



Estudo e desenvolvimento de sistema de detecção de som para identificação de focos de poluição sonora na cidade do Recife

Trabalho de Conclusão de Curso

Engenharia de Computação

THIAGO OLIVEIRA RIBEIRO

Orientador: Prof. Sérgio Campello Oliveira



Thiago Oliveira Ribeiro

**Estudo e desenvolvimento de sistema de detecção de
som para identificação de focos de poluição sonora na
cidade do Recife**

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Engenharia de Computação
Escola Politécnica de Pernambuco
Universidade de Pernambuco

Orientador: Prof. Sérgio Campello Oliveira

Recife - PE, Brasil
novembro de 2015

Thiago Oliveira Ribeiro

Estudo e desenvolvimento de sistema de detecção de som para identificação de focos de poluição sonora na cidade do Recife/ Thiago Oliveira Ribeiro. – Recife - PE, Brasil, novembro de 2015-

49 p.

Orientador: Prof. Sérgio Campello Oliveira

Trabalho de Conclusão de Curso – Engenharia de Computação

Escola Politécnica de Pernambuco

Universidade de Pernambuco, novembro de 2015.

1. Som. 2. Ruído. 3. Recife. 4. Detecção. 5. Monitoramento. 5. Internet. I. Prof. Sérgio Campello Oliveira. II. Universidade de Pernambuco. III. Escola Politécnica. IV. Título

Este trabalho é dedicado à minha família.

Agradecimentos

Em primeiro lugar gostaria de agradecer a Deus, o Autor e Mantenedor de todas as coisas, No qual baseio minha fé, por ter sido meu refúgio indispensável nos mais delicados momentos que se passaram desde o início da minha graduação até aqui.

Em segundo, à minha família: meus pais Enoque e Luciana e minha irmã Sara, por sempre acreditarem em mim e também por todo o incentivo e suporte em todas as horas.

À minha namorada Mari, por toda paciência, carinho e amor demonstrados durante todo esse tempo que se passou desde o conceito deste projeto até sua execução final.

Em especial gostaria de agradecer aos que chamo de *guerreiros*. Aqueles que comigo viraram noites estudando, dedicaram tudo de si e fizeram parte da minha vida durante todos esses anos. Da primeira derivada e do primeiro laço de repetição até aqui, vocês foram peça chave nessa minha conquista. São eles meus amigos de UNIVASF: Bruno Sampaio, Washington Lacerda, José Matias, Eldon Costa, Danilo Macário, Ivonaldo Faustino, Renato Gabriel e Raí Tamarindo. Aos amigos do pensionato Lar do Estudante Daniel Pinheiro e José Elias, os quais acompanharam de perto algumas das etapas que passei longe de casa para chegar até aqui. Também de lá, diretamente do sertão, vai o meu agradecimento especial a um professor que marcou parte da minha graduação com sua maestria e paixão visíveis pela Matemática e pela Computação, Edson Araújo.

Aos meus grandes amigos dos tempos de *Oklahoma Christian University*, que fizeram do meu tempo nos Estados Unidos, um dos melhores da minha vida: Gustavo Moreira, Laércio Vitorino e Allan Amorim.

Aos grandes amigos que fiz e levarei para vida toda, os quais fizeram da minha experiência na POLI algo muito melhor do que eu imaginei: Hartur Brito, Gemerson Gerardo, Felipe Jorge, Felipe Morais, Helson Cruz, Jhonatan Santos e Ivaldo Camelo.

Aos meus amigos de Nerval: Lucas Nunes, Rafael Soares e Gildson Luiz.

A todos os professores dedicados e comprometidos com os quais que tive a oportunidade de aprender, em especial aos do Ensino Fundamental e Médio, que muitas vezes não são lembrados, mas que para mim foram também peças indispensáveis na minha decisão em me tornar um Engenheiro de Computação.

Ao meu professor orientador Sérgio Campello, por ter me passado parte do seu conhecimento em Eletrônica, pela gentileza e acompanhamento durante as etapas deste trabalho, o meu muito obrigado.

*“Truth is ever to be found
in simplicity, and not
in the multiplicity and
confusion of things.”*

Isaac Newton

Resumo

Recentemente, o aumento da poluição sonora no mundo e principalmente em áreas urbanas, tem sido encarado como um dos maiores problemas da sociedade moderna. Não menos danosa à saúde humana que outros tipos de poluição, a poluição sonora tem sido reconhecida como uma das maiores perturbações que afetam a qualidade de vida nas grandes cidades. Neste sentido, diversos órgãos de estado tem movido ações para combater a emissão de ruídos por parte de estabelecimentos, residências e centros com grande concentração de pessoas. O processo de controle desses níveis de som funciona a partir do envio de fiscais munidos de Medidores de Nível de Pressão Sonora para identificar locais infratores que estejam emitindo ruídos acima do permitido para determinados horários do dia. Por tratar-se de um processo demorado e sem possibilidade de monitoramento constante, o *hardware* desenvolvido neste projeto trata-se de um medidor de baixo custo montado em uma placa de prototipação, e um *software (mobile)* monitorador capaz de verificar em tempo real os níveis emitidos por locais da cidade do Recife, de modo a mapear e controlar a poluição sonora remotamente. O sistema comportou-se de maneira esperada ao ser capaz de medir valores em decibéis e enviá-los a um servidor *web*, onde tais dados são exibidos em um *software* desenvolvido para sistemas operacionais iOS.

Palavras-chave: Som. Detecção. Ruído. Recife. Monitoramento. Arduino. WiFi.

Abstract

Recently, the increase in noise pollution in the world and especially in urban areas, has been seen as one of the biggest problems of modern society. No less harmful to human health than other types of pollution, noise pollution has been recognized as one of the largest disturbances affecting the quality of life in big cities. In this regard, several state agencies have moved actions to combat the emission of noise by schools, residences and centers with large concentrations of people. The control process of these sound levels operates from sending officers using Sound Pressure Level Meters to identify offenders locations that are emitting above the allowed noise for certain times of day. As this is a long process without the possibility of constant monitoring, the project proposed aims to develop a low-cost hardware meter, and a mobile software tracker to be able to check real-time output levels by certain locations of Recife, in order to map and control noise pollution. The whole system worked as expected by being able to measure values in decibels and send them to a web server, from where they could be read and displayed in software (mobile app) developed for the iOS operating system.

Keywords: Sound. Noise. Detector. Monitoring. Arduino. WiFi.

Lista de ilustrações

Figura 1 – Exemplo de onda transversal em uma corda.	17
Figura 2 – Exemplo de onda longitudinal em um fluido.	17
Figura 3 – Diapasão: um objeto simples capaz de criar sons.	18
Figura 4 – Ilustração de uma onda sonora, ou onda de pressão.	19
Figura 5 – Exemplo de uma onda senoidal.	19
Figura 6 – Os sons que ouvimos são geralmente uma mistura de frequências. . . .	20
Figura 7 – Dois sons podem ter a mesma frequência e diferentes amplitudes. . . .	20
Figura 8 – Função $y = \log_{10}x$ e alguns valores de demonstração.	22
Figura 9 – A conviência da Escala Decibel.	22
Figura 10 – Medidor de Nível de Pressão Sonora Comum (ou <i>Decibelímetro</i>). . . .	23
Figura 11 – Diagrama de blocos de um Medidor de Nível de Pressão Sonora Básico. .	24
Figura 12 – Curvas de Fletcher–Munson: relação frequência-amplitude do ouvido. .	25
Figura 13 – Filtro de ponderação A.	25
Figura 14 – Medidor de Nível de Pressão Sonora Integrador.	26
Figura 15 – Diagrama de blocos ilustrando o <i>hardware</i> do medidor e seus componentes. .	28
Figura 16 – <i>Arduino UNO</i> e componentes: plataforma de desenvolvimento.	28
Figura 17 – Módulo ESP8266 Versão 1.	29
Figura 18 – <i>Chip</i> ESP8266 da <i>Espressif</i>	30
Figura 19 – <i>SparkFun Sound Detector</i> : um detector de som robusto.	31
Figura 20 – <i>Display LCD</i> comum: modo de saída de dados para o usuário.	31
Figura 21 – Segunda etapa do circuito do DDS: seguidor de envelope.	33
Figura 22 – Conexão entre <i>Arduino</i> e Detector de Som.	33
Figura 23 – Conexão entre <i>Arduino</i> , Módulo ESP8266 e Detector de Som.	34
Figura 24 – Conexão entre <i>Arduino</i> , display de LCD e demais componentes.	35
Figura 25 – Ilustração do projeto de <i>hardware</i> com todos os componentes.	35
Figura 26 – Projeto do <i>hardware</i> montado na <i>protoboard</i> medindo o som.	36
Figura 27 – Projeto do <i>hardware</i> montado na <i>protoboard</i> antes de medir: conectando- se à rede.	37
Figura 28 – iPhone: exemplo de dispositivo que roda o sistema operacional iOS. .	38
Figura 29 – Exemplo de método em <i>Objective-C</i> : muito semelhante a C.	39
Figura 30 – Telas iniciais do <i>software</i> de monitoramento.	41
Figura 31 – Tela de detalhes do local escolhido pelo usuário: dados do medidor. .	41
Figura 32 – Gráfico de medições do <i>hardware</i> × aplicativo <i>Decibels (software)</i>	43
Figura 33 – Sistema medindo 51.08 dB momentos antes de enviar.	44
Figura 34 – Gráfico com valores medidos pelo sistema em decibéis durante 100 minutos.	45

Figura 35 – *Software (iOS app)* efetuando a mesma leitura de 51.08 dB. 45

Lista de tabelas

Tabela 1 – Relação de sons comuns em termos de pressão em <i>Pa</i>	21
Tabela 2 – Tabela com valores em dB das medições do <i>hardware</i> e do aplicativo <i>Decibels</i>	42

Lista de abreviaturas e siglas

MNPS	Medidor de Nível de Pressão Sonora
DDS	Detector de Som
HTTP	<i>Hypertext Transfer Protocol</i>
SPL	<i>Sound Pressure Level</i>
IDE	<i>Integrated Development Environment</i>
API	<i>Application Programming Interface</i>
TCP/IP	<i>Transmission Control Protocol / Internet Protocol</i>

Sumário

1	INTRODUÇÃO	14
1.1	Motivação e Caracterização do Problema	14
1.2	Objetivo Geral	15
1.3	Objetivos Específicos	15
1.4	Estrutura do Documento	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Fundamentos e Conceito de Som	17
2.2	Características do Som	18
2.2.1	Frequência de um Som	19
2.2.2	Amplitude de um Som	20
2.3	A Escala Decibel	21
2.4	Como medir Níveis Sonoros	23
2.4.1	Medidor de Nível de Pressão Sonora (MNPS)	23
2.4.2	Curva de Ponderação A	24
2.4.3	MNPS Integrador	26
3	DESENVOLVIMENTO DO SISTEMA	27
3.1	Projeto do <i>Hardware</i>	27
3.2	Componentes Utilizados	27
3.2.1	O Arduino	27
3.2.2	Módulo ESP8266	29
3.2.3	Detector de Som	30
3.2.4	Display LCD	31
3.2.5	LEDs	31
3.3	Montagem dos componentes	32
3.3.1	Arduino e Detector de Som	32
3.3.2	Arduino e ESP8266	34
3.3.3	Arduino e LCD	34
3.3.4	Arduino e LEDs	34
3.3.5	Programa do Microcontrolador	36
3.4	Projeto de <i>Software</i>	37
3.4.1	Sistema Operacional iOS	38
3.4.2	<i>Objective-C</i>	38
3.4.3	<i>XCode</i>	39
3.4.4	Servidor Web: Plataforma <i>ThingSpeak</i>	39

3.5	Funcionamento do <i>Software</i> de Monitoramento	40
4	RESULTADOS E TESTES	42
4.1	Medições do <i>Hardware</i>	42
4.2	Processo de Atualização e Leitura na <i>ThingSpeak</i>	43
4.2.1	Escrita dos valores via <i>hardware</i>	43
4.2.2	Leitura dos valores via <i>software</i>	45
5	CONCLUSÕES E TRABALHOS FUTUROS	47
	Referências	48

1 Introdução

Este trabalho de conclusão de curso propõe-se ao estudo e desenvolvimento de um projeto de sistema computacional (de *hardware* e *software*), que seja capaz de medir e monitorar a intensidade de ruído em certos pontos da cidade do Recife.

