FACULDADES INTEGRADAS DE CARATINGA

FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

HARDWARE LIVRE: DESCRIÇÃO, UTILIZAÇÃO E IMPLEMENTAÇÃO DE UM PEDAL PARA GUITARRA

GLAUBERT ANTUNES CARVALHO

CARATINGA

2011

Glaubert Antunes Carvalho

HARDWARE LIVRE: DESCRIÇÃO, UTILIZAÇÃO E IMPLEMENTAÇÃO DE UM PEDAL PARA GUITARRA

Monografia apresentada ao Curso de Ciência da Computação das Faculdades Integradas de Caratinga como requisito parcial para a obtenção do título de bacharel em Ciência da Computação, sob orientação do professor Jacson Rodrigues Correia da Silva.

Caratinga

2011

Glaubert Antunes Carvalho

HARDWARE LIVRE: DESCRIÇÃO, UTILIZAÇÃO E IMPLEMENTAÇÃO DE UM PEDAL PARA GUITARRA

Monografia submetida à Comissão examinadora designada pelo Curso de Graduação em Ciência da Computação como requisito para obtenção do grau de Bacharel.

_____
Prof. Msc. Jacson Rodrigues Correia da Silva
Faculdades Integradas de Caratinga

_____
Prof. Msc. Fabrícia Pires Souza Tiola
Faculdades Integradas de Caratinga

_____
Prof. Msc. Zamith França Neto
Faculdades Integradas de Caratinga

Caratinga, 19/12/2011

Dedico este trabalho a minha irmã *Gleiciane*,
nos momentos difíceis, me lembro de você.

AGRADECIMENTOS

Agradeço primeiramente a DEUS, pois ele me deu o dom da vida e a força para continuar a viver nos momentos onde sozinho eu não pude.

Aos meus pais ADAIR e FILOMENA, que me ensinaram as lições mais valiosas “os valores de um ser humano” e me permitiram herdar suas qualidades e aprender com suas falhas.

Ao meu irmão GLAYSSON que acompanhou de perto, sempre se preocupou e cuidou ainda sem perceber, agradeço à minha irmã GLEICIANE (*em memória*) por me ensinar uma lição que só se aprende da pior maneira, “que é preciso amar as pessoas como se não houvesse amanhã” (Renato Russo).

A minha namorada GIÓRGIA, que me inspirou a começar e também a chegar ao fim deste trabalho, por me mostrar em um momento muito difícil uma razão para continuar a viver.

Ao meu orientador JACSON, por fazer o possível e o que talvez fosse para outros impossível para me ajudar com este trabalho, pela paciência, compreensão e dedicação, virtudes que o tornam além de tudo um excelente professor.

Aos professores que foram mais que professores, foram meus amigos, exemplos de caráter, dedicação e força de vontade, FABRÍCIA, GLAUBER, JACSON, MARTA, MÍRIAM e PAULO EUSTÁQUIO, em vocês eu me inspiro para seguir o meu caminho, espero que um dia possa ser como vocês.

Aos meus amigos CARLOS e WALLACE que estiveram comigo no momento mais difícil de minha vida, aos meus companheiros de trabalho em especial o amigo GENILSON.

Ao amigo MICHAEL e também a um grande amigo, pois sem ele eu jamais chegaria até aqui, LUAN CARLOS.

Tão importante quanto chegar ao destino, é percorrer o caminho, muitas vezes nos atalhos os mais brilhantes se perdem, mas os persistentes e perseverantes , com seu real esforço chegam ao fim.

RESUMO

Na última década o movimento do software livre expandiu-se muito, contribuindo para que muitas ferramentas de qualidade fossem criadas, oferecendo uma alternativa de viável à tecnologia proprietária. A ideologia do software livre ganhou muitos adeptos e o fato de não ter que pagar altos valores para ter acesso a tecnologia a tornou comum entre usuários do campo científico e acadêmico.

Os resultados positivos do software livre contribuíram para o surgimento de um novo movimento, o hardware livre. O hardware livre se baseia na ideologia de liberdade do software livre.

Este trabalho apresenta um estudo sobre as possibilidades provenientes do uso do hardware livre na comunidade acadêmica como forma de apresentar as vantagens da utilização do mesmo nesta área. Para isso foi realizado um estudo sobre os diferentes modelos de hardware livre existentes e foram apresentados alguns exemplos dos mesmos.

Com o objetivo de testar as possibilidades do hardware livre na comunidade acadêmica, foi criado um pedal de efeitos para guitarra. Para a criação do mesmo foi utilizado o hardware livre Brasuíno. O Brasuíno é uma plataforma de prototipagem de eletrônicos baseada no Arduino. Este trabalho também reúne uma série de informações que tem como objetivo servir de material de apoio às pessoas que desejarem realizar experiências com o Brasuíno e outros modelos baseados em Arduino.

Palavras-chaves: Hardware livre, Brasuíno, Arduino, Pure Data, Pedal para Guitarra.

ABSTRACT

In the last decade the free software movement has expanded greatly, contributing to many quality tools were created, offering a good alternative to proprietary technology. The ideology of free software has won many fans and not having to pay high amounts to access the technology, makes it common among users of science and academia.

In recent years, the positive results of free software contributed to the emergence of a new movement, the free hardware. The free hardware is based on the ideology of free software.

This work presented a study on the possibilities arising from the use of free hardware in the academic community in order to presnt the advantages of using the same in this area. For this was realized a study on the different existing models of free hardware and some examples of them.

In order to test the possibilities of free hardware on the academic, was created an effects pedal for guitar, to create the same, was used Brasuino free hardware. The Brasuino is an electronics prototyping platform based on the Arduino.

This work also brings together a wealth of information that aims to serve as a support material to persons wishing to conduct experiments with Brasuíno and other models based on Arduino.

Keywords: Free hardware, Arduino, Brasuino, Pure Data, Guitar Pedal.

Índice de ilustrações

Índice de tabelas

LISTA DE SIGLAS

AC – Corrente Alternada

BSD – *Berkeley Software Distribution*

CPU – Unidade Central de Processamento

CVS – Sistema de versões concorrentes

DC – Corrente Contínua

IDE – Ambiente de Desenvolvimento Integrado

FM – Modulação em Frequência

GPL – *General Public License*

GPLV2+ – *General Public License Version v2+*

LED – Diodo Emissor de Luz

LGPL – *Lesser General Public License*

MIT – Instituto de Tecnologia de Massachusetts

OLED – Diodo Orgânico Emissor de Luz

ONU – Organização das Nações Unidas

PD – *Pure Data*

SD – *Secure Digital*

SIM – *Subscriber Identity Module*

TC – Transformador de Corrente

TCP – Protocolo de Controle de Transmissão

TI – Tecnologia da Informação

USB – *Universal Serial Bus*

Sumário

1 INTRODUÇÃO

Nos últimos anos, o movimento do software livre expandiu-se muito, tornando-se muito conhecido e tornando o software livre bastante utilizado, fato este que foi altamente propiciado pelo seu uso nas universidades e por pessoas interessadas em possuir sistemas e aplicações que fossem acessíveis a todos sem ser necessário desembolsar muito dinheiro para ter acesso a softwares de qualidade, tal que o comparasse aos maiores e mais utilizados softwares proprietários. Essa história é bem real ao observar-se os sistemas GNU/Linux, que hoje já fazem parte de laboratórios em diversas universidades e são utilizados em instituições governamentais, repartições públicas e organizações importantes da área de TI.

O software livre, a cada dia mais, está deixando de ser tendência para se tornar a melhor opção. Algo que pode ser predominante para que isso ocorra é a forte ideologia que está por trás destes projetos, a ideia de “ser livre”, apoiada por uma frase citada em 1975 pelo criador de um dos primeiros softwares livres: “Apoiemos nos ombros uns dos outros ao invés de pisar no pé” (OSIER-MIXON, 2010). A ideologia do software livre trouxe muitos adeptos a esse movimento, os quais contribuíram e contribuem para o seu crescimento. O grande sucesso desse movimento deu origem a um outro conceito, o de hardware livre.

Nos últimos anos têm-se buscado uma forma de aplicar os conceitos do software livre ao hardware, os defensores dessa ideia se apoiam na trajetória de sucesso trilhada pelo software livre e nos benefícios que podem surgir com a aplicação do conceito de liberdade ao hardware, mas se tratando do hardware alguns fatores podem ser apontados como limitadores para essa ideia. Uma questão inicialmente levantada ao tentar aplicar a ideia do software livre ao hardware é que o software pode ser facilmente duplicado e distribuído através da internet ou de outras formas sem que isso implique em custos financeiros, no caso do hardware, como é necessariamente composto de matéria, é impossível de ser produzido sem que sejam considerados os custos para tal.

O hardware livre tem a sua definição atual baseada na definição de open source para open source software (MÖLLER et al., 2011a), além de se basear na ideologia do software livre alguns dos fabricantes de hardware livre também utilizam as mesmas licenças utilizadas por softwares livres, como é o caso da OpenCores,

que utiliza as licenças GPL (General Public License), LGPL (*Lesser General Public License*) e BSD (*Berkeley Software Distribution*) em alguns de seus produtos (OPENCORES, 2011).

Este trabalho tem como objetivo difundir o conceito e as possibilidades do hardware livre no meio acadêmico, de forma que o mesmo possa beneficiar outros indivíduos desta área e despertar interesse daqueles que estão iniciando seus estudos sobre hardware livre. Para tentar alcançar este objetivo serão realizados testes com um modelo de hardware livre que é o Brasuíno. O Brasuíno é um hardware livre baseado em um projeto mundialmente conhecido, o Arduino.

Para realizar os testes com o Brasuíno, foi implementado um pedal para guitarra, o mesmo controla efeitos para os sons produzidos com a guitarra. Para isso, o pedal deverá ser conectado ao computador através do Brasuíno, que fará o controle de um *patch* PD (programa implementado no software Pure Data).

Este trabalho é também um material de apoio destinado àqueles que desejam realizar experiências com o Brasuíno ou outros modelos baseados em Arduino. Através dos resultados dos testes realizados nesse trabalho, são apresentadas as vantagens de se utilizar o hardware livre, em particular o Brasuíno, no intuito de tornar a experiência com o hardware mais agradável e interativa, apresentando como é possível criar objetos interessantes utilizando componentes básicos que muitas vezes podem ser extraídos de sucatas de eletrônicos. Essa experiência pode contribuir para o aprendizado em disciplinas teóricas, tal como circuitos lógicos, tornando possível a implementação de projetos que venham a interagir como o meio físico.

No capítulo 2 deste trabalho é apresentado um referencial teórico, onde na seção 2.1 são introduzidos conceitos necessários para o entendimento do hardware livre, as licenças e os tipos de componentes que são assim considerados, apresentando a vantagem da utilização dos mesmos e alguns exemplos de hardware livre existentes. A seção 2.2 faz uma referência ao Arduino, abordando brevemente seu histórico e o funcionamento desta plataforma, além de descrever alguns dos modelos existentes e alguns objetos criados com base em Arduino. A seção 2.3 faz referência ao Brasuíno, apresenta informações sobre como utilizá-lo e algumas possibilidades provenientes do uso mesmo. A seção 2.4 faz uma introdução sobre o áudio digital, e alguns das operações que podem ser efetuadas sobre o áudio digital. A seção 2.5 descreve brevemente o funcionamento do software PD e

apresenta alguns dos objetos utilizados na programação do mesmo, aplicando alguns conceitos apresentados na seção 2.4.

No capítulo 3, é abordada a metodologia utilizada para construção e utilização do pedal, apresentando um passo a passo de como o mesmo foi construído e o material necessário para tal. O capítulo 4 apresenta os resultados apresentados após a construção do pedal e uma análise sobre o Brasuíno e comparação de custos do mesmo com outras plataformas de prototipagem. O capítulo 5 faz a conclusão deste trabalho apresentada com base nos resultados e análises realizadas.

2 REFERENCIAL TEÓRICO

2.1 HARDWARE LIVRE

A definição de hardware de código aberto, tradução do termo *“Open Source Hardware”,* é um conceito recente e é altamente comum que a ele seja atribuído o nome “hardware livre”. De acordo com MÖLLER et al. (2011a), *“Open Source Hardware”* (OSHW) é um termo para artefatos tangíveis: máquinas, dispositivos ou outros objetos físicos cujo *design* foi disponibilizado ao público de modo que qualquer um pode construir, modificar, distribuir e utilizar estes artefatos.

O documento que define os “trabalhos culturais livres” está disponível em (MÖLLER et al., 2011b), definindo que não é uma licença, mas sim uma ferramenta para saber distinguir quais os trabalhos podem realmente ser considerados livres.

A nome hardware livre, diferente do que várias pessoas imaginam, não se refere ao preço, hardware livre não quer dizer hardware grátis. De acordo com a versão atual (1.1) da definição de hardware livre (MÖLLER et al., 2011b), o conceito de liberdade é definido como:

> liberdade de utilizar o trabalho e aproveitar os benefícios do seu uso; liberdade de estudar o trabalho e de aplicar o conhecimento dele adquirido; liberdade de fazer cópias e distribuí-las, em todo ou em parte, da informação ou expressão; e liberdade de fazer mudanças e melhoramentos, e de distribuir trabalhos derivados (MÖLLER et al., 2011b).

De acordo com OSIER-MIXON (2010), os maiores desafios para se aplicar os conceitos de software livre ao hardware são o custo para duplicar o hardware e o grande número de patentes existente. Mas apesar de ser impossível oferecer produtos grátis indefinidamente, deve-se lembrar que um produto físico é apenas a implementação de um design, que por sua vez pode ser disponibilizado gratuitamente através de uma licença aberta.

O outro grande impasse para o avanço do hardware livre é o grande número de patentes existentes sobre o hardware desenvolvido atualmente, no caso das *CPU's* por exemplo, existem atualmente mais de 100.000 patentes que cobrem até as instruções mais básicas, mas de acordo com Bruce Perens em COWBOY (2003), se referindo aos projetos desenvolvidos pela OpenCores, uma comunidade de desenvolvimento de hardware livre, o processo de contornar as limitações

impostas pelo interesse privado pode levar a resultados inovadores.

> Com os projetos da OpenCores, os engenheiros terão a oportunidade de fazer mais experimentos com o hardware do que nunca, ressaltou Perens. Nesse sentido, isto representa o mesmo tipo de mudança que vimos em 1994 no mundo do software. Os desenvolvedores, de uma hora para outra, passaram a poder programar em suas horas vagas, e veja só as implicações que isso teve COWBOY (2003).

O hardware livre, conforme já citado, tem como objetivo oferecer a liberdade a seus usuários para que possam usá-lo da forma que desejarem. Optar pelo hardware livre pode representar uma grande vantagem em diversos aspectos para diferentes usuários em diferentes situações, conforme será exibido na próxima seção. A logomarca oficial do hardware livre é apresentada na Figura 1.

