DOI: 10.5007/2175-7941.2013v30n3p579

NOVAS TECNOLOGIAS NO ESTUDO DE ONDAS SONORAS[+*]

Marisa Almeida Cavalcante
Pontifícia Universidade Católica
São Paulo – SP

Resumo

O presente trabalho propõe a construção de um Tubo de Kundt adaptado às novas tecnologias disponíveis para o ensino de física. Tem como principal objetivo utilizar a placa Arduino em experimentos didáticos envolvendo o estudo de ondas sonoras. O Arduino é uma placa de controle I/O baseada no micro-controlador Atmega (Atmel) e foi projetado inicialmente para fins didáticos. O fato da linguagem de programação utilizada e hardware serem do tipo open source (código aberto) possibilitou sua ampla difusão em diversas áreas. Uma das intenções deste projeto é difundir o uso deste recurso para fins educacionais e particularmente no estudo de ondas sonoras estacionarias em tubos, contribuindo com a melhora na abordagem deste conteúdo no ensino e aprendizagem de Física.

Palavras-chave: *Tubo de Kundt. Arduino. Processing. Simplot.*

Abstract

This paper proposes the construction of a Kundt tube adapted to new available technologies for teaching Physics. We propose to use the Arduino board in didactic experiments involving the study

[+] New technologies in the study of sound waves

[*] *Recebido: janeiro de 2013.*
Aceito: agosto de 2013.

of sound waves. The Arduino is a control board I /O, based on the microcontroller Atmega (Atmel) and it was, initially, designed for didactic purposes. The fact of the programming language and hardware are the open source type enabled their wide dissemination in various areas. This work aims to spread the use of this resource for educational purposes and particularly in the study of stationary sound waves in pipes, contributing to the improvement in the approach of this content in Physics teaching and learning.

Keywords: *Kundt's tube. Arduino. Processing. Simplot.*

I. Introdução

I.1 A placa Arduino

Arduino[1] é uma plataforma de hardware de código aberto de protótipos eletrônicos flexíveis. É destinado a artistas, designers, estudantes e pessoas interessadas em criar objetos ou ambientes interativos

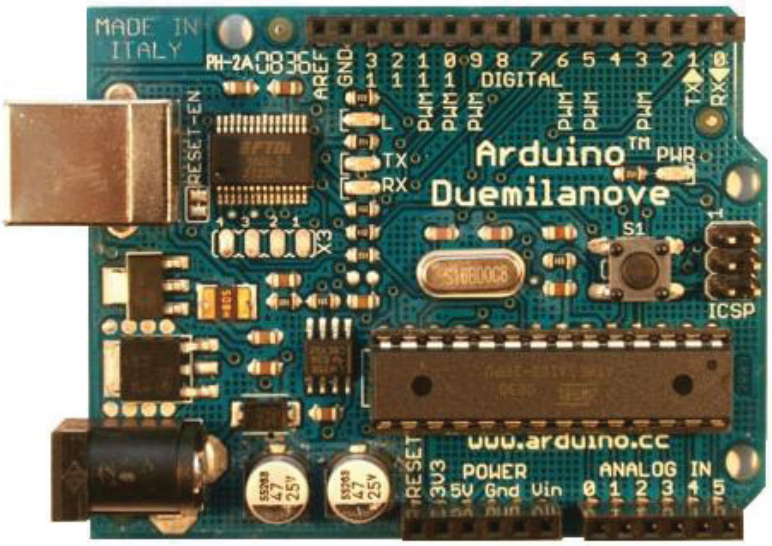


Fig. 1– Placa micro-controladora Arduino (versão Duemilanove, 2009)[1].

O Arduino é um microcontrolador que permite associação com uma vasta gama de sensores relacionados à pressão, intensidade luminosa, movimento, etc., e permite acionar motores, LEDs e outros atuadores, que podem estar ou não correlacionados aos sinais provenientes destes sensores. Desta forma, é possível enviar comandos a partir das respostas destes sensores, tendo em vista que ele apresenta além de um conversor A/D um micro controlador. O micro controlador na placa (Atmel AVR de 8 bits) é programado usando uma linguagem baseada em C e o ambiente de desenvolvimento Arduino® é baseado em linguagem Processing. Os projetos com Arduino são ajustados com computadores e minicomputadores, usando assim a saída USB. A conversão de sinais analógicos em digitais se dá em 10 bits obtendo resoluções de 1 parte em 1024[(2)].

A placa do Arduino pode ser montada ou comprada pré-montada e o software para programação pode ser baixado gratuitamente. Comercialmente, existem várias versões, em torno de 11 disponíveis no mercado, adaptadas de acordo com a necessidade do usuário. Além disso, os projetos para a montagem das diferentes placas estão disponíveis sob uma licença de código aberto, para que os usuários montem livremente suas próprias adaptações, além daquelas já comercializadas.

I.2 Linguagem Processing®

Processing[(3)] é uma linguagem de programação/ambiente de desenvolvimento e comunidade online que desde 2001 tem promovido a difusão de software dentro das artes visuais. Inicialmente criado para servir como software e ensinar fundamentos de programação de computadores dentro de um contexto visual, ele rapidamente evoluiu para uma ferramenta para criação de trabalho profissional.

A linguagem Processing é livre, alternativa *open source* (código aberto), tornando-se acessível para as escolas e alunos individuais. Seu status de código aberto incentiva a participação da comunidade e a colaboração que é vital para o seu crescimento. Programadores da comunidade contribuem com códigos, respondem a perguntas e constroem bibliotecas para ampliar as possibilidades do software. Livre para execução em qualquer plataforma de sistema operacional, dezenas de milhares de empresas, artistas, designers, arquitetos e pesquisadores usam Processing® para criar uma gama extremamente diversificada de projetos.

Fig. 2 – Linguagem de Programação Processing®.

I.3 O experimento do Tubo de Kundt

O estudo de ondas mecânicas é base para o entendimento de diversos fenômenos da natureza, entre os mais importantes, destaca-se o estudo de ondas sonoras. A compreensão deste fenômeno também é importante porque permite uma introdução a fenômenos ondulatórios encontrados em outras áreas da física, tais como eletromagnetismo, ótica e física moderna e apesar de sua relevância, pouca atenção tem sido dada às dificuldades encontradas pelos estudantes na aprendizagem de ondas mecânicas. Em particular, o número de trabalhos publicados sobre o tema é pequeno, pelo menos quando comparado a áreas, como mecânica ou eletromagnetismo[4 e 5].

Escolhemos reproduzir o experimento do Tubo de Kundt, utilizando-se de novas tecnologias, não apenas por sua importância histórica, mas pela ampla possibilidade que ele oferece para a compreensão do fenômeno de ressonância bem como a possibilidade de se obter a determinação experimental da velocidade do som no ar.

A montagem foi desenvolvida inicialmente[6] por August Adolf Eduard Eberhard Kundt, que foi um físico alemão nascido em Schwerin, especialista em som e a luz. Em 1866 desenvolveu o método de investigação de ondas sonoras através do ar em tubos. Com a observação da vibração através de tubos ele pode concluir a formação dos harmônicos e a possibilidade de medi-los. O aparato passou a ser chamado de Tubo de Kundt e permitiu ao cientista medir a velocidade do som em diversos gases e ambientes.

O Tubo de Kundt é uma montagem feita para ensaios acústicos que consiste em um tubo (vidro ou acrílico) que contém ar e pó de serra ou cortiça em seu interior. Pode ser provido ou não de um pistão ajustável em uma das suas extremidades. Nele produzem-se ondas estacionárias fazendo uma fonte sonora vibrar em uma determinada frequência. As vibrações são transmitidas para o pó de serra (ou de cortiça) pelo ar que está contido dentro do tubo. Observa-se que, quando ocorre ressonância, em certas regiões do tubo há acumulo de cortiça em algumas regiões que não apresentam vibrações longitudinais, essas regiões representam os nós da onda gerada e regiões onde há vibração do pó de cortiça, chamadas de ventres ou antinodos.

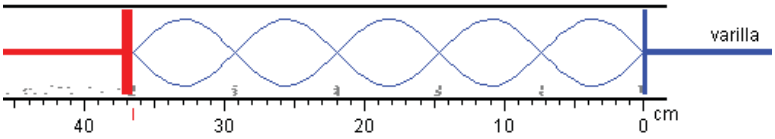


Fig. 3 – Tubo de Kundt esquematizado[1] *(7).*

Uma onda sonora(8) trata-se de uma onda mecânica, atuante no nível molecular, cujo fenômeno perceptível associado é o som. Diferentemente das ondas em uma corda, são invisíveis. O som se propaga num determinado meio, sempre que se produz uma variação de pressão no mesmo. Uma onda sonora se propaga numa sucessão de compressões e rarefações, e em cada material esses movimentos têm uma característica peculiar. Existe uma grandeza que dá conta dessas variações em um meio, que é o módulo volumétrico da elasticidade B. Este módulo leva em conta a variação de pressão e a variação fracional de volume e é definido como;

$$B = -\frac{\Delta p}{\left(\frac{\Delta V}{V}\right)} \tag{1}$$

No limite, para $\Delta V \to 0$ temos:

$$B = -V\left(\frac{dp}{dV}\right) \tag{2}$$

Ou ainda;

---

1 Figura disponível em: <http://www.sc.ehu.es/sbweb/fisica/ondas/acustica/kundt/kundt.htm>. Acesso em: 12 ago. 2013.

$$B = \rho \left(\frac{d\rho}{dp}\right) \quad (3)$$

Onde ρ representa a densidade volumétrica do gás.

