UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LORENA MARQUES CRUZ

ESTUDO SOBRE FORMAS DE COMUNICAÇÃO REMOTA COM O ESP8266 PARA AUTOMAÇÃO DE TAREFAS

FORTALEZA

2023

LORENA MARQUES CRUZ

ESTUDO SOBRE FORMAS DE COMUNICAÇÃO REMOTA COM O ESP8266 PARA AUTOMAÇÃO DE TAREFAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Sérgio Daher.

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C962e Cruz, Lorena Marques.
Estudo sobre formas de comunicação remota com o ESP8266 para automação de tarefas / Lorena Marques Cruz. – 2023.
65 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2023.
Orientação: Prof. Dr. Sérgio Daher.

1. NodeMCU. 2. Comunicação remota. 3. Automação. I. Título.

CDD 621.3

LORENA MARQUES CRUZ

ESTUDO SOBRE FORMAS DE COMUNICAÇÃO REMOTA COM O ESP8266 PARA AUTOMAÇÃO DE TAREFAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Sérgio Daher.

Aprovada em: 11/07/2023

BANCA EXAMINADORA

_____

Prof. Dr. Sérgio Daher (Orientador)

Universidade Federal do Ceará

_____

Prof. Dr. Arthur Plínio de Souza Braga

Universidade Federal do Ceará

_____

Prof. Dr. Clemilson Costa dos Santos

Universidade Federal do Ceará

AGRADECIMENTOS

À minha família, especialmente meu pai, meu irmão e minhas tias, por todo o esforço dedicado à minha educação e todo o apoio que recebi durante o curso.

Ao meu namorado, por todo incentivo e motivação que recebi nos momentos mais difíceis, por ter escutado minhas preocupações, me aconselhado e por sempre ser alguém com quem eu posso contar, seja nos momentos bons ou ruins.

Aos meus amigos de curso, com quem dividi tantas risadas, conhecimentos, receios, tristezas e alegrias durante toda a graduação.

Ao Prof. Dr. Clemilson Costa dos Santos, que foi meu orientador da bolsa de estudos e de pesquisa. Uma pessoa muito sábia e gentil, com quem sempre aprendi muito.

Ao Prof. Dr. Sérgio Daher, pela orientação e a disponibilidade prestadas no desenvolvimento deste trabalho.

“Nenhuma sociedade que esquece a arte de questionar pode esperar encontrar respostas para os problemas que a afligem.”

(*Zygmunt Bauman*)

RESUMO

Este trabalho apresenta um estudo sobre como estabelecer comunicação remota via Wi-Fi e Bluetooth entre dispositivos usando o chip ESP8266. Foram avaliados três modelos bastante populares de placas de desenvolvimento quanto aos seus recursos, opções de programação, custo e facilidade de uso. Com base na análise realizada, a placa NodeMCU V3 foi escolhida como a plataforma de prototipagem devido ao seu baixo preço de aquisição, suporte embutido para Wi-Fi e simplicidade de implementação. Além de explorar as características e funcionalidades desses componentes, este documento também se concentra no estudo das topologias de cada rede. Para melhor ilustrar e compreender como a comunicação ocorre entre os dispositivos e o ESP8266, foram realizados dois pequenos projetos com o intuito de exemplificar o envio e recebimento de dados por meio do uso de cada tecnologia de comunicação remota. Dessa forma, foi possível constatar a versatilidade do ESP8266 e o seu potencial como uma ferramenta eficaz e acessível para o desenvolvimento de soluções que requerem conectividade sem fio.

Palavras-chave: NodeMCU; Comunicação Remota; Automação;

ABSTRACT

This paper presents a study on how to estabilish remote communication via Wi-Fi and Bluetooth between devices using ESP8266 chip. Three very popular models of development boards were evaluated for their features, programming options, cost and ease of use. Based on the analysis performed, the NodeMCU V3 board was chosen as the prototyping plataform due to its low acquisition price, built-in Wi-Fi support and ease of implementation. In addition to exploring the features and functionality of these components, this document also focuses on studying the topologies of each network. To better illustrate and understand how the communication between devices and the ESP8266 occurs, two small projects were carried out in order to exemplify the sending and receiving of data through the use of each remote communication technology. Therefore, it was possible to verify the versatility of ESP8266 and its potential as an effective and accessible tool for the development of solutions that require wireless connectivity.

Keywords: NodeMCU; Remote Communication; Automation;

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Segundo Black (1998, apud LUZ; KUIAWINSKI, 2006), a automação consiste na implementação de técnicas com o intuito de tornar um determinado processo ou sistema automático, englobando, nesse conceito, tanto os serviços que devem ser realizados quanto a fabricação de produtos e as trocas de informações.

Essas são características que ajudaram a automação a se tornar peça-chave na competitividade de muitas empresas. Sendo possível, ao aplicá-la, reduzir os custos com mão-de-obra humana e com estoque, diminuir o tempo que é empregado no desenvolvimento de projetos, além de tornar mais fácil a modificação, a fabricação e o aumento da qualidade dos produtos (ARAÚJO JÚNIOR; CHAGAS; FERNANDES, 2003).

Diante disso, é fácil associar a palavra automação aos ambientes industriais, mas, apesar de seu desenvolvimento estar fortemente enraizado no processo evolutivo industrial, sua aplicação está ganhando cada vez mais espaço em outras áreas. Assim como Lima (2003) aborda, a implementação dos processos de automação não se limita às indústrias e tem contribuído significativamente com o progresso da engenharia, da ciência etc.

Além disso, pode-se notar a automatização de processos em vários produtos utilizados no dia a dia, como uma máquina de lavar roupas, as portas automáticas dos *shoppings* e as lâmpadas que acendem ou apagam de acordo com a resposta de um sensor de presença. Logo, é possível constatar que a automação é almejada não só pelos grandes empresários, mas também pelos consumidores, passando a ser integrada nas características de diversos produtos.

Com base nesse cenário onde a sociedade demonstra uma crescente busca por maior praticidade, precisão, acessibilidade e conforto, a integração da automação com as tecnologias de comunicação remota ganha bastante relevância, incentivando, assim, o desenvolvimento de aplicações que utilizam internet, bluetooth e sinal infravermelho para o envio e recebimento de dados entre dispositivos.

É possível observar a utilização dessas interações em carros inteligentes, sistemas de automação residencial, como aqueles presentes nas *smart homes,* e em aparelhos de monitoramento remoto de usinas. Dentre essas e outras aplicações, se destaca o conceito de Internet das Coisas ou *Internet of Things* (IoT) que, segundo Neves (2021), consiste nos atuais esforços destinados à conexão de diversos dispositivos físicos às redes de comunicação.

Tendo em vista o número crescente de pessoas e aparelhos conectados à internet, este trabalho visa explorar métodos de comunicação remota com o ESP8266 para a construção de aplicações simples, com o objetivo de disseminar conhecimento acerca do uso de IoT e incentivar o desenvolvimento de soluções de baixo custo para automatização de processos.

1.1 Justificativa

O desejo de automatizar tarefas, sejam elas simples ou complexas, não se restringe ao ambiente industrial. Em uma sociedade que possui uma dependência tecnológica cada vez maior, a demanda por inovações que facilitem a execução de qualquer processo cotidiano é crescente.

Surge, então, a necessidade do desenvolvimento de estudos que garantam diferentes visões acerca dos desafios e das possibilidades decorrentes da implementação das novas tecnologias, mas que também consigam democratizá-las para que um maior número de pessoas consiga usufruí-las.

Nesse contexto, este documento tem o propósito de fornecer uma base de estudos acerca das possibilidades de comunicação remota com o ESP8266 que possa contribuir com o desenvolvimento de pesquisas e projetos de baixo custo na área da Internet das Coisas.

1.2 Objetivos

1.2.1 Objetivo Geral

Esse trabalho foi desenvolvido com o objetivo de explorar algumas possíveis formas de comunicação remota com o ESP8266 como solução viável para automatizar tarefas simples e complexas no contexto da IoT, com o intuito de democratizar o acesso às novas tecnologias e promover inovação acessível a um público mais amplo.

1.2.2 Objetivos Específicos

1. Analisar as topologias dos meios de comunicação remota: Wi-Fi e Bluetooth.
2. Projetar uma solução IoT simples para possibilitar o monitoramento de temperatura e umidade de ambientes para o estudo das comunicações remotas.
3. Propor os materiais a serem utilizados e o esquemático simplificado da solução.

4. Detalhar como o fluxo de dados ocorre durante o uso de cada tecnologia de comunicação sem fio.

1.3 Organização do Trabalho

O presente trabalho foi estruturado da seguinte maneira:

a) Capítulo 2: são discutidos os conceitos de microcontrolador e placa de desenvolvimento, bem como o papel de cada um no desenvolvimento de projetos IoT. Além disso, serão analisadas três placas, destacando as vantagens e desvantagens do uso de cada uma.
b) Capítulo 3: contém uma rápida revisão bibliográfica sobre os cenários de conectividade no âmbito da IoT, uma análise de como o NodeMCU pode atuar neles e como são as topologias dos meios de comunicação remota que serão utilizados.
c) Capítulo 4: apresenta os possíveis modos de operação Wi-Fi do ESP8266 e um pequeno projeto para exemplificar o uso desse tipo de comunicação.
d) Capítulo 5: possui um exemplo de como a funcionalidade bluetooth pode ser adicionada e utilizada no ESP8266.

2. MICROCONTROLADORES E PLACAS DE DESENVOLVIMENTO APROPRIADAS PARA IOT

Uma das principais características que tornam o desenvolvimento de soluções IoT bastante desejado, é a capacidade de otimizar sistemas e processos maneira inteligente e eficiente. Diante dessa importante qualidade, destaca-se a atuação de um componente: o microcontrolador. O uso desse elemento é essencial para a implementação de algoritmos sofisticados com o intuito de gerenciar dispositivos, adquirir e processar dados.

Nessa seção, são discutidos os conceitos de microcontroladores, destacando sua função e importância para a automatização de tarefas. Além disso, são abordadas alternativas para facilitar, acelerar e reduzir os custos decorrentes do desenvolvimento de soluções que necessitam desses chips.

2.1 Microcontroladores

Segundo Souza (2005), é possível definir um microcontrolador como um pequeno componente eletrônico, provido de uma inteligência programável, empregado no controle de processos lógicos. Nele, estão embutidos vários recursos necessários para o controle de periféricos, tais como: memória de programa, memória de dados, *timers*, portas de entradas e/ou saídas paralelas, conversores analógicos-digitais etc.

Os microcontroladores estão bastante presentes no cotidiano das pessoas. Eles são encontrados na composição de controles remotos, eletrodomésticos, ferramentas de medição, brinquedos e outros sistemas embarcados. Mesmo despercebidos, esses componentes são grandes difusores da automação de processos, pois são capazes de executar tarefas específicas com grande rapidez e precisão.

As instruções a serem processadas pelos microcontroladores são desenvolvidas e gravadas no formato de programa. Sendo assim, é possível afirmar que a aplicabilidade desses componentes eletrônicos está vigorosamente relacionada a sua programação (OLIVEIRA NETO; QUEIROGA; MONTEIRO, 2012).

Dentro da IoT, os microcontroladores recebem o importante papel de se comunicar com os objetos físicos, como sensores e atuadores, para que eles possam ser monitorados e/ou

controlados à distância, constituindo, assim, uma espécie de ponte entre os periféricos e a interface digital à qual o usuário final tem acesso.

2.2 Uso de placas de desenvolvimento em projetos IoT

Elaborar um circuito eletrônico desde o início utilizando microcontroladores não é uma tarefa fácil e requer um amplo conhecimento na área de eletrônica. Além disso, considerando que durante o processo de desenvolvimento seria necessário produzir uma placa para cada versão do produto, essa é uma prática que se tornaria bastante custosa para o projeto.

Com base nisso, considerando que um conjunto de repetidos processos de avaliação e *feedback* são aplicados durante o desenvolvimento de uma solução IoT para definir os componentes de hardware e software mais adequados, é necessário considerar alternativas que agilizem as etapas de prototipagem e refinamento, tornando-as mais rápidas e baratas. Nesse sentido, o uso de plataformas como Arduino e Raspberry Pi pode ser considerada como uma opção viável em termos de tempo e custo.

O Arduino e o Raspberry Pi são plataformas de hardware constituídas de uma placa de circuito impresso (PCB) com um chip e outros circuitos adicionais disponíveis para o suporte do mesmo. Elas são montadas e utilizadas com objetivo de facilitar a prototipagem e o desenvolvimento do firmware.

Essas placas possuem um ambiente de programação simples de código aberto (*open source*) e vários módulos e kits de desenvolvimento de fácil integração disponíveis no mercado. Com isso, a sua utilização requer menos tempo e um menor custo do que o projeto e a fabricação de placas de circuito impresso a cada iteração dos testes. Por isso, esse capítulo tem o objetivo de explorar as placas de desenvolvimento como um todo, destacando suas vantagens e desvantagens.

