**UNIVERSIDADE DE SÃO PAULO**

**ESCOLA DE ENGENHARIA DE SÃO CARLOS**

HEITOR GOMES DA SILVA

# Sintetizador musical baseado em microcontrolador ARM7 para criação de trilha sonora utilizando controle por varredura matricial

São Carlos

2011

**HEITOR GOMES DA SILVA**

# SINTETIZADOR MUSICAL BASEADO EM MICROCONTROLADOR ARM7 PARA CRIAÇÃO DE TRILHA SONORA UTILIZANDO CONTROLE POR VARREDURA MATRICIAL

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo.

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Prof. Dr. Marcelo Andrade da Costa Vieira

São Carlos

2011

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

S586s

Silva, Heitor Gomes da
Sintetizador musical baseado em microcontrolador ARM7 para criação de trilha sonora utilizando controle por varredura matricial / Heitor Gomes da Silva ; orientador Marcelo Andrade da Costa Vieira -- São Carlos, 2011.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2011.

1. Microcontrolador. 2. ARM. 3. AT91SAM7X256. 4. SAM7-EX256. 5. Sintetizador musical. I. Titulo.

*Dedico este trabalho especialmente à minha família e minha amada noiva, que sempre se mantiveram ao meu lado, apesar de toda a distância.*

# AGRADECIMENTOS

Ao Prof. Dr. Marcelo Andrade da Costa Vieira, pelo apoio na concepção, elaboração e orientação deste trabalho.

Ao Prof. Dr. Evandro Luís Linhari Rodrigues, pelo suporte material, cedendo o espaço físico de seu laboratório e o kit de ensino para o desenvolvimento deste projeto.

Aos professores da graduação que sempre estiveram dispostos a orientar e ajudar durante o período da graduação.

Aos amigos que estiveram sempre juntos, convivendo dia após dia dentro e fora da faculdade.

Aos colegas e amigos que ajudaram, sempre que possível, a sanar as dúvidas no desenvolvimento deste projeto.

# LISTA DE SIGLAS

LED – Light-Emitting Diode (Diodo Emissor de Luz)

TCC – Trabalho de Conclusão de Curso

RISC – Reduced Instruction Set Computer

MCU – Microcontroller Unit

ARM – Advanced RISC Machine

ROM – Read-Only Memory

LCD – Liquid Crystal Display (Display de Cristal Líquido)

SPI – Serial Peripheral Interface

PWM – Pulse-Width Modulation (Modulação por Largura de Pulso)

CPU – Central Processing Unit (Unidade Central de Processamento)

MUX – Multiplexador

CI – Circuito Integrado

BJT – Bipolar Junction Transistor (Transistor Bipolar de Junção)

PCB – Printed Circuit Board (Placa de Circuito Impresso)

RTT – Real Time Timer

SLCK – Slow Clock

MCK – Master Clock

# RESUMO

SILVA, H. G. **Sintetizador musical baseado em microcontrolador ARM7 para criação de trilha sonora utilizando controle por varredura matricial**. Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

Neste trabalho de conclusão de curso foi desenvolvido um protótipo de um sintetizador musical baseado em varredura matricial. O instrumento musical confeccionado consiste de uma matriz de 5x16 botões com LED que fazem a interface com o usuário, em que cada linha representa uma nota musical e a varredura é feita coluna por coluna. O projeto foi concebido utilizando-se como base um kit de ensino SAM7-EX256, produzido pela Olimex, e que possui um MCU ARM7 (AT91SAM7X256), fabricado pela Atmel. Como resultado, obteve-se um protótipo que é capaz de reproduzir diferentes notas e compor diversos acordes, de acordo com os botões que são pressionados, além de poder controlar a velocidade de varredura e a tonalidade da escala musical reproduzida.

Palavras Chave: Microcontrolador, ARM, AT91SAM7X256, SAM7-EX256, sintetizador musical.

# ABSTRACT

SILVA, H.G. **ARM7 microcontroller based music synthesizer used to create a matrix scanning control soundtrack**. Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

In this work of completion it was developed a musical synthesizer prototype based on matrix scanning control. The built musical instrument consists in a 5x8 LED matrix with push switches that works as an interface with the user. Each row represents a different musical note and the scanning process is executed column by column. The Project was conceived using the SAM7-EX256 development board, produced by Olimex. This board has an ARM7 based MCU (AT91SAM7X256), by Atmel. The final result was a prototype capable of reproducing different notes and many chords, depending on which push button is pressed. Other feature present in this prototype is the capability to control the scanning speed and the tone of the musical scale played.

Key words: Microcontroller, ARM, AT91SAM7X256, SAM7-EX256, musical synthesizer.

# SUMÁRIO

# LISTA DE FIGURAS

# Capítulo 1

## 1. Introdução

O mundo da música é um dos muitos em que a eletrônica vem se desenvolvendo e ganhando cada dia mais importância. É difícil imaginar qualquer produção musical sendo concebida sem a utilização de algum instrumento ou alguma forma de captação e reprodução que não utilize sequer algum componente elétrico ou eletrônico.

Existem instrumentos cuja essência é analógica, no entanto utiliza-se a eletrônica para a captação e reprodução da onda sonora gerada, como no caso de um violão ou um piano em um concerto musical. Existem outros casos em que o som produzido é captado e processado através de circuitos elétricos, como no caso de uma guitarra elétrica com um pedal de distorção. E há ainda instrumentos cujo sinal sonoro é totalmente produzido de forma eletrônica, como no caso do teclado eletrônico ou do sintetizador.

O sintetizador é um instrumento musical eletrônico capaz de gerar sons através da manipulação de sinais elétricos. Ele pode tanto reproduzir timbres de outros instrumentos musicais, como o som de um piano, uma guitarra ou um instrumento de percussão, quanto gerar novos timbres ou sons impossíveis de serem obtidos de maneira natural.

A Yamaha produziu um tipo de sintetizador musical chamado de Tenori-on (Figura 1.1), que consiste de um display com uma matriz de 16x16 botões com LEDs, que podem ser pressionados de forma a produzir diferentes sequências musicais. É neste instrumento que está baseado o protótipo produzido neste trabalho de conclusão de curso. O funcionamento do projeto é inspirado na funcionalidade de “varredura sequencial da tela” do dispositivo original. Neste modo, a tela do instrumento é lida continuamente e é tocada a nota correspondente aos botões que estão ativados no momento da leitura. Na matriz de botões, cada linha horizontal representa uma nota musical e as colunas determinam cada “acorde” da sequência.

A utilização de microcontroladores para esta aplicação é uma alternativa muito eficiente, já que, a partir dele, é possível processar o sinal de áudio, armazenar dados e controlar os periféricos do sistema.

**Figura 1.1 - O Sintetizador Tenori-on da Yamaha**

## 1.1. Objetivos do trabalho

O objetivo geral deste Trabalho de Conclusão de Curso foi o de desenvolver um projeto baseado em microcontrolador para construção de um instrumento musical eletrônico cuja interface com o usuário é um teclado matricial. O projeto é baseado no sintetizador musical da Tenori-on, da Yamaha, e visa produzir um produto semelhante com tecnologia nacional e com um custo reduzido, através da utilização do microcontrolador fabricado pela Atmel, o AT91SAM7X256, presente no *kit* de desenvolvimento SAM7-EX256, produzido pela Olimex.

No desenvolvimento do projeto, os objetivos específicos abordados foram:

a) Estudo do funcionamento do microcontrolador da família ARM7;
b) Estudo do protocolo de comunicação com SPI para interface com o *display* de LCD;
c) Geração de sinal de áudio a partir de um PWM interno ao microcontrolador;
d) Programação em C direcionada a microcontroladores;
e) Projeto, desenvolvimento e confecção de um circuito eletrônico específico;
f) Interface entre o circuito eletrônico projetado e o microcontrolador AT91SAM7X256.

# Capítulo 2

# 2. Fundamentação Teórica

## 2.1. O Microcontrolador ARM

O microcontrolador funciona como um computador completo integrado dentro de um único *chip*. Além do microprocessador, ele possui memória e diversos periféricos programáveis, tais como temporizadores, portas paralelas de entrada e saída, conversores AD/DA, entre outros (HEATH, 2003). A sua versatilidade o torna perfeito para aplicações em sistemas embarcados, estações de controle, sistemas de automação e etc.

Os microcontroladores ARM (*Advanced RISC Machine*) são uma família de MCU que possuem um núcleo baseados na arquitetura RISC (*Reduced Instruction Set Computer*) de 32 bits. Esta arquitetura tem como característica uma estrutura de instruções mais simplificada, que tem como objetivo otimizar a velocidade de execução do MCU. Além das instruções de 32 bits da arquitetura ARM, a partir da família ARM7TDMI foi criado um *set* de instruções de 16 bits denominado *Thumb*. Estas instruções, evidentemente, são mais simplificadas e possuem algumas limitações em relação às instruções de 32 bits, no entanto reduzem consideravelmente o espaço de memória consumido pelo código do programa. Posteriormente, a partir da família ARM1156 foi introduzido o *Thumb-2*, que complementa algumas limitações do seu predecessor através da introdução de algumas instruções de 32 bits. Dessa forma, o *Thumb-2* procura o equilíbrio entre o desempenho do conjunto de instruções ARM de 32 bits e a densidade de código do *Thumb*. Outra característica importante dos microcontroladores ARM é o baixo consumo de energia, fazendo com que sejam amplamente utilizados em dispositivos móveis e em circuitos embarcados (PEREIRA, 2007).

A elaboração da arquitetura ARM foi feito pela empresa Acorn (inicialmente ARM era a sigla para *Acorn RISC Machine*). Posteriormente foi identificado o potencial dessa tecnologia e outras empresas tornaram-se parceiras no seu desenvolvimento, entre elas a Apple e a VLSI e foi criada a ARM Holding, detentora da licença desta arquitetura. Atualmente diversos fabricantes de semicondutores oferecem microcontroladores baseados nesta tecnologia, entre eles Atmel, Texas Instrument, Freescale, STMicroeletronics e NXP Semiconductors (Wikipedia, 2011).

Dentre os microcontroladores ARM existem diversas famílias que possuem algumas características específicas, dependendo da geração. Dentre todas as famílias ARM, as mais populares são a ARM7, ARM9, ARM11 e Cortex, sendo a última, amplamente utilizado em aparelhos portáteis, como *tablets* e *smartphones* (ARM Ltd., 2011).

Uma das características que diferenciam estas gerações é a quantidade de estágios de *pipeline*. O *pipeline* representa a quantidade de instruções que o CPU pode deixar na "fila" para as execuções. Isso significa que não é necessário aguardar todos os ciclos de execução de uma instrução para que a próxima seja iniciada. Na prática, esse processo otimiza a velocidade de execução do programa, diminuindo a quantidade de ciclos de *clock* entre uma instrução e outra. No ARM7 existem três estágios de *pipeline* (Atmel Corporation, 2009), enquanto que nas famílias ARM9, ARM11 e Cortex existem, respectivamente, cinco, oito e onze estágios.

A versatilidade e a eficiência dos microcontroladores da família ARM fez com esta seja uma das arquiteturas mais bem sucedidas no mercado atualmente. Ela é utilizada em inúmeras aplicações que exigem desde um baixo consumo de potência até um alto desempenho de processamento.

Na Tabela 1 é apresentado um resumo de algumas famílias ARM, com a especificação de sua respectiva arquitetura, tipo de núcleo e alguns dispositivos que utilizam essa tecnologia (Wikipedia, 2011).

**Tabela 1 - Tabela de famílias da arquitetura ARM**

| ARM Family | ARM Architecture | ARM Core | Dispositivos |
|---|---|---|---|
| ARM7 | ARMv3 | ARM700 | |
| | | ARM710 | |
| | | ARM710a | |
| ARM7TDMI | ARMv4T | ARM7TDMI(-S) | Game Boy Advance, Nintendo DS, Apple iPod, Lego NXT |
| | | ARM710T | Psion Series 5mx, Psion Revo/Revo Plus/Diamond Mako |
| | | ARM720T | Zipit Wireless Messenger |
| | | ARM740T | |
| ARM9TDMI | ARMv4T | ARM9TDMI | |
| | | ARM920T | Hewlett-Packard HP-49/50 Calculators, Garmin Navigation Devices, TomTom navigation devices |
| | | ARM922T | |
| | | ARM940T | GP2X (second core), Meizu M6 Mini Player |

| | | | |
|---|---|---|---|
| ARM9E | ARMv5TE | ARM946E-S | Nintendo DS, Nokia N-Gage, Canon PowerShot A470, Canon EOS 5D Mark II |
| | | ARM966E-S | |
| | | ARM968E-S | |
| | ARMv5TEJ | ARM926EJ-S | |
| | ARMv5TE | ARM996HS | |
| ARM11 | ARMv6 | ARM1136J(F)-S | Nokia E90, Nokia N93, Nokia N95, Nokia N82, Zune, HTC Dream, HTC Magic, Motorola i1, Motorola Z6 |
| | ARMv6T2 | ARM1156T2(F)-S | |
| | ARMv6ZK | ARM1176JZ(F)-S | Apple iPhone (original e 3G), Apple iPod touch (1ª e 2ª Geração), Motorola RIZR Z8, Motorola RIZR Z10, Nintendo 3DS |
| | ARMv6K | ARM11 MPCore | |
| Cortex-A | ARMv7-A | Cortex-A5 (MPCore) | |
| | | Cortex-A8 | Apple iPhone 3GS, Apple iPod touch (3ª e 4ª Geração), Google Nexus S |
| | | Cortex-A9 MPCore | Apple iPad 2, LG Optimus 2X, LG Optimus 3D, Motorola Atrix 4G, Motorola DROID BIONIC, Motorola Xoom |
| | | Cortex-A15 MPCore | |

## 2.2. O Microcontrolador Atmel AT91SAM7X256

O AT91SAM7X256 é um microcontrolador produzido pela Atmel e que faz parte da série SAM7X de microcontroladores com base na família ARM7TDMI de 32 bits. Além do *set* de instruções ARM, esta família possui suporte ao *set* de instruções *Thumb* de 16 bits.

Este MCU contém 256KBytes de memória de programa organizada em 1024 páginas de 256 Bytes e memória interna SRAM de 64KBytes (Atmel Corporation, 2009).

Além disso, possui os seguintes periféricos:

- Controlador de memória (MC)
- Controlador de *reset* (RC)
- Gerador de *Clock* (CKGR)
- Controlador para controle de potência (PMC)
- Controlador avançado de interrupções (AIC)
- Unidade de *debug* (DBGU)

- Temporizador de intervalo periódico (PIT)
- *Windowed Watchdog* (WDT)
- Temporizador de tempo real (RTT)
- Duas portas paralelas de I/O controladas (PIO)
- Treze canais controlados para periféricos com Acesso Direto à Memória – *Peripheral DMA Controller* (PDC)
- Uma porta USB 2.0 *Full Speed* (12 Mbits por segundo)
- Interface Ethernet MAC 10/100 base-T
- Controlador CAN
- Controlador serial síncrono (SSC)
- Dois controladores USART – *Universal Synchronous/Asynchronous Receiver Transmitters*
- Dois canais mestre/escravo para comunicação SPI – *Master/Slave Serial Peripheral Interfaces* (SPI)
- Temporizador/Contador – três canais de 16 bits (TC)
- Quatro canais de PWM de 16 bit (PWMC)
- Interface *Two-wire* (TWI)
- Oito canais de conversores A/D de 10 bits
- Pinos de I/O que suportam até 5V, incluindo quatro drives de corrente que suportam até 16mA cada
- Fontes de alimentação
  - – Regulador de 1,8V, que fornece correntes de até 100mA para o núcleo do MCU e para componentes externos
  - – Fonte de alimentação para I/O de 3.3V VDDIO e fonte de alimentação para *flash* de 3.3V VDDFLASH
  - – Fonte de alimentação de 1.8V VDDCORE para o núcleo do MCU com detector de queda de tensão.

O fabricante ainda garante estabilidade de funcionamento até a condição extrema de operação em 55 MHz, alimentado com 1.65V e a 85°C.

Na Figura 2.1, é possível observar o diagrama de blocos do MCU.

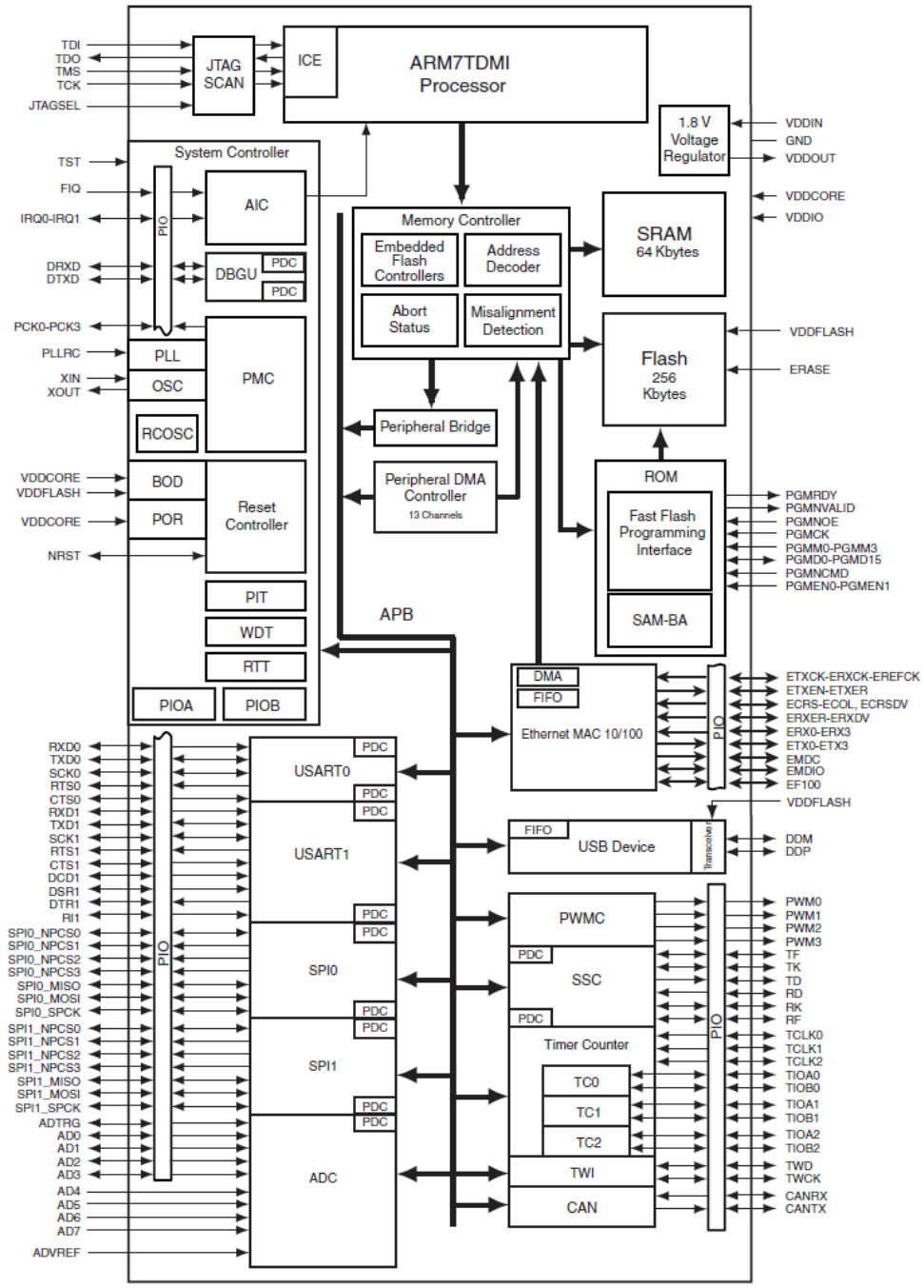



**Figura 2.1 - Diagrama de blocos do MCU AT91SAM7X256**

Na Tabela 2 está presente a pinagem do microcontrolador com encapsulamento do tipo LQFP de 100 pinos mostrado na Figura 2.2 e na Figura 2.3.

