

**ISABELLA BOLOGNA SALOMÃO
RENATO DE OLIVEIRA FREITAS**

**REDE DE DISPOSITIVOS PARA MONITORAMENTO
DE QUALIDADE E CONFORTO EM ESCRITÓRIOS**

São Paulo
2020

**ISABELLA BOLOGNA SALOMÃO
RENATO DE OLIVEIRA FREITAS**

**REDE DE DISPOSITIVOS PARA MONITORAMENTO
DE QUALIDADE E CONFORTO EM ESCRITÓRIOS**

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista com ênfase em Computação.

**ISABELLA BOLOGNA SALOMÃO
RENATO DE OLIVEIRA FREITAS**

**REDE DE DISPOSITIVOS PARA MONITORAMENTO
DE QUALIDADE E CONFORTO EM ESCRITÓRIOS**

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista com ênfase em Computação.

Área de Concentração:
Engenharia Elétrica

Orientadores:
Prof. Dr. Gustavo P. Rehder
Prof.^ª Dra. Cíntia Borges Margi

São Paulo
2020

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Catalogação-na-publicação

Salomão, Isabella Bologna

Rede de Dispositivos para Monitoramento de Qualidade e Conforto em Escritórios / I. B. Salomão, R. O. Freitas -- São Paulo, 2020.
p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.INTERNET DAS COISAS 2.CONFORTO NO TRABALHO
3.SENSORIAMENTO REMOTO I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais
II.t. III.Freitas, Renato de Oliveira

Prof.^a Dra. Cíntia Borges Margi

Prof. Dr. Gustavo P. Rehder

Dedicamos esse trabalho aos amigos
da equipe ThundeRatz

AGRADECIMENTOS

Ao nosso amigo Thalles Carneiro, que nos ajudou diretamente no desenvolvimento do projeto.

Aos nossos orientadores, Gustavo Rehder e Cintia Margi, por toda a ajuda e no quanto nos ensinaram ao longo do ano. Ao técnico Carlos Ramos do LME que nos ajudou nos testes do projeto, e ao professor Vitor Nascimento do PSI que nos tirou muitas dúvidas no desenvolvimento do projeto.

Isabella: À minha família, minha mãe Rosangela, meu irmão Leonardo, meu pai Moacir e meu companheiro Kim, que me apoiaram para chegar até aqui.

Aos amigos que fiz na ThundeRatz, em especial ao Gustavo Hama, Renzo Abensur, Lucas Haug, Leonardo Baltazar, Felipe Gomes, Jean Mello e Gustavo Barranova pelo companheirismo ao longo desse ano tão peculiar e terem me ajudado na jornada do autoconhecimento. Ao restante da equipe pelos anos incríveis e os ensinamentos que levarei para a vida toda.

Aos meus amigos do Jaraguá, Guilherme, Flávio, Lucas Bredariol, Catarine, Lucas Botassio e Igor, colegas desde a infância e que se mantêm presentes mesmo com a distância física.

Renato:

“Amigo é um mago do meigo abraço”

(Emicida)

RESUMO

Soluções para o monitoramento de parâmetros que remetem à qualidade e conforto de ambientes internos vêm se tornando interessantes, dado ao aumento no tempo que pessoas passam nesse tipo de ambiente, como escritórios. A partir da coleta desses dados, é possível adotar medidas para tornar o local estudado mais saudável e confortável para as pessoas ali presentes. O objetivo deste trabalho é desenvolver uma rede de dispositivos eletrônicos *Open Source* capaz de monitorar escritórios fechados, realizando a medição através de sensores de dados referentes à qualidade do ar, temperatura, luminosidade e ruído, além de coletar a opinião das pessoas sobre sua sensação de conforto no ambiente, para apresentar relatórios sobre o local em uma plataforma, que pode ser utilizada como base para tomar ações a fim de garantir o conforto de seus ocupantes. A rede de dispositivos proposta é implementada utilizando o protocolo *Bluetooth Mesh*, uma tecnologia que aparece cada vez mais no escopo de Internet das Coisas e ambientes inteligentes, em conjunto com Wi-Fi e o protocolo MQTT, visando enviar os dados coletados pelos diversos sensores para uma plataforma hospedada em nuvem.

Palavras-Chave – Rede de Sensores Sem Fio. Internet das Coisas. Rede Bluetooth Mesh. Escritório Inteligente.

ABSTRACT

Solutions for monitoring parameters that refer to quality and comfort of indoor environments have become interesting, given the increasing time that people spend in this type of places, such as offices. Based on the collection of this data, it is possible to take action to make the studied place healthier and more comfortable for the people in it. The main goal of this work is to develop an Open Source electronic device network capable of watching out closed office environments, using sensors to take measurements of air quality, temperature, luminosity and noise and, in addition, acquire people's opinions on their sensation of comfort at the given place, to present data reports in a selected platform that can be used to take actions in order to ensure comfort and the overall environment quality. The proposed device network is implemented by using the Bluetooth Mesh protocol, a growing technology used more and more on the Internet of Things and smart enviroment applications, in addition to Wi-Fi and MQTT protocols, aiming to send the collected data to a cloud platform.

Keywords – Wireless Sensor Network. Internet of Things. Bluetooth Mesh Network. Smart Office.

LISTA DE FIGURAS

1	Árvore de objetivos do projeto.	17
2	Arquitetura em camadas do protocolo BLE. Retirado de [1]	26
3	Comparação de performance entre 4 tecnologias de rede sem fio, em topologia de malha, de acordo com 9 parâmetros. Retirado de [2] (traduzido).	29
4	Composição lógica de um nó BLE Mesh. Retirado de [3]	31
5	Arquitetura geral da solução proposta	34
6	Diagrama de Blocos simplificado do dispositivo	36
7	Arquitetura da rede implementada	41
8	Diagrama de Blocos de Hardware do Protótipo	42
9	Esquemático do Protótipo	43
10	Design da PCB no Altium Designer	44
11	Placa fabricada e montada	44
12	Camada alto nível do firmware dos dispositivos	45
13	Arquitetura dos Sensores	46
14	Diagrama de tempos do modo <i>forced</i> . Retirado de [4]	46
15	Fluxograma da rotina de escrita em um registrador virtual do sensor AS7262	48
16	Fluxograma da rotina de leitura de um registrador virtual do sensor AS7262	49
17	Fluxograma da lógica de firmware para operação do sensor AS7262	49
18	Fluxograma da rotina de execução de um comando para interface com o sensor SGP30	50
19	Lógica de execução de firmware para coleta de dados do sensor SGP30	50
20	Arquitetura do Firmware de Coleta de Feedback	51
21	Telas com as perguntas do feedback	52
22	Máquina de Estados da coleta de Feedback	53

23	Arquitetura do módulo de Conectividade	54
24	Estrutura de dados principal utilizada pelos modelos da rede BLE Mesh	54
25	Composição lógica dos nós da rede BLE Mesh	55
26	Arquitetura lógica da rede BLE Mesh	56
27	Exemplo de envio de um bloco de dados ao servidor	57
28	Conexão do Grafana com a fonte de dados InfluxDB	63
29	Exemplo de consulta feita em um painel do Grafana	63
30	Arquitetura do servidor em nuvem	65
31	Case Mecânico	66
32	Testes de intensidade da luz	69
33	Testes de qualidade do ar	70
34	Testes de temperatura e umidade	71
35	Montagem final do dispositivo	72
36	Coleta dos sensores no IDF-Monitor	72
37	Gráficos de coletas do Gateway no Dashboard	73
38	Telas do aplicativo nRF-Mesh	73
39	Tela do app nrf mesh - escolha do grupo	74
40	Últimas medições dos indicadores no Dashboard	74
41	Feedbacks no Dashboard	75
42	Exemplo de ocorrência de alerta de temperatura no Dashboard	75
43	Exemplo de recebimento de notificações de alerta no serviço Slack	76
44	Bill of Materials (Lista de Materiais)	80
45	Arquitetura do Firmware	81

LISTA DE TABELAS

1	As concentrações de TVOC e seu impacto. Retirado de [5].	20
2	Tabela comparativa de estudos em dispositivos de monitoramento de ambientes	22
3	Comparação de dispositivos presentes no mercado.	23
4	Análise Comparativa de Protocolos de Comunicação para Sistemas IoT: MQTT, CoAP, AMQP e HTTP. Retirado de [6]	28
5	Comparação de processadores, preços em dólar. Elaborado pelos autores. . . .	38
6	Registradores físicos do sensor AS7262. Retirado de [7] (traduzido).	48
7	<i>Measurements</i> presentes no banco de dados	60
8	<i>Tag Keys</i> e <i>Field Keys</i> que identificam cada tipo de sensor e dados coletados .	60
9	Tabela de redirecionamento configurada para o NGINX	64
10	Tabela de Custos do Hardware	66
11	Tabela de dados coletados na calibração do sensor AS7262	69

SUMÁRIO

1	Introdução	15
1.1	Motivação	15
1.2	Objetivos	16
1.3	Árvore de Objetivos	16
2	Estado da Arte e Trabalhos Relacionados	18
2.1	Indicadores de Qualidade e Conforto	18
2.1.1	Regulamentações e Normas	18
2.1.2	Qualidade do Ar	19
2.1.2.1	CO_2	19
2.1.2.2	VOC	20
2.1.3	Conforto Visual	20
2.2	Trabalhos Relacionados a Qualidade de Ambientes Internos	21
2.3	Soluções no Mercado	23
2.4	Tecnologias Relevantes	25
2.4.1	Protocolos de Rede	25
2.4.2	Protocolos de Aplicação	26
2.4.3	Topologias de Rede	28
2.4.4	Redes Bluetooth Mesh	30
3	Especificação	34
3.1	Requisitos Técnicos	34
3.1.1	Rede de Sensores	34
3.1.2	Parâmetros de Medição	35

3.1.3	Software	35
3.2	Especificação Técnica	36
3.2.1	Arquitetura do Dispositivo	36
3.2.1.1	Sensores	36
3.2.1.2	Feedback	37
3.2.1.3	Processador	38
3.2.1.4	Alimentação	39
3.2.2	Protocolos de Comunicação e Arquitetura da Rede	39
3.2.3	Software	40
3.3	Considerações da proposta apresentada	41
4	Desenvolvimento	42
4.1	Hardware	42
4.1.1	Esquemático	43
4.1.2	<i>Printed Circuit Board (PCB)</i>	43
4.2	Firmware	44
4.2.1	Sensores	45
4.2.1.1	BME280	45
4.2.1.2	AS7262	47
4.2.1.3	SGP30	48
4.2.1.4	Microfone	50
4.2.2	Feedback	51
4.2.3	Conectividade	53
4.2.3.1	Bluetooth	53
4.2.3.2	Wi-Fi	56
4.3	Software	58
4.3.1	Eclipse Mosquitto	58

4.3.2	InfluxDB	59
4.3.3	Telegraf	60
4.3.4	Grafana	62
4.3.5	Servidor	64
4.4	Mecânica	65
4.5	Custos	66
5	Verificação do Projeto	68
5.1	Validação dos Subsistemas	68
5.1.1	Intensidade da Luz - AS7262	68
5.1.2	Qualidade do Ar - SGP30	70
5.1.3	Temperatura e Umidade - BME280	71
5.2	Validação do Dispositivo	71
6	Considerações Finais	77
6.1	Conclusões do Projeto	77
6.2	Perspectivas de Continuidade	78
6.2.1	Hardware	78
6.2.1.1	Alimentação	78
6.2.2	PCB	78
6.2.3	Firmware	78
6.2.3.1	Microfone	78
Apêndice A – Imagens		80
A.1	BOM	80
A.2	Arquitetura do Firmware	80
Referências		82

1 INTRODUÇÃO

1.1 Motivação

Com o aumento do tempo que as pessoas passam em ambientes fechados, como escritórios, há um crescente interesse em monitorar e controlar tais ambientes, visando uma melhora na saúde e conforto das pessoas, impactando a qualidade de vida e até a produtividade. Espaços que implementam essas soluções são comumente chamados de prédios inteligentes (*smart buildings*, do inglês). É possível até mesmo que esse controle seja utilizado para uma atuação de maneira energeticamente sustentável e, dentro desse contexto, surgem os *green buildings* (em português, construções sustentáveis) [8] [9].

No desenvolvimento de construções civis sustentáveis, torna-se necessário que seja pensada a implementação automatizada deste monitoramento dos ambientes desde o projeto da edificação e sua concepção, ocorrendo de forma integrada à construção. Uma pesquisa mais aprofundada sobre o conforto dos ambientes pode interferir no projeto, de modo que sejam repensados materiais utilizados e sistemas de aquecimento, ventilação, iluminação, dentre outros. Assim como a sua implementação, pesquisas na área de conforto têm se tornado cada vez mais importantes.

Foi com essa necessidade e a proposta de desenvolver um dispositivo eletrônico que o professor Vanderley M. John, do departamento de Construção Civil da Poli (PCC) e coordenador do CICS (Centro de Inovação em Construção Sustentável da USP) [10], entrou em contato conosco. A fim de viabilizar o estudo de qualidade e conforto em ambientes internos (ou IEQ, do inglês *Indoor Environmental Quality*), faz-se necessária a medição de diversos indicadores no ambiente e uma coleta periódica da opinião das pessoas, atrelada às medições quantitativas para que, dessa forma, seja possível identificar como uma afeta a outra.

Podemos elencar 4 parâmetros principais que metrificam a qualidade de ambientes, sendo eles qualidade térmica, luminosa, olfativa (qualidade do ar) e acústica. Estudos e soluções presentes no mercado que abordam a coleta de parâmetros de ambientes fechados, em sua maioria, são focados em monitorar apenas alguns desses indicadores, discutidos no capítulo

2 deste trabalho, além de poucos apresentarem um esforço em observar a real percepção de conforto das pessoas que ocupam tal local e correlacioná-la com os dados medidos.

O desenvolvimento de um dispositivo embarcado tem, portanto, além de uma aplicação prática monitorando a qualidade para as pessoas, também grande utilidade em pesquisas de construção civil e arquitetura, com medições mais precisas e incluindo um elemento muitas vezes deixado de lado: o fator humano.

Em edifícios, escritórios são hoje os que ocupam a maior área física e tem o maior consumo de energia, sendo sistemas de iluminação, aquecimento e resfriamento (como ar condicionado) os principais causadores do alto consumo [9]. Por isso, escritórios são o nicho escolhido para o estudo de conforto pelo departamento PCC e, consequentemente, para a implementação dessa rede de dispositivos.

1.2 Objetivos

O objetivo deste trabalho é o projeto de uma rede de dispositivos sensoreados que viabiliza o estudo de qualidade e conforto em escritórios, fazendo coletas de ao menos uma informação relativa aos principais indicadores de qualidade do ambiente. Para uma cobertura completa de grandes escritórios é importante que existam diversos dispositivos espalhados, conectados em rede. Esse grande volume de dados coletados será salvo em um banco de dados e apresentado de forma gráfica em uma plataforma para facilitar a interpretação e análise durante sua aplicação, como ferramenta de estudo.

Além da coleta de dados que remetem à qualidade geral do local estudado, propõe-se também que cada dispositivo seja capaz de fazer a coleta de *feedback* dos ocupantes através de uma interface no próprio dispositivo com perguntas que abordam o estado atual do ambiente, sendo necessário que os dispositivos estejam próximos à pessoa da qual serão coletadas as opiniões relativas ao conforto para que seja possível relacionar dados medidos pelos sensores com o conforto percebido pelas pessoas.

1.3 Árvore de Objetivos

A árvore de objetivos é uma representação gráfica dos meios necessários, chamados objetivos específicos, para alcançar o objetivo geral do projeto. Para atingir o objetivo geral do projeto, que é a realização de uma rede de dispositivos, com monitoramento do ambiente e apresentação dos dados, a seguinte árvore de objetivos foi elaborada, com a porcentagem de

dedicação ao lado de cada um.

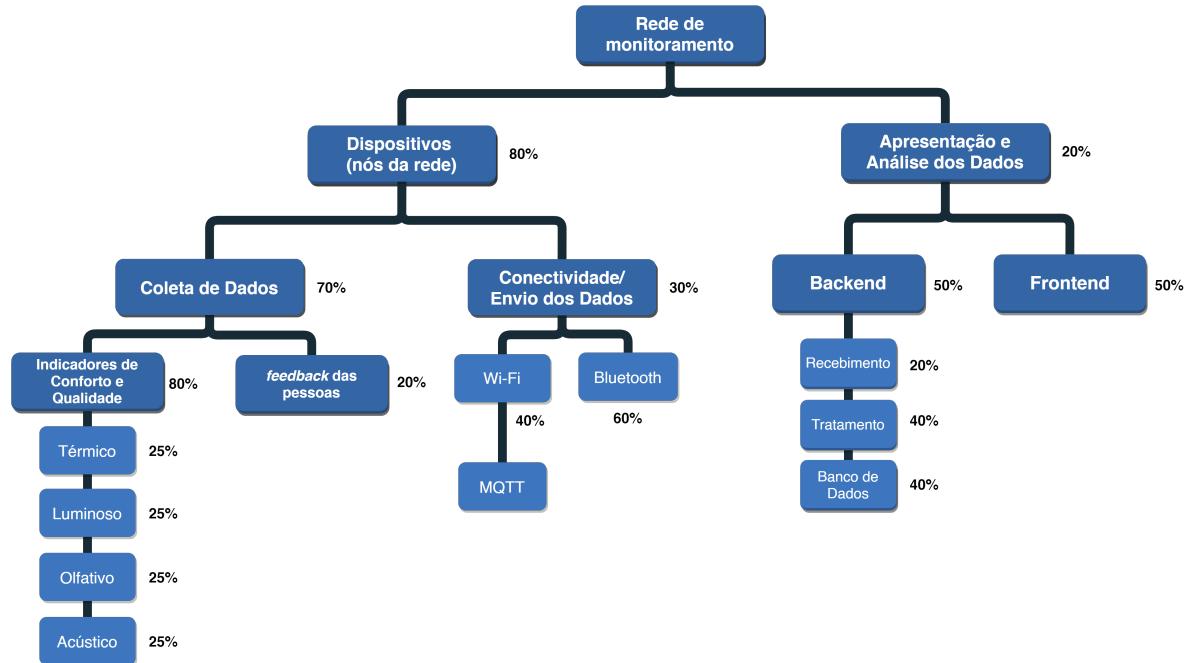


Figura 1: Árvore de objetivos do projeto.

2 ESTADO DA ARTE E TRABALHOS RELACIONADOS

2.1 Indicadores de Qualidade e Conforto

Como qualidade e conforto são termos subjetivos, vamos tratar aqui como “qualidade do ambiente” as condições recomendadas por normas e pesquisas com base em indicadores. Isto é, será considerado um ambiente de boa qualidade o que atender às faixas de operação pré-determinadas, funcionando como um aviso para o ocupante caso as medidas indiquem que os parâmetros do ambiente estão fora do recomendado.

Já o conforto será atrelado à percepção do usuário quanto ao ambiente. Apesar de o ambiente ser considerado saudável ou de qualidade, existem muitos fatores que afetam a sensação das pessoas, de forma que apenas a definição de uma faixa de operação não implica em bem-estar.

O conforto e a qualidade em ambientes internos são determinados através de quatro principais indicadores: **térmico, acústico, luminoso e olfativo/qualidade do ar** [11].

2.1.1 Regulamentações e Normas

A legislação brasileira determina os valores máximos e mínimos dos indicadores de conforto no ambiente para que haja boas condições de trabalho através das seguintes normas:

NR17 do Ministério do Trabalho [12]

17.5. Condições ambientais de trabalho.

17.5.2. Nos locais de trabalho onde são executadas atividades que exijam solicitação intelectual e atenção constantes, tais como: salas de controle, laboratórios, escritórios, salas de desenvolvimento ou análise de projetos, dentre outros, são recomendadas as seguintes condições de conforto:

a) níveis de ruído de acordo com o estabelecido na NBR 10152, norma brasileira registrada no INMETRO;

b) índice de temperatura efetiva entre 20 °C (vinte) e 23 °C (vinte e três graus centígrados);

[...]

d) umidade relativa do ar não inferior a 40 (quarenta) por cento.

17.5.2.1. Para as atividades que possuam as características definidas no subitem 17.5.2, mas não apresentam equivalência ou correlação com aquelas relacionadas na NBR 10152, o nível de ruído aceitável para efeito de conforto será de até 65 dB (A)

[...]

17.5.3.3. Os níveis mínimos de iluminamento a serem observados nos locais de trabalho são os valores de iluminâncias estabelecidos na NBR 5413, norma brasileira registrada no INMETRO.

NBR 10152 [13] para Escritórios

Salas de reunião: 30 - 40 dB(A)

Salas de gerência, Salas de projetos e de administração: 35 - 45 dB(A)

Salas de computadores: 45 - 65 dB(A)

Salas de mecanografia: 50 - 60 dB(A)

NBR 5413 [14]

Para escritórios: 500 - 750 - 1000 lux

2.1.2 Qualidade do Ar

Não há, na legislação brasileira, regulamentação de níveis de concentração de dióxido de carbono (CO_2) e compostos orgânicos voláteis, ou VOC (do inglês, *Volatile organic compounds*), que definem a qualidade do ar. A norma a seguir fala a margem esperada, mas também sem recomendações de operação.

ISO 16017-2:2003

is applicable to the measurement of airborne vapours of VOCs in a concentration range of approximately 0,002 mg/m³ to 100 mg/m³ individual organic for an exposure time of 8 h

Por conta disso, nos baseamos em estudos que tentam relacionar as concentrações de CO_2 e de VOC com efeitos na saúde e produtividade das pessoas.

2.1.2.1 CO_2

Segundo [15], o CO_2 apresenta concentrações mais altas em ambientes fechados, onde é esperado entre 700 e 2.000 ppm, em comparação a cerca de 400 ppm em ambientes abertos em áreas urbanas [16].

O CO_2 , além de ser um gás asfixiante e perigoso em altas concentrações (acima de 4.0000 ppm), também pode afetar a saúde quando em níveis moderados (abaixo de 2.000 ppm). De acordo com o Winsconsin Department of Health Services[17], são os efeitos causados na saúde:

- 250 a 400 ppm: Normal, concentração ambientes abertos
- 400 a 1.000 ppm: concentração típica em lugares fechados com pessoas, com boa circulação de ar
- 1.000 a 2.000 ppm: pode causar cansaço e falta de ar
- 2.000 a 5.000 ppm: dores de cabeça, sonolência, e falta de ar mais intensa. Baixa concentração, perda de atenção, aumento da frequência cardíaca, e náusea
- 40.000 ppm: Pode causar séria insuficiência respiratória, danos permanentes ao cérebro, coma e até a morte

2.1.2.2 VOC

Os compostos orgânicos voláteis são partículas que ficam suspensas no ar, podendo se originar de produtos (sintéticos ou naturais) utilizados no ambiente, como tintas, solventes, produtos de limpeza e perfumes, que podem causar odor perceptível pelo ser humano [15].

A concentração esperada é entre 0,2 e 0,5 mg/m³, ou entre 62 e 150 ppb, e mesmo que nem todos os compostos presentes no ar sejam nocivos à saúde, por precaução é recomendado que estes sigam os níveis apresentados na tabela 1.

Nível de TVOC	Preocupação
Abaixo de 93 ppb	Baixa
93 a 150 ppb	Pouca
150 a 310 ppb	Moderada
310 a 930 ppb	Alta

Tabela 1: As concentrações de TVOC e seu impacto. Retirado de [5].

2.1.3 Conforto Visual

Além da **intensidade da luz incidente**, cujos níveis são estabelecidos na legislação, a **temperatura da cor** da luz incidente também tem grande relevância. Há muitos anos, sabe-se que a luz azul emitida, de maior temperatura, causa danos à retina [18].

2.2 Trabalhos Relacionados a Qualidade de Ambientes Internos

O artigo “Development of low-cost indoor air quality monitoring devices: Recent advancements”, de 2020 da Universidade do Porto [19] traz uma comparação de projetos focados em monitoramento de qualidade do ar. A tabela 2 mostra os estudos que possuem uma maior relevância para o nosso projeto.

