UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUCAS FOLLE

Implementação de Síntese FM na Plataforma Arduino Due

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Marcelo de Oliveira Johann

Porto Alegre
2015

AGRADECIMENTOS

Aos meus pais por me sustentarem durante meu tempo na universidade. A todos os professores do Instituto de Informática da UFRGS pelo conhecimento repassado. Ao meu orientador Marcelo de Oliveira Johann por ter me selecionado para trabalhar em uma bolsa de estudos, me aceitar como orientando em uma área de interesse pessoal e pelas diversas horas de trabalho em conjunto. E aos professores Marcelo Pimenta Soares e Renato Perez Ribas por aceitarem fazer parte da banca de avaliação deste trabalho.

RESUMO

Este trabalho apresenta a implementação de um sintetizador FM na plataforma Arduino Due, tendo como inspiração o sintetizador Yamaha DX7. Com o uso dessa plataforma, componentes eletrônicos, placas de prototipação, um teclado MIDI e um DAC externo, foi possível criar um instrumento musical com grande capacidade de expressão artística. A geração de timbres, assim como no DX7, é feita através de 32 algoritmos predefinidos com 6 operadores dispostos em arranjos seriais e paralelos, com cada um possuindo seu próprio gerador de envelope. Os parâmetros de cada operador e envelope podem ser mudados em tempo real, com o usuário podendo notar as influências no som sintetizado imediatamente.

Palavras-chave: Arduino Due. Síntese FM. Sintetizador. Yamaha DX7.

Implementation of FM Synthesis in the Arduino Due Board

ABSTRACT

This thesis shows the implementation of a FM synthesizer in the Arduino Due Board, having as a basis the Yamaha DX7 synthesizer. With the use of this board, electrical components, prototype boards, a MIDI keyboard and an external DAC, a musical instrument with a vast artistic expression capability was able to be built. The timbre generation, just like in the DX7, is made with the use of 32 pre-defined algorithms with 6 operators laid out in serial and parallel arrangements, and with each one having its own envelope generator. The parameters of each operator and envelope can be changed in real-time, making the user able to tell the influences in the synthesized sound right away.

Keywords: Arduino Due. FM Synthesis. Synthesizer. Yamaha DX7.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADSR	Attack, Decay, Sustain, Release
ARM	Advance RISC Machine
DAC	Digital to Analog Converter
FM	Frequency Modulation
MIDI	Musical Instrument Digital Interface
RISC	Reduced Instruction Set Computer
UFRGS	Universidade Federal do Rio Grande do Sul

SUMÁRIO

1 INTRODUÇÃO

A síntese FM foi criada por John M. Chowning em Stanford no ano de 1967. Foi a segunda patente que mais rendeu dinheiro à universidade (mais de $20 milhões) e a responsável pelo primeiro sintetizador digital de sucesso no mercado: o Yamaha DX7 (SALISBURY, 1997). A implementação da síntese requer poucas linhas de código e consegue gerar diversos timbres complexos, fato que colaborou para sua popularização e grande adoção no mercado, podendo ser encontrada em placas de som de computadores, videogames e outros sistemas digitais.

"The DX7 was an instant hit all over the world, and both the instrument and its sound soon became driving forces of the pop music of the eighties. [..] many of its technologies and features also greatly influenced how synths would be developed thereafter." (YAMAHA CORPORATION, 2014, cap. 2). O Yamaha DX7 se tornou um marco para os sintetizadores digitais e introduziu uma alternativa ao mercado dominado por sintetizadores analógicos. Com um visual mais arrojado, menor número de controladores, presença de um visor digital, possibilidade de salvar e carregar timbres de um cartucho e um preço competitivo, o DX7 fugiu dos padrões da época e se mostrou algo totalmente inovador.

A empresa Arduino produz microcontroladores com diversas especificações. O modelo mais comum, o Arduino UNO, possui um processador de 8 bits com uma velocidade de clock de 16MHz. Essas especificações não são suficientes para se realizar projetos de áudio sofisticados. Porém, em 2012 foi lançada a plataforma Arduino Due, com um processador ARM de 32 bits e uma velocidade de clock de 84MHz. Esse incremento nas especificações tornam o Due uma plataforma capaz de realizar projetos de áudio mais avançados e apto a rodar um sintetizador FM.

2 SÍNTESE FM

"The remarkable acceptance of the "X" Series musical instruments produced by Yamaha has led to an equally remarkable need for understanding how the theory of frequency modulation (FM) synthesis can be effectively used in the creation of musical sounds" (CHOWNING; BRISTOW, 1986, p. 8). A síntese FM consiste em utilizar um formato de onda básico (mais comumente seno, porém também podem ser utilizados os formatos de onda quadrada, triangular e dente de serra) e pelo menos dois osciladores (também chamados de operadores) em série. O último operador de uma série é chamado de portador e é ouvido diretamente pelo usuário. Já os operadores acima dele são chamados de moduladores e podem ou não ser ouvidos diretamente. As saídas dos operadores moduladores sempre servem de entrada para um outro operador e são utilizadas para modulá-lo, e como consequência, todos os operadores subsequentes da série. Essa modulação é o que permite a criação dos mais variados timbres. A diferente disposição dos operadores disponíveis em série e paralelo forma um algoritmo.

> If a sinusoid is given a frequency which varies slowly in time we hear it as having a varying pitch. But if the pitch changes so quickly that our ears can't track the change if the change itself occurs at or above the fundamental frequency of the sinusoid—we hear a timbral change. (PUCKETTE, 2007).

Para compreender como a modulação de um operador afeta o outro, um exemplo de síntese FM com dois operadores como descrito por Chowning e Bristow (1986) é dado abaixo. Considerando:

c: razão da frequência do operador portador
m: razão da frequência do operador modulador

No caso de um usuário estar pressionando a tecla lá central de um teclado, todos os operadores de um algoritmo irão receber a frequência dessa nota (440Hz). As variáveis c e m multiplicam essa frequência para que os operadores possam modifica-las, tendo assim a possibilidade de terem frequências diferentes uns dos outros. Com base nisso, caso a variável $c = 2.00$, a frequência da portadora será dobrada (880Hz) Agora considerando a seguinte situação:

$f_c = 440\text{Hz}$ (frequência do opeardor portador)
$f_m = 110\text{Hz}$ (frequência do operador modulador)

$$\Delta \mathrm{f} = 220\mathrm{Hz}\ (\text{variação de frequência})$$

Para o acaso descrito acima, tem-se a modulação ocorrendo da seguinte forma: a frequência do operador portador ($440Hz$) está variando de maneira continua numa velocidade de 110Hz (frequência do operador modulador) por uma quantia de 220Hz (variação de frequência) acima e abaixo de sua frequência. Com base nisso, é possível notar que o timbre gerado por essa modulação pode, por exemplo, ser reproduzido uma oitava a baixo, bastando dividir-se as três frequências pela metade.

Outro fator importante é o índice de modulação, uma relação que pode ser descrita da seguinte maneira:

$$I = \frac{\Delta f}{f_m}$$

Conforme exemplificado anteriormente, quando mudamos a frequência do operador portador e modulador por um mesmo fator, a variação de frequência também deve ser mudada pelo mesmo fator de modo a manter constante o número de componentes de frequência no espectro. Mantendo-se o índice de modulação constante, podemos garantir que esse escalonamento ocorra. Sendo assim, o índice se modulação governa o número e amplitudes dos componentes de frequência no espectro. É importante notar para as seções posteriores que de modo a simplificar a programação dos operadores (que podem variar de moduladores e portadores), a Yamaha definiu o nível de um operador modulador como sendo o seu índice de modulação. Sendo assim, o nível de um operador modulador controla a sua intensidade de modulação enquanto o nível de um operador portador controla o seu volume.

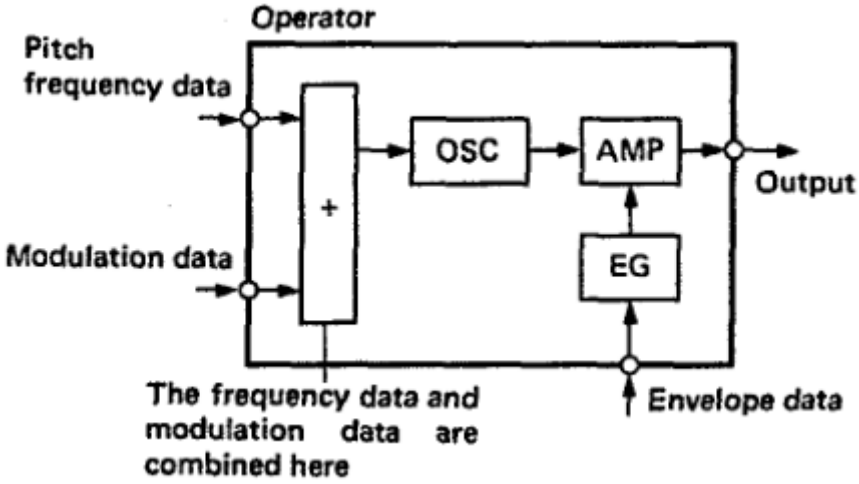
3 YAMAHA DX7

"In 1983 the Yamaha Corporation released […] a synthesizer that would change the course of modern music. This machine, of course, was the model DX7, and, for the first time ever, digital audio synthesis was made available to the musician at a reasonable price." (MASSEY, 1986, p. 11). Após ter obtido a licença da síntese FM da Universidade de Stanford em 1973, a Yamaha começou a trabalhar na prototipação de sintetizadores FM. Foi somente após 8 anos que a empresa conseguiu introduzir produtos que fizessem uso da tecnologia no mercado: o gerador de tons F-70 e o teclado GS-1 (YAMAHA CORPORATION, 2014). Porém, devido aos altos preços e a presença de apenas dois operadores (o que limitava demais a criação de timbres), ambos os produtos não obtiveram sucesso comercial. Foi somente em 1983 com a chegada do DX7 às prateleiras que a Yamaha conseguiria introduzir de vez a síntese FM no mercado. Com um DAC de 15bits, uma taxa de amostragem de 50000Hz, 6 operadores e 32 algoritmos, o DX7 era capaz de criar uma grande gama de timbres, fato que somado ao seu preço competitivo ajudou-o a tornar-se um sucesso comercial.

