UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO FINAL DE GRADUAÇÃO

PROJETO E IMPLEMENTAÇÃO DE UM SINTETIZADOR SONORO

Brasília

2009

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROJETO E IMPLEMENTAÇÃO DE UM SINTETIZADOR SONORO

Renato Cavalcante de Almeida Gomes
Walkiria Eyre de Oliveira
Marcus Vinicius Guimarães Borges

Professor Orientador: Adson Ferreira Rocha

1. RESUMO

O objetivo deste trabalho é o projeto e implementação de um sintetizador sonoro, voltado para produção musical e criação de sons.

O projeto se divide em dois módulos principais. O primeiro consiste em um gerador sonoro construído utilizando-se hardware de PC. Foi feita esta escolha devido à facilidade de programação, baixo custo e alto poder de processamento.

O segundo módulo consiste em um sistema de controle microprocessado composto por um teclado semelhante ao de um piano, um visor de LCD e alguns botões para configuração dos parâmetros. Estes parâmetros são enviados ao módulo gerador de som através de uma porta serial e um protocolo específico chamado MIDI. O módulo gerador, de posse destes parâmetros, gera o som com características de timbre correspondentes aos parâmetros de entrada.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

2. INTRODUÇÃO

2.1. O que é um sintetizador?

Um sintetizador é um instrumento musical eletrônico capaz de produzir sons a partir de sinais elétricos. Estes sons são produzidos através da geração, combinação e modificação de diferentes sinais. Uma vez produzidos, estes sinais elétricos são amplificados e levados até caixas de som ou fones de ouvido.

O objetivo principal de um sintetizador é criar notas musicais com diversos timbres. O usuário deve informar ao sintetizador qual nota musical deve ser gerada e quais as características do timbre. Existem diversas formas de se controlar um sintetizador. A mais popular é utilizando-se um teclado similar ao de um piano para informar a nota musical desejada e alguns botões no painel para informar as características do timbre a ser obtido.

Podemos dividir os sintetizadores em dois grandes grupos. O primeiro é daqueles que se destinam a produzir sons imitando instrumentos já existentes, com um piano, um saxofone, um violão ou a voz humana. O segundo é daqueles que tem como objetivo produzir sons totalmente novos e inéditos. Neste trabalho o objetivo será o segundo caso.

2.2. Arquitetura escolhida

O sintetizador projetado se utilizará da arquitetura mais usada em sintetizadores comerciais, a chamada síntese subtrativa. Este tipo de síntese é o mais popular, devido à facilidade de operação e os ótimos resultados obtidos. O som é gerado primeiramente por um sinal com muitos harmônicos, como uma onda quadrada ou dente-de-serra, e é posteriormente modelado através de filtros, geradores de envoltória e efeitos sonoros.

2.3. MIDI

A comunicação entre os módulos de controle e geração sonora é feita através de um protocolo digital chamado MIDI (Musical Instrument Digital Interface). O MIDI é um protocolo padrão que permite que diversos dispositivos troquem informação ente si. Por ser um protocolo digital, não sofre dos males inerentes ao mundo analógico, como ruídos e instabilidade devido à variação de temperatura.

O MIDI é um sistema de comunicação digital simplex, serial e assíncrono, o que significa que o sinal só percorre em uma direção, só necessita de dois condutores (um condutor de referência e um para o sinal), e não necessita de pulsos de clock para que os dispositivos entrem em sincronia. A transmissão se dá a 31500 bauds e segue o padrão

8N1, o que sinifica 8 bits por palavra, sem paridade, um start bit que deve ser 0 e um stop bit que deve ser 1. Para evitar “ground loops”, somente uma ponta da conexão está referenciada ao terra (no transmissor), enquanto a outra ponta (receptor) é isolada opticamente através de um opto-coupler. Um opto-coupler nada mais é que um encapsulamento com um LED (diodo emissor de luz) e um LDR (resistor dependente da luz). O sinal de transmissão na verdade está conectado ao LED, e só flui corrente no LDR quando o LED está acesso, o que significa que o sinal está isolado eletricamente e acoplado opticamente. A maioria dos opto-couplers são relativamnte lentos e possuem baixos slew-rates, o que limita a quantidade de dispositivos MIDI que podem ser ligados em cascata, já que o baixo slew-rate pode distorcer demais o sinal e impedir que o dispositivo receptor entenda a mensagem. O sinal de transmissão MIDI segue o seguinte padrão:

Nível lógico 1 => sem corrente fluindo => LED apagado => sem corrente fluindo no LDR

Nível lógico 0 => com corrente fluindo => LED aceso => com corrente fluindo no LDR

As palavras de 8 bits transmitidas pelo protocolo são padronizadas e associadas a comandos específicos. Cada palavra de 8 bits (um byte) corresponde a um comando MIDI. A notação utilizada será a hexadecimal, onde cada byte é representado por um número de 0x00 a 0xFF.

Todo comando MIDI é composto por um *status byte* e um ou mais *data bytes*. O *status byte* informa qual o comando (parâmetro) a ser transmitido e o *data byte* informa o valor do parâmetro em questão. Em binário, todo *status byte* inicia com “1”, o que significa que em decimal os *status bytes* pertencem ao *range* de 128 a 256. Já os *data bytes* iniciam sempre em “0”, preenchendo um *range* de 0 a 127. Esta diferenciação permite ao dispositivo receptor distinguir um *status byte* de um *data byte* de maneira mais eficiente.

Os comandos utilizados neste projeto são os seguintes:

* 0x90 – NOTE ON – Indica que uma nota musical deverá ser tocada. Este *status byte* deve ser seguido de dois *data bytes*, 0xAA e 0xBB, onde AA é a nota musical a ser tocada e BB é a intensidade com que a mesma deverá ser executada.

* 0x80 – NOTE OFF – Indica que uma nota musical deve ser interrompida. Este *status byte* deve ser seguido de dois *data bytes*, 0xAA e 0xBB, onde AA é a nota musical a ser interrompida e BB é a intensidade com que a mesma foi solta.

* 0xB0 – CONTROL CHANGE – Indica que um parâmetro de controle será enviado em seguida.

* 0x63 e 0x62 – NRPN (Non-Registered Parameter Number) – São parâmetros de controle reservados para envio de quaisquer parâmetros de configuração entre

dispositivos. Neste projeto este comando é utilizado para enviar os parâmetros de configuração do módulo de controle para o módulo gerador de som. Comandos NRPN são definidos por 2 bytes. Por serem *data bytes*¸ cada um pode assumir um valor de 0 a 127 (0x00 a 0x7F). NRPNs possibilitam a transmissão de até 16129 parâmetros diferentes. O comando 0x99 é seguido pelo byte mais significativo enquanto o comando 0x98 é seguido pelo byte menos significativo. Por serem parâmetros de controle devem ser precedidos por 0xB0.

* 0x06 – DATA ENTRY – Indica que o próximo byte enviado é o valor do último parâmetro NRPN escolhido. Por ser um parâmetro de controle deve ser precedido por 0xB0.

Exemplos:

a) Ao enviar os bytes 0x90, 0x3C e 0x7F é transmitida a informação de tocar a nota correspondente ao número hexadecimal 3C (referente ao dó central do piano), com intensidade 7F (intensidade máxima).

b) Ao enviar os bytes 0xB0, 0x63 e 0x00 seguidos de 0xB0, 0x62 e 0x02 é transmitida a informação para selecionar o parâmetro NRPN de número 0x0002.

c) Ao enviar os bytes 0xB0, 0x06 e 0x30 é transmitida a informação de alterar o valor do ultimo parâmetro NRPN selecionado para 0x30.

3. O PROJETO

3.1. Módulo de geração sonora

Este módulo consiste em um PC rodando um software de geração sonora em ambiente Windows. Este software é construído baseado na tecnologia VST (Virtual Studio Technology).

A tecnologia VST é um conjunto de bibliotecas que permitem a programação de softwares voltados para gravação, produção musical e áudio profissional. É uma tecnologia amplamente empregada no mercado atual e tornou-se padrão em aplicações de áudio profissional para PC, principalmente em ambiente Windows.

Uma das grandes vantagens de seu uso é que o programador não precisa implementar as funções de entrada/saída pois tudo é tratado automaticamente pelas bibliotecas e softwares. A preocupação do programador é somente de projetar as funções do sintetizador.

A criação de um módulo VST pode ser feita de duas formas. Programação em linguagem C ou utilizando uma linguagem visual em um ambiente gráfico de desenvolvimento. Foi escolhida a segunda opção pois oferece maior velocidade e versatilidade no processo de desenvolvimento e programação. O software de programação visual utilizado neste trabalho, Cinestar Synthedit, é de utilização gratuita no regime de *shareware*.

O projeto do módulo VST no software Cinestar Synthedit pode ser visto no anexo 04.

O módulo VST, depois de compilado, é salvo em um arquivo de extensão .dll. Tal arquivo, para ser devidamente executado e utilizado, necessita ser aberto em um software *host*, que faz a integração do módulo VST com as portas de entrada/saída do PC.

As portas de entrada/saída utilizadas neste trabalho são:

- ENTRADAS: porta MIDI, via pela qual o módulo gerador de som recebe as informações necessárias para seu funcionamento oriundas do módulo de controle.

- SAÍDAS: saída de áudio estéreo da placa de som do PC, composta por um canal esquerdo e um canal direito.

O módulo de controle envia via protocolo MIDI a informação de qual tecla foi pressionada, e quais são os parâmetros escolhidos para geração sonora. Estas informações são recebidas pelo PC através da porta MIDI. O software *host* recebe estas informações e as repassa para o módulo VST. De posse destas informações o módulo

VST efetua todo o processamento necessário e tem como saída um sinal sonoro correspondente. Este sinal resultante é então devolvido ao software *host* que por sua vez leva este sinal à saída de áudio da placa de som do PC.

