UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

Sistema para identificação de Áudio utilizando Linux embarcado em microcontrolador ARM

Marcos Machado de Abreu

Orientador: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2012

Marcos Machado de Abreu

Sistema para identificação de Áudio utilizando Linux embarcado em microcontrolador ARM

Trabalho de Conclusão de Curso
Apresentado à Escola de Engenharia de São Carlos
Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Evandro Luis Linhari Rodrigues

São Carlos
2012

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

Machado de Abreu, Marcos

M321s Sistema para identificação de Áudio utilizando Linux embarcado em microcontrolador ARM / Marcos Machado de Abreu; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2012.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012.

1. Linux embarcado. 2. ARM. 3. Processamento de áudio. I. Título.

FOLHA DE APROVAÇÃO

Nome: Marcos Machado de Abreu

Título: "Sistema para identificação de Áudio utilizando Linux embarcado em microcontrolador ARM"

Trabalho de Conclusão de Curso defendido e aprovado em 27/11/2012,

com NOTA 9,0 (Nove, zero), pela Comissão Julgadora:

Prof. Associado Evandro Luís Linhari Rodrigues (Orientador)
SEL/EESC/USP

Profa. Dra. Maria Stela Veludo de Paiva
SEL/EESC/USP

Prof. Associado Ivan Nunes da Silva
SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

Dedicatória

Dedico este trabalho aos meus familiares, especialmente meus pais, Nilda e Marcos que sempre me apoiaram e não mediram esforços para manter meus estudos em São Carlos e, além disso, pela compreensão e suporte nas barreiras e desafios superados no trajeto desta conquista.

Aos meus companheiros de sala que transformaram estes 5 anos em São Carlos uns dos anos mais felizes e inesquecíveis de minha vida. Dentre estes amigos, uma dedicatória especial para aqueles, os quais que além da sala de aula convivi e aprendi além de tudo, Ivan, Marcelo, Nayra e Rafael.

Agradecimentos

Agradeço primeiramente à Deus, por todas as oportunidades e portas que foram abertas para mim em toda minha vida.

Agradeço aos meus pais, juntamente com minha irmã, avos e minha namorada, pelo apoio nos momentos de indecisão e de superação de desafios.

Agradeço ao meu orientador Evandro, que providenciou ideias e suporte para a realização do trabalho, e também aos alunos Patrícia e Aliel que gentilmente cederam uma excelente base para início do projeto. A todos um sincero obrigado.

Sumário

1. Introdução 25
   1.1. Objetivo 28
   1.2. Estrutura da Monografia 29
2. Conceitos e Descrição técnica 31
   2.1. Sistemas Embarcados 31
   2.2. Linux 32
   - *Rootfilesystem* *32*
   - *Kernel* *32*

   2.3. Kit *Tiny* 6410 *Friendly Arm* 33
3. Descrição do projeto 35
   3.1. Website 36
   3.2. Banco de dados 37
   3.3. Algoritmos de Codificação 39
   - *Áudio Fingerprint* *39*
   - *Echoprint* *41*
   - *Chromaprint* *44*

4. Implementação e Desenvolvimento 51
   4.1. Compilação dos algoritmos 51
   - *Echoprint* *51*
   - *Chromaprint* *54*

   4.2. Instalação e configuração do banco de dados 56
   4.3. Desenvolvimento do *website* 61
   - *Sistema de Cadastro de usuário* *61*
   - *Campo de login* *63*
   - *Alterar senha* *63*
   - *Administrador* *64*
   - *Dados* *65*
   - *Submeter* *65*
   - *Áudio FP* *68*

   4.4. Configurações adicionais 69
5. Resultados e análises 71

6. Considerações Finais..... 77
Referências Bibliográficas..... 78
APÊNDICE A – Imagens do *website*..... 81

Lista de Tabelas

Tabela 1: Comparação entre os algoritmos 48
Tabela 2: Tempo de resposta de cada algoritmo 71
Tabela 3: Tempo de resposta do servidor de cada algoritmo 72
Tabela 4: Tempo de resposta do *site* 73
Tabela 5: Tempo de resposta da submissão de dados 75

Lista de Gráfico

Gráfico 1: Tempo de execução dos algoritmos x músicas 71
Gráfico 2: Tempo de resposta do servidor x músicas 72
Gráfico 3: Tempo de resposta do sistema x Músicas (*echoprint*)..... 74
Gráfico 4: Tempo de resposta do sistema x Músicas (*chromaprint*)..... 74

Lista de Figuras

Figura 1: Kit Olimex SAM9 L926, retirado do Manual OLIMEX. 26
Figura 2: Exemplo de atuação do *Kernel* em uma máquina. 33
Figura 3: Plana Tiny 6410, retirado do MANUAL FriendlyArm Tiny6410 34
Figura 4: Diagrama das funções do site 35
Figura 5: Layout da página principal 36
Figura 6: Visualização do banco de dados pelo phpMyAdmin 38
Figura 7: Detecção de onsets, BELLO et al (2005) 42
Figura 8: Sinal nomal e sinal gravado OTA, simulados no Audacity 43
Figura 9: Fluxograma do funcionamento do Echoprint 44
Figura 10: Espectrograma, gerado com base no MANUAL Audacity 45
Figura 11: : Transformações e resultado - Chroma features BARTSCH (2005). 46
Figura 12: Janelas do Filtro 47
Figura 13: Exemplo chromaprint, retirado da BASE DE DADOS do chromaprint 49
Figura 14: Fluxograma do funcionamento do Chromaprint 50
Figura 15: Janela para inserir a senha do mysql 57
Figura 16: Tela de instalação - Configuração do sistema 58
Figura 17: Tela para inserir senha do Mysql 59
Figura 18: Tela de início phpMyAdmin 59
Figura 19: Execução dos comandos na janela de comando 61
Figura 20: Informações necessárias para o cadastro 62
Figura 21: Fluxograma Lógica - do cadastro 62
Figura 22: Questionário para enviar dados da música 66
Figura 23: Fluxograma do funcionamento da página submeter 67
Figura 24: Layout da página Áudio FP 68
Figura 25: Fluxograma - Página Áudio FP 69
Figura 26: Banco de dados como as músicas enviadas para teste 76

Lista de Símbolos

ARM	*Advanced Risc Machine*
FP	*Finger Print*
ID	Impressão Digital
OTA	*Over the air*
Npk	*New peak signal*
HTTP	*HyperText Transfer Protocol Secure*
SQL	*Structured Query Language*
FLAC	*Free Lossless Audio Codec*

Resumo

Esta monografia visa abordar uma subárea de Sistemas Digitais com franca possibilidade de crescimento e grande potencial tecnológico que é o trabalho com sistemas embarcados, Linux embarcado, utilizando microprocessador de arquitetura *ARM* explorando o ambiente *Web.* O *hardware* e o sistema operacional empregados foram usados para análise e codificação de áudio (músicas em formato mp3) que, posteriormente, baseado em banco de dados, permitem a identificação do áudio em questão. Foram utilizados dois algoritmos para a codificação do áudio para a realização de comparações e discussões a respeito de suas metodologias e desempenhos. Depois de realizados os ensaios sobre tempo de execução e eficiência de resposta, foi possível observar a superioridade em termos de processamento obtida pelo *Chromaprint* sobre o *Echoprint*. Este projeto disponibiliza uma contribuição em forma de uma plataforma que se utiliza dos recursos que o ambiente *Web* oferece, para aplicação de metodologias de reconhecimento de áudio, sem restrição de acesso para usuários.

Palavras-chave: *Linux embarcado*, ARM, Processamento de áudio.

Abstract

This monograph aims to address a subarea of Digital Systems with clear possibility of technological growth and great potential for different applications that works with embedded systems, embedded Linux, using ARM microprocessor architecture exploiting the Web environment The hardware and operating system employed were used to analysis and encoding audio (music in mp3 format) that subsequently based on database, allowed the identification of the audio in question. We used two algorithms for audio encoding to perform comparisons and discussions about their methodologies and performance. After conducted the tests on runtime efficiency and response it was observed the superiority in terms of processing obtained by Chromaprint over the Echoprint algorithm. This project provides a contribution in terms of a platform that uses the Web environment resources for application of methodologies for audio recognition, without restricting user access.

Keywords: Embedded Linux, ARM architecture, audio processing.

1. Introdução

A evolução dos computadores é o que guia o crescimento das tecnologias existentes no mundo, desde o surgimento do ENIAC e dos computadores feitos com válvulas na década de 1950, até os dias atuais com os supercomputadores, automação industrial e comercial. O avanço tecnológico é o que permite a evolução das outras áreas, como a medicina, biologia, etc, oferecendo equipamentos casa vez mais precisos.

O desenvolvimento não se limita à capacidade e velocidade de processamento dos processadores, mas também é "empurrado" pela enorme variedade de componentes e sistemas existentes no mercado, que são desenvolvidos de acordo com a necessidade dos consumidores/usuários.

Com a necessidade de desenvolver componentes que conseguissem controlar máquinas de modo sistemático à distância, de forma compacta e com objetivos bem definidos, surgiram os sistemas embarcados na década de 1960.

De forma genérica, sistemas embarcados podem ser definidos como sistemas que contem pelo menos um microprocessador, o qual é responsável por uma atividade específica. Apresentam basicamente os mesmos componentes que um PC (processador, memória, interface, assim por diante), mas diferenciam-se no ponto em que se limitam a executar bem uma única tarefa, repetidamente.

O primeiro sistema embarcado a ser produzido e colocado em teste foi o *Apollo Guidance Computer,* que foi "embutido" no projeto *Apollo* e na época foi considerado o item de maior risco no projeto.

Com o tempo, surgiram os primeiros sistemas fabricados em larga produção, que também eram voltados para a área militar e operavam em funções específicas, como por exemplo, um computador guia de um míssil americano.

Como o sistema embarcado mostrou-se importante para os projetos militares, não demorou muito para surgirem as primeiras máquinas comerciais com estes componentes.

Com a diminuição gradual do tamanho dos componentes e de seu valor agregado, juntamente com o aumento da capacidade de processamento, tornou-se lucrativo inserir estes componentes nos equipamentos. Além disso, em muitos casos é interessante substituir componentes analógicos como transistores, resistores e potenciômetros, por um algoritmo executado internamente aos microprocessadores.

Atualmente, a utilização de sistemas embarcados é intensa e de certo modo chega a superar em números a procura de PC. Muitas vezes compra-se um equipamento sem saber que o mesmo faz uso da tecnologia e do conceito de sistemas embarcados (equipamentos

domésticos, brinquedos). Pode-se verificar hoje os embarcados produzidos em larga escala para a área comercial, como, celulares, kits de aprendizado, *pagers*, mp3/mp4, linha branca, calculadoras, videogames, etc.

A Figura 1 mostra uma placa de aprendizado (kit de desenvolvimento com ARM9) com dimensões reduzidas que pode receber sistemas operacionais como Linux e Windows e assim compor a parte principal de um sistema embarcado.

Figura 1: Kit Olimex SAM9 L926, retirado do Manual OLIMEX.

Como apresentado na Figura 1, alguns destes equipamentos, devido à necessidade ou até mesmo para pesquisas, trabalham com sistema operacional embarcado, particularmente naquelas soluções que necessitam de uma múltiplas interfaces, com o usuário e/ou com outros sistemas.

O sistema operacional (SO) permite a interatividade entre máquina e usuário, ou seja, é possível organizar cada processo com uma interface adequada, sincronizando com os dispositivos de manipulação, teclado, mouse, tela, entre outros dispositivos que estiverem acoplados ao equipamento.

O SO tem sua história iniciada na segunda geração da computação, década de 1950, com o surgimento de seu conceito apenas e, a partir daquele momento, vem crescendo e atualmente apresentam grande número e variedade. Esta variedade pode ser notada facilmente em uma loja de celulares, onde se encontram celulares com o Windows Mobile, Android, BlackBerry OS, Symbian, iOS, Palm OS, etc.

Essa variedade vem acompanhada de grandes recursos internos, que combinados com os embarcados adequados, auxiliam no desenvolvimento de novidades tecnológicas impulsionando o desenvolvimento e fortalecendo a evolução da eletrônica digital.

A grande e crescente capacidade de processamento dos microcontroladores embarcados e também da grande variedade e apoio ao usuário fornecido pelos SOs

existentes no mercado, abrem portas para o início de várias pesquisas realizadas nesta área, e é seguindo esta linha de sistemas embarcados com sistema operacional que surge o trabalho aqui apresentado.

1.1. Objetivo

O objetivo é propor uma plataforma com acesso via Web para reconhecimento de áudio, composta por microcontrolador *ARM* em ambiente de desenvolvimento *Linux*, onde o usuário submete um arquivo ou um trecho de áudio e através de algoritmos matemáticos de processamento de áudio, o mesmo deve ser identificado.

1.2. Estrutura da Monografia

A monografia está dividida em 6 capítulos, sendo os primeiros capítulos dedicados à uma introdução ao assunto e também, à elaboração de uma descrição dos principais conceitos e ferramentas utilizadas para o desenvolvimento do projeto em questão. Já o capítulo 4 foi elaborado para descrição do desenvolvimento, abrangendo a compilação dos algoritmos e também a elaboração e formatação do *website. O*s capítulos finais são referentes às análises dos resultados obtidos e também sobre algumas conclusões finais sobre o projeto.

2. Conceitos e Descrição técnica

Neste capítulo serão apresentados alguns conceitos que foram de extrema importância no desenvolvimento do projeto, junto com o equipamento que foi utilizado para a elaboração do mesmo.

2.1. Sistemas Embarcados

De forma geral, sistemas embarcados podem ser definidos, segundo CUNHA (2008), como microprocessadores que realizam uma ou poucas atividades determinadas, repetidamente, com alta capacidade de processamento, evitando a paralisação ou ocorrência de pane no sistema.

Normalmente nestes sistemas, o programa a ser executado é armazenado em memória FLASH ou ROM, de início já é possível prever uma limitação importante no quesito memória, diferentemente de um PC onde as informações podem ser alocadas em disco rígido. Esta memória de armazenamento não elimina a necessidade da existência de memória RAM em um embarcado.

Dependendo de sua finalidade, um sistema embarcado pode ou não apresentar uma interface com usuário (teclado, mouse, tela, entre outros), pois pode realizar apenas uma ou poucas operações simples, por exemplo, disparar um bip quando certo botão for acionado. Este exemplo de sistema pode não apresentar uma interface por questão de simplicidade e segurança do projeto, mas os que necessitam de inúmeras tarefas, *feedback's,* análise e alterações de dados, certamente apresentarão interface que permita a manipulação e também o acompanhamento das variáveis existentes no sistema.

Outra vantagem atual é a boa variedade de arquiteturas de microprocessadores/microcontroladores existentes no mercado *como ARM, ATMEL, PIC, PowerPC*, entre outras, o que gera uma grande possibilidade de escolha da melhor opção para cada aplicação.

Neste trabalho optou-se pela utilização da arquitetura *ARM*, que foi primeiramente nominada, baseado em GOMES, LEITE e CAETANO (2005), como *Acorn RISC Machine*, quando criada pela empresa *Acorn Computers Limited* em 1984, mas passou a ser chamada de *Advanced RISC Machine*. Esta arquitetura possui processador de 32 bits e é usada em larga escala dentro da eletrônica digital e informática.

Foi desenvolvida para atender os sistemas educacionais britânicos, mas com o tempo e seu desempenho, tornou-se o primeiro processador com conjunto reduzido de informações (RISC) comercial. Atualmente apresentam características importantes como baixo consumo de energia e pouca área de ocupação de silício para sua fabricação.

2.2. Linux

Linux é certamente um dos sistemas operacionais mais difundidos e utilizados no mundo. Possui características atraentes para ser empregado em sistemas embarcados, como, o *Kernel* (núcleo), que pode ser definido na instalação dependendo da aplicação que o usuário necessitar. Este sistema foi desenvolvido na década de 1990 por Linus Torvalds. Hoje em dia é considerado o mais significativo participante da *Linux Foundation,* e é um *software* livre (GPL), que pode ser usado em praticamente qualquer aplicação, e melhor ainda, pode ser modificado livremente sendo oferecido em versões distribuídas com diferentes licenças.

Atualmente o SO Linux pode ser encontrado em diversas distribuições, as quais possuem configurações diferentes (interfaces gráficas, acesso ao código fonte, etc), que podem diferenciar uma distribuição da outra. Cada uma tem suas particularidades que atendem as necessidades de cada desenvolvedor em específico. Pode-se citar algumas distribuições como o Ubuntu, Fedora, Slackware, Kurumim, Debian, entre outros.

Em relação à sua estrutura interna, pode-se dizer que o Linux é composto basicamente por 2 sistemas: O *Kernel*, *Rootfilesystem* e um programa externo, *Bootloader*, responsável por carregar o Linux na memória.***Bootloader***

O *bootloader*, também chamado de gerenciador de *boot*, realiza o "*boot*" no sistema, ou seja, é responsável por carregar o sistema operacional, junto com suas configurações, escolhido previamente pelo desenvolvedor. Feito isto, o sistema passa obter o controle sobre o equipamento.

Este gerenciador não é obrigatoriamente um único programa, podendo ser encontrado como uma cadeia de programas, e também pode ser realizado de maneiras distintas. Esta variedade de *bootloaders* e sua flexibilidade chegou à um ponto que pode-se trabalhar mais de um sistema operacional em uma máquina.

- ***Rootfilesystem***

Traduzido para o português o termo *Root Filesystem* tem-se "sistema de arquivo raiz", o que define satisfatoriamente o que realmente é este sistema. É o sistema responsável pela leitura e escrita de arquivos. Todos os arquivos com as configurações internas, as bibliotecas, juntamente com os scripts, são armazenados no *rootfilesystem*.

- ***Kernel***

Kernel, ou núcleo, é basicamente o centro que gerencia a comunicação entre usuário, memória, hardware acoplado e barramento de comunicação. Serve de ponte entre aplicativos e o processamento em nível de hardware. Na Figura 2 pode-se ter uma ideia da atuação deste sistema.

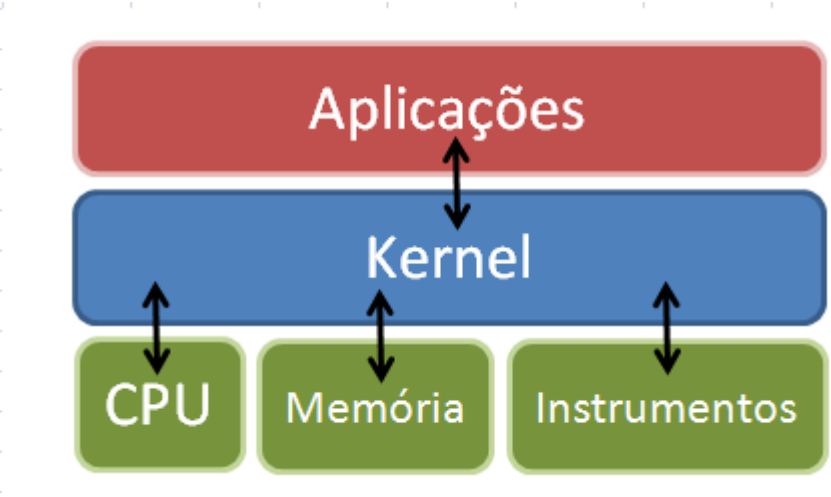


Figura 2: Exemplo de atuação do *Kernel* em uma máquina.

