Universidade do Minho
Escola de Engenharia
Departamento de Electrónica Industrial

Sistema de Reconhecimento de Apito em Ambientes Ruidosos

Paulo Ricardo Neves Carvalho

Dissertação submetida à Universidade do Minho para obtenção do grau de Mestre em Electrónica Industrial e Computadores

Junho 2008

Dissertação realizada sob a orientação cientifica do
Professor António Fernando Macedo Ribeiro
Professor associado do Departamento de Electrónica
Industrial da Universidade do Minho

Ninguém é tão ignorante que não tenha algo a ensinar. Ninguém é tão sábio que não tenha algo a aprender.

Autor: Blaise Pascal

Resumo

Num jogo de futebol robótico os robôs futebolistas normalmente recebem os comandos do árbitro através de sistemas de comunicação sem fios. Contudo, um dos objectivos que as equipas de futebol robótico possuem é de colocar os robôs a interpretar directamente o apito do árbitro. Este documento descreve o desenvolvimento de um sistema electrónico que faz esta tarefa, chamado Sistema de Reconhecimento de Apito em Ambientes Ruidosos.

O Sistema de Reconhecimento de Apito em Ambientes Ruidosos tem como principal objectivo reconhecer um apito, contudo existem outros objectivos que constam no desenvolvimento do projecto, o sistema deve ter a máxima imunidade ao ruído possível, deve ser pouco volumoso de modo a poder ser usado em qualquer tipo de robô futebolista, deve também conseguir distinguir 2 apitos diferentes, e ser ajustável ao apito e nível de ruído envolvente.

A solução para este sistema passa pelo uso de microcontroladores, onde se adquire o sinal sonoro através de um Conversor Analógico-Digital (ADC) e se faz determinado processamento verificando-se então se o sinal sonoro corresponde ao sinal sonoro de um apito. Para fazer chegar o sinal sonoro à entrada do ADC, é necessário transformar o sinal sonoro num sinal eléctrico, esta tarefa faz-se com um microfone e respectivo pré-amplificador, estes elementos compõem o hardware do sistema.

O processamento de sinal para verificar se o sinal é de apito ou não, consiste em calcular uma Transformada Rápida de Fourier (FFT) para se obter o espectro do sinal, e consequentemente ter-se mais percepção sobre o mesmo. De seguida desenvolveu-se para este sistema um algoritmo de percepção que analisa o espectro e devolve uma decisão, apito ou não apito.

Para se poder ajustar o sistema ao apito e ao nível de ruído, o sistema possui 4 parâmetros ajustáveis. Para se ter uma boa percepção de como os parâmetros influenciam a decisão criou-se um simulador de parâmetros e criou-se também um software de interface com a porta série, para o microcontrolador enviar uma amostra do espectro para o PC, dando ao utilizador do sistema uma boa percepção do espectro que se pretende reconhecer.

Abstract

In a robotic football game, the robots receive the referee commands through a wireless communication system. However, there is a growing interest in perceiving a traditional referee whistle directly. This work describes the development of such an electronic system, called Whistle Recognition System in Noise Environments.

The Whistle Recognition System in Noise Environments has as main objective not just perceiving a whistle, but also to be as immune as possible to noise, being small so that it can be used in any robot, being able to distinguish two different whistles, and easily adjustable to different whistles and level of surrounding noise.

The solution requires the use of a microcontroller, which is used to acquire the signal through an Analog to Digital Converter (ADC) and to processe it in order to check whether the signal is from a whistle or not. In order to get the signal to the ADC input, it is necessary to transform the sound signal in an electrical signal, which is achieved through a microphone and its pre-amplifier, being these elements the hardware of the system.

The necessary signal processing consists of calculating a Fast Fourier Transform (FFT) to obtain the signal spectrum, and therefore to have more understanding of the same signal. Then, a perception algorithm was developed for this system, which analyzes the spectrum and returns a decision, whistle or not whistle.

In order to adjust the system to a specific whistle and to a certain level of noise, the system has 4 adjustable parameters. To have a good understanding of the parameters influence, a parameters simulator was developed as well as software for interfacing with the serial port. The microcontroller sends a spectrum sample to the PC, giving the system operator a good understanding of the spectrum to be recognized.

Agradecimentos

Ao meu pai Joaquim Carvalho, à minha mãe Deolinda Carvalho e à minha irmã Sofia Carvalho, por todo o apoio e força que me deram durante todo o tempo de realização do projecto.

Ao meu orientador Doutor Fernando Ribeiro, pela confiança que teve em mim em me dar a oportunidade de fazer este projecto e por toda a ajuda prestada no seu desenvolvimento.

Aos meus colegas de laboratório André Oliveira, Bruno Matos, Cristiano Santos, João Guimarães, Luís Pacheco e Sílvia Reis, pela companhia e toda a ajuda dada no dia a dia durante o trabalho no projecto. E ao colega de curso Sérgio Silva por me ter ajudado na fase de análise a ruídos e testes ao sistema final com a produção de alguns ruídos pretendidos.

Aos professores Gerardo Rocha, Carlos Lima e Jorge Cabral do Departamento de Electrónica Industrial da Universidade do Minho pela ajuda prestada no âmbito das suas áreas de conhecimento.

Índice

Índice de Figuras

Lista de Acrónimos

ADC – *Analog to Digital Converter*

ALU – *Arithmetic Logic Unit*

AMPOP – Amplificador Operacional

ATD – *Automatic Threshold Detector*

DAC – *Digital to Analog Converter*

DFT – *Discret Fourier Transform*

DSC – *Digital Signal Controller*

DSP – *Digital Signal Processor*

FFT – *Fast Fourier Transform*

HMM – *Hidden Markov Models*

LED – *Light Emitting Diode*

LSB – *Least Significant Bit*

LVQ – *Learning Vector Quantization*

MCU – *Microcontroller Unit*

MIPS – *Million Instructions Per Second*

MSB – *Most Significant Bit*

PCB – *Printed Circuit Board*

PSV – *Program Space Visibility*

SMD – *Surface Mount Device*

TTL – *Transistor Transistor Logic*

UART – *Universal Asynchronous Receiver Transmitter*

Capítulo 1 – Introdução

O desenvolvimento tecnológico tem vindo a crescer a um ritmo considerável durante as últimas décadas, permitindo o crescimento de áreas de estudo, entre as quais, a robótica, uma área onde têm sido feitos muitos desenvolvimentos. No crescimento da robótica houve sempre uma tendência em colocar características humanas nos robôs, a este facto estão associados o uso sensorial e a inteligência artificial, estes dois factores permitem ao robô imitar os sentidos humanos, dando-lhe percepção do mundo que o rodeia, e fazer uma interpretação desse mundo.

Tendo em vista a criação de sistemas que imitem os sentidos humanos, neste caso a audição, começaram a aparecer os primeiros algoritmos de reconhecimento de fala. Estes sistemas usam algoritmos baseados em teorias estatísticas e matemáticas, e requerem muito boa capacidade computacional para serem implementados, existindo alguns que usam até um método de treino que vai adaptando parâmetros de identificação da palavra para determinado orador (ex. HMM – *Hidden Markov Models*). Nos dias de hoje existem já alguns sistemas que usando estes algoritmos conseguem reconhecer fala de forma relativamente fiável. Contudo, nem todos os robôs necessitam de reconhecer fala, em alguns casos apenas se pretende usar um sistema que imite a audição para reconhecer sons, como é o caso dos robôs futebolistas que apenas necessitam de "ouvir" o apito do árbitro. Este trabalho consiste no desenvolvimento de um sistema de reconhecimento de apito que possa ser usado por robôs futebolistas.

Um dos aspectos a ter em consideração quando se fala em sistemas de reconhecimento de fala ou sons é a imunidade ao ruído, porque o mundo é real e num mundo real não existem falas ou sons isolados, existe sempre ruído que pode ser muito ou pouco, logo um sistema de reconhecimento deve ter uma robustez ao ruído adequada ao ambiente onde esse sistema irá funcionar. Isto faz com que existam sistemas de reconhecimento com maior ou menor robustez ao ruído. Um sistema a funcionar num ambiente quase sem ruído, tem sempre mais hipóteses de sucesso no que diz respeito ao reconhecimento da própria fala ou som, pelo contrário, quanto mais ruidoso for o ambiente, menos hipóteses de sucesso tem o sistema de reconhecimento, sendo que para um sistema de reconhecimento de fala, a partir de um certo nível de intensidade de ruído, este deixa praticamente de ter sucesso.

1.1 Objectivos

O objectivo principal deste projecto é fazer um sistema electrónico que reconheça um apito com a máxima fiabilidade para o máximo de ruído ambiente possível, portanto trata-se de um sistema de reconhecimento de som, um sinal de um simples som não tem a complexidade de um sinal de fala, porque o som é praticamente constante enquanto existe, ao contrário de um sinal de fala. Então, se o sinal que se pretende reconhecer é mais simples do que um sinal de fala, o sistema para reconhecer esse som poderá também ser mais simples do que um sistema de reconhecimento de fala, visto que estes são muito complexos, quase sempre funcionam em computadores, o que os torna muito dispendiosos e volumosos. Posto isto, assume-se como objectivo deste projecto a simplicidade, de modo a fazer um sistema compacto e de baixo custo.

Este sistema tem também como objectivo conseguir distinguir 2 apitos diferentes, de modo a poder ser usado em robôs futebolistas quando ocorrem 2 jogos lado a lado, porque o apito do arbitro de um jogo, ouve-se no outro campo e o robô só pode interpretar sons do apito do arbitro do seu próprio jogo. A figura seguinte dá uma maior percepção deste problema.

fig 1 - 4 campos de robôs futebolistas próximos. (tirada de [1])

Como o sistema deve reconhecer, e só reconhecer determinado apito, tem que ser ajustável ao apito que pretende reconhecer. E como os jogos podem decorrer em diferentes ambientes, com diferentes ruídos predominantes, e com maior ou menor

intensidade de ruído, o sistema deve ser também ajustável ao ruído. Logo o sistema deve ser dotado de alguns parâmetros ajustáveis, tais que se consiga ajustar o sistema a determinado apito e ao ambiente.

Por último, o sistema deve ser genérico, ou seja, deve ser desenvolvido de modo a que possa ser utilizado por qualquer tipo de robô futebolista, e não ser desenvolvido apenas para um tipo de robô futebolista.

1.2 Método de resolução

Tendo em conta que os métodos de reconhecimento de fala não conseguem cumprir todos os objectivos do projecto, porque necessitam de um sistema com elevada capacidade computacional e não têm suficiente robustez ao ruído, estes são métodos descartados para a resolução deste problema. Então para que se possa fazer um projecto simples, compacto e de baixo custo, a solução é usar um microcontrolador, e criar software para esse dispositivo que realize o processamento de sinal adequado para fazer o reconhecimento de apito, esse software deve ser construído de modo a que existam parâmetros ajustáveis de acordo com os objectivos do projecto. Um microfone capta o sinal sonoro para a entrada do microcontrolador, este por sua vez deve devolver na saída o resultado, apito ou apenas ruído. A figura seguinte ilustra o método de resolução adoptado para a resolução deste problema.

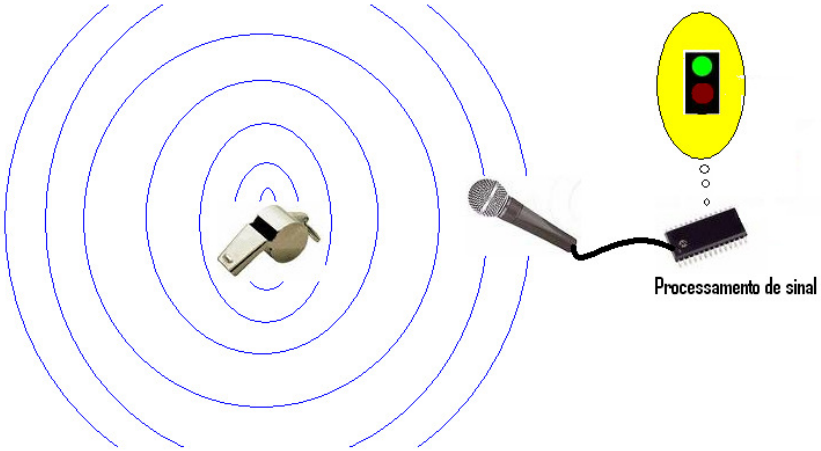


fig 2 – Funcionamento geral do sistema.

O sistema necessita também de um pequeno hardware de interface entre o microfone e o microcontrolador. O microfone utilizado deve ser escolhido pelas suas características técnicas, mas também pelo seu tamanho, este deve ser o menor possível, logo não pode ser igual ao representado na figura anterior, este foi desenhado para outros objectivos, e contém já pré-amplificador e filtros adequados ao objectivo que foi criado. Este sistema deve usar um microfone simples, ou seja, apenas o transdutor, e deve conter separadamente o pré-amplificador e filtros adequados a este sistema, estes elementos constituem o hardware de interface.

Capítulo 2 - Estado da Arte

Nesta fase foi feito um estudo a projectos semelhantes, em objectivos ou conteúdos, ao projecto em causa neste documento, foram vistos vários projectos, dos quais os 4 mais relevantes estão aqui descritos:

- **Reconhecimento de Apito para o RoboCup –** Projecto desenvolvido pela equipa de futebol robótico de Milão. Pode ser encontrado na referência [2].
- **Carro Robótico com Reconhecimento de Voz** – Projecto criado por Andre Harison e Chirag Shah. Pode ser encontrado na referência [3].
- **Implementação em Microcontrolador de um Sistema de Reconhecimento de Comandos de Voz para Interface Homem – Máquina em Sistemas Embebidos** – Projecto criado por Carlos Bernal Ruiz, Francisco E. Garcia Tapias, Bonifacio Martin del Brio e Antonio Bono-Nuez do Departamento de Engenharia Electrónica e Comunicações da Universidade de Saragoça, Espanha. Pode ser encontrado na referência [5].
- **Microsoft Speech Recognition Engine** – Software comercial da Microsoft para reconhecimento de fala.

2.1 Reconhecimento de Apito para o RoboCup

Este projecto foi o único encontrado que tem o mesmo objectivo principal do projecto descrito neste documento, ou seja, reconhecer um apito em ambientes ruidosos. Foi desenvolvido por uma equipa de futebol robótico de Milão, em linguagem de programação de alto nível (C++), para funcionar num computador com sistema operativo Linux.

Neste projecto, é captado o sinal de 64 amostras a uma frequência de amostragem de 8 kHz para o computador, através de um microfone interno, esse sinal é amplificado na placa de som do computador. De seguida é feita uma FFT – *Fast Fourier Transform*

para calcular o espectro do sinal, aplicando-se então um mecanismo que funciona como filtro, eliminando algumas amostras, em seguida o resultado é submetido a um programa de inteligência artificial que dirá se essas amostras são de sinal de apito. Existe ainda um programa de treino para a inteligência artificial. A figura seguinte ilustra o funcionamento do sistema.

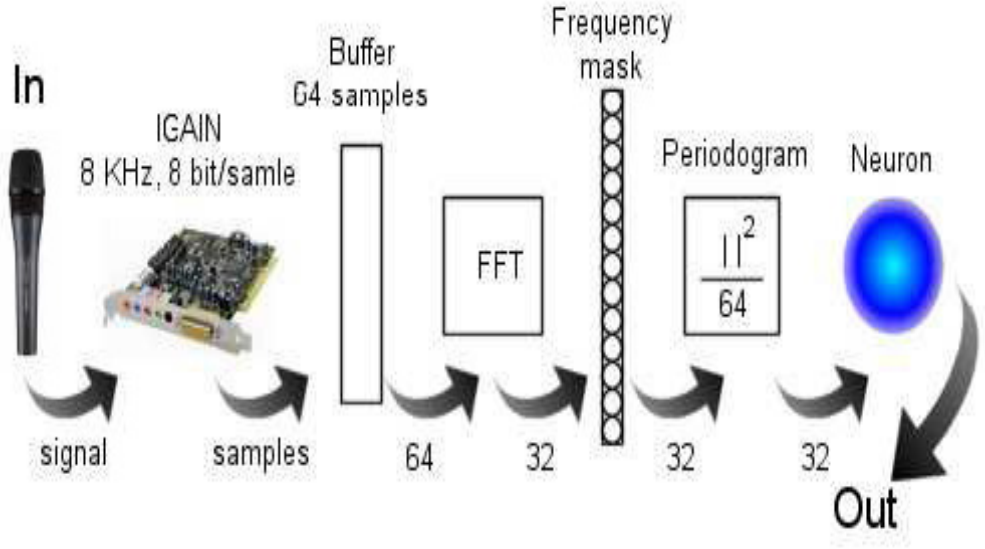


fig 3 – Funcionamento do sistema de reconhecimento de apito para o RoboCup. (tirada de [2])

Trata-se de um sistema construído para o mesmo objectivo principal do projecto descrito ao longo deste documento, contudo um sistema deste tipo não cumpre todos os objectivos referidos no capítulo I, no que diz respeito à generalidade e à complexidade os 2 sistemas diferem. Este sistema foi desenvolvido pela equipa de futebol robótico de Milão, para ser usado nos seus próprios robôs, ou seja, foi desenvolvido tendo por base o sistema já usado pelos robôs futebolistas da sua equipa. Podendo eventualmente ser adaptado com a devida autorização.

Outro aspecto que diferencia os 2 sistemas é a complexidade, este sistema foi desenvolvido para ser usado num computador, e tem obrigatoriamente que correr num computador, dada a sua complexidade computacional, principalmente com o uso de inteligência artificial. Não pode ser aplicado um sistema destes num microcontrolador. Deste modo o sistema é muito volumoso e caro, o que na realidade não se faz sentir porque os robôs onde vai ser usado este sistema já tem o computador, apenas é acrescentado software. Este projecto não tem desenvolvimento de hardware, todo o hardware de interface necessário está na placa de som.

Em resumo, este projecto trata-se de um software desenvolvido para um tipo de robôs, enquanto que o projecto em causa neste documento trata-se de um pequeno dispositivo para ser adicionado a qualquer tipo de robô futebolista que pretenda fazer o reconhecimento de apito.

2.2 Carro Robótico com Reconhecimento de Voz

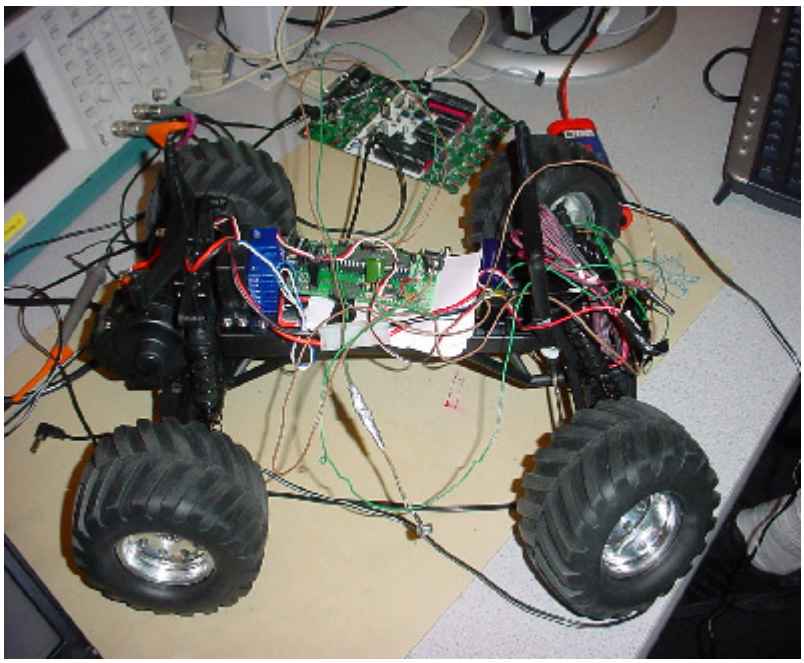


fig 4 - Imagem do carro robótico. (tirada de [3])

Este projecto trata-se de um carro robótico que interpreta comandos de voz para ser controlado.

O sistema utilizado para fazer o reconhecimento dos comandos de voz faz a aquisição do sinal para um microcontrolador através de um microfone seguido de um pré-amplificador e um filtro passa-banda ligados à entrada de um ADC – *Analogic to Digital Converter* interno ao microcontrolador. É feita uma aquisição de 256 amostras a uma frequência de amostragem de 4 kHz. O funcionamento geral do sistema de reconhecimento de voz do carro está representado na figura seguinte.

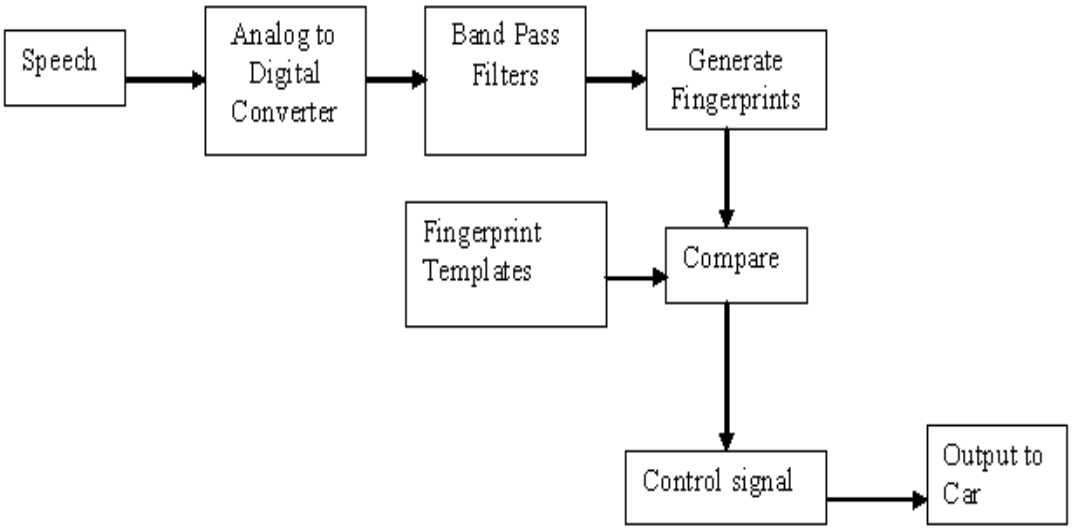


fig 5 – Diagrama de blocos do carro robótico. (tirada de [3])

O algoritmo deste projecto, funcionam com base numa na comparação entre imagens da palavra a reconhecer, a que chamam de *fingerprints.* Existe um conhecimento prévio dos *fingerprints*, ou seja, é necessário criar previamente os *fingerprints* de exemplo da palavra que se pretende reconhecer para durante o reconhecimento serem comparados com os *fingerprints* do sinal captado.

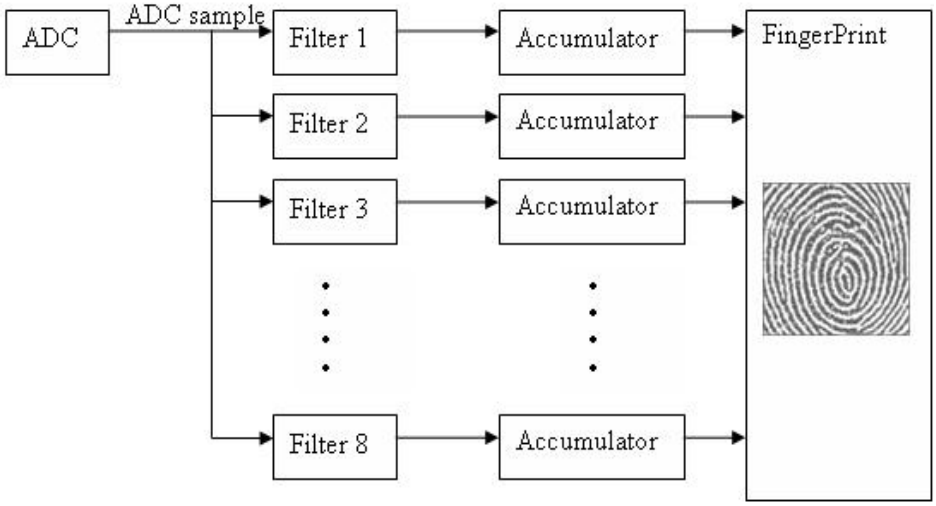


fig 6 – Sistema de geração dos *fingerprints.* (tirada de [3])

Neste projecto usa-se o microcontrolador ATmega32, tem apenas 8 bits e funciona com velocidades de relógio até 16 MHz. O sistema requer apenas um pequeno hardware de interface entre o microfone e o ADC do microcontrolador. Desta forma consegue-se fazer um sistema de reconhecimento de voz compacto e de baixo custo, como o pretendido no projecto de reconhecimento de apito descrito neste documento,

contudo o algoritmo é demasiado sensível a ruídos, o que faz com que só se consiga controlar o carro em ambientes muito limpos de ruído, descartando-se o uso deste algoritmo em sistemas para robôs futebolistas.

2.3 Implementação em Microcontrolador de um Sistema de Reconhecimento de Comandos de Voz para Interface Homem – Máquina em Sistemas Embebidos

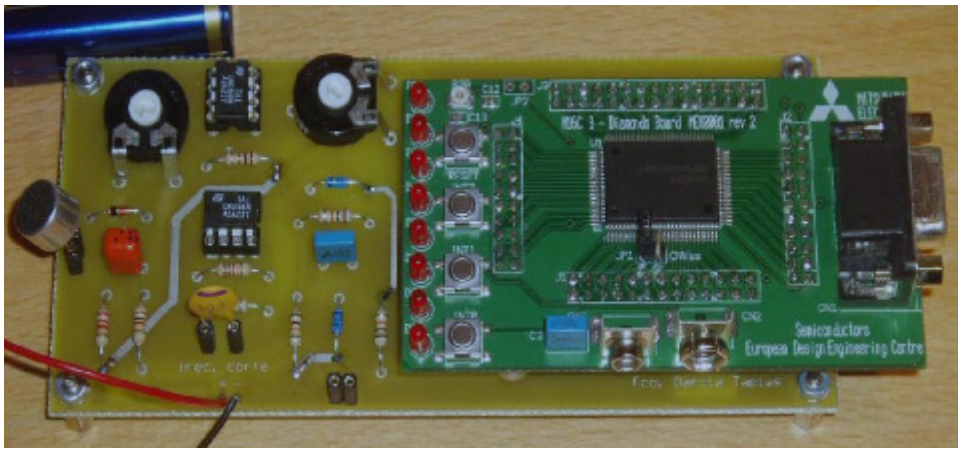


fig 7 – Placa resultante do projecto de reconhecimento de voz em microcontrolador. (tirada de [5])

Apesar de os algoritmos de reconhecimento de fala serem hipótese descartada para este projecto, dada a sua exigência computacional e a sua reduzida robustez ao ruído, faz sentido descrever alguns desses sistemas comparando-os com este sistema, já que, como neste projecto, o objectivo é colocar a máquina a interpretar sinais sonoros, no caso dos sistemas de reconhecimento de fala, esses sinais correspondem a palavras.

O projecto de reconhecimento de voz em microcontrolador desafia a capacidade dos microcontroladores na implementação de sistemas de reconhecimento de fala, como já foi referido, dada a complexidade computacional dos sistemas de reconhecimento de fala, estes quase sempre são implementados em computadores. Este sistema não obedece a essa regra e usa um algoritmo de reconhecimento de fala relativamente robusto implementado num microcontrolador, contudo o sistema é bastante limitado.

Este projecto usa um sistema de reconhecimento de fala baseado em redes neuronais, um algoritmo que exige alguma capacidade computacional enquanto corre, mas essa capacidade computacional pode ser dada por um microcontrolador, contudo o que torna as redes neuronais em algoritmos exigentes a nível computacional é o seu treino, este precisa de um sistema com capacidade computacional bastante elevada.

Neste projecto é separada a parte de treino da restante das redes neuronais, fazendo-se o treino num computador, então envia-se para o microcontrolador os parâmetros treinados, e o sistema de redes neuronais corre no microcontrolador. Nas figuras seguintes encontram-se respectivamente o funcionamento de um sistema de fala baseado em redes neuronais e o diagrama de blocos deste sistema.

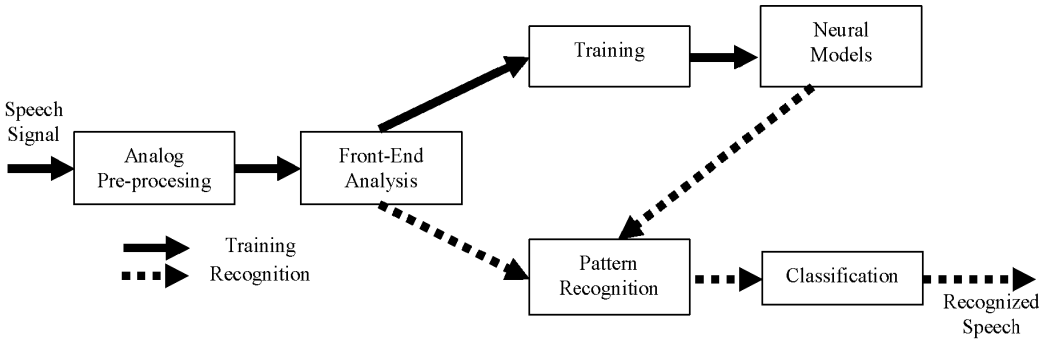


fig 8 - Funcionamento de um sistema de fala baseado em redes neuronais. (tirada de [5])

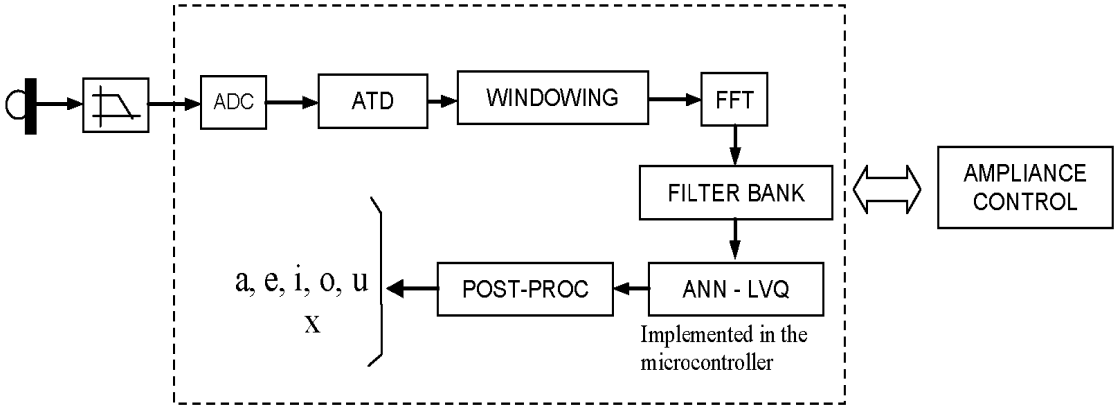


fig 9 – Digrama de blocos do sistema de reconhecimento de voz em implementado em microcontrolador. (tirada de [5])

Neste projecto é captado um sinal sonoro para um ADC de um microcontrolador, o ADC tem 10 bits de resolução, é feita a aquisição de 512 amostras a uma frequência de amostragem de 2 kHz. De seguida o bloco ATD - *Automatic Threshold Detector* decide quando se faz a próxima aquisição de sinal. O bloco WINDOWING determina o tempo máximo que a voz pode durar. De seguida é feita uma FFT para obter o espectro do sinal em função do tempo, e então aplica-se um mecanismo que funciona como filtro. Os 2 blocos seguintes dizem respeito às redes neuronais, onde são criados os LVQ's – *Learning Vector Quantization* e comparados com os LVQ's obtidos no treino, resultando no reconhecimento de uma letra neste caso.

Este projecto de reconhecimento de voz usa o microcontrolador de 16 bits Mitsubishi M16C, este dispositivo possui 20 kB de memória RAM, e funciona com velocidades de relógio até 24 MHz.

O projecto de reconhecimento de fala implementado em microcontroladores, tem a grande vantagem de ser feito em sistemas embebidos, ao contrário dos restantes sistemas de reconhecimento de fala, contudo não possui a capacidade de reconhecimento da maioria dos sistemas, este apenas consegue reconhecer letras.

Em comparação com o sistema de reconhecimento de apito em ambientes ruidosos, um sistema deste tipo consegue cumprir parte dos objectivos, é simples, pequeno e de baixo custo. Tendo método de treino, consegue ser ajustável, contudo não possui robustez ao ruído e fiabilidade suficientes conforme os objectivos do projecto de reconhecimento de apito.

2.4 Microsoft Speech Recognition Engine

O Microsoft Speech Recognition Engine é um software da Microsoft que permite ao utilizador criar um documento falando para o computador, este software escreve as palavras ditas pelo utilizador, e também interpreta comandos de voz (ex: "*delete*"). Pode escrever não só a palavra, como também letra a letra. O método de reconhecimento de fala usado é o HMM – *Hidden Markov Models*.

Este software corre sobre o Microsoft Word, ou seja, funciona como uma aplicação do próprio Word, e pode funcionar em Windows XP e Windows Vista.

Inicialmente apenas foi criado para interpretar palavras, letras ou comandos em inglês, contudo a Microsoft Portugal tem vindo a desenvolver uma versão deste software para fazer o mesmo reconhecimento em português.

O Microsoft Speech Recognition Engine apresenta resultados relativamente razoáveis no que diz respeito à interpretação das palavras, apresentando uma boa percentagem de reconhecimentos correctos, contudo estes resultados ainda não são suficientemente bons para que se possa criar um documento com a comodidade necessária usando só este software. Existem muitos conflitos no que diz respeito à distinção, por parte do software, entre comandos e palavras, e para as palavras de

pronúncia mais complexa, a percentagem de sucesso é muito reduzida. Na referência [4] encontra-se um vídeo com uma demonstração deste software.

O sistema de reconhecimento de fala aqui apresentado, embora tenha objectivos algo semelhantes com o sistema de reconhecimento de apito, é um sistema diferente em quase todos os aspectos, trata-se de um software bastante complexo que corre num computador, ao contrário do sistema de reconhecimento de apito que tem que ser simples, e também deve ter mais fiabilidade e robustez ao ruído. Outro aspecto de diferenciação é a distância entre o orador, ou arbitro no caso do apito, e o microfone, no Microsoft Speech Recognition Engine o orador deve estar próximo do microfone, enquanto que no reconhecimento de apito, o arbitro pode estar a uma distancia maior do microfone.

2.5 Conclusões do Estado da Arte

Na pesquisa do estado da arte para este projecto foram encontrados vários projectos que reconhecem som. Seja esse som, palavras, letras ou um simples som como o apito. Quanto à robustez ao ruído e fiabilidade, esses projectos variam muito, sendo que, por norma os mais robustos são também os mais complexos, que requerem sistemas de elevada capacidade computacional. Não existindo nenhum projecto que cumpra integralmente os objectivos do projecto de reconhecimento de apito em ambientes ruidosos. Para se cumprirem os objectivos tem que se utilizar um método mais simples que o dos projectos implementados em computadores, a complexidade dos projectos do carro robótico e do sistema de reconhecimento de voz implementado em microcontrolador é aceitável, contudo no que diz respeito à robustez ao ruído e fiabilidade o projecto de reconhecimento de apito tem a necessidade de ter melhores resultados. Isto cria um grande desafio para o desenvolvimento do projecto, e também torna este projecto numa novidade, já que ainda não foi encontrado nenhum sistema de reconhecimento de apito implementado num sistema compacto.

Capítulo 3 – Teoria do Som

3.1 Definição de Som

Para um projecto onde o objectivo é fazer o reconhecimento de determinado som, é importante saber-se o que o som é na realidade, ou seja, descrever o som do ponto de vista da física.

"O som é a propagação de uma frente de compressão mecânica ou onda longitudinal." Referência [6] – Wikipedia.

"O som propaga-se no ar através de um movimento ordenado das partículas que o constituem." Referência [7] - Ministério da Ciência e da Tecnologia

"A definição da acústica aponta para a variação rápida da onda de pressão num meio." Referência [8]

"As ondas sonoras são vibrações mecânicas com uma frequência variável." Referência [9]

Estas são algumas das definições de som encontradas nas respectivas referências.

As cordas vocais, um instrumento musical, um apito, ou qualquer fonte sonora provocam vibrações ou oscilações em partículas, e a essas oscilações chama-se de som. As partículas são as responsáveis pela propagação do som, ou seja, quando entram em contacto entre si, a oscilação vai ser transmitida entre partículas. A referência [7] tem uma animação onde se pode ter melhor percepção da vibração das partículas. O som chega ao ouvido através das partículas de ar que o rodeiam, que ao vibrarem fazem também vibrar o tímpano. Contudo não são somente as partículas de ar que propagam o som, qualquer tipo de material sólido, líquido ou gasoso pode propagar o som, este apenas não se propaga no vácuo.

Para que se possa entender melhor o som, e até fazer processamento de som, não basta saber que o som é uma oscilação de partículas, é necessário trabalhar com

grandezas, então, esta vibração de partículas é representada por um gráfico onde o eixo das abcissas representa o tempo em segundos e o eixo das ordenadas representa a variação da pressão acústica em N/m^2 (ou Pascal) num determinado ponto da trajectória das partículas. A figura seguinte representa um som puro, ou seja, esse gráfico é uma sinusóide.

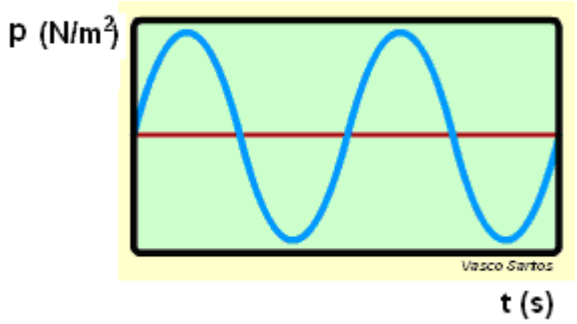


fig 10 – Gráfico representativo de um som puro. (adaptada de [7])

3.2 Amplitude, Frequência e Fase

A amplitude de um sinal sonoro, é o maior desvio de pressão acústica em relação ao valor de pressão do repouso, no caso de um som puro, a amplitude do sinal acústico corresponde à própria amplitude da sinusóide, como se encontra ilustrado na figura seguinte, onde está representada por A.

Como se trata de um valor de pressão acústica, a amplitude é expressa em N/m^2.

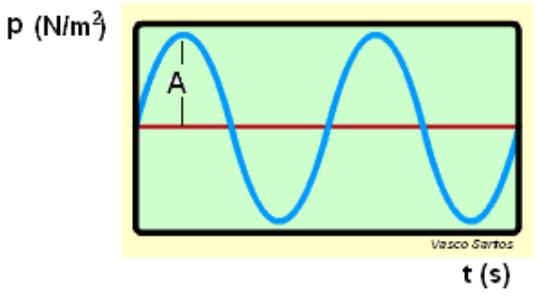


fig 11 – Representação da amplitude de um som puro. (adaptada de [7])

A amplitude de um som pode ser entendida como o volume do som, ou seja, quando se aumenta o volume de determinada fonte sonora, está-se a aumentar a amplitude da pressão acústica desse sinal, isto é, aumenta-se a oscilação das partículas

que compõem o som. Este aumento de amplitude faz também com que o sinal sonoro atinja uma distância maior.

A frequência de um sinal sonoro corresponde à frequência da sinusóide que representa a variação de pressão acústica para um som puro. Define-se como sendo o número de vezes que um ciclo da sinusóide se repete durante um segundo. Pode ser calculada como o inverso do período dessa sinusóide, onde o período é o tempo em segundos de um ciclo. A unidade respectiva à frequência é o Hz.

$$f = \frac{1}{T} \qquad \textbf{eq (1)}$$

Onde:

f - Frequência do sinal.

T - Período do sinal.

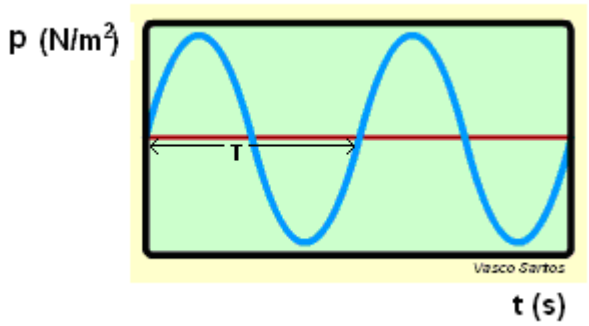


fig 12 – Representação do período de um som puro. (adaptada de [7])

A frequência é a grandeza fundamental de um som, conseguem-se distinguir sons entre si porque têm frequências diferentes, caso contrário, se dependessem apenas da amplitude, seria sempre o mesmo som, mais alto ou mais baixo.

Com o aumento da frequência os sons vão ficando mais agudos, já para frequências mais baixas, os sons são mais graves. O ouvido humano consegue captar sons numa escala máxima de 16 Hz a 20 kHz, esta escala apenas pode ser atingida por uma criança, com o passar da idade a frequência superior deixa de ser 20 kHz e vai diminuindo, dependendo de pessoa para pessoa, um valor normal dessa frequência para um adulto é por exemplo 15 kHz. É à gama de frequências entre16 Hz e 20 kHz que se chama simplesmente de sons, abaixo de 16 Hz chama-se infra-som e acima de 20 kHz chama-se ultra-som.

A fala humana tem normalmente frequências na gama entre 20 Hz a 2 kHz, contudo a voz humana consegue atingir até 12 kHz, este valor pode ser atingido por exemplo por uma cantora de ópera.