O funcionamento básico do módulo de geração sonora é dado pelo seguinte diagrama de blocos:

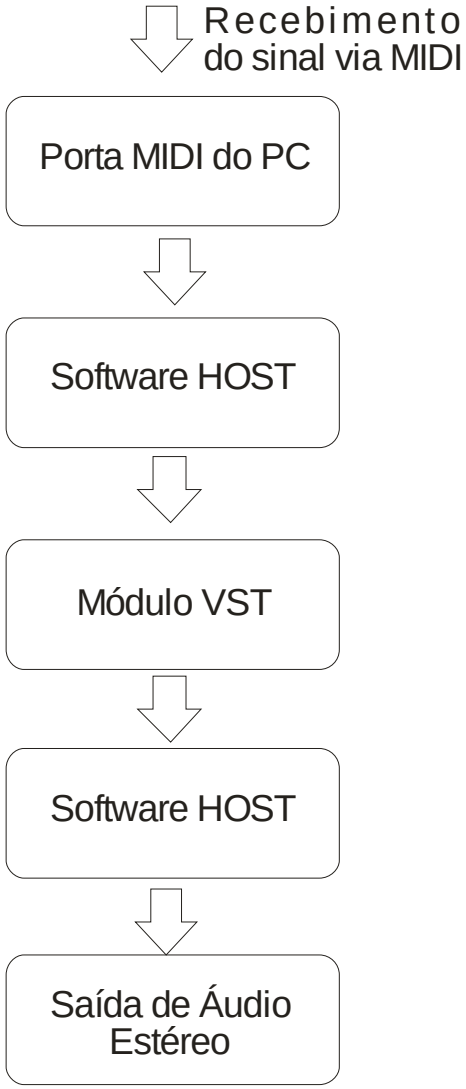


Figura 1 – Diagrama de blocos básico do módulo de geração sonora

O módulo VST possui o seguinte diagrama de blocos:

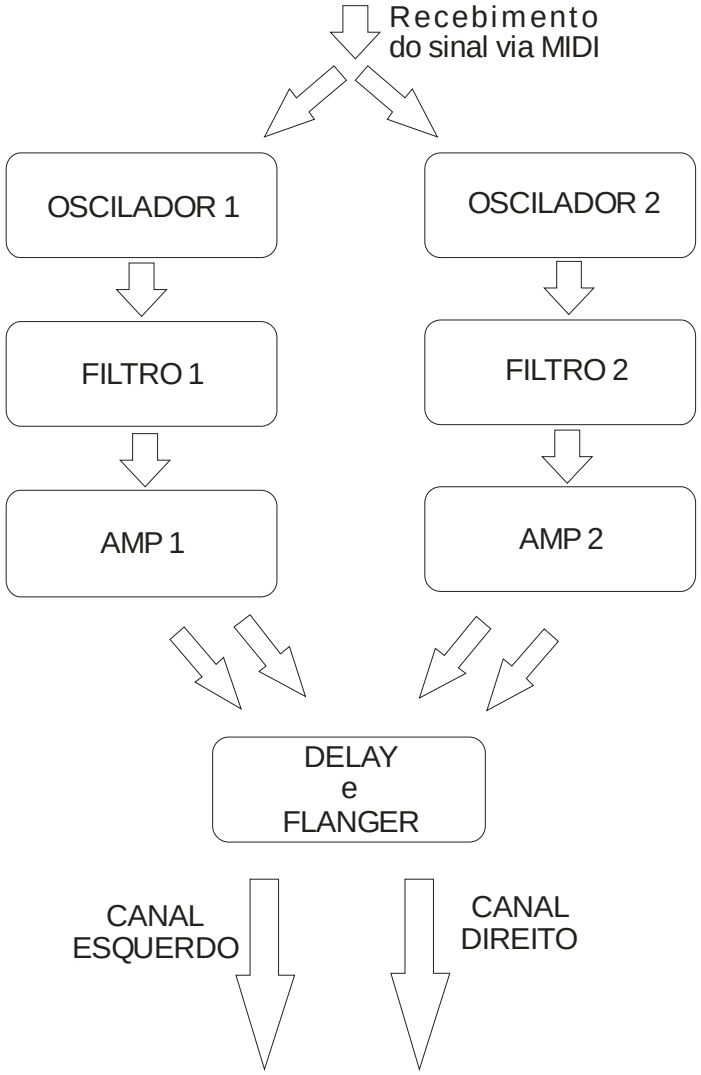


Figura 2 – Diagrama de blocos do funcionamento interno do sintetizador VST

O som é inicialmente gerado nos osciladores, que tem como saída uma forma de onda básica. O sintetizador implementado neste trabalho possui 2 osciladores. Cada um pode ser configurado independentemente e pode reagir de maneira diferente às informações de entrada. A frequência da forma de onda gerada depende da nota musical pressionada. Para cada nota musical temos uma frequência correspondente. A frequência de referência é 440Hz, que corresponde à nota LÁ central do piano. A partir desta, subindo uma oitava a frequência é dobrada e descendo uma oitava a frequência é dividida por dois. Uma tabela contendo as frequências das 128 notas musicais suportadas pelo protocolo MIDI pode ser vista a seguir:

Nota musical	Oitava	Número MIDI	Frequência
C	-5	0	8.1757989156
C#/Db	-5	1	8.6619572180
D	-5	2	9.1770239974
D#/Eb	-5	3	10.3008611535
E	-5	4	10.3008611535
F	-5	5	10.9133822323
F#/Gb	-5	6	11.5623257097
G	-5	7	12.2498573744
G#/Ab	-5	8	12.9782717994
A	-5	9	13.7500000000
A#/Bb	-5	10	14.5676175474
B	-5	11	15.4338531643
C	-4	12	16.3515978313
C#/Db	-4	13	17.3239144361
D	-4	14	18.3540479948
D#/Eb	-4	15	19.4454364826
E	-4	16	20.6017223071
F	-4	17	21.8267644646
F#/Gb	-4	18	23.1246514195
G	-4	19	24.4997147489
G#/Ab	-4	20	25.9565435987
A	-4	21	27.5000000000
A#/Bb	-4	22	29.1352350949
B	-4	23	30.8677063285
C	-3	24	32.7031956626
C#/Db	-3	25	34.6478288721
D	-3	26	36.7080959897
D#/Eb	-3	27	38.8908729653
E	-3	28	41.2034446141
F	-3	29	43.6535289291
F#/Gb	-3	30	46.2493028390
G	-3	31	48.9994294977
G#/Ab	-3	32	51.9130871975
A	-3	33	55.0000000000
A#/Bb	-3	34	58.2704701898
B	-3	35	61.7354126570
C	-2	36	65.4063913251
C#/Db	-2	37	69.2956577442
D	-2	38	73.4161919794
D#/Eb	-2	39	77.7817459305
E	-2	40	82.4068892282
F	-2	41	87.3070578583
F#/Gb	-2	42	92.4986056779
G	-2	43	97.9988589954
G#/Ab	-2	44	103.8261743950
A	-2	45	110.0000000000
A#/Bb	-2	46	116.5409403795
B	-2	47	123.4708253140
C	-1	48	130.8127826503

C#/Db	-1	49	138.5913154884
D	-1	50	146.8323839587
D#/Eb	-1	51	155.5634918610
E	-1	52	164.8137784564
F	-1	53	174.6141157165
F#/Gb	-1	54	184.9972113558
G	-1	55	195.9977179909
G#/Ab	-1	56	207.6523487900
A	-1	57	220.0000000000
A#/Bb	-1	58	233.0818807590
B	-1	59	246.9416506281
C	0	60	261.6255653006
C#/Db	0	61	277.1826309769
D	0	62	293.6647679174
D#/Eb	0	63	311.1269837221
E	0	64	329.6275569129
F	0	65	349.2282314330
F#/Gb	0	66	369.9944227116
G	0	67	391.9954359817
G#/Ab	0	68	415.3046975799
A	0	69	440.0000000000
A#/Bb	0	70	466.1637615181
B	0	71	493.8833012561
C	1	72	523.2511306012
C#/Db	1	73	554.3652619537
D	1	74	587.3295358348
D#/Eb	1	75	622.2539674442
E	1	76	659.2551138257
F	1	77	698.4564628660
F#/Gb	1	78	739.9888454233
G	1	79	783.9908719635
G#/Ab	1	80	830.6093951599
A	1	81	880.0000000000
A#/Bb	1	82	932.3275230362
B	1	83	987.7666025122
C	2	84	1,046.5022612024
C#/Db	2	85	1,108.7305239075
D	2	86	1,174.6590716696
D#/Eb	2	87	1,244.5079348883
E	2	88	1,318.5102276515
F	2	89	1,396.9129257320
F#/Gb	2	90	1,479.9776908465
G	2	91	1,567.9817439270
G#/Ab	2	92	1,661.2187903198
A	2	93	1,760.0000000000
A#/Bb	2	94	1,864.6550460724
B	2	95	1,975.5332050245
C	3	96	2,093.0045224048
C#/Db	3	97	2,217.4610478150
D	3	98	2,349.3181433393

D#/Eb	3	99	2,489.0158697766
E	3	100	2,637.0204553030
F	3	101	2,793.8258514640
F#/Gb	3	102	2,959.9553816931
G	3	103	3,135.9634878540
G#/Ab	3	104	3,322.4375806396
A	3	105	3,520.0000000000
A#/Bb	3	106	3,729.3100921447
B	3	107	3,951.0664100490
C	4	108	4,186.0090448096
C#/Db	4	109	4,434.9220956300
D	4	110	4,698.6362866785
D#/Eb	4	111	4,978.0317395533
E	4	112	5,274.0409106059
F	4	113	5,587.6517029281
F#/Gb	4	114	5,919.9107633862
G	4	115	5,919.9107633862
G#/Ab	4	116	6,644.8751612791
A	4	117	7,040.0000000000
A#/Bb	4	118	7,458.6201842894
B	4	119	7,902.1328200980
C	5	120	8,372.0180896192
C#/Db	5	121	8,869.8441912599
D	5	122	9,397.2725733570
D#/Eb	5	123	9,956.0634791066
E	5	124	10,548.0818212118
F	5	125	11,175.3034058561
F#/Gb	5	126	11,839.8215267723
G	5	127	12,543.8539514160

Tabela 1 – Notas musicais e suas respectivas frequências

Cada oscilador possui também um sub-oscilador, que gera uma forma de onda idêntica à do oscilador porém com metade de sua frequência, isto é, uma oitava abaixo. O volume do sub-oscilador pode ser controlado independentemente.

O sinal de saída dos osciladores é então levado aos filtros, chamados de Filter 1 e Filter 2. Ambos os filtros são configuráveis independentemente. O filtro 1 processa o sinal vindo do oscilador 1 e o filtro 2 processa o som vindo do oscilador 2, podendo ser passa-baixas, passa-faixas, passa-banda ou rejeita-banda. Esta é uma etapa importante na característica sonora, pois são os filtros que retiram determinados harmônicos presentes nos sinais vindos dos osciladores.

O sintetizador possui também dois amplificadores, chamados de Amplifier 1 e Amplifier 2. O chamado amplificador 1 processa o sinal vindo do filtro 1 enquanto o amplificador 2 processa o sinal vindo do filtro 2. Este processamento consiste em ajuste

de volume, geração de envoltória de amplitude e balanço de volume entre os canais esquerdo e direito, com ajustes independente para ambos.

Os sinais resultantes do amplificador 1 e amplificador 2 são então mesclados em apenas um canal esquerdo e um canal direito, onde passam por efeitos sonoros chamados *flanger* e *delay*, comentados posteriormente neste trabalho.

O sintetizador conta também com 4 geradores de envoltória e 2 LFOs (Low Frequency Oscilators, ou Osciladores de Baixa Frequência) que podem ser configurados para atuar sobre diversos parâmetros. Os parâmetros que podem ser controlados por eles são:

Parâmetro	Descrição
Pulse Width	Largura de pulso do sinal “pulse” dos osciladores
Pitch	Frequência do sinal dos osciladores
Cutoff	Frequência de corte dos filtros
Resonance	Ressonância dos filtros
Level	Volume do amp 1 e amp 2
Pan	Balanço de volume entre os canais esquerdo e direito

Tabela 2 – Parâmetros controláveis pelos geradores de envoltória e LFOs

3.2. Módulo de controle

O módulo de controle consiste em um teclado semelhante ao de um piano, um visor de LCD modelo AGTechnologies AGM 1602B e alguns botões para configuração dos parâmetros.

Todo o sistema é processado por um microcontrolador PIC18F452, fabricado pela Microchip. Este foi o microprocessador de escolha devido à facilidade de programação, ampla oferta de literatura gratuita na internet e bom desempenho.

Foi utilizada a linguagem C para programação do dispositivo. Todo o desenvolvimento foi realizado no ambiente gráfico MPLab, disponibilizado gratuitamente pela Microchip.