O crescente número de queixas de barulho excessivo por parte da população, e os diversos danos à saúde adventos deste tipo de poluição, foram alguns dos fatores que desencadearam a escolha deste tema.

Este capítulo descreve a introdução da monografia, e está organizado em 3 seções. Na Seção 1.1 são descritas tanto a motivação para a execução deste trabalho, quanto a definição do problema.

Posteriormente, nas Seções 1.2 e 1.3 são apresentados os objetivos gerais e específicos, bem como a proposta de solução do projeto. Por fim, a Seção 1.4 detalha a organização da monografia.

1.1 Motivação e Caracterização do Problema

Atualmente, a humanidade tem experimentado um de seus maiores males concebidos por si própria: a poluição sonora. Especialmente em áreas urbanas, essa forma de poluição, não menos preocupante que outras como atmosférica ou ambiental, tem sido devidamente reconhecida como uma das maiores perturbações que afetam a qualidade de vida nas grandes cidades [1].

O conceito de ruído geralmente refere-se a sons não desejados que possam afetar negativamente a vida humana ou animal [2].

De fato, em um típico ambiente urbano, rodeado de construções, avenidas movimentadas, bares e restaurantes, a população vivendo nesses centros encontra-se inevitavelmente exposta a uma variedade desses sons, que além de interferir no seu bem-estar cotidiano, pode vir a trazer graves consequências à saúde, sendo uma das mais preocupantes a Perda Auditiva Induzida por Ruído (PAIR) [3].

Segundo guia da *World Health Organization* (WHO), agência da Organização das Nações Unidas (ONU) incumbida de assuntos ligados à saúde pública internacional, já em 1999 estimava que mais de 120 milhões de pessoas ao redor do mundo sofriam de PAIR devido a poluição sonora nos centros urbanos [4].

No Brasil, o problema já é considerado um grave caso de saúde pública, contribuindo no aumento de depressão e outras graves doenças [5].

Sendo assim, pensando na redução e combate ao barulho excessivo, a Prefeitura da Cidade do Recife (PCR) junto ao Ministério Público de Pernambuco, secretarias e órgãos do Governo do Estado, firmaram uma parceria formalizando a responsabilidade de cada uma das instituições na fiscalização de estabelecimentos de serviços, lojas e bares da capital pernambucana [6].

Também, a fim de alertar sobre os prejuízos do excesso de ruído, foi atribuída a PCR, dentre outras definições, a de realizar campanhas educativas sobre o tema em estabelecimentos e escolas municipais e privadas com o intuito de multiplicar a informação [6].

Portanto, este trabalho visa mapear e controlar os altos níveis de ruído encontrados nos principais pontos da cidade do Recife, a partir do desenvolvimento de um sistema que monitore o som do ambiente no qual estiver instalado, e exiba estes dados de maneira autônoma, para que os órgãos competentes instruam o local emissor a amenizar o barulho no mesmo momento em que for detectado o excesso, a partir da medição viabilizada pelo sistema.

1.2 Objetivo Geral

Este trabalho tem como objetivo principal o estudo e implementação de um sistema computacional a partir de um projeto de *hardware*, capaz de medir a intensidade do barulho gerado em certos locais da cidade do Recife, e de uma aplicação em *software* que viabilize o acesso dos dados em qualquer lugar, identificando assim os locais emissores dos mais altos níveis de ruído.

1.3 Objetivos Específicos

A implementação do projeto também envolveu outros fins, os quais foram atingidos:

- Estudo sobre a natureza do som e sua medição;
- Proposta e desenvolvimento de um medidor de baixo custo em *hardware*;
- Desenvolvimento de aplicação em *software* para visualizar informações obtidas pelo medidor;
- Realização de testes do sistema em certo local na cidade do Recife.

1.4 Estrutura do Documento

Este trabalho está dividido em 5 capítulos, incluindo este que conta com uma introdução a respeito do tema e objetivos do projeto. Em seguida o Capítulo 2 traz um estudo a respeito das propriedades do som, meios de medi-lo, e demais conceitos necessários para a compreensão do sistema desenvolvido neste trabalho.

Já no Capítulo 3, é descrito as etapas do processo que resultou no sistema proposto. O Capítulo 4 apresenta os resultados obtidos e dados de testes efetuados com o sistema em funcionamento.

Finalmente, o Capítulo 5 encerra o trabalho com uma discussão a respeito do que foi desenvolvido, e possíveis melhoramentos e novas ideias para o projeto.

2 Fundamentação Teórica

Este capítulo busca apresentar alguns conceitos fundamentais a respeito do som e suas características, bem como explicitar meios de medir sua intensidade.

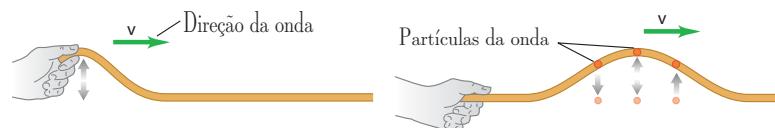
Também é detalhada a teoria que fundamenta os componentes e ferramentas utilizadas tanto no projeto do medidor em *hardware*, quanto do *software* de monitoração dos dados.

2.1 Fundamentos e Conceito de Som

De todas as formas de ondas mecânicas encontradas na natureza, uma das mais importantes são as *ondas sonoras*. É possível categorizar ondas com base no movimento das partículas no meio em que a onda se propaga [7]. Os dois tipos mais notáveis desta categoria são:

- **Ondas Transversais:** são aquelas em que a direção de vibração das partículas no meio, é *perpendicular* à direção de propagação da onda.

Figura 1 – Exemplo de onda transversal em uma corda.



Fonte: extraído de [8].

- **Ondas Longitudinais:** são aquelas em que a direção de vibração das partículas no meio, é *paralela* à direção de propagação da onda.

Figura 2 – Exemplo de onda longitudinal em um fluido.



Fonte: extraído de [8].

Segundo a definição mais geral, o **som** é uma onda longitudinal em um meio. Embora capaz de propagar-se através de meio sólido, líquido ou gasoso, o *ar* é o meio mais usual para se estudar conceitos sobre o som [8].

Todos os sons que escutamos, são vibrações se propagando através do ar, onde um objeto vibrador comprime as moléculas de ar (comprimindo-as juntas) e depois as espalha (lançando-as a uma certa distância). Mesmo com as flutuações na pressão, as moléculas de ar tendem a se manter na mesma posição, em média [9].

Essas rápidas e súbitas mudanças na pressão do ar, fazem vibrar o tímpano, e assim sinais nervosos são enviados ao cérebro e interpretados como som [9].

Figura 3 – Diapasão: um objeto simples capaz de criar sons.



Fonte: extraído de [10].

Um exemplo de objeto simples capaz de criar som e demonstrar seu conceito, é o *diapasão*. A medida que as pontas vibram para frente e para trás, elas empurram as partículas de ar ao seu redor.

O movimento das pontas para frente, empurram as moléculas de ar para a direita, e o recuo cria uma área de menor pressão, permitindo que as partículas sejam movidas de volta para a esquerda.

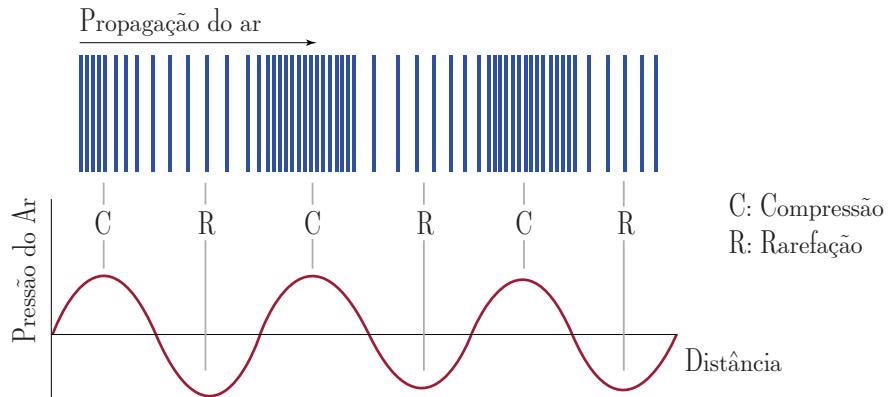
É importante ressaltar que o som *não é* uma onda *transversal* e sim *longitudinal*, criada pelas compressões e rarefações de ar.

Como se pode ver na Figura 4, utilizando-se o diapasão como exemplo, podemos mostrar uma *representação do som por uma onda transversal senoidal*, com o intuito de ilustrar a natureza senoidal das flutuações de pressão no tempo.

2.2 Características do Som

Dentre propriedades que caracterizam uma onda sonora, as duas consideradas mais importantes são: *frequência* e *amplitude*.

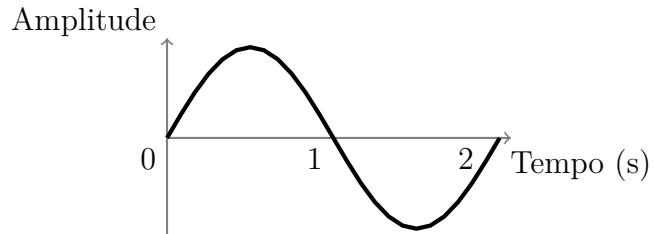
Figura 4 – Ilustração de uma onda sonora, ou onda de pressão.



Fonte: extraído de [11].

Uma das formas mais simples de onda é a *onda senoidal*. Embora ondas puramente senoidais sejam raramente encontradas na natureza, uma simples onda senoidal demonstra claramente estes conceitos, uma vez que todos os outros sons podem ser considerados como combinações de ondas senoidais [9].

Figura 5 – Exemplo de uma onda senoidal.



Fonte: extraído de [9].

2.2.1 Frequência de um Som

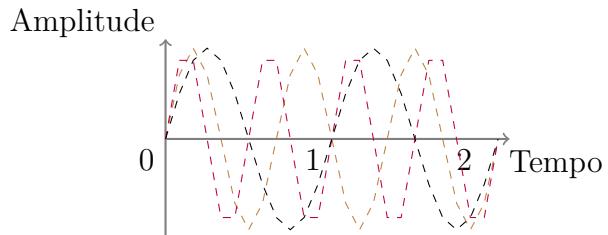
Como mencionado anteriormente, uma onda sonora é definida como uma onda de pressão que se propaga por um meio. Ao propagar-se pelo ar, a pressão atmosférica varia periodicamente.

Logo, a *frequência* de um som é definida como o número de variações de pressão por segundo, e é medida em *Hertz* (Hz), ou número de ciclos por segundo.

O som que escutamos é geralmente radiado em todas as direções, a partir de um meio vibrante, e é uma combinação de muitas frequências diferentes como se pode ver na

Figura 6.