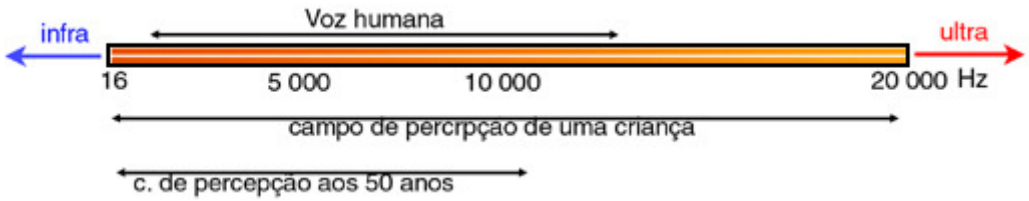


fig 13 – Descrição de sons com a variação da frequência. (tirada de [9])

Num determinado ambiente nunca existe apenas um som puro, existem vários sons com frequência e amplitude diferentes, na realidade quando vários sons existem ao mesmo tempo estes são somados, e a representação gráfica do som resultante obtém-se somando os gráficos dos sons puros que o compõem.

Para além de os sons terem frequências e amplitudes diferentes, estes podem não começar todos ao mesmo tempo, ou em sincronia, então surge outra grandeza denominada de fase. Esta grandeza diz respeito ao tempo, que determinado som puro está atrasado ao adiantado em relação a uma referência escolhida. A fase pode ser expressa em segundos, ou fazendo uma análise trigonométrica à sinusóide que representa o som, pode-se expressar em graus (º). Na figura seguinte pode notar-se a importância da fase entre 2 sinais sonoros.

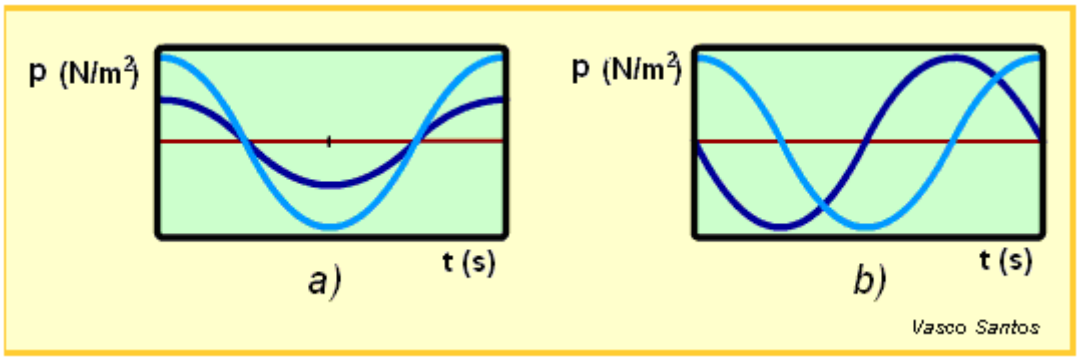


fig 14 – a) 2 sinais com a mesma frequência e fase; b) 2 sinais com a mesma frequência mas com fases diferentes. (adaptada de [7])

Estas 3 grandezas (amplitude, frequência e fase) servem de base na representação de determinado sinal sonoro, ou seja, qualquer sinal sonoro é representado pela soma de sinais sonoros puros, cada um com a respectiva amplitude,

frequência e fase. Contudo, dependendo do sinal sonoro, a sua representação pode ser pouco perceptível, ou seja, a soma dos sons puros que o compõe resulta em algo que não se assemelha a uma sinusóide, isto pode verificar-se na figura seguinte, onde se somam até 3 sinusóides puras. Nota-se que essa soma resulta num sinal bastante complexo. Normalmente o número de sinusóides que compõe um sinal sonoro é muito maior que 3, o que o torna num sinal bastante mais complexo que o sinal final representado na figura seguinte.

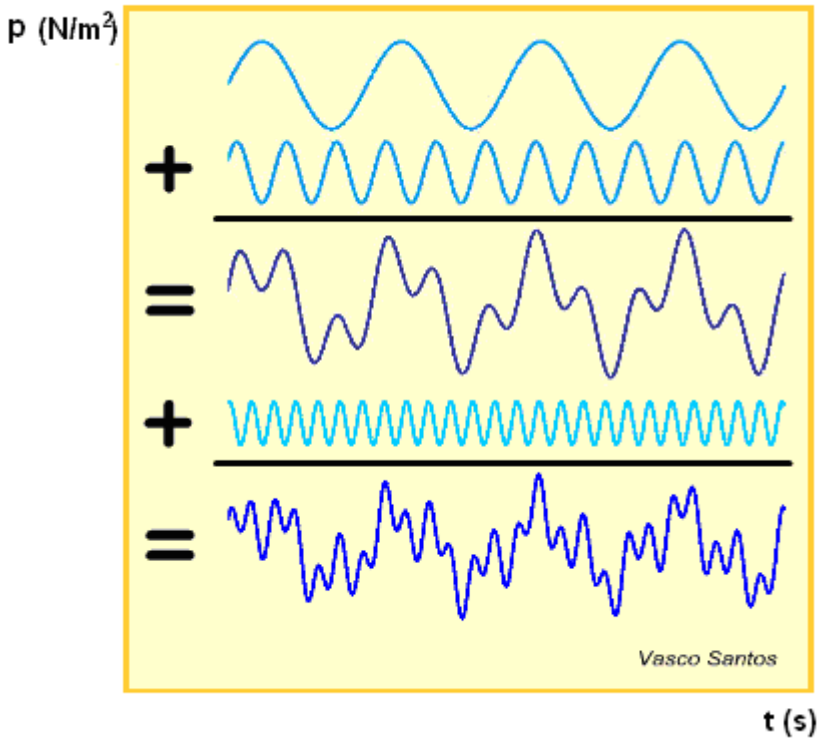


fig 15 – Soma de 3 sinais sonoros puros. (adaptada de [7])

3.3 Intensidade Sonora

A intensidade sonora é uma grandeza definida como potência por unidade de área (W/m^2), por sua vez a potência define-se como a energia emitida por determinada fonte sonora por unidade de tempo.

Por conveniência, é normal usar-se o nível de intensidade sonora, onde a unidade de medida é o decibel (dB), é uma unidade de medida adimensional, representa a fracção da variação da mesma intensidade sonora expressa em W/m^2, em relação ao

estado de repouso em escala logarítmica. A intensidade sonora em dB pode ser calculada pela seguinte formula:

$$I(dB) = 10\log(I / I_0) \qquad \textbf{eq (2)}$$

$I(dB)$ – Nível de intensidade sonora em dB

I – Intensidade sonora em W / m^2

I_0 - Intensidade sonora em repouso, $I_0 = 10^{-12} W / m^2$

Na tabela seguinte encontra-se o nível de intensidade sonora médio para vários exemplos de sons. Os níveis de intensidade sonora a partir de 90 dB são níveis perigosos para o ouvido humano quando está exposto constantemente a esses sons.

dB	**EXEMPLOS**
30	Biblioteca silenciosa, sussurro leve
40	Sala de estar, geladeira, quarto longe do trânsito
50	Trânsito leve, conversação normal, escritório silencioso
60	Ar condicionado com 6 m de distância, máquina de costura
70	Aspirador de pó, secador de cabelo, restaurante barulhento
80	Tráfego médio de cidade, colector de lixo, despertador com 60 cm de distância
90	Metro, motocicleta, tráfego de caminhão, máquina de cortar grama
100	Camião de lixo, serra eléctrica, furadeira pneumática
120	Concerto de Rock em frente as caixas de som, trovão
140	Espingarda de caça, avião a jacto
180	Lançamento de foguetão

Tabela 1 – Níveis de intensidade sonora. (tirada de [10])

Capítulo 4 – Análise de sinais sonoros

Na análise aos objectivos do projecto, estabeleceu-se um método de resolução que passa por fazer processamento do som num microcontrolador, como está representado na figura 2, mas antes de entrar em mais detalhe no projecto do sistema, é necessário estudar os sinais em causa, para que se possa tomar decisões ao pormenor acerca do sistema de reconhecimento de apito.

Como foi visto anteriormente, um sinal sonoro normalmente é composto por vários sinais sonoros puros somados, e representando directamente essa soma, o sinal pode ser bastante complexo sendo muito difícil ter uma percepção desse sinal. Então nesta fase do trabalho, o objectivo é fazer uma análise aos sons puros que compõe determinado sinal. Esta análise é chamada de análise espectral, trata-se de uma análise ao espectro do sinal, espectro esse que consiste num gráfico da amplitude em função da frequência, que equivale a ter uma sinusóide de uma certa frequência com a respectiva amplitude. Existe também o espectro da fase, mas este não é muito relevante em sinais sonoros.

4.1 Software Utilizado

Para fazer a análise espectral, utilizou-se um software adequado, o **FFT MusEV**. Este software foi desenvolvido no Departamento de Química da Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da Universidade de S. Paulo, Brasil. É um software gratuito e de uso exclusivamente didáctico, encontra-se disponível na referência [11].

O software **FFT MusEV** recebe ficheiros .wav, sendo necessário outro software para gravar esses mesmos ficheiros, o software utilizado para essa tarefa foi o gravador de áudio do Windows. Após receber o ficheiro, o **FFT MusEV** faz uma FFT – Fast Fourier Transform de $2^n (n = 1,2,3...)$ pontos escolhidos pelo utilizador e mostra o gráfico correspondente onde no eixo das abcissas tem-se a frequência em Hz, e no eixo das ordenadas tem-se a amplitude, o programa não especifica a escala da amplitude, contudo como está a lidar com sinais sonoros, essa amplitude trate-se de um valor de pressão, expressa em N/m^2. No gráfico também não é mostrado o valor dessa

amplitude, contudo permite comparar as amplitudes das diferentes sinusóides com frequência diferentes, sendo esta comparação suficiente para a análise pretendida.

4.2 Sinais Analisados

Quanto aos sinais a analisar, o mais relevante é o próprio sinal de apito, de modo a verificar quais as características que permitem a sua identificação. Feitas algumas análises a sons produzidos com um apito, verifica-se que o espectro do apito tem a forma representada na figura seguinte.

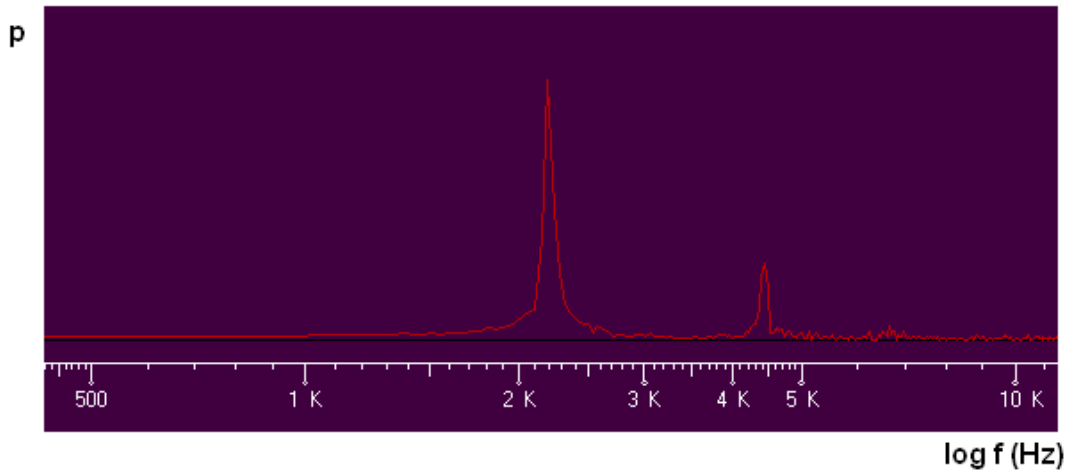


fig 16 – Espectro de um som de apito.

Na imagem anterior verifica-se que o som de um apito possui a sinusóide de maior amplitude à frequência de 2.2 kHz, a esta frequência chama-se de frequência de pico, verifica-se também que existe outro pico em 4.4 kHz, contudo de menor amplitude. Com esta imagem é possível caracterizar o som de um apito, contudo não se pode verificar que a forma seja sempre esta e os valores de pico sejam sempre os valores aqui referidos. Portanto é exigida uma análise à variação do apito com a variação de alguns factores, que à partida possam ter influência no sinal sonoro produzido pelo apito. Através do estudo teórico e de alguma experiência com o software FFT MusEV, os factores de possível influência no sinal escolhidos para análise são os seguintes:

- Pessoa que sopra
- Intensidade de sopro
- Distância ao microfone
- Apito (instrumento)
- Instante de tempo em que é captado o sinal

4.2.1 Resultado da Análise dos Vários Factores

Pessoa que sopra

Para analisar a influência, da pessoa que sopra no apito, no espectro do som resultante, fez-se análises a sons de apito soprado por 4 pessoas diferentes, onde se pode verificar que a pessoa que sopra tem influência significativa no espectro do sinal, para a mesma pessoa a frequência de pico do espectro, mantém-se dentro de uma pequena gama (caso não altere significativamente a intensidade de sopro), para outra pessoa a frequência de pico pode alterar-se significativamente. Na figura seguinte mostra-se 2 espectros do sopro de 2 pessoas diferentes.

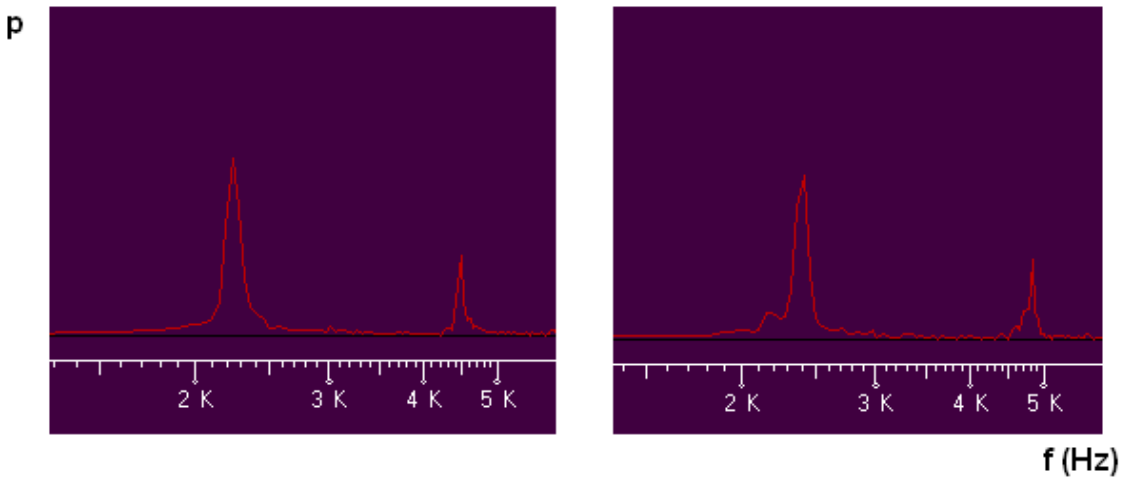


fig 17 – Espectros de sinal de apito para 2 pessoas diferentes.

Intensidade de sopro

Para avaliar os efeitos da intensidade de sopro no espectro do som de apito, fez-se apenas testes com uma pessoa a soprar com diferentes intensidades, onde se verificou

que com o aumento da intensidade de sopro, aumenta não só a amplitude do espectro, como também aumenta a frequência de pico do mesmo. De seguida representa-se o resultado deste teste para uma intensidade de sopro baixa na figura a) e o resultado para uma intensidade de sopro elevada na figura b).

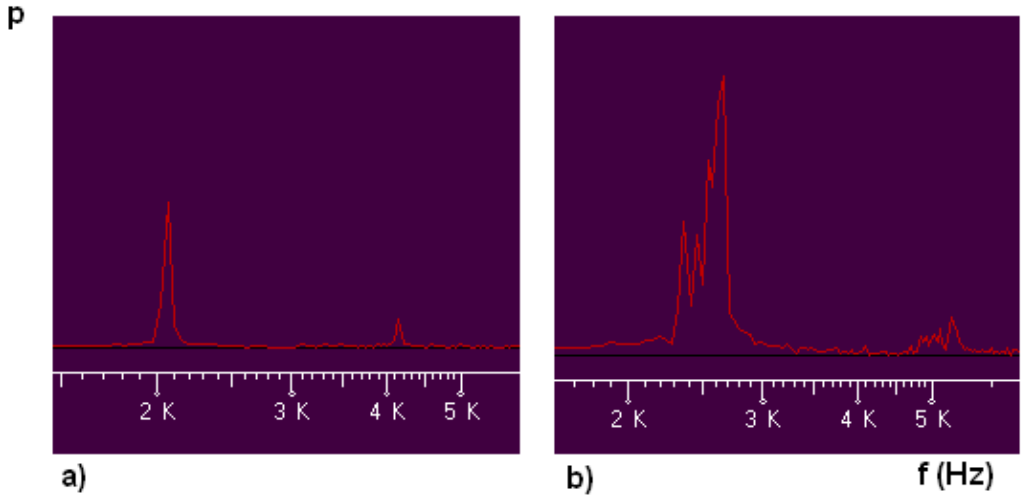


fig 18 – a) Espectro do sinal de apito com intensidade de sopro baixa; b) Espectro de sinal de apito com intensidade de sopro elevada.

Distância ao microfone

Para analisar os efeitos da distância entre o microfone e o apito no espectro do som captado pelo microfone, fez-se testes com uma pessoa a soprar perto do microfone e longe do microfone (cerca de 10 metros). Nestes testes verificou-se que a distância ao microfone é um factor de forte influência no espectro do som captado, quando se aumenta a distância, a amplitude do espectro diminui, e para a distância de 10 metros o 2º pico que existia um pouco antes dos 5 kHz deixa praticamente de existir. Como se pode verificar na figura seguinte.

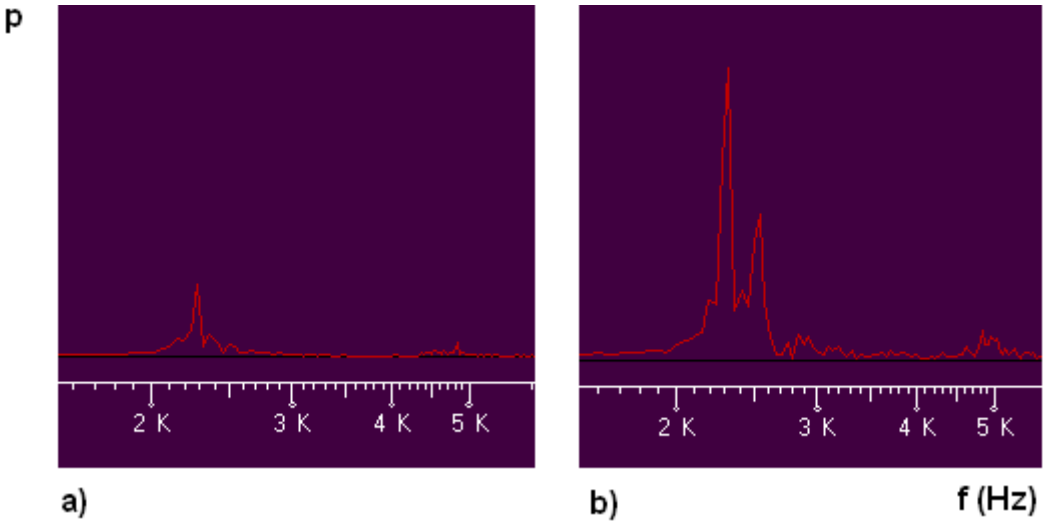


fig 19 – a) Espectro do sinal do som de apito com o apito afastado do microfone; b) Espectro do sinal do som de apito com o apito próximo do microfone.

Apito (instrumento)

A influência do apito no espectro resultante é um factor do qual depende um dos objectivos deste projecto, distinguir 2 apitos diferentes. Para se cumprir este objectivo é necessário que para apitos diferentes o sinal sonoro produzido e consequentemente o espectro desse sinal sejam diferentes. Então fez-se o teste com a mesma pessoa a soprar em 2 apitos diferentes, verificando-se que de facto os sinais produzidos pelos apitos diferem na frequência de pico, contudo essa diferença não é tão significativa como a diferença da variação de outros factores (ex.: intensidade de sopro). Este facto dificulta a tarefa de distinguir 2 apitos diferentes. A figura seguinte mostra o resultado do teste para 2 apitos diferentes.

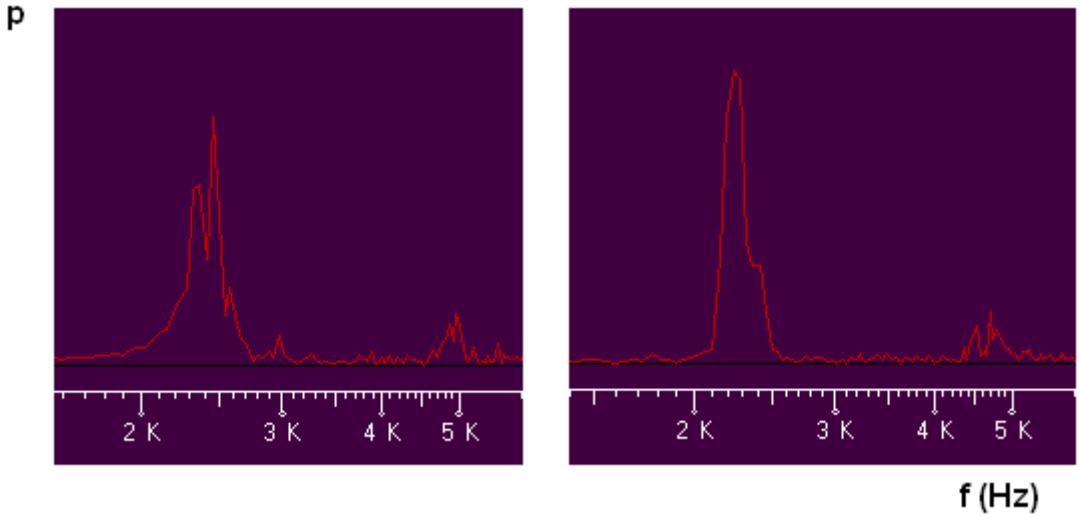


fig 20 - Espectros de sinal de apito para 2 apitos diferentes.

Instante de tempo em que é captado o sinal

Quando alguém sopra num apito não consegue produzir um som exactamente igual desde quando começa a apitar até terminar, então faz todo o sentido analisar várias vezes o espectro do mesmo som de apito para vários instantes diferentes. Nesta análise verificou-se que de facto o instante em que é captado o sinal tem influência no espectro resultante, quer na amplitude, quer na frequência de pico e na forma do espectro. Na figura seguinte mostra-se o resultado deste teste para duas situações.

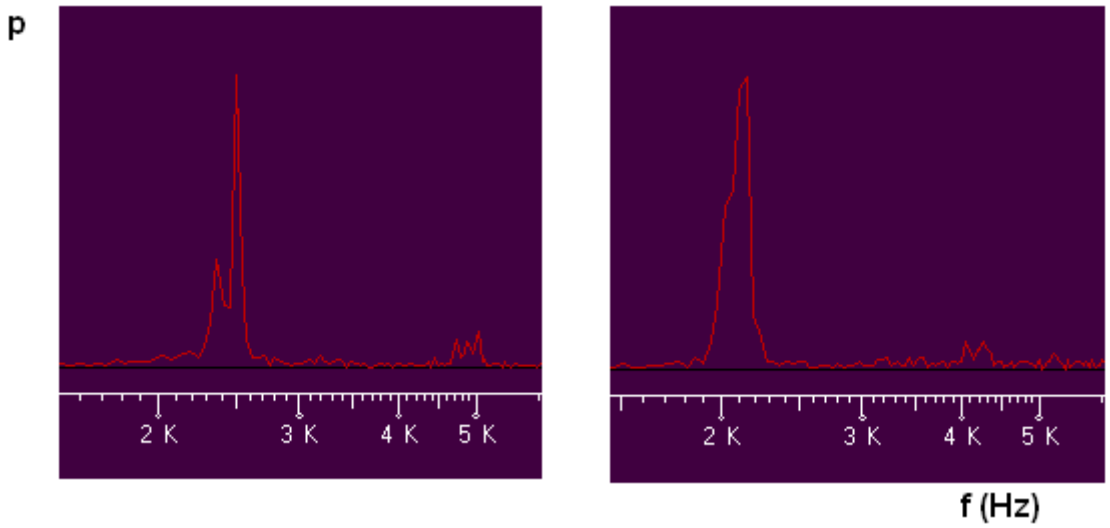


fig 21 – Espectros de sinal de apito do mesmo som captado em instantes diferentes.

4.2.2 Análise de Ruídos

Um dos maiores objectivos do sistema é a imunidade ao ruído, o sistema deve reconhecer um apito e ignorar todos os sons que não forem produzidos pelo apito. Então torna-se importante conhecer o espectro de uma larga gama de ruídos, para se poder criar um sistema que garanta rejeição a esses ruídos. Contudo, como existe uma infinidade de tipo de ruídos é impossível fazer uma análise a todos, então optou-se por fazer análise a ruídos que aparentemente têm som parecido com o apito e outros tipos de ruídos que provocam um som normalmente forte para a percepção humana. Os ruídos que foram analisados são os seguintes:

- Assobio
- Assobio de dedos
- Palmas
- Aspirador

4.2.3 Resultado da Análise de Ruídos

Assobio

O assobio aqui analisado é um assobio normal, feito por uma pessoa utilizando apenas os lábios para tal.

Como resultado desta análise verifica-se que o som do assobio forma quase uma sinusóide pura com 1.2 kHz de frequência, ou seja, o seu espectro é praticamente nulo excepto em 1.2 kHz. O espectro obtido neste teste encontra-se na figura seguinte.

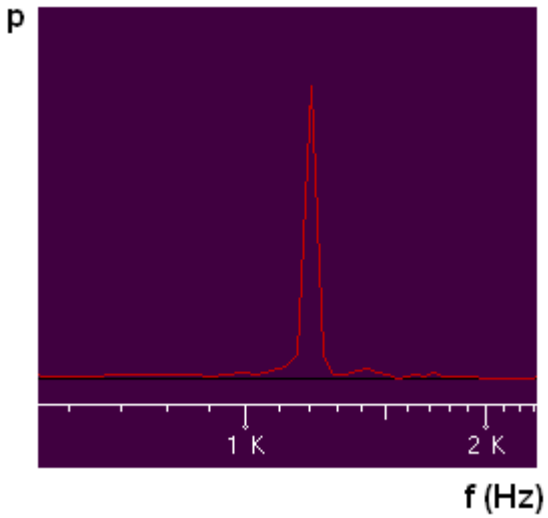


fig 22 – Espectro do som do assobio.

Assobio de dedos

O assobio de dedos aqui analisado é feito por uma pessoa colocando os dedos na boca e assobiando, este tipo de assobio é muito usado, por exemplo, por adeptos de futebol para demonstrar o seu descontentamento, na maioria das vezes com o arbitro, dado que é um assobio muito forte.

Na análise espectral deste som verificou-se que um assobio de dedos é muito irregular, ou seja, em instantes de tempo diferentes pode variar muito a sua frequência de pico, esta pode ter valores inferiores a 1.5 kHz como também pode chegar a 2.5 kHz. Quanto à sua forma, é muito semelhante à forma do apito. Estas características tornam o assobio de dedos num ruído difícil de distinguir do apito, dado que em certos instantes ele tem um espectro semelhante, contudo é menos constante do início ao fim do som,

comparando com o apito. De seguida apresenta-se 3 espectros correspondentes a 3 momentos diferentes do mesmo assobio de dedos.

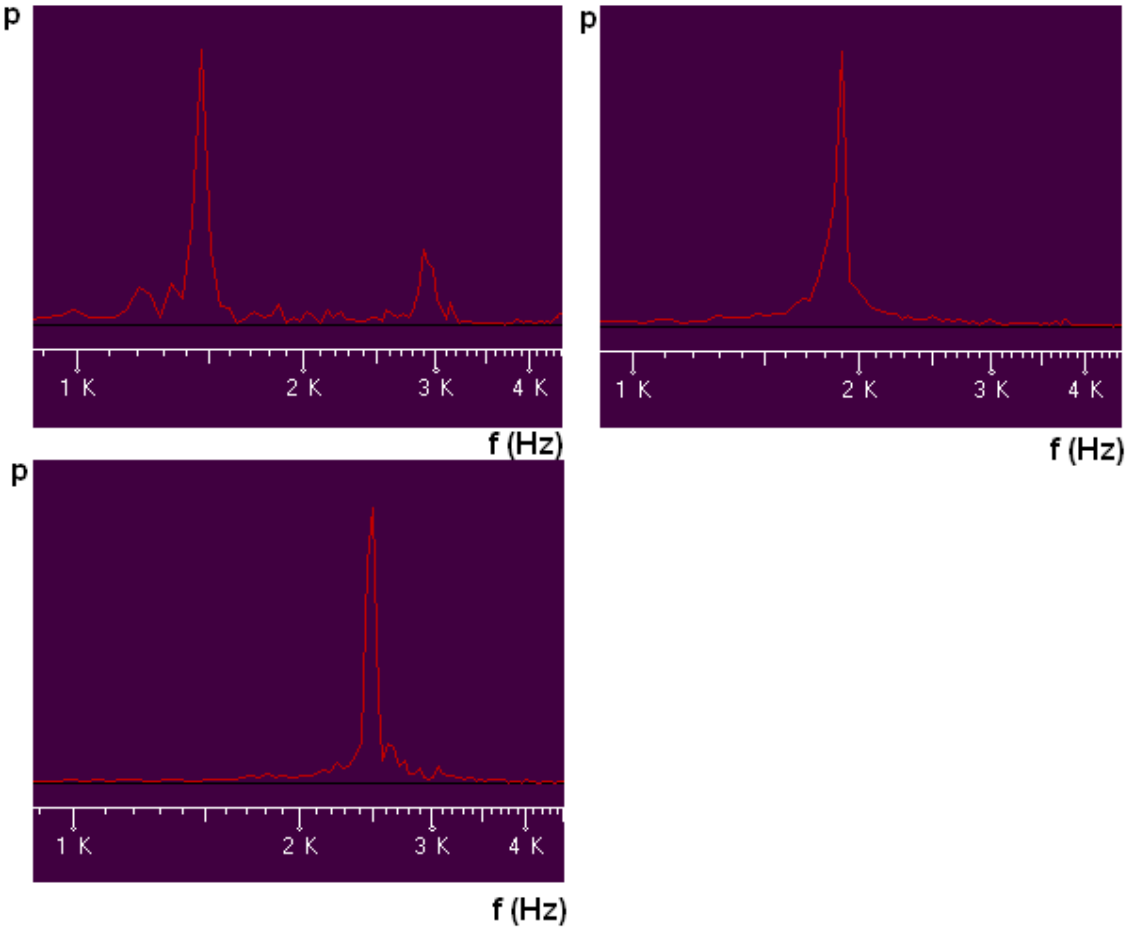


fig 23 – Espectros de assobio de dedos.

Palmas

O som das palmas embora não seja um som com parecenças com o som de apito, é um som muito forte e na qual os robôs futebolistas estão muito sujeitos, daí a conveniente análise do seu sinal sonoro. Neste teste o som das palmas foi feito por 8 pessoas numa sala fechada.

A característica mais relevante do som das palmas é a sua instabilidade, dado que este muda a cada instante, o seu espectro não tem uma forma que se possa considerar fixa. Podem existir frequências de amplitude elevada na gama do espectro do apito, o que leva a que este ruído possa influenciar no reconhecimento de apito.

Na figura seguinte apresenta-se 3 momentos do som de palmas.

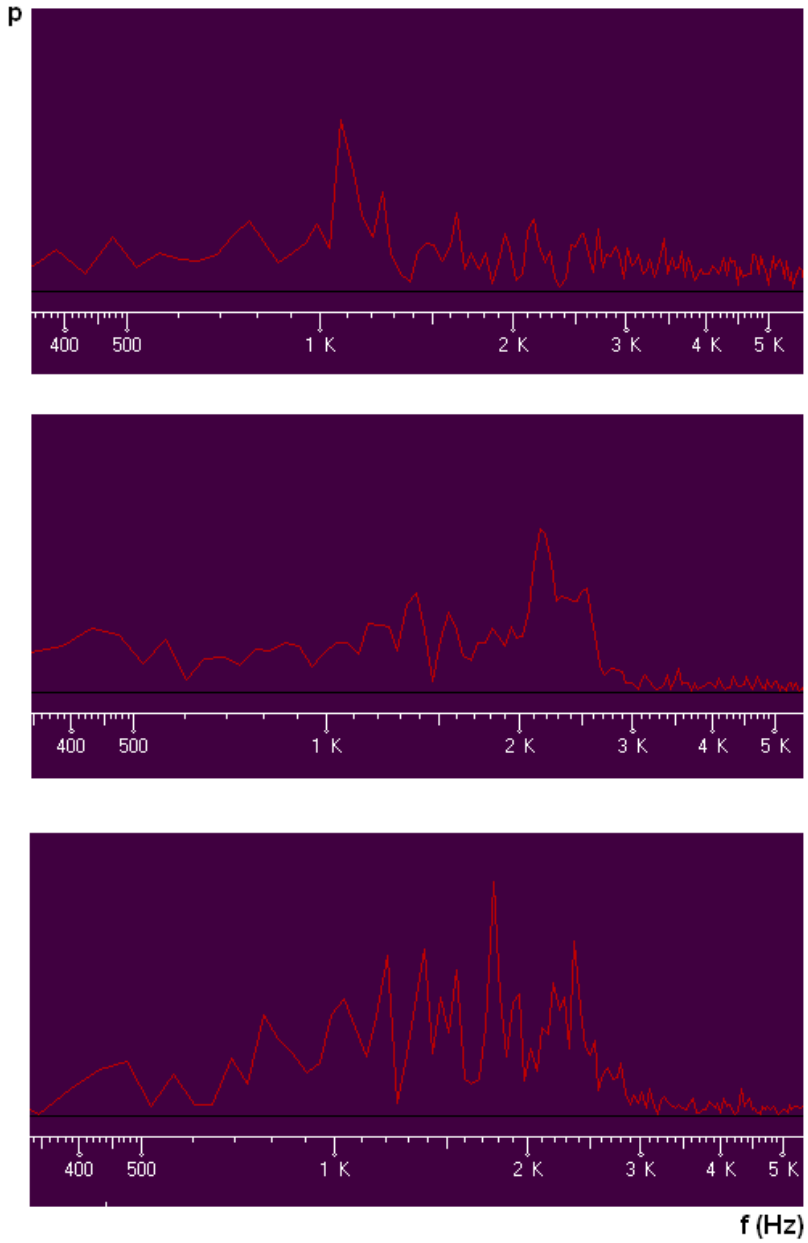


fig 24 – Espectros de sons de palmas.

Aspirador

Tal como o som de palmas, o som de um aspirador não tem quaisquer parecenças com o som de apito, contudo sendo um som forte é conveniente analisa-lo como ruído.

O resultado da análise diz que de facto o aspirador é um som com elevada amplitude mas que praticamente é composto unicamente por uma sinusóide de 300 Hz, esta frequência não tem qualquer influência no espectro do som de apito, já que este possui as frequências de maior amplitude acima dos 2 kHz. O espectro do som do aspirador está representado na figura seguinte.

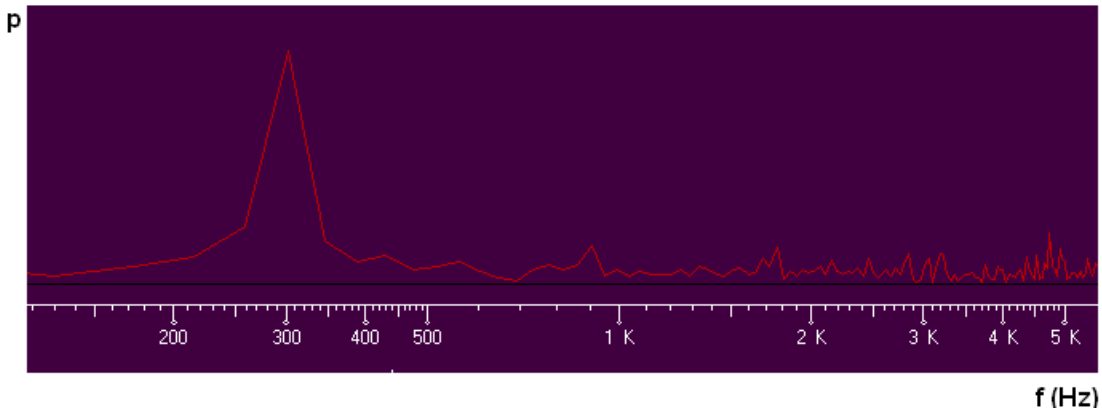


fig 25 – Espectro do som de um aspirador.

4.3 Conclusões da Análise de Sinais Sonoros

Primeiramente na análise à variação do som de apito com os vários factores verificou-se que o seu espectro varia muito com a intensidade de sopro e com a distância, isto leva a que o sistema tenha que possuir alguma tolerância ao fazer o reconhecimento de apito, já que a intensidade de sopro e a distância podem variar no mesmo reconhecimento.

A distinção entre 2 apitos é possível de ser feita, contudo torna-se difícil de o fazer com um bom grau de fiabilidade, já que existe pouca variação da frequência de pico de um apito para o outro e a tolerância que o sistema deve possuir, devido ao facto da variação de outros factores, é superior à variação entre os diferentes apitos.

Ruídos como o assobio normal e o aspirador não têm qualquer influência no reconhecimento de apito, já ruídos como o assobio de dedos e as palmas podem ter influência, porque podem possuir espectros com frequências de pico iguais à do som de apito, estes devem ser filtrados pela sua instabilidade, sendo o som das palmas bastante mais instável.

Capítulo 5 – Desenvolvimento do Projecto e Hardware

5.1 Desenvolvimento do Projecto

Nesta fase do projecto tem-se já o objectivo principal assim como objectivos secundários bem definidos, existe também um conhecimento dos projectos semelhantes já existentes, assim como, algoritmos usados no reconhecimento de fala e sons. Existe também um estudo acerca do tipo de sinal que se pretende reconhecer, e há também um conhecimento acerca de alguns possíveis ruídos que se possam encontrar. Então estão reunidos todos os factores para o começo do desenvolvimento pormenorizado do sistema de reconhecimento de apito.

Para definir o sistema em pormenor, é necessário tomar decisões quanto ao tipo de processamento de sinal a fazer, e como já foi visto no capítulo 2, os algoritmos de reconhecimento de fala não são adequados dada a sua complexidade computacional e também pelo facto que não são adequados a ambientes ruidosos. No capítulo 3, verificou-se a elevada complexidade de um sinal sonoro, este é composto por várias sinusóides somadas, de diferentes amplitudes e frequências, e o resultado dessa soma normalmente é um sinal que aparenta não obedecer a nenhuma formula matemática, este facto agrava-se quando o ruído é adicionado, por exemplo, um sinal sonoro de um som de apito com ruído, é a soma de todos os sons puros que compõe o sinal, olhando para um gráfico do sinal resultante não se consegue distinguir se existe um apito dada a complexidade do sinal resultante. Então a solução passa por calcular o espectro do sinal, ou seja, calcular as sinusóides que compõe o sinal ou por outras palavras calcular a amplitude do sinal em função da frequência. Este cálculo é feito através de uma transformada de Fourier, a transformada de Fourier na sua forma normal, implica que a entrada seja um sinal contínuo, e como este sistema é desenvolvido para ser aplicado em microcontrolador, esta transformada não pode ser aplicada, contudo existe a DFT – *Discret Fourier Transform*, trata-se de uma transformada de Fourier para sinais discretos, ou ainda a já falada FFT, que é uma versão da DFT, mas com um algoritmo de cálculo optimizado para reduzir a complexidade, necessitando de menos recursos para ser implementada, e diminuindo o tempo de execução. A FFT é a solução mais

adequada para microcontroladores, já que estes têm uma capacidade de cálculo muito limitada, sendo assim a escolhida para este projecto.

Após o cálculo do espectro, é necessário interpreta-lo, ou seja, "olhar" para o espectro e decidir se se trata de sinal de apito, é portanto necessário criar um programa de percepção para realizar esta tarefa.

Quanto à aquisição do sinal, esta tem que passar obrigatoriamente por um microfone, este deve ser pequeno, logo usa-se um microfone simples (sem quaisquer circuitos acoplados), isto implica que se tenha que acrescentar um circuito pré-amplificador. Como o microfone devolve um sinal analógico usa-se um ADC para o converter em digital, este ADC deve ser interno ao microcontrolador, para que não implique o aumento do volume do sistema. Por norma este tipo de sistema utiliza um filtro, passa-baixo ou passa-banda, entre a saída do pré-amplificador e a entrada do ADC, para poder eliminar alguns ruídos e ajudar a garantir o teorema de Nyquist que será falado mais à frente, contudo este sistema não possui este filtro, dado que no tipo de abordagem que se faz por software o incumprimento do teorema de Nyquist praticamente não causa problemas.

Tomada as decisões o sistema apresenta o diagrama de blocos representado na figura seguinte.

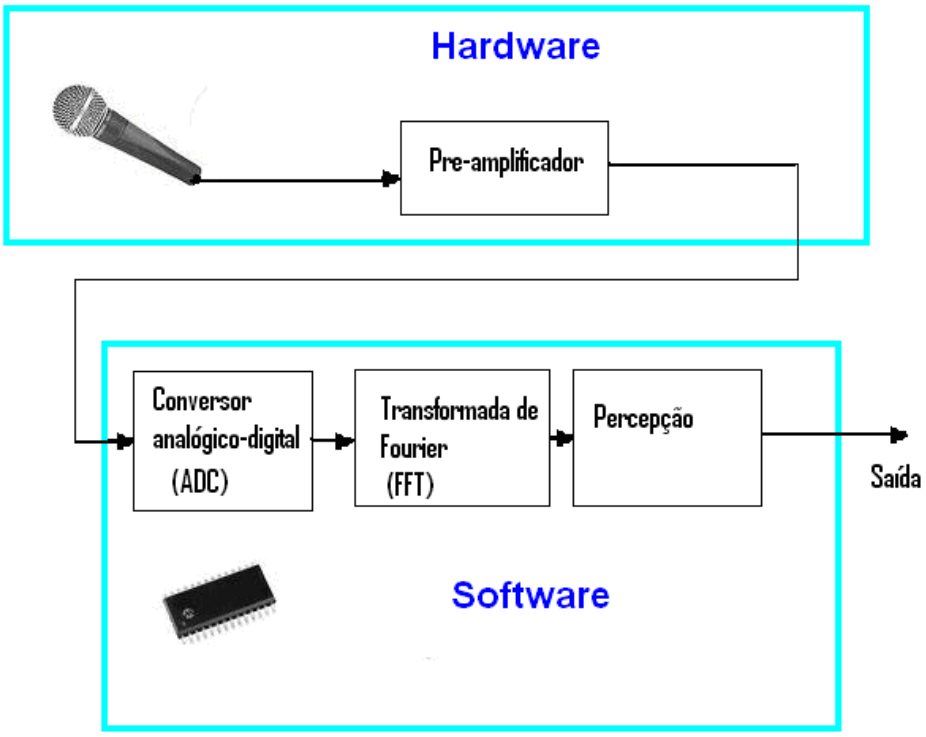


fig 26 – Diagrama de blocos do sistema.

