UNIVERSIDADE FEDERAL DE ALAGOAS

CENTRO DE TECNOLOGIA

ENGENHARIA AMBIENTAL E SANITÁRIA

Adelson da Silva Santos

Montagem de Protótipo de Detector de Gases de Combustão

Maceió

2021

Adelson da Silva Santos

Montagem de Protótipo de Detector de Gases de Combustão

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Ambiental e Sanitária como requisito para a obtenção do título de Bacharel em Engenharia Ambiental e Sanitária.

Maceió

2021

Universidade Federal de Alagoas – UFAL
Centro de Tecnologia – CTEC
Coordenação do Curso de Engenharia Ambiental e Sanitária
Cidade Universitária – Campus A. C. Simões
Tabuleiro do Martins – CEP 57072-970 – Maceió – Alagoas
Tel: (0**82) 3214-1281 – Fax: (0**82)3214-1625

Ata de defesa do Trabalho de Conclusão de Curso

Ao(s) 30 dias(s) do mês de SETEMBRO de 2021 realizou-se às 15h00min, por meio de videoconferência, a defesa do **Trabalho de Conclusão de Curso** do(a) discente **Adelson da Silva Santos** intitulado "MONTAGEM DE PROTÓTIPO DE DETECTOR DE GASES DE COMBUSTÃO ". A Banca Examinadora foi constituída por Christiano Cantarelli Rodrigues (Orientador), Selêude Wanderley da Nóbrega (Coorientadora), Karina Ribeiro Salomon e Marcio Gomes Barboza. Após a apresentação do discente pelo orientador, ele expôs seu trabalho, sendo, logo a seguir, arguido pelos componentes da Banca Examinadora. O Trabalho de Conclusão de Curso obteve as seguintes notas de cada um dos avaliadores: Karina Ribeiro Salomon (**10,00**) e Marcio Gomes Barboza (**10,00**), resultando numa média (**10,00**). Os registros de notas e de solicitação de correções estão documentados nos formulários de notas e de correções, respectivamente, preenchidos pelos avaliadores. E, para constar, foi lavrada a presente ata que vai assinada pelos componentes da Banca Examinadora.

Maceió, 30 de setembro de 2021.

Christiano Cantarelli Rodrigues
(Orientador) - CTEC/UFAL)

Selêude Wanderley da Nóbrega
(Coorientadora – CTEC/UFAL)

Karina Ribeiro Salomon
(CTEC/UFAL)

Marcio Gomes Barboza
(CTEC/UFAL)

Para minha mãe, que me ensinou que inteligência sem doses de afeto não vale de nada.

E para meu Pai, que sempre me incentivou a amar a área de exatas.

AGRADECIMENTOS

Primeiramente a Deus pela saúde e força para superar as dificuldades.

Aos meus pais, José Adelson e Vanilda, que sempre me incentivaram nos estudos, por me ensinarem a não desistir e não me permitirem desanimar.

À minha irmã, Aurea, e à minha namorada, Ingrid, que diariamente me ensinam o que é ter força de vontade.

Ao meu orientador, Christiano Rodrigues, que teve a paciência e compreensão para aturar minha ausência em vários momentos, e pelo auxílio imediato quando necessário.

À minha amiga, Ana Letícia, pelas inúmeras correções e sugestões quanto à escrita e apresentação deste projeto.

A todos os professores dos quais tive a honra de ser aluno, que me guiaram pelo caminho do conhecimento permitindo hoje que eu pudesse concluir este trabalho.

Aos amigos e amigas que, com carinho e companheirismo, contribuíram para fortalecer minha saúde emocional e mental durante todo o curso de Engenharia Ambiental e Sanitária. Em especial, agradeço ao Augusto (exemplo de programador), ao Wallef (o gigante mais gentil que conheço), à Heloize Maria (irmã de coração), à Pietra (minha eterna chefinha), ao Lucas Henrique, Alexandre, Matheus Sousa e Vitor (pelos momentos de descontração e alegria), à Eduarda, Elber, Mateus Normande, Lucas Mendes e Heloisa (companheiros de estágio).

Ao Centro Acadêmico de Engenharia Ambiental e Sanitária, pela oportunidade de conviver com pessoas incríveis que buscam sempre enaltecer e melhorar o curso.

“Fale, amigo, e entre.”

Portão de Moria.

RESUMO

Tendo em vista as preocupações com a poluição atmosférica e os danos à saúde que essa poluição pode acarretar, governos seguem as recomendações da Organização Mundial da Saúde para embasar suas normas e resoluções referentes às restrições das concentrações de substâncias nocivas que são descartadas na atmosfera. Para uma efetiva fiscalização, faz-se necessário empregar equipamentos de medição da concentração destes poluentes, o que ainda não é uma realidade plena no Brasil. Outra preocupação é o monitoramento da qualidade do ar no interior de ambientes fechados – residências, locais de trabalho, centros comerciais, escolas e hospitais - onde a presença de poluentes atmosféricos pode apresentar níveis de concentração até cinco vezes maiores que nos ambientes externos. No entanto, devido aos custos de aquisição e à necessidade de mão de obra especializada para operação, o emprego de equipamentos tradicionais de monitoramento da qualidade do ar torna-se inviável para monitoramento de ambientes fechados. Diante disto, este trabalho teve o objetivo de utilizar sistemas embarcados e tecnologia de baixo custo para o desenvolvimento e montagem de uma pequena estação de monitoramento da qualidade do ar para ambientes internos e classificar a qualidade do ar em um ambiente fechado tendo como referência a legislação vigente. Foram monitorados os poluentes: CO, CO_2, $MP_{2,5}$, MP_{10} e GLP. Os resultados obtidos, mostraram que o particulado $MP_{2,5}$ e o CO chegaram a atingir algumas vezes níveis superiores aos propostos pela legislação vigente. A estação monitorou um ambiente fechado por um período de quarenta e três dias, sendo de fácil utilização e apresentando os resultados de forma simples e intuitiva.

Palavras-chave: Monitoramento de Gases, Sistema de Medição, Plataformas Embarcadas.

ABSTRACT

In view of concerns about air pollution and the damage to health that this pollution can cause, governments follow the recommendations of the World Health Organization to base their rules and resolutions regarding restrictions on concentrations of harmful substances that are discharged into the atmosphere. For an effective inspection, it is necessary to use equipment to measure the concentration of these pollutants, which is still not a full reality in Brazil. Another concern is the monitoring of indoor air quality in closed environments – homes, workplaces, shopping centers, schools and hospitals – where the presence of air pollutants can present concentration levels up to five times higher than in outdoor environments. However, due to acquisition costs and the need for specialized labor for the operation, the use of traditional air quality monitoring equipment becomes unfeasible for monitoring indoors. Therefore, this work aimed to use embedded systems and low-cost technology for the development and assembly of a small air quality monitoring station for indoor environments and to classify the air quality in a closed environment with reference to the legislation current. The following pollutants were monitored: CO, CO_2, $PM_{2.5}$, PM_{10} and GLP, where it was found that $PM_{2.5}$ particulates and CO sometimes reached levels higher than the limit proposed by current legislation. The station monitored a closed environment for a period of forty-three days, being easy to use and presenting the results in a simple and intuitive way.

Keywords: Gas Monitoring, Measurement System, Embedded Platforms.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

As	Arsênio
ANVISA	Agência Nacional de Vigilância Sanitária
Cd	Cádmio
CETESB	Companhia Ambiental do Estado de São Paulo
CH_4	Metano
CO	Monóxido de Carbono
CO_2	Dióxido de Carbono
CONAMA	Conselho Nacional do Meio Ambiente
EPA	Environmental Protection Agency
GLP	Gás Liquefeito do Petróleo
GPIO	General Purpose Input/Output
H_2S	Sulfeto de Hidrogênio
HCl	Ácido Clorídrico
HD	Hard Disk
HDMI	High-Definition Multimedia Interface
HF	Ácido Fluorídrico
HNO_3	Ácido Nítrico
IEMA	Instituto de Energia e Meio Ambiente
IoT	Internet of Things
$MP_{0,1}$	Material Particulado de diâmetro 0,1 µm
MP_{10}	Material Particulado de diâmetro 10 µm
$MP_{2,5}$	Material Particulado de diâmetro 2,5 µm

NH3	Amônia
Ni	Níquel
NO	Óxido Nítrico
NO_2	Dióxido de Nitrogênio
O_3	Ozônio
OMS	Organização Mundial da Saúde
Pb	Chumbo
ppm	Partes Por Milhão
SO_2	Dióxido de Enxofre
SO_3	Trióxido de Enxofre
USB	Universal Serial Bus

SUMÁRIO

1. INTRODUÇÃO

A poluição atmosférica chama a atenção da sociedade acadêmica em todo o mundo, devido aos problemas de saúde que acarreta. Em sua maioria, estes malefícios são consequência da má qualidade do ar nos interiores dos domicílios e de locais de trabalho, onde a população passa, em média, de 80% a 90% de seu tempo (SCHIRMER *et al*., 2011).

Segundo Cançado *et al.* (2006), pode-se definir a poluição atmosférica como a presença de substâncias estranhas na atmosfera - podendo ser resultantes da atividade humana ou de processos naturais - em concentrações suficientes para interferir direta ou indiretamente na saúde, na segurança e no bem-estar dos seres vivos.

Estudos da Agência de Proteção Ambiental dos Estados Unidos (EPA) indicam que os níveis de concentração de poluentes podem ser de duas a cinco vezes maiores em ambientes internos do que nos externos (SCHIRMER *et al*., 2011). De acordo com Azuaga (2000), entre os danos ao meio ambiente e à saúde humana causados pela emissão dos poluentes atmosféricos, destacam-se a acidificação de rios e florestas, o aumento de problemas respiratórios e circulatórios na população e a perda do bem-estar.

Cientes disto, diversos países como Estados Unidos da América e os países que compões a União Europeia passaram a formular padrões de qualidade do ar e a estabelecer limites de tolerância para a emissão de poluentes desta natureza, visando minimizar os efeitos à saúde da população (MARTINS *et al.*, 2001). No entanto, no Brasil há poucas unidades de monitoramento de qualidade de ar (IEMA, 2014). Apenas dez estados e o Distrito Federal monitoram a qualidade do ar através de 371 estações (VORMITTAG *et al.*, 2021). O dado torna-se ainda mais preocupante ao se observar que existem poucos estudos brasileiros referentes à poluição em ambientes fechados (SCHIRMER *et al*., 2011).

Diante do cenário apresentado, evidencia-se a necessidade de maiores cuidados e do monitoramento da qualidade do ar de interiores, bem como da adoção de medidas mais rigorosas e especificas (SCHIRMER *et al*., 2011). No entanto, equipamentos que façam monitoramento de gases de forma contínua possuem um custo monetário bastante elevado, podendo requerer também mão de obra especializada para manuseio e manutenção. Além disso, tais equipamentos não são portáteis e, muitas vezes, necessitam

de instrumentação de apoio (DO AMARAL, 2007) o que torna seu emprego em ambientes interiores inviável.

Todavia, com o avanço da tecnologia, é possível criar e reinventar tais processos de forma mais eficiente e econômica. Como exemplo, cita-se o barateamento e a diminuição do tamanho dos componentes de acesso à internet, que possibilitaram o acesso de equipamentos e aparelhos mais simples à rede global (LEITE, 2016). A partir destes avanços, criou-se o conceito da Internet das Coisas (IoT, do inglês *Internet of Things*), que vem causando um grande impacto no modo como a sociedade vive e recebido tanto a atenção da comunidade acadêmica quanto da indústria, devido ao seu potencial de uso (SANTOS *et al*., 2016).

Frente às informações aqui apresentadas, este trabalho objetivou montar um equipamento de baixo custo para monitorar gases de combustão e materiais particulados em ambientes fechados. Para isto, utilizou-se um microcontrolador NodeMCU, que constitui uma placa de desenvolvimento de fácil utilização e aprendizado, além de sensores específicos para os parâmetros a serem monitorados, que apresentam baixo custo de aquisição e são encontrados com facilidade no mercado atual.

2. OBJETIVOS

Este trabalho teve como objetivo criar uma pequena estação de monitoramento da qualidade do ar, de baixo custo e de fácil utilização, para ambientes internos. Os objetivos específicos consistiram em:

- Gerar modelos de sensibilidade para a concentração do gás de cada sensor;
- Montar uma miniestação de monitoramento, utilizando uma placa de desenvolvimento NodeMCU, que, com os sensores de gás da família MQ-X.
- Monitorar gases de combustão e inflamáveis (CO e Gás Liquefeito de Petróleo) e, com um sensor de presença, monitorar partículas suspensas;
- Desenvolver um algoritmo de controle em linguagem C/C++;
- Classificar a qualidade de ar de acordo com a Resolução do CONAMA nº 491/2018;
- Elaborar um banco de dados para armazenar os dados coletados pelo microcontrolador; e
- Disponibilizar as informações de forma visual e de fácil entendimento para o usuário, com o auxílio da linguagem de programação Python.

3. REVISÃO DE LITERATURA

3.1 Qualidade do Ar

No início do século XX, pensava-se que o ar que respiramos jamais estaria em má condição para a manutenção da vida. Contudo, a qualidade do ar passou a se tornar uma das maiores preocupações da humanidade (RUSSO, 2010).

O grande aumento das emissões de poluentes, em sua maior parte, é proveniente da queima de combustíveis fosseis oriunda da rápida urbanização e da industrialização que ocorre em todo o globo. Cerca de 50% da população mundial está potencialmente exposta a níveis maiores de poluentes atmosféricos apenas por viverem em zonas urbanas. Os outros 50%, que estão sobretudo em países em desenvolvimento, utilizam principalmente combustíveis derivados de biomassa como fonte de energia, aquecimento e iluminação, o que os deixa potencialmente expostos a gases danosos derivados da queima destes materiais (ARBEX *et al.*, 2012).

Segundo o Instituto de Energia e Meio Ambiente (IEMA, 2014), a degradação da qualidade de ar é causada por fatores como: taxas de emissões de efluentes, localização e concentração das fontes poluidoras, características físico-químicas dos poluentes e dispersão e reações químicas que acontecem entre eles.

A má qualidade do ar é associada ao agravamento de doenças respiratórias, cardiovasculares e neurológicas (IEMA, 2014). Estimativas globais sugerem que a poluição ambiental externa cause 1,15 milhões de mortes em todo o mundo por ano, enquanto a poluição ambiental interna acarreta aproximadamente 2 milhões de mortes por ano. A Organização Mundial da Saúde estima que, no Brasil, a poluição atmosférica cause cerca de 20 mil mortes por ano, enquanto a poluição do ar no interior dos domicílios causa cerca de 10,7 mil óbitos por ano (*apud* ARBEX *et al.*, 2012).

Desde os anos 1990, mais da metade dos ambientes internos - como apartamentos, escolas e até hospitais - tinham ar de má qualidade (WHO, 1990). Diante disto, o acompanhamento e a elaboração de diagnósticos são de fundamental importância. No entanto, os equipamentos de monitoramento da qualidade do ar têm um valor monetário alto, o que dificulta sua aquisição e montagem (JACOMINO *et al.*, 2009).

3.2 Poluentes

De acordo com o Art. 2°, Inciso I, da Resolução do Conselho Nacional de Meio Ambiente (CONAMA) N° 491 de 2018, um poluente atmosférico é definido como qualquer forma de matéria em quantidade, concentração, tempo, ou mais características, que tornem ou possam tornar o ar improprio ou nocivo à saúde, inconveniente ao bem-estar público, danoso aos materiais, à fauna e flora ou prejudicial à segurança, ao uso e gozo da propriedade ou às atividades normais da comunidade (CONSELHO NACIONAL DO MEIO AMBIENTE, 2018). O Quadro 1 mostra a classificação das substâncias poluentes.

Para determinar os valores de concentração de um poluente específico, há alguns instrumentos de gestão da qualidade do ar, sendo um deles o padrão de qualidade do ar. No Brasil, os parâmetros regulamentados pela legislação ambiental para a qualidade do ar referem-se aos poluentes: material particulado (MP_{10} e $MP_{2.5}$), dióxido de enxofre (SO_2), dióxido de nitrogênio (NO_2), ozônio (O_3), fumaça, monóxido de carbono (CO), partículas totais em suspensão (PTS) e chumbo (Pb).

Quadro 1 - Classificação das substâncias poluentes

Compostos de Enxofre	Compostos de Nitrogênio	Compostos Orgânicos	Monóxido de Carbono	Compostos Halogenados	Metais Pesados	Material Particulado	Oxidantes Fotoquímicos
SO2	NO	Hidrocarbonetos Álcoois	CO	HCl	Pb	Mistura de compostos no estado sólido ou líquido	O3
SO3	NO2	Aldeídos		HF	Cd		Formaldeído
Compostos de Enxofre Reduzidos (H2S, Mercaptanas, Dissulfeto de Carbono)	NH3	Cetonas		Cloretos	As		Acroleína
	HNO3	Ácidos Orgânicos		Fluoretos	Ni		PAN
Sulfatos	Nitratos						

Fonte: Adaptado do site da CETESB (cetesb.sp.gov.br) (2021).

3.3 Padrão de Qualidade do Ar

De acordo com a Resolução CONAMA Nº 491 de 2018, o padrão de qualidade do ar é um instrumento de gestão que determina o valor de concentração limite de um poluente específico, associado a um intervalo de tempo de exposição, a fim de preservar o meio ambiente e a saúde da população.

Ainda de acordo com a normativa, os Padrões Nacionais de Qualidade do Ar são parte estratégica do Programa Nacional de Controle da Qualidade do Ar (PRONAR) e foram estabelecidos para serem cumpridos de forma gradativa. Os padrões de qualidade intermediários e final constam no Anexo I da supracitada resolução, com o Padrão Intermediário 1 (PI-1) entrando em vigor no dia de publicação da Resolução. Ressalta-se que os padrões de qualidades do ar final (PF) são os valores guia definidos pela Organização Mundial de Saúde (OMS), em 2005.

Tabela 2 - Padrão Nacional de Qualidade do Ar

Poluente Atmosférico	Período de Referência	PI-1	PI-2	PI-3	PF
Material Particulado - MP10	24 horas	120 µg/m³	100 µg/m³	75 µg/m³	50 µg/m³
	Anual	40 µg/m³	35 µg/m³	30 µg/m³	20 µg/m³
Material Particulado - MP2,5	24 horas	60 µg/m³	50 µg/m³	37 µg/m³	25 µg/m³
	Anual	20 µg/m³	17 µg/m³	15 µg/m³	10 µg/m³
Dióxido de Enxofre - SO2	24 horas	125 µg/m³	50 µg/m³	30 µg/m³	20 µg/m³
	Anual	40 µg/m³	30 µg/m³	20 µg/m³	-
Dióxido de Nitrogênio - NO2	1 hora	260 µg/m³	240 µg/m³	220 µg/m³	200 µg/m³
	Anual	60 µg/m³	50 µg/m³	45 µg/m³	40 µg/m³
Ozônio - O3	8 horas	140 µg/m³	130 µg/m³	120 µg/m³	100 µg/m³
Fumaça	24 horas	120 µg/m³	100 µg/m³	75 µg/m³	50 µg/m³
	Anual	40 µg/m³	35 µg/m³	30 µg/m³	20 µg/m³
Monóxido de Carbono - CO	8 horas	-	-	-	9 ppm
Partículas Totais em Suspensão - PTS	24 horas	-	-	-	240 µg/m³
	Anual	-	-	-	80 µg/m³
Chumbo - Pb	Anual	-	-	-	0,5 µg/m³

Fonte: Adaptado do Anexo I da Resolução Conama Nº 491 de 2018

Também se estabeleceu níveis de atenção, de alerta e de emergência, no intuito que os órgãos ambientais locais elaborem medidas preventivas, com o objetivo de evitar graves e iminentes riscos à saúde da população. Sendo os níveis declarados no Anexo III da mesma resolução.

Tabela 3 – Níveis Criticos para Poluentes

Nível	Poluentes e Concentrações					
		Material Particulado				
	SO2 µg/m³ (média de 24h)	MP10 µg/m³ (média de 24h)	MP2,5 µg/m³ (média de 24h)	CO ppm (média móvel de 8h)	O3 µg/m³ (média móvel de 8h)	NO2 µg/m³ (média móvel de 1h)
Atenção	800	250	125	15	200	1130
Alerta	1600	420	210	30	400	2260
Emergência	2100	500	250	40	600	3000

Fonte: Adaptado do Anexo III da Resolução Conama Nº 491 de 2018

A resolução do Conama Nº491 também define o Índice de Qualidade do Ar (IQAR) como o valor adimensional que relaciona as concentrações dos poluentes monitorados aos possíveis efeitos à saúde. O Índice de Qualidade do Ar - IQAR é usado essencialmente para informar à população as condições da qualidade do ar de forma simples e direta. Para calculá-lo, utiliza-se a equação (1).

$$IQAr = I_{ini} + \frac{I_{fin} - I_{ini}}{C_{fin} - C_{ini}} * (C - C_{ini}) \qquad (1)$$

Na qual:

I_{ini} = valor do índice que corresponde à concentração inicial da faixa.

