

Relatório 1  
Projeto em Eletrônica I - EEL7801

Universidade Federal de Santa Catarina - UFSC  
Professora: Daniela Ota Hisayasu Suzuki

Luiz Augusto Frazatto Fernandes: *17202752*  
Leonardo José Held: *17203984*

02 de Maio de 2019

Nota: O projeto todo, incluindo este documento e os demais códigos de simulação e de projeto podem ser encontrados em <https://github.com/leonheld/EEL7801>

# Chapter 1

## Metodologia

### 1.1 Modulação e demodulação de sinais

#### 1.1.1 Motivação da escolha do algoritmo

Escolheu-se o processo de modulação por Chaveamento de Deslocamento de Frequência (FSK, em inglês). São usadas duas frequências ótimas para se representar 0 e 1, e ambas são obtidas experimentalmente: a partir de testes realizados com o transdutor (microfone), é gerada uma curva normal, em que a resposta desse ao sinal recebido é ótima para uma frequência específica (frequência da onda portadora). São, então, obtidas duas outras frequências equidistantes do centro da curva gaussiana, e a cada uma é associado um valor binário.

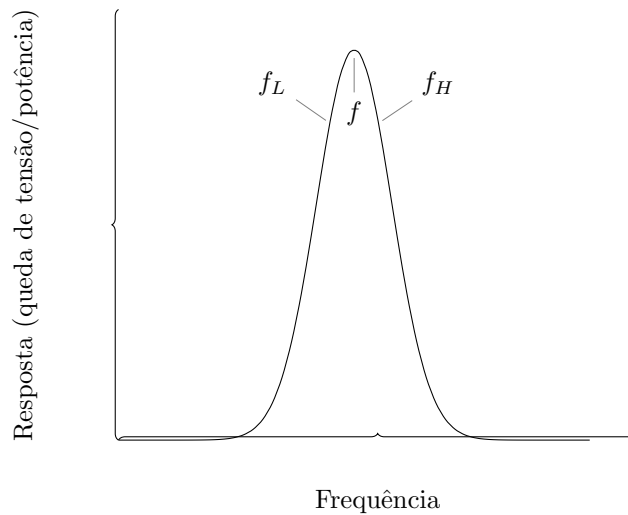
O processo de modulação, em si, consiste na transformação de um sinal PWM em um analógico (que é controlado por um STM32) que, por sua vez, é emitido por um tweeter. O sinal (sonoro) é recebido por um microfone controlado por outro STM32, que realizará o processo de demodulação do sinal.

#### 1.1.2 Algoritmo da modulação (FSK)

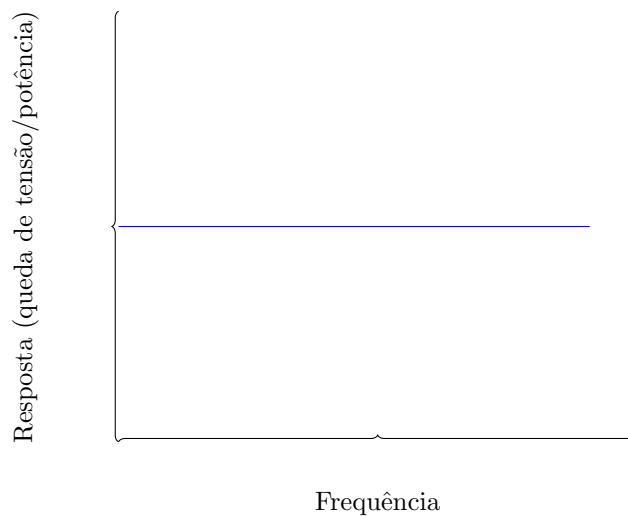
Empiricamente são testadas (com o microfone) diferentes frequências emitidas pelo tweeter e recebidas pelo microfone. A de melhor resposta (dada, no gráfico abaixo, por  $f$ ) é associada à onda portadora (carrier). É, então, estabelecido um desvio  $\Delta f$ , e, a partir desse, são determinadas:

$$\begin{aligned}f_L &= f - \Delta f \\f_H &= f + \Delta f\end{aligned}$$

de tal forma que à  $f_H$  se associa o bit 1, e à  $f_L$ , 0.



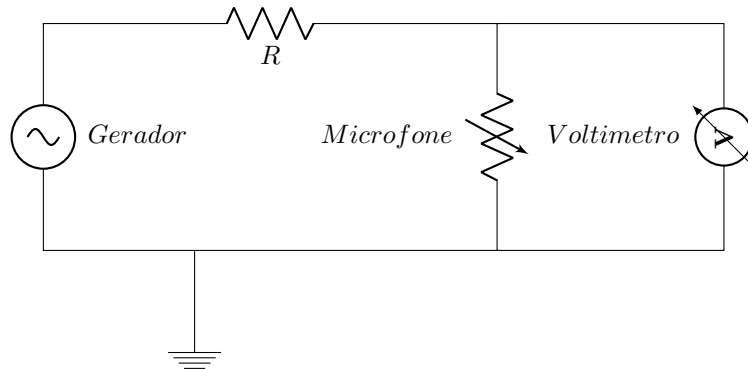
Há, também, uma segunda possibilidade: de o microfone não reagir significativamente a diferentes frequências, mas de ainda assim conseguir diferenciá-las. Isto é: o microfone identifica quando duas frequências recebidas são diferentes, mas sua curva de queda de tensão se aproxima de uma constante, como no exemplo a seguir:



Caso esse comportamento seja verificado, o que há de se fazer é, simplesmente, escolher duas frequências quaisquer para se trabalhar no algoritmo.

