UNIVERSIDADE FEDERAL DO PARANÁ

EDUARDO HENRIQUE KESSLER
FELIPE JOSÉ PEIXOTO RIBEIRO

CONTROLE E AUTOMAÇÃO DE LUMINÁRIAS LED COM PWM

CURITIBA
2018

EDUARDO HENRIQUE KESSLER
FELIPE JOSÉ PEIXOTO RIBEIRO

CONTROLE E AUTOMAÇÃO DE LUMINÁRIAS LED COM PWM

Trabalho de conclusão de Curso de Graduação apresentado a disciplina TE264 , do Curso de Engenharia Elétrica com ênfase em Sistemas Eletrônicos Embarcados, do Departamento Acadêmico de Engenharia Elétrica da Universidade Federal do Paraná, como requisito parcial para obtenção do título de Engenheiro Eletricista.
Orientador: Prof. Dr. Marcelo Eduardo Pellenz

CURITIBA
2018

TERMO DE APROVAÇÃO

EDUARDO HENRIQUE KESSLER
FELIPE JOSÉ PEIXOTO RIBEIRO

CONTROLE E AUTOMAÇÃO DE LUMINÁRIAS LED COM PWM

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica com Ênfase em Sistemas Eletrônicos Embarcados da Universidade Federal do Paraná como requisito à obtenção do título de Engenheiro Eletricista, pela seguinte banca avaliadora:

_____

Prof. Dr. Marcelo Eduardo Pellenz
Orientador – Departamento de Engenharia Elétrica, UFPR

_____

Prof. Dr. Armando Heilmann
Departamento de Engenharia Elétrica, UFPR

_____

Prof. M.Sc. Waldomiro Soares Yuan
Departamento de Engenharia Elétrica, UFPR

Curitiba, 04 de dezembro de 2018

AGRADECIMENTOS

Eu, Eduardo Kessler, agradeço primeiramente aos meus pais, Rosemeri e Marcilio, que sempre me apoiaram e me incentivaram a concluir esta graduação. Sem eles, eu sei que não estaria aonde estou agora. Junto a eles, agradeço a toda minha família, pelas palavras de incentivo e carinho demonstrado durante todos esses anos.

Agradeço a minha esposa Dryhele, por toda compreensão, apoio, carinho e paciência durante toda esta jornada, o que foi fundamental para que eu me tornasse uma pessoa melhor e para poder fazer este momento se tornar uma realidade.

Aos meus amigos, agradeço por todos os momentos juntos e por sempre me motivarem a nunca desistir dos meus sonhos. Agradecimento especial as amizades que cultivei ao longo desta graduação, pois poderei levar isto para o resto da vida.

Agradeço a Universidade Federal do Paraná e a todos os professores do curso de Engenharia Elétrica, por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram aos seus alunos, não somente por terem ensinado, mas por terem nos feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

Eu, Felipe Ribeiro, agradeço primeiramente a Deus por ter me dado força e saúde para superar os momentos de dificuldade.

Aos meus pais pelo amor, incentivo e apoio incondicional.

A minha namorada Larissa pela paciência e apoio durante toda essa jornada.

A esta universidade, seu corpo docente, pelos ensinamentos.

Por fim, sou grato a todos que de alguma formam direta ou indiretamente participaram da realização desse projeto.

"Para mim viver é estar continuamente motivado. O significado da vida não é simplesmente existir, sobreviver, mas sim crescer, alcançar e conquistar."

Arnold Schwarzenegger

RESUMO

A automação de industrias, comércios, serviços entre outros setores é uma condição cada vez mais necessária devido as demandas do mercado atual. A necessidade da melhoria continua de alguns aspectos como: velocidade, eficiência e qualidade, são fatores primordiais para o desenvolvimento das automações. Assim nos surgiu como um desafio, a implementação de tecnologias (Controle: Microcontroladores, Comunicação: *Bluetooth* e Interface: *Android*) no setor de iluminação, para garantir que os novos produtos, da fabricante de sistemas de iluminação, *Lumicenter Lightining*, possam acompanhar as mudanças no mercado, com um ganho de eficiência energética e uma gama maior de funções disponíveis aos usuários. Essas melhorias visam, manter a empresa “forte” no mercado de iluminação e aumentar a lucratividade com produtos mais abrangentes e de melhor qualidade.

Palavras-chaves: Automação, Iluminação, eficiência energética.

ABSTRACT

Industry, commerce, services, among other areas automation is a trending condition wich has become more and more necessary due to current market's demand. The need for some aspects such as speed, efficiency and quality continuous improvement is a faremost factor to the development of automations. Thus, it has come up to us, as a challenge, technologies mplementation (Control: Microcontrollers, Communication: Bluetooth and Interface: Android) in the lightning field, in order to assure the new products, from lightning systems producer, Lumicenter Lightning, may keep track of market changes, with an energy efficiency gain and a larger range of functions available to users. Such improvements aim to keep the compan "strong" in the lightning market, as well as increase profitability with more overarching and better quality products.

Key-words: Automation, Lighting, energy efficiency.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ARM	*Advanced RISC Machine*
HTML	*HyperText Markup Language*
IBGE	Institudo Brasileiro de Pesquisa e Estatística
IBM	*International Business Machine*
IDE	*Integrated Development Environment*
I2C	*Inter-Integrated Circuit*
MCU	*Microcontroller Unit*
POO	Programação Orientada a Objetos
PWM	*Pulse Width Modulation*
SIG	*Special Interest Group*
SRAM	*Static Random Access Memory*
SPI	*Serial Peripheral Interface*
USART	*Universal Synchronous Asynchronous Receiver Transmitter*
WPAN	*Wireless Personal Area Network*

SUMÁRIO

1 INTRODUÇÃO

Utilizada como grande ferramenta de desenvolvimento a luz artificial, pode ser considerada uma das grandes descobertas da humanidade e se tornou um elemento indispensável na vida cotidiana. A luz artificial surgiu como uma derivação do fogo, centenas de anos depois passou a ser a gás e apenas no século XIX com a descoberta de Thomas Alva Edison de corpos sólidos capazes de fornecer luz, quando excitados por uma fonte elétrica, houve o surgimento da primeira lâmpada elétrica.

A lâmpada então ganhou uma grande projeção, devido aos seus benefícios econômicos e sociais, gerou mudanças nos costumes e hábitos da sociedade, e se tornou um item indispensável. Essa grande expansão na utilização de lâmpadas elétricas, também gerou um aumento consecutivo na demanda energética. Visto que esses sistemas de iluminação representam uma parcela considerável no consumo mundial de energia. Diante desse cenário e do surgimento das preocupações relacionadas a questões de sustentabilidade, eficiência energética, questões ambientais e também ao custo crescente para produzir energia. Fez-se necessário o desenvolvimento de lâmpadas mais eficientes.

Referente ao ganho de eficiência de sistemas de iluminação, esse ganho está diretamente atrelado ao desenvolvimento da microeletrônica que permitiu o conhecimento das características físicas dos materiais semicondutores, possibilitando assim o desenvolvimento dos primeiros diodos emissores de luz, conhecido como LED e também na potência dissipada pela luminária durante um período de tempo.

No entanto essa nova tecnologia possuía certos requisitos para permitir seu funcionamento, diferente das lâmpadas fluorescentes convencionais ou outros modelos conhecidos na época. Os LEDs são considerados em termos leigos, lâmpadas sólidas, pois não possuem nenhum gás ou filamento para desempenhar sua função. No entanto eles necessitam de uma eletrônica embarcada para permitir seu correto funcionamento. Uma luminária LED atualmente é formada pelos seguintes componentes: Em primeiro, o LED, responsável por promover o efeito luminoso. Posteriormente, o *driver*, que é um componente de fundamental importância para permitir que o LED consiga gerar a luz com eficiência. Esse componente pode ser considerado o “motor da luminária”, pois é ele que controla a intensidade da corrente elétrica aplicada no LED, controla a tensão

que recebe da rede elétrica e possui a capacidade de se adaptar a alterações ocorridas na rede elétrica (corrente alternada) mantendo sempre os mesmos parâmetros de alimentação para o LED (corrente continua). Fatores esses que ditam o comportamento e a eficiência do LED durante toda sua vida útil. Por fim, outros elementos como: dissipador de calor, elementos ópticos (lente, refletor e difusor), contato elétrico e parte física compõe a estrutura final.

Diante do crescente aumento da utilização das tecnologias de luminárias a LED, para sistemas de iluminação em larga escala que possuem impactos significativos no quesito de gasto de energia de fabricas, galpões, comércios entre outros estabelecimentos. O controle adaptativo desses *drivers* pode permitir economias de energia. Através da mudança da cor do LED que é diretamente proporcional ao consumo de energia. Essa mudança pode seguir cenários pré-configurados de acordo com as necessidades do cliente luminárias.

1.1 OBJETIVO GERAL

Realizar o desenvolvimento de um aplicativo para o sistema operacional Android e uma central de controle dos *drivers* de LED PWM. Este aplicativo irá se comunicar por *bluetooth* com a central de controle e poderá realizar as seguintes funções em cada luminária LED conectada a esta rede:

- Liga - Desliga;
- Dimerização do fluxo luminoso;
- Criação e gravação de cenários de iluminação;

1.2 PÚBLICO ALVO

Com o intuito de facilitar o controle do sistema de iluminação residencial ou corporativo, será desenvolvido um projeto que possibilita qualquer pessoa, com acesso a um *smartphone* ou *tablet* que possua sistema operacional Android a se interessar por este controle e automação do sistema de iluminação. Foco do produto se mantém a empresas, construtoras, *designers*, projetistas e usuário final.

1.3 DIFERENCIAL DO TRABALHO

O diferencial deste projeto será realizar o desenvolvimento do controle e automação de luminárias LED com drivers PWM de maneira digital, tendo em vista que será um projeto para implementação como produto na empresa *Lumicenter Lighting*, e atualmente, o controle de iluminação vendido por esta empresa é totalmente analógico.

Será um diferencial também criar o sistema de controle que realiza a conectividade de luminárias que utilizem drivers de LED com tecnologia PWM.

1.4 METODOLOGIA DE DESENVOLVIMENTO DO ESTUDO

É notório que o uso de dispositivos móveis vem crescendo em todo o mundo e que o mercado de aplicativos para sistema operacional Android está bastante aquecido. Em conjunto com este crescimento e tendo em vista o aumento da procura por sistemas de controle de iluminação, foi iniciado o projeto em questão. Neste contexto, este projeto iniciará com o estudo de programação orientada a objeto, para desenvolvimento do aplicativo. Foram utilizados conhecimentos adquiridos durante experiência profissional e acadêmica, visando aprofundamento em criação de programas para sistema operacional Android. As pesquisas se estenderam para modelo de módulo *Bluetooth* que será utilizado, juntamente com qual microcontrolador usar, após ter definido que a arquitetura será ARM com cortex M4.

Através dos conceitos pesquisados, é realizada a homologação dos componentes que serão utilizados, onde se define itens como *hardware* e microcontrolador do fabricante NXP, SILICON LABS ou ST, e o módulo *Bluetooth* homologado a partir de amostras de diversos distribuidores, analisando custo benefício. Para desenvolvimento do aplicativo, o *software* escolhido será o *Android Studio*, pois é um *software* que provê um ambiente de desenvolvimento, *debug*, testes e *profile* multiplataforma para Android.

Dentro do aplicativo, deve ser incluído o menu para realizar conexão sem fio. Após conectado, devem ser desenvolvidas opções para reconhecer as luminárias conectadas na central de controle e opções para criação de cenário, assim como opção individual de controle das luminárias.

Quanto ao desenvolvimento do *hardware*, será montado num *kit* de desenvolvimento a ser definido, contemplando a escolha do *chip* do microcontrolador, da mesma maneira que será montado o módulo *bluetooth* e os conectores para os cabos

que serão distribuídos até os *drivers* PWM. A programação do microcontrolador será realizada de maneira simultânea com o desenvolvimento do aplicativo, para que possa ser testado o sistema no geral, com a conectividade funcionando, conseguir enviar o sinal para a central e realizar os comandos nas luminárias.

1.5 RESULTADOS FUNDAMENTAIS A SEREM ATINGIDOS

Além de resultados na aprendizagem com profundidade em temas extremamente atuais e relacionados diretamente a engenharia elétrica, a criação de aplicativos para sistema operacional Android e automação do sistema de iluminação, pode-se destacar como resultado fundamental a conectividade entre as luminárias de determinado local e facilidade para controle do fluxo luminoso de ambiente residencial ou corporativo.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica consiste em ler, selecionar, interpretar e discutir o material de pesquisa, que foi realizada através de textos, artigos, livros, periódicos e afins. Através de conversas e orientação, foram levantados os principais pontos pertinentes a este projeto e com isso realizada a estruturação deste capítulo.

2.1 LINGUAGEM DE PROGRAMAÇÃO: JAVA

Java é uma linguagem de programação computacional lançada pela primeira vez pela *Sun Microsystems* na década de 90. Em sua origem, com o *Green Project*, cujos mentores foram Patrick Naughton, Mike Sheridan, e James Gosling, este projeto não tinha como intenção criar uma linguagem de programação, mas sim de antecipar a nova geração que aconteceria na área da informática e programação. Estes idealizadores da iniciativa acreditavam que em pouco tempo os aparelhos domésticos e os computadores teriam uma ligação (JAVA, 2018).

Como qualquer linguagem de programação, a linguagem Java tem sua estrutura própria, regras de sintaxe e paradigma de programação. O paradigma de programação da linguagem Java baseia-se no conceito de POO (Programação orientada a objetos), que os recursos da linguagem suportam. Esta linguagem deriva da linguagem C, portanto várias regras de sintaxe são semelhantes entre elas. Por exemplo, os blocos de códigos são modularizados em métodos e delimitados por chaves ({ e }) e variáveis são declaradas antes que sejam usadas (PERRY, 2016).

Com o crescimento exponencial da *web*, a *Sun* percebeu que poderia utilizar a ideia criada em 1992 para rodar pequenas aplicações dentro do *browser*. A semelhança era que na *internet* havia uma grande quantidade de sistemas operacionais e *browsers*, e com isso é grande a vantagem em poder programar numa única linguagem, independente da plataforma. Foi aí que o Java 1.0 foi lançado: focado em transformar o *browser* de apenas um cliente (*thin client* ou terminal burro) em uma aplicação que possa também realizar operações avançadas, e não apenas renderizar HTML, que é uma linguagem de marcação utilizada para produção de páginas *web* (CAELUM, 2014).

O foco da linguagem Java é ser aplicado em projetos de médio a grande porte, onde pode haver várias pessoas programando e a linguagem se torna universal. Criar a primeira versão de uma aplicação utilizando Java, mesmo que utilizando IDEs (Integrated Development Environment) de alto nível, será mais trabalhoso que muitas linguagens *script* ou de alta produtividade. Além disso, existe uma grande quantidade de bibliotecas gratuitas para realizar os mais diversos trabalhos é um ponto positivo que justifica a adoção da linguagem Java (CAELUM, 2014).

Cada linguagem é única e se destaca para uma aplicação específica, porém, o uso de Java está em ascensão para várias aplicações em diversos produtos, pois existe grande conectividade entre plataformas (Linux, Unix, Windows, Android).

2.2 PROGRAMAÇÃO ORIENTADA A OBJETOS

Com o intuito de facilitar o trabalho de desenvolvedores de *software*, surge a programação orientada a objetos (POO), cuja intenção é desenvolver um *software* visando o uso do cliente final, ou seja, garantir que a entrega seja fiel ao solicitado. Uma particularidade da POO é fazer o programador pensar nas características do desenvolvimento de forma distintas, transformando-as assim em objeto, aplicando propriedades e métodos.

> A Programação Orientada a Objetos (POO) não é uma linguagem de programação, não é uma plataforma de programação, não é um tipo de programa, é um paradigma de programação. A grosso modo, é um jeito de programar representando o que existe no mundo real: objetos, que possuem características e ações (MITSISTEMAS, 2018).

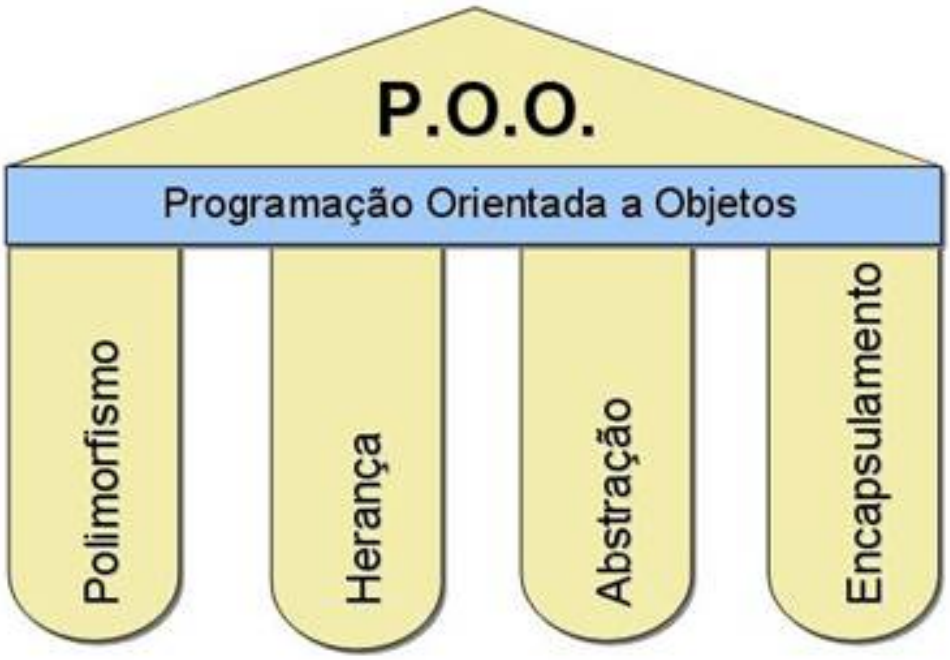
Na POO o programador é responsável por moldar os objetos, e explicar para estes objetos como eles devem interagir entre si. Os objetos “conversam” uns com os outros através do envio de mensagens, e o papel principal do programador é especificar quais serão as mensagens que cada objeto pode receber, definindo suas permissões, e também qual a ação que aquele objeto deve realizar ao receber aquela mensagem em específico (MONQUEIRO, 2007).

Programação estruturada, a maneira tradicional de se programar, é bastante simples e clara. Entretanto, essa simplicidade não “escala”: à medida que programas

crescem, é mais difícil mantê-los. A orientação a objetos é, provavelmente, a solução mais popular para esse problema, e Java é a linguagem de referência para esse paradigma de programação. Na linguagem Java, programas são compostos por unidades chamadas objetos. Os objetos possuem métodos (código) e atributos (dados). Para declarar métodos e atributos, escrevem-se classes, que são modelos a partir dos quais os objetos são criados. Além disso, é possível criar classes novas a partir de outras, usando herança, e classes podem ser agrupadas, para melhor compreensão, em pacotes (DEVMEDIA, 2010).

Na figura 2.1, são mostrados os pilares da programação orientada a objetos, ou seja, o polimorfismo, herança, abstração e encapsulamento.

FIGURA 2.1 – Programação Orientada a Objetos: Estrutura



Fonte: (MITSISTEMAS, 2018)

Para se programar de maneira efetiva em orientação a objetos, não se deve restringir apenas aos conceitos citados em parágrafo anterior. Dominar todos os recursos que a linguagem permite utilizar é crucial para o bom desenvolvimento.

2.3 SISTEMA OPERACIONAL ANDROID

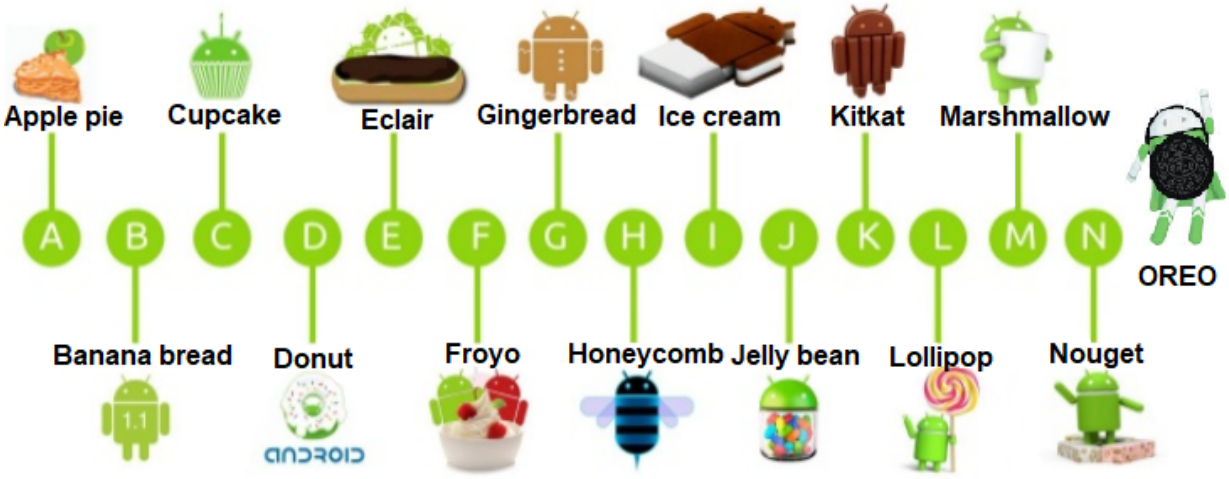
O inicio do sistema operacional Android está diretamente relacionado à empresa Android Inc., que foi fundada por Andy Rubinera, Nick Sears e Chris White, em

outubro de 2003 na cidade de Palo Alto, Califórnia. Inicialmente, a empresa desenvolvia tecnologia totalmente independente de outras empresas e mantinha os seus projetos em absoluto segredo. Dentre os seus principais objetivos, estava o desenvolvimento de um sistema operacional avançado para câmaras digitais. Algum tempo depois, percebeu-se que o mercado para tais dispositivos não era grande suficiente, então a equipe desviou os seus esforços para produzir um sistema operacional para *smartphones*, rivalizando assim com outros sistemas da categoria, como por exemplo, o Symbian, desenvolvido pela Nokia e Windows Mobile, que era uma subempresa da Windows (GUIMARÃES, 2013).

A Google realizou a compra desta companhia em agosto de 2005, aproximadamente dois anos depois de sua fundação. Andy Rubinera liderou a equipe de desenvolvedores no trabalho de uma plataforma móvel baseada em linux. Foram os primeiros passos que culminaram, em 5 de novembro de 2007, no lançamento do projeto intitulado Android, cujo objetivo era desenvolver um sistema para dispositivos móveis sob o padrão aberto (disponível ao público) e construído sobre o *kernel* do linux. O projeto Android está ligado à *Open Handset Alliance*, que é um consórcio de empresas de tecnologia composta por empresas como a Google, Sony, Samsung, operadores de telefonia e fabricantes de dispositivos (GUIMARÃES, 2013).

Para batizar cada uma das quinze versões lançadas até 2018, a empresa resolveu dar nomes de sobremesas, conforme figura 2.2

FIGURA 2.2 – Cronograma de versões do Android



Fonte: (SCARDUELLI, 2017)

Um dos principais motivos pela escolha do Android como sistema operacional para desenvolvimento deste projeto é seu grande número de usuários e grande material disponível para pesquisa. De acordo com o jornal O Estado de São Paulo, em matéria publicada em março de 2018, a plataforma Android já alcançou 2 bilhões de usuários pelo mundo todo. Além disso, de acordo com dados do IBGE (Instituto Brasileiro de Geografia e Estatísticas), em pesquisa publicada em fevereiro de 2018, 77,1% da população com dez anos de idade ou mais já possuem acesso a *smartphone*.

2.4 BLUETOOTH

O *bluetooth* é um padrão de comunicação de curto alcance, baixo custo e baixo consumo de energia amplamente utilizado no mercado de transmissão de dados sem fio.

Esta tecnologia foi pensada na década de 90 para substituir a transmissão de dados via cabos pela empresa Ericsson, que depois se tornou a Sony-Ericsson Corporation. *Bluetooth* tem se tornado largamente utilizado em inúmeros dispositivos e já representa uma parcela significativa do mercado *wireless*. No meio de tantos dispositivos que utilizam esta tecnologia, é possível incluir os dispositivos inteligentes, como telefones celulares, computadores e periféricos (*mouse*, teclados, *joystick*, impressoras, etc) e dispositivos embarcados num geral, como os utilizados em automóveis (SIQUEIRA, 2006).

O nome *bluetooth* nasceu no século X e foi dado em homenagem ao rei da Dinamarca da época, rei Harald Bluetooth. Este rei foi bastante engajado e trabalhou como diplomata entre países europeus, fazendo com que vários acordos comerciais fossem estabelecidos entre si. Os projetistas do *bluetooth* adotaram este nome pois este meio de comunicação permite que diferentes produtos possam se comunicar um com o outro (SIQUEIRA, 2006).

O projeto da especificação do *bluetooth* começou realmente quando a empresa criadora desta tecnologia se uniu com empresas maiores, como Intel Corporation, International Business Machine (IBM), Nokia e Toshiba Corporation. Estas companhias fundaram a Special Interest Group (SIG) no ano de 1998. No ano seguinte outras empresas destes segmentos entraram para SIG e foi com esta união que permitiu o desenvolvimento das especificações desta nova tecnologia, com padrões considerados

abertos para assegurar aceitação e compatibilidade com diversos dispositivos. A tecnologia *Wireless Personal Area Network (WPAN)* baseada na especificação *bluetooth* passa a ser um padrão IEEE, sob denominação 802.15.1 WPANs (SIQUEIRA, 2006).

O *bluetooth* permite a transmissão de dados sem fio através de *piconets*. As *piconets* são formadas por um método mestre e por diversos escravos, dos quais até 7 podem ocupar o espaço de ativos, enquanto o restante considera-se escravos estacionados (podendo chegar até 255). Cada um destes ganha um “endereço de estacionamento” de 8 bits. No caso deste projeto, a ligação é ponto a ponto, mas quando se trata de *bluetooth*, também existe a possibilidade de conexão ponto a multiponto, com mais que dois aparelhos na mesma transmissão, mas não é aplicada neste projeto (UFRJ, 2012).

