

大连理工大学本科毕业设计（论文）

基于深度学习的嵌入式语音控制无人机设计

Embedded Voice Control Drone Design Based on Deep Learning

学 院（系）： 电气工程学院

专 业： 电气工程及其自动化

学 生 姓 名： oscarxchen

学 号： 201781000

指 导 教 师： 刘老师

评 阅 教 师： 李老师

完 成 日 期： 2021.6.22

大连理工大学

Dalian University of Technology

原创性声明

本人郑重声明：本人所呈交的毕业设计（论文），是在指导老师的指导下独立进行研究所取得的成果。毕业设计（论文）中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究成果做出重要贡献的个人和集体，均已在文中以明确方式标明。

本声明的法律责任由本人承担。

作者签名：

日期：2021.6.21

关于使用授权的声明

本人在指导老师指导下所完成的毕业设计（论文）及相关的资料（包括图纸、试验记录、原始数据、实物照片、图片、录音带、设计手稿等），知识产权归属大连理工大学。本人完全了解大连理工大学有关保存、使用毕业设计（论文）的规定，本人授权大连理工大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业设计（论文）。如果发表相关成果，一定征得指导教师同意，且第一署名单位为大连理工大学。本人离校后使用毕业毕业设计（论文）或与该论文直接相关的学术论文或成果时，第一署名单位仍然为大连理工大学。

论文作者签名：

日 期：2021.6.21

指导老师签名：

日 期：2021.6.21

摘 要

无人机作为一种经济方便的飞行器已经在如物流、航拍、安防等众多领域发挥着重要的作用，但时至今日无人机的操作门槛依然很高，培养合格的无人机操作员需要大量的时间和金钱成本。本课题设计了一种使用用户的口授语音指令对无人机进行飞行控制的方法，以求降低无人机的操作成本，进一步推广无人机的应用范围。

本课题使用具有门控卷积网络结构的开源语音识别项目 **MASR** 作为基本的语音识别算法核心，进行用于无人机飞行控制的语音识别程序开发，结合已有开源无人机 **Z410** 作为主要硬件载体，添加机载电脑树莓派，并进行用于用户语音进行飞行控制的飞行控制程序开发，完成了一个基于深度学习的嵌入式语音控制无人机设计。

系统硬件由无人机，飞行控制板 **Pixhawk** 和机载电脑树莓派组成。其中以苍穹四轴开源无人机 **Z410** 作为主要硬件载体，将树莓派 **3B+** 作为机载电脑，和开源硬件飞行控制板 **Pixhawk2.4.8** 协同作为无人机的中控系统。

系统软件部分由语音识别程序和基于 **Dronekit** 的飞行控制程序组成，其作用分别是对用户语音指令进行识别并对无人机进行飞行控制。语音识别程序运行于 **PC** 端上，以门控卷积神经网络为语音识别算法核心，用门控单元取代常见的激活函数以获得更好的性能，拥有包括录制用户语音，语音识别，依据识别结果发送对应的无人机控制命令，并具有一定的模糊匹配和冲突指令检查的功能。飞行控制程序运行于机载电脑树莓派上，使用 **Dronekit** 对无人机进行所需的飞行控制，以达成使用用户语音对无人机进行飞行控制的目标。

语音识别程序和飞行控制程序基于库 **Paramiko** 进行协同，这是一个用于 **SSH** 登录的 **Python** 第三方库。主要思路为树莓派和 **PC** 通过树莓派 **WIFI** 热点进行局域网组网，语音识别程序 **SSH** 登录至树莓派，依据语音识别结果执行特定的 **Dronekit** 程序脚本，例如起飞命令，以达成语音控制无人机的系统目标。

最后通过对系统进行实验与数据分析，论证系统的可用性。主要包括基于容器化技术的语音识别程序测试和基于 **Dronekit** 的飞控连接测试。实验结果表明，系统对于无人机的语音控制已完成原理性验证，用户可以通过自然语音对无人机进行简单控制的系统设计目标已完成。

本系统的更多信息和文档可见 <https://github.com/OscarXChen/Drone-Voice-Control-with-Mandarin-Chinese>。

关键词：自动语音识别 (ASR)；卷积神经网络 (CNN)；无人机 (UAV)；树莓派 (Raspberry pie)；Paramiko；Dronekit

Embedded Voice Control Drone Design Based on Deep Learning

Abstract

As an economical and convenient aircraft, UAVs have played an important role in many fields such as logistics, aerial photography, security, etc. However, the threshold for UAV operation is still very high today, and it is necessary to train qualified UAV operators. A lot of time and money costs. This subject has designed a method of using the user's dictated voice commands to control the drone in order to reduce the operating cost of the drone and further promote the scope of application of the drone.

This system uses the open source speech recognition project MASR with a gated convolutional network structure as the core of the basic speech recognition algorithm to develop speech recognition programs for UAV flight control. The system combines the existing open-source UAV Z410 as the main hardware carrier, adds the onboard computer Raspberry Pi, and develops the flight control program for user voice for flight control, and completes an embedded voice-controlled unmanned aerial vehicle based on deep learning. Machine design.

The system hardware consists of UAV, flight control board Pixhawk and onboard computer Raspberry Pi. Among them, the four-axis open source UAV Z410 is the main hardware carrier, the Raspberry Pi 3B+ is used as the onboard computer, and the open source hardware flight control board Pixhawk 2.4.8 is used as the UAV's central control system.

The system software is composed of a voice recognition program and a flight control program based on Dronekit. Its functions are to recognize the user's voice commands and control the UAV's flight. The speech recognition program runs on the PC, and uses gated convolutional neural network as the core of the speech recognition algorithm. The gated unit replaces the common activation function to obtain better performance. It has the ability to record user voice, voice recognition, and according to the recognition result. Send corresponding UAV control commands, and have certain fuzzy matching and conflict command checking functions. The flight control program runs on the onboard computer Raspberry Pi, and uses Dronekit to perform the required flight control of the drone to achieve the goal of using the user's voice to control the drone.

The speech recognition program and the flight control program are coordinated based on the library Paramiko, which is a Python third-party library for SSH login. The main idea is that the Raspberry Pi and PC conduct LAN networking through the Raspberry Pi WIFI hotspot, the voice recognition program SSH logs in to the Raspberry Pi, and executes specific Dronekit program scripts based on the voice recognition results, such as take-off commands, to achieve the system goal that controlling drones with voice. Finally, through experiments and data

analysis of the system, the usability of the system is demonstrated. It mainly includes voice recognition program test based on containerization technology and flight control connection test based on Dronekit. The experimental results show that the system has completed the principle verification of the voice control of the UAV, and the system design goal that the user can simply control the UAV through natural voice has been completed.

More information and documentation of this system can be found at <https://github.com/OscarXChen/Drone-Voice-Control-with-Mandarin-Chinese>.

Key Words: Automatic Speech Recognition (ASR); Convolutional Neural Network (CNN); Unmanned Aerial Vehicle (UAV); Raspberry pie; Paramiko; Dronekit

目 录

摘 要.....	I
Abstract	II
1 绪论.....	1
1.1 课题的研究目的和意义.....	1
1.2 无人机语音控制的研究现状.....	2
1.2.1 语音识别领域的国内外研究.....	2
1.2.2 结合无人机控制的语音识别研究.....	3
1.3 论文主要内容和工作安排.....	5
1.3.1 主要内容.....	5
1.3.2 章节安排.....	6
2 无人机语音控制理论研究.....	8
2.1 语音识别算法.....	8
2.1.1 语音识别的基本流程.....	8
2.1.2 基于神经网络的语音识别方法.....	9
2.1.3 端到端的语音识别方法.....	12
2.1.4 门控卷积神经网络.....	14
2.2 基于 Dronekit 的无人机控制.....	15
2.3 本章小结.....	16
3 语音控制无人机系统硬件部分.....	17
3.1 无人机硬件系统组成部分.....	17
3.2 无人机中控系统——飞行控制板 Pixhawk.....	17
3.3 无人机中控系统——机载电脑树莓派.....	18
3.4 机载电脑树莓派和飞行控制板 Pixhawk 与 PC 上位机的通讯.....	19
3.4.1 机载电脑树莓派和飞行控制板 Pixhawk 的连接通讯.....	19
3.4.2 PC 上位机和飞行控制板 Pixhawk 的连接通讯.....	20
3.4.3 机载电脑树莓派和 PC 上位机的连接通讯.....	21
3.5 本章小结.....	21
4 语音控制无人机软件部分.....	23
4.1 软件系统组成部分.....	23
4.2 语音识别程序.....	23
4.2.1 基于门控卷积神经网络的语音识别算法.....	23

4.2.2	基于 PC 端的语音识别程序	23
4.2.3	基于 Docker 的语音识别程序移植与语言模型加载	27
4.3	基于 Dronekit 的飞行控制程序设计	29
4.3.1	飞行控制程序基本流程	29
4.3.2	用于飞行控制的 Dronekit 与上位机的连接通讯	31
4.3.3	基于机载电脑树莓派的飞行控制程序	33
4.3.4	用于语音识别和无人机控制的程序设计	33
4.3.5	Ardupilot 开发环境和用于飞行模拟调试的 Dronekit-SITL	33
4.4	本章小结	34
5	系统调试及实验结果	35
5.1	系统环境配置	35
5.1.1	PC 端基本环境配置	35
5.1.2	基于容器化技术的语音识别程序测试	37
5.2	PC 上位机和机载电脑树莓派以及飞控板的连接和通讯	39
5.2.1	基于有线连接的 Dronekit 飞控连接测试	39
5.2.2	基于虚拟化技术的 SSH 远程登陆测试	39
5.2.3	基于 Paramiko 库的 SSH 远程登录测试	41
5.2.4	无人机语音控制系统整体调试	42
5.3	系统应用拓展	43
结 论	44
参 考 文 献	45
修改记录	47
致 谢	48

1 绪论

1.1 课题的研究目的和意义

无人机一般来说指的是通过使用远程无线电信号传输或者是使用搭载于其上的自动飞行控制程序来进行所需的飞行任务的不载人飞行器。对比传统的有人机，无人机最主要的特点就是放弃使用人类作为“飞行控制中枢”，一般来说使用特制的飞行控制算法来进行飞行控制。无人机一般被认为在上世纪 40 年代出现，时至今日，由于无人机及相关技术的快速发展，主要由于自动控制理论的发展和大规模的廉价芯片生产，无人机在军用、民用、商用领域中都发挥着巨大的作用，诸如军事情报侦察，传输线路巡线，物流运输等等。

但是，即使是入门的商品级无人机仍然是一个具有一定危险性的工业载具而不可视为一个简单的玩具，训练一个合格的无人机操作员仍然需要消耗大量时间与金钱，在无人机应用正在加速的当下，无人机操作的复杂性与专业性仍然制约着无人机的广泛使用。

无人机的发展时间虽然较短，但其发展已经呈现出了向自动化与智能化发展跟进的趋势，在过去的十多年中，市场上也零碎出现了一些可以使用语音指令进行控制的无人机，但是由于制作成本以及语音识别准确率等原因，并没有成为主流发展方向。

自然语言交互是人类最基本最自然的交互形式，一直是学术界和工业界在人机交互方面的关注重点。随着过去几十年语音识别技术的发展，具有语音交互性质的产品正逐渐走入我们的生产生活，越来越多的研究人员致力于让机器在尽可能广泛的场景下，尽可能好地理解人类的语音指令。

最近几年，由于计算机科学的发展，越来越多的语音识别技术开始走出实验室并成功随产品落地，例如基于循环神经网络技术的 Deep Speech 开源语音识别引擎的出现带来了大批拥有语音交互能力的产品，为人机交互领域注入新活力。

可以预见，若是可以通过用户的口授语音命令对无人机进行飞行控制，那么无人机的操作门槛可以大幅降低，无人机的应用也会更加广泛，甚至有可能像现今的智能手机一样，迈向“人手一机”的时代。但是目前的语音控制无人机往往具有技术复杂，可靠性不高，控制延迟时间久等的问题，一旦考虑到无人机本身的危险属性，这些不确定因素都使得无人机的语音控制系统的缺点是不可接受的。本课题使用的主要技术，包括基于卷积门控神经网络的语音识别算法，基于 Dronekit 的飞行控制代码，树莓派机载电脑，Pixhawk 开源硬件架构虽然各组件都较为复杂，但是它们都是相对成熟的技术，难点在于将系统进行有机组合调试，且调试难度在可接受范围内，也就是说在高耦合的模块内

技术难度相对高，在低耦合的模块间技术难度相对低，以这种模块化设计思想指导，进行基于深度学习的嵌入式语音控制无人机设计。

1.2 无人机语音控制的研究现状

1.2.1 语音识别领域的国内外研究

(1) 国外语音识别研究现状

国外语音识别研究发展较早，至今有约 70 年历史，语音识别技术大体上以 1980 年为分割，经历了由隐马尔科夫模型（HMM）到人工神经网络（ANN）的变化。

1952 年，早在 70 多年前的贝尔实验室是目前公认的最早语音识别系统，该系统拥有对 10 个英文字母进行发音识别的功能^[1]。

1960 年后，几大经典语音识别算法渐渐成熟，语音识别领域蓬勃发展。有代表性的主要有线性预测编码（Linear predictive coding，一般简称为 LPC）和名为动态时间规整 DTW 的技术（全称 Dynamic Time Warping）技术。此外，还有两个重要的理论也随之发展，主要有称为 VQ（Vector Quantization），又名矢量量化，和业界广泛使用的隐马尔科夫模型（HMM）理论^[2]。

其中，基于高斯混合模型与隐马尔可夫模型协作的方案（一般称为 GMM-HMM）来对声学模型进行建模分析与通过使用 N-Gram 来对所需的语言模型进行建模分析的思路逐渐成为了语音识别框架的主流技术方案^[3]，这种技术路线一般被认为是基于统计分析实行的。

1980 年后，语音识别领域的研究重点逐渐转向了连续化识别、非特定人使用、多词汇识别的方向。与此同时，在技术路线上逐渐由原先的匹配模式变为统计模型^[1]。同一时期，虽然将 HMM 应用于人类语音识别的假设并没有得以证明，但由于其出色的实际应用表现，HMM 渐渐崭露头角，成为最早的应用广泛的大词汇量连续语音识别（Large Vocabulary Continuous Speech Recognition, LVCSR）的主流声学建模方法，同时受益于计算机科学领域的进步，以 ANN 为代表的语音建模方法开始蓬勃发展^[4]。

1990 年后，技术层面上来说，ANN 的各类分支，例如相对一般的前馈神经网络（FNN），拥有时间维度的循环神经网络（RNN），可以更好处理图像化以及各种可以转化为图像化数据的卷积神经网络（CNN）都开始被应用于语音识别技术^[5]；商业层面上来说基于语音识别技术的商业产品已经开始规模化生产使用^[1]。

2011 年，俞栋、邓力等人提出了基于上下文相关的深度神经网络以及隐马尔可夫模型。他们的工作相对于过去的 GMM-HMM 系统在连续的语音识别任务中获得了比较好的成果。随后，语音识别相关研究逐渐转向了基于循环神经网络（Recurrent Neural

Networks, RNN) 的建模方法和使用卷积神经网络 (Convolutional Neural Networks, CNN) 的结构^[3]。

如今, 主流的商用语音识别技术主要分为由传统数字信号处理技术延续而来的相对低成本的 HMM 和由新兴的深度学习领域分支出来, 泛用性更好的深度神经网络 (DNN) 两个研究方向, 但几乎所有先进的语音识别引擎都使用 DNN 作为算法核心^[5]。

(2) 国内语音识别研究现状

与之相对, 国内在语音识别领域起步相对较晚, 发展速度相对较慢。目前一般认为最早的研究可追溯至 1958 年。在这一年中中国科学院声学所制造除了一个可以对 10 各元音进行语音识别的系统, 其仅仅是使用电子管电路完成的。在 1973 年后, 其正式开始了使用计算机技术进行语音识别的研究路线^[1]。

1986 年, 语音识别领域成功被列为“863”计划的研究课题, 从这以后, 中国国内的语音识别技术逐渐迈上了一个新的阶段, 开始加速发展^[1]。

近年来, 随着国内人工智能领域的爆发式发展, 以科大讯飞和百度语音为代表的几家企业在技术上不断深耕, 例如科大讯飞在语音框架中对于 DFCNN 技术的应用和百度在声学模型构建中对于 Deep CNN 技术的应用^[2]。如今我国在语音识别领域可以说处于世界领先水平。

