UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

Gustavo Ceolin

SISTEMA DE CONTROLE DE ILUMINAÇÃO INTELIGENTE

Santa Maria, RS
2019

Gustavo Ceolin

SISTEMA DE CONTROLE DE ILUMINAÇÃO INTELIGENTE

Trabalho de Graduação apresentado ao curso de Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de **Bacharel em Engenharia de Computação.**

Orientador: Prof. Dr. José Eduardo Baggio

Santa Maria, RS
2019

Gustavo Ceolin

SISTEMA DE CONTROLE DE ILUMINAÇÃO INTELIGENTE

Trabalho de Graduação apresentado ao curso de Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de **Bacharel em Engenharia de Computação.**

Aprovado em 19 de julho de 2019:

_____

José Eduardo Baggio, Dr. (UFSM)
(Presidente/Orientador)

_____

Carlos Henrique Barriquello, Dr. (UFSM)

_____

Rafael Vedovato Rubin

Santa Maria, RS
2019

AGRADECIMENTOS

A meu professor e orientador, Dr. José Eduardo Baggio, pela amizade, ensinamentos e oportunidade de crescimento pessoal e profissional ao longo desses anos de convívio.

Aos meus grandes amigos: Kaciane Marques, Larissa Calegaro Sarturi e Winnie Silva pelo apoio, amizade e agradável convívio, que tornou o desenvolvimento desse trabalho muito mais fácil.

A todos os demais amigos e amigas, cuja paciência apoio e compreensão contribuíram para o desenvolvimento deste trabalho.

Aos meus pais, Nicanor e Marlene Ceolin, pela dedicação e constante incentivo ao desenvolvimento intelectual de seus filhos.

A minhas irmãs Lucieli e Silvana, por toda a alegria e força, mesmo que muitas vezes à distância.

A todas as pessoas e instituições que contribuíram direta ou indiretamente para a conclusão deste trabalho.

RESUMO

SISTEMA DE CONTROLE DE ILUMINAÇÃO INTELIGENTE

AUTOR: Gustavo Ceolin
ORIENTADOR: Prof. Dr. José Eduardo Baggio

Este trabalho apresenta a implementação de sistemas de controle de iluminação de lâmpadas incandescentes. Foi desenvolvido um dispositivo composto por um microcontrolador NodeMCU, um sensor de iluminação, e um TRIAC, que pode controlar o brilho de uma lâmpada, através do controle do ângulo de disparo do TRIAC. O sistema proposto permite monitoramento remoto e gerenciamento de iluminação em um local, através de um sistema que aciona e controla o brilho de uma lâmpada. O usuário poderá ajustar o brilho da lâmpada manualmente, por meio de um aplicativo Android desenvolvido, ou ativar o modo automático, onde é possível se especificar a quantidade de iluminação desejada no ambiente. No modo automático, o microcontrolador NodeMCU detecta o brilho do seu ambiente através do sensor de luz TEMT6000 e, juntamente com um sistema de controle PI, define o brilho ideal para a lâmpada, de forma a complementar a iluminação já existente na sala, mantendo a iluminação constante, compensando automaticamente alguma alteração em qualquer outra fonte de luz presente na sala. Estes sistemas energeticamente eficientes proporcionam uma intensidade de iluminação constante, permitem maior conforto e reduzem o consumo de energia.

Palavras-chave: Microcontrolador. NodeMCU. ESP-12. ESP8266. Lâmpada. Controlador PI. Controle de Luminosidade. TRIAC.

ABSTRACT

SMART LIGTHING CONTROL SYSTEM

AUTHOR: Gustavo Ceolin
ADVISOR: Prof. Dr. José Eduardo Baggio

This study presents the implementation of a lighting control systems of incandescent lamps. We have developed a device composed by a NodeMCU microcontroller, an illumination sensor, and a TRIAC, which can control the brightness of a lamp by controlling the TRIAC firing angle. The proposed system allows remote monitoring and lighting management of a place, through a system that drives and controls the brightness of a lamp. The user will be able to adjust the brightness of the lamp manually, through a developed Android application, or activate the automatic mode, which allows you to specify the amount of illumination desired in the environment. In the automatic mode, the NodeMCU microcontroller will detect the brightness of its surroundings by TEMT6000 Light Sensor and together with a PI control system will set the ideal brightness for the lamp. This energy-efficient systems provide a constant illumination intensity, allowed greater comfort and reducing energy usage.

Keywords: Microcontroller. NodeMCU. ESP-12. ESP8266. Lamp. PI control. Lighting control. TRIAC.

LISTA FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AMQP	*Advanced Message Queuing Protocol*
CoAP	*Constrained Application Protocol*
HTTP	*Hypertext Transfer Protocol*
LDR	*Light Dependent Resistor*
LED	*Light Emitting Diode*
MIT	*Massachusetts Institute of Technology*
MQTT	*Message Queuing Telemetry Transport*
PI	Controlador Proporcional Integral
PID	Controlador Proporcional Integral Derivativo
TRIAC	*Triode for Alternating Current*
USB	*Universal Serial Bus*
XMPP	*Extensible Messaging and Presence Protocol*

SUMÁRIO

1 INTRODUÇÃO

A rápida evolução da tecnologia gera a oportunidade de proporcionar maior qualidade de vida, podendo aumentar o bem-estar, promover mais segurança e racionalizar o consumo de energia. Pensando nesses fatores surge a ideia de automatizar uma residência, facilitando ações realizadas no dia a dia. A automação residencial vem ganhando espaço no mercado nos últimos tempos e o avanço da tecnologia nos permite, cada vez mais, desenvolver sistemas de automação residencial com um alto nível de controle, utilizando sistemas embarcados, que estão cada vez mais potentes.

A variedade e quantidade de dispositivos que podem ser automatizados em uma residência são enormes, como por exemplo controle de iluminação, controle de acesso a residência por biometria, acionamento de periféricos por comando de voz e muito mais. O foco deste projeto será na automatização do sistema de iluminação.

O sistema de iluminação das residências comuns se dá pelo uso de interruptores físicos, que permite apenas acender e apagar as luzes. Com a automação é possível controlar por um smartphone ou tablet não só o apagar e acender das luzes, como também o seu brilho, ou então, configurar para que tudo seja feito de forma automática com o principal objetivo de aumentar o conforto dos moradores.

Em um sistema de automação residencial deve-se ter também preocupações com o consumo de energia elétrica, pois é um recurso relativamente caro e limitado. Além de gerar comodidade e conforto, a automatização do sistema de iluminação também é eficaz para a economia de energia, já que o brilho das lâmpadas pode ser ajustado automaticamente, através de *dimmers* que controlam sua tensão, conforme necessário.

No mercado, hoje em dia, existem poucas opções à venda de *dimmers* que funcionem de forma remota, e estas poucas opções geralmente têm custo bastante elevado. Visando obter soluções para essas questões, este projeto propõem a criação de um sistema de iluminação, controlado por meio de um aplicativo Android, com a capacidade de ligar, desligar, ajustar o brilho e definir modo de funcionamento automático para uma lâmpada.

1.1 Motivação

Em 1879, Thomas Edison inventava a lâmpada elétrica incandescente. A partir daí mudanças impactantes aconteceram na sociedade: conforto, facilidade e desenvolvimento tecnológico foram as que mais se destacaram. Desse momento em diante as lâmpadas elétricas se tornaram cada vez mais comuns e acessíveis, e hoje em dia dificilmente se encontram residências ou estabelecimentos sem lâmpadas elétricas. Em vista disso, o aperfeiçoamento da iluminação gerada por lâmpadas é de suma importância para o conforto da população e para a economia de energia elétrica, de modo a utilizar esse recurso da forma mais eficiente possível.

Iluminação residencial está diretamente ligada ao conforto, produtividade e consumo de energia elétrica. Tendo isso em mente, é possível implementar um sistema que atenda todas essas necessidades.

1.2 Objetivo

O objetivo geral deste trabalho consiste em fazer a dimerização manual e automática de uma lâmpada implementando um sistema de controle, no qual, através de sensores que detectam a luminosidade do ambiente, o brilho da lâmpada seja ajustado automaticamente, de forma a manter a iluminação ambiente constante, mesmo diante de alteração em outra fonte de iluminação. Com um sistema automático desses, a iluminação ambiente pode manter as luzes com pouco brilho, quando se tem sol entrando pela janela, e na medida que a iluminação solar vai diminuindo as lâmpadas vão aumentando o brilho de forma a manter a iluminação ambiente conforme desejado. Este projeto também tem como objetivo, desenvolver um aplicativo Android em que seja possível controlar remotamente as funções descritas.

1.3 Estrutura do trabalho

Este trabalho está dividido em quatro capítulos. O capítulo 2 trata da fundamentação teórica, e contém informações referentes a recursos e dispositivos utilizados no projeto.

No capítulo 3 é apresentado o desenvolvimento do trabalho, que se divide em quatro subtópicos. Inicia com a parte do hardware, mostrando o circuito projetado e explicando os passos necessários para desenvolvê-lo, explicando seu funcionamento. Logo após vem a parte do software que explica onde e como foi possível fazer a programação do código para o

microcontrolador. Ainda no desenvolvimento temos o subtópico que descreve o servidor MQTT utilizado no projeto e explica como configurá-lo e utilizá-lo. Por final o ultimo subtópico explica e mostra o aplicativo Android desenvolvido para o projeto.

No capítulo 4 e 5 apresentam-se respectivamente os resultados obtidos com o projeto e na conclusão temos as considerações finais.

2 REVISÃO LITERÁRIA

2.1 Sistemas de Controle

Sistemas de controle são utilizados para controlar o comportamento de um sistema em que a saída dependa da entrada. Existem dois tipos de sistema de controle, o sistema de malha aberta e o sistema de malha fechada.

Os sistemas de malha aberta não possuem realimentação, ou seja, não pode realizar compensação caso ocorra alguma perturbação no sistema. Este sistema depende exclusivamente da sua entrada (NISE, 2012, p36).

Sistemas de malha fechada são aqueles que possuem um controle de realimentação, ou seja, as perturbações causadas no sistema podem ser corrigidas através da diferença entra a sua entrada (referência) e a saída do sistema. Essa diferença é chamada de erro. O sistema de malha fechada compensa o efeito das perturbações realimentando o sistema através da malha de realimentação (NISE, 2012, p37).

2.1.1 Controladores

A maior parte da teoria de controle automático estudadas hoje é atribuída a Nicholas Minorsky. Seu desenvolvimento teórico aplicado à condução automática de navios levou ao que hoje é chamado de controladores proporcional, integral e derivativo (PID). (NISE, 2012, pg33).

Controlador é um recurso no sistema de controle em malha fechada que usa o erro como realimentação. São utilizados para fazer com que um sistema tenha o funcionamento e desempenho determinado. Existem três principais controladores, que são: proporcional, integral e derivativo. O foco neste projeto será no controlador PI.

2.1.2 Controlador PI

O controlador proporcional integral (PI), como o nome já diz, é a junção dos controladores proporcional e integral. O controlador proporcional é essencialmente um amplificador com ganho ajustável (OGATA, 2014, pg21). O controlador integral tem a função de eliminar o desvio do *offset* tendo o objetivo de manter a variável próximo ao valor desejado, mesmo após algum distúrbio (FREITAS, 2014).

"A atuação de um controlador PI corresponde à soma de uma ação proporcional com uma ação integral. Desta forma pode-se melhorar a resposta transitória com a contribuição da ação proporcional, enquanto a ação integral corrige o erro de estado estacionário." (ARAÚJO, F. M. U., 2007, p.21).

A ação do controle PI é definida pela fórmula (1):

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^1 e(t)dt \tag{1}$$

ou, então, a função de transferência:

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{K_i}{s}\right) \tag{2}$$

(OGATA, 2014, pg21).

2.2 Microcontrolador

Microcontrolador é definido em SOUZA (2005, pg21) como um pequeno componente eletrônico, que possui uma inteligência programável, utilizado no controle de periféricos, tais como LEDs, displays, relés, sensores diversos, entre outros. Afirma ainda que o microcontrolador é programável, pois a lógica de operação é estruturada na forma de um programa e gravada dentro do componente.

A vantagem de se usar um microcontrolador é a possibilidade de programá-lo, para que se adapte à finalidade desejada, possibilitando o desenvolvimento de diversos projetos de hardware e software.

2.2.1 ESP8266

O ESP8266 é um chip, desenvolvido pela empresa Espressif Systems e foi lançado no mercado em meados de agosto de 2014. Além de ser compacto, o grande diferencial do ESP8266 é que possui integrado um sistema de comunicação Wi-Fi próprio.

A tabela 1 abaixo mostra as especificações técnicas do ESP8266 (ESPRESSIF, 2018).

