LINCOLN TEODORO DE CARVALHO XAVIER DA SILVA

PROTÓTIPO IOT PARA MONITORAMENTO DE RUÍDOS E ANÁLISE DE CONFORTO SONORO

LAVRAS – MG

2022

LINCOLN TEODORO DE CARVALHO XAVIER DA SILVA

PROTÓTIPO IOT PARA MONITORAMENTO DE RUÍDOS E ANÁLISE DE CONFORTO SONORO

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. DSc. Hermes Pimenta de Moraes Junior
Orientador

Prof. MSc. Douglas Henrique Siqueira Abreu
Coorientador

LAVRAS – MG

2022

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Silva, Lincoln Teodoro de Carvalho Xavier

Protótipo IoT para Monitoramento de Ruídos e análise de conforto sonoro / Lincoln Teodoro de Carvalho Xavier da Silva. – Lavras : UFLA, 2022.

65 p. : il.

Monografia()–Universidade Federal de Lavras, 2022.

Orientador: Prof. DSc. Hermes Pimenta de Moraes Junior.

Bibliografia.

1. TCC. 2. Monografia. 3. Dissertação. 4. Tese. 5. Trabalho Científico – Normas. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

LINCOLN TEODORO DE CARVALHO XAVIER DA SILVA

PROTÓTIPO IOT PARA MONITORAMENTO DE RUÍDOS E ANÁLISE DE CONFORTO SONORO

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Aprovada em 02 de Setembro de 2022.

Prof. MSc. Douglas Henrique Siqueira Abreu	UFLA
Prof. DSc. Hermes Pimenta de Moraes Júnior	UFLA
Prof. DSc. Bruno de Abreu Silva	UFLA
Prof. DSc. Thomaz Chaves de Andrade Oliveira	UFLA

gov.br Documento assinado digitalmente
HERMES PIMENTA DE MORAES JUNIOR
Data: 20/09/2022 19:06:01-0300
Verifique em https://verificador.iti.br

Prof. DSc. Hermes Pimenta de Moraes Junior
Orientador

gov.br Documento assinado digitalmente
DOUGLAS HENRIQUE SIQUEIRA ABREU
Data: 20/09/2022 19:09:54-0300
Verifique em https://verificador.iti.br

Prof. MSc. Douglas Henrique Siqueira Abreu
Co-Orientador

LAVRAS – MG
2022

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus, que sempre tem me protegido, e me dado forças para acordar todos os dias com saúde e disposição para seguir em frente.

Agradecer aos meus pais que mesmo distantes sempre me ajudaram e apoiaram durante esta jornada.

E também agradecer aos meu orientadores Hermes, e Douglas, que me orientaram e ajudaram durante o desenvolvimento deste trabalho.

Agradecer também aos membros da banca pela disponibilidade.

RESUMO

Na sociedade atual estamos sempre buscando formas de solucionar problemas para melhorar a qualidade de vida no nosso cotidiano, e o bem estar pessoal e social é de grande importância tanto para a qualidade quanto para a produtividade. A poluição sonora é um fator importante na nossa produtividade e esta não recebe tanta atenção quanto deveria, muitos ambientes de trabalho não são fiscalizados, ambientes com contínua exposição a barulho ou barulho muito alto são prejudiciais à saúde. É preciso monitorar esses ambientes, para que se tenha uma conscientização da importância de um ambiente livre de poluição sonora, e em casos extremos poderia até mesmo ser aplicado uma penalização. Estes fatores motivaram o desenvolvimento deste trabalho que propõe uma solução através de um protótipo IoT, para realizar as medições da qualidade acústica de um ambiente. A internet das coisas é uma tecnologia que vem sendo muito utilizada atualmente e ela oferece muitas ferramentas para ajudar nesse caso, sendo a principal delas o monitoramento e coleta de dados remoto. Este trabalho tem como objetivo principal propor e desenvolver um protótipo de hardware, utilizando conceitos de IoT, que deverá ser capaz de monitorar ruídos de ambientes específicos por meio de um microfone, analisar esses ruídos, e então classificar esses ambientes tendo como base a normas ABNT NBR 10.151 e 10.152. O hardware desenvolvido serve de base para trabalhos futuros, tais como, reproduzir vários desses protótipos para que seja feito um mapeamento de ruído urbano por exemplo, ou então utilizar esse monitoramento para iniciar ações de conscientização, ou até mesmo utilizá-lo para que emita avisos de ambientes insalubres.

Palavras-chave: Internet das Coisas; Monitoramento de ruídos; Saúde

ABSTRACT

Nowadays we are always searching for to improve the quality of our daily lives, and the well-being plays a big role on this. The sound pollution is an important factor in our productivity and it is not getting the attention it should. Many work environments are not supervised. Environments with continuous exposure to loud noises are harmful to health. It is required to supervise those environments for an increased awareness of a free noise environment's importance. In case of extreme conditions, a penalty could be applied. This factors motivated the development of this work, which propose an IoT solution to measure the quality of an environment regarding the noise monitoring. The Internet of things offers a lot of tools to act in this project both in remote collecting and monitoring. This work's main goal is the development of a hardware prototype using IoT concepts. The hardware should be capable of monitoring noises from an specific environment, analyse, and classify whether the environments follow the ABNT 10.151 and 10.152 standards. The hardware developed suits as a base for future projects, like developing copies of this hardware for mapping the noise from a area of a big company, or use this monitoring to start consenting actions, or it could be used to notify an unhealthy environment.

Keywords: IoT; Noise monitoring; Health

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Na sociedade atual são sempre buscadas novas formas de solucionar problemas para melhorar a qualidade de vida do cotidiano das pessoas. Nessa busca, o bem estar pessoal e social é de grande importância tanto para a qualidade quanto para a produtividade. A poluição sonora é um fator importante na produtividade das pessoas, e esta não recebe tanta atenção quanto deveria. A poluição sonora é uma das principais fontes de contaminação de um ambiente urbano (ALAM et al., 2020), e muitos ambientes de trabalho não são fiscalizados. Ambientes com contínua exposição a barulho ou barulho muito alto são prejudiciais à saúde. A norma regulamentadora 15 (Ministério do Trabalho e Previdência, 2022) detalha a quantidade máxima de exposição para os níveis de ruído acima de 85 decibéis. É preciso monitorar esses ambientes para que se tenha uma conscientização da importância de uma ambiente livre de poluição sonora, e em casos extremos poderia até mesmo ser aplicada uma penalização. Segundo Alam et al. (2020), essa poluição só tende a aumentar no meio urbano, devido ao crescimento na quantidade de veículos, e atividades que são realizadas nesses ambientes. Esses fatores motivaram o desenvolvimento deste trabalho que propõe uma solução através de um protótipo IoT, para realizar as medições da qualidade sonora de ambientes externos e internos. Esse dispositivo IoT é capaz de monitorar os níveis de ruído em um ambiente remotamente através da conexão com um banco de dados via wifi. O dispositivo também alerta fisicamente através de LEDs quando os níveis de ruído estão dentro dos limites ou fora deles. A internet das coisas é uma tecnologia que vem sendo muito utilizada atualmente e ela oferece muitas ferramentas para ajudar neste projeto, sendo a principal delas o monitoramento e coleta de dados remotos.

1.1 Objetivos

Este trabalho tem como objetivo principal propor e desenvolver um protótipo de hardware, utilizando conceitos de IoT, que deverá ser capaz de monitorar ruídos de ambientes específicos, analisar esses ruídos, e então classificar esses ambientes com base nas normas ABNT NBR 10.151 e 10.152.

1.1.1 Objetivos Específicos

- Desenvolver um hardware para o monitoramento de ruídos;

- Implementar um software para realizar as conversões de entrada analógica;
- Exportar os dados em tempo real para um banco de dados;
- Analisar os dados e classificar a qualidade sonora de um ambiente.

1.2 Justificativa

Com o avanço da tecnologia, cada vez mais soluções vêm sendo desenvolvidas para melhorar a qualidade de vida e produtividade das pessoas, principalmente dos trabalhadores. No entanto, o conforto acústico é um problema que não vem recebendo tanta atenção ultimamente.

Segundo Bistafa (2018), a exposição a níveis elevados de ruído podem ser prejudiciais à saúde, causando perda de audição, queda de desempenho, stress, pertubação no sono, entre outros. Para Ganime et al. (2010), ruído é um som indesejável e que gera incômodo, podendo causar doenças ou até mesmo atrapalhar no processo de comunicação. Na Europa é obrigatório desde 2002 o mapeamento e monitoramento de ruídos para cidade com mais de 250 mil habitantes (EUROPEU; EUROPEIA, 2002). Já no Brasil, apenas algumas cidades fazem essa regulamentação.

Todos esses motivos contribuíram para o desenvolvimento deste trabalho, que tem como primeiro objetivo implementar uma solução IoT para o monitoramento remoto de ruídos, e que pode futuramente ser ampliado para realizar mapeamentos de áreas estratégicas tais como escritórios, cidades, bibliotecas, ambientes onde a contínua exposição a ruídos possa ser prejudicial às pessoas ali presentes. Com esse monitoramento remoto, é possível saber em que área o nível de ruído está elevado e assim poder tomar as medidas necessárias para resolver o problema.

1.3 Metodologia

Para a realização deste trabalho foi escolhido um tipo de pesquisa exploratório, onde foram pesquisados trabalhos que abordavam o tema monitoramento de ruídos para analisar qual a melhor forma de se coletar pressão sonora de um ambiente, e também analisar quais os equipamentos utilizados e métodos de coleta utilizados.

Foram realizadas pesquisas para analisar opções de hardware de baixo custo disponível no mercado para desenvolvimentos de projetos IoT. Após essa pesquisa, é preciso decidir qual a plataforma de desenvolvimento se adéqua melhor ao projeto. Também foram feitas pesquisas

dos tipos de microfone adequado para a captura de ruídos. Com microfone e placa decididos foram pesquisados quais os componentes necessários para a montagem do circuito juntamente com mais três LEDs, para o alertar no local onde o dispositivo esta instalado os níveis de ruído. Com o hardware completo, iniciou-se então a pesquisa dos softwares necessários para a implementação do código. Foi pesquisado qual interface de desenvolvimento tem suporte para a placa selecionada, e qual linguagem de programação suporta. Por último foi pesquisado quais as melhores alternativas gratuitas para a criação de um banco de dados.

Foi utilizada uma pesquisa experimental para validar os dados obtidos ao final dos testes realizados. Essa validação foi feita através da comparação de precisão dos resultados do microfone escolhido com um outro microfone profissional. Para realizar o monitoramento e coleta de dados foi utilizado um protótipo de IoT que é capaz de enviar em tempo real informações para um servidor, os testes podem ser reproduzidos em qualquer área onde há acesso a internet. Com os resultados dos testes foi feita uma análise dos dados obtidos.

1.4 Estrutura do texto

O trabalho está dividido em cinco capítulos, sendo o primeiro esta introdução; no segundo capítulo é abordada uma contextualização sobre como é feita a captura e processamento de ruídos, assim como as normas ABNT NBR 10.151 e 10.152 utilizadas para a calibração do equipamento e classificação dos ruídos respectivamente.

No terceiro capítulo é feita uma contextualização sobre IoT e sua importância para este trabalho.

No quarto capítulo é explicada a construção do protótipo IoT utilizado neste trabalho, todo seu hardware e componentes, os softwares e linguagens utilizadas. Por final, os testes realizados e seus resultados são apresentados.

No quinto e último capítulo, conclusão e possíveis trabalhos futuros são abordados.

2 MONITORAMENTO DE RUÍDOS

Este capítulo trata de como foi feita a captura e processamento de ruídos, as fórmulas utilizadas para a conversão em dB SPL (do inglês, *Decibel Sound Pressure Level*), e também das normas ANBT NBR 10.151 e 10.152.

2.1 Introdução ao monitoramento de ruídos

De acordo com Alías e Alsina-Pagès (2019), antes da IoT a análise da acústica de um ambiente era realizada por profissionais, que gravavam e realizavam a análise de um determinado ambiente. O problema é que este método não consegue acompanhar a demanda da analise de ruídos, mas com o avanço da tecnologia e da IoT, hoje já é possível mapear automaticamente ambientes de uma cidade através de uma rede de sensores acústicos sem fio, também conhecido como WASN (do inglês, *Wireless Accoustic Sensors Netwok*)(ALÍAS; ALSINA-PAGÈS, 2019). WASN é uma rede que utiliza sensores acústico de baixo custo interligados por uma rede, para realizar um mapeamento automático de ruídos urbanos. Os sensores coletam o nível de equivalência dos ruídos e enviam para um servidor nuvem, o servidor então gera o mapeamento de ruídos de uma área urbana (ALÍAS; ALSINA-PAGÈS, 2019).

Quando se trata de monitoramento ruídos é necessário um ou vários sensores para realizar a coleta dos dados. Segundo Alías e Alsina-Pagès (2019) os tipos de sensores podem ser classificados de acordo com sua capacidade computacional e acurácia:

- **Sensores de alta precisão:** Sensores de alto custo e alta precisão, muito utilizados no estudo detalhado da acústica de um ambiente. São sensores que utilizam microfones IEC classe 1. A IEC(do inglês, *Internal Eletrotechinical Comission*) classifica os instrumentos de medição acústica em duas classes. Instrumentos de classe 1, são instrumentos de alta precisão destinados para uso em laboratório, já instrumentos de classe 2 são para uso geral e de menor performance (NARANG; BELL, 2008). Devido ao seu alto custo, instrumentos de classe 1 geralmente são inviáveis em projetos que necessitam muitos sensores.