O teclado é composto por 61 teclas. Cada tecla, ao ser pressionada, ativa uma chave mecânica que faz contato entre dois pinos de um conector, formando uma matriz de contatos. O diagrama esquemático da matriz de contatos do teclado é mostrado a seguir:

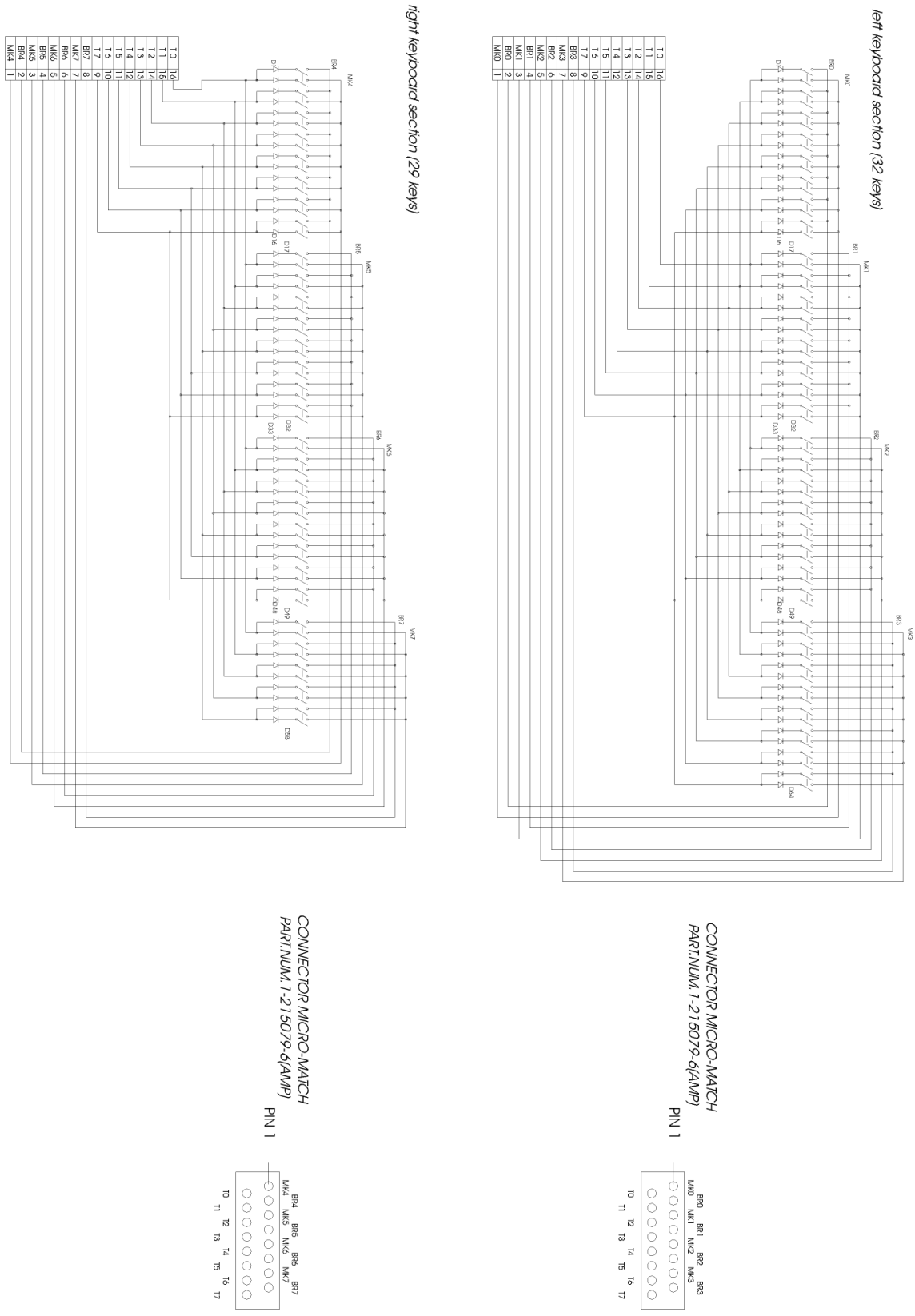


Figura 3 – Diagrama esquemático da matriz de contatos do teclado de 61 teclas

O microcontrolador, após identificar qual tecla foi pressionada ou liberada, envia a informação para o módulo gerador de som através de uma porta serial utilizando o protocolo MIDI.

O visor de LCD permite a visualização e configuração de todos os parâmetros do sintetizador.

É possível configurar cada um dos 91 parâmetros de configuração através de 7 botões. São eles:

Botão	Descrição
Menu + **Menu -**	Permitem a navegação através dos 17 menus de configuração
Param + **Param -**	Permitem a navegação atraves dos parâmetros existentes em cada menu de configuração
Valor + **Valor -**	Permitem o ajuste do valor de cada parâmetro
Valor	Botão giratório que permite o ajuste do valor de cada parâmetro de maneira mais rápida

Tabela 3 – Botões utilizados para navegação no menu e alteração dos parâmetros

O diagrama de blocos do módulo de controle pode ser visto na figura a seguir:

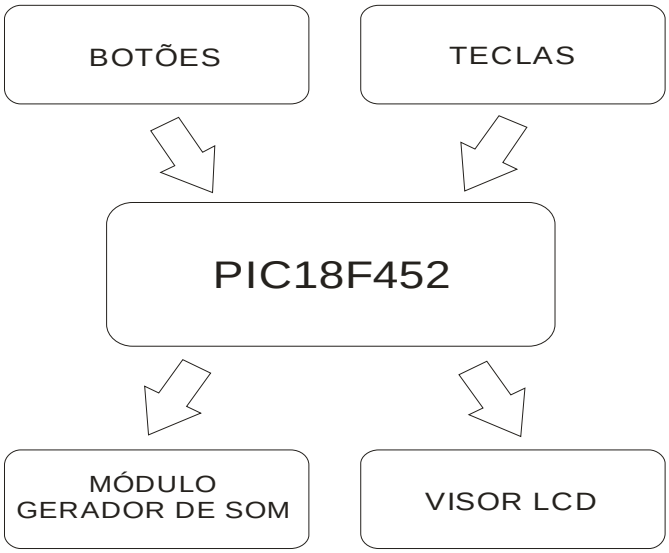


Figura 4 – Diagrama de blocos do módulo de controle

O diagrama esquemático do módulo de controle pode ser visto no anexo 03. O código fonte do microcontrolador PIC18F452 pode ser visto no anexo 02.

O microcontrolador identifica quando algum dos 7 botões foi pressionado e realiza a tarefa correspondente, seja exibindo na tela o parâmetro selecionado ou alterando seu valor. Os botões “Menu+”, “Menu-“, “Param+” e “Param-“ desencadeiam somente as funções de navegação do menu. Os botões “Valor+”, “Valor-“ e o botão giratório “Valor” desencadeiam as funções de armazenamento dos novos valores dos parâmetros e também as funções de envio destes valores ao módulo de geração sonora. Desta maneira economiza-se poder de processamento e não ocupa-se o canal de comunicação, já que a informação dos parâmetros só é enviada quando algum valor é modificado.

O software contido no microcontrolador executa as tarefas em um loop infinito seguindo o seguinte diagrama:

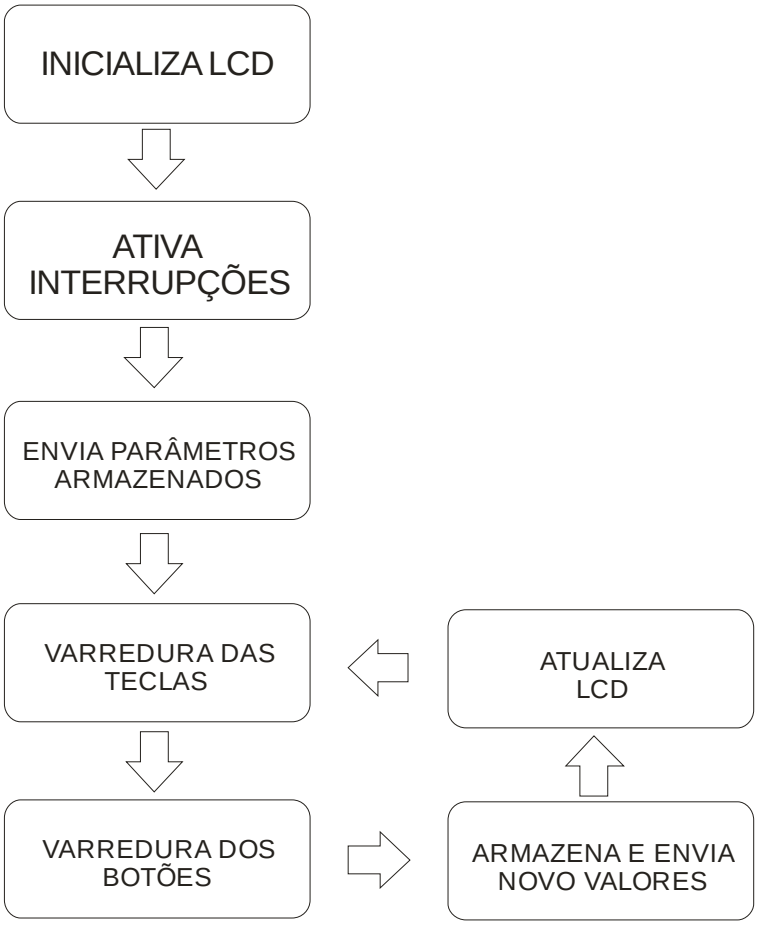


Figura 5 – Diagrama de ações do microcontrolador

3.3. Parâmetros de configuração

Cada parâmetro de configuração tem um efeito específico sobre o som e é enviado do módulo de controle para o módulo gerador de som através de mensagens NRPN. Cada parâmetro responde a um número NRPN específico.

Os parâmetros de configuração, suas possíveis faixas de valores e uma breve descrição são listados a seguir:

MENU 0 – CONFIG

Permite a configuração de parâmetros gerais que afetam o funcionamento de todo o sintetizador.

- Portamento – faixa de valores: [0...127] - Permite uma mudança gradual na frequência entre duas notas musicais tocadas consecutivamente. Valor 0 resulta em uma mudança instantânea na frequência do som. Valor 127 resulta em um longo tempo de transição entre a frequência das duas notas tocadas consecutivamente.

- Mono Mode – faixa de valores: [ON/OFF] – Quando colocado em ON, este parâmetro limita a quantidade de notas tocadas simultaneamente a uma única nota. Ao tocar uma nova nota, o som da nota anteriormente pressionada é cortado abruptamente.

- Retrigger – faixa de valores: [ON/OFF] – Quando colocado em ON, este parâmetro força um reinício em todos os geradores de envoltória sempre que uma nova nota é tocada. Se colocado em OFF, os geradors de envoltória só serão reiniciados se uma nota for pressionada após a nota anterior ter sido liberada. Só surte efeito se o parâmetro “Mono Mode” estiver colocado em ON.

- OSC Sync – faixa de valores: [ON/OFF] – Quando colocado em ON, este parâmetro sincroniza os dois osciladores do sintetizador. Isto significa que toda vez que

o sinal do oscilador 2 inicia um novo ciclo (isto é, no momento que o sinal cruza o valor zero indo do positivo pro negativo), a fase do sinal do oscilador 1 é levada a 0. Este recurso fornece resultados bastante interessantes quando a frequência do oscilador 1 é variada independentemente do oscilador 2, por exemplo, utilizando-se um LFO ou um gerador de envoltória (envelope, em inglês) aplicado ao Pitch Modulation do oscilador 1. O efeito sonoro resultante se assemelha a um filtro passa-baixas com altíssima ressonância.