5.2 Hardware

O hardware é parte fundamental na aquisição de sinal do sistema de reconhecimento de apito, este tem a função de captar o sinal sonoro e colocar um sinal eléctrico, proporcional ao sinal sonoro, na entrada do ADC. O hardware neste sistema engloba um microfone e um pré-amplificador. Todo o hardware deve ser minimizado quando projectado, de acordo com o objectivo de tornar o sistema resultante o mínimo volumoso possível.

5.2.1 Microfone

O microfone é um transdutor, dispositivo que converte som num sinal eléctrico [12].

O microfone é o primeiro elemento do sistema de reconhecimento de apito, fazendo a ligação entre o "mundo" e o sistema electrónico. Um sistema electrónico interpreta, gera ou altera sinais, mas apenas sinais eléctricos, se o objectivo é processar um sinal sonoro com um sistema electrónico, a primeira tarefa a fazer é transformar esse mesmo sinal sonoro num sinal eléctrico, é esta a função do microfone. Dito de outro modo, o microfone transforma energia sonora em energia eléctrica, gerando uma tensão eléctrica em função da pressão acústica que o rodeia. A figura seguinte ilustra a função do microfone, seguindo o exemplo de um som puro.

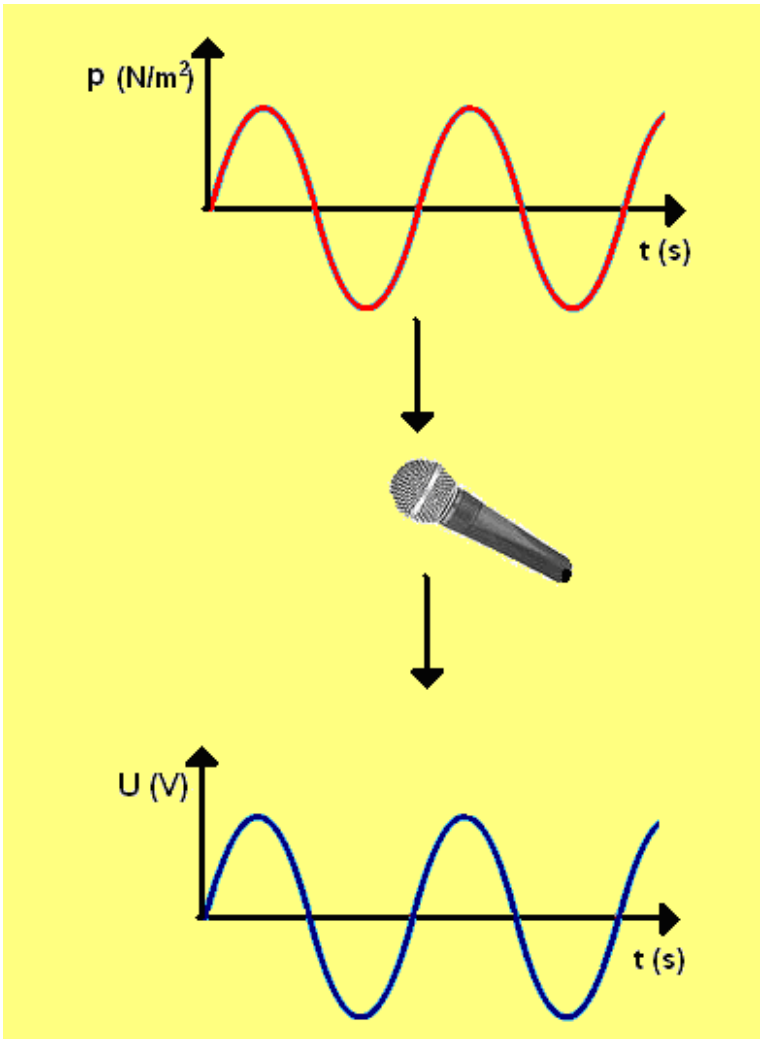


fig 27 – Funcionamento do microfone para um sinal sonoro puro.

Microfone do Tipo Condensador

O microfone escolhido para este projecto é um microfone do tipo condensador, tendo em conta que é um microfone de baixo custo, com um pequeno volume e que necessita apenas de um circuito pré-amplificador relativamente simples. Apesar da sua simplicidade, o microfone tipo condensador cumpre todos requisitos exigidos ao microfone do projecto de reconhecimento de apito, isto leva a que seja o escolhido para o projecto.

fig 28 – Microfone tipo condensador. (tirada de [14])

O microfone tipo condensador, funciona como um condensador cuja capacidade varia em função da pressão acústica a que está sujeito, seguido de um semicondutor com determinado ganho, que faz a ligação com o restante circuito. Para que possa gerar tensão em função da pressão tem que ser alimentado com uma tensão contínua, o funcionamento eléctrico deste tipo de microfone está representado na figura seguinte.

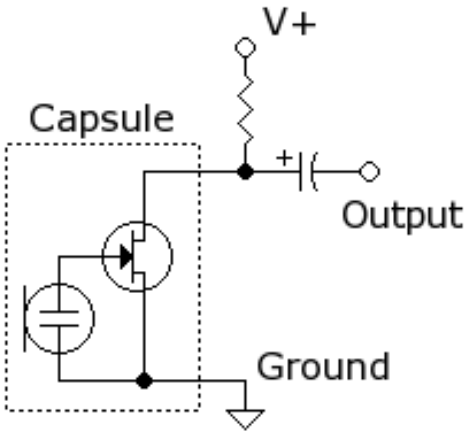


fig 29 – Funcionamento eléctrico do microfone tipo condensador. (tirada de [14])

Ganho do Microfone

Um microfone transforma pressão acústica em tensão eléctrica, então o ganho, também chamado de sensibilidade, deveria ser a relação entre a tensão em volts e a pressão acústica em Pascal (ou N/m^2), contudo esse ganho depende da tensão de alimentação, então usa-se um ganho relativo à tensão de alimentação, sendo a tensão de saída correspondente ao sinal sonoro dada de acordo com a seguinte fórmula:

$$v_o = V_i \times G \times p \Leftrightarrow G = \frac{v_o}{V_i \times p} \qquad \textbf{eq (3)}$$

v_o - Sinal de tensão de saída correspondente ao sinal sonoro em V

V_i - Tensão contínua de alimentação em V

p - Pressão acústica em pa

G – Ganho do microfone em pa^{-1}

É de notar que desta forma o ganho exprime-se em pa^{-1}, tal como se fosse calculado como o inverso da pressão, o que não corresponde ao verdadeiro cálculo, isto porque quando se divide a tensão de saída pela tensão de entrada, o resultado é adimensional, ou seja, é um ganho sem unidades. Então por convenção exprime-se esse ganho em dB, e o ganho do microfone passa a ser dado em dB/pa, calculando-se de acordo com a seguinte fórmula:

$$g = \frac{10\log(v_o / V_i)}{p} \quad , \qquad g \text{ - Ganho do microfone em } dB/pa \qquad \textbf{eq (4)}$$

O microfone utilizado tem um ganho de -45 dB/pa.

Resposta em Frequência

O ganho, ou sensibilidade, do microfone deve ser fixo, contudo este sofre pequenas variações com a variação da frequência. Esta variação da sensibilidade com a frequência faz com que seja adicionado um ganho indesejável sobre a própria sensibilidade. É este ganho indesejável que é analisado na resposta em frequência, ou seja, a resposta em frequência pode representar-se num gráfico da variação do ganho indesejável com a frequência.

O microfone tipo condensador utilizado tem uma gama de frequência de funcionamento nominal entre 50 Hz a 15kHz. Isto significa que a resposta em frequência nessa gama deve ser o mais próximo de 0 dB, já que é a gama onde o fabricante assegura um correcto funcionamento do microfone. Na figura seguinte representa-se graficamente a resposta em frequência do microfone utilizado.

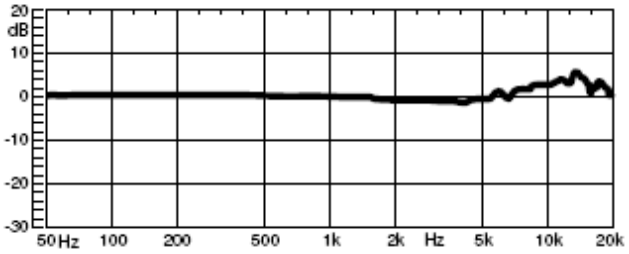


fig 30 – Resposta em frequência do microfone utilizado. (tirada de [13])

No gráfico da resposta em frequência verifica-se que o microfone tem um ganho indesejável de 0dB entre 50 Hz e 5kHz, acima de 5kHz esse ganho começa a ser ligeiramente superior. Contudo o ganho indesejável não provoca quaisquer efeitos neste projecto, porque lida-se com frequências apenas até 4kHz.

5.2.2 Circuito Pré-amplificador

O circuito pré-amplificador para este projecto, não tem que cumprir exigências muito rigorosas, no que diz respeito à qualidade de som, este apenas tem que possuir um ganho linear, devendo ser o mais simples possível, para ocupar pouco espaço no dispositivo final.

Os circuitos pré-amplificadores podem ser baseados em transístores ou em AMPOP's – Amplificador Operacional, sendo pré-amplificadores com transístores normalmente utilizados para aplicações mais simples, e os pré-amplificadores com AMPOP's normalmente utilizados para aplicações mais complexas onde se exige maior qualidade de som.

Visto que o pré-amplificador com transístor cumpre os requisitos de linearidade deste projecto, e visto que é mais simples, é o utilizado neste projecto. O esquemático do circuito está representado na figura seguinte.

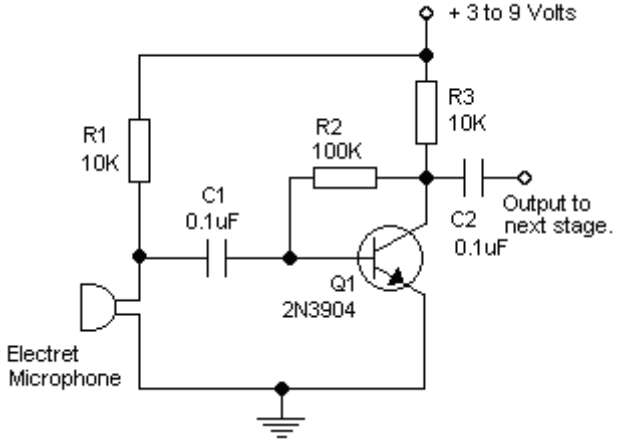


fig 31 – Circuito pré-amplificador. (tirada de [15])

Capítulo 6 – Microcontrolador e Software

A maior parte das tarefas do sistema de reconhecimento de Apito são feitas por software, tarefas essas que incluem o ADC, a FFT e o programa de percepção. O ADC tem a função de fazer a aquisição do sinal analógico, ou seja, transformar o sinal analógico em volts da saída do pré-amplificador em valores numéricos, ou seja, num sinal digital. A FFT trata-se do método utilizado para o cálculo espectral, por sua vez, ao programa de percepção cabe-lhe a tarefa de tomar a decisão final, ou seja, verificar se o espectro se trata ou não de um sinal de apito.

Todas as tarefas do software devem ser implementadas num único microcontrolador, logo este deve ser capaz de realizar as 3 tarefas. Na escolha do microcontrolador deve-se ter em conta a sua capacidade computacional, dado que a FFT contém cálculos bastante complexos, como se pode verificar no capítulo seguinte onde se descreve o seu cálculo. O microcontrolador deve também ter um ADC interno, de modo a se fazer a conversão analógica-digital por software, não sendo necessário adicionar qualquer componente.

6.1 Escolha do Microcontrolador

O microcontrolador trata-se de um pequeno dispositivo electrónico programável capaz de fazer o controlo de outros dispositivos, este possui uma unidade digital de processamento, onde faz todos os cálculos ordenados pelo programa que contém. É muito semelhante a um microprocessador, contudo enquanto que o microprocessador apenas faz o processamento, o microcontrolador tem já embebidos sistemas de memória e periféricos. A forma de um microcontrolador é um pequeno chip, que varia em número de pinos e em encapsulamento. A figura seguinte mostra os vários tipos de formas que pode ter um microcontrolador, neste caso para o fabricante Microchip.

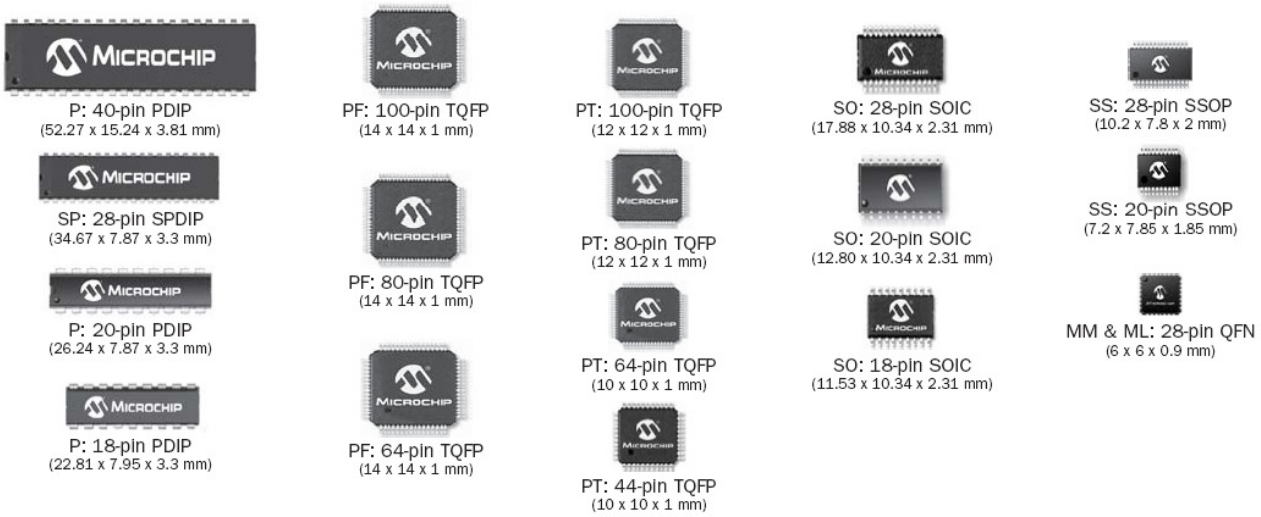


fig 32 – Vários formatos de microcontroladores. (tirada de [16])

O microcontrolador é o componente onde todas as tarefas realizadas por software vão ser efectuadas, este funciona como cérebro do dispositivo.

O desenvolvimento do software e os métodos para realização de cada tarefa em particular, dependem do tipo de arquitectura do microcontrolador usado, isto torna a escolha do microcontrolador uma tarefa de grande importância no desenvolvimento deste projecto. Normalmente os microcontroladores distinguem-se em 2 tipos:

- **MCU's (*Microcontroller Unit)*** – São microcontroladores simples, têm uma unidade de cálculo (ALU – *Arithmetic Logic Unit*) simples, que faz cálculos aritméticos e lógicos básicos, têm vários portos de entrada e saída, e vários tipos de periféricos. Estes dispositivos trabalham com velocidades de relógio na ordem das dezenas de MHz, e operam com registos de 8, 16 ou 32 bits.

- **DSP's (*Digital Signal Processor*) –** São microcontroladores cuja arquitectura está optimizada para fazer processamento de sinal em tempo real, incluindo a ALU, que consegue fazer cálculos bastante complexos num único ciclo de relógio. Trabalham com velocidades de relógio na ordem das centenas de MHz e normalmente são de 32 bits.

A utilização de um MCU para este projecto não é adequada, dado que a maioria dos MCU's não possuem capacidade de cálculo suficiente para realizar as operações

pretendidas. Mesmo escolhendo um MCU de 16 ou 32 bits que possua capacidade de cálculo para fazer as operações, estes dispositivos não estão optimizados para fazer operações de processamento de sinal, o que levava a uma baixa performance na execução das tarefas deste projecto, e tornaria a programação mais elaborada o que leva a que tenha uma maior probabilidade de erro.

Os DSP's têm em geral uma capacidade de cálculo bastante mais elevada que os MCU's, e têm também uma arquitectura adequada às tarefas que se pretendem efectuar neste projecto, conseguindo-se obter maior performance e probabilidades de erro bastante reduzida. Contudo são dispositivos normalmente mais caros, e que necessitam de placas de desenvolvimento para serem programados, sendo estas placas normalmente também muito caras, isto levaria a um forte aumento no preço do projecto. Não sendo os DSP's hipótese descartada de imediato, este factor levou a que fosse procurada outra solução. Nesta procura foi encontrado outro tipo de microcontrolador, os DSC's:

- **DSC's (*Digital Signal Controller*)** – São dispositivos híbridos, possuem arquitectura de MCU's e arquitectura de DSP's incorporadas no mesmo chip, funcionam com velocidades de relógio em geral um pouco superiores aos MCU's e operam com registos de 16 bits.

Os DSC's não são tão usados quanto os MCU's e os DSP's, visto que são dispositivos relativamente recentes (começaram a existir em 2004), e existem poucos fabricantes deste tipo de microcontroladores, apenas os fabricantes Microchip, Texas Instruments e Freescale possuem séries de DSC's, sendo os mais populares, os fabricados pela Mcrochip, com as séries dsPIC30F e dsPIC33F. Na figura seguinte está representado a variação da performance em função do preço comparando os DSC's da Microchip com os MCU's e DSP's em geral.

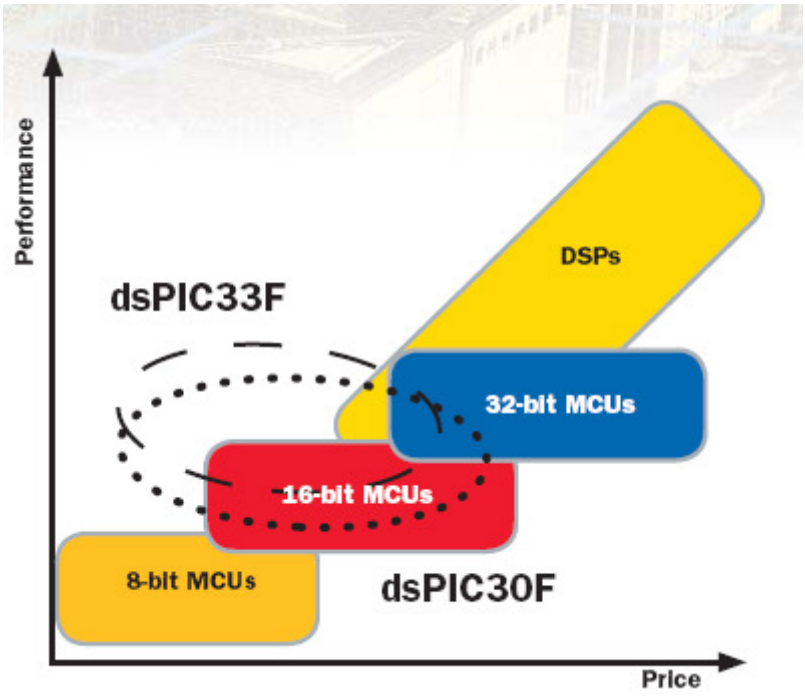


fig 33 – Performance em função do preço dos DSC's da Microchip comparativamente aos MCU's e DSP's em geral. (tirada de [17])

Como se pode ver na figura, usando-se um DSC, consegue-se ter boa performance a preços relativamente baixos. Visto que estes dispositivos tem também arquitectura de processamento de sinal, são também adequados para este projecto tal como os DSP's, com a vantagem que não necessitam de placas de desenvolvimento demasiado complexas, os DSC's são programados com um simples programador tal como os MCU's, sendo estes programadores normalmente bastante mais baratos que as placas de desenvolvimento dos DSP's. Dado não haver inconvenientes com o uso deste tipo de microcontroladores neste projecto, o DSC é a solução escolhida para fazer as tarefas de software.

Escolhido o tipo de microcontrolador a utilizar, é necessário optar por um fabricante e um modelo. Quanto ao fabricante, optou-se por usar um DSC da **Microchip**, tendo em conta os seguintes motivos:

- É o fabricante com a maior gama de DSC's;
- Os seus DSC's são optimizados para o uso de linguagem C.
- O software de programação é livre, e existem versões livres dos compiladores de C.

- A Microchip envia amostras gratuitas de vários tipos de microcontroladores para estudantes, com as quais pode ser feito o desenvolvimento do projecto;
- São os DSC's mais utilizados, o que torna a sua aprendizagem mais fácil e mais rápida, visto que se encontra facilmente exemplos de código.

Entre os DSC's da Microchip, escolheu-se o modelo **dsPIC30F4013** dado que é um modelo com boa capacidade de cálculo, e na qual o fabricante envia amostras com o encapsulamento PDIP, este encapsulamento é o único que permite o encaixe do microcontrolador directamente em *breadboard*, o tipo de placas que vão ser utilizadas no desenvolvimento do projecto.

6.2 Microcontrolador dsPIC30F4013

O dsPIC30F4013 é um microcontrolador de 16 bits fabricado pela Microchip, as suas características gerais mais relevantes são as seguintes:

- Arquitectura *Harvard*;
- Instruções de 24 bits com o tamanho do *opcode* variável;
- Instruções optimizadas para linguagem C;
- 48 kB de memória de código (memória Flash);
- 2 kB de memória RAM interna;
- 1kB de memória EEPROM;
- Até 33 fontes de interrupção;
- Operação até 120 MHz (30 MIPS - *Million Instructions Per Second*);
- Interface com vários protocolos de comunicação (ex.: I^2C e porta série);
- ADC de 12 bits com 13 canais de entrada e frequência de amostragem até 200 ksps;
- Hardware próprio para processamento digital de sinal (ex.: somador de 40 bits e multiplicador 17 × 17 bits).

Na referência [18] encontra-se o manual de referência da família de microcontroladores dsPIC30F, este manual descreve com o máximo detalhe toda a arquitectura dos microcontroladores dessa família. Na referência [19] encontra-se o

datasheet do microcontrolador dsPIC30F4013, com os detalhes necessários à sua utilização.

Na figura seguinte representa-se o diagrama de blocos de toda a arquitectura em geral, para todos os microcontroladores da família dsPIC30F, onde se pode visualizar o enquadramento do hardware de processamento de sinal na restante arquitectura do microcontrolador.

NOTA: Os valores de memória da figura seguinte dizem respeito aos valores máximos para a família dsPIC30F, sendo atingidos apenas pelos microcontroladores de topo dessa família, esses valores para o microcontrolador utilizado neste projecto são inferiores. Os verdadeiros valores podem ser encontrados na parte de organização da memória deste documento, ou no respectivo datasheet do microcontrolador [19]).

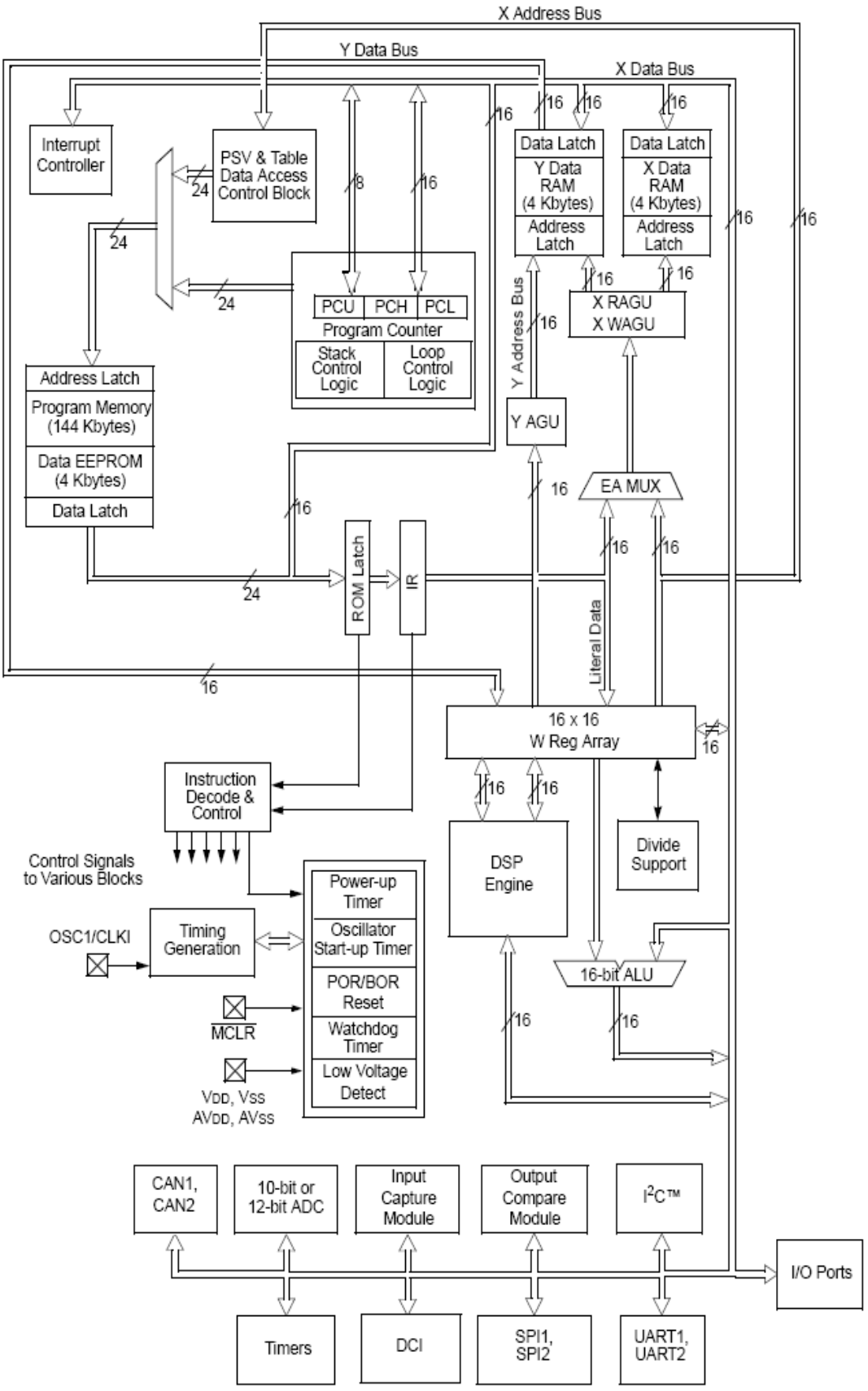


fig 34 – Diagrama de blocos dos microcontroladores da família dsPIC30F. (tirada de [18])

Como se pode verificar na figura anterior, paralelamente à ALU do microcontrolador existe um bloco chamado '*DSP Engine'*, esse bloco contém o hardware para processamento de sinal, ou seja, trata-se de uma ALU mais complexa para fazer operações matemáticas também mais complexas exigidas quando se pretende fazer processamento de sinal, e que feitas com a ALU convencional necessitariam de muitos ciclos de relógio para serem efectuadas, além de ocuparem outros recursos do microcontrolador. Então usando-se instruções especiais, pode-se fazer o cálculo nesta unidade em vez de fazer na ALU convencional. A figura seguinte mostra o conteúdo do bloco '*DSP Engine'*.

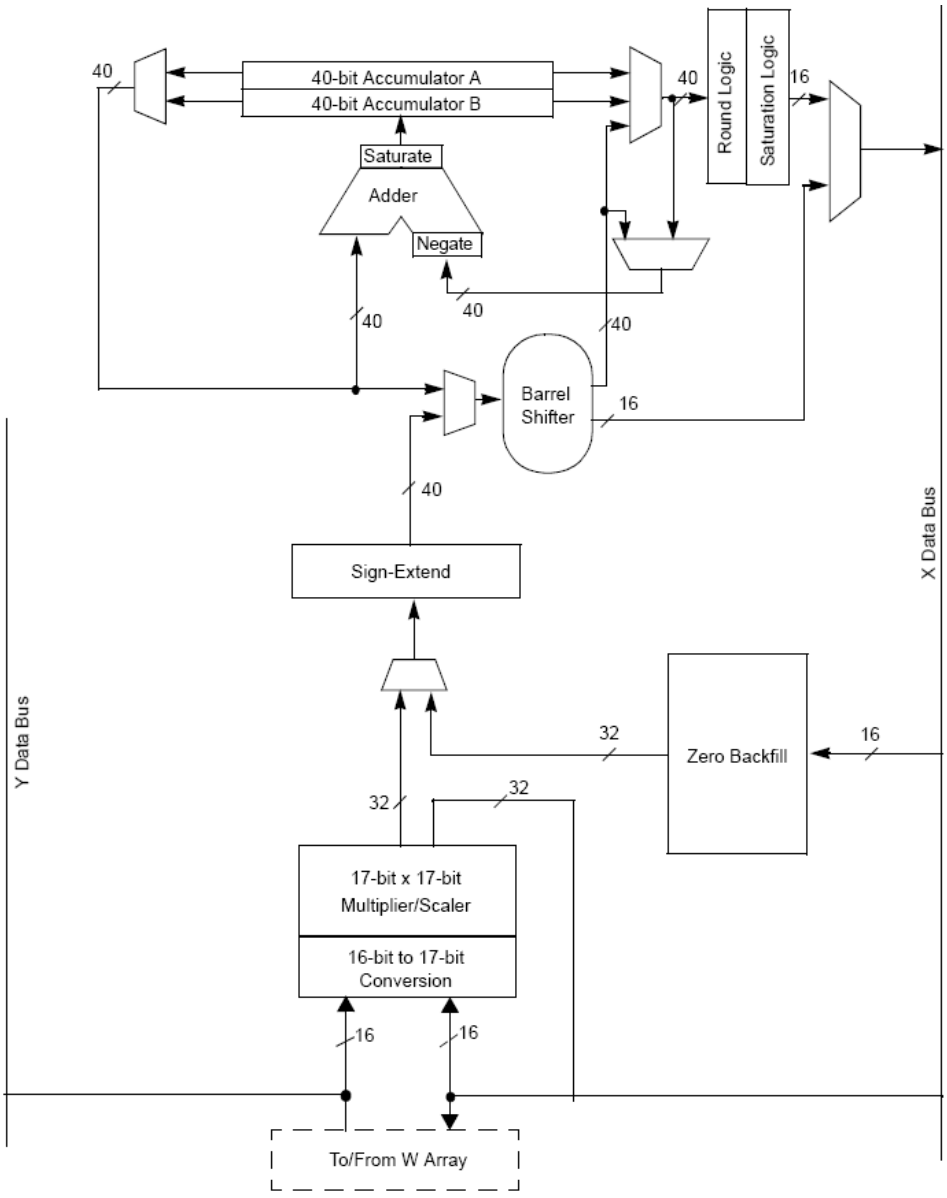


fig 35 – Diagrama de blocos da unidade de processamento digital de sinal ('*DSP Engine'*). (tirada de [18])

6.2.1 Organização da Memória

O microcontrolador dsPIC30F4013 possui uma arquitectura *Harvard*, isto significa que há uma separação entre memória de código e memória de dados, ou seja, existe um barramento para memória de código e outro para memória de dados. Neste microcontrolador a memória de dados está dividida em memória X e memória Y, devido a este facto tem adicionado outro barramento, de acordo com a figura 25.

A memória de dados é constituída por uma memória RAM de 2 kB, dos quais 1 kB compõe a memória X, e o restante 1 kB compõe a memória Y. À memória de dados, pode ser adicionada opcionalmente 32 kB que pertencem à memória de código (memória flash), esta memória pode ser acedida com funções especiais e pode ser usada apenas para guardar constantes. Pode-se se usar esses 32 kB da memória de código como memória de dados para constantes, como sendo parte da própria memória de dados, não necessitando das funções especiais para ser acedida, para tal deve-se accionar o modo de operação PSV – *Program Space Visibility*. Na figura seguinte representa-se a organização da memória de dados do microcontrolador utilizado neste projecto.

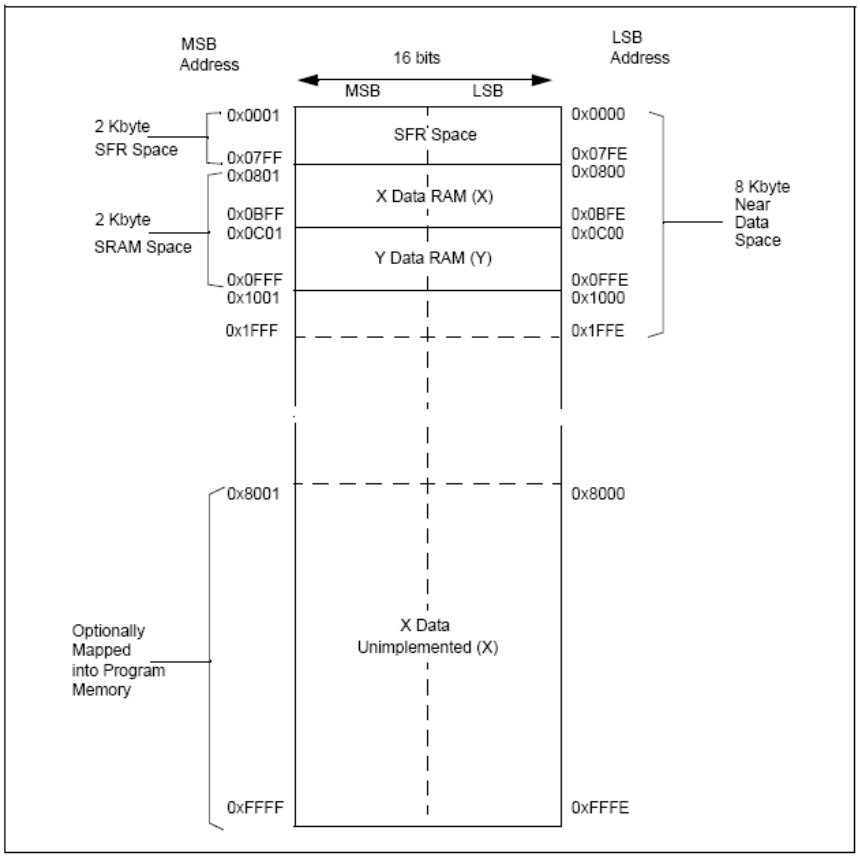


fig 36 – Organização da memória de dados no microcontrolador dsPIC30F4013. (tirada de [19])

A memória de código é do tipo memória flash, e tem uma capacidade de 48 kB, sendo as instruções de 24 bits (= 3 bytes), esta permite escrever até 16k (16384) instruções. No caso de se usar o modo PSV, os últimos 32 kB da memória de código passam a poder ser utilizados para dados. Para além da memória flash destinada a guardar código, existe uma memória EEPROM de 1kB também mapeada na memória de programa, esta memória pode ser usada para guardar dados e mante-los quando a alimentação do microcontrolador é desligada, já que é uma memória não volátil. Na figura seguinte representa-se todo o mapa da memória de programa do dsPIC30F4013 onde estão incluídas as memórias flash e EEPROM.

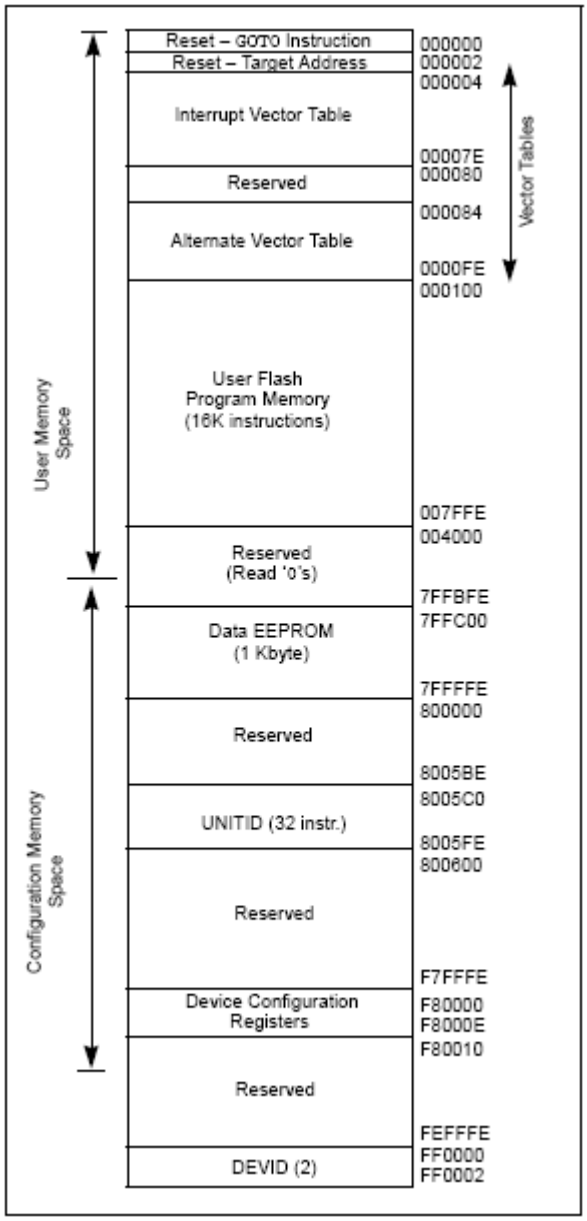


fig 37 – Organização da memória de programa do microcontrolador dsPIC30F4013. (tirada de [19])

6.2.2 Linguagem de programação

Os microcontroladores executam instruções de baixo nível, ou seja, são programados numa linguagem de baixo nível, chamada assembler. As instruções assembler estão ao nível dos registos e endereços, tendo cada microcontrolador o seu próprio conjunto de instruções (I*nstruction Set*). Isto faz com que determinada tarefa tenha diferente implementação em diferentes microcontroladores. As instruções de baixo nível assembler que compõem o programa estão escritas na memória de código, no caso do microcontrolador dsPIC30F4013 são instruções de 24 bits.

O uso de linguagem de baixo nível tem vantagens no que diz respeito à manipulação de variáveis, já que se consegue manipular ao bit. Contudo dependendo complexidade da tarefa, o uso deste tipo de linguagem pode não ser adequado, visto que as instruções assembler normalmente só manipulam posições de memória, fazem somas ou subtracções, e dependendo do microcontrolador podem também fazer multiplicações. Isto torna as instruções de baixo nível demasiado básicas para realizar tarefas mais complexas, visto que são necessárias muitas instruções para a realização dessas tarefas, e sendo elas escritas por um programador humano, a probabilidade de haver erros é muito elevada.

Este projecto envolve cálculos de processamento de sinal, estes cálculos envolvem números imaginários ($a + j.b$), o que os torna demasiado complexos para serem implementados em baixo nível. Por outro lado vai ser usado um microcontrolador para fazer esses cálculos, onde é necessário fazer manipulações ao bit, para configurar registos ou optimizar cálculos. Então torna-se necessário usar uma linguagem onde se consiga trabalhar em alto nível e baixo nível, a linguagem que se enquadra melhor nestas características é a **linguagem C**. É uma linguagem de baixo nível e alto nível, que consegue trabalhar com funções de alto nível como também consegue fazer manipulações de baixo nível, daí a sua escolha para este projecto.

6.2.3 Programação do microcontrolador dsPIC30F4013 em linguagem C

Como já foi visto, o microcontrolador só executa instruções assembler, então para que o programador possa escrever o código em linguagem C, é necessário um

mecanismo que transforme a linguagem C escrita pelo programador em linguagem assembler para o microcontrolador, esse mecanismo é chamado de compilador. O compilador usado para programar o microcontrolador usado neste projecto é o Microchip MPLAB C30 (versão estudante). Este compilador trata-se de um software que corre sobre a plataforma de programação dos microcontroladores da Microchip (MPLAB IDE). Esta plataforma envia o código assembler para a memória de código do microcontrolador através de um programador, neste caso foi usado o Microchip MPLAB ICD2.

