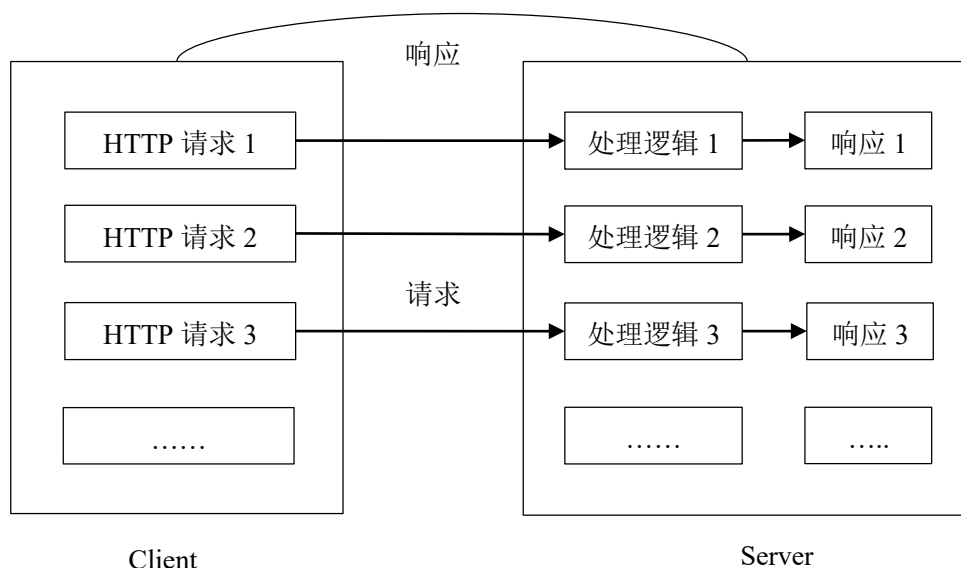


图 4-11 Flask 工作模式

商家管理平台有五大功能模块，登录注册功能，查看当前订单功能，查看历史订单功能，智能分析，查看室内环境状态功能。如图 4-12 所示。下面将详细介绍这五大功能。

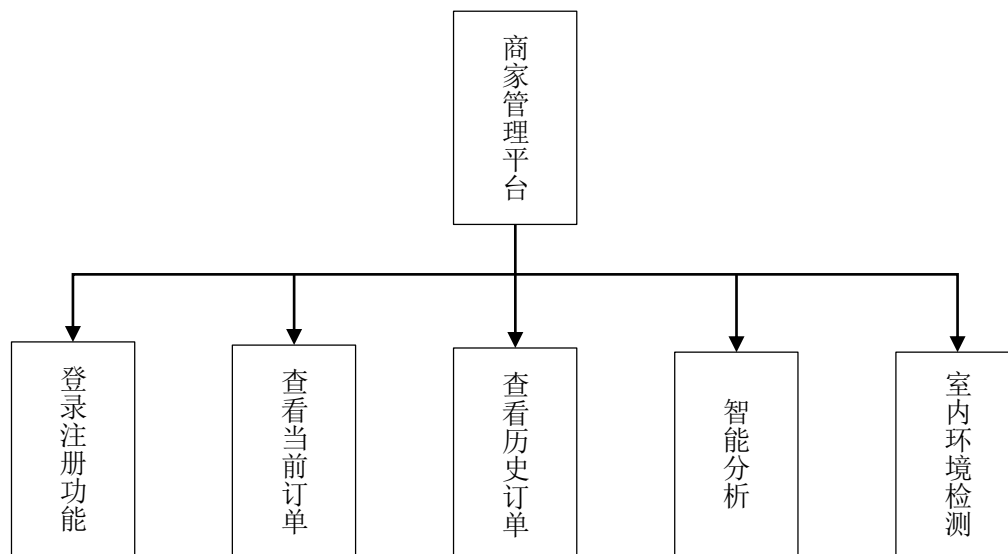


图 4-12 商家管理平台功能图

4.3.1 登录注册功能

(1) 登录注册功能描述

商家管理员通过在商家管理平台的注册界面上注册账号成功后，在登录界面输入已经注册的账号，即可进入商家管理平台主界面。

(2) 登录注册功能界面设计

商家管理平台的登录注册界面设计非常的简单明了，易于操作。商家管理员在注册界面按照要求完成注册后，在登录界面输入账号和密码，点击登录事件后，系统首先获取输入框里的内容，之后在数据库内进行查找，如果匹配成功即可进入商家管理平台首页。

商家管理平台注册界面如图 4-13 所示

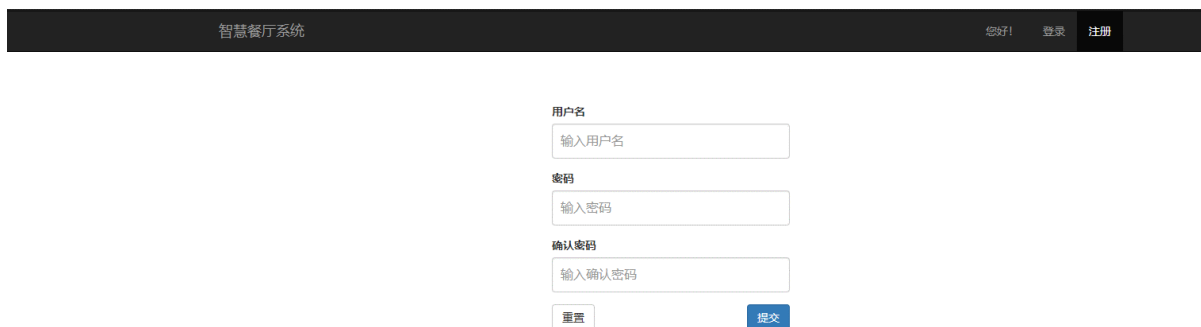
The image shows the registration interface of the Merchant Management Platform. At the top, there is a dark header bar with the text '智慧餐厅系统' (Smart Restaurant System) on the left and '您好! 登录 注册' (Hello! Login Register) on the right. The main content area is white and contains three input fields: '用户名' (Username) with placeholder text '输入用户名', '密码' (Password) with placeholder text '输入密码', and '确认密码' (Confirm Password) with placeholder text '输入确认密码'. Below these fields are two buttons: a '重置' (Reset) button and a '提交' (Submit) button.

图 4-13 商家管理平台注册界面

商家管理平台登录界面如图 4-14 所示

The image shows the login interface of the Merchant Management Platform. It has the same dark header bar as the registration page, with '智慧餐厅系统' (Smart Restaurant System) on the left and '您好! 登录 注册' (Hello! Login Register) on the right. The main content area is white and contains two input fields: '用户名' (Username) with placeholder text '输入用户名' and '密码' (Password) with placeholder text '输入密码'. Below these fields are two buttons: a '重置' (Reset) button and a '登录' (Login) button.

图 4-14 商家管理平台登录界面

4.3.2 查看当前订单功能

(1) 查看当前订单功能描述

商家管理员登录成功后进入商家管理平台主界，点击左侧的查看当前订单按钮，即可实时的显示当前订单详情。订单详情主要包含桌号，人数，菜品，总价，订单时间，服务等信息。

(2) 查看当前订单功能界面设计

商家管理平台的查看当前订单功能界面设计清晰明了，商家管理员可以直观的查看当前订单详情，清楚的了解每个餐桌的情况，当用餐接收后，客人完成了支付功能，当前订单界面会自动将完成的订单删除，并将该订单添加到历史订单界面里。

商家管理平台查看当前订单界面如图 4-15 所示

订单详情						
桌号	人数	菜品	总价	订单时间	服务	
5	5	牛油火锅*1 毛肚*1 土豆*1 娃娃菜*1 金针菇*1 藕片*1	166.00	2021-03-25 20:36:50	无	收到
9	9	鸳鸯火锅*1 虾滑*1 毛肚*1 火锅牛排*1 土豆*1	206.00	2021-03-25 20:34:45	无	收到

图 4-15 商家管理平台查看当前订单界面

4.3.3 查看历史订单功能

(1) 查看历史订单功能描述

商家管理员登录成功后进入商家管理平台主界，点击左侧的查看历史订单按钮，即可显示历史订单详情。订单详情主要包含编号，桌号，人数，菜品，总价等信息。

(2) 查看历史订单功能界面设计

商家管理平台的查看当前订单功能界面设计和查看当前订单界面类似，商家管理员可以清楚的查看历史订单详情，该界面统计了已经完成的订单，方便商家管理员对以往的数据进行统计分析等等。

商家管理平台查看历史订单界面如图 4-16 所示

历史订单详情						
编号	桌号	人数	菜品	总价	订单时间	
7	3	3	牛油火锅*1 鸳鸯火锅*1 菌汤*1 番茄*1 三鲜*1 虾滑*1 毛肚*1 火锅牛排*1 精品小肥羊*1 秘制羊肉*1 土豆*1 娃娃菜*1 金针菇*1 藕片*1 甜玉米*1	582.00	2021-03-22 23:23:38	
8	2	2	牛油火锅*1 虾滑*1 毛肚*1 金针菇*1	169.00	2021-04-08 20:39:23	
9	2	5	牛油火锅*1 鸳鸯火锅*1 菌汤锅底*1 番茄锅底*1 三鲜锅底*1 虾滑*1 毛肚*1 火锅牛排*1 精品小肥羊*1 秘制羊肉*1 土豆*1 娃娃菜*1 金针菇*1 藕片*1 甜玉米*1	582.00	2021-04-09 12:39:45	
10	3	6	牛油火锅*1 虾滑*1 土豆*1	119.00	2021-04-09 14:55:04	