MENU 1 e 2 – OSCILATOR 1 e 2

Permite a configuração dos dois osciladores básicos.

- Wave – faixa de valores: [Sine/Sawtooth/Ramp/Triangle/Pulse/WhiteNoise/ PinkNoise] – Seleciona a forma de onda gerada pelos osciladores. Permite a escolha entre onda senoidal, dente-de-serra, função rampa, onda triangular, pulso, ruído branco e ruído rosa. Quando a onda pulso é selecionada, o parâmetro "Pulse Width" passa a fazer efeito no som resultante. O White Noise (ruído branco) é um som produzido pela combinação de sons de todas as frequências audíveis em igual amplitude. O Pink Noise (ruído rosa) é um som que também possui todas as frequências audíveis, mas cuja amplitude do espectro de frequências é inversamente proporcional à frequência. White Noise e Pink Noise produzem o mesmo som independente da nota musical pressionada.

- Pulse Width – faixa de valores: [1%...99%] – Varia a largura do trecho positivo do pulso em relação ao trecho negativo. Só surte efeito no som se a onda "pulse" for selecionada no parâmetro "wave".

- Pulse Width Modulation – faixa de valores: [ENV1/ENV2/ENV3/ENV4/LFO1/ LFO2/MOD/VEL] – Permite selecionar qual fonte de modulação irá afetar a largura de pulso configurada em “Pulse Width”.

- Pulse Width Modulation Level - faixa de valores: [0...127] – Ajusta o quanto a fonte de modulação selecionada em “Pulse Width Modulation” irá afetar a largura de pulso.

- Pitch - faixa de valores: [-36...+36] – Permite transpor a frequência dos osciladores em intervalos de 1 semitom.

- Fine - faixa de valores: [-50...+50] – Permite modificar a frequência dos osciladores em intervalos de um centésimo de semitom.

- Pitch Modulation - faixa de valores: [ENV1/ENV2/ENV3/ENV4/LFO1/LFO2/ MOD/VEL] - Permite selecionar qual fonte de modulação irá afetar a frequência do oscilador.

- Pitch Modulation Level - faixa de valores: [0...127] – Ajusta o quanto a fonte de modulação selecionada em “Pitch Modulation” irá afetar a frequência do oscilador.

- Wave Level – faixa de valores: [0...127] – Ajusta o volume principal do oscilador.

- Sub Level – faixa de valores: [0...127] – Cada oscilador possui um sub-oscilador que gera um sinal de igual timbre mas com a metade de sua frequência, isto é, uma oitava abaixo do som principal. Este parâmetro ajusta seu volume.

- Sub Detune – faixa de valores: [-50...+50] – Ajusta a frequência do sub-oscilador em intervalos de centésimos de semitom.

MENU 3 e 4 – FILTER 1 e 2

Permite a configuração dos dois filtros digitais do sintetizador. O filtro 1 atua sobre o oscilador 1 e o filtro 2 atua sobre o oscilador 2.

- ON/OFF – faixa de valores: [ON/OFF] – Liga ou desliga o filtro.

- Type – faixa de valores: [LOW-PASS/HIGH-PASS/BAND-PASS/BAND-REJECT] – Seleciona o tipo de filtro entre passa-baixas, passa-altas, passa-faixa e rejeita-faixa.

- Cutoff – faixa de valores: [0...127] – Ajusta a frequência de corte do filtro. Quanto mais alto o valor, maior a frequência.

- Resonance – faixa de valores: [0...127] – Ajusta a quantidade de ressonância do filtro. Quanto mais alto o valor, maior o pico na frequência de corte do filtro.

- Cutoff Modulation - faixa de valores: [ENV1/ENV2/ENV3/ENV4/LFO1/LFO2/ MOD/VEL] - Permite selecionar qual fonte de modulação irá afetar a frequência de corte do filtro.

- Cutoff Modulation Level – faixa de valores: [0...127] – Ajusta o quanto a fonte de modulação selecionada em “Cutoff Modulation” irá afetar a frequência de corte do filtro.

- Resonance Modulation - faixa de valores: [ENV1/ENV2/ENV3/ENV4/LFO1/ LFO2/MOD/VEL] - Permite selecionar qual fonte de modulação irá afetar a ressonância do filtro.

- Resonance Modulation Level – faixa de valores: [0...127] – Ajusta o quanto a fonte de modulação selecionada em “Resonance Modulation” irá afetar o parâmetro de ressonância do filtro.

MENU 5 e 7 – AMPLIFIER 1 e 2

Permite o ajuste de parâmetros que afetam a amplitude do sinal gerado. O AMPLIFIER 1 age sobre o som oriundo do FILTER 1 e o AMPLIFIER 2 age sobre o som oriundo do FILTER 2.

- Level – faixa de valores: [0...127] – Ajusta o volume geral do som que se originou no oscilador 1 e passou pelo filtro 1.

- Amplifier Modulation - faixa de valores: [ENV1/ENV2/ENV3/ENV4/LFO1 /LFO2/MOD/VEL] - Permite selecionar qual fonte de modulaão irá afetar o volume do Amplifier 1.

- Amplifier Modulation Level – faixa de valores: [0...127] – Ajusta o quanto a fonte de modulação selecionada em “Amplifier Modulation” irá afetar o volume do Amplifier 1.

- Pan – faixa de valores: [-63...+63] – Permite o ajuste da centralização do som do Amplifier 1 – Um valor de -63 significa que o som será direcionado totalmente ao canal esquerdo. Um valor de +63 significa que o som será direcionado totalmente ao canal direito. Valor zero indica que o som será reproduzido igualmente nos canais esquerdo e direito.

- Pan Modulation - faixa de valores: [ENV1/ENV2/ENV3/ENV4/LFO1/LFO2/ MOD/VEL] - Permite selecionar qual fonte de modulação irá afetar o pan do Amplifier 1.

- Pan Modulation Level – faixa de valores: [0...127] – Ajusta o quanto a fonte de modulação selecionada em “Pan Modulation” irá afetar o Pan do Amplifier 1.

MENU 6 e 8 – AMP ENVELOPE 1 e 2

Permite o ajuste do envelope (gerador de envoltória) dedicado exclusivamente a agir sobre o volume do amplifier 1 e amplifier 2. Se os parâmetros “Mono Mode” e “Retrigger” estiverem ambos em ON, este envelope será reiniciado toda vez que uma nova tecla for pressionada, mesmo que a nota anterior não seja liberada. Este envelope possui os seguintes parâmetros:

- Attack – faixa de valores: [0...127] – Ajusta o tempo que o volume leva para ir do valor zero ao seu valor máximo. Quanto maior o valor, maior o tempo.

- Decay - faixa de valores: [0...127] – Ajusta o tempo que o volume leva para ir do valor máximo atingido após o tempo de attack até o valor configurado em “sustain”. Quanto maior o valor, maior o tempo.

- Sustain – faixa de valores: [0...127] – Ajusta o volume que o amplifier 1 ou 2 irá assumir após o decorrer dos tempos de Attack e Decay. O volume permanecerá no valor configurado neste parâmetro enquanto a nota estiver sendo pressionada.

- Release – faixa de valores: [0...127] – Ao liberar uma nota, o volume do som levará um tempo para ir a zero. Este parâmetro ajusta este tempo. Valor zero significa que ao ser liberada a nota, o volume irá instantaneamente a zero.

MENU 9, 10, 11 e 12 – ENVELOPE 1, 2, 3 e 4

Permitem o ajuste dos parâmetros de Envelope 1, 2, 3 e 4. Estes quatro envelopes (geradores de envoltória) podem ser associados a vários parâmetros do sintetizador, bastando para isso seleciona-los nos parâmetros “XXX Modulation” e ajustar o quanto o envelope irá afetar o parâmetro em “XXX Modulation Level”, onde XXX é um parâmetro qualquer de escolha.

- Attack – faixa de valores: [0...127] – Ajusta o tempo que o envelope leva para ir do valor zero ao seu valor máximo. Quanto maior o valor, maior o tempo.

- Decay - faixa de valores: [0...127] – Ajusta o tempo que o envelope leva para ir do valor máximo atingido após o tempo de attack até o valor configurado em “sustain”. Quanto maior o valor, maior o tempo.

- Sustain – faixa de valores: [0...127] – Ajusta o volume que o envelope irá assumir após o decorrer dos tempos de Attack e Decay. O volume permanecerá no valor configurado neste parâmetro enquanto a nota estiver sendo pressionada.

- Release – faixa de valores: [0...127] – Ao liberar uma nota, o envelope levará um tempo para ir a zero. Este parâmetro ajusta este tempo. Valor zero significa que ao ser liberada o envelope irá instantaneamente a zero.

MENU 13 e 14 – LFO 1 e 2

Permite o ajuste dos LFOs (Low Frequency Oscilators, ou osciladores de baixa frequencia). Estes osciladores permitem uma mudança cíclica e contínua em diversos parâmetros do sintetizador, bastando para isso seleciona-los nos parâmetros “XXX Modulation” e ajustar o quanto o LFO irá afetar o parâmetro em “XXX Modulation Level”, onde XXX é um parâmetro qualquer de escolha.

- Wave – faixa de valores: [Sine/Sawtooth/Ramp/Triangle/Pulse/WhiteNoise/ PinkNoise] – Seleciona a forma de onda gerada pelos LFOs. Permite a escolha entre

onda senoidal, dente-de-serra, função rampa, onda triangular, pulso, ruído branco e ruído rosa. Os dois ruídos permitem uma operação pseudo-randômica do LFO.

- Pulse Width - faixa de valores: [1%...99%] – Varia a largura do trecho positivo do pulso em relação ao trecho negativo. Só surte efeito no som se a onda “pulse” for selecionada no parâmetro “wave”.

- Frequency – faixa de valores: [0...127] – Ajusta a frequência de operação do LFO. Quanto mais alto o valor, maior a frequência.

MENU 15 – DELAY

Permite o ajuste do efeito de *delay* no som do sintetizador. Este efeito consiste na repetição do som já tocado, proporcionando a sensação de ricocheteamento do som. Os parâmetros permitem ajuste independente de tempo de repetição nos canais esquedo e direito.

- Time L – faixa de valores: [0...127] – Ajusta o tempo das repetições no canal esquerdo. Quanto mais alto o valor, maior o intervalo de tempo entre as repetições.

- Time R – faixa de valores: [0...127] – Ajusta o tempo das repetições no canal direito. Quanto mais alto o valor, maior o intervalo de tempo entre as repetições.

- Feedback – faixa de valores: [0...127] – Permite o ajuste da quantidade de repetições, que vão diminuindo de volume com o tempo. Quanto mais alto o valor, maior o número de repetições do *delay* antes que o efeito cesse.

- Mix – faixa de valores: [0%...100%] – Ajusta o balanço de volume entre o sinal original e o sinal repetido. 0% significa que não será ouvida nenhuma repetição. 100% significa que o som original não será ouvido, somente as repetições. 50% significa que o som original e o repetido possuem o mesmo volume.

MENU 16 – FLANGER

Permite o ajuste do efeito de *flanger* no som do sintetizador. Este efeito consiste na divisão do som em dois caminhos. Um tem sua frequência alterada variando no tempo, assumindo valores um pouco menores e um pouco maiores que o valor original. O outro caminho permanece inalterado. Os dois sinais são então reunidos, causando um interessante efeito de cancelamento de frequências que varia no tempo.