3.1 Operadores

"At the heart of the FM tone generator lies the operator—a fundamental component used to generate and modify sound" (YAMAHA, 2014). Todos os tons produzidos pelo Yamaha DX7 são obtidos a partir de uma onda senoidal salva em uma tabela na sua memória. Essa onda, osciladores e um gerador de envelope formam um operador. Para cada nota, o operador lê um valor da tabela de seno baseado em seus parâmetros, a frequência da nota (e sua posição na amostra anterior), e suas possíveis outras entradas: valores de outros operadores e realimentação (feedback) do valor de um operador na amostra anterior. Após o valor ser obtido, é aplicado sobre ele o envelope ADSR e o seu nível. Com base no algoritmo escolhido pelo usuário, a saída do operador pode então ser usada para modular um outro operador ou ir direto para a saída de áudio do teclado.

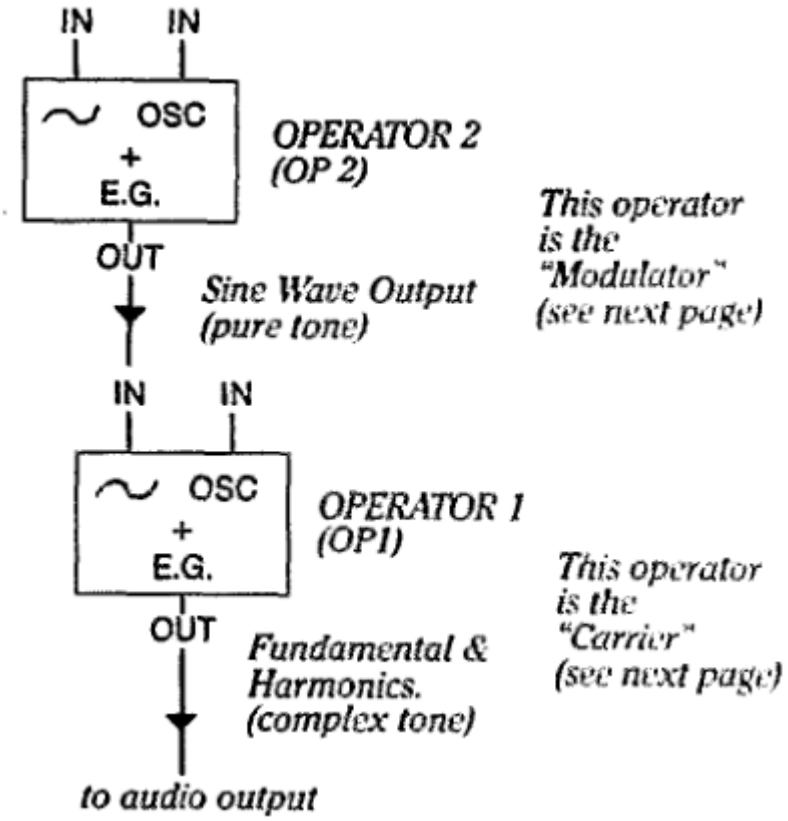
Imagem 3.1 – Visão interna de um operador



Fonte: Yamaha Corporation (1999, p. 23).

Na imagem abaixo é possível observar a disposição de 2 operadores quando um operador modula outro. Como o operador 2 é o primeiro operador da série, sua saída será sempre uma onda senoidal. Porém, a partir do momento que esta onda passa pelo operador 1, seu formato começa a se tornar mais complexo, ocorrendo assim a síntese FM. Caso o nível do operador 2 seja aumentado, o número de harmônicas no timbre irá aumentar, tornando-o mais “brilhante” (YAMAHA CORPORTATION, 1999).

Imagem 3.2 – Operador modulando outro operador

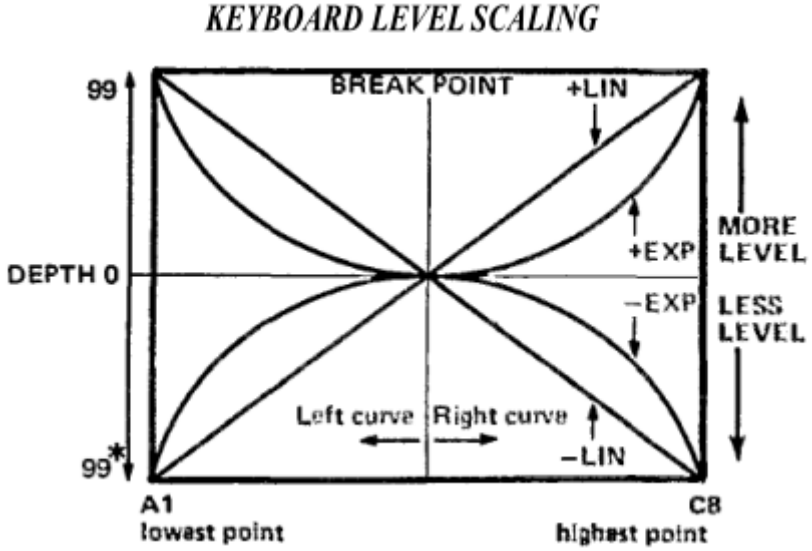


Fonte: Yamaha Corporation (1999, p. 23).

3.1.1 Escalonamento de Nível (Level Scaling)

O escalonamento de nível permite variar o nível que cada operador exerce sobre uma nota com base na distância dela à uma nota chamada de ponto de quebra. O usuário escolhe um ponto de quebra (break point) e uma função (linear positiva, linear negativa, exponencial positiva ou exponencial negativa) a ser aplicada nas notas do lado direito e nas notas do lado esquerdo do ponto de quebra. Além disso, deve-se definir o valor máximo que as funções podem atingir, no intervalo de [0, 99]. Com base nesses parâmetros, cada nota recebe um ponto na função que tem como início o ponto de quebra e varia de acordo com o seu tipo até a última nota do teclado para ambos os lados. Aumentando-se o valor do parâmetro de um lado implica que a função daquele lado possuirá uma maior resolução de valores para representar sua curva, o que acaba por garantir um crescimento mais gradual. Como a função sempre tem início no ponto de quebra escolhido, quando mais distante se estiver dele, maior a diferença de nível obtida. O valor obtido é somado ao nível do operador (não ultrapassando o valor máximo de 99) e utilizado na computação.

Imagem 3.3 – Escalonamento de nível (level scaling)

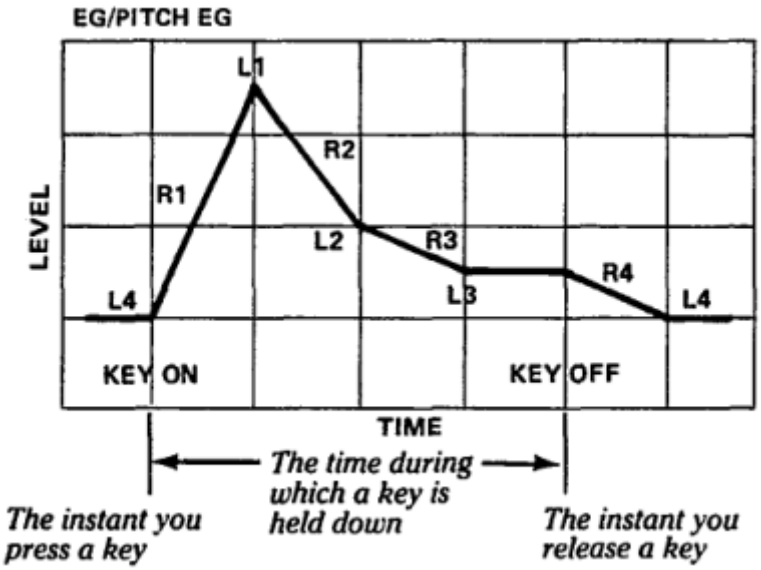


Fonte: Yamaha Corporation (1999, p. 39).