A velocidade de propagação de uma onda sonora é proporcional à raiz quadrada da razão entre o módulo de elasticidade e a densidade meio em questão. Maior densidade implica em menor velocidade, enquanto maior elasticidade implica em maior velocidade.

$$v = \sqrt{\frac{B}{\rho}} \quad (4)$$

Uma expressão muito útil para a velocidade de propagação do som no ar[9], é fornecida na equação 5 em que o valor seu valor é fornecido em função da temperatura em graus Celsius, onde:

$$v = 330{,}4 + 0{,}59\ T\ (m/s) \quad (5)$$

I.3.1 Ondas sonoras

Considere um alto-falante produzindo ondas sonoras segundo o esquema da Fig. 4[10].

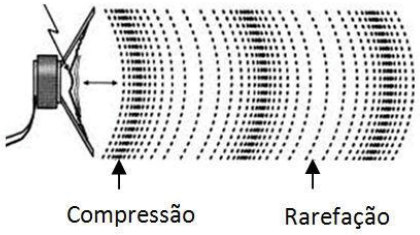


Fig. 4 – Ondas sonoras geradas por um alto-falante[2].

A perturbação iniciada pelo alto-falante se propaga pelo tubo devendo, obedecer à equação de D' Alembert, onde $\psi(x,t)$, representa a grandeza física que se propaga ao longo do eixo horizontal x com uma dada velocidade v.

$$\frac{\delta^2\psi(x,t)}{\delta x^2} = \frac{1}{v^2}\frac{\delta^2\psi(x,t)}{\delta t^2} \quad (6)$$

[2] Figura disponível em <http://www.infoescola.com/fisica/ondas-longitudinais/>. Acesso em: 13 ago 2013.

Enquanto a perturbação se propaga ao longo do tubo, teremos alterações no volume e pressão do gás. Vamos considerar uma situação simplificada, na representada na Fig. 5. Em um instante t_1, dois elementos de volume estão nas suas respectivas posições de equilíbrio e, em um instante posterior $t_2 = t_1 + \Delta t$, eles sofreram os deslocamentos de acordo com a equação (6).

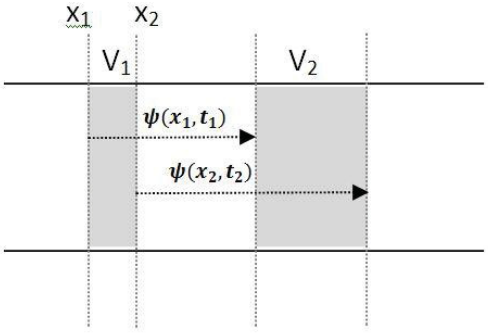


Fig. 5 – Dois elementos de volume sofrendo deslocamento nos instantes t_1 e t_2.

Em primeiro lugar o elemento de volume inicial V_1 é dado por:

$$V_1 = A(x_2 - x_1) = A\Delta x \tag{7}$$

A variação de volume é dada por:

$$\Delta V = A(\psi_2 - \psi_1) = A\Delta\psi \tag{8}$$

Substituindo (7) e (8) em (1) temos:

$$\Delta p = -B\frac{\delta\psi}{\delta x} \tag{9}$$

Substituindo (4) em (9) temos;

$$\Delta p = -\rho v^2 \frac{\delta\psi}{\delta x} \tag{10}$$

Observe que a relação obtida em (10) envolve uma derivação o que implica em uma defasagem entre o deslocamento e a pressão igual a $\pi/2$[11]. Isso significa que pontos de nós de deslocamento correspondem a antinodos de pressão e vice-versa. A Fig. 6 mostra esta defasagem.

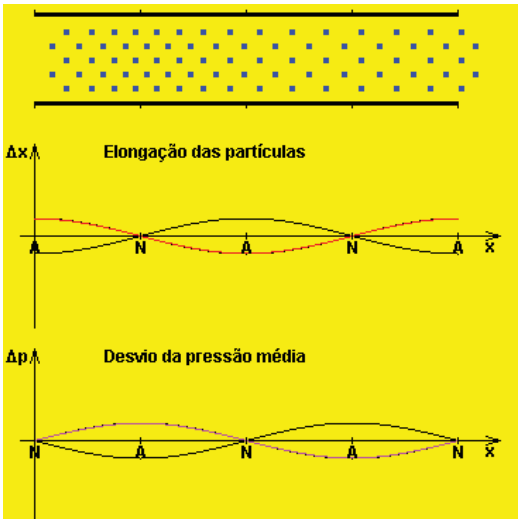


Fig. 6 – Simulação de computador[3] *mostra o deslocamento do ar em um tubo com extremidades abertas, além das variações de pressão e elongação das partículas*[(12)].

I.3.2 Ondas Estacionárias

Considere um tubo de comprimento L, fechado em uma das extremidades com um alto-falante na extremidade oposta, conforme Fig. 7.

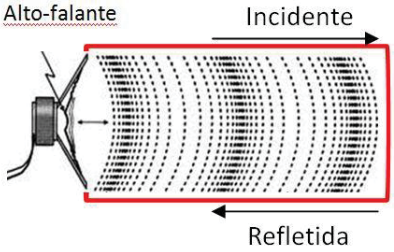


Fig. 7 – Ondas sendo refletidas na extremidade do tubo.

[3] Applet disponível em: <http://www.walter-fendt.de/ph14br/stlwaves_br.htm>. Acesso em: 13 ago 2013.

Considerando a propagação no eixo horizontal a superposição das ondas incidente e refletida conduz a seguinte relação:

$$\psi(x,t) = [2Asen(kx)]cos\,(wt) \qquad (11)$$

sendo $k= 2\pi/\lambda$ e $w=2\pi f$

O sistema mais simples, em que uma onda estacionária pode ocorrer é a reflexão entre duas superfícies rígidas, Fig. 8.

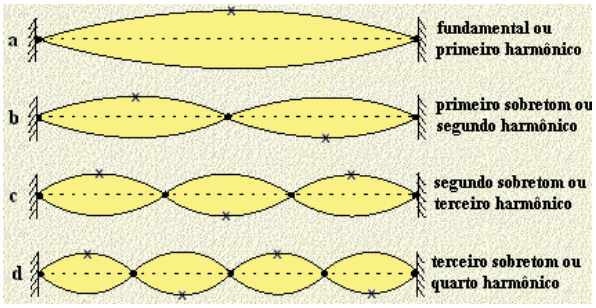


Fig. 8[4] *– Ondas estacionárias*[(13)].

A onda estacionária realiza incessantemente o caminho entre os dois refletores, retraçando as mesmas posições que são relacionadas com alguns comprimentos de onda específico (e, por consequência, frequências específicas), relacionados com a distância entre as superfícies.

Note que, para uma onda longitudinal, devido à defasagem $\pi/2$ existente entre deslocamento e pressão, pontos de nó de deslocamento correspondem à máxima variação de pressão (antinodos ou ventres) e vice-versa.

É possível determinar, a partir das distâncias entre as superfícies refletoras (L), algumas características interessantes acerca das ondas estacionárias. Se considerarmos um tubo fechado nos dois extremos, a maior onda possível tem metade de um comprimento de onda, equivalente à distância entre os refletores (Fig. 10).

[4] Figura disponível em: <http://www.feiradeciencias.com.br/sala10/10_15.asp>. Acesso em: 13 ago. 2013.

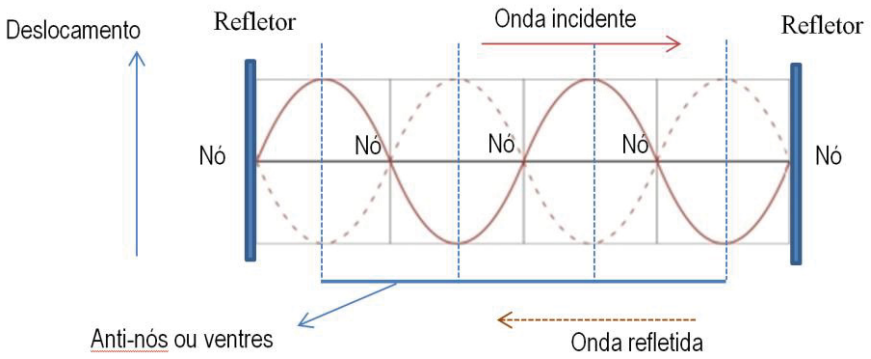


Fig. 9 – Nós e Antinós em ondas estacionárias de deslocamento.