- Rate – faixa de valores: [0...127] – Ajusta a frequência com que o som alterado varia sua frequência. Valores altos resultam em uma frequência alta.

- Feedback – faixa de valores: [0...127] – Permite o ajuste do quanto a frequênca do som alterado irá ser modificada para mais e para menos. Valores mais altos resultam em maiores variações de frequência.

- Mix – faixa de valores: [0%...100%] - Ajusta o balanço de volume entre o sinal original e o sinal modificado. 0% significa que o sinal modificado não é reunido com o sinal original. 100% significa que o som modificado terá a mesma amplitude (volume) do som original.

Uma tabela reunindo todos os parâmetros e suas respectivas faixas de valores e números NRPN pode ser vista no anexo 01.

4. CONCLUSÕES

5. REFERÊNCIAS BIBLIOGRÁFICAS

[1] FORTUNE; SCHOFFHAUSER; HAUPT. Visual VST/i Programming. A Comprehensive Guide to Creating VST-FX and Instruments with Synthedit. Wizoo, 2007.

[2] ZELENOVSKY, Ricardo; MENDONÇA, Alexandre. Programação e Projeto com a Família 8051. MZ Editora, 2005.

[3] HOLZNER, Steven. C++ Black Book. Makron Books, 2002

[4] CSS C Compiler Help File Ver 4.

[5] Synthedit Help File Ver 1.0170.

[6] BERLIM, Ricardo. Implementação de uma DAW utilizando Software Livre. Brasília, 2007.

[7] Summary of MIDI Messages. Disponível em: http://www.midi.org/techspecs/midimessages.php Último acesso em 14 jul 2009.

[8] PIC18F452 Datasheet, Microchip Technologies Inc, 2007.

6. ANEXOS

6.1. ANEXO 01 – TABELA DE PARÂMETROS

Menu - Parâmetro	Parâmetro	Faixa de valores	Número NRPN
0 - CONFIG			
0	Portamento	ON/OFF	0
1	Mono Mode	ON/OFF	1
2	Retrigger	ON/OFF	2
3	OSC Sync	ON/OFF	3
1 - OSCILATOR 1			
4	wave	Sine, Sawtooth, Ramp, Pulse, Triangle, White Noite, Pink Noise	4
5	pulse width	1% - 99%	5
6	pwm	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	6
7	pwm level	0 ... 127	7
8	pitch	-36 ... +36	8
9	fine	-50 ... +50	9
10	pitch mod	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	10
11	pmod level	0 ... 127	11
12	wave level	0 ... 127	12
13	sub level	0 ... 127	13
14	sub detune	-50 ... +50	14
2 - OSCILATOR 2			
15	wave	Sine, Sawtooth, Ramp, Pulse, Triangle, White Noite, Pink Noise	15
16	pulse width	1% - 99%	16
17	pwm	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	17
18	pwm level	0 ... 127	18
19	pitch	-36 ... +36	19
20	fine	-50 ... +50	20
21	pitch mod	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	21
22	pmod level	0 ... 127	22
23	wave level	0 ... 127	23
24	sub level	0 ... 127	24
25	sub detune	-50 ... +50	25
3 - FILTER 1			
26	ON/OFF	ON/OFF	26
27	TYPE	LOW-PASS, HIGH-PASS, BAND-PASS, BAND-REJECT	27
28	CUTOFF	0 ... 127	28
29	RES	0 ... 127	29
30	CUTOFF MOD	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2,	30

		MOD, VEL	
31	CMOD LEVEL	0 ... 127	31
32	RES MOD	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	32
33	RMOD LEVEL	0 ... 127	33
4 - FILTER 2			
34	ON/OFF	ON/OFF	34
35	TYPE	LOW-PASS, HIGH-PASS, BAND-PASS, BAND-REJECT	35
36	CUTOFF	0 ... 127	36
37	RES	0 ... 127	35
38	CUTOFF MOD	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	37
39	CMOD LEVEL	0 ... 127	39
40	RES MOD	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	40
41	RMOD LEVEL	0 ... 127	41
5 - AMP 1			
42	LEVEL	0 ... 127	42
43	AMP MOD	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	43
44	AMOD LEVEL	0 ... 127	44
45	PAN	-63 ... +63	45
46	PAN MOD	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	46
47	PMOD LEVEL	0 ... 127	47
6 - AMPENV1			
48	ATTACK	0 ... 127	48
49	DECAY	0 ... 127	49
50	SUSTAIN	0 ... 127	50
51	RELEASE	0 ... 127	51
7 - AMP 2			
52	LEVEL	0 ... 127	52
53	AMP MOD	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	53
54	AMOD LEVEL	0 ... 127	54
55	PAN	-63 ... +63	55
56	PAN MOD	ENV1, ENV2, ENV3, ENV4, LFO1, LFO2, MOD, VEL	56
57	PMOD LEVEL	0 ... 127	57
8 - AMPENV2			
58	ATTACK	0 ... 127	58
59	DECAY	0 ... 127	59
60	SUSTAIN	0 ... 127	60
61	RELEASE	0 ... 127	61
9 - ENV1			
62	ATTACK	0 ... 127	62
63	DECAY	0 ... 127	63
64	SUSTAIN	0 ... 127	64

65	RELEASE	0 ... 127	65
10 - ENV2			
66	ATTACK	0 ... 127	66
67	DECAY	0 ... 127	67
68	SUSTAIN	0 ... 127	68
69	RELEASE	0 ... 127	69
11 - ENV3			
70	ATTACK	0 ... 127	70
71	DECAY	0 ... 127	71
72	SUSTAIN	0 ... 127	72
73	RELEASE	0 ... 127	73
12 - ENV4			
74	ATTACK	0 ... 127	74
75	DECAY	0 ... 127	75
76	SUSTAIN	0 ... 127	76
77	RELEASE	0 ... 127	77
13 - LFO1			
78	WAVE	Sine, Sawtooth, Ramp, Pulse, Triangle, White Noite, Pink Noise	78
79	PULSE WIDTH	1% - 99%	79
80	FREQUENCY	0 ... 127	80
14 - LFO2			
81	WAVE	Sine, Sawtooth, Ramp, Pulse, Triangle, White Noite, Pink Noise	81
82	PULSE WIDTH	1% - 99%	82
83	FREQUENCY	0 ... 127	83
15 - DELAY			
84	TIME L	0 ... 127	84
85	TIME R	0 ... 127	85
86	FEEDBACK	0 ... 127	86
87	MIX	0% ... 100%	87
16 - FLANGER			
88	Rate	0 ... 127	88
89	Feedback	0 ... 127	89
90	Mix	0% ... 100%	90