2.3. Kit *Tiny* 6410 *Friendly Arm*

O kit *Tiny* 6410 é fabricado pela empresa chinesa *Friendly Arm*, especialista no projeto de confecção de placas que fazem uso da arquitetura ARM, como a *Tiny* 210 que possui core Samsung S5PV210, a *Mini* 2440, entre outras.

Referente à placa utilizada no projeto, *Tiny* 6410, seguem algumas características e informações:

- CPU: Samsung S3C6410A ARM1176JZF-S 533 Mhz.
- RAM: 128 MB / 256 MB DDR RAM, 32 *Bits*.
- *Flash*: 256 MB / 1 GB NAND *Flash*.
- Saída do usuário: 4 LED's.
- Saída e entrada analogical.
- Interfaces: USB, SERIAL, LCD, Câmera CMOS, SPI, Tela sensível ao toque.
- Sistemas Operacionais: Windows CE 6, Linux 2.6, Android e XUbunto.
- Alimentação: 5 volts.
- Botão *Reset*.
- Entrada para acoplar cartão SD.
- *Beeper*: PWM *Buzzer*
- RTC: *Real time clock*, alimentado por baterias

Mais informações sobre a placa, como desenhos, *data sheets* dos componentes existentes, pode ser encontradas no MANUAL *FriendlyArm* Tiny6410. Na Figura 3 podem ser vistas algumas das entradas e componentes citados.

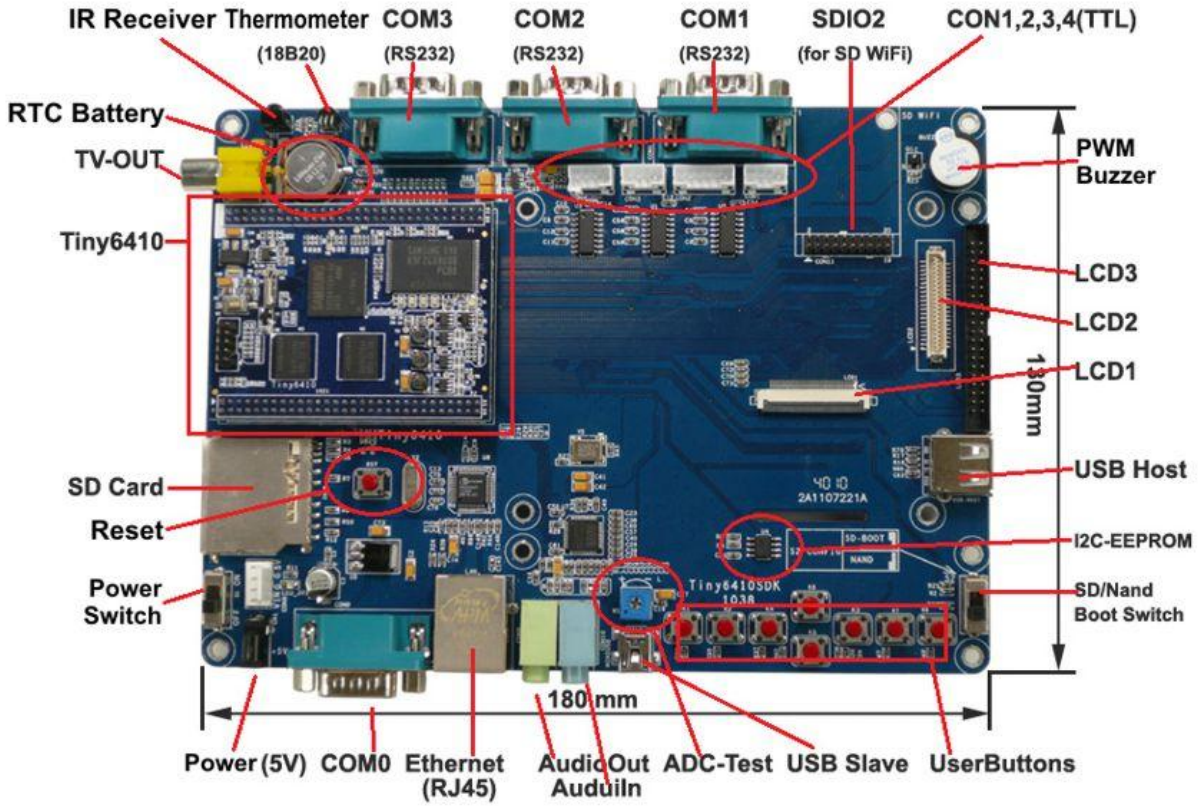


Figura 3: Plana Tiny 6410, retirado do MANUAL FriendlyArm Tiny6410

3. Descrição do projeto

O kit *Tiny* 6410, como descrito anteriormente, possui diversas entradas e saídas para periféricos, porém para a realização deste trabalho não será necessário utilizar todo este arsenal, apenas foi usada a entrada de cabo *ethernet,* uma entrada USB para utilização do teclado e a tela *LCD* para visualização dos comandos/diretórios armazenados no kit.

A entrada de rede, *Ethernet* (RJ45), é importante, pois é através desta porta que o *webite* se manterá no ar, dando suporte aos usuários. Em suma, os algoritmos responsáveis pela codificação do áudio e os responsáveis pela interface gráfica do *site*, estão armazenados na memória NAND da placa, e são acessados pelos usuários via *web* no endereço referente ao protocolo de internet (IP) da placa.

Os algoritmos responsáveis pela codificação do áudio são considerados o coração do endereço *web*, bombeando informações para que cada subsistema tenha o "suporte" necessário para realizar sua respectiva função de acordo com o esperado. Se ocorrer algum imprevisto, os sistemas adjacentes irão "falhar" exibindo mensagens de notificação. Dentre estes sistemas, pode-se citar o banco de dados, o de submissão de dados e o reconhecimento do áudio propriamente dito.

Paralelo a todo este processo, existe a etapa de *Login*/Cadastramento, que deve ser executada pelo usuário antes de usufruir das atividades que o *site* possui. Digamos que esta etapa gera um pouco de segurança, pois se tem o controle dos usuários que enviam os dados. A Figura 4 exibe um diagrama relacionado ao site, com as etapas que devem ser seguidas.

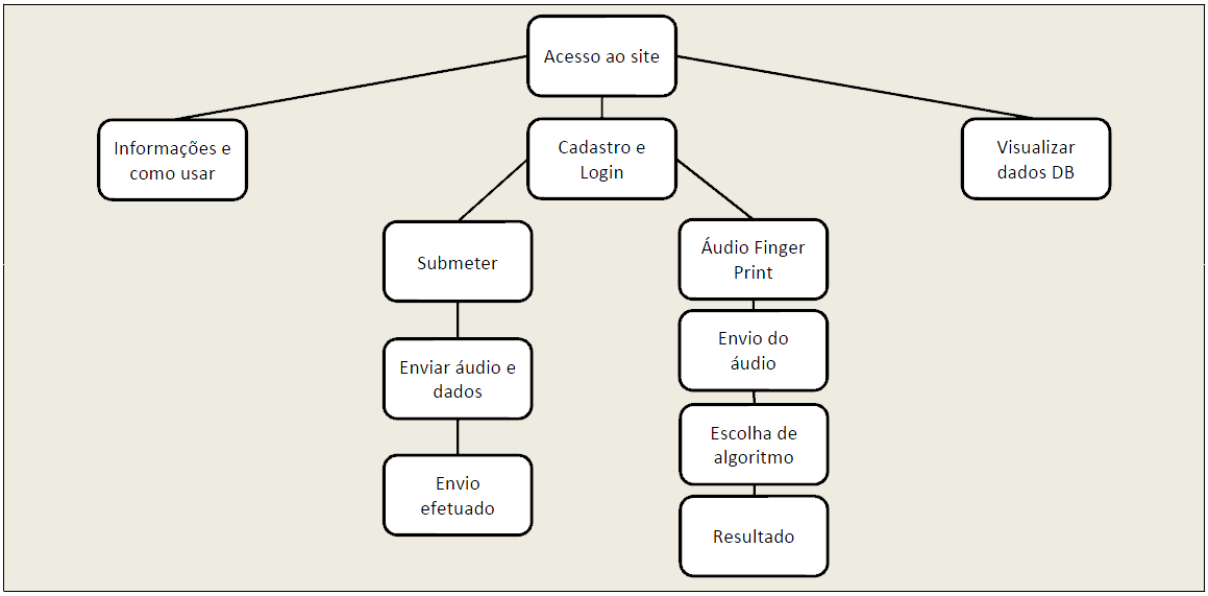


Figura 4: Diagrama das funções do site

3.1. Website

Manter o projeto todo em execução via *web*, é interessante no ponto de disponibilizar o projeto para consultas de qualquer pessoa e também receber informações que contribuirão para o crescimento da ideia/projeto em questão.

Como dito anteriormente, o *site* é o meio de comunicação entre o usuário e os programas, mantendo-os na *web* podem ser acessados de qualquer lugar e a qualquer momento e, evidentemente, com a limitação de disponibilidade de acesso à *internet*. Na Figura 5 tem-se o *layout* da página principal do *site*.

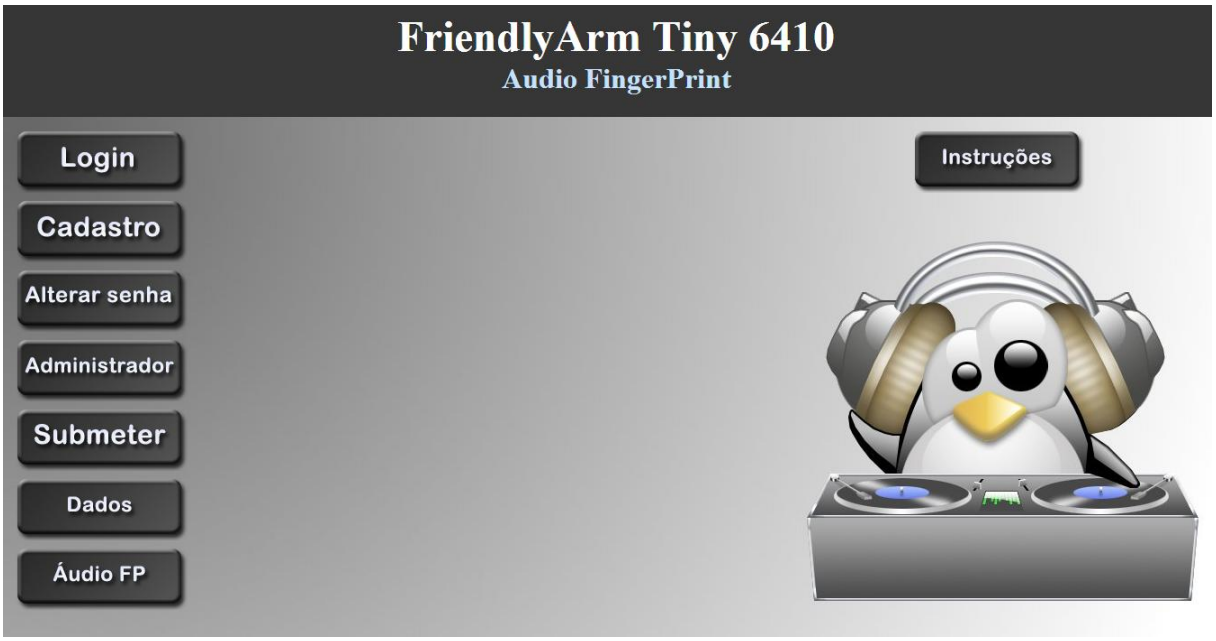


Figura 5: Layout da página principal

É constituído de ícones os quais são responsáveis por iniciar cada um dos subsistemas envolvidos. Possui 7 *links,* localizados no canto esquerdo, que representam as principais atividades executadas pelo *site*, como pode ser visto na Figura 5, e um link localizado no canto superior direito, sendo este responsável por explicar ao usuário como navegar pela página.

Dentre as 7 funções principais e o *link* adjacente, tem-se:

- ***Login*:** Responsável pelo início da sessão de um usuário já cadastrado.
- **Cadastro:** Arquiva dados das pessoas interessadas em participar do *site*, e que ainda não foram cadastradas, ou seja, que estejam realizando o 1º acesso.

- **Alterar Senha:** Caso o usuário necessite de realizar uma alteração de senha, seja por segurança ou outros motivos. Só será necessário que saiba a senha anterior.
- **Administrador:** É um campo voltado apenas para o desenvolvedor e/ou administrador do site. Nele podem ser analisados os dados de restrição de cada usuário.
- **Dados:** Dentro deste campo é possível analisar todas as músicas armazenadas dentro do banco de dados.
- **Áudio FP:** "*Audio Finger Print*" é o *link* responsável pelo reconhecimento de áudio, dentro dele deve-se carregar o arquivo e escolher entre duas opções de algoritmos.
- **Instruções:** É um *link* que auxilia o usuário a navegar pelo site, contendo passo a passo o que deve ser feito para cada uma das aplicações existentes.

Estes itens formam o corpo da página na web e dão suporte para toda a estrutura do projeto. Seu desenvolvimento e método de criação serão descritos posteriormente.

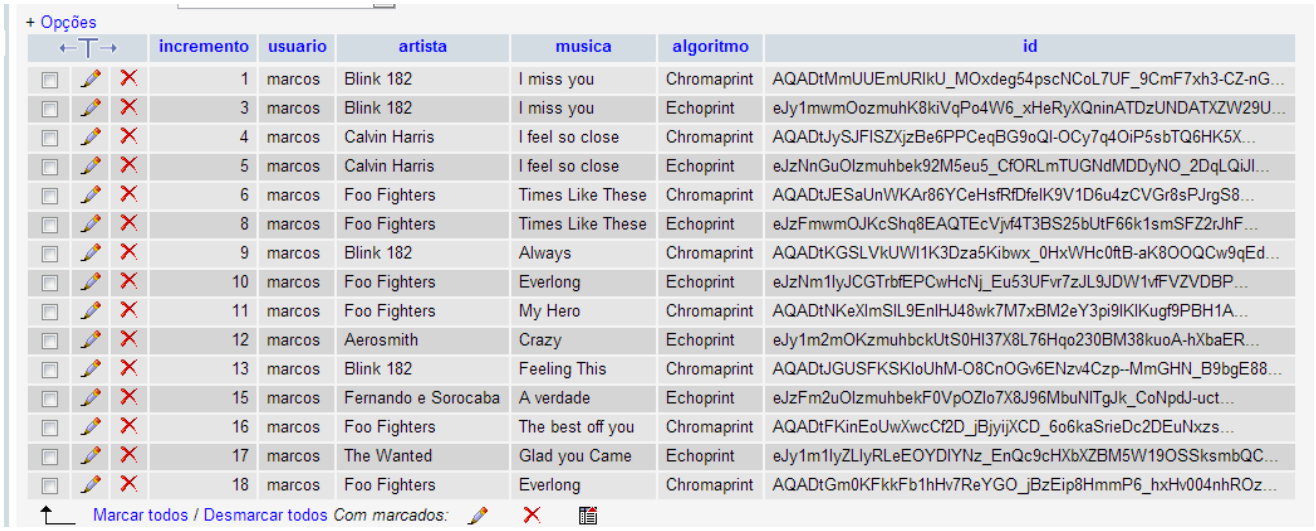
3.2. Banco de dados

O banco de dados é um sistema de armazenamento de informações que podem ser de grande sigilo ou não, mas certamente são de grande importância para o funcionamento e consultas futuras. Neste caso, o banco de dados é peça fundamental, principalmente, para o funcionamento do sistema de consulta, fazendo parte do algoritmo de busca interno e também para simples consultas realizadas por usuários.

Pode-se considerar como banco de dados, seguindo MATTOSO (2007), qualquer forma de armazenamento de dados e informações interessantes para certa pessoa, empresa sistema, e até mesmo um arquivo de extensão *txt* pode funcionar como armazenamento de dados. Porém, com o intuito de realizar o armazenamento com mais segurança e também organização, foi utilizado o *MySQL*, que é um sistema de gerenciamento de banco de dados utilizando a linguagem SQL como interface, e também a ferramenta *phpMyAdmin*, aplicativo *Web* que possibilita controlar e manipular o *MySQL*, juntamente com suas tabelas e bases de dados.

Com a utilização do *phpMyAdmin* a manipulação dos banco de dados se torna muito mais rápida e prática, pois é possível ver todo seu conteúdo com uma interface gráfica extremamente intuitiva em forma de tabelas, como é evidenciado na Figura 6.

A formatação em formas de tabelas deixam ainda mais intuitiva as buscas que serão realizadas internamente. Internamente o banco responsável pelo armazenamento das informações referentes às músicas está dividido em 6 colunas, cada uma com informações importantes para cada sistema ou algoritmo dentro do projeto, e o banco responsável pelo sistema de cadastro/*login* do site está dividido em duas tabelas, cada uma com 5 colunas de informações.



+ Opções

	incremento	usuario	artista	musica	algoritmo	id
	1	marcos	Blink 182	I miss you	Chromaprint	AQADtMmUUEmURlkU_MOxdeg54pscNCoL7UF_9CmF7xh3-CZ-nG...
	3	marcos	Blink 182	I miss you	Echoprint	eJy1mwmOozmuhK8kiVqPo4W6_xHeRyXQninATDzUNDATXZW29U...
	4	marcos	Calvin Harris	I feel so close	Chromaprint	AQADtJySJFISZXjzBe6PPCeqBG9oQI-OCy7q4OiP5sbTQ6HK5X...
	5	marcos	Calvin Harris	I feel so close	Echoprint	eJzNnGuOlzmuhbek92M5eu5_CfORLmTUGNdMDDyNO_2DqLQiJI...
	6	marcos	Foo Fighters	Times Like These	Chromaprint	AQADtJESaUnWKAr86YCeHsfRfDfelK9V1D6u4zCVGr8sPJrgS8...
	8	marcos	Foo Fighters	Times Like These	Echoprint	eJzFmwmOJKcShq8EAQTEcVjvf4T3BS25bUtF66k1smSFZ2rJhF...
	9	marcos	Blink 182	Always	Chromaprint	AQADtKGSLVkUWI1K3Dza5Kibwx_0HxWHc0ftB-aK8OOQCw9qEd...
	10	marcos	Foo Fighters	Everlong	Echoprint	eJzNm1IyJCGTrbfEPCwHcNj_Eu53UFvr7zJL9JDW1vfFVZVDBP...
	11	marcos	Foo Fighters	My Hero	Chromaprint	AQADtNKeXlmSIL9EnlHJ48wk7M7xBM2eY3pi9lKlKugf9PBH1A...
	12	marcos	Aerosmith	Crazy	Echoprint	eJy1m2mOKzmuhbckUtS0HI37X8L76Hqo230BM38kuoA-hXbaER...
	13	marcos	Blink 182	Feeling This	Chromaprint	AQADtJGUSFKSKIoUhM-O8CnOGv6ENzv4Czp--MmGHN_B9bgE88...
	15	marcos	Fernando e Sorocaba	A verdade	Echoprint	eJzFm2uOlzmuhbekF0VpOZlo7X8J96MbuNlTgJk_CoNpdJ-uct...
	16	marcos	Foo Fighters	The best off you	Chromaprint	AQADtFKinEoUwXwcCf2D_jBjyijXCD_6o6kaSrieDc2DEuNxzs...
	17	marcos	The Wanted	Glad you Came	Echoprint	eJy1m1IyZLlyRLeEOYDlYNz_EnQc9cHXbXZBM5W19OSSksmbQC...
	18	marcos	Foo Fighters	Everlong	Chromaprint	AQADtGm0KFkkFb1hHv7ReYGO_jBzEip8HmmP6_hxHv004nhROz...

