注：此处无需更改。

阅后删除此文本框。



本科生毕业设计(论文)

北京理工大学本科生毕业设计（论文）题目

The Subject of Undergraduate Graduation Project (Thesis) of Beijing Institute of Technology

注：此处是论文中英文题目，中文题目，居中，字体：华文细黑，加黑，字号：二号，行距：多倍行距1.25，间距：段前、段后均为0行，取消网格对齐选项。英文题目，与中文题目对应，居中，字体：Times New Roman，字号：三号，加黑，行距：多倍行距1.25，间距：段前、段后均为0行，取消网格对齐选项。阅后删除此文本框。

|  |  |
| --- | --- |
| 学 院： |  |
| 专 业： |  |
| 班 级： |  |
| 学生姓名： |  |
| 学 号： |  |
| 指导教师： |  |

注：此处按照实际情况填写即可。打印（宋体，三号）或手写都可以。阅后删除此文本框。

20XX 年 月 日

原创性声明

注：页眉内容无需更改。阅后删除此文本框。

本人郑重声明：所呈交的毕业设计（论文），是本人在指导老师的指导下独立进行研究所取得的成果。除文中已经注明引用的内容外，本文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

特此申明。

本人签名： 日 期： 年 月 日

关于使用授权的声明

本人完全了解北京理工大学有关保管、使用毕业设计（论文）的规定，其中包括：①学校有权保管、并向有关部门送交本毕业设计（论文）的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存本毕业设计（论文）；③学校可允许本毕业设计（论文）被查阅或借阅；④学校可以学术交流为目的,复制赠送和交换本毕业设计（论文）；⑤学校可以公布本毕业设计（论文）的全部或部分内容。

本人签名： 日 期： 年 月 日

指导老师签名： 日 期： 年 月 日

北京理工大学本科生毕业设计（论文）题目

注：此处是中文题目，居中，字体：黑体，加黑，字号：小二，行距：单倍行距，间距：段前、段后均为1行，取消网格对齐选项。阅后删除此文本框。

摘　要

注：此处无需更改。阅后删除此文本框。

本文……。

摘要正文选用模板中的样式所定义的“正文”，每段落首行缩进2个字符；或者手动设置成每段落首行缩进2个汉字，字体：宋体，字号：小四，行距：固定值22磅，间距：段前、段后均为0行。【阅后删除此段】

摘要是一篇具有独立性和完整性的短文，应概括而扼要地反映出本论文的主要内容。包括研究目的、研究方法、研究结果和结论等，特别要突出研究结果和结论。中文摘要力求语言精炼准确，本科生毕业设计（论文）摘要建议300-500字。摘要中不可出现参考文献、图、表、化学结构式、非公知公用的符号和术语。英文摘要与中文摘要的内容应一致。【阅后删除此段】

关键词：北京理工大学；本科生；毕业设计（论文）

注：此处字体：黑体、小四号、加粗；

一般选3-8个单词或专业术语，且中英文关键词须对应。阅后删除此文本框。

注：页脚内容为页码，宋体、五号，居中排列。阅后删除此文本框。

The Subject of Undergraduate Graduation Project (Thesis) of Beijing Institute of Technology

注：正文设置成每段落首行缩进2字符，字体：Times New Roman，字号：小四，行距：固定值22磅，间距：段前、段后均为0行。阅后删除此文本框。

注：此处是英文题目，居中，字体：Times New Roman，加黑，字号：三号，行距：单倍行距，间距：段前、段后均为1行，取消网格对齐选项。阅后删除此文本框。

Abstract

In order to study……

Key Words: BIT; Undergraduate; Graduation Project (Thesis)

注：Key Words与摘要正文之间空一行。Key Words与中文“关键词”一致（3-8个）。词间用分号间隔，末尾不加标点；Times New Roman，小四，加粗。阅后删除此文本框。

目　录

注：此处无需更改。阅后删除此文本框。

[摘　要 I](#_Toc8720747)

[Abstract II](#_Toc8720748)

[第1章 一级题目 1](#_Toc8720749)

[1.1 二级题目 1](#_Toc8720750)

[1.1.1 三级题目 1](#_Toc8720751)

[结　论 3](#_Toc8720752)

