Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

MARCELO AIRES MOREIRA

DESENVOLVIMENTO DE UM SISTEMA EMBARCADO PARA REPRODUÇÃO DE ÁUDIO DIGITAL CONTROLADO LOCAL E REMOTAMENTE

Campina Grande, Paraíba
Abril de 2015

MARCELO AIRES MOREIRA

DESENVOLVIMENTO DE UM SISTEMA EMBARCADO PARA REPRODUÇÃO DE ÁUDIO DIGITAL CONTROLADO LOCAL E REMOTAMENTE

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrônica, Sistemas Embarcados, Áudio Digital.

Orientador:
Antonio Marcus Nogueira Lima, PhD

Campina Grande, Paraíba
Abril de 2015

MARCELO AIRES MOREIRA

DESENVOLVIMENTO DE UM SISTEMA EMBARCADO PARA REPRODUÇÃO DE ÁUDIO DIGITAL CONTROLADO LOCAL E REMOTAMENTE

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrônica, Sistemas Embarcados, Áudio Digital.

Aprovado em: ___/__ /___

_____
AVALIADOR

_____
ORIENTADOR
Antonio Marcus Nogueira Lima, PhD

Campina Grande, Paraíba
Abril de 2015

Dedico este trabalho à minha família, por estar sempre ao meu lado, do meu primeiro ao último dia de realização do curso de Engenharia Elétrica.

AGRADECIMENTOS

Agradeço primeiramente a Deus, criador do universo e de tudo que existe, pelo seu Amor, sua Misericórdia e Graça.

À minha família, que esteve sempre junto comigo ajudando a superar as dificuldades e desafios enfrentados durante o curso.

Ao professor Antonio Marcus, por sempre estar disponível e disposto a orientar o desenvolvimento deste trabalho.

Aos técnicos Lúcio e Emanuel, que me ajudaram na execução de algumas etapas de projeto.

Aos colegas do laboratório e-Robótica, por compartilharem experiências acadêmicas e práticas ao longo do curso.

Aos meus amigos, os quais proporcionaram momentos de descontração e lazer.

E por último, agraço ao coordenador, professores e funcionários do curso de graduação de Engenharia Elétrica, os quais contribuíram para minha formação como aluno.

"A sabedoria se acha entre os idosos? A vida longa traz entendimento? Deus é quem tem sabedoria e poder; a Ele pertencem o conselho e o entendimento. O que Ele derruba não se pode reconstruir; quem Ele aprisiona ninguém pode libertar."

Jó 12:12-14.

RESUMO

Esse projeto trata do desenvolvimento e montagem de um sistema embarcado capaz de executar áudio em formato digital. O sistema é dotado de um Controle Local e um Controle Remoto via Bluetooth e dispõe de uma memória de armazenamento do tipo disco rígido. A capacidade de armazenamento do MP3 player é de 8GB e ele é capaz de reproduzir áudio digital nos formatos .MP3 e .WAV. Para o desenvolvimento de hardware, utilizou-se componentes eletrônicos analógicos e digitais integrados entre si por meio de um microcontrolador Arduino Micro, uma protoboard e Placas de Circuito Impresso (PCB's). Para o desenvolvimento de Software, utilizou-se as linguagens de programação C e Objective C, e os softwares P-CAD, Xcode e SketchUp.

Palavras-chave: Sistema Embarcado; Áudio em formato digital; Controle Local; Controle Remoto; Arduino Micro; PCB's; Linguagens de Programação C e Objective C; P-CAD; Xcode; SketchUp.

ABSTRACT

This Project is about the development and set up of an Embedded System that is able to execute audio in digital format. The system has two different control modes: Local and Remote via Bluetooth and has an external data memory integrated to its hardware. The device storage capability is 8GB and it is able to reproduce .MP3 and .WAV files. For Hardware development, it was used analog and digital electronic components integrated together using an Arduino Micro microcontroller, a breadboard and Printed Circuit Boards (PCB's). For Software development it was used C and Objective C programming languages and P-CAD, Xcode and SketchUp softwares.

Key Words: Embedded System; Audio in digital format; Different control modes; External data memory. Arduino Micro; PCB's; C and Objective C programming languages; P-CAD; Xcode; SketchUp.

SUMÁRIO

Índice de Figuras x

Índice de Tabelas xii

1. Introdução 1

2. Fundamentação Teórica 4

2.1. Áudio Digital 4

2.2. Codificação de Áudio Digital 6

2.2.1. Técnicas de Codificação de Áudio Digital 6

2.2.2. Codificador de Áudio Padrão 7

2.3. Formatos de Áudio Digital 8

2.3.1. Evolução dos Formatos de Áudio Digital 8

2.3.2. Classificação e aplicações de diferentes formatos de Áudio Digital 10

2.3.3. Formato de Áudio MP3 11

2.4. MP3 Players 12

2.4.1. Cenários de Uso 13

2.4.2. Importância 14

3. Justificativa 15

4. Objetivos 16

4.1. Objetivo Geral 16

4.2. Objetivos Específicos 16

5. Metodologia 18

6. Execução e Resultados 19

6.1. Protótipo de Papel 19

6.2. Diagrama de Verplank 20

6.3. Montagem e teste de um MP3 básico numa protobobard 22

6.3.1. Hardware 22

6.3.2. Software 24

6.4. Desenvolvimento de um Controle Local para o dispositivo 27

6.4.1. Hardware 27

6.4.2. Software 28

6.5. Desenvolvimento de um Controle Remoto para o dispositivo 32

6.5.1. Hardware 32

6.5.2. Software 34

6.6. Desenvolvimento de uma Interface de Comunicação Intuitiva 35

6.7. Reprodução do circuito numa PCB 38

6.8. Projeto de uma estrutura de transporte para as PCB'S 48

6.9. Correção de Software 49

7. Resultado Final 50

8. Conclusão 53

9. Trabalhos futuros 55

9. Referências 56

Anexo 1 58

Índice de Figuras

Figura 1 - Fonógrafo [4] 2
Figura 2 - Gramofone [4] 2
Figura 3 - Disco de vinil [5] 3
Figura 4 - Walkman [5] 3
Figura 5 - Ipod Classic [6]. 3
Figura 6 - Exemplos dos processos de amostragem, quantificação e codificação [7]. . 5
Figura 7 - Codificador de Áudio Padrão[8]. 7
Figura 8 - Evolução dos codificadores de áudio entre 1977 e os dias atuais [8]. 8
Figura 9 - Protótipo de Papel do MP3 Player, mostrando o detalhamento de seu funcionamento [14]. 19
Figura 10 - Diagrama de Verplank do MP3 Player [15]. 21
Figura 11 - Arduino Micro 22
Figura 12 - MP3 and Midi Breakout Board for VS1053. 22
Figura 13 - CI HEF4050BP. 22
Figura 14 - Conector Audio Jack TRS 3.5mm. 23
Figura 15 - Micro SD Card 8GB. 23
Figura 16 - Cabo USB Padrão. 23
Figura 17 - Interface de comunicação com o Arduino. 26
Figura 18 - Esquema de montagem dos botões switch. 27
Figura 19 - Bluetooth RF Transceiver Module serial RS232 TTL HC-05. 33
Figura 20 - Circuito completo do MP3 Player montado em protoboard. 33
Figura 21 - Sofware Xcode mostrando o arquivo .xib. 35
Figura 22 - Sofware Xcode mostrando o arquivo .m. 37
Figura 23 - Interface virtual para comunicação entre o MacBook e o MP3 Player. 37
Figura 24 - Esquemático do circuito do MP3 Player Completo. 38
Figura 25 - Componente Arduino no P-CAD. 39
Figura 26 - Componente Módulo de Bluetooth no P-CAD. 39
Figura 27 - Componente VS1053 no P-CAD 39
Figura 28 - Componente Conector Audio Jack no P-CAD. 39
Figura 29 - Componente HEX4050BP no P-CAD. 39
Figura 30 - Componente SDcard no P-CAD. 39
Figura 31 - Componente Switch Grande no P-CAD. 39
Figura 32 - Componente Switch Pequeno no P-CAD. 39
Figura 33 - Componente Resistor no P-CAD. 40
Figura 34 - Conexões da PCB 1. 40
Figura 35 - Posicionamento dos componentes da PCB 1. 41
Figura 36 - Etapa 1 do roteamento da PCB 1 41
Figura 37 - Etapa 2 do roteamento da PCB 1. 42
Figura 38 Etapa 3 do roteamento da PCB 1. 42
Figura 39 - Etapa 4 do roteamento da PCB 1. 43

Figura 40 Delimitação da PCB 1. 43
Figura 41: Etapa 1 do desenho de layout da PCB 2. 44
Figura 42 - Etapa 2 do desenho de layout da PCB 2. 44
Figura 43 Etapa 2 do desenho de layout da PCB 2. 45
Figura 44 - Face frontal da PCB1 fabricada. 45
Figura 45 Face traseira da PCB1 fabricada. 46
Figura 46 - Face frontal da PCB2 fabricada. 46
Figura 47 - Face traseira da PCB2 fabricada. 47
Figura 48 - Resultado da acoplamento das PCB's 1 e 2. 47
Figura 49 - Visualização da Caixa 3D de transporte do MP3 Player no SketchUp. 48
Figura 50 – Resultado Final do MP3 Player. 50
Figura 51 - Controle remoto do dispositivo. 51

Índice de Tabelas

Tabela 1 - Principais formatos de áudio utilizados no mundo [1]..... 11
Tabela 2 - Informações gerais sobre os players pioneiros [9]. 13

1. INTRODUÇÃO

A capacidade do ser humano de ouvir desempenha um papel importante na percepção e assimilação de uma enorme quantidade de informações acerca do ambiente que o rodeia [1].

O som é a propagação de uma frente de compressão mecânica ou onda mecânica; esta onda propaga-se de forma circuncêntrica tridimensional, a partir de meios materiais possuidores de massa e elasticidade, como os sólidos, líquidos ou gasosos. Segundo [2], o som afeta-nos em quatro aspectos: fisiologicamente, psicologicamente, cognitivamente e ao nível comportamental. Como tal, a qualidade de vida e saúde são diretamente afetadas pelos sons a que somos submetidos. Desse modo, o homem sempre se dedicou a inventar sistemas que permitissem a reprodução e a amplificação da capacidade de ouvir [1].

Ate 1877, o registro ou gravação do som era impraticável. Todos os sons tinham que ser produzidos no momento e no local, para que pudessem ser ouvidos. Em 1849, porem, Innocenzo Manzetti criou o telégrafo, o qual representou o primeiro sistema capaz de receber ondas sonoras e reproduzi-las em sua saída. Mas foi somente em 1877 que Thomas Edison criou o primeiro dispositivo com capacidade para armazenar informação (auditiva) sonora, de forma a permitir uma reprodução posterior, através do dispositivo chamado de "*phonograph*". A era do áudio analógico começara [2].

Porém, a variedade de sistemas e serviços relacionados com as comunicações audiovisuais só veio ganhar particular expressão no final do século XX, durante a explosão da chamada "era digital", a partir da flexibilização e maior disponibilidade dos recursos tecnológicos relacionados com a aquisição, processamento, armazenamento e transmissão da informação audiovisual [1]. Durante esse período foram criados os primeiros ***MP3 Players*** da história da humanidade, os quais possuíam uma capacidade de armazenamento muito limitada e grande tamanho físico. Mais tarde surgiram novas tecnologias de compressão capazes de proporcionar uma redução considerável no tamanho do dispositivo, bem como redução do ruído de codificação, de modo que os MP3 Players foram capazes de impactar substancialmente a vida das pessoas.

Outro fator responsável pela diminuição do espaço físico dos dispositivos eletrônicos foi o desenvolvimento da microeletrônica, sobretudo nos anos 1959 e 1962,

com o descobrimento do efeito transistor e a criação dos primeiro Circuitos Integrados (CI's) comerciais, respectivamente [3]. Os CI's são capazes de integrar inúmeros componentes eletrônicos analógicos e digitais sobre um espaço físico de escala nanométrica.

A evolução dos dispositivos de reprodução e armazenamento de áudio entre os anos 1877 e 2009 pode ser visualizada abaixo:

- Fonógrafo – 1877, ilustrado na Figura 1.
- Gramofone – 1888, ilustrado na Figura 2.
- Fita Magnética – 1920
- Disco de Vinil – 1940, ilustrado na Figura 3.
- Fita Cassete – 1970
- Walkman – 1979, ilustrado na Figura 4.
- CD - 1979
- Discman – 1984
- MP3 Player – 1997
- Ipod Classic/ Samsung SPH-M100 – 2001, ilustrado na Figura 5.
- Ipod Nano/ Nokia N91 – 2005
- Ipod touch/ Iphone – 2007
- Samsung Galaxy – 2009

Figura 1 - Fonógrafo [4]

Figura 2 - Gramofone [4]

Figura 3 - Disco de vinil [5]

Figura 4 - Walkman [5]

Figura 5 - Ipod Classic [6].

Desse modo, considerando a importância tecnológica e o impacto social promovido pela criação do MP3 Player, este projeto visa desenvolver um sistema embarcado que possa funcionar como um MP3 player dotado de um controle de dados via Bluetooth, a partir da integração de diferentes componentes eletrônicos analógicos e digitais por meio de um microcontrolador.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. ÁUDIO DIGITAL

Da maneira como aparece na natureza, o som representa um sinal analógico de tempo contínuo, uma vez que sua amplitude pode assumir valores contínuos no tempo, e o sinal está definido para qualquer valor de tempo. Após captado na natureza, o sinal analógico geralmente é convertido para um sinal digital de tempo discreto, por um uma serie de razões, incluindo [7]:

- Sinais digitais são menos sensíveis a ruído externo;

- Sinais digitais podem ser processados com auxílio de um computador;

- Armazenamento de informações digitais geralmente é mais barato;

- Sistemas digitais dissipam menos potência;

- Sistemas digitais são mais robustos a erros de transmissão [2];

- Sinais digitais podem ser compactados.

Sinais digitais de tempo discreto são sinais em que a amplitude assume apenas valores discretos, estando o sinal definido apenas para valores discretos de t. O processo de digitalização de um sinal analógico geralmente envolve 3 etapas:

- Amostragem: É a etapa responsável por converter o sinal de tempo continuo em um sinal de tempo discreto;
- Quantificação: É a etapa responsável por condicionar a amplitude do sinal a assumir apenas valores discretos;
- Codificação: Essa etapa tem a função de atribuir uma nova representação ao sinal de modo a diminuir o espaço físico necessário para seu armazenamento.

A Figura 6 apresenta um exemplo dos processos de amostragem, codificação e quantização de sinais digitais.

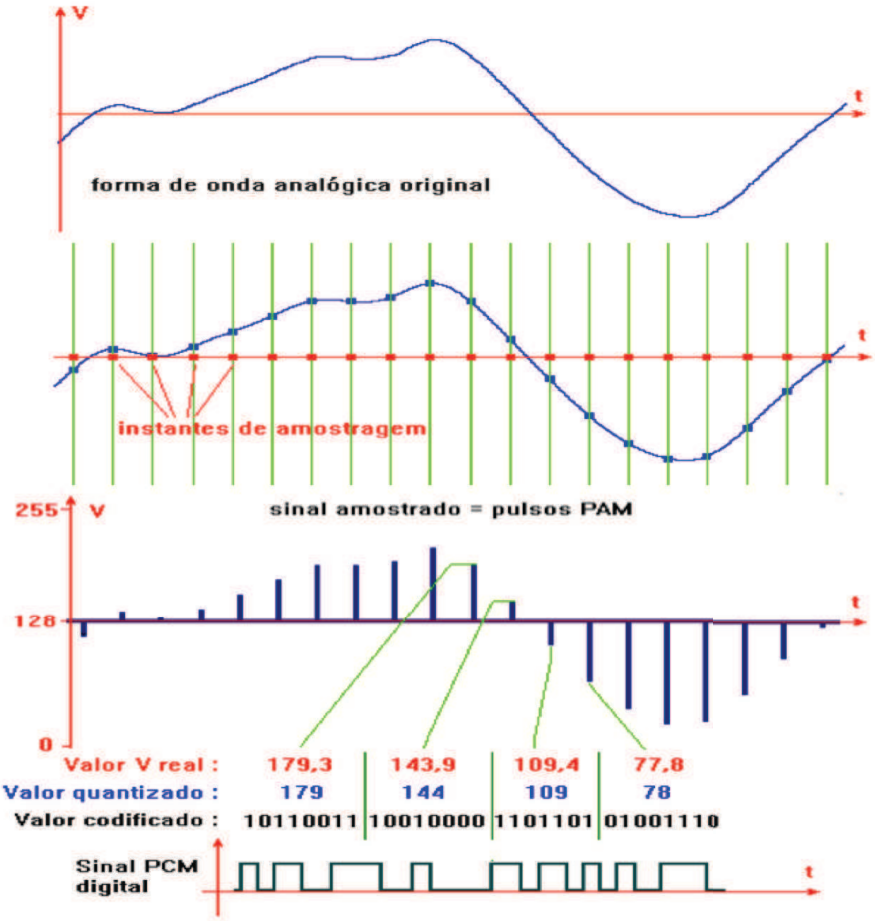


Figura 6 - Exemplos dos processos de amostragem, quantificação e codificação [7].

Percebe-se que inicialmente o sinal é tomado apenas em um número limitado de amostras, posteriormente ele é quantizado com 8 bits, de modo que sua amplitude terá 2^8 níveis, variando apenas entre 0 e 255. Por último, o sinal é codificado utilizando-se a codificação binária.

Aqui dar-se-á enfoque à etapa de codificação, uma vez que esta é a etapa que se encontra relacionada com a classificação dos formatos de áudio digital (i.e. WAVE, MP3, Mp2, AC2).