图 4-16 商家管理平台查看历史订单界面

4.3.4 智能分析功能

(1) 智能分析功能描述

商家管理员登录成功后进入商家管理平台主界面，点击左侧的统计按钮，即可实时

的显示对餐厅人数，点餐信息等的统计。方便商家管理员分析比对，做出科学的决策。

(2) 智能分析功能界面设计

商家管理平台的智能分析界面设计非常的简单明了。商家管理员在商家管理平台主界面点击统计按钮，即可动态地显示对餐厅人数，点餐信息等的统计分析。

商家管理智能分析界面如图 4-17 所示

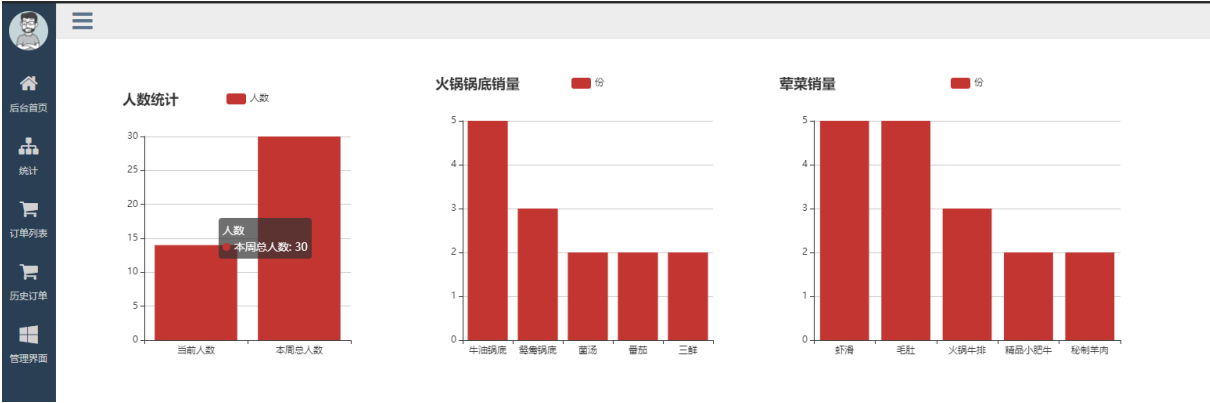


图 4-17 商家管理平台智能分析界面

4.3.5 查看室内环境状态功能

(1) 查看室内环境状态功能描述

商家管理员登录成功后进入商家管理平台主界面，点击左侧管理界面的按钮，即可实时的显示对餐厅环境检测的状态，商家管理员可以更方便的查看餐厅的环境，可以在界面上远程控制灯的开关，如果发生火灾险情，会及时的发现并处理。

(2) 查看室内环境状态功能界面设计

商家管理平台的查看室内环境状态界面设计的十分简单明了，商家管理员可以清楚的查看火焰传感器，烟雾传感器的状态，以及对灯的控制，易于操作。

商家管理平台查看室内环境状态界面如图 4-18 所示



图 4-18 查看室内环境状态界面

4.4 服务器端设计

4.4.1 点餐 APP 服务器设计

商家管理平台需要处理点餐 APP 客户端发送过来的各种请求,包括订单信息提交、是否呼叫服务员、订单结算支付等功能。因此我们需要建立一个 TCP/IP 服务器端来一直接收客户端的数据传送。由于就餐高峰期,客人人流量大,所以服务器采用多线程的方式接收客户端发送的数据。服务器与客户端通信示意图如图 4-19 所示:

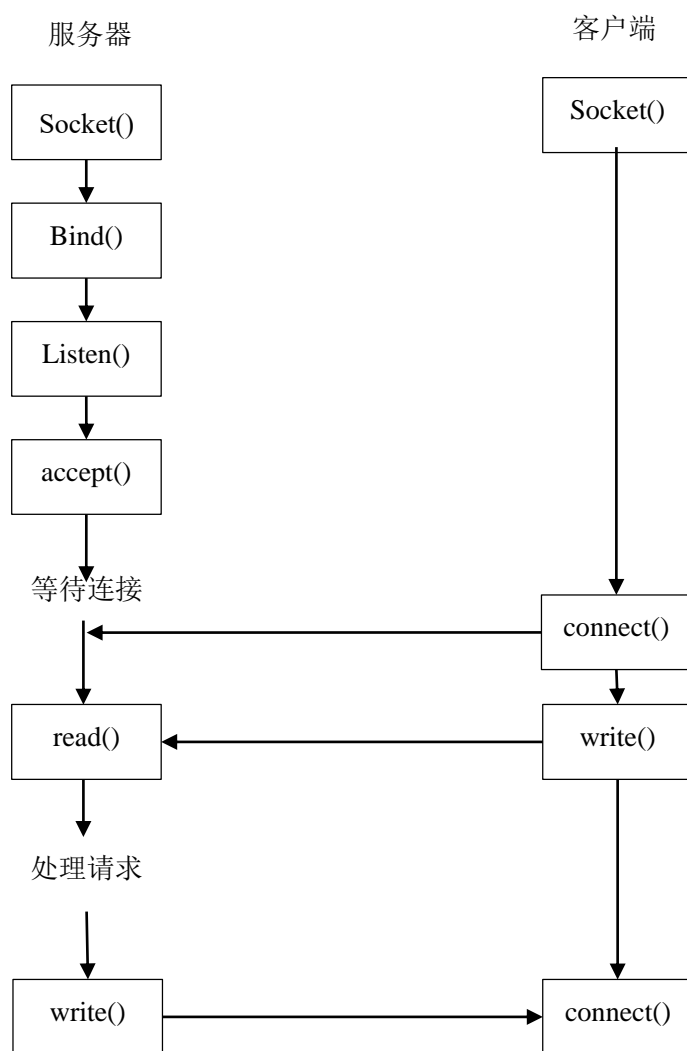


图 4-19 通信流程图

点餐 APP TCP/IP 服务器核心代码:

```
_____piserver.py

tcp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

tcp_server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, True)

tcp_server_socket.bind((get_host_ip(), 8080))
```

```
tcp_server_socket.listen(10)
```

```
while True:
```

```
    print('新用户[%s]连接' % str(ip_port))
```

```
    new_tcp_socket, ip_port = tcp_server_socket.accept()
```

```
    thread_msg = threading.Thread(target=recv_msg, args=(new_tcp_socket, ip_port))
```

```
    thread_msg.setDaemon(True)
```

```
    thread_msg.start()
```

piserver.py

4.4.2 室内环境检测装置服务器设计

商家管理平台同时需要处理物联网模块发送的数据，并实时的显示在网页上。商家管理平台为服务器端，室内环境检测装置为客户端，客户端不断检测室内环境，收集数据，一定时间间隔将数据发送到服务器端，服务器端接收到数据后，存入到数据库里，并实时的更新在网页上，供商家管理员及时的查看。

室内环境检测装置 TCP/IP 服务器核心代码：

piserver.py

```
tcp_server_socket.listen(10)
```

```
while True:
```

```
    new_tcp_socket, ip_port = tcp_server_socket.accept()
```

```
    thread_msg = threading.Thread(target=recv_msg, args=(new_tcp_socket, ip_port))
```

```
    thread_msg1 = threading.Thread(target=send_msg_led1, )
```

```
    thread_msg.setDaemon(True)
```

```
    thread_msg1.setDaemon(True)
```

```
    thread_msg.start()
```

```
    thread_msg1.start()
```

piserver.py

4.5 数据库设计

4.5.1 数据库概念结构设计

对于系统要实现数据的存储、访问等数据库操作，必须首先进行数据的连接。在软件设计中，数据是最核心的组成部分。在本设计中，主要有客人实体和座位实体。UserId 用来唯一标记客人，对当前系统唯一。客人的账号属性，密码属性，不可为空也不能重

复。为座位实体进行编号 `seatId` 作为主码，拥有人数属性、点餐信息属性、总价属性、订单日期属性、是否呼叫服务员标记属性。是否呼叫服务员标记是用来标记当前客人是否有呼叫服务。