**Tabela 2 - Tabela de pinagem do AT91SAM7X256**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | ADVREF | 26 | PA18/PGMD6 | 51 | TDI | 76 | TDO |
| 2 | GND | 27 | PB9 | 52 | GND | 77 | JTAGSEL |
| 3 | AD4 | 28 | PB8 | 53 | PB16 | 78 | TMS |
| 4 | AD5 | 29 | PB14 | 54 | PB4 | 79 | TCK |
| 5 | AD6 | 30 | PB13 | 55 | PA23/PGMD11 | 80 | PA30 |
| 6 | AD7 | 31 | PB6 | 56 | PA24/PGMD12 | 81 | PA0/PGMEN0 |
| 7 | VDDOUT | 32 | GND | 57 | NRST | 82 | PA1/PGMEN1 |
| 8 | VDDIN | 33 | VDDIO | 58 | TST | 83 | GND |
| 9 | PB27/AD0 | 34 | PB5 | 59 | PA25/PGMD13 | 84 | VDDIO |
| 10 | PB28/AD1 | 35 | PB15 | 60 | PA26/PGMD14 | 85 | PA3 |
| 11 | PB29/AD2 | 36 | PB17 | 61 | VDDIO | 86 | PA2 |
| 12 | PB30/AD3 | 37 | VDDCORE | 62 | VDDCORE | 87 | VDDCORE |
| 13 | PA8/PGMM0 | 38 | PB7 | 63 | PB18 | 88 | PA4/PGMNCMD |
| 14 | PA9/PGMM1 | 39 | PB12 | 64 | PB19 | 89 | PA5/PGMRDY |
| 15 | VDDCORE | 40 | PB0 | 65 | PB20 | 90 | PA6/PGMNOE |
| 16 | GND | 41 | PB1 | 66 | PB21 | 91 | PA7/PGMNVALID |
| 17 | VDDIO | 42 | PB2 | 67 | PB22 | 92 | ERASE |
| 18 | PA10/PGMM2 | 43 | PB3 | 68 | GND | 93 | DDM |
| 19 | PA11/PGMM3 | 44 | PB10 | 69 | PB23 | 94 | DDP |
| 20 | PA12/PGMD0 | 45 | PB11 | 70 | PB24 | 95 | VDDFLASH |
| 21 | PA13/PGMD1 | 46 | PA19/PGMD7 | 71 | PB25 | 96 | GND |
| 22 | PA14/PGMD2 | 47 | PA20/PGMD8 | 72 | PB26 | 97 | XIN/PGMCK |
| 23 | PA15/PGMD3 | 48 | VDDIO | 73 | PA27/PGMD15 | 98 | XOUT |
| 24 | PA16/PGMD4 | 49 | PA21/PGMD9 | 74 | PA28 | 99 | PLLRC |
| 25 | PA17/PGMD5 | 50 | PA22/PGMD10 | 75 | PA29 | 100 | VDDPLL |

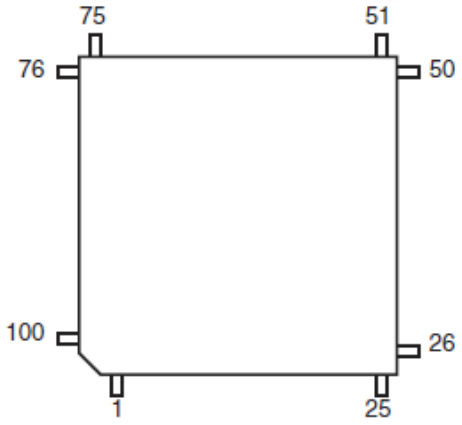



**Figura 2.2 - Esquema de pinagem do AT91SAM7X256**


**Figura 2.3 - Imagem do MCU**

## 2.3. *Pulse Width Modulation* (PWM)

PWM ou Modulação por Largura de Pulso é uma técnica para controlar a potência de um sinal transmitido através de uma série pulsos com ciclo de trabalho (*duty cycle*) variável. O valor de tensão médio fornecido a uma carga é controlado por uma chave eletrônica que liga e desliga rapidamente. Quanto maior o tempo que a chave está ligada, maior é o ciclo de trabalho da onda e maior é a potência fornecida pela fonte de alimentação. Assim, a

potência média fornecida à carga é regulada pelo *duty cycle* (ou ciclo de trabalho) do PWM (SEDRA, et al., 2000).

No AT91SAM7X256 é possível programar tanto o período quanto o *duty cycle* do PWM e é possível determinar a polaridade do sinal de saída, ou seja, ele pode ser iniciado tanto em nível lógico 1 quanto em nível 0 (Figura 2.4).

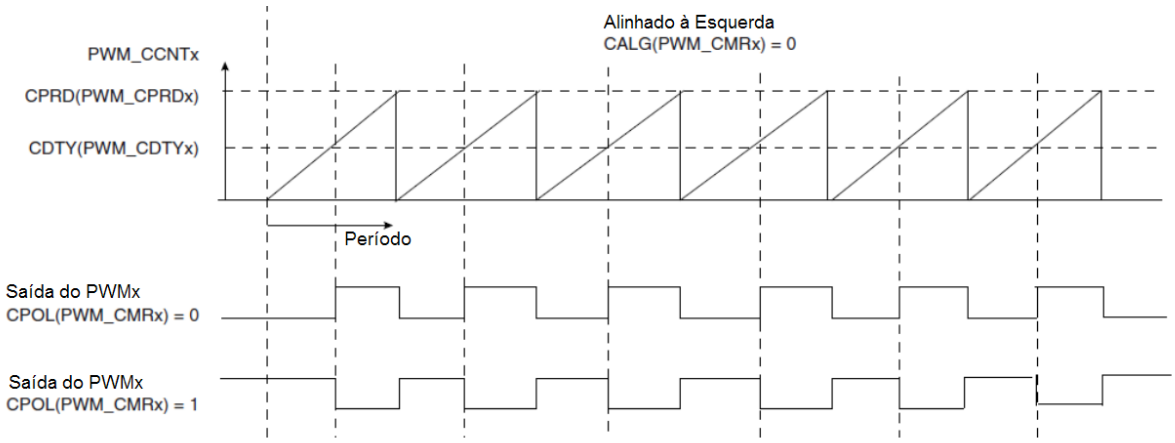



**Figura 2.4 - Ciclos do PWM**

Na Figura 2.5 é possível visualizar o diagrama de blocos da estrutura do PWM presente no MCU.

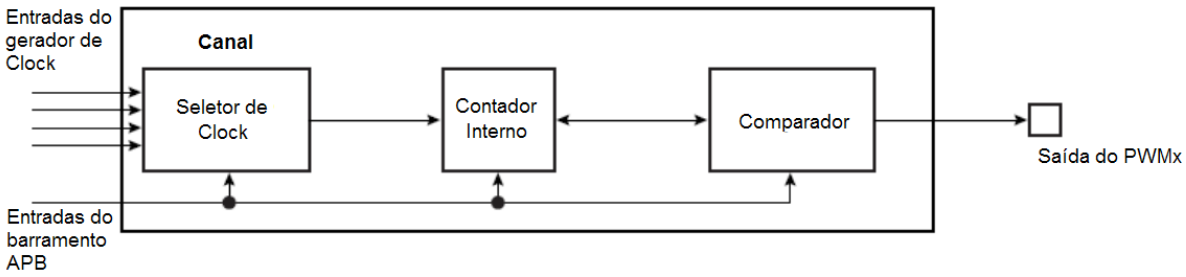



**Figura 2.5 - Diagrama de blocos do PWM do MCU**

## 2.4. Tela de LCD Nokia 6100

A tela de LCD Nokia 6100 é um dos periféricos presentes no *kit*. Um dos critérios para a adoção deste dispositivo é, sem dúvida, o valor de mercado dele. Com a produção em larga escala para os aparelhos celulares Nokia 6100, o custo de um display deste modelo é consideravelmente baixo e é uma ótima escolha para utilização em projetos. É

possível encontrar este LCD por valores inferiores a R$ 30,00, o que é um considerado um ótimo preço para uma tela de LCD colorida.

Este *display* possui como característica resolução de 132 x 132 pixels e reprodução de até 4096 cores, 12 bits de profundidade por pixel – 4 bits para o vermelho, 4 bits para o verde e 4 bits para o azul (LYNCH, 2007).

Mesmo sendo fabricado pela Nokia, o controlador gráfico do LCD também é fabricada por outras duas empresas: Philips e Epson. Identificar o fabricante do controlador é muito importante, pois cada empresa possui algumas características distintas o controle do *display*. A Figura 2.6 ilustra o *display* utilizado no *kit* da Olimex.

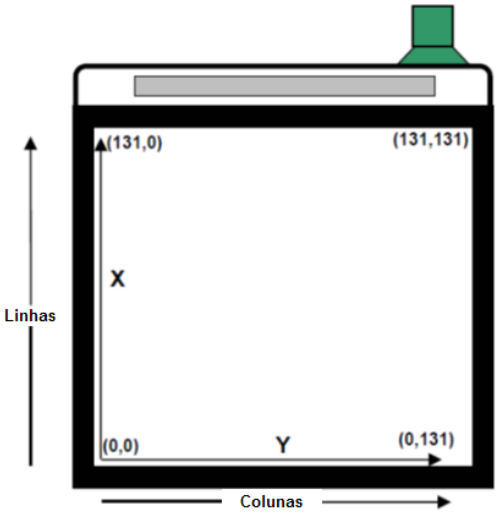



**Figura 2.6 - Ilustração do LCD**

A comunicação entre o LCD e o kit é feita via protocolo SPI serial possuindo dois sinais principais enviados para o mesmo (Figura 2.7):

- SCLK: *clock* para a sincronização com o dispositivo;
- SDIN: sinal unidirecional de dados (a tela é somente-escrita).

Os dados são enviados em pacotes de 9 bits, sendo oito deles de dados e um de controle (explicitando se os dados formam um comando ou uma informação).

Como o processador ARM7 utiliza palavras de 32 bits, os dados referentes ao LCD são armazenados nos 9 bits menos significativos, e os outros vinte e três são perdidos.

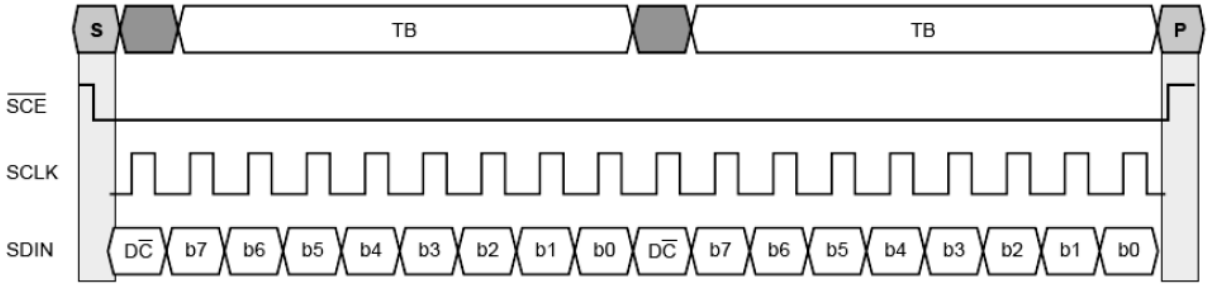



**Figura 2.7 - Forma de onda para a comunicação SPI**

O controlador gráfico Epson S1D15G00, utilizado para o projeto, possui dois modos diferentes para especificar as componentes RGB de cada *pixel*, selecionadas por programação:

- 12 bits por *pixel*:

Cada componente do sistema RGB é definido por 4 bits.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| R | R | R | R | G | G | G | G |
| B | B | B | B | R | R | R | R |
| G | G | G | G | B | B | B | B |

Comando RAMWR (escrita na memória)

Dado: Vermelho e verde do 1º *pixel*

Dado: Azul do 1º *pixel*; Vermelho do 2º *pixel*

Dado: Verde e azul do 2º *pixel*

- 8 bits por pixel:

As componentes R e G possuem 3 bits cada, e a componente B é definida em 2 bits.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| R | R | R | G | G | G | B | B |

Comando RAMWR (escrita na memória)

Dado: Vermelho, verde e azul – 8 bits por *pixel*

## 2.5. A Comunicação SPI

A comunicação SPI (*Serial Peripheral Interface*) é um *link* de dados síncrono que possibilita a comunicação entre dispositivos. A interface SPI é essencialmente um registrador que transmite serialmente dados para outras interfaces SPI. Durante a transferência de dados, um sistema SPI se comporta como Mestre e pode transmitir dados para outros sistemas SPI, denominados Escravos.

Um sistema SPI consiste de quatro linhas de dados principais:

a) *Master Out Slave In* (MOSI): linha de dado que interconecta a saída do registrador Mestre e a(s) entrada(s) do(s) registrador(es) Escravo(s).
b) *Master In Slave Out* (MISO): linha de dado que interconecta a entrada do registrador Mestre e a(s) saída(s) do(s) registrador(es) Escravo(s).
c) *Serial Clock* (SPCK): linha de controle gerada pelo Mestre e dirigida até o Escravo a fim de regular a transmissão dos bits.
d) *Slave Select* (NSS): linha de controle que permite a ativação (ou não) por hardware dos Escravos.

Na Figura 2.8 a seguir é visualizada a distribuição dos pinos entre os dispositivos mestre e escravo, no caso, três escravos.

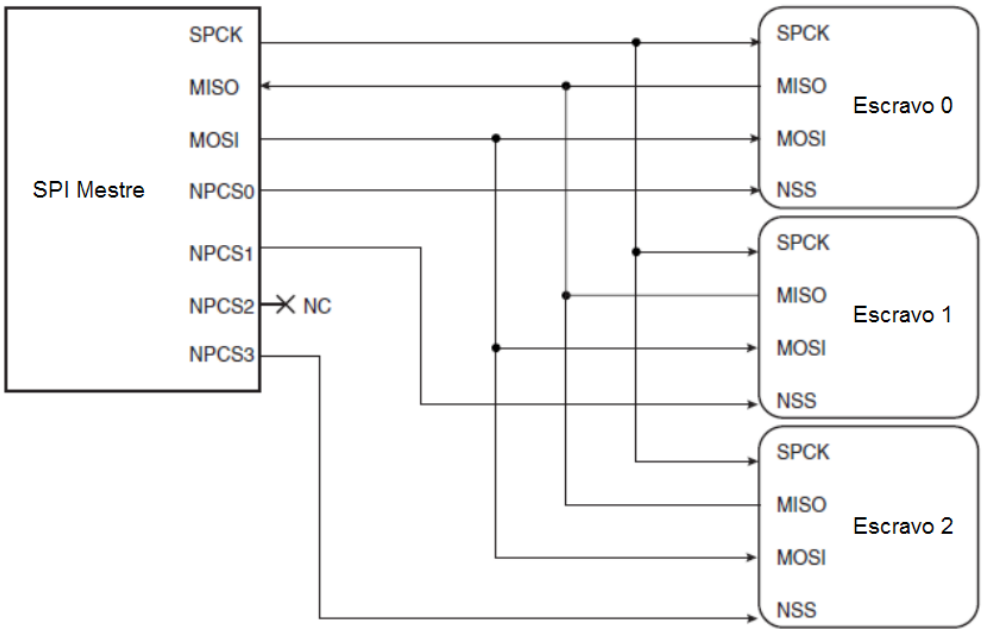



**Figura 2.8 - Esquema de ligação entre dispositivos Mestre e Escravo na comunicação SPI**

Para se trabalhar com a comunicação SPI, primeiramente existem três características principais a serem analisadas e programadas conforme a necessidade do usuário:

- Linhas de I/O: Os pinos utilizados podem ser multiplexados com as linhas das portas paralelas de I/O. O programador deve primeiro programar estas linhas para serem utilizadas como função SPI.

- Gerenciamento de Energia: O SPI pode ser alimentado pelo controlador PMC (*Power Management Controller*), assim este deve ser configurado antes de habilitar o *clock* SPI.
- Interrupção: A Interface SPI tem uma linha de interrupção conectada ao AIC (*Advanced Interrupt Controller*). Assim, para gerenciar a interrupção SPI exige-se a programação do AIC antes de utilizar a SPI.

Em se tratando de transferência de dados na comunicação SPI deve-se levar em conta a polaridade e as fases disponíveis na transmissão de dados. O clock, por exemplo, tem sua polaridade programada pelo bit CPOL e sua fase programada pelo bit NCPHA. Como cada um desses dois parâmetros possui dois possíveis estados, ao todo acabam existindo quatro combinações, assim, para cada mestre/escravo uma combinação deve ser adotada.

Analisando a descrição funcional de um dispositivo SPI, observam-se dois tipos distintos de operação: modo mestre e modo escravo. De forma resumida a operação em modo mestre é programado escrevendo '1' no bit MSTR no registrador *Mode Register*. Os pinos NPCS0 a NPCS3 são todos configurados como saída e os pinos MISO torna-se entrada ao passo que MOSI torna-se saída do transmissor. Caso contrário, se o bit MSTR é escrito como '0' então o dispositivo fica em modo escravo.

### 2.5.1. Operação no Modo Mestre (*Master*)

- *Clock* operado pelo gerador de *baud rate*, este controla os dados transferidos do dispositivo escravo conectado no SPI *bus*.
- A transferência inicia quando o controlador escreve algum dado no SPI_TDR (*Transmit Data Register*), dado este transferido para o *Shift Register* (ShiftR), ação indicada pelo bit TDRE (*Transmit Data Register Empty*) no Registrador *Stats Register* (SPI_SR), e alocado na linha MOSI.
- O bit TDRE indica também se um novo dado pode ser carregado no SPI_TDR, caso isso ocorra durante uma transmissão, o dado, obrigatoriamente, se mantém em espera até o encerramento da mesma.
- O final de uma transferência é indicado pela *flag* TXEMPTY.
- A transferência de um dado a partir do *Receive Data Register* (SPI_RDR) para o SPI_RDR é indicado pelo bit RDRF (*Receive Data Register Full*) no registrador SPI_SR, bit é zerado quando o dado recebido é lido.

Na Figura 2.9 pode ser visto o diagrama de bloco quando um dispositivo atua no modo mestre.

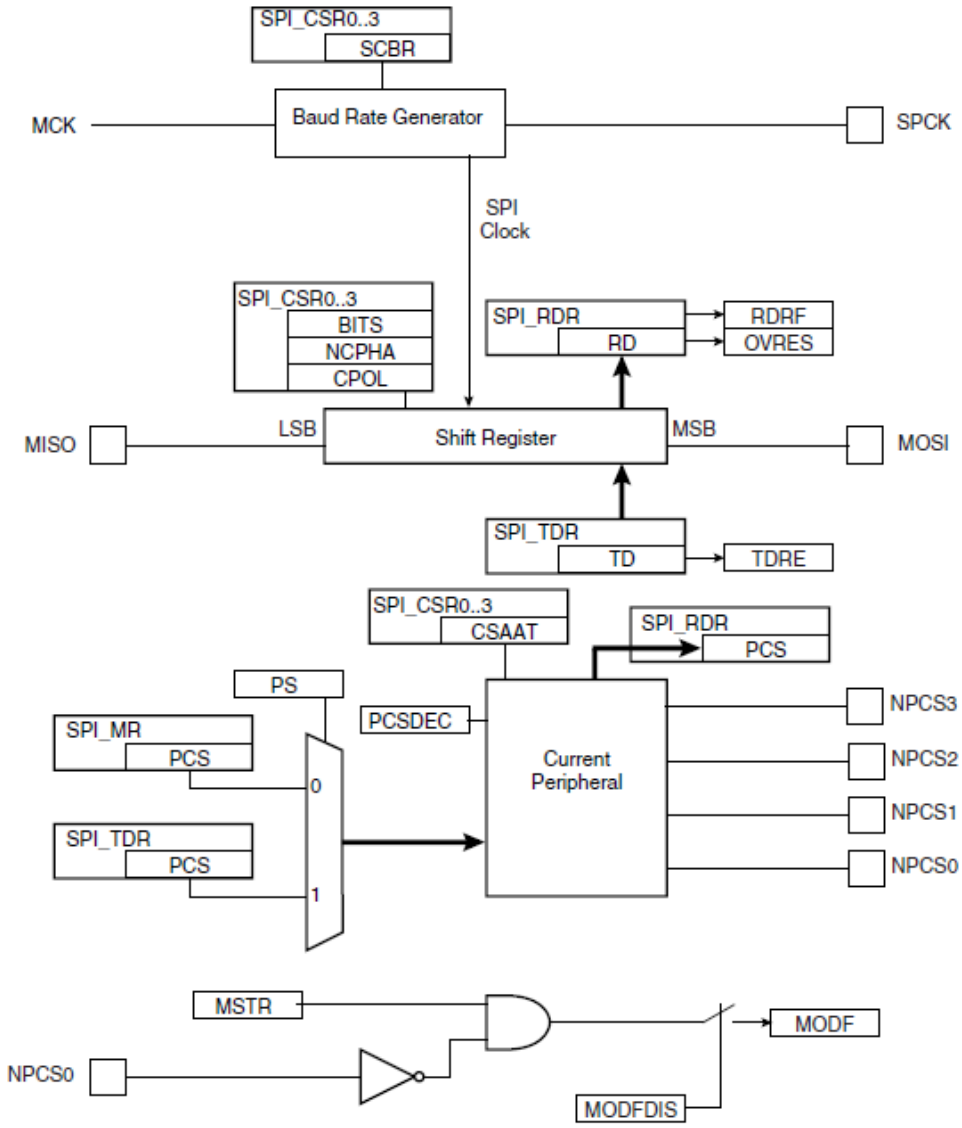



**Figura 2.9 - Diagrama de blocos na operação em modo Mestre**

### 2.5.2. Operação no Modo Escravo (*Slave*)

- Os bits dos dados são processados de acordo com o *clock* fornecido pelo pino SPCK.

- O pino NSS precisa ser definido em 0 para ocorrer a seleção do dispositivo e o reconhecimento do *clock* vindo do mestre, processando o número de bits definidos pelo campo BITS do *Chip Select Register* (SPI_CSR0).
- Após os bits terem sido processados, o dado recebido é transmitido para o SPI_TRD e o bit RDRF é colocado em 1.
- Os registrados SPI_TDR, SPI_RDR e o Shift Register tem atuação semelhante ao que ocorre no modo mestre.

Na Figura 2.10 pode ser visto o diagrama de blocos quando o dispositivo atua no modo escravo.