2.2. CODIFICAÇÃO DE ÁUDIO DIGITAL

A importância da codificação dá-se devido ao fato de que o áudio digital de alta qualidade requer uma grande capacidade de memória para seu armazenamento. Torna-se então necessário a utilização de técnicas de codificação ou compressão de áudio digital, de modo a reduzir o espaço físico necessário para seu armazenamento [1]. A seguir serão apresentadas as principais técnicas de codificação de áudio digital, assim como o codificador de áudio padrão.

2.2.1. TÉCNICAS DE CODIFICAÇÃO DE ÁUDIO DIGITAL

A codificação de áudio digital se baseia em diferentes técnicas. As principais técnicas utilizadas são a Codificação da Fonte, Codificação do Canal, Resiliência a erros e Ocultação de erros.

A Codificação da Fonte tem o objetivo de eliminar ou atenuar características redundantes do sinal original de modo a acarretar o menor número de perdas possível e pode ser obtida por meio de 4 métodos diferentes:

- Redundância Estatística: atribuição de palavras mais curtas a amostras repetidas no domínio do tempo.
- Redundância Temporal: redução do tempo utilizado para codificar amostras repetidas.
- Redundância Espectral: atribuição de palavras mais curtas a amostras repetidas no domínio da frequência.
- Redundância Perceptual: Eliminação de amostras imperceptíveis ao ouvido humano.

A codificação do Canal visa minimizar a ocorrência de erros durante a transmissão do sinal digital. A técnica de Resiliência a erros, por sua vez, tem objetivo de minimizar o impacto dos erros no sinal digital após a sua decodificação. Por fim, a técnica de Ocultação de erros tem o objetivo de minimizar o impacto dos erros na qualidade percebida do sinal [1].

2.2.2. CODIFICADOR DE ÁUDIO PADRÃO

O sinal de representação mais básico é o PCM (*Pulse Code Modulation*) que é um sinal binário obtido por amostragem, quantificação e codificação usando sempre o mesmo número de bits por amostra. O processo de codificação de áudio parte do sinal PCM e envolve 5 etapas. O diagrama de blocos de um Codificador de Áudio Padrão pode ser visualizado na Figura 7:

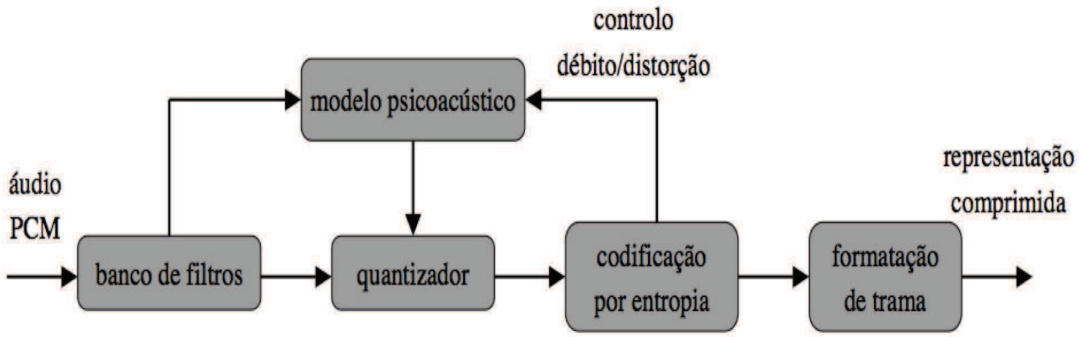


Figura 7 - Codificador de Áudio Padrão[8].

O banco de filtros é responsável por transformar o sinal do domínio do tempo para o domínio da frequência e posteriormente agrupar o sinal em um conjunto de bandas de frequência. O modelo psicoacústico analisa as bandas de frequência e mascara as frequências que são imperceptíveis ao ouvido humano. O quantificador condiciona a amplitude do sinal de acordo com as oportunidades de mascaramento impostas pelo modelo psicoacústico. A codificação entrópica atribui palavras mais curtas a símbolos do sinal que ocorrem com mais frequência, técnica também conhecida por redundância estatística [1].

Com base nas etapas de codificação, é então formulada uma palavra digital contendo informações especificas de cada codificador, no que diz respeito a configuração, sincronização, partilha de bits, dentre outras informações.

2.3. FORMATOS DE ÁUDIO DIGITAL

2.3.1. EVOLUÇÃO DOS FORMATOS DE ÁUDIO DIGITAL

Todos os codificadores ou formatos de áudio de alta qualidade atuais derivam de estratégias de codificação ensaiadas desde os anos 60, porém o primeiro codificador a contribuir de fato para definição dos codificadores normalizados atuais foi o codificador *Adaptive Transform Coding* (ATC). A partir do codificador ATC, inúmeros outros codificadores vieram a surgir nos anos posteriores. A Figura 8 contém um diagrama cronológico da evolução dos codificadores entre 1977 e os dias atuais [1].

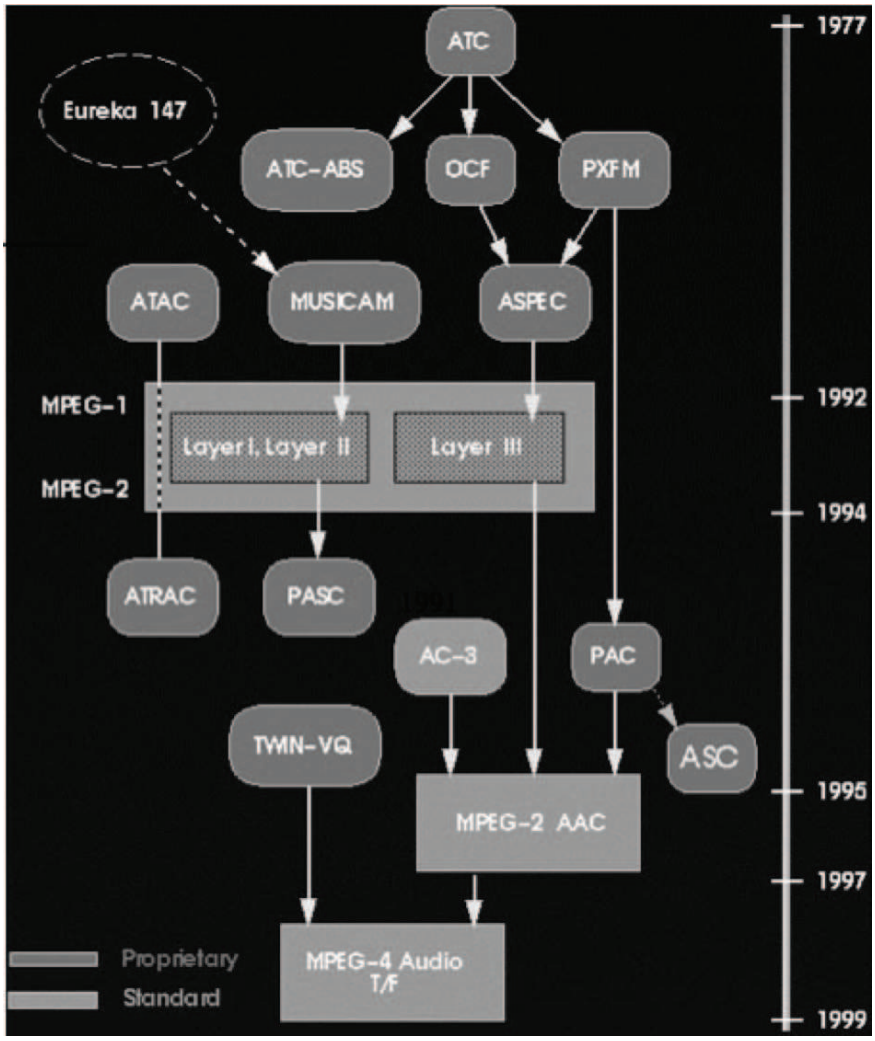


Figura 8 - Evolução dos codificadores de áudio entre 1977 e os dias atuais [8].

O codificador ATC realiza Codificação por blocos de 128 amostras, multiplicados por uma janela temporal, Utiliza uma Transformada Discreta de Fourier (DFT) ou Transformada Discreta do Cosseno (DCT), e também utiliza uma informação lateral, através da sub-amostragem e quantificação da informação logarítmica do espectro, de modo que é capaz de realizar uma adaptação contínua do passo de quantificação.

O codificador ATC-ABS é semelhante ao ATC, mas realiza escolha dinâmica do comprimento da transformada.

O codificador OCF também é semelhante ao ATC, mas utiliza transformada DTC modificada ou MDCT, usa quantificação não uniforme (de acordo com uma função de potencia), atribui palavras mais curtas aos símbolos mais prováveis da fonte (codificação de Huffman) e utiliza de dois mecanismos iterativos: um para controle da taxa de compressão e outro para controle do ruído de quantificação.

O codificador PXFM é semelhante ao OCF mas utiliza a transformada DFT ao invés da MDCT e introduz a codificação de um par de canais de áudio estereofônicos, promovendo processamento da soma e diferença de seus sinais, aumentando o ganho de codificação.

O codificador ATRAC também utiliza uma MDCT. O comprimento da MDCT pode ser alterado através de uma comutação dinâmica de janelas em função da estacionaridade do sinal.

O codificador MUSICAM analisa o sinal por meio de um banco de filtros de resposta ao impulso finita (FIR) pseudo-espelhos em quadratura (QMF) 32 bandas de frequência e uma FFT conjugada com um modelo pscicoacustico, de modo que é possível obter-se uma estimativa do limiar de mascaramento de cada banda e realizar ajuste dinâmico da taxa de compressão.

O codificador ASPEC utiliza uma transformada MDCT de comprimento variável por comutação dinâmica de janelas e uma Transformada Rápida de Fourier (FFT) conjugada com um modelo pscicoacustico, de modo que obtém uma estimativa do limiar de mascaramento de cada banda e também pode ajustar dinâmico da taxa de compressão. Esse codificador ainda pode codificar uma informação lateral baseada nos códigos de Huffman.

O codificador MPEG-1 Obtido a partir da superposição de 3 camadas hierárquicas. As camadas 1 e 2 tiveram suas características herdadas do codificador MUSICAM. A Camada 3 herda características do codificador ASPEC e denomina-se MP3. A codificação MP3 é feita por um banco de filtros pseudo-QMF de 32 bandas e uma MDCT.

O codificador MPEG-2 baseia-se no MPEG-1, mas possui frequências de amostragem reduzidas, debito Binario mais baixo e uma codificação de um canal extra para baixas frequências

O codificador MPEG-4 foi criado com o objetivo de ser versátil a diversos objetos multimídia. Sua codificação da fala é feita por técnicas de HVXC (*Harmonic Vetor eXcitation code*) e CELP (*Code Excited Linear Prediction*) e a codificação de áudio genérico é feito através de métodos de tempo/frequência

O codificador Dolby-AC3 foi criado como Alternativa à MPEG-2. Ele utiliza uma MDCT com janelas de Kaiser-Bessel, o número de amostras varia entre 256 e 512 e utiliza uma informação de áudio analógico gravado na película para sobrepor informações corrompidas de áudio digital [1].

2.3.2. CLASSIFICAÇÃO E APLICAÇÕES DE DIFERENTES FORMATOS DE ÁUDIO DIGITAL

A definição e classificação dos formatos de áudio digital dá-se a partir das técnicas de codificação utilizadas, bem como as taxas de compressão associadas, de modo que as diferenças funcionais entre os formatos de áudio digital os levam a ser utilizados em aplicações de tecnologia variadas.

A taxa de compressão é definida como o quociente entre o número de bits utilizados para representar um sinal na notação PCM e o número de bits utilizados para representar o sinal utilizando uma técnica de codificação mais eficiente, como mostrado na equação abaixo:

$$Taxa\ de\ compressão = \frac{Nº\ de\ bits\ na\ notação\ PCM}{Nº\ de\ bits\ utilizando-se\ técnica\ eficiente}$$

Isso significa que quando um codificador eficiente utiliza uma taxa de compressão de 5:1, por exemplo, isso quer dizer que o seu processo de codificação resulta numa redução do número bits utilizados para representar um sinal de áudio em 5 vezes, quando comparado com o número de bits utilizados para representar o mesmo sinal de áudio na notação PCM padrão.

A Tabela 1 apresenta exemplos de aplicação e os fatores de compressão associados aos principais formatos de áudio utilizados no mundo.

Formato de áudio	Fator de Compressão	Exemplos de aplicação
NICAM	< 2	Som digital estéreo em televisão analógica
MPEG-1 camada 1	4:1	Cassete digital compacta (DCC)
MPEG-1 camada 2	6:1	Difusão de áudio digital (CAB)
MPEG-1 camada 3	≥ 7:1	Distribuição na Internet, leitores MP3
MPEG-2 BC	6:1	DVD (Digital versatile disc), DVB
MPEG-2 AAC	≥ 8:1	Internet, difusão por rádio satélite
AC-3 (Dolby)	6:1	DVD, cinema (Dolby Digital), HDTV
MPEG-4 áudio T/F	≥ 9:1	Cenários usando áudio interactivo, DVB
Apt-X100 (APT)	4:1	Reportagem, estúdios rádio, cinema (DTS)
ATRAC (Sony)	5:1	MiniDisc, cinema (SDDS)
PAC (AT&T/Lucent)	≥ 8:1	Difusão rádio por satélite

Tabela 1 - Principais formatos de áudio utilizados no mundo [1].

2.3.3. FORMATO DE ÁUDIO MP3

O formato de áudio digital hoje conhecido como MP3 começou a ser desenvolvido em 1987 por pesquisadores do Instituto Fraunhofer IIS, através do projeto EU147 *Digital Audio Broadcasting* (DAB) dirigido por professores da Universidade de Erlange, na Alemanha [9].

Em 1988, a Organização Internacional de Padronização (ISO) convocou diferentes empresas e criou o grupo denominado *Motion Picture Expert Group* para formular um padrão de codificação digital de áudio e vídeo normalizado, a partir da compilação de diferentes técnicas de forma colaborativa, resultando na criação do formato digital hoje conhecido como MPEG-1[1].

O formato MPEG-1 foi desenvolvido a partir da superposição de 3 camadas hierárquicas. Mais tarde, por uma serie de razões, incluindo conveniência, a camada 3 do formato MPEG-1 recebeu a abreviação de MP3.

A codificação de MP3 é feita através de uma estrutura híbrida de análise, a qual combina um banco de filtros pseudo-QMF (*Quadrature Mirror Filter*) ou Filtro de Espelhos em Quadratura de 32 bandas, com uma Transformada de Cosseno Discreta Modificada (MDCT) [1].

A estrutura de um arquivo de MP3, também chamada de trama de dados, é essencialmente composta por cinco partes: Cabeçalho (Header), Informação de proteção contra erros (CRC), Informação Lateral, Dados Principais e Dados Auxiliares.

O cabeçalho é composto por 32 bits contendo informações a respeito da sincronização, a versão do MPEG e o número de camadas, o modo do CRC, a taxa de transmissão e a frequência de amostragem, bit de amortecimento, bit de privacidade, modos estéreo, direitos autorais e fabricantes.

O CRC (*Cyclic Redundancy Check*) pode se encontrar nos modos ativado, ou desativado. Se ativado, 16 bytes são adicionados a trama para detectar os dados mais propensos a erros de transmissão.

A informação lateral, entre outras funções, contém informação a respeito da localização do ultimo bit de dados, transmissão dos fatores de escala, localização de grânulos e das informações auxiliares, coeficientes do espectro de frequência, tabelas de decodificação Huffman e amplificação.

Os dados principais são basicamente fatores de escala e bits de Huffman decodificados usando as informações laterais [10].

Os dados auxiliares são opcionais e contêm informações extras como título das músicas, nome do artista e gênero das músicas [11].

2.4. MP3 PLAYERS

Os primeiros MP3 players portáteis foram desenvolvidos na Coréia por Kwang-su Moon e Jung-ha Hwang em novembro de 1997. Eles possuíam memória do tipo flash, se chamavam MPMan e MPStation e tinham tamanho físico e capacidade de armazenamento (64 Mbits) muito próximas a de um Walkman. Por outro lado, possuíam a vantagem da possibilidade de trocar de música, recurso que o Walkman ainda não possuía na época. Seguindo o exemplo dos coreanos, as empresas Diamond Multimídia, Creative Labs e Sony também criaram seus players com memória flash logo em seguida.

Em 1999, foi criado o primeiro MP3 player com memória do tipo disco rígido pela HanGo, posteriormente também pelas empresas Creative Labs e Archos. Esses players, por sua vez, eram bem maiores e mais pesados do que os de memória flash, porém possuíam uma capacidade de armazenamento muito maior [9]. A Tabela 2 contém informações gerais a respeito dos MP3 players pioneiros.

Empresa	Digital cast	Diamond media	Creative Labs	Sony	Hango	Creative Labs
Modelo	MP-Man F20	Rio PMP 300	Nomad	NW-MS7	PJB-100	Nomad Jukebox
Data de lançamento	Fev 98	Set 98	Abr 99	Set 99	Nov 99	Set 00
Preço ($)	200	200	170	432	750	500
Tamanho (cm)	17.83	14	14	10.29	47.22	104.78
Peso (g)	68	68	64.1	66.9	303.3	340.2
Tipo de Memória	Flash	Flash	Flash	Flash	Flash	HD
Formato de Música	MP3	MP3	MP3	ATRAC3	MP3	MP3
Capacidade de Memória (MB)	64	64	64	64	4800	6000
Taxa de transferência (MB/s)	2	2	2	12	12	12

Tabela 2 - Informações gerais sobre os players pioneiros [9].

Desse modo, verifica-se que os MP3 players criados entre os anos de 1998 e 2000 custavam entre 200 e 750 dólares, possuíam uma capacidade de armazenamento entre 64 e 6000 Mbytes e tinham tamanho físico entre 10 e 100 cm (4 a 40 polegadas) aproximadamente.