Figura 6 – Os sons que ouvimos são geralmente uma mistura de frequências.



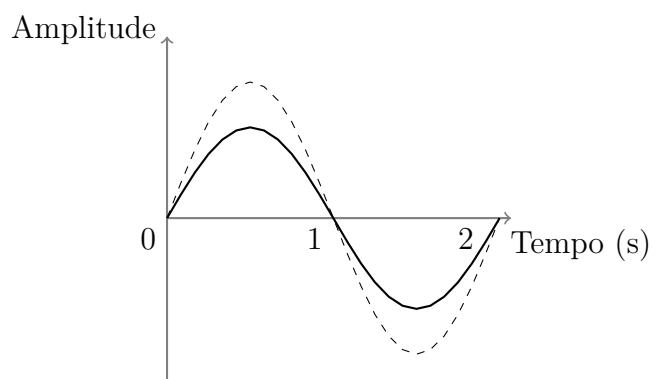
Fonte: extraído de [12].

Um ser humano jovem e saudável consegue escutar uma faixa de frequência de 20 Hz a 20.000 Hz. Na meia idade, esta faixa muda para 70 a 14.000 Hz. Um instrumento como o piano, tem uma faixa de 31.5 Hz a 8.000 Hz [12]. Ou seja, o ser humano consegue escutar uma enorme variedade de sons.

2.2.2 Amplitude de um Som

Uma outra propriedade do som é a sua amplitude. Quanto maior for a amplitude da pressão, maior a altura percebida do som [7]. Sons podem possuir amplitudes diferentes, mesmo se possuírem a mesma frequência como mostra a Figura 7.

Figura 7 – Dois sons podem ter a mesma frequência e diferentes amplitudes.



Fonte: extraído de [12].

Um som de maior amplitude possui uma maior variação de pressão do que um som de menor amplitude. Define-se a **pressão** p em um certo ponto, como sendo a força normal por unidade de área, ou seja, a razão entre dF e dA [8]:

$$p = \frac{dF}{dA} \quad (2.1)$$

Caso a pressão seja a mesma em todos os pontos de uma superfície plana finita com área A , então:

$$p = \frac{F}{A} \quad (2.2)$$

onde F é a força normal em um lado da superfície.

A unidade utilizada para expressar pressão e variação de pressão é o *Pascal* (Pa), em homenagem ao cientista e filósofo francês Blaise Pascal.

$$1 \text{ pascal} = 1 \text{ Pa} = 1 \text{ N/m}^2 \quad (2.3)$$

O ouvido humano consegue perceber uma faixa muito ampla de pressões sonoras, sendo $20 \mu\text{Pa}$ ($20 \cdot 10^{-6} \text{ Pa}$) o valor de pressão do som mais suave que um humano pode detectar. Este valor é chamado de **limiar da audição**.

Em algumas situações, como no caso de um lançamento de uma nave espacial, são produzidos valores de pressão sonora de aproximadamente $200.000.000 \mu\text{Pa}$.

Tabela 1 – Relação de sons comuns em termos de pressão em Pa .

Fonte do Som	Pressão do Som (em μPa)
Lançamento de nave espacial	200.000.000
Orquestra Completa	20.000.000
Trem a <i>diesel</i> em alta velocidade a 25 metros	200.000
Conversa normal	20.000
Cochicho suave a 2 metros	2.000
Sala de transmissão vazia	200
Limiar da audição humana	20

Como se pode ver na Tabela 1, lidar com números que variam tanto quanto os valores de pressão sonora em Pa , é uma tarefa incoveniente (de 20 a 2.000.000.000).

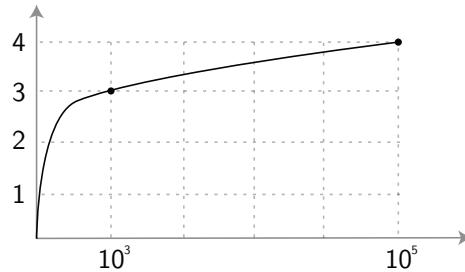
Uma maneira mais simples é utilizar a *Escala Logarítmica*, que é a base para a *Escala Decibel*, discutida na Seção 2.3.

2.3 A Escala Decibel

Como visto na seção acima, a aplicação direta de uma escala linear para medir pressão sonora, em Pa , levaria ao uso de números enormes. No entanto, utilizando-se de

uma escala logarítmica, podemos representar tais valores altos em menores, tornando a manipulação mais simples como vemos na Figura 8.

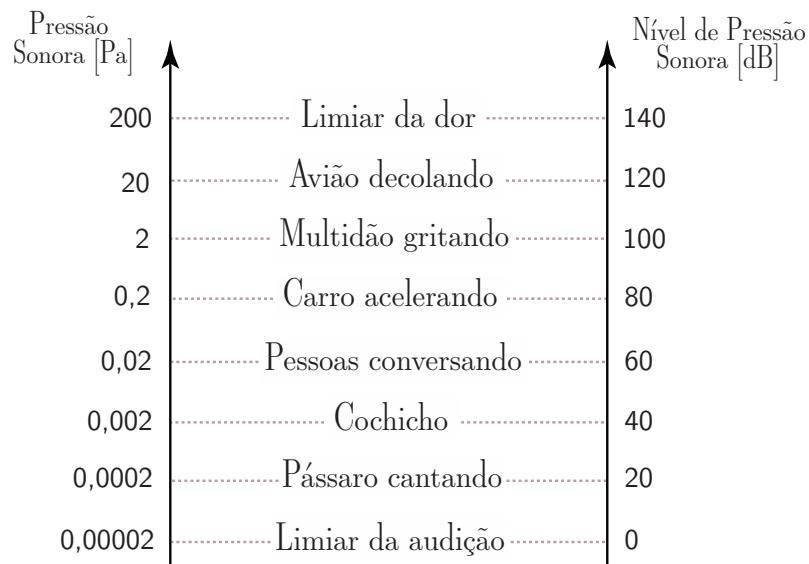
Figura 8 – Função $y = \log_{10}x$ e alguns valores de demonstração.



Fonte: extraído de [13].

Além do mais, nossos ouvidos não respondem a estímulos de maneira linear, e sim logarítmica. Logo, percebe-se que é mais prático expressar valores de pressão sonora a partir de uma proporção logarítmica de um valor medido com um valor de referência. Esta proporção logarítmica é chamada de **decibel** ou **dB**, em homenagem ao trabalho de Alexander Graham Bell, o inventor do telefone [8].

Figura 9 – A conviência da Escala Decibel.



Fonte: extraído de [14].

É importante frisar que o decibel *não* é uma unidade para medir som, e sim uma medida *relativa* [15]. Em termos de nível de pressão sonora em decibéis, tem-se o limiar

da audição como sendo 0 dB e o limiar da dor como sendo 130 dB, como se pode ver na Figura 9. Em certas situações, pode ocorrer rompimento do tímpano com valores em cerca de 190 dB.

Portanto, o valor de *Nível de Pressão Sonora* em decibéis, abreviado para *SPL* (*Sound Pressure Level*) ou L_p (*Level of pressure*), é obtido pela fórmula:

$$SPL = 20 \cdot \log_{10} \left(\frac{p}{p_0} \right) \quad (2.4)$$

sendo p (RMS) o valor medido em Pa , e p_0 (RMS) o valor de referência padrão que é de $20 \mu Pa$ – limiar da audição humana.

Segundo a *Brüel & Kjær*, empresa multinacional e maior fabricante e fornecedora de equipamentos de medição de som e vibração, embora existam outros parâmetros do som que podem ser medidos como intensidade e potência, em casos de avaliação de nocividade e incômodo de fontes sonoras (que é o caso deste trabalho), o fator importante a se considerar é a *pressão sonora* [14].

2.4 Como medir Níveis Sonoros

2.4.1 Medidor de Nível de Pressão Sonora (MNPS)

O instrumento utilizado para medir níveis de pressão sonora é chamado de *Medidor de Nível de Pressão Sonora* (MNPS), ou mais conhecido por *Decibelímetro*.

Um MNPS trata-se de um instrumento projetado para responder ao som de maneira semelhante ao ouvido humano, e reproduzir medidas de níveis de pressão sonora.

Figura 10 – Medidor de Nível de Pressão Sonora Comum (ou *Decibelímetro*).



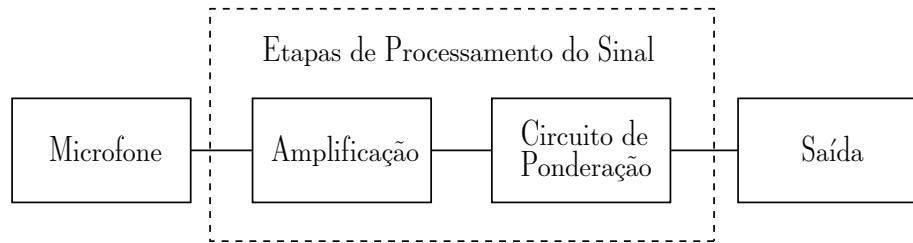
Fonte: extraído de [16].

Embora exista uma infinidade de aparelhos como este, que diferem em detalhes, todos compartilham de um sistema que consiste em um microfone, um circuito processador do sinal captado, e uma unidade de leitura.

Detalhadamente, algumas etapas que envolvem as etapas de um Medidor de Níveis de Pressão Sonora são:

1. Um microfone que é utilizado para captar o som e convertê-lo em um sinal elétrico equivalente;
2. Após a leitura e conversão do som em sinal elétrico, um circuito *pré-amplificador* entra em ação para amplificar (aumentar a amplitude) o sinal, devido ao sinal gerado pelo microfone ser de baixa amplitude.
3. Com o sinal amplificado, vários tipos de processamento podem ser aplicados, sendo o mais comum a aplicação de um *filtro de ponderação A*;
4. *Display* de saída de dados.

Figura 11 – Diagrama de blocos de um Medidor de Nível de Pressão Sonora Básico.



Fonte: o autor.

2.4.2 Curva de Ponderação A

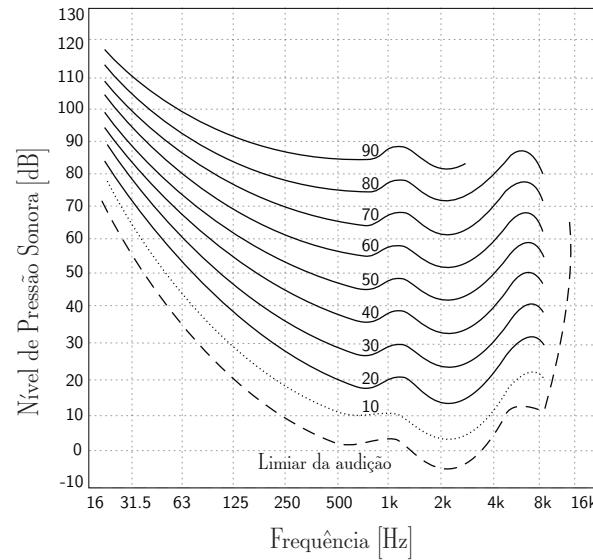
No ano de 1933, os pesquisadores Fletcher e Munson publicaram um conjunto de curvas mostrando a sensibilidade do ouvido humano a amplitude de um som, comparado a sua frequência.

Essas curvas mostravam que o ouvido humano é mais sensível a sons que possuem uma frequência entre 3000-4000 Hz. Sons com frequências abaixo ou acima desta faixa devem possuir uma amplitude maior para serem igualmente altos como se pode ver na Figura 12.

Alguns anos depois desta publicação, tais curvas foram utilizadas no primeiro padrão *American National Standards Institute* (ANSI) para medidores sonoros, o qual incorporou a curva de ponderação A e B (esta logo caiu em desuso).

