ISSN: 3065-1220 Volume 1, Number 1, August 2024

Hans汉斯

嵌入式技术与智能系统

Embedded Technology and Intelligent Systems

Qian Ru Shi Ji Shu Yu Zhi Neng Xi Tong

2024年8月1卷1期

https://www.hanspub.org/journal/etis

Editorial Board

编委名单

ISSN: 3065-1220

https://www.hanspub.org/journal/etis

Embedded Technology and Intelligent Systems,Vol.1, No.1, 1-49
嵌入式技术与智能系统, 1 卷 1 期
Published Online August 2024 in Hans (https://www.hanspub.org/journal/etis)

TABLE OF CONTENTS
目录

期刊信息
期刊中文名称：《嵌入式技术与智能系统》
期刊英文名称：Embedded Technology and Intelligent Systems
期刊缩写：**ETIS**
出刊周期：双月刊
语　　种：中文

出版机构：汉斯出版社(Hans Publishers, https://www.hanspub.org/)
编辑单位：《嵌入式技术与智能系统》编辑部
主　　编：何立民，北京航空航天大学教授
网　　址：https://www.hanspub.org/journal/etis

订阅信息
订阅邮箱：sub@hanspub.org
订阅价格：180 美元每年

广告服务
联系邮箱：adv@hanspub.org

创刊词

《嵌入式技术与智能系统》终于创刊了，它是《单片机与嵌入式系统应用》转刊后，成为推动我国人工智能新时期嵌入式系统领域发展的一件大事。

众所周知，上世纪，30 年代，图灵与其同时代的学者们提出了“万物皆可为函数”、万物皆可计算的“可计算原理”，并推出了“图灵机模型”，从而，在理论上奠定了人工智能的计算机软硬件基础。“一切皆可计算”，从数值计算到逻辑计算，开创了人类智力计算的探索征程。50 年代，一批年轻学者确立了智力计算的人工智能学科地位；70 年代，带有“图灵机模型”灵魂的“微处理器”诞生，人工智能终于从理论研究进入到一个嵌入式智能感知、控制与人类智力仿真的实践应用时代。

上世纪 70 年代初，微处理器诞生，随后迅速形成中央微处理器(CPU)与微控制器(MCU)两大分支。前者用以构成通用计算机，后者用来构式嵌入式计算机。1972 年，中央处理器 8008 诞生，1981 年，基于 8080 的通用计算机 IBM-PC 诞生；1976 年、1980 年则分别诞生了 MCS-48 与 MCS-51 微控制器。由此拉开了人工智能领域两大分支的发展道路。微控制器基础上的嵌入式系统突出其感知与控制能力，实现电子系统的智能化控制；通用计算机系统则在高速海量计算基础上，实现人类的智力仿真。

物联网诞生以前，通用计算机与嵌入式系统形成了人工智能中并行发展的两大领域。通用计算机沿着“深蓝”、“沃森”、“Alpha go”、“ChatGPT”一路逛奔；嵌入式系统则从早期传统电子系统的智能化改造，逐步深化到物联网时代的智慧机器人、工业互联网、分布式智能系统、无人工厂、无人驾驶系统等人工智能领域。未来，人工智能探索中，两大领域的交叉融合是一大趋势。在众多人工智能的落地应用中，还必须依靠嵌入式系统的智能感知与控制技术。

当前，人工智能已进入深水区，无论是万物互联的智能系统，还是人工智能大模型的落地应用，对嵌入式系统软硬件技术提出了更高的要求。嵌入式系统硬件在 MCU 基础上向 AI 芯片进军；嵌入式系统软件在物联网操作系统、集成开发环境、智能应用软件等领域迅速崛起。嵌入式系统在提升了计算能力后，极大地满足了智能系统的边缘计算及机器学习等高算力计算要求，而在其原有的感觉与控制领域，也迅速向物理对象识别、机器人、无人驾驶系统等高端人工智能领域进军。

值得骄傲的是，人工智能领域，中国人早在上世纪 80 年代初就及时引进了微控制器与嵌入式技术，并形成了大规模的应用高潮，1987 年“全国单片机学会”正式成立，全国工科院校相关专业、众多国际半导体知名企业，电子技术、计算机界专家、学者、工程师等齐聚在学会周围。2020 年，由学会发起，国内嵌入式系统专业期刊《单片机与嵌入式系统应用》问世了，以及其后“嵌入式系统联谊会”的成立，形成了我国嵌入式应用推广的核心力量。

2023 年，随着我国在集成电路领域面临的巨大挑战，《单片机与嵌入式系统应用》转刊为《集成电路与嵌入式系统》。转刊后，为了继承和发扬我国嵌入式系统领域 20 多年来得之不易的成果，原有的编委会团队另辟蹊径，在汉斯中文开源期刊学术交流平台上创办了这本《嵌入式技术与智能系统》开源期刊。它是一本关注传统嵌入式技术与新兴智能系统前沿技术最新进展的国际中文期刊。不仅深入探讨嵌入式技术，还广泛涵盖嵌入式系统在多个行业领域(如人工智能、物联网和智能工业等)的创新应用，旨在吸引对这些领域有深厚兴趣的作者和读者群体。杂志不仅关注传统嵌入式技术的应用，如与人工智能、物联网、机器人、汽车各行业的结合的新技术推广应用，还致力于发表人工智能、大数据、集成电路等前沿学术领域的创新算法和工程实现方法的论文。

新期刊依然会遵循“专家办刊”的指导方针。因为我们有强大的嵌入式专家团队，“嵌入式系统联谊会”是我们的坚强后盾。相信依靠这些队伍，我们依然能延续过去的辉煌，为我国智能系统领域做出更大的贡献。在此，也一并感谢为本刊创刊中努力支持与帮助我们的专家、学者与同仁们，再次谢谢！！

何立民 2024 年 7 月

Embedded Technology and Intelligent Systems 嵌入式技术与智能系统,2024, 1(1), 1-10
Published Online August 2024 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2024.11001

Hans汉斯

嵌入式软件开发的三个趋势

何小庆[1]，何灵渊[2]

[1]嵌入式系统联谊会，北京
[2]博通公司，森尼韦尔美国

收稿日期：2024年7月4日；录用日期：2024年7月25日；发布日期：2024年8月5日

摘要

智能时代的嵌入式系统离不开高性能、高效的软件和先进的软件开发方式。本文介绍了近期嵌入式软件开发的三个趋势：1)边缘计算作为一种在本地处理和分析数据的方式正在快速发展，边缘计算与人工智能的结合正将智能计算从以云为中心的模型中转移出来；2) 虚拟化技术是今天高算力多核处理器计算系统采用的全新解决方案，容器技术则可通过简化嵌入式软件开发、部署和维护来助力复杂嵌入式系统的管理；3) DevOps的概念和实践正在逐步渗透进入嵌入式软件开发中，助力加快软件交付速度，提高应用程序质量和稳定性。

关键词

嵌入式系统，边缘计算，人工智能，虚拟化，软件开发

ThreeTrends in Embedded Software Development

Xiaoqing He[1], Lingyuan He[2]

[1]Embedded System Association, Beijing
[2]Broadcom Inc., Sunnyvale, USA

Received: Jul. 4th, 2024; accepted: Jul.25th, 2024; published: Aug.5th, 2024

Abstract

High-performance efficient software and advanced software development techniques are keys to embedded systems in the era of intelligence. This paper describes three current trends in embedded software development: 1) Edge Computing is progressing quickly as a method to process and

文章引用:何小庆, 何灵渊.嵌入式软件开发的三个趋势[J].嵌入式技术与智能系统,2024,1(1):1-10.
DOI: 10.12677/etis.2024.11001

analyze data locally. Edge Artificial Intelligence is changing the cloud-central method of intelligent computing; 2) Virtualization is the new solution for high-performance multi-core processor systems. The container technology is streamlining the management of complex embedded systems by simplifying software development, deployment, and maintenance; 3) The concept and practice DevOps is breaking ground in embedded software development. It speeds up delivery, elevates quality, and promotes stability throughout the development process.

Keywords

Embedded System, Edge Computing, Artificial Intelligence, Virtualization, Software Development

1. 产业背景

2024 年 2 月 IDC 发布的数据显示，2023 年中国物联网(IoT)连接量超 66 亿个，未来 5 年复合增长率约 16.4%[1]。随着物联网设备数量的增加和应用场景的扩展，嵌入式系统在连接和管理大规模物联网设备方面将发挥更加重要的作用。嵌入式系统需要能够处理大规模的数据，并提供安全、可靠的连接和通信功能，以支持各种物联网应用，包括智能城市、工业自动化、智能健康等。

在物联网发展过程中，边缘计算正在快速兴起。随着对实时性和隐私性要求的加强，边缘计算作为一种在本地处理和分析数据的方式变得越来越受欢迎。边缘计算能够实现快速响应和即时决策，同时减少对云端资源的依赖。嵌入式系统在边缘设备上正扮演着重要角色，边缘计算的发展趋势要求嵌入式系统的性能和能效不断提升，以满足边缘计算场景下的需求。

随着人工智能(AI)和机器学习(ML)在各个领域的应用不断深入，物联网和边缘计算的普及，嵌入式系统将有能力更加普遍地整合人工智能技术。这种能力包括在嵌入式设备上实现复杂的 AI 算法和模型推理，以提供更智能、更自适应的功能和服务，例如在自动驾驶汽车、智能机器人和医疗设备等领域应用。

2. 近期市场研究

关于嵌入式软件开发趋势的研究和讨论一直在积极进展中，VDC Research 的多位分析师撰写了一份关于 2024 年物联网、嵌入式和工业技术预测的报告，报告概述了影响硬件和软件市场的关键主题和趋势，并对未来的技术发展做出了一系列预测[2]。报告中预测 AI 正在从云端数据中心稳步迁移到网络边缘，包括嵌入式设备。嵌入式和边缘 AI 市场对特定行业应用部署的优化需求日益增长，这种趋势推动边缘 AI 硬件变得更加面向应用。报告还预测云原生(Cloud Native)开发解决方案未来将得到普及，新冠疫情期间云原生软件开发解决方案取得了巨大的成功，嵌入式工程社区正迅速适应这一开发模式，逐渐摆脱传统的本地开发方式。

知名的嵌入式软件专家和培训师，Beningo Embedded Group 的创始人 Jacob Beningo，在 2024 年 1 月嵌入式软件的五个趋势的文章中指出，嵌入式软件团队正在快速采用 DevOps。它为团队提供了自动化构建、测试和部署过程的方法[3]。Integrated Computer Solutions (ICS)的 CEO Peter Winston 在 2024 年 3 月一篇博客文章中指出：未来六大趋势对嵌入式开发将产生深远影响，它们是网络安全要求、设备的共

存和互操作性、新芯片架构、连接技术、AI/ML 集成，和面向服务的架构。这些趋势肯定会在短期内迅速重塑产品开发过程，并在未来持续产生影响[4]。

综合以上的观点，在本文中我们将重点讨论 AI/ML 集成与边缘智能，虚拟化/容器和混合部署，以及 CI/CD 和 DevOps 这三大嵌入式软件开发趋势。

3. 趋势之一：AI/ML 集成与边缘智能

3.1. 边缘计算与边缘 AI

微软对边缘计算的定义是：边缘计算是一种分布式计算框架，允许 IoT 设备快速处理网络边缘的数据并对其采取行动[5]。微软给出了一个边缘计算示例：远程仓库中的安全相机使用 AI 识别可疑活动，并且仅将该特定数据发送到主数据中心进行即时处理。相机不会不断传输拍摄的全部视频片段，而是只发送相关的片段，避免每天 24 小时传输给网络带来负担，从而释放公司的网络带宽和计算处理资源用于其他用途。

边缘 AI 正在将智能计算从以云为中心的模型中转移出来，并使其更接近数据源。推动这一转变的动机是减少网络流量，允许时间关键型应用(例如制造、自主系统)进行实时决策，以及通过本地处理数据来加强隐私性。

边缘 AI 减少了对超大规模 AI 提供商的依赖，并促进了更广泛的 AI 应用。它在医疗保健、汽车和机器人技术领域具有推动变革的潜力，能够重塑这些行业的运营范式。展望未来，AI 将对不同类型的边缘产生不同的影响。不同类型的边缘计算如图 1 所示[6]。

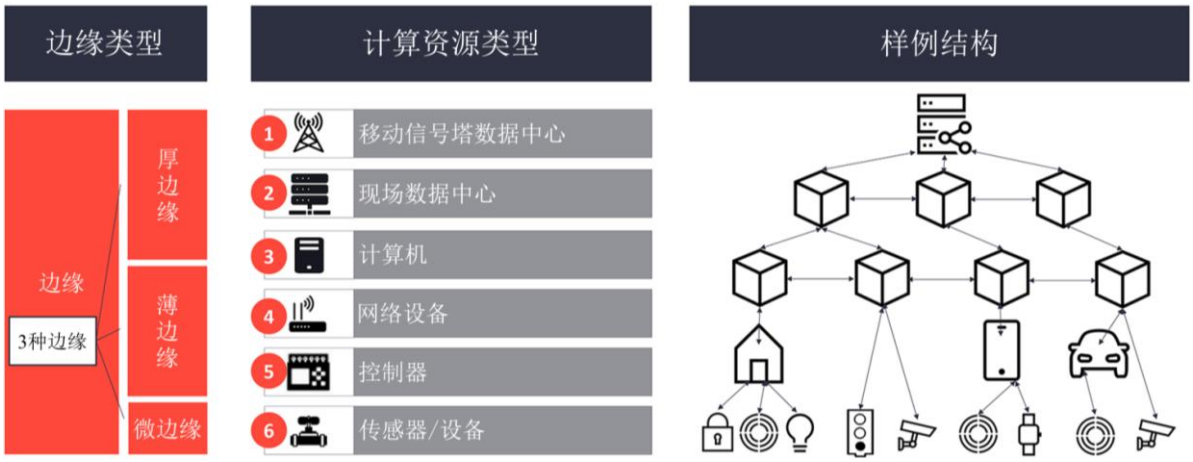

Figure 1. Three types of edges and common edge devices
图 1.三种类型的边缘和常见的相关设备

厚边缘(Thick Edge)是指配备了高性能计算部件(例如高端中央或图形处理单元)的计算资源，通常位于数据中心，旨在处理计算密集型任务/工作负载，例如数据存储和分析。薄边缘(Thin Edge)是指智能控制器、网络设备和计算机，它们聚合了来自传感器的数据。微边缘(Micro Edge)是指生成数据的智能传感器和设备。

3.2. 边缘 AI 的细分

边缘 AI 是在边缘计算的设备上部署 AI 模型，从而在不依赖云连接的情况下实现 AI 推理和决策。边缘 AI 也分为厚边缘 AI、薄边缘 AI 和微边缘 AI。

厚边缘 AI 的定位是在边缘服务器支持执行多个 AI 推理模型，包括针对本地敏感数据场景的 AI 模型训练或再训练。薄边缘 AI 则是利用网关、IPC(工业计算机)和 PLC(可编程逻辑控制器)在网络边缘进行 AI 处理，增强现有传感器和设备的智能化能力。微边缘 AI 支持将 AI 直接集成到传感器中，提高智能系统的可扩展性，并使日常设备能够做出自主决策。

在微边缘，微型机器学习(Tiny Machine Learning, TinyML)是重要的技术。这个技术让在 MCU 等资源受限(如内存、计算和能耗限制)的小芯片上运行机器学习模型和算法成为可能。区别于现在流行的大模型，TinyML 的尺寸远远小于一般的机器学习模型，和大模型更是相差甚远。现在很多的物联网设备上都已经用上了 TinyML 这项技术，例如智能家居(恒温器和智能灯光系统)和健康监测(心率和睡眠质量)等[7]。

在 Embedded World 2024 展会上 TinyML 基金会展区展示的一个案例中，TinyML 被集成到日常物品和工具中，使他们能够自主执行决策功能，而无需云连接(如图 2 所示)。这种方法能增强隐私性和数据安全性。

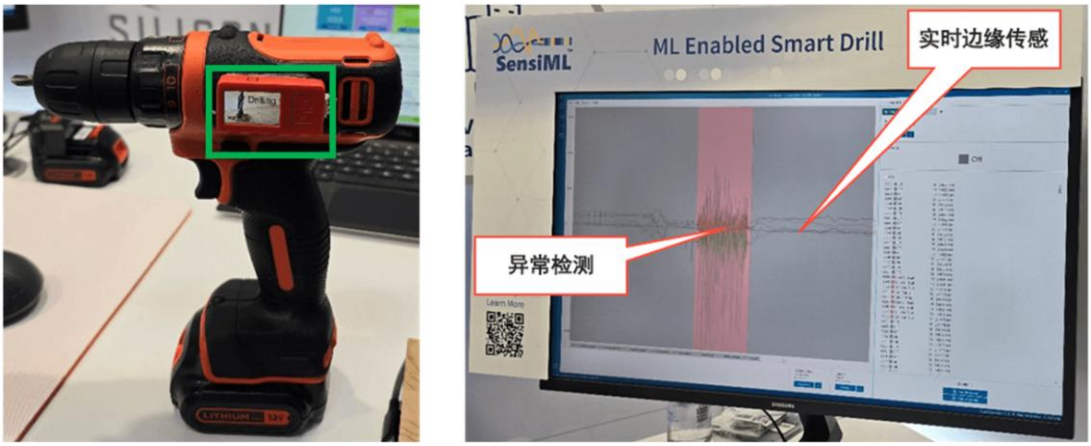

Figure 2. An electric drill and the TinyML development tool in an ESP32-based edge sensing IoT kit
图 2.基于 ESP32、用于边缘传感的物联网开发套件中的电钻和 TinyML 开发工具

3.3. 边缘 AI 的三个趋势

今天边缘 AI 呈现出三大发展趋势：厚边缘 AI 训练、薄边缘和微边缘 NPU(神经处理单元)加速，以及微边缘 TinyAI 赋能。第一个趋势是 AI 模型训练正在从云转移到数据中心或微服务器等厚边缘位置。从技术上看这是可能的，因为边缘微服务器集成了高性能 CPU 和 GPU，可在边缘实现强大的计算，包括 AI 训练和多种 AI 推理功能。人工智能训练也可以在地端数据中心进行，这样做有四大好处：减少对云基础设施的依赖，降低成本，增强隐私性，以及提高人工智能应用程序在边缘设备上的响应能力。

在边缘设备中集成专用 NPU 可大大增强 AI 推理能力。NPU 可以降低功耗，改进能耗管理，并助推高效的多任务处理，这使 AI 能够在功耗敏感和延迟的关键型应用(如可穿戴设备和传感器节点)中部署。Arm 的数据表明，用于 AI 推理的 ARM Cortex A55 配置和 Arm Cortex A55 + Arm Ethos U65 NPU 比较，后者将 70%的 AI 推理从 CPU 转移到 NPU，推理性能提高了 11 倍。

微边缘和薄边缘 AI 实现了自主决策的本地化。将支持 AI 的芯片组直接集成到蜂窝物联网设备中的趋势正在兴起，这标志着物联网设备正在向能够进行本地化决策的智能、自主物联网系统转变。这一趋势预计会对智慧城市和工厂等行业产生重大影响，并带来显著优势，包括实时数据处理、减少延迟以及通过更小的外形尺寸提高效率。

嵌入式开发者对人工智能的开发非常重视。2023 年 Embedded.com 嵌入式开发现状调查报告显示，

针对在嵌入式开发中利用先进的技术能力这一问题，嵌入式人工智能和机器学习最受关注，紧随其后的是嵌入式视觉和语音功能(如图 3 所示) [8]。

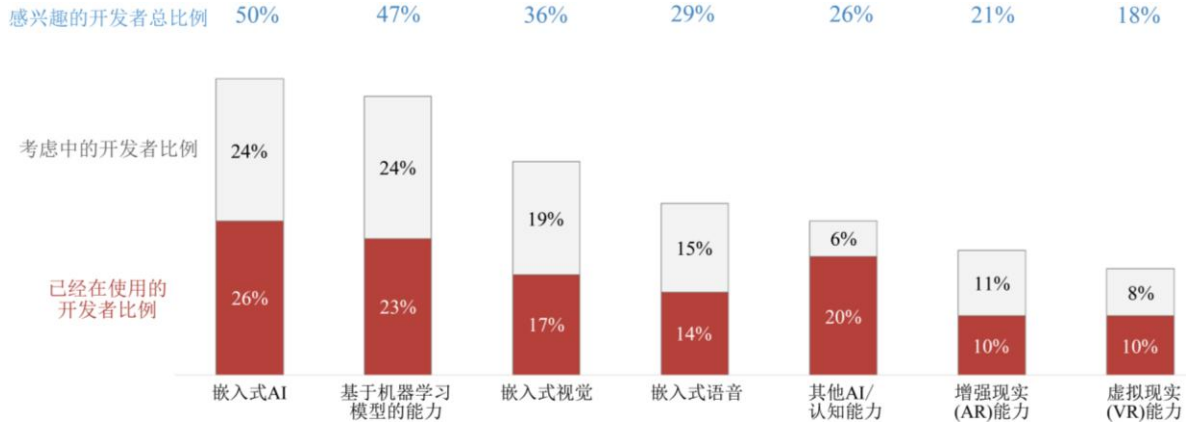

Figure 3. Using advanced technologies in embedded system development
图 3.在嵌入式开发中利用先进的技术能力