Estudo	Parâmetros monitorados	Processador e comunicação
Parkinson et al.[20]-[21]	Temperatura, Umidade CO_2 , TVOC, CO, Vel. do Ar Som Intensidade Luminosa	ARM-Cortex On-board e cloud storage LTE
Karami et al.[22]	Temperatura, Umidade CO_2 , VOC Intensidade Luminosa Ocupação (PIR)	Arduino UNO Cloud storage Zigbee VOLTTRON Software
Carre and Williamson[23]	Temperatura, Umidade CO_2 , Vel. do Ar Intensidade Luminosa Som, Ocupação (PIR)	Arduino Mega 2560 On-board e cloud storage 3G
Tiele et al.[24]	Temperatura, Umidade CO_2 , TVOC, CO Intensidade Luminosa	Feather M0 On-board storage
Chanthakit and Rattanapoka[25]	Temperatura, Umidade CO Intensidade Luminosa	ESP8266 MQTT Protocol Dados não são salvos (projeto futuro)
Scarpa et al.[26]	Temperatura, Umidade CO_2 Intensidade Luminosa Movimento (IR)	Arduino Onboard e cloud storage Wi-Fi (módulo ESP8266) (continua)

(conclusão)

Estudo	Parâmetros monitorados	Processador e comunicação
Jiang and Huacon[27]	Temperatura, Umidade	Raspberry Pi 3B
	CO_2	Cloud storage
	Som	Thingspeak platform
Salamone et al.[28]-[29]	Temperatura, Umidade	Arduino
	CO_2 , Vel. do Ar	On-board e cloud storage
	Intensidade Luminosa	Wi-Fi shield BlueSmRF (Bluetooth)
Ali et al.[30]	Temperatura, Umidade	Arduino PRO Mini
	CO_2	On-board storage
	Intensidade Luminosa	Sem comunicação wireless
	Ocupação (PIR)	(projeto futuro)
Brunelli et al.[31]	Temperatura, Umidade	Jennic NXP JN5148 SoC
	CO_2	Online storage
	Intensidade Luminosa	IEEE802.15.4/ZigBee PRO complaint module
Tapashetti et al.[32]	Temperatura, Umidade	(Wi-Fi) Marvell 88 MW302
	CO_2 , Gas	Cloud storage (AWS)
	Intensidade Luminosa	
Marques and Pitarma[33]		ESP8266 (Wi-Fi)
	CO, CH_4 , Eth	Cloud storage
		Thingspeak platform
Martín-Garín et al.[34]	Temperatura, Umidade	ESP8266 (Wi-Fi)
	CO_2	On-board e Cloud storage

Tabela 2: Tabela comparativa de estudos em dispositivos de monitoramento de ambientes

Vemos que as medições mais comuns são de qualidade do ar, e térmica (temperatura e umidade), sendo poucos os estudos onde são realizadas aquisições dos quatro indicadores para ambientes internos.

Quando há alguma comunicação nos dispositivos, Wi-Fi foi a solução mais utilizada, a fim de armazenar os dados coletados em nuvem. Ainda assim, todos os estudos mostravam dispositivos que operam de forma individual, não em rede.

O artigo *Targeted occupant surveys: A novel method to effectively relate occupant feedback with environmental conditions* [35] implementa um sistema de coleta de opinião sobre o estado do ambiente em um escritório através de questionários enviados para pessoas automaticamente pela plataforma *Qualtrics Survey Software* [36], além de um sistema de monitoramento de qualidade do ar, temperatura e umidade, com objetivo de relacionar a percepção pessoal com as condições do ambiente de maneira efetiva. Esse trabalho também foca apenas em dois indicadores e não possui nenhuma forma de conectividade com a Internet para disponibilizar as medições realizadas a algum sistema externo centralizado.

2.3 Soluções no Mercado

Projeto	Térmico	Luminoso	Acústico	Ar	Conectividade
Multi comfort [37]	Sim*	Sim*	Sim*	Sim*	Sim*
MC350[38]	Sim*	Sim*	Sim*		Bluetooth, App
Metriful Sense[39]	Temperatura, Umidade, Pressão	Intensidade	Volume, Freq.	VOC	
CoMoS[40]	Temperatura, Umidade, Veloc. Ar	Intensidade			Wi-Fi, SW Web
HC tech[41]	Temperatura, Umidade	Intensidade			Sigfox, SW Web
ECOMLITE [42]	Temperatura, Umidade, Pressão		Volume	CO_2 , CO, VOC, NO_2	Wi-Fi, Zigbee, Ethernet, SW Web
Netatmo[43]	Temperatura, Umidade,		Volume	CO2	Wi-Fi, App
Senlab O[44]	Temperatura, Umidade	Intensidade			LoRa
Comfort Click[45]	Temperatura, Umidade		Volume		Wi-Fi, App

Tabela 3: Comparação de dispositivos presentes no mercado.

(*) Solução não detalhada pela construtora

A tabela 3 mostra algumas das principais soluções encontradas no mercado para monitoramento de ambientes fechados. Ao observar essas soluções, é possível ver que a maioria atende a apenas alguns dos indicadores.

Multi Comfort [37], a solução mais completa encontrada, trata-se de uma sistema desenvolvido pela construtora Saint-Gobain de forma integrada à construção do edifício. Apesar de atender aos 4 indicadores de qualidade e conforto, essa solução não apresenta detalhes sobre as medições que são feitas, quais os elementos medidos ou a precisão dessas medições, assim como sobre a sua conectividade. Além de ser de difícil integração com outros dispositivos do mercado, apresenta grande dificuldade para ser implementado em um ambiente já construído. Nesses contextos, é introduzido o equipamento MC350[38] — da mesma empresa —, mas que não atende mais a todos os indicadores.

Em seguida, temos o Metriful Sense [39], que também atende a todos os indicadores de conforto, que é um produto novo em *crowdfunding*. Diferentemente das demais, esse não é um dispositivo completo, mas uma plataforma de sensores projetada para o monitoramento de ambientes. Necessita de uma interface com outro processador e possivelmente com um módulo *wireless* para que haja conexão com uma plataforma de análise.

Os demais produtos atendem em média apenas dois dos indicadores de conforto, sendo apenas o conforto térmico presente em todos.

O conforto luminoso, quando presente, é atendido de forma superficial, sendo medida apenas a intensidade da luz e não a cor da luz, que é um elemento importante na sensação de conforto [46]. Já no monitoramento da qualidade do ar, apenas uma das soluções [42] (cuja especificação está disponível) mede ao menos os dois principais elementos indicadores [15].

Vemos, assim como na literatura, que nenhum dos produtos existentes no mercado atende a todos os requisitos levantados para a nossa aplicação.

Além disso, nenhuma das soluções encontradas envolve diretamente a opinião das pessoas frequentando o ambiente na análise dos seus dados, apenas no controle do ambiente quando esse não atende os requisitos de qualidade ou de conforto.

Vale destacar, por fim, que na maioria dos casos essas soluções são fechadas, com protocolos que dificultam a integração dos dispositivos com outros que possam complementá-los em seu funcionamento, ou de forma que seja possível uma centralização da análise dos dados

coletados. Desenvolvendo um projeto *open-source* e com protocolos padrão de comunicação sem fio, isso deixa de ser um problema.

2.4 Tecnologias Relevantes

2.4.1 Protocolos de Rede

Uma rede de sensores sem fio (WSN, do inglês *Wireless sensor networks*) é composta de vários nós equipados com sensores, processador e um transceptor de rádio. Estes nós podem, assim, coletar informações do ambiente, processar dados e se comunicar com outros dispositivos [47].

Com o crescimento do conceito de Internet das Coisas, novas tecnologias para conexão rápida, segura e fácil de dispositivos são criadas e utilizadas por desenvolvedores em inúmeras aplicações. Segundo a pesquisa *Embedded Markets Study* realizada pela empresa Aspencore [48] e apresentada pelos meios EETimes [49] e Embedded [50] em 2019, as interfaces sem fio mais utilizadas por desenvolvedores em projetos de sistemas embarcados são Wi-Fi, *Bluetooth Low Energy (BLE)* e *Bluetooth Classic*. Essa pesquisa também aponta os protocolos de comunicação sem fio mais utilizados, dentre eles *BLE mesh*, Zigbee, 6LoWPAN e Thread.

Wi-Fi é o nome dado a uma família de tecnologias designadas para comunicação sem fio baseada no padrão IEEE 802.11 [51], amplamente utilizada em redes de área local (em inglês *local area network*, LAN) para prover acesso à Internet como uma alternativa à tecnologia cabeada Ethernet. Essa tecnologia opera nas faixas de 2,4GHz e 5GHz, sendo que a primeira permite uma taxa de transmissão de até 600 Mbits/s e a segunda até 1 Gbit/s em casos mais extremos [52]. A topologia comum de uma rede Wi-Fi é dada por um dispositivo que atua como ponto de acesso que disponibiliza o sinal sem fio para que dispositivos ao seu redor possam se conectar à rede, em uma topologia de rede em estrela. Deste modo, os dispositivos têm que estar a uma distância de poucos metros do ponto de acesso para que a comunicação não seja afetada, sendo essa distância cerca de 76m a 122m em ambientes internos [53], dependendo da antena utilizada.

Outra norma comumente utilizada em redes sem fio é a IEEE 802.15.4, que define especificações de camada física e de enlace da pilha de protocolos OSI para redes de comunicação sem fio que operam com baixa taxa de transmissão de dados (em inglês, *Lower Rate Wireless Personal Area Network*, ou LR-WPAN), que também utiliza a banda de frequência de 2,4GHz [54]. Os nós participantes da rede podem ser do tipo “função completa” (FFD, do inglês *full-function device*), agindo como coordenador da rede, podendo retransmitir mensagens para

outros nós, ou do tipo “função reduzida” (RFD, do inglês *reduced-function device*), destinados a serem dispositivos simples e com poucos requisitos de comunicação, podendo apenas se comunicar com FFDs, possibilitando aplicações com menor custo energético.

Por último, dentre as tecnologias citadas anteriormente, o Bluetooth surgiu com objetivo de substituir a conexão por fios entre dispositivos móveis comuns como celulares e fones de ouvido, porém já evoluiu muito e pode ser encontrada em diversas aplicações. O IEEE padronizou a tecnologia como IEEE 802.15.1, mas atualmente mantida por outra organização, a *Bluetooth Special Interest Group* (Bluetooth SIG), composta por mais de 35 mil empresas parceiras, que gerencia o desenvolvimento e define os padrões da tecnologia. Em 2009, a Bluetooth SIG definiu uma nova versão de baixo consumo energético, chamada de *Bluetooth Low Energy* (BLE), visando aplicações em que não é necessária comunicação contínua como, por exemplo, sensores que coletam dados em intervalos de tempo espaçados. O BLE vem se tornando uma das principais tecnologias para aplicações IoT, principalmente por estar presente na maioria dos dispositivos mais comuns utilizados no dia-a-dia das pessoas, como computadores, *smartphones* e *tablets*, o que facilita a comunicação do usuário com outros dispositivos específicos. As aplicações BLE operam de 2,4 a 2,485GHz com taxas de transmissão entre 0,27 e 1,37 Mbps [55].

2.4.2 Protocolos de Aplicação

O protocolo BLE possui uma especificação que engloba desde a camada física até a de aplicação, como mostrado na figura 2, diferentemente dos outros dois protocolos apresentados.

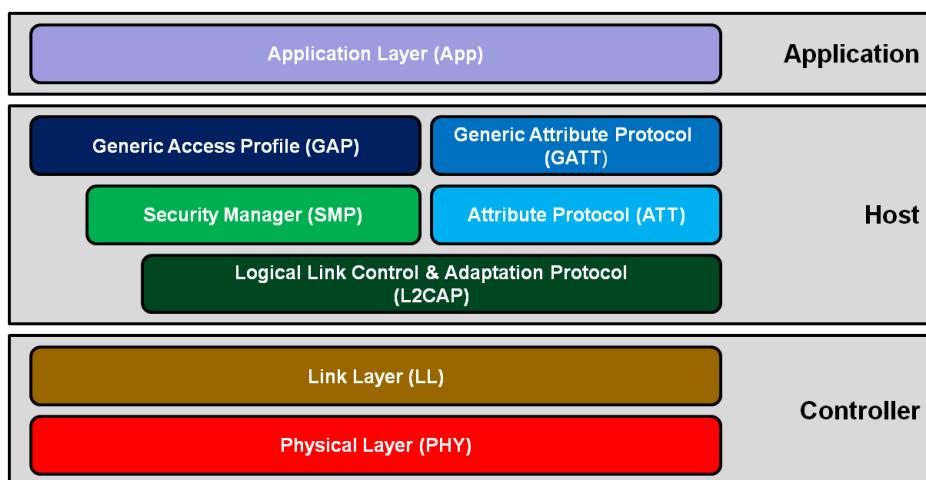


Figura 2: Arquitetura em camadas do protocolo BLE. Retirado de [1]

Para o caso do Wi-Fi no contexto de IoT, em que dispositivos simples e com pouca capacidade de processamento, como sensores, precisam estar conectados a servidores via In-

ternet, há a necessidade do uso de protocolos de aplicação também mais simples. O protocolo HTTP é um dos principais protocolos de aplicação utilizados na Web, porém possui algumas características que o tornam não muito adequado para aplicações nesse contexto, como a necessidade de longos cabeçalhos e regras antes do corpo da mensagem, que podem ocasionar *overheads* quando tratamos que dispositivos com pouca capacidade de processamento, a necessidade do cliente esperar por uma resposta do servidor e a comunicação ser unidirecional, sempre iniciada pelo cliente para o servidor.

Um protocolo comumente utilizado em aplicações IoT é o *Message Queuing Telemetry Transport* (MQTT) [56], implementado sobre a pilha TCP/IP, criado pela IBM em 1999, que utiliza o modelo *Publisher/Subscriber*, onde dispositivos MQTT publicam mensagens em tópicos em um dispositivo centralizado, chamado *Broker*, e se inscrevem em outros tópicos para receberem mensagens. O papel do *Broker* é receber as mensagens publicadas nos diferentes tópicos e redirecioná-las para os dispositivos que estão inscritos nesses tópicos. Desse modo, o protocolo MQTT apresenta grande escalabilidade, facilitando a implementação de redes com muitos dispositivos. É um protocolo orientado à conexão, já que a comunicação entre Cliente e *Broker* utiliza TCP na camada de transporte, podendo também ser utilizado TLS/SSL para segurança.

Visando utilizar a arquitetura de APIs REST presentes na Web em redes de dispositivos com baixa capacidade de processamento, facilitando a interoperabilidade com HTTP, o grupo de trabalho CoRE (*Constrained RESTful Environments*) do IETF definiu o protocolo CoAP (*Constrained Application Protocol*) no RFC7252 [57]. O CoAP faz uso do UDP, com cabeçalho fixo de 4 bytes, fazendo com que os datagramas sejam relativamente pequenos, dependendo apenas do tamanho da mensagem a ser transmitida, e tornando a comunicação mais leve, já que o UDP, diferente do TCP, não é voltado à conexão, ou seja, nenhum tipo de *handshake* precisa ser estabelecido entre os dispositivos. Isso possui um lado negativo, que é uma confiabilidade menor na entrega dos dados pela camada de transporte, sendo que o mecanismo de confiabilidade de dados precisa ser implementado na própria camada de aplicação. Assim como no HTTP, são definidas requisições GET, POST, PUT e DELETE para interação entre os dispositivos através de URLs, porém quem atua como servidor é o próprio dispositivo IoT, recebendo as requisições de um agente externo e respondendo adequadamente com os dados requisitados, o que facilita o gerenciamento de uma rede com vários dispositivos.

O artigo [6] faz a uma comparação desses protocolos, adicionando ainda um quarto, o AMQP, resumida na tabela 4.

Criteria	MQTT	CoAP	AMQP	HTTP
1. Year	1999	2010	2003	1997
2. Architecture	Client/Broker	Client/Server Client/Broker	or Client/Broker Client/Server	or Client/Server
3. Abstraction	Publish/Subscribe	Request/Response Publish/Subscribe	or Publish/Subscribe Request/Response	or Request/Response
4. Header Size	2 Byte	4 Byte	8 Byte	Undefined
5. Message Size	Small and Undefined (up to 256 MB maximum size)	Small and Undefined (normally small to fit in single IP datagram)	Negotiable and Undefined	Large and Undefined (depends on the web server or the programming technology)
6. Semantics/ Methods	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close	Get, Post, Put, Delete	Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close	Get, Post, Head, Put, Patch, Options, Connect, Delete
7. Cache and Proxy Support	Partial	Yes	Yes	Yes
8. Quality of Service (QoS)/ Reliability	QoS 0 - At most once (Fire-and-Forget), QoS 1 - At least once, QoS 2 - Exactly once	Confirmable Message (similar to At most once) or Non-confirmable Message (similar to At least once)	Settle Format (similar to At most once) or Unsettle Format (similar to At least once)	Limited (via Transport Protocol - TCP)
9. Standards	OASIS, Eclipse Foundations	IETF, Eclipse Foundation	OASIS, ISO/IEC	IETF and W3C
10. Transport Protocol	TCP (MQTT-SN can use UDP)	UDP, SCTP	TCP, SCTP	TCP
11. Security	TLS/SSL	DTLS, IPSec	TLS/SSL, IPSec, SASL	TLS/SSL
12. Default Port	1883/ 8883 (TLS/SSL)	5683 (UDP Port)/ 5684 (DLTS)	5671 (TLS/SSL), 5672	80/ 443 (TLS/SSL)
13. Encoding Format	Binary	Binary	Binary	Text
14. Licensing Model	Open Source	Open Source	Open Source	Free
15. Organisational Support	IBM, Facebook, Eurotech, Cisco, Red Hat, Software AG, Tibco, ITSO, M2Mi, Amazon Web Services (AWS), InduSoft, Fiorano	Large Web Community Support, Cisco, Contiki, Erika, IoTivity	Microsoft , JP Morgan, Bank of America, Barclays, Goldman Sachs, Credit Suisse	Global Web Protocol Standard

Tabela 4: Análise Comparativa de Protocolos de Comunicação para Sistemas IoT: MQTT, CoAP, AMQP e HTTP. Retirado de [6]

2.4.3 Topologias de Rede

Os protocolos de rede citados possuem, em geral, topologias de rede baseadas em estrela, com vários dispositivos periféricos conectados a um central, ou em árvore, que pode ser vista como várias redes em estrela interconectadas. No segundo caso, apenas os dispositivos centrais de cada rede em estrela poderia se comunicar com as outras, atuando como um dispositivo de borda. Desse modo, a conexão entre dispositivos periféricos e central da rede se dá de forma ponto-a-ponto, o que gera uma dependência forte do central e, caso ele apresente algum problema, pode afetar o funcionamento da rede inteira.

O crescimento e evolução das aplicações IoT requer topologias mais novas e complexas, com a capacidade de se ajustarem dinamicamente caso ocorra alguma mudança na rede. Com isso em mente, soluções de redes em malha começam a surgir nos protocolos anteriormente citados. Nessa topologia, os nós ficam interligados de forma não-hierárquica, permitindo a comunicação *many-to-many* entre os dispositivos da rede, possibilitando que dados de um

ponto qualquer seja enviado a outro ponto qualquer da rede.

Dada essa necessidade, as especificações dos protocolos de comunicação começaram a adicionar soluções de rede em malha, como BLE Mesh, 802.11s (Wi-Fi mesh) e Thread, que é construída em cima da especificação 6LoWPAN. O protocolo Zigbee já possui suporte a redes em malha desde sua primeira concepção. Essas definições são feitas ao nível da camada de rede, se utilizando das especificações das respectivas tecnologias para implementar novas maneiras de roteamento de dados entre os dispositivos.

O artigo *Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies*, publicado pela *Future Internet* [2] faz uma análise comparativa entre diferentes tipos de protocolos de comunicação atuando na topologia de rede em malha: família IEEE 802.15.4 (Zigbee e Thread), IEEE 802.11 (em especial a IEEE 802.11s que define redes em malha), LoRa Mesh e IEEE 802.15.1 (BLE Mesh).

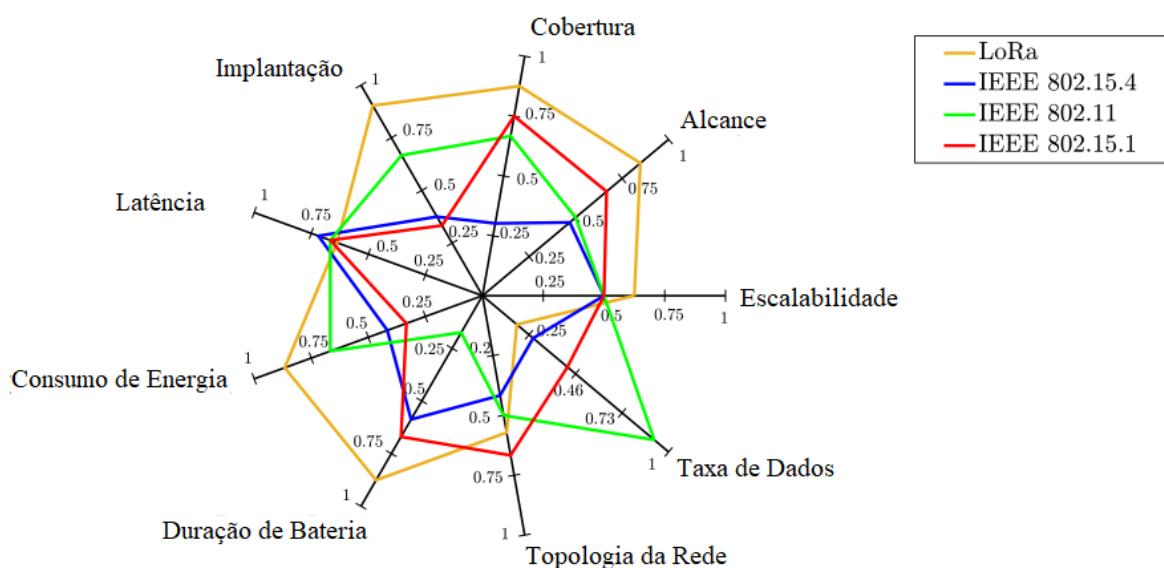


Figura 3: Comparação de performance entre 4 tecnologias de rede sem fio, em topologia de malha, de acordo com 9 parâmetros. Retirado de [2] (traduzido).

O gráfico mostrado na figura 3 pode ser utilizado para resumir de maneira completa a pesquisa realizada em [2]. O protocolo LoRa foge do escopo deste projeto, pois é destinado para comunicações entre distâncias quilométricas e possui taxa de transmissão de dados muito baixa, por isso será desconsiderado como alternativa para a aplicação deste projeto. É possível observar os seguintes pontos:

- A tecnologia Wi-Fi é claramente a que possui maior taxa de transmissão de dados, com área de cobertura e alcance razoáveis, mas com a desvantagem de possuir um consumo de energia elevado.

- Redes BLE Mesh são as mais eficientes em termos de consumo de energia, mostrado também no artigo [58], possuem grande área de cobertura, já que uma rede pode possuir até 32 mil nós [59], e possuem uma taxa de transmissão de dados alta, comparada com as soluções baseadas na IEEE 802.15.4.
- Todas as soluções apresentadas possuem alta escalabilidade e níveis de latência comparáveis.

2.4.4 Redes Bluetooth Mesh

Como citado anteriormente, *Bluetooth Mesh* é um padrão de rede em malha baseado no protocolo *Bluetooth Low Energy* (BLE) com a finalidade de permitir a comunicação do tipo *many-to-many* entre os dispositivos de uma mesma rede. Essa tecnologia se baseia nos seguintes princípios:

- *multi-hop*: mensagens enviadas são reencaminhadas por vários outros dispositivos até que atingam seu destino, permitindo que os dispositivos de origem e destino não estejam diretamente interconectados, o que possibilita criação de redes extensas.
- *multi-path*: cópias de uma mensagem são enviadas por diversos caminhos diferentes até chegar ao destino, sendo a primeira a chegar processada e as outras cópias descartadas, provendo redundância de dados no nível de rede.
- *multicast*: dispositivos enviam uma mesma mensagem para vários destinos diferentes.