A curva de ponderação A é a mais frequentemente utilizada para medição sonora pois é a que mais reflete verdadeiramente a maneira como os seres humanos escutam (atenuando somente as frequências audíveis).

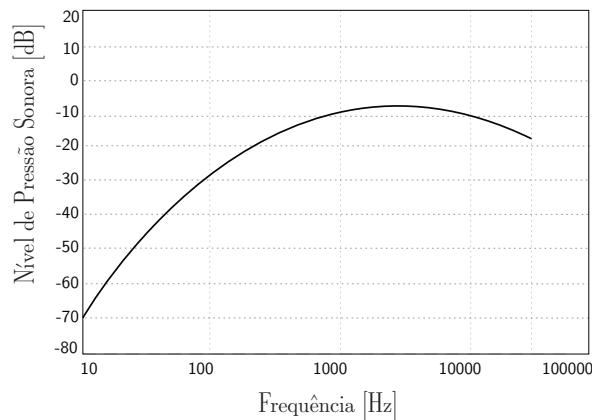
Figura 12 – Curvas de Fletcher–Munson: relação frequência-amplitude do ouvido.



Fonte: extraído de [17].

Como se pode ver na Figura 13, a Curva de Ponderação A cobre o intervalo inteiro de frequências de 10 Hz a 10kHz. No entanto, o formato dela se aproxima das frequências percebidas pelo ouvido humano.

Figura 13 – Filtro de ponderação A.



Fonte: extraído de [18].

As medidas feitas utilizando esta curva são expressas em dB(A) para mostrar que foi utilizado uma Curva de Ponderação A no processo de medição. Nesse contexto, níveis sonoros são expressos em $L_{p(A)}$.

2.4.3 MNPS Integrador

Como citado na Seção anterior, um MNPS comum é capaz de medir a amplitude de um som em determinado momento.

Porém, isto pode não ser algo muito útil quando é necessário tirar uma média dos níveis sonoros de um ambiente por um certo espaço de tempo, uma necessidade comum para fábricas, por exemplo, que necessitam monitorar a intensidade do ruído das máquinas.

Figura 14 – Medidor de Nível de Pressão Sonora Integrador.



Fonte: extraído de 10.

Para esta tarefa, o instrumento utilizado é um pouco mais sofisticado (e caro) que um MNPS comum, e se chama *Medidor de Nível de Pressão Sonora Integrador*, onde a palavra *integrador* diz respeito a soma dos valores durante um período de tempo.

Além do mais, muitos deles também possuem um *data-logger* interno, capaz de gerar gráficos e dados estatísticos a respeito dos valores de pressão sonora obtidos por determinados períodos de tempo.

3 Desenvolvimento do Sistema

Este capítulo busca explicar os passos que envolveram a criação e desenvolvimento tanto do projeto do medidor em *hardware* quanto do *software* monitorador dos valores de níveis sonoros. Na Seção 3.1, são exibidos os componentes utilizados e suas conexões. Na Seção 3.2, cada um dos componentes utilizados no projeto são detalhados e são detalhados os motivos para sua utilização.

Já na Seção 3.3, um passo-a-passo é desenvolvido, indo desde as primeiras conexões da placa até o projeto final na *protoboard*.

Ao final do capítulo, na Seção 3.4, são explicitadas as principais ferramentas usadas para o desenvolvimento do *software* para iOS, bem como telas e detalhamento a respeito de sua utilização para monitoramento remoto.

3.1 Projeto do *Hardware*

Como já mencionado anteriormente, um MNPS Integrador é uma ferramenta cara, tornando ainda mais cara a compra de vários aparelhos para monitorar constantemente os níveis de pressão sonora em ambientes.

Neste contexto, o projeto de *hardware* aqui apresentado trata-se de uma solução de baixo custo, com possibilidade de monitoramento *online* via *software*, das estações que tiverem o dispositivo instalado.

3.2 Componentes Utilizados

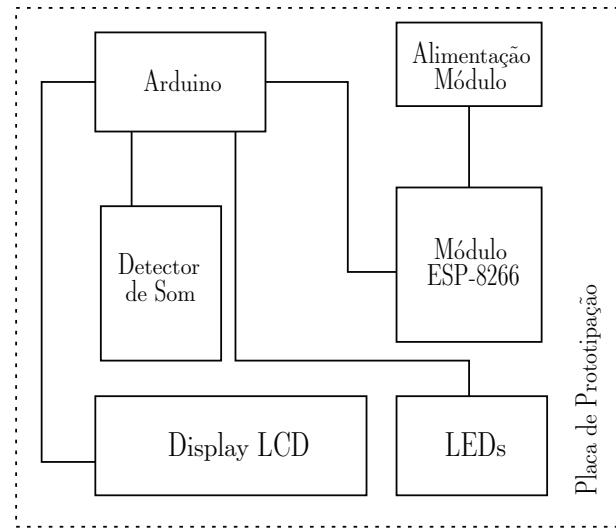
A Figura 15 ilustra as conexões entre os componentes utilizados e as próximas Seções descrevem cada um deles, detalhando suas funcionalidades e motivos que levaram a sua escolha nesta aplicação.

3.2.1 O Arduino

Arduino é um plataforma de desenvolvimento de microcontrolador com uma linguagem de programação intuitiva que pode ser utilizada com o *Arduino Integrated Development Environment* (IDE), a partir do qual programas podem ser escritos e gravados no microcontrolador embutido na placa.

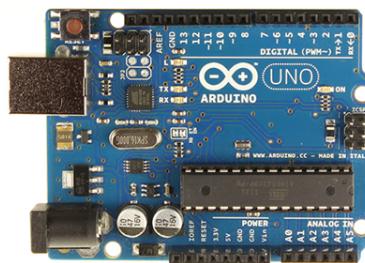
Toda a família de placas *Arduino* possui um conjunto básico de funcionalidades que incluem:

Figura 15 – Diagrama de blocos ilustrando o *hardware* do medidor e seus componentes.



Fonte: o autor.

Figura 16 – *Arduino UNO* e componentes: plataforma de desenvolvimento.



Fonte: exploring arduino.

- Microcontrolador *Atmel* ;
 - LEDs de *debug* e serial *TX/RX*;
 - Interfaces de programação/comunicação USB;
 - Pinos de Entrada/Saída;
 - Reguladores de tensão e conexões de energia;
 - Conectores *In-Circuit Serial Programmer* (ICSP);

Na Figura 16 vê-se alguns detalhes do modelo de *Arduino* utilizado para o desenvolvimento deste projeto e seus componentes nomeados.

Ao conectar sensores, atuadores, auto-falantes, módulos ou algum tipo de circuito integrado aos pinos de entrada/saída do Arduino, o mesmo pode se tornar uma poderosa

ferramenta para controlar basicamente qualquer sistema, com um custo extremamente baixo.

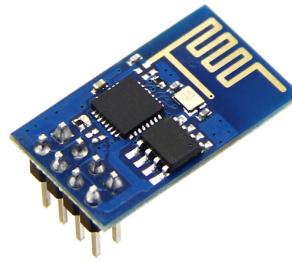
Um fato importante a mencionar é que o Arduino é um *open-source hardware*. Isto significa que esquemáticos, código fonte e arquivos de projeto estão todos disponíveis gratuitamente.

3.2.2 Módulo ESP8266

O Módulo ESP8266 trata-se de uma pequena placa projetada pela *Espressif Systems*, uma companhia Chinesa com sede em Shanghai.

Como descrito em sua documentação, o ESP8266 é uma solução de rede WiFi que pode ser utilizada como ponte com microcontroladores para adicionar a capacidade de Internet.

Figura 17 – Módulo ESP8266 Versão 1.



Fonte: extraído de [19].

Por conter um microprocessador, portas de entrada e saída, memória RAM (*Random Access Memory*) e *flash*, este poderoso módulo dispensa até mesmo a necessidade de utilizar um microcontrolador externo, podendo rodar programas de maneira independente.

Em suma, trata-se de um microprocessador alternativo que possui suporte a WiFi e TCP/IP (*Transmission Control Protocol / Internet Protocol*) embutidos, por um valor semelhante ao de um *Arduino*.

Existem uma série de estilos de placa do ESP8266. Porém, por motivos como custo, robustez do projeto, e disponibilidade, foi escolhida a primeira versão desta placa, a ESP-1.

Todas as placas contém o mesmo *chip* ESP8266 (Figura 18). O que difere-as são componentes como a quantidade de portas de entrada/saída, o estilo dos conectores, quantidade de memória *flash* disponível e assim por diante. Algumas das especificações do *chip* ESP8266 incluem:

- Protocolo 802.11 b/g/n (WiFi);

Figura 18 – *Chip ESP8266 da Espressif.*

Fonte: extraído de [19].

- Pilha de protocolo TCP/IP integrada;
- CPU de 32-bits de baixo consumo;
- Consumo de energia $< 1mW$ em modo *standby*;

Devido a necessidade de acesso *WiFi* do projeto do medidor desenvolvido neste trabalho, outras alternativas além do Módulo ESP8266 foram consideradas.

No entanto, devido a crescente comunidade de entusiastas do ESP e crescente fóruns de discussão, preço baixo da versão 1, tamanho do módulo, dentre outros, foi decidido utilizá-lo e integrá-lo ao *Arduino UNO*, que passou a contar com a capacidade de Internet a partir disso.

Assim, tornaria-se possível utilizar os dados do medidor processados no *Arduino*, e enviá-los para a Internet, onde o *software* desenvolvido também neste trabalho, poderia então acessá-los.

Os detalhes da montagem e funcionamento são descritos na Seção 3.3.

3.2.3 Detector de Som

O detector de som *SparkFun Sound Detector* é uma pequena placa produzida pela *SparkFun Electronics*, uma revendedora de componentes eletrônicos situada em *Niwot, Colorado*, nos Estados Unidos. Trata-se de uma placa que combina um microfone e alguns circuitos de processamento.

Em suas saídas, como se pode ver na Figura 19, encontramos uma indicação binária da presença de som (GATE), sendo nível ALTO para som e nível BAIXO para silêncio, uma saída de áudio (AUDIO), e também uma representação analógica da amplitude do som (ENVELOPE). Para ligá-la, basta alimentar VCC e GND.

O valor relativamente baixo, facilidade de configuração, e principalmente a saída que representa a amplitude sonora, foram os fatores determinantes na escolha deste circuito integrado para o projeto do medidor.

Figura 19 – *SparkFun Sound Detector*: um detector de som robusto.



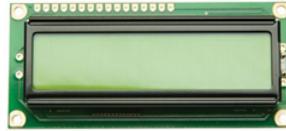
Fonte: extraído de [20].

3.2.4 Display LCD

Como dispositivo de saída do *hardware*, foi utilizado um *Liquid Crystal Display* (LCD) comum semelhante ao da Figura 20, de 16 colunas e 2 linhas.

Este *display* utiliza 16 pinos de conexão que incluem sinais de dados, controle e alimentação.

Figura 20 – *Display LCD* comum: modo de saída de dados para o usuário.



Fonte: extraído de [21].

Apesar da possibilidade de monitoramento via *software*, entendeu-se que seria interessante que os locais possuindo o *hardware* instalado e em funcionamento, pudessem ter uma ideia do *status* da medição do som instantaneamente.

Assim, decidiu-se por incorporar tal *display* para incrementar o *hardware* desenvolvido.

3.2.5 LEDs

Por fim, embora o *status* de monitoramento possa ser lido diretamente do *display*, ou acessado via *software*, existem algumas situações em que pode ser necessário verificar o *status* somente olhando para o *hardware* do medidor.

