CENTRO UNIVERSITÁRIO DE BRASÍLIA – UNICEUB FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS - FATECS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

EDILSON LEAL SANTOS

AUTOMAÇÃO RESIDENCIAL PARA MONITORAMENTO DE TEMPERATURA, UMIDADE E CONTROLE DE ILUMINAÇÃO UTILIZANDO ESP8266

Orientador: Prof. MSc. Luciano Henrique Duque

Brasília - DF

2016

EDILSON LEAL SANTOS

AUTOMAÇÃO RESIDENCIAL PARA MONITORAMENTO DE TEMPERATURA, UMIDADE E CONTROLE DE ILUMINAÇÃO UTILIZANDO ESP8266

Trabalho de conclusão de Curso apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito para obtenção do título de Engenheiro da Computação.

Orientador: Prof. MSc. Luciano Henrique Duque

Brasília - DF

2016

EDILSON LEAL SANTOS

AUTOMAÇÃO RESIDENCIAL PARA MONITORAMENTO DE TEMPERATURA, UMIDADE E CONTROLE DE ILUMINAÇÃO UTILIZANDO ESP8266

Trabalho de Conclusão de Curso apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito para obtenção do título de Engenheiro da Computação.

Orientador: Porf. MSc. Luciano Henrique Duque

BANCA EXAMINADORA

Prof. MSc. Luciano Henrique Duque

Orientador

Prof. Layany Zambrano Horta Damazio

Examinadora

Brasília - DF

2016

AGRADECIMENTOS

Muitos obstáculos enfrentei até chegar aqui, a família é essencial em momento felizes e momentos difíceis.

Agradeço aos meus amigos e familiares por todos os momentos felizes e tristes que passamos.

Agradeço aos meus pais Edite Leal Santos e Hélio Andrade Santos (em memória) por todos os conselhos, conversas, carinho, dedicação, por me fazer acreditar que o estudo é o melhor caminho, por me conduzir no caminho do "bem", apoiando em minhas decisões e por me ensinar que o homem sem Deus não é nada.

Agradeço aos meus filhos Luan Silva, Cauã Silva e Lucca Silva por ter me mostrado o quanto é gostoso ser pai. Muito obrigado por todos os momentos felizes que passamos juntos, por todas as coisas que aprendi com vocês, por serem crianças tão obedientes, carinhosas e traquinas, por compreenderem as vezes que me ausentei por causa dos estudos. Amo vocês porcarias.

Agradeço a minha amada irmã Elaine Andrade Leal por cada conselho, por acreditar em min, por orar por mim e minha família, por escutar meus apelos e desabafos, por ser a melhor irmã do mundo. Muito obrigado minha irmã, assim como tenho muito orgulho de você, espero que tenha orgulho de mim. Obrigado pelas sobrinhas maravilhosas que você e meu cunhado que tenho muito apreço Fabio Barreto, me concederam a dádiva de ser tio.

Dedicação em especial à minha amiga, esposa e companheira Ana Silvia, num casamento longo e duradouro que chega aos 13 anos de confiança, carinho, dedicação, paciência, compreensão, e muito amor. A ti amor da minha vida, dedico esta monografia, e tudo que fiz foi numa mera tentativa de te impressionar e fazer com que você tenha orgulho de mim. Obrigado por ter me dado a oportunidade de te conhecer e por deixar que eu faça parte de sua vida. Tenho a plena convicção que é com você que quero passar todos os dias de minha vida. Te amo minha Porcaria <3.

CITAÇÃO

"Você tem que encontrar o que você gosta. E isso é verdade tanto para o seu trabalho quanto para seus companheiros. Seu trabalho vai ocupar uma grande parte da sua vida, e a única maneira de estar verdadeiramente satisfeito é fazendo aquilo que você acredita ser um ótimo trabalho. E a única maneira de fazer um ótimo trabalho é fazendo o que você ama fazer. Se você ainda não encontrou, continue procurando. Não se contente. Assim como com as coisas do coração, você saberá quando encontrar. E, como qualquer ótimo relacionamento, fica melhor e melhor com o passar dos anos. Então continue procurando e você vai encontrar. "

- Steve Jobs, trecho de discurso em Stanford,2005.

AUTOMAÇÃO RESIDENCIAL PARA MONITORAMENTO DE TEMPERATURA, UMIDADE E CONTROLE DE ILUMINAÇÃO UTILIZANDO ESP8266

RESUMO

Este trabalho tem como objetivo desenvolver uma automação residencial que permita controlar remotamente diversos dispositivos encontrados na maioria das residências, permitindo assim o acionamento de lâmpadas, tomadas elétricas, fitas de LED's e ainda monitorar a umidade e temperatura do ambiente. Através da rede sem fio, será gerado um modelo funcional visando maior comodidade e acessibilidade aos dispositivos mencionados. Utilizando o microcontrolador ESP8266 que possui um processador Risc de 32 bits, em conjunto com outros componentes eletrônicos, será possível controlar remotamente os dispositivos da casa. O ESP8266 será conectado à rede sem fio da casa, assim, permitido o acionamento e monitoramento dos componentes em um determinado ambiente.

Palavras-Chave: Tecnologia, Automação Residencial, Iluminação, ESP8266.

ABSTRACT

This work aims to develop a residential automation that allows to remotely control various devices found in most houses, thus allowing the activation of lamps, electrical outlets, LED strips and also monitor the environment's humidity and temperature. Through the wireless network, it will generate a functional model seeking greater convenience and accessibility to the mentioned devices. Using the micro controller ESP8266 which has a 32 bits Risc processor, alongside other electronic components, it will be possible to remotely control the house devices. The ESP8266 will be connected to the house's wireless network, therefore allowing the activation and monitoring of the aforementioned components.

Keywords: Technology, Residential automation, Lighting, ESP8266.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAS DE SIGLAS

A	Ampere
AC	Alternating Current
AP	Access Point
API	Application Programming Interface
DC	Direct Current
GHz	Giga-Hertz
GND	Graduated Neutral Density Filter
GPIO	General-purpose input/output
HD	Hard Disk
IDE	Integrated Development Environment
IoT	Internet of Things
I²C	Inter-Integrated Circuit
LAN	Local Area Network
MHz	Mega-Hertz
NBR 5410	Norma Brasileira Regulamentadora 5410
PC	Computador Pessoal
PWM	Pulse Width Modulation
RF	Radio Frequency
SoC	System on Chip
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
V	Volts
VCC	Collector Supply Voltage
W	Watts

CAPÍTULO 1 INTRODUÇÃO

Nos dias atuais, a internet é comum em nossas vidas. Já estamos bastante acostumados com ela em nossos Tablets, Smartphones e Computadores. Mais nem sempre foi assim. A internet nasceu da Arpanet como uma rede experimental financiada pelos militares dos Estados Unidos da América (EUA), interligando grandes universidades e centros de pesquisas no ano 1969. Ela interligava grandes computadores permitindo que pesquisadores compartilhassem capacidade de processamento entre seus equipamentos. Em 1970 a Arpanet permitiu o acesso a universidades fora dos EUA, criando assim a base da Rede Mundial de Computadores. (CANARIM, 2012)

Na década de 1980, a Internet começou a criar forma, como a conhecemos hoje. Somente na década de 1990, foi aberta comercialmente ao público, assim nascendo a Rede Mundial de Computadores. Inicialmente os primeiros sites eram formados somente por textos e o mínimo de funcionalidades. Só em 1995 surge o conceito de *home-page*, uma página com botões, desenhos 3D e janelas, servindo de menu para ligar todo conteúdo do site. O foco agora era a aparência do site. (MARTINS, 2012)

A história da internet, pode ser resumida nas seguintes fases: Primeiro veio a rede de computadores evoluindo para a rede de pessoas e comunidades como por exemplo o Orkut, Facebook, LinkedIn, Blogs entre outros, e atualmente estamos na fase da Internet das Coisas (*IoT – Inernet of Things*), que passa a interligar vário tipos de objetos e dispositivos inteligentes que vão interagir entre si e conosco, tornando nosso dia a dia mais fácil. (MARTINS, 2012)

O empresário Kevin Ashton, na década de 1990, foi o primeiro a utilizar o termo "*Internet of Things*", quando fazia parte de uma equipe do Auto-ID Center do MIT, que descobriu uma maneira de conectar objetos à Internet via uma etiqueta RFID. (SAS, 2013)

Com a chegada dos computadores pessoais e a internet, as chamadas "casas inteligentes", que tem como base para seu desenvolvimento a Internet das Coisas, começaram a evoluir de forma positiva. Pesquisas realizadas nos Estados Unidos revelam que 84% das construtoras se preocupam em entregar residências com algum tipo de tecnologia. Assim atendendo uma crescente demanda por parte do consumidor do mercado imobiliário, relacionado a tecnologia em suas residências. Sistemas automatizados estão cada vez mais solicitados por este público. (VOLTIMUM, 2014)

Automação residencial é o conjunto de serviços com intuito de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação por meios de serviços e sistemas tecnológicos integrados. No Brasil, o mercado de automação residencial, aos poucos, está evoluindo equiparado com os mercados mais evoluídos. Novos profissionais estão sendo formados para atender esta crescente demanda em função das diferentes tecnologias e suas complexidades, instalação e programação (VOLTIMUM, 2014).

Ao longo de décadas estamos sendo inundados com aparelhos diversos, mas somente nos últimos anos que conseguimos enxergar o verdadeiro potencial da Internet das Coisas. Com o avanço da tecnologia, os sensores foram popularizados e as pessoas passaram a compreender o uso desta tecnologia, tanto para uso pessoal quanto profissional (SAS, 2013).

Temos como exemplo de funcionalidades da Internet das Coisas, as seguintes situações: imagine que durante o trajeto de volta para casa, você tenha a possibilidade de controlar a temperatura do ambiente para um melhor conforto, verificar as câmeras de segurança, ou simplesmente acionar a cafeteira entre outros equipamentos, tudo isso de forma remota. Outra possibilidade é a de reduzir o consumo de combustível através de um dispositivo que controle o fluxo do tráfego manipulando os semáforos evitando congestionamentos (SAS, 2013).

Há muitos anos a automação é uma realidade nas industrias, com intuito de automatizar atividades como controle de auto fornos em siderúrgicas, que é uma atividade inadequada ao ser humano, atividades rotineiras que podem ser executadas por máquinas com o intuito de redução de custos, entre outros. A cada dia que passa, o custo com automação residencial vem diminuindo, tornando mais viável investir neste ramo.

O microcontrolador proposto é o ESP8266 fabricado pela empresa Espressif Systems, permitindo assim, conectar tais dispositivos a rede sem fio padrão IEEE 802.11., como por exemplo, os relés que conectado a este microcontrolador, permite o acionamento de lâmpadas e tomadas elétricas via uma página *HTML*.

Este trabalho traz um conceito de automação residencial visando o bem-estar e conforto de uma residência. Utilizando o microcontrolador ESP8266, será possível controlar remotamente o acionamento de lâmpadas, tomadas e fitas de *LED,* via *WEB Browser* e ainda monitorar a temperatura e umidade do ambiente. Será utilizado um microcontrolador para cada coisa a ser controlado. Para o acionamento de lâmpadas será utilizado um sensor de iluminação

LDR (Light Dependent Resistor) que irá indicar se a lâmpada está acesa ou apagada já que o controle da lâmpada poderá ser feito também pelo interruptor de parede conectado a um módulo relé por ligação *three-way*. O acionamento da tomada será utilizando um modulo relé. Já a fita de *LED*, será mostrado no WEB Browser, um leque de cores para que o usuário selecione a cor desejada para a fita de *LED*, também será disponibilizado, botões para efeitos randômicos para tal fita. Por fim, via WEB Browser o usuário poderá monitorar a temperatura e a umidade de um determinado ambiente.

Portanto, da mesma forma que ocorreu uma revolução na vida das pessoas com o surgimento dos S*martphones, Tablets*, a Internet das Coisas já é uma nova revolução, fazendo com que as casas inteligentes se tornem indispensáveis aos padrões de qualidade de vida atuais.

1.1. OBJETIVOS DO TRABALHO

1.1.1. OBJETIVO GERAL

O objetivo geral deste projeto é desenvolver uma automação residencial que permita via rede sem fio, o controle remoto de diversos dispositivos residenciais. Sendo estes, o monitoramento de umidade e temperatura do ambiente, acionamento de tomadas de uso geral, controle de iluminação de ambientes através de lâmpadas e fitas de Leds.

1.1.2. OBJETIVOS ESPECÍFICOS

Para atender o objetivo geral descrito acima, os seguintes objetivos específicos serão atendidos individualmente:

- ✓ *Desenvolver o código, em linguagem C/C++ que permita controlar o microcontrolador ESP8266 onde será hospedado a página WEB com o auxílio de um módulo SD CARD especifico;*

- ✓ *Elaborar o código fonte da página em HTML, CSS e JavaScript, possibilitando a interação do usuário com o microcontrolador ESP8266 via rede sem fio, assim, permitindo o acionamento e monitoramento das coisas a través de um WEB Browser;*
- ✓ *Projetar estrutura elétrica e componentes;*
- ✓ *Criar o circuito unificando os diferentes sistemas de uma instalação elétrica residencial, utilizando componentes eletrônicos diversos;*
- ✓ *Efetuar testes de comunicação entre software e hardware;*
- ✓ *Projetar maquete didática para cada um dos dispositivos com intuito de apresentação a banca;*

1.2. METODOLOGIA

O esquemático abaixo demonstra resumidamente a interligação do microcontrolador *ESP8266* aos diversos dispositivos a serem controlados, que por sua vez, realiza o acionamento de diferentes sistemas de uma residência, sendo controlado por qualquer dispositivo que consiga executar um código HTML via WEB Browser, desde que este dispositivo esteja conectado a mesma à rede sem fio dos demais dispositivos.

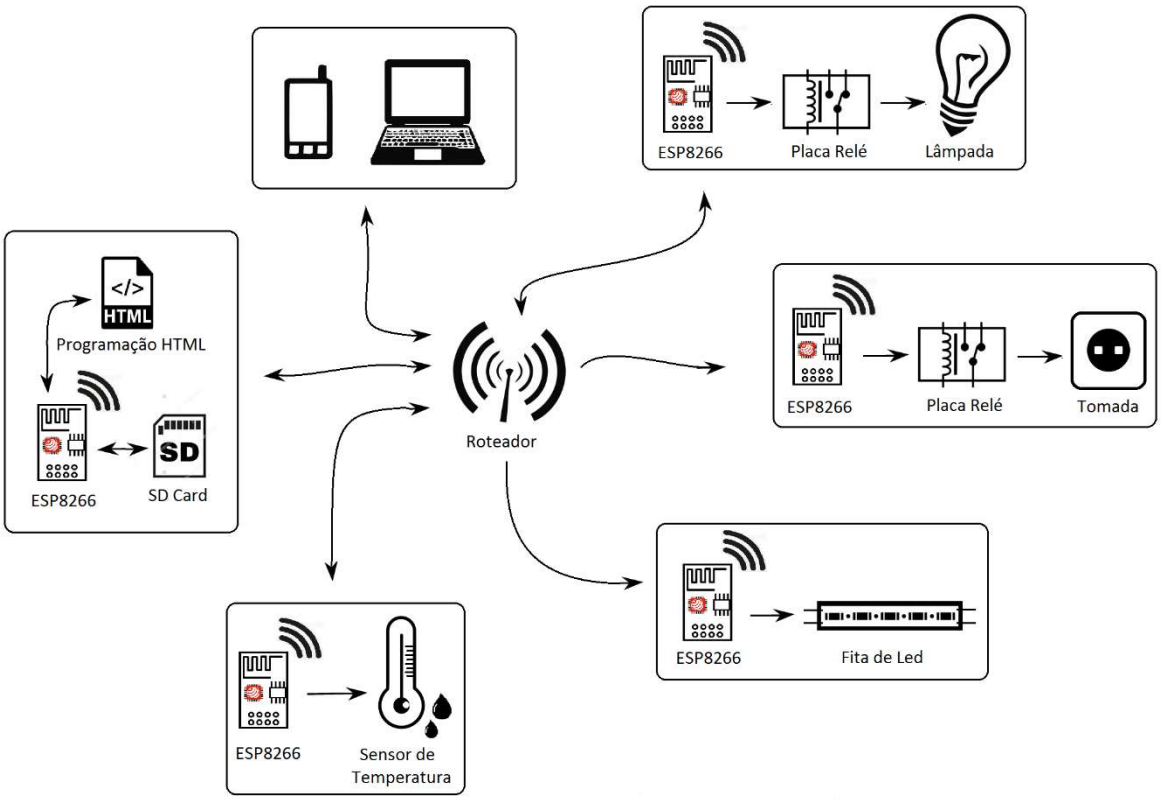

Figura 1.1 - Metodologia Fonte: Autor

Primeira etapa: Realização de pesquisas bibliográficas acerca do microcontrolador *ESP8266,* rede *Wi-Fi* baseado no padrão IEEE 802.11, funcionamento do módulo relé, sensor de umidade e temperatura DHT11, fita de *LED* RGB endereçável, módulo para cartão SD, linguagem de programação C/C++, HTML, CSS e JavaScript.

Segunda etapa: Estudo e desenvolvimento da linguagem de programação C/C++ com intuito de acionar via microcontrolador ESP8266, módulo relé através das portas digitais (GPIO´s), que controlará o acionamento de lâmpadas e tomadas.

Terceira etapa: Utilizando o microcontrolador e a linguagem de programação citado na segunda etapa, será possível o controle de uma fita de *LED* RGB endereçável numa gama de cores disponíveis e ainda alguns efeitos randômicos.

Quarta etapa: Ainda com o microcontrolador ESP8266 será desenvolvido um algoritmo na linguagem de programação C/C++, para realizar medições de temperatura e umidade do ambiente, fornecidos por um sensor especifico.

Quinta etapa: Estudo e desenvolvimento da linguagem de programação HTML para interação do usuário com os dispositivos a serem controlados/monitorados via Web Browser conectado a uma rede sem fio através de um computador, Smartphone ou Tablet.

Sexta etapa: Será realizado a integração do software com o hardware com intuito de controlar/monitorar os diversos recursos citados nas etapas anteriores.

Sétima etapa: Será feito uma bateria de testes em cada uma das funcionalidades previstas no projeto em diferentes sistemas operacionais, (Windows, Android, IOS), relatando em quais sistemas teremos sucesso de utilização destas.

Oitava etapa: Confecção e montagem dos painéis com todos os dispositivos para a demonstração das funcionalidades deste projeto.

1.3.MOTIVAÇÃO

Ouvimos muito falar sobre Internet das Coisas (IoT). Na maioria dos sites, revistas e programas televisivos que falam sobre tecnologia, estão citando que IoT já é a tecnologia do futuro. A cada dia novos objetos estão conectados à internet visando a comodidade das pessoas eliminando a necessidade de intervenção humana em diversos aspectos.

Esta tecnologia é apresentada como uma solução em potencial para melhorar a vida das pessoas. Cada vez mais, Grandes empresas como a Symantec e a IBM estão investindo bilhões de dólares na contratação de funcionários para a área de pesquisa e desenvolvimento de IoT. (ARAUJO)

Em poucos anos, o profissional de tecnologia que não dominar a Internet das Coisa, estará muito defasado para o mercado de trabalho.

Com o intuito de maior conforto, conectividade e conveniência, este trabalho acadêmico traz o desenvolvimento de uma automação residencial, permitindo o controle de remoto de diferentes dispositivos de um determinado ambiente, interligando o microcontrolador ESP8266 com placa relé dentre outros dispositivos, assim, através de uma página Web via HTML controlar/monitorar remotamente por meio da rede sem fio.

1.4.RESULTADOS ESPERADOS

O resultado final do projeto é a implementação do sistema de hardware e software facilitando o controle das coisas com uma interface de fácil usabilidade e implementação, já que cada dispositivo a ser controlada, será por meio da rede sem fio, assim, tendo o menor impacto na instalação elétrica de uma residência.

É esperado que o software tenha uma integração das funcionalidades em um único lugar, que será apresentado ao usuário via aplicação Web.

Após a implementação do que está sendo proposto neste trabalho acadêmico, o usuário poderá controlar lâmpadas, tomadas, intensidade e cor em uma fita de LED e ainda monitorar a temperatura e a umidade de um determinado ambiente.

1.5.TRABALHOS CORRELATOS

A seguir são apresentados alguns trabalhos com objetivo e características semelhantes deste trabalho.

1.5.1. SISTEMA CONTROLADOR DE ILUMINAÇÃO DE AMBIENTES ATRAVÉS DE INTERFACE COMPUTADORIZADA.

Neste trabalho, o objetivo foi desenvolver um sistema para o controle do acionamento de lâmpadas de forma remota, por meio do microcontrolador Arduino Duemilanove, conectado a um computador pessoal trocando informações pela serial USB, por meio de uma página web que por sua vez está hospedada em um computador pessoal. O circuito conta com sensores de luminosidade, sensores de presença e componentes eletrônicos que são conectados ao microcontrolador, controlados por uma página WEB. Dentre as funcionalidades, está previsto o acionamento de lâmpadas em tempo real ou que permita ao usuário configurar horários para tal acionamento e ainda identificar se as lâmpadas estão acesas ou apagadas (ALMEIDA, 2010)

1.5.2. AUTOMAÇÃO RESIDENCIAL PARA CONTROLE DE ILUMINAÇÃO, SEGURANÇA E MONITORAMENTO USANDO O ARDUINO MEGA.

O Lucas propõe neste TCC (SOMBRA, 2016), a criação de uma automação residencial que unifique o sistema elétrico, alarme e monitoramento de temperatura através da rede sem fio, utilizando uma página HTML por meio de uma Ethernet Shield conectada a um microcontrolador Arduino Mega. A ideia é centralizar todo acionamento em uma central através do microcontrolador conectado a *Ethernet Shield*, que por sua vez está conectado a um roteador sem fio, assim possibilitando que os computadores, *Smartsphones* e T*ablets,* controlem os diversos dispositivos propostos como acionamento de lâmpadas, fechadura eletrônica, sirenes de um alarme, entre outros.

1.5.3. COMPARATIVOS ENTRE TRABALHOS CORRELATOS.

Os dois trabalhos preveem acionamento de lâmpadas por meio de uma página WEB, porém, o que está sendo proposto neste trabalho é uma automação residencial via rede sem fio em cada dispositivo a ser controlado. Cada dispositivo a ser controlado ou monitorado, receberá um endereço IP graças ao microcontrolador ESP8266, disponibilizado por um roteador residencial.

Outra vantagem é que não será necessário de um computador pessoal ou um módulo *Ethernet Shield* para rodar o serviço de *WEB Server*, já que este microcontrolador ESP8266, é possível disponibilizar tal serviço.

Finalizando, como cada lâmpada ou cada dispositivo a ser controlado/monitorado terá um microcontrolador ESP8266, assim minimizando o impacto da instalação elétrica. Não é necessária a maximização da fiação, tendo em vista que tal microcontrolador será controlado individual por meio da rede sem fio através de um microcontrolador central.

1.6. ESTRUTURA DO TRABALHO

A estrutura desta monografia consiste de 5 capítulos que tratam os assuntos descritos abaixo:

CAPÍTULO 1 – Introdução, objetivos, metodologia, motivação para a escolha do tema, resultados esperados, Trabalhos correlatos.

CAPÍTULO 2 – Referencial teórico dos sensores, microcontrolador ESP8266, LED RGB, entre outros componentes eletrônicos.

CAPÍTULO 3 – Desenvolvimento do trabalho proposto.

CAPÍTULO 4 - Testes e resultados em cada um dos servidores.

CAPÍTULO 5 – Conclusão e trabalhos futuros.

CAPÍTULO 2 REFERENCIAL TEÓRICO

Este capítulo é destinado a explanação dos assuntos que vão subsidiar o desenvolvimento do sistema de automação proposto.

2.1.AUTOMAÇÃO RESIDENCIAL

O conceito de automação residencial é definido de diversas formas, mas sempre com a premissa de possibilitar ao usuário usufruir o máximo de qualidade de vida, dentro da habitação. Muratori, por exemplo, define automação residencial como sendo o conjunto de serviços com intuito de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação por meios de serviços e sistemas tecnológicos integrados (MURATORI e DAL BÓ, 2014). Já a Associação Espanhola de Domótica define automação residencial como sendo, a instalação de tecnologia em residências, através do uso de equipamentos que se comunicam interativamente seguindo instruções de um programa estabelecido pelo usuário, com a possibilidade de alterações de acordo com seus interesses e necessidades, tendo como objetivo, melhorar a qualidade de vida, aumentar a segurança e viabilizar o uso racional dos recursos para seus habitantes (ASSOCIAÇÃO ESPLANHOLA DE DOMÓTICA. 2015).

De acordo com os conceitos acima, a automação residencial tem a função de auxiliar as pessoas nas tarefas realizadas no cotidiano, sendo impulsionada por argumentos como conforto, segurança e até mesmo sustentabilidade. A automação residencial é possível a partir da integração de dispositivos e serviços de forma centralizada e programável, permitindo o controle e supervisão de várias tarefas de forma automatizada (MURATORI e DAL BÓ, 2014).

O sistema de automação dentro de uma residência pode abranger a instalação elétrica, controlando por exemplo, a iluminação de um ambiente, abertura de persianas e cortinas, acionamento de tomadas, monitoramento de itens de segurança como alarmes contra intrusos, controle de acesso, sistemas de vigilância, entre outros. Este tipo de automação pode variar de acordo com a real necessidade do usuário.

Este trabalho consiste no desenvolvimento de uma automação residencial que visa o bem-estar e conforto do usuário, utilizando o microcontrolador ESP8266, que consiste em controlar remotamente através da rede sem fio, o acionamento de lâmpadas, o controle de tomadas e de iluminação e ainda monitorar a temperatura e umidade o ambiente.

De acordo com Muratori, e de forma análoga ao que já acontece com os veículos automotores atualmente, em breve, os dispositivos de controle dentro de uma residência, serão acessórios que passarão a ser itens de série (MURATORI e DAL BÓ, 2014).

