UNIVERSIDADE DE UBERABA
RAFAEL SOUSA CHAGAS
YGOR FELÍCIO DE MATTOS ARAÚJO

SISTEMA DE ILUMINAÇÃO INTELIGENTE VISANDO O CONTROLE RESIDENCIAL DE FORMA AUTÔNOMA

UBERLÂNDIA – MG
2023

RAFAEL SOUSA CHAGAS
YGOR FELÍCIO DE MATTOS ARAÚJO

SISTEMA DE ILUMINAÇÃO INTELIGENTE VISANDO O CONTROLE RESIDENCIAL DE FORMA AUTÔNOMA

Trabalho apresentado à Universidade de Uberaba, como parte das exigências à conclusão do componente Trabalho de Conclusão de Curso, da 10ª etapa, do curso de Engenharia de Computação, da UNIUBE, Campus Uberlândia.

Orientador: Prof Msc. Júlio Almeida Borges

UBERLÂNDIA – MG
2023

SISTEMA DE ILUMINAÇÃO INTELIGENTE VISANDO O CONTROLE RESIDENCIAL DE FORMA AUTÔNOMA

Rafael Sousa Chagas; Ygor Felício de Mattos Araújo
rafael.chagas@edu.uniube.br; ygor.felicio@edu.uniube.br
Júlio Almeida Borges
julio.borges@uniube.br

RESUMO

Este trabalho tem como objetivo o desenvolvimento de um sistema de iluminação inteligente para automação residencial utilizando os conceitos de *Internet of Things (Internet das Coisas)*, com a intenção de funcionar de forma autônoma, com o mínimo de intervenção humana. Um sistema de iluminação comum possui comando apenas para ligar ou desligar o dispositivo, portanto, não há um aproveitamento da luz solar. Desta forma, o sistema dispõe de um sensor de luminosidade, para o controle preciso da iluminação do ambiente, como a função de ligar ou desligar automaticamente de acordo com a luz solar incidente. É utilizado um microcontrolador ESP8266, responsável por interligar as lâmpadas juntamente com os sensores e a programação foi feita utilizando a interface de desenvolvimento Arduino.

Palavras-chave: IoT, sistema, sensor, automação e dispositivo.

INTELLIGENT LIGHTING SYSTEM FOR AUTONOMOUS HOME CONTROL

ABSTRACT

This work aims to develop an intelligent lighting system for home automation using Internet of Things concepts, with the intention of operating autonomously with minimal human intervention. A common lighting system just has the control to turn the device on or off, so there is no utilization of sunlight. This way, the system has a luminosity sensor for precise control of ambient lighting, with the function of turning it on or off automatically according to the incident sunlight. A microcontroller ESP8266 is used, responsible for interconnecting the lamps together with the sensors, and the programming was done using the Arduino development interface.

Keywords: IoT, system, sensor, automation, device.

LISTA DE FIGURAS

LISTA DE SIGLAS

ADC	*Analog to Digital Converter*
AP	*Acess Point*
BPS	*Bits Per Second*
CSS	*Cascading Style Sheets*
GPIO	*General Purpose Input/Output*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
I2C	*Inter-Integrated Circuit*
IDE	*Integrated Development Environment*
IOT	*Internet of Things*
IP	*Internet Protocol*
KBYTES	*Kilobytes*
LDR	*Light Dependent Resistor*
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
ROM	*Read-Only Memory*
SPI	*Serial Peripheral Interface*
STA	*Station*
TCP	*Transmission Control Protocol*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*
V	*Volts*
VAC	*Voltage Alternating Current*
VCC	*Voltage Common Collector*
VDC	*Voltage Direct Current*
WEB	*World Wide Web*
Wi-Fi	*Wireless Fidelity*

SUMÁRIO

1 INTRODUÇÃO

Nessa seção serão apresentadas a contextualização, justificativa e objetivos do projeto desenvolvido.

1.1 Contextualização e justificativa

"A Internet das Coisas, em inglês, Internet of Things (*IoT*) é uma referência à habilidade de diferentes tipos de objetos conseguirem estabelecer conexão com a internet, desde eletrodomésticos até carros. Portanto, esses itens conseguem coletar e transmitir dados a partir da nuvem." (Carvalho, 2021)

"Atualmente, já é possível encontrar dispositivos *IoT* sendo utilizados tanto em situações comuns da vida diária como no âmbito profissional. Desse modo, essas ferramentas tecnológicas estão contribuindo para o acontecimento da transformação digital no mundo." (Carvalho, 2021)

Além destes conceitos, o termo "Internet das Coisas" (*IoT*) está sendo muito discutido e utilizado para realizar a automação de processos, ao permitir controlar e monitorar uma residência com uso de sensores e atuadores (Biegelmeyer, 2015). A ampla implantação de tecnologias da *IoT* na indústria automobilística poderia economizar US$100 bilhões anualmente em reduções de acidentes, de acordo com McKinsey (2015).

Diante disto, a comunicação viabiliza o envio de comandos para componentes inteligentes, caracterizando assim um sistema inteligente e autônomo, o que corrobora para utilizar tecnologias digitais agregadas às disciplinas tradicionais.

Outro ponto bastante relevante é a gestão do consumo de energia elétrica para a diminuição dos gastos (Stevan Junior & Farinelli, 2018). Na contemporaneidade, muita energia é desperdiçada devido ao tempo que dispositivos ficam ligados sem necessidade.

> Um exemplo prático da utilização da *IoT* e que está ligado ao conceito de casa inteligente é o controle de iluminação residencial, esta automação consegue trazer grandes benefício como praticidade, estética e facilidade. No entanto, até mesmo uma simples automatização da iluminação residencial gera controles precários, onde a iluminação é ativada mesmo quando não é necessária. Esses sistemas, por exemplo, costumam definir horários específicos para ativar a iluminação sem considerar fatores sazonais, onde a iluminação natural pode estar por mais tempo. (Borges, 2023)

Neste caso, um sistema inteligente de iluminação permitiria um melhor aproveitamento da iluminação natural, podendo gerar uma economia no consumo de energia. Nesse projeto, será desenvolvido um sistema de iluminação inteligente, que poderá ser usado com a mínima intervenção humana possível para controlá-lo. Funcionará conforme a luminosidade do ambiente, de forma automatizada.

1.2 Objetivos

Nessa subseção, serão apresentados o objetivo geral e objetivos específicos do projeto desenvolvido.

1.2.1 Objetivo Geral

Desenvolver um sistema inteligente de iluminação, utilizando o conceito da Internet das Coisas (*IoT*), que irá permitir um melhor aproveitamento da iluminação natural, podendo gerar uma economia no consumo de energia.

1.2.2 Objetivos Específicos

Os objetivos específicos são:

- Definir qual sensor será usado no protótipo;
- Definir o meio de comunicação entre o dispositivo microcontrolado e o sensor;
- Levantar custos e analisar a viabilidade econômica;
- Realizar a montagem de um protótipo;
- Analisar o consumo de energia e verificar se houve economia;
- Avaliar o desempenho do sistema implementado;

2 FUNDAMENTAÇÃO TEÓRICA

Nessa seção será apresentada a fundamentação teórica do projeto desenvolvido.

2.1 Base Teórica

A Internet das Coisas (*IoT*) — sensores e atuadores conectados por redes a sistemas de computação — recebeu enorme atenção nos últimos cinco anos (McKinsey, 2015).

> Conectar todos esses objetos diferentes e adicionar sensores a eles, adiciona um nível de inteligência digital a dispositivos que de outra forma seriam burros, permitindo que eles comuniquem dados em tempo real sem envolver um ser humano. A Internet das Coisas está tornando a estrutura do mundo ao nosso redor mais inteligente e responsiva, mesclando os universos digital e físico.
>
> Praticamente qualquer objeto físico pode ser transformado em um dispositivo *IoT* se puder ser conectado à Internet para ser controlado ou comunicar informações. Uma lâmpada que pode ser ligada usando um aplicativo de smartphone é um dispositivo *IoT*, assim como um sensor de movimento ou um termostato inteligente em seu escritório ou um poste de luz conectado. (Ranger, 2020).

Os sensores transformam energia em sinal elétrico que, por sua vez, pode ser recebida pelo microcontrolador digitalmente e por seguinte fazer o controle do sistema (Fernandes, 2011). São o ponto chave do sistema, pois o controle do sistema de iluminação se baseia neles.

No mercado, há várias opções de sensores que podem ser usados para o objetivo do projeto. Por exemplo, sensores de presença e de detecção de luz solar.

Atualmente, é bastante comum usar dispositivos inteligentes como o google home, amazon Alexa, entre outros. Devido a sua facilidade, já que são controlados por voz, bastando apenas uma palavra para serem acionados, e para entregar o resultado ao comando do usuário, são utilizados recursos de inteligência artificial. No mercado, lâmpadas capazes de serem integradas com esses dispositivos já são vendidas.

Uma lâmpada da marca philips, modelo HUE, custa em torno de R$ 250.00, compatível com a amazon Alexa, entretanto, é um valor alto e não possui nenhum tipo de sensor integrado, o sensor de luminosidade é vendido separadamente pela marca por R$ 300.00. Portanto, o conjunto saí, por no mínimo, R$ 550.00, sem contar com o dispositivo inteligente.

Um sistema de iluminação inteligente comum vai possuir apenas um tipo de controle, para ligá-lo ou desligá-lo e será preciso usar um aplicativo para isto, ou seja, necessita de intervenção humana. Portanto, não é totalmente automatizado.

Sistemas de iluminação que utilizam lâmpadas com sensores de presença embutidos já existem no mercado, porém, esse sensor não é suficiente, já que o mesmo pode detectar o movimento até de um inseto.

A automação tem como principal objetivo garantir conforto e operação eficiente, fazendo com que o sistema funcione o dia inteiro sem falhas. Portanto, a automação visa a facilidade para que o sistema de iluminação seja ligado ou desligado de forma autônoma.