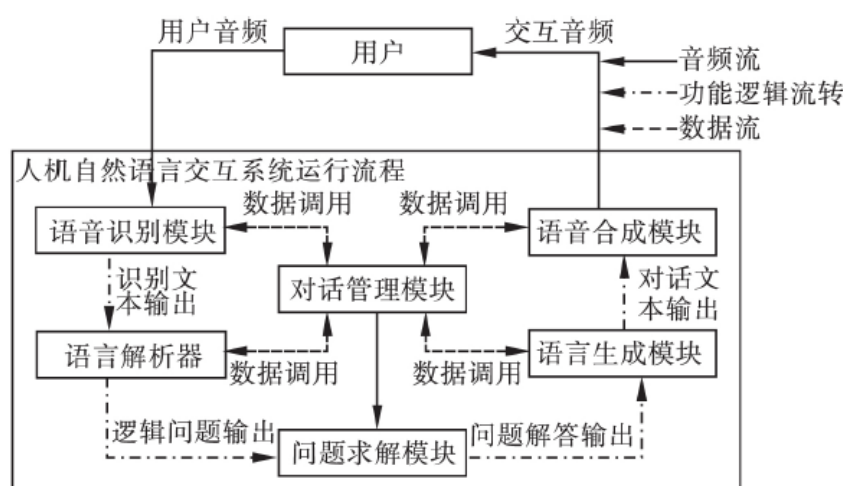


图 1.1 人机自然语言交互系统流程

1.2.2 结合无人机控制的语音识别研究

从上个世纪的七十年代到现在, 无人机技术由于其具有廉价性、广泛性等的特点而备受青睐, 在世界的各个角落的军用和民用抑或是商用领域都承担着重要的作用^[6]。

在语音控制无人机研究的最初阶段，主要考虑到机载的处理器性能上的不足，大多数研究人员都会选择将要处理的数据传送到为了数据处理而创立的地面基站来进行运算数据处理，之后再通过各种通讯手段例如数传电台来传回给无人机以对无人机的运动进行相应控制。一般认为在这个过程中没有人类进行直接控制参与，所以可以认为这种也属于广泛意义上的智能控制，但一旦考虑到地面站的性能和无人机机载处理器的性能限制，还有远程通信带来的不可靠性，这种控制方式的鲁棒性并不是很好，同时自主性也有待提升^[7]。

在这样的背景下，我们可以发现，当下阶段有关于无人机的智能控制方面的研究工作还有许多需要认真解决的问题，而许多的研究结果近适合于实验室场景，并不适合于实际需要的生产生活场景，这就对模型的鲁棒性提出了较大的考验。

另外，由于室内搜索救助、室外跟踪等目的不同，对应智能 UAV 的设计架构和传感器的使用也不完全相同^[7]。

2001 年，美国斯坦福大学研制出 WITAS(Wallenberg laboratory for research on Information Technology and Autonomous Systems)多模态对话系统用于控制无人机设备，可以识别操作人员的语音命令并对智能机器人进行控制，这是最早的一批使用语音控制的无人机系统设计。

之后，美国空军实验室先后在 2003 年，2008 年论证了语音控制相比于传统的手动控制而言可以让任务的错误执行率大幅降低并开始逐步在有人机的控制系统中加入语音控制模块用以辅助飞行员操控飞机。美国 Voice Flight Systems 公司在 2009 年进一步推进了语音控制飞机的发展，其语音识别产品 VFS101 获得 FAA 认可，该系统具有非特定人语音识别功能，并且不需要对飞行员进行预先语音训练。到了 2011 年，英国 BA E 系统公司为“台风”战斗机研制出了一种新型的具有语音识别辅助系统的头盔，这是首批语音控制系统在现役装备上的运用。

对于无人机的语音控制方面，在 2004 年美国 MIT 的研究人员实现了一种有人机对无人机制导系统，它可以让有人机的飞行员通过语音指令控制其它无人机。在之后的几年间，一系列研究表明，使用语音控制无人机可以带来更好的可靠性，比如美国空军研究实验室 Williamson D T, Draper M H 和 Calhoun GL 对比了在无人机地面控制站模拟器中采用语音控制和传统手动输入控制的效果，结果表明采用语音控制在错误率比传统手工输入控制降低 33% 的基础上，还可以节省大约 40% 的时间^[8]。Maza I, Caballero F 和 Molina R 等通过大量实验，发展了在地面站系统可以使用的简单可行的语音控制系统^[8]。

现在,越来越多的学者和研究团队开始尝试在工程意义上对无人机和语音控制系统进行结合,比较有代表的有如南京航空航天大学的王冲在嵌入式平台上通过提取 MFCC 参数使用 DTW 方法进行无人机语音指令识别,识别率达到 93%^[8]。上海理工大学的应捷和韩旭将安卓手机作为语音识别处理硬件,设计了一套无人机语音识别系统,识别率达到 84%^[8]。在 2019 年,已有提出已有研究者设计应用于无人机的手势和语音控制平台,可以同时进行手势和语音识别,架构图见图 1.2^[9],此处使用的地面站处理平台是树莓派。

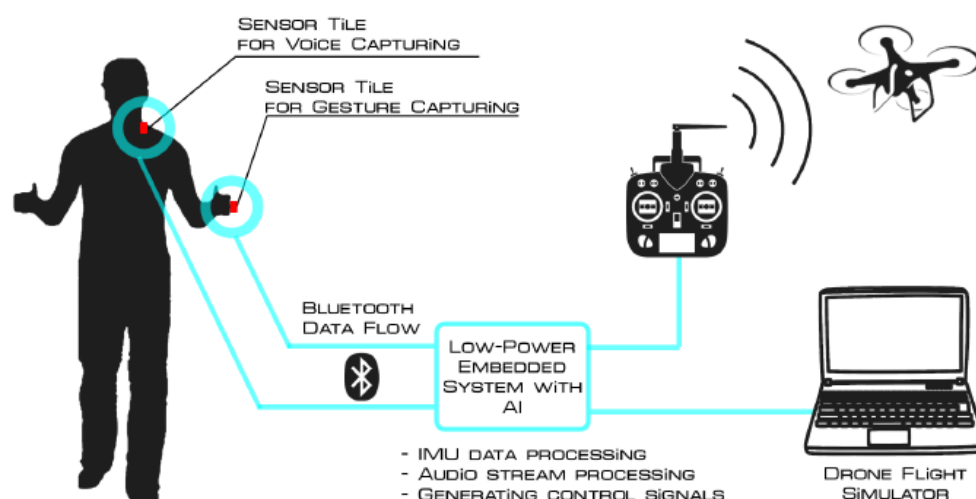


图 1.2 一已实现的语音识别和手势识别无人机系统架构图

1.3 论文主要内容和工作安排

1.3.1 主要内容

课题主要目的为实现使用语音对无人机进行基本控制,包括用于语音识别和无人机控制的软件系统设计和用于验证系统可用性的硬件系统设计。

课题设计系统主要由 PC 上位机,机载电脑树莓派,飞控板 Pixhawk 组成,使用运行于 PC 上位机的语音识别程序对用户口授语音进行识别,提取并分析用户命令并将其发送至机载电脑树莓派,以操作飞控板 Pixhawk 使得无人机进行相应飞行动作,完成使用语音控制无人机的系统设计目标,即论文标题,基于深度学习的嵌入式语音控制无人机设计。

课题的理论研究内容包括语音识别算法研究和无人机控制方法两大类，其中以语音识别算法为主要论述内容，包括语音识别的基本方法和课题设计系统使用的语音识别方法。

系统的软件部分是核心研究内容，在语音识别方面，基于具有卷积门控神经网络结构的开源语音识别项目 **MASR** 开发用以识别用户语音指令的语音识别程序，并添加修改了语音输入部分，添加了模糊匹配和命令词提取功能，使得其算法适配于无人机语音控制的使用场景。在无人机控制方面，系统使用树莓派 **WIFI** 热点进行局域网组网，在语音识别程序中调用 **Paramiko** 库以 **SSH** 登录至树莓派，运行基于 **Dronekit** 的无人机控制程序脚本，脚本包含如起飞等简单命令，以此实现语音控制无人机的系统目标。

系统的硬件部分主要包括由苍穹四轴开源无人机 **Z410** 和由机载电脑树莓派和飞控板 **Pixhawk** 组成的无人机中控系统。飞控板 **Pixhawk** 主要用于维持无人机姿态自稳等基础性飞行控制功能，机载电脑树莓派主要用于获取 **PC** 上位机发送的识别后的语音控制指令，再将具体的命令通过串口传给飞控板 **Pixhawk** 以完成语音控制无人机的系统目标。

最后通过对系统模块进行实验并分析获得的结果，论证系统设计的可靠性并对系统模块进行原理性验证。其中包括基本环境配置和基于容器化技术的语音识别程序测试和系统的联合调试。

1.3.2 章节安排

本论文的章节安排如下：

第一部分：绪论。主要讲述了语音控制无人机的发展历程和课题背景，以及国内外关于语音控制飞行器的实验研究以及发展历程，还有本论文主要内容及工作安排。

第二部分：无人机语音控制理论研究。介绍了经典的语音识别方法和端到端的语音识别方法，并对本文使用的门控卷积神经网络进行了深入介绍，同时探讨了基于 **Dronekit** 的无人机控制

第三部分：语音控制无人机系统硬件设计。主要介绍了系统验证无人机 **Z410** 和由飞行控制板 **Pixhawk** 和机载电脑树莓派组成的无人机中控系统。

第四部分：语音控制无人机系统软件设计。主要讲述了软件系统的组成部分，包括语音识别程序和基于 **Dronekit** 的飞行控制程序，分别作为用语音来进行无人机系统控制的“语音”部分和“控制”部分。其中详细介绍了语音识别程序的工作流程和环境搭建，飞行控制库 **Dronekit** 及用于飞行控制的部分核心代码原理，还介绍了用于自动 **SSH** 登录的 **Paramiko** 库，展示了软件系统的设计与分析。

第五部分：系统调试及实验结果。主要进行了基本环境配置，系统各模块连接通讯调试，系统联合调试试验，并对上述三个试验结果进行结果分析，得出项目可靠性与实用性程度分析。

2 无人机语音控制理论研究

2.1 语音识别算法

2.1.1 语音识别的基本流程

(1) 语音输入

语音输入过程即使用可以理解为高级麦克风的拾音器电路，将用户口头输入的声音信号录制转换为电信号，再经过信号放大，信号降噪和信号 AD 转换等电路进行相对基础的基于硬件的数字信号预处理。

(2) 语音预处理

语音预处理的主要过程有预加重和更为重要的加窗分帧操作。前者的主要任务是尽可能减小由于口唇辐射导致的对于采集音频的不良影响^[10]，而在音频信号在空气介质和电路的传递过程中，高频部分往往由于干扰或者带宽等问题丢失概率相对较高，故而一般需要通过这种预加重的方式对高频部分进行加权处理，使得信号频谱变得平缓^[11]，提高音频的分辨率（在高频段部分）。分帧的目的是将待处理的额录制语音切割为长度相当的语音片段。根据信号处理理论，所有的信号都可以被简单分为平稳的和非平稳的这两个类别，而音频信号属于典型的非平稳信号，但是大多数的包括语音处理在内的信号处理方法一般都只能处理平稳信号，故需要使用切片划分的方法，使得其达到平稳信号的假设，也即将其分成许多的时长相对短小的信号，使得其的稳定性更好^[12]。一般来说，认为将其划分为 15-30ms 的小片段即可认为其对于短时平稳假设符合程度较好，如此便有助于进行后续的信号处理工作开展。

此外，加窗还可以使得抽样端点边缘的语音波形得以加强，核心思想是使用连续平滑的数学函数拟合阶跃且不连续的物理波形。

(3) 语音特征提取

传统的深度学习的语音识别的第一个步骤一般来说是使用提取语音特征值的方法来进行^[13]，而 MFCC（一般称为梅尔倒谱系数）正是一种使用比较广泛的语音识别中经常会利用到的特征值，其是基于人耳的听觉非线性指数特性来进行设计的。目前已有的研究认为，当语音信号的频段在临界值一千赫兹以下时，人耳与语音信号的响应可以近似认为是一个线性性质较为强的关系，但是较为棘手的是，一旦当语音信号的频段在临界值一千赫兹以上，两者的线性关联程度就会由之前的较强变为较弱。进一步的研究表明，一般我们可以认为它们呈类似对数关系，这是一种典型的非线性特征。

基于这个基本事实，我们设计了 MFCC 特征，一般操作方法是先将原本是线性分布排列的频谱经过一个非线性映射，将其转换到基于听觉特性而设计的 Mel 非线性频谱上，之后再转换为倒谱。由于这个过程有称之为倒映射的过程，所以 MFCC 也被称为梅尔倒谱

MFCC 相比于过去曾经采用过的 LPCC 在鲁棒性上有更好的优势，且其的设计是符合人耳的非线性指数听觉特性的，同时它在低信噪比的恶劣环境下也能进行效果相对较好的识别，故其使用较多。在提取 MFCC 特征参数后，以此构建声学模型，并可以为之后结合语音模型作好准备^{[12][10]}。

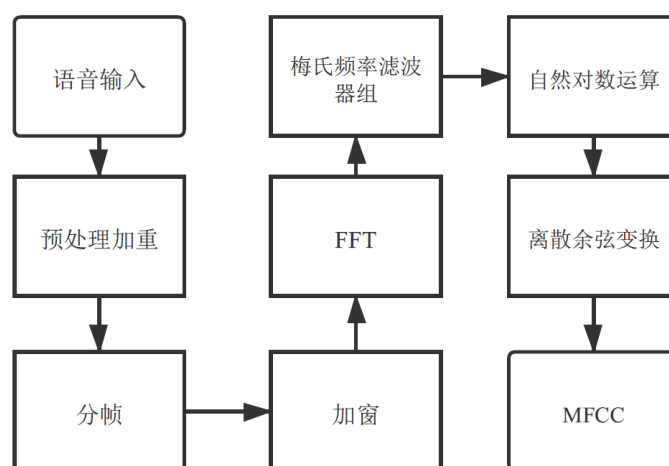


图 1.3 MFCC 参数提取流程

2.1.2 基于神经网络的语音识别方法

（1）神经网络概述

神经网络的本质是一个函数，对于给定的输入值，经过复杂的内部函数结构，输出其对应值，其成功之处主要在于拟合性好并且泛用性强，以下为神经网络的基本原理。

神经元(neuron)是神经网络的基本计算单元，也被称作节点(node)或者单元(unit)。它可以接受来自其他神经元的输入或者是外部的数据，然后计算一个输出。每个输入值都有一个权重(weight),权重的大小取决于这个输入相比于其他输入值的重要性。然后在神经元上执行一个特定的加权和函数,这个函数会将该神经元的所有输入进行加权和操作。

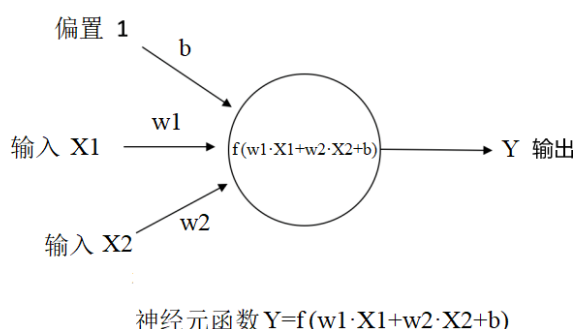


图 2.1 基本的神经元

我们可以通过上图发现，除了一般认为的权重外，还有一个偏置位输入，其大小为 1，此处的函数 **function** 可以被认为是一个用以引入非线性成分的激活函数，其主要作用在于给神经元的结构引入非线性扰动。显然在现实的物理世界中，所有的数据结构都不是线性分布的，故引入非线性扰动可以使得其对于客观世界有更好的拟合性质。

前向神经网络一般被认为是一种结构相对来说较为基础的人工神经网络，它的基本特征结构可以参照下面列出的这张神经网络结构图所示。

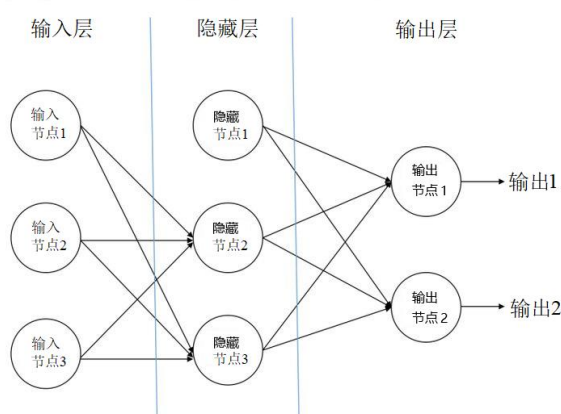


图 2.2 前向神经网络结构图

如上图所示，这个神经网络分为 3 个网络层，分别是输入层，隐藏层和输出层，每个网络层都包含有多个神经元，神经元可以分为以下三种情况。

输入神经元位于输入层，主要是传递来自外界的信息进入神经网络中，比如图片信息，文本信息等，这些神经元不需要执行任何计算，只是作为传递信息，或者说是数据进入隐藏层。

隐藏神经元位于隐藏层，隐藏层的神经元不与外界有直接连接，它都是通过前面的输入层和后面的输出层与外界有间接的联系，因此称之为隐藏层，上图只是有 1 个网络层，但实际上隐藏层的数量是可以有很多的，远多于 1 个，当然如果没有隐藏层也是可接受的，若是如此，则网络就会变成只有输入层和输出层的情形了。隐藏层的神经元会执行计算，将输入层的输入信息通过计算进行转换，然后输出到输出层。

输出神经元一般来说是位于输出层，其功能是为了使得信息（来自于隐藏层）输出至模型外，也即将最后的结果进行输出。

根据已有的研究成果，只要大于等于三层的神经网络就可以实现以任意精度拟合任意函数的效果，之后神经网络开始受到学术界和工业界的重视。本文中使用的基于卷积门控神经网络的语音识别算法正是基于基础的前向神经网络的演变而来的。

