Centro Universitário Positivo - UnicenP
Núcleo de Ciências Exatas e Tecnológicas – NCET
Engenharia da Computação
Eduardo Theiss Przysiezny

MP3 Player

Curitiba
2005

Centro Universitário Positivo - UnicenP
Núcleo de Ciências Exatas e Tecnológicas – NCET
Engenharia da Computação
Eduardo Theiss Przysiezny

MP3 Player

Monografia apresentada à disciplina de Projeto Final, como requisito parcial à conclusão do Curso de Engenharia da Computação. Orientador: Prof. Valfredo Pilla Jr.

Curitiba
2005

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

NCET – Núcleo de Ciências Exatas e Tecnológicas
UNICENP – Centro Universitário Positivo
CF – *Compact Flash*
LCD – *Liquid Crystal Interface* (Display de Cristal Líquido)
MPEG – *Moving Pictures Experts Group*
ISO – *International Standards Organization*
DA – Digital Analógico

RESUMO

Com o intuito de estudar mídias alternativas de armazenamento e decodificação MP3, neste trabalho foi desenvolvido um MP3 *Player Standalone*, ou seja, um aparelho capaz de reproduzir arquivos de áudio gravados no formato MP3.

Para decodificação MP3 foram estudados chips decodificadores MP3 e o formato MP3 em geral, como mídia de armazenamento foi estudado o cartão de armazenamento do tipo *Compact Flash*, que compõem um das grandes variedades de mídia de armazenamento compacta dos dias de hoje, e também o sistema de arquivos FAT16.

Apesar de basicamente o projeto englobar MP3 e mídia de armazenamento *Compact Flash*, também serão necessários neste projeto estudos sobre displays LCD, microprocessadores, sinais analógicos e digitais.

Palavras-Chave: MP3, Compact Flash, FAT16, decodificação, armazenamento.

ABSTRACT

With intention to study alternative storage medias and decoding MP3, in this work has been developed an MP3 Player Standalone, or either, a device capable to reproduce archives of audio recorded in MP3 format.

For decoding MP3 have been studied MP3 decoder chips and the MP3 format in general, as media of storage has been studied the storage card Compact Flash, that compose one of the great varieties of compact storage medias of the present, and the FAT16 File System.

Although basically the project is the MP3 format and the Compact Flash storage media, also have been necessary in this project studies on LCD, microprocessors, analogical and digital signals.

Keywords: MP3, Compact Flash, FAT16, decoding, storage.

1 INTRODUÇÃO

Atualmente, com o desenvolvimento de dispositivos de armazenamento cada vez menores e de maior capacidade, existe um grande foco no desenvolvimento de aparelhos digitais com tamanho reduzido e grande capacidade de armazenamento. Dentre estes dispositivos podemos citas máquinas fotográficas digitais, *palm tops*, telefones celulares, entre outros.

Uma das tecnologias de armazenamento que vêm sendo muito empregada nestes tipos de aparelhos eletrônicos é a memória flash.

Com o intuito de estudar esta tecnologia e também outra tecnologia que a algum tempo esta em evidência, não só pela sua alta taxa de compressão, mas por questões éticas, este projeto teve por objetivo o desenvolvimento de um dispositivo reprodutor de arquivos de áudio no formato MP3 que funcione sem a necessidade de um computador.

2 FUNDAMENTAÇÃO TEÓRICA

Aqui é apresentada a fundamentação teórica para o projeto, que envolve pesquisas na área de circuitos digitais, unidades de armazenamento e compressão de arquivos de áudio.

2.1 Unidades de Armazenamento

Mídias de armazenamento secundárias, e dispositivos de armazenamento existem em quatro categorias principais: magnética, óptica, magneto-óptica, e de estado sólido. Este projeto teve o armazenamento em estado sólido.

2.1.1 Memória Flash

Com o aumento da popularidade de dispositivos portáteis como máquinas fotográficas digitais, registradores de voz, telefones celulares e computadores, houve também um aumento de necessidade por dispositivos de memória pequenos e baratos. O mais utilizado é o chamado de memória flash.

A memória flash usa chips de estado sólido para armazenar seus arquivos. De algum modo, estes chips são como os chips de RAM usados dentro de um computador, mas eles têm uma diferença importante. Eles não requerem nenhuma bateria e não perdem seus dados quando a bateria acaba. Seus arquivos são retidos indefinidamente sem qualquer força fornecida aos componentes do Flash. Estes chips são empacotados dentro de um recipiente com conectores e a unidade inteira é chamada de um cartão (CURTIN, 2001).

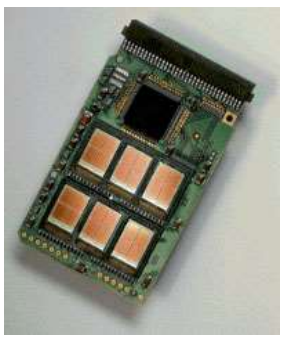


Figura 1 - Se você tira a cobertura de um cartão de memória flash, você encontra circuitos de estado sólido.

Até recentemente, a maioria dos cartões de flash estiveram enquadrados no padrão PC Card (PCMCIA) que é extremamente usado em computadores laptop. Porém, com o crescimento do mercado da máquina fotográfica digital e outros dispositivos, foram introduzidos formatos menores. Como resultado da competição, as máquinas fotográficas apóiam uma confusa variedade de cartões de memória flash que são incompatíveis entre si, que incluem os seguintes tipos: Cartões de PC, *Compact Flash*, *Miniature Card*, *Smart Media*, Cartões de Multimídia, *Memory Sticks*, entre outros. Este projeto utilizou o cartão do tipo *Compact Flash*.

2.1.2 Compact Flash (CF)

O *Compact Flash* (CF) é um dispositivo de armazenamento em massa muito pequeno, ele foi introduzido em 1994, pela Sandisk Corporation. Os cartões têm 1.433-polegadas (36,4 mm) de largura por 1.685-polegadas (42,8 mm) de comprimento, mais ou menos o tamanho de uma caixa de fósforos e utilizam a popular arquitetura ATA que emula um disco rígido. Naquela época parecia que o CF seria o vencedor no jogo da memória flash, era o formato de armazenamento de máquina fotográfica mais amplamente utilizado (CURTIN, 2001). O formato é apoiado por um número grande de companhias que forma a *Compact Flash Association* (COMPACT FLASH ASSOCIATION, 2004 (C)) para promover a adoção de um único formato de armazenamento de dados para máquinas fotográficas digitais.

O modo de operação suportado pelo CF é de 3.3V ou 5V, seu conector é similar ao conector PCMCIA, mas com 50 pinos, suportam nível de vibração de aproximadamente 2.000 Gs e seus dados são protegidos por tecnologias dinâmicas internas de gerenciamento de defeito e tecnologias de correção de erros (COMPACT FLASH ASSOCIATION, 2004 (C)).

O armazenamento no Compact Flash pode ser de diversos formatos, para este projeto foi utilizado o padrão FAT16 de armazenamento, compatível com o sistema operacional MS Windows ®.

2.1.3 Sistema de Arquivos FAT

Todos os sistemas de arquivos FAT foram desenvolvidos originalmente para a arquitetura IBM PC. A importância disto é que o sistema de arquivos FAT

armazenado na estrutura de dados do disco é todo “little endian”, ou seja, bytes em endereços menores são menos significativos. Se olharmos para uma entrada FAT32 armazenada em um disco como uma série de 4 bytes, sendo primeiro o byte 0 é o último o byte 3, teremos 32 bits números de 00 até 31 (onde 00 é o bit menos significativo e 31 é o bit mais significativo) armazenados da seguinte maneira:

Byte[3] = 31,30,29,28,27,26,25,24
Byte[2] = 23,22,21,20,19,18,17,16
Byte[1] = 15,14,13,12,11,10,09,08
Byte[0] = 07,06,05,04,03,02,01,00

Isto é importante, pois se a máquina é “big endian” será necessário traduzir os dados de “big” para “little endian” conforme são lidos ou armazenados dados no disco (MICROSOFT CORPORATION, 2000).

O volume do sistema de arquivos FAT é dividido em quatro regiões básicas, que estão armazenadas nesta ordem no volume:

0 – Região Reservada
1 – Região FAT
2 – Região do diretório root
3 – Região de arquivos e diretórios

A primeira estrutura de dados importante de um volume FAT é chamada de “BIOS Parameter Block” (BPB), que está localizada no primeiro setor do volume na Região Reservada. Este setor é algumas vezes chamado setor de boot, setor reservado ou setor 0, mas o mais importante é que este é o primeiro setor do volume.

Na tabela 1 pode ser visualizada a estrutura que compõe a região reservada do volume FAT.

Tabela 1 - Região Reservada do volume FAT

Nome do Campo	Offset (byte)	Tamanho (bytes)	Descrição
BS_jmpBoot	0	3	Instrução Jump para código de boot. Este campo tem dois formatos permitidos: **jmpBoot[0] = 0xEB, jmpBoot[1] = 0x??, jmpBoot[2] = 0x90** ou **jmpBoot[0] = 0xE9, jmpBoot[1] = 0x??, jmpBoot[2] = 0x??** **0x??** Indica que qualquer valor de 8 bits é permitido neste byte. Este campo forma a instrução de 3 bytes Intel x86 branch incondicional (jump), que realiza o jump para o código de boot do sistema operacional. Este código normalmente ocupa o resto do setor 0. Qualquer uma das duas formas é permitida, sendo normalmente utilizada a forma **jmpBoot[0] = 0xEB**.
BS_OEMName	3	8	"MSWIN4.1" Existem várias confusões sobre este campo. Este campo é somente uma string. Os sistemas operacionais microsoft não são influenciados por este campo, enquanto que outros sistemas FAT são. Por esta razão a string "MSWIN4.1" é o valor recomendável para este campo. Caso se deseje colocar outro valor para o campo, é possível, mas pode resultar que diversos sistemas FAT não reconhecerão o volume. O valor deste campo normalmente indica que sistema operacional formatou o volume.
BPB_BytsPerSec	11	2	Quantidade de bytes por setor. Este campo pode conter somente os seguintes valores: 512, 1024, 2048 ou 4096. Se for desejavel uma maior compatibilidade com sistemas antigos, deve ser utilizado o valor 512. Sistemas operacionais microsoft suportam todos os valores.
BPB_SecPerClus	13	1	Número de setores por unidade de alocação(cluster). Este valor deve ser uma potência de 2 e maior que 0. Os valores legais são 1, 2, 4, 8, 16, 32, 64 e 128. Note que nunca deve ser utilizado um valor que resulte em "Bytes por Cluster" (BPB_BytsPerSec * BPB_SecPerClus) maior que 32Kbytes (32 * 1024).
BPB_RsvdSecCnt	14	2	Número de setores reservados na Região Reservada do volume. Este campo deve ser diferente de 0, em volumes FAT16 e FAT12 este campo deve ser sempre 1.
BPB_NumFATs	16	1	Quantidade de estrutura de dados FAT no volume. Este campo normalmente contém o valor 2 para volumes FAT de qualquer tipo, mas qualquer valor maior que 2 ou o valor 1 é permitido.
BPB_RootEntCnt	17	2	Para volumes FAT12 e FAT16 este campo contém a quantidade de entradas de diretório (32 bytes) armazenadas no diretório root. Para os volumes FAT12 e FAT16, este campo deve sempre conter uma quantidade que quando multiplicada por 32 resulte em um múltiplo de BPB_BytsPerSec. Para maior compatibilidade, volumes FAT16 devem usar o valor 512.
BPB_TotSec16	19	2	Este é um campo antigo de 16 bits que contém a quantidade de setores do volume. Este campo pode conter o valor 0. Caso contenha 0, então o campo BPB_TotSec32 deve ser diferente de 0. Para volumes FAT32, este campo deve sempre ser 0.
BPB_Media	21	1	0xF8 é o valor padrão para mídias não removíveis. Para mídias removíveis é normalmente utilizado o valor 0xF0. Este campo pode conter os valores 0xF0, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE ou 0xFF. O mais importante sobre este campo é que o valor deste campo deve também estar no primeiro byte da FAT[0].