O acesso ao meio é realizado através dos saltos em frequência em que o *bluetooth* também consegue evitar interferência entre os vários dispositivos operantes em uma mesma frequência. Depois de estabelecida a conexão entre o mestre e seus componentes denominados escravos, cabe ao mestre ditar os saltos que serão efetuados e seguidos pelos escravos. Existem 79 frequências que podem ser utilizadas, cada uma espaçada da outra pela frequência de 1MHz. A cada segundo são feitos 1600 saltos. Desse jeito, mesmo que duas redes partilhem da mesma frequência, a interferência causada por isso será muito pequena ou não existirá, porque ocorre somente durante uma pequena fração de segundo (UFRJ, 2012).

Em qualquer conexão por rede sem fio a segurança da informação é um fator critico e bastante importante. Pensando neste quesito, são aplicadas técnicas de criptografia nos dados transmitidos pelo *bluetooth*, além de senha de acesso para conexão no módulo que é utilizado neste produto.

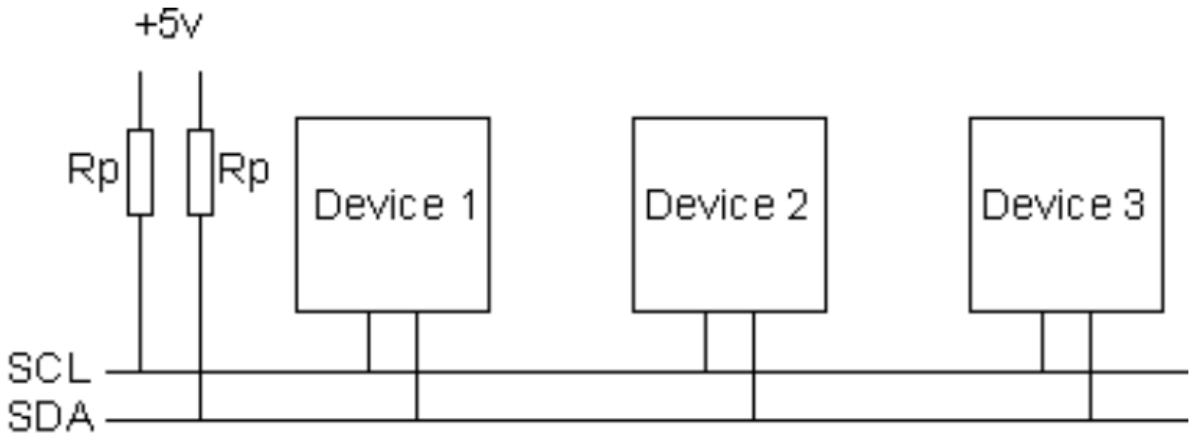
2.5 COMUNICAÇÃO I2C

O protocolo I2C descreve o funcionamento de um barramento de comunicação serial que utiliza basicamente dois fios. Inventado pela empresa *Philips Semiconductor* no início da década de 90, este protocolo é muito utilizado para conectar periféricos de baixa velocidade a placas-mãe, microcontroladores, CIs diversos e afins. Tanto a unidade de controle quanto os periféricos devem possuir implementação e suporte I2C, seja via *hardware* no próprio SoC ou utilizando CIs externos, ou até mesmo via

software, através de um método chamado *bit-bang*, onde o funcionamento do protocolo é emulado bit a bit (UNIVASF, 2015).

O barramento I2C é composto de dois fios, *SDA (Serial Data)* e *SCL (Serial Clock)*, além da alimentação VDD e GND, tipicamente de 3,3 ou 5V. Os fios de comunicação possuem *pull-ups*, como pode ser visto na figura 2.3. Devido ao tamanho do endereço, que pode ser de 7, 10 e até 16 bits, e pela restrição de espaço, o número de nós em um único barramento é bastante limitado. Desta maneira não se pode ultrapassar poucos metros de fios com esta comunicação, pois a capacitância total máxima, algo em torno de 400pf, impede o funcionamento correto do barramento (UNIVASF, 2015).

FIGURA 2.3 – Representação da comunicação via I2C



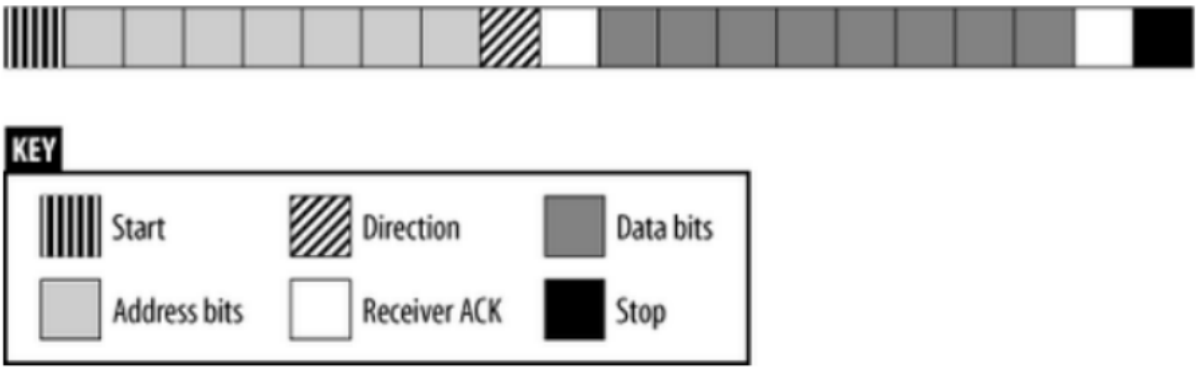
Fonte: (UNIVASF, 2015)

Na comunicação I2C, cada byte transmitido deve ser confirmado pelo receptor. Após a transmissão do oitavo bit de dado, o mestre libera a linha SDA e gera um pulso adicional de *clock* em SCL. Isto aciona o receptor, que, então, deve confirmar o recebimento do byte colocando a linha SDA em baixo. Se o receptor não realizar este processo, o mestre aborta a transmissão e torna medidas apropriadas de tratamento de erro (UNICAMP, 2018).

Qualquer número de bytes pode ser transmitido em um pacote I2C. Se o receptor está impossibilitado de receber mais bytes, ele aborta a transmissão segurando o *clock* em nível baixo, forçando o transmissor a aguardar até que o SCL seja liberado (UNICAMP, 2018).

A figura 2.4 exemplifica estes aspectos apontados, mostrando qual a sequência para transmissão dos pacotes de informação.

FIGURA 2.4 – Pacote de informação via I2C



Fonte: (UNICAMP, 2018)

Esta transferência de dados ocorre segundo uma taxa de 100kHz, utilizando endereçamento com 7 bits no modo normal e para o endereçamento de modo rápido, considerar 3,4MHz com 10 bits. Este protocolo atende diversos tipos de dispositivos, podendo realizar interface com EEPROMS, *flash*, algumas memórias RAM, relógios de tempo real, temporizadores *watchdog* e diversos microcontroladores (UNICAMP, 2018).

2.6 SOFTWARE ANDROID STUDIO

O *Android Studio* é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos *Android*. Anunciado em 2013, durante a *Google I/O*, este desenvolvedor inicialmente seria apenas para uso de aplicativos em *smartphones*, mas atualmente permite a criação também para *smartwatches*, *tablets*, carros e tudo mais que se relacionar ao sistema operacional (STUDIO, 2018).

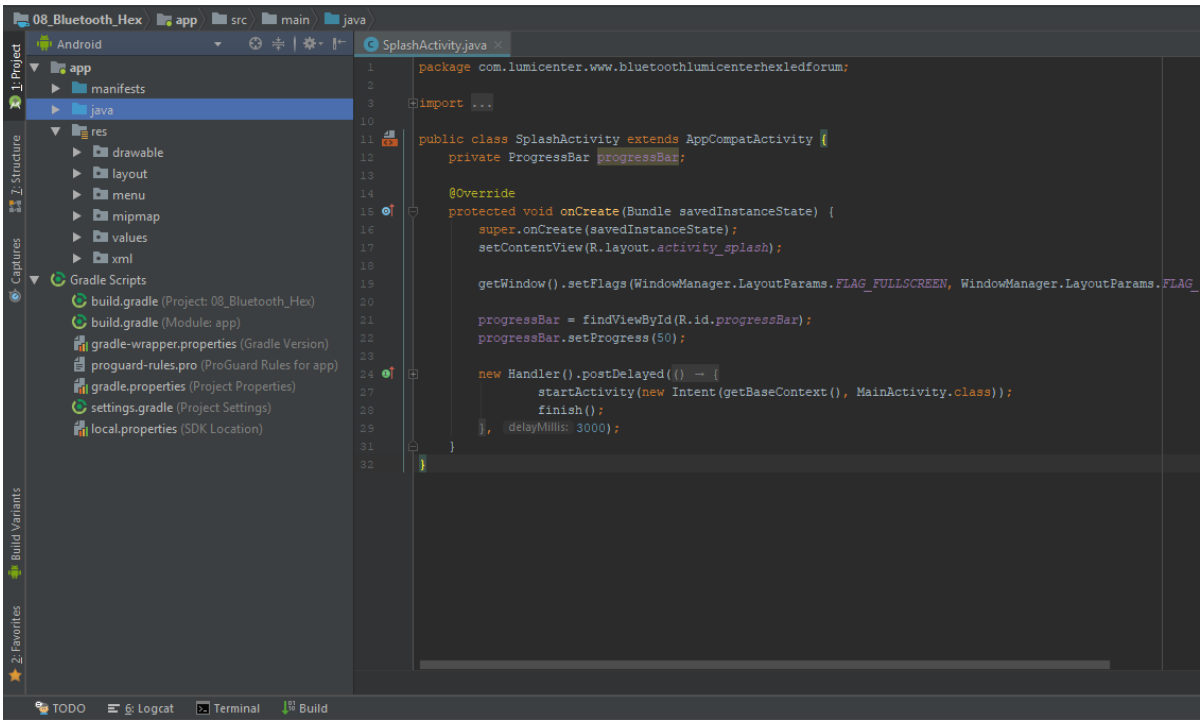
Além do editor de código e das ferramentas de desenvolvedor avançados desta IDE, o *Android Studio* oferece diversos recursos para aumentar sua produtividade na criação de aplicativos *Android*, conforme apresentado em sua página oficial (developer.android.com) (STUDIO, 2018):

- Um sistema de compilação flexível baseado no Gradle;
- Um emulador rápido com inúmeros recursos;
- Um ambiente unificado para você poder desenvolver para todos os dispositivos Android;
- Instant Run para aplicar alterações a aplicativos em execução sem precisar compilar um novo APK;

- Modelos de códigos e integração com GitHub para ajudar a criar recursos comuns dos aplicativos e importar exemplos de código;
- Ferramentas e estruturas de teste cheias de possibilidades;
- Ferramentas de verificação de código suspeito para detectar problemas de desempenho, usabilidade, compatibilidade com versões e outros.

Este *software* permite a criação de *layouts* grandes e complexos, a partir de uma interface simples e eficiente, onde é possível desenvolver, visualizar as telas do aplicativo em tempo real, simular e depurar o projeto em questão. Na figura 2.5 é mostrado o *layout* inicial de desenvolvimento desta IDE.

FIGURA 2.5 – Layout do software Android Studio



Fonte: (Os autores, 2018)

O Android Studio usa o *Gradle* como o sistema de compilação de base, com outros recursos específicos do *Android* sendo disponibilizados pelo *Android Plugin for Gradle*. Esse sistema de compilação é executado como uma ferramenta integrada no menu do Android Studio e de forma independente na linha de comando. É possível usar os recursos do sistema de compilação para personalizar e ampliar o processo de compilação, criar diversos arquivos APK do seu aplicativo com diferentes recursos usando o mesmo projeto e os mesmos módulos, além de reutilizar códigos e recursos nos conjuntos de origem.

2.7 PWM - MODULAÇÃO POR LARGURA DE PULSO

O PWM (*Pulse Width Modulation*) refere-se ao conceito de pulsar em certa frequência um sinal digital em um condutor (SILVEIRA, 2016). Possui inúmeras aplicações na eletrônica, aplicando-se em simulação de tensão estática variável e é comumente aplicada no controle de motores, aquecedores, *LEDs* e no caso deste projeto, tensão DC.

Um dispositivo quase puramente digital como um microcontrolador pode trabalhar com entradas e saídas que possuem apenas dois possíveis estados: ligado ou desligado (0 ou 1). Assim, é possível facilmente usá-lo para controlar a tensão contínua, por exemplo, ligando e desligando o mesmo em certa frequência. Da mesma forma que você pode usá-lo para controlar quase qualquer dispositivo elétrico usando dispositivos adequados (TEIHAL, 2014).

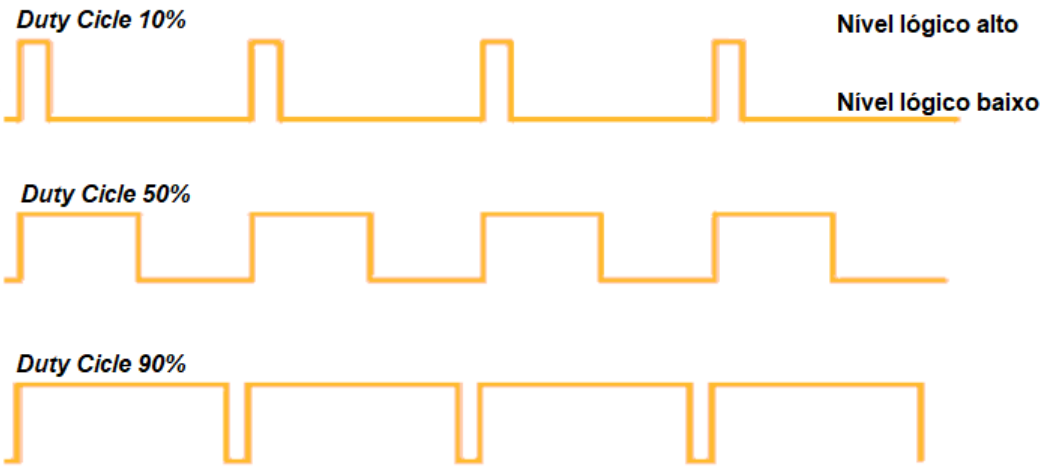
A modulação por largura de pulso se estabelece como uma maneira de codificar digitalmente níveis de sinal analógico (alto ou baixo). Nesta técnica, através do uso de contadores de alta resolução, o ciclo de trabalho de uma onda quadrada é modulado para codificar um nível de sinal analógico específico para que então ele atenda os requisitos de uma aplicação desejada, trabalhando com seu *duty cicle* personalizado conforme necessidade do projeto (SILVEIRA, 2016).

De modo geral, para gerar o sinal PWM, o circuito amplificador aceita a entrada do sinal de *feedback* e uma referência de tensão estável. O comparador realiza a verificação entre tensão de saída do amplificador de erro com a rampa do oscilador, produzindo uma largura de pulso modulada. A saída do comparador é aplicada à lógica de comutação, cuja saída vai para o MOSFET de energia externa. A lógica de comutação fornece a capacidade de ativar ou desativar o sinal PWM aplicado ao MOSFET de energia (DAVIS, 2004).

Este sinal modulado PWM é considerado puramente digital, pois independente do instante de tempo, a alimentação de corrente contínua ou está em nível lógico alto (totalmente ligada) ou em baixo (totalmente desligada). O valor da fonte de tensão ou de corrente contínua é fornecida à carga analógica por meio de uma série repetitiva de impulsos 1 e 0 em determinada frequência para alcançar o *duty cicle* desejado. A figura 2.6 mostra três sinais PWM diferentes, sendo que o primeiro possui a saída com ciclo de trabalho de 10%, ou seja o sinal está em nível lógico alto por 10% do tempo

total mostrado graficamente. Já os outros dois sinais, mostram estes mesmos aspectos, porém com ciclos de 50 e 90%, respectivamente (SILVEIRA, 2016).

FIGURA 2.6 – Exemplo de sinal PWM e seu *Duty Cicle*

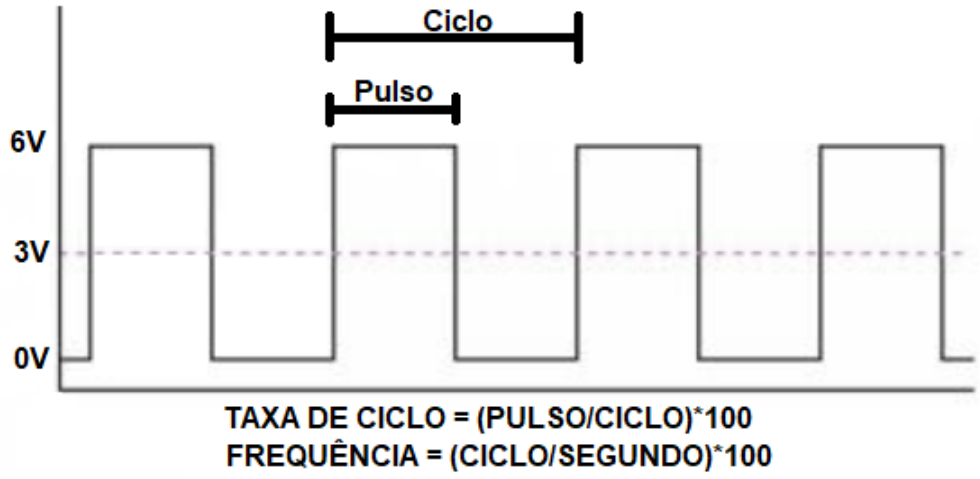


Fonte: (SILVEIRA, 2016) ADAPTADO

Uma das principais vantagens de aplicar a modulação por pulsos é que o sinal permanece digital em todo instante de tempo, desde o processador até o sistema controlado e nenhuma conversão de digital para analógico se faz necessária. Ao manter o sinal digital, os efeitos de ruído são minimizados pois um ruído só pode afetar um sinal digital se ele for forte o suficiente para alterar uma lógica 1 para uma lógica 0 ou vice-versa (SILVEIRA, 2016).

Na figura 2.7 é possível identificar na modulação de tensão pelo tempo, quais são os pulsos, qual o ciclo e realizar o cálculo da taxa deste ciclo e sua frequência.

FIGURA 2.7 – Sinal PWM para demonstrar cálculos



Fonte: (SILVEIRA, 2016) ADAPTADO

2.8 MICROCONTROLADOR NXP

A LPC80X é uma família *MCU (Microcontroller Unit)* de 32 bits baseada em Arm Cortex-M0 de baixo custo, que opera em frequências de CPU de até 15 MHz. A família de MCUs LPC80X suporta até 32 kB de memória flash e até 4 kB de SRAM.

Esta família de microcontroladores possui um núcleo otimizado para energia, pequena área ocupada em pacotes populares e opções de deslocamento de nível graças aos seus trilhos de força separados. O complemento periférico do LPC80X inclui um motor CRC, interfaces de barramento *I2C (Inter-Integrated Circuit)*, até dois *USARTs (Universal Synchronous Asynchronous Receiver Transmitter)*, uma interface *SPI (Serial Peripheral Interface)*, interface de toque capacitivo (cap touch), um temporizador multi-taxa, temporizador de auto-ativação, um propósito geral 32 contador de bits / temporizador, um ADC de 12 bits, um DAC de 10 bits, um comparador analógico, portas de E/S configuráveis por função através de uma matriz de comutador, um mecanismo de correspondência de padrão de entrada, unidade lógica programável (PLU) e até 30 pinos de E/S de uso geral (NXP, 2018).

2.8.1 ARQUITETURA ARM

Processadores e microcontroladores *ARM (Advanced RISC Machine)* são baseados em RISC, têm seu projeto focado em baixo consumo de energia e apresentam tamanho reduzido do chip. Essas características tornam a arquitetura ideal para aplicações que sofrem restrições de consumo de energia, e de tamanho dos dispositivos. De fato, processadores ARM podem ser encontrados em grande variedade de aparelhos, e, entre eles, estão muitos que já fazem parte do cotidiano de inúmeros usuários, como *Smartphones*, *Tablets*, televisores, *eReaders*, etc (FORTE, 2015).

Na evolução desta arquitetura, desde sua invenção trouxe diversas capacidades ao processador ARM, tornando-o mais popular entre dispositivos embarcados, além de fazer dele objeto de interesse em projetos acadêmicos cada vez mais elaborados. A história da arquitetura ARM se inicia na década de 1980, quando a empresa Acorn Computer assina um contrato com a emissora britânica BBC, afim de projetar a arquitetura do processador que seria usado no projeto *Computer Literacy Project*. Com seu sucesso, nasceu o primeiro processador da família ARM, o ARM1, que foi utilizado como coprocessador no computador da BBC e foi sucedido por versões melhores com

o tempo de uso (FORTE, 2015).

A fundação da empresa ARM Ltd., atual desenvolvedora deste tipo de arquitetura, veio após grande sucesso de venda dos produtos, ainda fabricados pela VLSI Technology Inc. Fundada por Acorn Computer, VLSI Technology e Apple Computer, foi mantida a decisão da não fabricação de processadores, mas sim realizar licença de arquiteturas projetadas para fabricantes de semicondutores. Após isto, o nome passou de *Acorn RISC Machine* para *Advanced RISC Machine*.

> Desde o fornecimento do endereço IP para os chip até a entrega dos serviços em nuvem que permitem que as organizações gerenciem com segurança a implantação de produtos em todo o seu ciclo de vida, a Arm oferece uma solução completa de Internet das Coisas (IoT) para nossos parceiros e clientes. Está enraizado na nossa profunda compreensão do futuro da computação e segurança (ARM, 2017).

A arquitetura utilizada no projeto é ARM Cortex-M0+, que tem como principal característica a eficiência energética de seu processamento. Dentre seus principais benefícios, segundo especificação apresentada pelo fabricante, estão:

- Potência extremamente baixa: O mais eficiente em termos energéticos de todos os processadores ARM. O Cortex-M0+ atinge um consumo de energia abaixo de 4µW/MHz, enquanto alcança um desempenho de 2,46 CoreMark/MHz (ARM, 2017);
- Versátil: O Cortex-M0+ inclui funcionalidades opcionais que permitem aos programadores o alcance numa solução ideal para a finalidade de uma ampla gama de aplicações. Incluem a interface I/O de ciclo único para um controle mais rápido, o *MTB (Microtrave Trace Buffer)* para depuração aprimorada e outros que são comuns a todos os processadores Cortex-M (ARM, 2017);
- Processador Cortex-M0+ é binário: Compatível com o Cortex-M0 e é compatível com binário ascendente para todos os outros processadores Cortex-M, fazendo com que o *software* reutilize uma vantagem real. Os desenvolvedores também se beneficiam do amplo ecossistema de ferramentas integradas, software e base de conhecimento da parceria ARM (ARM, 2017).

2.9 ILUMINAÇÃO

Sistemas de iluminação automatizados e com tecnologia *LED* em ambientes corporativos, estão cada vez mais sendo utilizados, como forma de melhorar o desempenho dos funcionários através do aumento do conforto e no controle de fluxo luminoso, além de também contribuir com a saúde do empregado, visto que esses passam grande parte do dia em seus ambientes de trabalho e necessitam de iluminação adequada para realizarem suas atividades.

Em conjunto com essas características, cenários personalizados permitem diminuir o consumo de energia (através da dimerização) ao que de fato é necessário, para manter as condições mínimas de trabalho (500 luxes segundo norma NBR ISO 8995-1). Além desses aspectos uma boa iluminação corporativa, pode harmonizar e valorizar o projeto arquitetônico empregado em determinados ambientes, espelhando o projeto e a essência que a empresa deseja apresentar. Sendo assim, a iluminação corporativa, pode ser avaliada, como mais do que algo apenas funcional. Ela colabora em diversos aspectos relacionados ao desempenho da empresa e projetos bem desenvolvidos podem agregar valor e destaque a produtos, salas de reunião, escritórios e residência de consumidores finais.

Quando se trata de iluminação pessoal e residencial, projetos arquitetônicos que envolvem sistemas digitais para manipulação de fluxo luminoso, criação de cenários em cômodos de residência e alteração da temperatura de cor das luminárias está em destaque mundialmente, segundo projetos apresentados na EXPOLUX 2018, que é a maior feira de iluminação do Brasil.

2.9.1 *DRIVERS* PARA LUMINÁRIAS *LED*

Os equipamentos eletrônicos para iluminação estão em constante desenvolvimento, visando eficiência energética e boa relação custo benefício, sendo uma grande preocupação por parte dos fabricantes e desenvolvedores de luminárias, estar a par dessas novas tecnologias. Na área de iluminação essa tendência está crescendo bastante e as luminárias de *LED* estão cada vez mais em alta, principalmente em grandes empresas, visando diminuição de custo dos gastos com energia e a necessidade em aumentar eficiência energética (alto fator de potência).

Os *LEDs* são dispositivos que operam com corrente contínua passando por

eles para seu funcionamento ideal. Para que isso ocorra, são utilizados circuitos acionadores, denominados *drivers* de *LED*, que em essência são fontes de corrente contínua com tensão de saída constante numa faixa específica. Quando se trata de alimentar um *LED*, não é recomendado aplicar uma fonte de tensão direta sem um resistor ou um *driver*, pois sem o *driver* ligado nas placas de *LED* (circuito com vários *LEDs* em série), qualquer variação da fonte de tensão, pode acarretar numa grande alteração da corrente direta que irá para a saída, o que pode diminuir a vida útil do *LED* e enfraquecer considerável sua confiabilidade.

As luminárias de *LED* são constituídas por ligações contendo vários *LEDs* geralmente de baixa potência e não podem ser conectadas diretamente à rede elétrica, pois as tensões e correntes nominais são incompatíveis, sendo necessária a conexão de *drivers* que convertem a tensão alternada da rede em uma saída com tensão e corrente contínuas. Os *drivers* propiciam o controle das grandezas envolvidas no acionamento das luminárias de *LED*, disponibilizando tensões e correntes dentro dos limites especificados pelo fabricante, desta forma, assegurando segurança e funcionamento adequado das mesmas. Pode-se dizer, que estes circuitos funcionam como os reatores eletrônicos para lâmpadas fluorescentes, podendo ser embutidos ou separados no conjunto de luminárias, dependendo dos fabricantes (A. CARVALHO B. C., 2014). Em exemplo prático, uma placa de *LED* que possui 8 *LEDs* em série necessita de um *driver* que forneça 24V de tensão e 400mA de corrente contínuos. Considerando um *LED* padrão, geralmente necessita de 3V e 50mA passando por cada um.

