

Sergio Fernando Nunes Junior

Projeto de Bateria Eletrônica em Arduino

Guaratinguetá - SP
2017

Sergio Fernando Nunes Junior

Projeto de Bateria Eletrônica em Arduino

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em engenharia elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em engenharia elétrica.

Orientador: Dr. Leonardo Mesquita

Guaratinguetá - SP
2017

N972p Nunes Junior, Sergio Fernando
Projeto de bateria eletrônica em Arduino / Sergio Fernando Nunes
Junior – Guaratinguetá, 2017.
51 f : il.
Bibliografia: f. 43-44

Trabalho de Graduação em Engenharia Elétrica – Universidade
Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2017.
Orientador: Prof. Dr. Leonardo Mesquita

1. Baterias elétricas 2. Arduino (Controlador programável) I. Título

CDU 621.352

Ana Cristina Figueiredo Loureiro
Ana Cristina Figueiredo Loureiro

Bibliotecária

CRB 8/7094

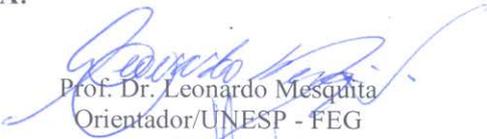
Sergio Fernando Nunes Junior

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO
PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE
"GRADUADO EM ENGENHARIA ELETRICA"

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE
GRADUAÇÃO EM ENGENHARIA ELETRICA

Prof. Dr. Leonardo Mesquita
Coordenador

BANCA EXAMINADORA:


Prof. Dr. Leonardo Mesquita
Orientador/UNESP - FEG


Eng. José Marcelo de Assis Wendling Junior
CTIG - UNESP


Prof. Dr. Fernando Ribeiro Filadelfo
UNESP - FEG

Dezembro/2017

dedico este trabalho
de modo especial, à minha família

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, fonte da vida e da graça. Agradeço pela minha vida, minha inteligência, minha família e meus amigos,

ao meu orientador, *Prof. Dr. Leonardo Mesquita* que jamais deixou de me incentivar. Sem a sua orientação, dedicação e auxílio, o estudo aqui apresentado seria praticamente impossível.

aos meus pais *Sergio e Eliana*, que apesar das dificuldades enfrentadas, sempre incentivaram meus estudos.

às funcionárias da Biblioteca do Campus de Guaratinguetá pela dedicação, presteza e principalmente pela vontade de ajudar,

aos funcionários da Faculdade de Engenharia do Campos de Guaratinguetá pela dedicação e alegria no atendimento.

“Música é vida interior, e quem tem vida interior jamais
padecerá de solidão.”

Artur da Tavola

RESUMO

O trabalho consiste da especificação, desenvolvimento, projeto e implementação de uma bateria eletrônica de baixo custo utilizando como elemento central a plataforma eletrônica Arduino, baseado no microcontrolador ATmega2560, visando a disponibilização do projeto para fins de reprodução educacional tanto na área da eletrônica quanto na área musical.

Para desenvolver o projeto foi estudado formas de se montar a estrutura física da bateria, considerando piezoelétricos e captadores feitos com madeira e EVA. Foi estudado, também, diversos modos de se trabalhar com áudio em programação, contemplando os formatos MIDI, WAVE e as bibliotecas que trabalham com formatos de áudio.

PALAVRAS-CHAVE: Bateria eletrônica; Microcontrolador ATmega2560; Sintetizador eletrônico; MIDI; WAVE; Piezoelétrico.

ABSTRACT

The work consists of the specification, development, design and implementation of a low-cost electronic drum using the Arduino electronic platform, based on the ATmega2560 microcontroller, as a central element, aiming at making the project available for educational reproduction purposes both in electronics and in musical area.

To develop this project was studied ways to assemble the physical structure of the drum, considering piezoelectric and pickups made with wood and EVA. Several ways of working with audio in programming have also been studied including MIDI, WAVE and libraries that work with audio formats.

KEYWORDS: Electronic drum; Microcontroller ATmega2560; Electronic synthesizer; MIDI; WAVE; Piezoelectric.

LISTA DE ILUSTRAÇÕES

Figura 1 – Kit de bateria básico.....	15
Figura 2 – Kit de bateria acústica.....	16
Figura 3 – Kit de bateria eletrônica.....	17
Figura 4 – Forma canônica do arquivo WAVE.....	19
Figura 5 – Etapas da modulação PCM (código de pulso).....	21
Figura 6 – Tipos de mensagem MIDI.....	23
Figura 7 – Modelo do sensor piezoelétrico.....	26
Figura 8 – Placa ARDUINO MEGA.....	27
Figura 9 – Visão geral do captador.....	31
Figura 10 – Detalhe do furo central.....	32
Figura 11 – 3 arcos sedo a base, a estante central e o topo da bateria.....	33
Figura 12 – Arco central e detalhe dos conectores em t no cano de 90cm.....	34
Figura 13 – Detalhe de como é a disposição dos furos no suporte.....	34
Figura 14 – Montagem final da bateria.....	35
Figura 15 – Esquema da fiação passada no interior da armação.....	35
Figura 16 – Pedal do chimbal.....	36
Figura 17 – Circuito eletrônico da bateria.....	37

LISTA DE TABELAS

Tabela 1 – Material para a estrutura	32
Tabela 2 – Custos	40

SUMÁRIO

1	INTRODUÇÃO	12
1.1	MOTIVAÇÃO E OBJETIVOS	13
2	REVISÃO BIBLIOGRÁFICA	14
2.1	TIPOS DE BATERIA	14
2.1.1	Bateria Acústica	14
2.1.2	Bateria Eletrônica	16
2.2	AUDIO E PROGRAMAÇÃO	18
2.2.1	Formato <i>wave</i>	18
2.2.1.1	<i>RIFF (Resource Interchange File Format)</i>	18
2.2.1.2	<i>PCM (Pulse Code Modulation)</i>	20
2.2.2	MIDI	21
2.2.2.1	Transmissão dos dados	22
2.2.3	Pure Data	23
2.3	BIBLIOTECAS	23
2.3.1	SPI (<i>Serial Peripheral Interface</i>)	24
2.3.2	SD	24
2.3.3	TMRPCM	24
2.3.4	MIDI e MIDIUSB	25
2.4	HARDWARE	25
2.4.1	Sensor Piezoelétrico	25
2.4.2	Placa Arduino	26
3	DESENVOLVIMENTO	28
3.1	Possíveis métodos para a construção de uma bateria eletrônica	28
3.1.1	Utilizando Pure Data	28
3.1.2	Utilizando MIDI	29
3.1.3	Placa Arduino reproduzindo áudio	29
3.2	Método Utilizado	30
3.2.1	Captadores	31
3.2.2	Estrutura	32
3.2.3	Montagem na protoboard e na placa Arduino	36
3.2.4	Código	37

4	RESULTADOS	39
4.1	HARDWARE	39
4.2	SOFTWARE.....	39
4.3	SISTEMA GERAL.....	40
4.4	CUSTOS.....	40
5	CONCLUSÃO	41
6	REFERÊNCIAS	43

1 INTRODUÇÃO

Este trabalho tem como objetivo projetar e implementar uma bateria musical eletrônica, sendo feita de maneira simples e didática, para que possa ser reproduzida por amantes da música e da eletrônica, possibilitando que crianças sem condições financeiras de adquirir um instrumento convencional possam praticar e aprimorar sua técnica.

A bateria consiste em utilizar sensores piezoelétricos, que quando sofrem uma deformação geram uma tensão elétrica em seus terminais, para detectar as vibrações dos *pads* que serão atingidos pelas baquetas. Este sinal de tensão analógico é convertido em um sinal digital e, caso seja superior a um nível mínimo predeterminado de acordo com o código armazenado no microcontrolador, o sistema eletrônico reproduzirá a amostra de áudio correspondente.

A bateria possui três componentes principais sendo eles: os *pads* (sensores), a estrutura ou estante onde ficarão dispostos os *pads* para que o usuário possa atingi-los com as baquetas, e o microcontrolador ATmega2560 (que fará o papel do sintetizador tocando as amostras de áudio). Os *pads* são compostos por três elementos, a saber: uma placa de madeira, na qual estão afixados os sensores piezoelétricos, o próprio piezoelétrico e por fim uma camada de espuma EVA utilizada para amortecer a batida e proteger o piezoelétrico (elemento sensor).

A estante é uma estrutura feita a partir de canos PVC de meia polegada. O material foi escolhido visando à redução do custo final do instrumento musical e também por ser de fácil acesso e utilização, pois qualquer pessoa poderá estar apta a realizar a montagem da bateria.

A placa Arduino escolhida foi a baseada no microcontrolador ATmega2560 devido a sua maior capacidade de processamento e também a quantidade de portas analógicas disponíveis, possibilitando deste modo, utilizar um número maior de *pads* na confecção da bateria. O código (*software*) proposto também foi pensado e desenvolvido de modo a ser de fácil compreensão, utilização e atualização por aqueles que estão iniciando na programação. O programa desenvolvido baseia-se na biblioteca denominada "*tmrpcm*", cuja principal função é reproduzir (converter) sons no formato WAVE através das portas (saídas) "*PWM*" (*do inglês Pulse Width Modulation*) presentes na placa Arduino.

Este documento está estruturado da seguinte forma: o Capítulo 2 apresenta uma revisão bibliográfica com conceitos técnicos e históricos acerca do projeto. O Capítulo 3 apresenta a metodologia utilizada e outras metodologias alternativas que podem ser úteis para projetos musicais. Por fim, o Capítulo 4 apresenta os resultados e o Capítulo 5 apresenta a conclusão do trabalho.