2.4.1. CENÁRIOS DE USO

Com o avanço da tecnologia, a fabricação de MP3 players tem se tornado cada vez mais flexível, uma vez que as novas técnicas de codificação têm proporcionado uma melhoria da qualidade do áudio digital, os novos métodos de compressão têm permitido uma diminuição dos espaços de memória utilizados para seu armazenamento, e o descobrimento de novos materiais tem promovido uma melhoria da qualidade dos dispositivos eletrônicos.

Impulsionadas pela obtenção de maior lucro, as empresas de um modo geral procuram se adequar às necessidades dos consumidores. Desse modo, os MP3 players

atuais vem ganhando novos modelos, formas, tamanhos e cores, proporcionando um aumento da gama de aplicações, bem como um aumento de seus cenários de uso.

Os Cenários de Uso representam as formas e contextos de utilização dos MP3 players, levando-se em consideração os ambientes em que são usados e as finalidades de sua utilização. Hoje em dia existem inúmeros cenários de uso dos MP3 players, dentre os quais pode-se citar: Utilização dos players para ouvir música dentro do carro; Utilização dos players para ler ou estudar; Utilização dos players para fazer exercícios físicos dentro da academia ou durante a caminhada; Utilização dos players em eventos sociais, cerimônias e festas, entre outros.

2.4.2. IMPORTÂNCIA

É comprovado cientificamente que a música, e consequentemente os players, é capaz de estimular o cérebro, e alterar os níveis de concentração e reações emotivas, tanto de forma objetiva, quanto subjetiva [12].

Uma pesquisa feita por Cockrill [13] mostrou a importância dos MP3 players em diferentes áreas. Os participantes foram privados da utilização dos players e tiveram de registrar seus sentimentos por uma semana. O resultado da pesquisa mostrou que para determinadas pessoas, os MP3 players são capazes de melhorar o desempenho em atividades esportivas, aumentar a capacidade de concentração, melhorar o desempenho dos estudos e do trabalho, aumentar os níveis de relaxamento e a qualidade do sono, causar alteração de humor, entre outros.

3. JUSTIFICATIVA

A justificativa de realização do projeto fundamenta-se em dois aspectos principais: um aspecto técnico e outro acadêmico.

O aspecto técnico esta relacionado com os cenários de uso, uma vez que a utilização de um MP3 Player com Bluetooth é capaz de resolver problemas intrínsecos a ambientes que possuem a necessidade de controle de dados de forma remota. Um exemplo desse cenário de uso é a utilização dos players para reproduzir áudio digital em ambientes dotados de arquiteturas vastas e assimétricas, como lojas e shoppings, em que o controle manual dos tocadores de música torna-se inviável. Desse modo, poderia se utilizar o player com Bluetooth para realizar o controle de dados a longa distância.

O aspecto acadêmico é que este relatório final constituir-se-á de um recurso que poderá ser utilizado por outros pesquisadores para o desenvolvimento de novos projetos científicos uma vez que se encontra dotado de uma descrição detalhada das etapas de desenvolvimento e integração dos componentes utilizados na construção do player.

4. OBJETIVOS

4.1. OBJETIVO GERAL

O objetivo geral do projeto é desenvolver um sistema embarcado capaz de executar áudio em formato digital. O sistema deverá ser dotado de duas formas de controle: Controle Local e Controle Remoto e possuir um dispositivo de armazenamento de dados removível para conter os arquivos de áudio a serem reproduzidos.

4.2. OBJETIVOS ESPECÍFICOS

1) Montar e testar um MP3 Player básico numa Protoboard

Este objetivo visa obter um MP3 básico que possa ler um conjunto de músicas contidas num SD card, e posteriormente reproduzí-las uma a uma, na ordem em que se encontram organizadas dentro do SD card, numa saída de áudio para fone de ouvido ou caixa de som.

2) Desenvolver um Controle Local para o dispositivo

Este objetivo visa obter um controle local para o dispositivo, de modo que o usuário possa ter controle sobre o funcionamento do Player a partir da interação com o hardware atrelado ao sistema embarcado do dispositivo.

3) Desenvolver um Controle Remoto para o dispositivo

Este objetivo visa obter um controle remoto para o dispositivo, de modo que o usuário possa realizar o controle do funcionamento do player a partir de um MacBook via Bluetooth.

4) Desenvolver uma Interface de Comunicação Intuitiva

Este objetivo visa desenvolver uma interface de comunicação intuitiva entre o usuário e MP3 Player, de modo que o usuário posa controlar o funcionamento do player de forma simples.

5) Reproduzir o circuito numa Placa de Circuito Impresso (PCB)

Este objetivo visa reproduzir o circuito do sistema embarcado do dispositivo em Placas de Circuito Impresso, de modo a obter-se uma redução do tamanho físico do dispositivo, bem como torná-lo mais resistente a choques mecânicos.

6) Projetar uma estrutura de transporte para as PCB's

Este objetivo busca projetar uma estrutura de transporte para o dispositivo ou caixa 3D que envolva o as PCB's por completo, de modo a tornar o dispositivo mais fácil de transportar e mais resistente a choques mecânicos.

7) Correção de Software

Este objetivo visa corrigir falhas de software a fim de tornar o sistema mais robusto a erros e melhorar o seu funcionamento.

5. METODOLOGIA

Para realização da etapa inicial de projeto, a montagem de um sistema embarcado funcionando como um MP3 player básico, montou-se o circuito numa protoboard utilizando-se diferentes componentes eletrônicos analógicos e digitais conectados a um microcontrolador Arduino Micro. Depois escreveu-se e carregou-se um código em C para dentro do arduino utilizando o compilador DevC.

Para realização da segunda etapa de projeto, desenvolvimento de um controle local para o dispositivo, integrou-se botões switches e resistores ao sistema, e realizou-se algumas modificações de software.

Para realização da terceira etapa de projeto, desenvolvimento de um controle remoto para o dispositivo, integrou-se um modulo de Bluetooth ao sistema embarcado e fez-se novas adaptações de software no sistema.

Para realização da quarta etapa de projeto, desenvolvimento de uma interface intuitiva para controle do dispositivo (GUI), utilizou-se o software Xcode, um ambiente de desenvolvimento de aplicativos para MAC, para criar uma interface virtual que facilitasse a comunicação entre o usuário e o player.

Para realização da quinta etapa de projeto, reprodução do circuito numa Placa de Circuito Impresso, utilizou-se o software P-CAD para desenhar o layout das placas, e posteriormente fabricou-se as PCB's a partir do processo de remoção de cobre utilizando uma máquina especializada.

Para realização da sexta etapa de projeto, projeto de uma estrutura de transporte para as PCB's ou caixa 3D, realizou-se diversas medições das placas de circuito impresso e utilizou-se o software SketchUp para modelagem da caixa 3D.

Para realização da sétima etapa de projeto, correção de software, fez-se diferentes testes do funcionamento do hardware e software do dispositivo, e posteriormente alterou-se as linhas de código do programa principal em C para corrigir falhas de software do sistema.

6. EXECUÇÃO E RESULTADOS

Inicialmente, para tornar as especificações de projeto mais claras, desenhou-se um protótipo de papel e um Diagrama de Verplank do MP3 player, os quais estão mostrados nos itens 6.1. e 6.2. Posteriormente procedeu-se com as etapas de desenvolvimento descritas nos itens 6.3 a 6.9.

6.1. PROTÓTIPO DE PAPEL

O Protótipo de papel constitui-se de um esquema desenhado em papel que detalha o funcionamento de um dispositivo ou objeto durante a sua utilização, bem como as interações que ocorrem entre ele, o seu usuário, e outros objetos do ambiente que o cerca com os quais o dispositivo também interage. Um protótipo de papel do MP3 Player está mostrado na Figura 9.

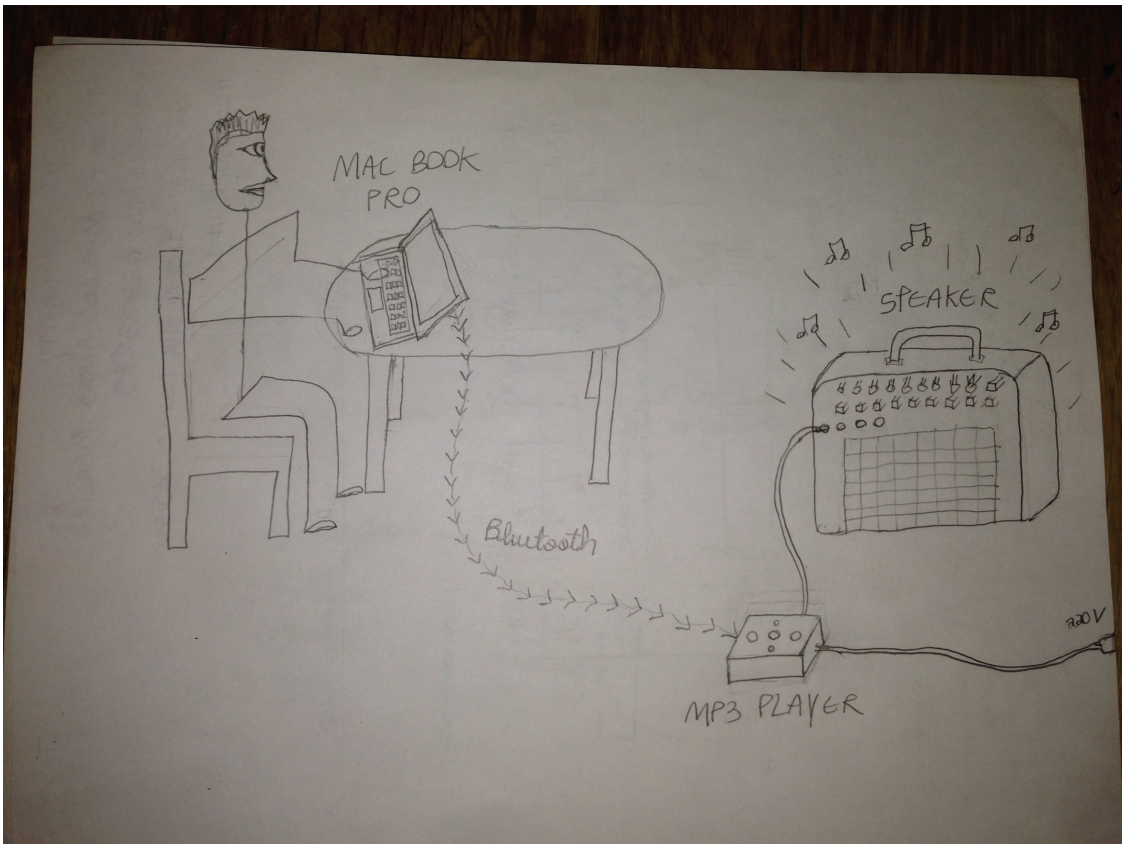


Figura 9 - Protótipo de Papel do MP3 Player, mostrando o detalhamento de seu funcionamento [14].

A Figura 9 mostra um dos exemplos de utilização do MP3 Player. Neste caso, o usuário se encontra sentado em uma cadeira, controlando um MP3 player por meio de seu MacBook, via Bluetooth.

O MacBook deve ser capaz de enviar diferentes comandos ao MP3 player, como por exemplo 'passar para a próxima música' ou 'voltar para a música anterior'. O MP3 Player, por sua vez, deve ser capaz de receber e processar os comandos enviadas pelo MacBook, bem como apresentar botões switches atrelados ao seu hardware (como mostrado na Figura 9) de modo que o usuário também tenha a possibilidade de realizar um controle local do dispositivo. O MP3 ainda deve possuir um conector de saída de áudio, o qual poderá ser conectado a um fone de ouvido ou a uma caixa de som (speaker) como mostrado na Figura 9.

6.2. DIAGRAMA DE VERPLANK

O Diagrama de Verplank foi criado pelo Engenheiro Bill Verplank com o objetivo caracterizar produtos e sistemas interativos do ponto de vista prático [15]. Desse modo, o Diagrama de Verplank evidencia as necessidades e preocupações que devem ser consideradas durante o processo de desenvolvimento de um dispositivo interativo, de acordo com sua aplicação. Um Diagrama de Verplank é composto por 6 partes:

- Idea: ideia principal do dispositivo;
- Methaphor: metáfora que exemplifique a ideia principal;
- Model: modelo do dispositivo, geralmente desenhado;
- Display: forma de visualização de informações através do dispositivo;
- Error: Condição de erro de funcionamento do dispositivo;
- Scenario: Cenário de uso do dispositivo;
- Task: tarefa desempenhada pelo dispositivo;
- Control: Formas de controle do dispositivo.

Assim, analisando-se as características e funcionalidades do MP3 Player, foi possível montar seu Diagrama de Verplank associado, originalmente escrito em inglês, o qual está mostrado na Figura 10.

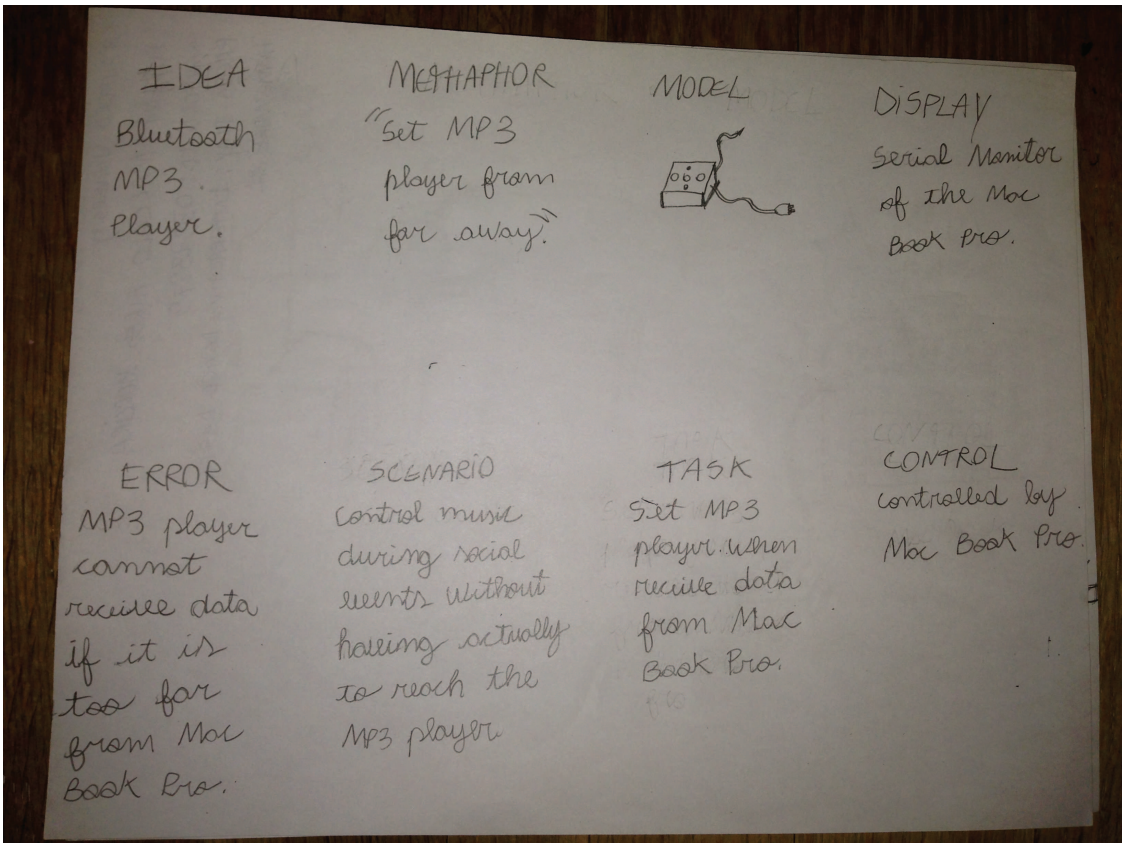


Figura 10 - Diagrama de Verplank do MP3 Player [15].

Nessa Figura temos:

- Idea: MP3 player com Bluetooth;
- Methaphor: " Controlar o MP3 player de uma longa distância";
- Modelo: Desenhado na Figura;
- Display: Monitor serial do MacBook Pro.
- Error: MP3 player não é capaz de receber dados quando localizado em uma posição muito distante do MacBook;
- Scenario: Controlar música durante eventos sociais sem ter que alcançar o dispositivo de fato;
- Task: Configurar o MP3 quando receber dados do MacBook Pro;
- Control: Controlado pelo MacBook Pro.

6.3. MONTAGEM E TESTE DE UM MP3 BÁSICO NUMA PROTOBOBARD

Para iniciar o projeto, pesquisou-se o esquema de montagem em protoboard e o código em C de um MP3 básico, os quais são mostrados nos itens 5.1.1 e 5.1.2..

6.3.1. HARDWARE

Para montagem do circuito em protoboard utilizaram-se os seguintes componentes:

- Arduino Micro: Microcontrolador da família do Arduino, mostrado na Figura 11;

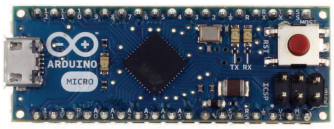


Figura 11 - Arduino Micro

- MP3 and Midi Breakout Board for VS1053: Placa integrada para decodificação de áudio, mostrada na Figura 12;

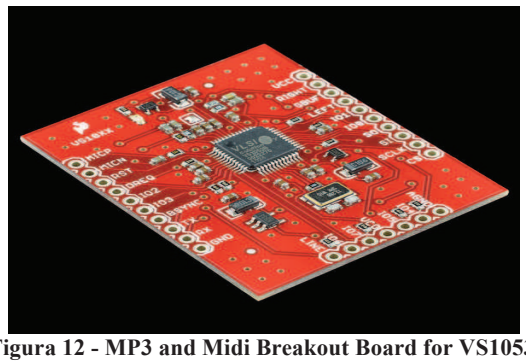


Figura 12 - MP3 and Midi Breakout Board for VS1053.