I_{fin} = valor do índice que corresponde à concentração final da faixa.

C_{ini} = concentração inicial da faixa onde se localiza a concentração medida.

C_{fin} = concentração final da faixa onde se localiza a concentração medida.

C = concentração medida do poluente.

No entanto, faz-se necessário pontuar que a Resolução CONAMA Nº 491 de 2018 define apenas o índice para a qualidade boa. Assim sendo, cabe órgãos ambientais

estaduais e distritais competentes definir os demais índices. A Companhia Ambiental do Estado de São Paulo (CETESB), adota os índices apresentados na Tabela 4.

Tabela 4 – Estrutura do índice de qualidade do ar adotado pela CETESB

Qualidade	Índice	MP10 µg/m³ (média de 24h)	MP2,5 µg/m³ (média de 24h)	O3 µg/m³ (média móvel de 8h)	CO ppm (média móvel de 8h)	NO2 µg/m³ (média móvel de 1h)	SO2 µg/m³ (média móvel de 1h)
N1 - Boa	0-40	0-50	0-25	0-100	0-9	0-200	0-20
N2 - Moderada	41-80	>50-100	>25-50	>100-130	>9-11	>200-240	>20-40
N3 - Ruim	81-120	>100-150	>50-75	>130-160	>11-13	>240-320	>40-365
N4 - Muito Ruim	121-200	>150-250	>75-125	>160-200	>13-15	>320-1130	>365-800
N5 - Péssima	>200	>250	>125	>200	>15	>1130	>800

Fonte: Adaptado do site da CETESB (cetesb.sp.gov.br) (2021)

Embora se avalie a qualidade do ar para todos os poluentes, a classificação utiliza o critério da precaução e é determinada pelo maior índice calculado, ou seja, o pior caso (CETESB, 2021). Para efeito de divulgação, relaciona-se a classificação da qualidade aos efeitos à saúde, de acordo com o Quadro 5.

Quadro 5 – Qualidade do ar e efeito à saúde

Qualidade	Índice	Significado
N1 - Boa	0-40	-
N2 - Moderada	41-80	Pessoas de grupos sensíveis (crianças, idosos e pessoas com doenças respiratórias e cardíacas) podem apresentar sintomas como tosse seca e cansaço. A população, em geral, não é afetada.
N3 - Ruim	81-120	Toda a população pode apresentar sintomas como tosse seca, cansaço, ardor nos olhos, nariz e garganta. Pessoas de grupos sensíveis podem apresentar efeitos mais sérios na saúde.
N4 - Muito Ruim	121-200	Toda a população pode apresentar agravamento dos sintomas como tosse seca, cansaço, ardor nos olhos, nariz e garganta e ainda falta de ar e respiração ofegante. Pessoas de grupos sensíveis podem apresentar efeitos ainda mais grave.
N5 - Péssima	>200	Toda a população pode apresentar sérios riscos de manifestações de doenças respiratórias e cardiovasculares. Aumento de mortes prematuras em pessoas de grupos sensíveis.

Fonte: Adaptado do site da CETESB (cetesb.sp.gov.br) (2021).

3.4 Gases selecionados e seus efeitos à saúde

Neste trabalho, selecionou-se para monitoramento o Monóxido de Carbono e os Materiais Particulados MP_{10} e $MP_{2,5}$, além do Dióxido de Carbono e do Composto Orgânico Volátil Total Equivalente - como indicadores para a renovação de ar em ambiente fechado - e do Gás Liquefeito do Petróleo (GLP) - por ser um gás inflamável.

3.4.1 Monóxido de Carbono (CO)

Gerado na queima de combustíveis de origem orgânica, é um gás incolor e inodoro. Pode ser encontrado em maiores concentrações em centros urbanos, devido à alta intensidade de uso de veículos automotores (CETESB, 2020).

O monóxido de carbono, quando presente na corrente sanguínea - por possuir uma alta afinidade pela hemoglobina - forma a carboxihemoglobina, que impede o

transporte de oxigênio no sangue, e, com isso, afeta principalmente os sistemas nervoso central e cardiovascular, podendo causar morte (SCHIRMER, 2011).

O CONAMA adotou como nível de concentração limite de CO o valor definido pela Organização Mundial de Saúde (OMS), de 9 ppm, como máxima média móvel obtida no dia em um período de referência de 8 horas.

3.4.2 Gás Liquefeito de Petróleo (GLP)

É uma mistura de gases obtida do refino do petróleo. Na forma gasosa, é incolor e inodoro - por este motivo, o composto t-butil mercaptana é misturado a ele para ser de fácil detecção caso haja vazamentos. Devido à sua densidade, o GLP desloca o ar atmosférico, podendo diminuir a concentração de oxigênio respirável e causar falta de ar, fadiga, diminuição da visão, dor de cabeça, decréscimo de atividade motora, coma e morte (CETESB, 2020).

De acordo com o manual técnico para o gás GLP da Petrobrás, a combustão deste gás pode ocorrer em um ambiente que atinja a concentração de 2%, desde que se tenha uma fonte de ignição (PETROBRAS, 2019).

3.4.3 Materiais Particulados

Podem variar em tamanho, concentração, formato, área de superfície e composição química, dependendo da fonte de emissão. Por este motivo, são classificados de acordo com seu diâmetro em partículas totais de suspensão: partículas com 30 µm, inferiores a 10 µm (MP_{10}), inferior a 2,5 µm ($MP_{2,5}$) e partículas de tamanho inferior a 10 nm ($MP_{0,1}$) (ARBEX, 2012).

Afeta principalmente crianças, idosos e portadores de doenças pré-existentes, podendo causar inflamações respiratórias, obstrução de vias aéreas e atingir o sistema circulatório.

3.4.4 Dióxido de Carbono

De acordo com o Observatório Mauna Loa, no Havaí, atualmente a concentração de dióxido de carbono na atmosfera é de 417 ppm (NASA, 2021). No entanto, para ambientes internos, a concentração de dióxido de carbono é maior, devido ao processo natural de produção deste gás pela respiração humana e à baixa taxa de ventilação e renovação do ar em ambientes fechados (SATISH *et al*., 2012).

A Resolução Nº 9/2003 da Agência Nacional de Vigilância Sanitária (ANVISA) estabelece critérios sobre a qualidade do ar interior em ambientes climatizados artificialmente. Um dos critérios nela adotados é o uso do CO_2 como indicador de renovação de ar externo. Para o conforto e bem-estar, é recomendável que a concentração de CO_2 não ultrapasse 1000 ppm para ambientes internos.

3.4.5 Compostos Orgânicos Voláteis

São gases resultantes da queima incompleta e da evaporação de combustíveis e outros produtos orgânicos. Muitos desses compostos são precursores do ozônio e alguns podem causar efeito adversos à saúde (CETESB, 2021). No entanto, no Brasil não existe regulamentação que adote uma concentração limite de COVs no ar.

Diante do exposto, levando em consideração os poluentes avaliados neste trabalho, o Quadro 5 apresenta um resumo dos valores adotados para emissão dos gases a serem estudados.

Tabela 6 - Padrão de qualidade do ar adotado.

Poluente Atmosférico	Período de Referência	Concentração Aceitável Adotada	Unidade de medida	Referência
Monóxido de carbono	8 horas	9	ppm	CONAMA
Partículas totais em Suspensão (MP10 + MP2,5)	24 horas	240	µg/m³	CONAMA
Dióxido de Carbono	5 minutos	1000	ppm	ANVISA
GLP	5 minutos	2000	ppm	PETROBRÁS
COVs	5 minutos	-	ppb	-

Fonte: Autor (2021)

3.5 Placa de Desenvolvimento

São CPUs de pequeno porte, com todos os componentes necessários para o controle de um processo por meio de linguagens de programáveis. Para isso, tem-se como requisito apenas uma fonte de alimentação para seu funcionamento básico (SOUZA, 2005).

- Espressif ESP32 Lilygo TTGO T-Display

A Lilygo TTGO T-Display é uma placa de desenvolvimento que possui um microprocessador ESP32, produzido pela Espressif e dotado com funcionalidade Wi-Fi e Bluetooth. Essa placa conta com uma tela de cristal líquido embutida, um Header com 24 GPIOs (podendo ser programados utilizando a linguagem de programação C/C++ ou Lua) e uma USB tipo C usada para alimentação. O Quadro 3 e Figura 1 apresentam as características e foto da placa LilyGo TTGO Esp32.

Tabela 7 - Especificações do LilyGo TTGO Esp32

LilyGO TTGO T-display	
Microprocessador	ESP32 Xtensa dual-core
Tensão de Alimentação	3.3 - 5V
Pinos I/O Digitais	13
Pinos Input Analógicos	11
Memória RAM	520 kb
Frequência do processador	240 Mhz
Wireless	802.11 b/g/n
Saída Gráfica	Possui Display integrado 1.4"
Bluetooth	BLE Standart
Conector Ethernet	Não possui
Preço	R$ 71,39*

Fonte: Site LilyGO e Site Americanas (LilyGO, 2021, www.lilygo.com; Americanas, 2021, www.americanas.com.br) *Dólar cotado em R$5,52

Figura 1 - LilyGO TTGO ESP32

Fonte: Site LilyGO (LilyGO, 2021, www.lilygo.com)

3.6 Sensores

3.6.1 Família de Sensores MQ-X:

Fabricados pela empresa Chinesa Hanwei Eletronics, destacam-se por serem sensores de baixo custo, por possuir uma longa vida útil e por fornecerem uma resposta rápida. Em contrapartida, faz-se necessário que estejam sempre em uma temperatura ótima de funcionamento, estabelecida pela própria fabricante, para maior precisão na detecção do gás. Para tal, a fabricante recomenda que o equipamento fique ligado por um tempo mínimo antes de qualquer medição; dependendo do sensor, este pode chegar a 48 horas.

Esses sensores são compostos por um tubo de cerâmica micro AL203, uma camada sensitiva de óxido de estanho, um eletrodo de medição e um aquecedor, que são fixados em uma crosta feita de plástico e rede de aço inoxidável (datasheet do sensor MQ-7), a descrição do material das partes dos sensores MQ-X estão na Tabela 4.

Quadro 8 - Componentes dos Sensores MQ-X

Partes	Material
Camada Sensitiva	SnO2
Eletrodo	Ouro
Linha do Eletrodo	Platina
Bobina do Aquecedor	Liga de Níquel-Cromo
Tudo de Cerâmica	Oxido de Alumínio
Trançado Anti-explosão	Aço inoxidável
Anel de Fixação	Cobre revestido com Níquel
Base de Resina	Baquelita
Pinos	Cobre revestido com Níquel

Fonte: Datasheet do sensor MQ-7 (2021)

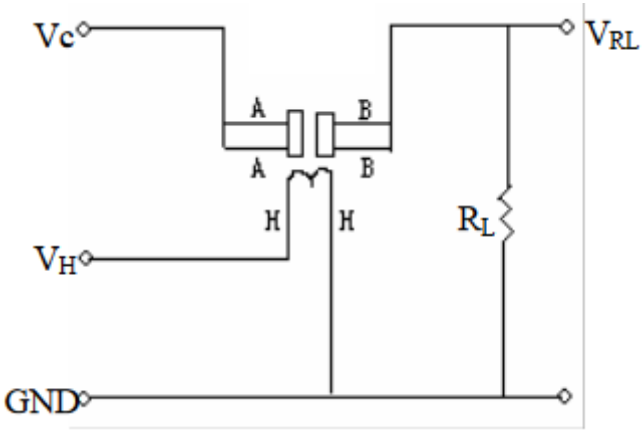
Cada sensor possui uma curva característica de sensibilidade, por meio da qual é possível determinar a concentração do gás medido por sua resistência. Essas curvas diferem entre os modelos dos sensores e estão inclusas nos seus respectivos *datasheets* (documentação eletrônica), disponibilizados pelo próprio fabricante. Pode-se descobrir a resistência do sensor a partir da relação:

$$\frac{R_s}{R_L} = \frac{V_c - V_{RL}}{V_{RL}} \qquad (2)$$

Na qual R_s é a Resistencia do sensor, R_L é a Resistencia da carga, V_c é a Tensão de alimentação e V_{RL}é a Tensão de resistência da carga.

O circuito do sensor se baseia em um divisor de tensão: sua camada sensitiva emite uma resistência proporcional à concentração do determinado gás em contato e - a partir de uma relação entre a resistência de carga, a tensão de entrada e a tensão da carga - é possível determinar a resistência do sensor e estimar a concentração de gás captada por ele (Figura 2; Equação 1).

Figura 2 - Circuito Elétrico dos sensores MQ-X



Fonte: *Datasheet* do sensor MQ-4 (2021)

Para este projeto, utilizou-se os sensores da família MQ identificados no Quadro 9.

Quadro 9 - Sensores da Família MQ utilizados neste trabalho

Sensores Escolhidos		
Sensor	Gás sensível	Preço
MQ-5	Gás Liquefeito de Petróleo	R$ 12,08
MQ-7	Monóxido de Carbono	R$ 23,90

Fonte: Autor (2021)

3.6.1.1 Sensor MQ – 5

O sensor MQ-5 apresenta sensibilidade para o GLP, com uma maior capacidade de detecção para GLP; deste modo, no presente trabalho, utilizou-se o sensor para medi-lo. As condições ambientais que otimizam a eficiência de funcionamento desse sensor são representadas na Tabela 10.

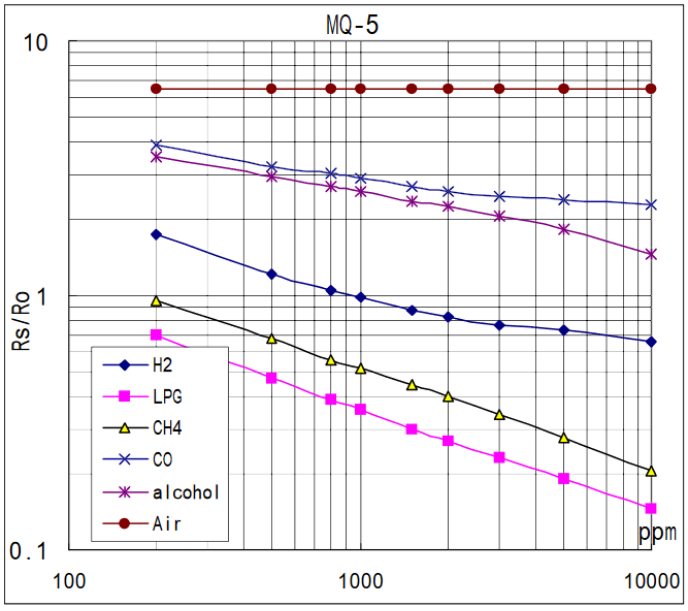
Tabela 10 - Condições ambientais de uso do sensor MQ-5

MQ-5	
Parâmetros Ambientais	Condições técnicas
Temperatura	-10°C a 50°C
Umidade Relativa	Abaixo de 95%
Concentração de Oxigênio	21%
Tempo de Aquecimento	24h

Fonte: Adaptado do *datasheet* do sensor MQ-5 (2021)

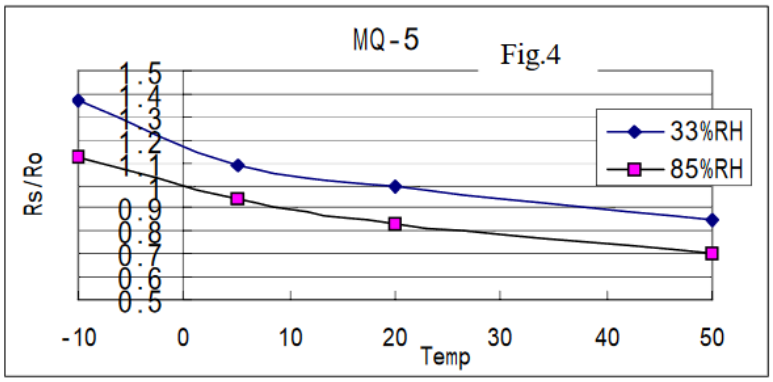
Seu gráfico das curvas de resposta para os gases detectáveis (Figura 3) e a curva de influência em relação a temperatura e umidade (Figura 4) são disponibilizados pela empresa, estando em função de $\frac{R_S}{R_0}$ e tendo a concentração dos gases dada em partes por milhão (ppm). Nele, R_S é a resistência do sensor em resposta à determinada concentração do gás e R_0 é a resistência do sensor em resposta a um ambiente contendo uma concentração de 1000 ppm de hidrogênio em ar limpo.

Figura 3 - Curvas características de resposta aos gases do sensor MQ-5



Fonte: *Datasheet* do sensor MQ-5 (2021)

Figura 4 - Curva da influência em relação a temperatura e umidade no sensor MQ-5



Fonte: *Datasheet* do sensor MQ-5 (2021)

3.6.1.2 Sensor MQ – 7

Apresenta sensibilidade para o gás monóxido de carbono, como constatado pela alta variação da resistência para os gases monóxido de carbono e hidrogênio (Figura 5). A eficiência do sensor em questão depende das seguintes condições ambientes estão indicados na Tabela 11.

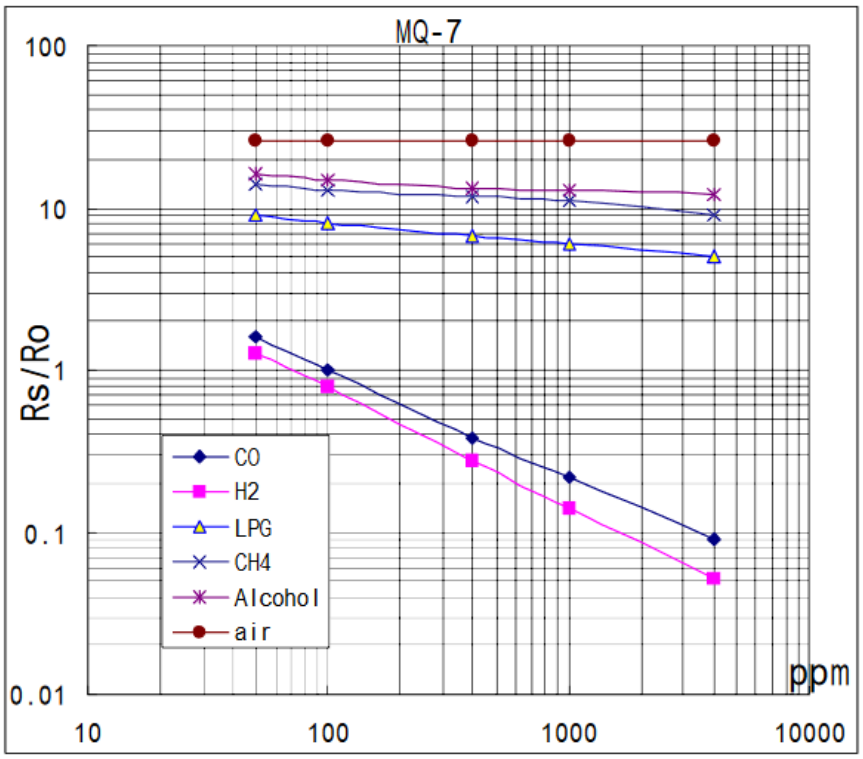
Tabela 11 - Condições ambientais de uso do sensor MQ-7

MQ-7	
Parâmetros Ambientais	Condições técnicas
Temperatura	-20°C a 50°C
Umidade Relativa	Abaixo de 95%
Concentração de Oxigênio	21%
Tempo de aquecimento	24 horas

Fonte: Adaptado do *Datasheet* do sensor MQ-7 (2021)

Na Figura 5, observa-se as curvas características de resposta aos gases detectáveis pelo sensor. O gráfico é dado em função de $\frac{R_s}{R_0}$ e a concentração dos gases é dada em partes por milhão (ppm), sendo R_s a resistência do sensor em resposta à determinada concentração do gás e R_0 a resistência do sensor em resposta a um ambiente contendo uma concentração de 1000 ppm de hidrogênio e limpo em relação aos gases que influenciam no sensor.