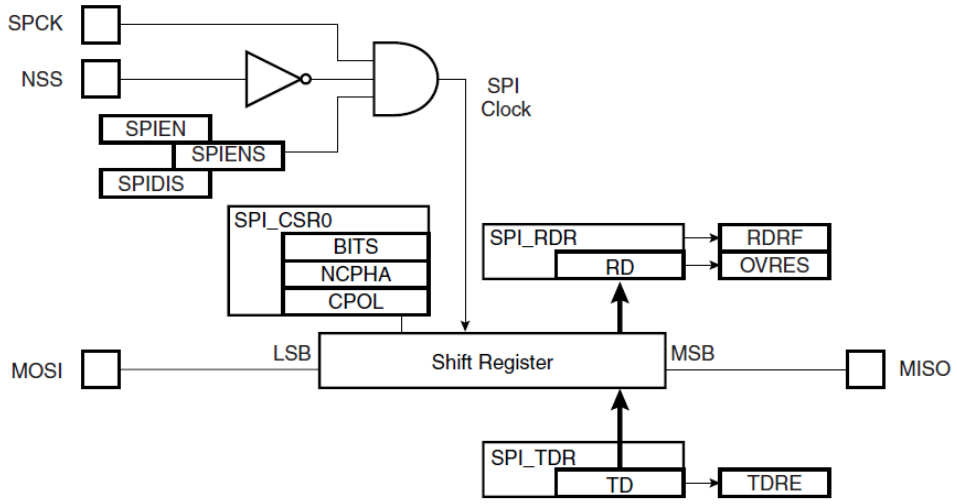



**Figura 2.10 - Diagrama de blocos em modo Escravo**

Na Tabela 3 são apresentados os registradores que fazem a interface da comunicação SPI, porém suas descrições completas podem ser encontradas no manual do microcontrolador.

**Tabela 3 - Tabela de registradores do SPI**

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Control Register | SPI_CR | Write-only | --- |
| 0x04 | Mode Register | SPI_MR | Read-write | 0x0 |
| 0x08 | Receive Data Register | SPI_RDR | Read-only | 0x0 |
| 0x0C | Transmit Data Register | SPI_TDR | Write-only | --- |
| 0x10 | Status Register | SPI_SR | Read-only | 0x000000F0 |
| 0x14 | Interrupt Enable Register | SPI_IER | Write-only | --- |
| 0x18 | Interrupt Disable Register | SPI_IDR | Write-only | --- |
| 0x1C | Interrupt Mask Register | SPI_IMR | Read-only | 0x0 |
| 0x20 - 0x2C | Reserved | | | |
| 0x30 | Chip Select Register 0 | SPI_CSR0 | Read-write | 0x0 |
| 0x34 | Chip Select Register 1 | SPI_CSR1 | Read-write | 0x0 |
| 0x38 | Chip Select Register 2 | SPI_CSR2 | Read-write | 0x0 |
| 0x3C | Chip Select Register 3 | SPI_CSR3 | Read-write | 0x0 |
| 0x004C - 0x00F8 | Reserved | – | – | – |
| 0x004C - 0x00FC | Reserved | – | – | – |
| 0x100 - 0x124 | Reserved for the PDC | | | |

# Capítulo 3

## 3. Materiais e Métodos

### 3.1. Concepção do Projeto

O desenvolvimento deste projeto baseou-se no sintetizador musical Tenori-on, produzido pela Yamaha, mais especificamente no seu modo de varredura sequencial.

O Tenori-on possui um teclado de 16x16 botões com LED. No modo de varredura, a matriz tem suas colunas “lidas” de forma sequencial e contínua. A reprodução das notas ou dos acordes é relacionada com os botões pressionados da coluna no instante da leitura. Além disso, cada linha da matriz representa uma nota musical, ou uma frequência de onda distinta, sendo que as linhas superiores representam as notas mais agudas, enquanto que as linhas inferiores representam as notas mais graves (Figura 3.1).

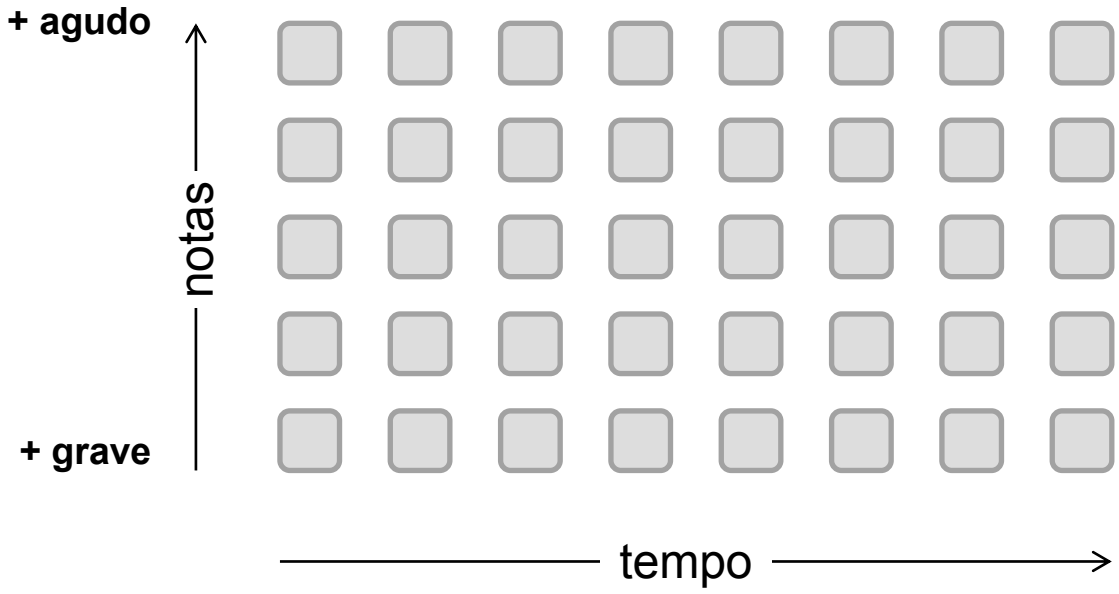



**Figura 3.1 - Esquema do teclado matricial**

No projeto desenvolvido, para representar o teclado confeccionou-se um circuito contendo uma matriz de 5x8 botões com LED para indicar se o botão está ou não pressionado.

Este teclado é alimentado pelo próprio kit de desenvolvimento, com o qual se comunica por meio das portas paralelas de I/O. No kit, são utilizados um potenciômetro para regular a velocidade de varredura do teclado e um *joystick* que altera a frequência da onda produzida ao ser posicionado para cima ou para baixo, além do alto-falante (ou da saída de fone) para a reprodução do áudio e a tela LCD para ilustrar a imagem de fundo.

O MCU AT91SAM7X256 é responsável por gerenciar e se comunicar com os periféricos, conduzir as rotinas de controle do programa e de gerar o sinal de áudio para o alto-falante.

## 3.2. A Geração do Sinal de Áudio

A geração do sinal de áudio é feita pelo próprio microcontrolador, através do acesso a uma tabela de valores salva na própria memória do MCU, que contém os pontos dos sinais senoidais equivalentes a cada nota musical. Quando há uma combinação de notas, formando um “acorde”, basta realizar uma operação de soma com cada onda senoidal, sendo assim possível obter a forma de onda resultante.

Para a escolha das notas, definiu-se a mesma escala utilizada pelo Tenori-on – a escala pentatônica maior. Em uma oitava, seguindo uma escala maior iniciando em Dó, por exemplo, temos a sequência de notas e as frequências equivalentes (CDCC USP) apresentadas na Tabela 4:

**Tabela 4 - Tabela de notas e relação de frequências**

| Nota | Dó – I | Ré – II | Mi – III | Fá – IV | Sol – V | Lá – VI | Si – VII |
|---|---|---|---|---|---|---|---|
| Frequência | f | 9f/8 | 5f/4 | 4f/3 | 3f/2 | 5f/3 | 15f/8 |

A escala pentatônica maior é formada pelas notas I-II-IV-V-VI. Dessa forma, a relação de frequência entre elas é dada por f – 9f/8 – 4f/3 – 3f/2 – 5f/3.

Temos que a equação de onda no tempo contínuo é dada por (Oppenheim, et al., 1975):

$$u(T) = A \cdot \sin(2\pi f \cdot T)$$

Para se obter uma boa resolução da onda, definiu-se uma amostragem de 100 pontos por período T da onda, e os pontos começaram a ser definidos através das seguintes equações:

$$u_I(k) = \sin\left(2\pi \cdot \frac{k}{100}\right)$$

$$u_{II}(k) = \sin\left(2\pi \frac{9}{8} \cdot \frac{k}{100}\right)$$

$$u_{IV}(k) = \sin\left(2\pi \frac{4}{3} \cdot \frac{k}{100}\right)$$

$$u_V(k) = \sin\left(2\pi \frac{3}{2} \cdot \frac{k}{100}\right)$$

$$u_{VI}(k) = \sin\left(2\pi \frac{5}{3} \cdot \frac{k}{100}\right)$$

Com essas equações obtêm-se os valores entre -1 e 1 dos pontos da senóide equivalente, como pode ser observado na Figura 3.2.

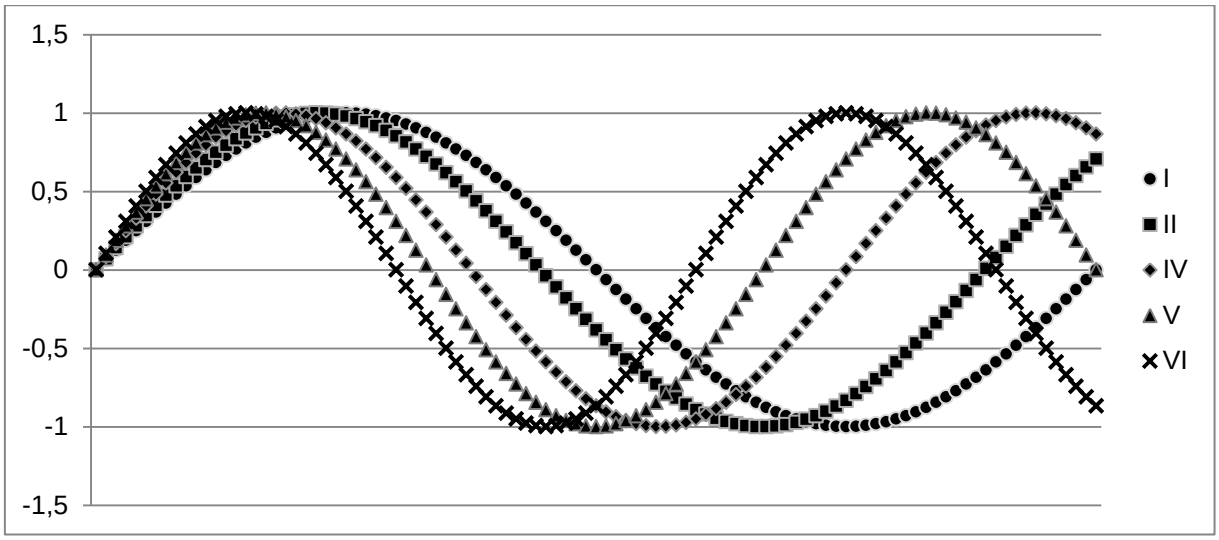



**Figura 3.2 - Representação das ondas referentes a cada nota musical da escala pentatônica**

Observa-se que com 100 pontos para cada senóide temos um período exato somente para o sinal I (equivalente à primeira nota da escala), porém os outros sinais, como possuem frequência maior, apresentam frações a mais de um período.

Armazenar os pontos dessa forma causaria problemas na continuidade das notas que não fossem a I (primeira nota da escala) em uma reprodução sequencial, como pode ser observado na Figura 3.3, o que ocasionaria distorção no sinal sonoro, não representando a nota desejada.

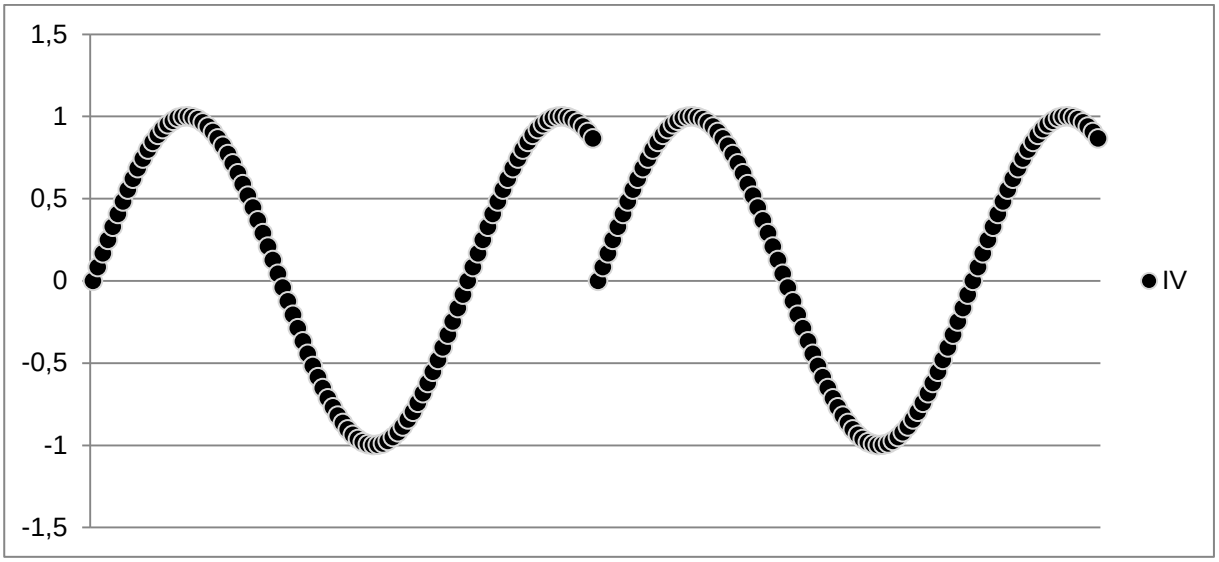



**Figura 3.3 - Falta de continuidade da onda ao armazenar apenas 100 pontos**

Para obter continuidade para todas as ondas foi preciso determinar o mínimo múltiplo comum entre as cinco frequências. É possível observar que o valor 24 é o denominador comum entre todas as notas. Com isso, após 24 períodos de onda da nota I, ou 27 períodos de onda da nota II, ou 30 períodos de onda da nota IV, e assim sucessivamente, temos o encontro de todos os sinais no mesmo ponto, garantindo, assim a continuidade de todas as notas, como é possível observar na Figura 3.4.

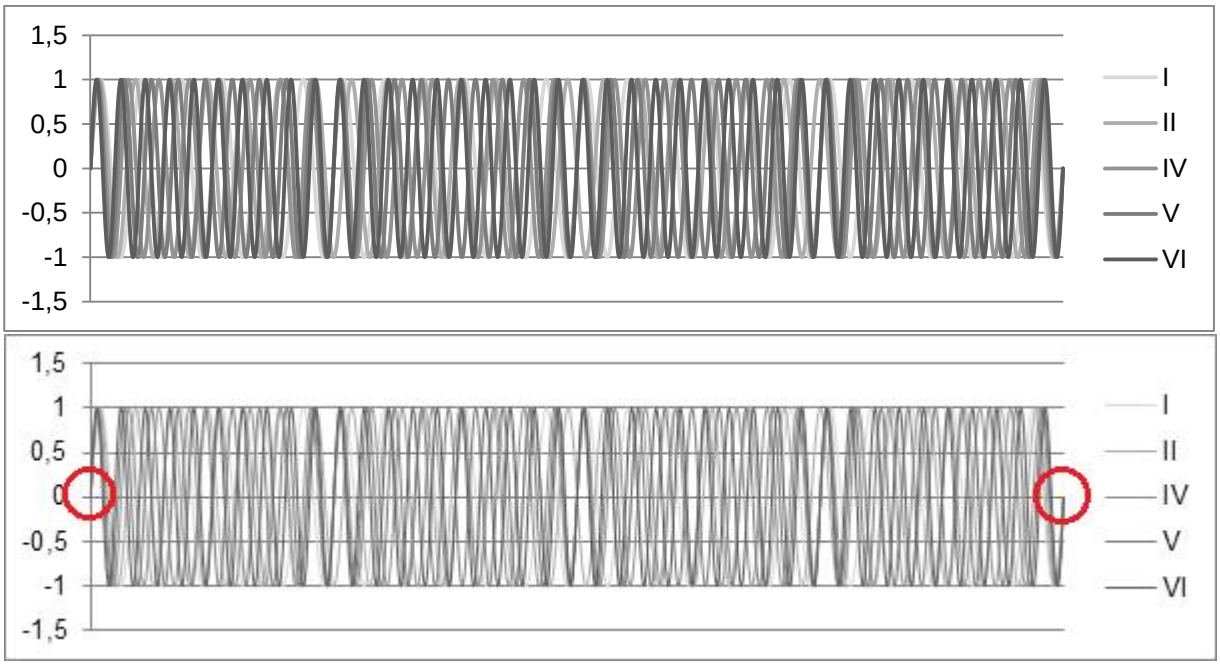



**Figura 3.4 - Sequência de 24 períodos de onda da frequência fundamental**

Outro detalhe a ser considerado antes de gerar a tabela com os valores das notas é que, pela configuração feita do PWM, os valores da onda devem estar em um intervalo entre 0 e 1023. Por tanto, o limite do sinal resultante de qualquer combinação de ondas não pode ultrapassar este valor. Assim, foi verificado qual é o valor máximo dos pontos do sinal para o caso extremo em que todas as cinco ondas estão somadas. Com os sinais limitados entre -1 e 1 o ponto mínimo e máximo são respectivamente -4,8 e 4,8. Transladando o sinal para que ele fique sempre maior que zero, o ponto máximo se torna 9,6. Dividindo-se 1023 por 9,6 obtém-se um valor igual a 106,5. Este é o número que corresponde ao coeficiente máximo que se pode multiplicar os pontos das ondas para que elas nunca extrapolem o limite de saturação imposto pelo PWM.

No projeto, os valores foram multiplicados por 100, fazendo com que os pontos das senóide ficassem em um intervalo entre -100 e 100. Esses valores foram armazenados arredondados (na forma de números inteiros) em uma matriz declarada ***const int*** com dimensão [5][2400] e, inclusive, com valores negativos. O processo de deixar o valor do sinal resultante no intervalo entre 0 e 1023 se dá nas rotinas do software programado.

## 3.3. O kit de Desenvolvimento da Olimex – SAM7-EX256

Como já foi mencionado anteriormente, o projeto foi desenvolvido utilizando-se alguns dos periféricos oferecidos pelo kit produzido pela Olimex – SAM7-EX256. Além do microcontrolador ARM7 AT91SAM7X256, a seguir são listadas algumas de suas características:

- Conector padrão JTAG com ARM de 2x10 pinos para programação e *debug*;
- *Display* de LCD colorido NOKIA 6610 128x128 TFT 12 bit com *back light*;
- Ethernet 10/100 PHY com KS8721BL.
- Conector USB;
- Duas interfaces para RS232;
- Conector para cartão SD/MMC;
- *Joystick* com 4 direções mais função de pressionar;
- Dois botões;
- Entrada e saída de áudio com conectores para microfone e alto-falante/fone de ouvidos;
- Alto-falante incluído na própria placa com potenciômetro para controle de volume;
- Potenciômetro conectado ao ADC.

- Termistor conectado ao ADC;
- Regulador de tensão de 3,3V e limite de corrente de 800mA;
- Entrada para fonte de alimentação de 6V AC ou 9V DC;
- LED para a fonte de alimentação;
- Filtro capacitivo para a fonte de alimentação;
- Circuito e botão de *RESET*;
- Cristal de 18,432 MHz;
- Pinos de acesso a portas do microcontrolador;

No desenvolvimento do projeto, utilizou-se, dentre os periféricos disponíveis no kit, a tela de LCD, o *joystick*, os botões, o potenciômetro conectado ao ADC, o alto-falante, a saída de áudio e os pinos de acesso às portas I/O do microcontrolador.

Na Figura 3.5 e na Figura 3.6 é possível visualizar as imagens do kit e do seu esquemático do circuito, com destaque nos periféricos utilizados.