4. 趋势之二：虚拟化、容器和混合部署

4.1. 虚拟化

虚拟化技术是操作系统的一个重要技术，在嵌入式系统中呈现蓬勃发展的趋势。借助于对底层处理器内核、内存和外设的抽象，这种技术使得多个虚拟机可以运行在同一个物理处理器上。虚拟化提供了多操作系统的运行环境，例如可以在同一个设备中同时运行高实时性操作系统(如 FreeRTOS)和通用的操作系统(如 Linux)。虚拟化技术是今天高算力多核处理器计算系统采用的全新解决方案，它平衡了通用性与可靠性两方面需求。嵌入式虚拟化的典型应用在智能汽车电子系统中，比如在一个智能座舱系统中同时运行高实时性操作系统和人机交互操作系统[9]。

智能工业场景下的混合关键系统应用可以借助多核处理器系统以及虚拟化技术部署，一个国产的虚拟化操作系统解决方案见图 4。该方案基于 Intewell 实时操作系统，是一个针对数控系统、面向低成本硬件的可配置混合异构系统解决方案[10]。

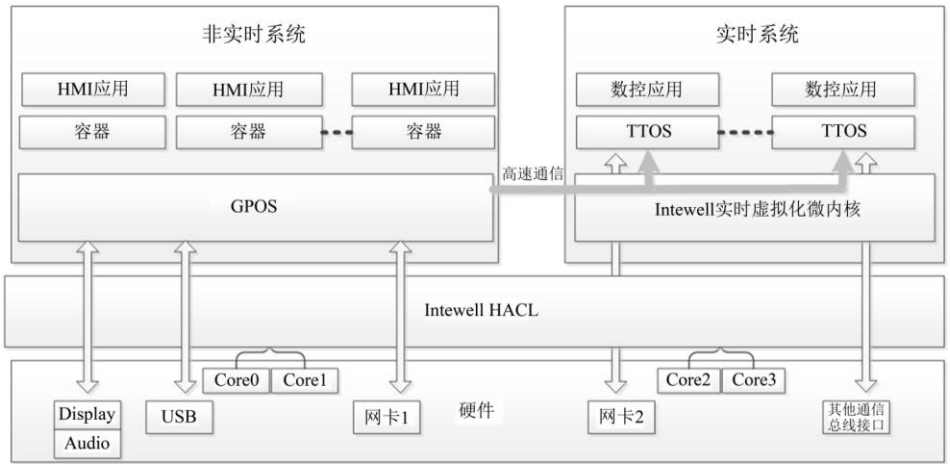

Figure 4. Intewell configurable hybrid heterogeneous system architecture
图 4. Intewell 可配置混合异构系统体系结构

4.2. 容器

容器是一种轻量级、可移植、隔离的软件环境。通过容器技术，开发人员可以将应用程序及其依赖项打包在一起，确保应用在不同平台上的一致性运行，简化应用开发、部署和管理流程。容器和虚拟机具有相似的资源隔离和分配功能，容器虚拟化软件环境，而虚拟化则是虚拟化硬件。

编排器在容器技术中起到重要的作用，像 Kubernetes 这样的编排器旨在自动执行容器化应用程序的部署、扩展和管理。Docker 是 Linux 环境下最常见的容器平台，随着云计算和大数据的广泛应用，一部分虚拟化正在被轻量级的容器技术替代，编排器的作用也变得越来越重要。有关于编排器性能和能耗的研究受到关注，例如郑忠斌等提出利用线性规划在提高资源利用率的同时降低碳足迹[11]。

4.3. 实时容器及其应用

实时容器是基于实时操作系统(RTOS)实现的容器机制，贴近 Docker 用户习惯，符合开放容器规范(OCI—Open Container Initiative)，实现轻量化、安全和便捷的容器部署。

目前，国内针对实时容器对外公开的研究较少，比较有代表性的有翼辉基于 SylixOS 开发的实时容器 ECS(Edge Container Stack)[12]。ECS 是轻量级实时容器技术，使用 ECS 实时容器的 SylixOS 内核依然有着与标准版本 SylixOS 相当的性能。ECS 实时容器提供给每个容器隔离的运行环境。完整独立的容器运行环境保证了容器内的应用免受环境和其它应用的影响，使容器的运行环境符合预期，提升容器内应用的安全性。

邓广宏等指出，受限于开发时间以及软件生态支持，ECS 实时容器尚不成熟，难以应对构建大型容器云的需求[13]。国外对于实时容器的研究相对丰富，大多基于实时 Linux 补丁与 Docker 的组合实现实时容器。NXP 的专家[14]重点研究了资源和成本受限的 MCU/MPU 领域的容器解决方案，提出了借鉴 Android 容器技术的一种解决方案——MICROEJ VEE，该方案可用于运动手表、智能洗衣机、智能电表和工业打印机等场景，其软件架构如图 5 所示。

Figure 5.A small real-time container solution: MICROEJ VEE
图 5.一种小型实时容器方案：MICROEJ VEE

总之，容器技术可通过简化嵌入式软件开发、部署和维护来助力复杂嵌入式系统的管理。开发者在使用容器时需要注意平衡容器的优势和容器之间通信带来的复杂性和性能损失。

4.4. 混合关键系统部署

什么是混合关键系统(Mixed Criticality Systems，简称 MCS)？英国约克大学的专家给出了下面的定义：混合关键系统是指包含两个或更多不同关键性级别的组件的系统[15]，例如安全关键、任务关键和非关键组件。这些系统通常存在于汽车和航空电子领域中的复杂嵌入式系统中，它们正在向 MCS 发展，以满足与成本、空间、重量和功耗等非功能需求相关的严格标准。

混合关键系统的研究和工程实践旨在通过部署、隔离和调度等技术手段，实现多系统混合部署，并实现系统间彼此隔离保护，通过调度提升资源利用率。具体在技术层面，学术界侧重调度方法研究，西北工业大学的一项研究中作者提出了一种基于异构多核系统的混合关键任务调度算法[16]。工业界更多关注时空隔离下的混合关键性部署，保证各子系统间互相隔离，资源采用静态分配，因此整体资源利用率不高。为了解决这一问题，华为提出基于 openEuler 的混合关键性系统解决方案[17]，其架构如图 6 所示。

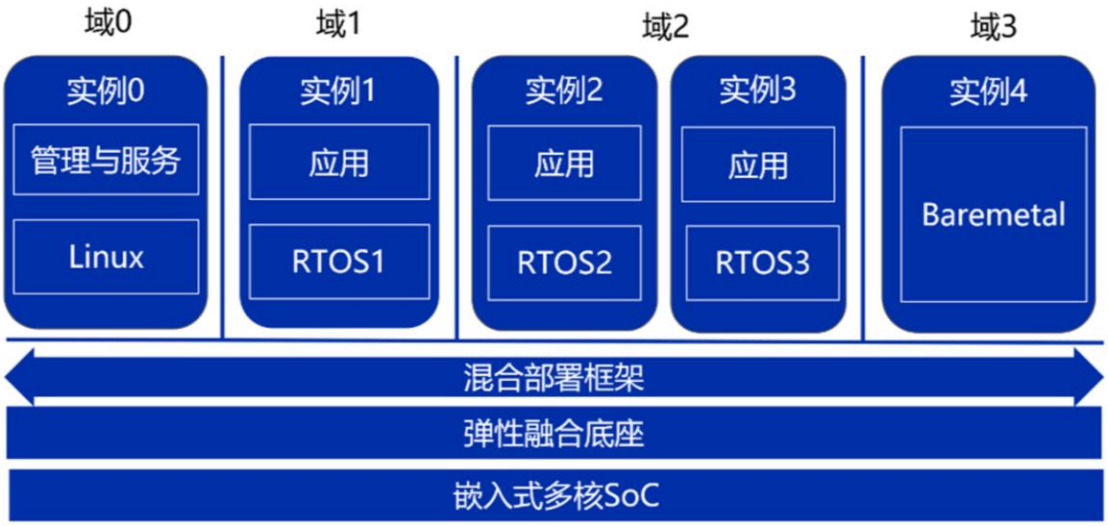

Figure 6.A mixed criticality system architecture
图 6.混合关键性系统架构

混合关键系统的实现可以依托弹性融合底座，底座的核心是容器和虚拟化技术。该解决方案中推荐使用轻量级容器 isula，嵌入式虚拟化推荐使用 ZVM 和 Jailhouse。ZVM(Zephyr-based Virtual Machine)是基于 Zephyr 实时操作系统实现虚拟化功能的开源项目，该项目是由湖南大学嵌入式与网络计算湖南省重点实验室主导，目前可以实现同时启动一个 Linux 与多个 Zephyr RTOS，从而在同一硬件平台上实现混合内核部署[18]。混合关键系统目前主要应用在智能制造、机器人、能源、军工和航空航天等强实时和高安全领域。

5. 趋势之三：CI/CD 和 DevOps

5.1. 概述

DevOps 是一种软件开发实践，可促进开发与运维之间的协作，从而更快、更可靠地交付软件。DevOps 通常被理解一种文化，将开发者、流程和方法连接在一起来提供持续价值[19]。DevOps 在 IT 产业被广泛地使用，其具体做法是 CI/CD(持续集成/持续交付)，这是一种通过在应用开发阶段引入自动化技术快速迭代向客户交付应用的方法。与 CI/CD 关联的步骤通常被统称为 CI/CD 管道(Pipeline)，由开发和运维团队以敏捷开发方式协同完成。

随着云计算、容器、微服务等技术的蓬勃发展，云原生的概念已被市场普遍接受。云原生技术是DevOps 实践的核心组成部分，通过使用云原生工具和流程，可以实现自动化的应用程序开发、测试和部署，从而加快软件交付速度，提高应用程序质量和稳定性。DevOps 一站式平台则将云原生开发模式融合到产品中，为广大开发者提供好用易用的云原生研发管理解决方案，包含敏捷项目管理、代码管理、自动化测试管理，CI/CD 流水线等功能，让开发、测试、部署全流程与云原生底座平台无缝结合，降低嵌入式开发者上手的门槛，加快 CI/CD 和 DevOps 云原生技术在嵌入式开发中的落地。

5.2. CI/CD 和 DevOps 在嵌入式开发中的实践

苏黎世应用科技大学嵌入式系统学院(InES)在工业协议认证测试中，采用基于云原生的 CI/CD 管道技术的自动化编排方法。具体步骤有以下几个：测试用例与测试环境与 Azure DevOps CI/CD 结合，Pipelines(编排)的集成，PROFINET(工业以太网)的概念验证，以及创建、验证单元测试环境。最后的结论是这种方法改变了传统的 V 模型(验证模型)开发流程，即从瀑布式转变到敏捷开发，加快开发周期，提高效率[20]。

西门子对当前汽车电子产业的软件工程开发做了详尽分析，西门子的研究员指出车厂 OEM 和 OEM 的许多一级供应商正在采用软件工厂的开发方法，比如从传统的汽车开发中分离出来的大众集团软件公司 CARIAD 和捷豹路虎的爱尔兰香农和英国曼彻斯特软件开发团队[21]。汽车软件开发正在拥抱 CI/CD，支持多个车型和平台上整个生命周期的持续的软件更新。典型的 CI/CD 开发过程如图 7 所示。

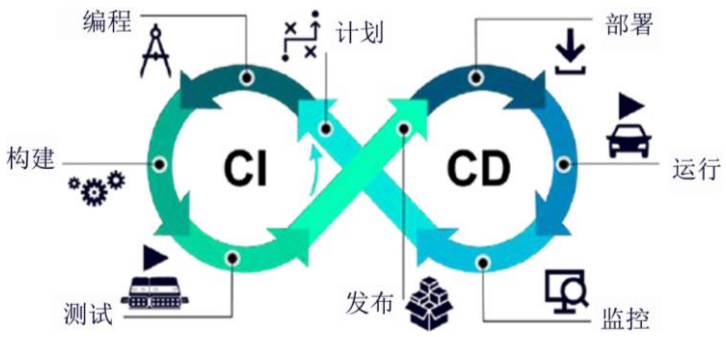

Figure 7.A typical CI/CD development process
图 7.典型的 CI/CD 开发过程

5.3. 嵌入式 CI/CD 和 DevOps 工具

传统的嵌入式软件开发工具是一个集成开发环境，将编辑、编译、构建和调试合成在一个以工程包为核心的开发环境中，具有入门门槛低、集成度高以及针对性强等特点，缺点也很明显：跨平台、升级和第三方集成不方便。典型的传统开发工具如 IAR Embedded Workbench。

过去十年来，在高端微控制器中运行的软件的复杂性显着增加，比如实时操作系统(RTOS)和通信堆栈已经取得了广泛应用。随着物联网的普及，附加软件层通信协议(MQTT)和高级安全性管理(例如基于云的设备管理)在物联网应用中是必需的。同时，GUI 库、DSP 库、机器学习软件框架等中间件也增加了手动集成软件组件的复杂性。

为了改善嵌入式开发过程，Arm 开发了 CMSIS-Toolbox 实用程序，它是一个开源项目，也是 Open-CMSIS-Pack 的一部分[22]。Arm 于 2014 年推出了 CMSIS-Pack，它是 CMSIS(通用微控制器软件接口标准)版本 4 的一部分。CMSIS-Toolbox 支持使用不同工具和不同工作流程，帮助嵌入式软件开发人员更轻松地协作，如图 8 所示。

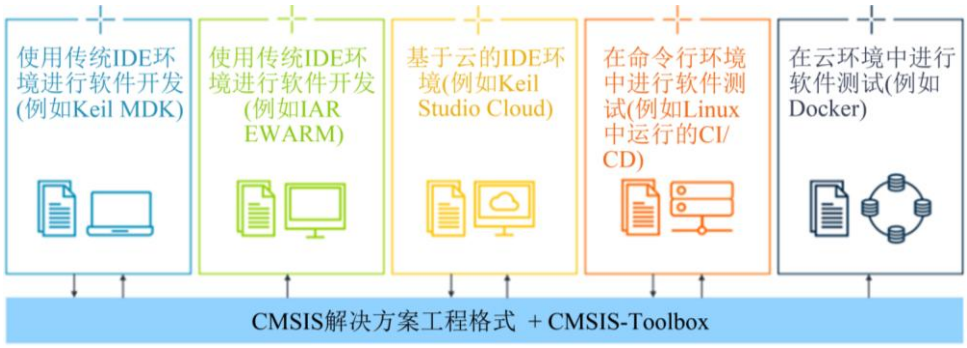

Figure 8. An open workflow involving CMSIS-Toolbox
图 8.CMSIS-Toolbox 开放的工作流程

市面上已经出现了整合云和 CI/CD 的嵌入式开发平台，如 Wind River Studio(下文中简称为 Studio)。Studio 是一个云原生工具集，用于开发、部署、运营和服务关键任务智能系统[23]。它为开发嵌入式设备软件平台和应用程序而构建，包括交叉编译系统和嵌入式设备测试系统，可查看端到端工作流程的状态，以便诊断定制化流水线中的复杂问题。Studio 在基于云的基础设施上提供一套高度集成的工具，实现嵌入式软件开发的自动化，并针对平台开发人员、应用程序开发人员和测试人员实现自定义的 CI/CD 流程。

Wind River(风河)是全球知名的嵌入式软件企业，产品包括边缘设备的实时操作系统、实时 Linux 和虚拟化技术，以及边缘开发和云计算解决方案，应用覆盖航空、航天、工业、汽车和通信。风河在嵌入式系统耕耘数十年，是嵌入式操作系统业界的常青树[24]。

6. 结语

在智能系统时代，嵌入式系统重要挑战来自软件开发。过去 40 年，集成电路在摩尔定律推动下快速发展，遗憾的是软件发展远远滞后了。当前，嵌入式软件开发面临巨大挑战。以自动驾驶为代表的智能应用，一方面必须以高安全等级为基础，另一方面，由于高算力多核芯片的超速发展，基础软件需不断更新以适配新型异构多核、包含 AI/GPU 等不同芯片的处理器体系架构。此外，微内核操作系统发展提速，混合部署应用需要虚拟化和容器技术支撑[25]。

展望未来，业界需要研究嵌入式软件开发趋势，把握电子信息产业最新需求，梳理软件工程实践经验，改进嵌入式软件开发方法，勇于采用新技术新平台，从而迎接人工智能时代嵌入式软件的新机遇。

参考文献

[1] IDC. 5G 引领物联网连接增长,IDC 发布中国物联网连接量预测[EB/OL]. https://www.idc.com/getdoc.jsp?containerId=prCHC51842824,2023-02-02.

[2] Rommel,C.,Hoffenberg,S.,Mandell, D.,*et al*. (2024) 2024 IoT, Embedded & IndustrialTechnology Predictions. https://www.vdcresearch.com/vdcc/wp-content/uploads/2024/01/2024-IoT-Embedded-Industrial-Tech-VDC.pdf

[3] Beningo,J. (2024) 5 Embedded Software Trends to Watch in 2024. https://www.designnews.com/embedded-systems/5-embedded-software-trends-to-watch-in-2024

[4] 微软公司. 什么是边缘计算? [EB/OL]. https://azure.microsoft.com/zh-cn/resources/cloud-computing-dictionary/what-is-edge-computing/,2024-01-03.

[5] Winston,P. (2024) 6 Embedded Software Development Trends to Watch in 2024. https://www.ics.com/blog/6-embedded-software-development-trends-watch-2024

[6] IoT Business News (2024)The Top 6 Edge AI Trends—As Showcased at Embedded World 2024. https://iotbusinessnews.com/2024/04/30/34354-the-top-6-edge-ai-trends-as-showcased-at-embedded-world-2024

[7] Lin,J.,Zhu,L.G., Chen, W.-M.,*et al*. (2024) Tiny Machine Learning Projects. https://hanlab.mit.edu/projects/tinyml

[8] Embedded.com (2023)The Current State ofEmbedded Development. https://www.embedded.com/wp-content/uploads/2023/05/Embedded-Market-Study-For-Webinar-Recording-April-2023.pdf

[9] 何小庆. 嵌入式实时操作系统的昨天、今天和明天[J].中国计算机学会通讯, 2023(2):80-85.

[10] 郭建川,殷灿菊. 面向数控机床异构系统架构设计的操作系统[J].单片机与嵌入式系统应用,2022(3):8-10.

[11] 郑忠斌,李世强,费海平. 一种基于 Kubernetes 的工业物联网的新型调度[J].单片机与嵌入式系统应用,2021,21(6):8-10.

[12] 翼辉信息. ECS 实时容器[EB/OL]. https://www.acoinfo.com/product/5328/?category=42&subCategory=7350&curCategory=5328,2022-05-07.

[13] 邓广宏,张棋恒. 基于混合关键系统的容器调度架构设计[J].计算机科学, 2023,50(z1):901-905.

[14] Patel,M.(2024) Containers for Cost Optimized MCUs and MPUs: A Game-Changer for Embedded Systems. https://www.linkedin.com/posts/maulin-patel-6a927a9_containers-for-cost-optimized-mcus-and-mpus-activity-7193335080805715971-BeMp

[15] Burns,A. andDavis, R.I. (2017)A Survey of Research into Mixed Criticality Systems.*ACM Computing Surveys*(*CSUR*),**50**, 1-37.https://doi.org/10.1145/3131347

[16] 赵瑞姣,朱怡安,李联. 基于异构多核系统的混合关键任务调度算法[J]. 计算机工程,2018,44(2):51-55.

[17] 余德钊. 实时内核 UniProton 及其混合关键性部署的实践[EB/OL]. https://www.esbf.org/wp-content/uploads/2023/08/202308_YDZ.pdf,2023-08.

[18] openEuler. openEuler 开源新项目,嵌入式实时虚拟机 ZVM 介绍[EB/OL]. https://www.openeuler.org/zh/blog/20230325-ZVM/20230325-ZVM.html,2023-03-23.

[19] 微软公司. DevOps 教程——简介[EB/OL]. https://azure.microsoft.com/zh-cn/solutions/devops/tutorial/,2024-07-01.

