泰山科技学院毕业设计(论文)任务书

设计(论文)题目_____基于 Python3 的内网设备故障监测系统设计与实现_____

学生姓名____二级学院___专业 _班级___

指导教师_____职称__讲师_____ 联系电话___

教师单位_____下任务日期年__月___日

主要研究内容、方法和要求	本文介绍了基于 Python3 的图形化内网设备故障监测系统的三个方面的研究。首先，易用性研究包括图形化-功能块拖动模块、图形化-界面模块和新手引导模块的研究，通过将 Scratch 3-web 平台嵌套在 pyqt5 框架生成的子界面中以及调用相关函数对功能进行说明来提高用户的操作体验。其次，用户行为分析研究可以帮助开发者更好地了解用户的需求和行为习惯。最后，技术研究包括故障监测研究、网络模块研究和监测结果展示模块研究，利用高并发模块进行设备扫描、构建 API 并使用 Pandas 等数据分析框架对监测结果进行展示。本文旨在优化系统的易用性，减少用户学习成本，并提高用户对系统的满意度。
进度计划	第 1 到 3 周：收集相关文献； 第 3 到 4 周：完成设计方案初步设计； 第 5 到 9 周：完成毕业设计详细设计； 第 10 到 11 周：完善毕业设计并撰写毕业论文； 第 12 周：准备毕业设计的答辩
主要参考文献	
指导教师签字：__ 月 日 教研室主任签字：__ 年 日	

摘　要

网络设备互联在现代化的工作和生活中越来越重要，网络设备故障检测软件的易用性随着网络的不断发展越来越受到人们的关注。各家网络公司只愿开发更益于本公司的专业网络监测设备。

本文主要介绍了一种基于图形化编程的内网故障检测系统。该系统采用 Python3 平台，并将图形化编程与传统内网故障检测系统相结合。用户可以利用基于 PyQt5 框架编写的界面与 Scratch 3-web 平台相结合，生成功能链，使用功能链与后端 Python3 进行交互，实现内网设备信息查询和管理，为用户提供内网设备快速跟踪渠道。最后，利用 Python3 和 Pandas 编写的功能模块对内网故障检测系统产生的结果 json 文件进行读取分析，将分析结果以表格和网络拓扑图的形式展现给用户。该系统解决了传统内网故障检测系统中的专业软件易用性问题，提高了用户的使用体验。

基于 Python3 的图形化内网设备故障监测系统能够对内网设备从设备信息、故障检测、实时监控、易于操作等方面为用户提供便利，确保用户在内网设备出现故障时使用该系统做到更快的诊断和确认故障设备。它克服了传统内网扫描系统存在的专业化程度高、付费高等问题，形成了一种更加易用的内网设备扫描系统，降低专业扫描设备的使用的门槛。

【关键词】内网故障监测　图形化编程形式　专业软件易用化

ABSTRACT

The interconnection of network devices is becoming increasingly important in modern work and life, and the ease of use of network device fault detection software is receiving increasing attention with the continuous development of the network. Each network company is only willing to develop professional network monitoring equipment that is more beneficial to our company.

This article mainly introduces an internal network fault detection system based on graphical programming. The system adopts the Python 3 platform and combines graphical programming with traditional internal network fault detection systems. Users can combine an interface based on the PyQt5 framework with the Scratch3-web platform to generate a function chain, which can be used to interact with backend Python3 to achieve internal device information query and management, providing users with a fast tracking channel for internal devices. Finally, use the functional modules written in Python3 and Pandas to read and analyze the JSON file generated by the internal network fault detection system, and present the analysis results to users in the form of tables and network topology diagrams. This system solves the usability issue of professional software in traditional intranet fault detection systems and improves the user experience.

The graphical internal network device fault monitoring system based on Python 3 can provide convenience for users in terms of device information, fault detection, real-time monitoring, and ease of operation for internal network devices, ensuring that users can use the system to diagnose and confirm faulty devices faster when there is a fault in the internal network device. It overcomes the problems of high specialization and high payment in traditional intranet scanning systems, forming a more user-friendly intranet device scanning system and reducing the threshold for the use of professional scanning equipment.

【Keywords】IntranetScanningSystem graphicalProgramming professional software usability

目　录

前　言

现在，自己动手 DIY 家用网络设备并在生活中使用 DIY 家用网络设备已经成为家用网络设备圈的“新时尚”。然而，DIY 家用网络设备就意味着学习更多更难的专业知识。其中，传统内网设备故障监测系统因涉及大量的专业网络知识和较复杂的操作为刚接触 DIY 家用网络设备新用户平稳过渡“新手期”造成了相当大的困难。这一方面是传统的内网设备故障监测系统面向的更多是专业网络运维人员，另一方面是由于不同品牌的内网设备故障监测系统生产厂商只愿面向本公司产品开发相应的付费软件设备。为刚接触 DIY 家用网络设备的新用户开发一款低成本、易学习的内网设备故障监测系统在 DIY 网络设备行业中已经成为一个热点问题。

图形化编程思想的诞生为“零基础”用户解决学习传统编程语言困难的问题提供了全新的解决思路,为编程行业提供了新的动力。相比传统编程，图形化编程产品凭借其可视化操作、无需加深计算机知识、快速开发、可视化调试的特点，一经推出在编程教育行业实现了快速成长，是所有“零基础”用户入门编程的最佳平台。除此之外，鉴于 Python3 的易读性和应用范围广的特点，我们将基于 Python3 平台研究设计并构建内网设备故障监测系统。在此平台上结合图形化编程思想的优势，创建面向 DIY 网络设备新用户更容易入手的图形化内网设备故障监测系统功能界面。不仅可以解决 DIY 网络设备新用户在学习传统的内网设备故障监测系统时平稳过渡“新手期”困难的问题,还可以为 DIY 网络设备新用户拓展学习传统编程语言模块。本文系统全程开源免费，开源社区志愿维护，减少了 DIY 网络设备新用户在学习知识之外的不必要开支。

本文利用图形化编程思想通过图形化编程产品 Scratch 3 与传统编程语言 python3 相结合，实现在传统内网设备故障监测系统上的图形化功能应用，解决了刚接触 DIY 家用网络设备新用户对传统内网设备故障监测系统低成本、易学习的要求。

第一章　绪　论

第一节　研究背景

近年来，伴随着网络设备的不断发展和家庭可支配收入的提高，家用网络设备产业在我国得到了飞速发展，并逐步形成世界第一大家用网络设备市场。得益于家用网络设备产业的快速发展和电子商务平台在中国的大量普及，家用网络设备产业链上的生产厂商不仅加快了产品的推革出新，还建成了网络商务平台对家用网络设备芯片、网卡等零件进行单独售卖新渠道。除此之外，关注和学习家用网络设备及其相关专业知识的人越来越多。现在，自己动手 DIY 家用网络设备并在生活中使用 DIY 家用网络设备已经成为家用网络设备圈的“新时尚”。然而，DIY 家用网络设备就意味着学习更多更难的专业知识。其中，传统内网设备故障监测系统因涉及大量的专业网络知识和较复杂的操作为刚接触 DIY 家用网络设备新用户平稳过渡“新手期”造成了相当大的困难。这一方面是传统的内网设备故障监测系统面向的更多是专业网络运维人员，另一方面是由于不同品牌的内网设备故障监测系统生产厂商只愿面向本公司产品开发相应的付费软件设备。为刚接触 DIY 家用网络设备的新用户开发一款低成本、易学习的内网设备故障监测系统在 DIY 网络设备行业中已经成为一个热点问题。

图形化编程思想的诞生为“零基础”用户解决学习传统编程语言困难的问题提供了全新的解决思路,为编程行业提供了新的动力。相比传统编程，图形化编程产品凭借其可视化操作、无需加深计算机知识、快速开发、可视化调试的特点，一经推出在编程教育行业实现了快速成长，是所有“零基础”用户入门编程的最佳平台。除此之外，鉴于 Python3 的易读性和应用范围广的特点，我们将基于 Python3 平台研究设计并构建内网设备故障监测系统。在此平台上结合图形化编程思想的优势，创建面向 DIY 网络设备新用户更容易入手的图形化内网设备故障监测系统功能界面。不仅可以解决 DIY 网络设备新用户在学习传统的内网设备故障监测系统时平稳过渡“新手期”困难的问题,还可以为 DIY 网络设备新用户拓展学习传统编程语言模块。本文系统全程开源免费，开源社区志愿维护，减少了 DIY 网络设备新用户在学习知识之外的不必要开支。

本文利用图形化编程思想通过图形化编程产品 Scratch 3 与传统编程语言 python3 相结

合，实现在传统内网设备故障监测系统上的图形化功能应用，解决了刚接触 DIY 家用网络设备新用户对传统内网设备故障监测系统低成本、易学习的要求。

第二节　研究意义

一、理论实践意义

将传统的内网设备故障监测系统与图形化编程思想相结合，研究设计并降低传统内网设备故障监测系统的使用门槛。进一步丰富了计算机初学者教育领域软件产品开发的方向、并为其理论的探讨提供了实际参考价值。

二、现实意义

虽然少部分传统的内网设备故障监测系统开发厂商一直在提倡专业知识的简单化，但是相关应用还比较少。现已知面向“零基础”用户的教学产品，多半由社会组织为公益事业无偿开发或教育培训公司面向内部学员开发的简单、抽象、付费教学产品。鉴于以上原因，基于 Python3 的图形化内网设备故障监测系统为刚接触 DIY 家用网络设备新用户开辟了一条新的学习道路。它能够有效降低学习成本、提高学习效率。同时它也有力的增加了刚接触 DIY 家用网络设备新用户学习的信心。除此之外，全程开源免费以及开源社区维护的加入，降低了“零基础”用户除学习之外的成本投入。

目前图形化编程产品主要应用于教育行业，而在 DIY 网络设备领域大多是利用传统内网设备监测系统对网络设备互联进行监测，需要先把计算机网络和协议、网络设备、特殊名词、相关英语等学完之后进行操作，导致刚入门的 DIY 网络设备新手需要先投入巨大的学习成本。而本文系统的出现为解决上述问题提供了可能；同时，也能有效地让初学者完整的学习网络和编程知识，为开源软件开发提供新的模版、为刚接触 DIY 家用网络设备新用户开辟一条新的学习道路。

第二章　图形化内网设备故障检测系统的理论和技术

本文基于 Python3 的图形化内网设备故障监测系统主要包含三个核心：网络协议发送与接收、故障监测结果解析与展现、图形化功能编程。网络协议发送与接收和故障监测结果解析与展现主要依靠 python3 开发工具实现，并非本文核心技术，将在后面的章节进行讲解。图形化功能编程作为图形化编程思想在本文系统中的应用，作为本文核心技术，重点解决新用户对传统内网设备故障监测系统的低成本、低学习难度的要求，将在本章详解。

第一节　为什么使用图形化编程形式

图形化编程在编程方面是解决“零基础”用户更快学习编程思想和降低编程学习难度的首选方法，它是一种使用拖拽式编程来创建和编辑代码的编程技术。图形化编程运用了图形用户界面（GUI）和拖拽式编程。它利用了人们对图形的直观感受，即基于形状、大小、颜色、位置和方向等因素实现了更直观、易于理解的编程方式，其中，形状是人们最容易辨认和记忆的特征，位置和方向则提供了空间关系和结构的信息。因此图形化编程技术能够广泛应用在教育软件行业,并且有着十分广泛的应用前景。