BPB_FATSz16	22	2	Este campo é a quantidade (16 bits) de setores ocupados por uma FAT12/16. Para volumes FAT32, este campo deve ser 0.
BPB_SecPerTrk	24	2	Setores por trilha para interrupção 0x13. Este campo é relevante apenas para mídias que possuem geometria (volumes são divididos em trilhas por múltiplas cabeças e cilindros) e são visíveis pela interrupção 0x13.
BPB_NumHeads	26	2	Número de cabeças para interrupção 0x13. Este campo é relevante apenas para mídias que possuem geometria (volumes são divididos em trilhas por múltiplas cabeças e cilindros) e são visíveis pela interrupção 0x13.
BPB_HiddSec	28	4	Quantidade de Setores escondidos precedendo a partição que contem este volume FAT. Este campo geralmente é relevante para mídias visíveis através da interrupção 0x13. E mídias não particionadas este campo deve ser sempre 0.
BPB_TotSec32	32	4	Este é o campo novo de 32 bits que contém a quantidade de setores do volume. Este campo pode conter o valor 0. Caso contenha 0, então o campo BPB_TotSec16 deve ser diferente de 0. Para volumes FAT32, este campo deve sempre ser diferente de 0.
BS_DrvNum	36	1	Este campo contém o número do driver para interrupção 0x13. Valor 0x00 para disquetes e 0x80 para discos rigído.
BS_Reserved1	37	1	Campo reservado, utilizado pelo Windows NT.
BS_BootSig	38	1	Contém o valor 0x29.
BS_VolID	39	4	Número de série do volume.
BS_VolLab	43	11	Label do Volume.
BS_FilSysType	54	8	Tipo do sistema de arquivos. ("FAT12", "FAT16" ou "FAT32"). Este campo não serve como base para definir o tipo de sistema de arquivos é apenas um campo informativo.

A próxima estrutura importante é a FAT proprieamente dita. Esta estrutura define uma lista ligada simples de “extensões” (clusters) de um arquivo. Um apontamento para um diretório FAT é nada mais do que um arquivo regular com um atributo especial indicando que isto é um diretório. Outra informação importante sobre diretórios é que o conteúdo do “arquivo” é uma série de entradas de diretório (32 bytes). A FAT mapeia a região de dados do volume através do número do cluster. O primeiro clustes de dados é o cluster 2.

Para calcular o ínicio da região de dados é necessário primeiramente calcular a quantidade de setores que o diretório root ocupa, conforme fórmula apresentada na equação 1.

*RootDirSectors = ((BPB_RootEntCnt * 32) + (BPB_BytsPerSec – 1)) / BPB_BytsPerSec; [equação 1]*

O ínicio da região de dados, o primeiro setor do cluster 2, é calculado da seguinte maneira, , conforme fórmula apresentada na equação 2.

*FirstDataSector = BPB_ResvdSecCnt + (BPB_NumFATs * FATSz16) + RootDirSectors; [equação 2]*

Um diretório FAT é nada mais do que um “arquivo” composto de uma lista linear de estruturas de 32 bytes. O único diretórios especial que sempre deve estar presente é o diretório root. Para volumes FAT12/16 o diretório root está localizado em uma região fixa, logo após a última estrutura FAT e possui um tamanho fixo de setores (RootDirSectors). Para FAT12/16 o primeiro setor do diretório root é relativo ao primeiro setor do volume FAT e pode ser calculado conforme segue, conforme fórmula apresentada na equação 3.

*FirstRootDirSecNum = BPB_ResvdSecCnt + (BPB_NumFATs * BPB_FATSz16); [equação 3]*

A tabela 2 mostra a estrutura que compõe uma entrada de diretório de 32 bytes.

Tabela 2 - Entrada de Diretório de 32 bits

Nome do Campo	Offset (byte)	Tamanho (bytes)	Descrição
DIR_Name	0	11	Nome curto.
DIR_Attr	11	1	Atributos do arquivo: 0x01 - Somente leitura 0x02 - Oculto 0x04 - Sistema 0x08 - Volume 0x10 - Diretório 0x20 - Arquivo 0x0F - Nome longo
DIR_NTRes	12	1	Reservado, para uso pelo Windows NT.
DIR_CrtTimeTenth	13	1	Milisegundos da criação do arquivo.
DIR_CrtTime	14	2	Hora em que o arquivo foi criado.
DIR_CrtDate	16	2	Date em que o arquivo foi criado.
DIR_LstAccDate	18	2	Data de último acesso ao arquivo.
DIR_FstClusHI	20	2	Word mais significativo que indica o primeiro cluster do arquivo. Deve conter 0 para entradas FAT12/16.
DIR_WrtTime	22	2	Hora da última escrita no arquivo.
DIR_WrtDate	24	2	Data da última escrita no arquivo.
DIR_FstClusLO	26	2	Word menos significativo que indica o primeiro cluster do arquivo.
DIR_FileSize	28	4	Campo de 32 bits que representa o tamanho do arquivo em bytes.

A partir do primeiro cluster de um arquivo pode ser calculado o setor onde este arquivo está localizado, conforme fórmula apresentada na equação 4.

*ThisFATSecNum = BPB_ResevdSecCnt + ((DIR_FstClusLO * 2) / BPB_BytsPerSec); [equação 4]*

2.2 Display de Informações

Para auxílio ao usuário, faz-se necessária a disponibilização de informações a respeito da música, tais como gênero, álbum, nome da música, entre outras. Os arquivos de áudio no formato MP3 possuem uma propriedade, chamada ID3 *tag*, que armazena estas informações, esta propriedade foi utilizada em conjunto com um display de cristal líquido (LCD) para disponibilizar informações e auxiliar ao usuário.

2.2.1 ID3 Tag

Ao produzirem o formato MP3 para compressão de áudio, este teve uma taxa de compressão tão proeminente e ainda uma qualidade muito boa, que foi adaptado logo como o padrão para música digital. Mas faltava a possibilidade de se incluir informação textual nos arquivos que utilizavam este formato (ID3 ORG, s.d.). De repente, alguém (Erik Kemp) teve a idéia de um bloco de 128 bytes que residiria no fim do arquivo, este bloco iria incluir título, artista, álbum, ano, gênero e comentário. A idéia foi implementada e logo Michael Mutschler estendeu o bloco, previamente chamado de ID3 *tag*, incluindo também a trilha do CD que originou o arquivo MP3 em questão. Ele usou os dois últimos bytes do comentário para isso, e nomeou esta variante de ID3 *tag* v1.1. A figura 2 mostra o formato do arquivo MP3 incluindo a Tag ID3 v1 e sua variante 1.1.

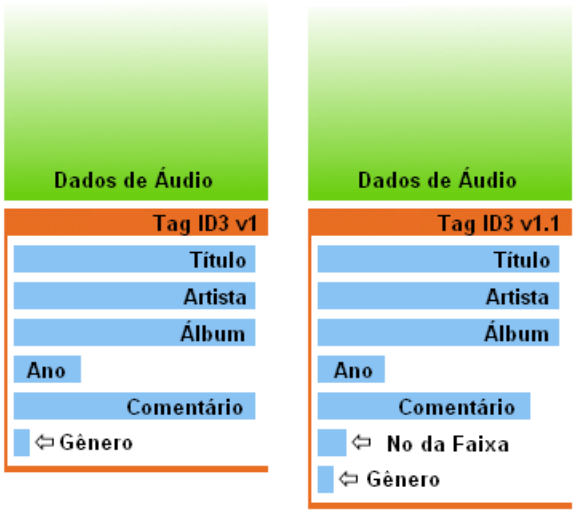


Figura 2 - Formato do arquivo MP3 incluindo as Tags ID3 v1 e v1.1

A tabela 3 mostra a estrutura que compõe as Tags ID3 v1 e v1.1.

Tabela 3 - Estrutura que compõe *tags* ID3 v1 e v1.1

Dados	Tag ID3 v1	Tag ID3 v1.1
Título	30 caracteres	30 caracteres
Artista	30 caracteres	30 caracteres
Álbum	30 caracteres	30 caracteres
Ano	4 caracteres	4 caracteres
Comentário	30 caracteres	28 caracteres
Gênero	1 byte	1 byte
No da Faixa	-	1 byte

Possuindo ainda muitas limitações surgiu recentemente a versão 2 do bloco ID3, que visa melhorar muitas limitações do padrão antigo.

2.3 MP3

Por volta de 1980, no Instituto Fraunhofer na Alemanha um grupo de pesquisadores desenvolveu um algoritmo para compressão de áudio chamado Eureka-EU 147. Um ano mais tarde foi criado o *Moving Pictures Expert Group* (MPEG), que possibilitou ao instituto trabalhar em conjunto com a *International Standards Organization* (ISO). Assim, diversas tecnologias para codificação de áudio

e vídeo foram criadas com base no MPEG. Entre estas estava o MPEG Layer 3, o popular MP3 (JONES, 2000).

Em 1997, Tomislav Uzelac um desenvolvedor da *Advanced Multimedia Products*, criou o AMP MP3 *Playback Engine*, que é considerado o primeiro reprodutor MP3. Logo uma dupla de universitários, Justin Frankel e Dmirty Boldyrev, utilizou a *engine* para criar o WinAMP.

Mas quem popularizou o MP3 foi Shawn Fanning, ao desenvolver um software que facilitava a troca deste tipo de arquivo através da internet, o Napster (JONES, 2000).

O MP3 é um formato de compressão que diminui arquivos de áudio sem perda significativa na qualidade do áudio. Este algoritmo permite uma compressão do áudio com uma razão de 11 para 1 (11:1), que rende um arquivo de aproximadamente 4 Mbytes para uma música de 3 minutos.

As deficiências do ouvido humano são a base técnica para compressão MP3. O princípio de funcionamento básico do mp3 é buscar num sinal de áudio normal, como um arquivo *wave*, todos os sinais redundantes e irrelevantes que não sensibilizam nosso ouvido. O algoritmo de compactação do mp3 corta freqüências muito altas, acima dos 20kHz, que não são audíveis pelo ouvido humano. Só aí já são muitos bits economizados. Em qualquer música, se duas freqüências muito próximas forem “tocadas” ao mesmo tempo, nosso ouvido somente ouvirá a mais forte, ou seja, o mp3 simplesmente diminui o número de bits desse sinal mais fraco e mantém os bits do sinal mais forte, diminuindo assim o tamanho final do arquivo na proporção 11:1 (qualidade semelhante ao CD).

A tabela 4 mostra algumas performances típicas da codificação MP3.

Tabela 4 - Performances típicas da codificação MP3

Qualidade de Som	Largura de Banda	Modo	Taxa de Bits (bitrate)	Razão de Compacatação
Som de Telefone	2,5 kHz	Mono	8 kbps	96:1
Melhor que ondas curtas	4,5 kHz	Mono	16 kbps	48:1
Melhor que rádio AM	7,5 kHz	Mono	32 kbps	24:1
Similar ao rádio FM	11 kHz	Stereo	56..64 kbps	26..24:1
Próximo de CD	15 kHz	Stereo	96 kbps	16:1
CD	>15 kHz	Stereo	112..128 kbps	14..12:1

3 ESPECIFICAÇÃO TÉCNICA

A descrição a seguir apresenta os módulos de hardware e software identificando quais as tecnologias, técnicas e procedimentos utilizados.

3.1 Descrição Geral do Sistema

O projeto consiste em um dispositivo capaz de reproduzir arquivos de áudio codificados no formato MP3. Este dispositivo não depende do microcomputador para a reprodução destes arquivos, trata-se de um dispositivo *standalone*. Um dispositivo portátil que pode ser utilizado dentro de um automóvel.

Para armazenamento dos arquivos de áudio no formato MP3 este dispositivo utiliza um cartão de memória do tipo *compact flash*, este tipo de cartão é muito utilizado nos dias atuais em diversos dispositivos digitais portáteis, como máquinas fotográficas digitais, *palm tops*, entre outros.