2.2.REDE SEM FIO

Em 1892, Guglielmo Marconi, demostrou a capacidade de se comunicar com pessoas em movimento através do rádio, fornecendo contato continuo com navios navegando pelo canal inglês. A partir de tal feito, novos métodos e serviços de comunicação sem fio tem sido estudados, com intuito de sempre buscar a melhor forma de transmissão de radiofrequência (RF), bem como sua miniaturização, que tornaram os equipamentos portáteis ainda menores, mais baratos e mais confiáveis (RAPPAPORT, 2009).

Rede sem fio nasceu da necessidade de dois ou mais dispositivos eletrônicos trocar informações sem uma ligação física entre eles. Na mesma época que surgiram os notebooks, empresas começaram a trabalhar para descobrir uma forma de conecta-los a internet através da rede sem fio. Então surgiu a ideia de equipar os ambientes e os notebooks com transmissores e receptores de rádio de ondas curtas para permitir a comunicação, nascendo assim a LANs sem fio por várias empresas (TANENBAUM e WETHERALL, 2011).

Como várias empresas começaram a comercializar LANs sem fio, surgiu o problema de incompatibilidade, os dispositivos equipados com um rádio da marca X, não funcionava com dispositivos da marca Y. Para resolver este problema, a indústria decidiu criar um padrão de LAN sem fio, assim o Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE), que já havia padronizado a LAN com fio recebeu a tarefa de elaborar um padrão de LANs sem fio. Assim, o padrão de LAN sem fio recebeu a denominação 802.11, popularmente conhecida como WiFi. Hoje as redes 802.11, equipa notebooks, telefones móveis, tablets entre outros (TANENBAUM e WETHERALL, 2011).

As LANS sem fio caracterizadas pelo padrão 802.11, se tornaram muito populares, sendo o padrão utilizado em escritórios, locais públicos como aeroportos e dentro das residências, sendo utilizadas para conectar dispositivos móveis como smartphones, laptops e outros a internet, além de permitir que dispositivos vizinhos sem acesso à internet possam se comunicar. Ainda temos problemas de interferência que variam até mesmo com pequenas mudanças no ambiente como reflexão de ondas de rádio em objetos sólidos, ocasionando lentidão e dificuldades de acesso (TANENBAUM e WETHERALL, 2011).

A Figura 2.1 mostra um exemplo de infraestrutura de rede LAN sem fio entre computadores em diversos ambiente e seus deslocamentos. Os pontos de acessos (*Access Point*s - *AP*), são responsáveis por enviar o sinal de rádio aos seus clientes, em um determinado range de alcance que pode variar de acordo com o ambiente e ou a tecnologia empregada. Os hospedeiros sem fio são os laptops que podem estar fixos em um ambiente ou em movimento alternando entre APs.

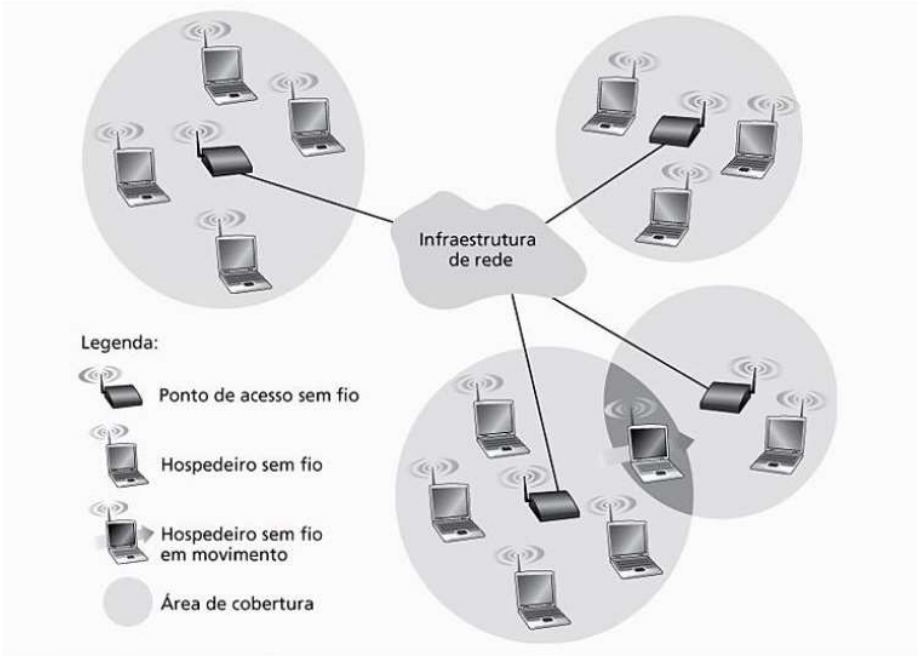

Figura 2.1 - Elementos de uma rede sem fio Autor: (KUROSE e ROSS, 2010)

Utilizando esta tecnologia Wifi, será possível uma conectividade entre o usuário e o microcontrolador ESP8266 com o intuito de controlar o sistema de automação proposto. Tal

controle é viável, pois o microcontrolador proposto possui esta tecnologia embutida, fornecendo assim protocolos de comunicação necessários para conectar-se remotamente.

2.3. MICROCONTROLADOR ESP8266

Microcontrolador é um circuito integrado que compõem em seu interior um processador para realizar cálculos, barramento para comunicação entre esse processador e seus periféricos (memoria, porta serial e porta paralela, conversor analógico/digital). É um componente que pode ser programado para execução de diversas funções específicas (KOLBAN, 2015), como exemplo dos acionamentos e controles especificados no tópico 2.1AUTOMAÇÃO RESIDENCIAL.

Atualmente no mercado existe uma grande variedade de microcontroladores, mas o que tem sido mais utilizado em inúmeros projetos é o Arduino, graças a sua arquitetura aberta e de fácil utilização. Apesar de muitos desses dispositivos terem capacidades de manipulação de *General-purpose input/output* (GPIO), *Inter-Integrated Circuit* (I^2C), *Serial Peripheral Interface* (SPI), *Universal asynchronous receiver/transmitter* (UART) entre outros, assim como o Arduino, não possuem acesso à internet sendo necessário trabalhar em conjunto com um hardware complementar (KOLBAN, 2015).

Neste contexto é apresentado o ESP8266, um microcontrolador fabricado pela empresa chinesa Espressif Systems. Este microcontrolador se destaca principalmente pelo seu tamanho reduzido, baixo custo e pouco consumo de energia, ideal para explorar a onda de Internet das Coisas (IoT). Tais propriedades vem de sua arquitetura, *System on Chip* (SoC), de baixo consumo de energia e alto desempenho *wireless* que está de acordo com o padrão IEEE802.11bgn (ESPRESSIF). A arquitetura SoC (traduzindo, sistema em um chip), é um conjunto de processadores, memórias e interconexões sob medida para um domínio de aplicação (CIPOLI).

O esquema elétrico da plataforma ESP8266X, seus principais componentes e funcionamento será apresentado a seguir.

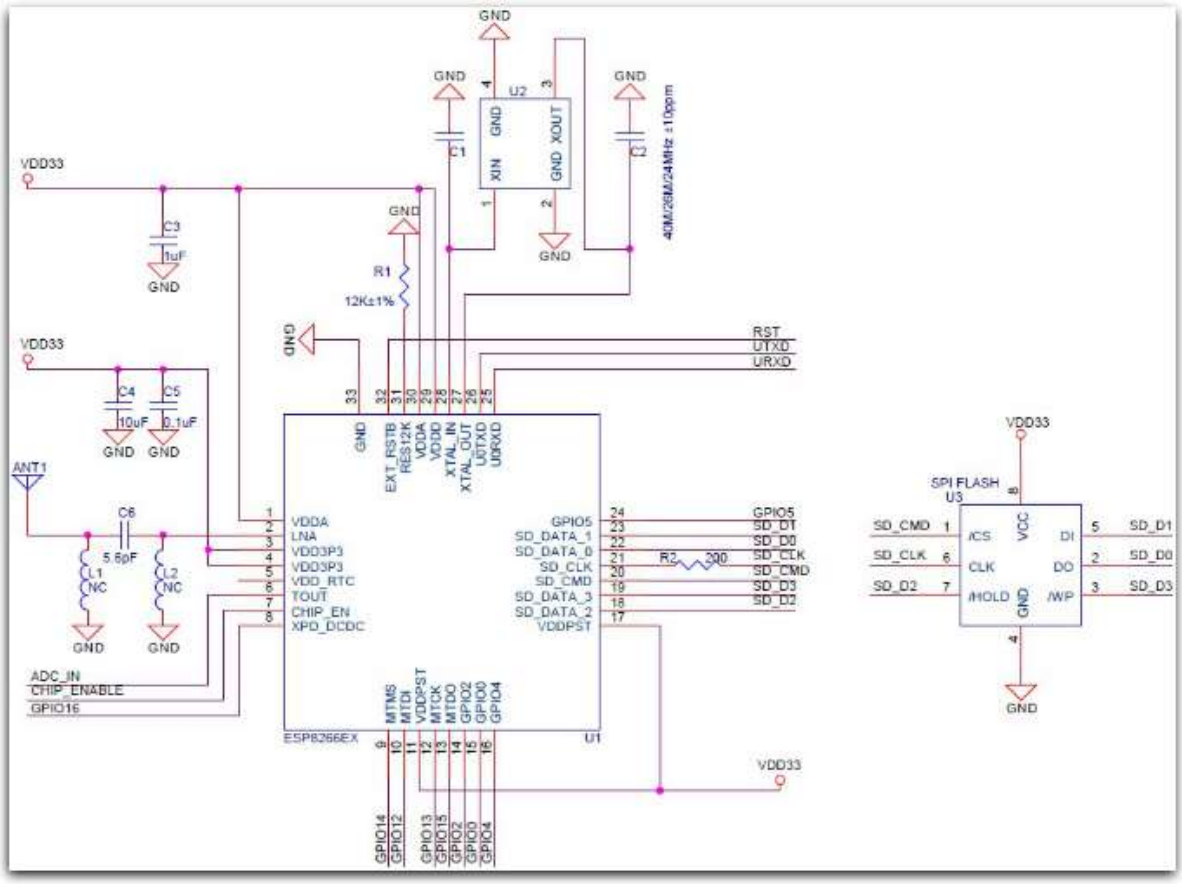

Figura 2.2 - Esquema elétrico módulo ESP8266 X. Fonte: https://espressif.com/en/products/hardware/esp-wroom-02/resources

Conforme ilustrado na Figura 2.2, o componente ESP8266EX corresponde ao microcontrolador principal, esse componente é basicamente um SoC, possui arquitetura Risc, processador interno Tensila L106 de 32 bits com velocidade de clock de 80MHz podendo chegar a 160MHz em *overclock*, memória RAM interna de aproximadamente 50KB, possui 17 GPIOs, sendo que destes, 6 GPIOs são de uso exclusivo do módulo ESP, restando 10 GPIOs para conexões de sinais digitais, e 1 GPIO para sinal analógico, conta também com WI-FI integrado padrão 802.11. Os 6 GPIOs de uso exclusivo e serve para que o módulo ESP acesse sua memória flash acoplada de 512KB, podendo ser substituída por uma memória de até 16MB.

Para um melhor entendimento das características do módulo ESP8266X, é apresentado a Tabela 2.1, que reúne as principais características deste microcontrolador.

Propriedades	Valor
Tensão	3,3V ~ 3,6V DC
Corrente de Consumo	10μA – 170mA
Memória flash acoplável	16MB max (512k normal)
Processador	Tensilica L106 32 bit
Velocidade do processador	80 - 160MHz
RAM	32K + 80K
GPIOs	17 (multiplexado com outra função)
Apoio 802.11	b/g/n/d/e/i/k/r
Conexões máximas TCP simultaneamente	5
Faixa de temperatura operacional	-40ºC ~ 125ºC

Tabela 2.1 - Principais características do ESP8266. Fonte: (KOLBAN, 2015).

Com intuito de esclarecer ainda mais o funcionamento do ESP8266, destaco aqui algumas características principais referente ao Wi-Fi (ESPRESSIF).

- ✓ *Gestão de Qualidade de Serviço (QoS) integrado;*
- ✓ *Engine integrada para criptografias WEP, TKIP, AES e WPA;*
- ✓ *Suporte aos protocolos 802.11 b/g/n;*
- ✓ *Wi-Fi Direct (P2P), soft-AP;*
- ✓ *Pilha de protocolo TCP/IP integrada com suporte a IPv4;*
- ✓ *Wi-Fi em frequência de 2.4GHz com suporte a WPA e WPA2;*
- ✓ *Potência de saída em +20dBm no modo 802.11b;*
- ✓ *Suporte a uma variedade de antenas;*

O transmissor/receptor de rádio frequência (RF), suporta os seguintes canais de acordo com IEEE802.11b/g/n (ESPRESSIF).

Nº Canal	Frequência (MHz)
1	2412
2	2417
3	2422
4	2427
5	2432
6	2437
7	2442
8	2447
9	2452
10	2457
11	2462
12	2467
13	2472
14	2484

Tabela 2.2 - Canais e Frequências de operação do ESP8266EX

Destinado a plataformas móveis, o ESP8266 provê a capacidade de incorporar funções Wi-Fi em outros sistemas a um custo baixo, e foi projetado para projetos de mobilidade, eletrônicos vestíveis e a Internet das Coisas (*Internet of Things* – IoT) (KOLBAN, 2015).

O microcontrolador ESP8266 é escolhido para o desenvolvimento do projeto proposto, mas como foi citado anteriormente, o Arduino por ser uma plataforma open source, de fácil utilização e com mais tempo de mercado, é o microcontrolador mais conhecido e utilizado da atualidade, e para reforçar a escolha do ESP8266 é apresentado abaixo Tabela 2.3, com o comparativo dos dois modelos de microcontroladores e suas principais características.

	ESP8266	Arduino (Uno)
WI-FI integrado	Sim – padrão 802.11	Não
GPIOs	17	14 (20 Incluindo as analógicas)
Entradas Analógicas	1	6
Canais PWM	8	6
Velocidade de Clock	80MHz	16MHz
Processador	Tensilica	Atmel
SRAM	45KBytes	2KBytes
Flash	512kb ou mais (Separado)	32Kb (On Chip)
Tensão de Operação	3.3V	5V
Corrente Máxima por I/O	12mA	40mA
UART (hardware)	1 ½	1
Rede	Embutido	Separado
Documentação	Pobre	Excelente
Maturidade	Inicial	Maduro

Tabela 2.3 - Comparativo entre ESP8266 e Arduino (Uno) Fonte: (KOLBAN, 2015).

Conforme pode ser observado na Tabela 2.3, o ESP8266, já possui em seu sistema WI-FI integrado, o que facilita o desenvolvimento do projeto proposto, em relação ao Arduino que necessita de um módulo a parte, para que possa realizar comunicação via rede sem fio. Quanto a velocidade de clock, memorias Flash e SRAM, o módulo escolhido é muito superior a plataforma Arduino Uno.

Existem no mercado vários modelos de módulos equipados com o chip ESP8266EX, com algumas diferenças, como a quantidade de GPIOs disponíveis, e uma variedade de tamanho. Na Figura 2.3 é possível visualizar uma comparação da variedade de tamanhos entre os módulos, comparado com uma moeda no centro da imagem (CURVELO, 2015).

Figura 2.3 - Alguns módulos do ESP8266. Fonte: http://www.embarcados.com.br/modulo-esp8266/

Para o projeto proposto, serão utilizados os módulos ESP-12E e o ESP-01 onde suas principais características serão descritas a seguir.

Abaixo é apresentado modelo do módulo ESP-01, através da Figura 2.4 que mostra uma imagem real.

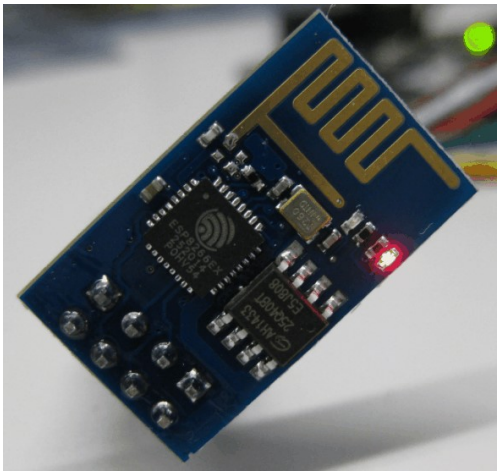

Figura 2.4 - ESP8266 modelo ESP-01. Fonte: http://www.embarcados.com.br/modulo-esp8266/

Para o funcionamento deste módulo, é necessário ligar os pinos 3 e 5 em nível lógico alto (VCC). O pino 3 (CH_PD) é responsável por colocar o módulo em funcionamento (nível

lógico alto), ou em modo de hibernação (nível lógico baixo), sendo que este último é muito utilizado para aplicações onde se torna necessário o racionamento de energia, como por exemplo o uso de baterias, assim, colocando o módulo em um tipo de “sono leve” podendo ser “acordado” em qualquer momento. O pino 5 (RST), é o pino responsável por reiniciar o módulo, onde, se colocado em nível lógico baixo reinicia o sistema. Com a Figura 2.5, pode ser verificado o esquema elétrico das ligações do modelo ESP-01.

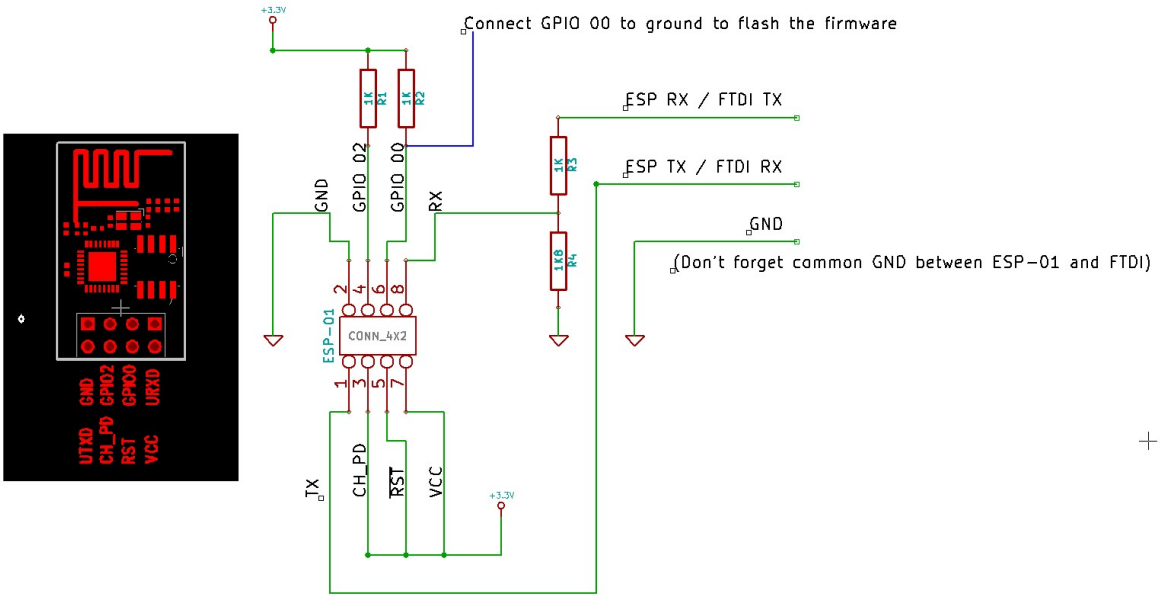

Figura 2.5 - Esquema elétrico módulo ESP-01 Fonte: https://cdn.hackaday.io/images/1926001420602279408.png

Os pinos de alimentação do ESP8266, modelo ESP-01, funcionam conforme a seguir:

✓ *VCC este pino recebe uma tensão de 3,3 volts máximo de 3,6 volts. Seu consumo é de até 300mA;*

✓ *CH_PD este pino é responsável por habilitar ou desabilitar o ESP8266, quando ligado no nível lógico alto (VCC), estará habilitada, caso contrário desabilitada em standby;*

✓ *GND pino de aterramento;*

✓ *Este modelo ESP-01 possui quatro portas digitais que podem ser configuradas com entrada/Saída, operam em 3,3 volts e fornecem no máximo de 12mA. Alguns destes pinos possuem funções específicas conforme mostrado a seguir;*

- ✓ *TX e RX pinos utilizados para receber e transmitir serial de dados;*
- ✓ *GPIO0 pino de entrada e saída de dados e pode ser utilizada como PWM. Para atualização do firmware, esta porta é colocada em nível baixo (GND);*
- ✓ *GPIO2 pino de entrada e saída dados e pode ser utilizada como PWM;*

Como foi descrito acima, o módulo ESP-01 possui 2 portas digitais (GPIO 0 e GPIO 2) disponíveis para se conectar sensores e ou atuadores. Assim apresento aqui o módulo ESP-12E, que disponibiliza todas as portas disponíveis do chip ESP8266X descritos no início deste capitulo. Desta forma, o esquema elétrico deste módulo é o mesmo apresentado na Figura 2.2. Através da Figura 2.6 é mostrado uma imagem real deste módulo.

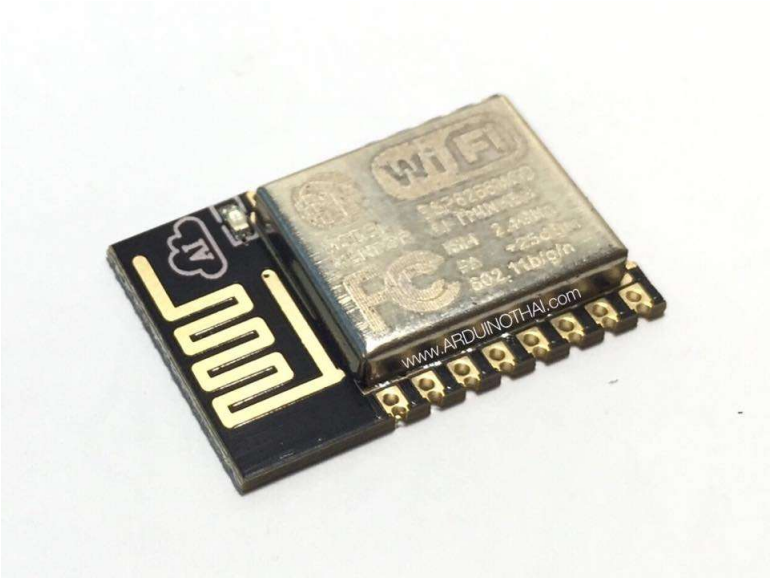

Figura 2.6 - ESP8266 Modelo ESP-12E. Fonte: https://www.arduinothai.com/product/679/esp8266-wifi-moduleesp-12e

Destaco que tanto para o módulo ESP-01 quanto o módulo ESP-12E, se faz a necessidade de um conversor USB Serial conhecido como módulo FTDI. Este módulo possui a capacidade de converter sinais USB em sinal serial, facilitando assim a comunicação entre o computador e os módulos em questão, através dos pinos TX e RX, podendo também alimentar o módulo com os pinos VCC (3,3V) e GND. A figura a seguir mostra a pinagem das portas disponíveis no módulo ESP-12E.

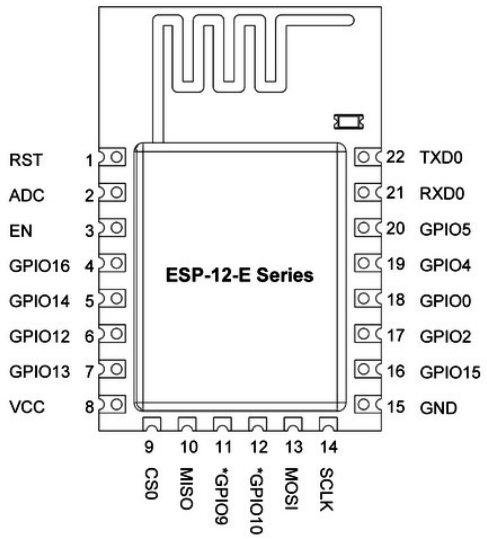

Figura 2.7 - Descrição das GPIOs disponíveis no módulo ESP-12E

Estes módulos, serão conectados ao roteador wireless trafegando dados através de sockets na camada de transporte do modelo OSI, utilizando protocolos TCP e UDP garantindo assim a interconexão bidirecional entre processos executados na rede LAN/WLAN, com intuito de realizar o monitoramento e os acionamentos solicitados pelo usuário através da página WEB, utilizando o protocolo HTTP, encontrado na camada de aplicação do modelo OSI (TANENBAUM e WETHERALL, 2011).

2.4. ACIONAMENTO DE CARGAS ELÉTRICAS COM RELÉS

Relé é um dispositivo comutador eletromecânico composto de um contato metálico que abre ou fecha, por meio de um campo eletromagnético induzido numa bobina. Esse dispositivo, tem grande importância na eletrônica desde sua invenção em 1837 por Willian Fothergill Cooke, Charles Wheatstone e Edward Davy, sendo muito utilizado na eletrônica para ligar e desligar dispositivos através da comutação de seus contatos. (BRAGA, 2012).

O relé é composto de dois circuitos internos isolados eletricamente, um acionado com baixa tensão e outro com alta ou baixa tensão e potência, sendo o primeiro circuito constituído de uma bobina com dois terminais que recebe um sinal elétrico (DC) que percorre as espiras

dessa bobina gerando um campo magnético, que por sua vez, aciona o segundo circuito, composto de 02 contatos fixos denominados normalmente aberto (NA) e normalmente fechado (NF) e uma parte móvel que é ligada ao contato NF. O campo magnético criado desloca o contato móvel para o pino NA, fazendo a corrente fluir por esse contato, possibilitando o acionamento de uma carga elétrica ligada a ele, seja, um motor ou lâmpada.

A Figura 2.8 é mostrado o esquemático de funcionamento do relé.