6.2. ANEXO 02 – CÓDIGO FONTE DO MICROCONTROLADOR PIC18F452

```
//-------------------------------------------------------
// Universidade de Brasilia - Faculdade de Tecnologia
// Projeto final de graduacao em engenharia eletrica
// 1/2009
//
// mcu.c - Codigo do sintetizador sonoro
//
// Autores:
// Renato Gomes
// Walkiria Eyre
// Marcus Vinicius
//
// Professor Orientador: Adson
//
//-------------------------------------------------------


#include <18F452.H>
#include <string.h>
#include <stdlib.h>

#fuses H4
#use delay (clock=40000000)   // Cristal de 10Mhz com PLL
#use rs232 (BAUD = 31250, XMIT=pin_C6, RCV=pin_c7, BITS=8)  //Config
serial


// Declaracao de variaveis

int menu_pos=0;
int lcd_menu_atual;
int lcd_parametro_atual;
int8 k, i, byte_recebido;
int1 controle_teclas[128];



void transmite1(a){              // Transmite um unico byte
   putc(a);
}


void transmite3(a, b, c){        // Transmite 3 bytes
   putc(a);
   putc(b);
   putc(c);
}


void transmite_nrpn(a, b){          // Transmite mensagem NRPN
transmite3(0xB0, 0x63, 0x00);
transmite3(0xB0, 0x62, a);
transmite3(0xB0, 0x06, b);
}
```

```
#int_rda                              // Codigo da interrupcao de
receber
void recebeu(){                       // byte pela porta serial
   byte_recebido = getc();
   transmite1(byte_recebido);
}


int patch_atual[91] = {                  // Configuracao inicial dos
parametros
   0, 0, 1, 0, 1, 50, 0, 0, 64, 60,
   0, 0, 127, 127, 56, 1, 50, 0, 0, 64,
   68, 0, 0, 127, 127, 72, 0, 0, 127, 0,
   0, 0, 0, 0, 0, 0, 127, 0, 0, 0,
   0, 0, 127, 0, 0, 64, 0, 0, 0, 0,
   127, 0, 127, 0, 0, 64, 0, 0, 0, 0,
   127, 0, 0, 0, 127, 0, 0, 0, 127, 0,
   0, 0, 127, 0, 0, 0, 127, 0, 0, 50,
   20, 0, 50, 15, 25, 35, 65, 30, 20, 105,
   60};


// Declaracao de variaveis de texto:

const char numbers_0_100[101][4] = {
   "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
   "10", "11", "12", "13", "14", "15", "16", "17", "18", "19",
   "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
   "30", "31", "32", "33", "34", "35", "36", "37", "38", "39",
   "40", "41", "42", "43", "44", "45", "46", "47", "48", "49",
   "50", "51", "52", "53", "54", "55", "56", "57", "58", "59",
   "60", "61", "62", "63", "64", "65", "66", "67", "68", "69",
   "70", "71", "72", "73", "74", "75", "76", "77", "78", "79",
   "80", "81", "82", "83", "84", "85", "86", "87", "88", "89",
   "90", "91", "92", "93", "94", "95", "96", "97", "98", "99", "100"};

const char numbers_63[128][4] = {
   "-64", "-63", "-62", "-61", "-60",
   "-59", "-58", "-57", "-56", "-55", "-54", "-53", "-52", "-51", "-
50",
   "-49", "-48", "-47", "-46", "-45", "-44", "-43", "-42", "-41", "-
40",
   "-39", "-38", "-37", "-36", "-35", "-34", "-33", "-32", "-31", "-
30",
   "-29", "-28", "-27", "-26", "-25", "-24", "-23", "-22", "-21", "-
20",
   "-19", "-18", "-17", "-16", "-15", "-14", "-13", "-12", "-11", "-
10",
   "-9", "-8", "-7", "-6", "-5", "-4", "-3", "-2", "-1", "0",
   "+1", "+2", "+3", "+4", "+5", "+6", "+7", "+8", "+9", "+10",
   "+11", "+12", "+13", "+14", "+15", "+16", "+17", "+18", "+19",
"+20",
   "+21", "+22", "+23", "+24", "+25", "+26", "+27", "+28", "+29",
"+30",
   "+31", "+32", "+33", "+34", "+35", "+36", "+37", "+38", "+39",
"+40",
   "+41", "+42", "+43", "+44", "+45", "+46", "+47", "+48", "+49",
"+50",
   "+51", "+52", "+53", "+54", "+55", "+56", "+57", "+58", "+59",
"+60",
```

```
   "+61", "+62", "+63"};


const char numbers_0_127[128][4] = {
   "000", "001", "002", "003",
   "004", "005", "006", "007",
   "008", "009", "010", "011",
   "012", "013", "014", "015",
   "016", "017", "018", "019",
   "020", "021", "022", "023",
   "024", "025", "026", "027",
   "028", "029", "030", "031",
   "032", "033", "034", "035",
   "036", "037", "038", "039",
   "040", "041", "042", "043",
   "044", "045", "046", "047",
   "048", "049", "050", "051",
   "052", "053", "054", "055",
   "056", "057", "058", "059",
   "060", "061", "062", "063",
   "064", "065", "066", "067",
   "068", "069", "070", "071",
   "072", "073", "074", "075",
   "076", "077", "078", "079",
   "080", "081", "082", "083",
   "084", "085", "086", "087",
   "088", "089", "090", "091",
   "092", "093", "094", "095",
   "096", "097", "098", "099",
   "100", "101", "102", "103",
   "104", "105", "106", "107",
   "108", "109", "110", "111",
   "112", "113", "114", "115",
   "116", "117", "118", "119",
   "120", "121", "122", "123",
   "124", "125", "126", "127"};

const char espacos[14] = "             ";

const char lcd_menu[17][17] = {
   "|CONFIG|        ",
   "|OSCILATOR 1|   ",
   "|OSCILATOR 2|   ",
   "|FILTER 1|      ",
   "|FILTER 2|      ",
   "|AMPLIFIER 1|   ",
   "|AMP ENVELOPE 1|",
   "|AMPLIFIER 2|   ",
   "|AMP ENVELOPE 2|",
   "|ENVELOPE 1|    ",
   "|ENVELOPE 2|    ",
   "|ENVELOPE 3|    ",
   "|ENVELOPE 4|    ",
   "|LFO 1|         ",
   "|LFO 2|         ",
   "|DELAY|         ",
   "|FLANGER|       "};

const char lcd_parametro[43][17] = {
   "Portamento: ",      // 0   (CONFIG)
   "Mono Mode: ",       // 1
```

```
   "Retrigger: ",       // 2
   "OSC Sync: ",        // 3
   "Wave: ",            // 4   (osc 1 e 2)
   "Pulse Width: ",     // 5
   "P.Width Mod:",      // 6
   "P.W.M.Level: ",     // 7
   "Pitch: ",           // 8
   "Fine: ",            // 9
   "Pitch Mod: ",       // 10
   "P_Mod Level: ",     // 11
   "Wave Level: ",      // 12
   "Sub Level: ",       // 13
   "Sub Detune: ",      // 14
   "ON/OFF: ",          // 15   (Filter 1 e 2)
   "Type: ",            // 16
   "Cutoff: ",          // 17
   "Resonance: ",       // 18
   "Cutoff Mod: ",      // 19
   "C_Mod Level: ",     // 20
   "Res Mod: ",         // 21
   "R_Mod Level: ",     // 22
   "Level: ",           // 23   (Amp 1 e 2)
   "Amp Mod: ",         // 24
   "A_Mod Level: ",     // 25
   "Pan: ",             // 26
   "Pan Mod: ",         // 27
   "P_Mod Level: ",     // 28
   "Attack: ",          // 29   (Envelopes)
   "Decay: ",           // 30
   "Sustain: ",         // 31
   "Release: ",         // 32
   "Wave: ",            // 33   (LFO 1 e 2)
   "Pulse Width: ",     // 34
   "Frequency: ",       // 35
   "Time L: ",          // 36   (Delay)
   "Time R: ",          // 37
   "Feedback: ",        // 38
   "Mix: ",             // 39
   "Rate: ",            // 40   (Flanger)
   "Feedback: ",        // 41
   "Mix: "              // 42
   };


void init_lcd(){                    // Rotina de inicializacao do LCD

   delay_ms(500);

   output_a(0x03);         // 1
   delay_us(160);
   output_high(PIN_A5);
   delay_us(200);
   output_low(PIN_A5);

   delay_ms(5);

   output_a(0x03);         // 2
   delay_us(160);
   output_high(PIN_A5);
   delay_us(200);
   output_low(PIN_A5);
```

```
delay_us(150);

output_a(0x03);         // 3
delay_us(160);
output_high(PIN_A5);
delay_us(200);
output_low(PIN_A5);
delay_us(200);

output_a(0x02);         // 4
delay_us(160);
output_high(PIN_A5);
delay_us(200);
output_low(PIN_A5);
delay_us(200);

output_a(0x02);         // Function set
delay_us(160);
output_high(PIN_A5);
delay_us(200);
output_low(PIN_A5);
delay_us(200);

output_a(0x08);
delay_us(160);
output_high(PIN_A5);
delay_us(200);
output_low(PIN_A5);
delay_us(200);

output_a(0x00);        // Display
delay_us(160);         // Cursor off
output_high(PIN_A5);   // Blinks off
delay_us(200);
output_low(PIN_A5);
delay_us(200);

output_a(0x0C);
delay_us(160);
output_high(PIN_A5);
delay_us(200);
output_low(PIN_A5);
delay_us(200);

output_a(0x00);         // Clear
delay_us(160);
output_high(PIN_A5);
delay_us(200);
output_low(PIN_A5);
delay_us(200);

output_a(0x01);
delay_us(160);
output_high(PIN_A5);
delay_us(200);
output_low(PIN_A5);
delay_us(200);

output_a(0x00);         // Entry mode set
delay_us(160);
```

```
   output_high(PIN_A5);
   delay_us(200);
   output_low(PIN_A5);
   delay_us(200);

   output_a(0x06);
   delay_us(160);
   output_high(PIN_A5);
   delay_us(200);
   output_low(PIN_A5);
   delay_us(200);

}


void lcd_w (char letra){         // Funcao pra escrever no LCD

   char letra_t;
   letra_t = letra;
   swap (letra_t);

   bit_clear (letra_t, 6);
   bit_clear (letra_t, 5);
   bit_set (letra_t, 4);

   bit_clear (letra, 6);
   bit_clear (letra, 5);
   bit_set (letra, 4);

   delay_us(140);
   output_a(letra_t);
   output_high(PIN_A5);
   delay_us(100);
   output_low(PIN_A5);

   output_a(letra);
   output_high(PIN_A5);
   delay_us(100);
   output_low(PIN_A5);

}

void lcd_pos (char pos){      // Funcao para posicionar o cursor

   char pos_t;
   pos_t = pos;
   swap (pos_t);

   bit_clear (pos_t, 6);
   bit_clear (pos_t, 5);
   bit_clear (pos_t, 4);

   bit_clear (pos, 6);
   bit_clear (pos, 5);
   bit_clear (pos, 4);

   delay_us(140);
   output_a(pos_t);
   output_high(PIN_A5);
   delay_us(100);
   output_low(PIN_A5);
```

```
   output_a(pos);
   output_high(PIN_A5);
   delay_us(100);
   output_low(PIN_A5);


}



void patch_init_send(){          // Funcao pra mandar config
armazenada

int8 aux5;                       // Declara variavel auxiliar

   for (aux5=1; aux5<92; aux5++){
      transmite_nrpn(aux5, patch_atual[aux5]);
      delay_ms(2);
   }
}


void edit_menu_refresh(){           // Atualiza info no LCD

   switch (menu_pos) {
      case 0: lcd_menu_atual = 0;         // config - portamento
            lcd_parametro_atual = 0;
            break;
      case 1: lcd_menu_atual = 0;         // config - mono mode
            lcd_parametro_atual = 1;
            break;
      case 2: lcd_menu_atual = 0;         // config - retrigger
            lcd_parametro_atual = 2;
            break;
      case 3: lcd_menu_atual = 0;         // config - OSC Sync
            lcd_parametro_atual = 3;
            break;
      case 4: lcd_menu_atual = 1;         // osc1 - wave
            lcd_parametro_atual = 4;
            break;
      case 5: lcd_menu_atual = 1;         // osc1 - pulse width
            lcd_parametro_atual = 5;
            break;
      case 6: lcd_menu_atual = 1;         // osc1 - p.width mod
            lcd_parametro_atual = 6;
            break;
      case 7: lcd_menu_atual = 1;         // osc1 - pwm level
            lcd_parametro_atual = 7;
            break;
      case 8: lcd_menu_atual = 1;         // osc1 - pitch
            lcd_parametro_atual = 8;
            break;
      case 9: lcd_menu_atual = 1;         // osc1 - fine
            lcd_parametro_atual = 9;
            break;
      case 10: lcd_menu_atual = 1;        // osc1 - pitch mod
            lcd_parametro_atual = 10;
            break;
      case 11: lcd_menu_atual = 1;        // osc1 - pmod level
            lcd_parametro_atual = 11;
```

```
      break;
case 12: lcd_menu_atual = 1;        // osc1 - wave level
      lcd_parametro_atual = 12;
      break;
case 13: lcd_menu_atual = 1;
      lcd_parametro_atual = 13;      // osc1 - sub level
      break;
case 14: lcd_menu_atual = 1;
      lcd_parametro_atual = 14;      // osc1 - sub detune
      break;
case 15: lcd_menu_atual = 2;
      lcd_parametro_atual = 4;      // osc2 - wave
      break;
case 16: lcd_menu_atual = 2;
      lcd_parametro_atual = 5;      // osc2 - Pulse width
      break;
case 17: lcd_menu_atual = 2;
      lcd_parametro_atual = 6;      // osc2 - Pwm
      break;
case 18: lcd_menu_atual = 2;
      lcd_parametro_atual = 7;      // osc2 - pwm level
      break;
case 19: lcd_menu_atual = 2;
      lcd_parametro_atual = 8;      // osc2 - pitch
      break;
case 20: lcd_menu_atual = 2;
      lcd_parametro_atual = 9;      // osc2 - fine
      break;
case 21: lcd_menu_atual = 2;
      lcd_parametro_atual = 10;     // osc2 - pitch mod
      break;
case 22: lcd_menu_atual = 2;
      lcd_parametro_atual = 11;     // osc2 - pmod level
      break;
case 23: lcd_menu_atual = 2;
      lcd_parametro_atual = 12;     // osc2 - wave level
      break;
case 24: lcd_menu_atual = 2;
      lcd_parametro_atual = 13;     // osc2 - sub level
      break;
case 25: lcd_menu_atual = 2;
      lcd_parametro_atual = 14;     // osc2 - sub detune
      break;
case 26: lcd_menu_atual = 3;        // Filter 1 - on/off
      lcd_parametro_atual = 15;
      break;
case 27: lcd_menu_atual = 3;        // Filter 1 - Type
      lcd_parametro_atual = 16;
      break;
case 28: lcd_menu_atual = 3;        // Filter 1 - Cutoff
      lcd_parametro_atual = 17;
      break;
case 29: lcd_menu_atual = 3;        // Filter 1 - Res
      lcd_parametro_atual = 18;
      break;
case 30: lcd_menu_atual = 3;        // Filter 1 - Cutoff MOD
      lcd_parametro_atual = 19;
      break;
case 31: lcd_menu_atual = 3;        // Filter 1 - CMOD Level
      lcd_parametro_atual = 20;
      break;
```

```
case 32: lcd_menu_atual = 3;        // Filter 1 - Res MOD
      lcd_parametro_atual = 21;
      break;
case 33: lcd_menu_atual = 3;        // Filter 1 - RMOD Level
      lcd_parametro_atual = 22;
      break;
case 34: lcd_menu_atual = 4;        // Filter 2 - on/off
      lcd_parametro_atual = 15;
      break;
case 35: lcd_menu_atual = 4;        // Filter 2 - type
      lcd_parametro_atual = 16;
      break;
case 36: lcd_menu_atual = 4;        // Filter 2 - cutoff
      lcd_parametro_atual = 17;
      break;
case 37: lcd_menu_atual = 4;        // Filter 2 - res
      lcd_parametro_atual = 18;
      break;
case 38: lcd_menu_atual = 4;        // Filter 2 - cutoff mod
      lcd_parametro_atual = 19;
      break;
case 39: lcd_menu_atual = 4;        // Filter 2 - cmod level
      lcd_parametro_atual = 20;
      //offset_valor = 11;
      break;
case 40: lcd_menu_atual = 4;        // Filter 2 - res mod
      lcd_parametro_atual = 21;
      //offset_valor = 12;
      break;
case 41: lcd_menu_atual = 4;        // Filter 2 - rmod level
      lcd_parametro_atual = 22;
      //offset_valor = 13;
      break;
case 42: lcd_menu_atual = 5;        // amp 1 - level
      lcd_parametro_atual = 23;
      //offset_valor = 11;
      break;
case 43: lcd_menu_atual = 5;        // amp 1 - amp mod
      lcd_parametro_atual = 24;
      //offset_valor = 9;
      break;
case 44: lcd_menu_atual = 5;        // amp 1 - a_mod level
      lcd_parametro_atual = 25;
      //offset_valor = 13;
      break;
case 45: lcd_menu_atual = 5;        // amp 1 - pan
      lcd_parametro_atual = 26;
      //offset_valor = 11;
      break;
case 46: lcd_menu_atual = 5;        // amp 1 - pan mod
      lcd_parametro_atual = 27;
      break;
case 47: lcd_menu_atual = 5;        // amp 1 - p_mod level
      lcd_parametro_atual = 28;
      break;
case 48: lcd_menu_atual = 6;        // amp env 1 - attack
      lcd_parametro_atual = 29;
      break;
case 49: lcd_menu_atual = 6;        // amp env 1 - decay
      lcd_parametro_atual = 30;
      break;
```

```
case 50: lcd_menu_atual = 6;        // amp env 1 - sustain
      lcd_parametro_atual = 31;
      break;
case 51: lcd_menu_atual = 6;        // amp env 1 - release
      lcd_parametro_atual = 32;
      break;
case 52: lcd_menu_atual = 7;        // amp 2 - level
      lcd_parametro_atual = 23;
      break;
case 53: lcd_menu_atual = 7;        // amp 2 - amp mod
      lcd_parametro_atual = 24;
      break;
case 54: lcd_menu_atual = 7;        // amp 2 - a_mod level
      lcd_parametro_atual = 25;
      break;
case 55: lcd_menu_atual = 7;        // amp 2 - pan
      lcd_parametro_atual = 26;
      break;
case 56: lcd_menu_atual = 7;        // amp 2 - pan mod
      lcd_parametro_atual = 27;
      break;
case 57: lcd_menu_atual = 7;        // amp 2 - pmod level
