基于 zigbee 的智能家居协议转换系统的实现

专业：计算机科学与技术　　年级:2015 级　　学生：高婷婷　指导教师：孙桂鸿

摘　要

近年来, 随着人工智能技术的不断成熟, 其应用范围变得愈来愈大。在智能化浪潮的冲击下，我国生产制造业不断向着智能制造方向转型升级，家居产品所具有的自动化、智能化含量也不断提升。同时，随着智能家居产品概念被越来越多的人群所接受，科技改变生活的理念也渐渐渗透到人们生活中的各个领域。在传统智能家居中，设备大量的采用有线传输 RS485 连接模式，尤其在对接空调、地暖等 485 接口的设备时，需要将这些设备的 485 接口通过长距离的布线管道和弱电箱中的中控相连，使得现场施工周期长，成本高，难以维护，并存在布线困难以及有些环境无法布线等诸多缺点。

为了针对原有空调、地暖等设备对接到先前中控设备需要使用长距离的 RS485 布线进行优化，开发一款基于 zigbee 的智能家居协议转换系统，实现 zigbee 与 RS485 协议之间的转换，可以将设备直接安装在空调、地暖等设备周围，然后通过 zigbee 无线协议直接与中控（网关）进行对接，可大量地减少布线和施工的难度，方便智能家居产品的对接，设计该产品将会使智能家居产品连接更加高效和安全。

本论文研究分析了智能家居 zigbee 协议转换系统的硬件电路整体设计，介绍了硬件各个模块的电路设计与实现。智能家居协议转换系统包含了负责 zigbee 与 RS485 协议转换的硬件电路板，以及用于控制系统入网退网操作、zigbee 和 RS485 协议相互转换的烧录程序。通过软硬件结合，设计出一款适用于当前智能家居行业中不同网络间的无线传输协议系统。

智能家居协议转换系统采用 zigbee 无线通信技术作为组网技术，将传统的家居系统有线布线方式改成无线传输方式，解决传统家居行业中布线繁琐、维护成本高等痛点。

关键词：zigbee 无线网络　　RS485　　协议装换　　智能家居

目　　录

1 绪论

1.1 研究背景及意义

据相关数据统计，2017 年中国智能家居市场规模突破 3000 亿元，规模十分庞大，预计未来几年智能家居行业市场规模将进一步扩大，截止 2018 年我国智能家居市场将近 4000 亿[1]。目前智能家居产品中，家电类产品智能化较多，空调、地暖、冰箱、洗衣机等家电是智能化需求最高，而大多数中央空调、地暖等设备对接到开关面板都需要使用长距离的 485 进行布线，存在布线繁琐、施工周期长、维护成本高等缺点。为了解决以上痛点，急需研制一款体积小、成本低、高效便捷的智能家居协议转换装置是当下智能家居行业所迫切需要的。

1.2 研究现状

经过数年的发展变革，智能家居市场在国内的热度一度上升[2]，大多资本企业十分看好智能家居行业发展潜力，全球智能家居行业前景看好。“轻量化、系统化”是未来智能家居长期发展的大势所在，我国各大企业推动智能家居从豪宅市场向普通居民家庭普及，消费趋势也从中高端到大众化，让绿色健康、安全环保的智慧生活早日进入千家万户，让生活变得更简单和极致。

随着物联网等高新技术的发展，越来越多的技术应用于智能家居行业。该行业拥有着强而有力的各类无线通讯技术，zigbee、Z-Wave、5G 技术、蓝牙、人工智能等技术越来越成熟，与智能家居技术的深度融合将产生强大的影响力，逐渐应用于智能家居领域。在国家政策的驱动下，智能家居产品将会越来越普及，做到让寻常百姓家也能够轻而易举的享受可用的 AI 赋能的智慧家居，同时以“轻”入手，在安装、时间、成本等方面提升用户价值，满足大家对高品质生活的需求。

在过去智能家居系统的综合布线方式，使得智能家居产品一直停留于高端市场。装修前需要长时间布线，并且需要专业人士的设计也是制约智能家居发展的一大因素。因此 zigbee 无线传输技术的广泛应用，有效解决了传统有线连接方式所带来的弊端，数字无线技术在全球也得到了大规模得应用拓展，其灵活、便捷、高效、无盲点等特征倍受广大技术人员青睐。

1.3 研究内容

本项目研究的目的就是为了提供高效便捷的智能家居协议转换系统，能够将

开关面板

传输得来的指令数据通过 zigbee 无线传输至协议转换装置，再输出到带有 485 接口的家电设备上，可实现开关面板与带有 485 接口的家电设备进行无线传输，操作简单高效。地产装修人员可直接将本协议转换系统连接到家电设备的输出端，进行设置相应工作环境使其自动工作，提高了使用效率，降低了现场布线成本和维护成本。

2 主要技术介绍

2.1 ARM 单片机开发

单片机是一个具有 CPU，随机存储器 RAM，只读存储器 ROM，多个 I/O 端口和中断系统，具有数据处理能力的定时器/计数器等功能的小型完善的微机系统[3]。

学习使用单片机之前，需要对模拟电路、数字电路基础、C 语言以及汇编语言等基本理论知识有一定的了解。通过对单片机的工作原理和内部结构进行深入了解，根据电路原理图，使用 IO 口进行读写电平操作从而实现对外围电路的控制，必要时可以借助芯片手册对单片机内部寄存器的定义、芯片外围电路的设计和典型应用电路的使用规范进行学习了解。

采用单片机开发其最大优点是体积小，功耗低，输入输出接口简单，集成度和性价比高，具有超快唤醒速度，配备可扩展功率放大器和高性能 MCU。本系统选用 ARM 公司 Cortex 系列 EFR32 单片机，具有兼容 zigbee 和蓝牙的功能，引出多个 GPIO，可以任意配置成想要的外设，超快唤醒速度，配备可扩展功率放大器、集成平衡-不平衡转换器和高性能 MCU，且 Silicom Lab 在 zigbee 无线领域位居前列。

2.2.1 特性

（1）芯片特性：

32 位 ARM Cortex-M4 内核，最高工作频率为 40MHz[4]，带有 DSP 指令和浮点单元，可以实现高效率的数字信号处理，并拥有 64kB RAM 和 512kB 闪存，在 EFR32MG 系列设备中引脚兼容（耐受 5V 电压的引脚除外）。拥有 12 信道外围设备反射系统，可以实现与外围设备间进行自主交互，系统拥有集成功率放大器（PA），Tx 功率高达 20dBm，拥有 31 个 GPIO 口。

（2）芯片逻辑组成：

CPU：主要由运算器、控制器、中断及部分外部特殊功能寄存器组成，是用来对计算机中的指令进行解析并处理。

ROM：工作过程中只能读取数据，而不能改写数据，主要用来存放程序、表格和数据。

RAM：用于存放既可读又可写的数据，是可以与 CPU 之间进行直接交换数据的一种内部存储器。

GPIO 口：引脚共有 31 个 GPIO 口，可以任意配置成想要的外设，如串口收发，SPI 通讯，IIC 通讯等，该模块可用于协调器、路由器以及终端设备开发。

2.2.2 引脚结构

本系统硬件系统使用的是 Cortex 系列 EFR32 单片机，封装为 48Pin_DIP 类型，引脚结构如图 2-1 所示。

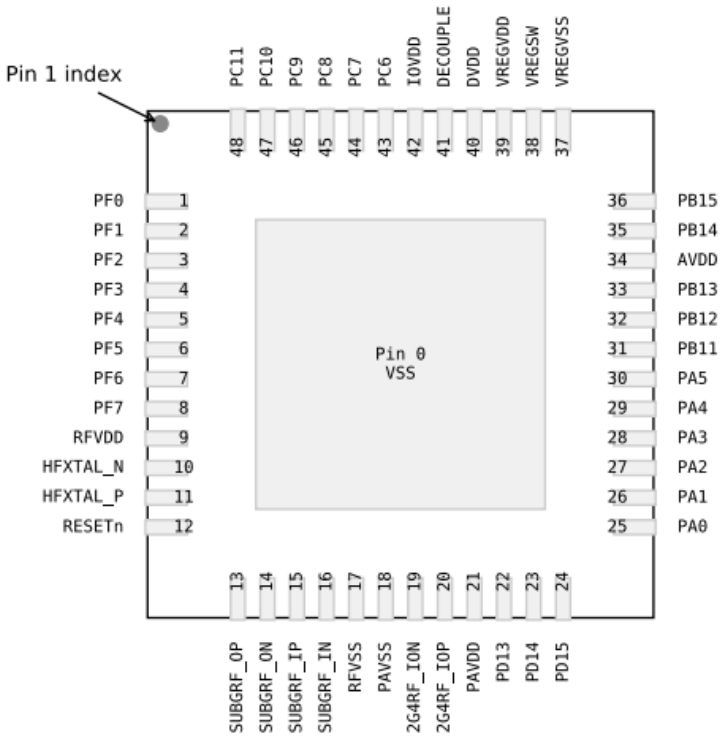

图 0-1 EFR32MG13 _DIP48 引脚图

2.2 zigbee 技术

在系统开发前，对目前无线家居各协议进行对比，其主要协议有蓝牙、Z-Wave、WIFI、zigbee 等常见协议应用于智能家居行业[5]。蓝牙由于其传输距离短，点对点通讯，不适用智能家居组网要求；Zwave 为树状组网，其节点容量偏低，稳定性差，容易受到干扰，无加密，安全性不好，在国内使用的非民用频段，无法普及使用；Wifi 部署方便，但不支持低功耗，但适合大数据量，高速率传输，在该领域中只能够起到协助作用。在智能家居系统中，其最看重的是系统的稳定性、可用性、敏捷性与安全性，而近几年 zigbee 技术在智能家居行业被各大企业大范围应用。

zigbee 技术是一组基于 IEEE 802. 15. 4 无线标准研发的有关组网、应用软件

和通讯安全方面的技术，主要用于功耗较低、距离较短且传输速率不高的各种电子设备之间进行信息传递，并且具有独立的无线电通信标准[6]。支持自组网，系统容量大，传输速度快，抗干扰力强，安全性强，可扩展性强采用 2.4G 频段，有相关智能家居控制国际标准，更适用于智能家居系统。