①图形用户界面（GUI）

图形用户界面提供了直观、易于理解的用户界面，使用户能够轻松地理解和使用软件应用程序。使用户可以使用鼠标和键盘进行操作，而不必使用命令行或文本界面进行操作，为用户提供了极大的便利。

②拖拽式编程

相比于传统编程语言，图形化编程可以通过图形来表示编程语言的逻辑结构和程序流程，让刚接触 DIY 家用网络设备新用户更直观、易懂的理解编程知识。通过拖拽控件、设置属性、连接事件等方式来组合编写程序，而不需要编写代码。

第二节　图形化内网设备故障监测技术

随着计算机的发展，编程语言也逐渐进入了人们的视野之中，编程语言作为软件核心开始被人们慢慢重视起来，随着软件涉及的行业越来越多，面对非软件开发人员的编程语言也变多了起来，其中面向“零基础”用户的编程语言虽然不多，但极具代表性。图形化编程语言、函数式编程语言、标签语言这三类语言是现在的编程教育机构面向“零基础”用户的首选。它们的优缺点以及使用场景如表 2.1 所示

表 2.1 各语言优缺点对比表

语言	优点	缺点	使用场景
图形化编程语言	图形化编程语言易于学习和使用、图形化编程语言可以提高程序的可视化程度、图形化编程语言可以扩展应用程序的功能	图形化编程语言的灵活性受限制、图形化编程语言在扩展和修改程序时可能更加困难	游戏开发、学生教育
函数式编程语言	高度的抽象性、更加灵活、可读性	内存消耗、难以理解、学习曲线比较陡峭	并发处理和多线程编程、数据处理和大数据分析、人工智能和机器学习
标签语言	易于学习、易于维护	学习成本较高、复杂性更高	网页开发、自动化测试、数据库管理

通过对三类低难度编程语言的比较可以发现，想设计本文系统解决用户需求，需要采用图形化编程语言才能实现。

第三节　图形化内网设备故障监测技术

一、图形化内网设备故障监测基本模式

基于 Python3 的图形化内网设备故障监测系统是指刚接触 DIY 家用网络设备新用户通过图形化编程技术，利用图形化编程技术特有的更易于学习的语言优势与传统内网设备故障监测系统相结合，以完成刚接触 DIY 家用网络设备新用户对开发一款低成本、低学习难度的内网设备故障监测系统要求。

图形化编程在技术上运用了图形用户界面（GUI）技术和拖拽式编程技术手段,实现了更直观、易于理解的编程方式。它将代码编写过程中的诸多细节隐蔽掉，只要使用鼠标和

图形化界面，就可以实现代码的编写和操作，大大提高了编程效率。同时，为“零基础”用户进一步学习传统编程语言、了解网络知识、在学习计算机知识中平稳过度提供了一种新的学习方法。

二、图形化内网设备故障监测技术原理

“图形化编程”这种编程形式，最早是由麻省理工学院媒体实验室推出的，旨在解决传统文本化编程难度大、门槛高、学习成本高等问题，利用图形用户界面（GUI）技术和拖拽式编程技术手段使编程的更易学习、更易调试、更便于协作。而基于 Python3 的图形化内网设备故障监测系统正是利用图形化编程的优势，达到为刚接触 DIY 家用网络设备新用户对降低编程难度的要求。

第三章　图形化内网设备故障监测系统需求分析

第一节　业务逻辑分析

为解决 DIY 网络设备新用户所遇到的各种挑战,通过 DIY 网络设备新用户对传统内网设备故障监测系统的操作流程进行更深入的研究与分析,确定了传统内网设备故障监测系统的操作逻辑。

当前传统内网设备故障监测系统，一般由专业网络分析企业自行研发。并且，该类系统大部分采用会话式页面设置系统故障监测过程中可能使用的多种功能,通常是一个主控制面板，在面板功能表中列出系统故障监测过程中可能使用的功能，让用户对此功能表上的功能自行勾选。除此之外，传统内网设备故障监测系统的监测结果会返回一个表格来展示内网设备目标的通讯状态。监测的内网设备都有一个状态图标，用于指示设备是否正常工作。如果设备出现故障或异常，系统会自动向管理员发送警报，管理员可以通过界面查看警报和详细信息，以便尽快解决问题。传统内网设备故障监测系统还可以提供历史记录、分析和报告功能，以帮助管理员了解和改进网络设备。

传统内网设备故障监测系统的优点在于稳定可靠、专业性强、探测目标信息多备受专业网络运维人员的青睐，但是复杂控制的界面、高额使用费、专业的知识过多、异国语言为主等特点，为刚接触 DIY 家用网络设备新用户平稳过渡“新手期”造成了相当大的困难。

第二节　用户需求分析

为了能充分的做好系统需求，所以在进行系统开发之前，本人针对使用传统内网设备监测系统的众多 DIY 网络设备新用户做调研，并总结得出以下用户需求：有新手教程、操作简单易用、通过该软件可以学习更多的计算机知识、监测准确、可以查看历史检测数据、免费易下载、持续更新。

有新手教程：新手教程的作用是帮助“零基础”用户快速入门，了解基本的知识和操作，从而节省时间、提高效率，避免走弯路和犯一些初学者常犯的错误。所以，本文系统

利用 pyqt5 的描述功能构建完善的悬浮框教学系统。

操作简单易用：简单易用的操作界面可以让用户更容易理解和使用软件，从而提高用户的满意度和使用体验。简单易用的操作界面可以降低用户学习软件的难度和时间。所以，通过使用 Scratch 3-web 图形化编程平台与本文系统子界面相结合，使用户使用本文系统监测网络设备前，拖动提前预备的系统故障监测过程中可能使用的多种功能块，组合形成功能链完成对该系统的监测过程所需功能的设置，从而降低用户学习成本。

通过该软件可以学习更多的计算机知识：通过一个软件学习更多的相关专业知识，降低学习时间成本。所以，本文系统使用 Scratch 3-web 图形化编程平台与本文系统子界面相结合，可以降低传统内网设备监测系统的操作难度，而且，Scratch 3-web 图形化编程平台本身就是图形化编程产品，可以使用户提前学习计算机编程知识。

监测准确：准确的扫描结果可以让用户更好地了解网络设备情况，也为本文系统奠定良好的用户基础。所以，本文系统需要对内网设备多次扫描持续监测，提前多做系统测试。

可以查看历史检测数据并单独保存：监测结果保存本地，可以让用户更好的分析。所以，本文系统不仅将每次检测的结果保存本地，还保存为 json 文件为用户后期进行数据分析、导入数据库提供基础。

免费易下载、持续更新：只有不断更新不断进步才能让本文系统实现为用户彻底解决传统内网设备监测系统易用性的问题，所以，全程开源免费以及开源社区维护的加入实现了这一目的。

第三节　功能需求分析

通过对业务逻辑研究、用户需求进行深层次的研究和探讨,明确用户对基于 Python3 的图形化内网设备故障监测系统的功能要求,其功能要求大致分为三大板块。

第一板块：通过 Scratch 3-web 平台与本文系统子界面相结合，使用户使用本文系统监测前，拖动提前预备的多种功能块，组合形成功能链完成对该系统的监测过程所需功能的设置。从而彻底解决 DIY 家用网络设备新用户学习传统内网设备故障监测系统时控制界面操作困难的问题。

第二板块：利用 python3 构建传统内网设备故障监测系统，为第一板块提供基础的平台支撑。除此之外，本文系统全程开源免费，开源社区志愿维护，减少了 DIY 网络设备新用户在学习知识之外的不必要开支。

第三板块：本文系统利用 pyqt5 的描述功能构建完善的悬浮框教学系统，当用户鼠标指针悬停在本文系统某一功能图标上，在鼠标指针周围实现功能文本解释悬浮框。从而彻底解决 DIY 家用网络设备新用户学习传统内网设备故障监测系统时控制界面学习困难的问题。

第四章　图形化内网设备故障监测系统设计

第一节　系统设计的原则与规划

一、原则

基于对 DIY 网络设备新用户需求的理解，我们在设计本文系统时要基于以下原则：系统性、完整性、兼容性、稳定性、适用性。

系统性：虽然系统与系统之间具有不同的结构与不同的系统模块，但每个系统的所有的模块都是一个系统中必不可少的存在，他们共同构建了一个系统。为了使这些不同的模块能够更好的组合在一起，一定要做到让系统的代码能够统一起来，这样模块与模块的结合才不至于混乱。还要做到设计思路一成不变，这样更有利于系统出现故障的时候能够快速的解决问题。

完整性：作为一个完整的系统，一定要保证系统的各项功能完全，要能够具有对数据进行采集、分析、处理等必须具备的功能，还需要满足系统创立前用户期望的需求。

兼容性：是指系统能够对操作系统具有较强的适应能力，能够满足在不同的操作系统下还可以流畅完整的运行下来，还应该让各区域的代码能够模块化，便于系统的维护。减少模块与模块之间的联系，在一个模块出现问题停止工作的时候保证其他模块可以继续稳定运行下去。

稳定性：系统应该具有良好的稳定性以及安全性，系统需要提高自己的防护能力以确保减小经济损失，一个良好的系统还应时刻对系统内的数据进行备份等功能，以应对突发状况。

适用性：系统的开发应该尽可能对开发的成本进行降低，在满足安全性的前提下尽可能低成本的满足客户的要求，降低成本也可以提高系统的利润，还应做到尽量让系统简洁，尽量减少冗杂代码的使用，更有利于工程师的理解和对系统的维护。

二、规划

本系统采用了螺旋开发模型，使用 Enterprise Architect 设计软件采用 uml2 对本文系统进行设计。鉴于本系统特点我们将采用螺旋开发模型，螺旋开发模型共分四阶段：阶段 1

（难点）：实现 python3 网络协议数据包的发送与接收、阶段 2：加入高并发模块、阶段 3：加入 pyqt5 实现图形化展示、阶段 4：加入 Scratch 3 实现图形化功能拖动。如下图 4.1 螺旋开发模型示例所示。

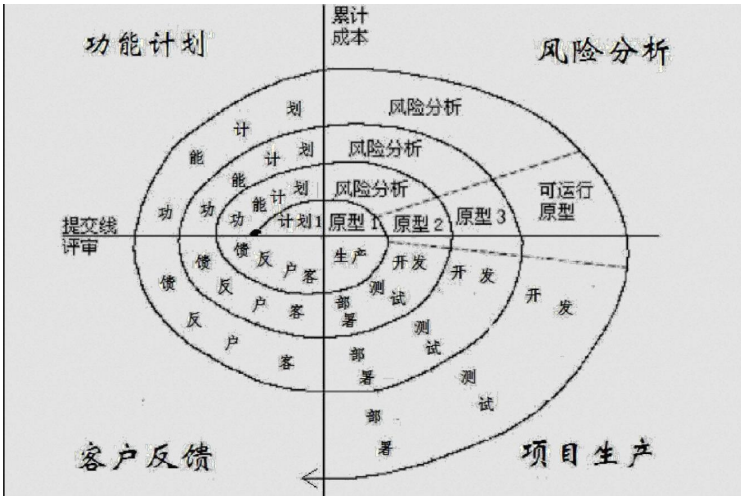

图 4.1 螺旋开发模型示例

概括来说系统开发第一步：完成系统基础搭建，基于 python3 平台完成高并发模块、网络协议模块等模块的构建和实现。系统开发第二步：在 python3 基础搭建的平台上完成基于 pyqt5 的 GUI 界面搭建。系统开发第三步：基于 pyqt5 搭建的 GUI 界面平台上完成对基于 scratch3 的图形化编程开发的界面进行嵌入。如下图 4.2 开发概况所示。