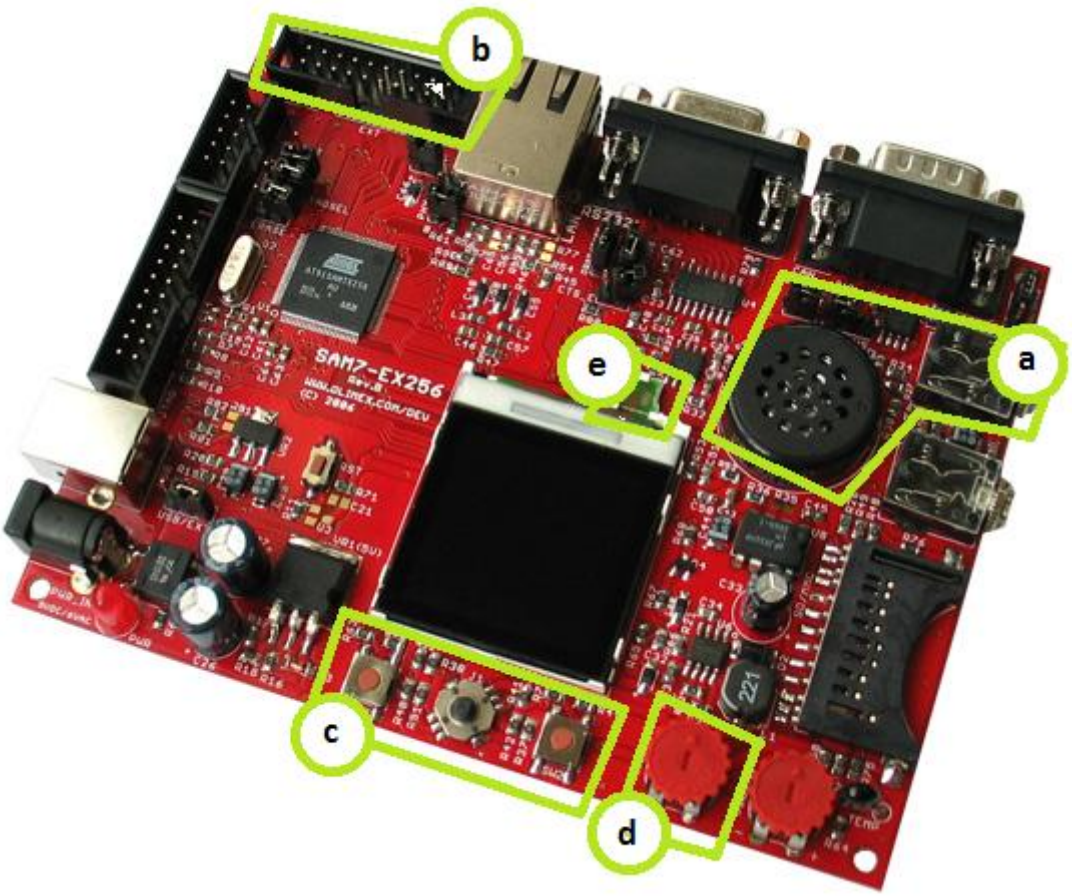



**Figura 3.5 - Kit SAM7-EX256**

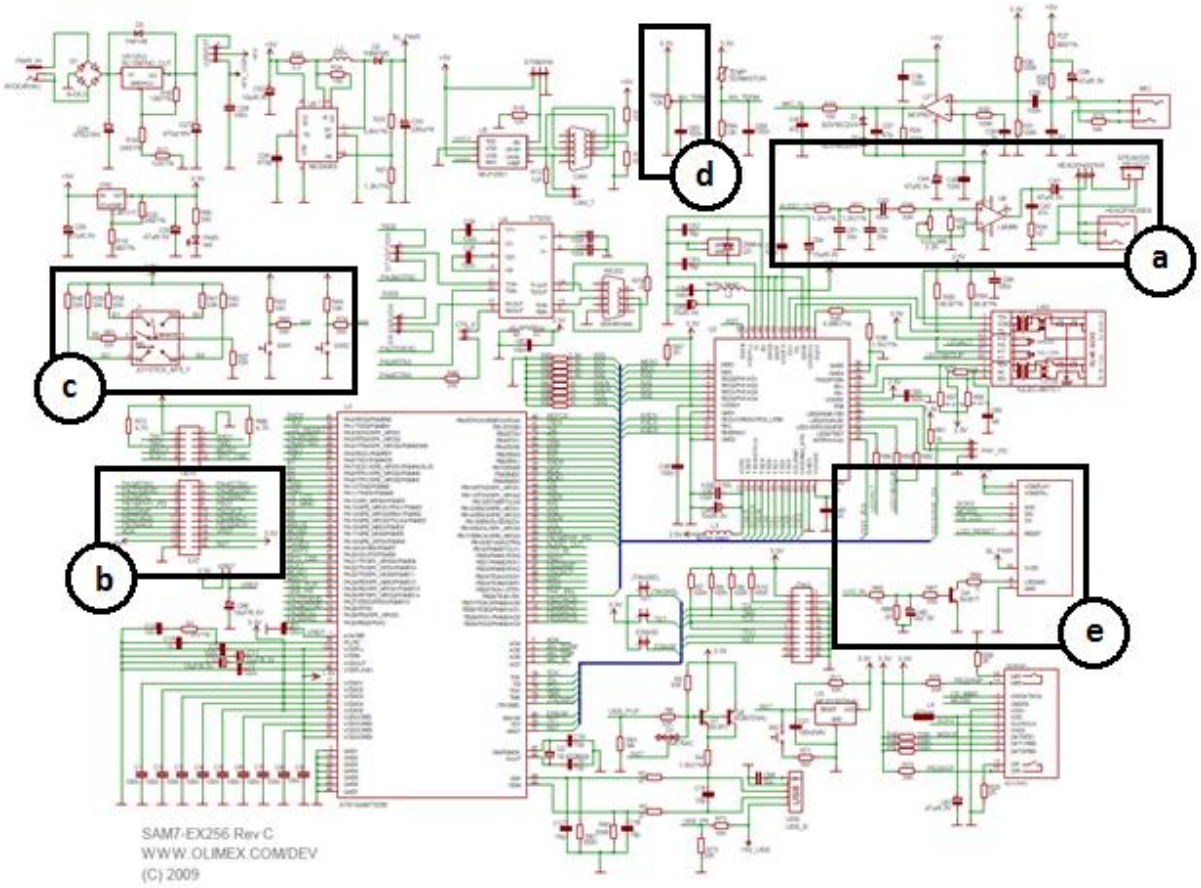



**Figura 3.6 - Esquemático do kit SAM7-EX256**

**a)** A interface para saída de áudio:

A saída do PWM está ligada à entrada do amplificador de áudio, como pode ser visto na Figura 3.7, através do pino denominado AUDIO_OUT. É possível observar neste circuito a presença de um potenciômetro para o controle de volume e duas saídas, um HEADPHONE e outro SPEAKER. A escolha por qual saída utilizar é feita através do *jumper* denominado como HEADPH/SPKR.

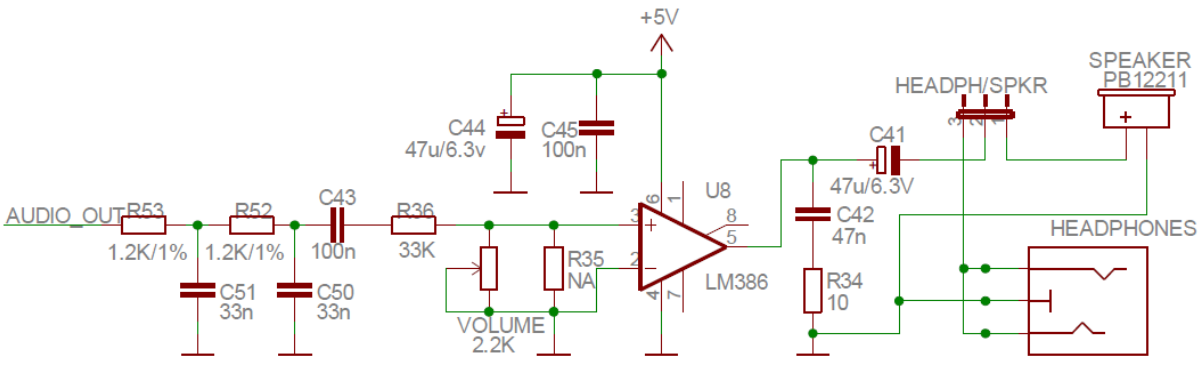



**Figura 3.7 - Interface de saída de áudio**

**b)** O conector de 20 pinos para interface com as portas paralelas I/O:

Na Figura 3.8 é possível visualizar os pinos do AT91SAM7X256 que podem ser acessados por periféricos externos, além dos pinos que podem ser utilizados como fonte de alimentação. Para a interface com o teclado confeccionado, utilizou-se os pinos 3, 4 e 5 (PA27, PA28 e PA29) como output do MCU, os pinos 8, 11, 12, 13 e 14 (PB21, PB27, PB28, PB29 e PB30) como *input* do MCU e os pinos 17 e 19 (+5V e GND) para a alimentação do teclado. Mais adiante será explicado com mais detalhes as funções de cada pino da porta paralela utilizado.

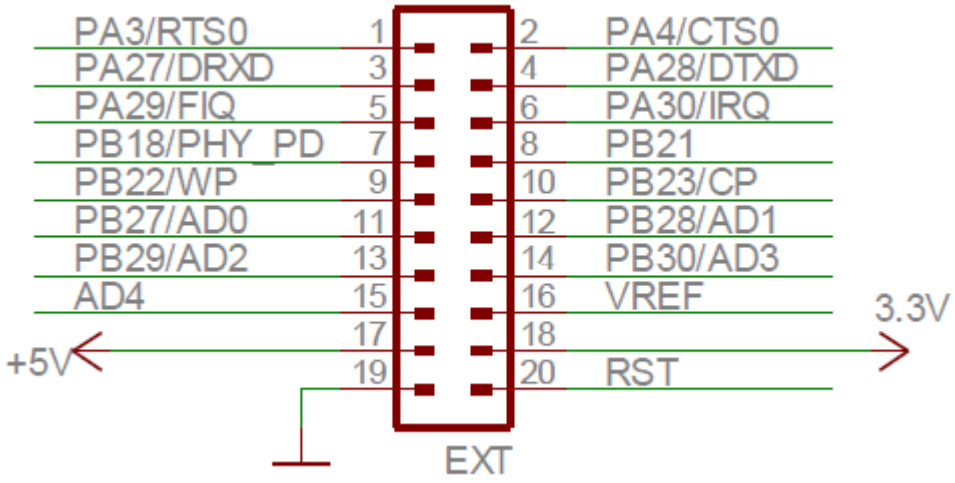



**Figura 3.8 - Conector de 20 pinos para interligação de periféricos externos com o MCU**

**c)** *Joystick* e botões

Na Figura 3.9 é possível observar o *joystick* e botões liga/desliga (*switch buttons*). Nota-se que cada posição do *joystick* e que cada botão está ligado a uma porta distinta do MCU e a ativação se dá em nível baixo, ou seja, se os botões não estão pressionados o MCU lê a entrada em nível alto.

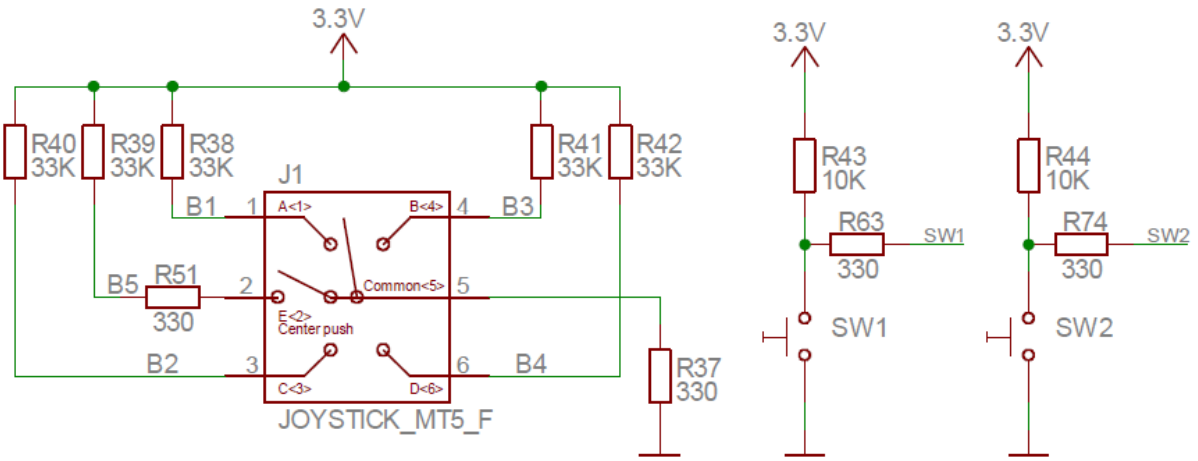



**Figura 3.9 - *Joystick* com cinco posições (UP-DOWN-LEFT-RIGHT-PUSH) e dois switch buttons**

**d)** Potenciômetro conectado ao conversor AD do MCU

Na Figura 3.10 é possível observar o circuito simples responsável por gerar a entrada do conversor AD do AT91SAM7X256. É simplesmente um potenciômetro (*Trimpot*) que varia a sua resistência, fazendo com que o sinal no pino NA_TRIM varie entre 0 e 3,3V. O capacitor serve como filtro, para evitar ruídos de alta frequência.

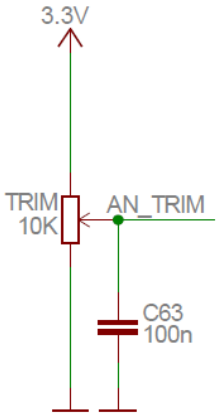



**Figura 3.10 - Potenciômetro conectado à entrada AD do MCU**

**e)** Interface com o controlador gráfico do LCD

Na Figura 3.11 tem-se o esquemático da interface do MCU com o *display* de LCD. É possível visualizar os pinos SCK0, MOSI0 e CS_LCD, responsáveis pela comunicação SPI entre o microcontrolador (mestre) e o LCD (escravo).

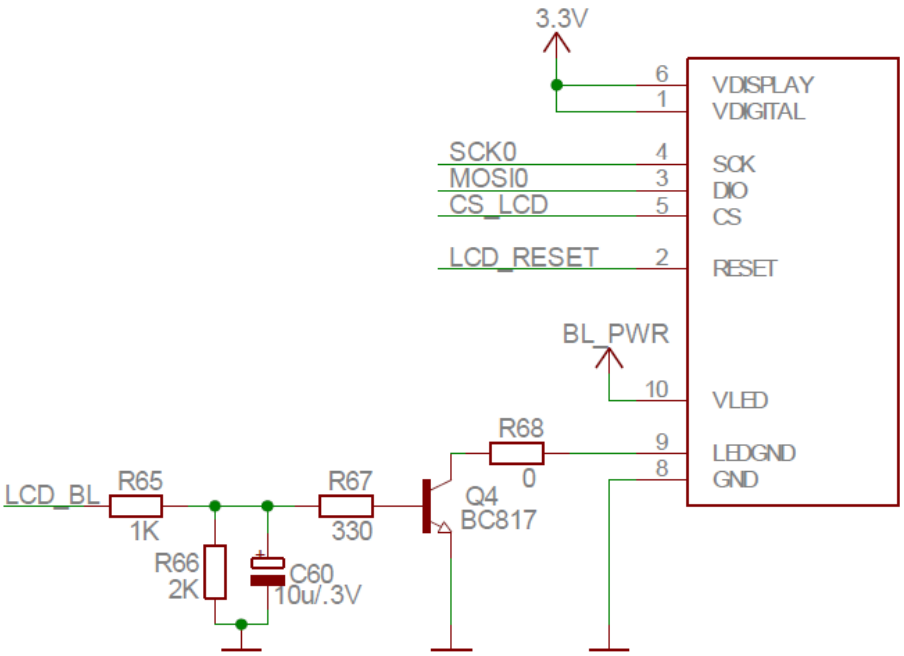



**Figura 3.11 - Interface com o controlador do LCD**

## 3.4. O Teclado para Interface com o Usuário

Foi projetado um teclado composto por uma matriz de botões para reproduzir as notas musicais desejadas pelo usuário.

O protótipo executado é composto por cinco linhas, cada uma representando uma nota musical, e por oito colunas, totalizando quarenta botões com LEDs associados a cada um deles, que permanecem acesos quando o botão correspondente é acionado, e mais um LED por coluna para indicação da leitura do MCU. Ou seja, quando o programa está lendo uma determinada coluna o LED associado a ela permanece aceso, para facilitar a visualização do usuário.

### 3.4.1. Projeto do Circuito

Para fazer a seleção das colunas e a leitura dos botões pressionados, utilizou-se multiplexação para cada coluna, através de três bits de controle, podendo, dessa forma, associar cada uma das oito colunas com uma posição específica, ou seja, 000 sendo a

primeira coluna, 001 a segunda, 010 a terceira e assim sucessivamente. Para realizar esta tarefa, utilizou-se o multiplexador 74151 (Figura 3.12).

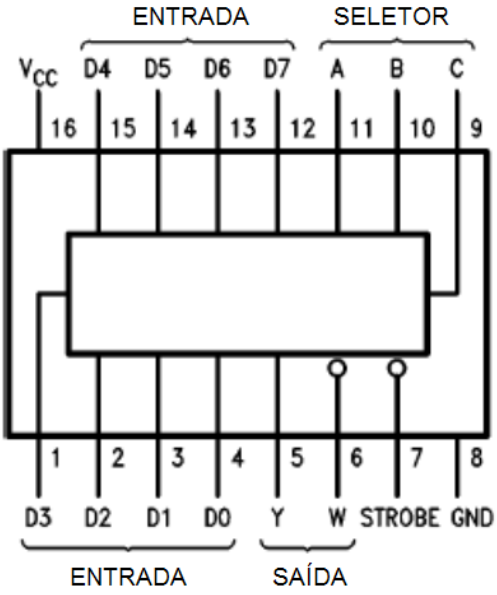



**Figura 3.12 - Multiplexador 74151**

Com um multiplexador (MUX) é possível selecionar e fazer a leitura do valor de um botão específico para cada das oito colunas. Como existem cinco linhas, ou cinco botões por coluna, utilizaram-se cinco MUX, cada um responsável por uma nota. Dessa forma, ao enviar um sinal de controle, todos os MUX estarão selecionando a mesma coluna, no entanto cada um estará conectado a um botão específico. A saída de cada MUX é conectada a um pino específico de I/O do MCU para que seja feita a sua leitura.

A Figura 3.13 mostra um esquemático simplificado do circuito apresentando a ligação de apenas uma coluna nos MUX. Observa-se que para o funcionamento do projeto, deseja-se que os LEDs estejam acesos quando o botão está pressionado, portanto com a chave aberta os LEDs estão desligados e o nível lógico na entrada do MUX é zero. No caso contrário, com a chave fechada, os LEDs se acendem e o nível lógico na entrada do MUX é 1. Temos, ainda, que o 74151 possui tanto uma saída normal como uma saída invertida. Optou-se por se utilizar a lógica positiva do 74151 (saída Y – pino 5) no desenvolvimento do projeto (National Semiconductor, 1989).

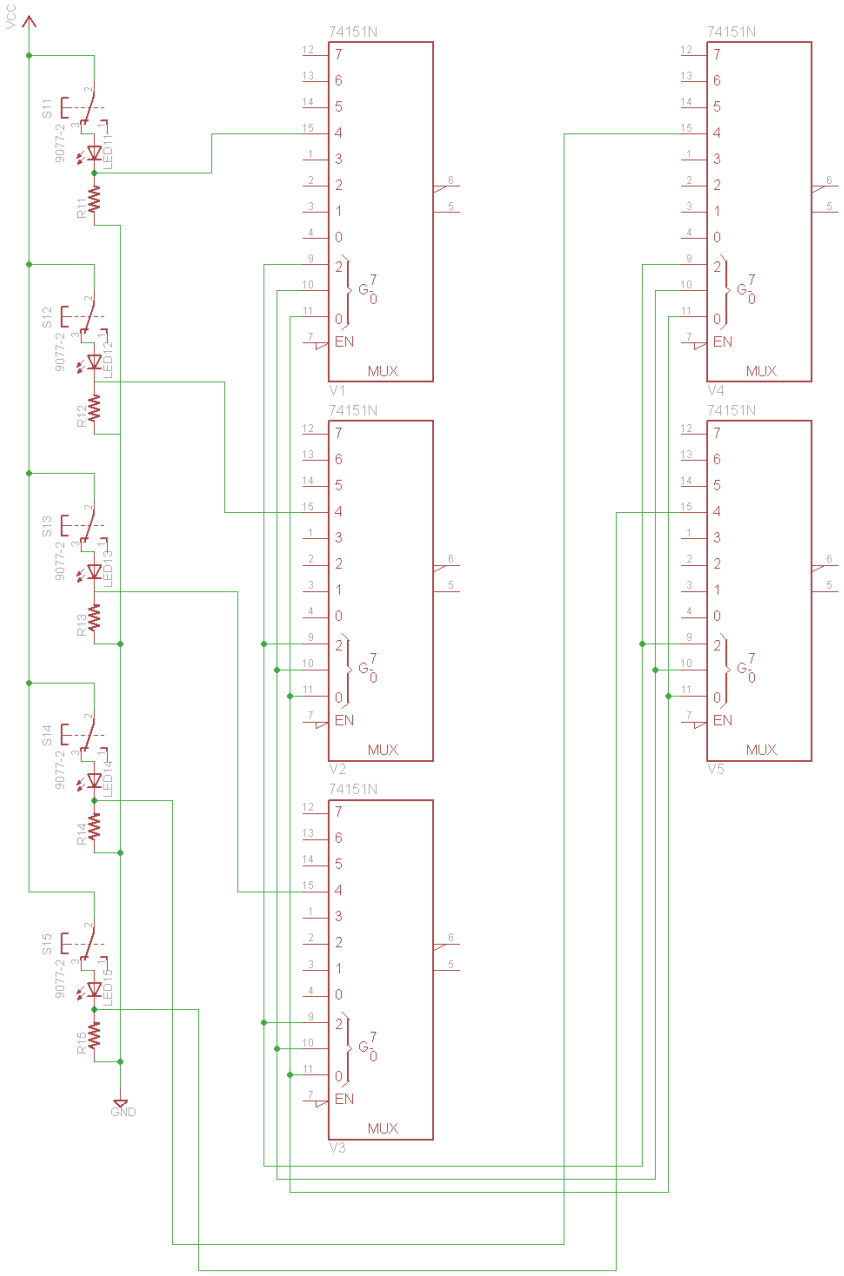



**Figura 3.13 - Esquemático representando a ligação de uma coluna nos multiplexadores**

Para cada coluna há, também, um LED para indicar o instante da leitura. Para esse circuito utilizou-se o CI 74138, que é um decodificador 3 para 8 (Figura 3.14) . Utiliza-se o mesmo código que chega à entrada de controle dos multiplexadores como selecionador de entrada do decodificador. Abaixo, pode-se observar, na Tabela 5, a tabela verdade do 74138 (Fairchild Semiconductor, 2000).

