



Universidade Federal
do Rio de Janeiro

Escola Politécnica

USO DO MICROCONTROLADOR ESP8266 PARA AUTOMAÇÃO RESIDENCIAL

Ricardo Rodrigues Oliveira

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Marcos Vicente de Brito
Moreira
Públio Macedo Lima

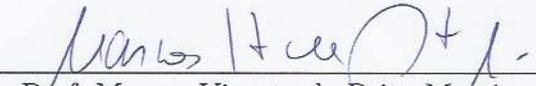
Rio de Janeiro
Fevereiro de 2017

USO DO MICROCONTROLADOR ESP8266 PARA AUTOMAÇÃO
RESIDENCIAL

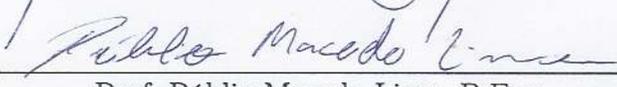
Ricardo Rodrigues Oliveira

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA
POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU
DE ENGENHEIRO DE AUTOMAÇÃO.

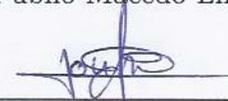
Examinado por:



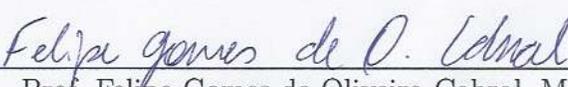
Prof. Marcos Vicente de Brito Moreira, D.Sc.



Prof. Públio Macedo Lima, B.Eng



Prof. Oumar Diene, D.Sc.



Prof. Felipe Gomes de Oliveira Cabral, M.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2017

Rodrigues Oliveira, Ricardo

Uso do Microcontrolador ESP8266 para automação residencial/Ricardo Rodrigues Oliveira. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2017.

XII, 42 p.: il.; 29, 7cm.

Orientadores: Marcos Vicente de Brito Moreira

Públio Macedo Lima

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Controle e Automação, 2017.

Referências Bibliográficas: p. 29 – 30.

1. ESP8266. 2. Automação Residencial. 3. Iluminação. I. de Brito Moreira, Marcos Vicente *et al.* II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Controle e Automação. III. Título.

*"When you feel down and out /
Sing a song, it'll make your
day" (White, Maurice)*

Agradecimentos

Gostaria de agradecer a Deus pela oportunidade e motivação de realizar esse projeto. Aos meus pais, Sergio e Lúcia, que me apoiaram e se esforçaram para que eu sempre pudesse ter todas as oportunidades que fossem possíveis, por terem sido meu suporte e exemplo ao longo da minha vida pessoal e acadêmica, junto com minhas irmãs Helena e Beatriz.

A todos os integrantes do LCA que de alguma forma me ajudaram nesse projeto, seja com materiais, testando o sistema, dando ideias e opiniões que muito vieram a acrescentar no mesmo. Em especial agradeço ao meu orientador Professor Marcos Vicente Moreira, pelas ideias para o projeto e todo seu auxílio e disponibilidade, por ter sido o professor que em sala me inspirou a realizar esse projeto e me fascinar pela área de automação. Também de forma especial agradeço ao meu coorientador, Públio Lima, por toda paciência, ensinamentos e auxílio prestados desde o primeiro minuto no qual ele assumiu esse posto, por cada conselho, seja ele prático ou teórico, que me foi fornecido nesse longo período de elaboração do projeto, pela grande ajuda na parte escrita também, principalmente no uso do $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$.

Agradeço também aos meus colegas de faculdade, que ao longo de todos esses anos foram parte ativa da minha vida acadêmica e junto comigo viveram as alegrias, angústias, tristezas e conquistas que a vida na universidade pode oferecer. Pelos trabalhos, risadas, grupos de estudos e apoio moral que me deram em todos esses anos, sou muito grato. De forma especial quero agradecer as minhas duas grandes amigas, Isabel de Oliveira Marques e Livia Chaves Paravidino, pela ajuda com todo o material necessário para meus estudos, pelos resumos, cadernos etc. Gostaria também de forma especial agradecer ao José Guilherme Monteiro, pela ideia de utilizar o ESP8266 para o projeto, que acabou por se tornar o coração do mesmo, e ao meu primo Gabriel Lira, pela ajuda com a soldagem de componentes, empréstimo da protobord e de ideias.

Também queria agradecer aos meus amigos de fora da faculdade, por todo apoio emocional ao longo dessa jornada, principalmente aqueles que estiveram ao meu lado nos momentos de dificuldade e desânimo. Em especial gostaria de agradecer ao meu amigo Julio Wei, pela ajuda com a parte técnica, pelo apoio, por toda a disponibilidade e pelos empréstimos de materiais.

Por último mas certamente não menos importante, agradeço à minha noiva Anna Raquel Tavares, por todo o apoio e suporte moral e emocional ao longo dessa jornada de realização desse projeto, muito obrigado por nunca ter me deixado desistir e ter ficado ao meu lado todo esse tempo.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Automação.

USO DO MICROCONTROLADOR ESP8266 PARA AUTOMAÇÃO RESIDENCIAL

Ricardo Rodrigues Oliveira

Fevereiro/2017

Orientadores: Marcos Vicente de Brito Moreira
Públio Macedo Lima

Curso: Engenharia de Controle e Automação

Apresenta-se, neste trabalho como implementar um sistema de lâmpadas controladas através do celular, utilizando-se para isso o microcontrolador ESP8266. O objetivo é verificar a possibilidade de realização de projetos de automação residencial com o uso desse controlador para diversos fins. Para tal foi desenvolvido um circuito, um software implementado no ESP8266 e uma página web para interface celular-ESP8266.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

Ricardo Rodrigues Oliveira

February/2017

Advisors: Marcos Vicente de Brito Moreira
Públio Macedo Lima

Course: Automation and Control Engineering

In this work, we present how to implement a illumination system controled by a smartphone, with the use of the microcontroler ESP8266. The main objective is to check the possibility to develop home automation projects with this controler for several purpose. In order to do so, we developed a circuit, a software implemented in the ESP8266 microcontroler and a web page to make a interface between the smartphone and the ESP8266.

Sumário

Lista de Figuras	j
Lista de Tabelas	k
Lista de Abreviaturas	l
1 Introdução	1
1.1 Motivação	1
1.1.1 Automação residencial	1
1.2 Objetivo	2
2 Revisão Bibliográfica	3
2.1 ESP8266	3
2.2 ESP-12	4
2.2.1 NodeMCU	8
2.3 Arduino	8
2.4 Programação do ESP8266	11
2.4.1 IDE do Arduino	12
3 Método proposto para controle de lâmpadas	16
3.1 Circuito	17
3.2 Programação do ESP8266	21
3.2.1 Função setup()	21
3.2.2 Função loop()	22
3.3 Página Web	23
4 Resultados e Discussões	26
4.1 Dificuldades apresentadas	26
4.1.1 Erros mais comuns	26
4.2 Conclusão e Trabalhos futuros	27
Referências Bibliográficas	29

A Código - variáveis e bibliotecas	31
B Código - Função Setup()	32
C Código - Função loop()	34
D Código - Página Web	40

Lista de Figuras

2.1	Soquete para o ESP8266	5
2.2	Circuito	7
2.3	NodeMCU	8
2.4	Arduino Uno	9
2.5	Arduino Uno ESP8266 Wifi Shield	9
2.6	Comparação	10
2.7	Xbee	11
2.8	IDE do Arduino	12
2.9	IDE- preferências	13
2.10	Inserindo a URL	14
2.11	BoardManeger	14
2.12	Installando ESP8266	15
3.1	FDTI USB	17
3.2	Circuito 2	19
3.3	Conversor lógico	20
3.4	Circuito entre o ESP8266 e as lâmpadas	20
3.5	Foto do circuito	21
3.6	Pagina Web (Primeira versão)	25
3.7	Pagina Web (versão final)	25
4.1	espfail - falha ao carregar o programa	26
4.2	espfail - falha na conexão USB	27
4.3	espcomfail - watch dog rst	27

Lista de Tabelas

2.1	Tabela geral de especificações do ESP8266.	4
2.2	Tabela de pinos do ESP-12 tiradas do (livro)	6
2.3	Comparação entre o ESP8266 e o Arduino (Hardware)	10

Lista de Abreviaturas

AT	Como são conhecidos os comandos de Hayes, AT significa "atenção", p. 7
CSS	Cascading Style Sheets, linguagem de folha de estilo utilizada junto com outra linguagem de marcação, como o HTML, p. 22
GPIO	Entrada/Saída de uso geral, p. 5
HTML	Linguagem de marcação de hipertexto, linguagem utilizada na construção de páginas web, p. 22
IDE	Ambiente para desenvolvimento integrado, p. 7
IP	Protocolo de Internet, número dado à algum dispositivo conectado à rede, p. 17
MAC	Controle de Acesso de mídia, p. 8
PCB	Placa para circuito impresso, p. 2
Ping	Envio de uma mensagem para testar a conectividade entre equipamentos, p. 17
SSID	Identificador do conjunto de serviços, é o nome dado à rede WiFi, p. 17
URL	Localizador Uniforme de Recursos, p. 13
USB	Barramento Serial Universal, p. 2
VCC	Tensão de Alimentação, p. 6
firmware	instruções operacionais programadas em um hardware, p. 7
opensource	Código Aberto, p. 8

Capítulo 1

Introdução

1.1 Motivação

A presença de tecnologia na vida cotidiana tem se tornado cada vez mais forte e ligada à cultura da sociedade atual. Um dos conceitos por trás dessa mudança de realidade é o conceito de automação. A automação é uma realidade na sociedade desde o século XVIII, com a invenção da máquina a vapor, que trouxe esse fator ao setor industrial. Conforme o avanço da tecnologia, cada vez mais a automação está presente na vida cotidiana, não se restringindo somente a esse setor da economia. Com isso, diversos avanços já foram feitos para facilitar ou possibilitar novas formas de se realizar as atividades cotidianas e cada vez mais o homem se vê cercado de novas tecnologias que vão transformando a forma com que o mesmo interage com o mundo a sua volta, uma vez que a tecnologia possibilita o indivíduo a realizar coisas que antes eram impossíveis.

1.1.1 Automação residencial

Como dito em [1], automação vem do latim, significa mover-se por si, ou seja, automação é o ato de tornar alguma atividade ou sistema para que ele seja capaz de comandar seu próprio funcionamento, diminuindo a interferência humana ao mesmo. Já a automação residencial, também conhecido como domótica, é uma aplicação tecnológica que visa aplicar os conceitos da automação para a vida cotidiana, para simplifica-la e suprir algumas de suas necessidades, além de gerar mais conforto e segurança, para um ambiente doméstico. De uma forma geral a automação residencial tem como objetivo conseguir manipular o maior número possível de elementos e tarefas domésticas, fazendo com que as tarefas mais complexas sejam facilitadas e as que são simples, porém repetitivas gastem menos do tempo e da energia dos indivíduos.