[20] Lone,O.,Stasiak,T.,Meisterhan,J.,*et al*. (2024) The Future of Industrial Protocol Certification Testing:CI/CD Pipelines, Cloud-Based, Automated and Orchestrated. *Embedded World Conference*, Nuremberg,9-11 April2024, 399-402.

[21] Morris,B.(2024) What Is Coming Next with Software Defined Vehicles? An Examination of the Trends Predicted over the Coming Years.*Embedded World Conference*,Nuremberg,9-11 April2024,304-309.

[22] Yiu,J.(2024) Simplifying the Integration of Software Components in Modern Microcontroller Systems.*Embedded World Conference*,Nuremberg,9-11 April2024,29-35.

[23] Wind River. Wind River Studio. https://www.windriver.com/studio

[24] 何小庆. 嵌入式操作系统风云录——历史演进与物联网未来[M].北京:机械工业出版社, 2016: 18-19.

[25] 何小庆. AIoT 时代的嵌入式技术与人才培养[J]. 单片机与嵌入式系统应用,2020,20(9):6.

Embedded Technology and Intelligent Systems 嵌入式技术与智能系统,2024, 1(1), 11-22
Published Online August 2024 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2024.11002

Hans汉斯

面向独居老人的智能居家监护系统

董敏，谭皓禹，杨礼铭，沈煜，陈章韶，毕盛

华南理工大学计算机科学与工程学院，广东广州

收稿日期：2024年7月4日；录用日期：2024年7月25日；发布日期：2024年8月5日

摘要

随着人口老龄化程度加深，社会养老负担加重，处理好全社会的养老问题十分重要。在全球范围内，老人身体健康受到许多致命疾病的威胁。而独居老人生活中缺少家人照顾，心理上缺少慰藉，导致患病率更高，同时发生意外也无法及时得到救助。面临精神、健康、意外风险三重困境，因此，他们具有更大的健康风险。本文提出的系统基于云－边－端架构实现，由云端服务器、感知控制和应用服务构成，实现了语音服务和老人应急服务，即老人有语音需求时可以及时提供相应服务，老人摔倒时可以给老人送药并向前端发送照片和警报；前端交互模块则由微信小程序实现，使家人能远程关注家中老人的身体健康状况。

关键词

独居老人，语音识别，姿态检测，智能家居

An Intelligent Home Monitoring System for the Elderly Living Alone

MinDong, HaoyuTan, LimingYang, YuShen, ZhangshaoChen, ShengBi

School of Computer Science & Engineering, South China University of Technology, GuangzhouGuangdong

Received: Jul. 4th, 2024; accepted: Jul.25th, 2024; published: Aug.5th, 2024

Abstract

With the deepening of population aging and the increasing burden of social elderly care, it is very important to handle the issue of elderly care for the whole society. On a global scale, the physical health of the elderly is threatened by many deadly diseases. However, elderly people living alone lack family care and psychological comfort, leading to a higher incidence of illness and the inabili-

文章引用:董敏, 谭皓禹, 杨礼铭, 沈煜, 陈章韶, 毕盛.面向独居老人的智能居家监护系统[J].嵌入式技术与智能系统,2024,1(1):11-22.DOI: 10.12677/etis.2024.11002

ty to receive timely assistance in case of accidents. Faced with a triple dilemma of mental, health, and unexpected risks, they have greater health risks. The system proposed in this article is based on a cloud edge end architecture, consisting of cloud servers, perception control, and application services. It realizes voice services and emergency services for the elderly, which means that when the elderly have voice needs, they can provide corresponding services in a timely manner. When the elderly fall, they can be given medicine and sent photos and alerts to the front end; the front-end interaction module is implemented by a WeChat mini program, allowing family members to remotely monitor the physical health status of the elderly at home.

Keywords

The Elderly Living Alone, Speech Recognition, Attitude Detection, Smart Home

1. 引言

第七次人口普查公报[1]结果显示，全国人口中 65 岁及以上人口为 190,635,280 人，占总人口数的 13.50%，其中 65 岁以上的独居老人人口总数为 1988 万人，占 65 岁以上老人总数的 10.43%。65 岁以上老年人的前 4 大死亡原因分别是呼吸系统疾病、循环系统疾病、肿瘤和意外伤害[2]，而独居老人在面对上述各种疾病或意外伤害时具有更高的死亡率[3]，这说明独居老人具有更大的健康风险。同时由于老年人生理机能变化，活动量减少，行动不便，缺乏照顾，他们发生意外伤害的可能性随年龄的增长而不断上升，研究表明意外跌倒是我国老年人意外死亡的第一因素[4]，当独居老人发生意外跌倒时，若无法及时得到救助，可能危及生命，导致悲剧的发生。

本文设计的面向独居老人的智能居家监护系统可以通过语音识别联动服务机器人实现对老人的需求做出相应反应；还能采用基于毫米波雷达点云数据的深度学习方式检测老人的姿态，并联动服务机器人；在出现老人摔倒或呼救等危险情况时，系统会及时通过前端交互界面给家人发送警报，同时机器人能够自动导航至老人所在房间，提供应急服务。该系统实现具体包含了五个功能模块：云端服务器模块、语音识别模块、姿态检测模块、机器人联动服务模块、前端交互模块。

1.1. 研究现状

随着第四次科技革命推动的物联网、信息技术、大数据和云计算等技术的发展，养老服务水平也被极大地提升，部分智能家居技术也被用于养老服务中[5]。基于养老的智能家居服务用于满足老年人安全、独立、健康和援助等需求，在照顾老年人方面发挥着重要作用[6]。当前的智能家居的适老化研究主要集中在对老人身体健康指标的监测和老人异常状态检测，例如突然跌倒等。其中，在健康监测方面有不少研究将智能手环与智能家居系统结合，实现对老年人的健康检测[7]，通过让老年人佩戴包含加速度计、陀螺仪、脉搏传感器和体温传感器的智能手环，采集这些数据送往中控平台进行分析，以获得老年人的体温和心跳频率的健康状态，同时通过分析加速度计和陀螺仪的时序数据，推断老年人的运动姿态，判断其是否有摔倒的迹象。同时国内外也有利用视觉检测技术结合深度学习来检测老人是否发生跌倒[8][9]。但是调查显示，穿戴类设备并不适合给老年人使用[10]，一是老年人容易遗忘佩戴设备；二是穿戴类设备

容易丢失。基于视觉检测的视频监控能够完成非接触式的摔倒检测任务，但近几年因为家庭网络摄像头遭到入侵而造成隐私泄露的事件层出不穷，大部分老年群体也并不希望时刻有摄像头监控自己。对于老人的紧急求助需求，目前的主流产品均为一键呼叫器，求助按钮或随身佩戴，或固定在房间某处，而固定在房间某处的呼叫器更是不便于甚至无法为突发危险情况的老人提供紧急求助服务。调研市场发现，目前面向老人的监护产品还存在着功能单一、不成系统，只“监”无“护”，即无法为发生危险情况的独居老人及时提供应急服务的问题。从关心独居老人的健康状况出发，针对上述的不足，本文设计实现了一个可以实时监测突发危险情况，且能根据需求提供便利服务的面向独居老人的智能居家监护系统。

1.2. 本系统特色

本文面向独居老人的智能居家监护系统的特色有以下三点：

第一，我们采用毫米波雷达实现老人的姿态检测，克服了主流摔倒检测产品的局限性。基于毫米波雷达的姿态检测是非接触式的检测方式，克服了穿戴类设备的不足；同时也解决了基于视觉的视频监控的痛点，既防止隐私泄露，也满足了老人不愿意被监视的心理需求。

第二，我们使用语音识别的方式解决独居老人的求助需求。只要在房间内布置一个语音识别模块，老人在该房间内的求助语音都能够被接收并发送给家人，相比按钮求助类产品更加方便。

第三，我们将姿态检测功能和语音识别功能组成系统，并在系统内添加了机器人联动服务模块。服务机器人能够根据姿态检测模块和语音识别模块检测到的信息提供相应的服务，解决了只“监”无“护”的问题。

2. 系统总体方案设计

本文基于云－边－端架构，实现了面向独居老人的智能居家监护系统，其主要功能有云端服务器、感知控制和应用服务，总体功能架构如图 1 所示。

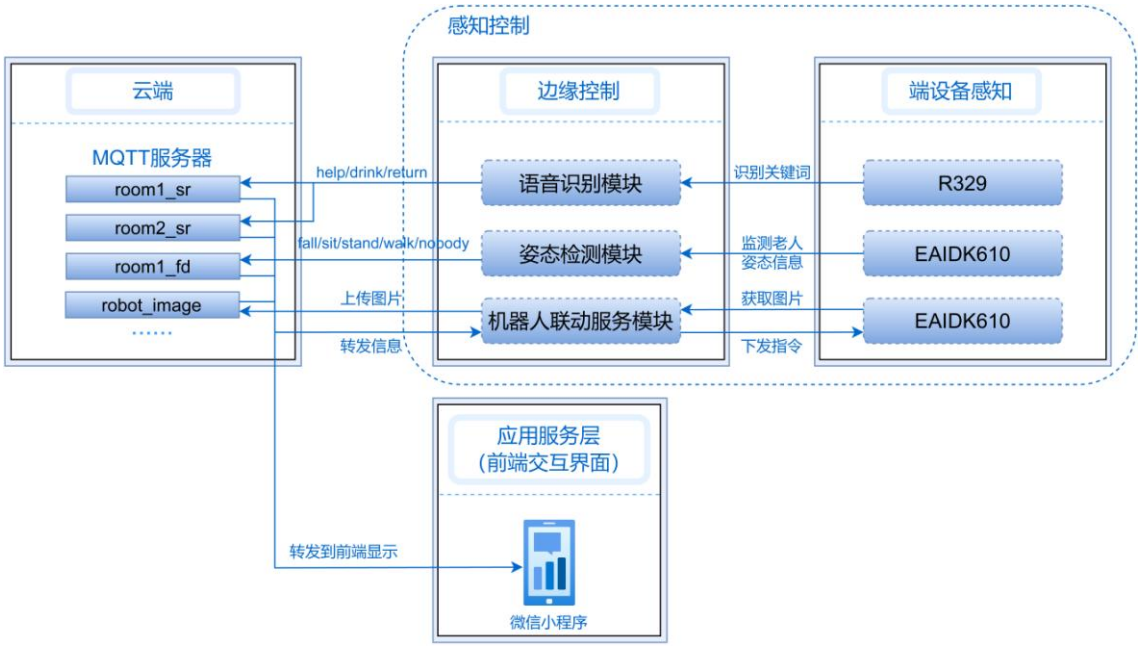

Figure 1. Overall functional architecture diagram of the system
图 1.系统总体功能架构图

云端服务器为系统内部各模块提供通信服务。

感知控制方面，语音识别模块主要对老人的语音需求进行感知，采用 R329 开发板[11]作为开发平台来实现。该模块可以识别到老人的关键词需求，比如喝水、救命等，并向云端服务器下的相应话题发布对应的需求信息。

姿态检测模块主要负责以非接触式检测识别老人在房间内的姿态，采用毫米波雷达以及基于 RK3399 芯片的 EAIDK610 开发平台[12]进行实现。该模块可以识别房间内老人是否摔跤以及一些常见姿态，并向云端服务器下的相应话题发布对应的姿态信息。

机器人联动服务模块主要负责在老人有相应需求或者有危险时能够及时提供帮助或发送警报。通过使用 EAIDK610 开发平台控制机器人，实现机器人联动服务模块。它可以订阅语音识别模块和姿态检测模块的话题，从云端服务器接收到对应话题的信息，然后控制服务机器人做出对应行动。

应用服务方面，前端交互模块由微信小程序实现，用户可以进入房间页面查看各房间状况、语音需求以及危险情况报警等，使家人能够远程关注老人在对应房间的状态信息。

3. 语音识别模块

语音识别模块使用到的硬件设备为采用 ARM v8 架构的 R329 开发板。它基于全志 R329 芯片设计，搭载双核 ARM CortexTM-A53，同时内置高性能的周易 AIPU 处理器，使得它支持智能语音和视频图像处理，也能够直接跑一些人脸识别模型、语音识别模型等。在本模块中，R329 主要用于部署语音识别模型实现关键词识别功能，能够在识别老人关键词需求后发送相应需求信息给云端服务器的对应话题下。

考虑了多种老人在家可能提出的需求，并借助关键词识别功能实现了对这些需求的几种不同关键词的识别，语音识别模块在识别到关键词后向云端服务器下的相应话题发布该关键词对应的需求信息。比如，当老人在房间 1 发出“喝水”“送水”“渴了”等需求时，语音识别模块会识别这三个关键词，并向服务器下的“room1_sr”话题发布信息“drink”；当老人在房间 2 发出“救命”“有人吗”等需求时，语音识别模块会识别这两个关键词，并向服务器下的“room2_sr”话题发布信息“help”。

选取到适合居家助老服务环境下的模型参数，采用了不同尺寸的语言模型和不同的模型输入长度，并对其识别错误率和延迟时间进行比较，得出最后的模型参数选择。

1) 模型输入长度选择。取声学模型 am_7332，语言模型 lmS，语音声音在 50dB 左右，我们修改模型输入长度分别为 128，192，256 进行实验，得到结果如表 1 所示。

Table 1. Average error rate and delay time under different model input lengths
表 1.不同模型输入长度下的平均错误率和延迟时间

模型输入长度	128	192	256
平均错误率	0.728	0.138	0.131
解码时间/ms	1.552	7.72	15.691
关键词解码时间/ms	3.772	5.398	5.699

通过分析表 1 结果，当模型输入长度选取 192 时，识别率和长度为 256 时一样高，同时相比之下也具有更及时的响应。因此将模型输入长度设置为 192。

2) 模型尺寸选择。取声学模型 am_7332，语音声音在 50dB 左右，模型输入长度为 192，我们修改语言模型尺寸进行实验，得到结果如表 2 所示。

通过分析表 2 结果，三种尺寸识别率相差不大。因此，我们选择模型 lmS 部署到开发板上，这样可以占用最少的内存，同时该模型 0.862 的识别率也较高，足够满足识别语音的需求。

Table 2. Average error rate and delay time for different language model sizes
表 2.不同语言模型尺寸下的平均错误率和延迟时间

语言模型尺寸	lmS 12MB	lmM 104MB	lmL 750MB
平均错误率	0.138	0.109	0.099
解码时间/ms	5.72	5.919	6.284
关键词解码时间/ms	5.398	5.745	5.89

3) 为了使该模块根据关键词识别概率向服务器发送对应消息，需要设定一个概率限值来标识确实成功识别到关键词。取声学模型 am_7332，语言模型 lmS，模型输入长度为 192，用不同分贝音量，在不同距离下进行实验，得到结果如表 3 所示。

Table 3. Recognition probabilities at different decibel volumes and distances
表 3.不同分贝音量和不同距离下的识别概率

音量 / 距离	30dB	50dB
0.5m	0.248	0.968
1m	0.133	0.535
2m	0.004	0.073

由表 3 中数据可见，当声音较小时识别概率会大幅降低，当距离较远时模块也能采取到关键词但识别概率都低于 0.01。因此，考虑到老人遇到紧急情况身体较虚弱声音不大，同时排除房间外的其他声音干扰，我们将概率限值设为 0.1。当模块的关键词识别概率高于 0.1，语音识别模块就会向服务器发送相应信息。

4. 姿态检测模块

姿态检测模块所需要使用的硬件设备主要是 TI 公司型号为 IWR6843ISK 的毫米波雷达和采用 ARM 架构的 RK3399 芯片人工智能开发平台 EAIDK610 开发板。在本模块中 EAIDK610 用于部署姿态检测模型，完成从毫米波雷达获取数据、数据预处理，推理得出姿态检测结果并通过云端服务器将报警信息发送到用户客户端的功能。

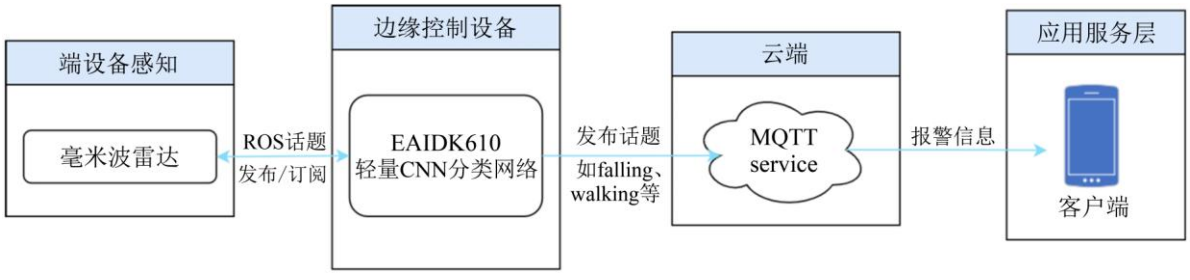

Figure 2. Overall architecture of attitude detection module
图 2.姿态检测模块总体架构

姿态检测模块的总体结构如图 2 所示，毫米波雷达通过发射一定频率的电磁波并利用多普勒效应检测区域内物体的运动趋势，即当范围内的物体靠近雷达时会收到频率更高的回波，远离时会受到频率更底的回波，以此计算物体与毫米波雷达之间的位置关系及物体的速度，这类计算在硬件内部即可完成，生成四元组数据。EAIDK610 开发板接收毫米波雷达所发布的话题获得该四元组，并通过数据预处理之后获得固定大小的网络输入后送入轻量 CNN 分类网络进行推理。推理结果发送至云端服务器，并通过服务器将该结果发送至应用服务层的微信小程序，以实现提示、报警功能。

4.1. 数据集的构建与数据预处理

毫米波雷达被设定为只检测并发送范围内活动物体的四元组信息，这使得在开发板订阅毫米波雷达所发送的四元组数据并推理的时候会出现数据截断的可能。考虑到人体姿态活动是一种具有时序性之的一系列连续活动，在网络推理的过程中需要利用到序列的信息，并且神经网络数据的输入需要固定维度大小，因此在训练网络阶段和推理预处理阶段都设计了一种等待队列如下图 3 所示，保证在收到固定大小为的 $N = 90$ 个四元组数据点之后才打包成[90, 4]的矩阵，作为网络的输入送至神经网络中进行推理，提高了网络推理的稳定性与推理结果的准确性。

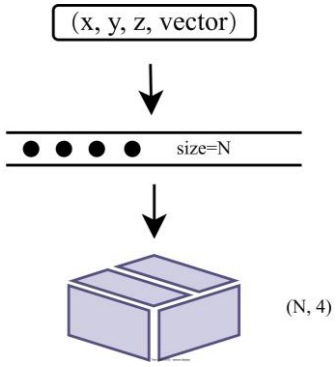

Figure 3. Waiting queue
图 3.等待队列

4.2. 神经网络的结构与训练

神经网络的具体结构如下图 4 所示。神经网络主干由 7 个部分构成，四元组点云数据经过等待队列打包之后作为神经网络的输入传入至网络之中，以此经过批标准化、卷积层、池化层、Dropout 层、Flatten 层和全连接层后输出最终的四分类结果。

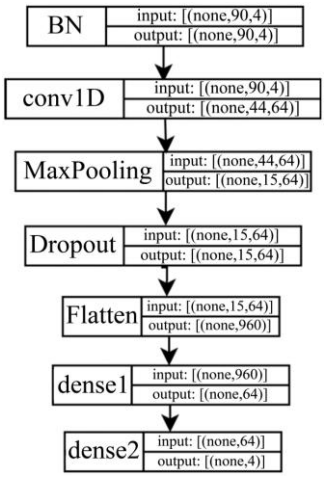

Figure 4. Neural network structure
图 4.神经网络结构

实验所使用的硬件配置为：AMD Ryzen 9 5900HS，主频 3.3GHz，8 核 16 线程；加速显卡 NVIDIA GeForce RTX3050Ti 8GB 显存。使用后端为 TensorFlow-GPU v2.11.0 的 Keras v2.11.0 作为训练模型的深度学习库。训练使用的数据集总数为 24,404 个，4 个不同的标签各占 6101 个。训练开始前，先用 shuffle 函数对从 ROSBAG 中处理得到的数据集进行随机打乱，然后使用 train_test_split 函数对数据集切分，数据集总数的 80%作为训练集，20%作为测试集。在训练过程中，将训练集的 20%作为验证集以评估模型的训练效果。训练中设置学习率为 0.0005，epochs 设定为 40，batch_size 设定为 128。模型训练过程中的损失函数如图 5 所示，在验证集上的 loss 为 0.00897，准确率达到 99.63%。

