UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CURITIBA
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

AMANDA LÚCIA CARSTENS RAMOS
JOSÉ EDUARDO LIMA DOS SANTOS

SISTEMA INTEGRADO DE AUTOMAÇÃO RESIDENCIAL COM COMUNICAÇÃO SEM FIO

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2015

AMANDA LÚCIA CARSTENS RAMOS
JOSÉ EDUARDO LIMA DOS SANTOS

SISTEMA INTEGRADO DE AUTOMAÇÃO RESIDENCIAL COM COMUNICAÇÃO SEM FIO

Trabalho de Conclusão de Curso de Graduação, apresentado ao Curso de Engenharia de Controle e Automação, do Departamento Acadêmico de Eletrotécnica, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Engenheiro.
Orientador: Prof. Me. Guilherme Luiz Moritz
Coorientador: Eng. Mario Pires Ramos

CURITIBA
2015

Amanda Lúcia Carstens Ramos
José Eduardo Lima dos Santos

Sistema Integrado de Automação Residencial com Comunicação sem Fio

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro de Controle e Automação, do curso de Engenharia de Controle e Automação do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 10 de fevereiro de 2015.

_____
Prof. Paulo Sergio Walênia, Esp.
Coordenador de Curso
Engenharia de Controle e Automação

_____
Prof. Marcelo de Oliveira Rosa, Dr.
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia de Controle e Automação do DAELT

ORIENTAÇÃO

_____
Guilherme Luiz Moritz, Me.
Universidade Tecnológica Federal do Paraná
Orientador

_____
Mario Pires Ramos
Ericsson Telecomunicações SA.
Co-Orientador

BANCA EXAMINADORA

_____
Guilherme Luiz Moritz, Me.
Universidade Tecnológica Federal do Paraná

_____
Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná

_____
Winderson Eugenio dos Santos, Dr.
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia de Controle e Automação.

Aos nossos pais, irmãos e a toda nossa família que, com muito carinho e apoio, não mediram esforços para que nós chegássemos até esta etapa de nossas vidas.

AGRADECIMENTOS

Aos professores que de alguma forma contribuíram para nossa formação com exemplos e contraexemplos. Pela dedicação, orientação e apoio.

Ao nosso orientador, pelo empenho dedicado a este trabalho, pelas mensagens trocadas em período de férias e pelo relacionamento construído ao longo dos anos.

Ao nosso coorientador, pelos sábados e tardes doadas ao nosso projeto e pelas orientações, não só profissionais como pessoais.

Aos nossos pais, pelo amor, incentivo e apoio incondicional. E por proporcionarem condições que nos permitiram realizar essa conquista.

Aos nossos amigos que incentivaram o projeto e comemoraram essa conquista em conjunto. Em especial aos nossos colegas Milton pelas inúmeras revisões e Anderson pela ajuda com a programação Java, ambos se dispuseram a contribuir o máximo possível.

A todos que direta ou indiretamente fizeram parte da nossa formação e torceram pelo nosso sucesso, o nosso muito obrigado.

RESUMO

RAMOS, Amanda L. C. SANTOS, José Eduardo L. dos. **Sistema integrado de automação residencial com comunicação sem fio.** 2015. 67 f. Trabalho de Conclusão de Curso (Curso de Engenharia de Controle e Automação), Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

Este trabalho apresenta uma introdução aos protocolos predominantemente utilizados em automação residencial: Wifi e 802.15.4, assim como toda a comunicação necessária para a realização de projeto em automação residencial, utilizando como exemplo de módulo um sistema de iluminação. Foram utilizados rádios CC2530 da Texas Instruments para a rede 802.15.4, e para se comunicar com os microcontroladores o usuário utiliza um celular ou *tablet* com sistema operacional Android. A comunicação é feita através de um servidor em Java utilizando métodos de comunicação como UART (Servidor-Coordenador) e *Socket* (Servidor-Android). O objetivo principal do projeto é a comunicação entre os dispositivos, de maneira que seja fácil a integração de outros módulos posteriores, sendo estes configurados e controlados pelo usuário. O sistema também oferece a possibilidade de acesso remoto aos parâmetros, se o usuário estiver conectado no aplicativo compatível, que também foi desenvolvido em Java.

Palavras chave: Automação residencial. Comunicação sem fio. 802.15.4. Wifi. Sistema de iluminação.

ABSTRACT

RAMOS, Amanda L. C. SANTOS, José Eduardo L. dos. **Integrated Domotic System with Wireless Communication. .** 2015. 67 f. Trabalho de Conclusão de Curso (Curso de Engenharia de Controle e Automação), Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

This paper presents an introduction to the protocols predominantly used in domotic: Wifi and ZigBee, as well as all communication necessary to perform design in domotic, using as example a lighting system. It was used Texas Instruments radios CC2530, for the 802.15.4 network, to communicate with this radios the user uses a mobile phone or a tablet with Android operating system. A Java server is used to communcate with the mobile phone (via sockets) and with the radios (via UART). The main objective of the project is the communication between devices, so that it is simple to integrate other modules that could be configured and controlled by the user. This system will also offer the possibility of remote access to the parameters if the user is logged in a compatible application, which was also developed in Java.

Keywords: Domotic. Wireless communication. 802.15.4. Wifi. Lighting system.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

3G	Terceira Geração de Padrões e Tecnologias de Telefonia Móvel
AP	*Access Point*
APP	Aplicativo
AURESIDE	Associação Brasileira de Automação Residencial
BSS	*Basic Service Set*
CD	*Compact Disc*
DS	*Distribution System*
DVD	*Digital Versatile Disc*
ESS	*Extended Service Set*
FFD	*Full-Function Device*
IEEE	Institute of Electrical and Electronics Engineers
IP	*Internet Protocol*
IR	*Infrared*
LCD	*Liquid Crystal Display*
LED	*Light Emitting Diode*
MAC	*Medium Access Control*
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
PWM	*Pulse Width Modulation*
RFD	*Reduced Function Device*
SoC	*System on a Chip*
STA	*Station*
UART	*Universal Asynchronous Receiver/Transmitter*
TCP	*Transmission Control Protocol*
WLAN	*Wireless Local Area Network*
WPAN	*Wireless Personal Area Network*

SUMÁRIO

1 INTRODUÇÃO

1.1 TEMA

Com o passar dos anos o homem começou a criar e desenvolver tecnologias que melhorassem sua qualidade de vida. Sistemas que antes eram usados apenas em ambientes industriais passaram a fazer parte das residências também. Algumas atividades que fazem parte do dia a dia, que muitas vezes passam despercebidas, agora podem ser feitas através de sistemas automatizados. Estes proporcionam, segundo Silva (2009, p.12) maior segurança e economia, além de tempo livre para realizar outras atividades.

Com o avanço da automação para ambientes residenciais surgiu o termo Domótica, que vem da fusão da palavra latina *domus*, que significa casa, e da robótica. A domótica é um domínio da automação que tem como proposta a melhoria da qualidade de vida, o bem-estar e a redução dos afazeres domésticos repetitivos. Segundo Vecchi e Ogata (2013), a domótica procura uma melhor integração nas áreas de segurança, de comunicação, controle e gestão de fluídos.

Uma das principais preocupações dos projetistas de automação residencial, segundo a Associação Brasileira de Automação Residencial (AURESIDE), deve ser a integração entre os diversos sistemas de controle envolvidos. Outra precaução é a criação de uma interface intuitiva para o usuário, apesar de serem sistemas com funções cada vez mais complexas.

1.2 DELIMITAÇÃO DO TEMA

Tendo em vista que o tema é amplo, foram aplicados os conhecimentos dos autores desta proposta através de um módulo específico dentro da domótica. Esse sistema foi implementado utilizando-se comunicação exclusivamente sem fio de forma a tentar aproveitar ao máximo os sistemas de roteamento já existentes nas residências, sistemas esses mais difundidos e que facilitariam a implementação de uma automação integrada sem a necessidade de fazer modificações estruturais nas residências. O sistema proposto é um módulo de iluminação.

1.3 PROBLEMA

O caso de estudo foi realizado em uma residência que possui uma limitação quanto à instalação de plantas de automação residencial por ser uma construção antiga. O objetivo do estudo é projetar um sistema de automação residencial com comunicação sem fio, e implementá-lo. Portanto, como uma das vantagens do projeto proposto é a facilidade de implantação em casas que já estão em uso. Tomou-se como premissa a existência de um roteador que execute o protocolo que será utilizado.

Alguns dos desafios previstos pelos autores do projeto são:

- Complexidade no desenvolvimento da interface entre o usuário e o sistema, pois foi necessário o desenvolvimento de um aplicativo para celular ou *tablet* que realize essa comunicação;
- Obter novos conhecimentos não adquiridos durante o curso para o desenvolvimento do protocolo de comunicação entre os dispositivos;
- A escolha do microcontrolador adequado para as plantas controladas que atenda às necessidades de processamento e quantidade de portas E/S a um preço acessível;
- A integração do sistema como um todo, de forma que o usuário consiga, através um aplicativo, interagir com os sistemas de controle e comando.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Apresentar um sistema modular integrado de automação residencial, no qual além dos sistemas aqui apresentados possa ser aplicado a outros periféricos posteriormente. Este sistema deve ser de flexível e fácil instalação para ser utilizado em residências já estruturadas.

1.4.2 Objetivos Específicos

Seguem de forma pormenorizada os objetivos que se pretende alcançar.

- Revisar a teoria sobre os protocolos ZigBee e Wifi;
- Escolher o controlador e protocolo a ser utilizado;
- Desenvolver um sistema integrado baseado em uma planta de luminosidade;
- Elaborar o *hardware* através de um microcontrolador, especificado na seção de requisitos do projeto;
- Desenvolver um programa que integre o módulo da planta de iluminação e o usuário, com base em sistemas operacionais como MS-Windows e Android.

1.5 JUSTIFICATIVA

Atualmente vive-se na era digital, com a dependência de equipamentos eletrônicos aumentando enormemente no dia-a-dia da população. Segundo a Ericsson (2011), espera-se que até 2020, 50 bilhões de dispositivos estejam conectados às redes, como dispositivos eletrônicos, máquinas industriais, eletrodomésticos, sensores, atuadores e etc. A Figura 1 apresenta um gráfico indicando o número de dispositivos conectados à rede de acordo com o tempo. Estes sistemas são utilizados para facilitar os procedimentos e otimizar o tempo nas indústrias, para aumentar a confiabilidade e a precisão.

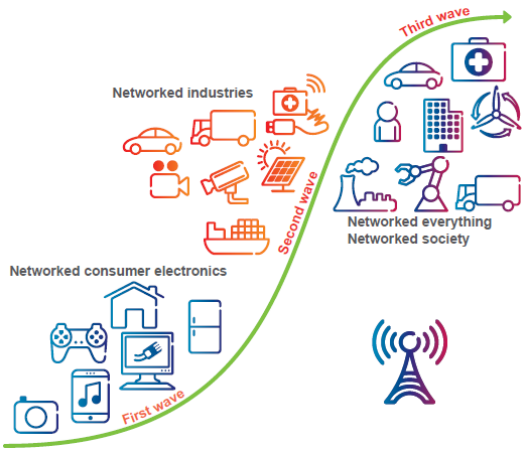


Figura 1 - Dispositivos conectados na rede
Fonte: Ericsson White Paper (2011)

A automação residencial tem se tornado um tema cada vez mais comum, espera-se para os próximos anos um aumento na disponibilidade de dispositivos conectados à rede que facilitem as atividades residenciais diárias e também em outras áreas como monitoração de sinais vitais de pessoas com saúde debilitada, reduzindo o tempo de início de atendimento aos mesmos.

Estes dispositivos deverão estar integrados à rede doméstica e terem acesso à *internet* com os devidos cuidados de segurança necessários. Ainda são poucos os que têm acesso a essa tecnologia, mas espera-se que aumente a acessibilidade com o tempo, sendo portanto uma das justificativas do projeto a criação de um sistema integrado com preço acessível. Através de pesquisas feitas recentemente, percebe-se que a tecnologia usada hoje no Brasil é importada, fazendo com que esse recurso apresente preços mais elevados.

De acordo com Amorim et al. (2014), no Brasil, as empresas e indústrias buscam importar tecnologia e componentes eletrônicos, pois acreditam que em lugares como EUA, Canadá e União Europeia além de universidades e institutos serem mais reconhecidos, há mais incentivo à pesquisa e infraestrutura.

Outra variável importante na proposta é a utilização de redes sem fio, pois os produtos existentes no mercado hoje são, em sua grande maioria, sistemas feitos com comunicação por cabos ou utilizando protocolos de comunicação sem fio independentes da estrutura de rede já disponível em grande parte das residências, como será demonstrado melhor na seção 3.1. A comunicação sem fio faz com que o sistema tenha a vantagem de poder ser instalado em residências já estruturadas, apesar de possuir alguns problemas com segurança, os quais devem ser solucionados.

1.6 PROCEDIMENTOS METODOLÓGICOS

Primeiramente, estudou-se os possíveis microcontroladores que poderiam ser utilizados no processamento dos comandos envolvidos, levando-se em consideração velocidade de processamento, número de entradas e saídas que serão necessárias, custo, complexidade de programação e plataformas disponíveis. Após escolhido o microcontrolador, foi estudado e selecionado um protocolo para a comunicação sem fio entre o servidor e os módulos, tendo em vista a simplicidade de implementação desse sistema nas residências. A estrutura prevista para o projeto pode ser observada na Figura 2.

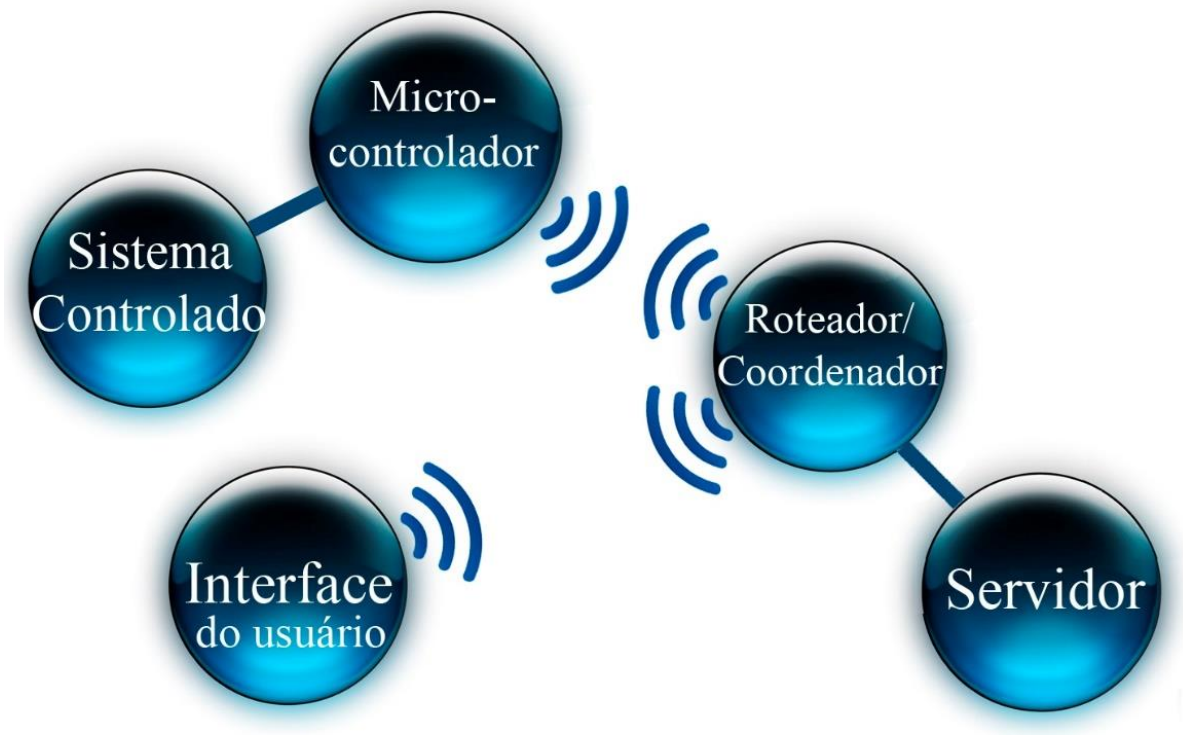


Figura 2 – Estrutura prevista para o projeto
Fonte: Autoria Própria.

O servidor faz a interligação entre a interface do usuário e o microcontrolador. O roteador/coordenador é o equipamento que recebe o sinal da interface ou do microcontrolador e o repassa para o servidor, onde é feita a interpretação do código recebido e gerado um novo código para envio em resposta. No microcontrolador que são feitas as operações de comando e controle, quando/se necessário.

Quanto ao módulo implementado, teve-se como ideia principal a realização de um sistema de automação utilizando como exemplo um único módulo, o de iluminação, com o objetivo de focar na comunicação entre os dispositivos e exemplificar através de um módulo básico.