1.2 Objetivo

O objetivo deste trabalho é desenvolver algumas tecnologias para a automação residencial, se baseando principalmente em diminuir os custos para o acesso dessa tecnologia a um público mais amplo. Sendo assim, esse trabalho irá focar principalmente no uso do microcontrolador ESP8266 para esse fim, devido ao seu baixo custo. Toda a comunicação de controle será feita via WiFi, que é o meio de comunicação do ESP8266. Toda a parte eletrônica foi projetada para poder ser colocada em uma Placa PCB, de forma a ficar pronta para instalação em seus devidos ambientes e que permita atualizações e a resolução de problemas de software via USB. Logo, neste trabalho foi desenvolvido uma tecnologia, um sistema para acionar até 5 lâmpadas via WiFi, com o ESP8266 acionando o relé que ligaria lâmpada respectiva à saída desejada, sendo esse relé conectado a um interruptor de forma a fazer uma ligação threeway, que permite que tanto o relé quanto o interruptor acionem a lâmpada. Todo o controle e acionamento dos dispositivos deve ser feito via celular, ou outros dispositivos com comunicação WiFi, que estejam conectados à rede local.

O restante deste trabalho está dividido da seguinte forma. No capítulo 2 se encontra a Revisão Bibliográfica, onde todo material pesquisado para esse trabalho é descrito e explicado, demonstrando assim o que já tem sido feito sobre o assunto e onde foram encontradas as informações relevantes para esse projeto.

No capítulo 3, é descrito todo o desenvolvimento do projeto, nesse capítulo se encontra a descrição de cada parte do circuito eletrônico utilizado bem como a programação feita, explicando seus códigos e as estratégias de programação utilizadas.

No capítulo 4 são apresentados os resultados obtidos, bem como os problemas mais recorrentes ao longo do projeto, além de discussões sobre projetos futuros.

Todos os códigos desenvolvidos nesse trabalho foram colocados em anexo para consulta e melhor entendimento.

Capítulo 2

Revisão Bibliográfica

2.1 ESP8266

O ESP8266 é um microcontrolador produzido pela empresa Espressif Systems. Esse microcontrolador possui um sistema de comunicação WiFi próprio, que é o seu grande diferencial, por esse motivo ele é largamente utilizado como módulo WiFi para outros microcontroladores, como o Arduíno, por exemplo, apesar de possuir um processador próprio, como visto nos trabalhos [2], [3] e [4] e de ser possível utilizar somente o ESP8266 para criar sistemas embarcados. Alguns trabalhos já foram feitos visando a utilização desse microcontrolador com diversas linguagens de programação e objetivos, como visto em [2], [3], [5] e [6]. Uma vantagem do ESP8266 é o seu baixo custo, geralmente na faixa de 20 a 50 reais.

O Esp8266 está no mercado desde 2014, o que faz com que ainda existam poucos artigos e trabalhos sobre ele. A principal publicação sobre o mesmo é o livro [2], que demonstra quais são as formas de programar o ESP8266 e quais são as suas utilidades, bem como a eletrônica necessária para realizar os projetos demonstrados.

Como dito em [2] e [4] existem diversos tipos de modelos do ESP8266, como o ESP-1, ESP-12, Esp Olimex por exemplo. Como o processador é o mesmo para todos os modelos do ESP8266, mudando apenas o número de pinos de entrada e saída (GPIO) disponíveis, memória disponível e o espaçamento entre os pinos, com isso, foi escolhido se utilizar o ESP-12, por ele ter um número razoável de entradas e saídas para esse projeto (seis) e pela facilidade de obtê-lo.

A Tabela 2.1, tirada do [2], mostra as especificações técnicas do ESP8266:

Tabela 2.1: Tabela geral de especificações do ESP8266.

Voltagem	3.3V
Consumo de Corrente	10 μ A
Memória Flash	16MB max (512k normal)
Processador	Tensilica L106 32 bit
Velocidade do processador	80-160MHz
RAM	32K + 80K
GPIOs	17(multiplexada com outras funções)
Analogico para digital	1 entrada com 1024 de resolução
Suporte 802.11	b/g/n/d/e/i/k/r
Maxima corrente de conexão TCP	5

Existem diversos outros trabalhos publicados na internet que se utilizam do ESP8266, como os supracitados [3] e [6], mas ele ainda carece de registros mais formais e mais abrangentes. Apesar da sua comunidade de usuários já ser considerável, seu uso no meio acadêmico ainda é pouco definido.

2.2 ESP-12

Esse é o módulo com mais entradas e saídas disponíveis (nove ao todo). Como mostrado em [2], com isso é possível testar diversos dispositivos utilizando apenas uma placa ESP8266. Para utilizá-lo em placas PCB normais ou em protoboards é necessário um soquete com o espaçamento usual de pinos, uma vez que a placa não possui o espaçamento padrão entre os pinos. A placa mais o soquete ficam juntos como mostrado na Figura 2.1:

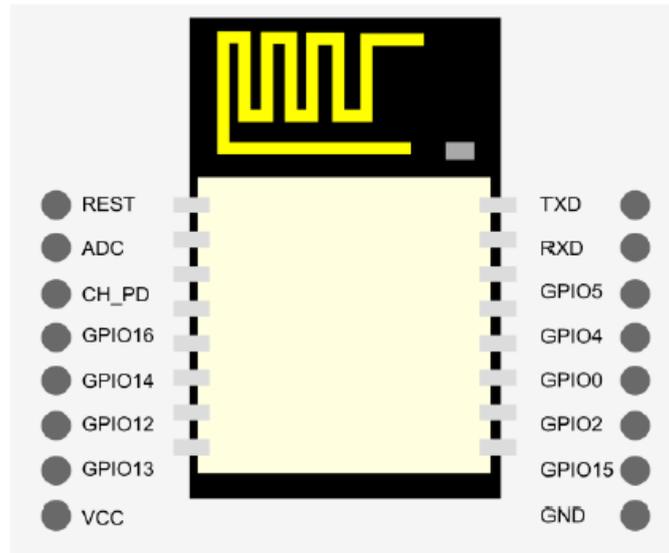


Figura 2.1: ESP8266 com soquete para uso em placas PCB

Na Figura 2.1 pode-se observar o posicionamento de cada pino do ESP8266. Na tabela 2.2 estão explicitados as especificações de cada pino assim como suas particularidades. É importante destacar que os pinos dados como GPIO são os passíveis de serem utilizados como entrada e saída do controlador.

Tabela 2.2: Tabela de pinos do ESP-12 tiradas do (livro)

Nome	descrição
VCC	3.3V
GPIO 13	Pode ser usado como SPI MOSI
GPIO 12	Pode ser usado como SPI MOSI
GPIO 14	Pode ser usado como SPI MOSI
GPIO 16	
CH_PD	Enable do ESP8266 Nível lógico 1 - Ativado Nível lógico 0 - Desativado
ADC	Entrada analógico para digital
Reset	1 - Normal 0 - Reset
TXD	Tramissão serial
RXD	Recepção serial
GPIO 4	GPIO normal
GPIO 5	GPIO normal
GPIO 0	Deve estar em nível lógico 1 para inicialização do programa (boot mode) e 0 para carregar o programa (flash mode)
GPIO 2	deve estar em nível lógico 1 para inicialização do programa (boot mode)
GPIO 15	Deve estar em nível lógico 0 para carregar o programa (flash mode) e para inicialização do programa (boot mode)
GND	Terra

Como apresentado na Tabela 2.2 alguns pinos são necessários para carregar o script. Foi-se sugerido que o circuito ficasse como mostrado na Figura 2.2, segundo [2].

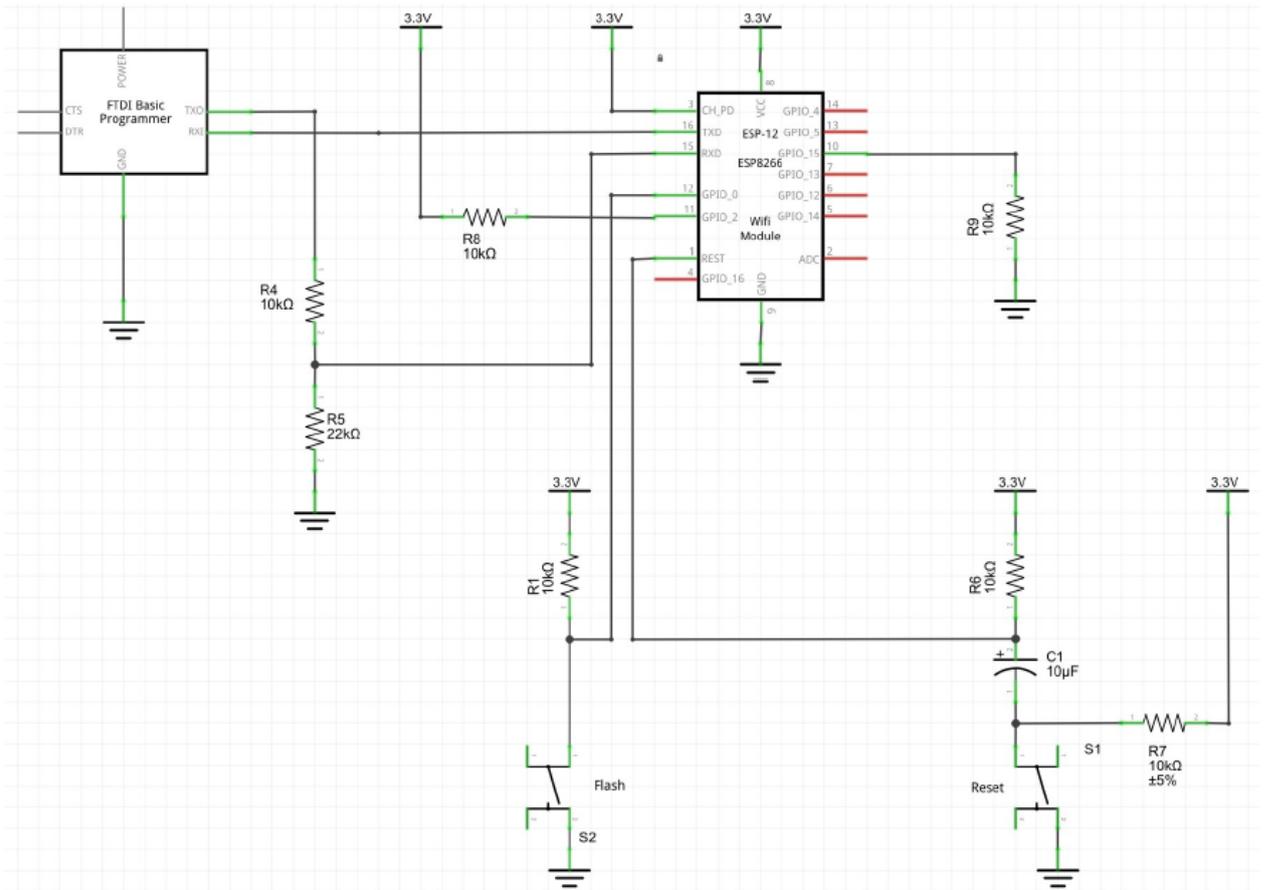


Figura 2.2: Circuito com o ESP8266

Como será visto mais adiante, neste trabalho o circuito utilizado foi um pouco diferente do apresentado na Figura 2.2, sem, porém, deixar de se basear no mesmo. Pode-se perceber que tanto em [4], quanto em [3] foram utilizados circuitos parecidos para atender os requisitos vistos na Tabela 2.2. Algumas adições foram feitas seguindo as propostas feitas em [7]. Como é especificado em [7] e como será explicado mais adiante é muito comum que o ESP8266 faça reset automaticamente, e isso pode ser causado por diversos motivos. Por isso fez-se a escolha de seguir algumas dicas dadas em [7], principalmente a adição de dois capacitores, um entre o ground e o VCC do circuito e outro junto aos pinos de alimentação do ESP8266.