2.3 RS485 技术

RS485 是从早期的 RS232 发展出的串行通讯标准，具有协议简单，开放性高，容易对接，开发成本低等优点，并且规定了电气标准，但是没有规定协议。主要应用在工业自动化领域，它改善了 RS232 只能实现点对点通讯的不足，还可以进行组网，采用主从式组网，但组网结构落后，网络稳定性差。在智能家居系统中常见的 485 设备有电动窗帘、家庭影音、报警主机、空调、地暖等。采用手拉手式的布线，施工过程复杂，组网模块数量有限，两个设备同时发出请求，容易形成冲突。

2.4 zigbee 协议栈

一般来说，协议的具体实现是协议栈形式，是连接用户和协议的桥梁，可理解为类型库和代码库，便于开发人员进行上层调试引用，协议栈较底部层级和应用是相互制衡、相互独立的，开发人员可以通过调用底层协议栈来使用这个协议，可以促使无线收发数据。zigbee 协议栈可以分为四层：物理层(PHY)、媒体访问控制层(MAC)、网络层(NWK)及应用层(APL)[7]。zigbee 协议栈的每个层级中的接口都能输入或输出为簇格式的数据。zigbee 协议栈层次结构如图 2-2 所示。

图 0-2 Zigbee 协议栈层次结构

3 系统需求分析

智能家居协议转换系统能够实现 zigbee 与 RS485 协议之间的转换，可以将设备直接安装在空调、地暖等带有 485 接口设备的周围，提高施工人员现场布线效率，可以与智能家居产品进行高效对接，从而解决传统智能家居带来的布线繁琐、施工周期长、维护成本高等缺点。

3.1 功能分析

智能家居协议转换系统的硬件电路为 zigbee 与 RS485 无线通讯的协议装换提供支持，并分析设计系统运行所需要的工作电路，完成了本系统转换系统的硬件电路设计。并使用 C 语言编写基于 zigbee 协议栈的 zigbee 和 RS485 收发两大模块，为协议转换系统提供软件支持。

3.2 用户及其特点

智能家居协议转换系统主要提供给各大房地产商、现场布线人员使用，现场布线人员一般拥有较好的受教育程度，对新的知识及技能具有良好的接受能力和学习能力，通过阅读系统使用说明就能很快理解并使用智能家居协议转换系统。

3.3 系统可行性分析

3.3.1 系统在技术上的可行性

智能家居协议转换系统的硬件电路设计部分使用 ORCAD capture 和 ALLEGO 工具进行电气连接。通过学习使用 ORCAD capture 软件绘制电路原理图，提供可连续的仿真数据，是市面上特点鲜明、功能丰富、操作简单、用户界面设计简洁的原理图制作软件。该软件拥有许多可直接使用的控件，并且可以在线下载使用各元件数据库中的元件，有助于提高绘制原理图的效率。使用 ALLEGO 工具遵循 PCB 布线规则进行走线，并调整元器件布局，能够避免信号线间相互干扰，是具有简易操作功能的一款电路板设计软件，内含仿真和 PCB 各器件间连接功能，高效自动化编辑，使设计人员加快设计 PCB 板。

软件部分采用 C 语言进行开发，C 语言具有逻辑严谨，条理清晰，符合硬件开发的特性。并产生少量的机器语言，具备跨平台能力及可移植性高，可读性高，

运行速度快等优点。开发人员只需要使用简洁的语句就可以灵活的进行 C 语言开发，表达方式灵活实用，数据类型和运算符丰富，而且允许直接访问物理地址，可以对硬件地址进行操作。

（1）供电模块选型介绍

由于本系统采用 5V、3.3V 的供电方式，而输入电源采用的是 12V 供电方式，评估之后决定采用 DC-DC 和 LDO 两种方案进行供电模块的电路设计。

1） DC-DC（12V—5V）电源模块选型分析如下：

方案一 开关电源：它将常见恒流电压转变为低压直流的电源电压，二极管和绝缘栅双极型晶体管为开关电源中的主要器件。但由于开关频率低、驱动效率低，使其在整流电路和需要软件启动的电路中被少量使用。

方案二 SY8292：是一个高效率 800 千赫，自适应恒定关闭时间控制的非同步降压直流-直流调节器，能够提供 2A 输出电流。SY8292 可以在一个宽范围内运行输入电压范围从 5V 到 40V，为了减少电路导通带来的损耗，其集成了 RDS 和主开关模块。SY8292 的外部电路电容、电感封装小，具有输出电压稳定、纹波小的特点，并使用 800 kHz 开关频率。

选择及论证：相比于开关电源，SY8292 稳压芯片的输出纹波较小、导通损耗更低、输出电压值更精准。本系统设计的主要目的是为了保证系统性能更加稳定，不轻易受到输出电压波动的干扰。而开关电源的输入电压不能过大，虽然电源内部可以通过调节 TL431 的两个分压电阻，但电压过大还是会导致其他环节受保护停止工作。

因此最终选择稳压芯片 SY8292 作为稳压模块的芯片。

2）LDO（5V—3.3V）电源模块选型分析如下：

方案一 稳压管：常见稳压管利用 pn 结反向击穿状态，电流在很大范围内变化而电压基本不变，制成的起稳压作用的二极管[8]。稳压管在处于击穿状态时，它的两端电压基本保持不变，尽管通过稳压管的电流变化很大，但起到的稳压作用极小。

方案二 AZ1117H-3.3TRG1：它是一种低失率的三端调整器[9]，是一款只用于降压的芯片。并且能够稳定输出特定电压，保证了主控芯片供电的稳定性，其外围电路设计更加简单，更有利于集成电路设计，固定输出电压为 1.2V、1.5V、1.8V、2.5V、3.3V、5.0V。

选择及论证：相比于稳压管，AZ1117H-3.3TRG1 的功耗更低、稳定性更强，且能够输出固定电压值，抗干扰能力强。

因此最终了选择稳压芯片 AZ1117H-3.3TRG1 作为稳压模块的芯片。

（2）485 模块选型介绍

方案一 TP75176E：是一个真正的多点标准芯片，利用差分输入接收机实现最大抗噪性和共模抑制，接收器很容易满足相应驱动程序支持的数据速率[10]，具有高速率传输、解码译码（将 TTL 转成 16 进制）、保护电路的特性。

方案二 HN232CP：是一款 TTL 转 232 电路和 TTL 转 485 电路的结合芯片[11]，具有超强的系统稳定性、可适用于大部分接口电路，响应速度较快，但是其成本过高。

选择及论证：相比于 HN232CP，TP75176E 性价比高，电路简洁，抗噪声性强，由于本系统主控芯片无 RS232 接口，只有 TTL 的串口，因此选用 TP75176E 已经足够实现 485 转 TTL 的功能，满足主控和被控设备直接的通讯。

因此最终了选择稳压芯片 TP75176E 作为稳压模块的芯片。

3.3.2 系统在操作上的可行性

智能家居协议转换系统使用便捷，并且能够通过查看指示灯状态判断入网、退网情况，将 zigbee 的 UART 串口连接上 485PHT 后，即可实现 zigbee 转 485 的功能，从而实现与家居设备间的无线对接，相信能较大程度应用于智能家居市场，被各大地产商所接受并使用。

4 系统设计与实现

4.1 系统总体设计

本系统采用 Simplicity Studio 开发工具进行主要的软件开发工作，实现 zigbee 收发模块和 485 收发模块两大功能模块。硬件部分的侧重点主要使用 ORCAD capture 和 ALLEGO 工具进行原理图、PCB 设计，为了使智能家居开关面板与带有 485 接口的设备能够进行更稳定、安全的无线传输，使用中控加强双向无线通讯。其中，开关面板主要用于设置设备模式，网关实现网络互连，协议转换系统用于实现 RS485 与 zigbee 协议间的相互转换，而 485 接口设备一般为空调、地暖等设备。首先点击开关面板，通过网关将接收到的 zigbee 无线数据传输到协议转换系统，进行 zigbee、485 协议间相互转换后从而控制设备执行相应操作。系统总体结构框架如图 4-1 所示。