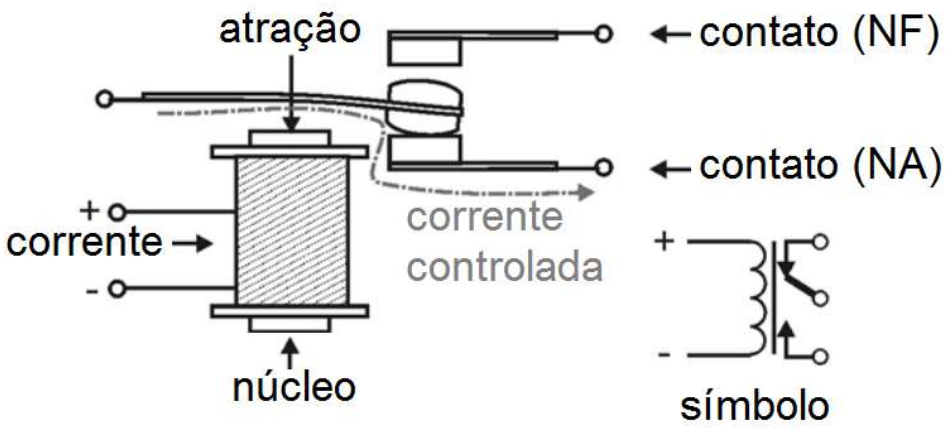

Figura 2.8 - Esquemático Relé. Fonte: Autor

Ao se utilizar o relé comandado através de um circuito com microcontrolador deve-se tomar algumas precauções, pois, quando o relé é desligado é gerada em sua bobina, por indução magnética, uma corrente contraria àquela que aciona o circuito, podendo causar algum dano ou até mesmo queimar o circuito controlador. Outro fator que deve ser levado em consideração, quanto sua utilização, é o consumo de corrente necessário para sua ativação, que nos reles mais comuns são cerca de 25mA.

Para o acionamento de cargas elétricas nesse projeto é utilizado um modulo relé (Figura 2.8), que basicamente é um circuito composto de um relé que suporta em seus contatos uma tensão de até 250VAC ou 30VDC e até 10A de corrente. No circuito de ativação da bobina é ligado em paralelo um diodo (componente que permite que a corrente flua somente em um sentido) e um acoplador óptico, utilizado como amplificador de corrente, para fornecer a corrente necessária para ativação do módulo relé, assim protegendo o circuito ativador da corrente de retorno.

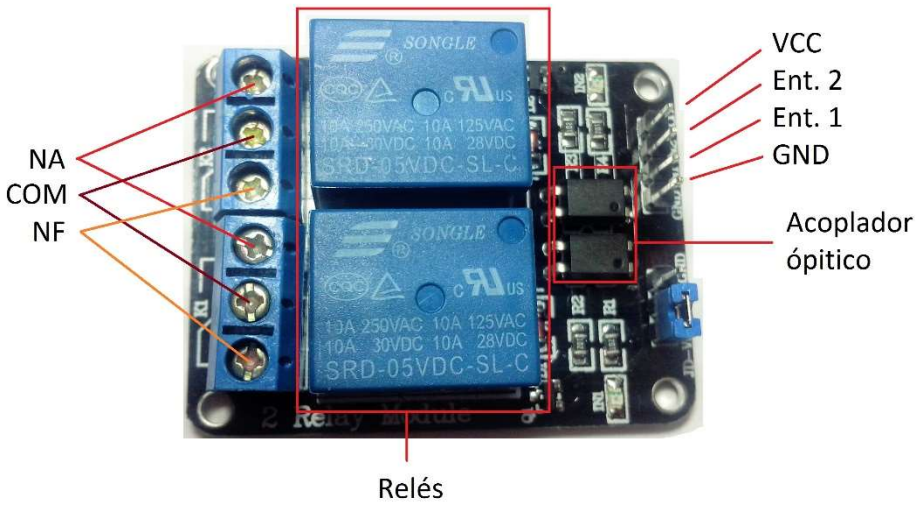

Figura 2.9 - Módulo Relé de 2 relés equipado com acoplador óptico. Fonte: Autor

O esquema elétrico abaixo mostra o circuito elétrico do módulo relé.

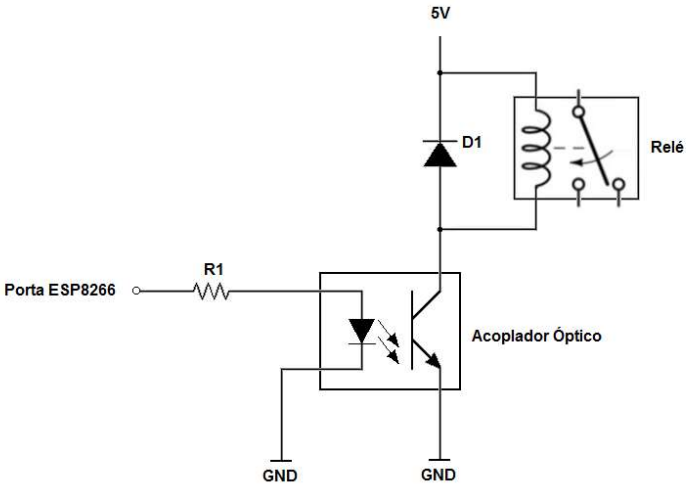

Figura 2.10 - Esquema elétrico módulo relé. Fonte: Autor

O módulo relé utilizado neste projeto (Figura 2.10) será comandado por uma das portas digitais do ESP8266, através do envio de um sinal de tensão de 3,3V em seu circuito de ativação, com o intuito de realizar o acionamento de uma lâmpada LED 220V 9W, em conjunto

com um sensor de luminosidade (LDR), responsáveis por retornar ao microcontrolador o status da carga, no caso ligada ou desligada.

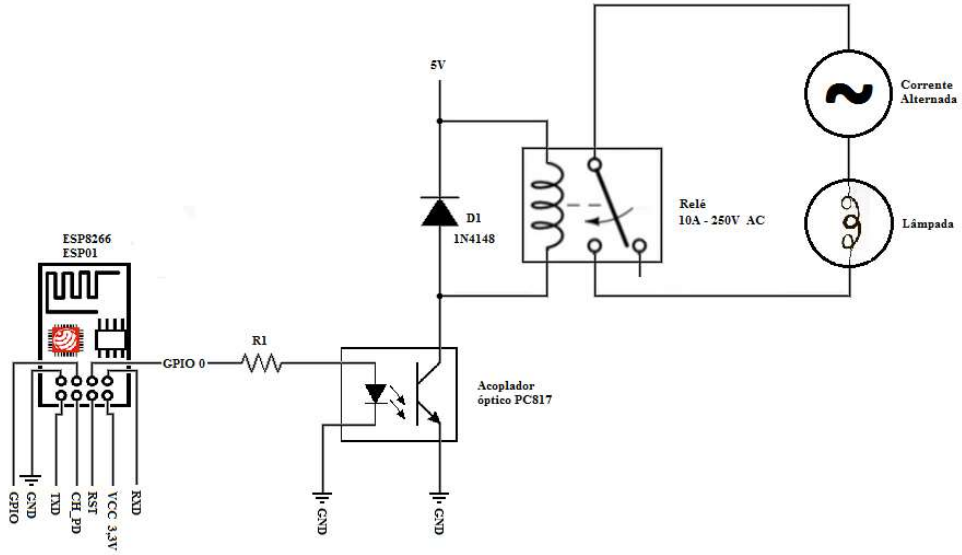

Figura 2.11 - Esquema elétrico do acionamento de uma lâmpada. Fonte: Autor

2.5. MEDIÇÃO E MONITORAMENTO DE UMIDADE E TEMPERATURA

Para disponibilizar ao usuário a funcionalidade de monitoramento da temperatura e umidade de um determinado ambiente de uma residência, é utilizado neste projeto o sensor DHT11 produzido pela AOSONG uma empresa chinesa. O sensor proposto trás em suas especificações um sensor digital permitindo realizar leituras de temperatura de 0 a 50 graus Celsius (ºC) e leituras de umidade de 20 a 90% (TECHNOLOGIES, 2012). Na Figura 2.12 é demonstrado as dimensões do sensor.

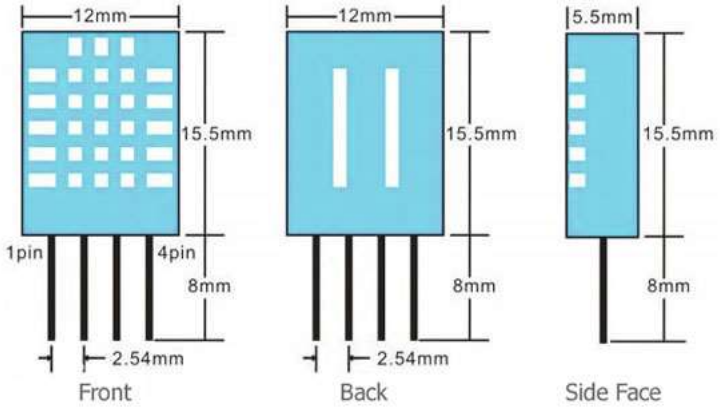

Figura 2.12 - Sensor DHT11. Fonte: www.aosong.com

O sensor DHT11 é formado por um sensor capacitivo para medir a umidade e por um termistor para medir a temperatura, ambos são dispositivos elétricos que têm sua resistência elétrica alterada termicamente. Este sensor é calibrado nos laboratórios durante sua fabricação. A precisão quanto a umidade pode variar ± 5% enquanto a de temperatura, pode variar ± 2ºC e ambos tem um tempo de resposta de <5s e pode ser alimentado com uma tensão DC de 3 a 5V consumindo uma corrente de 200uA a 500mA (TECHNOLOGIES, 2012).

O interessante neste sensor é o protocolo que é utilizado para transferir dados. Todas as leituras dos sensores são transferidas através de um único pino. A fim de enviar dados através do pino, é descrita a forma como os dados são transferidos, de modo que o transmissor e o receptor pode entender o que é “dito”.

É feito um pedido ao sensor DHT11 enviando-lhe um sinal baixo (GND) por 18ms, o mesmo muda do modo de baixo consumo de energia para o modo de funcionamento, à espera do receptor completando o sinal de partida elevando o sinal alto (VCC) por 40us. O DHT11 envia uma resposta automática indicando que recebeu seu pedido ao receptor. A resposta é 54ns em baixa (GND) e 80us em alta (VCC). Uma vez que é concluída, o DHT11 envia uma resposta de dados de 40 bits, que incluem a umidade e informações de temperatura e retorna para o modo de baixo consumo de energia. (TECHNOLOGIES, 2012). A Figura 2.13 demonstra como é feito esta comunicação entre o microcontrolador e o DHT11.

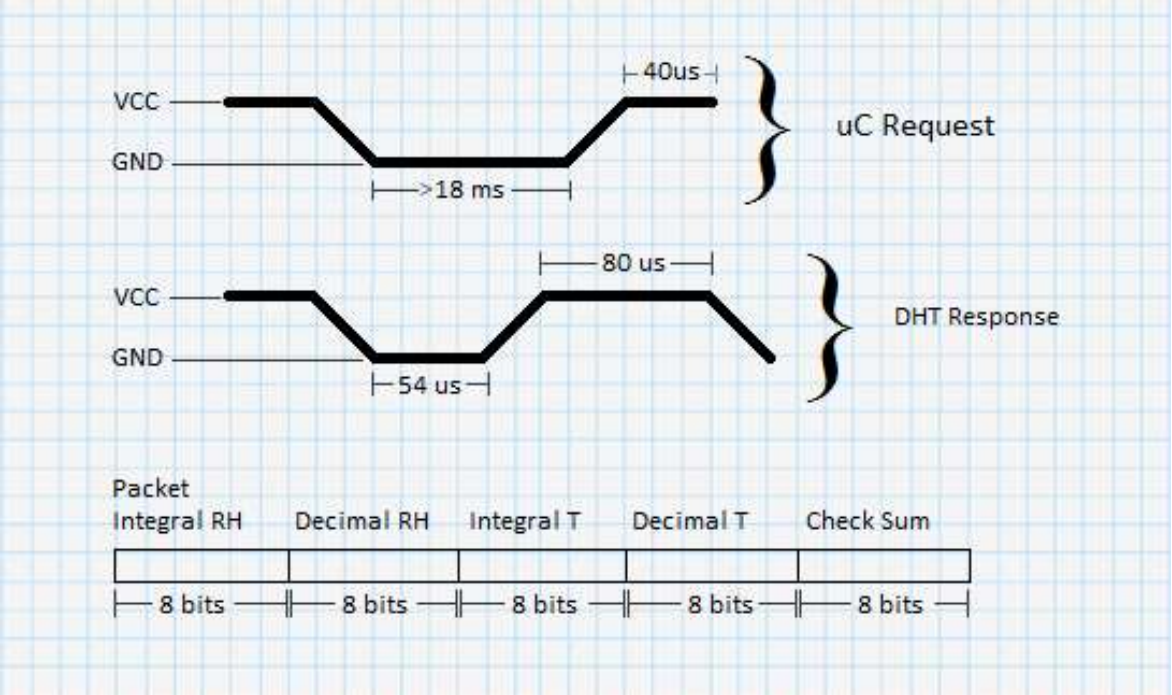

Figura 2.13 - Comunicação entre microcontrolador e DHT11. Fonte: http://robocraft.ru/files/datasheet/DHT11.pdf

2.6. FUNCIONAMENTO DE LEDS RGB

LED (Diodo Emissor de Luz), é um diodo semicondutor com junção PN que quando uma corrente atravessa esta junção, ocorre um estimulo e emissão que se concentra principalmente na faixa do infravermelho, gerando uma radiação de frequência muito bem definida que depende do tipo de material usado no semicondutor. Seu surgimento foi por volta de 1960 e desde então foi se tornando um dos componentes eletrônicos mais utilizados. Disponíveis em vários tamanhos, formatos, cores e intensidades.

Um LED RGB consiste em três LEDs encapsulados em um mesmo dispositivo, podendo ser controlado individualmente. A sigla RGB vem das suas cores primárias, vermelho (red – R), verde (green – G) e azul (blue – B) que combinadas pode gerar uma infinidade de cores (REIS, 2016). Figura 2.14 demonstra o esquemático de LED RGB, Anodo Comum e Catodo Comum.

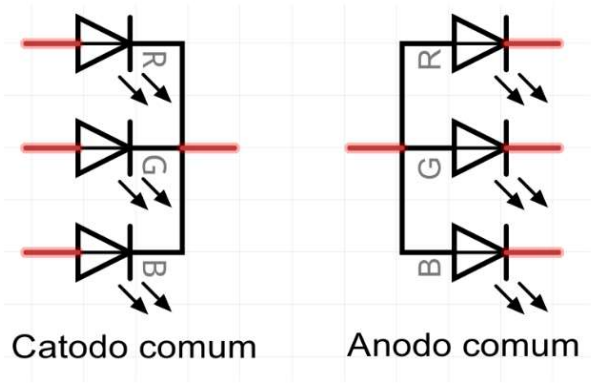

Figura 2.14 - Esquemático LED RGB catodo comum e anodo comum Fonte: https://www.mbeckler.org/microcontrollers/rgb_led/

Eles possuem quatro terminais, sendo que três deles são conectados a cada LED de cor individual, e o quarto terminal é comum a todos. Utilizando a técnica de PWM (*Pulse Width Modulation*), que por sua vez permite ajustar o nível de brilho em cada um dos LEDs, é possível controlar cada cor primária do LED RGB, gerando inúmeras cores distintas. Modulação por largura de pulso (PWM), é a prática de modular a largura de pulso de um sinal, usado nesta aplicação para controlar a potência média enviada a cada LED. O LED pisca numa frequência muito alta que o olho humano não consegue acompanhar por causa de um fenômeno chamado persistência de visão (BECKLER, 2009).

A persistência de visão ocorre quando uma imagem é vista por apenas uma fração de segundos, mas continua a ser “vista” pelo cérebro mesmo depois que a imagem original sumiu ou se moveu. Assim como acontece nas televisões e em telas de cinemas, que mudam rapidamente as imagens dando a impressão de movimento contínuo. Quando se liga e desliga o LED rapidamente, o cérebro é levado a enxergar a “média” do brilho baseado no ciclo de trabalho (largura de pulso ou *duty cicle*), do sinal aplicado ao LED (BECKLER, 2009).

A Figura 2.15 mostra 3 larguras de pulsos diferentes, 50%, 10% e 90%. Estes valores são o percentual que o sinal fica em nível lógico, em um determinado tempo.

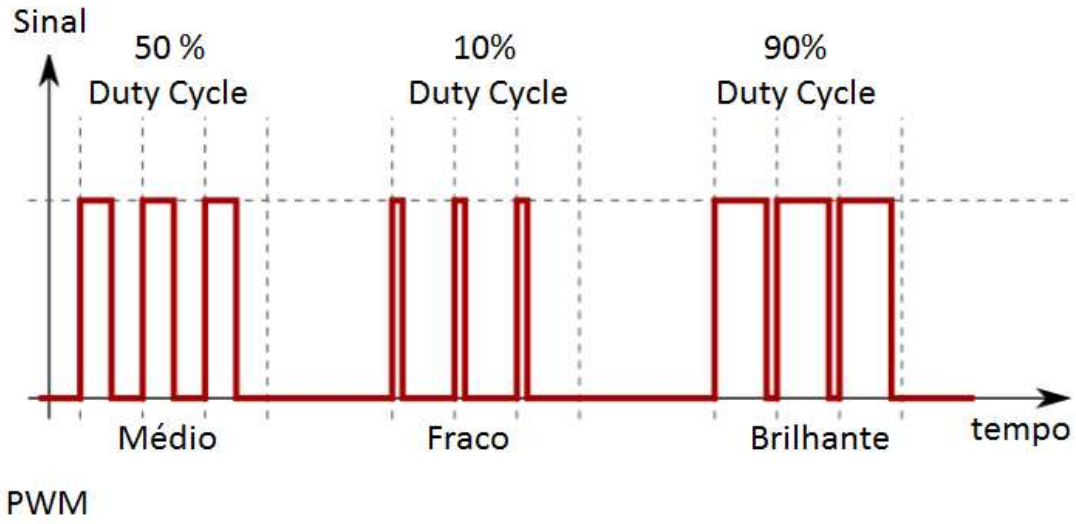

Figura 2.15 - Amostragens de duty cicle. Fonte: (BECKLER, 2009).

2.7. AMBIENTE DE DESENVOLVIMENTO PARA ESP8266

Existem várias plataformas de desenvolvimento para programar o microcontrolador ESP8266. Uma delas é a Sming framework, que teve sua primeira aparição em meados de 2015. Utiliza como linguagem de programação nativa o C++, e seu foco principal é uma alta eficiência no uso de memória e performance. Com este framework, é possível reaproveitar a vasta gama de bibliotecas que o Arduino IDE dispõe. Um dos grandes diferenciais do framework Sming é o fato do Sming ser baseado em eventos, ao contrário do Arduino que é baseado em um loop. Ele conta com um ambiente multi-plataforma e pode ser construído em Windows, Mac OS X e Linux (MINATEL, 2016).

Assim como o Sming framework, o Arduino IDE é um ambiente multi-plataforma, tem como linguagem de programação nativa o C e o C++, atualmente está na versão 1.6.12. Sua ideia principal é introduzir a programação a pessoas não familiarizadas como desenvolvimento de software. Seu editor de código possui vários recursos que ajudam o programador na identificação de erros e fácil compilação. A IDE é livre, e pode ser baixado por qualquer pessoa em seu site oficial, onde também está disponibilizado uma gama de códigos de exemplos (BELENZINHO, 2015).

Para o projeto proposto, utilizarei o Arduino IDE pois com esta plataforma, encontrei o maior número de documentação sobre o microcontrolador ESP8266.

A Figura 2.16, mostra a interface IDE indicando seus menus de acesso.

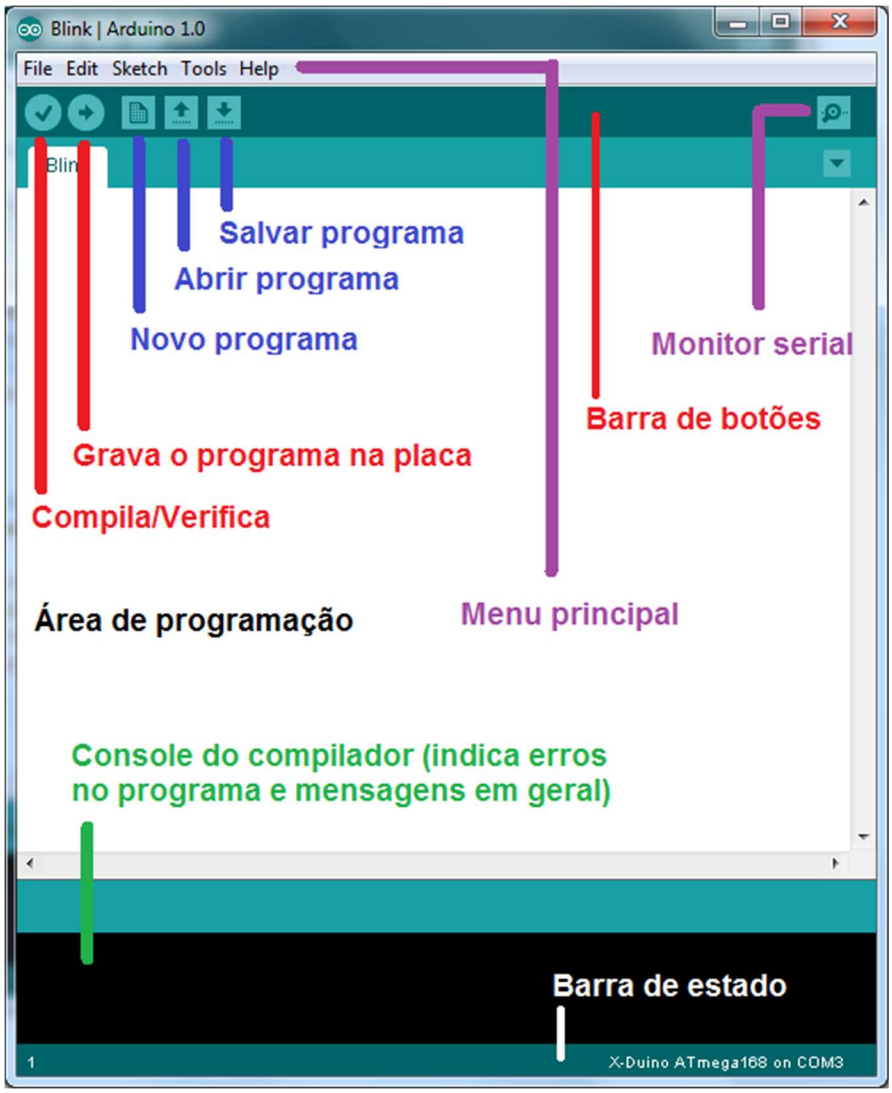

Figura 2.16 - Interface do Arduino IDE. Fonte: http://www.robotizando.com.br/pt-br/curso-de-arduino-aula-3-ambiente-de-desenvolvimento-integrado/

Como pode ser visto na figura anterior, o Arduino IDE é um ambiente de desenvolvimento completo podendo programar na linguagem C/C++, validar a sintaxe do código, submeter no programa ao microcontrolador acompanhando via console os erros e mensagens em geral. Pode também abrir projetos pré-carregados permitindo melhora-los, outro recurso que se destaca é o serial *monitor* onde via programação podemos obter valores de

variáveis obtidas a partir de sensores externos, eliminado a necessidade de um visor LCD externo para tal (ARDUINO.CC, 2016) .

Para que o ESP8266 funcione no Arduino IDE, é necessário instalar a versão 1.6.5, pois foi nesta versão que foi adicionada algumas funcionalidades que permitiram a IDE se tornasse compatível com o ESP8266. Com esta versão a maioria das bibliotecas disponíveis no para o Arduino, são compatíveis com o ESP8266 (KOLBAN, 2015). É necessário também instalar as placas dos módulos disponíveis para o ESP8266X.

CAPÍTULO 3 DESENVOLVIMENTO DO TRABALHO PROPOSTO

Este capítulo destina-se ao desenvolvimento do trabalho proposto de automação residencial bem como detalhes de sua confecção e ilustração do que foi feito com base dos conceitos teóricos apresentados no CAPÍTULO 2 .

3.1.DESCRIÇÃO DO SISTEMA PROPOSTO

O sistema proposto busca a criação de um protótipo de automação com base no uso do microcontrolador ESP8266 que disponibiliza o acesso remoto via uso de aplicação HTML através de uma rede Wi-Fi, possibilitando o acionamento de atuadores que controlam a iluminação, tomadas, fitas de LED e com um sensor especifico é possível monitorar a temperatura e umidade do ambiente.