Figura 1: Logo original do hardware livre
(ZLATAR 2011).

2.1.1 As Vantagens do Hardware Livre

Uma importante questão a ser compreendida é das vantagem provenientes do uso do hardware livre. Quando a mesma questão é remetida ao software livre, várias pontos podem ser inicialmente considerados, entre eles, benefícios como o custo financeiro, a liberdade de poder utilizá-lo e customizá-lo para atender a interesses próprios e a possibilidade de entendê-lo e estudá-lo (razão pela qual pode ser altamente vantajoso no campo científico e acadêmico). O hardware livre

compartilha de muitas dessas vantagens, e ainda que não tenha evoluído tanto quanto o software livre, devido a alguns limitadores, algumas dessas vantagens devem ser consideradas por aqueles que tenham interesse no uso e estudo do hardware. Abaixo são descritas de forma mais detalhada as vantagens de utilizar o hardware livre, em especial no campo acadêmico:

- Acessibilidade financeira: liberdade para utilizar o conteúdo existente sem pagar por registros ou licenças, o que o torna altamente viável para utilização em projetos interativos realizados nas bancadas de laboratórios , tais como robôs ou controle de sensores e atuadores.
- Liberdade de criar e registrar: poder criar e distribuir projetos utilizando de licenças geralmente já existentes para os dispositivos livres, possibilitando que tais projetos sejam vistos e melhorados por uma grande comunidade de usuários ao redor do mundo.
- Entender os componentes: assim como no software, poder saber como são realizadas as operações por trás da máquina, entender melhor como funcionam os componentes como processadores, controladoras, placas de vídeo, afim de estudá-los, entendê-los e melhorá-los.
- Contribuir com uma causa: ser colaborador de um projeto que pode trazer benefícios de diversas formas como é o caso do OpenEnergyMonitor (um projeto para desenvolvimento e construção de uma ferramenta de monitoramento de energia em código aberto) (OPENENERGYMONITOR, 2011).

A maioria desses benefícios não seria possível ao se tratar do hardware proprietário, cujos objetivos são geralmente influenciados pelo ganho financeiro, sendo assim necessário fazer uso de restrições legais e manutenção de forte sigilo sobre os processos utilizados na fabricação de seus componentes.

Atualmente alguns laboratórios de hardware disponíveis para o uso acadêmico, podem contar com recursos de hardware limitados, impossibilitando a realização de experiências de integração dos softwares criados e com dispositivos que possam realizar as funções desejadas. A introdução do hardware livre pode permitir maior acessibilidade a esse nível, criando ideias novas e possibilidade de torná-las mais reais.

Alguns exemplos de hardware livre são vistos a seguir:

- Ronja – Um projeto de tecnologia livre para um enlace de dados ponto-a-ponto. O dispositivo tem alcance de 1,4KM de distância e um link estável de 10Mbps *full duplex* (KULHAVÝ, 2011). A Figura 2 apresenta brevemente um esquema de funcionamento do Ronja em uma casa.

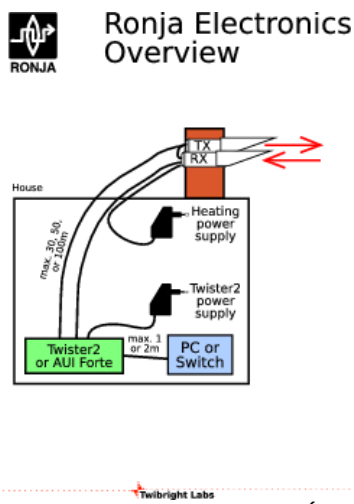


Figura 2: Ronja overview (KULHAVÝ, 2011).

- RepRap – Uma impressora 3D livre capaz de imprimir objetos de plástico. Como muitas peças da RepRap são feitas de plástico e ela pode imprimir essas partes, RepRap é uma máquina auto replicante, a RepRap pode ser construída por qualquer um com tempo e material necessário (REPRAP, 2011). A Figura 3 ilustra o RepRap Mendel (ao lado do computador), um dos modelos existentes da impressora 3D.

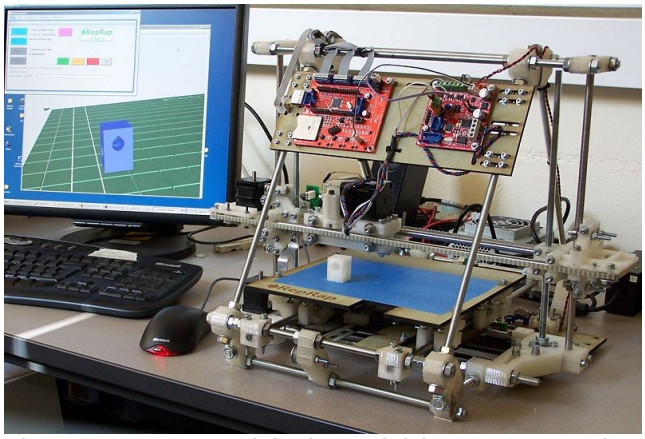


Figura 3: RepRap Original Mendel (REPRAP, 2011).

- Uzebox – O Uzebox é um console de jogos retrô (para jogos antigos) baseado em um microcontrolador AVR de 8 bits (UZEBOX, 2011). O Usebox é apresentado na Figura 4, sendo representado pela placa na qual o controle de videogame está ligado.

Figura 4: Uzebox AVCore & Gamecard (UZEBOX, 2011)

O estudo do hardware livre, em particular do Brasuíno, pode trazer a experiência e levar a criatividade a um novo nível, tornando possível a realização de experimentos com custo financeiro mais acessível e alta viabilidade do ponto de vista acadêmico, possibilitando aos membros do curso de Ciência da Computação das Faculdades Integradas de Caratinga, realizar experimentos práticos de integração de hardware e software, bem como melhorar práticas de laboratórios em disciplinas como Arquitetura de Computadores e Sistemas Lógicos.

A seção seguinte se objetiva a apresentar informações sobre o Arduino, apresentando as vantagens de utilizá-lo e também apresentando alguns projetos com base no mesmo, esse conteúdo também pode servir como base para usuários interessados em utilizar outros modelos compatíveis com o Arduino, tais como Roboduino, Freeduino e Brasuíno.

2.2 ARDUINO

Existem hoje diversos modelos de hardware livre disponíveis no mercado com os mais diferentes propósitos, entre eles um modelo criado no Instituto de Design Interativo Ivrea na Itália no ano de 2005, o Arduino (CALVO E ALAEJOS, 2011).

Segundo Máximo Banzi em (CALVO E ALAEJOS, 2011) o Arduino foi criado com o intuito de fazer uma ferramenta para os estudantes que fosse mais moderna e acessível que as ferramentas existentes no mercado naquele momento, principalmente porque a ferramenta que todos utilizavam, a chamada “Basic Stamp”, tinha o custo considerado alto para os estudantes. De acordo com Banzi, o Arduino foi criado para auxiliar no ensino de Design de Interação, uma disciplina que tem a sua principal metodologia a prototipagem, que consiste na criação de diferentes objetos que interajam com o meio físico. *“Existem muitas definições de Design de Interação, mas a que eu prefiro é: Design de Interação é o design de qualquer experiência interativa”* (BANZI, 2008).

O estudo do Design interativo se objetiva a estudar projetos que possam interagir com o ambiente real. Segundo NOBLE (2009), “interação pode ser definida como a troca de informações entre dois ou mais participantes ativos”.

Quando os termos interação e programação são relacionados, um dos elementos da interação é algum tipo de sistema de computador ou elemento de controle. A pessoa para quem o sistema interativo está sendo projetado é chamado de usuário, que por sua vez está utilizando o que é denominado sistema. Existem diferentes termos utilizados como: interação homem máquina ou *design* de experiência, esses termos geralmente se referem à mesma coisa, que é: projetar algum tipo de sistema que seja capaz de interagir com pessoas de maneira significativa (NOBLE, 2009).

O Arduino é uma plataforma de computação física de código aberto composta por uma placa simples de entrada/saída (E/S) e um ambiente de desenvolvimento que utiliza a linguagem Processing (BANZI, 2008). Ainda, segundo ARDUINO (2011a), Arduino é uma plataforma de prototipagem de eletrônicos, possui código aberto e é baseada em hardware e software de fácil utilização, sendo assim destinado a artistas, designers, hobbystas e qualquer pessoa interessada em criar

objetos ou ambientes interativos.

Desenhado para ser utilizado por iniciantes que não tem experiência com softwares ou eletrônicos, com o Arduino é possível criar objetos que respondem ao controle de luz, som, toque e movimento. O Arduino tem sido utilizado na criação de uma grande variedade de coisas, entre elas estão instrumentos musicais, robôs, jogos e esculturas de luz (MARGOLIS, 2011).

Através das entradas disponíveis na placa Arduino é possível fazer uma leitura do ambiente recebendo sinais de diferentes sensores e a partir dessas leituras, interagir com o ambiente controlando a luz, motores e outros atuadores. Um microcontrolador, uma espécie de microcomputador em um chip, é responsável por processar os sinais de entrada e enviar os retornos para as saídas, esse microcontrolador é programado através da IDE (Ambiente de Desenvolvimento Integrado) Arduino. Os projetos criados com Arduino podem funcionar de maneira independente ou comunicar-se com um software no computador (ARDUINO, 2011a).

De acordo com Maximo Banzi (um dos criadores do Arduino) em BANZI (2008), o Arduino se difere das outras plataformas nos seguintes aspectos: É um ambiente multi-plataforma; É baseado na IDE de programação Processing, um ambiente de desenvolvimento de fácil utilização, utilizado por artistas e designers; É programado através de um cabo USB (Universal Serial Bus), não uma porta serial, fato que pode ser importante já que muitos computadores modernos não têm portas seriais; É hardware e software livre, portanto é possível baixar o diagrama de circuito, comprar os componentes necessários e montá-lo por conta própria, sem pagar nada para os fabricantes do Arduino; Existe uma comunidade ativa de usuários, portanto, existem diversos usuários para auxiliar na criação de projetos; O Arduino foi desenvolvido em um ambiente educacional e portanto é viável para criar projetos com resultado rápido.

Existem atualmente diferentes placas oficiais Arduino em produção, cada qual com as suas propriedades específicas que as tornam aplicáveis a diferentes situações (ARDUINO, 2011b).

Os projetos de referência do Arduino são distribuídos sobre a licença Creative Commons, versão 2.5. Os termos da licença estão disponíveis em CREATIVE COMMONS (2011).

Existem atualmente vários modelos do Arduino, alguns são apenas revisões de versões anteriores (algumas descontinuadas) com alguns ajustes e melhorias,

existem também modelos que diferem entre si oferecendo diferentes funcionalidades e podendo ser utilizados em diferentes situações. As principais versões do Arduino estão disponíveis em ARDUINO (2011b). Versões mais antigas e informações sobre as mesmas podem ser encontradas em ARDUINO (2011f), a seguir são apresentados alguns dos modelos oficiais de placas Arduino e suas características básicas:

- Arduino Uno – Uma placa baseada no microcontrolador ATmega328 que possui 14 pinos de entrada/saída, o Arduino Uno é uma referência à plataforma Arduino, pode ser alimentado por uma conexão USB ou com uma fonte externa de alimentação (ARDUINO, 2011c).
- Arduino Ethernet – Também baseado no microcontrolador ATmega328, esse modelo possui um conector ethernet permitindo funcionalidades tais como comunicação com o protocolo TCP, possui também um leitor de cartão SD, que pode ser utilizado para disponibilizar arquivos na rede (ARDUINO, 2011d).
- Arduino BT – O Arduino BT foi criado inicialmente com base no microcontrolador Atmega168, mas é atualmente fornecido com o microcontrolador Atmega328 e o módulo *bluetooth* Bluegiga WT11. O Arduino BT pode ser alimentado por uma corrente de 1.2v a 5.5v, é utilizado para comunicação sem fio com telefones, computadores, ou outros dispositivos *bluetooth*, mas não é compatível com fones de ouvido *bluetooth* ou outros dispositivos de áudio (ARDUINO, 2011e)**.**
- Arduino Lilypad: O Arduino Lilypad foi designado para uso em dispositivos vestíveis (roupas, mochilas, acessórios de vestuário), foi projetado e desenvolvido pela professora Leah Buechley do MIT Media Lab e a empresa norte americana SparkFun Electronics (ARDUINO, 2011g).

Existem ainda, diferente dos modelos oficiais apresentados no site do Arduino outras diversas placas baseadas em Arduino montadas por empresas e pessoas em todo o mundo, algumas delas são modificadas de acordo com os projetos de seus montadores, outras são um clone exato de algumas versões do Arduino. Em FREEDUINO (2011) é possível ver uma lista com diversas placas compatíveis com Arduino.

Além das diferentes versões disponíveis do Arduino existem também os *shields*, que, segundo ARDUINO (2011h), são uma espécie de placa filha que pode

ser conectada sobre a placa Arduino expandindo assim as suas capacidades adicionando ao Arduino dispositivos como: receptores GPS, monitores LCD, módulos Ethernet e diversos outros encontrados em ARDUINO (2011i).

Uma lista completa de *shields* Arduino oficiais e não oficiais está disponível em OXER (2011), abaixo são apresentados alguns destes *shields* e descritas suas respectivas funcionalidades.

- rMP3 Playback Shield – Toca musicas/sons e lê/grava dados em cartão SD. O rMP3 permite a seu usuário integrar a reprodução de mp3 de alta qualidade com projetos de protocolo serial TTL, o rMP3 lê cartões SD ou SDHC nos formatos FAT12, FAT16 e FAT32, possui também um conector estéreo de 1/8 para fones de ouvido (ROGUE ROBOTICS, 2011).
- Arduino GSM Shield – Um *shield* GSM/GPRS que utiliza o módulo SYMCOM SIM900. Um exemplo de funcionalidade desse *shield* seria: ativar ou desativar um dispositivo eletrônico (como uma câmera de vídeo ou alarme) através de um SMS. O GSM *shield* possui um *slot* para cartão SIM (chip utilizado em telefone celular) e um conector para antena de sinal, o GSM shield pode ser visto em ELETTRONICA IN (2011).
- TouchShield Slide – *Shield* com *display widescreen* OLED *touch screen* para o Arduino com resolução de 320x240. O TouchShield Slide roda comandos de processamento gráfico e pode ser utilizado na criação de dispositivos tais como games portáteis e adicionar outras funcionalidades gráficas ao Arduino (LIQUIDWARE, 2011).

Os propósitos para os quais o Arduino é utilizado são variados, mas grande parte deles segue um mesmo padrão, o dos dispositivos interativos. Um dispositivo interativo basicamente recebe uma leitura de um sensor, processa a resposta e envia um sinal a um determinado atuador, que vai interagir com o ambiente, conforme apresentado na Figura 5.