图 4.2 开发概况

第二节　系统功能模块

通过对 DIY 网络设备新用户的需求进行分析总结，发现该类用户对软件的要求宽泛、设计技术较多，所以，本文系统设计从前端模块、后端模块和新手入门引导模块这三模块

中对系统功能做出了定义。（为保证文章结构合理性，Scratch 3-web 平台与本文系统子界面相结合实现本文系统图形化编程作为核心技术不再单独列出，具体位置前端模块第二部分。）

前端模块主要为主界面和子界面，子界面中包括图形化编程功能模块。

后端模块主要为功能链分析过程、监测结果保存模块和监测结果返回接收与显示模块。

新手入门引导模块主要是对新手进行引导，对涉及软件的业务和功能进行介绍。具体的功能模块如图 4.3 所示。

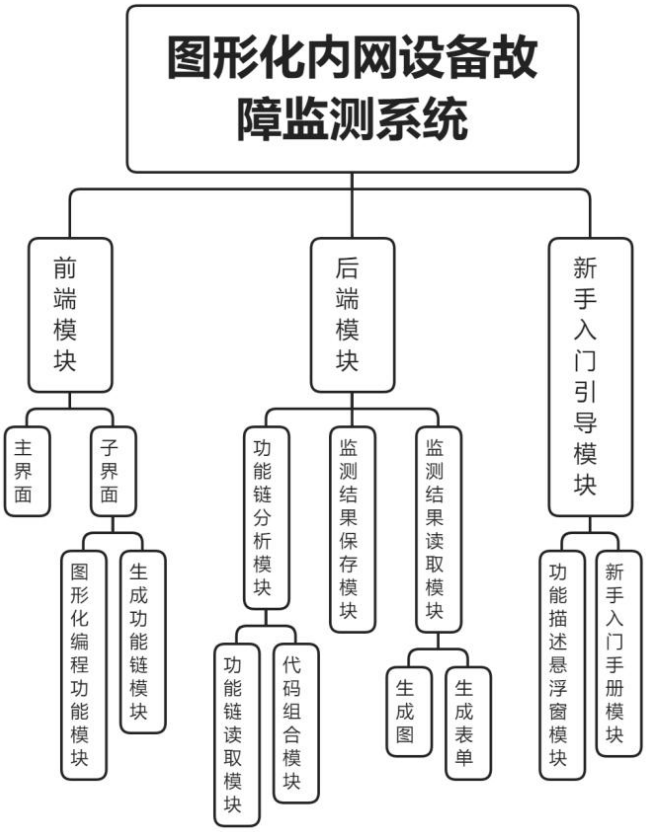

图 4.3 图形化内网设备故障监测系统功能模块

一、前端模块

①主界面

主界面及其主界面相关功能主要通过基于 Python3 平台的 pyqt5 框架构建。

主界面主要分为三部分，子界面展示区、子界面属性展示区和全部子界面信息统计及状态展示区。

子界面展示区：展示新建的监测扫描，用户可以在该展示区拖动和排列子界面。

子界面属性展示区：当用户点击子界面的时候，会展示该子界面信息，包括：创建时

间、扫描状态、扫描结果（简介）信息等。

全部子界面信息统计及状态展示区（树状图）：展示子界面名称和扫描状态

内网设备故障监测系统主界面功能展示如图 4.4 所示。实际如图 4.5 所示。

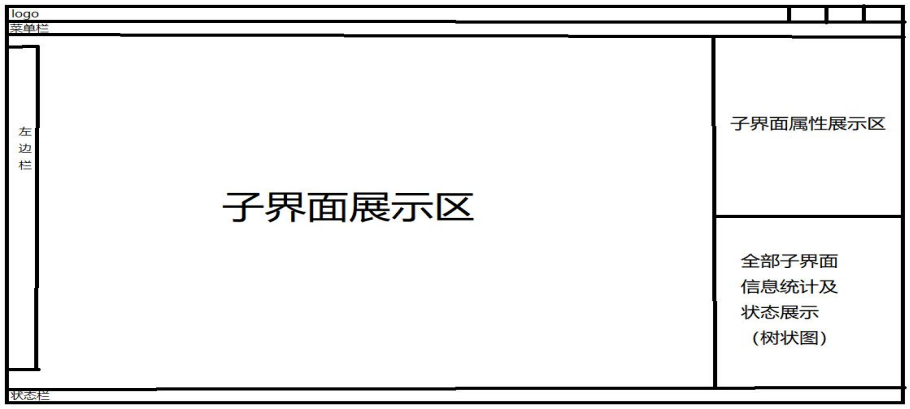

图 4.4 图形化内网设备故障监测系统主界面功能展示

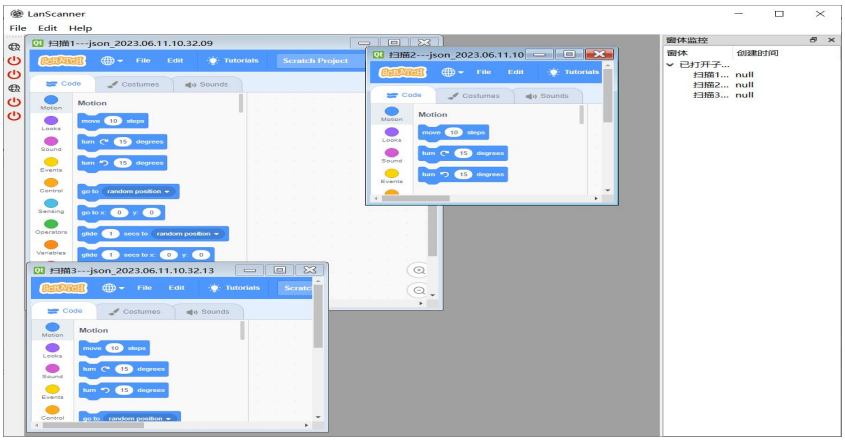

图 4.5 图形化内网设备故障监测系统主界面实际展示

②子界面-图形化编程功能拖动模块

本文图形化编程功能模块核心包括：功能链、合约表。

功能链：Scratch 3-web 嵌套在子界面中的每块图形化编程功能拖动模块相对应的 web 后端代码都是一组字符串，其中，“开始”模块实际上是在 web 后端新建一个字典，其他模块拖到“开始”模块下，就会将对应功能模块 web 后端的字符串加入该字典，该字典就是功能链。

合约表：合约表是作者利用 Scratch 3-web 中自定义图形化编程模块功能，面向本文系统单独开发的图形化编程模块与后端功能链分析模块之间的合约，例如：关于协议方面的

合约表，部分展示如表 4.1 所示。

表 4.1 本系统协议合约表部分展示

标号	功能
0	不使用协议模块
5	ICMP 协议模块
3	TCP 协议模块
4	ARP 协议模块

子界面中间内容区的图形化编程功能拖动实现，主要通过 python3-web 相关技术将 Scratch 3-web 平台嵌套在基于 Python3 平台的 pyqt5 框架生成的子界面中。图形化内网设备故障监测系统子界面如图 4.6 所示。

大类功能选择区：可以对协议、高并发模块、加密等大类功能进行选择，大类功能下有更加具体的功能模块。例如：点击“协议”按钮可以弹出 ICMP 协议功能拖动模块、TCP 协议功能拖动模块等。

子类功能选择区：对大类功能选择区选择的功能模块进行展示。例如：点击“协议”按钮可以弹出 ICMP 协议功能拖动模块、TCP 协议功能拖动模块等。

图形化编程功能约束模块（在本文系统后端模块的功能链读取模块中）：功能链读取模块会对功能组合进行检测，如果出现非法组合会弹出警告框。例如：数据流先通过 TCP 模块后通过多线程模块，顺序错误会造成多线程模块无法使用。

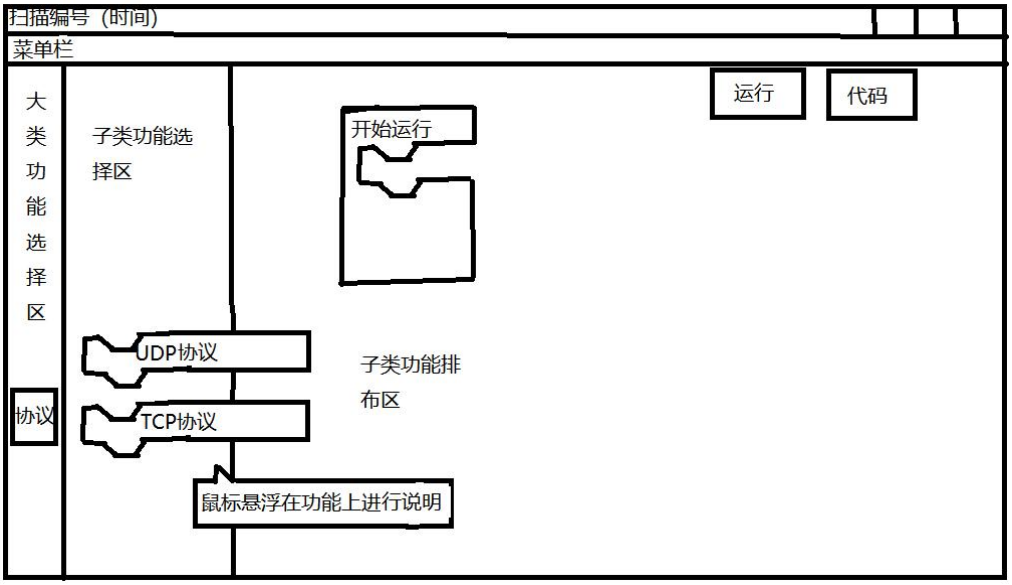

图 4.6 图形化内网设备故障监测系统系统子界面

拓展一：为什么使用 Scratch 3-web 平台来实现图形化编程在本文系统中的应用？

随着图形化编程的发展，类图形化编程语言也得到了进步，为加快图形化编程语言在新用户中的应用和普及，众多开发图形化编程的方法也得到了开发者实践。根据本文系统环境我们选择以下方法作为备用开发图形化编程方法：pyqt7 开发方法、web 开发方法、Scratch 3-web 平台自定义模块开发方法。

pyqt7 开发方法：由于 pyqt7 开发图形化编程语言的相关模块最近发布、与本文系统兼容性差尚在测试阶段，该方法不予通过。

web 开发方法：该方法有大量的开发者实践项目作为参考，由于 web 开发方法对开发者要求较高、涉及技术较多，该方法不予通过。

Scratch 3-web 平台自定义模块开发方法：Scratch 3-web 平台最近更新了用户自定义拖动模块，用户自定义拖动模块包含：拖动模块前端内容自定义和拖动模块后端代码自定义，符合本文系统要求及开发者难度要求，该方法给予通过。

拓展二：为什么不使用 Scratch 3-web 平台中的模块直接组合完成本文系统内网监测模块所需代码？

Scratch 3-web 平台中更详细的模块会提高 DIY 网络设备新用户的学习难度，其次，Scratch 3-web 平台只是实现本文系统图形化编程形式的众多选择之一，随着图形化编程语言的发展，实现本文系统图形化编程形式会有更多的新方法出现并替代 Scratch 3-web 平台方法。