（2）基于神经网络的语音识别方法

神经网络在语音识别领域应用广泛，一般认为从 2010 年后，在工业界就主要使用基于神经网络及其各种变种方法进行语音识别，传统的语音识别方法正在被工业界抛弃，或是成为高校科研领域的“原理性验证”产物，已经脱离了生产实际。究其原因，应是具有过程复杂，使用难度大，而现在的硬件技术进步已经可以很大程度上弥补使用神经网络算法带来的计算负担。

语音识别的基本方法中，考虑到发挥重要作用的语言模型，我们仍然需要对此进行深入研究。一般认为对于语言是中文的语音识别领域，使用声学模型使得录制的音频数据变换为带有西文拼音的样式，再使用加载语言模型的手段将其转换为最终用户需要的汉字字符串结构。

在上个世纪初，一般使用基于统计的语言模型建模，主要方法是使用浅层的前馈神经网络模型。由于其作为一种连续的空间语言模型，其平滑词概率分布函数可以相对较好地解决词频较低的词对模型的干扰问题，故在语音识别领域取得了较好的成效^[12]。

从理论上说，通过神经网络进行密度函数实现要比传统的隐马尔可夫模型中广泛采用的高斯混合密度函数表现得更为一般一些。由此神经网络的性能在理论上终是将会超过隐马尔可夫模型的，但是考虑到神经网络模型训练的开销，隐马尔可夫模型也还在许多地方使用。但是一般认为，神经网络系统比隐马尔可夫模型系统更紧凑、更规则，有利于更高效的实现^[14]。此外，与其他的语音识别用的分类器相比，深度神经网络 DNN 的主要优势在于可以对于语音帧与帧之间的联系加以加强利用^[15]，因为显然语音是一个上下文相关度较高，或者说时间维度关联性强的数据。

循环神经网络取得这种优势的主要原因在于，其记忆能力可以涵盖完整的语音序列，而前馈神经网络只能利用一个有限长度内的信息。然而，正因为循环神经网络的记忆能

力可以涵盖整个语音序列，因此其层间和层内需要更为复杂的连接，这也带来了巨大的计算量，直接限制了循环神经网络在实时性上的性能^[8]。

虽然 RNN 有两个很明显的问题，先是效率问题，它需要逐词进行处理，后一个词要等到前一个词的隐状态输出以后才能开始处理；再是如果传递距离过长还会有梯度消失、梯度爆炸和遗忘问题，但是这些问题在本项目中应该可以接受。

像语音控制这样的体机界面对于操作者而言更加直观，对比使用控制器的情形，操作者一般仅需更少的训练就可达到同样的无人机的操作水平。因此，已经有许多工作在开发不同类型的体机界面无人机操作方式^{[16][17]}，与语音识别 RNN 框架结构，例如图 4.1.3 中所示^[9]。较为广泛验证的语音识别错误率大约在 10% 上下^[18]。

本项目预计在 PC 上位机上搭载已量化的端到端语音识别模型，结合树莓派 3B+ 系统，尽可能在实时性和成本可接受的情景下，完成离线语音识别，并将结果转换为无人机控制命令后发送至无人机飞行控制板，以达成语音控制无人机的目的。

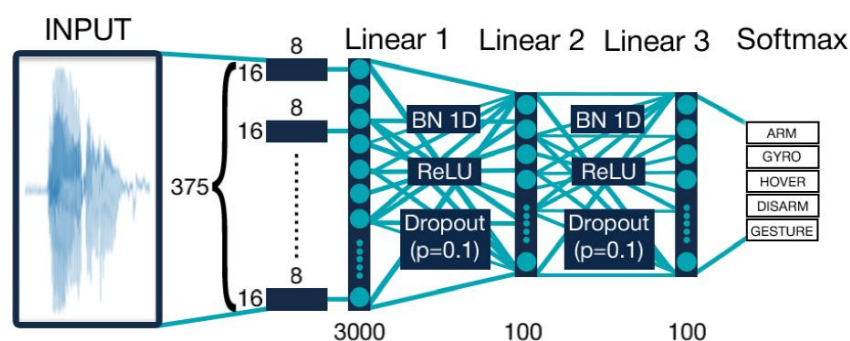


图 2.3 用于语音识别的神经网络架构图

2.1.3 端到端的语音识别方法

(1) 端到端的语音识别方法

使用端到端（End to End）思想进行语音识别是最近几年发展起来的一种相对新兴的语音识别架构，其主要优点在于使用者不需过度在意中间的运算过程，仅需在输入的输入，在输出端输出，即意为“端到端”。这种语音识别框架一般来说拥有以下两种方法实现，基于 CTC 的训练或者和使用基于 sequence to sequence 抑或是采用注意力（attention）机制。下图是 Google 在 2018 年 interspeech 上对比传统语音识别系统的示意图。

Typical Speech System

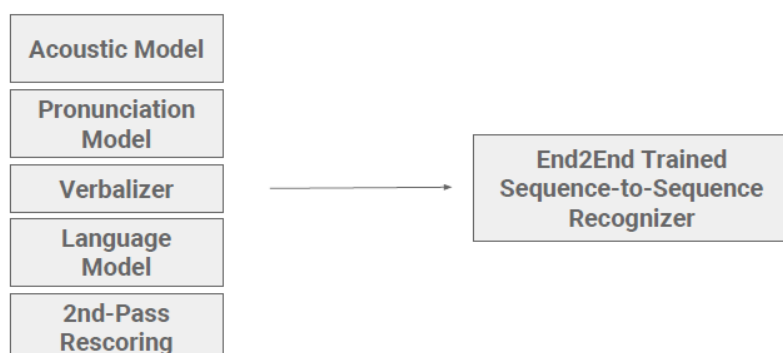


图 2.4 典型的语音识别系统和端到端框架的语音识别系统对比

过去常用的语音识别系统一般来说由许多个相对独立的模块分立组成，一般包括声学模型和发音词典以及语言模型等。在这些模块中，声学 and 语言模型都是需要进行一些训练的，而以往这些训练一般都是独自进行，也就是说他们各自有他们各自的目标函数，在实际使用中，训练的结果很可能并没有追求语音识别这一“全局最优”目标，而是由于各种因素掉到了目标函数的设置这一“局部最优”的陷阱中去。一个常见的例子是声学模型的目标是将语音训练的概率最大化，但是语言模型的训练目标却是将困惑度最小化，在实际的使用场景中，很可能使得分模块的训练方法得出的结论并不是理论上的最优性能。一般来说，针对这个问题有使用端到端训练（end-to-end training）和使用端到端模型（end-to-end models）这两种解决办法。

前者一般来说指意思是将声学模型和语言模型结合到一起，一起以系统的全局最优目标进行声学模型训练，由于这种方法在尝试解决分立训练的问题，这种方法可以视作一种相对初级的“端到端”的训练模式。但是从原理上显而易见，这种方法并没有真正解决分立训练的问题，因为语言模型仍然是独自训练的，此时就需要使用后者的方法。

在后者的解决方案中，系统不再区分具体的语言模型、声学模型等等，而是直接将其黑箱化，使用一个相对复杂的神经网络将输入和输出端联系起来，这种训练方法往往需要较大时长的训练强度，但是像这样使用一个神经网络来发挥原有各个模块的功能的方法，才是真正的“端到端”，可以解决前面“局部最优”和“全局最优”的冲突问题。

新兴的端到端的语音识别框架拥有降低过去方法一些尚未证明的假设的优点，但正像大多数神经网络模型一样，也具有难以训练的缺点。在实践中，如果目标是获得相对好的结果，那么往往需求许多的训练语料，而这些训练语料可能在研究领域并不现实，而在工业界却要有明确的预期才可能投入如此多的资源进行训练。一般来说，端到端这样的语音识别模型输入使用梅尔语谱，拥有可以忽略中间过程直接产生对应的自然语言

的文本的能力，子词、词、字母等都可以是端到端的语音识别的输出。在当前业界使用子词作为输出是相对较为常用的做法，一般用 `sentence piece` 等工具将文本进行切分（和自然语言处理类似）。

2.1.4 门控卷积神经网络

（1）卷积神经网络

本课题的语音识别模块核心算法是门控卷积神经网络（`Gated Convolutional Network`）。

卷积神经网络一般缩写为 `CNN`，是一种具有卷积层（`Convolutional Layer`）结构的神经网络，一般使用具有二维空间维度的卷积层，用以处理图像和各种可以方便转化成类似结构的数据，在课题中，即语音数据。相对于过去的算法和其他结构的神经网络，卷积层可以相对更加有力地对图片地局部信息进行高效特征提取，继而方便进一步进行数据处理。

`CNN` 的工作流程主要如下所述：给模型输入数据（图片，或者数学上等效转化为图片的数据，如语音数据），经过卷积层，非线性化（激活函数）处理，池化层，后接全连层，最后得到输出结果。

`CNN` 在本质上是一种输入到输出的映射，它能够学习大量的输入与输出之间的映射关系，而不需要任何输入和输出之间的精确的数学表达式。其优点在于权重共享策略减少了需要训练的参数，相同的权重可以让滤波器不受信号位置的影响来检测信号的特性，使得训练出来的模型的泛化能力更强；池化运算可以降低网络的空间分辨率，从而消除信号的微小偏移和扭曲，从而对输入数据的平移不变性要求不高。

众所周知，现代硬件非常适合高度可并行化的模型，同时考虑到卷积神经网络（`CNN`）易于捕获上下文语境的特点，我们可以相对容易地将序列内容并行化计算，并且与循环网络相比，`CNN` 的速度显著提升。为了利用 `CNN`，已有研究人员提出了用于语言建模的门控线性单元（`GLU`），它在卷积输出上实现了简化的门控机制。`GLU` 被引入序列来学习序列，它优于基于门控递归单位（`GRU`）和 `LSTM` 的机器翻译模型。受 `GLU` 成功和 `LSTM` 门控机制的启发，改进的门控循环架构被提出并应用于匹配任务的“比较 - 聚合”框架。门控卷积网络配备了输出和遗忘门。输出门调制当前卷积层的输出。同时，由先前卷积层创建的上下文信息由遗忘门调制并存储在存储器单元中。门控机制进一步优化了信息流动的路径，并为匹配任务带来了更好的性能。

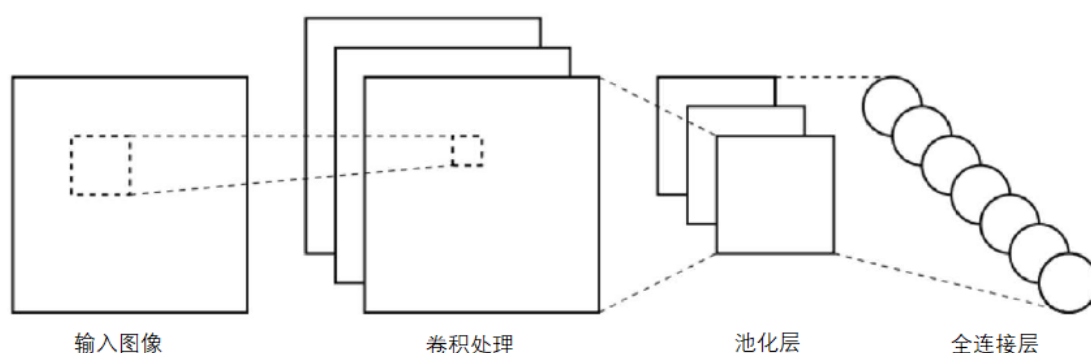


图 2.5 常见的卷积神经网络结构示意图

(2) 门控单元与门控卷积网络在语音识别的应用

相比于卷积网络，门控卷积神经网络引入了一种新的门控机制，用门控时间卷积代替了递归网络中常用的递归连接，该机制可以减轻梯度传播的问题。网络结构类似于 Facebook 在 2016 年提出的 Wav2letter。但是使用的激活函数不是 ReLU 或者是 HardTanh，而是 GLU（门控线性单元）。因此称作门控卷积网络。根据以往的实验，使用 GLU 的收敛速度比 HardTanh 要快。相比于常见的激活函数，课题使用门控单元作为激活函数，以获得更好的性能。

2.2 基于 Dronekit 的无人机控制

(1) Dronekit 概述

DroneKit-Python 是一个用于控制无人机的 Python 库。Dronekit 提供了用于控制无人机的 API，其代码独立于飞控，单独运行在机载电脑（Companion Computer）或其他设备之上，通过串口或无线的方式经 Mavlink 协议与飞控板通信。除了 Dronekit-Python 以外，还有 DroneKit-Android 以及 DroneKit-Cloud 的 API 供不同的开发者使用。API 通过 Mavlink 与飞控通信。它提供对连接飞控的遥测，状态和参数信息的编程访问，并实现任务管理和对飞行器运动和操作的直接控制。

Dronekit 的实质是通过发送和接受 Mavlink 消息，向飞控发送控制指令、从飞控获取各种状态信息。Dronekit 的所有内置功能都是基于 Mavlink 实现的。

(2) Dronekit 基本 API

用于使用 Python 语言连接无人机飞控的 Dronekit 开发库中，API 的设置和使用和常见的开发库有所区分，以下以常见的用于飞控连接和测试的 connect 函数进行简要的风格介绍，这些也是本课题开发的飞行控制程序中所多次使用的。

用于飞控连接的 connect 函数，其原型为 `def connect(ip, initialize = True, wait_ready = None, status_printer=errprinter, vehicle_class = None, rate = 4, baud = 115200, heartbeat_timeout = 30, source_system = 255, use_native = False)`。其具体含义如下所述，ip 意为目标无人机的地址；wait_ready 设置为 None 意为函数返回之前，是否等待属性参数下载完成（通常设置为 True）；status_printer 设置为 errprinter 是为了输出诊断信息的途径，默认输出到运行程序的 cmd；vehicle_class 是对该函数的返回值的说明，其中提到的“vehicle”类是 Dronekit 中定义的一个类，将其设置为 None 即规定“Vehicle”类本身或它的任意子类都可以是合法的返回值类型；将 rate 设置为 4 即是对数据流的刷新频率进行人为规定，一般来说选择 4Hz 即可应对大多数情况，并且这一设定往往也为众多开发者所默认；接下来的三个参数设置，即为设置串口连接时候的默认的波特率为 115200，设置默认的无人机连接超时时间为 30 秒，设置默认的目标无人机 Mavlink ID 是 255，最后将 use_native 设置为 False 即意为使用预编译的解析器。

包括 connect 函数常见的控制 API 主要为连接飞控用的 `connect('com3', wait_ready=True, baud=115200)`；设置飞行模式用的 `vehicle.mode = VehicleMode("GUIDED")`，其中在使用 Dronekit 来进行飞行控制时必须设为 Guided 模式；在初始点开始起飞并飞行至指定高度的 `arm_and_takeoff(aTargetAltitude)`；以速度模式控制飞机运动，也是常见的通程序调用 API 控制飞行速度的函数 `set_body_velocity(velocity_x, velocity_y, velocity_z, duration)`；用以简单偏转无人机机头朝向的函数 `condition_yaw(heading, relative=False)`，其中方向控制变量 heading 为正时向左偏转，为负时向右偏转，相对偏转控制标志变量 relative 值为 True 时，意味着是相对于当前机头朝向进行偏转操作。

2.3 本章小结

本章主要对于语音识别算法和无人机控制进行了深入研究。在语音识别算法部分，介绍了语音识别的经典方法和端到端的语音识别方法，进而深入介绍了本课题使用的门控卷积神经网络，提出了门控卷积神经网络对比常见的网络的主要区别和主要优点，是本文语音识别用户指令的基础。在无人机控制方法部分，介绍了本文使用的基于 Dronekit 的无人机控制方法，是后文具体实现无人机语音控制功能的基础。

3 语音控制无人机系统硬件部分

3.1 无人机硬件系统组成部分

无人机硬件系统包括 PC 上位机，实验用无人机 Z410，飞行控制板 Pixhawk 和机载电脑树莓派四大部分。

PC 上位机用于收集用户口授指令并将其转化为便于飞行控制的文本字符串，以 WIFI 热点和机载电脑树莓派进行局域网热点组网通讯。

实验用无人机为苍穹四轴的 Z410 无人机。其主控为 Pixhawk2.4.8，含 SE100 型号 GPS 和 500mw 数传。螺旋桨尺寸为 10 寸，轴距尺寸为 410mm。机载电脑为树莓派 3B+，遥控器为富斯 I6。充电器为镍氢电池充电器，动力电池型号为 5200mAh/3S，平衡充电器型号为 A400。机架的上层板采用 3mm 厚的碳纤维版，加 3mm 厚的电路基材，上下连接结构，机身下层布置安装机载电脑和动力锂电池，动力方面，采用 T-motor 电机与好盈电调的动力组合，整机最大起飞重量为 5kg。

无人机中控系统包括飞行控制板 Pixhawk 和机载电脑树莓派。其中 Pixhawk 是已集成好的用于飞行控制的开源硬件板，机载电脑树莓派是课题进行飞行控制的关键，其使用数据连接线与飞行控制板 Pixhawk 连接，接受来自 PC 上位机的指令，作为语音控制无人机系统的中介桥梁。

3.2 无人机中控系统——飞行控制板 Pixhawk

Pixhawk 飞控板是采用了 ST 公司生产的 STM32 系列芯片为主要处理器，由 3DR 公司研发，主要功用是运算和分析机载各传感器（例如惯性测量单元陀螺仪，加速度计，磁力计等）数据，进行飞行控制的开源硬件飞控板。



