Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica
Graduação em Engenharia Biomédica

LARYSSA MOREIRA MORAES

APLICAÇÃO IOT NA DETECÇÃO DE RUÍDOS EM UNIDADES DE TERAPIA INTENSIVA NEONATAL

Uberlândia
2023

LARYSSA MOREIRA MORAES

APLICAÇÃO IOT NA DETECÇÃO DE RUÍDOS EM UNIDADES DE TERAPIA INTENSIVA NEONATAL

Trabalho apresentado como requisito parcial de avaliação na disciplina Trabalho de Conclusão de Curso de Engenharia Biomédica da Universidade Federal de Uberlândia.

Orientador: Sérgio Ricardo de Jesus Oliveira

_____

Assinatura do Orientador

Uberlândia
2023

AGRADECIMENTOS

Agradeço a Deus, por me dar forças para realizar meus sonhos.
Aos meus pais, por me dar ferramentas para chegar até aqui.
Ao Rodrigo, pela companhia e confiança em todo o processo.
Aos meus amigos, por me acompanharem e ajudarem no caminho, fazendo dele um pouco mais fácil, com atenção especial para a Isabela, Myllena e Matheus.
À Ana Paula Michalichem, Nayara Dayrell de Carvalho e ao Hospital de Clínicas de Uberlândia, pela inspiração para este projeto.
E agradeço ao meu orientador, Professor Dr. Sérgio Ricardo de Jesus Oliveira, pela compreensão, confiança e ensinamentos.

Email do autor: laryssam6@hotmail.com

RESUMO

O desenvolvimento fisiológico, psicológico e neurológico de um recém-nascido prematuro é profundamente influenciado pelo ambiente em que ele é inserido. Quando um bebê nasce prematuro e requer cuidados de saúde específicos, como os oferecidos em uma Unidade de Terapia Intensiva Neonatal (UTINEO), a importância do ambiente se torna ainda mais significativa. Entre os vários fatores que afetam o desenvolvimento desses bebês, destacam-se os ruídos sonoros presentes no ambiente, que podem ter um impacto tanto positivo quanto negativo em seu desenvolvimento, conforme evidenciado por estudos. Quando os níveis de ruído sonoro, como aqueles gerados por máquinas e equipamentos médicos, ultrapassam um certo limiar, os efeitos se tornam prejudiciais. Esses ruídos sonoros excessivos podem afetar parâmetros fisiológicos do bebê prematuro, resultando em consequências desfavoráveis que podem impactar no seu desenvolvimento. Portanto, a monitoração constante dos níveis de ruído sonoro em uma UTINEO é essencial para garantir que o ambiente seja propício ao bem-estar e ao desenvolvimento saudável desses pacientes. Nesse contexto, este estudo buscou desenvolver um protótipo capaz de monitorar os níveis de ruído na UTINEO e notificar o usuário sempre que esses níveis ultrapassassem os limites considerados seguros. Para alcançar esse objetivo, foi empregado o protocolo MQTT, que possibilita a implementação da Internet das Coisas (IoT) no projeto, permitindo a disponibilização em tempo real dos dados de monitoramento de ruído sonoro. Portanto, o projeto visa criar uma ferramenta para fornecer um ambiente controlado e seguro para bebês prematuros, onde a tecnologia IoT desempenha um papel fundamental na monitoração e gestão dos níveis de ruído, contribuindo assim para o desenvolvimento saudável desses recém-nascidos na UTINEO.

ABSTRACT

The physiological, psychological, and neurological development of a premature newborn is profoundly influenced by the environment in which they are placed. When a baby is born prematurely and requires specific healthcare in a Neonatal Intensive Care Unit (NICU), the importance of the environment becomes even more significant. Among the various factors affecting the development of these infants, the sound noises present in the environment, as evidenced by studies, can have both positive and negative impacts on their development. When sound noise levels, such as those generated by machines and medical equipment, exceed a certain threshold, harmful effects become evident. These excessive sound noises can affect the physiological parameters of the premature baby, leading to unfavorable consequences that may impact their development. Therefore, constant monitoring of sound noise levels in a NICU is essential to ensure that the environment is conducive to the well-being and healthy development of these patients. In this context, this study aimed to develop a prototype capable of monitoring sound noise levels in the NICU and notifying the user whenever these levels exceeded the considered safe limits. To achieve this goal, the MQTT protocol was employed, enabling the implementation of the Internet of Things (IoT) in the project, allowing real-time availability of noise monitoring data. Thus, the project aims to create a tool to provide a controlled and safe environment for premature babies, where IoT technology plays a fundamental role in the monitoring and management of noise levels, thereby contributing to the healthy development of these newborns in the NICU.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	-	*Application Programming Interface*
dB	-	Decibel
dBA	-	Decibel ponderado em A
dBFS	-	Decibel ao fundo de escala
dB SPL	-	Decibel ao nível de pressão sonora
HTTP	-	*Hypertext Transfer Protocol*
Hz	-	Hertz
I2S	-	*Integrated Inter-IC Sound Bus*
IoT	-	*Internet of Things*
M2M	-	*Machine-to-machine*
MEMS	-	*Micro-Electro-Mechanical Systems*
RN	-	Recém - nascido
SCK	-	Clock
SD	-	*Serial Data*
UTINEO	-	Unidade de Terapia Intensiva Neonatal
WS	-	*Word Select*
WWW	-	*World Wide Web*

SUMÁRIO

1. INTRODUÇÃO

1.1. Motivação

Os ruídos têm um grande efeito no desenvolvimento de um recém-nascido (RN), principalmente se este for prematuro, podendo ser benéfico ou não. Estudos comprovam que sons como os provindos dos pais, suas vozes, podem auxiliar no desenvolvimento motor e de linguagem, e, quando ainda no útero, auxilia no desenvolvimento e na maturação de sistemas sensoriais. Do outro lado, quando o prematuro se encontra em uma Unidade de Terapia Intensiva Neonatal (UTINEO), por exemplo, ruídos como alarmes, conversas, circulação de pessoas, equipamentos em funcionamento, quando em excesso, podem apresentar um risco para o desenvolvimento do bebê [1, 2, 3].

A UTINEO tem o papel de ser um ambiente propício para o conforto, desenvolvimento e cura dos bebês, sendo assim, existem parâmetros que ela deve ser capaz de fornecer para que isso aconteça. Temperatura, suporte à vida e possibilidades de tratamentos são exemplos desses parâmetros. Porém, esses fatores trazem com eles os ruídos [2].

Há uma preocupação crescente com a exposição a elevados níveis de ruído nas UTINEO. Estudos têm demonstrado que o ruído excessivo pode causar efeitos fisiológicos e comportamentais adversos em bebês prematuros, incluindo aumento da frequência cardíaca, estresse e distúrbios do sono [4].

Bebês prematuros apresentam limitações fisiológicas e a imaturidade de diversos sistemas biológicos, como o sistema nervoso central, sendo mais suscetíveis aos efeitos do ambiente. Quando expostos ao estresse, neste caso, o excesso de barulho, há alterações na frequência respiratória e cardíaca, pressão sanguínea e saturação de oxigênio. Além dos efeitos diretos desses sintomas, há também o aumento de gasto calórico e toda a energia que seria direcionada para o desenvolvimento, agora será gasta nessas alterações, devido ao aumento da frequência cardíaca e aumento do consumo de oxigênio. A longo prazo, a exposição pode causar perda auditiva e pode estar associada a diferentes distúrbios de desenvolvimento [5,6,7,8].

Além dos efeitos diretos, ruídos excessivos também estão associados com o aumento do número de erros e acidentes entre a equipe de saúde. Portanto, reduzir os níveis de ruído e manter um ambiente silencioso dentro da UTINEO é fundamental para promover o desenvolvimento saudável e a recuperação dos pacientes [4].

A norma ABNT NBR 10152:2017, que fala sobre níveis de pressão sonora em ambientes determina que para berçários, os níveis devem ser de, no máximo, 45 dBA, considerando a

tolerância. E a *American Academy of Pediatrics* recomenda que, em UTINEO, os níveis de ruído devem ser monitorados e mantidos abaixo de 45 dB [9, 10].