Voltagem de operação	2.5V - 3.6V
Consumo de corrente	80 Ma
CPU	Tensilica L106 32-bit processor
Frequência	80 – 160 MHz
Memória RAM	32K + 80K
Conectividade	802.11 b/g/n

Tabela 1 - Especificações do chip ESP8266

Com o objetivo de facilitar o uso do chip ESP8266, vários fabricantes utilizam dos recursos presentes no chip para criar módulos e placas. Esses módulos variam principalmente em tamanho, número de pinos e tipo de conexão com o computador. A placa de desenvolvimento utilizada neste projeto é a NodeMCU. Esta placa utiliza o módulo ESP-12, como pode ser visto na Figura 1 abaixo:

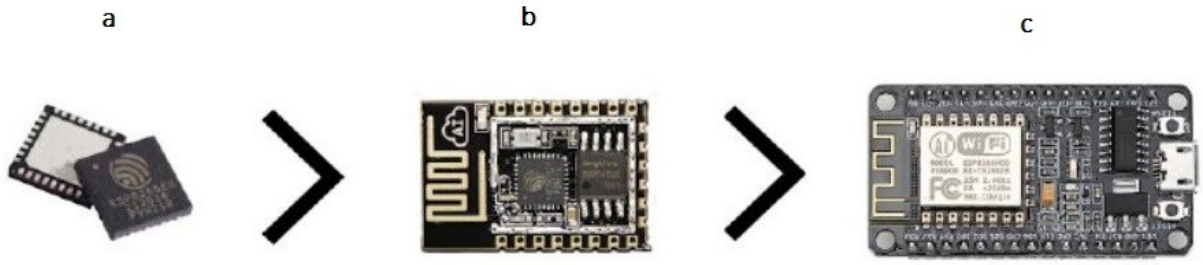

Figura 1 - (a) chip ESP8266, (b) módulo ESP-12, (c) Placa NodeMCU

2.2.2 NODEMCU-ESP12

O módulo NODEMCU-ESP12 é uma placa criada com o objetivo de facilitar o desenvolvimento de aplicações para o módulo ESP8266 ESP-12. Esta placa contém todos os circuitos necessários para fazer o modulo ESP-12 funcionar, como interface Serial-USB, regulador de tensão e botões de controle e pinos que permitem o seu uso em Protoboards.

A alimentação na placa NodeMCU pode ser feita pelo conector USB (5V) ou então pelo pino VIN com uma tensão máxima de 9V. A placa contém um regulador de tensão (AMS1117), responsável por garantir que o módulo ESP-12 não queime, já que ele trabalha apenas com tensão de 3,3V.

O esquema da placa NodeMCU e sua pinagem pode ser visto na figura 2 abaixo:

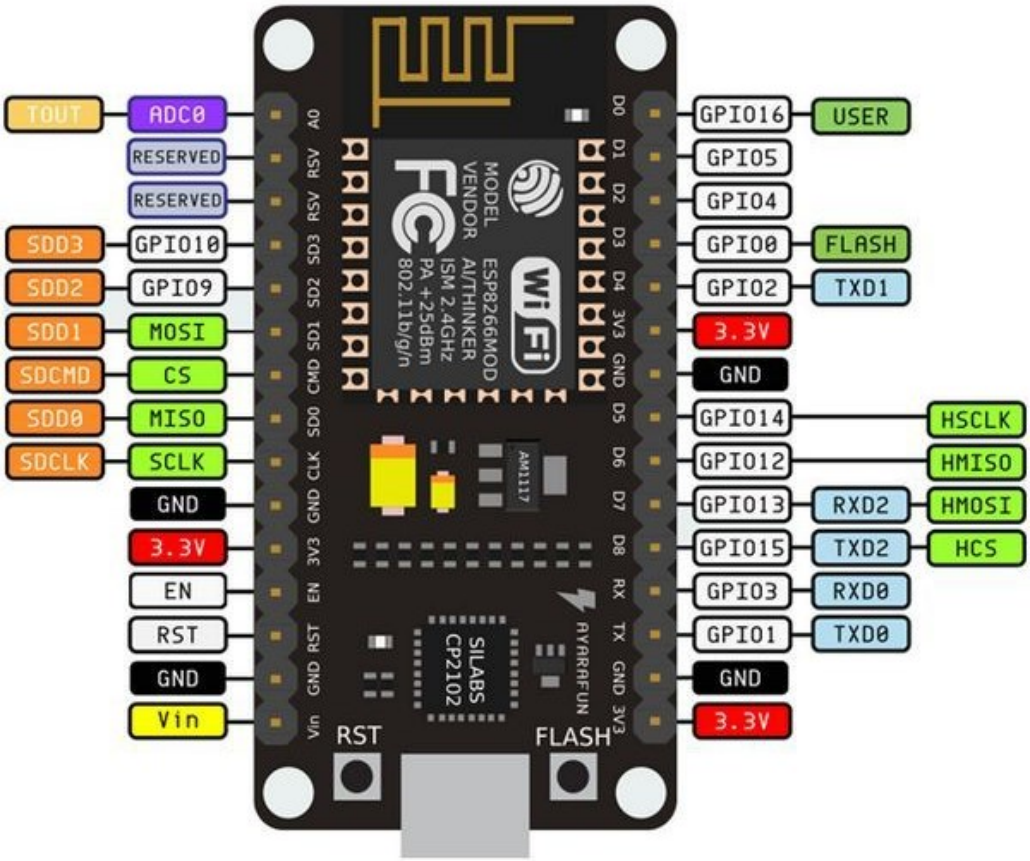

Figura 2 - Pinagem da placa NodeMCU

Os pinos utilizados no projeto e suas respectivas funções estão na tabela 2 abaixo:

Nome	Descrição
VIN	Alimentação de 3,3V
ADC	Entrada analógico/Digital
GPIO4	Pino D2 de entrada ou saída
GPIO5	Pino D1 de entrada ou saída
GND	Aterramento

Tabela 2 - Pinos utilizados da placa NodeMCU

2.3 Optoacoplador

Um optoacoplador, ou optoisolador, combina um LED infravermelho e um fotodiodo em um único encapsulamento. Tem um LED no lado da entrada e um fotodiodo na saída. (BOYLESTAD, 2013, pg731).

O optoacoplador funciona da seguinte forma: A corrente elétrica passa pelo LED infravermelho, e ao fazer isso aciona o fotodiodo, permitindo a passagem de corrente elétrica entre os terminais do lado oposto. Dessa forma os dois lados do optoacoplador ficam isolados eletricamente. O símbolo do optoacoplador pode ser visto na figura 3.

A principal vantagem de um optoacoplador é o isolamento elétrico estabelecido entre os circuitos de entrada e saída. Como são dispositivos que operam por meio de um feixe de luz, tem a capacidade de transmitir sinais de um circuito para outro, sem a ligação elétrica.

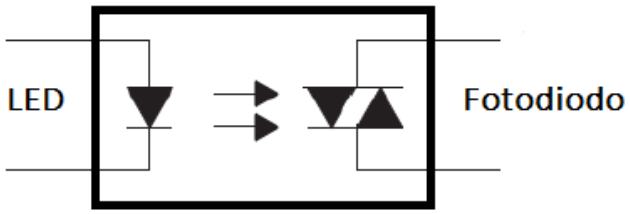

Figura 3 - Esquema elétrico de um optoacoplador

2.4 Ponte retificadora

Uma ponte retificadora converte tensão alternada em tensão contínua. Segundo Malvino (2016, pg 97), a ponte retificadora consiste em quatro diodos devidamente posicionados estrategicamente como se pode ver na figura 4. Os diodos D1 e D2 conduzem durante o semiciclo positivo, e D3 e D4 conduzem durante o semiciclo negativo. Durante os dois semiciclos, a tensão e a corrente na carga têm o mesmo sentido. Desta forma, a tensão que entrou alternada, aparece sempre positiva e no mesmo sentido na saída da ponte retificadora.

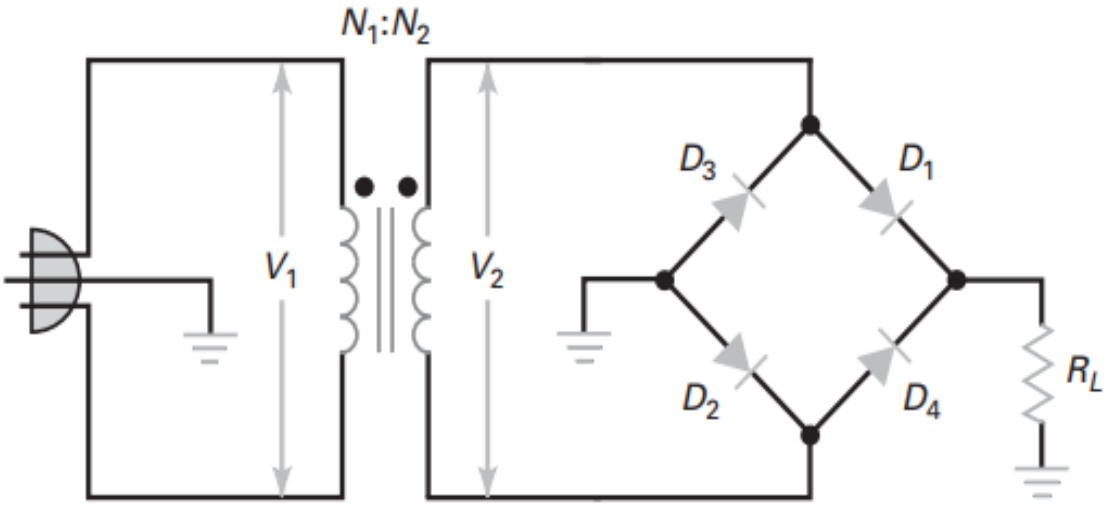

Figura 4 - Esquema de uma ponte retificadora. Retirado de Malvino (2016, pg. 98)

2.5 TRIAC

TRIAC é um componente eletrônico com a capacidade de conduzir corrente elétrica de forma bidirecional. É muito comum sua utilização para o controle de fase ou potência. Segundo Schuler (2013, pgG18) o TRIAC é um dispositivo bidirecional de controle em onda completa que se equivale à conexão de dois retificadores controlados a silício em antiparalelo.

Os Dimmers com TRIAC são utilizados em baixa potência e têm atuação no controle de sistemas de iluminação. Esse controle eletrônico do dimmer com TRIAC possibilitou o controle remoto da intensidade luminosa em diversos ambientes, tais como: cinemas, teatros, residências e entre outros.

O TRIAC é composto por três terminais, MT1, MT2 e *gate*, de acordo com a figura 5. O seu funcionamento é relativamente simples, ao receber uma tensão ou corrente em seu *gate*, o TRIAC começa a conduzir corrente entre os terminais MT1 e MT2. A condução entre os terminais permanecerá até que a corrente que circula nele seja zero.

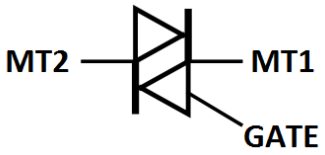

Figura 5 - Esquema de um TRIAC

2.6 Fonte de Tensão

A fonte de tensão nada mais é do que um conversor de tensão. Neste projeto foi utilizado a fonte bi-volt Hi-link HLK-PM03, que tem a capacidade de converter os 220V AC da rede local para os 3,3V CC, necessários para a alimentação da placa NodeMCU e os optoacopladores do circuito.

2.7 Sensor de luminosidade TEMT6000

O Sensor de Luz Ambiente TEMT6000 é um sensor baseado em um fototransistor do tipo NPN, capaz de detectar a luz ambiente de uma forma simples e muito mais precisa que os sensores que utilizam LDR para detecção de luz. O sensor envia um sinal analógico proporcional à quantidade de luz. Quanto maior a incidência de luz, maior é o valor na saída. O Sensor de Luz Ambiente TEMT6000 é capaz de trabalhar com tensões de 3,3V à 5V. O sensor e seu esquema elétrico podem ser visto na figura 6.

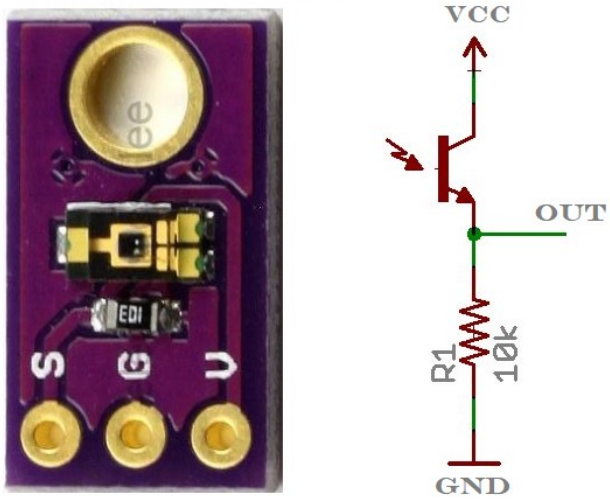

Figura 6 - Sensor TEMT6000 e seu respectivo esquemático

2.8 Internet das coisas

Hoje em dia estamos rodeados de dispositivos que suportam alguma forma de tecnologia de computação e comunicação. Estes dispositivos são por exemplo, smartphones, tablets, notebooks, sensores, entre outros. A tendência é que, com o passar do tempo, estes dispositivos

interajam cada vez mais entre si, por meio de uma rede de comunicação como a internet. A interação entre diferentes dispositivos tem o objetivo de obter comodidade e praticidade, facilitando o dia a dia das pessoas.