[参考文献 4](#_Toc8720753)

[附　录 6](#_Toc8720754)

[致　谢 7](#_Toc8720755)

注：在目录页面中点击鼠标右键，选择“更新域”，在弹出窗口中选择“更新整个目录”，确定即可自动生成目录。章、节标题和页码，字体：宋体，字号：小四，不加粗。阅后删除此文本框。

第1章 一级题目

1.1 研究背景与意义

[6]在半导体等技术的快速发展下，嵌入式设备逐渐在生活中逐渐普及，嵌入式系统作为其核心组成部分，[1]被广泛应用于工业控制、物联网、汽车电子和医疗设备等领域，执行如控制流水线上各种电机和执行器，进行远程监控等重要任务[2]从定义上，嵌入式系统是嵌入到对象体系中的专用计算机系统，它的有几种特征区别于通用计算系统。

首先是资源约束性：嵌入式系统应用的设备通常只有很有限的内存、处理器频率以及能源供应，而这样的性能也就使得设计者需要在在安全方面与执行效率间权衡，开发者需要将资源利用率提升为设计的第一性原则。以网络协议栈为例，传统七层模型在嵌入式场景中就面临着性能问题，需要进行结构上的删减，如这次参考使用的smoltcp协议栈剔除了会话层、表示层等非必要层级，将核心功能聚焦于数据传输与路由，来提高代码的执行效率。以固定窗口替代动态拥塞控制算法，牺牲了带宽的自适应能力，使单数据包处理延迟更加可预测和稳定。还进行了内存固化，通过预分配通信缓冲区而不是在运行时动态申请，来消除内存碎片化风险，确保中断响应时间的确定性。

其次是实时性要求：工业控制等场景需要毫秒级响应，而许多安全校验的时间开销直接影响系统确定性，这也同样影响了各种协议的设计方案。

最后是长生命周期：[5]嵌入式系统以应用为中心，面向产品，需要有较长的软件生命周期。嵌入式设备的系统通常需要在没有人员进行维护的情况下进行长时间的运行，如汽车电子有十余年的服役期要求，而这些要求也就需要系统具有充分的可靠性保证。

因为针对嵌入式设备的系统有上述要求，使得针对嵌入式系统的网络协议栈也需要特殊设计，而不能套用传统的网络协议栈模式。

本次研究主要针对的嵌入式系统内核是用rust进行改写的sel4微内核，微内核的设计理念是最小化内核功能，仅提供最基本的机制，减少内核负责的功能并放入用户态来实现功能间的解耦，本质就是实现模块化、从而增加灵活性和提高安全性。[8]而sel4自 2004 年开始开发，于2014年实现开源，是L4微内核模型基础上设计出来的全球首个通过形式化验证的操作系统内核。形式化验证研究的起点就是软件复杂度的提升会导致错误增多，这对任务关键型和安全关键型系统构成重大挑战，但构建可信系统的核心在于操作系统，而内核作为运行在特权模式的组件，其任何缺陷均可能导致系统性失效，所以即使证明复杂，sel4依然使用了高可信度的方法进行了形式化验证，进一步保证了其可靠性。

在另一方面，sel4在安全性的考虑上也进行了特殊的设计，其通过比传统linux更为严格的能力机制实现安全高效的访问控制，从而提供了细粒度的资源管理能力。此机制使得任务能够严格遵循最小权限原则，降低了外部攻击和意外错误的风险。其能力机制设计实现了资源抽象与令牌化，也就是所有系统资源均被抽象为不可伪造的能力令牌。任何操作必须持有对应能力令牌才可以执行。最小特权原则使得其内核启动时通过显式授权链初始化能力空间。例如，仅允许特定线程访问指定的内存区域。

[4]rel4使用rust对sel4进行重写的原因是rust作为一种系统级编程语言，其拥有优秀的内存管理机制和并发模型，在成为近年来操作系统编程的可靠工具。传统的嵌入式内核依赖开发者手动管理内存，容易导致缓冲区溢出、悬垂指针和数据竞争等问题，[7]而rust与其他语言最显著的差异就是所有权机制，他强制编程者在给出的限制下编程，从而保证变量所有权的可靠安全性。也就是rust 的所有权机制和借用检查器在编译期即可消除这些问题，[3]实验表明，在不使用unsafe语法的情况下，**内存与并发漏洞发生率为0%，也就是可以充分避免内存风险。**而其无需依赖垃圾回收等机制和方便的编译器环境适合作为系统编程语言。如上面所述，嵌入式系统因为其运行特性需要较高的可靠性和性能要求，rust语言正适合开发对可靠性和性能要求苛刻的系统组件。