1.2. Objetivos

Considerando as motivações expostas, o objetivo deste trabalho foi desenvolver um sistema que utilizasse Internet das Coisas (IoT), por meio do protocolo MQTT, e fosse capaz de monitorar níveis de ruído em um ambiente e gerar uma notificação em um aplicativo mobile caso estes níveis estivessem acima do determinado, gerando uma ferramenta que permitisse a adequação do ambiente da UTINEO e provendo qualidade de tratamento e auxiliando no desenvolvimento do bebê.

Assim, os objetivos específicos deste projeto podem ser resumidos nos seguintes tópicos:

- Utilizar um sensor para captar níveis de ruído em um ambiente;
- Utilizar MQTT para enviar os dados obtidos para um sistema de armazenamento;
- Notificar o usuário por meio de uma aplicação mobile;

2. FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo serão abordados alguns conceitos chaves para o entendimento do projeto.

2.1. IoT e o protocolo MQTT

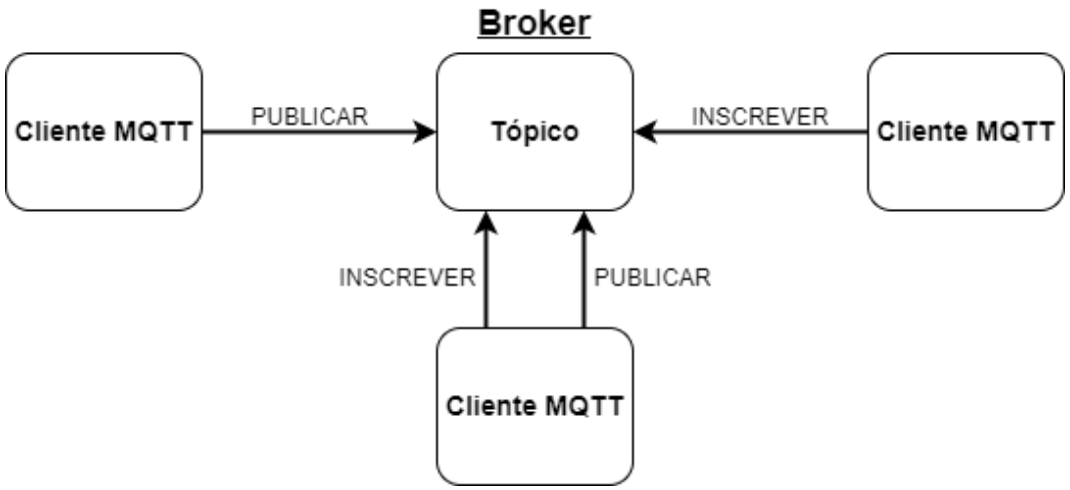
Internet das Coisas é o termo dado a uma rede de dispositivos e componentes interconectados se comunicando entre si e/ou com a nuvem. Estes dispositivos, como relógios, cortinas e geladeiras, fazem uso de sensores e atuadores para responder e atender os seus usuários, disponibilizando os dados coletados de forma otimizada. Este conceito já é posto em prática desde a década de 1990 em objetos cotidianos e vem acompanhando o ritmo da evolução tecnológica, sendo cada vez mais inserido no dia a dia da comunidade [11].

Uma forma de se trabalhar com Internet das Coisas é utilizando MQTT. MQTT é um protocolo de comunicação leve ideal para comunicações machine-to-machine (M2M) e IoT. Inventado por Andy Stanford-Clark (IBM) e Arlen Nipper (Arcom, now Cirrus Link) para situações em que o ambiente possui recursos limitados, é um protocolo de fácil implementação e agnóstico de dados [12].

O MQTT segue o padrão *publish/subscribe*, ou seja, publicação/inscrição. Este padrão consiste no transporte de pacotes de mensagens entre os publicadores e os inscritos, sendo

intermediado pelo servidor, o broker. Neste modelo, um cliente se inscreve em um tópico, enviando um pacote de inscrição ao broker, o broker armazena essa informação e envia um pacote de volta confirmando a inscrição do cliente, nesses pacotes estarão configurações relacionadas a forma como futuras trocas de dados ocorrerão. Este cliente será conhecido como inscrito (*subscriber*). Do outro lado, um segundo cliente envia uma mensagem a ser publicada em algum tópico para o broker, este cliente será conhecido como publicador (*publisher*). O broker, ao receber a mensagem, deverá enviá-la para os inscritos do tópico ao qual ela foi publicada, não tomando conhecimento do conteúdo, e somente da forma como essas mensagens devem ser transportadas e para quem. Como o principal papel do broker é a administração dos pacotes a serem transferidos, sua capacidade de armazenamento é limitada, sendo dedicada principalmente para mensagens que podem ter sido perdidas. O mecanismo básico de funcionamento do MQTT está ilustrado na Figura 1 [12].

Figura 1 - Diagrama de funcionamento do modelo *publish/subscribe*



Os publicadores não entram em contato direto com os inscritos e toda a administração das mensagens é feita pelo broker, possibilitando uma separação temporal já que publicador e inscrito não precisam estar funcionando simultaneamente uma separação espacial, pois eles não precisam ter conhecimento um do outro. Além disso, é promovida ainda uma separação de sincronização entre os clientes, como cada um não precisa interromper suas funcionalidades ao publicar ou receber mensagens [12].

O MQTT trabalha com níveis de qualidade de serviço, estes são selecionados na publicação de uma mensagem e quando um cliente se inscreve em um tópico. Os níveis de qualidade de serviço determinam qual a confiabilidade da entrega das mensagens do respectivo

tópico podendo ser 0 (mensagem enviada no máximo uma vez, sem confirmação de recebimento), 1 (mensagem enviada ao menos uma vez, podendo ser recebida mais de uma vez) ou 2 (mensagem enviada exatamente uma vez, com confirmação de entrega) [12].

No projeto, o protocolo MQTT foi utilizado para projetar a comunicação entre o hardware do projeto com o banco de dados e broker utilizado, e entre ele e o aplicativo mobile. Nessa mecânica, o hardware é o publicador e o aplicativo mobile é o inscrito. E o tópico é nomeado de acordo com o local no qual o hardware será inserido.

2.2. Adafruit IO

Adafruit IO é um serviço de nuvem feito para fornecer ferramentas de visualização e manipulação de dados para projetos. Algumas das ferramentas são fornecidas no formato de API (*Application Programming Interface)*, ou seja, uma interface que integra dois softwares para compartilhamento de funcionalidades e/ou se comunicando um com o outro [13,14]

Uma das ferramentas é a API MQTT, que permite a conexão dos *feeds* com diferentes sistemas utilizando o protocolo MQTT, disponibilizando os dados online, em tempo real e permitindo também que o usuário os publique a partir de sua plataforma [15].

Além da API MQTT, o Adafruit IO também disponibiliza uma API HTTP (*Hypertext Transfer Protocol*), permitindo que requisições HTTP sejam feitas para os dados do *feed*, como requisições de grupos de dados ou dados específicos [16].

Como base da estrutura do Adafruit IO têm-se os *feeds.* Eles armazenam os dados e informações relacionadas a eles, podendo funcionar como tópicos para o protocolo MQTT, e assim foram utilizados no projeto [17,18].

Os serviços anteriormente citados estão disponibilizados na versão gratuita da plataforma, apresentando maior taxa de envio de dados e outras funcionalidades na versão paga [17,18].

2.3. Integrated Inter-IC Sound Bus (I2S)

Integrated Inter-IC Sound Bus (I2S) é um protocolo de comunicação entre circuitos integrados focado em transporte de dados de som. Criado na década de 1980 pela *Philips Semiconductors* (atualmente NPX) com o objetivo de facilitar o desenvolvimento de equipamentos e componentes digitais e de conversão analógico-digital criando uma interface única de comunicação [19].

Este protocolo utiliza três linhas para a conexão entre os componentes: *serial data* (SD),

word select (WS) e *clock* (SCK). SD é a linha destinada para o transporte de dados, WS é a linha que indica qual canal de aúdio transmitirá (direito ou esquerdo, já que se trata de uma comunicação estéreo) e SCK é o sinal de *clock* [19].