O padrão foi idealizado visando possibilitar e encorajar interoperabilidade de dispositivos totalmente diferentes, desenvolvidos por diferentes fabricantes, como, tomado como exemplo um prédio inteligente, sensores que medem o estado atual do ambiente e enviam mensagens para controlar equipamentos de ar condicionado, cortinas e luzes inteligentes, bastando seguir as normas especificadas pela Bluetooth SIG de comunicação. Uma dessas padronizações é a definição de mensagens que serão compartilhadas entre os nós da rede. Cada nó possui estados que definem sua condição atual — ligada ou desligada, no caso de uma lâmpada — e, para que outro nó possa interagir com este, ele pode enviar mensagens que vão alterar ou ler o valor atual de tal estado. São elas:

- **SET**: mensagens desse tipo servem para alterar o valor do estado
- **GET**: enviada por um nó para ler o valor de um estado de outro nó

- **STATUS:** utilizada por um nó para notificar outros nós o valor do seu estado atual

Cada elemento de uma rede possui um endereço específico que o identifica, utilizado por outros nós para o envio de mensagens. Porém, a especificação também define o modelo de comunicação *Publish/Subscribe*, onde são definidos grupos com endereços fixos em que os nós podem publicar mensagens ou se inscrever para receber todas as mensagens publicadas nesse endereço. Isso faz com que, em redes grandes, não seja necessário saber o endereço de cada nó especificamente para o envio de mensagens, apenas os endereços dos grupos que cada nó deve publicar suas mensagens ou se inscrever, permitindo trocar dispositivos da rede em caso de problemas sem que a funcionalidade da rede seja afetada, bastando apenas manter os endereços dos grupos pré-definidos.

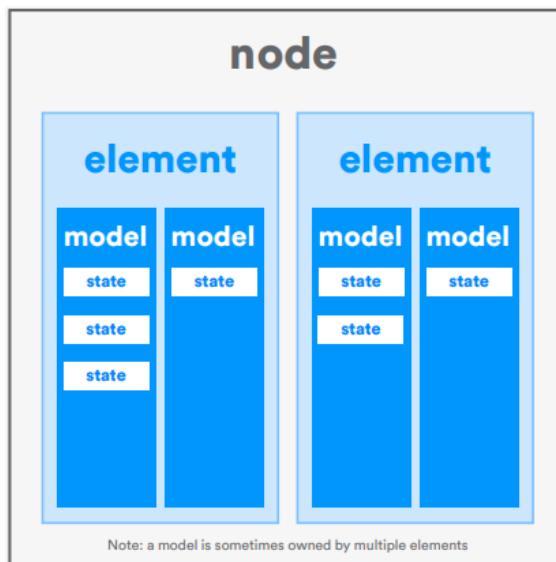


Figura 4: Composição lógica de um nó BLE Mesh. Retirado de [3]

A estrutura lógica de um nó da rede está representada na figura 4. Podemos dizer que um nó na rede equivale a um conjunto de um ou mais **elementos** (ou *elements*, em inglês), que é a estrutura lógica endereçável na rede, sendo que cada elemento em si é constituído por um ou mais **modelos** (ou *models*). Dessa forma, um nó (dispositivo físico) pode conter vários elementos que definem diferentes funcionalidades lógicas para esse dispositivo. Modelos são estruturas padronizadas que realizam um tipo específico de tarefa, constituídos por estados que o definem e mensagens pré-estabelecidas utilizadas para a interação com outros Modelos.

Seguindo a especificação, os Modelos podem ser do tipo **Cliente**, que não possuem estados e servem apenas para interagir com os Modelos do tipo **Servidor**, os quais possuem estados. O exemplo mais comum dessa definição é o caso de um interruptor interagindo com uma lâmpada: o interruptor implementaria o *Generic OnOff Client* e a lâmpada o *Generic OnOff*

Server, dado que quem possui o estado de ligada ou desligada é a lâmpada (servidor), alterado ou lido pelo interruptor (cliente).

Para que um dispositivo faça parte de uma rede e possa se comunicar dentro desta, ele necessita de uma chave específica, utilizada para criptografar todas as mensagens enviadas dentro de tal rede, chamada de Chave de Rede (*Network Key*, em inglês, ou *NetKey*). Esta chave é única e serve para definir uma rede específica. Todos os dispositivos que possuirem essa chave conseguem se comunicar entre si. Um mesmo dispositivo pode fazer parte de mais de uma rede, possuindo assim mais de uma *NetKey*. Além disso, os Modelos precisam de uma outra chave para interagir com os outros, chamada de Chave de Aplicação (*Application Key*, em inglês, ou *AppKey*), de forma que apenas os Modelos que possuírem a mesma *AppKey* conseguem se comunicar. Esse conceito torna possível a criação de subredes dentro de uma mesma rede BLE Mesh para diferentes aplicações, como separar dispositivos em cômodos de uma casa, diferentes salas em um prédio etc.

O processo de definição de quais redes o dispositivo fará parte (atribuindo-o as chaves necessárias) e quais os endereços que ele deve se inscrever e publicar mensagens é chamado de **Provisionamento**, realizado por um dispositivo especial com o papel de **provisionador**, sendo este o definidor inicial de uma rede e suas chaves. Normalmente, esse dispositivo é um *smartphone* que permite um controle mais interativo pelo usuário, mas pode ser também qualquer outro dispositivo que esteja equipado com essa função. Como mencionado, o processo de provisionamento consiste em atribuir a *NetKey* que define a rede ao dispositivo, assim como quais Modelos possuirão determinadas *AppKeys*.

O Bluetooth SIG define 52 Modelos padrões, listados e especificados em [59], divididos em quatro grandes grupos: modelos genéricos, modelos para sensores, modelos específicos para aplicações de iluminação e modelos relacionados a controle de tempo. Também podemos notar que todo Modelo Servidor possui um Modelo Cliente correspondente e vice-versa. Além desses modelos padrões, desenvolvedores podem projetar seus próprios modelos customizados, chamados de Modelos de Fornecedores (*Vendor Models*, em inglês), sendo necessário definir tanto o formato de dados que eles possuirão (Estados) e as mensagens que devem ser utilizadas para interagir com esse modelo.

Existem bilhões de dispositivos que implementam o protocolo Bluetooth sendo produzidos todo ano [60], porém muitos deles não são capazes de fazer parte de uma rede BLE Mesh nativamente, seja por não possuírem as camadas necessárias por escolha dos desenvolvedores, ou por simplesmente terem sido produzidos antes da especificação ser concluída. Para aproveitar essa grande quantidade de dispositivos presentes no mercado e tornar possível a interação

deles com redes BLE Mesh, existe um tipo especial de dispositivo que pode estar presente na rede, chamado **Proxy**, capaz de se comunicar através da especificação do BLE Mesh e pelos perfis GATT [61] ao mesmo tempo, traduzindo mensagens presentes na rede mesh para que esses dispositivos BLE possam também fazer parte da rede. Essa implementação é a que torna possível que um *smartphone* atue como um dispositivo provisionador, como citado anteriormente, que faça o papel de Cliente para acender luzes inteligentes em uma casa ou faça requisições para obter dados dispositivos com sensores, facilitando a interação de um usuário comum com diversas implementações de redes. O fato de que quase toda pessoa que possui um *smartphone* é capaz de interagir com qualquer rede BLE Mesh de forma nativa é um dos fatores que tornam essa tecnologia muito promissora para aplicações voltadas diretamente aos usuários, como dispositivos conectados em *smart homes* e *smart buildings*.

3 ESPECIFICAÇÃO

Neste capítulo, é discutida a solução proposta para o problema apresentado no capítulo 1, desenvolvida com bases nos requisitos técnicos definidos nas seções a seguir e nas pesquisas mostradas no capítulo 2. Em termos gerais, a arquitetura da solução, ilustrada na figura 5, é composta por uma rede de dispositivos sensoreados que se comunicam com um servidor em nuvem, sendo este responsável por armazenar os dados medidos em um banco de dados e, por fim, mostrá-los através de uma aplicação web com gráficos e tabelas, de forma que todos os dados sejam concentrados para facilitar sua análise. Ao final deste capítulo, revisitamos essa arquitetura, apresentando-a de forma mais detalhada com as tecnologias escolhidas para a realização do projeto.

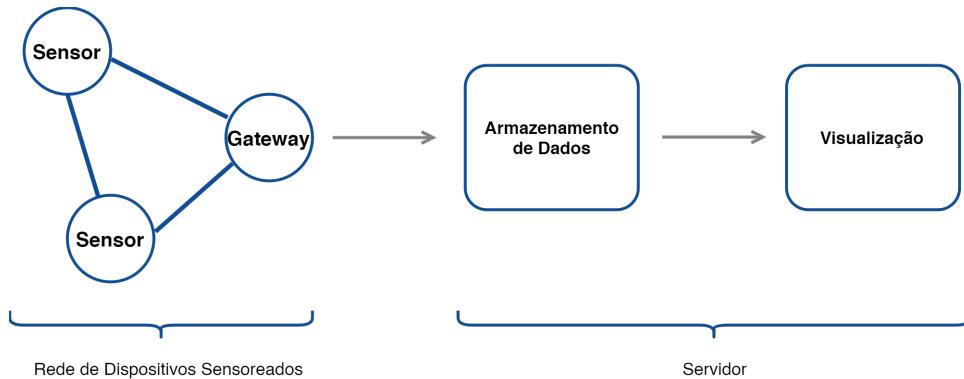


Figura 5: Arquitetura geral da solução proposta

3.1 Requisitos Técnicos

3.1.1 Rede de Sensores

Com base na pesquisa sobre os protocolos de rede mais adequados para aplicações de redes de sensores no contexto de Internet das coisas, optamos por utilizar a tecnologia *Bluetooth Low Energy* (BLE) em conjunto com Wi-Fi.

A fim de disponibilizar os dados coletados pelos diferentes sensores acerca do ambiente

para posterior análise, se faz necessária uma conexão à Internet dos dispositivos desenvolvidos. Essa conexão é realizada através do protocolo Wi-Fi, conectando o dispositivo em si a um ponto de acesso presente no ambiente analisado.

Já que o objetivo é instalar os dispositivos em escritórios, não seria ideal que todos eles ficassesem conectados à rede Wi-Fi do local. Como mostrado na seção 2.4.3, o protocolo Wi-Fi possui alto consumo energético, outro ponto negativo já que visamos alimentar os dispositivos com baterias. Dessa forma, o protocolo BLE será utilizado para interconectar os dispositivos, dado seu baixo consumo de energia, alta escalabilidade e alta taxa de transmissão de dados, quando comparado com as outras tecnologias apresentadas.

3.1.2 Parâmetros de Medição

Definimos as métricas de **qualidade do ambiente** com base nos níveis considerados saudáveis pela literatura e no que diz a legislação brasileira e as normas técnicas.

Não apenas esses elementos são importantes, mas também a combinação deles afeta a percepção de conforto pelas pessoas [62]. Assim, faz-se mais necessário que haja uma medição completa dos elementos presentes no ambiente a ser estudado.

Para cada um dos indicadores, decidimos medir as seguintes informações:

- Térmico: temperatura ambiente e umidade relativa
- Acústico: ruído ambiente
- Luminoso: intensidade e cor da luz incidente
- Qualidade do ar (e Olfativo): CO_2 e VOC (*volatile organic compounds*)

3.1.3 Software

Visando a disponibilidade dos dados para análise a longo prazo, precisamos que esses sejam armazenados em algum banco de dados. Soluções de armazenamento em nuvem são boas alternativas, pois possibilitam o acesso remotamente, além de garantir maior robustez.

Por fim, as medições coletadas devem ser apresentadas de forma gráfica através de um painel de dados (*data dashboard*), para que possam ser visualizadas de forma simplificada.

3.2 Especificação Técnica

Dados os requisitos apresentados na seção 3.1, definimos aqui as tecnologias utilizadas para a implementação dos três grandes blocos do projeto: o dispositivo sensoreado em si, os protocolos de comunicação para o compartilhamento dos dados coletados e a estrutura do servidor que recebe esses dados e os apresenta para o usuário final.

3.2.1 Arquitetura do Dispositivo

A partir da definição dos requisitos técnicos, chegamos a seguinte arquitetura do dispositivo:

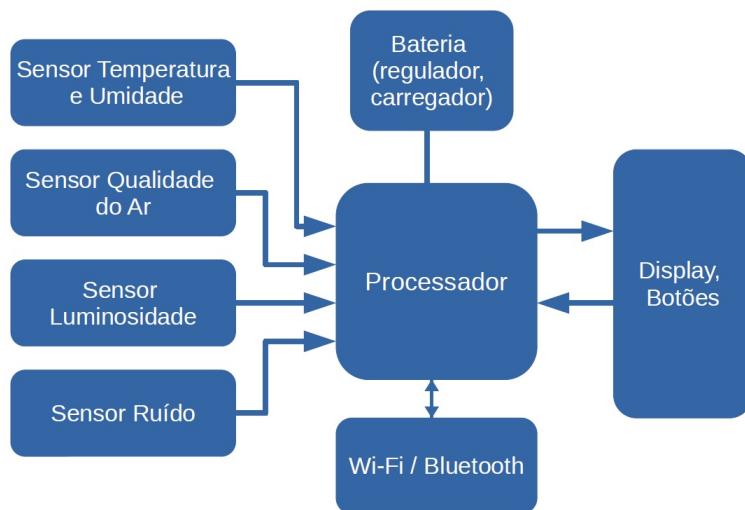


Figura 6: Diagrama de Blocos simplificado do dispositivo

3.2.1.1 Sensores

Com o objetivo de atender aos critérios apresentados para o monitoramento, foram escolhidos os seguintes sensores:

- **AS7262** [7], da AMS:

Atende aos requisitos de medição de *conforto luminoso*.

Medidas: Intensidade e cor da luz incidente.

A cor da luz, nesse sensor, é medida através de 6 canais, correspondendo aos espectros de luz vermelha (650 nm), laranja (600 nm), amarela (570 nm), verde (550 nm), azul (500 nm) e violeta (450 nm), ao invés de simples RGB, com resolução de 16 bits.

Comunicação: I²C, SPI ou UART (configurável)

- **BME280** [4], da Bosch:

Atende aos requisitos de *conforto térmico*.

Medidas:

- Temperatura entre -40 e 85 °C, com precisão de ±1.0 °C
- Umidade relativa com precisão de ±3%
- Pressão entre 300 e 1100 hPa, com precisão ±1 hPa

Comunicação: SPI ou I²C

- **SGP30** [63]:

Sensor para medições de aplicação *indoor*.

Medidas:

- TVOC entre 0 ppb e 60000 ppb, com resolução de 1 ppb
- CO₂ entre 400 ppm e 60000 ppm, com resolução de 1 ppb

Comunicação: I²C

- **Microfone de Eletreto:**

Em conjunto com um circuito amplificador, atende aos requisitos de *conforto acústico*.

Medida: intensidade do som no ambiente

Comunicação: Analógica, precisão de 12 bits (resolução do conversor analógico-digital do ESP32).

3.2.1.2 Feedback

Para interação com os usuários, foi implementado um sistema de coleta de *feedback* nos dispositivos, utilizando um *display* LCD OLED SSD1306 [64] de 128x64 pixels, com driver de comunicação I²C, e 2 botões do tipo *push button*. Perguntas acerca do estado atual do ambiente serão mostradas nesta tela junto às opções de resposta para cada uma e os usuários utilizam os botões para selecionar a resposta desejada.

O objetivo é que a interface seja simples, apenas perguntando se o usuário está confortável em relação aos quatro indicadores de conforto, como “Está confortável com a temperatura?” no caso de conforto térmico, e com respostas de “Sim” ou “Não”, para que a coleta do *feedback* não consuma muito tempo.

3.2.1.3 Processador

De acordo com a mesma pesquisa da Aspencore citada na seção 2.4.1 [48], 61% dos projetos de sistemas embarcados usam processadores de 32-bits, e 65% utiliza algum tipo de sistema operacional.

A partir das especificações dadas, listamos os periféricos necessários ao microcontrolador e buscamos as principais opções existentes no mercado para atuar como processador central do dispositivo.

Como a comunicação dos dispositivos é uma funcionalidade crucial, foi dada a preferência para os *System-on-a-Chip* (SoCs), ou Sistema em um chip, ao invés de microcontroladores e módulos *wireless* independentes. SoC é o nome dado a circuitos integrados que englobam processadores (ou microcontroladores, usualmente em dispositivos embarcados), memórias, dentre outros módulos, como circuitos para comunicação sem fio, personalizados para uma aplicação [65]. Assim, chegamos a três opções de SoCs com **bluetooth** integrado, mostradas na tabela 5.

Fabricante	CI	Preço	Placa de Desenvolvimento	Preço da Placa
Nordic Semi	BMD350	\$11,30	BMD350-EVAL	\$89,00
Espressif Systems	ESP32	\$3,80	ESP32-DevKitC	\$10,00
STMicroelectronics	BlueNRG-2	\$3,50	BlueNRG-Tile	\$50,00

Tabela 5: Comparação de processadores, preços em dólar. Elaborado pelos autores.

Analizando principalmente os ambientes de desenvolvimento, a documentação disponível, o preço dos CIs e de seus kits de desenvolvimento, optamos pela família **ESP32** [66].

Além do Wi-Fi como diferencial no SoC, a Espressif possui um bom suporte e ferramentas de desenvolvimento focadas em BLE e Wi-Fi, em especial para o uso de BLE Mesh, e um dos menores preços, assim considerado o melhor custo-benefício.

Especificações do ESP32: [67]

- **Processador:** Xtensa 32-bits, dual core
- **Wi-Fi:** 802.11 b/g/n
- **Bluetooth:** v4.2 BR/EDR e BLE

3.2.1.4 Alimentação

Para alimentar os dispositivos, foi pensado inicialmente em utilizar uma bateria recarregável, de forma que o dispositivo seja portátil e não necessite da rede elétrica para funcionar. A tensão da bateria seria então regulada para 3.3V a fim de alimentar todos os circuitos da placa, que operam nessa tensão.

Com um conector USB e um circuito apropriado integrado na placa, seria possível recarregar a bateria sem interromper o funcionamento do dispositivo. Para fins de teste, como a placa de desenvolvimento ESP32 possui um USB micro e um regulador de 3.3V, optamos por utilizá-lo como alimentação para os demais módulos. Os dispositivos ficam, assim, conectados ao computador ou a tomada durante os testes.

3.2.2 Protocolos de Comunicação e Arquitetura da Rede

Os dispositivos são interconectados por meio de uma rede *Bluetooth Mesh*, disponibilizando dados de medições dos sensores e *feedback* ao longo do dia por toda a rede. Essa arquitetura utiliza o conceito de rede por inundação, no qual os dados de um nó são enviados para vários outros nós, que atuam como retransmissores desses dados para outros dispositivos dentro do seu alcance, o que aumenta a área de cobertura da rede e sua confiabilidade, já que se um dos dispositivos se desconectar, outras rotas estão disponíveis para a propagação dos dados.

Apenas um dos dispositivos da rede está conectado também à Internet via Wi-Fi, atuando como um gateway para a rede, sendo assim necessário que este também esteja conectado à uma fonte fixa de energia. Essa conexão permite o envio dos dados coletados pela rede para um servidor externo ao sistema, possibilitando o acesso remoto aos dados.

O artigo [68] faz uma comparação dos protocolos mais utilizados em aplicações IoT, citados na seção 2.4.2, de maneira prática, utilizando um *testbed* muito parecido com o que utilizamos para o desenvolvimento desse projeto — microcontrolador ESP8266 ao invés do ESP32. A partir das conclusões presentes no artigo, vemos que o protocolo MQTT se mostrou mais estável em relação ao CoAP ao utilizar esse dispositivo restrito, aguentando maior carga de dados e maior confiabilidade na entrega dos mesmos, o que o torna ideal para a aplicação desse projeto.

3.2.3 Software

O uso do protocolo MQTT na rede de dispositivos torna necessária a implementação de um *MQTT Broker* para a recepção dos dados coletados. Utilizamos o Eclipse Mosquitto [69], uma implementação de código aberto disponibilizada pela Eclipse Foundation, bastante utilizado em projetos que usam essa tecnologia.

Há também, para esse projeto, a necessidade de armazenar os dados recebidos, como discutido anteriormente. Existem vários tipos de bancos de dados, separados em dois grandes grupos: bancos de dados relacionais, como PostgreSQL e MySQL, e os não relacionais (NoSQL), como o MongoDB. Dentro do segundo grupo, existe uma subcategoria que engloba os bancos de dados de séries temporais (*Time Series Data Bases*, TSDBs), especialmente criados para armazenar dados que remetem a evolução ao longo do tempo de alguma métrica, como medições de sensores ao longo do dia, e que são otimizados para tratar esses tipos de dados especificamente. O artigo [70] faz um comparativo entre bancos de dados relacionais e de série temporal, mostrando o quanto eficiente o segundo tipo é em lidar com dados gerados num cenário de IoT. O artigo mostra também várias opções de TSDBs, optando pelo InfluxDB, da empresa InfluxData, que possui grande suporte a integração a diferentes linguagens de programação, possui código aberto e de uso livre sob a licença MIT e apresenta um ótimo desempenho com o tratamento de dados, sendo ideal para o uso nesse projeto, no qual possuímos leituras de diferentes sensores em vários dispositivos que monitoraram o estado do ambiente ao longo do tempo.

Para que os dados recebidos pelo MQTT Broker sejam inseridos no banco de dados, utilizamos o Telegraf, da mesma desenvolvedora do InfluxDB, um programa designado para fazer a integração entre serviços, conectando fontes de dados aos destinos desejados. Esse serviço também é distribuído sob a licença MIT, com código aberto e de uso livre, com capacidade de se conectar com várias fontes de dados através de *plugins* de entrada, atuando como um MQTT Client no contexto desse projeto, se inscrevendo nos tópicos referentes às medidas de sensores que serão recebidas pelo MQTT Broker. Ao mesmo tempo, possui diferentes *plugins* de saída, que redirecionam os dados recebidos para os destinos desejados — banco de dados InfluxDB, nesse caso. Esse sistema de *plugins* para realizar as conexões entre os serviços já faz parte da solução Telegraf, bastando apenas configurá-los com as necessidades do projeto.

Por último, o painel de dados utilizado para apresentar os dados coletados foi implementado utilizando o Grafana, uma aplicação web que possibilita criar diferentes tipos de gráficos, tabelas e alertas e conectá-los com dados presentes em bancos de dados de maneira interativa.

Esse serviço também possui código aberto sob a licença Apache 2.0, que permite seu uso de forma livre. Ele possui integração com o InfluxDB embutida, o que possibilita acessar os nossos dados salvos através de *queries* e mostrá-los visualmente nos diferentes tipos de interfaces.

Todos esses serviços citados anteriormente são suportados por diferentes sistemas operacionais e, idealmente, precisam ser executados em um mesmo ambiente, como um servidor Linux, que pode ser implementado tanto em sistemas de hardware mais simples e embarcados, como um Raspberry Pi, quanto em sistemas mais robustos com mais memória e capacidade de processamento, ligados a uma rede local no ambiente monitorado para o fácil acesso aos dados. Porém, visando a alta disponibilidade dos dados coletados como um dos requisitos do projetos, foi utilizada uma solução de servidor em nuvem para hospedar os serviços citados, tornando possível acessá-los remotamente e de forma confiável. Existem várias empresas que oferecem soluções de servidores privados em nuvem (VPS, do inglês *Virtual Private Server*), e com preços equivalentes, como a *Digital Ocean*, *Linode*, *Google Cloud* e *IBM Cloud*, porém a *Amazon Web Services (AWS)* é a que oferece o maior tempo de uso grátis (1 ano contra cerca de 3 meses para os outros), tornando essa a melhor escolha para a realização desse projeto.

3.3 Considerações da proposta apresentada

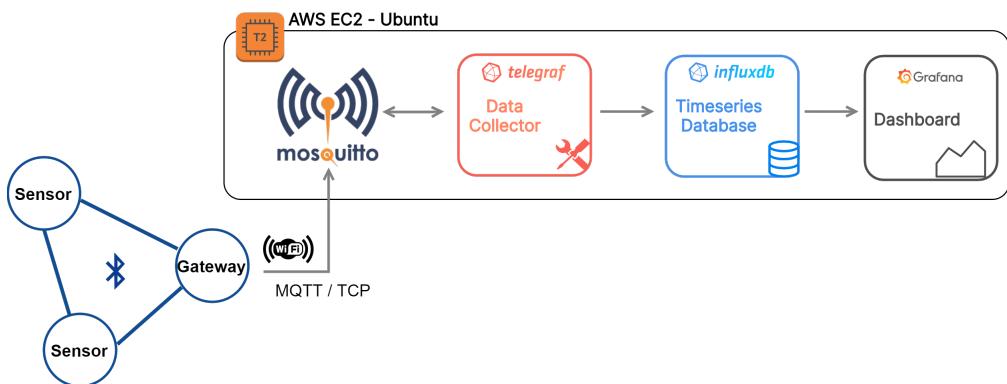


Figura 7: Arquitetura da rede implementada

A arquitetura da rede, então, tem a forma apresentada na figura 7, na qual a rede de dispositivos, interconetados através do protocolo BLE Mesh, envia os dados coletados pelos diferentes sensores e as respostas de *feedback* do usuário para o servidor via Wi-Fi, utilizando-se do protocolo MQTT, implementados no dispositivo gateway. O servidor, por sua vez, possui o Eclipse Mosquitto como *MQTT Broker*, o qual recebe os dados publicados pela rede, o Telegraf, responsável pela ponte entre o *MQTT Broker* e o banco de dados InfluxDB. Por fim, os dados são apresentados através de uma plataforma robusta de visualização, o Grafana, altamente configurável e que facilita a interpretação dos parâmetros coletados.