Existem divergências em relação ao conceito de Internet das Coisas (*Internet of Things* – IoT), não há um conceito único considerado unânime. De modo geral, entende-se como um ambiente de objetos e pessoas interconectados com uma rede unificada, como a internet, por meio de variados tipos de sensores.

O termo Internet das Coisas foi usado pela primeira vez por Kevin Ashton, do MIT, em 1999. De acordo com Ashton, as pessoas precisam se conectar com a internet devido à falta de tempo provocada pela rotina do novo cotidiano. Deste modo será possível armazenar dados com uma precisão cada vez maior. Ashton ainda fala que esses registros podem ser úteis, por exemplo, para a economia de recursos naturais e energéticos (Ashton, 2009).

O fato é que não se pode dizer ao certo o que o futuro reserva para este mundo em que pessoas e dispositivos estão cada vez mais conectados. A Internet das Coisas, já é uma realidade, mas certamente está longe de atingir seu máximo potencial em prol da humanidade.

2.9 Protocolos de comunicação

Protocolos de comunicação podem ser definidos como conjuntos de normas que permitem que dois ou mais dispositivos se comuniquem entre si via internet. Estes protocolos são responsáveis por pegar os dados a serem transmitidos pela rede e dividi-los em pedaços pequenos, chamados pacotes. Os pacotes carregam em si informações sobre o endereçamento de origem e destino.

Em muitos casos os dispositivos que fazem parte da IoT são criados para fins específicos e não possuem um hardware muito complexo. Por este motivo os protocolos usados em IoT costumam ser mais simples.

Existem diversos tipos de protocolos de comunicação voltados a IoT que são bastante utilizados atualmente, como: *Messsage Queue Telemetry Transport* (MQTT), *Constrained Application Protocol* (CoAP), *Extensible Messaging and Presence Protocol* (XMPP) e *Advanced Message Queuing Protocol* (AMQP) (KARAGIANNIS, V). O protocolo utilizado neste projeto será o MQTT, por sua simplicidade e praticidade de utilização.

2.9.1 MQTT

O protocolo *Messsage Queue Telemetry Transport* (MQTT) foi criado e desenvolvido pela IBM na década de 90. O MQTT é um protocolo leve e flexível para comunicação, baseado no Protocolo de Controle de Transmissão (TCP). O MQTT é um protocolo de mensagem com suporte para a comunicação assíncrona entre as partes conectadas, portanto seu uso é possível em ambientes de rede que não são confiáveis (Yuan M, 2017).

O protocolo MQTT funciona com a utilização de um *broker* que pode conectar inúmeros dispositivos, chamados clientes. O *broker* é um servidor que recebe as mensagens de todos os clientes conectados e, em seguida, encaminha essas mensagens para seus clientes destino (Yuan M, 2017).

O funcionamento do MQTT é baseado no princípio de publicação e leitura de mensagens em tópicos, ou seja, sempre que um cliente enviar um dado ao *broker*, ele deve especificar em qual tópico a informação deve ser enviada e, toda a aplicação que quiser receber informação, deverá se inscrever no tópico desejado. Sempre que o *broker* detectar alguma atualização nas informações em um de seus tópicos, todos os clientes desse tópico serão atualizados com esta nova informação.

3 DESENVOLVIMENTO

O desenvolvimento do presente projeto dividiu-se em três etapas, o projeto do hardware, a implementação do software e o aplicativo Android. Para o hardware foi elaborado um circuito com a capacidade de controlar a tensão de uma lâmpada dimerizável. Este controle só é possível graças a um microcontrolador interligado ao circuito. Este microcontrolador tem o recurso de conexão WI-FI, que é utilizado para conectar-se a um servidor MQTT, com o qual haverá troca de informações.

Para que o microcontrolador e o circuito funcionem da maneira desejada, foi implementado um software. Este software foi feito na plataforma Arduino IDE e gravado na memória interna do microcontrolador. O software é responsável pelo controle do funcionamento do circuito e pela conexão do microcontrolador com o servidor MQTT.

O projeto conta com dois modos de operação, modo manual e modo automático. No modo manual o usuário pode ligar, desligar e controlar a tensão da lâmpada através de um aplicativo de smartphone desenvolvido para o projeto. Neste mesmo aplicativo, é possível ativar

o modo automático, em que o usuário deve apenas definir o número de lux desejado no ambiente. Após a quantidade de lux ser definida pelo usuário, a lâmpada terá sua tensão regulada automaticamente para alcançar a luminosidade ambiente desejada. O aplicativo se conecta indiretamente com o microcontrolador, através do servidor MQTT, tornando possível controlar a lâmpada de qualquer lugar do mundo.

3.1 Hardware

O hardware foi projetado tem como objetivo controlar a tensão de uma lâmpada dimerizável. O esquemático completo do circuito foi desenvolvido através do site EasyEDA[1] e pode ser visto na figura 7.

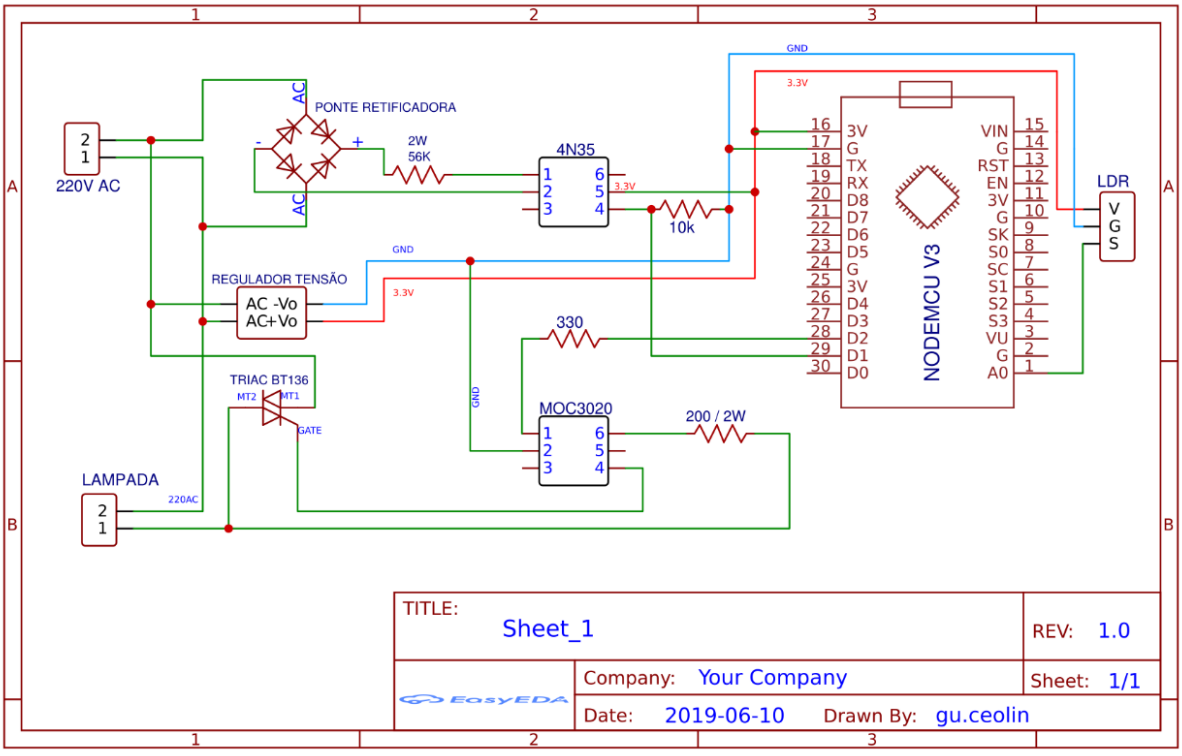

Figura 7 - Esquemático completo do circuito. Criado no site EasyEDA

O circuito conta basicamente com o microcontrolador NodeMCU, uma fonte que converte a tensão CA de entrada de 220V para CC 3,3V, TRIAC e outros componentes eletrônicos.

[1] https://easyeda.com

O principal responsável por controlar a potência da lâmpada e consequentemente o seu brilho é o TRIAC. Neste projeto o TRIAC utilizado foi o BT136, que suporta tensão de até 600V entre seus *Mains Terminals* (MTs). Como explicado anteriormente, no tópico 2.5, o TRIAC começa a conduzir corrente entre seus MTs quando recebe uma tensão ou corrente em seu *gate*, e a condução entre os MTs só encerra quando esta tensão for igual a zero. Com isso é possivel controlar o ângulo de disparo do TRIAC, possibilitando controlar a tensão média de saída.

Para calcular o ângulo de disparo do TRIAC precisa-se saber que a tensão da rede elétrica brasileira opera a uma frequência de 60Hz, o que equivale a um período de onda de 16,66 ms. O microcontrolador será o responsável por calcular o ângulo de disparo e ativar o *gate* do TRIAC, mas para isso é necessário sincronizar o circuito com a rede elétrica, em outras palavras, o microcontrolador precisa detectar o momento exato que a senoide da rede elétrica corta o eixo x do plano cartesiano (y=0). Para que isso seja possível, o primeiro passo a se fazer é retificar o sinal de onda, para depois, juntamente com o optoacoplador 4N35, fazer a detecção da passagem por zero.

Para retificar o sinal de onda foi montada uma ponte retificadora com 4 diodos, modelo 1N4007. Após o sinal passar pela ponte e ser retificado, ele terá uma frequência de 120Hz e um período de onda de 8,333 ms. Na figura 8 pode-se comparar o sinal de onda da rede elétrica com o sinal de onda retificado.

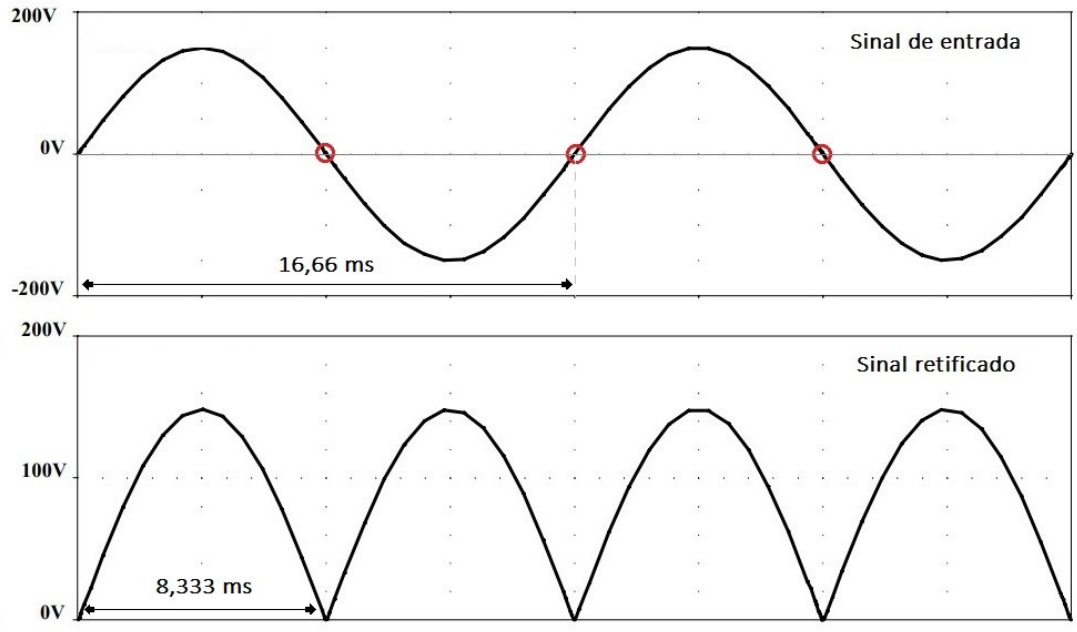

Figura 8 - Comparação entre sinal de onda de entrada e sinal retificado

A detecção de passagem por zero é feita aplicando o sinal de onda retificado no ânodo do optoacoplador 4N35, e alimentando o mesmo com uma tensão de 3,3V CC, vindo da fonte. O emissor do optoacoplador terá como sinal de saída um pulso em nível logico baixo, cada vez que o sinal da onda retificada passar por zero. Por fim conecta-se o emissor do 4N35 à entrada digital D1 do microcontrolador NodeMCU.