A estrutura do projeto como um todo é detalhada no capítulo 3. E, finalmente, os sistemas propostos foram executados (capítulo 4) e integrados em uma residência e então apresentados à banca através de fotos e vídeos.

1.7 ESTRUTURA DO TRABALHO

O trabalho tem a estrutura abaixo apresentada.

Capítulo 1 – Introdução: são expostos o tema, as delimitações da pesquisa, o problema e a premissa, os objetivos da pesquisa, a justificativa, os procedimentos metodológicos e a estrutura geral do trabalho.

Capítulo 2 – Revisão Teórica: é apresentado um estudo teórico aprofundado dos protocolos existentes, dos prós e contras de cada um e dos microcontroladores. Assim como um estudo detalhado dos equipamentos já existentes no mercado.

Capítulo 3 – Especificações do Projeto: é abordada a descrição dos processos de funcionamento do sistema de automação residencial.

Capítulo 4 – Execução do Projeto: tendo como base as Especificações do Projeto, neste capítulo foi feito um protótipo funcional com o módulo proposto.

Capítulo 5 – Considerações Finais: As análises finais de todo o trabalho de conclusão de curso são feitas neste capítulo.

Capítulo 6 – Trabalhos Futuros: Neste capítulo são apresentados os possíveis trabalhos que poderiam dar continuidade ao trabalho iniciado.

2 REVISÃO TEÓRICA

2.1 PROTOCOLO ZIGBEE

O protocolo ZigBee integra o padrão IEEE 802.15 no subgrupo IEEE 802.15.4 – *WPAN Low Rate*. Este padrão é caracterizado por ter uma baixa taxa de transmissão, curto alcance, baixa potência, baixo custo e facilidade de implantação. Sendo que, para Siemeintcoski e Tobias (2008), uma das principais aplicações é a automação predial.

Outras aplicações podem ser observadas na Figura 3.

Figura 3 - Aplicações do ZigBee
Fonte: Adaptado de Siemeintcoski e Tobias (2008, Parte 1 p. 72)

Estão previstos três tipos de dispositivos em uma rede implementando o protocolo ZigBee, o coordenador, o roteador e o dispositivo final. Sendo que há duas categorias de dispositivos finais: os chamados FFD (Dispositivo de Função Completa) e RFD (Dispositivo de Função Reduzida), que se diferenciam em questões de complexidade e comunicação com os outros dispositivos. Por exemplo, interruptores precisam estar alertas por tempos mais longos e eles mesmos devem acionar os dispositivos que farão o controle, por essa razão para tanto é utilizado um dispositivo FFD, enquanto sensores, por outro lado, apenas recebem requisições para ter acesso aos dados e, então, desligam novamente, para tal aplicação são utilizados dispositivos RFD.

De acordo com Siemeintcoski e Tobias (2008) e Farahani (2008), em todos os processos são necessários pelo menos dois dispositivos: o coordenador e um dispositivo final. O coordenador é responsável pela transmissão de mensagens, ligar e desligar os outros dispositivos da rede, formar a rede, alocar endereços, entre outros. O dispositivo final é responsável por operações de baixo nível, como monitoração de variáveis, mudança de estado de entradas e saídas, etc.

O roteador ZigBee não é um dispositivo necessário à rede, mas apresenta características interessantes quando adicionado. Este aparelho é utilizado quando há a necessidade de aumentar o número de dispositivos conectados à rede, assim como quando é desejado ampliar o alcance físico da rede.

O dispositivo final possui um modo *sleep*, ou seja, quando não está em uso permanece adormecido, dessa maneira, economiza-se energia. Tanto o roteador quanto o coordenador não podem ‘dormir’ como os dispositivos finais, pois têm como compromisso repassar as mensagens entre os dispositivos.

Estes dispositivos podem ser configurados em diversas topologias diferentes, sendo que as quatro principais são: topologia par, estrela, árvore e em malha. A primeira, a mais simples, é formada apenas pelo coordenador e um dispositivo final ou roteador e é pouco utilizada. Na topologia em estrela, também bastante simples, há um único coordenador que recebe e envia as mensagens para todos os dispositivos finais. Nessa configuração, todas as mensagens passam pelo coordenador.

Na topologia em árvore são utilizados os três tipos de dispositivos, sendo que todas as mensagens devem passar ou por um roteador ou por um coordenador, porém os dispositivos finais não se comunicam entre si. Na topologia em malha são utilizados, também, todos os tipos de dispositivos, mas diferente da anterior os dispositivos finais de alto nível (FFD) podem se

comunicar entre si sem o intermédio de um roteador/coordenador. As quatro topologias podem ser visualizadas na Figura 4.

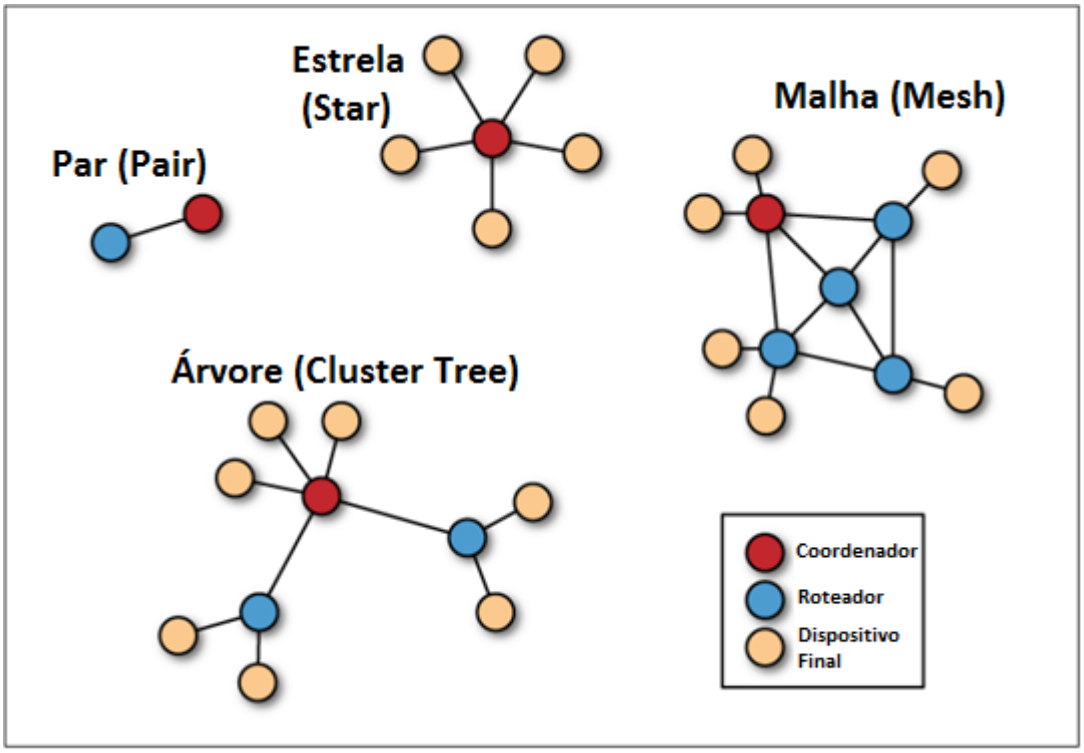


Figura 4 - Topologias ZigBee
Fonte: Adaptado de FALUDI, 2011.

2.1.1 Prós e Contras

As principais características que tornam o protocolo ZigBee uma ótima opção para sistemas de automação residencial são: facilidade na instalação, transferência de dados com confiança, operação com custo de implementação baixo, baixo consumo de energia, simplicidade de programação, baixo preço dos rádios e a operação em diversos modelos de topologias (ponto-a-ponto, estrela, árvore e malha).

Um dos possíveis contratempos é a baixa taxa de transferência (entre 20 Kbps e 250 Kbps), podendo se tornar um grande problema em sistemas que exijam uma maior transferência de dados e baixa latência, como por exemplo, transmissões de áudio e vídeo, pois o atraso na recepção e entrega de dados pode causar problemas na comunicação dos usuários pela latência gerada.

2.2 PROTOCOLO IEEE 802.11 (WIFI)

Segundo O'hara e Petrick (2004), o protocolo IEEE 802.11 foi implementado pela primeira vez em 1997 com o objetivo de substituir as redes cabeadas fornecendo o mesmo nível de segurança, taxa de dados e alto poder de processamento.

Este protocolo se encontra dentro do grupo de redes sem fio mais conhecidas, juntamente com os protocolos Bluetooth e ZigBee, os quais se diferenciam dentro das categorias WLAN/WPAN portanto, possuem taxas de dados, áreas de cobertura e aplicações distintos.

Ainda de acordo com O'hara e Petrick (2004), o padrão IEEE 802.11 define especificações das camadas físicas e de acesso ao meio (MAC) e a sua arquitetura é composta de vários elementos como: o cliente ou estação (STA), o ponto de acesso (AP), o conjunto básico de serviços (BSS), o sistema de distribuição (DS) e o conjunto estendido de serviços (ESS).

Os pontos de acesso, também conhecidos como nós, são aqueles que coordenam a comunicação entre os clientes da rede dentro da célula de comunicação de rede sem fio (BSS). O sistema de distribuição faz a interligação e comunicação entre os pontos de acesso, enquanto o conjunto estendido de serviços envolve vários conjuntos básicos de serviço, possibilitando os clientes se locomoverem por onde passam duas BSS diferentes com pontos de acesso conectados sem perder a conexão com a rede. Os componentes e suas disposições podem ser observados na Figura 5.

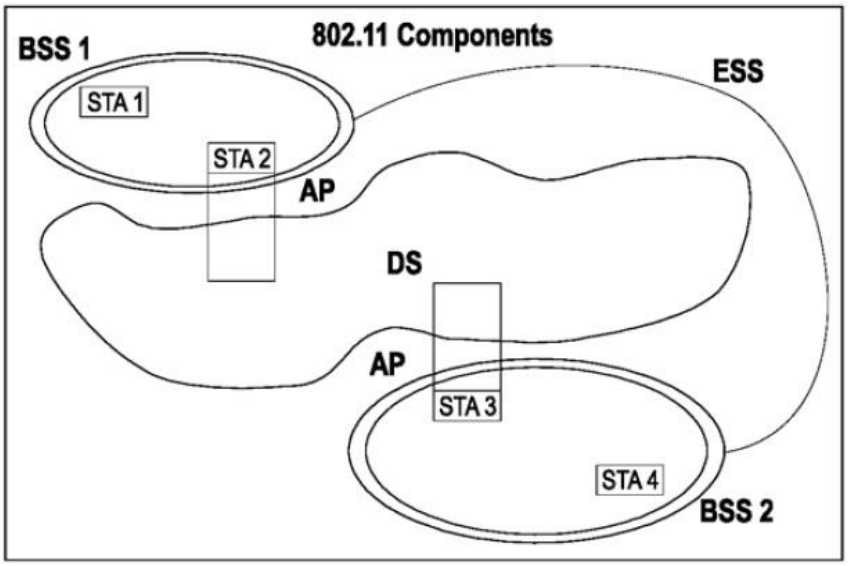


Figura 5 - Componentes da rede com protocolo IEEE 802.11
Fonte: IEEE 802.11™ WIRELESS LOCAL AREA NETWORKS

O Wifi é um conjunto de especificações para redes locais sem fio com base no padrão IEEE 802.11. Sua primeira versão foi disponibilizada em 1999 como IEEE 802.11b e posteriormente foram lançadas várias versões aprimorando a ‘original’ como as IEEE 802.11g, IEEE 802.11n e a mais atual e ainda em desenvolvimento 802.11ac. Todas as novas versões são compatíveis com as anteriores, se diferenciando apenas na taxa e no modo de transmissão.

Tabela 1 - Diferenças entre versões do protocolo IEEE 802.11

Versão do Protocolo IEEE 802.11	*Taxa de Transmissão (Mbps)*	*Modo de Transmissão*	*Frequência utilizada (GHz)*	*Área de Alcance Máxima (m)*
IEEE 802.11 b	1 – 11	DSSS	2,4	100
IEEE 802.11 g	1 – 54	OFDM | DSSS	2,4	140
IEEE 802.11 n	150 – 600	MIMO-OFDM	2,4 e 5	250
IEEE 802.11 ac	433 – 6000	MU-MIMO	5	50 (*indoor*)

Fonte: www.ieee802.org/11

2.2.1 Prós e Contras

Apesar de ainda ter um custo intermediário de implantação e serviço, uma das principais vantagens da utilização de redes Wifi, além da alta taxa de transmissão necessária em certas aplicações, é que já possuem uma ampla estrutura instalada.

Entretanto, o número de equipamentos utilizando a mesma faixa de frequência (2,4 GHz) influencia diretamente no funcionamento da sua própria rede, como as redes vizinhas, fornos micro-ondas e telefones sem fio, já que a interferência e a taxa de ocupação são altas. Outro ponto negativo é o consumo energético dos módulos, pois os equipamentos precisam estar alertas o tempo todo para realizar a troca de informações, fazendo com que, quando comparado a outros protocolos como 802.15.4, o gasto com energia seja consideravelmente alto. Ainda comparando com o protocolo 802.15.4, o custo e a complexidade dos rádios Wifi também são mais altos.

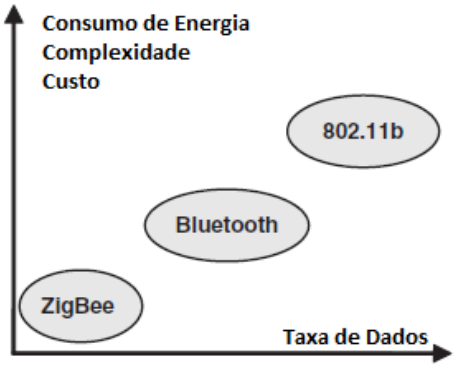
2.3 ESCOLHA DO PROTOCOLO UTILIZADO

Optou-se pela aplicação dos dois protocolos mencionados. Para ganhar em velocidade e economia de energia nos processos que não exigem alta taxa de dados, como os comandos de iluminação, foi utilizado o protocolo ZigBee.

Pela Figura 6, é evidente que a utilização do protocolo ZigBee é a melhor escolha em sistemas que exigem uma taxa de dados menor que 1Mbps e um grande alcance (entre 10 e 100m). Este protocolo possui uma taxa de dados maior que a do Bluetooth, possibilitando maior velocidade, e um alcance equivalente ao do padrão IEEE 802.11b (Wifi).

Além dos requisitos apresentados, o protocolo ZigBee oferece baixo consumo de energia, baixa complexidade de programação e implantação e baixo custo dos rádios. Já o protocolo Wifi encontra-se no outro extremo destas características entre os protocolos comparados, ainda na Figura 6. Porém, este é utilizado no projeto com o intuito de possibilitar o acesso remoto.

É evidente que a utilização do protocolo Wifi se faz necessária quando se deseja aproveitar o celular ou o *tablet* para a interface de comando, pois assim não é preciso uma modificação destes, visto que estes dispositivos já se comunicam por esse protocolo. Se fosse uma rede inteiramente ZigBee, seria inevitável adicionar, de alguma maneira, um rádio compatível ao celular/*tablet*.



	Taxa de Dados	Alcance Típico	Exemplos de Aplicação
ZigBee	20 to 250 Kbps	10–100 m	Redes de Sensores sem Fio
Bluetooth	1 to 3 Mbps	2–10 m	Fones sem Fio Mouse sem Fio
IEEE 802.11b	1 to 11 Mbps	30–100 m	Conecção com Internet sem Fio

Figura 6 - Comparando o protocolo ZigBee com Bluetooth e IEEE 802.11b
Fonte: Adaptado de FARAHANI (2008, p. 24)

3 ESPECIFICAÇÕES DO PROJETO

As especificações do projeto são disponibilizadas em três níveis de acordo com a teoria de engenharia de sistemas, teoria utilizada para explicar de maneira clara o projeto a ser desenvolvido. O primeiro nível, Concepção do Sistema, é descrito de maneira que qualquer pessoa, com conhecimentos técnicos ou não, consiga entender a proposta geral do projeto, assim como os objetivos finais e o estudo de mercado da área. No segundo, Requisitos do Sistema, é apresentada uma visão um pouco mais detalhada de cada projeto, resumindo a concepção do projeto em requisitos de funcionalidade. Enquanto no terceiro, Requisitos de Projeto, o projeto é descrito minuciosamente e são especificadas as funcionalidades características que implementam os requisitos do sistema, para esta seção é exigido um conhecimento técnico mínimo.

Este modelo, divido em três níveis, é amplamente utilizado para projetos em engenharia de sistemas, porém cada uma dessas etapas seriam um documento a parte. Para este projeto em questão, por seu volume, foram resumidos em três seções separadas.

3.1 CONCEPÇÃO DO SISTEMA

O sistema de automação residencial tem como objetivos principais garantir ao usuário conforto, segurança, economia, comodidade, velocidade, flexibilidade e interatividade. O sistema aqui apresentado proporciona um único módulo de iluminação.