- CI HEF4050-BP: CI divisor de tensão 5 para 3.3V, mostrado na Figura 13;

Figura 13 - CI HEF4050BP.

- Conector Audio Jack TRS 3.5mm: conector de áudio, mostrado na Figura 14;

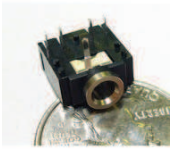


Figura 14 - Conector Audio Jack TRS 3.5mm.

- Micro SD Card 8GB, mostrado na Figura 15;

Figura 15 - Micro SD Card 8GB.

- Cabo USB Padrão, mostrado na Figura 16;

Figura 16 - Cabo USB Padrão.

Uma vez obtidos todos os componentes, pode-se montar o circuito numa protoboard fazendo-se as seguintes ligações [17]:

Partindo do MicroSD card:		
Micro SD card **MISO**	→	Arduino MISO **MI**
Micro SD card **GND**	→	Arduino **GND**
Micro SD card **SCK**	→	HEF4050BP **2**
Micro SD card **3.3V+**	→	Arduino **3.3V**
Micro SD card **GND**	→	Arduino **GND**
Micro SD card **MOSI**	→	HEF4050BP **4**
Micro SD card **CS**	→	HEF4050BP **6**

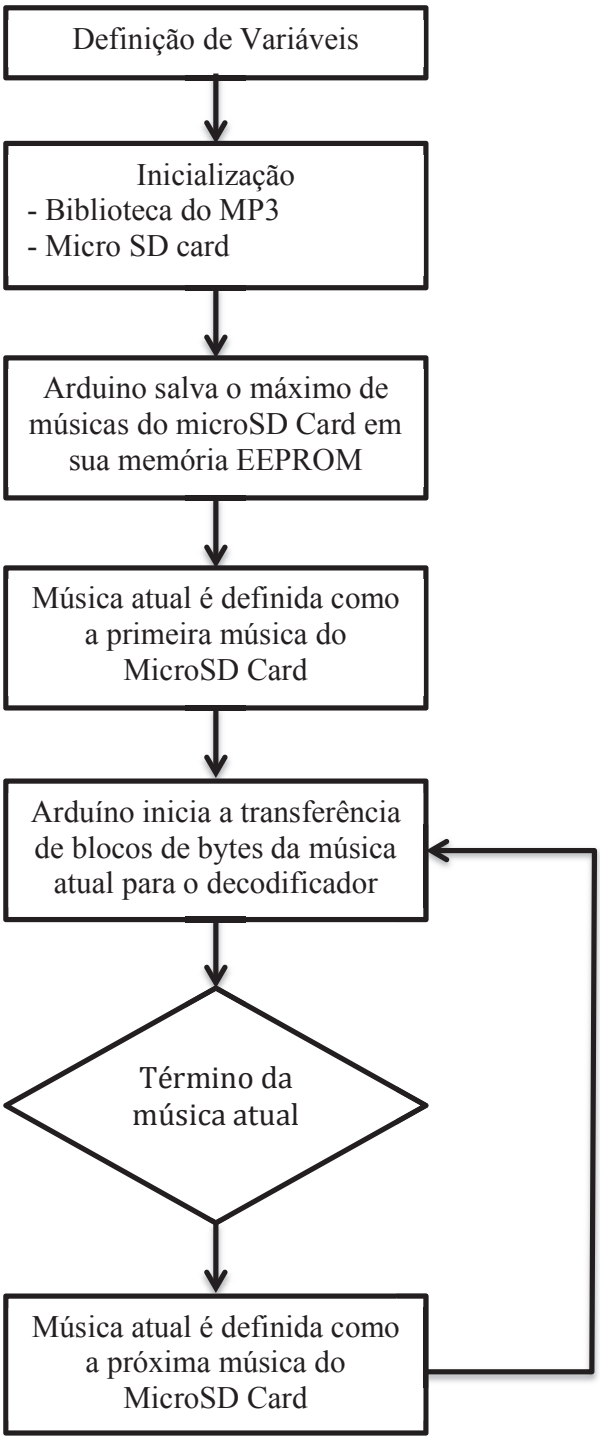
Partindo do HEF4050BP:		
HEF4050BP **1**	→	Arduino **3.3V**
HEF4050BP **8**	→	Arduino **GND**
HEF4050BP **11**	→	Arduino **A1**
HEF4050BP **14**	→	Arduino **A0**
HEF4050BP **3**	→	Arduino **SCK**
HEF4050BP **5**	→	Arduino MOSI **MO**
HEF4050BP **7**	→	Arduino **SS**

Partindo do VS1053:		
VS1053 **CS**	→	HEF4050BP **15**
VS1053 **SCLK**	→	HEF4050BP **2**
VS1053 **SI**	→	HEF4050BP **4**
VS1053 **SO**	→	Arduino MISO **MI**
VS1053 **Vcc**	→	Arduino **3.3V**
VS1053 **GND**	→	Arduino **GND**
VS1053 DCS **BSYNC**	→	HEF4050BP **12**
VS1053 **DREQ**	→	Arduino **A2**
VS1053 **RST**	→	Arduino **RESET**
VS1053 **LEFT**	→	AUDIO JACK **2**
VS1053 **GBUF**	→	AUDIO JACK **1**
VS1053 **RIGHT**	→	AUDIO JACK **3**

Com as conexões feitas na protoboard, o próximo passo foi escrever o código em C, o qual está mostrado na seção 6.3.2, e carregá-lo no arduino.

6.3.2. SOFTWARE

O código em C utilizado para controlar o circuito, foi adaptado de [16] e encontra-se no Anexo 1. A rotina do código utilizado pode ser visualizado a partir do diagrama de blocos mostrado abaixo:



Para carregar o programa no arduino, utilizou-se o compilador DEVC, cuja interface está mostrada na Figura 17, conectou-se o arduino micro no MacBook através do cabo USB padrão e clicou-se no botão UPLOAD, no canto superior esquerdo, com o símbolo: =>.

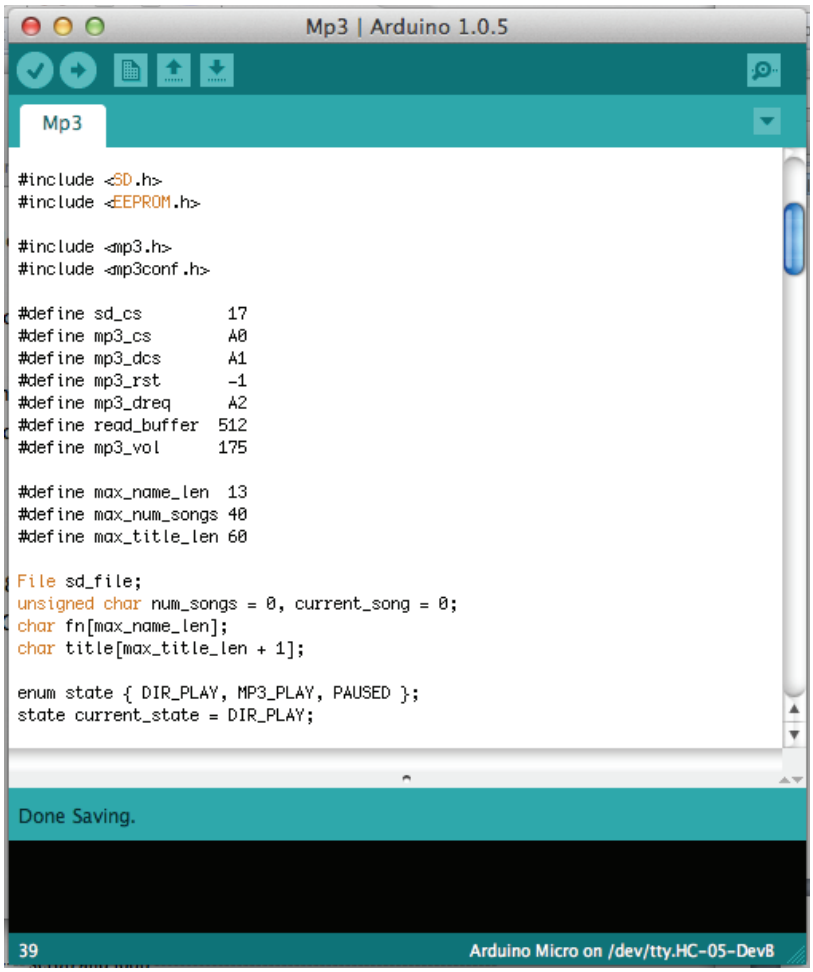


Figura 17 - Interface de comunicação com o Arduino.

Desse modo, foi testado o MP3 player básico, o qual funcionou bem. Neste ponto, o MP3 player era capaz apenas de ler um conjunto de músicas do SD card e tocar todas elas iniciando da primeira até a última, de acordo com a ordem em que aparecem no SD Card, sucessivo e ininterruptamente, até que o arduino deixe de ser alimentado por energia.

6.4. DESENVOLVIMENTO DE UM CONTROLE LOCAL PARA O DISPOSITIVO

O objetivo do controle local é atrelar 5 botões ao hardware do dispositivo, de modo que se possa ter controle do funcionamento do MP3 Player. Os botões devem possuir as seguintes funcionalidades:

- Botão 1: 'Dar Play ou Pause no MP3';
- Botão 2: 'Voltar para a música anterior';
- Botão 3: 'Ir para a próxima música';
- Botão 4: 'Baixar o volume';
- Botão 5: 'Aumentar o volume'.

6.4.1. HARDWARE

Desse modo, para implementar o hardware utilizou-se:

- 3 switches grandes para o controle da música (botões 1, 2 e 3);
- 2 switches pequenos para o controle do volume (botões 4 e 5);
- 5 resistores de 10kΩ, cada um deles ligado a cada um dos botões.

O esquema de montagem foi o seguinte:

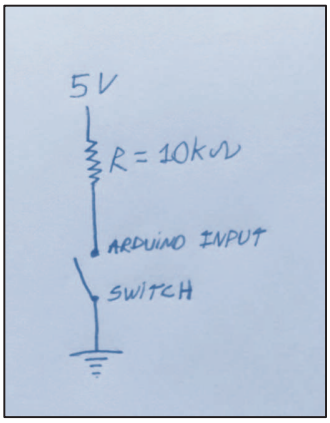


Figura 18 - Esquema de montagem dos botões switch.

Desse modo quando o switch se enontra aberto, o pino de entrada do arduino (Arduino Input) recebe 5V. Por outro lado, quando o switch se enontra fechado, tem-se uma queda de tensão no resistor R, e o pino de entrada do arduino (Arduino Input) recebe 0V.

6.4.2. SOFTWARE

Para implementação do software utilizou-se dois tipos de lógica: pooling e interrupção.

A lógica de pooling consiste em constantemente checar se um evento ocorreu no laço principal do programa. Caso o evento ocorra, deve-se fazer as configurações desejadas no dispositivo.

Já a lógica de Interrupção consiste em ajustar o software de acordo com a ocorrência de um evento no hardware, como o acionamento de um botão. Nesse caso, ao receber um comando acionado por hardware, o software para imediatamente o que está fazendo (execução de funções do loop infinito) e pula para as linhas de código relacionadas com a interrupção. Assim, só depois de executar todas as linhas de código dentro da função de interrupção, é que o programa retorna para o loop principal, no mesmo ponto em que tinha sido interrompido [18].

Assim, para evitar o esforço computacional, geralmente se utiliza a técnica de pooling para controlar eventos que ocorrem com mais frequência, uma vez que a sua condição de funcionamento será checada inúmeras vezes pelo programa principal. Por outro lado, a técnica de interrupção é utilizada para controlar eventos que ocorrem com menos frequência, uma vez que seu funcionamento ocorre fora do loop principal, de modo que a utilização de funções de interrupção com muitas linhas de código ou a ocorrência de um número grande de interrupções durante um curto espaço de tempo pode comprometer o desempenho do programa principal.

Desse modo, estudando-se as características do funcionamento do player, escolheu-se os tipos de lógica a serem utilizados por cada um dos botões:

- Para o controle dos botões relacionados com o controle do volume (botões 4 e 5), utilizou-se a técnica de pooling, uma vez o ajuste do volume pode ocorrer mais de uma vez durante a reprodução de uma música;

- Para o controle dos botões relacionados com o controle da música (botões 1, 2 e 3) utilizou-se a técnica de interrupção, uma vez que sua ocorrência é menor durante a duração de 1 música.

Quando se utiliza botões switches um erro comum que pode ocorrer é o Boucing, o qual consiste na tendência de dois contatos de metais de dispositivos eletrônicos de

gerar múltiplos sinais de saída quando seus contatos abrem ou fecham. Uma solução para o Boucing é a utilização da técnica de Debouncing aliada ás técnicas de pooling ou interrupção. Deboucing é qualquer tipo de hardware ou software que garanta que apenas um sinal será ativado a partir da ocorrência do fechamento ou abertura de dois contatos elétricos [19].

Uma forma simples de implementar o debouncing via software é salvar o momento de ocorrência da primeira interrupção e comparar com o momento da ocorrência da próxima interrupção, para verificar se o tempo entre as duas interrupções é menor do que uma pequena fração de segundos (algumas centenas de milissegundos). Nesse caso assume-se que a ativação das interrupções não foram geradas pelo usuário, uma vez que o intervalo de frequência entre elas foi muito pequeno.

Dessa forma para implementar o software de controle dos botões 1, 2 e 3 inicialmente atrelou-se cada um dos pinos do arduino em que os botões estão conectados fisicamente a 3 interruptores do arduino, inserindo-se as seguintes linhas de código no setup():

```
attachInterrupt(2, PlayPause, RISING);
attachInterrupt(0, NextSong, RISING);
attachInterrupt(1, PreviousSong, RISING);
```

Desse modo, quando a interrupção ocorre, as funções PlayPause(), NextSong() e PreviousSong() são ativadas. Essas três funções foram implementadas em conjunto com as técnicas de debouncing descritas anteriormente e podem ser visualizadas nas linhas de código a seguir, as quais foram inseridas no loop():

```
void PlayPause()
{
 static unsigned long last_interrupt_time = 0;
 unsigned long interrupt_time = millis();
 // If interrupts come faster than 200ms, assume it's a bounce and ignore
 if (interrupt_time - last_interrupt_time > 200)
 {
  if(current_state == DIR_PLAY){
   current_state = PAUSED;
  }
  else{
   current_state = DIR_PLAY;
```

```
    }
  }
  last_interrupt_time = interrupt_time;//digitalWrite(pin, LOW);
}

void NextSong()
{
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  // If interrupts come faster than 200ms, assume it's a bounce and ignore
  if (interrupt_time - last_interrupt_time > 200)
  {
    sd_file.close();
    current_state = DIR_PLAY;
  }
  last_interrupt_time = interrupt_time;//digitalWrite(pin, LOW);
}

void PreviousSong()
{
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  // If interrupts come faster than 200ms, assume it's a bounce and ignore
  if (interrupt_time - last_interrupt_time > 200)
  {
    sd_file.close();
    if(current_song == 0){
      current_song = num_songs-2;
    }
    else{
      current_song = current_song - 2;
    }
    current_state = DIR_PLAY;
   }
  last_interrupt_time = interrupt_time;//digitalWrite(pin, LOW);
}
```

Posteriormente, para controlar os botões 4 e 5, utilizou-se a técnica de pooling em conjunto com debouncing. A implementação em C foi feita adicionando-se as linhas de código a seguir ao loop():

```
buttonState1 = digitalRead(buttonPin1);
 if (buttonState1 == HIGH) {
   static unsigned long last_interrupt_time = 0;
   unsigned long interrupt_time = millis();
   // If interrupts come faster than 200ms, assume it's a bounce and ignore
   if (interrupt_time - last_interrupt_time > 100)
   {
     if(vol>0){
     vol = vol-10;
     }
   }
   last_interrupt_time = interrupt_time;//digitalWrite(pin, LOW);
 }

 buttonState2 = digitalRead(buttonPin2);
 if (buttonState2 == HIGH) {
   static unsigned long last_interrupt_time = 0;
   unsigned long interrupt_time = millis();
   // If interrupts come faster than 200ms, assume it's a bounce and ignore
   if (interrupt_time - last_interrupt_time > 100)
   {
     if(vol<254){
     vol = vol+10;
     }
   }
   last_interrupt_time = interrupt_time;//digitalWrite(pin, LOW);
 }

Mp3.volume(vol);
```

Depois de testar todos os botões e ter certeza que todos estavam funcionando bem, procedeu-se para o passo seguinte, que era instalar um controle remoto para o dispositivo.

6.5. DESENVOLVIMENTO DE UM CONTROLE REMOTO PARA O DISPOSITIVO

O objetivo inicial do controle remoto era realizar uma comunicação sem fio entre o MacBook e o Arduino, de forma que o usuário pudesse controlar o MP3 de uma certa distância. Desse modo, quando o usuário apertasse algumas das teclas do MacBook, e depois apertasse <enter>, sinais de controle deveriam ser enviados do MacBook ao MP3.

Desse modo, definiu-se os comandos que devem ser enviados ao player a partir do pressionamento de teclas do Macbook:

- Envio do caractere 2 via Macbook: Ativa comando 'Play/Pause' do MP3 player;

- Envio do caractere 3 via Macbook: Ativa comando 'Ir para a proxima música' do MP3 player;

- Envio do caractere 1 via Macbook: Ativa comando 'Voltar para a música anterior' do MP3 player;

- Envio do caractere + via Macbook: Ativa comando 'Aumentar o volume' do MP3 player;

- Envio do caractere - via Macbook: Ativa comando 'Baixar o volume' do MP3 player.

A seguir, serão apresentados os detalhes de desenvolvimento de hardware e software do controle remoto do dispositivo.