1.1 MOTIVAÇÃO E OBJETIVOS

Nas cidades de Guaratinguetá e Aparecida existem diversos projetos sociais que utilizam a música como forma de trabalho, ensinando crianças carentes a tocar um instrumento, aprendendo não apenas música, mas também disciplina, companheirismo e o respeito. Porém, aprender a tocar um instrumento é uma atividade que demanda horas e horas diárias de estudo e prática, o que muitas vezes não é possível para esses alunos, uma vez que nem todos têm acesso aos instrumentos devido ao alto custo (uma bateria custa em torno de três mil reais), uma bateria eletrônica de baixo custo pode ser uma solução para parte destes alunos. Outro ponto é a existência de colégios técnicos na região que podem se beneficiar de um projeto de uma bateria eletrônica utilizando Arduino com o intuito de iniciar e motivar seus alunos na área da informática e da eletrônica. Por estes motivos, o trabalho conta com os seguintes objetivos:

1. Estudar e apresentar as possíveis formas de se trabalhar com áudio em programação;
2. Projetar sensores e estrutura de uma bateria eletrônica;
3. Desenvolver o código em Arduino para uma bateria eletrônica;
4. Expor as possibilidades de melhoria deste projeto;
5. Indicar outros modos de se desenvolver uma bateria eletrônica;
6. Tornar o projeto o mais simples e didático possível para alunos de música ou eletrônica;

Conforme foi dito, a grande motivação deste projeto é o uso para ensinar tanto alunos de música quanto de eletrônica, e por este motivo foi escolhido o Arduino por ter um custo mais baixo e fácil entendimento e aplicação, de modo que estudantes possam ter acesso a este projeto.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados e explicados os componentes necessários para a montagem deste modelo de bateria eletrônica. Também apresentaremos um pouco da história da bateria convencional e eletrônica.

2.1 TIPOS DE BATERIA

A música é composta por três elementos básicos (DESCOMPLICANDO A MÚSICA, 2017): Melodia (sucessão de sons combinados, ou seja, a parte cantável da música), harmonia (combinação de sons simultâneos em acordes servindo de base para a melodia) e ritmo (marcação do tempo de uma música muitas vezes chamado de “batida”) sendo este terceiro de maior relevância para este trabalho.

O ritmo da música pode ser marcado de várias formas como, por exemplo, batendo palmas como em uma cantiga de roda, batendo os pés como faz uma tropa marchando ou com o uso de tambores que deu origem a bateria como a conhecemos hoje.

2.1.1 Bateria Acústica

No século passado, até aproximadamente, a década de 10, as bandas contavam com dois ou mais percussionistas para tocar caixa, bumbo e demais instrumentos, até que o primeiro pedal foi desenvolvido o que, juntamente com a estante para caixa, possibilitou um único músico percutir mais de um instrumento simultaneamente formando assim o kit de bateria básico composto por bumbo, prato de condução, caixa e um tom, conforme mostrado na figura 1 ([wikipedia, 2017](#)).

Figura 1 – Kit de bateria básico.



Fonte.: (Bateras-se, 2017)

Em meados da década de 40 surgiu o *hi-hat* composto de dois pratos acionados com o pé compondo assim a bateria que conhecemos hoje, apresentada na figura 2, composta por:

1. **Prato de condução:** prato metálico com sonoridade aguda e semelhante a um sino próprio para conduzir o ritmo de uma música;
2. **Surdo:** Tambor com grande diâmetro e altura produzindo um som mais grave, duradouro e forte;
3. **Tom-tons:** Tambores com dimensões bem menores do que as do surdo produzindo sons de tonalidade média e geralmente afinados cada um em uma nota diferente;
4. **Bumbo:** Tambor maior do que surdo dando batidas mais graves e secas quando percutido pela maceta acoplada ao pedal;

5. **Caixa:** Também conhecido como tarol, consiste em um tambor com duas membranas sendo que em contato com a pele inferior há uma esteira composta de molas de arame que ressonam com a pele superior produzindo um som repicado;
6. **Chimbal:** Também conhecido como *hi-hat* ou contratempo (nome utilizado no *jazz*) consiste de dois pratos montados com suas cavidades voltadas um para o outro e atrelados a um pedal que quando acionado os une podendo ser percutido estando aberto, fechado ou produzindo com um acionamento mais vigoroso do pedal.

Figura 2 – Kit de bateria acústica



Fonte: (WIKIPEDIA,2017A)

2.1.2 Bateria Eletrônica

Na década de 50, começaram a surgir os sintetizadores, teclados capazes de produzir os mais diferentes timbres e tonalidades modulando sinais elétricos.

Na década de 70 o professor Brian Groves desenvolveu a primeira bateria eletrônica que utilizava como sensor um ímã que, quando tinha seu suporte tocado, se movia dentro de uma bobina gerando assim um sinal elétrico que seria sintetizado no circuito principal composto por centenas de transistores, o que tornava inviável a sua comercialização.

No fim da década de 70 surgiram então as primeiras baterias que utilizavam o sistema de amostragem no qual, através de um gerador de funções, produzia o som correspondente a peça que fora percutida (gatilho). Este sistema foi sendo aprimorado e é o mais comum hoje em dia, reproduzindo uma gravação ao invés de utilizar um gerador de funções. Pode-se ainda encontrar modelos que utilizam modelagem matemática para gerar o som (BADNESS,1991).

A Figura 3 apresenta uma bateria eletrônica portátil que é o modelo mais utilizado por sua praticidade, porém há um modelo chamado híbrido ou trigado que consiste em utilizar uma bateria comum com seus componentes abafados e sensores que captam a batida gerando um sinal elétrico a ser processado.

Figura 3 – Kit de bateria eletrônica



Fonte: (WIKIPEDIA, 2017B)

2.2 ÁUDIO E PROGRAMAÇÃO

Nesta seção são pontuados e explicados os formatos de áudio utilizados durante a pesquisa e execução do projeto.

2.2.1 Formato wave

De acordo com o Soundfile(2017), WAVE é um formato de áudio criado pela Microsoft e IBM sendo a abreviação de *Waveform Audio File Format* sendo também um subconjunto da especificação RIFF, utilizado para o armazenamento de arquivos multimídia. A informação que o WAVE armazena nada mais é do que os códigos correspondentes as amostras analógicas do áudio no formato de código de pulsos (PCM).

O WAVE é um formato de áudio que não possui compressão e que, portanto, tem fidelidade quanto ao som que o originou, porém, isto gera a necessidade mais espaço de armazenamento.

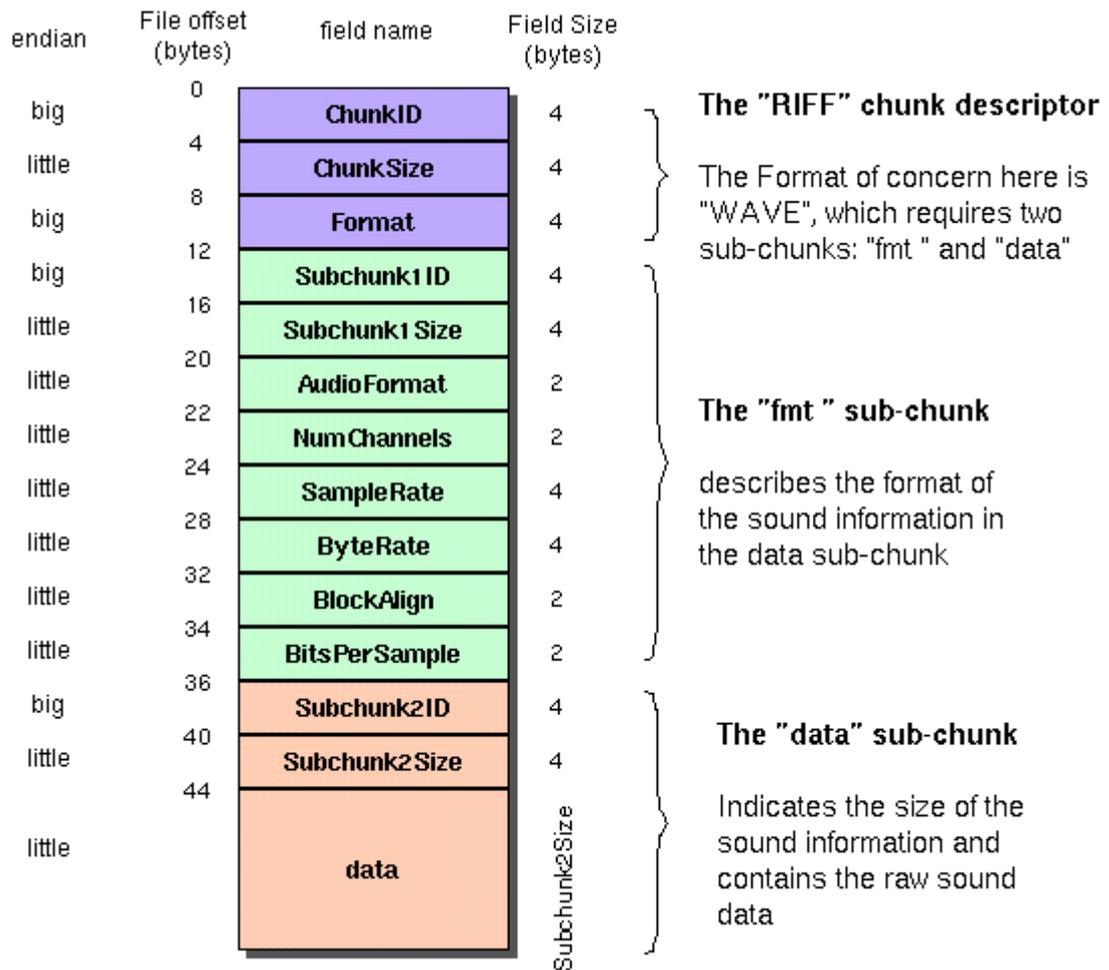
2.2.1.1 RIFF (*Resource Interchange File Format*)

A especificação RIFF consiste em um cabeçalho seguido de uma sequência de fragmentos de dados, também conhecidos como *chunk*. O WAVE geralmente é estruturado como um arquivo RIFF dividido em duas partes: A primeira é um “fmt” que especifica o formato de dados e, a segunda, contém os dados das amostras sonoras conforme apresentado na Figura 4, que apresenta a forma completa de um arquivo WAVE

A forma completa inicia-se com a descrição do RIFF, apresentado na cor roxa, que contém o seguinte:

- *ChunkID*: 4 bytes que contém as letras “RIFF” no formato ASCII, serve para identificar que o arquivo se baseia no método RIFF.
- *ChunkSize* 4 bytes que indicam o tamanho total do arquivo em bytes desconsiderando os 8 bytes referentes aos campos *ChunkID* e *ChunkSize*
- *Format*: 4 bytes que contém as letras “WAVE” no formato ASCII , indicando qual o formato do arquivo.