③子界面-生成功能链模块

功能链原理：Scratch 3-web 嵌套在子界面中的每块图形化编程功能拖动模块相对应的 web 后端代码都是一组字符串，对这些功能模块设置就相当于修改相对应的字符串内容。其中，“开始”模块实际上是在 web 后端新建一个字典，其他模块拖到“开始”模块下，就会将对应功能模块 web 后端的字符串加入该字典。

除此之外，当用户点击“开始扫描”按钮，功能链就会将字典解析成 json 发送到本文系统后端的功能链读取模块进行分析（参考 web 前后端分离）。

例如：ICMP 协议对内网设备进行故障监测扫描的功能块组合。用户在前端将所需要的模块拖动到“开始”模块下，与此同时后端将该模块后端的字符串加入开始模块的字典。

注意：由于功能开发不完善，现还需要将模块所需属性设置完成后再拖动。

图形化内网设备故障监测系统子界面功能模块组合演示如图 4.7 所示。

图 4.7 内网设备故障监测系统系统子界面功能模块组合演示

当用户点击开始扫描后 Scratch 3-web 会将后端的组合功能链通过 json 的形式发送到本文系统的功能链分析模块，对应上文 ICMP 协议对内网设备进行故障监测扫描的功能块组合 json 链如图 4.8 所示。

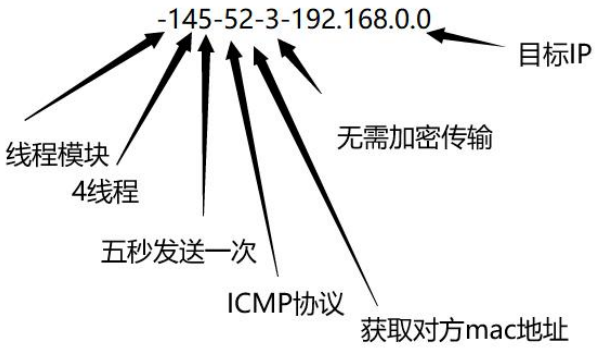

图 4.8 图形化内网设备故障监测系统功能链

子界面-生成功能链模块如下流程图所示，图形化内网设备故障监测系统功能链组合流程图如图 4.9 所示。

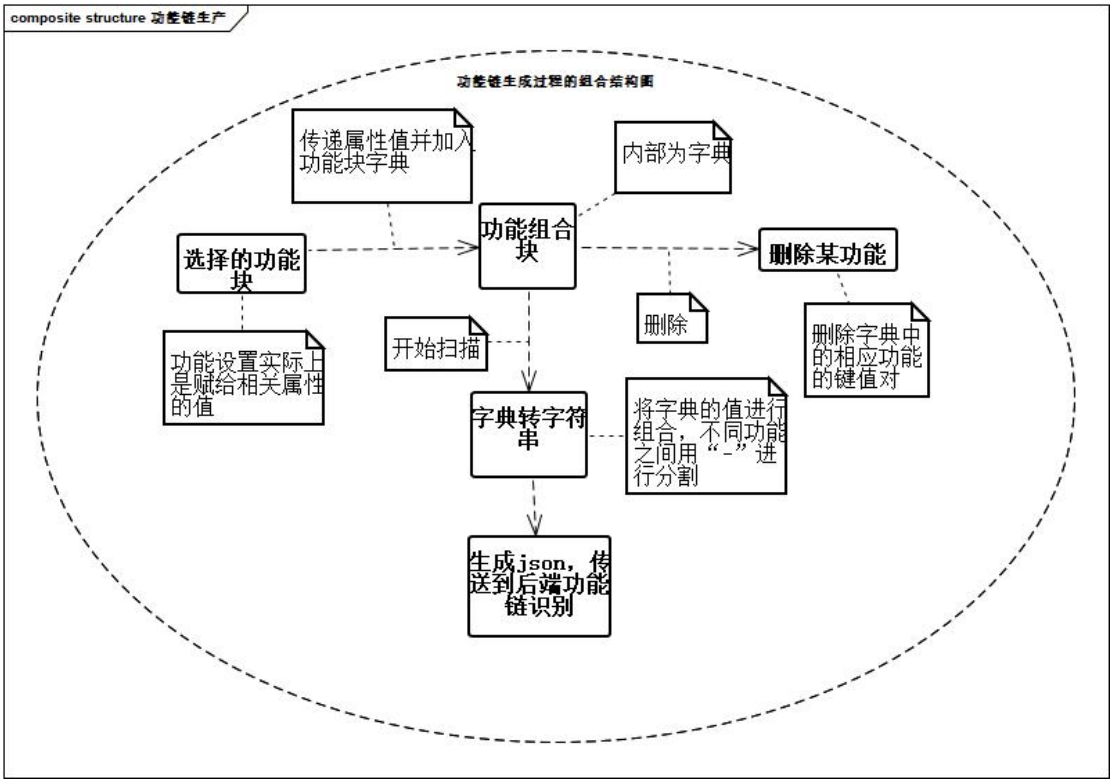

图 4.9 图形化内网设备故障监测系统功能链组合流程图

二、后端模块

Scratch 3-web 平台当用户开始监测扫描后会将功能链以 json 的方式传输给后端功能链读取模块。功能链读取模块实际上是对子界面 Scratch 3-web 平台功能链字典进行还原（类

似 web 前后端分离的后端），还原到原来的字典带入本文系统的监测扫描模块。

①功能链分析模块-功能链读取模块

此模块通过合约表分析并还原子界面 Scratch 3-web 平台产生的功能链。本文系统还原功能链如图 4.10 所示。

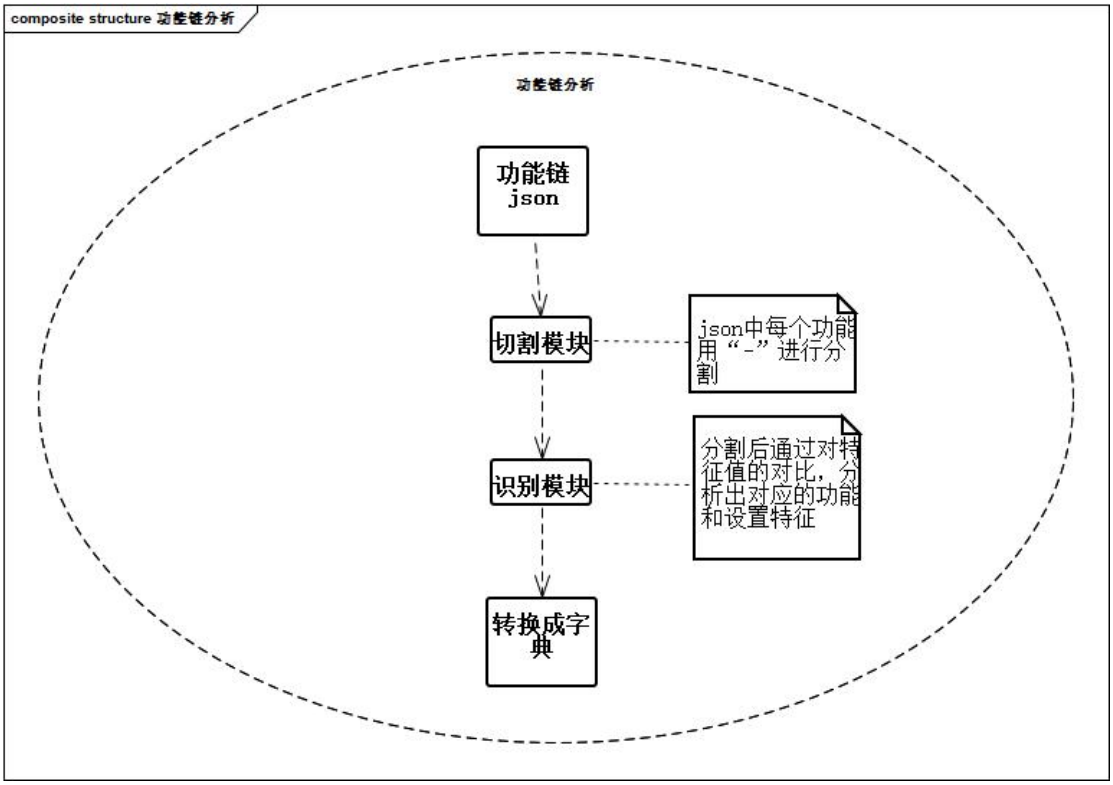

图 4.10 图形化内网设备故障监测系统还原功能链

②功能链分析模块-代码组合模块

功能链读取模块将功能链还原之后发送到代码组合模块。

代码组合模块的目的：实现功能链由字典转换为监测扫描中的属性设置。

代码组合模块的原理：将功能链的与相关属性赋值，监测扫描时带着相关属性，到关键节点功能的时候进行选择。图形化内网设备故障监测系统功能链如图 4.11 所示。

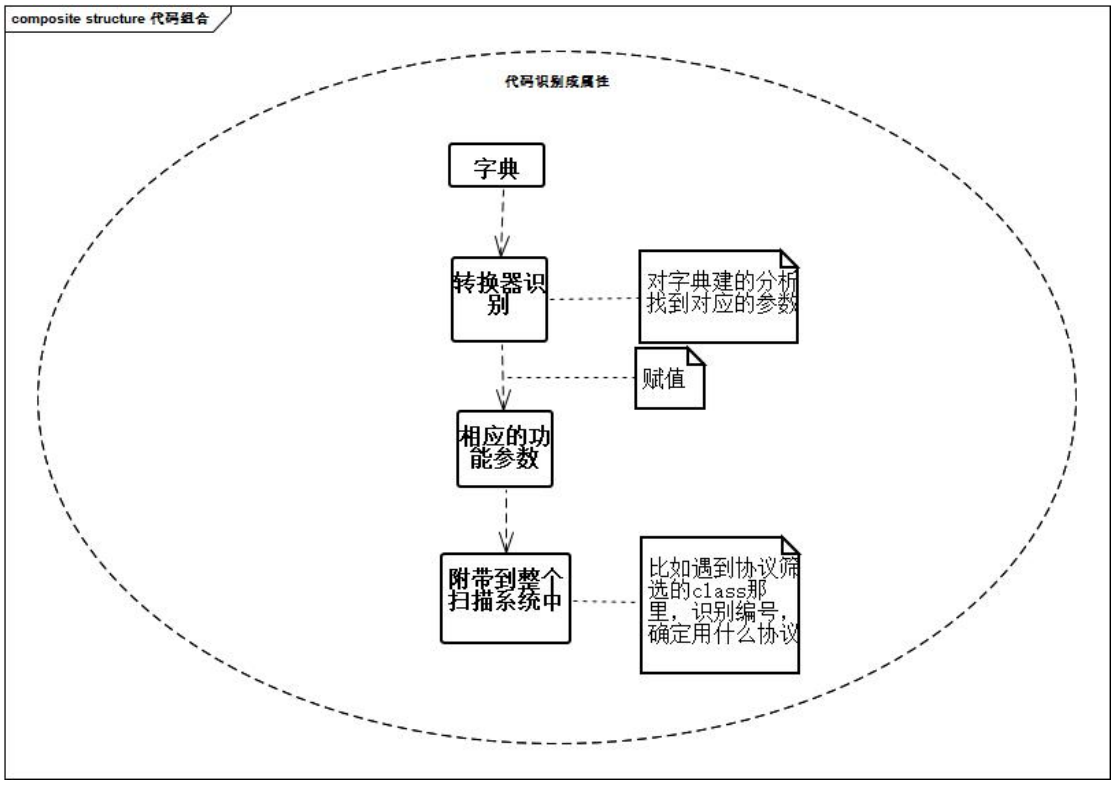

图 4.11 图形化内网设备故障监测系统功能链

三、新手入门引导模块

①功能描述悬浮窗模块

利用 pyqt5 框架的调用 QToolTip.showText()函数，对相关功能进行重点说明。当用户鼠标指针悬停在本文系统某一功能图标上，在鼠标指针周围实现该功能文本解释悬浮框。