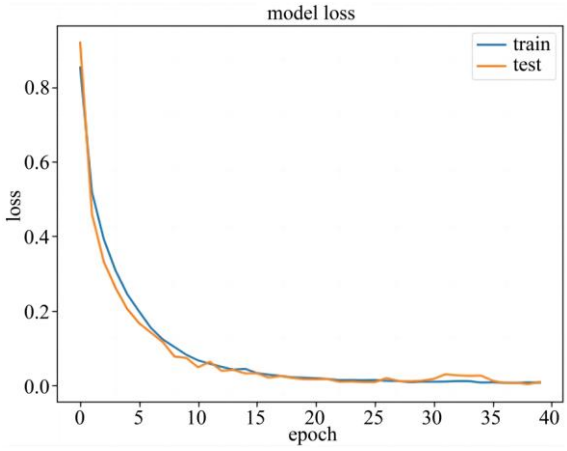

Figure 5. Loss function diagram
图 5.损失函数图

4.3. 神经网络的测试

经过训练阶段的拆分，得到的测试集分布为 falling、sitting、standing、walking 的标签各有 1194、1286、1189、1212 个。将测试数据集和标签输入到训练好的网络之中进行测试，得到预测结果如下图 6 所示。训练得到的神经网络在测试集上的总体准确率达到 99.63%，其错报的几率不足 0.5%，能够满足对身处卧室、浴室或客厅等各种场所的老人进行姿态检测的需要。

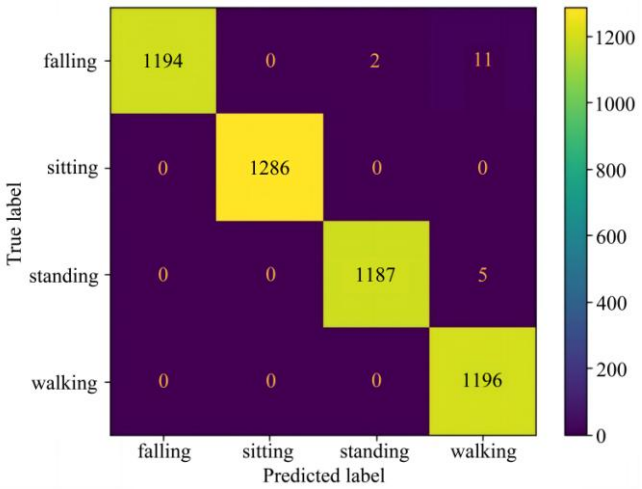

Figure 6. Test set prediction results
图 6.测试集预测结果

为了试验姿态检测模块识别准确率，还设置了七轮测试：测试者在每一轮的测试中以随机顺序做站、坐、走、摔倒四个动作，并记录下当时的识别结果，如表 4 所示。由表中数据可以看出，姿态检测模块的识别率较高，只发生了一次误判，该次误判可能是因为测试者测试时动作之间衔接过快。

Table 4. Recognition results of testers'actions in different test wheels
表 4.不同测试轮中对测试者动作的识别结果

动作 / 轮数	站(stand)	坐(sit)	走(walk)	摔倒(fall)
1	stand	sit	walk	fall
2	stand	sit	walk	fall
3	stand	sit	walk	fall
4	fall	sit	walk	fall
5	stand	sit	walk	fall
6	stand	sit	walk	fall
7	stand	sit	walk	fall

综上所述，本文设计的神经网络对于基于毫米波雷达点云数据的人体活动识别有较高的准确率，可以完成对人体姿态进行检测的任务，并与其他模块结合进行系统的测试实验。

5. 机器人联动服务模块

机器人服务功能主要分为两类。一类为语音服务，即机器人通过语音识别模块响应老人的语音指令，目前实现了机器人送水到老人所在房间、老人不需要机器人后命令其返回充电座两个功能用于展示。一类为老人异常应急服务，即在语音识别模块或姿态检测模块识别到老人出现异常状况时，能够自动导航至相应房间，送来急救药物、拍摄现场照片并上传至服务器，使家人能通过小程序及时了解老人状态并采取紧急措施。

EAIDK610 微处理器平台主要负责控制机器人的逻辑行为，机器人的功能建立在导航的基础上，导航需要地图，因此首先需要在家中通过激光雷达扫描建立平面地图。成熟的建图方法众多，根据任务场

景选择了在小场景下建图算力消耗较少且地图精度较高的 Gmapping 算法。选择 Navigation Stack 作为导航框架，该框架由多个导航组件构成，其中核心组件 map_server 用于加载建图算法创建的地图；amcl 算法估计机器人在地图中的位姿；move_base 用于路径规划和控制机器人的行动，其中包括了全局路径规划器(A*算法)和局部轨迹规划器(TEB 算法)。这些组件共同工作，令机器人能够在复杂环境中实现自主导航。

服务机器人的部署分两步，一是创建家里的平面地图，二是手动标记各个房间的导航目标点以及机器人的充电座。首先开启激光雷达和 Gmapping 节点，遥控机器人在家中各个房间探索，探索完毕后保存地图。然后开启编写好的房间标记节点，通过可视化工具 RVIZ 依次标记各个房间的目标点，这些目标点会保存至文档中，供后续联动服务使用。至此，机器人的前期部署完成。

机器人的联动时序图如图 7。

当老人发出正常的语音指令，比如“送水”时，语音识别模块能够识别出来并通过云端服务器转发指令至机器人控制模块，机器人控制模块则会发布导航目标点，使服务机器人导航到老人所在的房间；当老人希望机器人离开房间时，说一声“回去”，机器人就会收到相应指令并导航至充电座；而当老人发生意外，比如心脏不舒服，呼喊“救命”时，或者是摔倒时，语音识别模块或姿态检测模块能够及时检测到异常并上报服务器，服务器将相关消息转发至微信小程序和机器人控制模块，此时机器人控制模块收到求救指令并控制带着急救药物的机器人导航至指定房间，拍摄照片并上传至服务器，家人可以及时通过小程序看到老人的情况，以便及时采取相应措施帮助老人脱离险境。

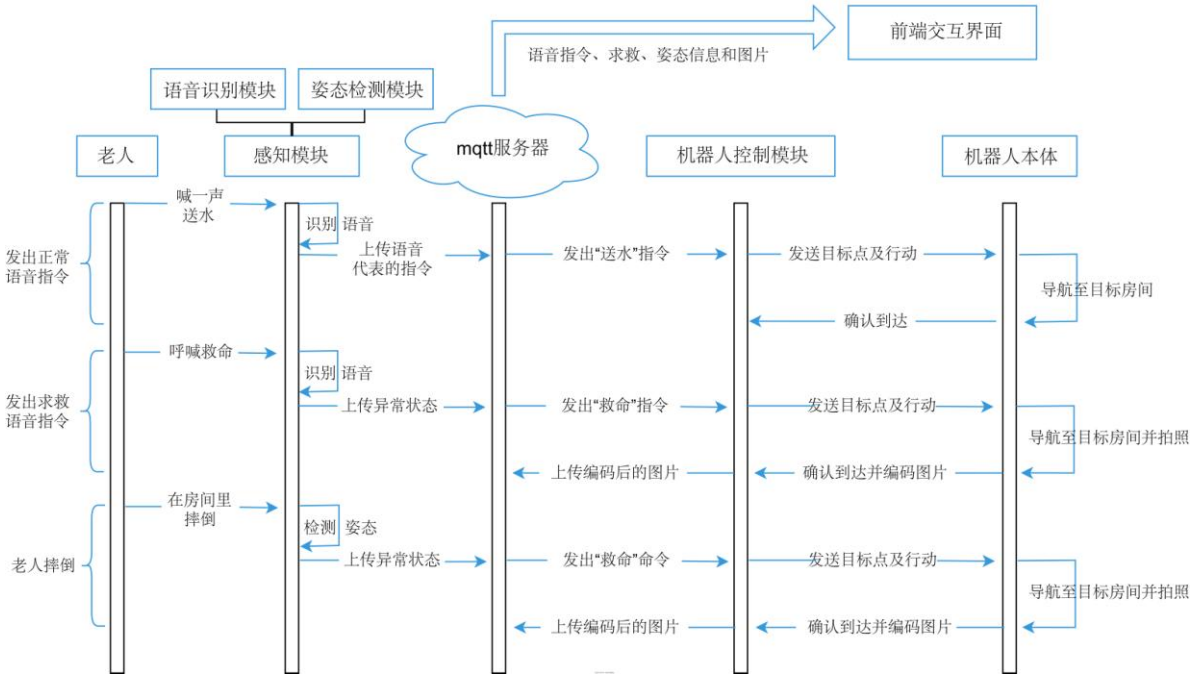

Figure 7. Sequence diagram of robot linkage service
图 7.机器人联动服务时序图

6. 云端服务器

采用 MQTT (Message Queuing Telemetry Transport)协议作为系统内部通信方式，同时把 mqtt 服务器和用于反向代理的 nginx 服务器都采用 docker 容器的方式部署在云服务器中。

mqtt 服务器与各模块通信如图 8。mqtt 服务器与语音识别模块、姿态检测模块、机器人联动服务模块之间可以直接使用 mqtt 协议通信。

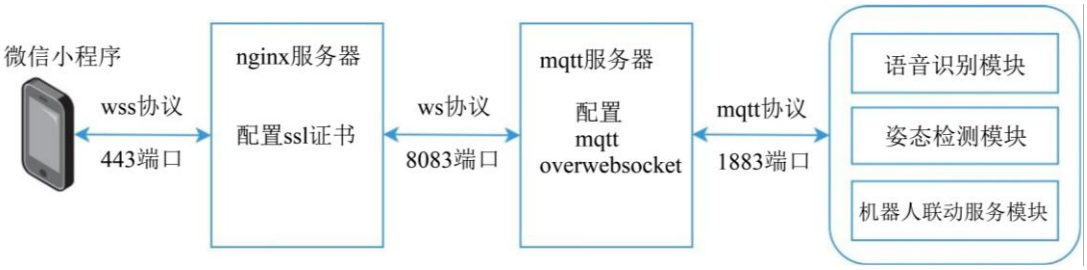

Figure 8. Schematic diagram of communication between mqtt server and various modules
图 8.mqtt 服务器与各模块通信示意

mqtt 服务器与前端通信较为复杂。由于微信小程序必须使用 wss(Secure WebSocket)协议保证通信安全，因此我们首先需要在 mqtt 服务器中将 mqtt 协议封装在 ws(WebSocket)协议中(配置 mqttoverwebsocket 功能)，由域名经过 ssl 认证的反向代理服务器 nginx 将 ws 协议升级为 wss 协议，最终完成 mqtt 服务器与微信小程序的通信。

7. 前端交互模块

前端交互模块基于微信小程序实现了创建/删除房间、查看房间状态、调整房间状态、查看历史记录、查看现场等功能，如图 9 所示。

Figure 9. Front end function display
图 9.前端功能展示

8. 系统测试

在测试之前，需要先对两个实验房间进行部署，其中语音识别模块在两个房间都有部署，姿态检测模块只部署在房间 1，机器人的充电基座(起始出发点)设置在走廊，各模块实验场景部署如图 10 所示。

Figure 10. Experimental scene diagram
图 10.实验场景图

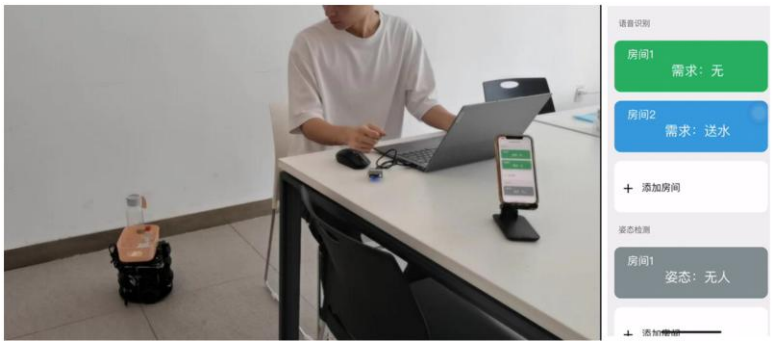

Figure 11. The robot arrives at Room 2 and delivers water to the target navigation point
图 11.机器人到达房间 2 目标导航点送水

1) 语音识别与机器人联动功能

进行语音服务需求的测试，测试者坐在房间 2 说“喝水”，语音识别模块识别到关键词，并将对应需求发到云端服务器的“room2_sr”话题下。机器人订阅到信息后，导航至房间 2 为测试者送水。机器人到达目标导航点，并且小程序收到信息，如图 11 所示。

进行异常情况的语音需求测试，测试者在房间 1 呼喊“救命”，语音识别模块识别到关键词，并将对应需求发到云端服务器的“room1_sr”话题下。机器人订阅到信息后，就会导航到房间 1 为测试者送药，并将目标导航点处的照片通过云端服务器发送到小程序。在小程序中长按房间 1 卡片，点击“查看现场”，可以看到现场照片，如图 12 所示。

Figure 12. Testers shout “help”, robots deliver medicine and take photos on site
图 12.测试者呼喊“救命”，机器人送药以及现场拍照

2)姿态检测与机器人联动功能

主要测试姿态检测与机器人联动的功能，测试方法为在实验场地部署毫米波雷达和载有姿态检测模块的 EAIDK610 开发板，测试者在实验场地做走、站、坐、摔倒四个动作，观察系统是否正确识别到姿态信息并将对应信息发送到云端服务器，以及机器人是否及时、准确地去到对应房间的目标导航点。

前期部署完成后，当机器人订阅到摔倒(“fall”)信息时，机器人联动服务模块便会控制携带药品的机器人到房间 1 的目标导航点为摔倒的测试者送药，并将警报信息和现场照片通过云端服务器发送到小程序显示，如图 13 所示。

Figure 13. Detecting a person's fall and issuing an alarm through millimeter wave radar
图 13.通过毫米波雷达检测到人摔倒并警报

通过以上实验测试，整个系统能较好的实现独居老人居家监护功能。

9. 结束语

针对社会老龄化程度的不断加深的背景，本文提出并设计实现了一种面向独居老人的智能居家监护系统，基于云－边－端架构来设计系统的总体功能框架，包括了云端服务器，部署到端设备上的语音识别、姿态检测、机器人联动服务、前端交互等功能模块，并进行了测试验证。

参考文献

[1] 国家统计局.第七次全国人口普查公报(第五号) [EB/OL]. https://www.gov.cn/, 2021-05-11.

[2] 李庆瑜.老年人意外伤害及社区干预研究[J].护理研究(中旬版),2006(29):2635-2636.

[3] 黄润龙,杨春.我国孤寡独居老人的构成及其生活状况研究[J].人口与社会,2021,37(5):26-37.

[4] 徐大麟,孙伯燕,王萍玉.1986~2003 年上海市静安区老年人意外死亡的特点和趋势[J].中华老年医学杂志,2004(11):59-61.

[5] Beard,J.(2010)InnovativeApproachestoDealingwithPopulationAgeing.*Gerontechnology*,**9**,64. https://doi.org/10.4017/gt.2010.09.02.002.00

[6] Zhang,Q.,Li,M.andWu,Y.(2020)SmartHomeforElderlyCare:DevelopmentandChallengesinChina.*BMC Geriatrics*,**20**,Article No. 318.https://doi.org/10.1186/s12877-020-01737-y

[7] Hu, B.D.C., Fahmi, H., Yuhao, L., *et al*. (2018) Internet of Things (IOT) Monitoring System for Elderly. 2018 *International Conference on Intelligent and Advanced System* (*ICIAS*),Kuala Lumpur,13-14 August 2018, 1-6. https://doi.org/10.1109/ICIAS.2018.8540567

[8] Lafuente-Arroyo,S.,Mart ń-Mart ń,P.,Iglesias-Iglesias,C.,Maldonado-Basc ón,S.andAcevedo-Rodr íguez,F.J.(2022)RGBCamera-BasedFallenPersonDetectionSystemEmbeddedonaMobilePlatform.*Expert Systems with Applications*,**197**,Article 116715.https://doi.org/10.1016/j.eswa.2022.116715

[9] 蒋阳阳. 基于 YOLOv3 的居家老人摔倒检测系统的设计与实现[D]: [硕士学位论文]. 南京: 南京邮电大学,2021.

[10] Steele,R.,Lo,A.,Secombe,C.andWong,Y.K.(2009)ElderlyPersons'PerceptionandAcceptanceofUsingWirelessSensorNetworkstoAssistHealthcare.*International Journal of Medical Informatics*,**78**,788-801. https://doi.org/10.1016/j.ijmedinf.2009.08.001

[11] MaixSense 简介[EB/OL]. https://wiki.sipeed.com/hardware/zh/maixII/M2A/maixsense.html,2022-10-27.

[12] EAIDK-610 公开教程(接口编程、数据采集、视觉分析)Python 版[EB/OL]. https://aijishu.com/a/1060000000362337,2022-10-25.

EmbeddedTechnologyandIntelligentSystems 嵌入式技术与智能系统,2024, 1(1), 23-30
Published Online August 2024 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2024.11003

移植Cortex-M程序到RV32中的问题

林金龙

北京大学软件与微电子学院，北京

收稿日期：2024年7月4日；录用日期：2024年7月25日；发布日期：2024年8月5日

摘要

具有开源、简单和灵活等特点，RISC-V架构受到业界广泛关注。近年来，市场上相继出现了多款RISC-V架构微处理器，32位RISC-V架构MCU正逐步进入Cortex-M MCU应用领域。本文针对将应用程序从RV32移植到Cortex-M的需求，分析RV32与Cortex-M结构、编程模型和过程调用规范等方面的不同之处，提出程序移植过程中遇到的问题，提出方法和建议，并进行相关性能分析和比较。

关键词

RISC-V，RV32，Cortex-M，程序移植

Problems in ProgramPorting from Cortex-M to RV32 Programs

JinlongLin

SchoolofSoftwareandMicroelectronics, PekingUniversity, Beijing

Received: Jul. 4th, 2024; accepted: Jul.25th, 2024; published: Aug.5th, 2024

Abstract

As a simplicityand flexibility open source microprocessor architecture, RISC-V has received widespread attention in the industry. In recent years, a lot of RISC-V microprocessors have been emerging in the market. The MCUs with the core of RV32 core are gradually entering into the application fields of Cortex-M MCUs. This article introduces the differences between RV32 and Cortex-M in terms of structure, programmer's model, and procedure call convention, discusses the problems during the program porting processfromCortex-MtoRV32, proposes some suggestions, and conducts relevant performance comparison.

