

2 de Maio de 2011

O Maestro

Projecto e Seminário 2010/2011

Relatório Final

*Instituto Superior de Engenharia de Lisboa*

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

**Discentes:**  
Ana Correia, 31831

* Email : a31831@alunos.isel.pt
* Telefone: 918750435

Diogo Cardoso, 32466

* Email: a32466@alunos.isel.pt
* Telefone: 913288292

**Discentes:**  
Ana Correia, 31831

* Email : a31831@alunos.isel.pt
* Telefone: 918750435

Diogo Cardoso, 32466

* Email: a32466@alunos.isel.pt
* Telefone: 913288292

.

**Orientadores:**  
Pedro Sampaio

* Email : psampaio@cc.isel.ipl.pt

Artur Ferreira

* Email: arturj@ isel.pt

..................

Índice

[1. Introdução 5](#_Toc296182005)

[1.1 Objectivos e Descrição do Projecto 5](#_Toc296182006)

[1.2 Análise de Recursos 6](#_Toc296182007)

[1.3 Organização do documento 7](#_Toc296182008)

[2. Algoritmo de Goertzel 8](#_Toc296182009)

[2.1 Introdução 8](#_Toc296182010)

[2.2 Descrição 9](#_Toc296182011)

[2.3 Características 11](#_Toc296182012)

[3. Infra-estrutura de processamento de sinal 12](#_Toc296182013)

[3.1 Introdução 12](#_Toc296182014)

[3.2 Implementação do Algoritmo 12](#_Toc296182015)

[3.3 Instrumento de estudo 13](#_Toc296182016)

[3.4 Testes ao algoritmo de Goertzel 14](#_Toc296182017)

[3.4.1 Introdução 14](#_Toc296182018)

[3.4.2 Preparação 14](#_Toc296182019)

[3.4.3 Descrição e Resultados 14](#_Toc296182020)

[3.5 Tratamento da Resolução do Goertzel 16](#_Toc296182021)

[3.6 Filtragem do sinal 18](#_Toc296182022)

[3.6.1 Filtros *FIR* 19](#_Toc296182023)

[3.7 Controlador dos filtros Goertzel 20](#_Toc296182024)

[3.7 Tempos absolutos de Processamento 21](#_Toc296182025)

[4. Conclusões 22](#_Toc296182026)

[4.1 Goertzel vs Transformada de Fourier (FFT) 22](#_Toc296182027)

[4.2 Portabilidade 24](#_Toc296182028)

[4.3 Trabalho Futuro 24](#_Toc296182029)

[4.3.1 Planeamento 25](#_Toc296182030)

[Referências 26](#_Toc296182031)

**Índice de Figuras**

[Figura 1 - Funcionamento do Maestro. 5](file:///C:\Users\Sorcha\Desktop\relatóriointercalar3183132466VFina.docx#_Toc292127097)

[Figura 2- Arquitectura de *Software* do Projecto 7](file:///C:\Users\Sorcha\Desktop\relatóriointercalar3183132466VFina.docx#_Toc292127098)

[Figura 3 - Esquema de um filtro de *Goertzel.* 9](file:///C:\Users\Sorcha\Desktop\relatóriointercalar3183132466VFina.docx#_Toc292127099)

[Figura 4 - Máquina de Estados de um filtro de *Goertzel*. 14](file:///C:\Users\Sorcha\Desktop\relatóriointercalar3183132466VFina.docx#_Toc292127100)

[Figura 5 - Diagrama de blocos do processamento de sinal. 20](file:///C:\Users\Sorcha\Desktop\relatóriointercalar3183132466VFina.docx#_Toc292127101)

[Figura 6 - Funcionamento do *Goertzel Controller*. 20](file:///C:\Users\Sorcha\Desktop\relatóriointercalar3183132466VFina.docx#_Toc292127102)

[Figura 7 - Planeamento 24](#_Toc292127103)

**Índice de Tabelas**

[Tabela 1- Frequências e diferenças entre frequências(D.C.A) de um piano. 15](#_Toc292127123)

[Tabela 2 - Resultados do teste com sinais compostos por múltiplas sinusoides 17](#_Toc292127124)

[Tabela 3 - Algumas frequências da Tabela 1. 18](#_Toc292127125)

[Tabela 4 - Valores de N e das frequências de amostragem para as frequências do piano 19](#_Toc292127126)

[Tabela 5 - Resultado do calculo do tempo de processamento do algoritmo de *Goertzel*. 21](#_Toc292127127)

# 1. Introdução

Este documento consiste no relatório de progresso do projecto O Maestro. Aborda-se essencialmente a componente de processamento de sinal desenvolvida até ao momento. Em relação ao estabelecido na proposta de projecto, verificou-se que o tempo de estudo e implementação necessário para realizar o algoritmo de Goertzel foi subdimensionado. Assim para ser possível implementá-lo e utilizá-lo para a detecção de frequências, foi necessário despender mais tempo de desenvolvimento do que o previsto; este atraso deve-se maioritariamente à falta de conhecimento dos elementos do grupo na área de processamento de sinal.