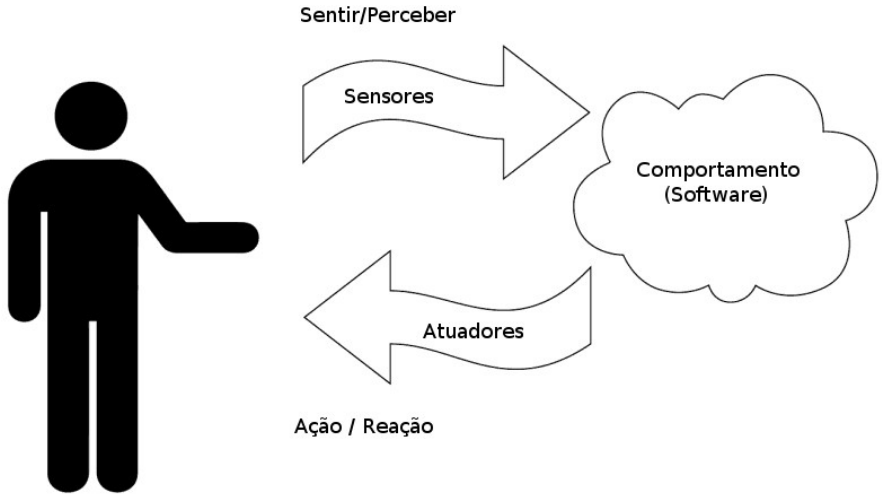


Figura 5: Dispositivo Interativo (BANZI, 2011).

O Arduino já é utilizado como base em muitos projetos em todo o mundo, esses projetos variam desde experimentos simples para fazer acender alguns LEDs até projetos que podem ser muito úteis de diferentes formas, abaixo são apresentados alguns projetos baseados em Arduino:

- Alarma de Sismos – Um projeto criado por um jovem chileno que detecta e avisa sobre possíveis terremotos no Chile com alguns segundos de antecedência através da conta “@AlarmaSismos” no twitter, o projeto utiliza um detector de terremotos doméstico que é conectado através de um circuito a uma placa Arduino, essa placa é ligada ao servidor de internet e envia as mensagens pelo twitter (HEIM, 2011). A Figura 6 ilustra o AlarmaSismos, na parte de cima o detector de terremotos e abaixo a placa Arduino.

Figura 6: Alarma Sismos (HEIM, 2011)

- Drum Master – O Drum Master é uma espécie de controlador de drum (tambores/instrumentos de percussão tal como a bateria) composto de duas partes: o módulo de hardware (composto por uma placa Arduino e uma coleção de circuitos para auxiliar na obtenção das informações dos sensores) que é ligado a um computador através de uma porta USB e o software Slave Drum, escrito em Python. Quando um sensor é acionado, o Drum Master converte o sinal recebido do mesmo para um valor digital e envia esse valor (e a porta na qual o sensor foi detectado) para o software, que reproduz a amostra de áudio correspondente ao sinal recebido (ARDUINO, 2011j). O Drum Master atua como um módulo utilizado em baterias eletrônicas, a Figura 7 ilustra o hardware do Drum Master.

Figura 7: Drum Master (ARDUINO, 2011j)

- Open Energy Monitor – Um projeto para desenvolver e construir ferramentas de código aberto para monitoramento e análise de energia, eficiência energética e geração renovável distribuída. O monitor de energia AC pode medir a energia utilizada por uma casa ou edifício inteiro, ele usa um sensor de transformador de corrente (TC) para medir a corrente no circuito elétrico e um adaptador de energia AC-AC para medir a tensão da rede. A partir dessas medidas, o Arduino calcula: Potência real, Potência Aparente, Fator de Potência, frequência e potência em kwh. Os valores podem ser enviados para um computador para gerar registros e gráficos, podem ser exibidos em um monitor lcd de 7", registrados em

um *pendrive* ou enviados para a internet através de um *shield* ethernet ARDUINO (2011j). Um esboço do Open Energy Monitor é apresentado na Figura 8.

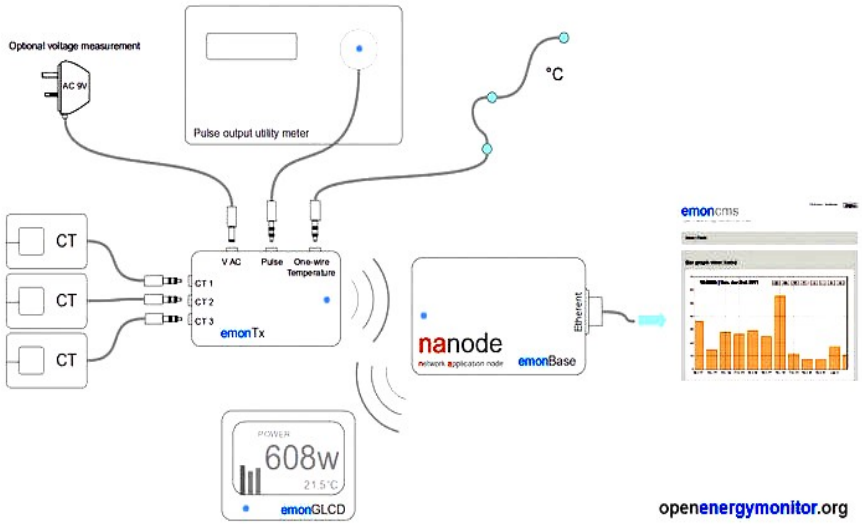


Figura 8: OpenEnergyMonitor (OPENENERGYMONITOR, 2011).

Outros projetos com Arduino podem ser encontrados em ARDUINO (2011j), o Arduino pode ser adquirido através da loja Arduino ou através de algum dos distribuidores cujos contatos estão disponíveis em ARDUINO (2011m).

Como o Arduino é hardware livre é possível montá-lo a partir das especificações disponíveis na seção de hardware no site do fabricante, não sendo necessário comprá-lo (ARDUINO, 2011b). Os componentes necessários para montagem da placa são geralmente encontrados em lojas de eletrônica.

Conforme descrito nos termos da licença, aqueles que desejarem também montar sua própria versão do Arduino e comercializá-lo poderão fazê-lo, existe apenas a limitação para os que desejarem utilizar o nome “Arduino”, para estes, faz-se necessário conhecer e seguir as regras sugeridas pela comunidade e equipe de desenvolvimento, disponíveis em ARDUINO (2011k).

2.2.4 Lixo Eletrônico e Arduino

Um grande problema que vêm sendo debatido nos últimos anos é o crescente aumento do volume de lixo eletrônico produzido nos países emergentes, o gráfico a seguir ilustra essa questão.

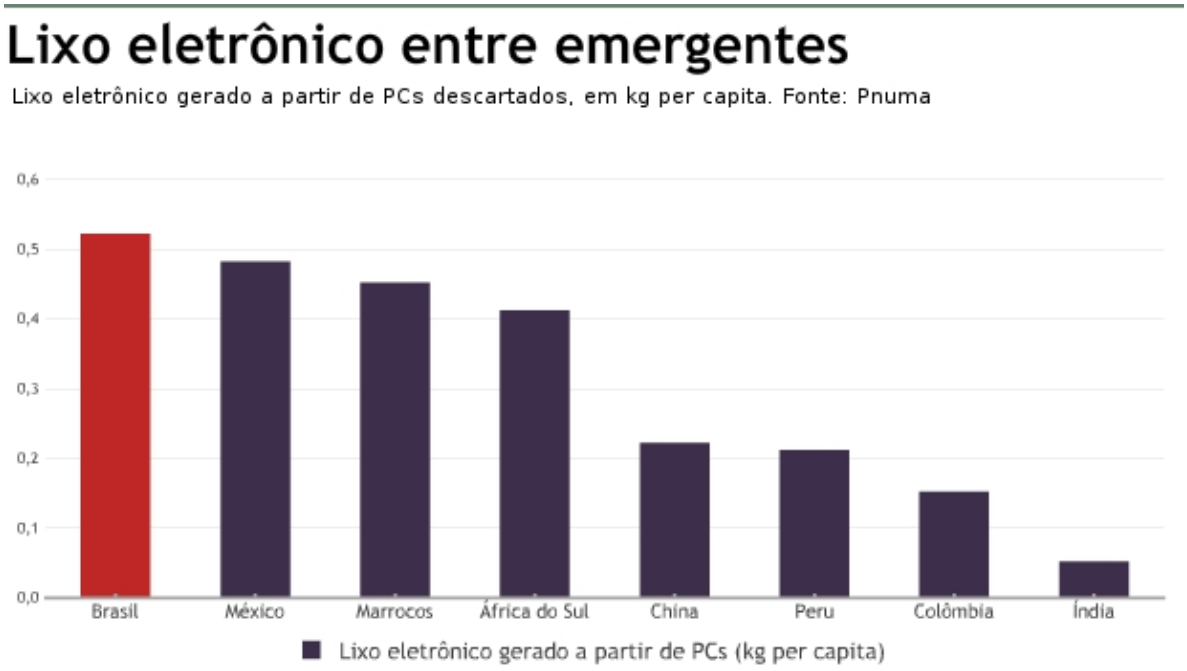


Figura 9: Lixo eletrônico entre emergentes (CHADE, 2010).

No gráfico apresentado na Figura 9, o Brasil é apresentado como o pais que produz o maior volume de lixo eletrônico (em Kg per capta) gerado a partir do descarte de PCs, segundo estudo realizado pelo programa da ONU para o meio ambiente disponível em JAMIL (2010), isso torna a necessidade de tratamento desta questão ainda mais visível, sendo assim, o aproveitamento do lixo eletrônico se torna não só viável, mas também necessário para redução dos impactos causados pelo mesmo.

Cada vez mais, equipamentos eletrônicos são inseridos no mercado sem que sejam previstas medidas para o fim do seu ciclo de vida, o destino dado a esses equipamentos geralmente é o lixo. No entanto, essa não é a opção mais viável segundo BEIRIZ (2011), de acordo com o autor, o motivo pelo qual as pessoas

descartam irregularmente esse tipo de lixo é o desconhecimento dos impactos que o mesmo pode causar ao meio ambiente e à saúde humana, além de desconhecerem o fato de que grande parte dos objetos descartados poderiam ser reaproveitados de alguma forma.

Os usuários de projetos *open source* tal como o hardware livre geralmente são adeptos à ideia de reaproveitamento desse tipo de material (lixo eletrônico), que pode ter diferentes destinos nas mãos de pessoas criativas, é o que pode ser visto em SILVA (2011), que cita uma reportagem realizada por Marisa Silva é exibido um exemplo da reutilização de material que seria por outros considerado “lixo”. O clube Vega, de São Paulo, criou um projeto com base em Arduino para controlar seus efeitos de iluminação, são mais de duas mil luzes que compõem uma tela de 40m² que é controlada através de um software *open source,* nesse projeto são utilizados vários processadores usados, além de milhares de metros de fio reaproveitado.

As seções a seguir são destinadas a criação de um material de apoio, para os usuários interessados em fazer experiências com o Brasuíno ou outro modelo baseado em Arduino.

2.3 BRASUÍNO O ARDUINO DO BRASIL

Entre os vários clones e derivações do Arduino, um modelo em específico é utilizado como ferramenta neste trabalho, o “Brasuíno”. A escolha do Brasuíno para este trabalho foi motivada por diferentes fatores que são mencionados a seguir:

- Em primeira instância a opção de uma ferramenta livre com uma versão brasileira conforme anunciado por seu fabricante, o Brasuíno utiliza o prefixo “BRAS” como forma de oficializar a sua origem (HOLOSCÓPIO, 2011a);
- O Brasuíno é produzido por uma empresa que apoia fortemente a causa dos projetos livres, a Holoscópio Tecnologia. A Holoscópio contribui para o Kernel Linux com correções de falhas, criação de drivers de dispositivos e novas funcionalidades para o sistema;
- A Holoscópio incentiva a reutilização de equipamentos eletrônicos

sucateados, como forma de reduzir os impactos causados por estes ao serem descartados indevidamente, lembrando que muitos componentes que podem ser úteis em projetos criativos podem ser adquiridos nesse tipo de material;

- O Brasuíno foi criado com o intuito de fazer uma ferramenta de fácil utilização para usuários hobbystas e estudantes e é recomendável para utilização em projetos de automação e robótica;
- Possui um custo viável se comparado a seus demais concorrentes, conforme apresentado na Tabela 4 da seção 4;
- O Brasuíno é 100% livre e os esquemáticos estão disponíveis na página oficial do Brasuíno (HOLOSCÓPIO, 2011a);
- Possui suporte em português e em breve disponibilizará material de apoio ao usuário iniciante;

Os fatores citados podem favorecer para que o Brasuíno seja uma plataforma compatível com as necessidades em laboratórios de hardware e em experiências acadêmicas, como criação de robôs e automatização de objetos ou ambientes, o fato de ser uma versão brasileira (com suporte em português) pode facilitar o acesso para iniciantes em hardware e experiências com dispositivos interativos.

A versão do Brasuíno disponível atualmente é o BS1. O BS1 é um hardware baseado no Arduino Uno e é também compatível com os *shields* do Arduino Uno e o software Arduino, detalhes sobre a utilização do Brasuíno BS1 podem ser vistos na seção 2.3.1, o Brasuíno BS1 pode ser adquirido pela loja da Holoscópio em HOLOSCÓPIO (2011b) e a Figura 10 ilustra um exemplo desta placa na cor preta (eclipse).

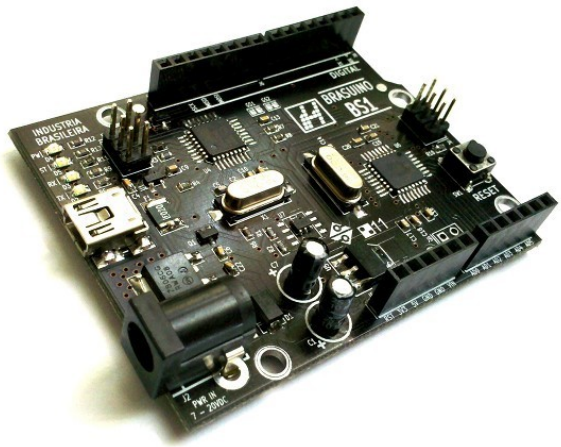


Figura 10: Brasuíno (HOLOSCÓPIO, 2011b)

2.3.1 Utilizando o Brasuíno BS1

A configuração de hardware do Brasuíno BS1 é apresentada na Tabela 1:

Tabela 1: Configuração do Brasuíno BS1.

