学校代码： 10246
学　　号：15210240091

硕 士 学 位 论 文

（专业学位）

基于终端用户开发的残疾人智能家居系统设计与实现

Design and Implementation of a Smart Home System for the Disabled Based on End-User Development

院　　系：　计算机科学技术学院
专业学位类别(领域) ：　计算机技术
姓　　名：　唐祥轩
指 导 教 师：　丁向华　副教授
完 成 日 期：　2018 年 3 月 5 日

指导小组成员名单

顾　宁　　教　授

张　亮　　教　授

卢　暾　　副教授

丁向华　　副教授

目录

摘要

残疾人由于生理缺陷在家居生活中面临诸多困难，特别是重症残疾人，他们缺乏一定的自理能力，无法独立生活，严重影响了自己和家人的生活质量。智能家居系统的出现改变了残疾人传统的家居环境，新的交互技术和家居自动化可以进一步弥补残疾人的生理缺陷，为残疾人独立进行家居生活提供技术支持。然而，现有的残疾人智能家居系统提供的服务和功能相对单一，个性化支持程度低，并不能满足残疾人及其家庭的差异化需求。残疾人无法根据自身的残疾状况以及家居环境，定制符合自己意愿的家居自动化方案。

针对上述问题，结合智能移动设备的便捷性和丰富的功能支持，能够为用户提供多种交互方式，非常符合残疾人的实际需求，因此本文在 Android 系统上设计并实现了基于终端用户开发的残疾人智能家居系统，并充分考虑了系统的性能需求。本文的主要工作如下：

1、设计并实现了基于唯一事件源的事件触发规则引擎。通过对主流终端用户开发工具的事件触发规则范式进行分析，提出基于唯一事件源的事件触发规则原型，保证了规则的表达性和简单性，同时可以适应移动平台性能需求，而后利用广播通信和服务监听机制，实现了对应的事件触发规则引擎。

2、提出了面向终端用户开发的服务组件设计方法。从终端用户角度出发，定义了事件触发规则中三种不同类型的组件模型以及对应的注册管理模型，屏蔽了不同服务组件底层实现的复杂性，降低了服务编辑过程的复杂程度；从服务提供者角度出发，提出了不同类型设备的组件实现方法和性能优化策略，保障了系统的扩展性需求。

3、实现了基于终端用户开发的残疾人智能家居系统，并将残疾人无障碍自理平台的相关设备组件化，结合语音、手势、网络天气等服务组件，实现残疾人智能家居自动化管理。最后分别从系统性能和用户评价两个角度设计实验，验证了本文系统的稳定性和易用性。

关键字：残疾人、智能家居、终端用户开发、事件触发规则
中图分类号：TP3

Abstract

Due to physical defects, the disabled face many difficulties in their daily life. In particular, some severely disabled people lack self-care ability and cannot live independently, which seriously affected the quality of life of themselves and their families. The advent of smart home has changed the traditional home environment for them. New interactive technologies and home automation can further compensate for the physical defects, which provide technical support for them living independently at home. However, the existing disabled smart home systems provide relatively simple services, and the degree of individualized support is low, which cannot meet the differentiated needs of the disabled persons and their families. They cannot customize their home automation solutions to meet their needs.

In view of the above problems, combined with the convenience and rich functional support of mobile devices, it can provide users with multiple interaction modes, which is in line with the actual needs of the disabled. Therefore, we design and implement a smart home system based on the End-User Development on the Android system, and fully considers the performance requirements of the system. The main work of this article is as follows:

1. The paper designs and implements a trigger-action rule engine based on a unique event. By analyzing the rule paradigm of the mainstream End-User Development tools and clarifying their advantages and disadvantages, the paper proposes a trigger-action rule paradigm based on a unique event, which considers the performance limitations of mobile platforms and ensured the expressiveness and simplicity of the rule. Finally, the paper implements the trigger-action rule engine with the broadcast communication and service monitoring mechanisms.

2. The paper proposes a service component design method based on End-User Development. From the perspective of the end-user, this article defines three different types of component models based on trigger-action rule, and designs the registration management model for the three types of components, which hides the complexity of the underlying implementation of different service components and reduces the complexity of the service editing process. From the perspective of service providers, this paper proposes component implementation methods and performance optimization

strategies for different types of devices, which ensures the scalability requirements of the system.

3. The paper implements a smart home system for the disabled based on End-User development, and incorporated the relevant equipment of the disabled into the system so as to realize the automatic home automation management. Through user evaluation and system performance analysis, the paper verified the system usability and stability.

Keywords: the disabled, smart home, End-User Development, Trigger-Action rule

Chinese Library Classification:TP3

第一章 绪论

1.1 研究背景

长期以来，残疾人由于身体缺陷，在社会生活中一直处于弱势地位，是人类社会中一个特殊的群体。近年来，随着社会保障制度不断完善，他们的生活状况逐渐得到重视，已经成为衡量一个国家或地区经济发展水平和文明程度的重要标志。

在我国，残疾人一直是一个庞大的群体，根据我国第二次残疾人抽样数据调查显示，我国残疾人总数约 8268 万人，占全国总人口数的 6.34%，残疾人家庭总数约 7050 万户，占全国家庭总户数的 17.80%[1]。由于现阶段我国社会发展不平衡，残疾人的工作和生活水平与正常人相差较大，大部分残疾人家庭处于贫困状态。为了改善他们的生活，我国不断加强残疾人事业建设，根据最近的残疾人事业发展统计公报显示，我国残疾人事业在“十三五”计划中取得巨大进展，其成果涵盖医疗康复、文化教育、劳动就业以及社会保障多个方面[2]。

虽然，随着我国残疾人事业不断发展，残疾人的生活水平逐渐得到改善，但是由于我国残疾人口较多，且分布广泛，接近 75%的残疾人分散在农村[1]，公共服务难以覆盖，他们的日常生活仍然面临巨大困难。特别是重度残疾人，他们往往缺乏一定的自理能力，如果没有家人或者医护人员的帮助，根本无法独立进行家居生活。残疾人无法独立进行家居生活，不但严重影响了残疾人自身的生活质量，更是让残疾人家庭面临经济、心理和生活的多重压力。

在传统的家居环境中，残疾人之所以面临各种困难，是由于家居环境中的设施彼此独立，且控制和操作的方式各异，残疾人可能因为身体缺陷，缺乏操控某些设备的能力。例如，对于正常人而言打开窗户是一件很简单的事情，但是对于肢体存在重度残疾人而言，由于丧失行动能力，可能根本无法移动到窗户旁，更不可能自己打开窗户。因此，如果可以通过技术手段为残疾人提供帮助，弥补他们的身体缺陷，增强他们独立进行家居生活的能力，那么对改善残疾人的生活质量以及残疾人家庭的经济生活水平会有重要意义。

智能硬件技术的发展，为改善这一状况提供了技术途径，大量的残疾人智能辅具开始进入残疾人的日常生活，改变了残疾人的工作和生活方式。其中，以物联网技术[3]为基础的智能家居系统尤为显著，其通过网络、通信和计算机相关技术将原本分散的家居设备集中调度，可以为用户提供统一便捷的控制方式，屏蔽了原来各个设备之间的差异性，让部分残疾人独立控制家中相关设备成为可能。

以智能家居为基础的家居自动化，使得家居环境更加智能，通过配置合适的自动化方案，残疾人用户不再需要亲自操控设备，系统可以根据传感器提供的环境信息合理调度设备以适应环境变化。

智能家居技术的发展确实改善了残疾人的家居生活[4,5,6,7]，但是还远远不能满足残疾人的实际需求。因为传统智能家居系统主要都是面向正常人设计，所提供的服务和功能相对单一，缺乏对于残疾人的特殊支持。事实上，残疾人相对于正常人而言，他们具备更多的特殊性，对于智能家居系统也有特殊的功能需求。一方面，残疾人对于智能设备的依赖程度更高，需要智能家居系统能够提供更便利的环境交互方式，从而弥补他们的身体缺陷，便于他们对于家居环境的控制；另一方面，残疾人的实际需求差异很大，具体取决于残疾类型，残疾程度，个人偏好，特定场景等等，且随着时间推移这种需求可能会不断变化，因此有必要为他们提供个性化的智能家居服务[8]。

根据上述需求，本文提出一个基于终端用户开发的残疾人智能家居系统，以帮助残疾人及其家庭成员实现家居自动化服务的个性化定制。由于移动设备的便捷性以及日益丰富的功能支持，能够为用户提供多种交互方式，且移动设备普及程度高，非常符合残疾人的实际需求，所以本文系统主要基于Android平台设计，并对系统性能进行优化，以保证系统能够为残疾人提供稳定服务。通过对终端用户开发相关技术分析以及移动设备的性能考量，本文对现有的事件触发规则范式进行优化和拓展，明确区分事件和条件的概念，提出基于唯一事件源的事件触发规则范式，降低自动化服务的编辑复杂度，减少事件监听造成的性能消耗。在事件触发规则引擎的设计和实现过程中，通过分别设计自动化服务的存储模型和执行模型，保障了自动化服务的存储和调度效率。然后，本文对所有的服务功能进行了抽象，并基于事件触发规则设计了组件模型，屏蔽了不同服务组件底层的复杂性，保障用户可以无差别的使用所有服务组件。最后，本文基于残疾人无障碍自助管理平台的智能家居设备，以及语音、手势、网络天气等服务功能，实现了残疾人智能家居系统。

1.2 国内外研究现状

物联网技术的发展以及智能家居系统的出现确实在很大程度上改善了残疾人的家居生活[4,5,6,7]。为了能够设计出符合残疾人需求的智能家居系统，本文分析和研究了以往关于残疾人智能家居系统的相关文献和技术。

智能家居系统已经不是一个新的概念[9]，它的出现对提高残疾人日常生活水平具有重大意义，结合智能辅助设备和语音识别等技术改变了传统残疾人家庭的生活方式[10]。智能家居系统，利用网络、通信和计算机相关技术将分散的家

居设备以及智能设备集中于家居环境，通过合理的调度可以为用户提供舒适安全的家居环境。为了让智能家居系统更好的为残疾人服务，大量的研究集中在为残疾人提供便利的环境交互方式，合适的家居自动化过程以及家居安全性策略[11]。

残疾人智能辅具和交互技术的发展，为残疾人提供了身体缺陷的弥补措施，使得残疾人可以独自使用智能设备完成工作和生活。例如基于无线网络的智能家居系统，肢残疾人可以依靠头戴鼠标实现对家居设备的控制[12]。眼球和视线跟踪技术的发展，进一步简化了残疾人控制家居设备的方式，这类智能眼镜使得残疾人可以通过视线实现对家居设备的控制[11,13,14]。另外，语音识别技术的快速发展，让残疾人感受到与家居环境的无障碍交流，逐渐成为残疾人智能家居系统不可或缺的重要部分。智能手持设备的普及为残疾人提供了更为丰富的环境交互手段，包括触屏感应，手势识别，语音识别，语音阅读等等[15]。根据有关数据显示智能辅助技术已经多达 5 万多种[16]，逐渐覆盖了残疾人生活的方方面面[17]。通过物联网技术将残疾人相关的设备关联组合，形成的自动化服务可以进一步实现家居环境的自动化管理，避免残疾人自己操作带来的问题。

虽然丰富的智能硬件、便捷的交互技术以及家居自动化定制方案能够为残疾人提供更好的家居服务，但还远远不能满足残疾人的需求，这具体取决于残疾人的残疾类型，残疾程度，个人偏好，家庭背景以及实时的情景，因此有效地利用残疾人所处的环境信息，对提高残疾人的生活质量至关重要[8]。Sohn 等人提出一种基于用户信息不断演化的个性化服务框架 SOLVE-D，能够根据家居环境信息为残疾人提供环境感知的个性化服务[8]，其主要依据残疾人的行为数据和环境信息为残疾人提供个性化的服务推荐，该技术需要依靠一定的硬件设备支持，且不能自动化的执行推荐结果。随着人工智能技术的发展，利用人工智能算法为残疾人提供个性化的服务吸引了大量的相关技术人员。他们充分利用用户的行为数据，分析残疾人的行为习惯，提取用户特征，从而根据以往的数据和当前的环境预测残疾人的下一步行为[18,19,20]。但是这些系统忽略了用户与环境的交互过程，所有的服务都是自动化，用户无法参与决策，事实上残疾人用户的意愿也应该是环境上下文信息应该考虑的重要因素。Hussein 等人提出一种基于两种神经网络的智能家居系统，学习和预测残疾人用户的下一步行为，并采取行动[21]，该系统实现了用户的中间决策，并针对不同用户提供语音和界面交互方式，但是该系统复杂程度高，需要特定的软硬件支持，用户参与决策的方式仍然受限。

无论是传统的家居自动化服务还是基于人工智能的个性化的家居自动化服务，其服务方案的设计和实现都是专业人员完成。对于传统的家居自动化服务，设计人员通过计算机语言实现特定的几个设备之间的关联关系，设置相关的数据过滤器或预定义规则，从而实现设备自动化管理。对于人工智能的个性化的家居

自动化服务，其设备的选择和关联仍然有专业人员设计，而在自动化规则的制定是计算机根据用户的行为数据分析生成的，因此更加的符合用户的实际需求，具备一定的个性化因素。

事实上，这些设计和开发残疾人家居自动化服务的专业人员并不知道残疾人家居环境的具体情况，也无法确定残疾人实际生活中想要将哪些设备关联在一起，因此难以设计出真正符合残疾人实际需求的家居自动化方案，自动化服务的个性化程度仍然有限。其实真正了解残疾人实际生活需求的是残疾人自身和他的看护人员，他们是最终的服务受益者，也是最了解残疾人家居环境实时状况的人，因此在自动化服务设计和执行中应该拥有最终的发言权。但是家居自动化服务的制定有很高的技术要求，无论是残疾人还是残疾人的看护人员，他们往往都不具备计算机技术背景，难以通过计算机语言来管理智能设备，更加无法实现自动化服务中复杂的逻辑设计，因此他们的实际需求难以表达。为了解决终端用户的这一实际需求，需要终端用户开发的相关技术的支持，但是目前残疾人智能家居系统还缺乏该技术的相关研究。

1.3 本文工作与贡献

本文的主要工作是分析了目前残疾人智能家居系统所面临的问题，提出了基于终端用户开发相关技术的解决方案，结合移动设备的便捷性和丰富的功能支持，实现了最终的残疾人智能家居系统。在系统的设计和实现过程中，本文充分考虑了 Android 平台性能限制，通过优化事件触发规则范式，以及设计合适的服务存储模型和执行模型，降低了性能消耗，并通过性能对比实验验证了系统稳定性。从终端用户的角度出发，本文为了屏蔽不同功能服务底层实现的复杂性，抽象了服务组件模型，为用户提供了统一便捷的组件编辑方法。最后，本文实现了残疾人智能家居系统，并通过用户评估验证了系统的易用性。本文研究的主要工作有以下几点：

1、提出唯一事件源的事件触发规则范式。借鉴现有的事件触发规则范式，分析他们的表达性和复杂性，结合移动平台性能需求，提出基于唯一事件源的事件触发规则范式原型。

2、基于唯一事件源的事件触发规则范式，设计和实现了 Android 系统下的事件触发规则引擎，包括事件触发规则的存储模型和执行模型，以及事件、条件和操作的管理器功能。

3、提出面向终端用户开发的服务组件实现方法。基于事件触发规则原型，设计了事件、条件和操作三类组件模型，以及对应的组件注册管理模型。并根据组件关联的设备类型，提出每种设备类型组件的实现方案和优化策略。

4、实现了基于终端用户开发的残疾人智能家居系统，包括服务管理、设备管理以及服务编辑功能。并将残疾人无障碍自理平台的家居设备组件化，结合其他服务组件，实现对家居设备的定制化自动管理。

1.4 本文的组织结构

本文主要分为六章，具体组织结构如下：

第一章，介绍当前残疾人智能家居系统的研究背景和意义，目前残疾人智能家居系统国内外研究现状，以及本文的主要研究内容。

第二章，首先介绍了终端用户开发的相关概念以及技术，然后对比分析几个主流的终端用户开发工具，总结目前终端用户开发技术的主要实现方法。

第三章，详细描述了基于唯一事件源的事件触发规则引擎的实现过程，包括事件触发规则原型构建，数据模型设计，以及规则引擎各个功能模块的具体实现。

第四章，详细介绍了面向终端用户开发的服务组件的实现方法，包括事件触发规则的三种组件模型设计，服务组件的注册和管理方法，以及不同设备类型的组件实现方案。

第五章，介绍了基于终端用户开发的残疾人智能家居系统的各个功能实现，并设计实验对系统进行了性能检测和用户评估。

第六章，总结本文的主要工作内容，并对本文研究的进一步发展做出了分析。

第二章 相关工作

现阶段残疾人智能家居系统的相关研究仍然存在很多的不足，特别是家居自动化服务缺乏个性化，难以适应残疾人及其家庭的复杂多变的实际需求。针对这些问题，本文拟通过终端用户开发的相关技术，为残疾人提供编辑和改变家居自动化方案的途径，使得家居自动化方案能够满足残疾人自身及其家庭个性化需求。