## 1.1 Objectivos e Descrição do Projecto

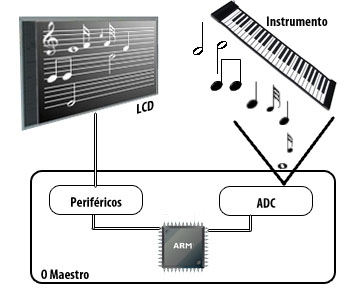
A ilustra o diagrama de blocos dos elementos do projecto e a interacção entre os mesmos.

Figura - Funcionamento do Maestro.

*O Maestro,* será um sistema dedicado sobre a arquitectura *Advance Risk Machine* (*ARM7TDMI)* [[1](#ARM11)] que tratará de obter notas musicais produzidas por determinado instrumento e apresentá-las sob a forma de uma pauta musical. Para a recolha de amostras será utilizado o *Analog to Digital Converter* (*ADC)* associado ao microcontrolador. Para o *input* e *output* irá ser usado um *Liquid Crystal Display* (*LCD*) gráfico *touch screen* como ilustra a .   
A componente de software deste projecto está dividida em três camadas, tal como se apresenta na :

1. *Hardware*, responsável por interagir directamente com os periféricos internos e externos do microcontrolador.
2. Abstracção ao *hardware*, responsável por definir a ponte entre a camada aplicacional e o *hardware*.
3. Aplicacional, responsável pelo controlo do *input* e *output* do utilizador, gestão da aplicação e ainda é a camada onde o algoritmo de *Goertzel* será implementado.

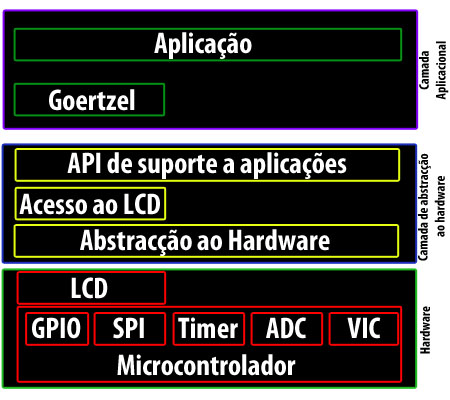
****

Figura - Arquitectura de *Software* do Projecto

## 1.2 Análise de Recursos

Após a análise dos requisitos do projecto, constatou-se que os problemas mais relevantes são a recolha e processamento das amostras de som. As frequências que se pretende captar e processar estão na banda de 27 Hz a 4186 Hz. Assim, é necessário, respeitando o teorema de *Nyquist* [[2](#1)]*,* no mínimo utilizar uma frequência de amostragem superior a 8372 Hz. O *ADC* funciona com 10 *bits* por amostra num intervalo de amplitude de 0 a 3 V, com frequência de amostragem até 400 kHz logo é uma solução adequada para a banda de frequência que se pretende processar.

Para a implementação do projecto vão ser utilizados os seguintes recursos:

* Microcontrolador baseado na arquitectura ARM7TDMI - LPC2294 da NXP [[3](#Kei11)].
* *LCD* R*G*B gráfico (320x240 *pixels*) com *touch screen*.
* Ferramentas open-source da GNU para desenvolvimento sobre a arquitectura ARM7TDMI.
* O periférico *ADC* do Microcontrolador LCP2294.

## 1.3 Organização do documento

Este documento está dividido em 4 secções.

Na secção 2 consta a descrição do algoritmo de *Goertzel* [[2](#1)] e as motivações para a escolha deste algoritmo para o projecto .

Na secção 3 apresenta-se o trabalho realizado até ao momento, nomeadamente a implementação do algoritmo de *Goertzel*, a resolução para problemas detectados nos testes realizados sobre o algoritmo.

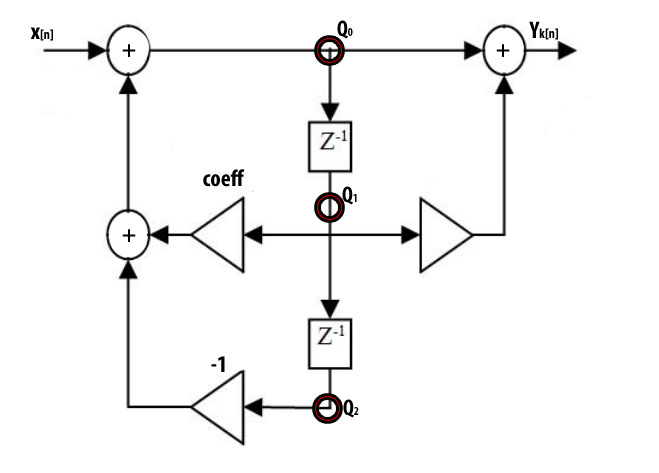
Por fim a secção 4 contém as conclusões do trabalho realizado até ao momento, bem como o trabalho futuro do projecto.