2.2 Microcontrolador

O projeto foi montado utilizando um microcontrolador chamado ESP8266, este dispositivo integra o chip ESP8266, uma interface *USB*-serial e um regulador de tensão de 0V (volts) a 3.3V. A programação pode ser realizada por meio do uso da linguagem LUA ou da *IDE* do Arduino, utilizando a comunicação através de um cabo micro-*USB*. Além disso, ele vem com uma antena incorporada e um conector micro-*USB* para conexão ao computador, além de oferecer 11 pinos de entrada/saída e um conversor analógico-digital. Possui as seguintes especificações técnicas:

- ESP8266 ESP-12E
- *Wireless* padrão 802.11 b/g/n
- Antena embutida
- Modos de operação: STA/AP/STA+AP
- Suporta 5 conexões *TCP/IP*
- Porta *GPIO* (Entrada/Saída de Propósito Geral): 11
- *GPIO* com funções de *PWM*, *I2C*, *SPI*, etc
- Tensão de operação: 4.5 ~ 9V
- Taxa de Transferência: 110-460800 *bps*
- Conversor analógico digital (*ADC*)

O módulo ESP8266 possui arquitetura *RISC* (Conjunto de Instruções Reduzido) de 32 *bits*, possuindo 32 *KBYTES* de memória *RAM* para instruções; 96 *KBYTES* de *RAM* para dados; 64 *KBYTES* de *rom* para *boot*, sendo fabricado pela Espressif. Esse microcontrolador, por sua vez, é responsável pela conexão *WiFi*, utilizando o protocolo *TCP/IP*.

2.3 Módulo relé

Um módulo relé é um dispositivo eletrônico que consiste em vários componentes essenciais para controlar a operação de um relé eletromagnético. O componente central do módulo é o próprio relé eletromagnético. Ele consiste em uma bobina que, quando energizada, cria um campo magnético que ativa um interruptor mecânico (geralmente chamado de contato) para abrir ou fechar circuitos elétricos.

Um transistor, geralmente do tipo NPN (Negativo-Positivo-Negativo) ou PNP (Positivo-Negativo-Positivo), é usado para acionar a bobina do relé. Ele atua como um interruptor controlado eletronicamente, permitindo que uma corrente pequena controle uma corrente maior necessária para acionar o relé. Em sistemas que usam transistores NPN, um resistor de base é usado para limitar a corrente que flui para a base do transistor. Em sistemas que usam transistores PNP, um resistor de emissor é utilizado para limitar a corrente que flui para o emissor do transistor.

Um diodo é frequentemente colocado em paralelo com a bobina do relé. Este diodo é chamado de diodo de proteção ou diodo de supressão. Sua função é fornecer um caminho para a corrente induzida pela bobina do relé quando a energia é desligada. Isso protege outros componentes eletrônicos no circuito contra picos de tensão prejudiciais.

Alguns módulos relé incluem um opto acoplador (também conhecido como acoplador óptico ou opto-isolador). Ele isola eletricamente a parte de controle (baixa potência) da parte de potência (alta potência) do circuito, aumentando a segurança e evitando interferências. Conectores e terminais são fornecidos para facilitar a conexão do módulo relé a outros dispositivos ou circuitos;

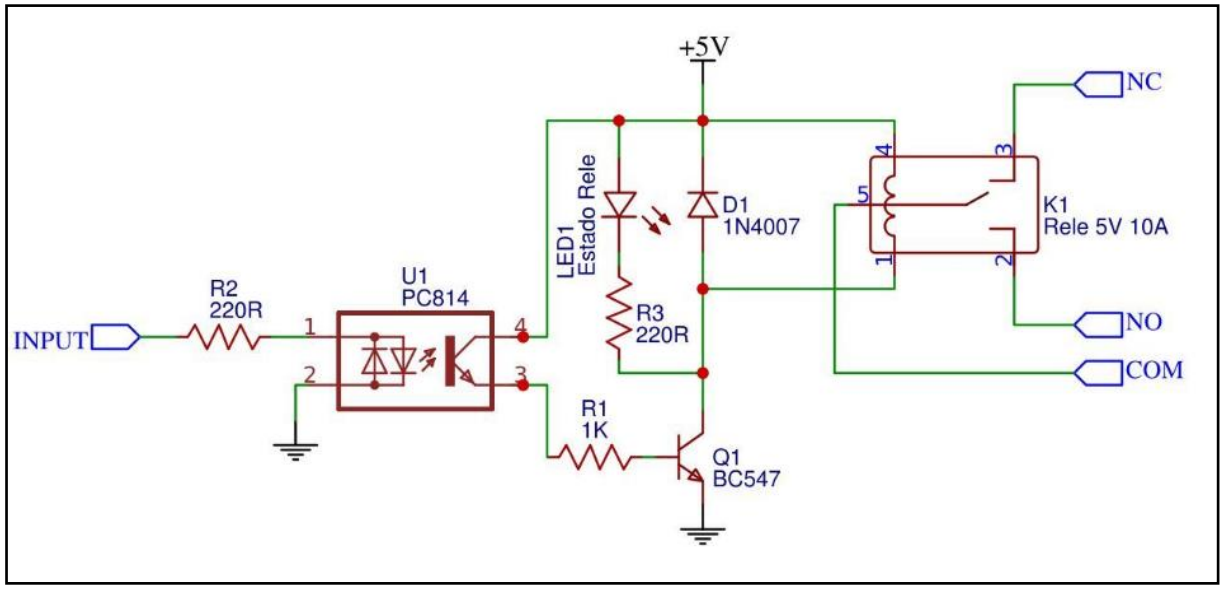
Esses componentes trabalham em conjunto para permitir o controle de dispositivos de alta potência por meio de sinais de baixa potência, oferecendo proteção contra interferências e picos de tensão. O design específico pode variar entre diferentes fabricantes e modelos de módulos relé.

O módulo relé possui apenas saídas digitais, podendo controlar cargas maiores e dispositivos como motores *AC* (corrente alternada) ou *DC* (corrente contínua), eletroímãs,

solenoides e lâmpada como a usada no protótipo. Possui um led para indicar o estado da saída, sendo vermelho para nível lógico baixo (desligado) e verde para nível lógico alto (ligado).

O módulo funciona com tensão de 5V, e pode acionar cargas de até 220 *VAC* (voltagem corrente alternada) ou 30 *VDC* (voltagem corrente contínua), suportando a corrente máxima de 10 amperes. Possui três pinos de saída, chamados de normalmente fechado, comum e normalmente aberto, respectivamente. Quando a bobina está energizada, o comum fecha circuito com o normalmente aberto, permitindo a passagem de corrente. Enquanto com a bobina não energizada, o comum fecha circuito com o normalmente fechado, interrompendo a passagem de corrente.

Figura 1 – Módulo relé



Fonte: Acervo dos autores (2023)

2.4 Sensor de Luminosidade LDR

O sensor de luminosidade *LDR* (resistor variável conforme incidência de luz), é um componente eletrônico do tipo passivo, ou seja, não gera e nem requer energia para operação, variando a resistência de acordo com a intensidade da luz que incide sobre ele. Conforme a medida que a intensidade da luz aumenta, a sua resistência diminui. É utilizado principalmente para controle de iluminação, podendo operar entre a temperatura de –30 até 70 graus celsius.

Figura 2 – Sensor LDR

Fonte: PLATT, 2018

2.5 Protocolos de Comunicação

Na comunicação digital, o *TCP/IP* (Protocolo de controle de Transmissão/Protocolo de Internet) e o *HTTP* (Protocolo de Transferência de Hipertexto), desempenham papéis principais sendo a base que sustenta a estrutura que permite a troca de informações na internet e são utilizados neste projeto.

Os protocolos de comunicação referem-se a interações entre diversos dispositivos, o que demanda a presença de um software para conduzir essas interações, o qual é, por sua vez, implementado em cada hardware de rede. Neste caso, um exemplo que pode se usar para explicar melhor a funcionalidade do protocolo é o de transmitir uma informação (conhecido também como "pacote" ou "dado") de um celular para um microcontrolador, ambos devidamente configurados de maneira correta. Ao realizar a transferência de um pacote entre os dispositivos, existe a necessidade de conhecer o receptor dessa informação, para isso são utilizados protocolos de comunicação. Em virtude disso, o protocolo é uma norma, o qual estabelece um padrão de comunicação para que os dispositivos sejam interconectados. (Silva, 2019)

O protocolo de internet (*IP*) é uma sequência numérica única atribuída a cada dispositivo conectado a uma rede de computadores para comunicação. O protocolo *TCP/IP* nasceu com o intuito de progredir com a independência do equipamento e sistema operacional, a compatibilidade é um ponto importante para que a conectividade seja adquirida por um vasto número de equipamentos. Além de apenas conectar máquinas, o *TCP/IP* recupera pacotes de dados que em seu caminho se perderam por alguma oscilação ou falha de envio, e tenta retorná-los. A queda ou oscilação de sinal que afeta negativamente o envio de pacotes pode estar

atrelado à algum mal contato tanto por parte do *host* quando destinatários, a recuperação ocorre não necessariamente em tempo real, mas em qualquer ponto da transmissão de dados. (Camargo, 2020)

O *HTTP* (*Hypertext Transfer Protocol*) é um protocolo utilizado para envio de textos, mídias e outros dados pela Internet. Baseado no paradigma de solicitação/resposta, este protocolo viabiliza a interação entre um cliente e um servidor. Durante esse processo, o cliente submete uma requisição ao servidor e permanece em espera até receber a resposta correspondente. (A. Silva, 2021)

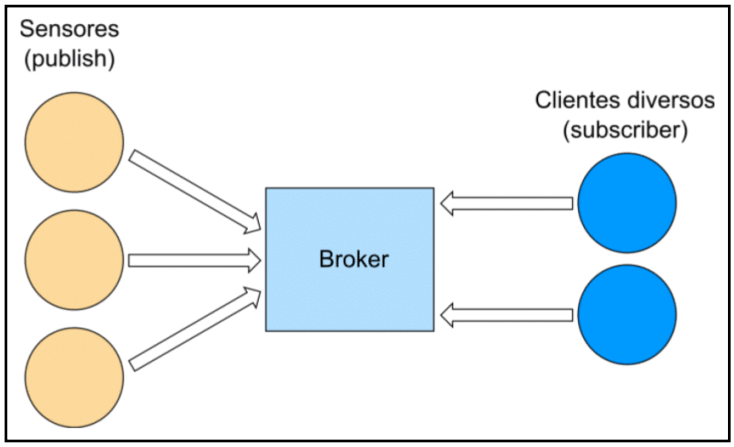
2.6 Topologia de Rede

Na área da *IoT*, é necessário compreender certos conceitos previamente, como os termos *device* (dispositivo) e *broker* (intermediário), que se referem a diferentes componentes na arquitetura de comunicação.