Como o circuito projetado opera em duas tensões distintas, foi necessário a utilização de fotoacopladores para fazer o isolamento elétrico. Para este projeto foram utilizados o fotoacoplador MOC3020, que será citado mais a frente, e o optoacoplador 4N35. Ambos são responsáveis por fazer o isolamento elétrico do circuito, separando as partes que trabalham em alta tensão da parte do controle, que opera em 3,3V CC.

Com o circuito de detecção por zero funcionando já é possível calcular o ângulo de disparo do TRIAC. Como temos um sinal de onda retificado, a cada 8,333 milissegundos o circuito de detecção de zero vai gerar uma interrupção no microcontrolador. A fins didáticos chamaremos o tempo de 8,333 milissegundos de $t3$. A partir da interrupção gerada no microcontrolador, será feito o cálculo do tempo $t1$, que é o tempo de *delay* necessário para controlar a potência da lâmpada.

Como este projeto possui dos modos de operação, manual e automático, existem duas maneiras de calcular o tempo $t1$. O modo manual é o mais simples, pelo smartphone, o usuário ajusta a tensão da lâmpada conforme sua vontade, em uma escala de zero a 100. O microcontrolador recebe este valor, denominado dim, e calcula o tempo $t1$ pela fórmula (3):

$$t1 = 83 * (100 - dim) \quad (3)$$

A variação do tempo de *delay* $t1$ (em microssegundos) é o responsável por controlar a tensão da lâmpada. Quanto maior o tempo $t1$, mais tarde ocorrerá o disparo do TRIAC, resultando em um brilho menor na lâmpada. Por exemplo, se o tempo de *delay* $t1$ for zero significa que o disparo do TRIAC não terá atraso, e consequentemente a lâmpada ficará com brilho máximo. Por outro lado, se o tempo $t1$ for de 4,165 ms, o disparo do TRIAC ocorrerá na metade do sinal senoidal, fazendo com que a tensão na lâmpada seja de 50%.

Em resumo, a cada $t1$ us o microcontrolador enviará um pulso para disparar o gate do TRIAC. Como forma de proteção elétrica ao sistema o pulso de saída gerado pelo microcontrolador será enviado, pelo pino digital de saída D2, para o ânodo do MOC3020, que

funcionará como um isolante elétrico, como foi explicado anteriormente. O MOC3020 ativará então o *gate* do TRIAC. As conexões do MOC com o TRIAC podem ser vistas na figura 9.

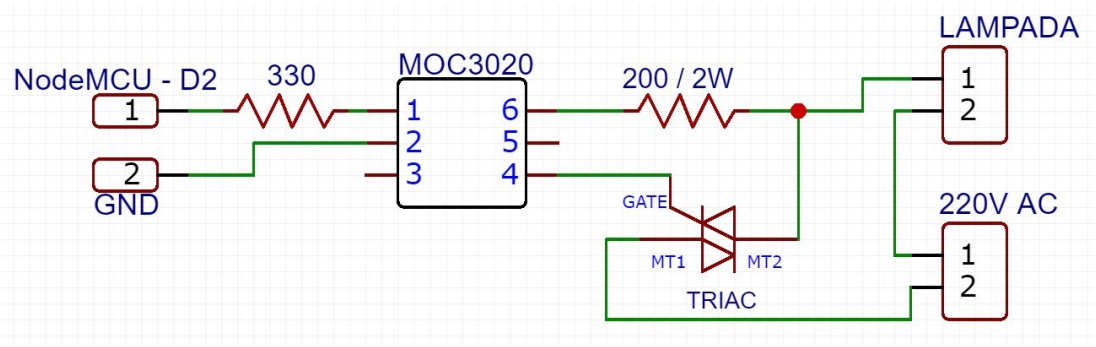

Figura 9 - Conexão entre MOC3020 e TRIAC

Diferentemente do modo manual, no modo automático quem define a tensão da lâmpada é o microcontrolador. Neste modo o usuário apenas definirá no smartphone o valor desejado referente aos luxes no ambiente. Para que seja possível calcular o tempo $t1$, o circuito conta com um sensor de luminosidade TEMT6000 conectado na entrada analógica A0 do microcontrolador. A entrada A0 lê valores de 0 a 1023 do sensor de luminosidade, que não são proporcionais ao número de lux no ambiente. Precisa-se então calibrar o sensor para que ele meça os luxes de forma correta.

Para calibrar o sensor foi necessário o auxílio de um luxímetro. Foram tiradas várias medidas do luxímetro e do sensor TEMT6000 em diferentes níveis de luminosidade e com os pontos gerados obteve-se uma reta, como pode ser visto na figura 10. Com auxílio do software online WolframAlpha[2] obteve-se o gráfico e a equação da reta. Com a equação da reta foi possível calibrar o sensor via software no Arduino IDE obtendo valores bastante satisfatórios, comparados aos valores lidos pelo luxímetro.

[2] https://www.wolframalpha.com/

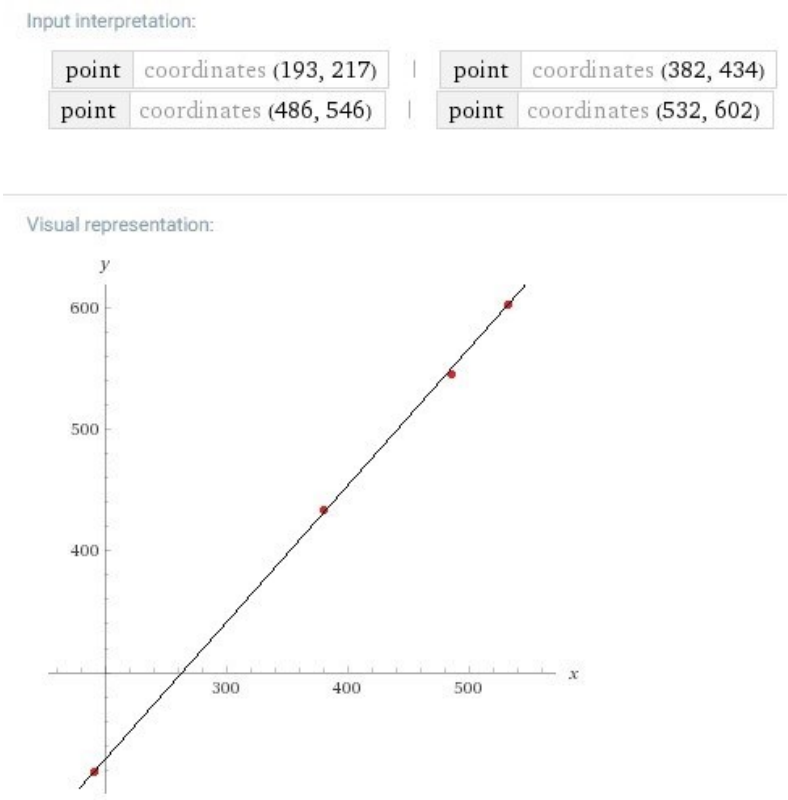

Figura 10 - Gráfico da reta com pontos no formato (sensor , luxímetro)

O funcionamento do modo automático funciona da seguinte forma: O usuário define quantos lux deseja no ambiente, o microcontrolador recebe este valor, compara com a leitura do sensor de luminosidade, já calibrado em lux, e através de um controlador PI, calcula o valor de $t1$, que definirá a potência necessária a ser entregue na lâmpada para atingir os lux previamente definidos no aplicativo para o ambiente.

Para calcular o tempo $t1$ no modo automático também foi necessário implementar um controlador PI. Inicialmente foi realizado a medição do máximo de lux que a lâmpada incandescente de 120W usada no projeto consegue entregar. A uma distância de 60 cm o luxímetro marcou 200 luxes.

Através do software Matlab, por meio da ferramenta *PID Tuner*, foi possível projetar o controlador PI de forma simples. O tempo de resposta foi ajustado manualmente na ferramenta para 2 segundos, como pode ser visto na figura 11. Após alguns testes experimentais, foi possível perceber que ao definir um tempo de resposta muito baixo a lâmpada fica num estado de pisca-pisca, não conseguindo ajustar sua tensão de forma adequada. A função de transferência do controlador PI resultante do *PID Tuner* está representada na equação a seguir:

$$C = \frac{K_i * 1}{s} \tag{4}$$

Sendo $K_i = 0.005$ e o ganho estático $s = 200$.

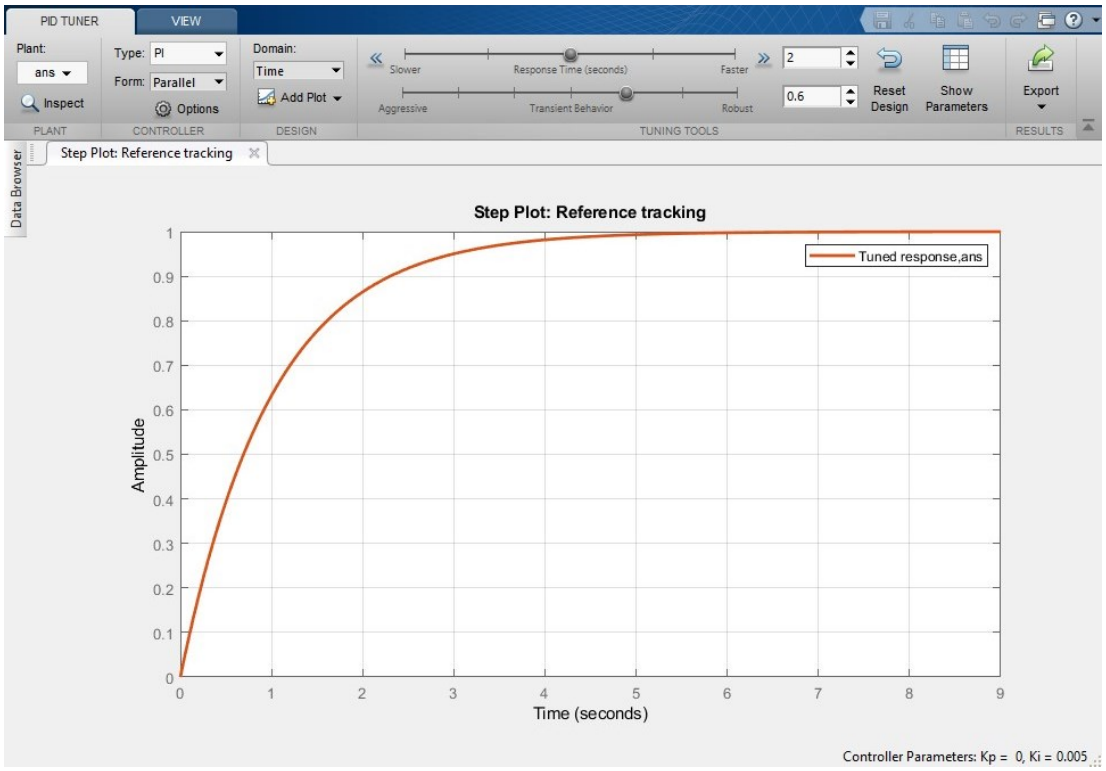

Figura 11 - Gráfico do tempo resposta. Criado pelo PID Tuner, no Matlab

Quando se trabalha com microcontroladores é preciso que o sistema seja em tempo discreto, pois para a implementação das leis de controle os sinais não devem ser contínuos e sim amostrados. O Matlab possui uma ferramenta para fazer a discretização do sistema. Através do comando $c2d(sys, Ts)$, em que sys é a função de transferência a ser discretizada e Ts é o tempo de amostragem, a equação (4) da função de transferência foi discretizada, resultando na equação abaixo:

$$c = \frac{K_i * T_s}{z - 1} \tag{5}$$

Sendo $K_i = 0.005$ e $T_s = 0{,}00833$.