Existem várias topologias utilizadas em circuitos de *drivers*, sendo bastante comum encontrar circuitos *buck*, *boost*, *buck-boost* ou até mesmo *flyback*, mas para este funcionamento deste sistema, a entrada do *driver* que realmente importa é a de PWM, que receberá o sinal do controlador e com isso, irá controlar a saída, modificando corrente ou realizando o desligamento do *driver*. Na figura 2.8 é mostrado o produto que é utilizado na empresa *Lumicenter*. Nela é possível verificar que em suas extremidades estão os conectores para entrada de sinal PWM (conector duplo na cor preto e vermelho) e ao lado, alimentação do produto (conector triplo nas cores cinza e verde). Esta figura é importante para melhor entendimento de como será a disposição dos cabos na ligação do sistema de iluminação.

FIGURA 2.8 – Driver *LED* Lumicenter

Fonte: (LUMICENTER LIGHTING, 2018)

O sinal PWM que o produto recebe do controlador proposto é tratado internamente e enviado ao circuito integrado interno, que realiza o controle destes pulsos e com isso, altera a frequência de chaveamento do componente MOSFET, que faz com que aumente ou diminua a relação de corrente que irá para o circuito de saída do tipo *buck-boost*.

2.9.2 CENÁRIOS DE ILUMINAÇÃO

Criar um cenário de iluminação consiste em configurar um conjunto de luminárias de um ambiente específico para fluxo e temperatura de cor específica afim de criar um ambiente com destaque diferenciado. Por exemplo, nas figuras 2.9, 2.10 e 2.11, é apresentado um mesmo ambiente, com as mesmas luminárias, só que com destaques diferentes para cada situação.

FIGURA 2.9 – Cenário 1 para ambiente "festa"

Fonte: (SCENARIO, 2014)

FIGURA 2.10 – Cenário 2 para ambiente "leitura"

Fonte: (SCENARIO, 2014)

FIGURA 2.11 – Cenário 3 para ambiente "relax"

Fonte: (SCENARIO, 2014)

Nestes três ambientes, os destaques são diferentes. No primeiro, chamado de "cenário festa", o destaque da iluminação está para itens decorativos do ambiente, como arandelas, mesas de centro e objetos num geral, com temperatura de cor baixa e fluxo luminoso com baixa intensidade.

No "cenário leitura", a temperatura de cor sobe, perto de 6000K, a distribuição do fluxo luminoso se torna mais uniforme e o ambiente cria um clima de atenção a quem estiver presente.

No "cenário *relax*", o destaque fica para iluminação indireta, com baixa temperatura de cor e baixo fluxo luminoso, indicada para horários matutinos e noturnos, onde este cenário gera a vista do consumidor, maior tranquilidade e relaxamento.

2.9.3 DIMERIZAÇÃO

Dimerização é a capacidade de controlar a intensidade de luz emanada pelo *LED*, ou seja, é possível aumentar ou diminuir a luminosidade do ambiente de acordo com a corrente que passa pelos *LEDs*. Dentre os valores do dimmer, estão a possibilidade de desenvolver ambientes decorativos mais agradáveis ou utilizar pensando somente em eficiência energética, pois com esta tecnologia é possível adequar a luz emitida pelas luminárias conforme horário do dia, mantendo sempre a mesma

luminosidade do ambiente, através da compensação de luz artificial por luz natural do sol.

Para realizar a variação da corrente de saída em *drivers* de *LED*, o sistema de dimerização não é mais por triacs como em lâmpadas comuns, mas agora cada tipo de sistema de dimerização possui suas particularidades e funções específicas. Os mais comum encontrados no mercado atualmente são os dimmers 1-10V, DALI e PWM.

0-10 ou 1-10V é um dos primeiros e mais simples sistemas de dimerização de iluminação eletrônica a LED. Nele, o sinal de controle é uma tensão contínua que pode ser variada de 0 a 10 volts.O *driver* de *LED* possui uma entrada exclusiva para este sinal e ao conectar o dimmer, a iluminação controlada deve escalonar sua saída de modo que, em 10V, a intensidade da luz deva estar em 100% de sua capacidade máxima, e a 0V a saída deva desligar ou dimerizar ao mínimo estabelecido pelo fabricante (geralmente 5%). Estes *drivers* devem responder a estas tensões em vários padrões, fornecendo curvas de saída lineares para saída de tensão, saída de fluxo luminoso ou controle pela potência. Os receptores geralmente tem uma impedância de entrada nominal de 100 a 20k Ohms, com 0,1 a 0,2 mW de potência em 10V, corrente de 30mA e frequência de 50 ou 60Hz (LUMICENTER, 2017). Um exemplo prático de utilização desta tecnologia é aplicar o sinal de 10V na entrada de dimerização num *driver* 0-10V com saída nominal de 1A, onde neste caso irá manter o valor padrão. Ao alterar a tensão contínua para 5V, ele deve dimerizar e mudar a corrente de saída para 500mA.

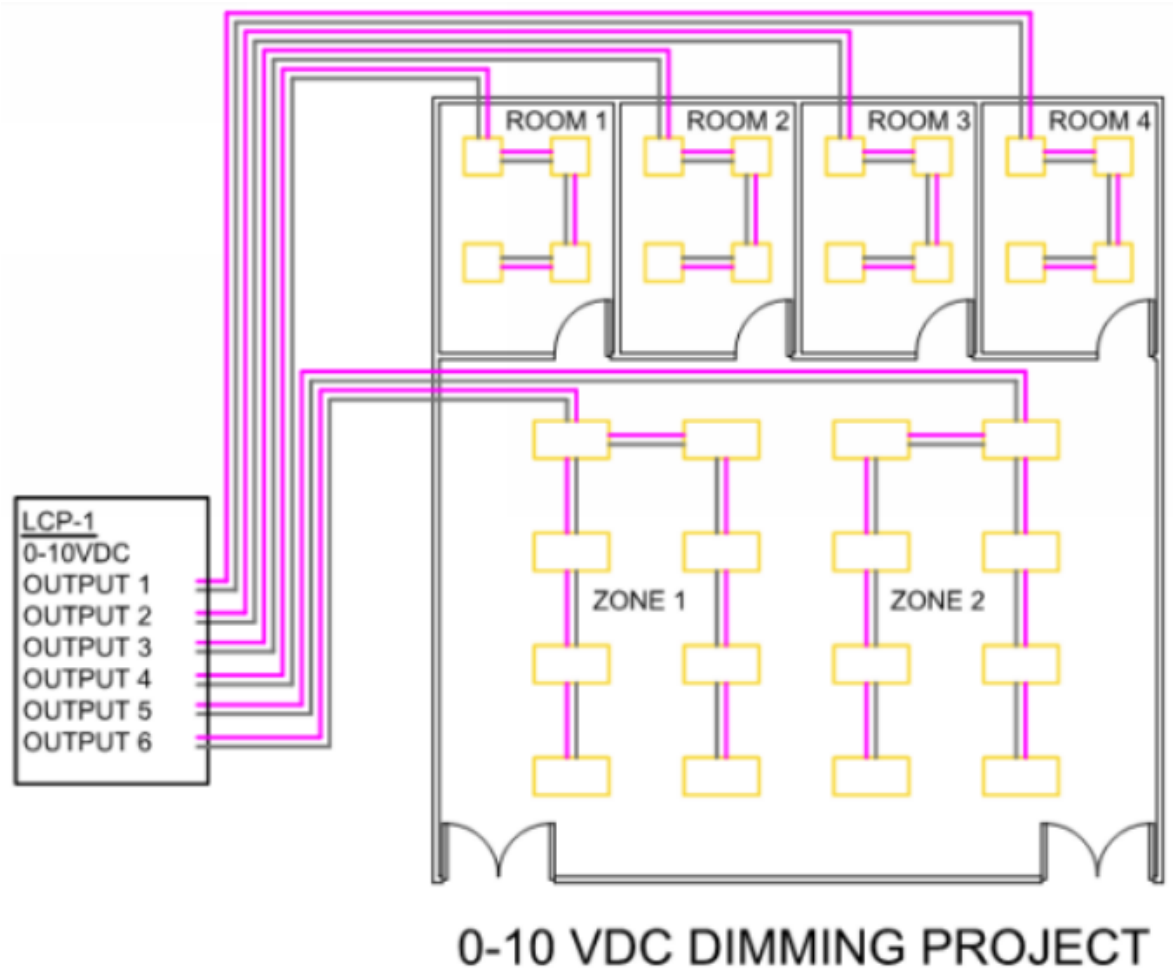
DALI é a abreviatura de "*Digital Addressable Lighting Interface*". Trata-se de um padrão internacional especificado pela norma IEC 60929 que independe dos fabricantes dos produtos, o que assegura a intercambialidade e a interoperabilidade de dispositivos dimerizáveis de vários fabricantes. Isto dá aos projetistas, fabricantes de luminárias, construtores, instaladores e usuários finais a segurança de várias fontes de fornecimento. (LUME, 2015).

DALI é um protocolo dedicado puramente ao controle de iluminação, ou seja, ele não pode ser usado para controlar outros sistemas em paralelo. Entretanto, é eficaz para a seleção de informações como, por exemplo, a situação de luminárias com defeito. Para isto, é bastante útil realizar interface com os sistemas da automatização de edifícios quando é necessário supervisão remota desta rede. Cada unidade na rede de DALI possui um endereço, consequentemente é possível comunicar-se diretamente com cada componente. Com somente um par do cabo de controle é possível controlar

diversos grupos diferentes de aparelhos(LUME, 2015).

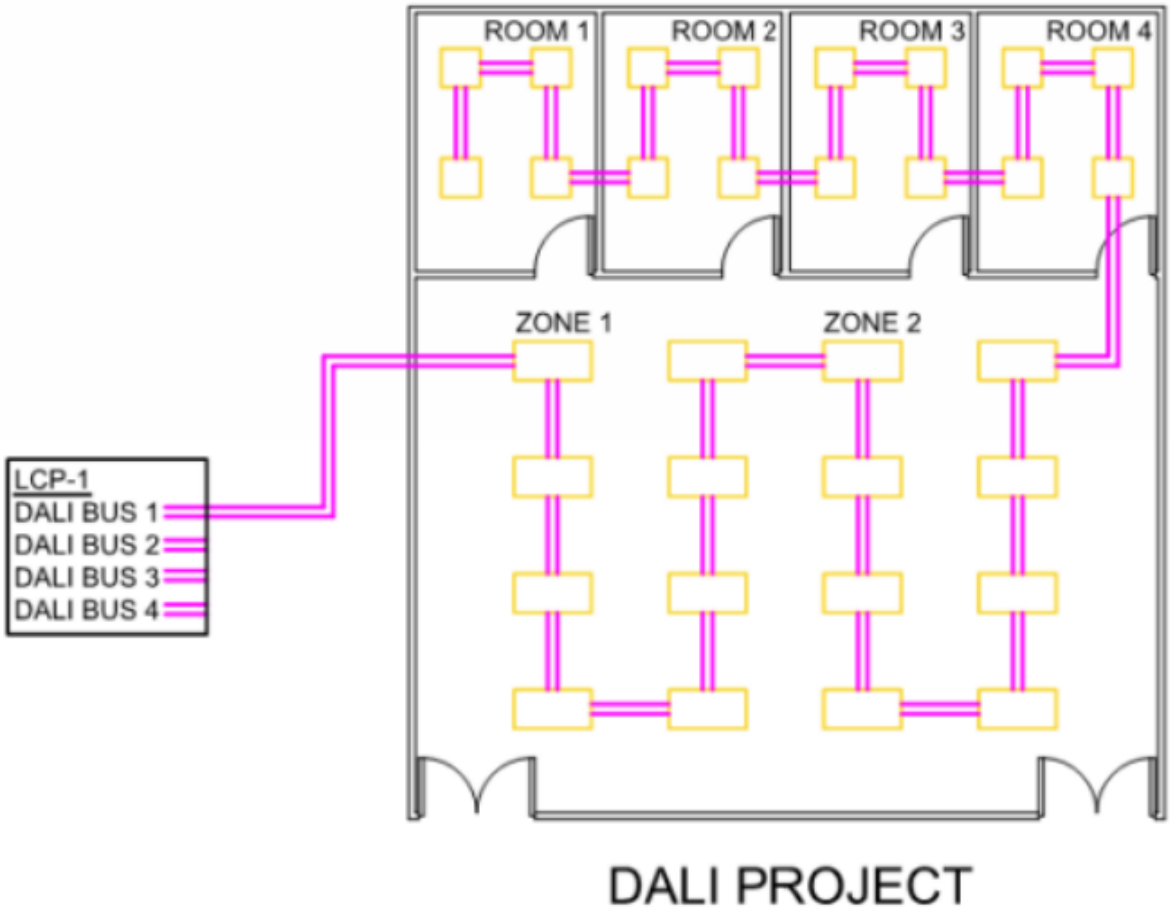
Nas figuras 2.12 e 2.13 estão apresentadas as principais diferenças entre os sistemas 0-10V e DALI, que é a maneira como devem ser instalados, mostrando que com um único par de cabos é possível realizar a conexão de todas as luminárias no sistema DALI.

FIGURA 2.12 – Instalação de sistema 0-10V



Fonte: (BEGCONTROLS, 2014)

FIGURA 2.13 – Instalação de sistema DALI



Fonte: (BEGCONTROLS, 2014)

Ao contrário do sistema 0-10V que necessita de uma fonte de tensão contínua para realizar o controle do ambiente ou no sistema DALI que funciona com dimmer e controlador, onde se comunicam através de sinais digitais e enviam os mesmos para as luminárias, a solução proposta pela equipe funciona puramente através de pulsos PWM, emitidos pelo conversor de sinal montado *hardware*. A vantagem de adquirir este sistema é que inibe a utilização de dimmer físico como em outras soluções apresentadas, deixando o sistema mais barato, pois utiliza o próprio *smartphone* e mais acessível, não havendo necessidade de locomoção até o interruptor.

2.9.4 TECNOLOGIA *TUNABLE WHITE*

Outro diferencial do projeto é poder utilizar a tecnologia *tunable white* nas luminárias, a partir de luminárias com LEDs específicos e desenvolvimento de algoritmo no aplicativo apresentado.

Luminárias que possuem esta tecnologia podem realizar o ajuste da tempe-

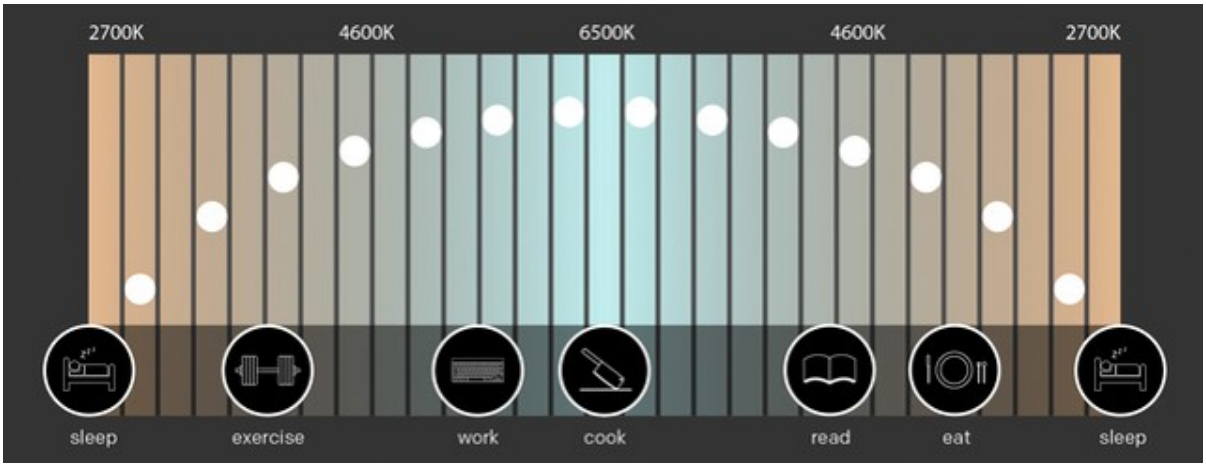
ratura de cor da luz emitida pelos *LEDs*. Este é um avanço tecnológico que segundo revistas como *LEDs Magazine - 2017*, é uma tendência mundial. Esta capacitação do sistema em mudar a cor da iluminação traz diversas vantagens e agregação de valor ao sistema adquirido pelo usuário. O *tunable white*, ou tecnologia que permite a mudança da temperatura de cor, é um tema que está em ascendente discussão e em alta por diversos fabricantes de luminárias, inclusive, foi o tema vencedor do Nobel de medicina no ano de 2017 (GLOBO, 2017).

A regulação dos horários do sono é o exemplo mais óbvio da importância do ritmo circadiano para o funcionamento do organismo humano. Ao longo de um período aproximado de 24 horas, genes regulam a liberação do hormônio melatonina. Durante a noite, no escuro, os níveis aumentam e, com a claridade, baixam. Mas o ciclo circadiano e relógio biológico — tema que rendeu a três pesquisadores americanos o Nobel de Medicina do ano 2017 — são responsáveis pelas variações diárias no metabolismo e sua alteração crônica está relacionada com graves doenças, incluindo o câncer (GLOBO, 2017).

> Experimentos com animais mostram que a destruição do relógio biológico é incompatível com a sobrevivência, não por acaso a Academia resolveu premiar o trio americano — comentou o doutor em neurociência Fernando Mazzilli Louzada, professor do Departamento de Fisiologia da UFPR. — Pensar em sono é o mais evidente, mas hoje já sabemos, por exemplo, que situações crônicas de alteração no ritmo circadiano estão relacionadas com aumento no risco de câncer de mama (GLOBO, 2017).

Na figura 2.14, são mostradas as temperaturas de cor do ciclo circadiano em relação ao período do dia e a posição do sol numa região. Nesta figura é possível ver as diferenças de temperatura de cor e é possível constatar que nos horários da manhã e no final da tarde, o sol apresenta cores quentes, chegando em 2700 Kelvin, e que em horários mais próximos do meio dia, esta temperatura é considerada fria, variando entre 4600 e 6500 Kelvin. 6500K é considerada a temperatura de cor mais fria do sol e acima disso, a cor passa a apresentar tonalidades de azul.

FIGURA 2.14 – Representação do ciclo circadiano em função das horas do dia



Fonte: (BRAUN, 2017)

As cores amareladas do ciclo do sol, consideradas temperaturas de cor quentes, são vistas na parte do amanhecer e entardecer. É esta temperatura de cor que auxilia no sono pela parte da manhã, fazendo acordar com maior qualidade e ajudam o corpo a reconhecer o horário do final do dia, para completar o ciclo. Recomenda-se para momentos de lazer, relaxamento e ambientes de festa. Já as cores brancas, ou cores frias, são ideais para ambientes que requerem atenção, como escritórios, cozinhas, hospitais e para realizar leituras num geral.

3 DESENVOLVIMENTO

Para melhorar a organização da equipe em relação a tempo, prazos e divisão eficiente de tarefas, foi elaborado um cronograma detalhado com as etapas a serem seguidas, até a conclusão do projeto final em questão. É notável a eficiência na gestão de tarefas e prazos quando é elaborado um cronograma detalhado das atividades. O mesmo foi desenvolvido via planilha do *software* Excel e pode ser visto no apêndice A, na figura 6.1.

O cronograma foi dividido com base em discussões entre integrantes do grupo e professor orientador. As três etapas estão destacadas devido as entregas previstas pelo calendário oficial da universidade.

3.1 DEFINIÇÃO DO PROJETO

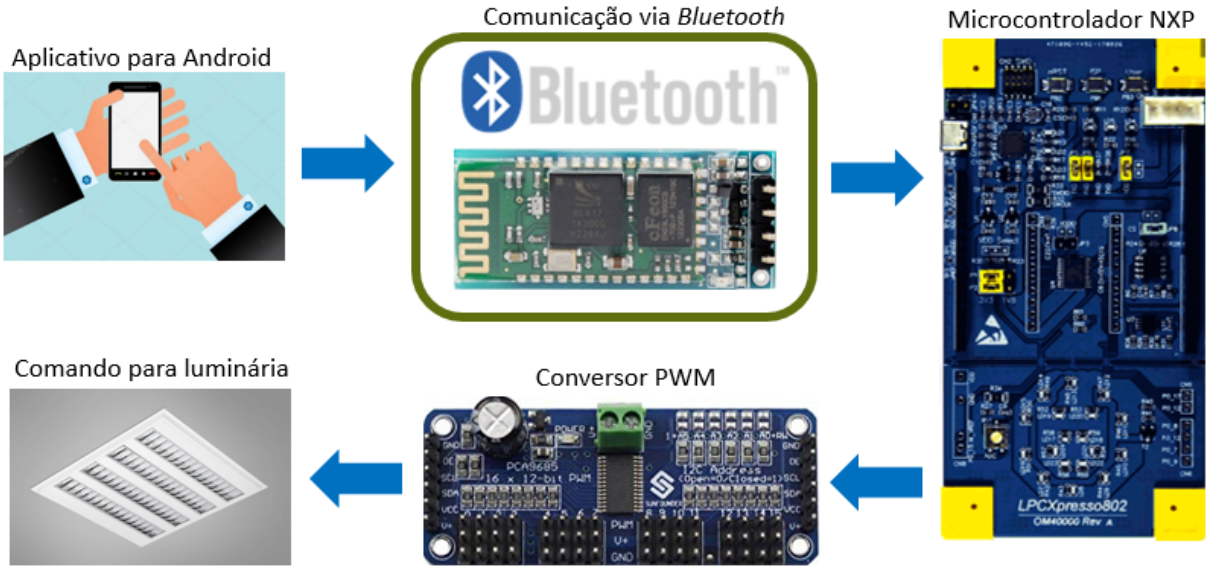
Seguindo as tendências e demandas no mercado de iluminação, as quais foram apresentadas na feira internacional da indústria da iluminação - EXPOLUX 2018 (a maior feira do setor em território nacional). Houve a necessidade de criação desse projeto e consecutivamente a sua pré-apresentação na feira. Seguindo a tendencia no controle e adaptação dos sistemas de iluminação via aplicativo para *smartphone*, realizado pela empresa Grupo Lumicenter Lighting. Diante de projetos de iluminação cada vez mais personalizados e que atendam as necessidades e particularidades dos clientes. Esse projeto possui como funções especificas a criação de cenários, dimerização de fluxo luminoso e mudança da temperatura de cor das luminárias no ambiente, que é de suma importância para o desenvolvimento tecnológico dos sistemas de iluminação das empresas em questão.

A definição dos componentes foi uma tarefa crucial no desenvolvimento do projeto e está discriminada em seções deste capítulo, para que os resultados obtidos a partir do circuito eletrônico montado estejam bem fundamentados.

Para a melhor visualização e entendimento do desenvolvimento deste projeto, na figura 3.1 está o diagrama da implementação deste sistema, mostrando o inicio da aplicação, a qual também é a interface de acesso do usuário, através do *smartphone* e comunicação via *bluetooth*, passando pelo microcontrolador, que transmite ao módulo

conversor PWM, o qual transmite os sinais para o *driver* de *LED* posicionado na luminária.

FIGURA 3.1 – Esquemático do *Hardware* implementado



Fonte: (OS AUTORES, 2018)

3.2 DEFINIÇÃO DO PROFESSOR ORIENTADOR

A ideia foi apresentada ao professor especialista em protocolos de comunicação sem fio Dr. Marcelo Eduardo Pellenz, o qual mostrou interesse pelo assunto proposto e aceitou orientar-nos nesse desafio.

3.3 MÓDULO *BLUETOOTH* HC-06

O módulo *bluetooth* escolhido para este projeto foi o HC-06, que é um componente de fácil acesso, baixo custo e de fácil configuração. É este dispositivo que fará a comunicação entre o celular ou *tablet* com o microcontrolador. A configuração deste módulo se dá por comandos AT, enviados através de comunicação serial com o computador. O parâmetro que deve ser configurado é a velocidade de comunicação (*baud rate*), que neste caso é de 9600. Nesta mesma etapa é inserido o nome do módulo que será encontrado pelo aplicativo desenvolvido e sua senha de acesso, para evitar conexões indesejadas.

Este dispositivo funciona de modo escravo, ou seja, outros dispositivos é que devem se conectar nele. Fisicamente ele possui quatro pinos de conexão, o transmissor

e receptor (TX e RX) são ligados no receptor e transmissor do microcontrolador, respectivamente. Os outros dois pinos são da alimentação do módulo, conforme mostrado na figura 3.2, deve ser alimentado com 3,3V, que vem de alimentação secundária fornecida pelo microcontrolador.

FIGURA 3.2 – Módulo bluetooth HC-06

Fonte: (OS AUTORES, 2018)

As especificações deste módulo, segundo *datasheet* do fabricante são as seguintes:

- Versão do *bluetooth* 2.0: Atende as especificações de todos os aparelhos *smartphones* e *tablets* utilizados no projeto;
- Tensão de funcionamento 3,3 até 5V com corrente 50mA: Valores fáceis de serem encontrados nas saídas do próprio microcontrolador, o que facilita as ligações elétricas;
- Taxa de transmissão de 2 Mbps: Taxa padrão para este tipo de comunicação, o que permite ao usuário não conseguir identificar *delay* significativo na transmissão de dados;
- Frequência de operação de 2,4 GHz: Frequência padrão para estes dispositivos e compatível com o restante do projeto;
- Modulação GFSK *(Gaussian Frequency Shift Keying)*: Modelo de modulação simplificado e padrão para dispositivos *bluetooth* de baixo custo;
- Suporte para perfil escravo: Será utilizado como modo escravo, portanto suporta as necessidades do projeto;

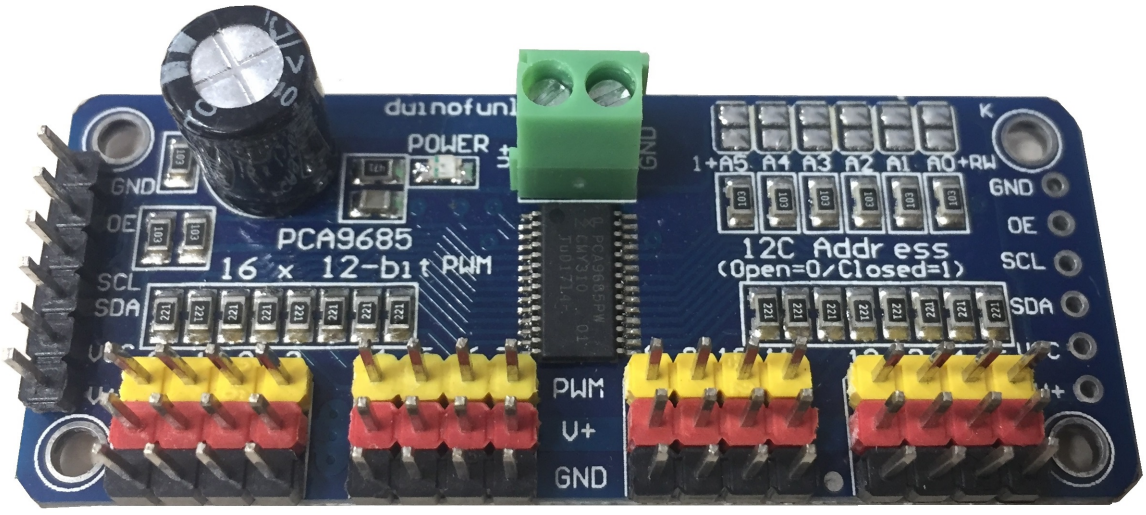
- Conexão através de quatro pinos: TX, RX, VDD e GND: Conexão simplificada e de fácil manuseio, atendendo ao requisito de portabilidade do controlador;
- Dimensão do produto 4,3 cm de comprimento, 1,6 cm de largura e 0,7 cm de altura: Tamanho analisado e viável em aplicações comerciais.