Para a leitura dos arquivos do cartão de memória foi utilizado um microprocessador RISC ATMEL da família ATMEGA, que também lê informações sobre as músicas, para posteriormente disponibilizar estas informações em um display de cristal líquido. Além destas funções, o microprocessador também executa funções de comandos sobre as músicas do cartão, tais como: ir para a próxima música, voltar para música anterior, pausar, parar, reproduzir, entre outras. Estas funções são chamadas através de botões do tipo push-button.

Depois de realizada a leitura da música armazenada no cartão, esta é decodificada. Este trabalho é realizado por um chip decodificador de MP3, o qual transforma o arquivo em áudio digital e o transforma em áudio analógico através de um conversor DA (Digital-Analógico), com uma saída de áudio para um fone de ouvido ou uma caixa de som.

3.2 Especificação de Hardware

A implementação do hardware foi dividida em módulos:

- Módulo de Microprocessamento
- Módulo de Armazenamento
- Módulo de Comunicação

- Módulo de Controle
- Módulo de Display de Informações
- Módulo de Decodificação

A figura 3 apresenta o diagrama em blocos dos módulos de hardware.

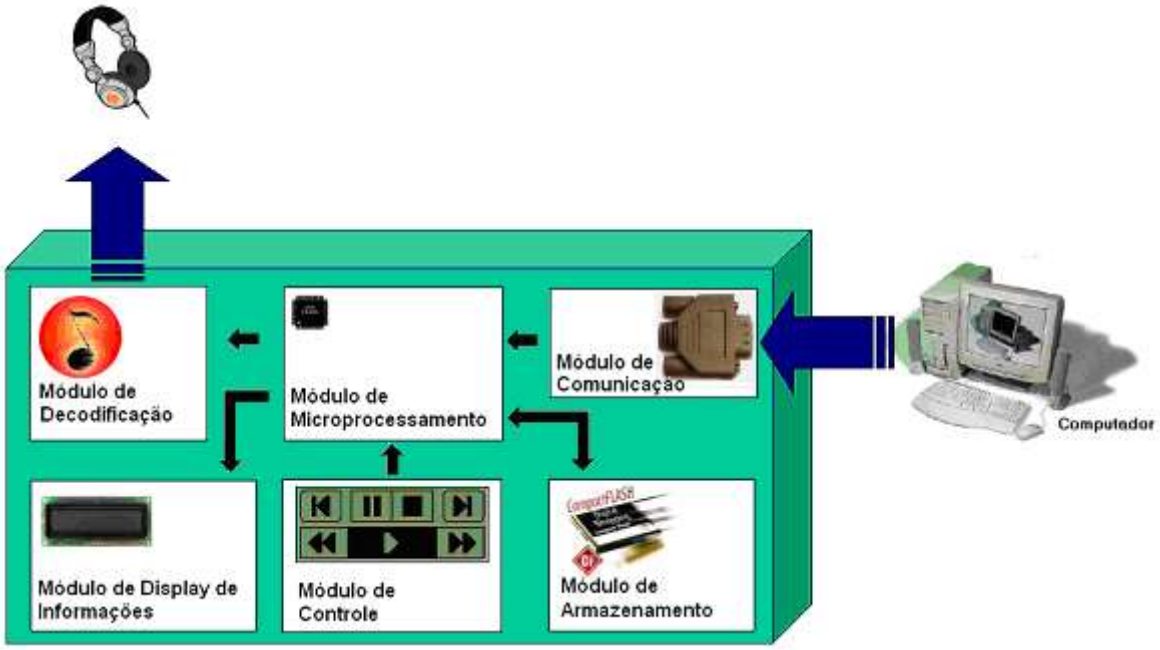


Figura 3 - Diagrama em blocos dos módulos de *hardware*

3.2.1 Módulo de Microprocessamento

O módulo de microprocessamento é a parte do hardware responsável por realizar todo o controle e processamento de recursos do MP3 Player É neste módulo que está localizado o *firmware* do dispositivo.

As principais funções deste módulo são: leitura de arquivos no módulo de armazenamento, controle de comunicação com o computador, controle de reprodução dos arquivos no formato MP3 e exibição de informações no display de cristal líquido.

3.2.2 Módulo de Armazenamento

O módulo de armazenamento é a parte do hardware responsável por realizar o armazenamento de arquivos que são utilizados pelo dispositivo, este módulo é controlado pelo módulo de microprocessamento.

Este módulo utiliza o sistema de armazenamento de arquivos FAT16 compatível com vários sistemas operacionais de computadores e pode armazenar qualquer tipo de arquivo, mesmo arquivos que não estejam no formato MP3.

3.2.3 Módulo de Comunicação

O módulo de comunicação é a parte do hardware responsável por realizar a interface entre o dispositivo e o computador, este módulo é controlado pelo módulo de microprocessamento e foi desenvolvido utilizando-se comunicação serial.

3.2.4 Módulo de Controle

O módulo de controle é a parte do hardware responsável por enviar comandos para o módulo de microprocessamento, de forma a executar funções de reprodução do áudio armazenado em formato MP3, bem como, navegação e configuração de funções disponíveis no dispositivo.

Os sinais de controle enviados para o módulo de microprocessamento são gerados através de acionamentos, realizados pelo usuário, em botões do tipo push-buttons.

3.2.5 Módulo de Display de Informações

O módulo de display de informações é a parte do hardware responsável por exibir informações de controle e configuração do dispositivo, bem como, informações do arquivo no formato MP3 que estiver sendo reproduzido no momento, este módulo é controlado pelo módulo de microprocessamento.

São disponibilizadas informações de volume atual, nome do artista e nome da música que está sendo reproduzida no momento e lista de músicas para seleção.

3.2.6 Módulo de Decodificação

O módulo de decodificação de MP3 é a parte do hardware responsável por realizar a decodificação do arquivo armazenado em formato MP3, conversão DA do resultado obtido durante a decodificação e controle digital de volume, este módulo é controlado pelo módulo de microprocessamento.

3.3 Especificação de Software

3.3.1 Descrição Geral do Sistema

A implementação de *software* foi divida em duas partes, implementação de *firmware* e de *software* para PC.

O *firmware* é o *software* que será desenvolvido para realizar o controle das funções do *hardware* do dispositivo, toda sua implementação foi realizada através de técnicas de desenvolvimento estruturado.

O *software* para PC é o *software* implementado através de técnicas de orientação a objetos e é utilizado pelo usuário para edição de informações da música e troca de arquivos entre o cartão de memória e o PC.

3.3.2 Diagrama em blocos

A figura 4 representa o diagrama em blocos do *firmware* (A) e *software* (B) desenvolvidos.

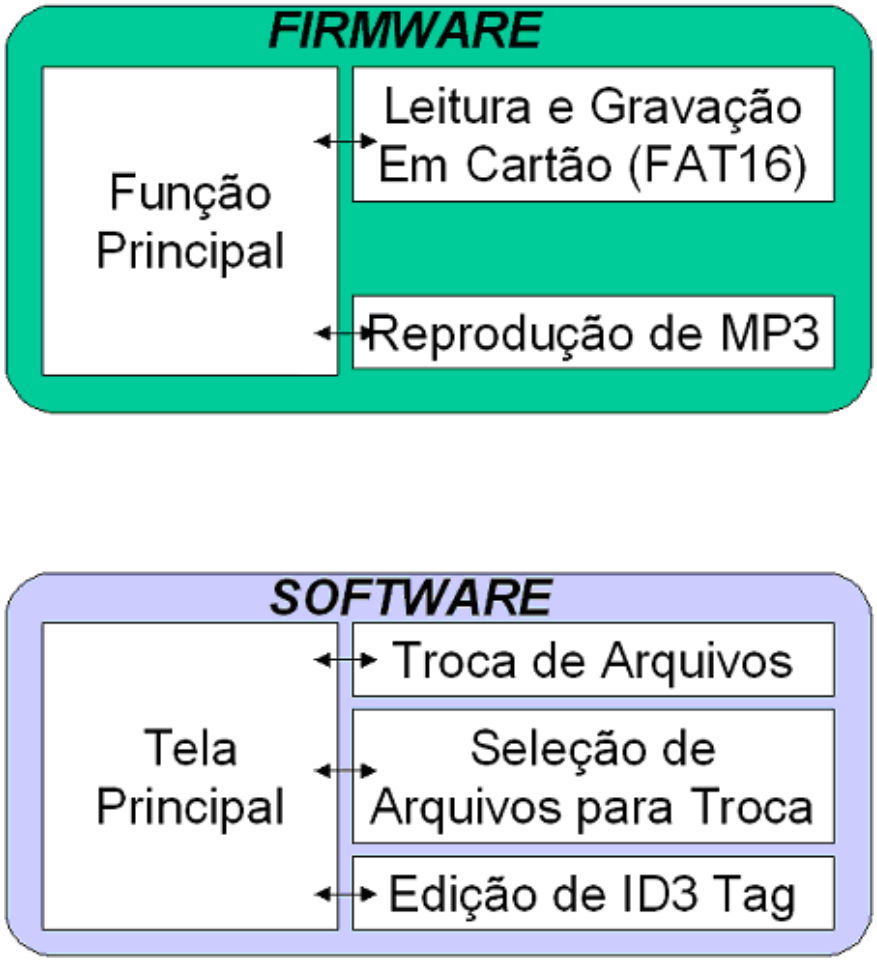


Figura 4 - Diagrama em blocos do *firmware* (A) e *software*(B)

3.3.3 Protótipo de Interface

A figura 5 apresenta o protótipo de interface com o usuário para edição de informações da música MP3, bem como transferência de arquivos entre o dispositivo e o PC.