1.2 国内外研究现状

sel4作为内核不是一个完整的操作系统，有很大的可开发空间，而rust作为近年的热门语言，有许多研究人员使用rust针对sel4进行进一步研究。

因为对于用户态组件的需求，sel4官方已经开始探索rust在sel4上的应用，为开发人员提供了rust 语言在 seL4 微内核用户态开发的完整工具链包括sel4、sel4-sys等api的 Rust 绑定和运行时支持等，希望通过rust的内存安全特性，来增强 sel4 用户态组件的可靠性，同时保留 seL4 内核的形式化验证优势。

在用户态组件实现上，谷歌在2022年推出了使用rust编写用户态组件的为嵌入式机器学习设备设计的开源操作系统KataOS，seL4解决内核安全，Rust解决用户态安全，从而解决智能设备，如摄像头、传感器中数据的安全性问题。尽管基于现有技术，KataOS仍有技术上的修改突破，如修改sel4内核以支持Rust编写的根服务器动态回收内存，同时保持安全性，使用上述sel4官方提供的sel4-sys来提供 rust 绑定，优化机器学习的工作负载。

1.3论文内容与结构安排

为了提升应用于rel4的网络协议栈性能，在选择smoltcp作为协议栈的基础上，对其进行性能上的提升，选择通过对smoltcp进行异步上的修改来提高收发的吞吐量，去更好的发挥rel4在实现异步后性能上的优势。

论文总共分为五章。

（1）第一章为绪论，主要介绍开发的协议栈目标系统rel4的研究背景、意义与针对sel4微内核使用rust开发的相关项目的研究进展。

（2）第二章为相关技术说明，介绍了rust提供的异步async、await模型的相关执行流程以及优势，为后续针对smoltcp协议栈的异步优化奠定理论基础。

（3）第三章为smoltcp协议栈介绍，通过分析其收发模型来得到针对性的异步优化方案。

（4）第四章进行具体的实现细节介绍，如何进行相应的修改来实现异步，从而得到性能的提升，并给出相应分析。

（5）第五章对优化后协议栈的测试结果进行说明和分析，并给出后续的研究方向

第二章：相关理论介绍

2.1 rust异步机制

本次选择的smoltcp在很多处理上为了适配嵌入式系统的特性从而进行了简化，设计上选择同步作为后续调用的方式，而异步模型在遇到阻塞操作的时候会让出线程的所有权而不是阻塞，从而提升了运行的效率。当异步函数执行遇到未就绪的操作时，会立即返回控制权，允许线程处理其他任务。从而实现高效的资源复用。Rust通过future、执行器和waker等结构和机制构建了这一模型，可以从这几个部分对其进行理解。

2.2 future的核心机制与执行原理

Future trait是Rust异步模型的基础也是核心，其本质是一个状态机，拥有两个状态pending和ready，并拥有一个成员函数poll。在运行过程中，Future会在其poll函数被调用时会触发状态的修改。当调用poll的时候任务还未完成时函数会返回pending并保存waker；当外部事件执行到特定程度时触发Waker，继而再次调用poll函数进行状态的转换，如果完成任务，就返回一个ready元祖结构体，其中存储完成的结构数据。这里trait特征的含义是可以针对不同的自定义的的数据结构实现这个future trait，而不是rust中固定的结构，在面向不同需求的时候只需要定义不同的poll函数就可以实现对于各种任务的异步实现。除此之外Rust还有Pin语法的设计来控制future的内存地址，保证数据的值在内存中不会移动导致引用错误，继而保证了引用结构的安全性问题，这也符合rust对于安全性的高要求。