4 DESENVOLVIMENTO

Para o desenvolvimento do projeto, dividimos as tarefas entre 3 áreas: Hardware, Firmware e Software.

No **hardware** estará concentrado todo o desenvolvimento da eletrônica dos dispositivos nós da rede. O **firmware** comprehende todo o software embarcado no dispositivo, desde a comunicação com o hardware até a comunicação sem fio, entre os dispositivos da rede e do dispositivo *gateway* com a plataforma. O **software**, por fim, trata do desenvolvimento relacionado à plataforma onde os dados são armazenados e apresentados.

Por fim, foi desenvolvido um case mecânico simples, impresso em 3D, para proteger os componentes.

4.1 Hardware

A partir das especificações técnicas, foi elaborado um novo diagrama de blocos, conforme ilustrado na figura 8.

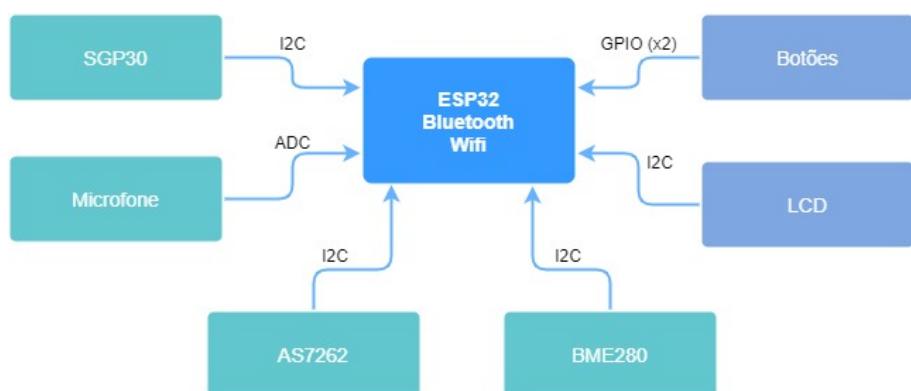


Figura 8: Diagrama de Blocos de Hardware do Protótipo

Quando possível, foram utilizados kits de desenvolvimento e módulos que nos permitisse uma validação mais rápida do hardware nessa fase inicial, permitindo avançar mais com o firmware e fazer testes em campo.

O DevKitC do ESP32 possui integrado um USB, através do qual é possível alimentar os demais subcircuitos da placa, não existindo a necessidade de uso de bateria nessa etapa.

4.1.1 Esquemático

Para o design do hardware, utilizamos o software de CAD de PCB *Altium Designer 20* [71], através da licença estudantil. Os arquivos estão disponíveis em [72].

A partir do diagrama de blocos, foi desenvolvido o esquema elétrico, ilustrado na figura 9.

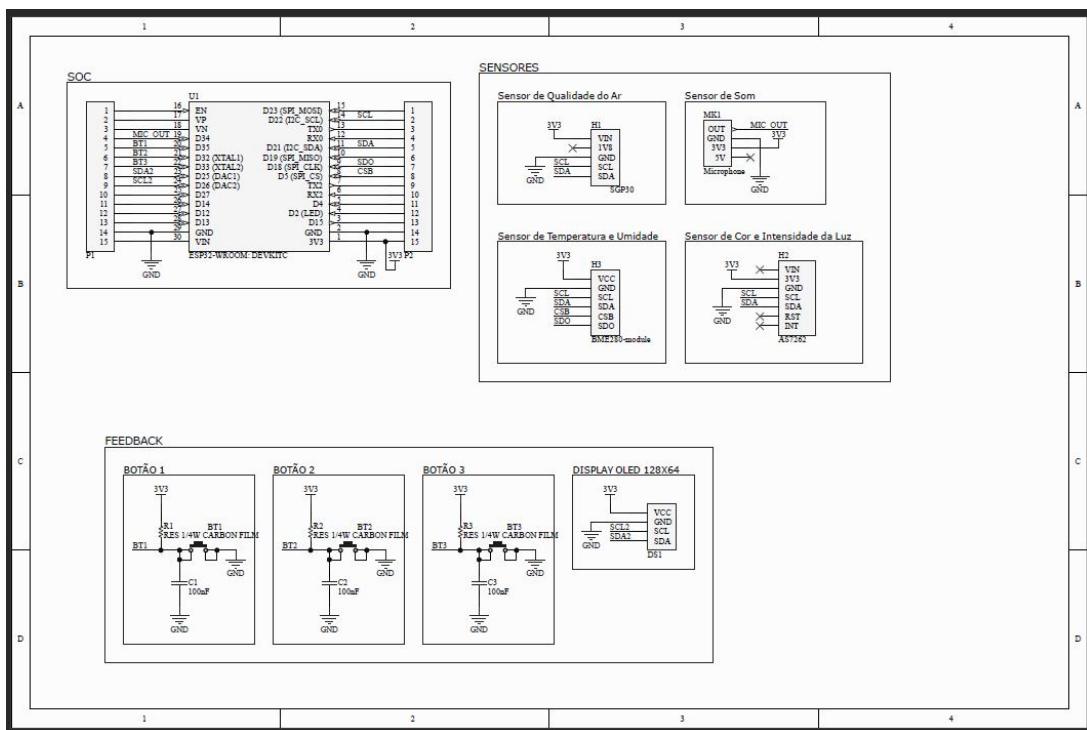


Figura 9: Esquemático do Protótipo

4.1.2 Printed Circuit Board (PCB)

A partir do esquema elétrico foi feito um desenho de PCB (*Printed Circuit Board*, ou Placa de Circuito Impresso), *Single Layer*, e dimensões finais 60x70 mm. As visões 2D e 3D da placa são mostradas na figura 10.

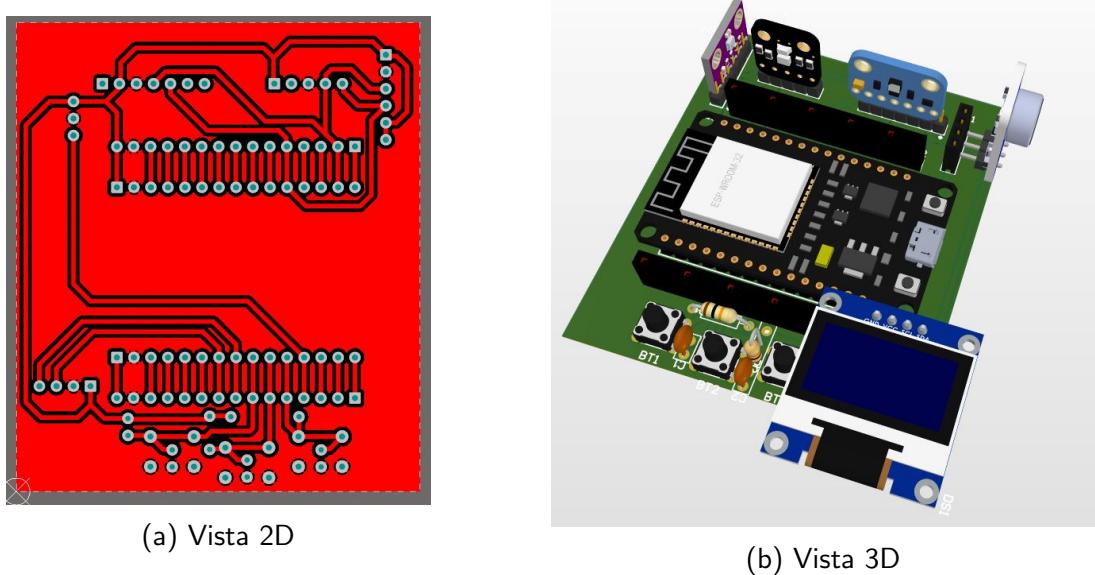


Figura 10: Design da PCB no Altium Designer

A placa mostrada na figura 11 foi fabricada em fibra por uma fresadora CNC e montada usando barras de pinos. O procedimento foi o mesmo para as placas dos demais dispositivos da rede.

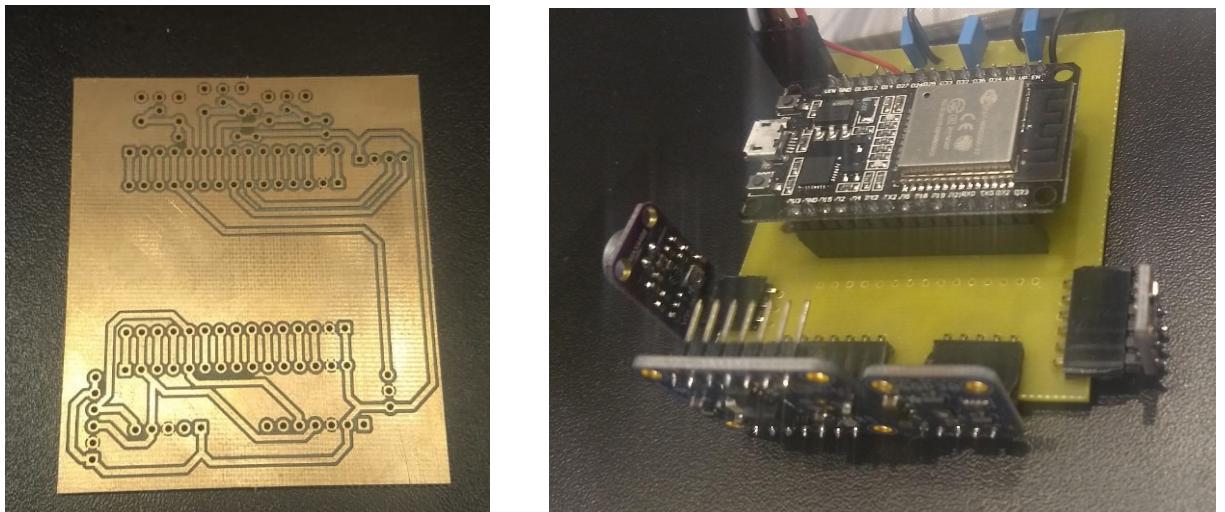


Figura 11: Placa fabricada e montada

4.2 Firmware

O desenvolvimento do firmware foi realizado utilizando o **ESP-IDF** (*Espressif IoT Development Framework*), disponibilizado pela Espressif [73] gratuitamente (sob Apache License).

O ESP-IDF é composto por API, ferramentas, componentes e fluxos de trabalho (*workflows*) que permitem o desenvolvimento de aplicações utilizando o ESP32.

Além disso, utiliza o FreeRTOS, que é o principal sistema operacional de tempo real (RTOS) para microcontroladores [74], de código aberto (*MIT open source license*), e com suporte para diversas famílias de microcontroladores.

Como o ESP32 possui 2 núcleos de processamento, o FreeRTOS usado no IDF é uma versão modificada da padrão, permitindo que exista multiprocessamento simétrico (SMP, *symmetric multiprocessing*).

Outras bibliotecas, proprietárias ou disponíveis em código aberto, foram utilizadas para a integração de todos os periféricos.

A camada mais alta do software nos dispositivos foi separada em três grupos: sensores, feedback e rede; mostrado no diagrama da figura 12.

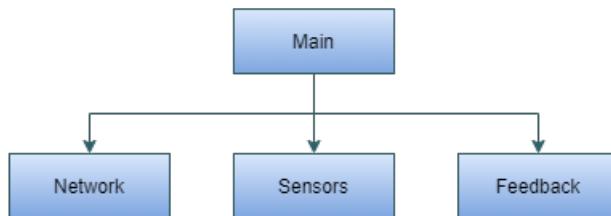


Figura 12: Camada alto nível do firmware dos dispositivos

A arquitetura completa do firmware pode ser visualizada no Anexo 2.

4.2.1 Sensores

Para os sensores, desenvolvemos bibliotecas baseadas no ESP-IDF e validamos o funcionamento dos mesmos individualmente realizando algumas coletas de dados do ambiente. O módulo de firmware para esta parte está separado conforme a figura 13, em que cada sensor possui a sua própria biblioteca, além de uma estrutura independente que controla o acesso ao barramento físico I²C e uma última responsável pela sincronização da execução do programa através de *timers*.

4.2.1.1 BME280

O BME280 é um sensor da Bosch que realiza medições de temperatura, umidade e pressão no ambiente. Possui 3 modos de operação: [4]

- *sleep*: baixo consumo de energia, registradores podem ser lidos, mas não realiza novas operações de medição.

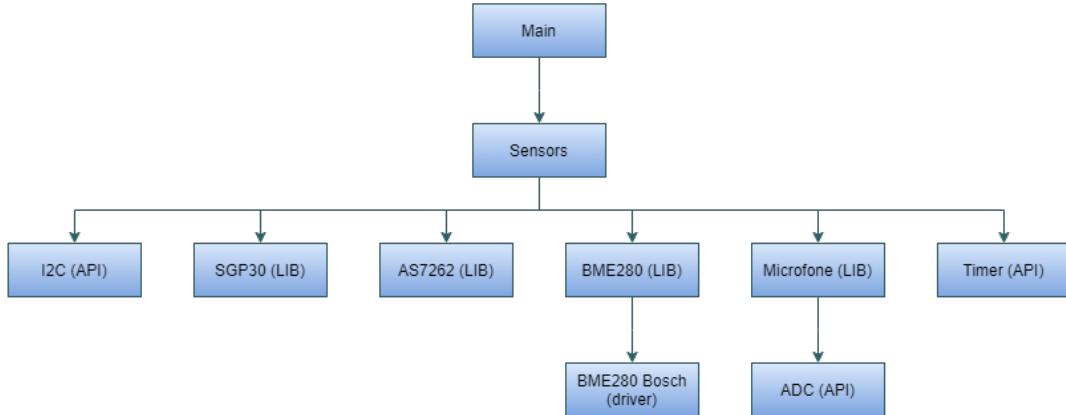


Figura 13: Arquitetura dos Sensores

- *forced*: realiza uma medição, salva os resultados, e volta ao modo *sleep*, como mostrado no diagrama da figura 14.
- *normal*: realiza medições periodicamente. O consumo durante o *standby* entre medições é maior que o consumo no modo *sleep*.

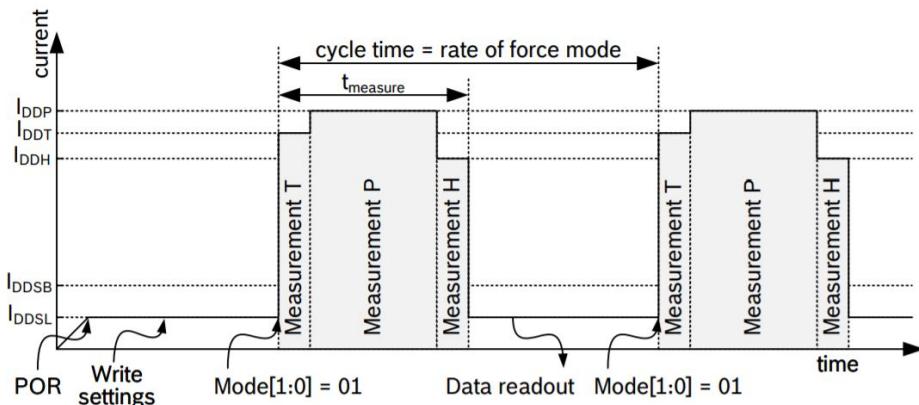


Figura 14: Diagrama de tempos do modo *forced*. Retirado de [4]

O *datasheet* recomenda que, para aplicações do tipo **monitoramento climático**, seja utilizado o modo *forced*, com cerca de 1 medição por minuto. Assim, o sensor consumirá uma corrente de aproximadamente $0.16\mu\text{A}$, sendo o modo de maior economia de energia.

A Bosch Sensortech possui um driver para o BME280, com código fonte em C e disponibilizado abertamente [75], que permite sua integração ao firmware em microcontroladores de qualquer fabricante. Utilizando o driver como base, desenvolvemos uma biblioteca para a utilização do BME280 no ambiente ESP-IDF, utilizando comunicação I²C, disponível em [76].

4.2.1.2 AS7262

Para a medição de parâmetros relacionados a luminosidade, foi utilizado o sensor AS7262, da AMS, que possui filtros internos capazes de separar a luz visível de acordo com 6 comprimentos de onda: 450, 500, 550, 570, 600 e 650 nm - que correspondem às cores violeta, azul, verde, amarela, laranja e verde, respectivamente. Cada um dos filtros é lido por conversores analógico-digital de 16 bits de resolução independentes que salvam os dados em registradores internos a cada conversão realizada. Possui 4 modos de operação:

- **Modo 0:** realiza medições contínuas, disponibilizando dados apenas referentes às cores violeta, azul, verde e amarela, sendo que os registradores de dados das cores vermelha e laranja ficarão com valor nulo.
- **Modo 1:** realiza medições contínuas, disponibilizando dados apenas referentes às cores verde, amarela, laranja e vermelha, sendo que os registradores de dados das cores violeta e azul ficarão com valor nulo.
- **Modo 2:** realiza medições contínuas para todos os canais.
- **Modo 3:** realiza medições para todos os canais, porém no modo *One-Shot*.

A diferença entre os modos contínuos e o modo *One-Shot* é que, no segundo, o usuário controla quando as medições deverão ser realizadas, enquanto no modo contínuo as medições são realizadas automaticamente a partir da sua inicialização.

Outra configuração disponível para que o usuário escolha é o ganho que é aplicado nos canais de leitura, podendo ser escolhido dentre os valores 1x, 3.7x, 16x e 64x.

A interface com o sensor pode ser feita através dos protocolos I²C, utilizada nesse projeto, ou UART, através de comandos AT. Foi desenvolvida uma biblioteca na linguagem C no formato adequado para ser utilizada no ESP-IDF com a implementação das principais funções do sensor. Ao acessar o barramento I²C, só é possível visualizar 4 endereços de registradores físicos, de acordo com a tabela 6.

O endereçamento dos registradores de aplicação, como os que armazenam os dados coletados e modos de operação se dá de modo virtual. O *datasheet* descreve a lógica de escrita e leitura nesses registradores virtuais, demonstradas nas figuras 15 e 16, respectivamente.

A lista com todos os endereços dos registradores virtuais pode ser encontrada em seu *datasheet* [7]. Com essas rotinas de leitura e escrita implementadas, basta realizar a configuração desejada para a aplicação e depois ler os valores captados pelo sensor.

Registrador	Descrição	Nota
Device Slave Address	Endereço I ² C de 8 bits	Endereço = 1001001x (0x49), onde x = 1 representa leitura e x = 0 representa escrita
STATUS	Registrador de STATUS	Endereço 0x00. Sinaliza se os registradores de Leitura e Escrita estão disponíveis
WRITE	Registrador de Escrita I ² C	Endereço 0x01. Valor de 8 bits a ser escrito pelo Mestre
READ	Registrador de Leitura I ² C	Endereço 0x02 Valor de 8 bits a ser lido pelo Mestre

Tabela 6: Registradores físicos do sensor AS7262. Retirado de [7] (traduzido).

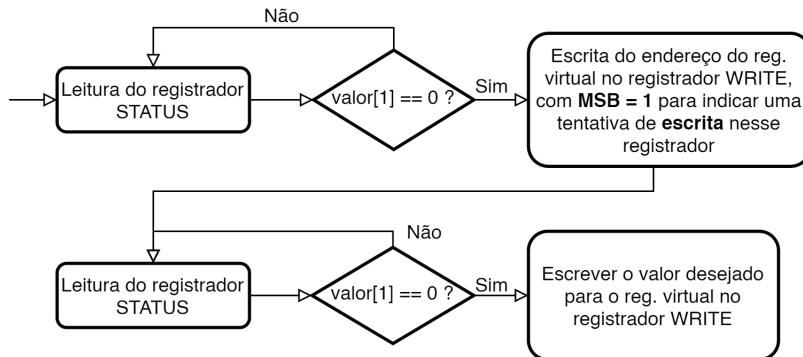


Figura 15: Fluxograma da rotina de escrita em um registrador virtual do sensor AS7262

A figura 17 mostra a lógica de aplicação implementada neste projeto. Primeiramente, é feita a operação de *Soft Reset* no sensor para reinicializá-lo, depois é feita a verificação do registrador que indica a versão do Hardware, que deve ser igual a 0x40. Feita essa verificação, escreve-se o valor 0b10 para inicializar a captação de valores no **Modo 2**: modo contínuo, referentes a todos os canais de medição. O valor de cada canal é armazenado em 4 registradores, totalizando 32 bits de valores em notação de ponto flutuante para cada canal. A etapa final da aplicação é realizar a leitura desses 24 registradores (4 por canal, 6 canais no total) e salvá-los em um vetor de 6 posições, para que possam ser manipulados posteriormente.

A biblioteca desenvolvida para a interação com o sensor pode ser acessada em [77].

4.2.1.3 SGP30

O sensor utilizado para medições dos parâmetros referentes à qualidade do ar foi o SGP30, da Sensirion. A partir de medições de concentração de hidrogênio e etanol no ar, um algoritmo interno ao sensor consegue estimar o total de componente orgânico volátil (TVOC), em partes

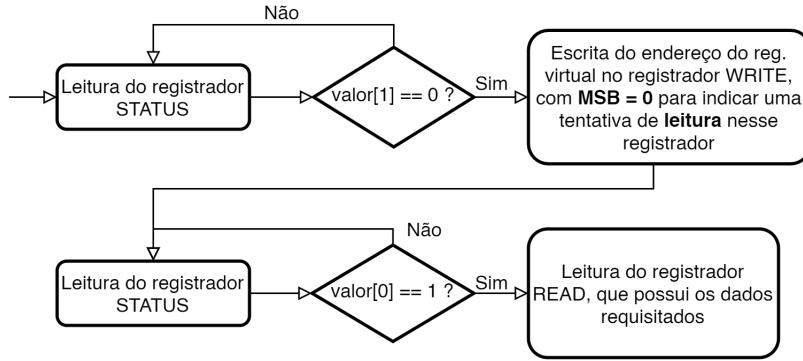


Figura 16: Fluxograma da rotina de leitura de um registrador virtual do sensor AS7262

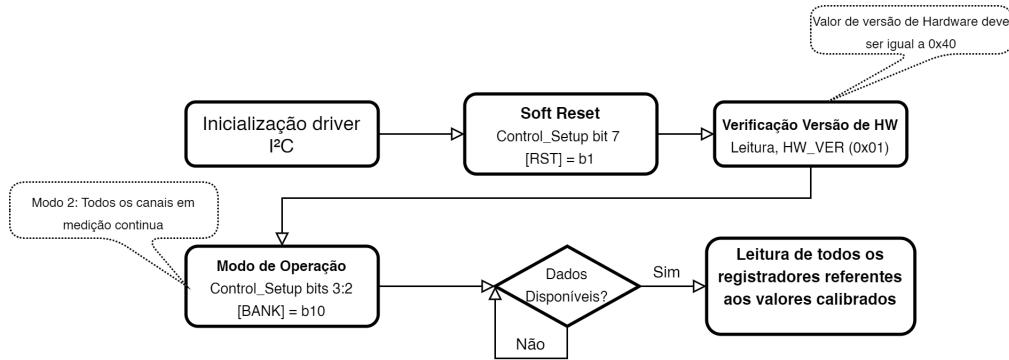


Figura 17: Fluxograma da lógica de firmware para operação do sensor AS7262

por bilhão (ppb), e o equivalente de CO_2 , em partes por milhão (ppm), presentes no ambiente.