6.5.1. HARDWARE

O desenvolvimento do hardware foi feito a partir da integração de um modulo de Bluetooth ao circuito principal do dispositivo. O módulo utilizado foi o Bluetooth RF Transceiver Module serial RS232 TTL HC-05, o qual pode ser visualizado na Figura 18.

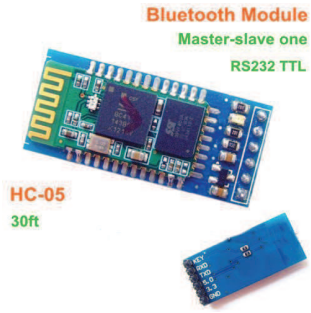


Figura 19 - Bluetooth RF Transceiver Module serial RS232 TTL HC-05.

As conexões feitas foram as seguintes:

Partindo do Módulo de Bluetooth		
Bluetooth **Vcc**	→	Arduino **5V**
Bluetooth **GND**	→	Arduino **GND**
Bluetooth **RX**	→	Arduino **11**
HEF4050BP **TX**	→	Arduino **10**

Desse modo, com a integração do módulo de Bluetooth ao sistema embarcado, pode-se montar o circuito completo do MP3 Player em protoboard, o qual pode ser visualizado na Figura 20.

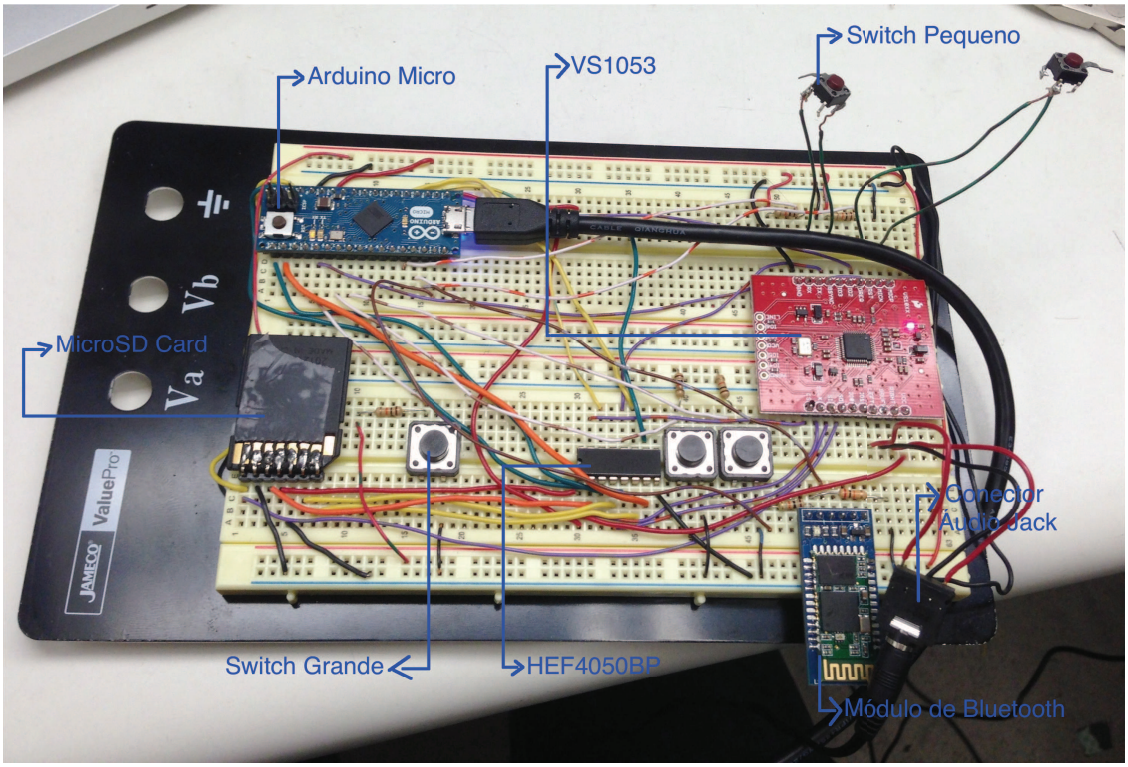


Figura 20 - Circuito completo do MP3 Player montado em protoboard.

6.5.2. SOFTWARE

O arduino micro já apresenta uma forma de comunicação serial integrada ao seu chip, a qual é do tipo UART, e se encontra disponível a partir da utilização dos pinos 0 (RX) e 1 (TX), porém uma vez que esta porta já está sendo utilizada pelo programa, foi necessário recorrer a outros métodos de comunicação.

A solução encontrada foi utilizar a função SoftwareSerial do arduino, a qual foi desenvolvida com o intuito de replicar a comunicação serial utilizando-se software. Desse modo, é possível ter-se várias portas de software serial com velocidade de até 115200 bps [20].

Assim, para realizar a implementação em software da comunicação do MacBook com o player utilizou-se a função SoftwareSerial, e inicialmente se instanciou o objeto Genotronex e conectou-se os pinos RX e TX do Bluetooth aos pinos 11 e 10 do arduino, respectivamente, adicionando-se as seguintes linha de código fora das funções setup() e loop():

```
SoftwareSerial Genotronex(10, 11);
```

Depois, para monitorar o recebimento de bits pelo MP3, adicionou-se as seguintes linhas de código ao loop() [21]:

```
if (Genotronex.available()){
  BluetoothData=Genotronex.read();

  if(BluetoothData=='1'){   // if number 1 pressed ....
   PreviousSong();
   Genotronex.println("Going back to the previous song.");
  }
  if (BluetoothData=='2'){// if number 0 pressed ....
   PlayPause();
   Genotronex.println("Play/Pause song.");
  }
  if (BluetoothData=='3'){// if number 0 pressed ....
   NextSong();
   Genotronex.println("Going to the next song");
  }
  if(BluetoothData=='='){   // if number 1 pressed ....
   if(vol<254){
   vol = vol+10;
```

```
    Genotronex.println("Volume Increased");
    }
  }
  if(BluetoothData=='-'){   // if number 1 pressed ....
    if(vol>0){
    vol = vol-10;
    Genotronex.println("Volume Decreased");
    }
  }
```

Desse modo, o programa é configurado para receber os caracteres 2, 3, 1 e chamar as funções PreviousSong(), PlayPause() e NextSong() para controlar as músicas do player, ou os caracteres – e = (o mesmo que + no teclado do Mac), para controlar o volume do player.

6.6. DESENVOLVIMENTO DE UMA INTERFACE DE COMUNICAÇÃO INTUITIVA

Neste etapa de projeto, procurou-se desenvolver uma interface de comunicação intuitiva entre o usuário e o MP3 Player, de modo que o usuário pudesse realizar o controle do MP3 Player de forma simples.

Desse modo, usando o software Xcode, inicialmente desenhou-se o modelo de interface desejado no arquivo .xib do Xcode, como mostrado no centro da Figura 21.

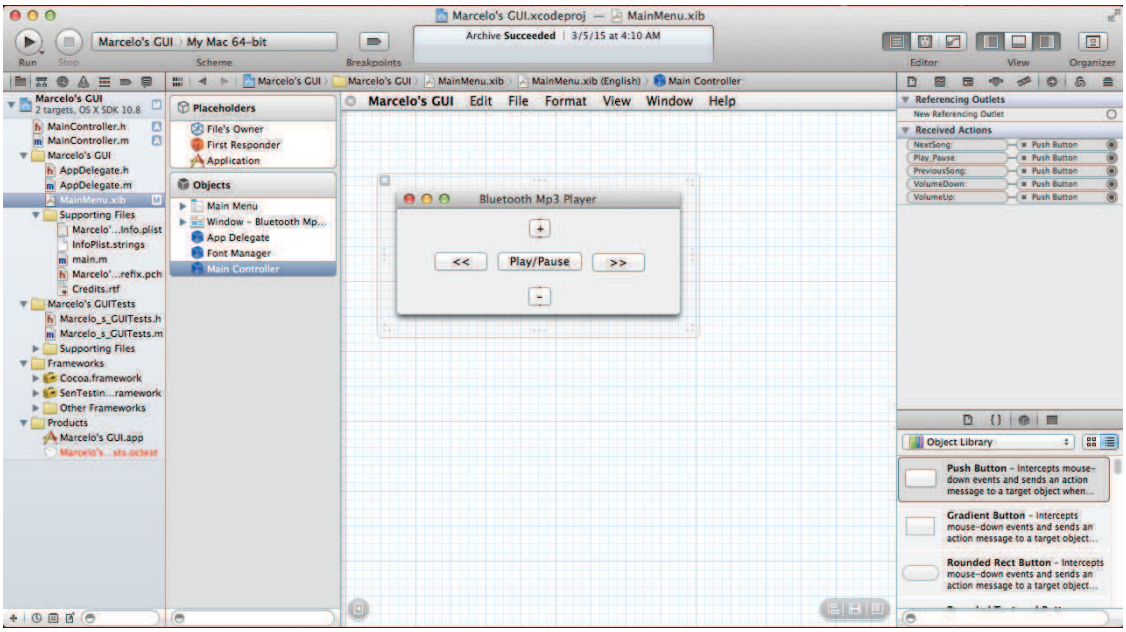


Figura 21 - Sofware Xcode mostrando o arquivo .xib.

Depois, atrelou-se o pressionamento de cada um dos botões da interface visual, com a chamada de 5 diferentes funções, como mostrado do lado direito da Figura 21:

- Pressionamento do botão >>, chama a função 'NextSong';
- Pressionamento do botão Play/Pause, chama a função 'Play_Pause';
- Pressionamento do botão <<, chama a função 'PreviousSong';
- Pressionamento do botão -, chama a função 'VolumeDown';
- Pressionamento do botão Play/Pause, chama a função 'VolumeUp';

Posteriormente, escreveu-se o código mostrado abaixo, na linguagem de programação Objective C:

```
#import "MainController.h"

@implementation MainController

-(IBAction)Play_Pause:(id)sender{
    popen("echo '2' > /dev/tty.HC-05-DevB", "r");
}

-(IBAction)NextSong:(id)sender{
    popen("echo '3' > /dev/tty.HC-05-DevB", "r");
}

-(IBAction)PreviousSong:(id)sender{
    popen("echo '1' > /dev/tty.HC-05-DevB", "r");
}

-(IBAction)VolumeUp:(id)sender{
    popen("echo '=' > /dev/tty.HC-05-DevB", "r");
}

-(IBAction)VolumeDown:(id)sender{
    popen("echo '-' > /dev/tty.HC-05-DevB", "r");
}

@end
```

Esse código funciona da seguinte maneira: cada uma das funções criadas ('NextSong', 'Play_Pause', PreviousSong', 'VolumeDown', 'VolumeUp') ao serem chamadas, enviam os caracteres 2, 3, 1, = e – para a porta /dev/tty.HC-05-DevB, por meio da função popen(). A porta /dev/tty.HC-05-DevB é a porta que se comunica com o Bluetooth do sistema embarcado do MP3 player.

O MP3 player foi configurado para receber os caracteres 2, 3, 1, = e –, e realizar o controle o funcionamento do dispositivo, durante a etapa de 'desenvolvimento de software para controle remoto do dispositivo'(item 6.5.2)

Assim, colocou-se o código escrito em Objective C dentro do arquivo .m do Xcode, como mostrado na Figura 22.

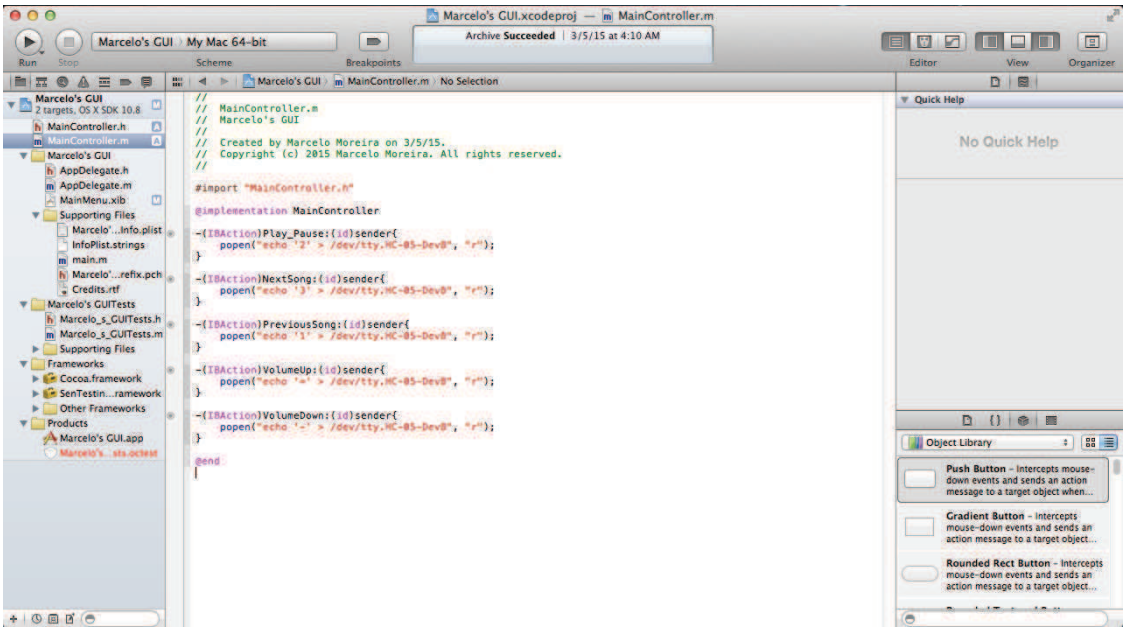


Figura 22 - Sofware Xcode mostrando o arquivo .m.

E depois pressionou-se os comandos Product->Archive para construir a interface mostrada na Figura 23.

Figura 23 - Interface virtual para comunicação entre o MacBook e o MP3 Player.

6.7. REPRODUÇÃO DO CIRCUITO NUMA PCB

Para reproduzir o circuito em Placas de Circuito Impresso, inicialmente, desenhou-se o esquemático do circuito da Figura 20, o qual está mostrado na Figura 24.

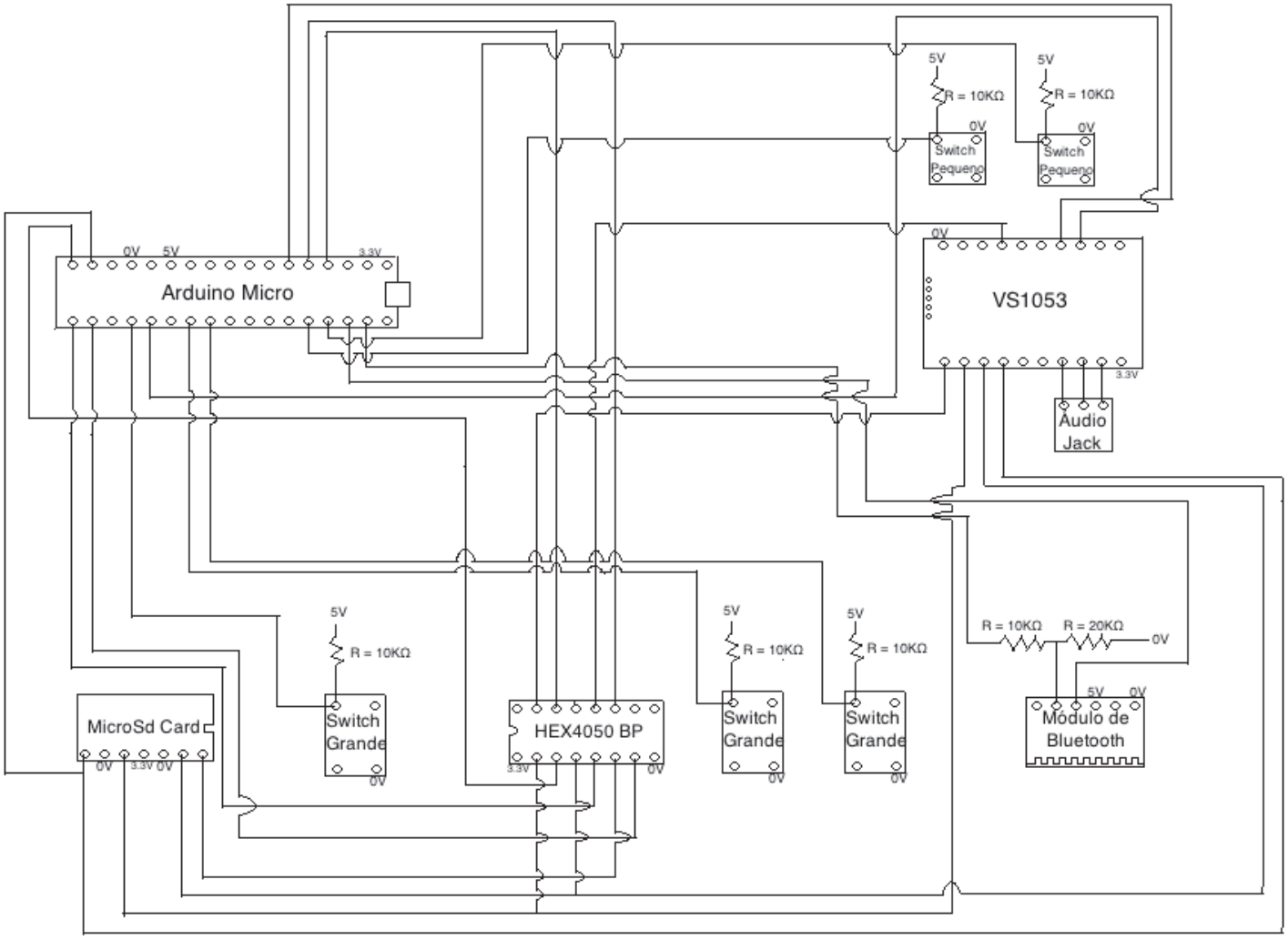


Figura 24 - Esquemático do circuito do MP3 Player Completo.