虽然终端用户开发（EUD）的概念由来已久[22]，且逐渐应用于人们的工作和生活，但是缺乏主流的工具应用于残疾人智能家居环境。为了能够更好地将其应用于残疾人智能家居环境，本章将首先介绍终端用户开发的概念及发展，然后对比分析了现今几个较为流行的终端用户开发工具。

2.1 终端用户开发

终端用户开发（EUD）的思想始于 20 世纪 90 年代初期，一些研究学者提倡让最终用户参与软件设计，以解决软件设计所面临的挑战：软件使用情况的不断变化，现实环境的复杂多变以及未来需求的不可预测性[22]。基于这些问题，他们认为软件系统在设计过程中应该考虑系统的可剪裁性，使得软件在后期使用过程中可以被用户修改[22,23]。依据软件可能被修改的方法，早期的终端用户研究人员将这些软件大致分为三类：1、提供多种预期的替代方案供用户选择；2、将软件提供的功能组件组合成新的服务；3、改变软件本身的功能[23]。

基于第一种设计思想，一些智能家居系统开始提供用户自定义家居自动化方案的功能，其主要通过预先设计服务方案，提供设备数据过滤器，使得用户可以根据自己的需要调整方案或者设置合适的过滤数据，以适应不同场景需求，这在一定程度上可以解决用户需求的差异性问题[24]。但是，这样的解决方案存在很大限制，随着设备和需求的增加，系统规则会逐渐复杂化，扩展性差，规则与设备的耦合程度高。所以，这些系统往往只是支持一些简单的任务定制，或者服务于特定领域的技术人员。

主流的终端用户开发框架都基于第二种方法，一方面是因为第一种方法在功能的扩展性方面存在很大限制，难以满足终端用户的需求，另一方面是第三种方法实现难度较大，大部分软件在设计初期都会确定一个功能范围。基于最初的终端用户开发思想，一些相关研究逐渐出现，Fischer 等人提出了播种，进化增长和再播种模型（SER）[25]，以及基于知识的设计环境（DODE），该环境能够为用户提供了一个不断变化的空间，且用户可以塑造该系统的功能[26]。然而受限于

一定的软硬件环境，终端用户开发思想只在研究环境中被使用[22]。

随着智能设备的逐渐普及，终端用户自身所处的环境和使用的工具逐渐发生变化，新的需求促进了终端用户开发思想的进一步发展和变化。Lieberman 等人将最终用户开发定义为允许非软件开发人员的终端用户能够通过系统提供的方法和工具创建，修改及扩展软件功能，潜在的终端用户可以利用现有的功能和工具创建新的软件行为[27]。由于早期终端用户开发主要集中在研究环境，Lieberman 等人认为终端用户开发中的终端用户应该是指那些不需要技术支持的领域专家，这些专家只是需要软件能够为他们提供特定的拓展功能以满足他们研究特定领域的需要[27]。然而随着人们周围软硬件工具的逐渐丰富，人们的生活环境变得越来越可塑，最终用户的群体也逐渐发生了变化[22]，Ko 等人认为终端用户的定义范围已经不应该局限于研究人员，而应该是使用终端用户开发工具的各类人群[28]。随着终端用户群体的范围的扩大，终端用户开发工具也面临新的挑战，即终端用户群体的差异性。

自从终端用户开发思想被提出，已经有大量的终端用户开发工具出现[29,30,31,32]。然而,这些工具往往需要终端用户具有一定的编程技巧，并不能满足新的终端用户群体的需要。这些潜在的需求促使新的人机交互范式诞生，为终端用户提供更为便捷的环境配置工具。通过调查发现，近几年来主流的终端用户框架基本都是基于事件-条件-行为（ECA）规则设计[33]，并且能够很好的满足终端用户的大部分需求，例如 IFTTT[34]，Tasker[35]等。虽然大多终端用户框架都建立在 ECA 规则之上，但是为了满足终端用户的心理需求，往往在工具模型设计上存在一定差异，具体表现为如下三种：

1、过滤列表原型[30,36,37]。这个原型要求用户按照事件触发响应的顺序建立服务过程。具体表现为依据事件-条件-行为划分步骤，每个步骤含有相应的列表，当条件或事件选定时，后续的行为会根据选定的条件进行筛选，只保留能够响应当前条件的行为。例如，在需要保存语音资料的时候，条件设为语音资料，那么编辑保存行为时只会显示与语音保存相关的操作，即行为列表依赖于前期条件的选择。

2、接线图原型[30,36]。这个模型类似传统的工作流模型。所有的触发事件和行为以图标表示，然后通过方向箭头来表示事件和行为的关系。具体表现为，用户首先选定触发事件源头的图标，然后选定想要触发的行为节点图标，从而构成图形。随后利用图标所提供的事件和行为列表来构成具体的服务。相比于过滤列表，接线图原型表现更为直观，但是一旦服务增多，节点间的连线将变得极其复杂，使得服务难以维护[30]。

3、拼图游戏原型[30,36,37,38]。这个原型类似于接线图，但是更加直观表现

了现实环境，是能直接映射现实环境的替代方案。具体表现为现实环境与图形基本一一对应，通过耦合或开关工具将不同图形进行联系，并确定相应的事件行为，同样是通过列表展示组件所能提供的功能。但是，接线图所面临的问题，该模型一样存在，不同的是该模型在空间的表现力上更直观和丰富。

除了工具模型的差异，规则引擎也是影响终端用户开发工具用户群体的重要因素。规则引擎支持的规则越复杂，那么它所能表现的实际行为就越丰富，而规则工具的使用难度也越大。例如 JBoss Drools[39]，OpenRules[40]，IBM WebSphereJRules[41]，它们虽然提供了简化规则编程的机制，但非技术人员仍然难以使用，JESS[42]小巧灵活，速度快，但是语法难度大，都无法适应终端用户开发工具的需求。为了适应新的用户群体需求，终端用户开发工具必须要权衡规则引擎的表达性和简单性。IFTTT[34]，Tasker[35]等基于发布-订阅事件驱动的终端用户开发工具已经被证明可以被非技术用户正确使用，且支持移动平台，具备很好的便捷性，但是在表现力方面仍然有所欠缺。

总体而言，终端用户开发技术的发展已经让终端用户修改软件以适应其自身需求的愿望得以实现，虽然相关的技术工具仍然存在诸多问题，例如表达性和易用性之间的矛盾，但是足以满足大部分人的生活和工作需求。而且，随着物联网技术的发展，已经吸引了一些学者想要通过终端用户开发的思想为智能环境提供更多的可能性[24,36, 43,44]，但是仍然缺少成熟的应用框架，更加缺少对于残疾人这种特殊的终端用户群体的研究。事实上，终端用户开发工具所能提给终端用户修改软件、工具及环境的能力是现阶段残疾人智能家居环境所欠缺的，也是能够让残疾人将自身意愿融入家居环境的有效途径。

2.2 终端用户开发框架

大量的终端用户开发工具已经出现在人们的日常生活中，改变了人们的工作和生活的方式，例如 IFTTT[34]，Tasker[35]，Atooma[45]等等。使用这些工具，一方面可以将原本复杂的工作变得简单化，使得工作更具效率，另一方面可以与身边的智能环境更方便的交互，使人们居住的生活环境更加舒适。为了更好的了解现今终端用户开发的主流技术，接下来将分别介绍和分析几个终端用户开发工具。

2.2.1 IFTTT

IFTTT（If This Then That）是目前较为流行的一个 Web 应用程序，并支持移动端版本（Android 和 iOS）[34]，其运行界面如图 2.1 所示。IFTTT 通过事件触

发规则将各种信息功能联系起来，从而让互联网更好的为人们服务。在 IFTTT 的功能定义中，“this”是事件触发条件 Trigger，“that”是事件触发后需要执行从操作 Action，无论是 Trigger 还是 Action 都是基于具体的网站服务，称之为 channel，例如 Facebook，Twitter，Gmail，Flickr 等等[36,37,44]。IFTTT 将这些分散的网站服务集中起来，通过事件触发规则让用户可以编辑符合自己需求的网络服务，例如用户 A 收到一条视频短信，想要转发到 Facebook 上，原本需要分别进行操作来完成转发，过程复杂，但是通过 IFTTT 用户只需要编辑一条规则“如果收到视频短信那么转发到 Facebook”，就可可以通过简单操作实现自动转发。但是，IFTTT 所能实现的服务较为简单，无论是 Trigger 还是 Action 都只能支持一个事件或行为，表现力上极为欠缺。虽然根据一些研究和调查显示用户大部分的功能需求单一的触发行为足以实现[37]，但是终端用户开发工具不应该限制用户创建复杂行为的能力。

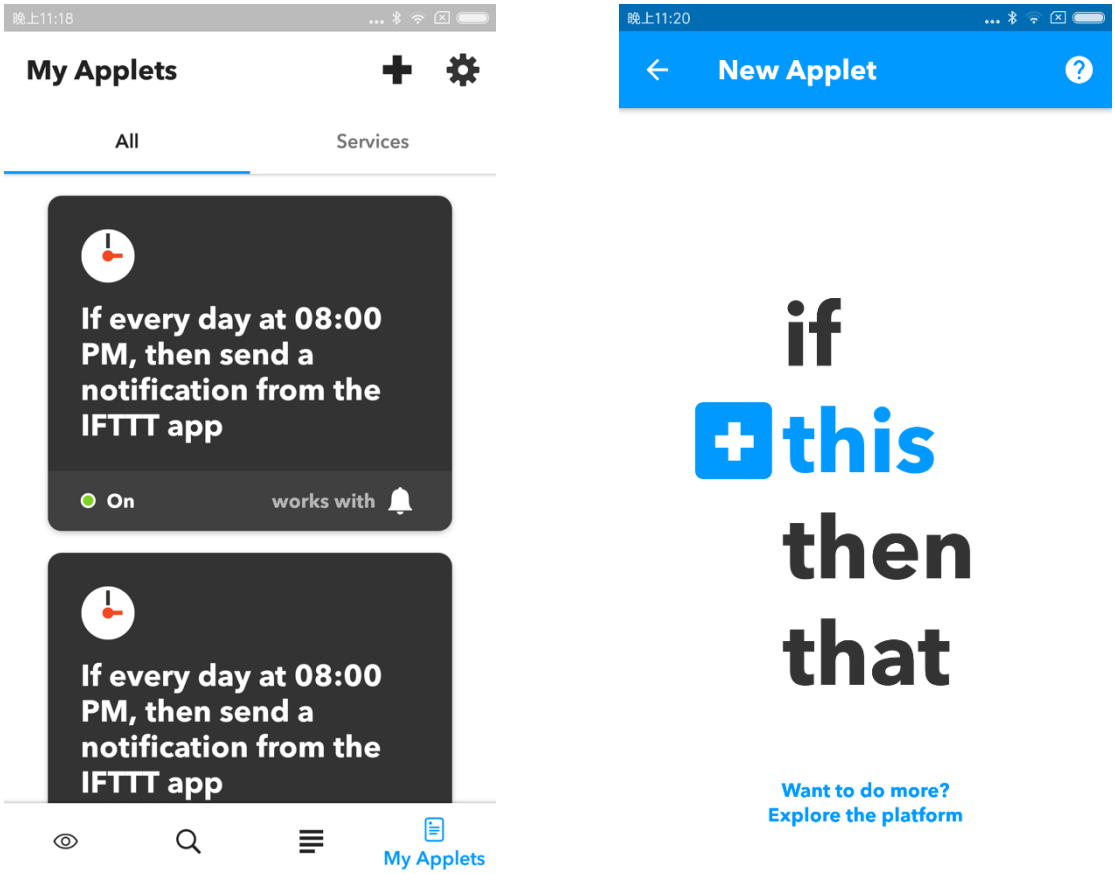

图 2.1 IFTTT 运行界面

2.2.2 Tasker

随着移动设备的普及，吸引了一部分终端用户开发的研究人员开始考虑将相关技术应用于移动设备上，其中 Tasker[35]便是较为成功的一个工具。Taker 发展于 2009 年，并与 Android 设备兼容，同年在 Android Developer Challenge 2 中获得第三名，一直是一款最受欢迎的 Android 应用[38]，其运行界面如图 2.2 所示。

图 2.2 Tasker 运行界面

Tasker 将用户配置的服务称为配置文件（Profile），每个配置文件包含背景（Contexts）和任务（Taskers），当 Contexts 条件符合时执行 Taskers 所包含的所有操作（Action）[38]。在 Tasker 中，配置文件最多可以包含四个条件，且 Taskers 可以包含一个或多个 Action，使得用户定义复杂服务成为可能。在实际的定义过程中，用户需首先指定一个条件，并立即添加一个或多个动作，然后再选择是否添加额外的附加条件，最后保存。

通过 Tasker，用户能够充分的调度设备和网络提供的服务来满足他们的需求，有效的提高工作效率，例如用户可以定时开启应用，设置能耗提醒，消息免打扰，

网络自动切换等等。然而，有利必有弊，Tasker 确实能够提供更为丰富功能服务，但是与之而来的弊端也特别明显。实际上，Tasker 的使用门槛远远高于 IFTTT，由于可以控制的地方太多，提供的服务复杂多样，往往在编辑服务时出现错误，很难适用于一般的用户，使用难度较大。

2.2.3 Atooma

Atooma[45]是一个移动应用程序，目前有 Android 和 iOS 两个版本，其运行界面如图 2.3 所示。Atooma 相对于 Tasker 而言较为年轻，首次亮相是在 2012 年 9 月，区别于 Tasker，这款 APP 提供免费下载，备受瞩目[38]。

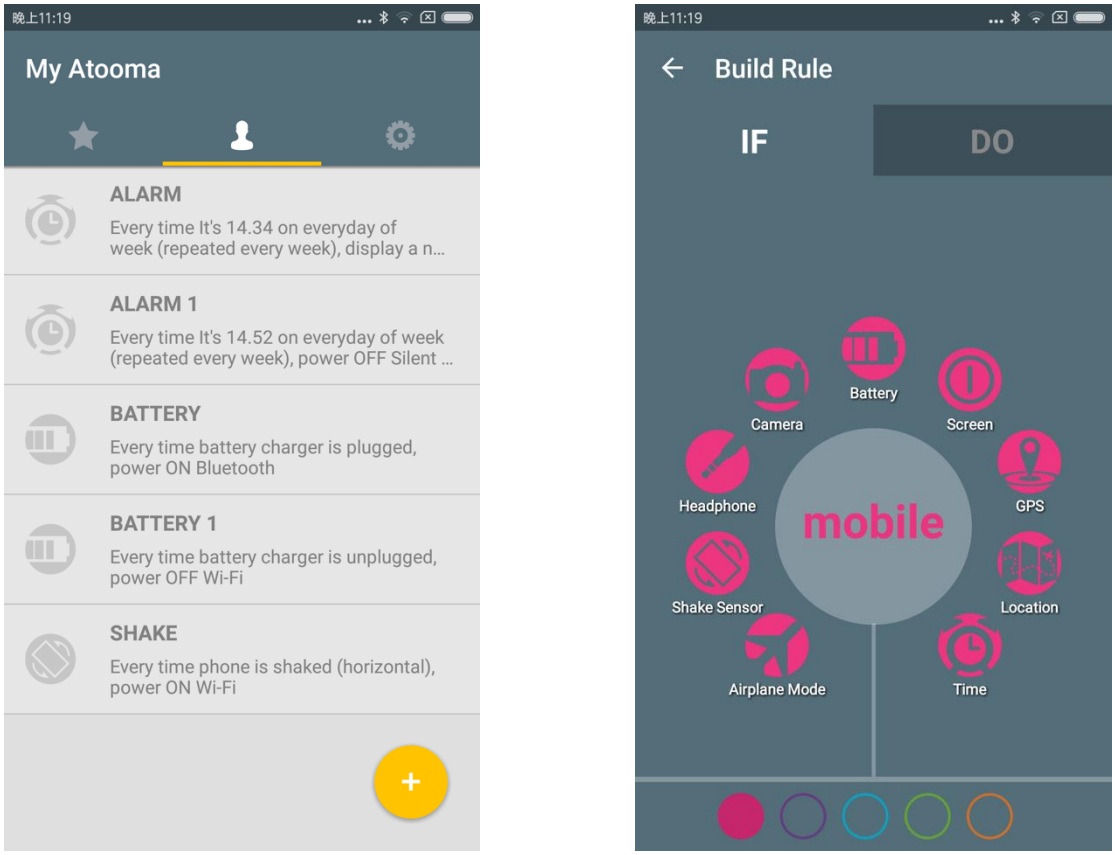

图 2.3 Atooma 运行界面

Atooma 在一定程度上与 IFTTT 类似，基于 if/do 的条件动作模型，只是 Atooma 的条件和规则可以扩充到五个，以支持用户更为复杂的需求。同时，无论是 if 还是 do 都区分为 5 中类型，分别是系统应用（粉红色），应用程序（紫色），对象（蓝色），插件（绿色），文件（橙色），不同类型中支持的服务各不相同，Atooma 最多可以支持 5 个类型相同或不同的条件和行为[38,43]。在实际