3.2 Gerador de Envelope

Cada operador no DX7 possui um gerador de envelope. Um envelope age mudando a amplitude de uma onda de modo a tentar emular nuances de instrumentos. O envelope gerado no DX7 é muito parecido com um envelope do padrão ADSR, mas possui um número maior de parâmetros, o que dá ao usuário maior liberdade. O usuário deve passar como entrada 4 níveis e 4 velocidades, sendo todas no intervalo de [0, 99]. Ao tocar-se uma tecla, o envelope começa a ir do nível 4 ao nível 1 com velocidade 1 (fase equivalente ao ataque do ADSR). Depois vai do nível 1 ao nível 2 com velocidade 2 (fase equivalente ao decaimento do ADSR). E então vai do nível 2 ao nível 3 com velocidade 3 (fase equivalente à sustentação do ADSR). Enquanto a nota não ser solta, o envelope continuará estável no nível 3. Quando a nota for solta, o envelope vai do nível 3 ao nível 4 com velocidade 4 (fase equivalente ao repouso do ADSR). Após isso, o envelope chega ao fim e a nota não pode mais ser ouvida.

Imagem 3.4 – Envelope do Yamaha DX7

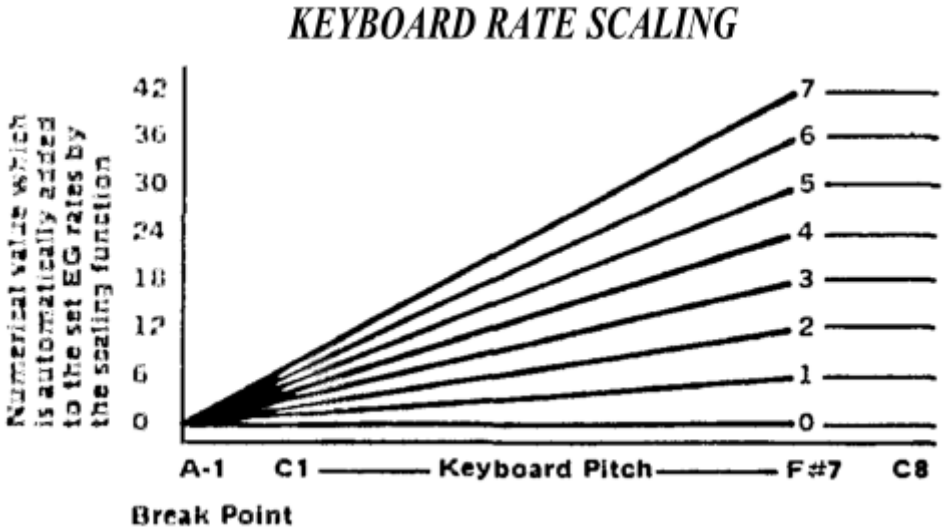


Fonte: Yamaha Corporation (1999, p. 26).

3.2.1 Rate Scaling

Assim como o nível dos operadores pode mudar de acordo com a nota tocada, a velocidade do envelope também pode. Isso permite que o DX7 se aproxime melhor de instrumentos acústicos, onde o comportamento do som varia das notas mais graves para as mais agudas (YAMAHA CORPORATION, 1999). O escalonamento de velocidade possui 8 níveis. O nível inicial é totalmente zerado, não acarretando em mudanças. Com cada nível subsequente, o intervalo dos possíveis valores é aumentado em 6 unidades. As notas então são gradualmente atribuídas um valor da faixa relativa ao nível que será prontamente somado às velocidades dos envelopes. Desse modo as notas mais graves possuem uma variação de velocidade menor e as notas mais agudas possuem uma variação maior.

Imagem 3.5 – Escalonamento de velocidade (rate scaling)

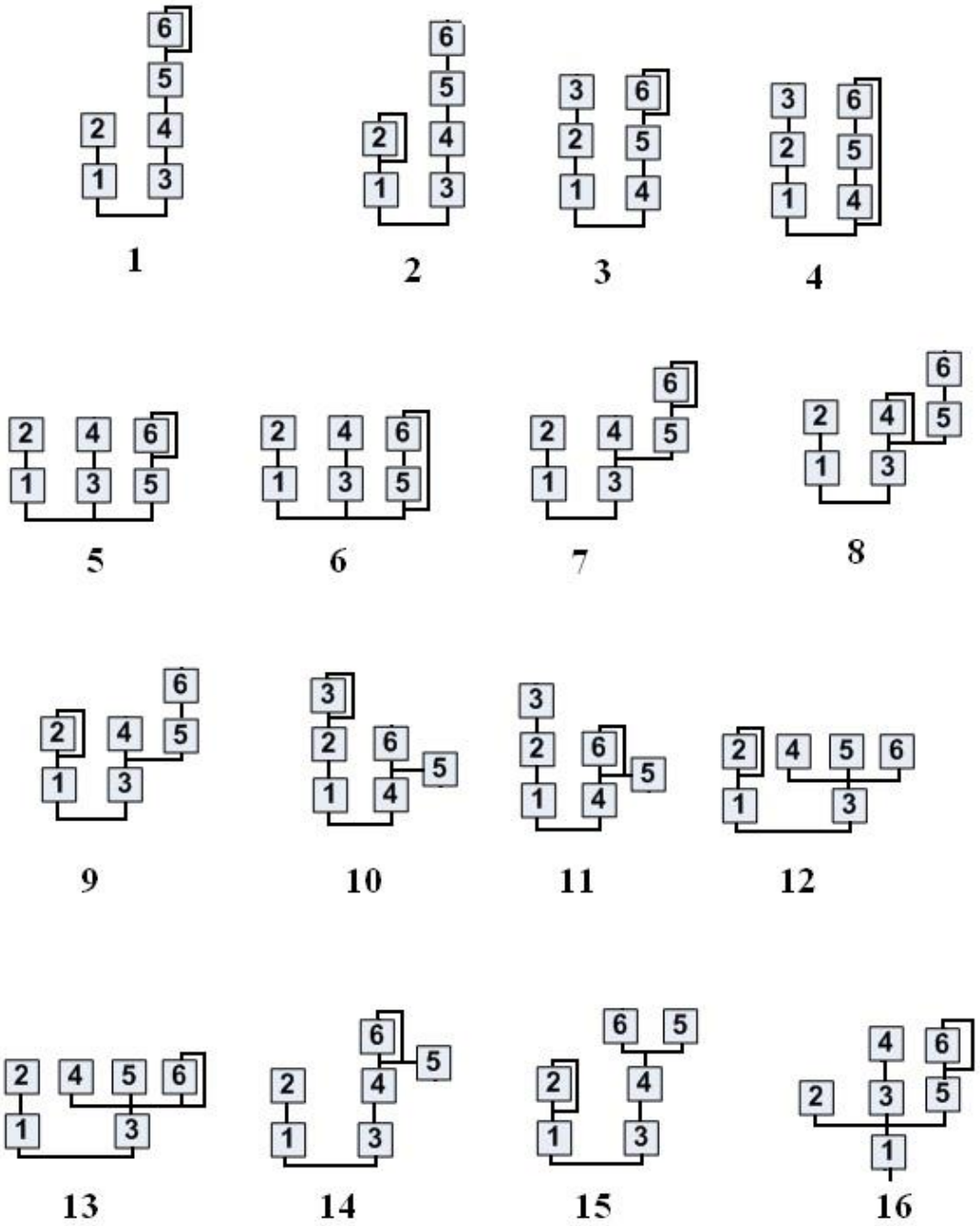


Fonte: Yamaha Corporation (1999, p. 39).