3.4 MÓDULO CONVERSOR PWM PCA9685

O módulo conversor modelo PCA9685 é um controlador *LED* de 16 canais controlado por barramento do tipo I2C. Cada uma de suas saídas tem seu próprio controlador PWM individual de frequência fixa de resolução 12 bits (4096 passos) que opera com frequência programável podendo variar de 40 a 1000 Hz, contendo um ciclo de trabalho que é ajustável de 0 a 100% para permitir que os *LEDs* na saída possuam valor de brilho específico.

Dentre suas principais vantagens, podemos destacar a possibilidade de configurar cada saída individualmente, podendo deixar ativada ou não; Este controlador configura a mesma frequência de operação PWM para todas as suas saídas, facilitando sua configuração. Suas saídas podem atuar de modo cascata, podendo ligar outras transformar uma saída em 16 de mesmo sinal, totalizando até 256 luminárias controladas pelo sistema. Seu tamanho fisicamente é uma vantagem, pois consegue oferecer diversas funções em tamanho mínimo, como visto na figura 3.3

FIGURA 3.3 – Módulo conversor PWM PCA9685



Fonte: (OS AUTORES, 2018)

Através de seu *datasheet*, retirado do *site* da fabricante NXP, tem-se como

tópicos as principais características deste controlador:

- Interface de barramento I2C compatível com *Fast-mode* de 1 MHz: Comunicação padrão via protocolo I2C, que será utilizada no *hardware*;
- Alteração dos estados pode ser programável para atualização das saídas *byte* a *byte* ou todas ao mesmo tempo: Recurso crucial para aplicabilidade de tempo real do projeto;
- Possui 6 pinos de endereçamento de *hardware*: É possível configurar até 62 dispositivos PCA9685 no mesmo barramento I2C, aumentando a possibilidade do total de sinais enviados;
- O recurso de reinicialização do *software* permite que o dispositivo seja reinicializado pelo barramento I2C (vindo do microcontrolador).

Para realizar a ligação deste módulo conversor no sistema, utiliza-se a saída auxiliar para alimentação do microcontrolador (3,3V), e através do protocolo de comunicação I2C, é recebido o sinal para cada umas de suas 16 saídas e cada saída é ligada diretamente a luminária de *LED*.

3.5 MICROCONTROLADOR ARDUINO E NXP

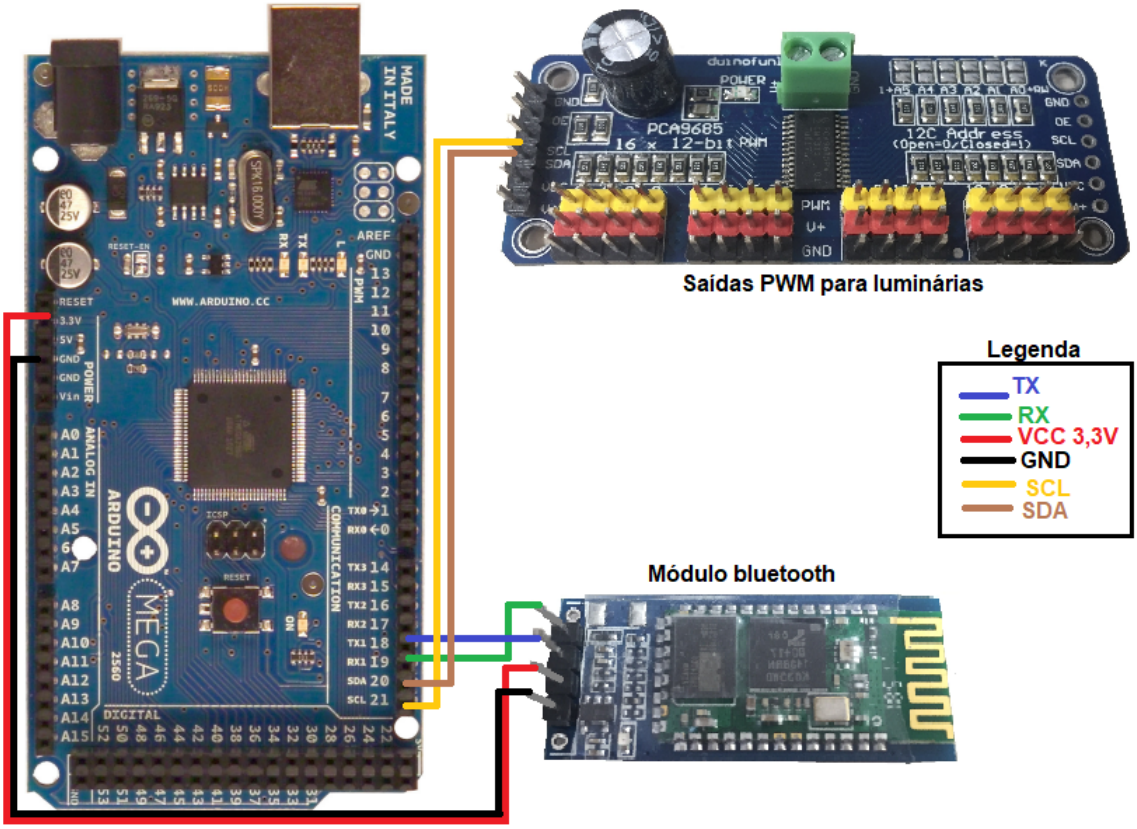
Como um dos objetivos desse projeto, é montar um produto, possível de ser comercializado em território nacional e que carregue a logo da empresa patrocinadora Lumicenter. O trabalho foi iniciado com um microcontrolador de uso geral, acessível, e de fácil implementação e com a possibilidade de portabilidade para outros controladores, de padrão industrial, portanto, no protótipo foi utilizado um Arduino MEGA.

O Arduino é uma plataforma de prototipagem eletrônica *open-source* que se baseia em *hardware* e *software* flexíveis e de fácil utilização. É destinado principalmente para o desenvolvimento inicial de protótipos de baixo-custo. O microcontrolador na placa é programado em linguagem de programação nativa do próprio Arduino, baseada na linguagem *wiring*, e o ambiente de desenvolvimento Arduino, baseado no ambiente *processing*. Os projetos desenvolvidos com o Arduino podem ser autônomos ou podem comunicar-se com um computador para a realização da tarefa (ARDUINO, 2018).

O desenvolvimento com o Arduino MEGA se deu pois suas especificações atendem aos requisitos do projeto, quando tratando de prototipagem inicial. Ele possui o microcontrolador ATmega2560, opera em tensão de 5V, pode ser ligado com fonte

simples de 12V, possui 54 pinos I/O, 256kB de memória *flash*, 8kB de SRAM e 4kB em EEPROM. A ligação do Arduino no sistema de controle projetado foi dada conforme figura 3.4.

FIGURA 3.4 – Esquemático do circuito com microcontrolador Arduino



Fonte: (OS AUTORES, 2018)

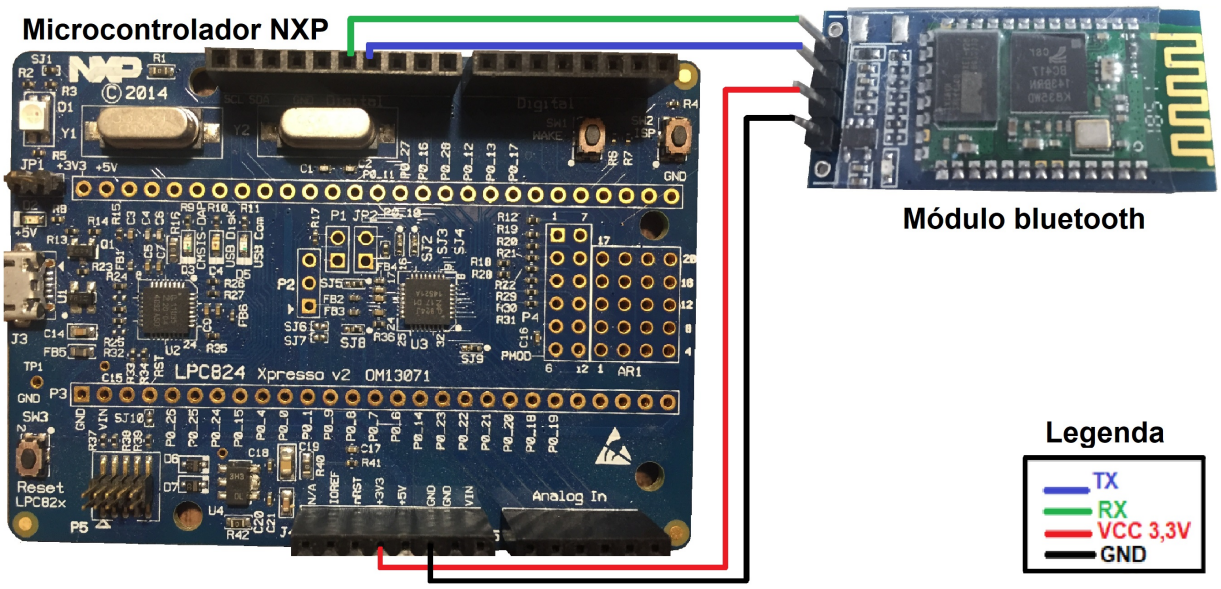
Para realizar a escolha do sistema eletrônico embarcado que irá substituir o arduino no futuro do projeto, foram levados em conta vários requisitos básicos que atendam ao desenvolvimento do projeto. O custo por unidade, um fornecedor com o *lead time* curto para entrega dos produtos, facilidade de prototipagem, suporte aos protocolos de comunicação, bibliotecas disponíveis para desenvolvimento e possibilidade de utilizar gravador do *chip* microcontrolador, afinal de contas deve ser considerada a produção em massa destes produtos. Avaliando estes pré requisitos, o microcontrolador do fabricante NXP, da família que possui arquitetura ARM, LPC82X, mais especificamente o LPC824 foi o escolhido.

Este microcontrolador possui 32 bits baseado em arquitetura ARM Cortex-M0+ de baixo custo, com frequência de CPU chegando em até 30MHz. Ele suporta até

32KB de memória flash e 8kB de SRAM, possui também quatro barramentos I2C e três USARTs.

A principal motivação para escolher a substituição do Arduino para o microcontrolador da NXP é que para o desenvolvimento de soluções corporativas, onde o cliente utilizará estes produtos em condições adversas por tempo indeterminado, a demanda de equipamentos eletrônicos mais resistentes e principalmente com maior confiabilidade. Como melhoria, é possível substituir o Arduino e o conversor PWM do protótipo inicial apenas pelo microcontrolador NXP e módulo *bluetooth*. Na figura 3.5 é mostrado o esquemático de ligação quando alterado para microcontrolador do fabricante NXP.

FIGURA 3.5 – Esquemático do circuito com microcontrolador NXP



Fonte: (OS AUTORES, 2018)

3.6 APLICATIVO ANDROID

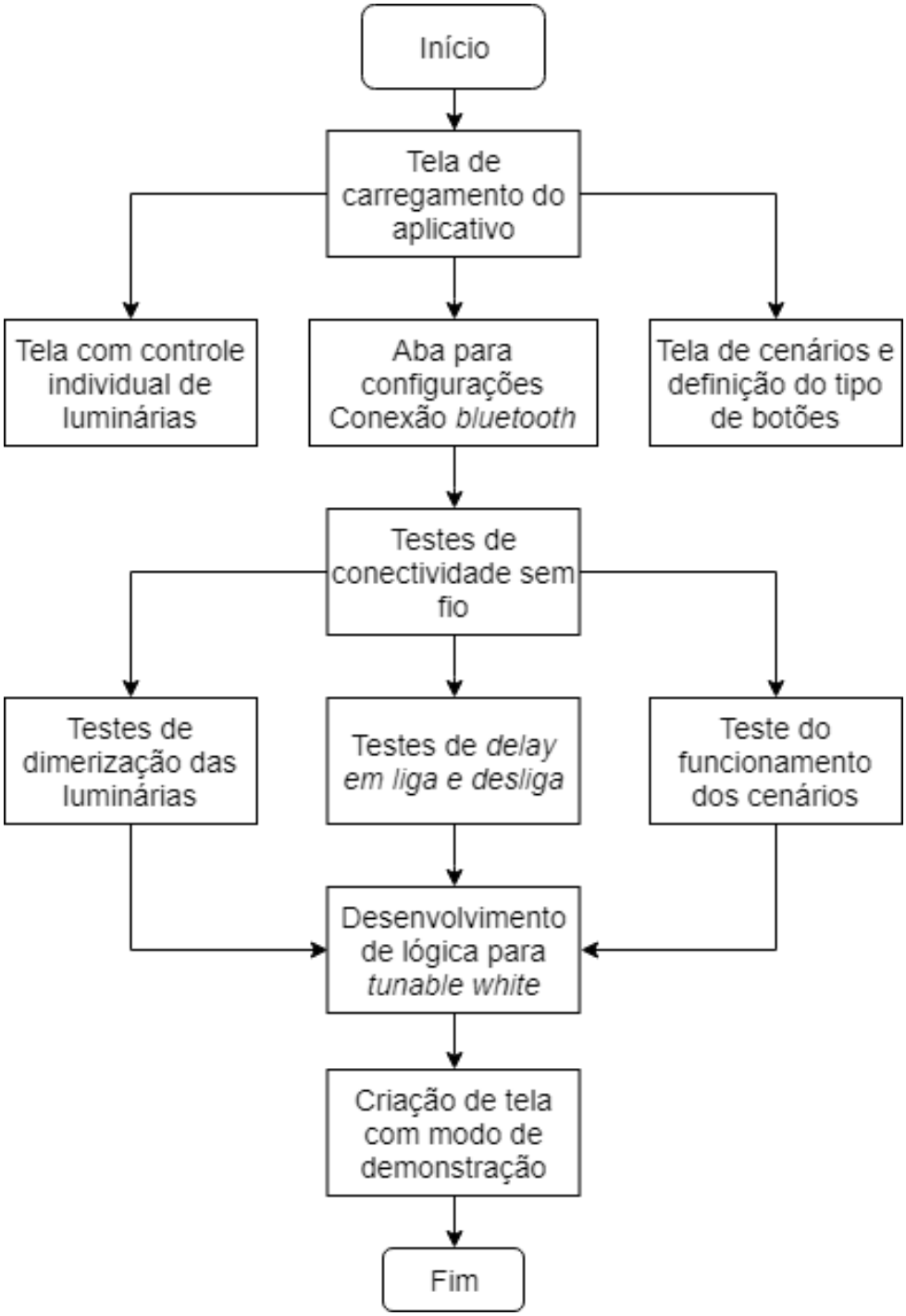
O aplicativo para o sistema operacional Android é o grande desafio deste projeto, devido a sua complexibilidade e necessidade em aliar telas que atraiam usuários pelo *design* agradável aliado a funcionalidades que agreguem valor para o sistema de iluminação num geral. Este aplicativo foi desenvolvido via Android Studio, que é o *software* ambiente de desenvolvimento integrado (IDE) nativo de desenvolvimento dos aplicativos.

O desenvolvimento começou com as telas iniciais do programa (interface do

usuário), onde foram investidas horas de trabalho, para garantir um *layout* agradável e da fácil utilização ao usuário. A segunda tela desenvolvida foi a de controle individual das luminárias ligadas ao controlador, a terceira tela é a de controle dos cenários de iluminação desenvolvidos, após estas telas principais, a aba de controle para acessar as ferramentas e conexão da comunicação sem fio *bluetooth* foi aperfeiçoada, pois após esta etapa, começaram os testes de conectividade e de funcionalidade das aplicações.

Toda a programação realizada foi na linguagem de programação Java, e por orientação da equipe de desenvolvimento da empresa *Lumicenter Lighting*, foi definido que as telas estarão dentro de apenas uma *activity*, ou seja, ao realizar mudança de tela, não é necessário realizar a troca dessas *activitys* e sim apenas sobrescreve-las, para facilitar a comunicação das informações dentro do aplicativo e não ocasionar perda de pacotes do *bluetooth*. Na figura 3.6, é mostrado o diagrama com as etapas de desenvolvimento do aplicativo Android, melhorando a visualização e compreensão de como foi realizado.

FIGURA 3.6 – Diagrama sequencial para desenvolvimento do aplicativo



Fonte: (OS AUTORES, 2018)

3.6.1 *LAYOUT*

Conforme apresentado na figura 3.6, as telas foram sendo desenvolvidas uma a uma, começando pelas suas funcionalidades e implementando botões e controladores que agradem ao usuário final. Cada tela após o seu desenvolvimento, foi analisada e aprovada pela equipe de *marketing* da empresa *Lumicenter*. O *layout* começou com a

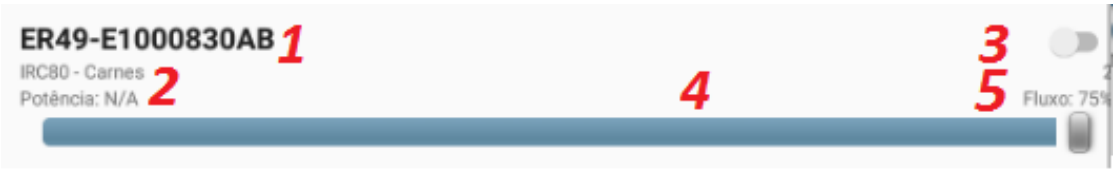
tela de inicialização, que precisava apresentar a logomarca do produto com seu nome que ainda é confidencial, devido a sigilosidade que tem este projeto até seu lançamento oficial como produto. Na figura 3.7, é mostrado o nome do projeto na abertura do aplicativo pelo usuário.

FIGURA 3.7 – Nome do projeto apresentado na tela inicial

Fonte: (OS AUTORES, 2018)

A segunda tela desenvolvida começou com o posicionamento de cada luminária individualmente, focada na funcionalidade para iniciar os testes com o aplicativo se comunicando. Ela deve possuir a chave de liga/desliga das luminárias, a barra de dimerização do fluxo luminoso, variando de 5 a 100% da corrente que o *driver* disponibiliza para sua saída, o nome ou modelo da luminária na posição e informação da potência que está consumindo no momento. A figura 3.8 possui o posicionamento de cada item mencionado e abaixo dela, a legenda.

FIGURA 3.8 – Configurações da luminária na tela de controle individual



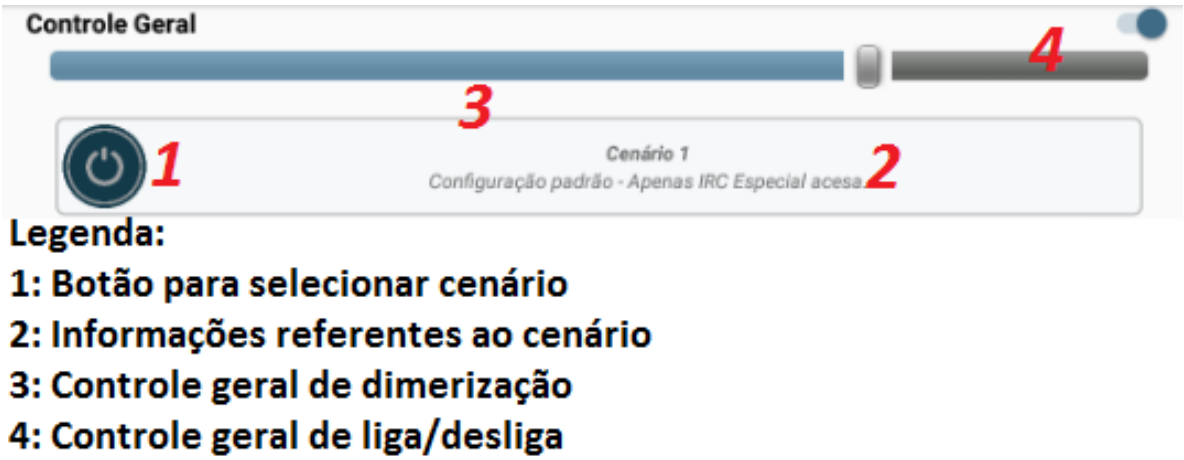
Legenda:

1: Nome atribuido a luminária
2: Descritivo da luminária
3: Chave liga/desliga
4: Barra de dimerização
5: Medidor de fluxo luminoso

Fonte: (OS AUTORES, 2018)

Na terceira tela desenvolvida, que podemos considerar como a tela principal do aplicativo, constam os cenários de iluminação criados, onde cada cenário possui características especificas de iluminação, por exemplo, cenário 1 irá ligar as duas primeiras luminárias, desligar as próximas duas e dimerizar em 50% do fluxo o restante, o cenário 2 se dá por outra configuração e assim sucessivamente. Na figura 3.9, é realizado um descritivo de cada item individual do cenário e mostra a barra de controle geral com chave liga/desliga de todas as luminárias. Como esta é a tela principal, foi sugerido pela equipe de *marketing* a inserção deste controle geral, caso o usuário queira mudar a iluminação como um todo sem sair desta tela.

FIGURA 3.9 – Botão de cenário e configuração geral



Fonte: (OS AUTORES, 2018)

Na barra lateral são mostrados os itens em que é possível acessar o aplicativo, mostrando o início que é a tela de cenários, o grupo de luminárias, que seleciona cenários com possibilidade de alteração dos parâmetros, tela com configuração individual das luminárias, ferramentas onde é possível verificar *status* do *bluetooth* e o comando para habilitar o *bluetooth* do dispositivo. Na figura 3.10 estão as informações principais citadas neste texto de maneira ampliada para melhor visualização do *layout* desenvolvido.

FIGURA 3.10 – Aba lateral com telas que podem ser acessadas e configurações

Fonte: (OS AUTORES, 2018)

4 RESULTADOS

No inicio do projeto foi realizada a análise do escopo em conjunto com o embasamento adquirido por todo o conteúdo da fundamentação teórica e levantamento de requisitos obrigatórios. Tendo em foco tudo o que foi apresentado neste projeto, a equipe desenvolveu um modelo de *layout* para cada uma das telas que realizam a interface entre homem e máquina.

Baseando-se nos modelos de interface homem-máquina propostos anteriormente no desenvolvimento desse projeto e com os conhecimentos adquiridos ao decorrer da graduação, relacionados à programação orientada a objeto (Java). Foram desenvolvidas todas as telas do aplicativo, que poder ser visualizadas no Apêndice B, entre as figuras 7.1 e 7.5. Essas telas tiveram seu *Layout* desenvolvido no software Android Studio, e todas as telas passaram por revisão e aprovação da equipe de *marketing* da empresa *Lumicenter*

A partir dos modelos de tela desenvolvidos nas etapas anteriores, começou o trabalho de configuração da comunicação sem fio, fato que nos permitiu dar continuidade no aplicativo com a configuração dos botões, barras de dimerização e testes para verificar funcionalidades do aplicativo de modo geral. A *activity* principal desenvolvida é mostrada no Apêndice E.

Antes de instalar as luminárias no sistema de controle, foram realizados testes de transmissão de dados *bluetooth* e dos sinais de saída PWM. Estes dados foram coletados por um tempo determinado, afim de reunir informações suficientes para constatar o bom funcionamento de todasas funções do produto. No Apêndice C, esta uma parte dos dados *bluetooth* que foram registrados e salvos em forma de dados de *MAC address* para mostrar os pacotes recebidos e perdidos. Como fica inviável inserir no Apêndice C todos os pacotes enviados, foram transformados em números o número total de pacotes enviados, recebidos e índice de falhas. Conforme figura 4.1, os pacotes foram transmitidos entre *tablet* e módulo *bluetooth* e analisados pelo próprio Arduino MEGA.