②新手入门手册模块

此模块为 PDF 文档，对本文系统所有操作进行详细解读。

第三节　图形化内网设备故障监测技术

一、系统软件新功能更新

原理：服务器保存一个功能表，本地保存一个功能表。上传本地功能表与服务器功能表进行比对。如果有缺少则利用 TCP 协议进行传输下载。更新流程如图 4.12 内网设备故障监测系统更新功能流程图。

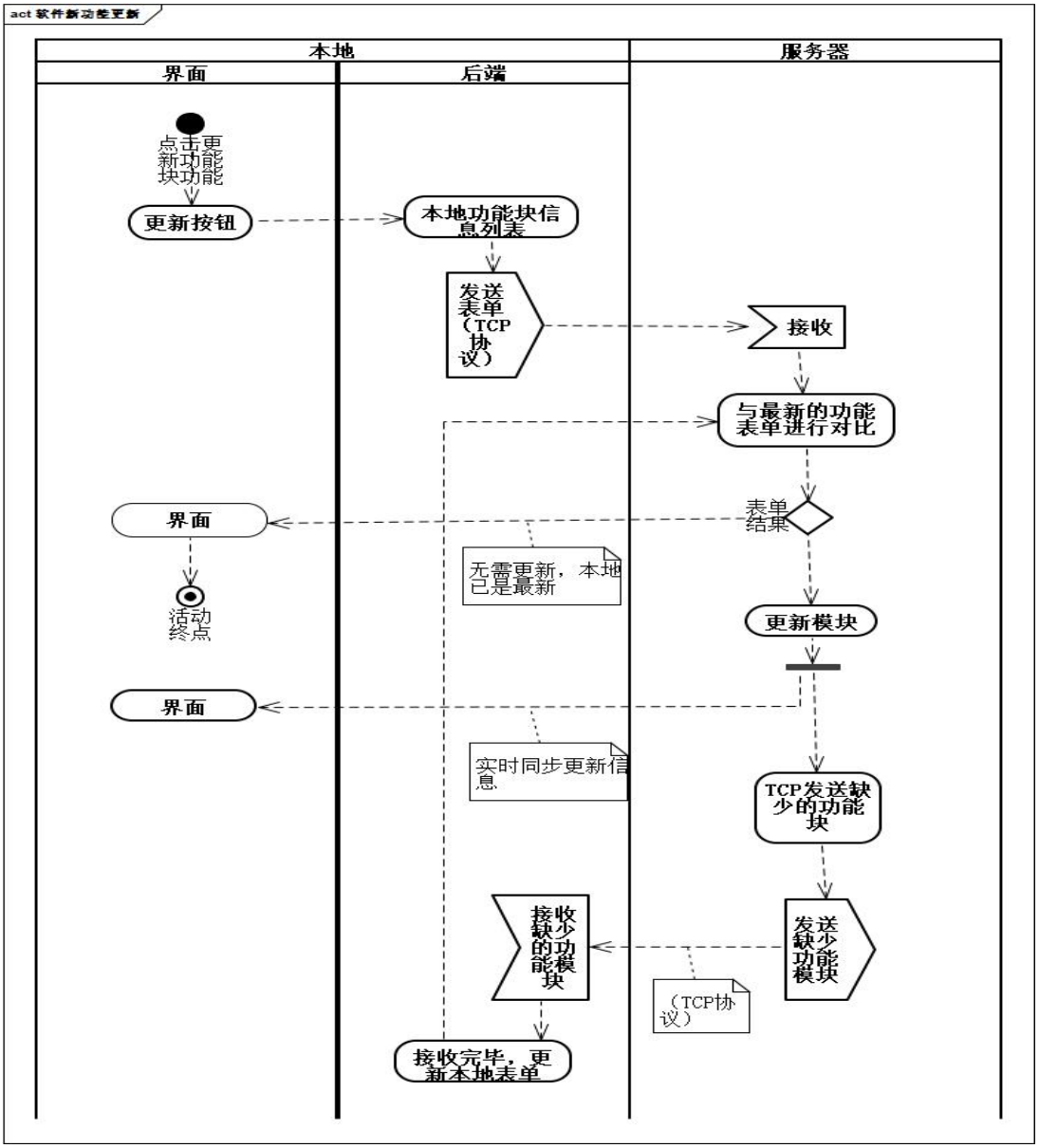

图 4.12 内网设备故障监测系统更新功能流程图

二、系统监测扫描过程

上一节系统功能模块的前后端模块已经将本文系统监测扫描的功能设置完美展现，之后是具体扫描过程，如下图图 4.13 和 4.14 所示。

界面流程图，主界面内网设备故障监测系统扫描流程-主界面图如图 4.13 所示和子界面内网设备故障监测系统扫描流程-子界面图如图 4.14 所示。

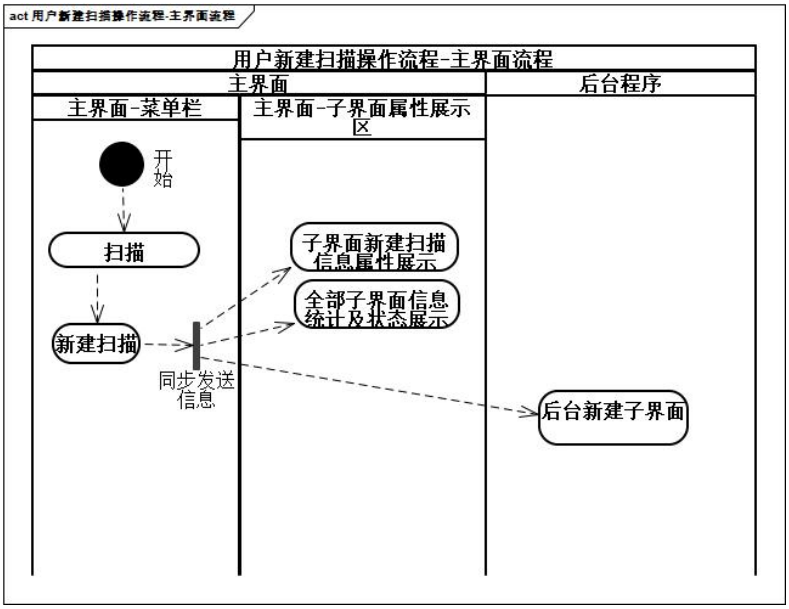

图 4.13 内网设备故障监测系统扫描流程-主界面图

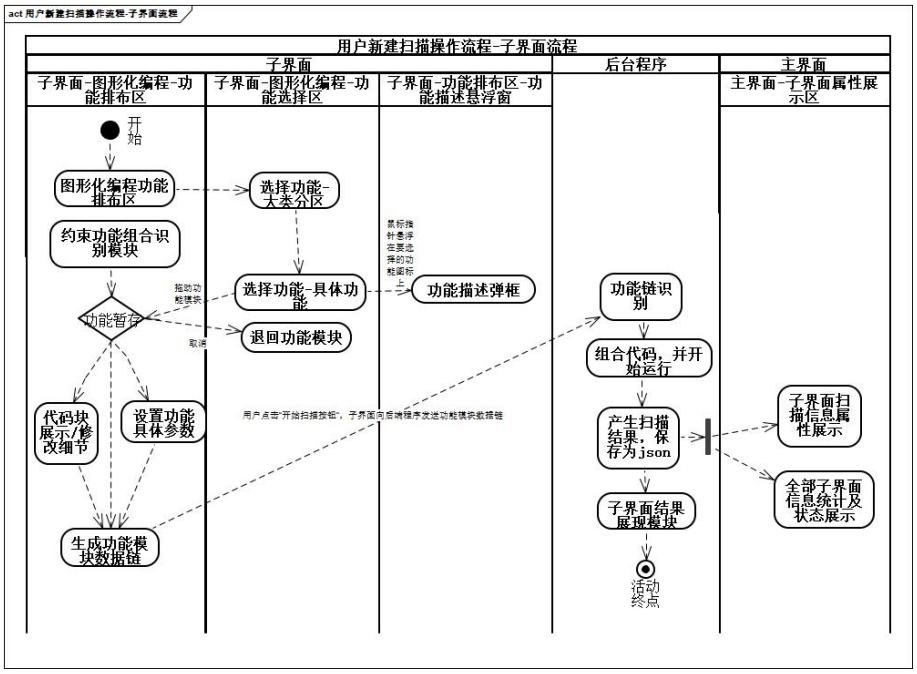

图 4.14 图形化内网设备故障监测系统扫描流程-子界面图

后端整体扫描，图形化内网设备故障监测系统扫描流程-后端图如图 4.15 所示

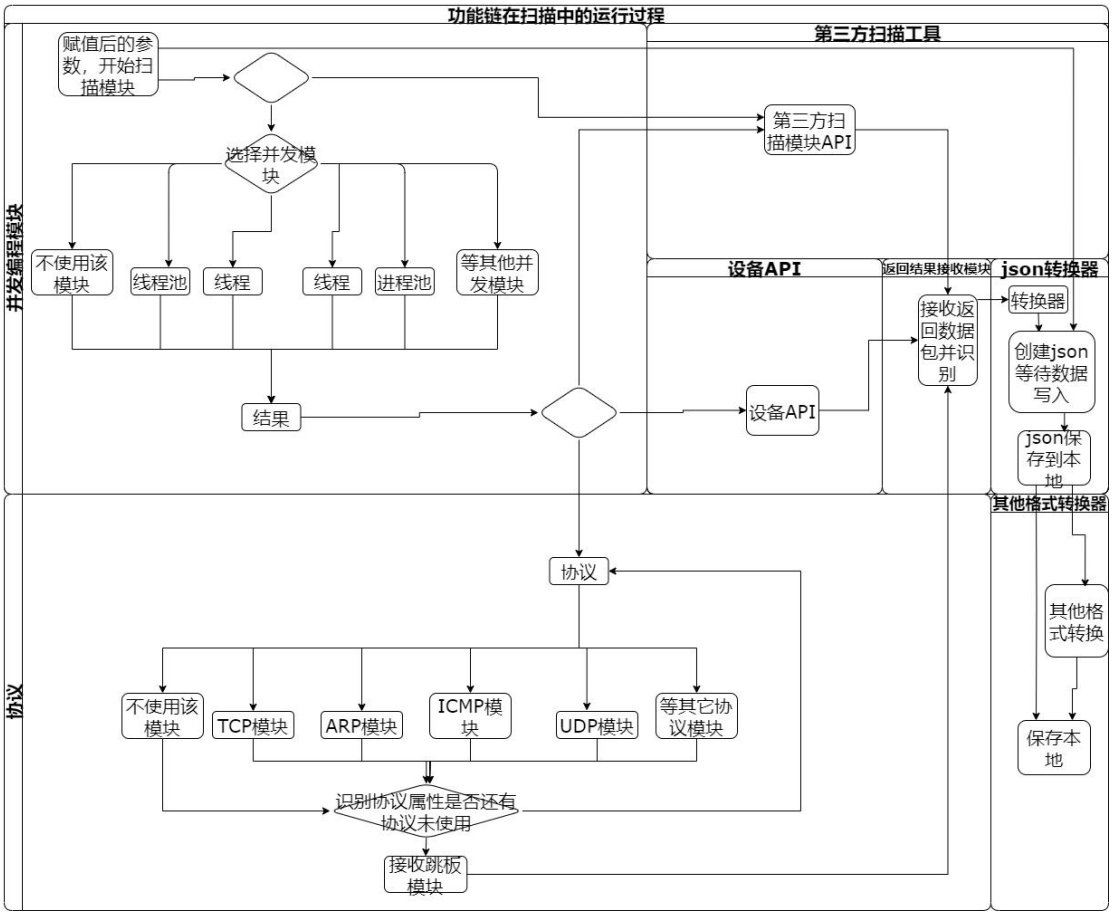

图 4.15 内网设备故障监测系统扫描流程-后端图

拓展：内部数据包发送以 ICMP 协议进行举例，ICMP 数据包结构图如图 4.16

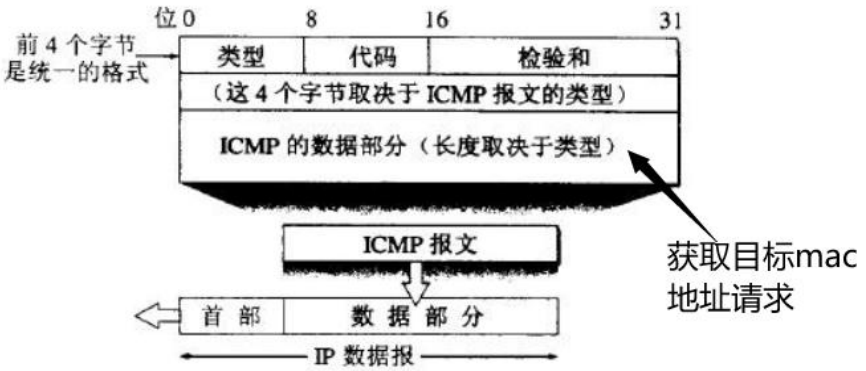

图 4.16 ICMP 数据包结构图

扫描发送后等待数据包返回，通过线程 ID 来确定是哪个线程的数据包。

对于返回包我们进行数据包分割，获取内容。根据返回包类型代码对内容进行识别 ICMP 返回包类型代码如图 4.17。

类型	代码	名称	查询	差错
0	0	回应应答(Echo Reply)	√	
3		目的地不可达		√
	0	网路不可达		√
	1	主机不可达		√
	2	协议不可达		√
	3	端口不可达		√
	4	需要分片和不需要分片标记置位		√
	5	源路由失败		√
	6	目的网络未知		√
	7	目的主机未知		√
	8	源主机被隔离		√
	9	目的网络的通告被禁止		√
	10	目的主机的通信被禁止		√
	11	对请求的服务类型 ToS，目的网路不可达		√
	12	对请求的服务类型 ToS，目的主机不可达		√
	13	由于过滤,通信被强制禁止		√

图 4.17 ICMP 返回包类型部分表

之后线程将返回包传到 pandas-API 模块，pandas 自动将返回的数据包进行整理，保存为 json，json 文件名为子页面标题（日期时间）。

三、系统扫描结果展示

原理：对返回包 json 的分析，实际就是分析 json，如图 4.18 所示。

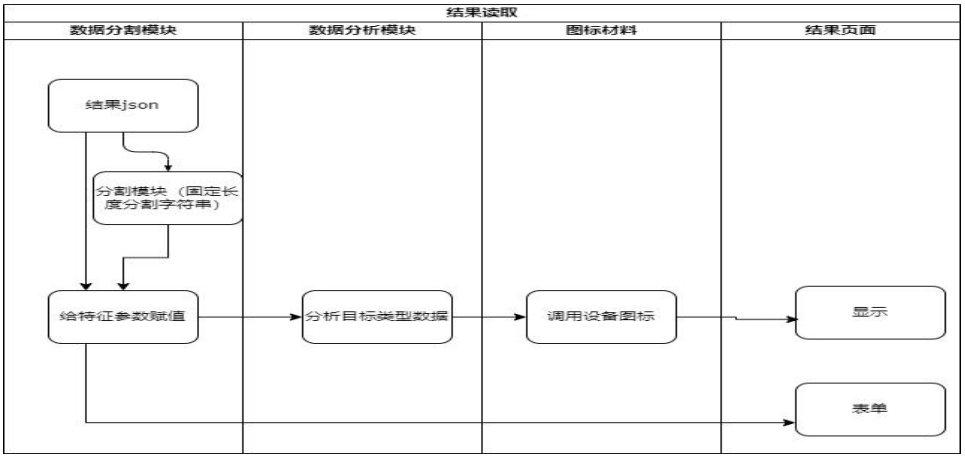

图 4.18 返回包 json 读取

四、系统监测结果保存设计

由于监测结果受用户监测扫描功能设置的影响，所以该表只做参考。

表 4.2 监测扫描返回数据存储表

字段名	类型	长度	是否为 NULL	键	注释
id	int		否	key	自增序列