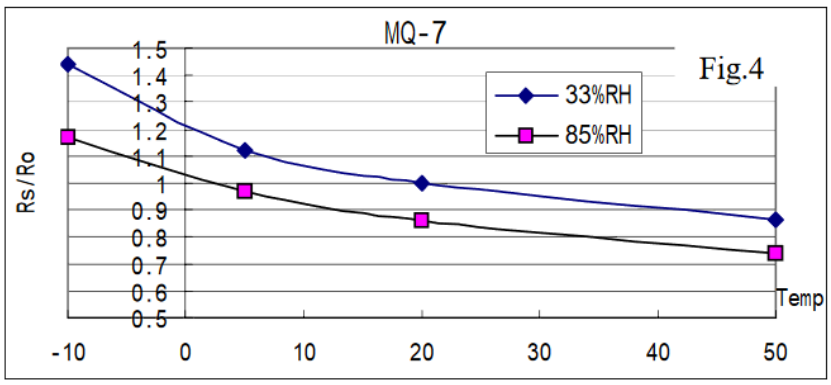
Figura 5 - Curvas características de resposta aos gases do sensor MQ-7



Fonte: *Datasheet* do sensor MQ-7 (2021)

Na Figura 6, tem-se o gráfico da influência da temperatura e da umidade relativa:

Figura 6 - Curva da influência em relação a temperatura e umidade no sensor MQ-7



Fonte: *Datasheet* do sensor MQ-7 (2021)

3.6.2 Sensor de Gás CCS811

É um sensor digital de ultra-baixa potência, operando com tensão de 3.3 volts. Ele integra um sensor de gás de óxido de metal utilizado para detectar muitos compostos orgânicos voláteis totais (operando em uma faixa de 0 a 1187 ppb) ou níveis equivalentes

de dióxido de carbono (operando em uma faixa de 400 a 8192 ppm), especificamente para monitoramento da qualidade do ar interno (SPARKFUN, 2016). Seu preço é foi de R$109,91 na cotação do dólar de R$5,52. A Tabela 12 apresenta as condições ambientais de operações do sensor e a Figura 7 apresenta uma imagem do sensor.

Tabela 12 – Condições ambientais para uso do sensor de CO_2 e COV's

Sensor CCS811	
Parâmetros Ambientais	Condições técnicas
Temperatura	-5°C a 50°C
Umidade Relativa	10 a 95%

Fonte: Adaptado do *datasheet* do sensor CCS811

Figura 7 – Sensor de gás CCS811

Fonte: Adaptado do site Americanas (www.americanas.com.br) (2021)

3.6.3 Sensor de Poeira GP2Y1010AU0F:

Produzido pela empresa SHARP, e com um preço de R$49,84 na cotação do dólar a R$5,52, pode ser utilizado para detecção de fumaça e de partículas finas presente no ar. Seu funcionamento se dá a partir de um sistema contendo um diodo emissor de luz infravermelha e de um fototransistor, arranjados diagonalmente para detectar a luz refletida na poeira (OLIVEIRA, 2016). A Tabela 13 apresenta as condições ambientais de operação desse sensor e a Figura 8 apresenta uma imagem do sensor.

Tabela 13 - Condições ambientais para uso do sensor de poeira

Sensor de poeira	
Parâmetros Ambientais	Condições técnicas
Temperatura	-10°C a 60°C
Umidade Relativa	Abaixo de 90%

Fonte: Adaptado do *datasheet* do sensor GP2Y1010AU0F (2021)

Figura 8 - Sensor de Poeira GP2Y1010AU0F

Fonte: Adaptado do site Aliexpress (www.aliexpress.com) (2021)

3.6.4 Sensor de Temperatura e Umidade DHT22:

Consiste em um sensor de temperatura e de umidade, que oferece como vantagens um baixo custo, com um preço de R$ 25,09 na cotação do dólar de R$5,52, e uma boa precisão, além de uma rápida resposta e saída digital. Apresenta-se as características desse sensor na Tabela 14, enquanto a Figura 9 apresenta a imagem do sensor de temperatura e umidade.

Tabela 14 - Informações do sensor de temperatura e umidade

Sensor DHT22	
Tensão de entrada	3,3V a 6,0V
Tipo de saída	Sinal Digital
Elemento de detecção	Capacitor de polímero
Alcance de operação	Umidade: 0 a 100% Temperatura: -40°C a 80°C
Precisão	Umidade: +/- 2% Temperatura: +/- 0,5°C
Resolução	Umidade: 0,1% Temperatura: 0,1°C
Período de Detecção	2 segundos

Fonte: Adaptado do *datasheet* do sensor DHT22 (2021)

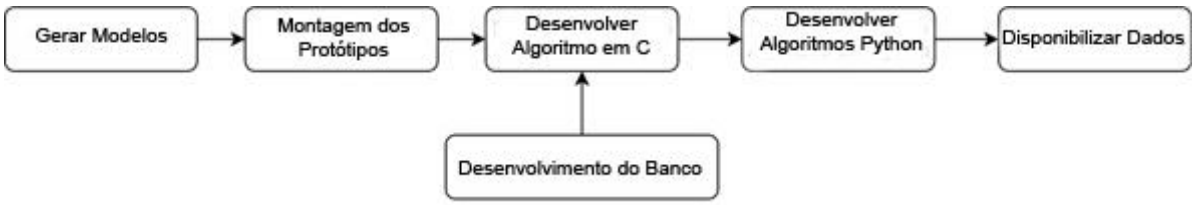
Figura 9 - Sensor DHT22

Fonte: Adaptado do site Aliexpress (www.aliexpress.com) (2020)

4. METODOLOGIA

A Figura 10 mostra as principais etapas metodológicas deste trabalho. Detalhar-se-á cada uma delas nesta seção.

Figura 10 - Metodologia empregada no presente estudo.



Fonte: Autor (2021).

4.1 Gerando Modelos para Sensores de Gás

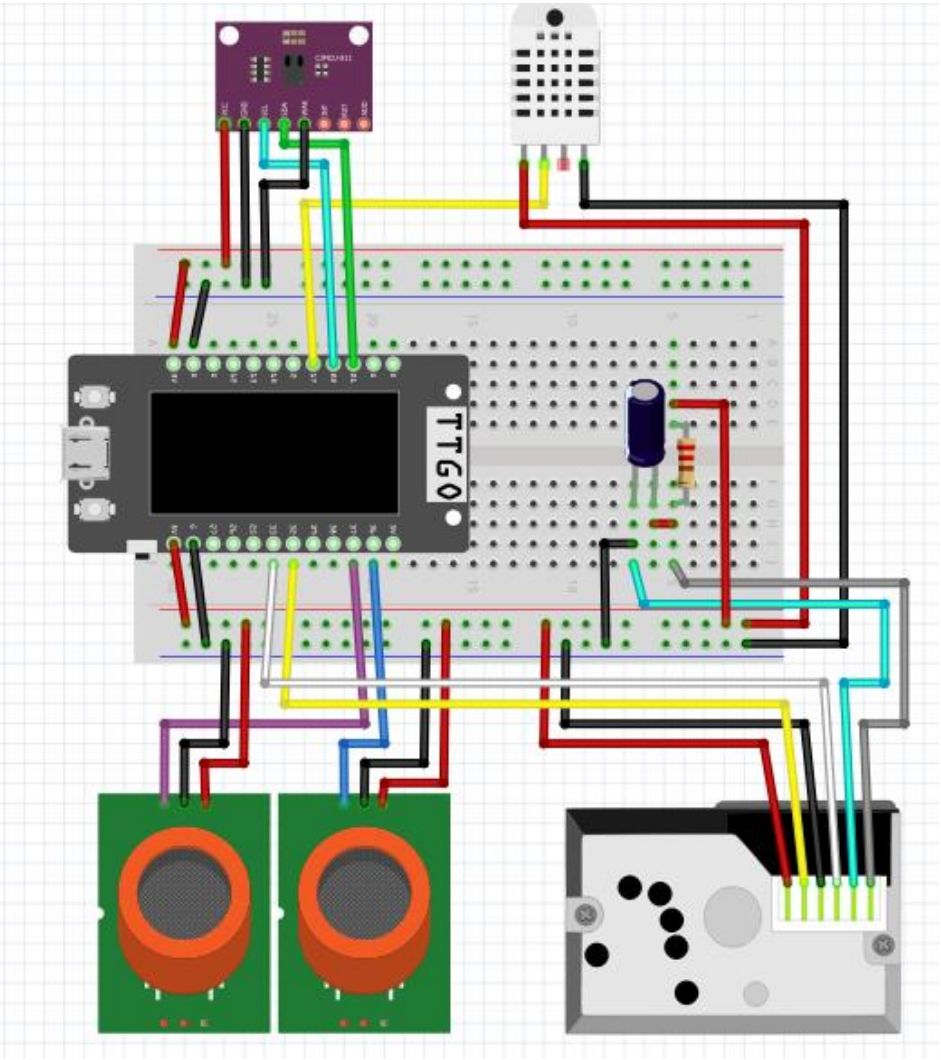
A partir das curvas contidas no *datasheet* de cada sensor, coletou-se vários pontos da curva, que serviram como dados de entrada (*input)* para um algoritmo de ajuste de curva. Este determinou qual a função exponencial que mais se aproximou da curva característica do determinado sensor e, por conseguinte, recomendada para gerar as equações das curvas a serem utilizadas no algoritmo de cada protótipo.

4.2 Montagem dos Protótipos

A montagem do sistema usa o LilyGo TTGO ESP32 (Figura 11), que já dispunhas de uma tela de 0.93”, de modo que não foi necessária a aquisição de uma tela como mostrador.

Ademais, como o ESP32 já dispunha de comunicação wireless, com programação mais elaborada pôde-se utilizar desta para dispor as informações em outro dispositivo, como smartphone e/ou computadores.

Figura 11 - Desenho do circuito utilizando um ESP32 TTGO T-Display



Fonte: Autor (2021)

Todo o projeto teve um custo aproximado de R$ 300,00 na cotação do dólar de R$5,52, valor muito abaixo dos detectores multigases vendidos no mercado.

'

4.3 Monitoramento dos gases

O monitoramento aconteceu no município de Pilar, em Alagoas, região metropolitana de Maceió, no bairro do Torrão, Rua Nicolau Mota. A residência na qual foi feito o monitoramento está a aproximadamente trinta metros da margem da lagoa Manguaba (Figura 12). O vento predominante no local é o Sudeste com rajadas frescas vindas da direção da lagoa.

Figura 12 – Localização do monitoramento

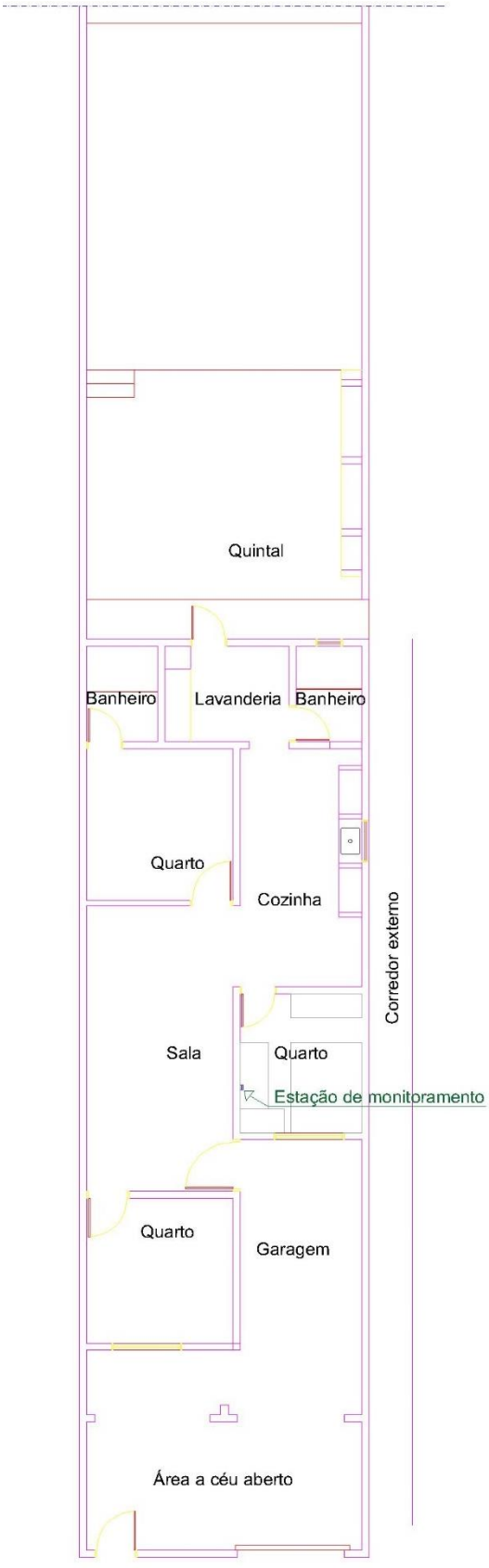
Fonte: Autor (2021)

A rua onde está localizada a residência tem um baixo volume de tráfego, no entanto, no período de monitoramento, a aproximadamente dez metros de distância da residência, houve uma residência em obra na qual pode ter ocorrido um aumento na concentração de partículas suspensas do local. Outro fator externo que pode ter aumentado a concentração de partículas suspensas foram o exercício do trabalho das Margaridas, que diariamente, aproximadamente às 16:00 horas, varriam toda a rua, o que levantava uma camada de poeira.

O ambiente monitorado foi um dos quartos da residência. O quarto possui uma janela com vista para a garagem da residência que, por ser uma garagem aberta, recebe um fluxo de ar diretamente da direção da rua. O quarto, através da porta, tem ligação direta com a cozinha da residência, cozinha esta que possui um grande fluxo de ar devido a uma janela que dá acesso a um corredor que afunila o vento que provém da lagoa. Devido a este ambiente apresentado, o fluxo do ar variou bastante no interior do quarto, por vezes o fluxo do ar adentrando pela janela e saindo pela porta em direção à cozinha, por vezes fazendo o oposto.

O quarto é mantido predominantemente com a janela e a porta aberta, fechando ambos apenas pela noite momentos antes do seu usuário dormir e abrindo-os assim que o usuário acorda. O quarto possui uma escrivaninha com uma pequena biblioteca e dois computadores, um guarda-roupas e uma cama (Figura 13), sendo feito faxinas semanalmente. O equipamento desenvolvido passou a maior parte do tempo de monitoramento na parede, entre a porta e a janela, a uma altura de 1,5 metros do chão.

Figura 13 – Planta da residência



Fonte: Autor (2021)

4.4 Desenvolvimento dos Algoritmos (C e Banco de Dados)

Montou-se o protótipo na medida em que se desenvolveu os algoritmos. Em paralelo, construiu-se um banco de dados, com uma tabela que recebe unicamente de uma estação de monitoramento, constando horário de medição e valor medido para cada sensor.

4.5 Disponibilizando os Dados

Utilizou-se a pequena tela OLED do microcontrolador para visualizar os dados instantâneos com um código de cores que remete à qualidade do ar de acordo com o IQAR do valor medido (Quadro 15).

Quadro 15 – Indicação das cores na tela

Sensor de poeira	
Qualidade / Nível Crítico	Cor
N1 - Boa	Fonte Verde
N2 – Moderada	Fonte Amarelo
N3 - Ruim	Fonte Laranja
N4 – Muito Ruim	Fonte Vermelha
N5 - Péssima	Fonte Púrpura
Normal	Fundo Preto
Atenção	Fundo Amarelo
Alerta	Fundo Laranja
Emergência	Fundo Vermelho

Fonte: Autor (2021)

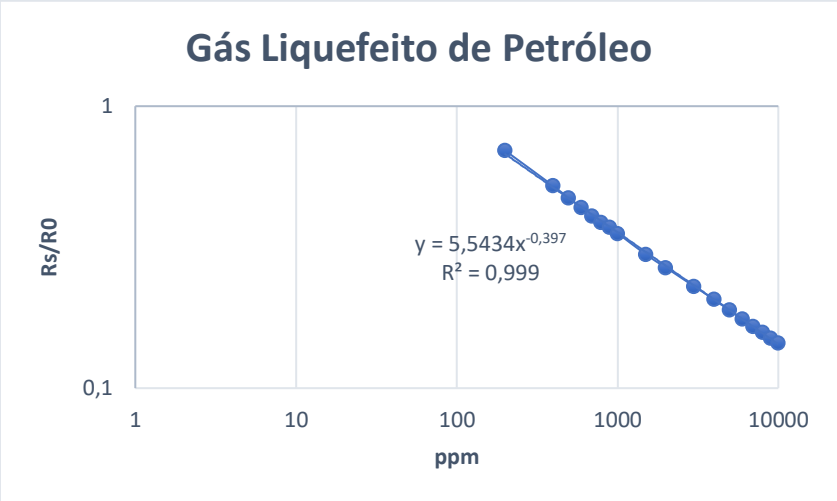
As informações coletadas foram armazenadas no banco de dados, no qual o usuário tem a opção de acessar um arquivo tipo .csv, contendo todos os dados e todas as tabelas resultantes das medições de cada sensor. Também se desenvolveu formas de visualização gráfica para cada gás monitorado, sendo elas: histórico das médias de leitura a cada cinco minutos, média móvel como requerido na Resolução do Conama e histórico do índice de IQAR para cada poluente como requerido na Resolução do Conama.

5. RESULTADOS OBTIDOS

5.1 Gerando Modelos

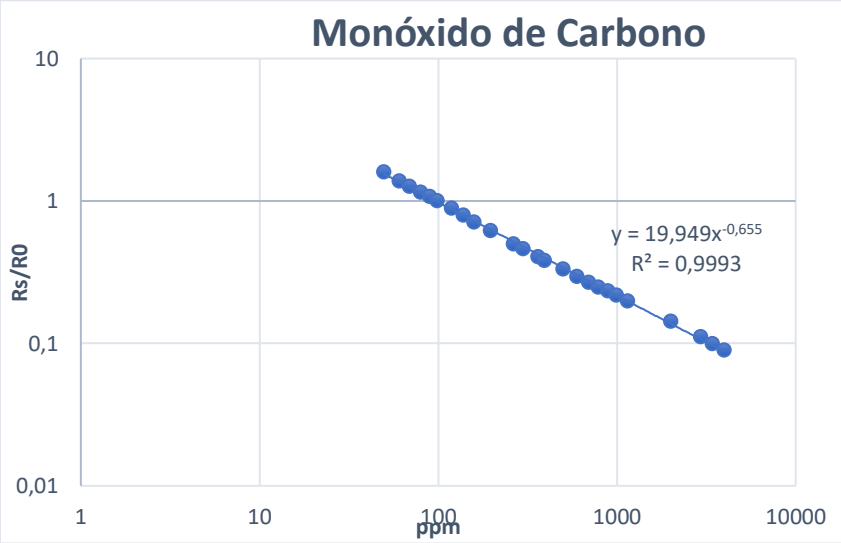
Extraiu-se vários pontos das curvas características em resposta à concentração dos gases contidas no *datasheet* dos sensores MQ-5 e MQ-7. Para isso, utilizou-se a aplicação web *WebPlotDigitizer*. A partir dos pontos, criou-se uma curva de tendência para o gás liquefeito de petróleo utilizando o sensor MQ-5 (Figura 14); para o monóxido de carbono, utilizando o sensor MQ-5 (Figura 15). As equações das curvas se encontram na Tabela 16.

Figura 14 – Curva gerada a partir dos pontos coletados do sensor MQ-5 para GLP



Fonte: Autor (2021)

Figura 15 – Curva gerada a partir dos pontos coletados do sensor MQ-7 para Monóxido de Carbono



Fonte: Autor (2021)

Tabela 16 – Equações Geradas para cada gás

Sensor	Gás	Equações Geradas	R^2
MQ-5	Gás Liquefeito de Petróleo	$\frac{Rs}{R0} = 5{,}5434\ x^{-0{,}397}$	0,999
MQ-6	Monóxido de Carbono	$\frac{Rs}{R0} = 19{,}949\ x^{-0{,}655}$	0,9993

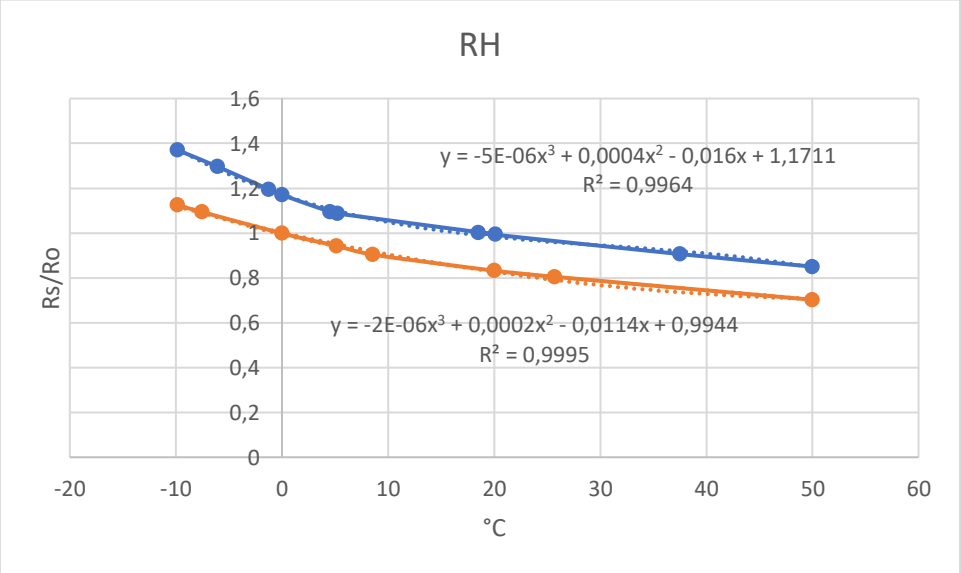
Fonte: Autor (2021)

Na qual $\frac{Rs}{R0}$ é o valor processado a partir do sinal do sensor e o x o valor da concentração do gás em partes por milhão.