Um *device* se refere a um objeto físico ou sistema que possui a capacidade de se conectar a uma rede e interagir com outros componentes. Esses dispositivos frequentemente possuem sensores, atuadores e módulos de comunicação.

Um *broker* é um intermediário que facilita a comunicação e coordenação entre diferentes dispositivos em uma mesma rede. No protocolo *MQTT* (Transporte de Filas de Mensagem de Telemetria), comumente utilizado na *IoT*, um *broker* é responsável por gerenciar o sistema de mensagens, sendo assim, atua como um ponto de controle para garantir a comunicação.

Figura 3 – *Broker*



Fonte: Acervo dos autores (2023)

2.7 Blynk

Com o objetivo de proporcionar uma melhor experiência ao usuário, o sistema de iluminação inteligente foi integrado ao aplicativo Blynk, e também, uma interface web com acesso pelo navegador, a qual será mostrada posteriormente.

Blynk é uma plataforma que permite criar aplicativos para dispositivos da Internet das Coisas (*IoT*) de maneira fácil e intuitiva, sem a necessidade de programação complexa. Possui um editor, onde é possível apenas arrastar e soltar para criar aplicativos *iOS* e *android* sem código, tal como demonstrado na figura a seguir. Além de um painel para gerenciar dispositivos, usuários e dados. (Blynk, 2023)

Na atualidade, o aplicativo Blynk conta com mais de oitocentos mil dispositivos conectados globalmente, em média são processadas mais de 300 bilhões de mensagens de hardware e o tempo médio de construção de um protótipo com esse ambiente de desenvolvimento é de dois dias. (Blynk, 2023)

Sendo assim, trata-se de uma plataforma do tipo *low-code*, o que significa uma abordagem de código reduzido, a qual se destaca por ser uma solução acessível no cenário de desenvolvimento *IoT*, visto que para este projeto, não foi necessário adquirir um plano pago.

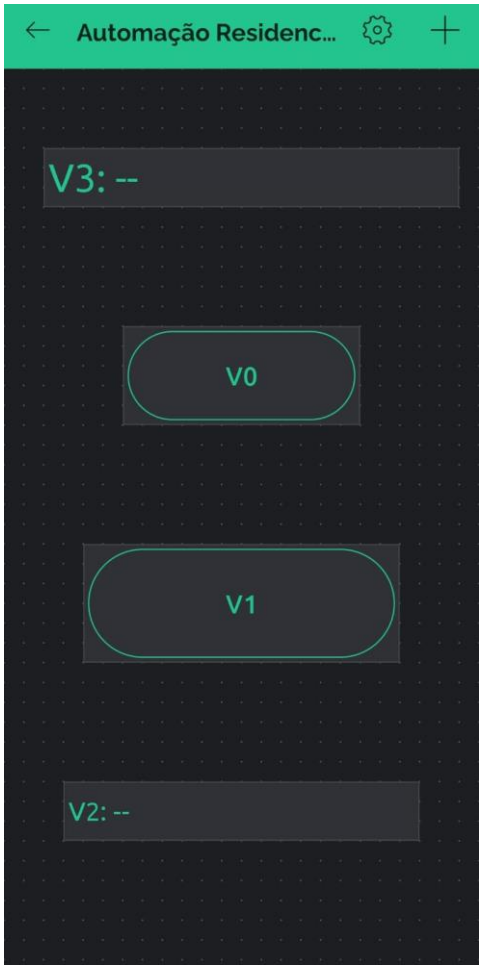
Figura 4 – Painel de controle Blynk

Fonte: Acervo dos autores (2023)

O design mostrado acima é personalizável, o usuário precisa apenas arrastar e soltar os elementos gráficos para ajustar de acordo com suas preferências.

É essencial para a comunicação entre esta aplicação e o ESP8266, configurar os seguintes pinos virtuais, V0 para a função ligar ou desligar a lâmpada, V1 para ativar ou desativar o sensor de luminosidade, V2 para receber um dado do tipo *string* do IP local gerado e V3 para o título desejado ao acessar a interface de controle, nesse caso, foi adotado o título “Controle de Lâmpadas”. Estes pinos virtuais serão posteriormente utilizados na programação do microcontrolador.

Figura 5 – Configuração do aplicativo Blynk



Fonte: Acervo dos autores (2023)

Este aplicativo possui uma *cloud* (nuvem) incluída para a conexão dos dispositivos, a qual atua como um *broker*, logo, facilitando a troca de informações entre os dispositivos conectados. Portanto, ela é responsável por receber e processar os dados oriundos dos dispositivos.

O microcontrolador executa a biblioteca Blynk, que se comunica com o servidor Blynk *Cloud*. O aplicativo em um dispositivo móvel interage com o servidor, enviando comandos ou solicitações à nuvem, como dados do sensor ou atualizações de *status* (estado) e ela os encaminha para o dispositivo correspondente. Sendo assim, o aplicativo é atualizado conforme necessário.

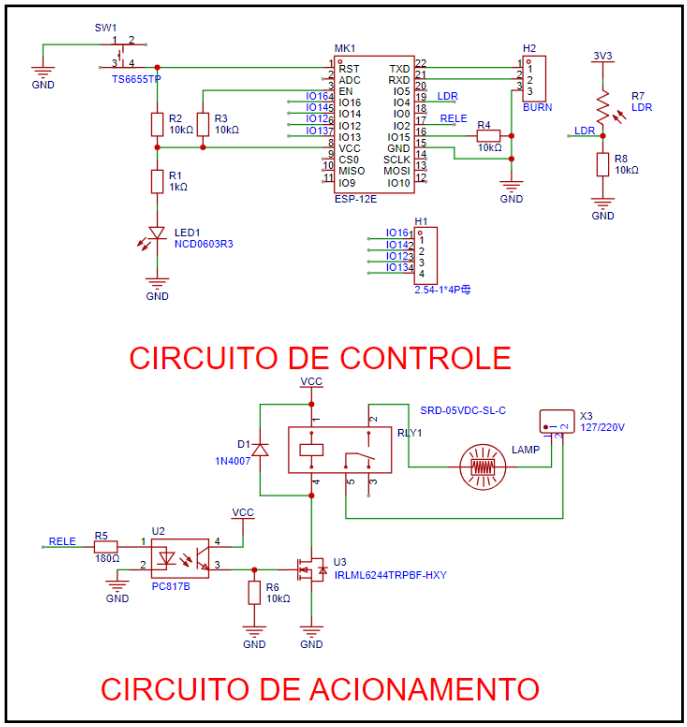
3 DESENVOLVIMENTO

Nessa seção será apresentado todo o desenvolvimento do projeto, que consiste no esquema elétrico, componentes, instalação dos pacotes necessários, na montagem, no processo de impressão 3D e na programação do circuito.

3.1 Esquema elétrico

A figura abaixo representa o esquemo elétrica do projeto de um sistema de iluminação inteligente. O esquema destaca a interconexão dos componentes, incluindo o sensor de luminosidade, microcontrolador e módulo relé, sendo assim, serve como um guia para a implementação prática.

Figura 6 – Esquema elétrico

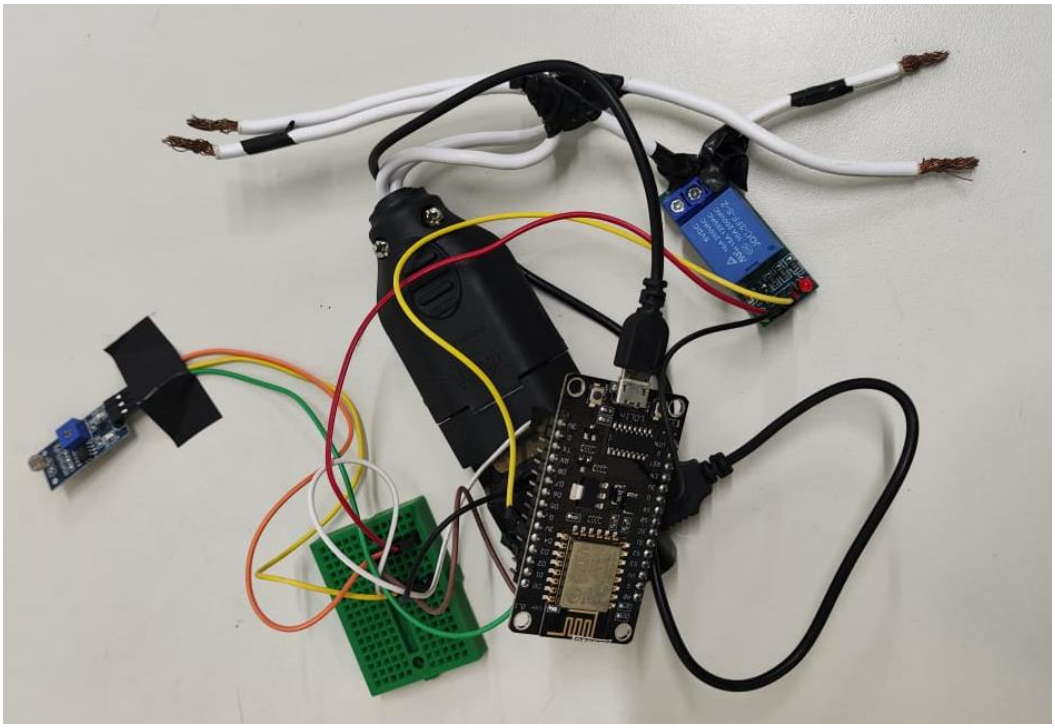


Fonte: Acervo dos autores (2023)

3.2 Componentes

Este projeto iniciou com a pesquisa dos componentes necessários. Para o protótipo, os seguintes componentes foram utilizados: ESP8266, modulo relé de 1 canal 5V, sensor de luminosidade *LDR*, tomada de 3 pinos fêmea, tomada de 2 pinos macho, fonte *USB* de 5V, soquetes para lâmpadas, 2 lâmpadas de 9W, 1 metro de fio de 10mm, 1 *protoboard* (placa de prototipagem) 170 pontos, *8* jumpers (ponte) e cabo micro-*USB*.