Figura 4 – Forma canônica do arquivo WAVE



Fonte: SOUNDFILE,2017

O sub-*chunk* "fmt", apresentado na cor verde, descreve as informações do formato de som e contém:

- *Subchunk1ID*: 4 bytes que contém as letras "fmt" no formato ASCII, indicando o início do primeiro sub-bloco;
- *Subchunk1Size*: 4 bytes que indica o tamanho do resto do sub-bloco sendo 16 para PCM.
- *Audio Format*: 2 bytes que indicam se há ou não compressão no áudio
- *NumChannels*: 2 bytes que indica quantos canais o áudio tem
- *SampleRate*: 4 bytes que indicam a taxa de amostragem do sinal de áudio

- *ByteRate*: 4 bytes que indicam o número de bytes necessários para se ter 1 segundo de áudio.
- *BlockAlign*: 2 bytes que indica o número de bytes necessários para guardar uma amostra do áudio, incluindo todos os canais
- *BitsPerSample*: 2 bytes que indica quantos bits foram utilizados para quantizar o áudio.

Por fim o *sub-chunk* “data”, apresentado em laranja, que contém as informações do áudio contendo:

- *Subchunk2ID*: 4 bytes que contém as letras “data” no formato ASCII indicando o início do sub-bloco data;
- *Subchunk2Size*: 4 bytes que indicam o número de bytes(amostras) do áudio, ou seja, o indica o número de amostras do áudio.
- *Data*: Contém os bytes das amostras do áudio, é o áudio propriamente dito.

2.2.1.2 PCM (*Pulse Code Modulation*)

De acordo com Moecke(2006) PCM é um método (código) utilizado para converter o sinal analógico para um sinal digital usando um conversor A/D através de amostragem e quantização, ou seja o sinal é amostrado em uma frequência constante e quantizado para o nível mais próximo de acordo com a resolução dada pela quantidade de bits utilizados no processo.

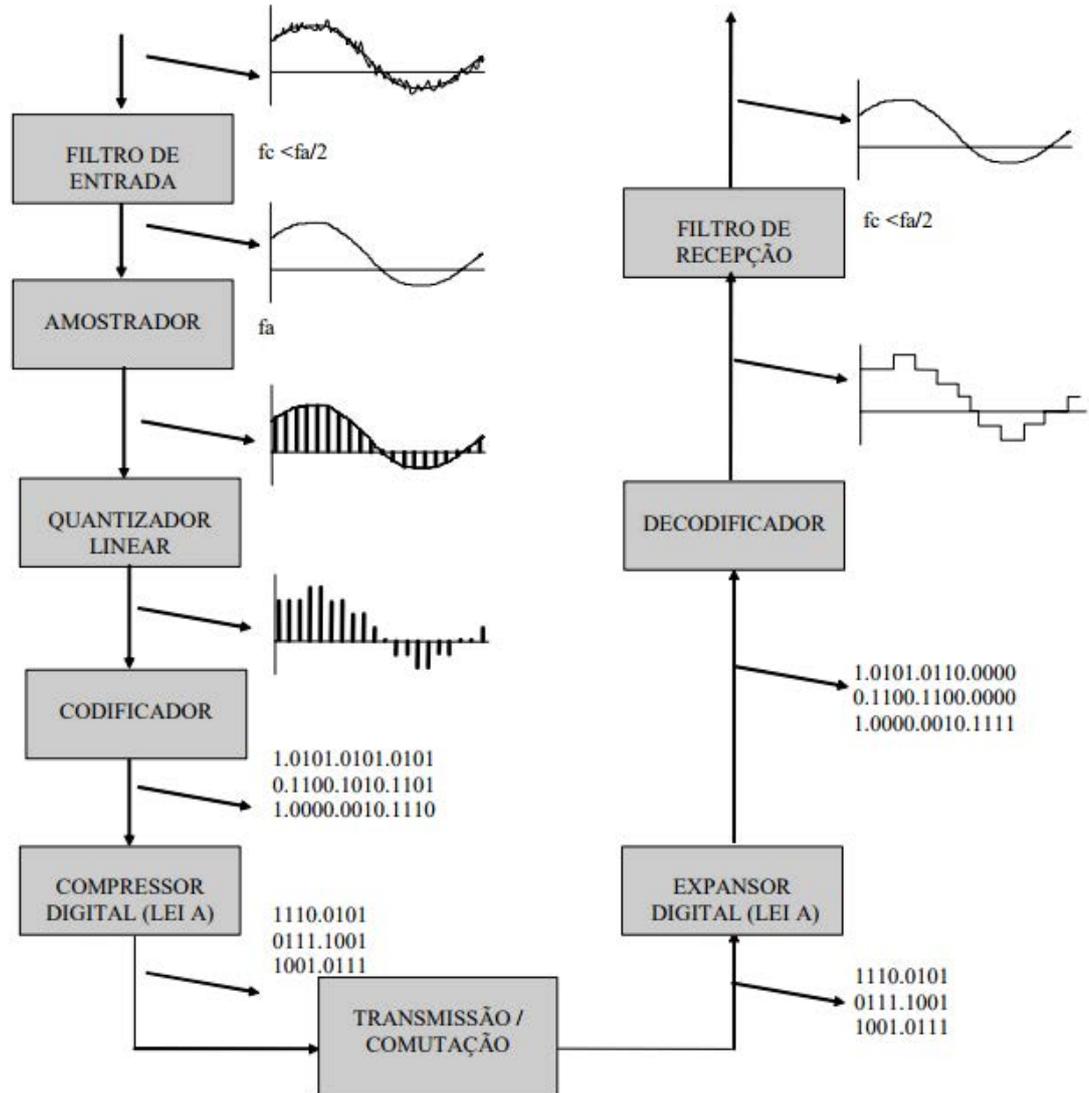
A amostragem consiste capturar o valor da onda em intervalos de tempo predeterminados, (taxa de amostragem). Este processo é realizado através do acionamento de uma chave em instantes predeterminados.

A quantização consiste em tomar o valor analógico amostrado e convertê-lo para um nível discreto baseado em valores predeterminados, espaçados de acordo com a quantidade de bits disponíveis para este processo.

A Figura 5 apresenta o processo completo de geração e recuperação de dados no formato PCM, onde, inicialmente, o sinal de entrada é filtrado, depois passa pelo amostrador que irá marcar/capturar os valores instantâneos, passa pelo quantizador e pelo codificador que

irá converter o sinal analógico em níveis digitais, é transmitido e por fim decodificado no receptor executando o processo inverso.

Figura 5 – Etapas da modulação PCM (Código de Pulso)



Fonte: Moecke, 2006

2.2.2 MIDI

De acordo com a associação dos fabricantes de MIDI (COMITTEE, 1995) o formato MIDI, sigla para *Musical Instrument Digital Interface*, surgiu como uma forma de sincronizar o desempenho de instrumentos padronizando a comunicação entre os sintetizadores. Em

outras palavras, é um protocolo de comunicação que possibilita a transmissão de dados entre instrumentos, computadores e sintetizadores.

O MIDI não é um sinal de áudio, mas tem uma função parecida com a de uma partitura transmitindo informações com o pressionamento de teclas, manipulação de *joysticks*, altura (tonalidade), intensidade (amplitude), duração e outras informações de um determinado som, de modo que uma sequência de dados MIDI representa uma melodia, porém não há uma especificação clara do timbre, sendo deixado para que os receptores o definam, podendo um mesmo arquivo MIDI gerar sons diferentes para sintetizadores diferentes, como por exemplo, dois cantores com mesma habilidade de entoar uma melodia, porém, com vozes diferentes, gerando sons diferentes para a mesma partitura.

A transferência de informação MIDI ocorre por uma interface *full duplex* através de uma UART (*Universal Assynchronous Receiver Transmitter*) serial e assíncrona com *baud rate* de 31250 baud (+/-1%), com um start bit seguidos de 8 bits de dados e por final um stop bit. No total são 10 bits em um período de 320 microssegundos. Pode se comunicar com até 16 canais independentes.

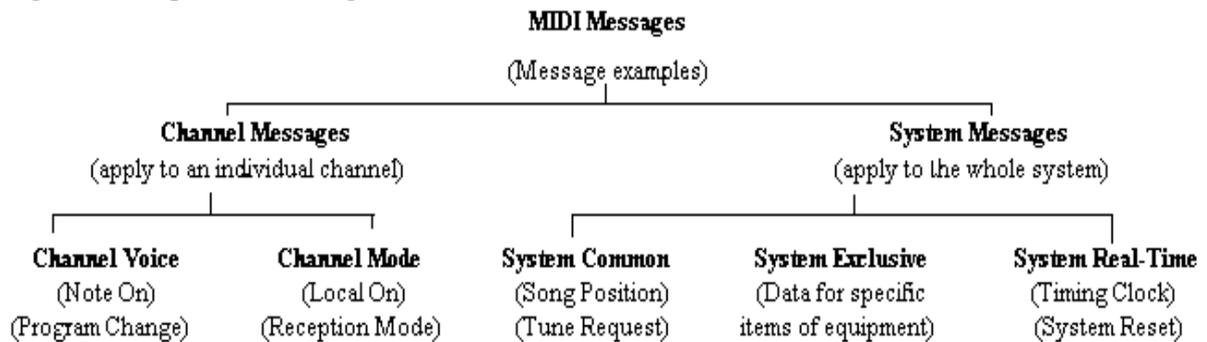
2.2.2.1 Transmissão dos dados

O *hardware* necessário para que o protocolo MIDI funcione são os conectores DIN de 5 pinos e uma unidade de processamento (*Digital Signal Processor*).

As portas MIDI (que utilizam conectores DIN de 5 pinos) possuem cada qual sua função específica sendo a *in* e a *out* responsáveis por receber e enviar dados entre os aparelhos enquanto a *thru* envia uma réplica do sinal recebido na porta *in* para a porta *out* funcionando como retransmissor.