Também foi necessário extrair o valor para o ar sem a presença dos gases, tendo $\frac{Rs}{R0}$ constantes em ambos os sensores, para critério de calibração. Obteve-se para o MQ-5 o valor de 6,4754 e para o MQ-7 o valor de 361,286.

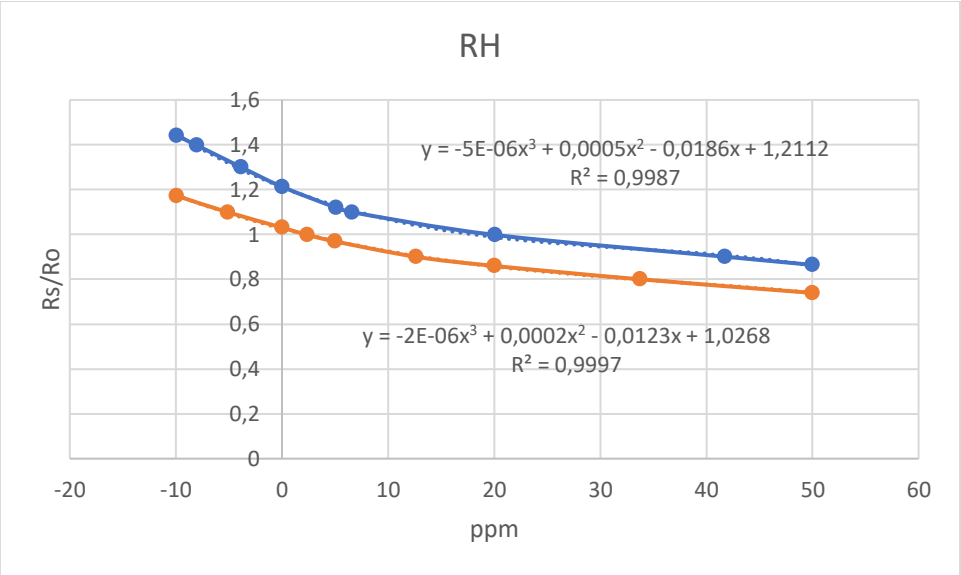
Usando o mesmo processo, criou-se as curvas para ajuste de valor medido perante a variância da umidade e da temperatura (Figura 16 e Figura 17).

Figura 16 – Dependência do sensor MQ-5 sobre temperatura e umidade



Fonte: Autor (2021)

Figura 17 - Dependência do sensor MQ-7 sobre temperatura e umidade



Fonte: Autor (2021)

Analogamente ao que foi descrito anteriormente, gerou-se suas equações (Tabela 17):

Tabela 17 - Equações de ajustes para umidade e temperatura geradas para cada gás

Sensor	Gás	Umidade	Equações Geradas	R^2
MQ-5	Gás Liquefeito de Petróleo	33%	$\frac{Rs}{R0} = -0{,}000005 * T^3 + -0{,}0004 * T^2 - 0{,}016T + 1{,}1711$	0,9964
		85%	$\frac{Rs}{R0} = -0{,}000002 * T^3 + -0{,}0002 * T^2 - 0{,}0114T + 0{,}9944$	0,9995
MQ-6	Monóxido de Carbono	33%	$\frac{Rs}{R0} = -0{,}000005 * T^3 + -0{,}0005 * T^2 - 0{,}0186T + 1{,}2112$	0,9987
		85%	$\frac{Rs}{R0} = -0{,}000002 * T^3 + -0{,}0002 * T^2 - 0{,}0123T + 1{,}0268$	0,9997

Fonte: Autor (2021) * T representa a temperatura.

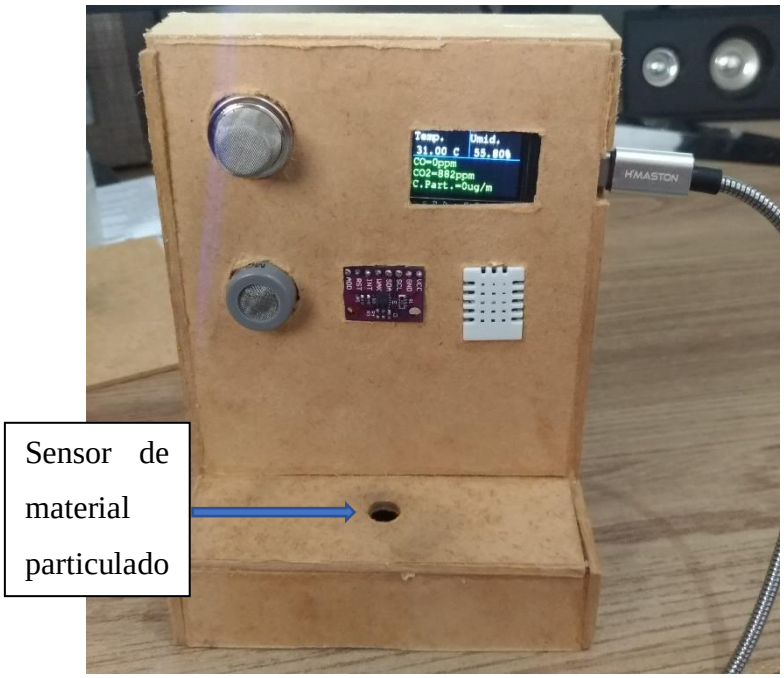
5.2 Montagem do Protótipo

Montou-se o protótipo utilizando fios jumpers como conectores. Ademais, produziu-se uma caixa com placas de madeira para proteção do protótipo, de maneira que os sensores

ficaram em um posicionamento apropriado para o monitoramento (Figura 18). Uma fonte de 5 V e 2.0 A, ligada diretamente no microcontrolador, alimenta todo o circuito.

Desenvolveu-se o circuito em paralelo com os algoritmos; para isso, adicionou-se um sensor por vez e então desenvolveu-se as funções necessárias ao seu funcionamento adequado.

Figura 18 – Protótipo montado



Fonte: Autor (2021)

5.3 Desenvolvimento dos Algoritmo

Foram utilizadas de três linguagens computacionais diferente, são elas: C++ para o algoritmo de controle e visualização de dados instantâneos, SQL para o banco de dados e Python para visualização de dados processados e disponibilidade dos dados.

5.4 Algoritmo de Controle

Desenvolveu-se o código fonte do microcontrolador na IDE do Arduino (versão 1.8.13), configurada para o uso de microcontroladores NodeMCU.

Figura 19 – Código fonte Parte 1

TCC_ESP_TTGO

```
/*Chamando Bibliotecas*/
#include "DHTesp.h"
#include "Adafruit_CCS811.h"
#include "GP2YDustSensor.h"
#include <TFT_eSPI.h> // Hardware-specific library
#include <SPI.h>
#include <Wire.h>
#include <math.h>
#include <WiFi.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>

/*Definindo pinos*/
#define pin_DHT 17 //Usando o DHT22
#define pin_MQ7 36 // Usando o sensor MQ7 - CO
#define pin_MQ5 37 // Usando o sensor MQ5 - GLP
#define SHARP_LED_PIN 33 // Usando o sensor de pariculado
#define SHARP_VO_PIN  32 // Usando o sensor de pariculado

/*Definindo constantes*/
#define RL_MQ7 10000
#define R0_MQ7 1301.57
#define RL_MQ5 20000
#define R0_MQ5 17280.05
```

Fonte: Autor (2021)

Na Figura 19 mostra-se o início do código fonte, no qual se importa as bibliotecas necessárias para o funcionamento de todo o projeto, dispostas na Tabela 18.

Na Figura 19 também se definiu os valores de constantes utilizadas no decorrer do código, referentes aos pinos utilizados do microcontrolador, aos valores da resistência de carga indicado no *datasheet* do respectivo sensor de gás e ao valor R0, que é a resistência do sensor em ambiente sem a presença do gás. Este valor foi coletado em uma calibração em ambiente aberto.

Na Figura 20, mostra-se a declaração de todas as variáveis globais e objetos das classes que estão sendo utilizadas em todo código. Já na Figura 21, tem-se uma função que verifica se a temperatura e a umidade estão dentro dos critérios e ajusta o valor da razão $\frac{Rs}{R0}$ de acordo com as equações da Tabela 17.

Tabela 18 – Bibliotecas utilizadas no código fonte

Biblioteca	Descrição
DHTesp	Criada por Bernd Giesecke, uma biblioteca para leitura de sensores de temperatura e umidade da família DHT para uso em ESP32 e ESP8266 (GITHUB, 2021a).
Sharp GP2Y* Dust Sensor	Criada por Lucian Sabo, é uma biblioteca para o uso de sensores de poeira e é compatível com microcontroladores NodeMCU e Arduino (GITHUB, 2019).
TFT_eSPI e SPI	Bibliotecas de gráfico e suporte para vários modelos de display (GITHUB, 2021b).
Math	Biblioteca para uso de funções matemáticas.
WiFi	É uma biblioteca que permite que um microcontrolador se conecte à internet, funcional para Arduino utilizando modulo WiFi e controladores ESP32.
Adafruit_CCS811	Biblioteca para controle do sensor CCS811. (GITHUB, 2021c).
MySQL_Connector_Arduino	É uma biblioteca para comunicação de microcontroladores com bancos de dados MySQL (GITHUB, 2020).

Fonte: Autor (2021)

Figura 20 – Código fonte Parte 2

```
/*Variaveis Globais */
String nome = "primeiro prototipo";
float umidade, temperatura, mp;
int co, co2, cov, particulado, glp;
int coMedio, coSoma, contCO, mpMedio, mpSoma, contMP, glpMedio, glpSoma, contGLP = 0;
int covMedio,covSoma, contCOV, co2Medio, co2Soma, contCO2 = 0;
int leitura_analogica_MQ7, leitura_analogica_MQ5;
int intervalo_envio = 300000;  //Tempo de envio para o banco
int contador_temporal, aux_temporal= 0;
bool flag;
bool flagCO2, flagCOV, flagGLP = true;
//float R0_MQ5;

/*Variaveis WiFi*/
char ssid[] = "********";
char pass[] = "********";

/*Variaveis do Banco*/
IPAddress server_addr(85,10,205,173);
char user[] = "**********";
char password[] = "**********";

char INSERT_SQL[] = "INSERT INTO estacoes.prototipo01 (temperatura, umidade, co, mp, glp, co2, covt) VALUES(%s, %s, %s, %s, %s, %s, %s)";
char query[128];

/*Chamando Objetos em variaveis globais*/
TFT_eSPI tft = TFT_eSPI(); // Chama a biblioteca para a tela LCD
DHTesp dht;
WiFiClient client;
MySQL_Connection conn((Client *)&client);
Adafruit_CCS811 ccs;
GP2YDustSensor sensorParticulado(GP2YDustSensorType::GP2Y1010AU0F, SHARP_LED_PIN, SHARP_VO_PIN);
```

Fonte: Autor (2021)

Figura 21 – Código fonte parte 3

```
376 float Ajuste_R(float razao, int sensor){
377   float ajustado, valor_85, valor_33, umid, temp;
378 
379   temp = dht.getTemperature();
380   umid = dht.getHumidity();
381 
382   if(temp > 50){
383     Serial.println(F("Erro, temperatura acima de 50°C."));
384     return NAN;
385   }
386   if(temp < 0){
387     Serial.println(F("Erro, temperatura abaixo de 0°C."));
388     return NAN;
389   }
390   if(umid > 85){
391     Serial.println(F("Erro, umidade acima de 85%."));
392     return NAN;
393   }
394   if(umid < 33){
395     Serial.println(F("Erro, umidade abaixo de 33%."));
396   }
397 
398   switch(sensor){
399     case 7:
400       valor_85 = -0.000002 * pow(temp, 3) + 0.0002 * pow(temp, 2) - 0.0123 * temp + 1.0268;
401       valor_33 = -0.000005 * pow(temp, 3) + 0.0005 * pow(temp, 2) - 0.0186 * temp + 1.2112;
402 
403       ajustado = razao * (valor_85 + (valor_33 - valor_85) * (33 - umid)/(33-85));
404       break;
405 
406     case 5:
407       valor_85 = -0.000002 * pow(temp, 3) + 0.0002 * pow(temp, 2) - 0.0114 * temp + 0.9944;
408       valor_33 = -0.000005 * pow(temp, 3) + 0.0004 * pow(temp, 2) - 0.016 * temp + 1.1711;
409 
410       ajustado = razao * (valor_85 + (valor_33 - valor_85) * (33 - umid)/(33-85));
411       break;
412   }
413 
414   return ajustado;
```

Fonte: Autor (2021)

Figura 22 – Código fonte parte 4

```
334 float Calibracao(int sensor){
335   int leitura_analogica;
336   float resistencia_em_ar_limpo, r0_Calibrado;
337 
338   switch (sensor){
339     case 7:
340       resistencia_em_ar_limpo = 0;
341       for (int i = 0; i<100; i++){
342         leitura_analogica = analogRead(pin_MQ7);
343         resistencia_em_ar_limpo += RL_MQ7 * ((5/(0.001221 * leitura_analogica))-1);
344         delay(100);
345 
346         Serial.print(F(" Calibrando MQ7: "));
347         Serial.print(i);
348         Serial.println(F("%"));
349       }
350       resistencia_em_ar_limpo = resistencia_em_ar_limpo / 100;
351       r0_Calibrado = resistencia_em_ar_limpo / (Ajuste_R(36.1286, sensor));
352       delay(1000);
353       break;
354 
355     case 5:
356       resistencia_em_ar_limpo = 0;
357       for (int i = 0; i<100; i++){
358         leitura_analogica = analogRead(pin_MQ5);
359         resistencia_em_ar_limpo += RL_MQ5 * ((5/(0.001221 * leitura_analogica))-1);
360         delay(100);
361 
362         Serial.print(F(" Calibrando MQ5: "));
363         Serial.print(i);
364         Serial.println(F("%"));
365       }
366       resistencia_em_ar_limpo = resistencia_em_ar_limpo / 100;
367       r0_Calibrado = resistencia_em_ar_limpo / (Ajuste_R(6.4754, sensor));
368       delay(1000);
369       break;
370   }
371     return r0_Calibrado;
372 }
```

Fonte: Autor (2021)

Na Figura 22, tem-se a função de calibração dos sensores, executada depois do tempo de aquecimento. Essa serve para coletar o valor da resistência em um ambiente sem a presença dos gases que influenciam o sensor. Nessa função, faz-se uma média de cem leituras dos sensores em ar limpo e utiliza-se a equação (2), tendo como razão $\frac{Rs}{R0}$ o valor retirado dos datasheets para o ar.

Figura 23 – Código fonte parte 5

```
/***********Insere no banco***************/
void Envia_Dados(){
  sprintf(query, INSERT_SQL, String(temperatura), String(umidade), String(coMedio), String(mpMedio), String(glpMedio), String(co2Medio), String(covMedio));
  if (!conn.connect(server_addr,3306, user, password)){
    conn.close();
    Conecta_Banco();
  }
  MySQL_Cursor * cur_mem = new MySQL_Cursor(&conn);
  cur_mem -> execute(query);
  delete cur_mem;
  Serial.println(F("Dados enviados!"));
}

/***********Conecta no banco****************/
void Conecta_Banco(){
  Serial.println(F("Conectando ao banco de dados..."));
  while(!conn.connect(server_addr,3306, user, password)){
    Serial.println(F("Conexão falhou!"));
    conn.close();
    delay(100);
    Serial.println(F("Conectando ao Banco novamente."));
  }
  Serial.println(F("Conectado ao Banco de dados"));
}

void Conecta_WiFi(){
  Serial.print(F("Conectando no WiFi..."));
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.print("");
  Serial.print(F("WIFI CONECTADA!"));
  Serial.println("");
}
```

Fonte: Autor (2021)

A Figura 23 mostra três funções: uma para conectar à internet, outra para conectar ao banco de dados e outra para inserir os dados no banco de dados.

Ainda há mais três funções no código fonte: a função *Display*, que atualiza as informações na tela; a função *setup*, responsável por iniciar os sensores, iniciar tela, executar as funções de conexão com internet e banco e aquecer os sensores; e a função *loop*, responsável por fazer as medições e executar a função *Display* a cada cinco segundos aproximadamente, além de executar a função de envio da média dos dados coletados a cada cinco minutos.

Disponibiliza-se o código em sua integridade na seção Anexos.

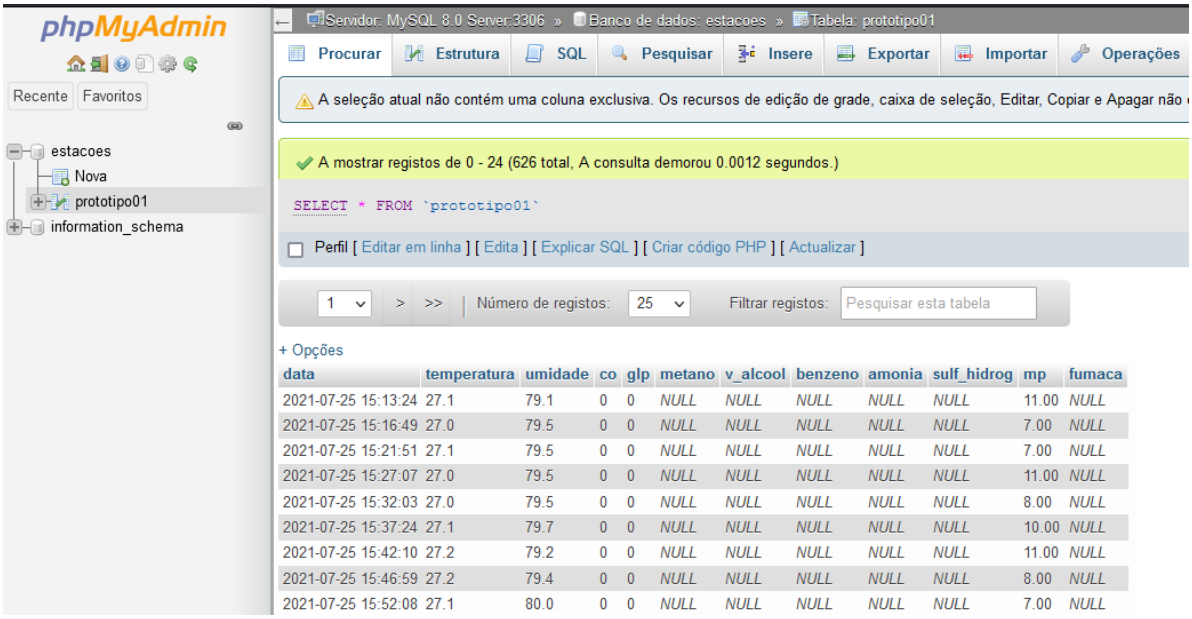
5.5 Banco de Dados

Utilizou-se da plataforma db4free.net para criar um banco de dados online e gratuito do MySQL. A plataforma é bastante utilizada para testes e aprendizado de banco.

Nela, criou-se um banco de dados com uma única tabela preparada para receber os dados dos gases exclusivamente para um único protótipo desenvolvido. Também se implementou campos para outros possíveis gases, além de um campo que registra o horário no qual um conjunto de dados foi guardado (Figura 24).

A plataforma utilizada para a hospedagem do banco de dados é gratuita para testes, e, muitas vezes, a comunicação do equipamento desenvolvido com a plataforma era perdida. Com isso, o equipamento fazia a reconexão automaticamente, o que exigia um tempo de reconexão, o que levava a paralização do sistema por vários segundos até que a conexão fosse concluída. Este problema deve ser resolvido adquirindo um serviço de hospedagem de banco.

Figura 24 – Administrador do banco contendo os campos da tabela e exemplo dos dados



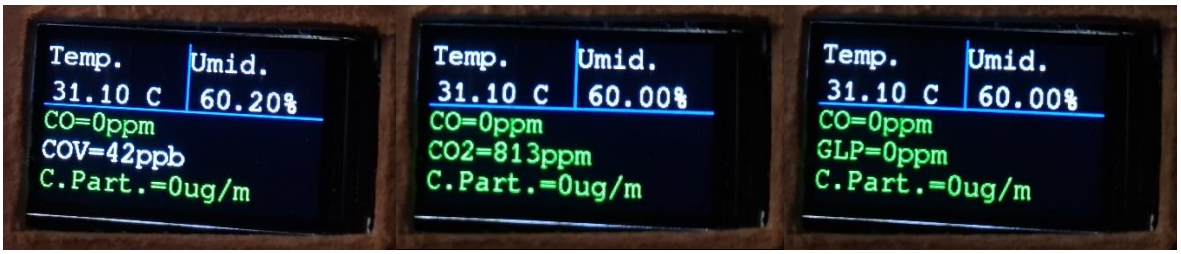
Fonte: Autor (2021)

5.6 Visualização dos dados

Os dados instantâneos podem ser visualizados no display OLED do microcontrolador, que exibe a temperatura e a umidade relativa em caixas na parte superior da tela. Caso esses parâmetros estejam superiores ou inferiores às condições de operação para a medição dos gases através dos sensores, a fonte ficará em amarelo.