Figura 7 – Componentes utilizados

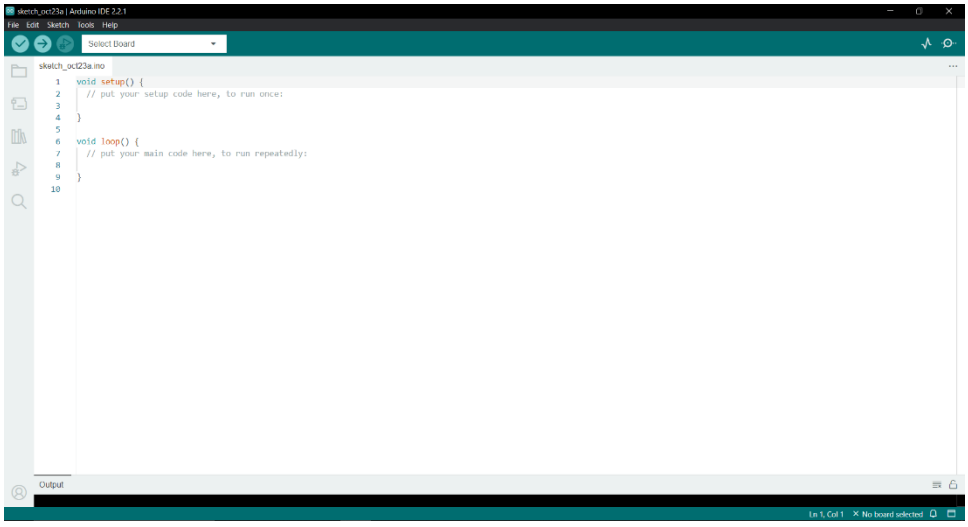


Fonte: Acervo dos autores (2023)

3.3 Instalação dos pacotes necessários

Para a programação da placa de desenvolvimento foi utilizada a IDE do arduino. Sendo necessário baixar pacotes para reconhecimento do ESP8266, como visto nas figuras a seguir.

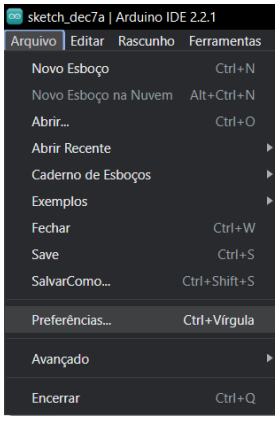
Figura 8 – Interface da IDE do arduino



Fonte: Acervo dos autores (2023)

Antes de fazer o download dos pacotes, é necessário adicionar o link onde esses dados serão buscados.

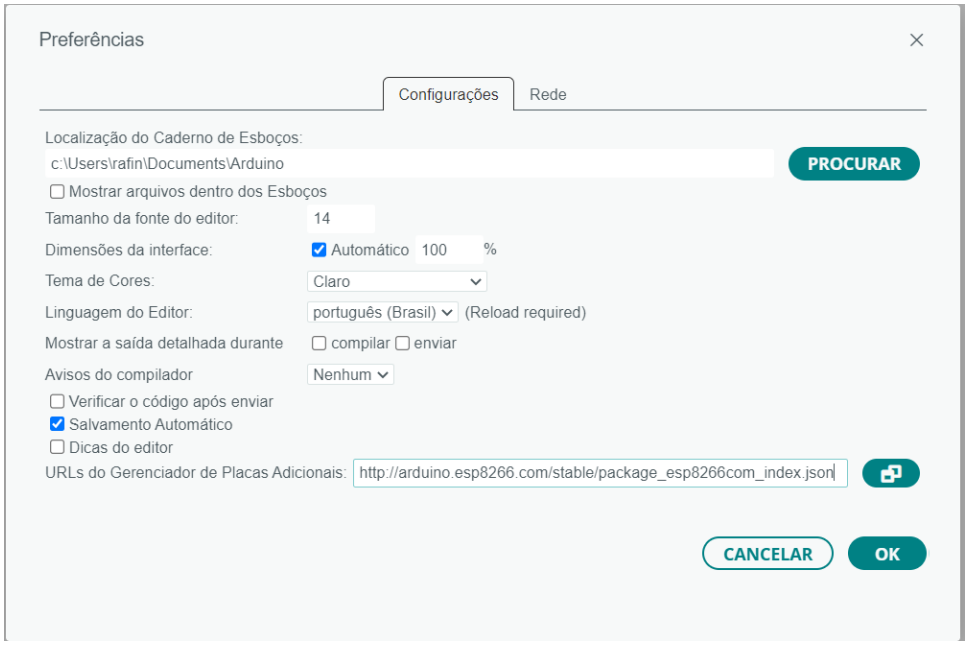
Figura 9 – Primeira passo da instalação



Fonte: Acervo dos autores (2023)

Uma nova janela será aberta, onde no campo “URLs adicionais para Gerenciadores de Placas”, será necessário colar o seguinte endereço: http://arduino.esp8266.com/stable/package_esp8266com_index.json

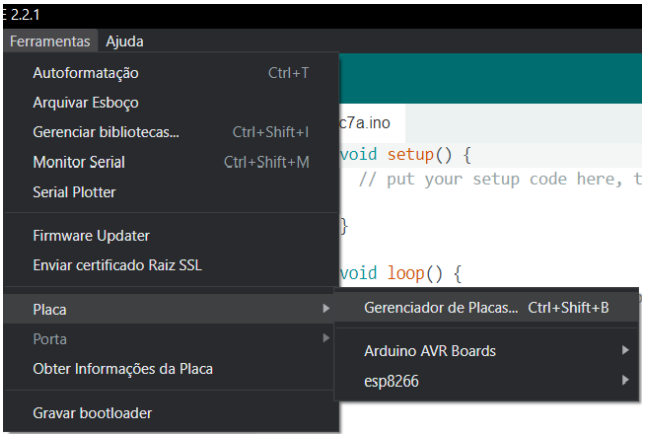
Figura 10 – Adição de endereço



Fonte: Acervo dos autores (2023)

Conforme a figura 4, o download é feito na aba “Gerenciador de Placas”. Sendo necessário expandir o menu “Ferramentas” para a opção aparecer.

Figura 11 – Local do *download*



Fonte: Acervo dos autores (2023)

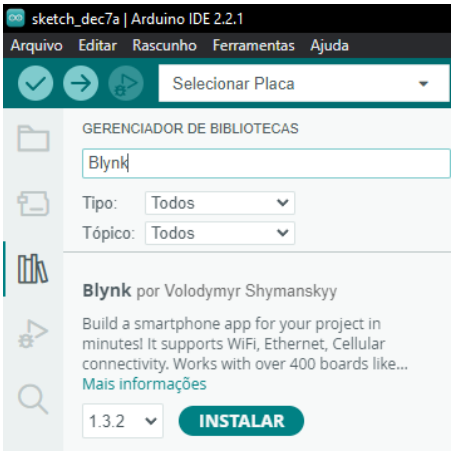
Após acessar o gerenciador de placas, basta digitar “ESP8266” na barra de pesquisa e fazer a instalação do pacote, conforme a figura 5.

Figura 12 – Instalação do pacote para reconhecimento do módulo

Fonte: Acervo dos autores (2023)

E também é necessário baixar a biblioteca Blynk para a integração com o aplicativo. Sendo preciso expandir o menu “Ferramentas” para mostrar a opção “Gerenciar Bibliotecas”. Logo, para aparecer a biblioteca, basta digitar “Blynk” na barra pesquisa, como mostrado na figura em sequência.

Figura 13 – Instalação de biblioteca para integração



Fonte: Acervo dos autores (2023)

3.4 Montagem do circuito

Para a montagem do circuito, foram seguidos os passos descritos abaixo.

1. Conexão do Módulo Relé: A porta D4 do ESP8266 foi designada como *output* e conectada ao pino de sinal (*IN*) do módulo relé por meio de um *jumper*. Essa configuração permite ao ESP8266 controlar o estado do relé para ativar ou desativar as lâmpadas de forma remota.
2. Integração do Módulo LDR: A porta D2 do ESP8266 foi configurada como *input* e conectada ao pino de tensão de saída (*DO*) do módulo *LDR* utilizando um *jumper*. Essa conexão permite ao ESP8266 receber informações em nível lógico de tensão de 0 a 3.3V, que resultará no controle de luminosidade ambiente, possibilitando a automação da iluminação com base na luz natural disponível.
3. Alimentação e Conexões de Energia: Três *jumpers* foram utilizados para interligar o pino de 3V do *ESP8266* com os pinos de *VCC* do módulo relé e do módulo *LDR*, com o auxílio de uma *protoboard* de 170 pontos. Isso assegura o fornecimento adequado de energia para esses componentes. Três *jumpers* adicionais foram empregados para conectar os pinos de *GND* do *ESP8266*, do módulo relé e do módulo *LDR* à *protoboard*, estabelecendo uma referência comum de terra.
4. Conexão dos Soquetes das Lâmpadas: Os soquetes das lâmpadas foram conectados ao terminal normalmente aberto (*NO*) do módulo relé. Essa configuração permite que o relé controle o circuito das lâmpadas, interrompendo ou permitindo o fluxo de corrente conforme comandos do ESP8266.
5. Alimentação do ESP8266: Uma tomada fêmea foi conectada aos fios de entrada do circuito, permitindo a alimentação do ESP8266 com 5V por meio de uma fonte *USB* e de um cabo micro-*USB*. Essa abordagem simplifica a alimentação do microcontrolador.
6. Acoplamento das Lâmpadas: Uma lâmpada de 9W foi acoplada a cada soquete. O controle do relé possibilita a ativação e desativação destas lâmpadas de forma remota, proporcionando flexibilidade no gerenciamento da iluminação.
7. Conexão e Rede Elétrica: Uma tomada macho foi adicionado ao fio de alimentação do circuito, possibilitando a conexão do sistema à rede elétrica padrão de 220V. Isso completa a configuração do circuito de iluminação inteligente para controle residencial autônomo.