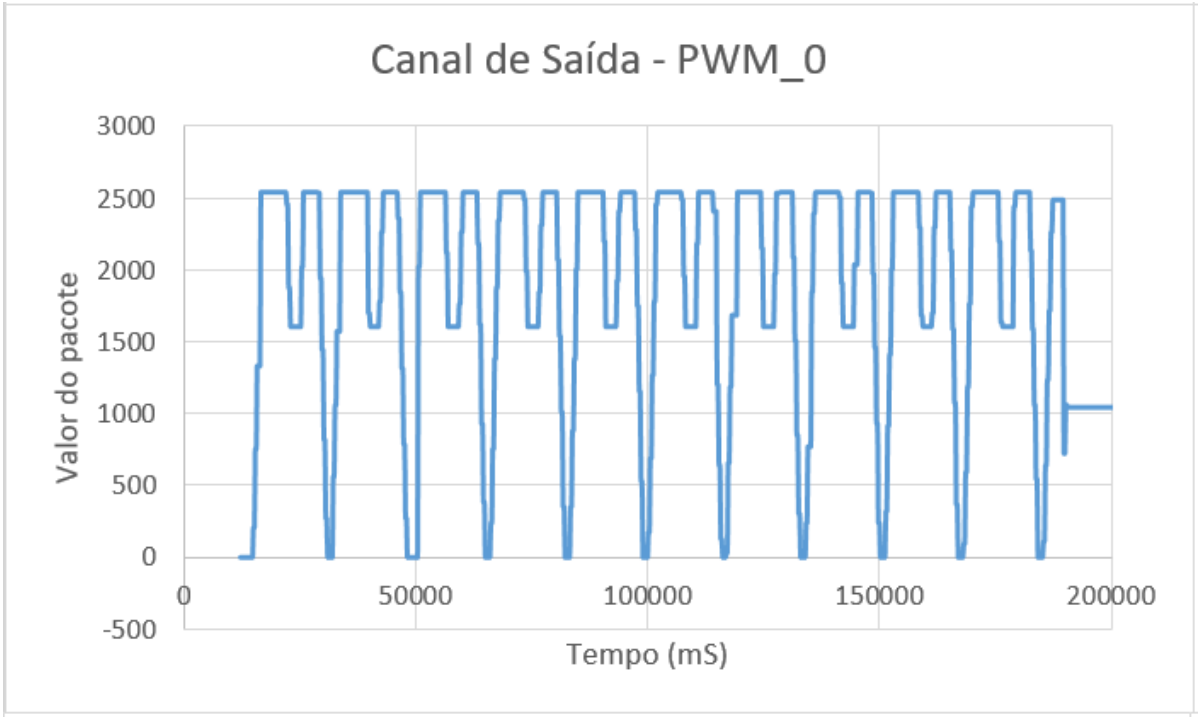
FIGURA 4.1 – Análise de pacotes de dados recebidos via *bluetooth*

Pacotes recebidos	**4452**
Pacotes perdidos	**359**
Indice de falhas	**8,06%**

Fonte: (OS AUTORES, 2018)

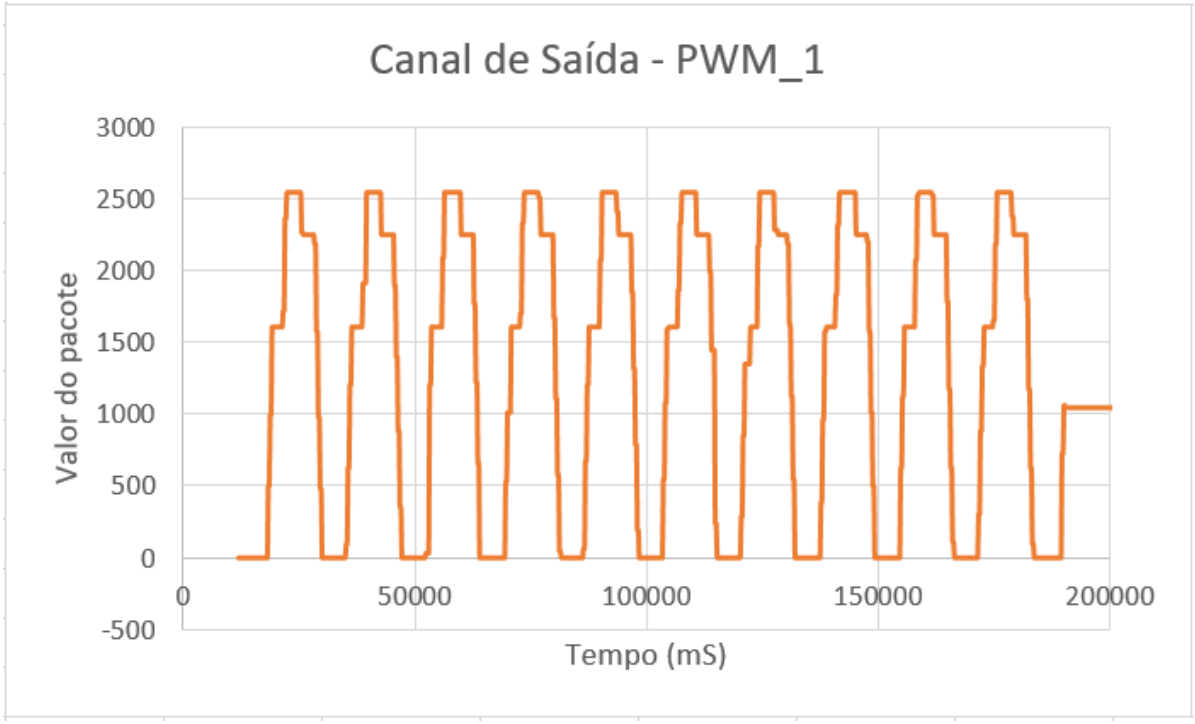
Do mesmo modo que foi realizada a análise dos dados enviados via *bluetooth*, também foi realizado o registro dos dados nas saídas PWM. Esses valores foram convertidos em gráficos e apresentam as formas de saídas diferentes de acordo com a interação do usuário. Foram registrados todos os 4452 pacotes enviados via *bluetooth* e separados conforme sua respectiva saída. Esses pacotes podem ser visualizados na tabela apresentada no Apêndice D. Como não é viável inserir todos os dados que foram registrados, foi realizado de forma gráfica a demonstração dos dados nestas quatro saídas, nas figuras 4.2, 4.3, 4.4 e 4.5.

FIGURA 4.2 – Dados enviados por pacotes na saída 0 PWM



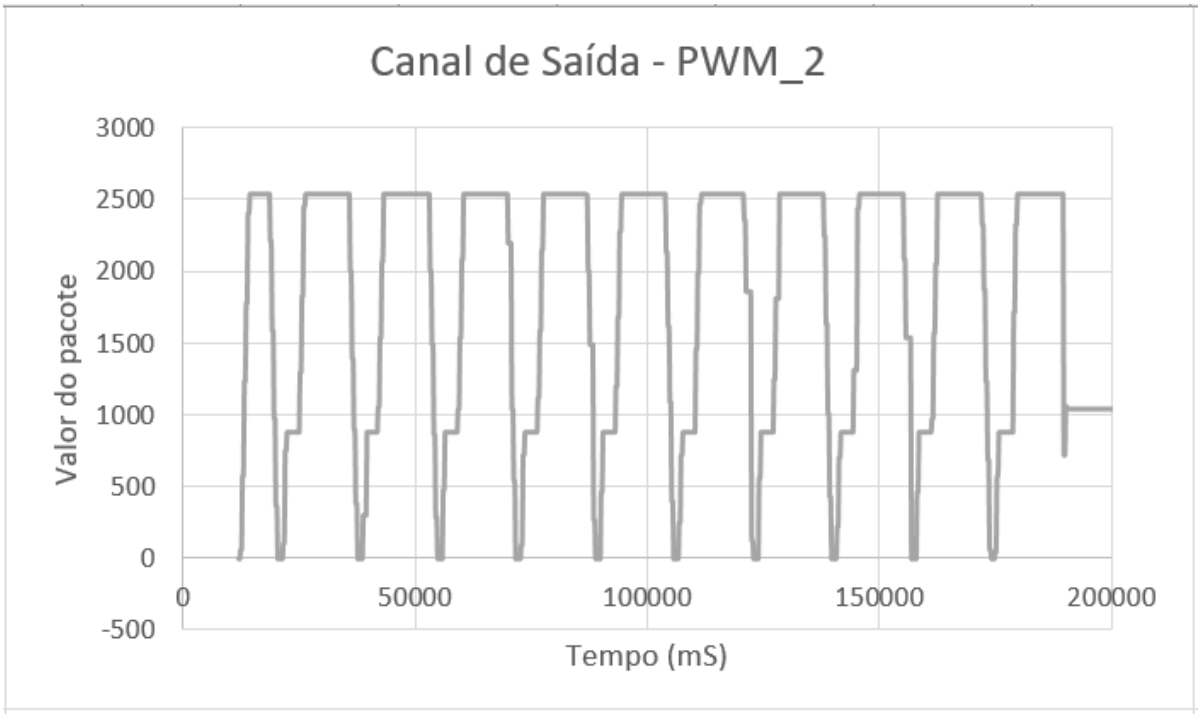
Fonte: (OS AUTORES, 2018)

FIGURA 4.3 – Dados enviados por pacotes na saída 1 PWM



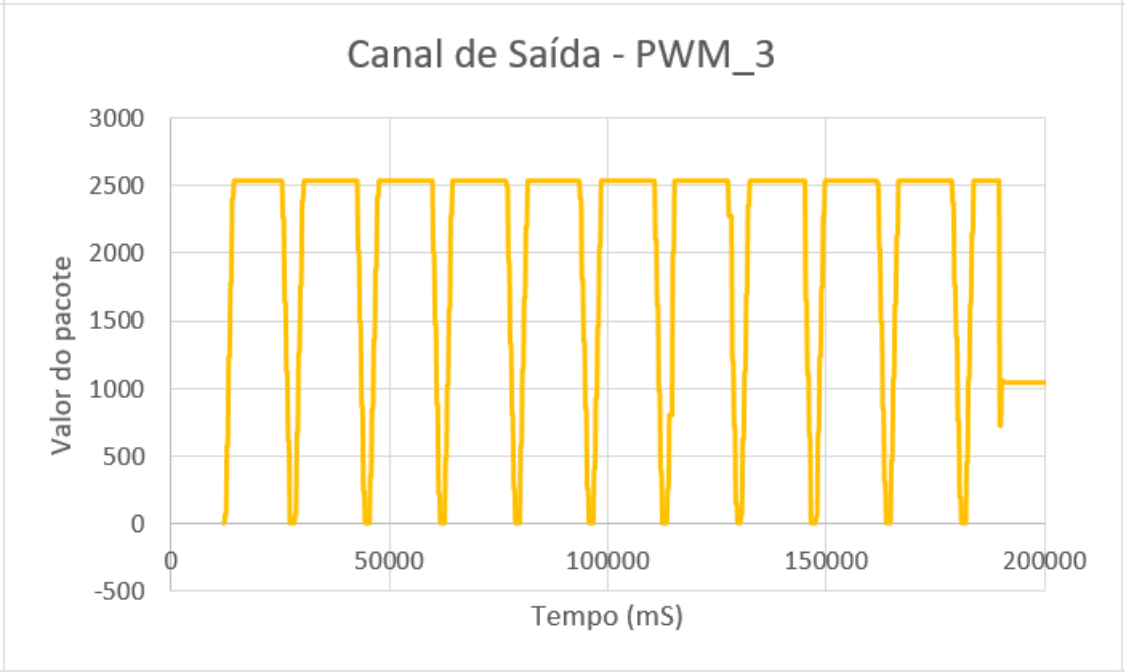
Fonte: (OS AUTORES, 2018)

FIGURA 4.4 – Dados enviados por pacotes na saída 2 PWM



Fonte: (OS AUTORES, 2018)

FIGURA 4.5 – Dados enviados por pacotes na saída 3 PWM



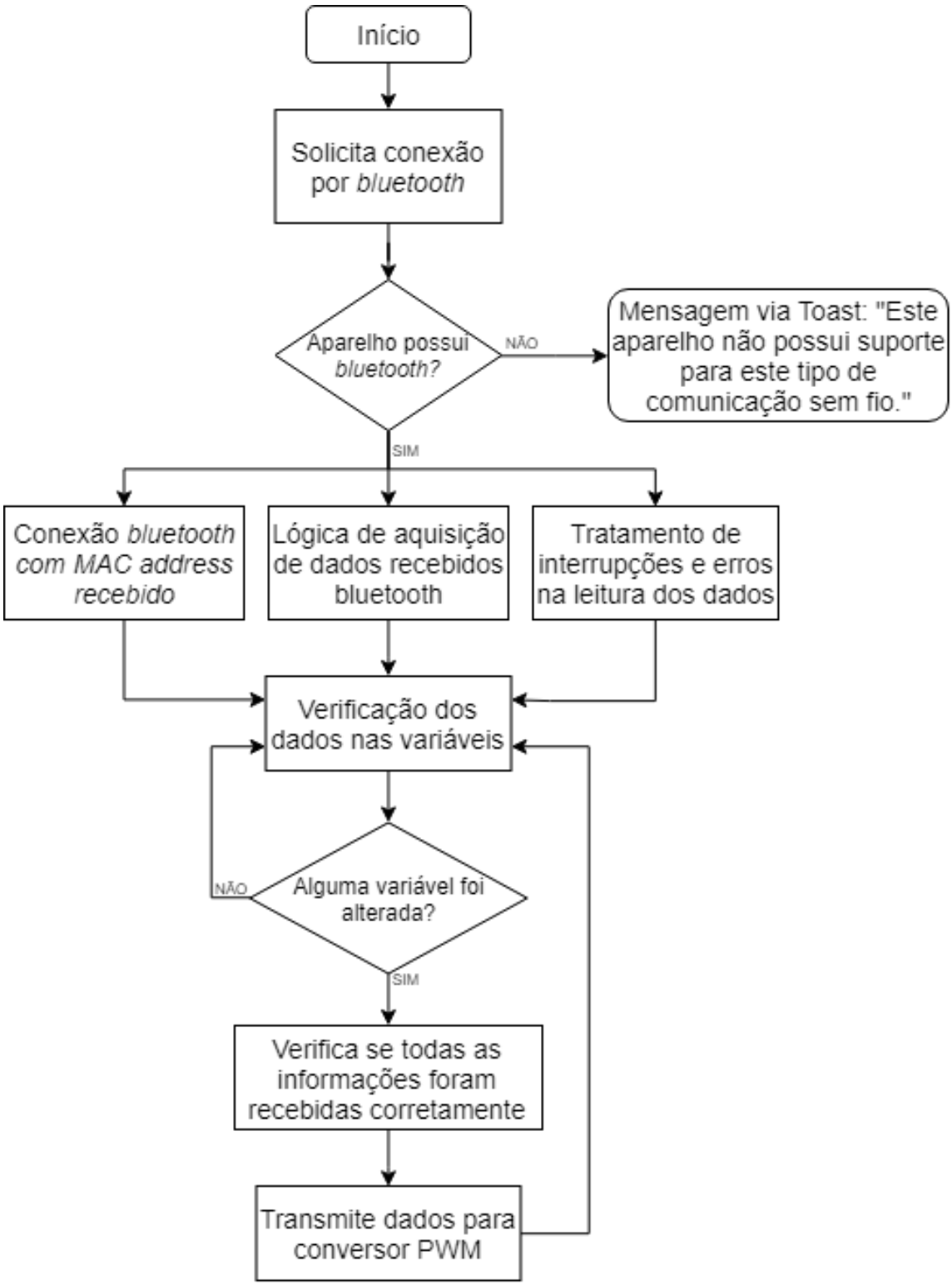
Fonte: (OS AUTORES, 2018)

O manifesto foi a primeira classe considerada na criação do aplicativo, pois todo desenvolvimento para Android precisa ter um arquivo *AndroidManifest.xml* no diretório raiz. Este arquivo possui as informações essenciais sobre o aplicativo desenvolvido em um sistema Android, necessárias para o sistema realizar sua configuração inicial, antes que ela possa executar o código do aplicativo. Entre outras funções, essa classe serve para nomear o pacote do Java para o diretório, descrever cada componente do aplicativo, declarar o nível mínimo de Android API que o aplicativo exige, listar as bibliotecas às quais o aplicativo deve se vincular, qual a tela inicial a ser exibida, permissões do aplicativo, como utilização do *bluetooth* entre outras funcionalidades (STUDIO, 2018).

No diagrama mostrado na figura 4.6, é mostrada a sequência lógica da conexão do *bluetooth* e sequência dos dados no aplicativo. Como funciona em apenas uma *activity*, a conexão não precisa ser reestabelecida dentro do algoritmo, pois as telas se sobrescrevem sempre que o usuário alterna entre elas. Primeiro é feito uma verificação para saber se o *smartphone* que está utilizando o aplicativo suporta este tipo de conexão, após isso, é validado através do *MAC address* do aparelho com o módulo, os dados enviados pelo aplicativo são recebidos e é enviado um bit de segurança para confirmar se o pacote chegou ao seu destino. Após a verificação dos pacotes enviados,

o microcontrolador espera uma alteração de variável para saber se muda alguma saída que está passando pelo conversor PWM. Caso não altere nada, o sistema continua perguntando se houve variação, entrando em *looping* até que haja alguma mudança.

FIGURA 4.6 – Diagrama de comunicação *bluetooth* transmissão de dados no aplicativo



Fonte: (OS AUTORES, 2018)

Os resultados referentes a mudança na temperatura de cor das luminárias foram atingidos. As luminárias possuem a capacidade de alternar entre 3000K e 6000K, varrendo toda esta faixa de temperatura de cor. A barra para realizar esta função está posicionada ao final da página inicial, onde se encontram também os cenários pré definidos de temperatura de com (3000K, 4000K e 5000K). Na figura 4.7, é mostrado que o usuário pode interagir com esta barra, alternando a temperatura de cor do sistema inteiro como um todo. Não é possível alterar temperatura de cor das luminárias individualmente, pois devido a irregularidades na iluminação, fica desagradável ao olho humano luminárias num mesmo ambiente com temperaturas de cor quentes e frias juntas.

FIGURA 4.7 – Cenários pré definidos e barra para mudança de temperatura de cor

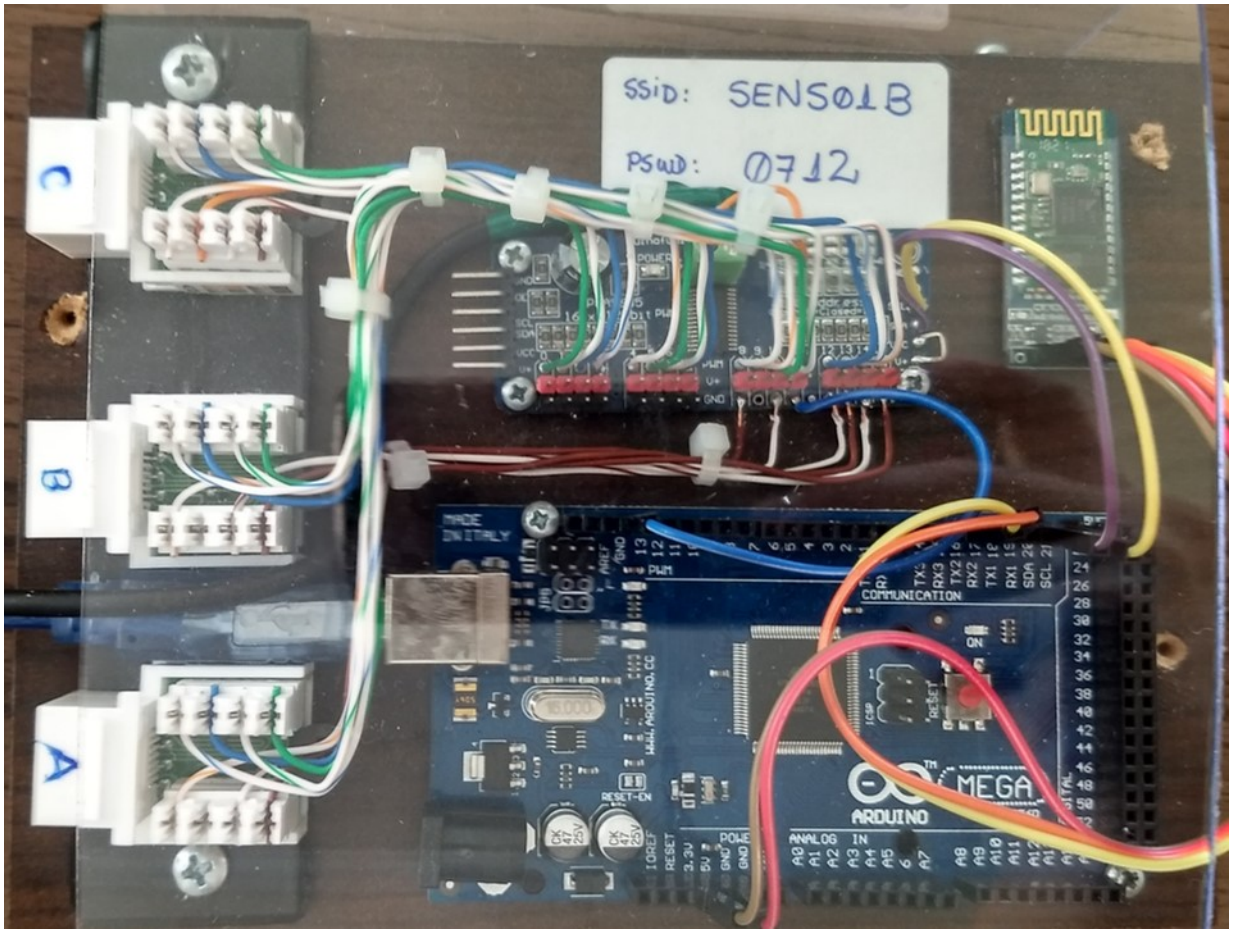
Fonte: (OS AUTORES, 2018)

Estes resultados de temperatura de cor foram capturados utilizando o equipamento *Spectrometro* Digital do fabricante Upr Tek, modelo MK350N PREMIUM. Equipamento este capaz de colher da luz emitida em seu sensor informações como quantidade de Lux, temperatura de cor, constatação de *flicker* entre outras.

4.1 IMPLEMENTAÇÃO EM *HARDWARE*

A implementação em *hardware* consiste na montagem do controlador, ou seja, após a configuração dos dispositivos, realizar a montagem física do protótipo de *hardware* que será utilizado para teste de campo e futuramente se tornará um produto a ser comercializado pela empresa *Lumicenter Lighting*. Na figura 4.8 é mostrada a foto tirada do protótipo inicial, chamado de protótipo *Alpha*.

FIGURA 4.8 – Protótipo Alpha da central de comando



Fonte: (OS AUTORES, 2018)

São utilizadas duas fontes convencionais 5V constantes para alimentação do microcontrolador e do conversor PWM. No canto inferior direito da figura 4.8 estão os cabos USB que estão realizando esta alimentação ao circuito.

No canto superior direito está posicionado o módulo de comunicação *bluetooth* HC-06, ele deve estar sempre posicionado com sua extremidade da antena para fora do protótipo, com intuito de evitar ruídos e garantir uma conexão estável. Do módulo sem fio, saem quatro conexões, as duas de alimentação 3,3V que o microcontrolador

fornece e outras duas que são referente ao transmissor e receptor, responsáveis por trocar informações com o CPU.

No canto inferior esquerdo está o microcontrolador Arduino, onde está programado as variáveis do circuito, que são registradas de 0 a 15, pois são enviados 16 sinais para o conversor PWM realizar análise, conversão e transmissão dos dados. Este microcontrolador que é utilizado como protótipo será alterado para o fabricante NXP, utilizando a arquitetura ARM, mas este protótipo será utilizado enquanto durarem os testes de *hardware*. No controlador estão conectados os cabos que recebem sinal do módulo de comunicação sem fio e dele saem os cabos para comunicação com o conversor de sinal PWM. Esta comunicação é realizada via protocolo I2C.

Cada saída marcada como A, B ou C pode ligar até sete luminárias no circuito, devido a utilização de conectores do tipo RJ-45 que possuem quatro pares trançados (8 fios), Possibilitando a ligação de 7 luminárias(um par cada luminária) e um par é dedicado para o comum das luminárias (GND).

O código desenvolvido e utilizado no Arduino consta no Apêndice F. Nele é possível verificar a inserção das bibliotecas, as declarações das variáveis e o tratamento dos dados que estão sendo recebidos da comunicação sem fio e passando via protocolo de comunicação I2C para o conversor PWM.

5 CONCLUSÃO

Com o crescimento cada vez maior de clientes que pedem por soluções de iluminação não mais genéricas e sim pessoais, atendendo suas necessidades específicas de eficiência, automatização e conforto, surge a necessidade deste projeto. Graças a todo este avanço que está acontecendo no ramo de iluminação *LED*, nos trouxe a concluir este desenvolvimento.

Em exemplo que pode ser citado para demonstrar a força que este tipo de projeto está tendo em território nacional, é possível colocar as novidades vistas na EXPOLUX 2018, que é a maior feira tecnológica de iluminação do Brasil. Muitos fabricantes grandes estão apresentando soluções que envolvem conectividade e digitalização de seus sistemas de iluminação.

Um grande desafio deste projeto é a implementação das ideias chave, como criação de cenários de iluminação e atuação da temperatura de cor respeitando o ciclo circadiano, pois quase todos os casos, é uma novidade para potenciais clientes. Além de que o aplicativo deve ser bastante seguro, com fácil manuseio e com fácil instalação, travando mais uma barreira em desenvolvimento.

Após estudo avançado em programação orientada a objetos, a equipe alcançou todo o potencial para desenvolvimento em Android Studio e conseguiu realizar um programa simples e eficaz, além de um sistema *hardware* com fácil instalação e implementação, com possibilidade de manutenção até mesmo pelo usuário final, com suporte rápido.

Concluiu-se que o protocolo de comunicação ideal fosse o *bluetooth* devido ao seu baixo custo, baixo consumo de energia e implementação, pois com estudo realizado pela equipe, este protocolo de comunicação mostrou estar bem desenvolvido e bastante difuso entre todos os equipamentos eletrônicos, como *smartphones* e *tablets*.

Os resultados até agora foram satisfatórios, o cronograma está sendo seguido rigorosamente dentro do prazo e a equipe conseguiu alcançar seus objetivos de teste, pois a versão final do protótipo de *hardware* está em fase final de desenvolvimento com o microcontrolador da fabricante NXP. O *software* está funcional, com comunicação ativa e atuando de maneira correta no PWM, porém, ainda não foram implementadas todas as funcionalidades planejadas.

5.0.1 Trabalhos Futuros

Este projeto ainda esta no projeto piloto, sendo montado o primeiro protótipo, denominado *Alpha*. Para próximo passo em questão de *hardware*, será alterado o microcontrolador para o fabricante NXP, com algo mais robusto, que utilize arquitetura ARM e deixe o equipamento com maior confiabilidade.

No aplicativo, o próximo passo será abrir a venda para usuários finais. Atualmente a criação de cenários é pré definida pelos desenvolvedores, sendo uma solução apenas para iluminação corporativa e de estandes de demonstração, portanto a futura atualização será implementação de método para criação e alteração de cenários de iluminação.

O banco de dados com informações que o cliente deseja armazenar está em desenvolvimento mas será lançado em trabalhos futuros. Neste banco de dados estarão armazenados dados tais como conexões realizadas, tempo de cada luminária ligada, economia gerada com dimerização e *status* potência consumida pelo tempo.

REFERÊNCIAS

A. CARVALHO B. C., N. F. M. R. V. *Drivers de Lâmpadas de LED: Topologias, Aplicações e Desempenho*. [S.l.]: ES - Engineering and Science, 2014. v. 1. Citado na página 30.

ARDUINO. *Modelos de Arduino - MEGA*. 2018. <https://store.arduino.cc/usa/arduino-mega-2560-rev3>. Acesso em 28 set. 2018. Citado na página 43.

ARM. *ARCHITECTING A SECURE FOUNDATION*. 2017. <https://www.arm.com/>. Acesso em 03 set. 2018. Citado na página 28.

BEGCONTROLS. *Digital and Open - Moving Lighting Controls into the 21st Century*. 2014. <http://www.begcontrols.com/pdf/Miscellanneous/Digital%20and%20Open%20-%20DALI%20vs.%200-10%20V.pdf>. Acesso em 12 out. 2018. Citado 2 vezes nas páginas 35 e 36.