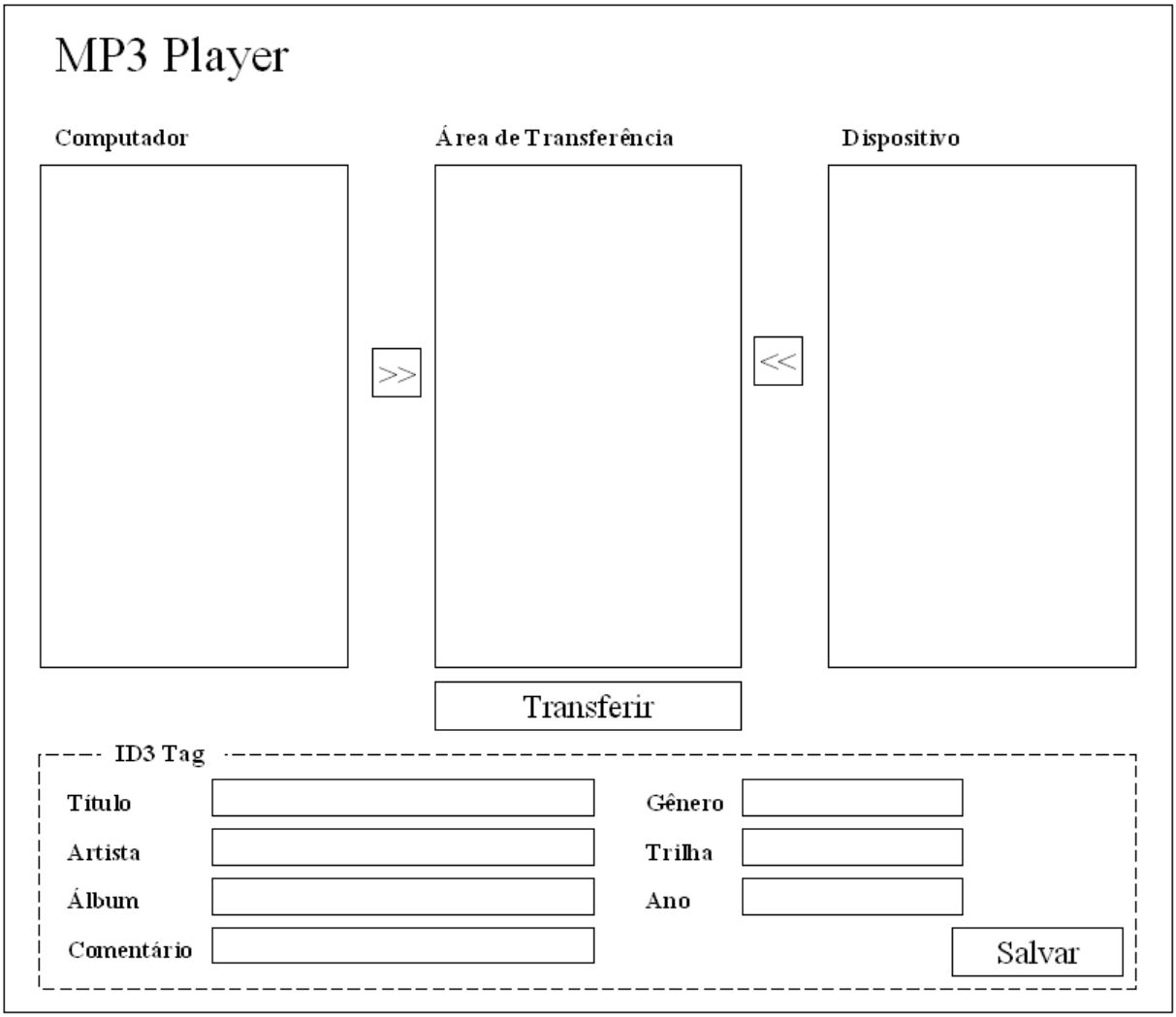


Figura 5 - Protótipo de *interface* com usuário

3.4 Validação

3.4.1 Validação de Hardware

A validação do hardware foi realizada em duas etapas, em uma primeira etapa a validação dos principais módulos de hardware é realizada durante a inicialização do sistema e em uma segunda etapa será realizada a validação dos módulos restantes através da interação do usuário.

A validação do módulo de microprocessamento é a primeira validação realizada, pois o funcionamento de todos os outros módulos depende do funcionamento deste módulo.

Logo após inicialização do módulo de microprocessamento, é realizada inicialização do módulo de display de informações onde é apresentado o logo da instituição de ensino e disponibilizadas as informações de inicialização do sistema.

O módulo de microprocessamento, então, envia sinais de controle para o módulo de decodificação gerando áudio no fone ligado ao dispositivo e validando então o módulo de decodificação. Em seguida, o módulo de microprocessamento envia sinais de controle para o módulo de display de armazenamento para inicializar o padrão de formatação FAT16.

Somente após esta inicialização é realizada a validação dos outros módulos, pois são estes módulos que realizam a interface entre o dispositivo e o usuário.

A validação dos módulos de controle e armazenamento depende de interação do usuário e é realizada em conjunto. Para isso, o usuário do dispositivo deverá clicar no botão de iniciar reprodução. O dispositivo irá realizar então a leitura de arquivos disponíveis no módulo de armazenamento, validando ambos os módulos e mostrando no display de cristal líquido as informações dos arquivos lidos. Em caso de falha será exibida uma mensagem de erro no display de cristal líquido.

A validação do módulo de comunicação é realizada com software de terminal no microcomputador.

3.4.2 Validação de Software

A validação do software é realizada através de interação do usuário.

Depois de realizada a inicialização do software o usuário poderá selecionar arquivos para transferir entre o dispositivo e o PC. Caso o arquivo seja do tipo MP3 no canto inferior da tela são apresentadas as informações da música para edição, o usuário poderá alterá-las conforme sua preferência.

Ao clicar no botão transferir, o software realiza comunicação com o cartão de memória, realizando troca de arquivos e finalizando a validação do software.

3.5 Recursos

A plataforma de desenvolvimento está relacionada às ferramentas e equipamentos utilizados durante o período de graduação juntamente com as necessidades do projeto bem como a familiaridade da linguagem de desenvolvimento e dos tipos de equipamentos e dispositivos que se encontra no mercado. Incorporando todo o hardware e software necessário para o desenvolvimento e implementação do projeto.

A tabela 5 apresenta os recursos necessários para constituição do ambiente de desenvolvimento.

Tabela 5 - Recursos Necessários para Desenvolvimento

Recurso
Computador Pentium III 1GHz, 128 Mb RAM
Sistema Operacional Microsoft Windows XP
Acesso à internet
Borland C++ Builder 5
AVR Studio
Microsoft Office 2000
Osciloscópio
Multímetro
Fonte de alimentação

3.6 Custos

A estimativa de custo/investimento do projeto toma como base de cálculo os valores dos produtos, dispositivos e horas de desenvolvimento associadas ao projeto. Esses valores refletem os gastos que foram necessários para o desenvolvimento do projeto como um todo.

Existem ainda alguns valores que não são considerados, por tanto não agregam valo real ao projeto, esses valores estão relacionados a alguns softwares e equipamentos que são cedidos pela universidade.

A tabela 6 descreve os custos tanto de software como de hardware desconsiderando os dispositivos, equipamentos entre outros, necessários para o desenvolvimento do projeto.

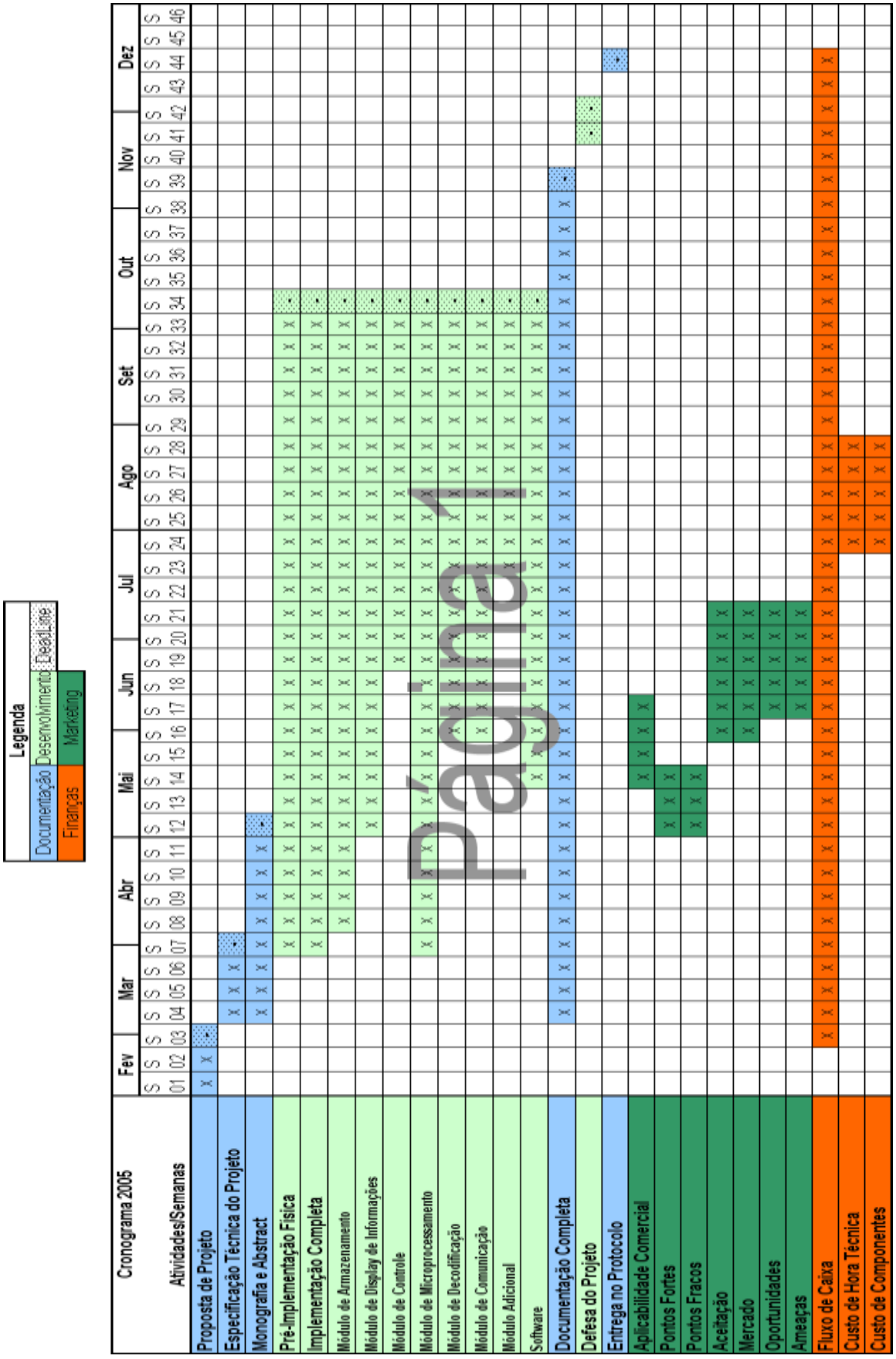
Tabela 6 - Estimativa de Custos

Recurso	**Quantidade**	**Custo**
Microcomputador Pentium III 1GHz, 128 Mb RAM	1	1600, 00 R$
Sistema Operacional Windows	1	500,00 R$
Borland C++ Builder	1	350,00 R$
Microprocessador	1	40,00 R$
Conversor DA	1	7, 00 U$
Connector Compact Flash	1	6,00 U$
Cartão Compact Flash 256 Mb	1	160,00 R$
Display de Cristal Líquido	1	30,00 U$
Horas de Desenvolvimento	350	8.750,00 R$
Taxas de Importação/Correio	1	150,00 R$
Outros Equipamentos de Laboratório		10000,00 R$
Total		21.550,00 R$ e 43,00 U$

3.7 Cronograma

A tabela 7 apresenta o cronograma do projeto em relação aos meses e semanas de desenvolvimento.

Tabela 7 - Cronograma



4 PROJETO

4.1 Projeto de *Hardware*

O projeto do hardware foi dividido em módulos, nos próximos tópicos é apresentado o projeto de cada módulo isolado.

O anexo I apresenta a lista de componentes necessários para o desenvolvimento do projeto.

4.1.1 Módulo de Microprocessamento

O anexo II apresenta o diagrama esquemático do módulo de microprocessamento.

A tabela 8 apresenta os sinais de interface do módulo de microprocessamento.