的服务创建过程中，首先用户必须制定一个 if 的类型，然后在类型中选择应用工具，然后设置该应用的操作，如此一个条件便创建成功了。在第一个条件创建成功后，用户可以继续添加最多四个条件或者直接添加动作，动作与条件的创建过程相同，最后保存成功运行，服务就会在后台自动运行了。

Atooma 另一个特点是 Atooma 配置的服务可以共享给他人，或者上传到社区，自己也可以到社区上下载他人的配置服务，简单便捷。

2.3 小结

通过上述的介绍可以发现基于终端用户开发的相关研究由来已久。随着智能设备的不断普及，人们的日常生活与智能设备的联系越来越紧密，新的终端用户开发工具可以帮助人们编辑和修改现有的工具和环境，从而改善人们的日常生活。然而，这些工具主要还是应用于正常人的工作和生活环境，缺少对于残疾人这类特殊群体的相关设计。为了能够让残疾人使用终端用户开发工具来编辑和修改家居环境以满足其个性化需求，本文将基于这些相关技术，设计和实现残疾人智能家居背景下的终端用户开发工具。

第三章 基于唯一事件源的事件触发规则引擎设计与实现

通过前两章的介绍，可以发现我国目前残疾人生活状况仍然严峻，而现有的残疾人智能家居系统无法满足残疾人个性化的家居生活需求。为了解决上述问题，本文提出基于终端用户开发技术的解决方案，并对现有的终端用户开发工具以及相关研究进行了分析，结合移动平台的优越性和性能限制，提出移动平台下基于唯一事件源的事件触发规则模型，并实现了对应的事件触发规则引擎。

3.1 事件触发规则范式设计

长期以来，无论是普通的智能家居系统还是残疾人智能家居系统，它们提供的服务方案都是由开发商设计定义的，并由拥有丰富经验的计算机开发人员实现，具有很强的技术元素。如果将家居服务方案从开发者手中移交到残疾人自己手中，那么所面临的主要问题是残疾人及其家庭成员缺乏一定的计算机背景知识，难以像开发人员一样通过计算机语言进行服务方案的开发。因此，需要为残疾人用户提供一种易于理解的，贴近自然语言的语言模型。

如果使用自然语言来描述残疾人家居生活中想要的服务，可以大致的描述为“在什么情况下，想要做什么事情”，这里的“情况”可能是时间、地点、事件或者其他的一些场景因素，“做什么事情”是指残疾人想要满足自己某种需求而做出的一些行为，可能是控制设备，也可能是某种提醒。根据以往的相关研究，这些事件行为很多是可以被规则化的表达出来的，而且能够被没有计算机背景的用户所理解的[37,43]。其中基于事件触发（TA）规则的范式在用户体验和满意度上有较高的评分，且现在主流的终端用户开发工具也都基于这种范式设计[43]：

IF<Condition>, Then<Action>

该范式虽然容易被没有技术背景的终端用户所理解，但是在表现力上存在一定的不足，例如 IFTTT 是标准的 if…then…模型，但是其只能表达单一事件和行为，如果想要表达复杂任务，就需要设置多个彼此关联的方案，难以满足残疾人的实际生活需求。

为了弥补表现力上的不足，部分系统通过布尔运算来增加 Condition 的复杂度，将 TA 规则原型描述为多个 Condition 和多个 Action 的组合，范式大致如下：

IF<Condition1 and/or Condition2 …>Then<Action1, Action2…>

该范式虽然可以弥补原来单一事件和行为在表现力上的不足，但是却与现实情境存在一定的差异，且用户可能会在理解和设计服务时存在困惑，从而导致实际执行结果与用户预期的结果存在很大偏差。导致这种错误的主要原因是该原型无法清楚描述事件和条件之间的差别。事实上，事件是指某种状态的改变，是一个瞬间行为，或者是一个短时间的行为。条件，或者说是环境信息，是指一种状态，这种状态能够维持一段时间，具有一定的延展性。具体到残疾人的家居生活中，“残疾人进入卧室”和“残疾人在卧室”就是事件和条件的区别。

错误的理解事件和条件，一方面可能会造成错误的结果，另一方面可能会造成系统本身的负担。例如，某一任务被描述为“如果残疾人进入卧室，且卧室的灯是关着的，且时间是晚上 6 点之后，那么打开卧室的灯”，被翻译成规则语言为“if Location=bedroom and light = off and time > 18:00，then turn on the light”，在这种情况下，系统需要监听三个事件源，分别是 Location，light 和 time，每个事件源的改变都会触发检测，造成系统资源的浪费。由于本文系统基于 Android 系统设计，所以系统资源有限，这样的方案难以应用于移动设备。

基于以上原因考虑，本文将服务描述为唯一事件源触发的任务，即将原来的多个 Condition 进行强行区分，选出一个 Condition 作为唯一被注册监听的事件源，而将其他 Condition 托管给相应的条件管理器，具体范式如下：

Once Event，match <Condition1,Condition2…>, take <Action1, Action2…>

通过这样的范式设计，可以将原本多个事件监听的缩减为一个，且不会影响最后执行结果，在一定程度上能权衡了规则的表达力和复杂性，也可以适应移动平台的性能要求，具体结构如图 3.1 所示。

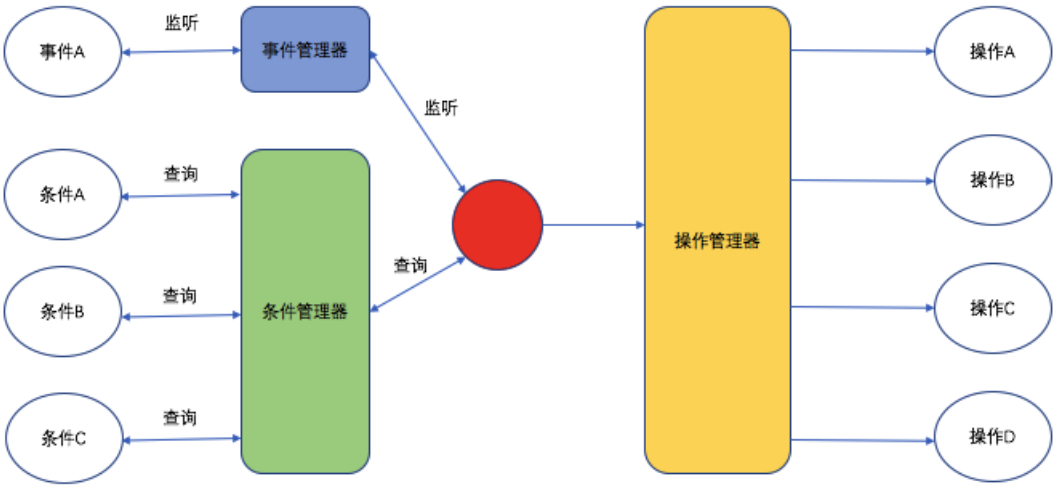

图 3.1 事件触发规则原型图

通过将条件验证转交给条件管理器来管理，每个任务只会创建一个事件监听对象，当监听对象返回事件触发结果后，当前任务才会去核对其他条件是否满足，进一步决定是否执行当前任务需要完成的操作。而条件管理器是所有任务共有的，且其执行状态不依赖于任何任务，并不会因为任务增加造成额外的资源消耗。

3.2 事件触发规则模型设计

在设计基于唯一事件源的事件触发规则引擎前，需要对该规则引擎的执行服务进行模型设计。根据服务的活动状态，本文分别设计了服务的存储模型和执行模型。存储模型是为了便于系统对服务进行存储和管理，执行模型则是为了便于规则引擎的调度和通信。

3.2.1 存储模型

基于唯一事件源的事件触发规则原型，本文将自动化服务设计为一个包含事件，条件、操作以及控制信息的对象 Project，其类结构如下：

```
class Project {
        private String name；
        private EventData eventData;
        private List<ConditionData> conditions;
        private List<OperationData> operations;
        private Boolean isActive;
        private Boolean isLoop;
}
```

name 用于区分服务方案的唯一标识，在服务存储和管理过程中，需要根据该标识进行检索。

eventData 是触发自动化服务的唯一事件源，是事件对象的数据对象，包含事件的比较逻辑和数据。系统根据该数据对象可以查找到对应的事件组件，并注册事件监听单元，执行服务监听。

conditions 是服务需要认证环境信息，是判定操作是否执行服务操作的辅助信息，理论上可以设置任意数量，甚至不设置。每个 Condition 是一个条件组件的数据对象，包含条件比较逻辑和数据。

operations 对应于设备的具体操作，可能是控制家居设备，也可能是警报或者其他操作，一个服务方案理论上可以包含多个操作，至少有一个。每个 Operation 对应一个操作组件的数据对象，包含操作组件名称和操作类型。

isActive 是作为判断当前服务是否激活的标志，未激活的服务不会被转换为对应的执行模型，不会被规则引擎调度，所以永远不会被执行。

isLoop 定义当前任务一次执行完毕后是否继续保持激活状态，如果设置为 false，那么当前的任务只会被执行一次，如果想要继续执行就必须重新手动激活；如果设置为 true，那么当前的任务会被重复监听并执行。

3.2.2 执行模型

由于自动化服务需要设置对事件源的状态监听，因此在服务的调度和执行过程中必须与对应的事件组件进行数据通信，来保证任务的正确触发和执行。存储模型虽然可以完整的表示家居服务，但是无法满足数据通信需求。

因此，在实际任务执行过程中，需要将自动化服务的存储模型 Project 转换为执行模型 Lotus，其类结构如下：

```
final class Lotus {
        private final Project project;
        private final Uri uri;
        private final PendingIntent lotusIntent，unsatisfiedLotusIntent;
        private AbstractSlot slot;
        private final BroadcastReceiver receiver;
}
```

uri 是可执行对象标识，格式为 lotus：//hashcode。每个 Lotus 拥有唯一的 uri，保证广播发送和接收时对象唯一，能够正确连接事件监听器，条件验证列表以及操作列表中的不同服务组件。

lotusIntent 是正常数据的通信对象。该对象是为了响应事件监听器正常被触发后发回的反馈信息，包含当前 Lotus 的标识信息。

unsatisfiedLotusIntent 是异常数据的通信对象。当事件组件异常，如断开连接后，该对象会反馈相应的异常消息，通知执行器关闭当前执行组件 Lotus。

slot 是事件组件监听单元，用于监听设备或服务的状态数据。在创建过程中，lotusIntent 和 unsatisfiedLotusIntent 会一起被注册的到 slot 中，从而实现 slot 与 Lotus 之间的数据通信。

receiver 是广播接收者，用于响应事件组件的反馈信息，并在处理分析后采取对应的措施，目前主要包括两种类型，分别是正常触发和组件异常。

project 是原任务对象。当事件监听器返回消息后，如果是正常触发，那么 Lotus 会读取当前的 Project 对象中的 Condition 列表，并一一验证是否满足，不满足则过滤此次事件响应，反之执行服务的所有 Operation。

3.3 事件触发规则引擎实现

基于唯一事件源的事件触发规则原型以及 Android 系统下数据存储与通信技术，本文将事件触发规则引擎设计为模型管理，事件管理，条件管理以及操作管理等几个功能模块，其结构图如图 3.2 所示。

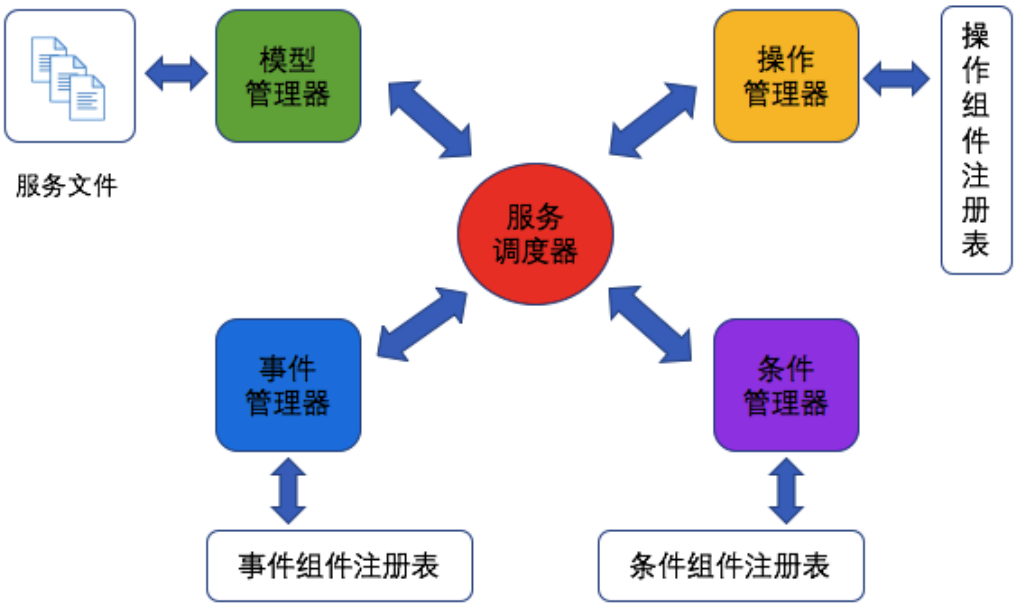

图 3.2 事件触发规则引擎结构图

规则引擎执行过程以及数据流向如图所示。首先，模型管理器将 json 文件保存的服务方案转化为存储模型 Project，并依次加载到系统中；当引擎启动后会遍历所有的 Project，将激活的 Project 转化为执行模型 Lotus，并将 Project 中的 Event

转换为事件监听单元 slot 注册到事件管理器中；当某个事件被触发后，slot 会触发广播发送者发送认证消息给对应的执行模型 Lotus；Lotus 在接收到正确的广播消息后，会将需要验证的条件传入条件管理器中，等待验证结果；当条件管理器验证完所有条件后，会将验证结果反馈给对应的执行模型 Lotus；如果验证通过，Lotus 会遍历所有的操作列表 Operation，通过操作管理器生成相应的 Operation 组件，并执行操作。

3.3.1 事件管理器

事件管理器主要负责事件组件与核心调度器之间的数据通信，具体功能主要包括事件监听单元生成和事件监听单元管理。其中，事件监听单元是事件管理器中的最小执行单位，同一个硬件可能同时有多个监听单元监听其行为。虽然不同类型的设备和服务的事件监听在具体实现上各不相同，但是为了便于事件管理器的统一管理，屏蔽设备之间的差异性，本文对事件监听单元进行了功能抽象，其类图如图 3.3 所示。