Como dito em [8], apesar do padrão ser o ESP8266, modelo ESP-12 tenha um firmware AT é aconselhável que seja instalado um outro firmware, o NodeMCU, que permite que o ESP8266 seja programado em Lua, linguagem de script de multiparadigma, que foi projetada para expandir aplicações, e também faz com que possa ser utilizado a IDE do Arduino para programá-lo, como foi o escolhido nesse projeto e será abordado mais a frente.

2.2.1 NodeMCU

O NodeMCU é um firmware opensource baseado no próprio ESP8266, segundo o próprio site do mesmo [9]. Como dito ele permite que o controlador seja programado em Lua e através da IDE do Arduino.

Para instalarmos o firmware primeiro o chip deve estar montado junto ao circuito, em módulo flash, ou seja, com o GPIO0 ligado ao Ground, como indicado na Figura 2.2. É importante fazer o reset do ESP8266 antes, ligando o pino rst ou Ground e depois o desconectando, como também é mostrado na Figura 2.2.

Em seguida se inicializa o executável do NodeMCU.

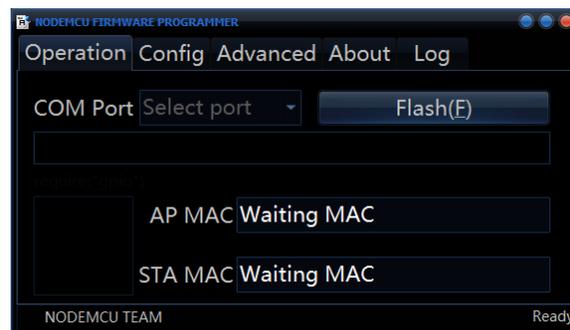


Figura 2.3: Interface para instalação do NodeMCU

Então, se deve selecionar a porta USB na qual o ESP8266 está conectado, na Figura 2.3 no campo "Select port". Ao selecionar o botão de Flash (ou pressionar a tecla F do teclado) o firmware iniciará a sua instalação. O número do AP MAC e STA MAC aparecerão em seus devidos locais, e a barra de status(no centro da interface) começará a ser preenchida conforme o avanço da instalação. Na aba log, é possível acompanhar o que está ocorrendo ao longo da instalação, e conferir eventuais problemas.

2.3 Arduino

O Arduino é uma plataforma eletrônica de código aberto, que possui seus próprios micro controladores e a sua própria IDE. O Arduino tem uma grande comunidade de usuários, diversos trabalhos e projetos que se utilizam dessa plataforma, pela sua facilidade de uso, eficiência e sua IDE gratuita, além do grande número de ferramentas já implementadas para o mesmo. Exemplos desses trabalhos são [10] e [11].

O Arduino é uma possível resolução para os problemas propostos por esse trabalho. Por isso ele será comparado com o ESP8266 para justificar a escolha pelo segundo. O Arduino Uno, apresentado na Figura 2.5 é a placa mais utilizada, por

não possuir um alto custo e ser bastante completa.



Figura 2.4: Arduino Uno

A primeira grande diferença entre o Arduino e o ESP8266 é a comunicação via Wifi. Nenhuma placa Arduino possui um protocolo de comunicação sem fio próprio e precisa de algum módulo instalado para isso. Caso o usuário deseje utilizar o Wifi para comunicação o próprio ESP8266 é muitas vezes utilizado como esse módulo de comunicação, mas o mais comum é utilizar o Wifi Shield, que é específico do Arduino.

Outra grande diferença que tende a favor da escolha pelo ESP8266 é o custo. O ESP8266 mais o soquete custa por volta de 50 reais, com a aquisição de uma porta para comunicação USB e a placa PCB para embarcar o circuito, o custo pode chegar a 70 Reais. Um Arduino Uno sem o módulo de comunicação Wifi custa por volta de 100 reais além de ser bem maior que o ESP8266, como demonstrado na Figura 2.5 retirada do [12].



Figura 2.5: Da esquerda para direita: Arduino Uno, ESP8266 e Wifi Shield

A Tabela 2.3, retirada de [2] faz também uma comparação entre os hardwares: Além disso a Figura 2.6, de [12], compara esses dados, e também o NodeMCU que é um tipo de ESP8266 mais robusto e completo.

Tabela 2.3: Comparação entre o ESP8266 e o Arduino (Hardware)

	ESP8266	Arduino
GPIOs	17(Menos geralmente estão expostas)	14 (20 incluindo analógicas)
Canais PWM	8	6
Velocidade do Clock	80 MHz	16 MHz
Processador	Tensilica	Atmel
SRAM	45KBytes	2KBytes
Flash	512Kb ou mais	32KB no chip
Tensão	3.3V	5V
Corrente Maxima Por I/O	12 mA	40 mA
UART	1 1/2	1
Networking	Incluso	Separado
Documentação	Pobre	Excelente
Maturidade	Recente	Maduro

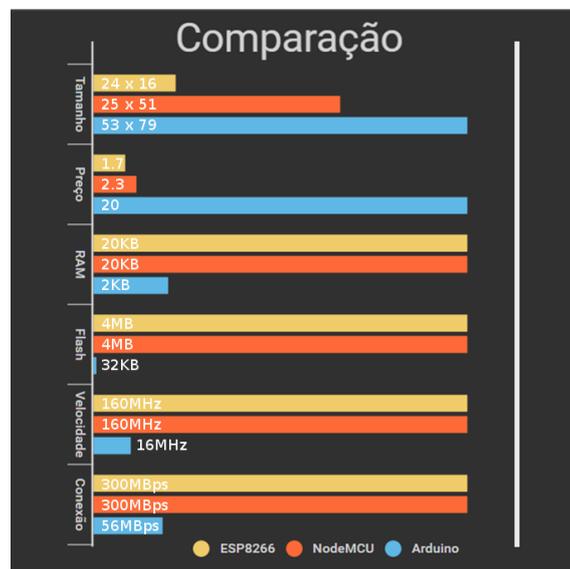


Figura 2.6: Gráfico de comparação

Outra solução para realizar este projeto seria utilizar um Arduino com um Shield Ethernet, que permite que o mesmo seja conectado à internet por um cabo ethernet, para controlar as lâmpadas utilizando uma comunicação via rádio frequência a partir do Xbee, mostrado na Figura 2.7.



Figura 2.7: Comunicador Xbee

O Xbee é um módulo de comunicação via rádio frequência, que possui diversos modelos. Ele é bastante utilizado por ser muito robusto, bem documentado e confiável.

Essa solução traz diversas limitações, a primeira é também o custo, já que só uma antena Xbee custa aproximadamente 200 reais, caso seja uma antena slave, que é bem menos robusta, uma antena master, que seria conectada junto ao Arduino e comandaria todas as outras antenas custa em torno de 400 reais. Esse custo tornaria colocar um módulo em cada conjunto de interruptores muito caro, e mesmo se fosse colocado o comando junto aos disjuntores da residência, utilizando assim somente duas antenas Xbee, o preço ainda sim seria muito elevado.

Porém como também é destacado tanto em [12] quanto em [2] o ESP8266 possui algumas desvantagens. Como se pode observar todas as saídas assim como sua alimentação é de 3,3 V, o que gera a necessidade de um conversor de nível lógico para a maioria dos sistemas que o utilizam, uma vez que essa tensão não é comum. Além disso, pelo ESP8266 ainda ser muito recente a sua documentação é bastante escassa, o que gera algumas dificuldades em compreender e solucionar problemas com o Hardware. Apesar da sua crescente comunidade de usuários, ainda faltam mais livros e tutoriais formais sobre o mesmo, o que dificulta a sua utilização, sendo necessário muitas vezes se utilizar da tentativa e erro para descobrir as soluções de problemas que possam vir a se apresentar. Outro ponto negativo do ESP8266 é a existência de somente uma entrada analógica em cada chip, que só suporta no máximo 1 V. Isso pode dificultar a utilização de sensores simples e mais uma vez se fazer necessário o uso de alguns conversores de voltagem em projetos mais complexos.

2.4 Programação do ESP8266

O ESP8266 permite que se rode programas carregados em sua própria memória. Com isso o ESP8266 pode funcionar de forma embarcada, sem precisar estar ligado

a outro dispositivo fisicamente, podendo fazer a comunicação somente via WiFi ou via sensores e saídas pelos pinos. Para carregar um programa ao ESP8266 ele deve ser compilado para linguagem de máquina, por isso é necessário um compilador.

Em [2], o autor demonstra diversas formas de se criar um programa para o ESP8266. A programação é feita geralmente na linguagem C. Para facilitar o processo de programar e compilar o programa, o programador pode lançar mão de uma IDE. Existem diversas IDE's disponíveis para o ESP8366, e muitos trabalhos, assim como [3] e [4], se utilizam da IDE do Arduino, por ela ser gratuita, possuir uma vasta documentação, e de uso simples.

2.4.1 IDE do Arduino

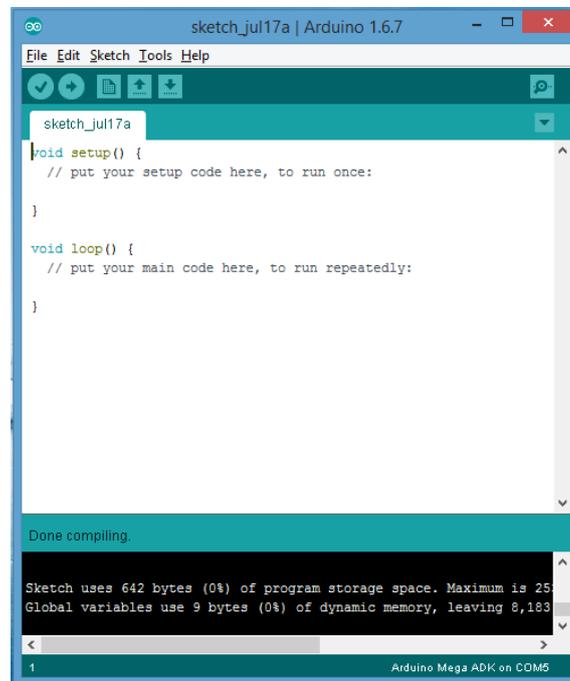


Figura 2.8: IDE do Arduino

Os programas na IDE do Arduino são denominados “sketchs” e se utilizam da linguagem C. A IDE já possui algumas bibliotecas e funções próprias, mas também permite que o usuário adicione outras.