Mostra-se os valores dos poluentes em três linhas. Em ordem, tem-se na primeira linha a medição de monóxido de carbono; na segunda linha, os valores de GLP, CO_2 e COV's, alternando-os a cada dois segundos; e na terceira e última linha, o valor de partículas totais em suspensão, representadas respectivamente por CO, GLP, CO_2, COV e C.Part (Figura 25).

Figura 25 – Display mostrando valores de medições instantâneas



Fonte: Autor (2021)

Os valores mostrados na tela do equipamento desenvolvido são de medições instantâneas, e, apesar de indicarem a qualidade do ar naquele momento pela cor, a legislação requer uma medição de média móvel para classificação da qualidade. Esta medição é feita apenas no algoritmo em Python.

5.7 Disponibilizando os Dados

Para a disponibilização dos dados históricos, criou-se um algoritmo em Python utilizando o ambiente de notebooks Jupyter do Google, o Google Colab.

Figura 26 – Código de Disponibilização de Dados parte 1

```
[2]  import mysql.connector as conn
     import pandas as pd
     import plotly.graph_objects as go
     import plotly.express as px
```

Fonte: Autor (2021)

Neste código (Figura 26), utiliza-se as bibliotecas *MySQL Connector*, útil para a conexão de bancos MySQL e possibilitadora de vários outros processos, depois de conectado; Pandas, uma biblioteca para manipulação de *dataframes*; e Plotly, uma biblioteca para gerar gráficos interativos.

Figura 27 – Código de Disponibilização de Dados parte 2

```
try:
  database = conn.connect(host="**.**.***.***", database="estacoes", user="*******", passwd="********", use_pure=True)
  query = "SELECT * FROM prototipo01;"
  df = pd.read_sql(query, database)
  database.close()
except Exception as e:
  database.close()
  print(str(e))
```

Fonte: Autor (2021)

Na Figura 27 está a célula de código que conecta com o banco de dados, requisita uma consulta de todos os dados da tabela prototipo01, cria um *dataframe* com todos estes dados retornados e fecha a conexão com o banco.

Para demonstração, fez-se medições no quarto do autor durante o intervalo de tempo do dia 25 de julho ao dia 06 de setembro de 2021. Na Figura 28 é possível visualizar o *dataframe* gerado a partir destas medições.

Figura 28 – *Dataframe* gerado a partir dos dados do banco

```
df
```

	data	temperatura	umidade	co	glp	metano	v_alcool	benzeno	amonia	sulf_hidrog	mp	fumaca	co2	covt
0	2021-07-25 12:13:24	27.1	79.1	0	0	None	None	None	None	None	11.0	None	NaN	NaN
1	2021-07-25 12:16:49	27.0	79.5	0	0	None	None	None	None	None	7.0	None	NaN	NaN
2	2021-07-25 12:21:51	27.1	79.5	0	0	None	None	None	None	None	7.0	None	NaN	NaN
3	2021-07-25 12:27:07	27.0	79.5	0	0	None	None	None	None	None	11.0	None	NaN	NaN
4	2021-07-25 12:32:03	27.0	79.5	0	0	None	None	None	None	None	8.0	None	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9781	2021-09-06 19:40:49	29.0	68.5	0	0	None	None	None	None	None	4.0	None	724.0	48.0
9782	2021-09-06 19:45:43	29.0	68.7	0	0	None	None	None	None	None	2.0	None	750.0	52.0
9783	2021-09-06 19:50:45	28.9	68.8	0	0	None	None	None	None	None	3.0	None	691.0	43.0
9784	2021-09-06 19:55:47	29.0	68.8	0	0	None	None	None	None	None	4.0	None	689.0	43.0
9785	2021-09-06 20:00:49	29.1	68.5	0	0	None	None	None	None	None	1.0	None	784.0	57.0

9786 rows × 14 columns

Fonte: Autor (2021)

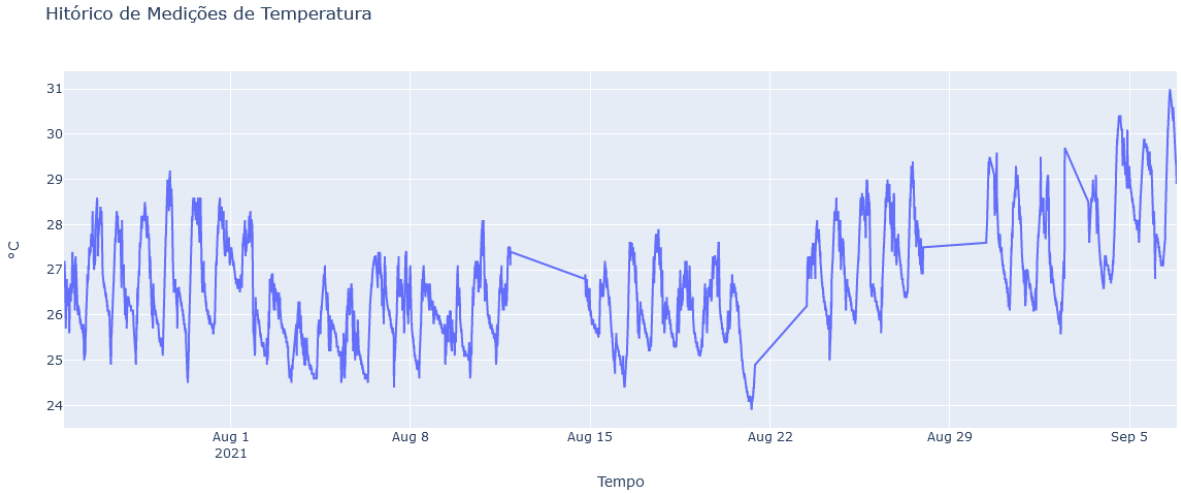
Figura 29 – Código de Disponibilidade de Dados parte 3

```
[7] fig = go.Figure(data=[go.Scatter(x=df['data'], y=df['temperatura'])], layout=go.Layout(title=go.layout.Title(text="Histórico de Medições de Temperatura")))
    fig.update_layout(xaxis_title="Tempo", yaxis_title="°C")
    fig.show()
```

Fonte: Autor (2021)

Utilizando do *dataframe*, na Figura 29, gerou-se os gráficos históricos de temperatura (Figura 30), de umidade do ar (Figura 31), concentração de monóxido de carbono (Figura 32), concentração de GLP (Figura 33) e de partículas em suspensão (Figura 34).

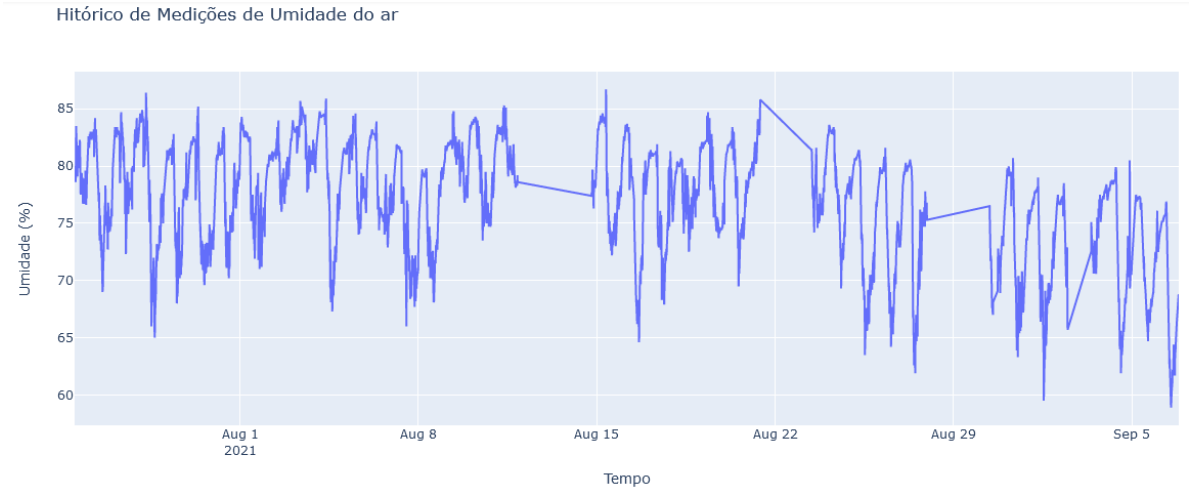
Figura 30 – Gráfico histórico da temperatura gerado pelo Código de Disponibilização de Dados



Fonte: Autor (2021)

A Figura 30 apresenta todo o histórico de temperatura no período monitorado, apresentando variações durante o dia, tendo picos máximo de temperatura diária geralmente no início da tarde e picos de temperatura mínima diária no início da madrugada.

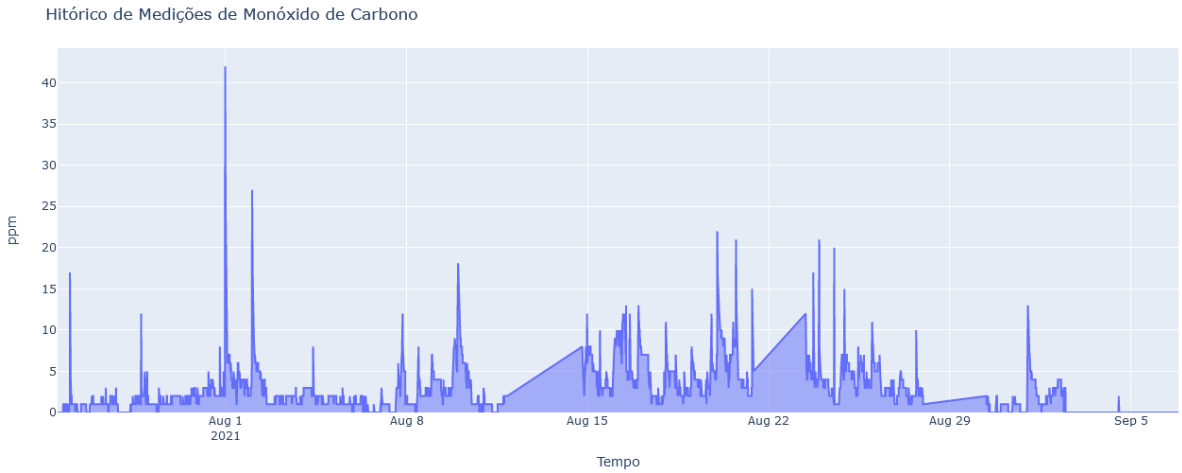
Figura 31 - Gráfico histórico da umidade do ar gerado pelo Código de Disponibilização de Dados



Fonte: Autor (2021)

A Figura 31 apresenta todo o histórico de umidade relativa do ar, apresentando sempre uma taxa elevada provavelmente devido à proximidade com a lagoa e a corrente de ar que vem de sua direção. Também foi possível perceber um aumento na taxa de umidade relativa do ar durante a noite quando o quarto monitorado permanece fechado.

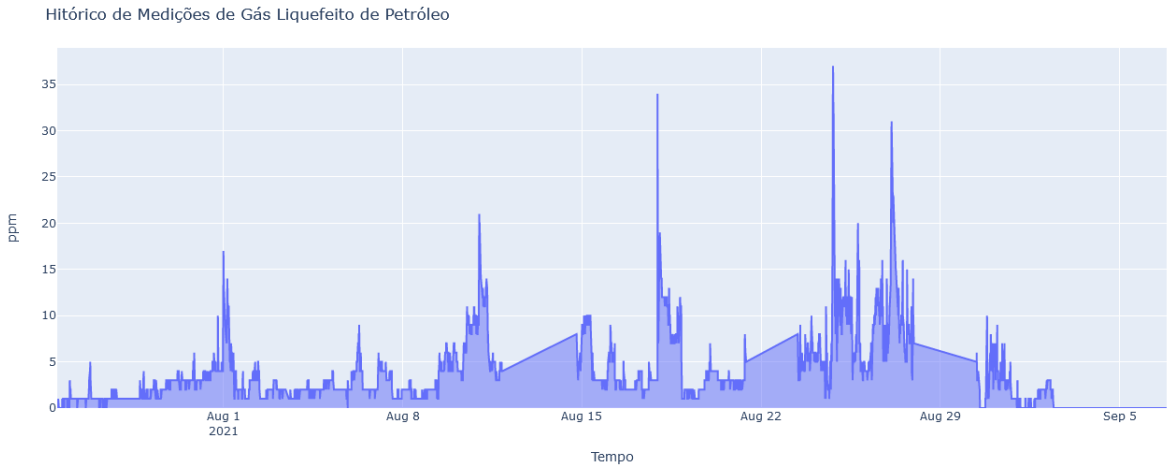
Figura 32 - Gráfico histórico de Monóxido de Carbono gerado pelo Código de Disponibilização de Dados



Fonte: Autor (2021)

A Figura 32 apresenta todo o histórico de medição do Monóxido de Carbono no período de monitoramento.

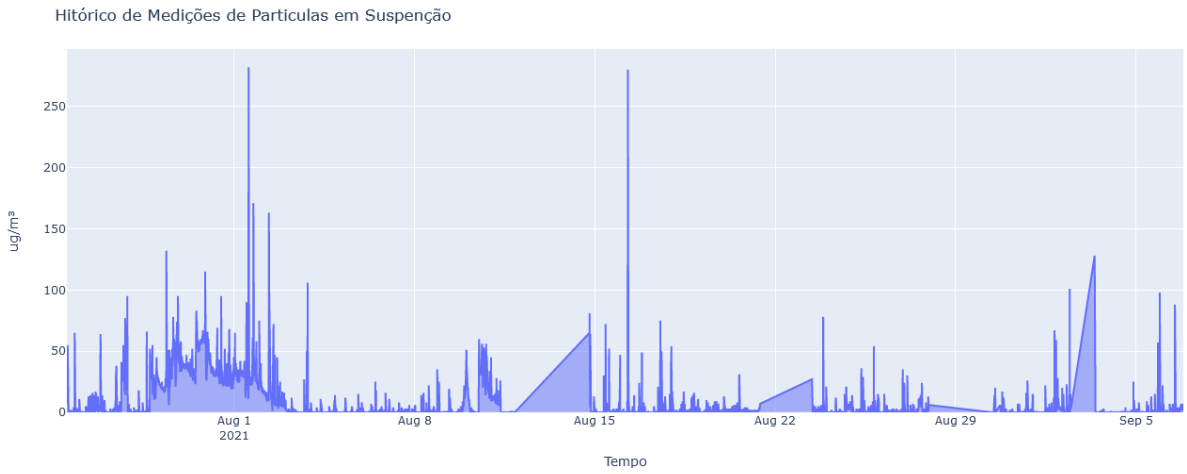
Figura 33 - Gráfico histórico de GLP gerado pelo Código de Disponibilização de Dados



Fonte: Autor (2021)

A Figura 33 apresenta as medições para GLP em todo o período de monitoramento, onde foi possível detectar uma pequena concentração desse gás que aparentemente provinha da cozinha, onde, quando feito uma verificação no local onde o botijão de gás está situado, foi detectado um vazamento do gás. Após a troca da válvula e mangueira instaladas no fogão, a concentração de GLP detectada no quarto diminuiu até chegar a zero.

Figura 34 - Gráfico histórico de Partículas em Suspensão gerado pelo Código de Disponibilização de Dados

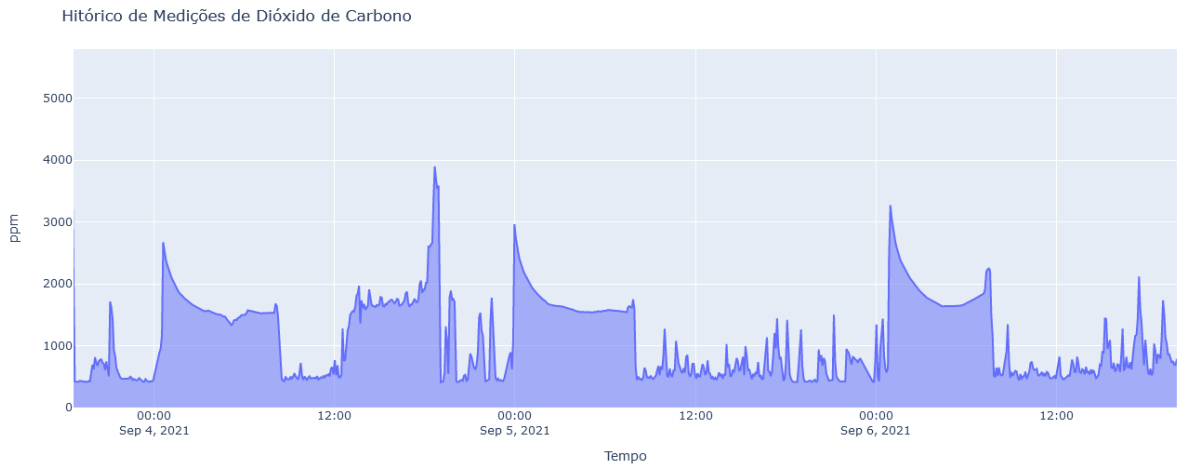


Fonte: Autor (2021)

A Figura 34 mostra todas as medições de partículas em suspensão em todo o período de medição.

A partir do dia 03 de setembro, adicionou-se e implementou-se o sensor CCS811, o que possibilitou o início das medições de CO_2 (Figura 35) e do seu equivalente em COV's (Figura 36).

Figura 35 – Gráfico histórico de Dióxido de Carbono gerado pelo Código de Disponibilidade de Dados

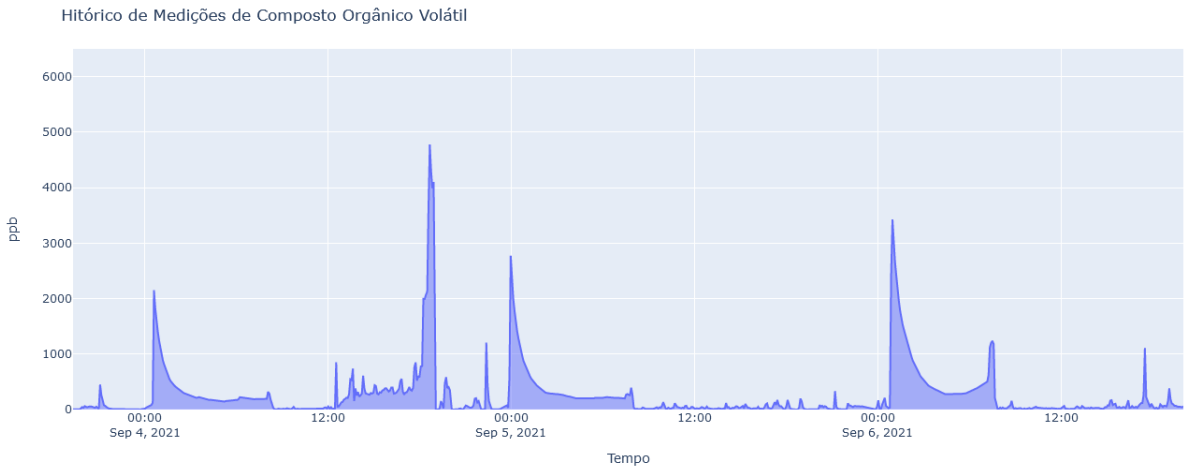


Fonte: Autor (2021)

Na Figura 35 está o histórico de todas as medições de Dióxido de Carbono desde quando o sensor foi implementado. Através dele, percebeu-se que as concentrações de CO_2 aumentavam bastante chegando a ultrapassar o limite indicado pela ANVISA no período em

que o quarto permanecia fechado e voltando a diminuir a concentração quando a porta e janela eram abertas.

Figura 36 – Gráfico histórico de Compostos Orgânicos Voláteis Totais gerado pelo Código de Disponibilidade de Dados



Fonte: Autor (2021)

A Figura 36 apresenta as medições de COVs a partir de quando o sensor foi implementado, apresentando picos geralmente no período da noite quando o quarto está fechado e não há um fluxo de renovação de ar no quarto.