      lcd_parametro_atual = 28;
      break;
case 58: lcd_menu_atual = 8;        // amp env 2 - attack
      lcd_parametro_atual = 29;
      break;
case 59: lcd_menu_atual = 8;        // amp env 2 - decay
      lcd_parametro_atual = 30;
      break;
case 60: lcd_menu_atual = 8;        // amp env 2 - sustain
      lcd_parametro_atual = 31;
      break;
case 61: lcd_menu_atual = 8;        // amp env 2 - release
      lcd_parametro_atual = 32;
      break;
case 62: lcd_menu_atual = 9;        // env 1 - attack
      lcd_parametro_atual = 29;
      break;
case 63: lcd_menu_atual = 9;        // env 1 - decay
      lcd_parametro_atual = 30;
      break;
case 64: lcd_menu_atual = 9;        // env 1 - sustain
      lcd_parametro_atual = 31;
      break;
case 65: lcd_menu_atual = 9;        // env 1 - release
      lcd_parametro_atual = 32;
      break;
case 66: lcd_menu_atual = 10;       // env 2 - attack
      lcd_parametro_atual = 29;
      break;
case 67: lcd_menu_atual = 10;       // env 2 - decay
      lcd_parametro_atual = 30;
      break;
case 68: lcd_menu_atual = 10;       // env 2 - sustain
      lcd_parametro_atual = 31;
      break;
case 69: lcd_menu_atual = 10;       // env 2 - release
      lcd_parametro_atual = 32;
      break;
case 70: lcd_menu_atual = 11;       // env 3 - attack
```

```
      lcd_parametro_atual = 29;
      break;
case 71: lcd_menu_atual = 11;       // env 3 - decay
      lcd_parametro_atual = 30;
      break;
case 72: lcd_menu_atual = 11;       // env 3 - sustain
      lcd_parametro_atual = 31;
      break;
case 73: lcd_menu_atual = 11;       // env 3 - release
      lcd_parametro_atual = 32;
      break;
case 74: lcd_menu_atual = 12;       // env 4 - attack
      lcd_parametro_atual = 29;
      break;
case 75: lcd_menu_atual = 12;       // env 4 - decay
      lcd_parametro_atual = 30;
      break;
case 76: lcd_menu_atual = 12;       // env 4 - sustain
      lcd_parametro_atual = 31;
      break;
case 77: lcd_menu_atual = 12;       // env 4 - release
      lcd_parametro_atual = 32;
      break;
case 78: lcd_menu_atual = 13;       // lfo 1 - wave
      lcd_parametro_atual = 33;
      break;
case 79: lcd_menu_atual = 13;       // lfo 1 - pulse width
      lcd_parametro_atual = 34;
      break;
case 80: lcd_menu_atual = 13;       // lfo 1 - frequency
      lcd_parametro_atual = 35;
      break;
case 81: lcd_menu_atual = 14;       // lfo 2 - wave
      lcd_parametro_atual = 33;
      break;
case 82: lcd_menu_atual = 14;       // lfo 2 - pulse width
      lcd_parametro_atual = 34;
      break;
case 83: lcd_menu_atual = 14;       // lfo 2 - frequency
      lcd_parametro_atual = 35;
      break;
case 84: lcd_menu_atual = 15;       // delay - time l
      lcd_parametro_atual = 36;
      break;
case 85: lcd_menu_atual = 15;       // delay - time r
      lcd_parametro_atual = 37;
      break;
case 86: lcd_menu_atual = 15;       // delay - feedback
      lcd_parametro_atual = 38;
      break;
case 87: lcd_menu_atual = 15;       // delay - mix
      lcd_parametro_atual = 39;
      break;
case 88: lcd_menu_atual = 16;       // flanger - rate
      lcd_parametro_atual = 40;
      break;
case 89: lcd_menu_atual = 16;       // flanger - feedback
      lcd_parametro_atual = 41;
      break;
case 90: lcd_menu_atual = 16;       // flanger - mix
      lcd_parametro_atual = 42;
```

```
         break;

}

lcd_pos (0x80);
lcd_w(lcd_menu[lcd_menu_atual]);

lcd_pos (0xC0);
lcd_w(lcd_parametro[lcd_parametro_atual]);



switch (menu_pos){
   case 4:                                    // TYPE 1
   case 15:
   case 78:
   case 81:
      switch (patch_atual[menu_pos]){
         case 0:
            lcd_w("Sine      ");
            break;
         case 1:
            lcd_w("Sawtooth  ");
            break;
         case 2:
            lcd_w("Ramp      ");
            break;
         case 3:
            lcd_w("Triangle  ");
            break;
         case 4:
            lcd_w("Pulse     ");
            break;
         case 5:
            lcd_w("WhiteNoise");
            break;
         case 6:
            lcd_w("Pink Noise");
            break;
      }
      break;
   case 5:                                    // TYPE 2 e 15
   case 16:
   case 79:
   case 82:
   case 87:
   case 90:
      lcd_w(numbers_0_100[patch_atual[menu_pos]]);
      lcd_w("%");
      lcd_w(espacos);
      break;
   case 6:                                    // TYPE 3
   case 10:
   case 17:
   case 21:
   case 30:
   case 32:
   case 38:
   case 40:
   case 43:
   case 46:
```

```
case 53:
case 56:
   switch (patch_atual[menu_pos]){
      case 0:
         lcd_w("ENV1      ");
         break;
      case 1:
         lcd_w("ENV2      ");
         break;
      case 2:
         lcd_w("ENV3      ");
         break;
      case 3:
         lcd_w("ENV4      ");
         break;
      case 4:
         lcd_w("LFO1      ");
         break;
      case 5:
         lcd_w("LFO2      ");
         break;
      case 6:
         lcd_w("MOD        ");
         break;
      case 7:
         lcd_w("VEL       ");
         break;
   }
   break;
case 8:                                    // TYPE 5
case 19:
   lcd_w(numbers_63[patch_atual[menu_pos]]);
   lcd_w(espacos);
   break;
case 9:                                    // TYPE 6
case 20:
case 14:
case 25:
   lcd_w(numbers_63[patch_atual[menu_pos]]);
   lcd_w(espacos);
   break;
case 1:                                   // TYPE 8
case 2:
case 3:
case 26:
case 34:
   switch (patch_atual[menu_pos]){
      case 0:
         lcd_w("OFF     ");
         break;
      case 1:
         lcd_w("ON      ");
         break;
   }
   break;
case 27:                                   // TYPE 9
case 35:
   switch (patch_atual[menu_pos]){
      case 0:
         lcd_w("LOW-PASS  ");
         break;
```

```
            case 1:
               lcd_w("HIGH-PASS ");
               break;
            case 2:
               lcd_w("BAND-PASS ");
               break;
            case 3:
               lcd_w("BAND-REJEC");
         }
         break;
      case 45:                                   // TYPE 12
      case 55:
         lcd_w(numbers_63[patch_atual[menu_pos]]);
         lcd_w(espacos);
         break;
      default:                                   // TYPE 4
         lcd_w(numbers_0_127[patch_atual[menu_pos]]);
         lcd_w(espacos);
         break;
   }

}



void testa_botoes(){                          // Checa se apertou botao
   output_d(0);            // *** Menu -
   if (!input(PIN_C5)) {
      while (!input(PIN_C5)){}
      switch (menu_pos){
      case 0:
      case 1:
      case 2:
      case 3: break;
      case 4:
      case 5:
      case 6:
      case 7:
      case 8:
      case 9:
      case 10:
      case 11:
      case 12:
      case 13:
      case 14: menu_pos=0; break;
      case 15:
      case 16:
      case 17:
      case 18:
      case 19:
      case 20:
      case 21:
      case 22:
      case 23:
      case 24:
      case 25: menu_pos=4; break;
      case 26:
      case 27:
      case 28:
      case 29:
```

```
case 30:
case 31:
case 32:
case 33: menu_pos=15; break;
case 34:
case 35:
case 36:
case 37:
case 38:
case 39:
case 40:
case 41: menu_pos=26; break;
case 42:
case 43:
case 44:
case 45:
case 46:
case 47: menu_pos=34; break;
case 48:
case 49:
case 50:
case 51: menu_pos=42; break;
case 52:
case 53:
case 54:
case 55:
case 56:
case 57: menu_pos=48; break;
case 58:
case 59:
case 60:
case 61: menu_pos=52; break;
case 62:
case 63:
case 64:
case 65: menu_pos=58; break;
case 66:
case 67:
case 68:
case 69: menu_pos=62; break;
case 70:
case 71:
case 72:
case 73: menu_pos=66; break;
case 74:
case 75:
case 76:
case 77: menu_pos=70; break;
case 78:
case 79:
case 80: menu_pos=74; break;
case 81:
case 82:
case 83: menu_pos=78; break;
case 84:
case 85:
case 86:
case 87: menu_pos=81; break;
case 88:
case 89:
case 90: menu_pos=84; break;
```

```
   }
}

output_d(1);            // *** Menu +
if (!input(PIN_C5)) {
   while (!input(PIN_C5)){}
   switch (menu_pos){
   case 0:
   case 1:
   case 2:
   case 3: menu_pos=4; break;
   case 4:
   case 5:
   case 6:
   case 7:
   case 8:
   case 9:
   case 10:
   case 11:
   case 12:
   case 13:
   case 14: menu_pos=15; break;
   case 15:
   case 16:
   case 17:
   case 18:
   case 19:
   case 20:
   case 21:
   case 22:
   case 23:
   case 24:
   case 25: menu_pos=26; break;
   case 26:
   case 27:
   case 28:
   case 29:
   case 30:
   case 31:
   case 32:
   case 33: menu_pos=34; break;
   case 34:
   case 35:
   case 36:
   case 37:
   case 38:
   case 39:
   case 40:
   case 41: menu_pos=42; break;
   case 42:
   case 43:
   case 44:
   case 45:
   case 46:
   case 47: menu_pos=48; break;
   case 48:
   case 49:
   case 50:
   case 51: menu_pos=52; break;
   case 52:
   case 53:
```

```
   case 54:
   case 55:
   case 56:
   case 57: menu_pos=58; break;
   case 58:
   case 59:
   case 60:
   case 61: menu_pos=62; break;
   case 62:
   case 63:
   case 64:
   case 65: menu_pos=66; break;
   case 66:
   case 67:
   case 68:
   case 69: menu_pos=70; break;
   case 70:
   case 71:
   case 72:
   case 73: menu_pos=74; break;
   case 74:
   case 75:
   case 76:
   case 77: menu_pos=78; break;
   case 78:
   case 79:
   case 80: menu_pos=81; break;
   case 81:
   case 82:
   case 83: menu_pos=84; break;
   case 84:
   case 85:
   case 86:
   case 87: menu_pos=88; break;
   case 88:
   case 89:
   case 90: break;
   }
}

output_d(2);            // *** Param -
if (!input(PIN_C5)) {
   while (!input(PIN_C5)){}
   if (menu_pos!=0 &&
      menu_pos!=4 &&
      menu_pos!=15 &&
      menu_pos!=26 &&
      menu_pos!=34 &&
      menu_pos!=42 &&
      menu_pos!=48 &&
      menu_pos!=52 &&
      menu_pos!=58 &&
      menu_pos!=62 &&
      menu_pos!=66 &&
      menu_pos!=70 &&
      menu_pos!=74 &&
      menu_pos!=78 &&
      menu_pos!=81 &&
      menu_pos!=84 &&
      menu_pos!=88){
      menu_pos--;}
```

```
}

output_d(3);            // *** Param +
if (!input(PIN_C5)) {
   while (!input(PIN_C5)){}
   if (menu_pos!=3 &&
      menu_pos!=14 &&
      menu_pos!=25 &&
      menu_pos!=33 &&
      menu_pos!=41 &&
      menu_pos!=47 &&
      menu_pos!=51 &&
      menu_pos!=57 &&
      menu_pos!=61 &&
      menu_pos!=65 &&
      menu_pos!=69 &&
      menu_pos!=73 &&
      menu_pos!=77 &&
      menu_pos!=80 &&
      menu_pos!=83 &&
      menu_pos!=87 &&
      menu_pos!=90){
      menu_pos++;}
}

output_d(4);            // *** Value -
if (!input(PIN_C5)) {
   while (!input(PIN_C5)){}
   switch (menu_pos){
      case 5:
      case 16:
      case 79:
      case 82:
         if (patch_atual[menu_pos]>1) {   // TYPE 2
            patch_atual[menu_pos]--;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
      case 8:
      case 19:
         if (patch_atual[menu_pos]>28) {  // TYPE 5
            patch_atual[menu_pos]--;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
      case 9:
      case 20:
         if (patch_atual[menu_pos]>14) {  // TYPE 6
            patch_atual[menu_pos]--;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
      default:
         if (patch_atual[menu_pos]!=0) {  // RESTO
            patch_atual[menu_pos]--;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
   }
}
```

```
output_d(5);            // *** Value +
if (!input(PIN_C5)) {
   while (!input(PIN_C5)){}
   switch (menu_pos){
      case 4:
      case 15:
      case 78:
      case 81:
         if (patch_atual[menu_pos]<6) {       // TYPE 1
            patch_atual[menu_pos]++;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
      case 5:
      case 16:
      case 79:
      case 82:
         if (patch_atual[menu_pos]<99) {     // TYPE 2
            patch_atual[menu_pos]++;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
      case 6:
      case 10:
      case 17:
      case 21:
      case 30:
      case 32:
      case 38:
      case 40:
      case 43:
      case 46:
      case 53:
      case 56:
         if (patch_atual[menu_pos]<7) {      // TYPE 3
            patch_atual[menu_pos]++;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
      case 8:
      case 19:
         if (patch_atual[menu_pos]<100) {    // TYPE 5
            patch_atual[menu_pos]++;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
      case 9:
      case 20:
      case 14:
      case 25:
         if (patch_atual[menu_pos]<114) {    // TYPE 6
            patch_atual[menu_pos]++;
            transmite_nrpn(menu_pos, patch_atual[menu_pos]);
         }
         break;
      case 1:
      case 2:
      case 3:
      case 26:
      case 34:
         if (patch_atual[menu_pos]<1) {      // TYPE 8
```

```
               patch_atual[menu_pos]++;
               transmite_nrpn(menu_pos, patch_atual[menu_pos]);
            }
            break;
         case 27:
         case 35:
            if (patch_atual[menu_pos]<3) {      // TYPE 9
               patch_atual[menu_pos]++;
               transmite_nrpn(menu_pos, patch_atual[menu_pos]);
            }
            break;
         case 87:
         case 90:
            if (patch_atual[menu_pos]<100) {    // TYPE 15
               patch_atual[menu_pos]++;
               transmite_nrpn(menu_pos, patch_atual[menu_pos]);
            }
            break;
         default:
            if (patch_atual[menu_pos]<127) {    // RESTO
               patch_atual[menu_pos]++;
               transmite_nrpn(menu_pos, patch_atual[menu_pos]);
            }
            break;
      }

   }
}


// Funcao principal

void main(){

   init_lcd();    // Inicializa LCD

   enable_interrupts(int_rda);   // Liga interrupcoes
   enable_interrupts(global);

   edit_menu_refresh();          // Mostra informacao no LCD


   // Rotina de inicialização:

   lcd_pos (0x80);
   lcd_w(" INICIALIZANDO  ");
   lcd_pos (0xC0);
   lcd_w("----------------");
   delay_ms(300);
   lcd_pos (0xC0);
   lcd_w("*---------------");
   delay_ms(300);
   lcd_pos (0xC0);
   lcd_w("**--------------");
   delay_ms(300);
   lcd_pos (0xC0);
   lcd_w("***-------------");
   delay_ms(300);
   lcd_pos (0xC0);
   lcd_w("****------------");
   delay_ms(300);
```

```
lcd_pos (0xC0);
lcd_w("*****----------");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("******---------");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("*******--------");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("********-------");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("*********------");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("**********-----");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("***********----");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("************---");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("*************--");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("**************-");
delay_ms(300);
lcd_pos (0xC0);
lcd_w("***************");
delay_ms(300);

patch_init_send();         // Envia parametros armazenados


while (true){              // Loop infinito


for(k=0; k<8; k++){        // FOR para ler teclas
   swap(k);
   output_d(k);
   swap(k);

   for(i=0; i<8; i++)

      {

      if ( !input(PIN_B0+i) && !controle_teclas[i+(k*8)]){
         controle_teclas[i+(k*8)] = 1;
         transmite3(0x90, 0x24+(K*8)+i, 0x40);
         }

      if ( input(PIN_B0+i) && controle_teclas[i+(k*8)]){
         controle_teclas[i+(k*8)] = 0;
         transmite3(0x80, 0x24+(K*8)+i, 0x40);
         }
```