De acordo com Swathi, Sandeep e Romani (2018), as placas de desenvolvimento em soluções IoT são escolhidas com base em cinco características:

- Custo;
- Suporte e opções para programação;
- Especificações de memória, processamento etc;
- Compatibilidade com sensores e atuadores;

- Confiabilidade do fornecedor.

No mercado atual, existe uma ampla diversidade de placas de desenvolvimento, cada uma com suas características e limitações. No entanto, esse trabalho analisa apenas três modelos bastante populares: Arduino UNO, Raspberry Pi 4 Model B e NodeMCU V3. A análise foi feita com base nos quatro primeiros pontos destacados anteriormente, deixando de fora a confiabilidade do fornecedor, tendo em vista que a comercialização não é um ponto a ser discutido aqui.

2.2.1 Arduino UNO

A plataforma Arduino foi introduzida ao mundo em 2005, como uma alternativa mais didática e mais barata para a criação de dispositivos eletrônicos capazes de interagir com o meio em que estão inseridos através de sensores e atuadores. As placas de desenvolvimento são construídas com microcontroladores das famílias Atmel AVR de 8-bit ou Atmel ARM de 32-bit. Esses chips podem ser facilmente programados utilizando as linguagens C ou C++ através do software Arduino IDE (LOUIS, 2016).

O fato do Arduino ser uma plataforma *open source* e *open hardware* possibilita que tanto seu hardware quanto seu software possam ser copiados e/ou modificados por terceiros, o que, por sua vez, permite o surgimento e a comercialização de novas versões, nacionais e internacionais, das placas. Dessa forma, os preços se tornam mais acessíveis e a ferramenta pode ser constantemente melhorada.

De toda a família Arduino, a placa Arduino UNO, que utiliza o microcontrolador ATmega328P, é a mais popular e, também, a mais documentada. A palavra “UNO” remete ao número um em italiano e foi escolhida para marcar o lançamento do software Arduino IDE 1.0 em 2010 (SWATHI; SANDEEP; RAMANI, 2018). É possível encontrar vários modelos de qualidade, produzidos por diferentes fabricantes, disponíveis para compra em sites e lojas físicas de eletrônica e robótica.

Figura 1 - Placa de desenvolvimento Arduino UNO.

Fonte: EMBARCADOS, Arduino UNO.[1]

A alimentação dessa placa pode ocorrer via USB ou por uma fonte de alimentação externa. Existe um regulador de tensão localizado próximo à entrada serial universal para que as variações na tensão de entrada sejam ajustadas para 5V (FERRONI et al., 2022).

Em relação ao armazenamento de dados, na Tabela 1, é possível verificar as especificações de memória referentes ao Arduino UNO.

Tabela 1 - Especificações de Memória do Arduino UNO.

Memória	Capacidade
Flash	32 KB
SRAM	2 KB
EEPROM	1 KB

Fonte: De autoria própria.

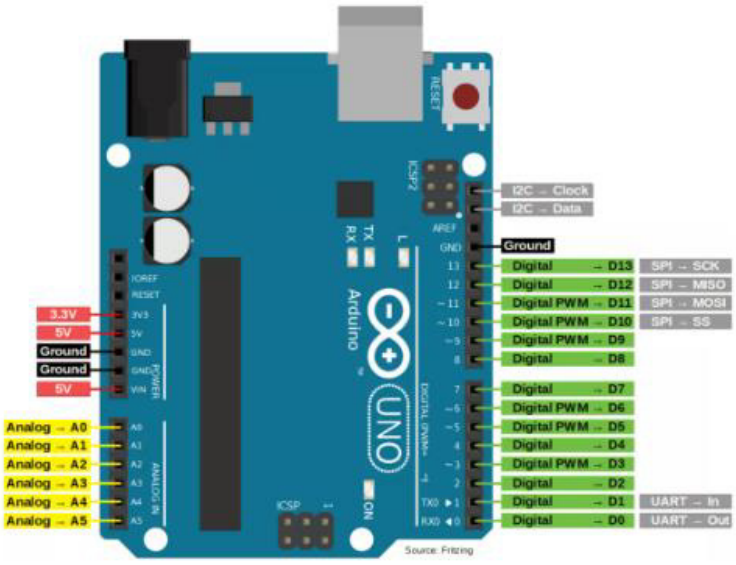
Na memória Flash, são armazenados os arquivos referentes ao programa (*sketch*) responsável por definir as instruções que o hardware deverá executar. Nela, 5 dos 32 KB, são utilizados pelo programa inicializador (*bootloader*). A SRAM é conhecida como memória de acesso aleatório estática e é nesse espaço que as variáveis utilizadas no programa serão manipuladas durante a execução do mesmo. Já a EEPROM, por ser um tipo de memória não-volátil, é responsável por armazenar as eventuais informações que os desenvolvedores desejam salvar.

[1] EMBARCADOS. **Arduino UNO**. Disponível em: https://embarcados.com.br/arduino-uno/. Acesso em: 02 nov. 2022.

Quanto ao processamento, o Arduino UNO possui uma velocidade de *clock* de 16MHz. Isso significa que a placa é capaz de executar 16 milhões de instruções por segundo, o que parece ser muito, mas não é. Isso, pois é preciso levar em consideração tudo o que o Arduino necessita para realizar tarefas simples. Em muitos projetos, os ciclos de clock são compartilhados com cálculos, comunicação I2C, leitura e escrita de pinos etc. O ato de ligar e desligar um LED pode custar até 50 ciclos de clock (ROBINSON, 2016).

Como pode ser observado na figura 2, a placa Arduino UNO possui 6 entradas analógicas e 14 pinos para entradas e saídas digitais, dos quais apenas quatro podem ser utilizados como saídas PWM. Os conectores fêmea da placa facilitam a sua integração com *protoboards* e outros módulos Arduino.

Figura 2 - Pinos do Arduino UNO



Fonte: LOBO DA ROBÓTICA, Arduino Uno Pinout – Desvendando O Arduino.[2]

Esses módulos, também conhecidos como *shields*, são placas com funcionalidades específicas utilizadas com o intuito de complementar as funções de uma placa. Existe uma grande variedade desses componentes no mercado. Na Figura 3, é possível observar módulos de: relés, displays LCD, sensores, ethernet, *joysticks* etc.

[2] LOBO DA ROBÓTICA. **Arduino Uno Pinout – Desvendando O Arduino.** Disponível em: https://lobodarobotica.com/blog/arduino-uno-pinout/. Acesso em 03 nov. 2022.

Figura 3 - Módulos Arduino.

Fonte: RANDOM NERD TUTORIALS, 25 Useful Arduino Shields That You Might Need to Get. [3]

Ao analisar as características apresentadas, constata-se que a facilidade de utilização do Arduino UNO e a grande compatibilidade com *shields* disponíveis no mercado são pontos fortes a serem considerados junto com a grande disponibilidade de tutoriais, fóruns e exemplos proporcionados por uma comunidade *online* bastante ativa. Porém, não é a melhor opção para soluções que necessitem de uma grande capacidade de processamento e memória.

A principal desvantagem de utilizar o Arduino UNO como placa principal em projetos IoT é a falta de suporte embutido para redes sem fio. Contudo, isso pode ser contornado utilizando um módulo WiFi Esp8266, o que tornaria o custo do projeto um pouco mais elevado. Além disso, o Node.js, um ambiente de execução JavaScript server-side, também pode ser utilizado junto com o Socket.IO, uma biblioteca para comunicação bidirecional baseada em eventos entre um navegador e um servidor web, para possibilitar a interação entre o usuário e o arduino via internet.

2.2.2 Raspberry Pi

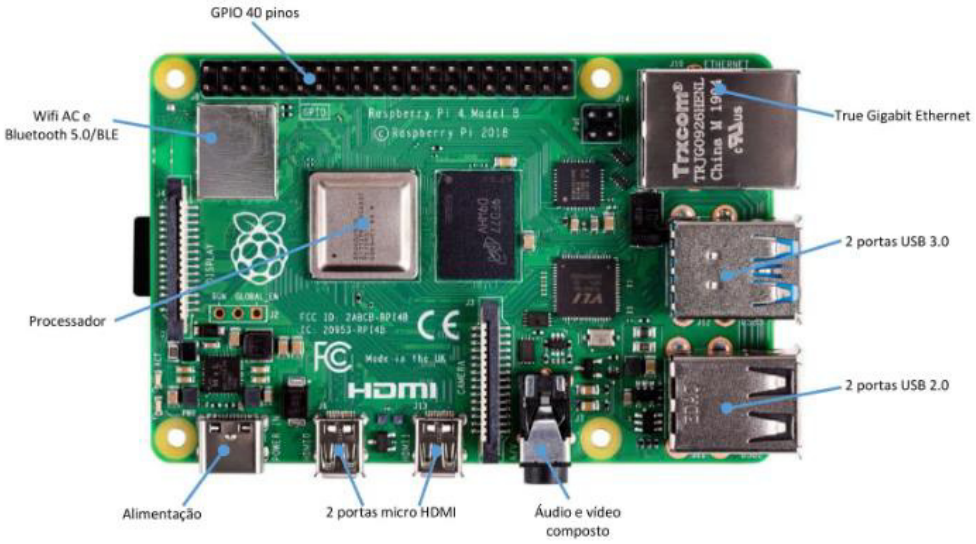
Criado com o nobre propósito de incentivar e fortalecer o ensino das ciências da computação em escolas do Reino Unido e em outros países em desenvolvimento, o Raspberry Pi consiste em um poderoso computador de tamanho reduzido constituído por uma única placa com componentes integrados, capaz de realizar todos os tipos de tarefas de computação e

[3]RANDOM NERD TUTORIALS. **25 Useful Arduino Shields That You Might Need to Get**. Disponível em: https://randomnerdtutorials.com/25-arduino-shields/. Acesso em: 8 nov. 2022.

interagir com diversos tipos de dispositivos através das portas de entrada e saída de uso geral (NAYYAR; PURI, 2016).

As placas de desenvolvimento Raspberry Pi utilizam chips do tipo SoC da família Broadcom. Esse tipo de componente possui uma arquitetura bem mais complexa do que um microcontrolador, podendo ainda contê-lo em sua estrutura.

Figura 4 - Raspberry Pi 4 Model B



FONTE: MAKERHERO, Lançamento! Raspberry Pi 4 Model B.[4]

Dentre os modelos disponíveis de placa, destaca-se o Raspberry Pi 4 Model B. Este modelo, que pode ser observado na Figura 4, vem equipado de um processador Broadcom BCM2711, com quatro núcleos de processamento independentes do tipo Cortex-A72 (*quad-core*) com arquitetura ARM v8 de 64 bits, capaz de rodar a 1.5 GHz. A memória RAM depende do modelo escolhido e pode variar entre 2, 4, e 8 GB. Além disso, a placa também possui um *slot* para cartão microSD (RASPBERRY, 2019).

Segundo o seu manual, em relação à conectividade, o Raspberry Pi 4 Model B apresenta uma boa variedade de opções. A comunicação com fio pode ocorrer por meio da porta Gigabit Ethernet ou através das quatro portas USB, sendo duas delas do tipo USB 2.0 e duas do tipo USB 3.0. Além disso, esta placa também apresenta a possibilidade de conectividade

[4] MAKERHERO. **Lançamento! Raspberry Pi 4 Model B**. Disponível em: https://www.makerhero.com/blog/lancamento-raspberry-pi-4-model-b/. Acesso em 05 nov. 2022.

sem fio integrada, podendo estabelecer comunicação via WiFi ou Bluetooth (RASPBERRY, 2019).

Esse modelo de Raspberry Pi ainda conta com 40 portas de entrada e saída de uso geral (GPIO), além de ser compatível com uma grande quantidade de *Shields* disponíveis no mercado. Assim como os computadores usuais, também apresenta opções de saída para áudio e vídeo. A placa possui duas portas HDMI capazes de conectar dois monitores simultaneamente com resoluções de 4k a 60 ou 30 quadros por segundo (RASPBERRY, 2019).

Quanto à programação, os modelos de placa Raspberry Pi suportam Python, C/C++ e Scratch por padrão, contudo, é possível utilizar outras linguagens desde que seus compiladores ou interpretadores sejam instalados no sistema operacional Raspbian, que é baseado no Debian Linux.

Todas essas características tornam o dispositivo em questão bem atrativo para o desenvolvimento de projetos IoT, principalmente para aqueles que necessitam de uma grande potência de processamento, memória e uma boa variedade de opções de conectividade. Assim como o Arduino, o Raspberry Pi possui uma comunidade bastante ativa e, por isso, existe muito conteúdo sobre seu uso na internet.

Em contrapartida, é preciso destacar que o Raspberry Pi 4 Model B possui um custo bem mais elevado em comparação a outras placas, como o Arduino UNO e o NodeMCU. Além disso, com um comprimento de 88 mm, uma largura de 58 mm e uma altura de 19,5 mm, as dimensões da placa podem ser consideradas inadequadas para projetos que necessitem de um tamanho reduzido.