Este protocolo é a forma com a qual, neste projeto, ocorre a comunicação entre microfone e placa de desenvolvimento, que será responsável por enviar os dados ao broker.

2.4. Processamento de sinal: filtragem e equalização

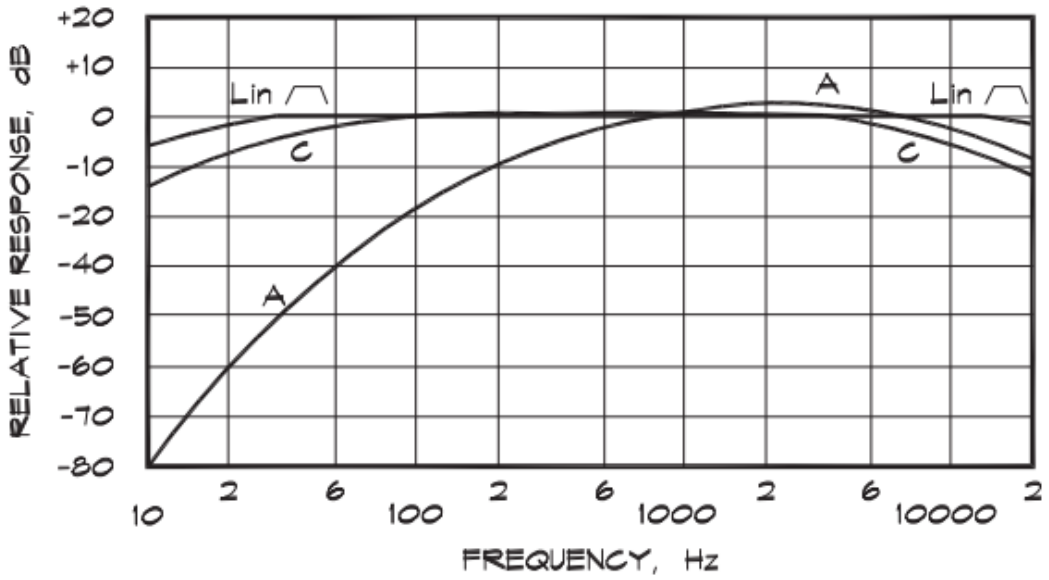
Existem diversas formas de se processar um sinal. Uma etapa essencial é a filtragem. Sua função é alterar os componentes de frequência do sinal, ou seja, quais frequências formam o sinal, selecionando quais delas estarão no sinal final. A filtragem permite selecionar quais frequências não serão afetadas pelos filtros enquanto quais são as indesejadas, que serão atenuadas e pode ser utilizada para equalizar um sinal de áudio, ou seja, remover componentes do sinal de forma a melhorar o desempenho de equipamentos de som no geral [20, 21].

Uma filtragem muito utilizada no processamento de áudio é a com filtros de segunda ordem. Filtros de segunda ordem possuem transições mais bruscas entre a faixa de frequência desejada e a indesejada do que filtros de primeira ordem [21].

Para sinais de áudio, há também as curvas de ponderação. Elas são filtros que atribuem pesos diferentes a frequências sonoras específicas, fazendo com que os seus respectivos níveis de pressão sonora sejam reduzidos ou não. A audição humana percebe de forma diferente certas faixas de frequência, tendo menor percepção de frequências baixas e altas. O intervalo de frequência que o ser humano consegue diferenciar é de 20 Hz a 20kHz, e por isso se faz necessário atribuir pesos diferentes a frequências diferentes [22, 23].

Há 3 tipos de curvas de ponderação, a ponderação A, C e Z. A ponderação A mimetiza o comportamento do ouvido humano, ressaltando o intervalo de frequência mais bem captado pelo ouvido, sua representação é o dBA. A ponderação C é usada para situações em que as frequências fora do intervalo de audição humana sejam também consideradas, já que para certos níveis de pressão sonora, o ouvido se torna mais sensível a elas. E a ponderação Z é usada em cenários onde é necessária a avaliação de equipamentos já que atribui pesos iguais a todas as frequências, e por isso também é conhecida como curva de ponderação plana. O comportamento delas pode ser observado na Figura 2, onde cada curva representa um tipo de ponderação. O eixo x representa a frequência sonora e o eixo y apresenta a resposta relativa de cada componente de frequência [22].

Figura 2 - Curvas de Ponderação.



Fonte: Long (2005)

Os dados do projeto devem ser processados e tratados antes de enviados ao broker, a equalização e ponderação são as formas indicadas para que isso seja feito.

2.5.Conversão de escala digital para decibel

Os dados obtidos de sensores muitas vezes devem ser convertidos de uma escala digital para a escala desejada de processamento e utilização deles. Como o projeto está inserido no contexto de ruídos sonoros, a conversão julgada como adequada é de escala digital para decibel.

Decibel é uma representação em escala logarítmica da razão entre dois valores. No contexto sonoro, ele é utilizado na medição da intensidade do som, ou seja, quanto maior, maior seu “volume”. Ele mede a relação entre os picos e depressões de uma onda sonora. O ouvido humano, quando expostos a sons com mais de 110 dB, pode sofrer danos significativos [25].

A equação padrão para conversão para decibel no contexto de pressão sonora é:

$$\boldsymbol{SPL = 20 \log_{10}\left(\frac{P}{P_0}\right)} \qquad (1)$$

SPL é o representado em dB, sendo que P é a pressão em questão e P_0 é a pressão de referência. Neste caso, a função logarítmica é acompanhada do coeficiente 20, pois a intensidade tem uma relação proporcional com o quadrado da amplitude da pressão [26].

Decibel, por si só, não é uma unidade de medida, devendo ser acompanhado da unidade

referente ao contexto que foi aplicado, como decibel de nível de pressão sonora (dBSPL), decibel ponderado em A (dBA) ou decibel de fundo de escala (dBFS). Decibel de nível de pressão sonora indica o nível em relação a pressão sonora, 0 dB SPL é equivalente a 20 µPa. Decibel de fundo de escala é relacionada a proporção em bits, 0 dBFS é o número máximo de bits do componente em questão. E decibel ponderado em A é o dB relacionado com a pressão sonora, mas filtrado com a curva de ponderação A [27, 22].

Para se converter de uma escala digital, ou seja, valores em bits, para decibel, deve se considerar o valor máximo que pode ser captado pelo sensor em questão (máxima entrada acústica), que corresponderá ao valor máximo digital, o valor mínimo que pode ser captado (ruído de entrada equivalente), a sensibilidade do sensor e a referência na qual essa sensibilidade foi obtida [28].

2.6. Microfones MEMS

Microfones MEMS (sistema microeletromecânico) são componentes constituídos por um sensor acústico e um circuito integrado de conversão para digital acoplados um ao outro. O sensor converte o estímulo de entrada, pressão sonora, em variações de capacitância, produzindo uma variação no sinal analógico que é convertido em digital pelo circuito integrado. Estes microfones são mais baratos e pequenos dispondo de uma grande qualidade de som [29].

Como microfones MEMS raramente são ideais, sua resposta em frequência precisa ser equalizada e ponderada [28].

2.7. Ruído de impacto e ruído contínuo

No contexto de ruídos sonoros, é necessário selecionar qual tipo de ruído será tratado. Cada tipo de ruído advém de algum evento e estes podem ser classificados.

Ruídos de impacto, por exemplo, são ruídos que ocorrem em intervalos maiores que um segundo, com seus picos de energia apresentando duração menor que um segundo. Enquanto ruídos contínuos são todos aqueles que não são de impacto [30, 31].

2.8.Protocolo HTTP

O HTTP (Protocolo de Transferência de Hipertexto) é um protocolo de comunicação que segue o modelo cliente-servidor. Ele funciona por meio de solicitações, que são enviadas pelo cliente, ao servidor web, que as responde com mensagens de estrutura específica que serão

processadas por navegadores e programas formando o pacote final a ser recebido pelo cliente. É um dos protocolos principais utilizados em sites da *World Wide Web* (WWW), podendo também ser empregados em aplicações. Neste protocolo, as requisições são sempre iniciadas pelo cliente, ou seja, um dado só pode ser enviado pelo servidor caso o cliente o tenha requisitado [32, 33].