A interface com esse sensor também é feita através do protocolo I²C, porém através de comandos pré-definidos, presentes nas páginas 9 e 10 do datasheet [63]. Cada comando é identificado por um valor de 16 bits e é responsável por realizar uma ação no sistema, como inicializar, realizar medições e ler valores fixados, como número de série.

Todos os dados enviados e recebidos pelo sensor contém ao final, além dos valores padrão, 8 bits de valores para verificação cíclica de redundância (do inglês *Cyclic Redundancy Check*, CRC), com a finalidade de validar os dados sendo transmitidos, utilizando o método de *checksum*, calculado com o seguinte polinômio: $CRC = 0h31(x_8 + x_5 + x_4 + 1)$ - em que x_i corresponde ao i-ésimo bit do valor utilizado para o cálculo. Os comandos de 16 bits tabelados já incluem os campos do CRC em sua composição, não sendo necessário calculá-los.

Portanto, para realizar interações com o sensor, o usuário deve realizar os passos ilustrados na figura 18, utilizando o comando desejado.

Para realizar a interação com esse sensor e obter os dados necessários para esse projeto, também desenvolvemos uma biblioteca nos mesmos moldes do ESP-IDF, em linguagem C, disponível em [78]. A rotina principal de coleta de dados é exemplificada na figura 19. O

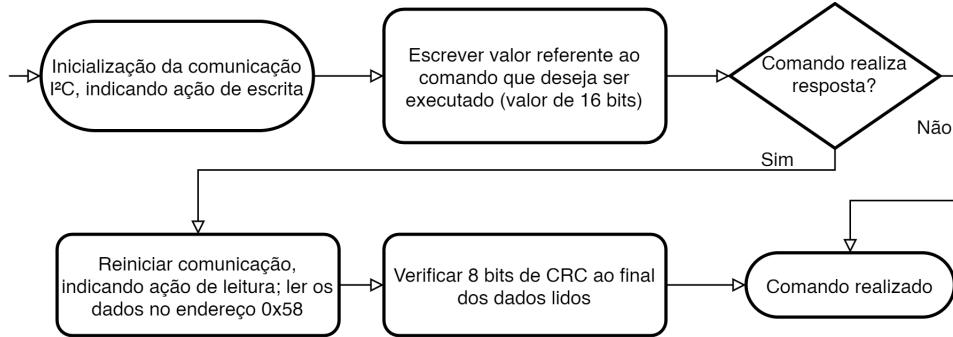


Figura 18: Fluxograma da rotina de execução de um comando para interface com o sensor SGP30

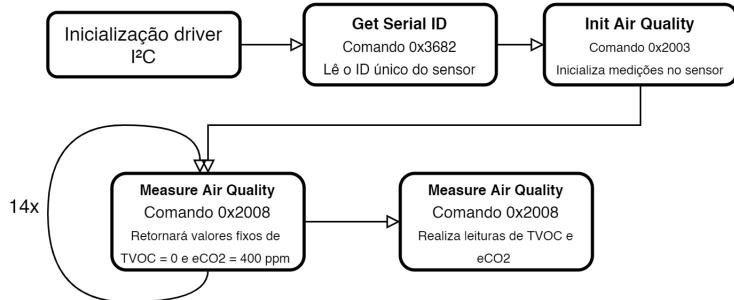


Figura 19: Lógica de execução de firmware para coleta de dados do sensor SGP30

datasheet indica que, para a correta inicialização do algoritmo que calcula as estimativas de *eCO*₂ e TVOC, é necessário que o usuário realize 14 medições espaçadas de 1 segundo, nas quais o sensor retorna valores fixos de *TVOC* = 0 e *eCO*₂ = 400 ppm. Após esse período, o sensor começa a responder com valores válidos referentes ao ambiente em questão.

Os valores retornados pelo comando *Measure Air Quality* são salvos em duas variáveis de 16 bits, a serem manipulados posteriormente.

4.2.1.4 Microfone

Para a medição de um indicador acústico, utilizamos um módulo com microfone de eletreto e um amplificador MAX4466, com ganho ajustável, da Adafruit [79]. A sua resposta é um sinal analógico entre 0 e 3,3V, correspondente ao sinal de áudio no ambiente, variando em torno do seu valor médio (1,65V).

A interface com o ESP32 deu-se através do conversor analógico digital (ADC) do próprio microcontrolador. Este ADC possui uma resolução máxima de 12 bits e uma amostragem máxima de 6kHz.

4.2.2 Feedback

Para a iteração com os usuários que fornecerão ao dispositivo seu feedback sobre o ambiente, foi elaborado um firmware cuja arquitetura é mostrada na figura 20.

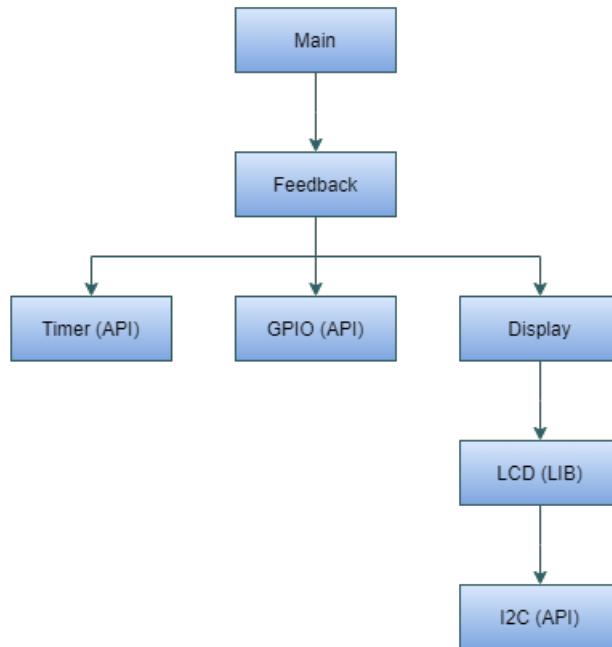


Figura 20: Arquitetura do Firmware de Coleta de Feedback

O firmware de feedback utiliza um *Timer* para controlar quando deve ocorrer uma coleta de feedback, a API de GPIO para controle dos botões e uma biblioteca para controle do display.

Como biblioteca para o *display* SSD1306, utilizamos um componente desenvolvido por Tara K [80] baseado no ESP-IDF [81]. Este utiliza uma porta I²C diferente da utilizada pelos sensores para que não haja uma competição desse recurso entre as *tasks* executadas em paralelo no sistema. Em *display.c* desenvolvemos as telas utilizadas durante a rotina de coleta de *feedback*, mostradas na figura 21.

A primeira pergunta apresentada é "Está confortável com a temperatura?", caso a resposta seja 'Não', é perguntado mais informações sobre o desconforto, "Muito quente?", para saber se a sensação térmica é de frio ou de calor. Esses dados são, na plataforma, associados à medição de temperatura.

Para o caso do som, apenas a pergunta "Está confortável com o ruído?" é feita, uma vez que o desconforto só poderia estar associado a uma alta intensidade.

Por fim, as perguntas associadas às medições da luz são "Está confortável com a luz?", e

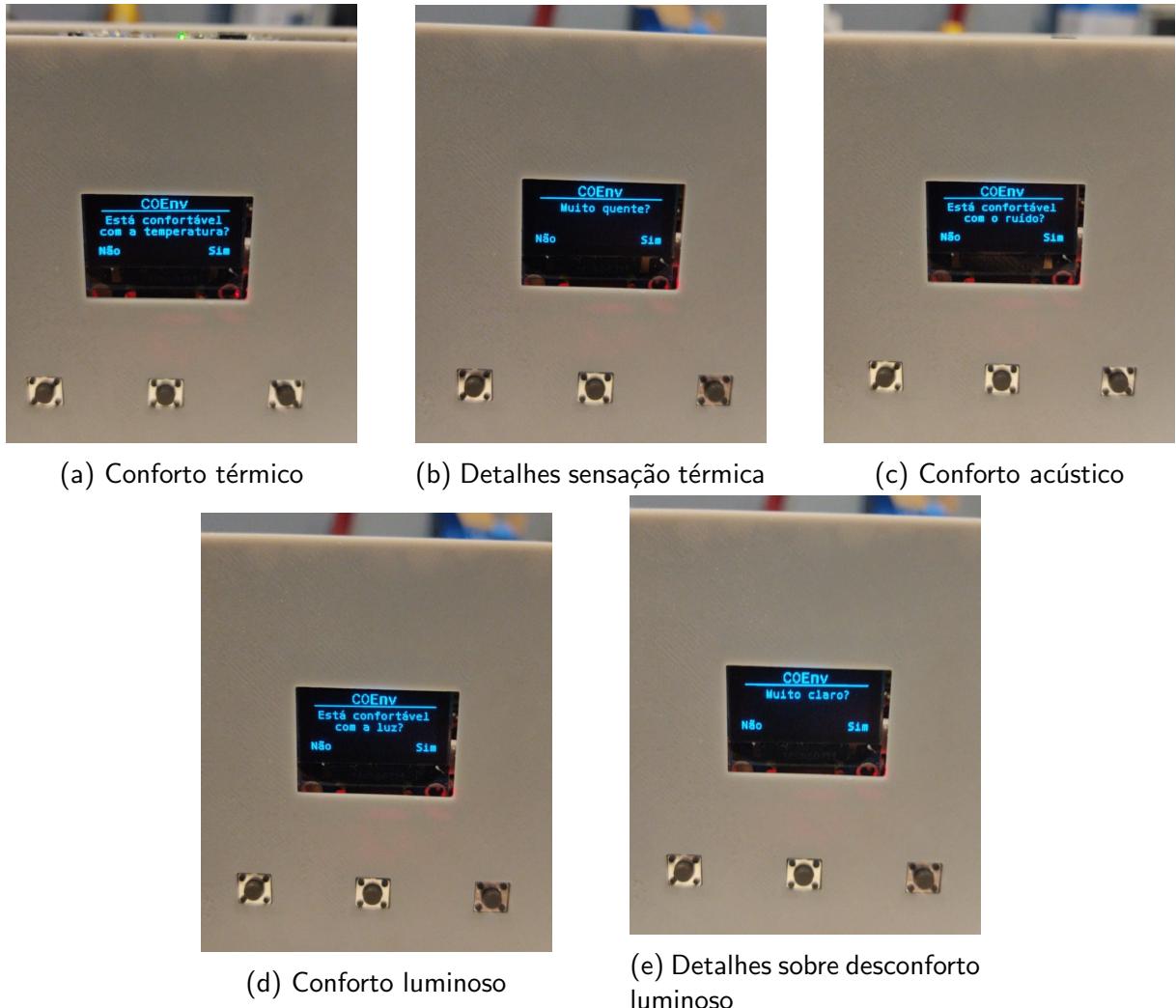


Figura 21: Telas com as perguntas do feedback

caso a resposta seja 'Não', "Muito claro?" para saber se o desconforto está relacionado a uma alta ou baixa intensidade da luz.

A tarefa de feedback inicialmente espera por um evento gerado por um *timer* ser colocado na fila de eventos adequada, identificando que deve ser realizada a coleta. A rotina de coleta segue, então, uma **máquina de estados** que foi implementada seguindo um *table-driven approach*, mostrada na figura 22. Os estados são determinados de acordo com as respostas às perguntas de conforto mostradas anteriormente na figura 21.

No início de uma coleta, é executado o estado inicial e a tarefa volta a esperar por um novo evento, que pode ser causado por um *timeout* ou um GPIO. Caso seja um GPIO, um novo estado é executado; caso seja um *timer* e o seu ID corresponder ao de *Timeout*, a coleta é encerrada. Como essa coleta é iniciada periodicamente, a fim de economizar energia quando

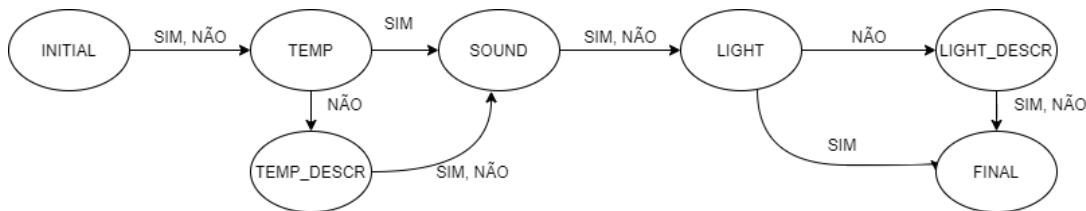


Figura 22: Máquina de Estados da coleta de Feedback

o usuário não estiver próximo ao dispositivo, ou não queira responder no momento, após 1 minuto sem resposta a coleta é encerrada automaticamente.

A mudança entre os estados acontece quando um dos botões laterais, correspondendo a respostas 'Não' e 'Sim', é apertado. De acordo com a resposta o futuro estado é definido.

No estado INITIAL mostra-se a primeira pergunta na tela, a estrutura com as respostas é zerada e um *timer* TIMEOUT é iniciado, a fim de desligar a tela e encerrar a rotina de feedback caso não haja resposta.

Em cada um dos estados intermediários, a resposta dos botões é armazenada, a tela é atualizada para uma nova pergunta, e o *timer* TIMEOUT é reiniciado. Quando ocorre um *timeout*, é executando o estado FINAL.

4.2.3 Conectividade

O módulo que trata da parte de rede do dispositivo foi separado em 2 partes, como exemplificado na figura 23: implementação dos modelos que definem a rede BLE Mesh, presente em todos os dispositivos, e a implementação da conexão com o MQTT Broker presente no servidor externo através do protocolo Wi-Fi, presente apenas no dispositivo gateway.

4.2.3.1 Bluetooth

Como citado em seções anteriores, os dispositivos desse projeto foram interconectados em uma rede BLE Mesh para que os dados coletados por eles sejam enviados pela rede para um dispositivo com função de *gateway*. Para realizar a implementação da rede, foi utilizada a biblioteca ESP-BLE-MESH, da Espressif, sendo essa baseada na biblioteca de Bluetooth Mesh do *Zephyr Project* [82], um projeto de RTOS mantido pela *Linux Foundation*, que disponibiliza toda a API necessária para a implementação de uma rede BLE Mesh customizada.

Por se tratar de uma aplicação com vários dados de sensores para serem enviados, foi criado um Modelo customizado que engloba todos os dados dos sensores que compõem um dispositivo, chamado de **Custom Sensor Model**, o qual foi implementado tanto o Modelo

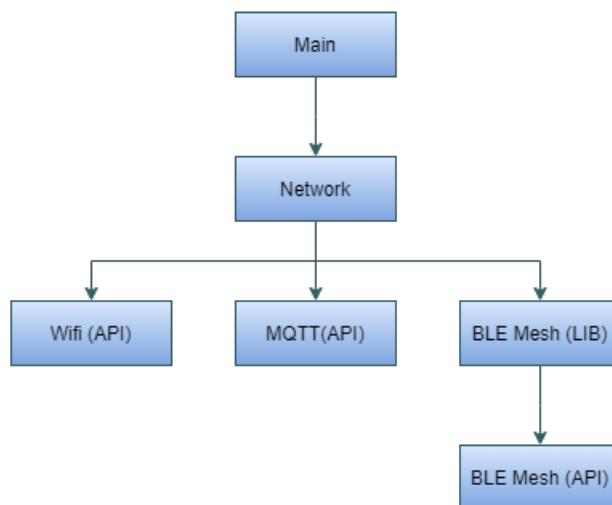


Figura 23: Arquitetura do módulo de Conectividade

Cliente quanto o Servidor. A estrutura de dados principal que define o estado do Modelo contém as variáveis listadas na figura 24.

<<struct>>	
model_sensor_data	
device_name:	char[6]
temperature:	float
pressure:	float
humidity:	float
tVOC:	uint16_t
eCO2:	uint16_t
noise_level:	uint16_t
red:	float
orange:	float
yellow:	float
green:	float
blue:	float
violet:	float

Figura 24: Estrutura de dados principal utilizada pelos modelos da rede BLE Mesh

A definição de um Modelo também requer que sejam definidas as mensagens utilizadas para interação, sendo elas:

- **GET**: mensagem utilizada para requisitar o estado atual do Modelo. Ao receber essa mensagem, o nó deve responder a requisição com uma mensagem de STATUS. O corpo

dessa mensagem pode ser enviado vazio, ou seja, apenas o endereço de destino precisa ser especificado.

- **SET**: mensagem utilizada para alterar o estado atual do Modelo Servidor por um agente externo (Cliente ou outro nó Servidor). Essa mensagem foi implementada sem que haja ACK, ou seja, não é enviada uma confirmação ao dispositivo transmissor que a mensagem foi recebida. Deve ser especificado o endereço de destino, seja um endereço de um elemento único ou um endereço de grupo da rede para publicar a mensagem, e o conteúdo deve conter os valores que serão atualizados no destino, de acordo com a estrutura anterior.
- **STATUS**: mensagem utilizada para notificar o estado atual do Modelo, normalmente enviada após receber uma mensagem do tipo GET. O conteúdo da mensagem é composto pela estrutura mostrada anteriormente.

Todos os dispositivos possuem as mesmas funcionalidades de coleta de dados de sensores, porém apenas um deles é responsável por também receber as medições de todos os outros, de modo que podemos separar os dispositivos em duas categorias diferentes: os dispositivos normais, chamados de *Sensor Nodes*, e o dispositivo central, chamado de *Gateway Node*. A composição lógica desses dois tipos de dispositivos está demonstrada na figura 25, sendo compostos por apenas um Elemento cada, com o *Configuration Server Model*, mandatório para esse tipo de dispositivo, já que é ele que exerce o papel de interlocutor com o dispositivo provisionador para que haja a atribuição das chaves NetKey e AppKey. Como pode ser observado na mesma imagem, os *Sensor Nodes* possuem o *Custom Sensor Client Model* e o dispositivo central possui o *Custom Sensor Server Model*.

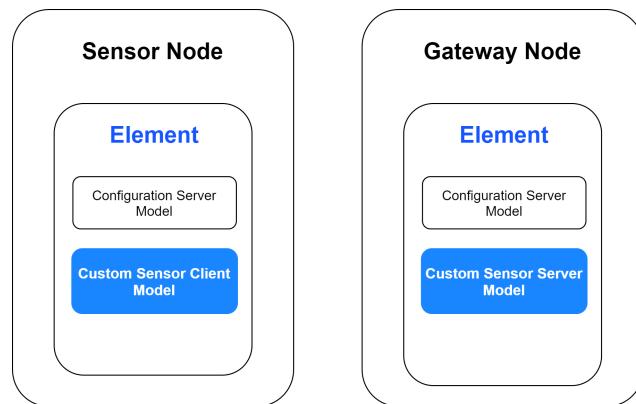


Figura 25: Composição lógica dos nós da rede BLE Mesh

Os dispositivos foram organizados de acordo com o modelo *Publish/Subscribe*, exemplificado na figura 26, no qual os *Sensor Nodes*, que possuem o Modelo Cliente, publicam

mensagens do tipo SET sempre no mesmo grupo, de endereço 0xC100, em que o *Gateway Node*, com o Modelo Servidor, está inscrito e recebe esses dados publicados, alterando seu estado. O conteúdo das mensagens enviadas é referente às medições dos sensores e, eventualmente, o resultado das perguntas da rotina de *feedback*, realizados por cada nó da rede. Ao receber uma mensagem nova, o dispositivo central atualiza seu estado e coloca essa mensagem numa fila sincronizada, que será lida e processada por outra parte do programa principal do dispositivo.

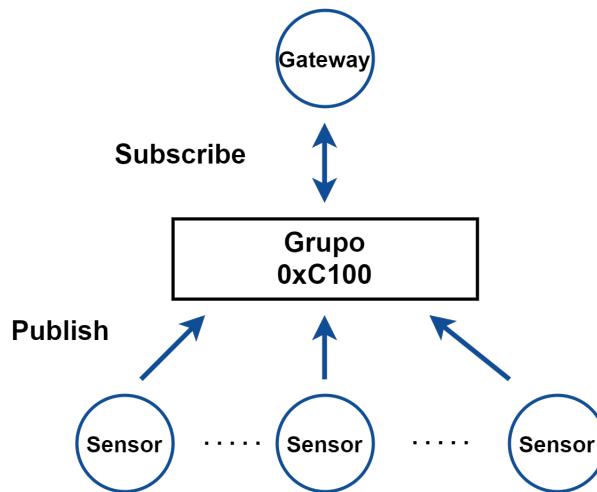


Figura 26: Arquitetura lógica da rede BLE Mesh

Essa fila sincronizada é a parte principal do dispositivo central, já que nela serão colocadas todas as mensagens recebidas dos nós da rede. Como citado na apresentação da estrutura de dados principal do Modelo, toda mensagem possui o nome do dispositivo que a enviou no início, antes dos dados dos sensores, sendo assim possível a identificação da mensagem em si após a sua retirada da fila e interpretação pelo programa principal.

4.2.3.2 Wi-Fi

A conexão do dispositivo gateway com a plataforma em nuvem é realizada utilizando o protocolo Wi-Fi. O ESP-IDF possui uma API pronta para ser utilizada para conectar o microcontrolador a um ponto de acesso Wi-Fi à escolha do usuário. Para isso, é necessário informar o SSID do ponto de acesso e a sua senha para conexão, armazenados em duas constantes no firmware. Definidas essas constantes, basta chamar a função de conexão fornecida pela API, que usará os valores definidos para realizar a conexão, bloqueando todo o programa até que a conexão seja estabelecida, obtendo endereços IPv4 e IPv6 (caso o ambiente de conexão seja capaz de fornecer os dois tipos de endereço para o dispositivo).

Com uma conexão com a Internet estabelecida, é necessário agora inicializar o dispositivo

como um cliente MQTT e conectá-lo com o *MQTT Broker* hospedado remotamente. Do mesmo modo, o ESP-IDF possui uma API que implementa as funcionalidades do protocolo MQTT utilizando conexão via TCP, para o qual é necessário informar o URL do *MQTT Broker* a ser conectado, além dos valores de usuário e senha para autenticação no mesmo. Esses valores são salvos em uma estrutura no firmware, utilizada por uma função para inicializar o dispositivo como um cliente e estabelecer a conexão, a qual exige que haja uma conexão com a Internet previamente estabelecida, como detalhado anteriormente. Se a conexão com o *MQTT Broker* for realizada com sucesso, um evento indicando essa conexão é recebido pelo dispositivo, estando pronto para enviar e receber mensagens.

O dispositivo gateway publica as mensagens em tópicos referentes a cada dispositivo da rede, ou seja, cada dispositivo possui um tópico referente aos seus dados, seguindo o formato `devices/device_name`, no qual *device_name* se refere ao nome do dispositivo definido na estrutura de dados do *Custom Sensor Model*, descrita na seção 4.2.3.1. O conteúdo das mensagens enviadas nos tópicos se refere às leituras dos diferentes sensores, tanto as recebidas via BLE Mesh quanto as leituras do próprio dispositivo gateway.

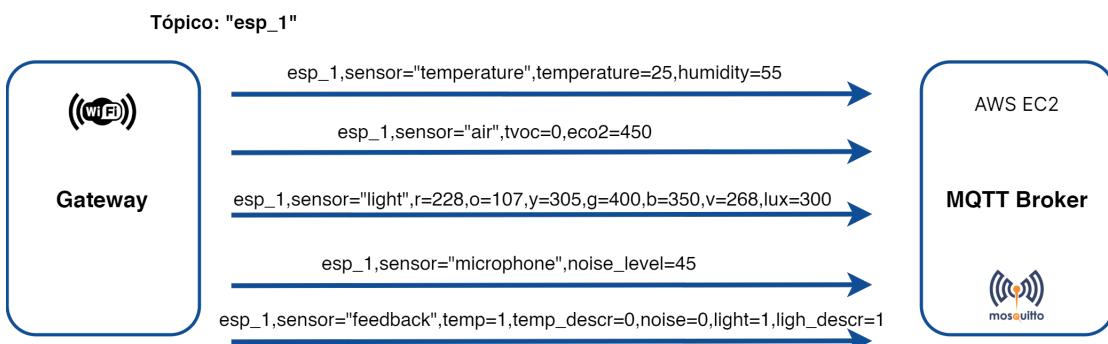


Figura 27: Exemplo de envio de um bloco de dados ao servidor

Durante uma rotina de envio de dados ao servidor, o dispositivo publica cinco mensagens em seguida no mesmo tópico, sendo essas mensagens atreladas sempre ao mesmo dispositivo. Cada uma das mensagens contém dados de um dos sensores, ou seja, teremos uma mensagem com valores de temperatura e umidade do BME280, uma com valores de TVOC e CO_2 do SGP30, uma com valores das 6 cores e a intensidade da luz do ambiente do AS7262 e uma com o valor de ruído ambiente captado pelo microfone, além de uma última mensagem com as respostas de *feedback* do usuário, exemplificados na figura 27. Essa rotina de envio acontece toda vez que o dispositivo *gateway* recebe dados via BLE Mesh ou quando uma medição dos seus próprios sensores é realizada. Todas essas mensagens seguem o formato a seguir, no qual **device_name** se refere ao nome do dispositivo ao qual essa medição pertence, **sensor_name** é o nome do sensor que realizou a medição e, por último, os valores de cada parâmetro lido:

```
device_name,sensor="sensor_name", value1=v1,value2=v2 ...
```

Este é o mesmo formato utilizado para inserir dados no InfluxDB, portanto enviá-los desta maneira para a plataforma facilita o seu tratamento posteriormente.