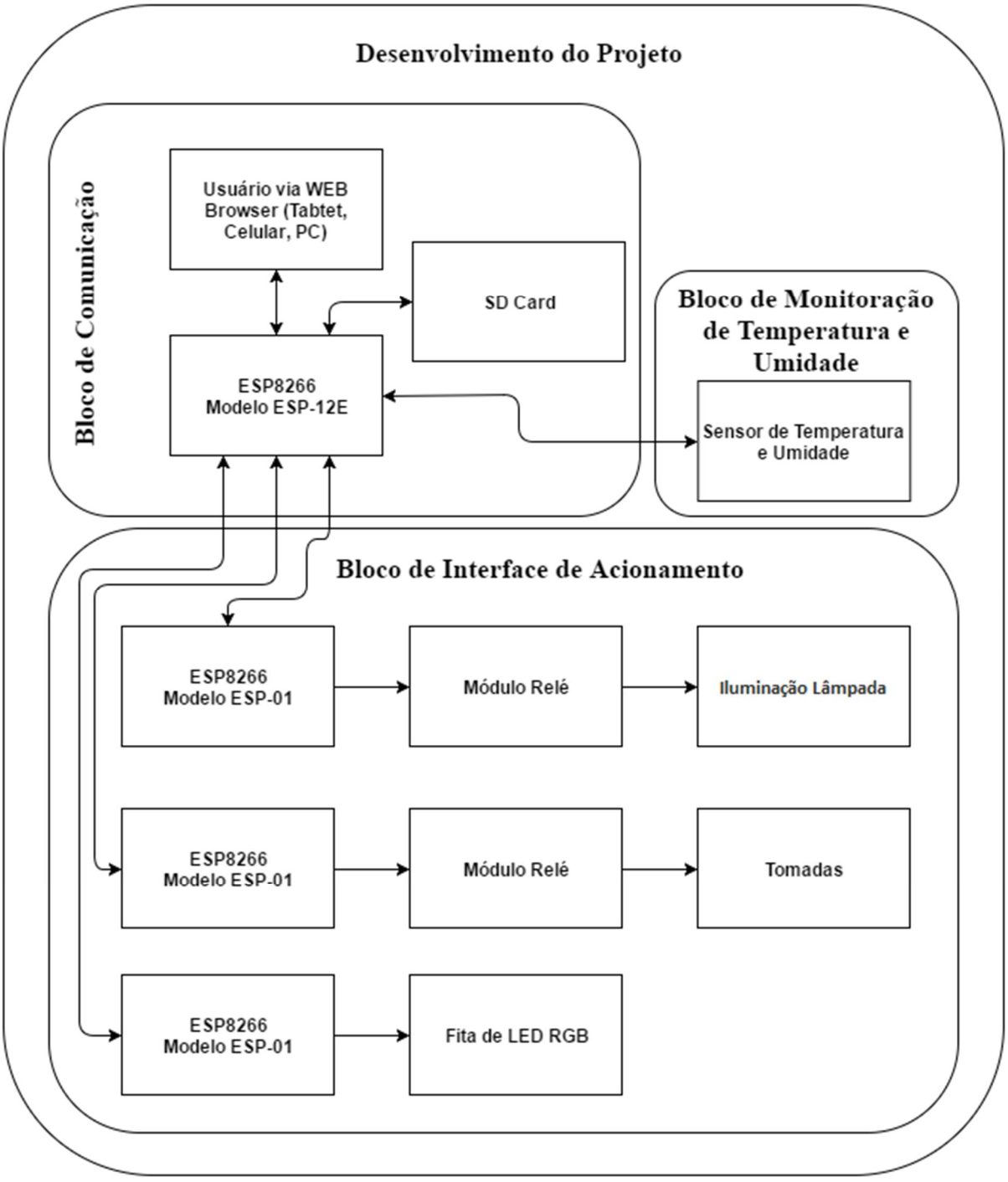

Figura 3.1 - Fluxograma do Protótipo de Automação proposto. Fonte Autor

O fluxograma (Figura 3.1), ilustra de forma breve, a lógica funcional da integração dos componentes, permitindo assim que este projeto de automação atinja os objetivos específicos relatados neste material. Para uma explicação transparente de como esta integração ocorre, o capítulo do desenvolvimento é dividido nas seguintes partes:

- ✓ *Bloco de Comunicação:*
  - . *Configuração do Roteador;*
  - . *Configuração do ESP8266;*
  - . *Servidor WEB;*
- ✓ *Bloco da Interface de Acionamento*
  - . *Acionamento das Lâmpadas;*
  - . *Acionamento das Tomadas;*
  - . *Acionamento da Fita de Led RGB;*
- ✓ *Bloco da Monitoração da Temperatura e Umidade*
  - . *Ornamentação do Protótipo Final na Maquete Proposta*

3.2. BLOCO DE COMUNICAÇÃO

Este item descreve os principais tópicos para o funcionamento do Bloco de Comunicação.

3.2.1. CONFIGURAÇÃO DO ROTEADOR

Por se tratar de uma automação através da rede LAN sem fio, primeiramente é configurado o roteador, alterando o nome (*SSID - Service Set IDentifier*) e a senha de acesso que neste projeto será com criptografia em WPA AES, pois é a partir destes dados que é configurado o microcontrolador em questão (Esta configuração será explicado no item 3.2.2). A criptografia WPA AES possui uma chave de 128 bits podendo gerar inúmeras senhas de acesso, tornando uma solução de baixo custo no que se diz a respeito de segurança para o projeto proposto. O roteador em questão é o RE057 fabricado pela empresa brasileira Multilaser. Vale ressaltar que pode ser utilizado qualquer roteador sem fio no padrão IEEE802.11b/g/n. A Figura 3.2 demonstra a configuração, e foi escolhido o *SSID* “TCCEdilson” e a senha “12345678”.

Figura 3.2 - Tela de configuração SSID e senha do roteador sem fio. Fonte: Autor.

Para um correto funcionamento da automação, é necessário atribuir um endereço IP estático a cada microcontrolador utilizado neste projeto, pois sem este, o microcontrolador poderá ganhar um novo IP toda vez que for reiniciado. A Figura 3.3 mostra a tela para atribuição estática do IP. É informado um nome, um IP e o endereço MAC para o dispositivo em questão. Vale ressaltar que é permitido adicionar mais de um dispositivo com IP estático, desde que não se repita o IP e o MAC.

Figura 3.3 - Tela de configuração do IP estático do Roteador. Fonte: Autor

O processo de comunicação não é concluído apenas com a atribuição do IP estático, é preciso configurar o microcontrolador para ter acesso a rede sem fio. Então é demonstrado em sequência como esta configuração é feita.

3.2.2. CONFIGURAÇÃO DO ESP8266

Para conectar o ESP8266 a rede sem fio descrita no item 3.2.1, devemos instalar no IDE do Arduino as placas disponíveis para o ESP8266X, pois é a partir da IDE que será feita a programação necessária para esta conexão. Assim, para esta instalação, é necessário a adição de uma URL específica, nas preferências do IDE no seguinte caminho: *Arquivo → Preferência → Additional Boards Manager URLs* como é mostrado na figura a seguir.

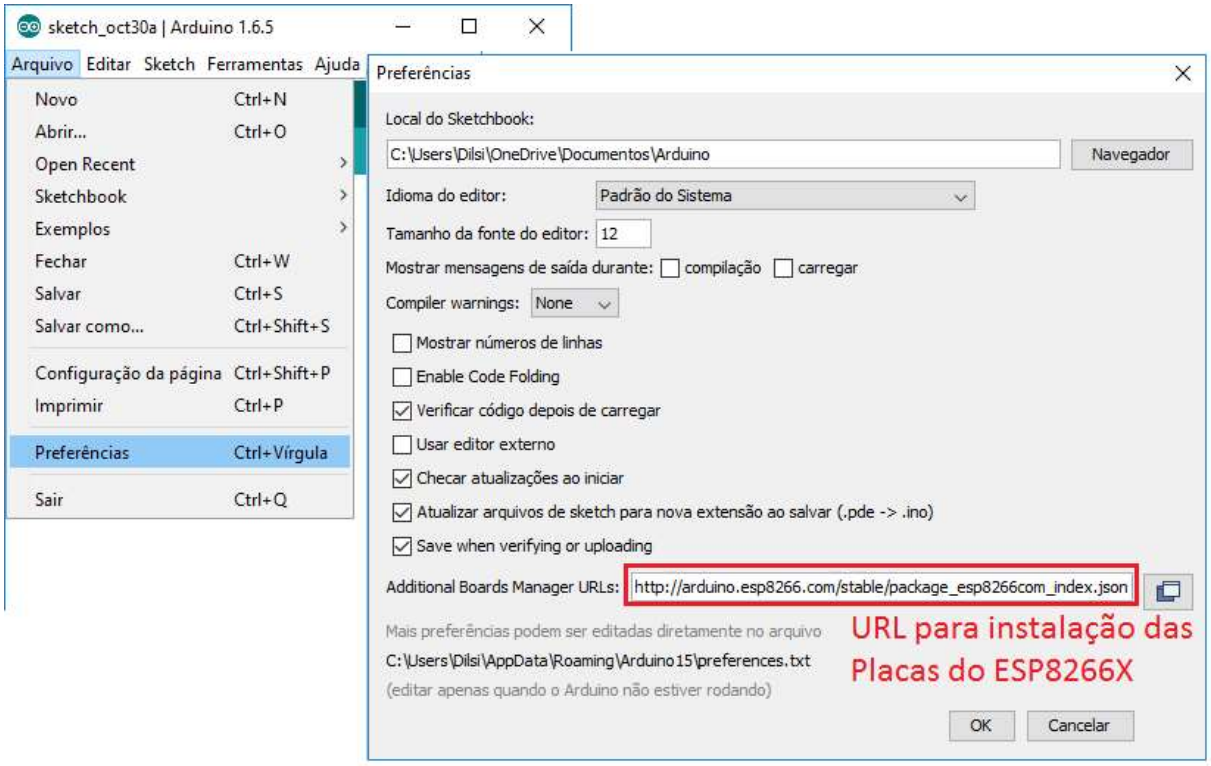

Figura 3.4 - Tela do IDE configuração placas ESP8266X. Fonte: Autor

Alterando a propriedade “preferências” do IDE como mostrado na Figura 3.4 à URL (http://arduino.esp8266.com/stable/package_esp8266com_index.json), se faz necessário para instalação destes módulos. Acessando o menu: *Ferramentas→ Placa: →Boards Manager...*, é aberto uma tela onde o IDE busca todas as placas instaladas, e realiza a busca automática. Assim que as placas são baixadas, é solicitado ao usuário sua instalação. Após sua instalação, o IDE torna disponível as diversas placas correspondentes ao módulo a ser utilizado. Como foi citado no referencial teórico, serão utilizados os seguintes módulos como solução para a automação: ESP-01 e o ESP-12E. Na imagem a seguir, é possível observar os módulos disponíveis após a instalação.

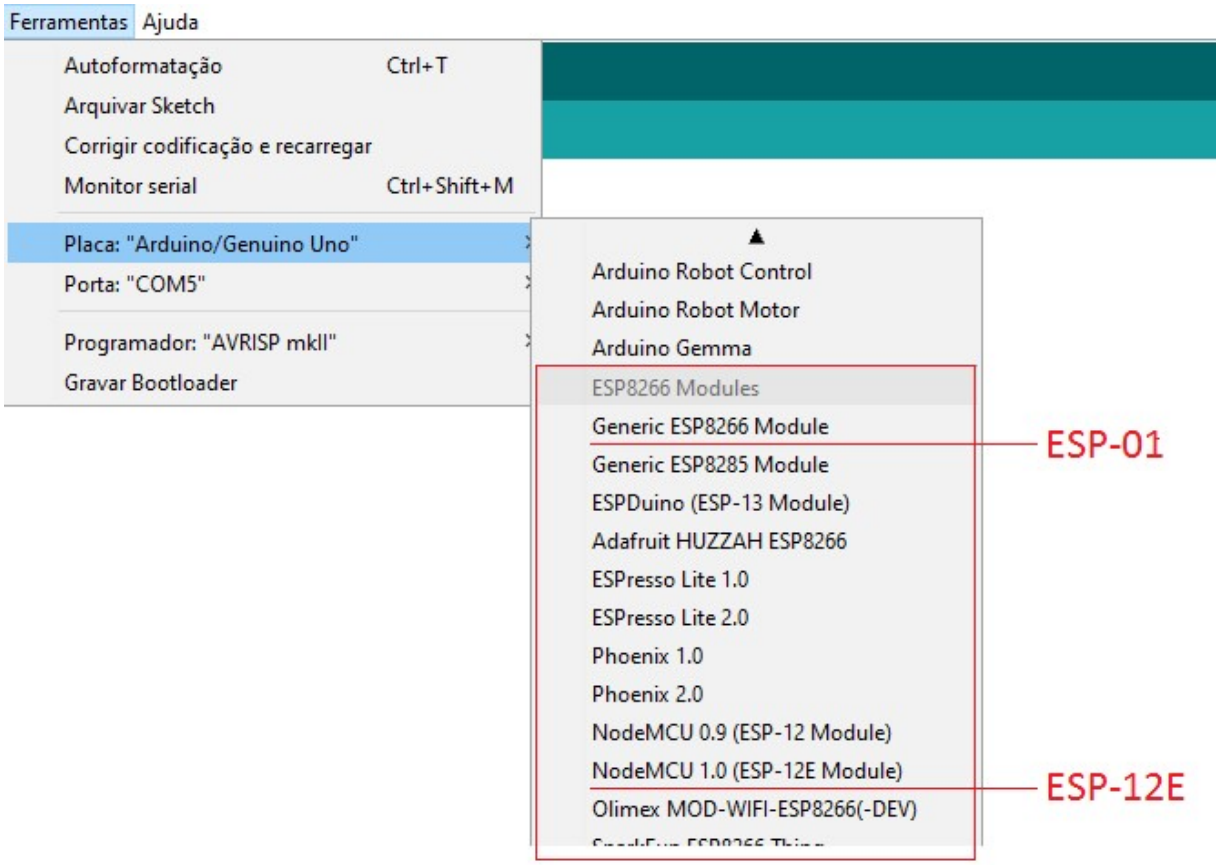

Figura 3.5 - Variedades placas instaladas para ESP8266X. Fonte: Autor

Após selecionar o módulo desejado, é possível desenvolver o código de programação em linguagem C/C++, para o sistema proposto, submetendo o mesmo ao microcontrolador, conectando-o ao computador através da porta lógica de dados USB, referenciando as bibliotecas de funcionamento do ESP8266.

Continuando a configuração da rede sem fio do ESP8266, é utilizado na codificação a biblioteca <ESP8266WiFi.h>, responsável por disponibilizar diversas funções para conexão *WiFi*. Cito aqui a principal função para tal feito, *WiFi.begin(ssid, password)*, que recebe o *SSID* e a senha, realizando de fato a comunicação ponto a ponto ao roteador sem fio

Figura 3.6 - Código para conexão ao roteador. Fonte: Autor.

Finalmente é criado uma conexão de ponto a ponto entre os dispositivos envolvidos. Todos os dispositivos de comunicação WiFi como o Servidor WEB quanto para o acionamento da lâmpada, das tomadas e da fita de LED, utilizarão o mesmo contexto para a conexão, adicionando um IP estático para cada um destes, permitindo que todos se comuniquem através de sockets utilizando protocolos da camada de transporte, dentre eles TCP e UDP.

Na sequência é demonstrado como o servidor WEB é configurado e como é disponibilizado a página WEB contendo o código HTML com suas funcionalidades.

3.2.3. SERVIDOR WEB

Servidor WEB é um programa responsável por aceitar pedidos HTTP de clientes, que no projeto proposto, está disponibilizado em todos os dispositivos de forma individual, ou seja, cada módulo ESP8266, é um servidor WEB.

O servidor WEB principal, é o denominado “ESP8266WebSite” configurado no IP estático192.168.43.100. Nele é onde está disponível o arquivo *index.htm*, responsável em conter o código HTML, sendo disponibilizado via socket ao browser, junto com outros arquivos para o correto funcionamento da página. Tais arquivos estão guardados em um cartão SD, através do módulo SD Card conectado ao microcontrolador. A comunicação do microcontrolador ao SD Card, se dá graças a biblioteca <SD.h>. Esta biblioteca possui uma variedade de funções para auxiliar a leitura e escrita de arquivos, tornando assim a aplicação ainda mais completa.

Assim que o usuário digita o IP do servidor ESP8266WebSite no browser, o microcontrolador envia o arquivo index.htm e seus arquivos dependentes, tornando possível a visualização da página WEB. O trecho de código abaixo é uma breve cópia da linguagem de programação responsável por tal feito:

```
bool loadFromSdCard(String path){ //Função responsável por enviar arquivos

    String dataType = "text/plain";//Declare a Variável com o tipo de Dado

    //Verifica se a requisição final é "/" adicionando o diretório.
    if(path.endsWith("/")) path += "index.htm";
    File dataFile = SD.open(path.c_str()); //Abre o arquivo no caminho solicitado

    //Verifica se encontrou o arquivo no diretório
    if(dataFile.isDirectory()){
        path += "/index.htm";//Define o arquivo index.htm
        dataType = "text/html"; //Muda o tipo de dado
        dataFile = SD.open(path.c_str();//Devolve o arquivo ao solicitante
    }
    //Caso não encontre o arquivo printa o uma mensagem de ERRO
    if (!dataFile){
        Serial.println("Erro no carregamento do cartão SD!");
```

```
                return false;
            }
            //Se o existir arquivos dependentes solicitados pelo Index.htm
            //muda o tipo de dado e disponibiliza estes arquivos.
            if (server.hasArg("download")) dataType = "application/octet-stream";
            dataFile.close();
            return true;
        }

void loop(void){
   server.handleClient();
}
```

Na função loop da IDE, o comando *server.handleClient();* tem a função de aguardar a conexão do browser. Assim que é enviada uma solicitação como citado no parágrafo anterior, o servidor WEB busca o arquivo index.htm na raiz do cartão SD e envia ao browser através da função *bool loadFromSdCard(String path).* Esta função verifica se existe o arquivo no cartão SD através endereço solicitado, devolvendo-o ao solicitante. Caso contrário, é informado tanto ao console quanto ao solicitante uma mensagem de erro.

Após o download destes arquivos, é apresentada finalmente ao usuário a página WEB, permitindo assim o acionamento das funcionalidades previstas neste projeto. Estas funcionalidades enviam ao Servidor WEB, requisições, que por sua vez, realiza uma nova requisição via socket ao servidor responsável por cada funcionalidade. Resumindo, o Servidor WEB principal (ESP8266WebSite), é responsável por realizar um roteamento das requisições de cada funcionalidade ao servidor responsável. O trecho de código abaixo é possível verificar como é solicitada a execução das funcionalidades.

```
void Setup(void) {
    server.on("/corFitaLed", HTTP_GET, corFitaLed);
    server.on("/tempUmi", HTTP_GET, tempUmi);
    server.on("/lampada", HTTP_GET, lampada);
    server.on("/tomada", HTTP_GET, tomada);
    server.onNotFound(handleNotFound);
...
}
```

A função *server.on();*, é responsável por identificar qual requisição foi enviada pela página WEB, chamando a função correspondente a requisição. Esta função recebe como parâmetro a funcionalidade, o tipo de requisição (HTTP_GET), e chama a função declarada no microcontrolador correspondente a funcionalidade. Como por exemplo, a funcionalidade de acionamento da lâmpada ("/lampada"), assim que a função *server.on();* identifica esta, é chamada a função void lampada() {}, que roteia uma nova requisição ao servidor ESP8266WebLampada, finalmente acionando a lâmpada. Desta mesma forma, são atendidas as solicitações aos demais servidores WEB (ESP8266WebTomada, ESP8266WebFitaLed). Mais detalhes destes serão descritos mais à frente.

Foi feito uma placa de circuito impresso para melhor acomodação dos componentes como o SD Card a ESP-12E e o sensor de umidade e temperatura para este servidor. O esquemático elétrico pode ser visto na Figura 3.7.

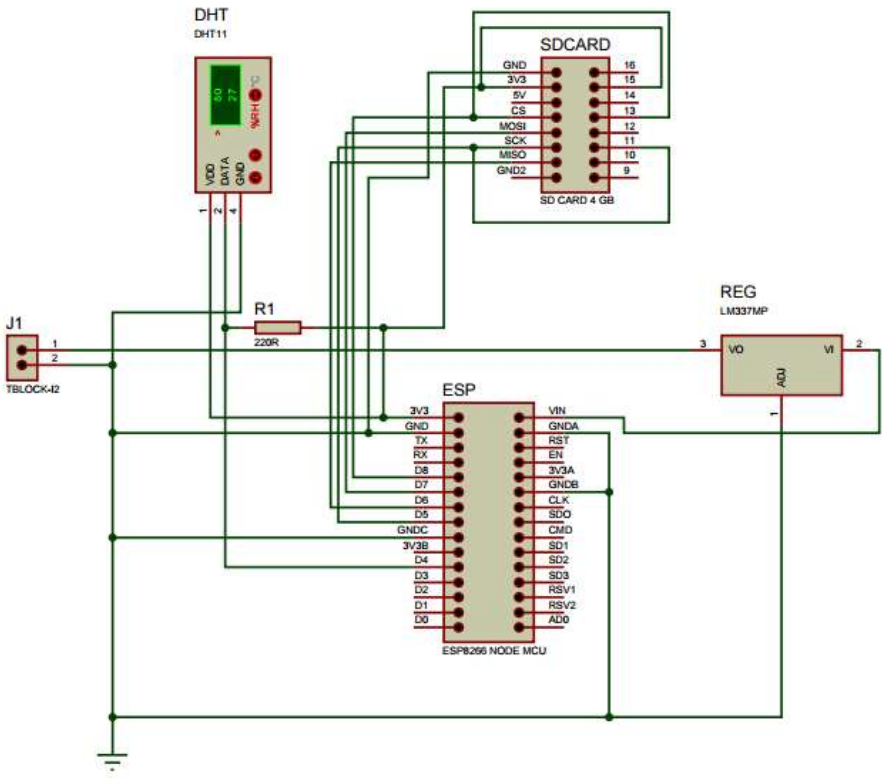

Figura 3.7 - Esquema elétrico da Placa do servidor ESP8266WebServer Fonte: Autor

3.3.BLOCO DE INTERFACE DE ACIONAMENTO

Como foi citado anteriormente, este projeto possui um servidor WEB para cada funcionalidade. Destaco aqui o funcionamento dos servidores que acionam a lâmpada, as duas tomadas controladas individualmente e a fita de LED.

3.3.1. ACIONAMENTO DA ILUMINAÇÃO

O processo de acionamento da lâmpada já foi explicado previamente neste documento, onde foi citado como exemplo a utilização da função *server.on();*, no item 3.2.3, sendo preciso assim trazer à tona o procedimento onde o microcontrolador aciona o relé e busca no sensor LDR o status da lâmpada, indicando se a lâmpada está acesa ou apagada. O servidor responsável por este acionamento é o ESP8266WebLampada configurado no IP 192.168.43.10.

Este status se dá graças o sensor LDR (resistor dependente da luz), que quando exposto a luz, altera seu valor de resistência elétrica para algumas centenas de Ohms e na ausência de luz, sua resistência é de milhares de Ohms. Com isso, este sensor é utilizado numa infinidade de aplicações (ELECTRONICA-PT). No projeto proposto, este sensor identifica se a lâmpada está acesa, retornando o valor (1), caso contrário retorna (0). Abaixo é demonstrado o esquema elétrico de como este sensor é utilizado.

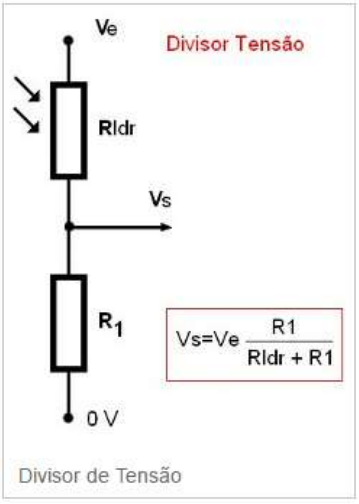

Figura 3.8 - Esquemático elétrico divisor de tensão com LDR. Fonte: http://www.electronica-pt.com/ldr

O divisor de tensão utilizando um sensor LDR (Rldr), se dá em função da intensidade de luz. A tensão presente em Vs, é determinada pela fórmula de divisão de tensão entre resistências. $Vs = Ve\frac{R1}{Rldr+}$.

- ✓ *Ve: Tensão de entrada*
- ✓ *Vs: Tensão de saída*
- ✓ *Rldr: Resistência do LDR*
- ✓ *R1: Resistência fixa*

A saída *Vs* do divisor de tenção, está conectado à porta de comunicação GPIO0 do microcontrolador (ESP8266WebLampada). Portanto, assim que é requisitado o status da lâmpada, o sistema realiza a leitura da função *void statusLampada() { },* que devolve ao solicitante o status da lâmpada. O trecho de código a seguir é possível observar é feito esta leitura.

```
void statusLampada() {
    //Concatena a mensagem para que seja devolvida.
    String message = ((String) digitalRead(PinLamp)) + ";" + ((String) digitalRead(PinLDR));
        returnMsgOK(message); //Devolve a mensagem ao requerente
        Serial.print ("Funcao statusLampada Rele1:"); //Printa no console um log de acesso.
        Serial.print (digitalRead(PinLamp));
        Serial.print (" ,LDR: ");
        Serial.println (digitalRead(PinLDR));
}
```

Finalmente é apresentado a partir da Figura 3.9 a ligação da lâmpada ao módulo relé e ao interruptor *threeway*. O fio neutro é conectado a um dos terminais da lâmpada e o fio fase é conectado ao interruptor que por sua vez está conectado por *threeway* ao módulo relé, onde, sai do relé um fio de retorno, permitindo assim o acionamento da lâmpada pelo sistema de automatização ou manualmente via interruptor. Para um melhor entendimento, na Figura 3.9, é possível observar como é feita esta conexão.

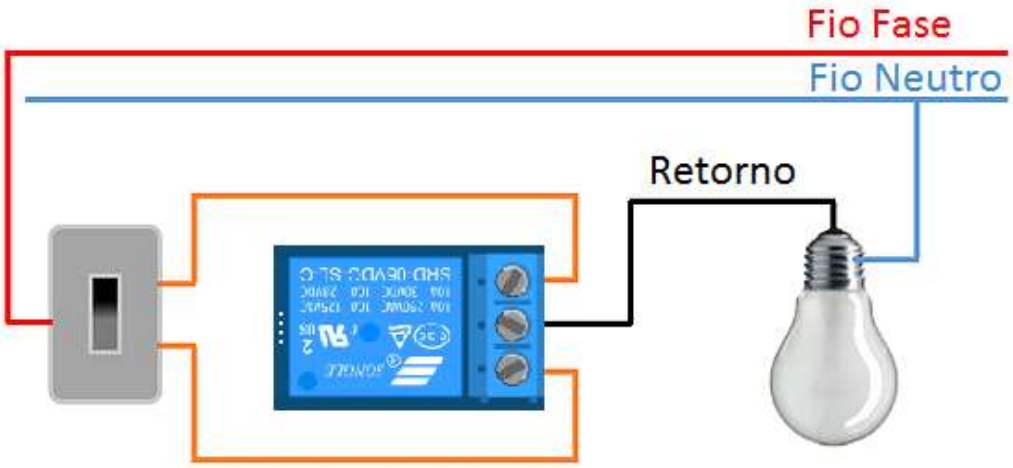

Figura 3.9 - Conexão threeway da iluminação. Fonte: Autor.