Desejou-se fazer um controle central através de um servidor que faz toda a interligação entre os microcontroladores espalhados pela casa e a interface do usuário. Essa comunicação é sem fio, com o objetivo de facilitar a implantação do sistema. Os microcontroladores tem a finalidade de gerenciar as entradas e saídas, assim como efetuar o controle, quando necessário, e o comando dos dispositivos. Sendo que, uma das principais metas é poder acoplar novos dispositivos de controle e comando quando desejado (sistema modular).

3.1.1 Estudo de Mercado

O atual mercado é caracterizado por grande volume de informação disponível e intensa globalização e como consequência o aumento da concorrência, causando o desenvolvimento de novas tecnologias por parte das empresas.

Segundo a AURESIDE, o mercado de automação residencial atravessa, atualmente, um momento de agitação caracterizado por crescimento físico de mercado, maior divulgação do tema e aumento da concorrência. Cada vez mais rapidamente são lançados novos produtos e solução no mercado, as tendências na área também estão em constante mudança. A associação estima que entre 2008 e 2010 houve um crescimento do número de projetos nesta área de 40% ao ano e que os preços finais dos produtos estão em queda a cada ano que passa.

As principais tendências desse mercado são a utilização de controles universais (como celulares e *tablets*), cuidados e monitoramento personalizado da saúde, uso de materiais sustentáveis e ambientalmente corretos, eficiência energética, entre outros. Segundo a AURESIDE, os sistemas dominantes nesse mercado são os de segurança, áudio e vídeo, iluminação e climatização.

No Brasil, as empresas que tem se especializado ou desejam entrar no mercado da automação residencial têm buscado soluções prontas, reconhecidas e que já estejam no mercado há algum tempo. A justificativa é simples, nenhuma delas está disposta a testar um novo produto que ofereça riscos ou que gere problemas de manutenção muito frequentes. Desta forma, foi feita uma breve pesquisa dos equipamentos utilizados atualmente por essas empresas.

Em geral os sistemas de automação residencial são bem parecidos, todos oferecem uma forma de controlar a casa com apenas um controle universal, que pode ser o próprio *smartphone*, *tablet* e até mesmo pelo computador. A estratégia para fazer isso é simples, por trás de todo o sistema de automação, há um computador ligado 24 horas por dia, que monitora, recebe e envia informações de todos os dispositivos conectados à rede. Esse computador mais conhecido como o servidor é o cérebro de todo o sistema.

Foram encontradas duas marcas que além de oferecerem o servidor também oferecem total suporte ao serviço, são elas a iSimplex e a Control4. A iSimplex oferece um servidor (Figura 7) que se comunica com dispositivos padronizados inicialmente, além disso oferece a alternativa de se integrar novos dispositivos de outras marcas para que possam ser também utilizados no sistema de automação. Oferece, ainda, uma interface muito intuitiva e moderna

para que o usuário não precise buscar um manual que lhe possibilite utilizar o sistema de automação.

O que foi percebido pelos autores do projeto durante as pesquisas é que as pessoas primeiramente buscam o entretenimento, como som ambiente, sala de cinema com projeção de imagem e etc. A iSimplex oferece também esse serviço integrado com o servidor, o *Media Center*, um dispositivo do tamanho de um gravador de DVD que tem uma interface própria para reprodução de vídeos, músicas e exibição de fotos. Neste dispositivo o usuário pode também inserir um CD ou DVD e fazer uma cópia digital que ficará armazenada em uma biblioteca digital também integrada com o servidor.

Figura 7 - Servidor da iSimplex
Fonte: iSimplex

Para a reprodução desse conteúdo é utilizado um *receiver*, um dispositivo capaz de concentrar várias entradas de vídeo para uma única saída de vídeo e áudio, normalmente ligado à um *Home Theater*. Essa parte da automação, de controle de dispositivos, utiliza um emissor de sinal infra vermelho, o qual também é conectado à rede e recebe comandos direto do servidor. O dispositivo encontrado no mercado foi o *iTach TCP/IP to IR* da Global Caché, Figura 8. A partir dele é possível fazer a gravação de todos os códigos infra vermelho dos controles remotos dos dispositivos a serem controlados e armazená-los no servidor. O Global Caché oferece 3 canais de saída para o controle de até 3 dispositivos. A essas saídas é ligado um cabo com emissor de infra vermelho que vai ligado diretamente ao receptor de cada um dos aparelhos a serem comandados. Através da interface do servidor o usuário consegue controlar todos os aparelhos da sala pelo *smartphone*, *tablet* ou computador e ainda configurar um mesmo botão para dar vários comandos ao mesmo tempo.

Figura 8 - iTach TCP/IP to IR da Global Caché
Fonte: Global Caché

Quanto ao controle de ambiente foi encontrado no mercado a Lutron, empresa especializada neste serviço. A Lutron oferece um sistema de comando de dispositivos sem fio, no qual uma central deve estar conectada ao servidor através da rede e é responsável por enviar os sinais aos dispositivos finais, como gerenciador de iluminação, sensores de presença, etc. Na parte da conexão com as lâmpadas há um gerenciador de iluminação instalado na parede com funções *On/Off* e dimmer para o controle das lâmpadas. Esse gerenciador recebe todos os retornos da instalação elétrica e comanda as lâmpadas de acordo com os sinais que ele recebe da central conectada ao servidor, dessa forma ainda há uma grande dependência de ajustes do cabeamento para que o sistema possa funcionar perfeitamente. Outro problema visto é que a topologia utilizada no sistema da Lutron é em estrela, o que limita a distância a que os dispositivos podem estar da central.

O gerenciador de iluminação também pode ser ligado a outros dispositivos finais. Um já utilizado e comercializado por eles é a cortina motorizada. Ela vai ligada da mesma forma que uma lâmpada ao gerenciador de iluminação mas com um cabo para a abertura e outro para o fechamento.

O sistema da Lutron pode ser integrado a qualquer servidor desde que seja ajustada a comunicação entre os dispositivos. Já existe essa integração com o servidor da iSimplex que é capaz de se comunicar com a central da Lutron para dar os comandos, dessa forma é gerada uma interface onde o usuário pode comandar as lâmpadas, cortinas ou configurar para que o sistema aja automaticamente de acordo com o horário do dia. Além de dar os comandos pela interface *mobile* do usuário a Lutron também disponibiliza um interruptor de pulsos que envia os comandos para a central.

O grande diferencial da Control4, é o fato de eles oferecerem todos os dispositivos apresentados anteriormente como produtos da própria empresa, ou seja, eles têm o seu próprio servidor e sistema operacional que integra todos os sistemas, tem o seu próprio sistema para o comando de luzes, tem um sistema para fazer a integração entre dispositivos já existentes na casa, usando principalmente comunicação serial e infravermelho, etc.

O projeto proposto é semelhante aos sistemas citados, uma vez que a proposta engloba o comando de iluminação tanto local quanto remotamente. Porém, apresenta uma maior simplicidade. De acordo com Eduardo Almeida *apud* Mourão (2014), "Os preços baixaram, a competência das empresas aumentou e as possibilidades só têm um limite: a nossa própria imaginação".

3.2 REQUISITOS DO SISTEMA

3.2.1 Planta de Iluminação

Com o objetivo de controlar a luminosidade do ambiente, o usuário poderá controlar o nível de luminosidade da lâmpada do cômodo através de um aplicativo de celular.

3.2.2 Integração

O projeto propõe ao usuário fazer a utilização de um serviço, inicialmente como o de iluminação, através de qualquer dispositivo portátil, como *smartphones* e *tablets* com sistema operacional Android. É utilizado um servidor localizado em um computador central, no qual são processadas as informações recebidas e enviadas para os microcontroladores disponíveis.

3.3 REQUISITOS DO PROJETO

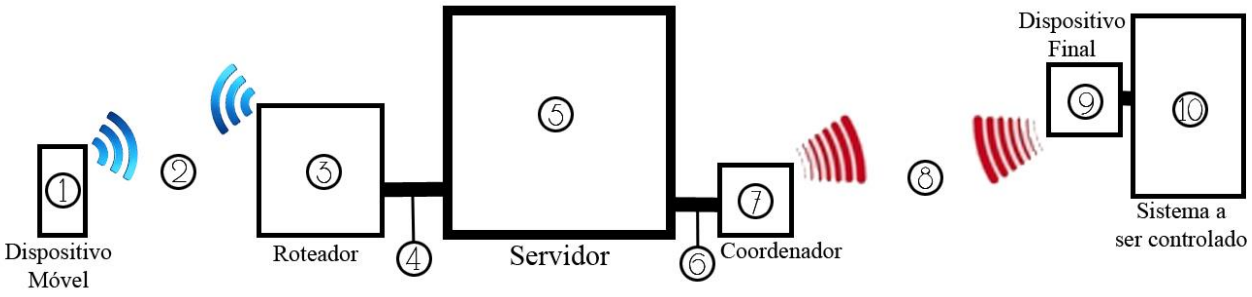


Figura 9 - Projeto detalhado
Fonte: Autoria Própria.

Com base na Figura 9, será detalhado o funcionamento do sistema a partir de cada item essencial a todos os controles envolvidos nesse projeto. Os itens estão numerados para melhor entendimento e facilitar a explicação.

O item 1, dispositivo móvel, representa um celular ou *tablet* com sistema operacional Android, no qual o usuário interage através da aplicação cliente com o próprio sistema. Através deste dispositivo o usuário pode alterar os parâmetros desejados e também verificar o estado atual dos dispositivos de automação. O aplicativo foi feito em Java e tem uma interação com o servidor (item 5) localizado no PC, a comunicação é realizada via Wifi (através do roteador (3)).

O item 5, Servidor, fica localizado em um computador central. O sistema operacional utilizado é o MS-Windows e a aplicação no servidor foi programada em Java para facilitar a integração com o aplicativo Android além de facilitar a portabilidade para outros sistemas operacionais. Foi desenvolvido um aplicativo no qual, além do servidor, consta um código para interpretação da mensagem enviada pelo dispositivo móvel e envio dos comandos desejados para o coordenador da rede ZigBee (item 7), os quais são direcionados para o dispositivo final requisitado (item 9).

Tanto o coordenador (7) quanto o dispositivo final (9) são microcontroladores CC2530 da Texas Instruments, este SoC (sistema em um chip) integra o meio físico IEEE ZigBee com um processador de uso geral, neste é possível implementar o aplicativo necessário para um sistema completo baseado na rede ZigBee.

Os itens 2, 4, 6 e 8 são referentes à comunicação entre os dispositivos principais. A interface entre o dispositivo móvel e o PC/servidor (itens 2 e 4) é realizada através de *sockets* TCP/IP sobre Wifi. Enquanto a comunicação entre o PC/servidor e o coordenador (item 6) é feito por UART, já que essa é a única comunicação que os dois dispositivos tem em comum. O item 8 apenas faz referência à comunicação sem fio ZigBee.

O item 10 representa o sistema que é comandado, no caso, a lâmpada. O item 9, dispositivo final da rede ZigBee, fica anexado ao sistema a ser controlado com um código de programação específico para o controle em questão.

3.3.1 Controle de Iluminação

Figura 10 - Sistema de iluminação
Fonte: Autoria Própria.

- O microcontrolador atua através de controle da largura de pulso no circuito de alimentação das lâmpadas encontradas no ambiente;
- O sistema mantém a luminosidade em um nível estipulado pelo usuário em porcentagem, sendo 100% o equivalente à 90% do máximo de iluminação disponível com as lâmpadas;
- Há duas possibilidades para o acionamento da iluminação no cômodo, por um botão de pulso com a função liga/desliga localizada no próprio cômodo (comando local) ou por um dispositivo móvel que possua um aplicativo compatível com o sistema (podendo ser um comando remoto).

3.3.2 Integração

- No computador roda um aplicativo em Java que representa o servidor do projeto;
- Há um computador central com sistema operacional MS-Windows, pois este é o mais comum em computadores domésticos;
- No servidor são feitas as interpretações das requisições e a geração de novas requisições;

- Há uma comunicação entre o servidor e o coordenador para permitir a transmissão dos comandos para os dispositivos finais;
- No dispositivo móvel há um aplicativo Java que serve de interface para acionamento dos dispositivos;
- Foram desenvolvidos aplicativos em linguagem C, os quais são executados pelos microcontroladores.
- Há a possibilidade de acesso remoto aos parâmetros desejados se o usuário estiver conectado no aplicativo cliente.

4 EXECUÇÃO DO PROJETO

Para a execução do projeto, primeiramente, foi realizado um simples planejamento para auxiliar no desenvolvimento do mesmo e no cronograma. Dividiu-se o projeto em cinco grandes grupos, cada qual com suas subdivisões demonstrados na Figura 11.

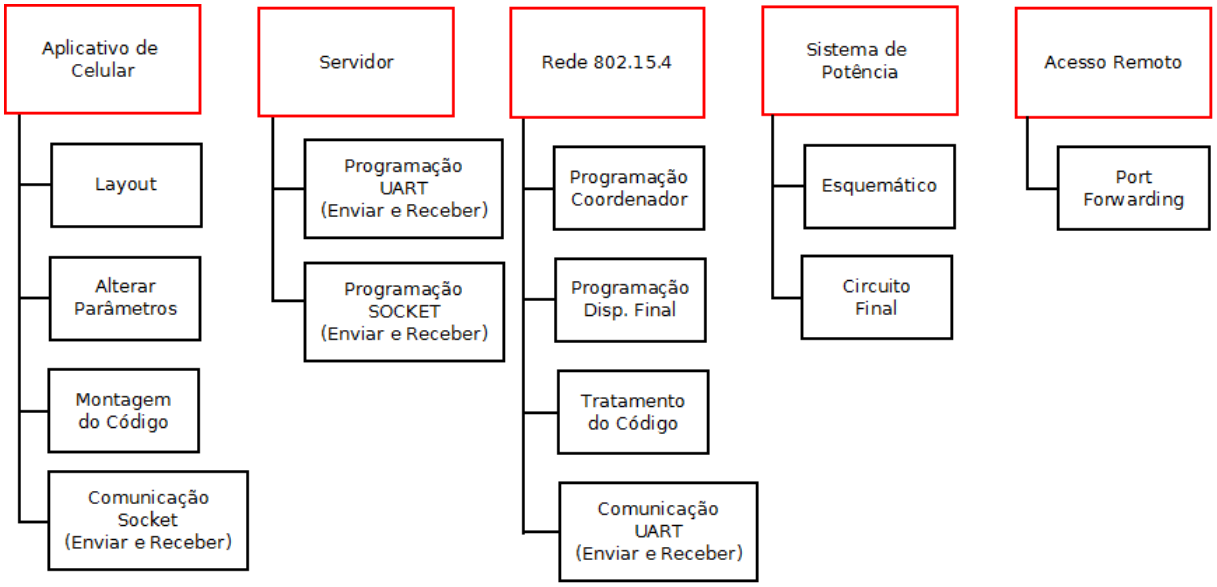


Figura 11 – Diagrama de blocos dos cinco principais grupos a serem trabalhados
Fonte: Autoria Própria.

O projeto começa inicialmente com o aplicativo de celular a partir do qual o usuário pode interagir com o sistema, modificando os parâmetros desejados em cada componente disponível. O aplicativo se comunica com o servidor para o mesmo interpretar o código enviado pelo celular e enviar o comando correto para o coordenador da rede 802.15.4. O mesmo irá receber o código e interpretá-lo a fim de redirecionar o comando para o dispositivo final correto. Por fim, o dispositivo final receberá o comando e deverá acionar uma lâmpada em um sistema de potência a partir de um sinal PWM (modulação por largura de pulso). Além de fazer esse comando via rede 802.11 local, também há disponível o acesso remoto da rede. Com o intuito de estabelecer os objetos que interagem e seus relacionamentos dentro do projeto, há um diagrama de sequências do projeto, o qual está demonstrado de forma simplificada na Figura 12.

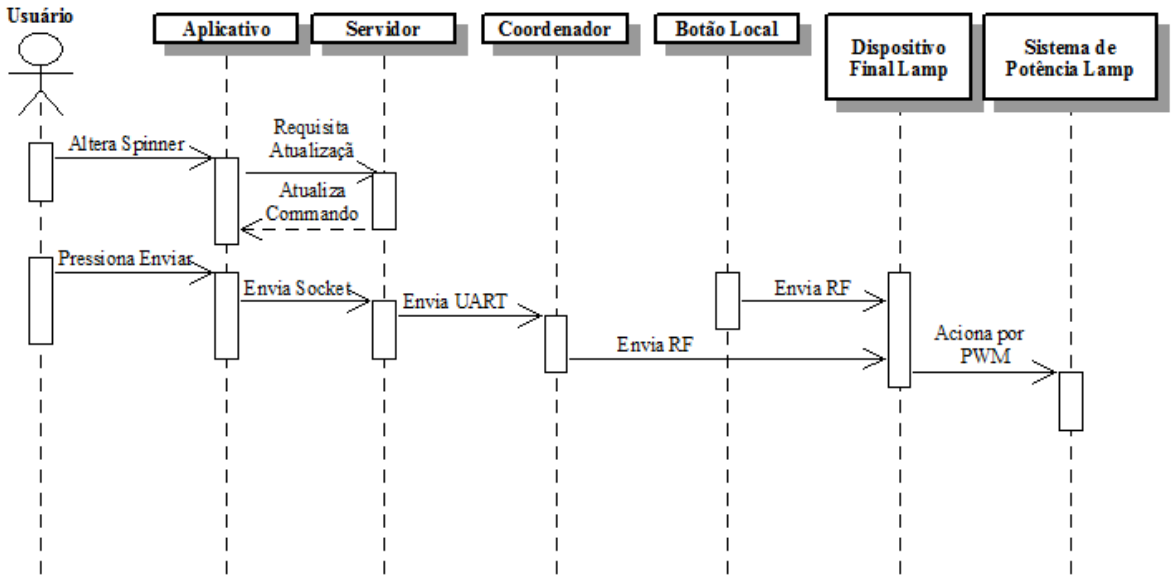


Figura 12 - Diagrama de sequência do projeto
Fonte: Autoria Própria.