**Tabela 5 - Tabela verdade do decodificador 74138**

| Entradas | | | | | Saídas | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hab. | | Seletor | | | | | | | | | | |
| G1 | G2 | C | B | A | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
| X | H | X | X | X | H | H | H | H | H | H | H | H |
| L | X | X | X | X | H | H | H | H | H | H | H | H |
| H | L | L | L | L | **L** | H | H | H | H | H | H | H |
| H | L | L | L | H | H | **L** | H | H | H | H | H | H |
| H | L | L | H | L | H | H | **L** | H | H | H | H | H |
| H | L | L | H | H | H | H | H | **L** | H | H | H | H |
| H | L | H | L | L | H | H | H | H | **L** | H | H | H |
| H | L | H | L | H | H | H | H | H | H | **L** | H | H |
| H | L | H | H | L | H | H | H | H | H | H | **L** | H |
| H | L | H | H | H | H | H | H | H | H | H | H | **L** |

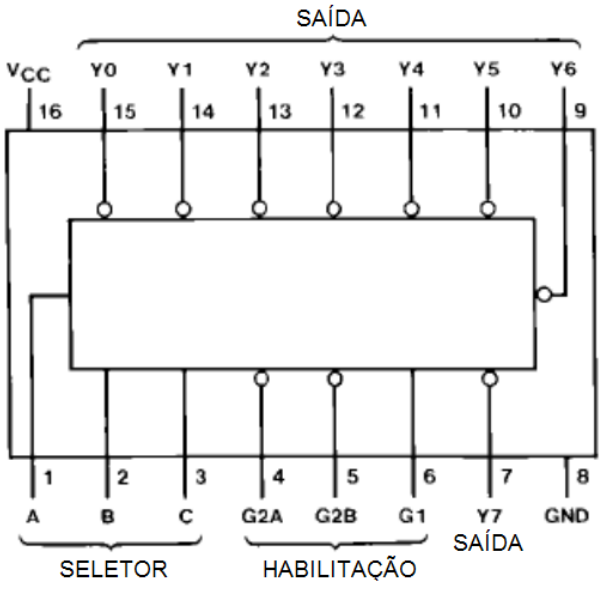



**Figura 3.14 - Decodificador 74138**

Observa-se que este CI possui sinal de saída ativo em *Low*. Para acionar o LED com este sinal, utilizou-se a saída do 74138 ligado à base de um transistor BJT PNP, funcionando como chave, como está ilustrado na Figura 3.15. Dessa forma, quando o BJT tem sua base alimentada pelo sinal *Low* (ativo) do decodificador, ele satura e entra em

condução, acendendo o LED. Quando ocorre o contrário, ou seja, a base do BJT é alimentado com o sinal alto (inativo) do decodificador, ele entra em corte e, portanto, o LED se mantém apagado.

A utilização de um transistor se faz necessária pelo fato de o CI não suportar em sua saída a corrente exigida pelo LED. A conexão do LED diretamente na porta de saída do 74138 causaria a danificação do componente por excesso de corrente.

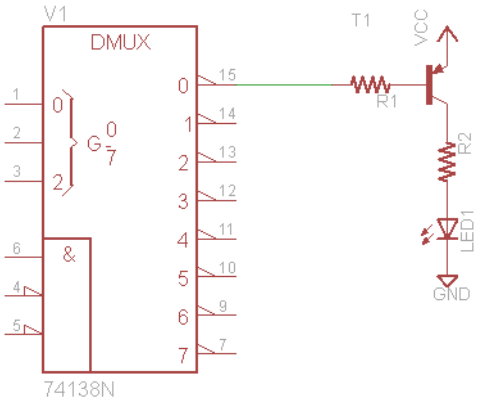



**Figura 3.15 - Acionamento do LED através do sinal de saída do 74138**

### 3.4.2. Cálculo dos Componentes

Como foi visto no item anterior, foram feitos dois circuitos independentes, apesar de utilizarem o mesmo sinal de controle. Um circuito para funcionar como interface com o usuário, contendo botões e LEDs e outro para indicar qual coluna que está sendo feita a leitura.

*a) A matriz de botões:*

A matriz de colunas é um circuito que aparenta ser complexo, por possuir muitos componentes, no entanto o funcionamento se dá por módulos individuais que têm uma estrutura simples, com cada botão representando um circuito individual, que se repete exatamente para todos os outros botões.

O módulo de um botão pode se encontrar em dois estados quando a chave está fechada (Figura 3.16).

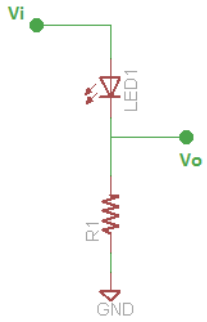



**Figura 3.16 - Circuito do botão com a chave fechada**

Para calcular o valor do resistor, foram levados em consideração os parâmetros do LED utilizado no projeto. O LED utilizado foi o BL-L314, que possui $V_f$=*2,25V* e $I_f$=*25mA* (BetLux Eletronics, 2001). Dessa forma, para $V_i$=*5V* temos (JOHNSON, et al., 2000):

$$R = \frac{(V_i - V_f)}{I_f}$$

Foi possível, então, determinar o valor do resistor a ser utilizado, que foi de ***120Ω*** (valor comercial).

Observa-se que quando a chave está fechada, para Vcc igual a 5V, o valor de $V_o$ é de 2,75V. Considerando que o 74151 reconhece como $V_{IH}$ tensões entre 2V e 5V, o sinal $V_o$ satisfaz a condição de nível lógico 1.

Em todos os outros estados, ou seja, com a chave aberta ou com a chave fechada porém com Vi sendo igual a 0, o sinal $V_o$ está aterrado, garantindo o nível lógico 0 para esses casos.

*b) LED de indicação de leitura de coluna:*

Para o circuito do LED de indicação de leitura de coluna, foi utilizado o circuito mostrado na Figura 3.17.

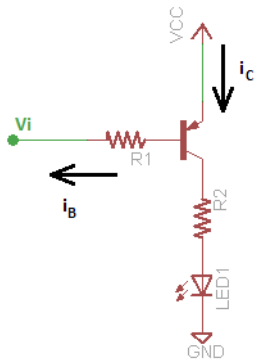



**Figura 3.17 - Circuito de ativação do LED através da saída do 74138**

O transistor, nesse caso, tem a função de chave eletrônica. Ele deve operar na saturação quando estiver conduzido e no corte quando não estiver conduzindo.

Para calcular o valor dos componentes deve-se saber a corrente que o transistor vai fornecer à carga, o $h_{fe}$ (ganho) do transistor e a tensão de entrada para colocar o transistor em modo de operação. Neste caso o transistor funciona como chave fechada com $V_i$ igual a 0 e como chave aberta com $V_i$ igual a 5V.

A corrente que será fornecida ao LED é a mesma descrita no item *a)* e, portanto, o método para calcular o resistor em série com o LED é o mesmo. Sendo assim $R_2$ tem o valor de **120Ω**.

Para o cálculo do resistor conectado à base do transistor deve considerar que ele opera no corte. Neste caso, temos:

$$i_B = \frac{i_C}{h_{fe}}$$

Para garantir o corte, multiplica-se $i_c$ por um fator de saturação forçada (*overdrive factor*), neste caso, igual a 2 (SEDRA, et al., 2000). E obtém-se:

$$i_B = 2\frac{i_C}{h_{fe}}$$

E temos que $R_1$ é dado por:

$$R_1 = \frac{\left((V_{CC} - V_{EB}) - V_i\right)}{i_B}$$

O transistor utilizado no projeto foi o **BC5558b**, que possui como características $V_{BE}$ igual a -0,7 V e $h_{fe}$ igual a 180 (Motorola Semiconductor, 1996). Utilizando-se estes

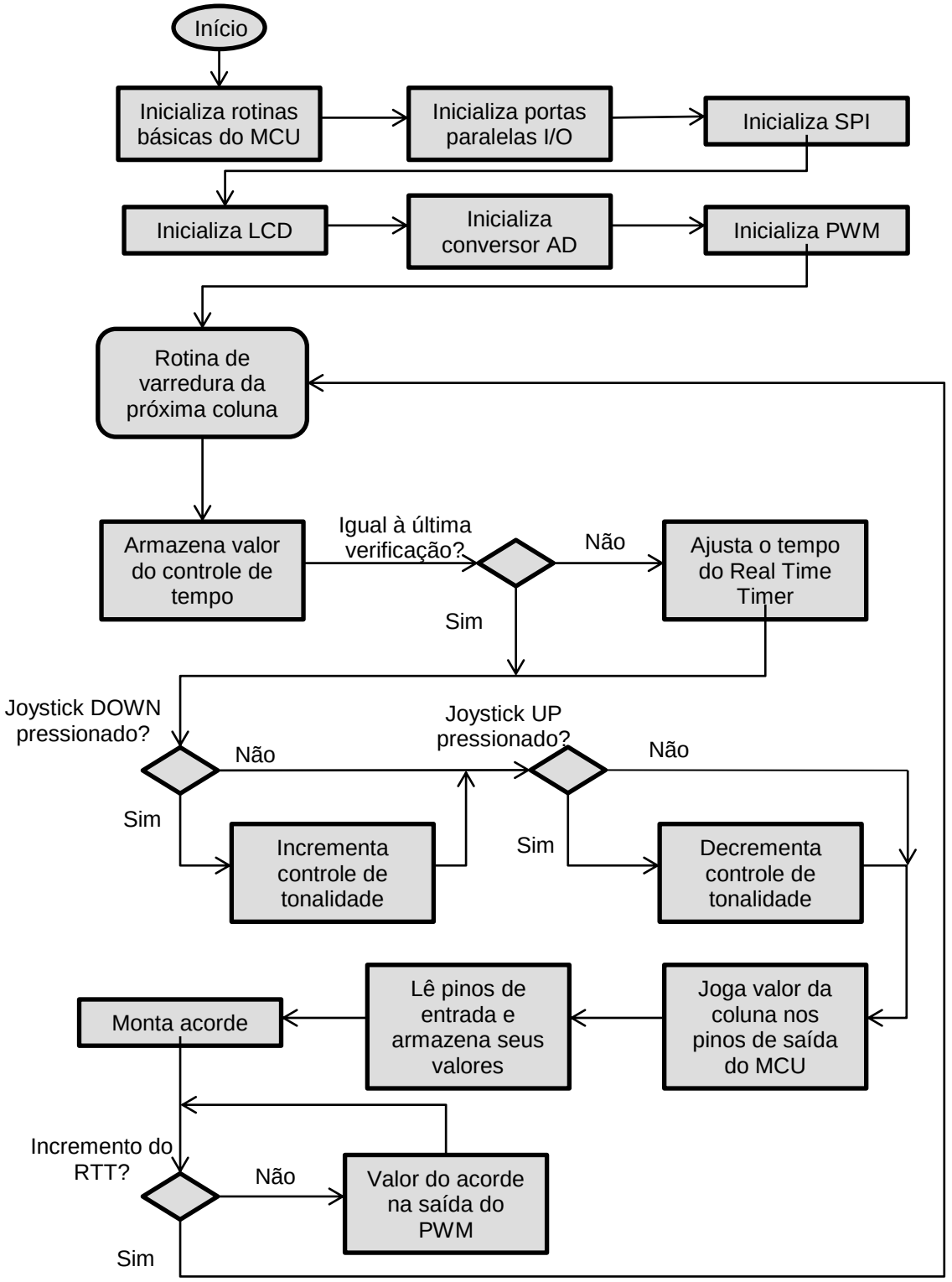
parâmetro e mais $i_c$ igual 25mA e $V_i$ igual a zero obtém-se que o resistor $R_1$ deve ser de **15kΩ** (valor comercial).

## 3.5. O Software

O programa para o microcontrolador AT91SAM7X256 foi criado em linguagem C, no ambiente de desenvolvimento Eclipse. O kit de SAM7-EX256 trás consigo um CD de instalação deste software, já contendo os *plug-ins* necessários para realizar o *debug* e a programação do MCU por meio da interface JTAG e também alguns projetos exemplos em que interface do ARM com os periféricos do kit já estão estruturados.

No site da Atmel, está disponível uma biblioteca “.h” onde está estruturada toda a pilha de comandos do AT91SAM7X256, com todos os seus registradores detalhados e endereçados da maneira correta.

### 3.5.1. Fluxograma do programa desenvolvido



### 3.5.2. Detalhamento das Rotinas

***a) Inicialização de Rotinas básicas do MCU***

Nessa etapa são definidos os parâmetros iniciais do AT91SAM7X256, como habilitação de *reset*, desabilitação do *watchdog* e *setup* dos osciladores do MCU

***b) Inicialização das portas I/O***

Nesta etapa é habilitado o *clock* para as duas portas paralelas PIOA e PIOB, além de definir quais as portas serão *input* e quais serão *output*. Para a comunicação com o teclado matricial definiu-se as portas PA27, PA28 e PA29 como *output* e as portas PB21, PB27, PB28, PB29 e PB30 como *input.*

***c) Inicialização do SPI***

A inicialização do SPI deve ser feita para realizar a comunicação com o LCD. Nesta etapa são habilitados os pinos SPCK0, NPCS0, MISO0 e MOSI0, é habilitado o *clock* para o SPI, o MCU é selecionado como mestre e são definidos os parâmetros da comunicação, como 9 bits de transferência, polaridade em estado inativo em nível alto e o *clock* da comunicação em 125kHz.

***d) Inicialização LCD***

Este processo realiza todos os passos necessários para configurar o *display* LCD para funcionar em um modo compatível com a interface implementada com o microcontrolador AT91SAM7X256, como a orientação do *display* para a visualização exata da imagem.

Nessa etapa é enviada ao *display* a imagem que fica como “papel de parede” do LCD. Essa imagem é convertida a fim de possuir as especificações de tamanho e as características do vetor RGB imposto pelo *display*. Essa imagem é armazenada na memória do MCU através de um vetor *const int.*

***e) Inicialização do conversor AD***

Este processo habilita o *clock* para a conversão AD e configura o conversor para operar com *trigger* ativo por software e resolução de 10 bits

***f) Inicialização do PWM***

Neste processo, o *clock* para o PWM é habilitado e é atribuída a funcionalidade do pino relativo à saída do PWM para executar essa função. São realizadas também as configurações referentes ao modo que o PWM irá operar. Neste caso determina-se o período do PWM e a geração ou o controle da sua saída é feito alterando o valor do *duty cycle*.

O controle do período é feito através do registrador CPRD (*Channel Period Register*). O valor atribuído a este registrador é dividido pelo MCK e como resultado tem-se o período do PWM.

***g) Programa Principal***

Após a realização de todas as rotinas de inicialização o programa principal começa a executar em *loop* infinito.

Primeiro faz-se a verificação do controle de velocidade de reprodução. A sua entrada é através do conversor AD e esse valor configura o *preset* do *Real Time Clock* (RTT). O RTT será o responsável por controlar o tempo que o MCU irá permanecer na rotina de reprodução do sinal de áudio.

O *preset* do RTT é o valor entre 0 e 65.535, que divide o *Slow Clock* (SLCK) do MCU para configurar a frequência do *timer* (Atmel Corporation, 2009).

$$f_{RTT} = \frac{f_{SLCK}}{RTPRES}$$

O SLCK é associado ao oscilador RC interno do MCU e varia sua frequência entre 22kHz e 42kHz, dependendo do dispositivo. Utilizando a frequência típica do oscilador (32kHz ou 32.768Hz para facilitar os cálculos) definiu-se como frequências mínimas e máximas do sintetizador musical os valores 2Hz e 16Hz. Esses valores, em consequência, devem ser as frequências mínima e máxima do SLCK. A partir daí é possível determinar que para $f_{min}$, temos RTPRES = 4000h e para $f_{max}$, temos RTPRES = 800h.

No controle do RTT pelo *Trimpot* determina-se frequência mínima quando o seu valor lido no conversor AD é zero e máxima quando é 1023. Deseja-se uma relação linear entre o valor fornecido pelo AD e o RTPRES.

$$V_{trimp} * \alpha + \beta = RTPRES$$

Para calcular α e β, basta resolver um sistema linear e é possível determinar que $\alpha = -14$ e $\beta = 16398$ (Figura 3.18).

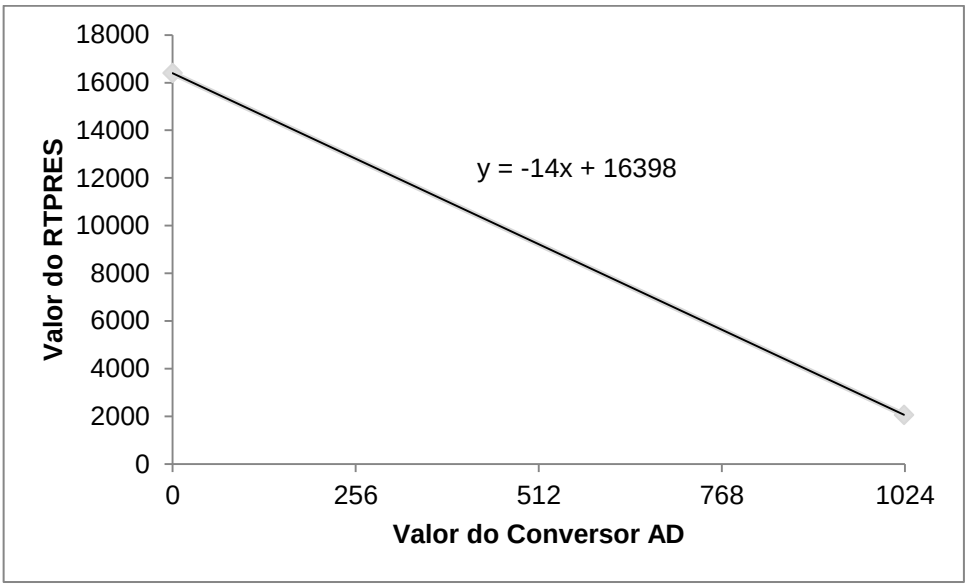



**Figura 3.18 - Relação linear entre o valor obtido pelo potenciômetro e o *preset* do RTT**

```
time_control = ADCGetChannel(ADC_CHN_6);

if(time_control!=time_rpt)
{
    //Determina o divisor do Slow Clock (SLCK), que possui clock típico de 32.768 Hz.
    //O preset é dado por um registrador de 16 bits (RTPRES) do Real Time Timer e é controlado
    //a partir do sinal do ADC. Os valores de ADC variam de 0 a 1023.
    //
    //Para time_control mínimo = 0, deseja-se RTPRES = 16384 = 4000h
    //  =>Obtêm-se a base de tempo do RTT igual a 2Hz (se SLCK = 32kHz)
    //Para tiem control máximo = 1023, deseja-se RTPRES = 2048 = 800h
    //  =>Obtêm-se a base de tempo do RTT igual a 16Hz (se SLCK = 32kHz)
    //
    RTPRES= (-14)*(time_control+1)+16398;
    pRTT->RTTC_RTMR = RTPRES;

    //Controle para não repetir a rotina caso o controle do tempo permaneça inalterado
    time_rpt=time_control;
}
```

**Figura 3.19 - Rotina de controle do tempo**

Com a base de tempo determinada (Figura 3.19) verifica-se se o usuário está alterando a tonalidade do conjunto de notas ao verificar se o *joystick* está sendo ativado para cima ou para baixo (Figura 3.20). A variável de controle dessa função irá servir como parâmetro para uma rotina de atraso entre a atribuição do valor de cada ponto do sinal de áudio, alterando a sua frequência. Por tanto quanto maior o valor da variável de controle de tonalidade, maior será o tempo de atraso, aumentando, assim, o período da onda final (diminui a frequência – tonalidade mais grave).

```
if(!((pPIOA->PIO_PDSR) & BIT8)) //DOWN Joystick abaixa o tom das notas
{
    if(tone<20)
    tone++;
}

if(!((pPIOA->PIO_PDSR) & BIT9)) //UP Joystick eleva o tom das notas
{
    if(tone>0)
    tone--;
}
```

**Figura 3.20 - Controle da tonalidade do conjunto de notas**

O processo de controle de leitura da coluna é feito através de um *loop for* que varia seu valor entre 0 e 7 (ou em binário 000 até 111). Esse valor é atribuído às portas PA27, PA28 e PA29 (Figura 3.21). Este valor é multiplexado no circuito do teclado para selecionar a coluna correspondente.

```
//Zera as portas PA27,PA28 e PA29
//em seguida atribui o valor binário correspondente
//à coluna do teclado às portas de saída
pPIOA->PIO_CODR = output;
pPIOA->PIO_SODR = (k << 27);
```

**Figura 3.21 - Atribuição do valor correspondente à coluna do teclado às portas I/O**

Com a coluna desejada já selecionada é feita a leitura dos botões correspondentes a cada nota através das portas PB21, PB27, PB28, PB29 e PB20. Cada valor é atribuído a uma posição distinta de um vetor, que será utilizado para montar o acorde correspondente através do acesso à tabela de valores salva na memória do MCU. Após ter sido montado o acorde, é feito um ajuste dos valores para que haja compatibilidade com a saída do PWM, já que os valores presentes na tabela contêm números negativos e o PWM deve receber valores entre 0 e 1023 (Figura 3.22).

```
//Lê notas
rd_note[0] = ((pPIOB->PIO_PDSR & BIT21));
rd_note[1] = ((pPIOB->PIO_PDSR & BIT27));
rd_note[2] = ((pPIOB->PIO_PDSR & BIT28));
rd_note[3] = ((pPIOB->PIO_PDSR & BIT29));
rd_note[4] = ((pPIOB->PIO_PDSR & BIT30));

//Define o "acorde"
for(i=0;i<5;i++)
{
    if(rd_note[i])
    {
        for(j=0;j<2400;j++)
            chord[j]+=notes[i][j];
    }
}
//Ajusta vetor para atribuí-lo à saída do PWM
for(i=0;i<2400;i++)
    chord[i]+=512;
```

**Figura 3.22 - Rotina de leitura dos botões e montagem do acorde**

Após a realização de todo o processo de controle da interface com o usuário e de montagem do sinal sonoro, finalmente esta onda é reproduzida através do PWM. O tempo que o sinal de áudio é reproduzido depende da frequência determinada para o RTT, ou seja, a som é ouvido até que haja o incremento do temporizador (Figura 3.23).

```
//Toca acorde
while((pRTT->RTTC_RTSR & AT91C_RTTC_RTTINC)==0) //Toca até o  incremento do RTT
{
    for(i=0;i<2400;i++)
    {
        PWMSetValue(chord[i]);
        Delay(tone); //controle da tonalidade
    }
}

//Reset da variável "chord"
for(i=0;i<2400;i++)
    chord[i]=0;
```

**Figura 3.23 - Rotina responsável pela reprodução do sinal sonoro**

# Capítulo 4

# 4. Discussão dos Resultados e Conclusão

## 4.1. O Teclado de Interface com o Usuário

A interface com o usuário foi construída em um *protoboard* para testes e foi confeccionado um circuito impresso para a posterior organização dos componentes. O resultado montado em *protoboard* pode ser visto na Figura 4.1.