Posteriormente, utilizando-se o software P-CAD Pattern Editor, criou-se cada componente do esquemático da Figura 24, como mostrado nas Figuras de 25 a 33.

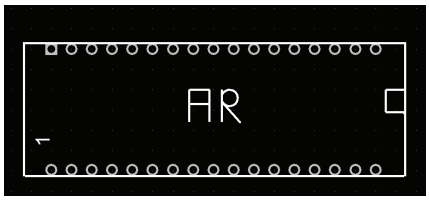


Figura 25 - Componente Arduino no P-CAD.

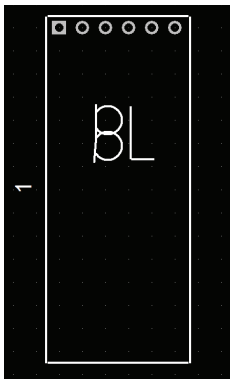


Figura 26 - Componente Módulo de Bluetooth no P-CAD.

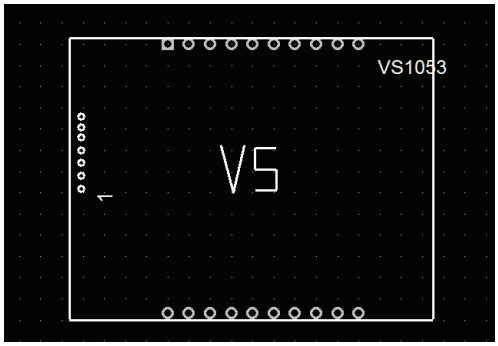


Figura 27 - Componente VS1053 no P-CAD

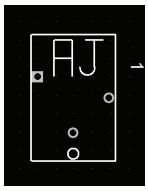


Figura 28 - Componente Conector Audio Jack no P-CAD.

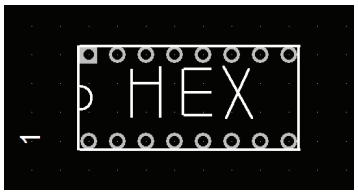


Figura 29 - Componente HEX4050BP no P-CAD.

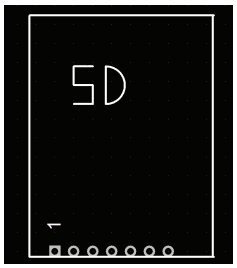


Figura 30 - Componente SDcard no P-CAD.

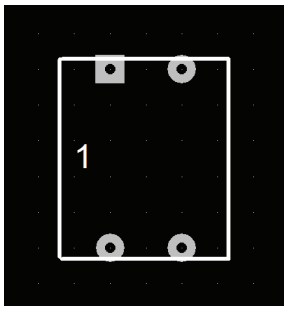


Figura 31 - Componente Switch Grande no P-CAD.

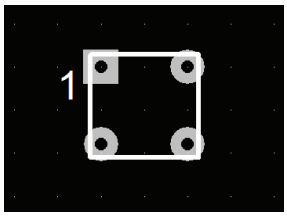


Figura 32 - Componente Switch Pequeno no P-CAD.

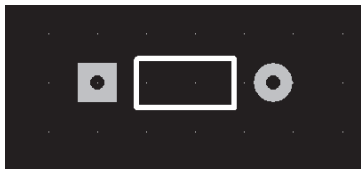


Figura 33 - Componente Resistor no P-CAD.

Posteriormente, utilizando-se o software P-CAD PCB, desenhou-se o layout do circuito da Figura 20 em duas placas de circuito impresso diferentes.

Para desenhar o layout da PCB 1, inicialmente fez-se as conexões entre os pinos dos componentes de acordo com as ligações do esquemático da Figura 24, como mostrado na Figura 34.

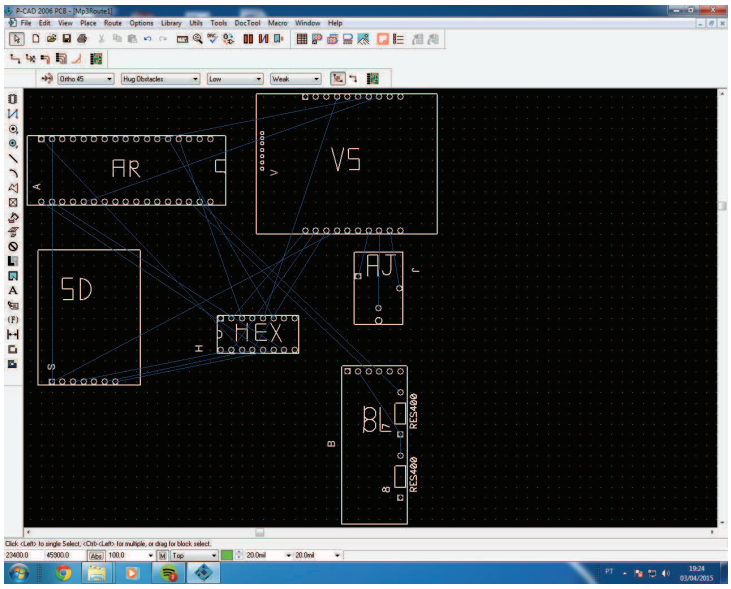


Figura 34 - Conexões da PCB 1.

Depois, mudou a posição dos componentes para encontrar posições que fossem capazes de gerar o menor espaço físico possível, como mostrado na Figura 35.

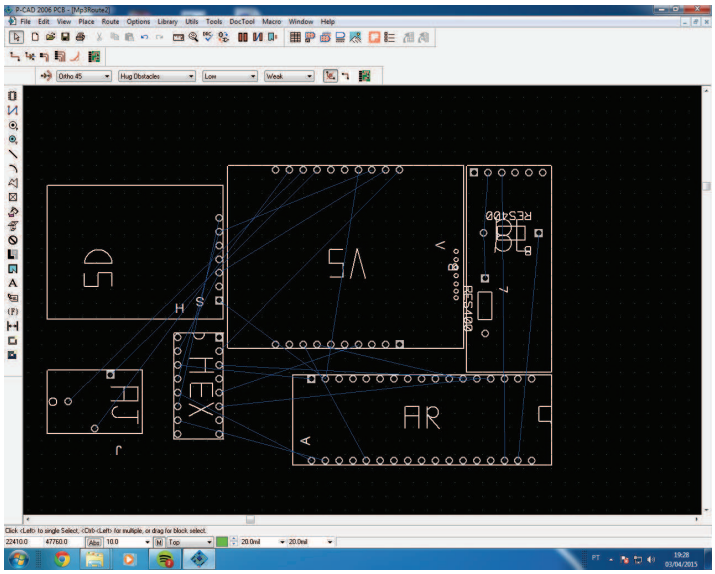


Figura 35 - Posicionamento dos componentes da PCB 1.

O critério de espaço utilizado foi a tentativa do MP3 Player completo, depois de pronto, caber dentro do bolso médio de uma Calça Jeans comum, o qual apresenta as dimensões médias de 18x12x10cm (largura, comprimento, altura).

Posteriormente, fez-se o roteamento da placa, que são as conexões metálicas que substituem os fios da protoboard, e que estão representadas pelas linhas vermelhas das Figuras 36 a 39.

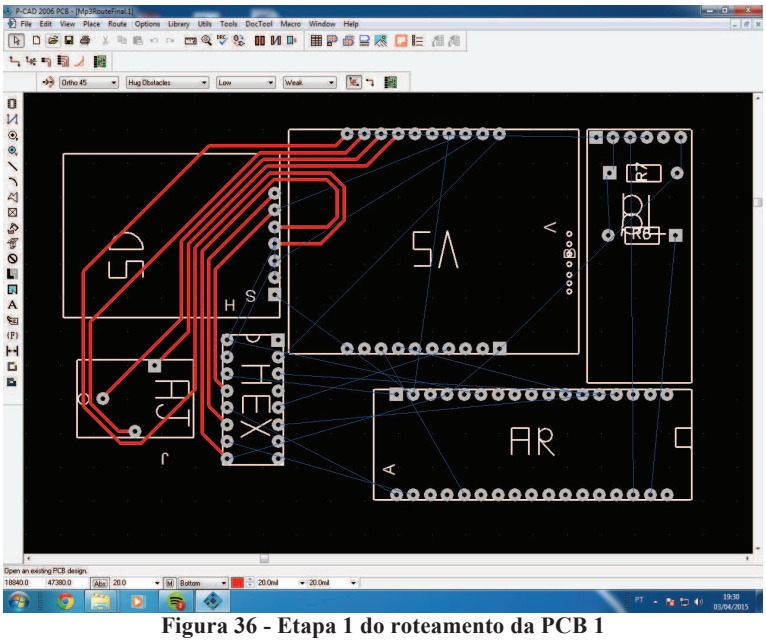


Figura 36 - Etapa 1 do roteamento da PCB 1

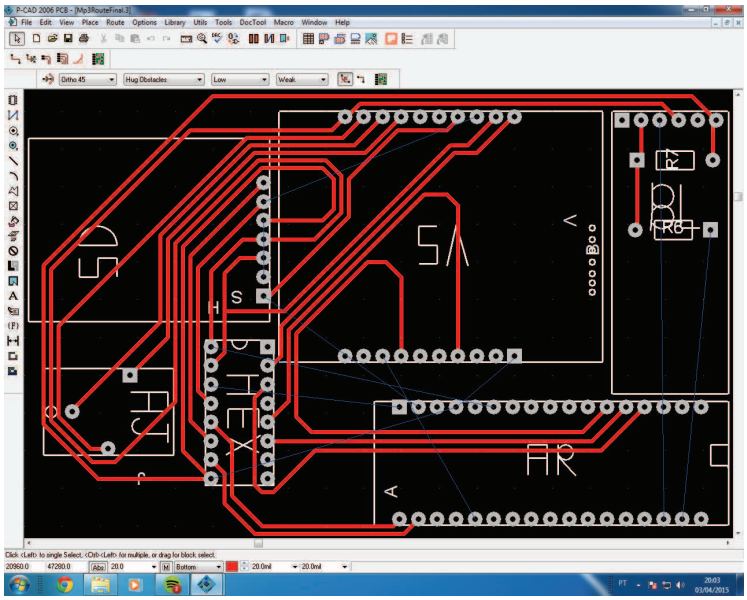


Figura 37 - Etapa 2 do roteamento da PCB 1.

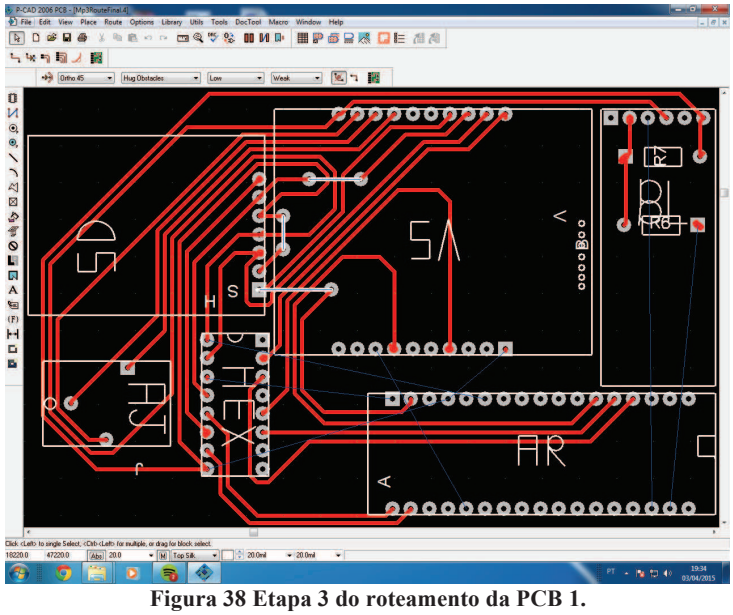


Figura 38 Etapa 3 do roteamento da PCB 1.

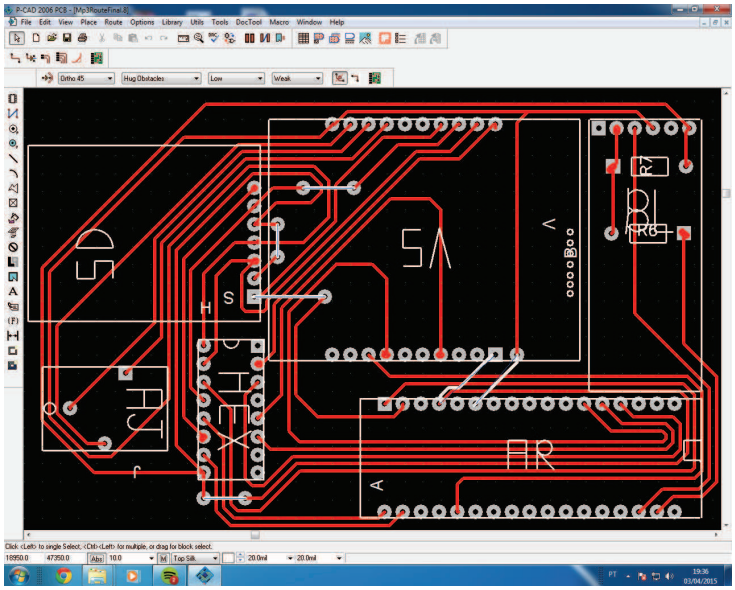


Figura 39 - Etapa 4 do roteamento da PCB 1.

E por último, fez-se a delimitação da placa, como mostrado na Figura 40.

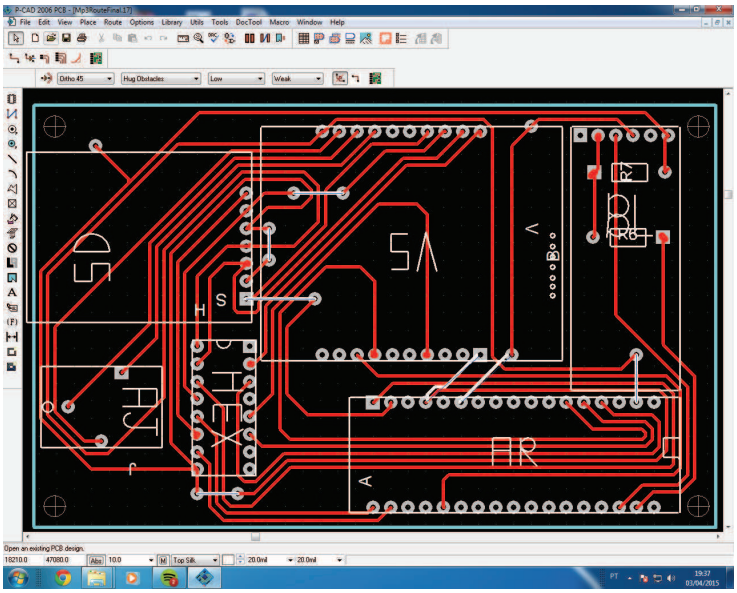


Figura 40 Delimitação da PCB 1.

Para desenho do layout da PCB2, inicialmente copiou-se a delimitação da placa 1, e inseriu-se os componentes restantes, como mostrado na Figura 41.

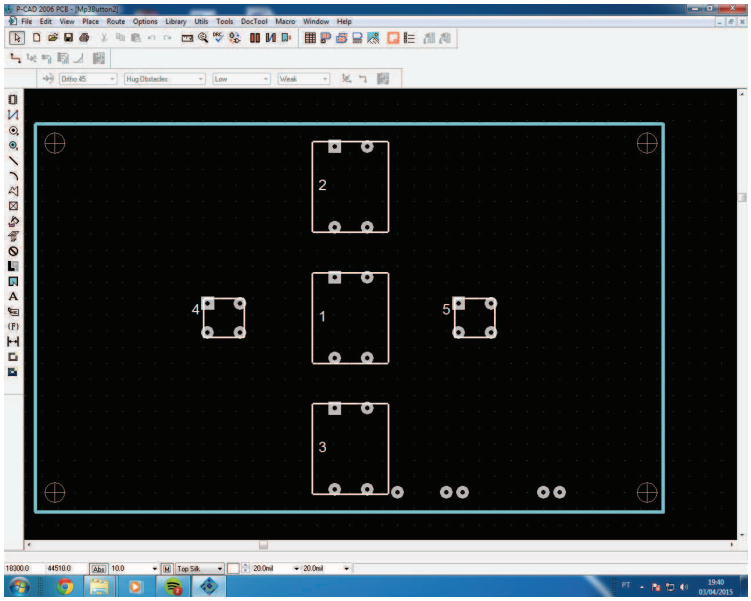


Figura 41: Etapa 1 do desenho de layout da PCB 2.

Posteriormente, fez-se as conexões entre os pinos dos componentes e fez-se o layout da placa, como mostrado nas Figura 42 e 43.

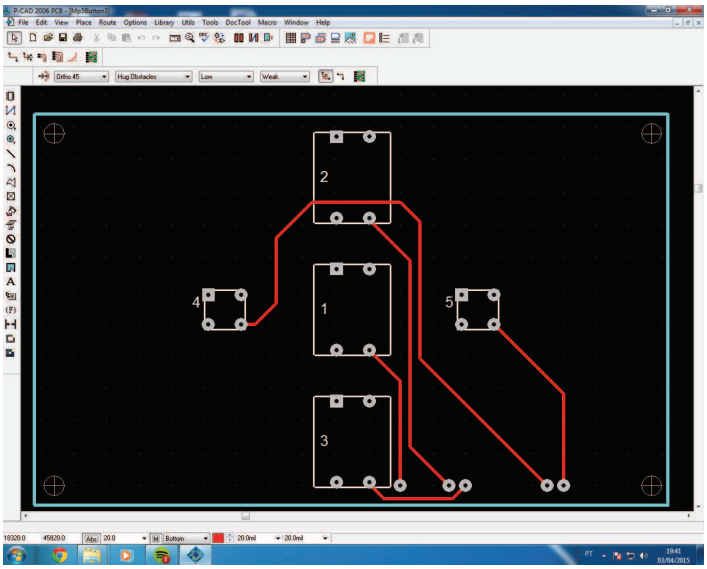


Figura 42 - Etapa 2 do desenho de layout da PCB 2.