Keywords

RISC-V,RV32,Cortex-M,Program Porting

文章引用:林金龙.移植 Cortex-M 程序到 RV32 中的问题[J].嵌入式技术与智能系统,2024,1(1):23-30.
DOI: 10.12677/etis.2024.11003

1. 引言

开源架构处理器 RISC-V[1]，在嵌入式系统中得到了越来越多的应用，近年多家处理器厂商发布了RV32 架构 MCU。2019 年 4 月，SiFive[2]发布了 Freedom E310；2020 年 2 月，兆易创新[3]发布 RV32 MCU GD32VF103；沁恒微电子[4]发布 CH32V、CH32X、CH32L 系列 RV32 MCU；先辑半导体[5]发布 HPM5XXX 和 HPM6XXX 系列 RV32 单核和多核 MCU；瑞萨电子[6]发布 RV32 汽车 MCU RH850/U2B 和通用 MCU R9A02G021 等。

随着 RISC-V MCU 的快速发展，其软件生态正逐步完善和丰富。Embedded Studio[7]、IAR[8]等主流商业 IDE 以及开源 IDE eclipse [9]等嵌入式软件开发环境支持 RISC-V MCU。一些主流操作系统已经支持 RISC-V 架构[10]。QEMU RISC-V 虚拟化平台[11]支持多种 RISC-V 处理器仿真；FreeRTOS 发布支持 RISC-V MCU 版本[12]；郃阳等[13]基于 QEMU RISC-V 实现 OpenHarmony 移植和优化；Nicholas Gordon 等[14]将 Kitten Lightweight Kernel 操作系统移植到 RISC-V；Luming Zhang[15]移植并优化 RISC-V UFEI Boot；Robert Balas 等[16]分析 RV32IMC 结构特点并提出程序方法。

第二届滴水湖中国 RISC-V 产业论坛现场调查[17]结果表明，RISC-V 架构处理器已经成功应用于无线连接芯片、工业控制芯片，网络通信芯片和边缘计算芯片等场景。目前，在移动通信、物联网和工业控制等嵌入式应用领域，ARM 架构处理器占市场主导地位[18]。在一些领域和应用场景，RISC-V 将对 ARM 的市场地位构成挑战。RV32 MCU 正争夺 ARM Cortex-M MCU 的市场份额。将 Cortex-M MCU 应用程序移植到 RV32 MCU，充分利用 Cortex-M MCU 的成熟生态，将有利于 RISC-V 的发展和推广。由于 RV32 和 Cortex-M 的结构和编程模式存在差异，尽管使用程序开发工具能够将源程序的编译、汇编和链接生成 RV32 执行程序，但在程序移植过程中仍然会遇到一些问题。

本文根据 RV32 与 Cortex-M 在结构、编程模型和过程调用规范等方面的不同点，分析应用程序从 RV32 移植到 Cortex-M 过程中遇到的问题，提出解决方法和建议，并进行相关性能分析和比较。

本文第一节简介 RISC-V 发展状况，讨论 RV32 的应用前景；第二节比较 RV32 与 Cortex-M 异常处理器机制，说明移植中断处理程序面临的问题；第三节对比 RV32 与 Cortex-M 指令架构，分析 RV32 指令模块组合对程序性能的影响；第四节讨论 RISC-V 与 ARM 处理器程序过程调用规范的差别，分析其对程序移植影响；第五节对论文工作进行总结。

2. 中断处理

在 ARM 架构处理器手册[19]和 RISC-V 架构处理器使用手册[20]中，用编程模式(Programmer’s model)表示处理器架构中涉及程序开发的内容。编程模式通常包括处理器支持的数据类型，通用和特殊功能寄存器、异常和中断响应机制和指令集等，异常和中断处理处理器基本功能。

通常将由外部信号引起的异常称为中断。中断处理是 MCU 应用系统的关键功能之一。中断处理功能包括两个部分，中断响应机制和中断服务程序(Interrupt Service Routine)。中断响应机制由处理器硬件结构决定，中断服务程序则与中断响应机制相关。

2.1. 中断响应机制

表 1 列出了 Cortex-M 和 RV32 中断响应机制的差异。将应用程序从 Cortex-M 移植到 RV32 时，需要修改中断处理功能相关程序。

Table 1. The exception handling of RV32 and Cortex-M
表 1.RV32 和 Cortex-M 异常管理

	Cortex-M	RV32
中断响应	向量	向量和非向量
上下文保存	自动	手动
C	可重定位(VTOR)	设置 mtvec
控制器	NVIC	定制

Cortex-M 仅支持向量中断响应，中断向量指向对应的中断服务程序入口，处理器启动后通过寄存器 VTOR 冲重定位中断向量表。RV32 支持向量和非向量两种中断响应方式，处理器器复位时缺省为非向量响应形式，中断响应时指向程序空间的起始地址，通常为 0x00。处理器启动后，通过设置机器模式异常向量基址寄存器 mtvec 设置中断响应模式和异常向量入口地址。

为了保证移植前后功能的一致性，将应用程序移植到 RV32 后仍保证向量中断响应方式。RV32 MCU 复位后首先执行初始化序，将中断向量表地址值的高 30 位写入 RV32 寄存器(CSR)mtvec 的 mtvec[31:2]，将 01 写入 mtvec [1:0]，选择向量中断响应模式。

目前市场上 RV32 MCU 外设类型、接口和控制方法与 Cortex-MMCU[3][19][20]相似，中断向量表结构的结构相近。表 2 对比了 Cortex-M MCU STM32F429 与 RV32 MCU GD32VF103 中断向量表结构。

Table 2. The interrupt vector table ofSTM32F429 and GD32VF103
表 2. STM32F429 与 GD32VF103 中断向量表结构

	STM32F429	GD32VF103
中断向量表	_vectors: .word __stack_end__ .wordReset_Handler …… .word ExternalISR0 .word ExternalISR1 ……	_vectors: j Reset_Handler .word 0 …… .word ExternalISR0e .word ExternalISR1 ……

如表 2 所示，RV32 MCU 中断向量中，除向量 0 外，每个 32 位向量值是对应中断服务程序的入口地址。由于复位后 RV32 缺省为非向量中断响应模式，需要首先设置中断响应模式和向量表基地址，向量 0 是跳转指令，跳转到复位启动程序入口。

2.2. 上下文处理

Cortex-MMCU 响应中断请求时，硬件自动依次将 xPSR，PC，LR，R12 以及 R3 - R0 压入栈中，保存上下文；中断服务程序返回时，硬件自动从栈中将数据弹回对应寄存器，恢复上下文。

RV32 MCU 响应中断请求时硬件自动保存 PC 到控制和状态寄存器 mepc，并保存特权模式至 mstatus.MPP，但不保存上下文相关的通用寄存器和其他特殊功能寄存器。从中断服务程序返回时，硬件仅自动恢复寄存器 mstatus 和 PC。将 Cortex-MMCU 中断服务程序移植到 RV32 MCU 时，需要在中断服务程序中添加保存和恢复上下文语句或函数。

Cortex-MMCU 应用程序和中断服务程序遵守 ARM 过程调用规范 AAPCS[17]，硬件在中断响应过程中自动保存 4 个特殊功能寄存器和 4 个参数寄存器 r0-r3 或称为 a0-a3。RISC-V 应用程序过程调用规范约定 8 个参数寄存器 x10-x17 或称为 a0-a7。与 Cortex-MMCU 自动保存的上下文内容对照，RV32 MCU 中断服务程序中需要保存和恢复上下文内容最小包括参数寄存器 a0-a7 和特殊功能寄存器。表 3 列出了 RV32 MCU 中断服务程序中保存和恢复上下文最小集的程序语句。

Table 3. The programs of save and restore context in RV32 MCU interrupt service routine
表 3. RV32 MCU 中断服务程序保存和恢复上下文程序语句

	保存上下文	恢复上下文
修改 sp	addisp, sp, -12*4	addisp, sp, -12*4

控制和状态寄存器	csrr x5, mepc
	sw x5, (sp)
	csrr x5, mstatus
	sw x5, 4(sp)
		lw x5, (sp)
		csrwmepc, x5
		lw x5, 4(sp)
		csrwmstatus, x5

通用寄存器	swra, 2*4(sp)
	sw a0, 3*4(sp)
	sw a1, 4*4(sp)
	sw a2, 5*4(sp)
	sw a3, 6*4(sp)
	sw a4, 7*4(sp)
	sw a5, 8*4(sp)
	sw a6, 9*4(sp)
	sw a7, 10*4(sp)
		lwra, 2*4(sp)
		lw a0, 3*4(sp)
		lw a1, 4*4(sp)
		lw a2, 5*4(sp)
		lw a3, 6*4(sp)
		lw a4, 7*4(sp)
		lw a5, 8*4(sp)
		lw a6, 9*4(sp)
		lw a7, 10*4(sp)

在中断服务程序的入口添加表 3 中保存现场语句或函数，在返回语句前添加恢复现场语句或函数。对于定制中断响应机制 Gd32VF103[21]等 MCU，则需要依据其使用手册编写中断处理相关的程序。

3. 指令集模块

Cortex-M MCU 采用 Thumb 指令集。RV32 MCU 采用模块化指令，生成应用程序时可以选择指令集模块及其组合。将 Cortex-M MCU 应用程序移植到 RV32 MCU 时，选择与 Thumb 指令集相应功能指令集模块组合，以保持移植前后程序功能和性能的一致性。

为了便于分析和评估性能，本文选择 STM32F429 和 FE310 分别作为 Cortex-M MCU 和 RV32 MCU 样本，使用 CoresMark[22]做性能分析；程序生成工具选择 Segger 公司 Embeddedstudio[7]；汇编和编译器选择 gcc，版本为 gnu4.2.1。

STM32F429 内核为 Cortex-M4[23]，指令集为 Thumb2。FE310 支持 RV32imac 指令集模块，处理器指令集功能是所有子模块功能的并集。表 4 对 STM32F429 与 FE310 指令集部分功能进行了比较。

Table 4. The partial functions of STM32F429 and FE310 instruction sets
表 4.STM32F429 与 FE310 指令集部分功能

功能	Thumb2	RV32I	M	A	C
算数/逻辑	是	是	-	-	-
乘法/除法	是	-	是	-	-
内存访问	是	是	-	-	-
原子	否	-	-	是	-

16 位指令	支持	-	-	-	是

生成 RV32 MCU 应用程序时，选择不同的指令集或指令集组合，将会影响程序的性能。

3.1. RV32i vs RV32im

Embedded studio 创建 FE310 应用程序工程时缺省使用 RV32i 指令集模块，该模块没有乘法和除法指令。程序中的乘除法运算能够正常编译，编译器通过调用由 RV32i 指令实现的运算函数库实现。如果在编译时选择 RV32im 指令集模块组合，则编译器将直接使用乘法和除法指令实现运算，提高程序执行速度。表 5 列出了 c 程序采用 RV32i 和 RV32im 指令集编译后生成汇编指令对照。

Table 5. The assembly instructions for RV32i and RV32im
表 5.RV32i 与 RV32im 汇编指令

C 函数	RV32i	RV32im

int add4(int p1, int p2, int p3, int p4)
{
return (p1 * p2 + p3 /p4);
}
	add4:
	mv s1, a2
	mv s2, a3
	call __mulsi3
	mv s0, a0
	mv a1, s2
	mv a0, s1
	call __divsi3
	add a0, s0, a0
	jrra
		add4:
		div a2, a2, a3
		mul a0, a0, a1
		add a0, a0, a2
		ret

表 6 列出了选择 RV32i 与 RV32im 指令集在 FE310 模拟器上运行 CoreMark 得分。结果表明，如果应用程序中含有乘法和除法运算，将 RV32i 改为 RV32im 指令集将提高程序运行速度。

Table 6. The CoreMark scores of RV32i and RV32im
表 6. RV32i 与 RV32imCoreMark 得分

	RV32i	RV32im
得分(/MHZ)	1.33	2.43

3.2. RV32im vs RV32imc

由于 MCU 处理器中存储资源受限，对 ROM 和 RAM 的需求是开发 MCU 应用程序关注的重点之一。Cortex-M 采用支持 16 位指令 Thumb 指令模式，以减少应用程序的体积。RV32i 和 RV32im 指令长度是 32 位。对于同一源程序，使用 RV32im 指令集生成的二进制目标程序的长度将会大于 Cortex-M 目标程序的长度，从而增加对存储资源的需求。选择 RV32imc 指令集组合，将指令长度从 32 位变为 16 位，减少所生成二进制目标程序的长度。表 7 列出了 Coremark 4 个主要文件不同指令集生成的二进制代码长度。

Table 7. The length of binary code generated from CoreMark’s main files (in bytes)
表 7. CoreMark 主要文件生成的二进制代码长度(字节)

目标文件	Cortex-M4	RV32i	RV32im	RV32imc
core_list_join.o	988	1864	1884	1224
core_matrix.o	760	1956	1356	868
core_state.o	622	1112	1104	732

core_util.o	174	308	284	178

从表 7 可见，RV32i 程序的平均长度是 Cortex-M4 的 2.05 倍，RV32imc 程序的平均长度是 Cortex-M4 的 1.2 倍。可见，在将程序从 Cortex-M 移植到 rv32 时，选择 RV32imc 指令集组合，将基本满足原系统对存储资源限制要求。

添加指令集模块“A”，选择 RV32imac 指令集组合，编译后主要文件二进制代码长度与 RV32imac 完全相同，CoreMark 得分 2.34/MHz，与选择 RV32imc 指令集组合时相近。

4. 过程调用规范

过程调用规范(Procedure Call Standard)定义了应用程序二进制接口(Application Binary Interface)，通常包括函数或过程调用中参数传递和结果返回方式，处理器寄存器使用，以及数据类型处理等内容。将 Cortex-MMCU 应用程序移植到 RV32 MCU 时，需要考虑 ARM 处理器和 RISC-V 处理器过程调用规范之间的差异。本节将讨论函数调用过程参数传递方式的差别对移植程序带来的影响。

ARM 架构过程调用规范(AAPCS)[24]约定：在函数和过程调用过程中，调用者(主程序)通过 4 个寄存器 r0-r3 或称为 a0-a3，向函数(被调用者)传递参数。如果参数超过 4 个寄存器数值范围，超出部分利用栈传递；函数通过 a0-a1 返回结果。RISV-V 过程调用规范(RISC-V Procedure Calling Convetion)[25]约定：调用者通过 8 个寄存器 x10-x17 或称为 a0-a7，向被调用者传递参数。如果参数超过 8 个寄存器数值范围，超出部分利用栈传递；被调用者利用 a0-a1 返回结果。表 8 列出了 6 个整数型参数 C 语言函数编译后所生成的 Cortex-M 和 RV32imac 汇编函数。

Table 8. The generated assembly functions from C function
表 8.由 C 函数生成汇编函数

C 函数	Cortex-M4 汇编	RV32imac 汇编

int add6(int p1, int p2, int p3, int p4,
int p5, int p6)
{
return(p1 + p2 + p3 + p4 + p5 + p6);
}
	.global add6
	.thumb
	.type add6, %function
	add6:
	adda0, a0, a1
	add a0, a0, a2
	add a0, a0, a3
	ldr a3, [sp]
	add a0, a0, a3
	ldr a3, [sp, #4]
	add a0, a0, 3
	bxlr
		.global add6
		.type add6, %function
		add6:
		adda0, a0, a1
		add a0, a0, a2
		add a0, a0, a3
		add a0, a0, a4
		add a0, a0, a5
		ret

如表 8 汇编函数所示，Cortex-M4 函数，参数 1 到参数 4 通过 a0-a3 传递，参数 5 和 6 通过栈传递。在 RV32imac 函数中，参数 1-6 通过 a0-a5 传递。Cortex-M4 和 RV32imac 利用 a0 返回结果。

由于访问栈的延时高于访问寄存器，在设计 Cortex-MMCU 应用函数时通常使参数不超过 4*32 位。将 Cortex-MMCU 应用程序移植到 RV32 MCU 时，利用多达 8*32 位寄存器参数传递特性，将减少函数和过程调用带来的延时。

5. 总结与展望

本文从 MCU 中断处理机制，指令集模块组合，以及程序过程调用规范三方面比较 Cortex-M 和 RV32 MCU 的差别，分析了这些差别对将应用程序从 Cortex-M MCU 移植到 RV32 MCU 的影响。为了兼容中

断处理程序，将 RV32 MCU 设置为向量中断响应方式，并在中断服务程序中添加保存和恢复上文语句。在生成应用程序时选择 RV32imc 指令集组合，实现高性能和小体积的应用程序。利用 MCU 寄存器在函数和过程调用过程传递更多参数，降低调用过程中的延时。

RV32 MCU 与 Cortex-M MCU 指令差别很大，指令之间的差异对程序移植性能优化带来挑战。未来将进一步探讨 RV32 程序移植中的性能优化问题。

参考文献

[1] Waterman, A. (2016) Design of the RISC-V Instruction Set Architecture. Ph.D. Dissertation, University of California. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.html

[2] SiFive FE310-G002 Manual v1p5.https://www.sifive.com/

[3] RISC-V. https://www.gigadevice.com/product/mcu/risc-v

[4] 青稞 RISC-V 通用系列[EB/OL]. https://www.wch.cn/products/productsCenter/mcuInterfacecategoryId=70, 2024-01-10.

[5] 微控制器[EB/OL]. https://www.hpmicro.com/product/product.html?id=07e8d638-1c6c-44d1-9e53-a48d8560ad78, 2024-01-10.

[6] Reness Introduces Industry's First General-Purpose 32-bit RISC-V MCUs with Internally Developed CPU Core. https://www.renesas.com/us/en/about/press-room/renesas-introduces-industry-s-first-general-purpose-32-bit-risc-v-mcus-internally-developed-cpu-core

[7] Embedded-Studio.https://www.segger.com/downloads/embedded-studio/

[8] The Leading Commercial Tools for RISC-V. https://www.iar.com/products/architectures/risc-v/

[9] Eclipse Embedded CDT (C/C++ Development Tools). https://projects.eclipse.org/projects/iot.embed-cdt

[10] Singhal, S.P., Sridevi, M., Narayanan, N.S., *et al*. (2021) Porting of eChronos RTOS on RISC-V Architecture. In: Hura, G.S., Singh, A.K. and Siong Hoe, L.,Eds.,*Advances in Communication and Computational Technology*, Springer, 1269-1279.https://doi.org/10.1007/978-981-15-5341-7_96

[11] Pieper, P., Herdt, H. and Drechsler, R. (2022) Advanced Embedded System Modeling and Simulation in an Open Source RISC-V Virtual Prototype. *Journal of Low Power Electronics and Applications*, **12**,Article 52. https://doi.org/10.3390/jlpea12040052

[12] 在 RISC-V 微控制器上使用 FreeRTOS[EB/OL]. https://www.freertos.org/zh-cn-cmn-s/Using-FreeRTOS-on-RISC-V.html, 2024-01-13.

[13] 邰阳, 韩昌刚, 全雨, 于佳耕, 武延军. 基于 QEMU RISC-V 架构的 OpenHarmony 标准系统移植[J]. 计算机系统应用, 2023, 32(11): 21-28.

[14] Gordon, N., Pedretti, K. and Lange, J.R. (2022) Porting the Kitten Lightweight Kernel. Operating System to RISC-V.*IEEE/ACM International Workshop on Runtime and Operating Systems for Supercomputers* (*ROSS*), Dallas,13-18 November 2022, 1-7.https://doi.org/10.1109/ROSS56639.2022.00008

[15] Zhang, L.(2022) The Porting and Optimization of RISC-V UEFI Boot. 2022 7*th International Conference on Intelligent Computing and Signal Processing* (*ICSP*), Xi'an, 15-17 April 2022, 840-843. https://doi.org/10.1109/ICSP54964.2022.9778600

[16] Balas, R. and Benini, L. (2021) RISC-V for Real-Time MCUs—Software Optimization and Microarchitectural Gap Analysis. 2021 *Design, Automation & Test in Europe Conference & Exhibition* (*DATE*), Grenoble,1-5 February 2021, 874-877. https://doi.org/10.23919/DATE51398.2021.9474114

[17] Arm Holdings Market Share across Key Technology Markets Worldwide 2020-2022. https://www.statista.com/statistics/1132112/arm-market-share-targets/

[18] 21 世纪经济报道. RISC-V 快速成长，哪些应用领域落地更快更完整? [EB/OL]. https://new.qq.com/rain/a/20221130A0AATG00.html, 2022-11-30.

[19] ARMv7-M Architecture Application Level Reference Manual. https://www.arm.com/

[20] RISC-V Foundation, the RISC-V Instruction Set Manual, Volume I: Unprivileged ISA. https://riscv.org/

[21] GigaDevice, GD32VF103 User Manual V1.2. fhttps://www.gd32mcu.com/data/documents/userManual/GD32VF103_User_Manual_Rev1.5.pdf

[22] Coremark 1.0. https://github.com/eembc/coremark

[23] STM32 Cortex®-M4 MCUs and MPUs Programming Manual. https://www.st.com.cn/resource/en/programming_manual/pm0214-stm32-cortexm4-mcus-and-mpus-programming-manual-stmicroelectronics.pdf

[24] Procedure Call Standard for the ARM Architecture. https://eecs.umich.edu/courses/eecs373/readings/ARM-AAPCS-EABI-v2.08.pdf

[25] Understanding RISC-V Calling Convention. https://cs.sfu.ca/~ashriram/Courses/CS7ARCH/assets/notebooks/RISCV/RISCV_CALL.pdf

Embedded Technology and Intelligent Systems 嵌入式技术与智能系统,2024, 1(1), 31-39
Published Online August2024 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2024.11004

Hans汉斯

基于MEMS的无线数字地震检波器

陈家烨[1]，林熙鹏[2*]

[1]哈尔滨工业大学机电工程学院，黑龙江哈尔滨
[2]福建平潭旭坤实业有限公司，福建福州

收稿日期：2024年7月4日；录用日期：2024年7月25日；发布日期：2024年8月5日

摘要

介绍一种完全自主开发的体积小的基于MEMS无线数字地震检波器。该检波器主要包括以下几个部分：MEMS传感器板、放大采集板、FPGA控制主板、无线触发接收板、无线WIFI模块板和供电电源设计等。该传感器的设计涉及多个关键技术点，包括微弱信号获取、低功耗设计、总体结构的合理布置、高精度和高灵敏度、无线数据通讯以及无线触发信号接收等。应对这些关键技术点，本设计首先对芯片和材料精挑细选、对电路和结构进行合理设计，满足该检波器各项技术指标。最后通过一系列的室内测试和野外试验，验证了该检波器的各项功能和性能，为地震勘探提供高精度、高灵敏度、稳定可靠的地震检波器。

关键词

MEMS加速度计，无线数字检波器，信号处理，地震勘探

WirelessDigitalSeismometerBasedonMEMS

Jiaye Chen[1],Xipeng Lin[2*]

[1]School of Mechatronics Engineering,HarbinInstituteofTechnology,Harbin Heilongjiang
[2]FujianPingtanXukunIndustrialCo.,LTD.,Fuzhou Fujian

Received: Jul. 4th, 2024; accepted: Jul.25th, 2024; published: Aug.5th, 2024

Abstract

This paper presents the development of a compact, independently developed, MEMS-based wireless digital seismometer. The seismometer primarily comprises several key components: the MEMS sensor board, amplification and acquisition board, FPGA control board, wireless trigger receiver board, WIFI module board, and power supply design. The design of this sensor involves ad-

*通讯作者。

文章引用:陈家烨, 林熙鹏.基于 MEMS 的无线数字地震检波器[J].嵌入式技术与智能系统,2024,1(1):31-39.
DOI: 10.12677/etis.2024.11004

dressing multiple critical technical points, including weak signal acquisition, low power consumption design, rational overall structural layout, high precision and sensitivity, wireless data communication, and wireless trigger signal reception.To address these critical technical points, the design carefully selects the appropriate chips and materials and reasonably designs the circuitry and structure to meet the various technical specifications of the seismometer. A series of indoor tests and field experiments were conducted to verify the functions and performance of the seismometer, providing a high-precision, high-sensitivity, stable, and reliable seismometer for seismic exploration.