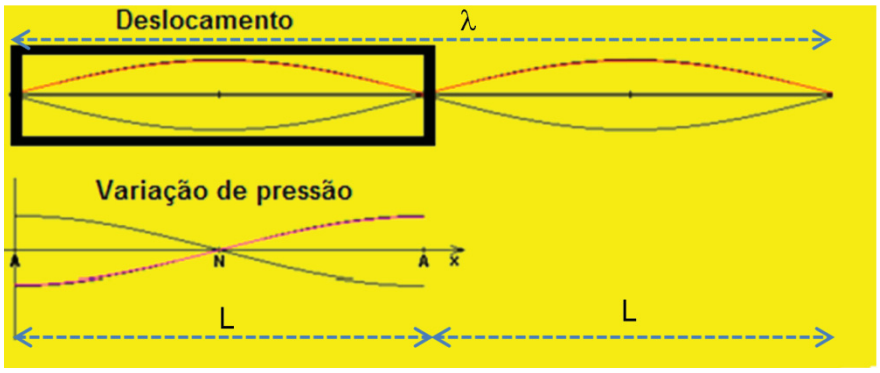


Fig. 10[5] *– O maior comprimento de onda (λ) para um tubo fechado dos dois lados deve ser igual a 2L. Na parte superior, representamos a onda de deslocamento e, abaixo, a variação de pressão, indicando a defasagem de π/2.*

Tal consideração implica em uma menor frequência dada por:

$$f_{\text{menor}} = \frac{v}{2L} \tag{12}$$

sendo

[5] Figura construída tomando por base o simulador disponível em: <http://www.walter-fendt.de/ph14br/stlwaves_br.htm>. Acesso em: 13 ago. 2013.

f_{menor} = frequência em Hertz (*Hz*);

L = distancia entre as superfícies refletoras (*m*);

λ_{maior} = maior comprimento de onda possível em um tubo fechado nos dois extremos (*m*);

v = velocidade do som no ar (*m/s*).

É fácil perceber que qualquer múltiplo inteiro (n) da metade do comprimento de onda estará contido entre esses dois refletores. Por esta razão existem infinitos valores de frequências possíveis, mas sempre múltiplos inteiros da relação *v/2L*, ou seja;

$$f_n = \frac{nv}{2L} \tag{13}$$

no qual f_n = frequência e *n* = 1, 2, 3...

Essas frequências são chamadas frequências modais de um tubo, equivalentes aos modos de vibração deste tubo (ou harmônicos).

Para tubos com apenas um lado fechado (uma única superfície refletora) e outro aberto, as ondas refletem de um lado do tubo e estão *"livres*" no lado oposto. Por isso a maior onda estacionária que pode estar contida no tubo apresenta um comprimento de onda igual a quatro vezes o comprimento do tubo (L). Para tubos abertos, teremos um ponto de nó para a pressão (rarefação do ar) no lado da abertura. O ponto de antinodo, ou máximo de pressão, só se forma na superfície refletora. Portanto, para tubos fechados em uma única extremidade, não existem frequências nodais em múltiplos pares da frequência fundamental.

A equação que define as frequências que podem existir neste tipo de tubo é:

$$fn = \frac{(2n+1).v}{4L} \tag{14}$$

na qual *fn* = frequência e n = 1, 2, 3...

Tais considerações mostram que ondas estacionárias são observadas em tubos, quando a onda incidente satisfaz certas condições de contorno. Uma extremidade aberta corresponde a um nó de pressão e, por conseguinte, um antinó de deslocamento, e uma extremidade fechada corresponde a um nó de deslocamento ou um antinó de pressão (máxima pressão). Na verdade, a variação de pressão só se anula um pouco adiante da extremidade aberta, a coluna de ar vibrante se estende um pouco além da extremidade aberta.

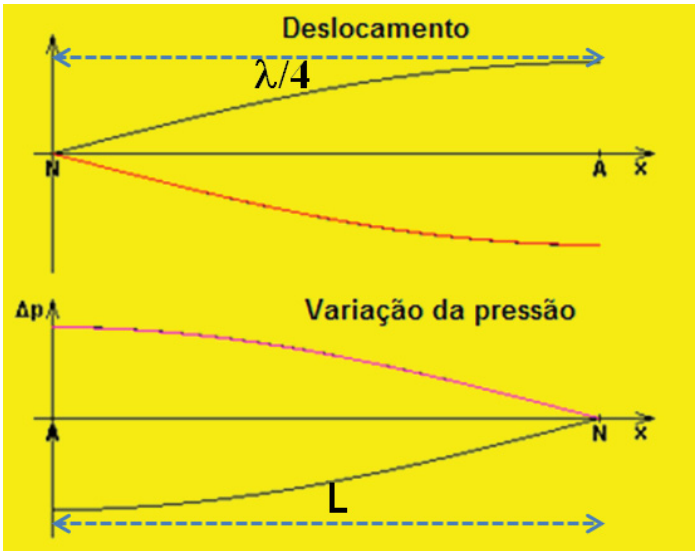


Fig. 11[6] *– Ondas estacionárias em tubo fechado.*

Uma pequena correção deve ser efetuada para tubos cujas paredes apresentam baixa espessura(14). Para um tubo de comprimento L e raio R devemos adicionar 0,6 R ao comprimento L para cada extremidade aberta.

Portanto, um tubo com uma das extremidades abertas o comprimento efetivo é dado por:

$$L_{ef} = L + 0{,}6\ R \tag{15}$$

Já para duas extremidades abertas teremos:

$$L_{ef} = L + 1{,}2\ R \tag{16}$$

O tubo com as duas extremidades abertas possui antinodos (ou ventres) de deslocamento nas nestas extremidades e consequentemente nó de pressão. Na Fig. 12, ilustramos a onda de deslocamentos para o primeiro modo de vibração, também chamado de modo fundamental.

[6] Figura construída tomando por base o simulador disponível em: <http://www.walter-fendt.de/ph14br/stlwaves_br.htm>. Acesso em: 13 ago. 2013.

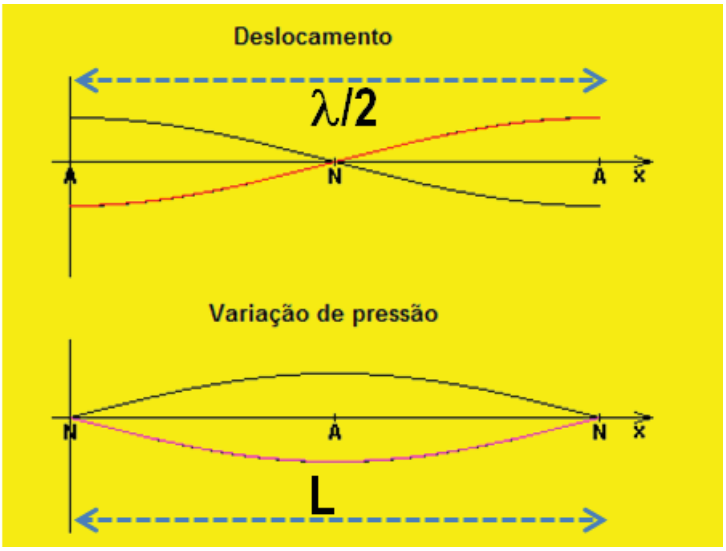


Fig. 12[7] *– Modo fundamental para tubo aberto nas duas extremidades*[(14)].

De acordo com a Fig. 12 nota-se que o comprimento de onda do modo fundamental[8] é dado por $\lambda = 2L$. Os demais modos de vibração apresentam comprimentos de onda dados por $\lambda = 2L/n$, ($n = 1, 2, 3...$), sendo n chamado de número do harmônico.

Sabendo que a velocidade de propagação de uma onda pode ser obtida pela equação (17), na qual "λ" corresponde ao comprimento de onda e "f" a frequência de vibração:

$$v = f\ \lambda \tag{17}$$

Isso implica que as frequências para os modos de vibração do tubo com duas extremidades abertas podem ser obtidas a partir da equação (18):

$$\text{fn} = \text{v} / \lambda = \frac{n\,v}{2\,L} \tag{18}$$

na qual n corresponde a um numero inteiro 1, 2, 3, ...

[7] Figura construída tomando por base o simulador disponível em: <http://www.walter-fendt.de/ph14br/stlwaves_br.htm>. Acesso em: 13 ago. 2013.

[8] Modo fundamental corresponde ao maior comprimento de onda possível e portando a menor frequência.

Vale ressaltar que, para n = 1, sempre teremos o modo fundamental de vibração do tubo.