Microcontrolador	ATMEGA328
Clock	16MHz
Memória EEPROM	1KiB
Memória RAM	2KB
Memória Flash	32KB
Voltagem entrada(limite)	7 a 20 volts
Entrada/Saída Digital	14 (6 com PWM)
Entradas Analógicas	8

O Brasuíno BS1 é equipado com o Atmega328, que é um microcontrolador de 8 bits e possui 32KB de memória que é utilizada para o armazenamento do código carregado na placa, possui entradas e saídas analógicas e digitais para se comunicar com os sensores e atuadores, o mesmo pode ser alimentado por uma fonte de corrente contínua de 7 a 20 volts.

Conforme mencionado anteriormente, o Brasuíno BS1 é uma plataforma compatível com o software Arduino, fato este que o torna favorável a usuários que não tenham grande conhecimento em programação. A IDE Arduino é um ambiente de desenvolvimento interativo que é utilizada para desenvolver o código que será portado ao Brasuíno, possui uma interface simplificada na qual estão disponíveis as opções de compilar o código e fazer a transferência do mesmo para a placa.

Para começar a utilizar o Brasuíno é necessário um computador com sistema operacional Linux, Windows ou Mac, um cabo USB A x mini B e uma placa Brasuíno BS1. Para programar o Brasuíno é necessário instalar o software Arduino e conectar a placa ao computador através de uma interface USB. A IDE Arduino é um ambiente escrito em Java e a linguagem de programação Arduino é baseada em C/C ++ (ARDUINO, 2011l), sendo assim aqueles que já tiveram algum contato com a linguagem C, terão facilidade em se identificar com a mesma. A IDE conta com alguns exemplos que são códigos prontos, bastando apenas fazer a transferência dos mesmos, os exemplos realizam operações tais como acender um LED, reproduzir um som, acionar um servo-motor. Um guia de instalação e configuração

do software Arduino está disponível no Anexo 1. Na próxima seção são apresentados os passos básicos para utilização da IDE Arduino com os exemplos fornecidos pela ferramenta.

2.3.2 Programando o Brasuíno

A IDE Arduino permite programar diferentes versões do Arduino e compatíveis, por isso é necessário selecionar a placa a ser programada, nesse caso, o Brasuíno BS1, que é uma versão compatível com o Arduino Uno, portanto basta selecionar no menu “Tools->Board->Arduino Uno”. Depois de selecionar a placa, é necessário escolher a porta de comunicação da mesma com o computador em “Tools->Serial Port”, a porta poderá variar de acordo com o hardware ou software do computador.

A Figura 8 exibe a IDE Arduino, explicando algumas das opções básicas do software. O *sketch* (programa em linguagem Arduino) apresentado na Figura 11 é executado como teste no Brasuíno BS1, esse código é um dos exemplos disponíveis na própria IDE, os exemplos podem ser encontrados no menu File->Examples.

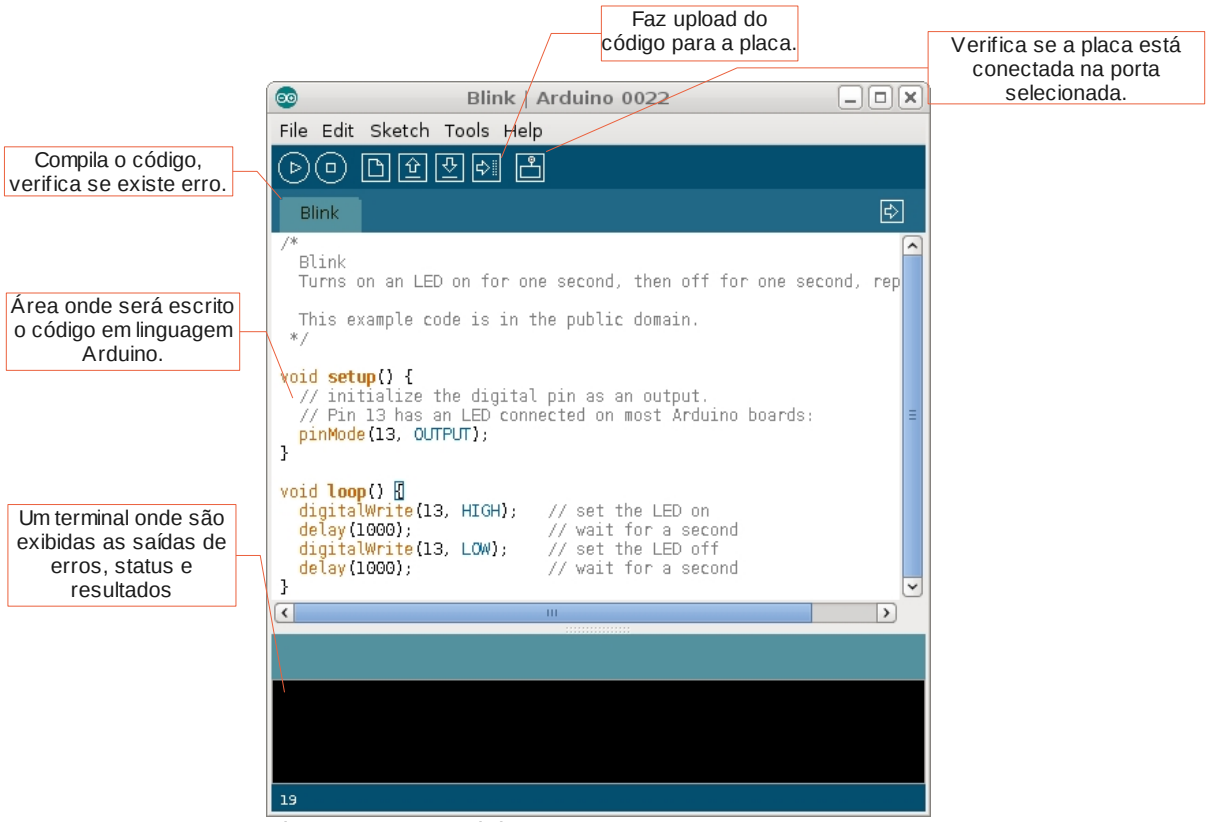


Figura 11: IDE Arduino

Pode-se dizer que as linhas de código acima representam um “Hello World (exemplo básico citado em muitas linguagens de programação)” no Brasuíno. Ao fazer a transferência do código para a placa, o Brasuíno se comunicará com o usuário piscando o LED “ST (Um dos componentes originais da placa BS1)”.

O Brasuíno pode ser utilizado ligado a um computador para controlar algum software ou ferramenta através das leituras recebidas e pode também ser utilizado de forma independente. Depois de fazer a transferência do código para a placa, a mesma pode ser desconectada do computador e inserida no objeto criado, sendo alimentada através de uma fonte de energia externa (como uma bateria de 9V).

Existem diversos desenvolvedores que se dedicam à criação de novas bibliotecas para o software Arduino, a exemplo dessas, a Firmata, que tem como objetivo permitir a comunicação do Arduino com outros softwares.

2.3.3 A Firmware Firmata

O Firmata é um protocolo genérico para comunicação com microcontroladores através de um computador. Existem versões do Firmata para diferentes linguagens de programação, o objetivo do Firmata é permitir aos seus usuários controlar diferentes softwares de computador utilizando o Arduino (FIRMATA, 2011). A biblioteca Firmata está disponível no software Arduino a partir da versão 0012, os exemplos podem ser acessados através do menu File->Examples->Firmata.

A próxima seção faz a definição de alguns conceitos necessários ao entendimento de como os efeitos do pedal para guitarra funcionam, e como os mesmos deverão ser implementados de forma a produzir o resultado desejado.

2.4 O ÁUDIO DIGITAL

O som é um fenômeno da física formado pelas vibrações no ar, alguns

instrumentos são comumente utilizados para vibrar o ar em frequências específicas, como as cordas de um violão, os tambores de uma percussão e pratos de uma bateria. Existe uma membrana no ouvido humano, que vibra de acordo com as vibrações do ar, o tímpano, o cérebro transforma estas vibrações no que é denominado som (KREIDLER, 2011).

Um termo comumente utilizado no áudio digital é o termo oscilador, os osciladores são definidos por PRINCIC et al. (2011), como “*os geradores de sinais básicos da música eletrônica*”. Através da combinação, filtragem ou modulação de osciladores praticamente qualquer som pode ser criado.

Segundo KREIDLER (2011), na música eletrônica os alto falantes são utilizados para gerar o som, os falantes possuem uma ou mais membranas que vibram para frente e para trás, fazendo com que o ar vibre também. As vibrações desta membrana podem ser controladas pelo computador, mas, como o som é um fenômeno físico ele é portanto, analógico. Os computadores por outro lado, são digitais, sendo assim trabalham apenas com números, para que seja possível efetuar operações como a análise e síntese de som no computador, é necessário que seja feita uma conversão do sinal analógico para o digital, e depois, uma conversão contrária (digital para analógico) para que o áudio possa ser percebido pelo dispositivo de saída de som, essa operação de conversão de sinais é feita pela placa de som. A Figura 12 ilustra como esta conversão de sinais é realizada.

> Primeiro, imagine um alto-falante. Ele move o ar na frente dele e faz um som. A membrana do alto-falante vibra de sua posição central (em repouso) para trás e para frente. O número de vezes por segundo com que ela vibra faz a frequência (a nota) do som que você ouve, e a distância que viaja a partir de seu ponto de repouso determina o ganho (o volume ou intensidade) do som. Normalmente, nós medimos frequência em Hertz (Hz) e a intensidade ou ganho em decibéis (dB) (PRINCIC et al., 2011).

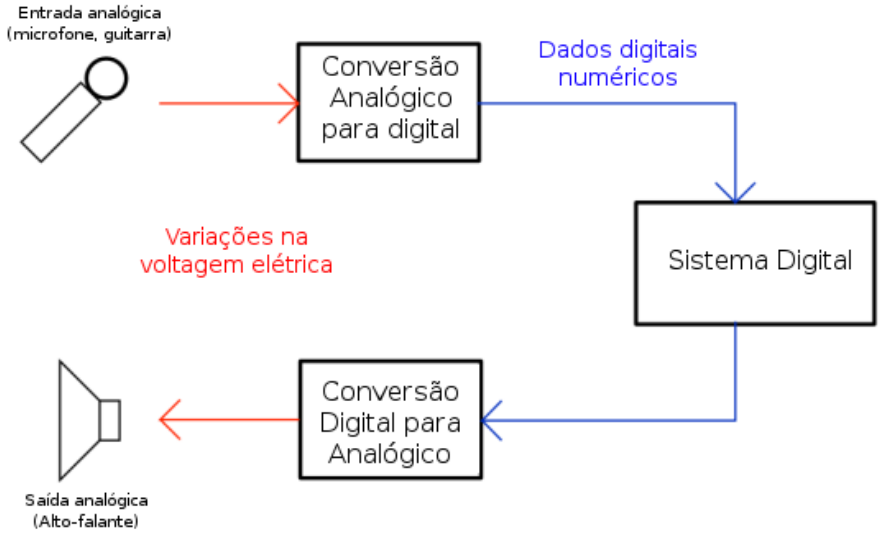


Figura 12: Conversão analógico/digital em um computador (PRINCIC et al., 2011).

Em Pure Data, sinais de áudio são representados por uma sequência de números que variam no intervalo de [-1, 1]. Assim, a forma de onda de cada oscilador foi programado para enviar os valores dentro desta faixa. O nome de cada oscilador refere-se a sua forma de onda, que é a forma de um período (ou um Hertz) desse oscilador. Formas de onda diferentes fazem sons diferentes.

2.4.1 Operações e Funções no Áudio Digital

Como já citado anteriormente, o sinal digital é composto apenas por números, portanto, todas as técnicas de efeito ou síntese no áudio digital consistem em operações matemáticas no sinal digital (PORRES, 2009). Nas seções a seguir são apresentados alguns tipos de operações efetuadas no áudio digital.

2.4.1.1 Onda Senoidal

Através de uma função, é possível gerar uma forma de onda no computador, a exemplo, a função seno, que gera um tom puro. A Figura 1.6 é um exemplo no qual o valor de Y (marcado em vermelho) varia de acordo com o ângulo alfa. Pode-se também fazer a analogia de que a variação de Y indica o deslocamento de um alto- falante do seu ponto de repouso (PORRES, 2009).

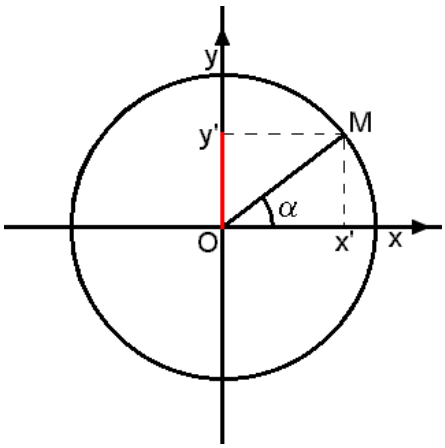


Figura 13: "*Y*" em função do ângulo alfa (PORRES, 2009).

2.4.1.2 Modulação em Anel

Segundo PORRES (2009), “a *Modulação em Anel ocorre quando multiplica-se um sinal de Áudio por outro*” e pode ser usada tanto para um efeito sonoro como também para Síntese (um tipo de operação realizada sobre o áudio). Ao ouvir dois tons seno que se aproximam entre sí, ouve-se cancelamentos da onda flutuante. Este fato ocorre devido à interação de duas ondas quase idênticas. A este fenômeno dá-se o nome de batidas. No exemplo a seguir são dadas ondas que oscilam entre 4 e 5 Hz, as ondas alternam entre somatórios e cancelamentos.

Na Figura 14 é apresentada uma onda que oscila em 4hz e na Figura 15 uma onda de 5hz, na Figura 16 o somatório das duas ondas anteriores e na Figura 17 o

somatório das ondas em um período de tempo mais longo. O resultado dessas operações é um aumento de pulsação e queda em volume (KREIDLER, 2011). Um exemplo de implementação da modulação em anel será apresentado mais adiante na seção 2.5 na Figura 21.

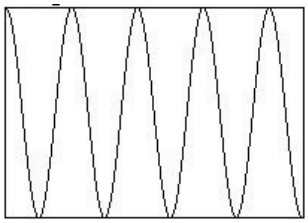


Figura 14: Onda 5hz (KREIDLER, 2011).

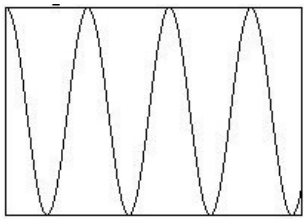


Figura 15: Onda 4hz (KREIDLER, 2011).

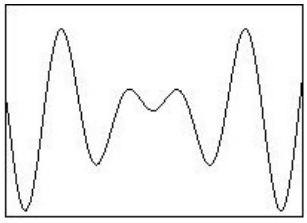


Figura 16: Somatório das ondas de 4 e 5Hz (KREIDLER, 2011).