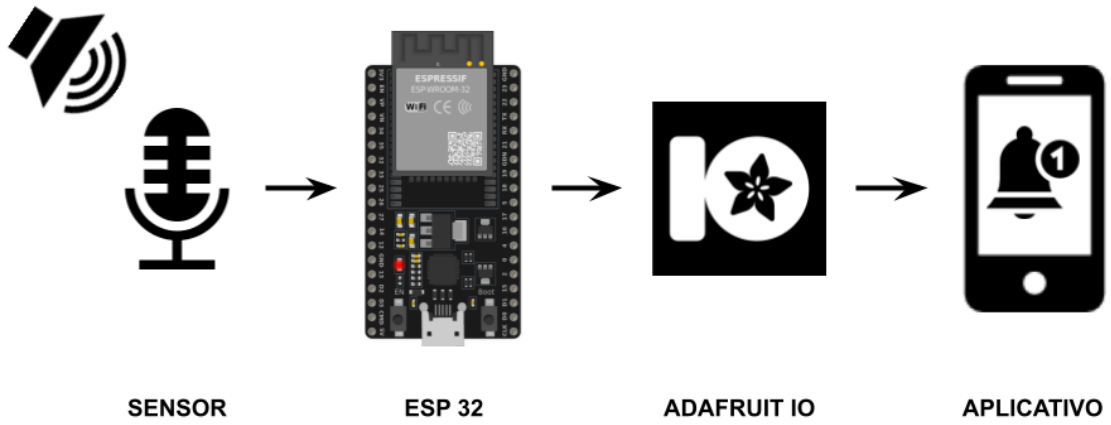
Este protocolo pode ser utilizado para requisições de arquivos de dados, permitindo que o usuário receba as informações na forma de um histórico, por exemplo.

3. DESENVOLVIMENTO

3.1. Estrutura do projeto

Para atingir os objetivos anteriormente citados, optou-se por um projeto em que um microfone irá se conectar com um microcontrolador e transmitir para eles os sinais captados do ambiente. O microcontrolador deverá processar os dados recebidos do sensor, fazendo os devidos tratamentos e os enviando, via protocolo MQTT, para um tópico específico no broker, o Adafruit IO, este que, por sua vez, transmitirá os dados para todos aqueles que estiverem conectados a ele, também via MQTT. O aplicativo, que deverá estar conectado ao broker, receberá os dados, plotando um gráfico e notificando o usuário quando os níveis de ruído estiverem acima do determinado. Também se considerou válida a geração de uma funcionalidade para compartilhar um grupo de dados obtidos do banco de dados disponibilizado no Adafruit IO. A Figura 3 representa a estrutura base do projeto de ponta a ponta.

Figura 3 - Estrutura do protótipo



3.2.Projeto de *Hardware*

Para o hardware, buscou-se uma estrutura simples, sendo composto por um microfone INMP441 e pelo microcontrolador ESP32 [34].

O microfone INMP441 utiliza o protocolo I2S para se comunicar com outros componentes e possui baixo consumo energético. Seu sinal de saída já é condicionado e convertido de analógico para digital. Este componente se trata de um microfone MEMS omnidirecional, ou seja, capta sons em todas as direções [35].

A utilização do INMP441 descarta a necessidade de um sistema codificador de áudio por utilizar a interface I2S e poder ser ligado diretamente ao ESP32 [35].

O ESP32 é um microcontrolador com conectividade Wi-Fi e Bluetooth integrada, de baixo consumo energético. Suas características permitem que ele possa ser utilizado em uma grande variedade de projetos para dispositivos móveis e aplicações IoT. Este componente permite também a conexão de mais de um sensor simultaneamente, devendo-se somente adaptar o *firmware* para a administração entre eles [36].

A forma de alimentação utilizada é uma de 3,0 V, não sendo especificada [37].

A Tabela 1 representa, a cada linha, quais pinos foram interconectados do ESP32 com o microfone INMP441.

Tabela 1 - Conexão entre ESP32 e INMP441

ESP32	INMP441
3V3	VDD
GND	GND
IO13	SD
IO2	SCK
IO15	WS

Para o hardware do protótipo, também se faz necessário um aparelho móvel, como um celular. Os únicos requisitos determinados para ele são o sistema operacional android e a conexão Wi-Fi. Neste projeto foi utilizado um Samsung Galaxy A52.

3.3.Projeto de Software

3.3.1. Constituição do projeto de software

Para este projeto, conclui-se que seria necessários dois componentes de software, um para formar o aplicativo mobile e um *firmware*, como é chamado o software que será executado no microcontrolador. Ambos se comunicarão por meio do protocolo MQTT via Wi-Fi, intermediados por um serviço de nuvem que será melhor explicado mais adiante.

3.3.2. Firmware

O INMP441 é conectado com o ESP32, e as conexões definem quais portas serão o WS,

CLK e o SD. Utilizando a Arduino IDE, foi desenvolvido o código para captação e seleção dos dados dessas portas. Recorreu-se a biblioteca I2S para periféricos do ESP32 para que fosse feita a conexão. Os dados obtidos por meio dessa biblioteca foram convertidos para decibel seguindo a Equação 1, equalizados de acordo com o INMP441 e filtrados com a curva de ponderação em A de acordo com o projeto de Kostoski [28]. No código, foi calculada a pressão sonora equivalente com os dados obtidos em um intervalo de 2 segundos, valor determinado pela taxa de envio para o broker, com uma frequência de amostragem de 48 kHz, estabelecida pelo design dos filtros. Esta pressão calculada foi publicada no tópico específico configurado para o local de medição [28, 38].

As especificações do INMP441 que foram utilizadas para o desenvolvimento do firmware se encontram na Tabela 2 [35].

Tabela 2 - Especificações do sensor INMP441

Parâmetro	Valor utilizado
Sensibilidade	-26 dBFS
Referência	94 dB SPL
Máxima Entrada Acústica	120 dB SPL
Ruído de entrada equivalente	33 dBA SPL
Resolução	24 bits

Fonte: INVENSENSE (2014)

3.3.3. Conexão MQTT

Para utilizar o protocolo MQTT, foi necessário selecionar um broker. Foi buscado um sistema que pudesse agir como um broker como também um banco de dados, optando-se pelo serviço de nuvem Adafruit IO [13,15].

A conexão do Adafruit IO com o ESP32 foi feita utilizando a biblioteca PubSubClient e o intervalo para envio de dados foi definido como um envio a cada 2 segundos, de acordo com as especificações MQTT do serviço [39,17].

A conexão do Adafruit IO com o aplicativo mobile foi feita utilizando a biblioteca paho-mqtt e configurações necessárias obtidas na documentação da API MQTT. A reconexão com o broker é feita em intervalos de 1 minuto [40, 17].

3.3.4. Aplicativo Mobile

O aplicativo mobile foi desenvolvido usando a linguagem JavaScript por meio do ecossistema Expo. Esta ferramenta facilita o desenvolvimento de aplicações mobile com React Native, fornecendo bibliotecas que otimizam o desenvolvimento e permitem que o projeto possa ser facilmente visualizado durante o desenvolvimento [41, 42].

O aplicativo mobile permite que o usuário selecione um local já pré-definido e nomeado de acordo com o microfone a qual seu tópico está relacionado. Ao se conectar com o tópico, o usuário começa a receber os dados com atrasos não significativos, conforme são publicados no *feed* do Adafruit IO. Ao receber dados, o aplicativo vai preenchendo um gráfico na interface. Este gráfico disponibiliza a hora que o dado foi recebido, o último valor recebido, e, caso o usuário deseje, o valor de um ponto específico selecionado por ele. A estrutura de tópicos utilizada constitui-se somente do nome do local ao qual o sensor seria inserido, podendo ter sua complexidade alterada, conforme a necessidade, através do código do aplicativo.

Como funcionalidade principal, determinou-se que as notificações deveriam acontecer para valores acima de 45 dBA, como justificado pela literatura, e, de forma arbitrária, determinou-se que o usuário seria notificado quando 3 medidas consecutivas ultrapassassem este limite, e com isso, ruídos de impacto não gerarão notificação, somente ruídos contínuos [9].