Tabela 8 - Sinais de interface do módulo de microprocessamento

Pino	Rótulo do Sinal	Tipo do Sinal	Descrição
U3-3	3V3	Digital	Função: Alimentação Amplitude: 3V3
U2-3	5V	Digital	Função: Alimentação Amplitude: 5V
J15-1 J15-2	GND	Digital	Função: Alimentação Amplitude: 0V
IC1-28	CF_!CE1	Digital	Função: Habilita Cartão Barramento: Controle Lógica: 1 (Cartão Desablitado) / 0 (Cartão Habilitado)
IC1-29	CF_!CD1	Digital	Função: Detecção do Cartão Barramento: Controle Lógica: 1 (Cartão Não Detectado) / 0 (Cartão Detectado)
IC1-30	CF_RST	Digital	Função: Reset do Cartão

			Barramento: Controle Lógica: 1 (Reset) / 0 (Estado Normal)
IC1-31	CF_!OE	Digital	Função: Habilita Saída do Cartão Barramento: Controle Lógica: 1 (Saída Desabilitada) / 0 (Saída Habilitada)
IC1-32	CF_!WE	Digital	Função: Habilita Escrita do Cartão Barramento: Controle Lógica: 1 (Escrita Desabilitada) / 0 (Escrita Habilitada)
IC1-4	CF_RDY	Digital	Função: Indica se o cartão está pronto para receber dados Barramento: Controle Lógica: 1 (Está pronto) / 0 (Não Está Pronto)
IC1-61	CF_D0	Digital	Função: Bit de dados 0 Barramento: Dados
IC1-60	CF_D1	Digital	Função: Bit de dados 1 Barramento: Dados
IC1-59	CF_D2	Digital	Função: Bit de dados 2 Barramento: Dados
IC1-58	CF_D3	Digital	Função: Bit de dados 3 Barramento: Dados
IC1-57	CF_D4	Digital	Função: Bit de dados 4 Barramento: Dados
IC1-56	CF_D5	Digital	Função: Bit de dados 5 Barramento: Dados
IC1-55	CF_D6	Digital	Função: Bit de dados 6 Barramento: Dados
IC1-54	CF_D7	Digital	Função: Bit de dados 7 Barramento: Dados
IC1-33	CF_A0	Digital	Função: Bit de endereços 0 Barramento: Dados
IC1-34	CF_A1	Digital	Função: Bit de endereços 1

			Barramento: Dados
IC1-43	CF_A2	Digital	Função: Bit de endereços 2 Barramento: Dados
IC1-2	SERIAL_RX	Digital	Função: Recepção de dados serial Barramento: Dados
IC1-3	SERIAL_TX	Digital	Função: Envio de dados para serial Barramento: Dados
IC1-9	C0	Digital	Função: Bit de Controle 0 Barramento: Controle Lógica: - (Definida por *firmware*)
IC1-8	C1	Digital	Função: Bit de Controle 1 Barramento: Controle Lógica: - (Definida por *firmware*)
IC1-7	C2	Digital	Função: Bit de Controle 2 Barramento: Controle Lógica: - (Definida por *firmware*)
IC1-6	C3	Digital	Função: Bit de Controle 3 Barramento: Controle Lógica: - (Definida por *firmware*)
IC1-25	C4	Digital	Função: Bit de Controle 4 Barramento: Controle Lógica: - (Definida por *firmware*)
IC1-26	C5	Digital	Função: Bit de Controle 5 Barramento: Controle Lógica: - (Definida por *firmware*)
IC1-51	KS_D/I	Digital	Função: Seleciona Função dos Dados do LCD Barramento: Controle Lógica: 1 (Dados) / 0 (Instruções)
IC1-50	KS_R/!W	Digital	Função: Seleciona Fluxo do Barramento de Dados do LCD Barramento: Controle Lógica: 1 (Leitura) / 0 (Escrita)
IC1-49	KS_EN	Digital	Função: Habilita LCD

			Barramento: Controle Lógica: 1 (Habilita) / 0 (Desabilita)
IC1-35	KS_D0	Digital	Função: Bit de dados 0 Barramento: Dados
IC1-36	KS_D1	Digital	Função: Bit de dados 1 Barramento: Dados
IC1-37	KS_D2	Digital	Função: Bit de dados 2 Barramento: Dados
IC1-38	KS_D3	Digital	Função: Bit de dados 3 Barramento: Dados
IC1-39	KS_D4	Digital	Função: Bit de dados 4 Barramento: Dados
IC1-40	KS_D5	Digital	Função: Bit de dados 5 Barramento: Dados
IC1-41	KS_D6	Digital	Função: Bit de dados 6 Barramento: Dados
IC1-42	KS_D7	Digital	Função: Bit de dados 7 Barramento: Dados
IC1-48	KS_CS1	Digital	Função: Habilita LCD Direito Barramento: Controle Lógica: 1 (Habilita) / 0 (Desabilita)
IC1-47	KS_CS2	Digital	Função: Habilita LCD Esquerdo Barramento: Controle Lógica: 1 (Habilita) / 0 (Desabilita)
U1-7	VS_SO	Digital	Função: Saída serial de dados do chip decodificador mp3 Barramento: Dados
U1-14	VS_SI	Digital	Função: Entrada serial de dados e instruções do chip decodificador mp3 Barramento: Dados / Instruções
U1-15	VS_SCLK	Digital	Função: Clock para entrada/saída serial de dados Barramento: Controle Lógica: Clock

U1-16	VS_!XCS	Digital	Função: Habilita entrada para chip Barramento: Controle Lógica: 1 (Desabilita entrada) / 0 (Habilita entrada)
U1-17	VS_!XRESET	Digital	Função: Reset do Chip Barramento: Controle Lógica: 1 (Normal) / 0 (Reset)
U1-8	VS_DREQ	Digital	Função: Requisição de Dados do chip decodificador Barramento: Controle Lógica: 1 (Requer Dados) / 0 (Não requer dados)
U1-18	VS_!XDCS/BSYNC	Digital	Função: Sincronismo do decodificador Barramento: Controle Lógica: 1 (Indica novo byte de áudio) / 0 (estado normal)
J14-5	ISP_MOSI	Digital	Função: In-System Programming MOSI Barramento: Dados
J14-3	ISP_!RST	Digital	Função: In-System Programming !Reset Barramento: Controle
J14-2	ISP_SCK	Digital	Função: In-System Programming Clock Barramento: Controle
J14-1	ISP_MISO	Digital	Função: In-System Programming MISO Barramento: Dados

4.1.2 Módulo de Armazenamento

O anexo III apresenta o diagrama esquemático do módulo de armazenamento.

A tabela 9 apresenta os sinais de interface do módulo de armazenamento.

Tabela 9 - Sinais de interface do módulo de microprocessamento

Pino	**Rótulo do Sinal**	**Tipo do Sinal**	**Descrição**
J6-13	5V	Digital	Função: Alimentação

J6-32 J6-38 J6-44			Amplitude: 5V
J6-1 J6-8 J6-10 J6-11 J6-12 J6-14 J6-15 J6-16 J6-17 J6-39 J6-50	GND	Digital	Função: Alimentação Amplitude: 0V
J6-7	CF_!CE1	Digital	Função: Habilita Cartão Barramento: Controle Lógica: 1 (Cartão Desablitado) / 0 (Cartão Habilitado)
J6-26	CF_!CD1	Digital	Função: Detecção do Cartão Barramento: Controle Lógica: 1 (Cartão Não Detectado) / 0 (Cartão Detectado)
J6-41	CF_RST	Digital	Função: Reset do Cartão Barramento: Controle Lógica: 1 (Reset) / 0 (Estado Normal)
J6-9	CF_!OE	Digital	Função: Habilita Saída do Cartão Barramento: Controle Lógica: 1 (Saída Desabilitada) / 0 (Saída Habilitada)
J6-36	CF_!WE	Digital	Função: Habilita Escrita do Cartão Barramento: Controle Lógica: 1 (Escrita Desabilitada) / 0 (Escrita Habilitada)
J6-37	CF_RDY	Digital	Função: Indica se o cartão está pronto para

			receber dados Barramento: Controle Lógica: 1 (Está pronto) / 0 (Não Está Pronto)
J6-21	CF_D0	Digital	Função: Bit de dados 0 Barramento: Dados
J6-22	CF_D1	Digital	Função: Bit de dados 1 Barramento: Dados
J6-23	CF_D2	Digital	Função: Bit de dados 2 Barramento: Dados
J6-2	CF_D3	Digital	Função: Bit de dados 3 Barramento: Dados
J6-3	CF_D4	Digital	Função: Bit de dados 4 Barramento: Dados
J6-4	CF_D5	Digital	Função: Bit de dados 5 Barramento: Dados
J6-5	CF_D6	Digital	Função: Bit de dados 6 Barramento: Dados
J6-6	CF_D7	Digital	Função: Bit de dados 7 Barramento: Dados
J6-20	CF_A0	Digital	Função: Bit de endereços 0 Barramento: Dados
J6-19	CF_A1	Digital	Função: Bit de endereços 1 Barramento: Dados
J6-18	CF_A2	Digital	Função: Bit de endereços 2 Barramento: Dados

4.1.3 Módulo de Comunicação

O anexo IV apresenta o diagrama esquemático do módulo de comunicação.

A tabela 10 apresenta os sinais de interface do módulo de comunicação.

Tabela 10 - Sinais de interface do módulo de comunicação

Pino	Rótulo do Sinal	Tipo do Sinal	Descrição
IC1-12	SERIAL_RX	Digital	Função: Recepção de dados serial Barramento: Dados
IC1-11	SERIAL_TX	Digital	Função: Envio de dados para serial Barramento: Dados

4.1.4 Módulo de Controle

O anexo V apresenta o diagrama esquemático do módulo de controle.

A tabela 11 apresenta os sinais de interface do módulo de controle.

Tabela 11 - Sinais de interface do módulo de controle

Pino	Rótulo do Sinal	Tipo do Sinal	Descrição
-	5V	Digital	Função: Alimentação Amplitude: 5V
-	GND	Digital	Função: Alimentação Amplitude: 0V
SW3	C0	Digital	Função: Bit de Controle 0 Barramento: Controle Lógica: - (Definida por *firmware*)
SW4	C1	Digital	Função: Bit de Controle 1 Barramento: Controle Lógica: - (Definida por *firmware*)
SW5	C2	Digital	Função: Bit de Controle 2 Barramento: Controle Lógica: - (Definida por *firmware*)
SW6	C3	Digital	Função: Bit de Controle 3

			Barramento: Controle Lógica: - (Definida por *firmware*)
SW7	C4	Digital	Função: Bit de Controle 4 Barramento: Controle Lógica: - (Definida por *firmware*)
SW8	C5	Digital	Função: Bit de Controle 5 Barramento: Controle Lógica: - (Definida por *firmware*)

4.1.5 Módulo de Display de Informações

O anexo VI apresenta o diagrama esquemático do módulo de display de informações.

A tabela 12 apresenta os sinais de interface do módulo de display de informações.

Tabela 12 - Sinais de interface do módulo de informações

Pino	**Rótulo do Sinal**	**Tipo do Sinal**	**Descrição**
J16-1	GND	Digital	Função: Alimentação Amplitude: 0V
J16-2	5V	Digital	Função: Alimentação Amplitude: 5V
J16-4	Dados/Instruções	Digital	Função: Seleciona Função dos Dados do LCD Barramento: Controle Lógica: 1 (Dados) / 0 (Instruções)
J16-5	Leitura/!Escrita	Digital	Função: Seleciona Fluxo do Barramento de Dados do LCD Barramento: Controle Lógica: 1 (Leitura) / 0 (Escrita)
J16-6	Habilitado	Digital	Função: Habilita LCD

			Barramento: Controle Lógica: 1 (Habilita) / 0 (Desabilita)
J16-7	Bit de Dados 0	Digital	Função: Bit de dados 0 Barramento: Dados
J16-8	Bit de Dados 1	Digital	Função: Bit de dados 1 Barramento: Dados
J16-9	Bit de Dados 2	Digital	Função: Bit de dados 2 Barramento: Dados
J16-10	Bit de Dados 3	Digital	Função: Bit de dados 3 Barramento: Dados
J16-11	Bit de Dados 4	Digital	Função: Bit de dados 4 Barramento: Dados
J16-12	Bit de Dados 5	Digital	Função: Bit de dados 5 Barramento: Dados
J16-13	Bit de Dados 6	Digital	Função: Bit de dados 6 Barramento: Dados
J16-14	Bit de Dados 7	Digital	Função: Bit de dados 7 Barramento: Dados
J16-15	CS1	Digital	Função: Habilita LCD Direito Barramento: Controle Lógica: 1 (Habilita) / 0 (Desabilita)
J16-16	CS2	Digital	Função: Habilita LCD Esquerdo Barramento: Controle Lógica: 1 (Habilita) / 0 (Desabilita)

4.1.6 Módulo de Decodificação

O anexo VII apresenta o diagrama esquemático do módulo de decodificação.

A tabela 13 apresenta os sinais de interface do módulo de decodificação.