# 2. Algoritmo de Goertzel

## 2.1 Introdução

O algoritmo de *Goertzel* foi criado por Gerald Goertzel em 1958. Este algoritmo calcula um coeficiente da transformada discreta de Fourier (*DFT* – *Discrete Fourier Transform*) através de um filtro recursivo [[4](#Rob01)]. Existem várias versões do algoritmo; neste documento trata-se uma versão optimizada que não tira partido de operações complexas para a detecção de frequências.

Figura - Esquema de um filtro de *Goertzel.*

A ilustra o diagrama de blocos do filtro de *Goertzel*.

## 2.2 Descrição

O algoritmo de *Goertzel* [[5](#Gen1)] [[6](#MarcadorPosição1)] detecta a presença de uma dada frequência através de amostragem do espectro do sinal nessa frequência. Calculado o valor do módulo do espectro de amplitude numa dada frequência e comparando-o com a energia total é possível verificar quanto é que a frequência contribui para a energia do sinal. Quanto menor a diferença entre a energia do sinal e a energia da frequência, maior é a contribuição da frequência para o sinal. Assim, definindo um limite nesta diferença é possível avaliar se uma frequência se encontra ou não presente no sinal [[7](#Gen111)].

O algoritmo é composto pelos seguintes componentes:

* Um coeficiente .
* Uma constante *k* que representa a frequência que se pretende detectar.
* O valor da frequência de amostragem *Fs.*
* O valor da frequência que se pretende detectar, *Fn.*
* O numero de amostras do sinal que irão ser processadas, *N*.

O valor do coeficiente e da constante *k* são calculados pelas seguintes expressões:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

A constante *k* tem o valor inteiro mais próximo resultante do arredondamento do resultado da equação (2).

A partir da pode deduzir-se a seguinte equação:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

A equação representa a evolução dos valores das unidades de atraso intermédias à medida que as N amostras "circulam" pelo filtro. O filtro guarda apenas os últimos dois estados intermédios para os usar posteriormente na geração de um novo.

Após o processamento de todos os elementos das *N* amostras o algoritmo de *Goertzel* retorna um valor de energia relativa através da equação .

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

Na realidade o algoritmo de *Goertzel* não retorna a energia total do espectro da frequência, isto é, este só retorna o valor da energia da componente positiva do espectro. Sendo assim é necessário multiplicar por dois para obter a energia total da frequência no espectro (5).

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

Para saber se uma dada frequência está presente no sinal é necessário comparar a energia total do sinal com a energia relativa da frequência assim é necessário calcular essa energia relativa com a equação (6):

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

|  |  |  |
| --- | --- | --- |
|  |  |  |

## 2.3 Características

O algoritmo optimizado de *Goertzel* não usa operações complexas, e como consequência a sua complexidade aritmética é reduzida necessitando apenas de multiplicações e adições, sendo *N* o números de amostras do sinal na entrada do filtro.

Em memória, em cada instante o algoritmo apenas necessita de ter a amostra actual e os valores intermédios , e , podendo ter em memória não volátil os valores de k e coeficientes.

Outra característica do Goertzel é este ser paralelizável uma vez que cada filtro é independente de outros que possam existir, podendo assim detectar várias frequências simultaneamente. Através de um banco de filtros de Goertzel, é possível detectar simultaneamente a presença de várias frequências.

A resolução em frequência do algoritmo é dada pela equação (7).

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

A resolução em frequência é o intervalo entre duas frequências detectáveis, ou seja, se tivermos duas frequências *a* e *b* para detectar, a diferença entre estas deve ser maior do que o valor da resolução (8).

|  |  |  |
| --- | --- | --- |
|  | para *b > a* | (8) |

Qualquer frequência entre o intervalo ] *a , b* [ que se pretenda detectar irá ser falsamente detectada sempre que *a* ou *b* estejam presentes no sinal.

Concluindo, todos os factores referidos anteriormente tornam o algoritmo de *Goertzel* bastante eficiente, escalável e implementado com pouca memória, tornando-o portável a qualquer tipo de arquitectura.

# 3. Infra-estrutura de processamento de sinal

## 3.1 Introdução

Após o estudo e análise do algoritmo de *Goertzel* escolheu-se a linguagem C++ como ferramenta de implementação. A escolha desta deveu-se mais uma vez ao factor de portabilidade de código para diferentes plataformas de hardware.

## 3.2 Implementação do Algoritmo

A representa a máquina de estados da implementação do algoritmo de *Goertzel*.

Figura - Máquina de Estados de um filtro de *Goertzel*. O estado "Calcular energia relativa" refere-se à equação

Como referido anteriormente (Equação e ) este algoritmo é recursivo e necessita de apenas três variáveis locais (*Q0*, *Q1* e *Q2*) para calcular o módulo do espectro de amplitude da frequência que se deseja detectar.