Marcar todos / Desmarcar todos *Com marcados:*

Figura 6: Visualização do banco de dados pelo phpMyAdmin

As 6 colunas que formam o banco de dados “musical” são:

- **Incremento:** É apenas um campo para o desenvolvedor, onde se pode conhecer quantas músicas existem no banco de dados, pois cada vez que uma é adicionada, o campo incremento tem seu valor acrescido de uma unidade.
- **Usuário:** Toda vez que o *site* receber algum áudio para codificação, será armazenado o responsável, ou seja, o usuário que enviou os dados, assim tem-se o controle do responsável por cada informação enviada.
- **Artista:** É o cantor/banda da música enviada
- **Música:** Nome da música
- **Algoritmo:** Nome do algoritmo usado para codificar, é importante para que não ocorra confusões no momento de realizar comparações das impressões digitais (ID).
- **Id:** São as impressões digitais resultantes da codificação do áudio por cada algoritmo em específico.

E as 5 colunas que completam as tabelas responsável pelo armazenamento de usuários são:

- **Incremento:** Tem a mesma função que anteriormente.
- **Nome:** Nome completo do usuário, enviado e armazenado no momento em que a pessoa faz seu cadastro no site.
- **Usuário:** Usuário de *login* para acesso ao site. Será escolhido pelo usuário no momento de cadastro; pode ser escolhido qualquer nome que ainda não tenha sido usado.
- **Email:** Email ativo do usuário; qualquer comunicação entre o administrador e o usuário será feita por este email.
- **Senha:** Este campo é responsável pela segurança da conta do usuário, somente ele e o administrador possuem acesso a este campo.

O armazenamento destas informações é uma etapa muito importante no desenvolvimento deste projeto. Com o banco de dados de músicas poupa-se tempo de execução e com isso tem-se o reconhecimento da música mais rápido. Além disso, o índice de acerto cresce, pois as informações armazenadas são "fiscalizadas" comprovando sua veracidade.

E o banco de dados de informações referentes aos usuários, executa um papel de segurança dentro do *site*, a partir do momento em que para fazer uso de todas as funções do site, o usuário deve estar com acesso liberado e, assim, tem-se o administrador tem a capacidade de analisar as atividades dos usuários.

3.3. Algoritmos de Codificação

Os algoritmos de codificação tem a missão de transformar o áudio recebido em um código, este deve ser robusto e capaz de aumentar a compatibilidade de sinais parecidos e diminuir para sinais diferentes. Essa compatibilidade será realizada com comparações do código gerado com os que estão armazenados dentro de um banco de dados.

Existem vários meios de realizar a codificação de certo áudio. Aqui será abordado um método denominado de *Áudio FingerPrint*, baseando-se no GUIA *audio fingerprint* e também em HAITSMA, KALKER (2002), o qual trabalha gerando uma impressão digital a partir de um sinal de música.

- **Áudio Fingerprint**

Um algoritmo de *FingerPrint* (FP) robusto, deve levar em conta as características de percepção do áudio . Quando se tem dois arquivos de áudio que são semelhantes para o ouvido humano, mesmo que seus códigos binários sejam diferentes, eles devem possuir

impressões digitais semelhantes, logo estas impressões não podem ser sensíveis às pequenas variações, como uma comparação *bit* a *bit*.

Um exemplo de comparação seria a impressão digital humana, que primeiramente é armazenada em um banco de dados e quando foi requisitada novamente apresenta a correta equivalência, mesmo com a impressão do sujeito, dono da impressão digital, sujeita à variações, como manchas e cicatrizes.

Algumas das mais utilizadas técnicas em impressões digitais são: análises de espectro, análise das bandas e “cruzamento de zero”.

A utilização de impressões digitais permite também a codificação do áudio depois de passar por uma compressão, o que é muito difícil ocorrer se fosse usado o método de codificação *bit* a *bit*. Mesmo podendo causar algumas alterações no áudio original, o FP é capaz de “superar” estas pequenas alterações. A compressão do áudio é um método utilizado em armazenamento do som, por exemplo, quando for gravar uma música alocada no PC em CD.

O sistema pode ser separado em algumas partes, sendo a primeira a obtenção e digitalização do sinal: determina-se qual o tamanho da amostra necessária para que uma impressão digital (ID) seja satisfatoriamente gerada. Uma ID é satisfatoriamente gerada se os dados nela contidos puderem ser utilizados para a identificação através da ID do sinal original.

O sinal pode ser obtido "*over the air*", isto é, quando o sinal é proveniente de um microfone ou outro transdutor, gerando algum tipo de ruído e “poluindo” o sinal armazenado. Para sistemas "*over the air*" se exige grande robustez, velocidade de processamento e ainda que a amostra necessária não seja demasiado grande, pois nesse caso ocorreria perdas de informações.

A segunda etapa é o processamento do sinal e analise para se gerar a ID, onde diversas abordagens podem ser utilizadas. Pode-se utilizar transformadas para análise em frequência, bancos de filtros, transformadas *Wavelet* para análise em frequência e tempo e, também, pode-se fazer análise de entropia, energia, técnicas de visão computacional e diversos outros métodos, todos ainda em estudo. Nos próximos tópicos serão abordados as duas técnicas utilizadas no projeto.

A última etapa consiste em tomar a ID obtida e compará-la a um banco de dados com IDs previamente selecionadas associadas aos dados relevantes que se deseja reaver.

Deve-se lembrar, novamente e enfaticamente, que o sistema não é uma busca por "*meta-tags*" ou comparação *bit* a *bit* de dados de áudio digitalizado: o sistema de impressão digital de áudio tenta fazer com que o computador "escute", tomando tão quanto possível parâmetros fisiológicos da audição, os sinais possam associar.

Diversas aplicações podem derivar de um sistema de "*Acoustic Fingerprint*": de duplicadores de "*playlists*", verificação de utilização de canções com *copyright*, separação de músicas em uma "*playlist*" desorganizada por artista ou estilo, ou simplesmente a curiosidade ao se escutar uma música que não se conhece.

Os programas utilizados no presente projeto são algoritmos já disponíveis e *open source*, eles são responsáveis por gerar a impressão digital. São eles: *Echoprint e Chromaprint.*

- ***Echoprint***

Processamento e manipulação de áudio

O *Echoprint* é um algoritmo, processamento baseado em ELLIS et al (2010) e ELLIS, WHITMAN, PORTER (2011), com código aberto (mais informações na BASE DE DADOS do echoprint), ou seja, qualquer pessoa que quiser fazer uso desta ferramenta pode fazê-lo, mas para isto é necessário alterar algumas configurações e também algumas bibliotecas, porém isto será discutido posteriormente.

O *Echoprint* trabalha com velocidade e eficiência gerando 12 *hashes,* conjunto de *bits,* gerados pelo algoritmo responsável por cifrar dados (geralmente representada na base hexadecimal), a cada segundo de música/áudio.

Com o intuito de ser robusto às modificações nos espectros e também às variações devido ao ruído que pode ser imposto em músicas gravadas "*over the air*", o algoritmo só leva em consideração o tempo relativo em sucessivos *onsets* do áudio.

Onset pode ser definido, com base em BELLO et al (2005), como o início de um som, ou seja, quando o áudio apresenta sua primeira nota musical, fazendo com que o espectro de amplitude caminhe do nível 0 para seu primeiro pico musical (Figura 7). Obtenção de algoritmos para detecção de *onset* é uma área prolífica de pesquisa. Diversos métodos para análise vem sendo estudados, desde analise de Fourier, analises em multi resolução, com informações no tempo e frequência.

A magnitude dá o sinal complexo passa-banda, e em cada banda é comparado com um limiar (*threshold*) exponencialmente em decaimento, e quando este limiar for ultrapassado o *onset* é gravado. Neste ponto o limiar recebe um novo valor, equivalente a 1,05 x Npk, onde Npk é o novo pico de sinal. Um algoritmo adaptativo leva em consideração os intervalos entre *onsets* (IOI), e neste caso decai o limiar quando estes intervalos forem menores e o aumentam caso ocorra o contrário. A taxa de ocorrência de reais *onsets*, no *Echoprint*, é de 1 *onset* por segundo por banda.

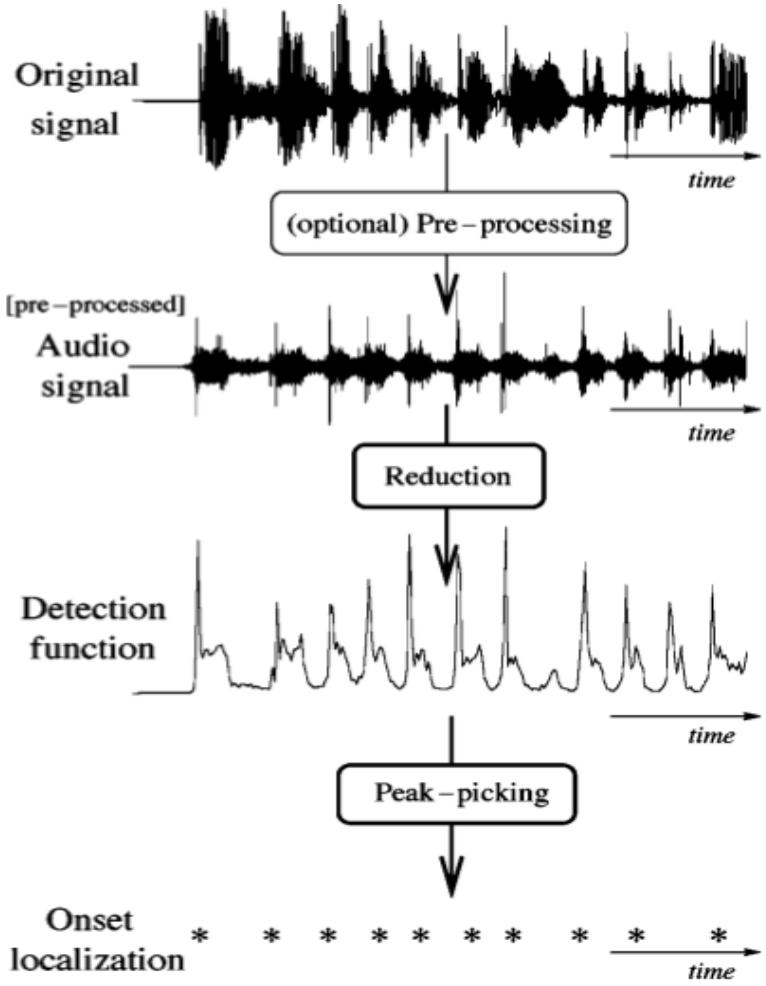


Figura 7: Detecção de onsets, BELLO et al (2005)

A formação de um *hash*, para identificação do áudio, ocorre com a junção de pares de IOI's consecutivos quantificados em unidades de 23,2 ms. Considerando que no meio do processo podem ocorrer ruídos, é considerado como o primeiro *onset* ele próprio e os 4 próximos, gerando assim uma robustez maior evitando erros. Logo, 6 *hashes* diferentes (IOI pares) são criados sucedendo os 4 *onsets*.

Assim, a taxa global de *hash* é de aproximadamente 8 (bandas) x 1 (*onset* por segundo) × 6 (*hashes* por início) ≈ 48 *hashes* / segundos. Considerando que cada *onset* é cerca de 1 segundo, o valor quantizado de IOI's é 1/.0232 = 43, ou cerca de 5-6 bits, e um par de *onsets* constitui cerca de 12 bits de informação. Estes bits combinados com os 3 bits de índice da banda geram o *hash* cru, o qual é armazenado dentro do arquivo.

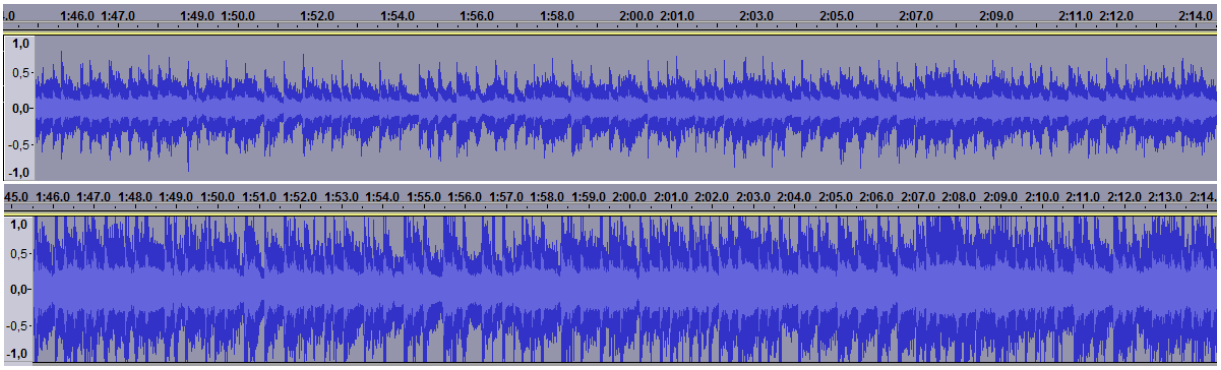


Figura 8: Sinal nomal e sinal gravado OTA, simulados no Audacity

No caso de um arquivo gravado "*over-the-air*", em que o sinal está sujeito a ruídos, faz-se o branqueamento antes de passar pelo processo de detecção dos *onsets*, descrito acima. O branqueamento do sinal tem por fim tornar o sinal com suas componentes descorrelacionadas e variância unitária. Na Figura 8 tem-se dois sinais do mesmo áudio, porém o segundo foi gravado OTA.

Equivalência entre códigos e músicas

Depois de realizar todo o procedimento de manipulação responsável por gerar um código, pelo o qual a música poderá ser identificada, chega-se no ponto em que esse código gerado deve ser comparado com outros para obter uma equivalência. O sistema *Echoprint* faz uso de um banco de dados que armazenam o código propriamente dito (ID) e também as informações denominadas *metadatas* (artista, álbum e nome da música).

O procedimento de busca ocorre através do *Apache Solr*, que é um servidor de código aberto e sem custo, que realiza a indexação de conteúdos e também realiza buscas, podendo ler arquivo de inúmeras extensões (*.txt, .pdf. docx*).

Dentro destes arquivos de armazenamento, cada código de áudio é dividido em segmento de 60 segundos, com seções adjacentes sobrepondo-se em 30 segundos (primeiro segmento 0s até 60s, segundo segmento de 30s até 90s, e assim sucessivamente), isto é feito para eliminar redundâncias quando uma música apresenta várias equivalências.

O servidor retorna as documentações que apresentam a maioria dos *query's* gerados pelo algoritmo, neste caso podem apresentar várias equivalências mesmo sendo muito raro. Para resolver este problema são selecionados os 15 mais parecidos e é feito um histograma para análise de parentesco, e consequentemente o que possuir um maior grau de igualdade será o documento equivalente ao código gerado.

Em caso de ocorrer mais de um documento da mesma música, será mantido o que tiver maior grau de parentesco e os outros serão excluídos, este procedimento ajuda a evitar redundâncias e também libera espaço para armazenamento de músicas diferentes.

Todas estas etapas devem ser seguidas, pois não funcionam de outra maneira e, também pulando alguma etapa, a chance de não ocorrer equivalência nos códigos e/ou

ocorrer com alguma fonte errada é grande. Para exemplificar o funcionamento, tem-se na Figura 9 um fluxograma de todo o processo.

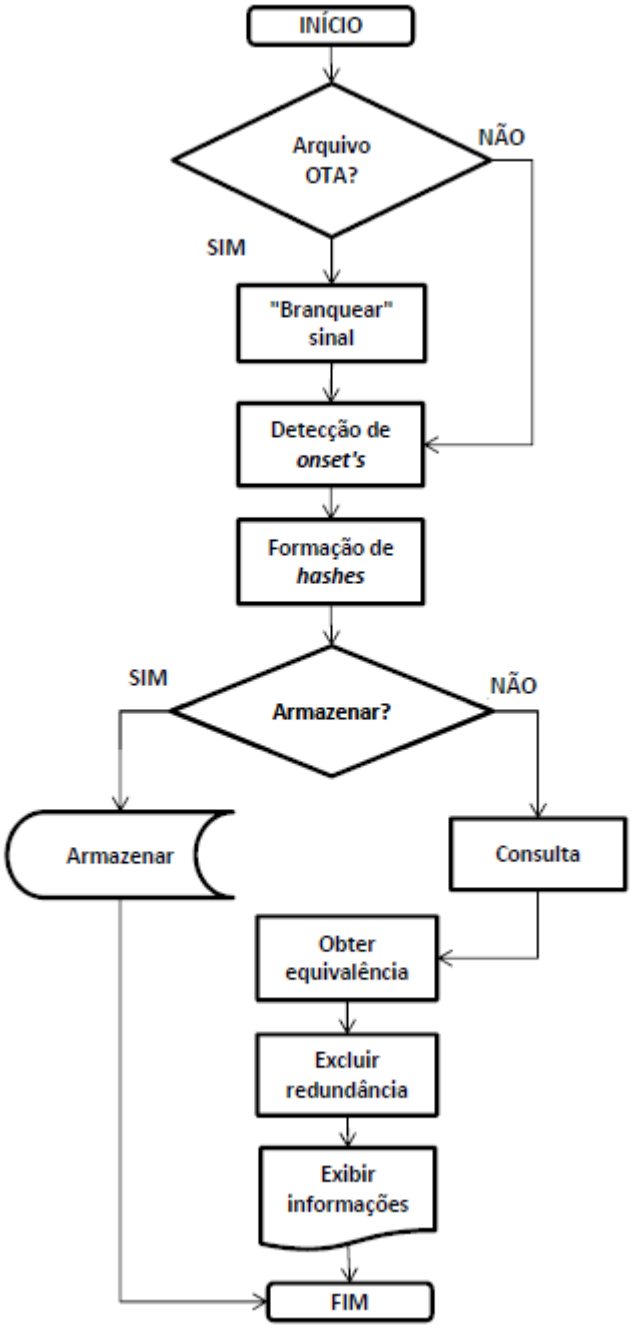


Figura 9: Fluxograma do funcionamento do Echoprint

- ***Chromaprint***

O *Chromaprint* é outro algoritmo *open source* que está vinculado com o trabalho e foi selecionado por trabalhar de uma maneira um pouco diferente do *Echoprint*, mas ainda fazendo uso do sistema de reconhecimento *fingerprint*.