3.5 Impressão 3D

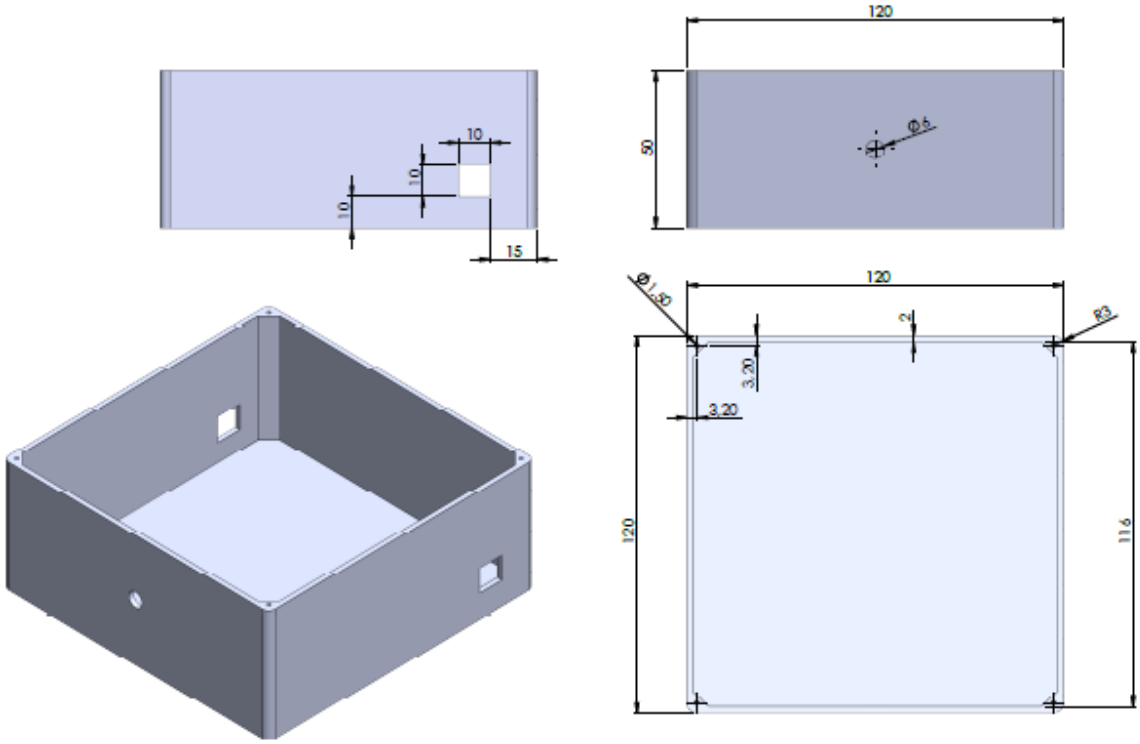
Para a montagem do circuito, foi confeccionado uma caixa em impressão 3D, para garantir a proteção dos componentes em ambientes expostos ao ar livre, a fim de evitar os efeitos adversos de fenômenos naturais, como chuva e vento.

Foi escolhido o filamento do tipo abs (acrilonitrila butadieno estireno) para a produção da caixa. É uma resina termoplástica produzida a partir do petróleo, sendo a base para a criação de materiais de vários tipos, e, devido à sua versatilidade, tem alto valor mercadológico na indústria. Ademais, é um material mais barato que filamento do tipo pla (ácido poliláctico) e possui resistência térmica de até 85º C.

A caixa possui as seguintes dimensões: 120mm de comprimento, 120mm de largura, 50mm de altura, 2mm de espessura em cada parede. Para permitir a passagem e o gerenciamentos dos cabos responsáveis por conectar a lâmpada, a estrutura conta com aberturas posicionadas nas laterais, com o tamanho de 10mm de altura e 10mm de comprimento. E também conta com uma abertura de 6mm de diâmetro localizada no topo, com a finalidade de maximizar a incidência de luz solar no sensor de luminosidade. Para possibilitar o fácil acesso ao circuito, a tampa da caixa é parafusada, cujo cada furo aonde vai o parafuso, possui 1.5mm de diâmetro, para minimizar o impacto no design.

É importante atentar-se para as medidas representadas nas figuras subsequentes, as quais estão em milímetros, sendo necessário converter, caso seja preciso trabalhar em outro tipo de unidade de medida.

Figura 14 – Esquema de impressão



Fonte: Acervo dos autores (2023)

3.6 Programação

Na implementação da parte de programação da placa de desenvolvimento, antes de começar o código, é necessário importar as bibliotecas responsáveis pela inicialização da conexão *Wi-Fi*, do servidor *WEB* e da plataforma Blynk no ESP8266, como mostrado acima. O servidor *WEB* e o Blynk desempenham um papel essencial no controle do sistema de iluminação, permitindo que os usuários enviem comandos para ativar ou desativar o sensor de luminosidade, além de oferecer a flexibilidade de ligar ou desligar a lâmpada se o sensor estiver desativado de maneira conveniente e eficaz.

Para a conexão ser estabelecida com a plataforma Blynk, é necessário definir as constantes referentes a identificação e autenticação do projeto.

- BLYNK_PRINT Serial: Envia mensagens de depuração para a portal serial.
- BLYNK_TEMPLATE_ID: ID do modelo Blynk correspondente.
- BLYNK_TEMPLATE_NAME: Nome do modelo Blynk.
- BLYNK_AUTH_TOKEN: Token de autenticação Blynk gerado.

Figura 15 – Definição de constantes

```
1   #define BLYNK_PRINT Serial
2   #define BLYNK_TEMPLATE_ID ""
3   #define BLYNK_TEMPLATE_NAME ""
4   #define BLYNK_AUTH_TOKEN ""
```

Fonte: Acervo dos autores (2023)

A fim de assegurar o correto funcionamento do projeto, é essencial a importação das bibliotecas específicas relacionadas ao hardware e às funcionalidades utilizadas. Essas bibliotecas fornecem os recursos e métodos importantes para a integração do código.

Para garantir a conexão bem-sucedida do dispositivo à rede *Wi-Fi*, é essencial incorporar no código as informações correspondentes de nome e senha, sem permitir qualquer discrepância em relação aos valores configurados nos dispositivos, assegurando simultaneamente que a rede esteja configurada para operar exclusivamente na frequência de 2.4 GHz, pois o ESP8266 não suporta a frequência 5 Ghz. Essa etapa na programação garante uma sincronia precisa entre as credenciais do módulo e da rede, bem como uma conformidade estrita com a frequência desejada, evitando qualquer desvio que possa comprometer a conectividade e a funcionalidade do sistema.

Figura 16 – Configurações iniciais

```
6   #include <ESP8266WiFi.h>
7   #include <BlynkSimpleEsp8266.h>
8   #include <ESP8266WebServer.h>
9
10  char ssid[] = "nome_do_wifi";
11  char password[] = "senha_do_wifi";
```

Fonte: Acervo dos autores (2023)

Conforme a figura abaixo, a linha 14 refere-se à inicialização do servidor web na porta 80. Isso significa que o ESP8266 estará ouvindo solicitações *HTTP* na porta 80. Na linha 16, a constante do tipo *int* (inteiro) *relayPin*, define o número do pino ao qual o relé está conectado. Neste caso, ele está conectado no pino de valor 2, que corresponde a saída D4 do microcontrolador utilizado neste projeto. Cada pino possui um valor específico, sendo essencial verificar o datasheet ao escolher o pino que será usado.

Na linha 17, é inicializada a constante denominada *LDRPin*, referente ao pino que o sensor de luminosidade está conectado, nesse caso é o pino de valor 4, que corresponde a saída

D2 do microcontrolador. E na próxima linha, é instanciada uma variável do tipo booleana *sensorEnabled*, ou seja, aceita apenas valor "*true*" (verdade) para habilitar ou "*false*" (falso) para desabilitar o sensor. Enquanto na linha 19, é inicializada uma variável chamada *relayState*, do mesmo tipo da anterior, portanto, restrita a mesma condição e é usada para rastrear o estado do relé, para identificar quando o módulo está ligado ou desligado. E as variáveis que começam com *TimeCount*, são utilizadas para certificar de que as variáveis do tipo *long*, que detêm o objetivo de cronometrar, sejam zeradas quando necessário.

Figura 17 – Definição de variáveis

```
14  ESP8266WebServer server(80);
15
16  const int relayPin = 2;
17  const int LDRPin = 4;
18  bool sensorEnabled = false;
19  bool relayState = false;
20  bool TimeCount1 = false;
21  bool TimeCount2 = false;
22  bool TimeCount3 = false;
23  long TimeOff;
24  long TimeOn1;
25  long TimeOn2;
```

Fonte: Acervo dos autores (2023)

A função *void setup*() é a primeira função a ser executada no programa, e é executada apenas uma vez. Para iniciar a comunicação serial, é usada a função *Serial.begin()*, cujo parâmetro serve para definir a taxa de transmissão para enviar e receber dados pela porta serial para comunicação, nesse contexto, a taxa usada foi de cento e quinze mil e duzentos bits por segundos.