- **Sensores de baixo custo e alto processamento:** São sensores que tem um preço agradável mas que não deixa de ter uma boa precisão, são recomendáveis em projetos onde

é necessário uma grande quantidade de sensores com boa precisão mas não é possível gastar com sensores de alta precisão.

- **Sensores de baixa custo e processamento:** Sensores mais baratos e com baixa precisão, são utilizados em projetos onde apenas é de interesse o valor equivalente do ruído.

De acordo com Saliba (2021) os principais medidores de ruídos são os dosímetros, calibradores acústicos, medidores de pressão sonoras e analisadores de frequências. Os detalhes do medidor acústico em dBSPL será explicada nas próximas sessões, já as restantes podem ser descritas como:

- **Analisadores de Frequências**: São medidores que analisam o espectro de um som. Esse espectro pode ser analisado através da pressão sonora em decibéis em função da sua frequência.
- **Calibrador acústico:** São medidores utilizados para a calibração de outros instrumentos, a calibração deve ser feita antes de qualquer medição. Para a calibragem deve ser reproduzido um tom de 1kHz para um valor medido de 94dB.
- **Dosímetros:** São instrumentos que caracterizam a exposição de um ambiente à ruídos. Estes instrumentos são capazes de medir a dose de um ruído, assim como os níveis equivalente de ruído.

2.2 Coleta de ruídos

Neste trabalho foi escolhido o microfone MAX 9814 para realizar a coleta de pressão sonora, o microfone está de acordo com as exigências definidas pela norma ABNT NBR 10151. Os detalhes e especificações técnicas deste componente podem ser encontrados no Capítulo 4 deste trabalho. Esse microfone foi escolhido por se tratar de um microfone que possui uma sensibilidade maior, sendo indicado para projetos acústicos. O seu tamanho e baixo custo também foram fatores para a sua escolha.

O MAX 9814 faz uma coleta de uma pressão sonora em sinal analógico. Graças ao seu ADC(*Analogic Digital Converter*) a conversão para sinal digital é feita na própria coleta. Esse valor coletado se trata de um valor de pressão sonora dos níveis equivalentes de tensão do sinal, assumindo valores de 12 bits, entre 0 e 4095. É preciso converter esse valor para pressão sonora

em decibéis. Essa conversão é feita primeiramente pelo calculo do valor eficaz RMS (do inglês, *Root Mean Square*) de um conjunto de amostras durante um segundo e em seguida é feito o calculo da conversão para dB SPL.

2.2.1 Valor Eficaz RMS

O valor eficaz RMS (do inglês, *Root Mean Square*) é uma fórmula que realiza a raiz quadrada média de um sinal de voltagem analógico. Essa fórmula é importante, pois os sinais lidos são analógicos, o que faz seus valores sofrerem uma variação no decorrer do tempo, diferente de um sinal digital que tem uma variação mais discreta. Essa fórmula consegue fazer esta conversão de um sinal AC para um DC equivalente, a fórmula do cálculo RMS é dado por:

$$Vrms = \sqrt{\frac{\sum sinal^2}{amostras}} \tag{2.1}$$

Onde é feito pela raiz quadrada de um somatório dos sinais medidos elevados a potência de dois, dividido pelo número sinais coletados. Agora o conjunto de valores analógicos foram convertidos para um único valor digital equivalente de amplitude em função do tempo.

2.2.2 Amplitude para dB

Após o calculo RMS é preciso fazer a conversão do valor de amplitude para decibéis. De acordo com Alonso (2016), a percepção do ouvido humano funciona de forma logarítmica, devido a isso foi selecionado uma grandeza logarítmica, o decibel, para realizar a medição da pressão sonora. O nível de pressão sonora em decibéis é definido pela seguinte fórmula:

$$dBSPL = 20 \cdot log\left(\frac{p}{pRef}\right)^2 \tag{2.2}$$

Onde: p: É o sinal de pressão sonora convertido pelo RMS.

$pRef$: É o valor de pressão de referência, em $20\mu Pa$ $(20x10-6N/m^2)$, calculado a um tom em uma frequência de 1Khz.

O valor de pressão de referência deve ser calculado com o cálculo RMS do sinal lido com um tom de 1kHz a 94dB, essa medição em 94dB deve ser realizada junto de um microfone calibrador auxiliar.

2.3 Normas ABNT NBR 10151 e 10152

Na Associação Brasileira de Normas Técnicas existem duas normas que tratam da regulamentação de medições de pressão sonora: NBR 10151 e NBR 10152. A norma NBR 10151 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2019) trata da medição de pressão sonora feitas em áreas habitadas, ela estabelece limites de ruídos aceitáveis assim como devem ser feitos a medição e calibração de equipamentos. Segundo a norma, a medição deve ser feita a pelo menos 0,5 m de paredes, teto e piso, e de pelo menos 1 m de distância de fontes de som significativa, como janelas e portas. A norma também exige que o microfone utilizado na medição deve atender às especificações da IEC 61672-1 ou da IEC 61094-4. Também especifica os horários que devem ser realizadas as medições, o horário diurno não há um limite definido, mas o horário noturno não deve começar depois das 22h e não deve terminar antes das 7h do dia seguinte. Os limites máximos de nível de pressão sonora em decibéis por locais e horários são definidos no Quadro 2.1.

Quadro 2.1 – Limites de níveis de pressão sonora em função dos tipos de áreas habitadas e do período

Tipos de áreas habitadas	RL_{Aeq} Limites de níveis de pressão sonora (dB)	
	Período diurno	Período noturno
Área de residências rurais	40	35
Área estritamente residencial urbana ou de hospitais ou de escolas	50	45
Área mista predominantemente residencial	55	50
Área mista com predominância de atividades comerciais e/ou administrativa	60	55
Área mista com predominância de atividades culturais, lazer e turismo	65	55
Área predominantemente industrial	70	60

Fonte: Norma NBR 10151

Já a norma NBR 10152 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 1987) apresenta os procedimentos para medição em ambientes internos de edificações. As especificações de microfone e posições de medições sãos as mesmas da NBR 10151, já a sua tabela de limites de níveis de pressão sonora é bem diferente, como pode ser visto nos quadros 2.2, 2.3 e 2.4:

Quadro 2.2 – Valores de referência para ambientes internos de uma edificação de acordo com suas finalidades de uso (continua)

Finalidade de uso	Valores de referência		
	RL_{Aeq} (dB)	RL_{ASmax} (dB)	RL_{NC}
Aeroportos, estações rodoviárias e ferroviárias			
Áreas de *check-in*, bilheterias	45	50	40
Salas de embarque e circulações	50	55	45
Centros comerciais (*shopping centers*)			
Circulações	50	55	45
Lojas	45	50	40
Praças de alimentação	50	55	45
Garagens	55	60	50
Clínicas e hospitais			
Berçários	35	40	30
Centros cirúrgicos	35	40	30
Consultórios	35	40	30

Fonte: Norma NBR 10152

Quadro 2.3 – Valores de referência para ambientes internos de uma edificação de acordo com suas finalidades de uso (continua)

Finalidade de uso	Valores de referência		
	RL_{Aeq} (dB)	RL_{ASmax} (dB)	RL_{NC}
Enfermarias	40	45	35
Laboratórios	45	50	40
Quartos coletivos	40	45	35
Quartos individuais	35	40	30
Salas de espera	45	50	40
Culturais e lazer			
Salões de festa	40	45	35
Restaurantes	45	50	40
Cinemas	35	40	30
Salas de concertos	30	35	25
Teatros	30	35	25
Templos religiosos pequenos (≤ 600 m^3)	40	45	35
Templos religiosos grandes (> 600 m^3)	35	40	30
Bibliotecas	40	45	35
Museus (exposições)	40	45	35
Estúdios de gravação audiovisual	25	30	20
Educacionais			
Circulações	50	55	45
Berçário	40	45	35
Salas de aula	35	40	30
Salas de música	35	40	30
Escritórios			
Centrais de telefonia (*call centers*)	50	55	45
Circulações	50	55	45
Escritórios privativos (gerência, diretoria etc.)	40	45	35
Escritórios coletivos (*open plan*)	45	50	40
Recepções	45	50	40
Salas de espera	45	50	40
Salas de reunião	35	40	30

Fonte: Norma NBR 10152

Quadro 2.4 – Valores de referência para ambientes internos de uma edificação de acordo com suas finalidades de uso (conclusão)

Finalidade de uso	Valores de referência		
	RL_{Aeq} (dB)	RL_{ASmax} (dB)	RL_{NC}
Salas de videoconferência	40	45	35
Esportes			
Ginásios de esportes e academias de ginástica	45	50	40
Hotéis			
Quartos individuais ou suítes	40	45	35
Salões de convenções	40	45	35
Áreas de serviço	50	55	45
Circulações	45	50	40
Residências			
Dormitórios	35	40	30
Salas de estar	40	45	35
Salas de cinema em casa (*home theaters*)	40	45	35
Outros			
Auditórios grandes (> 600 m^3)	30	35	25
Auditórios pequenos (≤ 600 m^3)	35	40	30
Cozinhas e lavanderias	50	55	45
Tribunais	40	45	35
NOTA O valor de RL_{Aeq} para dormitório é compatível a faixa estabelecida na ABNT NBR 10152:1987 e também para a condição de L_{Aeq} de até 65 dB em áreas externas urbanas para o período diurno e 55 dB para o período noturno, estabelecida na ABNT NBR 10151:2000, considerado o desempenho mínimo previsto pela ABNT NBR 15575-4:2013 de 25 dB para isolação de fachada em regiões Classe II (ver [2], [3] e [4]).			

Fonte: Norma NBR 10152

Neste trabalho, é de interesse os valores de referência em área mista com predominância de atividades culturais e de lazer, já que serão realizados medições de testes em um ambiente de convivência de uma universidade. Segundo a norma NBR 10151, o nível em dB de ruído nesse ambiente não deve ser superior a 65dB no período diurno, e 55dB no período noturno.

3 INTERNET DAS COISAS

Neste capítulo, é apresentada a contextualização da internet das coisas, o que é, seus objetivos, suas aplicações, dispositivos e protocolos de comunicação.

3.1 O que é IoT

A IoT, (do inglês, *Internet Of Things*) é um termo que foi citado em 1999 por Kevin Ashton (ASHTON, 1999). Para o autor, a internet das coisas surge para suprir a necessidade da coleta e processamento de dados, segundo ele a capacidade humana de acompanhar a demanda de dados não é suficiente, se os computadores soubessem como coletar esses dados e o que fazer com eles, os problemas da tecnologia moderna estariam resolvidos.

Segundo Galegale et al. (2016), a internet das coisas é uma continuação da internet em que objetos com capacidade computacional e de comunicação, são conectados à internet.

Já para Atzori, Iera e Morabito (2010) o conceito de IoT se trata da interação de uma variedade de objetos entre si, através de uma rede para atingir um objetivo em comum. Ele acredita que o maior potencial do IoT está em ajudar nas atividades do dia a dia.

Para Leite, Martins e Ursini (2017) essa tecnologia visa utilizar a interligação entre coisas possibilitando uma automatização do nosso cotidiano, e também possibilita uma aumento da inteligência nas áreas de saúde, logística, indústria entre outros. Segundo os autores, a IoT está realizando uma nova transformação digital, criando um mundo onde pessoas, dispositivos e ambientes estão todos conectados e interagindo entre si.

3.2 Dispositivos IoT

Nesta seção, serão abordadas a arquitetura dos dispositivos IoT disponíveis atualmente, suas características e suas limitações.

3.2.1 Arquitetura

Segundo Cheruvu et al. (2019), um dispositivo na perspectiva da manufatura pode ser definido como um componente físico que possui um hardware, software e *firmware*. Já na perspectiva de aplicação IoT, um dispositivo é um contexto lógico que utiliza interfaces para

a troca de informações estruturadas de acordo com uma interface definida (CHERUVU et al., 2019).

A arquitetura de um dispositivo IoT deve ser capaz de atuar em condições hostis, como temperaturas extremas, vibração, umidade e radiação ultravioleta (CHERUVU et al., 2019). De acordo com Santos (2018), a construção de uma arquitetura IoT pode ser divida em quatro blocos:

- "**Coisa**": O que será coletado ou monitorado.
- **Módulo de aquisição de dados**: Módulo responsável pela aquisição de sinais físicos e conversão desses em sinais digitais.
- **Módulo de processamento de dados**: Módulo que realiza todo o processamento e operações computacionais na borda.
- **Módulo de comunicações**: Módulo que permite a comunicação do dispositivo com sistemas de terceiros locais ou na nuvem.

3.2.2 Módulos IoT

De acordo com Oliveira (2021) microcontroladores são dispositivos capazes de realizar uma tarefa que precisa ser feita manualmente. São dispositivos que antigamente possuíam apenas interfaces de entrada e saída e eram utilizados na indústria. Com o avanço da tecnologia cada vez mais os microcontroladores foram se aprimorando, hoje eles já possuem memória RAM e EPROM, cristal oscilador. Os módulos mais recentes já são capazes de até possuir interfaces de comunicação, como wifi, e bluetooth. Essas novas tecnologias expandiram as aplicações que os microcontroladores são utilizados, como na IoT por exemplo. Arduino, Raspberry Pi e ESP são algumas das placas mais utilizadas, e suas principais características serão explicadas nas subseções a seguir.