客人实体和座位实体分别如图 4-20、如图 4-21 所示。

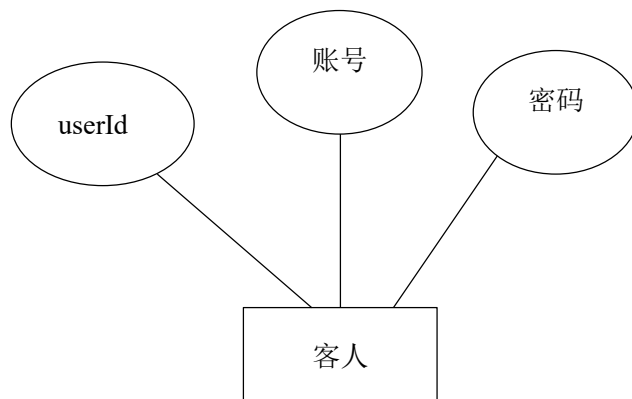


图 4-20 客人实体图

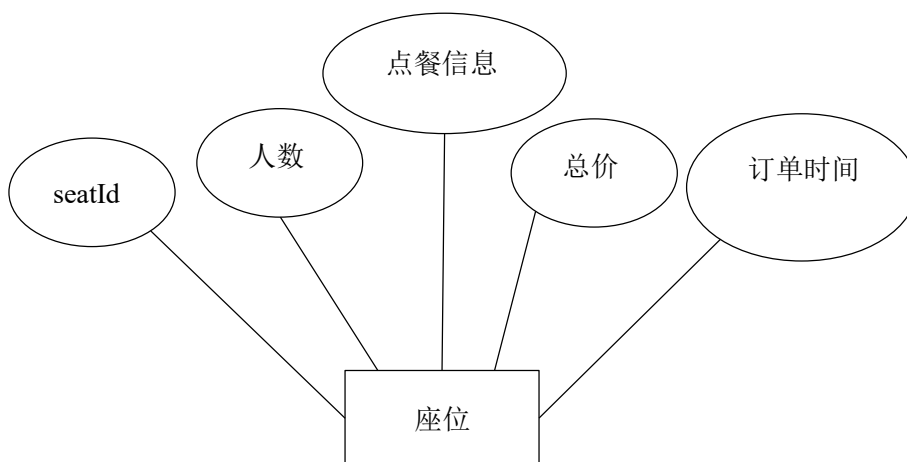


图-21 座位实体图

4.5.2 数据库逻辑结构设计

客人信息表如表 4-1 所示。

表 4-1 客人信息表 (User)

客人编号	账号名称	密码
userId(char)	UserName(char)	password (char)
1	小刘	0324383
2	小杨	5201525

座位信息表如 4-2 所示。

表 4-2 座位信息表 (Seat)

座位编号	人数	点餐信息	总价	订单日期	标志
seatId(char)	Num(int)	orders(char)	Prices(int)	date(TimeStamp)	flag(char)
01	5	鸳鸯火锅*1	206.00	2021-03-25 20:34:45	无
		虾滑*1			
		毛肚*1			
		火锅牛排*1			
		土豆*1			
02	6	牛油火锅*1	166.00	2021-03-25 20:36:50	呼叫服 务员
		毛肚*1			
		土豆*1			
		娃娃菜*1			
		金针菇*1			
		藕片*1			

5. 送餐机器人的详细设计

5.1 激光雷达节点

5.1.1 激光雷达介绍

在送餐机器人送餐的过程中，激光雷达传感器相当于人的“眼睛”，具有非常准确的测距能力，测距精度可以达到几厘米，本论文设计中使用的激光雷达传感器采用的是思岚科技的 RPLIDAR A1 激光雷达传感器，12 米半径的测量范围，8000 次/秒测量频率等性能完全满足设计需求。在树莓派固件中编写和编译 ROS 的激光雷达节点。激光雷达配置如表 5-1 所示：

表 5-1 激光雷达配置表

Items	Specification
Supply Voltage	5VDC \pm 5%
Power consumption	Less than 700mA
Measuring distance	120~3,500mm
Distance accuracy	120~499mm: \pm 15mm 500~3,500mm: \pm 5%
Distance precision	120~499mm: \pm 10mm 500~3,500mm: \pm 3.5%
Angular resolution	Step angle: approx, 1° (360° /360steps)
Safety	Lower safety class 1
Scan speed	300 \pm 10rpm
Interface	3.3V UART(230,400bps)
Weight	About 125g

5.1.2 激光雷达消息格式

在 ROS 开发中传感器数据都要被封装为相应格式的消息。雷达的消息格式如图 5-1 所示：

```

huanyu@ubuntu:~$ rostopic type /scan
sensor_msgs/LaserScan
huanyu@ubuntu:~$ rosmmsg show sensor_msgs/LaserScan
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities

```

图 5-1 消息格式表

5.2 送餐机器人启动节点

送餐机器人启动节点源码包位于 /home/huike/robot_ws/src/ 路径下，包名是 huanyu_robot_start，这个包是 ROS 中最重要的一個包。送餐机器人使用里程计主要作用是读取底盘驱动板的数据，发布里程计和 imu 的等话题。提供准确的里程计信息是非常重要的。送餐机器人的控制结构如图 5-2 所示，送餐机器人启动节点通过串口通信接口将数据发送给 STM32 控制板，接下来进行解算分析，得出每个轮子的速度，电机驱动主板通过传递的值控制电机的转速。

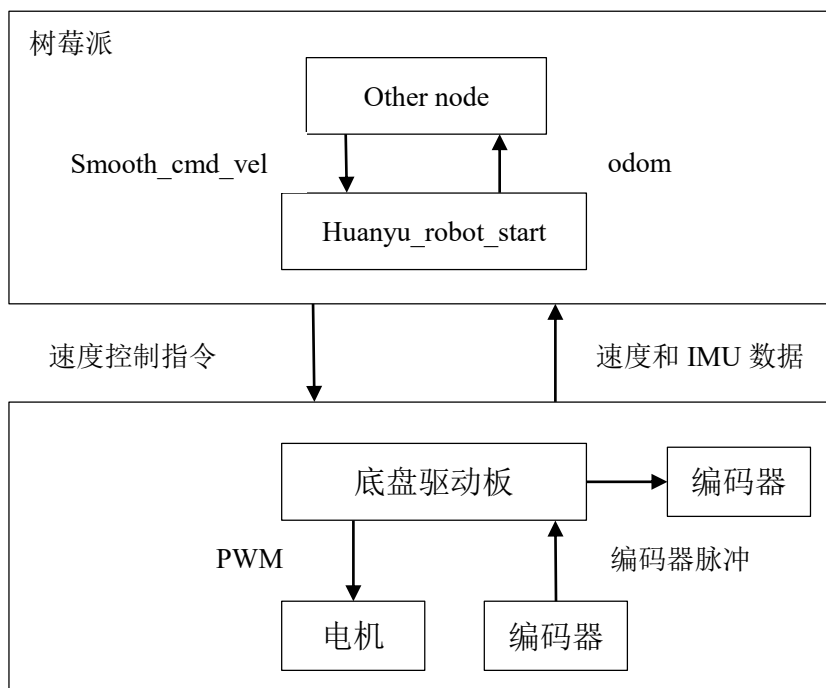


图 5-2 控制结构图

5.3 送餐机器人开发环境配置

由于 ROS 的分布式开发方式，我们也需要在 Ubuntu 主机上安装和树莓派相同版本的 ROS 系统。送餐机器人上电后，等待树莓派启动，启动后我们就可以查看到树莓派开放出来的 WIFI。用 Ubuntu 主机连接开放的 WIFI。确保 WIFI 已经正常连接，需要配置 Ubuntu 主机和树莓派之间的 ROS 网络配置。提供的树莓派固件已经配置和安装了 ssh，Ubuntu 主机可以通过 ssh 远程开发机器人了，首先 Ubuntu 主机需要安装 ssh 客户端。输入树莓派用户的登陆密码后，ssh 远程已经成功连接了。树莓派的固件中已经安装和配置了 nfs 服务器，在 Ubuntu 主机上我们只需要安装 nfs 客户端。

5.4 STM32 和 ROS 通信

5.4.1 通信内容和原理

底盘驱动板和 ROS 的通信是基于串口通信的，我们要将机器人的底层里程计、电池状态、IMU 姿态信息等数据提交到 ROS 系统当中。如图 5-3 所示。在定义 STM32 和 ROS 之间的通信协议时，需要考虑传输的速度、大小、数据内容等问题，然后进行统一的封装，最后调用串口发送函数将封装好的数据按照字节序发送出去。在数据通信协议的定义上，使用一个 union 类型来定义两个相同大小的域，一个是数据收发的缓存区数据 buffer，另一个是数据域结构体。当两个域的大小相同时（同时约定内存对齐为 1 个字节），这时只需要通过串口去操作收发缓存数据，即可完成数据的透传过程。