Processamento e manipulação de áudio

O *Chromaprint* faz grande uso da teoria de "Visão Computacional para Identificação Musical", ou seja, a imagem age como um fator praticamente soberano na identificação do áudio. O processamento deste algoritmo é baseado no GUIA CHROMA FEATURES (2004) e em KE (2005).

Geralmente quando "vemos" a música é na forma de ondas, Figura 9, porém esta representação não é muito significativa para a análise em questão, e sendo assim, será realizada uma manipulação nestas "ondas" transformando-as em um espectrograma de frequências, Figura 10, que será fonte de informações importantes como as alterações de intensidades que ocorrem em determinadas frequências no decorrer do tempo.

Muitos algoritmos de identificação musical trabalham nesta linha, analisando picos nas imagens, diferenças entre tempo e frequência, entre outros aspectos. O chromaprint, por sua vez, procura nesta imagem a formação de notas musicais, e sendo assim, tem-se 12 números binários que irão representar as 12 notas musicais (DÓ, DÓ#, RÉ, RÉ#, MI, FÁ, FÁ#, SOL, SOL#, LÁ, LÁ#, SÍ.). Este tipo de informação é chamado de características *Chroma.*

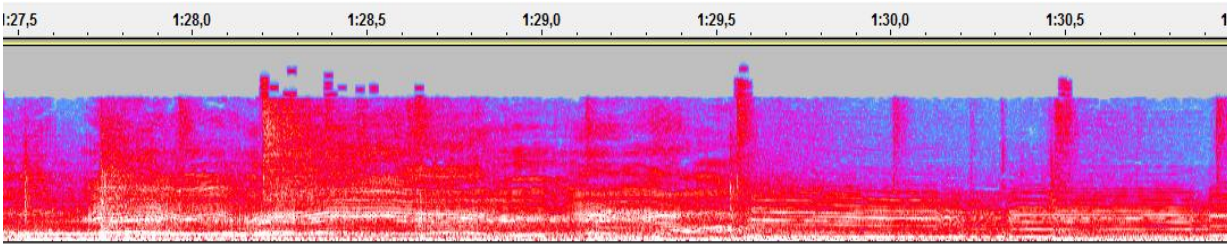


Figura 10: Espectrograma, gerado com base no MANUAL Audacity

Característica *Chroma* é uma representação interessante e poderosa para áudio, em que todo o espectro é projetado em 12 "caixas" que representam os 12 semitons distintos (ou *Chroma*) da oitava musical. Uma vez que, na música, observa-se exatamente uma oitava acima são percebidas como semelhantes, conhecendo a distribuição de "*Chroma*" pode-se obter informações úteis sobre o áudio, e pode até mesmo evidenciar similaridade musical que não era visível no espectro original.

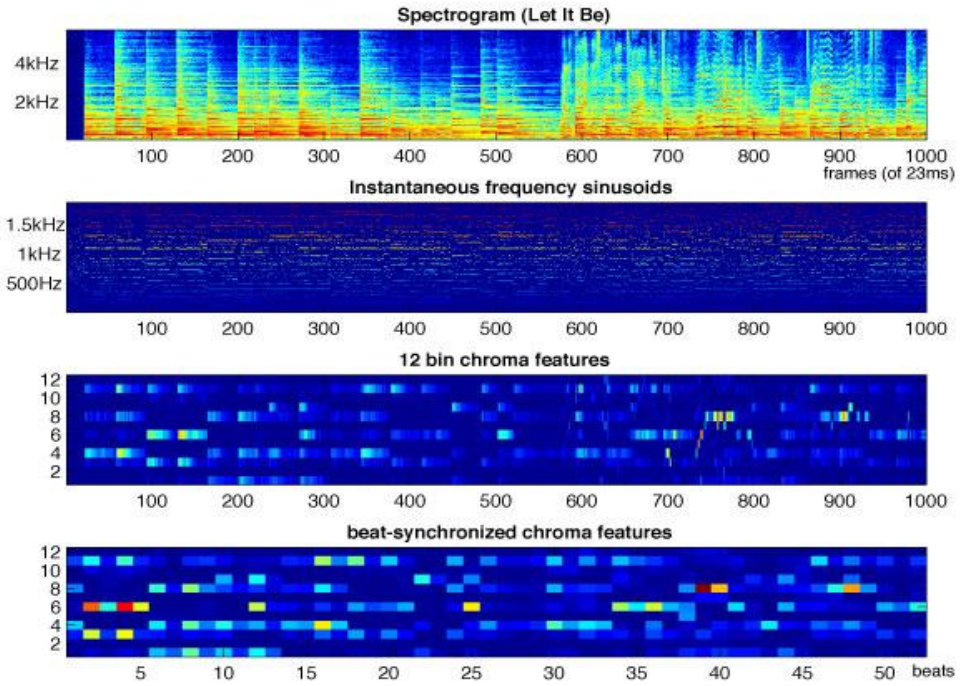


Figura 11: : Transformações e resultado - Chroma features BARTSCH (2005).

O tratamento do áudio como uma imagem, é basicamente o núcleo do algoritmo em questão, pois com o intuito de alcançar um sistema robusto, capaz de identificar sua identidade a partir de um trecho de áudio e também encontrar equivalência correta em meio a milhões de músicas, o *chromaprint* faz uso da transformação do sinal de áudio para sua representação em forma de espectrograma de frequências.

A necessidade desta transformação é explicada pelo fato de que o sinal representado na forma de amplitude x tempo (Figura 9) varia bruscamente quando a música estiver sujeita a certos tipos de ruídos ou mesmo alterações ocasionadas por diferentes meios de gravação.

Esta transformação pode ser obtida com a aplicação da *Short Term Fourier Transform* (STFT) (CRESSONI, 2001) ou Transformada de Fourier em tempo curto, que nada mais é do que a Transformada de Fourier (FT), mas em um trecho apenas no sinal. Aplicando a STFT pode-se obter uma análise espectral de qualidade, pois a FT não mostra ocorrências espectrais. A STFT apresenta um empecilho, pois devido ao seu “janelamento curto” causa interferência no sinal original, mas neste caso como só estamos interessados em um trecho do sinal, esta interferência não causará danos no decorrer do algoritmo.

Após este processo de transformação, de um pedaço do áudio em uma imagem, poderia ser realizada a comparação entre a imagem obtida, com um banco de dados e assim encontrar uma correspondência, já que nestas imagens pode-se encontrar

similaridades e discrepâncias em relação às outras, mas uma simples correlação feita desta maneira levaria um tempo elevado e também uma incerteza muito grande. Para resolver esta situação, a imagem gerada, antes de ser comparada, passará por filtros capazes de manter a informação relevante e eliminar os ruídos.

A classe de filtros utilizada deve ser capaz de evidenciar características que são capazes de diferenciar uma imagem da outra. Estas características são:

- Diferença de energia nas bandas de frequências vizinhas
- Diferença de energia através do tempo em uma determinada frequência
- Mudanças na frequência dominante ao longo do tempo
- Picos de energia no tempo em frequência determinada
- Picos de energia na frequência em um tempo determinado

Seguindo estas características usou-se o filtro *Haar waveletlike*, Figura 12.

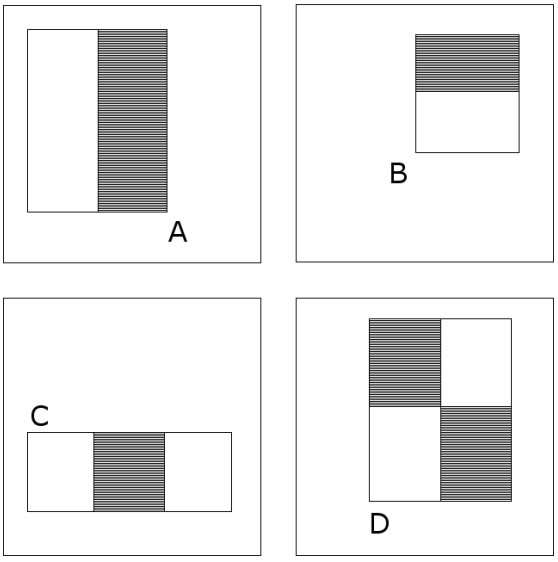


Figura 12: Janelas do Filtro

Após a filtragem da imagem, esta deve passar por um procedimento para que uma assinatura que a represente seja gerada e, para isso, o *chromaprint* faz uso de uma maneira bem simples para obter o áudio correspondente.

Equivalência entre códigos e músicas

Após a passagem dos filtros pelas imagens, para se obter o áudio correspondente faz-se apenas uma diferença entre as imagens binárias geradas depois da passagem dos filtros. Pode-se ver um exemplo na Figura 13 que foi retirado da BASE DE DADOS do *chromaprint.*

Como pode ser observada, a diferença entre a música *Haven* gravada de maneiras diferentes é muito pequena, e o mesmo acontece com a música *Under the ice,* porém quando comparadas entre si, *Haven* e *Under the ice*, apresentam uma diferença acentuada, e é desta maneira que o algoritmo busca suas equivalências dentro de seus arquivos.

O servidor do *Chromaprint* permite o acesso a usuário para busca e submissão de dados, e procedimento para a realização de todos os serviços *online* estão descritos passo a passo na BASE DE DADOS do *chrimaprint*.

Como feito para o *Echoprint*, na Figura 14 encontra-se um fluxograma resumindo todos os passos na execução do algoritmo em si, feita a descrição de dois poderosos algoritmos para identificação de áudio, é importante elaborar uma tabela comparativa sobre o meio de abordagem, de processamento e também de busca entre o *Echoprint* e o *Chromaprint.*

Tabela 1: Comparação entre os algoritmos

Algoritmos	Método	Processamento	Busca
Echoprint	Análise do Sinal	Detecção de *Onset*	Formação de histogramas
Chromaprint	Análise de espectrograma	Busca por "notas musicais"	"Diferença entre imagens"

Heaven FLAC

Heaven 32kbps MP3

Differences between Heaven FLAC and Heaven 32kbps MP3

Under The Ice FLAC

Under The Ice 32kbps MP3

Differences between Under The Ice FLAC and Under The Ice 32kbps MP3

Differences between Heaven FLAC and Under The Ice FLAC

Figura 13: Exemplo chromaprint, retirado da BASE DE DADOS do chromaprint

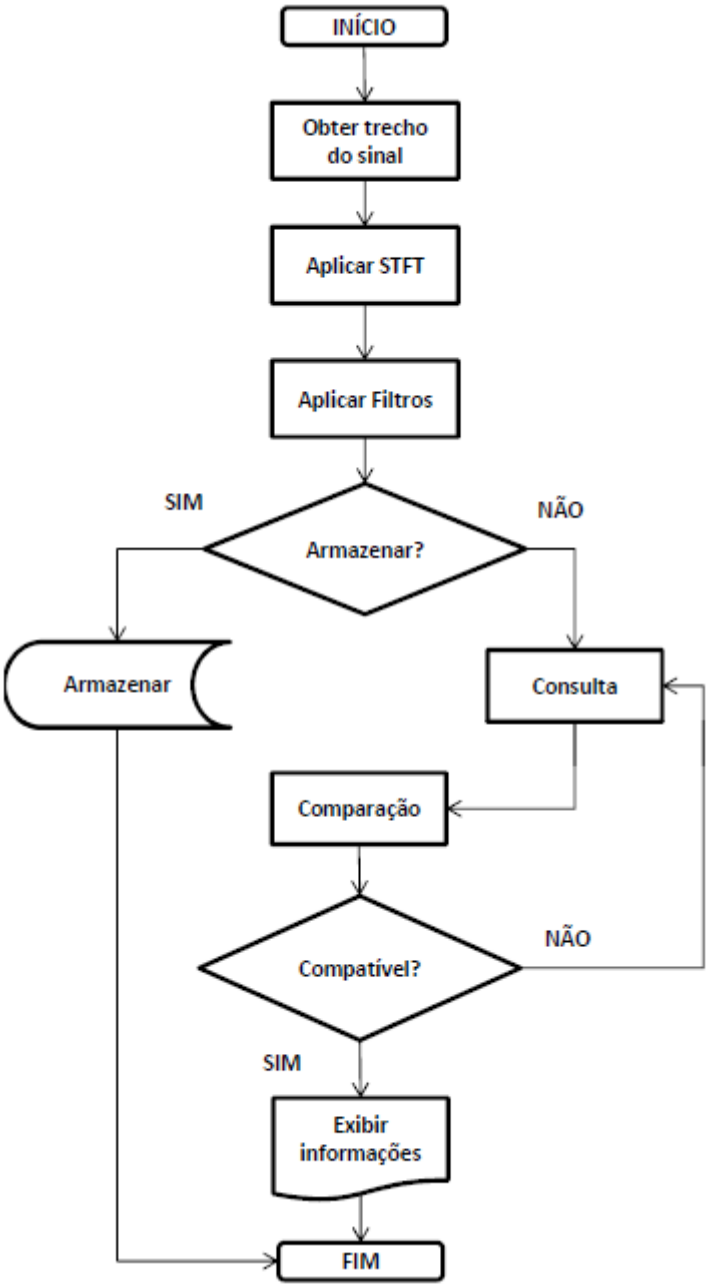


Figura 14: Fluxograma do funcionamento do Chromaprint

4. Implementação e Desenvolvimento

Nesta seção serão discutidos todos os passos e algoritmos desenvolvidos para que o projeto em questão fosse implementado, com todas suas funções em pleno funcionamento dentro do Kit Tiny6410. Para melhor organização e mais fácil compreensão, será abordado um sistema por vez, assim pode-se detalhar com mais cuidado as implementações.

4.1. Compilação dos algoritmos

Aqui serão abordadas todas as etapas que foram superadas para a plena execução do algoritmo. Será dividido em duas partes referentes aos dois algoritmos, pois são processos distintos. Informações sobre comandos e/ou procedimentos foram tiradas do FÓRUM brasileiro de usuários *Linux* e do GUIA *linux.*

- **Echoprint**

Primeiramente o pacote, nomeado *echoprint-codgen* com os algoritmos devem ser descarregados da BASE DE DADOS do *echoprint*. Na pasta conterá todos os arquivos pré-compilados, ou seja, ainda não possuem funcionalidade, precisam ser compilados no computador de uso para formação do arquivo executável, e assim serão reconhecidos pelo sistema do computador.

Porém antes de realizar a compilação, é importante instalar todas as bibliotecas e arquivos necessários para realizar esta ação e também para o perfeito funcionamento do programa. Para a instalação destes arquivos tem-se um comando muito importante denominado *apt-get,* que é utilizado como gerenciador de pacotes no Ubuntu e em outras distribuições *Linux*. Através desta ferramenta pode-se fazer atualização de pacotes (*apt-get update*) e também realizar o *download* de arquivos via terminal (*apt-get install* nome_arquivo_pacote) e por sua eficiência e simplicidade foi usado para instalação dos pacotes necessários.

Para realização da compilação é necessário que o Kit tenha compiladores das linguagens em que foram desenvolvidos os algoritmos, no caso C e C++, e com isso executamos o seguinte comando para pacote de compiladores *gcc* e *g++* (conjunto de compiladores para linguagens de programação):

```
# sudo apt-get install gcc g++
```

O comando *sudo* é importante para executar o comando *apt-get* como usuário *root,* caso contrário o sistema não permitirá que os pacotes sejam instalados.

Agora com os compiladores já instalados, pode-se passar para o *download* das bibliotecas que fazem parte do algoritmo. No *Echoprint* são necessárias as seguintes bibliotecas:

- **TagLib**

Segundo o GUIA TAGLIB, esta biblioteca é responsável pela leitura e manipulação de características de vários formatos populares de áudio (*mp3*, *mp4*, *wav*, entre outros), como *sample rate* e tamanho de faixa.

- **Ffmpeg**

É uma plataforma de computador que grava, converte e cria *stream* de áudio e vídeo em diversos formatos. Ffmpeg é um programa em linha de comando que é composto de uma coleção de *software* livre e bibliotecas de código aberto. Inclui libavcodec, biblioteca de codec de áudio e vídeo, e libavformat, um multiplexador/demultiplexador de conteúdo de áudio e vídeo, informações obtidas no GUIA FFMPEG.

- **Boost**

Com base no GUIA LIBBOOST, é uma biblioteca desenvolvida para ser usada em qualquer projeto e é representada por várias bibliotecas capazes de aumentar a funcionalidade da linguagem C++ (usada no projeto *Echoprint*).

- **Zlib**

Baseando-se no GUIA ZLIB, esta biblioteca é uma plataforma responsável pela compressão de dados.

Logo, para fazer o *download* destes arquivos, faz-se o uso do comando *apt-get,* novamente executando a seguinte linha de comando:

```
# sudo apt-get install ffmpeg libboost1.42-dev libtag1-dev zlib1g-dev
```

Se um pacote não for encontrado, pode ser algum nome incorreto, mas se não for é possível fazer uma busca, pois o arquivo procurado pode estar no sistema com outro nome (por exemplo a biblioteca *TagLib* que no sistema está como *libtag*). Para realizar uma busca dos pacotes existentes basta executar a seguinte linha de comando:

```
# sudo apt-cache search taglib
```

E, como resultado, tem-se todos os pacotes os quais contem *taglib* em suas descrições, assim analisando as informações contidas em cada arquivo encontra-se o nome do arquivo pelo qual estava procurando.

Com todas as bibliotecas e também os compiladores instalados no kit pode-se executar a compilação do arquivo. A compilação deste algoritmo ocorre através do comando *make* na linha de comando. O algoritmo em si é, de fato, uma pasta contendo várias funções que são chamadas pelo programa principal, e como compilar um a um é uma tarefa trabalhosa, existe o arquivo *makefile* que é responsável pelas chamadas e compilação geral, consiste de linhas de dependências/destino.