3.2.2.1 ESP

Criado pela empresa chinesa ESPRESSIF[1], e conhecido por suas placas ESP8266 e ESP32, são placas de baixo consumo e custo, e com conectividade wifi. Além de ter um

[1] <https://www.espressif.com>

hardware superior a sua versão anterior, o ESP32 também possui conectividade bluetooth BLE (do inglês, *Bluetooth Low Energy*). Placas ESP podem ser programadas em ambientes de desenvolvimento do Arduino utilizando linguagem C ou C++, ambientes de desnvolvimento Lua ou então em um ambiente de desenvolvimento RTOS(do inglês, *Real Time Operating System*). Placas ESP também são compatíveis com os periféricos Arduino, o que junto com suas conectividades wifi e bluetooth o faz uma ótima escolha para projetos IoT.

3.2.2.2 Arduino

Arduino é uma plataforma de desenvolvimento de sistemas embarcados livre para dispositivos de baixo custo (OLIVEIRA, 2021), seus módulos mais conhecidos são: Arduino uno; Arduino mega; e nano. Diferente do ESP suas placas não possuem módulo wifi, sendo preciso pagar um maior valor para uma versão da placa com wifi. As placas Arduino podem ser desenvolvidas no próprio ambiente de desenvolvimento Arduino. C e C++ são as linguagens suportadas pelas placas. Recomendado para projetos iniciantes e aplicações onde conectividade com internet não é necessário.

3.2.2.3 Raspberry Pi

Assim como o Arduino, também é uma plataforma de desenvolvimento livre. Seu grande diferencial das placas Arduino e ESP é o seu hardware robusto, tanto em capacidade de processamento quanto a de armazenamento. As placas Raspberry Pi também possuem várias conectividades, tendo suporte a wifi, bluetooth BLE, e ethernet. A plataforma pode ser desenvolvida em várias linguagens de programação, sendo algumas delas, Java, C++, Python, Fortran entre outras. Devido ao seu potente hardware, o seu custo também é maior em comparação as placas Arduino e ESP, sendo o Raspberry Pi recomendado para projetos em que esse poder de hardware seja realmente necessário.

3.2.3 Sensores e Atuadores

Os sensores e atuadores estão entre os modos de construção de um dispositivo IoT (MOUHA, 2021). Os sensores são dispositivos responsáveis pela coleta de dados que serão

enviados para uma rede, e para os atuadores que irão realizar alguma ação ou não de acordo com o dado coletado. Microfones, sensores de luz ambiente, acelerômetro e giroscópio, são exemplos de sensores.

Atuadores são dispositivos mecânicos ou eletromecânicos que controlam movimentos de objetos (MOUHA, 2021). Os atuadores podem ser elétricos, mecânicos, hidráulicos e pneumáticos. O atuador pneumático converte a energia formada pelo ar comprimido em um movimento linear ou rotacional. O atuador hidráulico utiliza a força da hidráulica para realizar movimentos lineares, oscilatórios ou rotacionais. Eletromecânicos convertem energia elétrica em torque mecânico. E um atuador elétrico oferece um torque de diversas formas.

3.3 Aplicações da IoT

Como já dito anteriormente, a IoT vem crescendo muito e vem sendo utilizada em várias áreas distintas, nesta seção serão explicadas as principais aplicações da IoT.

3.3.1 Saúde

De acordo com Forbes (2018), a tecnologia foi muito importante para a medicina moderna, e a IoT vem acelerando essa evolução ainda mais. Cada vez mais vidas vem sendo salvas graças ao uso da internet das coisas próxima ou até mesmo dentro de pacientes. Um exemplo de como a IoT vem mudando a medicina moderna é a substituição de aparelhos raio-X por aparelhos de IoT, onde o aparelho ao invés de imprimir uma radiografia, a imagem é enviada para a internet onde pode ser vista pelo médico. Há pesquisas sendo feitas onde médicos podem receber assistência de uma IA, que pode prever o diagnóstico com alta precisão (KUMAR et al., 2018).

A IoT támbem vem sendo muito utilizada para monitoramento na área da saúde, como o monitoramento de paciente. Dispositivos IoT estão sempre verificando a condição do paciente, e caso algo fora do comum aconteça, a equipe profissional responsável é notificada. Monitoramento dos próprios profissionais também está sendo feito, para que se conheça seus níveis de estresse de forma a evitar que as decisões do profissional sejam influenciadas durante esses momentos.

Já existem também dispositivos IoT que monitoram um paciente internamente: são sensores ingeríveis que conseguem coletar informações como hemorragia interna ou nível de pH no estômago. Ao final do uso, esses dispositivos são dissolvidos pelo próprio organismo.

3.3.2 Cidades Inteligentes

Segundo Halegoua (2020), cidade inteligente ou *smart city* é uma cidade onde tecnologias da informação e comunicação são fortemente implementadas para coletar dados com o objetivo de ajudar no monitoramento e aprimoramento do meio urbano. São tecnologias que podem ajudar na economia do consumo de energia, transporte e serviços de coleta de lixo por exemplo. Há várias aplicações já sendo utilizadas atualmente. Uma delas é a utilização de *smart grids* para o fornecimento de energia inteligente e sustentável. Em Amsterdam já estão oferecendo painéis solares para casas ligadas a um *smart grid*, assim é possível que essas casas acumulem energia durante o dia e a redistribua durante a noite (MILCHRAM et al., 2020).

Também já estão sendo utilizadas em algumas cidades as lixeiras inteligentes, que constantemente monitoram a capacidade da lixeira, e avisam quando estão cheias. Existem até lixeiras inteligentes que conseguem classificar o tipo de lixo para a coleta reciclável (HUH; CHOI; SEO, 2021).

Em algumas cidades já estão sendo utilizados sensores com inteligência artificial integrados a semáforos. Os sensores conseguem identificar o fluxo de veículos e ajustar o tempo do semáforo de acordo com as informações coletadas até então (BLOKDYK, 2019).

3.3.3 Agricultura

A internet das coisas também está presente na agricultura: sensores de coleta de dados permitem o monitoramento e análises em tempo real. O sensor é capaz de identificar fatores que podem atrapalhar na qualidade da safra, como condições climáticas, e qualidade do solo.

Outro uso da IoT na agricultura é a instalação de sensores que conseguem identificar o melhor momento e quantidade de água que deve ser utilizada para irrigar uma plantação. O sensor é capaz de notificar uma central ou até mesmo realizar a irrigação por conta própria.

Estufas inteligentes é outro exemplo onde vários dispositivos IoT conseguem controlar um ambiente. Antes, o controle de uma estufa demandava uma atividade humana e manual.

Hoje, isso pode ser realizado de forma automática, através de vários sensores e aparelhos interligados. Um sensor de calor consegue enviar informações para uma lâmpada caso a temperatura esteja abaixo do desejado ou então se comunica com o sistema de ventilação para refrescar o ambiente caso esteja muito quente.

3.4 Protocolos de Comunicação

Nesta seção será explicado quais são os principais protocolos de comunicação que dispositivos IoT utilizam para o envio de dados.

3.4.1 Zigbee

De acordo com Moraes e Takashi (2021), Zigbee é um padrão aberto de comunicação entre dispositivos através de uma rede sem fio. Os dispositivos se conectam através de um roteador que comunica via rádio *Zigbee* por um lado, e pelo outro conecta os dispositivos à internet. Uma rede Zigbee é capaz de suportar até 65 mil dispositivos em sua rede (MORAES; TAKASHI, 2021).

As grandes vantagens do protocolo Zigbee são o seu preço acessível, baixo consumo de energia, e fácil instalação. O protocolo Zigbee trabalha nas faixas de rádio a uma frequência de 2.4Ghz, e consegue enviar dados com uma taxa de transmissão de 250Kbits/s. Devido a sua baixa velocidade de transmissão de dados, este protocolo é muito utilizado e recomendado em projetos de casas inteligentes.

O protocolo Zigbee já pode ser encontrado em vários dispositivos IoT, como por exemplo dispositivos da linha *SmartThings* da Samsung em geladeiras inteligentes. Lâmpadas inteligentes produzidas pela Philips. E também pelo dispositivo *Echo* popularmente conhecido como Alexa, produzido pela Amazon.

3.4.2 MQTT

O protocolo MQTT (do inglês, *Machine Queuing Telemetry Transport*), segundo Hillar (2017), é um protocolo máquina para máquina (M2M), que por ser mais leve que o protocolo HTTP, se encaixa melhor para realizar a troca de dados entre dispositivos em tempo real através

da internet. O MQTT é muito utilizado em projetos de sistemas embarcados, IoT, e ultimamente vem sendo utilizado também em aplicações web e aplicativos móveis.

O MQTT utiliza um padrão publica-inscreve para sua comunicação, este padrão necessita de um servidor que é conhecido como *broker*. Neste padrão, todos clientes se conectam ao *broker*, o cliente que envia mensagem ao servidor é um publicador. O *broker* então filtra as mensagens para os clientes que tem interesse nela, esses clientes interessados são os inscritos.

As principais características do MQTT são:

- Capacidade de transmitir uma quantidade volumosa de dados sem causar gargalo.
- Facilidade em transmitir dados de um cliente para vários clientes.
- Proteção de mensagens através de criptografia.
- Baixo consumo de energia.

O protocolo MQTT pode ser encontrado em aplicações como, monitoramento de tráfego, quiosques de ponto de venda automatizados, em sistemas de supervisão e aquisição de dados (SCADA), entre outros.

3.4.3 LoRa

O protocolo LoRa(do inglês, *Long Range*) é um protocolo desenvolvido pela empresa Semtech[2], se trata de uma tecnologia de comunicação para longas distâncias através da tecnologia à rádio (POLITI, 2022). A maior vantagem do protocolo LoRa é como seu próprio nome diz, o seu longo alance, que pode chegar a até 12Km de distância. O seu baixo consumo de energia é também uma grande qualidade desse protocolo.

A comunicação do LoRa funciona através da comunicação de dispositivos com um *gateway*, estes *gateways* enviam os dados para um servidor. O protocolo LoRa também utiliza um técnica conhecida como *Chirp Spread Spectrum* que garante alta imunidade a interferência de ruídos no envio de dados. As ondas de rádio do LoRa atuam em uma frequência abaixo de 1Ghz.

Devido ao seu longo alcance e baixa velocidade de transmissão de dados, o protocolo LoRa é utilizado em aplicações de coleta e monitoramento remoto de sensores, geralmente em ambientes rurais e ambientes urbanos de difícil acesso.

[2] <https://www.semtech.com/>

3.4.4 Wifi

Wi-Fi é uma tecnologia de rede sem fio que permite que dispositivos troquem dados entre si por meio de um roteador sem fio (Cisco, 2022). Roteadores sem fio, permitem que os dispositivos compatíveis com wifi se conectem com a internet. O padrão IEEE 802.11 define os protocolos para a comunicação dos dispositivos com suporte wifi.

- **802.11-1997:** Primeira versão do padrão 802.11, apresentado em 1997, opera em uma frequência entre 2,4 Ghz e 2,4835 Ghz. Tem uma taxa de transmissão de 1Mb/s ou 2Mb/s.
- **802.11b:** Lançado em 1999, opera na mesma frequência que a versão de 1997, agora suporta taxas de transmissão de 5,5Mb/s e 11Mb/s. Sua área de cobertura alcança até 400 metros em ambientes abertos, e 50 metros em ambientes fechados.
- **802.11g:** Apresentado em 2003 como uma atualização da versão 802.11b, é compatível com a versão anterior, opera em uma frequência de 2,4 Ghz. Sua taxa de transmissão pode chegar até 54 Mb/s, possui área de cobertura muito semelhante ao 802.11b.
- **802.11ax:** Também conhecido como wifi 6, foi apresentado em 2021, pode operar em frequências de 2,4 Ghz e 5 Ghz. Sua taxa de transmissão pode chegar até 9,6Gb/s.

Neste trabalho foi utilizado o protocolo wifi 802.11g, devido sua disponibilidade e atendimento aos requisitos, para conectar o ESP32 a internet por meio de um roteador sem fio.

3.5 Futuro da IoT

A IoT ainda é uma tecnologia recente e que ainda tem muito campo para ser explorado, ela vem impactando em várias áreas e aplicações, para Nižetić et al. (2020) as principais áreas que a IoT deve ter impacto nos próximos anos são, a saúde, *smart cities* e transportes. Segundo os autores, a saúde tem muito potencial com o avanço da IoT para o aprimoramento na qualidade de tratamento de pacientes. A recente pandemia da COVID-19 serviu para a idealização de novos dispositivos ou aplicações IoT, para o monitoramento e controle de pandemias.

Segundo Nižetić et al. (2020), o problema de superpopulação de cidades também pode ser resolvido através do conceito de cidades inteligentes. Onde o aumento da eficiência de problemas convencionais poderiam diminuir os impactos da superpopulação. Como por exemplo

a implementação de semáforos inteligentes para a redução do trânsito em grandes cidades, ou até mesmo a implementação de *smart grids* para um distribuição de energia eficiente.

Mas com essas novas soluções também surgem novos problemas que segundo Nižetić et al. (2020) alguns deles são: a falta de matéria-prima para acompanhar essa grande produção de dispositivos IoT; a minimização no consumo de energia; o aumento do lixo eletrônico com o surgimento e produção de novos dispositivos. Esses são alguns dos desafios que surgiram com o avanço da IoT, e que vão ser discutidos nos próximos anos.

4 PROTÓTIPO IOT PARA O MONITORAMENTO DE RUÍDOS

4.1 Construção do Protótipo

Neste capítulo será abordado todo o desenvolvimento do protótipo, desde o seu hardware, até os softwares e códigos utilizados. Ao final do capítulo serão mostrados os testes realizados e seus resultados.