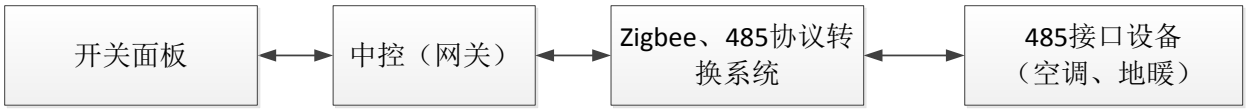

图 4-1 系统总体结构框架

4.2 硬件功能详细设计与实现

4.2.1 系统总体框架设计

协议转换系统硬件电路主要由 efr32 单片机、电源模块、485 通讯转换芯片、入网/退网电路、天线模块组成。本系统使用 38.4MHZ 晶振，使用电源稳压模块实现 12V 到 5V 的稳压设计，天线收发模块主要用于将接收到的 zigbee 信号传输到主控芯片中，RS485 模块和入网/退网按键分别用于收发 485 信号和配置系统入网和退网时的状态，而烧录插座用于烧写程序，协议转换系统框架图如图 4-2 所示。

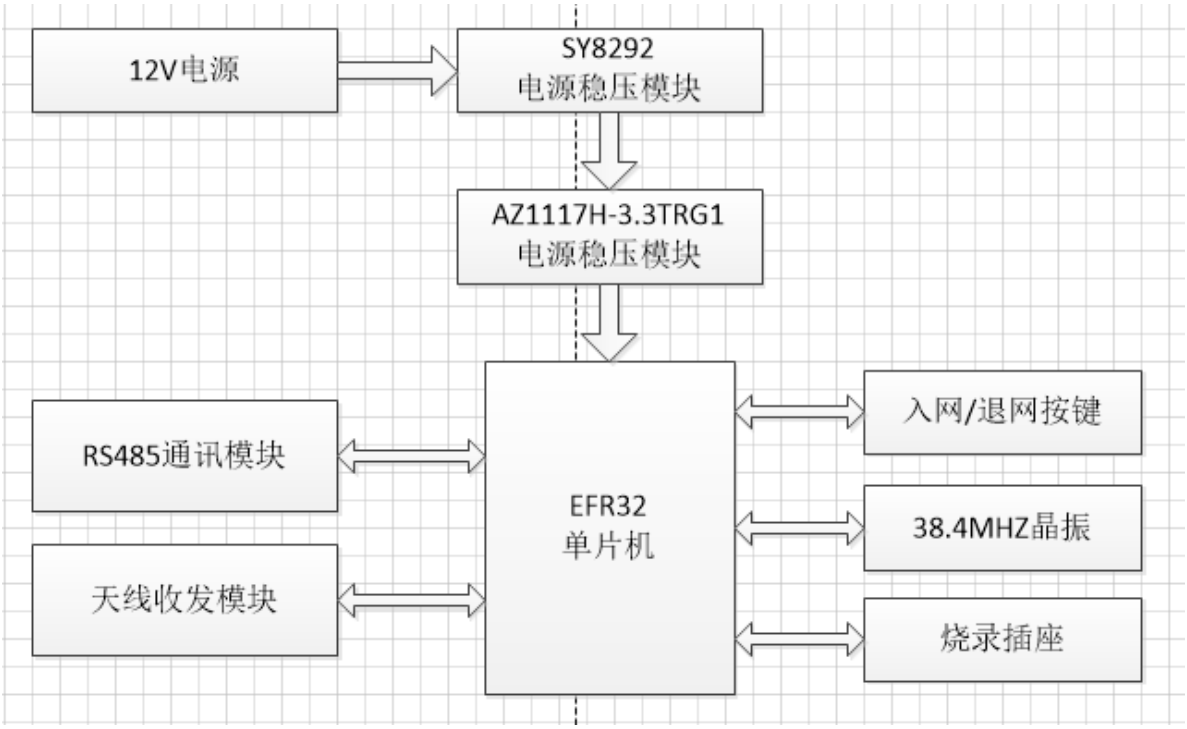

图 4-2 协议转换系统框架图

4. 2. 2 供电模块

本系统采用 SY8292 和 AZ1117H-3.3TRG1 两款 DC-DC 芯片进行稳压，下面分别对两种芯片电路设计进行详细介绍：

（1）SY8292 稳压电路设计

在选择元器件封装的过程中，为方便电路布板设计，本系统采用 SOP-6 的贴片封装 SY8292 稳压芯片对电路进行稳压设计。SY8292 引脚图如图 4-3 所示。

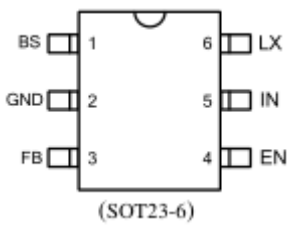

图 4-3 SY8292 引脚图

本设计采用 12V 直流供电，而系统的扩展测试接口的工作电压为 5V，所以采用 SY8292 稳压芯片进行 5V 稳压。为了降低电路的噪声纹波，在外围电路添加滤波电容，滤波电容越大，电路纹波则越小，如果不断加大滤波电容的数值，为了得到更准确的电压输出波形图，反而会使电流值过大，而损坏电流流经的元器件。从成本及效率方面考虑，通过预设电容值，对比输出波形图，得到与电路最符合的滤波电容参数。输出电压的大小主要取决于 R1、R2 的阻值大小，所以通过设定 R1 和 R2 来得到最合适的输出电压。为了使轻负荷下的功耗降到最低，最好同时为 R1 和 R2 选择大的电阻值。经过评估计算的得出，当 R1、R2 设置为 100K 和 13.7K 时，能够刚好计算得出 5V 电压值。12V-5V 的稳压电路设计如图 4-4 所示。

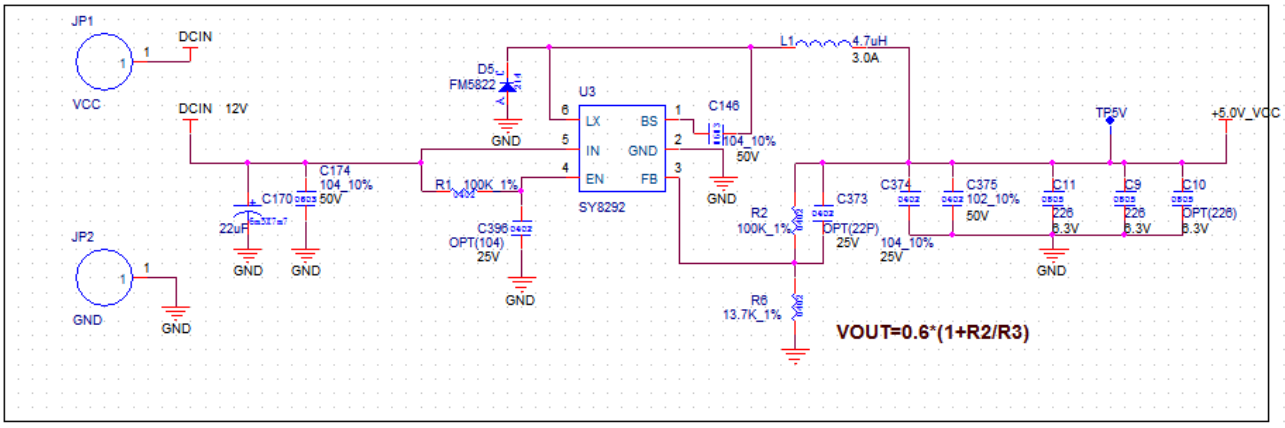

图 4-4 12V-5V 的稳压电路设计

（2）AZ1117H-3.3TRG1 稳压电路设计

由于本系统主控芯片及 RS485 通讯电路工作电压为 3.3V，所以采用 AZ1117H-3.3TRG1 芯片将从 SY8292 芯片输出的 5V 电压进行稳压为 3.3V。AZ1117H-3.3TRG1 将取样电阻、补偿电容、保护电路、大功率调整管等都集成在同一芯片上，固定输出电压为 1.2V、1.5V、1.8V、2.5V、3.3V、5.0V，是一款只用于降压的芯片，并且能够稳定输出特定电压，保证了主控芯片供电的稳定性，其外围电路更加简单易于设计，接入 5V 电压后，通过降压芯片进行降压，并外加滤波电容对电路进行滤波。为了可以直观判断系统是否已进入工作模式，本设计在输出电路周围外加了一个红色通电指示灯。5V-3.3V 的稳压电路设计如图 4-5 所示。