3.3 Algoritmos

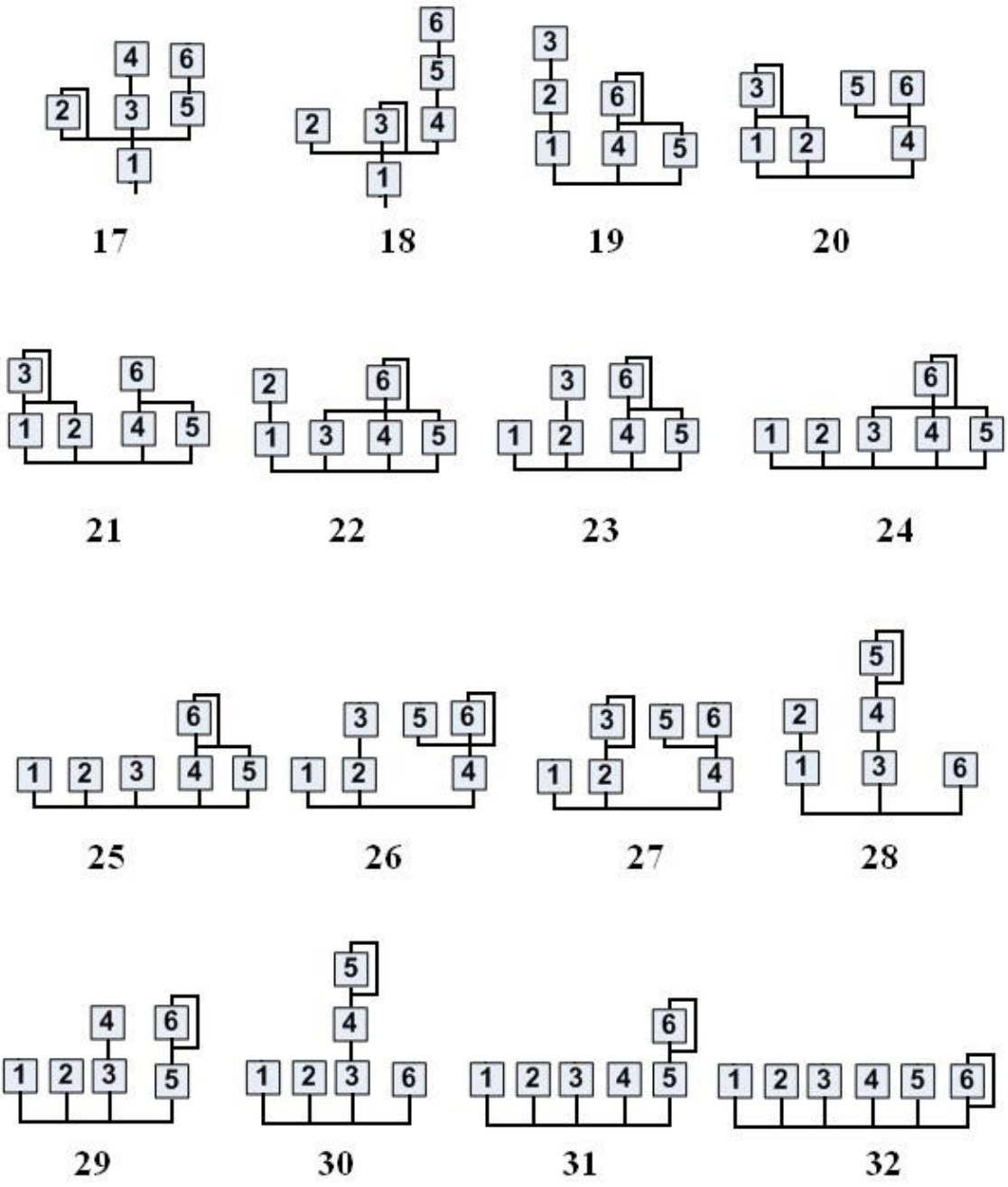
O Yamaha DX7 possui 32 algoritmos. Cada algoritmo é um arranjo de operadores de diferentes formas. Esses algoritmos possuem sempre 6 operadores, sendo que 1 deles sempre tem como entrada a realimentação do valor de um operador na amostra anterior. O valor de realimentação não precisa ser o valor do próprio operador, podendo-se realimentar um operador com seu valor após uma ou mais modulações. A quantidade de operadores portadores e operadores moduladores varia em cada algoritmo. Os operadores moduladores não são ouvidos diretamente. Não é necessário que um operador portador seja modulado, podendo apenas ter seu seno somado na saída do algoritmo. Quanto mais operadores em série, maior a possibilidade de se obter uma complexidade harmônica. Sendo assim, o potencial de se obter uma maior complexidade harmônica aumenta do algoritmo 32 em direção ao algoritmo 1 (YAMAHA CORPORATION, 1999).

Imagem 3.6 – Algoritmos 1 a 16



Fonte: Pettiby (2013).

Imagem 3.7 – Algoritmos 17 a 32



Fonte: Pettiby (2013).

4 ARDUINO DUE

O Arduino Due foi a primeira placa Arduino baseada em um micro controlador com núcleo de 32 bits ARM. Foi uma placa criada para suprir a necessidade de projetos que necessitam de maior poder de processamento e oferecer uma maior gama de possibilidades aos usuários. Se comparada com a placa Arduino padrão (Arduino UNO), obtém-se um clock mais que 5 vezes maior (16MHz contra 84 MHz), quase 4 vezes mais pinos de entrada e saída digitais (14 contra 54) e 2 vezes o número de pinos de entrada analógicos (6 contra 12). O Arduino Due também vem com um DAC embutido (saída analógica), permitindo a saída de áudio diretamente do microcontrolador. Além disso, possui um módulo de hardware dedicado para comunicação I2S, um protocolo serial usado por muitos DACs, permitindo assim conectar esses componentes e transmitir áudio com até 32 bits por amostra por canal (JOHANN, 2015). Esse maior poder de processamento aliado a maior capacidade de conexão à componentes torna o Arduino Due um candidato apto a se implementar projetos de áudio sofisticados, que requerem uma quantidade elevada de cálculos por segundo para atingir taxas de amostragem como 48000Hz.

Imagem 4.1 – Especificações técnicas do Arduino Due

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

Fonte: Arduino (2015).

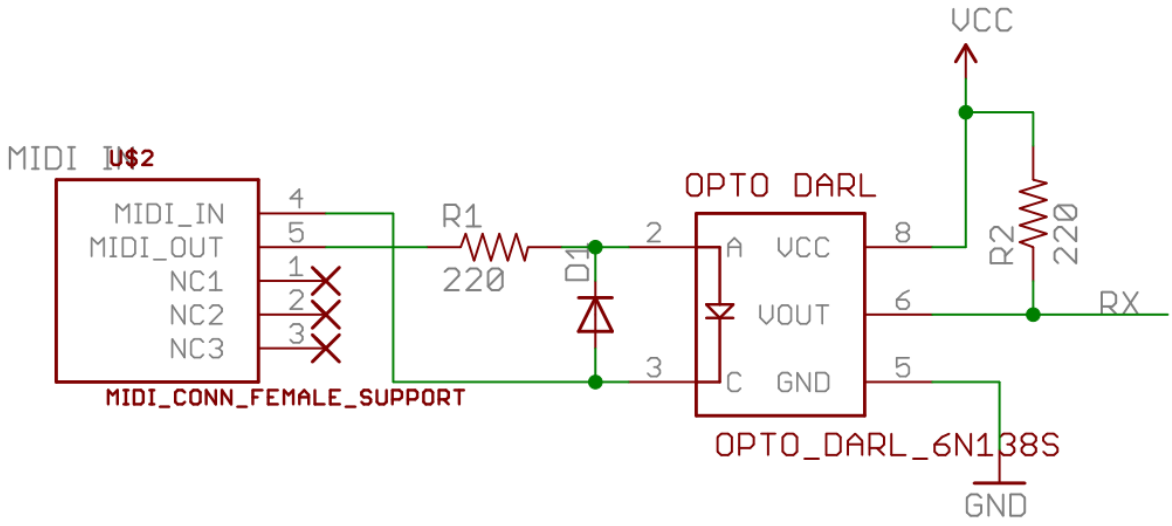
5 IMPLEMENTAÇÃO

Por ter sido implementado num microcontrolador, a implementação possui tanto uma parte em hardware (para entrada de parâmetros e controle do sintetizador) quando em software (para gerar o som).

5.1 Entrada MIDI

O uso do protocolo de comunicação MIDI permite que qualquer controlador com uma saída MIDI seja usado para se tocar o sintetizador no Arduino Due. Para confeccionar-se o circuito da entrada, a especificação de um shield vendido para Arduinos pela SparkFun Electronics foi utilizado com base.

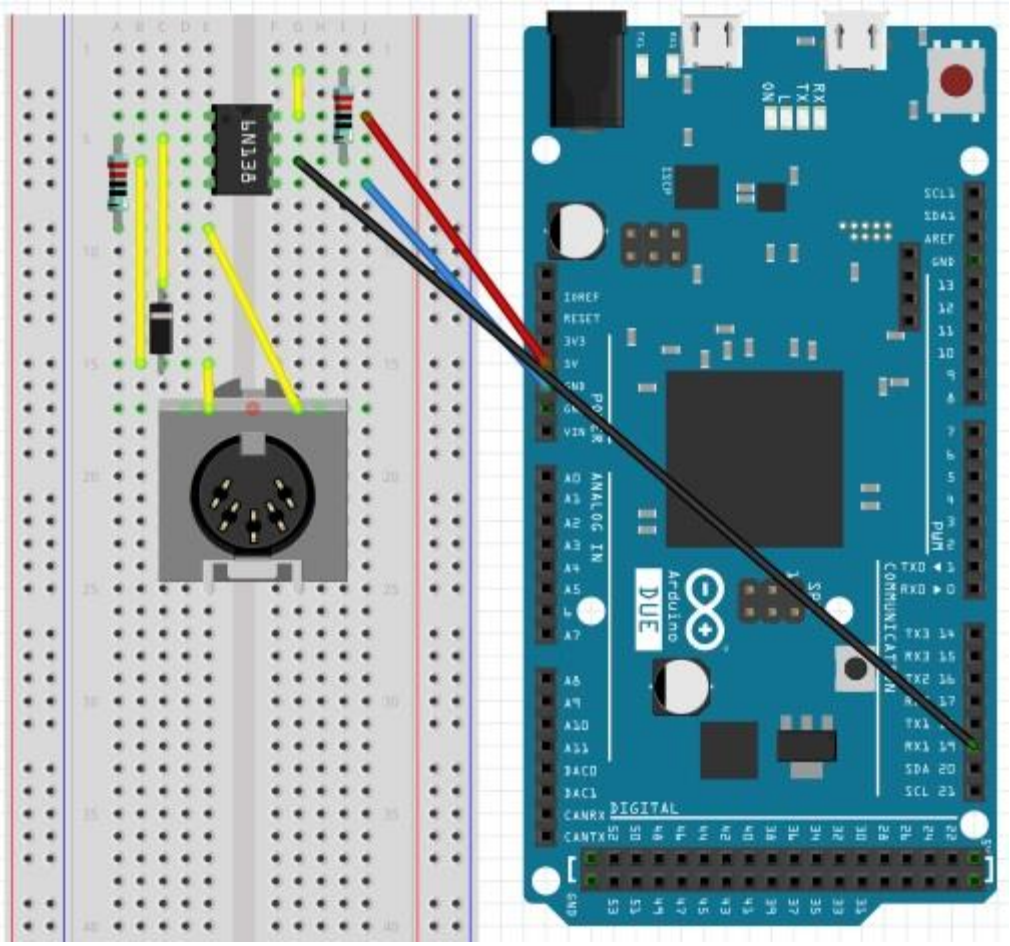
Esquemático 5.1 – Círcuito de uma entrada MIDI



Fonte: SparkFun Electronics (2009).