4.1.1 Hardware

Como já foi explicado no capítulo 3, os dispositivos IoT geralmente são hardwares de baixo desempenho. A capacidade de processamento e armazenamento de uma placa Arduino ou ESP é muito inferior à de um notebook comum por exemplo. Enquanto o processador da placa utilizada neste trabalho possui dois núcleos com 240Mhz, um processador celeron N4020 de notebook possui um núcleo de 1.1Ghz .O consumo de energia de uma placa também é muito importante, visto que muitos de dispositivos IoT são alimentados por uma bateria.

Nas próximas subseções será detalhada toda a especificação do hardware utilizado no trabalho, e também serão listados os componentes que foram utilizados na montagem do circuito.

4.1.1.1 ESP32

Muito utilizado em projetos de internet das coisas devido ao seu ótimo custo-benefício (pode ser encontrado atualmente por R$60,00), o ESP32 é um poderoso módulo fabricado pela empresa Espressif[1]. Ele contém conectividade wi-fi, bluetooth e um microprocessador que é capaz de atender a uma grande variedade de projetos, desde os mais simples até os mais complexos. É ele que realiza toda a coleta de dados assim como a sua comunicação através de uma rede wi-fi. Neste projeto, o modelo utilizado foi o ESP32 WROOM 32.

[1] <https://www.espressif.com/>

Figura 4.1 – ESP32 WROOM 32

Fonte: Shopee

Principais características.

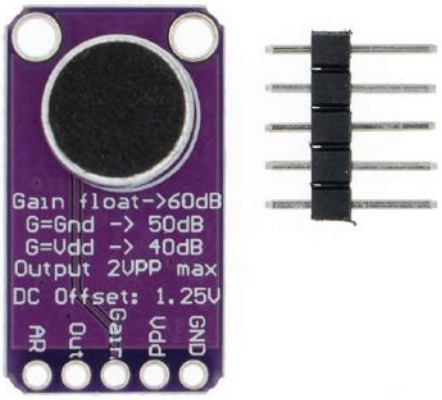
- Processador: XTensa 32-bit LX6 Dual Core 2-240Mhz
- Memória ROM: 448KBytes
- Memória RAM: 520Kbytes
- Memória Flash: 4MB
- Conexão Wifi 2.4Ghz (máximo de 150 Mbps)
- Bluetooth BLE 4.2
- Tensão de operação: 4,5 9V
- Conector micro-usb
- Conversor analógico digital (ADC)
- Compátivel com Arduino IDE

4.1.1.2 Microfone

Foi escolhido o microfone MAX9814 por se tratar de um microfone eletreto de baixo custo (Pode ser encontrado na faixa dos R$25,00). Seus diferenciais são o seu controle de

ganho automático(desligado neste trabalho) e sua alta sensibilidade, sendo uma ótima escolha para projetos acústicos.

Figura 4.2 – Sensor MAX9814



Fonte: Curtocircuito

AS principais caraterísticas são:

- Resposta de Frequência de 20Hz - 20 KHz;
- Tensão de Alimentação de 2.7V-5.5V;
- Ganho máximo de 40dB, 50dB ou 60dB;
- Controle automático de ganho;
- Baixa densidade de ruído de entrada.

4.1.1.3 Demais Componentes

Além do ESP32 e do microfone que são os principais componentes do hardware, foram utilizados para o correto funcionamento do circuito:

- 1 LED difuso verde 5mm;
- 1 LED difuso amarelo 5mm;
- 1 LED difuso vermelho 5mm;
- 3 Resistores de 220 Ω;

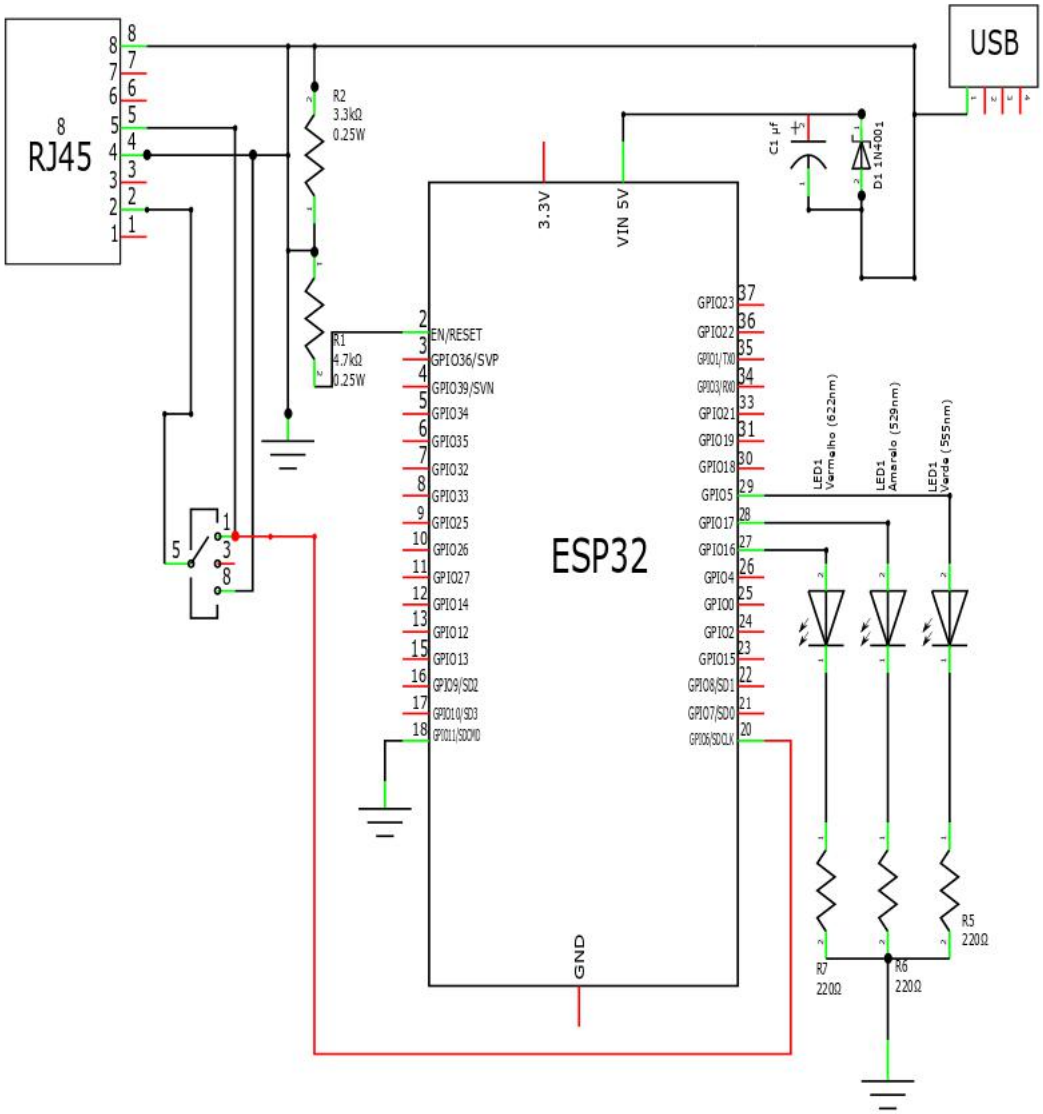
- 1 Resistor de 4.7K Ω;
- 1 Resistor de 3.3K Ω;
- 1 Diodo 1N4001;
- 1 Capacitor de 1µf;
- 1 Conector RJ45 Fêmea;
- 1 Conector USB.

4.1.1.4 Hardware Completo

O desenho do circuito completo do hardware foi feito pelo software Inkscape[2], uma ferramenta bem completa para desenho vetorial. A Figura 4.3 mostra como é feita a montagem do circuito, e a Figura 4.4 mostra o circuito do protótipo completo.

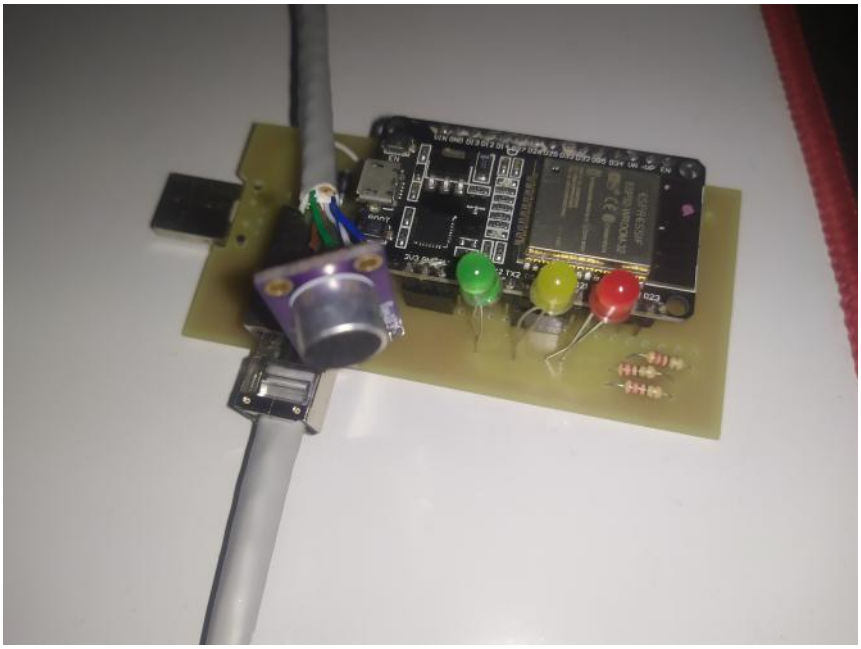
[2] <https://inkscape.org/pt-br/>

Figura 4.3 – Esquemático do Circuito



Fonte: Elaborado pelo autor

Figura 4.4 – Circuito Completo



Fonte: Elaborado pelo autor

4.1.2 Software

A principal ferramenta utilizada para o desenvolvimento do código-fonte para realizar a programação do ESP32 foi o Arduino IDE, um ambiente de desenvolvimento gratuito e de código aberto disponibilizado pela empresa Arduino[3]. Neste trabalho, o sistema operacional utilizado foi o Windows, mas a IDE também é compatível com sistemas operacionais Linux. A linguagem de programação suportada pela IDE é o C/C++, sendo essa então a linguagem utilizada na maior parte do projeto. Apesar de não se tratar de uma placa Arduino, o Arduino IDE disponibiliza bibliotecas compatíveis com outros módulos tais como o ESP32. Toda a configuração do ambiente para a compatibilidade será explica nas próximas seções.

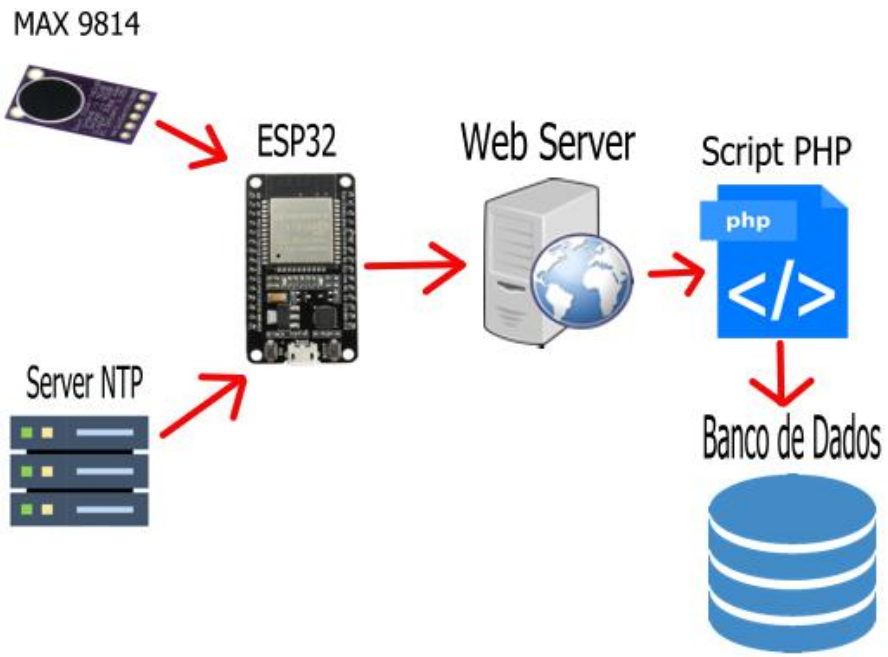
Além das tecnologias utilizadas para a programação do hardware, também foram utilizadas outras para a comunicação do ESP32 com um banco de dados: serviços MySQL e Apache, disponibilizados gratuitamente pelo XAMPP, para que seja feita uma comunicação em um servidor local. Para a comunicação com o banco de dados foi utilizado um servidor web para o envio de dados, essa comunicação através do servidor PHP acaba sendo mais eficiente e segura.

[3] <https://www.arduino.cc/>

Mais segura pois a maioria das bibliotecas MySQL para o ESP não suportam SSL e TLS (protocolos que podem ser implementados no servidor), e desta forma também não é preciso fornecer acesso remoto para um usuário MySQL, o que pode ser perigoso.

Mais eficiente pois os dados são processados por um script PHP, economizando memoria e CPU do ESP32. O script PHP consegue processar os dados com mais facilidade do que o script MySQL para o ESP32, simplificando o código do ESP32 na IDE Arduino. A Figura 4.5 mostra como ficou a arquitetura final do dispositivo.