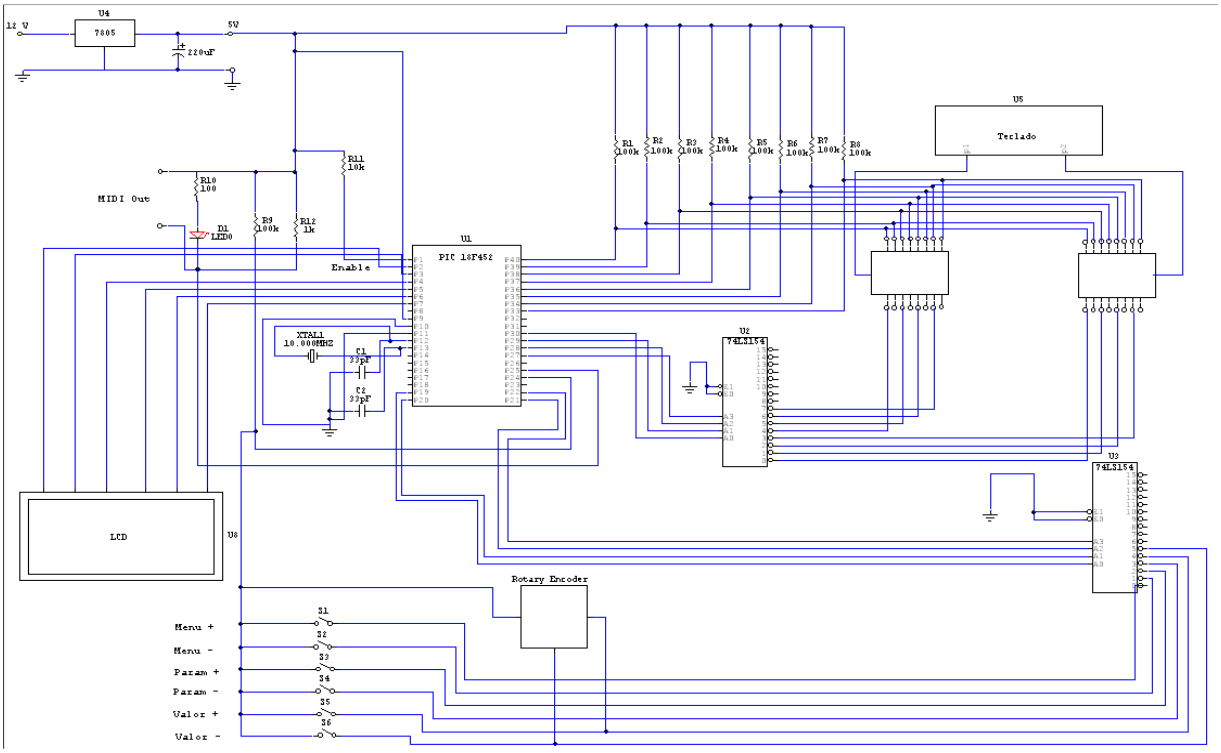
```
            }
        }

            testa_botoes();             // Verifica se algum botao foi
pressionado
            edit_menu_refresh();        // Atualiza informacoes no LCD e
                                        // armazena/envia novos valores de
                                        // parametros

       }

}
```

6.3. ANEXO 03 – DIAGRAMA ESQUEMÁTICO DO MÓDULO DE CONTROLE



6.4. ANEXO 04 – DIAGRAMA ESQUEMÁTICO DO PROJETO DO SOFTWARE SYNTHEDIT