II. Procedimentos: utilizando o Arduino

Para que seja possível utilizar o Arduino como placa de aquisição de dados no Tubo de Kundt, devemos antes de qualquer coisa construir um amplificador de áudio. Este amplificador deverá ampliar a tensão observada nos terminais de um microfone de eletreto para tensões perceptíveis nas entradas analógicas do Arduino. Neste trabalho, utilizamos o CI LM 386[15], muito utilizado em projetos desenvolvidos com o Arduino. O amplificador da Fig. 13 foi, então, montado em uma placa de fenolite (Fig. 14a).

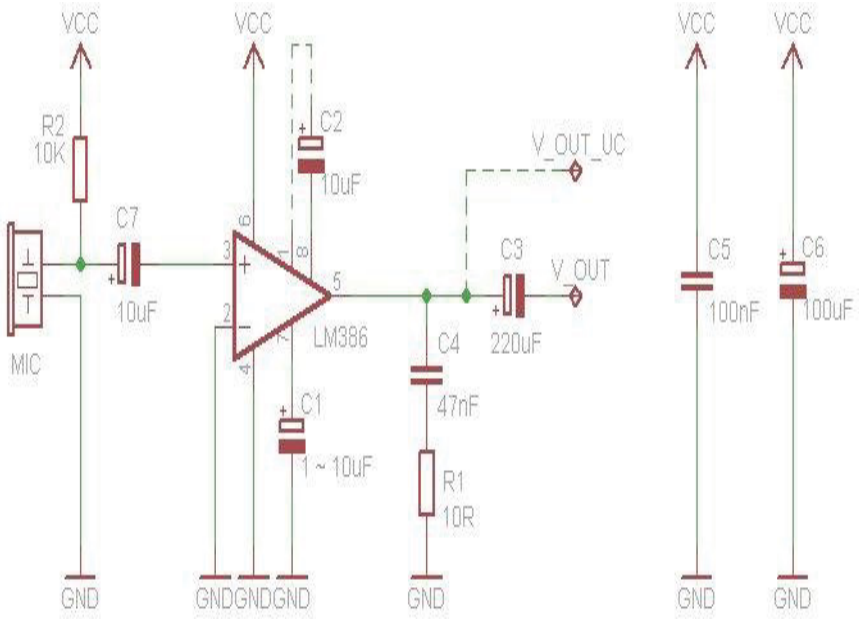


Fig. 13 – Esquema eletrônico do circuito amplificador utilizando CI LM386[16].

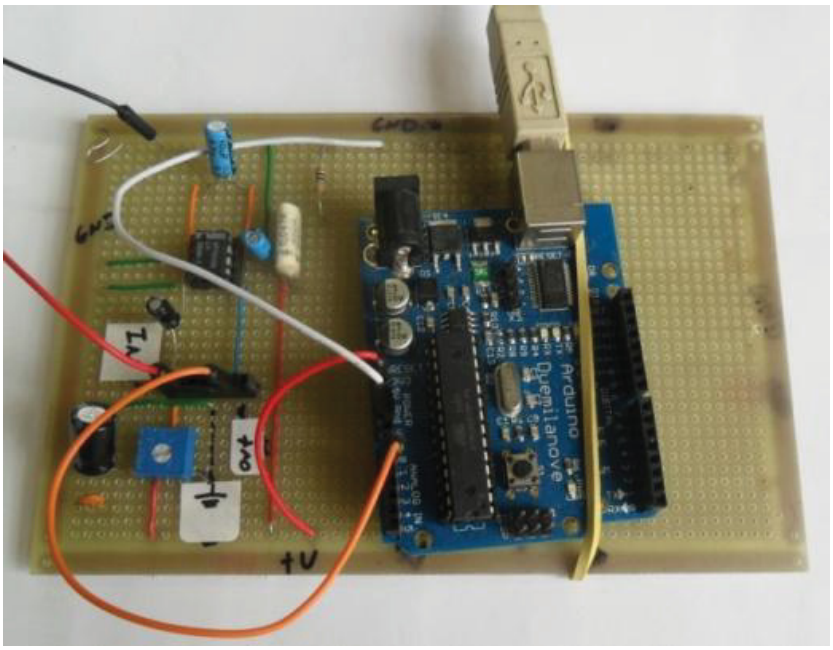


Fig. 14a – Circuito amplificador em placa fenolite.

O circuito foi acoplado a um microfone de eletreto (Fig. 14 a e b) com cabo de aproximadamente 1 metro, para ser introduzido por uma sonda ao longo do tubo contendo uma escala graduada em sua superfície externa.

Fig. 14b – Microfone associado com o circuito amplificador.

Diferentemente do projeto original, que utilizava pó de cortiça no seu interior e que mostrava as regiões de ventre e nós de deslocamento, procuramos medir a intensidade sonora ao longo do tubo com a movimentação de um microfone e analisada pela entrada analógica do Arduino (Anexo A) identificando assim os pontos de nó e antinó de pressão, confrontando os resultados com os previstos pela teoria. Utilizamos dois tubos um de 50 cm e outro de 1m c de comprimento, ambos abertos nas duas extremidades. Pontos em que se tem um nó de pressão apresentam menor intensidade sonora e pontos em que se tem antinós (ou ventres) de pressão temos máxima intensidade sonora.

Para visualização da intensidade sonora, utilizamos o Software Simplot®[17] que interpreta os dados adquiridos pela porta de aquisição analógica do Arduino, ligada diretamente a saída do amplificador, e após o processamento possibilita a representação gráfica em tempo real (Fig. 15).

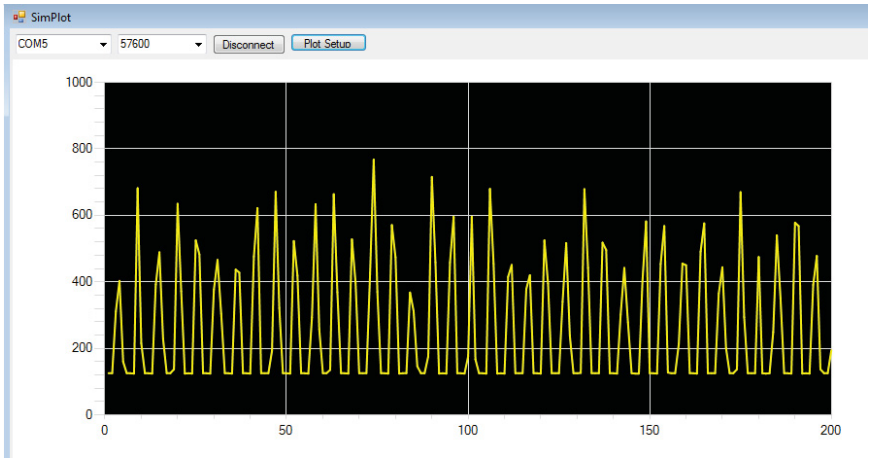


Fig. 15 – Simplot® em funcionamento, captando variação de sinal do Arduino em tempo real.

As diferentes frequências geradas no interior do tubo, são produzidas na saída de som do computador com utilização do software gratuito AudioSweepGen®[18] (Fig. 16).

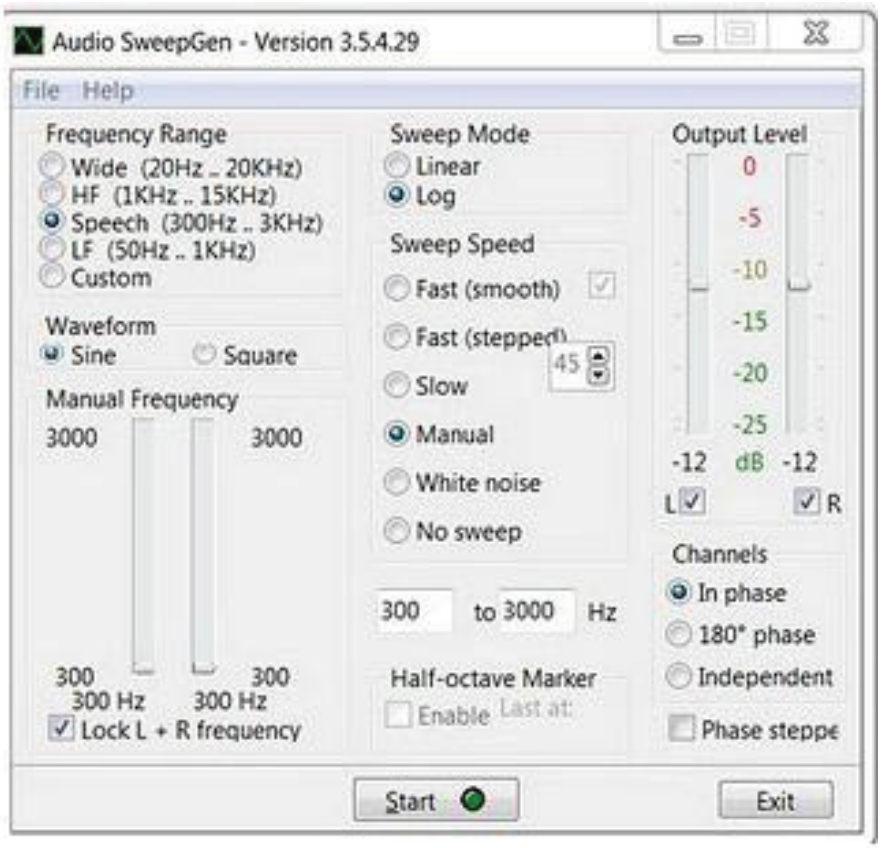


Fig. 16 – Janela de comando do Gerador de áudio Audio Sweepgen.