Keywords

MEMS Accelerometer, Wireless Digital Seismometer, Signal Processing, Seismic Exploration

1. 引言

近年来，随着计算机和电子技术的快速发展，并应用到地震勘探领域，同时也提高了探测技术和数据采集技术的水平。在交通领域[1][2]、建筑领域[3][4]、海洋领域[5][6]以及矿山领域[7][8]普遍存在着地震勘探，对我国甚至世界查找资源、地质普查以及安全预报做出卓越贡献。但是，用于分析地震勘探数据的理论和系统虽然已逐渐完善，但是，作为地震勘探中最关键的部件，地震检波器的发展较慢。目前，在国内应用得比较广泛的地震检波器种类是模拟地震检波器[9]，它是运用电磁感应的方式将地震波的振动信号转化为电信号并进行模拟传输。它存在精度不高、灵敏度低、频响范围小的缺点，满足不了日益发展的地震勘探的需要。在数字地震检波器的应用方面，国内尚处于起步阶段，主要由部分大学、检波器厂家和科研院所在进行相关的研发，但目前大都处于实验室研发阶段，还没有成熟的相关产品大范围地投入实际应用[10]。由此可见，开发一种高精度、高灵敏度、宽频带稳定可靠的地震检波器势在必行。

2. 地震检波器总体设计

2.1. 地震检波器组成

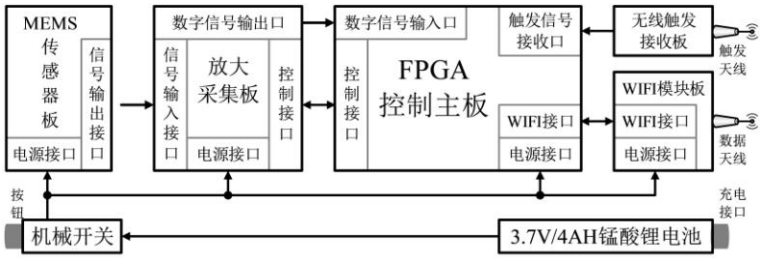

Figure 1. Block diagram of MEMS-based wireless digital seismic detector
图 1.基于 MEMS 的无线数字地震检波器结构框图

基于 MEMS 的无线数字地震检波器由 MEMS 传感器板、放大采集板、FPGA 控制主板、无线触发接收板、无线 WIFI 模块板和供电电源电路组成的。它的功能包括接收触发信号并感知地震信号，并把此

信号进行抗干扰处理后，转换成地震数字信号，存储该地震数字信号，同时可把地震数字信号以无线 WIFI 方式传输给与之关联的地震主机。另外，基于 MEMS 的无线数字地震检波器也可以通过无线 WIFI 接收地震主机传来的参数和命令。基于 MEMS 的无线数字地震检波器组成如图 1 所示。

2.2. 地震检波器设计的难点与对策

基于 MEMS 无线数字地震检波器设计的难点在于：1) 微弱信号的获取：微弱信号低至几百纳伏，而干扰信号与之相比要大得多，消除干扰信号尤为重要；2) 低功耗设计：本检波器野外连续工作时间长，而供电电池容量有限，必须采用低功耗设计；3) MEMS 传感芯片安装设计：要保证 MEMS 传感芯片获得高精度微弱信号，必须设计有与该 MEMS 芯片具有很好的耦合的固定机构；4) 传感器总体结构设计：合理设计线路板，布置各个部件得当，降低部件及器件间的相互干扰；5) 无线通讯设计：由于本检波器要求低噪声、低功耗、高精度和体积的小设计，致使无线传输功率比较小，设计时必须对传输距离和传输速率进行合理平衡。

针对以上难点，应对的方法有：1) 芯片选择低噪声，相关信号芯片必须是高精度、低温漂。设计有去除干扰信号专用电路。电源设计有 RC 滤波电路；2) 在充分满足性能和功能的要求下，选择低功耗芯片和功能模块，同时也采用低功耗电路设计，保证连续工作时间超过 12 小时；3) MEMS 传感芯片耦合结构设计：根据 MEMS 传感芯片形状大小在外壳内底部中心挖一个与该传感芯片大小相符的方坑，安装时把传感芯片紧密嵌入方坑，安装后使传感芯片紧密耦合传感器外壳底部，感知微弱信号；4) 总体结构设计：将传感器线路板分为三个部分：传感芯片及模拟信号处理模块，数字处理模块以及无线通讯模块，做到模拟电路、数字电路以及通讯高频电路分开，降低干扰信号。

2.3. 地震检波器工作原理

首先，本检波器和关联的地震主机联机后，接收地震主机传输的参数和命令，在 FPGA 控制主板的控制下，对放大采集板进行增益、采样间隔、采样点数以及各部件初始化设置，之后进入等待接收无线触发接收板传来的触发信号。当接收到触发信号时立即启动采集工作，MEMS 传感器板感知地震信号变为电信号，此信号经过放大采集板的放大、滤波和缓冲放大电路的信号调理，并在 FPGA 控制主板的控制下，通过放大采集板 24 位 AD 采集电路转换为数字信号并存储，同时利用无线 WIFI 模块板把所采集的地震数据传输给地震主机进行存储、处理和显示。

2.4. 地震检波器的技术指标

Table 1. Technical specifications of MEMS-based wireless digital seismic detector
表 1.基于 MEMS 无线数字地震检波器技术指标

性能指标	技术参数	性能指标	技术参数
处理器(FPGA)	120MHz	存储容量(GB)	8
采集方式	单次采集	触发方式	外或内信号触发
采集长度(dot)	2^k(k=9、10…18)	采集间隔(uS)	8×n(n=1、2、…、1000)
采集精度	24 位	动态范围(dB)	130
通讯协议	遵循标准 WIFI 协议	数据通讯(Kbps)	1000
物理量	加速度	灵敏度	1000mV/g
轴数	三轴	测量范围	±1.7g
频率范围	0.4Hz~1.6KHz	非线性误差	±0.2%
噪声密度	200μg/√Hz	尺寸	75mm×75mm×95mm

基于 MEMS 无线数字地震检波器整体技术指标如表 1 所示。

3. 地震检波器硬件设计

3.1. MEMS 传感器板

MEMS 传感器板核心芯片选择美国模拟器件公司(ADI)生产的 ADXL203 加速度芯片，它是一款具有高精度、高灵敏度、低噪声、低功耗的 iMEMS 型双轴加速计，性价比极高，其基本性能完全满足本检波器设计的要求。由于 ADXL203 芯片只有两轴，满足不了三轴的要求，现采用两片 ADXL203，通过结构和电路上的合理设计，完全可以满足三轴的要求，其结构设计示意图如图 2 所示。

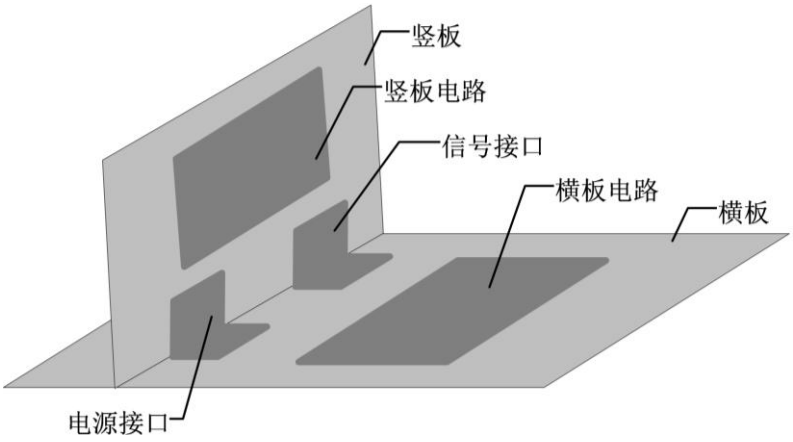

Figure 2. Schematic of triaxial detector using two ADXL203 chips
图 2.两个 ADXL203 芯片实现三轴检波器结构示意图

在图 2 中，MEMS 传感器板由两块板组成的，一块 ADXL203 板横向布置，简称横板，另一块 ADXL203 板纵向布置，简称竖板。其中，横板规定为 X 轴方向、Y 轴方向采集信号，竖板规定为 Z 轴方向采集信号，这样就可完成三轴功能。无论是横板，还是竖板，ADXL203 传感器的基本电路原理图如图 3 所示。

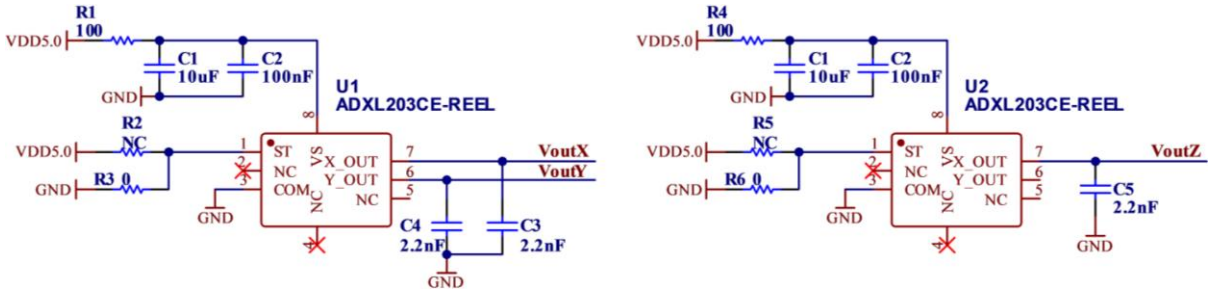

Figure 3. Basic circuit diagram of the ADXL203 sensor board,(a) horizontal board schematic,(b) vertical board schematic
图 3. ADXL203 传感器板基本电路原理图，(a)横板原理图，(b)竖板原图

3.2. 放大采集板

放大采集板电路包括前置放大电路和 A/D 采集电路。前置放大电路 1 的前端是 RC 差模共模电路，对 MEMS 传感器板 X 轴输出的振动信号进行抗干扰滤波，后经过放大及二阶低通滤波进行信号放大滤波后，再经过单端转差分滤波匹配电路输出到 AD 采集电路 1 转换为数字信号存储在 FPGA 控制主板存储器中。放大采集板电路原理图如图 4 所示。

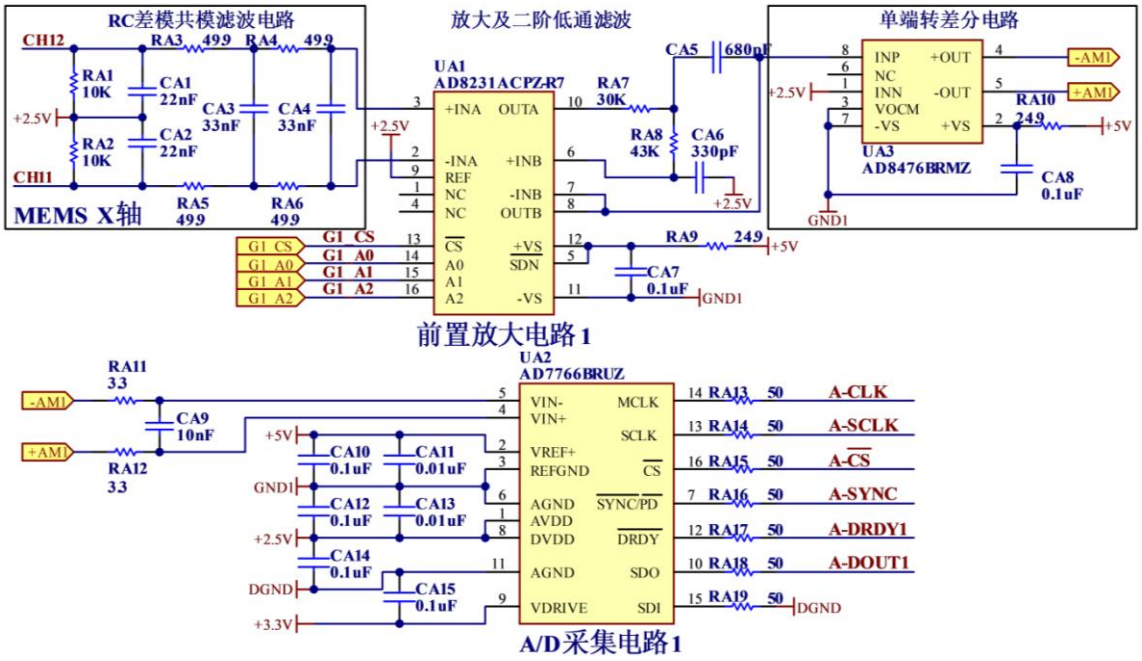

Figure 4. Basic circuit diagram of the amplification and acquisition board
图 4.放大采集板基本电路原理图

在图 4 中，放大器选择美国模拟器件公司(ADI)生产的一款高精度、低噪声、低失调以及低功耗的仪表放大器 AD8231，增益×1、×2、×4、×8、×16、×32、×64、×128，并可编程控制。采集器选择美国模拟器件公司(ADI)生产的一款高精度、低噪声、低功耗的模数转换器 AD7766，精度 24 位，采样率 125KHz。这两款芯片完全满足基于 MEMS 无线数字地震检波器设计要求。

3.3. FPGA 控制主板

FPGA 控制主板核心芯片选择美国 Altera 公司生产的 CycloneⅣ系列 EP4CE22F17C8。主要包括 EP4CE22F17C8 控制器、FLASH 存储电路、模数转换控制接口、放大器增益控制接口、无线触发接收板接口、WIFI 模块板接口、电源电路以及 3.7V 电池输入接口，其组成如图 5 所示。从图中可以看出，除了电源部分，其余均由 FPGA 芯片 EP4CE22F17C8 控制。

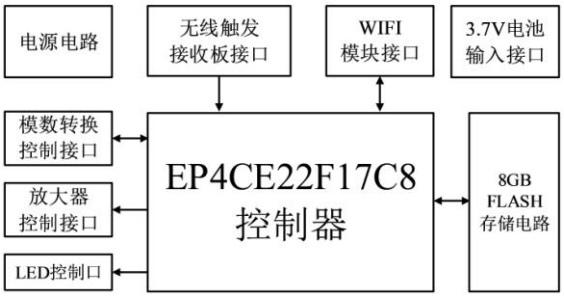

Figure 5. Block diagram of the FPGA control mainboard
图 5.FPGA 控制主板基本组成框图

3.4. 无线触发接收板

无线触发接收板核心是无线触发接收模块，其接收到的触发信号经过放大电路、绝对比较电路和触

发混合信号输出电路后，FPGA 控制主板接收到触发信号，启动采集地震信号。触发信号是模拟信号，传输极快，从发出触发信号至采集开始这段时间极短，不超过 20uS，这为不同检波器的同步采集提供至关重要的基础。无线触发接收板基本原理图如图 6 所示。

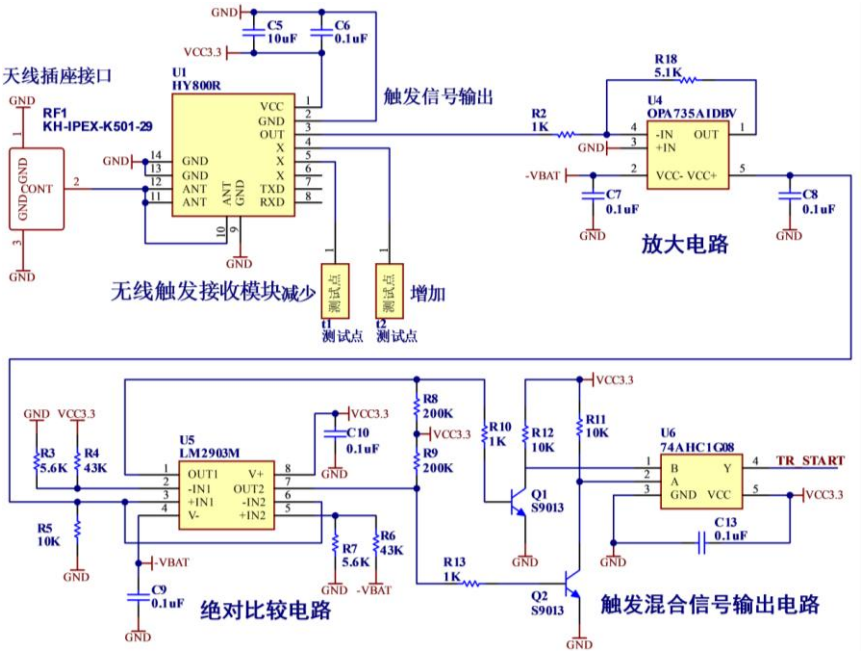

Figure 6. Basic circuit diagram of the wireless trigger receiver board
图 6.无线触发接收板基本电路原理图

3.5. 无线 WIFI 模块板

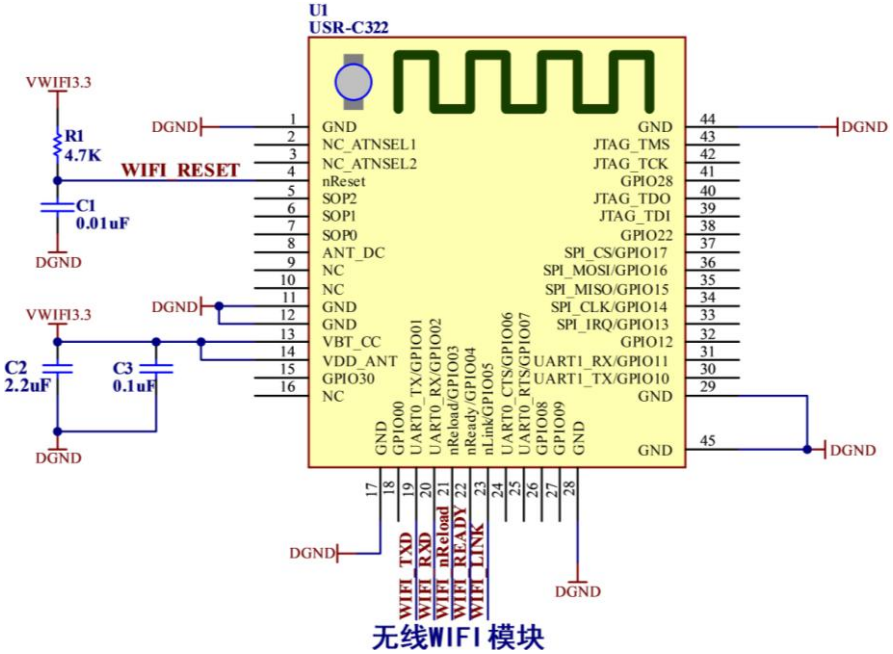

Figure 7. Basic circuit diagram of the wireless WIFI module board
图 7.无线 WIFI 模块板基本电路原理图

无线 WIFI 模块板核心是 WIFI 模块，完全遵循国际标准的 WIFI 协议，在 FPGA 控制主板的控制下，负责与相关的地震主机进行数据传输，传输速率 1000Kbps。无线 WIFI 模块基本原理图如图 7 所示。

3.6. 供电电源设计

电源供电部分采用适配器和内部电池双供电，当在野外施工时，适配器不能供电，内部电池开始供电。内部电池工作时间一般不超过 12 个小时，电源设计示意图如图 8 所示。