Como o *makefile* já esta presente no pacote do algoritmo descarregado, não é necessário sua criação, apenas iremos executá-lo com o comando *make*. O comando *make* por sua vez, geralmente vem embutido nos pacotes dos compiladores *gcc/g*++, mas caso não venha tem-se a necessidade de buscá-lo na *internet*, assim executa-se os seguintes instruções na linha de comando:

```
# sudo apt-get install make – Instalação do comando make
# cd /var/www/Echoprint – Entrar no diretório que contém o arquivo makefile
# make – Executar o make para compilação dos arquivos
```

Tudo ocorrendo dentro previsto, será gerado um arquivo executável com o nome de *echoprint-codgen*, e este é o arquivo responsável pela manipulação e codificação do áudio. Quando executado pedirá informações necessárias para seu funcionamento.

Execução do arquivo

Existem 2 etapas na execução do arquivo, sendo a primeira responsável por gerar os códigos e a segunda por realizar a buscar por correspondência dentro da documentação armazenada. Os códigos são gerados a partir da execução do *echoprint-codgen* e para sua execução é necessário passar 3 informações, o áudio, o tempo de início e o trecho do áudio que será abordado. Assim, a instrução executada na linha de comando é ("./" no início é o comando usado para executar arquivos):

```
# ./echoprint-codgen crazy.mp3 30 60 >código.txt
```

Executando esta linha de comando, o algoritmo irá gerar um arquivo denominado código. Este arquivo tem extensão *.txt* e contem informações sobre o áudio em questão (impressão digital gerada, nome do arquivo, entre outras). Tendo o *codigo.txt* pode-se realizar a busca dentro da documentação *online* do *echoprint*, e para isso usa-se o seguinte comando:

```
# curl -F "query=@codigo.txt"
http://developer.echonest.com/api/v4/song/identify?api_key=SUA_SENHA
```

O comando *curl* é responsável por obter dados de um servidor fazendo uso de algum protocolo de comunicação, HTTP, HTTPS, TELNET. Sendo assim ele executa a URL anexada tendo como entrada o arquivo *código.txt*. A resposta é obtida no terminal com várias informações, se tiver interesse apenas no nome e no artista da música a resposta deve ser filtrada, permanecendo apenas as informações desejadas.

- **Chromaprint**

As ferramentas que auxiliarão nesta etapa, e que já foram usadas anteriormente, não precisarão de instalação novamente, pode-se citar *gcc*, *g++*, *apt*, *make* e algumas bibliotecas que serão discutidas.

Para execução do *chromaprint* são necessárias 5 bibliotecas/aplicativos indispensáveis, *ffmpeg*, *fftw*, *taglib*, *boost* e *googletest*. Destas cinco, três já estão instaladas no kit (*ffmpeg, taglib e boost*), e assim tem-se que realizar o download dos outros 2 arquivos.

- **Fftm:** *Fastest Fourier Transform in the West* é uma biblioteca para Transformada de Fourier Discreta computacional.
- **Googletest:** Suporte para executar testes usando a linguagem computacional C++, retirado do GUIA GOOGLETEST.

Para realizar os *downloads* basta executar o comando abaixo no terminal, caso não for encontrado o arquivo, executar uma pesquisa utilizando o *apt-cache search* como descrito anteriormente.

```
# sudo apt-get install fftm googletest
```

Depois de executar a linha acima o algoritmo está pronto para ser compilado no kit. Neste caso, existem algumas maneiras distintas para a realização da compilação e aqui será descrito apenas o meio utilizado no projeto.

Ao abrir o pacote *Chromaprint* notou-se a não existência de *makefiles* e sim de *CMakeLists.txt*. Estes arquivos possuem um meio de compilação diferente dos *makefiles*. De início deve-se fazer o *download* do *CMake*, que é um sistema multiplataforma para

realizar geração automatizada. O *CMake* não gera o software final e sim os arquivos de geração padrão (*makefiles*), portanto, o processo de compilação com o *CMakeLists* possui duas etapas: a formação dos arquivos de geração padrão e a formação do software executável.Antes destes passos é importante fazer o *download* da ferramenta *cmake*, depois disto a compilação pode ser feita normalmente, como segue o código abaixo.

```
# sudo apt-get install cmake
# cd /var/www/chromaprint
# cmake –DCMAKE_BUILD_TYPE=Release –DBUILD_EXAMPLE=ON .
# make
# sudo make install
```

Depois da realização dos comandos citados é gerado um arquivo executável com o nome de *fpcalc* responsável pela formação dos códigos do áudio que serão utilizados para comparação musical.

Execução do arquivo

O mesmo comando utilizado para executar o *echoprint-codgen* é usado para o *fpcalc* (“*./*”), só mudam as entradas necessárias. Para o *chormaprint* só é requisitado o caminho para o áudio que será codificado, se estiver na mesma pasta que o *fpcalc* não é necessário e apenas o nome com extensão já é suficiente. Portando o comando de execução fica da seguinte maneira:

```
# ./fpcalc crazy.mp3 >codigo.txt
```

Assim, teremos o resultado gravado no arquivo *codigo.txt*, que será gerado ao término do processamento do algoritmo. Igualmente ao *echoprint,* este arquivo contém informações importantes para a busca ocorrer de maneira eficiente e sem erros.

A busca neste servidor é um pouco mais complicada, pois não existe um comando próprio para sua realização. O método que o servidor disponibiliza é confuso e bem susceptível a falhas, pois é um método manual. Primeiramente abre-se o arquivo *codigo.txt* e copia o campo *FINGERPRINT*(impressão digital gerada pelo código) e também o campo *DURATION*(duração do trecho usado na codificação). Estes dados devem completar o link de acesso ao servidor do *chromaprint*, como mostrado abaixo:

```
http://api.acoustid.org/v2/lookup?client=8XaBELgH&meta=recordings+releasegroups+compress&duration=DURATION&fingerprint=FINGERPRINT
```

Com a finalidade de eliminar todas estas etapas, foi desenvolvido um algoritmo em *PHP* capaz de deixar a execução automática, ou seja, ele abrirá o arquivo, retirará as informações necessárias e executará a seguinte linha de comando:

```
# curl -G http://api.acoustid.org/v2/lookup?client=SUA_SENHA"&"meta=recordings"&"duration='.$teste.'"&"fingerprint='.$id.'
```

Onde *$teste* e *$id* são as variáveis retiradas do arquivo *codigo.txt*, desta maneira a busca por equivalências torna-se automática e elimina todas aquelas etapas manuais e que poderiam causar algum erro.

Após rodar esta linha, o resultado será impresso na tela. Existem inúmeras informações e, novamente, é necessário realizar um filtro na resposta para apenas visualizar os dados desejáveis.

4.2. Instalação e configuração do banco de dados

Em um sistema que faz uso de banco de dados é importante, desde o início, instalá-lo e configurá-lo no processador que o próprio ficará armazenado. No projeto em questão, o banco de dados foi armazenado em um *core* disponível no laboratório *lavisim* chamado de "*opencore*", de endereço externo 143.107.235.37. Isto foi feito para aliviar a pressão sobre o kit *Tiny*6410, pois o banco de dados é um arquivo crescente e certamente o computador está mais preparado, em questão de memória, do que o kit.

Para o sistema do banco de dados funcionar corretamente existem certos pacotes de devem ser instalados inicialmente. Como o site/acesso ao banco de dados foi elaborado todo em linguagem *PHP* o computador deve ter os seguintes pacotes instalados:

- **Apache2:** Servidor *WEB*, é compatível com o protocolo HTTP. Suas funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive que o usuário escreva seus próprios módulos.
- **Mysql-server:** Servidor *mysql*, foi desenvolvido originalmente para lidar com bancos de dados muito grandes de maneira muito mais rápida que as soluções existentes e tem sido usado em ambientes de produção de alta demanda por diversos anos de maneira bem sucedida.

- **Php5:** é uma linguagem interpretada livre, usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes, capazes de gerar conteúdo dinâmico na *World Wide Web*.
- **Php-mysql**: Permite a elaboração de páginas em *PHP* para gerenciar banco de dados.

A instalação destes pacotes foi feita com os mesmos comandos utilizados nos procedimentos anteriores. Desta forma, segue o comando para realizar a instalação destes pacotes (estes pacotes devem ser instalados tanto no kit quanto no *opencore*):

```
# sudo apt-get install php5-mysql php5 mysql-server-5 apache2
```

No momento em que o *mysql* estiver sendo instalado será requisita uma senha (Figura 15) para o usuário *root* (administrador) do *mysql*. Esta senha pode ser escolhida de acordo com a preferência do usuário. Logo em seguida aparecerá uma tela semelhante à da Figura 15 pedindo uma confirmação de senha.

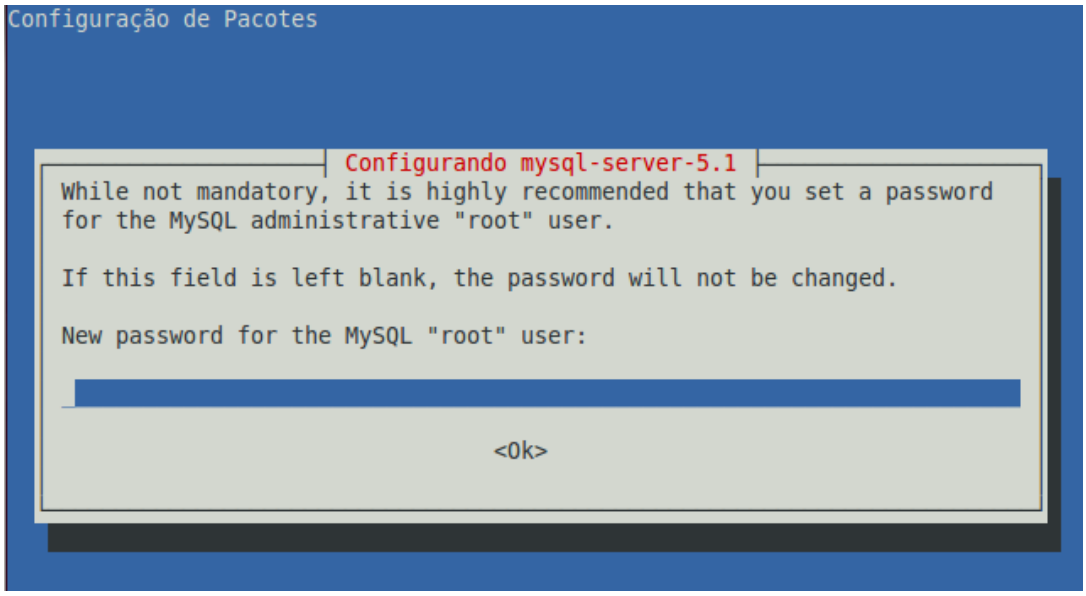


Figura 15: Janela para inserir a senha do mysql

Depois de concluída a instalação do *Mysql* é possível conferir se ocorreu tudo normalmente, bastando abrir o navegador e abrir o endereço http://localhost/. Se aparecer alguma mensagem do tipo "**It works!**" é porque não ocorreu erro durante toda a instalação e o *mysql* está sendo executado em seu computador.

Finalizada a instalação do *mysql* junto com os outros pacotes, tem-se que realizar a instalação da aplicação *WEB* utilizada para fazer as operações da administração do Sistema

de Gerenciamento de banco de dados *mysql* o *phpMyAdmin*. Sua instalação é realizada normalmente pelo comando abaixo:

```
# sudo apt-get install phpmyadmin
```

Durante o processo de instalação do *phpMyadmin*, aparecerão algumas telas e algumas irão requisitar informações para a configuração do sistema. Como se pode ver na Figura 16 irão aparecer algumas perguntas sobre a configuração do sistema, para todas, tecle ENTER, são configurações internas do sistema e não temos como alterar isto. Mais a frente a senha do *mysql* será requisitada, sendo assim digite a mesma senha de antes (Figura 17) e depois confirme. Realizado todo este procedimento pode-se averiguar a validade em http://localhost/phpmyadmin/ (Figura 18).

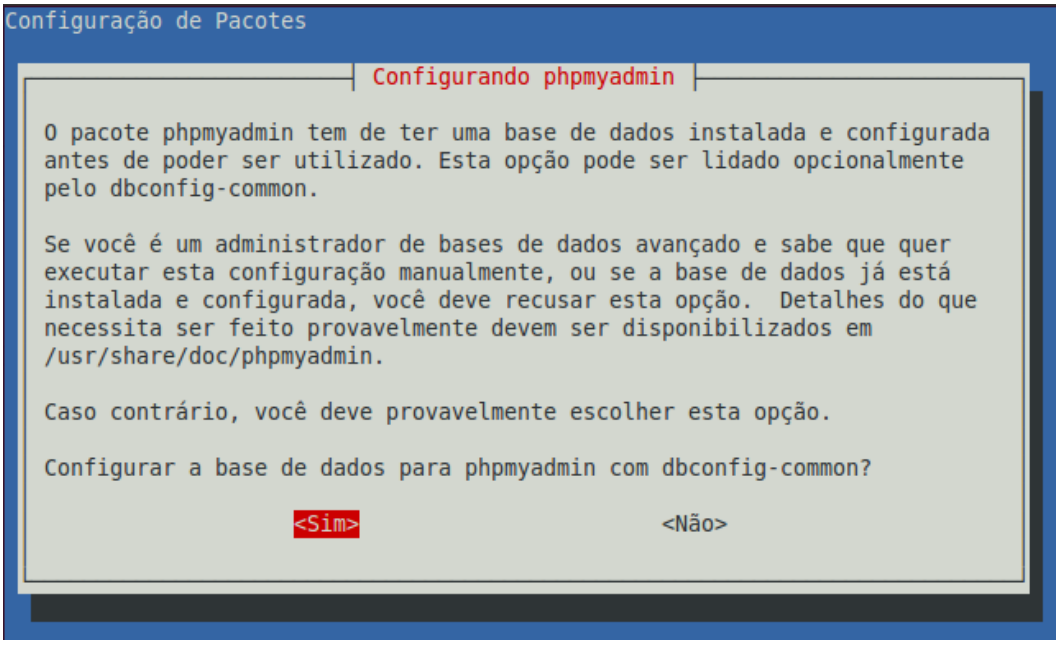


Figura 16: Tela de instalação - Configuração do sistema

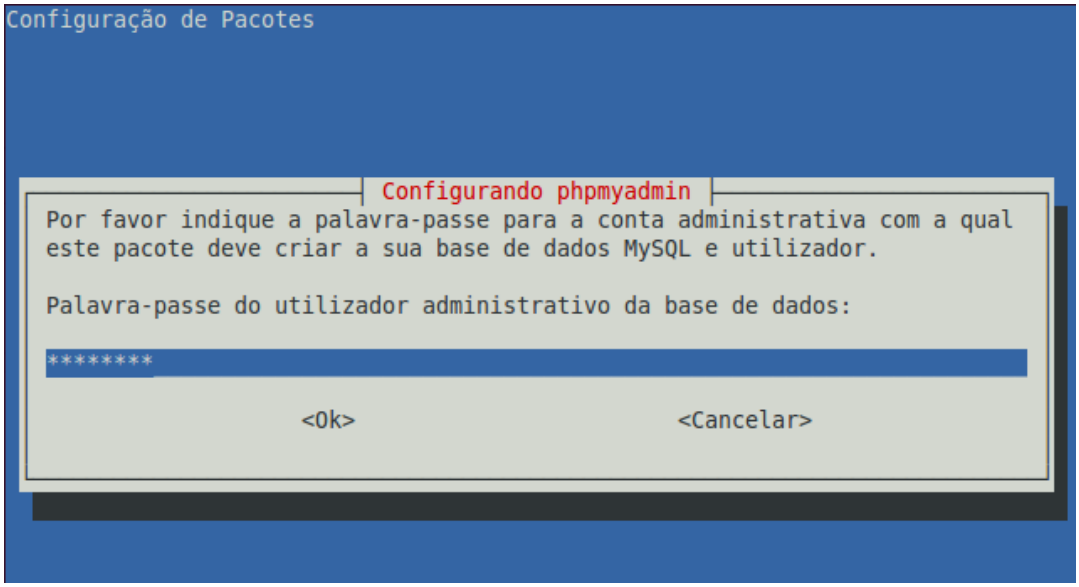


Figura 17: Tela para inserir senha do Mysql

Figura 18: Tela de início phpMyAdmin

Neste ponto, o projeto deparou-se com um problema, pois após todos os passos descritos ainda assim o sistema não funcionava, ocorriam erros de comunicação e leitura entre o kit e o *opencore,* mesmo com o conjunto *Mysql/phpMyAdmin* funcionando separadamente.

Após pesquisas e inúmeras tentativas, descobriu-se que os pacotes de *php* possuem em seus arquivos de configuração (*php.ini*) certas diretrizes que permitem ou não a liberação do sistema *mysql,* e como *default* do arquivo o *mysql* vem bloqueado. Para

solucionar isso basta abrir o arquivo *php.ini* e inserir no código a liberação do *mysql* (*extension=mysql.so*). Este procedimento pode ser feito seguindo o seguinte comando:

```
# nano /etc/php5/php.ini
```

O comando *nano* executa um papel de editor de texto, ou seja, a linha acima abrirá na tela o arquivo *php.ini*, que estará disponível para alterações. Quando abrir o arquivo, deve-se inserir a linha *extension=mysql.so*. Aparentemente a situação foi consertada, mas quando testado o sistema apresentou um erro de permissão, ou seja, o kit tentava acessar o banco de dados dentro do *opencore*, porém como não estava autorizado o sistema ficava travado.

Um meio de permitir o acesso foi estudado e aplicado, o meio consiste em entrar nas configurações do m*ysql* localizado no *opencore* e liberar o acesso para o IP interno do kit (10.235.0.138 com senha *lavisim*) diretamente por linha de comando. O procedimento passo a passo é evidenciado em seguida.