A aplicação possibilita também a desconexão do tópico selecionado, fazendo com que pare o recebimento de dados.

Como o protocolo de comunicação MQTT não permite a requisição de arquivos de dados, utilizou-se a API HTTP do Adafruit IO para executar uma requisição HTTP no *feed* do serviço de nuvem e assim disponibilizar para compartilhamento um arquivo (.csv) com os valores de níveis sonoros do último mês e a hora em que eles foram recebidos. Esta funcionalidade é executada quando um botão é pressionado após a seleção do *feed* ao qual se deseja exportar.

4. RESULTADOS

4.1. *Firmware* e *Hardware*

Para validar a integração entre o microcontrolador ESP32 e o sensor INMP441, foram realizados testes envolvendo variações sonoras, como modificações no volume da música reproduzida e até mesmo batidas de palmas nas proximidades do sensor. Essas interferências no ambiente sonoro resultaram em alterações proporcionais nos dados capturados pelo sensor ao longo do processo. Da mesma forma, foram conduzidos testes para verificar a conectividade com o serviço Adafruit IO, utilizando a própria plataforma para visualizar os dados recebidos.

O processamento dos dados do sensor levou em consideração as especificações do INMP441, e os valores obtidos se mostraram realistas e dentro da faixa de valores esperados. Não foram encontrados problemas significativos no processamento dos dados, que mostrou uma grande eficiência no cálculo das intensidades sonoras.

Quanto à conexão entre o ESP32 e o Adafruit IO, ficou evidente que estava funcionando conforme o esperado. O intervalo de tempo entre uma interferência sonora e a representação desses eventos nos dados exibidos no *feed* de dados se manteve consistente. Isso se deve ao fato de que o intervalo de medição dura 2 segundos e, quando ocorria algum evento sonoro durante esse intervalo, ele era refletido nos dados subsequentes. Esse comportamento demonstra a eficácia da integração entre o hardware e o serviço de nuvem, proporcionando uma monitoração precisa dos níveis de ruído no ambiente em tempo real.

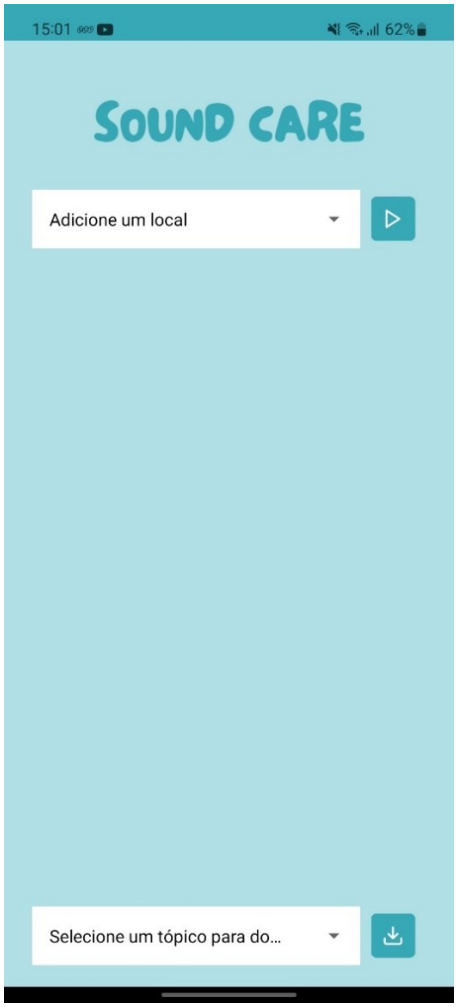
No protótipo obtido, apenas um sensor foi utilizado, mas o ESP32 é capaz de ter mais sensores acoplados a ele.

4.2.Interfaces em *software*

A seguir, das Figuras 5 a 10, estará representado os resultados obtidos no desenvolvimento do aplicativo mobile.

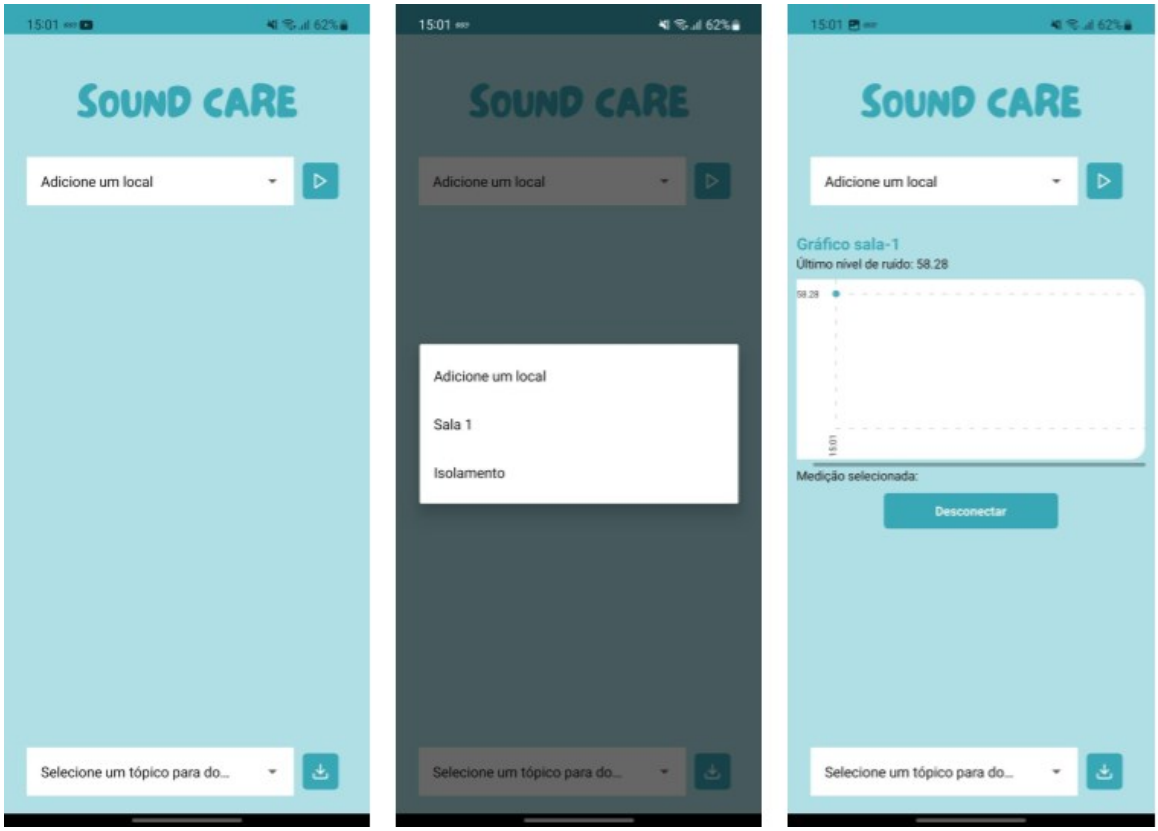
A Figura 4 mostra a tela inicial, onde o usuário pode observar o campo “Adicione um local”, botão com símbolo “*play”* ao lado, o campo “Selecione um tópico para download” e o botão com o símbolo “download” ao lado.

Figura 4 - Tela inicial



A Figura 5 mostra o processo de conexão a um local, ou seja, a um tópico. O usuário deve selecionar o tópico ao clicar em “Adicione um local” que fará com que a lista de locais apareça na tela.

Figura 5 - Processo de conexão com um local



Após selecionado, o usuário deve clicar no símbolo de "*play*" para iniciar a conexão, isto fará com que um gráfico surja na tela, atualizado automaticamente, como mostrado na Figura 6Figura 6. Caso o usuário queira saber o valor de um ponto específico do gráfico, basta clicar nele que o valor será exibido abaixo do gráfico, ao lado do campo "Medição selecionada", evidenciado na mesma figura.

O usuário pode se conectar a mais de um local simultaneamente, devendo somente repetir o processo de conexão, e a tela ficará como na Figura 7, tendo uma barra de rolagem para acomodar os gráficos. Caso o usuário deseje parar de receber dados de algum local específico, basta selecionar "Desconectar" abaixo do gráfico relacionado ao local.