A comunicação MIDI é feita através de mensagens que são compostas por sequências de bits que emitem um determinado tipo de *byte*, sendo que o intervalo 00h até 7Fh compreendem os *data byte*, responsáveis por carregar os parâmetros da informação enviada, enquanto de 80h até FFh são os *status byte*, que indicam qual o tipo de informação enviada. As mensagens podem ser classificadas conforme apresentado na Figura 6.

Figura 6 – Tipos de mensagens MIDI



Fonte: dilettantesdictionary.org¹.

2.2.3 Pure Data

De acordo com Drymonitis(2015) *Pure Data* é uma linguagem de programação visual e ambiente para áudio e imagens *open source*.

O *Pure Data* é baseado em uma interface gráfica composta, na maioria dos casos, por uma caixa que representa uma determinada função e são conectadas por linhas que representam os cabos de conexão em dispositivos de áudio analógico. Este tipo de programação também é conhecida como programação de fluxo de dados, pois o modo como as partes de um programa estão conectadas indicam como seus dados fluem de uma parte do programa para outro.

A programação visual é vantajosa por ser mais prática para a prototipagem, uma vez que não é necessário escrever uma grande quantidade de linhas de código para obter um resultado satisfatório, também é mais intuitiva. Porém existem algumas limitações técnicas, como o uso limitado para áreas que não sejam a música e também certa dificuldade em resolver problemas lógico-matemáticos devido ao modo diferenciado do fluxo de dados.

2.3 BIBLIOTECAS

A placa Arduino possui uma infinidade de bibliotecas e soluções já disponíveis *online* e que podem servir a aplicação que está sendo desenvolvida. Durante o projeto, algumas bibliotecas se mostraram como tendo grande potencial para o desenvolvimento do protótipo da bateria. Cada uma dessas bibliotecas será brevemente introduzida neste texto, porém,

¹ Disponível em: <http://www.dilettantesdictionary.org/index.php?search=1&searchtxt=ESS> acessado out 2017

nenhuma delas será detalhada uma vez que não há necessidade disto para compreender o código.

2.3.1 SPI (*Serial Peripheral Interface*)

Serial Peripheral Interface (SPI) é um protocolo de dados seriais síncronos usados por microcontroladores para se comunicarem rapidamente, a curtas distâncias, com um ou mais dispositivos periféricos, sendo que quase sempre o microcontrolador é o dispositivo mestre enquanto o periférico assume o papel do escravo.

Esta biblioteca permite que o microcontrolador ATmega2560 consiga comunicar-se com alguns componentes existentes na placa Arduino como, por exemplo, um cabo MIDI, um cartão SD ou uma porta USB através de bibliotecas e comandos específicos possibilitando a utilização e a comunicação do microcontrolador com outros periféricos ampliando, deste modo, a gama de possibilidades de utilização do microcontrolador (ARDUINO,2017).

2.3.2 SD

Esta biblioteca permite que o ATmega2560 leia e escreva arquivos dentro de um cartão SD conectado através de um *shield* (módulo eletrônico pré-desenvolvido para essa finalidade). O cartão SD ou SDHC precisa ter seus arquivos formatados como FAT16 ou FAT32 e utiliza arquivos com nomes curtos 8.3 (8 letras de nome e 3 de formato), sendo necessário adicionar o caminho do arquivo nas funções, já que o diretório de trabalho é sempre a raiz do cartão. Para que essa biblioteca funcione corretamente, faz-se necessário também o uso da biblioteca SPI (Arduino,2017).

2.3.3 TMRPCM

Esta biblioteca está disponível para *download* e instalação no compilador IDE da placa Arduino. Consiste em um player de WAV que utiliza uma saída PWM da placa para reproduzir o áudio sem necessidade de um circuito demodulador ou de *hardware* adicional. É necessário somente conectar essa saída a uma caixa de som ou um amplificador de guitarra (GITHUB,2017).

Recomenda-se que a utilização de arquivos WAV mono e configurados com 8 bits de resolução e amostragem de 32 kHz, pois com essas evita-se que o microcontrolador fique sobrecarregado e conseqüentemente lento ou superaqueça.

Os arquivos WAV podem ser facilmente encontrados na internet e, em alguns casos, a resolução ou taxa de amostragem será diferente da indicada anteriormente, mas isso é fácil de ser resolvido utilizando programas de edição de áudio como, por exemplo, com o *software* audacity. O arquivo devidamente formatado deve ser salvo no cartão SD.

Esta biblioteca poderia ser substituída por um sistema que utilizaria uma tabela com os valores de amplitude convertidos em um sinal PWM, que seria emitido utilizando amostragem e interrupção na frequência de amostragem demodulado e produziria o áudio, porém o uso da biblioteca torna o projeto mais fácil de ser reproduzido e modificado e também facilita a criação e alteração de sons.

2.3.4 MIDI e MIDIUSB

Existem basicamente duas bibliotecas que utilizam essa linguagem na plataforma Arduino, são elas a MIDI que possibilita enviar e receber informações através dos pinos da placa utilizando um cabo DIN, conforme foi dito anteriormente, ou a MIDIUSB que consiste no uso do próprio cabo USB da placa Arduino para enviar e receber mensagens no formato MIDI para o computador.

2.4 HARDWARE

Neste projeto existem basicamente dois componentes básicos de *hardware* necessários para a montagem da bateria, são eles: o sensor piezoelétrico e a placa Arduino.

2.4.1 Sensor Piezoelétrico

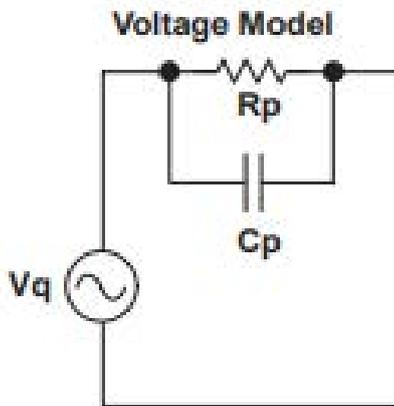
Conforme diz Karki(2000) a piezoeletricidade consiste na eletricidade oriunda da polarização de cristais dielétricos quando mecanicamente estressados ou do caminho inverso, gerando deformação mecânica ao aplicar eletricidade no cristal. Além do quartzo, do titanato de bário e do zirconato de chumbo, diversos materiais apresentam tais características.

A teoria piezolétrica se baseia no dipolo elétrico de um cristal iônico que quando em repouso tem seus dipolos positivos e negativos anulados mutuamente, devido à simetria do

material. Porém com o estresse mecânico a simetria é quebrada devido à deformação do material gerando, assim, um momento dipolo e conseqüentemente um campo elétrico através do cristal, originando uma carga elétrica proporcional a pressão aplicada. Os sensores piezoelétricos não são indicados para aplicações com tensão contínua ou aplicações estáticas, pois a carga decai com o tempo, porém, para aplicações dinâmicas ou com tensões alternadas são adequados.

O sensor piezoelétrico pode ser modelado como sendo uma fonte de tensão em série com um resistor e um capacitor paralelos entre si conforme mostrado na Figura 7.

Figura 7 – Modelo do sensor piezoelétrico



Fonte: Karki(2000).

A carga produzida depende da constante piezoelétrica do dispositivo enquanto a capacitância depende das dimensões e da constante dielétrica do material. Quando o material é deformado a capacitância se carrega com as cargas, porém quando o estresse mecânico é dissipado a carga do capacitor se descarrega através da resistência em paralelo com a carga e assim reinicia-se o processo.

2.4.2 Placa Arduino

A placa Arduino é o principal sistema de *hardware* e *software* de código aberto do mundo possibilitando que qualquer pessoa possa utilizá-lo fazendo uso de bibliotecas e documentações completamente grátis disponíveis online.

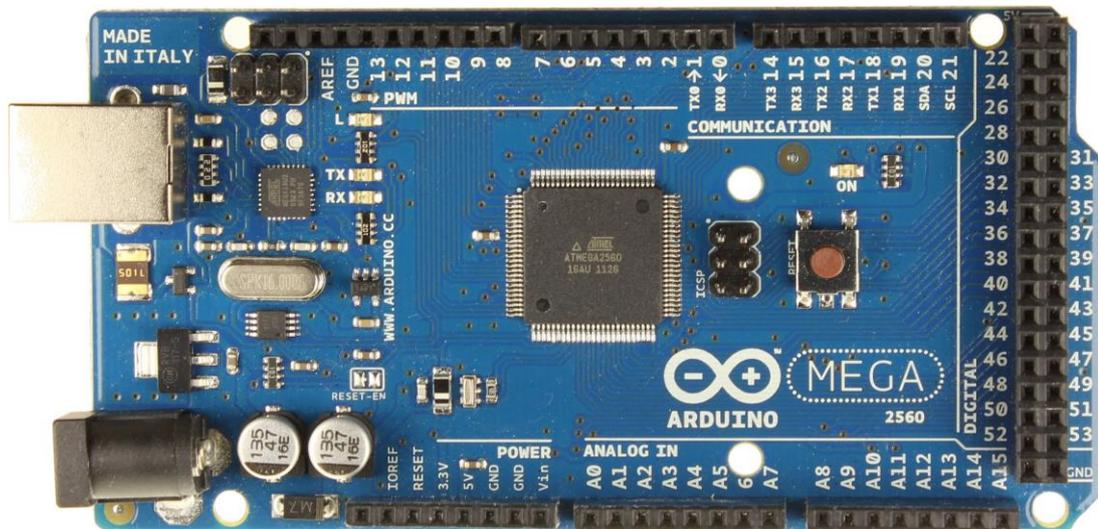
A placa Arduino MEGA 2560 é uma placa microcontrolada baseada no microcontrolador ATmega2560 tendo 54 entrada/saídas digitais sendo 14 com tecnologia

PWM, 16 entradas analógicas, 4UARTs, um cristal de 16MHz, conexão USB e outras características que podem ser usadas de acordo com a aplicação desejada.