Os programas nesse ambiente têm duas funções obrigatórias, a função setup e a função loop. A função setup é chamada toda vez que o dispositivo é ligado, sendo a primeira função a ser chamada. Nela se realiza todas as chamadas de inicialização do programa. No caso deste trabalho, é durante a função setup que o ESP8266 tenta se conectar na rede WiFi local.

Já a função loop, que é chamada necessariamente após a função setup, funciona como um laço, sempre ao seu final, a função é chamada novamente, até que um

comando seja dado para a função parar ou o dispositivo seja desligado. Nessa função são realizadas todas as chamadas que são pertinentes ao funcionamento normal da placa, o que o programa deve realizar ao longo do seu funcionamento. Neste trabalho, na função loop são tratadas as chamadas dos clientes que se conectarem à placa.

Por ser uma plataforma de código aberto já foi implementado a possibilidade de se utilizar o ESP8266 como uma placa selecionável na IDE ao download como demonstrado em [4]. Seguindo o passo a passo do mesmo, para instalar a placa foi feito o seguinte:

1. Abre-se a janela de preferências, como mostrado na Figura .2.9

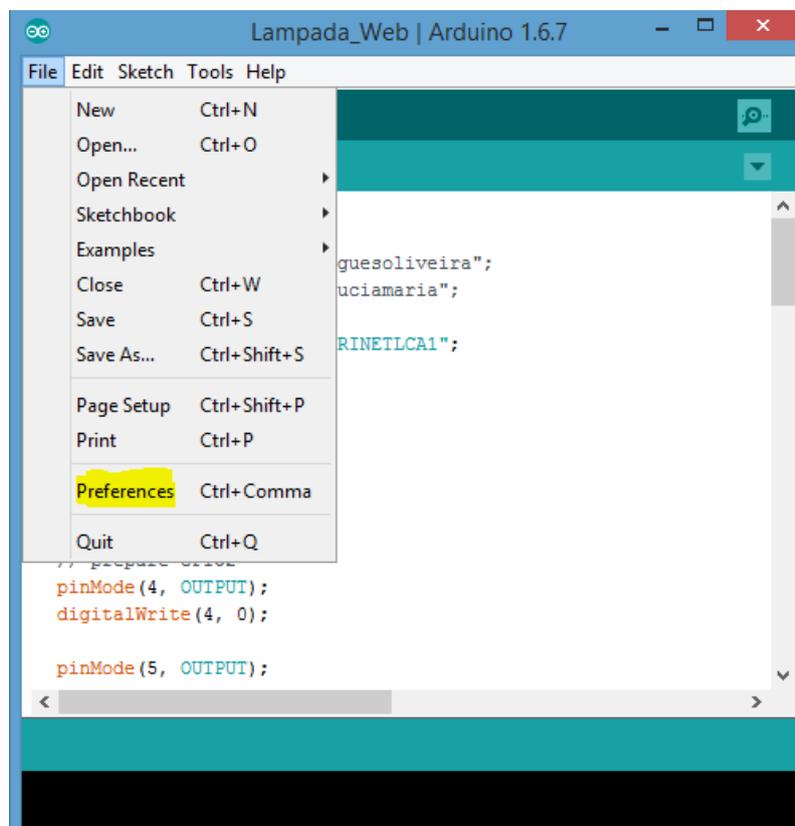


Figura 2.9: Janela de preferências na IDE do Arduino

2. Em seguida, se insere a seguinte URL no campo de gerenciamento de placas adicionais: http://arduino.esp8266.com/package_esp8266com_index.json, como é mostrado na Figura 2.10.

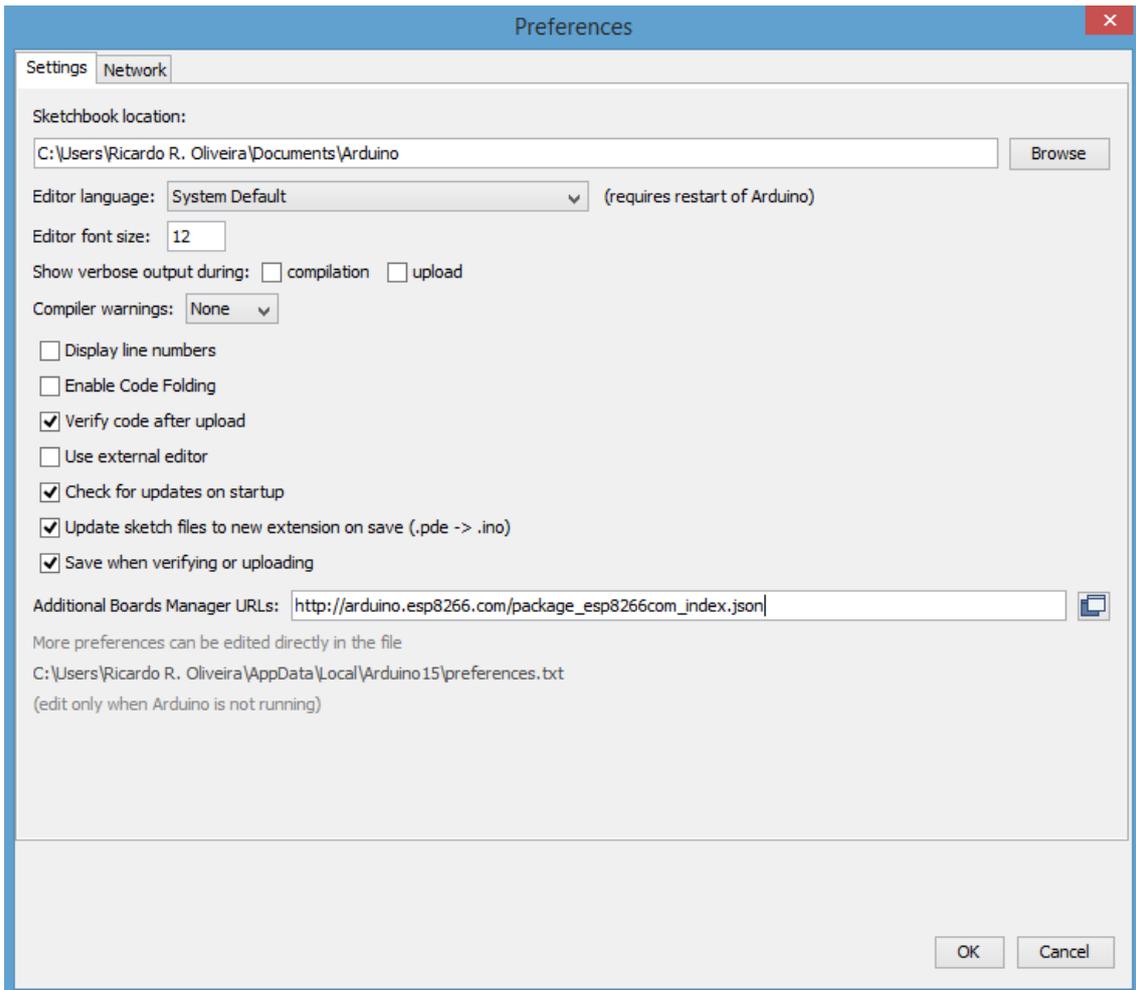


Figura 2.10: Inserindo a URL

3. Após isso, em ferramentas, placas, se seleciona gerenciador de placas, como mostrado na Figura 2.11.

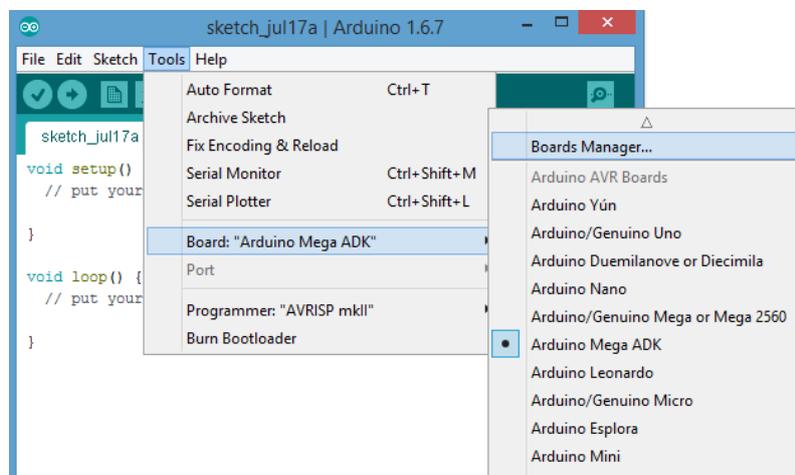


Figura 2.11: Gerenciador de Placas.

4. E instala-se a placa ESP8266, selecionando Gerenciador de placas o item do ESP8266, como na Figura 2.12.

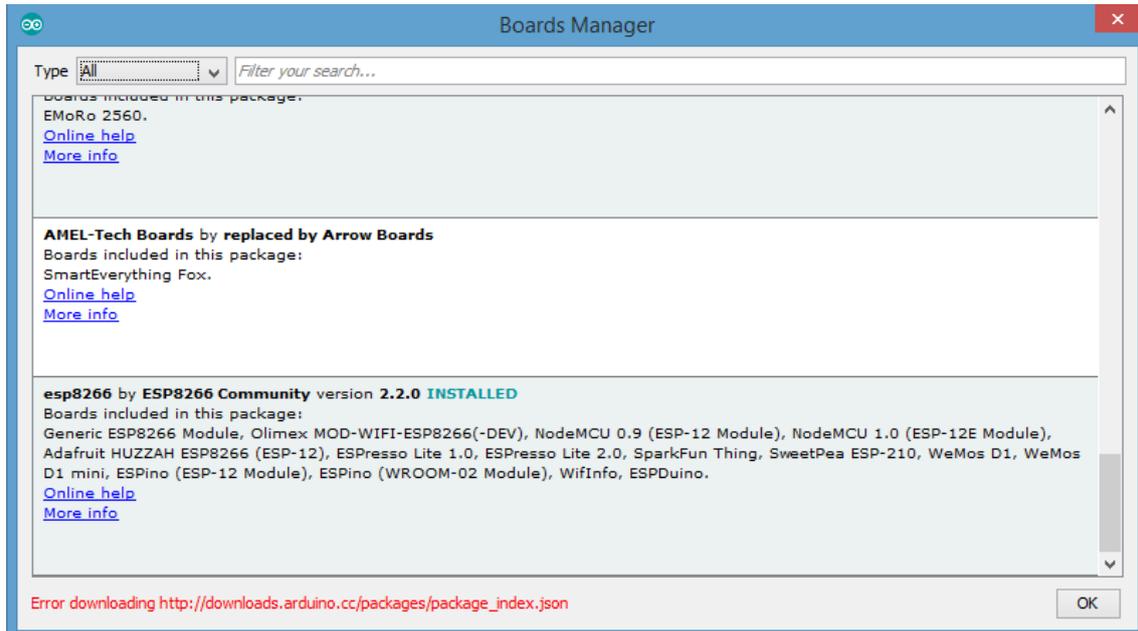


Figura 2.12: Instalando o ESP8266.

É importante destacar que a última versão disponível até o momento do projeto da IDE (1.6.7) foi a utilizada no mesmo.