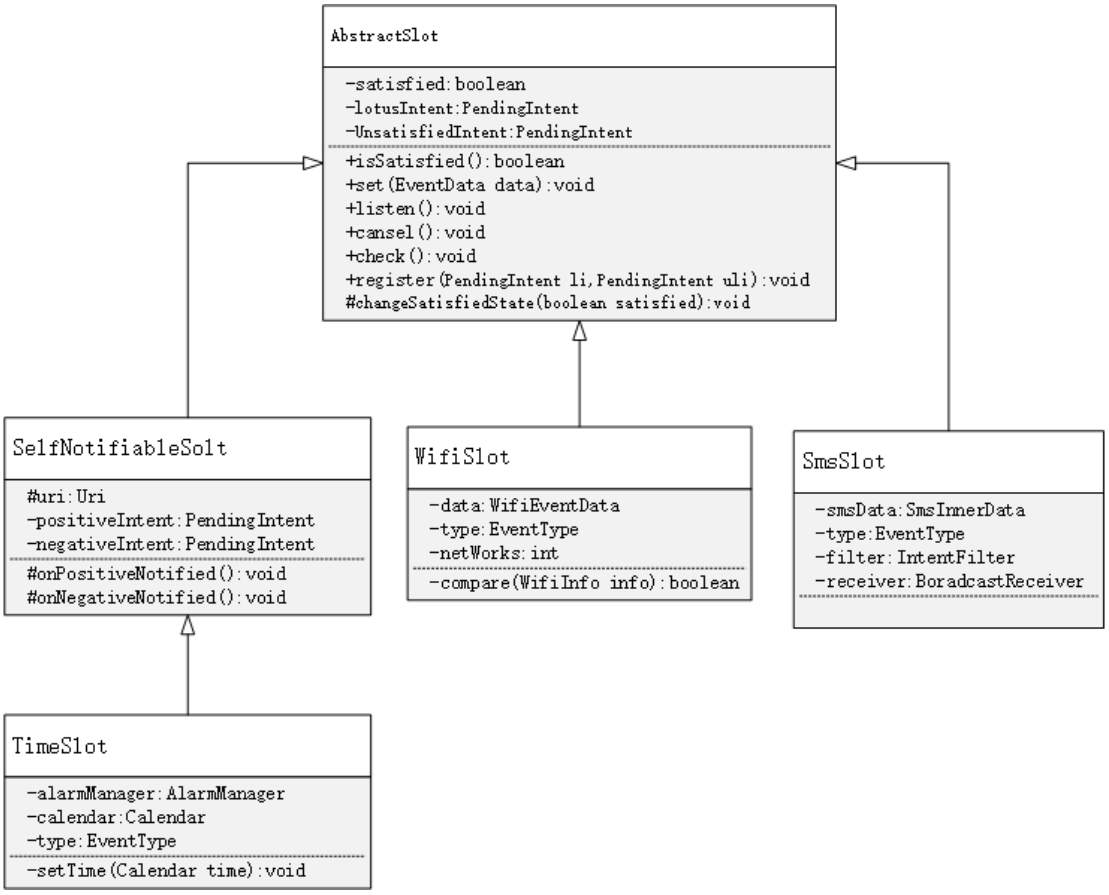

图 3.3 事件监听单元类图

所有的事件监听单元都实现自 AbstractSlot 及其子类，根据具体的事件组件行为，需要分别实现 AbstractSlot 中定义的方法。其中较为重要的方法包括监听开启 listen、监听关闭 cancel 和初次检测 check。这些是管理事件单元生命周期和验证事件触发条件的主要方法，只有能够正确开启和关闭监听单元才能够保证系统稳定，降低资源消耗。

事件管理器管理所有的事件监听单元的生命周期，包括事件单元的生成，执行和销毁。首先，当规则引擎启动后，会将原本的服务存储模型 Project 转换成 Lotus，同时对应的 eventData 也会被传入事件管理器中，事件管理器根据 eventData 去查找事件组件注册表，获取事件组件对象，并生成对应的事件监听单元。随后，事件管理器会将事件监听单元加入监听列表中，并开启服务监听，让事件监听单元监听事件设备的状态。最后，当事件设备状态满足时，事件监听单元会调用对应的广播发送者，发送事件被触发的事件广播，将监听结果反馈给核心调度器中的执行模型 Lotus。事件监听单元的生成过程如下：

事件监听单元生成

输入：EventData data

输出：EventSlot slot

过程：

```
eventPluginList <- EventRegistry.all() //获取事件组件列表
for eventPlugin in eventPluginList do //遍历列表，查找对应事件组件
    eventData <- eventPlugin.getData()
    if eventData.class = data.class then
        slot <- eventPlugin.slot()   //生成事件监听单元
        return slot
 end
```

3.3.2 条件管理器

条件管理器是管理条件组件的功能模块，用于获取环境的上下文信息。为了保证系统能够及时响应事件触发后导致的条件验证，本文将条件组件相关的数据

通信统一交由条件管理器来管理。条件管理器将所有的条件组件划分为延时条件和实时条件，延时类的条件需要条件管理器统一管理所有的延迟类服务，实时类服务则由条件管理器进行调度和验证。延迟类服务在本文中通过 Service 设计和实现，当系统启动事件监听后，条件管理器会启动所有相关的条件服务 Service，这些 Service 会定时更新缓存数据，或者监听相关设备发回的最新数据，具体设计如图 3.4 所示。

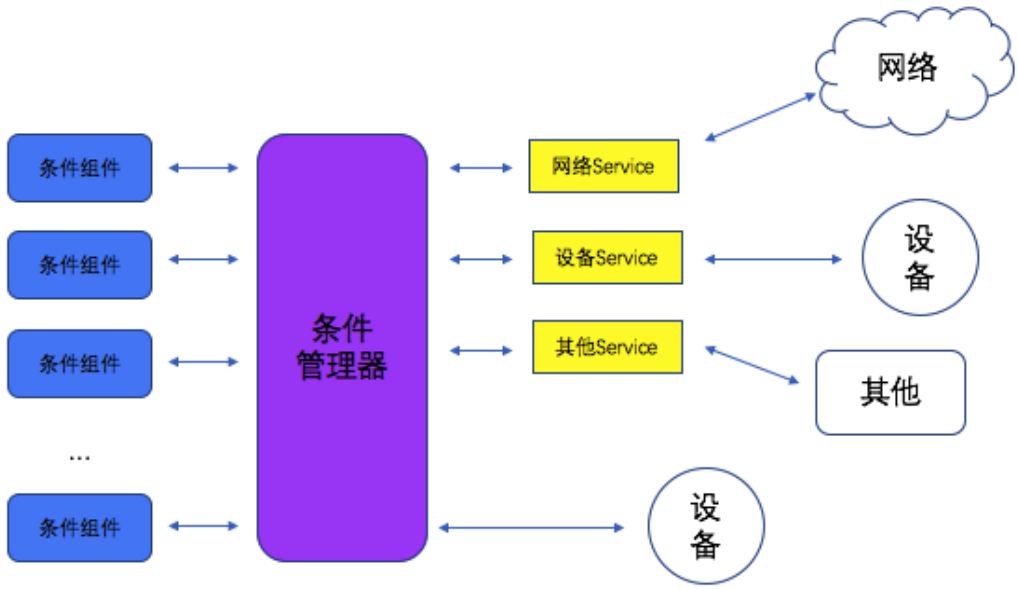

图 3.4 条件管理器结构图

当核心调度器需要为某条服务进行条件认证时，核心调度器会将所有待验证的条件数据传入条件管理器中，条件管理器会自动生成一个条件验证监听器，并将待验证的条件列表传入监听其中，依次进行条件认证。对于每一个待验证的条件，条件管理器会查找对应的条件组件，生成该组件的验证单元，进行条件验证。最后条件监听器会汇总所有的验证结果，并将结果反馈。

条件验证过程

输入： ConditionData data， Listener listener

过程：

```
conditionPluginList <- ConditionRegistry.all() //获取条件组件列表
for conditionPlugin in conditionPluginList do //遍历查找对应条件组件
    conditionData <- conditionPlugin.getData()
```

```
        if conditionData.class = data.class then
            matcher <- conditionPlugin.match() //生成验证单元
            result <- matcher.match(data) //验证条件数据
            listener.onResult(result) //反馈验证结果
    end
```

3.3.3 操作管理器

操作管理器相对于前面两个管理器而言功能较为简单，主要负责管理和调度操作组件，生成操作组件的执行单元。

操作管理器生成执行单元的方式与事件管理器类似，根据接收到的操作数据operationData，查询操作组件列表，获取对应的操作组件后，生成操作单元，并执行操作，具体过程如下：

执行单元生成与执行

```
输入：OperationData data
过程：
    operationPluginList <- OperationRegistry.all()
    for operationPlugin in operationPluginList do
        operationData <- operationPlugin.getData()
        if operationData.class = data.class then
            loader <- operationPlugin.loader()
            loader.load(data)
     end
```

根据执行单元的生成与执行过程，可以发现操作管理器与前面两个管理器相比存在明显区别。本文在对操作管理器进行设计时并没有实现执行结果的反馈机制，其主要原因有以下几点：1、设备类型复杂，部分设备或服务并不支持执行结果的反馈；2、等待设备或服务的执行结果会消耗一定的系统资源，增加系统负担；3、如果执行失败，那么可以通过定义复查服务进行环境检查，并不需要

一定要知道执行结果。基于上述几点原因，本文并没有实现操作组件的执行结果反馈机制，但是设计了反馈监听接口，以应对后续可能存在的需求。

3.4 小结

本章主要介绍了基于唯一事件源的事件触发规则引擎的设计与实现。通过对主流的事件触发规则范式进行分析，结合 Android 系统性能需求，提出基于唯一事件源的事件触发规则范式。并根据该规则范式，设计了对应的服务存储模型和执行模型，利用 Android 广播通信和服务监听机制，最终实现了事件触发规则引擎的各个功能模块。

第四章 面向终端用户开发的服务组件设计与实现

面向终端用户开发的服务组件设计是为了屏蔽不同设备或服务底层实现的复杂性，通过抽象组件模型，一方面可以实现组件统一化管理和调度，增加系统的扩展性，另一方面可以保障用户在编辑自动化服务的过程中，组件的编辑方式基本一致，降低编辑难度。

4.1 服务组件模型

本文将所有可以操作的设备、网络服务、Android 系统服务，传感器等都以组件的形式呈现给残疾人用户，他们可以将这些组件组合成具体的家居服务，来满足自身家居自动化服务的需要。服务组件是本文系统调度的单位，与具体的软硬件设备相关，根据功能可以被划分为事件组件、条件组件和操作组件。残疾人可以通过系统提供的编辑界面操作这些组件来完成自动化服务的编辑，而软硬件服务只需要实现相应的组件接口，就可将服务加入到系统之中供残疾人选择使用，从而丰富残疾人的家居生活。

为了便于管理器管理所有的组件，本文将所有的服务组件进行了抽象，其类图如图 4.1 所示。

PluginDef 接口是描述软硬件设备的公共接口，是连接组件服务的入口，包含该组件所需要的所有组成部分。id 是该组件的唯一标识，是区分不同组件的标志，用于注册和查询。name 是系统内组件名称。isCompatible 是用来验证当前组件是否可以被使用的标志，主要用于检测外部组件是否可用，例如传感器，电子设备等。checkPermission 和 requestPermission 是为了验证是否拥有权限，在 Android 系统中为了保障用户隐私，需要设置一定的权限，因此需要在执行过程中验证当前组件需要的权限，来保证不会被 Android 系统终止服务。data、view 分别是组件对应的数据和界面。

StorageData 是组件的数据模型，主要用于存储组件的状态数据，包含序列化与反序列化方法，以及数据格式的验证方法。

PluginViewFragment 是组件的 UI 接口，想要用户能够在组件选择界面查找的该组件，就必须实现该接口的所有方法。

虽然事件、条件和操作组件都实现自 PluginDef 接口，但是都对 Plugin 进行

了扩展和封装，以实现了不同的功能。

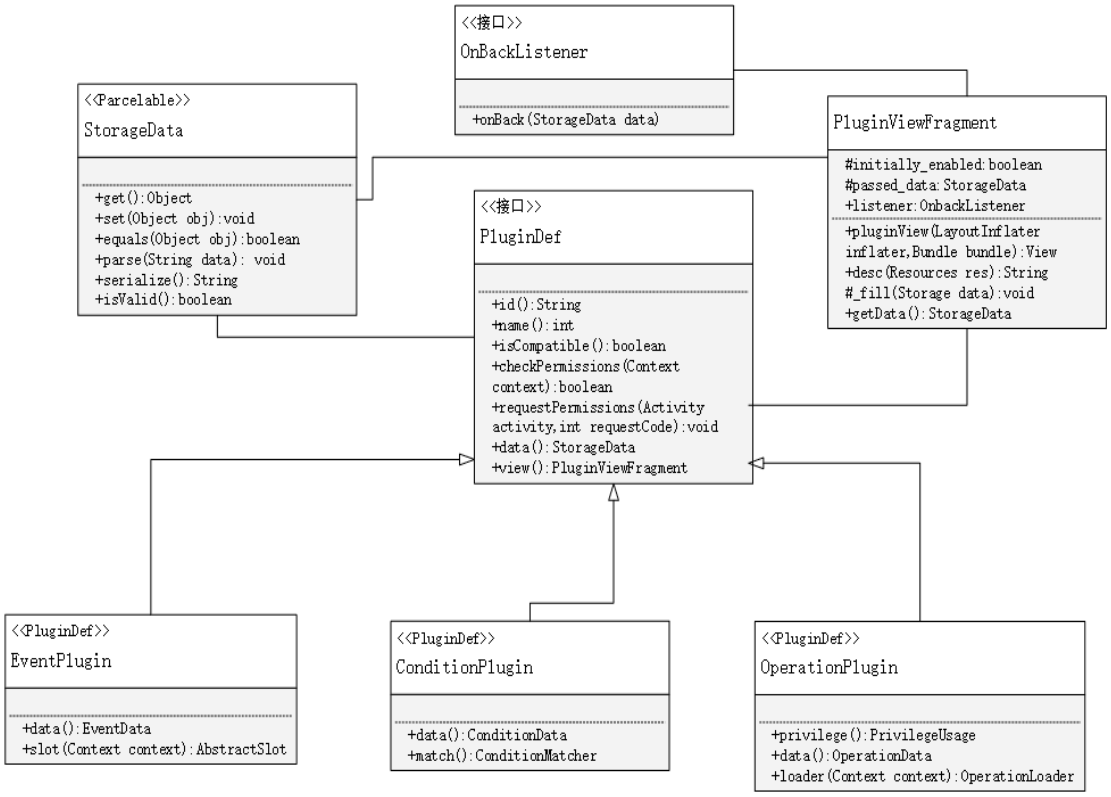

图 4.1 服务组件类图

4.1.1 事件组件设计

事件组件是系统的最重要的组件，是触发家居服务的首要条件。事件组件并不是由一个接口组成，主要包括以下几个方面，分别是 UI 界面、数据单元、监听单元以及组件接口，其类图如图 4.2 所示。

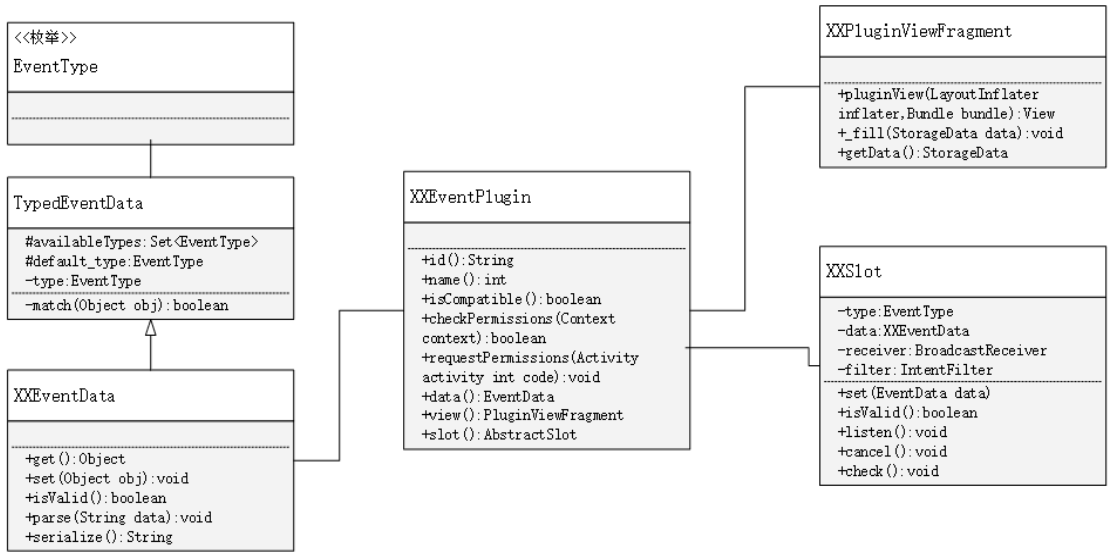

图 4.2 事件组价类图

EventData 是事件组件的数据模型，主要包括组件的状态类型，以及序列化与反序列化方法。不同的软硬件设备拥有不同的状态表示，例如电灯有开和关这两种状态，时间拥有早于、晚于和等于三种状态，温度传感器拥有低于，高于和等于三种状态等等。因此，设备状态具体可表示为逻辑和数值的组合。

Slot 是事件组件在监听过程中的可执行单位，之前已经介绍过其主要依靠 Android 广播机制与其他管理器进行通信，因此 Slot 包含 receiver 和 filter 两个成员用于标识当前的事件组件，接收和过滤广播消息。Slot 的主要方法是 listen，不同组件对于 listen 的实现存在巨大差异，该方法是监听具体软硬件状态的方法，对于 Android 系统服务，往往是通过注册监听广播来实现对该系统服务的监听，对于外接设备或者网络服务，需要实现具体的数据获取功能，往往需要开启单独的线程进行监听，然后再通过广播反馈给该组件。