Figura 4.5 – Arquitetura



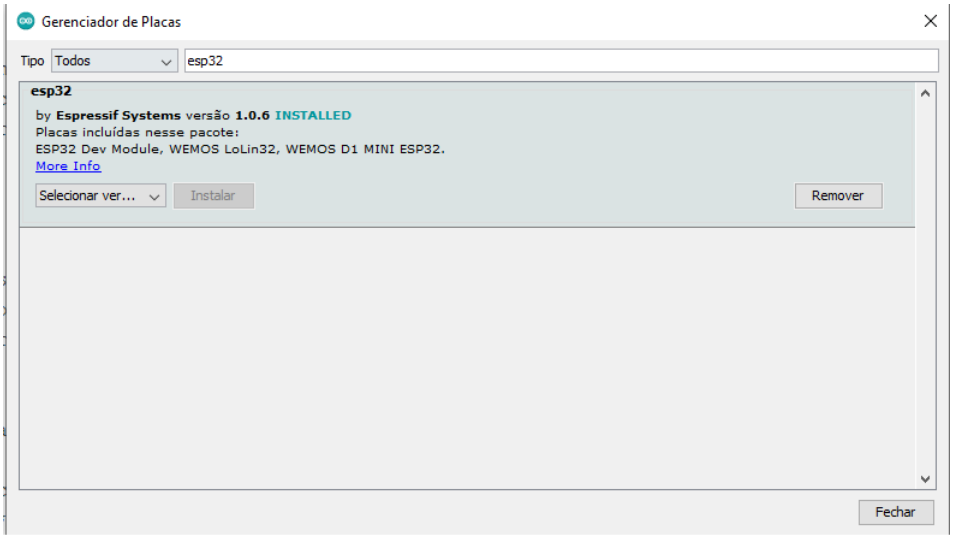
Fonte: Elaborado pelo autor

4.1.2.1 Configurando a IDE do Arduino para o ESP32

O ambiente de desenvolvimento integrado (IDE do inglês, *Internal Development Environment*) é uma IDE gratuita e open source quer pode ser adquirida no site <arduino.cc>. Com as configurações padrão, ele não funciona com ESP32, sendo necessária a instalação de algumas bibliotecas adicionais.

Primeiro é preciso adicionar as placas do ESP32 para a IDE, para isto basta navegar no menu de ferramentas, placas, gerenciador de placas, em seguida basta pesquisar pelas placas ESP32 disponibilizadas pela Espressif, como mostra a Figura 4.6

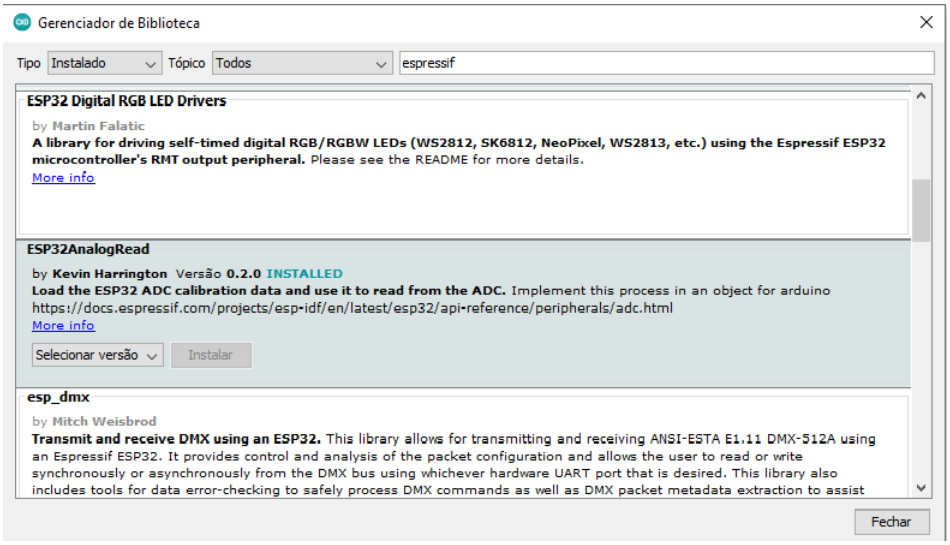
Figura 4.6 – Configurando Placa



Fonte: Arduino IDE

E por último é preciso instalar a biblioteca "ESP32 analogRead", para que a IDE consiga fazer a leitura da saída analógica do ESP32. Para isso é preciso acessar novamente o menu de ferramentas, gerenciar bibliotecas, e pesquisar pela biblioteca, como mostra a Figura 4.7

Figura 4.7 – Configurando Biblioteca



Fonte: Arduino IDE

Agora a IDE do arduino já está totalmente compatível e configurada para ser utilizada com a placa ESP32.

4.1.2.2 Configurando Banco de Dados MySQL

Primeiramente foi preciso criar um administrador MySQL com permissões apenas de localhost, para isso é preciso rodar os seguintes comandos:

Figura 4.8 – Criando Admin

```
-cd C:\XAMPP\mysql\bin
mysqladmin -u root password senhaDoRoot
```

Primeiro é acessado o diretório onde está localizado o MySQL no XAMPP e em seguida é inserido o comando com nome de usuário e senha. Em seguida é feito o login no MySQL através do comando:

Figura 4.9 – Login

```
mysql.exe -u root -p
```

Agora é preciso criar um usuário e conceder as devidas permissões para ele, para isso são inseridos os seguintes comandos no terminal:

Figura 4.10 – Usuario

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'nomeDoUsuario';
GRANT ALL PRIVILEGES ON *.* TO 'usuario'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

Com o usuário criado é preciso criar o banco de dados para armazenar os ruídos coletados, o banco de dados é criado digitando o seguinte comando:

Figura 4.11 – Cria Banco de Dados

```
CREATE DATABASE db_esp32 CHARACTER SET = 'utf8' COLLATE = 'utf8_general_ci';
```

Agora com o banco de dados já criado é preciso criar a tabela onde serão armazenados os dados dos ruídos, para isso é preciso criar três colunas. Uma para guardar a pressão sonora em dB, uma para guardar o horário em que o ruído foi coletado e uma para guardar o id do ruído. E para isso deve ser inserido os seguintes comandos:

Figura 4.12 – Cria Tabela de ruídos

```
USE db_esp32;

CREATE TABLE ruidos (
        ruido_id    INT UNSIGNED NOT NULL AUTO_INCREMENT,
        ruido_dBSPL DOUBLE DEFAULT 0.00,
        ruido_time  timestamp current_timestamp() ON UPDATE CURRENT_TIMESTAMP()
        PRIMARY KEY (ruido_id)
);
```

Agora o banco de dados já está pronto para ser utilizado, sendo necessária a configuração do código do ESP32 no arduino IDE.

4.1.2.3 Configurando Script PHP

Como já foi explicado anteriormente, para que a transferência e segurança dos dados seja otimizada, é preciso fazer com que o bando de dados e o ESP32 se comuniquem através de um script PHP que recebe os dados e envia para o bando de dados. Como mostra o código na Figura 4.13, o script recebe uma *query* com o horário em que foi coletado o ruído e também o valor do ruído. Em seguida ele faz um login no banco de dados com o usuário e endereço passados como parâmetros, então ele insere nas colunas corretas do banco de dados suas informações. Se tudo ocorrer corretamente, o script retorna uma mensagem notificando o sucesso da inserção, ou então ele encerra a aplicação e retorna um número de erro.

Figura 4.13 – Código do Script PHP

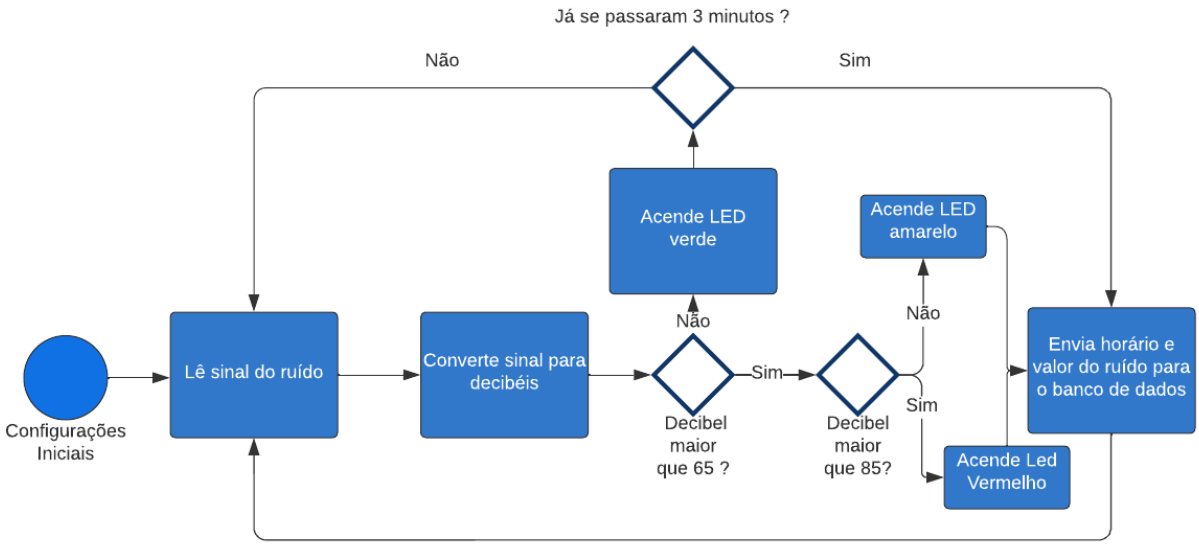
```
<?php

if(isset($_GET["horario"])) {
   $valor = $_GET["horario"]; //recebe horario

   $servername = "localhost";
   $username = "root";
   $password = "YOUR_ROOT_PASSWORD";
   $database_name = "db_esp32";

   // Cria conexao MySQL com o servidor
   $connection = new mysqli($servername, $username, $password,
        $database_name);
   // Checa conexao
   if ($connection->connect_error) {
      die("MySQL connection failed: " . $connection->connect_error);
   }

   $sql = "INSERT INTO ruidos (ruido_time) VALUES ($horario)";

   if ($connection->query($sql) === TRUE) {
      echo "New record created successfully";
   } else {
      echo "Error: " . $sql . " => " . $connection->error;
   }

   $connection->close();
} else {
   echo "valor is not set in the HTTP request";
}
?>
```

Fonte: Elaborado pelo autor

4.1.2.4 Código Principal

Como a interface de desenvolvimento trabalha apenas com linguagem C/C++, a mesma foi utilizada no desenvolvimento do código principal. Nesta subseção serão apresentadas apenas as principais funções do código, sendo elas: a coleta e conversão de ruídos; conexão com uma rede wi-fi; conexão com o servidor ntp; envio de informações para o banco de dados e configuração e tratamento dos leds. O fluxograma da Figura 4.14 mostra resumidamente como ciclo principal do código trabalha.

Figura 4.14 – Fluxograma do Código Principal



Fonte: Elaborado pelo autor

Coleta de ruídos e conversão para dBSPL

Para que seja possível a coleta de dados, é preciso configurar a saída analógica. Por padrão, o canal ADC do ESP32 vem com 10 bits, fazendo com que seja possível ler apenas de 0 a 1023 valores de pressão sonora. É preciso aumentar essa faixa para que se tenha dados com maior alcance e precisão, para isso é preciso incluir a biblioteca `adc.h` que permite aumentar o alcance para até 12 bits, garantido uma leitura entre 0 e 4095 de sinal. Essa biblioteca é adicionada no começo do código com a linha presente na Figura 4.15.

Figura 4.15 – Adicionando Biblioteca adc.h

```
#include <driver/adc.h>
```

Após adicionada a biblioteca, é preciso configurar a saída ADC para 12 bits, isso é feito inserindo as seguintes linhas de código na função setup (Figura 4.16).

Figura 4.16 – Configurando o ADC

```
adc1_config_width(ADC_WIDTH_BIT_12);
adc1_config_channel_atten(ADC1_CHANNEL_0,ADC_ATTEN_DB_0);
```

A primeira linha configura o canal ADC para trabalhar com 12bits e a segunda linha configura uma atenuação de 0 decibéis no canal. Com o canal já configurado é preciso coletar os sinais de ruído, para isso é criado um vetor de tamanho igual ao número de amostras que será coletado, pois será feita uma coleta em função do número de amostras, essa coleta é feita através das seguintes linhas de código:

Figura 4.17 – Coletando Sinais

```
microseconds = micros();
somaDoVetor=0;
 for (int i = 0; i < SAMPLES; i++){
      sinal[i] = adc1_get_raw(ADC1_CHANNEL_0);
 while (micros() - microseconds < sampling_period_us) {
     //loop vazio
 }
   microseconds += sampling_period_us;
}
```

Os sinais coletados precisam ser calculados em seu valor RMS(do inglês, *Root Mean Square*) para ser feita a conversão para dBSPL(do inglês, *Decibels Sound Pressure Level*). Para isso é feito o somatório do valor RMS de cada posição do vetor, como já explicado em capítulos anteriores, o valor RMS é dado pela Fórmula:

$$Vrms = \sqrt{\frac{\sum(sinal - b)^2}{amostras}} \tag{4.1}$$

Com a diferença que agora é preciso subtrair o sinal lido com o bias (b, na figura 4.1) do microfone. O bias pode ser calculado através do valor lido quando não há nenhum som sendo reproduzido no momento da leitura. E a conversão em dBSPL é calculada pela mesma fórmula apresentada no capítulo de coleta de ruídos com uma pequena diferença:

$$dBSPL = 20 \cdot log\left(\frac{p}{pRef} + 94\right) \tag{4.2}$$

Na nova fórmula é preciso somar a divisão das pressões ao valor constante de 94, pois ao realizar a divisão do RMS da pressão de referência estabelece-se o ponto 0.