Tabela 13 - Sinais de interface do módulo de decodificação

Pino	Rótulo do Sinal	Tipo do Sinal	Descrição
-	3V3	Digital	Função: Alimentação Amplitude: 3V3
U5-22 U5-26 U5-37 U5-40 U5-41 U5-47	GND	Digital	Função: Alimentação Amplitude: 0V
U5-30	SO	Digital	Função: Saída serial de dados do chip decodificador mp3 Barramento: Dados
U5-29	SI	Digital	Função: Entrada serial de dados e instruções do chip decodificador mp3 Barramento: Dados / Instruções
U5-28	SCLK	Digital	Função: Clock para entrada/saída serial de dados Barramento: Controle Lógica: Clock
U5-23	!XCS	Digital	Função: Habilita entrada para chip Barramento: Controle Lógica: 1 (Desabilita entrada) / 0 (Habilita entrada)
U5-3	!XRESET	Digital	Função: Reset do Chip Barramento: Controle Lógica: 1 (Normal) / 0 (Reset)
U5-8	DREQ	Digital	Função: Requisição de Dados do chip decodificador Barramento: Controle Lógica: 1 (Requer Dados) / 0 (Não requer dados)
U5-13	!XDCS/BSYNC	Digital	Função: Sincronismo do decodificador

			Barramento: Controle Lógica: 1 (Indica novo byte de áudio) / 0 (estado normal)

4.2 Projeto de *Firmware*

O projeto de *firmware* se refere ao *software* desenvolvido para o microcontrolador do MP3 Player, ou seja, o *firmware* é o responsável pelo controle de todas as funções do hardware. O desenvolvimento do mesmo foi realizado através de técnicas de programação estruturada.

A figura 6 representa o diagrama de contexto do projeto do *firmware*.

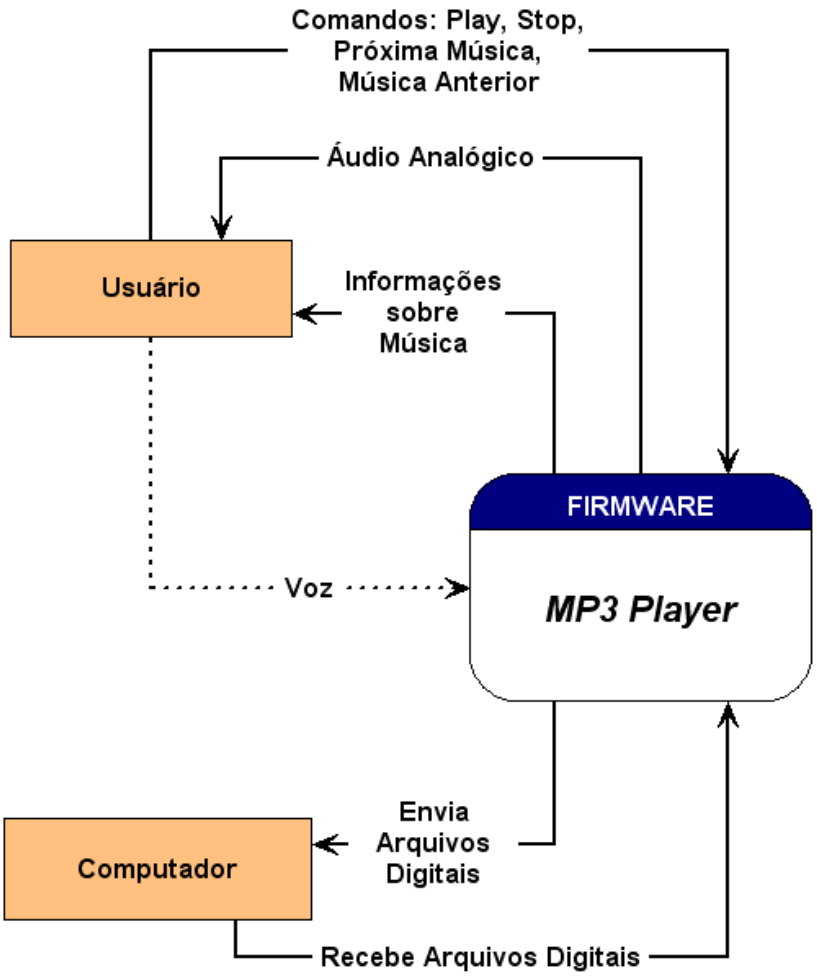


Figura 6 - Diagrama de contexto do projeto do *firmware*

A figura 7 representa o diagrama de estados do projeto do *firmware*.

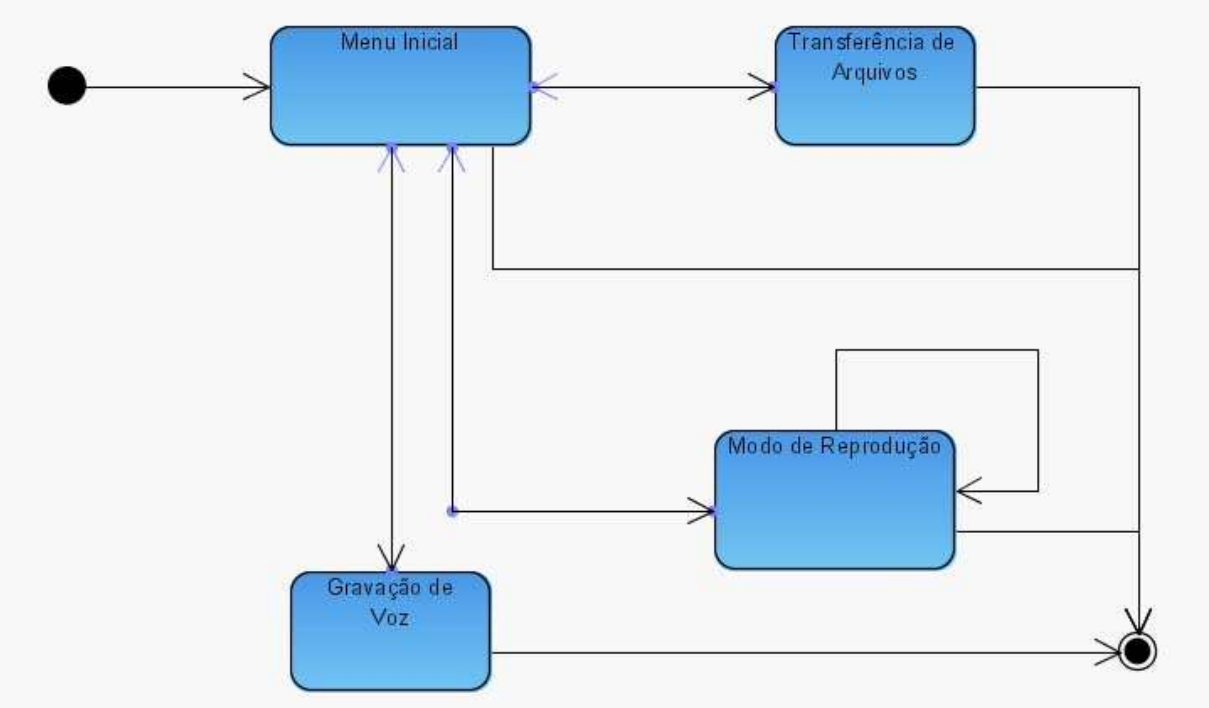


Figura 7 - Diagrama de estados do projeto do *firmware*

A figura 8 apresenta o fluxograma do projeto do *firmware*.

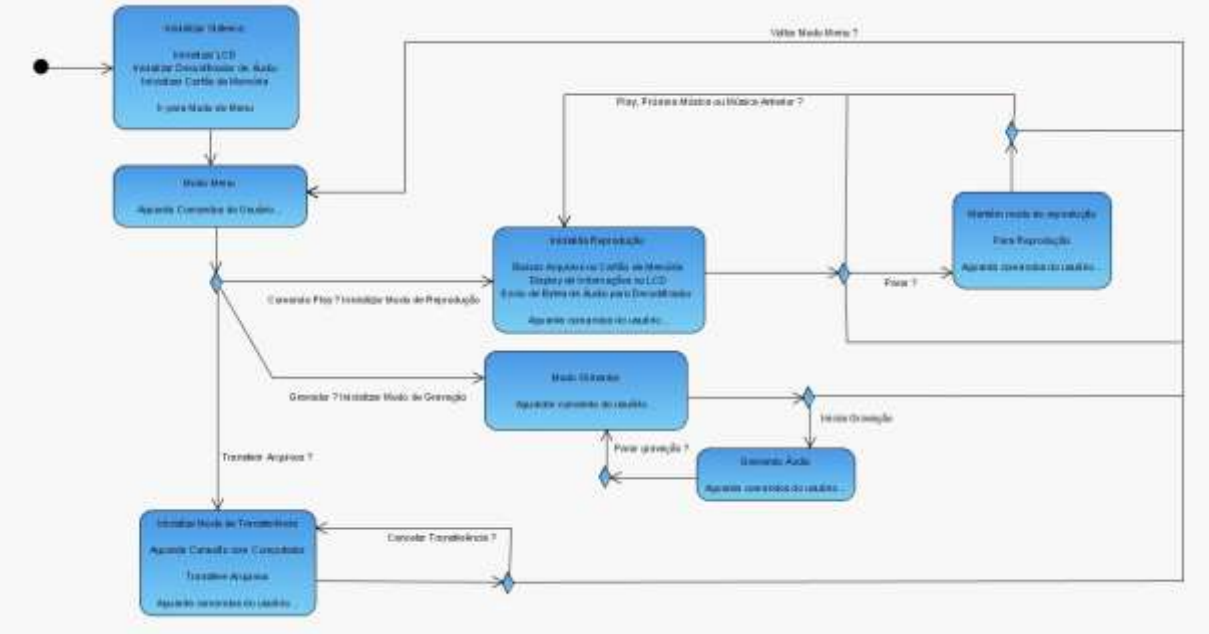


Figura 8 - Fluxograma do projeto do *firmware*

4.3 Projeto de *Software*

O software foi desenvolvido utilizando-se técnicas de orientação a objetos e sua especificação foi feita com base na UML.

Todas as tarefas relacionadas ao software são realizadas utilizando-se de arquivos armazenados no sistema de arquivos do computador, não existindo qualquer tipo de armazenamento em banco de dados. Portanto, não se aplica a especificação de um modelo relacional.

4.3.1 Diagrama de Casos de Uso

A figura 9 apresenta o diagrama de casos de uso do software.

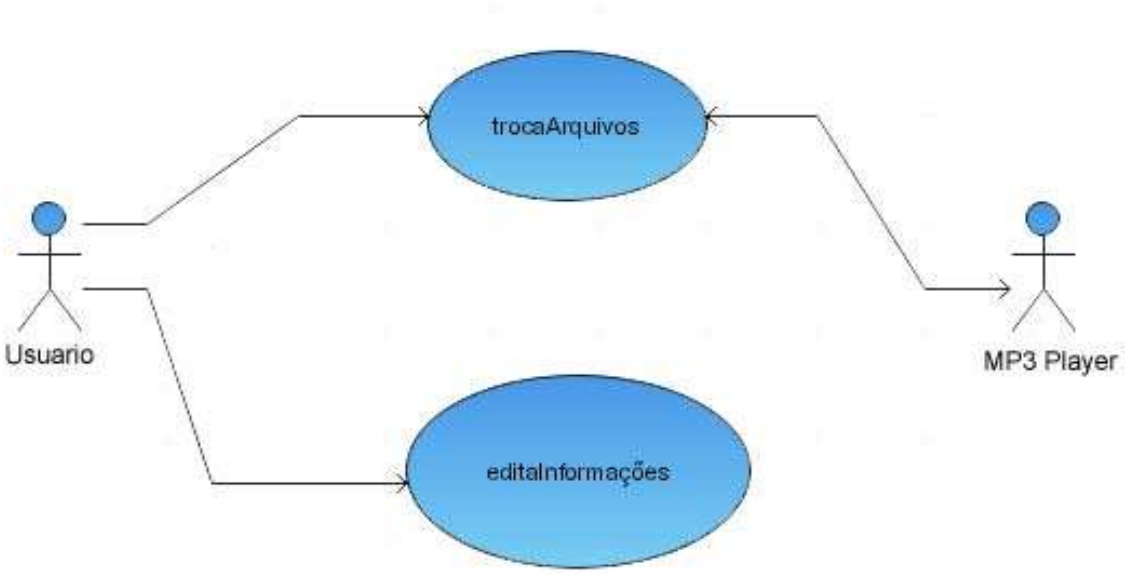


Figura 9 - Diagrama de Casos de Uso

4.3.1.1 Caso de Uso: trocaArquivos

Objetivo: Trocar arquivos entre o dispositivo MP3 Player e o computador do usuário do dispositivo.