Os cinco grandes grupos são explicados com mais detalhes a seguir.

4.1 APLICATIVO DE CELULAR

Essa entidade é utilizada para o usuário interagir com os dispositivos finais da rede, através de um *layout* simplificado e intuitivo. No começo do projeto os autores não tinham conhecimento prévio sobre como programar um aplicativo e nem como fazê-lo se comunicar com o servidor. Para tanto, foi necessária uma extensiva busca por conhecimento prático em fóruns e vídeos disponíveis na *internet*. Optou-se por programar um aplicativo para a plataforma Android, por ser uma plataforma aberta. A programação do mesmo foi feita em linguagem Java utilizando o programa Android Studio, disponibilizado pela própria Google para desenvolvimento de aplicativos.

Para a concretização do aplicativo foi necessário aprofundar os conhecimentos em alguns elementos básicos do layout, que serão brevemente explicados a seguir.

- *Spinner*: é um menu que oferece diversas opções. Quando está inativo ocupa apenas uma linha da tela, e quando está ativo aparecem todas as opções permitindo o usuário realizar a escolha.

- *Seekbar*: é uma barra de progresso. O usuário pode tocar e arrastar para os lados para definir o nível desejado.
- Botões: São botões comuns que ao serem acionados chamam funções específicas.
- Textos: Os textos podem ser fixos ou variáveis. Ou seja, receber um valor inicial e não ser modificado durante o andamento do aplicativo ou então, receber um valor de acordo com uma variável e ser modificado de acordo com a mesma.

Com o intuito de manter o aplicativo simples e funcional, idealizou-se o mesmo com apenas duas telas (Figura 13). Na tela principal tem informações como o nome dos alunos e da universidade e botões para as telas secundárias: Alterar Parâmetro e Adicionar Equipamento, além do botão para sair do aplicativo.

Na segunda tela tem os parâmetros a serem definidos pelo usuário para alteração de algum estado. Existem as opções: escolha do cômodo e escolha do componente em forma de *spinner* e a ação referente ao nível a ser configurado para tal componente em tal cômodo em forma de *seekbar*. Além disso, haveriam dois botões, um para enviar o código de mudança do parâmetro e outro para voltar a tela inicial.

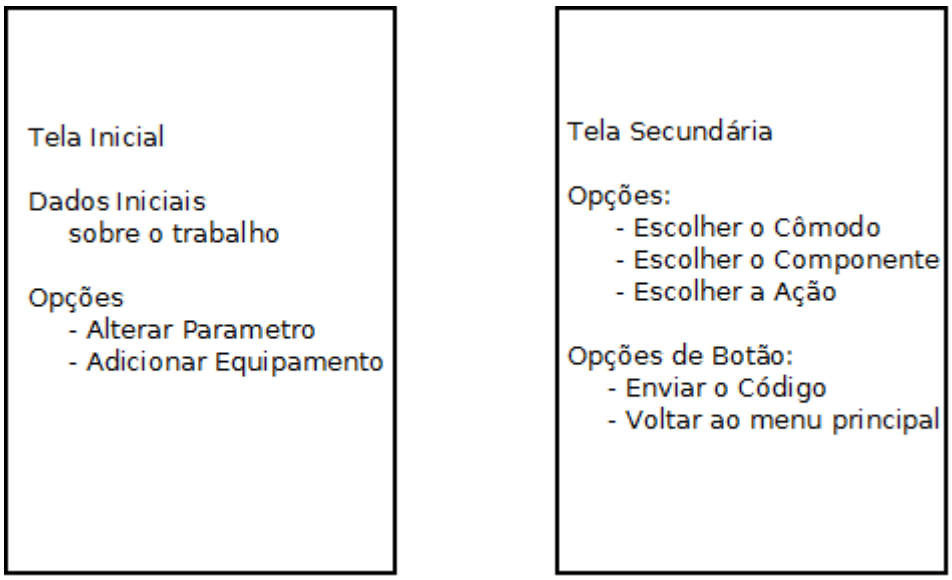


Figura 13 - Idealização do Aplicativo
Fonte: Autoria Própria.

O *layout* final do aplicativo pode ser visto na Figura 14 e alguns métodos do código comentado principal para a realização do mesmo encontram-se no apêndice.

Figura 14 - *Layout* final do aplicatvo
Fonte: Autoria Própria.

Ao enviar a ação desejada, o servidor monta um código que será enviado via comunicação UART para a rede 802.15.4. O código montado segue a seguinte lógica, onde os 'X' representam os números do correspondente cômodo ou componente.

Tabela 2 – Padronização dos códigos - Parte 1

Cômodo	***Código***	***Componente***	***Código***
Quarto	QXX	***Lâmpada***	LXX
Sala	SXX	***Ar Condicionado***	ARX

Fonte: Autoria Própria.

Juntamente com a seleção do cômodo e componente, o usuário deve indicar uma ação a ser realizada. Para as lâmpadas, deve escolher um nível de luminosidade em porcentagem e para o ar condicionado, deve escolher uma temperatura desejada entre 18 e 25 graus Celsius. Para a finalização do código, essa divisão foi chamada de Status e é montada da seguinte

maneira, onde YY representa a porcentagem desejada para o caso das lâmpadas (00 – 0% até ON – 100%) e ZZ a temperatura estimada.

Tabela 3 – Padronização dos códigos - Parte 2

Componente	*Código*
Lâmpada	SYY
Ar Condicionado	SZZ

Fonte: Autoria Própria.

Por fim, é necessário que o aplicativo se comunique com o servidor para o código poder ser utilizado pela rede 802.15.4. Essa comunicação foi feita via *sockets* TCP/IP sobre Wifi, uma comunicação simples baseada no mecanismo cliente-servidor. No aplicativo de celular roda a aplicação cliente, que faz a requisição de uma conexão *socket* utilizando o endereço IP e a porta na qual o servidor está esperando conexões.

4.2 SERVIDOR

Este elemento é o que interliga o aplicativo de celular com a rede 802.15.4, pois ele consegue se comunicar tanto via *socket* com o celular quanto via UART com o coordenador da rede, enquanto os outros dispositivos do projeto possuem métodos de comunicação limitados. Dessa forma, necessitou-se a programação do servidor baseada nesses dois tipos de comunicação. Para tanto, foi utilizado o programa Netbeans IDE versão 8.0.1 e a programação foi feita em Java para facilitar a interação com o aplicativo de celular.

Segundo pesquisa dos autores haveriam duas maneiras bastante conhecidas para o servidor se comunicar com o aplicativo de celular para a atualização dos parâmetros atuais (servidor → APP) e envio do código (APP → servidor), sendo elas via *WebService* e via *socket*. Optou-se por utilizar a segunda pela facilidade de configuração e de programação, e ainda haver um retorno razoavelmente bom para a aplicação desejada.

Segundo Hopson *apud* Silva (2009) existem dois modos de operação deste tipo de comunicação, o modo baseado em conexão e o modo sem conexão. Os dois modos se

diferenciam basicamente pela garantia de recebimento dos dados e pela ordem de chegada dos dados enviados. Adotou-se o primeiro modo pela garantia que o mesmo proporciona. Assim, implementou-se o servidor *socket* no computador e o cliente *socket* no aplicativo de celular.

O programa servidor permanece escutando se há alguma conexão sendo requisitada na porta onde a comunicação *socket* foi aberta. Enquanto o aplicativo cliente faz a requisição apenas em momentos específicos em que a transição de dados é necessária.

Já a comunicação entre o servidor e a rede 802.15.4 para o envio do código (servidor → CC2530) foi feita via comunicação serial, tendo em vista que os dois dispositivos se comunicam com facilidade via UART. A escolha deste tipo de troca de informações também se dá pelo fato de os acadêmicos já terem utilizado este método durante o curso e pela disponibilidade de *hardware* para isso. Foi utilizada uma placa FTDI modelo FT232R a qual pode ser visualizada no anexo B.

Em relação à programação destas comunicações, encontrou-se poucos obstáculos, tendo em vista que a própria Oracle disponibiliza exemplos bases em seu *site*. Alguns métodos utilizados estão disponíveis no apêndice B.

4.3 REDE 802.15.4

No começo do desenvolvimento do projeto os estudantes tentaram utilizar o Protocolo ZigBee como protocolo de comunicação, o que gerou grandes dificuldades devido a sua complexidade e pouca informação acerca do funcionamento de suas funções. Dessa forma, os estudantes optaram por formar a rede interna do sistema de comando da iluminação utilizando o Protocolo 802.15.4, que é uma simplificação do protocolo ZigBee.

4.3.1 Roteamento dos pacotes recebidos pela rede 802.15.4

Após o envio do pacote do servidor pela UART, a rede 802.15.4 fica responsável pelo recebimento do pacote e por tomar as decisões se tudo que será feito a partir das informações recebidas. A rede é composta de microcontroladores CC2530 implementados em placa

impressa para o módulo com dois botões, dois LED's e pinos para comunicação serial (ver Anexo A).

O coordenador da rede é o dispositivo responsável por fazer comunicação direta com o servidor e todos os outros dispositivos ligados à rede. Toda vez que um pacote é recebido pela UART, o coordenador da rede deve interpretar o pacote que é composto por uma *string* com todas as informações referentes ao cômodo, objeto e valor, descobrindo dessa forma à qual dispositivo final o pacote será direcionado e qual informação de valor deverá ser aplicado no dispositivo que recebê-lo. Grande parte do esforço de desenvolvimento foi investido nesse subsistema devido a falta de exemplos e por grande parte do conteúdo disponível ser incompleto ou de certa forma complexo para o nosso entendimento. Dessa forma resolvemos utilizar um código base mais simples disponibilizado pela Texas Instruments. O qual era empregado para dar comandos de liga e desliga à uma lâmpada utilizando o padrão IEEE 802.15.4.

Nesse código foram, então, implementadas as funções para utilizar a comunicação serial que seria feita com o servidor e as funções que seriam responsáveis por interpretar a *string* recebida. Para o envio dos pacotes aos dispositivos finais foram utilizadas as funções de comunicação entre os dispositivos já disponíveis no código. Nessa função, um dos parâmetros é o tipo de comando que será interpretado pela lâmpada.

Dois dispositivos finais foram implementados na rede em questão, um para o comando de luminosidade e outro para agir como um botão com função de ligar e desligar a lâmpada, sendo a rede montada na topologia malha. O dispositivo lâmpada usa as mesmas funções do código base para recebimento de pacotes. Quando ele recebe um pacote, ele testa o primeiro *byte* do vetor de dados para saber qual o comando do pacote. Caso seja comando de intensidade, é recuperado o valor que foi definido pelo usuário no aplicativo e utiliza na função do PWM para comandar a lâmpada. Caso seja comando do botão, ele testa a variável de estado atual e realiza a ação contrária (ligar ou desligar).

O dispositivo final botão é o mais simples. Utiliza as mesmas funções que os demais para comunicação dentro da rede 802.15.4 e sempre que o botão é pressionado envia um pacote para a lâmpada com o comando BUTTON_CMD.

O microcontrolador escolhido para formar os dispositivos do sistema de automação residencial foi o CC2530 da Texas Instruments, devido a sua grande disponibilidade, capacidade de implementar o Protocolo ZigBee que seria utilizado inicialmente pela equipe, sua grande disponibilidade de pinos de entradas e saídas, comunicação UART, geração de sinal PWM, tamanho reduzido, baixo custo. No anexo B pode-se ver o módulo do microcontrolador utilizado.

4.4 ACESSO REMOTO

O acesso remoto é uma conexão através da rede entre um computador local e um computador remoto, no caso o servidor e o celular respectivamente. Essa comunicação tem como propósito geral efetuar uma tarefa como por exemplo a administração remota do sistema e é proposta pelos autores pela versatilidade que a mesma proporciona. No projeto em questão a ideia é o usuário ter a capacidade de alterar os parâmetros dos componentes da residência sem estar na mesma, através de outra rede de transmissão de dados (como Wifi ou 3G).

Foi necessário o estudo do roteador disponível aos autores (modelo CISCO DPC3925), a fim de concretizar essa ideia, e configurá-lo conforme será explicado a seguir. O acesso aos parâmetros é feito colocando o IP do roteador da rede local em um navegador de *internet*. Configurou-se o *port forwarding* para realizar o redirecionamento de portas da rede.

4.5 SISTEMA DE POTÊNCIA

Para o controle da intensidade luminosa da lâmpada, foi desenvolvido um pequeno sistema de potência composto por um retificador com filtro capacitivo que entrega uma tensão de aproximadamente 180 Volts (V). Em paralelo com o capacitor foi colocado a carga, uma lâmpada de 220 V, 40 Watts (W) de potência, em série com um MOSFET IRF740. Na parte de comando foi utilizado o CC2530 com um dos pinos capaz de gerar sinal PWM ligado ao Gate da chave, dessa forma a onda PWM gera chaveamento do IRF740 com razão cíclica proporcional ao valor de intensidade recebido por esse dispositivo. Na Figura 15 é apresentado o esquemático do circuito.

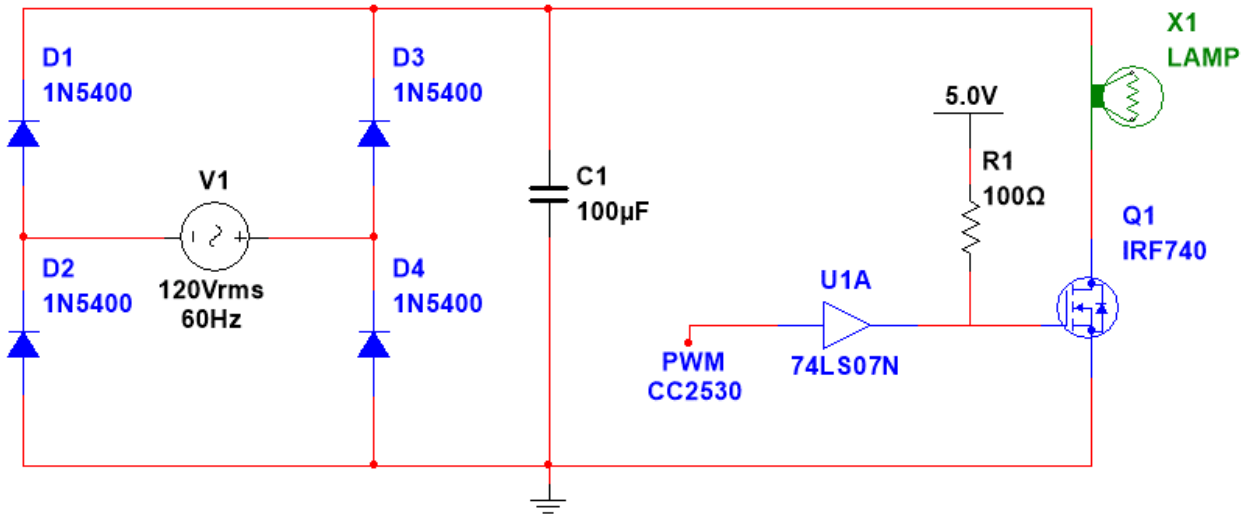


Figura 15 - Esquemático Circuito de Potência
Fonte: Autoria Própria.

O circuito final pode ser observado na Figura 16.