2.2.3 NodeMCU

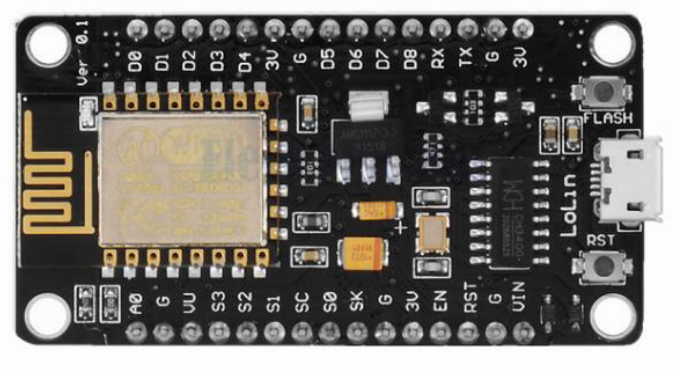
Lançado em 2014, O NodeMCU consiste em uma versátil plataforma de desenvolvimento *open source* e *open hardware* criada para ser utilizada em projetos IoT. Seu circuito conta com a presença do chip ESP8266, um SoC que possui todas as características essenciais de um computador, tais como: CPU, RAM, comunicação wifi, um sistema operacional RTOS e também um SDK (YUAN, 2017).

Dentre os modelos disponíveis, o V3 é bastante procurado. Essa placa pode ser alimentada via cabo USB/Micro USB e é constituída por um ESP8266 de 32 bits que, segundo seu *datasheet*, é capaz de operar em uma frequência de até 160 MHz, possui uma memória flash

de 4MB e uma memória RAM utilizável de aproximadamente 50kB. Além disso, existem 11 portas de entrada e saída (GPIO) para conexão com módulos e sensores.

Esse modelo, que pode ser observado na Figura 5, possui um comprimento aproximado de 58mm e uma largura em torno de 32 mm. Em comparação às outras placas analisadas anteriormente, é a que possui as menores dimensões, isso a torna uma boa opção para projetos que necessitam de um tamanho relativamente pequeno.

Figura 5 - NodeMCU V3



FONTE: NodeMCU – Uma plataforma com características singulares para o seu projeto IoT.[5]

O NodeMCU V3 contém um módulo ESP-12. Esse componente possui o SoC ESP8266 e outros recursos necessários. É importante destacar que existem duas variantes desse mesmo módulo: o ESP-12e e sua versão melhorada, o ESP-12f. Embora sejam muito semelhantes, há uma diferença no design da antena.

Com isso, é necessário ressaltar que o NodeMCU V3 facilita bastante a utilização do ESP8266, contudo, existem diversos módulos baseados no mesmo chip com especificações diferentes, como o ESP-07 e o ESP-01. Então, é preciso saber qual será o módulo utilizado no projeto e verificar se a placa de desenvolvimento suporta o modelo escolhido.

A programação do NodeMCU pode ser realizada por meio da linguagem LUA, com o uso de softwares como: ESPlorer, LuaLoader e NodeMCU PyFlasher. Além disso, também é possível utilizar a IDE do Arduino para programar essa placa, o que configura uma grande vantagem para quem já está familiarizado e tem facilidade com o uso desse ambiente de desenvolvimento.

[5] MASTER WALKER ELETRONIC SHOP. **NodeMCU – Uma plataforma com características singulares para o seu projeto IoT.** Disponível em: https://blogmasterwalkershop.com.br/embarcados/nodemcu/nodemcu-uma-plataforma-com-caracteristicas-singulares-para-o-seu-projeto-iot. Acesso em: 15 abr. 2023.

Esse modelo também é o mais barato entre as placas analisadas nesse documento e, assim como elas, conta com uma grande comunidade *online* que disponibiliza muitos recursos gratuitos que auxiliam no desenvolvimento de projetos. O NodeMCU também é compatível com vários *shields* disponíveis no mercado.

Como mencionado anteriormente, a placa possui suporte nativo para Wi-Fi, o que já a torna uma opção viável para soluções que requerem comunicação remota, mas outro ponto positivo desse modelo é a possibilidade da utilização do ESP-NOW, um protocolo que permite a comunicação sem fio entre dispositivos ESP, como o ESP8266 e o ESP32.

Diante disso, é possível constatar que as principais vantagens do uso do NodeMCU V3 em projetos IoT são as opções de programação, o suporte nativo para Wi-Fi, a compatibilidade com o ESP-NOW, o baixo custo de aquisição e o suporte oferecido pela comunidade desenvolvedora. Contudo, a placa possui recursos limitados, o que pode dificultar a execução de tarefas mais complexas que necessitem de uma capacidade maior de processamento e memória.

2.2.4 Qual placa será utilizada?

Diante das informações apresentadas nos tópicos anteriores, a melhor placa seria aquela que melhor atende às necessidades do projeto em questão. Como esse documento tem como objetivo explorar as possibilidades de comunicação remota entre dispositivos no desenvolvimento de soluções IoT de baixo custo, será utilizada a placa NodeMCU V3 que já vem com suporte embutido para Wi-Fi.

Além disso, a partir de uma pesquisa sobre o preço dos componentes em questão em três sites de venda, constatou-se que a placa NodeMCU V3 possui o menor preço de aquisição. O levantamento dos valores está presente na tabela a seguir.

Tabela 2 - Preços das Placas de Desnevolvimento

Site	Preço (R$)		
	Arduino UNO	NodeMCU V3	Rasp. 4 Model B
Mercado Livre	54,00	34,20	835,65
Amazon	51,75	41,72	857,97
AliExpress	25,79	15,64	640,43

Fonte: De autoria própria.

3 MEIOS DE COMUNICAÇÃO SEM FIO E SUAS TOPOLOGIAS

Drumond (2017) aborda que existem duas principais formas de conectar os dispositivos de uma solução no âmbito da Internet das coisas. Em uma delas, todos os elementos presentes na estrutura possuem, individualmente, a capacidade de se comunicar à internet. Na outra, dispositivos secundários estão conectados a um dispositivo central que, por sua vez, possui conexão com a internet.

O NodeMCU V3 é capaz de atuar nesses dois cenários. No primeiro, a conexão Wi-Fi é suficiente para estabelecer a comunicação com a internet. Já no segundo, é possível assumir o papel de elemento central ou secundário. Nesse caso, levando em consideração apenas as comunicações remotas, o Bluetooth se destaca como uma tecnologia auxiliar que permite a troca direta de informações entre o NodeMCU e os outros dispositivos.

Antes de explorar como as comunicações remotas via Wi-Fi e Bluetooth podem ser estabelecidas utilizando a placa NodeMCU V3, se faz necessário entender um pouco sobre o conceito de topologia e quais os padrões utilizados por essas tecnologias com o intuito de melhor compreender como os dispositivos se conectam e para determinar quais seriam os cenários mais adequados para a sua aplicação.

3.1 Topologias

Fernández (2015) define o termo topologia como a distribuição geográfica entre os nós e arestas de uma rede. Os nós, também conhecidos como *nodes*, são referentes aos dispositivos presentes na rede, como computadores, celulares, impressoras e servidores. Já as arestas, ou *links*, consistem nas conexões que ligam os nós. Com base nisso, pode-se inferir que a palavra topologia refere-se ao arranjo dos elementos integrantes de uma rede, ou seja, como os dispositivos estão conectados e como as informações são trocadas entre eles.

Existem duas categorias de topologias utilizadas para estruturar as redes de dados: a física e a lógica. A topologia física, como o próprio nome indica, envolve a disposição física das conexões entre os nós, cabos, fios etc. A topologia lógica, por sua vez, concentra-se no fluxo de dados que ocorre por meio da infraestrutura física da rede, seguindo as configurações predefinidas (FREUND, 2009, p. 176, apud MARROCO, 2015, p. 16).

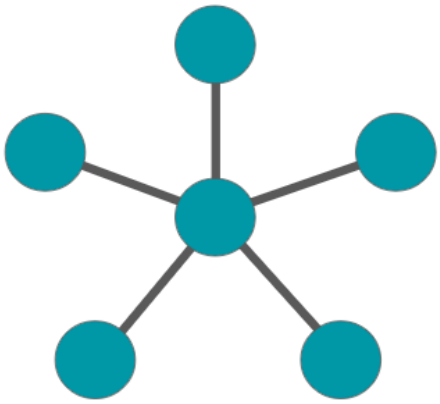
Fernández (2015) também aborda que existem dois tipos básicos de topologia, a ponto-a-ponto e a multiponto. A combinação das características desses tipos fundamentais é realizada com a finalidade de obter mais recursos e possibilita o surgimento de redes mais complexas chamadas estruturas mistas, como: estrela, anel e barramento. A seguir, serão discutidas com mais detalhes as topologias referentes ao Wi-Fi e Bluetooth.

3.2 Wi-Fi

O termo Wi-Fi se originou da abreviação de "*Wireless Fidelity*" e é utilizado para representar um conjunto de especificações técnicas para as redes locais sem fio, conhecidas como WLAN (*Wireless Local Area Network*). Essa tecnologia possibilita a conexão de diversos dispositivos a uma rede local, sem a necessidade de cabos, permitindo que acessem a internet ou os recursos compartilhados da rede de qualquer lugar, desde que estejam dentro da área de cobertura do sinal (ALECRIM, 2008).

A topologia mais comumente encontrada nas redes Wi-Fi, principalmente naquelas residenciais, é a estrela (ARAR, 2022). Como pode ser observado na figura a seguir, essa topologia é caracterizada por um conjunto de nós ligados a um nó central que, segundo Freund (2009, p. 177, apud MARROCO, 2015, p. 18), recebe toda a confiabilidade da rede e, caso seja danificado, pode comprometer o funcionamento de todo o sistema.

Figura 6 - Topologia Estrela



Fonte: De autoria própria.

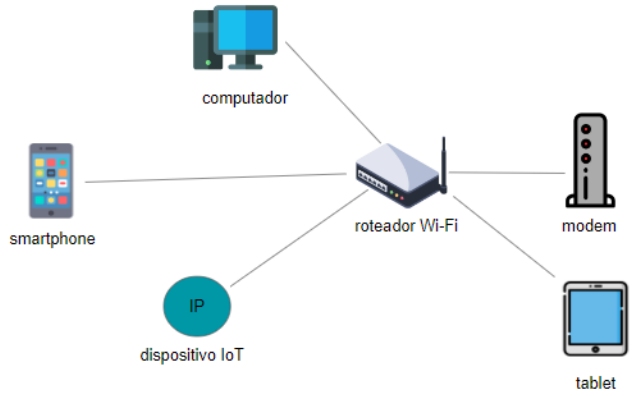
Em redes cabeadas, o ponto central presente na estrutura estrela pode ser um *hub*, *switch* ou um roteador, esse nó será o responsável por permitir a transmissão dos dados de um

dispositivo para o outro. Nas redes Wi-Fi, é mais comum encontrar um roteador sem fio, também conhecido como roteador Wi-Fi, assumindo essa função.

O roteador Wi-Fi é responsável por criar o ponto de acesso ao qual os dispositivos irão se conectar, a área de cobertura desse sinal é conhecida como *hotspot*. Esse aparelho, geralmente, está conectado a um modem que fornece o sinal de internet. Logo, além de permitir que os dispositivos se comuniquem dentro da rede local, o roteador se torna responsável por gerenciar o fluxo dos pacotes de dados entre os demais nós da rede e a internet (ALECRIM, 2019).

Diante disso, pode-se entender que o roteador Wi-Fi assume o papel de uma ponte entre os nós ligados a ele. Ao ser adicionado na rede, um dispositivo IoT atuaria como um nó adjacente da mesma forma que um celular, um computador, um *tablet* ou qualquer outro aparelho conectado ao ponto de acesso gerado pelo roteador sem fio. Esse esquemático pode ser observado na figura a seguir.

Figura 7 - Esquemático de uma rede Wi-Fi



Fonte: De autoria própria.

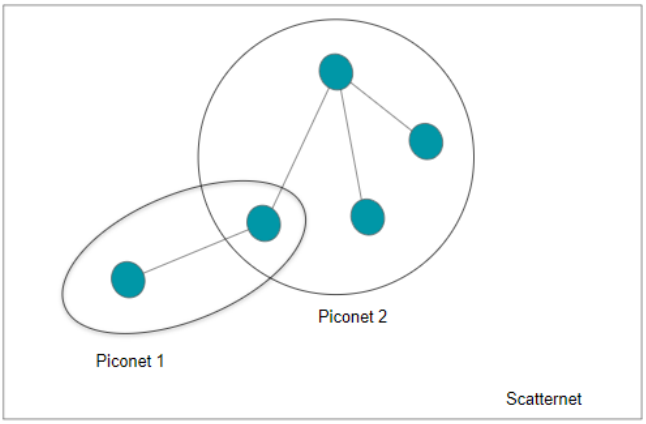
3.3 Bluetooth

O bluetooth consiste em uma tecnologia de curto alcance que utiliza radiofrequência para estabelecer comunicação sem fio entre dispositivos (VERMA; SINGH; KAUR, 2015). Por proporcionar uma comunicação rápida e eficiente, é amplamente utilizado para a conexão de fones de ouvido, caixas de som, impressoras, celulares e outros.