As grandes vantagens da placa Arduino são seu o baixo custo, bibliotecas e documentação grátis disponível *online*, capacidade significativa de processamento e grande facilidade de aprendizagem o que viabiliza nosso objetivo de criar uma ferramenta que possa ser reproduzida e compreendida por alunos de cursos de eletrônica, informática ou música.

Existem ainda outras soluções com características parecidas como, por exemplo, o *Raspberry Pi*, porem este utiliza linguagem mais complexa e possui um custo maior apesar de ter maior capacidade em termos de processamento. A figura 8 mostra a imagem da placa Arduino MEGA2560.

Figura 8 – Placa ARDUINO MEGA



Fonte: Adaptado de Arduino.cc

3 DESENVOLVIMENTO

O desenvolvimento é dividido em duas partes, a primeira será dedicada a apresentar modos possíveis de se fazer uma bateria eletrônica bem como suas vantagens e desvantagens e a segunda parte o método utilizado para a implementação e os resultados obtidos.

3.1 POSSÍVEIS MÉTODOS PARA A CONSTRUÇÃO DE UMA BATERIA ELETRÔNICA

Neste trecho do texto serão relatados os métodos encontrados durante a pesquisa bem como suas vantagens e desvantagens.

3.1.1 Utilizando *Pure Data*

Durante a pesquisa, o primeiro método encontrado foi encontrado em Drymonitis(2015) que consistia em utilizar a placa Arduino como uma interface entre os sensores piezoelétricos e o computador.

Inicialmente os sensores piezoelétricos são conectados em paralelo com um resistor de modo a dissipar a carga armazenada no piezoelétrico mais rapidamente após a batida. A placa Arduino é utilizada como uma interface entre os sensores e o computador, e o código utilizado basicamente detecta qual dos sensores foi ativado com um valor acima de um limiar mínimo. Os dados de qual sensor foi tocado e qual o valor máximo atingido é enviado ao computador, onde um programa em *pure data* irá tocar o áudio correspondente ao sensor com volume proporcional a intensidade da batida. Alguns botões foram acrescentados ao programa com o objetivo de alterar os sons da bateria bem como inserir alguns efeitos no áudio, como uma espécie de distorção.

Este projeto tem algumas vantagens claras, como a facilidade em se alterar o código *pure data* no computador e também a grande capacidade de processamento de um computador convencional, quando comparado a um microcontrolador ATmega, possibilitando uma maior qualidade de áudio Outro ponto que merece atenção é o controle da intensidade da batida o que cria uma experiência melhor para o usuário por se aproximar do dinamismo de um kit real.

As desvantagens desta técnica são a necessidade de se aprender a utilizar o *pure data* para poder compreender e alterar o protótipo, ainda que seja um sistema de fácil

aprendizagem, há métodos que dispensam esse novo aprendizado. Outro ponto é que, apesar de trazer maior qualidade de áudio e capacidade de processamento, a necessidade de se manter o protótipo conectado a um computador dificulta o transporte e aumenta o custo para tornar o módulo completamente autônomo e acaba se afastando do objetivo de criar um protótipo simples e eficiente, de modo que estudantes iniciantes de programação e/ou música possam reproduzir e compreender.

3.1.2 Utilizando MIDI

Considerando a utilização da linguagem MIDI, há mais de um modo de se desenvolver uma bateria eletrônica, incluindo alguns que, apesar de não serem viáveis por serem desvantajosas em relação a soluções mais baratas, são interessantes por abrir possibilidades para outros projetos na área da música.

A primeira e mais simples possibilidade é utilizar a placa Arduino para captar os sinais dos sensores e convertê-los e transmiti-los em formato MIDI (apresentando os dados de qual sensor foi pressionado e com qual intensidade), seja através de um cabo DIN para um computador ou um sintetizador ou através do próprio USB para o computador onde um *software* musical interpretaria os dados MIDI, produzidos pelo Arduino, e emitiria os sons correspondentes.

A grande vantagem deste método é a grande facilidade de se encontrar softwares, alguns até livres, que utilizem o formato MIDI como entrada para simular uma bateria, bem como utilizar softwares de gravação para criar uma trilha MIDI.

A desvantagem, novamente, consiste na necessidade de se ter um computador sempre acoplado a placa Arduino ou até mesmo um sintetizador que converteria o MIDI em áudio.

Além de utilizar a placa Arduino para gerar e transmitir a mensagem MIDI, este também poderia ser utilizado para interpretá-la e gerar o áudio como um sintetizador, isto seria inviável para este projeto por necessitar de um segunda placa Arduino, porém, é uma característica interessante para outros projetos na área da música.

3.1.3 Placa Arduino reproduzindo áudio

A placa Arduino possui entradas analógicas, porém, não possui saídas com a mesma característica, o que gera a necessidade de se criar outras formas de reproduzir áudio diretamente a partir desse kit sem a necessidade de utilizar um computador. Inicialmente é

explicado como a placa Arduino geraria a onda relativa ao áudio e depois como essas são emitidas considerando a ausência de saídas analógicas.

A primeira forma seria equacionar a faixa de áudio de modo que, de acordo com a passagem do tempo, o resultado da equação daria o valor instantâneo a ser reproduzido no alto falante fazendo com que a placa Arduino trabalhe de forma análoga a um gerador de funções e apesar desta solução utilizar menor quantidade de memória, seria necessário um grande poder de processamento para que pudesse ser utilizado.

Outro modo de reproduzir áudio através da placa Arduino seria o uso de uma tabela para armazenar os valores amostrados, a uma taxa de amostragem conhecida a exemplo do formato WAV que é basicamente uma tabela contendo os valores amostrados conforme foi explicado anteriormente. Utilizando uma interrupção, seria possível ler e reproduzir os dados armazenados nessa tabela em uma frequência de leitura adequada. O formato WAV poderia ter tratamento semelhante, porém, devido à baixa capacidade de memória da placa Arduino poderia ser utilizado um cartão de memória para armazenarem os arquivos.

Tendo em vista esses modos de se gerar uma onda, faz-se necessário um modo de colocá-la para fora do microcontrolador, considerando que não existe uma saída analógica, e foram pensadas 3 maneiras. A primeira consiste em utilizar um conjunto de saídas digitais e enviar o valor lido ou gerado matematicamente para este conjunto de saídas e depois conectar estas saídas a um conversor D/A, como era feito nos microprocessadores PIC que utilizavam este método para gerar áudio e outros tipos de onda. O segundo método consiste em utilizar a saída PWM da placa Arduino. Basicamente, modulam-se os valores correspondentes ao áudio em uma onda PWM que será depois demodulada através de um circuito eletrônico que possibilitará a obtenção da onda sonora. Por último, o método que foi escolhido para esse projeto foi utilizar a biblioteca TMRPCM que, basicamente, faz a leitura do arquivo de áudio e emite um sinal através da saída PWM, que pode ser conectada diretamente a um alto falante emitindo o áudio.

3.2 MÉTODO UTILIZADO.

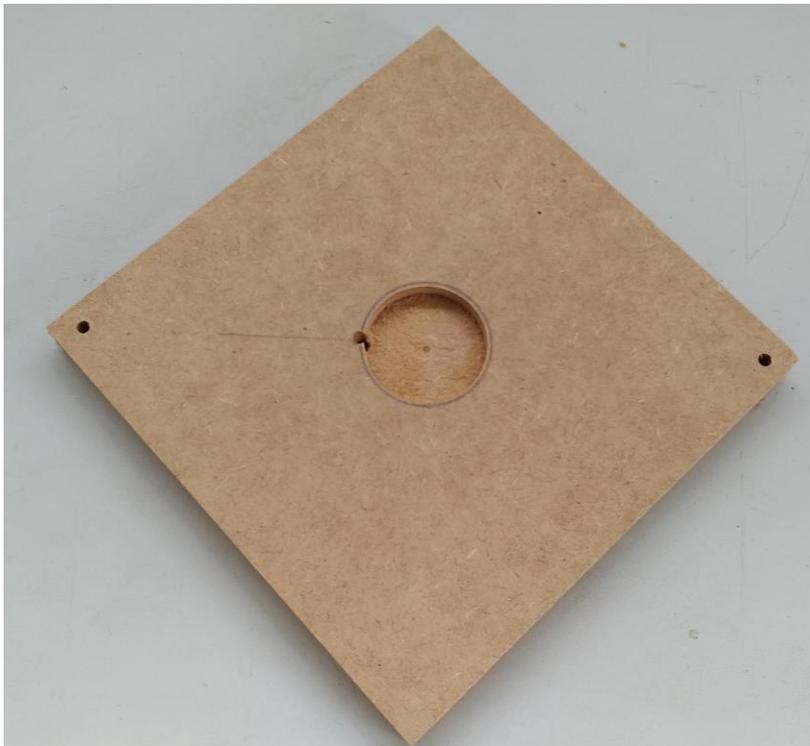
Neste trecho do texto está explicado como foi efetivamente projetada e montada esta bateria.

Como foi dito anteriormente, o objetivo deste projeto é construir uma bateria eletrônica do modo mais simples e barato possível, para que possibilite a reprodução por alunos de música ou de eletrônica.

3.2.1 Captadores

Os captadores da bateria consistem em placas de madeira de 15x15cm com um furo de 3,5cm de diâmetro no centro onde um sensor piezo elétrico será fixado. Existem, ainda, dois furos distantes 1 cm do vértice dos quadrados para facilitar a fixação dos captadores no hack e um furo tangente ao encaixe central por onde passarão os fios do piezoelétrico, conforme pode ser visto na Figura 9. Uma camada de EVA com 5mm de espessura é colocada sobre a madeira prendendo e protegendo o sensor piezoelétrico.

Figura 9 – Visão geral do captador



Fonte: Criado pelo autor

Na Figura 10 estão destacados o furo central com 3,5cm de diâmetro e o furo tangente a este por onde passam os fios soldados ao sensor piezoelétrico já tudo fixado em seu devido lugar perfeitamente acomodados no interior da madeira.