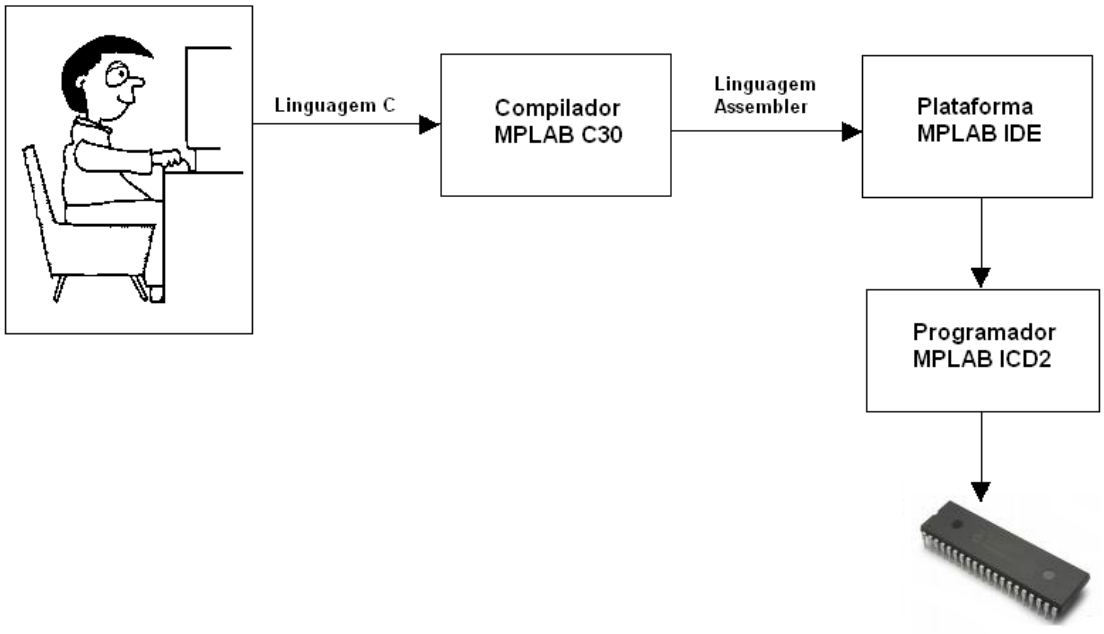


fig 38 – Passos na programação do microcontrolador

Capítulo 7 – Conversor Analógico-Digital (ADC)

O conversor Analógico-Digital tem a tarefa de transformar o sinal analógico de entrada x(t) em valores numéricos que possam ser interpretados pelo microcontrolador, ou seja, um sinal digital x[n] (n=1,2,3...).

O sinal analógico que liga à entrada do ADC é o sinal de tensão que está na saída do pré-amplificador. O sinal de saída do ADC será guardado num vector na memória do microcontrolador.

A conversão analógico-digital é feita em duas fases, a primeira é a **amostragem**, onde se representa o sinal analógico de entrada por um sinal discreto. A segunda fase é a **conversão**, onde o sinal amostrado vai ser convertido num sinal digital.

O microcontrolador usado tem ADC interno, deste modo evita o uso de hardware adicional para fazer esta tarefa, o ADC do microcontrolador tem as seguintes características:

- Amostragem do tipo *Sample and Hold*;
- Conversão feita pelo método Aproximações Sucessivas;
- Resolução de 12 bits;
- Frequência de amostragem até 200 ksps;
- 16 entradas analógicas;
- Acumulador de 16 palavras;
- Tensão de referência externa.

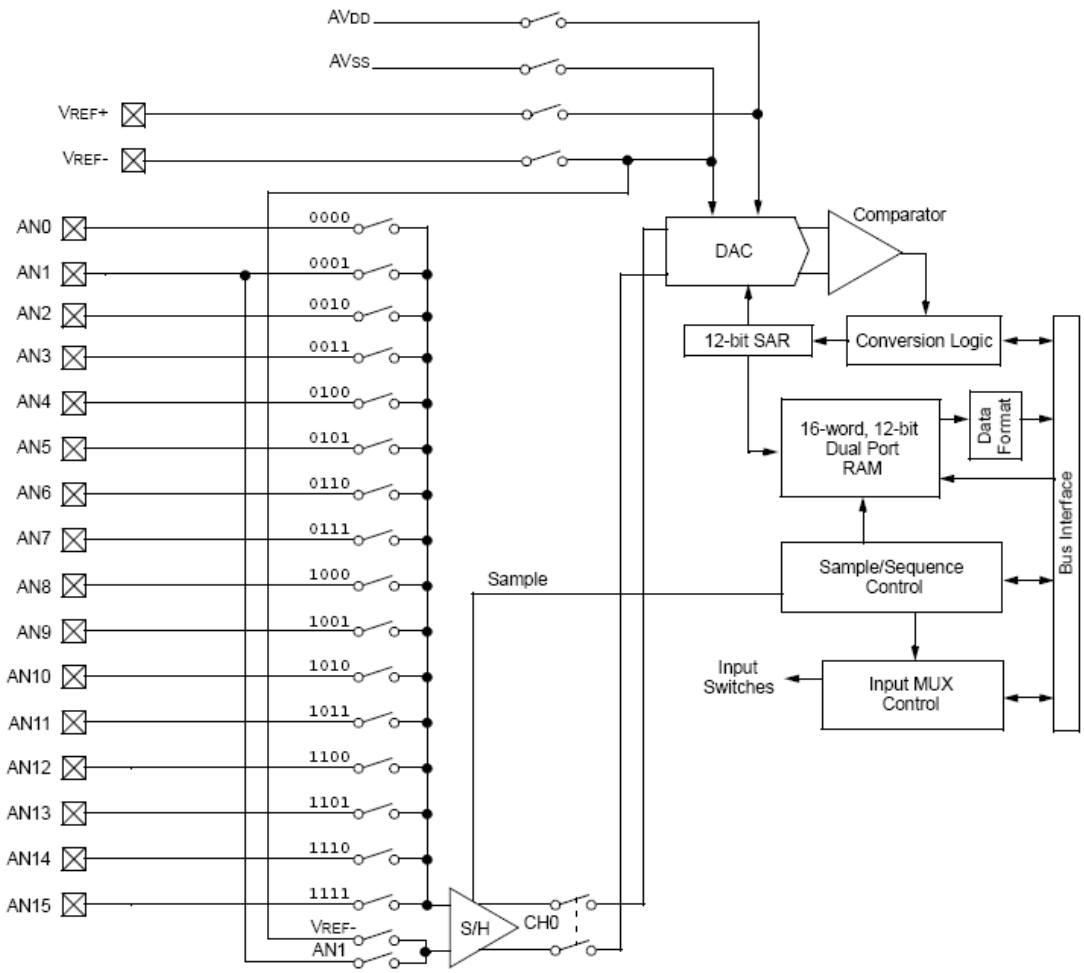
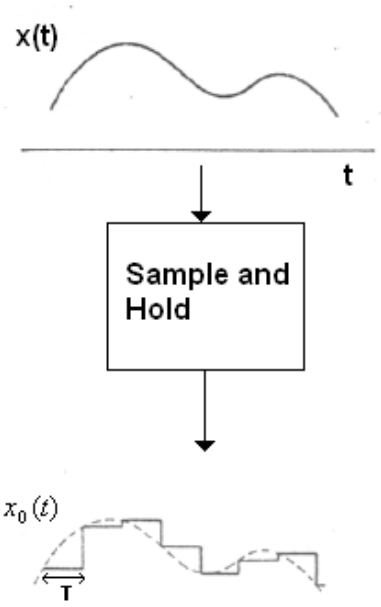


fig 39 – Diagrama de blocos do ADC do microcontrolador dsPIC30F4013. (tirada de [18]

7.1 Amostragem

A amostragem neste projecto é do tipo **S*ample and Hold*** de acordo com o ADC do microcontrolador utilizado, este tipo de amostragem toma o valor do sinal analógico no inicio do tempo de amostragem, e retém esse mesmo valor até que acabe o tempo de amostragem e comece outro tempo. A figura seguinte mostra o funcionamento deste tipo de amostragem.



Onde:

$$f_s = \frac{1}{T} \qquad \textbf{eq (5)}$$

T – Período de amostragem em s

f_s - Frequência de amostragem em Hz ou sps

t – Tempo em s

$x(t)$ – Sinal de entrada em V

$x_0(t)$ - Sinal amostrado em V

fig 40 – Principio de funcionamento da amostragem S*ample and Hold.* (Adaptada da aula 4 da referência [20])

7.1.1 Teorema de Nyquist

Segundo o teorema de Nyquist, a frequência de amostragem deve ser maior que o dobro da frequência máxima do sinal a ser amostrado.

$$f_s > 2f_M \qquad \textbf{eq (6)}$$

Onde:

f_s - Frequência de amostragem em Hz ou sps

f_M - Frequência máxima do sinal analógico em Hz

Uma amostragem correcta deve cumprir o teorema de Nyquist, só respeitando esta lei se consegue fazer a amostragem do sinal sem perda de informação do mesmo. Caso não se cumpra o teorema de Nyquist, todas as frequências do sinal analógico que estiverem acima da frequência máxima permitida serão mal interpretadas, sendo consideradas de frequência inferior, a este efeito chama-se de *aliasing* ou sub-amostragem.

Quando ocorre *aliasing*, a frequência do sinal amostrado, que deveria ser igual ao do sinal analógico, toma valores segundo a seguinte fórmula:

$$f' = |f - f_s| \qquad \textbf{eq (7)}$$

Onde:

f' - Frequência do sinal amostrado em Hz

f - Frequência do sinal analógico em Hz

f_s - Frequência de amostragem em Hz ou sps

Na figura seguinte dá-se o exemplo de uma amostragem de uma sinusóide com *aliasing*, neste caso a amostragem não é feita por *sample and hold*, mas por trem de impulsos.

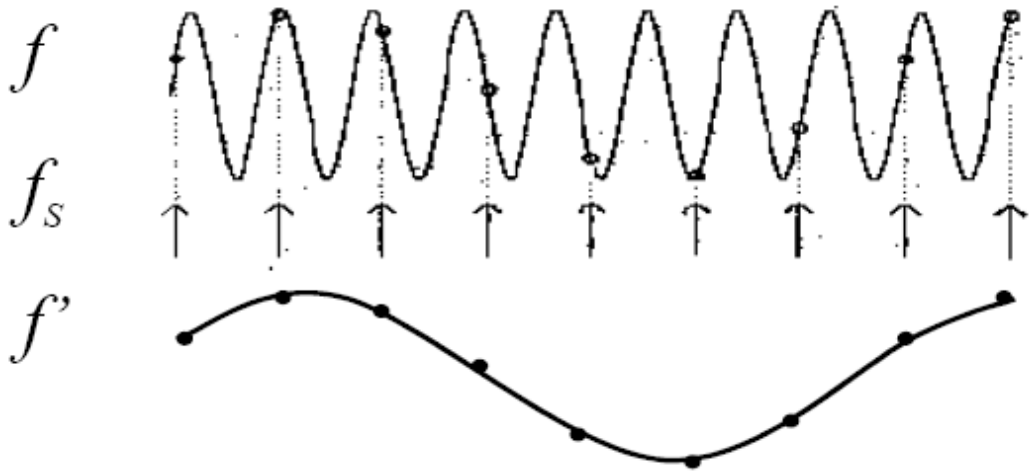


fig 41 – Efeito da sub-amostragem (*aliasing*). (tirada de [22])

Fazendo-se uma análise em frequência, a frequência do sinal amostrado é uma reflexão da frequência do sinal analógico sobre a frequência de amostragem, como mostra a figura seguinte.

NOTA: A análise em frequência será abordada com mais detalhe no capítulo 8.

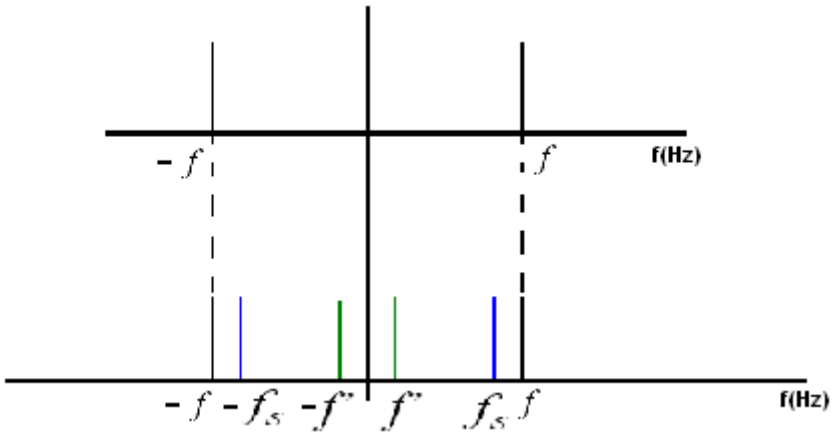


fig 42 – Análise em frequência do efeito da sub-amostragem.

7.2 Conversão

A conversão tem a função de transformar o valor em V de cada amostra do sinal, num valor numérico correspondente, de modo a formar um sinal digital. O ADC do microcontrolador dsPIC30F4013 é do tipo **Aproximações Sucessivas**. Este método de conversão baseia-se num princípio de comparação, onde começa a atribuir o valor ao bit mais significativo, e depois de comparar atribui ao bit seguinte, até chegar ao bit menos significativo. O conversor deste tipo é composto por um comparador de sinais analógicos, um DAC – *Digital to Analog Converter* e um bloco lógico. O comparador tem o objectivo verificar qual o sinal maior entre o sinal de entrada (uma amostra do sinal $x_0(t)$) e o sinal convertido após cada bit. O DAC transforma o sinal convertido após cada bit num sinal analógico, para que possa ser comparado no comparador analógico, e o bloco lógico atribui as operações a fazer de acordo com o resultado da comparação. A constituição e funcionamento do ADC estão ilustrados nas figuras seguintes. Onde:

A – Sinal analógico de entrada;
D – Valor analógico gerado pelo DAC;
B_n - Bit mais significativo;
B_0 - Bit menos significativo.

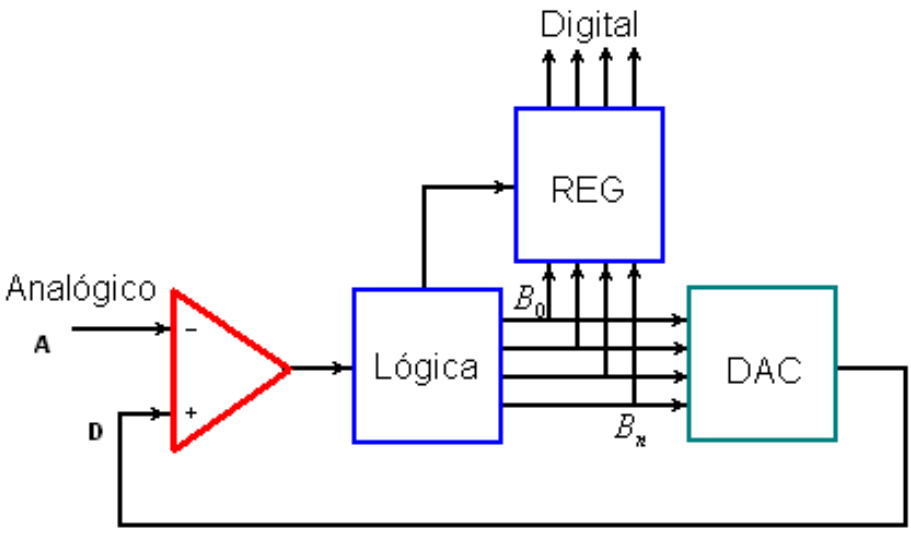


fig 43 – Diagrama de blocos de um ADC de aproximações sucessivas. (adaptada de [21])

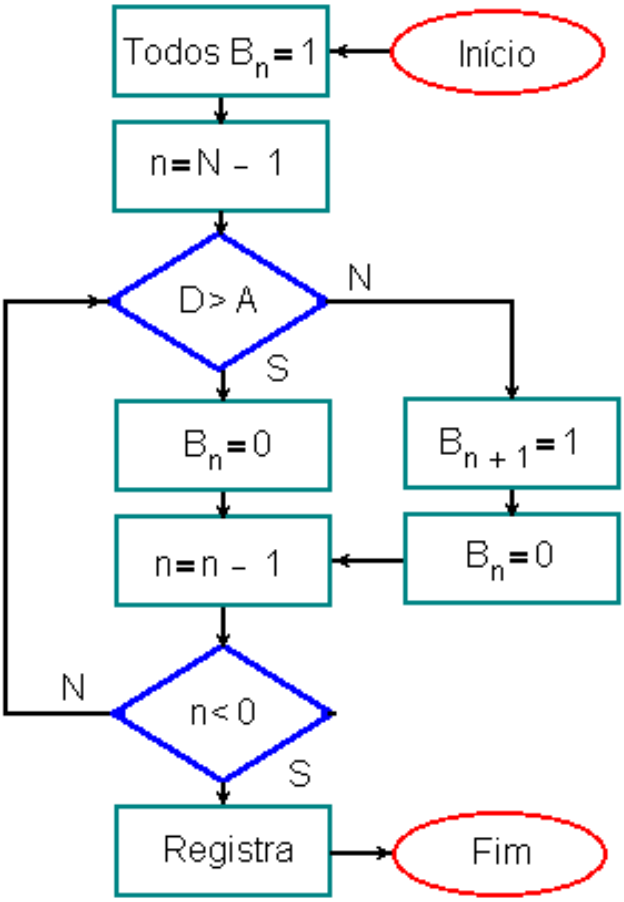


fig 44 – Fluxograma do funcionamento de um ADC de aproximações sucessivas. (tirada de [21])

7.3 Escolha da frequência de amostragem e número de amostras

Devido às características do microcontrolador, o ADC neste projecto vai ser realizado com uma amostragem *sample and hold* e uma conversão do tipo aproximações sucessivas, com uma resolução de 12 bits. Deve ser escolhida uma frequência de amostragem (f_s) adequada ao projecto, assim como o número de amostras por cada iteração do ADC.

Para a escolha da frequência de amostragem tem que ser feita uma verificação da frequência máxima do apito, e calcular a respectiva frequência de amostragem de acordo com o teorema de Nyquist (eq. 6). Esse valor de frequência máxima deve ser obtido de acordo com as análises a sinais do capítulo 4. Nessas análises verifica-se que o sinal de apito tem dois picos, um que varia entre 2.1 e 2.9 kHz e outro acima dos 4 kHz, contudo este último é de amplitude baixa e demasiado sensível à distância entre

microfone e apito, para distâncias grandes este praticamente deixa de existir. Então o segundo pico não vai ser considerado, e a frequência máxima deve ter em conta as frequências do primeiro pico. Estas frequências podem ser observadas na figura seguinte.

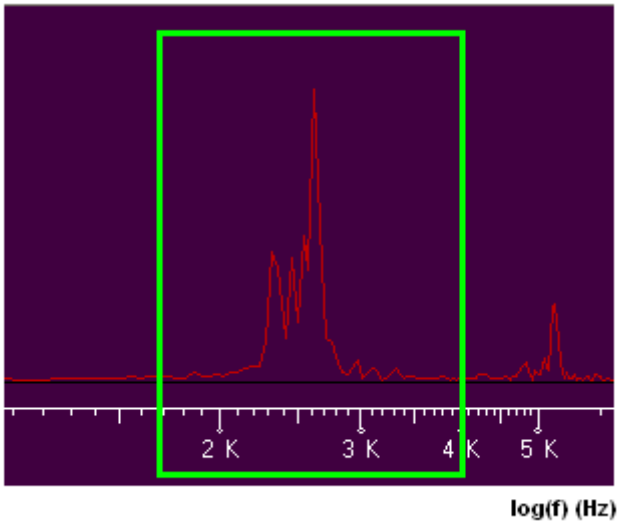


fig 45 – Espectro de um sinal de apito focando as frequências do primeiro pico.

Observando a figura anterior verifica-se que, excluindo o segundo pico, as frequências a partir de 3 kHz têm amplitude praticamente nula, contudo, visto não ser necessário muito rigor na escolha da frequência de amostragem, optou-se por considerar a frequência máxima do sinal 4 kHz, valor a partir do qual os sinais devem ter amplitude nulas. Então a frequência de amostragem usada é a frequência de Nyquist:

$$f_s = 2f_M \Leftrightarrow f_s = 2 \times 4000 = 8000 \rightarrow 8kHz$$

O número de amostras N, deve ser escolhido tendo em conta o bloco seguinte, ou seja, o cálculo do espectro (FFT), quanto maior for N, maior será a resolução da FFT, contudo se N for demasiado elevado o cálculo da FFT vai exigir mais memória e mais tempo. Para se fazer uma FFT, o número de pontos N tem que ser do tipo $2^n (n = 1,2,3...)$. Após alguns testes com o número de amostras no cálculo da FFT no microcontrolador usado, chegou-se à conclusão que o este tem boa capacidade para fazer a FFT até 256 amostras, deixando memória suficiente para a realização das outras tarefas, então optou-se por fazer uma aquisição de 256 amostras.

7.4 Implementação do ADC no Microcontrolador dsPIC30F4013

7.4.1 Modo de operação

Optando-se por uma conversão analógico-digital de 256 amostras a 8 kHz, é necessário estabelecer ainda alguns parâmetros e modos de operação no microcontrolador.

O ADC interno do dsPIC30F4013 permite guardar até 16 amostras (*buffer* de 16 palavras), visto que se pretende adquirir 256 amostras, é necessário usar um modo de operação que guarde no *buffer* 16 amostras e quando este encher chame uma interrupção para que se possa guardar essas amostras num vector onde ficarão todas as 256 amostras. Para accionar este modo de operação tem que se colocar os 4 bits SMPI do registo ADCON2 a nível '1'. Desta forma, de acordo com o manual de utilização [18] do microcontrolador, o modo de operação passa a ser o seguinte:

Buffer Address	Buffer @ 1st Interrupt	Buffer @ 2nd Interrupt
ADCBUF0	AN0 sample 1	AN0 sample 17
ADCBUF1	AN0 sample 2	AN0 sample 18
ADCBUF2	AN0 sample 3	AN0 sample 19
ADCBUF3	AN0 sample 4	AN0 sample 20
ADCBUF4	AN0 sample 5	AN0 sample 21
ADCBUF5	AN0 sample 6	AN0 sample 22
ADCBUF6	AN0 sample 7	AN0 sample 23
ADCBUF7	AN0 sample 8	AN0 sample 24
ADCBUF8	AN0 sample 9	AN0 sample 25
ADCBUF9	AN0 sample 10	AN0 sample 26
ADCBUFA	AN0 sample 11	AN0 sample 27
ADCBUFB	AN0 sample 12	AN0 sample 28
ADCBUFC	AN0 sample 13	AN0 sample 29
ADCBUFD	AN0 sample 14	AN0 sample 30
ADCBUFE	AN0 sample 15	AN0 sample 31
ADCBUFF	AN0 sample 16	AN0 sample 32

fig 46 – Modo de operação do ADC escolhido neste projecto. (tirada de [18])

De modo a se fazer a aquisição de 256 amostras, são necessárias 16 interrupções. Isto implica o uso de um contador para que se dê a ordem de paragem quando forem atingidas as 16 interrupções e consequentemente 256 amostras.

7.4.2 Definição da frequência de amostragem

Para de definir a frequência de amostragem é necessário calcular o período de amostragem (eq. 5):

$$f_s = \frac{1}{T} \Leftrightarrow T = \frac{1}{f_s} \Leftrightarrow T = \frac{1}{8000} = 125 \times 10^{-3} s \rightarrow 125 \mu s$$

Então para se definir uma frequência de amostragem optou-se por colocar o valor do período de amostragem no *timer 3*, esta operação faz-se com a configuração do registo PR3, de acordo com a seguinte fórmula:

$$PR3 = \frac{T}{T_{CY}} \qquad \textbf{eq (8)}$$

Onde:

PR3 – Valor para definir o tempo do *timer 3*

T – Período de amostragem em s

T_{CY} – Tempo de execução de uma instrução em s ($T_{CY} = 33.9ns$)

O valor a colocar no registo PR3 para obter um período de amostragem de 125 µs e uma respectiva frequência de amostragem de 8 kHz é:

$$PR3 = \frac{T}{T_{CY}} \Leftrightarrow PR3 = \frac{125 \times 10^{-6}}{33.9 \times 10^{-9}} = 3687$$

7.4.3 Definição do tempo de conversão

A conversão deve ser feita enquanto o sinal está a ser amostrado, devendo o tempo de conversão ser menor que o tempo de amostragem.

Existe um módulo de relógio que controla a conversão, esse módulo tem um tempo de relógio chamado T_{AD}, a conversão necessita de um tempo de $14 \times T_{AD}$, podendo o valor de T_{AD} ser escolhido através dos 6 bits ADCS do registo ADCON3, segundo a seguinte fórmula:

$$T_{AD} = \frac{T_{CY}(ADCS+1)}{2} \qquad \textbf{eq (9)}$$

Onde:

ADCS – Valor de 6 bits que define o tempo de conversão

T_{AD} – Tempo do relógio de conversão em s

T_{CY} – Tempo de execução de uma instrução em s ($T_{CY} = 33.9ns$)

Optou-se por usar o tempo de conversão máximo, ou seja, com os bits ADCS todos a nível '1', o que equivale ao valor 63 em decimal. Desta forma obtém-se um T_{AD} de:

$$T_{AD} = \frac{T_{CY}(ADCS+1)}{2} = \frac{33.9\times10^{-9}(63+1)}{2} = 1.0848\mu s$$

A conversão é feita em 14 ciclos T_{AD} ($14\times T_{AD} = 15.19\mu s$), obtendo-se então o tempo de conversão. Pode-se usar este tempo de conversão visto que é menor que o período de amostragem (15.19 µs <125 µs).

7.4.4 Definição do formato de dados

O microcontrolador dsPIC30F4013 trabalha com registos de 16 bits, e possui um ADC com resolução de 12 bits. Então é permitida a escolha do formato de dados, ou seja, o modo como os 12 bits se enquadram nos 16 bits. Neste projecto por conveniência com o tipo de variáveis usadas nos blocos seguintes optou-se por o formato de dados "*signed fractional* " ou *"1.15 fractional"* onde os últimos 4 bits são colocados a '0' e os 12 primeiros bits tomam o valor do ADC, sendo o primeiro bit de sinal.

Signed fractional (sddd dddd dddd 0000)

Para configurar o formato de dados para "*signed fractional*" coloca-se os 2 bits FORM do registo ADCON1 a nível '1'.

7.5 Algoritmo

Tomadas as decisões o ADC funciona de acordo com o seguinte algoritmo:

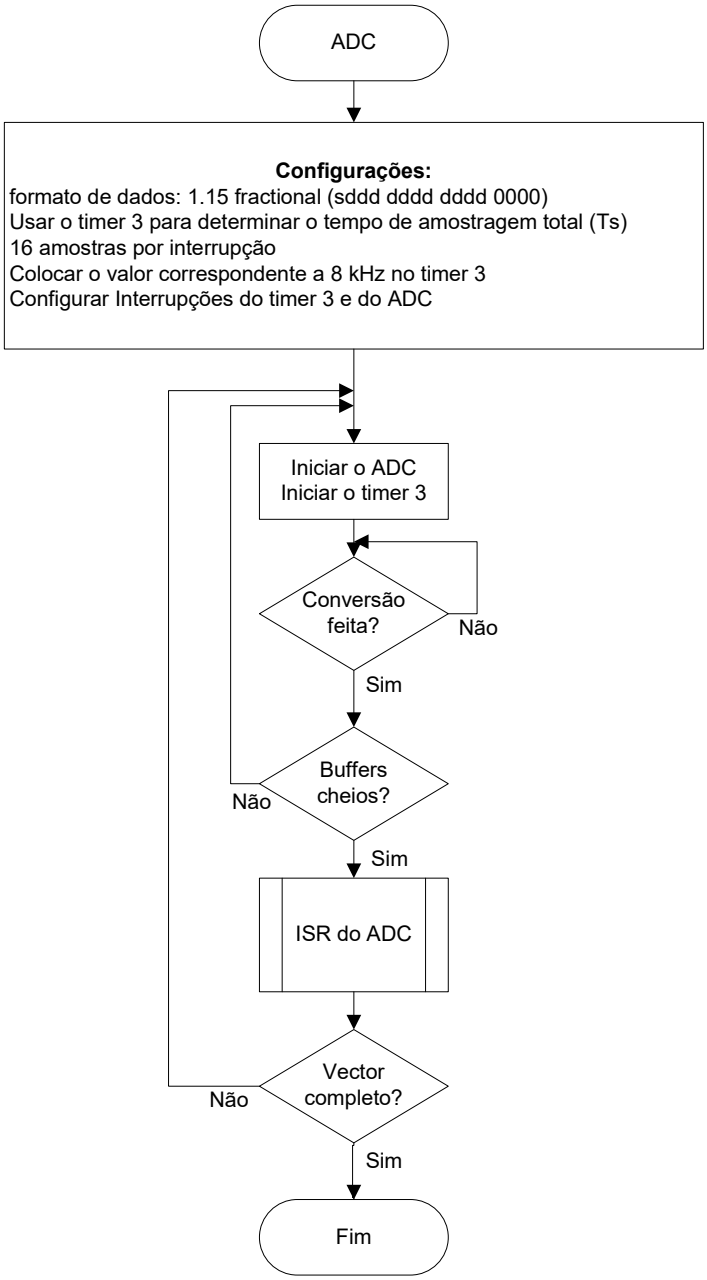


fig 47 – Algoritmo de implementação do ADC.

É na ISR – *Interrupt Service Routine* do ADC, após o acontecimento da interrupção, que o valor dos buffers é guardado para um vector, de modo a ficarem livres para umas novas 16 amostras, de acordo com o seguinte algoritmo:

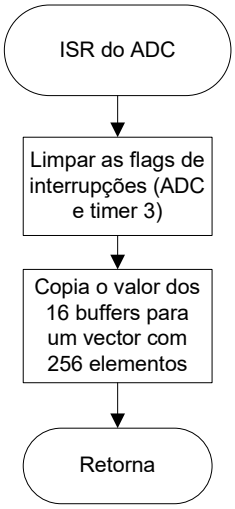


fig 48 – Algoritmo da ISR do ADC.

NOTA: Estes algoritmos foram desenvolvidos para se implementarem em linguagem C no microcontrolador dsPIC30F4013. O código que implementa o ADC está no ficheiro 'ADC.c', ficando a declaração de variáveis e funções externas no ficheiro 'ADC.h'. Todo este código encontra-se em anexo devidamente comentado.

Capítulo 8 – Cálculo do Espectro

"*Espectro sonoro é a distribuição, no domínio das frequências, do conjunto de todas as ondas que formam um som.*" Referência [23]

Como já foi dito, um sinal sonoro é composto pela soma de vários sinais sonoros puros, cada um representado por uma sinusóide. O espectro de um sinal é a representação desse mesmo sinal pelas sinusóides que o compõem, em vez da soma das mesmas. Esta representação é feita através de um gráfico da amplitude em função da frequência, a que se chama espectro do módulo, existindo também o espectro da fase que representa a fase em função da frequência.

Uma representação de um sinal pelas suas sinusóides é mais vantajosa para um projecto do género do reconhecimento de apito, visto que o espectro do sinal tem informação mais clara sobre o conteúdo do mesmo, isto é, olhando para o espectro vê-se directamente a composição do sinal, enquanto que olhando para o próprio sinal, no caso de ele ser relativamente complexo, vê-se uma "sopa" de sinusóides somadas, não se conseguindo ter percepção do seu conteúdo. Isto pode verificar-se na figura seguinte, que mostra um exemplo de um sinal sonoro bastante complexo.

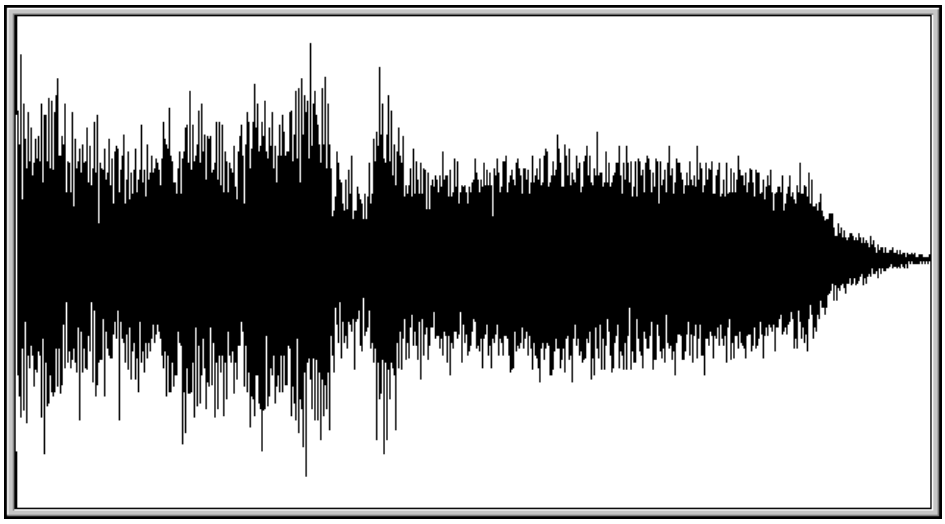


fig 49 – Representação de um sinal sonoro complexo em função do tempo. (tirada de [24])

Lidar com o espectro do sinal em vez do próprio sinal é sem dúvida a escolha correcta para o projecto de reconhecimento de apito, contudo não existe nenhum transdutor que devolva directamente o espectro do sinal, para captar sinais sonoros, usa-

se um microfone, devolvendo este um sinal eléctrico correspondente ao sinal sonoro em função do tempo. Então para se obter o espectro do sinal, tem que em primeiro lugar se obter o próprio sinal, e obter um mecanismo que transforme o sinal temporal em espectro. Esse mecanismo no caso de o sinal temporal ser contínuo pode ser feito através do cálculo de uma Transformada de Fourier. Para sinais discretos ou digitais este processo pode ser feito através de uma DFT – *Discrete Fourier Transform* ou na sua versão optimizada a FFT que já foi referida anteriormente.

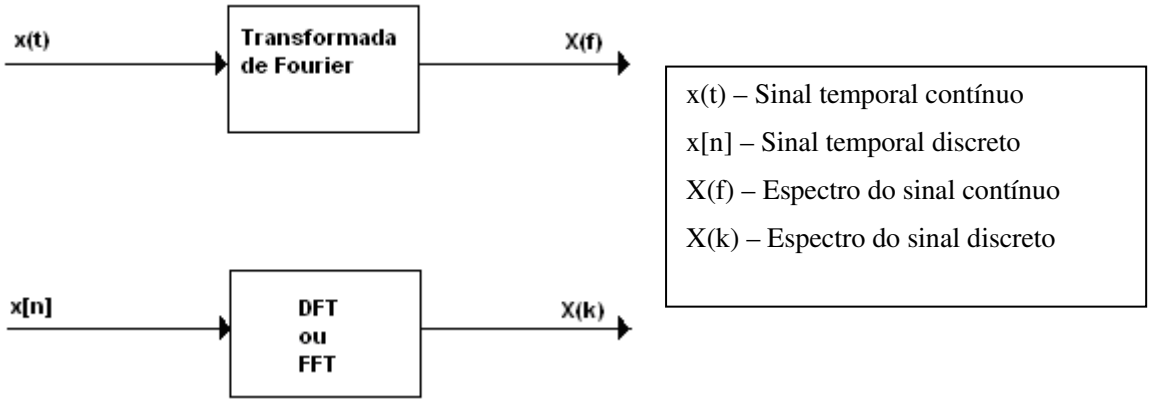


fig 50 – Diagrama de blocos da transformação de sinal temporal em espectro.

Neste projecto, o sinal temporal inicialmente é contínuo, contudo após a conversão analógica-digital este passa a ser discreto, para que possa ser processado no microcontrolador. Então para calcular o espectro do sinal adquirido tem que se utilizar uma transformada para sinais discretos, ou seja, a DFT ou a FFT.

8.1 DFT

A DFT trata-se de um método de cálculo espectral para sinais discretos e periódicos, esta transformada tem como entrada N amostras do sinal temporal (N é um número natural) e devolve na saída N valores correspondentes ao espectro do sinal de entrada. A fórmula para calcular a DFT é a seguinte:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk\frac{2\pi}{N}n}, k = 1,2,3....,N \qquad \textbf{eq (10)}$$

Onde:

X[n] – Sinal temporal

X(k) – Espectro do sinal

N – Número de amostras

Pode ser feito também o processo inverso, ou seja, transformar um espectro de N pontos num sinal temporal de N amostras, este processo é feito pela DFT Inversa, que é dada pela fórmula:

$$x[n] = \sum_{k=0}^{N-1} X(k) e^{jk\frac{2\pi}{N}n}, n = 1,2,3...., N \qquad \textbf{eq (11)}$$

O cálculo da DFT implica que o sinal de entrada seja periódico, estando-se a lidar com sinais reais, nunca se encontram sinais inteiramente periódicos. Então para se fazer o cálculo, tem que se assegurar que o sinal de entrada pode ser considerado periódico durante o tempo em que é capturado. A figura seguinte ilustra o resultado de uma DFT para uma onda quadrada de *duty cycle* 50%, sendo o cálculo feito com 10 amostras, para um período do sinal.

fig 51 – DFT de uma onda quadrada. (tirada de [20] – aula2.pdf)

No caso de o sinal de entrada não poder ser considerado periódico, este método de cálculo espectral não pode ser utilizado, podendo-se recorrer à DTFT – *Discrete Time Fourier Transform,* contudo o método de cálculo desta transformada não é tão simples, é necessária uma expressão matemática do sinal de entrada para que possa ser calculada, e não apenas as amostras do sinal, como no caso da DFT.

8.2 FFT

Para reduzir o tempo de cálculo em sistemas embebidos existe a Transformada Rápida de Fourier, normalmente designada de FFT, trata-se de um algoritmo que faz a DFT com um método que exige menos complexidade computacional que a própria

DFT, tornado assim o seu cálculo mais rápido. Tal como a DFT, o sinal de entrada de uma FFT tem que ser também discreto e periódico, com a limitação acrescida que o número de amostras N tem que ser do tipo $2^n, n = 1,2,3...$

A eficácia deste algoritmo deve-se ao facto de este conseguir reduzir o número de operações com números complexos, dado que estas operações requerem um tempo de cálculo considerável. O princípio de cálculo da FFT é idêntico ao da DFT, já que a FFT é apenas uma forma mais rápida de calcular a DFT quando N é do tipo 2^n. A FFT tira partido de certas simetrias que existem na expressão $e^{-j\frac{2\pi}{N}}$, da qual faz parte do cálculo da DFT.

Considerando $W_N = e^{-j\frac{2\pi}{N}}$, verifica-se que:

$$W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^* \qquad \textbf{eq (12)}$$

$$W_N^{kn} = W_N^{k(n+N)} = (W_N^{n(k+N)}) \qquad \textbf{eq (13)}$$

Então para optimizar o cálculo, divide-se as amostras do sinal de entrada em 2, os que têm n par e os que têm n ímpar, e calcula-se para cada parte a transformada (DFT) de N/2 pontos. De seguida volta a fazer-se nova divisão, e assim sucessivamente até se obter transformadas de apenas 2 pontos. Na figura seguinte ilustra este algoritmo para 8 (2^3) pontos, onde à esquerda estão representados os pontos do sinal temporal x[n] e à direita estão representados os pontos do espectro X(k).

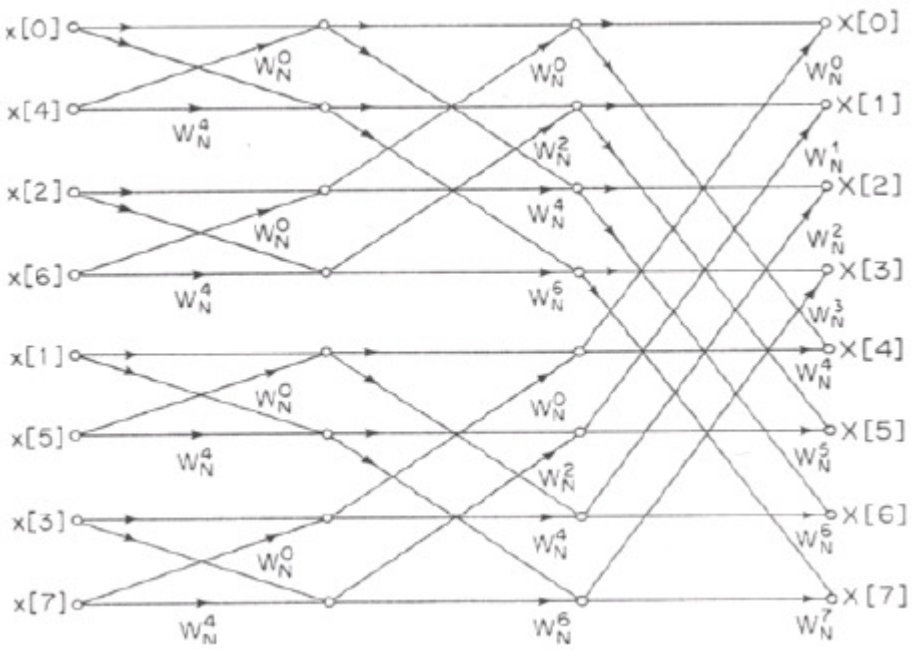


fig 52 – Algoritmo FFT para 8 pontos. (tirada de [20] – 2ªaula.pdf)

Seguindo o esquema do algoritmo, obtém-se cada ponto da FFT de acordo com as seguintes equações:

$$X(0) = x[0] + x[4].W_N^0 + W_N^0(x[2] + x[6].W_N^0) + W_N^0(x[1] + x[5].W_N^0 + W_N^0(x[3] + x[7].W_N^0))$$
$$X(1) = x[0] + x[4].W_N^4 + W_N^2(x[2] + x[6].W_N^4) + W_N^1(x[1] + x[5].W_N^4 + W_N^2(x[3] + x[7].W_N^4))$$
$$X(2) = x[0] + x[4].W_N^0 + W_N^4(x[2] + x[6].W_N^0) + W_N^2(x[1] + x[5].W_N^0 + W_N^4(x[3] + x[7].W_N^0))$$
$$X(3) = x[0] + x[4].W_N^4 + W_N^6(x[2] + x[6].W_N^4) + W_N^3(x[1] + x[5].W_N^4 + W_N^6(x[3] + x[7].W_N^4))$$
$$X(4) = x[0] + x[4].W_N^0 + W_N^0(x[2] + x[6].W_N^0) + W_N^4(x[1] + x[5].W_N^0 + W_N^0(x[3] + x[7].W_N^0))$$
$$X(5) = x[0] + x[4].W_N^4 + W_N^2(x[2] + x[6].W_N^4) + W_N^5(x[1] + x[5].W_N^4 + W_N^2(x[3] + x[7].W_N^4))$$
$$X(6) = x[0] + x[4].W_N^0 + W_N^4(x[2] + x[6].W_N^0) + W_N^6(x[1] + x[5].W_N^0 + W_N^4(x[3] + x[7].W_N^0))$$
$$X(7) = x[0] + x[4].W_N^4 + W_N^6(x[2] + x[6].W_N^4) + W_N^7(x[1] + x[5].W_N^4 + W_N^6(x[3] + x[7].W_N^4))$$

O Algoritmo da FFT consegue reduzir a complexidade para $N\log_2(N)$ operações com números complexos, no caso da DFT esta complexidade é de N^2. Para o caso deste projecto onde $N = 256$, o número de operações complexas é de 2048 no caso da FFT, e 65536 no caso da DFT. Portanto verifica-se que a FFT consegue ter um desempenho significativamente melhor que a DFT, no que diz respeito ao cálculo espectral deste projecto, daí a sua escolha para este projecto.