Ator: Usuário e MP3 Player.

Descrição: Este caso de uso se inicia quando o usuário conecta o computador ao dispositivo MP3 Player e inicia a seleção de arquivos do computador e/ou do MP3 Player para realizar a troca de arquivos entre eles.

Passos:

- O Usuário deve inicializar a conexão entre o computador e o dispositivo MP3 Player;
- O usuário deve selecionar os arquivos do computador a serem enviados para o dispositivo;
- O usuário deve selecionar os arquivos do dispositivo a serem enviados para o computador;
- O usuário deve acionar a função de troca de arquivos entre o computador e o dispositivo.

4.3.1.2 Caso de Uso: editaInformações

Objetivo: Editar informações da música, armazenadas na tag ID3 do arquivo mp3.

Ator: Usuário.

Descrição: Este caso de uso se inicia durante a seleção de arquivos para transferência e acionamento da função de edição de informações da música no software.

Passos:

- O usuário deve selecionar o arquivo do tipo MP3 e acionar a função de edição de informações da música;
- O usuário deve alterar as informações desejadas;
- O usuário deve gravar as alterações realizadas.

4.3.2 Diagramas de Seqüência

A figura 10 apresenta o diagrama de seqüência do caso de uso trocaArquivos.

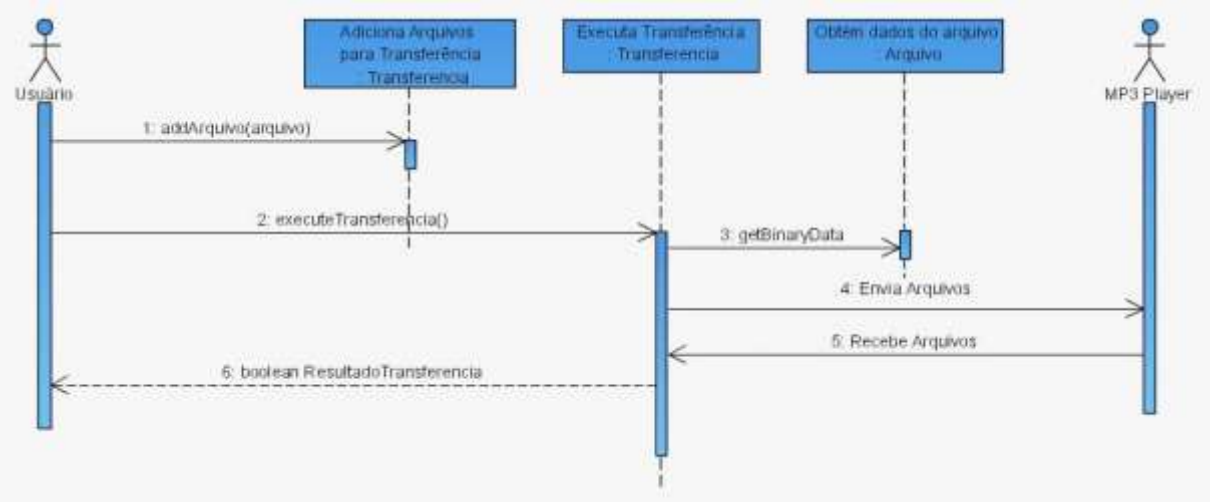


Figura 10 - Diagrama de seqüência do caso de uso trocaArquivos

A figura 11 apresenta o diagrama de seqüência do caso de uso editaInformações.

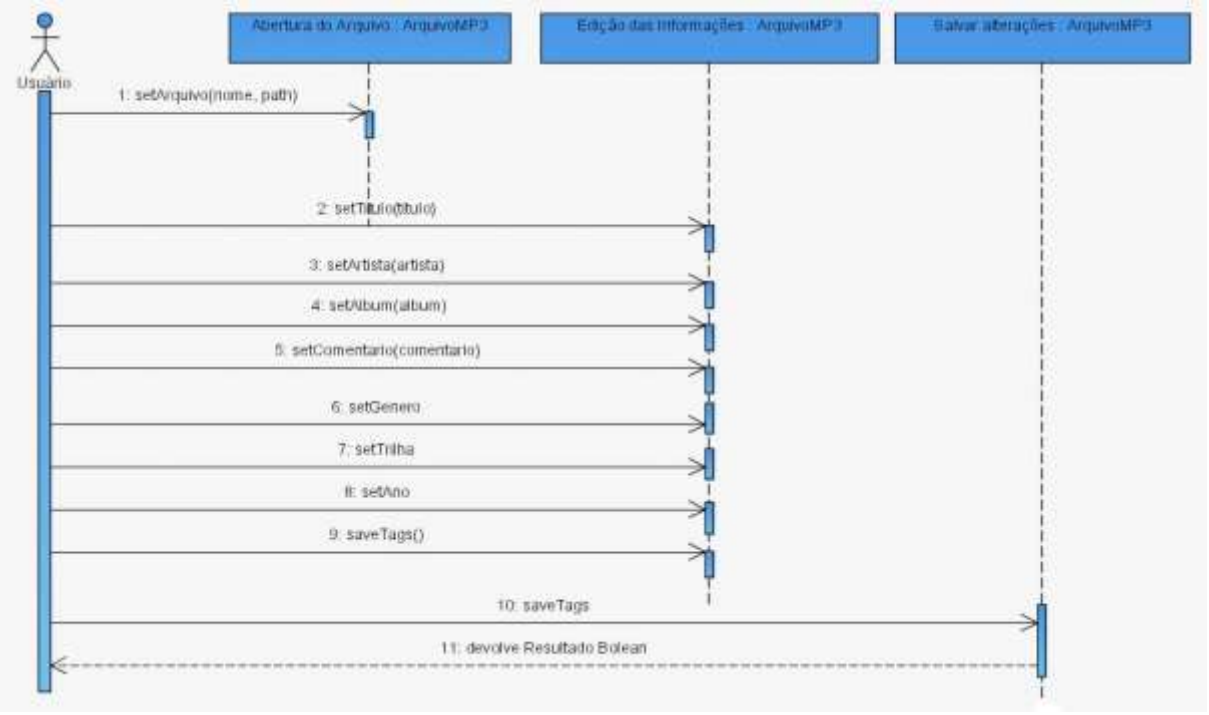


Figura 11 - Diagrama de seqüência do caso de uso editaInformações

4.3.3 Diagrama de Classes

A figura 12 apresenta o diagrama de classes do projeto de *software*.

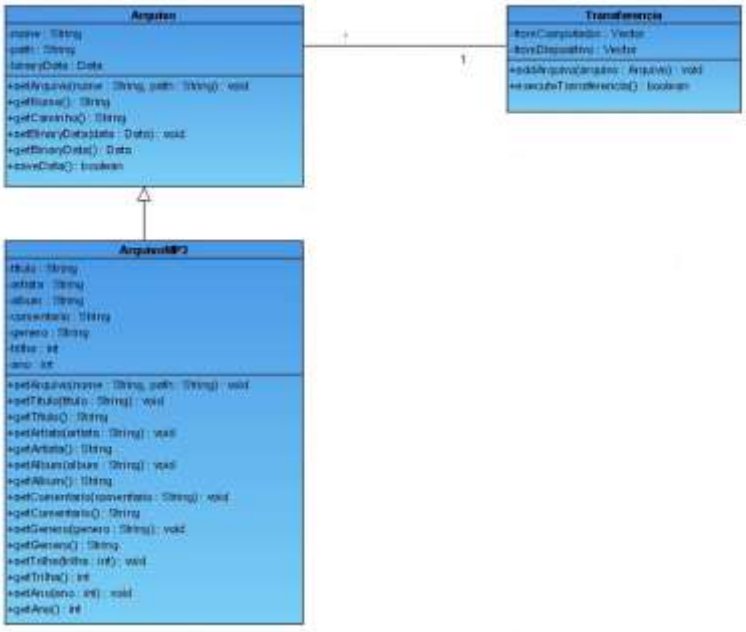


Figura 12 - Diagrama de Classes do projeto de *software*

4.3.3.1 Classe: Arquivo

A classe arquivo representa um arquivo armazenado no sistema de arquivos do computador.

Nome da Operação: **setArquivo**

Responsabilidade: Atribui à classe o arquivo arquivo armazenado no sistema de arquivos.

Pré-Condiçao: Devem ser informados o nome do arquivo e o caminho para o mesmo no sistema de arquivos e o arquivo deve existir no sistema de arquivos.

Pós-Condição: Arquivo é atribuído à classe.

Exceção: Caso o arquivo não exista é retornada uma mensagem de exceção.

Nome da Operação: **getNome**

Responsabilidade: Obtém o caminho do arquivo atribuído à classe.

Pré-Condição: Deve ter sido atribuído um arquivo à classe.

Pós-Condição: É retornado o caminho do arquivo atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getCaminho**

Responsabilidade: Obtém o nome do arquivo atribuído à classe.

Pré-Condição: Deve ter sido atribuído um arquivo à classe.

Pós-Condição: É retornado o nome do arquivo atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo à classe, é retornada uma mensagem de exceção.

Nome da Operação: **setBinaryData**

Responsabilidade: Define os dados binários armazenados no arquivo.

Pré-Condição: Deve ter sido atribuído um arquivo à classe.

Pós-Condição: São definidos os dados binários armazenados no arquivo.

Exceção: Caso não tenha sido atribuído um arquivo à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getBinaryData**

Responsabilidade: Obtém os dados binários armazenados no arquivo.

Pré-Condição: Deve ter sido atribuído um arquivo à classe.

Pós-Condição: São retornados os dados binários armazenados no arquivo.

Exceção: Caso não tenha sido atribuído um arquivo à classe, é retornada uma mensagem de exceção.

Nome da Operação: **saveData**

Responsabilidade: Armazenar os dados binários definidos no arquivo.

Pré-Condição: Deve ter sido atribuído um arquivo à classe.

Pós-Condição: São armazenados os dados do arquivo no sistema de arquivos.

Exceção: Caso não tenha sido atribuído um arquivo à classe, é retornada uma mensagem de exceção.

4.3.3.2 Classe: ArquivoMP3

A classe arquivo mp3 é uma extensão da classe arquivo e representa um arquivo do tipo MP3 armazenado no sistema de arquivos do computador.

Nome da Operação: **setArquivo**

Responsabilidade: Atribui à classe o arquivo do tipo MP3 armazenado no sistema de arquivos.

Pré-Condição: Devem ser informados o nome do arquivo e o caminho para o mesmo no sistema de arquivos, o arquivo deve existir no sistema de arquivos e ser do tipo MP3.

Pós-Condição: Arquivo MP3 é atribuído à classe.

Exceção: Caso o arquivo não exista é retornada uma mensagem de exceção.

Nome da Operação: **setTitulo**

Responsabilidade: Define o título da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É atribuído o título à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getTitulo**

Responsabilidade: Obtém o título da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É retornado o título atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **setArtista**

Responsabilidade: Define o artista da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É atribuído o artista à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getArtista**

Responsabilidade: Obtém o artista da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É retornado o artista atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **setAlbum**

Responsabilidade: Define o álbum da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É atribuído o álbum à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getAlbum**

Responsabilidade: Obtém o álbum da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É retornado o álbum atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **setComentario**

Responsabilidade: Define o comentário da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É atribuído o comentário à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getComentario**

Responsabilidade: Obtém o comentário da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É retornado o comentário atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **setGenero**

Responsabilidade: Define o gênero da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É atribuído o gênero à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getGenero**

Responsabilidade: Obtém o gênero da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É retornado o gênero atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **setTrilha**

Responsabilidade: Define o número da trilha da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É atribuído o número da trilha à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getTrilha**

Responsabilidade: Obtém o número da trilha da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É retornado o número da trilha atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **setAno**

Responsabilidade: Define o ano da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É atribuído o ano à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **getAno**

Responsabilidade: Obtém o ano da música MP3 armazenada no arquivo.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: É retornado o ano atribuído à classe.