3. waker与执行器

正如上面所说，如果没有在第一次poll的时候完成任务，Waker就在后续的poll中起到关键作用，其智能指针特性允许跨线程安全传递。当相应任务完成后调用wake方法将返回future的异步函数，重新加入执行队列吗，通过再次调用poll操作。

显然上述的所有poll如果需要通过编程者主动控制过于复杂，除了首次poll调用，其他的操作都是由执行器负责的，其负采用任务队列机制管理Future的生命周期，并在waker触发后对future进行poll操作来实现异步操作。而普通的异步实现就是对每个唤醒任务的控制都使用线程控制，不断的检查是否完成了相应的要求，之后返回执行，这种方法能实现异步，但是会消耗过多资源，性能上没有保证。而在这里rust异步模型中所使用的waker并不是独立使用线程进行的监控，在现实中，操作系统中会提供多路复用机制，从而实现一个线程同时阻塞等待多个事件，并且可以实现精准的触发，也就是面向网络协议栈的时候，往往有多个socket连接同时需要进行维护，如果需要实现发送接受的异步执行，会存在多个waker，这里运行时可以保证在同一个结构体的函数中注册的waker进行区分处理，从而一对一的进行唤醒，保证了异步触发的准确性。通过使用waker和执行器这种设计实现了**低成本的唤醒**。

4. async和await

async/await语法通过编译器转换为状态机实现，其转换过程保持零成本抽象原则。async块被降级为生成器式的状态机结构，每个await点对应状态迁移的分界。与JavaScript等语言不同，Rust在编译期即完成状态机展开，运行时仅需维护当前状态指针，避免了虚拟机解释开销。

这种转换产生的重要特性是线程局部性：除非显式跨线程调度，否则任务始终在初始线程执行。编译器通过控制流分析确保状态机符合Send/Sync约束，这种静态检查机制消除了竞态条件风险。值得注意的是，await语法实质是语法糖包裹的poll调用链，其协作式特性要求开发者避免长时间占用线程的同步操作。

**第三章：异步修改的方案设计**

3.1 Smoltcp的架构分析

smoltcp协议栈是我主要进行修改设计的对象。是一个轻量级、使用rust实现的嵌入式TCP/IP协议栈，主要面向嵌入式系统。作为一个网络协议栈，因为其面向嵌入式设备，可使用的资源有限，且没有操作系统提供运行时环境的条件，所以并不是以常见的osi七层模型进行的模块区分，并且相比于osi模型中应用层，表示层逐层向下传递的层次关系，smoltcp的模块也不是严格的高层低层封装关系，而是分为了socket，iface，storage，wire和phy五层，分别负责不同的功能，。

首先网络协议栈的根本目的就是与应用和设备衔接，提供数据的发送手段，Socket层就是是协议栈与应用层的交互模块，负责多种协议的抽象以及连接管理。该层实现了TCP、UDP、ICMP、DNS等协议的Socket抽象，但其设计理念与传统协议栈有所区别，在socket层中并不直接参与数据包的封装与传输过程，而是负责进行高层次的传输管理。具体针对本次主要进行修改TCP协议为例，Socket层为其实现了三次握手、流量控制、超时重传等协议的核心机制，而具体的数据包构建工作是交给其他层去执行。这种设计使得协议的处理逻辑与数据传输封装路径实现解耦，为协议扩展提供了灵活性。值得一提的是，虽然smoltcp的运行逻辑是同步，但是其中提供了async编译选项，也就是与异步相关的函数，包括对于waker的简单封装，还有包括waker的注册函数为后续的开发者提供了便捷的异步修改途径

相对与上面不负责封装管理高层细节的socket层，wire层就是协议栈负责各层封装和后续接受处理的协议编解码引擎，其设计遵循"可构建即可解析"的核心原则。该层包含了以太网帧、IP数据包、TCP/UDP报文等各层协议的构建器与解析器。例如在构建IP数据包时，wire层会自动计算校验和并填充必要字段，确保生成的报文完全符合协议规范，其中也可以通过函数控制是否进行校验和的添加，来进行性能上的管控，面向低传输准确性要求的情况可以不进行校验和的添加和检验从而减少在这部分消耗的处理器资源；在解析过程中，wire层则会进行严格的格式验证，丢弃不符合规范的数据包。这种双向一致性保障机制，使得协议栈具备自我验证能力，从根本上避免了报文本身错误导致的解析错误。而wire层通过零拷贝技术直接操作原始字节流，也提升了处理效率。