O tubo é então posicionado diante da caixa de som conectada ao PC (ou *notebook*) e diferentes valores de frequência podem ser fixados, podendo gerar ondas estacionarias no interior do tubo.

A Fig. 17 mostra o arranjo experimental utilizado. Nesta montagem podemos ver o amplificador operacional LM 386, o Arduino, o microfone, o tubo (transparente) de comprimento da ordem de 1 metro com uma fita métrica colada em sua superfície externa, a caixa de som e o *notebook*.

Fig. 17 – Montagem completa do Tubo de Kundt adaptado.

III. Resultados

Para um tubo de extremidades abertas e seguindo a disposição dos equipamentos da montagem da Fig. 17 deslocamos o microfone no seu interior com auxílio da sonda como indica a Fig. 18. A fita métrica colada na superfície externa do tubo permite determinar a posição x em que se encontra o microfone.

Fig. 18 – Sonda acoplada ao microfone, inseridos no interior do tubo.

Com um tubo de 1 metro de comprimento, temperatura de 24ºC e diâmetro interno de 2,3 cm, espera-se uma frequência de 171 Hz para harmônico fundamental e um comprimento de onda igual a 2,02 m (eq. 16).

Com os valores calculados de comprimento de onda (λ) e frequência (f), posicionaremos o microfone nos pontos de nó e antinó ao longo do tubo e verificaremos em tempo real a amplitude da intensidade sonora. Vale salientar que, devido à forma de aquisição, utilizamos a resolução do Arduino de 10 bits (1024 canais).

As Fig. 19 e 20 indicam a amplitude do sinal observado no Simplot, tanto para os pontos de máxima pressão (antinós) quanto para os pontos de nós de pressão para o tubo de 100 cm de comprimento.

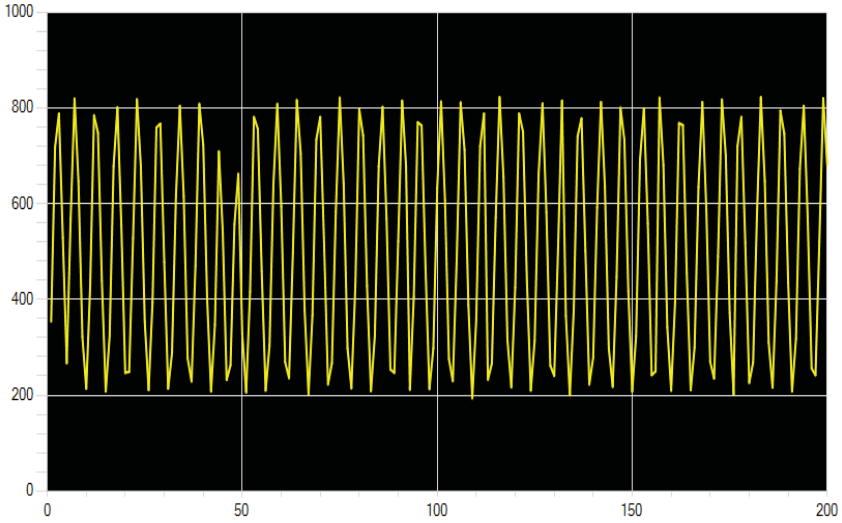


Fig. 19 – Intensidade sonora máxima no interior de um tubo de 1 metro observada com o microfone na posição 0,49m.

A Fig. 20 é, no nosso entender, uma das imagens mais extraordinárias deste experimento, pois apesar do microfone praticamente encostar-se ao alto-falante a intensidade sonora apresenta o seu valor mínimo. Naturalmente ocorre a pergunta: como é possível nos aproximarmos de uma fonte sonora e observar uma redução na intensidade do som? Esta constatação e no mínimo intrigante e fere a logica natural e ao senso comum. Para compreender este fenômeno é preciso estudar ondas estacionárias e ressonância, é preciso entender os modos de vibração característicos de tubos.

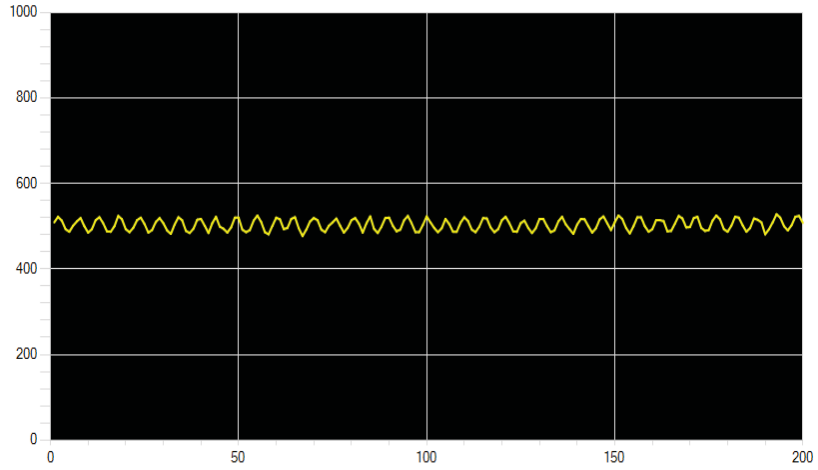


Fig. 20 – Intensidade sonora mínima observada no interior de um tubo de 1 m, com o microfone na posição 0,01m e 0,99m (microfone bem próximo ao alto falante e na extremidade oposta).

Utilizando as equações (13) e (16), calculamos os valores esperados para cada ponto de antinós (A) e nós (N) de pressão sonora à medida que variamos a posição do microfone ao longo dos tubos de 50 e 100 cm para os diferentes modos de vibração e lançamos estes valores nas tabelas 1 e 3, respectivamente. Nestas tabelas, indicamos por máximo (Max.) os pontos em que se espera observar maior amplitude de sinal no microfone e por mínimo (Min.) os pontos em que se esperamos observar pontos de menor amplitude de sinal.

Observe que, nestas tabelas, reservamos uma linha imediatamente abaixo do valor da varredura x para fornecer o comprimento de onda associado à onda estacionária produzida. Com este procedimento e considerando a velocidade do som (esperada para a temperatura local), determinamos a frequência que deve ser fixada no gerador de áudio para se observar cada um dos modos de vibração do tubo. Estes valores de frequência são lançados em uma nova tabela que possibilitará, através da varredura x do microfone ao longo do tubo, verificar o comprimento de onda experimental para cada modo de vibração.

As tabelas 2 e 4 fornecem, portanto, os valores da variável x obtidos experimentalmente para os pontos de máxima e mínima intensidade sonora em cada frequência previamente fixada no alto-falante, localizado em uma das extremidades do tubo de comprimento igual a 0,50 e 1,0m. A partir da variável x, é possível

determinar o comprimento de onda da onda associada a cada modo de vibração e, com a equação 17, obtém-se o valor da velocidade do som (lembrando que o valor de frequência fixada no gerador de áudio foi obtido nas tabelas 1 e 3).

Tabela 1 – Intensidade Sonora ao longo do tubo no harmônico fundamental (1º), 2º, 3º, 4º e 5º harmônicos, com tubo de 0,50m ***(valores esperados teoricamente).***

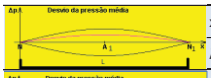
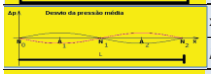
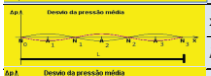
Pressão vs X	*X(m) , λ (m)*	*1° Mín.*	*1° Max.*	*2° Mín.*	*2° Max.*	*3° Mín.*	*3° Max.*	*4° Mín.*	*4° Max.*	*5° Mín.*
		N0	*A1*	*N1*	*A2*	*N2*	*A3*	*N3*	*A4*	*N4*
	x (posição)	-	0,26	0,51						
	λ	-	1,03	1,03						
	x (posição)	-	0,13	0,26	0,39	0,51				
	λ	-	0,51	0,51	0,51	0,51				
	x (posição)	-	0,09	0,17	0,26	0,34	0,43	0,51		
	λ	-	0,34	0,34	0,34	0,34	0,34	0,34		
	x (posição)	-	0,06	0,13	0,19	0,26	0,32	0,39	0,45	0,51
	λ	-	0,26	0,26	0,26	0,26	0,26	0,26	0,26	0,26

Tabela 2 – Intensidade Sonora ao longo do tubo no harmônico fundamental (1º), 2º, 3º, 4º e 5º harmônicos, com tubo de 0,50m (**valores obtidos experimentalmente**).