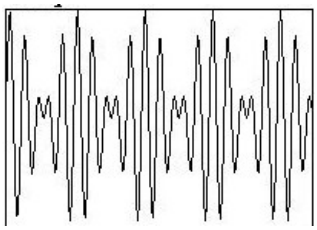


Figura 17: Somatório das ondas de 4 e 5Hz em período de tempo mais longo (KREIDLER, 2011).

2.4.1.3 Modulação de Freqüência (FM)

A síntese de modulação de frequência ou síntese FM, é usado para fazer alterações periódicas na frequência de um oscilador para obter uma frequência modulada. Para tal, basta apenas utilizar um oscilador para mudar a frequência de outro oscilador, na forma mais simples da Frequência Moduladora são utilizados dois osciladores, o primeiro é denominado portador (*carrier*), neste oscilador a frequência portadora (*carrier frequency*) será alterada ao longo do tempo. O segundo oscilador é o modulador, que vai alterar o valor da frequência de modulação (*modulation frequency*) (PRINCIC et al., 2011).

Ainda segundo Princic et al. (2011), ao utilizar um valor muito pequeno para a modulação, será ouvido uma vibração (*vibrato*) da frequência portadora. Caso a

quantidade de modulação seja grande, será ouvido um glissando (*sweeping)* da frequência portadora. A rapidez com que as mudanças são ouvidas na frequência portadora é determinada pela frequência moduladora.

Na síntese FM, é possível também ajustar o ganho do oscilador que controla a frequência de modulação, a esse ajuste de ganho é atribuído o nome de índice de modulação (*modulation index*). Assim sendo, pode-se dizer que os três parâmetros utilizados para modulação de frequência são: frequência portadora, frequência de modulação e índice de modulação (PORRES, 2009). A Figura 18 é um exemplo de onda com síntese FM, um exemplo de sua utilização é apresentado na Figura 22 da seção 2.5.

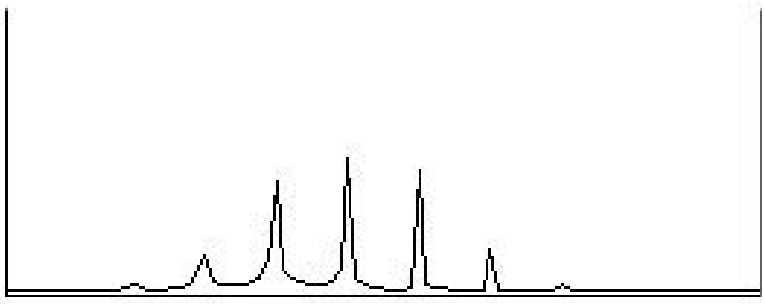


Figura 18: Exemplo de onda com síntese FM (KREIDLER, 2011).

2.5 PURE DATA

“PD (aka Pure Data) é um ambiente de programação gráfica em tempo real para áudio, vídeo e processamento gráfico” (PD, 2011). O núcleo do PD é escrito e mantido por Miller Puckette e inclui também o trabalho de vários desenvolvedores. O PD é software livre e também possui o benefício de ser multiplataforma, existem versões disponíveis para vários SO e o código fonte pode ser baixado do CVS.

O PD é comumente utilizado para criação de efeitos sonoros, música ao vivo, VeeJaying (performance/exibição de efeitos visuais em tempo real), análise de áudio, composição e controle de sensores e robôs, isso é possível pelo fato de que esses diferentes meios de comunicação são tratados como dados digitais dentro de um programa (PRINCIC et al., 2011).

A Programação no PD é realizada de forma muito interativa, o que faz com que o usuário se sinta muito mais próximo da manipulação de objetos do mundo real. A unidade mais básica do PD é uma “caixa”, a programação é feita “ligando” essas caixas de forma a criar diagramas que representem o fluxo de dados e ao mesmo tempo realizem as operações traçadas no diagrama. Como a programação ocorre em tempo real, o programa está sempre em execução (quando ativada a opção “*compute audio*” dentro do software), a criação do código ocorre no mesmo momento em que ele é executado e cada ação a ser realizada pelo programa terá efeito no momento em que for concluída (PRINCIC et al., 2011).

O PD foi utilizado nesse trabalho para criação de um *patch* que é controlado através do Brasuíno BS1, esse *patch* deve receber os sinais de áudio de uma guitarra que será conectada ao computador através de uma placa de som e distorcer o sinal de saída de áudio, que será percebido através do dispositivo de saída de áudio do computador, a Figura 19 ilustra a tela principal do PD, essa tela é um tipo de terminal que exibe o status das ações do sistema para o usuário. A Figura 20 é um exemplo de um *patch* PD, que apresenta como fica um programa pronto.

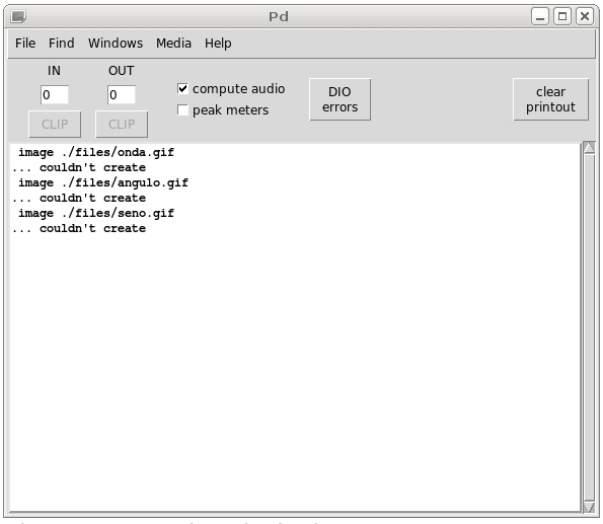


Figura 19: Janela Principal PD.

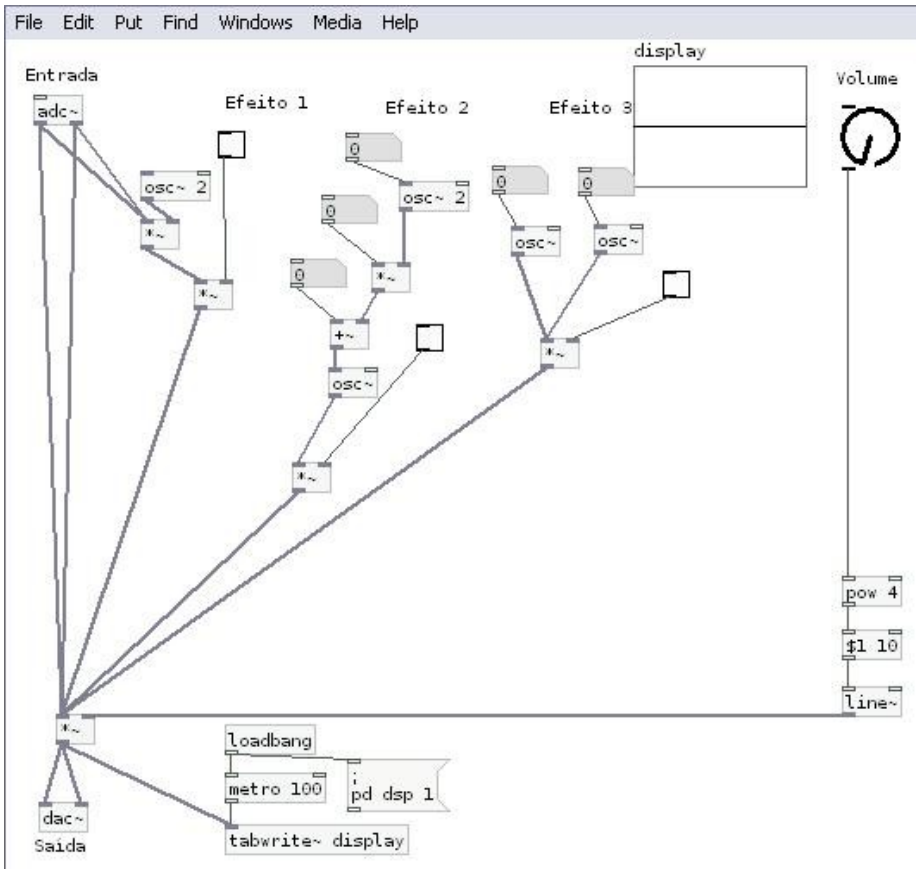


Figura 20: Exemplo de Programa PD.

2.5.1 Componentes do PD

Para o entendimento pleno de como foram criados os efeitos para o pedal, é necessário o conhecimento de algumas unidades do PD, as mesmas serão apresentadas a seguir e uma breve descrição sobre elas será apresentada:

- Objeto – Existem diferentes tipos de objeto no PD, cada tipo de objeto tem uma funcionalidade, os objetos podem ser “operadores (+, -, *, =)”, “variáveis (float, int, symbol)”, “funções (print, bang, route)” e outros que podem ser vistos acessando a ajuda do PD;
- Toggle – é um tipo de objeto que possui uma forma gráfica, tem uma entrada e uma saída e funciona com uma chave liga/desliga. Quando ativado o mesmo passa um valor (diferente de zero) em sua saída, esse

valor pode ser setado nas propriedades do objeto. O toggle pode receber uma entrada de outro objeto para ativá-lo/desativá-lo;

- Bang – um objeto e também possui uma forma gráfica, ao ser acionado, ele envia um sinal para a saída (o mesmo que recebeu na entrada), o bang possui uma entrada e uma saída. O objeto Bang pode ser utilizado para acionar o toggle;
- Number – uma caixa que exibe valores numéricos, os valores podem ser setados manualmente ou recebidos através da entrada, a saída de number envia o valor apresentado naquele instante na caixa;
- Array – uma Tabela com um vetor gráfico, a medida que os valores escritos nessa Tabela variam a linha do gráfico oscila, formando uma espécie de gráfico com valores dinâmicos atualizado em tempo real.
- Knob - um objeto gráfico similar a um controle de volume, pode ser utilizado para o controle de valores entre x e y.

2.5.2 Objetos PD

A seguir a descrição dos objetos utilizados no *patch* PD:

- adc~ - Conversor analógico para digital, recebe um sinal da entrada de áudio do computador e converte o mesmo em sinal digital, enviando esse sinal através de suas saídas, possui duas saídas sendo possível controlar o som estéreo;
- *~ - Um operador em sinais de áudio que permite multiplicar o sinal por outro objeto, possui duas entradas, permitindo ligar o sinal de áudio a uma delas e adicionar um controle de volume através da outra entrada. Possui também uma saída, na qual sairá o sinal de áudio multiplicado pelo objeto inserido na entrada;
- line~ - Objeto para fazer as alterações no áudio ocorrerem de forma linear, um exemplo é o controlador de volume, que para chegar ao máximo tem que percorrer todos os valores entre o mínimo e o máximo, possui duas

entradas para receberem os valores entre os quais vão oscilar e uma saída para para enviar o sinal linear;

- osc~ - O objeto osc~ cria uma onda senoidal, possui duas entradas, uma para definir a fase e a outra para a frequência, a saída de de osc~ é um sinal de áudio em formato de onda senoidal;
- loadbang - Não possui entrada, envia um bang através de sua saída assim que o *patch* é inicializado;
- metro - Envia uma série de bangs em uma taxa constante através de sua saída, possui duas entradas para setar o tempo de envio;
- tabwrite~ - O objeto tabwrite~ recebe como argumento o nome de uma Tabela, nesta Tabela serão escritos os dados recebidos na entrada, *ex.: tabwrite~ Tabela1*;
- dac~ - Conversor digital para analógico, recebe um sinal digital em suas entradas e o converte para analógico, enviando essa saída para o dispositivo de áudio (placa de som) do computador, permitindo ouvir o som em caixas de som por exemplo.
- arduino - O objeto arduino não é um objeto nativo do PD, este objeto foi criado com o objetivo de permitir que o PD se comunique com a placa Arduino através do protocolo Firmata. Conforme já mencionado, o Firmata pode ser gravado no Brasuíno, uma vez que o Firmata está carregado na memória do Brasuíno, basta adicionar o objeto arduino no *patch* PD, que será possível enviar e receber mensagens ao Brasuíno com a utilização dos devidos objetos, para utilizar o objeto arduino deve ser adicionado ao PD o *patch* “Pduino”, disponível em STEINER et al., (2011).

2.5.3 Sons e Efeitos no PD

Como a programação em PD é feita ligando caixas, basta apenas unir algumas caixas para criar determinados efeitos ou sons, se em algum instante esses efeitos são unidos ao objeto adc~, os efeitos serão inseridos sobre a entrada de som determinada. As figuras 21 e 22 são exemplos em PD de algumas das operações

efetuadas sobre o áudio que foram descritas na seção 2.4.1.

Modulação em Anel:

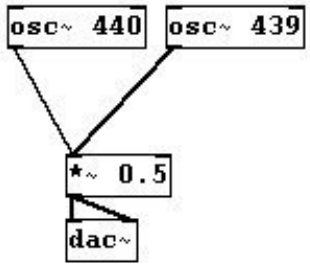


Figura 21: Modulação em anel (KREIDLER, 2011).

Modulação de Freqüência (FM) :

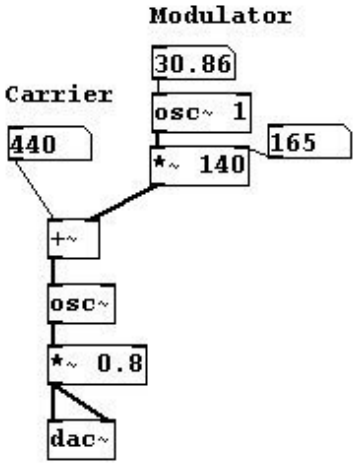


Figura 22: Modulação de frequência (KREIDLER, 2011).

Filtros: Assim como Como a luz, existe um ruído que contém todas as frequências audíveis é chamado de "ruído branco", tal como na luz, o ruído pode ser decomposto em outras partes, no PD é possível filtrar os ruídos. Para tal existem filtros como o *highpass*, que permite apenas as altas frequências passarem, enquanto as baixas frequências são suprimidas (PRINCIC, 2011). As imagens a seguir apresentam um diagrama representando o filtro *highpass* e o *lowpass* e um exemplo em PD com os mesmos.

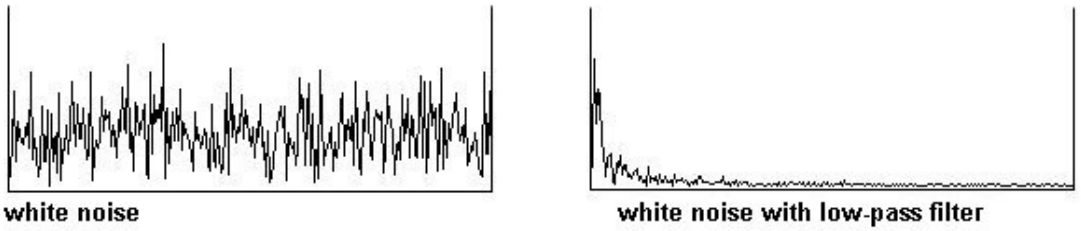


Figura 23: Ruído branco (esquerda) e ruído branco com filtro *lowpass* (esquerda) (PRINCIC et al., 2011).