### 1.1.3 Obtenção das frequências ótimas

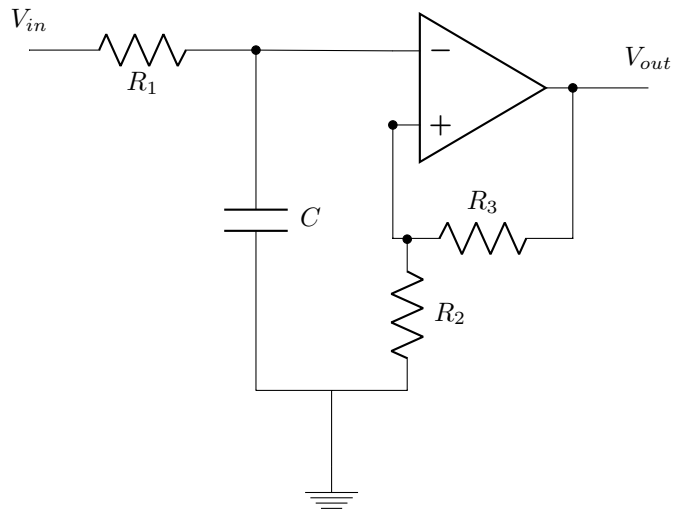
Por meio do circuito da (inserir número da imagem aqui) é medida a tensão/potência dissipada nos terminais do microfone. A curva (próxima à normal descrita acima) é gerada a partir da potência consumida/queda de tensão que é medida entre os terminais do microfone.



Quanto maior a queda de tensão no microfone, maior a energia por ele consumida em relação ao resto do circuito (divisor de tensão), logo, melhor será sua resposta aos estímulos sonoros testados.

#### 1.1.4 Tratamento do sinal analógico

Será utilizado, a priori, um filtro passa-baixa a fim de reduzir o ruído de alta frequência, além de um amplificador não inversor (que garantirá uma clara análise do sinal).



A frequência de corte do circuito acima é dada por

$$f_c = \frac{1}{2 \cdot \pi \cdot R_1 \cdot C} \quad (1.1)$$

E o ganho por

$$\frac{V_{out}}{V_{in}} = G = 1 + \frac{R_3}{R_2} \quad (1.2)$$

Espera-se trabalhar com frequências na faixa audível ( $20Hz$  a  $20kHz$ ). Portanto, pode-se colocar uma frequência de corte razoável de  $22\text{ kHz}$  a priori, já

que, dessa forma, o ruído de alta frequência filtrado não interferirá nos sinais desejados.

Para tal, usamos um capacitor com capacitância  $C = 0.1\mu F$ , o que nos resulta:

$$R_1 = \frac{1}{2 \cdot \pi \cdot 0.1 \cdot 10^{-6} \cdot 22 \cdot 10^3} \approx 72\Omega \quad (1.3)$$

Para avaliarmos o ganho, deixamos em  $R_3$  um potenciômetro e, assim, avaliamos, a posteriori, qual o melhor ganho para se projetar (aquele cuja análise posterior for facilitada).

## 1.2 Escolha de Microcontrolador ( $\mu C$ )

### 1.2.1 STM32

O microcontrolador foi escolhido na base da seguinte *criteria*:

- 1) Baixo Custo: O controlador deve ter um custo mínimo, com alta flexibilidade de prototipação. Deve incluir um programador integrado, ou possuir algum de baixo custo e facilmente acessível.
- 2) Alto suporte: O controlador deve ter suporte de compiladores, ambientes de programação, comunidade de software aberto e pela própria empresa que o fabrica.
- 3) Performance: A unidade deve ter uma performance que faz processamento de dados e sinais com relativa rapidez, de forma que a sua capacidade de processamento não dite o algoritmo e execução do programa.

Em consideração aos itens **1** e **2**, se optou por utilizar algum controlador da família ARM<sup>TM</sup>, indubitavelmente a família mais utilizada de processadores no mundo, e desta forma, acessível e de bom suporte.

O quesito performance nos faz olhar para as famílias de processadores que a ARM<sup>TM</sup> atualmente oferece, dentre as quais, os processadores Cortex-M3<sup>TM</sup> <sup>1</sup> parecem entrar nos quesitos.

De acordo com o página da empresa sobre o processador:

“The Arm Cortex-M3 processor is the industry-leading 32-bit processor for highly deterministic real-time applications.”