Harmônico	*Frequência (Hz)*	*X(m) L(m) v(m/s)*	*1° Min.*	*1° Max.*	*2° Min.*	*2° Max.*	*3° Min.*	*3° Max.*	*4° Min.*	*4° Max.*	*5° Min.*
1	334,73	x	-	0,27	0,50						
		λ	-	1,06	1,00						
		v	-	354,82	334,73						
2	669,46	x	0,01	0,13	0,26	0,37	0,51				
		λ	-	0,52	0,51	0,49	0,51				
		v	-	348,12	341,43	330,27	341,43				
3	1004,19	x	0,01	0,10	0,17	0,27	0,35	0,44	0,51		
		λ	-	0,40	0,33	0,36	0,35	0,35	0,34		
		v	-	401,68	331,38	361,51	351,47	353,48	341,43		
4	1338,93	x	0,01	0,07	0,14	0,20	0,27	0,33	0,40	0,46	0,51
		λ	-	0,28	0,28	0,27	0,27	0,26	0,26	0,26	0,26
		v	-	374,90	374,90	357,05	354,82	353,48	352,58	351,95	341,43

Temperatura		*23 °*
Vsom (esperado) =	*(330,4 + 0,59 x 23)*	*343,97 m/s*
R =	*2,3 cm*	*0,0230 m*
Lef = L + 1,2R	*0,50 + 1,2*(0,023/2)*	*0,51 m*

Velocidade do Som obtida para o tudo L = 50 cm:
Valor médio : 352,6 m/s
Desvio padrão : 16,7 m/s
Desvio padrão da média : 3,7 m/s
Valor obtido : (352,6 ± 3,7) m/s

O valor obtido incorpora o valor esperado com um grau de significância de 95%, com 2σ.

IV. Filtros sonoros

De acordo com os resultados anteriores, é possível notar que um tubo pode funcionar como um filtro sonoro, selecionando e/ou priorizando alguns valores de frequência, que lhe são características. Isso significa que, se inserirmos, por exemplo, um ruído branco[9] em um tubo de um dado comprimento L, teremos uma amplificação dos sinais com frequências coincidentes àquelas dos seus respectivos harmônicos.

No entanto, é importante notar que teremos maior ou menor intensidade sonora dependendo da posição em que o microfone se localiza, as posições dos nós de pressão ocupam posições bem determinadas para cada harmônico.

Para possibilitar a visualização das frequências ressonantes e suas regiões de antinós de pressão utilizamos um código de programação desenvolvido no Processing (Anexo B) que possibilita a visualização das componentes de frequência do sinal em tempo real.

Vale ressaltar que, para a visualização das componentes de frequência através do Processing, é preciso baixar algumas bibliotecas desenvolvidas pela comunidade de usuários e disponíveis na internet[(19)].

A Fig. 21 mostra as componentes de frequência observadas apenas na saída da caixa de som antes de acoplarmos o tubo de 1m de comprimento, a intensidade sonora apresenta praticamente a mesma todo o intervalo de frequência observado.

A Fig. 22 mostra o espectro do som observado no interior do tubo (1m) na posição central (0,50m). Observa-se maior intensidade sonora na faixa de frequência de 150 a 300 Hz. De acordo com a programação disponível no Anexo B, pode-se ver que cada coluna representa a faixa de aproximadamente 150 Hz. Convém lembrar que, na região central do tubo de extremidades abertas, teremos pontos de máxima intensidade sonora para os harmônicos impares (ver tabela 4).

[9] Quando se produz um ruído branco não há emissão de som com frequência preferencial, ou seja, há uma distribuição continua de distribuição de frequências. Na pratica é como um chiado intermitente.

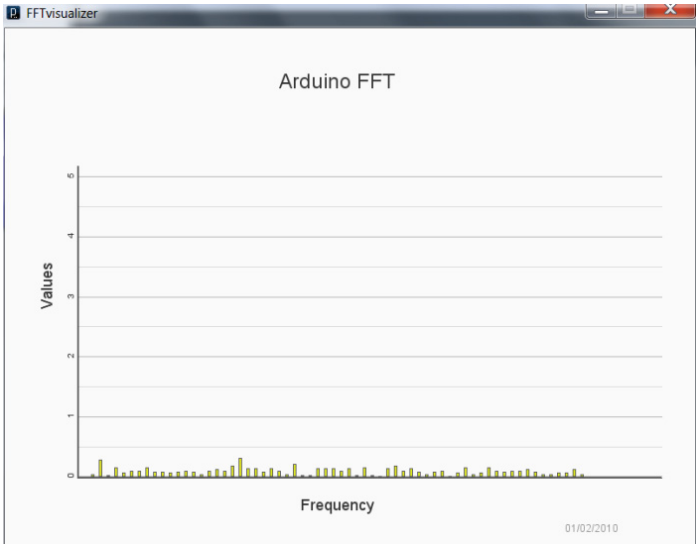


Fig. 21 – Ruído Branco captado pelo microfone fora do tubo.

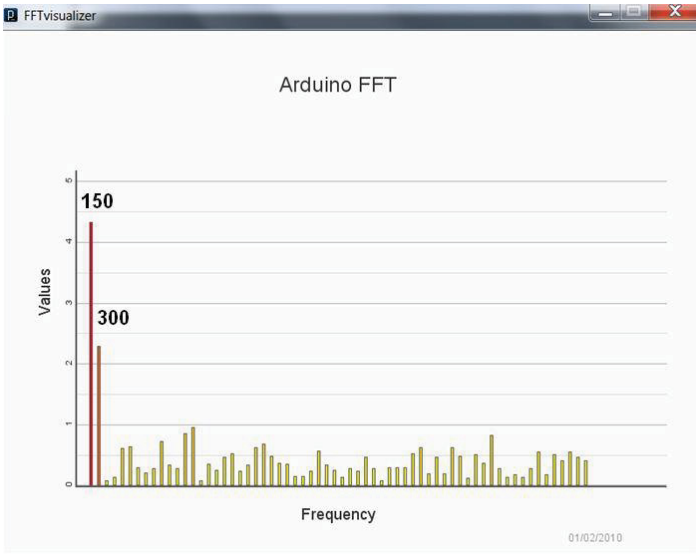


Fig. 22 – *Ruído Branco* captado pelo microfone dentro do tubo.

V. Análises e considerações finais

Com a adaptação de um experimento clássico de acústica às novas tecnologias disponíveis, foi possível explorar diversos conceitos de acústica e características ondulatórias. As possibilidades que os recursos computacionais oferecem permitem que sejam explorados e aprofundados os conceitos de acordo com a necessidade didática. Foi isso que se notou durante a realização deste trabalho, em que cada nova atividade proposta se revelavam novas possibilidades, tais como descoberta de novos softwares de aquisição, como, por exemplo, o Simplot, ou ainda funções e bibliotecas para uso na plataforma de desenvolvimento do Arduino. Estas descobertas foram conduzindo a melhorias técnicas e didáticas até então desconhecidas. Como os recursos propostos neste trabalho apresentam características de código aberto em hardware e software à contribuição da comunidade de desenvolvedores e criatividade (incluindo os alunos e professores que, no futuro, vierem a utilizar os recursos aqui apresentados), novas descobertas poderão ainda ser realizadas e, com poucas mudanças em linhas de programação e circuitos, a um custo relativamente baixo, novas oportunidades didáticas se abrem, transformando o aprendizado em algo mais dinâmico e orgânico.

Os próximos passos do projeto incluem a revisão da programação do Arduino para que as intensidades sonoras e análise de frequências sejam avaliadas automaticamente. Pretendemos controlar o movimento do microfone no interior do tubo através de um motor de passo de tal modo que será possível, por exemplo, observar um gráfico de intensidade sonora versus posição diretamente na tela do PC (ou *notebook*), o que certamente tornará o experimento mais interativo e didático. Além disso, pretendemos associar um mostrador LCD ao Arduino, de modo que o experimento possa ser realizado sem a necessidade de acoplamento de computadores.

Referências bibliográficas

1. ARDUINO**. Introdução ao Arduino**. Disponível em: <http://arduino.cc/en/>. Acesso em: 12 ago. 2013.

2. CAVALCANTE, M. A.; TAVOLARO, C. F. C. Física com Arduino para iniciantes. São Paulo: **Física na Escola**, v. 33, n. 4, 2011. Disponível em: <http://www.sbfisica.org.br/rbef/pdf/334503.pdf>. Acesso em: 13 ago. 2013.