8.3 Influência da Amostragem no Espectro

Independente do método de cálculo do espectro, a amostragem tem influências no resultado. Como já se verificou, o número de pontos na amostragem corresponde ao número de pontos do espectro até à frequência de amostragem. Então se existirem menos amostras, o espectro vai ter menos resolução. O outro aspecto a ter em conta é a influência da frequência de amostragem, segundo a teoria dos sinais discretos, o espectro de um sinal discreto é reflectido nos múltiplos da frequência de amostragem ($f_s, 2f_s, 3f_s ....$). Considerando um sinal x(t) com a seguinte forma:

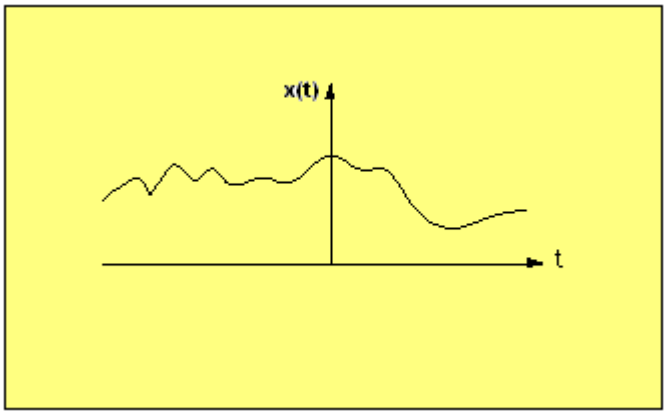


fig 53 – Sinal genérico x(t).

Supondo que este possui um espectro X(f) com a seguinte forma:

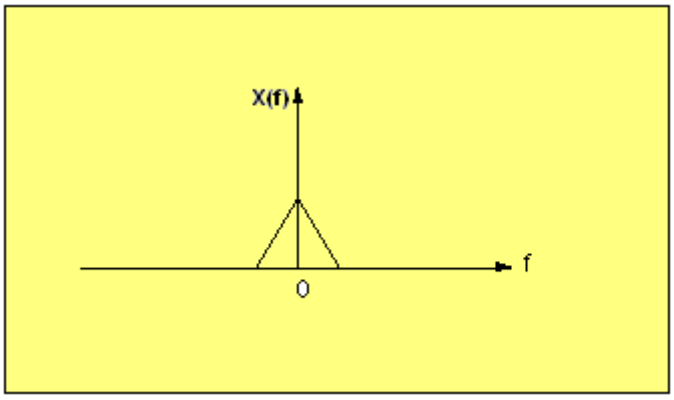


fig 54 – Espectro fictício de x(t).

Quando o sinal x(t) é amostrado com período de amostragem T, passa a ser um sinal x[n], onde $t = nT$, ficando o seu espectro a ser a reflexão de X(f) nos múltiplos da frequência de amostragem. Na figura seguinte pode-se observar o efeito da amostragem no espectro do sinal para uma frequência de amostragem superior à frequência de Nyquist.

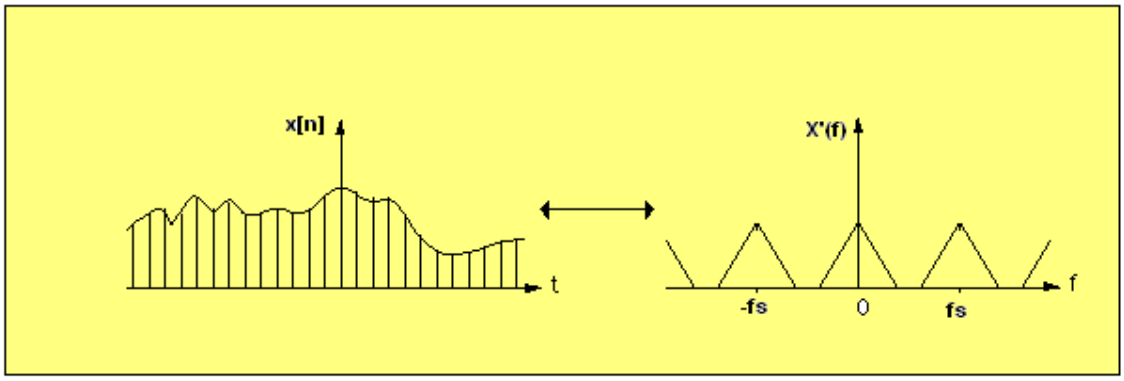


fig 55 – Espectro x[n] amostrado a uma frequência superior à frequência de Nyquist.

Quando a frequência de amostragem é inferior à frequência de Nyquist, ou seja, existe *aliasing*, a reflexão dos espectros cruza-se como mostra a figura seguinte. O cruzamento traduz-se na soma das 2 partes dos espectros.

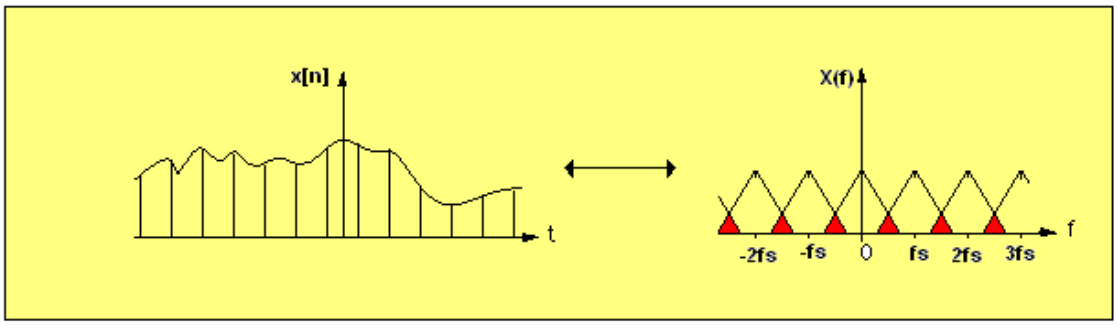


fig 56– Espectro x[n] amostrado a uma frequência superior à frequência de Nyquist.

8.4 Implementação da FFT no Microcontrolador dsPIC30F4013

Sendo os microcontroladores da família dsPIC30F destinados a aplicações de processamento de sinal, o fabricante tem uma biblioteca (dsp.h) com funções e tipos de variáveis próprias para processamento de sinal, que fazem uso do hardware de processamento de sinal que dispõe os DSC's.

As variáveis que são usadas nesta biblioteca são as seguintes:

- *fractional* – variável de 16 bits que varia entre -1 e 1, o primeiro bit é o bit de sinal e os números negativos encontram-se em C_2 (complemento para 2).
- *fractcomplex* – estrutura composta por duas variáveis do tipo *fractional,* para representar um número complexo na forma a+jb.

A biblioteca dsp.h possui várias funções de processamento de sinal, entre as quais estão incluídas funções para fazer directamente uma FFT.

8.4.1 Optimizações

A implementação da FFT no microcontrolador deve ter o máximo de optimização possível, porque apesar de ser um algoritmo menos exigente quando comparada com a DFT, é ainda um algoritmo que exige uma capacidade computacional que pode sobrecarregar o microcontrolador. Então todas as optimizações possíveis devem ser feitas.

Uma das operações de optimização realizada consiste no método de cálculo dos factores $W_N's$. É de notar que estes factores não dependem do sinal de entrada, dependem apenas do número de amostras N, e cada um deles do seu índice k.

$$W_N^k = e^{-j\frac{2\pi}{N}k} (k = 0,1,2...,N-1) \qquad \textbf{eq (14)}$$

Isto faz com que estes factores possam ser considerados constantes quando definido N. E sendo constantes podem ser calculados previamente, não sendo necessário o seu cálculo durante o processo de cálculo da FFT. Então estes factores são então declarados num vector de variáveis *fractcomplex* na memória de código e chamam-se durante o cálculo da FFT.

Outro aspecto a ter em conta no cálculo da FFT é a redundância do espectro. Um espectro de um sinal discreto repete-se nos múltiplos de f_s, e o espectro possui uma simetria em relação ao eixo das ordenadas. Estas características do espectro de sinais discretos, faz com que seja apenas necessário conhecer o espectro entre 0 e $f_s/2$ Hz (0-N/2), tudo o resto é informação redundante. Então o cálculo da FFT que por norma se calcula para N pontos, é apenas necessário ser calculado para N/2 pontos.

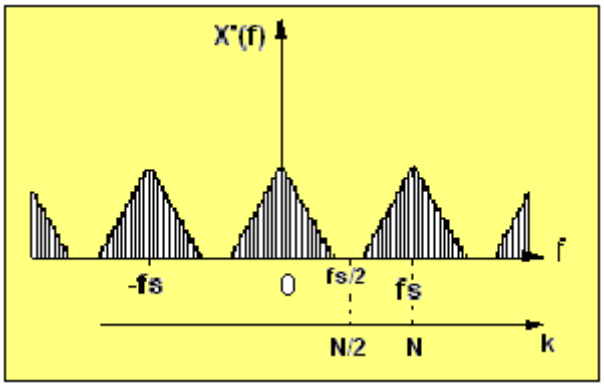


fig 57 – Simetrias do espectro de um sinal discreto.

8.4.2 Função FFTComplexIP

Na biblioteca dsp.h disponibilizada pela Microchip, o fabricante do microcontrolador utilizado, encontra-se uma função designada FFTComplexIP. Trata-se de uma função que faz os cálculos da FFT. Tendo porém que se ajustar os parâmetros que esta aceita e devolve. É esta a função utilizada neste projecto para fazer o cálculo da FFT, as suas características são as seguintes:

- Recebe um vector de números complexos (*fractcomplex)* com N elementos;
- Devolve o resultado também na forma de números complexos, guardando o resultado sobre o vector de entrada nas N/2 primeiras posições do mesmo;
- Permite chamar os factores W_N da memória de código ou da memória de dados.

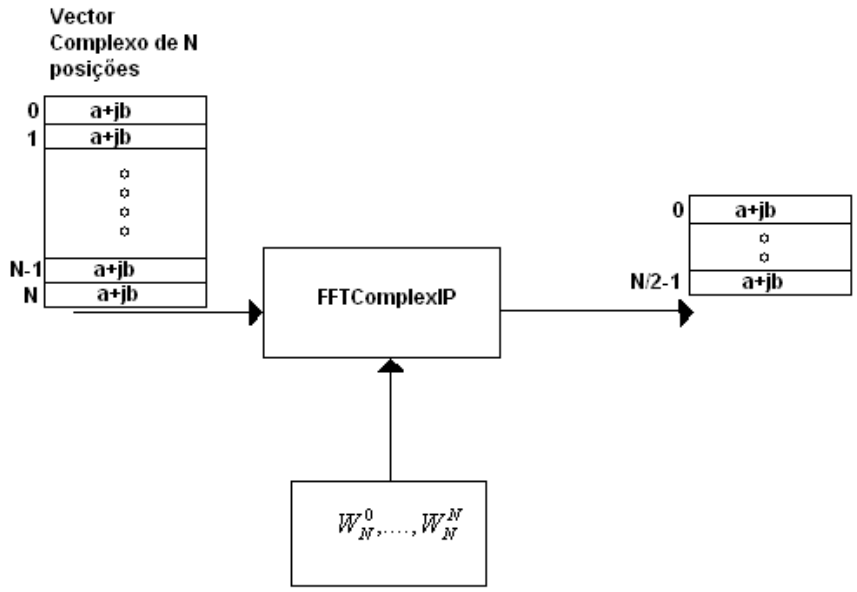


fig 58 – Funcionamento da função FFTComplexIP.

Esta função embora faça uma FFT conforme é pretendido para o projecto, difere em alguns parâmetros comparando-a com a FFT pretendida. Já que esta recebe e devolve números complexos, o que não deve acontecer na sua implementação neste projecto, ou seja, a FFT deste projecto recebe um vector de números reais, que se trata do sinal x[n] da saída do ADC. Quanto à saída a função FFTComplexIP não está em acordo com os objectivos para este projecto, já que devolve números complexos que representam o espectro da fase e módulo (amplitude) em conjunto, neste projecto é pretendido apenas o módulo separadamente. Posto isto verifica-se que é necessário fazer uma ligação entre a função e o restante projecto, que consiste em:

- Transformar o vector de saída do ADC num vector de números complexos;
- Calcular o módulo de cada número complexo devolvido pela função FFTComplexIP.

8.4.3 Transformação de um Vector de Números reais num Vector de Números Complexos

O vector de saída do ADC que representa o sinal temporal discreto, é composto por variáveis do tipo *fractional*, por sua vez um vector complexo é composto por variáveis do tipo *fractcomplex*, que são formadas por 2 variáveis do tipo *fractional*, uma

para a parte real e outra para a parte imaginária. Para se obter o vector complexo, iguala-se a parte real de cada elemento a cada valor do sinal de saída do ADC e iguala-se a parte imaginária a zero, como ilustra a figura seguinte.

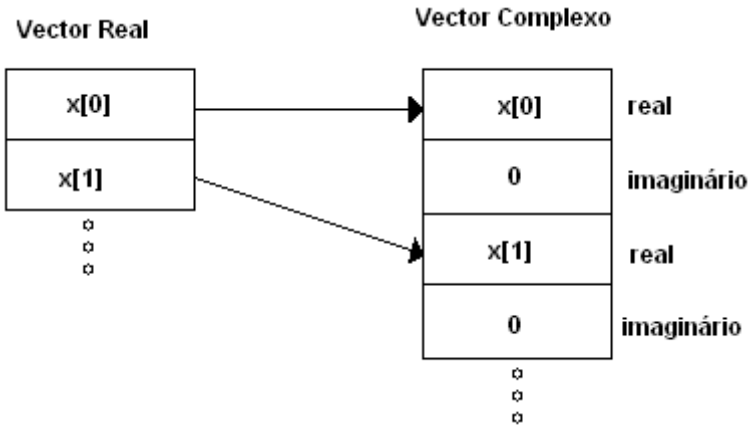


fig 59 – Transformação de um vector de números reais num vector de números complexos.

8.4.4 Cálculo do módulo do vector de números complexos

Um número complexo $a + jb$ tem um módulo dado pela expressão:

$$z = a + jb \Rightarrow |z| = \sqrt{a^2 + b^2} \qquad \textbf{eq (15)}$$

Então a implementação desta expressão no microcontrolador após o cálculo da FFT com a função referida, obtinha o módulo dos números complexos representativos do espectro. Contudo esta expressão tem um peso computacional muito elevado, sendo mesmo demasiado complexa para se implementar em conjunto com a FFT neste microprocessador, já que requer demasiada memória. Então a opção para calcular o módulo dos números complexos passa por fazer uma aproximação à expressão da eq. 15, aproximação essa que é dada pela expressão:

$$|z| \approx |a| + |b| \qquad \textbf{eq (16)}$$

Matematicamente não se pode concluir que as 2 expressões sejam semelhantes, contudo com a variação de *a* ou de *b*, as duas expressões têm a mesma tendência, ou seja, aumentar ou diminuir consoante *a* ou *b* aumentem ou diminuam também. Utilizando a eq. 16 em vez da eq.15 comete-se um erro de cálculo no que diz respeito ao

valor das amplitudes no espectro, contudo esse erro não tem qualquer influência para este projecto porque não altera a forma espectral.

Para implementar a eq. 16 é apenas necessário calcular o módulo de cada parte e soma-los. O método mais eficaz de calcular o módulo de um número real é passar os números negativos, que se representam em complemento para 2 (C_2) para complemento para 0 (C_0). Sendo então esse o método utilizado neste projecto para calcular o módulo de um número real, de acordo com o seguinte algoritmo.

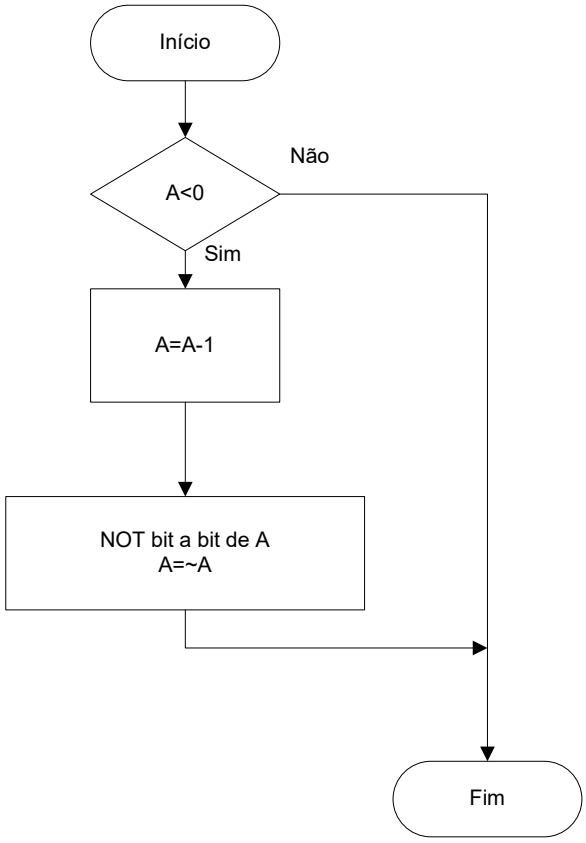


fig 60 – Algoritmo de cálculo do módulo de um número real.

8.5 Algoritmo Geral da FFT

Resumidamente o cálculo da FFT usa a função FFTComplexIP, adaptando a entrada e calculando o módulo da saída, os factores W_N declaram-se na memória de código como constantes. Então o algoritmo geral do cálculo consiste em transformar o vector de saída do ADC num vector complexo, de seguida calcular o espectro e no fim

calcular o módulo dos valores devolvidos para obter o espectro do módulo. Este algoritmo está representado na figura seguinte na forma de fluxograma.

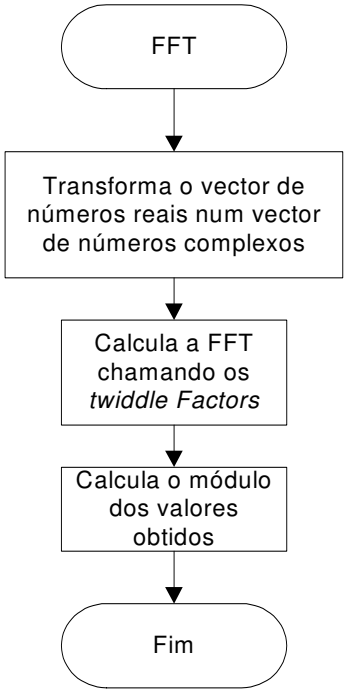


fig 61 – Algoritmo de cálculo da FFT.

NOTA: Os algoritmos da implementação da FFT foram desenvolvidos para se implementarem em linguagem C no microcontrolador dsPIC30F4013. O código que a implementa está no ficheiro 'fft.c', os factores W_N estão declarados no ficheiro 'twiddleFactors.c', ficando a declaração de variáveis e funções externas no ficheiro 'fft.h'. Todo este código encontra-se em anexo devidamente comentado.

Capítulo 9 – Percepção e Função *Main*

9.1 Função Main

A função *main* de um software é a função principal do mesmo, o código que esta contém é o código que irá correr quando o sofware começar a funcionar, sendo que o código das outras funções só começa a correr quando for chamado na função *main*. Então esta função gere todo o processo implementado no software, e deve incluir ciclos que garantam a repetição de todas as iterações conforme seja pretendido.

Neste projecto a função main inclui a implementação do algoritmo de percepção, chama as restantes funções (ADC e FFT), limpa variáveis que identificam o estado de determinada operação (flags) e implementa um ciclo infinito que garante que se esteja a verificar continuamente se existe ou não um sinal de apito.

9.1.1 Funcionamento Contínuo do Software

O software neste projecto faz a implementação da aquisição de sinal (ADC), cálculo espectral (FFT), e do algoritmo de percepção. Contudo o sistema deve estar constantemente "à escuta" de apito, logo tem que repetir continuamente essas funções, chamando-se ao conjunto dessas 3 funções (ADC + FFT + percepção) uma iteração. A figura seguinte representa o funcionamento do sistema fazendo várias iterações ao longo do tempo.

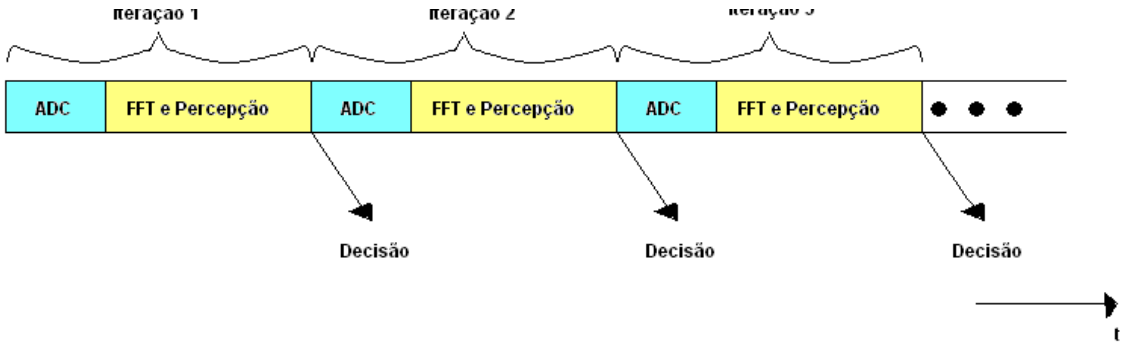


fig 62 – Funcionamento das iterações no projecto.

O tempo de execução de uma iteração deve ser bastante inferior ao tempo que dura o apito, para que se possa fazer várias iterações durante o tempo de duração do apito, e consequentemente ter um sistema mais fiável e mais flexível. Seguindo as cores dos processos da figura anterior, representa-se na figura seguinte o comportamento do sistema quando ocorre o som de um apito.

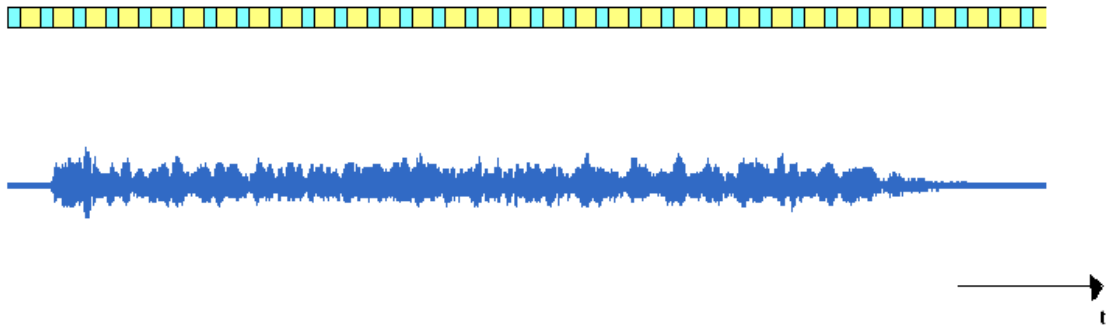


fig 63 – Funcionamento das iterações quando ocorre o som de um apito.

9.1.2 Algoritmo da *Main*

A função *main* deste projecto, primeiramente chama a função que faz o ADC, quando a *flag* de activação da FFT for activada, quer dizer que a aquisição está terminada, então chama a função que faz a FFT, após o cálculo da FFT executa a percepção, cujo código está na própria função *main*, como resultado desta última tarefa deve devolver uma decisão (apito ou não apito), esta decisão faz-se activando uma saída do microcontrolador que está ligada a uma saída da placa que compõe o sistema, assim como a um LED – *Light Emitting Diode* para se poder observar visivelmente o resultado da decisão daquela iteração.

O algoritmo da função *main* está ilustrado na figura seguinte sem pormenorizar o algoritmo de percepção, este está descrito mais à frente neste capítulo.

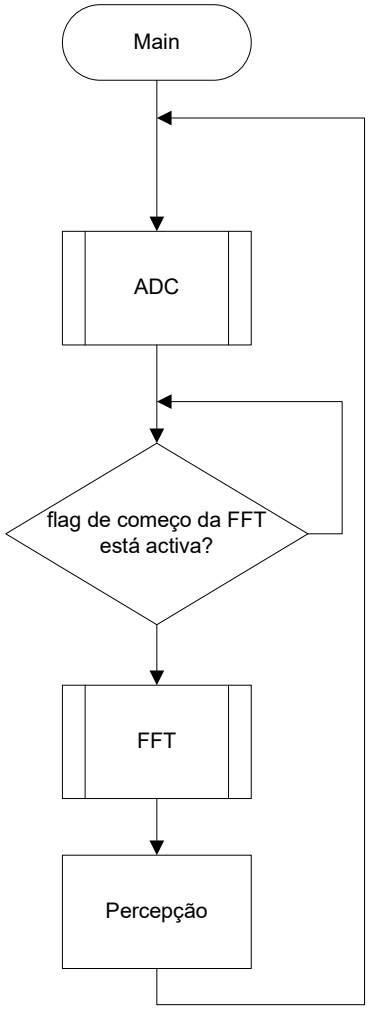


fig 64 – Algoritmo de implementação da *main*.

9.2 Percepção

Após calculada a FFT, segue-se a tomada de decisão, isto é, verificar se o espectro se trata de sinal de apito ou sinal sem som de apito. Para esta tarefa é necessário criar e implementar um algoritmo que tenha percepção do espectro calculado com a FFT, e devolva uma decisão.

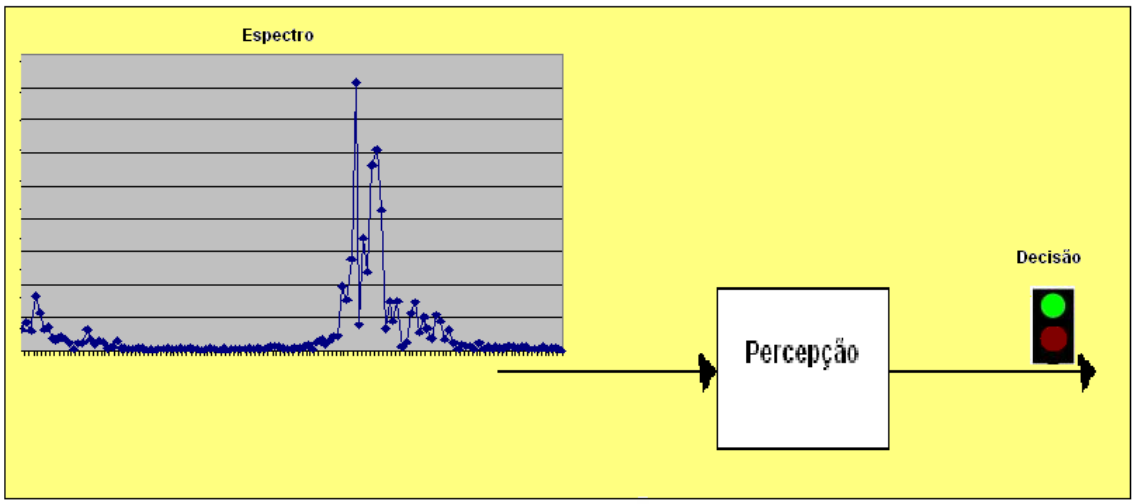


fig 65 – Diagrama de blocos indicando a função da percepção.

9.2.1 Funcionamento da percepção

Para se poder avaliar se o espectro se trata ou não de um sinal de apito, é necessário procurar características do mesmo que o possam distinguir de outros sinais. É de notar que o espectro de um sinal de apito tem as frequências de maior amplitude a rondar os 2.1 a 2.7 kHz aproximadamente. Esta é a característica que mais identifica um espectro de sinal de som de apito. Como se pode verificar no capítulo 4, a maior parte dos ruídos possui amplitudes baixas nessa gama (exceptuando possivelmente o assobio de dedos e palmas). Então se existir uma frequência de pico na gama de frequências obtida, e se a essa frequência corresponder uma amplitude relativamente alta pode dizer-se com algum grau de certeza que se trata de um som de apito. Este facto serve de base para a elaboração do algoritmo de percepção.

A percepção faz uma espécie de máscara ao espectro que recebe, criando uma zona de aceitação e uma zona de rejeição, em frequência e amplitude, e verifica se existe uma frequência na zona de aceitação, se existir então o sinal pode ser de apito, se não existir, é certo que o sinal não é de apito. Na figura seguinte representa-se esse funcionamento para um caso onde o espectro é de sinal de apito, na máscara a zona a preto representa a zona de rejeição enquanto que a zona a branco representa a zona de aceitação.

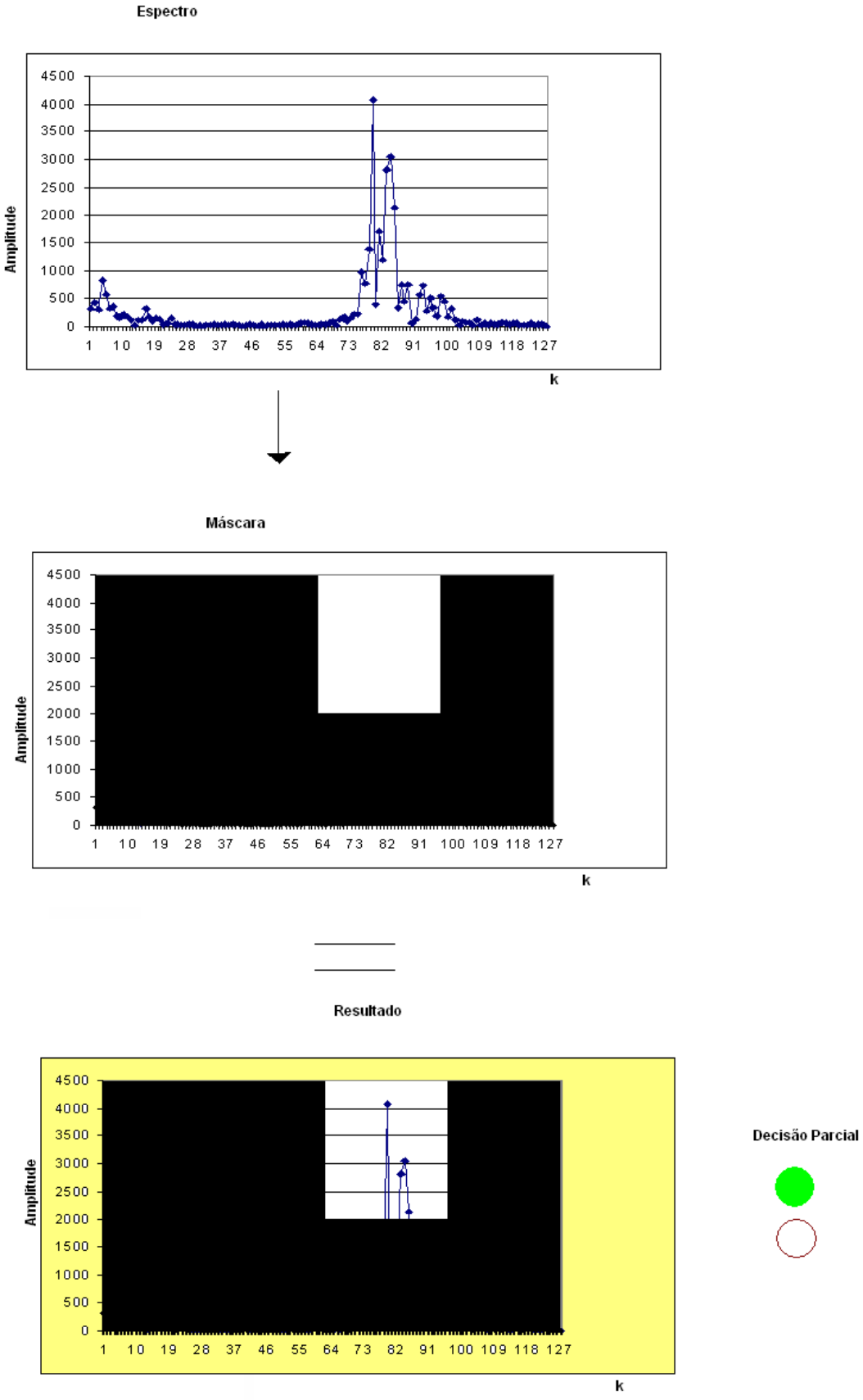


fig 66 – Funcionamento do algoritmo de percepção.

Esta decisão parcial resulta de uma iteração, contudo para dar mais robustez ao algoritmo de percepção considera-se que a decisão final de considerar o sinal, um sinal de apito, é só tomada após existirem R decisões parciais de igual resultado, ou seja, após R iterações a ser considerado apito, considera-se de facto apito. E as hipóteses de haver ruído nessa gama da máscara tornam-se ainda mais reduzidas.

9.2.2 Parâmetros

Um dos objectivos para o projecto de reconhecimento de apito é a capacidade de distinção entre 2 apitos diferentes, pode-se verificar que de facto 2 apitos diferentes podem ter frequências de pico diferentes, contudo a frequência de pico depende também de outros factores, como a pessoa que sopra ou a intensidade de sopro. Se fosse utilizada uma gama de frequências elevada como 2.1 a 2.7 kHz, o sistema detectaria todos os apitos em todas as condições, por outro lado tornar-se-ia mais sensível a ruídos. Então há o objectivo de distinguir 2 apitos com a máxima robustez ao ruído, implicando também a necessidade de poder ajustar essa gama de frequências para reconhecer determinado apito, ou seja ajustar a máscara. Tal como o valor de amplitude que deve ser ajustado para reconhecer à distancia máxima que o apito pode estar do sistema de reconhecimento. Estes factores obrigam a que o sistema tenha que ser dotado de parâmetros ajustáveis, para se poder adaptar a máscara, para tal o sistema utilizaram-se 3 parâmetros de ajuste da máscara, a frequência mínima (f1) a frequência máxima (f2) e a amplitude mínima (A). A influência destes parâmetros na máscara é representada na figura seguinte.

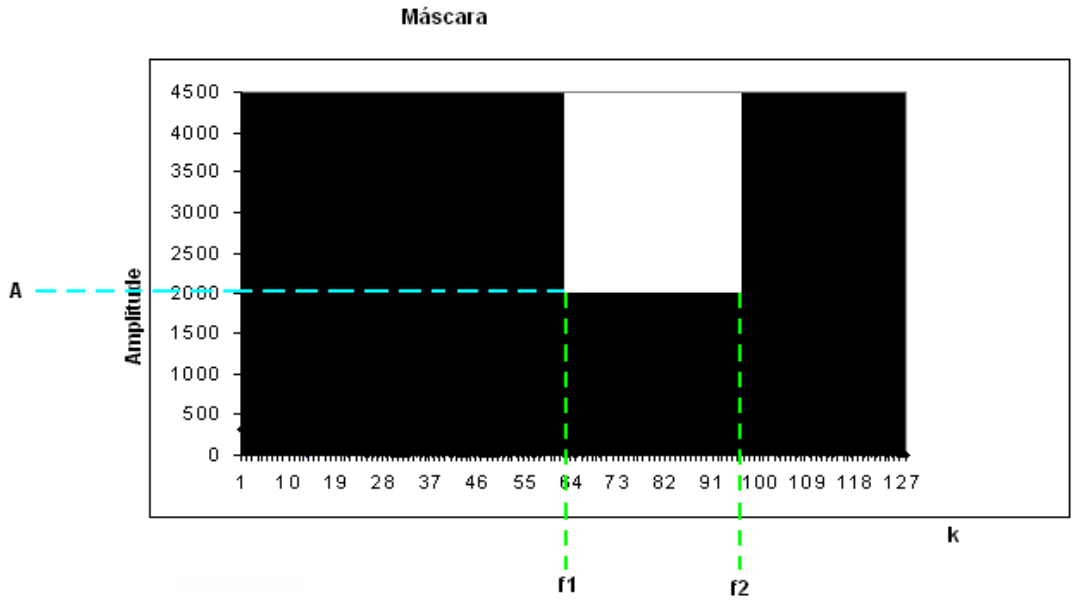


fig 67 – Influência dos parâmetros na máscara.

A introdução destes parâmetros no sistema de reconhecimento de apito, permite também um melhor controlo da robustez, já que permite alterar o tamanho da zona de aceitação. Isto é, com a diminuição da zona de aceitação, o sistema fica mais imune ao ruído, contudo pode haver iterações onde o sinal é de apito, mas não seja considerado. Deve-se usufruir desta particularidade do sistema para o adaptar ao nível e tipo de ruído, para ambientes onde exista muito ruído com frequências na gama do apito deve-se diminuir a zona de aceitação em amplitude e/ou gama de frequências.

O número de iterações consecutivas com decisões parciais positivas para que de facto se aceite a decisão final (R), é também um parâmetro ajustável do sistema. Este parâmetro serve apenas para ajustar o sistema ao nível de ruído, e não para adaptar ao apito. O parâmetro R deve ser usado para filtrar ruídos que podem ultrapassar a máscara tendo frequências com forte amplitude nessa gama, mas que sejam instáveis, como o caso do assobio de dedos e das palmas. Deve ser aumentado quanto mais ruidoso for o ambiente (ruídos influentes), contudo deve andar na ordem das unidades ou poucas dezenas, correndo o risco de ultrapassar o número de iterações que um sopro de apito demora (este valor ronda as dezenas de iterações). Outro inconveniente do aumento de R, é o tempo de resposta do sistema, já que para dar uma resposta definitiva necessita de fazer as R iterações, demorando R vezes o tempo de uma iteração.

O capítulo seguinte aborda métodos de ajuste destes parâmetros.

NOTA: Os 4 parâmetros ajustáveis podem ser modificados com a alteração do ficheiro que compõe o projecto, chamado 'parametros.h', este ficheiro contém apenas a definição dos parâmetros, sendo incluído no código do ficheiro que contem a *main* ('main.c').

9.2.3 Algoritmo da percepção

O Algoritmo de percepção inicialmente, verifica se existe algum ponto do espectro dentro da máscara, onde toma uma decisão parcial. Após essa tomada de decisão, existem 3 estados possíveis:

- Decisão negativa. **Não é apito**;
- Decisão parcial positiva e decisão final negativa. **Ainda não é apito**;
- Decisão parcial positiva e decisão final positiva. **É apito**.

O algoritmo de percepção pode ser representado pelo seguinte fluxograma:

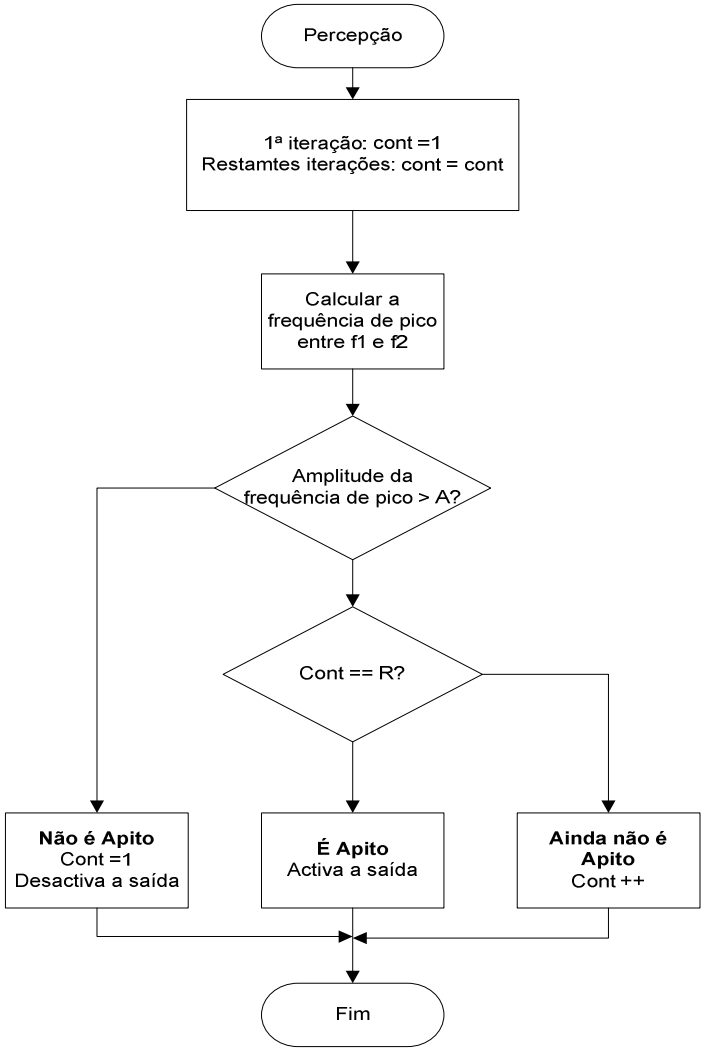


fig 68 – Algoritmo de percepção.

NOTA: A implementação dos algoritmos ilustrados neste capítulo (*Main* e percepção) é feita em linguagem C, no ficheiro 'main.c' do projecto. O código da implementação encontra-se em anexo.

Capítulo 10 – Ajuste dos Parâmetros

O sistema de reconhecimento de apito foi desenvolvido tendo em conta a necessidade de ser ajustado ao apito (instrumento) e ao ambiente envolvente. Para que o sistema tenha a possibilidade de ser ajustado foram criados 4 parâmetros, que já foram citados no capítulo anterior, esses parâmetros são os seguintes:

- **A** – Amplitude mínima da zona de aceitação;
- **f1** – Frequência mínima da zona de aceitação;
- **f2** – Frequência máxima da zona de aceitação;
- **R** – Número de iterações com resposta parcial positiva necessárias para uma resposta definitiva também positiva.