Depois de uma curta pesquisa de mercado, a placa STM32F103C8 da ST<sup>TM</sup> parece preencher bem os quesitos supraescritos, com um processador Cortex-M3<sup>TM</sup> de 72MHz de clock interno, é um dos microcontroladores mais potentes da categoria fabricados pela ST<sup>TM</sup>, o que preenche bem o requisito **3**.

Qualquer processador dessa linha da ST<sup>TM</sup> pode ser programado e debugado via uma interface chamada ST-Link<sup>TM</sup>, que também foi de fácil obtenção e

---

<sup>1</sup><https://developer.arm.com/ip-products/processors/cortex-m/cortex-m3>

baixo custo, e basicamente inclui um outro processador que conversa em tempo real com a placa alvo.

### 1.2.2 Suporte

O suporte oferecido tanto pela ARM<sup>TM</sup> quanto pela ST<sup>TM</sup> é extenso. A ST<sup>TM</sup> produz seu próprio software de pré-configuração e geração de código chamado STM32CubeMX<sup>TM</sup>, que é uma *Graphical User Interface* usada para gerar o código inicial, como configuração de clock interno e dos pinos do processador, por exemplo.

A ARM<sup>TM</sup> também mantém um compilador, um debugger e utilidades de programação para sua linha de processadores, todas as ferramentas baseadas no GCC, GDB e Gnu Binutils.<sup>2</sup>

Como software de upload de código e debugging, podemos usar o programa *openocd*<sup>3</sup>, também de código aberto que acessa a interface SWD (*Serial Wire Debug*) do ST-Link<sup>TM</sup> via terminal, com possibilidade de abrir uma janela de sessão de debugging via GDB.

### 1.2.3 Processo de Desenvolvimento

Desde a última versão, o STM32CubeMX<sup>TM</sup> consegue produzir um Makefile que pode ser compilado pelo programa *make* chamando o compilador de C cedido pela ARM.

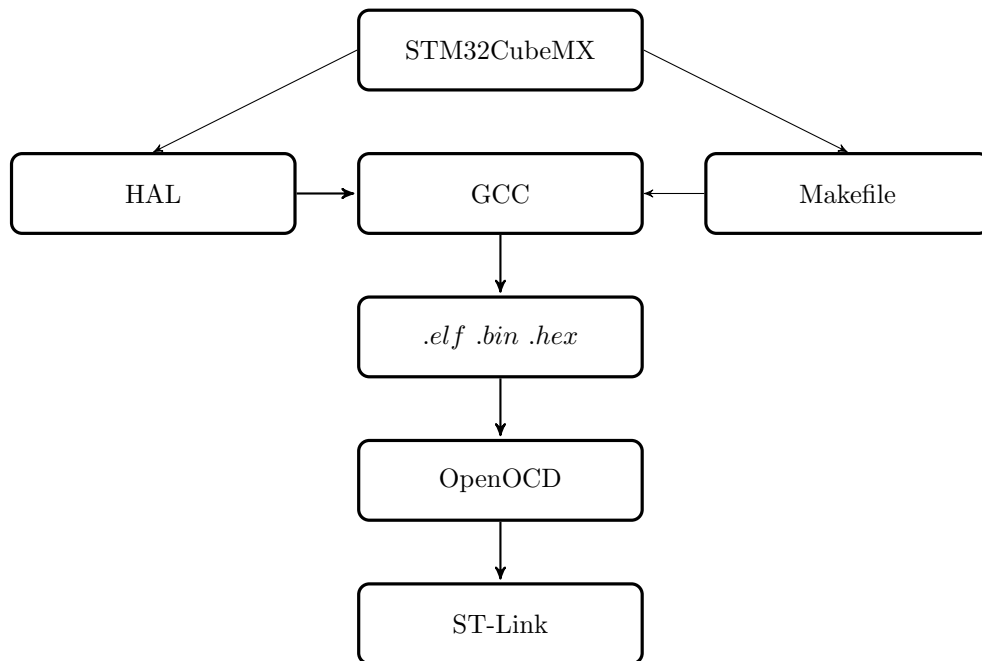
A partir disso, o compilador gera arquivos .hex, .elf e .bin, quaisquer podendo ser inseridos na memória interna do microcontrolador pelo openocd.

Para a programação, existe uma HAL - *Hardware Abstraction Layer* -, também fornecida pela ST<sup>TM</sup>, que é incluída nos sources do projeto no momento da inicialização do código pelo STM32CubeMX<sup>TM</sup> e possui funções específicas de baixo nível, mas tem as mesmas chamadas em quaisquer processador alvo que queira executar o código, aumentando a portabilidade. Também, a HAL é construída de forma a aumentar o nível de abstração, diminuindo significativamente a complexidade do código, permitindo desenvolvimento rápido e colaborativo, sem necessariamente conhecer todos os artifícios de baixo nível de cada processador.

---

<sup>2</sup><https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain>

<sup>3</sup><http://openocd.org/>





## Chapter 2

# Title

### 2.1 Title

#### 2.1.1 title