源 IP	char	20	否		
目标 IP	char	20	否		
通讯状态	char	100	否		
标识符	int		否		
Thread_id	char	20	否		
开始时间	char	40	否		
mac 地址	char		是		

第四节　系统架构模块

一、功能架构

内网设备故障监测系统的功能架构主要展示内网设备故障监测系统不同层可使用哪些功能。

内网设备故障监测系统的功能架构，一般包括三大部分:界面层、功能层、网络层、数据层、数据采集层。如图内网设备故障监测系统功能架构图 4.19 所示。

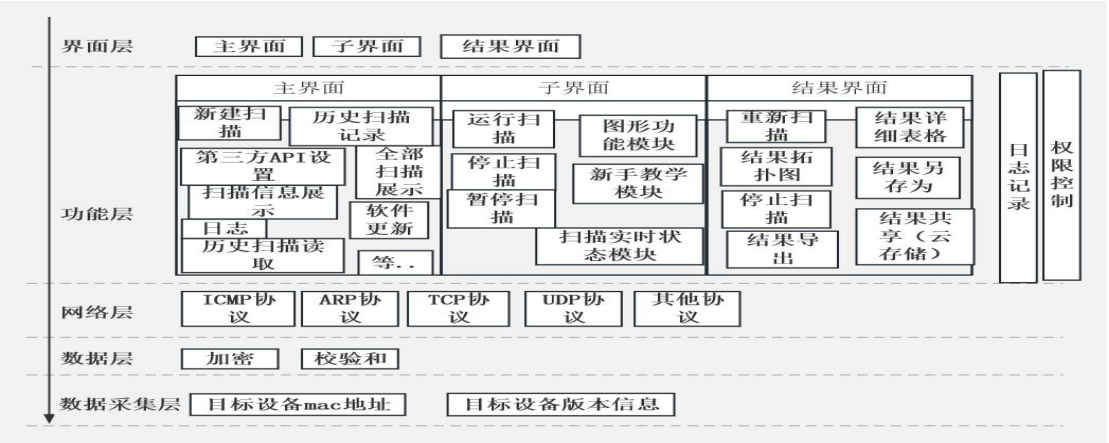

图 4.19 内网设备故障监测系统功能架构图

界面层：提供了基础的界面结构，为系统做出了大概的划分。功能层：对软件的功能

进行大致分类。网络层：网络层为上层提供通信协议。数据层：对输入输出的数据进行加密和解密。数据采集层：对输入输出的数据内容进行设置。

二、技术架构

内网设备故障监测系统技术框架展示了实现完整的内网设备故障监测系统功能结构所需的可使用在 python3 的技术支持。

内网设备故障监测系统的技术架构，一般包括三大部分:界面层、数据处理层、并发层、网络层。内网设备故障监测系统功能架构图如图 4.20 所示。

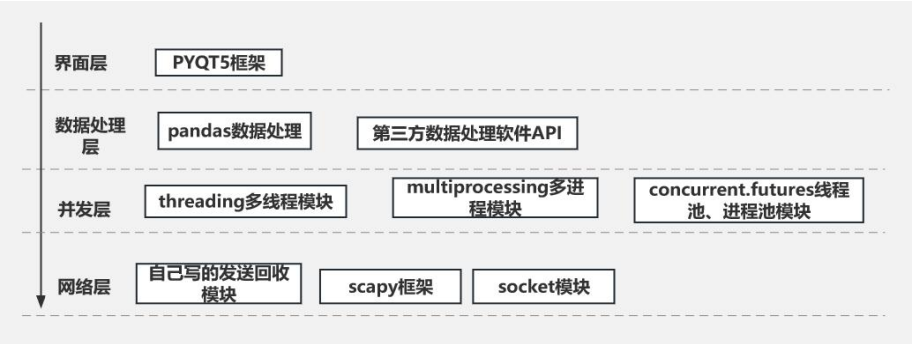

图 4.20 内网设备故障监测系统功能架构图

第五章　图形化内网设备故障监测系统系统实现

第一节　系统开发过程

基于 Python3 的图形化内网设备故障监测系统主要包括前端和后端部分，分别进行描述：

一、前端代码实现

主界面和子界面主要通过基于 Python3 平台的 pyqt5 框架构建，子界面中嵌套图形化编程部分由基于 Scratch 3-web 开源产品定向开发。主界面和子界面功能直接与 Python3 功能模块对接，子界面中嵌套图形化编程部分采用前后端分离的模式，需要通过功能链、合约表、后端功能链读取模块实现其功能。新手入门引导属于前端，由 pyqt5 悬浮框模块实现。图形化编程功能模块、生成功能链模块由基于 Scratch 3-web 开源产品定向开发。

二、后端代码实现

后端代码主要使用 Python3 语言实现，根据不同的模块要求采用不同的框架进行辅助。具体模块模块包括：

①扫描部分：

高并发模块：

multiprocessing 多进程模块、threading 多线程模块、concurrent.futures 线程池和进程池模块。

网络协议模块：

自己写的发送回收模块、scapy 框架、socket 模块。

第三方工具或设备扫描 API：

主要依赖对应设备或工具提供的基于 Python3 开发的相应模块。例如：nmap 则用 nmap-py、msf 则用 msgpack 和 MsfRpcClient

监测返回数据包识别模块：

struct 模块，由于已经将线程 ID、时间戳写入数据包内容部分，所以部分识别是由发送时写入数据包内容部分的特征码进行识别的。

②监测结果部分：

数据存储模块：

由于本系统给予用户更多的操作和自定义功能，我们将数据存储为 json。如果用户需要保存到数据库或 Excel 等其他类型文件，也可以通过相应的 pymysql 模块或 xlrd 模块转入用户期望的软件。