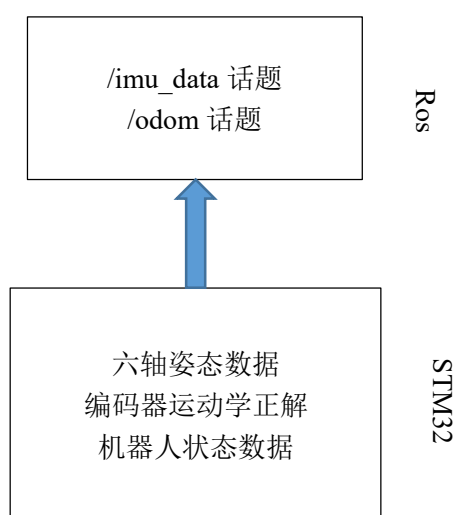


图 5-3 通信框架

5.4.2 STM32 的串口通信

在传输过程中，每一位数据占用固定的时间。串行接口的功能是将外部设备与 CPU

相连，通过串行传输实现信息的收发。如下图 5-4 所示：对于两个进行通信的端口，这些参数必须匹配。

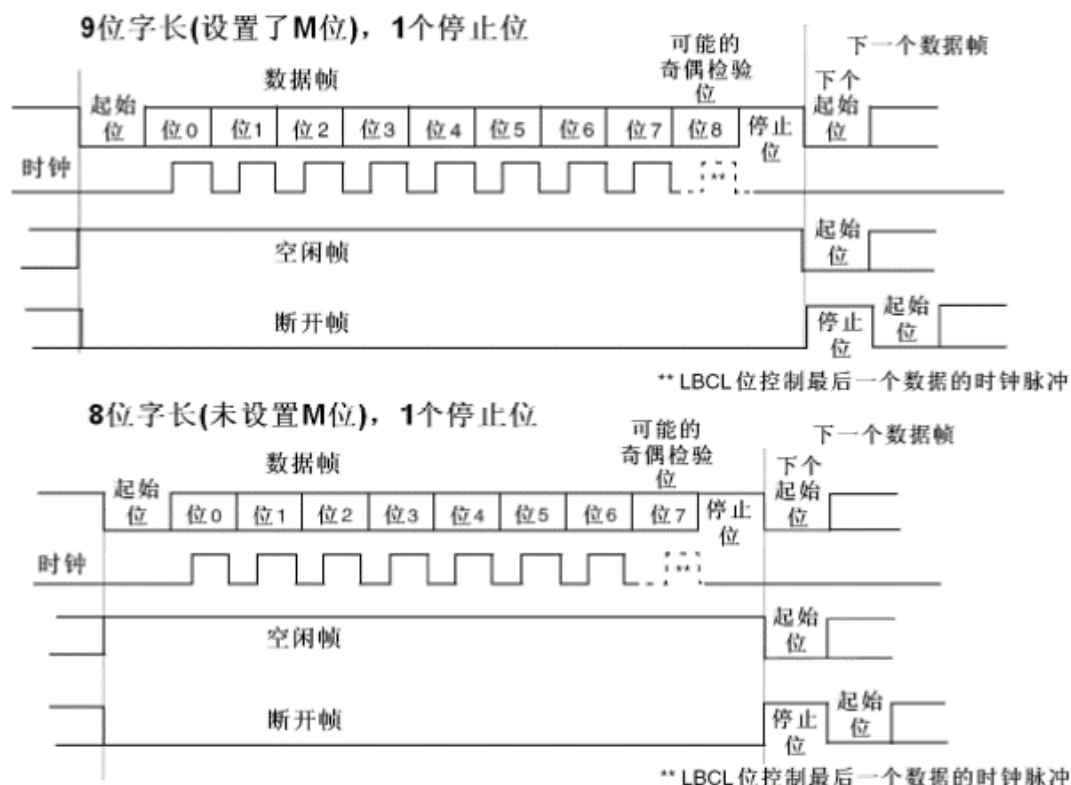


图 5-4 通信参数图

串行端口按位发送和接收字节。串行通信虽然比并行通信慢，但串行端口可以在一条线上发送数据，其他线路上用来接收数据^[11]。同时它可以用来进行长距离通信。通信采用 GND、发送、接收三行来完成。因为串口通信是异步的，所以端口可以在一行上发送数据，在另一行上接收数据。

在底盘驱动板中，初始化了 USART1 的串口接收和发送，用于透传数据到树莓派的 ROS 系统当中，在源码的 Huanyu_usart.c 文件中可以看到串口的初始化配置。配置好各个数据位，我们就可以随时通过串口发送数据。

5.5 Gmapping 建图算法

5.5.1 Gmapping 介绍

对餐厅地图构建包含两部分，送餐机器人的定位和餐厅环境的扫描，首先，我们先了解粒子滤波的原理，送餐机器人不断地通过运动、观测的方式，获取周围环境信息，逐步降低自身位置的不确定度，最终得到准确的定位结果。但是存在致命缺陷，一个是建图对送餐机器人的姿态有比较高的要求，另外一个频繁的重采样导致粒子耗散。

Gmapping 的出现改善了这些缺陷，gmapping 功能包集成了粒子滤波算法，为开发者省去了复杂的内部实现。如图 5-5 所示为 gmapping 功能包的总体框架图：

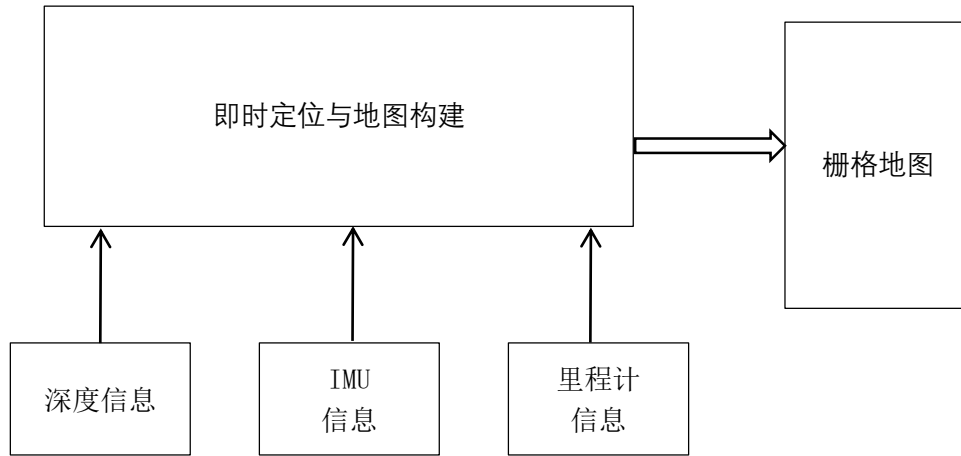


图 5-5 gmapping 功能包总体框架图

5.5.2 Gmapping 功能包实现的前提

(1) 提供了两个 TF。一个是 base_footprint 与 laser 之间的 TF，即机器人底盘与激光雷达之间的转换。我们在 huanyu_robot_start.launch 的开头提供了这个。使用静态 TF 转换启动文件。另一个是 base_footprint 和 odom_combined 之间的 TF，它是底盘和里程表原点之间的坐标转换^[12]。ODOM_combined 可以被认为是里程表开始的坐标系。

(2) /scan: 激光雷达数据，类型为 sensor_msgs/LaserScan。

5.6 送餐机器人局部路径规划算法

5.6.1 DWA 路径规划

送餐机器人的路径规划方法有很多，本论文设计中主要采用的是动态窗口法。送餐机器人通过在速度空间中采样多组速度，并模拟在这些速度下一定时间内的轨迹。在得到多组轨迹以后，对这些轨迹进行评价，选取最优轨迹对应的速度来驱动机器人运动^[13]。

5.6.2 送餐机器人的运动模型

在动态窗口算法中，要模拟机器人的轨迹，需要知道机器人的运动模型^[14]。

推导如下：假设送餐机器人只能做前进、后退和改变方向的动作。计算送餐机器人运动轨迹时，因为两个相邻时刻内时间间隔很短，送餐机器人运动距离非常短，可以把它的运动轨迹看成直线，即送餐机器人移动了 ($v\Delta t$)，要想得到在坐标系移动的位移 Δx 和 Δy ，将该段位移投影到坐标系的 X 轴和 Y 轴即可得出。

$$\Delta x = v\Delta t \cos(\theta t) \quad (5-1)$$

$$\Delta y = v\Delta t \sin(\theta t) \quad (5-2)$$

推算一段时间内的轨迹，只需要将这段时间位移增量累计求和就可以了：

$$x = x + v\Delta t \cos(\theta t) \quad (5-3)$$

$$y = y + v\Delta t \sin(\theta t) \quad (5-4)$$

$$\theta t = \theta t + \omega\Delta t \quad (5-5)$$

送餐机器人是全方位的，它的速度是 y ，计算轨迹方法：送餐机器人坐标系的速度在 y 轴上。只需将送餐机器人在坐标系 y 轴上的行走距离投影到世界坐标系：

$$\Delta x = vy\Delta t \cos\left(\theta t + \frac{\pi}{2}\right) = -vy\Delta t \sin(\theta t) \quad (5-6)$$

$$\Delta y = vy\Delta t \sin\left(\theta t + \frac{\pi}{2}\right) = vy\Delta t \cos(\theta t) \quad (5-7)$$

此时只需要将 y 轴的移动的距离叠加在之前计算的公式上即可：

$$x = x + v\Delta t \cos(\theta t) - vy\Delta t \sin(\theta t) \quad (5-8)$$

$$y = y + v\Delta t \sin(\theta t) + vy\Delta t \cos(\theta t) \quad (5-9)$$

$$\theta t = \theta t + \omega\Delta t \quad (5-10)$$

送餐机器人在相邻时间段的轨迹是圆弧。正如本论文中推导的那样，如果不是全向运动机器人，圆弧运动的半径为：

$$r = \frac{v}{\omega} \quad (5-11)$$

当旋转速度 $\neq 0$ 时，送餐机器人坐标为：

$$x = x - \frac{v}{\omega} \sin(\theta t) + \frac{v}{\omega} \sin(\theta t + \omega\Delta t) \quad (5-12)$$

$$y = y - \frac{v}{\omega} \cos(\theta t) - \frac{v}{\omega} \cos(\theta t + \omega\Delta t) \quad (5-13)$$