A rede formada pelos dispositivos bluetooth é conhecida como piconet, nela até 7 nós escravos podem estar conectados a um mestre. Ela pode ser definida como piconet simples,

quando há apenas um mestre e um escravo no arranjo, ou piconet *multi-slave*, quando há mais de um escravo conectado ao dispositivo mestre. Um conjunto de piconets pode formar uma rede mais complexa chamada de scatternet, como pode ser observado na figura a seguir.

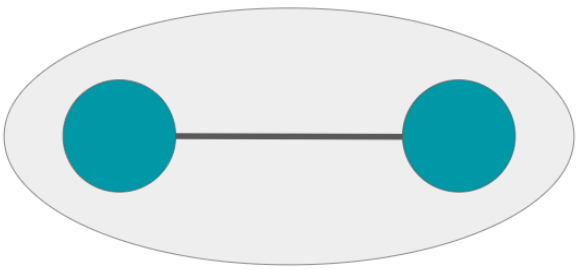
Figura 8 - Rede Scatternet



Fonte: De autoria própria.

A rede mais comumente utilizada, principalmente quando se refere ao bluetooth clássico, é a piconet simples. Ela pode ser identificada na conexão entre fones de ouvido e celulares. A topologia dessa rede é a ponto-a-ponto, também conhecida como P2P (*peer-to-peer*). Nessa configuração, que pode ser observada na Figura 9, dois dispositivos estão conectados diretamente entre si por meio de uma aresta.

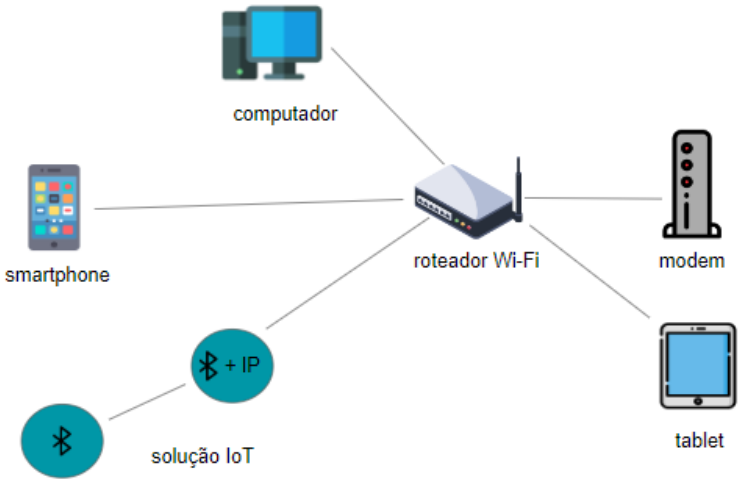
Figura 9 - Topologia ponto-a-ponto de uma rede piconet simples



Fonte: De autoria própria.

É possível utilizar a comunicação via Bluetooth para conectar dispositivos à internet (VERMA; SINGH; KAUR, 2015). Contudo, o Bluetooth também demonstra uma grande utilidade no segundo cenário descrito por Drummond, onde é capaz de atuar como uma ponte entre dispositivos que não possuem a capacidade de se conectar ao Wi-Fi, como pode ser observado a seguir.

Figura 10 - Ponte Bluetooth na rede Wi-Fi



Fonte: De autoria própria.

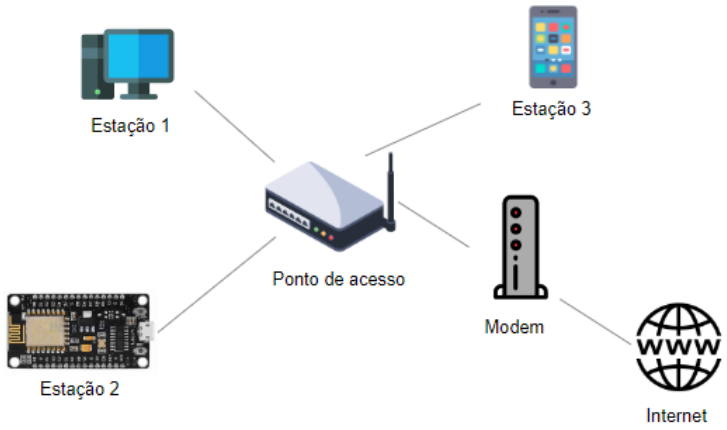
4 COMUNICAÇÃO VIA WI-FI

Quanto às possibilidades do uso de Wi-Fi para comunicação remota entre dispositivos, o ESP8266 apresenta uma boa diversidade de modos de operação. Essas configurações determinam como o chip irá se conectar e interagir com a rede. Logo, faz-se necessário melhor compreender cada opção para que seja possível aproveitar ao máximo a capacidade de conectividade desse chip.

4.1 Modos de comunicação Wi-Fi

O ESP8266 pode atuar em três modos operacionais Wi-Fi. O primeiro é conhecido como modo STA (*Station*), nele, como indicado na Figura 11, a placa é conectada a uma rede Wi-Fi externa e atua como uma estação, ou seja, um nó adjacente da rede, enquanto o roteador sem fio recebe o papel de nó central. Nesse caso, qualquer dispositivo conectado à mesma rede é capaz de se comunicar com a placa por meio do roteador, mesmo sem a presença de sinal de internet.

Figura 11 - NodeMCU operando no modo STA



Fonte: De autoria própria.

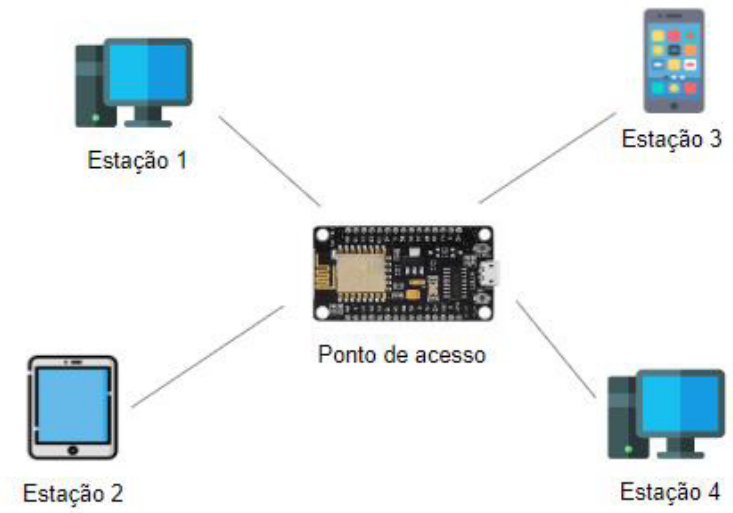
A utilização de uma rede externa que possui conexão à internet proporciona uma grande vantagem: o acesso a recursos remotos, como o armazenamento em bancos de dados. Isso permite que as informações enviadas pela placa sejam consumidas por aplicações com ferramentas de análise mais sofisticadas que possam ser acessadas de qualquer lugar do mundo.

Além disso, com o acesso à internet, o NodeMCU pode requisitar dados de outros servidores por meio do protocolo HTTP e interpretá-los de acordo com a lógica programada no ESP8266, possibilitando a comunicação entre usuários situados em localidades mais distantes e a placa, não restringindo a troca de informações ao limite geográfico da rede Wi-Fi à qual está conectada.

O segundo modo é o AP (*Access Point*). Como observado na Figura 12, o ESP8266 atua como um ponto de acesso, gerando sua própria rede Wi-Fi à qual outros aparelhos podem se conectar. Nesse caso, a comunicação entre os dispositivos fica limitada à sua rede individual, pois não há Wi-Fi externo, logo, também não há acesso à internet.

Uma das grandes vantagens desse modo é a possibilidade de estabelecer comunicação com a placa em lugares onde redes de Wi-Fi externas não possuem alcance ou o sinal é instável. Contudo, como mencionado anteriormente, não há conexão com a internet, então, os recursos de análise de dados e armazenamento são limitados pela capacidade do dispositivo.

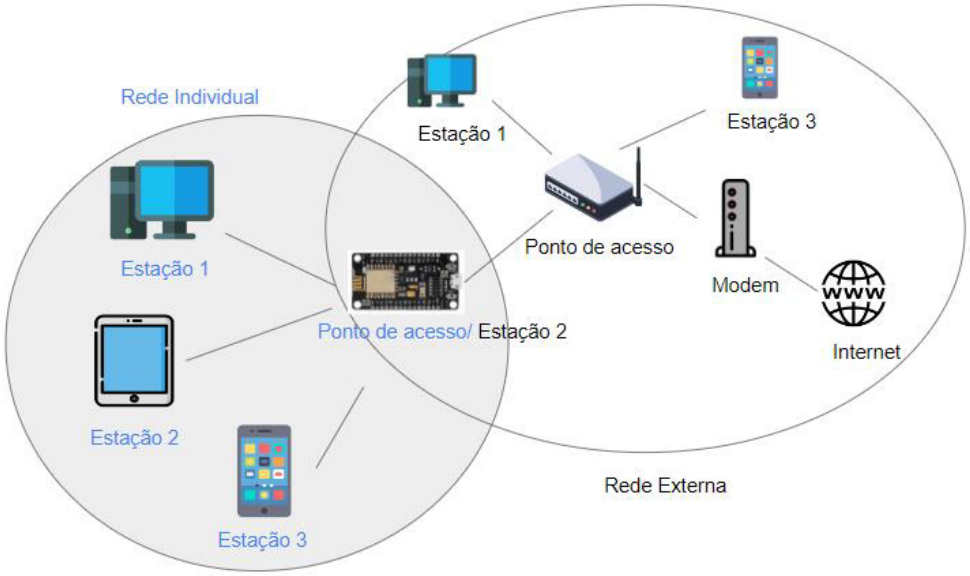
Figura 12 - NodeMCU operando no modo AP



Fonte: De autoria própria.

O terceiro modo, ilustrado na Figura 13, permite que a placa opere tanto como uma estação quanto um ponto de acesso (AP + STA). É uma característica que, além de oferecer versatilidade ao projeto, possibilita uma maior resiliência à solução, pois caso o sinal de Wi-Fi externo fosse perdido, ainda seria possível utilizar o Wi-Fi próprio.

Figura 13 - NodeMCU operando no modo AP+STA



Fonte: De autoria própria.

4.2 Exemplo 1: Monitoramento de ambientes através da rede local

Para demonstrar como o ESP8266 pode enviar e receber dados via Wi-Fi, foi desenvolvido um projeto simples de monitoramento de temperatura e umidade de ambientes utilizando o NodeMCU V3. Nesse pequeno exemplo, a placa monitora as informações de um sensor DHT11. Caso os valores recebidos ultrapassem os limites estabelecidos, um LED é acionado automaticamente.

Para o projeto, foram utilizados os materiais listados na Tabela 3.

Tabela 3 - Materiais Utilizados para o Projeto de Comunicação Wi-Fi (continua)

Quantidade	Material
1	Placa NodeMCU V3
1	Protoboard de 830 pontos
1	Sensor de Umidade e Temperatura DHT11
1	LED Verde
1	LED Vermelho
1	LED Azul

Tabela 3 - Materiais Utilizados para o Projeto de Comunicação Wi-Fi (conclusão)

Quantidade	Material
3	Resistores de 220 ohms para o LED
1	Resistor de 10k ohms para o sensor
13	Jumpers macho-macho

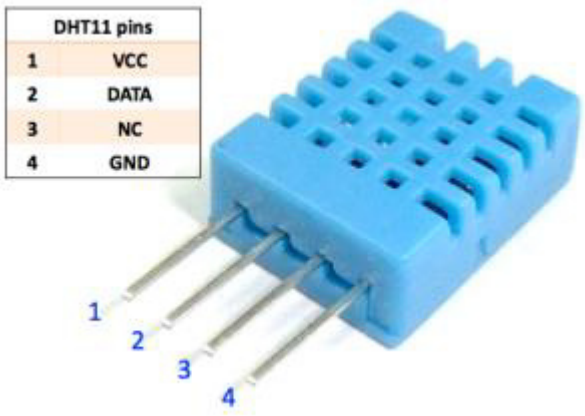
Fonte: De autoria própria.

Para a programação do chip, foi utilizada a IDE do Arduino com a linguagem baseada em C/C++. O código utilizado pode ser verificado no APÊNCICE A deste documento.

4.2.1 Esquemático das ligações;

O sensor DHT11, mostrado na Figura 14, pode ser alimentado por uma tensão de 3,3 a 5Vdc e é capaz de medir temperatura na faixa de 0 a 50° C com uma precisão de +/- 2°C. Já a faixa de medição de umidade é de 20 a 80% com uma precisão de +/- 5%. Para que seja possível realizar a leitura desse sensor, é necessário instalar as bibliotecas DHT a Adafruit Sensor Master.