Usando o comando $tf(c)$ convertemos a equação (5) em uma equação de transferência. O resultado pode ser visto na equação (6) abaixo. O valor de 0.00004167 será chamado de k_c e será usado para futuramente calcular o valor de D.

$$tf(c) = \frac{0.00004167}{z - 1} \tag{6}$$

O diagrama de blocos do controle de sistema pode ser visualizado na figura 12 abaixo:

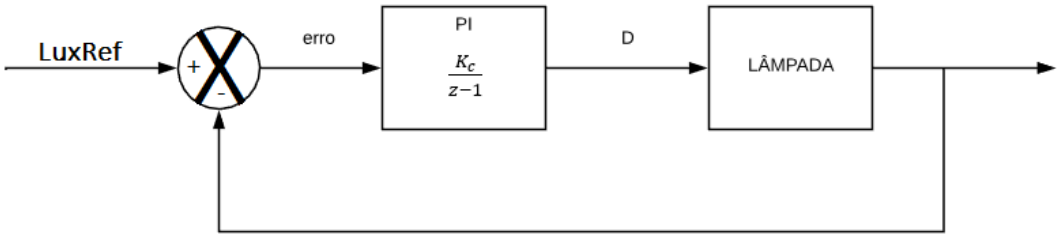

Figura 12 - Diagrama do controle de sistema

O erro será o valor de lux recebido pelo aplicativo, chamado LuxRef, menos o valor de lux lido pelo sensor TEMT6000. O valor do D é definido pela Equação (7) abaixo:

$$D = D + erro * K_c \tag{7}$$

Após calcular o valor de D podemos finalmente calcular o tempo $t1$. O cálculo se dá pela equação (8) abaixo:

$$t1 = 8333 * (1 - D) \tag{8}$$

Note que temos dois tempos $t1$ distintos, o tempo $t1$ do modo manual, calculado pela equação (3) e o tempo $t1$ do modo automático, calculado pela equação (8). Ambos foram implementados e são utilizados para controlar o disparo do TRIAC. Para que não haja conflito foi implementado no código uma função de controle, que impossibilita o modo automático e o modo manual trabalharem ao mesmo tempo. Todas as equações citadas acima foram

implementadas no software feito no Arduino IDE e o código final foi salvo na memória interna do microcontrolador NodeMCU.

3.2 Software

O software desenvolvido para este projeto foi desenvolvido e compilado na plataforma Arduino IDE, para a placa microcontroladora NodeMCU. O software do Arduino IDE não tem suporte para as placas baseadas no ESP8266 nativamente, então temos que instalar a biblioteca para a placa manualmente.

Para fazer a instalação da biblioteca deve-se adicionar a URL "http://arduino.esp8266.com/stable/package_esp8266com_index.json" abrindo o menu preferências, como mostrado na figura 13 abaixo:

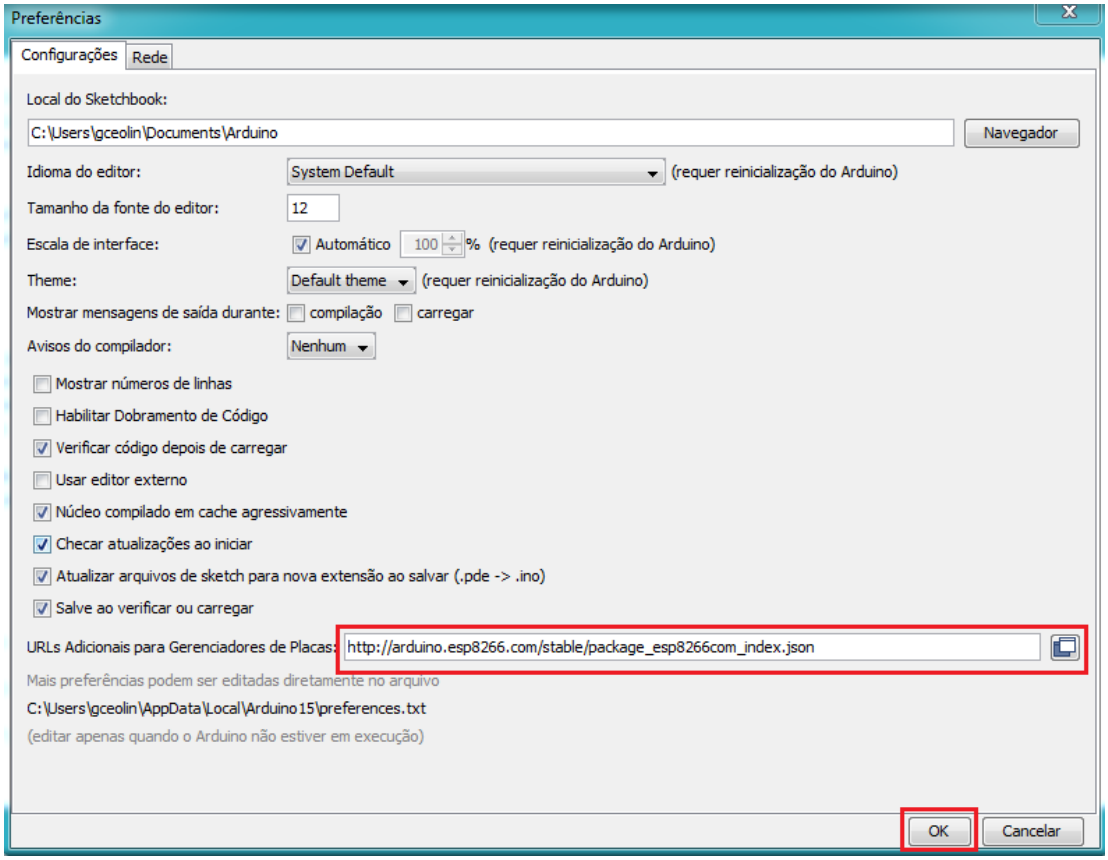

Figura 13 - Menu preferências no software Arduino IDE

Após isso deve-se instalar a placa, entrando no menu gerenciador de placas e instalando a placa esp8266, como mostrado na figura 14.

Figura 14 - Janela Gerenciador de placas do software Arduino IDE

Após a configuração da placa foi possível começar a programação do código. O código completo desenvolvido para este projeto está presente no ANEXO A.

O código conta com várias funções, que exercem diversas funcionalidades, tais como: Conexão com o servidor MQTT, controle de modo, cálculo do ângulo de disparo do TRIAC, salvar estado atual na memória interna, enviar e receber dados no servidor MQTT, entre outras.

Para facilitar o entendimento do código gravado no microcontrolador pode-se observar o fluxograma na figura 15 abaixo:

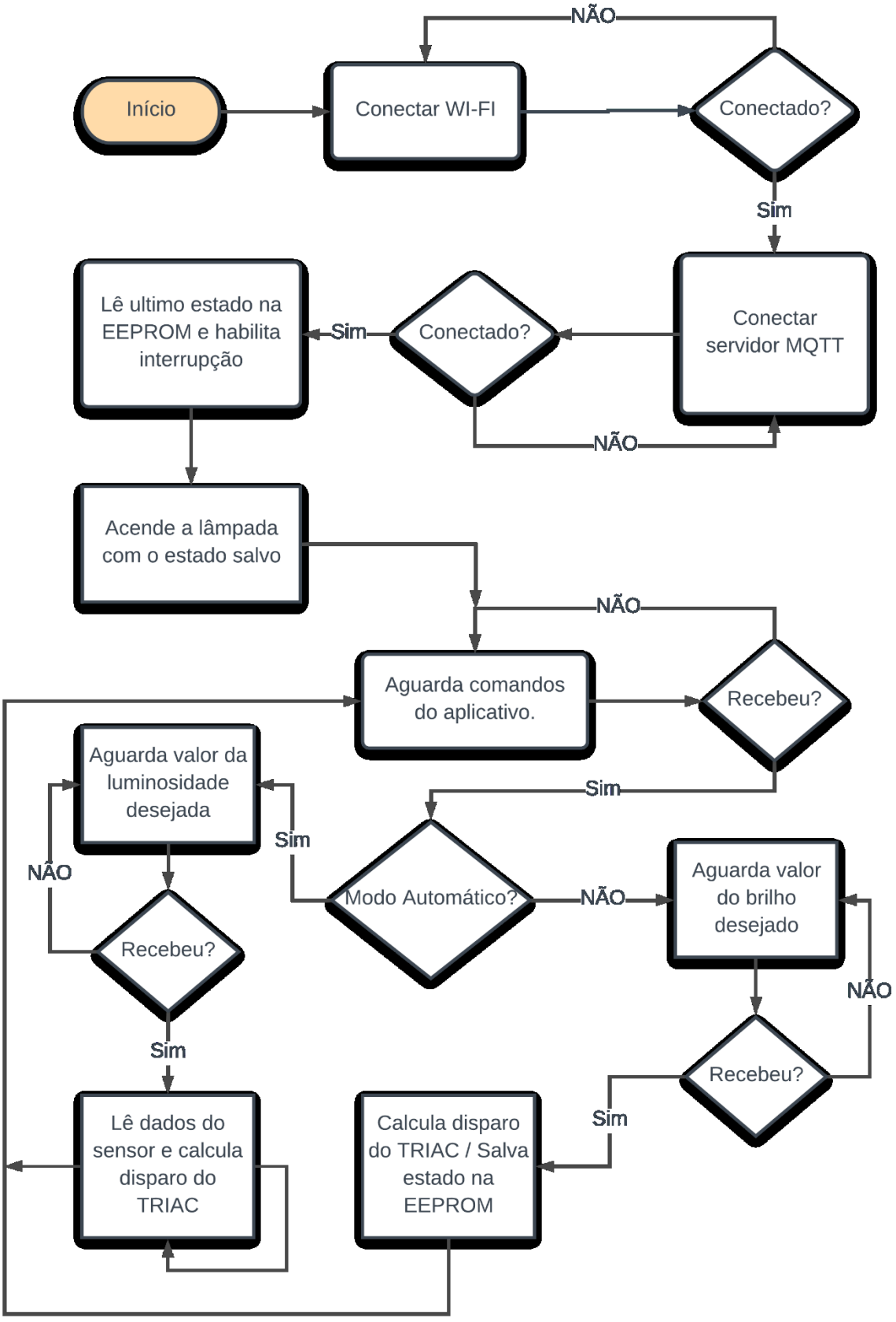

Figura 15 - Fluxograma do código gravado no microcontrolador

3.3 Servidor MQTT

Para que todo o projeto funcione perfeitamente é preciso um meio de comunicação que conecte o microcontrolador ao aplicativo desenvolvido. Para esta conexão foi escolhido usar um servidor MQTT. O servidor MQTT funciona como um intermediário entre microcontrolador e smartphone, como se fosse um servidor na nuvem. O servidor MQTT tem algumas vantagens sobre protocolos de comunicação diretos como por exemplo HTTP (*Hyper Text Transfer Protocol*). Além de ter uma conexão mais estável, uma das principais vantagens do MQTT é o fato de não ser necessário que todos os dispositivos estejam conectados na mesma rede WI-FI. Basta estar conectado à internet, independentemente do tipo de conexão e do lugar em que está conectado. Isto torna possível, por exemplo, controlar dispositivos diversos que estão em sua casa de qualquer lugar do mundo.

Existem diversos servidores MQTT que podem ser usados de forma gratuita, bastando apenas realizar um cadastro para utilizá-lo. O servidor escolhido para este projeto foi o CloudMQTT[3]. Após realizar um simples cadastro no site já é possível começar a configuração do servidor para utilizá-lo no projeto.

A configuração do servidor é bastante simples. Após o cadastro, basta criar uma nova instância, nomeá-la e escolher um plano, gratuito ou pago. O plano escolhido foi o gratuito, chamado de *Cute Cat*. Após a instancia ser criada, será redirecionado a uma página com os detalhes sobre o servidor referentes a conexão, que devem ser utilizadas em todos os dispositivos que queiram se comunicar com este servidor MQTT, como pode ser visto na figura 16. Estes dados serão utilizados para configurar a conexão com o servidor, tanto do microcontrolador quanto do aplicativo no smartphone.