Por este motivo, também foram utilizados os bastantes conhecidos diodos emissores de luz, *Light Emitting Diode* (LED), em três diferentes cores para indicar a intensidade do som de maneira intuitiva.

Um LED verde acesso indica uma intensidade de som *baixa*, um LED amarelo indica intensidade *média*, e um LED vermelho indica intensidade *alta*.

3.3 Montagem dos componentes

De posse dos componentes mencionados na Seção anterior, o circuito foi montado numa *protoboard* MSB-500 da fabricante nacional ICEL, seguindo o diagrama de blocos da Figura 15.

Para facilitar a explicação das conexões feitas na *protoboard*, decidiu-se separar as ligações em etapas que serão descritas a seguir e só ao final será apresentada a solução completa.

3.3.1 Arduino e Detector de Som

Como mencionado anteriormente, a saída **ENVELOPE** entrega um valor de tensão DC que corresponde ao valor de pico do sinal de entrada.

O sinal de entrada captado pelo microfone de eletreto possui uma tensão muito pequena, por isso a primeira etapa do circuito é responsável por amplificá-lo com um ganho aritmético de 100. Inclusive, a placa possui uma entrada para soldar um resistor (ou potenciômetro), caso seja necessário aumentar ou diminuir o ganho do amplificador.

O circuito que entrega o valor de saída do **ENVELOPE** trata-se de um circuito *Retificador de Precisão de Meia Onda* utilizando amplificador operacional. Assim, o amplificador inverte e amplifica tal sinal V_{in} (Figura 21).

Como se pode ver na Figura 22, a única porta de saída do Detector de Som (DDS) que foi utilizada foi a **ENVELOPE**, conectada diretamente a porta de entrada analógica A0 do *Arduino*. Segundo o *datasheet* do DDS, a alimentação ideal é 5V.

O sinal V_{in} é então amplificado e invertido. Quando a tensão de entrada é maior que zero, D_2 não está conduzindo e D_1 está, então a tensão de saída V_{out} é zero pois um lado do R_7 está conectado ao terra virtual e não há corrente passando naquele ponto.

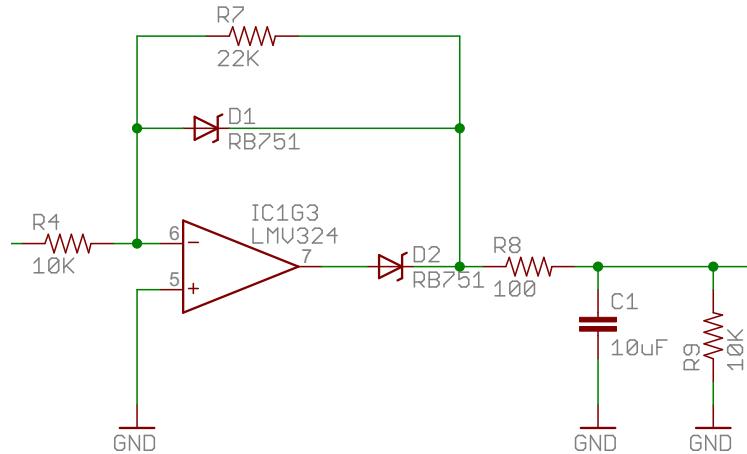
Quando V_{in} é menor que zero, D_2 está conduzindo e D_1 está como circuito aberto, e a saída é amplificada com um ganho de $\frac{-R_7}{R_4}$.

Por fim, o capacitor C_1 se carrega com o valor de tensão de pico, sendo então o responsável por armazenar o maior valor de tensão do sinal de entrada. Também, C_1 se descarrega através de R_9 , nos ciclos em que D_2 não está conduzindo.

Esta etapa do circuito implementa a equação:

Onde $-2,2$ é o ganho do amplificador dado por $\frac{-R_7}{R_1} = \frac{22k\Omega}{10k\Omega} = -2,2$.

Figura 21 – Segunda etapa do circuito do DDS: seguidor de envelope.



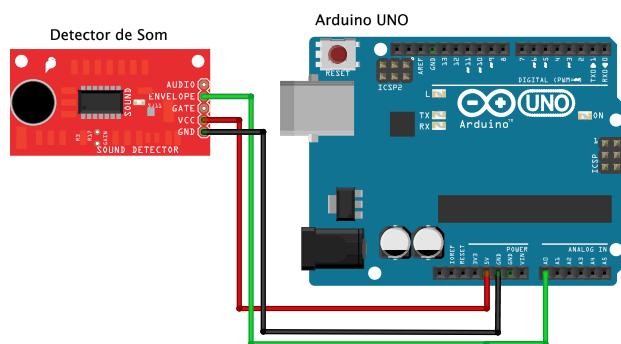
Fonte: extraído de [20].

```

if (Vin > 0)
    Vout = 0;
else
    Vout = Vin * -2,2

```

Portanto, ao efetuar as conexões do DDS com o *Arduino* como visto na Figura 22, é possível obter o valor de tensão de pico do som.

Figura 22 – Conexão entre *Arduino* e Detector de Som.

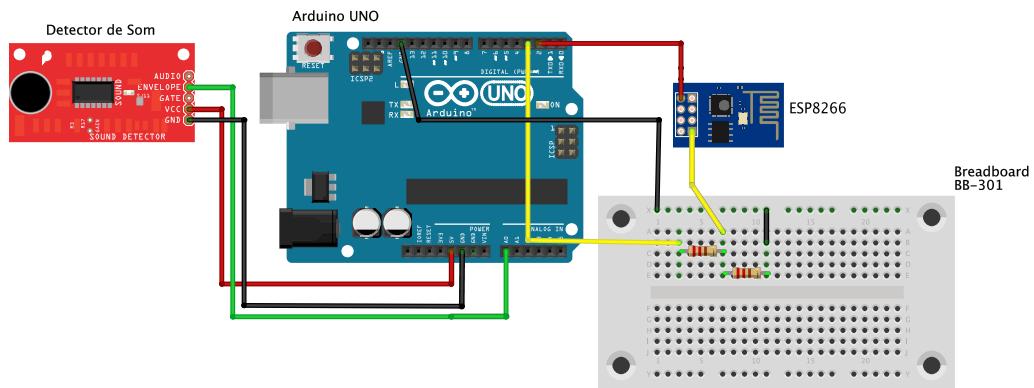
Fonte: o autor.

O *Arduino UNO* possui um conversor analógico digital de 10 bits de resolução, o que significa que é possível representar valores de 0 a 1023 ($2^{10} = 1024$). Isto significa que, para obter o valor de tensão de pico, é necessário fazer a conversão $\frac{(V_{cc} \cdot A_0)}{1024}$.

3.3.2 Arduino e ESP8266

Para se comunicar com o Módulo ESP8266, foi utilizada a biblioteca `SoftwareSerial`, que torna possível a utilização de portas comuns do Arduino para comunicação serial com outros dispositivos.

Figura 23 – Conexão entre *Arduino*, Módulo ESP8266 e Detector de Som.



Fonte: o autor.

Um outro detalhe importante é que as portas do ESP8266 funcionam em 3,3V. Sendo assim, para a conexão do módulo ao *Arduino*, foi utilizado um divisor de tensão (reduzindo o valor de tensão) para conectar a porta 3 (cuja saída é 5V) do *Arduino* à porta *RX* do ESP.

Como se pode ver na Figura 23, as portas 2 e 3 foram utilizadas para fazer a comunicação serial entre o *Arduino* e o Módulo (*Arduino* 2 como *RX* \Rightarrow *TX* do Módulo, *Arduino* 3 como *TX* \Rightarrow *RX* do Módulo). Esta configuração já permite fazer criar um programa para rodar no *Arduino* que efetue a leitura de um valor de amplitude sonora com o DDS, e envie para um servidor *web* utilizando o ESP8266.

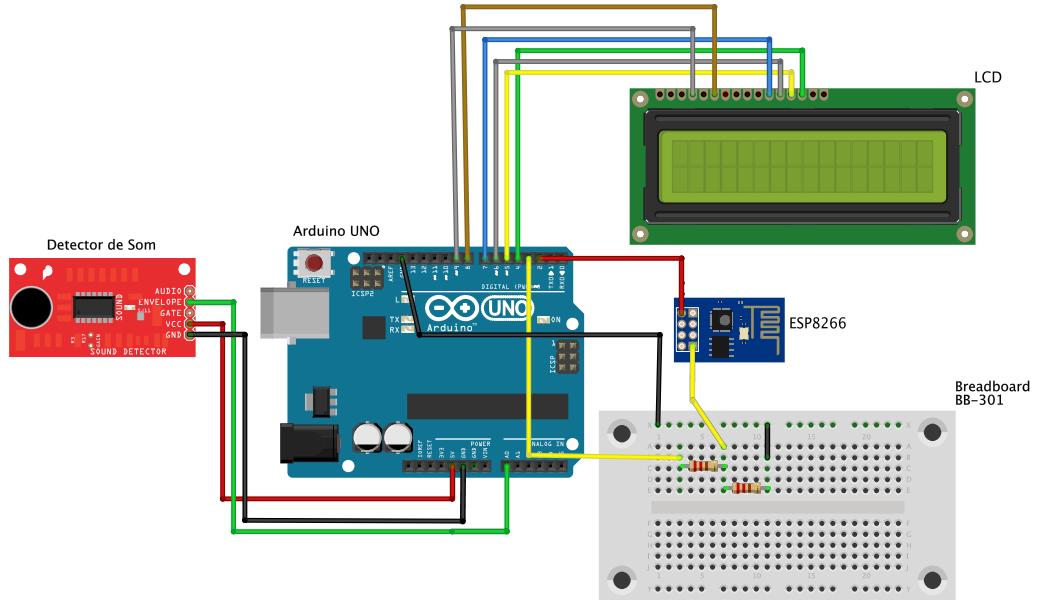
3.3.3 Arduino e LCD

As portas de 4 a 7 do *Arduino* foram conectadas aos pinos de dados do LCD. Já as portas 8 e 9 foram conectadas ao LCD para que fosse possível a leitura e escrita de dados.

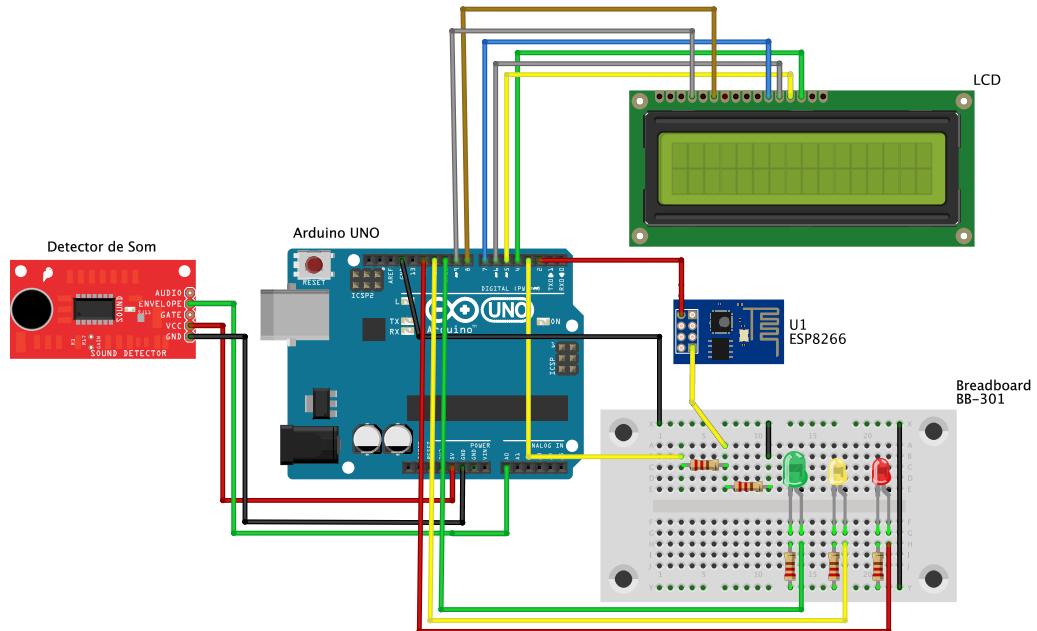
Os demais ajustes, incluindo *backlight* e contraste, bem como alimentação e terra, pode ser vistos no *hardware* final. O brilho do *backlight* foi ajustado utilizando um *trimpot* de $10k\Omega$, e o contraste, um resistor de $1k\Omega$.