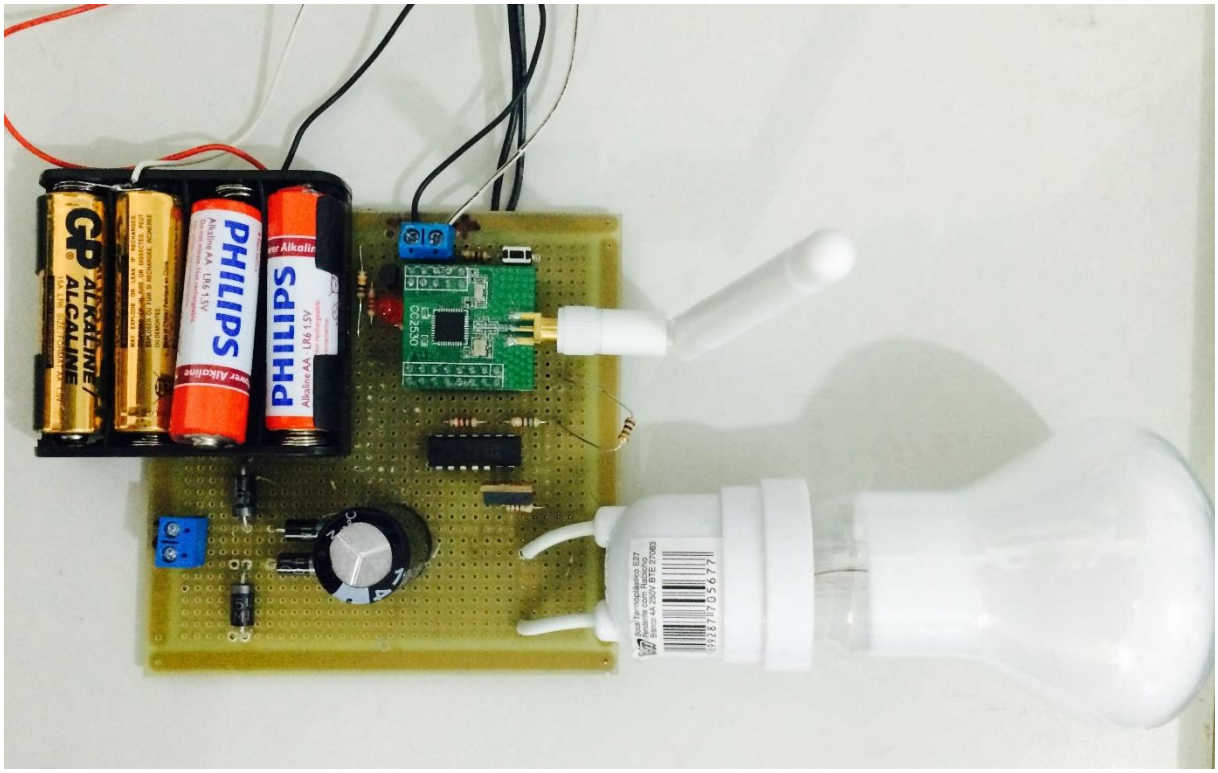


Figura 16 - Circuito Final de Potência
Fonte: Autoria Própria.

4.6 GASTOS ENVOLVIDOS

Por ser um projeto prático, seu desenvolvimento envolve alguns custos inevitáveis, os quais serão expostos a seguir.

Tabela 4 - Custos envolvidos com o projeto

Componente	Preço em reais (R$)
Microcontrolador CC2530	128,00 (32,00 cada)
Componentes Sistema de Potência	20,00
Componentes Placa Impressa para módulo CC2530	9,00 (3,00 cada)
Placa Impressa para módulo CC2530	15,00 (5,00 cada)

Fonte: Autoria própia.

Os custos evidenciados na tabela X são referentes aos gastos que os autores tiveram com o projeto entretanto, caso fosse um projeto real para implementação profissional teriam que ser levados em consideração outros custos como o do computador que serve de servidor e também os valores dos programas utilizados, pois os autores obtiveram licença para estudante para a utilização dos mesmos.

5 CONSIDERAÇÕES FINAIS

Durante o período estipulado a equipe obteve sucesso quanto ao desenvolvimento do projeto proposto. O desenvolvimento do aplicativo de celular foi uma surpresa agradável para a equipe, tendo em vista a facilidade de programação do mesmo sendo que não havia conhecimento prévio do assunto.

O desenvolvimento do servidor contou com três partes principais, sendo elas a programação da comunicação via *sockets*, da comunicação serial e o processamento dos dados recebidos. Quanto à programação dos *sockets* não houve complicações, entretanto a comunicação serial exigiu mais dos autores. Mesmo havendo vários códigos disponíveis, muitos deles não ofereciam funções completas para recebimento dos dados, fazendo com que a equipe investisse grande parte do seu tempo nessa estrutura, finalmente obtendo um resultado confiável.

Na rede ZigBee foi encontrada grande parte das dificuldades com as quais a equipe teve que lidar. Apesar de a Texas Instruments disponibilizar códigos base para o desenvolvimento de aplicações, os códigos são de difícil entendimento. Em resultado disso, a equipe optou por utilizar um código base mais simples, porém que não atua sobre o protocolo ZigBee, apenas implementa o protocolo IEEE 802.15.4. A partir desse ponto foram desenvolvidas funções específicas para cada dispositivo na rede levando em consideração qual seria o papel a ser desenvolvido pelo módulo. Mais uma intensa pesquisa foi feita para que fosse desenvolvida a comunicação serial via UART com o servidor, para a interpretação dos dados recebidos, para a configuração de pinos do microcontrolador como gerador de sinal PWM e etc.

Durante o desenvolvimento da rede foram encontrados grandes problemas que não foram resolvidos a tempo. Por exemplo, não foi encontrada uma solução para fazer com que o módulo da lâmpada enviasse uma mensagem para o coordenador toda vez que mudasse o seu estado, o que poderia gerar uma falha no sistema caso a lâmpada fosse desligada pelo botão, pois o usuário do aplicativo não saberia o estado atual da lâmpada caso não estivesse no mesmo ambiente que ela. A princípio acredita-se que esse problema é causado devido a limitações do código disponibilizado pela Texas Instruments, que foi utilizado para comunicação dentro da rede de comando, pois foram realizados exaustivos testes e percebeu-se que um mesmo dispositivo só poderia enviar ou receber mensagens, mas nunca os dois ao mesmo tempo.

A parte de potência para o acionamento da lâmpada não trouxe grandes problemas à equipe, uma vez que foram utilizados conhecimentos adquiridos em disciplinas do curso. É importante ressaltar os cuidados com a segurança que devem ser tomados ao desenvolver sistemas de potência, pois em altas tensões, podem ocorrer acidentes graves caso ocorra um curto circuito.

Por fim, o projeto desenvolvido foi um sucesso apesar de trabalhoso. Inicialmente acreditou-se poder desenvolver outros módulos para exemplificar o funcionamento do sistema além do módulo de iluminação. Entretanto, só vimos o real tamanho do projeto assim que o iniciamos e enfrentamos as dificuldades reais impostas pelo mesmo. Pôde-se perceber durante o desenvolvimento desse trabalho o grande leque de projetos que podem ser desenvolvidos a partir desse sistema básico desenvolvido pela equipe, que acabou tendo como foco a comunicação dos diversos sistemas.

6 TRABALHOS FUTUROS

No projeto apresentado deu-se maior importância a comunicação entre os dispositivos que se situam na rede como o celular, o servidor e os microcontroladores CC2530. Utilizando apenas um projeto como exemplo (o de iluminação). Como possíveis trabalhos futuros, pode-se apontar:

- A implementação de outros sistemas à serem comandados como:
    - Uma planta de temperatura: com o propósito de manter a temperatura desejada, o usuário pode definir, inclusive sem estar na casa, qual é essa temperatura, para que ao chegar em casa obtenha o conforto desejado. O sistema de controle de temperatura terá diversos sensores e um comando de ar condicionado por sinal infravermelho.
    - Um porteiro eletrônico: o usuário pode verificar quem está na frente da sua residência através de vídeo e se comunicar através de áudio pode fazer o uso do porteiro eletrônico. Assim como o visitante poderá acionar um botão e como consequência o servidor deve enviar um alerta aos dispositivos da rede compatíveis com o sistema. Para este módulo, sugere-se a utilização da versão 802.11g, tendo em vista que é uma versão que se encontra estável e com produtos no mercado completamente compatíveis com a mesma.
- A implementação de um banco de dados, pois facilitaria o acesso aos dados e diminuiria a complexidade do código implementado no servidor. Junto com essa solução, também seria viável a elaboração do código para adicionar novos equipamentos ao sistema de automação residencial via aplicativo de celular.

A utilização de *WebService* ao invés de *sockets* para a comunicação entre celular e servidor. Essa solução permitiria uma transição mais acessível para aplicativos que não sejam da plataforma Android.

REFERÊNCIAS

AURESIDE, **Associação Brasileira de Automação Residencial.** Disponível em: <http://www.aureside.org.br/>. Acesso em Dezembro de 2013.

AMORIN, Diego G.; DAGOSTIM, Eduardo D.; PEREIRA, Luiz T. do V.; BAZZO, Walter A. **TECNOLOGIAS IMPORTADAS, DESENVOLVIMENTO INDUSTRIAL BRASILEIRO: HISTÓRICO, TENDÊNCIAS E AEDUCAÇÃO.** In: Congresso Brasileiro de Ensino de Engenharia (COBENGE), Brasília. 2004. p. 1-10.

Control4: Home Automation and Smart Home Control. Disponível em: <http://www.control4.com/>. Acesso em Agosto de 2014.

ERICSSON WHITE PAPER: **More Than 50 Billion Connected Devices**. Fevereiro, 2011.

FALUDI, Robert. **Building Wireless Sensor Networks**. 1. Ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2011. 321 p.

FARAHANI, Shahin. **ZIGBEE WIRELESS NETWORKS and TRANSCEIVERS**. 1. Ed. Burlington, MA, USA: Elsevier Ltd., 2008. 360 p.

Global Caché: Partners in Connectivity. <www.globalcache.com>. Acesso em Agosto de 2014.

IEEE. **IEEE 802.11™ WIRELESS LOCAL AREA NETWORKS:** The Working Group for WLAN Standards. Disponível em: <www.ieee802.org/11>. Acesso em Agosto de 2014.

IAR SYSTEMS. Disponível em: <www.iar.com>. Acesso em Novembro de 2014.

iSimplex. Disponível em: <www.isimplex.com>. Acesso em Agosto de 2014.

Lutron. Disponível em: <http://www.lutron.com/en-US/Pages/default.aspx>. Acesso em Agosto de 2014.

MOURÃO, L. **O futuro na ponta dos dedos.** Disponível em: <http://www.aureside.org.br/imprensa/arq_constr_300.pdf> Acesso em: Agosto de 2014.

O'HARA, Bob; PETRICK, Al. **IEEE 802.11 handbook:** a designer's companion. 2. Ed. New York: IEEE, 2004. xxxvi, 364 p.

SIEMEINTCOSKI, Michael E.; TOBIAS, Orlando J. Rede industrial sem fio: estudo e aplicação de protocolo ZigBee - Parte 1. **Saber Eletrônica,** São Paulo, v. 44, n. 426, p. 70-75, julho 2008.

SIEMEINTCOSKI, Michael Eberle; TOBIAS, Orlando José. Rede industrial sem fio: estudo e aplicação de protocolo ZigBee. Parte 2. **Saber Eletrônica,** São Paulo, v. 44, n. 427, p. 52-58, ago. 2008.

SIEMEINTCOSKI, Michael Eberle; TOBIAS, Orlando José. Rede industrial sem fios: estudo e aplicação de protocolo ZigBee - Parte Final. **Saber Eletrônica,** São Paulo, v. 44, n. 428, p. 62-67, set. 2008.

SILVA, Denise S. da. **Desenvolvimento e Implantação de um Sistema de Supervisão e Controle Residencial**. 2009. 62 f. Dissertação (Mestrado em Engenharia Elétrica). Universidade Federal do Rio Grande do Norte, Natal-RN, 2009.

SILVA, Francisco A. da. **Comunicação de Computadores utilizando Sockets. Presidente Prudente**. Disponível em: < http://fipp.unoeste.br/~chico/comunicacao_socket/ >. Acesso em: Dezembro de 2014.

Texas Instruments. Disponível em: <www.ti.com>. Acesso em: Outubro de 2014.

VECCHI, Hermes F.; OGATA, Reinaldo Jiunji. **Edifícios Inteligentes.** Disponível em: <http://www.din.uem.br/ia/intelige/domotica/int.htm#introducao>. Acesso em: Dezembro de 2013.

APÊNDICE A – PROGRAMA (APLICATIVO DE CELULAR)

Constam nesta seção apenas alguns métodos do código principal relativo ao aplicativo de celular programado em Java pelo programa Android Studio. O programa completo encontra-se no CD em anexo na biblioteca da UTFPR Campus Curitiba ou disponível através do *link*: https://drive.google.com/folderview?id=0B_S__E3-ms68YUdNNnlWRFIyRkk&usp=sharing.

Os métodos onItemSelected() e alteraSpinnerObjeto() são referentes aos *spinners*, o que acontece quando são selecionados e modificados. O método alteraSlider() é para verificar se deve ser um *slider* referente à iluminação ou então à temperatura e também para realizar a recuperação do valor inicial da barra. O método onProgressChanged() é para atualizar em tempo real qual o valor atual da barra. Por fim, o método OnClickListener() é referente ao botão enviar e o que acontece quando o mesmo é pressionado.