上面两层完成了封装和发送管理后，其实就是完成了osi模型中的高层管理，接下来的任务就是底层与设备的对接，iface层在负责协调Socket层与wire层的交互的基础上，会去调用设备层的函数来进行实际的收发管控。该层通过同步轮询机制驱动整个协议栈的运转：在发送方向，iface层会定期收集各Socket的发送缓冲区数据，调用wire层进行协议封装，最终将完整的数据帧提交至物理设备；在接收方向，则从物理设备获取原始数据帧，经wire层解析后分发至对应Socket的接收缓冲区，交给socket层和应用进行后续的数据处理。这种显式的poll驱动模型，与常见操作系统内核的中断驱动架构形成鲜明对比，虽在实时性方面存在理论劣势，却显著降低了上下文切换开销，更适应资源受限的嵌入式场景。

storage层通过静态预分配的缓冲区管理策略，实现了确定性的内存使用。该层提供环形缓冲区、分片缓冲区等多种数据结构，支持零拷贝的数据传递。例如在TCP传输过程中，应用层数据直接写入Socket的发送环形缓冲区，iface层在poll周期内批量获取这些数据进行协议封装，整个过程避免了数据复制带来的性能损耗。这种设计对内存碎片化问题具有天然免疫力，特别适合长期运行的嵌入式系统。

phy层作为硬件抽象层，定义了统一的网络设备接口。该层支持环回接口、TUN/TAP虚拟设备、原始套接字等多种物理/虚拟设备的接入，通过trait抽象屏蔽具体设备的差异性。开发者可通过实现Device特质，其中对与设备进行了统一的定义，对不同的网络传输介质进行了传输函数的要求。对设备的传输能力也定义了四个关键属性：介质类型、最大传输单元、最大突发尺寸和校验和能力。对于之后实现的不同设备都需要填充上述属性为后续操作服务。如校验能力使用标志位定义，可以进行启用和关闭，体现了设计上的灵活性。在发送和接收上的主要设计点在于接收令牌（RxToken）和发送令牌（TxToken）。其要求发送和接受时首先获取令牌，也就是获得对于缓冲区的处理权，再interface层进行处理之后，通过consume方法进行实际的发送接收操作，分步进行保证了发送数据的正确处理，在完成构建后进行发送，避免因为中断等情况导致发送被临时打断，发送无效数据包，这种设计保证了报文构造的可靠性，还实现了高效的批量发送机制。

Smoltcp中是没有给出高层的封装的，也就是并没有一个函数真正封装了发送和接收的函数，其需要使用者去进行编写，也就提供了进一步进行逻辑修改的可能性，适合对其进行修改。

3.2 smoltcp收发处理分析

完成上述smoltcp各层功能以及相应函数的分析后，就可以结合得到他的运行逻辑。即一个网络设备维护多个socket连接，这部分是由socket层定义的，而发送是多个函数的调用结果，首先是调用socket的send函数，把数据存储到socket本身的缓冲区中，之后每个设备有一个interface，其会以同步的方式调用poll函数，而poll函数可以分为两个功能，一个是发送，也就是依次取出所有socket中的缓冲区的内容，首先进行数据报的封装，这部分由通过wire层的函数进行，之后存储到设备本身的发送缓冲区中，由设备驱动进行发送；而另一个功能是接收，在完成发送后，会检查设备本身的接受缓冲区，有数据的情况下将其取出并进行数据报解析，将解析结果放入相应的socket的接受缓冲区中，之后实际处理使用通过socket调用recv函数进行。

3.3 Embassy-net和axnet的介绍

其次是对embassy-net和starry-os中网络模块axnet的分析，二者都是在smoltcp的基础上进行了封装，为其提供更好的性能，从上面的分析可以看到socket的send函数，recv函数，还有interface的poll函数都是通过同步的方式，也就是通过轮询控制执行，axnet在实现中使用了WouldBlock机制来协调非阻塞I/O，这里它的设计目标并非直接实现异步I/O，而是通过与操作系统的调度协作，在用户态模拟类似异步的效果，从而进行性能提升，减少无效判断，任务在完成的情况下则返回值，未完成则返回WouldBlock。然后分为两种模式，在non-blocking模式下，子任务未完成，父任务立即返回WouldBlock；而在blocking模式下，子任务未完成，父任务轮询到其完成为止。轮询中交替检查网络包状态和任务状态。这样的实现相比与异步会因为重试多次上下文切换，还是会有性能消耗。