Foram utilizados um conector MIDI fêmea, 2 resistores de 220Ω, um diodo 1N4148 e um optoacoplador 6N138. Após a montagem, a comunicação com o Arduino Due ocorre por meio de um pino de entrada serial (RX).

Leiaute 5.2 – Entrada MIDI e comunicação com o Arduino Due

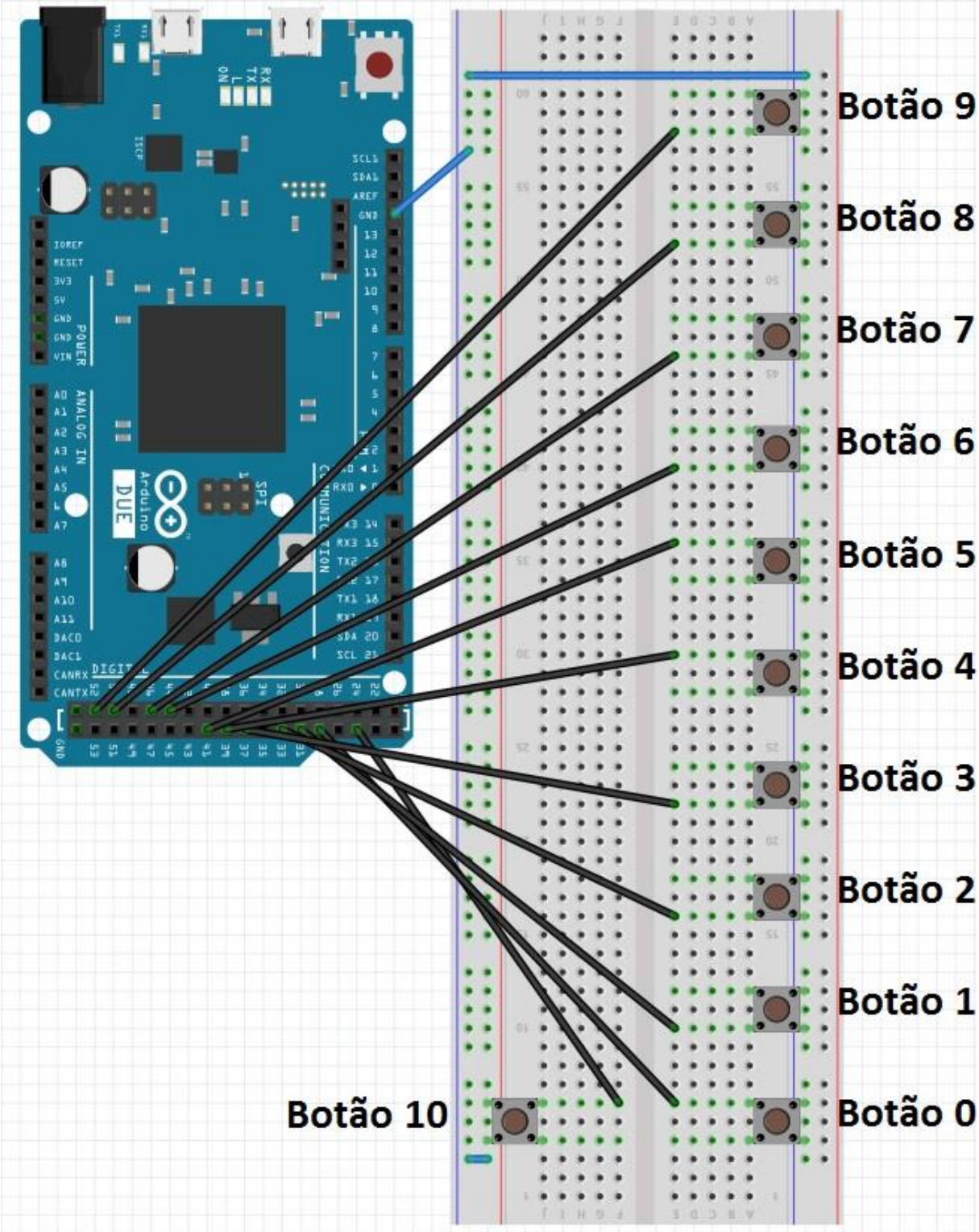


Fonte: Folle (2015).

5.2 Entrada de Parâmetros

A entrada de parâmetros e controle do sintetizador ocorre tanto por botões quanto por knobs. Os botões podem tanto mudar parâmetros internos quando a funcionalidade dos knobs. Todos estão ligados aos pinos digitais do Arduino Due. Um leiaute e uma tabela explicando a função de cada botão podem ser vistos abaixo.

Leiaute 5.3 – Botões de entrada do sintetizador



Fonte: Folle (2015).

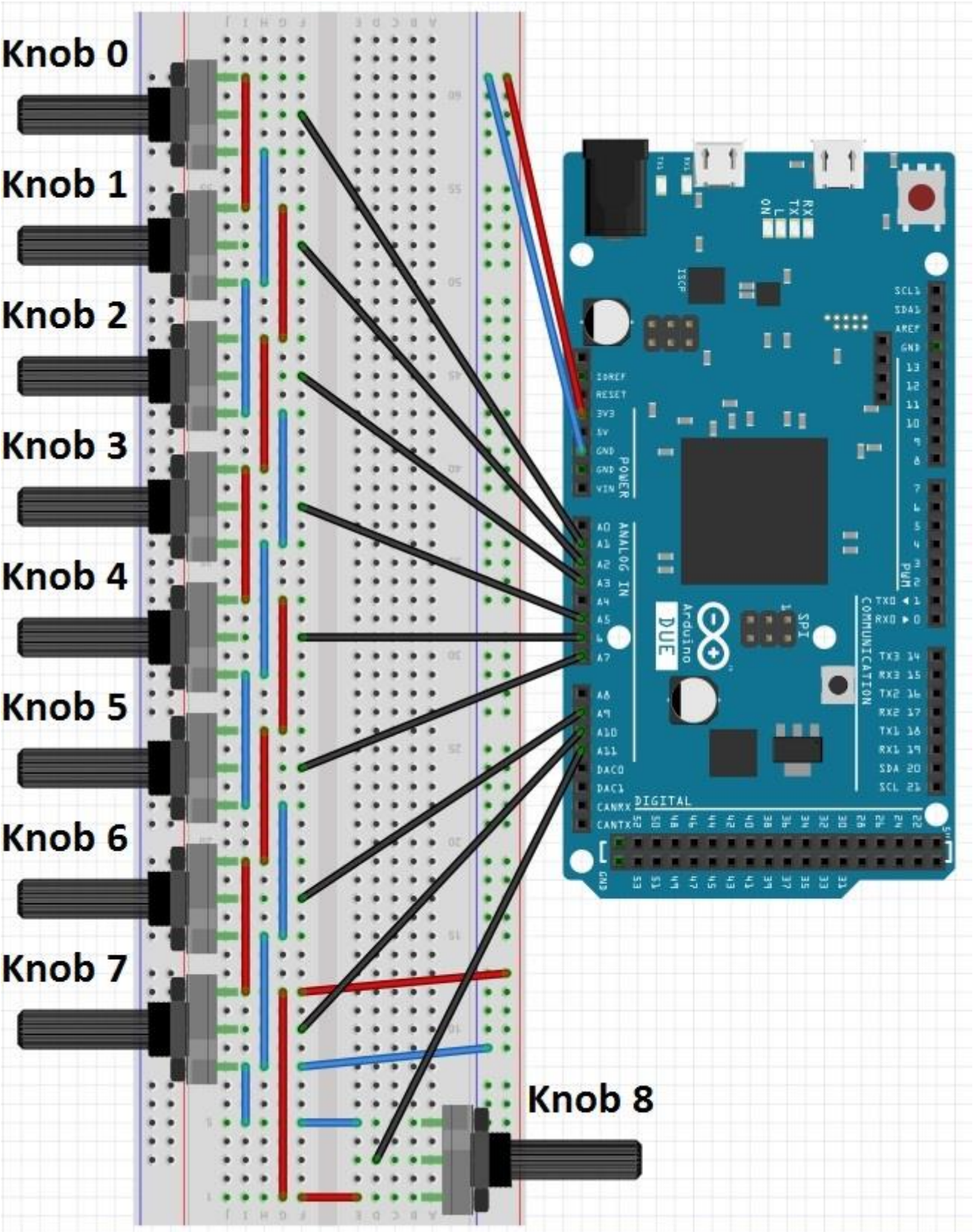
Tabela 5.1 – Função dos botões do sintetizador

Número do botão	*Função*
0	Mudar a função dos knobs para controlarem os parâmetros do operador 1
1	Mudar a função dos knobs para controlarem os parâmetros do operador 2
2	Mudar a função dos knobs para controlarem os parâmetros do operador 3
3	Mudar a função dos knobs para controlarem os parâmetros do operador 4
4	Mudar a função dos knobs para controlarem os parâmetros do operador 5
5	Mudar a função dos knobs para controlarem os parâmetros do operador 6
6	Mudar o algoritmo de síntese (incrementa para o próximo)
7	Mudar a função do escalonamento de nível do lado esquerdo do ponto de quebra
8	Mudar a função do escalonamento de nível do lado direito do ponto de quebra
9	Mudar a função dos knobs para controlarem os parâmetros do envelope de um operador
10	Mudar a função dos knobs para não lerem parâmetros (exceto pelo volume) / Selecionar um timbre programado previamente

Fonte: Folle (2015).