A pressão de referência deve ser medida através de uma calibração do valor RMS de um sinal quando exposto a um ruído de 1Khz em 94dB, para auxiliar na calibração foi utilizado um microfone já calibrado para descobrir o valor referente a 94dB. O cálculo do valor RMS e da conversão para dBSPL é mostrada nas seguintes linhas de código:

Figura 4.18 – Calculo RMS e Conversão dBSPL

```
for (int i = 0; i < SAMPLES; i++){
      somaDoVetor += sqrt(pow(sinal[i]-bias,2)/SAMPLES);
}

dBSPL = 20 * log10(somaDoVetor / referencia) + 94;
```

Com isso, o programa já é capaz de exibir o valor de pressão sonora convertido para decibéis.

Conexão com rede Wi-Fi

Para que seja possível enviar os dados coletados para um banco de dados, é preciso conectar o ESP32 à internet. O ESP possui um módulo Wi-Fi que pode ser facilmente configurado. Primeiramente é preciso incluir a biblioteca `wifi.h` no projeto, em seguida criar variáveis para armazenar a senha e o nome de usuário da rede que deseja se conectar, como mostra o seguinte trecho de código:

Figura 4.19 – Configuração Inicial Wi-Fi

```
#include <WiFi.h>

const char WIFI_SSID[] = "nomeDaRede";
const char WIFI_PASSWORD[] = "senhaDaRede";
```

Agora basta inicializar a conexão com a rede na função setup do programa: função `wifi.Begin()` da biblioteca wi-fi. Essa função recebe como parâmetro o nome e usuário da rede, em seguida é verificado o estado da conexão através da função `wifi.status()` para que seja retornada uma mensagem de conexão bem sucedida, junto com o IP da conexão, caso nada de errado aconteça. Essa conexão pode ser vista no trecho de código da Figura 4.20

Figura 4.20 – Conexão Wi-Fi

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.println("Connecting");
while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.print("Connected to WiFi network with IP Address: ");
Serial.println(WiFi.localIP());
```

Agora o ESP já está pronto para realizar requisições ou enviar informações para alguma outra aplicação.

Conexão com servidor NTP

Para esta aplicação é interessante saber o horário preciso em que um ruído foi coletado, e para isto será utilizado um servidor NTP. O NTP (*Network Time Protocol*) é um protocolo de internet para sincronização de relógios entre dispositivos. Neste trabalho, será utilizado o servidor "br.pool.ntp", para a requisição de horário.

Para a configuração do servidor NTP é preciso incluir a biblioteca time.h, pois será necessário utilizar uma função dessa biblioteca, também é necessário criar algumas variáveis extras para guardar os parâmetros de fuso horário, horário de verão, endereço do servidor NTP utilizado e ajudar no envio para o script PHP futuramente, como mostra o trecho de código:

Figura 4.21 – Configurando NTP

```
#include "time.h"
const char* ntpServer = "br.pool.ntp.org";//endereco do servidor ntp
const long  gmtOffset_sec = -10800;// UTC -3 * 60 * 60
const int   daylightOffset_sec = 0;//horario de verao desligado
String horario = "";//variavel para facilitar o envio ao bd
```

Agora é preciso configurar a IDE do arduino para que o ESP32 use o servidor NTP escolhido para a referência de horário, isto pode ser feito pela função `configTime` da biblioteca `time.h`, onde será passado como parâmetro o endereço do servidor NTP, o fuso horário e o horário de verão, esta linha deve ser inserida dentro da função setup do arduino, o trecho desta função é mostrada na Figura 4.22.

Figura 4.22 – Configurando o Horário do Arduino

```
configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
```

Para facilitar a requisição de horário, foi criada uma função que guarda a string do horário em uma variável global que será enviada para o banco de dados. A função recebe o horário do servidor NTP e armazena seus valores em uma variável, a quantidade de valores podem ser controlados editando os caracteres passados como parâmetro na função `strftime`. O trecho de código da função completa se encontra na Figura 4.23.

Figura 4.23 – Função para Receber Horário

```
void getHorario(){
  time_t rawtime;
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }
  char timeStringBuff[50];
  strftime(timeStringBuff, sizeof(timeStringBuff),
  "%A, %B %d %Y %H:%M:%S", &timeinfo);
  horario = String(timeStringBuff);
}
```

Com isto é possível receber o horário atual em qualquer parte do código chamando a função `getHorario()`.

Conexão com servidor HTTP

Como já foi explicado anteriormente, foi utilizado um servidor HTTP para realizar o envio das amostras para o banco de dados. Para que seja possível se conectar ao servidor inclui-se a biblioteca `HTTPClient.h`, ela fornece as funções necessárias para que a IDE do arduino se conecte ao HTTP. Para realizar o envio, é preciso informar o endereço do servidor, o diretório do script, e os dados a serem enviados, essas informações foram guardadas em variáveis. Também foi preciso criar um objeto do tipo HTTP para realizar a conexão, o trecho de código da Figura 4.24 mostra como foram feitos esses procedimentos.

Figura 4.24 – Configuração do HTTP

```
#include <WiFi.h>

HTTPClient http;
String HOST_NAME = "http://192.168.1.6";//ip do server http
String PATH_NAME   = "/script.php";//caminho do arquivo script
String PATH_NAME2   = "/scriptHora.php";//caminho do arquivo script
String queryString = "?valor=";//valor a ser enviado
String queryStringFinal = "";
String dbStr = "";
String queryHora = "";
```

Para facilitar o envio de dados foi criada uma função que recebe como parâmetro os dado a ser enviado e o script a ser utilizado, a função então faz a requisição para o servidor HTTP que irá rodar o script e guardar o dado recebido no banco de dados. A Figura 4.25 mostra como esta função foi implementada.

Figura 4.25 – Função do Serviço HTTP

```
void enviaParaBd(String var, String local){
  http.begin(HOST_NAME + local + var); //HTTP
  int httpCode = http.GET();
  if(httpCode > 0) {
    // Arquivo encontrado no server
    if(httpCode == HTTP_CODE_OK) {
      String payload = http.getString();
      Serial.println(payload);
    } else {
      Serial.printf("[HTTP] GET... code: %d\n", httpCode);
    }
  } else {
    Serial.printf("[HTTP] GET... failed, error: %s\n",
        http.errorToString(httpCode).c_str());
  }
 http.end();
}
```

Caso aconteça algum erro durante a comunicação com o servidor, a função retorna uma mensagem com o código do erro ocorrido.

Indicadores de perigo

Por fim foi implementada uma comunicação com os LEDs para que seja possível o usuário identificar quando o ruído está acima dos níveis de decibéis considerados saudáveis de acordo com a norma ABNT NBR 10151. Quando identificado um ruído acima de 65dB e menor que 85dB, o LED amarelo acenderá e chamará a função `getHTTP()` que irá enviar o ruído daquele momento e o seu horário. Caso o nível de pressão sonora ultrapasse 85dB, o LED vermelho é aceso e é enviado este ruído junto com horário para o banco de dados, se nenhuma dessas condições são atendidas, o LED verde é aceso e nada é enviado para o servidor MySQL.

Para que seja possível implementar as funções dos LEDs primeiro é preciso configura-los na função setup do arduino, é preciso configurar os pinos dos LEDs como saída, para isto é utilizada a função pronta do arduino `pinmode()` , como mostra a Figura 4.26.

Figura 4.26 – Configura os Pinos dos LEDs

```
pinMode(5, OUTPUT); //configura o pino 5 como saida
pinMode(18, OUTPUT); //configura o pino 18 como saida
pinMode(19, OUTPUT); //configura o pino 19 como saida
```

E a figura 4.27 mostra como foi feita a implementação dos condicionais dos LEDs, note que sempre que um LED diferente é aceso todos os outros são apagados.

Figura 4.27 – Condicionais para Controle dos LEDs

```
if(dBSPL < 65){
  digitalWrite(18, LOW);// DESLIGA LED AMARELO
  digitalWrite(19, LOW);// DESLIGA LED VEMRELHO
  digitalWrite(5, HIGH);//LIGA LED VERDE
}else if(dBSPL >= 65 and dBSPL < 85 ){
  digitalWrite(5, LOW);//DESLIGA LED VERDE
  digitalWrite(19, LOW);// DESLIGA LED VEMRELHO
  digitalWrite(18, HIGH);// LIGA LED AMARELO

  enviaParaBd(queryHora,PATH_NAME2);
  enviaParaBd(queryStringFinal,PATH_NAME);
}else if(dBSPL >= 85) {
  digitalWrite(5, LOW);//DESLIGA LED VERDE
  digitalWrite(18, LOW);// DELIGA LED AMARELO
  digitalWrite(19, HIGH);// LIGA LED VEMRELHO