而embassy-net中进行了进一步的提升，使用了异步的方式进行函数调用，基于异步运行时，将smoltcp的同步轮询接口改造为异步任务驱动模型。具体就是将send和recv定义为async函数，借用smoltcp中本身就有实现的waker机制，在准备好数据的情况下调用waker唤醒async函数，来实现异步执行，减少了空闲时的CPU占用，允许其他任务在等待网络事件时执行，提升系统整体效率。两种优化方式为我的后续性能优化改进提供了借鉴思路。

3.4修改方案

正文……

正文部分：宋体、小四；正文行距：22磅；间距段前段后均为0行。【阅后删除此段】

图、表居中，图注标在图下方，表头标在表上方，宋体、五号、居中，1.25倍行距，间距段前段后均为0行，图表与上下文之间各空一行。【阅后删除此段】

图-示例：【阅后删除此段】



图1-1 标题序号

结　论

注：此处无需更改。阅后删除此文本框。

本文结论……。

结论作为毕业设计（论文）正文的最后部分单独排写，但不加章号。结论是对整个论文主要结果的总结。在结论中应明确指出本研究的创新点，对其应用前景和社会、经济价值等加以预测和评价，并指出今后进一步在本研究方向进行研究工作的展望与设想。结论部分的撰写应简明扼要，突出创新性。【阅后删除此段】

结论正文样式与文章正文相同：宋体、小四；行距：22磅；间距段前段后均为0行。【阅后删除此段】

参考文献

注：此部分蓝色字体为注释，阅后可删除；黑色字体为具体示例。

阅后删除此文本框。

注：此处无需更改。阅后删除此文本框。

参考文献书写规范

参考国家标准《信息与文献参考文献著录规则》【GB/T 7714—2015】，参考文献书写规范如下：

1. 文献类型和标识代码

普通图书：M 会议录：C 汇编：G 报纸：N

期刊：J 学位论文：D 报告：R 标准：S

专利：P 数据库：DB 计算机程序：CP 电子公告：EB

档案：A 舆图：CM 数据集：DS 其他：Z

2. 不同类别文献书写规范要求

[2] “The Rust Programming Language: Its History and Why It Matters”, talentopia.com, 2020, accessed: May 21, 2022.

1. 陈星光,雷先华.嵌入式控制系统在工业控制中的关键应用研究[J].电子元器件与信息技术, 2023(11):38-41.
2. 何立民.嵌入式系统的定义与发展历史[J].单片机与嵌入式系统应用,2004,(01):6-8.
3. 胡霜,华保健,欧阳婉容,等.Rust语言安全研究综述[J].信息安全学报,2023,8(06):64-83.DOI:10.19363/J.cnki.cn10-1380/tn.2023.11.06.
4. 顾锡华.Rust语言在Web开发的应用研究[J].电脑知识与技术,2024,20(05):38-40.DOI:10.14004/j.cnki.ckt.2024.0186.
5. 马义德,刘映杰,张新国.嵌入式系统的现状及发展前景[J].信息技术,2001,(12):57-59.
6. 魏煜康.嵌入式SMP环境下的TCP/IP协议栈并行优化研究与实现[D].北京邮电大学,2024.DOI:10.26969/d.cnki.gbydu.2024.002188.
7. Ralf Jung, Jacques-Henri Jourdan, Robbert Krebbers, and Derek Dreyer. 2021. Safe systems programming in Rust. Commun. ACM 64, 4 (April 2021), 144–152. <https://doi.org/10.1145/3418295>
8. Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. 2010. SeL4: formal verification of an operating-system kernel. Commun. ACM 53, 6 (June 2010), 107–115. https://doi.org/10.1145/1743546.1743574

**期刊**

[序号] 主要责任者. 文献题名[J]. 刊名, 出版年份, 卷号(期号): 起止页码.