Figura 10 – Detalhe do furo central



Fonte: Criado pelo autor

3.2.2 Estrutura

A estrutura da bateria é composta pelos componentes descritos na tabela 1

Tabela 1 – Material para a estrutura

Descrição do item	Quantidade
1 metro de cano de pvc de ½”	2
90cm de cano de pvc de ½”	7
60 cm de cano de pvc de ½”	1
50 cm de cano de pvc de ½”	8
30 cm de cano de pvc de ½”	4
Conector T para cano de pvc de ½”	4
Conector 90° para cano de pvc de ½”	4
Conector 45° para cano de pvc de ½”	6
Abraçadeiras plásticas	50
Ventosas de borracha	5

Fonte: Criado pelo autor

O cano de PVC foi escolhido devido ao seu baixo custo e a possibilidade de utilizá-lo tanto como estrutura mecânica como eletroduto para a fiação dos captadores que deverá ser

passada e numerada durante a montagem da estrutura. Recomenda-se não utilizar cola para prender as peças da bateria de modo a facilitar no momento de passar os fios.

Inicialmente monta-se a base e os dois suportes onde ficarão fixados os pads com 1 peça de 90cm, 2 de 50cm e 2 conectores 45° 3 arcos como está apresentado na Figura 11.

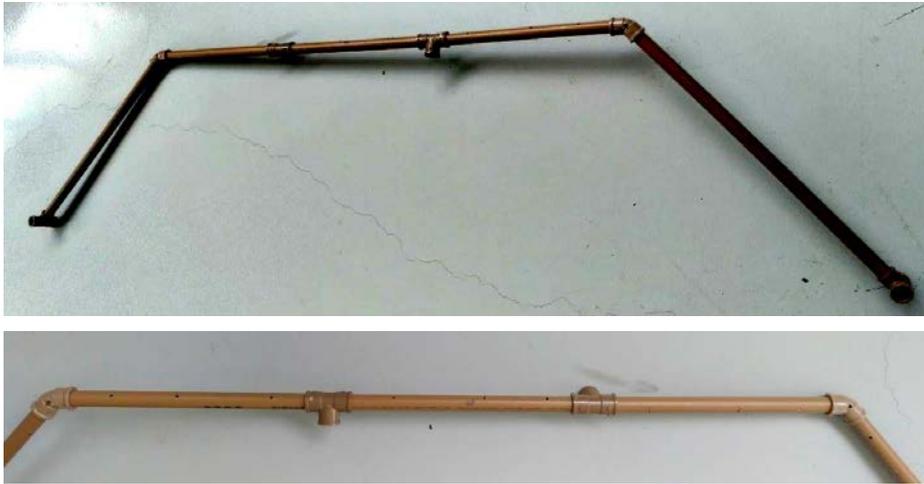
Figura 11 – 3 arcos sendo a base, a estante central e o topo da bateria



Fonte: Criado pelo autor

Dois desses arcos são a base e o topo e, nas pontas destes, estão colocados os 4 conectores de 90°. O terceiro será o nível central e, neste, são utilizados os 4 conectores em T, sendo dois nas pontas plugados pela base do T e os outros dois são conectados no cano de 90cm. Para isto ele foi cortado em 3 partes de 30cm e unidos novamente pelos conectores em T na horizontal pelo topo do T sendo, da esquerda pra direita, o primeiro com a base voltado para a parte convexa do arco (por onde sairão os fios) e o segundo voltado para baixo (onde será acoplado o cano de 60cm que irá segurar o bumbo). Este nível central da estante pode ser visto com detalhes na Figura 12

Figura 12 – Arco central e detalhe dos conectores em T no cano de 90cm



Fonte: Criado pelo autor

Faz-se furos em dois dos arcos onde ficam presos os captadores (sendo 2 em cada cano de 50cm e 3 em cada cano de 90cm) como mostra a Figura 13, bastando utilizar os captadores como medida, faz-se, ainda, 1 furo em cada conector de 45°, de modo a possibilitar a passagem dos fios sendo que no arco que vai ficar no topo da bateria. O furo deve apontar para baixo enquanto que no arco central deve apontar para cima.

Figura 13 – Detalhe de como é a disposição dos furos no suporte

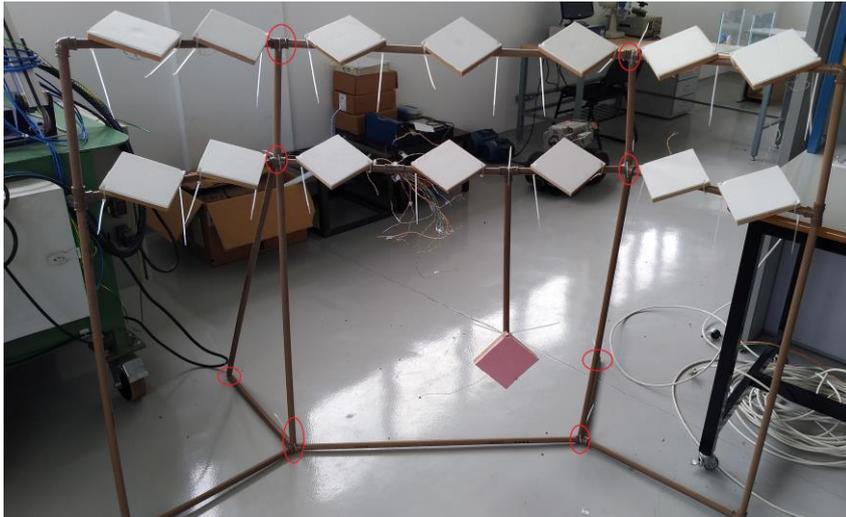


Fonte: Criado pelo autor

Posteriormente, deve-se montar a estrutura, conforme esta apresentado na Figura 14, utilizando os canos de 90cm para unir a base ao patamar central e, depois, os de 30cm para unir o patamar central ao topo e utilizar os canos de 50cm e de 1m para, juntamente com os conectores de 45° da base e do patamar central, formar 2 triângulos com base de 50cm na parte traseira da bateria de modo a dar sustentação a estrutura. A fiação deverá ser passada conforme a Figura 15. Depois, deve-se, por fim, colocar 5 ventosas, sendo duas nas pontas

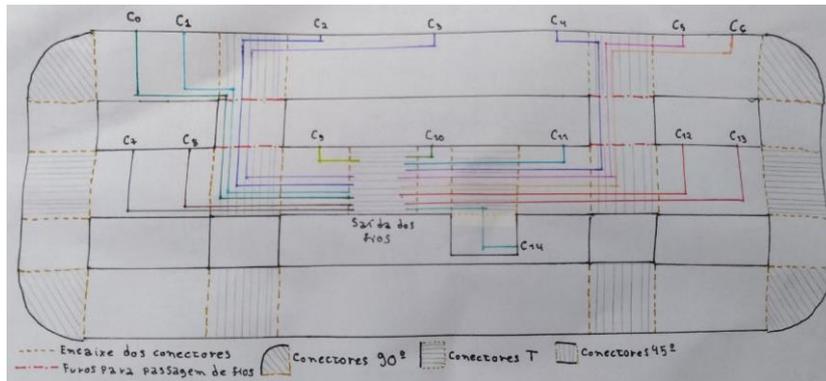
dos triângulos, 2 nos conectores de 90° da base e, o ultimo, no centro do cano de 90cm da base, para fixar a estrutura no solo. Na Figura 14 os círculos em vermelho marcam os locais onde, ao invés de conectores de cano, foram utilizadas abraçadeiras plásticas para unir os canos que se encontram nestes pontos.

Figura 14 – Montagem final da bateria



Fonte: Criado pelo autor

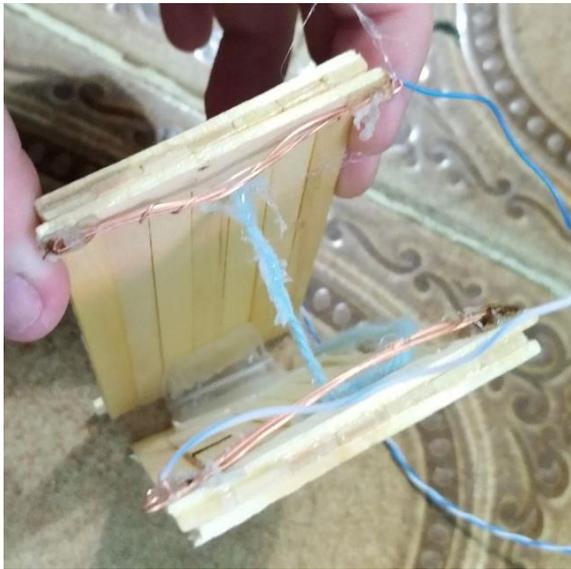
Figura 15 – Esquema da fiação passada no interior da armação



Fonte: Criado pelo autor

O pedal, para definir se o chimal está aberto ou fechado, foi feito utilizando palitos de sorvete quadrados. 3 palitos foram partidos ao meio de modo a ter 6 peças pequenas, depois foram montadas 2 placas nas quais foram colados na ponta uns enrolados de fios que servirão para fechar o circuito quando o pedal for pressionado. Para manter o movimento de abrir e fechar o pedal foi recortado um pedaço de plástico da capa de um DVD e dobrado de modo que ficasse aberto quando solto e as duas placas, já com o metal na ponta, foram colados a esse plástico. Os detalhes estão na Figura 16.