Enquanto a função *pinMode*() é preciso passar dois parâmetros dentro dos parênteses, um para identificar o pino e o outro parâmetro refere-se à configuração do pino, como entrada ou saída, nesse contexto, o *LDRPin*, é necessário configurar como entrada (*INPUT*), porque ele fornece informações ao microcontrolador sobre a intensidade da luz no ambiente. Em paralelo, é preciso configurar o *relayPin* como saída (*OUTPUT*), já que é usado para enviar sinais de controle para o acionamento do relé.

Na função *digitalWrite*(), segue a mesma parametrização, porém no segundo parâmetro, trata-se de configurar o estado do pino, nível lógico *HIGH* (alto) ou *LOW* (baixo), nesse caso é configurado como alto, o que indica que o módulo relé está inicialmente desligado.

Figura 18 – Primeira função a ser executada

```
27  void setup()
28  {
29    Serial.begin(115200);
30
31    pinMode(LDRPin, INPUT);
32    pinMode(relayPin, OUTPUT);
33    digitalWrite(relayPin, HIGH);
```

Fonte: Acervo dos autores (2023)

A função *server.on*() é usada para definir como o servidor deve lidar com solicitações *HTTP*, ou seja, para configurar as rotas e associá-las a funções de tratamento (ou "*handles*"), posteriormente as funções serão discorridas com mais detalhes. E a função *server.begin*(), serve para iniciar o processamento destas solicitações.

Figura 19 – Solicitações HTTP

```
36  server.on("/", HTTP_GET, handleAtivarSensor);
37  server.on("/ligar", HTTP_GET, handleLigar);
38  server.on("/desligar", HTTP_GET, handleDesligar);
39  server.on("/ativarSensor", HTTP_GET, handleAtivarSensor);
40  server.on("/desativarSensor", HTTP_GET, handleDesativarSensor);
41  server.begin();
```

Fonte: Acervo dos autores (2023)

O seguinte trecho de código se refere a integração com o Blynk, para iniciar a conexão, é preciso passar os parâmetros definidos anteriormente. O método *handleAtivarSensor*() é o responsável pela ativação do sensor, sendo assim, o aplicativo é inicializado com o sensor ativo. E o endereço *IP* gerado do servidor *WEB* local será utilizado adiante.

Figura 20 – Chamada de métodos da biblioteca Blynk

```
44    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
45
46    handleAtivarSensor();
47
48    Blynk.virtualWrite(V2, "IP Address: " + WiFi.localIP().toString());
49    Blynk.virtualWrite(V3, "Controle de lâmpadas");
50  }
```

Fonte: Acervo dos autores (2023)

A função *void loop*(), como o próprio nome diz, é a principal função e é executada continuamente, enquanto a placa está ligada. É usada para repetir de forma cíclica o código dentro da função, ou seja, o que precisa ser atualizado constantemente, como leitura de sensores, controle de atuadores, tomada de decisões, processamento de dados, etc. Como a função *server.handleClient*() está sempre aguardando novas solicitações *HTTP* dos clientes, é necessária ser chamada dentro da função *void loop*(). Quando um cliente faz uma solicitação *HTTP*, como acessar uma determinada rota ou *URL* no servidor, a função *server.handleClient*() é responsável por verificar qual rota foi solicitada e se a mesma existe, e então direcionar essa solicitação para a função de tratamento respectiva dessa rota. Essa função de tratamento será executada para responder à solicitação.

Portanto a função *server.handleClient*() desempenha um papel fundamental na execução de um servidor web no microcontrolador ESP8266, garantindo que ele seja capaz de receber, rotear e responder a solicitações *HTTP* de clientes, como navegadores da web ou outras aplicações que fazem solicitações *HTTP* para o dispositivo. Ela é chamada repetidamente dentro do loop principal do seu programa para garantir que o servidor web responda às solicitações continuamente.

Figura 21 – Ciclo principal

```
55  void loop()
56  {
57    server.handleClient();
```

Fonte: Acervo dos autores (2023)

É feito uma verificação sobre o estado do sensor, se o retorno for *true*, a execução segue para o próximo bloco.

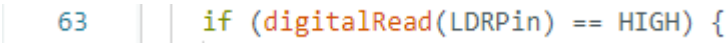
Figura 22 – Verificação do estado do sensor

```
60    if (sensorEnabled) {
```

Fonte: Acervo dos autores (2023)

Dentro do bloco anterior, se a leitura do nível lógico do sensor luminosidade for igual a *HIGH*, ou seja, se o mesmo não estiver detectando luz no ambiente, a execução passa para o bloco seguinte.

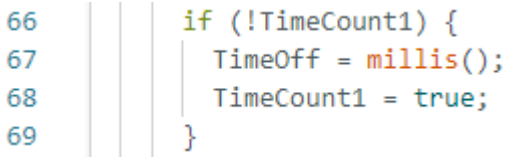
Figura 23 – Verificação de luz no ambiente



Fonte: Acervo dos autores (2023)

Como a variável booleana *TimeCount1* é inicializada como *false*, o seguinte bloco é executado, onde é iniciado um temporizador *TimeOff* utilizando a função *millis*(), que retorna o tempo em milissegundos desde o início do programa, dessa forma, é utilizado para contabilizar o tempo que a lâmpada está desligada. Sendo assim, a variável *TimeCount1* é definida como *true* para indicar que o primeiro temporizador foi iniciado.

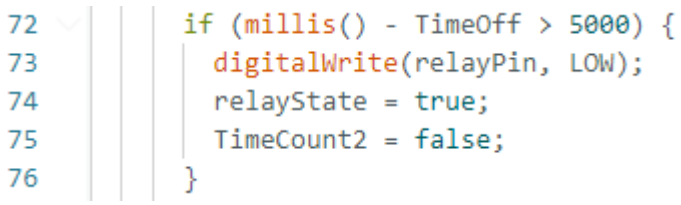
Figura 24 – Início do primeiro temporizador



Fonte: Acervo dos autores (2023)

Se o temporizador *TimeOff* ultrapassar 5000 milissegundos (5 segundos), a execução passa para o próximo bloco, onde o pino do módulo relé é definido como *LOW*, o que faz o sistema ligar a iluminação, logo, a variável *relayState* recebe o atributo *true*, informando que o relé está ativo, e o segundo *timer* chamado *TimeCount2 é* definido como *false,* pois será usado na sequência.

Figura 25 – Ativação após 5 segundos



Fonte: Acervo dos autores (2023)

Se o sensor de luminosidade estiver detectando luz, ou seja, em estado *LOW*, entra em outro bloco de execução, onde é verificado se a variável *TimeCount2* é igual a *false*, ou seja, se o segundo *timer* ainda não foi inicializado, referente ao tempo que a lâmpada está ligada. Portanto, se a condição for satisfeita, é inicializado um temporizador definido como *TimeOn1* e o *TimeCount2* recebe o valor *true* para indicar que o segundo temporizador foi iniciado.

Figura 26 – Inicio do temporizador para a lâmpada ligada

```
78      } else {
79
80        // If LDR is LOW, start the second timer
81        if (!TimeCount2) {
82          TimeOn1 = millis();
83          TimeCount2 = true;
84        }
```

Fonte: Acervo dos autores (2023)

É feita outra verificação para o caso de o *TimerOn1* ultrapassar 5000 milissegundos, o estado do módulo relé transita para *HIGH*, desligando a iluminação, logo, é atribuído *false* para a variável *relayState* para informar que o relé está desativado. E as variáveis *TimeCount1* e *TimeCount3* voltam ao valor *false*.

Figura 27 – Desativação da iluminação após 5 segundos

```
86        if (millis() - TimeOn1 > 5000) {
87          digitalWrite(relayPin, HIGH);
88          relayState = false;
89          TimeCount1 = false;
90          TimeCount3 = false;
91        }
```

Fonte: Acervo dos autores (2023)

Fora do bloco anterior, se a condição da variável *relayState* for igual a *true* e a *TimeCount3* possuir valor *false* for atendida, ou seja, se o relé estiver ativo e o terceiro temporizador não estiver indicando que foi acionado, é iniciado o temporizador *TimeOn2*. E é atribuído valor *true* para a variável *TimeCount3*, portanto, o terceiro temporizador é acionado.

Figura 28 – Início do segundo temporizador

```
96      if (relayState && !TimeCount3) {
97        TimeOn2 = millis();
98        TimeCount3 = true;
99      }
```

Fonte: Acervo dos autores (2023)

O seguinte trecho refere-se ao timer programado, para que durante a noite, o sistema de iluminação inteligente seja desligado após 3 horas consecutivas desde o seu acionamento.

Figura 29 – Iluminação automática

```
95    if (relayState && millis() - TimeOn > 10800000){
96      digitalWrite(relayPin, HIGH);
97    }
```

Fonte: Acervo dos autores (2023)

Se o sensor de luminosidade estiver desabilitado, ou seja, o estado do sensor estiver em *false*, todas as variáveis de controle dos temporizadores são reiniciadas com o valor *false*. O método *Blynk.run()* faz parte da biblioteca Blynk, e é utilizado para a comunicação do microcontrolador e o aplicativo no dispositivo móvel.

Figura 30 – Reinicio das variáveis

```
107   } else {
108     TimeCount1 = false;
109     TimeCount2 = false;
110     TimeCount3 = false;
111   }
112
113   Blynk.run();
114 }
```

Fonte: Acervo dos autores (2023)

O seguinte trecho de código é destinado para o tratamento do *input* V0 no Blynk, isso significa que a função será chamada quando o valor associado a este *input* for alterado no aplicativo. Este método possui a finalidade de ligar ou desligar a lâmpada no modo manual, ao clicar no botão referente no aplicativo.