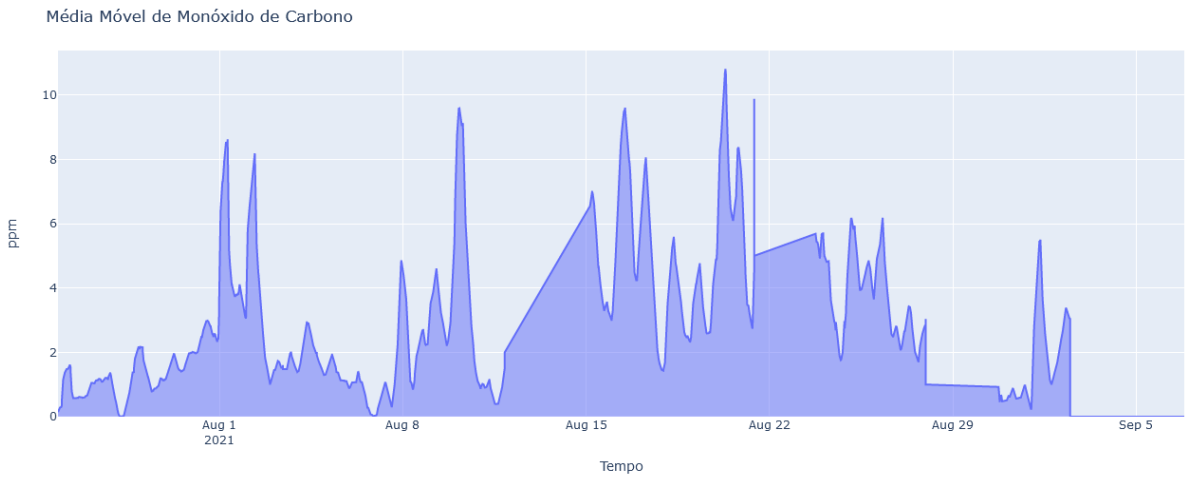
Utilizou-se o algoritmo da Figura 37 para gerar *dataframes* contendo a média móvel de 8 horas da série histórica dos poluentes Monóxido de Carbono (Figura 38) e COV (Figura 39), e a média móvel de 24 horas para o poluente Partículas em suspensão (Figura 40).

Figura 37 - Código de Disponibilidade de Dados parte 4

```
ultima_hora_medicao = df.iloc[-1]['data']
media_movel_CO = {'data': [], 'co': []}
media_movel_MP = {'data': [], 'mp': []}
media_movel_COVT = {'data': [], 'covt': []}
for i, linha in df.iterrows():

  #Calculando média movel de Monóxido de Carbono
  if (linha['data'] + timedelta(hours = 8)) <= ultima_hora_medicao:
    aux = df[df['data'] <= (linha['data'] + timedelta(hours = 8))]
    aux = aux[aux['data'] >= linha['data']]

    media_movel_CO['co'].append(aux.co.mean())
    media_movel_CO['data'].append(aux.iloc[-1]['data'])

    media_movel_COVT['covt'].append(aux.covt.mean())
    media_movel_COVT['data'].append(aux.iloc[-1]['data'])

  #Calculando média movel de particulas suspensas
  if (linha['data'] + timedelta(hours = 24)) <= ultima_hora_medicao:
    aux = df[df['data'] <= (linha['data'] + timedelta(hours = 24))]
    aux = aux[aux['data'] >= linha['data']]

    media_movel_MP['mp'].append(aux.mp.mean())
    media_movel_MP['data'].append(aux.iloc[-1]['data'])

media_movel_CO = pd.DataFrame(media_movel_CO)
media_movel_MP = pd.DataFrame(media_movel_MP)
media_movel_COVT = pd.DataFrame(media_movel_COVT)
```

Fonte: Autor (2021)

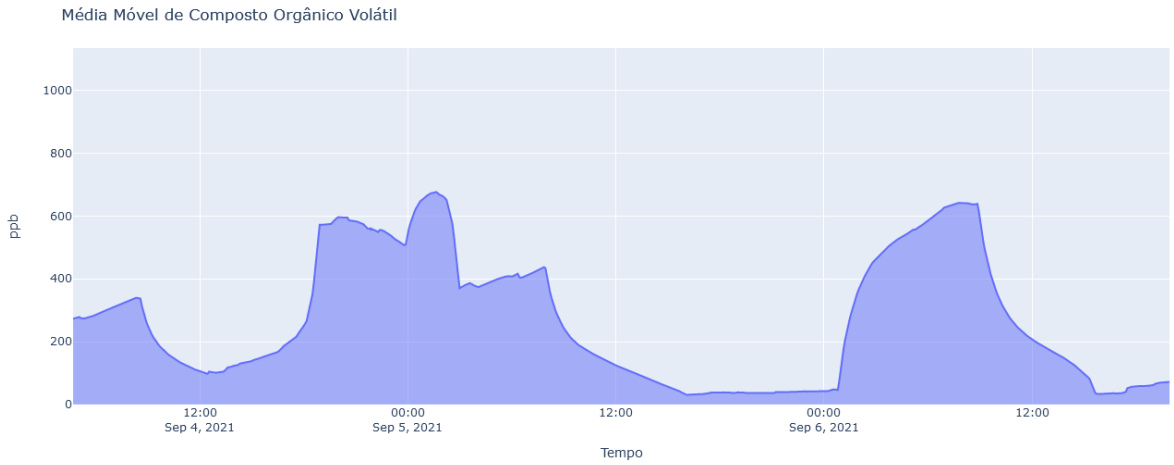
Figura 38 - Gráfico de Média Móvel para Monóxido de Carbono gerado pelo Código de Disponibilidade de Dados



Fonte: Autor (2021)

Através da média móvel de 8 horas para Monóxido de Carbono (Figura 38) é possível ver um gráfico mais suave.

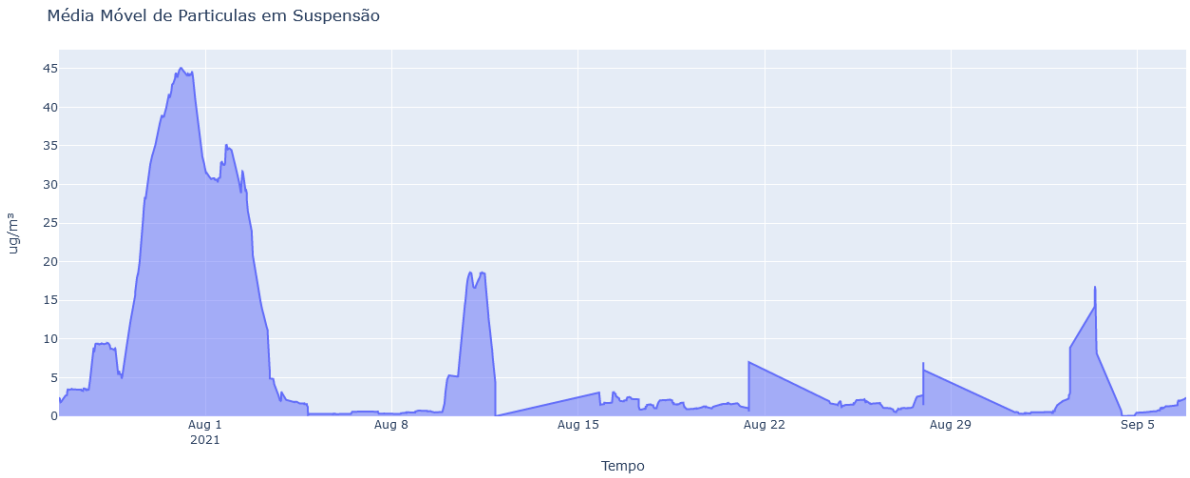
Figura 39 - Gráfico de Média Móvel para COV gerado pelo Código de Disponibilidade de Dados



Fonte: Autor (2021)

O mesmo gráfico foi feito para o COV (Figura 39), utilizando também média móvel de 8 horas, no entanto, apenas para acrescentar informação ao trabalho.

Figura 40 - Gráfico de Média Móvel para Partículas em Suspensão gerado pelo Código de Disponibilidade de Dados

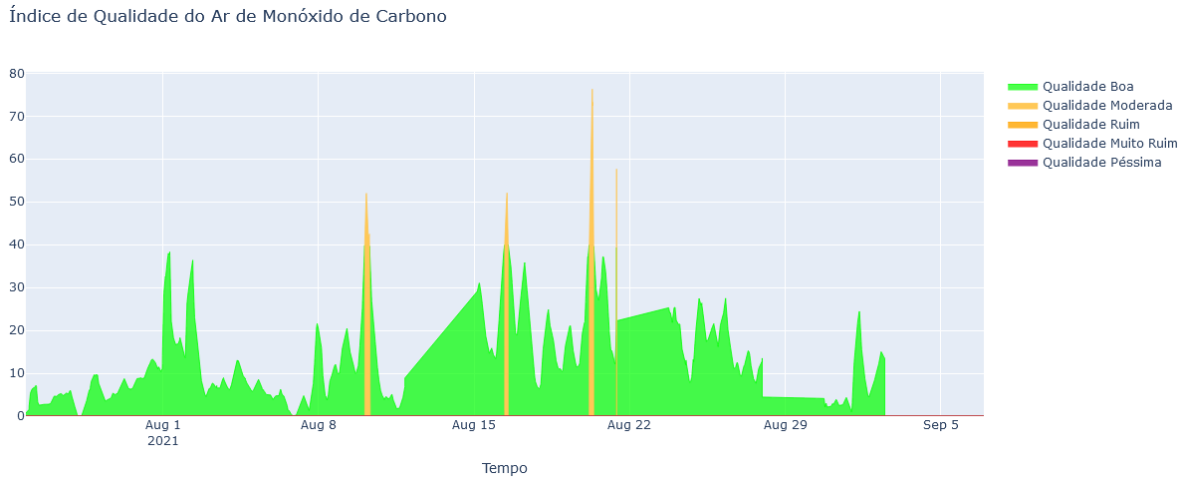


Fonte: Autor (2021)

Com o gráfico de média móvel de 24 horas para Partículas em Suspensão (Figura 40), foi possível identificar com mais clareza que a maior concentração de partículas em suspensão ocorreu nos primeiros dias de monitoramento.

Com os dados de média móvel dos poluentes, calculou-se toda a série histórica de IQAR utilizando os critérios de cálculo para Monóxido de Carbono (Figura 41), MP_{10} (Figura 42) e $MP_{2,5}$ (Figura 43), conforme foi exposto no Quadro 4.

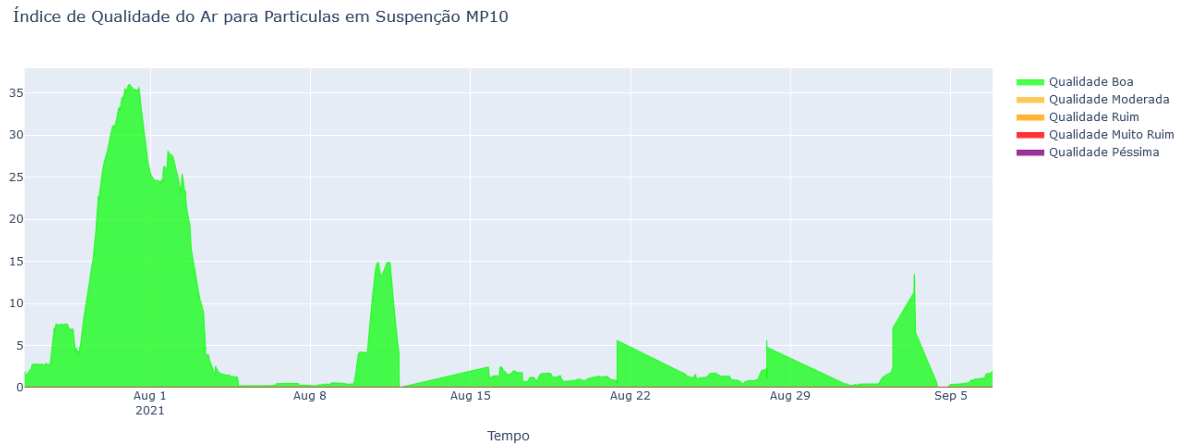
Figura 41 - Histórico do IQAR para Monóxido de Carbono



Fonte: Autor (2021)

O gráfico mostrando a série histórica de IQAR do Monóxido de Carbono (Figura 41) indica que a maior parte do período monitorado teve uma qualidade do ar boa, apresentando apenas quatro picos momentâneos que indicaram uma qualidade de ar moderada.

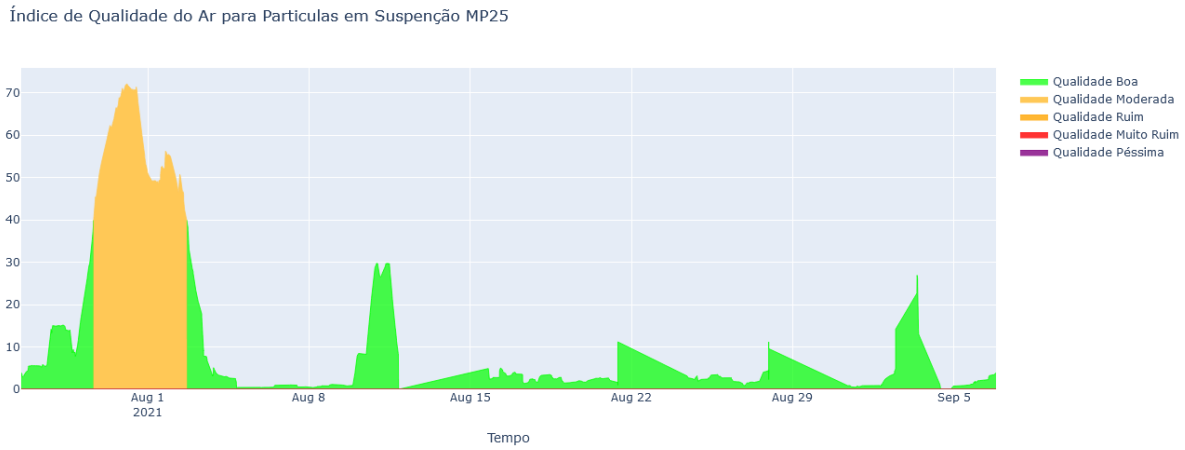
Figura 42- Histórico do IQAR para MP_{10}



Fonte: Autor (2021)

O gráfico da Figura 42 apresenta todo o histórico de IQAR para MP_{10}, indicando que em todo o período monitorado a qualidade do ar foi boa.

Figura 43- Histórico do IQAR para $MP_{2,5}$



Fonte: Autor (2021)

A Figura 43, apresentando o histórico do IQAR para $MP_{2,5}$, indicou que no início do monitoramento a qualidade do ar foi moderada, onde depois passou para boa e permanecendo assim até o fim do período de monitoramento.

Embora o algoritmo Python tenha calculado valores isolados de MP_{10} e $MP_{2,5}$, pontua-se que o sensor de partículas suspensas detecta as concentrações somadas. Desta forma, pelo princípio da precaução, é recomendável a medição utilizando os critérios para o pior caso, ou seja, para o $MP_{2,5}$. Vale salientar ainda, que o algoritmo possui uma célula que disponibiliza também os dados em arquivos de extensão .csv e .xlsx.

Os resultados apresentados indicam, de forma geral, que o protótipo montado bem como os sistemas utilizados para sua operação, operou de forma adequada na medição dos parâmetros propostos.

6. CONCLUSÃO

O estudo apresentado teve como proposta a construção de um sistema de monitoramento de gases de combustão para ambientes internos, com o uso de um microcontrolador ESP32 TTGO e de sensores que medem a concentração de gases CO, GLP e Partículas Suspensas. Também se adicionou, no decorrer do seu desenvolvimento, um sensor que mede a concentração de CO_2 e COV, no intuito de indicar se no ambiente interno está havendo ou não a renovação do ar.

A avaliação dos resultados obtidos indica que os objetivos propostos neste trabalho foram alcançados tendo em vista que:

- O equipamento desenvolvido tem custo acessível e é de fácil montagem e operação;
- Os resultados das medições apresentados pelo protótipo são diretos e intuitivos, facilitando assim o entendimento do usuário;
- O protótipo apresenta tamanho e forma que pode ser usado facilmente em diversos ambientes fechados;
- O custo do protótipo ainda pode ser reduzido, utilizando outros microcontroladores ESP32 mais baratos e sem tela, podendo-se desenvolver uma função de visualização dos dados via smartphone.

Os resultados e conclusões obtidos neste trabalho, indicam ainda, para trabalhos futuros, a sugestão de comparação dos resultados obtidos no protótipo com resultados obtidos utilizando detectores multigases vendidos no mercado e possíveis melhorias no design.

7. REFERÊNCIAS

AMS. **CCS811: Ultra-Low Power Digital Gas Sensor for Monitoring Indoor Air Quality.** 2016. Disponível em: <https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf.> Acesso em: 06 Set. 2021.

ARBEX, Marcos Abdo *et al*. **A poluição do ar e o sistema respiratório**. J. bras. pneumol., São Paulo, v. 38, n. 5, p. 643-655, out. 2012. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S180637132012000500015&lng=pt&nrm=iso>. Acesso em: 14 jan. 2020.

ARBEX, Marcos Abdo *et al*. **Queima de biomassa e efeitos sobre a saúde**. J. bras. pneumol. São Paulo , v. 30, n. 2, p. 158-175, Apr. 2004. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S180637132004000200015&lng=en&nrm=iso>. Acesso em: 27 jan. 2020.

AZUAGA, Denise. **Danos em ambientes causados por veículos leves no Brasil**. Rio de Janeiro: UFRJ, COPPE, 2000. Disponível em: <http://antigo.ppe.ufrj.br/ppe/production/tesis/dazuaga.pdf>. Acesso em: 30 jan. 2020.

CANÇADO, José Eduardo Delfini *et al*. **Repercussões clínicas da exposição à poluição atmosférica**. J. bras. pneumol., São Paulo, v. 32, supl. 2, p. S5-S11, May 2006. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S180637132006000800003&lng=en&nrm=iso>. Acesso em: 14 jan. 2020.

CETESB. **Ficha de Informação Toxicológica: GLP**. Disponível em:<https://cetesb.sp.gov.br/labo-

ratorios/wp-content/uploads/sites/24/2013/11/GLP.pdf>. Acesso em: 26 de Jan. 2020.

CETESB, **Poluentes.** Disponível em: < https://cetesb.sp.gov.br/ar/poluentes/>. Acesso em: 06 de Set. 2021.

CONSELHO NACIONAL DO MEIO AMBIENTE. 2018. **Resolução CONAMA nº 491/2018 – Dispõe sobre os padrões de qualidade do ar**. Conselho Nacional de Meio Ambiente – CONAMA, Brasil.

DO AMARAL, Maíra Freire Pecegueiro. **Avaliação De Sistema Para Monitoramento De Gás Amônia Em Galpões Avícolas Com Ventilação Negativa**. Diss. Universidade Federal de Viçosa, 2007. Disponível em: http://arquivo.ufv.br/dea/ambiagro/arquivos/Tese%20de%20Ma%C3%ADra%20Pecegueiro2007.pdf. Acesso em: 30 de Jan. 2020.

GITHUB. **GP2YDustSensor**. 2019. Disponível em: <https://github.com/luciansabo/GP2YDustSensor>. Acesso em: 27 jul. 2021.

GITHUB. **MySQL_Connector_Arduino**. 2020. Disponível em: <https://github.com/ChuckBell/MySQL_Connector_Arduino>. Acesso em: 27 jul. 2021.

GITHUB. **DHTesp**. 2021a. Disponível em: < https://github.com/beegee-tokyo/DHTesp>. Acesso em: 27 jul. 2021.

GITHUB. **TFT_eSPI**. 2021b. Disponível em: < https://github.com/Bodmer/TFT_eSPI>. Acesso em: 27 jul. 2021.

GITHUB. **Adafruit_CCS811**. 2021c. Disponível em: <https://github.com/adafruit/Adafruit_CCS811>. Acesso em: 06 set. 2021.

INSTITUTO DE ENERGIA E MEIO AMBIENTE. **Diagnóstico da rede de monitoramento da qualidade do ar no Brasil**. 2014, Disponível em: <http://www.forumclima.pr.gov.br/arquivos/

File/Rosana/Diagnostico_Qualidade_do_Ar_Versao_Final_Std.pdf>. Acesso em 14 jan. 2020.

INSTITUTO DE ENERGIA E MEIO AMBIENTE. **Padrões da qualidade do ar: Experiência comparada Brasil, EUA e União Europeia**. 2012. Disponível em: <https://iema-site-staging.s3.

amazonaws.com/padroes-final01.pdf> . Acesso em 14 de Jan. 2020

JACOMINO, Vanusa Maria Feliciano *et al*. **Avaliação da qualidade do ar em um polo produtor de ferro-gusa**. Eng. Sanit. Ambient., Rio de Janeiro, v. 14, n. 4, p. 511-520, Dec.

2009. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-41522009000400011

&lng=en&nrm=iso>. Acesso em 14 jan. 2020.

LEITE, José Roberto Emiliano; URSINI, Edson Luiz; MARTINS, Paulo S. **Integração da Internet das Coisas (IoT) com a RFID e Rede de Sensores: Revisão de Literatura**. 2016. Disponível em: <http://lcv.fee.unicamp.br/images/BTSym-16/proceedings/pa65-16-edited.pdf>. Acesso em 14 jan. 2020.

MARTINS, Lourdes Conceição *et al* . Relação entre poluição atmosférica e atendimentos por infecção de vias aéreas superiores no município de São Paulo: avaliação do rodízio de veículos. Rev. bras. epidemiol., São Paulo, v. 4, n. 3, p. 220-229, Nov. 2001. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1415790X2001000300008&lng=en&nrm=iso>. Acesso em: 14 Jan. 2020.