Figura 6 - Telas com o gráfico antes, após recebimento de dados e com ponto selecionado

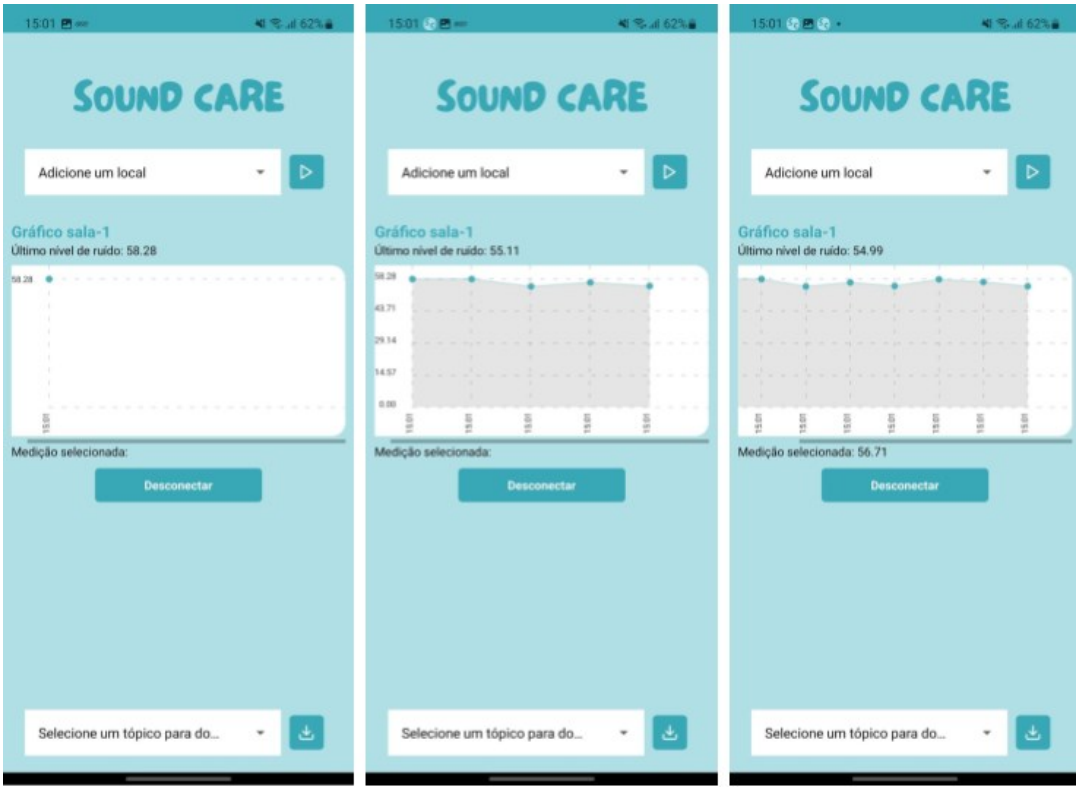
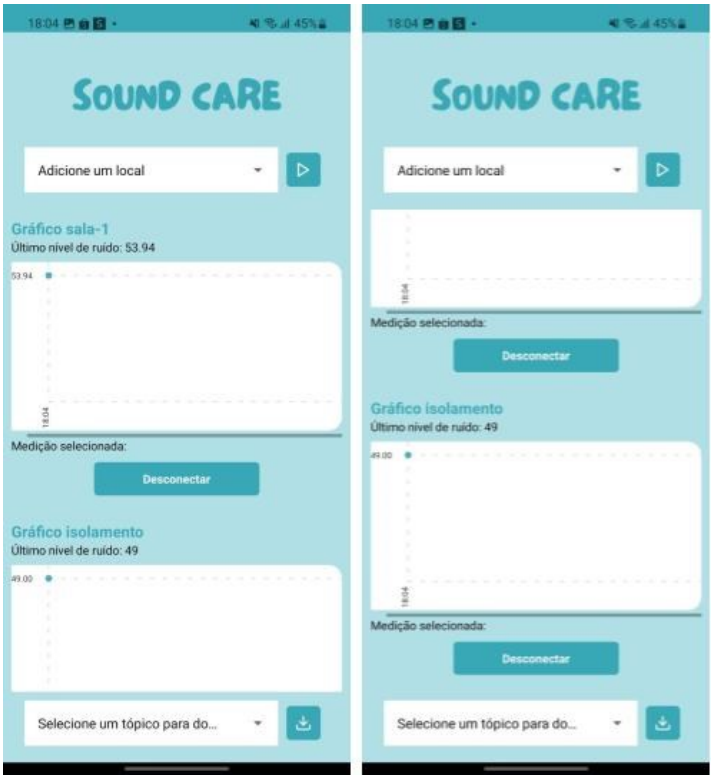


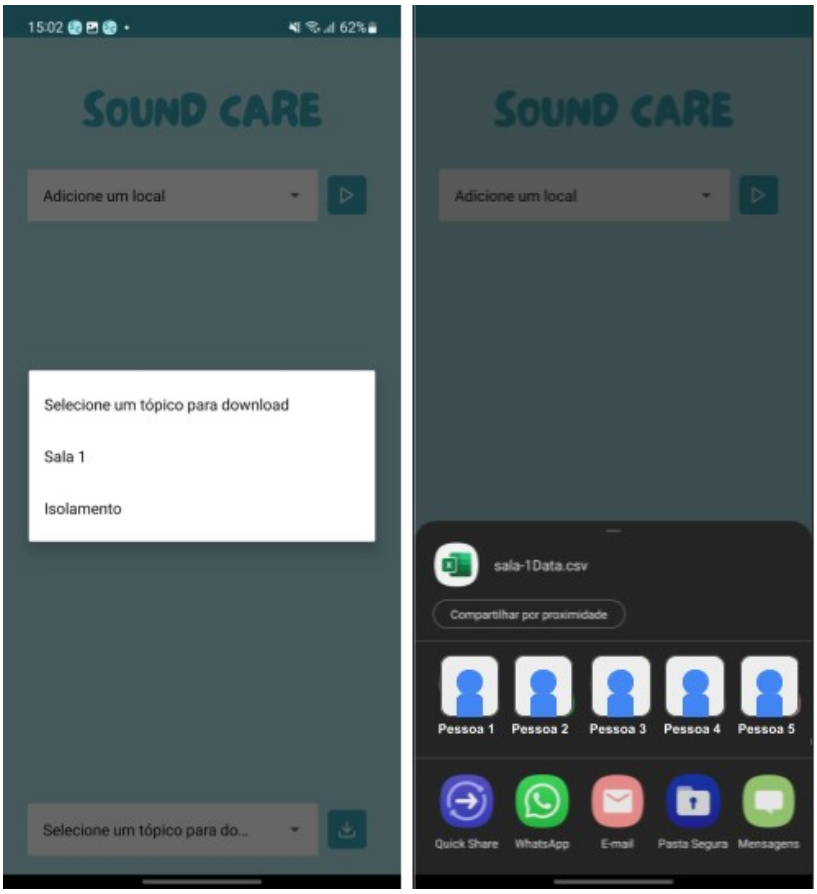
Figura 7 - Telas quando há conexão com mais de um local



Para a funcionalidade de compartilhamento de dados, um local deve ser selecionado na

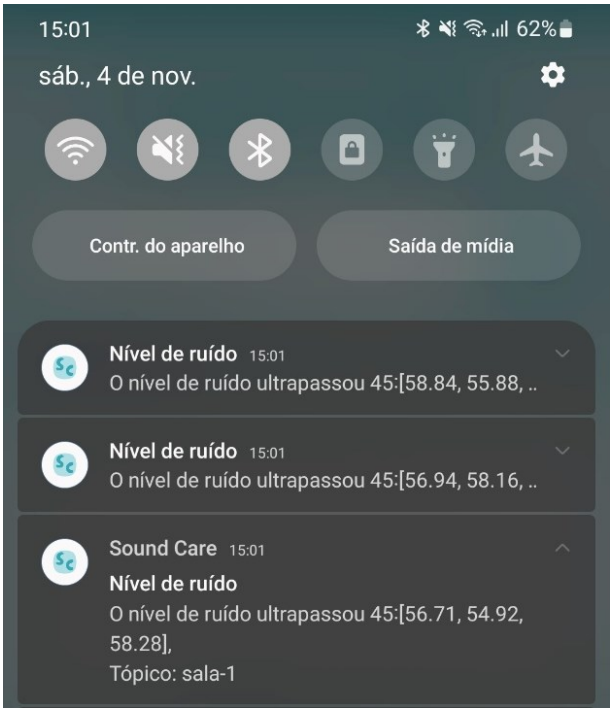
parte inferior da tela, ao selecioná-lo, basta clicar no botão com o símbolo de "*download"* que se abrirá uma aba para compartilhamento do arquivo (.csv). Este processo está exemplificado na Figura 8.