  enviaParaBd(queryHora,PATH_NAME2);
  enviaParaBd(queryStringFinal,PATH_NAME);
}
```

Para que o LED seja aceso ou apagado é preciso passar como parâmetro se ele recebe ou não corrente elétrica, `HIGH` para acender o LED, e `LOW` para desliga-lo.

4.2 Testes

Para a validação de precisão da medição do microfone do dispositivo, foi realizado um teste comparativo entre o microfone MAX9814 e o microfone Dayton Audio Imm-6. O Imm-6 é um microfone de qualidade profissional para calibração e gravação, ele é produzido pela empresa Dayton Audio[4]. Ele foi inicialmente desenvolvido para dispositivos Apple, mas já poussiu compatibilidade com dispositivos Android e Windows. Os testes foram realizados dentro de uma caixa de isolamento acústico. De um lado da caixa foram posicionados os microfones colados, e do outro lado, uma caixa de som que transmitiu tons para comparar o valor calculado por cada microfone. Após o posicionamento dos equipamentos, a caixa é fechada e então é reproduzido um tom através da caixa de som. Os valores medidos pelos microfones são monitorados. A caixa é feita de madeira MDF, é e coberta internamente por uma camada de lã de vidro e outra de espuma isolante.

Figura 4.28 – Posicionamento da Caixa de Som

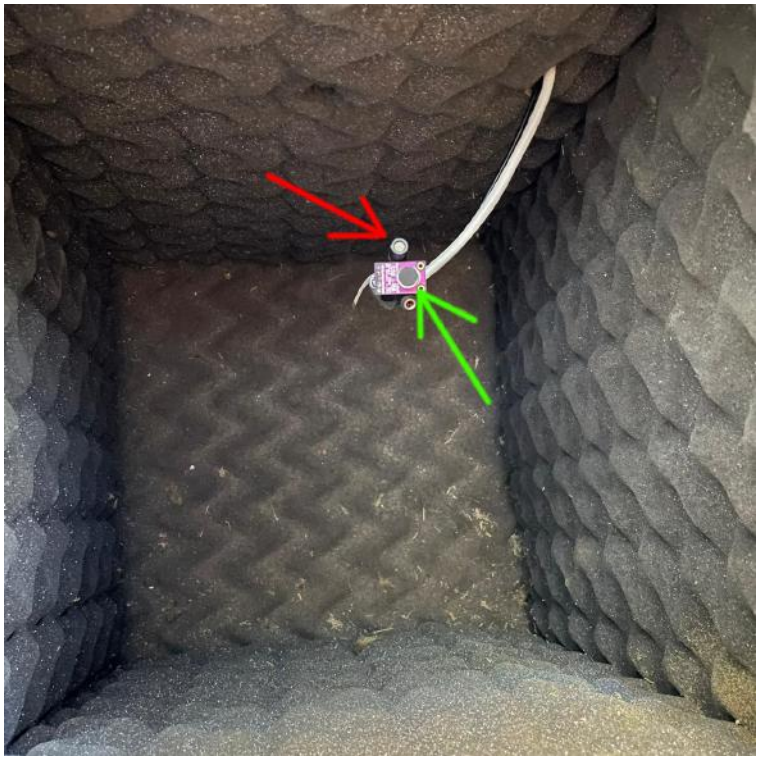
Fonte: Elaborado pelo autor

[4] <https://www.daytonaudio.com/>

Figura 4.29 – Caixa Fechada

Fonte: Elaborado pelo autor

Figura 4.30 – Posicionamento dos Microfones



Vermelho: Microfone Imm-6
Verde: Microfone MAX9814
Fonte: Elaborado pelo autor

Com o microfone validado, foram realizados testes de monitoramento de um ambiente. O ambiente monitorado foi um ambiente de convivência de uma universidade, nos períodos diurno e noturno. Para o período diurno, o teste teve início às duas e meia da tarde, e terminou às quatro e meia. O dispositivo foi configurado para enviar o ruído captado para um banco de dados, caso seu valor ultrapassasse 65 decibéis. Caso o dispositivo não detecte nenhum valor acima de 65 decibéis durante um período de três minutos, ele então envia o valor lido naquele momento. O dispositivo sempre envia o valor em decibéis junto do horário em que ele foi lido. O monitoramento noturno seguiu os mesmos critérios do diurno exceto que o valor mínimo de leitura para envio é de 55 dB. O teste noturno teve início às seis da tarde, e terminou às sete e meia.

As figuras 4.31 e 4.32 mostram o ambiente em que o teste foi realizado e o local em que o dispositivo foi instalado. Devido a autenticação extra da rede da universidade UFLA, foi necessário a conexão com a internet através de um roteador wifi auxiliar, que foi instalado próximo ao ESP32.

Figura 4.31 – Ambiente Monitorado

Fonte: Elaborado pelo autor

Figura 4.32 – Dispositivo Monitorador

Fonte: Elaborado pelo autor

4.3 Resultados

Após a realização dos testes com a caixa acústica juntamente com o microfone desenvolvido e um microfone profissional, foi possível obter valores bem próximos de calibragem em dB. A calibração do microfone desenvolvido ainda difere um pouco quando nos extremos de saturação, como pode ser visto na Tabela 4.1. A comparação começou com o menor valor que o microfone MAX9814 do ESP32 conseguia capturar: enquanto o ESP32 calculava 54dB, o microfone profissional calculava 51dB; quanto mais se aumentava o tom mais próximo os valores de calibração ficavam. À partir do valor v8 as calibrações começam a ficar idênticas, e começam e se dispersar novamente a partir do valor V_{20} lido.

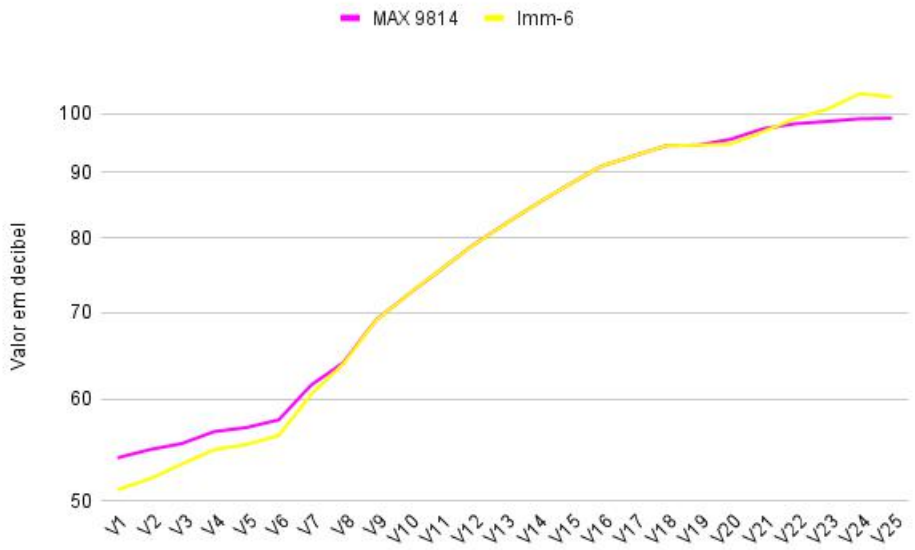
Quadro 4.1 – Validação de Calibragem

Valores	*ESP*32	*Microfone*
V_1	54	51
V_2	54,8	52
V_3	55,4	53,4
V_4	56,6	54,8
V_5	57,01	55,3
V_6	57,8	56,2
V_7	61,5	60,5
V_8	64,03	63,9
V_9	69,05	69,01
V_{10}	72,3	72,3
V_{11}	75,56	75,6
V_{12}	78,94	78,9
V_{13}	82,02	82
V_{14}	85,1	85,1
V_{15}	88,1	88,1
V_{16}	91	91
V_{17}	92,7	92,7
V_{18}	94,4	94,4
V_{19}	94,5	94,5
V_{20}	95,5	94,7
V_{21}	97,31	96,8
V_{22}	98,19	99,1
V_{23}	98,6	100,8
V_{24}	99,06	103,6
V_{25}	99,15	103,8

Fonte: Elaborado pelo autor

O gráfico da figura 4.33 consegue facilitar a visualização dos momentos em que os valores medidos de pressão sonora começam a convergir e divergir uns dos outros. É notável que há uma diferença entre os valores medidos, mas em sua grande maioria as medições são equivalentes, tendo diferenças apenas quando os valores se aproximam da saturação do microfone do ESP32.

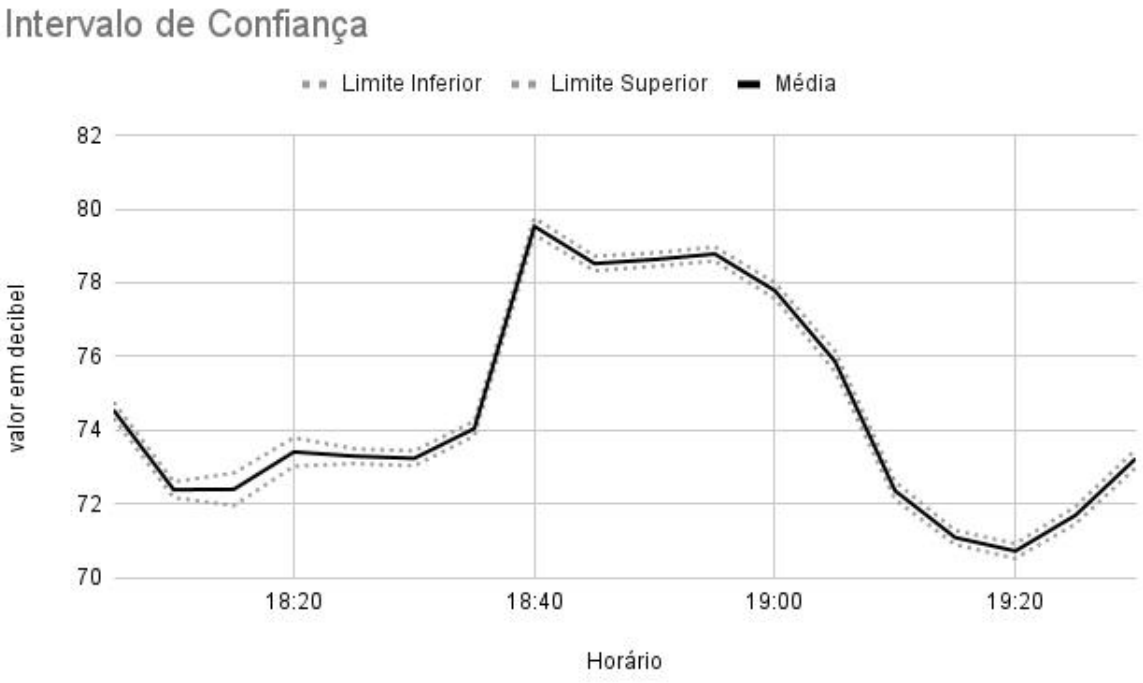
Figura 4.33 – Comparação entre medições



Fonte: Elaborado pelo autor

O resultado do monitoramento de ruídos durante o período noturno pode ser visto nos gráficos a seguir:

Quadro 4.2 – Gráfico do Intervaldo de Confiança do Monitoramento Noturno



Fonte: Elaborado pelo autor

O gráfico mostra a média dos valores num intervalo de 5 minutos juntamente dos limites de intervalo de confiança, dos ruídos em decibéis durante o período de monitoramento. A média dos ruídos se mantiveram sempre acima de 70 decibéis, o que é bem acima dos 55 decibéis exigidos pela norma NBR 10151. Os limites de confiança tiveram maior diferença no período de seis horas e 10 minutos, e seis horas e trinta minutos. No intervalo de seis horas e quarenta minutos e sete horas os valores atingiram uma média maior que os demais intervalos.

Também é possível analisar o monitoramento através Tabela 4.3:

Quadro 4.3 – Tabela de Monitoramento Noturno

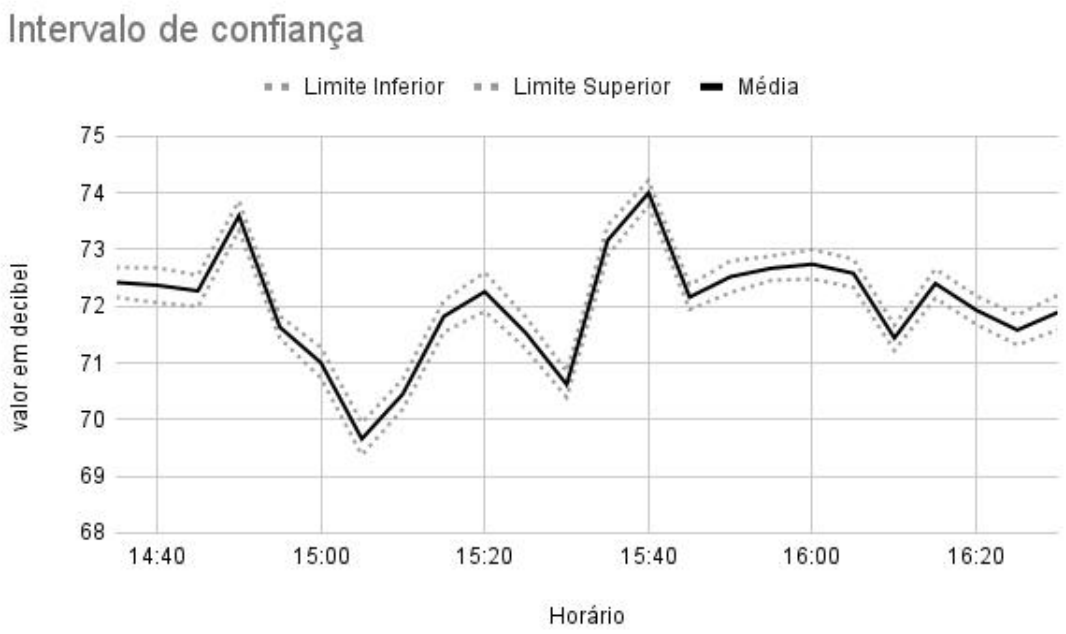
Ruído ID	Valor dB	Horário				
1	74,26	2022-08-25 18:00:00				
2	74,03	2022-08-25 18:00:00				
3	75,03	2022-08-25 18:00:01				
4	80,08	2022-08-25 18:00:02				
5	76,11	2022-08-25 18:00:02				
6	76,03	2022-08-25 18:00:03				
6033	72,63	2022-08-25 19:29:29				
6034	72,78	2022-08-25 19:29:29				
6035	74,29	2022-08-25 19:29:30				
6036	73,64	2022-08-25 19:29:31				
6037	73,26	2022-08-25 19:29:31				
6038	70,84	2022-08-25 19:29:32				
6039	69,57	2022-08-25 19:29:33				
6040	70,47	2022-08-25 19:29:33				
6041	69,68	2022-08-25 19:29:34				
6042	71,26	2022-08-25 19:29:35				
6043	72,18	2022-08-25 19:29:35				
6044	69,37	2022-08-25 19:29:36				
6045	69,89	2022-08-25 19:29:36				
6046	73,39	2022-08-25 19:29:37				
6047	71,87	2022-08-25 19:29:38				
6048	72,63	2022-08-25 19:29:38				
6049	72,5	2022-08-25 19:29:39				
6050	73,83	2022-08-25 19:29:40				
6051	70,49	2022-08-25 19:29:40	Valor MAX	Valor MIN	Moda	Média
6052	70,06	2022-08-25 19:29:41	93,09	66,21	73,4	73,75

Fonte: Elaborado pelo autor

Como é mostrado na tabela, o maior valor lido durante o período da noite foi de 93.09 decibéis, uma valor bem acima do limite estipulado pela norma. Já o menor valor registrado foi de 66.21 decibéis, que ainda é um valor acima do mínimo exigido pela norma. Os valores mais comuns dentre os 6052 ruídos registrados foi o de 73.4 decibéis. E por fim a média total dos valores foi de 73.75 decibéis.

Já os resultados do monitoramento diurno podem sem vistos nos gráficos a seguir:

Quadro 4.4 – Gráfico do Intervalo de Confiança do Monitoramento Diurno



Fonte: Elaborado pelo autor

O gráfico da média de monitoramento diurno se mostrou mais consistente do que o gráfico noturno, com picos máximo menor, e valores mais próximos durante o período monitorado. Os limites do intervalo de confiança se mostraram maior que o período noturno.

Quadro 4.5 – Tabela de Monitoramento Diurno

Ruído ID	Valor dB	Horário				
1	71,92	2022-08-26 14:30:00				
2	73,25	2022-08-26 14:30:00				
3	71,32	2022-08-26 14:30:01				
4	74,03	2022-08-26 14:30:02				
5	79,82	2022-08-26 14:30:02				
8190	70,39	2022-08-26 16:29:37				
8191	71,05	2022-08-26 16:29:37				
8192	70,84	2022-08-26 16:29:38				
8193	70,43	2022-08-26 16:29:38				
8194	69,6	2022-08-26 16:29:39				
8195	72,18	2022-08-26 16:29:40				
8196	70,24	2022-08-26 16:29:40	Valor MAX	Valor Min	Moda	Média
8197	70,2	2022-08-26 16:29:41	84,53	65,95	71,84	72,31

Fonte: Elaborado pelo autor

Como pode ser visto o valor máximo registrado no monitoramento diurno foi relativamente menor que o do período noturno, nove decibéis a menos do que o período noturno. O valor mínimo também foi menor que o monitoramento noturno, mas ainda assim o valor não satisfez o mínimo exigido pela norma (65dB para o período diurno). O valor mais frequente foi ode 71,84 decibéis, e a média de uma amostra de mais de 8mil ruídos foi de 72.31 decibéis.

Com os resultados foi possível monitorar os níveis de ruídos de um ambiente de convivência de uma universidade em períodos noturnos e diurnos. E com esses resultados é possível notar que medidas devem ser tomadas, já que em nenhum momento o dispositivo registrou um valor em decibéis menor que o exigido pela norma, tanto em período noturno quanto diurno. Primeiramente poderiam ser realizadas campanhas para conscientizar sobre os níveis de ruídos elevados naquele ambiente. E caso as campanhas não sejam suficientes, medidas efetivas devem ser tomadas.

5 CONCLUSÃO

5.1 Considerações finais

O principal objetivo deste trabalho foi mostrar o funcionamento de um protótipo IoT que fosse capaz de coletar ruídos de um ambiente e enviar os dados coletados para um banco de dados através de uma rede. O dispositivo desenvolvido conseguiu realizar medições bem próximas de um microfone já calibrado, sendo apenas limitado por seu hardware. Um dos grandes desafios encontrados no decorrer deste trabalho foi a conversão do sinal analógico de pressão sonora para digital, devido a falta de informações disponíveis na folha de especificações do microfone MAX 9814.

O dispositivo desenvolvido foi capaz de coletar ruídos com uma certa limitação na sua faixa de valores e precisão. Foi capaz de enviar dados para um banco de dados através de uma rede *WI-FI*. Graças ao módulo *WI-FI* do ESP32 é possível fazer um monitoramento a distância dos valores quando o nível de ruído ultrapassa aquele definido pela norma ABNT NBR 10151 e 10152. O dispositivo também demonstra fisicamente através de LEDs quando os níveis estão aceitáveis, próximos do limite e quando passaram os limites. Após testes, foi possível monitorar o ambiente de uma área de convivência da Universidade Federal de Lavras (UFLA) e analisar os dados coletados remotamente por um banco de dados.

Por fim o trabalho foi uma iniciativa que serve como base para vários trabalhos futuros na área de IoT e de monitoramento remoto de ruídos, o aperfeiçoamento do hardware existente também pode ser feito para se obter melhores resultados.

5.2 Trabalhos futuros

Alguns trabalhos futuros que podem ser feitos são a produção de mais dispositivos para realizar um mapeamento de ruído de uma área como um mapa de ruídos de uma universidade por exemplo. O trabalho também pode ser estendido utilizando uma análise mais detalhada dos ruídos utilizando a transformada rápida de Fourier (FFT) por exemplo. A implementação de uma ponderação A nas medições, também é uma possibilidade para extensão do trabalho. Também podem ser implementada técnicas de inteligência artificial que sejam capaz de classificar os tipos de ruídos. O dispositivo desenvolvido pode ser aprimorado utilizando um microfone profissional de qualidade superior, para ter um maior alance de valores medidos, e maior pre-

cisão, a calibração do microfone através de um decibelímetro de alta qualidade também pode aprimorar a qualidade de precisão.

REFERÊNCIAS

ALAM, P. et al. Noise monitoring, mapping, and modelling studies–a review. **Journal of Ecological Engineering**, v. 21, n. 4, 2020.

ALÍAS, F.; ALSINA-PAGÈS, R. M. Review of wireless acoustic sensor networks for environmental noise monitoring in smart cities. **Journal of sensors**, Hindawi, v. 2019, 2019.

ALONSO, C. A. dos S. C. **Tópicos de Matemática e Música na Educação Básica**. Tese (Doutorado) — PUC-Rio, 2016.

ASHTON, K. That 'internet of things' thing. In: . [S.l.: s.n.], 1999.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 10152: Níveis de ruído para conforto acústico**. [S.l.]: ABNT Rio de Janeiro, 1987.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 10151: Medição e Avaliação de pressão sonora em áreas habitadas - Aplicação de uso geral**. 2019.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787–2805, 2010. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128610001568>.

BISTAFA, S. **Acústica aplicada ao controle do ruído**. Blucher, 2018. ISBN 9788521212843. Disponível em: <https://books.google.com.br/books?id=RCO7DwAAQBAJ>.

BLOKDYK, G. **Smart Traffic Light a Complete Guide - 2020 Edition**. Emereo Pty Limited, 2019. ISBN 9780655913863. Disponível em: <https://books.google.com.br/books?id=NOBTyQEACAAJ>.

CHERUVU, S. et al. **Demystifying Internet of Things Security: Successful IoT Device/Edge and Platform Security Deployment**. Apress, 2019. ISBN 9781484228968. Disponível em: <https://books.google.com.br/books?id=YSKpDwAAQBAJ>.

Cisco. **O que é Wi-Fi?** 2022. https://www.cisco.com/c/pt$_br/products/wireless/what-is-wifi.html. Acessado em Agosto de 2022.

EUROPEU, P.; EUROPEIA, C. da U. Directiva 2002/49/ce, de 25 de junho. **Jornal Oficial das Comunidades Europeias L**, v. 189, n. 18.7, 2002.

FORBES. **How IoT Is Changing The Science Of Medicine**. 2018.

GALEGALE, G. P. et al. Internet das coisas aplicada a negócios-um estudo bibliométrico. **JISTEM-Journal of Information Systems and Technology Management**, SciELO Brasil, v. 13, p. 423–438, 2016.

GANIME, J. et al. O ruído como um dos riscos ocupacionais: uma revisão de literatura. **Enfermería Glob**, v. 19, p. 1–15, 2010.

HALEGOUA, G. **Smart cities**. [S.l.]: MIT Press, 2020.

HILLAR, G. **MQTT Essentials - A Lightweight IoT Protocol**. Packt Publishing, 2017. ISBN 9781787285149. Disponível em: <https://books.google.com.br/books?id=40EwDwAAQBAJ>.

HUH, J.-H.; CHOI, J.-H.; SEO, K. Smart trash bin model design and future for smart city. **Applied Sciences**, MDPI, v. 11, n. 11, p. 4810, 2021.

KUMAR, P. M. et al. Cloud and iot based disease prediction and diagnosis system for healthcare using fuzzy neural classifier. **Future Generation Computer Systems**, Elsevier, v. 86, p. 527–534, 2018.

LEITE, J. E.; MARTINS, P. S.; URSINI, E. L. A internet das coisas (iot): Tecnologias e aplicações. **School of Technology, University of Campinas (UNICAMP)**, 2017.

MILCHRAM, C. et al. Designing for justice in electricity systems: A comparison of smart grid experiments in the netherlands. **Energy Policy**, Elsevier, v. 147, p. 111720, 2020.

Ministério do Trabalho e Previdência. **NR-15: Atividades e operações insalubres**. 2022. https://www.gov.br/trabalho-e-previdencia/pt-br/composicao/orgaos-especificos/secretaria-de-trabalho/inspecao/seguranca-e-saude-no-trabalho/ctpp-nrs/norma-regulamentadora-no-15-nr-15. Acessado em Agosto de 2022.

MORAES, A. de; TAKASHI, V. **Segurança Em IoT: Entendendo os riscos e ameaças em IoT**. Alta Books, 2021. ISBN 9788550816548. Disponível em: <https://books.google.com.br/books?id=Jhk-EAAAQBAJ>.

MOUHA, R. A. Internet of things (iot). **Journal of Data Analysis and Information Processing**, Scientific Research Publishing, v. 9, n. 2, p. 77–101, 2021.

NARANG, P.; BELL, T. New iec standards and periodic testing of sound level meters. **Proceedings of the Internoise, Shanghai, China**, p. 26–29, 2008.

NIŽETIĆ, S. et al. Internet of things (iot): Opportunities, issues and challenges towards a smart and sustainable future. **Journal of Cleaner Production**, Elsevier, v. 274, p. 122877, 2020.

OLIVEIRA, S. de. **Internet das Coisas com ESP8266, Arduino e Raspberry Pi - 2ª edição**. Novatec Editora, 2021. ISBN 9786586057355. Disponível em: <https://books.google.com.br/books?id=T6AjEAAAQBAJ>.

POLITI, M. **Internet de las cosas aplicado a Generación Distribuida**. Editorial Autores de Argentina, 2022. ISBN 9789878719993. Disponível em: <https://books.google.com.br/books?id=iC1iEAAAQBAJ>.

SALIBA, T. M. **Manual prático de avaliação e controle do ruído: PPRA**. [S.l.]: LTr Editora, 2021. v. 12.

SANTOS, S. **Introdução à IoT: Desvendando a internet das Coisas**. [S.l.]: SS trader Editor, 2018.

APÊNDICE A – Código Fonte Arduino