[3] www.cloudmqtt.com

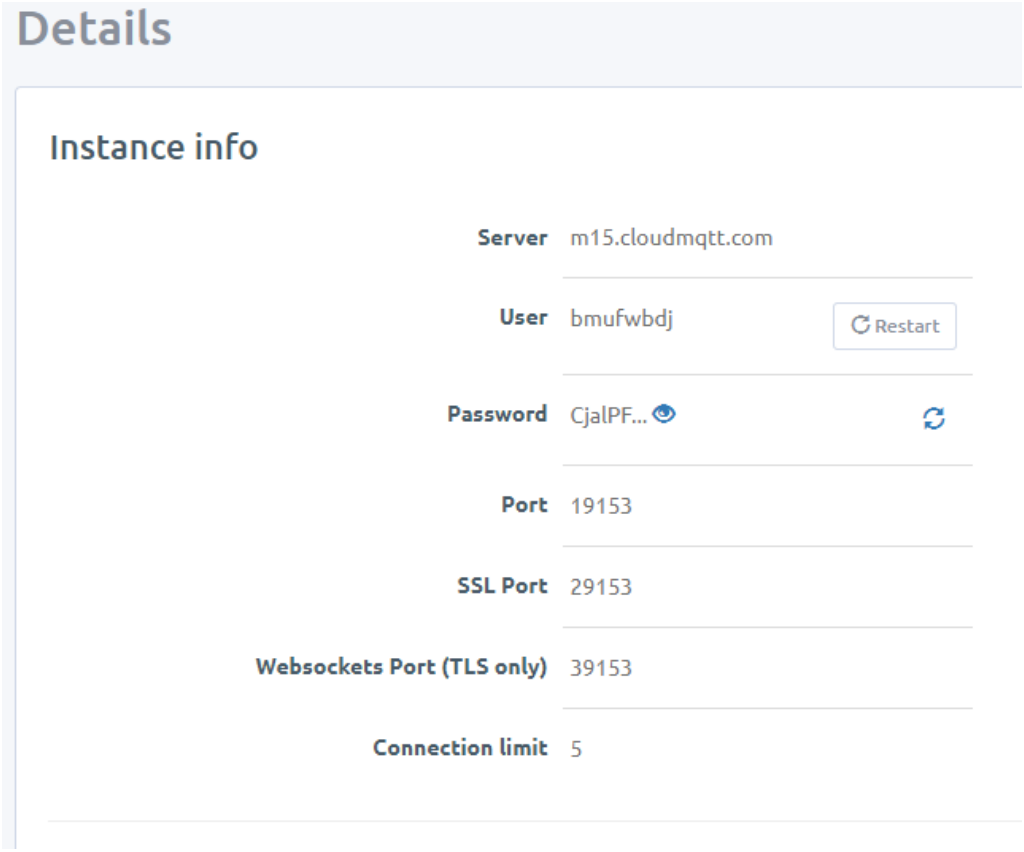

Figura 16 - Dados do servidor CloudMQTT

A troca de mensagens no protocolo MQTT é feita através de tópicos, onde os dispositivos podem ser *Publisher*, quando envia informação ao servidor, ou *Subscriber*, quando recebe informação do servidor. Usando como exemplo este mesmo projeto, o microcontrolador trabalha como *Publisher*, enviando as leituras de luminosidade do sensor para o servidor MQTT usando o tópico "sensor/LUX". Ao mesmo tempo o smartphone, pelo aplicativo, trabalha como *Subscriber*, lendo os valores do tópico "sensor/LUX" do servidor MQTT. Toda vez que o *Publisher*, neste caso o microcontrolador, enviar um novo dado ao tópico "sensor/LUX" todos os dispositivos que estiverem assinando o tópico, no caso, os *Subscribers*, receberão o novo dado.

3.4 Aplicativo Android

O aplicativo criado para se conectar com o servidor MQTT e, consequentemente controlar a placa NodeMCU, não exigiu nenhuma linha de código programada. Aproveitando os benefícios de usar a comunicação MQTT, não foi difícil encontrar alguns aplicativos

Android pré configurados. O aplicativo usado se chama “IoT MQTT Panel”, é gratuito e está disponível para download na loja virtual da Google, a Play Store.

Neste aplicativo é possível de forma fácil e rápida configurar a conexão com o servidor MQTT e montar um layout com os recursos necessários para a demanda do projeto.

Com o aplicativo instalado e aberto basta clicar no botão “+” para configurar uma conexão a um servidor MQTT. Nesta etapa serão usados os dados do servidor CloudMQTT criado anteriormente, encontrados na figura 16. Após o servidor estar configurado deve-se criar um novo “device”, clicando novamente no botão “+”. Nesta etapa basta colocar um nome e clicar em criar. Com o “device” criado basta escolher e adicionar um ou mais painéis no aplicativo, conforme necessário no projeto. Cada painel que troque informações com o servidor deve ser configurado individualmente como *Publisher* ou *Subscriber*, informando seus respectivos tópicos. O processo de criação do aplicativo pode ser visto na figura 17 abaixo:

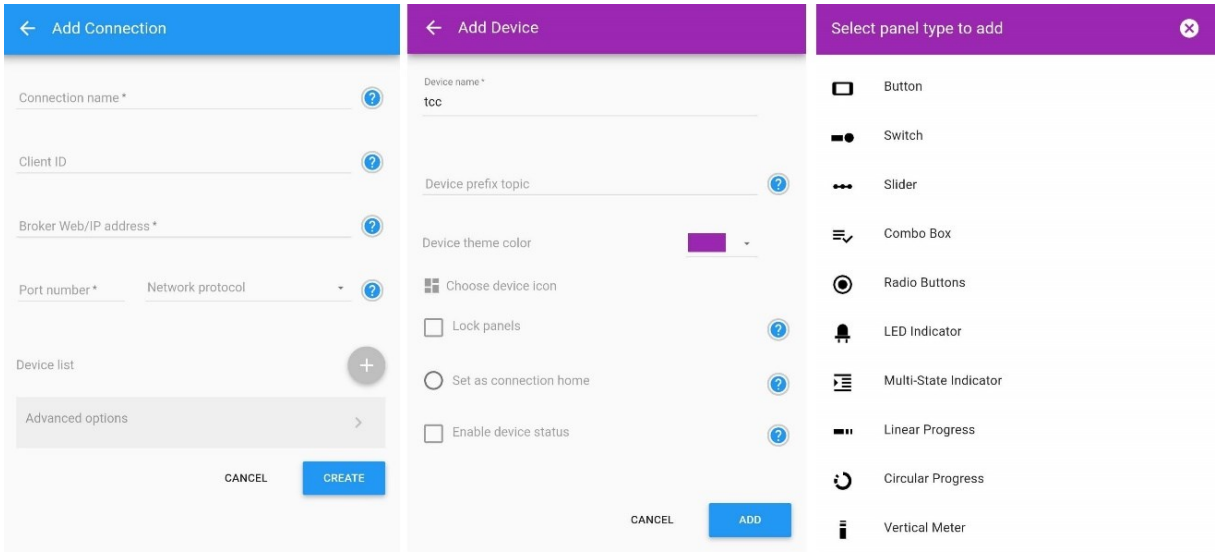

Figura 17 - Configuração e criação do aplicativo Android

O layout final do aplicativo elaborado pode ser visto na figura 18. Através deste aplicativo é possível ligar, desligar, controlar a tensão manualmente, ou, ativar o modo automático e definir a quantidade de lux desejada. O aplicativo ainda contém um campo que informa os luxes no ambiente em tempo real e um campo de monitoramento em forma de gráfico, que mostra a tensão da lâmpada, numa escala de zero a 100%, ao decorrer do tempo.

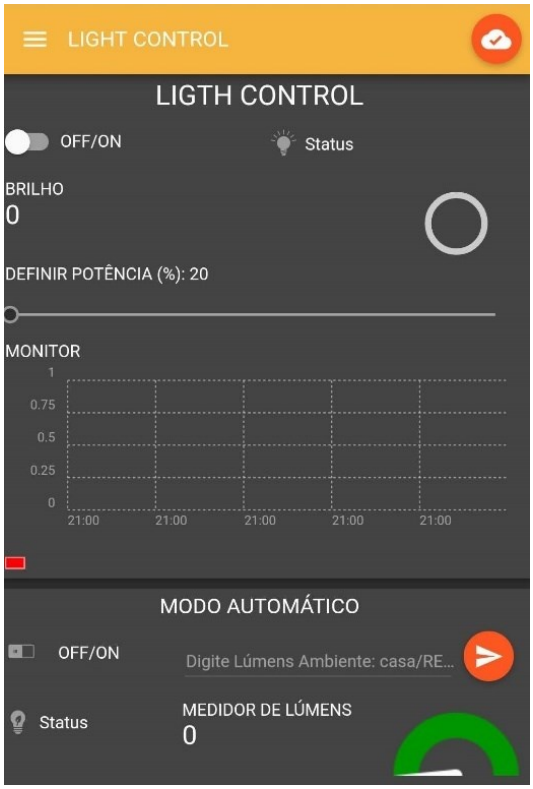

Figura 18 - Layout final do aplicativo desenvolvido

4 RESULTADOS

Com o circuito montado em uma protoboard, software e aplicativo desenvolvidos foi a hora de fazer os testes. Para evitar danos na protoboard relacionados a corrente alta, a parte do circuito que engloba a lâmpada, o TRIAC e a alimentação de entrada junto a rede elétrica foram soldados manualmente em uma placa fenolite perfurada. O circuito completo e a placa soldada podem ser vistos na figura 19 abaixo:

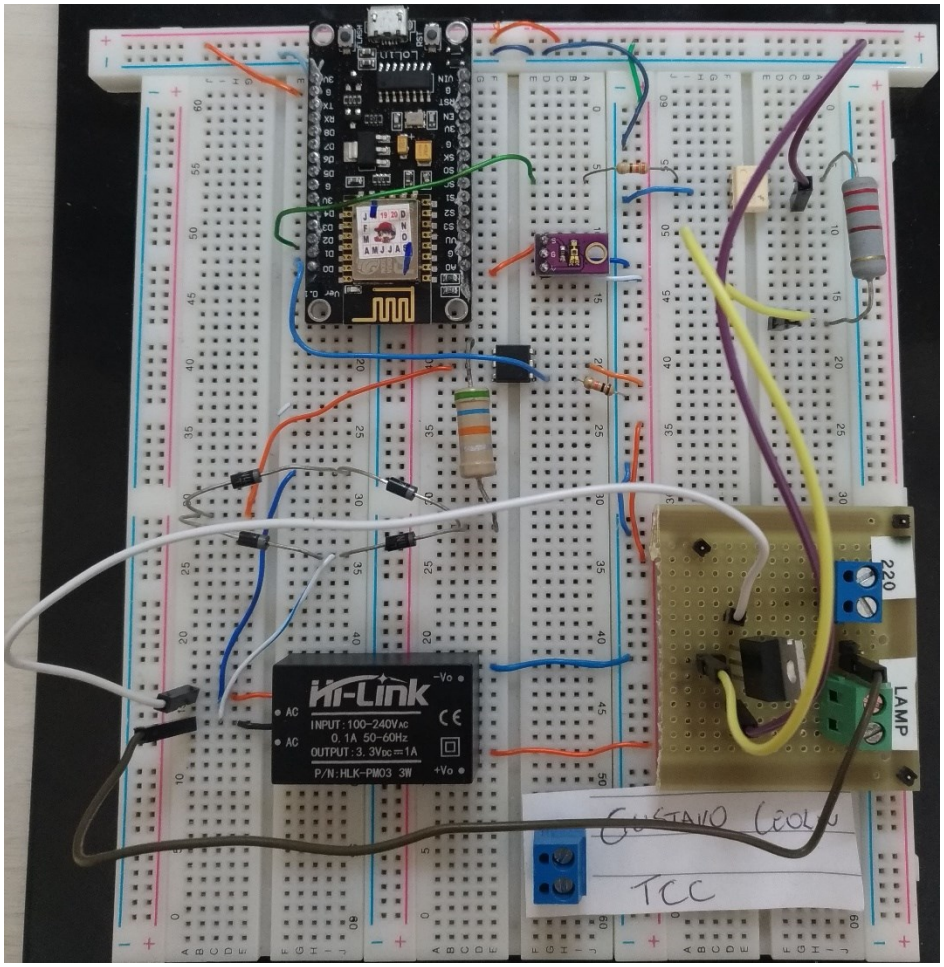

Figura 19 - Circuito montado na protoboard

O primeiro teste feito foi com o circuito que detecta a passagem por zero. O teste foi realizado com auxílio de um osciloscópio. Como era esperado, a cada semi-ciclo do sinal retificado, ao detectar a passagem no ponto do eixo cartesiano (y=0), é gerado um pulso em nível lógico baixo. O resultado do teste pode ser visto na figura 20 abaixo:

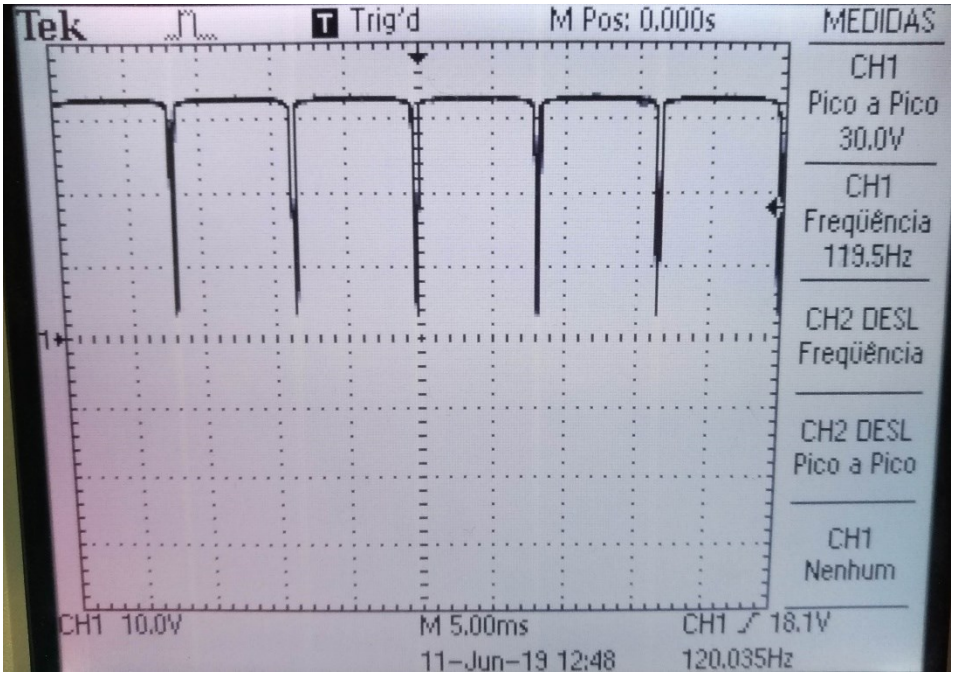

Figura 20 - Sinal da detecção de zeros

Com o detector de zero funcionando o próximo passo foi testar o disparo do TRIAC, comparando o sinal de onda de entrada, vindo da rede elétrica, com o sinal de onda de saída na lâmpada, controlado pelo disparo do TRIAC. Para fazer esse teste foi necessário usar duas ponteiras diferenciais conectadas no osciloscópio. Como pode ser observado na figura 21, a ponteira diferencial conectada no canal 1 do osciloscópio está medindo o sinal de entrada vindo da rede elétrica, enquanto a ponteira diferencial conectada no canal 2 está medindo a saída na lâmpada.