还有其他一些推导算法如下：

$$\Delta\theta = \omega\Delta t \quad (5-14)$$

$$\Delta x = v\Delta t \cos\left(\theta + \frac{\Delta\theta}{2}\right) \quad (5-15)$$

$$\Delta y = v\Delta t \sin\left(\theta + \frac{\Delta\theta}{2}\right) \quad (5-16)$$

5.6.3 速度采样

送餐机器人的轨迹运动模型产生后，根据速度就可以推算出轨迹。因此只需采样很多速度，然后分别推算轨迹，然后评价产生轨迹的好坏即可。在速度 (v, ω) 的二维空间中，存在无穷多组速度。但是根据送餐机器人本身的限制和环境限制可以将采样速度

控制在一定范围内^[15]:

送餐机器人本身受最大速度最小速度的限制:

$$Vm = \{v \in [vmin, vmax], \omega \in [\omega min, \omega max]\} \quad (5-17)$$

因为电机转矩有限, 存在最大加减速极限。窗口中的速度为送餐机器人实际能够达到的速度:

$$Vd = \left\{ (v, \omega) \left| \begin{array}{l} v \in [vc - vb\Delta t, vc + va\Delta t] \wedge \\ \omega \in [\omega c - \omega b\Delta t, \omega c + \omega a\Delta t] \end{array} \right. \right\} \quad (5-18)$$

基于送餐机器人的安全考虑: 为了能够在撞到障碍物前停下来, 在最大减速的条件下有一个速度范围:

$$Va = \{(v, \omega) | v \leq \sqrt{2 \cdot dist(v, \omega) \cdot vb} \wedge \omega \leq \sqrt{2 \cdot dist(v, \omega) \cdot \omega b}\} \quad (5-19)$$

其中 $dist(v, w)$ 为速度 (v, w) 对应轨迹上离障碍物最近的距离, 如下图 5-6 弧线所示:

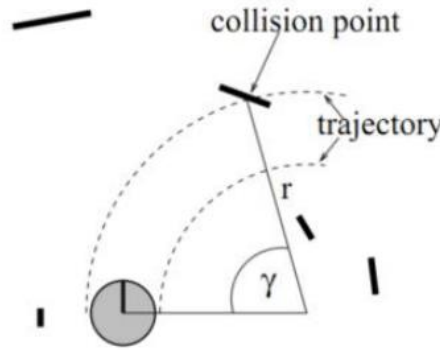


图 5-6 弧线图

6. 系统调试

6.1 Gmapping 建图

gmapping 建图是使用我们机器人上的激光雷达数据来创建环境的栅格地图，下面我们将使用已配置好的算法功能包来创建环境地图：

在建图前，先做以下准备：

首先，将机器人放置在四周 50cm 没有障碍物的地方（否则将影响后面地图在导航的使用）；

然后，打开机器人电源启动机器人；

然后，确认 ubuntu 开发环境已成功连接到机器人发出的 AP；

并确认已配置好 ROS 分布式组网；

并使用 ifconfig 查看开发环境的 IP 是不是和 ~/.bashrc 文件中的 ROS_HOSTNAME 变量的 IP 一样，若不是，则修改 ROS_HOSTNAME 变量的 IP，并 source ~/.bashrc 使之生效；

下面开始创建地图。

（1）启动机器人底盘节点

在 ubuntu 开发环境中使用 Ctrl+Alt+T 快捷键打开终端，然后通过 ssh 远程连接机器人，执行：

```
ssh huike@192.168.12.1
```

远程连接机器人后，则需要更新系统的时钟，命令如下：

```
sudo date -s "2021-1-9 16:15:30" （修改为当前的日期和时间）
```

其中，2021-1-9 16:15:30 为当前的日期和时间，请将其修改为你现在的日期和时间。

然后，我们再使用前面的 date 命令，查看当前系统的时间是否设置成功，命令如下：

```
date
```

然后，输入启动机器人底盘控制器节点命令：

```
roslaunch huanyu_robot_start Huanyu_robot_start.launch
```

接下图 6-1 所示，我们将看到机器人底盘控制器节点启动过程中的日志信息，机器人的雷达传感器、IMU 传感器和里程计等也都一起启动。

```

process[joint_state_publisher-6]: started with pid [7238]
process[robot_state_publisher-7]: started with pid [7241]
process[publish_odom-8]: started with pid [7246]
process[rplidarNode-9]: started with pid [7250]
[ INFO] [1607089057.591435159]: [ZHOUXUEWEI] base controller node start!
process[robot_pose_ekf-10]: started with pid [7257]
process[nodelet_manager-11]: started with pid [7261]
process[velocity_smoother-12]: started with pid [7266]
[ INFO] [1607089057.815737974]: RPLIDAR running on ROS package rplidar_ros. SDK
Version:1.12.0
[ INFO] [1607089057.841575881]: [ZHOUXUEWEI] Serial Port opened
[ INFO] [1607089057.960316159]: output frame: odom_combined
[ INFO] [1607089057.963857048]: base frame: base_footprint
RPLIDAR S/N: AAA99A86C0E09CC9A2E09DF7387B3079
[ INFO] [1607089060.351623843]: Firmware Ver: 1.28
[ INFO] [1607089060.351811861]: Hardware Rev: 5
[ INFO] [1607089060.353473583]: RPLidar health status : 0
[ INFO] [1607089060.983436750]: current scan mode: Boost, max_distance: 12.0 m,
Point number: 8.0K , angle_compensate: 2
[ INFO] [1607089061.181610879]: Initializing Odom sensor
[ INFO] [1607089061.182112675]: Initializing Imu sensor
[ INFO] [1607089061.231422472]: Odom sensor activated
[ INFO] [1607089061.231806138]: Kalman filter initialized with odom measurement
[ INFO] [1607089061.232233879]: Imu sensor activated

```

图 6-1 启动底盘节点成功图

(2) 启动 gmapping 建图节点

使用 Ctrl+Alt+T 快捷键再新打开一个终端，同样通过 ssh 远程连接机器人，然后启动 gmapping 建图节点，命令如下：

```
roslaunch huanyu_robot_start gmapping_slam.launch
```

接下来我们将看到建图节点启动过程的日志，出现“Registering First Scan”，则说明开始建图了，第一帧地图创建成功，如下图 6-2 中红框标记。

```

NODES
/
  slam_gmapping (gmapping/slam_gmapping)

ROS_MASTER_URI=http://192.168.12.1:11311

process[slam_gmapping-1]: started with pid [7139]
[ INFO] [1607084363.417069179]: Laser is mounted upwards.
-maxUrange 4 -maxUrange 5 -sigma 0.05 -kernelSize 3 -lstep 0.05 -lobsGain 3
-astep 0.05
-srr 0.01 -srt 0.02 -str 0.01 -stt 0.02
-linearUpdate 0.05 -angularUpdate 0.0436 -resampleThreshold 0.5
-xmin -1 -xmax 1 -ymin -1 -ymax 1 -delta 0.05 -particles 8
[ INFO] [1607084363.425970587]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0.15 0 -3.13296
n_count 0
Registering First Scan

```

图 6-2 建图节点

(3) 打开 Rviz

使用 Ctrl+Alt+T 快捷键再新打开一个终端，然后输入下面命令来打开 Rviz：

```
rviz
```

然后在打开的 Rviz 软件中，添加显示类型 RobotModel、LaserScan 和 PoseWithCovariance，并将 LaserScan 的 Topic 话题选择为 /scan，PoseWithCovariance 的 Topic 话题选择为 /robot_pose_ekf/odom_combined，以及 Covariance 右边的复选框不勾选，则下面我们将看到界面中出现了机器人的模型、机器人的方向箭头和激光雷达数据：

(4) 使用 arbotix_gui 移动机器人建图

使用 Ctrl+Alt+T 快捷键再新打开一个终端，然后输入 arbotix_gui 命令来打开程序 ArbotiX Controller GUI：

```
arbotix_gui
```