3.3.4 Arduino e LEDs

A última parte do projeto do *hardware*, como mencionado previamente, utiliza três LEDs nas cores verde, amarela e vermelha, para indicar a amplitude do som medido.

Figura 24 – Conexão entre *Arduino*, display de LCD e demais componentes.

Fonte: o autor.

Figura 25 – Ilustração do projeto de *hardware* com todos os componentes.

Fonte: o autor.

Para isto, foram utilizadas as portas 10, 11 e 12 do *Arduino* para acender o LED que corresponde a amplitude sonora do local (baixa, alta ou moderada). A Figura 25 ilustra o projeto que foi montado na *protoboard*.

3.3.5 Programa do Microcontrolador

O programa gravado no *Arduino* possui basicamente três partes (funções) que são:

1. `lcdSetup()`: configura inicialmente o LCD para escrever as informações pro usuário;
2. `restartESP()`: configura o ESP para operar no modo estação, se conectando a uma rede WiFi;
3. `calculateDecibels()`: Leitura do valor analógico em A0, cálculos para encontrar o valor em dB e envio de dados para um servidor *web*.

No programa escrito em Linguagem C, as diretivas de pré-processamento `WLAN_SSID` e `WLAN_PASS` definem a rede a qual o ESP irá se conectar.

Assim que o *hardware* é ligado, uma das primeiras etapas que são efetuadas após as configurações iniciais do LCD, é a de tentar conectar-se a rede local.

Em seguida, o programa lê o valor vindo do DDS através do pino analógico A0 do *Arduino*, fornece um valor que equivale ao valor de pico do som.

O projeto final do medidor na *protoboard* pode ser vistos tanto na Figura 26, exibindo os valores medidos em decibéis, quanto na Figura 27 que acontece na etapa inicial do sistema, assim que é ligado (por isso os LEDs estão apagados).

Este valor é então passado para a função `calculateDecibels()` que efetua o cálculo de acordo com a equação 2.4, tendo o valor de referência em decibéis sido estabelecido previamente a partir de um aplicativo de decibelímetro em *software* chamado *Decibels* (processo de calibração).

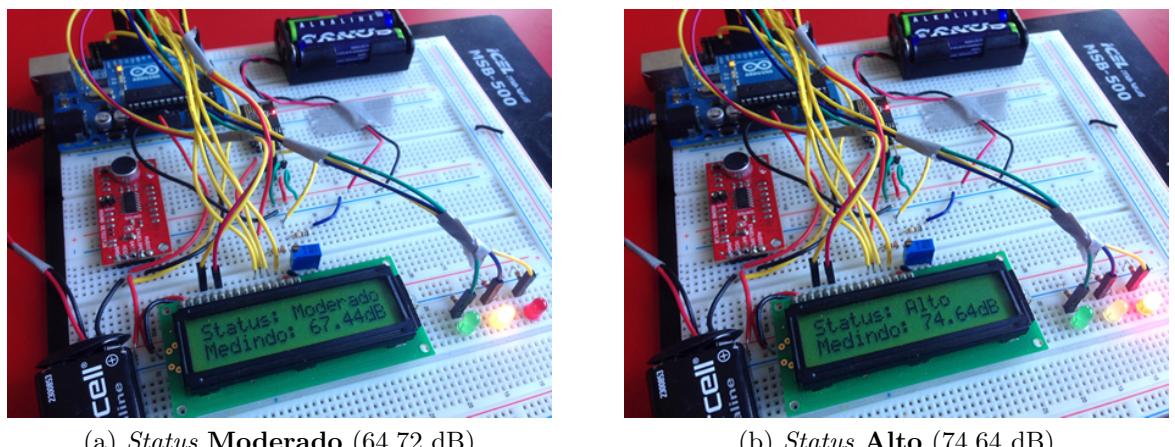


Figura 26 – Projeto do *hardware* montado na *protoboard* medindo o som.

Os valores lidos e menores ou iguais a 55 dB, caracterizam um ambiente *quieto*. Entre 55 dB e 70 dB, um ambiente de som *moderado*, e maiores que 70 dB para um

ambiente de som *alto*, valores estabelecidos baseando-se na NBR 10151 da Associação Brasileira de Normas Técnicas (ABNT).

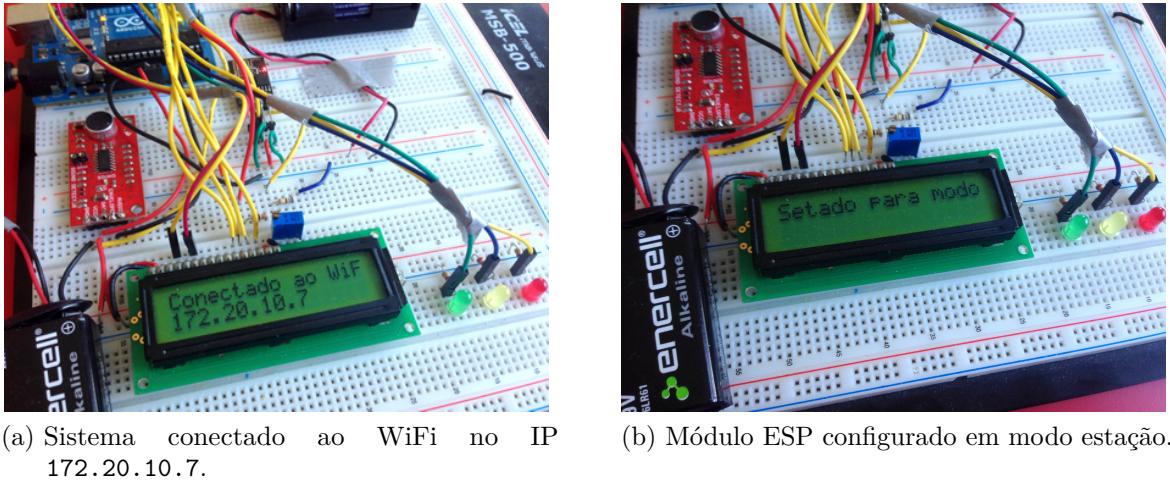


Figura 27 – Projeto do *hardware* montado na *protoboard* antes de medir: conectando-se à rede.

O projeto final do medidor na *protoboard* pode ser visto tanto na Figura 26, exibindo os valores medidos em decibéis, quanto na Figura 27 que exibe a etapa inicial do sistema, momentos após ser ligado (por isso os LEDs estão apagados).

3.4 Projeto de Software

O projeto do *software* de monitoramento dos dados medidos pelo *hardware* desenvolvido na Seção anterior ser uma aplicação *mobile*, levou em consideração o fato de que o Brasil ocupa a sexta posição em número de *smartphones* no mundo.

Embora dispositivos com sistema operacional *Android* ainda ocupem a primeira posição no mercado brasileiro (por fatores como variedade de aparelhos, faixa de preços e assim por diante), até julho deste ano, o sistema operacional *iOS* da *Apple* estava em terceiro lugar.

Logo, decidiu-se por desenvolver uma aplicação *mobile* para dispositivos *iOS*, que fosse capaz de localizar os pontos em que os dispositivos de *hardware* estão instalados, e ao mesmo tempo, pudesse exibir os valores medidos em tempo real.

As etapas do processo de desenvolvimento do *software* para dispositivos móveis rodando o sistema operacional *iOS*, é descrita a seguir juntamente com uma breve explicação das ferramentas utilizadas.

3.4.1 Sistema Operacional iOS

iOS é um sistema operacional da *Apple Inc.* que roda em dispositivos iPhone, iPad e iPod Touch. Além de gerenciar os recursos de *hardware* de tais dispositivos, tal sistema operacional (SO) também provê as tecnologias necessárias para o desenvolvimento de aplicações nativas.

Figura 28 – iPhone: exemplo de dispositivo que roda o sistema operacional iOS.



Fonte: extraído de [22]

O principal diferencial do iOS em relação aos demais SOs no que diz respeito ao desenvolvimento de aplicações, é a robustez do seu *kit* de desenvolvimento de *software*, *Software Development Kit* (SDK).

O iOS SDK possui uma série completa de ferramentas para desenvolver, instalar, rodar e testar aplicações para dispositivos iOS que são criadas utilizando-se de sua enorme gama de *frameworks* e do poder da linguagem *Objective-C*.

3.4.2 *Objective-C*

Embora a *Apple* tenha lançado recentemente uma nova linguagem (*Swift*) com o intuito de facilitar a programação e crescer o número de desenvolvedores para seus dispositivos, *Objective-C* desde a primeira versão do sistema operacional, tem sido a linguagem de programação utilizada para este fim.

Mesmo com o rápido crescimento de *Swift*, a maioria esmagadora de bibliotecas encontra-se implementada em *Objective-C*, sendo assim ainda bastante utilizada no desenvolvimento para iOS.

Objective-C é definida como uma modificação da linguagem C para prover a utilização de orientação a objetos e *dynamic runtime*, que é a capacidade de definir qual implementação de método chamar, antes de fazê-lo.

Ainda assim, *Objective-C* herda os tipos primitivos, sintaxe e estruturas de controle de fluxo de C.

Figura 29 – Exemplo de método em *Objective-C*: muito semelhante a C.

```
- (int)tamanhoDeCaracteres {
    int numeroDeCaracteres = 10;
    numeroDeCaracteres += 20;
    return numeroDeCaracteres;
}
```

Fonte: o autor.

3.4.3 XCode

A principal ferramenta no desenvolvimento de aplicações para iOS é o *XCode*. O *XCode* é um ambiente de desenvolvimento integrado, *Integrated Development Environment* (IDE), utilizado para construir desde a interface que o usuário vai utilizar (botões e telas), até o código que determina o comportamento da aplicação.

Devido ser uma ferramenta *all-in-one* e de fácil adaptação, o *XCode* é uma das razões que tornam o desenvolvimento de aplicações para *iOS* uma tarefa menos dispendiosa, aumentando a velocidade de programação e facilitando a depuração de código.

É importante salientar que o *XCode* também pode ser utilizado para criação de aplicações em linguagem C e C++.

3.4.4 Servidor Web: Plataforma *ThingSpeak*

Nas primeiras Seções descrevendo o comportamento desejado do sistema, foi estabelecido que os dados lidos pelo *hardware* seriam enviados pela Internet (através do Módulo ESP8266), e acessados via *software* pela aplicação móvel.

Deste modo, a plataforma escolhida como servidor *web* para abrigar tais dados foi a *ThingSpeak*, uma plataforma de dados *open-source* da *MathWorks Inc.*, empresa detentora do mundialmente conhecido *software* MATLAB.