Os knobs por sua vez estão conectados às entradas analógicas do Arduino Due. Eles sempre retornam valores no intervalo [0, 4095], mas o valor real do parâmetro pode ser diferente. Parâmetros de níveis (que são usados para acessar uma tabela posteriormente) são mapeados para valores no intervalo [0, 7]. O parâmetro do ponto de quebra do escalonamento de nível é mapeado para valores de notas MIDI no intervalo [21, 120] o que corresponde a notas entre A0 e C9, inclusive. Já os outros parâmetros são interpretados como variáveis de ponto fixo, com seus intervalos sendo específicos para cada parâmetro. A aritmética de ponto fixo foi utilizada devido à falta de operações nativas de ponto flutuante do Arduino Due. Um leiaute e uma tabela explicando a função de cada knob podem ser vistos abaixo.

Leiaute 5.4 – Knobs de entrada do sintetizador



Fonte: Folle (2015).

Tabela 5.2 – Função dos Knobs

Número do knob	*Parâmetros do operador "x"*	*Envelope do operador "x"*
0	Mudar o valor da razão	Mudar o valor do nível 4
1	Mudar o valor do nível	Mudar o valor da velocidade 1
2	Mudar o valor da realimentação	Mudar o valor do nível 1
3	Mudar o nível da velocidade	Mudar o valor da velocidade 2
4	Mudar o nível do escalonamento de velocidade	Mudar o valor do nível 2
5	Mudar o valor do escalonamento de nível (à esquerda do ponto de quebra)	Mudar o valor da velocidade 3
6	Mudar o ponto de quebra	Mudar o valor do nível 3
7	Mudar o valor do escalonamento de nível (à direita do ponto de quebra)	Mudar o valor da velocidade 4
8	Mudar valor do volume geral do sintetizador	

Fonte: Folle (2015).

O único knob que possui sempre a mesma função é o knob do volume. A tabela abaixo mostra os intervalos de valores atuais dos parâmetros após a utilização da aritmética de ponto fixo. Como todos os parâmetros em ponto fixo realizam uma multiplicação no sintetizador, o valor lido pelos knobs é utilizado para a operação e após um shift de bits para a direita se obtém o valor no intervalo desejado.

Tabela 5.3 – Intervalo dos parâmetros do sintetizador

Parâmetros	*Intervalos*
Razão	Ponto fixo: [0, 32), Q = 7
Nível	Ponto fixo: [0, 2), Q = 11
Realimentação	Ponto fixo: [0, 2), Q = 11
Velocidade	Nível: [0, 7]
Escalonamento de velocidade	Nível: [0, 7]
Escalonamento de nível (esquerda)	Ponto fixo: [0, 2), Q = 11
Ponto de quebra	Notas MIDI: [21, 120]
Escalonamento de nível (direita)	Ponto fixo: [0, 2), Q = 11
Níveis dos envelopes	Ponto fixo: [0, 1), Q = 12
Velocidades dos envelopes	[0, 4095]
Volume	Ponto fixo: [0, 1), Q = 12

Fonte: Folle (2015).

5.3 Bibliotecas

As seguintes bibliotecas foram utilizadas na realização do projeto:

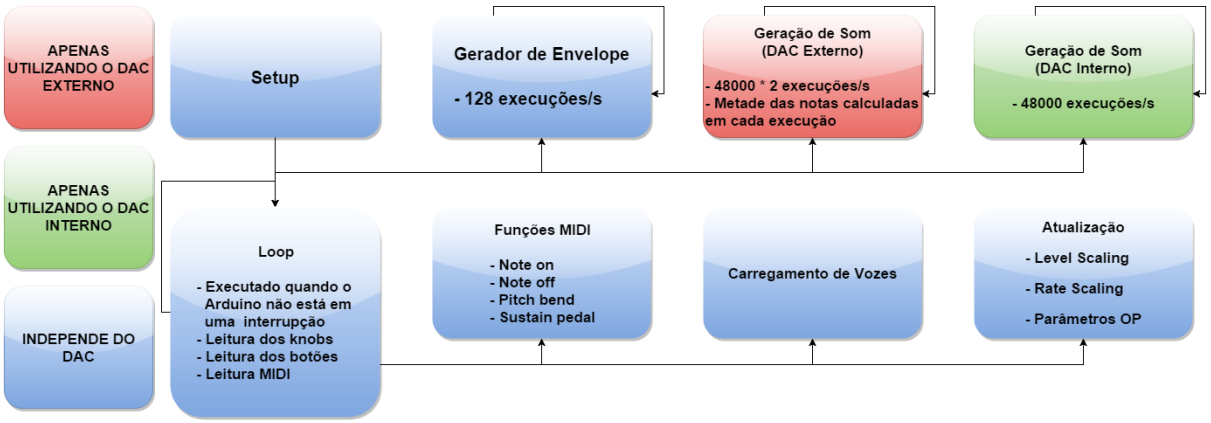
- digitalWriteFast: proporciona uma leitura e escrita mais rápida nos pinos digitais do Arduino Due. Assim é possível obter um tempo de resposta mais rápido às trocas de parâmetros, tanto pelos botões quanto pelos knobs (por estarem no mesmo laço).

- DueTimer: usada para se controlar a geração de som ao se utilizar o DAC interno e para o envelope ADSR.
- MIDI: usada para receber as mensagens MIDI de um controlador externo, de modo a ser possível tocar o sintetizador.
- HiFi: versão modificada da biblioteca (JOHANN, 2015). Usada para a comunicação com um DAC externo.

5.4 Parâmetros Pré-compilação

Antes da compilação é necessário definir se o código deverá ser gerado para trabalhar com o DAC interno do Arduino Due ou com um DAC externo por comunicação I2S. Isso garante que instruções de controle não sejam necessárias, garantindo um maior tempo para se processar as amostras.

Fluxograma 5.5 – Visão geral do código



Fonte: Folle (2015).

5.5 Tabela de Seno e Tabela de Passos

Para se obter o ponto de partida da síntese FM (uma onda de seno), foi construído um arranjo de 1024 posições com valores variando de -512 a 511, de modo a se obter o formato de uma onda senoidal. Percorrendo-se esta tabela com diferentes velocidades obtém-se frequências diferentes, ou seja, notas diferentes. Um segundo arranjo de tamanho igual ao número de notas possíveis de serem reproduzidas contém o tamanho do passo que cada nota deve dar em cada acesso à tabela de seno, tendo como base a taxa de amostragem de 48000Hz. Os valores foram obtidos com base em (CURTIS, 1996). Como a diferença entre passos é pequena, valores não

inteiros são necessários para sua representação. Os passos são então representados em ponto fixo, possuindo 11 bits para a parte decimal.

5.6 Operadores

Assim como o Yamaha DX7, o sintetizador possui 6 operadores. Para cada nota a ser reproduzida, cada operador tem seu valor calculado com base a uma consulta na tabela de seno. O índice a ser consultado é obtido pela soma de um acumulador de passos da nota e os outros possíveis argumentos: realimentação e operadores moduladores. Após obter o valor da tabela, o envelope ADSR é aplicado e por fim aplica-se o nível do operador. Como o nível do operador pode variar para cada nota (devido à velocidade MIDI e ao escalonamento de nível), cada nota tem o seu próprio valor de nível a ser utilizado para cada um dos operadores pré-calculado. Um exemplo do código dos operadores no algoritmo 1 pode ser visto abaixo.