结果展示模块：

主要通过 Pandas 与 pyqt5 的结合对 json 数据进行分析。

③功能链分析部分：

功能链读取模块和代码组合模块：主要由 Python3 通过条件判断进行操作。

④系统其他功能部分：

系统更新和日志记录：主要由 Python3 与云服务器相结合完成更新。

第二节　开发环境介绍

本项目开发工具主要包括：pycharm、Visual Studio Code。

代码托管平台：git bash-2.40.1.window.1

本项目开发环境包括：

pycharm 中 pip 环境如表 5.1 所示。

表 5.1 pip 依赖环境版本

软件	版本
Pymysql	1.0.3
Python	3.10.5
pyqt5	5.15.9
pyqt5-qt5	5.15.2
pyqt5-sip	12.12.1
PyQtWebEngine	5.15.6
PyQtWebEngine-qt5	5.15.2
Pandas	2.0.1
Pip	21.3.1
pyqt5-plugins	5.15.9.2.3

qt5-application	5.15.2.2.3
sqt5-tools	5.15.2.1.3
qtwidgets	1.1
xlrd	2.0.1

Visual Studio Code 中 npm 环境涉及软件版本众多，我们对重要本分进行举例如表 5.1 所示。

表 5.2 npm 依赖环境版本

软件	版本
scratch-gui	1.8.78

第三节　系统实现

由于本系统设计功能和模块众多，我们只介绍重要核心模块。

一、高并发模块

以多线程高并发模块举例，因为不确定返回包返回的时长，所以多线程部分主线程完成后子线程依旧可以继续运行。所以代码中主线程没有监听和主动关闭部分。

①主线程部分

如图 5.1 主线程代码所示。

```
def find_ip_queue(fixed_IP_segment):
  global time#一些必要的设置
  workqueue=queue.Queue(256)#队列上限
  threads=[]#线程队列
  threadID=1
  for i in range(0,256):# 组合新的ip（目标ip）
    ip = ('%s.%s' % (fixed_IP_segment, i))
    workqueue.put(ip)
  exit_flag = 0#队列是否为空特征码，1为空，0为非空
  while not exit_flag == 1:
    if workqueue.empty():
      exit_flag=1
    else:
      ip = workqueue.get()
      thread = MultiThreadedModule.myThread(threadID, ip)#线程模块
      thread.start()
      threads.append(thread)  # 线程管道集合
      threadID += 1
```

图 5.1 主线程代码

②子线程部分

如图 5.2 子线程代码所示。

```
class myThread(threading.Thread):
  def __init__(self, threadID,ip):
    threading.Thread.__init__(self)
    self.threadID = threadID
    self.ip=ip
  def run(self):
    ping_ip(ip_str=self.ip,name=self.name)#此处选择ping命令，icmp协议
def ping_ip(ip_str,name):
  pinger= icmp_ping.Pinger(target_host=ip_str)#只是一个传值
  pinger.ping(name)#开始运行
```

图 5.2 子线程代码

二、数据包发送接收控制模块

以 ICMP 网络协议模块举例，因为本系统涉及到故障监测，TCP、ARP、UDP 等有相应故障监测功能但没有 ICMP 故障涉及的全面。

数据包发送接收控制模块对数据包发送和接收进行整体控制。如图 5.3 数据包发送模块代码所示：

```
def ping_once(self,name):  # 3运行，用来检查，向目标主机发送一次查验
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)
    except socket.error as e:
        if e.errno == 1:
            e.msg += "ICMP消息只能从管理员用户进程发送"
            raise socket.error(e.msg)
    except Exception as e:
        print("Exception: %s" % (e))
    ThreadID = os.getpid() & 0xFFFF#线程id作为返回包区别线程的特征

    #发送
    self.send_ping(ThreadID,sock)#发送到目标主机（发送和计算校验和）
    #接收
    #self, sock, ID, timeout, target_addr,
    delay = IPP.IcmpPingPong.receive_pong(self,sock=sock, ID=ThreadID, timeout=self.timeout, target_addr=self.target_host, name=name)
    sock.close()
    return delay
```

图 5.3 数据包发送模块代码

三、网络协议模块

以 ICMP 网络协议模块举例，因为本系统涉及到故障监测，TCP、ARP、UDP 等有相应故障监测功能但没有 ICMP 故障涉及的全面。

网络协议模块主要是对要发送的数据包根据协议要求进行封装。之后传递给 socket 进行发送。如图 5.4 网络协议模块代码所示

```
def send_ping(self,ThreadID,sock):  # 4运行，只是发送
    # header,data= self.checksum_calculation(ThreadID)
    # packet = header + data
    # #该他了
    # sock.sendto(packet, (self.target_host, 1))
    header,data=IPS.IcmpPingSend.checksum_calculation(self,ThreadID)#校验和计算
    packet = header + data
    sock.sendto(packet, (self.target_host, 1))
校验和计算
class IcmpPingSend():
    def checksum_calculation(self,ThreadID):
        my_checksum = 0
        header_false = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, my_checksum, ThreadID, 1)
        #类型，代码，校验和，标识符，序号。
        #为什么发送8.0，因为这是个请求包，差错控制。参考https://blog.csdn.net/qq_40276626/article/details/120371213
        bytes_In_double = struct.calcsize("d")  # 给定字节数，d=8
        data = (192 - bytes_In_double) * "Q"  # 填充184个q，这里就是把前面的时间占位减去。
        data = struct.pack("d", time.time()) + bytes(data.encode('utf-8'))
        #计算真的校验和
        my_checksum = IcmpPingSend.do_checksum(header_false + data)  # 校验和，把icmp头和内容拼接起来，注意header和data加起来正好256
        header = struct.pack(
            "bbHHh", ICMP_ECHO_REQUEST, 0, socket.htons(my_checksum), ThreadID, 1 )
        return header,data
    def do_checksum(source_string):  # 5运行,计算校验和
        sum = 0
        max_count = (len(source_string) / 2) * 2
        count = 0
        while count < max_count:
            sum+= source_string[count] + (source_string[count + 1] << 8)
            count = count + 2#两个为一组
        if max_count < len(source_string):  # 如果是奇数，就把最后一个数拼接上
            sum += ord(source_string[-1])
        sum = (sum >> 16) + (sum & 0xffff)#十六进制
        sum = sum + (sum >> 16)
        answer = ~sum & 0xffff
        answer = answer >> 8 | (answer << 8 & 0xff00)##移位指的是对应二进制移位，例如1<<2 1往左两位就是001就是100又8421得4
        return answer
```

图 5.4 网络协议模块代码

四、监测返回数据包识别模块

以 ICMP 网络协议模块举例，因为本系统涉及到故障监测，TCP、ARP、UDP 等有相应故障监测功能但没有 ICMP 故障涉及的全面。

监测返回数据包识别模块主要是对要接收的数据包根据协议要求进行解包，并传递给数据库。如图 5.5 检测返回包识别模块代码所示

```
class IcmpPingPong():
    def receive_pong(self,sock, ID, timeout, target_addr, name): # 6接收返回（监听）,可运行（pong的意思就是监听）
        time_remaining = timeout
        while True:
            start_time = time.time() # 返回当前时间的时间戳（1970纪元后经过的浮点秒数）
            readable = select.select([sock], [], [], time_remaining)
            # 参考https://blog.csdn.net/samsam2013/article/details/78554073
            time_spent = (time.time() - start_time) # select的时间
            if readable[0] == []: # Timeout
                IcmpResult=icmp_result.start(19, 0, target_addr)
            PythonJsonConversion.StagingList(self,target_addr=target_addr,IcmpResult=IcmpResult,name=name,symbol=0)
                return
            time_received = time.time()
            recv_packet, addr = sock.recvfrom(1024)
            icmp_header = recv_packet[20:28] # 直接字节流截取输出，这里有极大的问题可能是网络层的，也可能是节取得有问题
            type, code, checksum, packet_ID, sequence = struct.unpack(
                "bbHHh", icmp_header
            ) # 有问题
            if packet_ID == ID:#判断是不是对应程序线程接收
                bytes_In_double = struct.calcsize("d") # calsize 计算按照格式 fmt 打包的结果有多少个字节。8
                time_sent = struct.unpack("d", recv_packet[28:28 + bytes_In_double])[0]
                IcmpResult=icmp_result.start(type, code, target_addr) # 进入数据库
                PythonJsonConversion.StagingList(self,target_addr=target_addr, IcmpResult=IcmpResult, name=name, symbol=1)
                return time_received - time_sent # 判断发送了多长时间，成功的，
                # 现在的时间-发包时间
                # 数据包的超时时间判断，time_remaining是超时时间
            time_remaining = time_remaining - time_spent # 超时时间-等待时间
            # 可能发送数据包（里面是0,0），发出去后两条路①超时没法出去②发出去了但是目标不可达就是里面是别的数
            if time_remaining <= 0:
                # 两个超时，一个是根本没发出去。一个是发出去啥都对了超时了
                IcmpResult=icmp_result.start(19, 0, target_addr, name) # 进入数据库
                PythonJsonConversion.StagingList(target_addr, IcmpResult, name, symbol=0)
                return # 超时才会返回
```

图 5.5 检测返回包识别模块代码

五、子界面生成模块

子界面生成，主界面可以保留并展示子界面信息，同时子界面可以显示对应的图形化编程。子界面不限制数量。如下图 5.6 所示

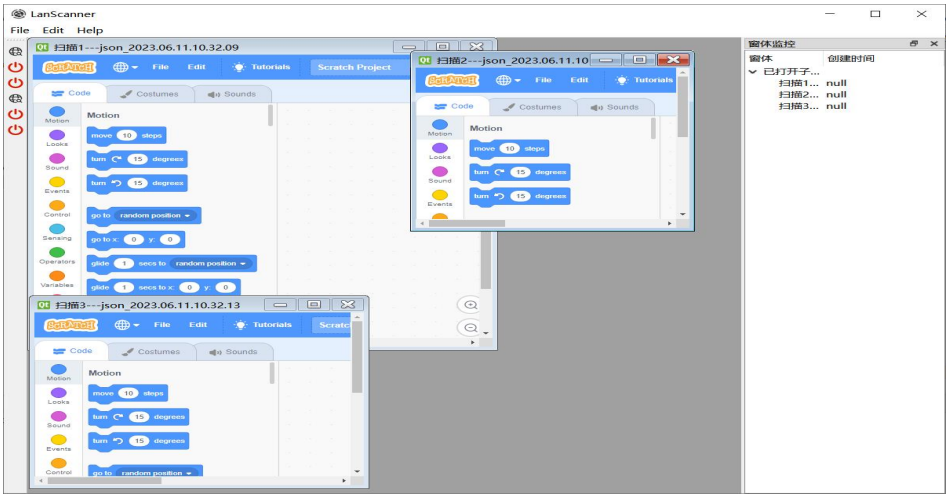

图 5.6 子界面视图

子界面生成模块核心代码如图 5.7 所示

```
class CreateChildWindow(QMainWindow):
    def CreateChildWindowMainWindow(self):
        self.setCentralWidget(self.mdi)
        self.sub = QMdiSubWindow()
        self.web_view = QWebEngineView()
        self.count = self.count + 1
        DEMPJ.PythonJson()
        JsonFileName=DEMPJ.PythonJson()#注意有没有()
        CreateChildWindowMainWindowName="扫描" + str(self.count)+"---"+JsonFileName.JsonFileName
        self.sub.setWindowTitle(CreateChildWindowMainWindowName)
        CTNSW.chuild.append(CreateChildWindowMainWindowName)
        CTNSW.CreateTreeNodeSuspensionWindowTree(self)
        self.sub.resize(600, 600)
        self.ChildWindowleftToolBar = QToolBar("ChildWindowRightToolBar")
        CCWLTB.CreateChildWindowLeftToolBarss(self)
        CSC.CSCsearch(self)
        CDJTPT.TreeSubject(self)
        self.sub.setWidget(self.ChildWindowleftToolBar)
        self.sub.setWidget(self.web_view)
        self.mdi.addSubWindow(self.sub)
        self.sub.show()
```