接下来我们就会看到打开的 ArbotiX Controller GUI 界面如下图 6-3 所示。

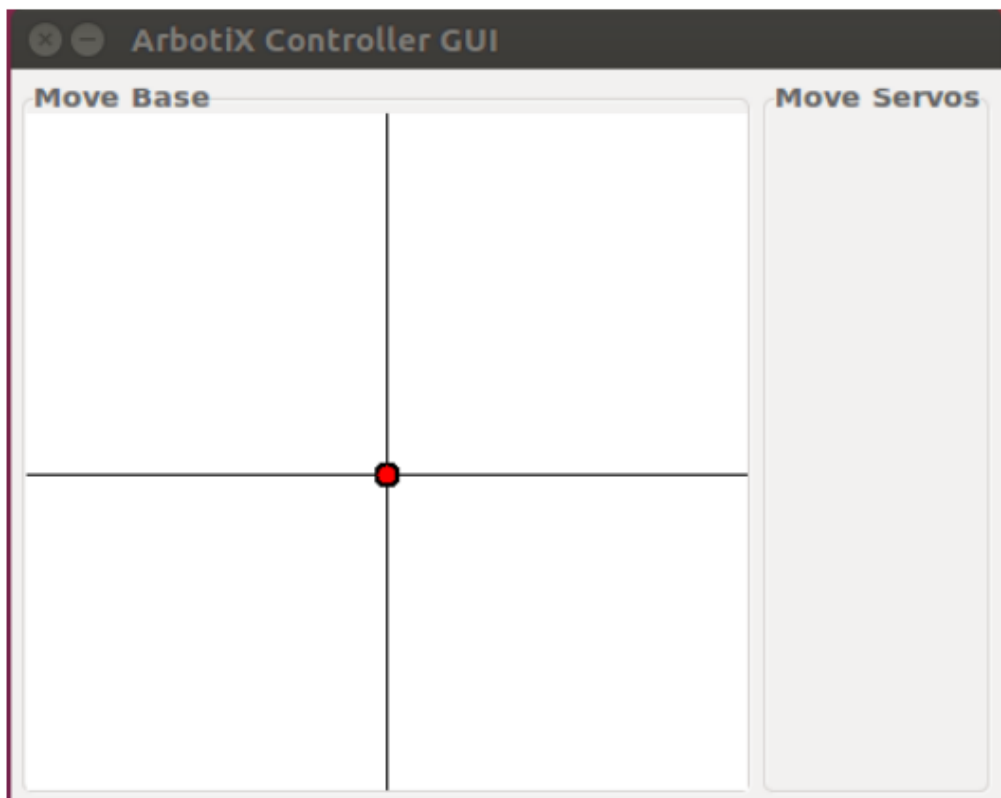


图 6-3 arbotix_gui

然后通过鼠标左键拖动界面中的红点来控制机器人移动，来扫描整个待建图的区域，如下图 6-4 所示，一个小房间内的地图示例。

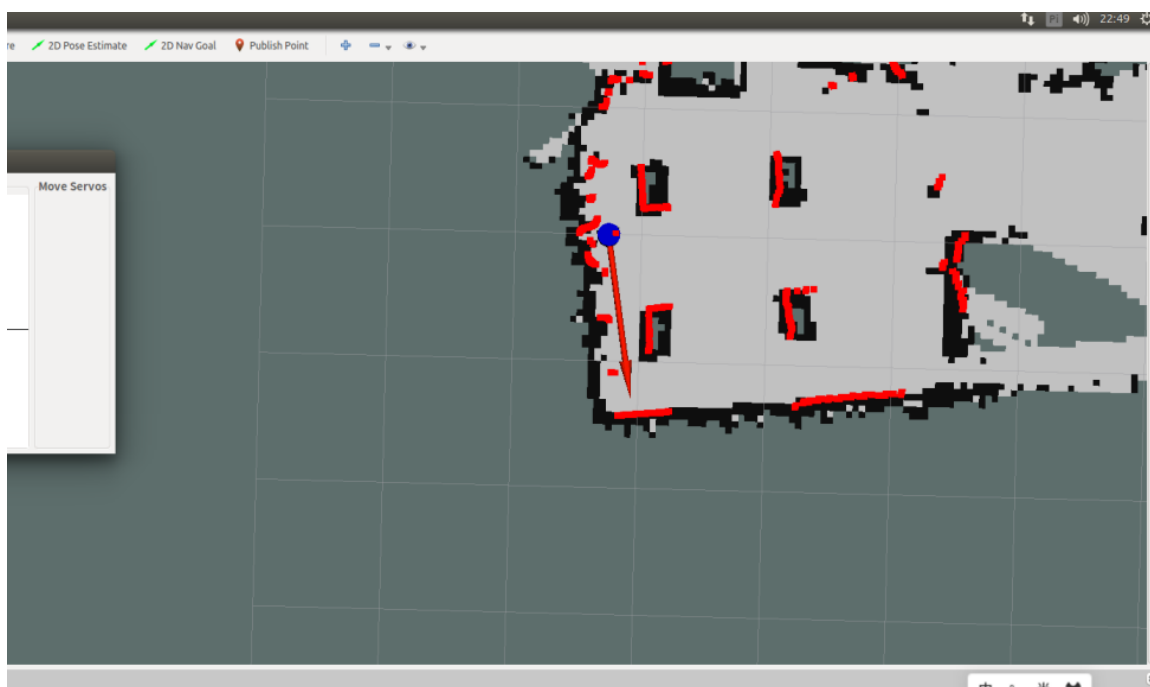


图 6-4 扫描图

待我们扫描完整个待建图区域后，下一步我们将开始保存地图。

(5) 保存地图

使用 `Ctrl+Alt+T` 快捷键再新打开一个终端，然后通过 `ssh` 远程连接机器人，然后首先 `cd` 进入到保存地图的目录下，命令如下：

```
cd robot_ws/src/huanyu_robot_start/map
```

此时我们通过 `ls` 命令可以看到此目录下已有地图，则需要使用 `rm` 将其删除，命令如下：

```
ls
```

```
rm map.*
```

现在我们将保存地图到当前目录下，命令如下：

```
roslaunch map_server map_saver -f map
```

到此，环境地图已经创建完毕，我们可以将前面启动的节点和软件关掉。

6.2 DWA 导航

在导航前，先做以下准备：

首先，将机器人放置在地图原点，且方向与建图时的一样，否则导航前需要初始化机器人的位姿；

然后，打开机器人电源启动机器人；

然后，确认 ubuntu 开发环境已成功连接到机器人发出的 AP；

并确认已配置好 ROS 分布式组网；

并使用 ifconfig 查看开发环境的 IP 是不是和 ~/.bashrc 文件中的 ROS_HOSTNAME 变量的 IP 一样，若不是，则修改 ROS_HOSTNAME 变量的 IP，并 source ~/.bashrc 使之生效。

下面开始送餐机器人导航。

(1) 启动机器人底盘节点

在 ubuntu 开发环境中使用 Ctrl+Alt+T 快捷键打开终端，然后通过 ssh 远程连接机器人，执行：

```
ssh huike@192.168.12.1
```

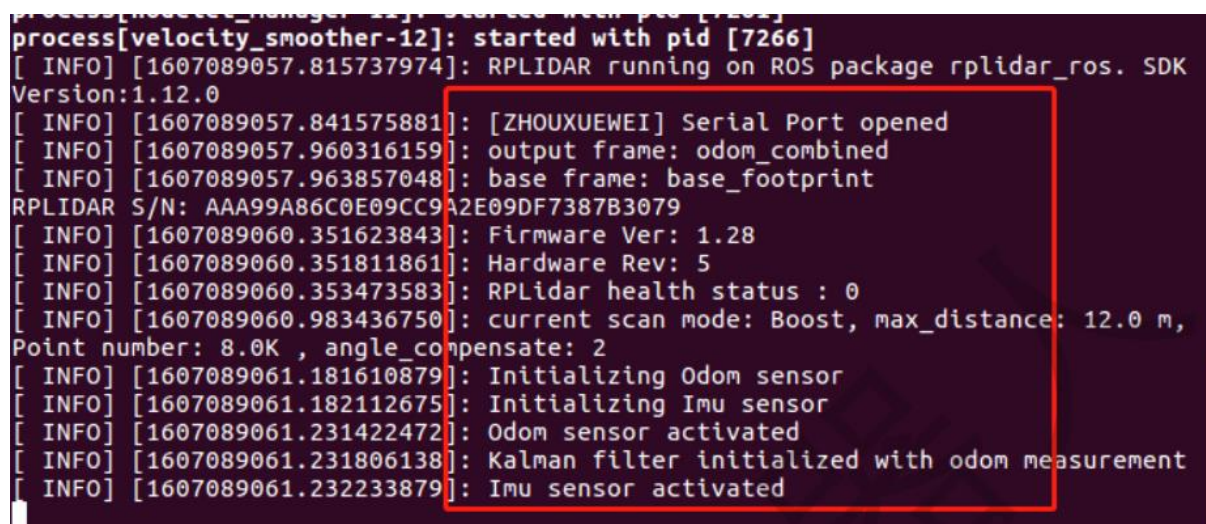
远程连接机器人后，则需要更新系统的时钟，命令如下：

```
sudo date -s "2021-1-9 16:15:30" （修改为当前的日期和时间）
```

然后，输入启动机器人底盘控制器节点命令：

```
roslaunch huanyu_robot_start Huanyu_robot_start.launch
```

接下图 6-5 所示，我们将看到机器人底盘控制器节点启动过程中的日志信息，机器人的雷达传感器、IMU 传感器和里程计等也都一起启动。



```
process[velocity_smoother-12]: started with pid [7266]
[ INFO] [1607089057.815737974]: RPLIDAR running on ROS package rplidar_ros. SDK
Version:1.12.0
[ INFO] [1607089057.841575881]: [ZHOUXUEWEI] Serial Port opened
[ INFO] [1607089057.960316159]: output frame: odom_combined
[ INFO] [1607089057.963857048]: base frame: base_footprint
RPLIDAR S/N: AAA99A86C0E09CC9A2E09DF7387B3079
[ INFO] [1607089060.351623843]: Firmware Ver: 1.28
[ INFO] [1607089060.351811861]: Hardware Rev: 5
[ INFO] [1607089060.353473583]: RPLidar health status : 0
[ INFO] [1607089060.983436750]: current scan mode: Boost, max_distance: 12.0 m,
Point number: 8.0K , angle_compensate: 2
[ INFO] [1607089061.181610879]: Initializing Odom sensor
[ INFO] [1607089061.182112675]: Initializing Imu sensor
[ INFO] [1607089061.231422472]: Odom sensor activated
[ INFO] [1607089061.231806138]: Kalman filter initialized with odom measurement
[ INFO] [1607089061.232233879]: Imu sensor activated
```