Imagem 5.6 – Operadores do algoritmo 1

```
// OP 1.
operators[1].value =
(((sineTable[(notes[i].stepAccumulator[1] >> STEP_SHIFT) & (SINE_TABLE_SIZE - 1)]
* notes[i].adsrStatus[1].value) >> ADSR_SHIFT) * notes[i].scaledLevel[1]) >> LEVEL_SHIFT;

// OP 0.
operators[0].value =
(((sineTable[((notes[i].stepAccumulator[0] >> STEP_SHIFT) + operators[1].value) & (SINE_TABLE_SIZE - 1)]
* notes[i].adsrStatus[0].value) >> ADSR_SHIFT) * notes[i].scaledLevel[0]) >> LEVEL_SHIFT;

// OP 5 and feedback.
operators[5].value = (((sineTable[((notes[i].stepAccumulator[5] >> STEP_SHIFT) + notes[i].feedback) & (SINE_TABLE_SIZE - 1)]
* notes[i].adsrStatus[5].value) >> ADSR_SHIFT) * notes[i].scaledLevel[5]) >> LEVEL_SHIFT;

notes[i].feedback = (operators[5].value * operators[5].feedback) >> FEEDBACK_SHIFT;

// OP 4.
operators[4].value = (((sineTable[((notes[i].stepAccumulator[4] >> STEP_SHIFT) + operators[5].value) & (SINE_TABLE_SIZE - 1)]
* notes[i].adsrStatus[4].value) >> ADSR_SHIFT) * notes[i].scaledLevel[4]) >> LEVEL_SHIFT;

// OP 3.
operators[3].value = (((sineTable[((notes[i].stepAccumulator[3] >> STEP_SHIFT) + operators[4].value) & (SINE_TABLE_SIZE - 1)]
* notes[i].adsrStatus[3].value) >> ADSR_SHIFT) * notes[i].scaledLevel[3]) >> LEVEL_SHIFT;

// OP 2.
operators[2].value = (((sineTable[((notes[i].stepAccumulator[2] >> STEP_SHIFT) + operators[3].value) & (SINE_TABLE_SIZE - 1)]
* notes[i].adsrStatus[2].value) >> ADSR_SHIFT) * notes[i].scaledLevel[2]) >> LEVEL_SHIFT;
```

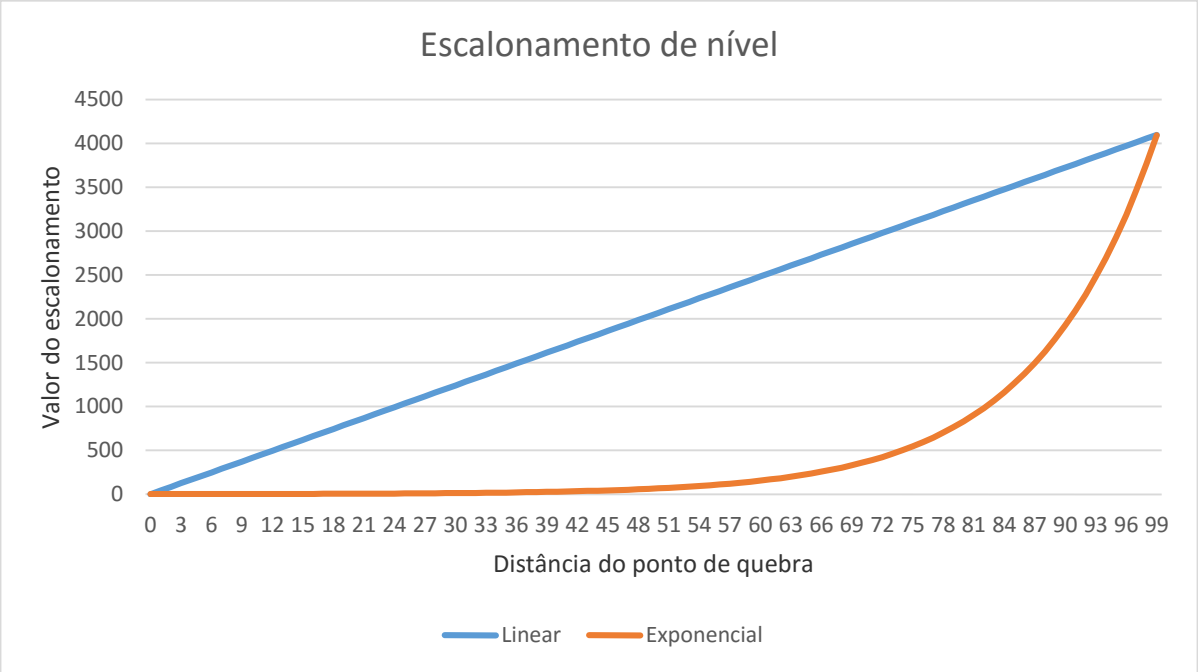
Fonte: Folle (2015).

5.6.1 Escalonamento de Nível

Para se emular o escalonamento de nível do DX7, foram salvas 2 curvas na memória do Arduino Due, com valores no intervalo [0, 4095]. Quando o parâmetro necessita ser calculado, pega-se o valor absoluto da distância de uma nota até o ponto de quebra e então utiliza-se ele para obter-se um ponto em uma curva. A curva pode ser linear positiva, linear negativa, exponencial positiva e exponencial negativa. O tipo da curva muda o arranjo que deve ser consultado, e o seu sinal define se o valor deve ser somado ou subtraído do valor de um operador

(sempre ficando dentro do limite [0, 4095]). O valor do escalonamento de nível, tanto esquerdo como direto, serve para definir o valor máximo que a curva selecionada pode atingir. Para evitar a necessidade de 4095 curvas para cada função, um mapeamento do valor da curva no intervalo de [0, 4095] para o intervalo de 0 ao valor do parâmetro é realizado.

Gráfico 5.7 – Escalonamento de nível

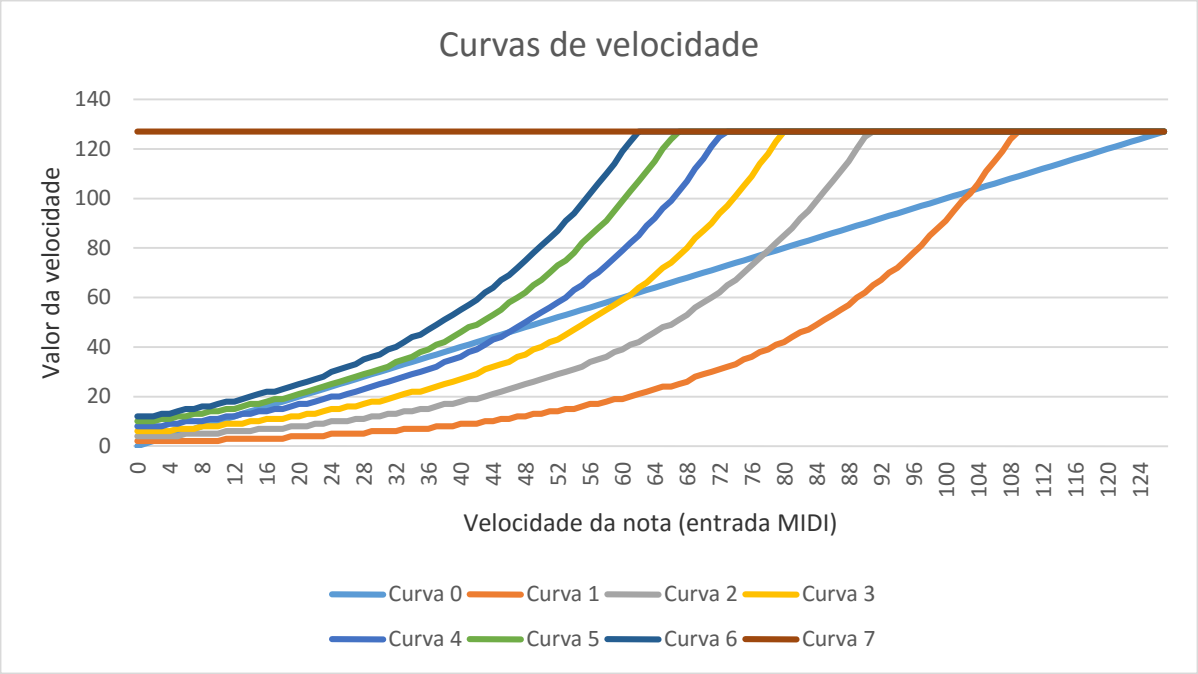


Fonte: Folle (2015).

5.6.2 Curvas de Velocidade MIDI

O sintetizador possui suporte à velocidade MIDI, alterando o valor do nível do operador com base na velocidade que o usuário pressiona a tecla de uma nota. O usuário deve selecionar uma das 8 curvas de velocidades disponíveis. O valor da velocidade lido é mapeado do intervalo da velocidade MIDI [0, 127] para o intervalo do operador, de 0 até o nível após a aplicação do escalonamento de nível. O valor obtido é então usado como sendo o nível do operador para a nota em questão. Como é possível observar no gráfico abaixo, há uma curva em que os valores lidos são iguais ao da velocidade retornada pelo protocolo MIDI, uma curva em que não importando o valor lido o valor retornado é sempre o máximo (seria o caso em que a velocidade está desligada), e 6 curvas exponenciais com incrementos de 2 unidades por nível de curva.

Gráfico 5.8 – Curvas de velocidade MIDI



Fonte: Folle (2015).