4.1.2 条件组件设计

条件组件是系统用于验证环境信息的服务组件，组成结构主要包括 UI 界面、数据单元、验证单元以及组件接口，其类图如图 4.3 所示。

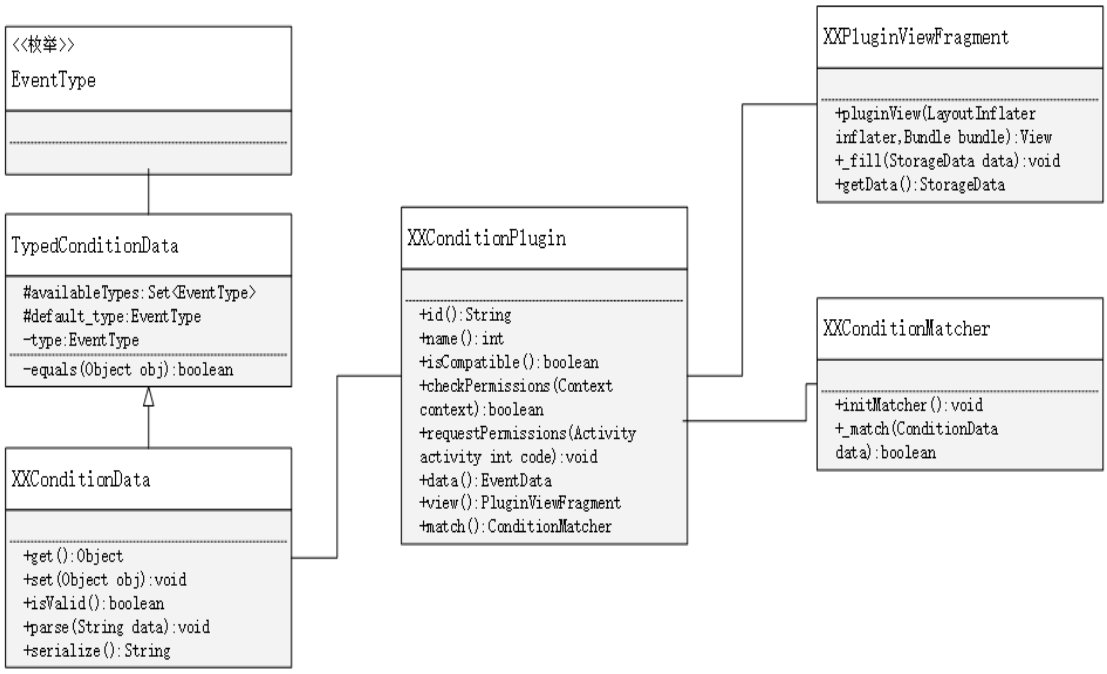

图 4.3 条件组件类图

ConditionData 是事件组件的数据模型，其数据结构与 EventData 类似，包括

逻辑和数据两个组成部分，以及数据的序列化与反序列化方法。

ConditionMatcher 是条件组件的验证单元，当需要验证某一设备或者服务是否满足条件时，需要通过该组件生成对应的验证单元，并将数据 data 传入进行验证。不同的条件组件对于_match 的实现方法可以各不相同，耗时类任务可以开启新线程进行验证，非耗时的可在主线程中进行验证，最终基于观察者模式将结果验证结果返回给条件管理器。

4.1.3 操作组件设计

操作组件在实现上与上述两类组件略有不同，操作组件是需要在运行过程中主动调用并执行的，不需要在服务开启阶段就执行初始化，且所有的操作都是组件的一次性操作，每次需要执行操作时，只需要将操作数据传入到操作管理器中，操作管理器会自动调用与之对应的操作组件。操作组件的类图如图 4.4 所示。

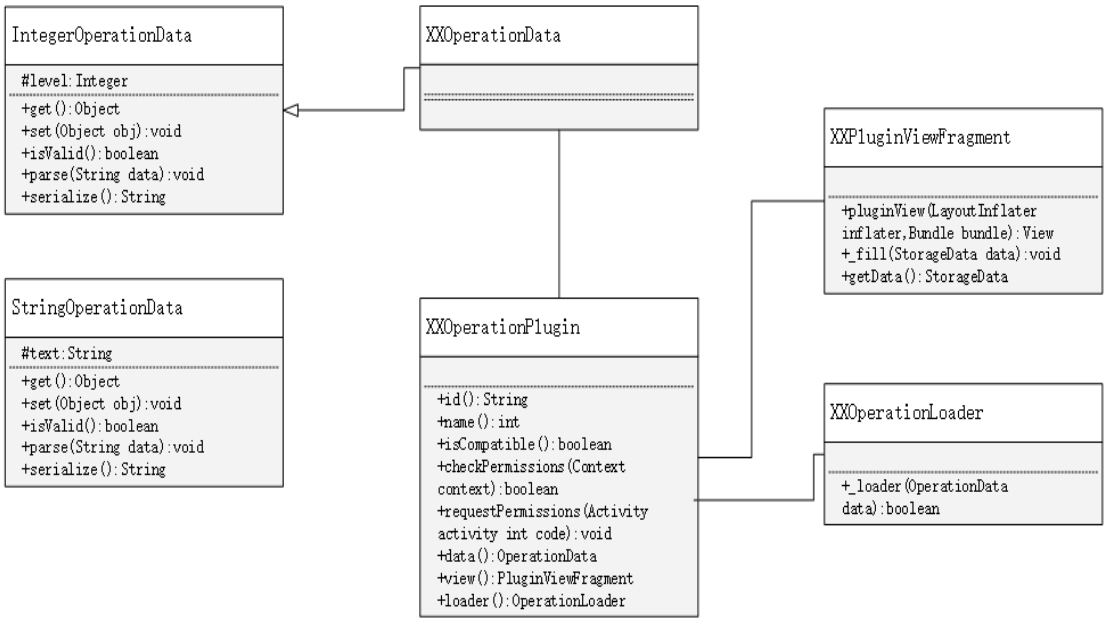

图 4.4 操作组件类图

OperationData 是操作组件的数据模型，是可操作设备能够被修改的状态数据。操作组件被描述为“使设备处于什么状态”，根据不同类型的软硬件设备表现状态的方式不同，可以将组件操作数据 OperationData 分为 Integer 和 String 两种类型，而具体操作则是将这些数据转化为软硬件设备能够支持的命令，例如智能设备控制就是基于 String 类行动操作码，因此只需要将 String 类型的操作码以

其可以识别的方式发送给设备就可以完成操作。

OperationLoader 是操作组件的执行单元，是本文系统控制设备或服务的主要接口。当需要改变某个设备或者服务的状态时，需要生成该组件的执行单元，并将所要改变的状态数据封装成设备可识别的格式，通过设备相关的通信技术，将设置数据传递给具体设备，实现对相应设备的数据设置。

4.2 服务组件注册

每个组件对应于一个硬件或者软件功能，可能是 Android 系统自带的功能如时间、日期、网络状态、短信、电话等等，也可能是通过 WiFi、蓝牙、局域网等通信技术接入的外部设备，如智能手环、蓝牙设备、智能手机等等，甚至是网络服务。为了统一化管理所有的的组件，需要对这些组件进行功能区分，即事件组件，条件组件和操作组件，然后分别对组件进行分类注册，而三类组件的注册对象都实现自组件注册模型，只是在管理的对象上存在差别。组件注册模型包括组件的动态注册方法，组件类列表以及组件对象列表，具体实现如下：

组件注册表

```
class PluginRegistry {
    final List<Class<? Extends PluginDef>> pluginClassList;
    final List<PluginDef> pluginList;
    Synchronized registerPlugin(Class<? Extends PluginDef> pluginClass){
        for (Class<? Extends PluginDef> klass: pluginClassList) {
            If( klass == pluginClass) return;
        }
        pluginClassList .add(pluginClass);
        pluginList.add(pluginClass.newInstance());
    }
}
```

基于该注册模型，三类组件可以分别注册到三个组件注册对象中，实现组件的分类及统一管理。当三类组件管理需要查询具体的组件时，可以调用注册类中

的组件查询方法，也可以获取组件的注册列表进行遍历，从而确定具体的组件实现。通过这样的模型设计可以实现规则引擎在调度组件时不区分具体的组件功能，只有在需要执行组件的方法时，才会动态生成组件对象，并执行响应的方法，提高了系统的兼容性和扩展性，降低了系统本身与具体设备之间的耦合程度。

4.3 服务组件实现

通过组件模型设计以及注册模型设计，一方面可以降低系统与具体设备或服务的耦合程度，提高系统的扩展性，另一方面可以降低用户在编辑服务过程中操作的复杂程度。为了进一步简化服务组件的实现过程，本文根据服务组件依赖的设备类型，将所有的服务组件划分为系统类型组件，网络类型组件以及外部设备组件，并为每种类型的服务组件提供实现方案。

4.3.1 系统类型组件实现

系统类型组件可以是Android手机系统能提供的服务功能，例如时间、日期、网络、广播、短信、电话、闹铃等等，也可以是 Android 手机中支持全局广播的手机应用，本文将这些类型的服务统称为系统服务。系统服务较为简单，不需要其他外部设备支持，只要获得相关的权限就可以调用这些服务，非常实用，能够在很多方面帮助残疾人改善生活，例如定时提醒事件、定时操作某种设备、自动收发短信、自动接听电话、铃声报警等等。无论是辅助残疾人日常，还是监控环境安全都非常有用，且不需要付出额外代价。

设计实现这类组件的主要途径是实现组件与系统服务之间的数据通信，包括监听系统服务的某项功能状态和调用系统服务的某项功能。具体的实现方法很多，但主流的方法主要有两个，分别是广播通信和调用系统服务管理对象。

广播通信的实现方式较为普遍，因为 Android 系统中大部分的功能都设置了广播发送者和广播接收者，只需要定义与之对应的广播接收者或广播发送者，即广播过滤器 filter 和必要的参数，就可以实现与相关系统服务之间的间接通信。而对于 Android 系统中其他的应用也可以通过定义全局广播实现不同应用之间的信息共享。具体实现原理如图 4.5 所示。

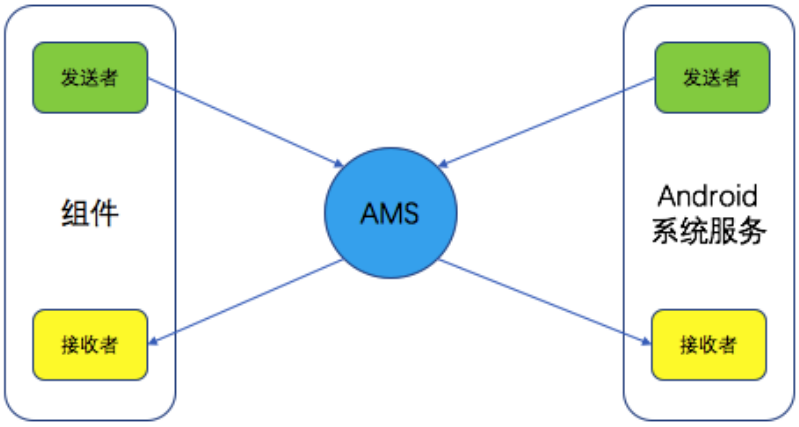

图 4.5 广播通信原理图

实现与 Android 系统服务通信的另一有效方法是调用对应的服务管理类。通过该方法可以直接监听和修改该服务的状态，但是这类实现方法只适用于 Android 系统自带的功能，不利于非系统类应用，具体实现原理如图 4.6 所示。当需要调用系统功能时，首先通过系统服务对应的 key 值去查找对应的服务管理类 Manager，该 Manager 拥有对应系统服务对象的管理和控制权限，并向外提供设置和获取数据的方法。

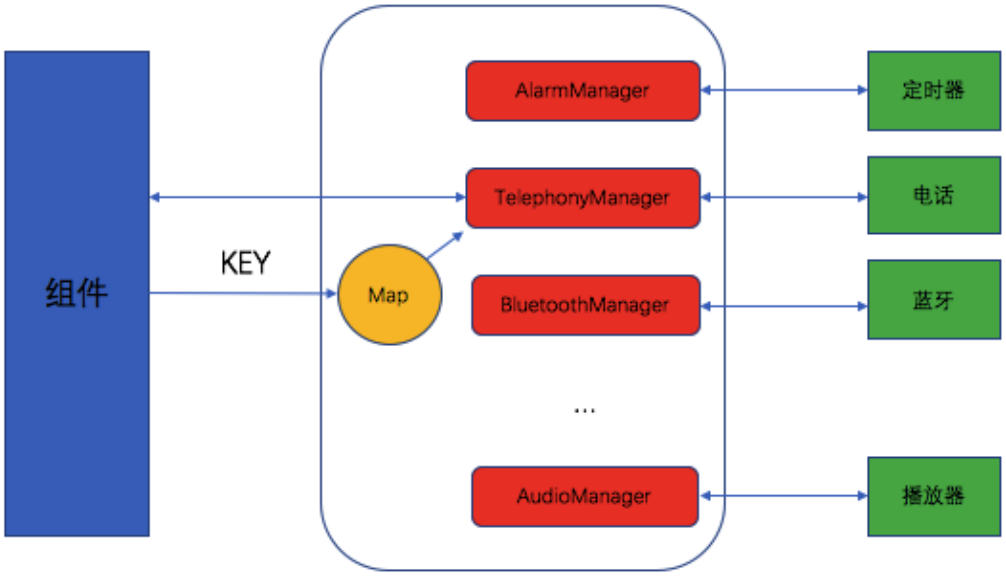

图 4.6 Android 系统服务通信原理图

4.3.2 网络类型组件实现

网络组件是指需要通过网络与具体的服务网站之间进行通信的功能组件，一般可以作为事件组件和条件组件，并不需要依赖具体的硬件设备，只需要保证 Android 手机处于联网状态就可以正常使用。这类组件可以为残疾人提供丰富的网络资源，能够有效提升残疾人的生活质量，且不需要付出太多的代价。

由于这类组件需要通过互联网与服务网站进行通信，因此在数据通信过程中往往存在一定的延迟，数据的准确性并不高。为了保证系统可以快速响应事件和验证条件，本文将这一类组件设计为缓存类的组件，其结构如图 4.7 所示。针对每个网络服务需要设计一个与之通信的接口，该接口可以与网络服务进行数据通信，获取网络数据，并将获取的数据封装成数据对象缓存在系统本地。当与该网络服务相关的组件需要获取数据信息时，只需要获取缓存数据即可。这样设计网络服务组件一方面可以避免重复建立网络连接导致的资源浪费，另一方面可以保证条件组件快速响应验证请求。

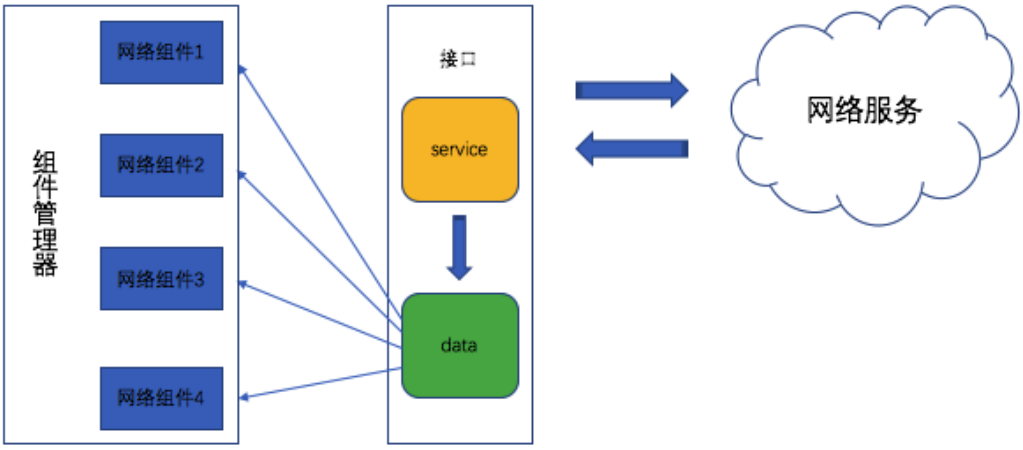

图 4.7 网络服务结构图

4.3.3 外部设备组件实现

外部设备组件是指需要依赖特定硬件设备的才能正确执行的服务组件，不同于前两种服务组件，该类型组件与硬件设备的耦合程度高，如果不通过合适的方法进行管理，将严重影响本文系统的适用范围。