Figura 31 – Funções do aplicativo

```
117   BLYNK_WRITE(V0) {
118     int value = param.asInt();
119
120     if (value == 1) {
121
122       if (sensorEnabled == false) {
123         digitalWrite(relayPin, LOW);
124       }
125
126     } else {
127       digitalWrite(relayPin, HIGH);
128     }
129   }
```

Fonte: Acervo dos autores (2023)

O seguinte trecho de código é destinado para o tratamento do *input* V2 no Blynk, sendo assim, a função é chamada quando o valor associado a este *input* for alterado no aplicativo. Este método é referente ao acionamento do sensor de luminosidade.

Figura 32 – Função referente ao sensor no aplicativo

```
132   BLYNK_WRITE(V1) {
133     int value = param.asInt();
134
135     if (value == 1) {
136       digitalWrite(relayPin, HIGH);
137       Blynk.virtualWrite(V0, 0);
138       sensorEnabled = true;
139     } else {
140       digitalWrite(relayPin, HIGH);
141       Blynk.virtualWrite(V0, 0);
142       sensorEnabled = false;
143     }
144   }
```

Fonte: Acervo dos autores (2023)

A interface *WEB* foi codificada utilizando a linguagem *HTML* (Linguagem de Marcação de Hipertexto) para o conteúdo estático e a linguagem *CSS* (Folhas de Estilo em Cascata) para a parte de estilização da página, a qual é totalmente personalizável. Primeiro, é codificada a página principal, correspondente ao utilizar o sistema com o sensor de luminosidade habilitado.

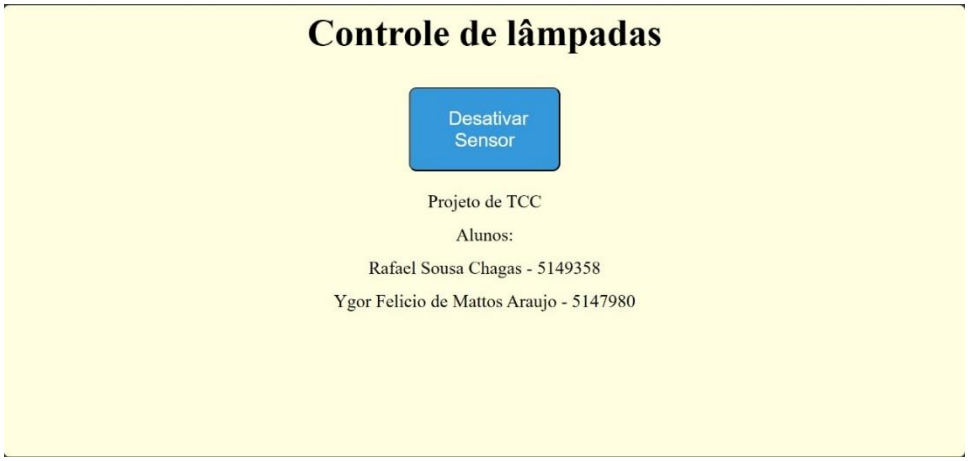
Figura 33 – Código da página principal

```
178 void handleLDR() {
179   String page = "";
180   page += "<head> ";
181   page += "<title>Automa&ccedil;&atilde;o Residencial</title>";
182   page += "</head>";
183   page += "<body>";
184   page += "<h1>Controle de l&acirc;mpadas</h1>";
185   page += "<a href='/desativarSensor'><button class='button' id='desativarSensor'>Desativar Sensor</button></a>";
186   page += "<div class='info'>";
187   page += "<p><font size='5'>Projeto de TCC</font></p>";
188   page += "<p><font size='5'>Alunos:</font></p>";
189   page += "<p><font size='5'>Rafael Sousa Chagas - 5149358</font></p>";
190   page += "<p><font size='5'>Ygor Felicio de Mattos Araujo - 5147980</font></p>";
191   page += "</div>";
192   page += "</body>";
193   page += "</html>";
194   server.send(200, "text/html", page);
195 }
196
```

Fonte: Acervo dos autores (2023)

A imagem abaixo representa o resultado do código acima, a página é iniciada com a opção de desativar o sensor em evidência, isto denota que o sistema já inicia com o sensor ativado. Para acessar as páginas é necessário estar conectado a mesma rede do microcontrolador e copiar o *IP* gerado na barra de endereços de um navegador *WEB*.

Figura 34 – Página Principal



Fonte: Acervo dos autores (2023)

Em sequência, a página codificada no seguinte trecho corresponde quando o sensor está desabilitado, ou seja, para o uso manual do sistema.

Figura 35 – Código da página secundária

```
149  void handleMain() {
150    String page = "";
151    page += "<head> ";
152    page += "<title>Automa&ccedil;&atilde;o Residencial</title>";
153    page += "</head>";
154    page += "<body>";
155    page += "<h1>Controle de l&acirc;mpadas</h1>";
156    page += "<div class='button-container'>";
157    page += "<a href='/ligar'><button class='button' id='ligar'>Ligar</button></a>";
158    page += "<a href='/desligar'><button class='button' id='desligar'>Desligar</button></a>";
159    page += "</div>";
160    page += "<div id='ativarSensorContainer'>";
161    page += "<a href='/ativarSensor'><button class='button' id='ativarSensor'>Ativar Sensor</button></a>";
162    page += "</div>";
163    page += "<div class='info'>";
164    page += "<p><font size='5'>Projeto de TCC</font></p>";
165    page += "<p><font size='5'>Alunos:</font></p>";
166    page += "<p><font size='5'>Rafael Sousa Chagas - 5149358</font></p>";
167    page += "<p><font size='5'>Ygor Felicio de Mattos Araujo - 5147980</font></p>";
168    page += "</div>";
169    page += "</body>";
170    page += "</html>";
171    server.send(200, "text/html", page);
172  }
```

Fonte: Acervo dos autores (2023)

A representação abaixo ilustra o resultado do código apresentado anteriormente, a página é iniciada com as opções de ligar ou desligar a lâmpada de forma manual, assim como de ativar o sensor de luminosidade.

Figura 36 – Página secundária

Fonte: Acervo dos autores (2023)

A seguinte função possui o objetivo de ligar a lâmpada. Primeiro é realizada uma verificação sobre o nível lógico do sensor, se está habilitado ou desabilitado. Caso o sensor esteja habilitado, ou seja, possui valor *true*, a função *handleLDR*() é chamada. Se o sensor estiver desabilitado, o estado do relé muda para *LOW*, ativando a iluminação do sistema, logo, a variável *relayState* recebe o valor *true*, informando que o relé está ativo e então envia "1" (nível lógico alto) para que seja feito o acionamento no botão de ligar a lâmpada no Blynk e a página principal seja atualizada.

Figura 37 – Função para ligar a lâmpada pelo aplicativo

```
200 void handleLigar() {
201   if (sensorEnabled == false) {
202     digitalWrite(relayPin, LOW);
203     relayState = true;
204     Blynk.virtualWrite(V0, 1);
205     handleMain();
206   } else {
207     handleLDR();
208   }
209 }
```

Fonte: Acervo dos autores (2023)

O próximo bloco de código refere-se à função de desligar a lâmpada. É realizada a mesma verificação vista no código acima. A diferença ocorre no estado do relé, que transita para *HIGH*, logo, a variável *relayState* recebe o valor *false*, informando que o relé está desativado e então envia "0" (nível lógico baixo) para que o botão de desligar a lâmpada seja desacionado no Blynk e a página principal seja atualizada.

Figura 38 – Função para desligar a lâmpada pelo aplicativo

```
211 void handleDesligar() {
212   if (sensorEnabled == false) {
213     digitalWrite(relayPin, HIGH);
214     relayState = false;
215     Blynk.virtualWrite(V0, 0);
216     handleMain();
217   } else {
218     handleLDR();
219   }
220 }
```

Fonte: Acervo dos autores (2023)

A parte sequente do código, aborda a função responsável pela ativação do sensor de luminosidade *LDR*. É verificado previamente se o sensor já está ativo, caso não, o estado do relé passa a ser *HIGH*, sendo assim, é atribuído valor *false* para a variável *relayState*. A variável de controle do sensor é definida como *true*, para sinalizar que o sensor foi ativado e então envia "1" (nível lógico alto) para que o botão de ativar o sensor seja acionado no Blynk e a página principal seja atualizada, para que seja feito o bloqueio dos botões de ligar ou desligar a lâmpada.

Figura 39 – Função para ativar o sensor pelo aplicativo

```
222  void handleAtivarSensor() {
223    if (sensorEnabled == false) {
224      digitalWrite(relayPin, HIGH);
225      relayState = false;
226      sensorEnabled = true;
227      Blynk.virtualWrite(V1, 1);
228    }
229    handleLDR();
230  }
```

Fonte: Acervo dos autores (2023)

Na seção seguinte do código, encontra-se a função que possui a finalidade de desativar o sensor de luminosidade. É certificado previamente se o sensor já está ativo, se não, o estado do relé transita para *HIGH*, sendo assim, é atribuído valor *false* para a variável *relayState*. A variável de controle do sensor é definida como *false*, para sinalizar que o sensor foi desativado e então envia "0" (nível lógico baixo) para que o botão de ativar o sensor seja desacionado no Blynk e a página principal seja atualizada, e os botões de ligar ou desligar a lâmpada sejam liberados para uso.