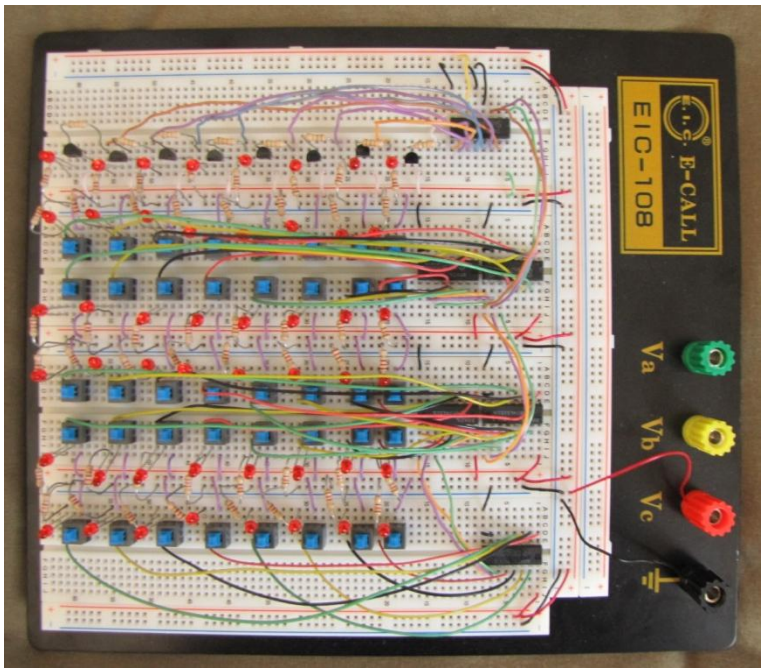



**Figura 4.1 - *Protoboard* com o teclado 5x8 para interface com o usuário**

A escolha por uma matriz 5x8 se mostrou suficiente para a confecção do protótipo. Com as cinco linhas foi possível obter uma escala pentatônica completa e com as oito colunas foi possível aproveitar todo o potencial dos multiplexadores. Dessa forma pôde-se, não só confeccionar um dispositivo funcional, mas também um instrumento que apresentasse, de fato, musicalidade. Isto seria algo que não ocorreria caso se utilizasse uma matriz com menos colunas (a sequência de notas seria repetitiva demais por possuir poucos passos de varredura para formar uma melodia) ou com menos linhas (quatro notas ou menos limitaria ainda mais as possíveis combinações harmônicas).

## 4.2. Formas de Onda e Temporização

Com o auxílio de um osciloscópio digital foi possível fazer a aquisição dos sinais utilizados no projeto, tais como o sinal de controle de varredura da matriz e os sinais de áudio gerados pelo PWM, para comparar com os resultados esperados.

### 4.2.1. Sinal de Controle de Varredura

Com o osciloscópio conectado às saídas PA27, PA28 e PA29 do MCU, foi possível identificar se os sinais produzidos representam as formas de onda desejadas para o controle de varredura da matriz.

Através das Figura 4.2, Figura 4.3 e Figura 4.4 e dos valores coletados e apresentados nas Tabela 6 e Tabela 7 é possível observar que a frequência gerada pelo pino PA27 é o dobro da frequência no pino PA28, que possui o dobro da frequência do sinal no pino PA29. Em consequência desse padrão, o sinal de controle que serve como entrada para os MUX e para o decodificador realmente segue a seguinte ordem:

| PA29 | PA28 | PA27 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

Logo a sequência de varredura da matriz é cumprida da forma correta.

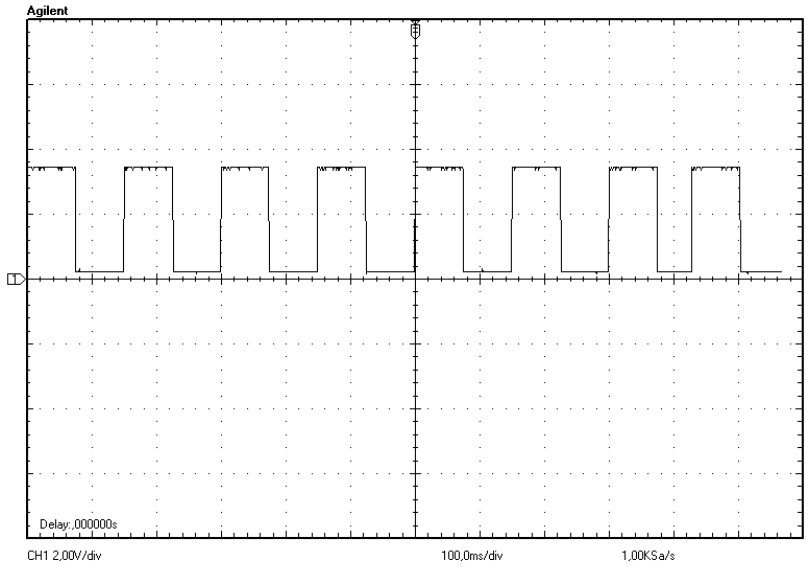



**Figura 4.2 - Sinal de controle no pino PA27 na frequência máxima**

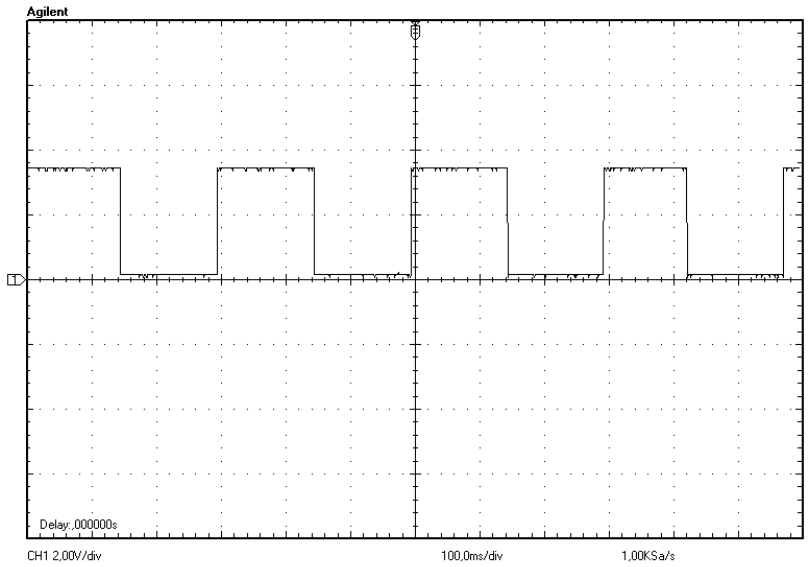



**Figura 4.3 - Sinal de controle no pino PA28 na frequência máxima**

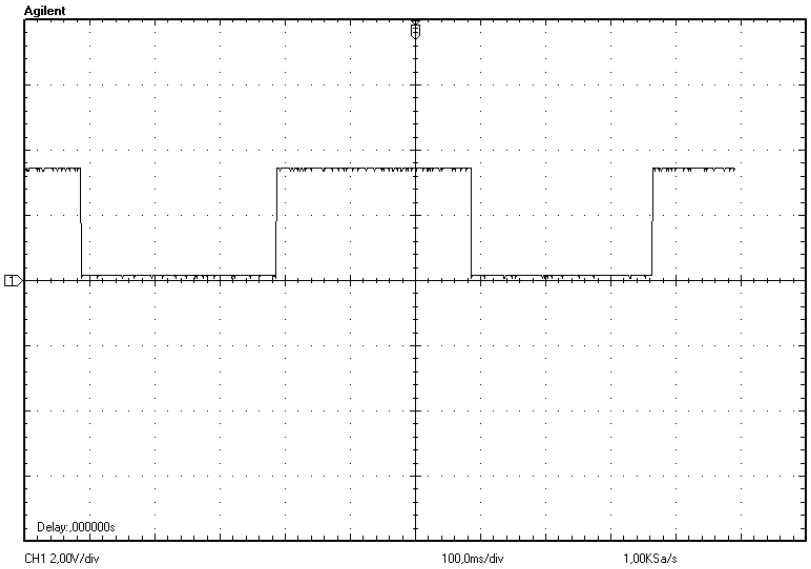



**Figura 4.4 - Sinal de controle no pino PA29 na frequência máxima**

**Tabela 6 - Frequência máxima do sinal de controle de varredura**

| Frequência Máxima | | | |
|---|---|---|---|
| | Frequência de onda quadrada (Hz) | VOH | VOL |
| PA27 | 6,711 | 0,08 | 3,36 |
| PA28 | 3,333 | 0,00 | 3,36 |
| PA29 | 1,669 | 0,00 | 3,36 |
| Frequência do RTT (2 x freq. PA27) | | | |
| 13,4 Hz | | | |

**Tabela 7 - Frequência mínima do sinal de controle de varredura**

| Frequência Mínima | | | |
|---|---|---|---|
| | Frequência de onda quadrada (Hz) | VOH | VOL |
| PA27 | 0,8584 | 0,08 | 3,36 |
| PA28 | 0,4274 | 0,00 | 3,36 |
| PA29 | 0,2132 | 0,00 | 3,36 |
| Frequência do RTT (2 x freq. PA27) | | | |
| 1,7 Hz | | | |

Observa-se que a frequência mínima e máxima são ligeiramente diferentes da frequência determinada no momento do projeto. Isso porque, para calcular estes valores, considerou-se a frequência típica do oscilador RC, que é de 32 kHz. Mas analisando os resultados, é possível observar que o SLCK opera na realidade com uma frequência de 28 kHz (13,4 x 2.062 e 1,7 x 16.398), no caso deste microcontrolador específico.

Caso fosse desejado manter as frequências de varredura mínima e máxima em 2 Hz e 16 Hz, respectivamente, bastaria refazer os cálculos para determinar o valor de RTPRES que corresponderiam a estes valores, com a frequência de SLCK igual a 28 kHz ao invés de 32 kHz.

Optou-se por manter os valores iniciais, pois a escolha feita anteriormente já havia sido tomada atribuindo-se valores arbitrários para as frequências mínima e máxima de varredura.

### 4.2.2. Sinal Sonoro

Através do osciloscópio digital foi possível capturar a forma de onda do sinal sonoro gerado pelo PWM e então compará-lo com a forma de onda teórica simulada. Além disso, foi possível observar o espectro de frequências obtido através da função transformada de *Fourier* presente no osciloscópio.

a) Uma nota:

Na Figura 4.5 é possível observar o sinal da saída do alto-falante referente à Nota I. Pela imagem da tela do osciloscópio é possível observar que um período possui aproximadamente 1,8 divisões. Sendo cada divisão 500us, isto determina que a frequência da nota seja de aproximadamente 1.111 Hz. Na Tabela 8 é possível visualizar o valor medido, sendo este igual a 1.124 Hz.

Na Figura 4.6 é possível observar a forma de onda simulada e notar que o resultado prático obtido condiz com o esperado.

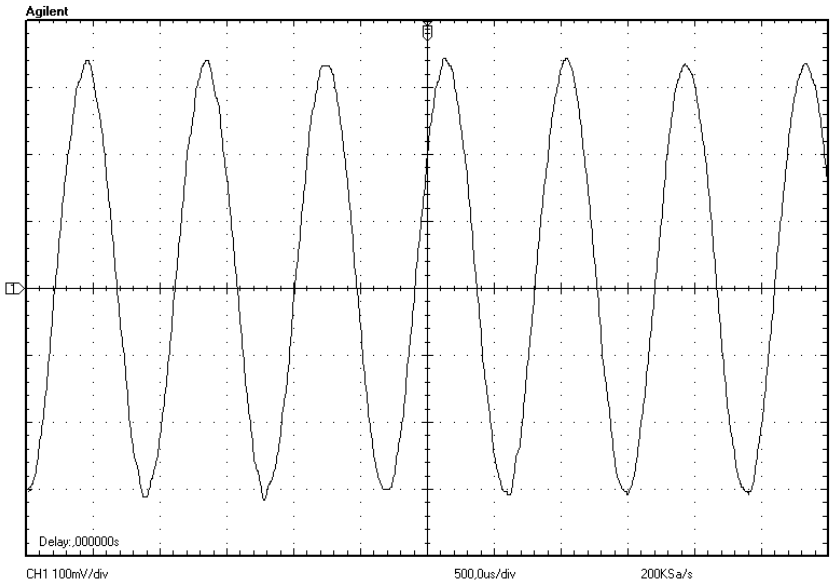



**Figura 4.5 - Nota I - Sinal na saída do alto-falante**

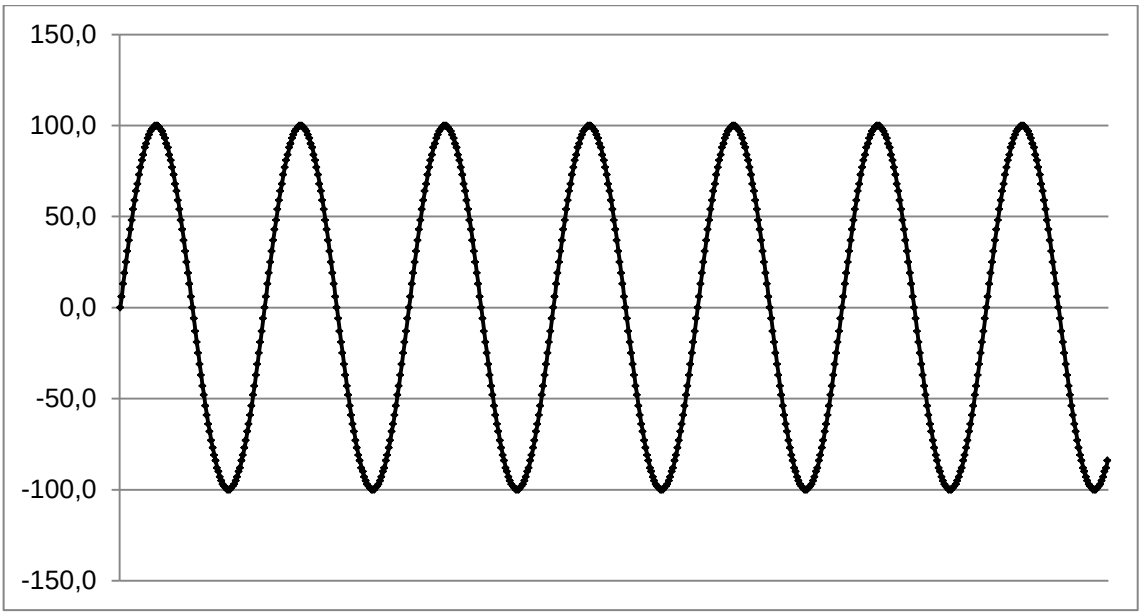



**Figura 4.6 - Nota I - Simulação**

b) Duas Notas:

Na Figura 4.7 é possível observar o sinal colhido na saída do alto-falante representando a combinação entre as notas I e II. Na teoria, a nota II deve ter a sua frequência 1,125 (9/8) vez maior que a nota I. É possível observar pela transformada de *Fourier* que a diferença entre as duas frequências fundamentais é de aproximadamente 0,45 divisões, ou 140 Hz, já que cada divisão representa 312,5 Hz.

Considerando essa diferença e sabendo que a frequência da nota I é de 1.124 Hz, a nota II possui aproximadamente 1.264 Hz. Observa-se que a razão entre as duas notas é igual a 1,125. Exatamente o valor desejado na teoria.

Comparando-se com a forma de onda simulada da Figura 4.8 observa-se que o resultado obtido está dentro do desejado.

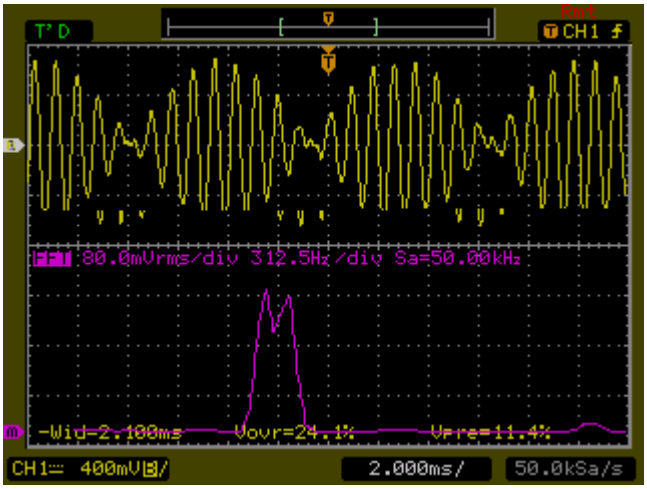



**Figura 4.7 - Notas I+II – Sinal na saída do alto-falante e sua transformada de *Fourier***

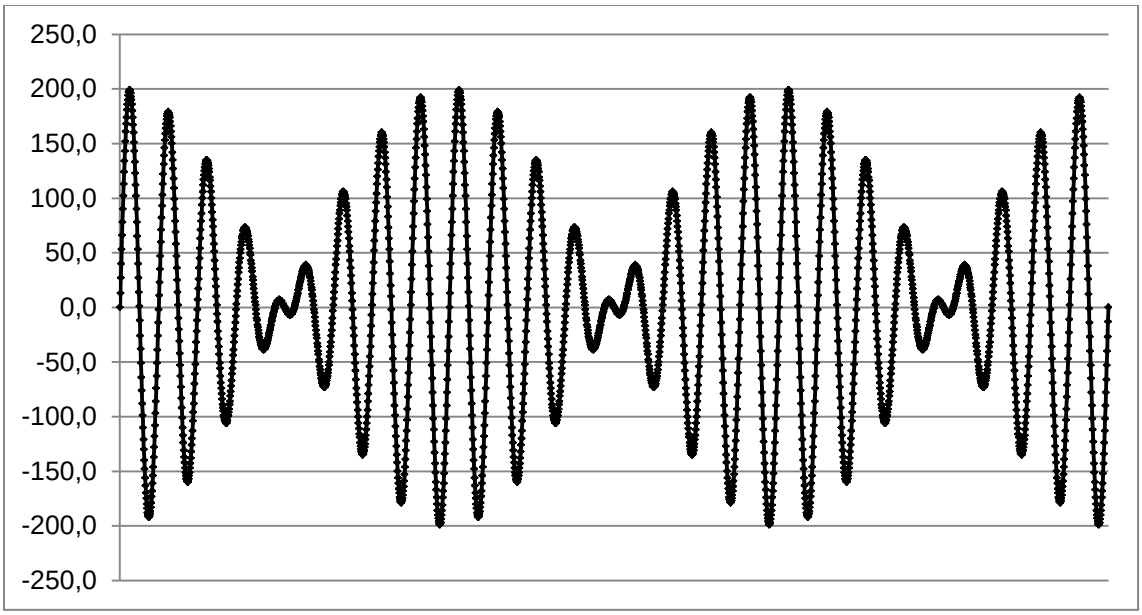



**Figura 4.8 - Notas I+II – Simulação**

c) Três Notas:

Neste teste foi adquirida a forma de onda resultante da combinação entre as notas I, II e IV. Pela transformada de *Fourier* (Figura 4.9) observa-se que o espectro referente à nota IV está localizado a aproximadamente 1,1 divisão da primeira nota, ou aproximadamente

343,4 Hz. Com a frequência da nota I igual a 1.124Hz, temos que a terceira nota possui frequência igual a 1467,4 Hz, sendo a razão entre as duas igual a 1,3. Sabendo-se que, na teoria, a razão deveria ser 1,333 (4/3) é possível observar que o resultado obtido está dentro do esperado.