Para que um utilizador deste sistema consiga ajustar os parâmetros, ele necessita ter percepção da sua influência no sistema, assim como da influência do ruído ambiente e da variação do espectro com apitos diferentes. Então foram criadas juntamente com este sistema ferramentas que conseguem dar informação necessária ao ajuste dos 3 primeiros parâmetros (A, f1, f2), o parâmetro R pode ser ajustado sem qualquer ferramenta de ajuda, já que se consegue ter uma boa percepção da sua influência quando é ajustado no próprio sistema de reconhecimento de apito.

As ferramentas de apoio ao ajuste de parâmetros que foram criadas são as seguintes:

- **Simulador de parâmetros –** Ficheiro Excel que gera um gráfico aproximado ao espectro do apito que vai reconhecer conforme o utilizador altera os parâmetros.
- **Software de interface com a porta série –** Software para o microcontrolador que envia os valores de determinado espectro, sendo esses valores recebidos num computador.

10.1 Simulador de Parâmetros

O simulador de parâmetros trata-se de uma ferramenta de apoio ao ajuste dos mesmos parâmetros, onde o utilizador pode alterar o valor dos parâmetros e, através da criação de um gráfico, verificar as consequências no espectro de apito que o sistema passará a aceitar quando esses valores dos parâmetros forem introduzidos. Neste ficheiro o utilizador pode alterar livremente os parâmetros ganhando mais percepção da sua influência no sistema de reconhecimento de apito, devendo no entanto ter noção do que os parâmetros representam, dando o exemplo dos parâmetros f1 e f2, o utilizador deve ter percepção de que f1 representa a frequência mínima da zona de aceitação e f2 representa a frequência máxima da mesma zona de aceitação, logo f2 tem que ser obrigatoriamente superior a f1.

10.1.1 Função Aproximada do Espectro do Apito

O Simulador representa por um gráfico a aproximação do espectro do apito consequente à alteração dos parâmetros, para tal é necessário aproximar o espectro do apito por uma função matemática. Como já se pode verificar em capítulos anteriores, vendo o espectro do som de apito de um modo simplificado, este tem uma frequência de pico que varia conforme alguns factores e as frequências inferiores e superiores têm uma amplitude que vai diminuindo quanto mais longe da frequência de pico estiver. Na figura seguinte representa-se o espectro do som do apito a vermelho, e uma simplificação do mesmo a verde.

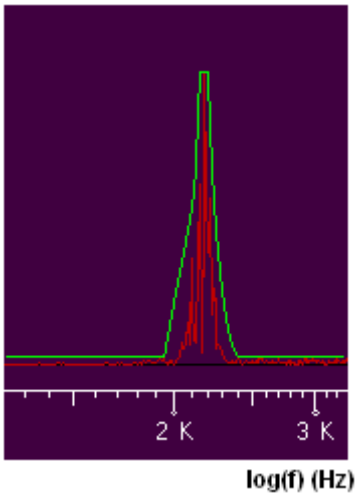


fig 69 – Simplificação do espectro do som de apito.

Feita esta simplificação, a função matemática utilizada para aproximar ao espectro é a seguinte:

$$f(x) = \begin{cases} 1, & x = f_{pico} \\ \left| \dfrac{1}{x - f_{fico}} \right|, & c.c. \end{cases}$$

eq (17)

A função de aproximação tem a seguinte representação gráfica:

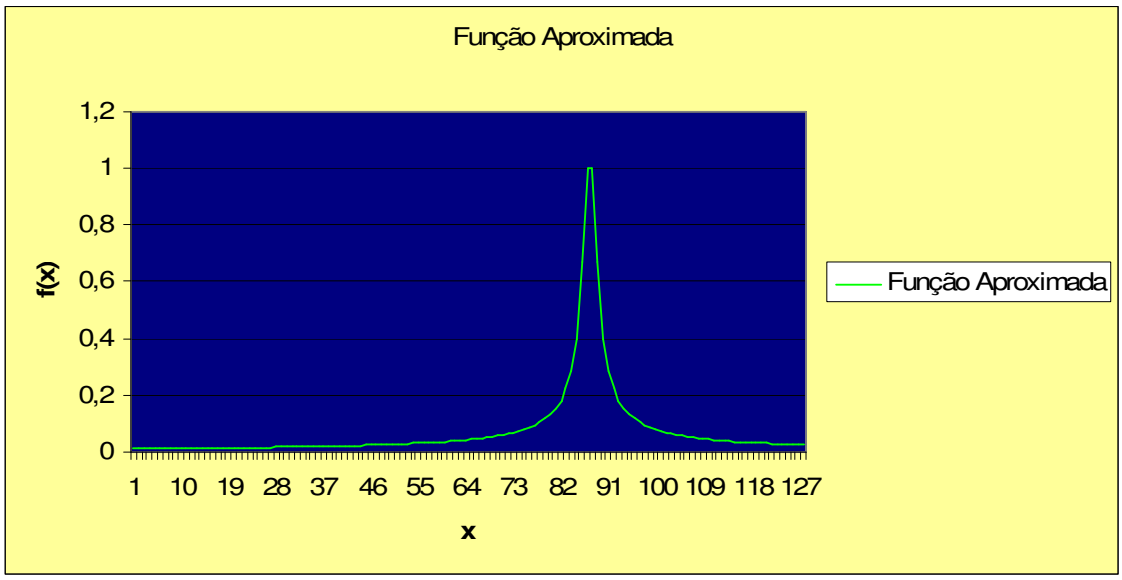


fig 70 – Gráfico da função aproximada ao espectro do som de apito.

Note-se a semelhança do gráfico da função aproximada com o rascunho do espectro simplificado da figura 69, então usa-se a eq (17) como função de aproximação ao espectro. Contudo a eq 17 não depende dos parâmetros ajustáveis, como é objectivo deste simulador, então para que a função de aproximação passe a depender dos parâmetros é necessário que f_{pico} dependa de f1 e f2, como o espectro do apito é aproximadamente simétrico, opta-se por calcular a frequência de pico pela média das frequências mínima e máxima, de acordo com a seguinte fórmula:

$$f_{pico} = \frac{f1 + f2}{2}$$

eq (18)

Para se fazer a simulação a uma escala semelhante à escala que o sistema utiliza, opta-se por multiplicar a função por a amplitude máxima que o espectro pode tomar no sistema de reconhecimento de apito, essa amplitude é de 2^{13} (8192). A escala temporal passa a ser discreta, de acordo com a FFT que também devolve um resultado discreto, ficando o eixo dos xx a representar o índice de frequência k, e não a própria frequência. Deste modo a expressão serve de referência para a função de aproximação é a seguinte:

$$f[k] = \begin{cases} 8192, & k = f_{pico} \\ 8192 \left| \dfrac{1}{k - \dfrac{f2+f1}{2}} \right|, & cc \end{cases}$$

eq (19)

10.1.2 Sobreposição da Máscara

Após achada a função que aproxima o espectro de apito, é necessário sobrepor-lhe a máscara para definir na função a zona de aceitação e a zona de rejeição na função. A máscara depende dos 3 parâmetros (A, f1 e f2), como já foi explicado no capítulo anterior, esta define a zona na qual terá que existir espectro de apito para que a resposta possa ser positiva. Neste simulador representa-se a função quando já está sobreposta a máscara, ficando a zona de aceitação representada a verde e a zona de rejeição representada a vermelho. Na figura seguinte mostra-se uma imagem do simulador os valores de f1 e f2 igual a 85 e 90 respectivamente, e o valor de A igual a 2000.

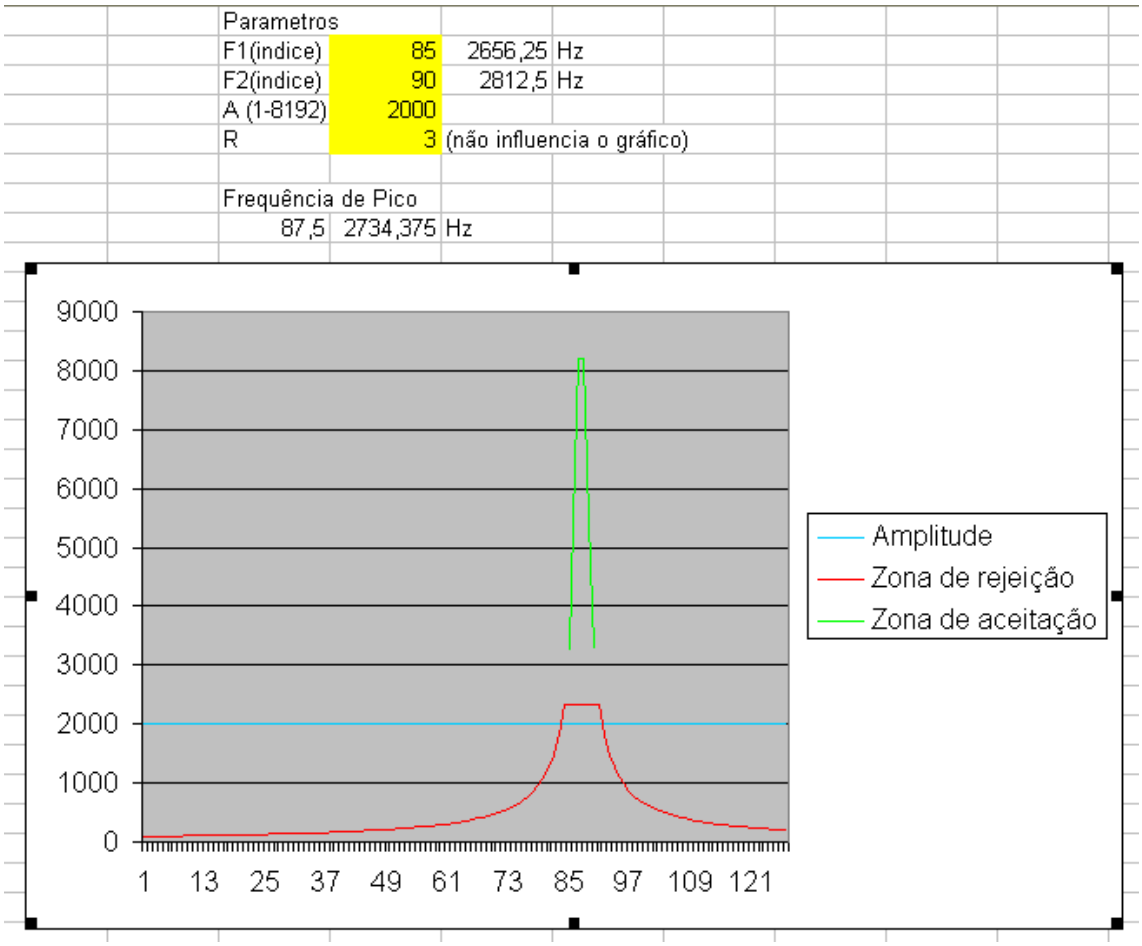


fig 71 – Imagem do simulador de parâmetros.

10.1.3 Alteração dos Parâmetros

O utilizador deste simulador deve apenas alterar os valores dos parâmetros, que correspondem às células de fundo amarelo, dado que as outras células são calculadas através do valor dos parâmetros de forma adequada. Os valores de f1 e f2 correspondem ao valor do índice de frequência k, tal como no sistema de reconhecimento de apito, sendo neste simulador calculado o respectivo valor em Hz na célula em frente de acordo com a seguinte fórmula:

$$f(Hz) = \frac{f_s(Hz) \times k}{N} \qquad \textbf{eq (20)}$$

Onde:

$f(Hz)$ - Frequência em Hz

$f_s(Hz)$ - Frequência de amostragem em Hz

k - Índice de frequência

N – Número de amostras do sinal temporal

10.2 Software de Interface com a Porta Série

O ajuste dos parâmetros de reconhecimento, depende do espectro do apito que se pretende reconhecer com o som de ruído ambiente predominante, isto implica que o utilizador do sistema de reconhecimento de apito tenha que conhecer este espectro, para poder ajustar os parâmetros da qual depende a máscara de reconhecimento. Existem várias ferramentas para calcular o espectro, como o software FFT MusEV que foi utilizado nas análises do capítulo 4, qualquer uma dessas ferramentas poderia ser utilizada para o utilizador do sistema de reconhecimento de apito, contudo essas ferramentas não calculam um espectro exactamente como o sistema de reconhecimento de apito, visto que existem parâmetros diferentes (número de amostras N, frequência de amostragem f_s, amplitude e escalas). Para uma maior facilidade de ajuste dos parâmetros do sistema da parte do utilizador, o espectro do apito e ruído ambiente, na qual ele deverá analisar anteriormente, deve ser do mesmo tipo do calculado no reconhecimento de apito. A solução então passa por ser o próprio sistema a calcular esse espectro, e de certa forma envia-lo para o utilizador.

O software de interface com a porta série foi criado no âmbito da tarefa de envio do espectro para o utilizador, ou seja, quando o utilizador quiser visualizar um espectro do apito e/ou do ruído ambiente ele utiliza este software, para que o sistema envie o espectro para um computador, onde pode fazer um gráfico do espectro e analisá-lo.

Quando o sistema está a correr com este software o seu funcionamento é parecido com o seu funcionamento durante o reconhecimento, sendo apenas alterado o software de percepção pelo software de envio pela porta série. Deste modo, o diagrama de blocos do software do sistema quando está carregado para enviar o espectro é o representado na figura seguinte.

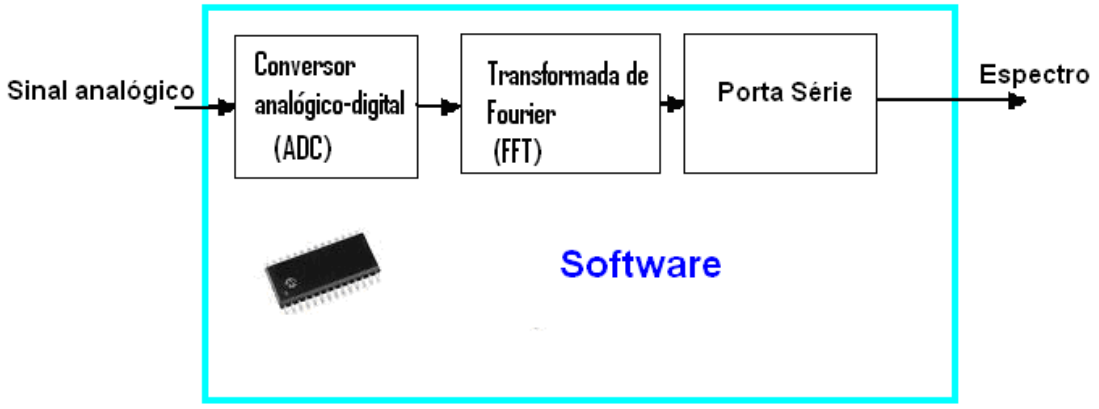


fig 72 – Diagrama de blocos do software quando envia o espectro pela porta série.

10.3 Implementação do Software da Porta Série no microcontrolador

10.3.1 Protocolo

O microcontrolador dsPIC30F4013 possui interface com a porta série através do módulo UART – *Universal Asynchronous Receiver Transmitter*, dada a sua compatibilidade com o protocolo RS-232, usado na porta série dos computadores, é usado para enviar o espectro para o computador.

No protocolo RS-232 a transmissão é assíncrona, e os dados podem ser enviados em pacotes de 8 ou 9 bits, para indicar o início da transmissão envia-se um bit de início (*start bit*), e para indicar o fim normalmente utiliza-se 1 bit de paragem (*stop bit)*, pode-se utilizar opcionalmente um bit de paridade para detecção de erros de transmissão. Os bits são enviados sequencialmente do bit menos significativo (LSB) para o bit mais significativo (MSB).

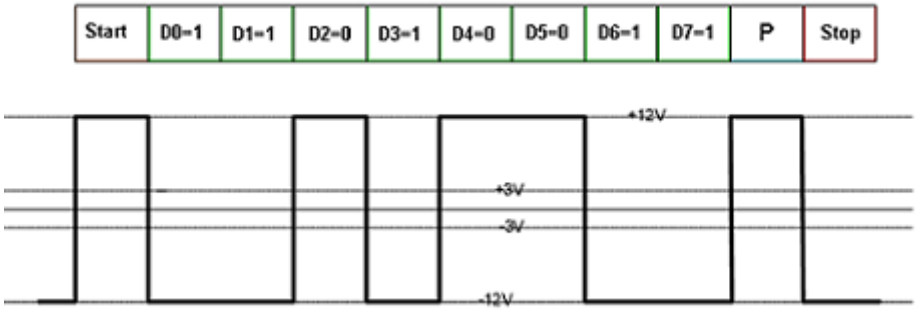


fig 73 – Funcionamento do protocolo RS-232. (tirada de [25])

Os níveis de tensão do módulo UART do microcontrolador e do protocolo RS-232 são diferentes. No protocolo RS-232 o nível lógico ‘0’ pode variar entre 3 e 12V, enquanto que o nível lógico ‘1’ pode variar entre -3 e -12V. O módulo UART coloca na saída níveis de tensão TTL – *Transistor Transistor Logic*, ou seja, 0V corresponde ao nível lógico ‘0’ e 5V corresponde ao nível lógico ‘1’. Isto leva a que seja necessário hardware adicional para transformar os níveis de tensão TTL em níveis de tensão RS-232.

Neste projecto é usado o circuito integrado MAX 233 para fazer a interface entre níveis de tensão TTL e RS-232, o seu *datasheet* encontra-se na referência [26]. O MAX 233 usa-se com a configuração de ligações ilustrada na figura seguinte.

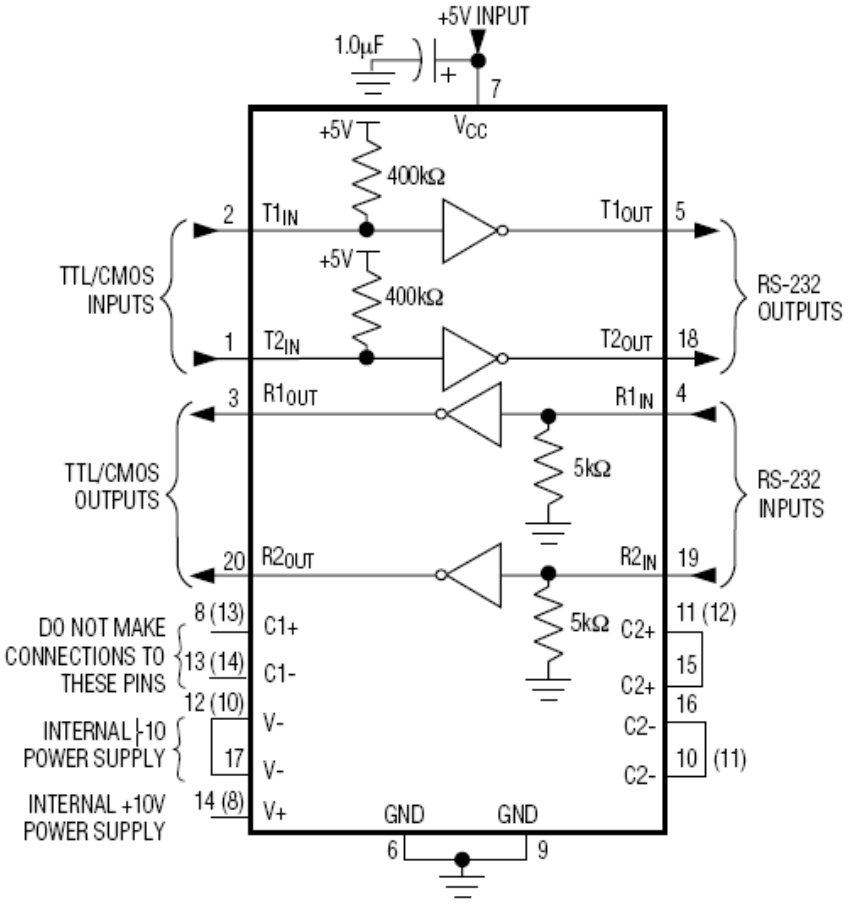


fig 74 – Configuração das ligações do circuito integrado MAX 233. (tirada de [26])

10.3.2 Configurações

O protocolo neste projecto é usado na sua configuração mais simples, ou seja, 8 bits de dados, 1 bit de paragem e sem bit de paridade. Optou-se também por utilizar um *baud rate* de 9600 bps, com esta velocidade praticamente não existem erros de envio, sendo uma velocidade suficiente para este tipo de interface. Com esta configuração os dados são enviados de acordo com a trama da figura seguinte.

fig 75 – Trama de dados usada na porta série.

Para se definir estas configurações no microcontrolador pode-se configurar os registos adequados de acordo com o manual dos microcontroladores da família dsPIC30F, referência [18], ou de outro modo, utilizando funções da biblioteca '*uart.h*'

onde se pode escolher as configurações da porta série sem alterar directamente os registos.

A configuração do *baud rate* é feita através da configuração do registo U1BRG, onde se deve colocar um valor de acordo com a seguinte fórmula:

$$U1BRG = \frac{F_{CY}}{16 \times baude\_rate} - 1 \qquad \textbf{eq (21)}$$

Onde:

F_{CY} – Frequência de relógio de ciclo de instrução (Hz) (neste projecto F_{CY} = 29.48 MHz).

10.3.3 Algoritmo

O software que envia o espectro através da porta série deve transformar o vector que compõe o espectro em tramas de 8 bits de dados para que se possam enviar através do protocolo RS-232 com as configurações escolhidas.

A função 'putsUART1' transforma uma *string* (conjunto de caracteres) nas tramas a enviar, e de seguida envia-as. Contudo o espectro é composto por valores, ou seja, por números inteiros, sendo então necessário converter esses números inteiros em *string* para que se possa utilizar a função 'putsUART1', esta tarefa é realizada pela função 'sprintf' que escreve na *string* pretendida o valor da variável indicada (esta pode ser de qualquer tipo). O algoritmo do envio do espectro pela porta série representa-se pelo seguinte fluxograma.

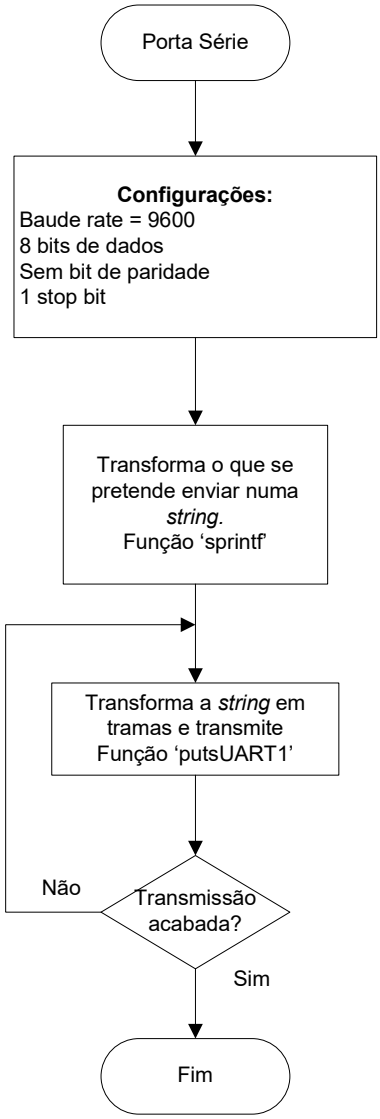


fig 76 – Algoritmo do envio do espectro pela porta série.

Capítulo 11 – Resultados

Após toda a implementação do sistema de reconhecimento de apito, chegou-se a um sistema final, na forma de uma placa de circuito impresso (PCB). Para testar o sistema final e mesmo durante a sua implementação foram feitos vários testes, cujos resultados se apresentam neste capítulo, de forma a demonstrar até que ponto os objectivos foram cumpridos.

11.1 Placa PCB

O sistema foi construído fisicamente numa placa de circuito impresso, tentando-se construir o mais compacto possível de modo a que o sistema fique o menos volumoso possível. O resultado é uma placa PCB de 103 mm × 50 mm ilustrada na figura seguinte.

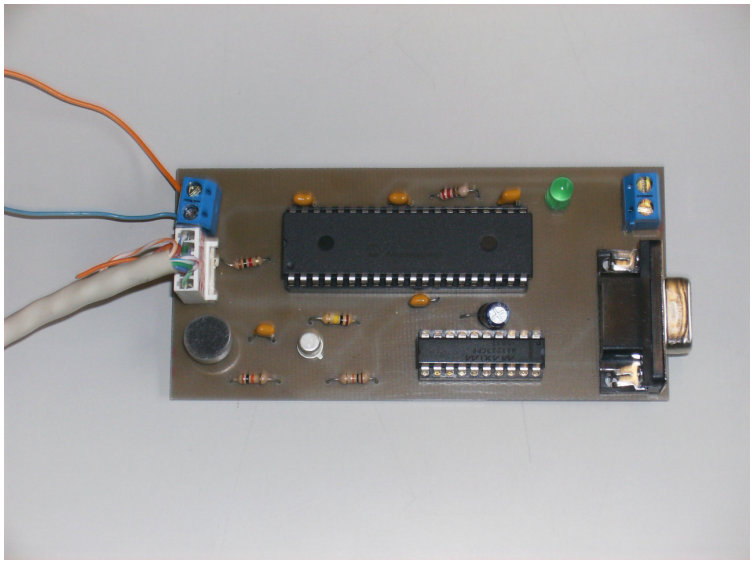


fig 77 – Placa PCB da montagem do sistema.

A placa PCB contém todos os componentes que fazem parte do projecto:

- Microfone;
- Componentes do pré-amplificador;
- Microcontrolador dsPIC30F4013;
- Circuito integrado MAX 233, de interface RS-232;
- LED indicador de resultado;
- Conector de saída do resultado;

- Conector de alimentação;
- Conector de programação;
- Conector da porta série.

NOTA: O esquemático e o desenho da placa encontram-se em anexo.

11.2 Testes ao ADC e FFT

Os testes ao correcto funcionamento dos blocos intermédios de aquisição de sinal (ADC) e cálculo espectral (FFT) foram realizados durante a sua implementação quando se achou conveniente, para detectar possíveis erros de implementação.

Para fazer estes testes foi usado o software da porta série, implementado para enviar o espectro, explicado no capítulo anterior, adaptado no caso dos testes ao ADC. Fez-se testes para vários sinais de entrada, sinusóides de diferentes frequências e sinais do som de apito, dos quais se passa a descrever 4 sinais distintos:

- Sinusóide de baixa frequência (100Hz);
- Sinusóide de média frequência (2kHz);
- Sinusóide de frequência máxima (4kHz);
- Som do apito.

O resultado dos testes (após o correcto funcionamento dos 2 blocos) é o esperado, ou seja, no caso das sinusóides, obtém sinusóides discretas com 256 pontos cada na saída do ADC, e um espectro onde no valor correspondente à sinusóide tem o valor da amplitude da mesma e os restantes valores são nulos. No teste ao sinal de apito, a saída do ADC apresenta um sinal sonoro discreto de 256 amostras, contudo como já foi visto, um sinal sonoro temporal é pouco perceptível, então o teste à funcionalidade dos 2 blocos para um sinal de apito faz-se apenas com o resultado da FFT, este apresenta uma forma semelhante à forma obtida nas análises do capítulo 4, como esperado, o que significa que o ADC e a FFT estão a funcionar correctamente. De seguida apresenta-se os resultados dos testes aos 4 sinais, na forma de gráfica.

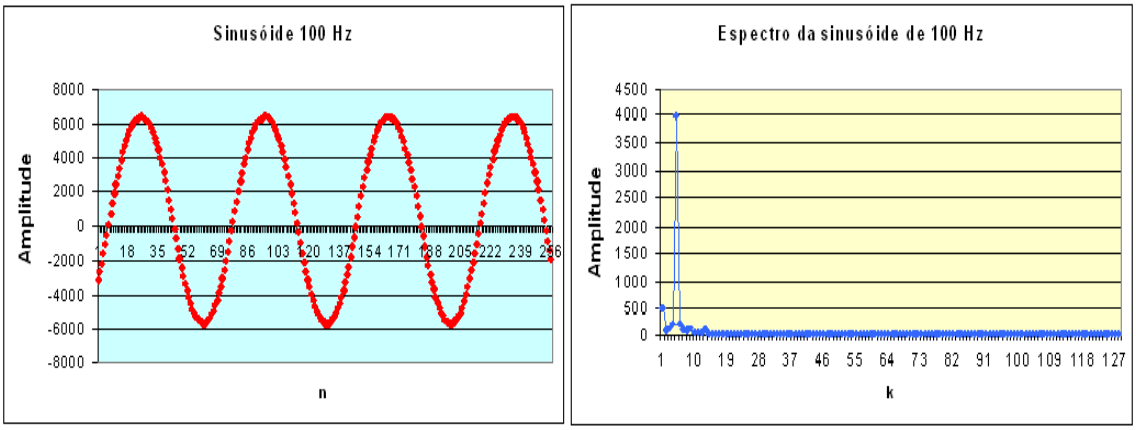


fig 78 – Sinusóide de 100 Hz e respectivo espectro no sistema.

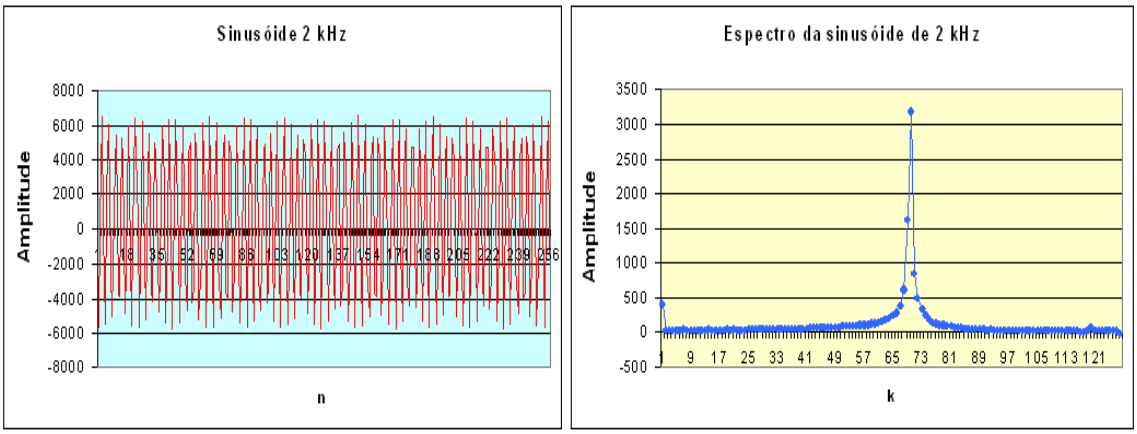


fig 79 – Sinusóide de 2 kHz e respectivo espectro no sistema.

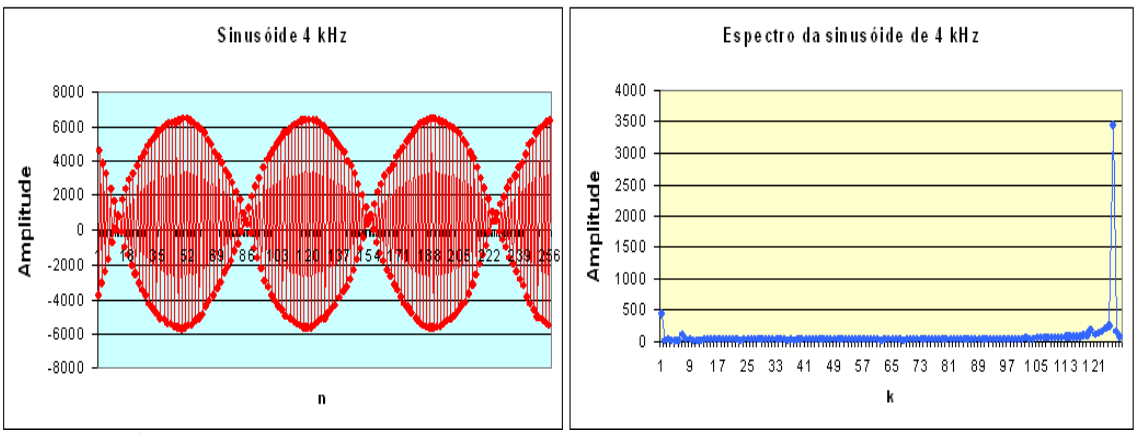


fig 80 – Sinusóide de 4 kHz e respectivo espectro no sistema.

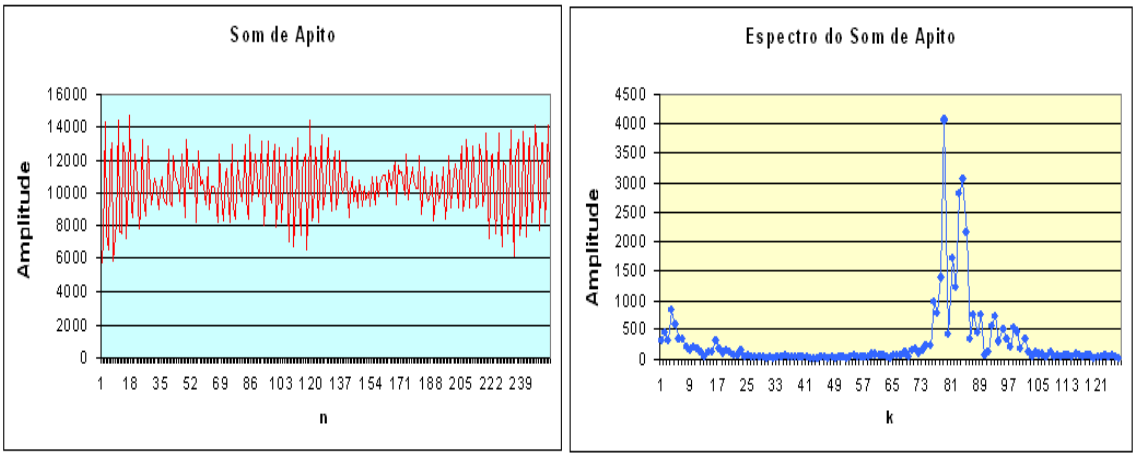


fig 81 – Som de apito e respectivo espectro no sistema.

11.3 Testes à Funcionalidade do Sistema

11.3.1 Reconhecimento de Apito

Reconhecer um apito é o principal objectivo deste sistema, após a implementação de todas as partes do sistema, o primeiro teste realizado foi reconhecer um apito, para tal usou-se uma máscara bastante larga, ou seja, com elevada tolerância, com os seguintes parâmetros da máscara:

- A=500;
- f1=65;
- f2=95

O parâmetro correspondente ao número de iterações (R) foi usada com o menor valor possível, ou seja, R=1, visto que aqui o objectivo não é filtrar ruído.

Após ajustados os parâmetros, fez-se o teste com uma pessoa a soprar num apito a cerca de 1 metros do sistema. **O resultado foi positivo**, o LED que estava apagado, ligou-se quando a pessoa soprou no apito, indicando que de facto existe um som de apito.

Após a realização de um teste com a máscara larga, usou-se o software de enviar o espectro pela porta série para conhecer o espectro a cerca de 10 metros, e ajustar o sistema para detectar aproximadamente até essa distância, o espectro obtido é o seguinte:

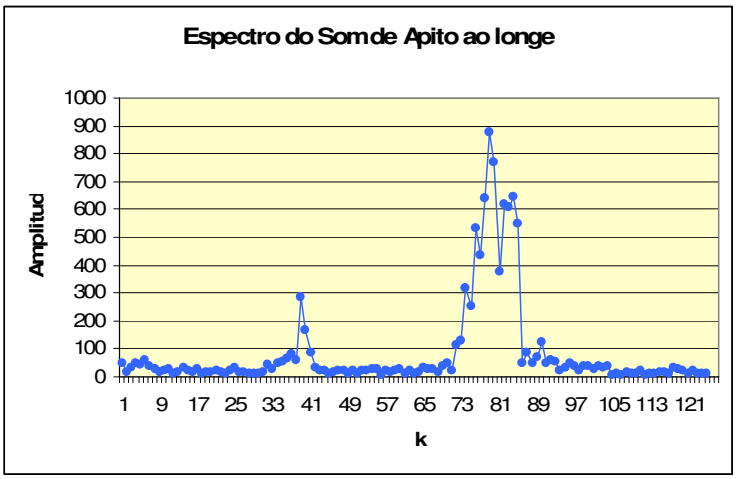


fig 82 – Espectro do som de apito a cerca de 10 metros.

Nesta amostra a frequência de pico está em k=79, e tem um valor de 875. Então com base nestes valores estreitou-se a máscara para os seguintes valores:

- A=850;
- f1=75;
- f2=85;

Com esta máscara, e mantendo-se R=1, a mesma pessoa soprou várias vezes aumentando a distância até chegar à distância que tinha servido de referência para obter o espectro, todas as vezes que se soprou **o resultado foi positivo**, ou seja, o LED ligou-se.

11.3.2 Imunidade ao Ruído

A imunidade ao ruído é também um objectivo de elevada importância para este sistema, o sistema deve ligar o LED apenas quando se fizer sentir o som de um apito em redor. Para testar a imunidade ao ruído utilizou-se a mesma pessoa com o mesmo apito dos testes ao reconhecimento de apito, utilizando-se também a máscara já ajustada:

- A=850;
- f1=75;
- f2=85;

Inicialmente utilizou-se R=1, para testar os ruídos que a máscara conseguia filtrar. Testou-se uma grande variedade de ruídos (gritar, assobiar, bater palmas, falar

várias pessoas ao mesmo tempo, etc.), verificou-se que com a máscara consegue-se filtrar quase todos tipos de ruído, sendo o **resultado positivo em quase todos os ruídos**, ou seja, o LED manteve-se desligado. Os ruídos que não obedeceram à regra, foram escassos bateres de palmas e alguns assobios de dedos, estes ruídos conseguem ultrapassar a máscara em algumas amostras, mas não mantêm o LED ligado enquanto duram, devido ao facto de serem muito inconstantes. Estes ruídos tinham sido já analisados nos testes do capítulo 4, onde se revelaram potenciais influentes no sistema. Para resolver este tipo de problemas o sistema tem o parâmetro R, este deve ser aumentado para filtrar estes ruídos.

Fizeram-se novos testes aumentando o parâmetro R, e verificou-se que **para R≥2 o som das palmas é filtrado**. Com R=2 continuou a haver ainda alguns assobios de dedos que ligavam o LED, então foi-se aumentando R e cada vez havia menos amostras de assobios de dedos que enganavam o sistema, até que **para R≥10 nenhum assobio de dedos conseguiu enganar o sistema**. Para o valor de R=10 o sistema reconhece o apito normalmente, contudo com uma pequena latência no tempo de resposta (menos de um segundo), o que implica que o apito tenha que ter obrigatoriamente um tempo de duração superior ao tempo da latência para ser reconhecido pelo sistema.

11.3.3 Distinção de 2 Apitos

O sistema de reconhecimento de apito foi projectado de modo a ser ajustável, um dos motivos que implicou o seu projecto deste modo, é o facto de tornar o sistema ajustável ao apito que se vai reconhecer, e consequentemente poder distinguir 2 apitos diferentes. Contudo verificou-se nas análises do capítulo 4 que apesar de 2 apitos terem uma frequência de pico do seu espectro também diferente, existem factores como a intensidade do sopro, que influenciam mais significativamente o valor da frequência de pico, fazendo com que os espectros possam ser iguais em algumas amostras e fazer a distinção entre 2 apitos deixe de ser possível.

Para testar o sistema na distinção de 2 apitos, ajustou-se os parâmetros para um apito, e soprou-se com o outro, para o sistema distinguir um apito do outro o LED deveria manter-se desligado. O resultado deste teste no sistema foi o esperado, **o sistema não consegui distinguir os 2 apitos**, ligando-se sempre o LED no sopro de qualquer um dos apitos.

Apesar de neste teste o sistema não ter conseguido distinguir 2 apitos, isto deve-se ao facto que fisicamente o som dos 2 apitos é igual em alguns instantes, e os apitos apesar de diferentes são os 2 apitos de árbitro, que produzem sons muito parecidos. O sistema continua com a capacidade de poder distinguir 2 apitos, basta que produzam um som bastante diferente, por exemplo, um deles ter uma frequência de pico acima dos 3 kHz.

Capítulo 12 – Conclusões e Trabalho Futuro

12.1 Conclusões

O principal objectivo no projecto do sistema de reconhecimento de apito em ambientes ruidosos é reconhecer um apito com a máxima robustez ao ruído, contudo os objectivos secundários de fazer um sistema pouco volumoso e de baixo custo, capaz de funcionar em qualquer tipo de robô futebolista, são factores que identificam este sistema e o distinguem dos outros sistemas de reconhecimento de som. Verificando-se este facto na pesquisa do estado da arte, onde não se encontrou nenhum sistema que cumpre integralmente os objectivos para este sistema.

O microcontrolador (DSC) escolhido é adequado para este sistema, visto que a sua arquitectura permite fazer operações de processamento de sinal, como o caso da FFT, em tempos bastante reduzidos, isto porque possui uma unidade de cálculo especial para determinadas operações utilizadas em processamento de sinal. Consegue-se também implementar neste microcontrolador operações de processamento de sinal com mais facilidade, devido à biblioteca dedicada a estas operações que o fabricante disponibiliza.

A qualidade do microfone escolhido é suficiente, visto que este consegue captar frequências na gama que se pretende analisar. O pré-amplificador consegue também colocar o sinal no nível de tensão desejado, contudo este é composto por vários elementos com uma dimensão considerável, o que contribui para um aumento significativo do volume final do sistema.