图 5.7 子界面生成模块核心代码

六、数据存储模块

扫描结果接收直接跳转数据存储模块，icmp 先进行数据分析（差错报文控制），根据结果和用户要求存储到文件或软件中。本段代码展示的是用户选择将结果保存到数据库中，注意数据库在扫描开始的时候已经创建好了，现在是将结果发送到对应数据库。

①部分识别代码

部分识别代码如下图 5.8 所示：

```
def start(type,code,target_addr):
    if type==0:
      if code==0:
        result='%s---回显应答'%target_addr
        print('%s---回显应答'%target_addr)
        pySQL.insert_new_data(target_ip=target_addr, icmp_result=result, symbol=1)
        return '%s---回显应答' % target_addr
    elif type==3:
      if code==0:
        print('%s---网络不可达' % target_addr)
        result='%s---网络不可达' % target_addr
        pySQL.insert_new_data(target_ip=target_addr, icmp_result=result)
        print("1")
        return '%s---网络不可达' % target_addr
      elif code==1:
        return '%s---主机不可达' % target_addr
        pySQL.insert_new_data(target_ip=target_addr, icmp_result=result, name=name)
        print("1")
      还有很多，代码类似不进行展示。
```

图 5.8 部分识别代码如下

②主要依据 icmp 差错报文控制，如下图 5.9 所示。

类型	代码	名称	查询	差错
0	0	回应应答(Echo Reply)	√	
3		目的地不可达		√
	0	网路不可达		√
	1	主机不可达		√
	2	协议不可达		√
	3	端口不可达		√
	4	需要分片和不需要分片标记置位		√
	5	源路由失败		√
	6	目的网络未知		√
	7	目的主机未知		√
	8	源主机被隔离		√
	9	目的网络的通告被禁止		√
	10	目的主机的通信被禁止		√
	11	对请求的服务类型 ToS，目的网路不可达		√
	12	对请求的服务类型 ToS，目的主机不可达		√
	13	由于过滤，通信被强制禁止		√

图 5.9 icmp 差错报文控制

③依赖 mysql 官方提供的 pymysql 模块对 sql 进行数据传输

如下图 5.10 所示

```
def db_template(sql_statement):
    db_connection = pymysql.connect(
        host='localhost',
        user='root',
        password='root',
        db='test',
        charset='utf8')
    # db_connection = database_connection()
    car = db_connection.cursor()  # pymysql.connect是高速公路，conn.cursor()是运送的小车
    try:
        sql_statement = sql_statement
        car.execute(sql_statement)  # 卸货
        db_connection.commit()
          # mysql返回值
        #print(car.fetchone())
    except Exception as e:  # 方法一：捕获所有异常
        # 如果发生异常，则回滚
        print("发生异常:", e)
        db_connection.rollback()#执行rollbak；语句后又回到最初的数据状态
    finally:
        # 最终关闭数据库连接
        car.close()
        db_connection.close()
def creat_table():
    print("aaaaaaaaa")
    # db_connection=database_connection()
    # creat_table_car = db_connection.cursor()  # pymysql.connect是高速公路，conn.cursor()是运送的小车
    time()
    sql_creat_table = 'create table {0}(id int(11)  AUTO_INCREMENT ,源IP char(20) DEFAULT NULL,' \
                      '目标IP char(20) DEFAULT NULL,通讯状态 char(100) DEFAULT NULL ,symbol int(2) default null ,' \
                      '线程运行名 char(20) DEFAULT NULL,时间 char(40),PRIMARY KEY ( id ))'.format(table_name)
    db_template(sql_creat_table)
def insert_new_data(target_ip, icmp_result,symbol=0,name='null'):#,locallhost,target_ip,icmp_result#需要这些数据
    ceshitime = datetime.datetime.now().strftime('%H%M%S')
    # table_name1=table_name
    locallhost1=locallhost
    target_ip1=target_ip
    icmp_result1=icmp_result
    sql_insert_data="insert into {0} values(null,'{1}','{2}','{3}','{4}','{5}','{6}');".format(table_name,locallhost1,target_ip1,icmp_result1,symbol,name,ceshitime)
    db_template(sql_insert_data)
```

图 5.10 pymysql 部分代码

第六章　图形化内网设备故障监测系统测试

通过对图形化内网设备故障监测系统的功能模块进行设计后，本章节对其进行软件测试。通过测试来确定该系统是否满足对内网设备检测的要求，并且能够发现系统的功能有没有缺陷，这样才能更好的提升系统的性能。本文旨在通过对内网设备故障监测系统进行功能模块测试，并对其测试环境、测试方案、预期结果和测试结果进行评价，以验证该系统的可行性和合理性。

第一节　系统测试的意义

软件系统测试的意义在于验证和验证软件系统的正确性、完整性、可靠性、安全性和易用性等方面，以保证软件系统能够满足用户的需求和期望。

一、有助于发现潜在的缺陷和问题

软件系统测试的主要目的是帮助发现潜在的缺陷和问题，早期发现和解决这些问题有助于提高软件系统的质量和可靠性，同时能够节省软件开发过程中的时间和成本。

二、提高软件系统的质量和可靠性

软件系统测试的另一个目的是提高软件系统的质量和可靠性，通过测试可以发现软件系统的弱点和缺陷，从而进一步完善和优化软件系统，以提高其性能、稳定性和安全性。

三、保障软件系统的安全性

随着互联网和移动互联网的普及，越来越多的软件系统被用于处理重要和敏感的信息，因此，软件系统的安全性也成为了软件测试的一个重要方面。通过对软件系统的安全测试，可以发现并解决可能存在的漏洞和问题，从而确保软件系统的安全性和数据的保密性。

四、增强软件系统的易用性

软件系统测试还可以帮助测试人员和用户发现软件系统的易用性问题，包括界面设计、操作流程、用户体验等方面，通过测试可以收集用户的反馈和建议，并进一步改进和优化软件系统的设计和功能，从而增强软件系统的易用性和用户满意度。

综上所述，软件系统测试在现代软件开发中具有非常重要的意义，它不仅可以发现和解决软件系统的缺陷和问题，提高软件系统的质量和可靠性，还可以保障软件系统的安全

性和数据的保密性，增强软件系统的易用性和用户满意度。因此，在软件开发的各个阶段都需要进行软件系统测试，以确保软件系统的质量和可靠性，提高软件系统的用户体验和用户满意度。

第二节　测试环境

测试平台物理环境，系统测试环境如表 5.1 所示。

表 5.1 系统测试环境表

参数	内容
内网环境	ENSP 模拟器模拟标准化考场环境
服务器	本地计算机；腾讯云服务器
运行环境	Windows10 Professional22H219045.2846
其他工具	Wireshark、vmware

测试平台内网环境，测试设备信息表如表 5.2 所示。

表 5.2 测试设备信息表

设备位置	设备种类	设备名称
教室 101 列 101-1	PC	PC101-1-1
	PC	PC101-1-2
	PC	PC101-1-3
教室 101 列 101-2	PC	PC101-2-1
监控室	PC	PC13-1
	二层交换机	列 101-1
		列 101-2
		教室 101
		1 楼汇聚层交换机
		监控室
	三层交换机	G1 教学楼
Cloud4	Vm	Kali
		Win10
	本地物理机	Win10（测试）

测试平台内网环境设备详细信息，测试设备详细信息表如表 5.3 所示。

设备名称	IP	Mac 地址	网关
PC101-1-1	192.168.101.5	54-89-98-73-09-5B	192.168.101.254

PC101-1-2	192.168.101.6	54-89-98-B2-47-32	
PC101-1-3	192.168.101.7	54-89-98-7C-7D-6D	
PC101-2-1	192.168.101.10	54-89-98-9D-24-4D	
PC13-1			
列 101-1			
列 101-2			
教室 101			
1 楼汇聚层交换机			
监控室			
G1 教学楼			
Kali	192.168.101.128	00-0C-29-34-8A-46	192.168.101.254
Win10	192.168.101.16	00-0C-29-AA-A9-6D	
物理机	192.168.101.50	00-0C-29-XX-8A-46	
VMnet8 网卡	192.168.101.1	00-50-56-C0-00-08	

表 5.3 测试设备详细信息表

第三节　测试详情

一、内网监测测试

对图形化内网设备故障监测系统的监测功能进行测试，测试方法与结果如表 5.4 所示：

表 5.4 监测功能测试

测试目的：测试本文系统对内网设备监测功能是否正常运行。	
检测方法：	检测结果：
1.在现有的内网设备环境中添加新的网络设备并开启设备网卡	在本文系统结果读取中显示新的网络设备图标和相关信息
2.在现有的内网设备环境中添加新的网络设备但关闭设备网卡	在本文系统结果读取中显示无新的网络设备加入
测试结果：本文系统对内网设备监测功能是正常运行，结果读取及时显示新加入的网络设备。	

二、图形化编程功能测试

对本文系统子界面图形化编程功能的各项拖动模块进行测试，测试方法与结果如表 5.5 所示：

表 5.5 图形化编程功能测试表

测试目的：检测本文系统子界面图形化编程功能块的各项拖动模块是否正常运行。

测试方法：	测试结果：
1.用户拖动图形化编程功能块能否与其他模块组合生成功能链	能够其他模块组合生成功能链
2.功能链通过合约表能否正确推出用户的图形化编程功能块组合	能正确推出用户的图形化编程功能块组合
3.通过本文系统的监测结果能否正确推出用户的图形化编程功能块组合	能正确推出用户的图形化编程功能块组合

三、备故障监测功能测试

对本文系统设备故障监测功能的准确度进行测试，本测试主要使用 ICMP 协议模块，测试方法与结果如表 5.6 所示：

表 5.6 故障监测功能表

测试目的：对本文系统设备故障监测功能的准确度进行测试。	
测试方法：	测试结果：
1.对内网环境已知网络设备的 ICMP 监测端口进行关闭	本文系统提示该网络设备端口不可达，ICMP 监测返回包为类型 3 代码 3 端口不可达
2.对内网环境已知网络设备的 ICMP 回复功能进行关闭	本文系统提示该网络设备协议不可达，ICMP 监测返回包为类型 3 代码 2 协议不可达
3.对内网环境已知网络设备的网卡进行关闭	ICMP 监测返回包为类型 3 代码 0 主机不可达

结　论

本章对本文基于 Python3 的图形化内网设备故障监测系统设计与实现进行总结与展望。

一、总结

本文基于 Python3 的图形化内网设备故障监测系统设计与实现，通过本文的研究和实验总结出以下问题:

针对其他开发者二次开发方面：

设计方面：虽然使用了 EA 软件并采用 uml2，但是系统部分细节设计依旧不清楚，系统部分细节功能并没有按规定详细设计出设计图。

规范方面：虽然使用了驼峰命名法等方法以及规范目录，但命名依旧随意、系统结构较为混乱，让其他开发者无法从文件名中直接联想文件内容。

开发方面：模块化开发较为混乱，部分方法无法实现更快更换方法的设定、部分方法没有进行注释，为其他开发者二次开发造成困扰。

针对用户方面：

界面方面：虽然细节功能很多，但部分功能并没有给用户带来实际意义。

功能方面：核心功能完成，但部分功能并未实现或开发不完整。

二、展望

后期将本系统上传到开源代码平台，让更多的开发者和志愿者丰富本文系统功能。让更多的有经验的开发者指出系统出现的问题，规范自我开发、更熟悉软件设计开发流程。降低入门 DIY 家用网络设备门槛的同时，让用户提前熟悉代码开发。

致　谢

参考文献

附　录

一、英文文献

二、中文文献