Além disso, a comparação entre a Figura 4.10 e a Figura 4.11 mostra que as formas de onda obtidas na prática e a simulada são equivalentes.

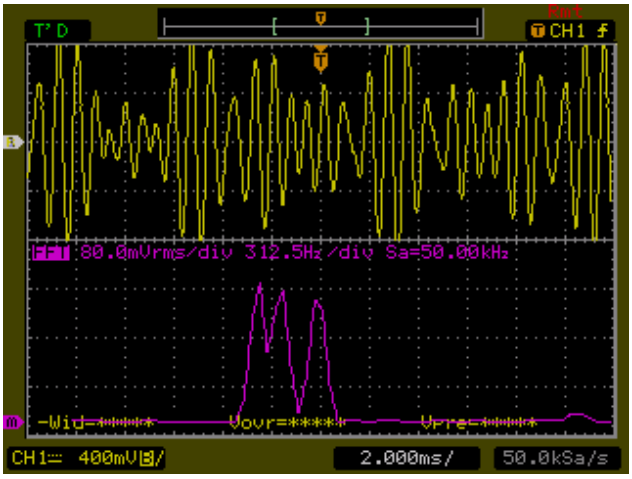



**Figura 4.9 - Notas I+II+IV - Saída do alto-falante e sua transformada de *Fourier***

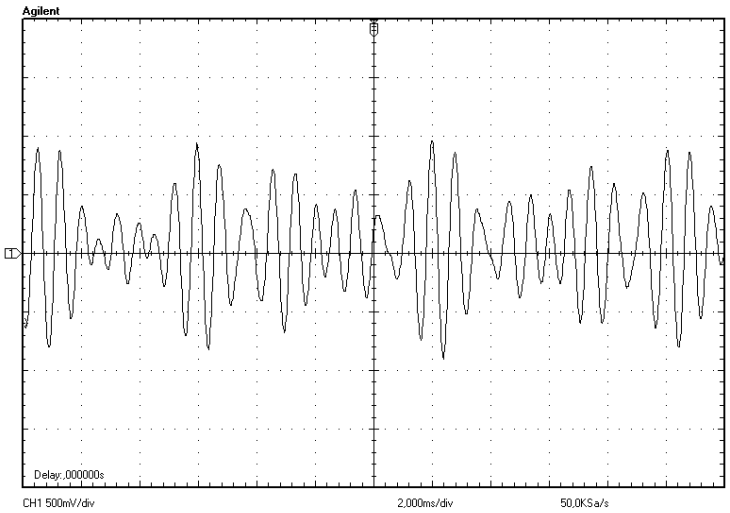



**Figura 4.10 - Notas I+II+IV - Saída do alto-falante**

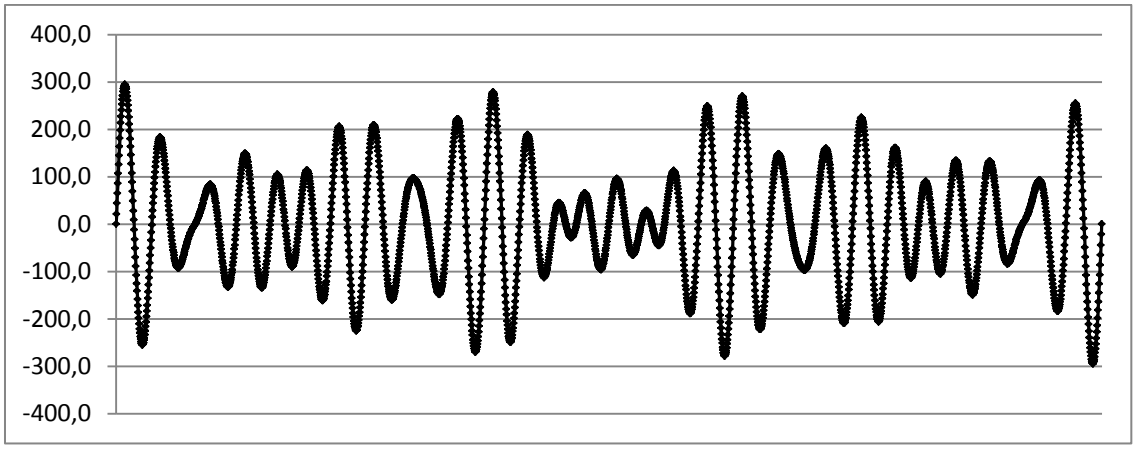



**Figura 4.11 - Notas I+II+IV – Simulação**

d) Quatro Notas:

Na Figura 4.12 é possível observar o sinal colhido na saída do alto-falante representando a combinação entre as notas I, II, IV e V. Na teoria, a nota V deve ter a sua frequência 1,5 (3/2) vez maior que a nota I. É possível observar pela transformada de *Fourier* que a diferença entre as duas frequências fundamentais é de aproximadamente 1,8 divisões, ou 562,5 Hz. Isso significa que a nota V possui frequência de aproximadamente 1.686,5 Hz e que a razão entre as duas notas é de 1,5. Por tanto, o resultado prático mostra compatibilidade com o esperado na teoria.

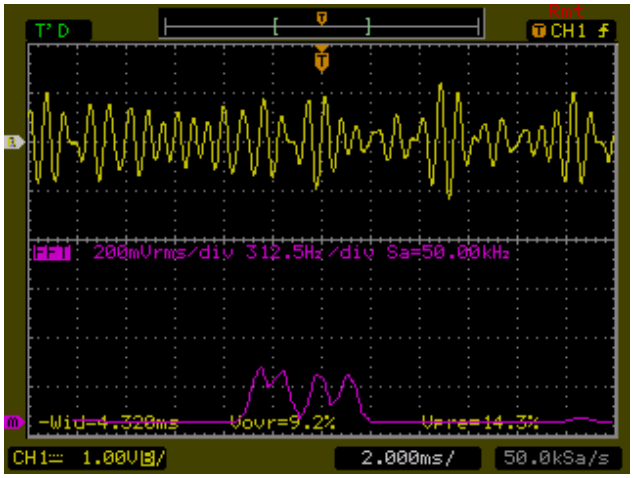



**Figura 4.12 - Notas I+II+IV+V - Saída do alto falante e sua transformada de *Fourier***

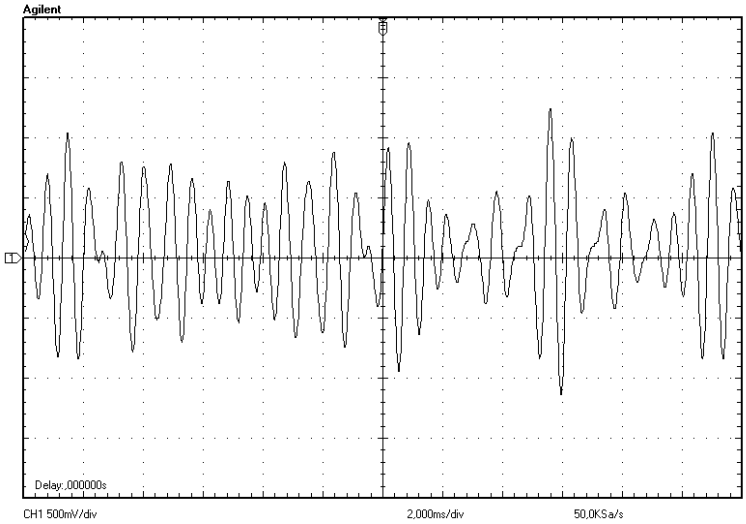



**Figura 4.13 - Notas I+II+IV+V - Saída do alto-falante**

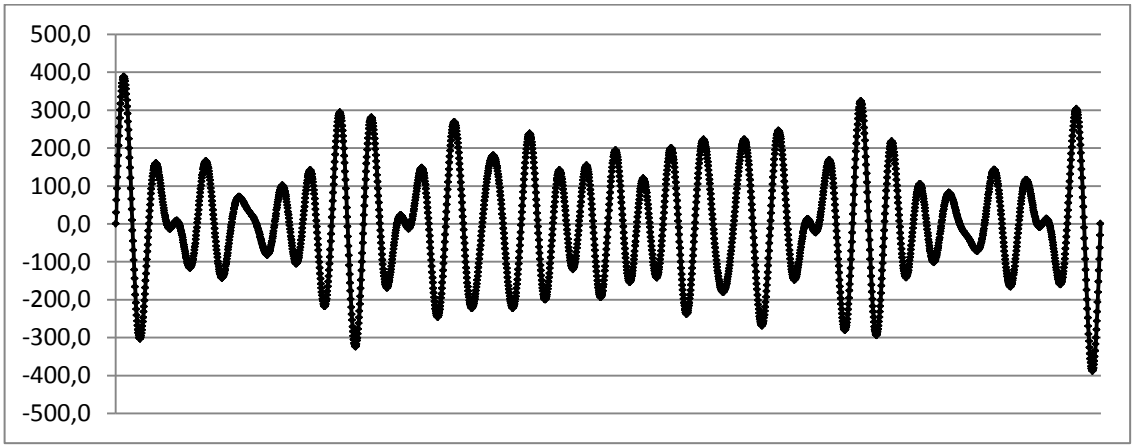



**Figura 4.14 - Notas I+II+IV+V – Simulação**

e) Cinco Notas:

Neste teste foi adquirida a forma de onda resultante da combinação todas as notas acionadas. Pela transformada de *Fourier* (Figura 4.15) observa-se que o espectro referente à nota V está localizado a aproximadamente 2,4 divisões da primeira nota, ou aproximadamente 750 Hz. Com a frequência da nota I igual a 1.124 Hz, temos que a quinta nota possui frequência igual a 1.874 Hz, sendo a razão entre as duas igual a 1,7. Sabendo-se que, na teoria, a razão deveria ser 1,667 (5/3) é possível observar que o resultado obtido está dentro do esperado.

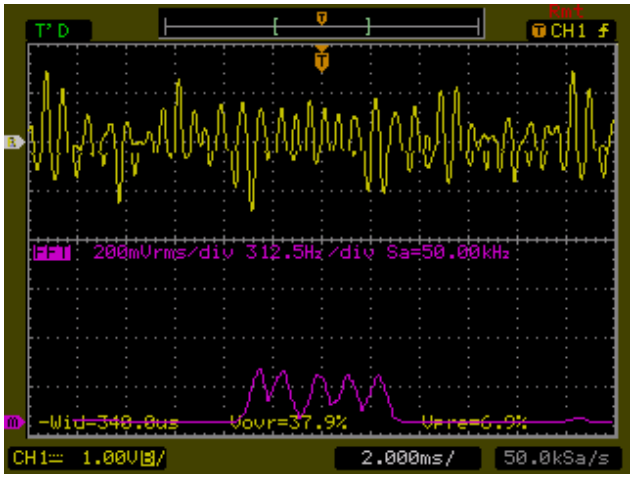



**Figura 4.15 - Notas I+II+IV+V+VI - Saída do alto-falante e sua transformada de Fourier**

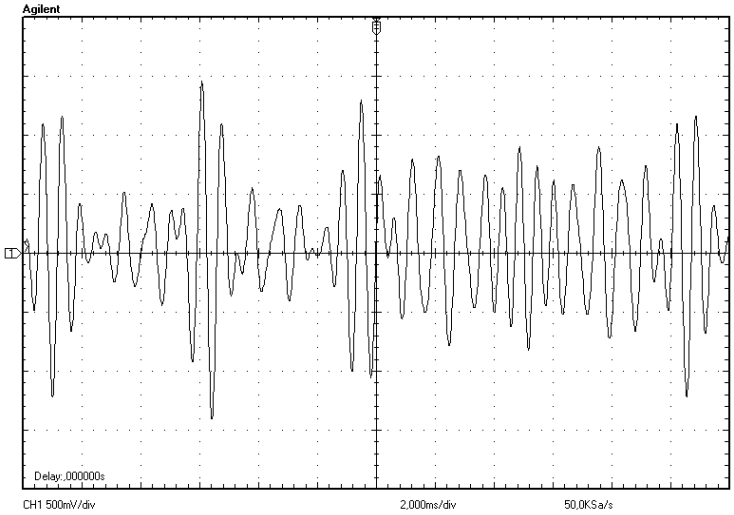



**Figura 4.16 - Notas I+II+IV+V+VI - Saída do alto-falante**

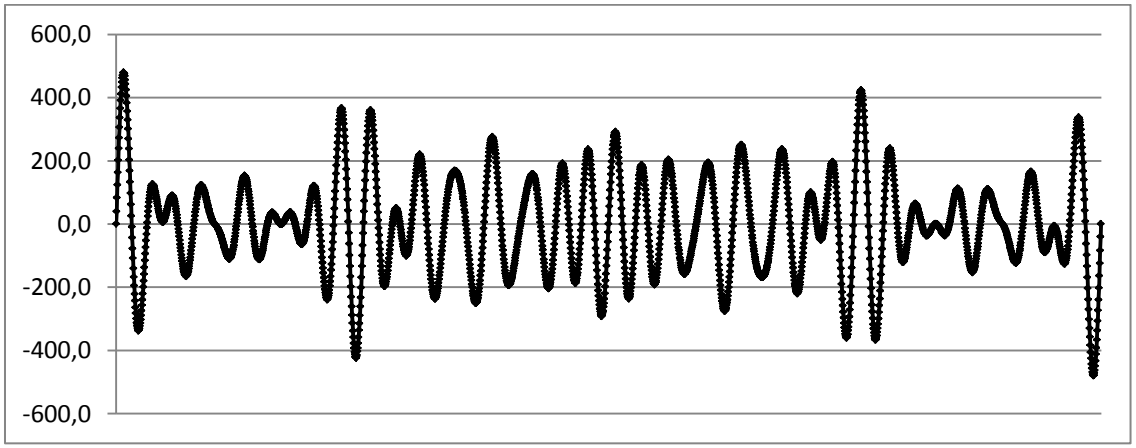



**Figura 4.17 - Notas I+II+IV+V+VI – Simulação**

Em todos os casos testados é possível notar que existe um pequeno espalhamento no espectro de frequência determinado pela transformada de *Fourier*. Teoricamente, se é gerado uma senóide pura, deveria haver somente um pico no espectro correspondente a frequência desta onda. No entanto, o processo de quantização do sinal produzido pelo microcontrolador acrescenta uma série de frequências diferentes no espectro além da fundamental.

Na prática, não é gerado uma senóide pura, mas sim, uma onda que possui como frequência fundamental a mesma frequência da senóide desejada, acrescentada de um espectro de frequências produzida pelo processo de quantização do sinal original.

f) Frequência das notas musicais:

Na Tabela 8 é mostrada a frequência de cada nota. A obtenção foi feita através do osciloscópio ao reproduzi-las uma de cada vez. Nota-se que os valores obtidos estão de acordo com o resultado desejado, com um erro máximo próximo de 1,2%.

**Tabela 8 - Frequências medidas e teóricas das notas musicais**

| | **Nota I** | **Nota II** | **Nota IV** | **Nota V** | **Nota VI** |
|---|---|---|---|---|---|
| **Frequência medida (Hz)** | 1.124 | 1.258 | 1.481 | 1.667 | 1.869 |
| **Relação de frequências** | *f* | *9f/8* | *4f/3* | *3f/2* | *5f/3* |
| **Frequência teórica (Hz)** | 1.124 | 1.265 | 1.499 | 1.686 | 1.873 |
| **Erro** | 0,0% | 0,5% | 1,2% | 1,1% | 0,2% |

## 4.3. Conclusão

No desenvolvimento do protótipo do sintetizador musical, foi possível trabalhar vários fundamentos, envolvendo desde o desenvolvimento de projetos de circuitos elétricos e eletrônicos até a aplicação de conceitos relacionados ao uso de microcontroladores.

Com certeza, desenvolver um projeto utilizando um microcontrolador baseado em uma arquitetura tão bem sucedida e com aplicações tão atuais quanto a ARM é uma experiência que agrega muito conhecimento.

Apesar de os microcontroladores da geração utilizada no *kit* (ARM7TDMI) já estarem um pouco defasados em relação às novas gerações, como a ARM9, ARM11 e os poderosos Cortex, é possível observar a versatilidade diante das diversas funcionalidades que esta arquitetura oferece ao usuário.

Foi possível observar que, mesmo apresentando os resultados dentro do escopo do projeto proposto, é possível agregar muitas outras funcionalidades utilizando-se os periféricos fornecidos pelo microcontrolador, como a comunicação USB ou USART, nativas no MCU ou a utilização de dispositivos de armazenamento de alta capacidade, como o SD/MMC presente no *kit*.

Desenvolver aplicações para microcontroladores é um exercício que estimula a busca por novas soluções e nos obriga a estar sempre agregando conhecimento, pois no ambiente digital a evolução é constante e há sempre novos dispositivos com novas funcionalidades sendo lançados constantemente no mercado.

# REFERÊNCIAS

**ALLWORTH, S. T. 1983.** *Introduction to Real-Time Software Design.* s.l. : McMillan Press, 1983.

**ARM Ltd. 2011.** ARM The Architecture for the Digital World. *ARM.* [Online] ARM Ltd., 2011. http://www.arm.com.

**Atmel Corporation. 2009.** *AT91 ARM Thumb-based Microcontrollers.* s.l. : Atmel Corporation, 2009.

**BERGER, A. S. 2001.** *Embedded Systems Design: An Introduction to Processes, Tools and Techiniques.* s.l. : CPM Books, 2001.

**BetLux Eletronics. 2001.** *Round Type LED lamp - BL-L314.* s.l. : BetLux Eletronics, 2001.

**BOYLESTAD, Robert L. e NASHELSKY, Louis. 2005.** *Dispositivos Eletrônicos e Teoria dos Circuitos.* s.l. : Prentice Hall, 2005.

**CadSoft Computer Inc. 2008.** *EAGLE - Easily Applicable Graphical Layout Editor Tutorial.* 2008.

**CDCC USP.** Física da Música - Frequências Musicais. *Centro de Divulgação Científico e Cultural - USP.* [Online] http://www.cdcc.usp.br/ondulatoria/musica2.html.

**Fairchild Semiconductor. 2000.** *DM74LS138 - DM74LS139 Decoder/Demultiplex.* 2000.

**HEATH, Steve. 2003.** *Embedded Systems Design.* 2003.

**JOHNSON, E. David, HILBURN, L. John e JOHNSON, R. Johnny. 2000.** *Fundamentos de Análise de Circuitos Elétricos.* s.l. : LTC - Livros Técnicos e Científicos Editora S.A., 2000.

**LYNCH, James P. 2007.** *Nokia 6100 LCD Display Driver.* 2007.

**MELLICHAMP, D. A. 1983.** *Real-Time Computing - With Application to Data Acquisition and Control.* s.l. : Van Nostrand Reinhold Company, 1983.

**Motorola Semiconductor. 1996.** *Amplifier Transistor PNP Silicon - BC556 B/BC557 A,B,C/BC558 B.* s.l. : Motorola Semiconductor, 1996.

**National Semiconductor. 1989.** *54150/DM54150/DM74150,54151/DM54151/DM74151A Data Selectors/Multiplexers.* 1989.

**Oppenheim, A. P. e Schafer, W. R. 1975.** *Digital Signal Processing.* s.l. : Practice-Hall, 1975.

**PEREIRA, Fábio. 2007.** *Tecnologia ARM: Microcontroladores de 32 Bits.* s.l. : Érica, 2007.

**SEDRA, Adel S. e SMITH, K. C. 2000.** *Microeletrônica.* São Paulo : Pearson Makron Books, 2000.

**Wikipedia. 2011.** Wikipedia The Free Encyclopedia. *Wikipedia The Free Encyclopedia.* [Online] 2011. http://en.wikipedia.org/.

# APÊNDICE A - Projeto do Circuito Impresso

O circuito impresso (PCB) é uma forma de tornar o projeto finalizado mais organizado e com um acabamento melhor, já que elimina a utilização de inúmeros fios para realizar as conexões entre os componentes, que podem, com o passar do tempo, se romper e causar mau contato.

O processo de criação do circuito impresso passa pelas etapas de criação do esquemático do circuito, dimensionamento da placa com o posicionamento dos componentes, criação dos arquivos responsáveis pelas instruções para a impressora e o fresamento do circuito em si.

*a) Esquemático do Circuito*

A criação do esquemático foi feita no software *Eagle 5.11*. Este software possui uma biblioteca com diversos componentes comerciais, com as suas dimensões e especificações de pinagem. Esta característica é muito útil para que depois seja realizado o posicionamento dos componentes na placa a ser confeccionada.

O esquemático pode ser visto na Figura A.1.