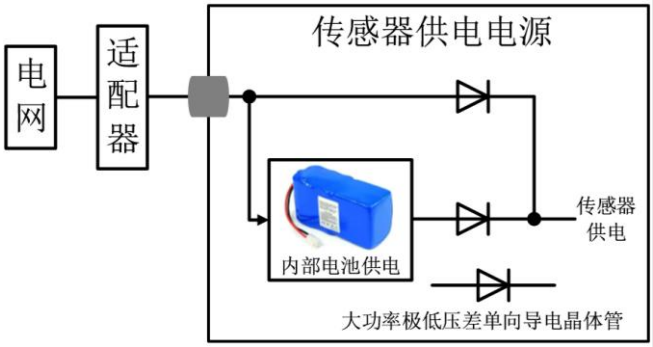

Figure 8. Block diagram of the power supply design
图 8.供电电源设计框图

4. 地震检波器软件设计

基于 MEMS 无线数字地震检波器软件设计主要包括初始化程序编程、信号处理程序编程(包括放大增益设置程序和 DSP 滤波程序)、数据采集控制编程、数据存取编程(FLASH 读取控制程序)以及无线通讯程序编程。当基于 MEMS 无线数字地震检波器工作时，其整体地震信号数据采集控制程序编程框图如下图 9 所示。

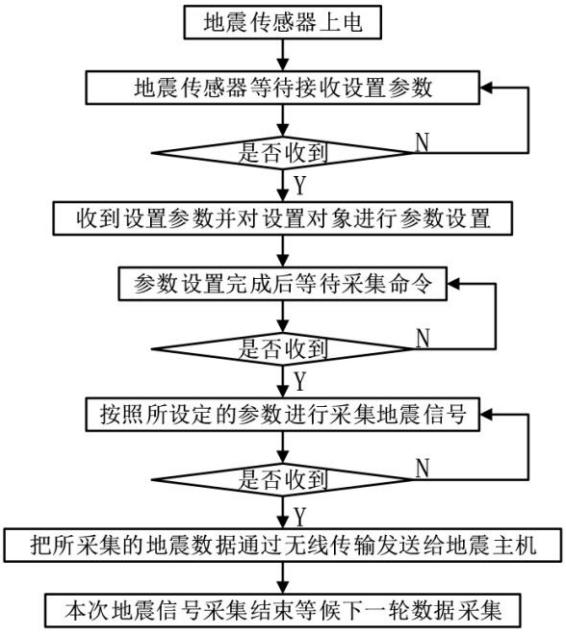

Figure 9. Block diagram of the seismic signal data acquisition design
图 9.地震信号数据采集设计框图

5. 地震检波器测试

为了验证设计的基于 MEMS 无线数字地震检波器的性能和功能是否能达到预期要求，进行了室内测试(如图 10 所示)和野外测试(如图 11 所示)来验证所设计的地震检波器是否能正确接收到地震波。

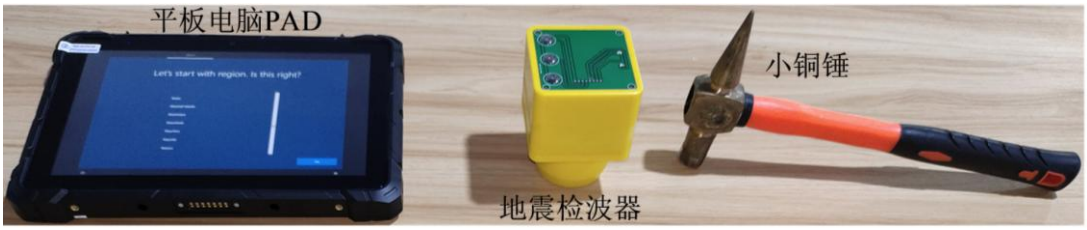

Figure 10. Schematic of indoor testing
图 10.室内测试示意图

Figure 11. Schematic of field experiment
图 11. 野外实验示意图

图 12 所示是在室内用相同大小的震源，分别在距离 0.3m、0.6m 以及 0.9m 处的振动试验得到的时域波形。

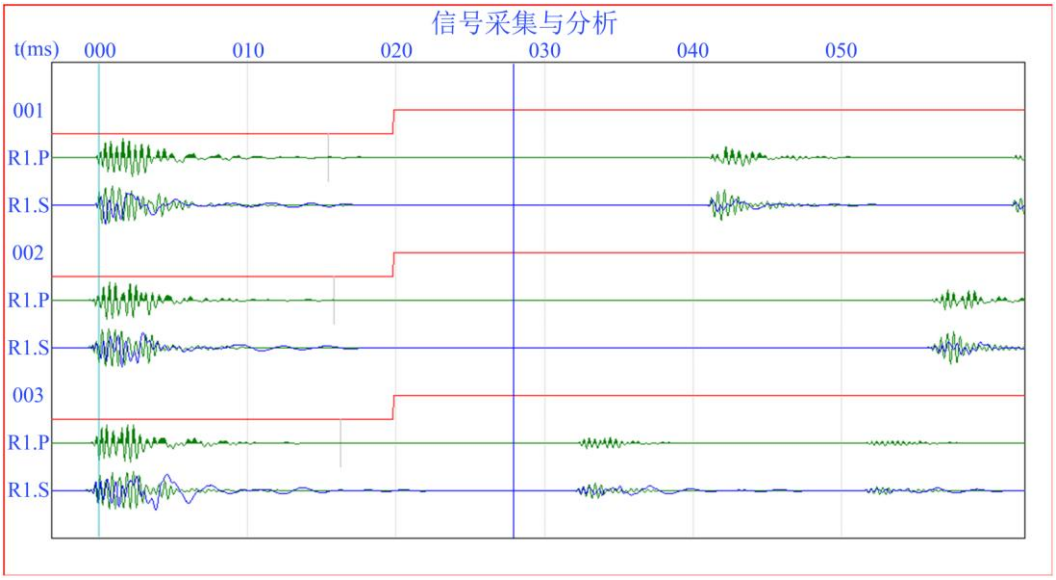

Figure 12. Time domain waveform of seismic waves
图 12.地震波时域波形

从室内和室外对本设计的基于 MEMS 无线数字地震检波器测试结果分析，其功能和性能满足设计的要求。

6. 结论

本文详细介绍并设计了一款基于 MEMS 无线数字地震检波器的硬件设计与实现，系统主要包括 MEMS 传感器板、放大采集板、传感器主板、无线触发接收板、无线 WIFI 模块板和供电电源设计等部分。设计的关键技术点如下：

1)微弱信号获取：ADXL203 传感器能够感知微弱的地震波并输出低至 1μV 的信号，因此在设计过程中，信号的获取和干扰的排除成为主要挑战。通过优化芯片选型、线路板的合理设计和合理的摆放结构，提高了信号的准确性和可靠性。

2)低功耗设计：为实现电池供电 12 小时持续工作，系统采用了低功耗设计策略。AD8231 仪表放大器、AD8476 缓冲放大器和 AD7766 模数转换器的选择，不仅保证了高精度、低噪声和低功耗，还在不同环境下具备稳定的性能表现。

3)高精度和高灵敏度：系统采用高精度放大器和 24 位高分辨率模数转换器，实现了对微弱地震波信号的准确捕捉和数字化处理。这样设计确保了信号的高灵敏度和数据采集的高效率。

4)无线通讯：利用 WIFI 模块进行信号的无线传输，增强了系统的便携性和实用性，适应了现代地震勘探的需求。

综上所述，本系统针对现代地震勘探中对高精度、高灵敏度和低功耗的需求，设计并实现了一套基于 MEMS 无线数字地震检波器。其创新点在于解决了微弱信号的获取与传输、低功耗设计及高效无线通讯等技术难题，具有广泛的应用前景。

参考文献

[1] 陈支兴.多源频率域地震勘探法在探测城市隧道洞室中的应用[J].隧道建设(中英文),2023, 43(9): 1583-1589.

[2] Lavoué, F., Coutant, O., Bouéé, P., Pinzon-Rincon, L., Brenguier, F., Brossier, R., *et al.* (2020) Understanding Seismic Waves Generated by Train Traffic via Modeling: Implications for Seismic Imaging and Monitoring. *Seismological Research Letters*, **92**, 287-300. https://doi.org/10.1785/0220200133

[3] 杨道林. 工程建设场地地震安全性评价应用研究[D]: [硕士学位论文].合肥: 安徽建筑大学,2014.

[4] 孟立朋,彭远黔,冉志杰,等.浅层地震勘探在工程选址中的应用及断层活动性鉴定[J].华北地震科学,2016, 34(4): 20-27.

[5] Fernando, B., Leng, K. and Nissen-Meyer, T. (2020) Oceanic High-Frequency Global Seismic Wave Propagation with Realistic Bathymetry. *Geophysical Journal International*, **222**, 1178-1194. https://doi.org/10.1093/gji/ggaa248

[6] 杜万兴.应用于海洋地震勘探的震源技术探讨[J].石化技术,2024, 31(5): 307-309.

[7] Cheng, J.Y., Jiang, H., Ji, G.Z. and Wu, H.(2015) Channel Wave Seismic Exploration Technology Based on Node Digital Seismograph in Underground Mine. *Coal Science & Technology*,**43**,25.

[8] 李江.花园煤矿床五采区岩浆岩分布地震预测[J].矿产勘查,2024, 15(5): 811-817.

[9] 段疾病.基于串行通信的地震检波器综合测试系统的设计实现[D]: [硕士学位论文].青岛: 中国海洋大学,2013.

[10] 罗忠琴,刘鹏,唐建益,等.煤矿隐蔽致灾因素地震勘探现状与发展方向[J].中国煤炭,2023, 49(1): 16-29.

Embedded Technology and Intelligent Systems 嵌入式技术与智能系统,2024, 1(1), 40-49
Published Online August 2024 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2024.11005

Hans汉斯

在MCU端部署GRU模型实现鼾声检测

许鹏*，王印鑫，宋岩

恩智浦半导体有限公司，荷兰埃因霍温

收稿日期：2024年7月4日；录用日期：2024年7月25日；发布日期：2024年8月5日

摘要

本研究旨在开发一种在资源受限的微控制器单元(MCU)上运行的方法，用以进行鼾声检测。不同于使用CNN进行声音检测的方式，我们采用门控循环单元(GRU)模型以对音频数据进行处理和分析。通过采用优化模型结构、模型量化等常用的模型优化方式，我们最终成功将GRU模型适配到低功耗的MCU平台，使其能够在不依赖外部计算资源的情况下，独立完成端侧的鼾声检测任务，无需联网。实验结果表明，该模型在保持较高准确性的同时，能够有效降低系统算力需求，满足移动健康监测设备的实时性与便携性要求。这一研究为鼾症患者的持续监测和睡眠健康管理提供了一种新的解决方案，同时也拓展了深度学习在嵌入式系统中的应用前景。

关键词

微控制器单元(MCU)，门控循环单元(GRU)，鼾声检测，人工智能应用

Deploying GRU Model on MCU for Snore Detection

PengXu*, YinxinWang, YanSong

NXP Semiconductors, Eindhoven, The Kingdom of the Netherlands

Received: Jul. 4th, 2024; accepted: Jul.25th, 2024; published: Aug.5th, 2024

Abstract

This study aims to develop a method running on a resource-constrained microcontroller unit (MCU) for snore detection. Unlike the approach of using CNNs for sound detection, we employ the Gated Recurrent Unit (GRU) model to process and analyze audio data. By adopting common model

*通讯作者。

文章引用:许鹏, 王印鑫, 宋岩.在 MCU 端部署 GRU 模型实现鼾声检测[J].嵌入式技术与智能系统,2024,1(1):40-49.DOI: 10.12677/etis.2024.11005

optimization techniques such as optimizing the model structure and model quantization, we ultimately succeeded in adapting the GRU model to a low-power MCU platform. This allows it to independently perform snore detection tasks on the edge without relying on external computing resources and without the need for internet connectivity. Experimental results indicate that while maintaining high accuracy, the model can effectively reduce the system's computational power requirements, meeting the real-time and portability needs of mobile health monitoring devices. This research provides a new solution for the continuous monitoring and sleep health management of patients with snoring disorders and also expands the application prospects of deep learning in embedded systems.

Keywords

Microcontroller Unit (MCU), Gated Recurrent Unit (GRU), Snore Detection, Artificial Intelligence Application

1. 引言

提起打鼾，可能大家都不会陌生，打鼾是睡眠呼吸障碍的一种常见表现，普遍存在于全球各地，影响着数百万人的生活质量和健康。可以说，打鼾是一个极端令人烦恼的生理现象，验证影响睡眠质量。对于许多患者而言，及时检测打鼾并采取相应措施至关重要。然而，尽管市场上已有多种方法用于鼾声检测，包括使用各种监测设备和进行复杂的睡眠研究，但这些方法大多数依赖于昂贵的设备和专业的操作，这在资源有限的环境中并不实用，也难以普及到广大需要的人群，就显得有些空中楼阁了。因此，开发一种简便、低成本且可靠的鼾声检测方法具有重要的实际意义，可以极大地提高普及度以及大众的认可读[1]。

近年来，人工智能在处理和分析声音数据方面取得了显著进展，尤其是深度学习技术已被成功应用于各种声音识别任务，例如语音识别和异常声音检测。正是这些技术的进步，为更加准确、可靠的鼾声检测提供了新的可能性。在 MCU 单元上实现深度学习模型的尝试，为端侧的计算和边缘智能开辟了新的可能性，使得在低成本的硬件设备上部署复杂的算法变得可行[2]。

本研究旨在探索使用深度学习架构——门控循环单元，并对其进行轻量级设计，以实现对鼾声的高效检测。GRU 模型能够捕捉到时间序列数据中的依赖关系，如鼾声的音频模式，使得其更加适合实时处理和分析。选择 MCU 单元作为运行平台，不仅因为其成本低和易于集成到便携式设备中，而且因为它适合于在资源受限的环境中操作，满足在各种环境下对鼾声进行实时监测的需求。

通过将 GRU 模型适配到基于 MCU 的呼吸辅助装置，我们可以将鼾声检测系统带入患者的日常生活环境中，从而实现实时、非侵入性的监测，提供即时反馈和必要的医疗干预。此外，该方法的发展也将推动深度学习在嵌入式系统中的应用，特别是在处理生理信号和促进移动健康监测方面，为广泛的生物医学应用开辟新的道路。本文接下来将详细介绍 GRU 模型的训练和调优过程，以及如何实现在微控制器单元上部署该模型。

2. GRU 模型简介

GRU 全称 Gated Recurrent Unit，直译叫做门控回归单元。是一种用于处理序列数据的循环神经网络(RNN)模型。RNN 指代递归神经网络(Recurrent Neural Network)，是一种常用于处理序列数据的神经网络架构[3]。与传统的前馈神经网络不同，RNN 具有记忆功能，非常适合于对序列数据进行处理，这是通过计算和维护状态变量来实现的。相较于传统的 RNN，GRU 具有更简单的结构和更高效的训练方式。它通过一种称为门控机制的方式来控制信息的流动，包括更新门和重置门。这些门控机制有助于模型决定在每个时间步上应该记住什么信息，以及应该忘记什么信息，从而更好地处理长序列数据。GRU 模型包括一个更新门和一个重置门。更新门决定了新的输入应如何保留，而重置门则决定了状态信息应如何忽略[4]。

图 1 是一个 GRU 的框图说明。

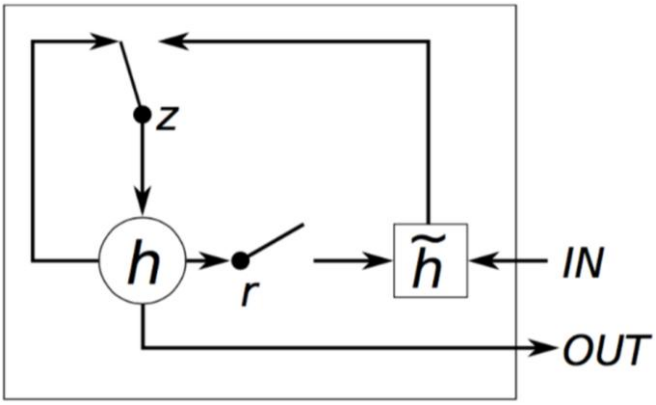

Figure1. GRU model. The switch is soft, is a weighted sum of the inputs, the sum of weights is 1
图 1. GRU 模型示意图。开关其实是软的，是彼此两个输入的带权和，且彼此权重的和为 1

接下来介绍下这两个门控单元的工作流程。首先是重置门 r，负责对状态量 H 进行逐点相乘，记作 $r \cdot H$ 。r 和 H 都是相同维度的向量，相乘的结果起到对记忆的信息筛选淡出的效果。之后将输入和计算后的状态结合成为新的候选状态 $\tilde{h}$ 。最终将结果进行更新，更新就是通过这个 z 门。计算方式是 $z \cdot H_{t-1} + (1-z) \cdot \tilde{H}$ ，也就是说将上一次计算的状态与当前的候选状态 $\tilde{H}$ 进行加权求和，得到本次的最终状态，当然也就是下一次的 H_{t-1} 。如此循环往复，直到训练结束，得到最终的模型[5]。

此外，与长短时记忆网络(LSTM)相比，GRU 模型减少了参数数量，一些情况下更容易训练，并且在计算上也更高效。这使得 GRU 成为处理序列数据时的一种流行选择。由于 GRU 模型的门控结构，它具有一定的记忆能力，能够更好地捕捉时间序列中的重要特征，并且相对于传统的 RNN 模型，GRU 模型在一定程度上缓解了梯度消失的问题，从而更适合处理长序列数据。

3. 在 PC 上进行 GRU 模型的训练与测试

3.1. GRU 模型训练

接下来，我们来看看如何训练一个 GRU 模型，模型训练平台选用 Keras，有需要的读者请自行安装 Keras 开发工具。我们这里主要给大家介绍模型搭建部分，这里假设我们的鼾声检测数据集已经准备好了，并将其划分为训练和测试数据集，分别命名为：(x_train, y_train), (x_test, y_test)。直接进行模型构建与训练：