A figura abaixo mostra o acionamento da lâmpada a partir da página WEB. Ao usuário clicar na imagem da lâmpada, a página WEB envia uma requisição e então a lâmpada se apaga

Figura 3.10 - Acionamento da lâmpada através da página WEB.

Abaixo esquema elétrico da placa de circuito impresso do servido ESP8266WebLampada.

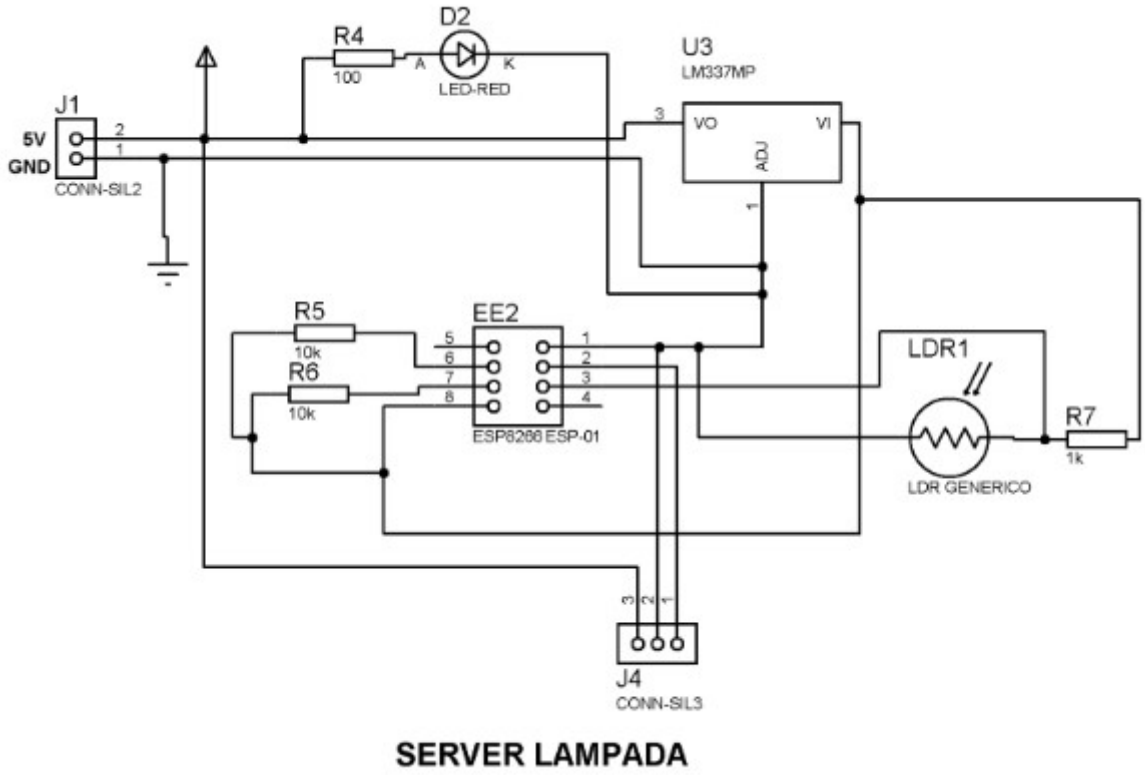

Figura 3.11 - Esquemático Placa Server Lâmpada. Fonte: Autor.

3.3.2. ACIONAMENTO DE TOMADAS ELÉTRICA

O servidor responsável pelo acionamento das tomadas é o de nome ESP8266WebTomada configurado com o IP 192.168.43.11. Assim que o usuário solicita o acionamento de uma das tomadas através da página WEB, este servidor recebe a requisição de qual tomada será ativada permitindo assim o acionamento de diversos dispositivos desde que não ultrapasse uma corrente elétrica de 10 Amperes a uma tensão de 220 Volts AC que é uma limitação tanto da domada quanto do módulo relé utilizado neste projeto. A figura abaixo é demostrada o esquemático elétrico das tomadas ao módulo relé.

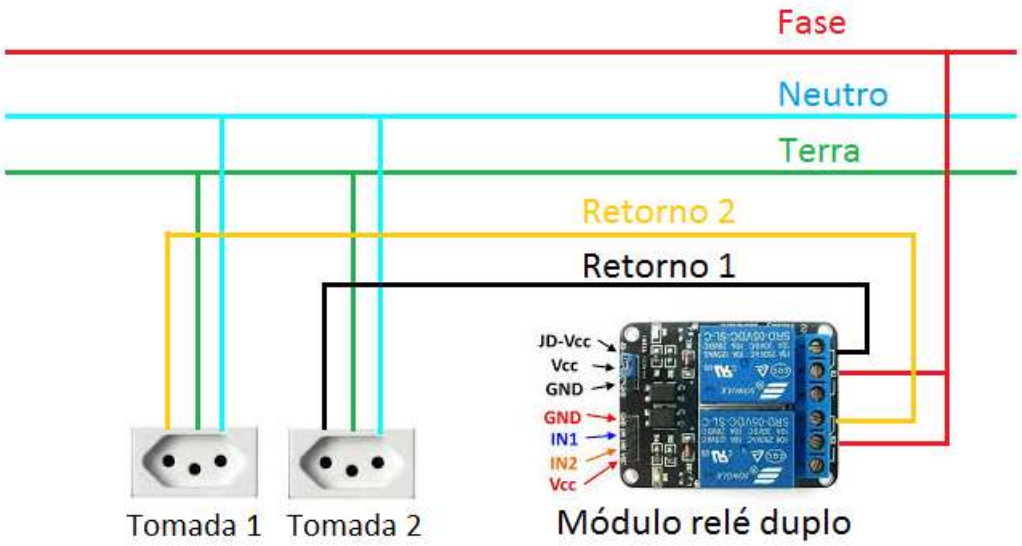

Figura 3.12 - Conexão das Tomadas ao módulo relé duplo. Fonte: Autor.

Para melhor disposição dos componentes utilizados para tal acionamento, foi confeccionado uma placa de circuito impresso interligando-a ao módulo relé. A figura do esquema elétrico a seguir.

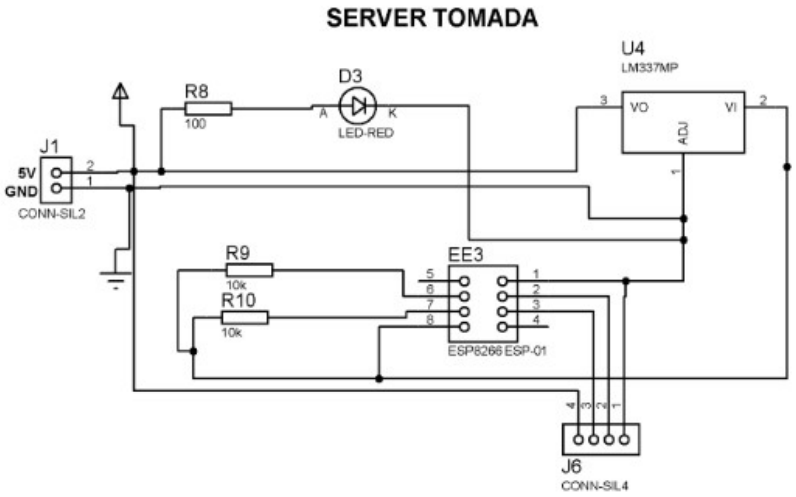

Figura 3.13 - Esquemático placa server Tomada. Fonte Autor.

O controle das tomadas é feito através da página WEB como mostra a figura abaixo.

Figura 3.14 - Botões On-Off para acionamento das tomadas. Fonte Autor.

3.3.3. ACIONAMENTO DA FITA DE LED RGB

Assim como a conexão e a solicitação de acionamento das funcionalidades por parte da página WEB citados anteriormente, se faz o acionamento deste servidor WEB. A diferença é que a página WEB passa por HTTP, os parâmetros das cores RGB e qual efeito num range de 0 a 6, sendo que 0 desliga a fita de LED.

O IP do servidor ESP8266WebFitaLed é 192.168.43.13, como já foi explicado anteriormente este IP é configurado no roteador sem fio. Trata-se de um IP estático. A fita de LED utilizada neste projeto é o modelo WS2811 de um metro de comprimento e que funciona numa tenção de 12Volts. Esta fita é popularmente apelidada de fita digital, pois ela só funciona com auxílio de um microcontrolador que através de uma biblioteca específica, é possível controlar um chip contido nela que possui um endereço a cada três LEDS RGB, permitindo assim que se acione a cada 3 LEDs numa cor diferente, realizando efeitos randômicos diversos através da biblioteca <Adafruit_NeoPixel.h> do Arduino IDE.

Esta fita possui 30 Leds porem com foi mencionado anteriormente, só é possível controlar 10 LEDs. O trecho de código abaixo em linguagem C\C++, é possível verificar como é passado a cor e o endereço da fita de LED.

```
//Teatro-estilo rastreamento luzes com efeito arco-íris
void theaterChaseRainbow(uint8_t wait) {
  for (int j=0; j < 256; j++) {     // Todas as 256 cores na roda
    for (int q=0; q < 3; q++) {
```

```
    for (int i=0; i < strip.numPixels(); i=i+3) {
      strip.setPixelColor(i+q, Wheel( (i+j) % 255));    //A cada terceiro pixel
    }
    strip.show();

    delay(wait);

    for (int i=0; i < strip.numPixels(); i=i+3) {
      strip.setPixelColor(i+q, 0);        //Desativar cada terceiro pixel
    }
   }
  }
}----------------------------------------------------------------------------------------------------------------
```

O trecho de código acima, é um dos efeitos proposto para este projeto. Ele é um efeito arco íris que vai circulando as cores por toda fita de LED, graças as estruturas de repetição contidas no código fonte. A função *strip.setPixelColor();* é responsável por enviar a fita a posição e a cor necessário para tal efeito.

Abaixo é mostrado como o usuário interage com a página WEB para alterar a cor e efeito desejado. Seu funcionamento é muito simples. O usuário seleciona a cor na paleta de cores e informa o efeito desejado, neste momento a página WEB envia ao servidor responsável pelo roteamento destas informações.

Figura 3.15 - Paleta de cores para controle da fita de LED. Fonte: Autor.

Para um melhor funcionamento da fita de LED, foi feito uma placa de circuito impresso para conectar o microcontrolador a fita. Esta placa tem um regulador de tenção 7805 responsável por regular a tensão de 12V fornecida pela fonte de alimentação para 5V que por sua vez alimenta um outro regulador de tensão (AMS1117), reduzindo ainda mais a tenção para 3,3V que é a tenção de funcionamento da ESP8266. Através da porta lógica GPIO2, os dados são passados a fita de LED WS2811. Abaixo o esquema elétrico da placa de circuito impresso.

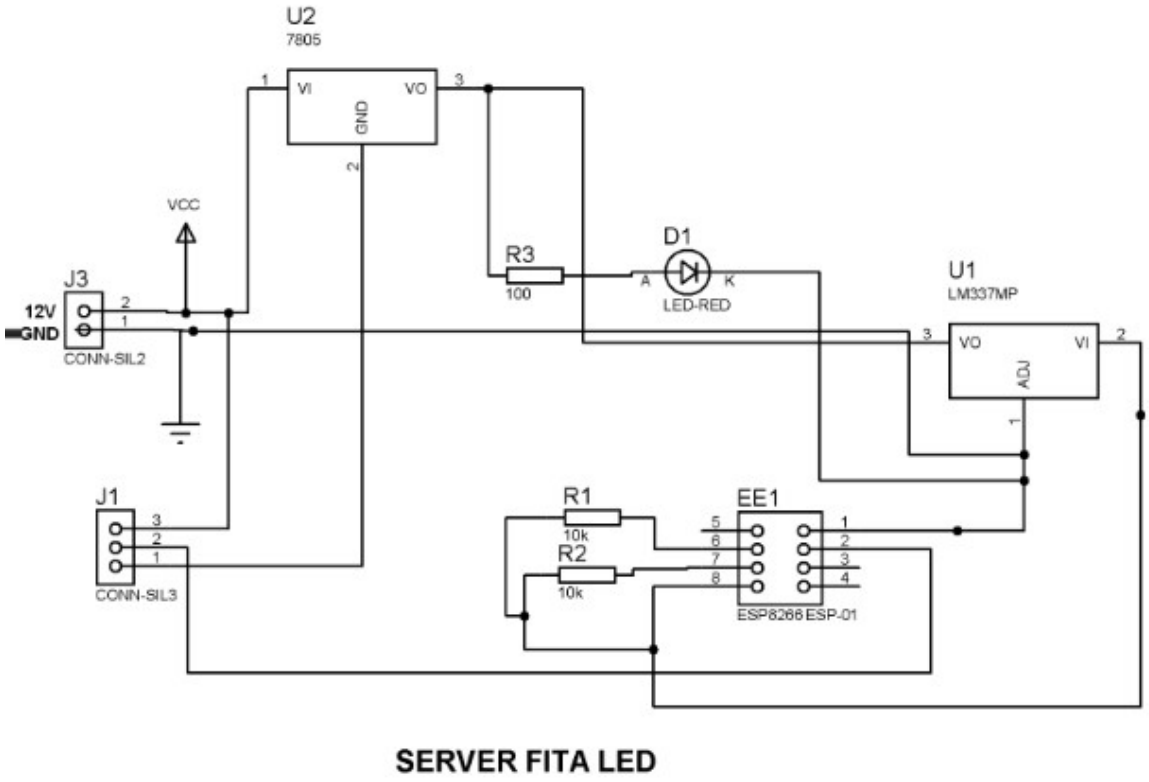

Figura 3.16 - Esquemático placa server Fita Led. Fonte: Autor.

3.4.BLOCO DE MONITORAÇÃO DA TEMPERATURA E UMIDADE

O sensor responsável por esta funcionalidade é o DHT11 que está instalada no servidor ESP8266WebServer. Assim que a página WEB solicita via HTTP dados da temperatura e Umidade o servidor busca estas informações como foi citado no item 2.5. A biblioteca utilizada é a #include <DHT.h> do Arduino IDE. Com ela é possível de forma mais fácil buscar tais dados. A função dht.readTemperature(); busca a temperatura em ºC e a função

dht.readHumidity(); busca o percentual da umidade do AR. Após chamar estas funções é só devolver a página WEB tais dados, que é mostrado para o usuário através de um gráfico como mostra a figura abaixo.

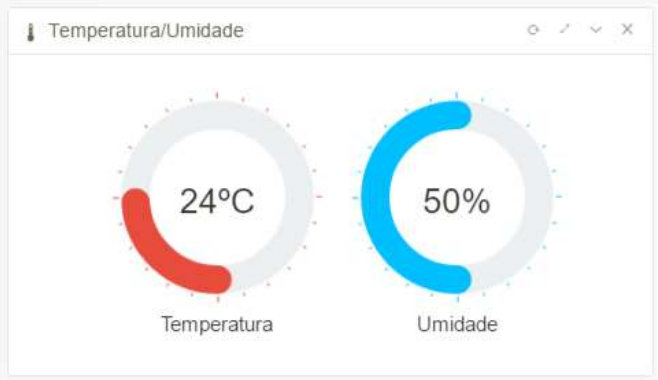

Figura 3.17 - Monitoramento de temperatura e umidade. Fonte: Autor

3.5.VERSÃO FINAL DO PROJETO

A versão final do projeto está dividida em duas partes. A primeira é a versão final da página WEB com uma interface intuitiva e de fácil manuseio. A segunda é a maquete composta por cinco totens. Cada totem é responsável por um tipo de acionamento, dando uma ideia ambientes separados em uma residência, e cada ambiente controlado por rede sem fio. Os totens são feitos de MDF comum acabamento estético de madeira.

A Figura 3.18 é resultado da primeira parte do projeto. Esta página foi desenvolvida na linguagem de programação HTML, java scritp e CSS permitindo a interação com os servidores WEB como citado neste material.

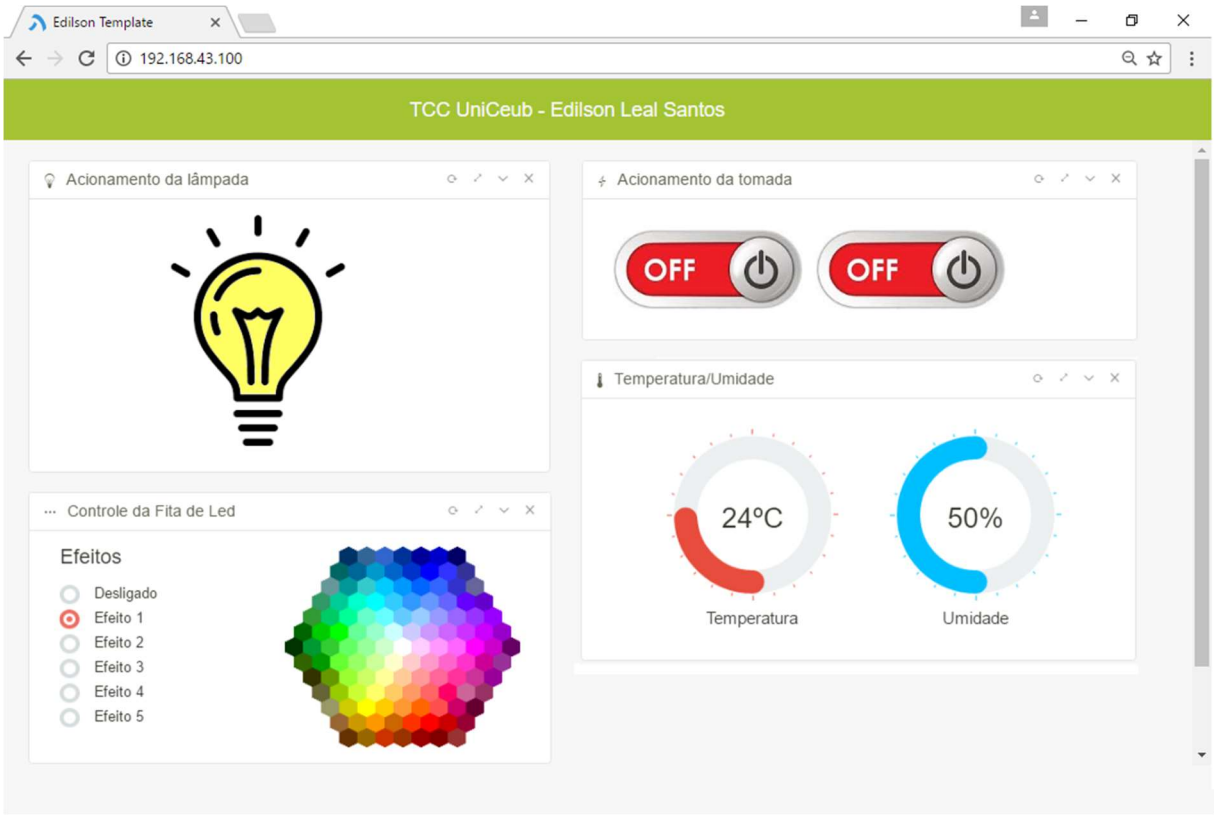

Figura 3.18 - Página WEB. Fonte: Autor.

Na sequência temos a Figura 3.19, que consiste na segunda parte do projeto. Esta maquete é do ESP8266WebServer, responsável por hospedar a página WEB, rotear serviços aos demais servidores além de buscar a temperatura e umidade através do Sensor DHT11. Este totem possui as seguintes dimensões (comprimento x altura x espessura): Base do totem 25 x

25 x 1,5 Cm e corpo de 25 x 35 x 1,5 Cm. Na parte de trás, possui duas tomadas 220/110 V AC de 10A, uma para a ligação da fonte de 5V DC para alimentação do circuito e outra para ligação do roteador WiFi que utiliza uma fonte de 9V DC. Estas duas fontes DC são bivolt.

Figura 3.19 - Maquete do servidor WEB ESP8266WebSite Fonte: Autor

Abaixo é possível observar a maquete do ESP8266WebLampada(Figura 3.20), responsável pelo acionamento da lâmpada e o retorno do status da mesma indicando se está acesa ou apagada através do sensor LDR. A ligação desta é do tipo ThreeWay permitindo que

a lâmpada seja acesa pelo relé e apagada pelo interruptor ou vice-versa. Este totem possui as seguintes dimensões (base x altura x espessura): Base do totem 25 x 35 x 1,5 Cm e corpo de 45 x 35 x 1,5 Cm. Na parte de trás, possui duas tomadas 220/110 V AC de 10A, uma para a ligação da fonte de 5V DC para alimentação do circuito e outra está disponível.

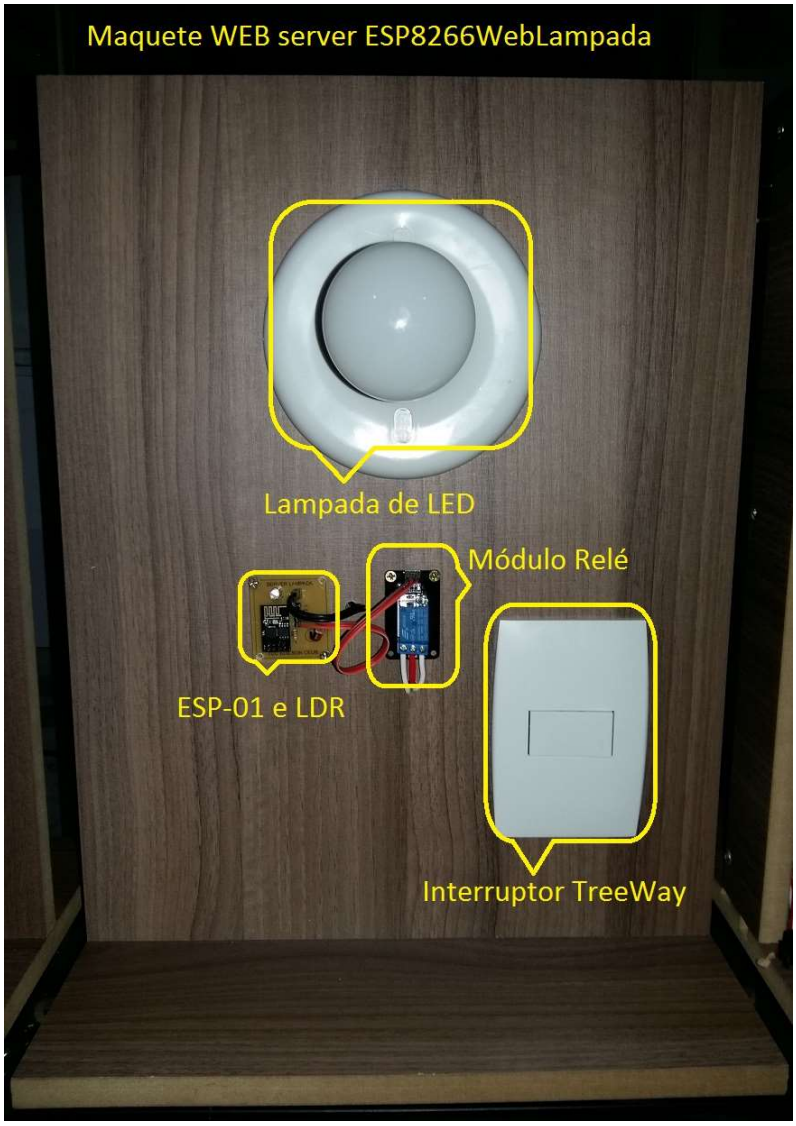

Figura 3.20 - Maquete Servidor WEB ESP8266WebLampada Fonte: Autor

Continuando, a maquete a seguir, é a ESP8266WebTomada. Ela é responsável por acionar as tomadas de forma individual através do microcontrolador ESP8266 modelo ESP-01 em conjunto com o módulo relé duplo, permitindo a conexão de aparelhos com corrente de até 10A na tensão 220V. Este totem possui as seguintes dimensões (base x altura x espessura): Base

do totem 25 x 35 x 1,5 Cm e corpo de 45 x 35 x 1,5 Cm. Na parte de trás, possui duas tomadas 220/110 V AC de 10A, uma para a ligação da fonte de 5V DC para alimentação do circuito e outra está disponível.

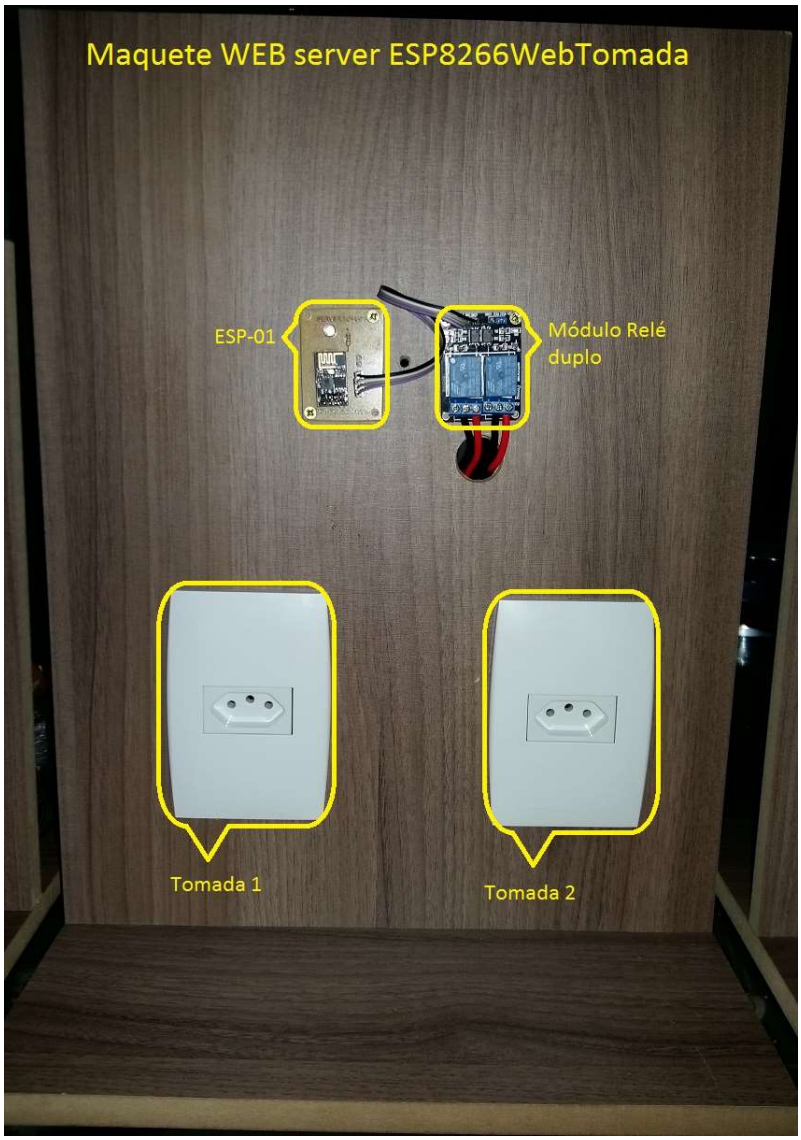

Figura 3.21 - Maquete Servidor WEB ESP8266WebTomada Fonte: Autor

Por último, a maquete ESP8266WEBFitaLed, permitindo assim o controle da fita de LED RGB que através da página WEB é possível controlar os efeitos randômicos disponíveis e ou alterar a sua cor com uma gama de cores disponíveis. Este totem possui as seguintes

dimensões (base x altura x espessura): Base do totem 25 x 35 x 1,5 Cm e corpo de 45 x 35 x 1,5 Cm. Uma fonte de 12V DC é utilizada para alimentação do circuito.