Exceção: Caso não tenha sido atribuído um arquivo mp3 à classe, é retornada uma mensagem de exceção.

Nome da Operação: **saveTags**

Responsabilidade: Armazena no arquivo atribuído à classe as tags alteradas pelo usuário.

Pré-Condição: Deve ter sido atribuído arquivo do tipo mp3 à classe.

Pós-Condição: São armazenadas no arquivo as informações da música.

Exceção: Caso não tenha sido atribuído um arquivo mp3 ã classe, é retornada uma mensagem de exceção.

4.3.3.3 Classe: Transferência

A classe transferência representa uma troca de arquivos entre o dispositivo e o computador.

Nome da Operação: **addArquivo**

Responsabilidade: Adiciona um arquivo para transferência.

Pré-Condição: Deve ser informado o arquivo.

Pós-Condição: É adicionada um arquivo para transferência.

Exceção: Caso a classe arquivo não tenha sido construída, é retornada uma mensagem de exceção.

Nome da Operação: **executeTransferencia**

Responsabilidade: Realiza a transferência dos arquivos entre o computador e o dispositivo MP3 Player.

Pré-Condição: Devem ter sido adicionado pelo menos um arquivo para transferência.

Pós-Condição: É realizada a transferência dos arquivos.

Exceção: Caso não tenham sido atribuídos arquivos à transferência, é retornada uma mensagem de exceção.

5 RESULTADOS

Todo o projeto de hardware foi implementado em placa de protótipo. Foram desenvolvidos e testados com sucesso os módulos de microprocessamento, armazenamento, decodificação, controle e display de informações. A implementação do módulo de comunicação que estava prevista para ser USB foi modificada para comunicação SERIAL por dificuldades em um comportamento estável do chip TUSB da Texas Instruments.

Todo o projeto de firmware foi desenvolvido utilizando-se linguagem C para microcontroladores ATMEL. Para compilação e desenvolvimento do código foi utilizada o WinAVR, um conjunto de ferramentas de desenvolvimento open source para microcontroladores Atmel AVR 8bit RISC.

No firmware foram implementados os algoritmos baixo nível para interface com o controlador de LCD KS0108, chip decodificador de MP3 VS1001k, cartão de memória Compact Flash. Ainda como algoritmos de baixo nível, foram implementados os algoritmos de padrão de formatação FAT16, o controle de interrupções vindos da unidade de controle e o algoritmo de comunicação serial.

Em um nível mais alto foram implementados os algoritmos para seleção de músicas, onde o usuário interage com o dispositivo através dos botões de controle para seleção dos arquivos de áudio para reprodução, e controle de reprodução MP3, onde o usuário pode realizar os comandos de volume, pausar, reproduzir, ir para próxima música, ir para música anterior e parar reprodução.

O algoritmo para reconhecimento do padrão de formatação FAT16 apresentou inconsistências, por este motivo não foi viável implementar o módulo de troca de arquivos. Desta maneira, o hardware de comunicação serial foi implementado apenas com funções de DEBUG.

Todo o projeto de software foi desenvolvido utilizando-se linguagem JAVA JDK1.4.2 para sistema operacional Windows, para compilação e desenvolvimento do código foi utilizada a IDE Eclipse para desenvolvimento de software e para desenvolvimento da Interface Gráfica foi utilizada a API SWT.

No software foram implementadas funcionalidades para seleção de arquivos do microcomputador, seleção de arquivos do cartão de memória, área de transferência de arquivos, exclusão de arquivos do microcomputador, exclusão de arquivos do cartão de memória, edição de informações ID3 dos arquivos MP3 localizados no computador e troca de arquivos entre o computador e o cartão de memória.

As funcionalidades de obtenção de lista de arquivos do dispositivo e troca de arquivos entre o PC e o dispositivo, não foram implementadas por motivos de inconsistência na implementação FAT16 do firmware do dispositivo, como forma alternativa desta funcionalidade foi implementada a função de troca de arquivos entre o cartão e o microcontrolador através de um leitor de cartões.

A figura 13 apresenta a foto do projeto de hardware.

Figura 13 - Foto do projeto de *hardware*

A figura 14 apresenta a tela do software desenvolvido para troca de arquivos entre o microcomputador e o dispositivo.

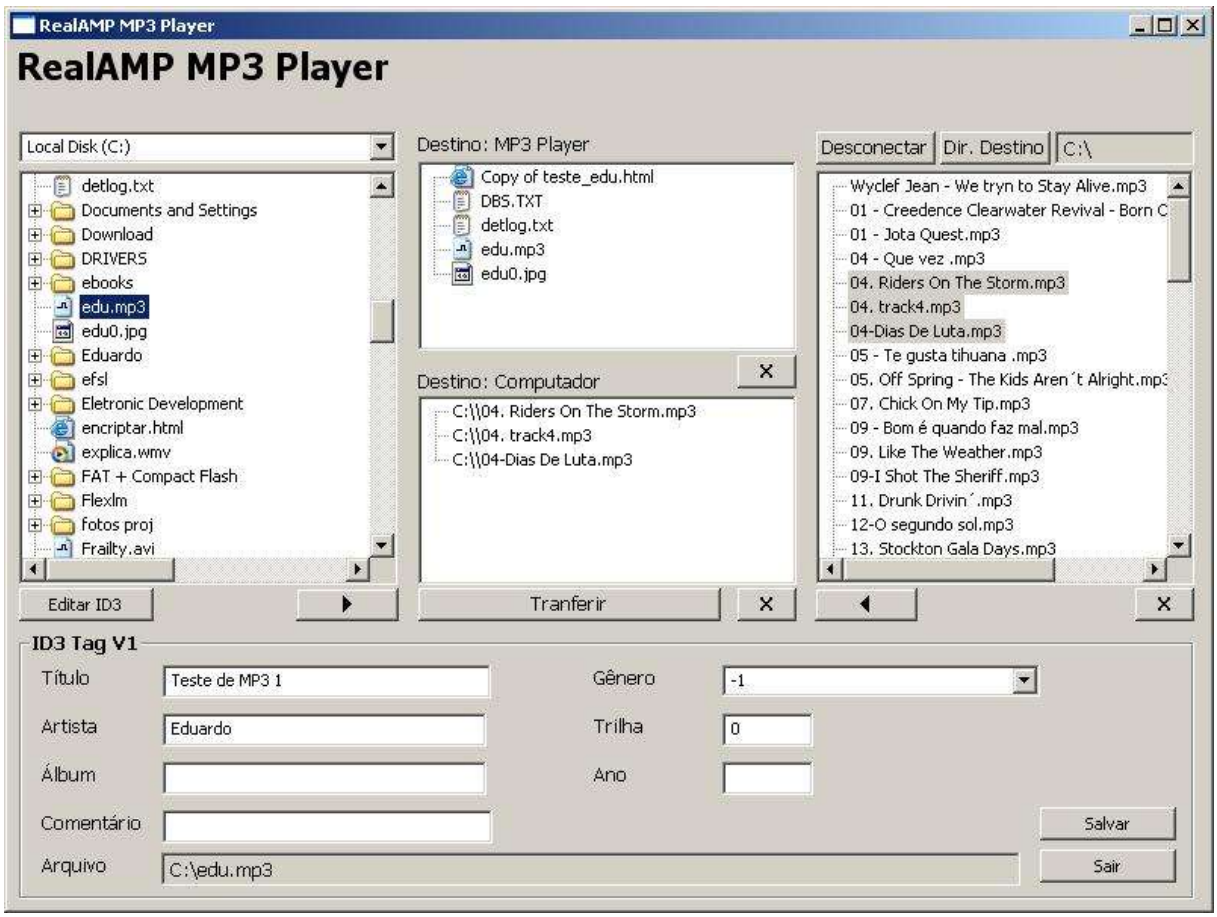


Figura 14 - Tela do *software* desenvolvido para troca de arquivos entre o microcomputador e o dispositivo

6 CONCLUSÃO

A utilização do microcontrolador AVR Atmel em conjunto com desenvolvimento em linguagem C no projeto, mostrou-se muito eficiente comparando-se com testes realizados com microcontroladores PIC, o que foi crucial para o bom andamento do projeto.

A utilização do cartão de memória Compact Flash mostrou-se de baixo custo para desenvolvimento de aparelhos digitais portáteis, mas mostrou-se também difícil pela quantidade de pinos para interface com o microcontrolador e pouco portátil se comparado com o tamanho dos cartões disponíveis atualmente, para atualizações futuras do projeto tem se idéia de utilizar como alternativa, para utilização do cartão de memória Compact Flash, a utilização de um cartão do tipo Security Digital.

O interfaceamento com o decodificador MP3 VS1001K se mostrou de baixo custo e fácil, por possuir um DA interno e realizar toda tarefa de conversão DA automaticamente.

A utilização do display se mostrou eficiente para um protótipo, mas para um produto comercial é necessário utilizar um display menor, para tornar o dispositivo portátil.

A maior dificuldade do projeto foi implementar no firmware um algoritmo consistente do padrão de formatação FAT16. Este é um algoritmo complexo e sua documentação é complicada. Em um futuro melhoramento do projeto este algoritmo deve ser melhorado de forma a também suportar os outros diversos padrões de formatação FAT.

Com a implementação do projeto foi possível verificar a viabilidade da implementação de um dispositivo MP3 Player comercial, sendo necessárias apenas algumas alterações no projeto inicial. Como substituição do LCD e Cartão de Memória por componentes menores para viabilizar a produção de um dispositivo portátil. Outros pontos onde o projeto poderia ser melhorado seriam a substituição do decodificador MP3 por um outro decodificador da mesma família, mas com mais funcionalidades e a substituição da comunicação serial por comunicação USB. E, principalmente, vê-se a necessidade de implementação de um algoritmo FAT consistente e genérico o suficiente para que o dispositivo suporte os diversos padrões de formatação FAT utilizados pelos usuários de cartões de memória.

7 REFERÊNCIAS BIBLIOGRÁFICAS

COMPACT FLASH ASSOCIATION. "***Information about Compact Flash***".
CFA - CompactFlash Association. (http://www.compactflash.org/info/cfinfo.htm), (11/02/2004) (A).

COMPACT FLASH ASSOCIATION. "***CF+ and CompactFlash Specification Revision 3.0***".
CFA - CompactFlash Association. (http://www.compactflash.org/specdl1.htm), (23/12/2004) (B).

COMPACT FLASH ASSOCIATION. "***CompactFlash Association, CF and CompactFlash Frequently Asked Questions***".
CFA - CompactFlash Association. (http://www.compactflash.org/faqs/faq.htm), (07/09/2004) (C).

CURTIN, D. P., "***Curso Rápido de Fotografia Digital***".
Imagem Digital. (http://www.imagem-digital.com/contents-f.htm), Trad. de Augusto Cavalcante,(14/05/2001).

MICROSOFT CORPORATION. "***FAT: General Overview of On-Disk Format***".
Microsoft Corporation. (http://www.microsoft.com/whdc/default.mspx), (06/12/2000).

ID3 ORG. "***The short history of tagging***". ID3 Org.
(http://www.id3.org/history.html), s.d.

JONES, C. "***MP3 Overview***".
(http://hotwired.lycos.com/webmonkey/00/31/index3a.html), (27/07/2000).

FRAUNHOFER. "***Audio & Multimedia MPEG Audio Layer-3***".
FRAUNHOFER – Institut Integrierte Schaltungen.
(http://www.iis.fraunhofer.de/amm/techinf/layer3/index.html), s.d.

SILVA, D. M. "**Guia de Consulta Rápida - UML**". Novatec - São Paulo, 2001.