为了降低本文系统与特定硬件设备的耦合程度，需要对所有的外部设备进行统一管理，用户可以根据自身拥有设备情况进行选择性添加，从而适用于不同的残疾人家居环境。

实现统一管理的方法，需要所以外部设备实现外部设备接口定义，其结构如下所示：

外部设备类结构

```
abstract class DeviceDef {
        String name ()
        void searchDevice (OnSearchListener listener, Context context)
        void connect (Context context)
}
```

DeviceDef 是所有外部设备需要实现抽象类，其中 name 是外部设备的名称，也是唯一标识符，用于区分不同设备。searchDevice 是当用户需要填加设备时调用的设备搜寻方法。connect 是当设备被加入系统时调用的方法，用于将该硬件设备加入可用外部设备列表中。除了这三个方法外，还需要根据具体的硬件设备实现数据通信方法和其他方法，才能保证外部设备组件的正常使用。

当外部设备实现 DeviceDef 后，需要将设备注册到设备列表中，这样用户才可以在外部设备列表中查看该设备状态，并选择是否启用该设备。外部设备注册模型如下所示：

设备注册表

```
class DeviceRegistry {
    final List<Class<? Extends DeviceDef>> classList;
    final List< DeviceDef > objectList;
    synchronized registerDevice(Class<? Extends DeviceDef > clazz){
        For (Class<? Extends DeviceDef > iclazz: classList) {
            If( iclazz == clazz) return;
        }
        classList.add(clazz);
        objectList.add(clazz.newInstance());
```

```
        }
    }
```

当外部设备定义并实现 DeviceDef 接口，并将其注册到外部设备注册表中，用户仍然无法在编辑自动化服务时看到该设备对应的组件。这样做是为了保证系统的稳定性和扩展性，将具体设备的使用权完全移交到用户手中，只有用户选择将设备加入系统，该设备才可以被系统真正识别和使用。而用户主动添加设备的过程主要是调用 DeviceDef 的 connect 方法，该方法与具体的设备连接方式相关，可以是基于 UDP 或 TCP 连接，也可以通过蓝牙等其他方式，实现方法各异但实现过程相似。

4.4 小结

为了屏蔽不同设备或服务底层实现的复杂性，为用户提供统一便捷的服务编辑方式，本章介绍了面向终端用户开发的服务组件实现方法。首先基于唯一事件源的事件触发规则，分别对事件组件、条件组件以及操作组件进行模型设计，并实现对应服务组件的注册类，以管理各类的服务组件。随后，根据服务组件所依赖的设备类型，介绍了系统类型组件，网络类型组件以及外部设备组件的实现方法和优化策略，增强系统的扩展性。

第五章 基于终端用户开发的残疾人智能家居系统实现与评估

前两章主要介绍了 Android 系统系下事件触发规则引擎与服务组件化设计，基于前两章的工作，本章进一步实现了基于终端用户开发的残疾人智能家居系统，并从系统性能和用户评价两个角度对系统进行了评估。

5.1 系统功能设计与实现

5.1.1 服务管理功能

服务管理功能是管理当前用户所有自动化服务的主界面，用户可以在该界面查看到所有的家居自动化服务，并控制每个服务的激活状态，其运行界面如图 5.1 所示。用户可以通过该界面删除和编辑已有的服务，也可通过右下方的按钮来打开增加新服务的界面。

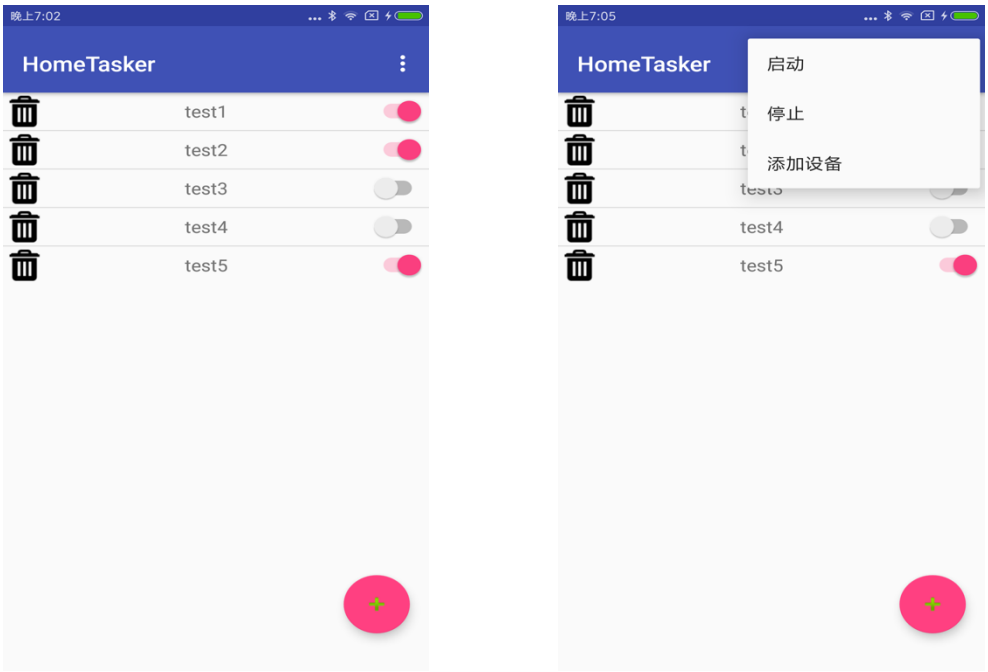

图 5.1 服务管理界面

该界面除了可以编辑和修改服务，还可以控制所有激活服务的启动和停止，只有在用户启动服务后，才会真正激活事件触发规则引擎去调度激活的服务，监测和控制家居环境。

5.1.2 设备管理功能

设备管理是管系统外部设备的操作界面。由于残疾人个体需求和家庭的环境不同，他们对于家居设备和智能设备的需求有很大差异，因此无法确定他们所拥有的设备情况。本文系统为了适应这种差异化需求，将所有设备组件化，并将外部设备统一管理。新的设备可以被加入系统之中，但是并不会强制要求用户配置该设备，用户可以根据自己的实际需求来进行选择，如果拥有设备且想将设备加入到自己的自动化服务中，用户可以通过设备管理功能来配置该设备。通过这种设计可以将系统支持的功能与用户可以使用的功能区分开，使得系统可适用于更多类型的家居环境，满足残疾人家庭的个性化环境配置。

设备管理界面如图 5.2 所示。用户可以查看当前系统支持的所有外部设备，以及这些设备是否已经被加入到当前环境之中。用户可以点击设备来实现设备连接和设备移除，其中设备连接过程与具体的设备有关，配置过程各有差异。

图 5.2 设备管理主界面

5.1.3 服务编辑功能

服务编辑界面是残疾人用户及其家庭成员编辑家居自动化服务的界面。该界面基于事件触发规则设计，如图 5.3 所示，从左到右依次是事件组件编辑界面、条件组件编辑界面和操作组件编辑界面。不同于以往的自动化编辑工具，本文系统在编辑自动化服务时并不要求事件、条件和操作的编辑顺序，用户可以自由在各个编辑界面间进行切换，而不影响其他编辑结果，进一步简化了用户的编辑过程，也增加了用户在编辑过程中的自由调配程度。

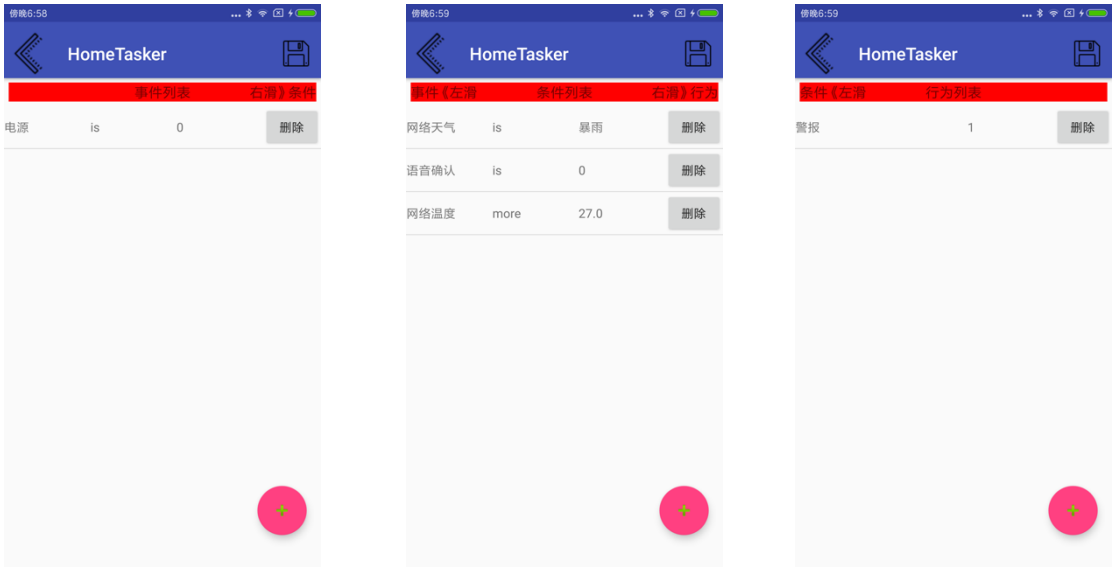

图 5.3 组件编辑界面

如图 5.3 中所示，每个界面都拥有添加组件的按钮，通过添加按钮可以打开该类型组件的列表。在服务配置过程中需要注意的是事件组件必须有且只能有一个，条件组件可以没有，也可以添加多个，操作组件至少有一个。当用户配置完成后就可以点击保存按钮，设置该服务的基本信息，如图 5.4 所示，需要保证方案名称唯一。

图 5.4 服务保存界面

5.2 系统性能评估

本文为了保障系统使用的便捷性，以及降低自动化服务的实现成本，最终系统是基于 Android 系统设计和实现。Android 移动设备不但拥有丰富的传感器资源以及网络通信支持，而且价格低廉，普及化程度高，非常契合残疾人的生活需求。然而，Android 移动设备相对传统计算机设备而言资源较为匮乏，内存较小，cpu 性能差，因此为了保证本文系统能够稳定的为残疾人提供服务，对系统进行性能评估是必不可少的。

5.2.1 Android 性能评估工具-Emmagee

Emmagee[46]是由网易杭州研究院 QA 团队开发的一款 Android 应用性能检测工具，可以监控单个 APP 的 cpu、内存、流量、启动耗时、电量等等性能变化。该工具的实现原理是通过获取待检测 app 的进程 id（PID），然后调用 Android 系统服务类如 WindowManger、MemoryInfo、CpuInfo 等获取相应的性能数据，最后生成性能报表。

该测试工具使用和配置方法较为简单。首先需要将 Emmagee 下载并安装到 Android 手机上，然后打开 Emmagee 就可查看到当前 Android 手机所安装是应

用，选中所要测试的 APP 就可以自动启动 app 并进行测试了。Emmagee 默认的采样频率是 5 秒，用户也可以通过设置功能来调整采样频率，并且可以设置邮箱将生成的性能报表发送到指定邮箱。

Emmagee 生成的性能报表是 csv 格式文件，其内容包括当前测试的机型、硬件设备信息、应用信息、以及内存消耗、cup 占用、流量、电压等等。通过这些数据可以基本观测出所测试应用运行过程中的性能状况。

5.2.2 实验设计

Android 应用性能测试相对而言拥有很多的不确定性，其中主要包含如下几个方面：

Android 系统和机型差异。Android 系统及相关技术之所以可以快速发展，主要原因是 Android 系统源码是开源的，可以让更多的技术人员去完善和修改它。然而这种开源化导致的问题也较为明显，不同 Android 硬件厂商会根据自身的硬件技术来修改系统，导致系统的多样化，使得 Android 应用适配难度增加，且性能数据差异较大。

Android 系统内存管理机制。作为面向移动端设备的智能系统，Android 系统设计之初就考虑到移动设备资源有限的问题，为此 Android 系统采用了特殊的内存管理机制：当某一应用被关闭后，系统并不及时回收该应用的内存资源，Linux 内核中的进程也依然存在，直到系统内存资源不足时，才会主动回收这部分内存。这种内存管理机制存在很大优势，首先可以保证用户再次启动应用时可也快速响应，其次可以实现内存资源的动态管理，优化系统资源分配。因此，开发人员常常在 Android 系统性能测试时发现同一应用在同一手机上每次启动时其内存消耗会存在一定程度的差异。

虽然，并不能保证每次性能测试时内存资源消耗的数据一致，但是如果系统稳定那么这个差值相差不会太大，且随着时间的推移系统消耗总会维持在一个相对稳定的状态。

为了验证本文系统性能的稳定性，本文对比分析了两个同类商业 APP（Atooma，Tasker）的性能数据，实验设计如下：

1、保证使用的组件和实现的服务基本相同。本文系统实现的组件功能和其他两个 app 在功能上存在差异，因此功能组件各不相同，不同组件所占用的系统资源存在差异，因此组件不同可能会导致实验结果差异很大。

2、对比无服务和有服务运行时性能数据变化。Android 应用由于 UI 和组件会消耗系统资源，所以不同应用难以真正对比数据，但是可以通过状态变化的相对值来评估系统的性能差异。

3、随机多次采样，取均值。Android 系统具有较高的随机性，具体取决于系统的运行状态，以及资源分配状况，因此需要多次采样数据，来保证数据的可靠性。

根据上述性能实验设计，本文首先对比三个应用在无服务状态的性能数据，然后分别测试三个应用在服务启动状态下性能数据的变化。其过程保证服务功能基本类似，其次所有数据都多次采样，并去除最小值和最大值，求取平均值，排除异常数据。

5.2.3 实验结果

依照实验设计，本文通过 Emmagee 分别采样了三个 app 在无服务状态下的性能数据。主要包括 app 的内存消耗，cup 占用状况两个方面。

图 5.5 是三个 app 在无服务状态下的内存占用情况。从内存占用曲线可以看出，三个 app 在无服务状态下内存占用都相对稳定，只是在数值上存在细微差距，其差距可能与应用的 UI 和 Android 组件的数量相关。

图 5.6 是三个 app 在无服务状态下的 cpu 占用情况。从图中曲线可以看出三个 app 在无服务状态下 cpu 占用都基本为 0，Atooma 和 Tasker 偶尔存在波动，但幅度较低基本可以忽略。