3. PROCESSING. Software Processing. Disponível em: <http://processing.org/>. Acesso em: 12 ago. 2013.

4. JARDIM, M. I. A.; ERROBIDART, N. C. G.; GOBARA, S. T. Levantamento dos trabalhos em Física que investigaram Ondas Sonoras**.** In: ENCONTRO DE PESQUISA EM ENSINO DE FÍSICA, XI, 2008, Curitiba. Disponível em: <http://www.sbf1.sbfisica.org.br/eventos/epef/xi/sys/resumos/T0186-1.pdf>. Acesso em: 12 ago. 2013.

5. SOUZA, A. R. **Experimentos em Ondas Mecânicas**. 2011. Dissertação (Mestrado) - Programa de Pós-graduação em Ensino de Física, Instituto de Física, Universidade Federal do Rio de Janeiro. Disponível em: <http://www.if.ufrj.br/~pef/producao_academica/dissertacoes/2011_Anderson_Souza/dissertacao_Anderson_Souza.pdf>. Acesso em: 12 ago. 2013.

6. KUNDT. A. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1002/andp.18662030402/abstract>. Acesso em: 12 ago. 2013.

7. ESQUEMA DO TUBO DE KUNDT (Fig. 3). Imagem disponível em: <http://www.sc.ehu.es/sbweb/fisica/ondas/acustica/kundt/kundt.htm>. Acesso em: 13 ago. 2013.

8. ACÚSTICA. Elementos de Acústica. Disponível em: <http://www.fisica.net/ondulatoria/elementos_de_acustica.pdf>. Acesso em: 13 ago. 2013.

9. CAVALCANTE, M. A.; TAVOLARO, C. F. C. Medir a velocidade do som pode ser rápido e fácil. São Paulo: **Física na Escola**, v. 4, n. 1, 2003. Disponível em: <http://www.sbfisica.org.br/fne/Vol4/Num1/a10.pdf>. Acesso em: 13 ago. 2013.

10. Figura disponível em: <http://www.infoescola.com/fisica/ondas-longitudinais/>. Acesso em: 13 ago. 2013.

11. SOUZA, A, R.; AGUIAR, A. C Pressão e deslocamento nas ondas sonoras. In: SIMPÓSIO NACIONAL DE ENSINO DE FÍSICA, XIX, 2011, Manaus, AM. Disponível em: <http://www.if.ufrj.br/~carlos/artigos/snef2011som.pdf>. Acesso em: 13 ago. 2013.

12. NOVAS TECNOLOGIAS. In: SIMPÓSIO NACIONAL DE ENSINO DE FÍSICA, XVII. Disponível em: <http://xviiisnefnovastecnologias.blogspot.com/2010/06/applet-ondas-em-tubos-sonoros.html>. Acesso em: 13 ago. 2013.

13. ONDAS ESTACIONÁRIAS (Fig. 8). Disponível em: <http://www.feiradeciencias.com.br/sala10/10_15.asp>. Acesso em: 13 ago. 2013.

14. BARROS, I. R.; BORDUQUI, H. G.; TURRA, A. E. **Ondas sonoras estacionárias em tubos – Tubo de Kundt**. Faculdade de Engenharia Mecânica, UNESP. Disponível em: <http://prope.unesp.br/xxi_cic/27_34937466879.pdf>. Acesso em: 13 ago. 2013.

15. DATASHEET CI LM386. Circuito Integrado. Disponível em: <http://www.national.com/ds/LM/LM386.pdf>. Acesso em: 13 ago. 2013.

16. ESQUEMA ELETRICO DO AMPLIFICADOR LM386: Disponível em: <http://lowvoltage.wordpress.com/2011/05/15/lm386-mic-amp/>. Acesso em: 13 ago. 2013.

17. SIMPLOT: software livre para a observação em tempo real nas entradas analógicas do Arduino. Disponível em: <http://sray.med.som.jhmi.edu/SCRoftware/simplot/>. Acesso em: 13 ago. 2013.

18. SOFTWARE DISPONÍVEL LIVREMENTE NA WEB; SWEEPGEN. Audio Sweepgen. Disponível em: <http://xviiisnefnovastecnologias.blogspot.com/2011/10/gerador-de-sinais.html>. Acesso em: 13 ago. 2013.

19. TUTORIAL PARA OBSERVAÇÃO DAS COMPONENTES DE FREQUENCIA SONORA COM O ARDUINO E PROCESSING. Disponível em: <http://labduino.blogspot.com.br/2011/10/fft-com-o-arduino-analisando-o-espectro.html>. Acesso em: 13 ago. 2013.

Anexo A

/* Programação para observação do espectro sonoro utilizando Arduino®

Este programa deve ser copiado na IDE do Arduino
O microfone é conectado a entrada analógica A0 e biblioteca utilizada é baseada

em http://elm-chan.org/works/akilcd/report_e.html

Bibliotecas disponíveis em
http://code.google.com/p/neuroelec/downloads/detail?name=ffft_Library.zip&can
=2&q=*/

```
#include <stdint.h>
#include <ffft.h>
#define  IR_AUDIO  0 // ADC channel to capture
volatile  byte  position = 0;
volatile  long  zero = 0;
int16_t capture[FFT_N];                    /* Wave captureing buffer */
complex_t bfly_buff[FFT_N];                /* FFT buffer */
uint16_t spektrum[FFT_N/2];                /* Spectrum output buffer */
void setup()
{
  Serial.begin(57600);
  adcInit();
  adcCalb();
  establishContact();  // send a byte to establish contact until Processing respon
}
void loop()
{
  if (position == FFT_N)
  {
    fft_input(capture, bfly_buff);
    fft_execute(bfly_buff);
    fft_output(bfly_buff, spektrum);
    for (byte i = 0; i < 64; i++){
      Serial.print(spektrum[i],BYTE);
    }
   position = 0;
  }
}
void establishContact() {
 while (Serial.available() <= 0) {
      Serial.print('A', BYTE);   // send a capital A
      delay(300);
```

```
  }
}
// free running ADC fills capture buffer
ISR(ADC_vect)
{
  if (position >= FFT_N)
    return;
  capture[position] = ADC + zero;
  if (capture[position] == -1 || capture[position] == 1)
    capture[position] = 0;
  position++;
}
void adcInit(){
  /*  REFS0 : VCC use as a ref, IR_AUDIO : channel selection, ADEN : ADC
Enable, ADSC : ADC Start, ADATE : ADC Auto Trigger Enable, ADIE : ADC
Interrupt Enable,  ADPS : ADC Prescaler  */
  // free running ADC mode, f = ( 16MHz / prescaler ) / 13 cycles per conversion
  ADMUX = _BV(REFS0) | IR_AUDIO; // | _BV(ADLAR);
//  ADCSRA = _BV(ADSC) | _BV(ADEN) | _BV(ADATE) | _BV(ADIE) |
_BV(ADPS2) | _BV(ADPS1) //prescaler 64 : 19231 Hz - 300Hz per 64 divisions
  ADCSRA = _BV(ADSC) | _BV(ADEN) | _BV(ADATE) | _BV(ADIE) |
_BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0); // prescaler 128 : 9615 Hz - 150 Hz
per 64 divisions, better for most music
  sei();
}
void adcCalb(){
  Serial.println("Start to calc zero");
  long midl = 0;
  // get 2 meashurment at 2 sec
  // on ADC input must be NO SIGNAL!!!
  for (byte i = 0; i < 2; i++)
  {
    position = 0;
    delay(100);
    midl += capture[0];
    delay(900);
  }
  zero = -midl/2;
```

```
  Serial.println("Done.");
}
```