Figura 16 – Pedal do chimbal



Fonte: Criado pelo autor

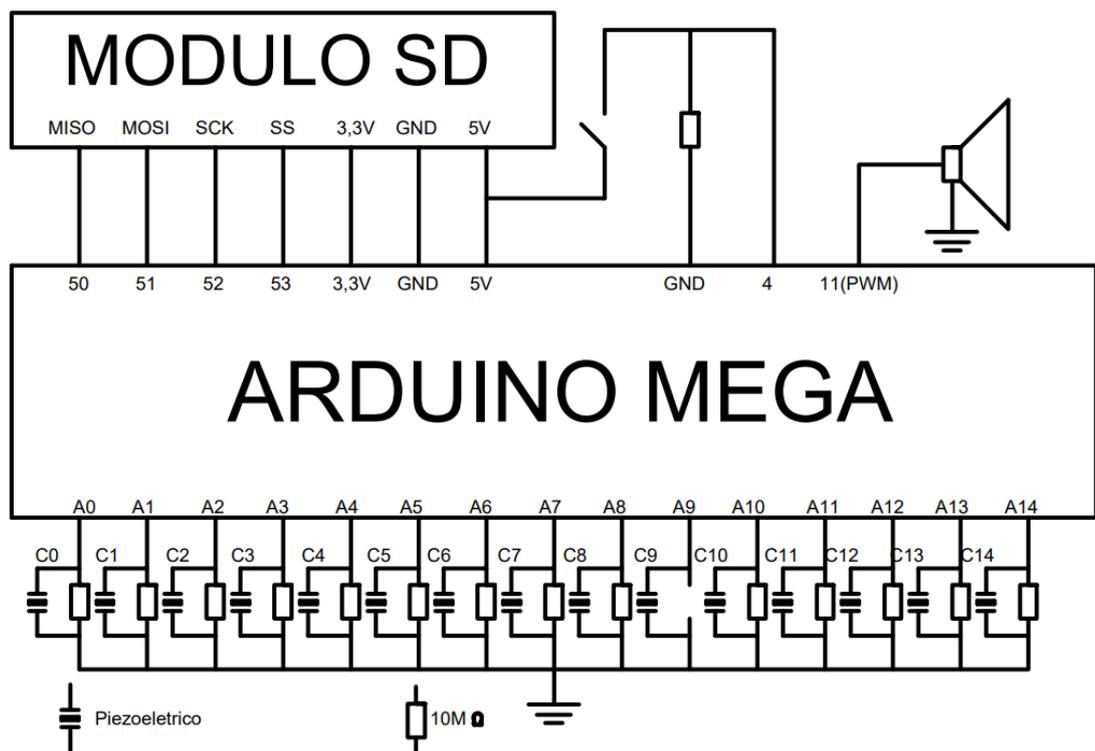
3.2.3 Montagem na protoboard e na placa Arduino

No protoboard são montados os 15 resistores de $10\text{M}\Omega$ conectados paralelamente aos piezoelétricos e o *pushbotton* com circuito *pullup* utilizado para marcar as posições aberto e fechado do chimbal

Na placa Arduino são conectados os piezoelétricos em paralelo com os resistor de $10\text{M}\Omega$ nas entradas analógicas, o leitor de cartão SD aos pinos correspondentes, o alto-falante à saída PWM, que foi definida no código, e a chave com o resistor de pull up a entrada digital número 4.

Na Figura 17 está desenhado o circuito como deve ser montado.

Figura 17 – Circuito eletrônico da bateria



Fonte: Criado pelo autor

3.2.4 Código

O código é bem simples e inicia definindo as bibliotecas a serem utilizadas (SD, SPI e TMRpcm), os pinos utilizados para a comunicação do cartão SD com a placa Arduino, o pino digital de entrada para o *pushbutton* que simulara o pedal do chimbau, a quantidade de sensores, o limiar mínimo da batida (valor mínimo que o sensor deve captar para evitar que os sensores sejam ativos acidentalmente), o delay entre a emissão de sons (para que haja tempo suficiente para que o sensor volta para o estado estacionário antes de nova emissão de áudio evitando que uma única batida ative o som duas vezes), o volume do áudio (qual o volume do áudio emitido pelo arduino), o pino de saída PWM por onde o áudio será emitido e verifica a conexão com o cartão SD.

Após a inicialização descrita, o código entra em um loop que, ciclicamente, atualiza os valores de cada entrada analógica de acordo com o sinal do piezo e, se algum deles for superior ao limiar mínimo, uma instrução reproduzirá o áudio correspondente ao captador acionado.

Há ainda um vetor denominado trava, que conta quantos ciclos o programa rodou antes de habilitar novamente o mesmo áudio para que uma única batida não ative mais de uma vez o mesmo som.

O código encontra-se reproduzido na íntegra e devidamente comentado nos apêndice A.

4 RESULTADOS

Os resultados serão apresentados considerando dois componentes principais da bateria sendo eles o hardware e o software e por fim um comentário geral do funcionamento.

4.1 HARDWARE

A estrutura suporta o peso dos sensores de forma bem estável e sem haver deformações nos canos posicionados horizontalmente, principalmente quando as ventosas estão devidamente fixadas ao solo dando maior firmeza e estabilidade a estrutura.

As dimensões da estrutura ficaram adequadas para o uso de alguém com aproximadamente 1,70m de altura, porém pode facilmente ser diminuída trocando os 4 canos que conectam a base ao primeiro patamar tornando assim possível o uso por alguém mais baixo.

Durante o uso da bateria, a estrutura suporta de forma eficiente as batidas mantendo-se estável ao ser tocada.

Os sensores mostraram-se eficientes em detectar as batidas, havendo apenas alguns pequenos problemas com um ou outro pad cujo piezo não estava bem encaixado, o que pode ser facilmente resolvido com a aplicação de cola quente entre o piezoelétrico e a madeira.

O pedal do chibal cumpriu perfeitamente sua função fechando o circuito quando necessário alterando, assim, o som do pad correspondente.

4.2 SOFTWARE

O código mostrou-se eficaz para solucionar o problema proposto, considerando todas as limitações do processador e também das bibliotecas.

O som produzido pela saída PWM do Arduino sofre certa deformação quando utilizado volume excessivo pela biblioteca TMRPCM, mas isso é facilmente resolvido, fixando o volume em um nível médio, e não prejudica o bom funcionamento da bateria.

O método utilizado para evitar que a bateria soe mais de uma vez com um único golpe funciona de forma adequada ao objetivo de iniciar estudantes de eletrônica e de música respeitando um intervalo de tempo entre uma batida e outra suficiente para até 4 toques por segundo dependendo do áudio tocado.

4.3 SISTEMA GERAL

Como um todo a bateria funcionou como esperado tendo como única limitação clara a impossibilidade de reproduzir mais de um arquivo WAVE por vez.

Outro problema detectado foi que algumas vezes ao se atingir um pad o som tocado era de outro, isto foi resolvido alterando o valor do limiar mínimo para disparar o áudio. Isto ocorre principalmente devido as características construtivas dos pads, haja visto que todos foram feitos a mão e não há uma forma de calibrá-los para que todos respondam da mesma forma ao toque das baquetas.

4.4 CUSTOS

Os valores gastos para a confecção da bateria estão descritos na tabela 2.

Tabela 2 – Custos

Descrição do item	Quantidade	Preço
Cano de pvc de ½”	15 metros(2 barras e meia)	R\$ 36,00
Conector T para cano de pvc de ½”	2	R\$ 2,00
Conector 90° para cano de pvc de ½”	4	R\$ 4,00
Conector 45° para cano de pvc de ½”	8	R\$ 8,00
Abraçadeiras plásticas	50	R\$ 7,00
Ventosas de borracha	5	R\$ 8,00
Piezoelétrico	15	R\$ 22,50
EVA	2 placas	R\$ 10,00
Fiação(cabo de rede)	15 metros	R\$ 22,50
Arduino MEGA	1	R\$ 80,00
Conector SD Arduino	1	R\$ 10,00
Total	-	R\$ 210,00

Fonte: Criado pelo autor

5 CONCLUSÃO

Os objetivos primários de criar uma bateria eletrônica em Arduino e de baixo custo foram atingidos com sucesso, visto que o projeto funciona e que seu custo é inferior a 250 reais, que corresponde a aproximadamente 5% do valor real de uma bateria eletrônica, sendo possível iniciar um aluno na música ou na programação com este projeto.

Durante o projeto pode-se notar que a biblioteca TMRpcm possui uma certa limitação quanto ao volume do áudio emitido uma vez que havendo 7 níveis de áudio os de número 1, 2 e 3 são muito baixos e os de número 5, 6 e 7 causam distorção no áudio e ruído sendo necessário fixar o volume no nível 4. Outra limitação é que a biblioteca é capaz de tocar um único áudio por vez, devido a limitações técnicas da capacidade de processamento da placa Arduino e, portanto, pode-se inferir que, apesar desta biblioteca atender as necessidades básicas do projeto, seria interessante a busca de uma solução melhor de modo que fosse possível alterar o volume do áudio emitido de acordo com a intensidade da batida dando assim maior dinamismo a bateria e utilizar outro método para a reprodução do áudio como, por exemplo, utilizar o sistema de interrupções para reproduzir mais de um arquivo simultaneamente.

Na pesquisa, uma das possibilidades descobertas mais interessantes consiste na capacidade da placa Arduino de enviar e receber sinais MIDI, que deve ser implementado em versões futuras do projeto, possibilitando, assim, que esta bateria seja utilizada para gravação de trilhas de áudio junto de um computador, com software adequado para tal tarefa, bem como possibilitar o uso do computador como sintetizador, dando, assim, uma nova opção de uso da bateria e eliminando a limitação de tocar um único áudio por vez.

Outra grande possibilidade seria adicionar um módulo *bluetooth*, através do qual poderia se selecionar em um smartphone, a partir de um banco de dados salvo no cartão SD, qual o áudio a ser reproduzido, de acordo com o captador ativado, dando assim maior flexibilidade em termos de quantidade de sons possíveis para o usuário.

Poderia-se também utilizar placas de maior capacidade de processamento como o *raspberry*, de modo a possibilitar melhor qualidade de áudio, reprodução de vários sons simultaneamente e ainda a gravação e transmissão do áudio da bateria.

Por fim, foi notado que, além do uso como bateria eletrônica, poderia ser utilizado para reabilitação e terapia de pessoas com alguma lesão ou limitação de movimentos, devido a

ampla gama de movimentos necessários para atacar os sensores, e, ainda, a possibilidade de se definir uma sequência para ser reproduzida, exercitando além do corpo a mente do usuário.