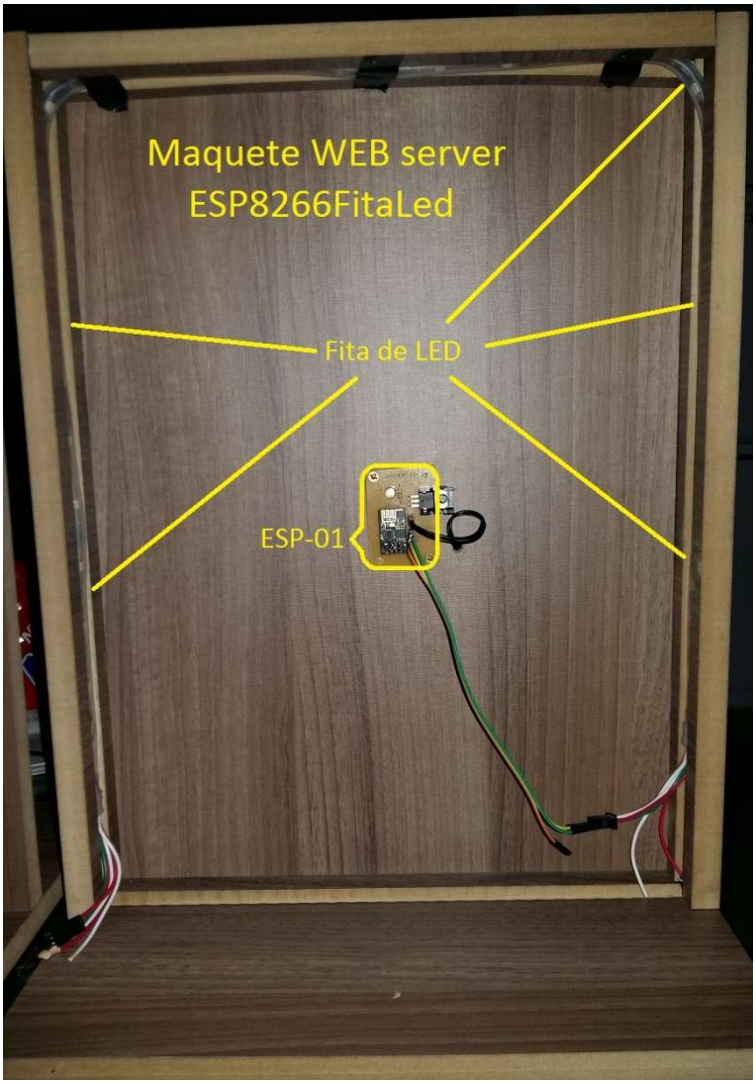

Figura 3.22 - Maquete Servidor WEB ESP8266WebFitaLed Fonte: Autor

CAPÍTULO 4 TESTES E RESULTADOS

Este capítulo tem por objetivo mostrar os testes e resultados realizados no decorrer do desenvolvimento, assim, validando se os objetivos específicos foram alcançados.

Estes testes estão divididos nos seguintes senários:

- ✓ ***Primeiro cenário:*** *Teste no bloco de comunicação;*
- ✓ ***Segundo cenário:*** *Teste no bloco de interface e de acionamento;*
- ✓ ***Terceiro Cenário:*** *Teste no bloco de monitoração de temperatura e umidade.*

4.1. PRIMEIRO CENÁRIO

Este cenário de teste tem como objetivo verificar se a interface do bloco de comunicação está funcionando corretamente.

O acesso remoto se dá graças ao uso do microcontrolador ESP-12E, que por sua vez está conectado ao roteador sem fio, disponibilizando assim o acesso ao servidor WEB. Como foi explicado no item 3.2.3, a página WEB está disponível através do módulo SD Card que está conectado ao microcontrolador, completando o servidor WEB. No diagrama a seguir (Figura 4.1), é possível observar a lógica para a realização deste teste.

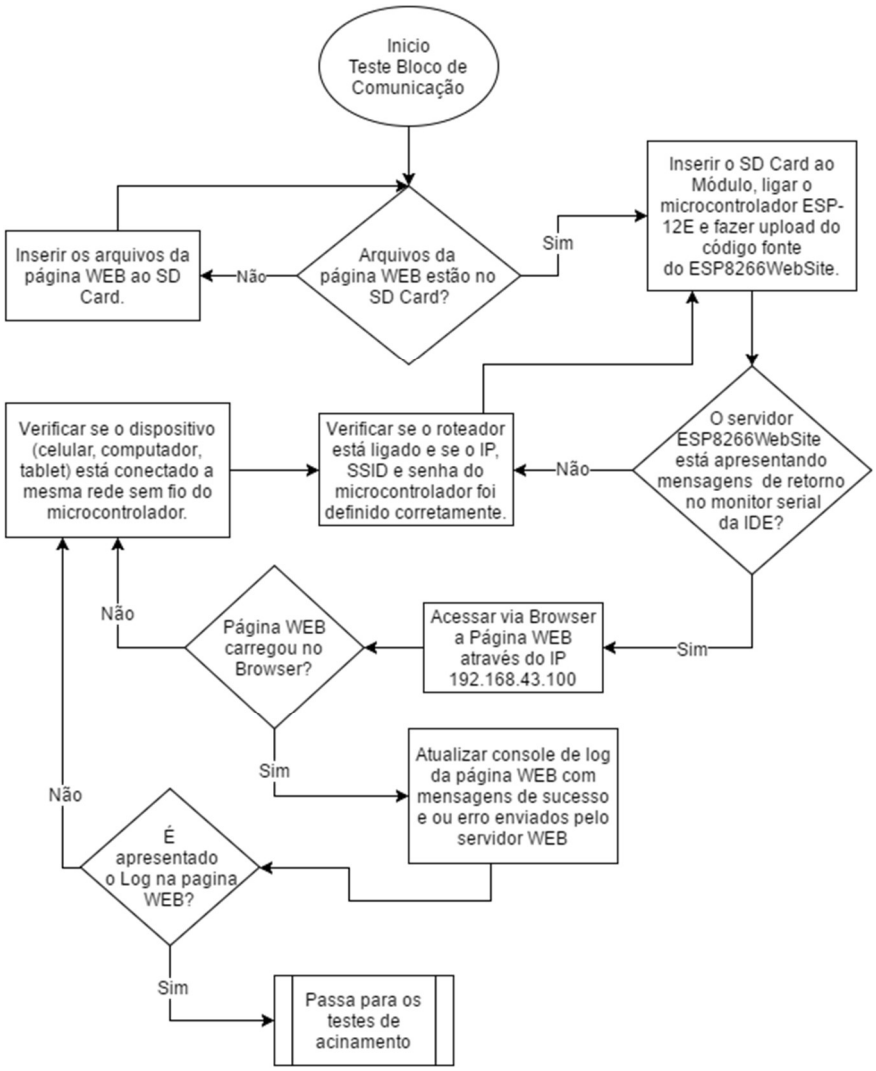

Figura 4.1 - Diagrama dos testes com o Bloco de Comunicação. Fonte: Autor.

Foi feito inicialmente o teste de comunicação acessando a página WEB através do IP 192.168.43.100 de um dispositivo remoto (Notebook), conectado a mesma rede sem fio do microcontrolador, com intuito de verificar o retorno de mensagens no *serial monitor* do Arduino IDE.

Na programação do código fonte no Anduino IDE, foram colocados *prints* que são considerados importantes para o desenvolvimento deste projeto, assim, apresentando no *serial monitor* da IDE. A função responsável por estas mensagens é o comando *"Serial.print(descrição da mensagem)"*, como pode ser visto na Figura 4.2, mostrando uma parte do trecho de código responsável pelas primeiras mensagens ao ligar o microcontrolador, e a inicialização da velocidade da porta serial (*"Serial.begin(115200);"*).

Figura 4.2 - Código que mostra no serial monitor a qual rede sem fio foi conectado. Fonte: Autor

A função *"Serial.println(descrição da mensagem)"* é responsável por enviar a mensagem ao *serial monitor***,** seguido de uma quebra de linha diferentemente da função *"Serial.print(descrição da mensagem)"*, que mostra a mensagem na mesma linha. A figura a seguir (Figura 4.3), mostra o teste realizado descrevendo as mensagens de inicialização do ESP8266WebServer no *serial monitor* da IDE.

Figura 4.3 - Mensagens de inicialização do Serial Monitor da IDE. Fonte: Autor.

Voltando ao teste de acesso da página WEB, quando o usuário acessa a página, é escrito no *serial monitor* cada arquivo para o correto funcionamento da página WEB, onde foi observado que todos os arquivos foram enviados neste teste.

Para uma melhor visualização, foi criado a funcionalidade de Log do microcontrolador, que permite ao usuário uma melhor visualização dos mais importantes registros de log. Esta funcionalidade está disponível na página WEB. Assim que o usuário realiza uma solicitação ao microcontrolador, este grava em uma variável as mensagens mais importantes, devolvendo à página WEB tais registros. Assim foi possível observar através do painel Log do microcontrolador, as requisições feitas através da página, como pode ser visto na Figura 4.4.

Figura 4.4 - Log do microcontrolador na página WEB. Fonte: Autor

Como foi citado anteriormente, ao receber a mensagem gerada pelo microcontrolador, é confirmado a integridade de envio e retorno da solicitação HTTP da página WEB. Caso ocorra algum erro na requisição HTTP, é mostrado uma mensagem em uma janela (Pop-up) ao usuário, informando o erro ocorrido.

O bloco de comunicação é responsável por fazer uma ponte entre a página WEB e cada um dos servidores do bloco da interface de acionamento (ESP8266WebLampada, ESP8266WebTomada, ESP8266WebFitaLed). Portanto foram realizadas 34 solicitações a cada 30 minutos aos servidores, que por sua vez, não obteve erro na comunicação via requisição HTTP como mostra tabela a seguir.

Servidor	Quantidade de Sucessos	Quantidade de Falhas	Taxa de Sucessos
ESP8266WebLampada	34	0	100%
ESP8266WebTomada	34	0	100%
ESP8266WebFitaLed	34	0	100%

Tabela 4.1 – Resultado dos testes de comunicação. Fonte: Autor.

Lembrando que a tabela acima mostra somente a conexão de ponte entre os servidores responsável por cada funcionalidade. Os testes de cada servidor são descritos a seguir.

4.2.SEGUNDO CENÁRIO

Neste cenário o Bloco da Interface de Acionamento, serão testadas as funcionalidades de cada servidor separadamente. Todos os servidores funcionam praticamente da mesma forma. Todos eles recebem requisições HTTP da página WEB, que por sua vez envia a funcionalidade e os parâmetros para o correto acionamento. Temos como exemplo o acionamento das tomadas (*http://192.168.43.11/acionaTomada?iTomada=0&iLigar=1*), onde é informado o IP do Servidor, em seguida o nome da funcionalidade e por último os parâmetros que são identificados logo após o identificador "?". Cada parâmetro é composto por um nome de uma variável e logo depois o valor, separados pelo identificador "&".

Este exemplo está solicitando ao servidor o acionamento da tomada 0 (*iTomada=0&iLigar=1*) e foi acompanhado na funcionalidade de Log do microcontrolador da página WEB. Tal funcionalidade já citado neste material item 4.1.

4.2.1. TESTE NO SERVIDOR ESP8266WEBLAMPADA

Este servidor recebe requisições HTTP enviada do servidor ESP8266WebSite, para o acionamento do relé ligando/desligando a lâmpada, devolvendo o status da lâmpada graças ao sensor LDR. Este sensor recebe a intensidade de luz do ambiente para verificar se a lâmpada

está acesa ou apagada. Para testar esta funcionalidade, o diagrama de blocos (Figura 4.5) foi .elaborado.

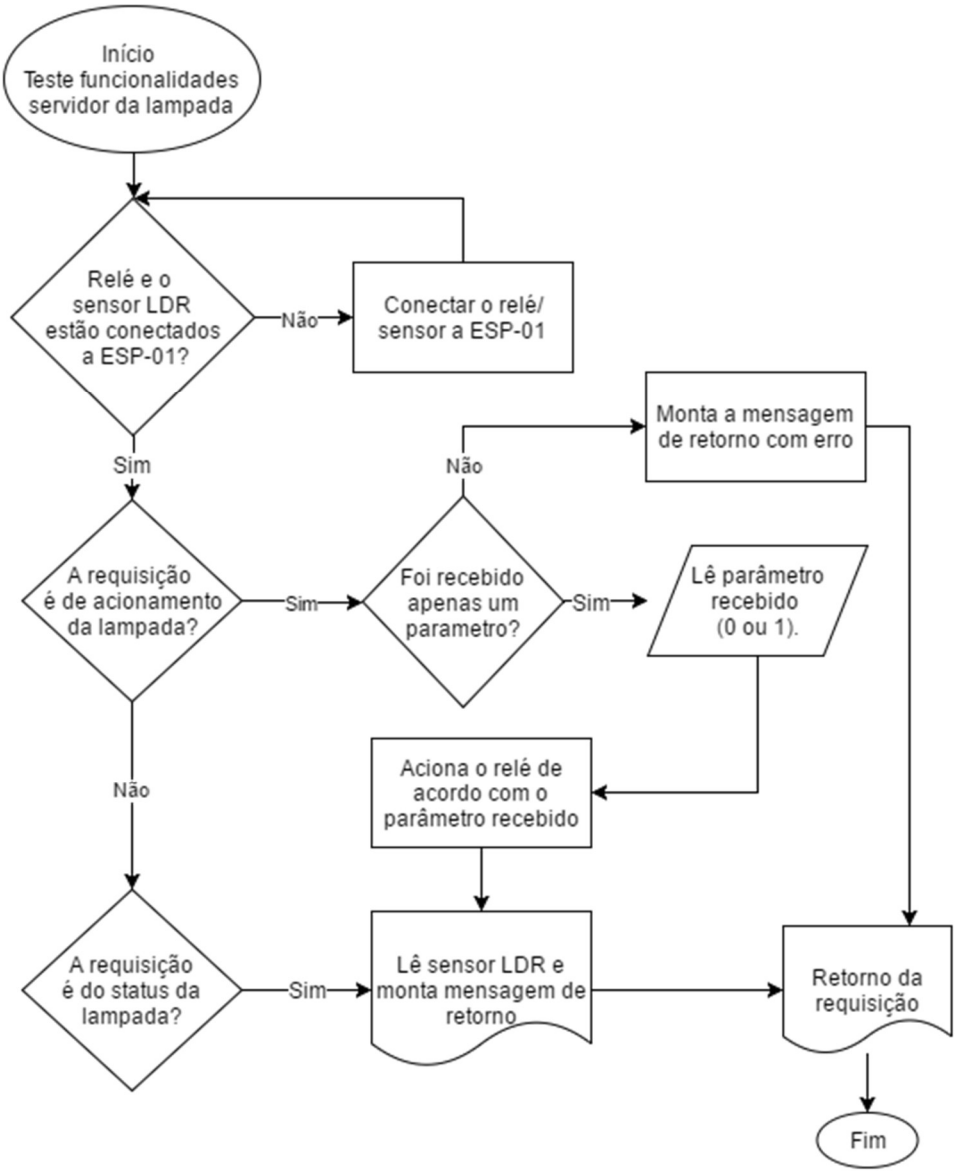

Figura 4.5 - Diagrama dos testes servidor da lâmpada. Fonte: Autor

O teste de acionamento da lâmpada se deu da seguinte forma: foi solicitado o acionamento pela página WEB, do módulo relé que por sua vez acionou a lâmpada, porém na devolução do status da lâmpada a partir do sensor LDR, o microcontrolador estava lendo o valor 1 indicando que a lâmpada estava desligada. Este erro acontece graças ao retardo (5 ~ 10ms), existente no acionamento do módulo relé utilizado neste projeto.

Para corrigir este problema, foi adicionado no código fonte da IDE do Arduino o comando *"delay(20);"*, que tem como objetivo pausar a execução do microcontrolador por 20ms, tempo suficiente para que o módulo relé acione a lâmpada, permitindo que o sensor LDR leia a intensidade de luz da lâmpada logo após esta pausa, devolvendo o status correto à página WEB.

Após esta correção, foi feito 35 acionamentos da lâmpada (ligar e desligar) e 35 solicitações de Status da lâmpada em um intervalo de 30 min via página WEB, obtendo sucesso na maioria dos acionamentos como foi evidenciado na Tabela 4.2.

Solicitação	**Quantidade de Sucessos**	**Quantidade de Falhas**	**Taxa de Sucessos**
Acionamento da Lâmpada	33	2	~94,29%
Status da Lâmpada	35	0	100%

Tabela 4.2 - Resultado de teste da lâmpada. Fonte: Autor

Ocorreu duas falhas no acionamento da lâmpada como mostra a Tabela 4.3, que ocorreram por causa da conexão WiFi entre o computador e o roteador quebrando a conexão com o servidor WEB. Nas duas falhas, o dispositivo estava conectado a outra rede WiFi disponível no ambiente de teste e após uma intervenção manual, voltou a funcionar. Já a requisição do Status da lâmpada não apresentou problemas.

A fórmula utilizada para achar a taxa de sucessos para o acionamento da lâmpada é apresentada logo a seguir:

$$Taxa\ de\ sucesso = \left(1 - \frac{Quantidad\ \ de\ falha}{Quantidade\ de\ acionamentos}\right) * 100$$

$$Taxa\ de\ sucesso = \left(1 - \frac{2}{35}\right) * 100$$

$$Taxa\ de\ sucesso \cong 94{,}3\%$$

Como não foi apresentado erro na solicitação do status da lâmpada, a taxa de sucesso é de 100%.

4.2.2. TESTE NO SERVIDOR ESP8266WEBTOMADA

Assim como o servidor da lâmpada, este servidor também recebe requisições HTTP com a diferença do módulo de relé que permite o acionamento de duas tomadas controladas individualmente. O módulo relé está conectado a ESP-01 nas portas GPIO 0 e a GPIO 2, permitindo o acionamento das tomadas.

O teste para esta funcionalidade, leva em consideração que o código fonte já está disponível no microntrolador. Sendo assim, foram feitos os seguintes teste:

1. O módulo relé está conectado ao modulo ESP-01?

   Não → Conectar o módulo relé a ESP-01.

   Sim → Passa para o próximo questionamento.

2. A requisição é de acionamento da lâmpada?

   Não → Passa para o questionamento do Status.

   Sim → Passa para o questionamento dos parâmetros

3. Foram recebidos dois parâmetros da página WEB?

   Não → Monta a mensagem de erro e devolve à página WEB.

   Sim → Lê os parâmetros recebidos, realiza o acionamento das tomadas de acordo com os parâmetros recebidos, lê o status das tomadas e devolve à página WEB parâmetros com tais status.

4. A requisição é de status das tomadas?

   Não → Monta a mensagem de erro e devolve à página WEB.

   Sim → Lê o status das tomadas e devolve à página WEB parâmetros com status das tomadas.

Foi feito um diagrama de blocos (Figura 4.6), para melhor visualização do teste da funcionalidade de acionamento das tomadas.

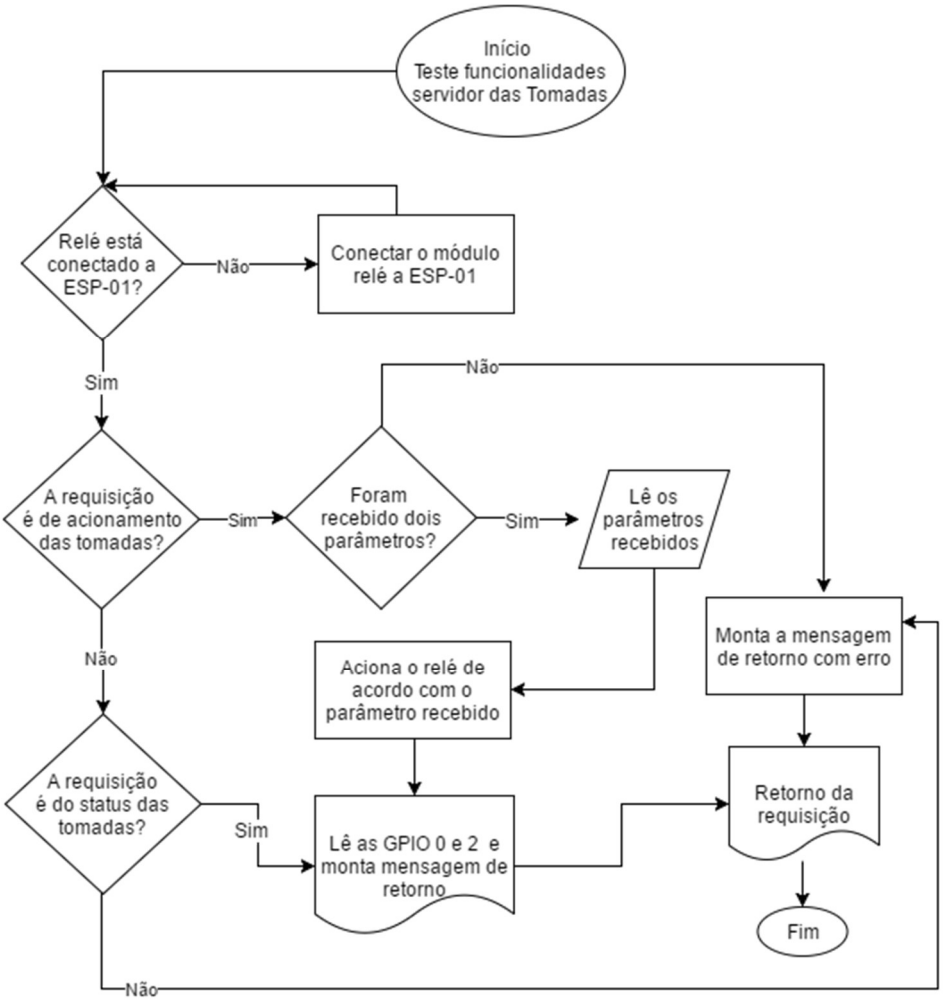

Figura 4.6 - Diagrama dos testes servidor das tomadas. Fonte: Autor

Para testar a funcionalidade de acionamento das tomadas foram feitas 35 solicitações entre ligar e desligar para cada tomada e 35 solicitações do status da tomada em um intervalo de 45 minutos totalizando ~3hrs. Para evitar o erro que aconteceu no acionamento da lâmpada (Tabela 4.2), foram removidas todas as redes WiFi do dispositivo (computador), deixando somente a rede do roteador WiFi, responsável pela conexão dos dispositivos de teste deste material. A Tabela 4.3 mostra os testes desta funcionalidade.

Solicitação	Quantidade de Sucessos	Quantidade de Falhas	Taxa de Sucessos
Acionamento da tomada 1	35	0	100%
Acionamento da tomada 2	35	0	100%
Status da tomada	35	0	100%

Tabela 4.3 - Resultados de testes das tomadas. Fonte: Autor.

Como mostrado na Tabela 4.3, foi obtido 100% de sucesso em todos os acionamentos. A Figura 4.7, mostra o log da página WEB de alguns destes acionamentos, indicando o início e o término da requisição HTTP para o acionamento/status das tomadas.