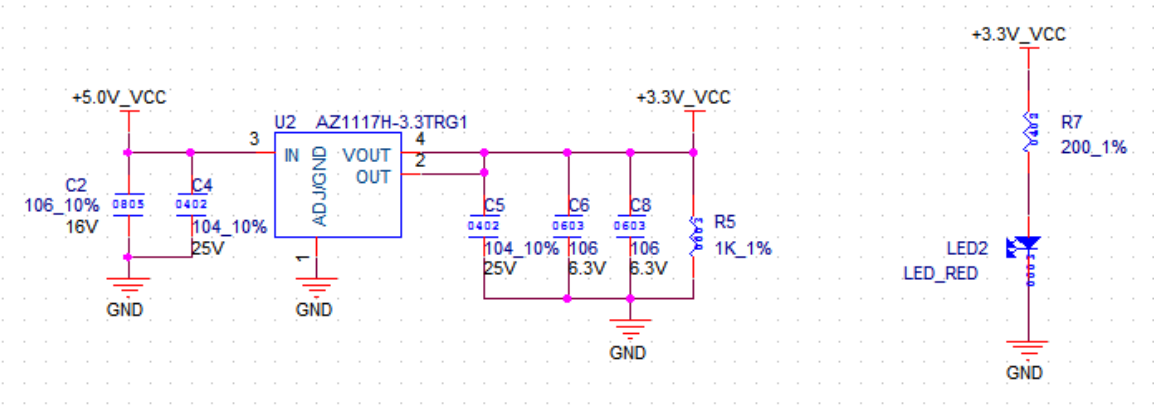

图 4-5 5V-3.3V 的稳压电路设计

4.2.3 zigbee 无线通讯模块

zigbee 无线通模块采用 ARM Cortex-M4 内核的 EFR32 芯片，其本身具有兼容 zigbee 和蓝牙的功能，采用的晶振频率是 38.4KHz。zigbee 无线通讯模块电路设计，主要包括主控芯片、天线、晶振、入网/踢网按键四大部分。入网/踢网结合按键进行设计，按下按键时，低电平导通长按 3 秒进行踢网操作，低电平信号传输至主控芯片，实现整个系统模块与中控相连，进入连网状态。天线模块使用 IPEX 端子封装形式，模块内部采用 50 欧姆的阻抗匹配电路，使天线模块效率最高，损耗最小，与主控芯片 RF 引脚连接，将天线接收到的 zigbee 信号传输到主控芯片中，通过编写软件程序代码将 zigbee 信号转换为 485 信号输出。zigbee 无线通讯模块整体电路设计如图 4-6 所示。

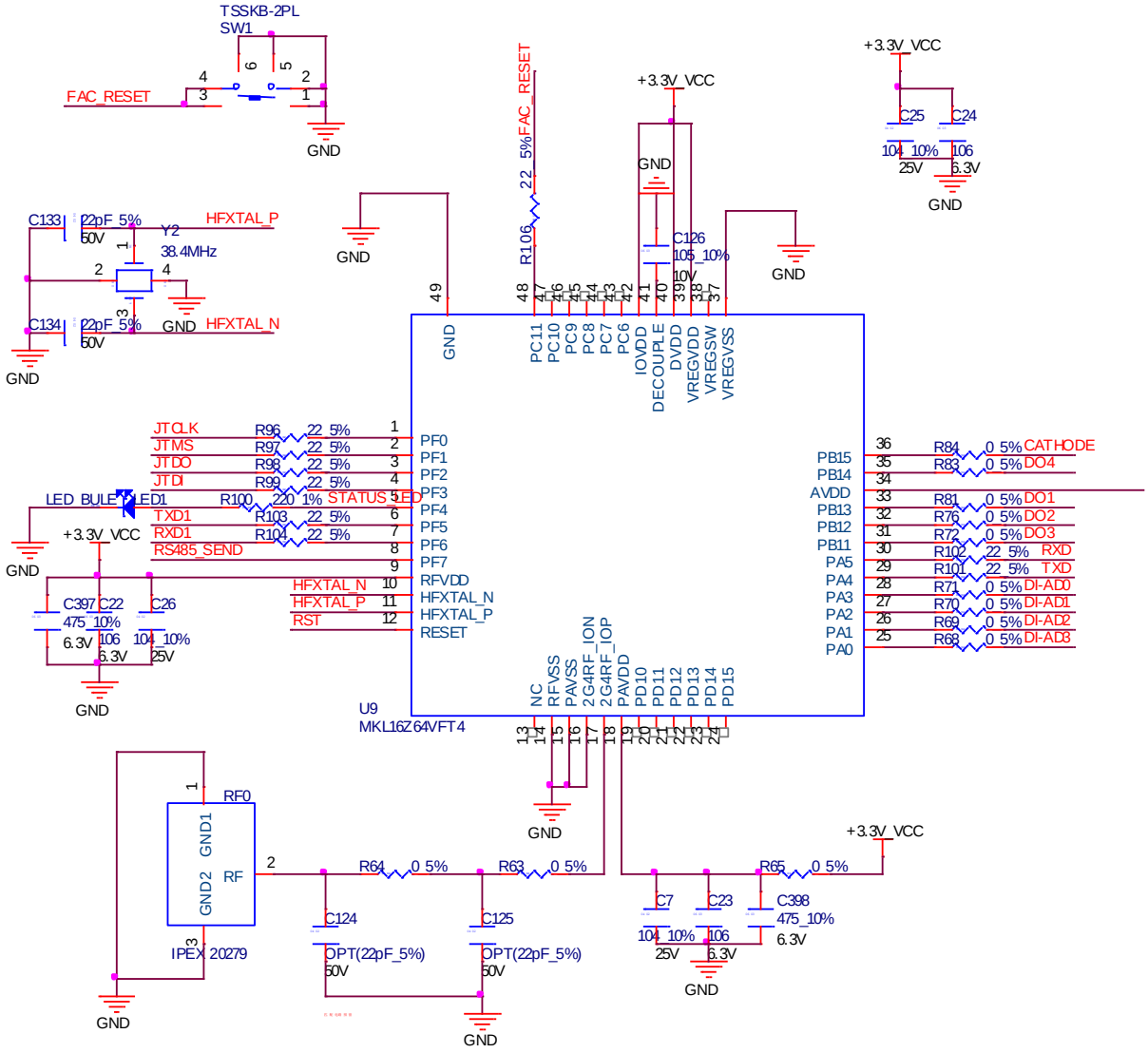

图 4-6 Zigbee 无线通讯模块整体电路设计

4.2.4 RS485 通讯模块设计

RS485 通讯模块采用的是 TP75176E 芯片，在本系统中用于收发 485 信号，是一个真正的多点标准芯片，利用差分输入接收机实现最大抗噪性和共模抑制，接收器很容易满足相应驱动程序支持的数据速率，具有高速率传输、解码译码（将 TTL 转成 16 进制）、保护电路的特性。芯片 DE 引脚用于接收 zigbee 信号，与主控芯片 RF 引脚进行电气连接，通过芯片 6、7 引脚与空调接口相连，最终实现协议转换系统与带 485 接口设备进行双向通讯。另外，电路中加入 F3、F4 保险丝进行电路保护，防止 A、B 铜柱反向传输，出现电压、电流波动或外部干扰的现象，使电路短路。RS485 通讯模块电路设计如图 4-7 所示。

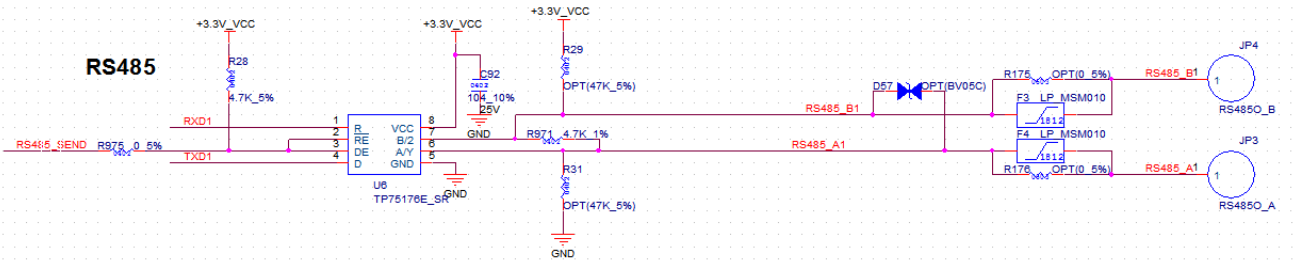

图 4-7 RS485 通讯模块电路设计

4.2.5 系统 PCB 设计

在 PCB 设计中，由于 DC-DC 部分电路的 PCB 布局布线对电路性能影响非常巨大，在这一学习设计过程中，也花费了大量时间，主板 PCB 设计如图 4-8 所示。因此在布局布线过程中应该尽量遵循如下要求：