Anexo B

/*Programação que deve rodar no processing */

```
// Feel Free to edit these variables ////////////////////////////
String  xLabel = "Frequency";
String  yLabel = "Values";
String  Heading = "Arduino FFT";
String  URL = "01/02/2010";
float Vcc = 255.0;    // the measured voltage of your usb
int NumOfVertDivisions=5;      // dark gray
int NumOfVertSubDivisions=10;  // light gray
int NumOfBars=64;    // you can choose the number of bars, but it can cause issues
                     // since you should change what the arduino sends
// if these are changed, backgroung image has problems
// a plain background solves the problem
int ScreenWidth = 800, ScreenHeight=600;
//////////////////////////////////////////////////////////
//  Serial port stuff ////////////////////////
import processing.serial.*;
Serial myPort;
boolean firstContact = false;
int[] serialInArray = new int[NumOfBars];
int serialCount = 0;
/////////////////////////////////////////////
int LeftMargin=100;
int RightMArgin=80;
int TextGap=50;
int GraphYposition=80;
float BarPercent = 0.4;
int value;
PFont font;
PImage bg;
int temp;
float yRatio = 0.58;
int BarGap, BarWidth, DivisounsWidth;
```

```
int[] bars = new int[NumOfBars];
void setup(){
 bg = loadImage("BG.jpg");
 /// NB SETTINGS ////////////////////////////////////////////////////////
 myPort = new Serial(this, "COM13", 57600);  // atenção para esta linha você deve
selecionar a porta COM que                  //o arduino está utilizando
 ////////////////////////////////////////////////////////////////////////
 DivisounsWidth = (ScreenWidth-LeftMargin-RightMArgin)/(NumOfBars);
 BarWidth = int(BarPercent*float(DivisounsWidth));
 BarGap = DivisounsWidth - BarWidth;
 size(ScreenWidth,ScreenHeight);
 font = createFont("Arial",12);
  textAlign(CENTER);
 textFont(font);
}
void draw(){
//  background(bg);     // My one used a background image, I've
 background(250);      // commented it out and put a plain colour
 //  Headings();          // Displays bar width, Bar gap or any variable.
 Axis();
 Labels();
 PrintBars();
//  Line();
//  Dots();
}
// Send Recieve data //
void serialEvent(Serial myPort) {
 // read a byte from the serial port:
 int inByte = myPort.read();
 if (firstContact == false) {
  if (inByte == 'A') {
   myPort.clear();          // clear the serial port buffer
   firstContact = true;     // you've had first contact from the microcontroller
   myPort.write('A');       // ask for more
  }
 }
 else {
  // Add the latest byte from the serial port to array:
```

```
   serialInArray[serialCount] = inByte;
   serialCount++;
   // If we have 6 bytes:
   if (serialCount > NumOfBars -1 ) {
for (int x=0;x<NumOfBars;x++){
 bars[x] = int (yRatio*(ScreenHeight)*(serialInArray[x]/256.0));
}
     // Send a capital A to request new sensor readings:
     myPort.write('A');
     // Reset serialCount:
     serialCount = 0;
   }
  }
}
/////// Display any variables for testing here//////////////
void Headings(){
 fill(0 );
 text("BarWidth",50,TextGap );
 text("BarGap",250,TextGap );
 text("DivisounsWidth",450,TextGap );
 text(BarWidth,100,TextGap );
 text(BarGap,300,TextGap );
 text(DivisounsWidth,520,TextGap );
}
void PrintBars(){
 int c=0;
 for (int i=0;i<NumOfBars;i++){
     fill((0xe4+c),(255-bars[i]+c),(0x1a+c));
   stroke(90);
   rect(i*DivisounsWidth+LeftMargin,   ScreenHeight-GraphYposition,
BarWidth,   -bars[i]);
   fill(0x2e,0x2a,0x2a);
//    text(float(bars[i])/(yRatio*(ScreenHeight))*Vcc,
i*DivisounsWidth+LeftMargin+BarWidth/2,   ScreenHeight-bars[i]-5-
GraphYposition );
//    text("A",   i*DivisounsWidth+LeftMargin+BarWidth/2 -5,   ScreenHeight-
GraphYposition+20 );
```

```
//    text(i,   i*DivisounsWidth+LeftMargin+BarWidth/2 +5,   ScreenHeight-
GraphYposition+20 );
  }
}
void Axis(){
  strokeWeight(1);
  stroke(220);
  for(float x=0;x<=NumOfVertSubDivisions;x++){
    int bars=(ScreenHeight-GraphYposition)-
int(yRatio*(ScreenHeight)*(x/NumOfVertSubDivisions));
    line(LeftMargin-15,bars,ScreenWidth-RightMArgin-DivisounsWidth+50,bars);
  }
  strokeWeight(1);
  stroke(180);
  for(float x=0;x<=NumOfVertDivisions;x++){
    int bars=(ScreenHeight-GraphYposition)-
int(yRatio*(ScreenHeight)*(x/NumOfVertDivisions));
    line(LeftMargin-15,bars,ScreenWidth-RightMArgin-DivisounsWidth+50,bars);
  }
  strokeWeight(2);
  stroke(90);
  line(LeftMargin-15, ScreenHeight-GraphYposition+2, ScreenWidth-
RightMArgin-DivisounsWidth+50, ScreenHeight-GraphYposition+2);
  line(LeftMargin-15,ScreenHeight-GraphYposition+2,LeftMargin-
15,GraphYposition+80);
  strokeWeight(1);
}

void Labels(){
  textFont(font,18);
  fill(50);
  rotate(radians(-90));
  text(yLabel,-ScreenHeight/2,LeftMargin-45);
  textFont(font,10);
  for(float x=0;x<=NumOfVertDivisions;x++){
    int bars=(ScreenHeight-GraphYposition)-
int(yRatio*(ScreenHeight)*(x/NumOfVertDivisions));
    text(round(x),-bars,LeftMargin-20);
```

```
  }
  textFont(font,18);
  rotate(radians(90));
  text(xLabel,LeftMargin+(ScreenWidth-LeftMargin-RightMArgin-
50)/2,ScreenHeight-GraphYposition+40);
  textFont(font,24);
  fill(50);
  text(Heading,LeftMargin+(ScreenWidth-LeftMargin-RightMArgin-50)/2,70);
  textFont(font);
  fill(150);
  text(URL,ScreenWidth-RightMArgin-40,ScreenHeight-15);
  textFont(font);
}
```

Anexo C

Programação que deve "rodar" na IDE do Arduino

```
/*
SimPlot Demo
Samples Analog Input and sends them over serial port to be plotted in SimPlot.
This Demo software uses the default serial port "Serial".
Upto to 4 channels of data can be plotted.
For details of SimPlot go to
www.negtronics.com/simplot
 */
void setup()
{
  Serial.begin(57600);
}
int buffer[20]; //Buffer needed to store data packet for transmission
int data1;
int data2;
int data3;
int data4;
void loop()
{
  //Read Analog channels. You can connect accelerometer, gyro, temperature sen-
sor etc to these channels
```

```
  data1 = analogRead(0);
  data2 = analogRead(1);
  data3 = analogRead(2);
  data4 = analogRead(3);
  //You can plot upto 4 channels of data. Uncomment only one of the options below
   plot(data1,data2,data3,data4);   //Plots 4 channels of data
//  plot(data1,data2,data3);      //Plots 3 channels of data
//  plot(data1,data2);            //Plots 2 channels of data
//  plot(data1);                  //Plots 1 channel of data
  delay(10); //Read and plot analog inputs every 10ms.
}
//Function that takes 4 integer values and generates a packet to be sent to SimPlot.
void plot(int data1, int data2, int data3, int data4)
{
  int pktSize;
  buffer[0] = 0xCDAB;             //SimPlot packet header. Indicates start of data
packet
  buffer[1] = 4*sizeof(int);      //Size of data in bytes. Does not include the header
and size fields
  buffer[2] = data1;
  buffer[3] = data2;
  buffer[4] = data3;
  buffer[5] = data4;

  pktSize = 2 + 2 + (4*sizeof(int)); //Header bytes + size field bytes + data
  //IMPORTANT: Change to serial port that is connected to PC
  Serial.write((uint8_t * )buffer, pktSize);
}
//Function that takes 3 integer values and generates a packet to be sent to SimPlot.
void plot(int data1, int data2, int data3)
{
  int pktSize;
  buffer[0] = 0xCDAB;             //SimPlot packet header. Indicates start of data
packet
  buffer[1] = 3*sizeof(int);      //Size of data in bytes. Does not include the header
and size fields
  buffer[2] = data1;
  buffer[3] = data2;
```

```
  buffer[4] = data3;
  pktSize = 2 + 2 + (3*sizeof(int)); //Header bytes + size field bytes + data
  //IMPORTANT: Change to serial port that is connected to PC
  Serial.write((uint8_t * )buffer, pktSize);
}
//Function that takes 2 integer values and generates a packet to be sent to SimPlot.
void plot(int data1, int data2)
{
  int pktSize;
  buffer[0] = 0xCDAB;             //SimPlot packet header. Indicates start of data
packet
  buffer[1] = 2*sizeof(int);      //Size of data in bytes. Does not include the header
and size fields
  buffer[2] = data1;
  buffer[3] = data2;
  pktSize = 2 + 2 + (2*sizeof(int)); //Header bytes + size field bytes + data
  //IMPORTANT: Change to serial port that is connected to PC
  Serial.write((uint8_t * )buffer, pktSize);
}
//Function that takes 1 integer value and generates a packet to be sent to SimPlot.
void plot(int data1)
{
  int pktSize;
  buffer[0] = 0xCDAB;             //SimPlot packet header. Indicates start of data
packet
  buffer[1] = 1*sizeof(int);      //Size of data in bytes. Does not include the header
and size fields
  buffer[2] = data1;
  pktSize = 2 + 2 + (1*sizeof(int)); //Header bytes + size field bytes + data
    //IMPORTANT: Change to serial port that is connected to PC
  Serial.write((uint8_t * )buffer, pktSize);
}
```