Durante a implementação do algoritmo teve-se de ter em conta a representação numérica das amostras, uma vez que estas deveriam ser o mais próximo possível dos cálculos teóricos. Com este factor em mente foram realizadas duas implementações, uma com valores inteiros e outra com valores decimais (*floating-point*). (mais a frente iremos falar das diferenças e opcoes)

## 3.3 Instrumento de estudo

Para testar e analisar o algoritmo de *Goertzel* foi necessário escolher um instrumento. Esta escolha foi realizada tendo em conta os seguintes critérios:

* Ter uma largura de banda elevada, para ser possível aplicar o algoritmo para frequências altas e baixas.
* Produzir frequências próximas, para testar a precisão e resolução do algoritmo.

Com estes critérios escolheu-se o **piano** como instrumento de teste e as suas frequências encontram-se representadas na .

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Frequências | D.C.A | Frequências | D.C.A | Frequências | D.C.A | Frequências | D.C.A |
| 27,5000 | - - - | **97,9989** | 5,5003 | **349,2280** | 19,6000 | **1244,5100** | 69,8500 |
| 29,1352 | 1,6352 | **103,8260** | 5,8271 | **369,9940** | 20,7660 | **1318,5100** | 74,0000 |
| 30,8677 | 1,7325 | **110,0000** | 6,1740 | **391,9950** | 22,0010 | **1396,9100** | 78,4000 |
| 32,7032 | 1,8355 | **116,5410** | 6,5410 | **415,3050** | 23,3100 | **1479,9800** | 83,0700 |
| 34,6478 | 1,9446 | **123,4710** | 6,9300 | **440,0000** | 24,6950 | **1567,9800** | 88,0000 |
| 36,7081 | 2,0603 | **130,8130** | 7,3420 | **466,1640** | 26,1640 | **1661,2200** | 93,2400 |
| 38,8909 | 2,1828 | **138,5910** | 7,7780 | **493,8830** | 27,7190 | **1760,0000** | 98,7800 |
| 41,2034 | 2,3125 | **146,8320** | 8,2410 | **523,2510** | 29,3680 | **1864,6600** | 104,6600 |
| 43,6535 | 2,4501 | **155,5630** | 8,7310 | **554,3650** | 31,1140 | **1975,5300** | 110,8700 |
| 46,2493 | 2,5958 | **164,8140** | 9,2510 | **587,3300** | 32,9650 | **2093,0000** | 117,4700 |
| 48,9994 | 2,7501 | **174,6140** | 9,8000 | **622,2540** | 34,9240 | **2217,4600** | 124,4600 |
| 51,9131 | 2,9137 | **184,9970** | 10,3830 | **659,2550** | 37,0010 | **2349,3200** | 131,8600 |
| 55,0000 | 3,0869 | **195,9980** | 11,0010 | **698,4560** | 39,2010 | **2489,0200** | 139,7000 |
| 58,2705 | 3,2705 | **207,6520** | 11,6540 | **739,9890** | 41,5330 | **2637,0200** | 148,0000 |
| 61,7354 | 3,4649 | **220,0000** | 12,3480 | **783,9910** | 44,0020 | **2793,8300** | 156,8100 |
| 65,4064 | 3,6710 | **233,0820** | 13,0820 | **830,6090** | 46,6180 | **2959,9600** | 166,1300 |
| 69,2957 | 3,8893 | **246,9420** | 13,8600 | **880,0000** | 49,3910 | **3135,9600** | 176,0000 |
| 73,4162 | 4,1205 | **261,6260** | 14,6840 | **932,3280** | 52,3280 | **3322,4400** | 186,4800 |
| 77,7817 | 4,3655 | **277,1830** | 15,5570 | **987,7670** | 55,4390 | **3520,0000** | 197,5600 |
| 82,4069 | 4,6252 | **293,6650** | 16,4820 | **1046,5000** | 58,7330 | **3729,3100** | 209,3100 |
| 87,3071 | 4,9002 | **311,1270** | 17,4620 | **1108,7300** | 62,2300 | **3951,0700** | 221,7600 |
| 92,4986 | 5,1915 | **329,6280** | 18,5010 | **1174,6600** | 65,9300 | **4186,0100** | 234,9400 |

Tabela - Frequências e diferenças entre frequências(D.C.A) de um piano.

## 3.4 Testes ao algoritmo de Goertzel

### 3.4.1 Introdução

Após a implementação do algoritmo de *Goertzel* foi necessário testá-lo em factores como o funcionamento básico do algoritmo, nomeadamente, como é que programaticamente se identifica a existência de uma frequência presente num determinado sinal. Na sequência dos testes iniciais foram detectados outros problemas não previstos até ao momento, concretamente o comportamento do algoritmo de *Goertzel* quando se está a detectar duas frequências tais que a diferença entre as mesmas é reduzida; de que tipo deveria ser utilizado para representar as amostras (inteiro, decimal); o comportamento do algoritmo quando duas ou mais frequências têm o mesmo valor de coeficiente .