- Alguns métodos referentes à segunda tela do aplicativo.

```
    //-----------------------------------------------------------
    //          Método onItemSelected
    //
    // Descritivo: Esse método é chamado toda vez que um item for
    //             selecionado em algum dos dois Spinners dis-
    //             poníveis na tela.
    //-----------------------------------------------------------
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
{

        String comodo_ant;
        String comp_ant;

        comodo_ant = comodo;
        comodo = (ComodoSpinner.getSelectedItem().toString());

        // Compara se o comodo anterior é igual ao atual
        if(comodo.equals(comodo_ant))
        {
            // Se sim, faz nada
        }
        else
        {
            // Se não, chama a função alteraSpinnerObjeto
            alteraSpinnerObjeto();
        }

        comp_ant = componente;
        componente = (ObjetoSpinner.getSelectedItem().toString());

        // Compara se o como anterior é igual ao atual
        if(componente.equals(comp_ant))
        {
            // Se sim, faz nada
        }
        else
        {
            // Se não, chama a função alteraSlider
            alteraSlider();
```

```
        }
    }

    //-----------------------------------------------------------
    //          Método alteraSpinnerObjeto
    //
    // Descritivo: Verifica qual cômodo foi selecionado no
    //             Spinner do Comodo e define qual array de
    //             objetos deve aparecer no segundo spinner.
    //-----------------------------------------------------------
    public void alteraSpinnerObjeto ()
    {
        if((ComodoSpinner.getSelectedItem().toString()).equals("Quarto 01")) {
            ArrayAdapter adapter2 = ArrayAdapter.createFromResource(this,
R.array.componente_quarto_1, android.R.layout.simple_spinner_item);
            ObjetoSpinner.setAdapter(adapter2);
            ObjetoSpinner.setOnItemSelectedListener(this);
        }
        if((ComodoSpinner.getSelectedItem().toString()).equals("Quarto 02")) {
            ArrayAdapter adapter2 = ArrayAdapter.createFromResource(this,
R.array.componente_quarto_2, android.R.layout.simple_spinner_item);
            ObjetoSpinner.setAdapter(adapter2);
            ObjetoSpinner.setOnItemSelectedListener(this);
        }
        if((ComodoSpinner.getSelectedItem().toString()).equals("Sala 01")) {
            ArrayAdapter adapter2 = ArrayAdapter.createFromResource(this,
R.array.componente_sala_1, android.R.layout.simple_spinner_item);
            ObjetoSpinner.setAdapter(adapter2);
            ObjetoSpinner.setOnItemSelectedListener(this);
        }
    }

    //-----------------------------------------------------------
    //          Método alteraSlider
    //
    // Descritivo: Verifica qual o tipo de Objeto escolhido para
    //             então definir a escala e a unidade mostradas
    //             em forma de texto.
    //-----------------------------------------------------------
    public void alteraSlider ()
    {
        String componente2 = (ObjetoSpinner.getSelectedItem().toString());
        Comp = componente2.split(" ");

        if(Comp[0].equals("Lâmpada"))
        {

            ((TextView) findViewById(R.id.text_valordesejado)).setText("0%
100%");

            // Recupera host e porta para comunicação socket
            String hostPort = txtHostPort;
            int idxHost = hostPort.indexOf(":");
            final String host = hostPort.substring(0, idxHost);
            final String port = hostPort.substring(idxHost + 1);

            // Instancia a classe de conexão com socket
            st1 = new SocketTask2(host, Integer.parseInt(port), 5000) {

                // É chamado toda vez que for chamada a função publishProgress
                protected void onProgressUpdate(String... progress)
                {
                    seekBar.setProgress(Integer.parseInt(progress[0]));
                    ((TextView) findViewById(R.id.text_valoratual)).setText(progress[0]
+ " %");
                    ((TextView)
findViewById(R.id.resposta_socket)).setText(progress[0]);
                }
            };

            // Envia para a classe st o código já montado
            st1.execute(componente2);
```

```
        }
        if(Comp[0].equals("Ar"))
        {
            seekBar.setProgress(30);
            ((TextView) findViewById(R.id.text_valordesejado)).setText("18°C
25°C");
            // Recupera host e porta para comunicação socket
            String hostPort = txtHostPort;
            int idxHost = hostPort.indexOf(":");
            final String host = hostPort.substring(0, idxHost);
            final String port = hostPort.substring(idxHost + 1);

            // Instancia a classe de conexão com socket
            st1 = new SocketTask2(host, Integer.parseInt(port), 5000) {

                // É chamado toda vez que for chamada a função publishProgress
                protected void onProgressUpdate(String... progress)
                {
                    seekBar.setProgress(Integer.parseInt(progress[0]));
                    ((TextView) findViewById(R.id.text_valoratual)).setText(progress[0]
+ " °C");
                    if(progress[0].equals("00"))
                        ((TextView) findViewById(R.id.resposta_socket)).setText("18");
                    else
                        ((TextView)
findViewById(R.id.resposta_socket)).setText(progress[0]);
                }

            };

            // Envia para a classe st o código já montado
            st1.execute(componente2);
        }
    }

    //-----------------------------------------------------------
    //          Método onProgressChanged
    //
    // Descritivo: esse método é chamado toda vez que a barra do
    //             seekBar for alterada de valor. Verifica o tipo
    //             de objeto para definir a escala e a unidade
    //             mostradas na tela.
    //-----------------------------------------------------------
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

        if(Comp[0].equals("Lâmpada"))
        {
            ((TextView) findViewById(R.id.text_valoratual)).setText(progress + " %");
        }
        if(Comp[0].equals("Ar"))
        {
            int numero = (progress / 14) + 18;
            ((TextView) findViewById(R.id.text_valoratual)).setText(numero + " °C");
        }
    }

    //-----------------------------------------------------------
    //          Método onClickListener
    //
    // Descritivo: esse método fica 'escuta' o botão Enviar.
    //             Caso seja pressionado, se conecta via Socket
    //             e monta o código a ser enviado ao servidor.
    //-----------------------------------------------------------
    public OnClickListener btnConnectListener;

    {
        btnConnectListener = new OnClickListener() {

            String codigo;

            // Método realizado quando o botão é pressionado
            public void onClick(View v) {
```

```
                // Recupera host e porta para comunicação socket
                String hostPort = txtHostPort;
                int idxHost = hostPort.indexOf(":");
                final String host = hostPort.substring(0, idxHost);
                final String port = hostPort.substring(idxHost + 1);

                // Monta o código relacionado com os componentes selecionados
                codigo = MontaCodigo();

                // Instancia a classe de conexão com socket
                st = new SocketTask(host, Integer.parseInt(port), 5000) {

                    // É chamado toda vez que for chamada a função publishProgress
                    protected void onProgressUpdate(String... progress)
                    {
                        ((TextView)
findViewById(R.id.resposta_socket)).setText(progress[0]);
                    }
                };

                // Envia para a classe st o código já montado
                st.execute(codigo);

            }

            // Função utilizada para montar o código em uma única String
            public String MontaCodigo() {
                String comod;
                String Compon;
                String Status;
                String[] comod2;
                String[] Compon2;
                String[] Status2;
                String cod1 = "";
                String cod2 = "";
                String cod3 = "";
                String codigo;

                // Recupera os dados: Comodo, Objeto e Status desejado
                comod = (ComodoSpinner.getSelectedItem().toString());
                Compon = (ObjetoSpinner.getSelectedItem().toString());
                Status = ((TextView)
findViewById(R.id.text_valoratual)).getText().toString();

                comod2 = comod.split(" ");
                Compon2 = Compon.split(" ");
                Status2 = Status.split(" ");

                // Monta o código
                if (comod2[0].equals("Quarto")) {
                    cod1 = "Q".concat(comod2[1]);
                }
                if (comod2[0].equals("Sala")) {
                    cod1 = "S".concat(comod2[1]);
                }
                if (Compon2[0].equals("Lâmpada")) {
                    cod2 = "L".concat(Compon2[1]);
                    cod3 = "S".concat(Status2[0]);
                    if(cod3.equals("S100"))
                        cod3 = "SON";
                    if(Integer.parseInt(Status2[0]) < 10)
                        cod3 = "S".concat("0".concat(Status2[0]));

                }
                if (Compon2[0].equals("Ar")) {
                    cod2 = "AR".concat(Compon2[2]);
                    cod3 = "S".concat(Status2[0]);
                }

                codigo = cod1.concat(cod2.concat(cod3));
```

```
            // Retorna o código pronto
            return (codigo);
        }


    };
}
```

APÊNDICE B – PROGRAMA (SERVIDOR)

Constam nesta seção apenas os métodos do código principal relativo ao servidor programado em Java pelo programa Netbeans IDE versão 8.2.1. O programa completo encontra-se no CD em anexo na biblioteca da UTFPR Campus Curitiba ou disponível através do *link*:
https://drive.google.com/folderview?id=0B_S__E3-ms68dnRDdllaTHlmYzg&usp=sharing.

Para inicializar o servidor utiliza-se o método main(), no qual são inicializados tanto a comunicação serial quanto via *sockets*. Para comunicação serial as principais ações são: ler dados, o que é possível através dos métodos LerDados(), run() e serialEvent() em conjunto e enviar dados, a partir dos métodos EnviaCodigo_uart() que de fato envia o código pela UART e pelo Enviacodigo() que prepara a *string* para ser enviada. Para a comunicação utilizando *sockets* apenas o aplicativo servidor é implementado, para tanto utiliza-se os métodos principais serve() a qual está sempre esperando alguma conexão e trata os dados recebidos quando de fato o evento acontece e sendResponse() para enviar alguma informação como retorno ao aplicativo de celular.

- Programa referente ao arquivo principal.

```
 //-------------------------------------------------------
 //          Método main
 //
 // Descritivo: Inicia as classes principais das comunicações
 //             serial e socket.
 //-------------------------------------------------------
    public static void main(String[] args) throws IOException,
gnu.io.PortInUseException, gnu.io.UnsupportedCommOperationException {

        String[] params = new String[2];
        params[0] = "192.168.0.16";
        params[1] = "8090";

        // Inicia a comunicação serial
        Serial SerialComm = new Serial("COM6", 9600, 100);
        SerialComm.connect();

        // Inicia a comunicação socket
        servidor.SocketComm.main(params);

        // Inicia o servidor
        ServerSocket servidor = new ServerSocket(12345);
        Thread thread = new IniciaServidor();
        thread.start();
    }


 //-------------------------------------------------------
 //          Método LerDados
 //
```

```
// Descritivo: Método que funciona apenas se a leitura estiver
//             habilitada. Obtém o fluxo de entrada do
//             processo e habilita algumas funções como
//             notifyOnDataAvailable.
//-------------------------------------------------------
   public void LerDados() {
       if (Escrita == false) {
           try {
               entrada = porta.getInputStream();
           }
           catch (Exception e) {
               System.out.println("SERIAL: Erro de stream: " + e);
               System.exit(1);
           }

           try {
               porta.addEventListener(this);
           }
           catch (Exception e) {
               System.out.println("SERIAL: Erro de listener: " + e);
               System.exit(1);
           }

           porta.notifyOnDataAvailable(true);

           try {
               threadLeitura = new Thread(this);
               threadLeitura.start();
           }
           catch (Exception e) {
               System.out.println("SERIAL: Erro de Thread: " + e);
           }
       }
   }

//-------------------------------------------------------
//          Método run
//
// Descritivo: Vai realizar atividades assim que o método
//             serialEvent receber alguma informação. Espera
//             um pequeno tempo para então imprimir na tela
//             o dado que recebeu.
//-------------------------------------------------------
   public void run() {

       while(true) {
           int   lenght  =   0;

           try {
               Thread.sleep(10);

               if(serial_timeout < 3) {
                   serial_timeout++;
               }
               else {
                   if(serial_timeout == 3) {

                       Dadoslidos      = (new String(bufferLeitura));

                       TrataDadosRecebidos (Dadoslidos);
                       System.out.print("SERIAL: Dados recebidos "+Dadoslidos+'\n');

                       bufferLeitura   = (new StringBuffer());

                       Dadoslidos      = (new String());

                       serial_timeout  =   10;
                       contador        =   0;
                   }
               }
           }
           catch (Exception e) {
```

```
                System.out.println("SERIAL: Erro de Thred: " + e);
            }
        }
    }

//-------------------------------------------------------
//          Método serialEvent
//
// Descritivo: É acionado quando houver um dado na porta RX.
//             Armazena os dados recebidos em um buffer.
//-------------------------------------------------------
    public void serialEvent(SerialPortEvent ev)
        {

        int novoDado = 0;
        serial_timeout      =       0;

        switch (ev.getEventType()) {
            case SerialPortEvent.BI:
            case SerialPortEvent.OE:
            case SerialPortEvent.FE:
            case SerialPortEvent.PE:
            case SerialPortEvent.CD:
            case SerialPortEvent.CTS:
            case SerialPortEvent.DSR:
            case SerialPortEvent.RI:
            case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
                    break;
            case SerialPortEvent.DATA_AVAILABLE:

                    try {
                        novoDado = entrada.read();
                        contador++;
                        bufferLeitura.append((char)novoDado);
                    }
                    catch(IOException ioe) {
                        System.out.println("SERIAL: Erro de leitura serial: " + ioe);
                    }

                    break;
        }
    }


//-------------------------------------------------------
//          Método EnviaCodigo_uart
//
// Descritivo: Método que funciona apenas se a escrita estiver
//             habilitada. Obtém o fluxo de saída ligado à
//             entrada do processo e envia o codigo nessa
//             saida.
//-------------------------------------------------------
    public static void EnviaCodigo_uart(byte buf_send[]) {

        if (Escrita==true) {
            try {
                saida = porta.getOutputStream();
            }
            catch (Exception e) {
                System.out.println("SERIAL: Erro.STATUS: " + e );
            }

            try {
                saida.write(buf_send);
                saida.flush();
                System.out.println("SERIAL: ENVIADO: " + buf_send );
            }
            catch (Exception e) {
                System.out.println("SERIAL: Houve um erro durante o envio. ");
                System.out.println("SERIAL: STATUS: " + e );
            }
        }
```

```
        else {
            System.exit(1);
        }
    }

//-------------------------------------------------------
//          Método Enviacodigo
//
// Descritivo: Função utilizada para preparar a string a ser
//             enviada via UART.
//-------------------------------------------------------
   public static void Enviacodigo (String codigo)
   {
       int string_lenght;
       int cnt_string;
       int cnt_buffer = 0;
       int cnt_clean;
       byte  tx_buffer[] = new byte [0xFF];

       string_lenght = codigo.length();

       for (cnt_string = 0; cnt_string < string_lenght; cnt_string++)
       {
               tx_buffer[cnt_buffer] = (byte) codigo.charAt(cnt_string);
               cnt_buffer++;
       }


       servidor.Serial.HabilitarEscrita();
       servidor.Serial.EnviaCodigo_uart(tx_buffer);
       servidor.Serial.HabilitarLeitura();

       //LIMPAR O BUFFER APÓS A TRANSMISSÃO
       for (cnt_clean = 0; cnt_clean < string_lenght; cnt_clean++)
       {
           tx_buffer[cnt_clean] = 0;
       }
   }


//-------------------------------------------------------
//          Método serve
//
// Descritivo: Abre a conexão via Socket utilizando os parâ-
//             metros selecionados e roda um loop onde fica
//             aguardando uma conexão. Quando há uma conexão
//             recebe o código e envia via UART.
//-------------------------------------------------------
      public void serve() {
            ServerSocket serverSocket = null;
            System.out.println("SOCKET: Iniciando servidor");
            try {
              // Cria a conexão servidora
              serverSocket = new ServerSocket(port, 1, InetAddress.getByName(host));

              // Fica esperando pela conexão cliente
              while (true) {
                    System.out.println("SOCKET: aguardando conexões");
                    Socket socket = null;
                    InputStream input = null;
                    OutputStream output = null;
                    RequisitaAtualizacao ();
                    try {
                        socket = serverSocket.accept();
                        input = socket.getInputStream();
                        output = socket.getOutputStream();
                        System.out.println("SOCKET: conectada");


                        // Transforma os dados recebidos em uma String
                        String requestString = convertStreamToString(input);
                        System.out.println(requestString);
```

```
                        //params[0]= requestString;
                        String PL = requestString.substring(0, 1);
                        if ((PL.equals("S")) || (PL.equals("Q")))
                        {
                            codigo = requestString;
                            System.out.println("comando comdo");
                            // Envia o código via UART
                            servidor.Enviacodigo(codigo);
                        }
                        else if ((PL.equals("L")))
                        {
                            String[] inicio = {cod[1], cod[2], cod[3], cod[4], cod[5],
cod[6]};

                            System.out.println("comando lamp");

                            int i = Integer.parseInt(requestString.substring(9, 10));
                            codigo = inicio[i-1];
                            System.out.println(codigo);
                        }
                        else if (PL.equals("A"))
                        {
                            System.out.println("comando ar");

                            String[] inicio = {cod[7], cod[8], cod[9]};

                            int i = Integer.parseInt(requestString.substring(16, 17));
                            codigo = inicio[i-1];
                        }

                        // retorna a string para o cel **TESTE**
                        String responseString = codigo;
                        sendResponse(output, responseString);
                        //   **FIM TESTE**
                    }
                    catch (Exception e) {
                        sendResponse(output, "ERROR");
                        continue;
                    }
                    finally {
                        // Fecha a conexão
                        try {
                            if (socket != null) {
                                socket.close();
                            }
                        }
                        catch (IOException e) {
                            continue;
                        }
                    }
                }
            }
            catch (IOException e) {
                return;
            }
            finally {
                if (serverSocket != null) {
                    try {
                        serverSocket.close();
                    }
                    catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }

    //-------------------------------------------------------
    //          Método sendResponse
    //
    // Descritivo: Envia o parâmetro responseString como retorno
    //             para o dispositivo que estiver conectado via
```

```
 //             socket.
 //-------------------------------------------------------------
       public void sendResponse(OutputStream output, String responseString) throws
IOException {
             System.out.println("SOCKET: resposta " + responseString + " enviada");
             output.write(responseString.getBytes());
       }
```

APÊNDICE C – PROGRAMA (REDE 802.15.4)

Consta nesta seção apenas os códigos principais relativos à rede 802.15.4 programada em C/C++ pelo programa IAR Embedded Workbench. O programa completo encontra-se no CD em anexo na biblioteca da UTFPR Campus Curitiba ou disponível através do *link*: https://drive.google.com/folderview?id=0B_S__E3-ms68RG91TktrYzFvSnM&usp=sharing.

Para programação dos dispositivos foi utilizado o software IAR Embedded Workbench IDE, que é uma plataforma de desenvolvimento em C/C++ destinada para a implementação dos códigos no microcontrolador CC2530. Para cada um dos dispositivos foi criado uma função em C específica para o seu funcionamento, sendo o do Coordenador denominado appCoordinator(), o da Lâmpada denominado appLight() e o do botão denominado appButton(). Todos esses códigos estão disponíveis para consulta no apêndice C. Para a compilação e depuração dos códigos foi utilizada a placa SmartRF05 Evaluation Board da Texas Instruments, a qual conta com diversos botões, LED's e um display de LCD para auxiliar na depuração do código. Maiores informações podem ser encontradas no site da Texas Instruments.

- Programa referente ao arquivo principal programado em todos os dispositivos.