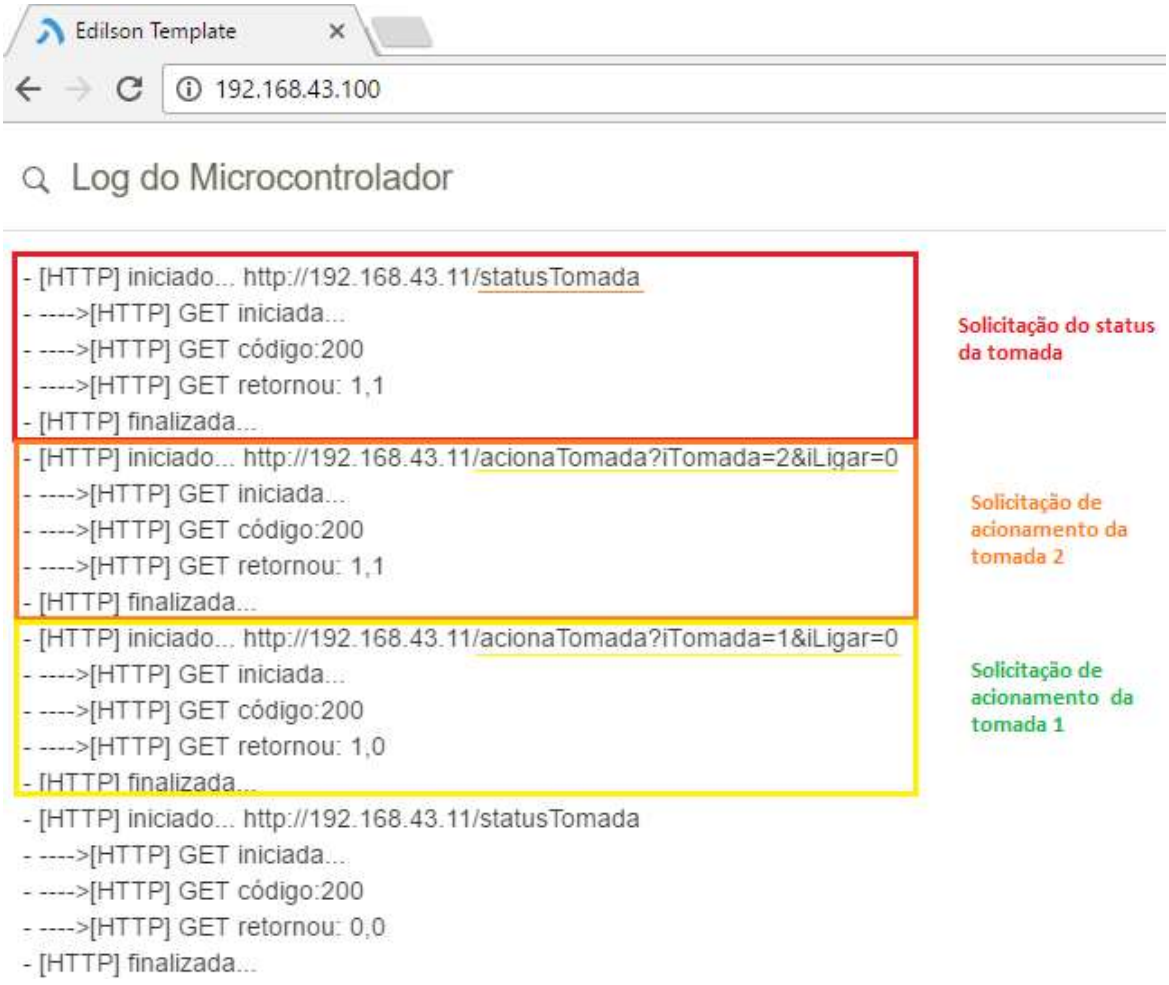

Figura 4.7 - Tela de Log na página WEB. Fonte: Autor.

O Log da página WEB, foi excelente para o teste proposto, onde foi possível observar a requisição HTTP concluindo seu ciclo de início e fim sem problemas. Através do log, é possível observar cada requisição, como por exemplo a de acionamento da tomada 2 (*http://192.168.43.11/acionaTomada?iTomada=2&iLigar=0*), chamando a requisição "*acionaTomada*" e logo após o caractere "?", vem os argumentos onde o argumento "*iTomada*", indica qual tomada ligar/desligar e o argumento "*iLigar*" indica se é para ligar ou desligar a tomada em questão, sendo que o valor "0" liga o relé e o valor "*1*" desliga o relé assim ligando/desligando a tomada.

4.2.3. TESTE NO SERVIDOR ESP8266WEBFITALED

Assim como os outros servidores WEB já citados neste material, este servidor também recebe requisições HTTP enviado pela página WEB também que o código fonte já está disponível no microcontrolador.

O servidor responsável pelo acionamento da fita de LED, recebe 4 argumentos sendo eles: "*R, G, B, Efeito"*. Os argumentos *R, G, B* são responsáveis pelas cores *R* = vermelho, *G* = verde, *B* = azul e o argumento *Efeito* é o responsável por informar ao micro controlador qual foi o efeito escolhido na página WEB.

A primeira coisa a ser verificada é se a fita de LED está conectada ao microcontrolador. Na sequência, é verificado pelo microcontrolador se a requisição é de acionamento da fita de LED "*corFitaLed*", caso positivo, é verificado se foi enviado na requisição 4 parâmetros, lendo-os e enviando a fita de LED. Caso contrário a qualquer destas afirmações devolve a página uma mensagem de erro.

Para um melhor entendimento do teste a ser realizado, foi feito um diagrame de teste na .Figura 4.8

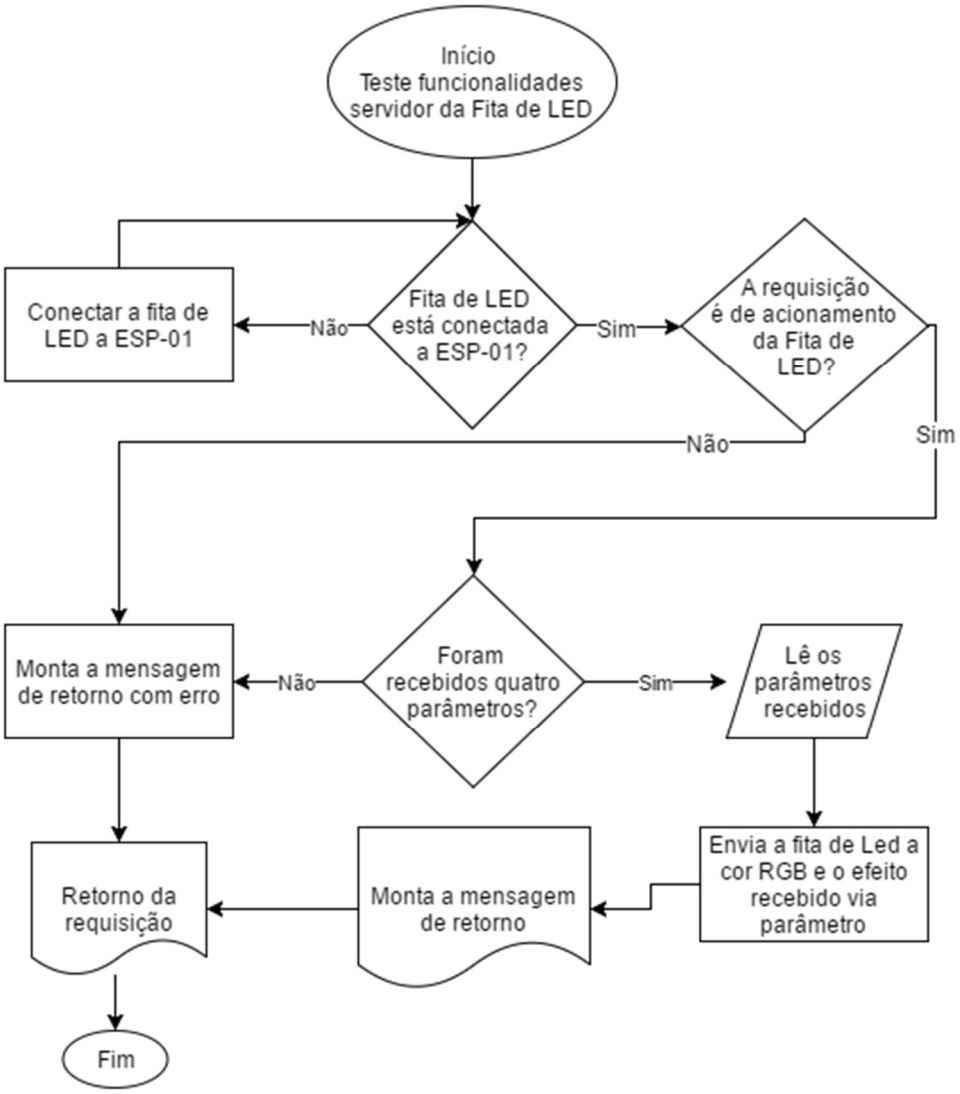

Figura 4.8 - Diagrama dos testes servidor da Fita de LED. Fonte: Autor

Os efeitos “1, 2, 3”, tem a necessidade da paleta de cores de 127 cores, ao clicar na cor desejada é enviado uma solicitação ao servidor WEB como é observado a seguir: “*http://192.168.43.13/corFitaLed?r=102&g=102&b=255&e=1*”, requisição é a “corFitaLed”, e os parâmetros é separado pelo caractere especial “&” como também já foi explicado neste material. Os efeitos “4, 5, 6”, são efeitos randômicos de cores randômicas controlados pelo microcontrolador.

Todos os efeitos foram testados e estão funcionais sem erros. A troca de cores e efeitos foram feitos num intervalo de 30 minutos para cada cor/efeito.

4.3.TERCEIRO CENÁRIO

Finalmente chega a vez de testar o bloco de monitoração de temperatura e umidade. Este é responsável por atender as requisições de temperatura e umidade da página WEB. O servidor onde está instalado o sensor DHT11 é o ESP8266WebServer. Para comparar os valores obtidos pelo sensor DHT11, é utilizado um outro sensor modelo WH2 fabricado pela empresa Aercus Instruments, que mostra em seu visor a temperatura e umidade do ambiente. Será utilizado estes dois sensores e com intuito de realizar um comparativo entre os valores obtidos. A seguir é mostrado uma imagem deste sensor.

Figura 4.9 - Sensor temperatura e umidade WH2. Fonte: http://www.aercusinstruments.com/aercus-instruments-ws1173-desktop-weather-station/

Foram realizadas 32 medições no período de 3 dias numa alternância de 1 hora como pode ser observado a seguir.

Sensor DHT11 Temperatura ºC	Sensor DHT11 Umidade %	Sensor WH2 Temperatura ºC	Sensor WH2 Umidade %
24	47	26,1	63
25	45	26,2	63
25	46	26,9	65
25	45	27,3	62
25	50	27,8	63
24	47	27,1	61
25	46	25,3	64
24	52	25,3	64
24	47	25,3	63
25	45	26,1	60
25	46	27,1	61
25	45	26,4	62
26	44	26,3	62
25	42	27	58
26	41	27,1	55
28	38	27,1	56
27	41	27,4	58
27	43	27,4	59
28	36	27,6	53
28	36	27,7	53
28	37	27,8	53
28	36	28	52
28	38	28	53
28	37	28,1	52
28	39	28,1	50
29	38	28,1	50
28	36	27,9	48
29	38	28,1	50
28	40	28,2	49
28	44	28,1	52
29	36	28,1	48

28	35	27,9	47

Tabela 4.4 – Medições dos sensores DHT11 e WH2. Fonte: Autor.

Com os dados obtidos na Tabela 4.4, foi plotado um gráfico para melhor visualização das medições.

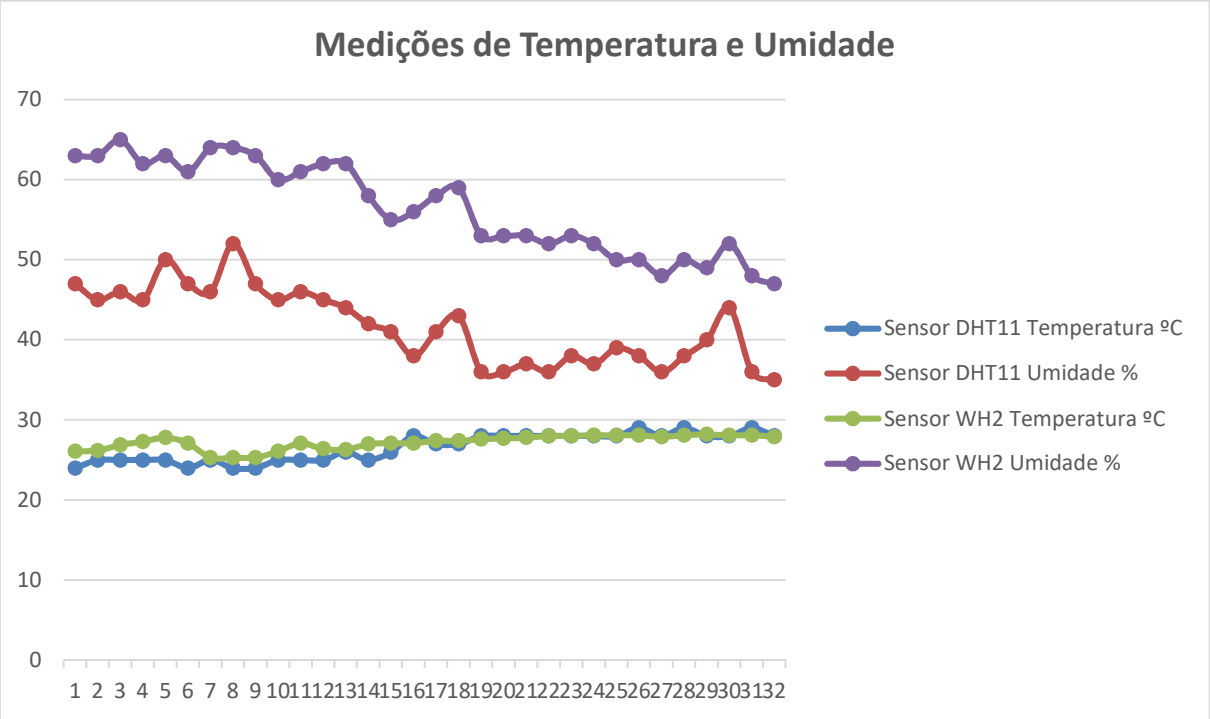

Figura 4.10 - Gráfico de medições dos sensores DHT11 e WH2. Fonte: Autor

Ainda com os dados obtidos nos sensores, é possível fazer uma média das variações de temperatura e umidade com a seguinte formula:

$$Variação = \frac{Valor\ WH2}{Valor\ DHT11} - 1$$

Com apoio da formula “Variação”, foi montado a seguinte tabela:

Variação Temperatura	Variação Umidade
8,75%	34,04%
4,80%	40,00%
7,60%	41,30%
9,20%	37,78%
11,20%	26,00%
12,92%	29,79%
1,20%	39,13%
5,42%	23,08%
5,42%	34,04%
4,40%	33,33%
8,40%	32,61%
5,60%	37,78%
1,15%	40,91%
8,00%	38,10%
4,23%	34,15%
-3,21%	47,37%
1,48%	41,46%
1,48%	37,21%
-1,43%	47,22%
-1,07%	47,22%
-0,71%	43,24%
0,00%	44,44%
0,00%	39,47%
0,36%	40,54%
0,36%	28,21%
-3,10%	31,58%
-0,36%	33,33%
-3,10%	31,58%
0,71%	22,50%
0,36%	18,18%
-3,10%	33,33%
-0,36%	34,29%

Tabela 4.5 - Variação da temperatura e umidade. Fonte: Autor.

Com os dados calculados da variação da temperatura e umidade temos o esboço do seguinte gráfico.

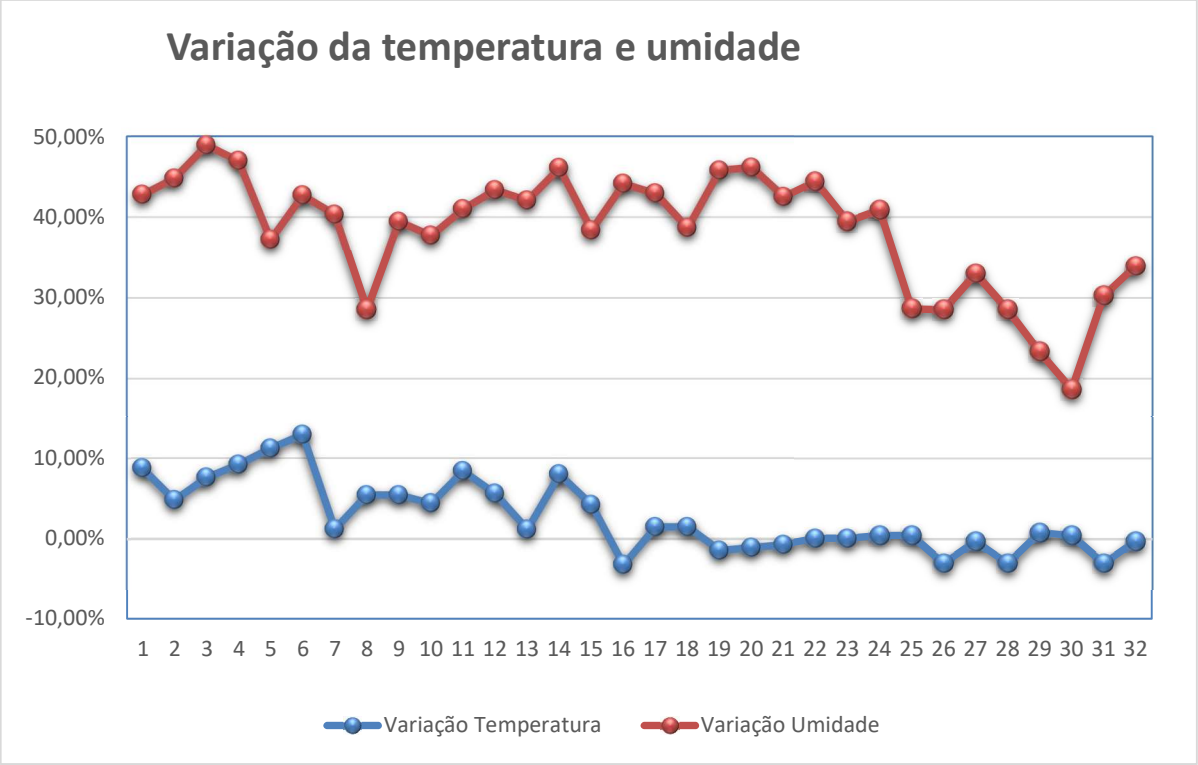

Figura 4.11 - Gráfico da variação de temperatura e umidade. Fonte: Autor.

Agora, para achar o percentual de variação (Média) entre os sensores, é utilizado a fórmula a seguir:

$$Média \quad \sum_{i=0}^{n} \frac{Variação}{n}$$

A média obtida para a variação da temperatura foi de 2,71%, e da umidade foi de 35,73%. A variação da temperatura está de acordo com os valores informados pelo fabricante dos dois sensores, já o sensor de umidade está muito acima dos valores informados pelo fabricante. Como foi descrito no item 2.5 deste material, o sensor DHT11pode variar a umidade ± 5% enquanto a de temperatura, pode variar ± 2ºC, o que não ocorreu nos testes realizados. A umidade teve uma variação muito grande, enquanto a temperatura teve uma variação aceitável.

CAPÍTULO 5 CONCLUSÃO

5.1.CONCLUSÕES

Este projeto de automação residencial, atingiu seu objetivo geral, permitindo através de uma rede sem fio e de uma forma interativa, controlar diversos dispositivos e ainda monitorar a temperatura e umidade de um ambiente residencial, atingindo assim o propósito, que é o trazer bem-estar, conforto, gestão energética, entre outros.

A página WEB permitiu de forma prática e interativa, a comunicação com os diversos dispositivos, permitindo ao usuário controles já explicados neste material.

Os servidores WEB apresentaram um bom comportamento funcional, atingindo assim o propósito de acionamento das diversas funcionalidade. Porém, como pode ser observado no item 4.3, o sensor DHT11, não se comportou como especificado pelo fabricante nas leituras de umidade.

Durante as pesquisas referentes ao microcontrolador ESP8266 modelos ESP-01 e ESP12-E, foi possível constatar que existe pouco material no ambiente acadêmico referenciando este microcontrolador. Portanto, este trabalho poderá vir a ser utilizado como referencial teórico para trabalhos futuros.

5.2.TRABALHOS FUTUROS

Visando o melhoramento do sistema proposto, descrevo algumas ideias para desenvolvimento futuro.

Desenvolver um aplicativo para as plataformas IOS e Android, visando uma melhor comodidade ao usuário.

Permitir o acesso remoto através da internet controlando assim as diversas funcionalidades a distância.

Desenvolver funcionalidades com outros atuadores e sensores, com intuito de melhorar ainda mais o conforto de uma residência.

Desenvolver um protótipo de fácil instalação, permitindo assim que qualquer usuário que tenha conhecimento básico em elétrica possa instalar.

CAPÍTULO 6 REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, G. C. D. repositorio uniceub. **Uniceub**, 22 nov. 2010. Disponivel em: <http://www.repositorio.uniceub.br/bitstream/123456789/3378/3/20516217.pdf>. Acesso em: 18 ago. 2016.

ARAUJO, M. AFRIKA. **IoT – Internet das Coisas**. Disponivel em: <http://www.afrikatec.com.br/iot-internet-das-coisas-motivacao-beneficios-e-seguranca/>. Acesso em: 20 ago. 2016.

ARDUINO.CC. Arduino Software (IDE). **Arduino.cc**, 2016. Disponivel em: <https://www.arduino.cc/en/guide/environment#>. Acesso em: 20 set. 2016.

BECKLER, M. L. microcontrollers/rgb_led/. **https:** //www.mbeckler.org/, 2009. Disponivel em: <https://www.mbeckler.org/microcontrollers/rgb_led/>. Acesso em: 05 out. 2016.

BELENZINHO, H. Território Livre. **Hacklest**, 2015. Disponivel em: <https://hackleste.wordpress.com/arduino/>. Acesso em: 05 out. 2016.

BRAGA, N. **RELÉS CIRCUITOS E APLICAÇÕES**. 1. ed. SÃO PAULO: INSTITUTO NEWTON BRAGA, v. 1, 2012.

CANARIM, P. O nascimento da internet. **webinsider**, 07 abr. 2012. Disponivel em: <https://webinsider.com.br/2012/04/07/o-nascimento-da-internet-comecou-na-2a-guerra-mundial/>. Acesso em: 22 ago. 2016.

CIPOLI, P. Canal Tech. **O que é um SoC**. Disponivel em: <https://canaltech.com.br/o-que-e/hardware/O-que-e-um-SoC/>. Acesso em: 01 out. 2016.

CURVELO, A. Embarcados. **Apresentando o módulo ESP8266**, 2015. Disponivel em: <http://www.embarcados.com.br/modulo-esp8266/>. Acesso em: 01 out. 2016.

ELECTRONICA-PT. electronica-pt. Disponivel em: <http://www.electronica-pt.com/ldr>. Acesso em: 02 nov. 2016.

ESPRESSIF. ESP8266. **resources**. Disponivel em: <http://www.espressif.com/en/products/hardware/esp8266ex/resources>. Acesso em: 10 jul. 2016.

KOLBAN, N. **Kolban's Book on the ESP8266**. Texas: [s.n.], 2015. Disponivel em: <http://neilkolban.com/tech/esp8266/>. Acesso em: 18 jun. 2016.

KUROSE, J. F.; ROSS, K. W. **Rede de Computadores e a Internet:** Uma Abordagem Top-Down. Tradução de Opportunity Translation. 5ª. ed. São Paulo: [s.n.], 2010.

MARTINS, A. A evolução da internet. **enfoquenet**, 12 mar. 2012. Disponivel em: <http://www.enfoquenet.com.br/e-marketing/a-evolucao-da-internet/>. Acesso em: 22 ago. 2016.

MINATEL, P. ESP8266 + Sming. **http:** //pedrominatel.com.br/, 2016. Disponivel em: <http://pedrominatel.com.br/esp8266/sming-um-framework-para-esp8266/>. Acesso em: 10 out. 2016.

MURATORI, J. R.; DAL BÓ, P. H. **Automação Residencial Conceitos e Aplicações**. 1ª. ed. Belo Horizonte: Educere, v. 1, 2014.

RAPPAPORT, T. S. **Comunicação sem fio:** princípios e práticas. Tradução de Daniel Vieira. 2ª. ed. São Paulo: Pearson Education do Brasil, 2009.

REIS, F. D. Como funciona um LED RGB, São Paulo, 13 jan. 2016. Disponivel em: <http://www.bosontreinamentos.com.br/eletronica/curso-de-eletronica/curso-de-eletronica-como-funciona-um-led-rgb/>. Acesso em: 30 set. 2016.

SAS. **Internet of Things (IoT)**, 2013. Disponivel em: <http://www.sas.com/pt_br/insights/big-data/internet-das-coisas.html>. Acesso em: 15 ago. 2016.

SOMBRA, L. G. Repositório Uniceub. **Uniceub**, 20 jun. 2016. Disponivel em: <http://www.repositorio.uniceub.br/bitstream/235/8693/1/21114229.pdf>. Acesso em: 05 ago. 2016.

TANENBAUM, A. S.; WETHERALL, D. J. **Redes de Computadores**. Tradução de Daniel Vieira. 5ª. ed. São Paulo: Pearson Education - Br, 2011.

TECHNOLOGIES, S. www.sunron.com, 20 jun. 2012. Disponivel em: <http://robocraft.ru/files/datasheet/DHT11.pdf>. Acesso em: 30 set. 2016.

VOLTIMUM. **Automação residencial:** histórico, definições e conceitos., 2014. Disponivel em: <http://static2.voltimum.com/sites/www.voltimum.com.br/files/pdflibrary/04_automacao_residencial1.pdf>. Acesso em: 2016 ago. 18.