[1] 余雄庆. 飞机总体多学科设计优化的现状与发展方向[J]. 南京航空航天大学学报, 2008, 40(4): 417-426.

[2] Hajela P, Bloebaumj C L, Sobieszczanski-Sobieski J. Application of Global Sensitivity Equations in Multidisciplinary Aircraft Synthesis[J]. Journal of Aircraft, 1990, 27(12): 1002-110.

**普通图书**

[序号] 主要责任者. 文献题名[M]. 出版地: 出版者, 出版年: 起止页码.

[3] 张伯伟. 全唐五代诗格会考[M]. 南京: 江苏古籍出版社, 2002: 288.

[4] O’BRIEN J A. Introduction to information systems[M]. 7th ed. Burr Ridge, III: Irwin, 1994.

**会议论文集**

[序号] 主要责任者．题名:其他题名信息[C]. 出版地: 出版者, 出版年.

[5] 雷光春. 综合湿地管理: 综合湿地管理国际研讨会论文集[C]. 北京: 海洋出版社, 2012.

**专著中析出的文献**

[序号] 析出文献主要责任者. 析出题名[M]//专著主要责任者. 专著题名. 出版地: 出版者, 出版年: 起止页码.

[6] 白书农. 植物开花研究[M]//李承森. 植物科学进展. 北京: 高等教育出版社, 1998: 146-163.

**学位论文**

[序号] 主要责任者. 文献题名[D]. 保存地: 保存单位, 年份.

[7] 张和生. 嵌入式单片机系统设计[D]. 北京: 北京理工大学, 1998.

[8] Sobieski I P. Multidisciplinary Design Using Collaborative Optimization[D]. United States -- California: Stanford University, 1998.

**报告**

[序号] 主要责任者. 文献题名[R]. 报告地: 报告会主办单位, 年份.

[9] 冯西桥. 核反应堆压力容器的LBB分析[R]. 北京: 清华大学核能技术设计研究院, 1997.

[10] Sobieszczanski-Sobieski J. Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems[R]. NASA CP-3031, 1989.

**专利文献**

[序号] 专利所有者. 专利题名:专利号[P]. 公告日期或公开日期[引用日期]. 获取和访问路径. 数字对象唯一标识符.

[11] 姜锡洲. 一种温热外敷药制备方案: 881056078 [P]. 1983-08-12.

**国际、国家标准**

[序号] 主要责任人. 题名: 其他题名信息[S]. 出版地: 出版者, 出版年: 引文页码.

[12] 全国信息与文献标准化技术委员会. 文献著录: 第4部分 非书资料: GB/T 3792.4-2009[S]. 北京: 中国标准出版社, 2010: 3.

**报纸文章**

[序号] 主要责任者. 文献题名[N]. 报纸名, 年(期): 页码.

[13] 谢希德. 创造学习的思路[N]. 人民日报, 1998-12-25(10).

**电子文献**

[序号] 主要责任者. 电子文献题名[文献类型/载体类型]. (发表或更新日期) [引用日期]. 获取和访问路径. 数字对象唯一标识符.

[14] 姚伯元. 毕业设计(论文)规范化管理与培养学生综合素质[EB/OL]. [2005-02-02]. 中国高等教育网教学研究.

关于参考文献的未尽事项可参考国家标准《信息与文献参考文献著录规则》（GB/T 7714—2015）

附　录

注：此处无需更改。阅后删除此文本框。

附录相关内容…

附录是毕业设计（论文）主体的补充项目，为了体现整篇文章的完整性，写入正文又可能有损于论文的条理性、逻辑性和精炼性，这些材料可以写入附录段，但对于每一篇文章并不是必须的。附录依次用大写正体英文字母A、B、C……编序号，如附录A、附录B。【阅后删除此段】

附录正文样式与文章正文相同：宋体、小四；行距：22磅；间距段前段后均为0行。【阅后删除此段】

致　谢

注：此处无需更改。阅后删除此文本框。

值此论文完成之际，首先向我的导师……

致谢正文样式与文章正文相同：宋体、小四；行距：22磅；间距段前段后均为0行。【阅后删除此段】