4.3 Software

O serviço de computação em nuvem escolhido para hospedar o banco de dados e a plataforma de visualização desse projeto foi o *Amazon Web Services* (AWS). O AWS possui dezenas de serviços para diferentes aplicações, dos quais foi escolhido o *Elastic Compute Cloud* (EC2), um serviço que permite que o usuário tenha uma máquina virtual com as características necessárias para hospedar as aplicações que desejar. O principal motivo que nos levou a escolher esse serviço de computação em nuvem foi o seu nível gratuito que, no caso do EC2, permite a configuração de uma máquina Linux do tipo *t2.micro* [83], a qual optamos pela distribuição Ubuntu 20.04, com 1 GiB de memória RAM e 8 GiB de armazenamento SSD e que pode ser utilizada pelo período de 1 ano sem cobranças, que é o necessário para a realização do protótipo do projeto. Ao configurar a máquina pelo portal do AWS EC2, é fornecida uma chave criptografada do tipo *Privacy-Enhanced Mail* (PEM) [84], utilizada para autenticar o acesso à máquina virtual através do protocolo SSH por um terminal externo, sendo essa a única forma de acessá-la.

Com acesso ao sistema da máquina virtual, podemos fazer a instalação e configuração dos serviços a serem utilizados. Todos eles possuem suporte integral ao sistema operacional escolhido, o que torna o processo de instalação muito simples, bastando apenas seguir as respectivas documentações oficiais.

4.3.1 Eclipse Mosquitto

A documentação de instalação do *Eclipse Mosquitto* para Ubuntu pode ser encontrada na sua página oficial [69], que é bem simples, resumida nos seguintes comandos:

```
$ sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
$ sudo apt update
$ sudo apt install mosquitto
```

Após a instalação, o serviço é inicializado automaticamente, executado na porta TCP 1883 do sistema. Inicialmente, o *broker* não possui nenhuma configuração de segurança, de

modo que qualquer cliente que saiba o endereço de IP público da máquina consegue publicar mensagens. Para adicionar uma camada de segurança, é necessário criar um arquivo de texto em que cada linha segue o formato *usuário:senha* que serão aceitos pelo *broker* como valores válidos para autenticação. Esse arquivo pode ser criptografado através do comando `mosquitto_passwd -U arquivo_de_autenticacao` para que as senhas não fiquem salvas em texto puro. O caminho para esse arquivo de autenticação deve ser informado no arquivo de configuração geral do serviço, presente em `/etc/mosquitto/conf.d/config.conf`. Feito isso, os clientes precisam informar o par *usuário:senha* para que consigam se conectar a esse *broker*.

A comunicação é feita sobre o protocolo TCP sem nenhuma criptografia nas mensagens trocadas entre os dispositivos, de forma que as credenciais citadas anteriormente sejam transmitidas em texto e possam ser interceptadas e lidas por terceiros. Uma forma de melhorar ainda mais a segurança da conexão seria adicionar uma forma de criptografia na camada de transporte com o uso de SSL, que pode ser utilizada em algum projeto futuro.

4.3.2 InfluxDB

Do mesmo modo que a aplicação anterior, a instalação do banco de dados InfluxDB ocorre de forma direta e simplificada, detalhada em sua página oficial [85] e, após a instalação, a aplicação será executada por padrão na porta TCP 8086 do sistema.

A interface de linha de comando da aplicação pode ser acessada através do comando `influx`, onde é possível realizar comandos e executar *queries*, de acordo com a documentação oficial. A linguagem de interação com o banco de dados é chamada de *InfluxQL*, muito parecida com a linguagem SQL. A primeira coisa a ser feita é criar um novo banco de dados utilizando o comando `CREATE DATABASE prototipo_v1`, utilizado para salvar os dados desse projeto. Para criar usuários para que outras aplicações consigam acessar o banco de dados, usamos o comando `CREATE USER <username> WITH PASSWORD '<password>'` e criaremos dois usuários de acesso: 'telegraf' e 'grafana', sendo que o primeiro possui acesso total ao banco de dados (leitura e escrita), já que ele será responsável por escrever os dados recebidos da rede de sensores, e o segundo com permissões apenas de leitura.

Todos os pontos inseridos no banco de dados possuem um *timestamp* atrelado a eles, no formato UTC definido no RFC3339 [86], que indica o momento em que o ponto foi salvo. Os valores que se referem aos dados armazenados de fato são chamados de **Field Values**, que podem ser números inteiros, valores booleanos, *strings* ou de ponto flutuante. Esses dados podem ser identificados através de outros valores, chamados de **Field Keys**, que servem para

identificar o dado salvo, ou seja, age como o nome da variável, enquanto o valor anterior age como o conteúdo de fato da variável salva.

Outro par de campos que servem para descrever o dado salvo é **Tag Keys** e **Tag Values**, ambos do tipo *string*, muito úteis quando é preciso buscar e agrupar dados. Todos esses campos citados são salvos dentro de um grupo chamado de **Measurement**, sendo que o nome desse grupo descreve todos os dados que estão contidos dentro dele.

Finalmente, uma **série temporal** é o conjunto de pontos que conta com os mesmos valores de *Measurement*, *Tag* e *Field Key*, com seus valores únicos identificados pelo *Field Value* em conjunto com o *timestamp*.

No contexto desse projeto, temos *Measurements* que separam cada dispositivo único na rede, como mostrado na tabela 7, *Tag Key* como sendo o nome de um dos quatro sensores únicos em um dispositivo ou *feedback* e *Field Key* os nomes dos diferentes parâmetros lidos, conforme a tabela 8.

Dispositivo	Measurement
Gateway	esp_1
Nó 1	esp_2
Nó 2	esp_3

Tabela 7: *Measurements* presentes no banco de dados

Sensor	Tag Key	Field Key
BME280	Temperature	temperature, humidity
SGP30	Air	eco2, tvoc
AS7262	Color	v, b, g, y, o, r, lux
Microfone	Noise	noise_level

Tabela 8: *Tag Keys* e *Field Keys* que identificam cada tipo de sensor e dados coletados

Ao final da seção 4.2.3.2, foi explicado que os dados eram publicados seguindo um formato que facilitaria o tratamento e sua inserção no banco de dados. De fato, esse formato segue exatamente o que foi explicado agora, com cada campo correspondendo a sua respectiva função conforme as atribuições anteriores. A separação dos campos ocorre da seguinte forma:

$$\underbrace{\text{device_name}}_{\text{Measurement}}, \underbrace{\text{sensor} = \text{"sensor_name"},}_{\text{Tag Set}} \underbrace{\text{value1} = v1, \text{value2} = v2 \dots}_{\text{Field Set}}$$

4.3.3 Telegraf

O Telegraf é a aplicação responsável por conectar o Mosquitto com o InfluxDB, se inscrevendo nos tópicos de dados enviados pelos dispositivos da rede e inserindo-os no banco de

dados. Como mencionado anteriormente, essa aplicação também é desenvolvida pela *Influx-data*, mesma empresa que mantém o InfluxDB, de modo que suas documentações são muito parecidas [87]. Logo, sua instalação também ocorre de maneira direta no sistema operacional escolhido através de poucos comandos.

Simplificadamente, esse programa tem como objetivo ligar diferentes fontes de dados a diferentes destinos através de plugins previamente desenvolvidos, bastando o usuário escolher quais utilizar. Esses plugins são separados em duas categorias principais, plugins de entrada (*input plugins*) e de saída (*output plugins*). No nosso caso, utilizamos o **MQTT Consumer Input Plugin** como plugin de entrada e o **InfluxDB v1.x Output Plugin** como saída. A configuração e escolha dos plugins utilizados é feita através de um arquivo de configuração presente no caminho `/etc/telegraf/telegraf.conf`, no qual a atribuição de valores às diferentes configurações de cada plugin é feita, seguindo as respectivas documentações.

O primeiro plugin tem a função de se conectar ao MQTT Broker e se inscrever nos tópicos em que são escritos os dados provenientes da rede de dispositivos, recebendo assim as mensagens publicadas referentes às diferentes medições de sensores e, como definido na seção 4.2.3.2, os tópicos seguem o formato `devices/device_name`, com conteúdo das mensagens no formato definido pelo InfluxDB. Dessa forma, definimos as configurações desse plugin:

```
[[ inputs.mqtt_consumer ]]
    servers = ["tcp://localhost:1883"]
    topics = [ "devices/#" ]
    data_format = "influx"

    username = <username_broker>
    password = <senha_broker>
```

A aplicação Mosquitto é executada no endereço `tcp://localhost:1883`, no qual o Telegraf se conecta utilizando o usuário e senha aqui configurados para autenticação, se inscrevendo nos tópicos que atendem o formato passado em *topics*, esperando mensagens no formato *influx*.

Já o segundo plugin tem a função de inserir todos os dados coletados pelo Telegraf em um banco de dados no InfluxDB. Precisamos apenas configurar a conexão do plugin com o banco de dados desejado, porque os dados coletados já estão no formato requisitado para a inserção. A configuração desse plugin ocorre no mesmo arquivo do anterior, adicionando os seguintes campos:

```
[[ outputs.influxdb ]]
```

```

urls = ["http://localhost:8086"]
database = "prototipo_v1"
username = "telegraf"
password = <senha_influxdb>

namepass = ["esp*"]

```

O Telegraf se conecta com o InfluxDB através do endereço `http://localhost:8086`, se autenticando no banco de dados indicado na variável `database` utilizando o par usuário e senha fornecidos. Utilizamos a variável `namepass`, que verifica quais dados possuem a expressão "esp" em seu início, com o propósito filtrar os dados que serão inseridos de fato por esse plugin no banco de dados. Essa expressão está presente nas mensagens recebidas, pois os nomes dos dispositivos (`device_name`) definidos no projeto são `esp_1`, `esp_2` e `esp_3`, já que temos apenas 3 dispositivos protótipos, que indicará também o `measurement` inserido no banco de dados.

Feitas essas configurações, as mensagens publicadas pelos dispositivos nos tópicos citados são encaminhadas para o Telegraf, que faz sua inserção no banco de dados. A partir desse ponto, os dados estão prontos para serem acessados e analisados.

4.3.4 Grafana

A plataforma de visualização de dados também pode ser instalada facilmente através de poucos comandos seguindo a sua própria documentação, presente em [88]. Logo após a instalação, a aplicação é executada na porta 3000 do sistema por padrão. Ao acessá-la pela primeira vez através de um navegador web utilizando o IP da máquina e a porta 3000, o usuário é levado a uma página de `login` para a definição da senha de acesso para o usuário administrador que, depois de definida, pode ser utilizada para acessar o ambiente do Grafana.

A aplicação possui um menu lateral onde é possível acessar parâmetros de configuração geral. A primeira configuração realizada é a conexão com o banco de dados, escolhendo a opção `Configuration → Data Sources` no canto esquerdo, que leva a uma página onde pode ser escolhido o tipo da fonte de dados a ser conectado a partir de uma lista com inúmeras conexões possíveis, onde selecionamos o InfluxDB, e depois é necessário preencher as informações de conexão, análogas às que foram utilizadas pelo Telegraf, como ilustrado na figura 28.

O Grafana possui diferentes opções de painéis como tabelas, gráficos de linhas e de barras, destinados a mostrar dados acessados através de consultas específicas ao banco de dados. Essas consultas, normalmente, retornam partes de uma das séries temporais referentes a medições de

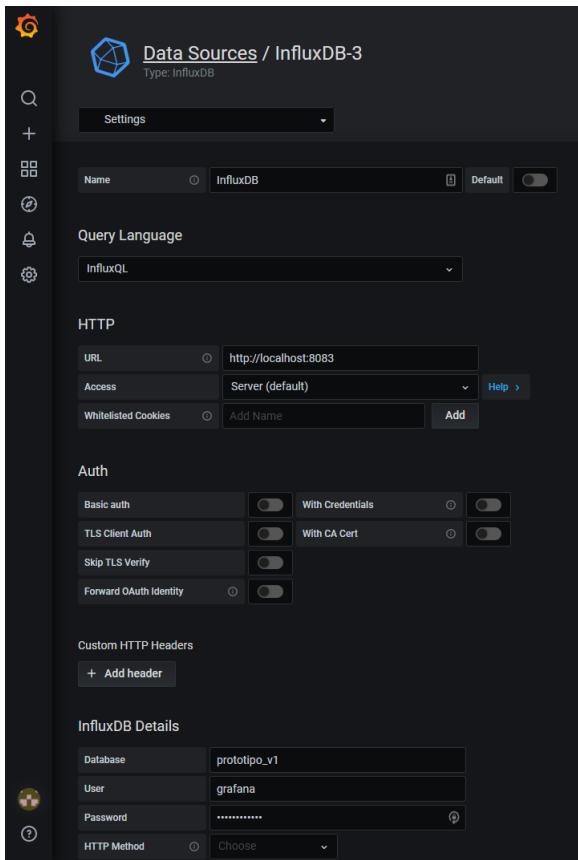


Figura 28: Conexão do Grafana com a fonte de dados InfluxDB

sensores em um período de tempo específico, determinado por uma variável presente no canto superior direito de todas as páginas do *dashboard*, onde o usuário pode selecionar a faixa de tempo desejada. A figura 29 ilustra como uma consulta pode ser configurada dentro de um painel, exemplificando o caso de acesso à série temporal referente às medições de temperatura do dispositivo *esp_1* para serem visualizados em um gráfico de linha.

```

FROM default WHERE sensor = "Temperature"
SELECT field(temperature)
GROUP BY Time series
    
```

Figura 29: Exemplo de consulta feita em um painel do Grafana

A configuração de painéis gráficos também permite que sejam estabelecidos valores limiares para as séries temporais monitoradas e, a partir desses limiares, configurar alertas que podem ser notificados em diferentes serviços externos, dentre eles alguns serviços de comunicação como Slack, Telegram, Microsoft Teams e email.

Os valores de limiar foram definidos com base nos níveis recomendados para cada indicador de qualidade e conforto medidos, discutidos na seção 2.1. O sistema de alertas foi configurado para avaliar a cada 10 minutos os últimos dados coletados e, se estiverem fora dos níveis recomendados, enviam notificações para um canal em um servidor do aplicativo Slack, para demonstrar um possível caso de uso em um escritório de uma empresa que faz uso desse meio de comunicação.

4.3.5 Servidor

Com objetivo de facilitar o acesso à plataforma de visualização, registramos o domínio coenv.tech gratuitamente por 1 ano através do programa dedicado a estudantes *Github Education* [89]. Como serviço de DNS, foi utilizado o *Cloudflare* [90], redirecionando acessos ao domínio adquirido ao IP público da máquina virtual fornecido pela AWS.

Cada aplicação citada até aqui roda em um porta diferente no sistema, porém acessá-las externamente através dessas portas não é seguro nem prático. Dado isso, utilizamos o NGINX, uma aplicação que atua como *reverse proxy*, ou seja, todas as requisições feitas ao nosso sistema passarão por essa aplicação, a qual redireciona as requisições para os serviços adequados. Deste modo, podemos deixar apenas as portas 80 (HTTP), 443 (HTTPS) e 1883 (MQTT/TCP) liberadas para aceitarem conexões externas e que serão monitoradas pelo NGINX.

As configurações de redirecionamento são definidas por regras presentes em um arquivo específico, onde são explicitados quais os destinos que devem receber as requisições, dependendo da URL de origem. Os valores escolhidos para os pares subdomínio - aplicação estão mostrados na tabela 9. Todos os subdomínios citados também foram adicionados às configurações do *Cloudflare* para que a tradução dos nomes seja feita corretamente.

Subdomínio	Aplicação Local
mosquitto.coenv.tech	Eclipse Mosquitto porta 1883
dashboard.coenv.tech	Grafana porta 3000
coenv.tech	Página HTML estática

Tabela 9: Tabela de redirecionamento configurada para o NGINX

Visando adicionar uma camada maior de segurança, utilizamos o *certbot*, um programa desenvolvido pela *Electronic Frontier Foundation* que facilita a adoção de certificados SSL emitidos gratuitamente pela *Let's Encrypt* [91] [92], para adicionar um certificado no nosso

servidor e permitir conexões seguras via HTTPS.

Portanto, temos as seguintes formas de acessar os serviços disponibilizados pelo servidor:

- Acesso ao Mosquitto MQTT Broker: <mqtt://mosquitto.coenv.tech/>
- Acesso à plataforma de visualização de dados: <https://dashboard.coenv.tech/>
- Acesso à página web com informações sobre o projeto: <https://coenv.tech/>

A figura 30 mostra o funcionamento do servidor recebendo as diferentes requisições e reecaminhando o tráfego para as aplicações correspondentes.

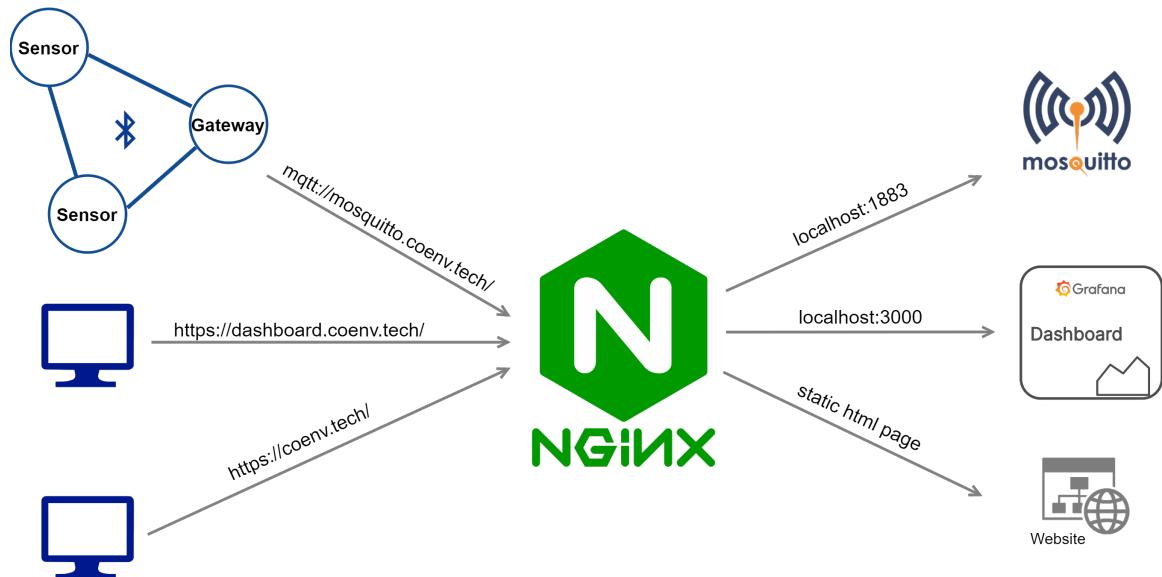


Figura 30: Arquitetura do servidor em nuvem

4.4 Mecânica

Foi projetado um invólucro mecânico para proteger a eletrônica dos dispositivos, modelado conforme a figura 31 e impresso em 3D.

Ele é composto por duas peças, uma base, onde fica fixada a placa, e uma tampa onde ficam fixados a tela OLED e os botões. As laterais abertas permitem que sejam feitos ajustes dos módulos com sensores sem afetar o funcionamento destes.

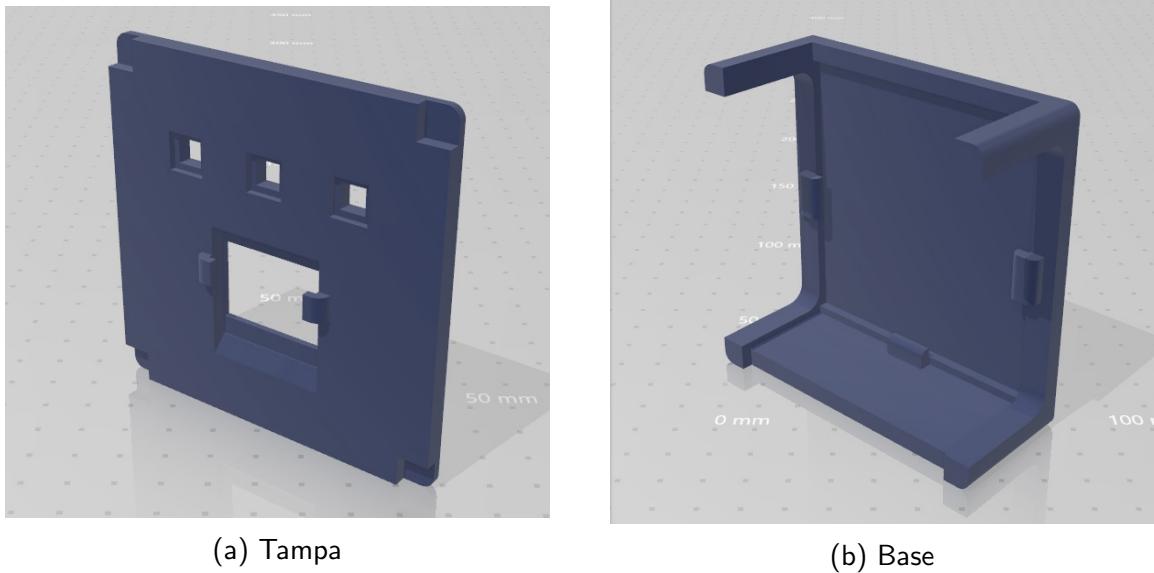


Figura 31: Case Mecânico

4.5 Custos

O software de CAD de PCB *Altium Designer 20* [71], utilizado para o design do hardware, é um software pago, que foi utilizado com uma licença estudantil no desenvolvimento desse projeto. Possui 30 dias de teste grátis, com um custo anual não divulgado pela empresa, apenas sob consulta.

A tabela 10 mostra os custos dos componentes utilizados no hardware dos 3 dispositivos.

Hardware			
Componente	Quantidade	Custo Unit.	Total
ESP32 DevKitC	3	R\$ 45,00	R\$ 135,00
BME280	3	R\$ 35,30	R\$ 105,90
SGP30	3	US\$ 19,95	R\$ 909,78*
AS7262	3	US\$ 19,95	R\$ 909,78*
Microfone MAX4466	3	R\$ 33,06	R\$ 99,18
LCD Oled SSD1306	3	R\$ 28,88	R\$ 86,64
Botões Push-Button	9	R\$ 0,37	R\$ 3,33
Capacitor 220nF	9	R\$ 0,10	R\$ 0,90
Resistor 1kOhm	9	R\$ 0,10	R\$ 0,90
Cabos USB micro	3	R\$ 9,90	R\$ 29,70
Placas de fibra de vidro	3	R\$ 4,00	R\$ 12,00
Total			R\$ 2.293,01

Tabela 10: Tabela de Custos do Hardware

(*) Valores convertidos para reais somados aos impostos de importação. Cotação utilizada: US\$ 1,00 → R\$ 5,59.

A fabricação das PCBs foi feita no PSI, em uma fresa CNC, podendo ter um custo adicional se o processo for terceirizado. Para a montagem final dos dispositivos, foram utilizados cabos com um custo aproximado de **R\$20**.

Os cases mecânicos foram impressos em PLA por uma impressora 3D Ender 3. Utilizando o software Ultimaker Cura [93] vemos que foram utilizados cerca de 49g de filamento PLA por case, totalizando um custo aproximado de **R\$18** para os três dispositivos (considerando em média R\$120,00/kg de filamento PLA).

Assim como a fresagem da PCB, o trabalho de impressão pode ter um custo adicional caso seja terceirizado. Gastos de energia elétrica não foram considerados.