### 3.4.2 Preparação

Para estes testes foi realizado um módulo de criação de sinusóides, no qual estas são criadas com os seguintes requisitos:

* Frequência de Amostragem(*Fs*) = 8800 kHz
* Amplitude(*A*) = 1000
* Frequência (*fo*) é passada como parâmetro

O cálculo das amostras da sinusóide é efectuado de acordo com a equação )

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

### 3.4.3 Descrição e Resultados

Os testes iniciais consistiram em a criar várias sinusóides com diferentes frequências, passá-las a um filtro de *Goertzel* e verificar se as frequências estavam presentes.

Após se verificar que o algoritmo estava devidamente implementado, uma vez que produziu resultados válidos para os testes iniciais, geraram-se sinais compostos por sinusóides de várias frequências, instanciou-se um filtro de *Goertzel* para cada frequência e aplicou-se o sinal a cada um dos filtros. A mostra os resultados obtidos neste teste.

|  |  |  |
| --- | --- | --- |
| Frequências(Hz) | Percentagem do sinal (%) | Encontrada |
| 55 | 21.8 | Sim |
| 110 | 22 | Sim |
| 440 | 15.1 | Sim |
| 880 | 15.6 | Sim |
| 1760 | 15.9 | Sim |
| 3520 | 16 | Sim |

Tabela - Resultados do teste com sinais compostos por múltiplas sinusoides

A **Percentagem do Sinal** representa a contribuição da frequência para o sinal.

Como se pode verificar nos resultados obtidos as frequências foram encontradas mas durante o teste detectaram-se problemas com a resolução em frequência do algoritmo (tal como apresentado na subsecção ), uma vez que ao testar o algoritmo com frequências com um intervalo curto este retornava a indicação da presença de frequências que na realidade não estavam presentes. O mesmo aconteceu a frequências com os mesmos coeficientes, por exemplo, existe uma frequência presente no sinal com um coeficiente *x* se existir outra frequência com o mesmo coeficiente o algoritmo irá dar falsos positivos uma vez que a única informação (relativa directamente à frequência) que o algoritmo usa para detectar se uma frequência está presente no sinal é o seu coeficiente.

Por fim foi ainda verificado que ao utilizar valores inteiros para a representação das amostras do sinal, os resultados são próximos dos resultados teóricos do algoritmo de *Goertzel*.

## 3.5 Tratamento da Resolução do Goertzel

Como foi referido nos resultados dos testes do algoritmo de *Goertzel*, as frequências cujo intervalo entre elas seja curto, o *Goertzel* considera falsamente que existem no sinal. Esta diferença entre valores de frequências designa-se por resolução em frequência sendo calculada com a equação (7).

Por exemplo, para os valores de e temos um . Isso significa que caso se queira detectar uma frequência com o valor de 440 Hz e que esta se encontre numa dada amostra, o algoritmo de *Goertzel* irá falsamente indicar que as frequências dentro do intervalo se encontram presentes no sinal, introduzindo assim um erro significativo ao processamento das amostras.

A solução ideal seria que o valor de fosse inferior a qualquer diferença entre frequências que se pretende detectar. Na encontram-se exemplos de algumas frequências que se pretende detectar e a diferença entre as mesmas. Para baixas frequências, a necessidade de ter resolução detalhada leva a que tenha que ser utilizado um número elevado de pontos *N*.

|  |  |
| --- | --- |
| Frequência | Diferença com a anterior |
| 27,5000 | - - - |
| 29,1352 | 1,6352 |
| 30,8677 | 1,7325 |
| 32,7032 | 1,8355 |
| 34,6478 | 1,9446 |
| ... | ... |
| 3520,0000 | 197,5600 |
| 3729,3100 | 209,3100 |
| 3951,0700 | 221,7600 |
| 4186,0100 | 234,9400 |

Tabela - Algumas frequências da Tabela 1.

Como ilustrado na , as diferenças entre as frequências são crescentes e enquanto que a resolução anteriormente calculada era adequada para as frequências superiores a 3000 Hz não o era para as frequências inferiores a 740 Hz. Assim, foi necessário fazer ajustes de modo a que a resolução nunca seja superior à diferença entre duas frequências consecutivas.

A solução mais intuitiva seria aumentar o divisor da equação , o *N*, para um valor mais próximo de *Fs*, por exemplo com um , o valor de seria 1 sendo inferior a todas as diferenças de frequências. O problema desta solução é que se aumentava consideravelmente o tempo de processamento do algoritmo aumentando igualmente a latência e diminuindo o tempo de resposta aos consumidores do processamento de sinal.

A segunda solução não tão evidente seria diminuir o valor de *Fs*, diminuindo assim também o valor de . A consequência desta solução seria que ao diminuir a frequência de amostragem estaria-se a diminuir o intervalo de frequências possíveis de serem detectadas, pelo teorema de *Nyquist*.