Figura 14 - Pinagem do Sensor DHT11



FONTE: Sensor DHT11 com Arduino.[6]

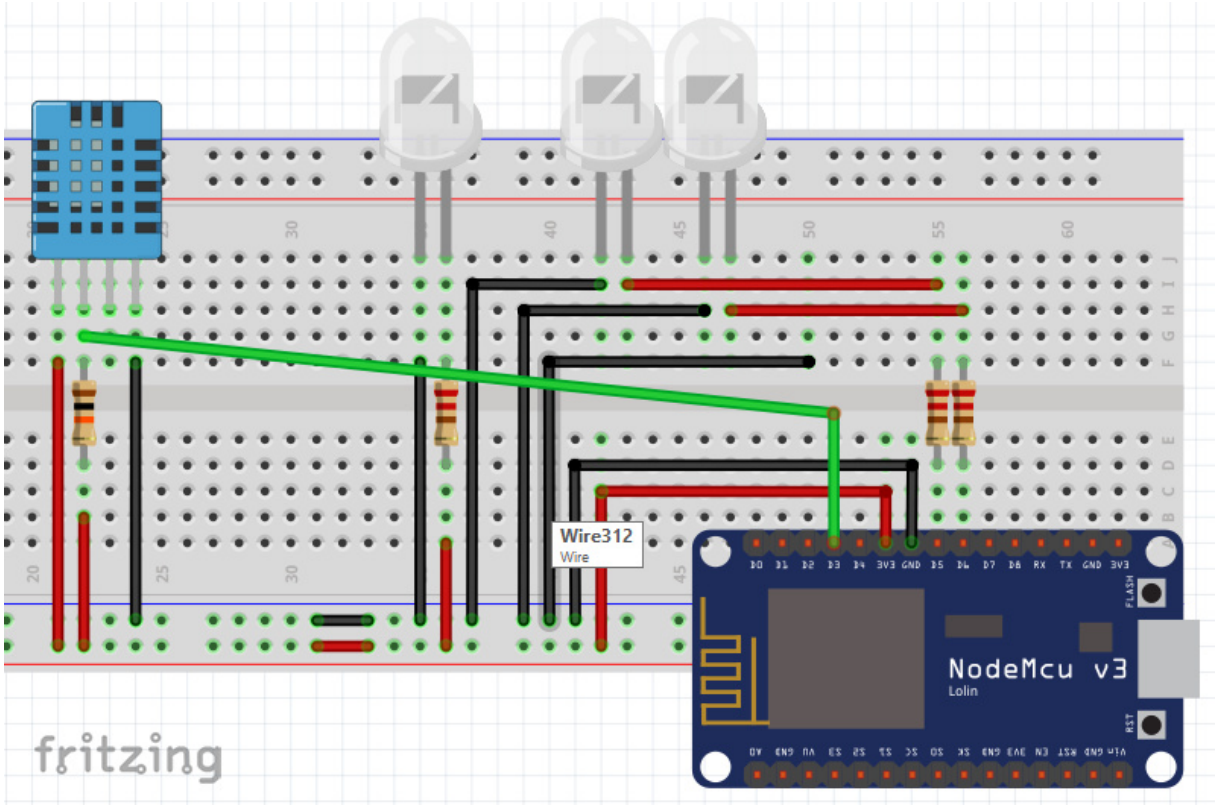
O esquemático de ligação dos componentes do projeto pode ser observado na Figura 15. A barra inferior da protoboard é energizada pelos pinos 3V3 e GND do NodeMCU.

[6] ROBOOTT. **Sensor DHT11 com Arduino**. Disponível em: https://roboott.wordpress.com/2015/08/02/sensor-dht11-com-arduino. Acesso em 16 jun. 2023.

O sensor DHT11 tem o primeiro pino (Vcc), localizado no lado esquerdo, ligado ao potencial positivo e o último (GND) conectado ao potencial negativo da protoboard. Um resistor de pull-up de 10k ohms é adicionado no segundo pino (DATA) com o intuito de estabilizar a comunicação com o pino D3 da placa, já o terceiro pino (NC) não será utilizado.

Cada LED teve seu cátodo ligado ao potencial negativo e seu ânodo conectado ao positivo. O LED verde foi utilizado para indicar que se a protoboard está energizada. O LED vermelho foi conectado ao pino D5 e acende caso a temperatura esteja maior do que o valor estabelecido. O LED azul foi conectado ao pino D6 e também foi utilizado para indicar se a umidade excedeu o valor pré-definido.

Figura 15 - Esquemático do Projeto de Comunicação Wi-Fi



Fonte: De autoria própria.

4.2.2 Modo de funcionamento da placa

O modo de operação do projeto é o AP + STA. O ponto de acesso é utilizado, inicialmente, para a configuração do dispositivo por meio de um servidor web, armazenando as informações de Wi-Fi externo, compartimento a ser monitorado e limites máximos de

temperatura e umidade. Após a configuração inicial, é possível monitorar o ambiente dentro da rede individual da placa.

Ao conectar o dispositivo em uma rede Wi-Fi com as configurações informadas no modo de ponto de acesso, o modo STA é habilitado. Assim, qualquer outro aparelho conectado à mesma rede externa é capaz de acessar as informações do servidor web.

Para que o ESP8266 pudesse atuar no modo AT + STA, foi necessário a inclusão da biblioteca ESP8266WiFi que é responsável por fornecer as funcionalidades necessárias para a conexão e configuração da rede Wi-Fi que a placa irá utilizar.

4.3 Como ocorre a comunicação?

A comunicação entre o usuário e a placa tem como intermediário um servidor *web* que pode ser criado com a utilização da biblioteca ESP8266WebServer. Ele irá responder às requisições HTTP e será o responsável por disponibilizar a interface ao qual o cliente terá acesso para definir as informações da rede e monitorar as informações de leitura do sensor.

O protocolo HTTP, do inglês *Hypertext Transfer Protocol*, é responsável pela transferência de dados entre um cliente (navegador) e um servidor. Segundo Pfister (2011, p. 30, apud DRUMOND, 2017, p. 17), ele atua em pares de requisição – resposta.

Existem vários métodos de solicitação, mas os utilizados nesse projeto são apenas o GET e o POST. Esses são os verbos mais utilizados e é muito comum encontrá-los em aplicações web. O primeiro é empregado quando o intuito é adquirir recursos do servidor, geralmente a página de um site. Já o segundo tem o propósito de enviar dados, como as informações de um formulário que são enviadas para o processamento no servidor (FONSECA, [s.d]).

Nesse projeto, o ESP8266 assume o papel de servidor. A função que será chamada quando a placa receber uma requisição é definida no método de setup. Ela será encarregada de processar os dados e retornar de uma página HTML como resposta para o cliente.

4.4 Funcionamento do Projeto

No código do projeto, é atribuído um IP estático e uma porta na qual o web servidor atende as requisições. Quando o dispositivo for utilizado pela primeira vez, após se conectar no

ponto de acesso “Teste-Monitor”, é preciso acessar o endereço número 1 da Tabela 4, com o intuito de configurá-lo. Nesse caso, o cliente recebe apenas a página web, presente na Figura 16, como resposta.

Figura 16 - Tela de configuração

Fonte: De autoria própria.

Ao enviar o formulário depois de preenchido pelo método HTTP POST, o NodeMCU recebe os dados do usuário. Caso as informações que a placa necessita já estejam definidas, então, são mostrados os endereços para a tela de monitoramento ilustrada na Figura 17.

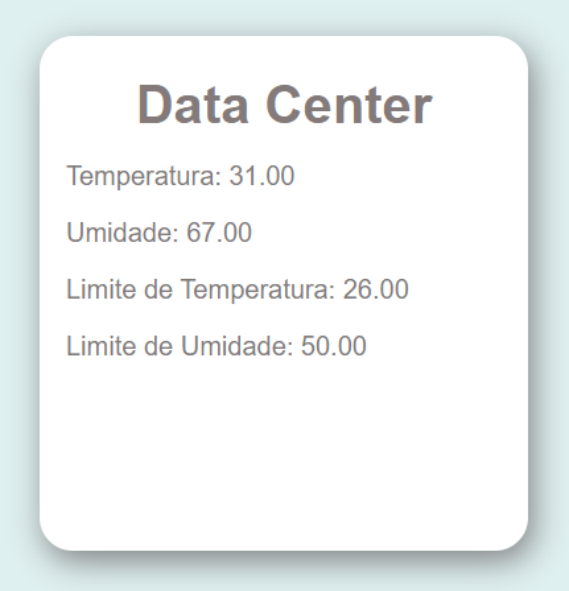
Figura 17 - Tela de endereços para monitoramento

Fonte: De autoria própria.

Ao acessar a rota indicada, uma função é acionada na placa, requisitando os valores de leitura do sensor de temperatura e umidade, os resultados são salvos no conteúdo da página web que é retornada ao cliente. Logo, nesse caso, o cliente está recebendo dados da placa.

Na tela de monitoramento, presente na Figura 18, estão os dados referentes aos valores lidos pelo sensor e aos limites estabelecidos pelo usuário anteriormente. Os LEDs vermelho e azul apenas ficam acesos quando o resultado da temperatura e da umidade, respectivamente, ultrapassam os limites configurados.

Figura 18 - Tela de monitoramento



Fonte: De autoria própria.

Tabela 4 - Links e respostas esperadas após o acesso

N°	Endereço	Resposta
1	http://192.168.4.1/	Retorna a página de configuração ou os links para o monitoramento remoto.
2	http://192.168.4.1/config	Recebe os dados do formulário via HTTP POST e envia os links para o monitoramento remoto.
3	http://192.168.4.1/reconfig	Retorna a página de configuração.
4	http://192.168.4.1/monitor	Retorna a página de monitoramento.

Fonte: De autoria própria.

Após conectar a placa a uma rede externa, é possível acessar todas as rotas da Tabela 4 utilizando o mesmo padrão indicado pelo *link* na tela de monitoramento. Assim, as funcionalidades do projeto ficam disponíveis para serem acessadas em qualquer ponto da rede. Isso é vantajoso porque quando algum aparelho está conectado ao ponto de acesso do ESP8266, não há conexão com a internet, impedindo o usuário de consumir outras aplicações.

4.5 Resultados Experimentais

Após a execução do projeto, constatou-se que a comunicação entre os nós das redes estabelecidas ocorreu conforme as descrições apresentadas nos tópicos iniciais desse capítulo. Quando o ESP8266 está atuando como ponto de acesso, todos os dispositivos conectados a ele podem acessar as suas funcionalidades e, assim, trocar dados. Já no modo estação, todos os aparelhos que possuem conexão com a mesma rede externa são capazes de se comunicar com o NodeMCU através dela.

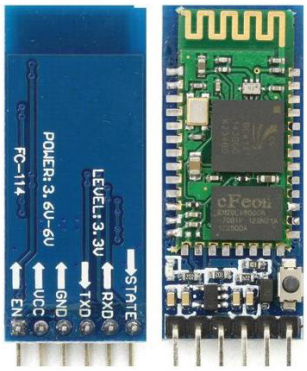
Além disso, após a configuração dos limites máximos de temperatura e umidade, os LEDs passaram a responder de acordo com o valor das leituras. Indicando que as configurações foram salvas e interpretadas com sucesso. Porém, foi observado que se a placa fosse desconectada na fonte de energia, as configurações eram perdidas, indicando a necessidade de implementar funções de armazenamento de dados.

Para melhoria do projeto, seria importante considerar a utilização de recursos externos, como armazenamento de informações em bancos de dados. Dessa forma, o ESP8266 atuaria como cliente ao enviar dados por meio do método POST. Além disso, seria interessante o uso de recursos que permitem a notificação via e-mail ou SMS em casos de valores de temperatura e umidade muito altos.

5 COMUNICAÇÃO VIA BLUETOOTH

O ESP8266 não possui Bluetooth embutido. Para que a troca de informações aconteça por meio desse protocolo, é necessário utilizar um módulo adicional com essa funcionalidade. Neste trabalho, o componente escolhido foi o HC – 05 que pode ser observado na Figura 19.

Figura 19 - Frente e verso do módulo HC - 05



Fonte: ARDUINO E ELETRÔNICA, Módulo Bt Bluetooth Master – Slave Hc – 05 (Ref. C. 003).[7]

Segundo sua documentação, o módulo possui uma tensão de alimentação de 3,6 a 6V, enquanto a tensão de funcionamento é corresponde a 3,3V. Os pinos Tx e Rx desse componente devem ser conectados aos pinos Rx e Tx da placa, respectivamente. O terminal EN é utilizado para colocar o componente no modo AT e permitir a configuração do bluetooth.

O módulo pode atuar como mestre ou como escravo. Ambos operam em modo *full-duplex*, ou seja, são capazes de enviar e receber dados dentro da rede, mas apenas o primeiro será responsável por solicitar a conexão, regular a transmissão de dados e o sincronismo entre eles (ALECRIM, 2021).