5 VERIFICAÇÃO DO PROJETO

5.1 Validação dos Subsistemas

Para os sensores e o display, ao desenvolver cada uma das bibliotecas citadas na seção 4.2, também desenvolvemos exemplos para a validação de cada um individualmente.

Utilizando a biblioteca de logging do ESP-IDF [94], no modo Info, pudemos analisar os dados coletados pelos sensores, assim como os cálculos realizados, através do **IDF monitor**, uma ferramenta do ESP-IDF que funciona como um terminal serial, comunicando-se com o ESP32.

5.1.1 Intensidade da Luz - AS7262

Para calcular a **intensidade** total da luz no ambiente fizemos alguns testes de calibração no LME, com o auxílio do técnico Carlos Ramos. Utilizando uma célula solar padrão com resposta conhecida, pudemos medir a intensidade real de uma luz branca incidindo sobre ela. Colocando em seguida o sensor AS7262, na mesma posição da célula, e incidindo a mesma luz cuja intensidade é agora conhecida, pudemos realizar medições mais precisas.

Com leituras para diferentes intensidades, somamos os dados lidos pelo sensor para cada espectro, uma vez que o sensor tem como resposta 6 valores proporcionais à energia por unidade de área da luz incidente em cada um de seus filtros internos. Dividimos esse valor pelo tempo de integração configurado, cujo funcionamento foi explicado na seção 4.2.1.2, obtendo assim um valor proporcional à intensidade da luz.

Com esses dados, mostrados na tabela 11, foi possível encontrar uma constante calibrada α para a conversão da leitura.

		Intensidade da Luz ($\mu W/cm^2$)				
Tempo de Integração	Espectro	11000	5000	20000	25000	30000
84ms (0x1E)	Violet	6003	8167	10877	13143	16216
	Blue	2451	3368	4512	5484	6822
	Green	11271	15485	20939	25654	30721
	Yellow	13911	19084	25809	30721	30721
	Orange	3404	4698	6732	7801	9853
	Red	1932	2618	3608	4279	5333
	Soma	38972	53420	72477	87082	99666
168ms (0x3C)	Violet	12033	16211	21585	26485	32478
	Blue	4896	6577	8868	11031	13647
	Green	22588	30874	41505	52153	61441
	Yellow	27844	38091	51252	61441	61441
	Orange	6662	9184	12357	15725	19672
	Red	3790	5146	6813	8689	10637
	Soma	77813	106083	142380	175524	199316

Tabela 11: Tabela de dados coletados na calibração do sensor AS7262

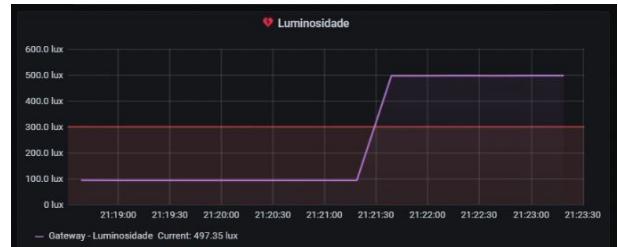
Encontramos a relação entre a soma lida pelo sensor e a intensidade da luz incidente como a constante $\alpha = \frac{Soma}{\Delta t * I} \approx 41,67$. Assim, podemos calcular os valores da intensidade lida no ambinete $I[\mu W/cm^2] = \frac{Soma}{\Delta t * \alpha}$. Como 1 lux equivale a $1,46 * 10^{-7}W/cm^2$, temos:

$$I[\text{lux}] = 0,146 * \frac{Soma}{\Delta t * \alpha}$$

Com esses valores de calibração, fizemos um teste mudando a luz incidente sob o sensor para validar a biblioteca completa. No dashboard é possível ver um aumento da intensidade da luz quando um abajur é ligado, em torno das 21h22, como mostra a figura 32.



(a) Arranjo do teste de luz



(b) Gráfico temporal das leituras

Figura 32: Testes de intensidade da luz

5.1.2 Qualidade do Ar - SGP30

Para verificar o funcionamento do sensor de qualidade do ar, fizemos dois testes, monitorando os valores coletados pelo sensor através do gráfico temporal do dashboard.

O primeiro teste foi substância em suspensão no ar. Após a calibração do sensor e uma estabilização das leituras, espirramos um ar logo acima do sensor SGP30 um odorizante de ambiente, como o mostrado na figura 34. No dashboard em 21h07 é possível ver o aumento na concentração de TVOC e de eCO2. Esse aumento é esperado pois o odorizante é composto de substâncias orgânicas voláteis e alguns gases comprimidos que geram CO2. Em seguida foi ligado um ventilador de mesa ao lado do sensor. Com uma maior circulação do ar, vimos uma diminuição da concentração de TVOC e eCO2 no ambiente.

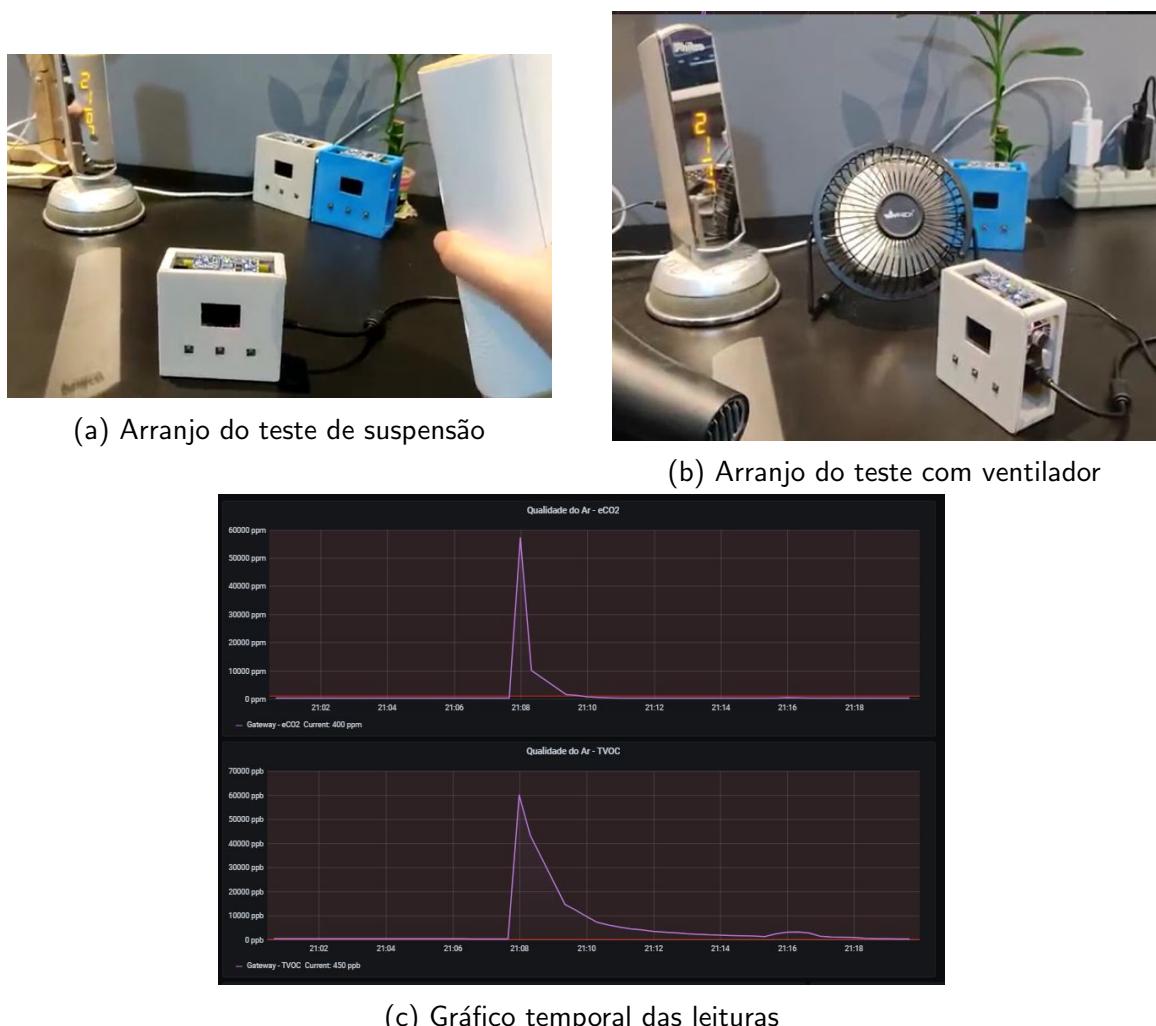


Figura 33: Testes de qualidade do ar

5.1.3 Temperatura e Umidade - BME280

Com um secador de cabelo, jogamos ar quente e seco sobre o sensor BME280. Pelo dashboard foi possível visualizar um aumento gradativo da temperatura e uma redução da umidade relativa do ar, após 21h15.

Com o mesmo teste feito com o ventilador 33b, em torno de 21h10, foi possível ver que o fluxo de ar influencia também na temperatura.

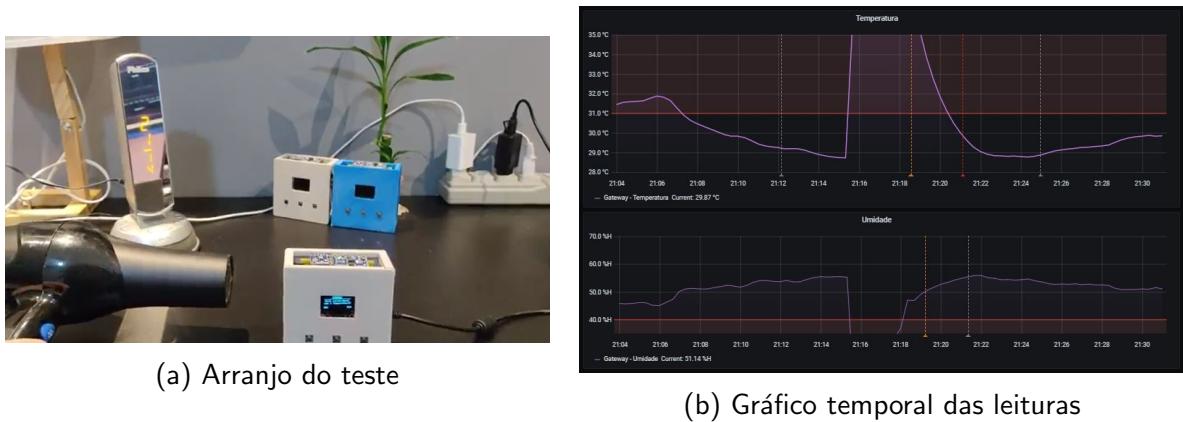


Figura 34: Testes de temperatura e umidade

5.2 Validação do Dispositivo

Foi fabricado um case mecânico em PLA utilizando uma impressora 3D e feita a montagem de um primeiro protótipo, como mostrado na figura 35, realizando os ajustes necessários no projeto para o encaixe de todas as peças.

Para validação de cada subcírcuito presente na PCB, utilizamos os mesmos exemplos citados para validação dos sensores e do display, até então testados usando uma protoboard de testes.

Com o firmware completo desenvolvido, fizemos testes com esse dispositivo, utilizando a mesma biblioteca de Log e o IDF monitor para visualizar as leituras. Os dados dos quatro sensores operando em paralelo podem ser vistos na figura 36.

Cada sensor tem sua coleta sendo feita em uma task individual, isto é, a leitura e os cálculos de um sensor é feito em paralelo aos demais. Dessa forma, a ordem das mensagens sendo apresentadas no monitor terminal podem ocorrer em diferentes ordens a cada iteração, e com dados de dois sensores se intercalando, por exemplo.

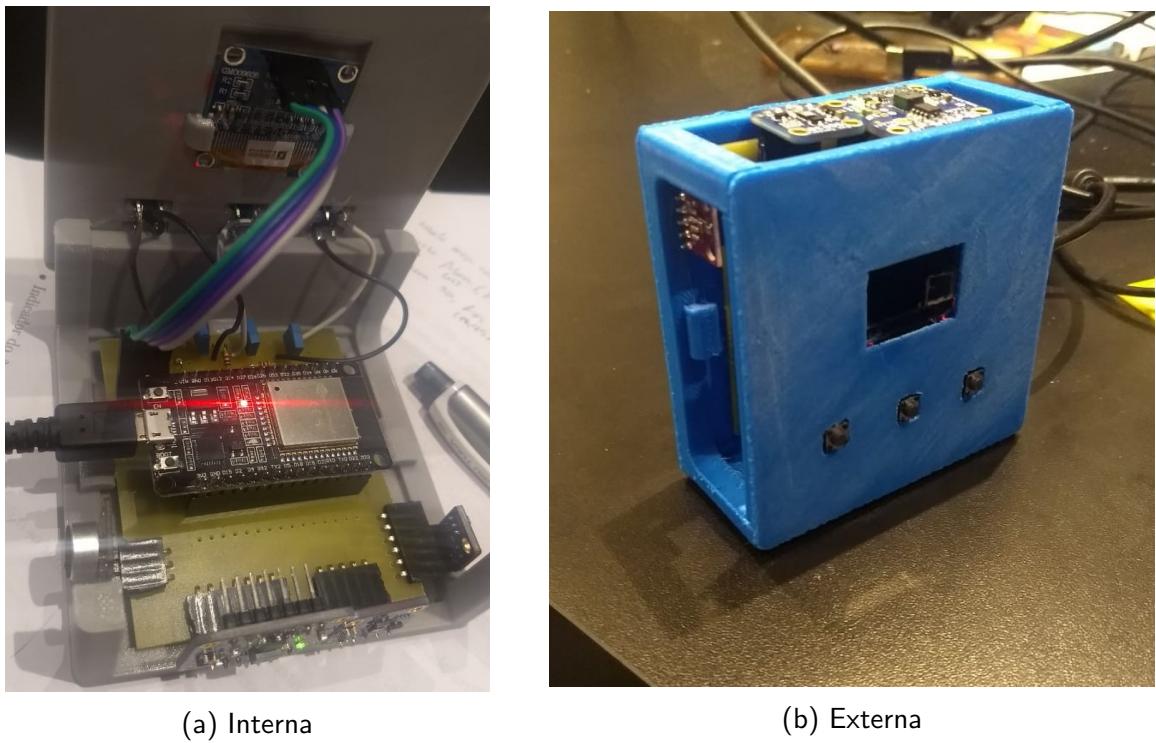


Figura 35: Montagem final do dispositivo

```
I (46038) SENSORS: Violet: 557.770752
I (46038) SENSORS: Blue: 109.263321
I (46048) SENSORS: Green: 1450.458008
I (46048) SENSORS: Yellow: 230.909409
I (46058) SENSORS: Orange: 2945.682373
I (46058) SENSORS: TVOC: 0, eCO2: 400
I (46058) SENSORS: Red: 775.602844
I (46058) SENSORS: Temperature: 31.639930
I (46068) SENSORS: LUX: 370.112885
I (46078) SENSORS: Pressure: 91876.118393
I (46088) SENSORS: Humidity: 45.582324
I (46098) NETWORK: Received 100 bytes from 192.168.4.1
```

Figura 36: Coleta dos sensores no IDF-Monitor

Em seguida, incluímos ao código o envio dos dados via Wi-Fi, como citado na seção 4.2.3.2, com o dispositivo operando como *gateway*. Os dados das coletas, agora salvos no banco de dados, foram apresentados no dashboard como mostra a figura 37.

Fabricamos outros dois dispositivos para criar a rede BLE-Mesh completa, inicialmente com um dos três dispositivos atuando como "gateway". A validação funcional de cada um deles seguiu o mesmo teste do primeiro, coletando dados do ambiente com os sensores e enviando para a plataforma via Wi-Fi.

Com os nós da rede validados, realizando a coleta e o envio das medições do ambiente, foram feitos os testes dos mesmos nós agora utilizando a arquitetura final proposta, com Bluetooth Mesh.



Figura 37: Gráficos de coletas do Gateway no Dashboard

Para provisionar os dispositivos na rede Mesh, utilizamos o aplicativo nRF Mesh [95], da Nordic Semiconductor. As etapas, comuns aos três dispositivos, estão mostradas na figura 38.

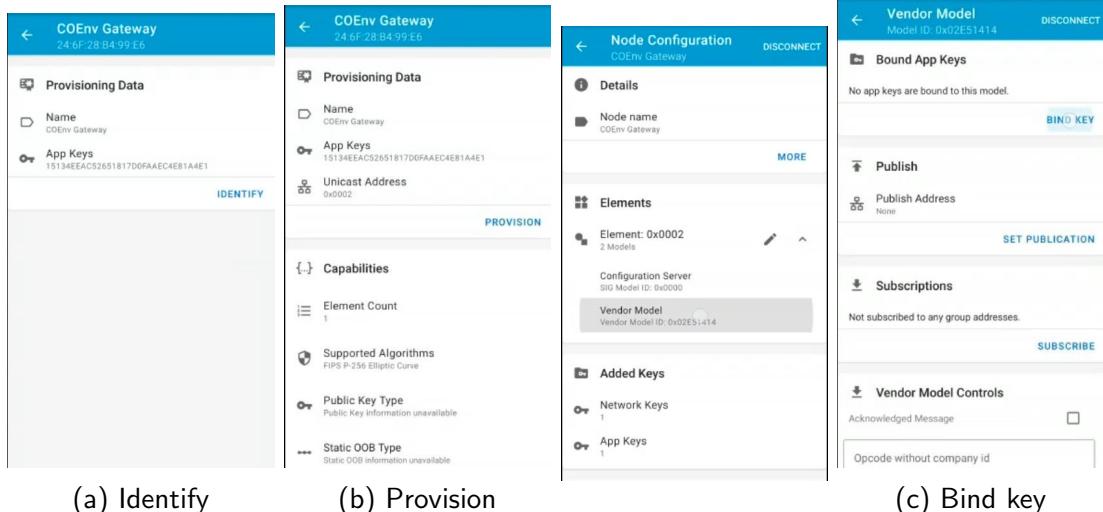


Figura 38: Telas do aplicativo nRF-Mesh

Além dessas etapas, o dispositivo que atuar como gateway - nesse caso o dispositivo 1 - precisa também se inscrever (subscribe) em um grupo, esse procedimento é mostrado na figura 39. Criamos um grupo com endereço 0xC100 para os testes da rede.

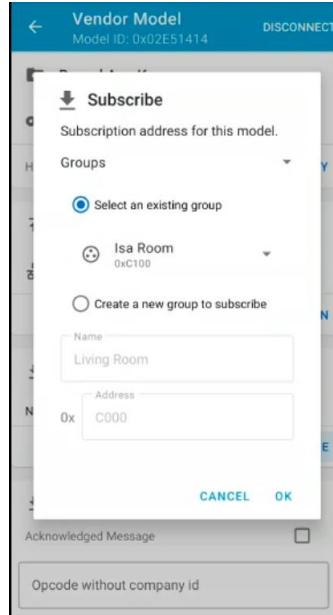


Figura 39: Tela do app nrf mesh - escolha do grupo

No dashboard foi possível visualizarmos os dados dos 3 dispositivos em gráficos temporais iguais aos utilizados nos testes funcionais do dispositivo (figura 37). Criamos também um segundo dashboard com as últimas leituras dos dispositivos, mostrado na figura 40, de forma a resumir o estado atual da rede de sensores.



Figura 40: Últimas medições dos indicadores no Dashboard

O dispositivo periodicamente inicia uma coleta de feedback do usuário, junto com uma medição dos indicadores do ambiente. As últimas respostas são apresentadas para o usuário como mostra a figura 41.

Feedback								
Gateway								
Time	Confortável com a ter	Sensação de Temper:	Temperatura	Confortável com o rui	Nível de ruído	Confortável com a luz	Sensação luminosa	
2020-12-21 06:04:51	Não	Muito quente	31 °C	Sim	67 dB	Sim	—	
Node 2								
Time	Confortável com a ter	Sensação de Temper:	Temperatura	Confortável com o rui	Nível de ruído	Confortável com a luz	Sensação luminosa	
2020-12-21 06:04:51	Não	Muito quente	30 °C	Não	42 dB	Sim	—	
Node 3								
Time	Confortável com a ter	Sensação de Temper:	Temperatura	Confortável com o rui	Nível de ruído	Confortável com a luz	Sensação luminosa	
2020-12-21 06:04:51	Não	Muito quente	31 °C	Sim	40 dB	Sim	—	

Figura 41: Feedbacks no Dashboard

Para cada parâmetro monitorado, foram definidos valores limiares segundo os níveis de indicação de qualidade de ambientes para escritórios. A partir desses limiares, foram configurados alertas que são acionados caso o valor médio de certo parâmetro dentro de uma janela de 5 minutos fique acima dos valores definidos e que enviam notificações para um canal no serviço de mensagens *Slack*.

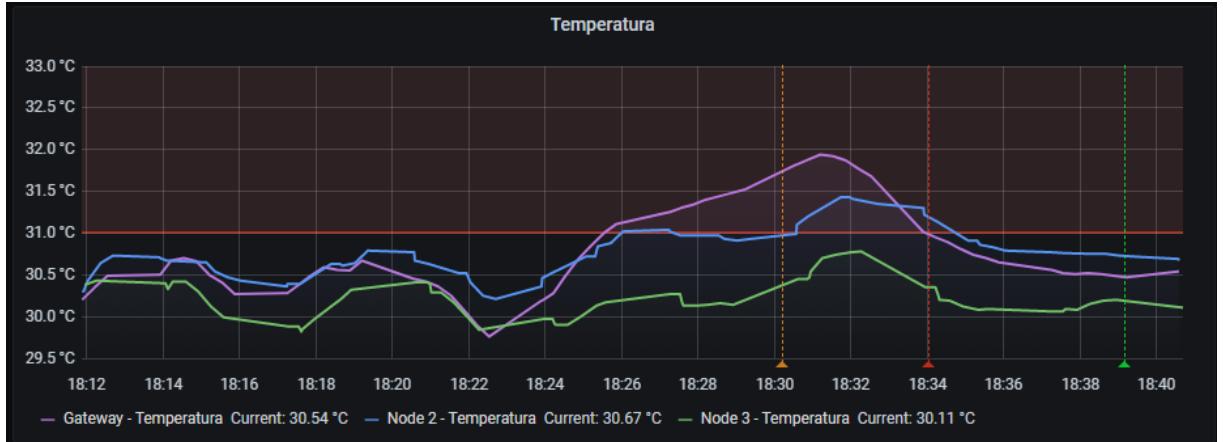
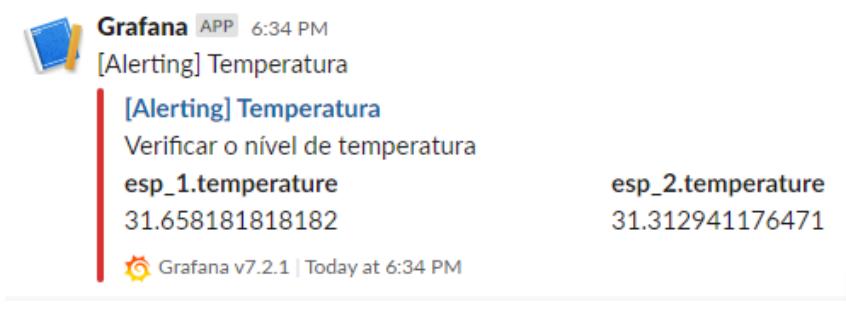
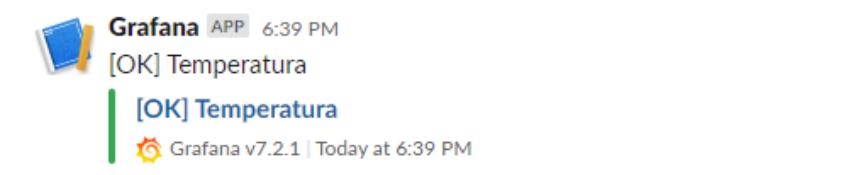


Figura 42: Exemplo de ocorrência de alerta de temperatura no Dashboard

A figura 42 ilustra um caso em que houve o disparo de uma notificação de alerta: às 18:30, o sistema percebeu que os valores presentes nas séries de cor roxa e azul estavam acima do limiar de 31 °C, que foram monitorados até às 18:34, quando ocorreu o disparo de uma notificação de alerta, ilustrada na figura 43a. Finalmente, podemos visualizar que os valores retornaram a níveis abaixo do limiar às 18:39, sendo enviada uma nova notificação para informar essa volta à normalidade, como visto na figura 43b.