No final a solução adoptada foi um misto das duas anteriores, a frequência de amostragem fica constante para que seja possível capturar a gama de frequências que se pretende, mas existe uma divisão desta realizada por software. Por exemplo para as primeiras frequências da o seu processamento será realizado com um e com um . Imaginando que existe um array de *N* posições onde são guardadas as amostras com uma frequência de amostragem de 8800 Hz, para que os dados sejam processados com um *Fs* de 275 Hz bastará que a indexação a esse *array* seja realizada com saltos de 32 posições uma vez que, .

|  |  |  |
| --- | --- | --- |
| ***Gama(Hz)*** | ***Fs (Hz)*** | ***N*** |
| **25,7 - 61,7354** | 275 | 200 |
| **65,4064 - 146,832** | 550 | 200 |
| **155,563 - 349,228** | 1100 | 200 |
| **369,994 - 830,609** | 2200 | 200 |
| **880 - 1975,53** | 8800 | 200 |
| **2093 - 4186,01** | 8800 | 100 |

Tabela - Valores de N e das frequências de amostragem para as frequências do piano.

Com esta solução construiu-se uma aplicação utilitária que tem como funcionalidade calcular os valores de *Fs* e *N* óptimos para capturar uma dada gama de frequências. Na encontra-se o resultado da execução da aplicação referida anteriormente.

## 3.6 Filtragem do sinal

Como descrito nos resultados dos testes ao algoritmo de Goertzel, o facto deste algoritmo apenas de basear no valor do coeficiente para detectar a presença de uma frequência num dado sinal, inviabiliza que existam frequências com o coeficiente igual, o problema é que ao resolver-mos o problema da resolução do algoritmo, agravou-se este problema ainda mais, uma vez que a probabilidade de existirem duas ou mais frequências com o mesmo coeficiente é alta, já que as frequências estão divididas em blocos com frequências de amostragem diferentes e valor de *N* diferentes. A demonstra alguns exemplos deste problema:

|  |  |
| --- | --- |
| Frequências(Hz) | Coeficiente |
| 110; 220; 440; 1760 | 0,61803 |
| 466,164; 116,54; 233,082 | 0,4743 |
| 293,665; 587,33; 2349,32 | -0,21282 |

Tabela - Exemplos de frequências com o mesmo coeficiente.

Uma vez que o problema está com os coeficientes a solução mais directa seria modular estes coeficientes até que todos os valores fossem diferentes, o problema desta solução é a complexidade de arranjar coeficientes diferentes para todas as 88 frequências quando estas não partilham valores de frequências de amostragem nem de *N*. A solução terá de ser algo exterior ao algoritmo e a sua configuração, portanto optou-se por uma filtragem de sinal.

Esta filtragem irá ser realizada para cada gama de frequências () de tal maneira a que as amostras passadas ao algoritmo de Goertzel estejam filtradas antes deste efectuar a verificação, evitando assim as falsas detecções.

### 3.6.1 Filtros *FIR*



Figura - Diagrama de blocos de um filtro *FIR*

Para filtrar as amostras foram utilizados filtros do tipo *FIR* (*Finite Impulse Response*), o seu funcionamento está ilustrado na .

|  |  |
| --- | --- |
|  | () |

A equação (10) representa as amostras filtradas, esta equação evidencia que o numero de amostras atrasadas presentes no filtro é o mesmo que o numero total de coeficientes .

Os coeficientes são calculados a partir da resposta impulsional de um filtro passa-banda, que por sua vez é calculado com a diferença da resposta impulsional de dois filtros passa-baixo como demonstra a equação (11).

|  |  |
| --- | --- |
|  | () |

Os valores de são dados pela equação:

|  |  |
| --- | --- |
|  | () |

A componente representa a frequência normalizada e é calculada com a seguinte expressão:

|  |  |
| --- | --- |
|  | () |

Foi ainda aplicada a janela Hamming [[**8**](#III11)] sobre a resposta impulsional do filtro de emaneira a minimizar o ganho das frequências próximas das frequências de corte.

(colocar historia de por os filtros com ganho de 1)

## 3.7 Controlador dos filtros Goertzel

A representa o *pipeline* de processamento de sinal utilizando o algoritmo de *Goertzel*.

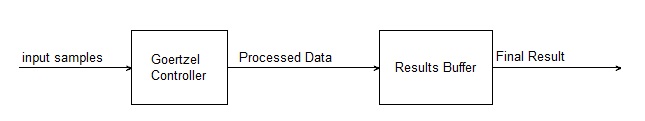
 Este tem apenas dois processos de manipulação de dados, o primeiro *Goertzel Controller* é responsável por analisar o sinal e referir que frequências estão presentes e o segundo, *Results Buffer* que será onde os resultados serão momentaneamente guardados de maneira a que seja possível estender futuramente as operações de processamento.

Figura - Diagrama de blocos do processamento de sinal.