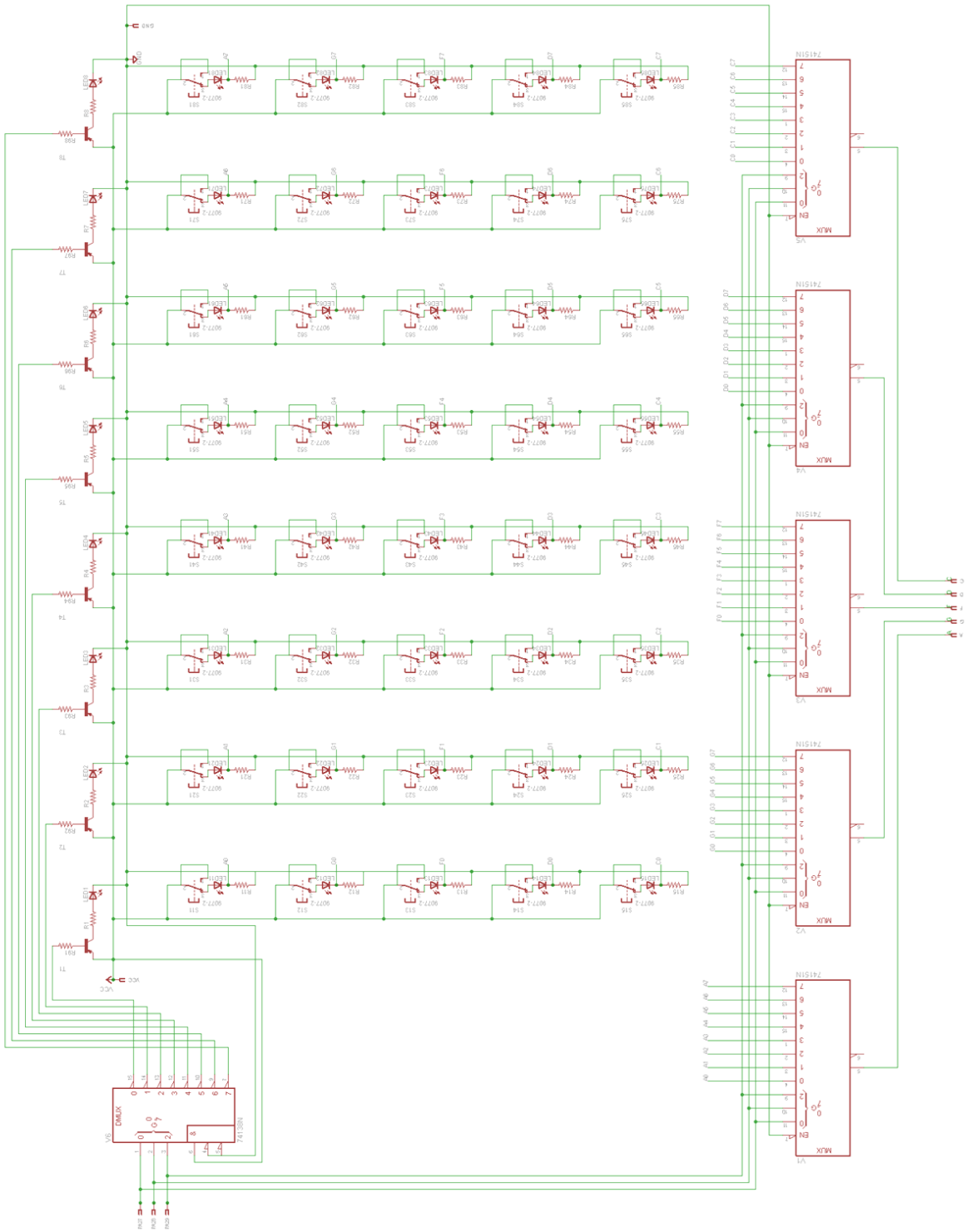


**Figura A.1 - Esquemático do circuito do teclado**

*b) Dimensionamento da Placa*

O software Eagle possui uma funcionalidade que transforma o esquemático do circuito criado em uma estrutura de componentes, que pode ser editado graficamente, para

dimensionar e posicionar os elementos do circuito da forma desejada (CadSoft Computer Inc., 2008).

Depois de posicionado os elementos é preciso especificar alguns parâmetros, como espessura da trilha e distância mínima entre uma trilha e outra. Com esses parâmetros, o programa é capaz de gerar através, de um algoritmo próprio, um caminho otimizado para as trilhas.

O resultado obtido nesse processo pode ser visto na Figura A.2.

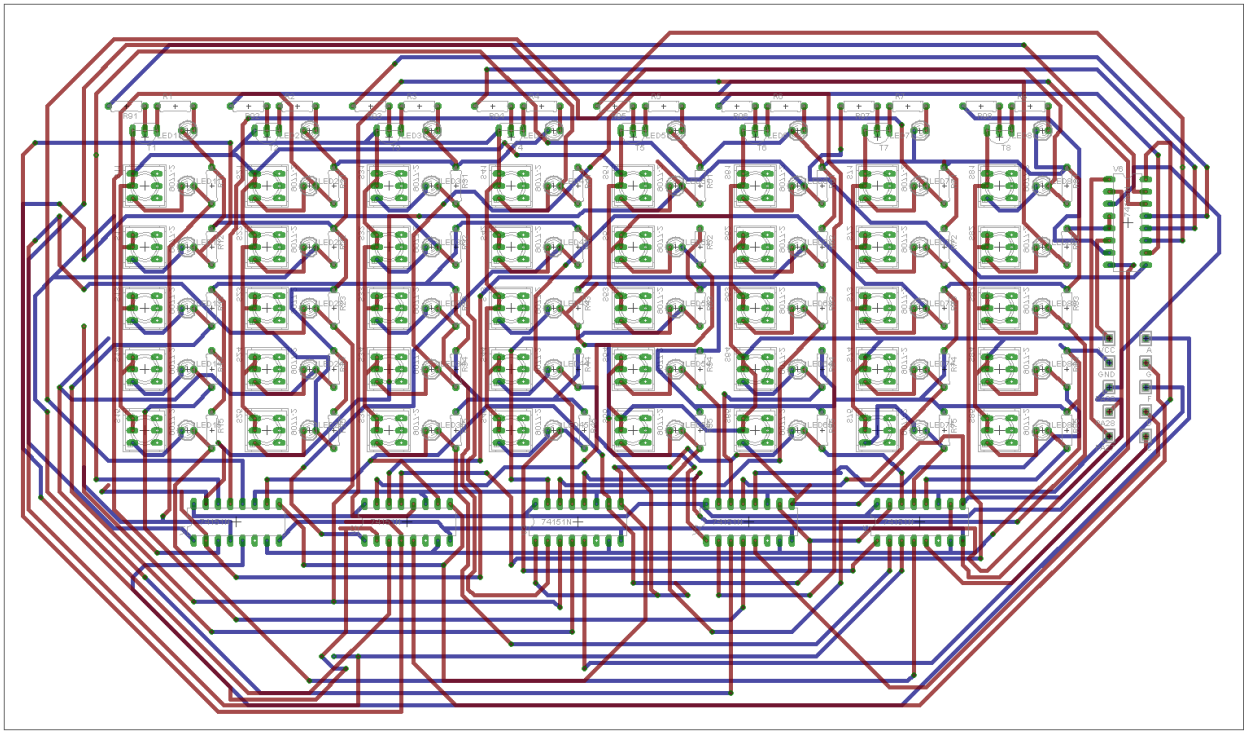


**Figura A.2 - Disposição do circuito com as trilhas desenhadas**

*c) Criação dos Arquivos de Exportação para a Impressora*

A partir do Eagle, deve-se exportar quatro arquivos para a criação do *layout*, que vai ser o arquivo base para ser impresso. Os arquivos são:

*.drl – mapeia informações de dimensão e posicionamento dos elementos do circuito

*.drd – exibe a posição dos furos (Figura A.3).

*.cmp – exibe as trilhas do lado dos componentes (*Top*) (Figura A.4).

*.sol – exibe as trilhas do lado da solda (*Botton*) (Figura A.5).

Esse arquivos devem ser importados no software CircuitCAM. Esse programa junta todos os elementos do circuito e gera um contorno ao redor das trilhas e *pads* por meio do comando "*Insulate"* (Figura A.6). Esse contorno será o caminho percorrido pela broca da fresa durante o processo de produção do circuito impresso.

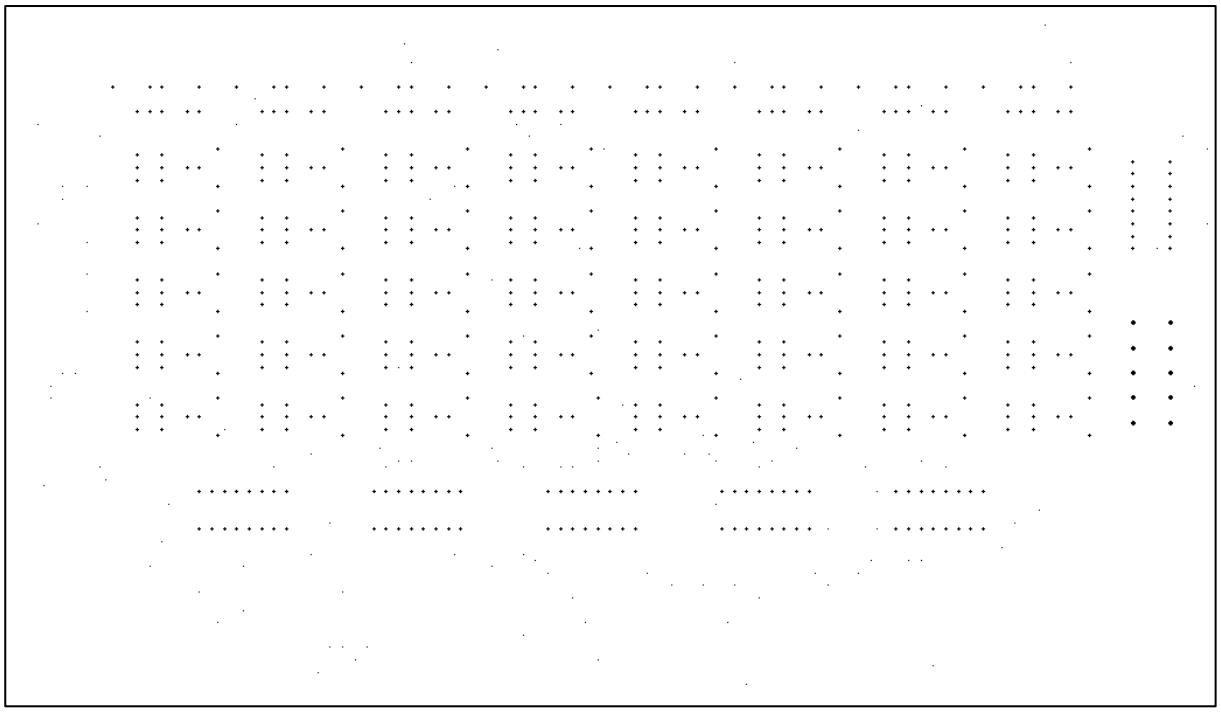


**Figura A.3 - Furos do circuito impresso**

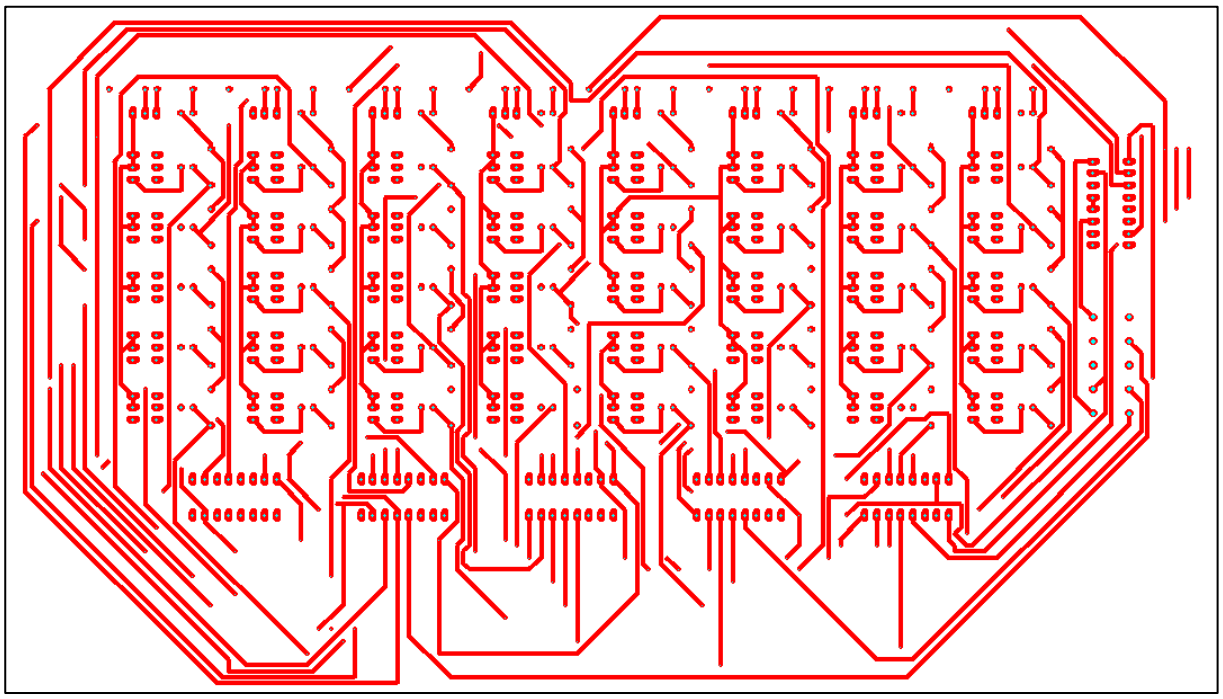


**Figura A.4 - *Top layer***

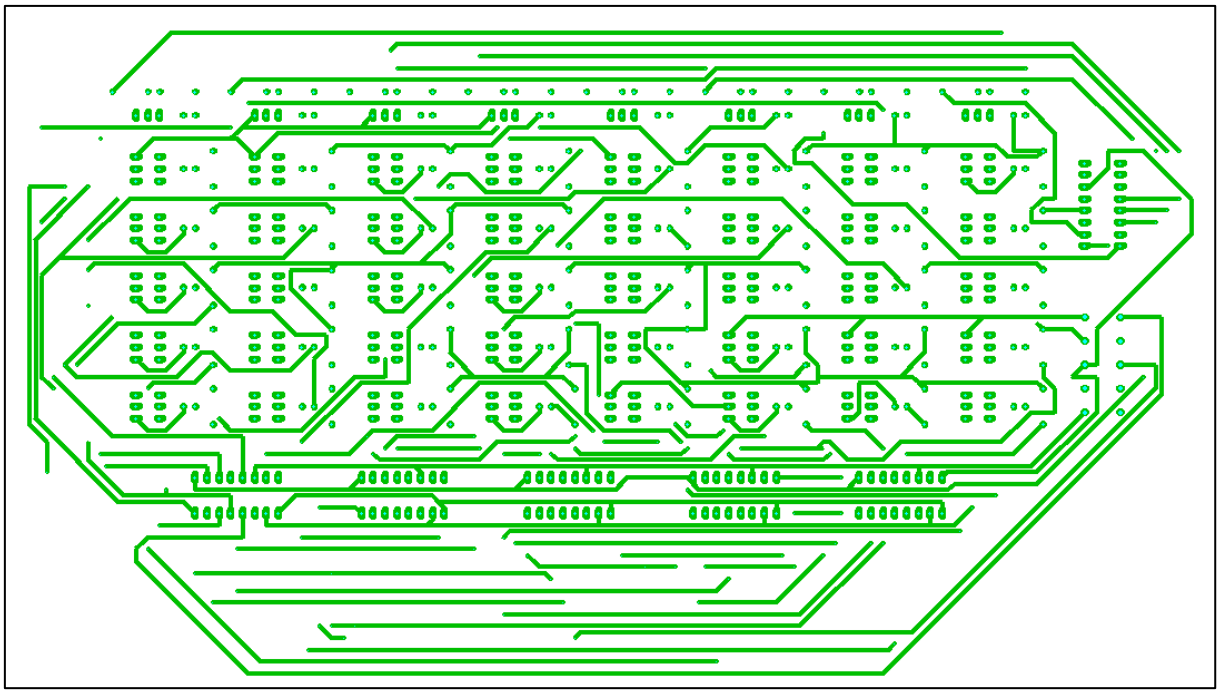


**Figura A.5 - *Botton layer***

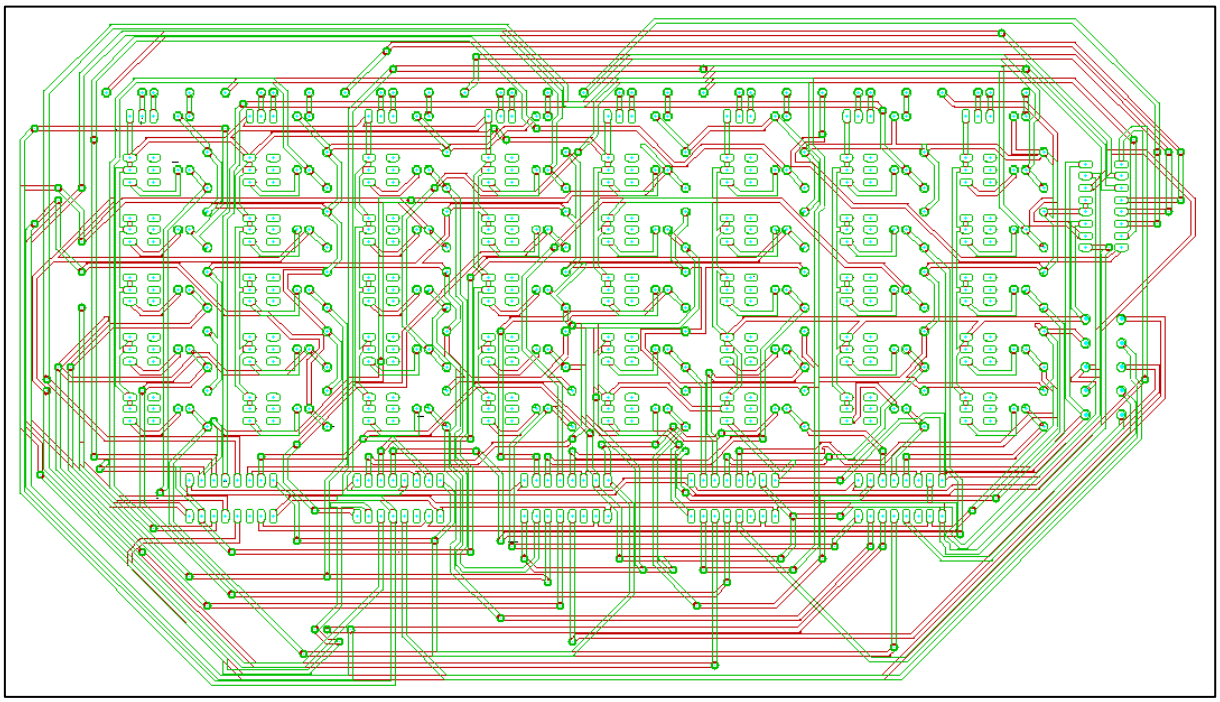


**Figura A.6 - *Insulate***

*d) Confecção da placa com uma fresa*

Com o programa CircuitCAM é possível exportar o arquivo .lmd, compatível com a fresa que está disponível no Departamento de Engenharia Elétrica da EESC-USP para a produção da placa do circuito impresso (Figura A.7).

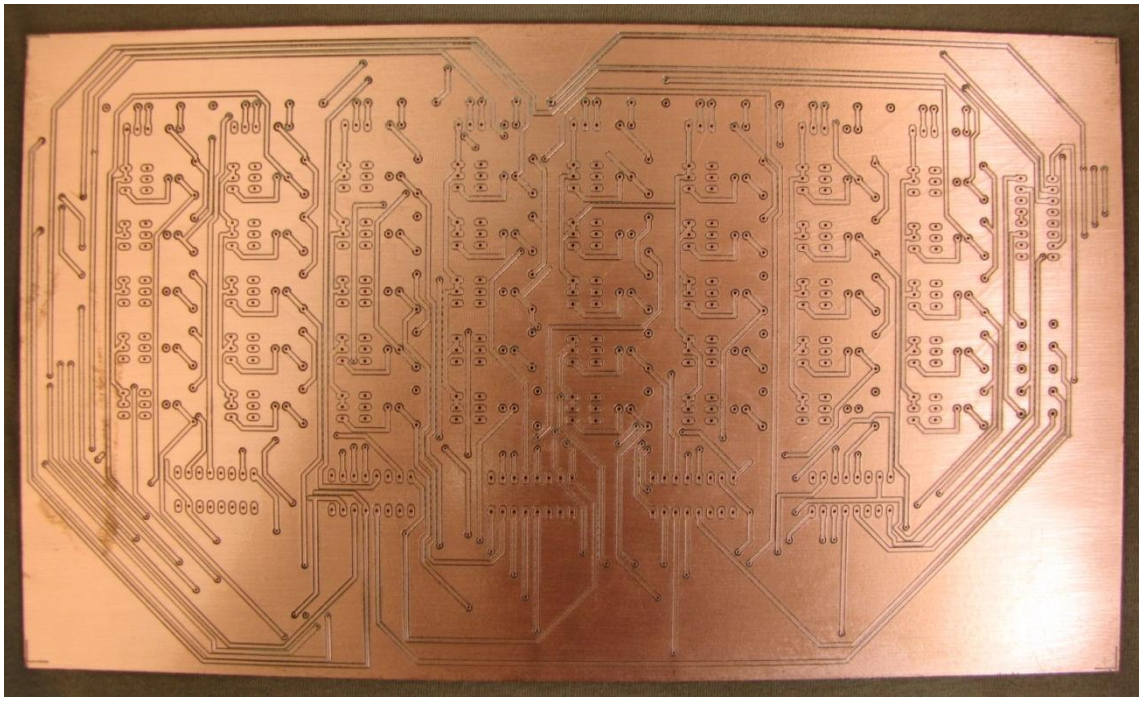


**Figura A.7 – PCB para montagem do teclado**