图 6-5 启动底盘节点图

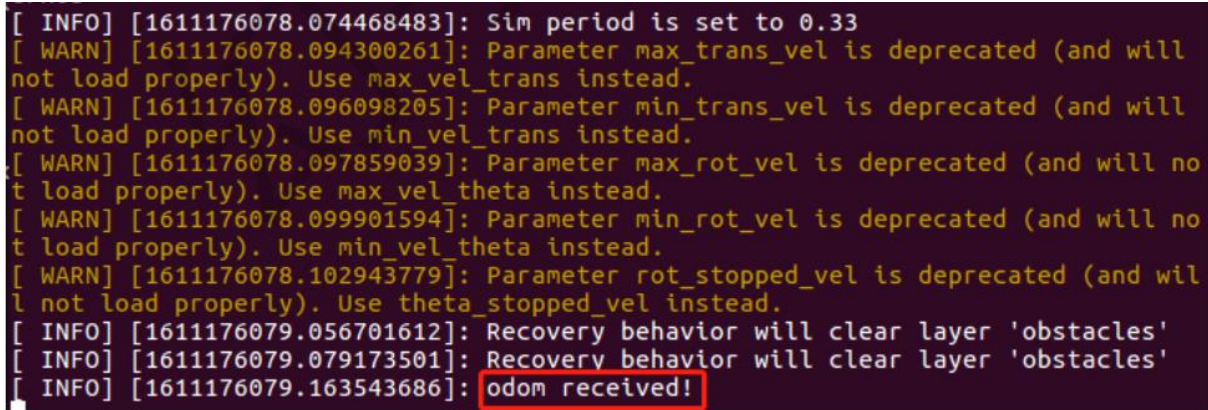
(2) 启动 DWA 导航节点

启动导航节点前，要确保机器人的底盘节点已启动。

使用 Ctrl+Alt+T 快捷键再新打开一个终端，同样通过 ssh 远程连接机器人，然后启动导航节点，命令如下：


```
roslaunch huanyu_robot_start navigation_slam.launch
```

接下来我们将看到导航节点启动过程的日志，当看到“odom received”，则说明导航节点启动成功，如下图 6-6 中红框标注。



```
[ INFO] [1611176078.074468483]: Sim period is set to 0.33
[ WARN] [1611176078.094300261]: Parameter max_trans_vel is deprecated (and will
not load properly). Use max_vel_trans instead.
[ WARN] [1611176078.096098205]: Parameter min_trans_vel is deprecated (and will
not load properly). Use min_vel_trans instead.
[ WARN] [1611176078.097859039]: Parameter max_rot_vel is deprecated (and will no
t load properly). Use max_vel_theta instead.
[ WARN] [1611176078.099901594]: Parameter min_rot_vel is deprecated (and will no
t load properly). Use min_vel_theta instead.
[ WARN] [1611176078.102943779]: Parameter rot_stopped_vel is deprecated (and wil
l not load properly). Use theta_stopped_vel instead.
[ INFO] [1611176079.056701612]: Recovery behavior will clear layer 'obstacles'
[ INFO] [1611176079.079173501]: Recovery behavior will clear layer 'obstacles'
[ INFO] [1611176079.163543686]: odom received!
```

图 6-6 启动导航节点

(3) 打开 Rviz

使用 Ctrl+Alt+T 快捷键再新打开一个终端，然后输入下面命令来打开 Rviz：

```
rviz
```

然后在打开的 Rviz 软件中，设置全局 Fixed Frame 为 map，添加显示类型 RobotModel、LaserScan 和 PoseArray，并将 LaserScan 的 Topic 话题选择为 /scan，MarkerArray 的 Topic 话题选择为 /move_base/TebLocalPlannerROS/teb_poses (用于显示机器人局部规划路径中的机器人位姿)，设置其显示颜色为绿色，则下面我们将看到界面中出现了机器人的模型和激光雷达数据。

(4) 测试 DWA 导航

在前面已启动送餐机器人底盘节点和导航节点后，下面我们来测试送餐机器人的导航效果，发布送餐机器人的导航目标：

在 Rviz 界面中使用鼠标左键点击界面上方工具栏中的 2D 导航按钮，然后出现绿色箭头，首先在地图中点击你想要送餐机器人去的地方，然后来设置送餐机器人到达目标点后的方向，最后松开鼠标，则设置送餐机器人的导航点完成。送餐机器人将接收指令，前往目标点。

7. 总结与展望

7.1 总结

本文主要研究了基于 SLAM 激光雷达的智能送餐机器人系统。为了减少用餐高峰期服务员的工作量以及疫情期间交叉感染的风险，提出了一种基于 SLAM 激光雷达的智能送餐机器人系统。该系统主要由智能点餐系统、室内环境检测装置和送餐机器人三部分构成，其中智能点餐系统包含顾客使用的点餐 APP 和商家使用的商家管理平台，智能点餐系统主要负责处理顾客的订单，其次还包括呼叫服务员功能和一键支付功能等。室内环境检测装置是对餐厅环境进行检测，并将检测结果实时的显示在商家管理平台。送餐机器人主要负责接收商家管理员的指令，将菜送到指定的位置，完成送餐。以上三部分相互配合，通力协作，完成从点餐、送餐及支付整个流程，实现物联网中人和物体的互联互通，从而提高了送餐效率和避免了交叉感染的风险。

由于用餐高峰期，需要处理大量数据，因此在设计送餐机器人系统时优先把稳定性放在首位。在经过大量的模拟测试后，得出如下结果，点餐 APP、室内环境检测装置与商家管理平台之间的数据传输可靠，数据无丢失，为接下来的送餐流程提供稳定性的保障。送餐机器人经过多次测试，能准确到达指定的餐桌不出错，且在送餐过程中遇到行人能及时自动避障，重新规划一条新的路线，完成送餐。

7.2 展望

本文设计的基于 SLAM 激光雷达的送餐机器人与传统的送餐模式相比又优点也有缺点。送餐机器人相比于服务员的送餐效率有明显的提高，以及减少疫情下人与人的交叉感染。但本文设计的送餐机器人还有一定的局限性，灵活性比较差，与客人交互较少，遇到复杂的问题还需要向服务员请求帮助。相信随着智慧时代的到来，在实现主要的送餐功能后，下一步将优化客人就餐的体验，满足更多客人的用餐需求。

从目前国内的智能机器人研究程度来看，送餐机器人有很好的发展空间和前景。近年来，人工智能发展速度十分迅速，人工智能给予了机器人的基本功能，比如视觉，听觉等。经过优化和改进，使得送餐机器人越来越智能化，相信不久的将来，送餐机器人将服务于平民大众。

参考文献

- [1] 沃佳龙. 基于 Arduino 平台的送餐机器人控制系统设计与实现[D].南昌大学,2018.
- [2] 赵玫,王建华.行业无形资产差异分析[J].物流工程与管理,2017,39(03):144-147.
- [3] 谢景明. Android 移动开发项目式教程[M]. 人民邮电出版社, 2015.
- [4] 陈明. 数据科学与大数据技术导论实验[M]. 北京: 北京师范大学出版社 , 2018.07.
- [5] 邓杰海, 全智龙, 周红娟. 基于 C#与 SQLite 的银行财政非税收入管理软件的研发[J]. 电脑知识与技术, 2017, 13(010):74-75.
- [6] 杨连贺;董禹龙;房超;毕璐琪;梁润宇;杨阳;彭进香. 全国高等院校应用型创新规划教材·计算机系列 Python 程序设计实用教程[M]. 北京: 清华大学出版社 , 2018.06.
- [7] 汪乐章,林娴,唐伊文,张国平.基于树莓派与计算机视觉的家庭火灾报警系统的设计与研究[J].电子测量技术,2019,42(08):83-87.
- [8] 史永胜, 胡双, 许梦芸,等. 一种查表与插值法在微控制器中的实现[J]. 陕西科技大学学报:自然科学版, 2015, 000(002):148-153.
- [9] 全国大学生电子设计竞赛组委会. 第十一届全国大学生电子设计竞赛获奖作品选编. 2013. 本科组[M]. 北京理工大学出版社有限责任公司, 2015.
- [10] 邱建全. 基于移动互联智慧学生餐厅的设计与应用[D].华侨大学,2015.
- [11] 袁东明, 史晓东, 陈凌霄. 现代数字电路与逻辑设计实验教程[M]. 北京邮电大学出版社, 2011.
- [12] 邹诗楠, 陈泽义, 王晓瀛,等. 基于 ROS 的 AGV 定位导航系统制作[J]. 电子元器件与信息技术, 2020, 004(001):P.59-61.
- [13] 张永妮. 智能机器人避障路径规划算法研究[J]. 中小企业管理与科技(上旬刊), 2016.
- [14] 中国国防工业企业协会. 2018 年地面无人系统大会论文集[C]. 北京: 科学技术文献出版社, 2018.12.
- [15] 刘彪, 柏林, 周科. 警用巡逻机器人导航系统设计及关键技术研究[J]. 中国安全防范技术与应用, 2018(3).
- [16] Robot-assisted feeding: A technical application that combines learning from demonstration and visual interaction.[J]. Technology and health care : official journal of the European Society for Engineering and Medicine,2020.