图 3.1 Pixhawk 实物图

它具有丰富的外设接口，可以方便地与外围设备进行相关通讯，其接口 Telem1 可用于连接接收机 Spektrum/DSM 接收机，接口 Telem2 可以用来对数传模块进行连接，接口 Telem3 可以对图传模块进行连接。

无人机的核心是飞控板，除此之外还需要大量飞行和支撑用的机械结构材料，例如机架，电子调速器，马达，电池，螺旋桨等在此处略去不表。

3.3 无人机中控系统——机载电脑树莓派

树莓派（Raspberry Pi），常简称为 RPi，是一个设计用途为学生学习 Linux 系统或进行基本的硬件开发而生的一个只有校园卡大小的“麻雀虽小五脏俱全”的小型电脑，系统基于 Linux，可以进行多种功用，一般面向学生和开发者群体，拥有廉价、功耗低等优点。

树莓派大致分为拿来当 HomeServer 用的服务器派，和用 GPIO 控制外设的嵌入式派。嵌入式派类似于 STM32，编程语言为 Python。服务器派则是基于 Linux 系统。本课程使用的是树莓派 3B+。

树莓派组件包括主板一块、散热片四张、外壳一个、16G microSD 卡一张、microSD 卡的读卡器一个、电源线及插头各一、HDMI 线一根、飞鼠（无线键盘鼠标二合一）一块，用于远程控制树莓派。以下两张图是示意用的树莓派 4B 实物，并不是搭载于无人机上作为机载电脑的树莓派 3B+。



图 3.2 树莓派 4B 主板



图 3.3 已安装散热片并烧录系统的树莓派 4B

下图为已安装在无人机货架上的树莓派 3B+, 其作用是充当机载电脑, 负责与 PC 上位机和飞控板 Pixhawk 通讯, 是 Dronekit 代码的运行环境。



图 3.4 安装在无人机货架上的树莓派 3B+

3.4 机载电脑树莓派和飞行控制板 Pixhawk 与 PC 上位机的通讯

3.4.1 机载电脑树莓派和飞行控制板 Pixhawk 的连接通讯

硬件上,使用 TTL 转 USB 模块进行机载电脑树莓派和飞行控制板 Pixhawk 的连接, 连接线接到 PIX 飞控的 telem2 口, 这是树莓派和 Pixhawk 的数据连接。

此外,树莓派需要稳定的 5V 供电电源才能稳定运行, 故将树莓派电源线连接至底板的 JST 口以提供稳定的供电电源。

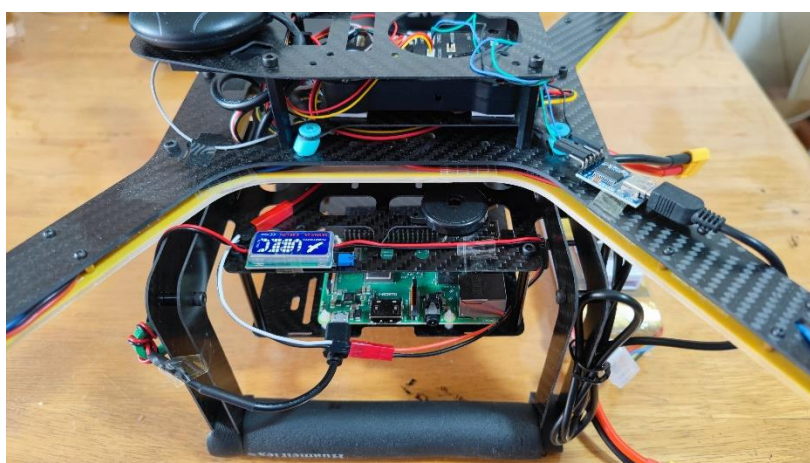


图 3.5 完整安装机载电脑树莓派的实验用无人机

软件上，机载电脑树莓派和飞行控制板 Pixhawk 主要使用通讯协议 Mavlink 继续通讯。

Mavlink (Micro Air Vehicle Link)，即微型飞行器连接通信协议，首次由 Lorenz Meier 在 2009 年初发布。Mavlink 协议是一种轻量级的更高层的开源通讯协议，其基础为基本的串口通讯。该协议主要用于与无人机和无人机载相关组件之间进行通信活动，拥有通信效率高，通信稳定性强，对多种编程语言支持度好并拥有支持高并发系统的特点。

Mavlink 传输时基本单位是消息帧，消息帧的结构如下图所示。

索引	名称	内容	值	含义
0	STX	数据包起始标志	0xFE(v1.0)/0x55(v0.9)	标识新消息的开始
1	LEN	有效载荷长度	0-255	标识该消息包中负载长度
2	SEQ	数据包序列号	0-255	消息发送序列号。用于检测数据包的丢失
3	SYS	系统ID	1-255	发送该包的系统ID。能够区分在同一网络中不同的MAV
4	COMP	组件ID	0-255	发送该包的组件ID,能够区分同一系统的不同发送设备。如: IMU和AutoPilot
5	MSG	消息ID	0-255	该ID定义了有效负载的“含义”以及负载应被如何正确的解码
6至n+6	PAYLOAD	消息载荷	0-255 字节	消息内部的负载信息
n+7	CKA	校验位 (低8位)	-	CRC校验码
n+8	CKB	校验位 (高8位)	-	CRC校验码

图 3.6 Mavlink 的消息帧结构

3.4.2 PC 上位机和飞行控制板 Pixhawk 的连接通讯

PC 上位机可以使用 Micro-USB 线与飞行控制板 Pixhawk 进行有线连接，进行通讯和一些仿真开发操作，但仅用于调试系统，在完整的系统中并不使用这种连接方式。在

本文的 4.3.2 节中，有关于这种连接通讯方式的进一步介绍；在本文的 5.1.3 中有关于这种连接通讯方式的实验测试结果。

3.4.3 机载电脑树莓派和 PC 上位机的连接通讯

机载电脑树莓派和 PC 上位机的通讯方式有两种，分别用于系统软件调试和系统硬件实验两类，它们都是基于局域网组网后的 SSH 技术进行的。

局域网组网的方法很多，本文使用的方法是让 PC 上位机连接树莓派开启的 WIFI 热点，此举为方便 SSH 登录所做。Secure Shell(SSH) 是一个建立在应用层上，由国际互联网工程任务组 IETF(The Internet Engineering Task Force) 制定的安全网络协议。它是一个可以提供足够的安全性的，专为远程登录会话和其他网络服务而生的，且可有效弥补网络漏洞的协议。我们可以通过 SSH，对传输数据进行加密，从原理上防止 DNS 或者 IP 欺骗。此外，使用 SSH 的一个额外的好处为其传输的数据都是经过压缩的，故其传输速度十分可观，该协议已成为大多数 Linux 内核（例如 Ubuntu 系统）的标准配置。

（1）基于虚拟化技术的用于系统软件调试的通讯方式

该通讯方式为在 PC 和树莓派局域网组网的基础上，使用虚拟化技术手动 SSH 远程登陆机载电脑树莓派。

具体为在 Windows10 系统中开启 Linux 系统的虚拟机，连接树莓派上的 WIFI 热点进行局域网组网，为下一步通讯创造基础条件。继而以超级账户权限使用 vim 软件更改存放路径在/etc 的 hosts 文件，添加树莓派的 IP 地址 10.42.0.1 于其中。由此，便可以使虚拟机中的 Ubuntu 系统正确解析机载电脑树莓派所对应的 DNS 地址，进行相对简单的 SSH 操作，即可使用 PC 机远程操作进行飞行控制的树莓派进行所需的控制命令。

在本文的 4.3.2 节中有关于虚拟化技术和该方式的进一步介绍；在本文的 5.2.2 节中有关于该方式的实验测试结果。

（2）基于 Paramiko 库的用于系统硬件实验的通讯方式

该通讯方式为在 PC 和树莓派局域网组网的基础上，调用 Paramiko 库用程序自动 SSH 远程登录机载电脑树莓派。

在本文的 4.3.2 节中有关于 Paramiko 库与该方式的进一步介绍；在本文的 5.2.3 节中有关于该方式的实验测试结果。

3.5 本章小结

本章对语音控制无人机系统的硬件设计进行了详细叙述，主要包括验证语音控制系统的无人机的硬件组成部分，由飞行控制板 Pixhawk 和机载电脑树莓派组成的无人机中

控系统。对于硬件组成部分进行了具体而细节的描述，包括到电机型号；因系统设计重心并不在此部分，对于飞行控制板 Pixhawk 和机载电脑树莓派的相关部分以简要介绍其功能为主，原理性分析为辅，其中还详细介绍了对于机载电脑树莓派在 Z410 无人机上的安装过程，本章内容是语音控制无人机系统的验证性部分展示。

4 语音控制无人机软件部分

4.1 软件系统组成部分

软件系统主要分为两大部分，语音识别程序，和基于 Dronekit 的飞行控制程序。

语音识别程序主要用于将用户口授的语音指令转化为便于计算机处理的.wav 格式，然后进一步将其转化为便于无人机飞行控制的文本字符串，并具有一定的模糊匹配和命令冲突检查功能，运行于 PC 上位机上，这是用语音来进行无人机系统控制的“语音”部分。

基于 Dronekit 的飞行控制程序主要用于基于语音识别程序的识别结果，结合飞控 Pixhawk 和 Dronekit 进行特定的飞行控制，运行于机载电脑树莓派上，其中还包含了 PC 上位机和机载电脑树莓派的局域网组网与 SSH 登录方法，这是用语音来进行无人机系统控制的“控制”部分。

4.2 语音识别程序

4.2.1 基于门控卷积神经网络的语音识别算法

本课题使用的语音识别算法是一个开源门控卷积神经网络（Gated Convolutional Network）项目 MASR，据其项目页介绍，其网络结构类似于 Facebook 在 2016 年提出的 Wav2letter。但是使用的激活函数不是 ReLU 或者是 HardTanh，而是 GLU（门控线性单元），因此称作门控卷积网络。根据实验，使用 GLU 的收敛速度比 HardTanh 要快，模型使用 AISHELL-1 数据集训练，共 150 小时的录音，覆盖了 4000 多个汉字。

4.2.2 基于 PC 端的语音识别程序

PC 端的语音识别程序以 Python 语言进行开发，用于获取并识别用户语音，提取其中命令词，并将命令发送给机载电脑树莓派，是连接用户语音和无人机中控系统的桥梁，其主要流程图可见下图所示。

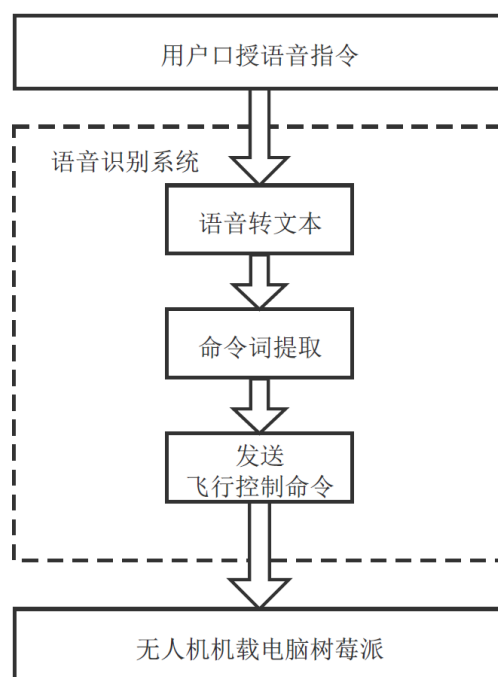


图 4.1 程序主要结构图

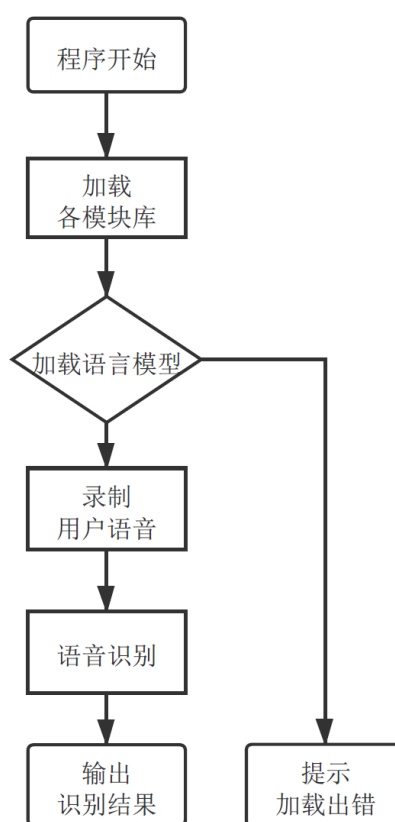


图 4.2 语音识别程序流程图

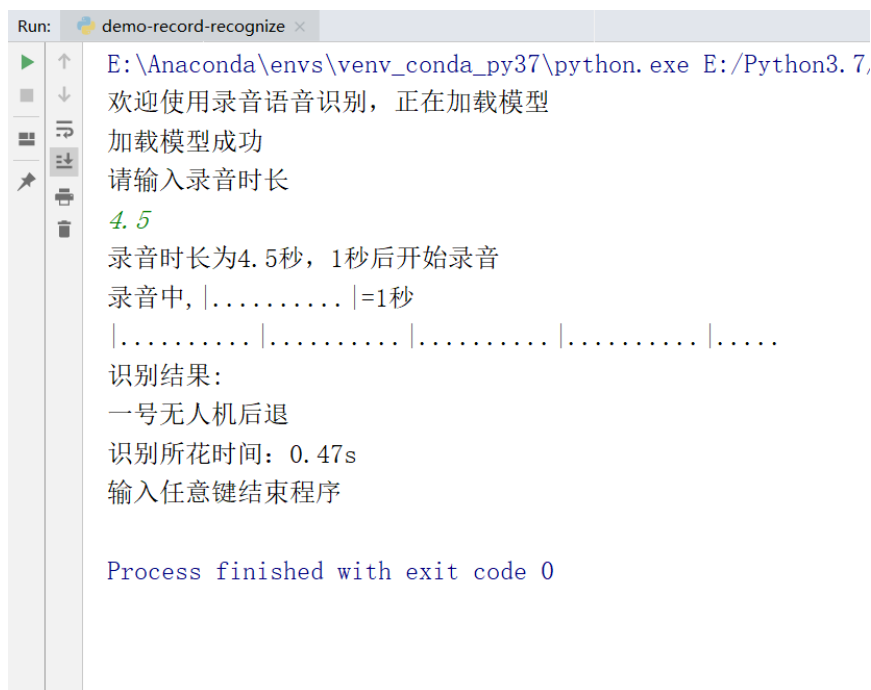
在 PC 平台基于开源项目 MASR 进行了语音识别程序的编写调试工作，并且完成了原理性验证，代码风格以主函数代码短小精悍，各模块内紧耦合，各模块间松耦合，基于类 GatedConv 进行方法操作而不是使用函数操作。以下为尚未加载语言模型的语音识别程序，可以发现其除了识别率上的弱项外，代码的原理性已获验证。

```
1 import _init_path
2 from models.conv import GatedConv
3 from record import record
4 import time
5 print("欢迎使用录音语音识别，正在加载模型")
6 ...
7
8 try:
9     model = GatedConv.load("gated-conv.pth") #如果加载预训练模型出错就捕获一下这个异常
10 except Exception as e:
11     print(e)
12     print("出现异常")
13 else:
14     print("加载模型成功")
15 record_time_input=eval(input("请输入录音时长\n"))
16 print("录音时长为{:}秒，1秒后开始录音".format(record_time_input))
17 time.sleep(1)
18 record("record.wav", time=record_time_input)
19 start_time=time.perf_counter()
20 text = model.predict("record.wav")
21 end_time=time.perf_counter()
22 print("")
23 print("识别结果:")
24 print(text)
```

图 4.3 语音识别程序运行代码（部分）

其中，在第 9 行使用语句 `model = GatedConv.load("gated-conv.pth")` 来加载已训练好的模型，考虑到模型加载有可能失败，使用了 `try-except-else` 段进行异常捕获，如果出现问题则返回异常类型。在第 18 行使用 `record()` 函数进行用户语音录制，意为将录制的时长自定义的音频保存为 "record.wav"，用以下一步进行语音识别。在第 20 行进行语音识别，使用 `model` 的 `predict` 方法对已保存好的用户语音进行识别，并将结果以文本形式返回给变量 `text`。

在没有加载语言模型的情况下，识别程度也基本达到了项目要求。但由于尚未加载语言模型，模型的识别率相对较低。但若使用缓慢，标准的普通话进行语音命令时，可以得到相对较好的语音识别结果，如下图所示结果。



```

Run: demo-record-recognize x
E:\Anaconda\envs\venv_conda_py37\python.exe E:/Python3.7,
欢迎使用录音语音识别, 正在加载模型
加载模型成功
请输入录音时长
4.5
录音时长为4.5秒, 1秒后开始录音
录音中, |..... |=1秒
|..... |..... |..... |..... |.....
识别结果:
一号无人机后退
识别所花时间: 0.47s
输入任意键结束程序

Process finished with exit code 0
    
```

图 4.4 语音命令为“一号无人机后退”的运行结果

考虑到实际使用场合中对于语音识别率的要求，本程序加载语言模型的必要性是十足的，下一节即将详细介绍基于 Docker 的语言模型加载，以其获得较好的语音识别率，同时也有利于程序的编写。