O *ThingSpeak* permite que dispositivos embarcados conectados a Internet possam armazenar e visualizar dados de sensores ou atuadores, como *Arduino*, *Raspberry Pi* e assim por diante. O funcionamento do *ThingSpeak* baseia-se no conceito de *canais*, os quais contém atributos de dados como *status*, localização e outros.

Após a criação de um canal, a plataforma fornece um identificador que pode ser utilizado nas chamadas da sua Interface de Programação de Aplicação, *Application Programming Interface* (API), para escrita e leitura de dados do canal a partir de requisições utilizando o Protocolo de Transferência de Hipertexto, *Hypertext Transfer Protocol* (HTTP). Além do mais, a *ThingSpeak* disponibiliza em sua documentação uma lista de palavras

chave para o entendimento de sua API e que serão utilizados na descrição dos resultados do projeto.

Dentre elas:

- **Canal:** nome do local onde os dados podem ser inseridos ou recuperados, identificado por um identificador numérico. Exemplo: 57789;
- **Campo:** localização específica (são possíveis até 8) para os dados inseridos no canal. Um campo pode armazenar caracteres alfanuméricos. Exemplo: `decibeis = 44.50`;
- **Localização:** latitude e longitude de onde o dado está sendo enviado;
- **Feed:** nome para o conjunto de dados armazenados no canal, como latitude, longitude, campos e outros;
- **Chave de Escrita:** um código de 16 dígitos que permite uma aplicação escrever dados no canal. Exemplo: `A04V0RJJ7F022KMJ`;
- **Chave de Leitura:** um código de 16 dígitos que permite uma aplicação ler dados do canal.

3.5 Funcionamento do *Software* de Monitoramento

Utilizando-se do *framework* de mapas do iOS, o *MapKit*, foi desenvolvida uma aplicação chamada *Recife SM* (*Recife Sound Monitoring*) capaz de exibir as localizações de pontos da cidade em que o *hardware* acima desenvolvido for instalado.

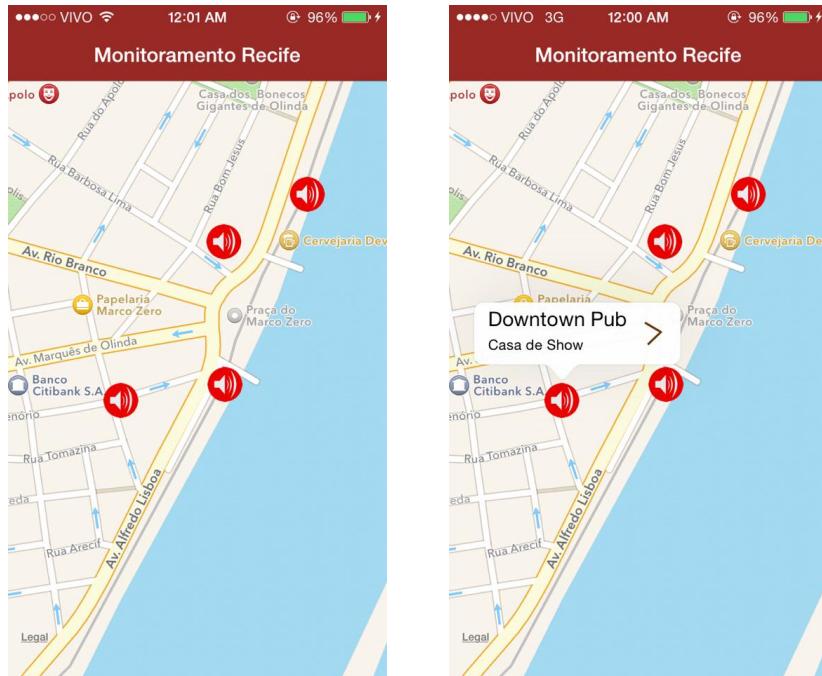
Supondo que o *hardware* tivesse sido instalado em quatro pontos do Recife Antigo, o usuário seria capaz de verificar informações sobre o barulho naquele local a qualquer momento, a partir do seu dispositivo iOS. Os locais escolhidos para a inserção no possível mapa de poluição sonora do Recife foram alguns estabelecimentos comerciais e um ponto turístico da cidade, na área da Praça do Marco Zero.

Como se pode ver na Figura 30, a tela inicial mostra os pontos da cidade para que o usuário possa escolher que local vai querer obter informações sobre o som do ambiente naquele momento.

O usuário então toca nos pontos do mapa e escolhe que local deseja saber detalhes, podendo monitorar o barulho no ambiente de qualquer lugar, com seu *smartphone* e acesso a Internet. É possível ver que o usuário tem a opção de escolher que local ele deseja monitorar. A seta do lado direito indica que é possível saber mais detalhes a respeito daquele local escolhido.

Figura 30 – Telas iniciais do *software* de monitoramento.

- (a) Tela inicial: pontos da cidade no mapa.
- (b) Usuário escolhe o local ao tocar no ponto do mapa.



Ao abrir a tela de Detalhes, como visto na Figura 31, o usuário além de visualizar o valor do som do ambiente em decibéis, pode também efetuar uma ligação ou comunicar por e-mail o local que esteja ultrapassando os limites estabelecidos pelos órgãos da cidade.

Figura 31 – Tela de detalhes do local escolhido pelo usuário: dados do medidor.



Fonte: o autor.

4 Resultados e Testes

Este capítulo descreve os testes realizados com as ferramentas desenvolvidas tanto em *hardware* quanto em *software*. Na Seção 4.1, o medidor foi ligado para efetuar medidas de valores de decibéis e compará-las com um medidor em *software*, calculando-se o erro entre os valores.

Em seguida, na Seção 4.2, foi analisado o comportamento do circuito durante a fase de envio de dados para a plataforma *ThingSpeak*, bem como exibido alguns detalhes do comportamento do programa do *Arduino* nas requisições HTTP.

Por fim, a Seção 4.2.2 explica os testes realizados no momento da leitura dos dados da plataforma *web*.

4.1 Medições do *Hardware*

Logo após o projeto desenvolvido, o sistema foi ligado para algumas medições e comparações. Como não foi possível obter um MNPS profissional para comparação, os valores obtidos foram comparados com os de uma aplicação MNPS em *software*, o aplicativo *Decibels*.

Para um ambiente de uma sala com pessoas conversando próximo aos medidores, obtiveram-se os seguintes valores:

Tabela 2 – Tabela com valores em dB das medições do *hardware* e do aplicativo *Decibels*.

<i>Hardware</i> desenvolvido	Aplicativo <i>Decibels</i>
66.52	68.13
75.22	79.15
66.03	68.20
65.89	69.17
73.69	79.91
68.65	68.23
66.52	70.12
66.99	68.79
67.12	69.87

De posse dos valores da Tabela 2, é possível calcular um erro médio que relaciona o que foi medido pelo *hardware* e o valor ideal medido pelo aplicativo. Assim, sendo x_i o valor medido pelo *hardware* e \hat{x}_i o valor ideal obtido do aplicativo, o erro médio absoluto e

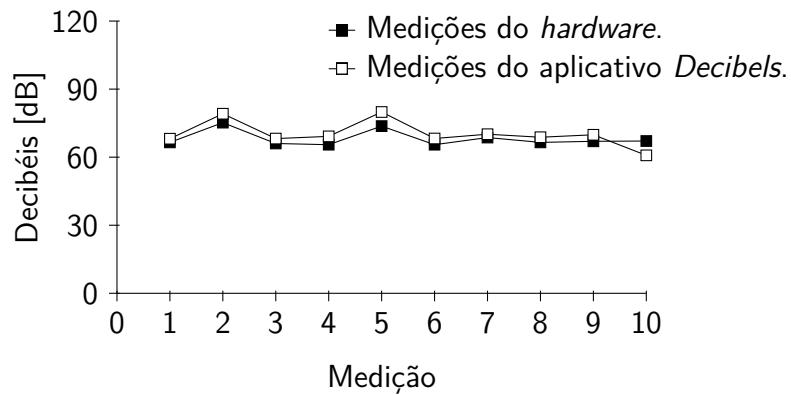
é dado por:

$$e = \frac{\sum_{i=1}^N |x_i - \hat{x}_i|}{N} = \frac{24.94}{10} \Rightarrow \boxed{e = 2.494} \text{ (dB)} \quad (4.1)$$

O erro absoluto e de 2.494 mostra que os valores medidos pelo *hardware* estão bem próximos de um MNPS confiável, mesmo que em *software*.

Também, a partir dos dados exibidos na Tabela 2, foi plotado o Gráfico 32 que ilustra a proximidade dos valores medidos.

Figura 32 – Gráfico de medições do *hardware* × aplicativo *Decibels (software)*.



Fonte: o autor.

O ideal seria o uso de um MNPS profissional para fazer outras comparações e definir a precisão do projeto desenvolvido a partir de outros testes e ambientes. No entanto, os resultados para o medidor foram bastante satisfatórios e a precisão condizente com o esperado.

4.2 Processo de Atualização e Leitura na *ThingSpeak*

4.2.1 Escrita dos valores via *hardware*

Como explicitado anteriormente, existem duas chaves de acesso (leitura e escrita) a um canal do *ThingSpeak*, sendo a de escrita utilizada para atualização do canal.

Após o desenvolvimento do *hardware* efetuando a leitura dos valores em decibéis e exibindo-os no LCD, um dos fatores para demonstrar o correto funcionamento do projeto seria o envio de dados para o servidor da *ThingSpeak*. Sendo assim, foi criado um canal na plataforma para o qual a API do *ThingSpeak* gerou duas chaves distintas de escrita e leitura. Também foi adicionado ao canal o campo **field1:Decibeis** para receber o valor medido pelo *hardware*.

Para a realização dos testes, foi criado um canal de exemplo (no modo de acesso privado) de um hipotético estabelecimento comercial sendo monitorado.¹ A cada 1 minuto, o programa rodando no *Arduino* realizou a chamada `void sendToThingSpeak(float valueInDecibels)` que recebia o valor final calculado em decibéis, e criava uma conexão TCP com o IP da *ThingSpeak* (184.106.153.149).

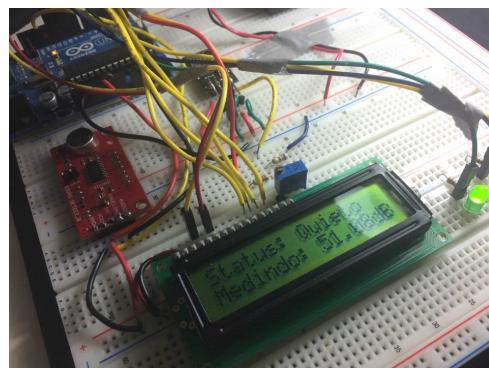
Sendo `IGPECOUJCH5I6IAY` a chave de escrita gerada pela plataforma, montou-se a chamada `GET` incubida de atualizar o canal.

Logo, a chamada montada no programa:

```
GET /update?key=IGPECOUJCH5I6IAY&field1=%s
```

(sendo `s` o valor em decibéis armazenado na variável `valueInDecibels`) fez a escrita no canal de acordo com o esperado. Na Figura 33 pode-se ver diretamente no LCD o sistema efetuando uma leitura de 48.58 dB momentos antes de iniciar o processo de envio para a *ThingSpeak* descrito acima:

Figura 33 – Sistema medindo 51.08 dB momentos antes de enviar.