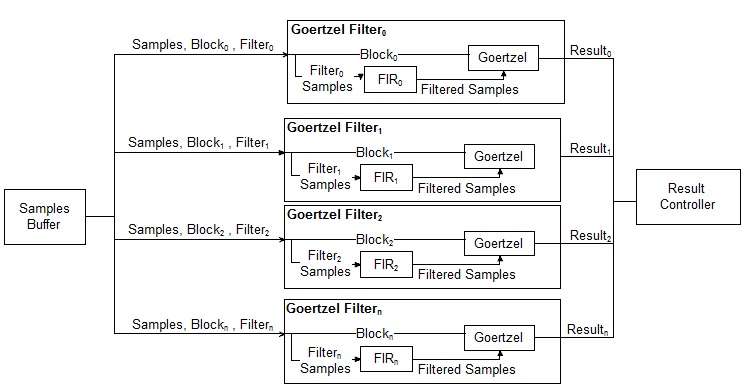
 Uma vez que o algoritmo de *Goertzel* é paralelizável criou-se uma infra-estrutura que tirasse partido dessa característica, de modo a controlar os seus filtros e armazenar os resultados como ilustra a .

Figura - Funcionamento do *Goertzel Controller*.

O Goertzel Controller é constituído por dois elementos chave, **o *buffer* de amostras** que além de salvaguardar as amostras, vai calculando a sua energia à medida que amostras são fornecidas, tirando assim a necessidade de percorrer todo o buffer de uma só vez, bem como problemas de concorrência para o acesso ao valor da energia do sinal de entrada.

O segundo elemento do *controller* são os filtros de Goertzel, estes são responsáveis por filtrar o sinal para a sua gama de frequências de produzir os resultados. Este filtro usa a mesma estratégia utilizada no buffer de ir fazendo trabalho à medida que seja possível para evitar tempo a iterar longos *arrays.* Uma particularidade destes filtros é que estes só invocam o Goertzel caso exista evidencias que alguma das frequências da gama que está a processar se encontra nas amostras, para tal este vai guardando a energia do sinal filtrado à medida que filtra as amostras, assim quando a condição para chamar o Goertzel (ter amostras suficientes para o fazer) esteja cumprida é comparado o valor da energia do sinal (calculado pelo *buffer*) com a energia do sinal filtrado (calculado no filtro) e o Goertzel é apenas invocado quando a energia do sinal filtrado esteja sobre um limiar configurável.

Uma das particularidades deste controlo é que os filtros podem produzir resultados em alturas diferentes, por exemplo enquanto que a frequência de 4186.01Hz necessita apenas de 179 amostras para ser possível saber se está presente no sinal ou não, a frequência 55Hz necessita de 2704 amostras, isto deve-se à resolução do algoritmo de Goertzel referida anteriormente, assim o controlador apenas "liberta" os resultados quando todos os filtros tenham acabado todo o seu processamento.

## 3.7 Tempos absolutos de Processamento

Um dos aspectos mais importantes na escolha de um algoritmo de processamento de sinal é o seu tempo de processamento. Para averiguar esses tempos foram realizados alguns testes que determinam quanto tempo demoraria o *Goertzel* a calcular se alguma das 88 frequências está presente numa dada amostra; apresentam-se os respectivos resultados na .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Tempo Total(ns) | Tempo Relativo(ns) | Arquitectura | Processador(GHz) |
| 200 | 6022000000 | 602200 | x86 | I5 2.4 |
| 200 | 3260000000 | 326000 | x64 | I7 1.6 |
| 200 | 2792000000 | 279200 | x64 | I5 2.8 |
| 2000 | 54117000000 | 5411700 | x86 | I5 2.4 |
| 2000 | 30967000000 | 3096700 | x64 | I7 1.6 |
| 2000 | 24585000000 | 2458500 | x64 | I5 2.8 |

Tabela - Resultado do calculo do tempo de processamento do algoritmo de *Goertzel*.

O **tempo total** representa o tempo de execução do teste; o **tempo relativo** representa quanto tempo demora o algoritmo de Goertzel a iterar cada uma das 88 frequências de tal forma a testar se esta está presente ou não; a frequência de amostragem utilizada para estes testes foi de 8800 Hz.

Os testes consistiram maioritariamente em dar uma amostra à implementação do algoritmo e esperar pelos resultados da detecção de frequência.

Os testes foram executados em várias arquitecturas com diferentes processadores de tal forma a que fosse possível visualizar como é que o algoritmo se comportava com diferentes capacidades de processamento.

# 4. Conclusões

Neste ponto de situação, conclui-se o bloco de processo de processamento de sinal (o subloco *Goertzel* na camada aplicacional da ). Com este bloco definiu-se *à priori* alguma da interface pública da API da camada de abstracção ao hardware ().