(a) Notificação de alerta



(b) Notificação de normalidade

Figura 43: Exemplo de recebimento de notificações de alerta no serviço Slack

6 CONSIDERAÇÕES FINAIS

6.1 Conclusões do Projeto

O monitoramento de ambientes internos é essencial para garantir que os níveis de qualidade referentes a parâmetros gerais do ambiente estejam de acordo com os valores recomendados à saúde humana. Além disso, a coleta de valores que consideram a percepção dos ocupantes de determinado ambiente em relação aos parâmetros monitorados é igualmente importante, já que as pessoas passam grande parte do tempo em lugares fechados, como escritórios, e a sensação de conforto pode levar a aumentos de produtividade e satisfação no geral.

Trabalhos relacionados que têm o objetivo de monitorar ambientes dificilmente realizam medições dos quatro indicadores de qualidade definidos na seção 3.1.2, sendo que maioria deles é focada em qualidade térmica e do ar.

A solução desenvolvida nesse projeto conseguiu realizar a medição e o monitoramento ao longo do tempo desses parâmetros e coletar opiniões sobre os mesmos periodicamente ao longo do dia, enviando-os a uma plataforma em nuvem onde podemos verificar se os parâmetros medidos estão dentro dos níveis recomendados para um escritório fechado, além de visualizar se os seus ocupantes estão confortáveis.

A rede de dispositivos se mostrou confiável, raramente apresentando alguma perda de dados. Por ser implementada utilizando Bluetooth Mesh, a rede pode ser incrementada além dos três dispositivos protótipos desenvolvidos nesse projeto, com a possibilidade de que sejam inseridos outros tipos de dispositivos que possam interagir com os do projeto.

6.2 Perspectivas de Continuidade

6.2.1 Hardware

6.2.1.1 Alimentação

Para que os dispositivos tornem-se portáteis e independentes da rede elétrica, ao invés de alimentá-los diretamente pelo conector micro USB do módulo ESP32 seria interessante utilizar uma bateria de Lítio-Polímero de uma célula (1S). Dentre as baterias recarregáveis, a bateria LiPo possui a maior densidade energética e é uma das principais soluções utilizadas hoje em dispositivos embarcados.

A bateria LiPo tem tensões de operação entre 3.5 e 4.2 Volts. Para alimentar o circuito é necessário elevar a tensão para 5V através de um regulador chaveado boost, e então regulando para 3.3V a fim de alimentar todos os subcircuitos do hardware, mantendo assim a tensão estável, com menores oscilações.

Para fazer a recarga da bateria de forma eficiente e segura, foi pensado em um circuito carregador utilizando o CI TP4056 [96], alimentado por 5V através de um conector USB-micro. Com esse CI é possível também que o circuito opere enquanto a bateria está sendo recarregada.

6.2.2 PCB

Seria interessante desenvolver uma segunda versão da PCB, agora utilizando os Cls dos sensores e o chip ESP32, sem utilizar os módulos e kits de desenvolvimento. Isso reduz o custo da placa, além de diminuir significativamente suas dimensões. +

6.2.3 Firmware

6.2.3.1 Microfone

Para a realização do cálculo do volume ambiente em decibels é necessário ponderar o sinal recebido de forma a levar em conta a sensibilidade do ouvido. Para realizar essa filtragem correspondente, é necessário que o sinal tenha uma amostragem de ao menos 40kHz, para termos assim uma banda de 20kHz (Teorema de Nyquist).

Como o ESP32 tem uma frequência máxima de amostragem de 6kHz, para fazer uma amostragem suficiente do microfone, seria necessário utilizar um conversor analógico-digital

externo, com comunicação digital com o ESP32. Uma alternativa a isso seria trocar o microfone por um sensor microeletromecânico (MEMS, do inglês *MicroElectroMechanical Sensor* com um conversor analógico-digital integrado.

Além disso, alguns ruídos parecem influenciar mais do que o volume de forma geral, em especial ruídos de alta frequência produzidos por máquinas. Para isso, seria interessante filtrar o sinal em diferentes frequências de interesse, isto é, as que mais afetam a sensação de conforto, para conseguir identificá-las e usar a intensidade nessas frequências como indicador de conforto.

APÊNDICE A – IMAGENS

A.1 BOM

Comment	Description	Designator	Footprint	LibRef	Quantity
SPST	SWITCH TACTILE SPST-NO 0.05A 24V	BT1, BT2, BT3	TACT SWITCH PHT	SWITCH TACTILE SPST-NO 0.05A 24V	3
VJ0603Y104JXJPW1BC	Multilayer Ceramic Capacitors MLCC - SMD/SMT 0603 0.1uF 16volts X7R 5%	C1, C2, C3	CAP CER DISK 100NF 50V	CAP CER 0.1UF 16V 5% 0603	3
Display OLED	MONOCHROME 1.3 128X64	DS1	OLED 128x64 - offset	Display OLED	1
SGP30	SGP30 AIR QUALITY SENSOR BREAKOUT	H1	SGP30 1x5	SGP30	1
AS7262	Adafruit AS7262 6-Channel Visible Light / Color Sensor Breakout	H2	AS7262	AS7262	1
BME280-module	SENSOR HUM/PRESS I2C/SPI BME280	H3	BME280 1x6	BME280-module	1
Microphone	Accessory Type:Electret Microphone; For Use With:MCU & Sensor Evaluation & Development Boards; Features:20-20kHz Electret Microphone, Adjustable Gain, Excellent Power Supply Noise Rejection ;RoHS Compliant: Yes	MK1	microphone module max4466	Microphone	1
Header 15	Header, 15-Pin	P1, P2	HDR FEMALE 1x15	Header 15	2
RES 1/4W CARBON FILM	RES ? OHM 1/4W 5% CARBON FILM	R1, R2	RES 10K 1/4W CARBON FILM	RES 1/4W CARBON FILM	2
RES 1/4W CARBON FILM	RES ? OHM 1/4W 5% CARBON FILM	R3	RES 10K 1/4W CARBON FILM_STAND	RES 1/4W CARBON FILM	1
ESP32-DEVKITC	EVAL BOARD FOR ESP-WROOM-32	U1	ESP32-WROOM-devkit	ESP32-DEVKITC	1

Figura 44: Bill of Materials (Lista de Materiais)

A.2 Arquitetura do Firmware

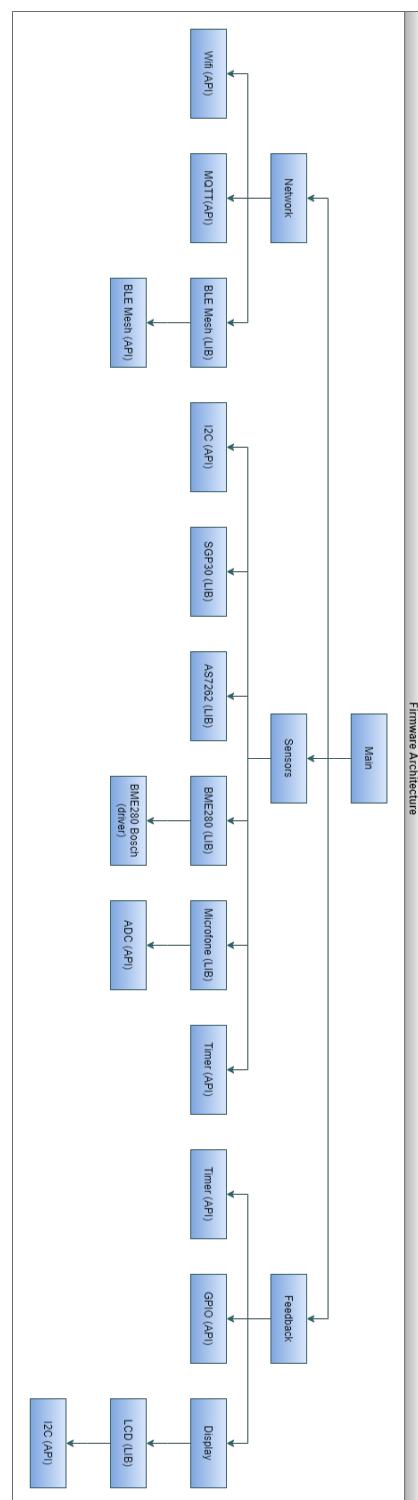


Figura 45: Arquitetura do Firmware

REFERÊNCIAS

- 1 MICROCHIP. *Introduction to Bluetooth Low Energy*. Disponível em: <<https://microchipdeveloper.com/wireless:ble-introduction>>.
- 2 CILFONE L. DAVOLI, L. B. A.; FERRARI, G. Wireless mesh networking: An iot-oriented perspective survey on relevant technologies. *Future Internet*, p. 27–31, 2019.
- 3 WOOLLEY, M. Bluetooth mesh models. p. 15–17, 2019.
- 4 DATASHEET BME280. [S.I.], 2018. Disponível em: <<https://br.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf>>.
- 5 GROUP, T. *What are acceptable VOC levels in the air*. Disponível em: <<https://www.tecamgroup.com/acceptable-voc-levels/>>.
- 6 NAIK, N. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. *2017 IEEE International Systems Engineering Symposium (ISSE)*, p. 1–7, 2017.
- 7 AMS. *Datasheet AS7262*. [S.I.], 2017. Disponível em: <https://br.mouser.com/datasheet/2/588/AS7262_DS000486_2-00-1513124.pdf>.
- 8 Kumar, A.; Hancke, G. P. An energy-efficient smart comfort sensing system based on the ieee 1451 standard for green buildings. *IEEE Sensors Journal*, v. 14, n. 12, p. 4245–4252, 2014.
- 9 Tuan Anh Nguyen and Marco Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and Buildings*, v. 56, p. 244–257, 2013.
- 10 CICS website (<http://cics.prp.usp.br/>). Disponível em: <<http://cics.prp.usp.br/>>.
- 11 Ciabattoni, L. et al. Iot based indoor personal comfort levels monitoring. *2016 IEEE International Conference on Consumer Electronics (ICCE)*, p. 125–126, 2016.
- 12 NR17. *Ergonomia*. [S.I.], 2018. Disponível em: <https://enit.trabalho.gov.br/portal/images/Arquivos_SST/SST_NR/NR-17.pdf>.
- 13 NBR10152. *Níveis de ruído para conforto acústico*. [S.I.], 1987. Disponível em: <http://www.joaopessoa.pb.gov.br/portal/wp-content/uploads/2015/02/NBR-10152-1987-Conforto-Ac_stico.pdf>.
- 14 NBR5413. *Iluminância de Interiores*. [S.I.], 1992. Disponível em: <<http://ftp.demec.ufpr.br/disciplinas/TM802/NBR5413.pdf>>.
- 15 A.P. Jones. Indoor air quality and health. *Atmospheric Environment*, v. 33, p. 4525–4564, 1999.
- 16 EARTH, C. *Daily CO2*. Disponível em: <<https://www.co2.earth/daily-co2>>.

- 17 CARBON Dioxide. Disponível em: <<https://www.dhs.wisconsin.gov/chemical/carbon dioxide.htm>>.
- 18 J. O'SHagan, M. Khazova and L. Price. Low-energy light bulbs, computers, tablets and the blue light hazard. *Eye*, v. 30, p. 230–233, 2016.
- 19 CHOJER, H. et al. Development of low-cost indoor air quality monitoring devices: Recent advancements. *Science of The Total Environment*, v. 727, p. 138385, 2020. ISSN 0048-9697. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0048969720318982>>.
- 20 PARKINSON, T.; PARKINSON, A.; de Dear, R. Continuous ieq monitoring system: Context and development. *Building and Environment*, v. 149, p. 15 – 25, 2019. ISSN 0360-1323. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360132318307467>>.
- 21 PARKINSON, T.; PARKINSON, A.; de Dear, R. Continuous ieq monitoring system: Performance specifications and thermal comfort classification. *Building and Environment*, v. 149, p. 241 – 252, 2019. ISSN 0360-1323. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360132318307522>>.
- 22 KARAMI, M.; MCMORROW, G. V.; WANG, L. Continuous monitoring of indoor environmental quality using an arduino-based data acquisition system. *Journal of Building Engineering*, v. 19, p. 412 – 419, 2018. ISSN 2352-7102. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2352710218301025>>.
- 23 CARRE, A.; WILLIAMSON, T. Design and validation of a low cost indoor environment quality data logger. *Energy and Buildings*, v. 158, p. 1751 – 1761, 2018. ISSN 0378-7788. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0378778817322016>>.
- 24 TIELE, A.; ESFAHANI, S.; COVINGTON, J. Design and development of a low-cost, portable monitoring device for indoor environment quality. *Journal of Sensors*, v. 2018, 2018.
- 25 Chanthakit, S.; Rattanapoka, C. *MQTT Based Air Quality Monitoring System using Node MCU and Node-RED*. 2018. 1-5 p.
- 26 SCARPA, M. et al. Development and testing of a platform aimed at pervasive monitoring of indoor environment and building energy. *Energy Procedia*, v. 126, p. 282 – 288, 2017. ISSN 1876-6102. ATI 2017 - 72nd Conference of the Italian Thermal Machines Engineering Association. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1876610217336482>>.
- 27 Jiang, B.; Huacón, C. F. Cloud-based smart device for environment monitoring. p. 1–6, 2017.
- 28 SALAMONE, F. et al. How to control the indoor environmental quality through the use of the do-it-yourself approach and new pervasive technologies. *Energy Procedia*, v. 140, p. 351 – 360, 2017. ISSN 1876-6102. Beyond NZEB Buildings (AiCARR 50th International Congress, Matera (I), 10-11 May 2017). Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1876610217355595>>.
- 29 SALAMONE, F. et al. A low-cost environmental monitoring system: How to prevent systematic errors in the design phase through the combined use of additive manufacturing and thermographic techniques. *Proceedings*, v. 1, n. 18, 2017.

- 30 ALI, A. S. et al. Open source building science sensors (osbss): A low-cost arduino-based platform for long-term indoor environmental data collection. *Building and Environment*, v. 100, p. 114 – 126, 2016. ISSN 0360-1323. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360132316300476>>.
- 31 Brunelli, D. et al. Povomon: An ad-hoc wireless sensor network for indoor environmental monitoring. p. 1–6, 2014.
- 32 Tapashetti, A.; Vegiraju, D.; Ogunfunmi, T. IoT-enabled air quality monitoring device: A low cost smart health solution. p. 682–685, 2016.
- 33 MARQUES GONÇALO; PITARMA, R. "a cost-effective air quality supervision solution for enhanced living environments through the internet of things". *Electronics*, v. 8, n. 2, 2019.
- 34 MARTÍN-GARÍN, A. et al. Environmental monitoring system based on an open source platform and the internet of things for a building energy retrofit. *Automation in Construction*, v. 87, p. 201 – 214, 2018. ISSN 0926-5805. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0926580517300729>>.
- 35 DUARTE, C.; PARKINSON, T. *Targeted occupant surveys: A novel method to effectively relate occupant feedback with environmental conditions*. 2020.
- 36 LLC, Q. Qualtrics - online survey software and insight platform. Disponível em: <<https://www.qualtrics.com/pt-br/research-core/pesquisa-software>>.
- 37 GOBAIN, S. MULTI COMFORT. Disponível em: <<https://multicomfort.saint-gobain.com/>>.
- 38 GOBAIN, S. MultiComfort 350. Disponível em: <<https://www.saint-gobain.com.br/mc350>>.
- 39 METRIFUL. Metriful Sense. Disponível em: <<https://www.metriful.com/>>.
- 40 SPACE, L. L. smart office. CoMoS Comfort Monitoring Station. Disponível em: <<http://www.livinglab-smartofficespace.com/en/research/heat-and-thermal-comfort/detail/comos-comfort-monitoring-station/>>.
- 41 HC Technologies. RETAIL SPACE COMFORT MONITORING. Disponível em: <<https://www.hc-technologies.com/retail-iot-solutions/store-ambient-comfort-monitoring>>.
- 42 ENTECH. ECOMLITE monitoring indoor comfort. Disponível em: <<https://www.entech.co.th/product/ecomlite/>>.
- 43 NETATMO. Smart Indoor Air Quality Monitor. Disponível em: <<https://www.netatmo.com/el-gr/aircare/homecoach>>.
- 44 LABS, S. Senlab O: Indoor Comfort and occupancy monitoring. Disponível em: <<https://sensing-labs.com/portfolio-item/21334/>>.
- 45 CLICK, C. Office: Smart Buildings. Disponível em: <<https://www.comfortclick.com/SmartBuilding/Solutions/Office>>.

- 46 Gao, Q.; Zhang, K.; Li, D. Research on visual comfort based on fuzzy neural network. p. 884–888, 2018.
- 47 ZHENG, A. J. J. Wireless sensor networks: A networking perspective. In: _____. [S.I.]: Wiley-IEEE Press, 2009. cap. 1, p. 35. ISBN 9780470167632.
- 48 AspenCore. *2019 Embedded Markets Study*: Integrating iot and advanced technology designs, application development and processing environments. embedded, 2019. Disponível em: <https://www.embedded.com/wp-content/uploads/2019/11/EETimes_EMBEDDED_2019_EMBEDDED_MARKETS_STUDY.pdf>.
- 49 ELETTRONIC ENGINEERING TIMES. *EETimes*. Disponível em: <<https://www.eetimes.com>>.
- 50 ASPENCORE. *embedded*. Disponível em: <<https://www.embedded.com>>.
- 51 IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, p. 1–3534, 2016.
- 52 INTEL. *Different Wi-Fi Protocols and Data Rates*. Disponível em: <<https://www.intel.com/content/www/us/en/support/articles/000005725/network-and-i-o/wireless-networking.html>>.
- 53 DONG, J. et al. Indoor passive ranging based on wifi. p. 1, 2014.
- 54 IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, p. 1–800, 2020.
- 55 NOVELBITS. *Bluetooth 5 BLE: Achieving maximum throughput*. Disponível em: <<https://www.novelbits.io/bluetooth-5-speed-maximum-throughput/>>.
- 56 MQTT.ORG. *MQTT Specifications*. Disponível em: <<https://mqtt.org/mqtt-specification/>>.
- 57 SHELBY, Z.; HARTKE, K.; BORMANN, C. *The Constrained Application Protocol (CoAP)*. RFC Editor, 2014. RFC 7252. (Request for Comments, 7252). Disponível em: <<https://rfc-editor.org/rfc/rfc7252.txt>>.
- 58 SIEKKINEN, M. H. M.; NURMINEN, J. How low energy is bluetooth low energy? p. 1–6, 2012.
- 59 BLUETOOTH SIG. *Mesh Networking Specifications*. Disponível em: <<https://www.bluetooth.com/specifications/mesh-specifications>>.
- 60 BLUETOOTH SIG. *2020 Bluetooth Market Update*. Disponível em: <https://www.bluetooth.com/wp-content/uploads/2020/03/2020_Market_Update-EN.pdf>.
- 61 BLUETOOTH SIG. *GATT Specifications*. Disponível em: <<https://www.bluetooth.com/specifications/gatt>>.

- 62 Li Huang, Yingxin Zhu, Qin Ouyang, Bin Cao. A study on the effects of thermal, luminous, and acoustic environments on indoorenvironmental comfort in offices. *Building and environment*, v. 49, p. 304–309, 2012.
- 63 SENSIRION. *Datasheet SGP30*. [S.I.], 2020. Disponível em: <https://br.mouser.com/datasheet/2/682/Sensirion_Gas_Sensors_SGP30_Datasheet-1843629.pdf>.
- 64 Solomon Systech. *128 x 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller*. [S.I.], 2008. Disponível em: <<https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf>>.
- 65 FLYNN, M. J.; LUK, W. Computer System Design: System-on-Chip. In: _____. [S.I.]: John Wiley and Sons, Inc, 2011. cap. Introduction to the Systems Approach.
- 66 ESPRESSIF. *ESP32*. Disponível em: <<https://www.espressif.com/en/products/modules/esp-wroom-32/overview>>.
- 67 ESPRESSIF SYSTEMS. *ESP32 Series datasheet*. [S.I.], 2020. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>.
- 68 COSMI, A.; MOTA, V. Uma análise dos protocolos de comunicação para internet das coisas. In: *Anais do III Workshop de Computação Urbana*. Porto Alegre, RS, Brasil: SBC, 2019. p. 153–166. ISSN 2595-2706. Disponível em: <<https://sol.sbc.org.br/index.php/courb/article/view/7475>>.
- 69 INC., E. F. *Eclipse Mosquitto™*. Disponível em: <<https://mosquitto.org>>.
- 70 LUZAR, A.; STANOVNIK, S.; CANKAR, M. Time series or relational database for edge and iot. In: _____. [S.I.]: Zenodo, 2018.
- 71 ALTIUM. *Altium Designer®*. Disponível em: <<https://www.altium.com/altium-designer/>>.
- 72 BOLOGNA, I.; FREITAS, R. *Repositório do projeto de Hardware*. Disponível em: <<https://github.com/co-env/hardware>>.
- 73 ESPRESSIF. *Espressif IoT Development Framework*. Disponível em: <<https://github.com/espressif/esp-idf>>.
- 74 SERVICES, A. W. *FreeRTOS Homepage*. Disponível em: <<https://www.freertos.org/>>.
- 75 SENSORTEC, B. *BME280 sensor driver*. Disponível em: <https://github.com/BoschSensortec/BME280_driver>.
- 76 BOLOGNA, I.; FREITAS, R. *ESP-IDF BME280 Library*. Disponível em: <<https://github.com/co-env/BME280>>.
- 77 BOLOGNA, I.; FREITAS, R. *ESP-IDF AS7262 Library*. Disponível em: <https://github.com/co-env/esp32_AS7262>.
- 78 BOLOGNA, I.; FREITAS, R. *ESP-IDF SGP30 Library*. Disponível em: <https://github.com/co-env/esp32_SGP30>.

- 79 INDUSTRIES, A. *Electret Microphone Amplifier - MAX4466 with Adjustable Gain.* Disponível em: <<https://www.adafruit.com/product/1063>>.
- 80 K, T. *SSD1306 Component for the ESP-IDF SDK.* Disponível em: <<https://github.com/TaraHoleInIt/tarablessd1306>>.
- 81 K, T. *Tara K's github page.* Disponível em: <<https://github.com/TaraHoleInIt>>.
- 82 FOUNDATION, T. L. *The Zephyr Project.* Disponível em: <<https://zephyrproject.org>>.
- 83 SERVICES, A. W. *Tipos de instância do Amazon EC2.* Disponível em: <<https://aws.amazon.com/pt/ec2/instance-types/>>.
- 84 KALISKI, B. *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services.* RFC Editor, 1993. RFC 1424. (Request for Comments, 1424). Disponível em: <<https://rfc-editor.org/rfc/rfc1424.txt>>.
- 85 INFLUXDATA. *Install InfluxDB OSS.* Disponível em: <<https://docs.influxdata.com/influxdb/v1.8/introduction/install/#installing-influxdb-oss>>.
- 86 NEWMAN, C.; KLYNE, G. *Date and Time on the Internet: Timestamps.* RFC Editor, 2002. RFC 3339. (Request for Comments, 3339). Disponível em: <<https://rfc-editor.org/rfc/rfc3339.txt>>.
- 87 INFLUXDATA. *Install Telegraf.* Disponível em: <<https://docs.influxdata.com/telegraf/v1.16/introduction/installation/>>.
- 88 LABS, G. *Install Grafana on Debian or Ubuntu.* Disponível em: <<https://grafana.com/docs/grafana/latest/installation/debian/>>.
- 89 GITHUB. *GitHub Education.* Disponível em: <<https://education.github.com>>.
- 90 CLOUDFLARE. Disponível em: <<https://www.cloudflare.com>>.
- 91 FOUNDATION, E. F. *Certbot.* Disponível em: <<https://certbot.eff.org>>.
- 92 LETS Encrypt. Disponível em: <<https://letsencrypt.org>>.
- 93 ULTIMAKER. *Ultimaker Cura 4.8.* Disponível em: <<https://ultimaker.com/software/ultimaker-cura>>.
- 94 ESPRESSIF. *Logging library.* Disponível em: <<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/log.html>>.
- 95 Nordic Semiconductor. *nRF Mesh (Version 2.4.4).* Disponível em: <<https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Mesh>>.
- 96 CORP, N. T. P. A. *TP4056 Datasheet.* [S.I.]. Disponível em: <<https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf>>.