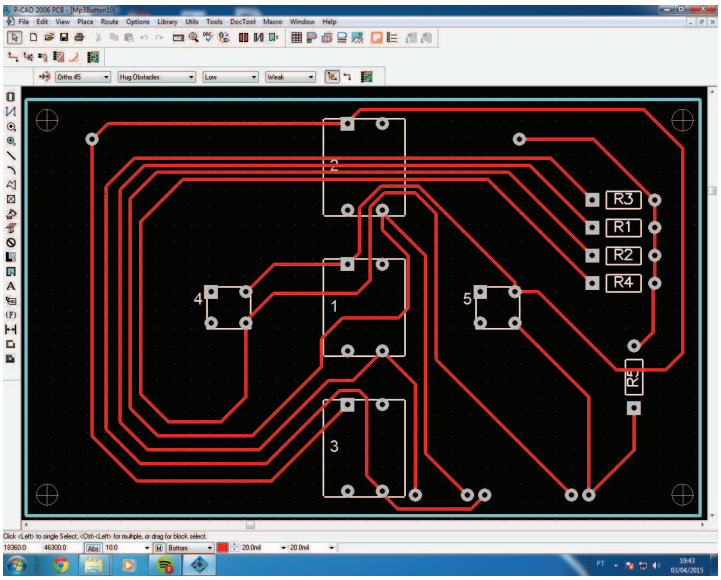


Figura 43 Etapa 2 do desenho de layout da PCB 2.

Terminado os layouts das PCB's 1 e 2, utilizou-se uma máquina especializada em fabricação de PCB's para fabricar as duas placas, e soldou-se os componentes reais em cada uma delas.

Os resultados da fabricação da PCB1 podem ser visualizados nas Figuras 44 e 45.

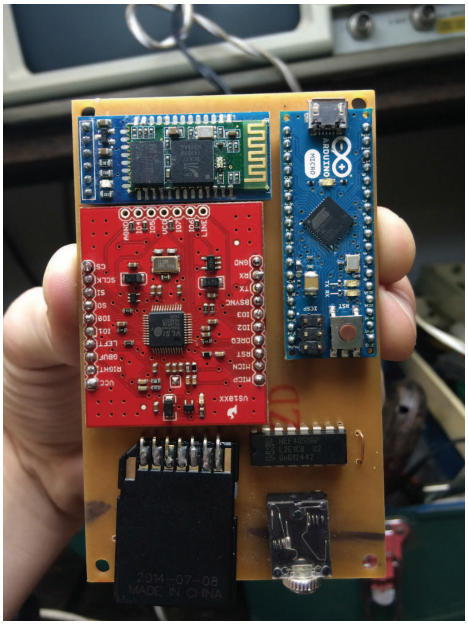


Figura 44 - Face frontal da PCB1 fabricada.

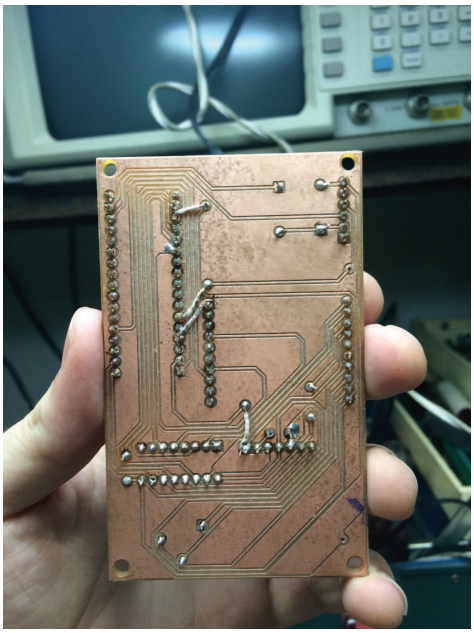


Figura 45 Face traseira da PCB1 fabricada.

Os resultados da fabricação das PCB2 podem ser visualizados nas Figuras 46 e 47.

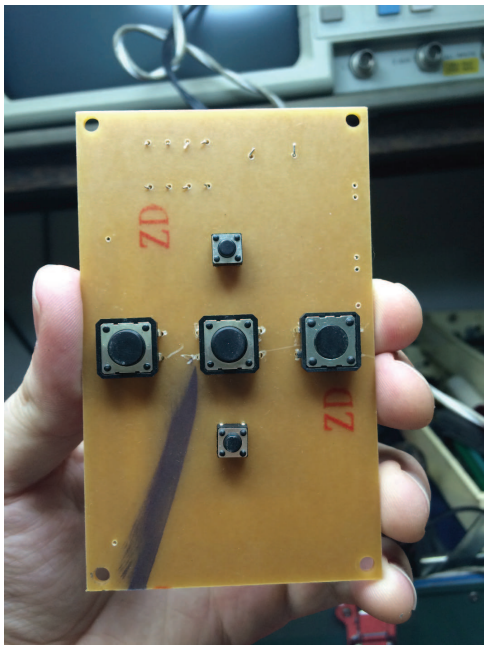


Figura 46 - Face frontal da PCB2 fabricada.

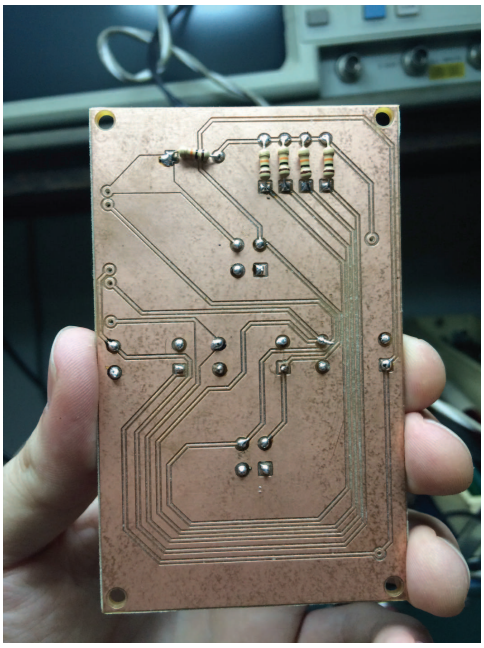


Figura 47 - Face traseira da PCB2 fabricada.

Por último, utilizou-se 4 tubos de material cerâmico, 4 parafusos de 1/8 (3.2 mm de diâmetro), 4 porcas de 1/8 e fios para acoplar uma placa na outra. O resultado pode ser visualizado na Figura 48.

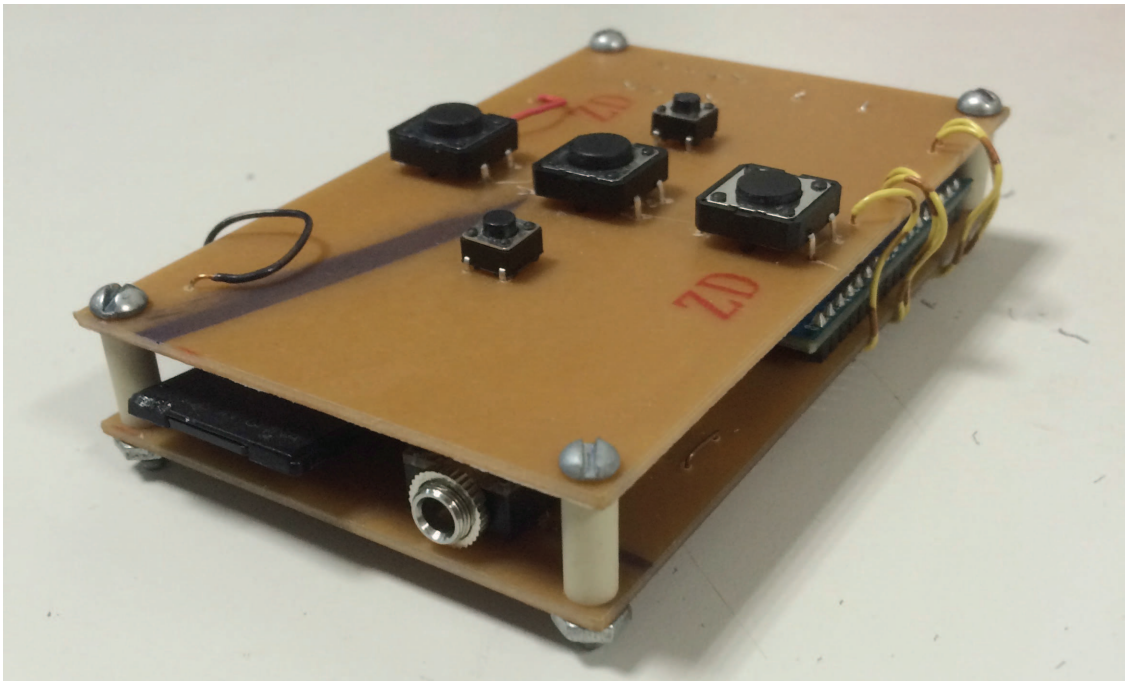


Figura 48 - Resultado da acoplamento das PCB's 1 e 2.

6.8. PROJETO DE UMA ESTRUTURA DE TRANSPORTE PARA AS PCB'S

Para projetar uma estrutura de transporte para as PCB's, modelou-se uma caixa 3D utilizando-se o Software SketchUp.

Inicialmente, mediu-se diversas dimensões de cada uma das PCB's. Posteriormente, utilizando o Software SketchUp, projetou-se uma caixa 3D que fosse compatível com as dimensões das PCB's.

Os critérios para o projeto da caixa 3D foram:

- As PCB's deveriam ficar totalmente envoltas pela caixa, ficando expostos apenas os componentes eletrônicos que são utilizados para interação do usuário com o dispositivo, que são os 4 botões switches, a saída do conector de áudio para o fone de ouvido, a saída do cabo que liga o Arduino Micro a energia e o MicroSD card;
- A espessura dos lados da caixa não deveriam ultrapassar 1 cm para não houvesse o aumento significativo do tamanho físico do dispositivo.

Assim, os resultados do projeto da caixa de transporte do dispositivo podem ser visualizados na Figura 49.

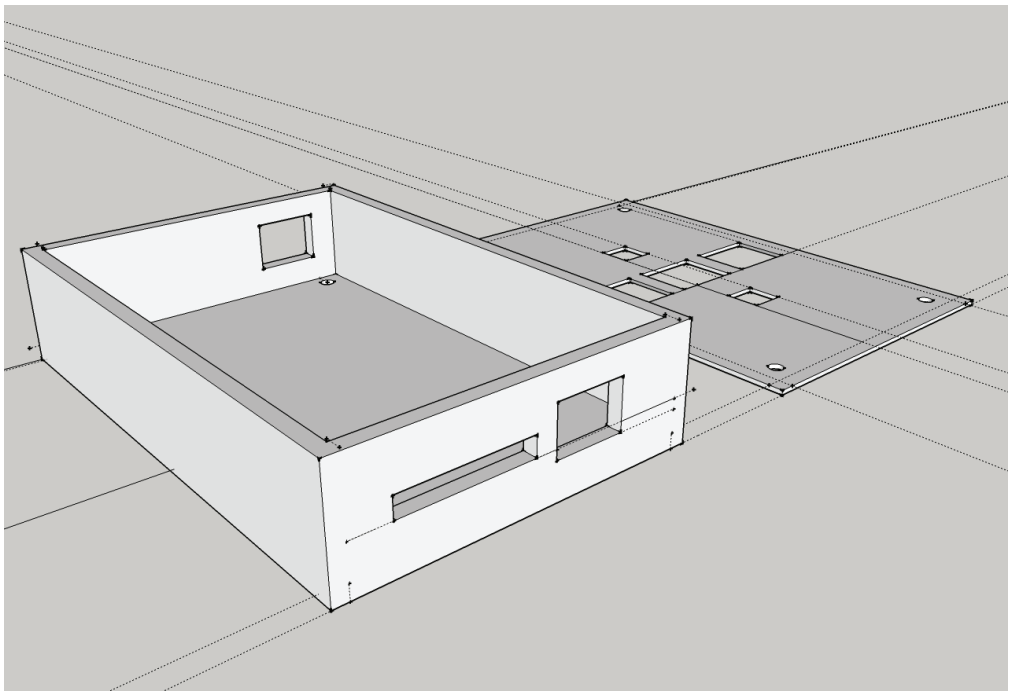


Figura 49 - Visualização da Caixa 3D de transporte do MP3 Player no SketchUp.

6.9. CORREÇÃO DE SOFTWARE

Durante a última etapa de projeto, inicialmente realizou-se diferentes testes de hardware e software do dispositivo, com objetivo de detectar possíveis erros ou aspectos de software que poderiam ser melhorados.

Desse modo, foi possível:

- Corrigir erros relacionados com alguns botões switches;
- Reduzir linhas de código do programa principal.

Assim, após o término desta etapa, concluiu-se o projeto. Os resultados finais obtidos podem ser visualizados a seguir.

7. RESULTADO FINAL

Ao final do projeto, obteve-se um sistema embarcado formado a partir da integração de diferentes componentes eletrônicos analógicos e digitais controlados por um microcontrolador do tipo Arduino Micro e conectados entre si por meio de duas Placas de Circuito Impresso interligadas através de uma estrutura mecânica composta de parafusos e porcas. O sistema ficou com as dimensões de 9.9x6.1x1.8 cm, funciona como um MP3 Player e pode ser visualizado na Figura 50.

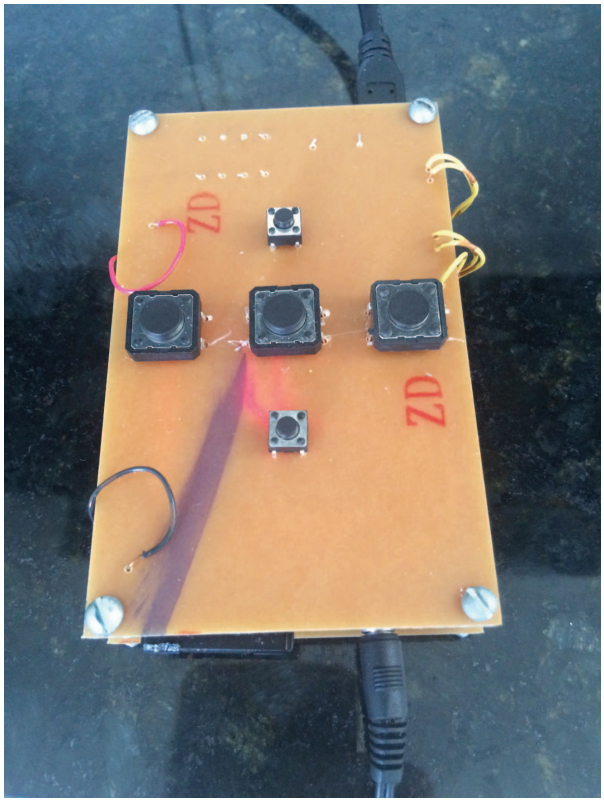


Figura 50 – Resultado Final do MP3 Player.

Para ligar o MP3 player, deve-se conectar o cabo que sai do arduino, mostrado na parte superior da Figura 50, direto na tomada (220V). Desse modo, o Arduino será inicializado automaticamente. Uma vez que o arduino é inicializado, as músicas que se encontram armazenadas no SD Card começam a ser reproduzidas.

As músicas podem ser ouvidas conectando-se um fone de ouvido ou caixa de som ao conector de saída de áudio acoplado ao sistema e mostrado na parte inferior direita da Figura 50. Para substituir as músicas do SDcard, o usuário deve retirar o microSD card de dentro de seu adaptador, mostrado na parte inferior esquerda da Figura 50, encaixá-lo a outro adaptador de MicroSD card, para então conectar o adaptador de SD no MAC e realizar a troca de músicas.

O controle do dispositivo pode ser feito de duas formas: local ou remotamente. O controle local é feito pressionando-se botões switches integrados no sistema embarcado do player, mostrados no meio da Figura 50, com os quais o usuário tem as opções de avançar para a próxima música, voltar para a música anterior, dar Play ou Pause e controlar o aumento ou diminuição do volume. Já o controle remoto é feito via Bluetooth através de uma interface virtual que pode ser visualizada no MacBook. Essa interface virtual está mostrada no lado esquerdo da Figura 51.