（1）确保开关电流通道走线最短，同时最小化由输入电容，高端 MOS 管和低端 MOS 管之间形成的电流环路；

（2）输入端陶瓷电容尽量靠近芯片电源输入管脚；

（3）其中输入电压信号，开关电流信号 SW，特别是 GND 信号做大面积覆铜处理，便于芯片散热以提升芯片稳定性；

（4）相同结构电路部分，尽可能采用“对称式”标准布局；比如相同的 DC-DC 电源模块；

（5）输入电容,芯片地形成的环路要小,并且电感和输出电容形成的环路也要小，简单来讲就是输入地,芯片地以及输出地要尽量靠近；

（6）为了方便 PCB 的处理，一般要求正面与背面的器件颜色，及各层的走线，设置成不同颜色，以示区别；

（7）在布局完成后，应该仔细重新检查各器件封装是否出错。

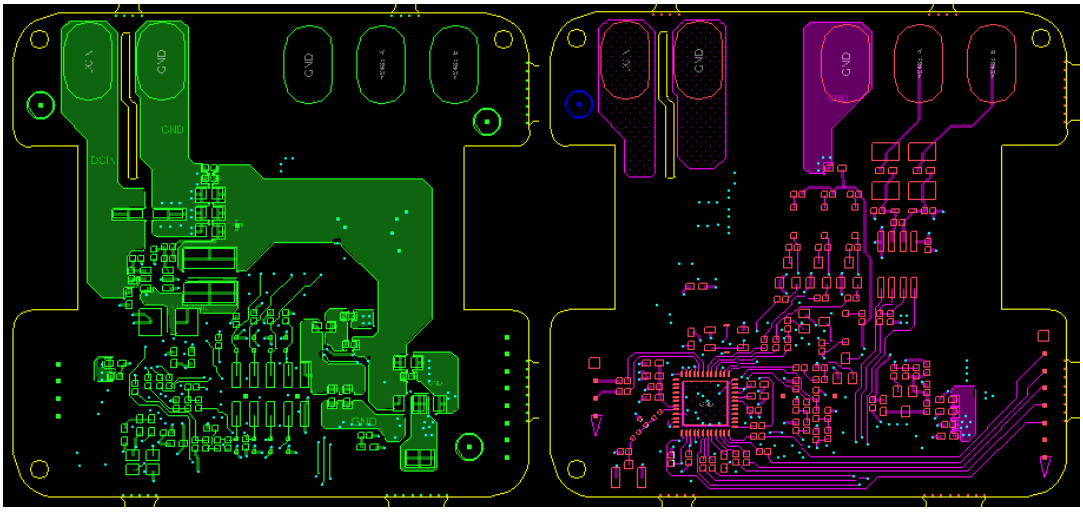

图 0-8 主板 PCB 设计

4.3 系统软件设计与实现

4.3.1 入网操作设计与实现

在设备没有连网情况下，系统无法进行数据接收与发送，所以需要预先判断网络状态。本系统通过蓝色入网指示灯来判断系统是否已入网，指示灯亮代表入网不成功，反正，入网成功。程序开始先检测设备的网络状态，得到状态后，判断是否已入网，若设备已经处于入网状态，说明设备之前已经执行过相关入网操作，无需重新配网；若设备还未进行入网操作，设备开始搜寻连接周围中控设备后，检测网络等待在超过10ms后是否检测到中控设备，若没有检测到中控设备，进行重新启动搜索；若网络等待未超过10ms时，观察入网指示灯是否处于熄灭状态，处于熄灭状态表示系统已入网成功，反之，每间隔5ms重新进行网络搜寻操作，直到检测到设备已成功入网。入网操作流程图如图4-9所示。

程序代码如下：

```
void networkEventFunction(void)
{
    static uint8_t networkWaitCount = 0;
    emberEventControlSetInactive(networkEventControl);
    EmberNetworkStatus state = emberAfNetworkState();
    emberAfCorePrintln("[%s] state = %d", __func__, state);
    if(state != EMBER_JOINED_NETWORK)
```

```
        {
            if(!isNetworkSteering)
            {
                if(networkWaitCount++ >= 10)
                {
                    networkWaitCount = 0;
                    emberAfCorePrintln("Firmware
Version:[%d]",EMBER_AF_PLUGIN_OTA_CLIENT_POLICY_FIRMWARE_VERSION);
                    emberAfPluginNetworkSteeringStart();
                    isNetworkSteering = true;
                }
            }
            if(!isNetworkLedLocked)
            {
                halToggleLed(NETWORK_STATUS_LED);
            }
            emberEventControlSetDelayMS(networkEventControl,
NETWORK_LED_BLINK_TOGGLE_TIME);
        }
        else
        {
            isNetworkInited = true;
            if(!isNetworkLedLocked)
            {
                halClearLed(NETWORK_STATUS_LED);
            }
        }
    }
```

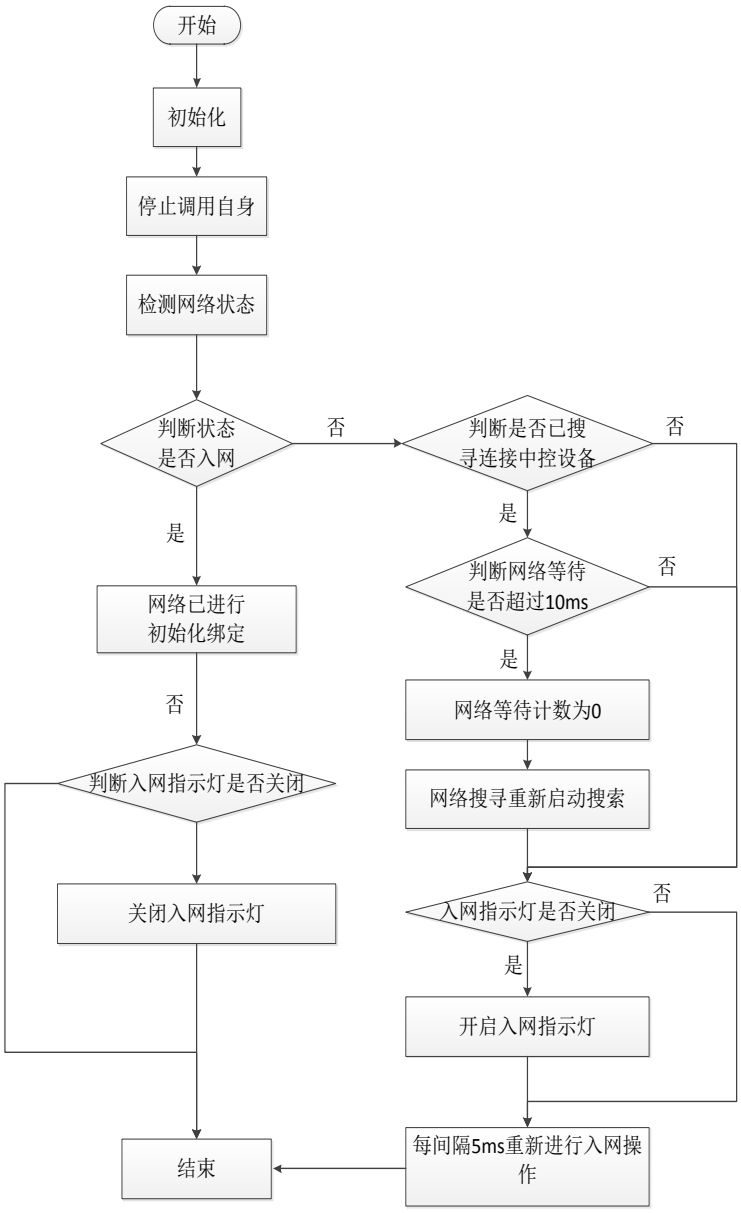

图 4-9 入网操作流程图

4.3.2 RS485 转 zigbee 数据转换功能设计与实现

预先设置串口数据接收的数组最大长度，用于储存485接口所发送的数据。由于485设备接口可能不断发送数据到协议转换器中，所以需要每间隔5ms的时间不断检测串口数据，接收到485数据后，如果接收到的数据不为空，许进一步判断设

备是否已经入网，否则将妨碍485数据进一步传输，通过调用zigbee底层协议栈功能函数，将485数据映射zigbee数据包数组后再通过应用层协议发送zigbee数据包。RS485转zigbee数据转换流程图如图4-10所示。

程序代码如下：

```
void uartRecvEventFunction(void)
{
    uint8_t msgBuf[UART_MAX_DATA_LEN] = {0};
    uint8_t appZclBuffer[UART_MAX_DATA_LEN + 3] = {0};
    uint16_t recvLen = 0;
    static uint8_t uartSeq = 0;
    emberEventControlSetDelayMS(uartRecvEventControl,
UART_RECV_INTERVAL_MS);
//读取串口数据
    emberSerialReadDataTimeout(HAL_SERIAL_PORT_USART1,
                               msgBuf,
                               UART_MAX_DATA_LEN,
                               &recvLen,
                               2,
                               5);
    if(recvLen > 0)
    {
        if(isFacTestCommand(msgBuf, recvLen))
        {
            setmanuModeCommand();
            return;
        }
        else if(isFacGetIEEECommand(msgBuf, recvLen))
        {
            getIeeeCommand();
            return;
        }
        if(emberAfNetworkState() == EMBER_JOINED_NETWORK)
        {
            uartSeq++;
```

```
                memset(appZclBuffer, 0, UART_MAX_DATA_LEN + 3);
                appZclBuffer[0]=0x01;
                appZclBuffer[1]=uartSeq;
                appZclBuffer[2]=0x81;
                memcpy(&appZclBuffer[3], msgBuf, recvLen);
//zigbee数据发送
                zigbeeSendCustomMsg(appZclBuffer, recvLen + 3);
            }
            emberAfCorePrint(" uart recv msglen =%d, message = ", recvLen);
            for(uint8_t i = 0; i < recvLen; i++)
            {
                emberAfCorePrint(" 0x%x", msgBuf[i]);
            }
            emberAfCorePrintln(" ");
        }
}
```