总体而言 HomeTasker 在无服务状态其性能数据稳定，且相对于两个商业应用而言由于没有复杂的 UI，性能消耗更低。

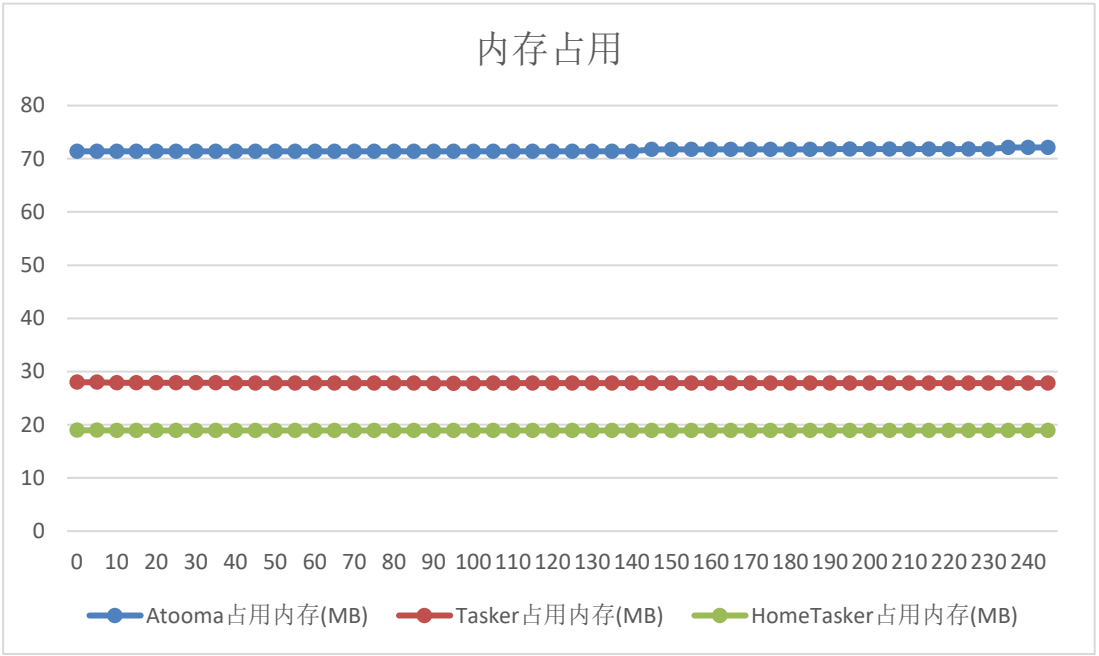

图 5.5 内存占用曲线图

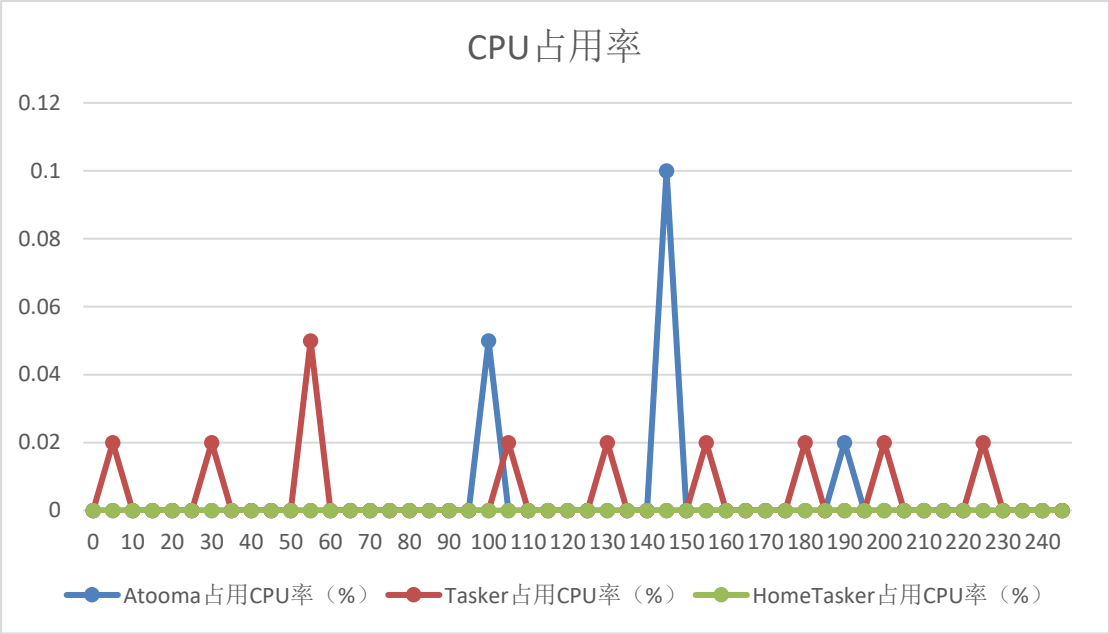

图 5.6　cpu 占用曲线图

在对比完三个 app 在无服务状态下的性能数据后，本文分别对三个 app 在执行多服务状态下的性能数据进行了测试。

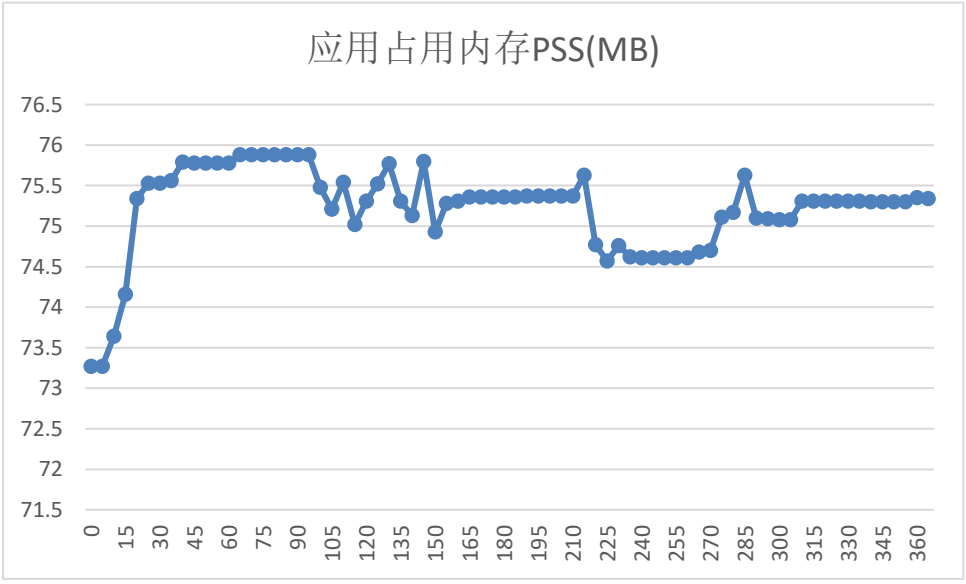

图 5.7 Atooma 服务执行过程内存占用曲线图

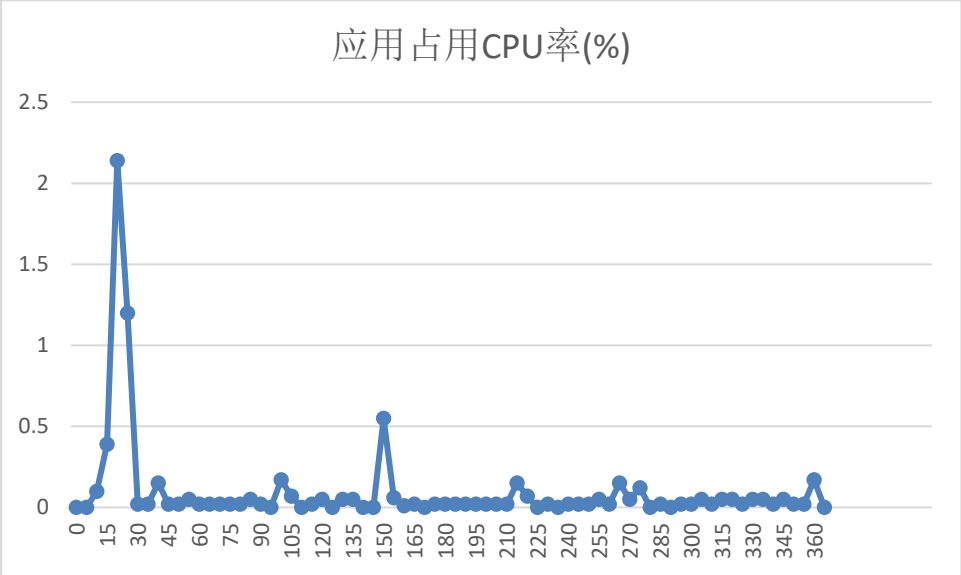

图 5.8 Atooma 服务执行过程 cpu 占用曲线图

图 5.7 和图 5.8 分别是 Atooma 多任务执行状态下的内存和 cpu 占用数据，可以看出在当服务增加后其内存数据略有提高，随着服务启动 cpu 数据存在明显波动，且内存数据也明显上升，随后又趋于稳定，虽然仍有波动，但幅度较小。从数据变化可以看出，Atooma 是在服务启动时将服务相关所有数据加载至内存，因此后续执行时性能数据变化不大。

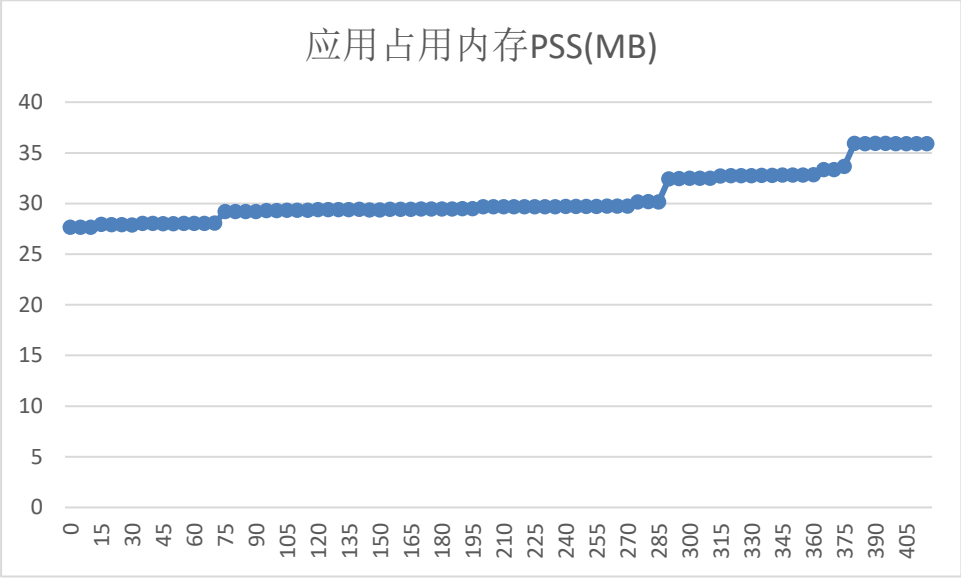

图 5.9 Tasker 服务执行过程内存占用曲线图

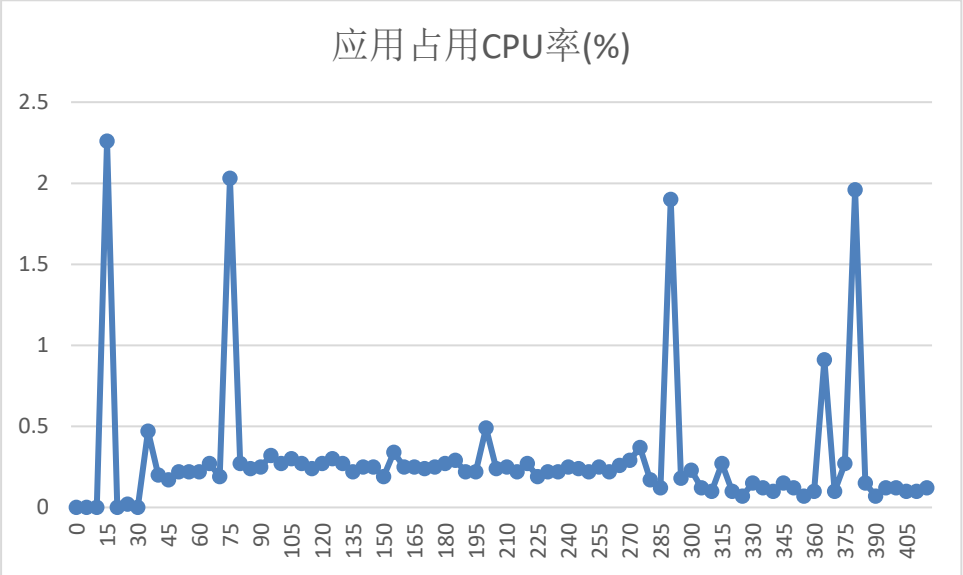

图 5.10 Tasker 服务执行过程 cpu 占用曲线图

图 5.9 和图 5.10 分别是 Tasker 在多服务执行状态下的内存和 cpu 的占用数据，可以看出在未开启服务时，Tasker 内存数据与无服务状态下的数据相近，当服务开启后期内存数据变化也不大。但是，随着服务被响应，Tasker 的 cpu 出现波动，且内存消耗也随之增加。因此，可以看出 Tasker 是在服务被响应后才会真正将与该服务相关的数据加载到内存并执行，这一点与 Atooma 不同，但是总体而言其数据变化仍较为稳定。

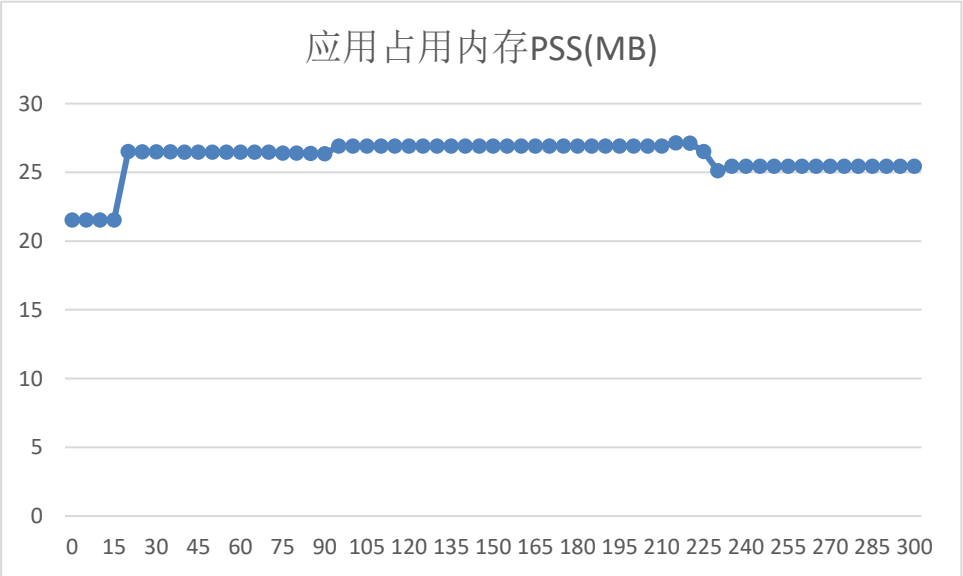

图 5.11 HomeTasker 服务执行过程内存占用曲线图

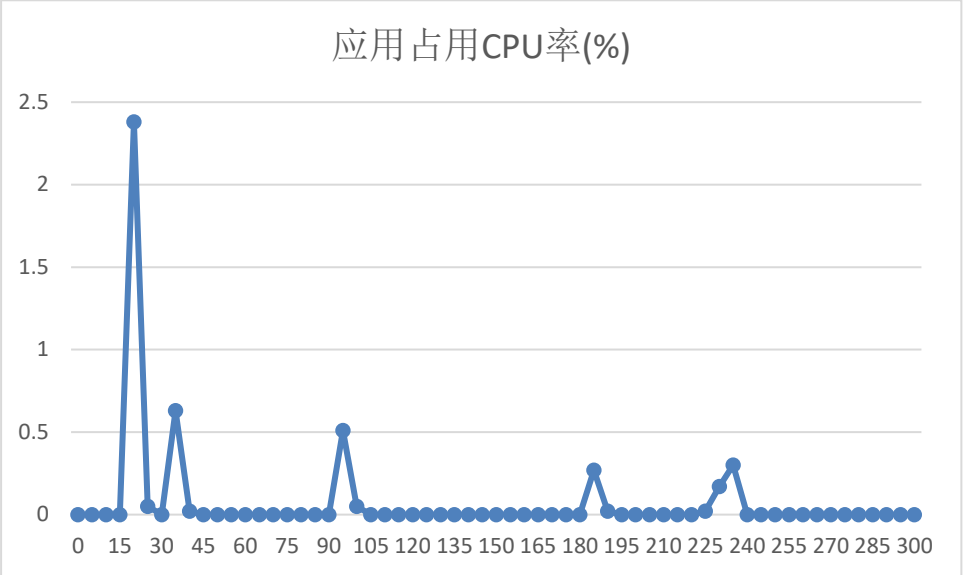

图 5.12 HomeTasker 服务执行过程 cpu 占用曲线图

图 5.11 和图 5.12 分别是本文系统 HomeTasker 在多服务下内存和 cpu 占用数据，可以看出 HomeTasker 的性能数据变化与 Atooma 相似。因为 HomeTasker 在服务启动后将激活的服务一次加载如内存，并进行配置，所有后续服务可以被快速响应，且不会在增加内存的消耗。

通过对比三个 APP 在不同执行状态下的性能数据，可以看出本文系统 HomeTasker 可以和现有的系统一样能够稳定提供服务，且资源消耗相对较低。

5.3 用户评价

最后，为了了解本文系统能否如预期一样可以被最终用户接受并使用，从而改善残疾人的家居生活，本文做了一项用户调查。此次调查邀请了《重症残疾人无障碍自理智能化控制与康复提醒系统》科研项目的最终用户、相关服务工作人员以及在读学生共计 5 人来试用本系统，然后从系统易用性、表达力、正确率以及使用体验对系统进行评分，评分范围为 1-5 分，最高分为 5 分，最低分为 1 分。

整个调查过程分为两步。第一步，我们为参与测试的用户提供一段 HomeTasker 的系统介绍视频，包括系统的主要功能，所有操作界面，目前支持的组件，以及编辑服务的过程和注意事项。当所有的准备工作结束后，参与测试的用户可以根据自己的意愿编辑至少 5 条不同的服务，并在我们的帮助下对这些服务进行正确性检测，我们记录其服务数量和总体的正确率。第二步，所有参与