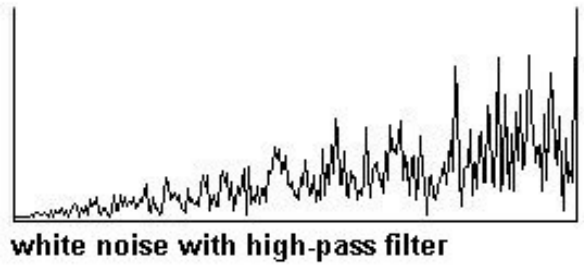


Figura 24: Ruído branco com filtro *highpass (*PRINCIC et al., 2011*)*.

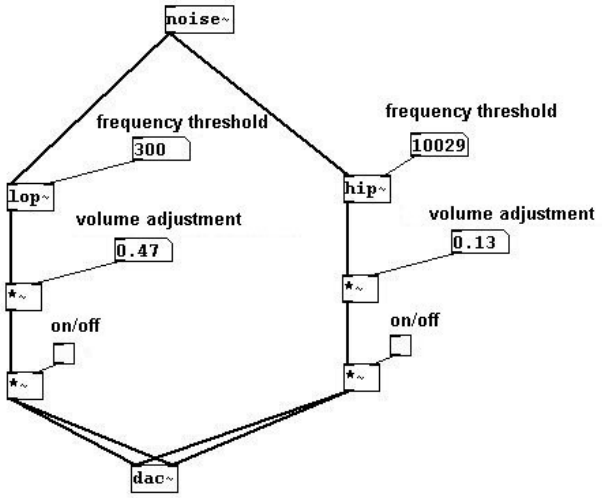


Figura 25: Filtros *lowpass* e *highpass* (PRINCIC et al., 2011).

A próxima seção se objetiva a apresentar uma experiência realizada com o uso do hardware livre (Brasuíno) e o software PD no intuito de criar um pedal para guitarra, na mesma são descritos os procedimentos utilizados para a construção do pedal e os componentes necessários para tal.

3 METODOLOGIA

Neste capítulo, serão abordadas as etapas do processo para aplicar o Brasuíno na construção do pedal para guitarra, a criação do pedal foi dividida em duas etapas principais, a primeira etapa foi a criação do hardware do pedal e a segunda, a implementação do software necessário para gerar os efeitos e controlá-los com o hardware criado.

A seção 3.1 apresenta o processo de criação do software, os objetos utilizados e a função dos mesmos. A seção 3.2 aborda o procedimento de construção da estrutura física do pedal e a seção 3.3 descreve o funcionamento e utilização do pedal.

3.1 SOFTWARE

Na etapa de pesquisa das ferramentas necessárias para desenvolvimento do projeto, o PD demonstrou-se ideal para a implementação dos efeitos para o pedal e após efetuar testes com alguns exemplos providos na documentação do PD, isto pode ser confirmado. Após a escolha do PD para criação dos efeitos, fez-se necessário uma forma de permitir que o programa criado com o PD, fosse controlado pelo Brasuíno.

Através do espaço do usuário (uma seção do site) Arduino, foi possível encontrar uma firmware que permite a comunicação do Brasuíno com diferentes softwares, o Firmata. Com a utilização do Firmata foi possível realizar a comunicação entre o Brasuíno e o PD, bastando apenas carregar o Firmata para a memória do Brasuíno através da IDE Arduino, assim foi possível também abstrair detalhes sobre a implementação de um protocolo para realizar esse tipo de comunicação, o que poderia aumentar consideravelmente a complexidade do projeto. A versão da firmware utilizada com o Brasuíno foi o StandardFirmata, esse firmware também está disponível no menu exemplos do software Arduino e no site oficial em STEINER et al. (2011). O código do StandardFirmata está disposto no

Anexo 2.

Ainda como forma de permitir a comunicação entre Brasuíno e PD, foi necessário a utilização de uma biblioteca no PD, o Pduino. O Pduino permitiu também abstrair detalhes sobre como ocorre em mais baixo nível a comunicação entre o PD e o Brasuíno.

Através do PD, foi implementado um *patch* responsável por receber os sinais de uma entrada da placa de som do computador, após receber os sinais de áudio alguns objetos foram utilizados para filtrar e modificar o som recebido e enviá-lo à saída da placa de som. O processo de construção do *patch* será descrito em detalhes a seguir e uma visão completa do mesmo está disponível no Anexo 4.

- Receber um sinal de áudio através do objeto “adc~”, o sinal foi produzido por uma guitarra conectada à entrada de microfone do computador;
- Conectar as saída do objeto “adc~” ao multiplicador principal;
- Conectar as saídas de “adc~” aos multiplicadores responsáveis por controlar os efeitos;
- Conectar o objeto “tabwrite~” à saída do multiplicador principal, para ativar o componente nomeado “display” no *patch*;
- Conectar o objeto gráfico “knob” ao objeto “line~” para que a mudança do volume ocorra gradativamente e não haja saltos grandes entre um volume e outro, depois conectar “line~” ao multiplicador principal;
- Conectar um dos pinos do objeto arduino (nesse caso foi utilizado o pino 2) ao objeto “toggle” do efeito que deseja ativar/desativar com o pedal.
- Conectar o multiplicador principal ao objeto “dac~” que enviará o áudio para a saída da placa de som;
- Adicionar os efeitos de modulação anel, síntese FM, oscilador simples e filtro highpass, e ligá-los aos multiplicadores que controlam os efeitos.

A construção do hardware do pedal é detalhado na seção seguinte, a qual apresenta em detalhes cada um dos componentes utilizados para tal.

3.2 HARDWARE

Para a construção do hardware do pedal foram utilizados alguns componentes/insumos listados a seguir:

- Brasuíno BS1 conectado via interface USB a um computadador;
- 1 resistor 10k;
- 1 pedal *footswitch*;
- 2 peças de madeira compensada 8mm, 14x9cm;
- 2 peças de madeira compensada 8mm, 2x7cm;
- 2 peças de madeira compensada 8mm, 2x14cm;
- 14 parafusos 20mm;
- 4 Suportes para placa Brasuíno.
- 4 fios de telefonia de 20cm;

A montagem foi realizada através dos seguintes passos:

- As peças de madeira foram montadas e anexadas de maneira que formassem uma caixa retangular com a tampa de cima aberta;
- Em seguida foi feito um furo em cima da tampa superior do mesmo, de modo que seja encaixado a chave Dpdt;
- Foram feitos dois furos em uma das laterais para acesso aos conectores de energia e USB e do Brasuíno;
- Os suportes da placa foram fixados na posição definida;
- Um fio foi ligado a um dos polos do conector Dpdt e conectado ao pino 5v do Brasuíno;
- Um segundo fio foi ligado ao pino GND do Brasuíno e soldado ao resistor de 10k;
- Dois fios foram ligados na outra extremidade do resistor de 10k, um deles foi ligado ao outro polo da chave Dpdt e o outro ligado no pino Digital 2 do Brasuíno.

Após finalizar as montagens, a caixa com o Brasuíno foi fechada com parafusos de forma a ficar mais apresentável e facilitar o acesso ao mesmo. Os testes do pedal foram realizados em um computador Intel Core I3 2100, 4GB RAM e HD 1TB, placa de som Realtek on-board. A guitarra utilizada foi uma Tagima T-zero e

a caixa de som, Staner Kute 60.

3.3 UTILIZANDO O PEDAL

O pedal de guitarra para guitarra não possui efeitos, o mesmo atua apenas como um controlador de um software de computador (nesse caso um *patch* PD), a função deste pedal, é conectar-se à um pedal *footswitch* e receber um sinal quando o pedal for acionado. Para a utilização do pedal é necessário conectar o mesmo a uma porta USB do computador, abrir o *patch* criado em PD, e conectar ao pedal, um outro pedal (*footswitch*), depois de ligar o dedal, basta conectar a guitarra à uma entrada do dispositivo de áudio do computador e aumentar o botão titulado volume no patch PD, depois de realizar estes procedimentos já é possível utilizar o pedal. As análises e resultados dos testes são apresentados na próxima seção.

4 ANÁLISE E RESULTADOS

O processo de construção do pedal teve a sua complexidade avaliada de acordo com algumas variáveis, essas variáveis serviram como base para uma análise de viabilidade na criação deste projeto e também podem servir de modelo para análise de viabilidade da utilização do Brasuíno em diferentes projetos. As principais variáveis a serem analisadas são: tempo de desenvolvimento do projeto, custo para o desenvolvimento e complexidade do código.

O tempo total gasto para implementação do projeto foi de aproximadamente 7 dias. Ao analisar de forma mais detalhada o processo de criação é possível dividi-lo em algumas fases, conforme apresentado na Tabela 2. Em seguida são apresentados na Tabela 3 os custos de componentes utilizados no projeto.

Tabela 2: Tempo necessário para implementação do pedal com Brasuíno BS1 classificado por etapas

Etapa	**Tempo gasto**
Pesquisa das ferramentas (hardware e software) necessárias	8 horas
Estudo da linguagem e ferramenta de desenvolvimento PD	4 dias
Estudo da linguagem e software Arduino	1 dia
Implementação do *patch* PD	2 dia

Tabela 3: Custo de componentes

Item	**Valor**
Placa Brasuíno BS1	R$ 77,00
1 resistor 10k	R$ 0
1 pedal *footswitch*	R$ 0
Peças de madeira (todas)	R$ 0
12 pregos 20mm	R$ 0,50
2 parafusos 20mm	R$ 0
4 Suportes para placa Brasuíno	R$ 0
4 fios de telefonia de 20cm	R$ 0

Conforme apresentado na Tabela 3, apenas três itens possuem seus valores declarados, isso ocorre porque o restante do material é reaproveitado, alguns desses materiais foram adquiridos de sucatas de eletrônicos ou lixo eletrônico, o que tornou mais fácil a realização de experiências com o Brasuíno, já que não foi necessário arcar com custos de todos os componentes.

Um outro ponto importante a se observar sobre a viabilidade do Brasuíno se comparado a outros modelos de hardware livre, é o custo da placa. A Tabela 4 realiza a comparação do custo do Brasuíno em relação a outros modelos comercializados no Brasil.

Tabela 4: Comparação de valores do Brasuíno e outros modelos de Arduino.

Item	**Valor**
Brasuíno BS1	R$ 90,00
Seeduino	R$ 99,90
Arduino Uno (original)	R$ 133,00
Arduino Uno (original)	R$ 136,00

A seguir são apresentadas algumas figuras do pedal para guitarra em diferentes ângulos proporcionando uma visão de como o mesmo foi montado.

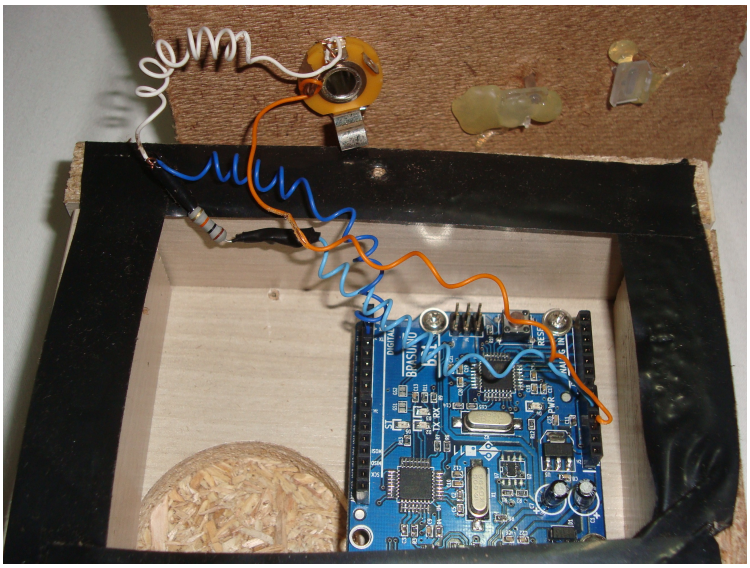


Figura 26: Visão superior do pedal aberto.

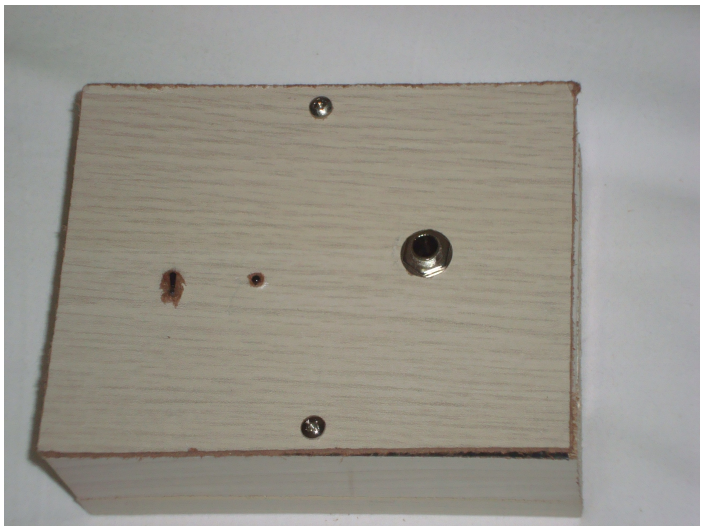


Figura 27: Visão superior do pedal fechado.

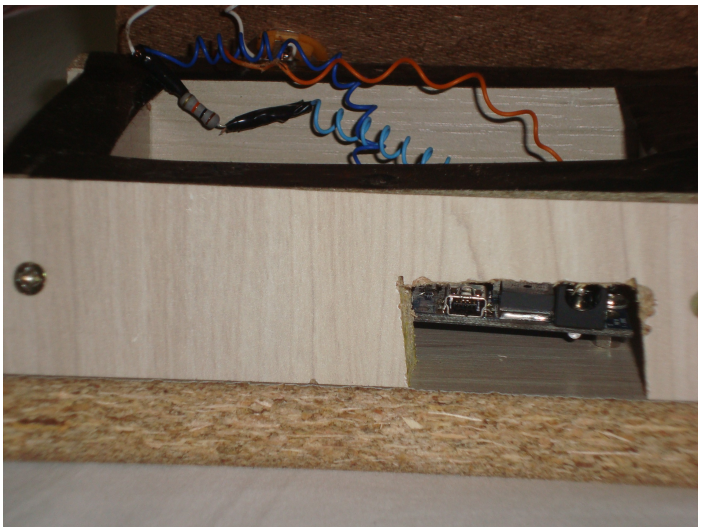


Figura 28: Visão frontal do pedal aberto.