REFERÊNCIAS

- ARDUINO. **Arduino Mega**. Disponível em: < <https://www.Arduino.cc/en/Main/ArduinoBoardMega>>. Acesso: 20 out. 2017
- ARDUINO. **SD library**. Disponível em : < <https://www.Arduino.cc/en/Reference/SD>>. Acesso: 20 out. 2017
- ARDUINO. **SPI library**. Disponível em : < <https://www.Arduino.cc/en/Reference/SPI>>. Acesso: 20 out. 2017
- BADNESS, Ray F. **Drum Programming**: a complete guide to program and think like a drummer. Anaheim Hills, CA: Centerstream, 1991. 64p.
- BATERAS-SE. **Serginho Herval. Roupas 30 anos** . Disponível em: < <http://bateras-se.blogspot.com.br/2012/05/serginho-herval-roupa-nova-30-anos.html> >. Acesso: 20 out. 2017
- COMITTEE, J. M. S. **MIDI 1.0 Detailed Specification**. 42. ed. Los Angeles, CA, 1995. Disponível em: < <http://oktopus.hu/uploaded/Tudastar/MIDI%201.0%20Detailed%20Specification.pdf>> Acesso: 20 out. 2017
- DESCOMPLICANDO A MUSICA. **O que é música**. Disponível em : < <http://www.descomplicandoamusica.com/o-que-e-musica/>>. Acesso: 20 out. 2017
- DILLETANTE`S DICTIONARY., 2017. Disponível em : < <http://www.dilettantesdictionary.org/index.php?search=1&searchtxt=ESS>>. Acesso: 20 out. 2017.
- DRYMONITIS, Alexandros. **Digital electronics for musicians**. [S.l.]: Apress, 2015. 490 p.
- GITHUB. **TMRPCM**. Disponível em : < <https://github.com/TMRh20/TMRpcm>>. Acesso em outubro de 2017
- KARKI, J. **Signal Conditioning Piezoelectric Sensors**. [S.l.], 2000. Disponível em : < <http://www.ti.com/lit/an/sloa033a/sloa033a.pdf>>. Acesso: 20 out. 2017.
- MOECKE, M. **PCM: modulação por código de pulso**. São José/SC, 2006. Disponível em : http://www.sj.ifsc.edu.br/~fabiosouza/Tecnologo/Telefonia%202/Modulacao_PCM_v2006.pdf f>. Acesso: 20 out. 2017.
- SOUNDFILE. **WAVE PCM soundfile format**. Disponível em : < <http://soundfile.sapp.org/doc/WaveFormat/>>. Acesso: 20 out. 2017.
- WIKIPEDIA. **Bateria (instrumento musical)**. Disponível em : < https://pt.wikipedia.org/wiki/Bateria_eletr%C3%B4nica>. Acesso: 20 out. 2017.
- WIKIPEDIA. **Bateria eletrônica**. Disponível em : < [https://pt.wikipedia.org/wiki/Bateria_\(instrumento_musical\)](https://pt.wikipedia.org/wiki/Bateria_(instrumento_musical))>. Acesso: 20 out. 2017.

BIBLIOGRAFIA CONSULTADA

ABEL, A. M. S.; LUIZ, S. G.; SHIGUE, C. **Sensores e atuadores piezoelétricos**. Escola de Engenharia de Lorena, São Paulo, 2010.

KERR, D. **MIDI: the musical instrument digital interface**. Cleveland, OH, 2009. Disponível em: <<http://dougkerr.net/Pumpkin/articles/MIDI.pdf>>. Acesso: 20 out. 2017.

APÊNDICE A – Código comentado

```

#include <SD.h> //inclui a biblioteca SD
#include <SPI.h> //inclui a biblioteca SPI
#include <TMRpcm.h> //inclui a biblioteca TMRpcm

#define pinSD 53 //define o pino de comunicação do SD sendo 10 para o uno e 53 para o
mega
TMRpcm tmrpcm;

const int hiHat = 4; //hihat aberto ou fechado

const int numeroDeSensores = 15; //Seta o numero de sensores utilizados
int lims[] = {300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300};
int trava[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int atraso = 50; //tempo de delay entre sons
int vol = 4; //volume da bateria
int pad[numeroDeSensores];

void setup() {
  pinMode(hiHat, INPUT);
  tmrpcm.speakerPin = 11; //saida de audio 5,6,11,46
  tmrpcm.setVolume(vol); //seta o volume
  Serial.begin(9600);

  if (!SD.begin(pinSD)) { //verifica a conexão do cartão SD
    Serial.println("Falha na conexão do cartão SD"); //Avisa que o cartão não esta
    corretamente conectado
    return;
  }
  else {
    Serial.println("Cartao SD conectado com sucesso");
  }
}

```

```
void loop() {
  pad[0] = analogRead(0);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[1] = analogRead(1);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[2] = analogRead(2);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[3] = analogRead(3);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[4] = analogRead(4);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[5] = analogRead(5);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[6] = analogRead(6);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[7] = analogRead(7);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[8] = analogRead(8);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[9] = analogRead(9);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[10] = analogRead(10);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[11] = analogRead(11);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[12] = analogRead(12);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[13] = analogRead(13);///salva o valor maximo obtido pelo piezo no vetor pad
  pad[14] = analogRead(14);///salva o valor maximo obtido pelo piezo no vetor pad

  //Toca o som0 referente ao pad numero 0
  if ((pad[0] > lims[0])&&(trava[0] > atraso)) {
    tmrpcm.play("som0.wav");
    //delay(tempo);
    Serial.print("pad 0");
    Serial.println(pad[0]);
    trava[0] = 0;
  }

  //Toca o som0 referente ao pad numero 1
  if ((pad[1] > lims[1])&&(trava[1] > atraso)) {
    tmrpcm.play("som1.wav");
    //delay(tempo);
    Serial.print("pad 1");
    Serial.println(pad[1]);
  }
}
```

```
trava[1] = 0;
}

//Toca o som0 referente ao pad numero 2
if ((pad[2] > lims[2])&&(trava[2] > atraso)){
  tmrpcm.play("som2.wav");
  //delay(tempo);
  Serial.print("pad 2");
  Serial.println(pad[2]);
  trava[2] = 0;
}

//Toca o som0 referente ao pad numero 3
if ((pad[3] > lims[3])&&(trava[3] > atraso)) {
  tmrpcm.play("som3.wav");
  //delay(tempo);
  Serial.print("pad 3");
  Serial.println(pad[3]);
  trava[3] = 0;
}

//Toca o som0 referente ao pad numero 4
if ((pad[4] > lims[4])&&(trava[4] > atraso)) {
  tmrpcm.play("som4.wav");
  //delay(tempo);
  Serial.print("pad 4");
  Serial.println(pad[4]);
  trava[4] = 0;
}

//Toca o som0 referente ao pad numero 5
if ((pad[5] > lims[5])&&(trava[5] > atraso)) {
  tmrpcm.play("som5.wav");
```

```
//delay(tempo);
Serial.print("pad 5");
Serial.println(pad[5]);
trava[5] = 0;
}

//Toca o som0 referente ao pad numero 6
if ((pad[6] > lims[6])&&(trava[6] > atraso)) {
    tmrpcm.play("som6.wav");
    //delay(tempo);
    Serial.print("pad 6");
    Serial.println(pad[6]);
    trava[6] = 0;
}

//Toca o som0 referente ao pad numero 7
if ((pad[7] > lims[7])&&(trava[7] > atraso)) {
    tmrpcm.play("som7.wav");
    //delay(tempo);
    Serial.print("pad 7");
    Serial.println(pad[7]);
    trava[7] = 0;
}

//Toca o som0 referente ao pad numero 8
if ((pad[8] > lims[8])&&(trava[8] > atraso)) {
    tmrpcm.play("som8.wav");
    //delay(tempo);
    Serial.print("pad 8");
    Serial.println(pad[8]);
    trava[8] = 0;
}
```

```
//Toca o som0 referente ao pad numero 9
if ((pad[9] > lims[9]) && (trava[9] > atraso)) {
    if (digitalRead(hiHat) == HIGH) {
        tmrpcm.play("som9f.wav");
        Serial.print("fechado ");
    }
    else {
        tmrpcm.play("som9a.wav");
        Serial.print("aberto ");
    }
    //delay(tempo);
    Serial.print("pad 9");
    Serial.println(pad[9]);
    trava[9] = 0;
}

//Toca o som0 referente ao pad numero 10
if ((pad[10] > lims[10]) && (trava[10] > atraso)) {
    tmrpcm.play("soma.wav");
    //delay(tempo);
    Serial.print("pad 10");
    Serial.println(pad[10]);
    trava[10] = 0;
}

//Toca o som0 referente ao pad numero 11
if ((pad[11] > lims[11]) && (trava[11] > atraso)) {
    tmrpcm.play("somb.wav");
    //delay(tempo);
    Serial.print("pad 11");
    Serial.println(pad[11]);
    trava[11] = 0;
}
```

```
//Toca o som0 referente ao pad numero 12
if ((pad[12] > lims[12])&&(trava[12] > atraso)) {
    tmrpcm.play("some.wav");
    //delay(tempo);
    Serial.print("pad 12");
    Serial.println(pad[12]);
    trava[12] = 0;
}
```

```
//Toca o som0 referente ao pad numero 13
if ((pad[13] > lims[13])&&(trava[13] > atraso)) {
    tmrpcm.play("somed.wav");
    //delay(tempo);
    Serial.print("pad 13");
    Serial.println(pad[13]);
    trava[13] = 0;
}
```

```
//Toca o som0 referente ao pad numero 14
if ((pad[14] > lims[14])&&(trava[14] > atraso)) {
    tmrpcm.play("some.wav");
    //delay(tempo);
    Serial.print("pad 14");
    Serial.println(pad[14]);
    trava[14] = 0;
}
```

```
trava[0] = trava[0] + 1;
trava[1] = trava[1] + 1;
trava[2] = trava[2] + 1;
trava[3] = trava[3] + 1;
trava[4] = trava[4] + 1;
```

```
trava[5] = trava[5] + 1;  
trava[6] = trava[6] + 1;  
trava[7] = trava[7] + 1;  
trava[8] = trava[8] + 1;  
trava[9] = trava[9] + 1;  
trava[10] = trava[10] + 1;  
trava[11] = trava[11] + 1;  
trava[12] = trava[12] + 1;  
trava[13] = trava[13] + 1;  
trava[14] = trava[14] + 1;  
trava[15] = trava[15] + 1;  
}
```