BRAUN, F. J. *"Circa Diem": A Luz e o Ciclo Circadiano*. 2017. <http://fabiolajb.blogspot.com/2017/02/circa-diem-luz-e-o-ciclo-circadiano.html>. Acesso em 14 out. 2018. Citado na página 38.

CAELUM. *APOSTILA JAVA E ORIENTAÇÃO A OBJETOS*. 2014. <https://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/>. Acesso em 11 ago. 2018. Citado 2 vezes nas páginas 16 e 17.

DAVIS, S. *Power: Pulse-Width Modulation*. 2004. <https://www.electronicdesign.com/power/power-pulse-width-modulation>. Acesso em 22 set. 2018. Citado na página 25.

DEVMEDIA. *Programação Orientada a Objetos com Java*. 2010. <https://www.devmedia.com.br/programacao-orientada-a-objetos-com-java/18449>. Acesso em 25 ago. 2018. Citado na página 18.

FORTE, C. H. V. *Processadores ARM: visão geral e aplicações*. 2015. <https://www.dcce.ibilce.unesp.br/ aleardo/cursos/arqcomp/Semin_ARM.pdf>. Acesso em 02 set. 2018. Citado 2 vezes nas páginas 27 e 28.

GLOBO, O. *O que é ritmo circadiano, tema dos vencedores do Nobel de Medicina*. 2017. <https://oglobo.globo.com/sociedade/ciencia/o-que-ritmo-circadiano-tema-dos-vencedores-do-nobel-de-medicina-21897059>. Acesso em 12 out. 2018. Citado na página 37.

GUIMARÃES, G. *A história do sistema operacional Android*. 2013. <http://www.dsc.ufcg.edu.br/ pet/jornal/agosto2013/materias/historia_da_computacao.html>. Acesso em 21 ago. 2018. Citado na página 19.

JAVA. *O que é a Tecnologia Java e porque preciso dela?* 2018. <https://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acesso em 10 ago. 2018. Citado na página 16.

LUME. *O que é DALI?* 2015. <http://www.lumearquitetura.com.br/pdf/ed21/ed_21_aula.pdf>. Acesso em 17 out. 2018. Citado 2 vezes nas páginas 34 e 35.

LUMICENTER. *Dimmer 0-10V*. 2017. <http://www.lumicenteriluminacao.com.br/catalogo/dimmer-0-a-10v-p3922/>. Acesso em 01 nov. 2018. Citado na página 34.

MITSISTEMAS. *Programação Orientada a Objetos: Por que aprender?* 2018. <http://www.mitsistemas.com.br/2017/08/28/programacao-orientada-a-objetos-por-que-aprender/>. Acesso em 28 ago. 2018. Citado 2 vezes nas páginas 17 e 18.

MONQUEIRO, J. C. B. *Programação Orientada a Objetos: uma introdução*. 2007. <https://www.hardware.com.br/artigos/programacao-orientada-objetos/>. Acesso em 25 ago. 2018. Citado na página 17.

NXP. *Datasheet LPC802 32-bit ARM Cortex-M0+ microcontroller Rev. 1.6*. 2018. <https://www.nxp.com/docs/en/data-sheet/LPC802.pdf>. Acesso em 29 ago. 2018. Citado na página 27.

PERRY, J. S. *Fundamentos da linguagem Java*. 2016. <https://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1/index.html>. Acesso em 11 ago. 2018. Citado na página 16.

SCARDUELLI, L. B. *Curso básico de Android*. 2017. <https://www.slideshare.net/lucasscarduelli/curso-bsico-android-aula-01>. Acesso em 23 ago. 2018. Citado na página 19.

SCENARIO. *Controlador de cenários*. 2014. <http://scenario.ind.br/static/media/uploads/hotsites/CompactWall/index.html>. Acesso em 12 out. 2018. Citado 2 vezes nas páginas 32 e 33.

SILVEIRA, C. B. *O que é PWM e Para que Serve?* 2016. <https://www.citisystems.com.br/pwm/>. Acesso em 21 set. 2018. Citado 2 vezes nas páginas 25 e 26.

SIQUEIRA, T. S. de. *Bluetooth – Características, protocolos e funcionamento*. 2006. <http://www.ic.unicamp.br/ ducatte/mo401/1s2006/T2/057642-T.pdf>. Acesso em 10 out. 2018. Citado 2 vezes nas páginas 20 e 21.

STUDIO, A. *Conheça o Android Studio*. 2018. <https://developer.android.com/studio/intro/?hl=pt-br>. Acesso em 20 ago. 2018. Citado 2 vezes nas páginas 23 e 54.

TEIHAL. *How Pulse Width Modulation works*. 2014. <http://www.ee.teihal.gr/labs/electronics/web/downloads/What_is_PWM_and_why_is_it_useful.pdf> Acesso em 22 set. 2018. Citado na página 25.

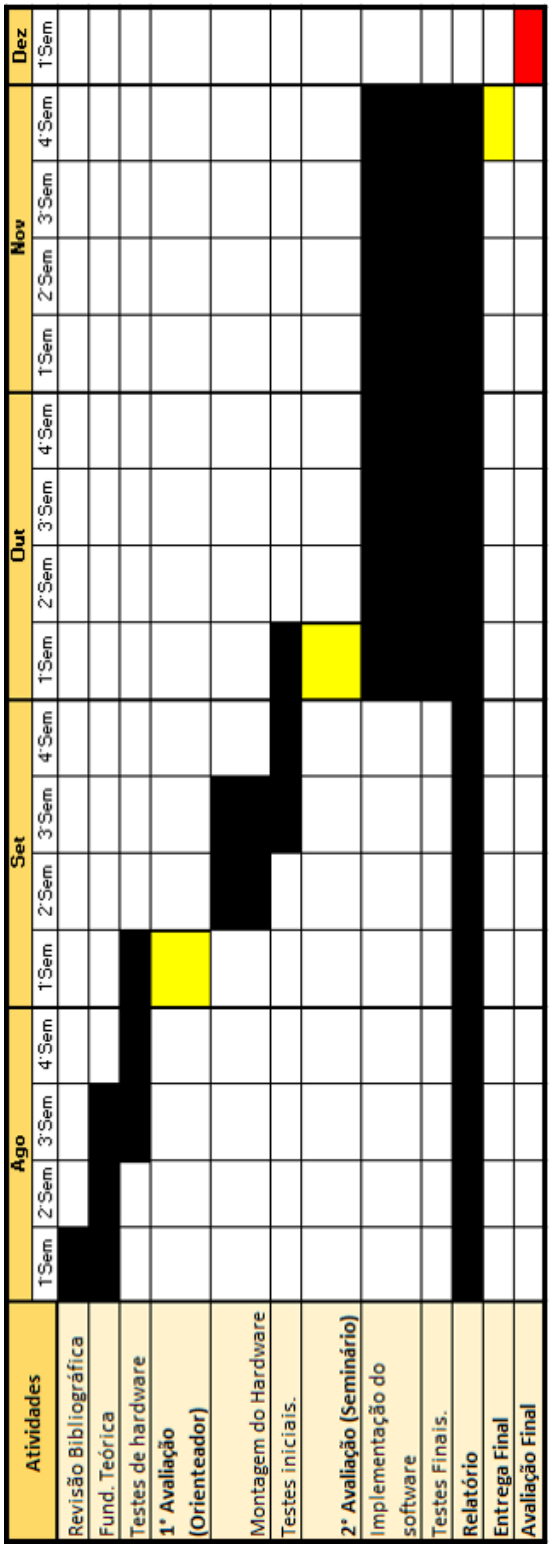
UFRJ. *Bluetooth – Camada fisica e de acesso ao meio*. 2012. <https://www.gta.ufrj.br/grad/09_1/versao-final/bluetooth/index.htm>. Acesso em 10 out. 2018. Citado na página 21.

UNICAMP. *Protocolos Seriais - I2C*. 2018. <http://www.dca.fee.unicamp.br/ rferra-ri/EA076_1s2018/Protocolo_I2C.pdf>. Acesso em 16 set. 2018. Citado 2 vezes nas páginas 22 e 23.

UNIVASF. *PROTOCOLO I2C*. 2015. <http://www.univasf.edu.br/ romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2C.pdf>. Acesso em 13 set. 2018. Citado na página 22.

6 APÊNDICE A - CRONOGRAMA

FIGURA 6.1 – Cronograma de atividades



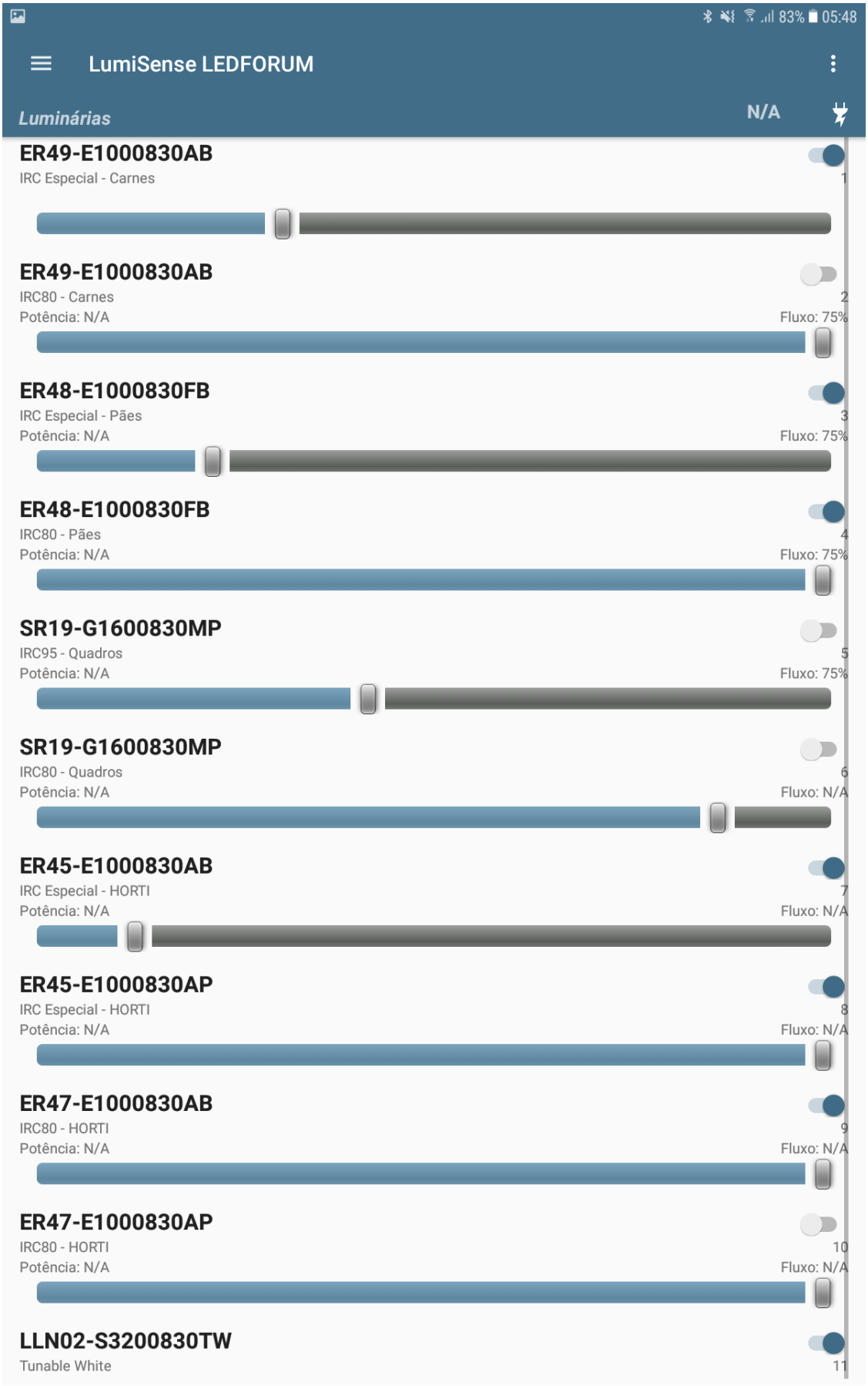
Fonte: (OS AUTORES, 2018)

7 APÊNDICE B - TELAS DO APLICATIVO

FIGURA 7.1 – Tela de inicialização do aplicativo

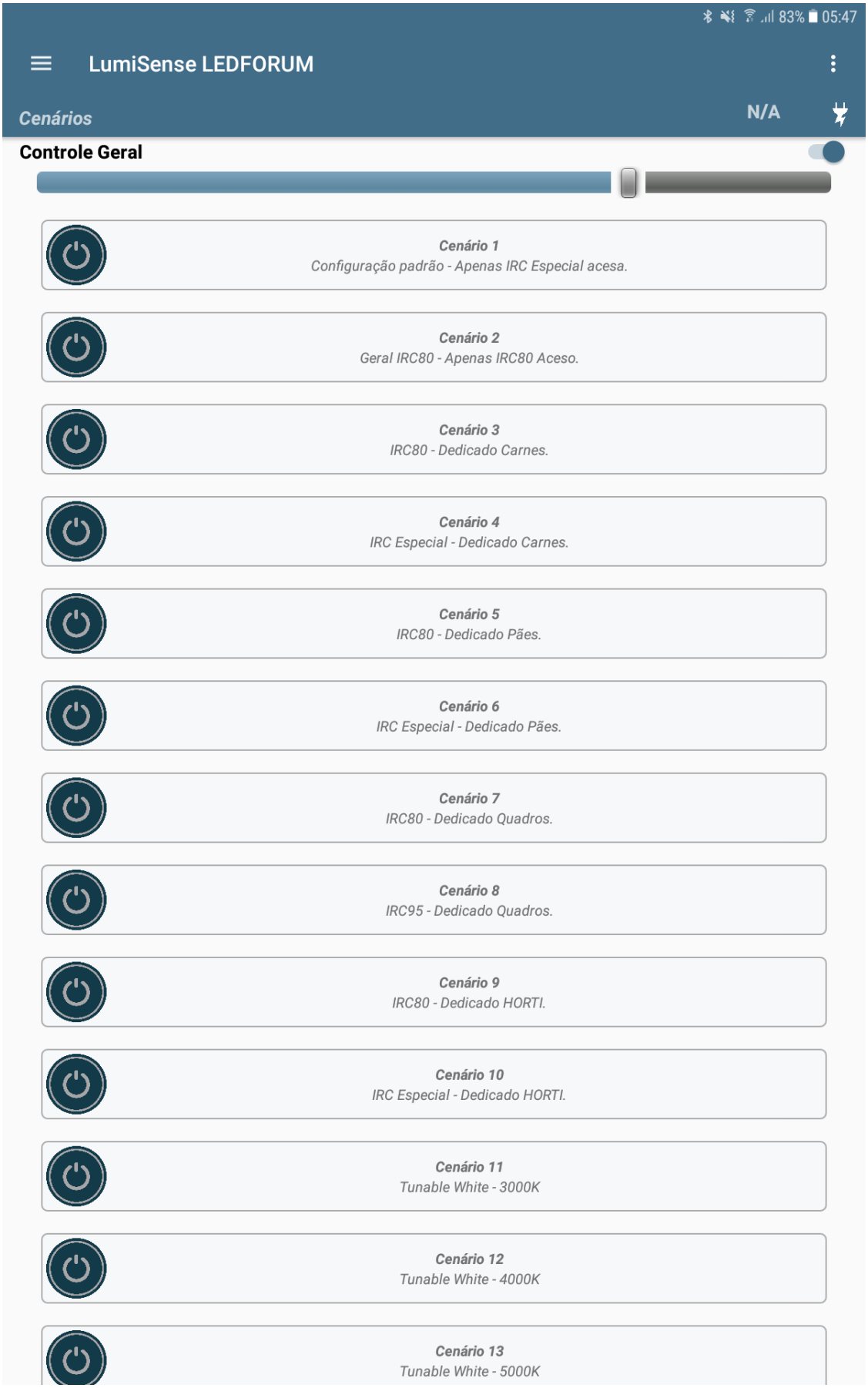
Fonte: (OS AUTORES, 2018)

FIGURA 7.2 – Tela de controle individual das luminárias



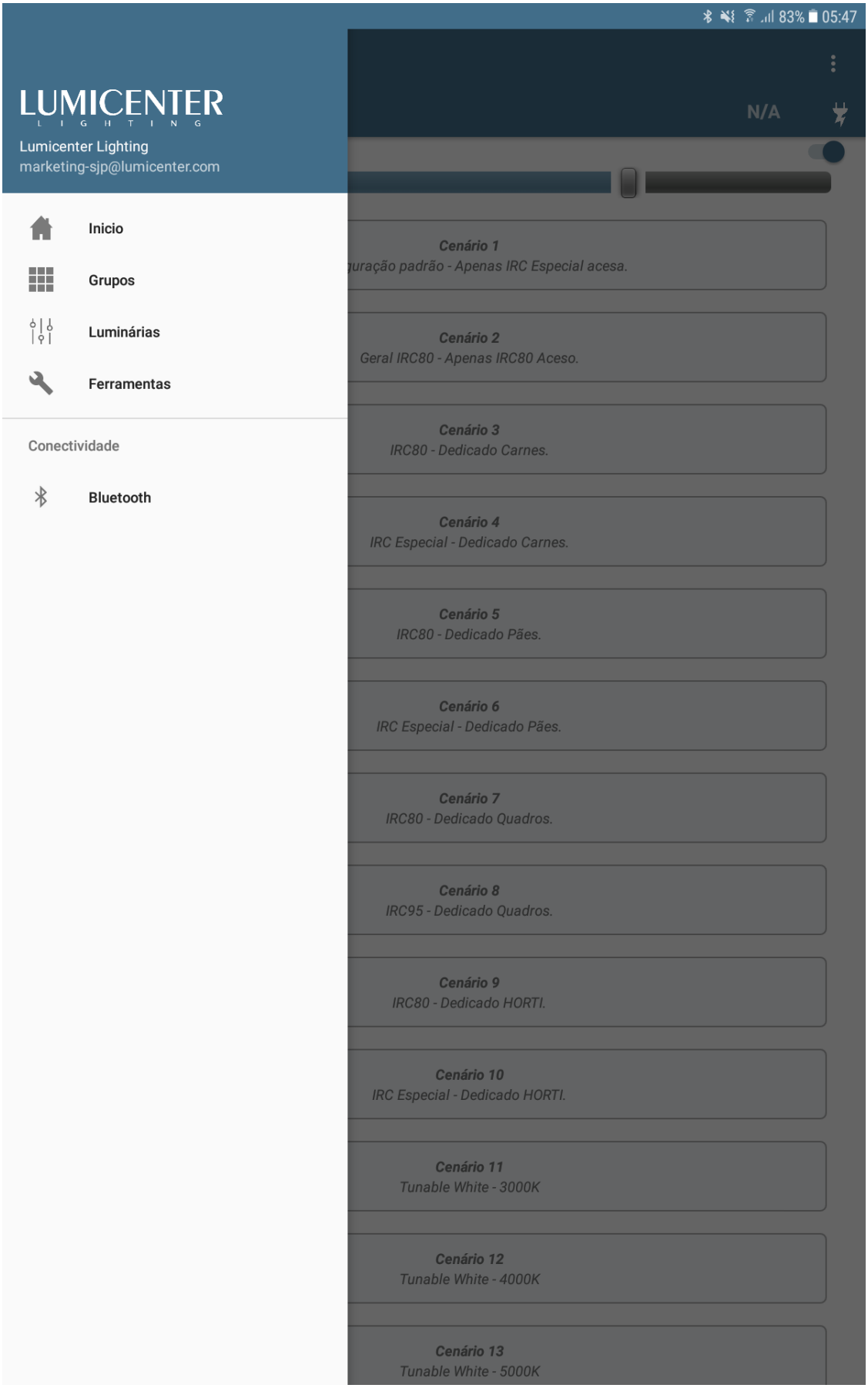
Fonte: (OS AUTORES, 2018)

FIGURA 7.3 – Tela de controle dos cenários de iluminação



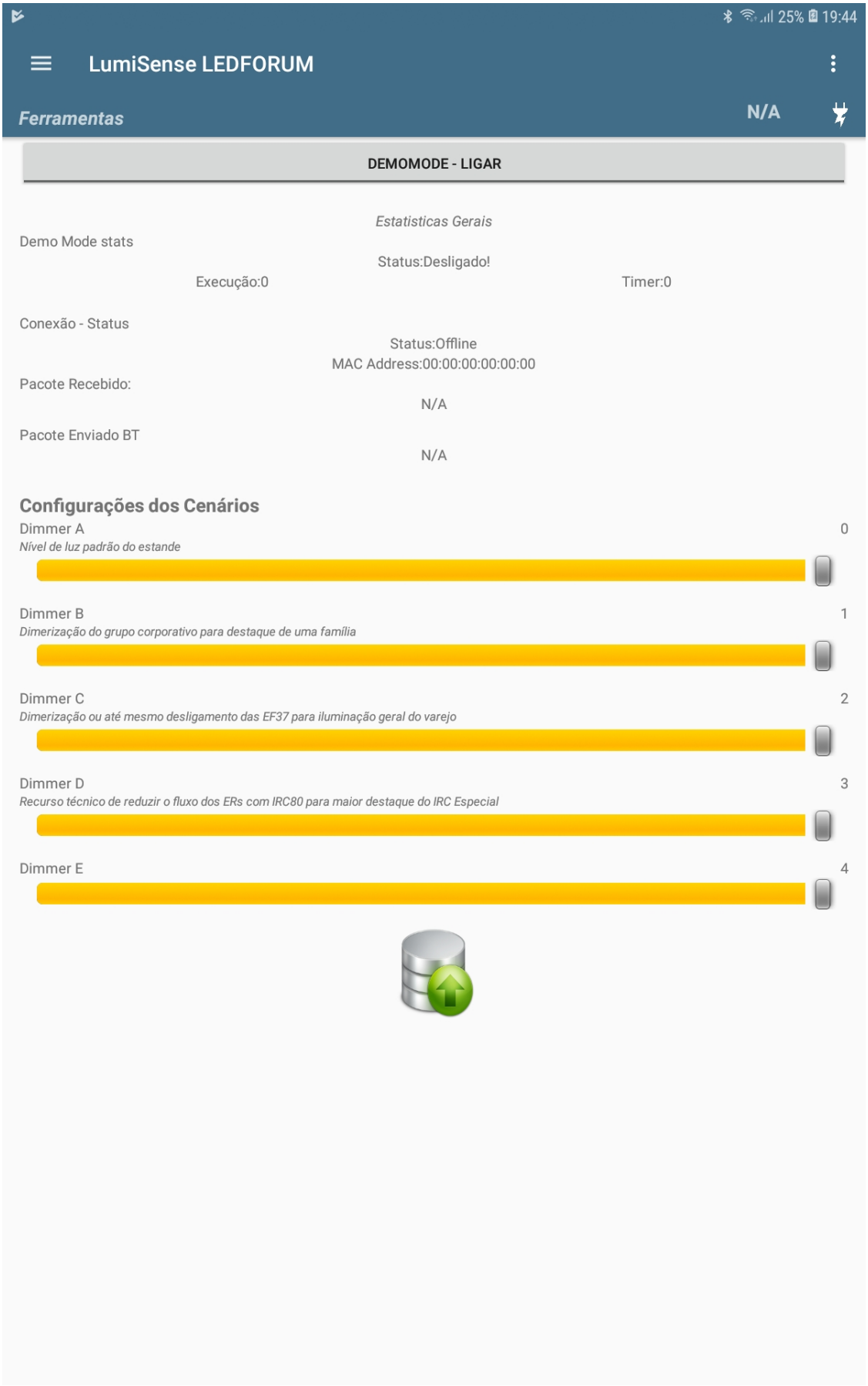
Fonte: (OS AUTORES, 2018)

FIGURA 7.4 – Aba de ferramentas do aplicativo



Fonte: (OS AUTORES, 2018)

FIGURA 7.5 – Tela de ferramentas



Fonte: (OS AUTORES, 2018)

8 APÊNDICE C - DADOS DE PACOTES *BLUETOOTH*

{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:76
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:80
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:84
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{2 :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:8804
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:92
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:96
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:10
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:10
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:10
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:11
{2 :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:11406
{200:200:200:200:200:200:20:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:11607
00:00 {200:200:200:200:200:200:200:1:C:1401:B
{200:200:200:200:200:200:200:C:1400:A :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:12
{200:200:200:200:200:200:200:1:C:1401:B :0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:2540:0:
{200:200:196:196:196:196:196:C:1380:A :0:0:64:64:64:64:64:0:0:0:0:0:0:0:0:2540
{176:176:176:176:200:176:176:1:C:1257:B :0:0:64:64:64:64:64:385:385:385:385:
{200:200:164:164:164:164:164:C:1220:A :0:0:577:577:577:577:577:385:385:385:3
{144:150:144:144:200:144:144:1:C:1071:B :0:0:577:577:577:577:577:898:802:898
{200:200:123:123:123:123:123:C:1015:A :0:0:1235:1235:1235:1235:1235:898:802:

{112:150:112:112:200:112:112:1:C:911:B :0:0:1235:1235:1235:1235:1235:1412:80

{200:200:89:89:89:89:100:C:856:A :0:0:1781:1781:1781:1781:1604:1412:802:1412

{70:150:70:70:200:70:70:1:C:701:B :0:0:1781:1781:1781:1781:1604:2086:802:208

{200:200:50:50:50:50:100:C:700:A :0:0:2407:2407:2407:2407:1604:2086:802:2086

{39:150:39:39:200:50:39:1:C:557:B :0:0:2407:2407:2407:2407:1604:2540:802:254

{200:200:16:16:16:16:100:C:564:A :0:0:2540:2540:2540:2540:1604:2540:802:2540

{0:150:0:0:200:50:0:1:C:401:B :0:0:2540:2540:2540:2540:1604:2540:802:2540:25

{187:200:0:0:13:13:87:C:500:A :208:0:2540:2540:2540:2540:1813:2540:802:2540:

{2 :208:0:2540:2540:2540:2540:1813:2540:802:2540:2540:0:2407:2540:2540:0:150

{153:200:0:0:47:47:53:C:500:A :754:0:2540:2540:2455:2455:2358:2540:802:2540:

{68:82:68:68:132:0:68:69:C:555:B :754:0:2540:2540:2455:2455:2358:2118:1893:2

{117:200:0:0:83:83:50:C:533:A :1331:0:2540:2540:1877:1877:2407:2118:1893:211

{100:48:102:102:98:0:102:103:C:655:B :1331:0:2540:2540:1877:1877:2407:1604:2

{8 :1331:0:2540:2540:1877:1877:2407:1604:2439:1572:1572:1636:2540:1572:1556:

{100:8:142:142:58:0:142:143:C:735:B :1331:0:2540:2540:1877:1877:2407:1604:25

{41:200:0:0:150:100:50:C:541:A :2540:0:2540:2540:802:1604:2407:1604:2540:930

{100:0:178:150:22:0:150:179:C:779:B :2540:0:2540:2540:802:1604:2407:1604:254

{8:200:0:0:150:100:50:C:508:A :2540:0:2540:2540:802:1604:2407:1604:2540:353:

{100:0:200:150:0:0:150:200:C:800:B :2540:0:2540:2540:802:1604:2407:1604:2540

{0:200:0:0:150:100:50:C:500:A :2540:0:2540:2540:802:1604:2407:1604:2540:0:80

{100:0:200:150:0:0:150:200:C:800:B :2540:0:2540:2540:802:1604:2407:1604:2540

{0:200:0:0:150:100:50:C:500:A :2540:0:2540:2540:802:1604:2407:1604:2540:0:80

9 APÊNDICE D - DADOS DE PACOTES NAS SAÍDAS PWM

MILLIS	PWM1	PWM2	PWM3	PWM4
12006	0	0	0	0
12206	0	0	0	0
12406	0	0	64	64
12606	0	0	64	64
12806	0	0	577	577
13004	0	0	577	577
13205	0	0	1235	1235
13405	0	0	1235	1235
13605	0	0	1781	1781
13805	0	0	1781	1781
14006	0	0	2407	2407
14204	0	0	2407	2407
14405	0	0	2540	2540
14606	0	0	2540	2540
14807	208	0	2540	2540
15005	208	0	2540	2540
15206	754	0	2540	2540
15406	754	0	2540	2540
15606	1331	0	2540	2540
15806	1331	0	2540	2540
16007	1331	0	2540	2540
16205	1331	0	2540	2540
16406	2540	0	2540	2540
16606	2540	0	2540	2540
16806	2540	0	2540	2540
17005	2540	0	2540	2540
17206	2540	0	2540	2540
17405	2540	0	2540	2540
17606	2540	0	2540	2540

17805	2540	0	2540	2540
18006	2540	0	2540	2540
18205	2540	0	2540	2540
18406	2540	497	2540	2540
18606	2540	497	2540	2540
18806	2540	994	2214	2540
19006	2540	994	2214	2540
19207	2540	1604	1588	2540
19406	2540	1604	1588	2540
19607	2540	1604	978	2540
19807	2540	1604	978	2540
20006	2540	1604	369	2540
20206	2540	1604	369	2540
20407	2540	1604	0	2540
20605	2540	1604	0	2540
20805	2540	1604	0	2540
21004	2540	1604	0	2540
21204	2540	1604	0	2540
21404	2540	1604	0	2540
21605	2540	1717	112	2540
21804	2540	1717	112	2540
22004	2455	2358	754	2540
22204	2455	2358	754	2540
22405	1877	2540	882	2540
22604	1877	2540	882	2540
22805	1604	2540	882	2540
23005	1604	2540	882	2540
23206	1604	2540	882	2540
23406	1604	2540	882	2540
23607	1604	2540	882	2540
23806	1604	2540	882	2540
24007	1604	2540	882	2540
24206	1604	2540	882	2540

24406	1604	2540	882	2540
24605	1604	2540	882	2540
24806	1604	2540	882	2540
25005	1604	2540	882	2540
25205	2021	2540	1299	2540
25405	2021	2540	1299	2540
25607	2540	2262	1829	2262
25806	2540	2262	1829	2262
26007	2540	2246	2455	1636
26207	2540	2246	2455	1636
26407	2540	2246	2540	1123
26607	2540	2246	2540	1123
26808	2540	2246	2540	609
27009	2540	2246	2540	609
27209	2540	2246	2540	0
27409	2540	2246	2540	0
27610	2540	2246	2540	0
27808	2540	2246	2540	0
28009	2540	2246	2540	0
28209	2540	2246	2540	0
28409	2540	2182	2540	64
28609	2540	2182	2540	64
28809	2535	1572	2540	673
29008	2535	1572	2540	673
29209	1941	978	2540	1267
29409	1941	978	2540	1267
29611	1444	481	2540	1765
29809	1444	481	2540	1765
30010	818	0	2540	2391
30210	818	0	2540	2391
30410	272	0	2540	2540
30611	272	0	2540	2540
30812	0	0	2540	2540

31010	0	0	2540	2540
31211	0	0	2540	2540
31411	0	0	2540	2540
31611	0	0	2540	2540

10 APÊNDICE E - CÓDIGO UTILIZADO NA *ACTIVITY* PRINCIPAL

```
package com.lumicenter.www.bluetoothlumicenterhexledforum;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothManager;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Build;
import android.os.Handler;
import android.os.Message;
import android.support.annotation.RequiresApi;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v4.widget.TextViewCompat;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.text.TextPaint;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ProgressBar;
import android.widget.SeekBar;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;
import org.w3c.dom.Text;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.UUID;
import static android.os.SystemClock.sleep;

public class MainActivity extends AppCompatActivity implements NavigationView.OnNa

    //Variaveis Globais
    //Button btnConexao;
    TextView textView;
    private Button btnGrupos;
    //final TextView textView = (TextView) findViewById(R.id.txtBT);
    //Button btnLed1, btnLed2, btnLed3;

    private static final int SOLICITA_ATIVACAO = 1; //Variavel para quando receber
    private static final int SOLICITA_CONEXAO = 2;
    private static final int MESSAGE_READ = 3; //Cdigo para falar que chegou mensa

    ConnectedThread connectedThread; //Thread de Conexo do Bluetooth
```

```
Handler mHandler;
StringBuilder dadosBT = new StringBuilder();

BluetoothAdapter meuBluetoothAdapter = null; //Inicializa o Bluetooth como Nul
BluetoothDevice meuDevice = null;
BluetoothSocket meuSocket = null;
boolean conexao = false;
private static String MAC = null;

UUID MEU_UUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb"); //Iden

//Botoes dos cenarios
ImageView btnCena1;
ImageView btnCena2;
ImageView btnCena3;
ImageView btnCena4;
ImageView btnCena5;
ImageView btnCena6;
ImageView btnCena7;
ImageView btnCena8;
ImageView btnCena9;
ImageView btnCena10;
ImageView btnCena11;
ImageView btnCena12;
ImageView btnCena13;
ImageView btnCena14;

private static TextView txtLuminaria1;
private static long tempoCLK = 0;
private static long tempoAux = 0;

//Variaveis para analise do sistema:
```

```
private static TextView tools_exec;
private static TextView tools_clock;
private static TextView tools_status;
private static TextView tools_mac;
private static TextView tools_received;
private static TextView tools_send;
private static TextView tools_demostats;

// ---------------------------------------------------------------
//  Inicio da declarao dos acessos globais as variaveis do cenrio
// ---------------------------------------------------------------

private static TextView percentualA;
private static TextView percentualB;
private static TextView percentualC;
private static TextView percentualD;
private static TextView percentualE;

public static SeekBar dimA;
public static SeekBar dimB;
public static SeekBar dimC;
public static SeekBar dimD;
public static SeekBar dimE;

Button btnSave; //Boto de salvar os cenarios

private static int A = 0;
private static int B = 0;
private static int C = 0;
private static int D = 0;
private static int E = 0;
```

```
public static LinearLayout scene1;
public static LinearLayout scene2;
public static LinearLayout scene3;
public static LinearLayout scene4;
public static LinearLayout scene5;
public static LinearLayout scene6;
public static LinearLayout scene7;
public static LinearLayout scene8;
public static LinearLayout scene9;
public static LinearLayout scene10;
public static LinearLayout scene11;
public static LinearLayout scene12;
public static LinearLayout scene13;
public static LinearLayout scene14;

//Interrupo do sistema
private static long initialTime;
private static Handler handler;
private static boolean isRunning;
private static final long MILLIS_IN_SEC = 1L;
private static final int SECS_IN_MIN = 60;

//Definio das teclas Switchs do programa.
public static Switch switch1;
public static Switch switch2;
public static Switch switch3;
public static Switch switch4;
public static Switch switch5;
public static Switch switch6;
public static Switch switch7;
public static Switch switch8;
public static Switch switch9;
public static Switch switch10;
```

```
public static Switch switch11;
public static Switch switch12;
public static Switch switch13;
public static Switch switch14;
public static Switch switchGeral;
public static SeekBar dim1;
public static SeekBar dim2;
public static SeekBar dim3;
public static SeekBar dim4;
public static SeekBar dim5;
public static SeekBar dim6;
public static SeekBar dim7;
public static SeekBar dim8;
public static SeekBar dim9;
public static SeekBar dim10;
public static SeekBar dim11;
public static SeekBar dim12;
public static SeekBar dim13;
public static SeekBar dim14;
public static SeekBar dim15;
public static SeekBar dimGeral;

private static TextView fluxo1;
private static TextView fluxo2;
private static TextView fluxo3;
private static TextView fluxo4;
private static TextView fluxo5;
private static TextView fluxo6;
private static TextView fluxo7;
private static TextView fluxo8;
private static TextView fluxo9;
private static TextView fluxo10;
private static TextView fluxo11;
```

```
private static TextView fluxo12;
private static TextView fluxo13;
private static TextView fluxo14;

private static TextView pot1;
private static TextView pot2;
private static TextView pot3;
private static TextView pot4;
private static TextView pot5;
private static TextView pot6;
private static TextView pot7;
private static TextView pot8;
private static TextView pot9;
private static TextView pot10;
private static TextView pot11;
private static TextView pot12;
private static TextView pot13;
private static TextView pot14;

private static TextView potGeral;

private static TextView pageName;

// ------------------------------------------
//  Variaveis da tela de Grupos
// ------------------------------------------

public static Switch btnGrupo1;
public static Switch btnGrupo2;
public static Switch btnGrupo3;
public static Switch btnGrupo4;
public static Switch btnGrupo5;
```

```
public static SeekBar conjunto1;
public static SeekBar conjunto2;
public static SeekBar conjunto3;
public static SeekBar conjunto4;
public static SeekBar conjunto5;
public static SeekBar conjunto6;

@RequiresApi(api = Build.VERSION_CODES.JELLY_BEAN_MR2)
@SuppressLint("HandlerLeak")
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    txtLuminaria1 = (TextView) findViewById(R.id.nome1);
    textView = (TextView) findViewById(R.id.textView);

    tools_clock = (TextView) findViewById(R.id.tools_clock);
    tools_exec = (TextView) findViewById(R.id.tools_exec);
    tools_mac = (TextView) findViewById(R.id.tools_mac);
    tools_received = (TextView) findViewById(R.id.tools_receive);
    tools_send = (TextView) findViewById(R.id.tools_send);
    tools_status = (TextView) findViewById(R.id.tools_status);
    tools_demostats = (TextView) findViewById(R.id.tools_demostats);

    // ------------------------------------------------------------
    //  Declarao das variaveis relacionadas aos cenarios
    // ------------------------------------------------------------

        percentualA = (TextView) findViewById(R.id.percentualA);
        percentualB = (TextView) findViewById(R.id.percentualB);
        percentualC = (TextView) findViewById(R.id.percentualC);
        percentualD = (TextView) findViewById(R.id.percentualD);
```

```
        percentualE = (TextView) findViewById(R.id.percentualE);

        dimA = (SeekBar) findViewById(R.id.dimA);
        dimB = (SeekBar) findViewById(R.id.dimB);
        dimC = (SeekBar) findViewById(R.id.dimC);
        dimD = (SeekBar) findViewById(R.id.dimD);
        dimE = (SeekBar) findViewById(R.id.dimE);
        btnSave = (Button) findViewById(R.id.btnSave);

    // ------------------------------------------------------------
    //  Declarao das variaveis relacionadas aos Grupos
    // ------------------------------------------------------------
    btnGrupo1 = (Switch) findViewById(R.id.btnGrupo1);
    btnGrupo2 = (Switch) findViewById(R.id.btnGrupo2);
    btnGrupo3 = (Switch) findViewById(R.id.btnGrupo3);
    btnGrupo4 = (Switch) findViewById(R.id.btnGrupo4);
    btnGrupo5 = (Switch) findViewById(R.id.btnGrupo5);
    conjunto1 = (SeekBar) findViewById(R.id.dimGrupo1);
    conjunto2 = (SeekBar) findViewById(R.id.dimGrupo2);
    conjunto3 = (SeekBar) findViewById(R.id.dimGrupo3);
    conjunto4 = (SeekBar) findViewById(R.id.dimGrupo4);
    conjunto5 = (SeekBar) findViewById(R.id.dimGrupo5);
    conjunto6 = (SeekBar) findViewById(R.id.dimGrupo6);

    // ------------------------------------------------------
    //  Declarao das variaveis relacionadas as luminrias
    // ------------------------------------------------------

        pot1 = (TextView) findViewById(R.id.pot1);
        pot2 = (TextView) findViewById(R.id.pot2);
        pot3 = (TextView) findViewById(R.id.pot3);
        pot4 = (TextView) findViewById(R.id.pot4);
        pot5 = (TextView) findViewById(R.id.pot5);
```

```
pot6 = (TextView) findViewById(R.id.pot6);
pot7 = (TextView) findViewById(R.id.pot7);
pot8 = (TextView) findViewById(R.id.pot8);
pot9 = (TextView) findViewById(R.id.pot9);
pot10 = (TextView) findViewById(R.id.pot10);
potGeral = (TextView) findViewById(R.id.potGeral);

pageName = (TextView) findViewById(R.id.pageName);

fluxo1 = (TextView) findViewById(R.id.fluxo1);
fluxo2 = (TextView) findViewById(R.id.fluxo2);
fluxo3 = (TextView) findViewById(R.id.fluxo3);
fluxo4 = (TextView) findViewById(R.id.fluxo4);
fluxo5 = (TextView) findViewById(R.id.fluxo5);
fluxo6 = (TextView) findViewById(R.id.fluxo6);
fluxo7 = (TextView) findViewById(R.id.fluxo7);
fluxo8 = (TextView) findViewById(R.id.fluxo8);
fluxo9 = (TextView) findViewById(R.id.fluxo9);
fluxo10 = (TextView) findViewById(R.id.fluxo10);

switch1 = (Switch) findViewById(R.id.switch1);
switch2 = (Switch) findViewById(R.id.switch2);
switch3 = (Switch) findViewById(R.id.switch3);
switch4 = (Switch) findViewById(R.id.switch4);
switch5 = (Switch) findViewById(R.id.switch5);
switch6 = (Switch) findViewById(R.id.switch6);
switch7 = (Switch) findViewById(R.id.switch7);
switch8 = (Switch) findViewById(R.id.switch8);
switch9 = (Switch) findViewById(R.id.switch9);
switch10 = (Switch) findViewById(R.id.switch10);
switch11 = (Switch) findViewById(R.id.switch11);
switchGeral = (Switch) findViewById(R.id.switchGeral);
```

```
        dim1 = (SeekBar) findViewById(R.id.dim1);
        dim2 = (SeekBar) findViewById(R.id.dim2);
        dim3 = (SeekBar) findViewById(R.id.dim3);
        dim4 = (SeekBar) findViewById(R.id.dim4);
        dim5 = (SeekBar) findViewById(R.id.dim5);
        dim6 = (SeekBar) findViewById(R.id.dim6);
        dim7 = (SeekBar) findViewById(R.id.dim7);
        dim8 = (SeekBar) findViewById(R.id.dim8);
        dim9 = (SeekBar) findViewById(R.id.dim9);
        dim10 = (SeekBar) findViewById(R.id.dim10);
        dim15 = (SeekBar) findViewById(R.id.dim15);

        dimGeral = (SeekBar) findViewById(R.id.dimGeral);

    //Conexo Bluetooth
    meuBluetoothAdapter = BluetoothAdapter.getDefaultAdapter(); //Passo a passo
    if (meuBluetoothAdapter == null) { //Verifica se o celular tem Bluetooth?
        // Device does not support Bluetooth
        Toast.makeText(getApplicationContext(), "Seu dispositivo no possui Blue
    } else if(!meuBluetoothAdapter.isEnabled()) { //Verifica se o celular esta
        Intent ativaBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENAB
        startActivityForResult(ativaBluetooth, SOLICITA_ATIVACAO); //Chama a vo
    }

    switchGeral.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeLi
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b)
            if(switchGeral.isChecked()) {
                statusLuminaria(true);
                atualizaGeral(dimGeral.getProgress());
            } else
            {
                statusLuminaria(false);
```

```
            atualizaGeral(dimGeral.getProgress());
        }
    }
});

//Menu Switcher
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
        this, drawer, toolbar, R.string.navigation_drawer_open, R.string.na
drawer.addDrawerListener(toggle);
toggle.syncState();

NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view
navigationView.setNavigationItemSelectedListener(this);

handler = new Handler();

switch1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListen
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b)

    }
});

dim1.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean b)

    }

    @Override
```

```
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });

    dim2.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean b)

        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });

    dim3.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean b)

        }
```

```
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});

dim4.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean b)

    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});

dimGeral.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean b)
            atualizaGeral(dimGeral.getProgress());
            if(switchGeral.isChecked()) statusLuminaria(true);
```

```
            else statusLuminaria(false);
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });

    scene1 = (LinearLayout) findViewById(R.id.scene1);
    scene2 = (LinearLayout) findViewById(R.id.scene2);
    scene3 = (LinearLayout) findViewById(R.id.scene3);
    scene4 = (LinearLayout) findViewById(R.id.scene4);
    scene5 = (LinearLayout) findViewById(R.id.scene5);
    scene6 = (LinearLayout) findViewById(R.id.scene6);
    scene7 = (LinearLayout) findViewById(R.id.scene7);
    scene8 = (LinearLayout) findViewById(R.id.scene8);
    scene9 = (LinearLayout) findViewById(R.id.scene9);
    scene10 = (LinearLayout) findViewById(R.id.scene10);
    scene11 = (LinearLayout) findViewById(R.id.scene11);
    scene12 = (LinearLayout) findViewById(R.id.scene12);
    scene13 = (LinearLayout) findViewById(R.id.scene13);
    scene1.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            cena1(view);
        }
    });
```

```
scene2.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        cena2(view);
    }
});
scene3.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        cena3(view);
    }
});
scene4.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        cena4(view);
    }
});
scene5.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        cena5(view);
    }
});
scene6.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        cena6(view);
    }
});
scene7.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
```

```
            cena7(view);
        }
    });
    scene8.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            cena8(view);
        }
    });
    scene9.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            cena9(view);
        }
    });
    scene10.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            cena10(view);
        }
    });
    scene11.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            cena11(view);
        }
    });
    scene12.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            cena12(view);
        }
    });
```

```
    scene13.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            cena13(view);
        }
    });

// ------------------------------------------------------------------------
//  Inicio da chamada dos botoes dos Grupos
// ------------------------------------------------------------------------

    btnGrupo1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeList
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b)
            switch3.setChecked(btnGrupo1.isChecked());

            dim3.setProgress(conjunto1.getProgress());
        }
    });

    conjunto1.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean b)
            switch3.setChecked(btnGrupo1.isChecked());
            dim3.setProgress(conjunto1.getProgress());
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
```

```
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});

btnGrupo2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeList
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b)
        switch2.setChecked(btnGrupo2.isChecked());
        switch4.setChecked(btnGrupo2.isChecked());
        switch7.setChecked(btnGrupo2.isChecked());

        dim2.setProgress(conjunto2.getProgress());
        dim4.setProgress(conjunto2.getProgress());
        dim7.setProgress(conjunto2.getProgress());
    }
});

btnGrupo3.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeList
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b)
        switch1.setChecked(btnGrupo3.isChecked());
        switch5.setChecked(btnGrupo3.isChecked());
        switch6.setChecked(btnGrupo3.isChecked());

        dim1.setProgress(conjunto3.getProgress());
        dim5.setProgress(conjunto3.getProgress());
        dim6.setProgress(conjunto3.getProgress());
    }
});

conjunto3.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
    @Override
```

```
        public void onProgressChanged(SeekBar seekBar, int progress, boolean b)
            switch1.setChecked(btnGrupo3.isChecked());
            switch5.setChecked(btnGrupo3.isChecked());
            switch6.setChecked(btnGrupo3.isChecked());

            dim1.setProgress(conjunto3.getProgress());
            dim5.setProgress(conjunto3.getProgress());
            dim6.setProgress(conjunto3.getProgress());
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });

    btnGrupo4.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeList
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b)
            switch8.setChecked(btnGrupo4.isChecked());
            switch9.setChecked(btnGrupo4.isChecked());

            dim8.setProgress(conjunto4.getProgress());
            dim9.setProgress(conjunto4.getProgress());
        }
    });

    conjunto4.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
```

```
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean b)
        switch8.setChecked(btnGrupo4.isChecked());
        switch9.setChecked(btnGrupo4.isChecked());

        dim8.setProgress(conjunto4.getProgress());
        dim9.setProgress(conjunto4.getProgress());
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});

conjunto6.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean b)
        dim15.setProgress(conjunto6.getProgress());
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
```

```
    }
});


mHandler = new Handler(){
    @Override
    public void handleMessage(Message msg) {

        if(msg.what == MESSAGE_READ){
            String recebidos = (String) msg.obj;

            dadosBT.append(recebidos); //Vai juntando os dados

            int fimProtocolo = dadosBT.indexOf("}");

            if(fimProtocolo > 0){ //Chegou o fim do Protocolo
                String protocoloCompleto = dadosBT.substring(0,fimProtocolo

                int sizeProtocol = protocoloCompleto.length();

                if(dadosBT.charAt(0) == '{') { //Possui o inicio do protoc

                    String dadosFinais = dadosBT.substring(1, sizeProtocol)
                    int  out1 =  dim1.getProgress();
                    int  out2 =  dim2.getProgress();
                    int  out3 =  dim3.getProgress();
                    int  out4 =  dim4.getProgress();
                    int  out5 =  dim5.getProgress();
                    int  out6 =  dim6.getProgress();
                    int  out7 =  dim7.getProgress();
                    int  out8 =  dim8.getProgress();
                    int  out9 =  dim9.getProgress();
```

```
int out10 = dim10.getProgress();
int out11 = 200;
int out12 = 200;
int out13 = dim15.getProgress();
int out14 = 200;
int out15 = 200;

//pot1.setText((R.string.pot1));

long potencia1 = ((10*(out1+11))/210);
long potencia2 = ((10*(out2+11))/210);
long potencia3 = ((10*(out3+11))/210);
long potencia4 = ((10*(out4+11))/210);
long potencia5 = ((30*(out5+11))/210);
long potencia6 = ((30*(out6+11))/210);
long potencia7 = ((20*(out7+11))/210);
long potencia8 = ((20*(out8+11))/210);
long potencia9 = ((20*(out9+11))/210);
long potencia10 = ((20*(out10+11))/210);
long potencia11 = 248;
long potencia12 = 0;
long potencia13 = 0;
long potencia14 = 0;

fluxo1.setText(("Fluxo: " + (((out1+11)*100)/210) + "%"
fluxo2.setText(("Fluxo: " + (((out2+11)*100)/210) + "%"
fluxo3.setText(("Fluxo: " + (((out3+11)*100)/210) + "%"
fluxo4.setText(("Fluxo: " + (((out4+11)*100)/210) + "%"
fluxo5.setText(("Fluxo: " + (((out5+11)*100)/210) + "%"
fluxo6.setText(("Fluxo: " + (((out6+11)*100)/210) + "%"
fluxo7.setText(("Fluxo: " + (((out7+11)*100)/210) + "%"
fluxo8.setText(("Fluxo: " + (((out8+11)*100)/210) + "%"
fluxo9.setText(("Fluxo: " + (((out9+11)*100)/210) + "%"
```

```
fluxo10.setText(("Fluxo: " + (((out10+11)*100)/210) + "

if(!switch1.isChecked())  {  out1 = 255; potencia1 = 0;
if(!switch2.isChecked())  {  out2 = 255; potencia2 = 0;
if(!switch3.isChecked())  {  out3 = 255; potencia3 = 0;
if(!switch4.isChecked())  {  out4 = 255; potencia4 = 0;
if(!switch5.isChecked())  {  out5 = 255; potencia5 = 0;
if(!switch6.isChecked())  {  out6 = 255; potencia6 = 0;
if(!switch7.isChecked())  {  out7 = 255; potencia7 = 0;
if(!switch8.isChecked())  {  out8 = 255; potencia8 = 0;
if(!switch9.isChecked())  {  out9 = 255; potencia9 = 0;
if(!switch10.isChecked()) { out10 = 255; potencia10 = 0
if(!switch11.isChecked()) { potencia11 = 0; }
if(switch11.isChecked()) { out15 = 255; }

pot1.setText( ("Potncia: " + potencia1 + "W"));
pot2.setText( ("Potncia: " + potencia2 + "W"));
pot3.setText( ("Potncia: " + potencia3 + "W"));
pot4.setText( ("Potncia: " + potencia4 + "W"));
pot5.setText( ("Potncia: " + potencia5 + "W"));
pot6.setText( ("Potncia: " + potencia6 + "W"));
pot7.setText( ("Potncia: " + potencia7 + "W"));
pot8.setText( ("Potncia: " + potencia8 + "W"));
pot9.setText( ("Potncia: " + potencia9 + "W"));
pot10.setText(("Potncia: " + potencia10 + "W"));

    long potenciaGeral = potencia1 + potencia2 + potenc
potGeral.setText(((potenciaGeral) + "W"));

    tools_received.setText(dadosFinais);

int checksum1 = out1+out2+out3+out4+out5+out6+out7;
int checksum2 = out8+out9+out10+out11+out12+out13+out14
```

```
                        int checksum = out1+out2+out3+out4+out5+out6+out7+out8+

                        sleep(10);
                        connectedThread.enviarBT("{"+ Integer.toHexString(out1)
                    }
                    dadosBT.delete(0,dadosBT.length()); //Para limpar o Buffer
                }
            }

        } //Evento que monitora o que chegar
    };

}

@Override
protected void onPause(){
    super.onPause();
    if(isRunning) {
        Toast.makeText(getApplicationContext(), "Pausado em: " + tempoCLK, Toas
        tools_demostats.setText("Desligado");
        isRunning = false;
        handler.removeCallbacks(runnable);
    }
}

void demomode(){
            //tools_status.setText(String.valueOf(meuSocket.isConnected()));

            //Para chamar a funo timer HANDLER
            if(!isRunning) {
                tools_demostats.setText("Ligado!");
                isRunning = true;
```

```
                initialTime = System.currentTimeMillis();
                handler.postDelayed(runnable, MILLIS_IN_SEC);
            }else{
                Toast.makeText(getApplicationContext(), "SEGUNDOS: " + tempoCLK
                tools_demostats.setText("Desligado");
                isRunning = false;
                handler.removeCallbacks(runnable);
            }
}

private static int demo1 = 0;
private static int demo2 = 0;
private static int demo3 = 0;
private static int demo4 = 0;
private static int demo5 = 0;
private static int demo6 = 0;
private static int demo7 = 0;
private static int demo8 = 0;
private static int demo9 = 0;
private static int demo10 = 0;
private static int demo11 = 0;
private static int demo12 = 0;
private static int demo13 = 0;
private static int demo14 = 0;
private static int demo15 = 0;

private final static Runnable runnable = new Runnable() {
    @Override
    public void run() {

            if(tempoCLK>7) tempoCLK=0;

            if (tempoCLK == 0) {
```

```
        demo1 = 200;
        demo2 = 0;
        demo3 = 0;
        demo4 = 0;
        demo5 = 0;
        demo6 = 0;
        demo7 = 0;
        demo8 = 200;
        demo9 = 0;
        demo10 = 0;
        demo11 = 0;
        demo12 = 0;
        demo13 = 0;
        demo14 = 0;
        demo15 = 0;
    }

    if (tempoCLK == 1) {
        demo1 = 0;
        demo2 = 200;
        demo3 = 0;
        demo4 = 0;
        demo5 = 0;
        demo6 = 0;
        demo7 = 0;
        demo8 = 0;
        demo9 = 200;
        demo10 = 0;
        demo11 = 0;
        demo12 = 0;
        demo13 = 0;
        demo14 = 0;
        demo15 = 0;
```

```
}

if (tempoCLK == 2) {
    demo1 = 0;
    demo2 = 0;
    demo3 = 200;
    demo4 = 0;
    demo5 = 0;
    demo6 = 0;
    demo7 = 0;
    demo8 = 0;
    demo9 = 0;
    demo10 = 200;
    demo11 = 0;
    demo12 = 0;
    demo13 = 0;
    demo14 = 0;
    demo15 = 0;
}

if (tempoCLK == 3) {
    demo1 = 0;
    demo2 = 0;
    demo3 = 0;
    demo4 = 200;
    demo5 = 0;
    demo6 = 0;
    demo7 = 0;
    demo8 = 0;
    demo9 = 0;
    demo10 = 0;
    demo11 = 200;
    demo12 = 0;
```

```
    demo13 = 0;
    demo14 = 0;
    demo15 = 0;
}

if (tempoCLK == 4) {
    demo1 = 0;
    demo2 = 0;
    demo3 = 0;
    demo4 = 0;
    demo5 = 200;
    demo6 = 0;
    demo7 = 0;
    demo8 = 0;
    demo9 = 0;
    demo10 = 0;
    demo11 = 0;
    demo12 = 200;
    demo13 = 0;
    demo14 = 0;
    demo15 = 0;
}

if (tempoCLK == 5) {
    demo1 = 0;
    demo2 = 0;
    demo3 = 0;
    demo4 = 0;
    demo5 = 0;
    demo6 = 200;
    demo7 = 0;
    demo8 = 0;
    demo9 = 0;
```

```
        demo10 = 0;
        demo11 = 0;
        demo12 = 0;
        demo13 = 200;
        demo14 = 0;
        demo15 = 0;
    }

    if (tempoCLK == 6) {
        demo1 = 0;
        demo2 = 0;
        demo3 = 0;
        demo4 = 0;
        demo5 = 0;
        demo6 = 0;
        demo7 = 200;
        demo8 = 0;
        demo9 = 0;
        demo10 = 0;
        demo11 = 0;
        demo12 = 0;
        demo13 = 0;
        demo14 = 200;
        demo15 = 0;
    }

    if (tempoCLK == 7) {
        demo1 = 0;
        demo2 = 0;
        demo3 = 0;
        demo4 = 0;
        demo5 = 0;
        demo6 = 0;
```

```
    demo7 = 0;
    demo8 = 0;
    demo9 = 0;
    demo10 = 0;
    demo11 = 0;
    demo12 = 0;
    demo13 = 0;
    demo14 = 0;
    demo15 = 200;
}

if(dim1.getProgress()>demo1) dim1.setProgress(dim1.getProgress()-1)
if(dim2.getProgress()>demo2) dim2.setProgress(dim2.getProgress()-1)
if(dim3.getProgress()>demo3) dim3.setProgress(dim3.getProgress()-1)
if(dim4.getProgress()>demo4) dim4.setProgress(dim4.getProgress()-1)
if(dim5.getProgress()>demo5) dim5.setProgress(dim5.getProgress()-1)
if(dim6.getProgress()>demo6) dim6.setProgress(dim6.getProgress()-1)
if(dim7.getProgress()>demo7) dim7.setProgress(dim7.getProgress()-1)
if(dim8.getProgress()>demo8) dim8.setProgress(dim8.getProgress()-1)
if(dim9.getProgress()>demo9) dim9.setProgress(dim9.getProgress()-1)
if(dim10.getProgress()>demo10) dim10.setProgress(dim10.getProgress(
if(dim11.getProgress()>demo11) dim11.setProgress(dim11.getProgress(
if(dim12.getProgress()>demo12) dim12.setProgress(dim12.getProgress(
if(dim13.getProgress()>demo13) dim13.setProgress(dim13.getProgress(
if(dim14.getProgress()>demo14) dim14.setProgress(dim14.getProgress(
if(dim15.getProgress()>demo15) dim15.setProgress(dim15.getProgress(

if(dim1.getProgress()<demo1) dim1.setProgress(dim1.getProgress()+1)
if(dim2.getProgress()<demo2) dim2.setProgress(dim2.getProgress()+1)
if(dim3.getProgress()<demo3) dim3.setProgress(dim3.getProgress()+1)
if(dim4.getProgress()<demo4) dim4.setProgress(dim4.getProgress()+1)
if(dim5.getProgress()<demo5) dim5.setProgress(dim5.getProgress()+1)
if(dim6.getProgress()<demo6) dim6.setProgress(dim6.getProgress()+1)
```

```
                if(dim7.getProgress()<demo7) dim7.setProgress(dim7.getProgress()+1)
                if(dim8.getProgress()<demo8) dim8.setProgress(dim8.getProgress()+1)
                if(dim9.getProgress()<demo9) dim9.setProgress(dim9.getProgress()+1)
                if(dim10.getProgress()<demo10) dim10.setProgress(dim10.getProgress(
                if(dim11.getProgress()<demo11) dim11.setProgress(dim11.getProgress(
                if(dim12.getProgress()<demo12) dim12.setProgress(dim12.getProgress(
                if(dim13.getProgress()<demo13) dim13.setProgress(dim13.getProgress(
                if(dim14.getProgress()<demo14) dim14.setProgress(dim14.getProgress(
                if(dim15.getProgress()<demo15) dim15.setProgress(dim15.getProgress(
            }
        }

};

public void statusLuminaria(boolean status){
    switch1.setChecked(status);
    switch2.setChecked(status);
    switch3.setChecked(status);
    switch4.setChecked(status);
    switch5.setChecked(status);
    switch6.setChecked(status);
    switch7.setChecked(status);
    switch8.setChecked(status);
    switch9.setChecked(status);
    switch10.setChecked(status);
    switch11.setChecked(status);
}

public void atualizaGeral(int status){
    dim1.setProgress(status);
    dim2.setProgress(status);
    dim3.setProgress(status);
    dim4.setProgress(status);
```

```
        dim5.setProgress(status);
        dim6.setProgress(status);
        dim7.setProgress(status);
        dim8.setProgress(status);
        dim9.setProgress(status);
        dim10.setProgress(status);
    }

    public void cena1(View view){
        statusLuminaria(true);
        atualizaGeral(0);

            botaoDesligado();
            btnCena1.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch4.setChecked(false);
        switch6.setChecked(false);
        switch9.setChecked(false);
        switch10.setChecked(false);

        dim1.setProgress(A);
        dim3.setProgress(A);
        dim5.setProgress(200);
        dim7.setProgress(A);
        dim8.setProgress(A);
        dim15.setProgress(1);
    }

    public void cena2(View view){
        statusLuminaria(true);
        atualizaGeral(0);
```

```
        //Muda o estado do boto
        botaoDesligado();
        btnCena2.setImageResource(R.drawable.ic_switch_pressed);

        switch1.setChecked(false);
        switch3.setChecked(false);
        switch5.setChecked(false);
        switch7.setChecked(false);
        switch8.setChecked(false);

        //Luminrias Fixas
        dim2.setProgress(A);
        dim4.setProgress(A);
        dim6.setProgress(200);
        dim9.setProgress(A);
        dim10.setProgress(A);

        // Definindo o Tunable White
        dim15.setProgress(1);
    }

    public void cena3(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena3.setImageResource(R.drawable.ic_switch_pressed);

        switch1.setChecked(false);
        switch3.setChecked(false);
        switch5.setChecked(false);
        switch7.setChecked(false);
```

```
        switch8.setChecked(false);
        switch11.setChecked(false); //Desliga o Tunable White

        //Luminrias Fixas
        dim2.setProgress(200);
        dim4.setProgress(0);
        dim6.setProgress(0);
        dim9.setProgress(A);
        dim10.setProgress(A);

        // Definindo o Tunable White
        dim15.setProgress(1);
    }

    public void cena4(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena4.setImageResource(R.drawable.ic_switch_pressed);


        switch2.setChecked(false);
        switch3.setChecked(false);
        switch5.setChecked(false);
        switch7.setChecked(false);
        switch8.setChecked(false);
        switch11.setChecked(false); //Desliga o Tunable White

        //Luminrias Fixas
        dim1.setProgress(200);
        dim4.setProgress(0);
```

```
        dim6.setProgress(0);
        dim9.setProgress(A);
        dim10.setProgress(A);

        // Definindo o Tunable White
        dim15.setProgress(1);
    }

    public void cena5(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena5.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch3.setChecked(false);
        switch5.setChecked(false);
        switch7.setChecked(false);
        switch8.setChecked(false);
        switch11.setChecked(false); //Desliga o Tunable White

        //Luminrias Fixas
        dim1.setProgress(0);
        dim4.setProgress(200);
        dim6.setProgress(0);
        dim9.setProgress(A);
        dim10.setProgress(A);

        // Definindo o Tunable White
        dim15.setProgress(1);
    }
```

```
public void cena6(View view){
    statusLuminaria(true);
    atualizaGeral(0);

    //Muda o estado do boto
    botaoDesligado();
    btnCena6.setImageResource(R.drawable.ic_switch_pressed);

    switch2.setChecked(false);
    switch4.setChecked(false);
    switch5.setChecked(false);
    switch7.setChecked(false);
    switch8.setChecked(false);
    switch11.setChecked(false); //Desliga o Tunable White

    //Luminrias Fixas
    dim1.setProgress(0);
    dim3.setProgress(200);
    dim6.setProgress(0);
    dim9.setProgress(A);
    dim10.setProgress(A);

    // Definindo o Tunable White
    dim15.setProgress(1);
}

public void cena7(View view){
    statusLuminaria(true);
    atualizaGeral(0);

    //Muda o estado do boto
    botaoDesligado();
```

```
        btnCena7.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch4.setChecked(false);
        switch5.setChecked(false);
        switch7.setChecked(false);
        switch8.setChecked(false);
        switch11.setChecked(false); //Desliga o Tunable White

        //Luminrias Fixas
        dim1.setProgress(0);
        dim3.setProgress(0);
        dim6.setProgress(200);
        dim9.setProgress(0);
        dim10.setProgress(A);

        // Definindo o Tunable White
        dim15.setProgress(1);
    }

    public void cena8(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena8.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch4.setChecked(false);
        switch6.setChecked(false);
        switch7.setChecked(false);
        switch8.setChecked(false);
```

```
        switch11.setChecked(false); //Desliga o Tunable White

        //Luminrias Fixas
        dim1.setProgress(0);
        dim3.setProgress(0);
        dim5.setProgress(200);
        dim9.setProgress(0);
        dim10.setProgress(A);

        // Definindo o Tunable White
        dim15.setProgress(1);
    }

    public void cena9(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena9.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch4.setChecked(false);
        switch6.setChecked(false);
        switch7.setChecked(false);
        switch8.setChecked(false);
        switch11.setChecked(false); //Desliga o Tunable White

        //Luminrias Fixas
        dim1.setProgress(A);
        dim3.setProgress(A);
        dim5.setProgress(0);
        dim9.setProgress(200);
```

```
        dim10.setProgress(200);

        // Definindo o Tunable White
        dim15.setProgress(1);
    }

    public void cena10(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena10.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch4.setChecked(false);
        switch6.setChecked(false);
        switch9.setChecked(false);
        switch10.setChecked(false);
        switch11.setChecked(false); //Desliga o Tunable White

        //Luminrias Fixas
        dim1.setProgress(A);
        dim3.setProgress(A);
        dim5.setProgress(0);
        dim7.setProgress(200);
        dim8.setProgress(200);

        // Definindo o Tunable White
        dim15.setProgress(1);
    }

    public void cena11(View view){
```

```
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena11.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch4.setChecked(false);
        switch6.setChecked(false);
        switch9.setChecked(false);
        switch10.setChecked(false);

        //Luminrias Fixas
        dim1.setProgress(A);
        dim3.setProgress(A);
        dim5.setProgress(200);
        dim7.setProgress(A);
        dim8.setProgress(A);

        // Definindo o Tunable White 3000K
        dim15.setProgress(1);
    }

    public void cena12(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena12.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
```

```
        switch4.setChecked(false);
        switch6.setChecked(false);
        switch9.setChecked(false);
        switch10.setChecked(false);

        //Luminrias Fixas
        dim1.setProgress(A);
        dim3.setProgress(A);
        dim5.setProgress(200);
        dim7.setProgress(A);
        dim8.setProgress(A);

        // Definindo o Tunable White 4000K
        dim15.setProgress(67);
    }

    public void cena13(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena13.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch4.setChecked(false);
        switch6.setChecked(false);
        switch9.setChecked(false);
        switch10.setChecked(false);


        //Luminrias Fixas
        dim1.setProgress(A);
```

```
        dim3.setProgress(A);
        dim5.setProgress(200);
        dim7.setProgress(A);
        dim8.setProgress(A);

        // Definindo o Tunable White 5000K
        dim15.setProgress(140);
    }

    public void cena14(View view){
        statusLuminaria(true);
        atualizaGeral(0);

        //Muda o estado do boto
        botaoDesligado();
        btnCena14.setImageResource(R.drawable.ic_switch_pressed);

        switch2.setChecked(false);
        switch4.setChecked(false);
        switch6.setChecked(false);
        switch9.setChecked(false);
        switch10.setChecked(false);


        //Luminrias Fixas
        dim1.setProgress(A);
        dim3.setProgress(A);
        dim5.setProgress(200);
        dim7.setProgress(A);
        dim8.setProgress(A);

        // Definindo o Tunable White 6000K
        dim15.setProgress(200);
```

```
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();

    if (id == R.id.nav_inicio) {
        // Inicio dos Handles para adicionar os itens na Lista
        switchToA();
    } else if (id == R.id.nav_grupos) {
        switchToB();
    } else if (id == R.id.nav_luminarias) {
        switchToC();
    } else if (id == R.id.nav_tools) {
        switchToE();
    } else if (id == R.id.nav_bluetooth) {
        if(conexao){//Verifica se existe alguma conexo
            //Desconectar a conexo atual
            try{
                meuSocket.close();
                conexao = false;
                //btnConexao.setText("Conectar BT");

                item.setTitle("Conectar");
                tools_status.setText("Offline");
```

```
                tools_mac.setText("00:00:00:00:00:00");
                Toast.makeText(getApplicationContext(), "Desconectado com Suces
            } catch (IOException erro){
                item.setTitle("Desconectar");
                Toast.makeText(getApplicationContext(), "Ocorreu um erro!", Toa
            }


        } else {
            //Conectar ento em uma rede...
                Intent abreLista = new Intent(MainActivity.this, ListaDispositi
                startActivityForResult(abreLista, SOLICITA_CONEXAO);
                item.setTitle("Desconectar");
        }
    }

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}

@Override
public void switchToA() {
    findViewById(R.id.include_luminarias).setVisibility(View.INVISIBLE);
    findViewById(R.id.include_grupos).setVisibility(View.INVISIBLE);
    findViewById(R.id.include_cenarios).setVisibility(View.VISIBLE);
    findViewById(R.id.include_tunable).setVisibility(View.INVISIBLE);
    findViewById(R.id.include_tools).setVisibility(View.INVISIBLE);
    pageName.setText("Cenrios");
}

@Override
public void switchToB() {
```

```
        findViewById(R.id.include_luminarias).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_grupos).setVisibility(View.VISIBLE);
        findViewById(R.id.include_cenarios).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_tunable).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_tools).setVisibility(View.INVISIBLE);
        pageName.setText("Grupos");
    }

    @Override
    public void switchToC() {
        findViewById(R.id.include_luminarias).setVisibility(View.VISIBLE);
        findViewById(R.id.include_grupos).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_cenarios).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_tunable).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_tools).setVisibility(View.INVISIBLE);
        pageName.setText("Luminrias");
    }

    @Override
    public void switchToD() {
        findViewById(R.id.include_luminarias).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_grupos).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_cenarios).setVisibility(View.INVISIBLE);
        findViewById(R.id.include_tunable).setVisibility(View.VISIBLE);
        findViewById(R.id.include_tools).setVisibility(View.INVISIBLE);
        pageName.setText("Tunable White");
    }

    @Override //Chamado em caso de ativao no programa
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        //super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode){
            case SOLICITA_ATIVACAO: //Chamado na inicializao do APK
```

```
        if(resultCode == Activity.RESULT_OK){ //Se o resultado da Solciita
            Toast.makeText(getApplicationContext(), "O Bluetooth foi ativad
        } else { //Se No vem para este MENU
            Toast.makeText(getApplicationContext(), "O Bluetooth no foi ati
            finish();
        }
        break;

    case SOLICITA_CONEXAO: //Chamado na inicializao do APK
        if(resultCode == Activity.RESULT_OK){ //Se o resultado da conexo f
            Toast.makeText(getApplicationContext(), "Dispositivo conectado
            MAC = data.getExtras().getString(ListaDispositivos.ENDERECO_MAC
            //Toast.makeText(getApplicationContext(), "MAC Final: " + MAC,

            tools_mac.setText(MAC);

            meuDevice = meuBluetoothAdapter.getRemoteDevice(MAC); //Recebe

            try {
                meuSocket = meuDevice.createRfcommSocketToServiceRecord(MEU

                meuSocket.connect(); //Conecta no Socket

                conexao = true;

                connectedThread = new ConnectedThread(meuSocket); //Inician
                sleep(100);
                connectedThread.start();

                tools_status.setText("CONECTADO!");
                //btnConexao.setText("Desconectar");
                //textView.setText("Desconectar");
                Toast.makeText(getApplicationContext(), "Conectado com suce
```

```
                } catch (IOException erro){

                    conexao = false;

                    tools_status.setText("ERROR");

                    Toast.makeText(getApplicationContext(), "Ocorreu um erro!",
                }

            } else { //Se No vem para este MENU
                Toast.makeText(getApplicationContext(), "No foi possvel conecta
            }
            break;

    }
}

//Parte da Comunicao do BT
private class ConnectedThread extends Thread {

    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket) {

        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        // Get the input and output streams, using temp objects because
        // member streams are final
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
```

```
        } catch (IOException e) { }

        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }

    public void run() {
        byte[] buffer = new byte[1024];  // buffer store for the stream
        int bytes; // bytes returned from read()

        // Keep listening to the InputStream until an exception occurs
        while (true) {
            try {
                // Read from the InputStream
                bytes = mmInStream.read(buffer);

                String dadosBluetooth = new String(buffer, 0, bytes); //Linha p


                // Send the obtained bytes to the UI activity
                mHandler.obtainMessage(MESSAGE_READ, bytes, -1, dadosBluetooth)
            } catch (IOException e) {
                break;
            }
        }
    }

    /* Call this from the main activity to send data to the remote device */
    public void enviarBT(String dadosEnviar) {
        byte[] msgBuffer = dadosEnviar.getBytes();

        try {
            tools_send.setText(dadosEnviar);
```

```
                mmOutStream.write(msgBuffer);
                //sleep(40);
            } catch (IOException e) {

            }
        }

    }
}
```