在将用户口授的语音指令转化为便于计算机处理的.wav 格式文件再转化为便于飞行控制的字符串后，进行基础的飞行控制命令词提取。其中，程序的设计包含一定的模糊匹配设计，例如意思相近的“停机”、“停止”、“停下”、“关机”等都对应同样的飞行控制指令“停机”。主要的命令词提取思想是将已转化为字符串的用户语音指令与预先设置的控制命令词进行遍历匹配，当发现匹配成功时，将待发送命令放入设置的命令缓存区，准备发送。一定的模糊匹配设计可以降低用户的语音控制学习成本，不用机械记忆特殊的命令词，而可以更贴近人直觉的词汇进行语音控制。

进一步，在命令缓存区进行冲突命令检查。例如当遇到用户口授“无人机左转，前进，不，立刻停机”时，检测到冲突的预先设置命令“左转”、“前进”和“停机”时，发送“停机”命令，忽视“左转”和“前进”命令。命令缓存区的设立目标之一是为了给予用户一个在紧急状态下的“急停按钮”，同时对于拒绝执行例如“无人机左转，不还是右转回去吧”这类明显冲突且包含撤销执行含义的命令。

发送飞行控制命令的部分在本文的 4.3.2 中有详细论述。

4.2.3 基于 Docker 的语音识别程序移植与语言模型加载

(1) 程序运行环境与容器化技术

任何程序的正常运行都需要相应的运行环境，但随着算法的空间复杂度和时间复杂度的增长，实现算法的程序所需的运行环境也变得愈发复杂。同样的程序和运行环境安装流程在不同的硬件和软件系统中很可能表现并不一致，例如许多开发者在运行 Windows 系统的 PC 上进行程序开发，但在运行 Linux 系统的服务器上运行程序。

要解决这样的环境冲突问题，主要的技术路线有使用虚拟机和容器技术两种，著名的开发软件 VMware 和 docker 正是他们的代表。

虚拟机技术是从硬件层面上对待模拟的操作系统进行模拟，以实现虚拟化运行的目的。但操作系统的运行需要占用许多内存开销，并且其虚拟出的虚拟机的性能往往难以达到宿主机的性能，从这个层面上说，如果使用虚拟机技术仅仅来运行程序的话，内存资源开销和利用率都并不可让人满意，从工程学角度甚至可以说是极大浪费。

容器技术的英文原名为“Container”，又可翻译为“集装箱”。集装箱由于具有各集装箱低耦合度，规格化且制式化，可反复使用并能快速装卸载等优点对海运商业的发展具有非常大的促进作用。正如集装箱之于海运业，容器化技术被创造的初衷正是为了减少程序的环境配置问题，使得程序的广泛性得以提升，让已编译的程序在不同的运行环境中都可以正常使用。

而容器技术区别于虚拟机在硬件层上虚拟化，其核心差别为容器技术是在操作系统之上进行虚拟化运作，其没有虚拟机造成的额外损耗，它的运行效率更高并且部署速度也更快^[21]。

(2) 基于容器化技术的 Docker 项目

Docker 是一个于 2013 年发布，基于 Go 语言的可实现容器化技术的开源项目。我们可以使用该项目来进行我们所需的容器化技术需求，例如快速部署程序运行环境。

“Docker”一词可直译为“码头装载工人”，如其 logo 所示，上半部是待运输的集装箱（程序环境），下半部是准备将其运载至（部署到）其他运行环境的鲸鱼。

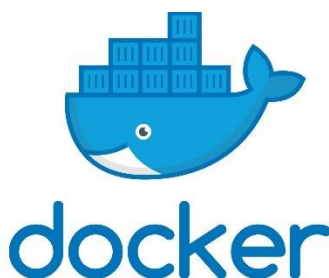


图 4.5 容器化技术 docker 项目的 logo

Docker 将目标程序及其各种环境依赖打包为 Docker 容器，运往其他运行环境以部署运行。正如集装箱里的货物和集装箱在哪个港口或者货船无关，由于容器化技术的使用，目标程序无论其硬件环境为 PC 还是单片机，软件环境是 Windows 还是 Linux，程序的表现一致性都将十分稳定。

Docker 的架构使用了常见的 CS (Client/Server) 架构，也就是客户机/服务机模式。客户机负责处理用户输入的例如 “docker run” 等各种命令，而真正工作进行复杂的程序处理的是服务机。客户机和服务机是一个逻辑上的概念划分，并不要求物理上具有两台计算机，例如本文实现的基于容器化技术 Docker 的 MASR 环境移植，正是在同一台 PC 机上由不同的程序分别充当客户机和服务机，协同进行语音识别的功能。

(3) 基于 docker 的 MASR 环境移植

MASR 项目的运行环境主要依赖为解释器 Python (3.6.8)，深度学习框架 Pytorch (1.0.1)，语音处理库 librosa (0.6.0)，CS 架构 flask (1.1.2)，麦克风接口调用包 Pyaudio (0.2.11) 和 beam search 解码器 ctcdecode。其中 ctcdecode 的安装由于官方项目页仅发布了待编译源码，为了避免在不同的软硬件运行环境中出现程序表现一致性不同的问题，课题使用容器化技术 Docker 进行 ctcdecode 的安装部署。

使用容器化技术 Docker 进行语言模型的加载时，在 Linux 终端中运行命令 “docker run -v \$(pwd):/workspace/masr -p 5000:5000 libai3/masr-env:latest”，以建立起 PC 机中的物理路径和镜像中的虚拟路径的对应关系，包括工作目录的映射和端口的映射，继而在本机的 5000 网络端口建立起了一个加载好语言模型的服务机，以便客户机使用其功能。

下图为基于 Docker 的加载了语言模型的部分程序代码，其中第 7 行 server = “http://localhost:5000/recognize” 设置本地服务器地址为 5000 端口，并以变量 server 存储服务器地址；第 15 行 r=requests.post(server, files=files) 对本地服务器发送语音识别请求。加载了语言模型后，语音识别程序在安静无噪音环境下的识别成功率获得较大提升，已可满足作为无人机语音控制的基本需要。

```
1 import requests
2 import _init_path
3 import feature
4 import time
5 from record import record
6
7 server = "http://localhost:5000/recognize"
8
9 record("record.wav", time=3)
10
11 f = open("record.wav", "rb")
12
13 files = {"file": f}
14 start_time=time.perf_counter()
15 r = requests.post(server, files=files)
16 end_time=time.perf_counter()
17 print("")
18 print("识别结果:")
19 print(r.text)
20 print("识别所花时间: {:.2f}s".format(end_time-start_time))
21 if ("起飞" in r.text) or ("启动" in r.text) or ("开始" in r.text):
22     print("发送 起飞 命令")
```

图 4.6 加载了语言模型的语音识别程序

4.3 基于 Dronekit 的飞行控制程序设计

4.3.1 飞行控制程序基本流程

树莓派系统上电先进行软硬件初始化, 和无人机飞控 Pixhawk 尝试通讯, 进而无人机飞行控制软件系统初始化, 建立上位机连接链路, 搜索 GPS 信号并进行各硬件传感器校准, 等待上位机或遥控器的进一步指令。当有控制命令经 Mavlink 协议传输到来时, 调用对应函数代码段, 解析实现控制目标所需变化的 PID 值, 基于 PID 控制系统反解出具体动作的四个电机的目标功率, 继而由 PWM 控制进行功率控制, 以最终达到预想的飞行动作, 下图为飞行控制程序的基本流程图。

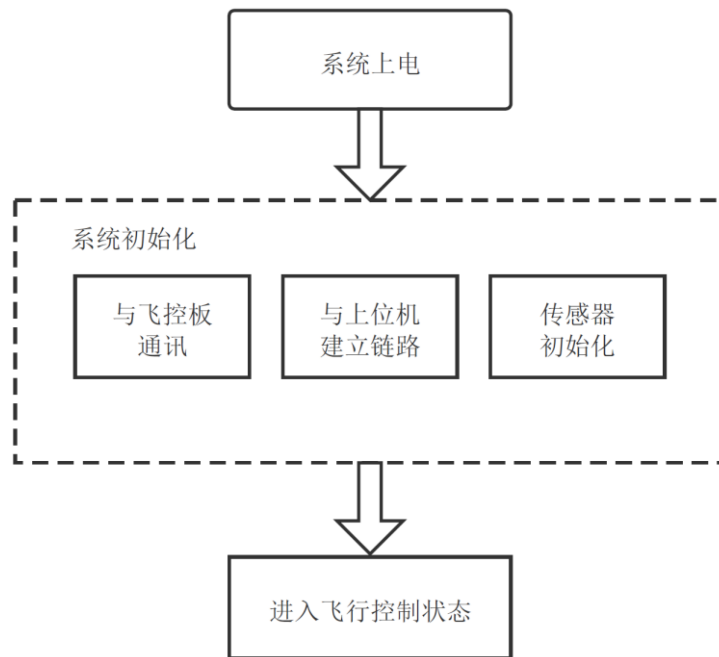


图 4.7 飞行控制程序结构图

对于飞行控制状态，详细流程图见下方所示。

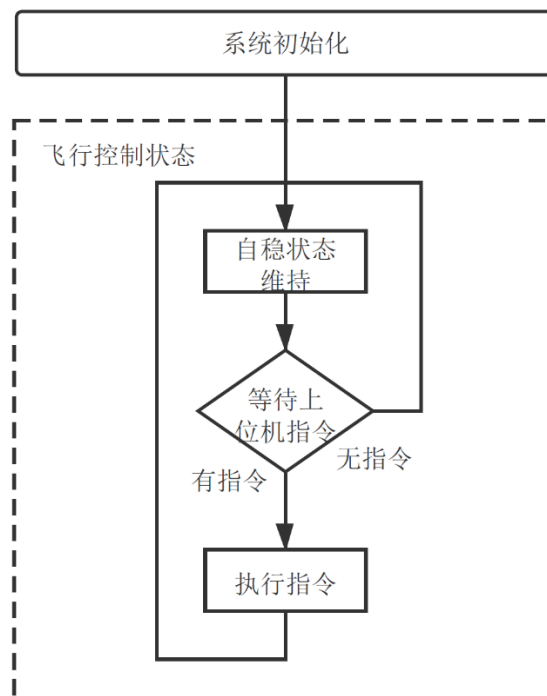
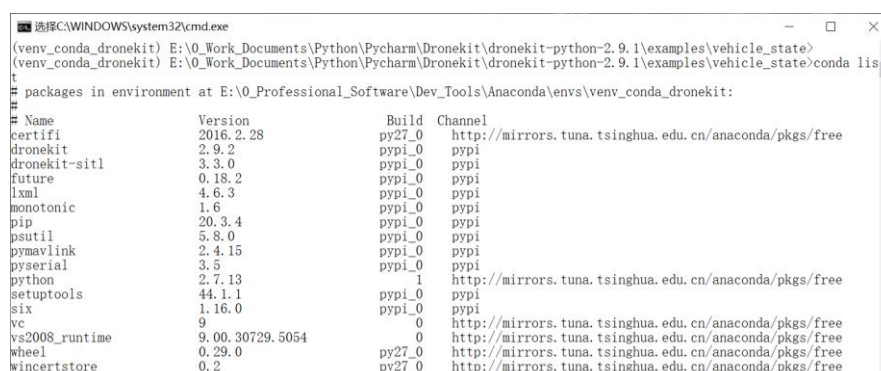


图 4.8 飞行控制状态流程图

4.3.2 用于飞行控制的 Dronekit 与上位机的连接通讯

如其官方文档所述，Dronekit 只支持到 Python2.x 版本，对于目前常见的 Python 语言解释器 Python3.x 版本兼容性并不好^[22]，故要使用 Dronekit 一般来说需要对开发环境进行新的设置。进行基本的环境配置后，环境需求如下图所示：



```

选择C:\WINDOWS\system32\cmd.exe
(venv_conda_dronekit) E:\0_Work_Documents\Python\Pycharm\Dronekit\dronekit-python-2.9.1\examples\vehicle_state>
(venv_conda_dronekit) E:\0_Work_Documents\Python\Pycharm\Dronekit\dronekit-python-2.9.1\examples\vehicle_state>conda list
# packages in environment at E:\0_Professional_Software\Dev_Tools\Anaconda\envs\venv_conda_dronekit:
#
# Name                      Version            Build             Channel
# -----
certifi                     2016.2.28          py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
dronekit                   2.9.2              py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
dronekit-sitl              3.3.0              py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
future                     0.18.2             py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
lxml                       4.6.3              py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
monotonic                  1.6                py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
pip                        20.3.4             py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
psutil                     5.8.0              py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
pymavlink                  2.4.15             py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
pyserial                   3.5                py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
python                     2.7.13             1                 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
setuptools                 44.1.1             py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
six                        1.16.0             py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
vc                          9                  0                 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
vs2008_runtime             9.00.30729.5054    0                 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
wheel                      0.29.0             py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
winertstore                 0.2                py27_0            http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free

```

图 4.9 Dronekit 运行所需的环境

(1) 使用 USB 串口对 PC 上位机和 Pixhawk 飞控板进行通讯

使用 Micro-USB 连接线，将 Pixhawk 飞控板和 PC 上位机进行连接，无需其他设备，PC 可直接对 Pixhawk 飞控板供电，运行命令 `python vehicle_state.py --connect com3` 以连接至 Pixhawk 飞控，对 Dronekit 库和 Pixhawk 飞控板进行基本连接测试。



```

C:\WINDOWS\system32\cmd.exe
(venv_conda_dronekit) E:\0_Work_Documents\Python\Pycharm\Dronekit\dronekit-python-2.9.1\examples\vehicle_state>python vehicle_state.py --connect com3
Connecting to vehicle on: com3
CRITICAL:autopilot:PreArm: Throttle below Failsafe
CRITICAL:autopilot:PreArm: Hardware safety switch

Get all vehicle attribute values:
Autopilot Firmware version: APM:Copter-3.6.11
Major version number: 3
Minor version number: 6
Patch version number: 11
Release type: rc
Release version: 0
Stable release?: True
Autopilot capabilities
Supports MISSION_FLOAT message type: True
Supports PARAM_FLOAT message type: True
Supports MISSION_INT message type: True
Supports COMMAND_INT message type: True
Supports PARAM_UNION message type: False
Supports ftp for file transfers: False
Supports commanding attitude offboard: True
Supports commanding position and velocity targets in local NED frame: True
Supports set position + velocity targets in global scaled integers: True
Supports terrain protocol / data handling: True
Supports direct actuator control: False
Supports the flight termination command: True
Supports mission_float message type: True
Supports onboard compass calibration: True
Global Location: LocationGlobal:lat=0.0,lon=0.0,alt=3.24
Global Location (relative altitude): LocationGlobalRelative:lat=0.0,lon=0.0,alt=3.246
Local Location: LocationLocal:north=None,east=None,down=None
Attitude: Attitude:pitch=0.0156093062833,yaw=2.93094587326,roll=0.00842101406306
Velocity: [0.0, 0.0, -0.01]
GPS: GPSInfo:fix=1,num_sat=0
Gimbal status: Gimbal: pitch=None, roll=None, yaw=None
Battery: Battery:voltage=0.694,current=1.69,level=69
BSP OK?: False
Last Heartbeat: 0.75
RangeFinder: RangeFinder: distance=None, voltage=None
RangeFinder distance: None
RangeFinder voltage: None
Heading: 167
Is Armable?: False
System status: STANDBY
Groundspeed: 0.0103330332786
Airspeed: 0.0
Mode: STABILIZE
Armed: False
Waiting for home location ...