```
#include <math.h>
#include <driver/adc.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include "time.h"



#define CHANNEL A0 //Canal ADC
#define SAMPLES 10000 //Numeros de amostras
#define SAMPLING_FREQUENCY 44100
//Frequencia de Amostragem, por Nyquist somente abaixo de 11050 Hz
#define referencia 65800
//Valor de refencia para conversao em decibeis
#define bias 1867
// constante calculada para retirar o bis do microfone

unsigned int sampling_period_us;
unsigned long microseconds;

unsigned long tempoFinal;

double sinal[SAMPLES];
//vetor de sinais analogicos
double somaDoVetor = 0;
//variavel para armazenar o calculo rms
double dBSPL = 0;
// variavel para armazenar conversao em dB

const char WIFI_SSID[] = "APE 604";
//nome da rede
const char WIFI_PASSWORD[] = "199199199";
//senha da rede
```

```
const char* ntpServer = "br.pool.ntp.org";
//endereco do servidor ntp
const long  gmtOffset_sec = -10800;
// UTC -3 * 60 * 60
const int   daylightOffset_sec = 0;
//horario de verao desligado
String horario = "";
//variavel para facilitar o envio ao bd

HTTPClient http;
String HOST_NAME = "http://192.168.1.6";
//ip do server http
String PATH_NAME   = "/script.php";
//caminho do arquivo script
String PATH_NAME2   = "/scriptHora.php";
//caminho do arquivo script
String queryString = "?valor=";
//valor a ser enviado
String queryStringFinal = "";
String dbStr = "";
String queryHora = "";

void checaLED(){
 //acende os LEDS de acordo com o nivel de ruido
  if(dBSPL < 65){
    digitalWrite(18, LOW);// DESLIGA LED AMARELO
    digitalWrite(19, LOW);// DESLIGA LED VEMRELHO
    digitalWrite(5, HIGH);//LIGA LED VERDE
    if(tempoFinal == 360){
      tempoFinal = 0;
      enviaParaBd(queryHora,PATH_NAME2);
      enviaParaBd(queryStringFinal,PATH_NAME);
    }
  }else if(dBSPL >= 65 and dBSPL < 85 ){
```

```
    tempoFinal = 0;
    digitalWrite(5, LOW);//DESLIGA LED VERDE
    digitalWrite(19, LOW);// DESLIGA LED VEMRELHO
    digitalWrite(18, HIGH);// LIGA LED AMARELO

    enviaParaBd(queryHora,PATH_NAME2);
    enviaParaBd(queryStringFinal,PATH_NAME);
  }else if(dBSPL >= 85) {
    tempoFinal = 0;
    digitalWrite(5, LOW);//DESLIGA LED VERDE
    digitalWrite(18, LOW);// DELIGA LED AMARELO
    digitalWrite(19, HIGH);// LIGA LED VEMRELHO

    enviaParaBd(queryHora,PATH_NAME2);
    enviaParaBd(queryStringFinal,PATH_NAME);
  }
}
void getHorario(){
  time_t rawtime;
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }
  char timeStringBuff[50]; //50 chars should be enough
  strftime(timeStringBuff, sizeof(timeStringBuff),
  "%A, %B %d %Y %H:%M:%S", &timeinfo);
  horario = String(timeStringBuff);
}

void enviaParaBd(String var, String local){
  http.begin(HOST_NAME + local + var); //HTTP
  int httpCode = http.GET();
  if(httpCode > 0) {
```

```
    // Arquivo encontrado no server
    if(httpCode == HTTP_CODE_OK) {
      String payload = http.getString();
      Serial.println(payload);
    } else {
      Serial.printf("[HTTP] GET... code: %d\n", httpCode);
    }
  } else {
    Serial.printf("[HTTP] GET... failed, error: %s\n",
    http.errorToString(httpCode).c_str());
  }
 http.end();
}

void setup() {
  sampling_period_us = round(1000000* (1.0 / SAMPLING_FREQUENCY));
  Serial.begin(115200);
  adc1_config_width(ADC_WIDTH_BIT_12);
  //configura o ADC para 12BITS (default = 10)
  adc1_config_channel_atten(ADC1_CHANNEL_0,ADC_ATTEN_DB_0);

  pinMode(5, OUTPUT); //configura o pino 5 como saida
  pinMode(18, OUTPUT); //configura o pino 18 como saida
  pinMode(19, OUTPUT); //configura o pino 19 como saida

  //conecta o esp a rede
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD); //Conecta ao WIFI
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
```

```
  Serial.println(WiFi.localIP());

  //init and get the time
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}

void loop() {
  microseconds = micros();
  somaDoVetor=0;
  for (int i = 0; i < SAMPLES; i++){
       sinal[i] = adc1_get_raw(ADC1_CHANNEL_0);
       // guarda valor da pressao sonora no vetor
  while (micros() - microseconds < sampling_period_us) {
      //loop vazio
   }
    microseconds += sampling_period_us;
  }

  //Gera valor RMS do sinal
  for (int i = 0; i < SAMPLES; i++){
        somaDoVetor += sqrt(pow(sinal[i]-bias,2)/SAMPLES);
  }

  dBSPL = 20 * log10(somaDoVetor / referencia) + 94;
  //calcula pressao sonora em dB

  getHorario();
  queryHora = horario;
  Serial.print(horario + " dBSPL: ");
  Serial.println(dBSPL);
  Serial.println(tempoFinal);

  dbStr = String(dBSPL,2);
  queryStringFinal = queryString + dbStr;
```

```
  checaLED();
  tempoFinal++;
}
```