Figura 29: Visão frontal do pedal fechado.

5 CONCLUSÃO

O entendimento do hardware é fundamental para a compreensão de algumas disciplinas de cursos como Engenharia Elétrica e Ciência da Computação, sendo alguns componentes tais como processadores, memórias e circuitos, utilizados comumente nas atividades de acadêmicos dessa área, a utilização do hardware livre, permite um melhor entendimento do hardware em suas diferentes formas, o que contribui também para o melhor envolvimento das pessoas com os níveis práticos das disciplinas que utilizam o hardware, como é o caso da disciplina de Arquitetura de Computadores.

Muitos componentes já possuem versões livres, interfaces de rede, vídeo e som são exemplos desses. A possibilidade de entender o que se passa por dentro desses componentes e saber como e do que são feitos, permite estudá-los de forma mais eficiente. O Brasuíno, como um modelo de hardware livre, possui à disposição dos usuários os esquemáticos e informações necessárias para que seja estudado e modificado por aqueles que se interessarem em compreender melhor o seu funcionamento.

A utilização do Brasuíno BS1 na realização do projeto apresentou resultados bastante satisfatórios, sendo possível integrá-lo ao software PD e realizar as funções desejadas, de ativar e desativar efeitos do *patch*. O Brasuíno pode também ser utilizado em uma grande diversidade de projetos que envolvem diretamente a integração do software ao hardware, fato este que pode agregar valor a experimentos realizados em disciplinas do curso de Ciência da Computação, entre estes a criação de robôs que interajam com o meio físico.

Para a criação do pedal com o Brasuíno, foram utilizados outros componentes, alguns desses componentes foram adquiridos do que viria a ser lixo eletrônico, os usuários do hardware livre geralmente são favoráveis à ideia de reaproveitamento desse tipo de material. Ainda que os projetos com hardware livre não sejam uma solução para o problema do lixo eletrônico, a conscientização de que parte desse material descartado pode ser útil de forma significativa é bem aceita.

O fato de utilizar ferramentas livres no desenvolvimento deste trabalho, contribuiu consideravelmente para a simplificação de algumas etapas na realização

do mesmo, isso pode ser observado através do uso da firmware Firmata e do *patch* Pduino, que podem ser citados como exemplo de aproveitamento de código, o que é um dos fatores que torna o uso de ferramentas livres ainda mais vantajoso.

Um outro ponto importante deste trabalho foi o estudo e utilização do software PD. Através das pesquisas realizadas em busca de uma ferramenta para a criação dos efeitos de áudio, o PD foi apontado como um ambiente robusto para a programação de áudio, vídeo e processamento gráfico. O estudo deste software permitiu o conhecimento e entendimento de conceitos interessantes como o de síntese de som e frequência.

Através dos estudos e apresentação do conteúdo de referencial teórico também foi possível reunir uma série de informações sobre hardware livre, Arduino e Brasuíno, o que pode possibilitou criar um material de apoio no idioma português, parte desse material é destinado à documentação do Brasuíno BS1, permitindo aos interessados em hardware livre, ter uma facilitação em sua experiência inicial com o Brasuíno.

Como sugestão para realização de trabalhos futuros, podem ser realizados diversos experimentos que são publicados por usuários do Arduino em seu site oficial, é possível ainda realizar diferentes melhorias no projeto do pedal criado neste trabalho, algumas sugestões são: aumentar o número de efeitos, aumentar o número de chaves para ativar/desativar os efeitos, inserir LED's e um display no pedal e inserir um potenciômetro para o controle do volume do pedal.

REFERÊNCIAS

ARDUINO. *Arduino – HomePage*. Disponível em: <http://arduino.cc/>. Acessado em: 15 Setembro 2011a.

ARDUINO. *Arduino - Hardware*. Disponível em: <http://arduino.cc/en/Main/Hardware>. Acessado em: 15 Setembro 2011b.

Arduino. *Arduino – ArduinoBoardUno*. Especificações técnicas. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardUno>. Acessado em: 17 Setembro 2011c.

ARDUINO. *Arduino – ArduinoBoardEthernet*. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardEthernet>. Acessado em: 17 Setembro 2011d.

ARDUINO. *Arduino – ArduinoBoardBluetooth*.Disponível em: <http://arduino.cc/en/Main/ArduinoBoardBluetooth>. Acessado em: 15 Setembro 2011e.

ARDUINO. *Arduino – Boards*. Disponível em: <http://arduino.cc/en/Main/Boards>. Acessado em: 23 Setembro 2011f.

ARDUINO. *Arduino – ArduinoBoardLilyPad*. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardLilyPad>. Acessado em: 22 Setembro 2011g.

ARDUINO. *Arduino – ArduinoShields*. Disponível em: <http://arduino.cc/en/Main/ArduinoShields>. Acessado em: 24 Setembro 2011h.

ARDUINO. *Arduino Playground - SimilarBoards*. Disponível em: <http://www.arduino.cc/playground/Main/SimilarBoards#goShie>. Acessado em: 15 Setembro 2011i.

ARDUINO. *Arduino Playground: ArduinoUsers.* Disponível em: <http://www.arduino.cc/playground/Projects/ArduinoUsers>. Acessado em: 17 Setembro 2011j.

ARDUINO. *Arduino – Policy*. Disponível em: <http://www.arduino.cc/en/Main/Policy>. Acessado em: 17 Setembro 2011k.

ARDUINO. *Arduino – Reference*. Disponível em: <http://arduino.cc/en/Reference/HomePage>. Acessado em: 15 Setembro 2011l.

ARDUINO. *Arduino - Software.* Disponível em: <http://arduino.cc/en/Main/Software>. Acessado em: 15 Setembro 2011m.

ARDUINO. *Arduino – Buy*. Disponível em: <http://arduino.cc/en/Main/Buy>. Acessado em: 25/11/2011n.

BANZI, Massimo. *Getting started with Arduino*, 1st Ed., Sebastopol, U.S.A: Brian Jepson, 2008, 128p.

BEIRIZ, Fernando Antonio Santos. G*estão ecológica de resíduos eletrônicos – proposta de modelo conceitual de gestão* . Dissertação apresentada ao Curso de Mestrado em Sistemas de Gestão da Universidade Federal Fluminense. Niterói, 2005.

CALVO, Rodrigo e ALAEJOS, Raúl. *Arduino The Documentary*. Laboral Centro de Arte y Creación Industrial. Disponível em: <http://arduinothedocumentary.org/>. Acessado em: 02 Novembro 2011.

COWBOY, Deck. *Hardware aberto?.* CMI – Centro de mídia intependente, 18 Dezembro 2003. Disponível em: <http://www.midiaindependente.org/pt/blue/2003/12/270057.shtml>. Acessado em: 15 Setembro 2011.

CREATIVE COMMONS. *Creative Commons - Attribution-ShareAlike 2.5 Generic* . Disponível em: <http://creativecommons.org/licenses/by-sa/2.5/deed.pt_BR>. Acessado em: 15 Setembro 2011.

CHADE, Jamil. *Brasil é o campeão do lixo eletrônico entre emergentes*. O Estado de S. Paulo 22 de fevereiro de 2010. Disponível em: <http://www.estadao.com.br/noticias/vidae,brasil-e-o-campeao-do-lixo-eletronico-entre-emergentes,514495,0.htm>. Acessado em: 26 Setembro 2011.

ELETTRONICA IN**.** *Arduino GSM shield* | *Open Electronics.* Disponível em: <http://www.open-electronics.org/arduino-gsm-shield/>. Acessado em: 24 Setembro 2011.

FIRMATA. *Main Page – Firmata*. Disponível em: <http://firmata.org/>. Acessado em: 01 Dezembro 2011..

FREEDUINO. *The World Famous Index of Arduino & Freeduino Knowledge.* Disponível em: <http://www.freeduino.org/>. Acessado em: 23 Setembro 2011.

GNU Project. O que é o Software Livre?. Disponível em <http://www.gnu.org/philosophy/free-sw>. Acessado em: 15 Setembro 2011.

HEIM, Anna. *Meet the Chilean Teen Who Warns of Earthquakes on Twitter.* TNW Latin America, 18 Julho 2011. Disponível em: <http://thenextweb.com/la/2011/07/18/meet-the-chilean-teen-who-warns-of-earthquakes-on-twitter/>. Acessado em: 15 Setembro 2011.

HOLOSCÓPIO. *Brasuíno*. Disponível em: <http://brasuino.holoscopio.com/>.

Acessado em: 15 Setembro 2011a.

HOLOSCÓPIO. *Brasuíno BS1 - Plataforma de Prototipagem* | *Holoscópio*. Disponível em: <http://loja.holoscopio.com/produtos/brasuinobs1>. Acessado em: 15 Setembro 2011b.

KREIDLER, Johannes. *Programming Electronic Music in Pd.* Disponível em: <http://www.pd-tutorial.com/english/index.html>. Acessado em: 01 Dezembro 2011.

KULHAVÝ, Karel 'Clock' et al. *About Ronja*. Disponível em: <http://ronja.twibright.com/about.php>. Acessado em: 15 Setembro 2011.

LIQUIDWARE. *Liquidware : Open Source Electronics.* Disponível em: <http://www.liquidware.com/shop/show/TSL/TouchShield+Slide>. Acessado em: 24 Novembro 2011.

MARGOLIS, Michael. *Arduino Cookbook*, 1st ed. O'Reilly Media, Incorporated, 2011.

MÖLLER, Erik et al. *OSHW/translations/pt - Definition of Free Cultural Works.* Disponível em: <http://freedomdefined.org/OSHW/translations/portuguese>. Acessado em: 51 Setembro 2011a.

MÖLLER, Erik et al. *Definition/Pt - Definition of Free Cultural.* Disponível em: <http://freedomdefined.org/Definition/Pt>. Acessado em: 18 Setembro 2011b.

NOBLE, Joshua. *Programming Interactivity.* 1st ed. O'Reilly Media, 2009.

OPENCORES. *Open Cores Projects.* Disponível em: <http://opencores.org/projects>. Acessado em: 15 Setembro 2011.

OPENENERGYMONITOR. *Home* | *OpenEnergyMonitor*. Disponível em: <http://openenergymonitor.org/emon/>. Acessado em: 15 Setembro 2011.

OSIER-MIXON, Jeffrey. *Hardware Aberto: Como e Quando Funciona.* IBM Corporation 2010 – Developer Works, 6 Dezembro 2010. Disponível em: <http://www.ibm.com/developerworks/br/library/os-openhardware/>. Acessado em: 15 Setembro 2011.

OXER, Jonathan. *Arduino Shiel List.* Disponível em: <http://shieldlist.org/>. Acessado em: 24 Novembro 2011.

PD. *Pure Data – PD Community Site*. Disponível em: <http://puredata.info/>. Acessado em: 25 Novembro 2011.

PORRES, Alexandre Torres. Apostila do Curso de Computação Musical por Alexandre Torres Porres (2009). Disponível em: <http://sites.google.com/site/porres/pd>. Acessado em: 25 Novembro 2011.

PRINCIC, Luka et al. *Pure Data.* Disponível em: <http://en.flossmanuals.net/pure-data/>. Acessado em: 25 Novembro 2011.

REPRAP. *RepRap – RepRapWiki*. Disponível em: <http://reprap.org/wiki/Main_Page>. Acessado em: 01 Dezembro 2011.

ROGUE ROBOTICS. *rMP3 - Playback Module | Rogue Robotics.* Disponível em: <http://www.roguerobotics.com/products/electronics/rmp3>. Acessado em: 24 Novembro 2011.

SILVA, Marisa. *Tecnologia na Balada*. Olhar Digital, 02 de Novembro de 2008. Disponível em: <http://olhardigital.uol.com.br/produtos/central_de_videos/tecnologia-na-balada>. Acessado em: 15 Setembro 2011.

STEINER, Hans-Christoph et al. *Pd Objectclasses*. Disponível em: <http://at.or.at/hans/pd/objects.html#pduino>. Acessado em: 01 Dezembro 2011.

UZEBOX. *Main Page – Uzebox*. Disponível em: <http://uzebox.org/wiki/index.php?title=Main_Page>. Acessado em: 01 Dezembro 2011.

ZLATAR, Mateo. *The open source hardware logo repository.* Disponível em: <http://oshwlogo.com/>. Acessado em: 02 Novembro 2011.

ANEXO 1

INSTALAÇÃO DA IDE ARDUINO

Instalação em sistemas Debian GNU/Linux:

Abra um terminal e digite as seguintes linhas de comando:

```
$ sudo apt-get install arduino-core
$ sudo apt-get install arduino
```

Os passos citados descrevem a instalação da IDE Arduino completa, para a instalação da mesma é necessária possuir algumas bibliotecas Java, caso o usuário não queira instalar as dependências, basta instalar somente o pacote “arduino-core” utilizando somente os comandos da primeira linha acima mencionado, nesse caso o usuário poderá compilar e instalar os *sketches* via linha de comando.

Depois de instalado basta digitar “arduino” em uma linha de comando e começar a programar.