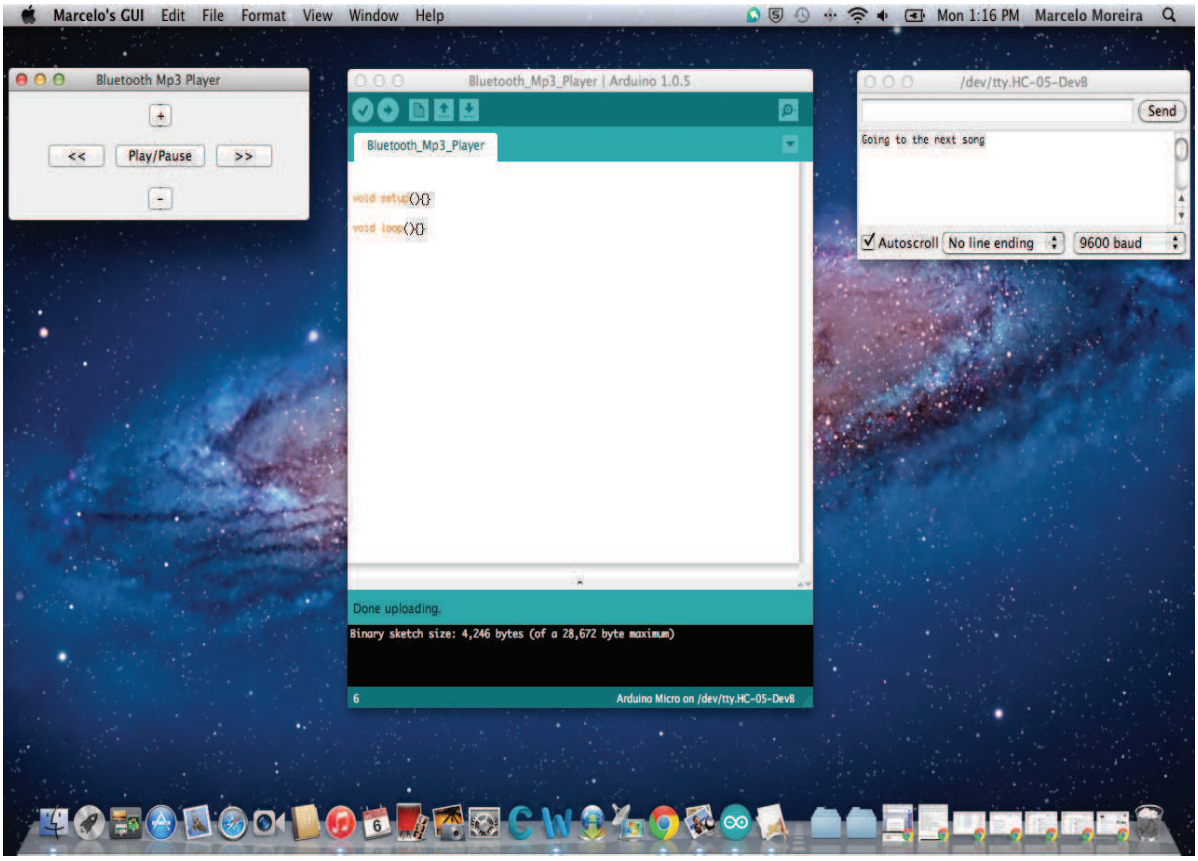


Figura 51 - Controle remoto do dispositivo.

A interface da Figura 51 apresenta botões virtuais com as mesmas funcionalidades dos botões integrados ao player, de modo que o usuário pode controlar o player remotamente pressionando os botões da tela virtual.

Para inicializar o controle remoto do dispositivo, o usuário deve primeiramente abrir o compilador DevC com um programa vazio, formado pelas funções Void loop() e Void setup() vazias, como mostrado no meio da Figura51, depois selecionar a porta de comunicação dev/tty.HC-05-DevB, clicando em tools->serial port-> dev/tty.HC-05-DevB e depois clicar em UPLOAD (segundo botão da esquerda para direita no canto superior esquerdo do DevC, indicado por =>). Posteriormente, deve-se abrir o monitor serial do DevC, mostrado no lado direito da Figura 51, e clicar-se no botão superior direito do DevC, simbolizado por uma lupa, também mostrado na Figura 51. E por último, para abrir a interface virtual do dispositivo, deve-se dar dois cliques em cima do aplicativo do Mac de nome Bluetooth MP3 Player, mostrado no lado esquerdo da Figura 51.

8. CONCLUSÃO

Este projeto apresentou o desenvolvimento e a montagem de um sistema embarcado funcionando como um MP3 Player básico. Utilizando-se de um Arduino Micro, uma placa integrada para decodificação de áudio VS1053, um CI HEF4050-BP, Micro SD Card 8GB e um Conector Áudio Jack TRS 3.5mm foi possível alcançar o objetivo proposto.

Considerando a importância da interação do usuário com o dispositivo desenvolvido, foram realizadas algumas modificações de software e adicionados alguns botões switches ao hardware do dispositivo, para que o usuário fosse capaz de adquirir um controle local sobre o funcionamento do player.

Posteriormente, impulsionado pelo problema enfrentado por alguns cenários de uso de MP3 players e pela necessidade de ter que alcançar o dispositivo para realizar o seu controle, desenvolveu-se um controle do dispositivo à distância, de modo que o usuário passou a poder controlar o player via Bluetooth através do envio de caracteres de um MacBook. O acionamento do controle remoto funcionou bem, entretanto, tornava-se necessário que o usuário soubesse quais os caracteres que deveriam ser enviados ao MP3 para realizar seu controle. Desse modo, para melhorar a usabilidade, facilitando a utilização do controle remoto pelos usuários, desenvolveu-se uma interface de comunicação mais simples e intuitiva, a qual apresenta botões virtuais dotados de símbolos fáceis de ser interpretados, os quais resultam no envio automático de caracteres ao MP3 player.

Assim, neste ponto já tinha-se um MP3 player funcionando bem e dotado de várias funcionalidades e formas de controle, porém o circuito ainda se encontrava montado em protoboard, com conexões elétricas realizadas por meio de fios, resultando em um circuito elétrico não muito firme e de grande espaço físico. Desse modo, com o intuito de tornar o hardware do dispositivo menor e mais rígido, propôs-se a fabricação do circuito em placas de circuito impresso.

Posteriormente projetou-se uma caixa de transporte para o dispositivo, para que ele fosse mais resistente a ação de agentes externos (poeira, choques mecânicos,

temperatura e etc.) e pudesse ser transportado com maior facilidade. E por último, buscou-se melhorar o software do sistema, para torná-lo mais resiliente a erros.

Desse modo, o legado acadêmico deixado por este trabalho é que, a partir da leitura deste relatório final de projeto, outros alunos deverão ser capazes de assimilar pelo menos os princípios básicos relacionados com os tópicos listados abaixo:

- Como Integrar diferentes componentes eletrônicos e analógicos utilizando um microcontrolador;
- Como desenvolver diferentes formas de controle para um dispositivo eletrônico;
- Como desenvolver uma Interface Gráfica do Utilizador (GUI);
- Como fabricar uma Placa de Circuito Impresso;
- Como modelar estruturas 3D.

Outro legado acadêmico deixado é que os esquemas de montagem e as linhas de código apresentadas neste relatório podem ser utilizadas por outros alunos para o desenvolvimento de novos players adaptados a outras propostas de projeto, de acordo com novos cenários de uso.

9. TRABALHOS FUTUROS

Analisando-se os resultados obtidos e o modo de funcionamento do player, foi possível propor algumas modificações a serem realizadas futuramente. Algumas das modificações propostas são:

- Imprimir a estrutura de transporte das PCB's, projetada no item 6.8 deste trabalho, utilizando-se uma impressora 3D;
- Gerar um sinal de erro para avisar ao usuário quando as músicas do SD card não forem lidas com sucesso;
- Tornar possível a transferência de músicas entre o MacBook e o Mp3 via Bluetooth;
- Tornar possível a visualização dos títulos das músicas presentes dentro do SD card do MP3 a partir da interface virtual no MacBook;
- Criar uma nova interface de comunicação adaptada ao PC, para que o player também possa ser utilizado de forma simples quando controlado via PC;
- Adicionar novas funcionalidades ao player, como por exemplo o controle das frequências grave, média e aguda.

9. REFERÊNCIAS

[1] FERREIRA, A., SALEMA, C., TRANCOSO, I., CORREIA, P. L., ASSUNÇÃO, P., FARIA, S. **Comunicações Áudio Visuais: Tecnologias, Normas e Aplicações.** Lisboa, Portugal. 2009.

[2] BARBOSA, L. C. M. **Uma abordagem ao áudio pela perspectiva de ensino**. Porto. Outubro de 2012.

[3] SWART, J. W**. Evolução de Microeletrônica a Micro-Sistemas.** CCS e FEEC – UNICAMP.

[4] **Centro da Memória da Eletricidade no Brasil.** http://www.memoriadaeletricidade.com.br/. Acessado em 10/03/2015.

[5] DUARTE, H. **Players de música têm mais de um século de história.** http://www.techtudo.com.br/noticias/noticia/2014/08/players-de-música-tem-mais-de-um-seculo-de-historia-veja-evolucao.html. Acessado em 10/03/2015.

[6] **Apple's iPod Classic now worth big bucks.** http://www.kgns.tv/home/headlines/Apples-iPod-Classic-now-worth-big-bucks-285501301.html. Acessado em 01/03/2015.

[7] VELOSO, L.R. **Processamento Digital de Sinais.** Universidade Federal de Campina Grande. 2014.

[8] FERREIRA, A. **Tópicos sobre tecnologias para Compressão de Sinais de Áudio.** FEUP. Janeiro de 2005.

[9] PENG, Y., SANDERSON, S. W. **Crossing the chasm with beacon products in the portable music industry**. Technovation. 2013.

[10] KORYCKI, R. **Authenticity examination of compressed audio recordings using detection of multiple compression and enconders'identification.** Forensic Science International. 2013.

[11] KEHTARMAVAZ, N., KIM, N. **Digital Signal Processing - Level Design Using Labview**. University of Texas at Dallas. 2005.

[12] TROST, W., SASCHA, F., SCHON, D., LABBE, C., PICHON, S., GRAND-JEAN, D., VUILLEUMIER, P. **Getting the beat: Entrainment of brain activity by musical rhythm and pleasantness. NeuImage. 2014.**

[13] COCKRILL, A., SULLIVAN, M., NORBURY, H. L. **Music Consumption: Lifestyle choice or addiction**. 2010.

[14] MOREIRA, M. A. **Final Project - EE47**. Stanford University. 2013.

[15] VERPLANK, B. **Interaction Design Sketchbook**. 2009.

[16] SIRKIN, D., SRINIVASAN, A. **PressPlay: Interactive Device Design Song Mp3 Example**. Stanford University. 2011, 2012.

[17] RUSSEL, D. J. **Introduction to Embedded Systems: Using ANSI C and the Arduino Development**. University of Nebraska – Lincoln. 2010.

[18] SIRKIN, D. **PressPlay: Interactive Device Design Lab 6 - Barebones Mp3 Player.** Stanford University. 2013.

[19] **Debouncing.**
http://whatis.techtarget.com/definition/debouncing. Acessado em 12/04/2015.

[20] **SoftwareSerial Library.**
http://arduino.cc/en/Reference/softwareSerial. Acessado em 12/04/2015.

[21] **Arduino and Bluetooth HC-05 Connecting easily.**
http://www.instructables.com/id/Arduino-AND-Bluetooth-HC-05-Connecting-easily/step3/Arduino-Code/. Acessado em 12/04/2015.

ANEXO 1

```
// ---- Inclusão de bibliotecas e definição de variáveis -------------------------------------
#include <SD.h>
#include <EEPROM.h>
#include <mp3.h>
#include <mp3conf.h>

// configuração dos pinos chip select do microsd e do VS1053 e dos pinos específicos do
VS1053.

#define sd_cs         17       // barramento 'chip select' para o microsd card.
#define mp3_cs        A0       // 'command chip select' conectado ao pino cs.
#define mp3_dcs       A1       // 'data chip select' conectado ao pino bsync.
#define mp3_rst       -1       // 'reset' conectado ao pino reset do VS1053.
#define mp3_dreq      A2       // 'data request line' conectado ao pino dreq.

// 'read_buffer' é a quantidade de dados lidos do microSD e enviados ao decodificador
VS1053.

#define read_buffer  512       // tamanho (bytes) do do leitor de buffer do microsd
#define mp3_vol      175       // volume padrão. Pode variar entre min=0 and max=254

// ---- variáveis globais ---------------------------------------------------
File sd_file;                  // objeto que representa um arquivo no microsd.

// variáveis para gravar o número de músicas e a música atual a ser tocada.

unsigned char num_songs = 0, current_song = 0;

//um array para guardar o nome do arquivo da musica atual na memoria ram do arduino.
char fn[max_name_len];

// o programa funciona como uma máquina de estados, a função enum é utilizada para
enumerar os seus estados; 'current_state' é o estado default do programa.

enum state { DIR_PLAY, MP3_PLAY, PAUSED };
state current_state = DIR_PLAY;

//---- Criação de funções -------------------------------------------------------
//é preciso abrir o arquivo de música que se quer escutar a partir de ‘sd_file_open’, antes
de transferir dados desse arquivo. Só se pode abrir um arquivo de cada vez.

void sd_file_open() {
// primeiro deve-se encontrar o nome do arquivo da musica atual na memoria eeprom.
```

```
  get_current_song_as_fn();

//depois deve-se abrir o arquivo usando o nome encontrado e guardado em fn.

  sd_file = SD.open(fn, FILE_READ);
}

//ler um numero de bytes do microsd card, depois encaminhar ele para a função play da
biblioteca do mp3, a qual direcionará os bits para o decodificador VS1053.

void mp3_play() {
  unsigned char bytes[read_buffer]; // buffer para ler e enviar bits ao VS1053.
  unsigned int bytes_to_read;       // número de bits lidos do microSD card.

//primeiramente complete o buffer 'bytes_to_read' com o número máximo de bits
possível, que pode ser no máximo do tamanho do 'read_buffer'. Isso é feito pela função
sd_file.read(). Esta função também é capaz de saber aonde deve começar a ler um novo
número de bits

//a função Mp3.play() envia os bits lidos do sdcard para o decodificador.
  bytes_to_read = sd_file.read(bytes, read_buffer);
  Mp3.play(bytes, bytes_to_read);

  // 'bytes_to_read' só é menor do que 'read_buffer' quando a música termina.

  if (bytes_to_read < read_buffer) {
    sd_file.close();

    // se o último estado era MP3_PLAY, então nós queremos ir para o estado PAUSED.

    if (current_state == MP3_PLAY) {
      current_state == PAUSED;
    }
  }
}

//continue tocando a current song (que fica se atualizando), ate que não haja mais músicas
a serem lidas no diretório.

void dir_play() {
  if (sd_file) {
    mp3_play();
  }

  else {
```

```
//Uma vez que o SD file não está aberto, a música atual deve ter terminado. Neste caso, deve-se ir para a próxima música ou voltar para a primeira música.

    if (current_song < (num_songs - 1)) {
      current_song++;
      sd_file_open();
    }

    else {
      current_song = 0;
      sd_file_open();
      //current_state == PAUSED;
    }
  }
}

// ---- setup e loop -----------------------------------------------------
//o setup é bem intuitivo. Inicializa-se a comunicação serial, a biblioteca do mp3 e os objetos do microSD card, e depois abre na biblioteca raiz a primeira musica a ser tocada.

void setup() {

   //inicialize a biblioteca do mp3 e defina o volume default. 'mp3_cs' é o pino chip select, 'dcs' é o pino chip de dados do chip select, 'rst' é o reset e 'dreq' é solicitação de dados. O pino ‘dreq’ é configurado automaticamente.
  Serial.begin(9600);
  Mp3.begin(mp3_cs, mp3_dcs, mp3_rst, mp3_dreq);
  Mp3.volume(mp3_vol);

//inicializar o microsd card, o qual checa o cartão, o volume e os objetos raiz.

  sd_card_setup();

//a próxima função coloca todas as músicas do diretório raiz na memória eeprom.

  sd_dir_setup();

  //depois abre-se o diretório raiz, antes do programa ir para o estado DIR_PLAY.

  sd_file_open();
}

//o loop contém os 3 estados que devem chamar as seguintes funções: dir_play(), o qual controla a música a ser tocada e mp3_play(), que é a função para reproduzir as músicas.

void loop() {
```

```
  switch(current_state) {

    case DIR_PLAY:
      dir_play();
      break;

    case MP3_PLAY:
      mp3_play();
      break;

    case PAUSED:
      break;
  }
}

// ---- funções  ------------------------------------------------------
// primeiro passo, declaração de variáveis.

File sd_root;                  // a partição do diretório raiz ('/') do SD.

//Depois, deve-se checar se o microSD está presente, se pode ser inicializado e se tem um
volume raiz válido. 'sd_root' é um apontador para objeto raiz do volume do cartão.

void sd_card_setup() {
  if (!SD.begin(sd_cs)) {
    serial.println("sd card failed\nor not present");
    return;
  }
  sd_root = SD.open("/");

  if (!sd_root) {
    serial.println("couldn't mount\nsd root volume");
    return;
  }
}

//para cada música no diretório atual, deve-se salvar o nome do seu arquivo na memoria
EEPROM.

void sd_dir_setup() {
  num_songs = 0;

  sd_root.rewindDirectory();

  while (num_songs < max_num_songs) {
//deve-se sair do laço depois de checar todas as músicas com o comando break.
```

```
    File p = sd_root.openNextFile();
    if (!p) break;

     //Deve-se considerar apenas os arquivos atuais (não deletados), e ignorar os pontos, as entradas dos diretórios e os subdiretórios.

    if (p.name()[0] == '~' || p.name()[0] == '.' || p.isDirectory()) {
      continue;
    }

     //agora, vamos pesquisar a posição do ponto que aparece depois do título da música, o qual deve estar não mais do que 8 letras após o início do título (‘max_name_len’ – 5).

    char i;

    for (i = max_name_len - 5; i > 0; i--) {
      if (p.name()[i] == '.') break;
    }
    i++;

     //agora vamos dizer ao programa quais os formatos de áudio que devem ser lidos e salvos na memória EEPROM do arduino.

    if ((p.name()[i] == 'M' && p.name()[i+1] == 'P' && p.name()[i+2] == '3') ||
        (p.name()[i] == 'W' && p.name()[i+1] == 'A' && p.name()[i+2] == 'V')) {

       //agora vamos salvar o título das músicas, incluindo o caractere ‘\0’ na posição 12, dentro de um byte na EEPROM.

      for (char i = 0; i < max_name_len; i++) {
        EEPROM.write(num_songs * max_name_len + i, p.name()[i]);
      }
      num_songs++;
    }
  }
}
//dado um índice de uma música particular a ser tocada, o programa deve ir para sua posição específica na EEPROM, reter o nome de seu arquivo e setar a variável ‘fn’ para esse nome.

void get_current_song_as_fn() {
  for (char i = 0; i < max_name_len; i++) {
    fn[i] = EEPROM.read(current_song * max_name_len + i);
  }
```