A nível de código, para o ESP8266, pode-se inferir que a transferência de informações por meio do bluetooth assemelha-se a uma comunicação serial padrão. Os dados são enviados e recebidos com o auxílio das funções de escrita e leitura da porta serial ao qual o módulo está conectado.

[7] ARDUINO E ELETRÔNICA**. Módulo Bt Bluetooth Master – Slave Hc – 05 (Ref. C. 003).** Disponível em: https://arduinoeeletronica.com.br/produto/modulo-bt-bluetooth-master-slave-hc-05/. Acesso em: 22 de jun. 2023.

5.1 Exemplo 2: Recebendo dados de monitoramento de ambiente via Bluetooth

Para melhor exemplificar a troca de informações entre o NodeMCU e outros dispositivos por meio da comunicação bluetooth, foi realizado um projeto de monitoramento de temperatura um pouco mais simples que o anterior. Nele, a placa irá receber os dados de leitura de sensor DHT11 presente em um Arduino UNO e enviar comandos para o mesmo com o auxílio de um módulo HC-05.

Para o projeto foram utilizados os seguintes componentes presentes na Tabela 5. A plataforma utilizada, assim como no exemplo anterior, foi a IDE do Arduino com a linguagem baseada em C/C++. Os códigos para esse experimento se encontram nos APÊNDICES B e C.

Tabela 5 - Materiais Utilizados para o Projeto de Comunicação Wi-Fi

Quantidade	Material
1	Placa NodeMCU V3
2	*Protoboard* de 830 pontos
1	Placa Arduino UNO
2	Módulo HC-05
1	LED Vermelho
1	Resistor de 220 ohms para o LED
1	Resistor de 1k ohms para o divisor de tensão
1	Resistor de 2k ohms para o divisor de tensão
1	Resistor de 10k ohms para o sensor
16	Jumpers macho-macho

Fonte: De autoria própria.

5.2 Esquemático de ligações

Como mencionado anteriormente, os pinos Rx e Tx do módulo HC-05 devem estar conectados aos pinos Tx e Rx de cada dispositivo, respectivamente. Contudo, não foram utilizados os pinos de transmissão e recepção das placas, tendo em vista que eles são utilizados para a comunicação serial da porta USB que é ocupada durante a gravação do código. Sendo assim, foi utilizada a biblioteca SoftwareSerial.

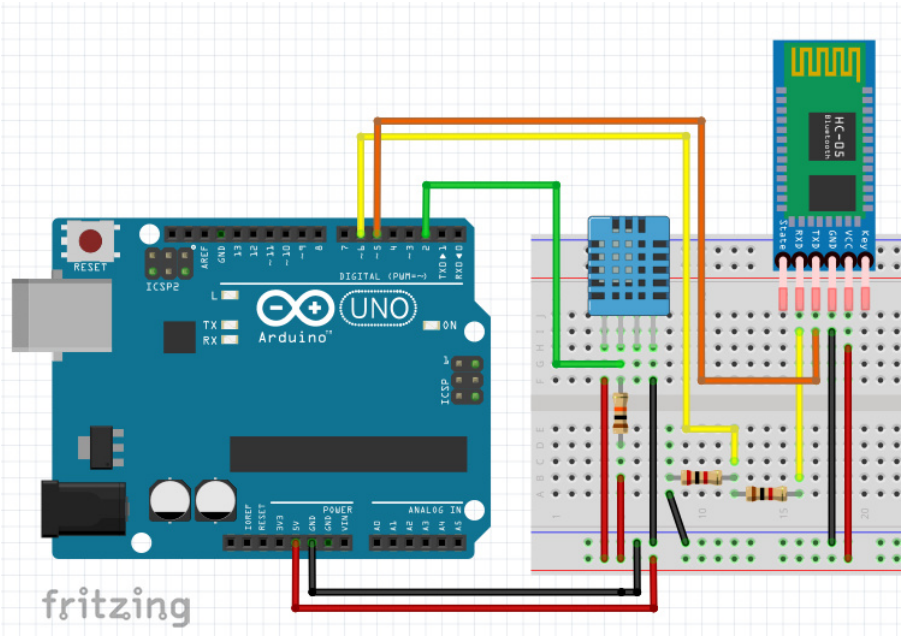
Essa biblioteca, segundo sua documentação, permite a comunicação serial em pares de pinos digitais. Sendo assim, é uma alternativa interessante para quando os terminais Rx e Tx não estão disponíveis. Contudo, é importante destacar que a SoftwareSerial possui algumas limitações. Não é possível transmitir e receber informações ao mesmo tempo e, caso haja mais de uma porta serial criada por ela, apenas uma pode receber dados por vez.

Além disso, quando a placa utilizada é o Arduino, a velocidade máxima de comunicação é 57600 bps para o receptor. Dependendo do modelo, também é necessário ter atenção quanto aos pinos utilizados. Segundo a documentação, no caso do modelo UNO, o terminal 13 não deve ser escolhido.

Diante disso, os pinos 5 e 6 do Arduino foram escolhidos para a função de receptor e transmissor, respectivamente, enquanto o pino 2 recebeu a função de leitura do sensor de umidade e temperatura. A conexão do DHT11 é similar àquela apresentada na seção 4.2.1 desse trabalho.

Tendo em vista que a tensão de saída dos pinos digitais do arduino são de 5V, foi necessário adicionar um divisor de tensão no seu terminal 6 (Tx) para garantir a tensão de 3,3V no receptor do HC-05 e evitar possíveis danos ao módulo bluetooth. O esquemático das ligações realizadas está ilustrado na Figura 20.

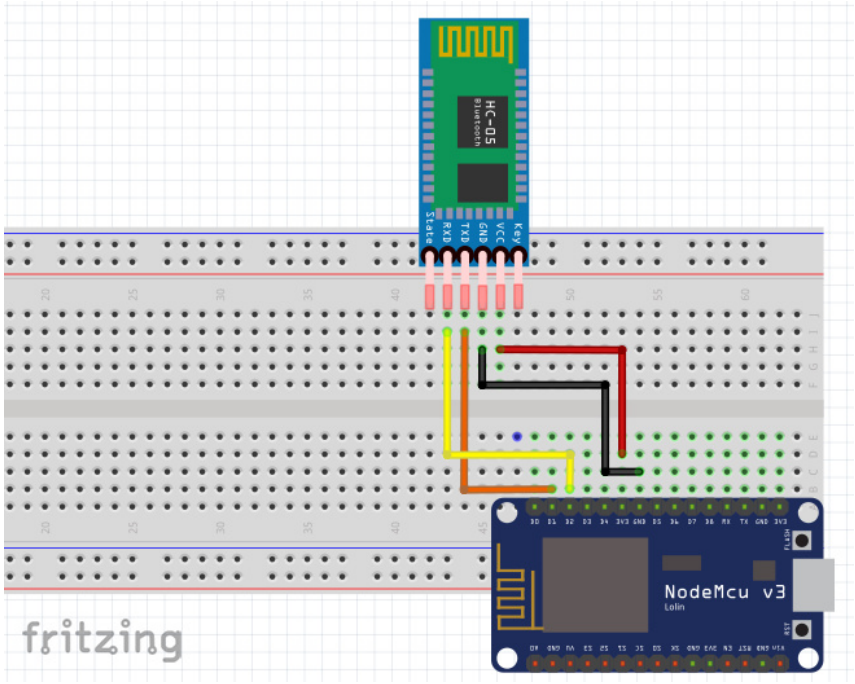
Figura 20 - Esquemático de ligações do Arduino UNO



Fonte: De autoria própria.

Para o NodeMCU, como a tensão de operação é 3,3V não há necessidade de divisores de tensão. Sendo assim, a ligação é feita de maneira simples, com o VCC, GND, Tx, Rx do módulo diretamente conectados aos terminais 3V3, GND, D1e D2 da placa, nessa respectiva ordem. Esse esquemático de ligação pode ser observado na Figura 21.

Figura 21 - Esquemático de ligações do NodeMCU



Fonte: De autoria própria.

5.3 Configuração do módulo bluetooth

O Arduino UNO foi escolhido para simular a troca de dados entre o NodeMCU e outros dispositivos pela sua facilidade de uso, mas qualquer outro aparelho poderia ser empregado no seu lugar. Para o código, não é relevante quem está na outra ponta da comunicação, a programação do chip ESP8266 seria a mesma.

Contudo, o papel do bluetooth dos outros dispositivos importaria muito, tendo em vista que apenas o mestre é capaz de iniciar a comunicação e gerenciar o fluxo de informações. Para esse projeto, o NodeMCU foi definido como mestre enquanto o Arduino UNO ficou como escravo.

Para configurar cada módulo foram realizados os passos presentes nas Tabelas 6 e 7. A função de cada comando AT efetuado nesta etapa pode ser encontrada na Tabela 8. É importante destacar a execução do terceiro código, pois é a partir dele que se torna possível verificar que o papel do bluetooth foi definido com sucesso.

Tabela 6 - Comandos para configuração do modo escravo

Passo	Comando AT	Resposta
1°	AT	OK
2°	AT+ROLE=0	OK
3°	AT+ROLE?	:ROLE=0
3°	AT+ADDR?	+ADDR: 1234:56:abcdef OK

Fonte: De autoria própria.

Tabela 7 - Comandos para configuração do modo mestre

Passo	Comando AT	Resposta
1°	AT	OK
2°	AT+ROLE=1	OK
3°	AT+ROLE?	+ROLE:1
3°	AT+CMODE=0	OK
4°	AT+BIND= 1234,56,abcdef	OK

Fonte: De autoria própria.

Tabela 8 - Descrição dos comandos AT

Comando AT	Descrição	Valores
AT	Testa se os comandos estão sendo respondidos	-
AT+ROLE	Define o papel do módulo	0 – escravo 1 - mestre
AT+ADDR	Indica o endereço do dispositivo	-
AT+CMODE	Define o modo de conexão	0 – endereços fixos 1 – qualquer endereço 2 – *slave-loop*
AT+BIND	Conecta o módulo a um endereço específico	-

Fonte: De autoria própria.

Após a configuração dos módulos, o dispositivo mestre conectou-se normalmente ao dispositivo escravo a partir do endereço especificado.

5.4 Funcionamento do projeto

Para cada placa, foi criada uma nova porta serial com o auxílio da biblioteca SoftwareSerial e é a partir delas que a comunicação ocorre. Como mencionado anteriormente, não é possível enviar e receber dados ao mesmo tempo, então, foi definido que, cada cinco segundos, o NodeMCU iria requisitar a leitura do sensor ao Arduino que, ao receber a solicitação retornaria a resposta.

Para isso, foram utilizadas as funções de leitura e escrita na porta serial criada em cada placa. É uma lógica bem simples que permite que ambos os dispositivos enviem e recebam dados, um de cada vez. O comportamento pode ser melhor compreendido ao analisar a Tabela 9.

Tabela 9 - Links e respostas esperadas após o acesso (continua)

Comando	Responsabilidade
espSerial.read()	Lê os dados recebidos pelo ESP8266

Tabela 9 - Links e respostas esperadas após o acesso (conclusão)

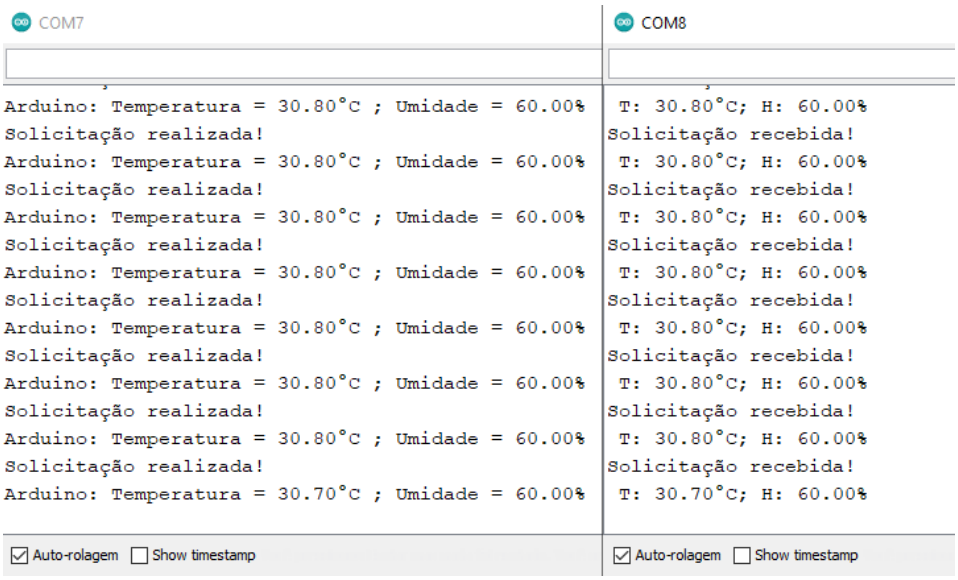
Comando	Responsabilidade
espSerial.println()	Envia os dados para o Arduino UNO
unoSerial.read()	Lê os dados recebidos pelo Arduino UNO
unoSerial.println()	Envia os dados para o ESP8266

Fonte: De autoria própria.