Figura 8 - Telas de compartilhamento de dados



Quando os níveis sonoros ultrapassam os limites, notificações são geradas assim como representado na Figura 10.

Figura 9 - Exemplo de notificação



Os mesmos testes feitos para validação do *firmware* e do *hardware* foram feitos para validação do aplicativo mobile, que também mostrou em seus gráficos, as variações correspondentes com as interferências sonoras geradas, sem atrasos significativos entre interferências sonoras e variações nos gráficos.

As notificações apresentaram comportamento adequado, sendo gerada quando 3 valores consecutivos passavam de 45 dBA, sem considerar a mesma medição mais de uma vez e sem misturar dados de *feeds* diferentes.

Dados foram enviados de forma simultânea entre mais de um *feed* para averiguar se este envio simultâneo iria interferir em outros gráficos e notificações. O aplicativo mostrou o funcionamento adequado.

5. DISCUSSÃO

O protótipo desenvolvido neste trabalho demonstrou um desempenho em conformidade com os objetivos estabelecidos anteriormente. O sensor conectado ao ESP32 foi capaz de captar de forma eficaz os sons do ambiente em que estava inserido. Após a devida manipulação dos dados para convertê-los para a escala dBA, esses dados foram enviados com sucesso para o *feed* correspondente. O processo de envio e recebimento dos dados pelo Adafruit IO ocorreu sem problemas, confirmando a funcionalidade do protocolo MQTT.

A última fase do projeto envolveu o desenvolvimento do aplicativo, que também operou de acordo com o esperado. O aplicativo gerou gráficos da forma esperada, sem interferências de outros *feeds*, e notificou o usuário nos intervalos de tempo apropriados quando três medições consecutivas excederam 45 dBA, demonstrando um comportamento consistente.

No decorrer do projeto, foram identificadas algumas limitações a serem consideradas. Uma delas refere-se ao intervalo de tempo entre os envios de dados para o broker. A magnitude do intervalo interfere na representação da realidade das variações sonoras, levando à dados mais fidedignos quanto maior a taxa de amostragem. Além disso temos o fato de que o projeto não foi calibrado, e, apesar dos cálculos terem sido feitos considerando as especificações dos componentes utilizados, não foi possível obter a exatidão das medidas.

Outra questão importante é a definição dos níveis de ruído de impacto e contínuo que devem acionar notificações. Atualmente, o sistema notifica apenas quando a intensidade sonora ultrapassa 45 dBA em três medições consecutivas.

Embora o protótipo tenha sido desenvolvido com foco na implementação em UTINEOs, essa solução apresentou poucas limitações quanto ao ambiente em que pode ser aplicada, restringindo-se apenas a ser possível ter uma conexão Wi-Fi, à forma de alimentação disponível e a qual dispositivo móvel será utilizado.

6. CONCLUSÃO

Os resultados obtidos neste trabalho atestam a eficácia do protótipo desenvolvido para monitorar os níveis de ruído em ambientes diversos, com um foco inicial na implementação em UTINEOs. A integração e o funcionamento do projeto como um todo demonstrou ser confiável, uma vez que variações sonoras no ambiente resultaram em alterações proporcionais nos dados capturados, que foram processados com precisão e enviados com sucesso para o *feed* no serviço Adafruit IO, permitindo aos usuários acompanhar os níveis de ruído em tempo real e receber notificações quando os valores ultrapassavam os limites estabelecidos. Os gráficos gerados pelo aplicativo refletiram de forma assertiva as interferências sonoras, sem atrasos significativos.

As limitações identificadas podem ser resolvidas ou mitigadas em futuros projetos, como o intervalo de tempo entre os envios de dados, que podem ser otimizados para refletir as condições sonoras com maior precisão, por exemplo, intervalos menores refletem de forma mais precisa as condições sonoras do ambiente. A possibilidade de explorar alternativas de broker ou considerar a assinatura de uma versão paga do Adafruit IO pode ser uma via para aumentar a taxa de dados enviados. No entanto, caso uma mudança de broker seja uma opção, também será essencial planejar uma nova estratégia de armazenamento dos dados, garantindo sua segurança e disponibilidade.

A falta de uma referência para a avaliação da precisão das medidas de intensidade sonora pode ser solucionada por meio de uma calibração adequada e possibilitando que potenciais inconsistências nos dados sejam corrigidas ou remediadas.

A definição dos níveis de ruído que acionam notificações também pode ser refinada para uma classificação mais precisa por meio de pesquisas e testes *in loco*, e, consequentemente, um funcionamento mais eficaz para o combate a excesso de ruídos.

A fonte de alimentação do ESP32 também deve ser considerada, se faz necessária a avaliação da opção mais adequada, levando em consideração o ambiente específico no qual o protótipo será implantado.

Uma sugestão adicional para aperfeiçoar o aplicativo móvel é investigar maneiras de expandi-lo para uma ampla variedade de plataformas, tornando-o compatível com diferentes dispositivos móveis. Isso permitirá que um número ainda maior de usuários aproveite os benefícios desse aplicativo em diversas configurações e cenários. Essas melhorias contribuirão significativamente para a eficiência e versatilidade do protótipo, tornando-o ainda mais valioso em suas aplicações potenciais.

Em resumo, o protótipo apresenta um grande potencial de aplicação em diversos cenários, podendo ser aplicado além das UTINEOs, como em ambientes industriais, residenciais e de segurança no trabalho. Com ajustes e melhorias contínuas, essa solução tem o poder de contribuir significativamente para o monitoramento e gestão dos níveis de ruído, promovendo ambientes mais seguros e saudáveis em benefício de recém-nascidos e usuários.

7. REFERÊNCIAS

1. KRUEGER, Charlene. ***Exposure to maternal voice in preterm infants: A review***. *Advances in Neonatal Care*, v. 10, n. 1, 2010. DOI: 10.1097/ANC.0b013e3181cc3c69. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2883123/. Acesso em: 29 out. 2023

2. CARDOSO, Sandra Maria Schefer; KOZLOWSKI, Lorena de Cássia; LACERDA, Adriana Bender Moreira De; MARQUES, Jair Mendes; RIBAS, Angela. ***Newborn physiological responses to noise in the neonatal unit***. Braz. j. otorhinolaryngol. (Impr.), *[S. l.]*, v. 81, n. 6, 2015.

3. PINEDA, Roberta; DURANT, Polly; MATHUR, Amit; INDER, Terrie; WALLENDORF, Michael; SCHLAGGAR, Bradley L***. Auditory Exposure in the Neonatal Intensive Care Unit: Room Type and Other Predictors. Journal of Pediatrics***, *[S. l.]*, v. 183, 2017. DOI: 10.1016/j.jpeds.2016.12.072.

4. ALMADHOOB, A.; OHLSSON, A. ***Sound reduction management in the neonatal intensive care unit for preterm or very low birth weight infants***. *Cochrane Database of Systematic Reviews*, v. 1, n. 1, 27 jan. 2020. DOI: https://doi.org/10.1002/14651858.CD010333.pub2. Acesso em: 03 nov. 2023.

5. GOLDSON, E. ***Nurturing the premature infant: Developmental interventions in the neonatal intensive care nursery***. New York: Oxford University Press. 1999.

6. BREMMER, P; BYERS, J. F; KIEHL. E. ***Noise and the premature Infant: Physiological Effects and Practice Implications.*** *Principles & Practice*, v.32, n. 4, p 446,454. 2003. DOI: https://doi.org/10.1177/0884217503255009. Disponível em: https://pubmed.ncbi.nlm.nih.gov/12903694/. Acesso em: 3 nov. 2023.