ANEXO A CÒDIGO FONTE ESP8266WebServer

Disponibilizo aqui o código fonte em linguagem de programação C/C++, escritos na interface Arduino IDE, responsável pelo funcionamento do microcontrolador ESP8266 responsável pelo servidor principal ESp8266WebServer.

```
/*
  Centro Universitário de Brasília
  Engenharia da Computação - FATECS
  Automação Residencial para monitoramento de Temperatura,
  umidade e controle de iluminação utilizando ESP8266
  Edilson Leal Santos - RA 20766608
  Código de controle WEB Server Principal
*/

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <SPI.h>
#include <SD.h>
#include <DHT.h>
#include <ESP8266HTTPClient.h>

#define dhtPIN 2

DHT dht(dhtPIN, DHT11);

const char* ssid = "TCCEdilson";
const char* password = "12345678";
const char* host = "ESP8266WebSite";

const String IPFitaLed = "192.168.43.13";
const String IPTomada = "192.168.43.11";
const String IPLampada = "192.168.43.10";
const String IPWebServer = "192.168.43.100";
```

```
String strLog = "";

MDNSResponder mdns;
ESP8266WebServer server(80);

static bool hasSD = false;
File uploadFile;

void returnOK() {
  server.send(200, "text/plain", "");
  Serial.println("Retorno OK.");
}

void returnMsgOK(String msg) {
  server.send(200, "text/plain", msg + retornaLogPagina () + "\r\n");
  Serial.print("Msg de retorno OK: ");
  Serial.println(msg);
  limpaLogPagina();
}

void returnFail(String msg) {
  server.send(500, "text/plain", msg + retornaLogPagina () + "\r\n");
  Serial.println("Retorno com falha!");
  limpaLogPagina();
}

bool loadFromSdCard(String path){
  String dataType = "text/plain";
  if(path.endsWith("/")) path += "index.htm";

  if(path.endsWith(".src")) path = path.substring(0, path.lastIndexOf("."));
  else if(path.endsWith(".htm")) dataType = "text/html";
  else if(path.endsWith(".css")) dataType = "text/css";
  else if(path.endsWith(".js")) dataType = "application/javascript";
  else if(path.endsWith(".png")) dataType = "image/png";
  else if(path.endsWith(".gif")) dataType = "image/gif";
  else if(path.endsWith(".jpg")) dataType = "image/jpeg";
  else if(path.endsWith(".ico")) dataType = "image/x-icon";
  else if(path.endsWith(".xml")) dataType = "text/xml";
```

```
  else if(path.endsWith(".pdf")) dataType = "application/pdf";
  else if(path.endsWith(".zip")) dataType = "application/zip";

  File dataFile = SD.open(path.c_str());
  if(dataFile.isDirectory()){
    path += "/index.htm";
    dataType = "text/html";
    dataFile = SD.open(path.c_str());
  }

  if (!dataFile){
    Serial.println("***Erro no carregamento do cartão SD!");
    return false;
  }

  if (server.hasArg("download")) dataType = "application/octet-stream";

  if (server.streamFile(dataFile, dataType) != dataFile.size()) {
    Serial.println("Enviado menos dados do que o esperado!");
    Serial.println(path);
  }

  dataFile.close();
  Serial.print("Carregando do Cartão SD: ");
  Serial.println(path);
  return true;
}

void handleNotFound(){
  if(hasSD && loadFromSdCard(server.uri())) return;
  String message = "SDCARD não detectado\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET)?"GET":"POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i=0; i<server.args(); i++){
    message += " NAME:"+server.argName(i) + "\n VALUE:" + server.arg(i) + "\n";
```

```
  }
  server.send(404, "text/plain", message);
  Serial.print(message);
  Serial.println("Fonte não manuseada!");
}

void setup(void){

  Serial.begin(115200); //Define a porta serial do console
  Serial.setDebugOutput(true); //Permite a visualização do console a partir de outra interface
  Serial.print("\n");

  WiFi.hostname(host); //Define o Nome do Dispositivo

  WiFi.begin(ssid, password); //Responsável por realizar a conexão WiFi ao Roteador

  Serial.print("Conectando a ");
  Serial.println(ssid);

  logPagina("Conectando a Rede " + (String) ssid);

  //Aguarda a conexão WiFi ao Roteador
  uint8_t i = 0;
  while (WiFi.status() != WL_CONNECTED && i++ < 20) {//Espera 10 segundo
    delay(500);
  }

  if(i == 21){ //Caso a conexão WiFi não se realize, informa ao console.
    Serial.print("***Conexão não realizada a rede ");
    Serial.println(ssid);

    logPagina("***Conexão não realizada a rede " + (String) ssid);

    while(1) delay(500);
  }

  Serial.print("Conectado! Endereço IP: ");
  Serial.println(WiFi.localIP()); //Retorna o Endereço IP.

  logPagina("Conectado! Endereço IP: " + IPWebServer);
```

```
if (MDNS.begin(host, WiFi.localIP())) { //Configura o Servidor DNS local
  MDNS.addService("http", "tcp", 80);
  Serial.println("MDNS foi iniciado");
  Serial.print("Voce pode conectar utilizando http://");
  Serial.print(host);
  Serial.println(".local");
  MDNS.update();
}
//Responsável por atender requisições HTTP.
server.on("/corFitaLed", HTTP_GET, corFitaLed);       //Fita de led
server.on("/tempUmi", HTTP_GET, tempUmi);             //Temperatura e Umidade
server.on("/acionaLampada", HTTP_GET, acionaLampada); //Acionamento da Lampada
server.on("/statusLampada", HTTP_GET, statusLampada); //Status da Lamada
server.on("/acionaTomada", HTTP_GET, acionaTomada);   //Acionamento das Tomadas
server.on("/statusTomada", HTTP_GET, statusTomada);   //Status das tomadas
server.on("/logServer", HTTP_GET, logServer);   //log do Servidor

server.onNotFound(handleNotFound); //Reporta erro caso não encontre a requisição HTTP

server.begin(); //Inicia o servidor HTTP
Serial.println("Servidor HTTP Iniciado...");

logPagina("Servidor HTTP Iniciado...");

Serial.println("Inicializando o SD card...");

if (SD.begin(SS)){ //Inicia a comunicação com o SD Card
   Serial.println("   SD Card inicializado.");
    logPagina("SD Card inicializado...");
   hasSD = true;
}else{
   Serial.println("***SD Card não inicializado ou ausente.");
   logPagina("***SD Card não inicializado ou ausente...");
}

dht.begin(); //Inicia a biblioteca do Sensor DHT11(Temperatura e Umidade)
Serial.println("Sensor DHT11 iniciado.");
logPagina("Sensor DHT11 iniciado...");
```

```
}

/*Função que retorna a Temperatura e Umidade do sensor DHT11 a Página Web*/
void tempUmi() {

 float t = random(0, 100) ;
 float h = random(0, 100) ;
 int iCt = 0;
 String RetTempUmi = "";

 Serial.println("Lendo sensor DHT11");

 logPagina("[HTTP] iniciado... http://" + IPWebServer + "/tempUmi");
 logPagina("--->Ler sernsor DHT11");

 t = dht.readTemperature();  //Busca a temperatura em ºC
 Serial.print("--->Valor de T:" + (String) t);

 h = dht.readHumidity();     //Busca o percentual da umidade do AR
 Serial.println("  Valor de H:" + (String) h);

 delay(50);

 while (t > 100 && h > 100 && iCt < 16){
  t = dht.readTemperature();  //Busca a temperatura em ºC
  h = dht.readHumidity();     //Busca o percentual da umidade do AR

  Serial.println("--->***Falha ao lêr o sensor DHT11 tentativa: " + (String) iCt);
  logPagina("--->***Falha ao lêr o sensor DHT11");
  logPagina("--->***Tentando novamente. Tentativa: " + (String) iCt);

  iCt++;
  delay(50);
 }

 if (t < 1000){

  RetTempUmi = ((String) t) + ";" + ((String) h);

  logPagina("--->Valor Temperatura: " + (String) t + "ºC umidade: " + (String) h + "%");
```

```
    Serial.println("--->Temp. = " + (String) t + "ºC Umi. = " +  (String) h + "%");
  } else {

    RetTempUmi = "-;-";
    logPagina("--->***Erro ao recuperar a temperatura e umidade.");

    Serial.println("--->***Erro ao recuperar a temperatura e umidade.");
  }

  logPagina("[HTTP] finalizada...");

  returnMsgOK(RetTempUmi);

  Serial.println("Solicitação finalizada...");

}

/*Função responsável por controlar a fita de LED roteando a
  requisição ao Microcontrolador responsável por tal feito*/
void corFitaLed() {

   String message = "";

   if(server.args() != 4) { //Verifica se foi passado 4 argumentos para o controle da fita.
    Serial.print ("***corFitaLed <> 4 argumentos");

   for (uint8_t i=0; i<server.args(); i++){
     message += " Parametro:"+server.argName(i) + " Argumento:" + server.arg(i) + "\n";
    }

    Serial.println (message);

    logPagina("***corFitaLed <> 4 argumentos");
    logPagina("*** -> " + message);

    return returnFail("BAD ARGS"); //Retorna erro a página web
  }

  uint8_t cor[] = {255,255,255};
```

```
  //Recebe os argumentos contendo as cores no formato RGB
  cor[0] = (uint8_t) server.arg(0).toInt(); //R
  cor[1] = (uint8_t) server.arg(1).toInt(); //G
  cor[2] = (uint8_t) server.arg(2).toInt(); //B

  int efeito = server.arg(3).toInt(); //recebe o argumento contendo o efeito

  //Realiza a requisição HTTP ao controlador responsável por tal feito passando os argumentos
  requisicaoHTTP ("http:///" + IPFitaLed + "/corFitaLed?r="
+cor[0]+"&g="+cor[1]+"&b="+cor[2]+"&e="+efeito);

}

/*Função responsável por controlar as Tomadas roteando a
  requisição ao Microcontrolador responsável por tal feito*/
void acionaTomada() {

  String message = "";

  if(server.args() != 2) { //Verifica se foi passado 2 argumentos.
    Serial.print ("***acionaTomada <> 2 argumentos");

    for (uint8_t i=0; i<server.args(); i++){
      message += " Parametro:"+server.argName(i) + " Argumento:" + server.arg(i) + "\n";
    }

    Serial.println (message);

    logPagina("***acionaTomada <> 2 argumentos");
    logPagina("*** -> " + message);


    return returnFail("BAD ARGS"); //Retorna erro a página web
  }

  //Recebe os argumentos contendo a tomada(1,2) e seu acionamento (0,1)
  int iTomada = server.arg(0).toInt();  //Tomada (1,2)
  int iLigar = server.arg(1).toInt();   //Acionamento (0,1)
```

```
  //Realiza a requisição HTTP ao controlador responsável por tal feito passando os argumentos
  requisicaoHTTP ("http://" + IPTomada + "/acionaTomada?iTomada=" + iTomada + "&iLigar=" + iLigar);

}

/*Função responsável por retornar o Status das tomadas roteando a
  requisição ao Microcontrolador responsável por tal feito*/
void statusTomada() {

  //Realiza a requisição HTTP ao controlador responsável por tal feito passando os argumentos
  requisicaoHTTP ("http://" + IPTomada + "/statusTomada");

}

/*Função responsável por controlar a lampada roteando a
  requisição ao Microcontrolador responsável por tal feito*/
void acionaLampada() {

  String message = "";

  if(server.args() != 1) { //Verifica se foi passado 1 argumento.
    Serial.print ("***acionaLampada <> 1 argumento");

    for (uint8_t i=0; i<server.args(); i++){
      message += " Parametro:"+server.argName(i) + " Argumento:" + server.arg(i) + "\n";
    }

    Serial.println (message);

    logPagina("***acionaLampada <> 1 argumento");
    logPagina("*** -> " + message);

    return returnFail("BAD ARGS"); //Retorna erro a página web
  }

  //Recebe o argumento contendo o acionamento (0,1)
  int iLigar = server.arg(0).toInt();

  //Realiza a requisição HTTP ao controlador responsável por tal feito passando os argumentos
  requisicaoHTTP ("http://" + IPLampada + "/acionaLampada?iLigar=" +iLigar);
```

```
}

/*Função responsável por retornar o Status da lampada roteando a
  requisição ao Microcontrolador responsável por tal feito*/
void statusLampada() {

   //Realiza a requisição HTTP ao controlador responsável por tal feito passando os argumentos
  requisicaoHTTP ("http://" + IPLampada + "/statusLampada");

}

/*Função responsável por requisição HTTP*/
void requisicaoHTTP (String sRequisicao){

  String sRetorno = "";

  HTTPClient http;

  Serial.print("[HTTP] iniciado...\n");
  Serial.println (sRequisicao);

  logPagina("[HTTP] iniciado... " + (String) sRequisicao);

  http.begin(sRequisicao); //HTTP

  Serial.print("      [HTTP] GET...\n");
  logPagina("---->[HTTP] GET iniciada... ");
  // Inicia a conecção e envia a requisção do cabeçalho HTTP
  int httpCode = http.GET();

  // Código http será negado em caso de erro
  if(httpCode > 0) {
     //Cabeçalho HTTP foi enviado e cabeçalho de resposta do servidor tem sido tratado
     Serial.printf("     [HTTP] GET... code: %d\n", httpCode);
     logPagina("---->[HTTP] GET código:" + (String) httpCode);

     // arquivo encontrado no servidor
     if(httpCode == HTTP_CODE_OK) {
        sRetorno = http.getString();
```

```
        Serial.println("     " + sRetorno);
        String sRet = sRetorno;
        sRet.replace(';',',');
        logPagina("---->[HTTP] GET retornou: " + sRet);
      }
  } else {
    sRetorno = "      ***[HTTP] GET... falou, erro: %s\n", http.errorToString(httpCode).c_str();
    logPagina("---->" + sRetorno);
  }

  http.end(); //Finaliza a conexão HTTP
  logPagina("[HTTP] finalizada...");
  if(httpCode > 0) {

    returnMsgOK(sRetorno); //retorno com Sucesso
  } else {

    returnFail(sRetorno); //retorno com Erro

  }

  Serial.println(sRetorno);

}

void loop(void){
  //Escuta possíveis requisições HTTP .
  server.handleClient();
}

void logPagina (String txt){

  if(txt!=""){
    strLog += ";" + txt;
  }

}

void limpaLogPagina (){
```

```
  strLog = "";

}

String retornaLogPagina (){

  return strLog;

}

void logServer(){

  //Retorna o solicitado a Página WEB.
 returnMsgOK("");

}
```

ANEXO B FONTE ESP8266WebLampada

Disponibilizo aqui o código fonte em linguagem de programação C/C++, escritos na interface Arduino IDE, responsável pelo funcionamento do microcontrolador ESP8266 responsável pelo servidor da lâmpada ESp8266WebLampada.

```
/*
  Centro Universitário de Brasília
  Engenharia da Computação - FATECS
  Automação Residencial para monitoramento de Temperatura,
  umidade e controle de iluminação uilizando ESP8266
  Edilson Leal Santos - RA 20766608
  Código de controle via WiFi da lâmpada com status
  IPLampada = "192.168.43.10";
*/
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

#define PinLDR 0
#define PinLamp 2

const char* ssid = "TCCEdilson";
const char* password = "12345678";
const char* host = "ESP8266WebLampada";

ESP8266WebServer server(80);

void returnOK() {
  server.send(200, "text/plain", "");
  Serial.println("Retorno OK.");
}

void returnMsgOK(String msg) {
  server.send(200, "text/plain", msg + "\r\n");
```

```
  Serial.print("Msg de retorno OK: ");
  Serial.println(msg);
}

void returnFail(String msg) {
  server.send(500, "text/plain", msg + "\r\n");
  Serial.println("Retorno com falha!");
}

void handleRoot() {
  server.send(200, "text/plain", "hello from esp8266!");
}

void handleNotFound(){

  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET)?"GET":"POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i=0; i<server.args(); i++){
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
}

void setup(void){

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.print("\n");

  WiFi.hostname(host);

  WiFi.begin(ssid, password);
  Serial.println("");
```

```
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Conectando a ");
  Serial.println(ssid);
  Serial.print("Endereço IP: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin(host)) {
    Serial.println("MDNS responder started");
  }

  server.on("/", returnOK);

  server.on("/acionaLampada", HTTP_GET, acionaLampada);
  server.on("/statusLampada", HTTP_GET, statusLampada);

  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("Servidor HTTP iniciado...");

  pinMode(PinLDR, INPUT);
  pinMode(PinLamp, OUTPUT);
  digitalWrite(PinLamp,HIGH);
}

void loop(void){
  server.handleClient();
}

void acionaLampada() {

   String message = "";

  if(server.args() != 1) {
    Serial.print ("acionaTomada <> 1 argumento");
```

```
  for (uint8_t i=0; i<server.args(); i++){
    message += " Parametro:"+server.argName(i) + " Argumento:" + server.arg(i) + "\n";
  }

  Serial.println (message);

  return returnFail("BAD ARGS");
 }

 int iLigado = server.arg(0).toInt();
 int iLDR = digitalRead(PinLDR);
 digitalWrite(PinLamp,!iLigado);

 delay(50);

 message = ((String) digitalRead(PinLamp)) + ";" + ((String) digitalRead(PinLDR));

 returnMsgOK(message);

 Serial.print ("Funcao acionaLampada Rele1:");
 Serial.print (digitalRead(PinLamp));
 Serial.print (" ,LDR: ");
 Serial.println (digitalRead(PinLDR));
}

void statusLampada() {

 String message = ((String) digitalRead(PinLamp)) + ";" + ((String) digitalRead(PinLDR));

 returnMsgOK(message);

 Serial.print ("Funcao statusLampada Rele1:");
 Serial.print (digitalRead(PinLamp));
 Serial.print (" ,LDR: ");
 Serial.println (digitalRead(PinLDR));
}
```

ANEXO C FONTE ESP8266WebTomada

Disponibilizo aqui o código fonte em linguagem de programação C/C++, escritos na interface Arduino IDE, responsável pelo funcionamento do microcontrolador ESP8266 responsável pelo servidor da lâmpada ESp8266WebTomada.

```
/*
  Centro Universitário de Brasília
  Engenharia da Computação - FATECS
  Automação Residencial para monitoramento de Temprestura,
  umidade e controle de iluminação uilizando ESP8266
  Edilson Leal Santos - RA 20766608
  Código de controle via WiFi com duas tomadas elétrica
  IPTomada = "192.168.43.11";
*/

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

#define PINTom1 0
#define PINTom2 2

const char* ssid = "TCCEdilson";
const char* password = "12345678";
const char* host = "ESP8266WebTomada";

ESP8266WebServer server(80);

void returnOK() {
  server.send(200, "text/plain", "");
  Serial.println("Retorno OK.");
}

void returnMsgOK(String msg) {
```

```
  server.send(200, "text/plain", msg + "\r\n");
  Serial.print("Msg de retorno OK: ");
  Serial.println(msg);
}

void returnFail(String msg) {
  server.send(500, "text/plain", msg + "\r\n");
  Serial.println("Retorno com falha!");
}

void handleRoot() {
  server.send(200, "text/plain", "hello from esp8266!");
}

void handleNotFound(){

  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET)?"GET":"POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i=0; i<server.args(); i++){
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);

}

void setup(void){

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.print("\n");

  WiFi.hostname(host);

  WiFi.begin(ssid, password);
```

```
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Conectando a ");
  Serial.println(ssid);
  Serial.print("Endereço IP: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin(host)) {
    Serial.println("MDNS responder started");
  }

  server.on("/", returnOK);

  server.on("/acionaTomada", HTTP_GET, acionaTomada);
  server.on("/statusTomada", HTTP_GET, statusTomada);

  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("Servidor HTTP iniciado...");

  pinMode(PINTom1, OUTPUT);
  pinMode(PINTom2, OUTPUT);
  digitalWrite(PINTom1,HIGH);
  digitalWrite(PINTom2,HIGH);
}

void loop(void){
  server.handleClient();
}

void acionaTomada() {

   String message = "";
```

```
  if(server.args() != 2) {
    Serial.print ("acionaTomada <> 2 argumentos");

    for (uint8_t i=0; i<server.args(); i++){
      message += " Parametro:"+server.argName(i) + " Argumento:" + server.arg(i) + "\n";
    }

    Serial.println (message);

    return returnFail("BAD ARGS");
  }

  int iTomada = server.arg(0).toInt();
  int iLigado = server.arg(1).toInt();

  if (iTomada ==1){
    digitalWrite(PINTom1,!digitalRead(PINTom1));
  }else{
    digitalWrite(PINTom2,!digitalRead(PINTom2));
  }
  delay(30);

  message = ((String) digitalRead(PINTom1)) + ";" + ((String) digitalRead(PINTom2));

  returnMsgOK(message);

  Serial.print ("Funcao statusTomada Rele1:");
  Serial.print (digitalRead(PINTom1));
  Serial.print (" ,Rele2: ");
  Serial.println (digitalRead(PINTom2));
}

void statusTomada() {

  String message = ((String) digitalRead(PINTom1)) + ";" + ((String) digitalRead(PINTom2));

  returnMsgOK(message);

  Serial.print ("Funcao statusTomada Rele1:");
```

```
Serial.print (digitalRead(PINTom1));
Serial.print (" ,Rele2: ");
Serial.println (digitalRead(PINTom2));
}
```

ANEXO D FONTE ESP8266FitaLed

Disponibilizo aqui o código fonte em linguagem de programação C/C++, escritos na interface Arduino IDE, responsável pelo funcionamento do microcontrolador ESP8266 responsável pelo servidor da fita de LED ESp8266WebFitaLed.

```
/*
  Centro Universitário de Brasília
  Engenharia da Computação - FATECS
  Automação Residencial para monitoramento de Temprestura,
  umidade e controle de iluminação uilizando ESP8266
  Edilson Leal Santos - RA 20766608
  Código de controle via WiFi da lâmpada com status
  IPFitaLed = "192.168.43.13";
*/

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <Adafruit_NeoPixel.h>

#define SetColorPIN 2
#define QtdLeds 10

Adafruit_NeoPixel strip = Adafruit_NeoPixel(QtdLeds, SetColorPIN, NEO_GRB + NEO_KHZ800);

const char* ssid = "TCCEdilson";
const char* password = "12345678";
const char* host = "ESP8266WebFitaLed";

uint8_t cor[] = {255,255,255};
int efeito = 0,ultEfeito=0;
uint16_t i=0, j=0, q = 0;


ESP8266WebServer server(80);
```

```
void returnOK() {
  server.send(200, "text/plain", "");
  Serial.println("Retorno OK.");
}


void returnFail(String msg) {
  server.send(500, "text/plain", msg + "\r\n");
  Serial.println("Retorno com falha!");
}

void handleRoot() {

  server.send(200, "text/plain", "hello from esp8266!");

}

void handleNotFound(){

  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET)?"GET":"POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i=0; i<server.args(); i++){
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);

}

void setup(void){

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.print("\n");
```

```
  WiFi.hostname(host);

  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Conectando a ");
  Serial.println(ssid);
  Serial.print("Endereço IP: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin(host)) {
    Serial.println("MDNS responder started");
  }

  strip.begin();
  strip.show(); // Initialize all pixels to 'off'


  server.on("/", returnOK);

  server.on("/corFitaLed", HTTP_GET, corFitaLed);

  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("Servidor HTTP iniciado...");
}

void loop(void){
  server.handleClient();
  defineEfeito(strip.Color(cor[0], cor[2], cor[1]),efeito,50);
}

void corFitaLed() {
```

```
Serial.print ("Argumentos: ");
Serial.println (server.args());

String message = "";

if(server.args() != 4) {
  Serial.print ("corFitaLed args <> 4");

  for (uint8_t i=0; i<server.args(); i++){
    message += " NAME:"+server.argName(i) + " VALUE:" + server.arg(i) + "\n";
  }

  Serial.println (message);

  return returnFail("BAD ARGS");
}

cor[0] = (uint8_t) server.arg(0).toInt();
cor[1] = (uint8_t) server.arg(1).toInt();
cor[2] = (uint8_t) server.arg(2).toInt();
efeito = server.arg(3).toInt();

Serial.print ( " R: ");
Serial.print (cor[0]);
Serial.print ( " G: ");
Serial.print ( cor[1]);
Serial.print ( " B: ");
Serial.println ( cor[2]);

Serial.print ( " teste: ");
Serial.println (strip.Color(cor[0], cor[2], cor[1]));

Serial.print ( " Argumento 4: ");
Serial.println (efeito);

//colorWipe(strip.Color(cor[0], cor[2], cor[1]));

returnOK();
delay(100);
```

```
  //defineEfeito(strip.Color(cor[0], cor[2], cor[1]),efeito,50);

}

void defineEfeito (uint32_t cor, int efeito,uint8_t espera){

  if (efeito !=ultEfeito){

    Serial.print ("efeito: ");
    Serial.println (efeito);
    ultEfeito = efeito;
  }

  switch(efeito){
    case 0: colorWipe(strip.Color(0, 0, 0), 0);    // Black/off
          break;
    case 1: colorWipe(cor, 50);
          break;
    case 2: theaterChase(cor, 50);
          break;
    case 3: strobo(cor, 50);
          break;
    case 4: rainbow(20);
          break;
    case 5: rainbowCycle(20);
          break;
    case 6: theaterChaseRainbow(50);
          break;
  }

}

void strobo(uint32_t c, uint8_t wait) {
  //for(uint16_t i=0; i<strip.numPixels(); i++) {

  if (i==strip.numPixels()-1){
    i=0;
  } else {
    i++;
```

```
  }
  colorWipe(c, 0);
  delay(wait);
  colorWipe(strip.Color(0, 0, 0), 0);
  delay(wait);

  //}
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
  }
}
void theaterChase(uint32_t c, uint8_t wait) {

  for (int j=0; j<10; j++) {  //do 10 cycles of chasing
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, c);    //turn every third pixel on
      }
      strip.show();

      delay(wait);

      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, 0);        //turn every third pixel off
      }
    }
  }
}

void rainbow(uint8_t wait) {
  //uint16_t i, j;

//  Serial.print ("Valor de j:");
//  Serial.print (j);
```

```
//  Serial.print (" Valor de i:");
//  Serial.println (i);
//
  if (i==strip.numPixels()&& j<=255){

     strip.show();
     delay(wait);
     j ++;
     i=0;

  } else if(j<=255){

    strip.setPixelColor(i, Wheel((i+j) & 255));
    i++;

  }else {

    i=j=0;
    delay(wait);
  }

// for(j=0; j<256; j++) {
//   for(i=0; i<strip.numPixels(); i++) {
//     strip.setPixelColor(i, Wheel((i+j) & 255));
//   }
//   strip.show();
//   delay(wait);
// }
}
// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
  }
```

```
}

//Theatre-style crawling lights with rainbow effect
void theaterChaseRainbow(uint8_t wait) {
  for (int j=0; j < 256; j++) {     // cycle all 256 colors in the wheel
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, Wheel( (i+j) % 255));    //turn every third pixel on
      }
      strip.show();

      delay(wait);

      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, 0);        //turn every third pixel off
      }
    }
  }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
  WheelPos -= 170;
  return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}
```