5.5 Resultado Experimentais

Ao executar o projeto, foi comprovado que a comunicação ocorre como o esperado e pode ser verificada por meio da interface de monitoramento das portas seriais da IDE do Arduino. Como pode ser observado na Figura 22, o NodeMCU estava conectado à porta COM 7 e, sempre que ele realizava uma requisição, ela era recebida pelo Arduino que, por sua vez, retornava os dados de temperatura e umidade.

Figura 22 - Resultados do projeto de comunicação bluetooth



Fonte: De autoria própria.

6 CONCLUSÃO

Diante do estudo realizado nesse trabalho, foi possível compreender os diferentes cenários de comunicação em soluções IoT e obter conhecimento acerca das topologias mais comuns das redes Wi-Fi e Bluetooth. Isso permitiu uma melhor visualização da troca de informações entre os elementos integrantes de uma rede.

A realização dos experimentos proporcionou uma análise prática de como a comunicação remota pode ser estabelecida entre dispositivos e o ESP8266 com o auxílio de cada tecnologia. Além disso, foi possível perceber como as placas de prototipagem facilitam e aceleram o desenvolvimento do firmware de uma solução.

Com base no que foi apresentado, pode-se entender que o ESP8266 é um componente de baixo custo que oferece uma grande versatilidade para projetos que necessitam de conectividade sem fio. O mesmo também pode atuar como uma ponte para os elementos que não possuem a capacidade se conectar à internet.

Sendo assim, o ESP8266 pode ser considerado um bom chip para o desenvolvimento de soluções que possuem o propósito de automatizar tarefas, pois além de demonstrar um potencial significativo para torná-las mais inteligentes, eficientes e acessíveis, possibilita o controle e monitoramento remoto dos dispositivos.

Para trabalhos futuros, recomenda-se a realização de experimentos com o intuito de investigar a possibilidade de interferência de sinal em aplicações com o ESP8266 que utilizam comunicação Wi-Fi e Bluetooth simultaneamente. Outra sugestão seria o estudo da implementação do protocolo ESP-NOW como alternativa para a comunicação sem fio entre dispositivos ESP.

REFERÊNCIAS

ALECRIM, Emerson. **O que é Wi-Fi? (conceitos e versões)**. Infowester, 2008. Disponível em: https://www.infowester.com/wifi.php. Acesso em: 10 jun. 2023.
ALECRIM, Emerson. **Diferença entre roteador, switch, modem e hub**. Infowester, 2019. Disponível em: https://www.infowester.com/hubswitchrouter.php. Acesso em: 11 jun. 2023.

ALECRIM, Emerson.**Bluetooth: o que é, como funciona, versões**. Infowester, 2021. Disponível em: https://www.infowester.com/hubswitchrouter.php. Acesso em: 27 jun. 2023.

ARDUINO. **SofwareSerial Library**. [S.l], 2023. Disponível em: https://docs.arduino.cc/learn/built-in-libraries/software-serial. Acesso em: 26 jun. 2023.

ARDUINO E ELETRÔNICA**. Módulo Bt Bluetooth Master – Slave Hc – 05 (Ref. C. 003).** Disponível em: https://arduinoeeletronica.com.br/produto/modulo-bt-bluetooth-master-slave-hc-05/. Acesso em: 22 de jun. 2023.

ARAR, Steve. **Star vs. Mesh Networking Topology: IoT Wireless Conectivity Fundamentals**. All about circuits, 2022. Disponível em: https://www.allaboutcircuits.com/technical-articles/star-vs-mesh-networking-topology-internet-of-things-wireless-connectivity-fundamentals/. Acesso em: 11 jun. 2023.

ARAÚJO JÚNIOR, A. P.; CHAGAS, C. V.; FERNANDES, R. G. **Uma rápida análise sobre automação industrial**. Departamento de Engenharia de Computação e Automação: UFRN, 2003. Disponível em: https://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_6.pdf. Acesso em: 23 abr. 2022.

DRUMOND, R. B. P. **Sistema de Monitoramento em Tempo Real de Consumo de Água via Web**. 2017. 67 p. TCC (Graduação em Engenharia Elétrica) – Universidade Federal do Ceará. Fortaleza, 2017.

EMBARCADOS. **Arduino UNO**. Disponível em: https://embarcados.com.br/arduino-uno/. Acesso em: 02 nov. 2022.

ESPRESSIF. **ESP8266EX Datasheet**. Versão 7.0 [S.l], 2023. Disponível em: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. Acesso: 07 abr. 2023.

ETECHNOPHILES. **HC-05 pinout, specifications, datasheet and HC05 Arduino connection**. Disponível em: https://www.etechnophiles.com/hc-05-pinout-specifications-datasheet/. Acesso 27 jun. 2023.

FERNÁNDEZ, Marcial Porto. **Rede de Computadores**. 2ª ed. Fortaleza, CE, Brasil: EdUECE, 2015.

FERRONI, E. H. et al. A plataforma arduino e suas aplicações. **Revista da UI_IPSatarém**, Santarém. v. 3, n. 2, p. 133-148, set de 2022. Disponível em: https://revistas.rcaap.pt/uiips/article/view/14354. Acesso em 03 nov. 2022.

FONSECA, E. **O que é HTTP, Request, GET, POST, Response, 200, 404?**. Treinaweb, [s.d]. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-http-request-get-post-response-200-404. Acesso em: 25 jun. 2023.

LIMA, F. S. **A automação e sua evolução**. Departamento de Engenharia de Computação e Automação: UFRN, 2003. Disponível em: https://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_16.pdf. Acesso em: 23 abr. 2022.

LOBO DA ROBÓTICA. **Arduino Uno Pinout – Desvendando O Arduino.** Disponível em: https://lobodarobotica.com/blog/arduino-uno-pinout/. Acesso em 03 nov. 2022.

LOUIS, L. Working Principle of Arduino and Using It As a Tool for Study and Research. **International Journal of Control, Automation, Communication and Systems**, [s.l], v. 1, n. 1, p. 21-29, mar/abr de 2016. Disponível em: https://airccse.com/ijcacs/papers/1216ijcacs03.pdf. Acesso em: 02 nov. 2022.

LUZ, G. B.; KUIAWINSKI, D. L. **Mecanização, Autonomação e Automação – Uma Revisão Conceitual e Crítica**. In: XIII SIMPEP, 2006, Bauru. Anais Eletrônicos. Bauru: UNESP, 2006. Disponível em: https://simpep.feb.unesp.br/anais/anais_13/artigos/1210.pdf. Acesso em: 23 abr. 2022.

MAROCCO, C. A. D. **Proposta de Topologia de Rede de Dados com Segurança e Foco na Produtividade Utilizando Ferramentas de Software Livre**. 2015. 65 p. Monografia (Pós-Graduação em Redes de Computadores com Ênfase em Segurança) – Centro Universitário de Brasília, Brasília, 2015. Disponível em: https://repositorio.uniceub.br/jspui/bitstream/235/8157/1/51307936.pdf. Acesso em: 22 mai. 2023.

MAKERHERO. **Lançamento! Raspberry Pi 4 Model B**. Disponível em: https://www.makerhero.com/blog/lancamento-raspberry-pi-4-model-b/. Acesso em 05 nov. 2022.

MASTER WALKER ELETRONIC SHOP. **NodeMCU – Uma plataforma com características singulares para o seu projeto IoT.** Disponível em: https://blogmasterwalkershop.com.br/embarcados/nodemcu/nodemcu-uma-plataforma-com-caracteristicas-singulares-para-o-seu-projeto-iot. Acesso em: 15 abr. 2023.

NAYYAR, A.; PURI, V. Raspberry Pi – A small, Powerful, Cost Effective and Efficient Form Factor Computer: A review. **International Journal of Advanced Research in Computer Science and Software Engineering**, [s.l], v. 5, n. 12, p. 720 – 737, nov/dez de 2016. Disponível em: https://www.researchgate.net/publication/305668622_Raspberry_Pi-A_Small_Powerful_Cost_Effective_and_Efficient_Form_Factor_Computer_A_Review/. Acesso em: 05 nov de 2022.

NEVES, M. A. T. **Internet das Coisas (IoT): Introdução e Visão Geral de Aplicações**. Orientador: José Simão de Paula Pinto. 2021. 53f. TCC (Graduação) – Gestão da Informação, Departamento de Ciência e Gestão da Informação, Universidade Federal do Paraná, Curitiba,

2021. Disponível em: https://acervodigital.ufpr.br/bitstream/handle/1884/70452/MATEUS-APARECIDO-TONIN-NEVES.pdf?sequence=1&isAllowed=y. Acesso em: 17 set. 2022.

NODEMCU. **NodeMCU Documentation**. [S.l], 2015. Disponível em: https://nodemcu.readthedocs.io/en/release/modules/wifi/. Acesso em> 20 jun. 2023.

OLIVEIRA NETO, B. B.; QUEIROGA, S. L. M.; MONTEIRO, P. F. **Aplicabilidade dos Microcontroladores em Inovações Tecnológicas**. In: VII CONNEPI, 2012, Palmas. Anais Eletrônicos. Palmas: IFTO, 2012. Disponível em: https://propi.ifto.edu.br/ocs/index.php/connepi/vii/paper/viewFile/2433/2526. Acesso em: 26 out. 2022.

RANDOM NERD TUTORIALS. **25 Useful Arduino Shields That You Might Need to Get**. Disponível em: https://randomnerdtutorials.com/25-arduino-shields/. Acesso em: 8 nov. 2022.

RASPBERRY. **RASPBERRY Pi 4 Computer Model B**. [S.l], 2019. Disponível em: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf. Acesso em 01 abr. 2023.

ROBINSON, S. **Speeding Up Arduino. Stack Abuse**, 2016. Disponível em: https://stackabuse.com/speeding-up-arduino/. Acesso em: 04 nov. 2022.

ROBOOTT. **Sensor DHT11 com Arduino**. Disponível em: https://roboott.wordpress.com/2015/08/02/sensor-dht11-com-arduino. Acesso em 16 jun. 2023.

SOUZA, David José. **Desbravando o PIC: Ampliado e Atualizado para PIC 16F628A**. 8ª ed. São Paulo, SP, Brasil: Érica, 2005.

SWATHI, K.; SANDEEP, T. U.; RAMANI, A. R. Perfomance Analysis of Microcontrollers Used In IoT Technology. **International Journal of Scientific Research in Science, Engineering, and Technology**, [s.l], v. 4, n. 4, p. 1268-1273, mar/abr de 2018. Disponível em: https://ijsrset.com/IJSRSET1844270. Acesso em: 30 out. 2022.

VERMA, M.; SINGH, S.; KAUR, B. An overview of Bluetooth Technology and its Communication Applications. **International Journal of Current Engineering and Technology**, [s.l], v. 5, n. 3, p. 1588-1592, mai/jun de 2015. Disponível em: http://inpressco.com/an-overview-of-bluetooth-technology-and-its-communication-applications/. Acesso em: 11 jun. 2023.

YUAN, M. **Getting to know NodeMCU and its devkit board**. IBM Developer, 2017. Disponível em: https://developer.ibm.com/tutorials/iot-nodemcu-open-why-use/.Acesso em: 07 abr. 2023.