Neste trabalho serão utilizadas duas bibliotecas específicas para o ESP8266, ESP8266WiFi.h e ESP8266Ping.h

Capítulo 3

Método proposto para controle de lâmpadas

A proposta deste projeto é criar circuitos para controlar elementos de uma residência através de um celular. Logo, pode-se dividir o projeto nas seguintes etapas:

1. Montagem do circuito
2. Programação do ESP8266
3. Comunicação com o telefone celular

O circuito foi montado em uma Protoboard, porém sendo desenvolvido com a intenção de uma fácil montagem em uma placa PCB.

Como o ESP8266 não possui uma entrada USB, como visto na Figura 2.1, a sugestão é que se utilize uma entrada USB FDTI para fazer a comunicação serial com o computador. Para este projeto foi escolhido uma entrada que tivesse tanto os sinais e a alimentação de 5V, que é a tensão normal de entradas USB, quanto 3.3V que é a tensão de alimentação e lógica do ESP8266. Caso seja utilizada uma tensão superior, o mesmo pode ter seus circuitos queimados. Assim, foi escolhido o Mini 5 V / 3.3 V Ft232rl Ftdi Usb, com a entrada Mini USB, mostrado na Figura 3.1. Nele é possível selecionar qual a tensão de saída e entrada dos pinos, evitando danos ao ESP8266.

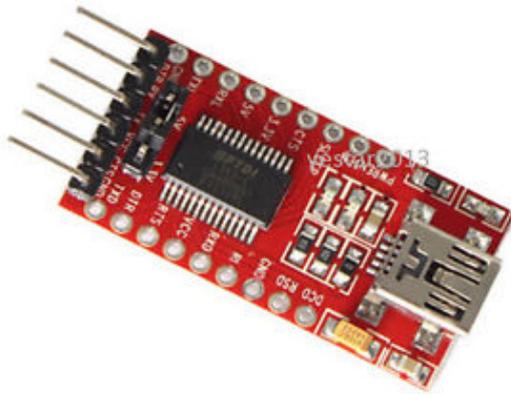


Figura 3.1: FDTI USB

Durante a etapa de testes e programação a porta USB FDTI foi utilizada também como fonte de 3.3V, o que gerou alguns problemas de estabilidade na alimentação do ESP8266. Assim, se passou a utilizar uma fonte regulável externa. Para embarcar o circuito foi utilizada ainda uma outra fonte, de 5 V, para alimentar os relés, como será explicado mais a frente.

A programação do ESP8266 foi feita na IDE do Arduíno usando a linguagem C. Foi utilizada a biblioteca ESP8266WiFi.h, que permite programar os comandos de rede WiFi com o ESP8266, como se conectar na rede WiFi, usando o SSID, que é o nome da rede WiFi a qual se deseja conectar o ESP8266, e a senha da mesma, criar um servidor, enviar e receber pedidos de clientes conectados no mesmo. Também foi utilizada a biblioteca ESP8266Ping.h, que permite enviar uma mensagem do tipo ping para um endereço web externo ou interno à rede a qual se está conectado.

Para facilitar a programação, escolheu-se alocar um servidor local no ESP8266 e criar uma página web no mesmo para que o usuário possa executar os comandos. A facilidade que essa escolha trás é que além de ser mais simples fazer uma página web do que um aplicativo web, a comunicação com o ESP8266 também é facilitada, uma vez que a página e os comandos advindos das entradas do usuário já estariam no próprio ESP8266, bastando que o usuário acesse o IP do mesmo e exiba a página.

3.1 Circuito

Além da entrada USB, para esse circuito foi utilizada uma um módulo de 8 relés srd-05vdc-sl-c, com apenas 6 em uso, e switches para servir de interruptores. Cada

relé foi ligado a uma lâmpada através de um interruptor threeway, para que tanto o interruptor quanto o relé controlado pelo ESP8266 possam ter controle sobre a lâmpada. Para isso, o relé utilizado tem dois contatos, normal para abrir (NA) e normal para fechar(NF), sem esses dois contatos não é possível fazer o threeway com o interruptor.

Como já foi dito, o circuito para carregar o programa no ESP8266 é um pouco diferente do circuito que inicializa e roda o programa.

O primeiro passo para montar o circuito para carregar o programa no ESP8266 é conectá-lo ao USB FDTI. É de suma importância que o USB FDTI esteja utilizando a voltagem de 3.3V. Caso não seja possível, é necessário se utilizar um conversor de nível lógico, pois uma voltagem superior a essa irá danificar o ESP8266. A ligação entre os dois dispositivos deve ser feita da seguinte maneira: O pino RDX (entrada de leitura) do ESP8266 deve ser ligada no Pino TDX (saida de transmissão) do USB FDTI, da mesma forma, o TDX do ESP8266 deve ser ligado no RDX do FDTI.

Em seguida, é necessário conectar os seguintes pinos do ESP8266: CH_PD, RST, GPIO0, GPIO15, da forma mostrada na Tabela 2.2. Para ter um sistema mais estável alguns componentes eletrônicos foram adicionados, eles são mais importantes para o funcionamento do sistema quando estiver rodando o programa, mas já podem ser adicionados antes de carregar o programa. Esses elementos são os capacitores e resistores já demonstrados na Figura 2.2 e mais dois capacitores. Como mostrado na sessão 2.2, o primeiro foi um capacitor de 470 uF entre o VCC e o ground do circuito e o segundo um de 0,1 uF entre o VCC e o ground do ESp8266, ambos com o objetivo de dar mais estabilidade ao circuito. Com a adição desses elementos o circuito ficará como mostrado na Figura 3.2:

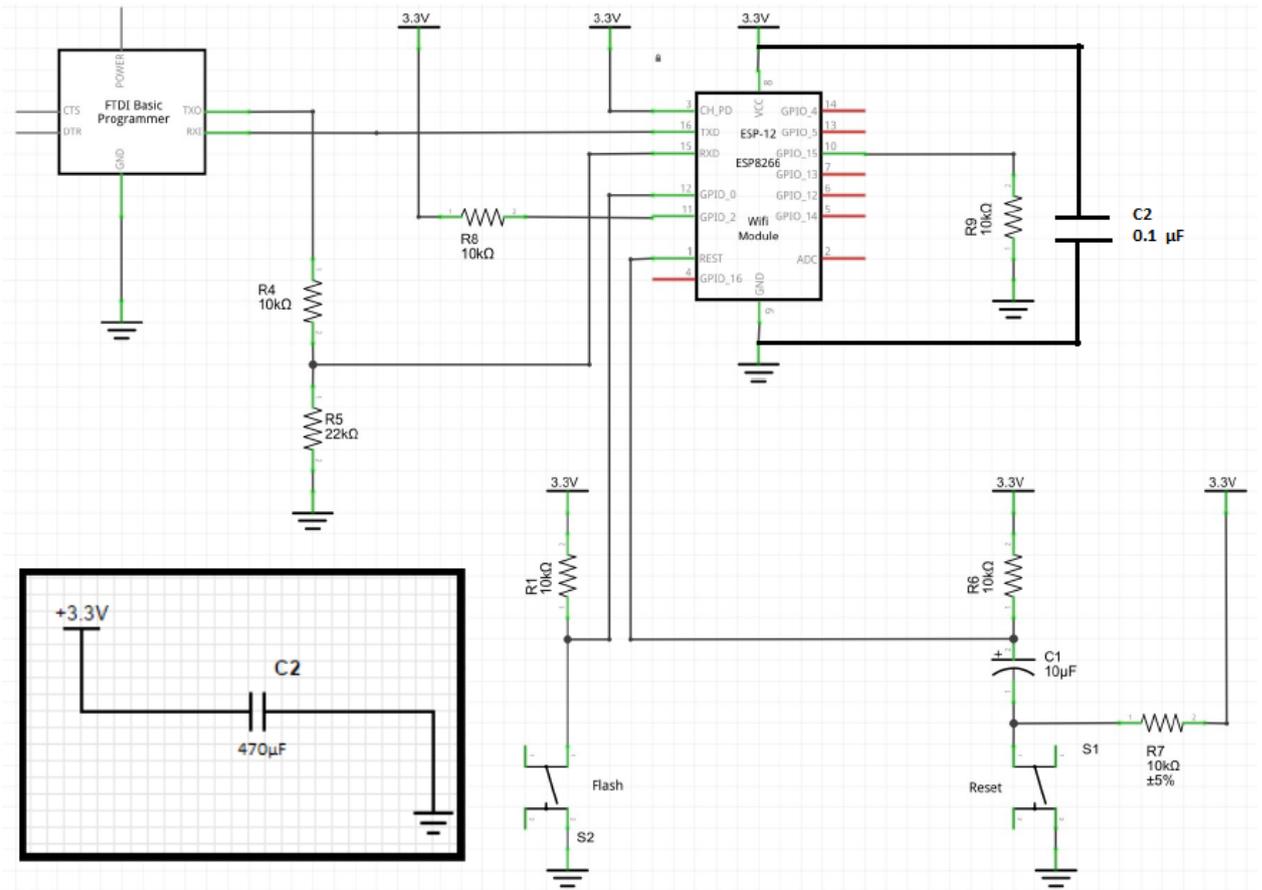


Figura 3.2: Circuito final para o acionamento do ESP8266

O pino de Reset sempre antes de carregar o programa deve ser conectado ao Ground e depois ao VCC novamente, para que o ESP8266 seja reiniciado e assim, seja possível carregar um novo programa, como é demonstrado na Figura 3.2.

Para carregar o programa é preciso colocar o ESP8266 em boot mode, para isso o GPIO 0 deve passar do nível lógico 0 para o nível 1 e o GPIO 2 deve ser colocado em nível lógico 1.

É importante reforçar a necessidade de se utilizar uma fonte independente para alimentar o ESP8266. É desaconselhado o uso do módulo USB como fonte, pois sua instabilidade faz com que muitas vezes o processador não possua energia suficiente para ligar, ou se manter funcionando, causando por várias vezes um reset no ESP8266 ou simplesmente o usuário não irá conseguir se conectar com ao ESP8266.

Para acionar as lâmpadas através do comando de controle dado pelo ESP8266 é necessário um conjunto de relés, um para cada lâmpada, e também um conversor de nível lógico. Isso se deve ao fato de que o relé utilizado nesse projeto só é acionado a partir de 5V.

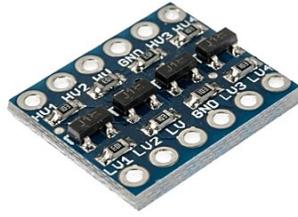


Figura 3.3: Conversor de nível lógico bidirecional

Para que o conversor, mostrado na Figura 3.3, funcione é necessário ligar ambas as tensões nele, tanto uma fonte de 3.3V (Pino LV, já que é a menor tensão) quanto uma de 5V (Pino HV, que é a tensão mais alta). Em seguida, deve se conectar o sinal que se deseja converter a um dos pinos, por exemplo L1, e no seu correspondente, nesse exemplo, H1 o sinal estará convertido para 5V, caso se deseje passar de 5V para 3.3V, é só fazer o processo inverso. Como Cada um dos conversores de nível lógico utilizados podem converter até 4 sinais. Então caso se deseje utiliza todas as saídas de controle do ESP8266 deveremos usar 2 conversores de nível lógico como esses.