Tal como as unidades de hardware, todas as unidades de software implementadas cumprem a sua função no sistema, ou seja, a unidade ADC consegue obter um sinal digital a partir do sinal analógico, com os parâmetros pretendidos. A FFT calcula o espectro como pretendido e a Percepção devolve uma decisão de acordo com o algoritmo que deu origem à sua implementação.

O algoritmo de percepção foi desenvolvido de um modo simples, com base em comparações com valores de uma máscara. Na prática verificou-se que este método consegue ter uma boa eficácia de reconhecimento e uma boa robustez ao ruído.

Nos testes ao sistema, verificou-se que o sistema reconhece sempre o apito, e que ajustando adequadamente os parâmetros, consegue filtrar todos os ruídos. Já no teste à

distinção entre 2 apitos, o sistema não cumpriu o objectivo, este insucesso deve-se ao facto de que os 2 apitos utilizados são muito semelhantes e em determinados instantes são fisicamente iguais, estando o sistema em si isento de culpas neste resultado. Contudo o sistema é capaz de o fazer a distinção entre 2 apitos, basta que sejam bastante diferentes, por exemplo, um deles produzir um som bastante mais agudo com frequência de pico acima dos 3 kHz.

O resultado do sistema é uma placa PCB com as dimensões 103 mm × 50 mm, considerando-se cumprido o objectivo de fazer um sistema pouco volumoso, contudo, o volume deste sistema pode ser menor sem alterar o seu funcionamento, alterando apenas o desenho da placa, utilizando tecnologia de montagem superficial SMD – *Surface Mount Device*, e componentes de encapsulamentos menores.

Este sistema foi desenvolvido para fazer o reconhecimento de apito em robôs futebolistas, contudo a sua utilidade não se limita a esta tarefa, já que alterando a máscara pode-se reconhecer outros tipos de sons.

12.2 Trabalho Futuro

O sistema de reconhecimento de apito está a funcionar correctamente, cumprindo os objectivos apresentados no início, o trabalho proposto para o melhoramento deste projecto tem em vista a redução do seu volume. A tecnologia de montagem superficial pode reduzir significativamente o tamanho da placa, então deve-se escolher componentes de encapsulamento mais pequeno que o utilizado, então desenhar-se e fazer uma placa com a tecnologia SMD. O tamanho do circuito pré-amplificador pode também ser reduzido significativamente, utilizando-se um circuito integrado compacto que faça esta tarefa, como os utilizados em dispositivos móveis.

Após se obter um apito de som mais agudo, deve-se testar de novo a distinção entre 2 apitos, obtendo-se valores dos parâmetros que permitam fazer a distinção de forma eficaz.

Referências

[1] Microsoft Hellhound – RoboCup 2006
http://www.microsoft-hellhounds.de/en/events/competitions/robocup06/

[2] Whistle Recogniser for RoboCup
http://robocup.elet.polimi.it/MRT/Projects/WR/HtmlDoc/index.htm

[3] Voice Recognition Robotic Car
http://ltc.cit.cornell.edu/courses/ee476/FinalProjects/s2006/avh8_css34/avh8_css34/index.html

[4] YouTube (video) – Microsoft Vista Speech Recognition Tested
http://www.youtube.com/watch?v=KyLqUf4cdwc

[5] IEEE (Artigo) – Microcontroller implementation of a voice command recognition system for human-machine interface in embedded systems
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/10734/33857/01612576.pdf?arnumber=1612576

[6] Wikipedia – definição de Som
http://pt.wikipedia.org/wiki/%C3%81udio

[7] Ministério da Ciência e da Tecnologia – Som
http://telecom.inescn.pt/research/audio/cienciaviva/index_osom.htm

[8] VIDEO.GRAFIAS – Som
http://www.univ-ab.pt/~bidarra/hyperscapes/video-grafias-107.htm

[9] Clix – Palco Principal - O que é o Som ? Alguns conceitos e aplicações
http://palcoprincipal.clix.pt/artigos/Artigo/o_que_e_o_som_alguns_conceitos_e_aplicacoes

[10] Mercado Livre – Níveis de Intensidade Sonora
http://guia.mercadolivre.com.br/niveis-intensidade-sonora-decibeis-dbspl-tabela-15008-VGP

[11] Universidade de São Paulo (download do software FFTMusEV)
http://artemis.ffclrp.usp.br/softwareP.htm

[12] Wikipedia – Microfone
http://pt.wikipedia.org/wiki/Microfone

[13] Guide for Electret Condenser Microphones
http://www.hosiden.co.jp/web/english/web/products/pdf/e_on06_mic.pdf

[14] Wikipedia – Electret microphone
http://en.wikipedia.org/wiki/Electret_microphone

[15] Simple audio preamplifier
http://www.reconnsworld.com/audio_simplepreamp.html

[16] Microchip – 16-bit Embedded Control Solutions
http://www.microchip.com/stellent/groups/dspic_sg/documents/devicedoc/en012562.pdf

[17] Microchip – dsPIC® Digital Signal Controllers
http://ww1.microchip.com/downloads/en/DeviceDoc/DS-70095K.pdf

[18] Microchip – dsPIC30F Family Reference Manual
http://ww1.microchip.com/downloads/en/DeviceDoc/70046E.pdf

[19] Microchip dsPIC30F3014/4013 Data Sheet
http://ww1.microchip.com/downloads/en/DeviceDoc/70138E.pdf

[20] Universidade do Minho (Departamento de Electrónica Industrial) – Disciplina de Processamento Digital de Sinal
http://disciplinas.dei.uminho.pt//index.php?op=YXVsYXM=

[21] Conversor Analógico-Digital
http://www.eletrica.ufpr.br/artuzi/apostila/cap4/pg11.html

[22] Universidade Federal do Paraná (Dep. De Engenharia Elétrica) – Disciplina de Instrumentação Eletrônica
http://www.eletrica.ufpr.br/marlio/mestrado/aulas/amostragem1.pdf

[23] Wikipedia – Espectro
http://pt.wikipedia.org/wiki/Espectro

[24] UNIFESP Virtual – Sons digitalizados
http://www.virtual.epm.br/material/ead/som.htm

[25] eletronica.org – Padrão Serial RS-232
http://www2.eletronica.org/artigos/eletronica-digital/padrao-serial-rs-232/

[26] +5V-Powered, Multichannel RS-232 Drivers/Receivers – Datasheet
http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf

Bibliografia

Site de Internet
Whistle Recogniser for RoboCup
http://robocup.elet.polimi.it/MRT/Projects/WR/HtmlDoc/index.htm

Site de Internet
Voice Recognition Robotic Car
http://ltc.cit.cornell.edu/courses/ee476/FinalProjects/s2006/avh8_css34/avh8_css34/index.html

Video
Microsoft Vista Speech Recognition Tested
http://www.youtube.com/watch?v=KyLqUf4cdwc

Artigo
IEEE - Microcontroller implementation of a voice command recognition system for human-machine interface in embedded systems
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/10734/33857/01612576.pdf?arnumber=1612576

Livro
Ifeachor, Emmanuel C; Jervis, Barrie W. *Digital Signal Processing: A Practical Approach* – 2nd Edition

Site de Internet
Apontamentos da disciplina de Processamento Digital de Sinal do Mestrado Integrado em Engenharia Electrónica Industrial e Computadores – Universidade do Minho
http://disciplinas.dei.uminho.pt//index.php?op=YXVsYXM=

Manual
Microchip – dsPIC30F Family Reference Manual
http://ww1.microchip.com/downloads/en/DeviceDoc/70046E.pdf

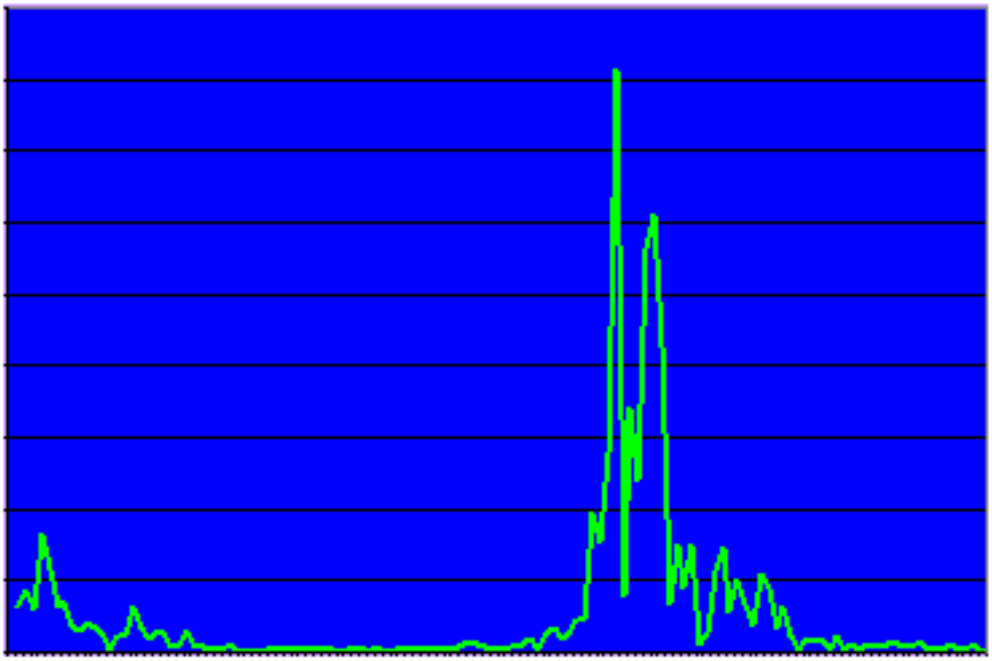
Manual
Microchip dsPIC30F3014/4013 Data Sheet
http://ww1.microchip.com/downloads/en/DeviceDoc/70138E.pdf

Manual
Microchip – MPLAB C Compiler C30 User's Guide
http://asl.epfl.ch/education/courses/MicroInfo/doc/51284D.pdf

Manual
Guide for Electret Condenser Microphones
http://www.hosiden.co.jp/web/english/web/products/pdf/e_on06_mic.pdf

Site de Internet
Ministério da Ciência e da Tecnologia – Som
http://telecom.inescn.pt/research/audio/cienciaviva/index_osom.htm

Anexo I – Manual de Utilização do Sistema de Reconhecimento de Apito em Ambientes Ruidosos



Este sistema de reconhecimento de apito, trata-se de um dispositivo que consegue reconhecer automaticamente um apito entre o ruído de fundo. Permite ajuste de parâmetros de modo a adaptar o sistema ao apito que pretende reconhecer e ao tipo de ambiente onde vai funcionar.

O sistema de reconhecimento de apito em ambientes ruidosos é constituído por:

- Dispositivo de reconhecimento (placa PCB);
- Software para o microcontrolador do dispositivo:
  - Software de reconhecimento;
  - Software para enviar o espectro para o PC;
- Software de simulação de parâmetros.

Nota: todo o software se encontra na pasta 'Apito final'. Deve ser mantida uma cópia de segurança dessa mesma pasta de origem para o caso de se ter feito alterações indesejáveis.

Para se poder usar todas as funções do sistema, são requeridos alguns softwares adicionais instalados no computador do utilizador, não sendo necessário utilizar exactamente todos os softwares aqui indicados, contudo os softwares aqui indicados estão devidamente testados, servindo de referência neste manual. Os softwares necessários à total utilização das funções do sistema são os seguintes:

- MPLAB IDE versão 6.2 ou superior;
- MPLAB C30;
- Microsoft Office Excel 2003;
- RComSerial (software de interface com a porta série).

É necessário ainda algum hardware para interface entre o dispositivo e o PC, para além do próprio PC, tal como no software, o hardware aqui indicado não é o único que se pode utilizar, mas já está testado e serve de referência para este manual. O hardware necessário consiste em:

- Programador de PIC's In-Circuit Debugger – Microchip MPLAB ICD2;
- Conversor USB - Série ou extensão da porta série.

As funções que o sistema permite realizar são as seguintes:

I. Fazer o reconhecimento de apito;
II. Ajustar parâmetros;
III. Enviar o espectro de determinado sinal;
IV. Simular os parâmetros.

I. Fazer o reconhecimento de apito

Esta é a função principal deste sistema, para tal existe um software de reconhecimento, sendo apenas necessário introduzir esse software no microcontrolador dsPIC 30F4013 do dispositivo de reconhecimento de apito. De seguida descreve-se ao pormenor a realização desta tarefa:

1. Ligar o programador Microchip MPLAB ICD2 ao PC através do cabo USB e o dispositivo ao programador através do cabo que o programador inclui. O dispositivo deve ser alimentado com uma fonte de 5V. As ligações devem ser feitas de acordo com as seguintes figuras. Na figura 1-a) mostra-se as ligações de todos os elementos e na figura 1-b) mostra-se só as ligações do dispositivo de reconhecimento de apito com mais pormenor.

fig. 1-a) Ligação entre PC, programador e dispositivo.

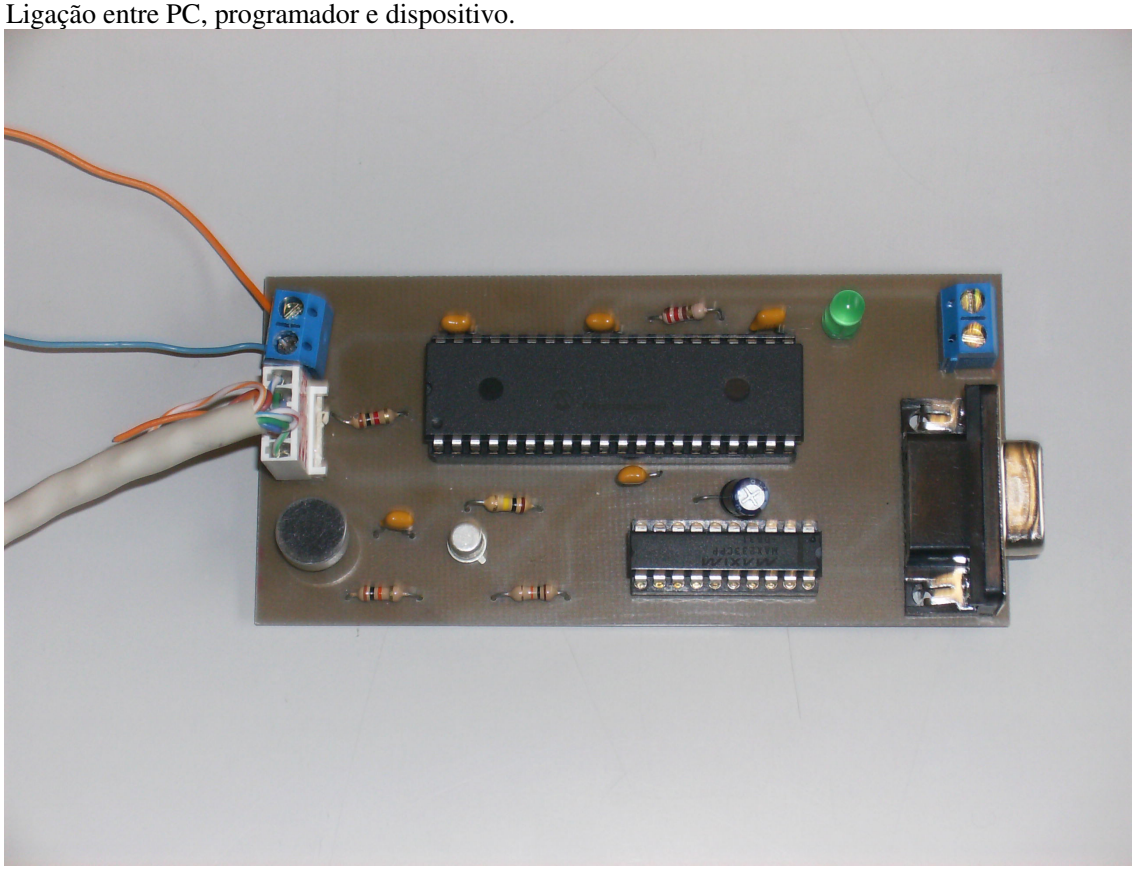


fig. 1-b) Ligações no dispositivo.

2. Abrir o ficheiro ‘Apito_final2.mcw’ com o programa MPLAB IDE, este ficheiro encontra-se na directoria ‘…..\Apito final\programa para reconhecimento’.

3. Fazer por software a conexão entre o MPLAB IDE e o ICD2, carregando no botão 'Reset and connect to ICD', a localização desse botão no MPLAB IDE encontra-se assinalada na figura seguinte.

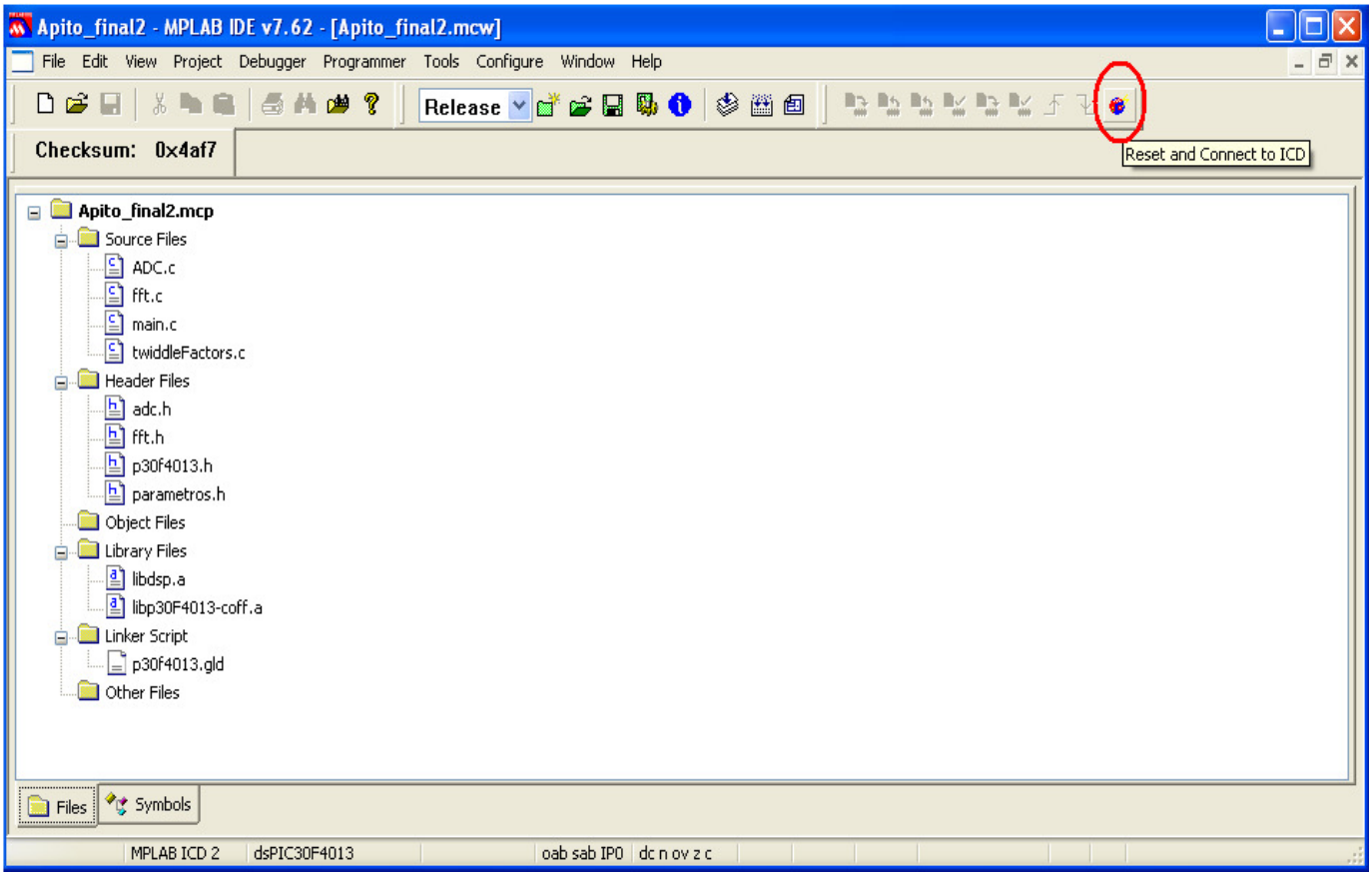


fig. 2 – Janela MPLAB IDE indicando o botão 'Reset and connect to ICD'.

4. Compilar o programa, fazendo click no botão 'Build All'.

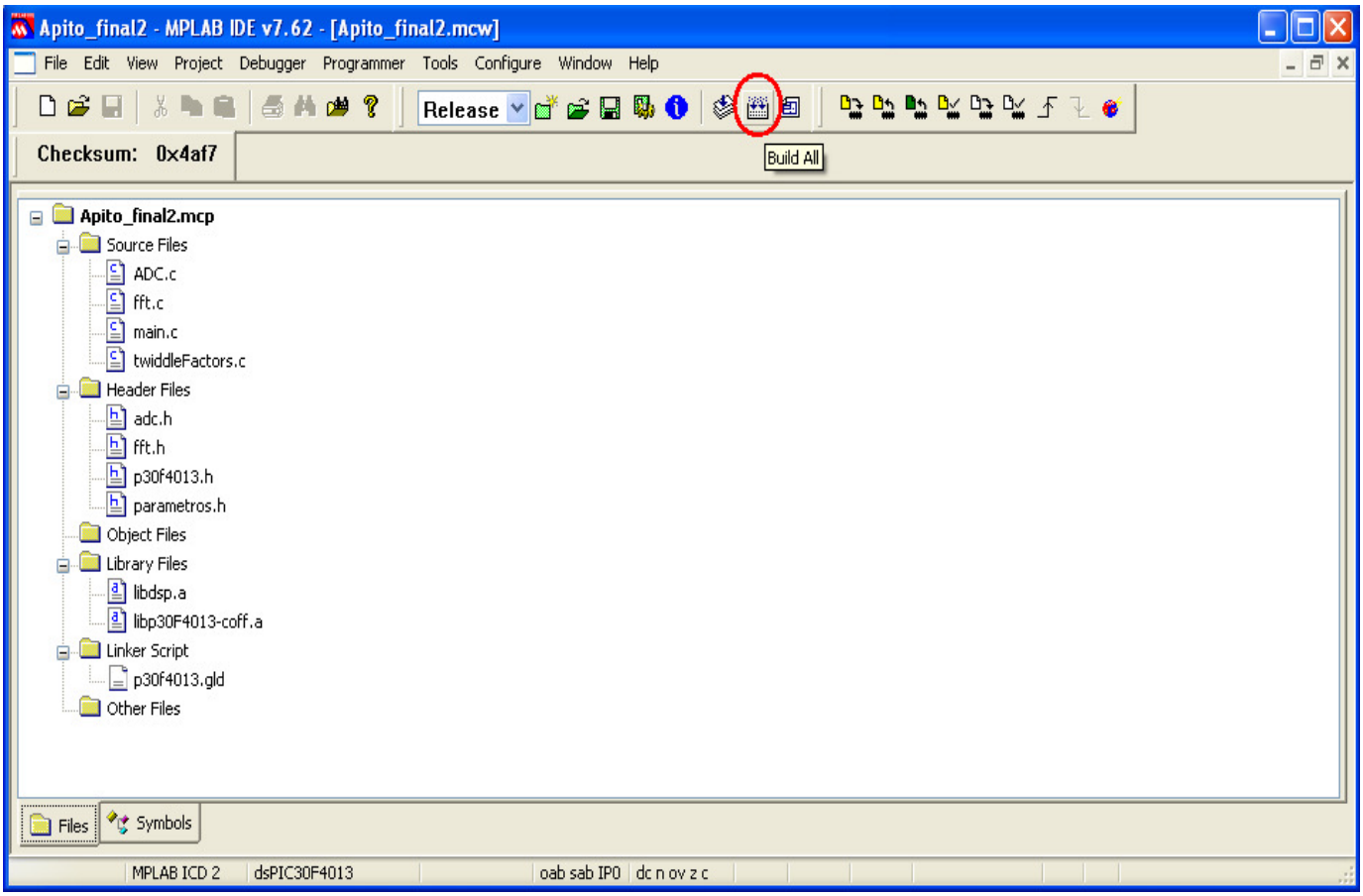


fig. 3 – Janela MPLAB IDE indicando o botão 'Build All'.

5. Enviar o programa para o microcontrolador dsPIC30F4013 do dispositivo, fazendo click no botão 'Program target device'.

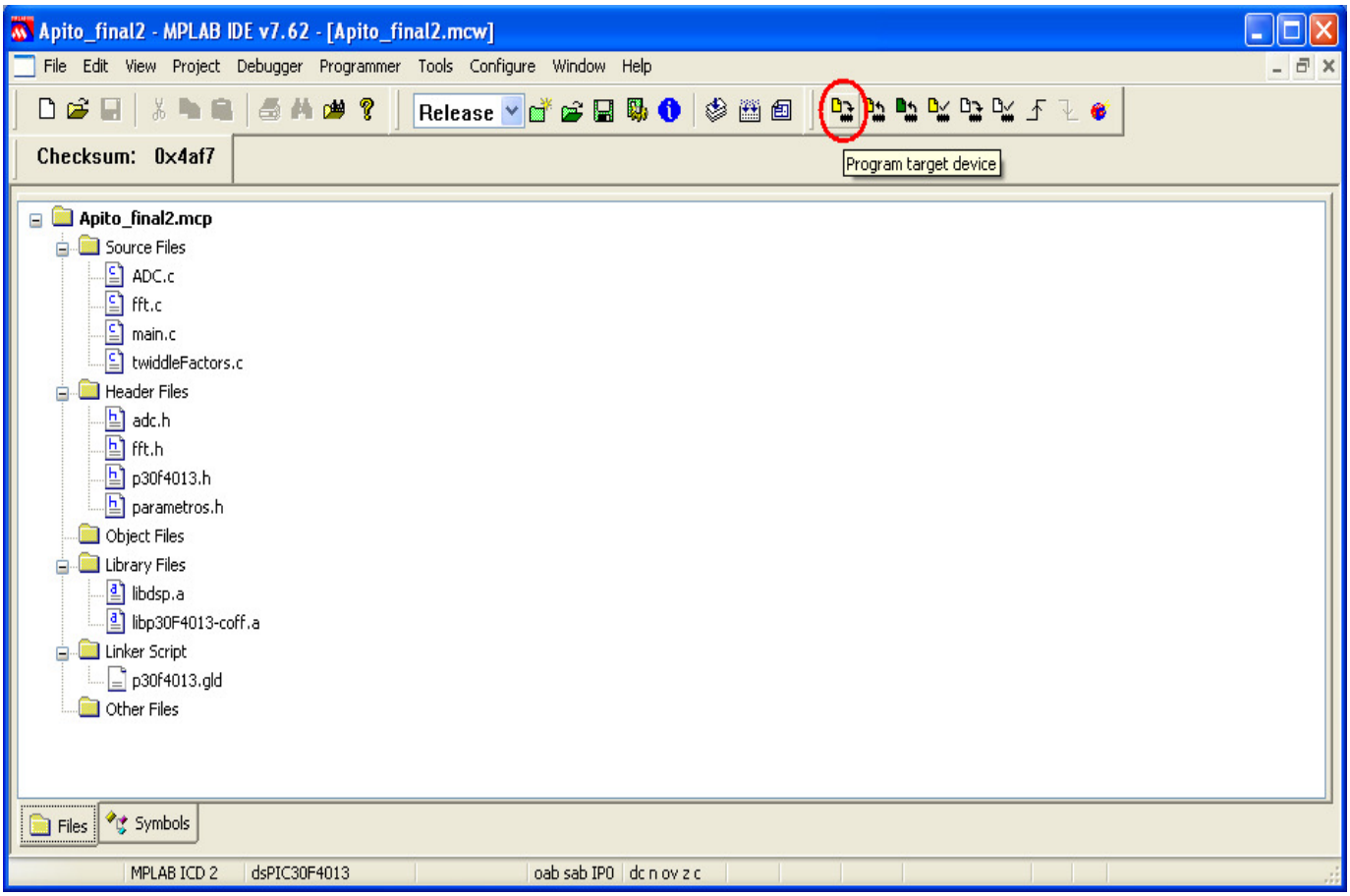


fig. 4 – Janela MPLAB IDE indicando o botão 'Program target device'.

6. Retirar o cabo de ligação entre o dispositivo e o ICD2.

Após estes passos, o dispositivo necessita apenas de estar alimentado com 5V para fazer o reconhecimento de apito. Enquanto o apito é detectado, o LED fica ligado e a saída é colocada a nível lógico'1' (5V), quando não é detectado apito, o LED fica apagado e a saída é colocada a nível lógico '0' (0V).

II. Ajustar parâmetros

O Sistema possui 4 parâmetros ajustáveis, estes parâmetros são:

- A – Amplitude mínima na selecção do sinal do sinal de apito.
- f1 –Frequência mínima na selecção do sinal de apito.
- f2 – Frequência máxima na selecção do sinal de apito.
- R – Número de repetições de sinal de apito necessárias para que seja considerado efectivamente sinal de apito.

O valor dos parâmetros deve ser obtido com a ajuda das tarefas III. E IV. Esta tarefa diz respeito apenas à alteração desses parâmetros no programa. Para tal é necessário abrir o ficheiro e antes de o compilar e enviar para o microcontrolador, modificar-lhe os parâmetros. De seguida descreve-se esta tarefa com todo o detalhe.

1. Fazer as ligações (igual ao passo 1 da tarefa I).

2. Abrir o ficheiro (igual ao passo 2 da tarefa I).

3. Fazer a conexão (igual ao passo 3 da tarefa I).

4. No MPLAB IDE abrir o ficheiro ‘parametros.h’.

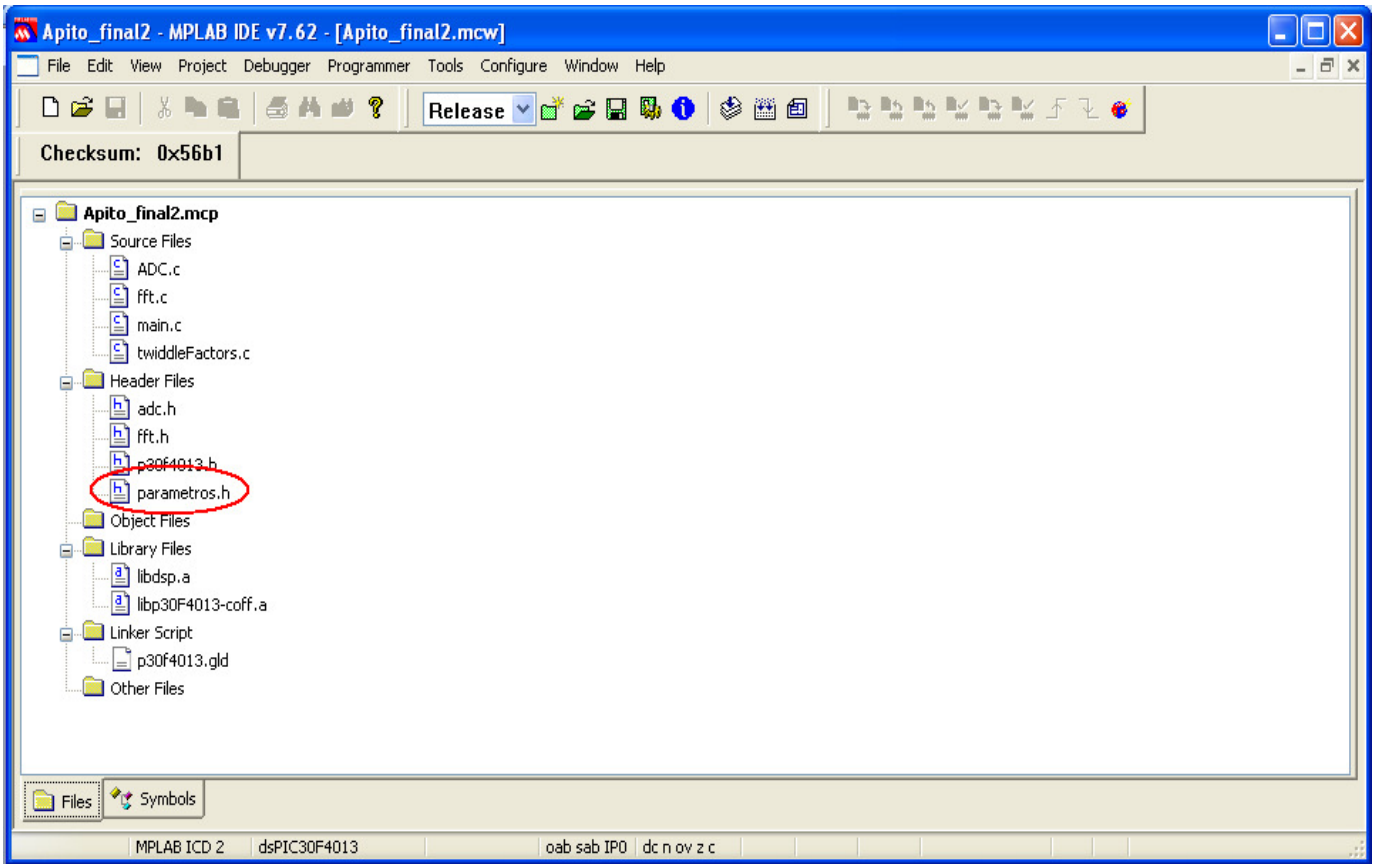


fig. 5 – Janela MPLAB IDE indicando o ficheiro ‘parametros.h’.

5. Alterar o valor dos parâmetros, para tal, altera-se o valor decimal no ‘#define’ correspondente.

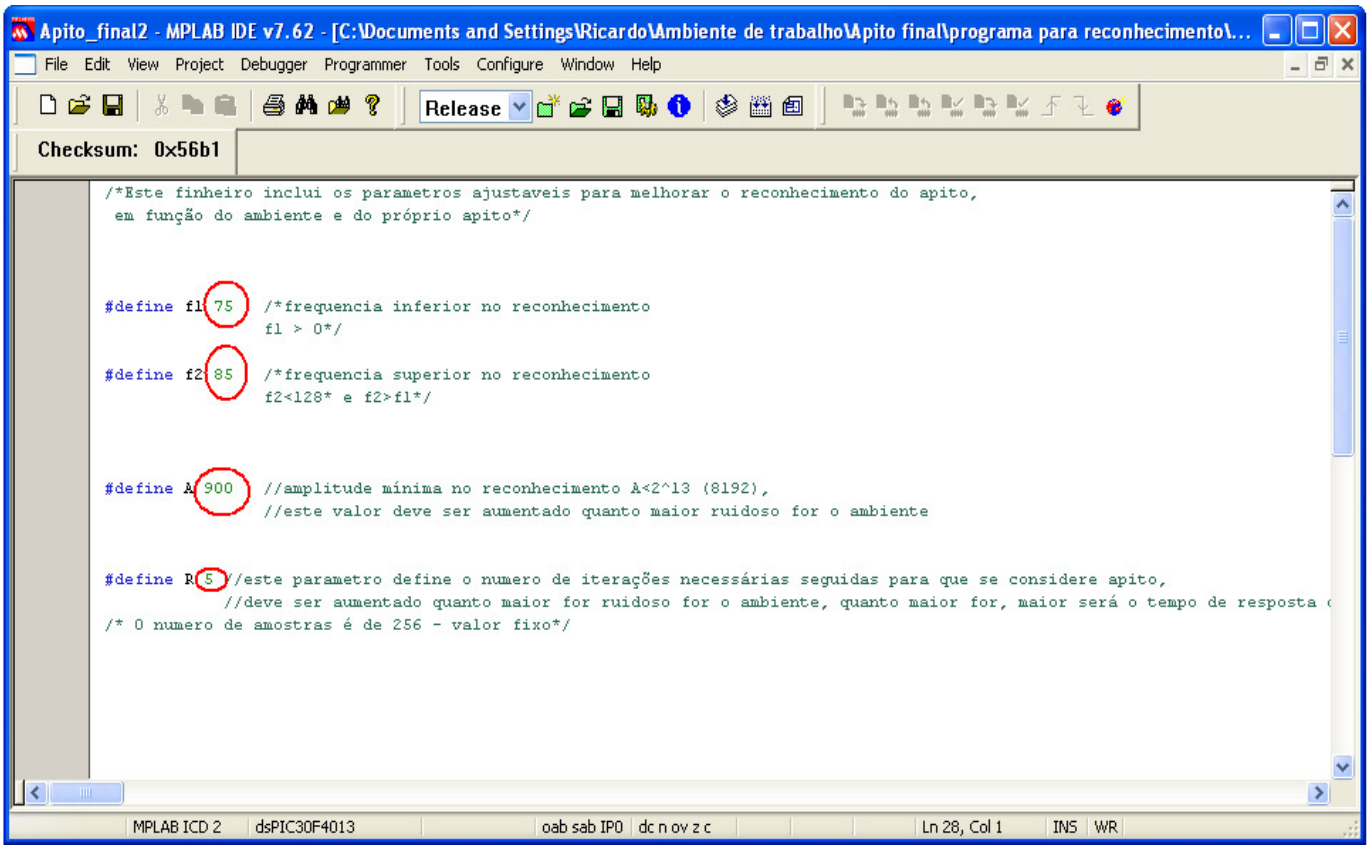


fig. 6 – Janela MPLAB IDE com o ficheiro 'parametros.h' aberto, indicando onde se altera o valor dos parâmetros.

6. Compilar o programa (igual ao passo 4 da tarefa I).

7. Enviar o programa para o microcontrolador (igual ao passo 5 da tarefa I).

8. Retirar as ligações (igual ao passo 6 da tarefa I).

Após estes passos o dispositivo fica a fazer o reconhecimento de apito com os novos parâmetros.

III. Enviar o espectro de determinado sinal

O dispositivo de reconhecimento de apito tem também a possibilidade de enviar para o PC o espectro de determinado sinal, para se ter percepção de como o sinal varia, por exemplo, em função do apito, do arbitro ou da distância que o microfone se encontra do apito. Para fazer esta tarefa existe um programa diferente do de reconhecimento, que calcula o espectro do sinal e envia-o através da porta série. São captadas 256 amostras do sinal, com uma frequência de amostragem de 8 kHz, o que faz com que o sinal seja captado durante 32 ms, após se dar a ordem para começar adquirir o sinal.

Para se fazer esta tarefa coloca-se o programa correspondente no microcontrolador do dispositivo, depois dá-se a ordem para começar e automaticamente o microcontrolador capta o sinal, calcula o espectro e envia-o para a porta série que estará ligada ao PC, onde os dados serão recebidos no programa de interface com a porta série RComSerial. De seguida deve-se guardar esses dados

num ficheiro .txt. Após tidos esses dados guardados, abre-se o Microsoft Excel e importa-se os mesmos dados, fazendo-se depois um gráfico com esses dados para visualizar o gráfico do espectro.

Os passos a seguir na execução desta tarefa encontram-se abaixo.

1. Ligar o programador Microchip MPLAB ICD2 ao PC através do cabo USB e o dispositivo ao programador através do cabo que o programador inclui. Para a conexão entre a porta série do dispositivo e o PC usa-se um conversor USB - Série. O dispositivo deve ser alimentado com uma fonte de 5V. As ligações devem ser feitas de acordo com as seguintes figuras. Na figura 7-a) mostra-se as ligações de todos os elementos e na figura 7-b) mostra-se só as ligações do dispositivo de reconhecimento de apito com mais pormenor.

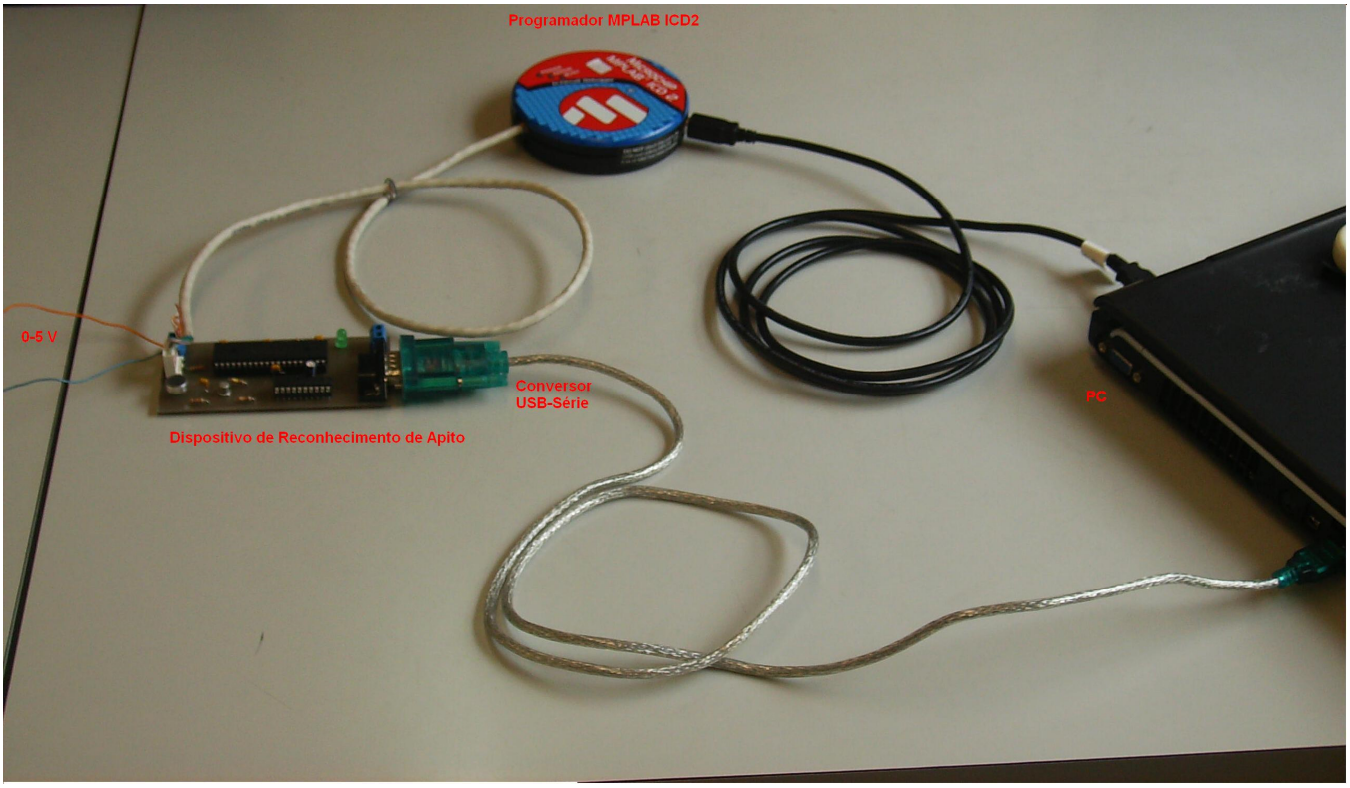


fig. 7-a) Ligação entre PC, programador e dispositivo, incluindo a ligação da porta série.