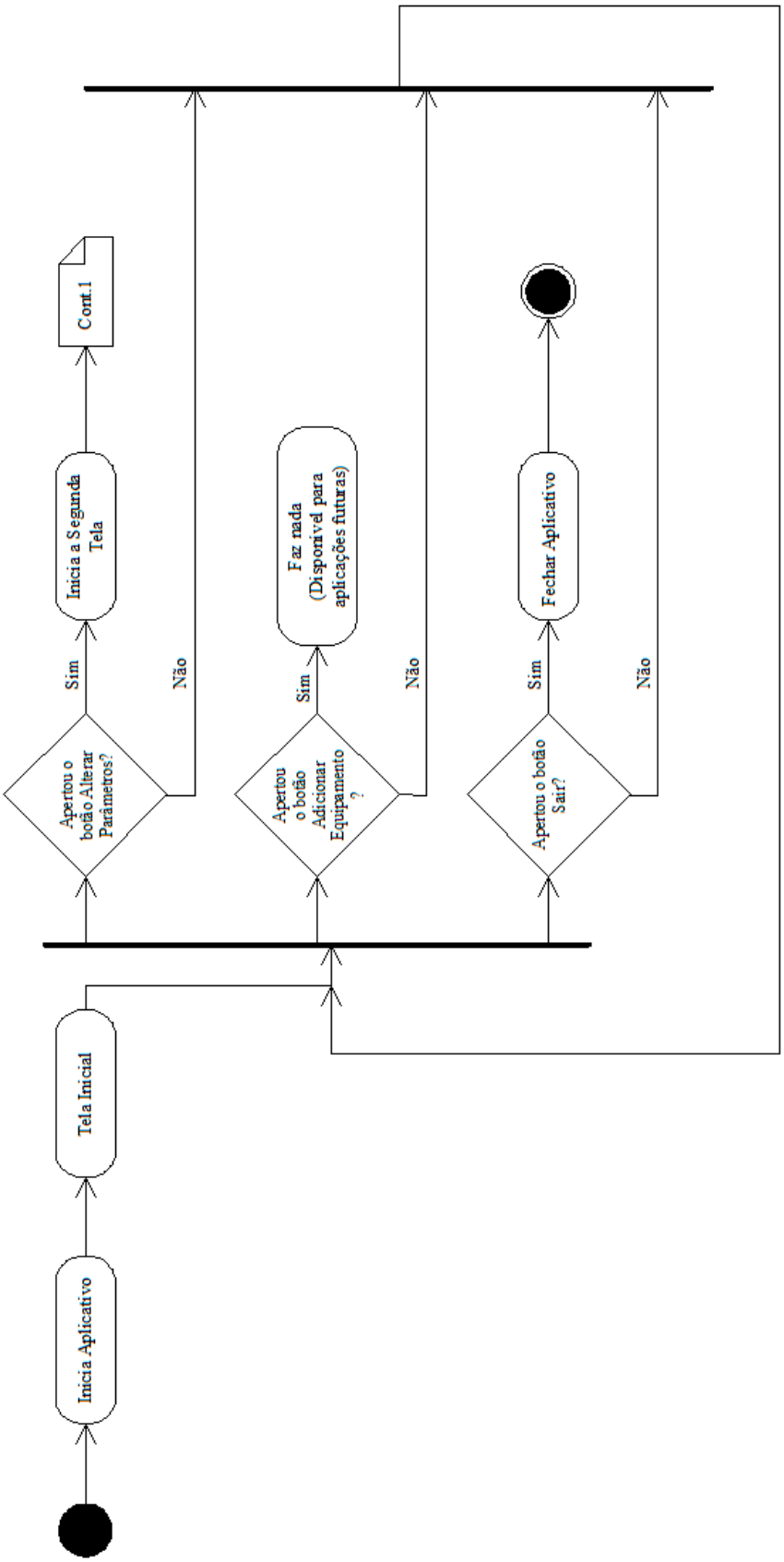
```
/*****************************************************************************
* @fn          appLight
*
* @brief       Código para a aplicação lâmpada. Fica em um loop infinito verificando
*                      se recebe dados via RF.
*
*/
static void appLight()
{

    // Inicialização do LCD
    halLcdWriteLine(HAL_LCD_LINE_1, "Disp.Final-Lamp.");
    halLcdWriteLine(HAL_LCD_LINE_2, " Rede 802.15.4 ");
    halLcdWriteLine(HAL_LCD_LINE_3, " TCC Automacao ");

#ifdef ASSY_EXP4618_CC2420
    halLcdClearLine(1);
    halLcdWriteSymbol(HAL_LCD_SYMBOL_RX, 1);
#endif
    pwm_init();

    // Inicializa a comunicação RF
    basicRfConfig.myAddr = LIGHT_ADDR;
    if(basicRfInit(&basicRfConfig)==FAILED) {
      HAL_ASSERT(FALSE);
    }

    basicRfReceiveOn();
```

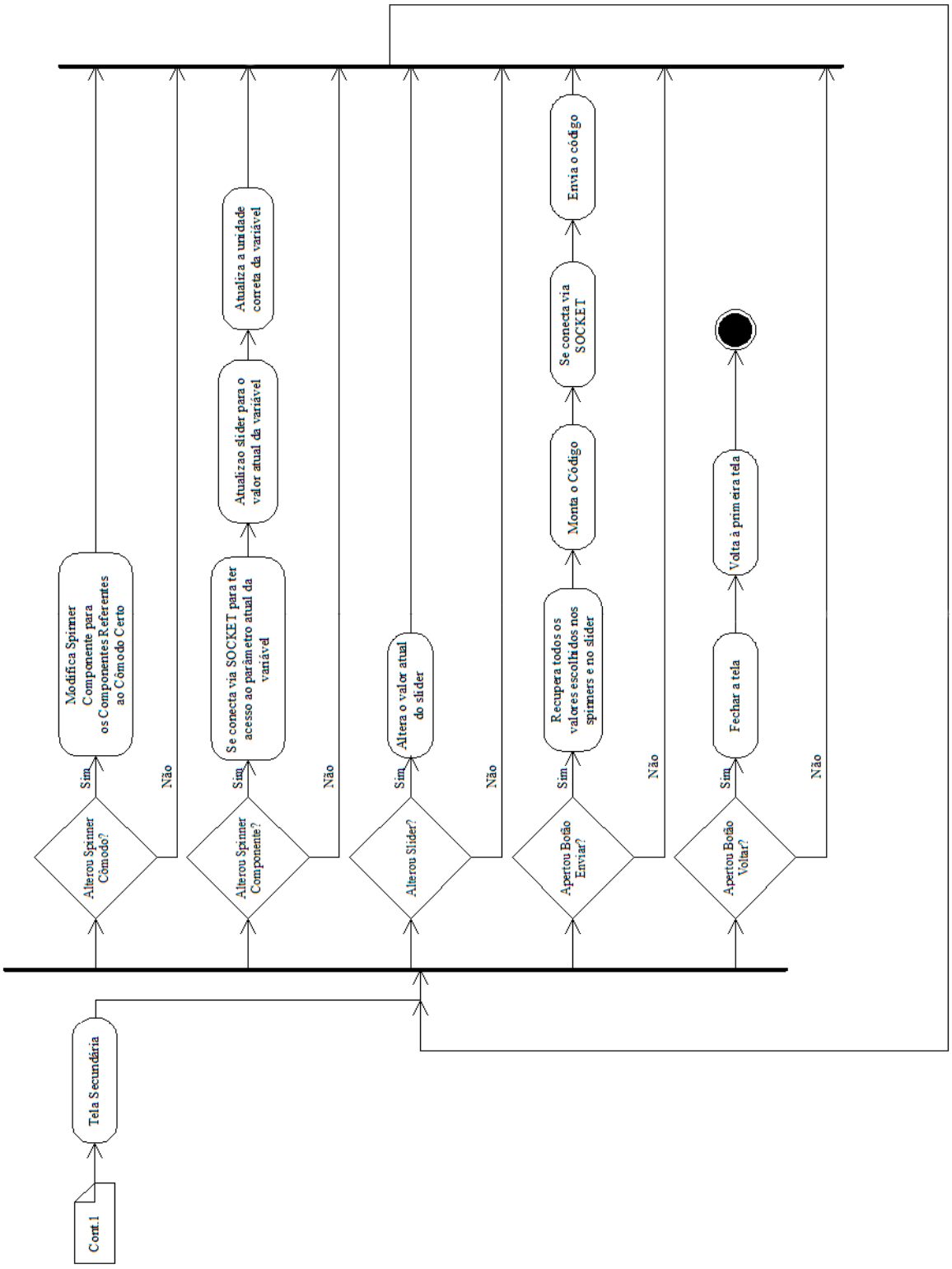
```
       // Indicativo de que o dispositivo está ligado
    halLedToggle(2);
    halMcuWaitMs(50);
    halLedToggle(2);

    // Loop infinito

    while (TRUE) {
              // Espera recever algum dado via RF
        while(!basicRfPacketIsReady());

                    if(basicRfReceive(pRxData, APP_PAYLOAD_LENGTH, NULL)>0)
                    {

                           // Verifica qual foi o comando recebido (PWM ou botão)
                           switch (pRxData[0]){

                                  case LIGHT_CMD:
                                   pwm_set(pRxData[1]);
                                   pTxValor_Atual_Lamp[1] = pRxData[1];
                                   break;

                                  case BUTTON_CMD:
                                                                        // Verifica qual o estado
atual da lâmpada para desligar ou ligar via botão local
                                   if(pTxValor_Atual_Lamp[1] > 0)
                                   {
                                       pwm_set(00);
                                       pTxValor_Atual_Lamp[1] = 0;
                                   }
                                   else
                                   {
                                       pwm_set(90);
                                       pTxValor_Atual_Lamp[1] = 90;
                                   }
                                   break;
                           }
                    }
    }
}


/*****************************************************************************
* @fn          appCoordinator
*
* @brief       Código para a aplicação do Coordenador. Fica em loop infinito esperando
*                       uma informação via UART ou via RF.
*
*/

static void appCoordinator()
{
    int flag;

       // Inicialização do LCD
    halLcdWriteLine(HAL_LCD_LINE_1, "  Coordenador  ");
    halLcdWriteLine(HAL_LCD_LINE_2, " Rede 802.15.4 ");
    halLcdWriteLine(HAL_LCD_LINE_3, " TCC Automacao ");

    // Inicializa a comunicação RF
    basicRfConfig.myAddr = COORD_ADDR;
    if(basicRfInit(&basicRfConfig)==FAILED) {
      HAL_ASSERT(FALSE);
    }

    basicRfReceiveOff();

       // Indicativo de que o dispositivo está ligado
    halLedToggle(2);
    halMcuWaitMs(50);
    halLedToggle(2);
```

```
    pTxData[0] = LIGHT_CMD;

    // Loop principal
    while (TRUE) {

        halIntOff();
        halMcuSetLowPowerMode(HAL_MCU_LPM_3); // Will turn on global
        halIntOn();

        flag = 0;

              // Verifica se o código atual é igual ao anterior
        for(int i = 0; i < 9; i++)
        {
                    if(codigo_recebido_ant[i] == codigo_recebido[i])
                    {
                           // Se sim, faz nada
                    }
                    else
                    {
                           // Se não, seta o flag
                           flag = 1;
                    }
        }

              // Se o código for diferente do anterior, faz as ações relativas
        if(flag)
        {
                    unsigned char comodo[1];
                unsigned char valor[2];

                    // recupera os valores
                    for(int i = 0; i < 9; i++)
                    {
                           if(i == 2)
                                  comodo[i-2] = codigo_recebido[i];
                           if(i > 6)
                                  valor[i-7] = codigo_recebido[i];
                    }

                    pTxData[1] = atoi(valor);

                    // verifica para qual cômodo é o comando
                    if(!strncmp(comodo,"1",1))
                    {
                           if(basicRfSendPacket(LIGHT_ADDR, pTxData,
APP_PAYLOAD_LENGTH))
                           {
                                  // indicativo de que enviou o comando
                        halLedToggle(1);
                        halMcuWaitMs(50);
                        halLedToggle(1);
                           }
                    }

                    if(!strncmp(comodo,"2",1))
                    {
                           if(basicRfSendPacket(LIGHT2_ADDR, pTxData,
APP_PAYLOAD_LENGTH))
                           {
                                  // indicativo de que enviou o comando
                        halLedToggle(1);
                        halMcuWaitMs(50);
                        halLedToggle(1);
                           }
                    }

                    // atualiza o valor anterior
                    for(int i = 0; i < 9; i++)
                    {
                           codigo_recebido_ant[i] = codigo_recebido[i];
                    }
```

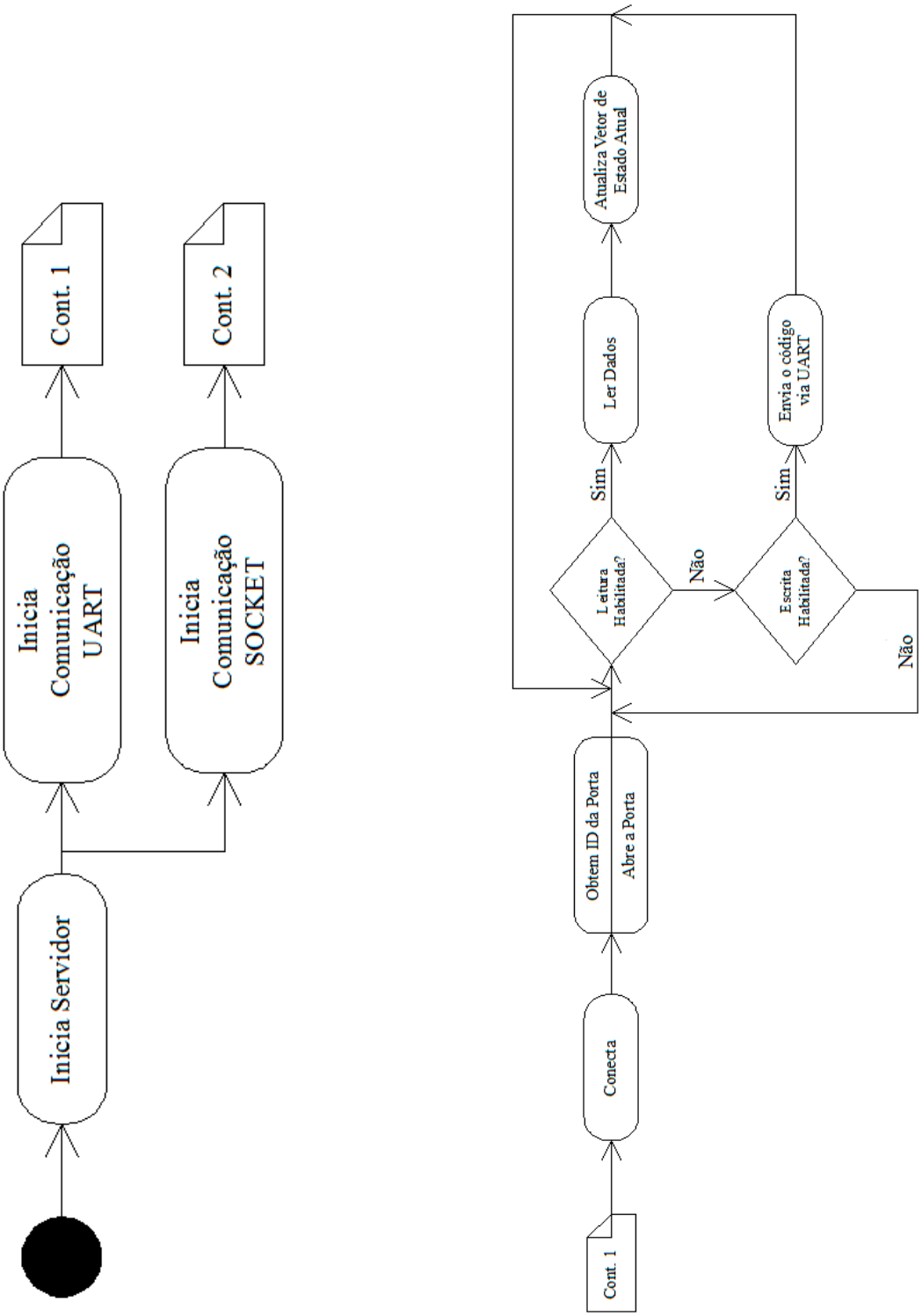
```
        }
    }
}

/*****************************************************************************
* @fn          appButton
*
* @brief       Código para a aplicação do Botão. Fica em loop infinito esperando
*                      o botão S2 ser pressionado.
*
*/
static void appButton()
{

       // Inicialização do LCD
    halLcdWriteLine(HAL_LCD_LINE_1, "Disp.Final-Botao");
    halLcdWriteLine(HAL_LCD_LINE_2, " Rede 802.15.4 ");
    halLcdWriteLine(HAL_LCD_LINE_3, " TCC Automacao ");

    // Inicializa a comunicação RF
    basicRfConfig.myAddr = BUTTON_ADDR;
    if(basicRfInit(&basicRfConfig)==FAILED) {
      HAL_ASSERT(FALSE);
    }

       // Indicativo de que o dispositivo está ligado
    halLedToggle(2);
    halMcuWaitMs(50);
    halLedToggle(2);

    basicRfReceiveOff();

    pTxData[0] = BUTTON_CMD;
    pTxData[1] = 0;

    // Loop infinito
    while (TRUE) {
              // Aguarda o botão (Button 1 ou S2) ser pressionado
        while(halButtonPushed() != HAL_BUTTON_1);

              // Envia o código para a lâmpada
        if(basicRfSendPacket(LIGHT_ADDR, pTxData, APP_PAYLOAD_LENGTH))
        {
                    // indicativo de que enviou o comando
                    halLedToggle(2);
                    halMcuWaitMs(50);
                    halLedToggle(2);
        }

              // Aguarda 1s para que o botão não envie várias mensagens de uma única vez
        halMcuWaitMs(1000);
    }

}
```