Após passar pelo conversor de nível lógico, o sinal de controle vai para o relé. Para o acionamento de uma lâmpada, a ligação feita com o ESP8266, relé e lampada é mostrada na Figura 3.4.

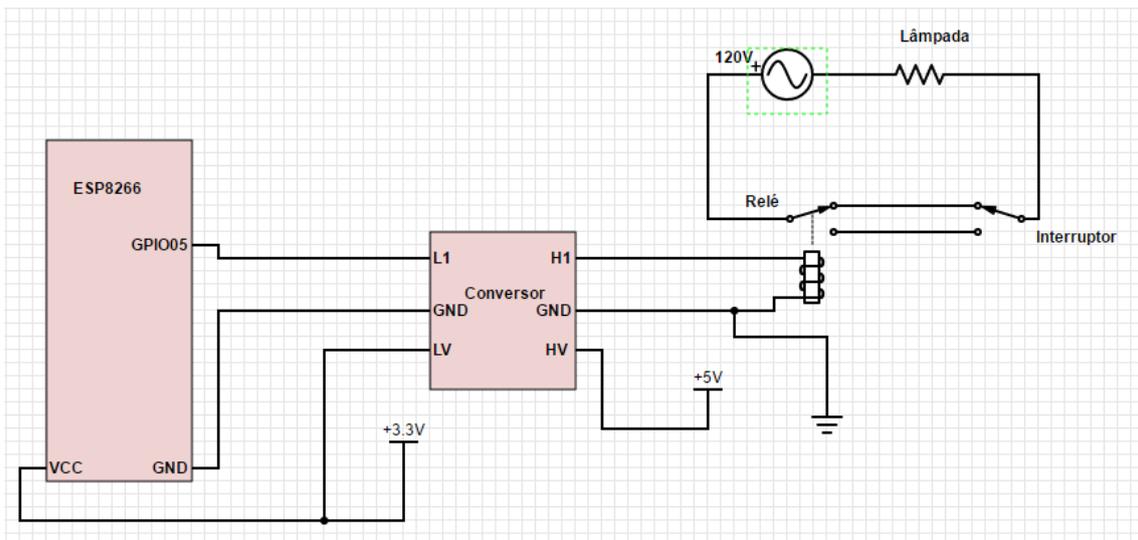


Figura 3.4: Circuito entre cada saída do ESP8266 e sua respectiva lâmpada

O circuito implementado na protoboard ficou como mostrado na Figura 3.5.

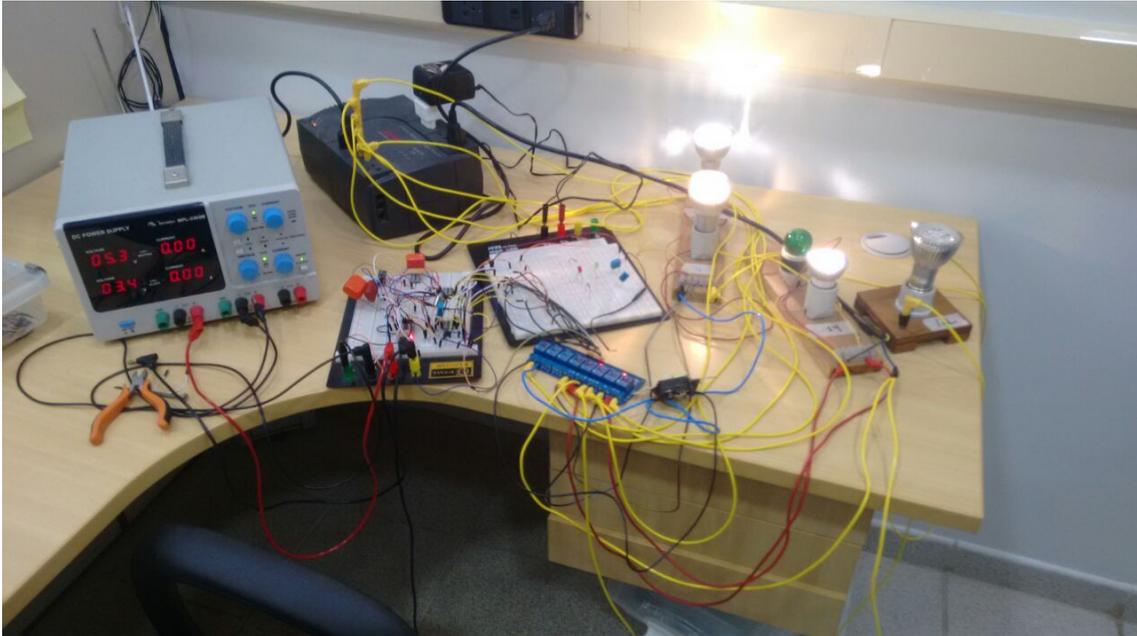


Figura 3.5: Foto do circuito completo montado na protoboard

3.2 Programação do ESP8266

O primeiro passo para a programação de todos os controladores é conectar o ESP8266 à rede local. Para isso é necessário criar uma variável tanto com o SSID quanto a senha da rede e em seguida, criar um servidor no próprio ESP8266. Para isso, é necessário criar uma variável do tipo `WiFiServer`, indicando também qual a porta do roteador será utilizada pelo ESP8266.

Toda vez que o ESP8266 se conecta com a internet o roteador o aloca a um IP local que esteja disponível. Para acessar o mesmo é necessário acessar esse endereço IP local. É possível fixar o IP de acesso diretamente pelo roteador utilizando o MAC do ESP8266, o que foi feito para esse projeto. Em projetos maiores, com vários dispositivos sendo utilizados, isso é de suma importância para poder se diferenciar cada um dos dispositivos.

Todas essas variáveis são criadas antes da declaração da função obrigatória `setup` no início do script, como demonstrado no Apêndice A.

3.2.1 Função `setup()`

Na função `Setup` serão chamados todos os comandos de inicialização do programa. Com isso, esse programa irá:

1. Iniciar a comunicação serial com o PC (fase de programação e diagnósticos).

2. Declarar os pinos utilizados como saídas.
3. Conectar o ESP8226 ao WiFi.
4. Iniciar o servidor.

Primeiramente deve se indicar a velocidade da comunicação serial (baud rate). Isso é determinado pelo comando `Serial.begin()`. O baud rate deve ser colocado em 115200 bits/s no caso do ESP8266 ESP-12, outros modelos do ESP8266 podem possuir um baud rate diferente. Isso permitirá uma comunicação com o computador melhor, sem problemas na hora de verificar o monitor serial.

É válido ressaltar que o número de pinos designados como saídas depende do número de lâmpadas que desejam ser acionadas. Para este projeto foi escolhido utilizar as seis saídas livres do ESP8266 para testar sua capacidade de controlar múltiplas lâmpadas.

O código da função `setup` está descrito no Apêndice B.

3.2.2 Função `loop()`

Na função `loop`, como pode ser visto no Apêndice C são chamados todos os comandos que são pertinentes ao funcionamento normal do controlador, ou seja, aqueles que devem ser chamados indefinidamente enquanto o sistema estiver funcionando.

Nessa função primeiramente é importante verificar se existe algum dispositivo (chamado de cliente) conectado ao ESP8266. Para isso foi criada uma variável `client` que vai verificar a existência de clientes e receber todos os comandos pertinentes a esse cliente. Cabe à função `loop` tratar os pedidos feitos pelo cliente, assim como apresentar para ele todas as informações necessárias.

O sistema apresentou um problema no servidor durante o desenvolvimento do projeto, que fazia com que o servidor ficasse fora do ar sem motivo aparente. Por isso foi adicionado um procedimento ao início da função `loop` que a cada 35 segundos ele envia um ping para o roteador, para verificar se o servidor ainda está no ar. O ping não é enviado em todas as vezes que se roda a função `loop` devido ao fato desse processo demorar de 5 a 7 segundos para ser executado, o que deixa o programa muito lento em cada comando.

A página web a ser exibida para o cliente conectado ao ESP8266 é gerada dentro da função `loop`. Logo, seus textos e botões foram declarados em HTML5 e CSS3 para uma variável do tipo `string`, que exibirá para o cliente com o comando `client.print(variável)` e `client.flush()`.

Um outro problema foi detectado com a conexão ao servidor. O ESP8266 apresentou dificuldades ao ter pelo menos 3 clientes conectados a ele ao mesmo tempo. Quando o número ultrapassava esse limite o sistema caía por alguns minutos e muitas

vezes não conseguia retornar sem que fosse reiniciado manualmente. Sendo assim, a solução encontrada foi retirar o usuário do servidor, pelo comando `client.stop()`, após a página ser carregada para ele, isso faz com que o cliente só ocupe parte da banda oferecida por um curto espaço de tempo, deixando assim o servidor livre quase 100% do tempo.

Apesar desse procedimento algumas vezes quando o cliente se conectava ao servidor mas o seu comando não era recebido pelo mesmo, com isso o cliente não era expulso do servidor, que estava esperando a chegada do comando, fazendo assim com que o servidor ficasse congestionado tentando se comunicar com esse servidor. Por isso foi adicionado um procedimento de "time out" que após 5 segundos o cliente é retirado do servidor, liberando assim a página para que novos clientes e comandos sejam enviados.

3.3 Página Web

A página web é onde todos os comandos para o ESP8266 são dados. Essa página é carregada aparecendo as opções de lâmpadas a serem acionadas por aquele dispositivo. Ao selecionar uma das lâmpadas o seu estado é trocado, ou seja, se estiver em nível 1 vai para nível 0 e vice e versa. Para isso, a página manda um comando GET para o servidor (ESP8266) e recarrega a mesma página. Através do comando GET enviado é possível determinar qual porta o usuário quer que seja acionada.

A página foi feita em HTML5 com o uso de CSS3. Ela foi adicionada diretamente ao corpo do programa do ESP866 como uma variável de texto chamada `buf`. Todo o corpo da página foi escrito nela, tanto a parte em HTML5 quanto os trechos em CSS3. As partes programadas em CSS3 devem ser indicadas pelo comando `<style>`. O código da página web esta apresentado separadamente no Anexo D.

Na página foi criado um botão para cada lâmpada. Para cria-lo foi criada uma classe em CSS3, chamada `bnt`, ainda no trecho denominado de `<head>` do código. Ao criar essa classe são declaradas todas as propriedades do botão, seu formato, cor de fundo, fonte de escrita, como será o `hover`, que é a animação quando se passa o cursor na área designada pelo botão no computador, e no celular é a animação ao selecionar o botão, entre outras. Na classe o `position` foi deixado como `static` para o botão ser posicionado de acordo com a chamada do mesmo no corpo do programa, chamado de `<body>`. Para criar a classe do botão foi utilizado o [13]. Utilizando o mesmo é possível criar o botão da forma desejada, e a própria plataforma prepara o código do mesmo.

Para posicionar os botões na página de forma organizada foram criados três parágrafos, iniciados pelo comando `<p>`, cada parágrafo corresponde a uma linha de botões, e cada parágrafo possui dois botões, como visto nas Figuras 3.6 e 3.7.