## 4.1 Goertzel vs Transformada de Fourier (FFT)

Quando é necessário resolver um problema que envolva detecção de frequências, normalmente, a primeira abordagem a tomar é usar a *Fast Fourier Transform* (*FFT*). Tanto a *FFT* como o algoritmo de *Goertzel* operam sobre vectores de *N* amostras. A *FFT* e o algoritmo de *Goertzel* diferenciam-se pelo facto de a *FFT* conseguir de uma só vez detectar várias frequências porque produz um vector com *N* coeficientes espectrais, enquanto que para cada filtro de *Goertzel* apenas é possível detectar a presença de uma frequência. Esta diferença reflecte-se nas diferentes complexidades computacionais destes dois algoritmos. Para um sinal com *N* amostras, o cálculo da sua *FFT* envolve a execução de *Nlog2(N)* multiplicações complexas.

Uma das razões de escolha do algoritmo de *Goertzel* em vez da *FFT* deve-se ao facto de esta necessitar uma quantidade substancial de memória, detectar todas as frequências numa dada largura de banda, ter uma elevada complexidade aritmética tornando-a mais lenta e pelo uso de valores decimais é menos portável do que o algoritmo de *Goertzel*. A outra razão é o facto de ser possível ter diferentes filtros de *Goertzel* em execução com diferentes configurações (*N* e *Fs*), podendo assim optimizar ainda mais o processamento das frequências.

## 4.2 Portabilidade

Durante toda a fase de desenvolvimento o objectivo principal sempre foi a portabilidade do código. Foram implementadas e testadas duas versões do Goertzel, uma com valores inteiros e outra com valores decimais. Embora os resultados de detecção de frequência tenham sido satisfatórios, a portabilidade fica sempre comprometida com a precisão do dispositivo de digitalização de sinal nomeadamente no que se refere à frequência de amostragem e número de bits por amostra. Assim, as duas implementações referidas acima tiveram em conta estes aspectos.

## 4.3 Trabalho Futuro

Futuramente irá ser realizada a familiarização com o hardware, nomeadamente as implementações dos *drivers* dos periféricos*.* Posteriormente irá ser portada toda a infra-estrutura implementada do algoritmo de *Goertzel.*

Em seguida, irá ser montada uma infra-estrutura de abstracção ao hardware de modo a abstrair a aplicação *O Maestro* do *target* onde se encontra a ser executada. Os principais componentes da infra-estrutura são:

* *Input* e *Output*, sendo o *input* neste caso específico um *ADC* e um *LCD* gráfico *touch*, para *output* o mesmo *LCD* utilizado para *input*.
* Suporte para multi-tarefa de forma a que seja possível executar várias instâncias de *Goertzel* simultaneamente.

Todas as fases anteriores terão um período continuado de testes, de forma a garantir um bom funcionamento.

Quando toda a infra-estrutura estiver concluída, prosseguirá-se à implementação da aplicação, com as devidas fases de teste e documentação.

### 4.3.1 Planeamento

A Figura 7 demonstra o planeamento do restante tempo para finalização do projecto.

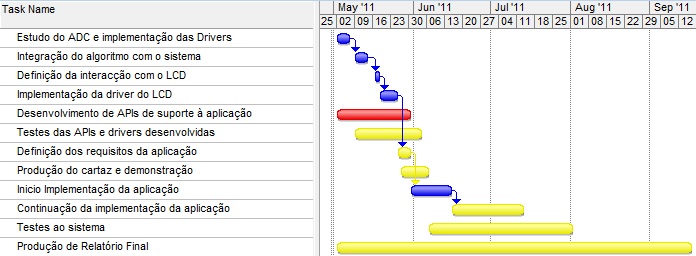


Figura 8 - Planeamento

**Legenda:**

 Diogo Cardoso

 Ana Correia

**** Ambos

# Referências

x

|  |  |
| --- | --- |
| [1] | ARM. The Architecture for the Digital World. [Online]. <http://www.arm.com> |
| [2] | R. Schafer A. Oppenheim, *Discrete-Time Signal Processing 2nd edition*.: Prentice Hall , 1999. |
| [3] | Keil. LPC2294 User Manual. [Online]. <http://www.keil.com/dd/docs/datashts/philips/user_manual_lpc2119_2129_2194_2292_2294.pdf> |
| [4] | Andrew G. Dempster, Izzet Kale Robert Beck, "Finite-Precision Goertzel Filters Used for Signal," vol. VOL. 48, no. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING, 2001. |
| [5] | Kevin Banks. The Goertzel Algorithm. [Online]. <http://www.eetimes.com/design/embedded/4024443/The-Goertzel-Algorithm> |
| [6] | R. Schafer A. Oppenheim, *Discrete-Time Signal Processing 2nd edition*.: Prentice Hall, 1999. |
| [7] | Gene Small. Detecting CTCSS tones with Goertzel's algorithm. [Online]. <http://www.eetimes.com/design/embedded/4025660/Detecting-CTCSS-tones-> |
| [8] | Julius O. Smith III. (2011, Junho) Center for Computer Research in Music and Acoustics. [Online]. <https://ccrma.stanford.edu/~jos/sasp/Hamming_Window.html> |

x

# 