NASA. **Carbon Dioxide.** 2021. Disponível em: < https://climate.nasa.gov/vital-signs/carbon-dioxide/>. Acesso em: 05 Set. 2021.

OLIVEIRA, L. A. ; PINHO, E. F. M. ; COSME, E. A. . **Tecnologia de monitoramento da qualidade do ar**. Revista de Ciências Exatas e Tecnológicas das Faculdades Santo Agostinho, v. 4, p. 55-64, 2016. Disponível em: <https://fasa.edu.br/assets/arquivos/files/Revista_FACET_v.4.n.2.2016.pdf>. Acesso em: 26 de Jan. 2020.

PETROBRAS. **Gás Liquefeito de Petróleo: Informações Técnicas.** 2019. Disponível em: <http://sites.petrobras.com.br/minisite/assistenciatecnica/public/downloads/manual-tecnico-gas-liquefeito-petrobras-assistencia-tecnica-petrobras.pdf>. Acesso em: 06 de Set. 2021.

RUSSO, P. R. (2010). **A qualidade do ar no município do Rio de Janeiro: análise espaço-temporal de partículas em suspensão na atmosfera**. Revista De Ciências Humanas, 1(1). Recuperado de https://periodicos.ufv.br/RCH/article/view/3491

SANTOS, Bruno P. *et al*. **Internet das coisas: da teoria à prática**. Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, v. 31, 2016. Disponível em: <https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf>. Acesso em 14 jan. 2020.

SATISH, Y.; MENDELL, M. J.; SHEKHAR, K.; HOTCHI, T.; SULLIVAN, D.; STREUFERT, S.; FISK, W. J. Is CO2 na Indoor Pollutant? Direct Effects of Low-to-Moderate CO2 Concentration on Humnan Decision-Making Performance. Environmental Health Perspective. V.120, p.2-35, 2012. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3548274/>. Acesso em: 05 Set. 2021.

SCHIRMER, Waldir Nagel *et al* . **A poluição do ar em ambientes internos e a síndrome dos edifícios doentes**. Ciênc. saúde coletiva, Rio de Janeiro, v. 16,n. 8, p. 3583-3590, Aug. 2011. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-81232011000900026&lng=en&nrm=iso>. Acesso em: 14 Jan. 2020.

SOUZA, David José. **Desbravando o PIC: Ampliado e Atualizado para PIC 16F628A**. 12ª ed. São Paulo, SP, Brasil: Érica, 2005.

VORMITTAG, E. D. M., DE ARAÚJO, P. A., CIRQUEIRA, S. S. R., WICHER NETO, H., & SALDIVA, P. H. N. **Análise do monitoramento da qualidade do ar no Brasil**. *Estudos Avançados*, v. 35, p. 7-30, 2021.

WORLD HEALTH ORGANIZATION (WHO *et al*.) **Air quality guidelines for Europe**. 2000.

WORLD HEALTH ORGANIZATION (WHO). **Indoor air quality: biological contaminants**: report on a WHO meeting, Rautavaara, 29 August -2 September 1988. Regional Office for Europe 1990. https://apps.who.int/iris/handle/10665/260557

8. ANEXOS

1 – CÓDIGO FONTE DO CONTROLADOR

```
/*Chamando Bibliotecas*/

#include "DHTesp.h"

#include "Adafruit_CCS811.h"

#include "GP2YDustSensor.h"

#include <TFT_eSPI.h> // Hardware-specific library

#include <SPI.h>

#include <Wire.h>

#include <math.h>

#include <WiFi.h>

#include <MySQL_Connection.h>

#include <MySQL_Cursor.h>


/*Definindo pinos*/

#define pin_DHT 17 //Usando o DHT22

#define pin_MQ7 36 // Usando o sensor MQ7 - CO

#define pin_MQ5 37 // Usando o sensor MQ5 - GLP

#define SHARP_LED_PIN 33 // Usando o sensor de pariculado

#define SHARP_VO_PIN  32 // Usando o sensor de pariculado


/*Definindo constantes*/

#define RL_MQ7 10000
```

```
#define R0_MQ7 1301.57

#define RL_MQ5 20000

#define R0_MQ5 17280.05


#define voltagem 5 //define a voltagem do sensor


/*Variaveis Globais */

String nome = "primeiro prototipo";

int nivel = 0;

float umidade, temperatura, mp;

int co, co2, cov, particulado, glp;

int coMedio, coSoma, contCO, mpMedio, mpSoma, contMP, glpMedio, glpSoma, contGLP =
0;

int covMedio,covSoma, contCOV, co2Medio, co2Soma, contCO2 = 0;

int leitura_analogica_MQ7, leitura_analogica_MQ5;

int intervalo_envio = 300000;  //Tempo de envio para o banco

int contador_temporal, aux_temporal= 0;

bool flag;

bool flagCO2, flagCOV, flagGLP = true;

//float R0_MQ5;


/*Variaveis WiFi*/

char ssid[] = "********";

char pass[] = "*********";
```

```
/*Variaveis do Banco*/

IPAddress server_addr(**,**,***,***);

char user[] = "********";

char password[] = "**********";


char INSERT_SQL[] = "INSERT INTO estacoes.prototipo01 (temperatura, umidade, co, mp, glp, co2, covt) VALUES(%s, %s, %s, %s, %s, %s, %s)";

char query[128];


/*Chamando Objetos em variaveis globais*/

TFT_eSPI tft = TFT_eSPI(); // Chama a biblioteca para a tela LCD

DHTesp dht;

WiFiClient client;

MySQL_Connection conn((Client *)&client);

Adafruit_CCS811 ccs;

GP2YDustSensor sensorParticulado(GP2YDustSensorType::GP2Y1010AU0F, SHARP_LED_PIN, SHARP_VO_PIN);


void setup(){

  pinMode(pin_MQ7, INPUT); //Iniciando pino como input

  pinMode(pin_MQ5, INPUT); //Iniciando pino como input


  tft.begin();          //Inicia a tela

  tft.setRotation(1);   //Faz a tela trabalhar na horizontal
```

```
Serial.begin(115200);   //Inicia o monitor serial

dht.setup(pin_DHT, DHTesp::DHT22);  // Conecta o DHT no Esp32

//Wire.begin(21,22);
//ccs.begin();
if(!ccs.begin()){
  Serial.println("Falha ao iniciar sensor!");
}

sensorParticulado.begin();    //Iniciando o sensor de particulado

                              /*configurações da tela*/
tft.fillScreen(TFT_BLACK);              //Cor de fundo preto
tft.setCursor(0,70);                    //Coloca cursor na coordenada (0,70)da tela
tft.setFreeFont(&FreeMono12pt7b);       //Fonte escolhida
tft.setTextColor(TFT_WHITE, TFT_BLACK); //Cor do texto
tft.println("CONECTANDO WIFI");         //Mostra na tela

WiFi.begin(ssid, pass);   //Inicia a conexão WiFi
Conecta_WiFi();           //Executa função de conexão e feedback

tft.fillScreen(TFT_BLACK);
```

```
tft.setCursor(0,70);
tft.println("CONECTADO!");
delay(2000);              //espera 2 segundos

tft.fillScreen(TFT_BLACK);
tft.setCursor(0,70);
tft.println("CONECTANDO BANCO");
Conecta_Banco();          //Executa função de conexão do banco de dados

tft.fillScreen(TFT_BLACK);
tft.setCursor(0,70);
tft.println("AQUECENDO SENSOR");
Serial.println(F("AQUECENDO O SENSOR"));
delay(10000);
/*tft.fillScreen(TFT_BLACK);
tft.setCursor(0,70);
tft.println("CALIBRANDO...");
R0_MQ7 = Calibracao(7);

tft.fillScreen(TFT_BLACK);
tft.setCursor(0,70);
tft.print("R0_MQ7=");
tft.println(R0_MQ7);
delay(3600000);
```

```
  tft.fillScreen(TFT_BLACK);
  tft.setCursor(0,70);
  tft.println("CALIBRANDO...");
  R0_MQ5 = Calibracao(5);

  tft.fillScreen(TFT_BLACK);
  tft.setCursor(0,70);
  tft.print("R0_MQ5=");
  tft.println(R0_MQ5);
  delay(3600000);
  */

  float linha = sensorParticulado.getBaselineCandidate(); //Indica um valor para calibração
  sensorParticulado.setBaseline(2.1*linha);              //Calibra utilizando o valor de calibração e um multiplicador
}

void loop(){
  float RS_MQ7, RSR0_MQ7, RS_MQ5, RSR0_MQ5;
  int ppmCO, ppmGLP;

  /*Leitura do sensor de Monóxido de Carbono*/
```

```
leitura_analogica_MQ7 = analogRead(pin_MQ7);
RS_MQ7 = RL_MQ7 * ((5/(0.001221 * leitura_analogica_MQ7))-1);
RSR0_MQ7 = Ajuste_R((RS_MQ7/R0_MQ7), 7);
ppmCO = pow(10, (-log10(RSR0_MQ7)+log10(19.949))/(0.655));

Serial.print(F("Leitura MQ7 CO: "));
Serial.print(leitura_analogica_MQ7);
Serial.print("   ");
Serial.print(ppmCO);
Serial.print(F("ppm\t"));

/*Leitura do sensor de GLP*/
leitura_analogica_MQ5 = analogRead(pin_MQ5);
RS_MQ5 = RL_MQ5 * ((5/(0.001221 * leitura_analogica_MQ5))-1);
RSR0_MQ5 = Ajuste_R((RS_MQ5/R0_MQ5), 5);
ppmGLP = pow(10, (-log10(RSR0_MQ5)+log10(5.543))/(0.397));

Serial.print(F("Leitura MQ5 GLP: "));
Serial.print(leitura_analogica_MQ5);
Serial.print("   ");
Serial.print(ppmGLP);
Serial.print(F("ppm\t"));

temperatura = dht.getTemperature();
```

```
umidade = dht.getHumidity();
particulado = sensorParticulado.getDustDensity();



flag = ccs.available();
if(flag){
  if(!ccs.readData()){
    //Serial.println(" ");

    co2 = ccs.geteCO2();
    cov = ccs.getTVOC();

   if(cov > 1187 and co2 > 8192 or co2 < 400){
    co2 = NULL;
    cov = NULL;
   }

    Serial.print(F("CO2: "));
    Serial.print(co2);
    Serial.print(F("ppm      COVT: "));
    Serial.print(cov);
    Serial.print(F("ppb"));

    if(cov != NULL and co2 != NULL){
```

```
      co2Soma += co2;
      covSoma += cov;
      contCOV += 1;
      contCO2 += 1;
    }

  }
}

co = ppmCO;
glp = ppmGLP;
mp = particulado;
mostrar_display();

mpSoma += mp;
contMP += 1;


if (umidade > 85 || umidade < 33 || temperatura > 50 || temperatura < 0){
    co = NULL;
    glp = NULL;
} else {
    coSoma += ppmCO;
    contCO += 1;
```

```
    glpSoma += ppmGLP;
    contGLP += 1;
  }

contador_temporal = millis();
if (contador_temporal - aux_temporal > intervalo_envio){
  aux_temporal = millis();

  tft.setCursor(220, 75);
  tft.println("E");
  if (WiFi.status() == WL_DISCONNECTED){
    WiFi.disconnect();
    WiFi.begin(ssid, pass);
  }
  mpMedio = mpSoma/contMP;
  coMedio = coSoma/contCO;
  glpMedio = glpSoma/contGLP;
  co2Medio = co2Soma/contCO2;
  covMedio = covSoma/contCOV;
  Envia_Dados();

  /*Relatório de envio*/
  Serial.println("");
  Serial.print("media MP: ");
```

```
  Serial.print(mpMedio);
  Serial.print(" media CO: ");
  Serial.print(coMedio);
  Serial.print(" media GLP: ");
  Serial.print(glpMedio);
  Serial.print(" media CO2: ");
  Serial.print(co2Medio);
  Serial.print(" media COVT: ");
  Serial.println(covMedio);

  mpSoma = 0;
  contMP = 0;
  coSoma = 0;
  contCO = 0;
  glpSoma = 0;
  contGLP = 0;
  co2Soma = 0;
  contCO2 = 0;
  covSoma = 0;
  contCOV = 0;
}
co = NULL;
mp = NULL;
glp = NULL;
```

```
  co2 = NULL;
  cov = NULL;

  delay(2000);
}

void mostrar_display()
{
  delay(dht.getMinimumSamplingPeriod());

    if(co >= 40 or mp >= 40){
      nivel = 3; //emergência
    }
    if(co >=30 and co < 40){
      nivel = 2; //alerta
    }
    if(mp >=420 and mp < 500){
      nivel = 2; //alerta
    }
    if(co >=15 and co < 30){
      nivel = 1; //atenção
    }
    if(mp >=250 and mp < 420){
      nivel = 1; //atenção
```

```
  }
  if (co < 15 and mp < 250){
    nivel = 0; //normal
  }

switch (nivel){
  case 0:
    tft.fillScreen(TFT_BLACK);
    break;
  case 1:
    tft.fillScreen(TFT_YELLOW);
    break;
  case 2:
    tft.fillScreen(TFT_ORANGE);
    break;
  case 3:
    tft.fillScreen(TFT_MAGENTA);
    break;
}

//tft.fillScreen(TFT_BLACK);
//tft.setTextColor(TFT_GREEN, TFT_BLACK);
tft.setCursor(0,20);
tft.setFreeFont(&FreeMono12pt7b);
```

```
tft.setTextColor(TFT_WHITE, TFT_BLACK);
tft.println("Temp.    Umid.");
tft.drawLine(0, 55,250,55, TFT_BLUE);
tft.drawLine(125, 0,125,55, TFT_BLUE);

//temperatura = dht.getTemperature();
//umidade = dht.getHumidity();
if(isnan(umidade)){
  Serial.println(F("Erro na leitura da umidade."));
}else{
  Serial.print(F("       Umidade:"));
  Serial.print(umidade, 1);
  Serial.print(F("%\t\t"));

if(isnan(temperatura)){
  Serial.println(F("Erro na leitura da Temperatura."));
}else{
  Serial.print(F("Temperatura:"));
  Serial.print(temperatura, 1);
  Serial.print("°C\t");

  tft.setCursor(5, 50);
  if(temperatura > 50 || temperatura < 0){
    tft.setTextColor(TFT_YELLOW, TFT_BLACK);
```

```
    tft.print(temperatura);
    tft.print(F("°C"));
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
  } else{
    tft.print(temperatura);
    tft.print(F(" C"));
  }
}

  Serial.print("\t");
  Serial.print(F("Concentração de Particulado: "));
  Serial.print(particulado);
  Serial.println(F("ug/m3"));

  tft.setCursor(135, 50);
  if(umidade > 85 || umidade < 33){
    tft.setTextColor(TFT_YELLOW, TFT_BLACK);
    tft.print(umidade);
    tft.println(F("%"));
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
  } else{
    tft.print(umidade);
    tft.println(F("%"));
  }
```

```
}



/*##############Cor para o Monóxido de Carbono####################*/
if(co >= 15){

  tft.setTextColor(TFT_PURPLE, TFT_WHITE);
  tft.print(F("CO="));
  tft.print(co);
  tft.println(F("ppm"));
  tft.setTextColor(TFT_WHITE, TFT_BLACK);

}
if(co > 13 and co <=15){
  tft.setTextColor(TFT_RED, TFT_WHITE);
  tft.print(F("CO="));
  tft.print(co);
  tft.println(F("ppm"));
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
}
if(co > 11 and co <=13){
  tft.setTextColor(TFT_ORANGE, TFT_WHITE);
  tft.print(F("CO="));
  tft.print(co);
```

```
  tft.println(F("ppm"));
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
}
if(co > 9 and co <=11){
  tft.setTextColor(TFT_YELLOW, TFT_WHITE);
  tft.print(F("CO="));
  tft.print(co);
  tft.println(F("ppm"));
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
}
if(co <= 9){
  tft.setTextColor(TFT_GREEN, TFT_WHITE);
  tft.print(F("CO="));
  tft.print(co);
  tft.println(F("ppm"));
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
}

//Alternador para mostrar na tela
if (flagGLP){
flagGLP = false;
flagCO2 = true;
flagCOV = false;
} else{
```

```
  if (flagCO2){
    flagGLP = false;
    flagCO2 = false;
    flagCOV = true;
  } else {
    flagGLP = true;
    flagCO2 = false;
    flagCOV = false;
  }
}
/*##############Cor para o GLP####################*/
if (flagGLP){
  if (glp > 2000){
    tft.setTextColor(TFT_RED, TFT_WHITE);
    tft.print(F("GLP="));
    tft.print(glp);
    tft.println(F("ppm"));
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
  }else{
    tft.setTextColor(TFT_GREEN, TFT_WHITE);
    tft.print(F("GLP="));
    tft.print(glp);
    tft.println(F("ppm"));
  }
```

```
}
/*##############Cor para o Composto Orgânico Volátil####################*/
if (flagCOV){
   if(cov>1187){
     tft.print(F("COV="));
     tft.println(F("ERRO"));
   } else {
     tft.print(F("COV="));
     tft.print(cov);
     tft.println(F("ppb"));
  }
}
/*##############Cor para o Dióxido de Carbono####################*/
if (flagCO2){

  if(co2<400 or co2>8192){
    tft.print(F("ERRO"));
  } else {
    if(co2 <= 1000){
      tft.setTextColor(TFT_GREEN, TFT_WHITE);
      tft.print(F("CO2="));
      tft.print(co2);
    }
    else{
```

```
      tft.setTextColor(TFT_RED, TFT_WHITE);
      tft.print(F("CO2="));
      tft.print(co2);
    }
  }
  tft.println(F("ppm"));
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
}

/*##############Cor para o Materiais Particulados####################*/
if(mp > 250){
  tft.setTextColor(TFT_PURPLE, TFT_WHITE);
  tft.print(F("C.Part.="));
  tft.print(particulado);
  tft.println(F("ug/m³"));
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
}
if(mp > 150 and mp <= 250){
  tft.setTextColor(TFT_RED, TFT_WHITE);
  tft.print(F("C.Part.="));
  tft.print(particulado);
  tft.println(F("ug/m³"));
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
}
```

```
  if(mp > 100 and mp <= 150){
    tft.setTextColor(TFT_ORANGE, TFT_WHITE);
    tft.print(F("C.Part.="));
    tft.print(particulado);
    tft.println(F("ug/m³"));
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
  }
  if(mp > 50 and mp <= 100){
    tft.setTextColor(TFT_YELLOW, TFT_WHITE);
    tft.print(F("C.Part.="));
    tft.print(particulado);
    tft.println(F("ug/m³"));
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
  }
  if(mp <= 50){
    tft.setTextColor(TFT_GREEN, TFT_WHITE);
    tft.print(F("C.Part.="));
    tft.print(particulado);
    tft.println(F("ug/m³"));
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
  }
}

/************Calibrando sensores de gás*****************/
```

```
float Calibracao(int sensor){
  int leitura_analogica;
  float resistencia_em_ar_limpo, r0_Calibrado;

  switch (sensor){
    case 7:
      resistencia_em_ar_limpo = 0;
      for (int i = 0; i<100; i++){
        leitura_analogica = analogRead(pin_MQ7);
        resistencia_em_ar_limpo += RL_MQ7 * ((5/(0.001221 * leitura_analogica))-1);
        delay(100);

        Serial.print(F(" Calibrando MQ7: "));
        Serial.print(i);
        Serial.println(F("%"));
      }
      resistencia_em_ar_limpo = resistencia_em_ar_limpo / 100;
      r0_Calibrado = resistencia_em_ar_limpo / (Ajuste_R(36.1286, sensor));
      delay(1000);
      break;

    case 5:
      resistencia_em_ar_limpo = 0;
      for (int i = 0; i<100; i++){
```

```
      leitura_analogica = analogRead(pin_MQ5);
      resistencia_em_ar_limpo += RL_MQ5 * ((5/(0.001221 * leitura_analogica))-1);
      delay(100);

      Serial.print(F(" Calibrando MQ5: "));
      Serial.print(i);
      Serial.println(F("%"));
    }
    resistencia_em_ar_limpo = resistencia_em_ar_limpo / 100;
    r0_Calibrado = resistencia_em_ar_limpo / (Ajuste_R(6.4754, sensor));
    delay(1000);
    break;
  }
   return r0_Calibrado;
}


/************Ajuste_R*************/
float Ajuste_R(float razao, int sensor){
  float ajustado, valor_85, valor_33, umid, temp;

  temp = dht.getTemperature();
  umid = dht.getHumidity();
```

```
if(temp > 50){
  Serial.println(F("Erro, temperatura acima de 50°C."));
  return NAN;
}
if(temp < 0){
  Serial.println(F("Erro, temperatura abaixo de 0°C."));
  return NAN;
}
if(umid > 85){
  Serial.println(F("Erro, umidade acima de 85%."));
  return NAN;
}
if(umid < 33){
  Serial.println(F("Erro, umidade abaixo de 33%."));
}

switch(sensor){
  case 7:
    valor_85 = -0.000002 * pow(temp, 3) + 0.0002 * pow(temp, 2) - 0.0123 * temp + 1.0268;
    valor_33 = -0.000005 * pow(temp, 3) + 0.0005 * pow(temp, 2) - 0.0186 * temp + 1.2112;

    ajustado = razao * (valor_85 + (valor_33 - valor_85) * (33 - umid)/(33-85));
    break;
```

```
    case 5:
      valor_85 = -0.000002 * pow(temp, 3) + 0.0002 * pow(temp, 2) - 0.0114 * temp + 0.9944;
      valor_33 = -0.000005 * pow(temp, 3) + 0.0004 * pow(temp, 2) - 0.016 * temp + 1.1711;

      ajustado = razao * (valor_85 + (valor_33 - valor_85) * (33 - umid)/(33-85));
      break;
   }

  return ajustado;
}

/***********Insere no banco***************/
void Envia_Dados(){
  sprintf(query, INSERT_SQL, String(temperatura), String(umidade), String(coMedio), String(mpMedio), String(glpMedio), String(co2Medio), String(covMedio));
  if (!conn.connect(server_addr,3306, user, password)){
    conn.close();
    Conecta_Banco();
  }
  MySQL_Cursor * cur_mem = new MySQL_Cursor(&conn);
  cur_mem -> execute(query);
  delete cur_mem;
  Serial.println(F("Dados enviados!"));
}
```

```
/*************Conecta no banco*****************/
void Conecta_Banco(){
  Serial.println(F("Conectando ao banco de dados..."));
  while(!conn.connect(server_addr,3306, user, password)){
    Serial.println(F("Conexão falhou!"));
    conn.close();
    delay(100);
    Serial.println(F("Conectando ao Banco novamente."));
  }
  Serial.println(F("Conectado ao Banco de dados"));
}
void Conecta_WiFi(){
  Serial.print(F("Conectando no WiFi..."));
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.print("");
  Serial.print(F("WIFI CONECTADA!"));
  Serial.println("");
}
```