APÊNDICE A – FIRMWARE DO EXEMPLO 1

```
#include<ESP8266WiFi.h> //fornece as funcionalidades Wi-Fi

#include<ESP8266WebServer.h> //fornece as funcionalidades do web servidor

#include <DHT.h> //permite a interação com sensores de temperatura e umidade da série DHT


#define DHTTYPE DHT11 //define que o sensor utilizado será o DHT11

DHT dht(D3, DHTTYPE); //define a porta conectada ao pino DATA do sensor DHT11


String ssidAP = "Teste-Monitor"; //define o SSID do ponto de acesso do ESP

const char* passwordAP = "12345678"; //define a senha do ponto de acesso do ESP


ESP8266WebServer server(80); //definindo um servidor na porta 80


//declaração de variáveis globais

float temperature;

float humidity;

String compartment = "";

float maxTemperature;

float maxHumidity;

String currentSSID = "";

String wifiStatus;


//o PROGMEN é utilizado para armazenar o HTML e javascript que serão retornados como resposta
às solicitações HTTP

const char html_page[] PROGMEM = R"=====(
```

```
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Monitor</title>
<style>
   body{
      /*background-color: #f1eded;*/
      background-color: #dff0f1;
      font-family: 'Open Sans', sans-serif;
      font-size: 20px;
      color: #857a7a;
   }
   h1{
      margin: 10px;
      text-align: center;
   }
   input {
      margin: 5px;
      height: 26px;
      padding: 3px;
      border: 1px solid #c3bbbb;
      border-radius: 4px;
   }
   form {
```

```
    text-align: center;
}
.container {
    height: 350px;
    width: 330px;
    background-color: white;
    border-radius: 25px;
    padding: 20px;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    box-shadow: 4px 8px 25px grey;
}
.button {
    display: block;
    min-width: 170px;
    height: 40px;
    background-color: #1abc9c;
    border: none;
    color: white;
    margin: 10px auto;
    padding: 10px;
    cursor: pointer;
    border-radius: 4px;
```

```
        }
        .invalid{
            border-color: red !important;
        }
    </style>
</head>
<body>
    <div class="container"></div>
    <script>
        function ValidateFields(){
            let form = document.getElementById('configForm');
            const fields = form.querySelectorAll('input');
            let invalidFields = 0;
            fields.forEach(function(field) {
                if (field.value.trim() == '') {
                    invalidFields++;
                    field.classList.add('invalid');
                } else {
                    field.classList.remove('invalid');
                }
            });
            if (invalidFields > 0) {
                alert('Todos os campos devem ser preenchidos!');
                return false;
            }
```

```
      return true;
    }
    let href = window.location.href;
    function reloadWindow(){
      if(href.includes('/monitor')){
        window.location.reload();
      }
    }
    setInterval(reloadWindow, 5000);
    let reconfig = document.getElementById("reconfig");
    if(reconfig != null){
     reconfig.addEventListener("click", function() {
      window.location.href = "/reconfig";
     });
    }
  </script>
</body>
</html>
)=====";

const char html_config_form[] PROGMEM = R"=====(
 <div class="container">
  <h1>Configuração</h1>
  <form name="configForm" id="configForm" action="/config" method="post" onsubmit="return
ValidateFields();">
```

```
    <input type="text" name="monitored-compartment" id="monitored-compartment"
placeholder="Compartimento" >
    <input type="text" name="max-temperature" id="max-temperature" placeholder="Limite de
Temperatura(°C)">
    <input type="text" name="max-humidity" id="max-humidity" placeholder="Limite de
Umidade(%)">
    <input type="text" name="ssid" id="ssid" placeholder="SSID do Wi-Fi">
    <input type="password" name="password" id="password" placeholder="Senha do Wi-Fi">
    <button type="submit" class="button">Configurar</button>
   </form>
 </div>
)=====";

void setup() {
 pinMode(D5, OUTPUT); //define o pino D5 como saída
 pinMode(D6, OUTPUT); //define o pino D6 como saída
 IPAddress staticIP(192, 168, 4, 2); //define um IP estático para acesso ao web servidor
 WiFi.softAP(ssidAP.c_str(),passwordAP); //ccnfigura o ESP para operar em modo AP

 Serial.print("Ponto de Acesso: ");
 Serial.print(ssidAP);
 Serial.print("IP:\t");
 Serial.println(WiFi.softAPIP()); //imprime o IP de acesso ao web servidor

 Serial.begin(115200); //define o baud rate
 delay(50);
```

```
  dht.begin(); //inicializa o sensor DHT


  server.on("/", handle_OnConnect); //chama a função handle_OnConnect caso uma requisição seja feita para a rota indicada (raiz)

  server.onNotFound(handle_NotFound);

  server.on("/config", handle_config); //chama a função handle_config caso uma requisição seja feita para a rota indicada (config)

  server.on("/reconfig", handle_reconfig); //chama a função handle_reconfig caso uma requisição seja feita para a rota indicada (reconfig)

  server.on("/monitor", handle_OnMonitor); //chama a função handle_OnMonitor caso uma requisição seja feita para a rota indicada (monitor)

  server.begin(); //Inicializa o servidor

  Serial.println("Servidor HTTP inicializado");



}


void loop() {

  server.handleClient(); //lida com as solicitações realizadas pelos clientes

  temperature = dht.readTemperature(); //lê a temperatura e salva o valor em uma variável

  humidity = dht.readHumidity(); //lê a umidade e salva o valor em uma variável

  verifyLimits();

}


String formString(){

  Serial.println("Nova configuração!");
```

```
  String page = FPSTR(html_page);
  String form = FPSTR(html_config_form);
  page.replace("<div class=\"container\"></div>", form); //substitui a div container por div com o
html do formulário
  return page;
}

String configString(String ssid, String wifiIP, String staticIP, String  connectionStatus){
  String hrefWiFi = wifiIP + "/monitor";
  String hrefESP = staticIP + "/monitor";
  String page = FPSTR(html_page);
  String resultString = "<div class=\"container\">\n";
  resultString += "<p>Status: " + connectionStatus + "</P>\n";
  if(connectionStatus == "conectado"){
    resultString += "<p>SSID: " + ssid + "</P>\n";
    resultString += "<p>Monitoramento pela rede Wi-Fi:<br><a href=\"\">http://"+ wifiIP +
"/monitor</a></P>\n";
    resultString.replace("href=\"\"", "href=\"http://" + hrefWiFi + "\"");
  } else {
    resultString += "<p>Reconfigure a rede Wi-Fi externa</P>\n";
  }
  resultString += "<p>Monitoramento rede do ESP:<br><a href=\"\">" + staticIP +
"/monitor</a></P>\n";
  resultString.replace("href=\"\"", "href=\"http://" + hrefESP + "\"");
  resultString += "<button type=\"submit\" class=\"button\" id=\"reconfig\">Reconfigurar</button>";
  resultString += "</div>";
```

```
  page.replace("<div class=\"container\"></div>", resultString); //substitui a div container por div com o html dos links de monitoramento
  return page;
}

String monitorString(float temperaturaLimite, float umidadeLimite, float temperaturaAtual, float umidadeAtual){
    String resultString = "<div class=\"container\">\n";
    resultString += "<h1>" + compartment + "</h1>\n";
    resultString += "<p id=\"temperature\">Temperatura(°C): " + String(temperaturaAtual) + "</p>\n";
    resultString += "<p id=\"humidity\">Umidade(%): " + String(umidadeAtual) + "</p>\n";
    resultString += "<p id=\"max-temperature\">Limite de Temperatura(°C): " + String(temperaturaLimite) + "</p>\n";
    resultString += "<p id=\"max-humidity\">Limite de Umidade(%): " + String(umidadeLimite) + "</p>\n";
    resultString += "</div>";
    return resultString;
}

void printConfig(){ //exibe as informações de configuração
  Serial.println(compartment);
  Serial.println(maxTemperature);
  Serial.println(maxHumidity);
  Serial.println(currentSSID);
  Serial.println(wifiStatus);
}
```

```
void handle_OnConnect() { //
  String page = "";
  IPAddress wifiIP = WiFi.localIP();
  String wifiIPString = wifiIP.toString();
  IPAddress softIP = WiFi.softAPIP();
  String softIPString = softIP.toString();
  if(compartment == "" && maxTemperature == 0 && maxHumidity ==0){ //caso as variáveis de configuração estejam vazias, exibe a página referente ao formulário
   page = formString();
   printConfig();
  } else { //caso contrário, mostra a tela com os links de monitoramento
   page = configString(currentSSID, wifiIPString, softIPString, wifiStatus);
   printConfig();
 }
 server.send(200, "text/html", page); //envia o código 200 e uma página html como resposta
}

void handle_NotFound() { //Função para lidar com o erro 404
 server.send(404, "text/plain", "Não encontrado");
}

void handle_reconfig(){ //chama a tela de configuração
 Serial.println("Reconfiguração!");
 String page = formString();
```

```
  server.send(200, "text/html", page); //envia o código 200 e uma página html como resposta
}

void handle_config(){ //lida com as informações enviadas pelo formulário de configuração
  Serial.println("Nova configuração!");
  if (WiFi.status() == WL_CONNECTED) { //caso esteja conectado, desconecta para evitar erros
    WiFi.disconnect();
    Serial.println("Desconectado!");
  }
  String ssidWifi = server.arg("ssid"); //pega o input com o name = ssid
  currentSSID = ssidWifi;
  String passwordWifi = server.arg("password"); //pega o input com o name = password
  maxTemperature = server.arg("max-temperature").toFloat(); //pega o input com o name = max-temperature
  maxHumidity = server.arg("max-humidity").toFloat(); //pega o input com o name = max-humidity
  compartment = server.arg("monitored-compartment"); //pega o input com o name = max-compartment
  if(!ssidWifi.equals("") && !passwordWifi.equals("")) { //caso a senha e o ssid sejam diferentes de vazio, aplica as novas configurações da rede wifi
     WiFi.begin(ssidWifi, passwordWifi);  //inicia a conexão Wi-Fi, possibilitando o uso do modo STA
     Serial.println("");
     int count = 0;
     while ( count < 18 ) {
          Serial.print(".");
          delay(500);
          if (WiFi.status() == WL_CONNECTED) {
```

```
            Serial.println("Conectado ao WiFi");
            Serial.print("IP: ");
            Serial.println(WiFi.localIP());
            wifiStatus = "conectado";
            handle_OnConnect();
            return;
           }
       count++;
     }
     Serial.println();
     wifiStatus = "desconectado";
     Serial.println("Timed out.");
     handle_OnConnect();
  }
}

void handle_OnMonitor() {
  Serial.print("Temperatura: ");
  Serial.print(temperature);
  Serial.println(" ºC");
  Serial.print("Umidade: ");
  Serial.print(humidity);
  Serial.println(" %");
  String page = FPSTR(html_page);
```

```
  String containerString = monitorString(maxTemperature, maxHumidity, temperature, humidity);
//pega a div com o html das informações de monitoramento
  page.replace("<div class=\"container\"></div>", containerString); //substitui a div container com a
div que possui o html de monitoramento
  server.send(200, "text/html", page); //envia o código 200 e uma página html como resposta
}

void verifyLimits(){ //verifica se os dados de leitura do sensor ultrapassaram os limites configurados
  if(maxTemperature != 0 && maxHumidity != 0){
    if(temperature > maxTemperature){
      digitalWrite(D5, HIGH);
    } else {
      digitalWrite(D5, LOW);
    }

    if(humidity > maxHumidity){
      digitalWrite(D6, HIGH);
    } else {
      digitalWrite(D6, LOW);
    }
  }
}
```

APÊNDICE B – FIRMWARE DO EXEMPLO 2 (DISPOSITIVO MESTRE)

```
#include <SoftwareSerial.h>

SoftwareSerial espSerial(D1, D2); //define uma nova porta serial com os pinos D1 como Rx e D2 como Tx

String data;
unsigned long previousTime = 0;  //salva o tempo da execução anterior
const unsigned long interval = 5000;  // define intervalo de 5 segundos

void setup(){
  Serial.begin(115200); //define o baud rate da porta serial
  espSerial.begin(38400); //define o baud rate da nova porta serial
  delay(2000);
}
void loop()
{
  unsigned long currentTime = millis(); //pega o tempo atual
  if(currentTime - previousTime >= interval){ //verifica se existe uma diferença de 5 segundos ou mais entre o tempo atual e a última requisição
    Serial.println("Solicitação realizada!");
    espSerial.println("1"); //envia o caractere 1 pela nova porta serial, indicando uma nova requisição
    previousTime = currentTime; //salva o tempo em que a equisição foi realizada
  }
```

```
  if (espSerial.available()) { //verifica se há dados disponíveis para leitura na porta espSerial
    Serial.write(espSerial.read()); //escreve na serial padrão o valor recebido
  }
}
```

APÊNDICE C – FIRMWARE DO EXEMPLO 2 (DISPOSITIVO ESCRAVO)

```
#include "DHT.h" //permite a interação com sensores de temperatura e umidade da série DHT
#include <SoftwareSerial.h>

#define DHTPIN 2 //define a porta na qual o pino DATA do sensor está conectado
#define DHTTYPE DHT11 //define o sensor que será utilizado

SoftwareSerial unoSerial(5, 6); //define uma nova porta serial com o pino 5 como Rx e 6 como Tx
DHT dht(DHTPIN, DHTTYPE);

void setup(){
  Serial.begin(115200); //define baud rate da porta serial padrão
  unoSerial.begin(38400); //define baud rate da nova porta serial
  dht.begin(); //inicializa o sensor DHT
  delay(2000);
}
void loop()
{
  if (unoSerial.available()) { //caso existam dados disponíveis para a leitura na nova porta serial unoSerial
    if(unoSerial.read() == '1'){ //verifica se o dado recebido é igual a 1. Indicando uma nova solicitação de leitura
      Serial.println("Solicitação recebida!");
      sendDataToESP(); //envia os dados de leitura do sensor DHT11 para o ESP
```

```
    }
  }
}

void sendDataToESP(){
  float humidity= dht.readHumidity(); //salva os dados de umidade
  float temperature = dht.readTemperature(); //salva os dados de temperatura
  Serial.print(" T: ");
  Serial.print(temperature);
  Serial.print("°C; ");
  Serial.print("H: ");
  Serial.print(humidity);
  Serial.println("% ");
  String data = "Arduino: Temperatura = " + String(temperature) + "°C ; Umidade = "
+String(humidity) +"%"; //monta a string que será enviada para o ESP
  unoSerial.println(data); //envia a string para o ESP por meio da escrita de dados na porta
serial
}
```