图 4-10 RS485 转 Zigbee 数据转换流程图

4.3.3 退网重启功能设计与实现

在设备需要停止网络服务或者重置出厂设置时，需要进行退网操作。用户通过长按退网按键使系统接收断开网络指令后，判断系统是否已经退网，若未成功接收到退网指令，将入网指示灯点亮，表示设备退网不成功，并每隔5ms重新执行退网重启函数；若成功接收到退网指令，执行退网指令，熄灭指示灯，恢复为工厂设置，通过zigbee底层协议栈调用内部系统重置后重新启动。退网操作流程图如图4-11所示。

程序代码如下：

```
void leaveNetworkAndReboot(void)
```

```
{
    emberAfWriteServerAttribute(1,
                                ZCL_BASIC_CLUSTER_ID,
                                ZCL_RS485_BAUDRATE_ATTRIBUTE_ID,
                                rs485BaudRateCmd[0].cmd,
                                ZCL_CHAR_STRING_ATTRIBUTE_TYPE);
      emberLeaveNetwork();
      emberAfBasicClusterResetToFactoryDefaultsCallback();
      optionBindingTableClearCommand();
      halInternalSysReset(RESET_SOFTWARE_REBOOT);}
```

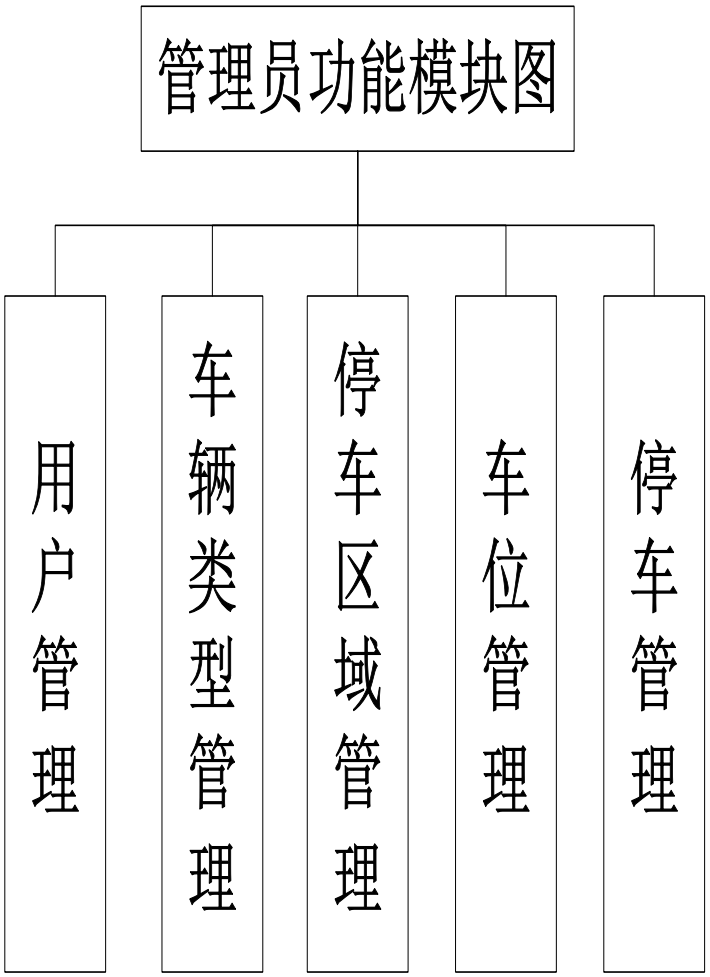

图4.12 退网操作流程图

4.3.4 zigbee 转 RS485 数据转换功能设计与实现

在本系统中，着重使用到zigbee转RS485的功能。首先使用三个标志位进行标识透传数据包，系统通过比对协议栈接收到的数据地址中zigbee数据包是否为透传数据，如果已检测到所接收的是透传数据包，通过调用zigbee底层协议栈功

能函数，读取zigbee数据包数据和长度，将485的GPIO口send引脚设置为高电平，表示可发送数据，再将串行数据发送到UART上，等待数据发送完成后将引脚拉低，切回为接收状态，防止传输错误数据。485采用的是数据包通信方式，在程序中加入适当延时处理，采用这种处理方式会使总线在状态切换时，有一个稳定的工作过程。zigbee转RS485数据转换流程图如图4-12所示。

程序代码如下：

```
boolean emberAfPreCommandReceivedCallback(EmberAfClusterCommand* cmd)
{
   if(cmd->apsFrame->clusterId==0xFD01 && cmd->buffer[0]==0x01 &&
cmd->commandId==0x80)
   {
      int ret = 0;
      uint32_t timeSendStart = halCommonGetInt32uMillisecondTick();

      rs485DataSend(cmd->buffer + cmd->payloadStartIndex,
                    cmd->bufLen - cmd->payloadStartIndex);

      emberAfCorePrint("uart sent ret=%d, time=%d, msglen=%d, message=
",
                       ret,
                       halCommonGetInt32uMillisecondTick() -
timeSendStart,
                       cmd->bufLen - cmd->payloadStartIndex);
      for(uint8_t i = 0; i < cmd->bufLen - cmd->payloadStartIndex; i++)
      {
         emberAfCorePrint(" 0x%x", *(cmd->buffer +
cmd->payloadStartIndex + i));
      }
      emberAfCorePrintln(" ");
      return true;
   }
   return false;
}
```

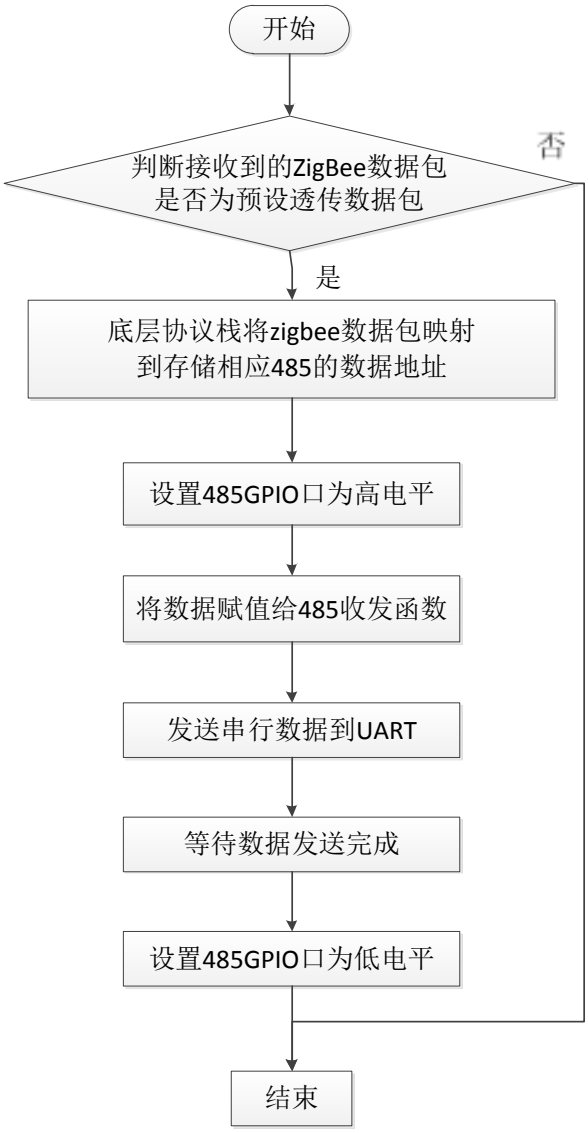

图 4-12 Zigbee 转 RS485 数据转换流程图

5 系统测试

5.1 系统测试说明

本系统主要为协议转换功能，所以在硬件电路系统测试过程中，首先要保证的是各个元器件各个模块是否连接正常、各个焊点是否虚焊，否则后面的一切努力都是空谈。硬件的测试主要分为电源方面的测试和信号方面的测试，并且需要分别测试测试系统的各个硬件模块的性能。由于本文篇幅有限，本文主要对稳压电路测试和系统纹波噪声测试进行介绍。测试过程中，使用万用表和示波器分别检查电路之间是否存在短路现象以及纹波检测指标，焊盘元器件比较接近的位置检查有没有短路，有趣需要检查电源和地之间是否存在短路现象。在排除硬件故障之后，在结合软件进行整个系统的测试，确保系统通讯功能稳定。

软件采用 silicomlabs 的 zigbee 协议栈进行开发，主要测试设备能够成功接收入网和退网指令，并执行入网退网操作，其次判断采集的数据是否能够通过协议转换系统回传至 PC 端，PC 端是否能够成功通过串口接收到 485 数据等。

5.2 硬件调试

5.2.1 稳压电路测试结果

本系统的输出电压分别为 3.3V 和 5V，采用 SY8292 和 AZ1117H-3.3TRG1 两款 DC-DC 芯片进行电路稳压。5V 电压测量结果如表 5-1 所示，3.3V 电压测量结果如表 5-2 所示。

表 5-1 5V 电压测量结果

序号	标准值（V）	测量值（V）	相对误差（%）
1	5.00	4.94	-0.12
2	5.00	5.01	0.20
3	5.00	5.00	0.00
4	5.00	4.98	-0.40
5	5.00	4.96	-0.08