Para adicionar esse botões à página é necessário criar uma aplicação para cada um deles, indicada pelo `<a>`. Dentro de cada aplicação a classe `bnt` é chamada. É preciso passar para essa aplicação dois atributos, o primeiro é o `href`, que indica qual comando será dado com a seleção deste botão, neste caso, o `href` vai ser o comando `GET` para acionamento da lâmpada. O segundo atributo é o título do botão (`title`), que será o nome do elemento, esse nome não aparece para o cliente. É possível também escolher o que se deseja que esteja escrito em cada botão, escrevendo o texto fora da chamada da aplicação mas antes do seu encerramento (``).

Ao clicar em cada botão o usuário está energizando ou desenergizando um relé, que liga ou desliga a sua respectiva lâmpada. Para esse acionamento o click no botão envia um comando `GET` ao ESP8266, indicando qual é o pino que deve ser acionado. Após esse envio a página é recarregada no dispositivo e como já foi dito, o cliente é retirado do servidor, sem porém, deixar de ter a página aberta para o usuário.

O CSS3 também foi utilizado para a criação do plano de fundo da página. Tudo foi feito na classe `body`, que comanda o corpo da página, ainda no `<head>` da página. Nesse trecho, foi escolhida a cor predominante do fundo, as imagens que aparecem e suas posições e as posições que elas se repetem. Esse fundo foi tirado do [14], uma galeria de planos de fundo para páginas web, com os códigos das mesmas.

Foi adicionada a seguinte linha ao código da página: `<meta name= "viewport"content= "width=device-width, initial-scale =1»`, para que ela se adaptasse ao tamanho da tela do dispositivo cliente, com isso, ao acessar o ESP8266 pelo celular a página ficará no tamanho ideal para o dispositivo.

Por fim, o logo do LCA foi adicionado na parte de inferior da página. O link da imagem ficou hospedado em uma página da internet, para adiciona-la, basta chamar o comando ``, passando para esse comando o atributo `scr`, que é o endereço da imagem na internet e `style`, que vai indicar quais as dimensões da imagem.

Para todos os elementos ficarem centralizados na página, foi utilizado o comando `<center>`.

A diferença entre a página com e sem uso de CSS3 pode ser percebida comparando a página da primeira versão e da versão final, nas Figuras 3.6 e 3.7, respectivamente.

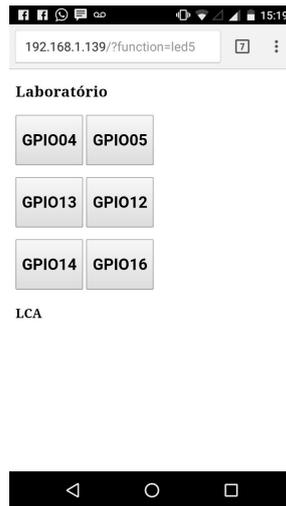


Figura 3.6: Página no celular (primeira versão)



Figura 3.7: Página no celular versão final

Essa opção de retirar o cliente do servidor só é viável pelo fato do sistema não possuir sensores, com isso o cliente não precisa estar conectado ao ESP8266, pois ele não envia nenhuma informação do sistema para o cliente.

Capítulo 4

Resultados e Discussões

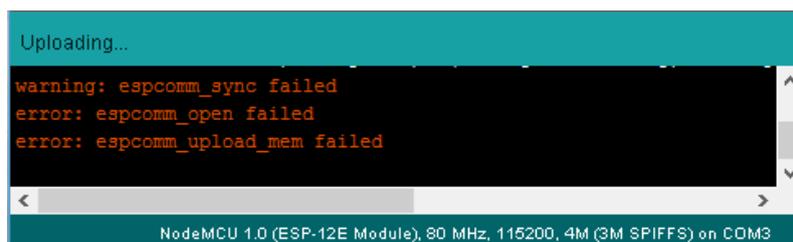
4.1 Dificuldades apresentadas

A falta de documentação acerca do ESP8266 dificulta bastante um primeiro projeto com ele, pois é muito difícil encontrar informações sobre ele no mesmo local, e informações de forma formal. Porém, ele é um dispositivo que merece uma exploração e a realização de projetos diferentes.

Ao longo deste projeto a maior dificuldade se deu em tentar estabilizar o ESP8266 que muitas vezes reiniciava sem motivo aparente como foi descrito anteriormente.

4.1.1 Erros mais comuns

Falha na comunicação

A screenshot of a terminal window with a teal header and footer. The header says "Uploading..". The main area is black with orange text showing three error messages: "warning: espcomm_sync failed", "error: espcomm_open failed", and "error: espcomm_upload_mem failed". The footer shows "NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM3".

```
Uploading..  
warning: espcomm_sync failed  
error: espcomm_open failed  
error: espcomm_upload_mem failed  
NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM3
```

Figura 4.1: Falha ao carregar o programa

Muito comum ao tentar subir o programa para o ESP8266. Geralmente ocorre quando alguma das ligações está feita errada, ou quando o ESP não foi reiniciado de forma adequada antes da instalação do programa, resultando na mensagem mostrada na Figura 4.1. Pode também ocorrer quando o cabo USB não está conectado devidamente, aparecendo a mensagem mostrada na Figura 4.2.

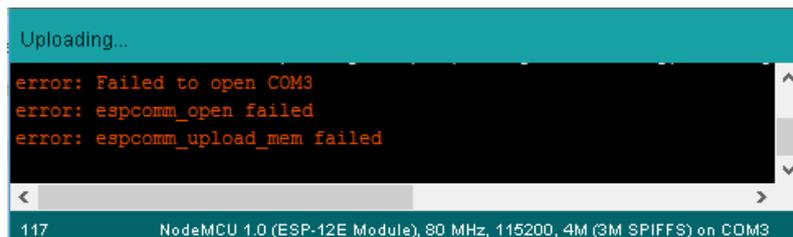


Figura 4.2: Falha devido a conexão USB

Watch dog reset

Como já foi dito anteriormente esse é um problema recorrente no ESP8266 e foram usadas diversas técnicas nesse projeto para contorná-lo. O watch dog reset ocorre por diversos motivos, sendo os mais comuns a carga insuficiente no pino de reset e também quando o sistema fica muito tempo inativo.

```
ets Jan  8 2013,rst cause:2, boot mode:(1,7)

ets Jan  8 2013,rst cause:4, boot mode:(1,7)

wdt reset
```

Figura 4.3: Mensagem enviada à comunicação serial quando um Watch Dog Reset ocorre. Tirada do Livro [2]

4.2 Conclusão e Trabalhos futuros

Os resultados finais obtidos foram bastante satisfatórios com a proposta inicial do projeto. Foi testado tanto o controle de poucas lâmpadas, quanto o de seis ao mesmo tempo, com diversos clientes comandando-as, e foi também feito um teste com uma lâmpada conectada a 10 metros de distancia do seu respectivo relé. Em todos os testes finais o sistema respondeu de forma satisfatória. O único problema apresentado foi que ao receber diversos comandos simultâneos, o sistema demorou cerca de 5 segundos responder a esses comandos.

O sistema se recuperava sozinho de uma queda de energia, simulada com o desligamento da fonte, se conectando novamente na internet sozinho quando religado.

Quando o usuário seleciona um dos botões da página, a lâmpada correspondente é acionada. O tempo de resposta costuma a ser aproximadamente 1 segundo mas pode demorar até 5 segundos, que geralmente ocorre quando o comando é dado durante o envio do ping.

Apesar das dificuldades apresentadas pela pouca maturidade do ESP8266 ele vale o investimento no campo da automação residencial, principalmente com evoluções

do mesmo já surgindo, como é o caso do ESP8266 ESP-32. Ainda podem ser feitos diversos projetos acadêmicos para explorar na totalidade as possibilidades com o ESP8266 e para facilitar a documentação do mesmo

Futuros projetos podem abordar outras tecnologias com o ESP8266, como luzes dimerizáveis, controles por infra vermelho, controles analógicos de motores, para persianas por exemplo, e principalmente a parte de segurança. Seria interessante realizar um projeto de um sistema maior, com mais elementos diferentes integrados a um aplicativo de celular que pudesse ser disponibilizado para mais de um celular com autenticação de usuários por senhas. Para projetos futuros também vale se fazer uso do ESP8266 ESP32, um novo modelo do ESP que tem diversas entradas analógicas e também possui saídas analógicas e um comunicador Bluetooth, além de melhor RAM, pinagem compatível com placas PCB, entrada USB própria entre outros. Esse modelo é mais caro e está há aproximadamente um ano no mercado.

Apesar das dificuldades apresentadas pela pouca maturidade do ESP8266 ele vale o investimento no campo da automação residencial, principalmente com evoluções do mesmo já surgindo, como é o caso do ESP8266 ESP-32. Ainda podem ser feitos diversos projetos acadêmicos para explorar na totalidade as possibilidades com o ESP8266 e para facilitar a documentação do mesmo.

Referências Bibliográficas

- [1] 2016. Disponível em: <<https://bridgeaudio.wordpress.com/2015/01/12/conceito-de-automacao-residencial-domotica/>>.
- [2] KOLBAN, N. *Kolban's Book on ESP8266*. 2016. Disponível em: <https://leanpub.com/ESP8266_ESP32>.
- [3] MINATEL, P. “Web server com ESP8266 e IDE Arduino”. 2016. Disponível em: <<http://pedrominate1.com.br/pt/arduino/web-server-com-esp8266-e-ide-arduino/>>.
- [4] SCHWARTZ, M. “Open Home Automation”. 2015. Disponível em: <<https://openhomeautomation.net/control-a-lamp-remotely-using-the-esp8266-wifi-chip/>>.
- [5] 2016. Disponível em: <<https://www.esp8266basic.com/>>.
- [6] 2016. Disponível em: <<http://www.instructables.com/id/ESP8266-WiFi-Controlled-Aircon-Remote/>>.
- [7] ST. AUBIN, D. “4 ways to eliminate ESP8266 resets”. 2016. Disponível em: <<http://internetofhomethings.com/homethings/?p=396>>.
- [8] 2016. Disponível em: <<https://blog.butecopensource.org/primeiros-passos-com-o-esp8266/>>.
- [9] 2016. Disponível em: <<https://nodemcu.readthedocs.io/en/dev/en/>>.
- [10] AFFONSO, V. “Aplicação de tecnologia NFC em circuitos eletrônicos em segurança com utilização de banco de dados em web e microcontrolador Arduino”. 2016.
- [11] EPIFANIO, G. “Monitoramento de variáveis elétricas de um sistema fotovoltaico com Arduino e Android”. 2015.
- [12] 2016. Disponível em: <<http://irving.com.br/esp8266/nodemcu-esp8266-o-modulo-que-desbanca-o-arduino-e-facilitara-a-internet-da>>.

- [13] WANDERSKI, S. “CSS Button Generator”. 2017. Disponível em: <<http://css3buttongenerator.com/>>.
- [14] VEROU, L. “CSS3 Patterns Gallery”. 2017. Disponível em: <<http://lea.verou.me/css3patterns/>>.