5.7 Gerador de Envelope

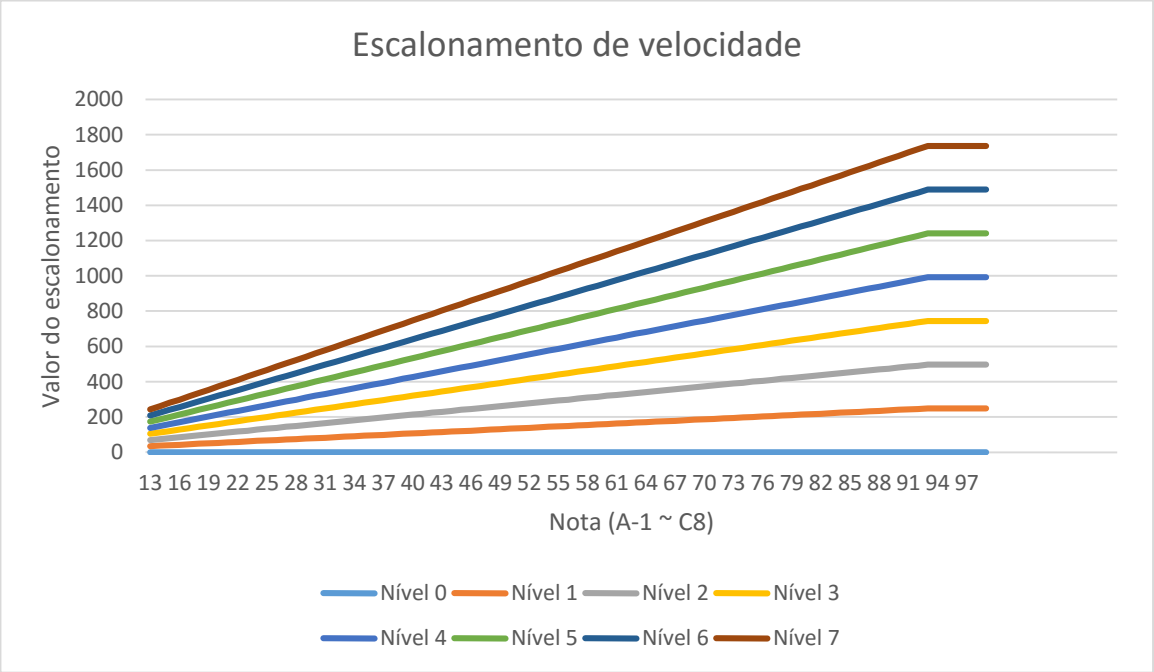
O sintetizador possui 6 envelopes, um para cada operador. O envelope possui os mesmos parâmetros do envelope do Yamaha DX7 (nível 1 até 4 e velocidade 1 até 4). Sua implementação foi feita com o auxílio de um timer que realiza 128 interrupções por segundo. Este valor foi aproximado com base em "LEVEL 1 (L1) is the level to which the operator begins moving as soon as you press down the key. The operator may reach L1 almost instantaneously, or it may take over a half a minute to reach LEVEL1, depending on the setting of RATE 1 (R1)" (Yamaha DX7 Manual, 1999, p. 26). Como os valores do nível e velocidade variam de 0 a 4095, considerando o caso mais demorado (nível x = 4095, nível (x+1) = 0 e velocidade x = 1), seriam necessárias 136.5 interrupções por segundo para o próximo nível ser atingido em meio segundo. Como o envelope pode demorar mais que meio segundo, a frequência de interrupções utilizada foi de 128 interrupções por segundo.

O incremento de cada interrupção para cada uma das 4 partes do envelope é calculado previamente e armazenado numa variável de ponto fixo com 19 bits de parte decimal, sendo necessário apenas somá-lo a cada interrupção. No caso de a nota ser solta antes do envelope chegar no nível 3, o valor dos incrementos é novamente calculado e o envelope volta à execução normal.

5.7.1 Escalonamento de Velocidade

Assim como o DX7, o sintetizador permite que as velocidades dos envelopes sejam gradualmente aumentadas das notas mais graves para as notas mais agudas. Existem 8 níveis possíveis de escalonamento. No nível 0, não ocorre nenhum tipo de escalonamento. Já para os níveis posteriores, cada nível aumenta o intervalo de escalonamento em 248 unidades (de modo a similar ao DX7). Assim, para cada nota, as velocidades dos envelopes devem ser somadas com o valor obtido pela curva. As curvas são lineares (por exceção das últimas notas), assim como o mapeamento dos seus valores para cada nota.

Gráfico 5.9 – Escalonamento de velocidade



Fonte: Folle (2015).

5.8 Algoritmos

Os algoritmos do sintetizador possuem a mesma disposição de operadores dos algoritmos do Yamaha DX7. Após a saída do algoritmo ser computada para cada nota ativa, o valor do volume geral do sintetizador é aplicado e a saída é enviada para o DAC interno ou para o DAC externo.

5.9 DAC Interno e DAC Externo

É importante notar que a maneira como cada amostra é computada depende do DAC que está sendo utilizado. O DAC interno foi usado para a fase de testes da implementação. Porém, como possui um ruído muito alto e apenas 12 bits de resolução, foi incluída a opção de se utilizar um DAC externo. No caso do DAC interno, a função de geração de som é chamada por um timer 48000 vezes por segundo (valor da taxa de amostragem). Todas as notas são então computadas e o valor da amostra é enviado ao DAC interno. Já no caso do DAC externo, o Arduino Due recebe uma interrupção por canal de áudio para cada amostra, o que ao se utilizar uma taxa amostragem de 48000Hz implica em 96000 interrupções. Além disso, ao chamar a função de geração de som, o DAC externo espera que a amostra já esteja pronta para ser lida, o que não é um comportamento natural para sintetizadores de sons (JOHANN, 2015). Como o sintetizador opera em modo monofônico, metade da polifonia é calculada na interrupção do canal esquerdo e armazenada em um buffer. A outra metade é então calculada na interrupção do canal direito e adicionada ao valor já calculado. Quando todas as notas forem computadas, seus valores são então armazenados do buffer para a variável que será lida pelo DAC externo. Desse modo a amostra estará sempre pronta logo no início das interrupções. Esse funcionamento também implica que as amostras requisitadas estarão sempre uma amostra atrasada.

6 CONCLUSÃO

A síntese FM permite a geração dos mais variados timbres (desde de timbres de ondas básicas até outros bastante complexos) com o uso de poucos recursos e poucas instruções de código. Além disso, o fato de ser totalmente digital permite a sua implementação nos mais diversos sistemas. A introdução de microcontroladores mais potentes no mercado como o Arduino Due permite uma exploração artística e de aplicações de áudio de maior complexidade. Fato este que tornou possível a implementação de um sintetizador FM. Apesar de ainda ser uma área pouco explorada, microcontroladores possuem a capacidade de serem usados para aplicações mais refinadas de áudio.

REFERÊNCIAS

YAMAHA CORPORATION. **DX7 Authorized Product Manual**: Digital Programmable Algorithm Synthesizer. Buena Park: Yamaha Corporation of America, 1999.

YAMAHA CORPORATION. **DX7/9 Service Manual**: Digital Programmable Algorithm Synthesizer. Hamamatsu: Nippon Gakki CO., LTD., 1983.

YAMAHA CORPORATION. **DX Simulator Manual**. Version 1.1. [S.l.:s.n], 2002.

CHOWNING, J.; BRISTOW, D. **FM Theory & Applications**: By Musicians for Musicians. Tokyo: Yamaha Music Foundation, 1986.

PUCKETTE, Miller. **The Theory and Technique of Electronic Music**. Draft. [S. l.]: World Scientific Publishing Co. Pte. Ltd., 2007.

ROADS, C. Sound Synthesis. In: ROADS, C. **The Computer Music Tutorial**. Cambridge: The MIT Press, 1996.

MASSEY, Howard. **The Complete DX7.**. Londres: Amsco Publications, 1986.

JOHANN, M. de O. **An Additive Synthesis Organ with Full Polyphony on Arduino Due**. 2015. 11 f. (Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

TOMLYN, Bo. How to Program the DX7: Understanding and Applying the Concepts of Digital FM Synthesis. **Keyboard Mag,** [S. l.], p. 66-71, jun. 1985. Disponível em: <https://homepages.abdn.ac.uk/mth192/pages/dx7/manuals/prgrmdx7.pdf>. Acesso em 07 out. 2015.

YAMAHA CORPORTATION. Yamaha Synth 40 Years History. 2014. Disponível em: <http://usa.yamaha.com/products/music-production/synthesizers/synth_40th/history/>. Acesso em 28 nov, 2015.

YAMAHA CORPORTATION. Yamaha DX7. Disponível em: <http://usa.yamaha.com/products/music-production/synthesizers/dx7/>. Acesso em 28 nov. 2015

ARDUINO. Arduino UNO. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardUno>. Acesso em 28 nov. 2015.

ARDUINO. Arduino Due. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardDue>. Acesso em 06 out. 2015.

SPARKFUN ELECTRONICS. MIDI_Breakout-v11. 21 dez. 2009. Disponível em: <https://www.sparkfun.com/datasheets/BreakoutBoards/BOB-09598-MIDI_Breakout-v11.pdf>. Acesso em 07 out. 2015.

SALISBURY, D. F. Stanford and Yamaha Pioneer Partnership in Music Synthesis. **Stanford News Service**, Stanford, jul. 1997. Disponível em: <http://news.stanford.edu/pr/97/970709sondiusxg.html>. Acesso em 28 nov. 2015.

BEST, F. Arduino Midi Library v4.2. Disponível em: <https://github.com/FortySevenEffects/arduino_midi_library>. Acesso em 6 out. 2015.

DELSAUSE. ArduinoDueHifi: An I2S audio codec driver library for the Arduino Due board. Disponível em: <https://github.com/delsauce/ArduinoDueHiFi>. Acesso em 6 out. 2015.

IVANSEIDEL. DueTimer: Timer library to work with Arduino Due. Disponível em: <https://github.com/ivanseidel/DueTimer>. Acesso em 6 out. 2015.

PETTIBY, C. Dubspot First Look: The Propellerhead PX7 FM Synthesizer In Reason 6.5 w/ Chris Pettiby. Disponível em: <http://blog.dubspot.com/propellerhead-px7-fm-synthesizer/>. Acesso em 1 dez. 2015.