7. PHILBIN, M; KLASS, P. *The full-term and premature newborn: Hearing and behavioral responses to sound in full term newborns.* ***Journal of Perinatology***, 20, S68-S76. 2000. DOI: https://doi.org/10.1038/sj.jp.7200441. Disponível em: https://www.nature.com/articles/7200441. Acesso em: 03 nov. 2023.

8. CATLETT, A; & HOLDITCH-DAVIS, D. ***Environmental stimulation of the acutely ill premature infant: Physiological effects and nursing implications***. *Neonatal Network*, 8(6), 19-25. 1990. Disponível em: https://pubmed.ncbi.nlm.nih.gov/2348812/. Acesso em: 3 nov. 2023.

9. *Committee on Environmental Health. Noise: A hazard for the fetus and newborn.* ***American Academy of Pediatrics***, 1997. DOI: 10.1542/peds.100.4.724. Acesso em: 03 nov. 2023.

10. ABNT - Associação Brasileira de Normas Técnicas. **NBR 10152:** acústica: avaliação do ruído em áreas habitadas, visando o conforto da comunidade: procedimento. Rio de Janeiro: ABNT, 2017. 30 p.

11. AWS. **O que é a Internet das Coisas (IoT)?** Disponível em: https://aws.amazon.com/pt/what-is/iot/. Acesso em: 25 out. 2023.

12. Hive MQ. ***MQTT & MQTT 5 Essentials A comprehensive overview of MQTT facts and features for beginners and experts***. Disponível em: https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf. Acesso em: 25 out. 2023.

13. RUBELL, B. ***Welcome to Adafruit IO*****.** Disponível em: https://learn.adafruit.com/welcome-to-adafruit-io. Acesso em:29 out. 2023.

14. AWS. **O que é uma API (interface de programação de aplicações)?** Disponível em: https://aws.amazon.com/pt/what-is/api/. Acesso em: 02 nov. 2023.

15. COOPER, J. **MQTT API**. Disponível em: https://learn.adafruit.com/adafruit-io/mqtt-api. Acesso em: 29 out. 2023.

16. Adafruit. **Adafruit IO HTTP API**. Disponível em: https://io.adafruit.com/api/docs/#about-the-api-docs. Acesso em: 30 out. 2023.

17. Adafruit. **Adafruit IO MQTT API**. Disponível em: https://io.adafruit.com/api/docs/mqtt.html#adafruit-io-mqtt-api. Acesso em: 29 out. 2023.

18. TREECE, T. ***Adafruit IO Basicas: Feeds*****.** 2015. Disponível em: https://learn.adafruit.com/adafruit-io-basics-feeds/overview. Acesso em: 02 nov. 2023.

19. *All About Circuits*. ***Introduction to the I2S Interface - Technical Articles***. Disponível em: https://www.allaboutcircuits.com/technical-articles/introduction-to-the-i2s-interface/. Acesso em: 28 out. 2023.

20. SOUSA, F. W. J. **Equalização de Sinais de Áudio.** 2004. Centro Federal de Educação Tecnológica de Minas Gerais. Disponível em: https://cursosdeaudiobh.com/wp-content/uploads/2020/04/Equalizacao_de_Sinais_de_audio_-_Pesquisa_de_est%C3%A1gio.pdf. Acesso em: 5 nov. 2023.

21. *Collimator*. ***What is a second order filter?*** 2023. Disponível em: https://www.collimator.ai/reference-guides/what-is-a-second-order-filter. Acesso em: 28 out. 2023.

22. FALCÃO, L. C. L. **O sonômetro e as curvas de ponderação**. 2019. Disponível em: https://www.concepcaoacustica.com/post/o-sonometro-e-as-curvas-de-ponderacao. Acesso em: 28 out. 2023

23. PUJOL, R; TRIGUEIRO-CUNHA, N. **Frequências ouvidas pelo ouvido humano e por alguns outros mamíferos.** 2018. Disponível em: https://www.cochlea.org/po/som/campo-auditivo-humano. Acesso em: 28 out. 2023.

24. LONG, Marshall. ***Architectural acoustics.*** 2. ed. Elsevier, 2014. 943 p.

25. GOGONI, R. **Qual a diferença entre frequência e decibéis?** 2019. Disponível em: https://tecnoblog.net/responde/qual-a-diferenca-entre-frequencia-e-decibeis/. Acesso em: 28 out. 2023.

26. PAVAN, T.Z**. Medidas da Onda Sonora**. Física Acústica. Disponível em: https://edisciplinas.usp.br/pluginfile.php/7579962/mod_resource/content/0/Aula8%20-

%20MedidasDaOndaSonora.pdf. Acesso em: 4 nov. 2023.

27. Magroove. **Decibéis: O que é dB? Entenda o decibel aplicado no áudio profissional**. 2019. Disponível em: https://magroove.com/blog/pt-br/decibeis/. Acesso em: 28 out. 2023.

28. KOSTOSKI, I. **ESP32-I2S-SLM**. Disponível em: https://hackaday.io/project/166867-esp32-i2s-slm. Acesso em: 28 out. 2023.

29. NTi. **Controle de Qualidade de Arrays MEMS Mic digitais**. Disponível em: http://www.nti-audio.com/pt/aplicacoes/controle-de-qualidade/conjuntos-de-mic-mems-digitais. Acesso em: 02 nov. 2023.

30. Vade Mecum Brasil. **Significado de RUÍDO DE IMPACTO**. Disponível em: https://vademecumbrasil.com.br/palavra/ruido-de-impacto. Acesso em: 3 nov. 2023.

31. Vade Mecum Brasil. **Significado de RUÍDO CONTÍNUO**. Disponível em: https://vademecumbrasil.com.br/palavra/ruido-continuo. Acesso em: 3 nov. 2023.

32. COSTA, M. B. **O que é HTTP.** 2021. Disponível em: https://canaltech.com.br/internet/o-que-e-http/. Acesso em: 02 nov. 2023.

33. Mdn Web Docs. **Uma visão geral do HTTP.** Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview. Acesso em: 02 nov. 2023.

34. THAT PROJECT. **ESP32 | INMP441 |** ***Tutorial - [Part.0] Set up I2S for Microphone***. Youtube, 24 fev. 2020. Disponível em: https://www.youtube.com/watch?v=m8LwPNXqK9o&ab_channel=ThatProject. Acesso em: 29 out. 2023.

35. INVENSENSE**. INMP441:** ***Omnidirectional Microphone with Bottom Port and I2S Digital Output***. 2014. Disponível em: https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf. Acesso em: 29 out. 2023.

36. *Espressif Systems.* ***ESP32 Overview***. Disponível em: https://www.espressif.com/en/products/socs/esp32. Acesso em: 29 out. 2023.

37. *Espressif Systems.* ***ESP-WROOM-32 Datasheet*****.** Disponível em: https://pdf1.alldatasheet.com/datasheet-pdf/view/1179101/ESPRESSIF/ESP-WROOM-32.html Acesso em: 4 nov. 2023.

38. *ESP-IDF Programming Guide***. I2S**. Disponível em: https://demo-dijiudu.readthedocs.io/en/latest/api-reference/peripherals/i2s.html. Acesso em: 29 out. 2023.

39. O'LEARY, N. **Arduino Client for MQTT**. 19 maio 2020. Disponível em: https://github.com/knolleary/pubsubclient. Acesso em: 29 out. 2023.

40. NPM. **paho-mqtt**. 2018. Disponível em: https://www.npmjs.com/package/paho-mqtt. Acesso em: 29 out. 2023.

41. Expo**. Expo.** Disponível em: https://expo.dev/. Acesso em: 30 out. 2023.

42. RYAN, Hubert. **O que é Expo?** 2021. Disponível em: https://hubertryanofficial.medium.com/o-que-%C3%A9-expo-714017eb8423#:~:text=O%20Expo%20%C3%A9%20um%20ferramente,um%20projeto%20removendo%20a%20complexidade. Acesso em: 30 out. 2023.