O código fonte do software Arduino está disponível em: https://github.com/arduino/Arduino

ANEXO 2

CÓDIGO NO BRASUÍNO: FIRMATA

```
#include <Firmata.h>
#include <Servo.h>
/*============================
 * VARIÁVEIS GLOBAIS
 *============================*/

/* Entradas analógicas */
int analogInputsToReport = 0; // bitwise vetor para armazenar o pino de reporte
int analogPin = 0; // contador para leitura dos pinos analógicos

/* Pinos Digitais */
byte reportPINs[TOTAL_PORTS];   // PIN == entrada
byte previousPINs[TOTAL_PORTS]; // PIN == entrada
byte pinStatus[TOTAL_DIGITAL_PINS]; // gravar pino status, saída padrão
byte portStatus[TOTAL_PORTS];

/* timer variables */
unsigned long currentMillis;     // gravar o valor atual de millis()
unsigned long nextExecuteMillis; // para comparação com currentMillis
int samplingInterval = 19;      // quantas vezes executar o loop principal (in ms)

Servo servos[MAX_SERVOS];

/*============================
 * FUNÇÕES
*============================*/

void outputPort(byte portNumber, byte portValue)
{
  portValue = portValue &~ portStatus[portNumber];
  if(previousPINs[portNumber] != portValue) {
    Firmata.sendDigitalPort(portNumber, portValue);
    previousPINs[portNumber] = portValue;
  }
}

/* -----------------------------------------------------------------------------
 * Checar todas as entradas digitais ativas para mudança de estado, depois adicione
os efentos
 * para a fila de saída serial usando Serial.print() */
void checkDigitalInputs(void)
{
  byte i, tmp;
  for(i=0; i < TOTAL_PORTS; i++) {
```

```
    if(reportPINs[i]) {
      switch(i) {
      case 0:
        outputPort(0, PIND &~ B00000011); // ignorar Rx/Tx 0/1
        break;
      case 1:
        outputPort(1, PINB);
        break;
      case ANALOG_PORT:
        outputPort(ANALOG_PORT, PINC);
        break;
      }
    }
  }
}

// ----------------------------------------------------------------------------
/*define o modo de pin para o estado correto e define os bits relevantes no
* vetor de dois bits que rastreia o status de entrada/saída e PWM Digital
 */
void setPinModeCallback(byte pin, int mode) {
  byte port = 0;
  byte offset = 0;

  // OBS: desconsidere em placas diferentes
  if (pin < 8) {
    port = 0;
    offset = 0;
  } else if (pin < 14) {
    port = 1;
    offset = 8;
  } else if (pin < 22) {
    port = 2;
    offset = 14;
  }

  if(pin > 1) { // ignore RxTx (pins 0 and 1)
    if (isServoSupportedPin(pin) && mode != SERVO)
      if (servos[pin - FIRST_SERVO_PIN].attached())
        servos[pin - FIRST_SERVO_PIN].detach();
    if(pin > 13)
      reportAnalogCallback(pin - 14, mode == ANALOG ? 1 : 0); // turn on/off reporting
    switch(mode) {
    case ANALOG:
      digitalWrite(pin, LOW); // disable internal pull-ups and fall thru to 'case INPUT:'
    case INPUT:
      pinStatus[pin] = mode;
      pinMode(pin, INPUT);
      portStatus[port] = portStatus[port] &~ (1 << (pin - offset));
      break;
```

```
    case OUTPUT:
      digitalWrite(pin, LOW); // desabilida PWM e cai em verdadeiro p/ 'case PWM:'
    case PWM:
      pinStatus[pin] = mode;
      pinMode(pin, OUTPUT);
      portStatus[port] = portStatus[port] | (1 << (pin - offset));
      break;
    case SERVO:
      // OBS: suporta Arduino Mega
      if (isServoSupportedPin(pin)) {
        pinStatus[pin] = mode;
        if (!servos[pin - FIRST_SERVO_PIN].attached())
          servos[pin - FIRST_SERVO_PIN].attach(pin);
      } else
        Firmata.sendString("Servo only on pins from 2 to 13");
      break;
    case I2C:
      pinStatus[pin] = mode;
      Firmata.sendString("I2C mode not yet supported");
      break;
    default:
          Firmata.sendString("Unknown pin mode"); // OBS: colocar msg de erro em
EEPROM
    }
    // Obs.: salvar status EEPROM  aqui, se for alterado
  }
}

void analogWriteCallback(byte pin, int value)
{
  switch(pinStatus[pin]) {
  case SERVO:
    if (isServoSupportedPin(pin))
      servos[pin - FIRST_SERVO_PIN].write(value);
    break;
  case PWM:
    analogWrite(pin, value);
    break;
  }
}

void digitalWriteCallback(byte port, int value)
{
  switch(port) {
  case 0: // pins 2-7 (don't change Rx/Tx, pins 0 and 1)
    // 0xFF03 == B1111111100000011    0x03 == B00000011
    PORTD = (value &~ 0xFF03) | (PORTD & 0x03);
    break;
  case 1: // pinos de 8 a 13 (14,15 estão desabilitados para o cristal)
    PORTB = (byte)value;
```

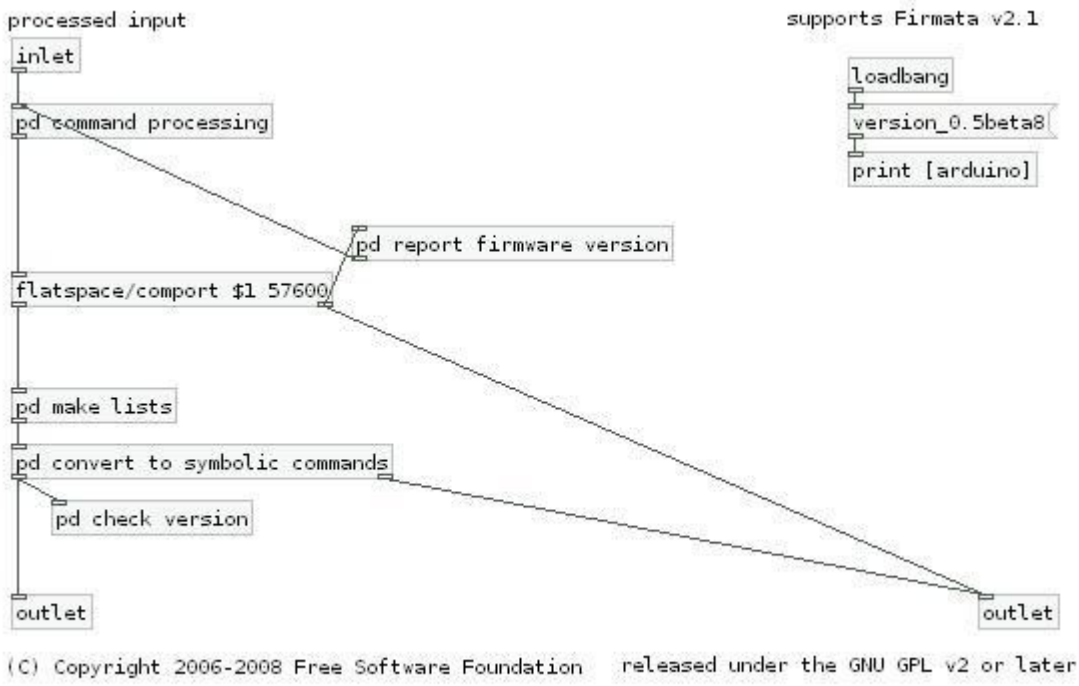
```
    break;
  case 2: // pinos analógicos usados com digitais
    byte pin;
    byte pinModeMask;
    for(pin=0; pin<8; pin++)
      if(pinStatus[pin] == OUTPUT)
        pinModeMask += 1 << pin;
    PORTC = (byte)value & pinModeMask;
    break;
  }
}

// -----------------------------------------------------------------------------
/* seta bits em um vetor de bits (int) para alternar o relatório dos pinos analógicos*/
//void FirmataClass::setAnalogPinReporting(byte pin, byte state) {
//}
void reportAnalogCallback(byte pin, int value)
{
  if(value == 0) {
    analogInputsToReport = analogInputsToReport &~ (1 << pin);
  }
  else { // tudo, exceto 0, permite comunicação desse pino
    analogInputsToReport = analogInputsToReport | (1 << pin);
  }
  // TODO: salvar status EEPROM  aqui, se for alterado
}

void reportDigitalCallback(byte port, int value)
{
  reportPINs[port] = (byte)value;
   if(port == ANALOG_PORT) // desativar o relatório de analógico quando usar o
digital
    analogInputsToReport = 0;
}

/*==============================
 * SYSEX-BASED commands
 *==============================*/

void sysexCallback(byte command, byte argc, byte *argv)
{
  switch(command) {
  case SERVO_CONFIG:
    if(argc > 4) {
          // essas vars estão aqui para maior clareza, eles serão otimizadas pelo
compilador
      byte pin = argv[0];
      int minPulse = argv[1] + (argv[2] << 7);
      int maxPulse = argv[3] + (argv[4] << 7);
```

```
      if (isServoSupportedPin(pin)) {
        // servos are pins from 2 to 13, so offset for array
        if (servos[pin - FIRST_SERVO_PIN].attached())
          servos[pin - FIRST_SERVO_PIN].detach();
        servos[pin - FIRST_SERVO_PIN].attach(pin, minPulse, maxPulse);
        setPinModeCallback(pin, SERVO);
      }
    }
    break;
  case SAMPLING_INTERVAL:
    if (argc > 1)
      samplingInterval = argv[0] + (argv[1] << 7);
    else
      Firmata.sendString("Not enough data");
    break;
  }
}

boolean isServoSupportedPin(byte pin)
{
    return ((FIRST_SERVO_PIN <= pin) && (pin <= (FIRST_SERVO_PIN +
MAX_SERVOS)));
}

/*=============
 * SETUP()
 *=============*/
void setup()
{
  byte i;

  Firmata.setFirmwareVersion(2, 1);

  Firmata.attach(ANALOG_MESSAGE, analogWriteCallback);
  Firmata.attach(DIGITAL_MESSAGE, digitalWriteCallback);
  Firmata.attach(REPORT_ANALOG, reportAnalogCallback);
  Firmata.attach(REPORT_DIGITAL, reportDigitalCallback);
  Firmata.attach(SET_PIN_MODE, setPinModeCallback);
  Firmata.attach(START_SYSEX, sysexCallback);

  portStatus[0] = B00000011;  // ignore Tx/RX pins
  portStatus[1] = B11000000;  // ignore 14/15 pins
  portStatus[2] = B00000000;

  for(i=0; i < FIRST_ANALOG_PIN; ++i) {
    setPinModeCallback(i,OUTPUT);
  }

//setar todas as saídas para 0 para se certificar de resistores internos estão
desligados
```

```
  PORTB = 0; // pins 8-15
  PORTC = 0; // analog port
  PORTD = 0; // pins 0-7

// TODO rethink the init, perhaps it should report analog on default

  for(i=0; i<TOTAL_PORTS; ++i) {
    reportPINs[i] = false;
  }

// TODO: salvar estado da EEPROM aqui

/*enviar entradas digitais aqui, se habilitado, para definir o estado inicial no
*Computador host, já que uma vez no loop (), este firmware só enviará
*Dados digitais sobre a mudança. * /

  if(reportPINs[0]) outputPort(0, PIND &~ B00000011); // ignore Rx/Tx 0/1
  if(reportPINs[1]) outputPort(1, PINB);
  if(reportPINs[ANALOG_PORT]) outputPort(ANALOG_PORT, PINC);

  Firmata.begin(57600);
}

/*==================
 * LOOP()
 *=================*/
void loop()
{
  /* LEITURA DIGITAL- o mais rápido possível, verificar as alterações e enviá-las * /
  checkDigitalInputs();
  currentMillis = millis();
  if(currentMillis > nextExecuteMillis) {
    nextExecuteMillis = currentMillis + samplingInterval;
     /* SERIALREAD - Serial.read() Usa um buffer de 128 bytes circular, de modo a
lidar com todos leituras seriais de uma só vez, ou seja esvaziar o buffer */
    while(Firmata.available())
      Firmata.processInput();
/* Manda FTDI escrever no bugger - certifique-se que o buffer FTDI não exceda
* 60 bytes. Idealmente, isso poderia enviar um "event character cada ms 4 a
* Acionar o buffer para despejo. * /

/* Ler Analógico - fazer todas as analogReads () uma vez por ciclo de pesquisa*/

    for(analogPin=0;analogPin<TOTAL_ANALOG_PINS;analogPin++) {
      if( analogInputsToReport & (1 << analogPin) ) {
        Firmata.sendAnalog(analogPin, analogRead(analogPin));
      }
    }
  }
```

ANEXO 3

PATCH PDUINO



ANEXO 4

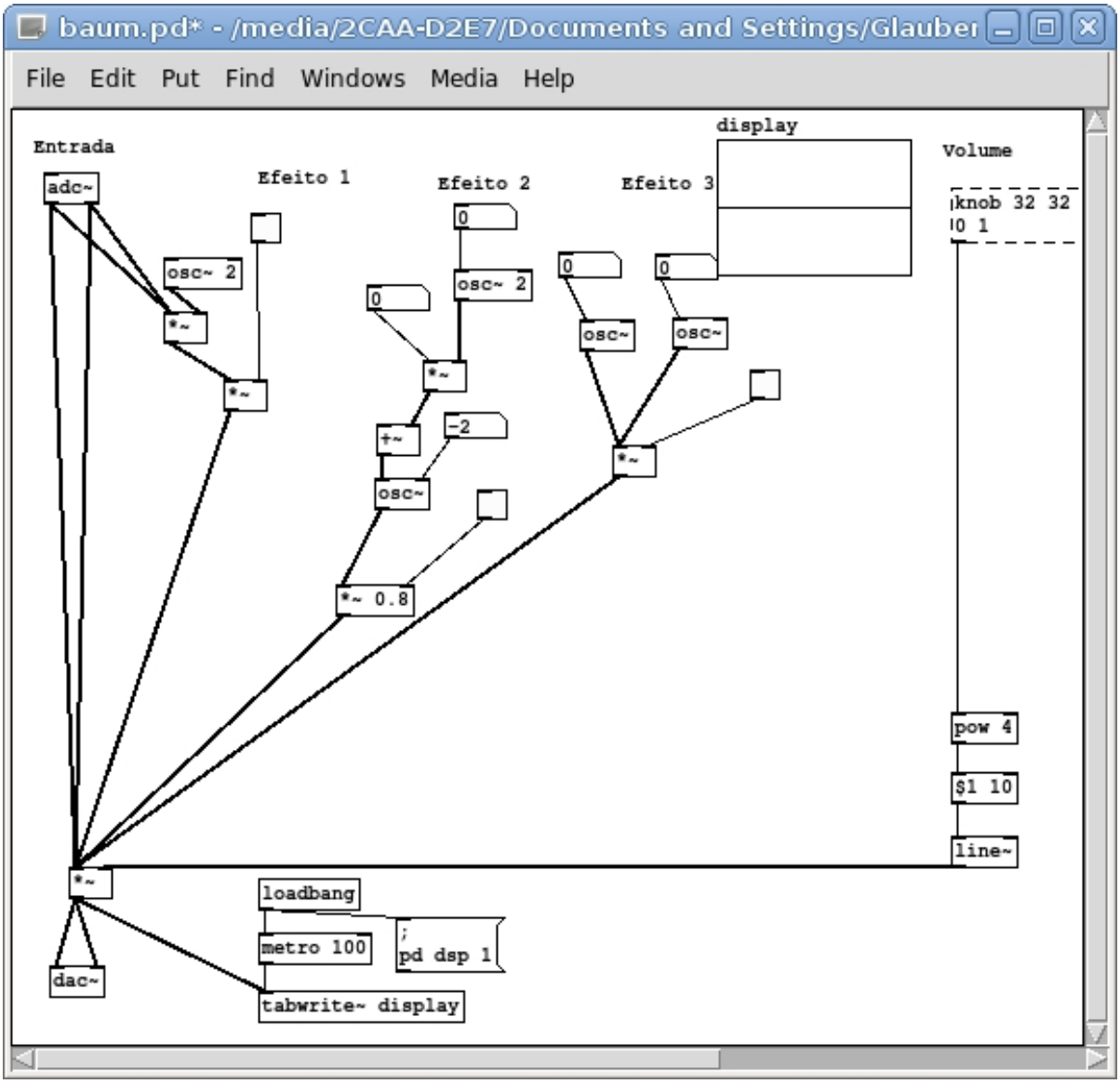


Figura 30: Patch com programa final.