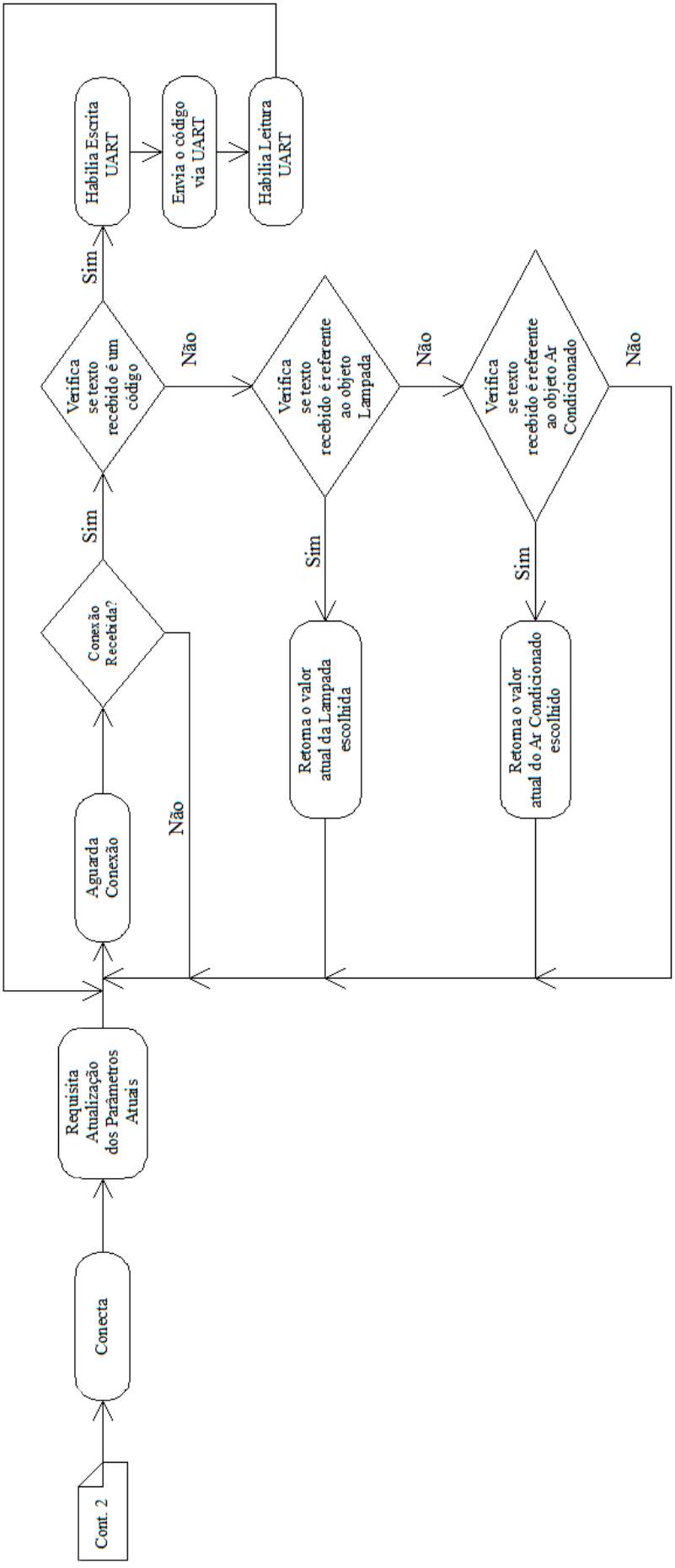
APÊNDICE D – DIAGRAMA DE ATIVIDADES UML DO APP. DE CELULAR



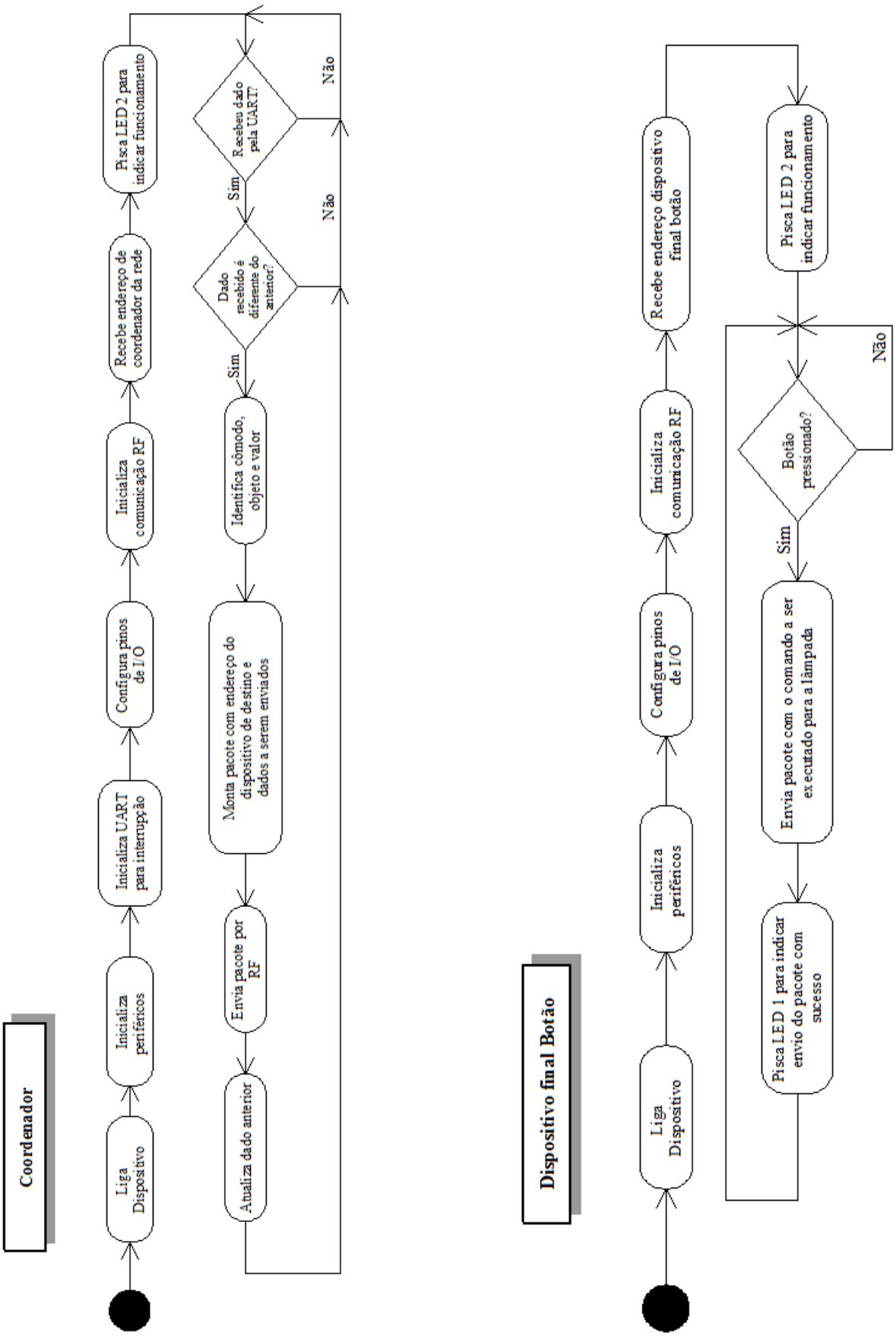

Cont.1
Tela Secundária
Alterou Spinner Cômodo?
Sim
Não
Modifica Spinner Componente para os Componentes Referentes ao Cômodo Certo
Alterou Spinner Componente?
Sim
Não
Se conecta via SOCKET para ter acesso ao parâmetro atual da variável
Atualizao slider para o valor atual da variável
Atualiza a unidade correta da variável
Alterou Slider?
Sim
Não
Altera o valor atual do slider
Apertou Botão Enviar?
Sim
Não
Recupera todos os valores escolhidos nos spinners e no slider
Monta o Código
Se conecta via SOCKET
Envia o código
Apertou Botão Voltar?
Sim
Não
Fechar a tela
Volta à primeira tela

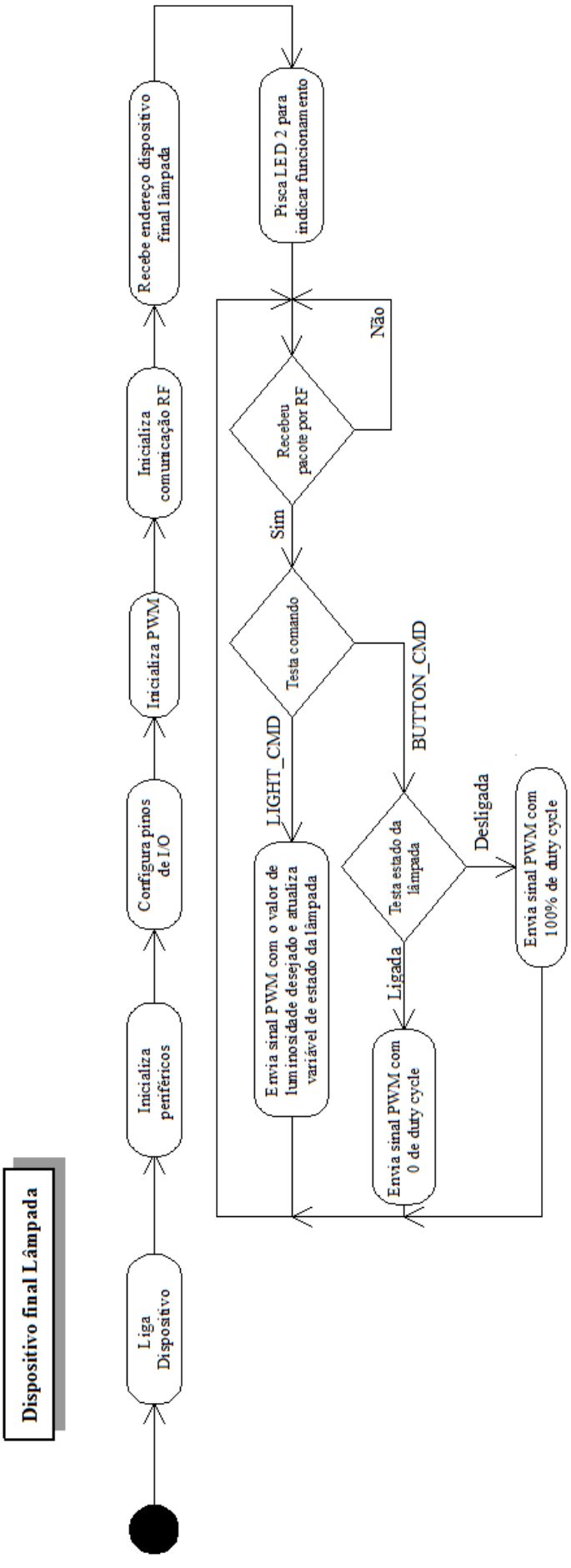
APÊNDICE E – DIAGRAMA DE ATIVIDADES UML DO SERVIDOR



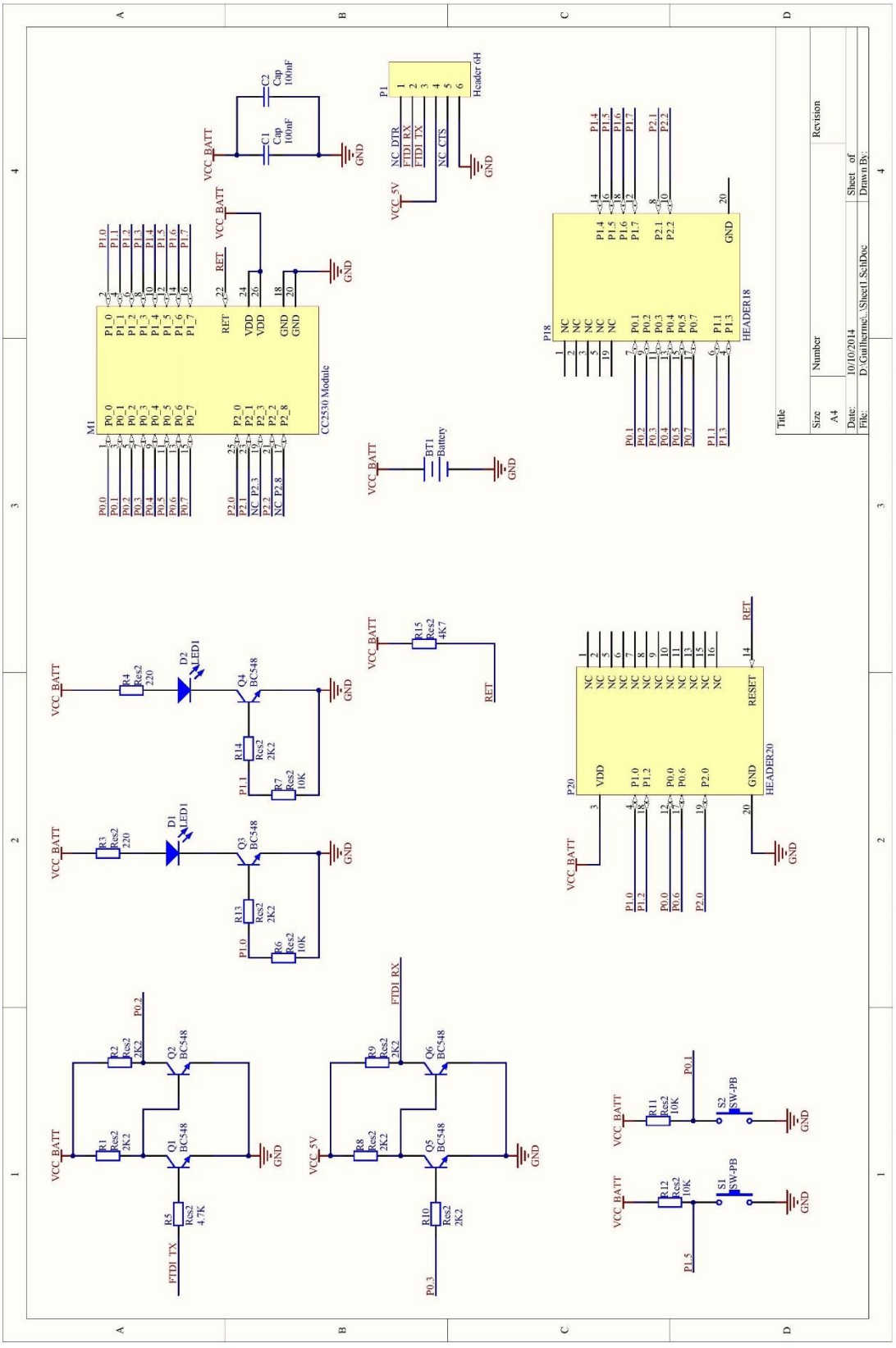

Cont. 2
Conecta
Requisita Atualização dos Parâmetros Atuais
Aguarda Conexão
Conexão Recebida?
Não
Sim
Verifica se texto recebido é um código
Não
Sim
Habilia Escrita UART
Envia o código via UART
Habilia Leitura UART
Verifica se texto recebido é referente ao objeto Lampada
Não
Sim
Retoma o valor atual da Lampada escolhida
Verifica se texto recebido é referente ao objeto Ar Condicionado
Não
Sim
Retoma o valor atual do Ar Condicionado escolhido

APÊNDICE F – DIAGRAMA DE ATIVIDADES UML DA REDE 802.15.4




Dispositivo final Lâmpada
Liga Dispositivo
Inicializa periféricos
Configura pinos de I/O
Inicializa PWM
Inicializa comunicação RF
Recebe endereço dispositivo final lâmpada
Pisca LED 2 para indicar funcionamento
Recebeu pacote por RF
Não
Sim
Testa comando
LIGHT_CMD
BUTTON_CMD
Envia sinal PWM com o valor de luminosidade desejado e atualiza variável de estado da lâmpada
Testa estado da lâmpada
Desligada
Ligada
Envia sinal PWM com 100% de duty cycle
Envia sinal PWM com 0 de duty cycle

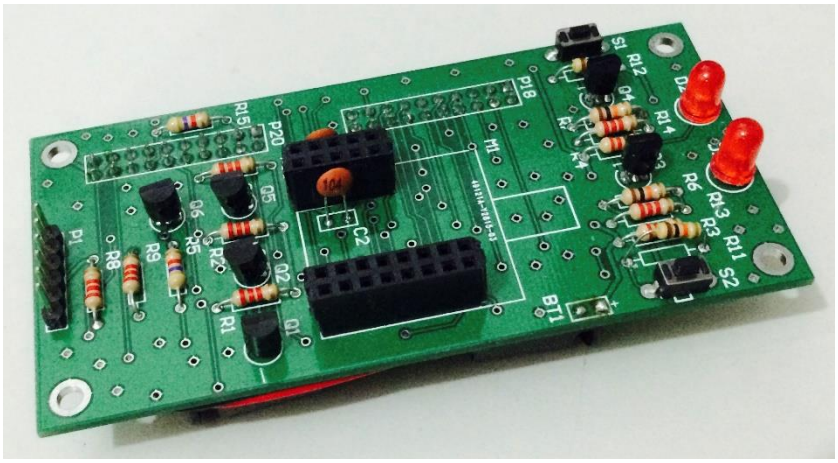
ANEXO A – ESQUEMÁTICO DA PLACA IMPRESSA PARA MÓDULO CC2530



ANEXO B – DISPOSITIVOS UTILIZADOS

Rádio CC2530

FTDI modelo FT232R



Placa Impressa para Módulo ZigBee

Placa SmartRF05EB – Texas Instruments