```

图 4.10 基本通讯测试程序 `vehicle_state.py` 的运行结果

(2) 使用 Paramiko 库通过 SSH 登录对 PC 上位机和 Pixhawk 飞控板进行通讯

Paramiko 库是一个用于在程序中使用 SSH 协议对远程服务器进行控制的 Python 第三方库，是 OpenSSH 的进一步封装。不同于常见的手动输入命令以控制远程服务器的方法，Paramiko 库可以让程序通过调用其函数和方法以实现远程服务器控制自动化。

通过一些局域网组网的手段，PC 上位机可以在程序中调用 Paramiko 库以 SSH 登录至机载电脑树莓派，进而执行 Dronekit 脚本以达成 PC 上位机和 Pixhawk 飞控板通讯的目的。局域网组网的方法有很多种，本文使用的方法为通过机载电脑树莓派建立 WIFI 热点以让树莓派和 PC 上位机处于同一个局域网内。

下图为使用 Paramiko 库进行程序自动 SSH 登录并执行命令的部分代码，主要流程为设置基本 IP 地址和端口，设置用户名和密码，SSH 登录后在树莓派上执行命令 python connect.py 文件，该命令意为使用 python 解释器执行 connect.py 文件，该文件如本文 2.2 节所述，是基本的飞控通讯程序，可以返回诸如此时无人机姿态角等信息。

```
1  # -*- coding: utf-8 -*-
2  import paramiko
3  # 输入用户名、密码、ip等
4  ip = "10.42.0.1"
5  port = 22
6  user = "lj"
7  password = "123456"
8  # 创建一个ssh对象
9  ssh = paramiko.SSHClient()
10 # 自动选择yes
11 ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
12 # 建立连接
13 ssh.connect(ip,port,user,password,timeout = 10)
14 # 输入linux命令
15 #ssh.exec_command('cd pix;pwd')
16 #ssh.exec_command('pwd')
17 while True:
18     # 等待输入命令
19     temp = str(input("input:"))
20     temp="python connect.py"
21     print(temp)
22     stdin,stdout,stderr = ssh.exec_command(temp)
23     # 输出命令执行结果
24     result = stdout.read()
25     print(result)
26     input("结束")
27 # 关闭连接
28 ssh.close()
```

图 4.11 涉及 SSH 登录的部分程序

4.3.3 基于机载电脑树莓派的飞行控制程序

飞行控制程序的理论核心目标是建立起“飞行控制目标”和“代码实现手段”之间的桥梁，此节中以两个示例介绍课题使用的飞行控制程序核心代码段的部分原理和具体实现。

以实现“目标无人机向左飞行，速度 0.2m/s，飞行时间 5 秒”为例，介绍课题中的飞行控制程序核心代码。其中，前后方向速度使用 `veloci_x` 表示，左右方向速度使用 `veloci_y` 表示，垂直高度方向速度使用 `veloci_z` 表示。则在控制区域中，先显式进行变量赋值如下所述 `velocity_x = 0;velocity_y = -0.2;velocity_z = 0;duration = 5`

再调用飞行控制函数 `send_body_ned_velocity(velocity_x, velocity_y, velocity_z, duration)`，即可实现目标。

进一步地，以实现“用户口授起飞指令并成功使得无人机起飞至默认高度”的目标，则需预先设置默认高度，例如 5 米，再调用起飞函数 `arm_and_takeoff(aTargetAltitude=5)` 即可，类似的“停机”等指令实现流程同理。而若是需要实现“用户口授左转指令并成功使得无人机左转默认度数 90 度”的目标，则需先对变量 `heading` 赋值为 90，再调用函数 `condition_yaw(heading, relative=True)` 即可，若要实现类似的“右转”命令则将变量 `heading` 赋值为 -90 即可。

4.3.4 用于语音识别和无人机控制的程序设计

(1) 程序流程设计

用户在 PC 端使用蓝牙耳机进行语音指令下达，运行在 PC 端的语音识别程序使用 `Pyaudio` 获取用户语音，将用户语音存储为便于计算机操作的 `.wav` 格式音频文件为下一步基于卷积门控神经网络的语音识别程序使用；语音识别程序将音频文件转换为便于飞行控制操作的字符串文本，继而进行用户命令词提取过程；程序检测预先设置的诸如“起飞”，“左转”，“右转”，“停止”，“前进”，“后退”的命令词是否包含其中，如若，则放弃本次流程，视为用户下达了无效操作，如是，则准备调用 `Dronekit` 提供的飞行控制 API，为控制无人机做准备；程序使用机载树莓派热点，基于 `Paramiko` 库进行自动 `SSH` 登录，调用已写好的存放于树莓派上的飞行控制程序，进行相应的飞行动作。至此，便实现了用户使用自己的语音来进行无人机控制，这种多终端多程序的协作系统也即本论文的名称，基于深度学习的嵌入式语音控制无人机设计。

4.3.5 Ardupilot 开发环境和用于飞行模拟调试的 Dronekit-SITL

(1) Ardupilot 和 Dronekit-SITL 概述

ArduPilot 一个开源的无人机开发系统，它具有成熟且开源的特性。它更为飞行模型行业所知的名字是 APM，其名字的 “Ardu” 取自原先最早的开发环境 Arduino，这是一款开源电子开发平台。APM 系统不仅支持传统直升机和多旋翼飞行器，还支持固定翼飞机，是一个跨平台跨硬件设备的通用性飞控系统。其目前对于许多飞控硬件板都有较好的支持，比如基于 ARM 的 32 位开源硬件架构 Pixhawk 等。

软件在环仿真 SITL(Software in the Loop)为仿真飞行的模拟器，该模拟器无需具体的无人机硬件即可正常使用。这个软件用 C++编译器构建仿真的自动驾驶环境，提供一个离线的 PC 机可执行文件，这可以用来测试无人机的行为和状态，在电脑内存中进行无人机飞行模拟。该系统的各个部分可以使用 UDP 进行连接，即拥有在某一特定计算机上运行的功能，又拥有在局域网络下另一台计算机上运行的能力。

(2) PC 上位机中的用于仿真调试的 Dronekit-SITL

很可惜的是用于仿真的 Dronekit-SITL（来自 ArduPilot）在机载电脑树莓派这般嵌入式平台上的支持度很低，但可在 PC 上位机上搭建基于 ArduPilot 的用于无人机代码仿真的 Dronekit-SITL 环境，再进行数据转发（例如使用 MavProxy 协议），实现对树莓派上的 Dronekit 代码通过 PC 上位机来进行仿真调试操作。此外，由于时间和技术限制，本文并未进行仿真调试开发，仅在此介绍本方法。

4.4 本章小结

本章内容主要包括语音控制无人机的软件部分，由语音识别程序和飞行控制程序两大分支组成，是本课题的核心研究内容。语音识别程序使用 Pyaudio 录制用户语音，使用基于门控卷积网络的语音识别算法进行语音识别，并通过添加模糊匹配和命令词提取这两个模块使得系统对于人类口授语音指令有更好的鲁棒性。飞行控制程序部分主要由介绍飞行控制库 Dronekit 组成，对其基本原理和主要 API 进行介绍，并详细说明了部分飞行控制函数的具体使用。此外，在 4.3.2 节中对于 Dronekit 代码和 PC 上位机的连接通讯方式还进行了深入分析，展示了软件系统的设计与实现。

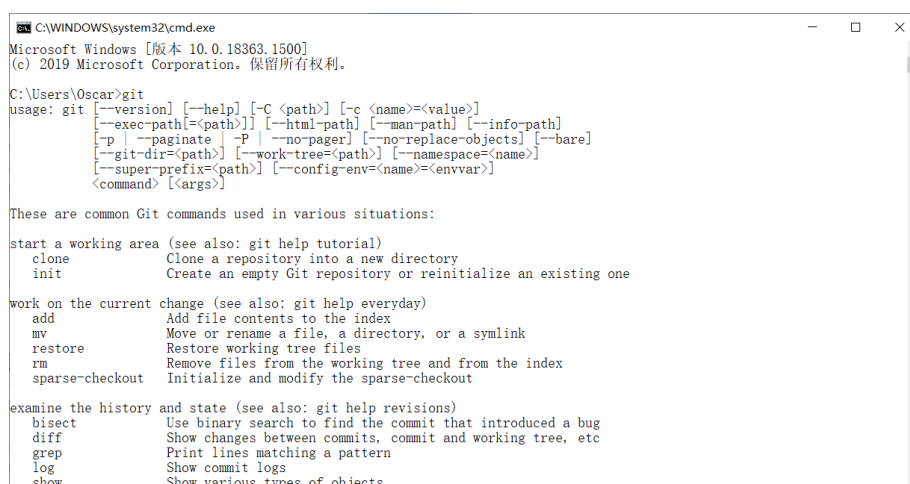
5 系统调试及实验结果

5.1 系统环境配置

5.1.1 PC 端基本环境配置

基本环境配置中使用的主要工具有版本管理工具 Git，第三方库与虚拟环境管理工具 Conda，深度学习库 Pytorch，

Git 是一个开源免费的分布式版本控制系统（VCS），在工业界中已取得了垄断性地位，使用率较高，多数开源项目都使用该系统进行版本控制。Github 则如其名，为 Git（版本控制系统）的 Hub（仓库中心），是使用 Git 作为其版本库格式进行项目托管的网站。Git 的基本操作有使用命令行界面和使用图形用户界面（GUI）两种风格，主要使用命令行进行一些基本设置，用户可以选择使用 GUI 来简化操作。



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.18363.1500]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Users\Oscar>git
usage: git [-version] [-help] [-C <path>] [-c <name>=<value>]
          [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
          [-p] [--paginate] [-P] [--no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          [--super-prefix=<path>] [--config-env=<name>=<envvar>]
          <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone             Clone a repository into a new directory
  init              Create an empty Git repository or reinitialize an existing one

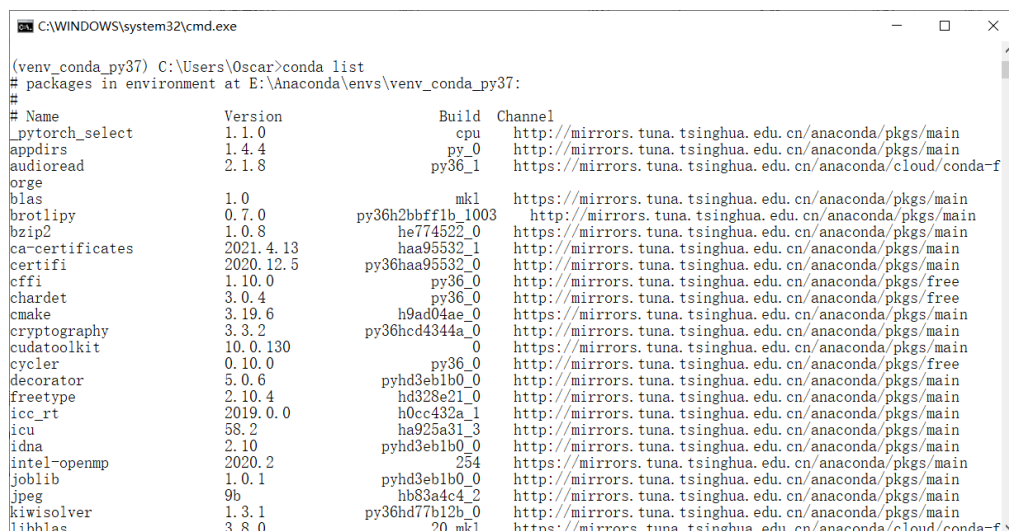
work on the current change (see also: git help everyday)
  add               Add file contents to the index
  mv                Move or rename a file, a directory, or a symlink
  restore           Restore working tree files
  rm                Remove files from the working tree and from the index
  sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
  bisect            Use binary search to find the commit that introduced a bug
  diff              Show changes between commits, commit and working tree, etc
  grep              Print lines matching a pattern
  log               Show commit logs
  show              Show various types of objects
  
```

图 5.1 Git 的命令行操作界面

在本课题的具体实践中，积攒了许多具体的实践细节，包括使用代理服务器访问解决 Github 访问速度过慢的问题，通过跳过证书检查来解决 SSL 证书错误的问题，通过手动设置代理端口 `git config --global http.proxy 192.168.137.1:25378` 来解决代理服务器无法访问 Github 的问题。

Conda 是一个开源跨平台语言无关的包管理与环境管理系统。允许用户方便地安装不同版本的二进制软件包与该计算平台需要的所有库。anaconda 是 conda 的一个发行版本，自带了许多常用的库，也是本课题主要使用的环境管理系统，它可以对不同工程激活不同虚拟环境避免不同版本的库互相干扰。