- [17] Liu Fei,Yu Hongliu,Wei Wentao,Qin Changcheng. I-feed: A robotic platform of an assistive feeding robot for the disabled elderly population[J]. Technology and Health Care,2020,28(4).
- [18] Priyam A. Parikh, Reena Trivedi,Jatin Dave. Trajectory Planning for the Five Degree of Freedom Feeding Robot Using Septic and Nonic Functions[J]. International Journal of Mechanical Engineering and Robotics Research,2020,9(7).

附 录

部分核心代码:

SendAsyncTask.java

```
package com.mhy.shoppingcar.adapter;

import android.os.AsyncTask;

import java.io.IOException;

import java.io.PrintStream;

import java.net.Socket;

public class SendAsyncTask extends AsyncTask<String, Void, Void> {

    @Override

    protected Void doInBackground(String... params) {

        String str = params[0];

        try {

            Socket client = new Socket("172.20.10.2", 8080);

            client.setSoTimeout(5000); // 获取 Socket 的输出流，用来发送数据给服务端

            PrintStream out = new PrintStream(client.getOutputStream());

            out.print(str);

            out.flush();

            out.close();

            client.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

        return null;

    }

}
```

SendAsyncTask.java

```

protected void onCreate(@Nullable Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_arc_menu);

    btn_tiancai = findViewById(R.id.tiancai);
    btn_tiancai.setOnClickListener(this);

    listView = findViewById(R.id.choosed);
    TextView sum = (TextView) findViewById(R.id.sum);

    if (MainActivity.total_price > 0) {

        List<Map<String, Object>> chooses = new ArrayList<>();

        ArrayList<Integer> lists = getIntent().getIntegerArrayListExtra("goods");

        for (int i = 0; i < lists.size(); i++) {

            if (lists.get(i) > 0) {

                Map<String, Object> list = new HashMap<>();

                list.put("name", goods_names[i]);

                list.put("num", lists.get(i));

                list.put("price", goods_price[i]);

                list.put("image", goods_imagges[i]);

                chooses.add(list);

            }

        }

        listView.setVisibility(View.VISIBLE);

        listView.setAdapter(new SimpleAdapter(this, chooses, R.layout.item_choosed,
            new String[]{"name", "num", "price", "image"}, new int[]{R.id.tv_name,
R.id.tv_edit_buy_number, R.id.tv_price, R.id.iv_photo}));

        btn_tiancai.setVisibility(View.VISIBLE);

        DecimalFormat decimalFormat = new DecimalFormat("0.00");

        sum.setText("当前消费" + decimalFormat.format(MainActivity.total_price) + "元");

    } else {

        listView.setVisibility(View.INVISIBLE);
    }
}

```

```

        btn_tiancai.setVisibility(View.INVISIBLE);

        sum.setText("欢迎光临! ");
    }

    ActionBar actionBar = getSupportActionBar();
    if (actionBar != null) {
        actionBar.setTitle("YArcMenuView");
    }

    mArcMenuView = findViewById(R.id.arc_menu);
    List<Integer> menuItems = new ArrayList<>();
    menuItems.add(R.drawable.zhifu);
    menuItems.add(R.drawable.diancan);
    menuItems.add(R.drawable.hujiao);
    mArcMenuView.setMenuItems(menuItems);
    mArcMenuView.setOnClickListener(new YArcMenuView.ClickMenuListener() {
        @Override
        public void clickMenuItem(int resId) {
            switch (resId) {
                case R.drawable.zhifu:
                    Toast.makeText(getApplicationContext()," 点 击 了 支 付 ",
Toast.LENGTH_SHORT).show();

                    new SendAsyncTask().execute("C" + sendData);
                    try {
                        Uri uri = Uri.parse("alipayqr://platformapi/startapp?saId=10000007");
                        Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                        startActivity(intent);
                    } catch (Exception e) {
                        Toast.makeText(ArcMenuTestActivity.this,"打开失败，请检查是否安
装了支付宝", Toast.LENGTH_SHORT).show();
                    }
                    break;

                case R.drawable.diancan:

```

```

        Toast.makeText(getApplicationContext()," 点 击 了 点 餐 ",
Toast.LENGTH_SHORT).show();

        Intent intent = new Intent(ArcMenuTestActivity.this, MainActivity.class);

        startActivity(intent);

        break;

    case R.drawable.hujiao:

        Toast.makeText(getApplicationContext()," 点 击 了 呼 叫 服 务 员 ",
Toast.LENGTH_SHORT).show();

        new SendAsyncTask().execute("B" + sendData);

        break;

    }

}

});

}

```

ArcMenuTestActivity.java

Appserver.py

```

import socket

import sqlite3

import threading

from collections import Counter

from common import get_host_ip

con = sqlite3.connect('db.db', check_same_thread=False)

cursor = con.cursor()

caiminglist = ['牛油火锅', '鸳鸯火锅', '菌汤锅底', '番茄锅底', '三鲜锅底', '虾滑', '毛肚', '火锅牛排', '精品
小肥羊', '秘制羊肉', '土豆', '娃娃菜', '金针菇', '藕片',

                '甜玉米']

lock = threading.Lock()

# 处理菜名数据

def bao_cai(recv_text):

    # 桌号

```



```

zhuohao = recv_text[1:recv_text.find("!")]

print("桌号为: ", zhuohao)

# 人数
renshu = recv_text[recv_text.find("!") + 1:recv_text.find("&")]

print("人数为: ", renshu)

# 菜名代号
caimingdaihao = recv_text[recv_text.find("&") + 1:recv_text.find("#")]

print("菜名代号: ", caimingdaihao)

# 价格
jiage = recv_text[recv_text.find("#") + 1:]

print("价格: ", jiage)

d1 = []

d2 = []

caiming = ""

n = len(caimingdaihao)

l = [caimingdaihao[i - 1] + caimingdaihao[i] for i in range(n) if i % 2 != 0]

for i in l:

    d1.append(caiminglist[int(i)])

    if i[0] == "0":

        d2.append(int(i))

print(d1)

daihao = ".join(str(i) for i in d2)

print(daihao)

count = Counter(list(d1))

for key, value in count.items():

    caiming += (key + "*" + str(value) + "\n")

print("菜名: ", caiming)

lock.acquire(True)

cursor.execute("Insert or replace into

```

CNAME(ID,renshu,NUM,SUM,daihao)

```

        values(?,?,?,?,?)",
        (zhuohao, renshu, caiming, jiage, daihao))

    con.commit()

    lock.release()

def recv_msg(new_tcp_socket, ip_port):
    recv_data = new_tcp_socket.recv(1024)
    while recv_data:
        recv_text = recv_data.decode('gbk')
        print('来自[%s]的信息: %s' % (str(ip_port), recv_text))
        if recv_text[0] == "A": # 接收菜名
            bao_cai(recv_text)
        elif recv_text[0] == "B":
            lock.acquire(True)
            cursor.execute("""UPDATE CNAME
                            SET fuwu = '呼叫服务员'
                            WHERE ID = (?);""", (recv_text[1]))
            con.commit()
            lock.release()
        elif recv_text[0] == "C":
            lock.acquire(True)
            cursor.execute("""INSERT INTO lishi (ID,renshu, NUM,SUM,CreateTime,daihao)
                            SELECT ID,renshu, NUM,SUM,CreateTime,daihao
                            FROM CNAME
                            WHERE ID = (?);
                            """, (recv_text[1]))
            cursor.execute("""DELETE FROM CNAME WHERE ID = (?);
                            """, (recv_text[1]))
            con.commit()
            lock.release()
    recv_data = new_tcp_socket.recv(1024)

```

```
new_tcp_socket.close()

print("ip 地址为:", get_host_ip())

tcp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

tcp_server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, True)

tcp_server_socket.bind((get_host_ip(), 8080))

tcp_server_socket.listen(10)

while True:

    new_tcp_socket, ip_port = tcp_server_socket.accept()

    print('新用户[%s]连接' % str(ip_port))

    thread_msg = threading.Thread(target=recv_msg, args=(new_tcp_socket, ip_port))

    thread_msg.setDaemon(True)

    thread_msg.start()
```

_appserver.py

致 谢

行文至此，落笔为终。四年大学生活即将结束，在许昌学院留下许多美好的回忆，纵使万般不舍，但心存感激，我将开始新的征途。

首先我要感谢我的论文指导老师张向群导师，从选题到硬件的选材，从设计提纲到最终的定稿，每一部分都离不开张向群老师的耐心指导。在此衷心感谢张向群老师！

其次，我要感谢我大学导师程菊明老师、宋运隆老师以及全体物联网工程专业的老师。老师是我们最好的朋友，他们引领我前进的方向，一路指导帮助我们，回忆当初老师带领我们参加比赛的场景，历历在目。再次感谢老师们的指导！

最后，感谢陪伴我大学四年的室友和同学，感谢我的父母对我的无私付出和全力支持，在面对困难时给予我鼓励，让我拥有跌倒了站起来的勇气。我会不断努力，成为你们的骄傲。愿你们身体健康，未来可期！

山水一程，三生有幸！言辞有尽，感谢一切！

许昌学院，我们后会有期！