O primeiro passo é entrar nas configurações *mysql* do *opencore* através do comando:

```
# mysql –u root -p
```

Para prosseguir será requisitada a senha do usuário *root* do *mysql*, que é a mesma senha escolhida pelo usuário no início da instalação. A partir do ponto em que a senha for validada estaremos dentro do sistema *mysql* que está rodando dentro do *opencore*, assim os comandos devem ser usados exatamente na sequência mostrada abaixo. Uma ilustração de todo o processo pode ser vista na Figura 19:

```
mysql> use mysql
mysql> GRANT ALL ON *.* to root@'10.235.0.138' IDENTIFIED BY 'lavisim';
mysql> FLUSH PRIVILEGES
mysql> EXIT
```

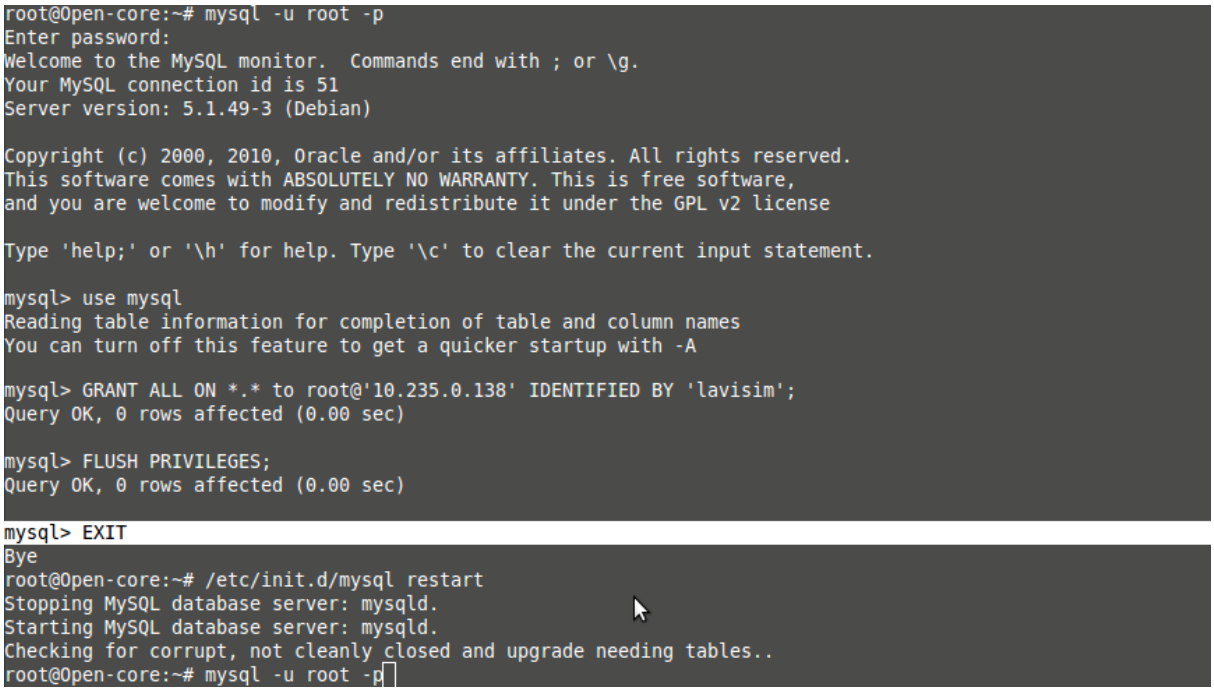


Figura 19: Execução dos comandos na janela de comando

Após ter feito a liberação de acesso para o kit *Tiny*6410, o sistema de acesso ao banco de dados funcionou perfeitamente.

4.3. Desenvolvimento do *website*

Como dito anteriormente, o site foi desenvolvido na linguagem *php*, as dúvidas sobre sintaxe e funções foram tiradas no GUIA PHP, e faz suporte para todo o desenvolvimento em relação aos algoritmos até o momento. Na Figura 5 tem-se uma imagem do *layout* da página principal do site, dela pode-se acessar 8 *links* diferentes os quais estão endereçados, cada um, com um sub-rotina diferente. Esta seção aborda o funcionamento de cada sub-rotina dentro do *website*. Todos os programas podem ser encontrados no endereço *opencore.eesc.usp.br* dentro da pasta “marcos”.

Mas antes de detalhar cada rotina, vale informar que todas estão ligadas com o sistema de *longin*/cadastro, ou seja, todos que entrarem no *site* só irão ter acesso ao seu conteúdo se estiverem realizado previamente o *login* e, consequentemente, cadastrados no *site.* Esta imposição serve como método de segurança, de fiscalização do conteúdo interno e eficiência do algoritmo.

- **Sistema de Cadastro de usuário**

Um sistema de cadastro pode ser desde um simples armazenamento de nome de usuário até um complexo preenchimento/armazenamento de inúmeras informações pessoais como endereço, telefone, nome de pais, e assim por diante. Como o objetivo de

inserir cadastro no projeto é ter controle de quem esta mandando informações para o banco de dados, não são necessária tantas informações e com isso foram criados campos para preenchimento de apenas nome completo, email, nome de usuário e senha, como pode ser visto na Figura 20.

Figura 20: Informações necessárias para o cadastro

Estas informações são enviadas para armazenamento no banco de dados, chamado de "*longin"*, e pode ser visualizado através da ferramenta online do *phpMyAdmin.* A lógica de armazenamento pode ser vista no fluxograma da Figura 21.

Toda vez que alguém vai fazer cadastro no site, é verificado o seu nome de usuário e email. Caso um dos campos já exista no banco de dados uma mensagem irá informar o usuário que o cadastro não foi concluído, pois o usuário já existe e/ou email já foi cadastrado.

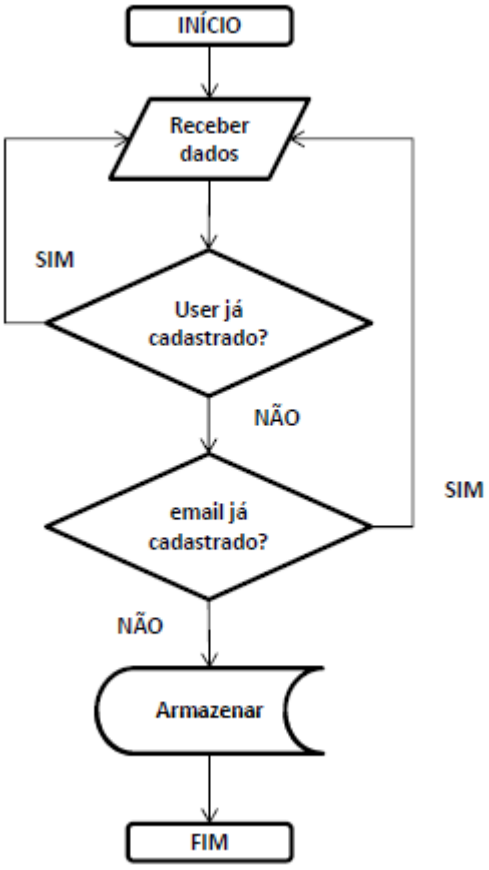


Figura 21: Fluxograma Lógica - do cadastro

- **Campo de *login***

Este é o *link* que libera o acesso para todas as outras páginas do site. Sem estar "logado" é impossível visualizar o conteúdo das outras rotinas.

Primeiramente os dados são enviados e através de uma comparação com os dados existentes no banco de dados, gerado pelo cadastro, é possível saber se os dados inseridos estão de acordo com algum usuário do site. Em caso de não ocorrer equivalência tem-se duas opções, o nome de usuário/senha estão incorretos ou não existe este cadastro, das duas maneiras um aviso aparece na tela informando o navegador.

Estando todos os dados (senha e nome de usuário) de acordo, o sistema inicia a sessão e libera o acesso para as outras páginas. A função que inicia a sessão em *php*, que foi usada no algoritmo, é ***$_SESSION['auth']=1;*** e para encerrar a sessão é feito a variável receber o valor negativo representado por "0" (***$_SESSION['auth']=0;***).

- **Alterar senha**

Esta página segue a lógica das descritas anteriormente, pois o ato de trocar a senha é simplesmente realizar uma busca no banco de dados, conferir se a senha digitada como senha atual é equivalente à armazenada, e juntamente a isto conferir se os dois campos com a senha atual são iguais entre si. Se as duas condições forem verdadeiras, o banco de dados será atualizado, ou seja, será inserida no campo da senha antiga a senha atual digitada pelo usuário.

A função que permite alterar dados dentro de um banco de dados é denominada ***UPDATE*** e deve ser usada com as seguintes informações:

```
UPDATE table SET campo = 'valor' WHERE condição;
```

A informação contida na variável "*table",* deve ser completada com o nome da tabela do banco de dados onde estão armazenados os dados cadastrais, neste caso ***usr***. A variável a ser carregado em "*campo"* é a coluna, referente à tabela informada, que contém a informação a alterada, neste caso "***senha"***. Por último o campo "*condição*" que irá localizar a linha correta para fazer alteração de senha (caso este campo seja esquecido todas linhas serão alteradas), neste caso ***"usuario = $usr"*** que faz com que a alteração seja feita apenas na linha que existe o usuário que está "logado".

A variável **"$usr"**, faz uma chamada o nome do usuário que deu início a seção, ou seja, o nome do usuário "logado". E assim o sistema só alterará a senha da linha que possuir na coluna "usuario" o nome de usuário que está requisitando a alteração de senha.

- **Administrador**

O campo administrador representa uma área onde o desenvolvedor ou o responsável pela administração do site pode realizar algumas tarefas com o intuito de organizar o site e, é claro, aumentar o nível de segurança.

Para começar, se qualquer outro usuário tentar entrar nesta página será bloqueado e redirecionado para a página principal do site. Dentro desta página estão alocadas mais três funções as quais são descritas abaixo.

Informações dos usuários

É uma página que mostra todas as informações sobre os usuários do site. Quando necessário, o administrador pode ver dados dos usuários, como emails e nomes de usuário. Todas estas informações são dispostas em forma de tabela.

Excluir usuários

Quando notada alguma infração por parte de algum usuário, como por exemplo, enviar alguma música com nome errado, se o administrador julgar que a pessoa deve ser excluída do sistema ele o pode fazer. Para isto apenas necessita do nome do usuário (que pode ser consultado no item *informações dos usuários*).

A função que disponibiliza a exclusão de alguma informação do banco de dados é representada pela sintaxe ***DELETE FROM*** e usada da seguinte maneira:

```
$sql = "DELETE FROM dados WHERE condição"
```

Onde "*dados"* representa o nome a tabela, quês estão contidas as informações dos usuários dentro do banco de dados, neste caso ***usr***, e a condição deve ser preenchida com o nome do usuário digitado pelo administrador. Se esta opção não for digitada serão deletadas todas as linhas do banco de dados. Para receber o nome do usuário dentro do sistema, usa-se a função em *php* ***$_POST['login'].***

Dados do Administrador

Em caso de uma mudança de administrador, isto pode ser feito nesta página informando nome de usuário e email do novo administrador. É claro que esta alteração só pode ser realizada pelo atual administrador. Neste caso é usada a mesma sintaxe para alteração de informações dentro do banco de dados (***UPDATE***).

As informações do administrador estão presentes no mesmo banco de dados, mas em tabela diferente para evitar alguma pane no sistema e confusões.

- **Dados**

Esta página é usada para exibir todas as músicas que existem no banco de dados, para o usuário fazer uma simples pesquisa ou adicionar alguma que não esteja na lista. Na tabela é mostrado o nome da música, o artista e qual algoritmo que foi usado em sua codificação. Além disto, é possível, com um clique no nome da música, ouvi-la diretamente do *youtube*.

A exibição na tela é feita de forma que todas as colunas tenham a mesma largura e todas as linhas possuam a mesma altura. A linha de programa que executa o redirecionamento para o *youtube* é:

```
<tdwidth=”25%”><center><a href=endereço><font>nome</font><center></a></td>
```

Onde o campo endereço deve ser preenchido com o *site* a ser redirecionado, porém deve-se inserir o nome da música e artista para abrir o *site* corretamente e isto é feito seguindo a lógica abaixo. As variáveis ***$escrever[‘musica’]*** e ***$escrever[‘artista’]*** representam o nome da música(que também deve ser inserida em *nome*) e o nome do artista, respectivamente:

```
http://www.youtube.com/results?search_query='.$escrever['musica'].'
'.$escrever['artista'].'&oq=paradise&gs_l=youtube.3.0.35i39j019.565.1689.0
.3264.8.8.0.0.0.0.189.807.4j4.8.0...0.0...1ac.1.wTo0HroqSoo"
target="_blank
```

- **Submeter**

O *link submeter* exerce um papel muito importante dentro do propósito do *site*, sendo que é responsável pelo armazenamento de novas músicas juntamente com todas suas informações (impressão digital, nome, artista, algoritmo usado), e assim quanto maior for o número de músicas armazenadas maior será a probabilidade de equivalência, além de tornar a busca mais rápida.

A página tem um sistema que evita falhas, e só receberá os dados se todos forem enviados corretamente, evitando assim que no banco de dados existam músicas sem artista ou sem a respectiva ID, e também recusa o envio quando o nome da música já se encontrar armazenada (evitando duplicatas).

O processo de enviar informações para o banco de dados é liderado pelo comando em *php* com sintaxe ***INSERT INTO***, deste modo recebemos as informações pela sintaxe ***$_POST['variable']*** e são enviadas para o banco de dados, referente aos dados musicais, pelo comando ***INSERT INTO***, que é usado da seguinte maneira:

```
$sql = "INSERT INTO dados1 (incremento, usuario, artista, musica,
algoritmo, id) VALUES ('', 'marcos', '$banda', '$musica', '$alg0',
'$id_sub')";
```

Já o arquivo de áudio, enviado juntamente com o questionário, Figura 22, é tratado de uma maneira diferente. O arquivo é recebido pela variável ***$_FILE['uploadfile']['name']*** e enviado diretamente para o diretório de trabalho onde estão localizados os arquivos de execução dos programas *echoprint* e *chromaprin.,* Assim quando esses programas forem acionados não precisam procurar o áudio em outros diretórios. Lembrando que o software foi programado para procurar o arquivo *submeter.mp3* ou *submeter.wav*, se o áudio não for enviado com esta específica sintaxe, o sistema será impossibilitado de executar a codificação do áudio.

Antes de enviar todas as informações para o banco de dados o programa verifica qual algoritmo foi escolhido pelo usuário, e a partir desta variável executa o *echoprint* ou *chromaprint* para gerar a impressão digital, que também será enviada para o banco de dados.

Após executado o algoritmo de codificação, tem-se todas as informações prontas para envio, nome da música, nome do artista, o algoritmo usado, a impressão digital e também o usuário que está emitindo estas informações. De uma maneira mais intuitiva pode-se ver o funcionamento desta página no fluxograma da Figura 23.

Figura 22: Questionário para enviar dados da música

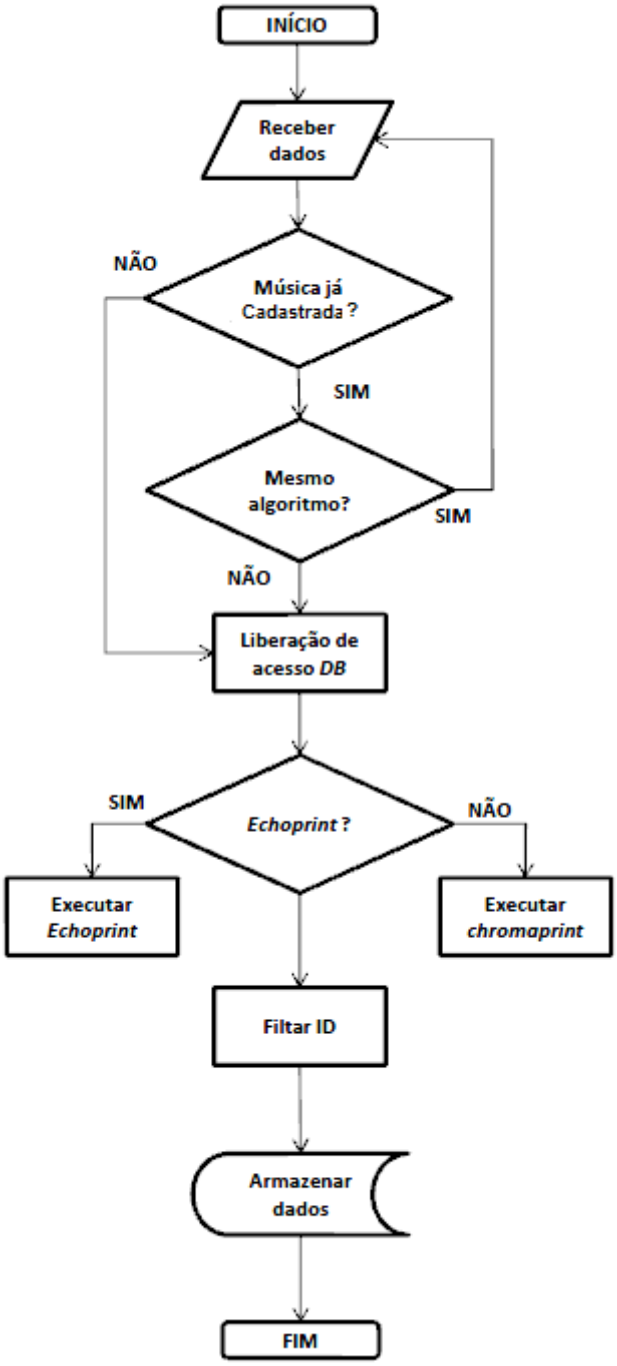


Figura 23: Fluxograma do funcionamento da página submeter

- **Áudio FP**

Esta página representa o sistema mais importante do projeto, pois é responsável por dar suporte aos algoritmos de codificação (*echoprint* e *chromaprint*). Em contrapartida à sua importância, a página não é muito complexa, conta com 3 ícones e com a descrição do projeto (Figura 24).

Os algoritmos, para funcionarem como esperado, necessitam de algumas variáveis de entrada e estas variáveis mudam de acordo com a escolha do algoritmo a ser usado. Logo, foi necessário o desenvolvimento de *links* separados para a execução de cada algoritmo. A única informação que é usada em ambos procedimentos é o arquivo de áudio, que por este motivo esta disposto em um *link* à parte dos algoritmos.

Antes da escolha entre os *softwares* de codificação, tem-se que enviar o arquivo que será codificado clicando no botão ***Recorder***. Existe uma restrição em relação ao nome da música, e ela deve ser enviada sempre com o nome de *musica.wav* ou *musica.mp3* dependendo de sua extensão.

Com o arquivo armazenado no seu devido lugar, entra-se na etapa de escolha de *software*, e neste caso existem duas escolhas: *echoprint* e o *chromaprint*. Independentemente da escolha o resultado aparecerá na tela, nome da música e nome do artista, logo, para o usuário, a escolha entre os métodos não é aspecto que irá influenciar. Porém, internamente, os algoritmos elaborados para a execução dos diferentes métodos são diferentes. Podem-se considerar os passos semelhantes que podem ser acompanhados no fluxograma da Figura 25.