```
(venv_conda_py37) C:\Users\Oscar>conda list
# packages in environment at E:\Anaconda\envs\venv_conda_py37:
#
# Name                    Version            Build    Channel
_pytorch_select           1.1.0              cpu      http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
appdirs                   1.4.4              py_0     http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
audioread                 2.1.8              py36_1   https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-f
orge
blas                      1.0                mkl      https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
brotlipy                  0.7.0              py36h2b..._1003 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
bzip2                     1.0.8              he77452..._0 https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
ca-certificates           2021.4.13          haa9553..._1 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
certifi                   2020.12.5          py36haa...95532_0 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
cffi                      1.10.0             py36_0   http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
chardet                   3.0.4              py36_0   http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
cmake                     3.19.6             h9ad04a..._0 https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
cryptography              3.3.2              py36hcd...4344a_0 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
cudatoolkit               10.0.130           0         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
cycler                    0.10.0             py36_0   http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
decorator                  5.0.6              pyhd3e...b1b0_0 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
freetype                  2.10.4             hd328e...21_0 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
icc_rt                     2019.0.0           h0cc432..._1 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
icu                        58.2               ha925a...31_3 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
idna                       2.10               pyhd3e...b1b0_0 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
intel-openmp              2020.2             254       https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
joblib                     1.0.1              pyhd3e...b1b0_0 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
jpeg                      9b                 hb83a4...c4_2 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
kiwisolver                 1.3.1              py36hd...77b12b_0 http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg...
libblas                   3.8.0              20_mkl   https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-f
```

图 5.2 Conda 的主要操作界面与风格

本课题所使用的语音识别程序所依赖的主要环境为解释器 python3.7.3，深度学习框架 torch1.0.1，语音处理库 librosa0.6.0，下载包的主要途径为通过 conda install 命令从清华镜像源下载。

Pytorch 是一个深度学习框架，其中封装了各类工具方便调用，需要 Cuda 驱动来调用 GPU。Cuda 是一个并行计算加速框架，深度学习需要使用 Nvidia 的 GPU 并行计算加速框架。

cuDNN 是一个神经网络加速包，它和 Cuda 有大致对应关系。cuDNN 的安装也是依据 Cuda 选择 cuDNN 版本选取的，本课题使用 cuDNN7.6.5，只需解压文件并将其加入 CUDA 安装目录即可。

在安装 Cuda 的过程中，需要查询本机显卡型号，下载 CUDA 版本。本课题使用的版本是 Cuda10.0。在此过程中发现 Cuda 官方支持列表更新延迟的问题，已通过其他方式解决，便于环境变量的配置。

在 Pytorch 的安装过程中，若要在线安装，则可利用清华镜像下载对应的 Pytorch-GPU，下载后使用 Anaconda 安装。具体为在 cmd 下使用 cd 命令加载至下载的.bz2 文件所在目录，使用配置“conda install numpy mkl cffi”。若不采用在线安装，采用本地安装，则使用配置“conda install --offline pytorch-1.2.0-py3.7_cuda100_cudnn7_1.tar.bz2”。在此期间，由于 conda 访问官方源过于缓慢，添加清华镜像源，由此发现了“http error”，通过改为 http 访问解决。后又因“pytorch”包与“torch”包混名在安装时报错“no package found error”，已解决。由于 conda 无法自动安装所需的较低版本的 torch，故在清华镜像站找到低版本的 pytorch 版本（1.2.0）并手动下载，使用 conda install -of

flne 参数离线安装。离线安装后, import torch 报错“没有 torch.c 文件”, 该问题实际为当前虚拟环境 python3.6.8 版本过低, 升级为 3.7 后解决。升级后 conda 自动升级了 torch 至高版本, 无法运行低版本程序, 通过另行新建虚拟环境解决。

课题选择使用的语音处理库为 librosa。该库无法在 conda 官方源安装, 指定从清华源安装即可。安装库文件后, import 库文件报错“无法加载所需.dll 文件”, 此实际为版本序列不兼容的问题, 将库 librosa 由 0.8.0 降级为 0.6.0, 将库 numba 降级为 0.42.0 后, 又由于 python3.7.3 版本过高, librosa0.6.0 与其不兼容。Python 降级适应 librosa 后, 又与 torch1.2.0 不兼容, 故又降级了 torch 至 1.0.1, 并手动安装, 最终解决问题。

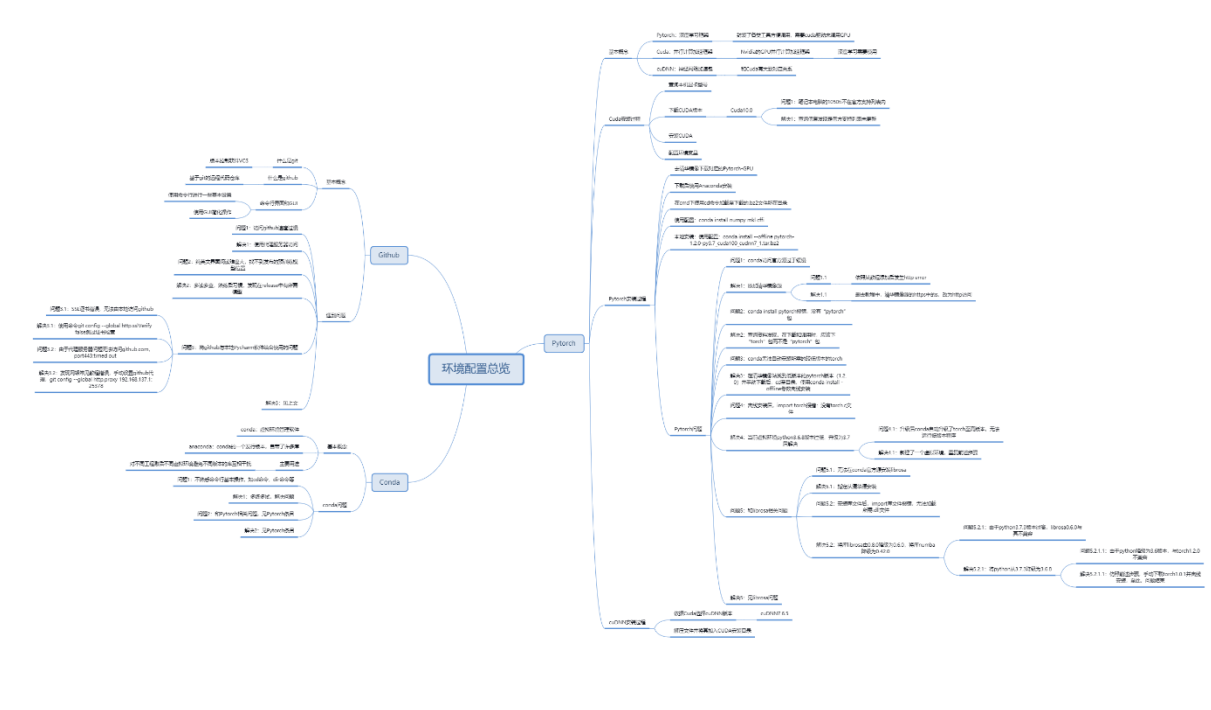


图 5.3 环境配置问题总览

5.1.2 基于容器化技术的语音识别程序测试

使用容器化技术 Docker 配置语音识别环境, 如下所示为容器运行界面窗口。

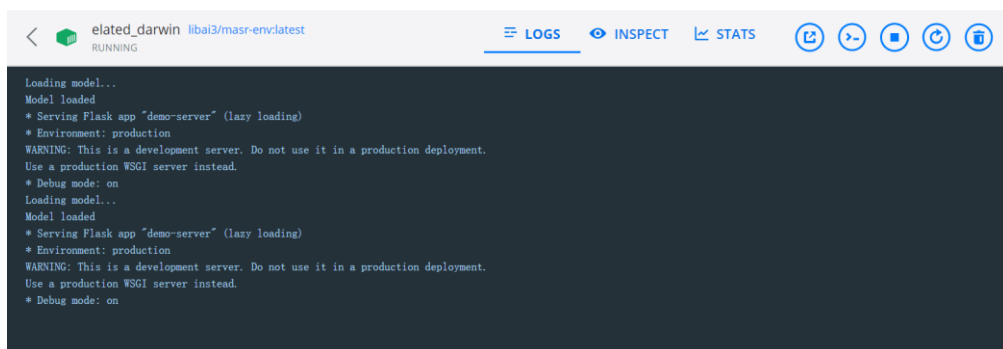


图 5.4 容器运行界面窗口

使用容器化技术加载语言模型后，口授指令“我的小无人机起飞”，查看语音识别程序的识别结果，程序正确提取出关键指令并输出起飞命令。可见系统的语音识别效果较好，并且可以从中提取出需要的关键命令词。

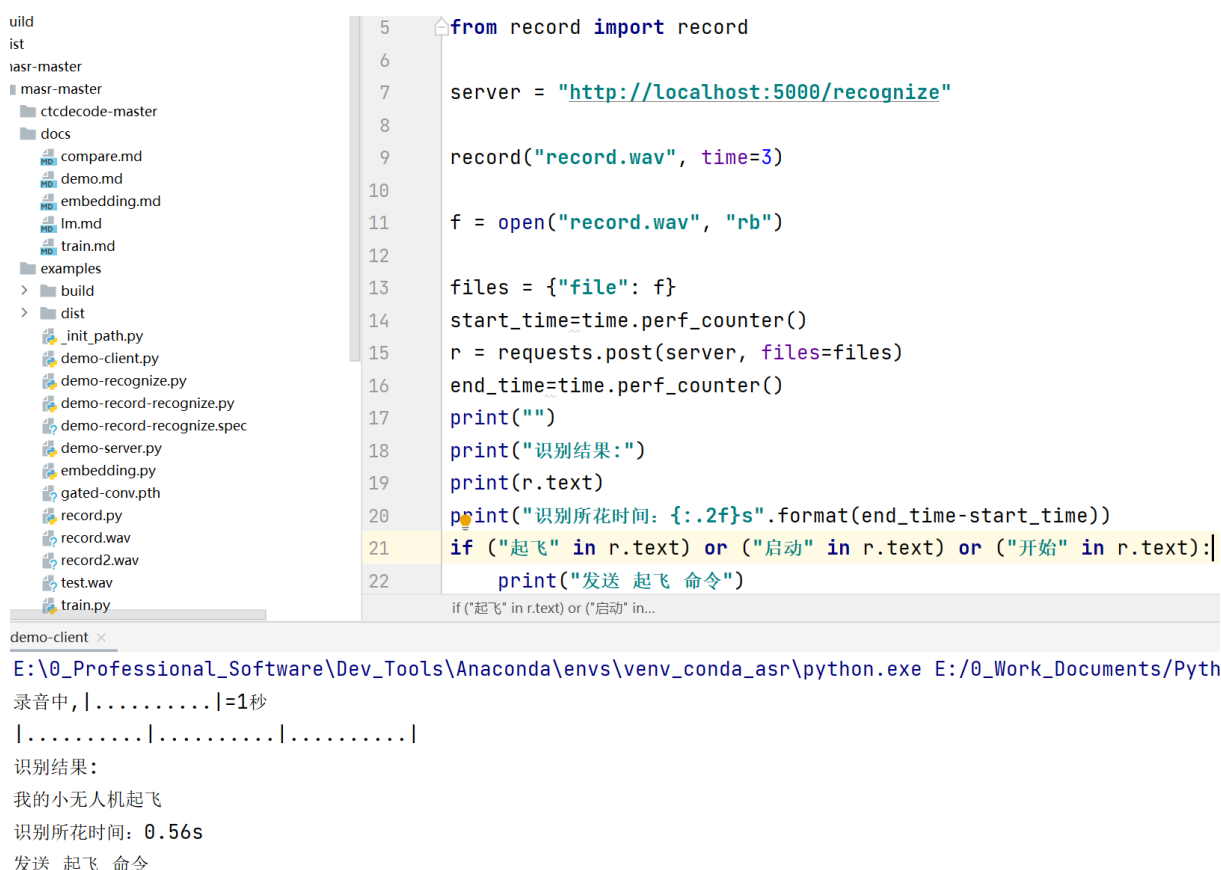
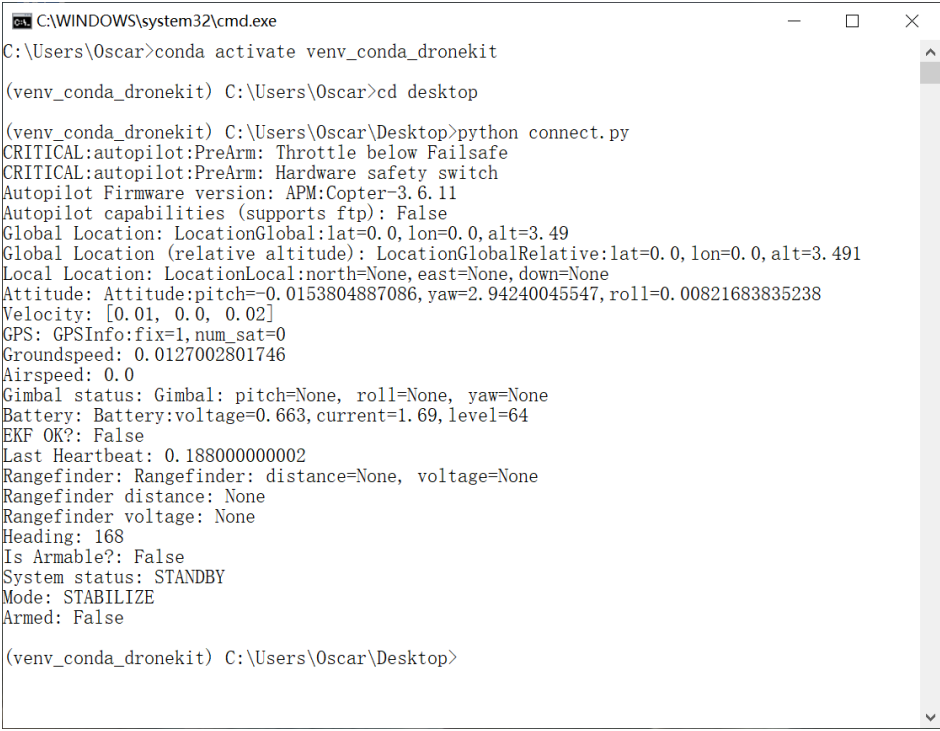


图 5.5 容器化技术加载语言模型并进行控制命令测试

5.2 PC 上位机和机载电脑树莓派以及飞控板的连接和通讯

5.2.1 基于有线连接的 Dronekit 飞控连接测试

以下为使用串口连接 PC 和飞控板 Pixhawk 后，运行通讯检测程序 `connect.py` 的结果，可见下图输出了无人机的基本信息。



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Oscar>conda activate venv_conda_dronekit

(venv_conda_dronekit) C:\Users\Oscar>cd desktop

(venv_conda_dronekit) C:\Users\Oscar\Desktop>python connect.py
CRITICAL:autopilot:PreArm: Throttle below Failsafe
CRITICAL:autopilot:PreArm: Hardware safety switch
Autopilot Firmware version: APM:Copter-3.6.11
Autopilot capabilities (supports ftp): False
Global Location: LocationGlobal:lat=0.0,lon=0.0,alt=3.49
Global Location (relative altitude): LocationGlobalRelative:lat=0.0,lon=0.0,alt=3.491
Local Location: LocationLocal:north=None,east=None,down=None
Attitude: Attitude:pitch=-0.0153804887086,yaw=2.94240045547,roll=0.00821683835238
Velocity: [0.01, 0.0, 0.02]
GPS: GPSInfo:fix=1,num_sat=0
Groundspeed: 0.0127002801746
Airspeed: 0.0
Gimbal status: Gimbal: pitch=None, roll=None, yaw=None
Battery: Battery:voltage=0.663,current=1.69,level=64
EKF OK?: False
Last Heartbeat: 0.1880000000002
Rangefinder: Rangefinder: distance=None, voltage=None
Rangefinder distance: None
Rangefinder voltage: None
Heading: 168
Is Armable?: False
System status: STANDBY
Mode: STABILIZE
Armed: False

(venv_conda_dronekit) C:\Users\Oscar\Desktop>
```

图 5.6 基本通讯程序 `connect.py` 运行结果

此通讯结果证明了 Dronekit 和飞控板 Pixhawk 通讯成功，并返回了有关信息。本程序运行的地点在一楼的二层运行，可见上图中返回的全球相对位置 `Global Location (relative altitude)` 中的相对高度 `alt` 为 3.49m 是准确的，GPS 模块初始化成功。这种连接方法主要用于只使用 PC 上位机和飞行控制板 Pixhawk 而不使用机载电脑树莓派进行软件和仿真调试的场合，但由于时间和技术限制，本文仅对这种连接方式进行了原理性验证，没有深入研究。

5.2.2 基于虚拟化技术的 SSH 远程登陆测试

在 PC 端基于虚拟化技术对语音识别程序进行实验，这种连接方法主要结合 `vim` 编辑器，用于远程编写和调试机载电脑树莓派上的基于 Dronekit 库的脚本文件。

使用基于虚拟机软件 VMware 的虚拟化 Ubuntu 系统，使用 SSH 远程登陆技术与树莓派进行通讯，下图为虚拟机界面。

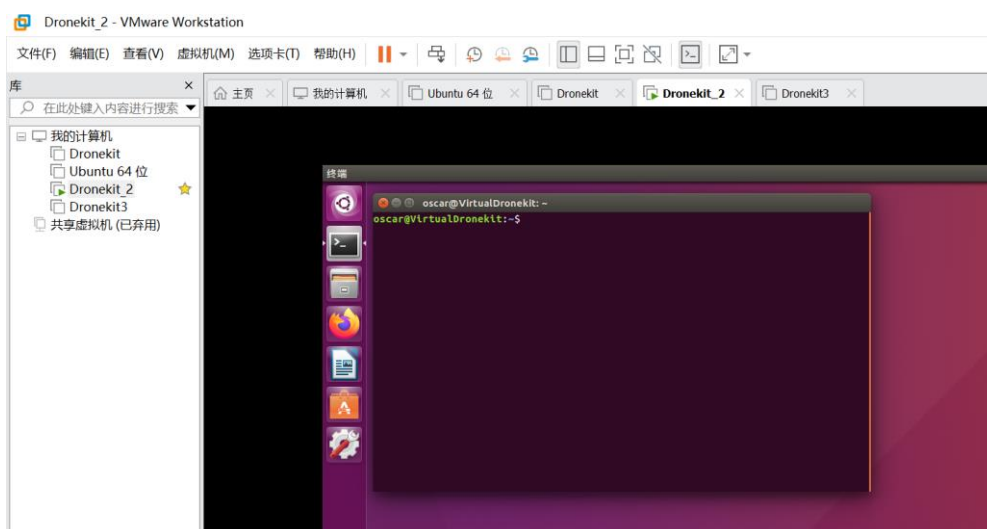


图 5.7 基于虚拟化技术软件 VMware 的虚拟化 Ubuntu 系统界面

下图为使用虚拟机中的 Ubuntu 系统进行 SSH 登录至树莓派上的 Ubuntu 系统并执行命令 `python connect.py` 的返回结果，该返回结果表明，运行 Windows10 系统的 PC 上位机可以通过虚拟机中的 Ubuntu 系统，使用 SSH 登录访问至树莓派上的 Ubuntu 系统，再使用 python 解释器运行基于 Dronekit 库的 `connect.py` 脚本文件，获得飞控的相关信息。

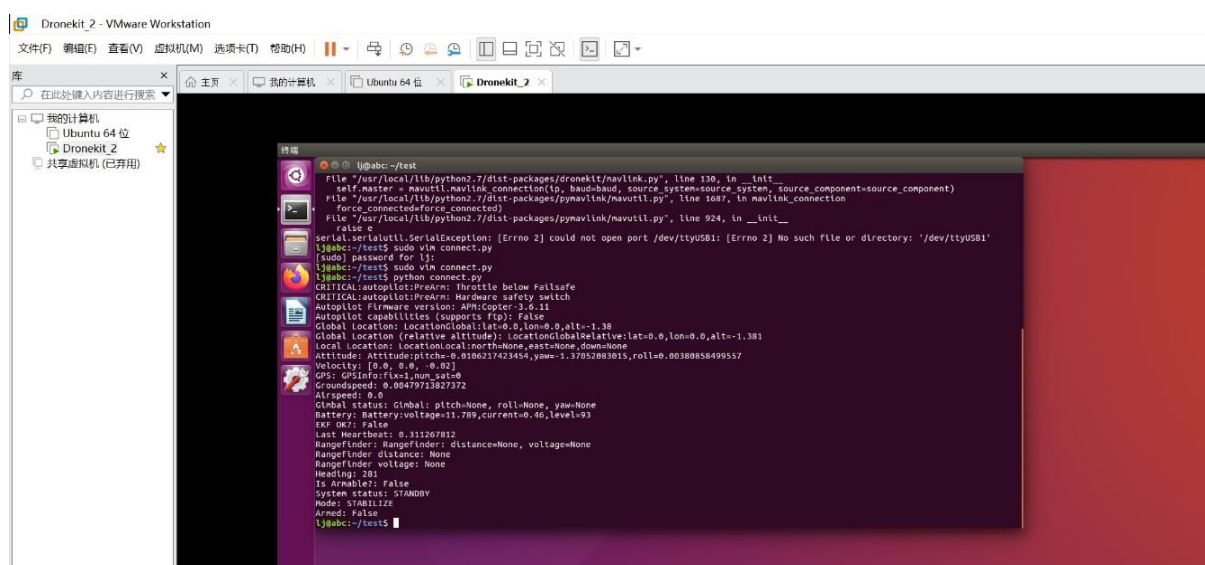


图 5.8 基于虚拟化技术的 SSH 远程登陆测试

5.2.3 基于 Paramiko 库的 SSH 远程登录测试

在 PC 端使用 Paramiko 库对系统进行自动远程登陆实验，这种通讯方式主要用于 PC 上位机的语音识别程序依据用户语音，自动 SSH 登录至机载电脑树莓派并执行对应脚本命令，是结合硬件调试的语音自动控制方法，也即具有实用化意义的自动远程登录方法。

下图为手持无人机并将其旋转，两次发送 `python connect.py` 命令获得的返回结果。图中框出部分为无人机姿态角 `pitch`, `yaw`, `roll`，其由于旋转而发生了变化，可见 PC 上位机与无人机通讯成功。