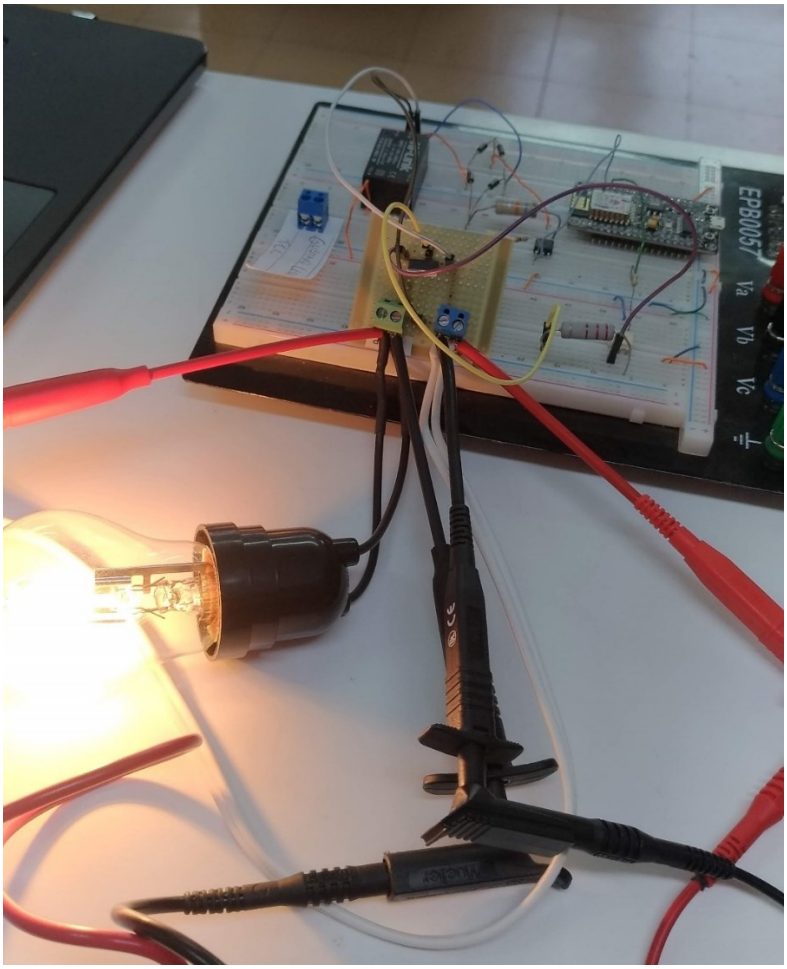

Figura 21 - Disposição das ponteiras diferenciais conectadas ao circuito

Diferentes níveis de tensão foram testados, a fim de comparar o sinal da onda de entrada e da onda saída. Na figura 22 a lâmpada estava com tensão máximo, resultando em sinais de onda de entrada e saída idênticos, como era esperado.

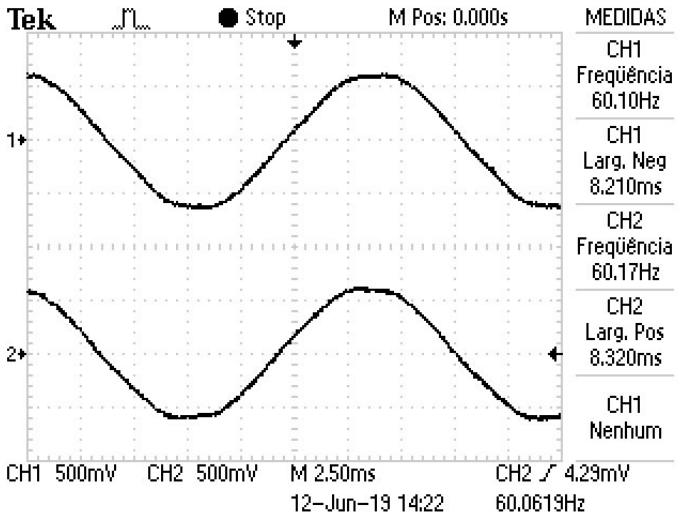

Figura 22 - Sinal da onda com potência máxima

No segundo teste a leitura do sinal foi realizada com a lâmpada desligada. Como é possível perceber na figura 23, o sinal de entrada da rede elétrica permanece o mesmo enquanto o sinal de saída na lâmpada é nulo. Isto ocorre, pois, ao definir a tensão como zero, não haverá o disparo do TRIAC.

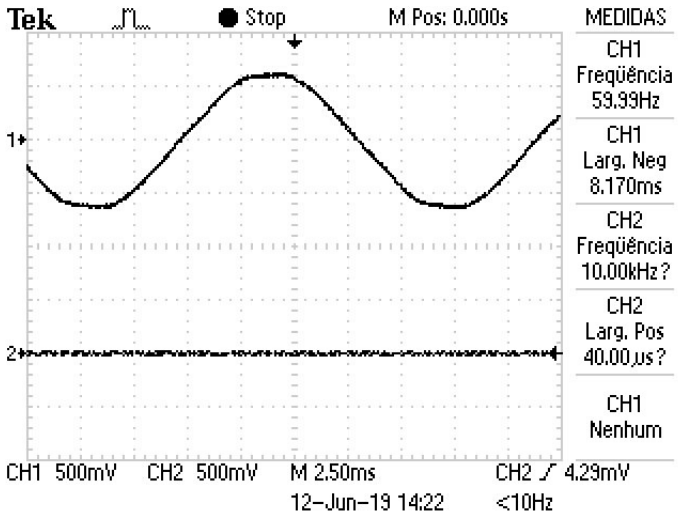

Figura 23 - Sinal da onda com lâmpada desligada

O terceiro teste foi realizado com a tensão da lâmpada definido em 50%. Nota-se pela figura 24 que o disparo do TRIAC é feito exatamente na metade do período de 8,333 ms, cortando o sinal senoidal de entrada pelo meio, o que resulta, em termos práticos, metade da tensão da lâmpada.

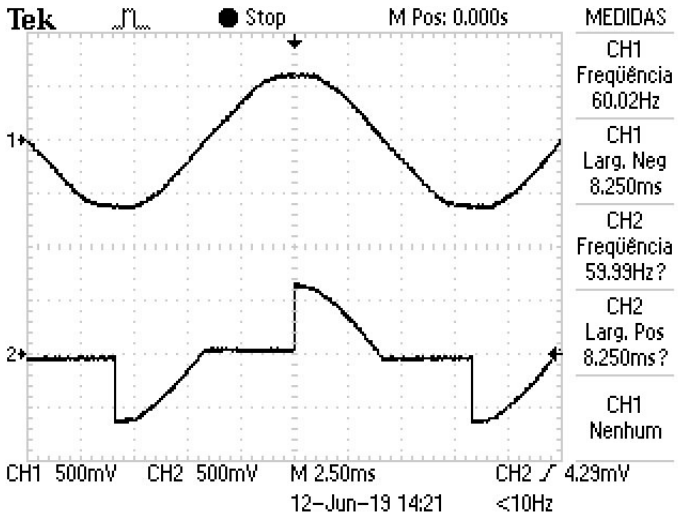

Figura 24 - Sinal da onda com potência de 50%

Os valores de tensão utilizados nos testes anteriores foram enviados pelo aplicativo Android desenvolvido. A tensão zero foi definido posicionando o botão OFF/ON na posição OFF, enquanto a tensão 50% e tensão 100% foram definidos pela barra deslizante, acima do campo de monitoramento da tensão, chamado “MONITOR”.

O Aplicativo também contém o modo automático, que ao ser ativado, desabilita o modo manual. No modo automático foi digitado o número de lux desejado e foi possível perceber a lâmpada variando a tensão automaticamente de forma gradual para manter o número de lux definido. Para isso foi usada a lanterna do próprio smartphone, focando a mesma em cima do sensor de luminosidade, simulando a luz solar. O campo chamado “MEDIDOR DE LUX” pôde ser usado para verificar se o funcionamento estava ocorrendo de forma adequada, como pode ser visto na figura 25.

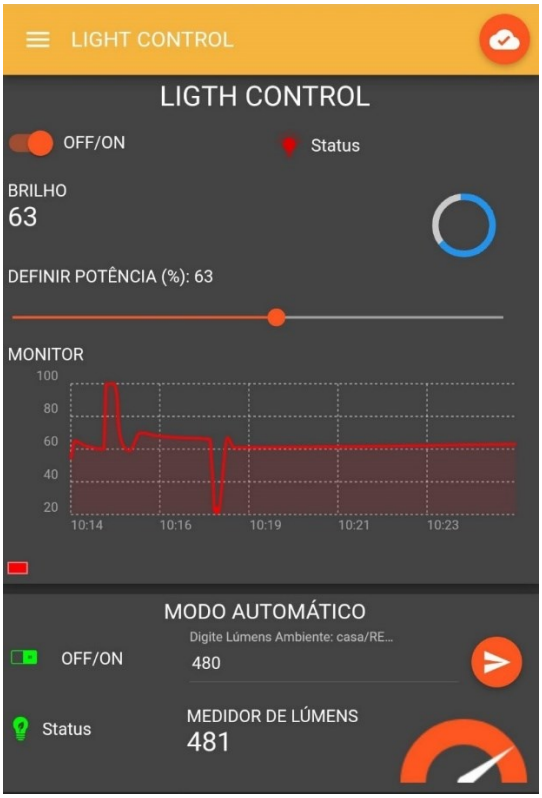

Figura 25 - Aplicativo com o modo automático ativado

Além dos testes com o circuito, foram realizados testes com a conexão do microcontrolador e do smartphone com o servidor MQTT, tanto por rede WI-FI, quanto por rede móvel (3G). Na maioria dos casos a conexão se mostrou bastante estável, e quando, em um caso, o microcontrolador perdeu a conexão, rapidamente ele se reconectou, graças a uma função presente no código que faz com que o microcontrolador se reconecte automaticamente.

O microcontrolador foi programado para que, toda vez que receba um valor diferente de tensão do aplicativo, salve o estado atual na memória EEPROM. Isso permite que, ao perder conexão ou em caso de queda de energia, ele retome o funcionamento com o último valor recebido.

A demonstração prática deste projeto encontra-se no Youtube e pode ser acessada através do link: https://youtu.be/el0kW4asWvE ou através do QR Code abaixo:

Scan me

5 CONCLUSÃO

Neste projeto foi proposto um método para o controle manual e automático do brilho de uma lâmpada, por meio de um aplicativo Android. A lei de controle proposta para o modo automático consiste em controlador PI que tem o objetivo de compensar o ganho, resultando no controle de potência da lâmpada.

O desempenho do projeto como um todo se mostrou bastante eficaz. Os resultados obtidos foram satisfatórios e cumpriram com os objetivos de controlar a iluminação de forma eficiente, rápida e prática. Vale mencionar que este projeto pode ser utilizado com qualquer lâmpada do tipo dimerizável, que trabalhe em tensão de 220V, não se restringindo apenas a lâmpadas incandescentes. Inicialmente a ideia era utilizar uma lâmpada LED bulbo dimerizável, porém seu alto custo e dificuldade de encontrar em lojas físicas acabaram por inviabilizar a aquisição. A escolha de uma lâmpada incandescente para o projeto se deve ao seu baixo custo e facilidade de aquisição.