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import GRU, Dense
# 构建 GRU 模型
model = Sequential()
model.add(GRU(128, input_shape=(64, 64), stateful=False, unroll=False))
model.add(Dense(2, activation='softmax'))
# 编译模型
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# 模型训练
model.fit(x_train, y_train, batch_size=128, epochs=10, validation_data=(x_test, y_test))
```

先看下这里所指定的输入，(64, 64)。怎么理解呢？上文讲过 GRU 模型实际上是每次只输入一个维度的数据，即数据应该是(1, 64)。这里好像有点出入。这就不得不解释下，这里的两个 64 分别代表什么。前一个 64 表示 timestep，第二个 64 表示音频特征的维度。那么就可以理解为：一次训练，我们是一次性对 64 组连续的语音特征进行训练，这样能够更好的处理他们之间的相关性。这里的音频特征是由一段固定长度时域数据提取而来。

这里需要注意的是，GRU 模型构建的时候，有两个参数，分别是 stateful 以及 unroll，这两个参数是什么意思呢：

stateful 参数：当 stateful 设置为 True 时，表示在处理连续的数据时，GRU 层的状态会被保留并传递到下一个时间步，而不是每个 batch 都重置状态。这对于处理时间序列数据时非常有用，例如在处理长序列时，可以保持模型的状态信息，而不是在每个 batch 之间重置。默认情况下，stateful 参数为 False。需要注意的是，若设置 stateful 为 True，需要手动管理状态的重置。

unroll 参数：默认情况下，unroll 参数为 False。当 unroll 设置为 True 时，表示在计算时会展开 RNN 的循环。通常情况下，对于较短的序列，unroll 设置为 True 可以提高计算速度，但对于较长的序列，可能会导致内存消耗过大，例如，上述模型如果 unroll=True，会展开 64 次相同的 GRU 所执行的操作，模型中包含大量节点。

我们再看看 stateful 参数。鼾声信号其实是时域音频信号，每个时间步之间实际上是存在有时间前后关系的，即前后的音频片段之间会组成一段完整的音频数据。因此，在实际使用时若要一次只处理单个时间步的数据，就需要设置 stateful 为 True。

3.2. GRU 模型测试

上面通过训练，得到了一个庞然大物。之所以庞大，是因为我们采用了 unroll 参数将其展开，发现模型其实由很多相同的模型块组成，GRU 模块如图 2 所示。

而这个模型块的数量和所设置的时间步是相关的。这里我们要给大家带来 GRU 模型的一个特殊性，那就是实际推理所用到的模型和训练时是不一样的。具体而言，训练时候，我们为了让 GRU 模型能够更好的学习相邻音频之间的关系，需要将模型输入设置为(64, 64)。而实际测试时候，由于模型已经具备了处理相邻音频的能力，我们也就不再需要将模型输入设置为(64, 64)，而是将其设置为(1, 64)，即每次只输入给模型一条音频特征，同时我们设置 stateful=True(请注意，与推理时相反，训练的时候 stateful 为 False！)，即每次模型会记录下当前所计算出来的状态信息，将其用于下一次的计算。这样化整为零，使输入减少到了以前的 1/64。隐藏状态的计算量也就按比例一起减少了。但要注意的是，最后一步用于把隐藏状态映射成输出的 Dense 层的计算次数却按比例增加了。不过，与减少的执行 GRU 模块的计算相比，

增加的计算比减少的计算要少得多。

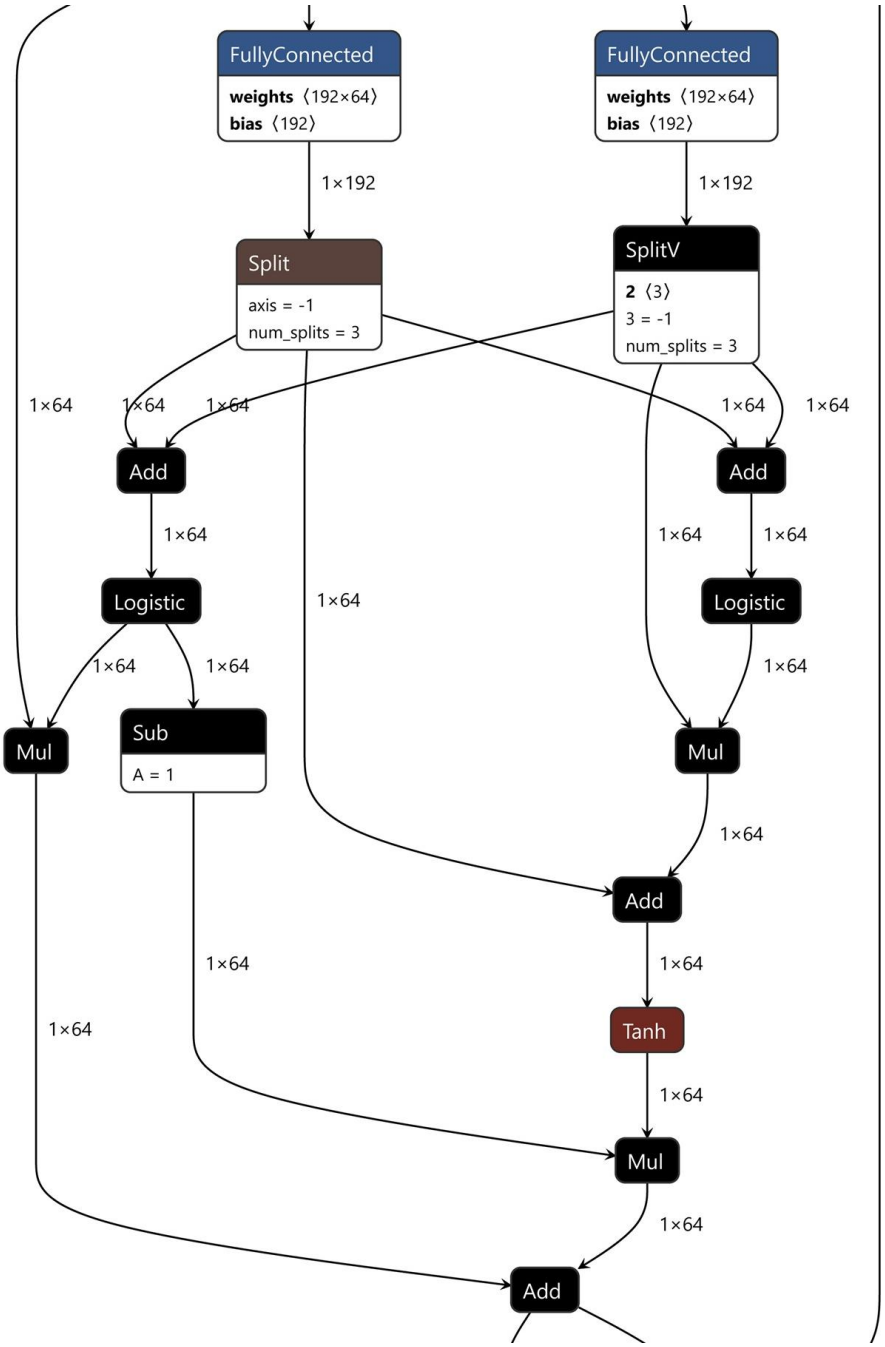

Figure 2. GRU model block
图 2.GRU 模型块

用于推理的模型构建代码修改如下：

```
# 构建新模型
new_model = Sequential()
new_model.add(GRU(1, batch_input_shape=(1, 1, 64), unroll=True, stateful=True))
new_model.add(Dense(2, activation='softmax'))
```

new_model.set_weights(model.get_weights())

注意到，最后一行，有一个 set_weights 函数，目的是将之前我们所训练出来的模型参数配置给这个新模型。看一下模型的变化，如图 3 所示。

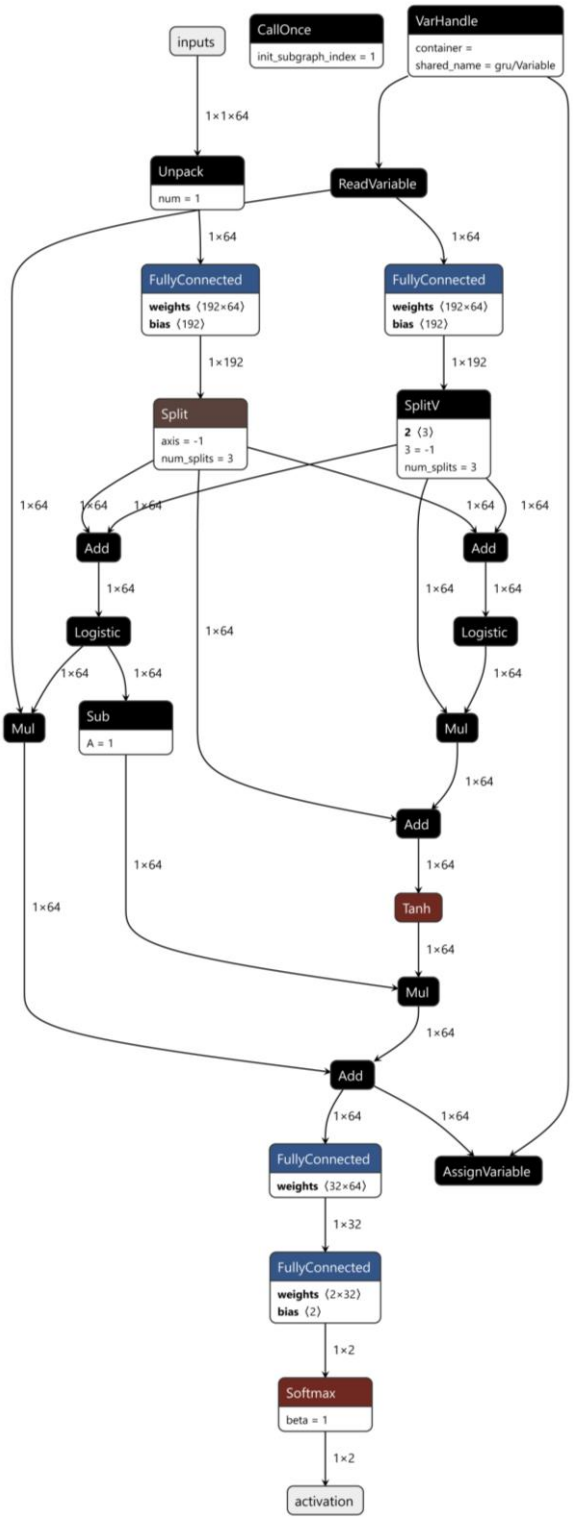

Figure 3. GRU model with time-step 1
图 3.时间步为 1 的 GRU 模型

图中红圈位置有一个 AssignVariable 节点，它的作用就是将本次所运行出来的状态量进行保存，以供下次推理使用。

3.3. GRU 模型量化

在模型部署前需要对模型进行量化，量化的原理就是将用浮点数表示的模型权重，映射为(−128, 127)之间的 8 位整数[6]。这样做的好处是可以减小模型尺寸，可以将其缩减到之前的近 1/4。再者，MCU 平台提供了对于具有 8 位整数类型权重的模型的推理加速能力。不过弊端就是，精度可能会有所下降，所幸的是 GRU 模块大量使用有上、下确界的 Sigmoid 和 Tanh 激活函数，有助于估计中间结果的值域。模型量化可以使用 NXP 公司所开发的 eIQ 工具，打开 eIQPortal 软件，找到 MODELTOOL，如图 4 所示：

Figure 4.eIQ Portal tools
图 4. eIQ Portal 工具

用它打开我们刚才重新制作的模型，点击左上角的 Convert 选择转换工具为 Tensor Flow Lite，并勾选 EnableQuantization，如图 5 所示。

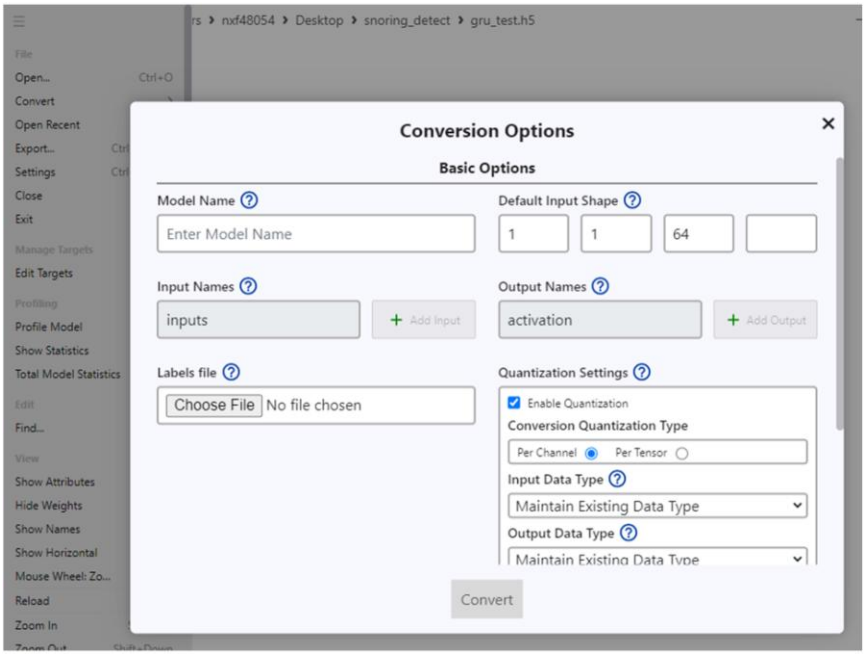

Figure 5. Model quantization
图 5.模型量化页面

点击 Convert 即可获得量化后的模型。

4. GRU 模型的在端侧的部署

我们选用了 NXP 的 i.MX RT1060 跨界 MCU 部署 GRU 模型。

4.1. i.MX RT1060 平台简介

i.MX RT1060 跨界 MCU 拥有 600MHz 的主频，可以提供卓越的计算能力、多种媒体功能以及实时功能，易于使用。i.MX RT1060 采用主频达 600MHz 的 Cortex®-M7 同时提供一流的安全保障。i.MX RT1060 MCU 支持宽温度范围，适用于消费电子、工业和汽车市场。拥有高达 1MB 片上内存，包括 512KB 的 FlexRAM，能被灵活的分配为 ITCM/DTCM/OCRAM。集成了高级 DCDC 和 LDO 的电源管理模块，简化电源序列。同时提供了多种内存接口，包括 SDRAM，RAWNAND FLASH，NORFLASH，SD/eMMC，Quad/Octal SPI，Hyper RAM/Flash 以及其他接口，可以连接诸如 WLAN，蓝牙，GPS，displays 以及摄像头传感器。同时具有丰富的音视频特性，包括 LCD 显示，显示加速器，摄像头接口，SPDIF 以及 I2S 音频接口。

4.2. 基于 SDK 中 eIQTFLm 工程进行模型集成

首先需要指出的是，GRU 模型相较于一般基于 CNN 的模型相比，其模型部署方法是一致的。特别地，eIQ MODEL TOOL 会把 GRU 模块降解成由最基本的 TensorFlowLite 所支持的算子所组成的模型结构。这样一来，就可以使用我们的 TensorFlowLite 引擎进行推理了。而对应到 MCU 上就是使用 tflite-micro 推理引擎进行推理。Tflite-micro 使用一种算子注册机制决定哪些算子的实现将被链接到最终生成的可执行映像中。这决定了为了将模型部署到 MCU 上，需要根据模型所需节点来修改注册算子的源文件。这里我们已 eIQ 中 label_image 的例子作为基础进行修改。找到工程中的 model_ds_cnn_ops_micro.cpp 文件，它就是刚才提到的注册算子的源文件，原来的例子包含了推理 label_image 所需的算子，我们对其进行修改如下：

```
tflite::MicroOpResolver&MODEL_GetOpsResolver()
{
static tflite::MicroMutableOpResolver<15>s_microOpResolver;
s_microOpResolver.AddAdd();
s_microOpResolver.AddAssignVariable();
s_microOpResolver.AddCallOnce();
s_microOpResolver.AddFullyConnected();
s_microOpResolver.AddLogistic();
s_microOpResolver.AddMul();
s_microOpResolver.AddReshape();
s_microOpResolver.AddReadVariable();
s_microOpResolver.AddSub();
s_microOpResolver.AddSplit();
s_microOpResolver.AddSplitV();
```

```
s_microOpResolver.AddSoftmax();
s_microOpResolver.AddTanh();
s_microOpResolver.AddUnpack();
s_microOpResolver.AddVarHandle();
return s_microOpResolver;
}
```

修改好之后，将模型的二进制数据转换成符合 C 数组定义格式的文本形式(例如，使用 xxd 工具)，并替换工程中 model_data.h 里面所包含的原始模型，就完成了所有关于模型的准备工作。当然对于数据预处理部分需要编写对应的 MIC 采集以及特征计算部分，限于篇幅本文就不再展开了。有兴趣的读者可以联系作者。

5. 实验验证

选择 i.MX RT1060 EVK 作为试验平台，整体组装方案以及 UI 设计如图 6 所示：

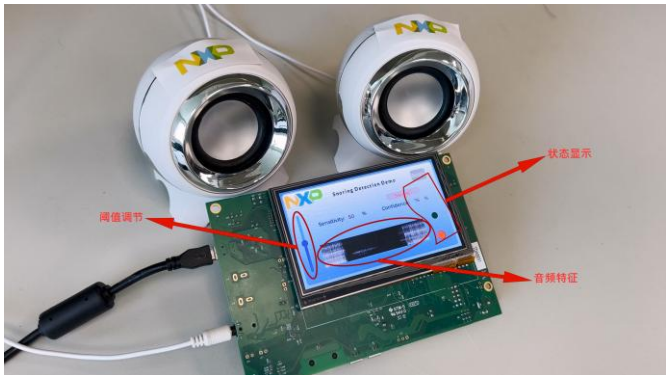

Figure 6. Overall architecture
图 6.整体组装方案

UI 页面主要分为三大显示区，分别是阈值调节区：可以通过滑动条灵活的调节检测阈值，提高系统的抗干扰能力；音频特征区：显示当前音频信号的特征图。状态显示区：显示当前模型预测结果，红灯表示为异常，绿灯表示为正常。

6. 结语

本研究实现了一种能够在资源有限的微控制器单元(MCU)上运行的鼾声检测方法，采用门控循环单元(GRU)模型对音频数据进行处理，并且结合模型优化方案，例如模型结构裁剪，模型量化等，成功将 GRU 模型适配到 MCU 平台，使得系统能够独立完成鼾声检测无需依赖外界计算资源，且无需联网。同时为了丰富产品显示，设计了一个人机友好的 UI 界面，可以实时显示系统识别状态，并可以调节系统检测阈值等。实验表明，该模型在保持高准确性的同时，降低了系统算力需求，满足移动健康监测设备的实时性和便携性。这一研究为鼾症患者提供了新的睡眠健康管理解决方案，并拓展了深度学习在嵌入式系统中的应用前景。

参考文献

[1] Nguyen, M. and Huang, J. (2022) Snore Detection Using Convolution Neural Networks and Data Augmentation.In: Long, B.T., Kim, H.S., Ishizaki, K., Toan, N.D., Parinov, I.A. and Kim, YH., Eds.,*Proceedings of the International*

Conference on Advanced Mechanical Engineering, Automation, and Sustainable Development 2021 (*AMAS*2021). AMAS 2021. Lecture Notes in Mechanical Engineering. Springer, Cham. https://doi.org/10.1007/978-3-030-99666-6_15

[2] Xie, J., Aubert, X., Long, X., van Dijk, J., Arsenali, B., Fonseca, P., *et al*. (2021) Audio-Based Snore Detection Using Deep Neural Networks. *Computer Methods and Programs in Biomedicine*, **200**, Article 105917. https://doi.org/10.1016/j.cmpb.2020.105917

[3] Goodfellow, I., Bengio, Y. and Courville, A. (2016) Deep Learning. MIT Press, 367-415.

[4] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., *et al*. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), Doha, October 2014, 1724-1734. https://doi.org/10.3115/v1/d14-1179

[5] Zhang, A., Lipton, Z.C., Li, M. and Smola, A.J. (2023) Dive into Deep Learning.Cambridge University Press.

[6] Krishnamoorthi, R. (2018) Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper.arXiv:1806.08342.https://doi.org/10.48550/arXiv.1806.08342

Call for Papers

Embedded Technology and Intelligent Systems

嵌入式技术与智能系统

国际中文期刊征文启事

https://www.hanspub.org/journal/etis

ISSN: 3065-1220

《嵌入式技术与智能系统》是一本开放获取、关注集成传统嵌入式技术与新兴智能系统的前沿研究最新进展的国际中文期刊，期刊特别注重软件算法、芯片设计与硬件实施的协同进展，以及理论研究与工程实践的紧密结合，面向学术界学者、产业界专家与工程师、学生及技术爱好者，关注中国领先产业集群的广阔发展潜力。本期刊强调发表原创性、创新性及具有实用价值的研究成果。 该期刊由汉斯出版社出版，全球发行，现诚邀相关领域的学者投稿。

投稿领域：

人工智能技术-边缘计算-端侧智能和大模型嵌入式应用
GPT-行业GPT以及GPT在嵌入式及智能系统研发中的应用
信息物理融合系统(CPS)-物联网技术-感知计算和无线传感网-泛在电力物联网-智能电表-储能技术-智能输变电
嵌入式系统结构-嵌入式操作系统与中间件-Linux、安卓和开源鸿蒙应用
实时操作系统-虚拟化和容器技术-混合关键系统
嵌入式软件形式化建模-软件测试和仿真-功能安全技术
嵌入式软件云原生技术-CI/CD和DevOpt-微服务
软硬件协同设计-开源指令集和开源芯片-RISC-V产业生态
嵌入式SoC技术--MCU 创新与生态-FPGA/DSP技术和应用
AI芯片和算法-存储技术-GPU 技术-视觉芯片及嵌入式显控应用
CAN和工业总线技术-时间敏感系统-电机控制-PLC和工业PC
无线通信技术-WiFi/蓝牙/Mesh/蜂窝/5G网络-物联网安全-低功耗设计
嵌入式系统课程改革-物联网和AI教学研究-职业教育-企业人才培养
嵌入式智能系统应用（智能家居、可穿戴设备、机器人、医疗电子、汽车电子和航空航天等）

征文要求及注意事项：

1. 稿件务求主题新颖、论点明确、论据可靠、数字准确、文字精炼、逻辑严谨、文字通顺，具有科学性、先进性和实用性；
2. 稿件必须为中文,且须加有英文标题、作者信息、摘要、关键词和规范的参考文献列表；
3. 稿件请采用WORD排版，包括所有的文字、表格、图表、附注及参考文献；
4. 从稿件成功投递之日起，在2个月内请勿重复投递至其他刊物。本刊不发表已公开发表过的论文。文章严禁抄袭，否则后果自负；
5. 本刊采用同行评审的方式，审稿周期一般为5~14日。

欲了解更多信息请登录 https://www.hanspub.org/journal/etis　　联系邮箱：etis@hanspub.org

Hans汉斯

嵌入式技术与智能系统

主编：何立民　北京航空航天大学教授

主办：汉斯出版社　珠海昊谷电子科技有限公司

编辑：《嵌入式技术与智能系统》编委会

网址：https://www.hanspub.org/journal/etis

电子邮箱：etis@hanspub.org

出版：汉斯出版社