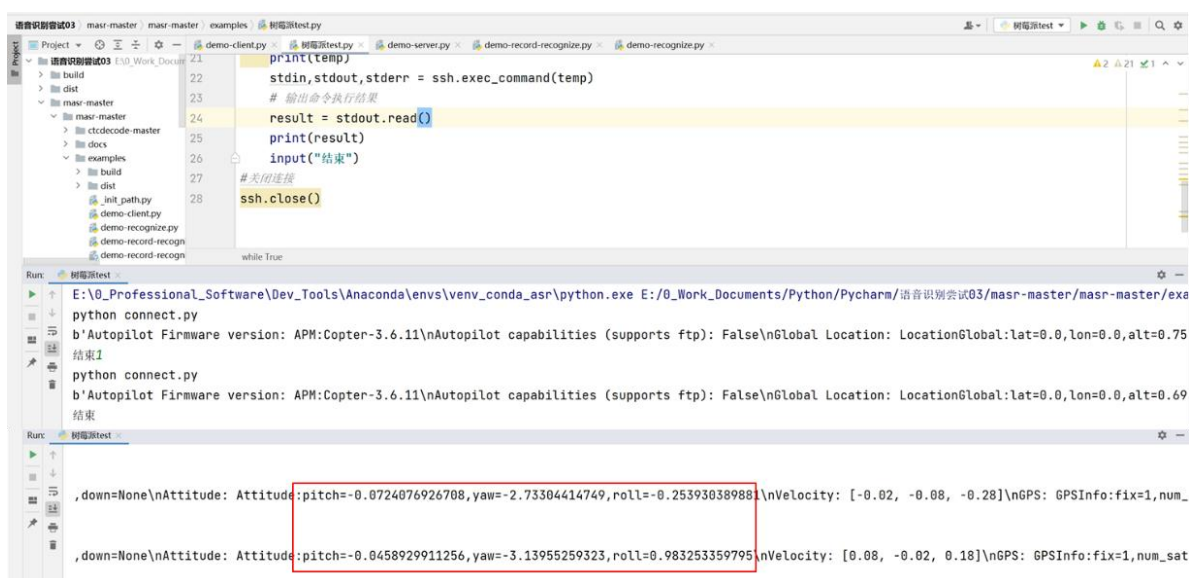


图 5.9 两次发送 `python connect.py` 命令的返回结果

5.2.4 无人机语音控制系统整体调试

将蓝牙麦克风插入 PC 机，通过设置 PC 上位机和机载电脑树莓派处于树莓派的 WIFI 热点下使其处于一个局域网内，在 PC 端运行 Docker 以便于语音识别程序加载语言模型，基于 Paramiko 库的自动远程登录程序已准备就绪，将无人机遥控器开机并对频以便于 Dronekit 控制无人机，机载电脑树莓派上的特定脚本如用于起飞并悬停的 `arm.py` 已放置于树莓派目录下，长按无人机安全锁以将其解除，完成无人机语音控制系统的前置准备工作。

运行语音识别程序，程序基本初始化，SSH 登录至机载电脑树莓派，加载语言模型，等待界面返回信息“录音中”时，口述语音指令“无人机起飞”，稍作等待，语音识别程序正确识别并提取出关键字“起飞”，于窗口提示用户“发送 起飞 命令”，进一步

于窗口提示用户“python arm.py”，意即使用 Paramiko 库的方法，在机载电脑树莓派使用 python 解释器运行已写好基于 Dronekit 用于起飞并悬停的脚本 arm.py，稍作等待无人机即起飞并悬停至 1 米，并在 arm.py 脚本控制下，悬停 1 秒后降落。

至此，完成了用户语音口授起飞命令，无人机成功起飞的系统效果，也即本系统，基于深度学习的嵌入式语音控制无人机系统实验成功。

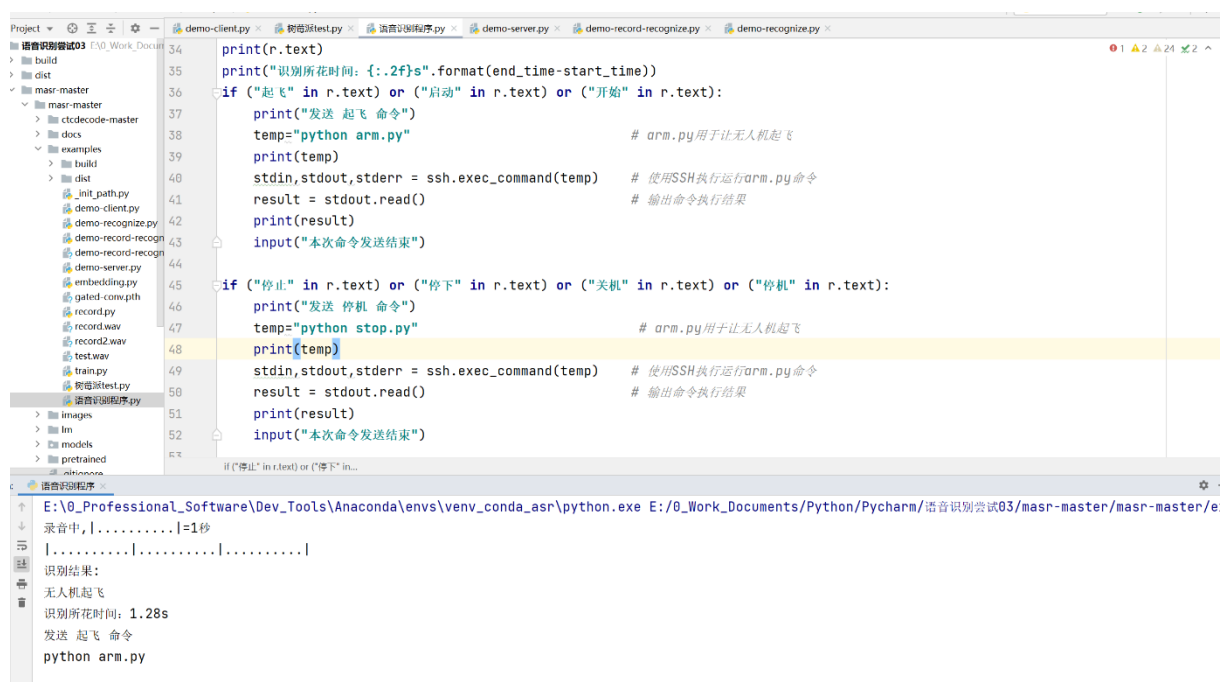


图 5.9 语音控制无人机系统的整体调试效果

5.3 系统应用拓展

本文基于相对成熟的语音识别算法和无人机飞行控制库提出了一种切实可行的语音控制无人机设计方案，可以由用户通过相对口语化的指令进行无人机控制。但由于时间和开发水平的不足，系统仍然存在如识别精度不够高，响应时间较长等问题，现提出以下解决方案。

(1) 在安静的室内，用标准的普通话进行口授指令的使用场景中，识别精度已经足够，但是以上两个条件若是一个不满足，例如在相对嘈杂的环境中或是普通话带有口音的话，识别精度会大大下降，出现诸如把“停止”识别为“停职”的情况。

目前考虑的解决方案主要有两个，进一步加强对识别结果的模糊匹配接受度，把类似“停职”的识别结果同样视为“停止”的命令，但是此举有可能会带来过度模糊匹配的问题，使得用户误操作无人机，如何把握模糊匹配的精细程度是一个相对困难的过程。

此外，还有一个解决方案是提升训练模型的时长，也即给模型更多的标注数据，使其学习情况更好，兼顾泛用性和准确性，但是这个方案除了提升模型训练时长带来的成本问题以外，还有可能带来过拟合的问题，故考虑相对合理的解决方案应该是给模型输入无人机语音控制领域的特殊标注语料，让模型牺牲一定的泛用性，作为一个专业的无人机语音控制模型，但该方案需要重新准备语音数据，主要考虑到时间成本和技术难度，本课题并没有采用，但在理论上这应是一条较为可行的道路。

(2) 系统采用的飞行控制板是 Pixhawk2.4.8，机载电脑为树莓派 3B+，考虑到对无人机语音控制目标的低延时性和高可靠性的要求，可以考虑从硬件上对飞行控制板和机载电脑进行升级，以更优的架构和更大的内存来执行语音控制任务，本课题虽考虑到时间和金钱成本暂未升级硬件，但此方案应是技术难度上较为小，升级可靠性较高的一种思路。

(3) 系统采用的 PC 上位机和机载电脑树莓派的连接方式是基于树莓派的 WIFI 热点进行局域网组网，但是这种组网方式受限于树莓派的功率限制，可靠通信距离并不远，考虑到实际使用语音控制无人机的场景，考虑可使用数传电台代替 WIFI 热点，获得更好的通信稳定性与可靠通信距离。

(4) 系统目前实现的无人机控制方案为通过执行已写好的简单脚本达成无人机控制的目的，这与理想状况中类似操作系统的实时响应还有一定距离，而类似操作系统般的实时响应语音控制才是真正实用化的无人机语音控制方案，这是下一步可提升的空间。

结 论

随着生产生活中越来越多无人机的广泛使用，无人机操作门槛的降低与可靠性的提升正成为一个有强烈实际需求的研究课题，系统提供了一种技术难度相对较低的基于深度学习的嵌入式语音控制无人机设计。

系统设计融合了神经网络模型，语音识别技术，嵌入式开发，无人机二次开发等多个领域的知识，使用具有门控卷积网络结构的开源语音识别项目 **MASR** 作为基本的语音识别算法核心，进行用于无人机飞行控制的语音识别程序开发，结合苍穹四轴开源无人机 **Z410** 作为主要硬件载体，添加机载电脑树莓派，并进行用于用户语音进行飞行控制的飞行控制程序开发，完成了一个基于深度学习的嵌入式语音控制无人机设计。

系统设计完成了以下三个方面的工作，包括硬件设计，软件设计和实验与分析部分。

系统硬件设计主要包括验证语音控制系统的无人机的硬件组成部分，由飞行控制板 **Pixhawk** 和机载电脑树莓派组成的无人机中控系统。本课题使用苍穹四轴开源无人机 **Z410** 作为主要硬件载体，将树莓派作为机载电脑，和开源硬件架构 **Pixhawk** 协同作为无人机的中控系统，协调无人机的各组件，以便于进行无人机二次开发，并验证无人机语音控制系统的可用性与可靠性。

系统软件设计主要包括语音识别程序和基于 **Dronekit** 的飞行控制程序，分别作为用语音来进行无人机系统控制的“语音”部分和“控制”部分。其中详细介绍了语音识别程序的工作流程和环境搭建，飞行控制库 **Dronekit** 及用于飞行控制的部分核心代码原理，展示了软件系统的设计与分析。语音识别算法核心为门控卷积神经网络，以门控单元取代常见的激活函数以获得更好的性能。基于该语音识别算法开发出的语音识别程序功能包括录制用户语音，语音识别，依据识别结果发送对应的无人机控制命令，并具有一定的模糊匹配和冲突指令检查功能。飞行控制程序由 **Dronekit** 编写，在语音识别程序中由 **Paramiko** 库进行自动 **SSH** 登录至机载电脑树莓派，并由其执行对应脚本控制飞控动作，以此完成语音控制无人机的系统目标。

最后通过实验与数据分析，对实验结果进行分析并论证系统设计的可靠性。主要包括系统环境配置和系统各模块的连接通讯测试以及系统整体调试。

在完成无人机语音控制这一设计目标的基础上，系统仍有诸多可以提升之处，同时考虑到实际生产生活的使用情景，系统的设计并不能直接投入使用，而更多作为一种原理性验证的方案，距离实物落地还有一定距离，下一阶段应将这些具体而微的问题解决，使得系统真正具有可以参加生产生活的能力。

参 考 文 献

- [1] 李雪林. 基于人机互动的语音识别技术综述[J]. 电子世界, 2018, 21.
- [2] 于镭, 林再腾. 基于香橙派的智能语音识别系统的设计[J]. 电子测量技术, 2019, 19.
- [3] 高建清, 万根顺, 吴重亮. 端到端语音识别的研究进展与挑战[J]. 中国安防, 2020.
- [4] 禹琳琳. 语音识别技术及应用综述[J]. 现代电子技术, 2013, 36(13): 43-45.
- [5] Amodei D, Anubhai R, Battenberg E, et al. End to end speech recognition in English and Mandarin[J]. 2016.
- [6] 闫晔. 有人机/无人机协同中的交互控制技术研究[D]. 长沙: 国防科学技术大学, 2007.
- [7] 张涛, 芦维宁, 李一鹏. 智能无人机综述[J]. 航空制造技术, 2013, 432(12): 32-35.
- [8] 周楠, 艾剑良. 基于 HMM 和 RNN 的无人机语音控制方案与仿真研究[J]. 系统仿真学报, 2020, 32(03): 464-471.
- [9] Menshchikov A, Ermilov D, Dranitsky I, et al. Data-driven body-machine interface for drone intuitive control through voice and gestures[C]//IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2019, 1: 5602-5609.
- [10] 秦鑫辉. 基于 Raspberry Pi 的语音传输及识别系统的设计与实现[D]. 太原: 山西大学, 2019.
- [11] 谭磊, 余欣洋, 罗伟洋, 曾维, 代云强. 基于深度学习的移动端语音识别系统设计[J]. 单片机与嵌入式系统应用, 2020, 20(09): 28-31+35.
- [12] 杨明翰. 深度学习语音识别系统在嵌入式端的研究[D]. 成都: 成都理工大学, 2019.
- [13] 彭燕子, 柏杰, 曹炳尧, 宋英雄. 基于 Kaldi 的 AI 语音识别在嵌入式系统中的应用研究[J]. 工业控制计算机, 2020, 33(09): 64-67.
- [14] Gowda S M, Rahul D K, Anand A, et al. Artificial Neural Network based Automatic Speech Recognition Engine for Voice Controlled Micro Air Vehicles[C]//2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT). IEEE, 2019: 121-125.
- [15] 侯一民, 周慧琼, 王政一. 深度学习在语音识别中的研究进展综述[J]. 计算机应用研究, 2017, 34(08): 2241-2246.
- [16] Hao Kang, Haoxiang Li, Jianming Zhang, Xin Lu, and Bedrich Benes. Flycam: Multitouch gesture controlled drone gimbal photography. IEEE Robotics and Automation Letters, 3(4): 3717 - 3724, 2018.
- [17] K. Natarajan, T.-H. Nguyen, and M. Mete. Hand gesture controlled drones: An open source library. In 1st International Conference on Data Intelligence and Security, pages 168 - 175. IEEE, 2018.
- [18] Chan W, Jaitly N, Le Q, et al. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition[C]//2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2016: 4960-4964.

- [19] 韩宇, 林志洁, 周旭宝. 无人机飞行控制程序在树莓派上的移植[J]. 浙江科技学院学报, 2016, 28(06): 433-438.
- [20] 张祖航, 曹著明, 薛翼飞. 无人机 APM-PIXHAWK 飞控应用研究[J]. 职业, 2017, 28.
- [21] tonnie. 20 分钟极简入门 Docker[EB/OL]. <https://zhuanlan.zhihu.com/p/86351416>, 2019-10-12.
- [22] 3D Robotics. Introducing DroneKit-Python[EB/OL]. <https://dronekit-python.readthedocs.io/en/latest/about/index.html>, 2016.

修改记录

（1）毕业设计（论文）题目修改

第一次修改记录：

原题目： 基于深度学习嵌入式语音控制无人机设计

修稿后题目： 基于深度学习的嵌入式语音控制无人机设计

（2）毕业设计（论文）内容重要修改记录

第二次修改记录：

修改前：论文第五章存在重大内容偏差，实验内容有了新的进展，论文架构也不完整，文中有许多小型格式错误。

修改后：基本全部完善。

记录人（签字）：

指导教师（签字）：

致 谢

时光匆匆，白驹过隙，又是一年六月，但却是在大工的最后一个六月，指尖敲打的是本科生涯的最后一个课题。毕设从去年 11 月的选题到如今的完成，中间经历了许多波折，但也收获了许多成果，让我有了新的提升和感受，我对自己的毕设感到自豪。

现在回头看来，本论文的选题对比大三时候的我的技术水平而言，是一个较大的挑战，但是毕业论文的过程不仅仅是一个“输出”知识的过程，更是一个结合已有知识，探索未知的“输入”知识过程，现在的我回头看过去一年的自己，感到已有较大的提升。在选题之时，有意选择了相对较难的方向，但是现在看来是高估了自己的知识水平，同时低估了完全陌生的系统上手的速度。

本论文可以视作我自己大二时参加的大创项目“四旋翼无人机主动避障设计”的精神续作，在那时我第一次认真尝试以“开发者”的角度研究无人机，而不仅仅只是以“使用者”的角度将其视为一个大号玩具。也正是由于当时的大创项目指导教师是刘老师，我才有幸在毕业论文选题中继续获得老师的帮助和指点。

本课题的完成，离不开我的指导老师刘老师教授的指点和同组陈学长的提挈，以及同组同学与伴侣 link 同学的支持。从选题时研究思路的教导到系统通信的方案建议，都离不开老师和学长的帮助，link 同学则数次在我怀疑自己时给了我坚持的勇气。

在课题的开始，我对于完成本项目所需的知识储备甚至都没有一个明晰的概念，在刘老师的数次详细论文指导下，在陈学长的几次技术指导下，我渐渐有了对项目的大致概念。在最早的选题阶段，正是在刘老师的引导下，我才能坚定地学习并使用深度学习来进行语音识别，我的本科专业对其并不是很熟悉，使用这样的识别算法对我而言可谓是知识的系统性欠缺。而之后我才意识到，这软件方面的使用仅仅只是冰山一角。此外，在论文撰写方面，刘老师给了我非常细致且深入的帮助，尤其在关于论文的结构安排部分。刚开始甚至有些文不对题，陷入了语音识别理论的细节纠缠之中。好在之后意识到，重新编排整理了结构，解决了这个问题。

除了学业上的指引，刘老师在一次次组会与指导中显露出的“不争”态度也让我对于人生选择有了一些新的感悟，只要自己的技术实力在，不需要过多纠结一些复杂而琐碎的纠葛，做好自己，做自己想要的，做好自己想要的，就是成功的人生。

需要特别感谢的，是我的父母和 link 同学在我一次次面对项目焦躁烦闷之时的耐心陪伴以及对我认真做论文的物质和精神支持，是你们让我有了完成课题的勇气。

最后再次对大学四年来对我有过课程指导的老师说一句，能成为你们的学生，我很开心，感谢老师们的教导。