6	5.00	5.03	-0.60

表 5-2 3.3V 电压测量结果

序号	标准值（V）	测量值（V）	相对误差（%）
1	3.30	3.38	-0.24
2	3.30	3.26	-0.12
3	3.30	3.30	0.00
4	3.30	3.32	0.06
5	3.30	3.24	0.18
6	3.30	3.30	0.00

经过多次电压测量发现，SY8292 和 AZ1117H-3.3TRG1 芯片输出电压稳定，且相对误差波动不大，符合电路设计要求。

5.2.2 系统纹波噪声测试结果

由于在电容上形成一个充电和放电的过程，从而会造成电压波动较大，且此波动的频率与开关管的开关频率是相同的，这种电压波动称为输出纹波，是电源测试中的一个很重要的标准，纹波越小越好，能够延长产品工作寿命。而噪声是开关电源自己本身产生的一种高频脉冲串，噪声电压的大小在很大程度上与开关电源的拓扑、测试时的外部电磁环境有一集 PCB 的布线方式有很大关系。

在测试过程中，使用示波器分别测试 3.3V 和 5V 电源电压的纹波大小和频率，5V 波形图测试结果如图 5-1 所示, 3.3V 波形图测试结果如图 5-2 所示。由于本系统主要在低频情况下工作，不需要测试在满偏情况下的纹波和频率，由于目前市面上几乎不存在开关频率达到 20MHZ 的电源芯片，所以选择在 20MHZ 带宽下进行测试。在 Inter-ATX12V 2.31 电源规范中，规定纹波标准值为电压值 1%，即+5V 输出纹波不得超过 100 毫伏,+3.3V 纹波不得超过 33 毫伏,通过在长周期和短周期情况下分别测试不同电压点，观察纹波是否会发生突变，用于验证电源的输出电压稳定性。纹波噪声测试结果如表 5-3 所示。

表 5-3 5V 纹波噪声测试结果

纹波 / 周期	5V	3.3V
50ms	68	26

50us 76 20

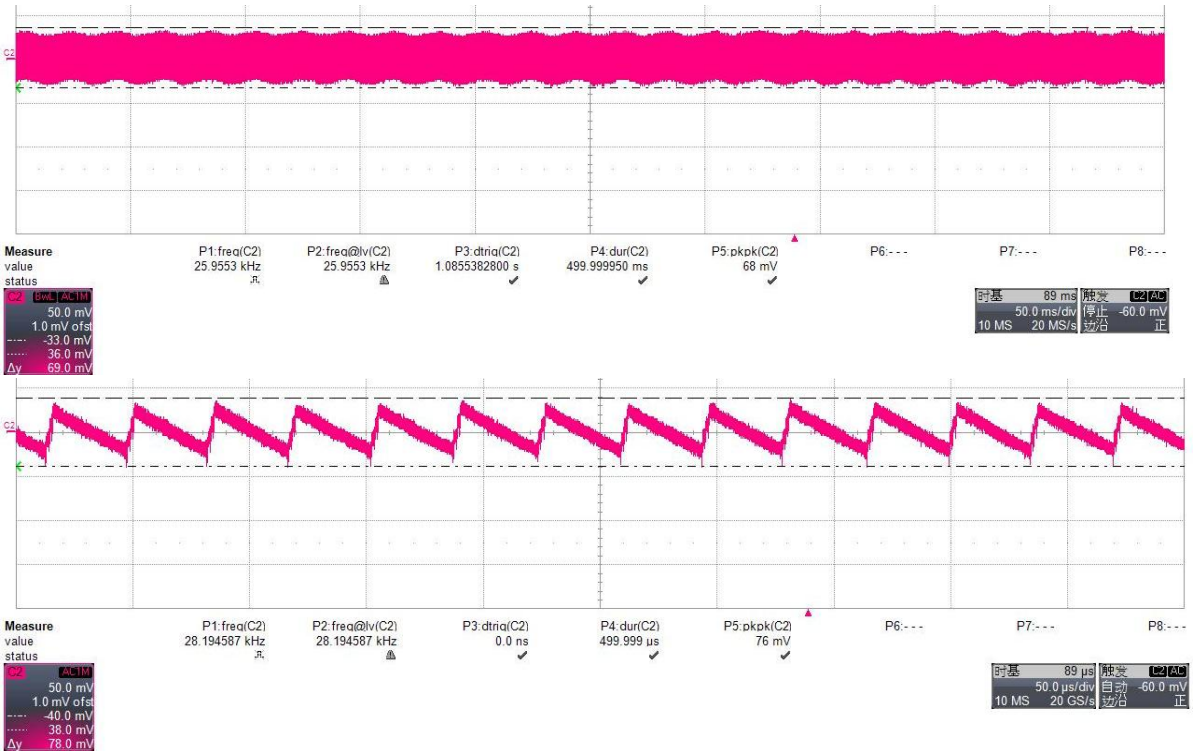

图 5-1 5V 波形图测试结果

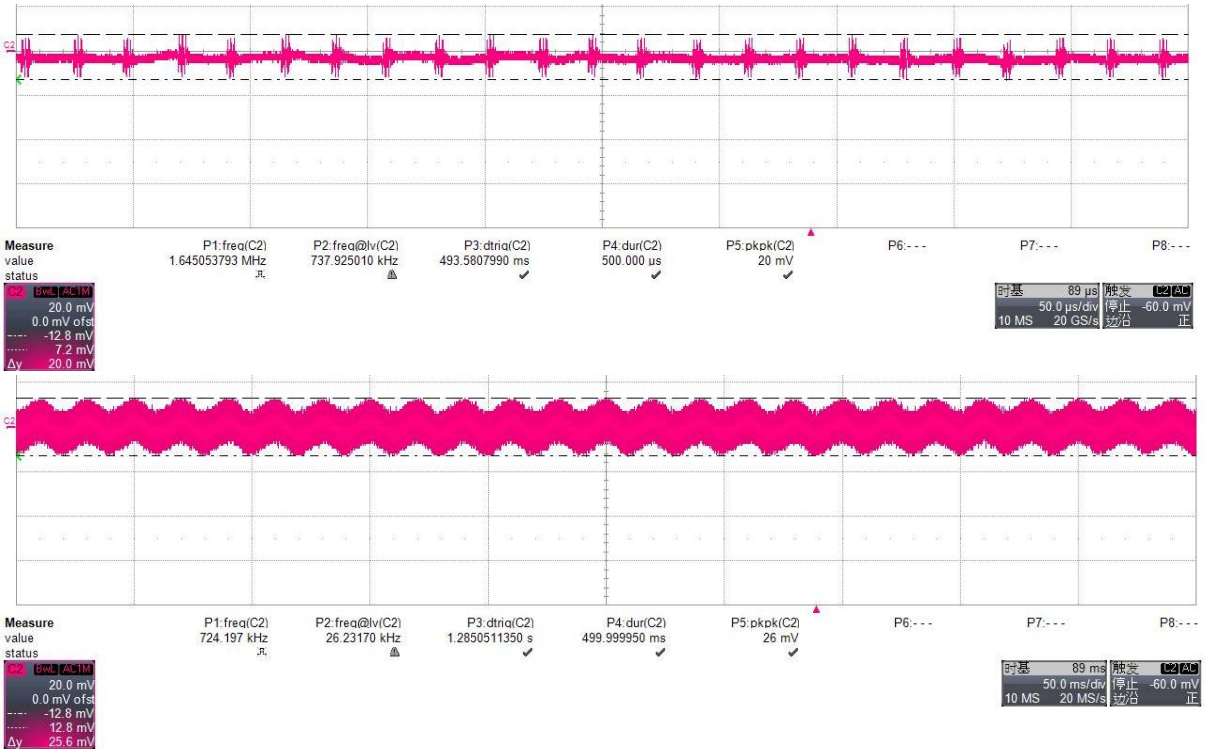

图 5-2 3.3V 波形图测试结果

5.3 系统功能测试

（1）本系统通过串口助手连接协议转换系统，接收 zigbee 指令后，通过程序解码转换，观察串口助手软件是否能够接收到完整数据指令，PC 端在接收串口数据时需要使用 USB 转 RS485 通讯转换器。测试硬件连接如图 5-3 所示。

图 5-3 串口测试硬件连接图

（2）通过配网操作设置，发生入网指令，观察入网指示灯是否熄灭。若未熄灭代表系统未成功连网。反之则代表系统入网成功，最终测试入网蓝色指示灯熄灭，设备成功入网，测试成功，如图 5-4 所示。