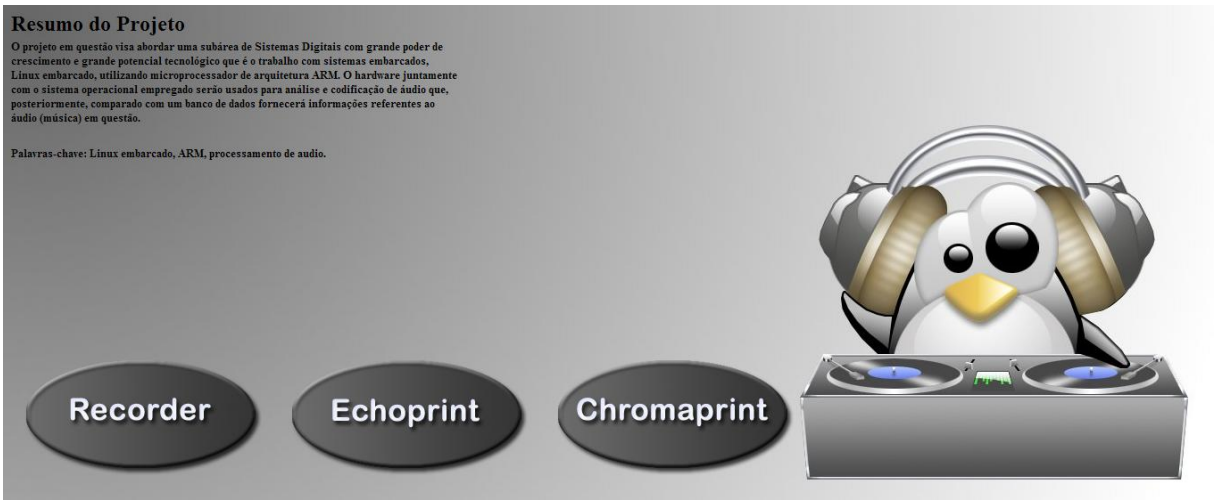


Figura 24: Layout da página Áudio FP

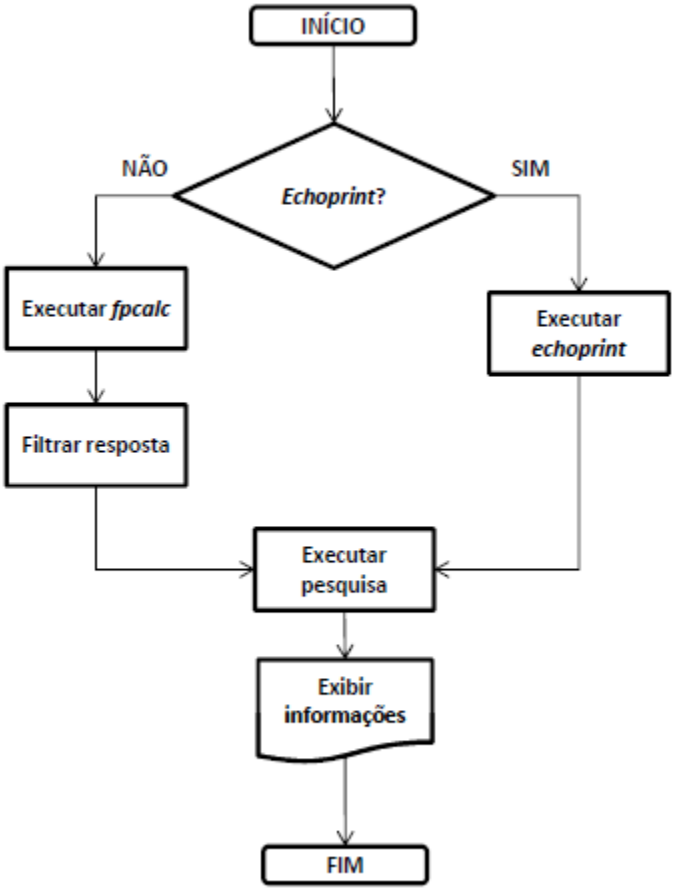


Figura 25: Fluxograma - Página Áudio FP

4.4. Configurações adicionais

O sistema de recebimento de áudio apresentou um pequeno problema. O sistema não aceitava a transferência de arquivos superiores a 2 MB, e a maioria dos arquivos de áudio superam esta faixa, sendo que uma música normal com aproximadamente 3 min apresenta uma média de 6 MB, sendo assim, o sistema não poderia permanecer com esta falha.

Feitas algumas pesquisas, descobriu-se que o pacote *php* vem definido com o limite máximo de transferência de arquivo para 2 MB, justamente a falha notada. Para reparar esta divergência deve-se alterar, novamente, algumas linhas de programa do arquivo *php.ini*, encontrado no diretório */etc/php5/php.ini*.

As linhas que devem ser alterados são:

- ***upload_max_filesize:*** Tamanho máximo do arquivo que será transmitido para o kit (foi alterado para 100 MB).
- ***post_max_size:*** Responsável também pelo tamanho do arquivo em transferência (foi alterado para 100 MB).

- ***max_input_time:*** Limite de máximo de tempo para recepção do arquivo (foi alterado para 300 segundo).
- ***max_executation_time:*** Limite máximo de tempo para a execução da transferência por completo (alterado para 300 segundos).
- ***memory_limit:*** Limite de memória, é importante deixar algum valor fixo, caso os outros limites falhem (alterado para 100 MB).

E a alteração deve ser feita como descrito anteriormente, com o comando *nano* para abrir o arquivo para edição, e alterá-lo manualmente seguindo cada linha descrita:

```
# nano /etc/php5/php.ini
```

Feita estas alterações, os arquivos que não estavam sendo enviados passaram a ser, e foi feito um teste com um arquivo de extensão *wav*, que apresentava 44 MB e a transferência ocorreu normalmente.

5. Resultados e análises

Este é o momento de avaliar cada etapa do projeto, desde os algoritmos de reconhecimento até o sistema de segurança/*login* desenvolvido para liberar acesso ao projeto via *web*. Foram realizados alguns testes a fim de avaliar a funcionalidade do sistema, bem como a sua eficiência.

Começando pelos algoritmos, é claro que sempre o objetivo é obter maior velocidade de resposta com uma alta porcentagem de acerto, e aqui também não é diferente, quanto mais rápido e mais eficiente o algoritmo é melhor, logo, foram feitos testes para analisar a velocidade de processamento de cada algoritmo com diferentes músicas. A Tabela 2 representa o tempo que cada algoritmo levou para codificar a música (teste feito via linha de comando fora do *web site*).

Tabela 2: Tempo de resposta de cada algoritmo

Nome	Execução pelo *Echoprint (s)*	Execução pelo *Chromaprint (s)*
Feeling This	55	48
Times Like These	59	51
Feel So Close	62	52
Glad You Came	76	52
Angel	64	50
Tempo médio	63,2	50,6

A Tabela 2 evidencia que o *chromaprint* apresenta uma ligeira vantagem em termos de velocidade de processamento perante o *echoprint.* Essa diferença pode ser bem notada no Gráfico 1, onde é perceptível que em todas as músicas a curva do tempo de execução do *echoprint* fica sempre em cima da curva do *chormaprint*. A diferença esta em torno de 10 segundos.

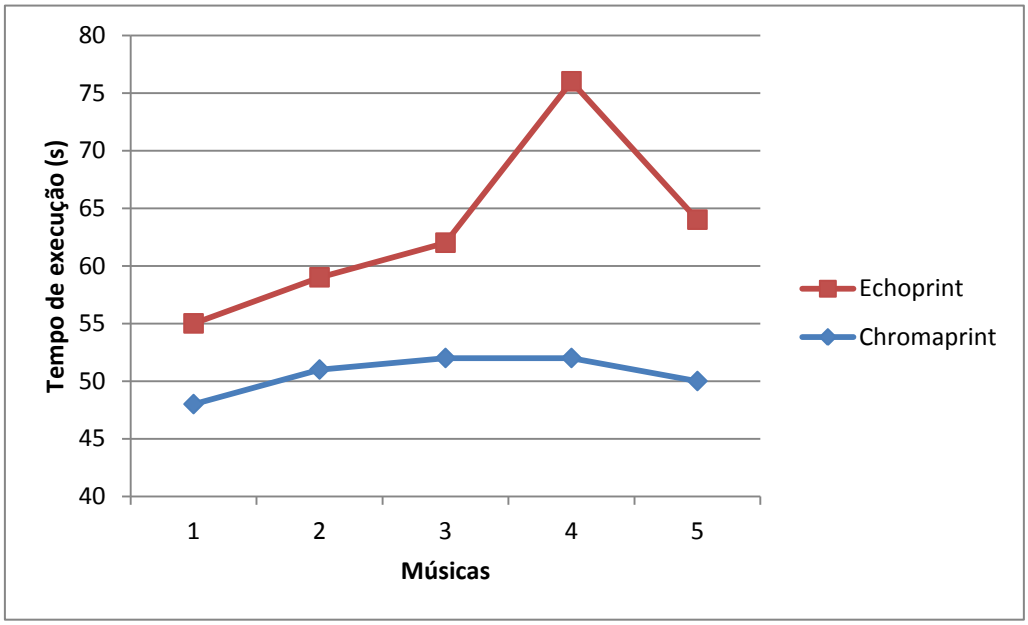


Gráfico 1: Tempo de execução dos algoritmos x músicas

Porém, a geração do código é apenas uma etapa do processo de reconhecimento de áudio, ainda resta a análise da velocidade de acesso ao banco de dados e por último, e o mais importante o nível de eficácia de cada sistema, ou seja, qual sua porcentagem de acerto de nome das músicas.

Com as mesmas músicas testadas anteriormente fez-se o teste de acesso/resposta do banco de dados: foram enviados seus códigos e outras informações necessárias para o servidor e analisado o tempo que levou para se obter uma resposta. Os resultados obtidos estão dispostos na Tabela 3.

Tabela 3: Tempo de resposta do servidor de cada algoritmo

Nome	Resposta do servidor *Echoprint (s)*	Resposta do servidor *Chromaprint (s)*
Feeling This	7	2
Times Like These	7	3
Feel So Close	7	3
Glad You Came	7	2
Angel	10	3
Tempo médio	7,6	2,6

Mais uma vez o *chromaprint* teve certa vantagem, e apresentou um desempenho melhor. A diferença entre os dois está em cerca de 5 segundos em média e para ver de uma maneira mais intuitiva, o Gráfico 2 apresenta duas curvas representando a resposta do servidor do *echoprint* e do *chromaprint.*

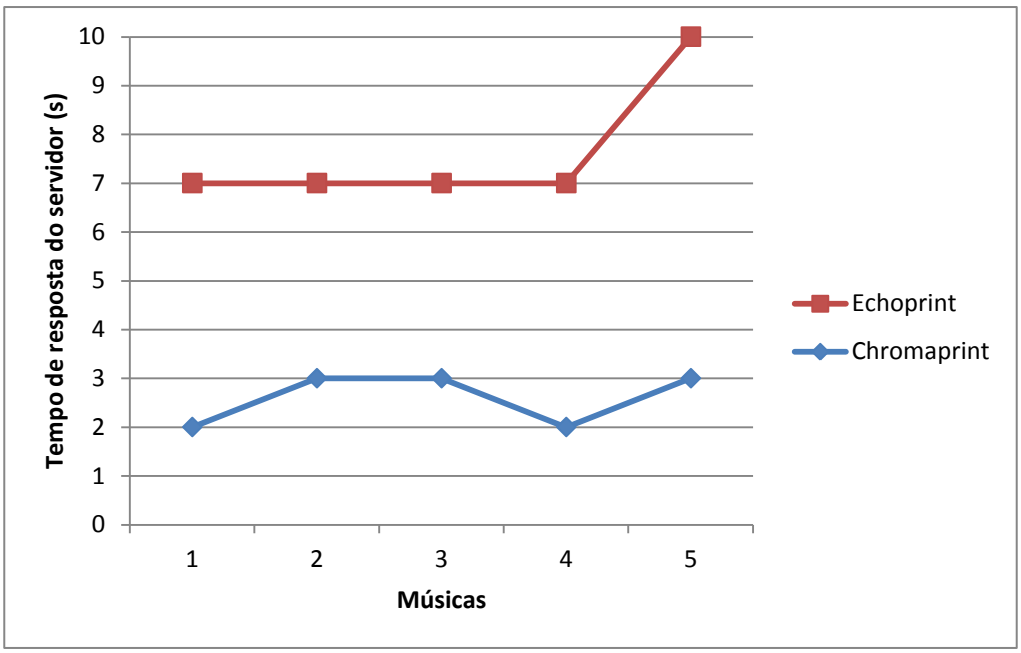


Gráfico 2: Tempo de resposta do servidor x músicas

Não é proveitoso ter um rápido processamento com um ágil acesso ao banco de dados e retornar uma resposta incorreta, por isso que a próxima análise é de extrema importância. Com estas mesmas músicas, que foram testadas, fez-se análise da resposta obtida.

Depois de realizadas as chamadas dos servidores, notou-se que os dois sistemas falharam em uma música (músicas diferentes), porém a falha não foi referente ao algoritmo e sim devido a inexistências destas músicas no banco de dados. É neste ponto que surge uma das funções do banco de dados no *website* do projeto, pois aumenta a probabilidade de acerto com o aumento de músicas armazenadas.

Baseando-se nos resultados mostrados, é evidente que o *chromaprint* leva vantagem em termo de velocidade de processamento, apresenta respostas mais rápidas tanto na codificação do áudio como no acesso às informações do banco de dados. Já em termos de respostas corretas ambos apresentaram o mesmo desempenho 75 %. É claro que seria necessário fazer testes com muito mais músicas para poder ter uma porcentagem mais precisa.

Um ponto positivo é que falhas foram devidas à ausência de código correspondente no banco de dados e não por alguma equivalência errônea.

Com toda esta análise feita com os algoritmos, pode-se avaliar o efeito que o site causa neste tempo de processamento e eficiência. O tempo de resposta certamente será acrescido, pois no procedimento de automatizar a busca foram introduzidos filtros e manipulações de arquivo de extensão *.txt*.

Os dados obtidos estão dispostos na Tabela 4. Estes valores representam o tempo de execução total, ou seja, tempo de resposta do algoritmo mais tempo de resposta do servidor.

Tabela 4: Tempo de resposta do *site*

Nome	Resposta do *website Echoprint (s)*	Resposta do *website Chromaprint (s)*
Feeling This	66	58
Times Like These	65	58
Feel So Close	60	58
Glad You Came	62	66
Angel	78	66

Para poder observar melhor o efeito do *site* sobre o tempo de resposta, foram elaborados os Gráfico 3 e Gráfico 4, fazendo um comparativo entre antes e depois de cada algoritmo, *echoprint* e *chromaprint* respectivamente.

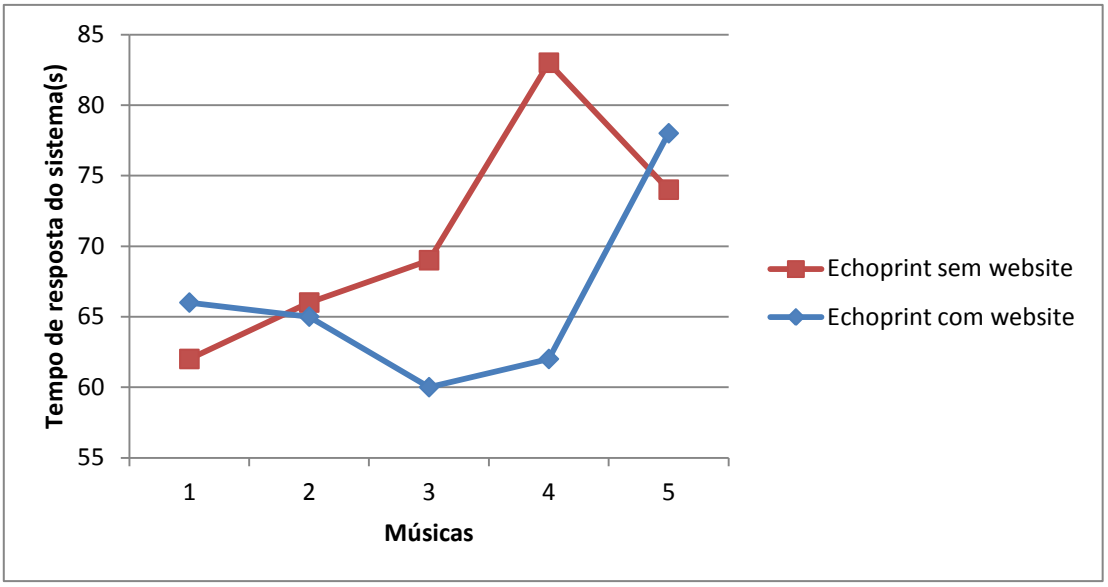


Gráfico 3: Tempo de resposta do sistema x Músicas (*echoprint*)

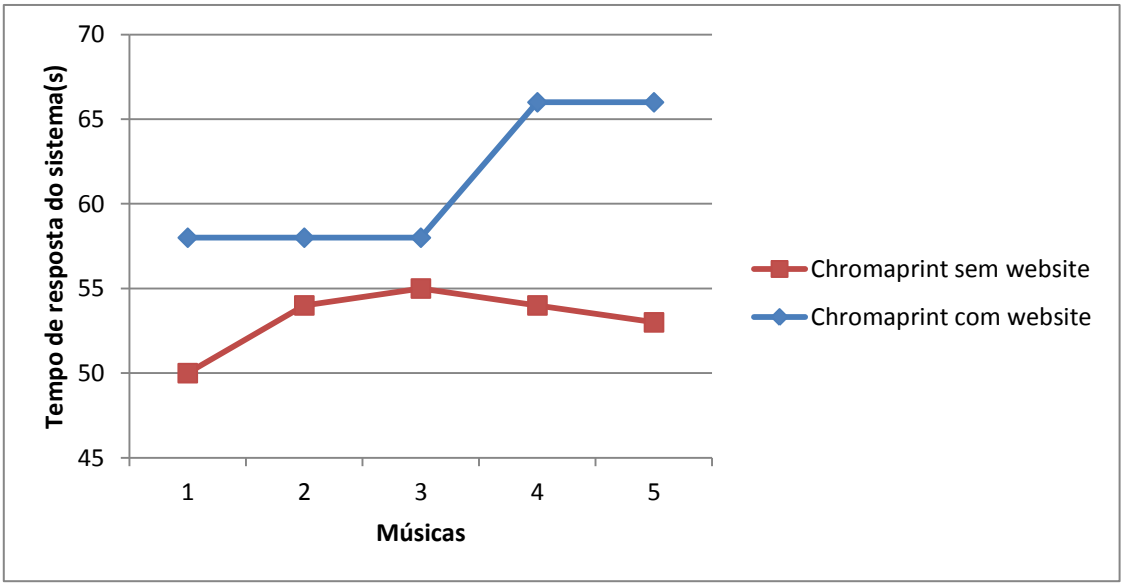


Gráfico 4: Tempo de resposta do sistema x Músicas (*chromaprint*)

É visível no Gráfico 4 que o *website* acrescentou alguns segundos no seu tempo de resposta, porém agora o sistema é todo automatizado e com uma interface, representada pelo *site*. Já no Gráfico 3 é notório que em três das cinco músicas testadas apresentaram um tempo de resposta menor, e isso se deve ao fato de que estas três músicas apresentarem seus códigos cadastrados no banco de dados do projeto.

O Gráfico 3 comprova que um dos objetivos, pelo qual o banco de dados foi desenvolvido, funciona como o esperado diminuindo o tempo de resposta e assim deixando o sistema mais rápido que o normal, porém isto apenas é válido para o *echoprint*.