O projeto construído também se mostrou relativamente barato, comparado ao que existe no mercado hoje em dia. Foram investidos cerca de 70 reais, sendo o microcontrolador a peça mais cara do projeto, custando pouco mais da metade do orçamento total. Fazendo uma busca no mercado não foi encontrado nenhum outro produto que consiga entregar as mesmas funcionalidades que este projeto na mesma faixa de preço. O dispositivo mais semelhante encontrado na pesquisa foi a lâmpada Yeelight, da marca Xiaomi, que custa em torno de 140 reais, porém esta lâmpada não conta com o recurso que controla seu brilho automaticamente, como o presente neste projeto.

A principal contribuição deste projeto está na implementação da lei de controle, que possibilita uma iluminação constante e regular. A consequência disso não está apenas no conforto e comodidade, mas também na economia de energia elétrica, pois neste caso, diferentemente do sistema de iluminação comum, a lâmpada só estará usando o brilho máximo quando realmente for necessário.

BIBLIOGRAFIA

ARAÚJO, F. M. U. **Sistemas de Controle: apostila**. Natal: Universidade Federal do Rio Grande do Norte, 2007. 96 p.

ASHTON, K. **Internet of Things**. RFID Journal. Disponível em: < https://www.rfidjournal.com/articles/view?4986 >. Acessado em: 2 maio. 2019.

BOYLESTAD, R. L.; NASHELSKY, L. **Dispositivos Eletrônicos e teoria de circuitos.** 11 ed. São Paulo: Pearson Education do Brasil Ltda, 2013. 731 p.

ESPRESSIF. **ESP8266EX Datashhet Versão 5.8**. 2018. Acesso em 20 abril.2019. Disponível em: < https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf >.

FREITAS, C. M. **Controle PID em sistemas embarcados**. Embarcados. Acessado em: 20 junho.2019. Disponível em: <https://www.embarcados.com.br/controle-pid-em-sistemas-embarcados/>.

KARAGIANNIS, V. et al. **A Survey on Application Layer Protocols for the Internet of Things**. 2015. 3 p.

MALVINO, A.; BATES, D. **Eletrônica**. 8 ed. São Paulo: AMGH Editora Ltda, 2016. 97 p.

NISE, NORMAN S. **Engenharia de Sistemas de Controle**. 6 ed. Rio de Janeiro: LTC, 2012.

OGATA, K. **Engenharia de controle moderno**. 5 ed. São Paulo: Prentice Hall, 2010. 21 p.

SCHULER, C. **Eletrônica I.** 7 ed. Porto Alegre: AMGH Editora Ltda, 2013. G18 p.

SOUZA, David José. **Desbravando o PIC: ampliado e atualizado para PIC16F628A**. 8 ed. Editora Érica Ltda 2005. 21 p.

YUAN, M. **Conhecendo o MQTT**. IBM. Disponível em: < https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html/>. Acessado em: 15 junho. 2019

ANEXO A - CÓDIGO IMPLEMENTADO NO MICROCONTROLADOR NODEMCU

```
#include <EEPROM.h>             //biblioteca para possibilitar uso da eeprom
#include <ESP8266WiFi.h>        //biblioteca para gerenciar o protocolo WiFi
#include <PubSubClient.h>

#define zc_pin D1     //define D1 como pino de detcção de passagem por zero
#define moc_pin D2    //define D2 como pino de controle do gate do triac
#define DEBUG
#define INTERVALO_ENVIO       1000
#define kc  0.00004167

const int analogInPin = A0; // define pino A0 a variavel
int sensorValue = 0;    // leitura bruta do sensor de luz
int media = 0; //variavel auxiliar para fazer media de leitura
int mediaF = 0; //variavel da media de 200 valores de SensorValue
volatile int tl = 0; // tempo tl
int dim = 0; //valor do dimmer recebido pelo app
int LUXSENSOR = 0; //valor da luminosidade após o calculo
int ultimoEnvioMQTT = 0; //tempo decorrido apos ultimo envio ao servidor
int AUTOM=0; //variavel para modo automatico
int memo=2; //variavel para evitar gravaçoes desnecessarias na EEPROM
int luxRef=120; //variavel para o numero de lumens recebidos pelo app
float d=0;  //variavel para o controle
float atraso=0; //variavel para o controle de modo automatico

//------------informações da rede WIFI-------------------
const char* ssid = "ASUS";                 //SSID da rede WIFI
const char* password =  "0123456789";    //senha da rede wifi
//const char* ssid = "NET_2GB8D9BE";                 //SSID da rede WIFI
//const char* password =  "COLORADO10";    //senha da rede wifi

//informações do broker MQTT - Verifique as informações geradas pelo cloudmqtt.com
const char* mqttServer = "m15.cloudmqtt.com";   //server
const char* mqttUser = "bmufwbdj";              //user
const char* mqttPassword = "CjalPFrkNgKS";      //password
const int mqttPort = 19153;                     //port
const char* mqttTopicSub = "casa/#";//tópico que sera assinado para valor do dimmer

WiFiClient espClient;
PubSubClient client(espClient);
```

```
//-----------------PRINCIPAL-------------------
void setup() {

  Serial.begin(115200);

  detachInterrupt(digitalPinToInterrupt(zc_pin));     //desabilita interrupção
  pinMode(moc_pin, OUTPUT);                 //define pino de controle do triac como saída

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {//enquanto nao conectado na wifi mostrar mensagem
  delay(500);

#ifdef DEBUG
    Serial.println("Conectando ao WiFi..");
#endif
  }
#ifdef DEBUG
  Serial.println("Conectado na rede WiFi"); //mensagem após conectar wifi
#endif

  client.setServer(mqttServer, mqttPort); //configura servidor mqtt
  client.setCallback(callback);

  while (!client.connected()) {//funcao para conectar no servidor mqtt
#ifdef DEBUG
    Serial.println("Conectando ao Broker MQTT...");
#endif

    if (client.connect("ESP8266Client", mqttUser, mqttPassword )) {
#ifdef DEBUG
      Serial.println("Conectado");
#endif

    } else {
#ifdef DEBUG
      Serial.print("falha estado  ");
      Serial.println(client.state());
#endif
      delay(2000);

    }
  }
```

```
   EEPROM.begin(16);//Inicia a EEPROM com tamanho de 16 Bytes
   Serial.print("Ultimo valor dim salvo na EEPRON: ");
   Serial.println(EEPROM.read(1));//Mostra no Monitor oque há antes de efetuar a gravação

   client.subscribe(mqttTopicSub);   //subscreve no tópico

  //habilita interrupções no pino ZC a cada interrupção na borda de descida,
  //chama a função controle
  attachInterrupt(digitalPinToInterrupt(zc_pin), controle, FALLING);

  if(memo==2){ //funcao para ler o ultimo valor da EEPROM  e salvar em dim.
               //executa toda vez que resetar, desconectar ou queda de energia
    dim = EEPROM.read(1);
  }
  else{
    Serial.print("ERRO EEPROM");
  }
}

//------------FUNCAO QUE SALVA ESTADO ATUAL NA EEPRON------------------- //
void salvaEEPROM (){
  //gravar ultimo valor recebido pelo app na memoria
  if (EEPROM.read(1) != dim)//Se não existir o numero escolhido (dim), irá gravar.
                            // Isto é necessário para evitar regravações desnecessárias.
   {
      EEPROM.write(1, dim);//Escreve no endereço "0", o valor "dim".
      EEPROM.commit();//Salva o dado na EEPROM.
      Serial.print("Novo valor dim Salvo na EEPRON: ");
      Serial.println(EEPROM.read(1));//Mostra o valor no endereço 0 novamente.
   }
   else //Se já existir o valor, irá avisar.
   {
      Serial.println("Dado dim ja cadastrado");
   }
}
```

```
//--------------------------FUNCAO DE CONTROLE AUTOMATICO-------------------------
void controle(){

  float erro=0;

  erro = luxRef - LUXSENSOR; //ERRO= lumens do app - lumens do sensor

  if (d>=1)
    d=1;

  if (d<=0)
    d = 0;

    d = d+ erro*(kc); // d(k+1) = erro*kc(k) + d(k)
    atraso=8333*(1-d);

   //FUNCAO PARA CONTROLAR MODO AUTOMATICO
   if(AUTOM==1){ //se modo automatico ativado
      t1=atraso;
      calculaDim();

    }
   else if(AUTOM==0){ //SE MODO AUTOMATICO DESATUVADO
      t1 = 83 * (100 - dim);     //calcula o tempo de delay para disparar o triac
      calculaDim();
    }
}
```

```
//------------CHAMADA DE VOLTA MQTT-------------------
//(função chamada quando qualquer informação de um dos tópicos subescritos chega)
void callback(char* topic, byte* payload, unsigned int length) {

  //armazena msg recebida em uma sring
  payload [length] = '\0';
  String strMSG = String((char*)payload); //armazena valor recebido em uma string
  String strTopic = String((char*)topic); //armazena nome do topico usado em uma string

  if(strcmp (topic,"casa/LAMP")==0){ //se topico = casa/LAMP
  if(strMSG=="LIGA"){ //se mensagem do topico = LIGA
    dim = EEPROM.read(1); //DIM recebe valor salvo na memoria eeprom
    salvaEEPROM();
  }
  else if(strMSG=="OFF"){ //se mensagem do topico = OFF
    dim=0;                // desliga a lampada
  }
  else if((dim <= 100)&&(dim >=20)){
    dim = strMSG.toInt(); //CONVERTE VALOR PARA INTEIRO
    salvaEEPROM();
  }
}
else if(strcmp (topic,"casa/RECLUX")==0){//se topico = casa/RECLUX
    Serial.println(strMSG);
    luxRef = strMSG.toInt(); //CONVERTE VALOR RECEBIDO PARA INTEIRO
}
else if(strcmp (topic,"casa/MODO")==0){ //se topico = casa/MODO
    AUTOM = strMSG.toInt(); //CONVERTE VALOR PARA INTEIRO
}
}
//------------FUNÇÃO QUE CALCULA O ANGULO DE DISPARO DO TRIAC-------------------
void calculaDim() {
//valor de tl é calculado na funçao controle()
  if (tl <= 300) {    //se tl for menor que 300 us a lâmpada fica com brilho máximo
    digitalWrite(moc_pin , HIGH);
  }
  else if (tl >= 8000) {    //se tl for maior que 8 ms a lampada fica desligada
    digitalWrite(moc_pin, LOW);
  }
  else if ((tl > 300) && (tl < 8000)) {    // controla o brilho da lampada
    delayMicroseconds(tl);                  // espera tl micorssegundos
    digitalWrite(moc_pin, HIGH);            // ativa o gate do triac por 10 microssegundos
    delayMicroseconds(10);
    digitalWrite(moc_pin, LOW);             // desativa o gate do triac
  }
  memo=0;
}
```

```
//------------FUNÇAO PARA RECONECTAR MQTT-------------------
void reconect() {
  //Enquanto estiver desconectado
  while (!client.connected()) {
#ifdef DEBUG
    Serial.print("Tentando conectar ao servidor MQTT");
#endif

    bool conectado = strlen(mqttUser) > 0 ?
                     client.connect("ESP8266Client", mqttUser, mqttPassword) :
                     client.connect("ESP8266Client");
    if (conectado) {
#ifdef DEBUG
      Serial.println("Conectado!");
#endif
      //subscreve no tópico
      client.subscribe(mqttTopicSub, 1);
    } else {
#ifdef DEBUG
      Serial.println("Falha durante a conexão.Code: ");
      Serial.println( String(client.state()).c_str());
      Serial.println("Tentando novamente em 10 s");
#endif
      //Aguarda 10 segundos
      delay(10000);
    }
  }
}
//----------------------FUNCAO QUE ENVIA DADOS PRO SERVIDOR-----------------------
void enviadados (){
  //armazena msg recebida em uma sring
  char MsgLuxMQTT[4];
  int luxM = LUXSENSOR;
  int dimM = dim;
  if(isnan(luxM))
  {
    #ifdef DEBUG
    Serial.println("Falha na leitura do sensor...");
    #endif
  }
  else
  {
    sprintf(MsgLuxMQTT,"%i",luxM);
    client.publish("envia/LUX",MsgLuxMQTT);
  }
}
```

```
//------------FUNÇAO LOOP------------------
void loop() {
  if (!client.connected()) {
    reconect();
  }

  //calcula lux a partir do sensor
    for(int i=0;i<200;i++){
        sensorValue = analogRead(analogInPin);
        delayMicroseconds(100);
        media=media+sensorValue;
    }
    mediaF=media/200;
    LUXSENSOR = (385*mediaF)/339;
    if ((millis() - ultimoEnvioMQTT) > INTERVALO_ENVIO)
    //CHAMA FUNCAO QUE ENVIA DADOS A CADA INTERVALO_ENVIO
  {
      enviadados();

      ultimoEnvioMQTT = millis();
  }
  media=0;
  mediaF=0;
  delay(500);
  client.loop();
}
```