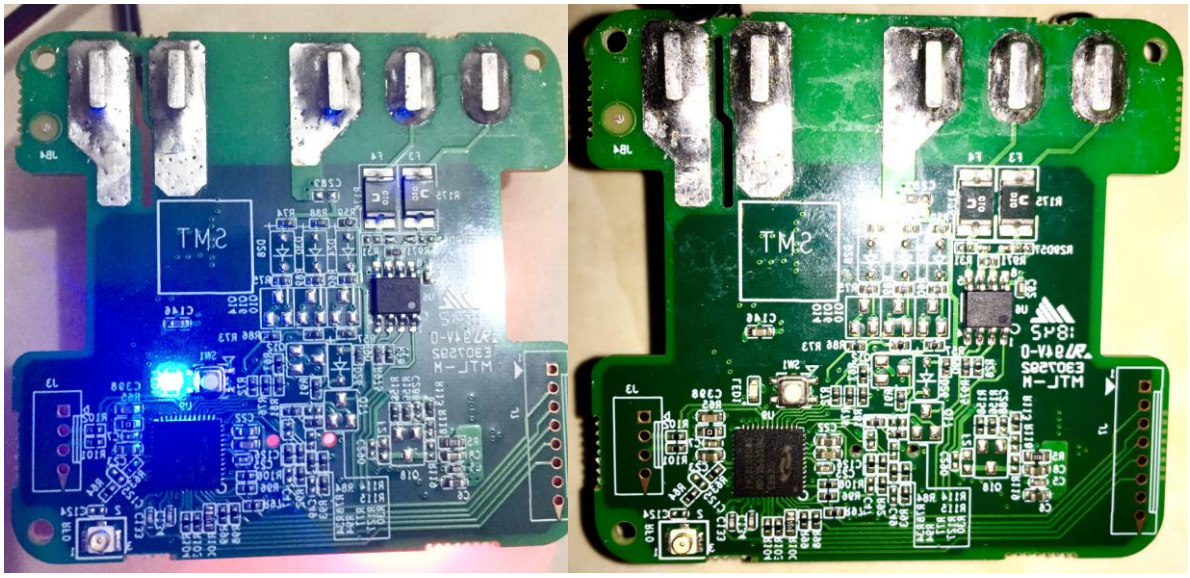

图 5-4 设备成功入网示意图

（3）zigbee 协议栈底层通过调用智能家居码库，每条指令依次对应不同的功能，如当开关面板点击开升高空调按钮时，串口软件则接收到 00 06 0F A2 00 1A AB 26 数据，当点击关闭空调时，串口软件则显示接收到 00 06 0F A2 00 00 8B 2D 数据。当没有执行任何操作时，系统将会发生自定义 zigbee 数据包至串口软件，表示设备仍然处于工作状态，其自定义指令为 00 03 00 00 00 08 45 DD，系统测试成功。串口接收数据指令界面如图 5-5 所示。

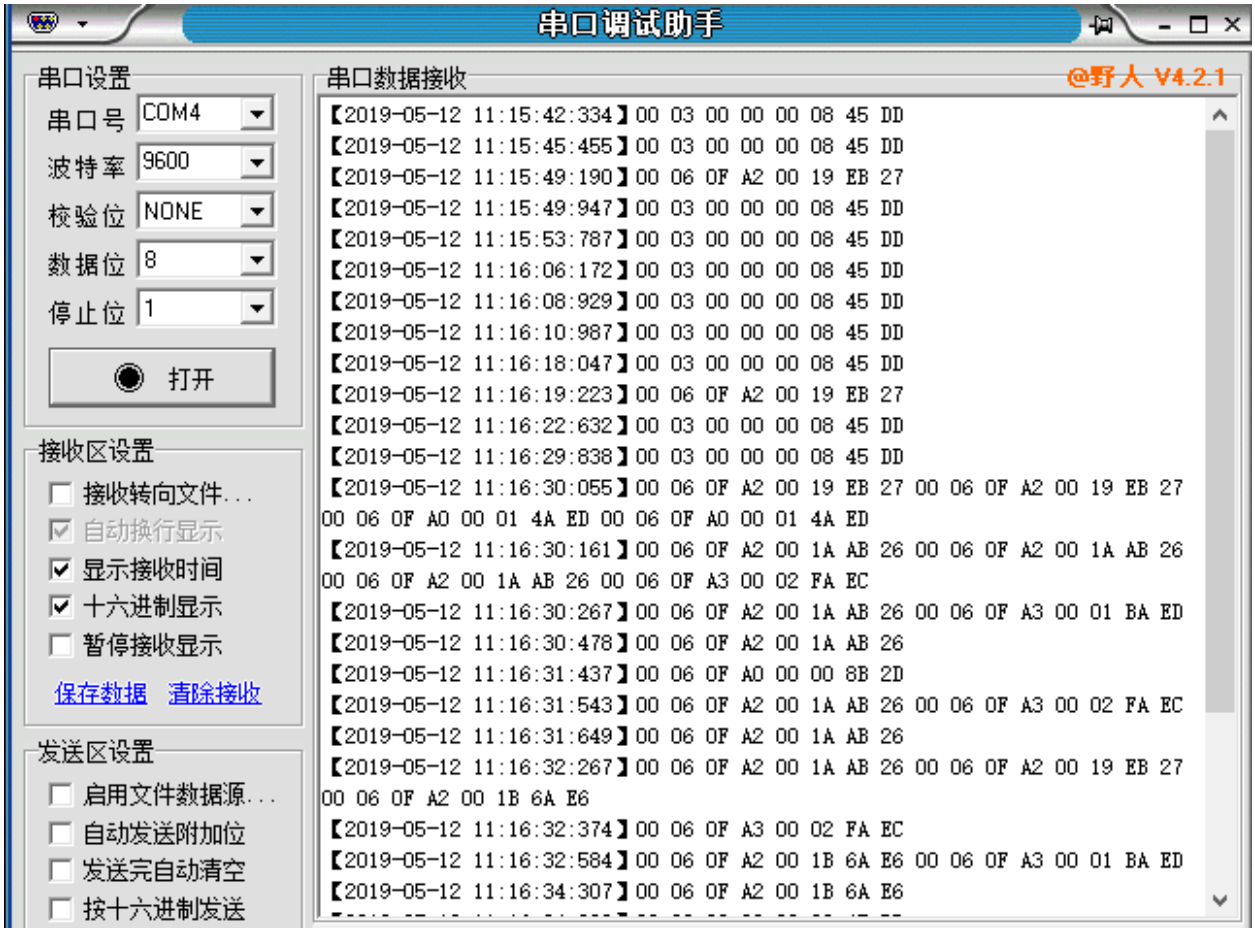

图 5-5 串口接收数据指令界面

结　论

本次毕业设计是主要以硬件电路设计为主的智能家居协议转换系统，系统的主要功能为通过无线传输方式，实现 zigbee 与 485 协议之间的相互转换。主要工作内容是完成系统硬件电路原理图、PCB 图纸设计，手工焊接电子元器件，主要焊接贴片和插件类电子元件，并使用示波器和万用表调试电路板判断输出电压、电流等参数与理论数值是否相符合；在软件程序设计方面，使用 C 语言编写基于 zigbee 协议栈的 zigbee 收发模块和 485 收发模块程序代码，并完成搭建系统整体环境，使用电脑端串口工具代替中央空调、地暖等大型家电设备进行调试演示。

硬件采用基于 32 位 ARM 系列 Cortex-M4 内核的 efr32 单片机，使用 zigbee 天线模块，使用两款稳压芯片对电源进行稳压设计，且在芯片的选型上，经过大量测试，最终确定最合适的方案。本系统设计解决了现场施工存在布线困难，而且施工周期长，难以维护，成本高，有些环境甚至无法布线等诸多缺点。

软件使用 silicomlab 平台进行开发，其自带的 zigbee 协议栈使得在编写程序代码更加方便高效，为软件开发人员减少了编程工作量。且系统自带的协议栈程序编写规范、条理清晰，便于后续系统调试与调用。

本系统对智能家居市场具有良好的应用意义，相信能够使各地产技术人员在试验系统过程中，感到便利，方便智能家居产品的对接，使智能家居产品连接更加高效和安全。

参考文献

[1] 王青.单片机与 PC 机通信的设计与仿真[J].实验室研究与探索,2016,35(12):100-103.

[2] 赵开理.基于单片机设计直流电机控制系统[D].南京:南京邮电大学电子与通信工程专业,2017.

[3] 党鹏,马菁.物联网在智能家居中的应用与发展[J].计算机产品与流通,2019(04):120.

[4] 周阳,周美娇,黄波,等.基于 C#的串口通信系统的研究与设计[J].电子测量技术,2015,38(07):135-140.

[5] 邓顺华.基于 zigbee 技术的智能家居中电器控制的研究[D].重庆：重庆理工大学电子信息工程专业,2016.

[6] 董泽龙,毋茂盛.SPI/UART 与 zigbee 协议转换模块设计[J].物联网技术,2015,5(12):32-34.

[7] 简单易用的 RS-232、RS-485 与 RS-422 转换方案[J].世界电子元器件,2019(02):17-20.

[8] 何智勇,徐丽萍.基于 Z-Stack 的 zigbee 协议栈组网过程研究[J].南京工业职业技术学院学报,2018,18(01):1-3.

[9] 马静.基于 CC2530 智能家居系统的 zigbee 协议栈的设计与实现[J].科技创新与生产力,2017(04):76-78.

[10] 张晓桃.基于 zigbee 无线数据传输系统的设计与实现[D].昆明昆明理工大学信息工程与自动化系,2017.

[11] Ravinder Yadav,Aravind Kilaru,Devesh Kumar Srivastava.Performance Evaluation of Word Count Program Using C#,Java and Hadoop[M].Berlin:Springer.2016,(02):63-91.