No caso do *chromaprint*, o *site* acaba elevando seu tempo de resposta. Isto ocorre pois seu tempo de busca no servidor era muito baixo, cerca de 2 segundo, e o sistema de busca pelo banco de dados, realizado no *website*, leva alguns segundos a mais pois é necessário realizar a conexão e depois a busca por toda sua extensão.

O sistema de segurança desenvolvido funciona de acordo com o esperado, onde todo o fluxo de informação do *site* é controlado pelo administrador, que pelo próprio sistema, tem a ferramenta de exclusão de membros e também exclusão de informações inadequadas/errôneas.

O procedimento de bloqueio de conteúdo para pessoas que não estejam "logadas" está em pleno funcionamento, ou seja, se tentar acessar qualquer página no *site* sem estar "logado", o conteúdo não será liberado e uma mensagem irá aparecer na tela, indicando para o membro realizar o *login* ou se cadastrar no site, caso ainda não o tenha feito.

Por último, uma análise da submissão de músicas também é plausível, pois desta maneira consegue-se retirar informações que revelam o tempo que o sistema leva para adicionar dados dentro do banco de dados.

Quando o usuário submete uma música para armazenamento no banco de dados, praticamente todas as rotinas do *website* são acionadas, pois é necessário receber o arquivo de áudio, analisar o banco de dados com o intuito de não enviar músicas repetidas, acionar um dos *softwares* escolhido, realizar a codificação e por último enviar as informações armazenadas para o banco de dados. Devido número elevado de operações sequenciais, sem dúvida, esta página é que levará mais tempo para dar alguma resposta. Na Tabela 5 pode-se observar o tempo desta operação.

Tabela 5: Tempo de resposta da submissão de dados

Nome	Resposta da submissão *Echoprint (s)*	Resposta da submissão *Chromaprint (s)*
Teenage Dream	129	119
Thank You For Loving Me	123	100
I Started a Joke	84	75
Mundo Animal	89	84
Quebra Mar	75	75
Too Much	158	146

Como esperado, o tempo de espera é elevado, mas é condizente, pois se juntarmos todos os tempos de execução anteriores teremos um número aproximado à média dos valores da Tabela 5. Outro fato que pode ser notado é que o fator que mais interfere na diferença entre os valores mostrados na Tabela 5 é o tamanho do arquivo, ou seja, quanto maior o arquivo maior o tempo de seu envio e, consequentemente, maior o tempo de

resposta de submissão dos dados. A música que levou maior tempo foi *Too Much*, justamente a que possui maior espaço de memória, 10,0 MB.

Na Figura 26, tem uma imagem do banco de dados depois da submissão destes arquivos, comprovando que a comunicação *website* e banco de dados está ocorrendo sem falhas. Note que devido ao tamanho as impressões digitais não são mostradas completamente, mas basta expandir a coluna *id* para analisá-las por completo.

	incremento	usuario	artista	musica	algoritmo	id
	1	marcos	Blink 182	I miss you	Chromaprint	AQADtMmUUEmURIkU_MOxdeg54pscNCoL7UF_9CmF7xh3-CZ-nG...
	3	marcos	Blink 182	I miss you	Echoprint	eJy1mwmOozmuhK8kiVqPo4W6_xHeRyXQninATDzUNDATXZW29U...
	4	marcos	Calvin Harris	I feel so close	Chromaprint	AQADtJySJFISZXjzBe6PPCeqBG9oQI-OCy7q4OiP5sbTQ6HK5X...
	5	marcos	Calvin Harris	I feel so close	Echoprint	eJzNnGuOlzmuhbek92M5eu5_CfORLmTUGNdMDDyNO_2DqLQiJl...
	6	marcos	Foo Fighters	Times Like These	Chromaprint	AQADtJESaUnWKAr86YCeHsfRfDfelK9V1D6u4zCVGr8sPJrgS8...
	8	marcos	Foo Fighters	Times Like These	Echoprint	eJzFmwmOJKcShq8EAQTEcVjvf4T3BS25bUtF66k1smSFZ2rJhF...
	9	marcos	Blink 182	Always	Chromaprint	AQADtKGSLVkUWl1K3Dza5Kibwx_0HxWHc0ftB-aK8OOQCw9qEd...
	10	marcos	Foo Fighters	Everlong	Echoprint	eJzNm1IyJCGTrbfEPCwHcNj_Eu53UFvr7zJL9JDW1vfFVZVDBP...
	11	marcos	Foo Fighters	My Hero	Chromaprint	AQADtNKeXlmSIL9EnIHJ48wk7M7xBM2eY3pi9IKlKugf9PBH1A...
	12	marcos	Aerosmith	Crazy	Echoprint	eJy1m2mOKzmuhbckUtS0HI37X8L76Hqo230BM38kuoA-hXbaER...
	13	marcos	Blink 182	Feeling This	Chromaprint	AQADtJGUSFKSKIoUhM-O8CnOGv6ENzv4Czp--MmGHN_B9bgE88...
	15	marcos	Fernando e Sorocaba	A verdade	Echoprint	eJzFm2uOlzmuhbekF0VpOZlo7X8J96MbuNITgJk_CoNpdJ-uct...
	16	marcos	Foo Fighters	The best off you	Chromaprint	AQADtFKinEoUwXwcCf2D_jBjyijXCD_6o6kaSrieDc2DEuNxzs...
	17	marcos	The Wanted	Glad you Came	Echoprint	eJy1m1IyZLlyRLeEOYDlYNz_EnQc9cHXbXZBM5W19OSSksmbQC...
	18	marcos	Foo Fighters	Everlong	Chromaprint	AQADtGm0KFkkFb1hHv7ReYGO_jBzEip8HmmP6_hxHv004nhROz...
	19	marcos	Katy Perry	Teenage Dream	Chromaprint	AQADtEmSRluSJloOAx9hB-LxEduLfJOA-8WX4zB31EqQH2clfQ...
	20	marcos	Katy Perry	Teenage Dream	Echoprint	eJzFm2mS3EiuhK8UC2I7DmK7_xHeh6g21bRsEmU2em39o72IZJ...
	21	marcos	Bon Jovi	Thank You For Loving Me	Chromaprint	AQADtKoiMWqK41Q3XA-aJ6IRPrqhHzkuPug1-Mtx6fiR34WOE-...
	22	marcos	Bon Jovi	Thank You For Loving Me	Echoprint	eJyIm1myMzeuradEgv1wSBCc_xDuB1Xc8qmKSOyHssPLtppUkg...
	23	marcos	Bee Gees	I Started a Joke	Chromaprint	AQADtFL0SfFQMT-uD0duXCOe42eRZ2gXbvgHV9PxWKilMdqRPX...
	24	marcos	Bee Gees	I Started a Joke	Echoprint	eJy1m2tuLLuthackUalew6Fe8x9CPnaA-GIDTf8wboCsnLTdVS...
	25	marcos	Mamonas Assasinas	Mundo Animal	Chromaprint	AQADtIlyRdKSBVNWPUE-gdVKjPnxa6hFCs0_zDza7PCUbRLR5C...
	26	marcos	Mamonas Assasinas	Mundo Animal	Echoprint	eJzVm1IyJLmORbfEmeByOlD7X0lfUM9K1WkmxIcs7XV_JLJSEe...
	27	marcos	CBJR	Quebra Mar	Chromaprint	AQADgk8p4ZyF0Bf2HGYe6DjxDFT7YjtaBg--Yw-qPld5tB9q6Y...
	28	marcos	CBJR	Quebra Mar	Echoprint	eJy1m2luLDcShK9EJrfkcbje_wjzhQzMGxso6ocxsBGQW-ouVj...
	29	marcos	Dave Matthews Band	Too Much	Chromaprint	AQADtFOUqEsShUHv43kQdjcSJktFoelPNDd6xkP0C5rPoWJy3J...
	30	marcos	Dave Matthews Band	Too Much	Echoprint	eJzVm2duJbmShbdEz4jI0O5_CfOdW8DTmwJE_RAKg0Gjo9XXZJ...

Marcar todos / Desmarcar todos *Com marcados:*

Figura 26: Banco de dados como as músicas enviadas para teste

6. Considerações Finais

Ao final do trabalho foi possível concluir que o projeto funcionou conforme o planejado, apresentando-se como uma ferramenta de reconhecimento de áudio, que além de propiciar o aumento do nível de acerto dos algoritmos em função da disponibilidade de um banco de dados local, o que aumenta o acervo de músicas para realizar a comparação, em muitos casos também diminuiu seu tempo de busca. Além destes benefícios, o sistema está disponível para qualquer usuário, bastando-se fazer o cadastro na página inicial do projeto, pois se utiliza das vantagens da disponibilização via *Web*.

Sobre a área de sistemas embarcados, pode-se concluir que continua em crescimento, mas ainda é possível constatar a ausência de concentração de assuntos pertinentes e bem documentados, o que pode inicialmente dificultar o trabalho. Porém, como existem inúmeros desenvolvedores pelo mundo, existem também muitos fóruns de discussões, onde todos compartilham informações importantes, e é destes fóruns que muitas vezes vem as informações que auxiliam na solução dos problemas que se apresentam no decorrer do desenvolvimento de projetos, como alguns aqui apresentados.

Referências Bibliográficas

FÓRUM brasileiro de usuários Linux. Disponível em: <http://www.brlinux.net>. Acesso em: 31 jul. 2012.

GUIA taglib. Disponível em: <http://taglib.github.com/>. Acesso em: 13 set. 2012.

BASE DE DADOS do algortimo *chromaprint* usado no desenvolvimento do projeto. Disponível em: <http://www.acoustid.org/chromaprint>. Acesso em: 20 ago. 2012

BASE DE DADOS do algortimo echo*print* usado no desenvolvimento do projeto. Disponível em: <http://www.echoprint.me>.Acesso em: 20 ago. 2012.

ELLIS, D.; WHITMAN, B.; PORTER, A. Echoprint – an open music identification service, ismir 2011. Disponível em <http://www.echoprint.me>.Acesso em: 30 ago. 2012.

BELLO, J.; DAUDET, L.; ABDALLAH, S.; DUXBURY, C. A tutorial on Onset Detection in Music Signals, IEEE Transactions on speech and audio processing, 2005. Disponível em <http://ieeexplore.ieee.org>. Acesso em: 05 set.2012.

MANUAL FriendlyArm Tiny6410. Disponível em <http://friendlyarm.net>. Acesso em 20 ju. 2012.

LEITÃO, G. Separação de fontes de sinais de áudio. 2009. Pós-Graduação em Engenharia elétrica e computação – PPGEE, Universidade Federal do Rio Grande do Norte, Natal. 2009

GUIA audio fingerprint: MusicBrainz: The open music encyclopedia. Disponível em: <http://musicbrainz.org>. Acesso em 20 ago.2012

HAITSMA, J.; KALKER, T. A Highly Robust Audio Fingerprinting System – Philips Research, 2002.

CHENG Y., Music Database Retrieval Based on Spectral Similarity, International Symposium on Music Information Retrieval (ISMIR) 2001, Bloomington, USA. October 2001

GUIA ffmpeg. Disponível em: < http://ffmpeg.org/>. Acesso em 13 set. 2012.

GUIA libboost. Disponível em: <http://www.boost.org/>. Acesso em: 13 set. 2012

ELLIS, D.; WHITMAN, B.; JEHAN, T.; LAMERE, P. The Echo Nest musical fingerprint. In Proceedings of the 2010 International Symposium on Music Information Retrieval, 2010.

BARTSCH, M.; WAKEFIELD,G.Audio Thumbnailing of Popular Music Using Chroma-Based Representations, 2005. Disponível em: <http://ieeexplore.ieee.org>. Acesso em: 05 set.2012.

GUIA zlib. Disponível em: < http://www.zlib.net/>. Acesso em 13 set. 2012.

GUIA Chroma features, 2004. Disponível em: <http://abrosa.ee.columbia.edu/matlab/chroma-ansyn> . Acesso em 30 out.2012.

KE, Y.; HOIEM, D.; SUKTHANKAR, R. Computer Vision for Music Identification, 2005. Disponível em: <http://www.cs.cmu.edu/~yke/musicretrieval/>. Acesso em: 15 set.2012.

GUIA php. Disponivel em: <http://php.net/manual/pt_BR/index.php>. Acesso em: 10 jul.2012.

GUIA Linux. Disponível em: <http:// www.vivaolinux.com.br>. Acesso em: 10 jul.2012.

MANUAL Audacity®, the Free, Cross-Platform Sound Editor. Disponível em <http://oficinademultimedia.files.wordpress.com/2011/09/manual_audacity.pdf>. Acesso em 13 out. 2012

MANUAL olimex SAM9-L9260. Disponível em : <http://www.olimex.com/Products/ARM/Atmel/SAM9-L9260>. Acesso em 5 out.2012.

GUIA googletest. Disponível em: <http://code.google.com/p/googletest/>. Acesso em 13 set. 2012

CUNHA, A.; O que são sistemas embarcados? 2008 Disponível em: <http://www.sabereletronica.com.br/secoes/leitura/274>. Acesso em: 07 out. 2012.

GOMES, P.; LEITE, T.; CAETANO, U. A Arquitetura ARM. Projeto de Sistemas Computacionais, Universidade de Campinas, 2005.

MATTOSO, M. Introdução a Banco de dados, 2007. Disponível em: <http://www.cos.ufrj.br/~marta/BdRel.pdf> . Acesso em: 19 out. 2012.

CRESSONI, P. *Short Time Fourier Transform* na Análise de Sinais Eletromiográficos, 2011. Relatório Final de trabalho de iniciação científica – Centro Universitário Fundação Santo André, Santo André, 2012.

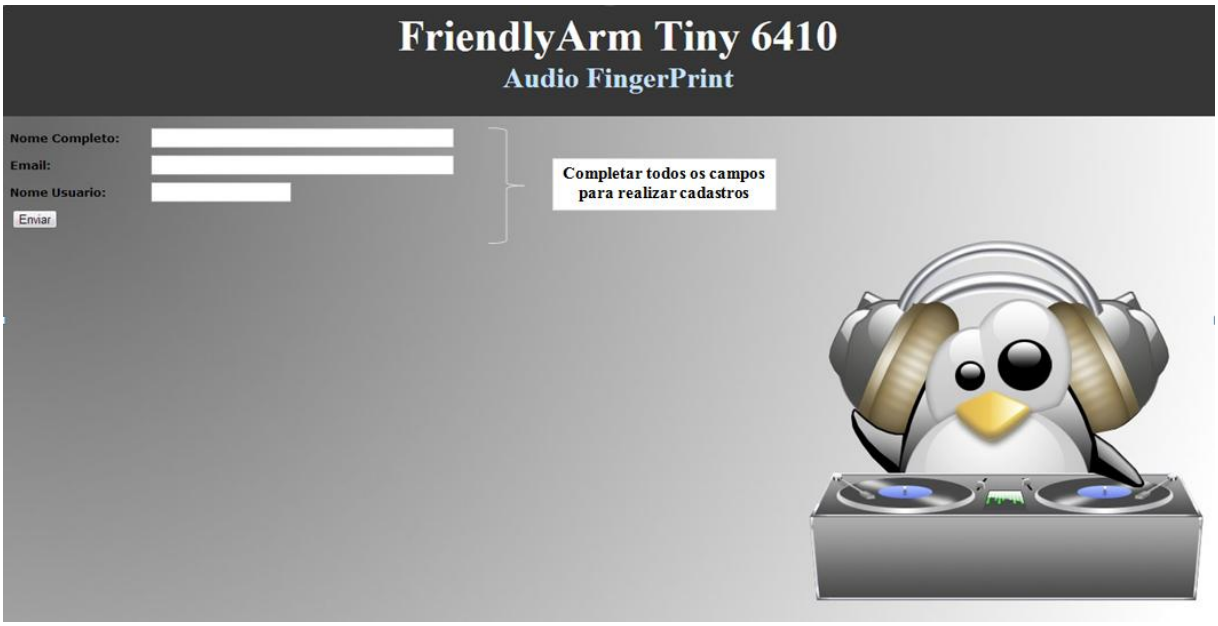
.

APÊNDICE A – Imagens do *website*

- **Página principal**

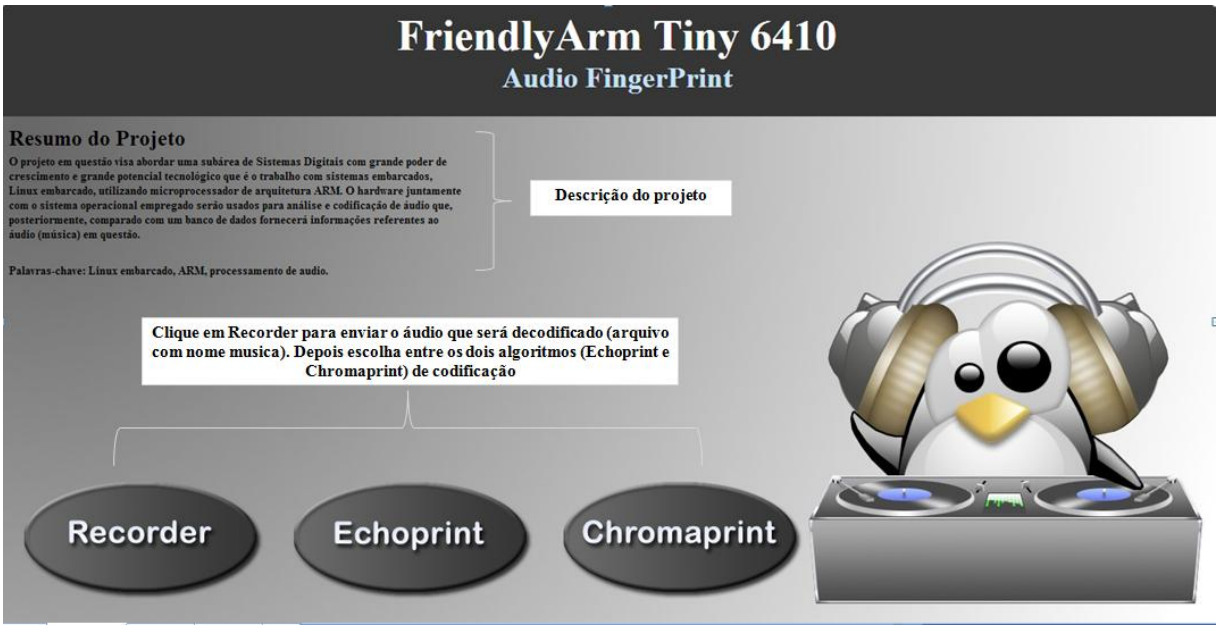
- **Página para realizar *login***

- **Página para realizar cadastro**



- **Página para submeter músicas**

- **Página para realizar o “reconhecimento” do áudio**



- **Página para exibição do conteúdo do banco de dados**