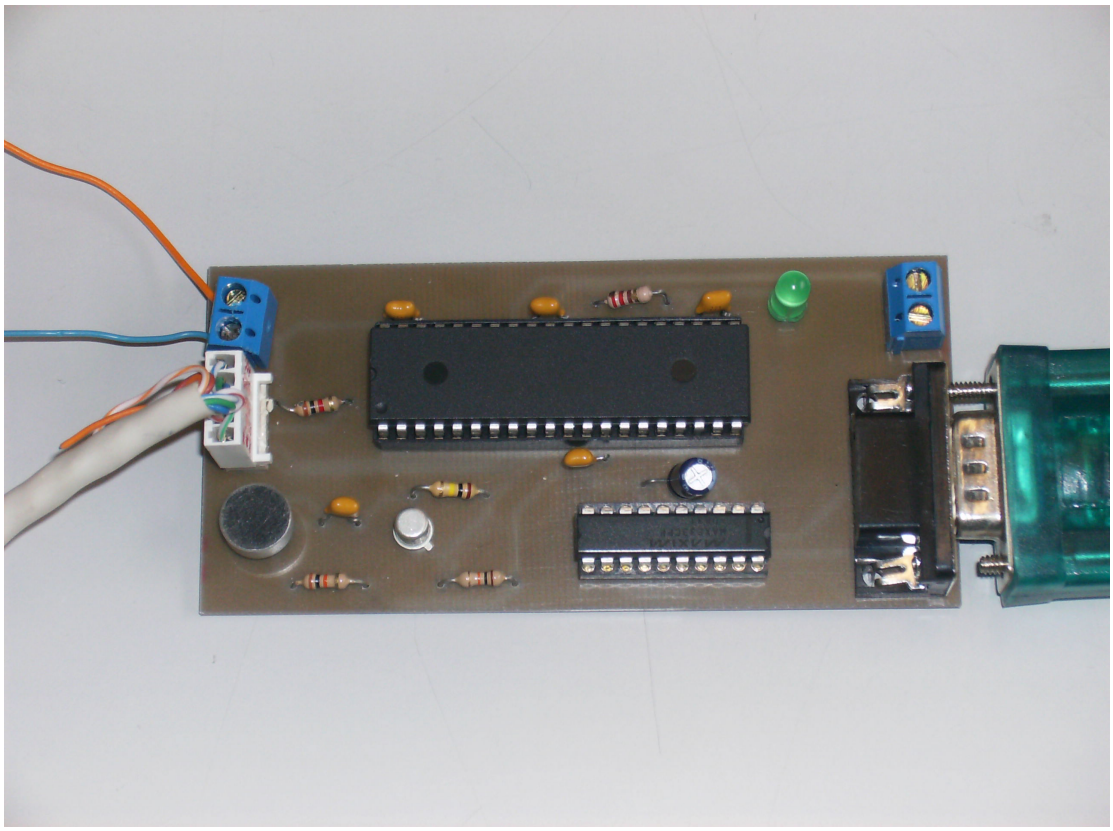


fig. 7-b) Ligações no dispositivo, incluindo porta série.

2. Abrir o ficheiro 'Apito_final1.mcw' com o programa MPLAB IDE, este ficheiro encontra-se na directoria '…..\Apito final\programa para enviar o espectro'.

3. Fazer a conexão (igual ao passo 3 da tarefa I).

4. Compilar o programa (igual ao passo 4 da tarefa I).

5. Enviar o programa para o microcontrolador (igual ao passo 5 da tarefa I).

6. Abrir e configurar o software RComSerial. Deve-se seleccionar a porta atribuída ao conversor USB, uma velocidade de 9600 bps, 8 bits de dados, sem bit de paridade e um bit de paragem. A figura seguinte mostra uma imagem do software com as configurações correctas

   NOTA: a porta atribuída ao conversor é indicada no 'gestor de dispositivos' do Windows no item 'portas (COM e LPT) '.

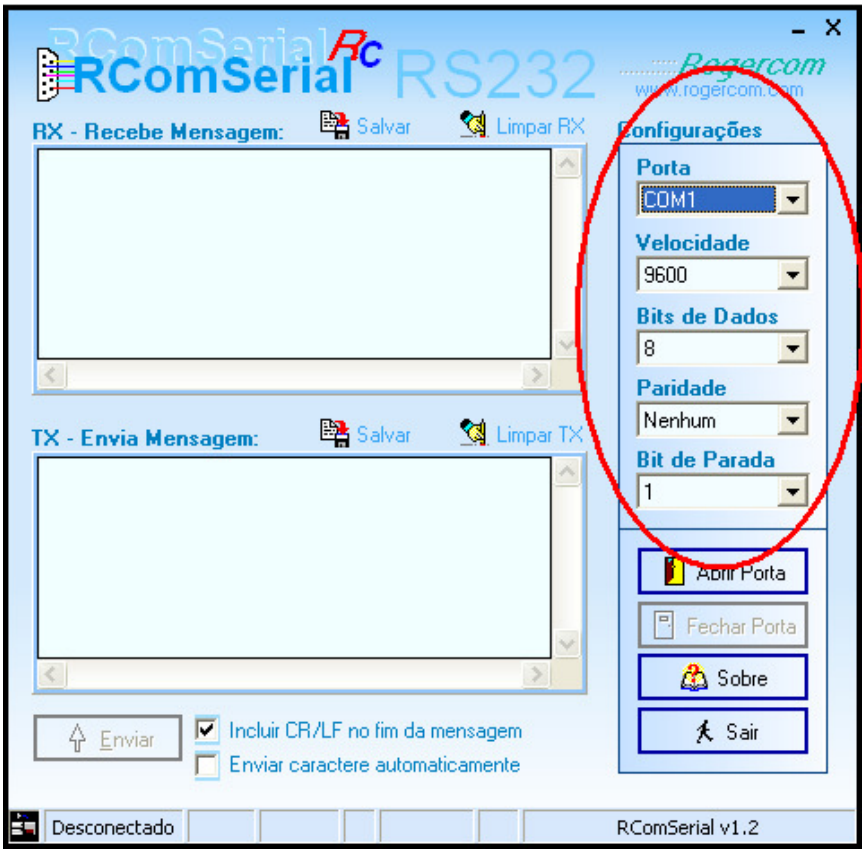


fig. 8 – Janela RComSerial indicando as configurações correctas.

7. Abrir a Porta através do botão 'Abrir Porta'.

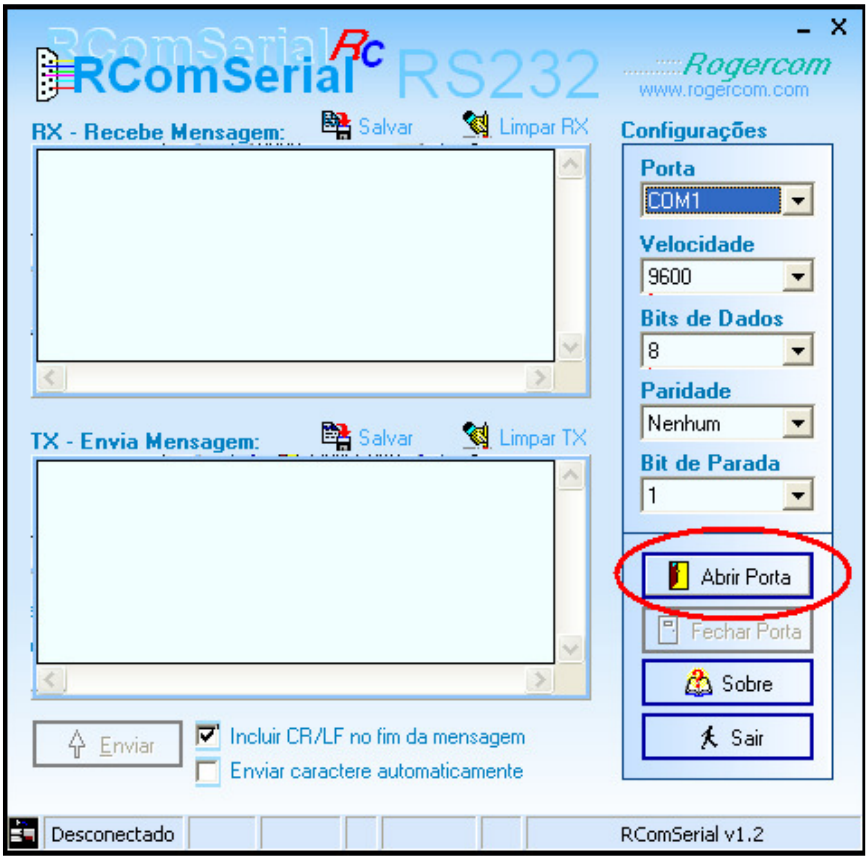


fig. 9 - Janela RComSerial indicando o botão 'Abrir Porta'.

8. Começar a produzir o som (ex.: começar a apitar).

9. Enquanto o som está a ser produzido, fazer click no botão ‘Release from Reset’ do MPLAB IDE, esta é a ordem de começo de captura. Automaticamente o microcontrolador captura o sinal durante 32 ms, calcula o respectivo espectro e envia-o para a porta série. Os valores respectivos ao espectro aparecem imediatamente no RComSerial.

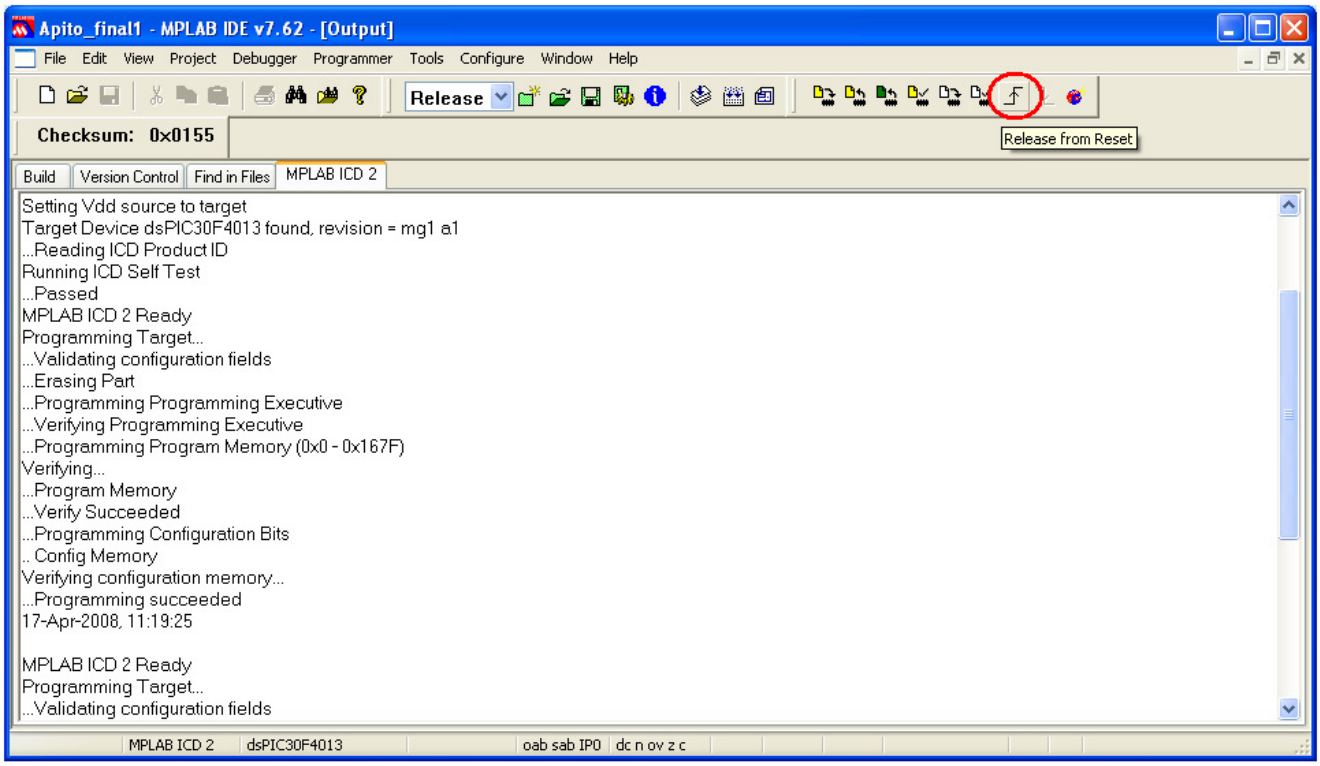


fig. 10 – Janela MPLAB IDE indicando o botão ‘Release from Reset’.

10. Guardar os dados do espectro num determinado ficheiro .txt carregando no botão ‘Salvar’.

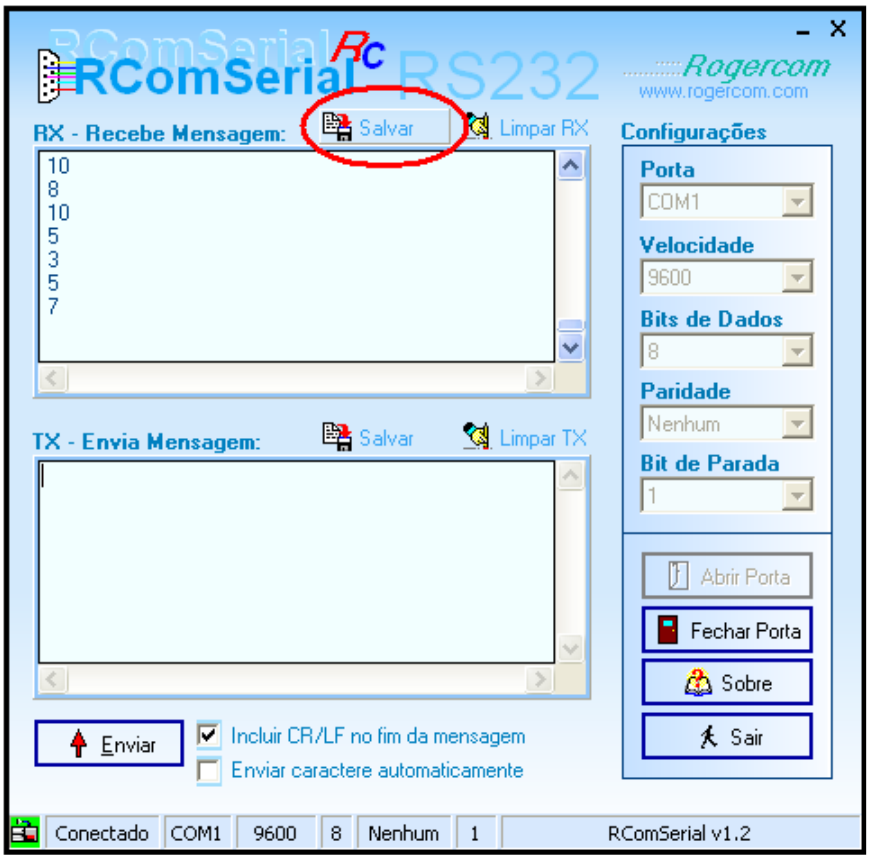


fig. 11 – Janela RComSerial indicando o botão 'Salvar'.

11. Abrir o Microsoft Excel e importar os dados gravados no ficheiro .txt.

    11.1. Abrir a opção 'Importar dados' ('Dados'-> 'Importar dados externos'-> 'Importar dados…').
    11.2. Seleccionar e abrir o ficheiro .txt a partir do respectivo directório.
    11.3. Prosseguir carregando no botão 'Seguinte' ficando definidas algumas opções de importação de dados por defeito, até que apareça o botão 'Concluir'.
    11.4. Nesta parte aparece uma janela que pergunta: "Onde deseja colocar os dados?", aqui faz-se click, com o botão esquerdo do rato, na colona onde se pretende que fiquem os dados (ex: coluna A). Após este passo os dados são colocados na coluna seleccionada.

12. Fazer o gráfico.
    12.1. Seleccionar a coluna onde estão os dados.
    12.2. Carregar no botão 'Assistente de gráficos'.
    12.3. Escolher a opção 'Linhas' no tipo de gráfico e a 1ª opção no subtipo de gráfico, de acordo com a figura seguinte.

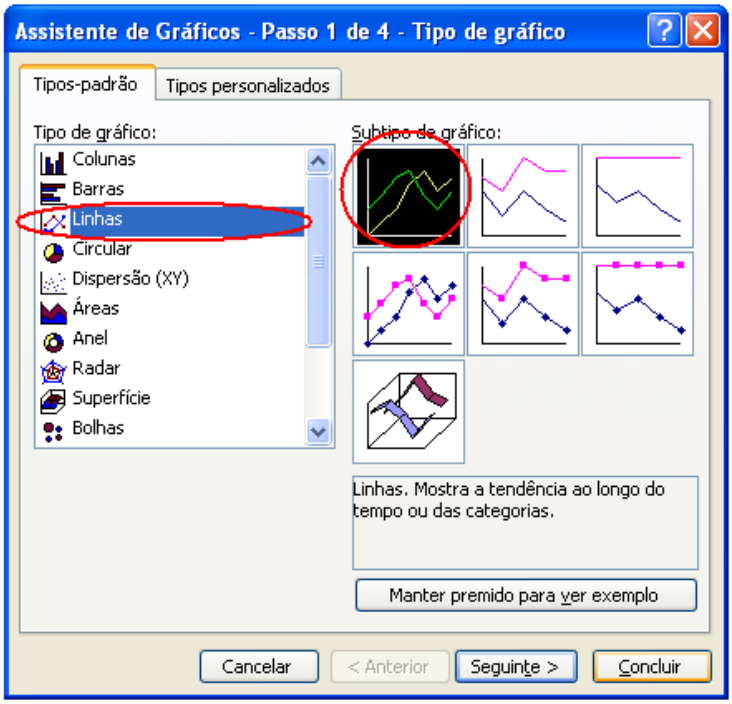


fig. 12 – Janela Assistente de gráficos do Excel indicando a opção que se deve seleccionar.

12.4. Carregar no botão 'Concluir', ficando o gráfico feito.

IV. Simular os Parâmetros

Para se ter uma melhor percepção da influência dos parâmetros no sistema de reconhecimento de apito foi criado um simulador, onde se pode mudar os parâmetros e ver graficamente a consequente variação das zonas de aceitação e rejeição. Para usar o simulador deve-se ter em conta os seguintes aspectos:

- O simulador encontra-se na directoria '… Apito final/Simulação.xls' e deve ser aberto com o Microsoft Excel.
- Os parâmetros estão nas células amarelas, as restastes células não devem ser alteradas.
- A zona de aceitação é representada a verde e a zona de rejeição é representada a vermelho.
- Os parâmetros f1 e f2 a alterar correspondem ao índice de frequência, o correspondente valor em Hz é calculado na célula da mesma linha na coluna seguinte.

Anexo II – Código Fonte em Linguagem C

Software para o Reconhecimento de Apito

ADC.c

```
#include "p30fxxxx.h"
#include "adc.h"

extern void porta_serie(void);
fractional valorADC[256]__attribute__ ((section (".xdata, data, xmemory"), aligned (512)));
fractional* iPtr=&valorADC[0];
unsigned int startfft;
unsigned int * adcPtr;
int cont=0;


//ADC_Init() é usado para configurar o ADC
// para converter 16 amostras num 1 canal de entrada por interrupção
//A frequência de amostragem é de 8KHz
//Timer3 é usado para definir o periodo de amostragem
//o pino de entrada é o AN5
void ADC_Init(void)
{

iPtr=&valorADC[0];
cont=0;

                //configuração de registos

                /*ADCON1*/

      ADCON1bits.FORM = 3; //formato de dados: 1.15 fractional (sddd dddd dddd 0000)
      ADCON1bits.SSRC = 2; //Usar o timer 3 para determinar o tempo de amostragem total (Ts)
      ADCON1bits.ASAM = 1; //começar amostragem automaticamente após a última conversão
estar completa


                /*ADCON2*/

                // os pinos de referência de tensão são:(AVdd e AVss)
      ADCON2bits.SMPI = 15; //16 amostras por interrupção


                /*ADCON3*/
      //pretende-se uma frequencia de amostragem (Fs) de 8KHz
      //Tempo total de amostragem Ts = 125 microsegundos (Ts = tempo de amostragem + tempo de
conversão)
      //A 29.4 MIPS, Tcy = 33.9 ns = Instruction Cycle Time
                /*configurar tempo de conversão*/
      //Tempo de conversão = 14*Tad e Tad > 667ns (para temperatura entre -40C e 125C)
      //Para configurar o Tad usa-se os bits ADCS<5:0>
          //de acordo com a seguinte formula
```

```
        //Tad = Tcy*(ADCS+1)/2
                //Para ADCS=63, Tad=1,0848 microsegundos
        //O tempo de conversão =14*Tad =15,19 microsegundos
        ADCON3bits.ADCS = 63;


                                /* Timer 3*/
        //O Timer 3 deve contar 125 microsegundos
        //Desses 125 microsegundos são 110 a fazer a amostragem + os 15 da conversão
            /*Cálculo do valor a colocar no PR3 (valor que o Timer vai contar)*/
                //PR3 =(time-out/Tcy) =(125 microsegundos/33.9 nanosegundos)=3685
                //Com este valor a verdadeira frequência de amostragem é 7998.5 Hz

        TMR3 = 0x0000;
        PR3 = SAMPLERATE;
        IFS0bits.T3IF = 0;
        IEC0bits.T3IE = 0;

                /*ADCHS*/

        ADCHS = 0x0005; //usar a entrada AN5 ligada ao MUX A

                /*ADCSSL Register*/
       //Este registo fica com os valores por defeito

                /*ADPCFG*/

        ADPCFG = 0xFFFF; //colocar todos os pinos como digitais
        ADPCFGbits.PCFG5 = 0; //colocar apenas AN5 como entrada analógica

                                /*funcionamento*/
                //Quando o Timer fizer time-out, começa a conversão
                //A interrupção é gerada quando os 16 buffers tiverem cheios

                //limpar a flag de interrpção do ADC
        IFS0bits.ADIF = 0;

        //Set the A/D interrupt enable bit
        IEC0bits.ADIE = 1;

        //Turn on ADC
        //deve fazer-se depois de configurar os outros registos
        ADCON1bits.ADON = 1;

        //Start Timer 3
        T3CONbits.TON = 1;
}

//rotina de serviço à interrupção (ISR) para a interrupção do ADC

void __attribute__((interrupt, auto_psv)) _ADCInterrupt(void)
{

        //Limpar a flag de interrupção do timer 3
        IFS0bits.T3IF = 0;

        //limpar a flag de interrupção do ADC
```

```
        IFS0bits.ADIF = 0;
int i = 0;
        //copiar o valor dos buffers para a variável "valorADC"
        adcPtr = &ADCBUF0;
                                        for (i=0;i<16;i++)
                                                {       *iPtr++ = *adcPtr++;

                                                }
 cont++;
if(cont>15)
{
        ADCON1bits.ADON = 0;

        startfft=1; // "startfft" funciona como uma flag de activação da fft,
                // quando forem adquiridos o numero de pontos pre-definidos,
                // coloca-se "startfft" a 1 para k a main chame a função fft

                        }
}
```

fft.c

```
#include <p30Fxxxx.h>
#include <dsp.h>
#include "fft.h"
#include "adc.h"
#include "parametros.h"



fractcomplex sigCmpx[FFT_BLOCK_LENGTH] __attribute__ ((section (".ydata, data, ymemory"),
aligned (FFT_BLOCK_LENGTH * 2 *2))); //o sinal de entrada para a fft tem k ser um array
complexo declarado na y-memory

extern const fractcomplex twiddleFactors[FFT_BLOCK_LENGTH/2]
__attribute__ ((space(auto_psv), aligned (FFT_BLOCK_LENGTH*2)));


int max;
int indice;

void fft(void)
{


fractcomplex *p_cmpx = &sigCmpx[0];
fractional *p_real = &valorADC[0];

        int i = 0;

        for ( i = 0; i < FFT_BLOCK_LENGTH; i++ )// A função k faz a FFT (FFTComplexIP)
necessita que os valores de entrada estejam no intervalo de valores [-0.5, +0.5], Neste ciclo faz-se
uma rotação à direita do sinal obtido do ADC, que equivale a fazer uma divisão por 2, já que o array
é do tipo fractional, onde os valores se encontravam no intervalo [-1, +1]


        {
        *p_real = *p_real >>1 ;
        *p_real++;
        }

        p_cmpx = &sigCmpx[FFT_BLOCK_LENGTH-1] ;
        p_real = &valorADC[FFT_BLOCK_LENGTH-1];

        for ( i = FFT_BLOCK_LENGTH; i > 0; i-- )  // A função FFTComplexIP necessita também
que o array de entrada seja complexo (fractcomplex)
// logo é necessário converter o array real num array complexo, para isso iguala-se a parte real ao
sinal
// real (valorADC) e iguala-se a parte imaginária a zero
        {
                (*p_cmpx).real = (*p_real--);
                (*p_cmpx--).imag = 0x0000;
        }



        /* Fazer a FFT */
```

```
        // O resultado é gravado com os bits em ordem invertida ( bit-reversed order )
        FFTComplexIP (LOG2_BLOCK_LENGTH, &sigCmpx[0], (fractcomplex *)
__builtin_psvoffset(&twiddleFactors[0]), (int) __builtin_psvpage(&twiddleFactors[0]));



        BitReverseComplex (LOG2_BLOCK_LENGTH, &sigCmpx[0]); //coloca os bits na ordem
correcta

for(i=f1;i<f2;i++)
        {
                if(sigCmpx[i].real<0)
                        {
                                sigCmpx[i].real=sigCmpx[i].real-1;      //faz o modulo de um
numero negativo, para isso subtrai-se 1 e de seguida faz-se uma NOT
                                sigCmpx[i].real=~sigCmpx[i].real;
                        }
                if(sigCmpx[i].imag<0)
                        {
                                sigCmpx[i].imag=sigCmpx[i].imag-1;   //faz o modulo de um
numero negativo, para isso subtrai-se 1 e de seguida faz-se uma NOT bit a bit
                                sigCmpx[i].imag=~sigCmpx[i].imag;
                        }
        sigCmpx[i].real=sigCmpx[i].real+sigCmpx[i].imag;

                if(max<sigCmpx[i].real)
                        {
                                max=sigCmpx[i].real;   //max guarda o valor da amplitude da
frequencia de pico entre f1 e f2
                                indice=i;                       //indice guarda o valor do
indice da frequencia de pico entre f1 e f2
                        }
        }



}
```

twiddleFactors.c

```
#include <dsp.h>
#include "fft.h"

 const fractcomplex twiddleFactors[] __attribute__ ((space(auto_psv), aligned
(FFT_BLOCK_LENGTH*2))) =
       {
        0x7FFF, 0x0000, 0x7FF6, 0xFCDC, 0x7FD9, 0xF9B8, 0x7FA7, 0xF695,
      0x7F62, 0xF374, 0x7F0A, 0xF055, 0x7E9D, 0xED38, 0x7E1E, 0xEA1E,
      0x7D8A, 0xE707, 0x7CE4, 0xE3F4, 0x7C2A, 0xE0E6, 0x7B5D, 0xDDDC,
      0x7A7D, 0xDAD8, 0x798A, 0xD7D9, 0x7884, 0xD4E1, 0x776C, 0xD1EF,
      0x7642, 0xCF04, 0x7505, 0xCC21, 0x73B6, 0xC946, 0x7255, 0xC673,
      0x70E3, 0xC3A9, 0x6F5F, 0xC0E9, 0x6DCA, 0xBE32, 0x6C24, 0xBB85,
      0x6A6E, 0xB8E3, 0x68A7, 0xB64C, 0x66CF, 0xB3C0, 0x64E8, 0xB140,
      0x62F2, 0xAECC, 0x60EC, 0xAC65, 0x5ED7, 0xAA0A, 0x5CB4, 0xA7BD,
      0x5A82, 0xA57E, 0x5843, 0xA34C, 0x55F6, 0xA129, 0x539B, 0x9F14,
      0x5134, 0x9D0E, 0x4EC0, 0x9B18, 0x4C40, 0x9931, 0x49B4, 0x9759,
      0x471D, 0x9592, 0x447B, 0x93DC, 0x41CE, 0x9236, 0x3F17, 0x90A1,
      0x3C57, 0x8F1D, 0x398D, 0x8DAB, 0x36BA, 0x8C4A, 0x33DF, 0x8AFB,
      0x30FC, 0x89BE, 0x2E11, 0x8894, 0x2B1F, 0x877C, 0x2827, 0x8676,
      0x2528, 0x8583, 0x2224, 0x84A3, 0x1F1A, 0x83D6, 0x1C0B, 0x831C,
      0x18F9, 0x8276, 0x15E2, 0x81E3, 0x12C8, 0x8163, 0x0FAB, 0x80F7,
      0x0C8C, 0x809E, 0x096B, 0x8059, 0x0648, 0x8028, 0x0324, 0x800A,
      0x0000, 0x8000, 0xFCDC, 0x800A, 0xF9B8, 0x8028, 0xF695, 0x8059,
      0xF374, 0x809E, 0xF055, 0x80F7, 0xED38, 0x8163, 0xEA1E, 0x81E3,
      0xE707, 0x8276, 0xE3F5, 0x831C, 0xE0E6, 0x83D6, 0xDDDC, 0x84A3,
      0xDAD8, 0x8583, 0xD7D9, 0x8676, 0xD4E1, 0x877C, 0xD1EF, 0x8894,
      0xCF04, 0x89BE, 0xCC21, 0x8AFB, 0xC946, 0x8C4A, 0xC673, 0x8DAB,
      0xC3A9, 0x8F1D, 0xC0E9, 0x90A1, 0xBE32, 0x9236, 0xBB85, 0x93DC,
      0xB8E3, 0x9593, 0xB64C, 0x975A, 0xB3C0, 0x9931, 0xB140, 0x9B18,
      0xAECC, 0x9D0E, 0xAC65, 0x9F14, 0xAA0A, 0xA129, 0xA7BD, 0xA34C,
      0xA57E, 0xA57E, 0xA34C, 0xA7BD, 0xA129, 0xAA0A, 0x9F14, 0xAC65,
      0x9D0E, 0xAECC, 0x9B18, 0xB140, 0x9931, 0xB3C0, 0x975A, 0xB64C,
      0x9593, 0xB8E3, 0x93DC, 0xBB85, 0x9236, 0xBE32, 0x90A1, 0xC0E9,
      0x8F1D, 0xC3A9, 0x8DAB, 0xC673, 0x8C4A, 0xC946, 0x8AFB, 0xCC21,
      0x89BF, 0xCF04, 0x8894, 0xD1EF, 0x877C, 0xD4E1, 0x8676, 0xD7D9,
      0x8583, 0xDAD8, 0x84A3, 0xDDDC, 0x83D6, 0xE0E6, 0x831C, 0xE3F5,
      0x8276, 0xE707, 0x81E3, 0xEA1E, 0x8163, 0xED38, 0x80F7, 0xF055,
      0x809E, 0xF374, 0x8059, 0xF695, 0x8028, 0xF9B8, 0x800A, 0xFCDC
       } ;
```

main.c

```
/* Reconhecimento de Apito */

#include <p30Fxxxx.h>
#include <dsp.h>
#include "fft.h"
#include "adc.h"
#include "parametros.h"

//Configuração de macros

  //usa-se um cristal externo, usando o modo PLL 16x, para um oscilador de de 7.37 MHz
//a frequência a que corre o projecto é 7.3728e+6*16/4 = 29.49 MIPS (Fcy) -- Tcy = 33.9
nanosegundos

_FWDT(WDT_OFF);                 //Watch-Dog desligado
_FBORPOR(MCLR_EN & PWRT_OFF);   //Enable MCLR reset pin and turn off the
                        //power-up timers.
_FGS(CODE_PROT_OFF);            //Disable Code Protection




int main()
{
int contador=1;
TRISBbits.TRISB3 = 0;

while(1)
{

startfft=0;


ADC_Init();


while(startfft==0);
max=0;
fft(); //chamar a função fft quando "startfft" estiver a 1, ou seja, quando acabar a aquisição



        if (max>A && contador==R)
                {
                                        LATBbits.LATB3 = 1;             //liga o LED
                }
        else
                {
                        if(max>A && contador<R) contador++;
                        else
                                        {
                                                LATBbits.LATB3 = 0;  //desliga o LED
                                                contador=1;
                                        }
                }
```

```
}
return 0;

}
```

ADC.h

```
#include <dsp.h>
#include "fft.h"
#include "parametros.h"


extern fractional valorADC[256];
extern fractional* iPtr;
extern unsigned int startfft;


extern void ADC_Init(void);
extern void __attribute__((__interrupt__)) _ADCInterrupt(void);
```

fft.h

```
#include<dsp.h>
/* Definições de contanstes */
#define FFT_BLOCK_LENGTH        256   // número de pontos da FFT
#define LOG2_BLOCK_LENGTH       8         // logaritmo de base 2 do num de pontos

extern int peakFrequencyBin;

extern fractcomplex sigCmpx[FFT_BLOCK_LENGTH] __attribute__ ((section (".ydata, data,
ymemory"), aligned (FFT_BLOCK_LENGTH * 2 *2)));
extern int peakFrequencyBin;
extern void fft(void);

extern int max;
extern int indice;
```

parametros.h

```
/*Este finheiro inclui os parametros ajustaveis para melhorar o reconhecimento do apito,
 em função do ambiente e do próprio apito*/


#define f1 75    /*frequencia inferior no reconhecimento
                                f1 > 0*/

#define f2 85    /*frequencia superior no reconhecimento
                                f2<128* e f2>f1*/



#define A 900   //amplitude mínima no reconhecimento A<2^13 (8192),
                                //este valor deve ser aumentado quanto maior ruidoso for o ambiente

#define SAMPLERATE 3685 /*Valor a colocar no registo do timer para definir a
                                                        //frequência de amostragem
                                                        //SAMPLERATE=(Ts/Tcy)
                                                        //Tcy=33.9 nanosegundos
                                                        //Ts=1/Fs=*/
#define R 5 //este parametro define o numero de iterações necessárias seguidas para que se considere
apito,
                        //deve ser aumentado quanto maior for ruidoso for o ambiente, quanto maior
for, maior será o tempo de resposta do sistema
/* O numero de amostras é de 256 - valor fixo*/
```

Software para Enviar o Espectro

NOTA: os ficheiros ADC.c, fft.c, twiddleFactors.c, ADC.h e fft.h são os mesmos utilizados no Software para o Reconhecimento de Apito.

porta_serie.c

```
#include"fft.h"
#include"adc.h"
#include<p30f4013.h>
#include<uart.h>



void __attribute__((__interrupt__)) _U1TXInterrupt(void)
{
IFS0bits.U1TXIF = 0;
}

void porta_serie(void)
{

char s[9];

char sk[20];
int k;

TRISFbits.TRISF2=1;
TRISFbits.TRISF3=0;




unsigned int baudvalue;

unsigned int U1MODEvalue;

unsigned int U1STAvalue;

CloseUART1();

/* Configuração das interrupções */
ConfigIntUART1(UART_RX_INT_DIS & UART_RX_INT_PR6 &
UART_TX_INT_EN & UART_TX_INT_PR0);



/* Configuração do modulo UART1 com 8 bits de dados, 1 stop bit e sem bit de paridade.*/

baudvalue = 192;
U1MODEvalue = UART_EN & UART_IDLE_CON & UART_ALTRX_ALTTX &
UART_DIS_WAKE & UART_DIS_LOOPBACK &
UART_EN_ABAUD & UART_NO_PAR_8BIT &
UART_1STOPBIT;
U1STAvalue = UART_INT_TX_BUF_EMPTY &
```

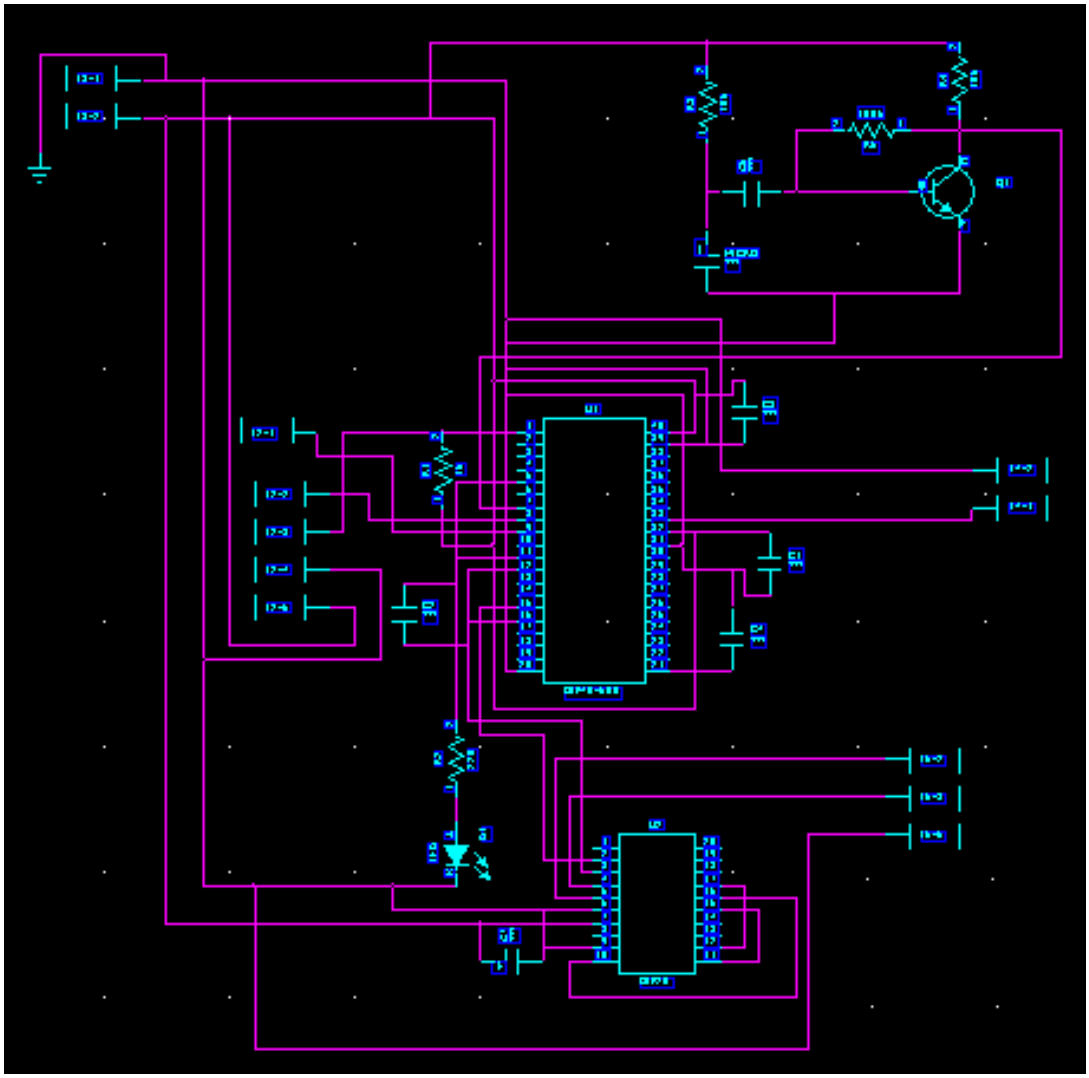
```
UART_TX_PIN_NORMAL &
UART_TX_ENABLE & UART_INT_RX_3_4_FUL &
UART_ADR_DETECT_DIS &
UART_RX_OVERRUN_CLEAR;
OpenUART1(U1MODEvalue, U1STAvalue, baudvalue);


for(k=1;k<127;k++)
{
//sprintf(sk,"  i%d-->",k); //enviar o indice
//putsUART1(sk);

sprintf(s," %d\r\n",sigCmpx[k].real);      //enviar o espectro (k=1;k<127)

//sprintf(s," %d\r\n",valorADC[k]); //enviar o valor do ADC (k=0;k<256)
putsUART1((unsigned int *)s);

}

}
```

main.c

```
/*Teste para reconhecimento de apito
Este programa faz a aquisição de um sinal sonoro, cálcula o espectro e envia
para a porta série o resultado, ou também se pode optar por enviar o resultado da aquisição*/

#include <p30Fxxxx.h>
#include <dsp.h>
#include "fft.h"
#include "adc.h"

//Configuração de macros


_FWDT(WDT_OFF);                 //Watch-Dog desligado
_FBORPOR(MCLR_EN & PWRT_OFF);   //Enable MCLR reset pin and turn off the
                        //power-up timers.
_FGS(CODE_PROT_OFF);            //Disable Code Protection



extern void porta_serie(void);

int main()
{
startfft=0;



ADC_Init();

            while(1){
if (startfft==1)
{

fft(); //chamar a função fft quando "startfft" estiver a 1, ou seja, quando acabar a aquisição
porta_serie(); //chama a função porta_serie
startfft=0;

}

}



return 0;
}
```

Anexo III – Esquemático e Desenho da Placa PCB

Esquemático



Desenho da Placa PCB