Figura 40 – Função para desativar o sensor pelo aplicativo

```
233  void handleDesativarSensor() {
234    if (sensorEnabled) {
235      digitalWrite(relayPin, HIGH);
236      relayState = false;
237      sensorEnabled = false;
238      Blynk.virtualWrite(V0, 0);
239      Blynk.virtualWrite(V1, 0);
240    }
241    handleMain();
242  }
```

Fonte: Acervo dos autores (2023)

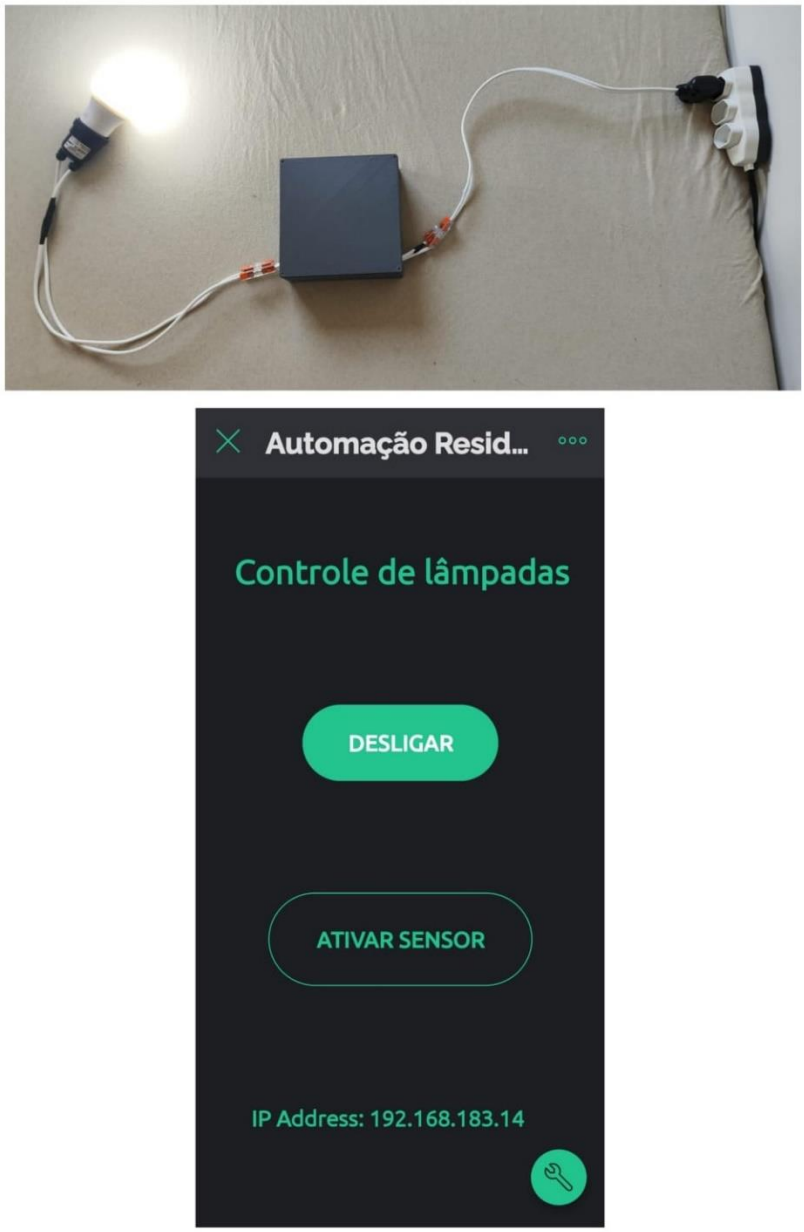
Após a conclusão e compilação do código, o microcontrolador efetua a execução. A representação abaixo demonstra o sistema de iluminação com o sensor de luminosidade ativado, visto que há luz natural suficiente no ambiente, a lâmpada permanece desligada.

Figura 41 – Sistema com o sensor ativado

Fonte: Acervo dos autores (2023)

A ilustração na sequência demonstra o sistema operando no modo manual, portanto, o sensor está desativado e o botão ligar está acionado, sendo assim, a lâmpada é acesa.

Figura 42 – Sistema operando no modo manual



Fonte: Acervo dos autores (2023)

Por fim, a programação foi implementada conforme as expectativas, resultando em um sistema de iluminação inteligente, confiável e robusto. O desempenho da execução dos métodos comprovou a rápida conexão à rede *wireless* e a resposta instantânea aos comandos do sistema.

4 RESULTADOS E DISCUSSÃO

A crescente preocupação com a eficiência energética e a busca por soluções sustentáveis têm impulsionado o desenvolvimento de tecnologias inovadoras em diversas áreas, incluindo o setor de iluminação. A iluminação inteligente, caracterizada pela integração de sensores e controles automatizados, visa otimizar o consumo de energia elétrica, proporcionando uma iluminação adequada de forma mais eficiente.

A eficácia do sistema de iluminação inteligente varia em cenários específicos, considerando variáveis como o tipo de ambiente, a demanda de iluminação e a resposta do sistema às mudanças de condições. Além dos aspectos econômicos e a viabilidade de implementação em diferentes contextos.

As lâmpadas são classificadas por sua potência, medida em watts (W), que representa a taxa de consumo de energia. Por exemplo, a lâmpada utilizada no projeto possui 9 watts, ou seja, a cada hora ligada, consome 9 watts. Se essa lâmpada ficar acesa por 1 hora por dia durante 30 dias, o consumo total será de 270 watts-horas (Wh). Para converter isso em quilowatt-hora (kWh), dividimos por 1000, resultando em 0,27 kWh. Compreender a potência e o tempo de uso é essencial para monitorar e gerenciar o consumo de energia de dispositivos elétricos em casa.

O custo de um quilowatt está em torno de R$ 0.95, sendo assim, uma hora a menos que a lâmpada fique acesa, a economia é de R$ 0.26 por lâmpada.

Imagine uma casa com múltiplos cômodos, cada um equipado com diversas lâmpadas. Se cada uma delas for utilizada por apenas uma hora a menos por dia, a economia acumulada ao longo do tempo pode resultar em uma redução notável na conta de energia.

A integração do sensor de luminosidade *LDR* desempenha um papel crucial na busca por uma economia de energia significativa. O objetivo desse sensor é otimizar o uso da iluminação, portanto, evitar que a lâmpada permaneça acesa de forma desnecessária, enquanto ainda há luz natural suficiente no ambiente e assim gerar uma economia.

A redução do consumo de energia em residências não apenas proporciona economias financeiras, mas também contribui para práticas sustentáveis, diminuindo a emissão de carbono do lar.

5 CONCLUSÃO

Este projeto teve como meta o desenvolvimento de um sistema de iluminação inteligente, priorizando o mínimo de intervenção humana e uma economia de energia significativa. A capacidade de controlar o sistema de iluminação por meio de dispositivos móveis, como smartphones ou tablets, adicionou uma camada adicional de conveniência, possibilitando aos usuários ajustarem a iluminação conforme suas necessidades e preferências, independentemente da localização.

Ao alcançar um equilíbrio entre automação inteligente e sustentabilidade, este projeto representa um passo significativo em direção a automação residencial. A minimização da intervenção humana, aliada à adaptação dinâmica às condições ambientais, destaca a capacidade do sistema em responder de maneira autônoma aos desafios do dia a dia.

Este trabalho foi concluído com êxito, cumprindo todos os requisitos e objetivos anteriormente estabelecidos, resultando em um sistema adaptável as variadas condições de iluminação natural. As avaliações em diferentes cenários contribuíram para o objetivo de realizar o controle residencial de maneira inteligente de acordo com as necessidades do usuário.

Para adicionar melhorias ao sistema, pode-se programar para ser possível ver a porcentagem de luz solar incidente nas lâmpadas e adicionar opções para o uso de configurações diferentes do padrão, como a opção de selecionar o horário de desligamento automático.

Por fim, o sistema inteligente desenvolvido, não apenas simplifica a gestão cotidiana, mas também contribui para a preservação de recursos energéticos, sendo uma solução prática e sustentável para o controle residencial de forma autônoma.

6 REFERÊNCIAS

ARDUINO. **Arduino IDE**. Disponível em: https://www.arduino.cc/en/software. Acesso em: 10 agosto 2023.

BIEGELMEYER, Anderson. **Desenvolvimento e aplicação de uma casa inteligente.** 2015. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) – Universidade de Caxias do Sul, Caxias do Sul, 2015.

BLYNK. **Blynk: a low-code IoT software platform**, c2023. Página inicial. Disponível em: https://blynk.io/. Acesso em: 27 novembro 2023.

CAMARGO, André Roberto de. **Correlação dos protocolos de comunicação principais da internet e modelo de rede OSI**. 2020. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) – Universidade Note do Paraná, Londrina, 2021.

CARVALHO, Cristiana. **Internet das coisas: entenda o que é e como funciona.** Disponível em: https://www.tecmundo.com.br/internet/230884-internet-coisas-entenda-funciona.htm. Acesso em: 12 novembro 2022.

FERNANDES, Fabio. **Sistema de Iluminação Inteligente através de Redes de Sensores Wireless.** 2011. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) – Universidade Estadual Paulista, Guaratinguetá, 2011.

INSTITUTO GLOBAL MCKINSEY. **Unlocking the potential of the Internet of Things.** Disponível em: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world. Acesso em: 18 novembro 2022.

JUNIOR, Sergio Luiz Stevan; FARINELLI, Felipe Adalberto. **Domótica: Automação residencial e casas inteligentes com Arduino e ESP8266.** Ponta Grossa, 2018.

PLATT, Charles. **Eletrônica Fácil**. São Paulo: Novatec, 2018.

RANGER, Steve. **What is the IoT? Everything you need to know about the Internet of Things right now.** Disponível em: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now. Acesso em: 15 novembro 2022.

SILVA, Ronald Gamba da. **Entregas de Penny: Desenvolvimento de um serious game para aprendizagem sobre redes de computadores.** 2019. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Universidade de Caxias do Sul, Caxias do Sul, 2019.

SILVA, Antonio Vinicius Ferreira e. **Uma análise comparativa das versões do protocolo HTTP: Evolução e pontos que ampliem o uso do HTTP/3**. 2021. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Centro Universitário Christus, Fortaleza, 2021.

7 AGRADECIMENTOS

Expressamos profunda gratidão ao dedicado orientador, Júlio Almeida Borges, cuja orientação foi vital. A todos que de maneira direta ou indireta contribuíram para que este trabalho fosse possível.

Agradecemos a todos os envolvidos!