Apêndice A

Código - variáveis e bibliotecas

```
// Import required libraries
#include <ESP8266WiFi.h>
#include <ESP8266Ping.h>

// WiFi parameters
const char* ssid = "LCA1";
const char* password = "*****";

//Endereço de IP do roteador
const IPAddress remote_host(192, 168, 1, 139);

//variáveis para contagem de tempo
long ti= millis(); //tempo inicial
long to=millis()+1; // tempo final
long inicio;
long fim;

//Endereço IP desejado
IPAddress ip(192, 168, 1, 139);

// The port to listen for incoming TCP connections
#define LISTEN_PORT          80

// Create an instance of the server
WiFiServer server(LISTEN_PORT);
```

Apêndice B

Código - Função Setup()

```
void setup() {
    Serial.begin(115200);
    delay(10);

    // prepare GPIO
    pinMode(4, OUTPUT);
    digitalWrite(4, 1);

    pinMode(5, OUTPUT);
    digitalWrite(5, 1);

    pinMode(12, OUTPUT);
    digitalWrite(12,1);

    pinMode(13, OUTPUT);
    digitalWrite(13,1);

    pinMode(14, OUTPUT);
    digitalWrite(14, 1);

    pinMode(16, OUTPUT);
    digitalWrite(16,1);

    // Connect to WiFi network
    Serial.println();
    Serial.println();
```

```
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid , password);
//Wait for a WiFi connection
while (WiFi.status() != WL_CONNECTED) {
Serial.println("Conectando...");
delay(10000);
}

Serial.println("");
Serial.println("WiFi connected");

// Start the server
server.begin();
Serial.println("Server started");
Serial.println(WiFi.localIP());
}
```

Apêndice C

Código - Função loop()

```
to=millis(); //Increment the time counter
void loop() {
    if (to-ti>=35000){ // a cada 35s o programa envia um ping
para o roteador para verificar se ainda há conexão com a internet
        ti=to; //restart the time counter
        if(Ping.ping(remote_host)) { //remote_host é
o ip do roteador
isso foi definido no inicio do programa
            Serial.println("Success!!");
            Serial.println(WiFi.status());
        }
        else { //Caso o ping não seja bem sucedido
o programa irá se desconectar e reconectar a rede WiFi
            Serial.println("Error :(");
            Serial.println(WiFi.status());
            WiFi.disconnect();

            WiFi.begin(ssid , password);
            Serial.println("reconnecting....");
            while (WiFi.status() != WL_CONNECTED) {
                delay(500);
                Serial.print(".");
            }
            Serial.println("reconnection successful");
        }
    }
}
```

```

WiFiClient client = server.available();
if (!client) {
    delay(10);
    Serial.print(".");
    return;
}

//wait for the client comand. If the command takes more than
5s the server will take it out.
Serial.println("new client");
inicio =millis();
while(!client.available()){
    fim = millis();
    Serial.print(".");
    delay(100);
    if (fim-inicio >=5000){
        client.stop();
        Serial.println("client time out")
        return;
    }
}

String req = client.readStringUntil('\r');
Serial.println(req);
client.flush();

//web page code

String buf = "";
buf+=" <!DOCTYPE HTML>";
buf+="<html>";
buf+=" <head> <title> OLHT Light </title >";
buf+=" <meta name=\"viewport\" content=\"width=device-width,
initial-scale =1\"> ";

buf+="<style >";
buf+= " <!--";

```

```

buf+= "body {";
buf+= "background-color:black;";
buf+= "background-image:";
buf+= "radial-gradient(white, rgba(255,255,255,.2) 2px,
transparent 40px),";
buf+= "radial-gradient(white, rgba(255,255,255,.15) 1px,
transparent 30px),";
buf+= "radial-gradient(white, rgba(255,255,255,.1) 2px,
transparent 40px),";
buf+= "radial-gradient(rgba(255,255,255,.4),
rgba(255,255,255,.1) 2px, transparent 30px);";
buf+= "background-size: 550px 550px, 350px 350px,
250px 250px, 150px 150px; ";
buf+= "background-position: 0 0, 40px 60px,
130px 270px, 70px 100px;";
buf+= "} ";
buf+= ".btn {";
buf+= "position:static;";
buf+= "top:100px;";
buf+= "left:10px;";
buf+= "margin:0px 7px 0px 0px;";
buf+= "padding:10px;";
buf+= "display:inline-block; ";
buf+= "background:#d93455;";
buf+= "background-image:-webkit-linear-gradient
(top, #d93455, #2b53b8);";
buf+= "background-image:-moz-linear-gradient
(top, #d93455, #2b53b8);";
buf+= "background-image:-ms-linear-gradient
(top, #d93455, #2b53b8);";
buf+= "background-image:-o-linear-gradient
(top, #d93455, #2b53b8);";
buf+= "background-image:linear-gradient
(to bottom, #d93455, #2b53b8);";
buf+= "-webkit-border-radius: 28;";
buf+= "-moz-border-radius: 28;";
buf+= "border-radius: 28px;";
buf+= "font-family: Courier New;";
buf+= "text-align: auto;";

```

```

buf+= " color: #ffffff;";
buf+= " font-size: 20px;";
buf+= " padding: 10px 20px 10px 20px;";
buf+= " text-decoration: none;";
buf+= "}";
buf+= ".btn:hover {";
buf+= " background: #2b53b8;";
buf+= " background-image: -webkit-linear-gradient
(top, #2b53b8, #d93455);";
buf+= " background-image: -moz-linear-gradient
(top, #2b53b8, #d93455);";
buf+= " background-image: -ms-linear-gradient
(top, #2b53b8, #d93455);";
buf+= " background-image: -o-linear-gradient
(top, #2b53b8, #d93455);";
buf+= " background-image: linear-gradient
(to bottom, #2b53b8, #d93455);";
buf+= " text-decoration: none;";
buf+= "}";
buf+= "-->";
buf+= "</style >";
buf+= "</head> ";

buf+= " <body> ";
buf+= " <center >";
buf+= " <h3> <font face= \"Times New Roman\" size= \"5\"
color=#FAFAFA> Laboratorio </font></h3>";
buf+= " <p>";
buf+= "<a class= \"btn\" href= \"?function=led4\"
title= \"Botão 1\">GPIO04</a>";
buf+= "<a class= \"btn\" href= \"?function=led5\"
title= \"Botão 2\">GPIO05</a>";
buf+= "</p> ";
buf+= "<p> ";
buf+= "<a class= \"btn\" href= \"?function=led13\"
title= \"Botão 3\">GPIO13</a>";
buf+= "<a class= \"btn\" href= \"?function=led14\"
title= \"Botão 4\">GPIO14</a>";
buf+= "</p>";

```

```

        buf+= "<p>";
        buf+= "<a class=\"btn\" href=\"?function=led12\"
title=\"Botão 5\">GPIO12</a>";
        buf+= "<a class=\"btn\" href=\"?function=led16\"
title=\"Botão 6\">GPIO16</a>";
        buf+= "</p> ";
        buf+= "<img src =\"http:// traffic . libsyn . com /
vidaestudantil / LCA . png \"
style = \"width : 200px ; height : 100px ; \" > ";
        buf+= "</center > ";
        buf+= " </body > ";
        buf+= " </html > ";

```

```

//send the page to the client
client.print(buf);
client.flush();
//stops the client
client.stop();
Serial.println("Client disonnected");

```

```

//read the client comand and switch the related pin
if (req.indexOf("led5") != -1)
digitalWrite(5, !digitalRead(5));

```

```

else if (req.indexOf("led12") != -1)
digitalWrite(12, !digitalRead(12));

```

```

else if (req.indexOf("led13") != -1)
digitalWrite(13, !digitalRead(13));

```

```

else if (req.indexOf("led14") != -1)
digitalWrite(14, !digitalRead(14));

```

```

else if (req.indexOf("led16") != -1)
digitalWrite(16, !digitalRead(16));

```

```

else if (req.indexOf("led4") != -1)
digitalWrite(4, !digitalRead(4));

```

```
else if (req.indexOf("led4_on") != -1)
digitalWrite(4, 1);
else if (req.indexOf("led4_off") != -1)
digitalWrite(4, 0);
else {
    Serial.println("invalid request");
    client.stop();
}
client.stop();
Serial.println("Client disconnected");
Serial.println(client.connected());
}
```

Apêndice D

Código - Página Web

```
<!DOCTYPE HTML>
<html>
<head> <title> OLHT Light </title>
<meta name="viewport" content="width=device-width,
initial-scale =1">

<style>
<!--
body{
background-color:black;
background-image:
radial-gradient(white, rgba(255,255,255,.2) 2px, transparent 40px),
radial-gradient(white, rgba(255,255,255,.15) 1px, transparent 30px),
radial-gradient(white, rgba(255,255,255,.1) 2px, transparent 40px),
radial-gradient(rgba(255,255,255,.4), rgba(255,255,255,.1) 2px,
transparent 30px);
background-size: 550px 550px, 350px 350px, 250px 250px, 150px 150px;
background-position: 0 0, 40px 60px, 130px 270px, 70px 100px;
}
.btn {
position:static;
top:100px;
left:10px;
margin:0px 7px 0px 0px;
padding:10px;
display: inline-block;
```

```

background: #d93455;
background-image: -webkit-linear-gradient(top, #d93455, #2b53b8);
background-image: -moz-linear-gradient(top, #d93455, #2b53b8);
background-image: -ms-linear-gradient(top, #d93455, #2b53b8);
background-image: -o-linear-gradient(top, #d93455, #2b53b8);
background-image: linear-gradient(to bottom, #d93455, #2b53b8);
-webkit-border-radius: 28;
-moz-border-radius: 28;
border-radius: 28px;
font-family: Courier New;
text-align: auto;
color: #ffffff;
font-size: 20px;
padding: 10px 20px 10px 20px;
text-decoration: none;

}

```

```

.btn:hover {
background: #2b53b8;
background-image: -webkit-linear-gradient(top, #2b53b8, #d93455);
background-image: -moz-linear-gradient(top, #2b53b8, #d93455);
background-image: -ms-linear-gradient(top, #2b53b8, #d93455);
background-image: -o-linear-gradient(top, #2b53b8, #d93455);
background-image: linear-gradient(to bottom, #2b53b8, #d93455);
text-decoration: none;
}

```

—>

```
</style>
```

```
</head>
```

```
<body>
```

```
<center>
```

```
<h3> <font face= "Times New Roman" size="5" color=#FAFAFA> Laboratório
```

```
</font></h3>
```

```
<p>
```

```
<a class="btn" href="\ "?function=led4\" title="Botão 1">GPIO04</a>
```

```

<a class="btn" href="\?function=led5\" title="Botão 2">GPIO05</a>
</p>
<p>
<a class="btn" href="\?function=led13\" title="Botão 3">GPIO13</a>
<a class="btn" href="\?function=led14\" title="Botão 4">GPIO14</a>
</p>
<p>
<a class="btn" href="\?function=led12\" title="Botão 5">GPIO12</a>
<a class="btn" href="\?function=led16\" title="Botão 6">GPIO16</a>
</p>
<img src ="http://traffic.libsyn.com/vidaestudantil/LCA.png"
style="width:200px; height:100px;" >
S</center>
</body>
</html>

```