2 – CÓDIGO DO BANCO DE DADOS

```
CREATE DATABASE EstacoesGases;
USE EstacoesGases;
CREATE TABLE prototipo1(
   id SERIAL PRIMARY KEY NOT NULL AUTO_INCREMENT,
   nome VARCHAR(50) NOT NULL,
   data TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
   temperatura DECIMAL(4,1) NULL,
   umidade DECIMAL(4,1) NULL,
   co INT NULL,
   glp INT NULL,
   metano INT NULL,
   v_alcool INT NULL,
   benzeno DECIMAL(10,2) NULL,
   amonia INT NULL,
   sulf_hidrog DECIMAL(10,2) NULL,
   mp DECIMAL(10,2) NULL,
   fumaca DECIMAL(10,2) NULL,
   cov INT NULL,
   co2 INT NULL,
);
```

3 – CÓDIGO DE DISPONIBILIZAÇÃO DE DADOS

```
!pip install mysql-connector-python


import mysql.connector as conn

import pandas as pd

import plotly.graph_objects as go

import plotly.express as px


from datetime import datetime, timedelta


try:

  database = conn.connect(host="**.**.***.***", database="estacoes", user="*******", pass
wd="*******", use_pure=True)

  query = "SELECT * FROM prototipo01;"

  df = pd.read_sql(query, database)

  database.close()

except Exception as e:

  database.close()

  print(str(e))


df
```

```
fig = go.Figure(data=[go.Scatter(x=df['data'], y=df['temperatura'])], layout=go.Layout(title=go.layout.Title(text="Hitórico de Medições de Temperatura")))

fig.update_layout(xaxis_title="Tempo", yaxis_title="°C")

fig.show()


fig = go.Figure(data=[go.Scatter(x=df['data'], y=df['umidade'])], layout=go.Layout(title=go.layout.Title(text="Hitórico de Medições de Umidade do ar")))

fig.update_layout(xaxis_title="Tempo", yaxis_title="Umidade (%)")

fig.show()


fig = px.area(df, x="data", y="co")

fig.update_layout(title="Hitórico de Medições de Monóxido de Carbono",xaxis_title="Tempo", yaxis_title="ppm")

fig.show()


fig = px.area(df, x="data", y="glp")

fig.update_layout(title="Hitórico de Medições de Gás Liquefeito de Petróleo",xaxis_title="Tempo", yaxis_title="ppm")

fig.show()


fig = px.area(df, x="data", y="co2")

fig.update_layout(title="Hitórico de Medições de Dióxido de Carbono",xaxis_title="Tempo", yaxis_title="ppm")

fig.show()
```

```
fig = px.area(df, x="data", y="covt")

fig.update_layout(title="Hitórico de Medições de Composto Orgânico Volátil",xaxis_title="T
empo", yaxis_title="ppb")

fig.show()


fig = px.area(df, x="data", y="mp")

fig.update_layout(title="Hitórico de Medições de Particulas em Suspenção", xaxis_title="Tem
po", yaxis_title="ug/m³")

fig.show()


ultima_hora_medicao = df.iloc[-1]['data']

media_movel_CO = {'data': [], 'co': []}

media_movel_MP = {'data': [], 'mp': []}

media_movel_COVT = {'data': [], 'covt': []}

for i, linha in df.iterrows():


 #Calculando média movel de Monóxido de Carbono

 if (linha['data'] + timedelta(hours = 8)) <= ultima_hora_medicao:

   aux = df[df['data'] <= (linha['data'] + timedelta(hours = 8))]

   aux = aux[aux['data'] >= linha['data']]


   media_movel_CO['co'].append(aux.co.mean())

   media_movel_CO['data'].append(aux.iloc[-1]['data'])


   media_movel_COVT['covt'].append(aux.covt.mean())
```

```
    media_movel_COVT['data'].append(aux.iloc[-1]['data'])


  #Calculando média movel de particulas suspensas
  if (linha['data'] + timedelta(hours = 24)) <= ultima_hora_medicao:
    aux = df[df['data'] <= (linha['data'] + timedelta(hours = 24))]
    aux = aux[aux['data'] >= linha['data']]


    media_movel_MP['mp'].append(aux.mp.mean())
    media_movel_MP['data'].append(aux.iloc[-1]['data'])


media_movel_CO = pd.DataFrame(media_movel_CO)
media_movel_MP = pd.DataFrame(media_movel_MP)
media_movel_COVT = pd.DataFrame(media_movel_COVT)


fig = px.area(media_movel_CO, x="data", y="co")
fig.update_layout(title="Média Móvel de Monóxido de Carbono", xaxis_title="Tempo", yaxis
_title="ppm")
fig.show()


fig = px.area(media_movel_COVT, x="data", y="covt")
fig.update_layout(title="Média Móvel de Composto Orgânico Volátil",xaxis_title="Tempo",
yaxis_title="ppb")
fig.show()


fig = px.area(media_movel_MP, x="data", y="mp")
```

```
fig.update_layout(title="Média Móvel de Particulas em Suspensão", xaxis_title="Tempo", ya
xis_title="ug/m³")

fig.show()


dicionario ={ 'Qualidade':['Boa', 'Moderada', 'Ruim', 'Muito Ruim', 'Péssima'],
            'Indice Inicial': [0, 40, 80, 120, 200],
            'Indice Final': [40, 80, 120, 200, 300],
            'MP10 Inicial': [0, 50, 100, 150, 250],
            'MP10 Final': [50, 100, 150, 250, 10000],
            'MP25 Inicial': [0, 25, 50, 75, 125],
            'MP25 Final': [25, 50, 75, 125, 10000],
            'CO Inicial': [0, 9, 11, 13,15],
            'CO Final': [9, 11, 13, 15, 10000]}
tabela_qualidade = pd.DataFrame(dicionario)
tabela_qualidade


def calculo_iqar_CO(ultimo_valor):
  for i, linha in tabela_qualidade.iterrows():
    if ultimo_valor < linha['CO Final'] and ultimo_valor >= linha['CO Inicial']:
      if linha['Qualidade'] == 'Péssima':
        iqar_CO = 400
      else:
        iqar_CO = linha['Indice Inicial'] + ((linha['Indice Final'] - linha['Indice Inicial']) / (linha['
CO Final'] - linha['CO Inicial'])) * (ultimo_valor - linha['CO Inicial'])
```

```
  return iqar_CO


def verifica_qualidade_CO(iqar_CO):
  for e, linha2 in tabela_qualidade.iterrows():
    if iqar_CO >= linha2['Indice Inicial'] and iqar_CO <= linha2['Indice Final']:
      return linha2['Qualidade']


lista_IQAr_CO = []
lista_qualidade_CO = []
for i, linha in media_movel_CO.iterrows():
  iqar = calculo_iqar_CO(linha['co'])
  lista_IQAr_CO.append(iqar)
  lista_qualidade_CO.append(verifica_qualidade_CO(iqar))
media_movel_CO['IQAr'] = lista_IQAr_CO
media_movel_CO['Qualidade'] = lista_qualidade_CO


def calculo_iqar_MP10(ultimo_valor):
  for i, linha in tabela_qualidade.iterrows():
    if ultimo_valor < linha['MP10 Final'] and ultimo_valor >= linha['MP10 Inicial']:
      if linha['Qualidade'] == 'Péssima':
        iqar_MP10 = 400
      else:
```

```
        iqar_MP10 = linha['Indice Inicial'] + ((linha['Indice Final'] - linha['Indice Inicial']) / (linh
a['MP10 Final'] - linha['MP10 Inicial'])) * (ultimo_valor - linha['MP10 Inicial'])



  return iqar_MP10


def calculo_iqar_MP25(ultimo_valor):
  for i, linha in tabela_qualidade.iterrows():
    if ultimo_valor < linha['MP25 Final'] and ultimo_valor >= linha['MP25 Inicial']:
      if linha['Qualidade'] == 'Péssima':
        iqar_MP25 = 400
      else:
        iqar_MP25 = linha['Indice Inicial'] + ((linha['Indice Final'] - linha['Indice Inicial']) / (linh
a['MP25 Final'] - linha['MP25 Inicial'])) * (ultimo_valor - linha['MP25 Inicial'])



  return iqar_MP25


def verifica_qualidade_MP10(iqar_MP10):
  for e, linha2 in tabela_qualidade.iterrows():
    if iqar_MP10 >= linha2['Indice Inicial'] and iqar_MP10 <= linha2['Indice Final']:
      return linha2['Qualidade']


def verifica_qualidade_MP25(iqar_MP25):
  for e, linha2 in tabela_qualidade.iterrows():
```

```
    if iqar_MP25 >= linha2['Indice Inicial'] and iqar_MP25 <= linha2['Indice Final']:
      return linha2['Qualidade']


lista_IQAr_MP10 = []
lista_qualidade_MP10 = []
for i, linha in media_movel_MP.iterrows():
  iqar = calculo_iqar_MP10(linha['mp'])
  lista_IQAr_MP10.append(iqar)
  lista_qualidade_MP10.append(verifica_qualidade_MP10(iqar))
media_movel_MP['IQAr MP10'] = lista_IQAr_MP10
media_movel_MP['Qualidade MP10'] = lista_qualidade_MP10


lista_IQAr_MP25 = []
lista_qualidade_MP25 = []
for i, linha in media_movel_MP.iterrows():
  iqar = calculo_iqar_MP25(linha['mp'])
  lista_IQAr_MP25.append(iqar)
  lista_qualidade_MP25.append(verifica_qualidade_MP25(iqar))
media_movel_MP['IQAr MP25'] = lista_IQAr_MP25
media_movel_MP['Qualidade MP25'] = lista_qualidade_MP25


mmco_boa = []
mmco_moderada =[]
mmco_ruim = []
```

```
mmco_muito_ruim = []
mmco_pessima = []
for i, linha in media_movel_CO.iterrows():
  if linha['Qualidade'] == "Boa":
    mmco_boa.append(linha['IQAr'])
    mmco_moderada.append(0)
    mmco_ruim.append(0)
    mmco_muito_ruim.append(0)
    mmco_pessima.append(0)
  elif linha['Qualidade'] == "Moderada":
    mmco_boa.append(0)
    mmco_moderada.append(linha['IQAr'])
    mmco_ruim.append(0)
    mmco_muito_ruim.append(0)
    mmco_pessima.append(0)
  elif linha['Qualidade'] == "Ruim":
    mmco_boa.append(0)
    mmco_moderada.append(0)
    mmco_ruim.append(linha['IQAr'])
    mmco_muito_ruim.append(0)
    mmco_pessima.append(0)
  elif linha['Qualidade'] == "Muito Ruim":
    mmco_boa.append(0)
    mmco_moderada.append(0)
```

```
    mmco_ruim.append(0)
    mmco_muito_ruim.append(linha['IQAr'])
    mmco_pessima.append(0)
  else:
    mmco_boa.append(0)
    mmco_moderada.append(0)
    mmco_ruim.append(0)
    mmco_muito_ruim.append(0)
    mmco_pessima.append(linha['IQAr'])

media_movel_CO['BOA'] = mmco_boa
media_movel_CO['MODERADA'] = mmco_moderada
media_movel_CO['RUIM'] = mmco_ruim
media_movel_CO['MUITO RUIM'] = mmco_muito_ruim
media_movel_CO['PESSIMA'] = mmco_pessima

fig = go.Figure()
fig.add_trace(go.Scatter(x=media_movel_CO['data'], y=media_movel_CO['BOA'], fill='tozeroy', name='Qualidade Boa', mode='lines', line=dict(width=0.8,color = 'rgb(0, 255, 0)'), fillcolor = 'rgba(0, 255, 0,0.7)'))
fig.add_trace(go.Scatter(x=media_movel_CO['data'], y=media_movel_CO['MODERADA'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 200, 86)'), name='Qualidade Moderada', fillcolor = 'rgb(255, 200, 86)'))
fig.add_trace(go.Scatter(x=media_movel_CO['data'], y=media_movel_CO['RUIM'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 165, 0)'), name='Qualidade Ruim', fillcolor = 'rgba(255, 165, 0,0.8)'))
```

```
fig.add_trace(go.Scatter(x=media_movel_CO['data'], y=media_movel_CO['MUITO RUIM'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 0, 0)'), name='Qualidade Muito Ruim', fillcolor = 'rgba(255, 0, 0, 0.8)'))

fig.add_trace(go.Scatter(x=media_movel_CO['data'], y=media_movel_CO['PESSIMA'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(128, 0, 128)'), name='Qualidade Péssima', fillcolor = 'rgba(128, 0, 128, 0.8)'))

fig.update_layout(title="Índice de Qualidade do Ar de Monóxido de Carbono", xaxis_title="Tempo", yaxis_title="")

fig.show()


mmmp10_boa = []

mmmp10_moderada =[]

mmmp10_ruim = []

mmmp10_muito_ruim = []

mmmp10_pessima = []

for i, linha in media_movel_MP.iterrows():

 if linha['Qualidade MP10'] == "Boa":

   mmmp10_boa.append(linha['IQAr MP10'])

   mmmp10_moderada.append(0)

   mmmp10_ruim.append(0)

   mmmp10_muito_ruim.append(0)

   mmmp10_pessima.append(0)

 elif linha['Qualidade MP10'] == "Moderada":

   mmmp10_boa.append(0)

   mmmp10_moderada.append(linha['IQAr MP10'])

   mmmp10_ruim.append(0)
```

```
    mmmp10_muito_ruim.append(0)
    mmmp10_pessima.append(0)
  elif linha['Qualidade MP10'] == "Ruim":
    mmmp10_boa.append(0)
    mmmp10_moderada.append(0)
    mmmp10_ruim.append(linha['IQAr MP10'])
    mmmp10_muito_ruim.append(0)
    mmmp10_pessima.append(0)
  elif linha['Qualidade MP10'] == "Muito Ruim":
    mmmp10_boa.append(0)
    mmmp10_moderada.append(0)
    mmmp10_ruim.append(0)
    mmmp10_muito_ruim.append(linha['IQAr MP10'])
    mmmp10_pessima.append(0)
  else:
    mmmp10_boa.append(0)
    mmmp10_moderada.append(0)
    mmmp10_ruim.append(0)
    mmmp10_muito_ruim.append(0)
    mmmp10_pessima.append(linha['IQAr MP10'])

media_movel_MP['MP10 BOA'] = mmmp10_boa
media_movel_MP['MP10 MODERADA'] = mmmp10_moderada
media_movel_MP['MP10 RUIM'] = mmmp10_ruim
```

```
media_movel_MP['MP10 MUITO RUIM'] = mmmp10_muito_ruim

media_movel_MP['MP10 PESSIMA'] = mmmp10_pessima


fig = go.Figure()

fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP10 BOA'], fill
='tozeroy', name='Qualidade Boa', mode='lines', line=dict(width=0.8,color = 'rgb(0, 255, 0)'),
fillcolor = 'rgba(0, 255, 0,0.7)'))

fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP10 MODERA
DA'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 200, 86)'), name='Qua
lidade Moderada', fillcolor = 'rgb(255, 200, 86)'))

fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP10 RUIM'], fil
l='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 165, 0)'), name='Qualidade Ru
im', fillcolor = 'rgba(255, 165, 0,0.8)'))

fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP10 MUITO R
UIM'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 0, 0)'), name='Qualid
ade Muito Ruim', fillcolor = 'rgba(255, 0, 0, 0.8)'))

fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP10 PESSIMA'
], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(128, 0, 128)'), name='Qualidad
e Péssima', fillcolor = 'rgba(128, 0, 128, 0.8)'))

fig.update_layout(title="Índice de Qualidade do Ar para Particulas em Suspenção MP10", xax
is_title="Tempo", yaxis_title="")

fig.show()


mmmp25_boa = []

mmmp25_moderada =[]

mmmp25_ruim = []

mmmp25_muito_ruim = []
```

```
mmmp25_pessima = []
for i, linha in media_movel_MP.iterrows():
  if linha['Qualidade MP25'] == "Boa":
    mmmp25_boa.append(linha['IQAr MP25'])
    mmmp25_moderada.append(0)
    mmmp25_ruim.append(0)
    mmmp25_muito_ruim.append(0)
    mmmp25_pessima.append(0)
  elif linha['Qualidade MP25'] == "Moderada":
    mmmp25_boa.append(0)
    mmmp25_moderada.append(linha['IQAr MP25'])
    mmmp25_ruim.append(0)
    mmmp25_muito_ruim.append(0)
    mmmp25_pessima.append(0)
  elif linha['Qualidade MP25'] == "Ruim":
    mmmp25_boa.append(0)
    mmmp25_moderada.append(0)
    mmmp25_ruim.append(linha['IQAr MP25'])
    mmmp25_muito_ruim.append(0)
    mmmp25_pessima.append(0)
  elif linha['Qualidade MP25'] == "Muito Ruim":
    mmmp25_boa.append(0)
    mmmp25_moderada.append(0)
    mmmp25_ruim.append(0)
```

```
    mmmp25_muito_ruim.append(linha['IQAr MP25'])
    mmmp25_pessima.append(0)
  else:
    mmmp25_boa.append(0)
    mmmp25_moderada.append(0)
    mmmp25_ruim.append(0)
    mmmp25_muito_ruim.append(0)
    mmmp25_pessima.append(linha['IQAr MP25'])

media_movel_MP['MP25 BOA'] = mmmp25_boa
media_movel_MP['MP25 MODERADA'] = mmmp25_moderada
media_movel_MP['MP25 RUIM'] = mmmp25_ruim
media_movel_MP['MP25 MUITO RUIM'] = mmmp25_muito_ruim
media_movel_MP['MP25 PESSIMA'] = mmmp25_pessima

fig = go.Figure()
fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP25 BOA'], fill
='tozeroy', name='Qualidade Boa', mode='lines', line=dict(width=0.8,color = 'rgb(0, 255, 0)'),
fillcolor = 'rgba(0, 255, 0,0.7)'))
fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP25 MODERA
DA'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 200, 86)'), name='Qua
lidade Moderada', fillcolor = 'rgb(255, 200, 86)'))
fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP25 RUIM'], fil
l='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 165, 0)'), name='Qualidade Ru
im', fillcolor = 'rgba(255, 165, 0,0.8)'))
```

```
fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP25 MUITO RUIM'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(255, 0, 0)'), name='Qualidade Muito Ruim', fillcolor = 'rgba(255, 0, 0, 0.8)'))

fig.add_trace(go.Scatter(x=media_movel_MP['data'], y=media_movel_MP['MP25 PESSIMA'], fill='tozeroy',mode='lines', line=dict(width=0.8, color = 'rgb(128, 0, 128)'), name='Qualidade Péssima', fillcolor = 'rgba(128, 0, 128, 0.8)'))

fig.update_layout(title="Índice de Qualidade do Ar para Particulas em Suspenção MP25", xaxis_title="Tempo", yaxis_title="")

fig.show()


df.to_csv("Dados do Banco.csv")

df.to_excel("Dados do Banco.xlsx")
```