11 APÊNDICE F - CÓDIGO UTILIZADO NO MICROCONTROLADOR

```
#include "TimerOne.h"
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

  char caractere = ' ';
  int clockSerial = 0;
  int inicio = 0;
  int checksum = 0;
  int checksuminicio = 0;
  int checksumpacote = 100;
  int fim = 0;
  int i = 0;

  int number = 0;

  String recebido = "";
  int recebidoNum = 0;
 int percentual = 0;
  int timeout = 0;
  int fadetime = 0;                                    //Fadetime in ms


int autoriza = 0;

void setup() {
      Timer1.initialize(1000); // Inicializa o Timer1 e configura para um perodo d
      Timer1.attachInterrupt(task); // Configura a funo callback() como a funo par

      Serial.begin(57600); //Serial de comunicao
```

```
      Serial2.begin(57600); //Serial do Bluetooth

      Serial.println("LumiSense Inicializando!");

void RecebePacote(){

    if(timeout<1000){
          timeout++;
    }

      if(clockSerial>200){
        clockSerial = 0;
        Serial2.print("{FRAG1}");
      }

        if (Serial2.available()) {     // If anything comes in Serial1 (pins 0 & 1
          timeout = 0;
          caractere = Serial2.read(); //Armazena a variavel lida da Serial para tr
      }

      if((inicio == 1)&&(caractere!=':')&&(caractere!='+')&&(caractere != '{')&&(ca
        recebido.concat(caractere);
      }

      if(caractere == '{')

      if(caractere == '+'){
          checksuminicio = 1; //Define o inicio da verificao do Pacote
      }

      if(caractere == ':'){
        if((checksuminicio==0)&&(caractere!='+')){
            number = (int) strtol(&recebido[0], NULL, 16); //Funo usada para conve
```

```
        if(checksuminicio==0)
        checksum = checksum + number;

            if((i<15)&&(number>=0)&&(number<=255)){ //Condio para adicionar de
              if(number<255){
                dimmeraux[i] = ((((200-(float)number)/255))*3250);
              }
              else{
                dimmeraux[i] = 4092;
              }
                //Serial.println(dimmeraux[i]);
            }

        i++;
        recebido = "";
        recebidoNum = 0;
    }
    if((checksuminicio==1)){
      checksumpacote = (int) strtol(&recebido[0], NULL, 16);
        recebido = "";
        recebidoNum = 0;
    }
}

if((caractere=='{')&&(fim==0)&&(i>0)&&(checksuminicio==1))
{
    inicio = 1;
    fim = 0;
    i = 0;
    checksuminicio = 0;
    checksum = 0;
}
```

```
        if((caractere == '}')&&(recebido=="FIM")&&(checksum!=checksumpacote)){
          inicio = 0;
          fim = 1;
          i = 0;
          recebido = "";
          autoriza == 0;
          checksuminicio = 0;
          checksum = 0;
          Serial.println("ERROR!\n");
        }
        caractere = ' ';
}

void loop() {
        if(autoriza == 1){
                  autoriza = 0;
                  dimmer[0] = dimmeraux[0];
                  dimmer[1] = dimmeraux[1];
                  dimmer[2] = dimmeraux[2];
                  dimmer[3] = dimmeraux[3];
                  dimmer[4] = dimmeraux[4];
                  dimmer[5] = dimmeraux[5];
                  dimmer[6] = dimmeraux[6];
                  dimmer[7] = dimmeraux[7];
                  dimmer[8] = dimmeraux[8];
                  dimmer[9] = dimmeraux[9];
                  dimmer[10] = dimmeraux[10];
                  dimmer[11] = dimmeraux[11];
                  dimmer[12] = dimmeraux[12];
                  dimmer[13] = dimmeraux[13];
                  dimmer[14] = dimmeraux[14];
                  dimmer[15] = dimmeraux[15];
```

```
}

if(fadetime>25000){
  //Serial.println(F("Dimmer enviado!\n"));
  pwm.setPWM(0, 0, dimmer[0]);
  pwm.setPWM(1, 0, dimmer[1]);
  pwm.setPWM(2, 0, dimmer[2]);
  pwm.setPWM(3, 0, dimmer[3]);
  pwm.setPWM(4, 0, dimmer[4]);
  pwm.setPWM(5, 0, dimmer[5]);
  pwm.setPWM(6, 0, dimmer[6]);
  pwm.setPWM(7, 0, dimmer[7]);
  pwm.setPWM(8, 0, dimmer[8]);
  pwm.setPWM(9, 0, dimmer[9]);
  pwm.setPWM(10, 0, dimmer[10]);
  pwm.setPWM(11, 0, dimmer[11]);
  pwm.setPWM(12, 0, dimmer[12]);
  pwm.setPWM(13, 0, dimmer[13]);
  pwm.setPWM(14, 0, dimmer[14]);
  pwm.setPWM(15, 0, dimmer[15]);

  Serial.print(dimmer[0]);
  Serial.print(" ");
  Serial.print(dimmer[1]);
  Serial.print(" ");
  Serial.print(dimmer[2]);
  Serial.print(" ");
  Serial.print(dimmer[3]);
  Serial.print(" ");
  Serial.print(dimmer[4]);
  Serial.print(" ");
  Serial.print(dimmer[5]);
  Serial.print(" ");
```

```
        Serial.print(dimmer[6]);
        Serial.print(" ");
        Serial.print(dimmer[7]);
        Serial.print(" ");
        Serial.print(dimmer[8]);
        Serial.print(" ");
        Serial.print(dimmer[9]);
        Serial.print(" ");
        Serial.print(dimmer[10]);
        Serial.print(" ");
        Serial.print(dimmer[11]);
        Serial.print(" ");
        Serial.print(dimmer[12]);
        Serial.print(" ");
        Serial.print(dimmer[13]);
        Serial.print(" ");
        Serial.println(dimmer[14]);
        //Serial.print(" ");
        //Serial.println(dimmer[15]);
        fadetime = 0;
      }

        fadetime++;
}

void task(){
  RecebePacote();
}
```