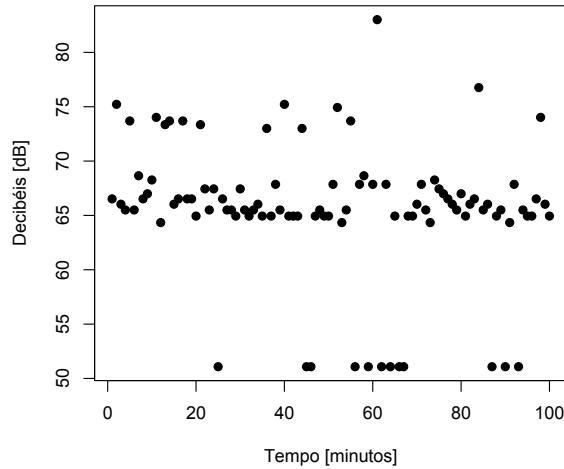
Fonte: o autor.

Para verificar a autonomia e estabilizade do sistema, o mesmo foi deixado ligado com alimentação de uma bateria de 9V e duas pilhas AA de 1,5V cada especialmente para o módulo ESP, medindo o ambiente de uma sala com pessoas conversando por pouco mais de uma hora e quarenta minutos. Abaixo vê-se o gráfico Tempo × Decibéis gerado com os valores obtidos.

O Gráfico 34 mostra que dos 100 valores de decibéis medidos, a maior parte se encontra entre 65 e 70 decibéis, o que resulta em um ambiente com *status* de moderado, o que era esperado como comportamento ideal (para a situação testada) no processo de medição pelo sistema.

¹ a *logo* do estabelecimento é exibida apenas a nível de ilustração e demonstração de funcionamento do sistema

Figura 34 – Gráfico com valores medidos pelo sistema em decibéis durante 100 minutos.



Fonte: o autor.

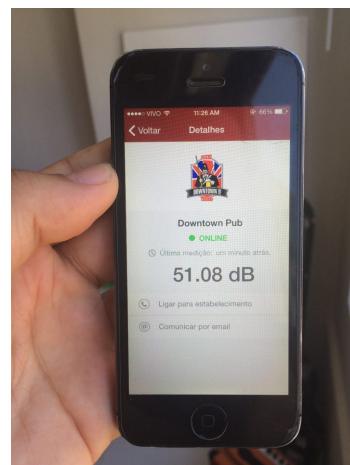
4.2.2 Leitura dos valores via *software*

Para efetuar a leitura dos valores escritos na plataforma a partir da chave de leitura (A04V0RJJ7F022KMJ), a aplicação rodando no iOS efetuou uma requisição a partir da URL <http://api.thingspeak.com/channels/53759/feed/last.json?key=A04V0RJJ7F022KMJ>, pois a API fornece essa chamada para acessar apenas o último dado enviado ao canal.

Como exibido na subseção anterior, o valor de 51.08 dB que o medidor apresentava no LCD foi enviado a plataforma e assim, tornou-se acessível pelo aplicativo.

A Figura 35 exibe o usuário efetuando a leitura do valor em decibéis enviado pelo *hardware* através de conexão WiFi com a rede local:

Figura 35 – *Software* (iOS app) efetuando a mesma leitura de 51.08 dB.



Fonte: o autor.

As requisições HTTP feitas ao *ThingSpeak* retornaram os dados em um formato conhecido

por *JavaScript Object Notation* (JSON), o qual contém o identificador da entrada, data de atualização e o valor do campo.

O *software* então recebe este JSON, e extrai a data de atualização e o valor que corresponde ao medido em decibéis. Para os testes, o retorno da chamada ao serviço foi do tipo:

```
{"created_at": "2015-12-02T05:08:20Z", "entry_id": 397, "field1": "51.08"}
```

Este retorno é relativo ao teste executado acima (sistema medindo 51.08 dB), o que possibilitou a exibição correta do valor na tela de Detalhes da aplicação. A cada nova chamada os dados de data, identificador e campo eram atualizados.

5 Conclusões e Trabalhos Futuros

Neste trabalho foi descrito o processo de desenvolvimento de um sistema de detecção de intensidade sonora, envolvendo a criação de um projeto em *hardware* e *software* capaz de medir um valor em decibéis e enviá-lo para um servidor *web*, automatizando o processo de medição de som.

Durante a realização de testes, percebeu-se que o comportamento do sistema se deu como esperado. Na parte do *hardware*, a fase de detecção e leitura do valor em decibéis, exibição do dado no LCD, LEDs de *status* e envio de dados para o servidor *web* a partir de uma conexão à rede local, foram etapas pré-estabelecidas na fase de projeto e que quando testadas, comportaram-se de maneira adequada.

Também na fase de leitura dos valores pela aplicação em *software* rodando no iPhone, desde a fase de compilação da aplicação no iOS, passando pela montagem da requisição HTTP, recebimento do objeto vindo da *ThingSpeak*, interpretação e exibição de dados para o usuário, também são parte objetivos estabelecidos e atingidos corretamente.

Um dos fatores de melhoramento seria a utilização de um MNPS profissional, que geraria valores mais confiáveis e colocaria a precisão do sistema à prova. Além do mais, um fator que também pode ser testado para atestar a confiabilidade do sistema é o uso contínuo por um longo período de tempo, que trata-se de um cenário mais realista para este tipo de projeto.

A transição do sistema para uma placa de circuito impresso trata-se de um projeto futuro, integrando todos os componentes em um único circuito capaz de ser colocado em um recipiente e instalado em locais desejados como um dispositivo eletrônico comum, eliminando a delicadeza e propensão a erros de ter o circuito montado numa placa de prototipação.

O desenvolvimento do *software* para outros sistemas operacionais expressivos do mercado seria de grande valia, uma vez que seria capaz de alcançar todas as plataformas *mobile* e aumentaria a popularidade da aplicação, ajudando cidadãos comuns a verificar a qualquer momento a intensidade dos sons emitidos por certos locais da cidade.

A possibilidade do sistema de monitorar o som em decibéis, sendo seu custo total de implementação muito mais baixo que o de um MNPS, e disponibilização deste dado para acesso através de uma aplicação em *software*, torna-o uma ferramenta bastante útil no combate a poluição sonora nos grandes centros urbanos.

Por fim, implementando-se a opção de comunicar o local dos possíveis danos à saúde causados pelo excesso de barulho (através de e-mail ou ligação) no momento em que tal excesso é verificado, o projeto ganha o potencial de trazer mudança no comportamento das pessoas, induzindo-as a controlar suas emissões de ruídos, contribuindo para um melhor convívio e qualidade de vida.

Referências

- [1] ABBASPOUR, M. et al. Hierarchical assessment of noise pollution in urban areas – a case study. *Transportation Research Part D: Transport and Environment*, v. 34, p. 95–103, 2015. ISSN 1361-9209. Citado na página 14.
- [2] MURPHY, E.; KING, E. A. *Environmental Noise Pollution*. 1. ed. [S.l.]: Elsevier, 2014. ISBN 9780124115958. Citado na página 14.
- [3] CACIARI, T. et al. Noise-induced hearing loss in workers exposed to urban stressors. *Science of The Total Environment*, v. 463–464, p. 302–308, 2013. ISSN 0048-9697. Citado na página 14.
- [4] BERGLUND, B.; LINDVALL, T.; SCHWELA, D. H. *Guidelines For Community Noise*. Avenue Appia 20, 1211 Genève 27, Suíça, 1999. Citado na página 14.
- [5] CARNEIRO, A. S. da S. *Poluição Sonora – Silento e o Barulho*. 3. ed. Rua do Imperador D. Pedro II, 473 - Recife, PE, 2012. Citado na página 14.
- [6] RECIFE, S. de Meio Ambiente e Sustentabilidade do. *Poluição Sonora*. [S.l.], 2012. Disponível em: <<https://meioambienterecife.wordpress.com/acoes/poluicao-sonora/>>. Acesso em: 07 de agosto de 2015. Citado na página 15.
- [7] HALLIDAY, D.; RESNICK, R.; WALKER, J. *Fundamentals of Physics*. 10. ed. [S.l.]: Wiley, 2013. ISBN 978-1118230718. Citado 2 vezes nas páginas 17 e 20.
- [8] YOUNG, H. D.; FREEDMAN, R. A.; FORD, A. L. *University Physics with Modern Physics*. 13. ed. [S.l.]: Addison-Wesley, 2011. ISBN 978-0321696861. Citado 4 vezes nas páginas 17, 18, 20 e 22.
- [9] INC., A. *Soundtrack Pro 3 – User Manual: Audio Fundamentals*. 13. ed. [S.l.], 2009. Citado 2 vezes nas páginas 18 e 19.
- [10] KASPER, C. *Good Vibrations–The Vibrational Qualities of Young Living Oils*. [S.l.], 2014. Disponível em: <<http://younglivingwoman.com/good-vibrations-the-vibrational-qualities-of-young-living-oils/>>. Acesso em: 17 de outubro de 2015. Citado na página 18.
- [11] CLASSROOM, T. P. *Sound Waves and Music – The Nature of a Sound Wave*. [S.l.], 2015. Disponível em: <<http://www.physicsclassroom.com/class/sound/Lesson-1/Sound-is-a-Pressure-Wave>>. Acesso em: 21 de setembro de 2015. Citado na página 19.
- [12] CORPORATION, C. *Noise Control For Buildings – Guidelines for Acoustical Problem Solving*. 1. ed. Valley Forge, PA, 2015. Citado na página 20.

- [13] DEPARTMENT, E. P. *Characteristics of Sound and the Decibel Scale*. [S.l.], 2015. Disponível em: <http://www.epd.gov.hk/epd/noise_education/web/>. Acesso em: 23 de outubro de 2015. Citado na página 22.
- [14] BRÜEL; SOUND, K. *Basic Concepts of Sound*. [S.l.], 2012. Disponível em: <http://cafefoundation.org/v2/pdf_tech/>. Acesso em: 03 de novembro de 2015. Citado 2 vezes nas páginas 22 e 23.
- [15] OIL, I. A. of; PRODUCERS, G. *Fundamentals for underwater sound*. [S.l.], 2008. Citado na página 22.
- [16] AZ-8921 Ljudmätare. [S.l.], 2015. Disponível em: <<http://younglivingwoman.com/good-vibrations-the-vibrational-qualities-of-young-living-oils/>>. Acesso em: 21 de setembro de 2015. Citado na página 23.
- [17] STUDIO Tone vs. Live Tone and the Fletcher Munson Curve. [S.l.], 2006. Disponível em: <<http://line6.com/support/page/>>. Acesso em: 19 de setembro de 2015. Citado na página 25.
- [18] A Weighting. [S.l.], 2012. Disponível em: <<http://www.diracdelta.co.uk/science/>>. Acesso em: 19 de setembro de 2015. Citado na página 25.
- [19] THE ESP8266. [S.l.], 2014. Disponível em: <<http://espressif.com/en/products/esp8266/>>. Acesso em: 15 de setembro de 2015. Citado 2 vezes nas páginas 29 e 30.
- [20] SOUND Detector Hookup Guide. [S.l.], 2013. Disponível em: <https://learn.sparkfun.com/tutorials/sound-detector-hookup-guide?_ga=1.41145250.1435560018.1439029156>. Acesso em: 18 de outubro de 2015. Citado 2 vezes nas páginas 31 e 33.
- [21] 16X2 LCD Display Yellow/green LED Backlight. [S.l.], 2014. Disponível em: <<http://www.hobbytronics.co.uk/lcd-16-2-backlight>>. Acesso em: 20 de novembro de 2015. Citado na página 31.
- [22] [S.l.], 2015. Disponível em: <<http://www.amazon.com/Apple-iPhone-16GB-White-Mobile/dp/B00F2LBS30>>. Acesso em: 02 de dezembro de 2015. Citado na página 38.