试用的用户，我们都会邀请他们填写一份调查问卷，主要内容如表格 5.1 所示。

表 5.1 用户调查问卷

1、学习使用 HomeTasker 是否容易（5）	
2、HomeTasker 是否简单易用（5）	
3、HomeTasker 是否可以满足您的日常需求（5）	
4、您对 HomeTasker 的使用方式是否满意（5）	
5、您总共编辑了几条服务	
6、您编辑的服务可以正确使用的有几条	

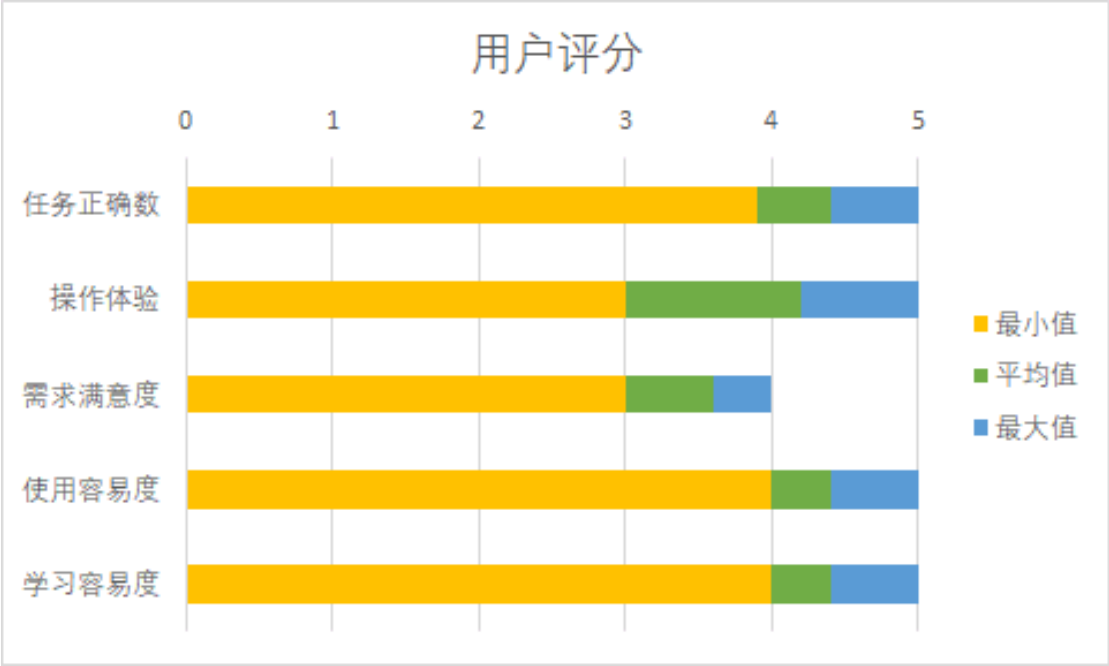

图 5.13 系统用户评分

针对测试用户反馈的数据，在进行一定处理后，其结果如图 5.13 所示。根据用户的评分结果，可以看出 HomeTasker 使用难度可以被接受，用户编辑的服务正确率也较高，但是图表中问题 3 的平均分数相对较低，表明 HomeTasker 的表现力还存在不足。随后我们进一步与部分参与测试的用户进行交流，这些用户表示 HomeTasker 所支持的组件相对较少，能够连接的设备有限，还不足以满足他们所有的日常需求，并希望可以将更多设备加入到系统之中。

通过此次用户调查显示，HomeTasker 在易用性，任务正确率以及操作体验上

表现良好，但是在系统表现力上与用户期望仍然存在一定差距，其主要原因是目前系统支持的设备较少，组件还不够丰富，这需要后期长期的努力来不断完善。

5.4 小结

本章首先介绍了本文系统 HomeTasker 的主要功能，以及使用方法。随后，利用 Android 系统性能监测工具，对比主流的终端用户开发工具，评估了 HomeTasker 的系统性能，结果基本符合预期，验证了系统的稳定性。最后，我们邀请了一些用户试用 HomeTasker，并从多个角度对系统进行评分，结果基本符合预期，验证了系统的易用性。

第六章 总结和展望

6.1 总结

本文针对目前残疾人智能家居系统存在个性化程度不高，难以满足残疾人复杂性需求的问题，设计了一款基于终端用户开发的残疾人智能家居系统。通过该系统，残疾人及其家庭成员可以编辑符合残疾人实际需求的家居自动化方案，在一定程度上实现家居环境的个性化定制，改善残疾人的家居生活环境，提高残疾人独立家居生活的能力。结合目前智能手机设备的便捷性和丰富的功能支持，能够为用户提供丰富的交互方式，非常符合残疾人的实际需求，因此本文系统是基于 Android 系统设计和实现的，并在设计和实现过程中进行了性能优化，保证系统的稳定性。本文主要工作包括以下几个方面：

1、通过调查了解我国目前残疾人的生活状况，分析残疾人生活所面临的主要问题。然后进一步调研目前国内外致力于改善残疾人生活的相关研究工作，以及较为成熟的技术成果。结合残疾人现状，分析总结目前这些技术面临的主要弊端，提出基于终端用户开发技术的智能家居系统解决方案。

2、调研目前主流的终端用户开发技术，分析事件触发规则的可行性，结合 Android 系统的特性，设计和实现 Android 系统下事件触发规则引擎，包括完整的服务模型定义，三类组件管理器的设计，以及各个功能模块间通信方式的设计。

3、基于唯一事件源的事件触发规则，本文将系统的服务组件划分为事件组件、条件组件和操作组件，根据每一种类型的组件特性，设计相应的数据模型、UI 界面、组件结构以及执行方式。基于这样模型设计，一方面可以保证系统的稳定性和扩展性，另一方可以简化用户编辑自动化服务的复杂程度。

4、设计和实现了基于终端用户开发的残疾人智能家居系统，具体包括服务管理、设备管理和服务编辑三大功能。并根据残疾人的实际生活需要，将残疾人智能家居设备、网络天气服务、语音手势等功能进行组件化融入系统之中，完善了系统功能。

6.2 展望

本文主要针对目前残疾人智能家居系统所面临的问题提出了一种基于终端用户开发的智能家居系统解决方案，并实现了事件触发规则引擎和服务组件化模型，最后基于残疾人无障碍自助管理平台的硬件设备实现了部分的服务组件来验证系统功能。虽然，目前系统主要的核心部分已经完成，但是与残疾人的实际需求以及系统的最终期望仍然有一定差距，还需要从多个方面来完善系统：

1、丰富服务组件。系统想要为更多的残疾人服务，就必须提供更多的残疾人相关的服务组件。因此需要进一步调研残疾人日常生活中的智能设备，并与这些设备的制造商沟通，尽可能尝试为系统增加更多的设备组件。另一方面也需要对各类残疾人相关的网络服务做调研，将残疾人相关的网络服务加入系统，为残疾人提供生活和工作支持。

2、增加安全策略。家居安全是残疾人智能家居系统应该关注的问题，虽然本文系统提供报警和通知组件，但是缺乏相关的安全监控组件，需要在后续的工作中重点考虑这方面的组件设计。

3、错误校验机制。由于本文系统的使用对象是残疾人、残疾人家人和残疾人服务工作者，这些人往往缺乏一定的计算机知识基础，所有为了保障他们能够编辑出正确可行的自动化方案，应该为他们提供错误校验支持。

参考文献

[1] 国家统计局.第二次全国残疾人抽样调查主要数据公报[EB/OL]. http://www.stats.gov.cn/tjsj/ndsj/shehui/2006/html/fu3.htm

[2] 中国残疾人联合会[EB/OL]. http://www.cdpf.org.cn/zcwj/zxwj/201703/t20170331_587445.shtml

[3] Atzori L, Iera A, Morabito G. The Internet of Things: A survey[J]. Computer Networks, 2010, 54(15):2787-2805.

[4] Dengler S, Awad A, Dressler F. Sensor/Actuator Networks in Smart Homes for Supporting Elderly and Handicapped People.[C]// International Conference on Advanced Information NETWORKING and Applications Workshops. IEEE, 2007:863-868.

[5] Haryanto R. Context-Awareness in Smart Homes to Support Independent Living[J].

[6] He J, Zhang Y, Huang G, et al. A smart web service based on the context of things[J]. Acm Transactions on Internet Technology, 2012, 11(3):1-23.

[7] Rasch K, Li F, Sehic S, et al. Context-driven personalized service discovery in pervasive environments[J]. World Wide Web-internet & Web Information Systems, 2011, 14(4):295-319.

[8] Sohn M, Jeong S, Lee H J. Self-Evolved Ontology-Based Service Personalization Framework for Disabled Users in Smart Home Environment[C]// Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. IEEE Computer Society, 2013:238-244.

[9] Edwards W K, Grinter R E. At Home with Ubiquitous Computing: Seven Challenges[C]// International Conference on Ubiquitous Computing. Springer-Verlag, 2001:256-272.

[10] LG, “LG TO SHOWCASE CONNECTED, EASY-TO-CONTROL SMART HOME APPLIANCES AT CES 2013,” http://www.lgnewsroom.com/newsroom/contents/62851, 2013

[11] Kong Q, Suyama T, Suyama T, et al. Selecting home appliances with smart glass based on contextual information[C]// ACM International Joint Conference on Pervasive and Ubiquitous Computing. ACM, 2016:97-108.

[12] 孔繁荣. 基于物联网的残疾人智能家居的设计与实现[J]. 软件产业与工程, 2014(3):42-46.

[13] 詹培忠, 汪江林, 杜煜. 一种眼控智能家居装置:, CN205427468U[P]. 2016.

[14] 杨忠豪. 面向残疾人的眼动交互技术研究[D]. 浙江大学, 2015.

[15] Kirkpatrick K. Existing technologies can assist the disabled[J]. Communications of the Acm, 2016, 59(4):16-18.

[16] EASTIN [DB/OL]. http://www.eastin.eu/en/searches/products/iso/.

[17] ABLEDATA [DB/OL]. http://www.abledata.com/manufacturers-and-distributors.

[18] Cook D J, Youngblood M, Iii E O H, et al. MavHome: An Agent-Based Smart Home[C]// IEEE International Conference on Pervasive Computing and Communications. IEEE Computer Society, 2003:521.

[19] Robles R J, Kim T H. Applications, Systems and Methods in Smart Home Technology: A Review[J]. International Journal of Advanced Science & Technology, 2013, 15.

[20] Rivera-Illingworth F, Callaghan V, Hagras H. A neural network agent based approach to activity detection in AmI environments[C]// Intelligent Environments, 2005. the Iee International Workshop on. IEE, 2005:92-99.

[21] Hussein A, Adda M, Atieh M, et al. Smart Home Design for Disabled People based on Neural Networks ☆[J]. Procedia Computer Science, 2014, 37:117-126.

[22] Maceli M G. Tools of the Trade: A Survey of Technologies in End-User Development Literature[C]// International Symposium on End User Development. Springer, Cham, 2017:49-65.

[23] Abraham R, Erwig M. Goal-Directed Debugging of Spreadsheets[C]// IEEE Symposium on Visual Languages and Human-Centric Computing. IEEE Computer Society, 2005:37-44.

[24] Desolda G, Ardito C, Matera M. Empowering End Users to Customize their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools[J]. ACM Transactions on Computer-Human Interaction, 2017, 24(2):58.

[25] Fischer G, Mccall R, Ostwald J, et al. Seeding, evolutionary growth and reseeding[C]// the SIGCHI conference. 1994:292-298.

[26] Fischer G. Domain-oriented design environments[J]. Automated Software Engineering, 1994, 1(2):177-203.

[27] Lieberman H, Paternò F, Klann M, et al. End-User Development: An Emerging Paradigm[D]. Springer Netherlands, 2006.

[28] Ko A J, Abraham R, Beckwith L, et al. The state of the art in end-user software engineering[J]. Acm Computing Surveys, 2011, 43(3):194-218.

[29] Blackwell AF (2004) End-user developers at home. Communications of the ACM 47(9):65–66. doi:10.1145/ 1015864.1015892

[30] Dahl Y, Svendsen R-M (2011) End-user composition interfaces for smart environments: a preliminary study of usability factors. In: Marcus A (ed) Design, user experience, and usability. theory, methods, tools and practice, vol 6770. Springer, Berlin Heidelberg, pp 118–127. doi:10.1007/978-3-642-21708-1_14

[31] Dey AK, Sohn T, Streng S, Kodama J (2006) ICAP: interactive prototyping of context-aware applications. In: Fishkin KP, Schiele B, Nixon P, Quigley A (eds) Pervasive computing, vol 3968. Springer, Berlin Heidelberg, pp 254–271. doi:10.1007/11748625_16

[32] Litvinova E, Vuorimaa P (2012) Engaging end users in real smart space programming. Proceedings of ACM Conference on Ubiquitous Computing (UbiComp), Pittsburgh, Pennsylvania, 1090–1095. doi: 10.1145/ 2370216.2370447

[33] J. F. Pane, C. A. Ratanamahatana, and B. A. Myers. 2001. Studying the language and structure in non- programmers' solutions to programming problems. International Journal of Human-Computer Studies 54, 2, 237–264.

[34] IFTTT [EB/OL]. https://ifttt.com

[35] Tasker [EB/OL]. https://tasker.joaoapps.com

[36] Cabitza F, Fogli D, Lanzilotti R, et al. Rule-based tools for the configuration of ambient intelligence systems: a comparative user study[J]. Multimedia Tools & Applications, 2017, 76(4):1-21.

[37] Ur B, Mcmanus E, Melwyn P Y H, et al. Practical trigger-action programming in the smart home[J]. 2014:803-812.

[38] Lucci G, Paternò F. Understanding End-User Development of Context-Dependent Applications in Smartphones[M]// Human-Centered Software Engineering. Springer Berlin Heidelberg, 2014:182-198.

[39] Drools [EB/OL]. https://www.drools.org/

[40] OpenRules [EB/OL]. http://openrules.com/

[41] WebSphereJRules [EB/OL]. https://www-01.ibm.com/software/integration/business-rule-management/jrules-family/

[42] JESS [EB/OL]. http://www.jessrules.com/

[43] Fogli D, Lanzilotti R, Piccinno A. End-User Development Tools for the Smart Home: A Systematic Literature Review[M]// Distributed, Ambient and Pervasive Interactions. Springer International Publishing, 2016.

[44] Vorapojpisut S. A Lightweight Framework of Home Automation Systems Based on the IFTTT Model[J]. Journal of Software, 2015.

[45] Atooma [EB/OL]. http://appcircus.com/apps/atooma

[46] Emmagee [EB/OL]. http://code.netease.com/project/Emmagee.html

致谢

时光飞逝，一转眼研究生生活就已经接近尾声了。犹记得当初怀着对科研与学术的憧憬，进入了复旦大学计算机学院的协同信息与系统实验室，三年的情景历历在目。这三年的时间里，经历过研究方向的迷茫，也体会过实现科研项目的种种困难，但是在老师和同学的帮助下，让我坚定了方向，并不断成长。在此，我要对所有实验室老师和同学表由衷的感谢。

首先，我要感谢我的导师丁向华老师在我的研究生生涯中给予的指导和帮助，特别是对于我的研究方向给予了很多宝贵意见，让我获益良多。而且，丁老师对于学术研究的认真严谨，也是我学习的榜样。

其次，我要感谢顾宁老师在我学习和生活中给予的关心和帮助。由于顾老师的帮助，我才能拥有丰富的实践机会，接触很多新的技术和学术研究，正是这些收获，我才确定了自己的研究方向。而且，在我的研究方向和研究内容上，顾老师也提出了很多的宝贵意见和指导，使我少走了很多弯路。

然后，我要感谢卢暾卢老师对我的科研工作和学习生活的关心和帮助。在我学习和研究遇到困难时，卢老师给予我很多指导意见，帮助我顺利解决问题。特别是论文撰写过程中，与卢老师的多次交流，让我获益良多。

最后，我要感谢三年来所有给予我帮助的老师和同学。感谢张亮老师在开题、中期给予的宝贵意见，感谢师兄师姐在学术研究上给予的帮助，感谢实验室所有同学在学习和生活中给予的帮助。

复旦大学
学位论文独创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。论文中除特别标注的内容外，不包含任何其他个人或机构已经发表或撰写过的研究成果。对本研究做出重要贡献的个人和集体，均已在论文中作了明确的声明并表示了谢意。本声明的法律结果由本人承担。

作者签名：_____ 日期：_____

复旦大学
学位论文使用授权声明

本人完全了解复旦大学有关收藏和利用博士、硕士学位论文的规定，即：学校有权收藏、使用并向国家有关部门或机构送交论文的印刷本和电子版本；允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文。涉密学位论文在解密后遵守此规定。

作者签名：_____ 导师签名：_____ 日期：_____