Universidade de Aveiro

**Information and Coding**

**Lab Work nº1**

****

Diogo Martins (108548) | XXXXXX (XXXXXX) | XXXXXX (XXXXXX)

Departamento de Eletrónica, Telecomunicações e Informática

19/10/2025

Index

[Summary 3](#_Toc210584568)

[Part 1 4](#_Toc210584569)

[Exercise 1 4](#_Toc210584570)

[Exercise 2 4](#_Toc210584571)

[Exercise 3 4](#_Toc210584572)

[Exercise 4 4](#_Toc210584573)

[Part 2 5](#_Toc210584574)

[Exercise 5 5](#_Toc210584575)

[Exercise 6 5](#_Toc210584576)

[Part 3 5](#_Toc210584577)

[Exercise 7 5](#_Toc210584578)

# Summary

The project consists of multiple C++ files with the purpose of exploring sound file’s properties and transforming them. The original repository came with a few pre-made programs such as **wav\_cp** which creates a copy of a given sound file, **wav\_hist** that outputs a basic histogram of the sound file and a **wav\_dct** which represents an example of use of the Discrete Cosine Transform (DCT). On top of that, some example sound files, libraries and makefiles were added to help development.

\*\*\*\*\*\*UPDATE FOR PART 2 AND 3\*\*\*\*\*\*

# Part 1

## Exercise 1

For this exercise, the idea is to modify **wav\_hist.h** and **wav\_hist.cpp** so that it can also output the histogram of the average of the channels (channel MID) and the average difference of the channels (channel SIDE).

To output channel MID, update() in **wav\_hist.h**was updated to calculate the expression (L+R)/2, which represents the addition between channels divided by two, returning the average of the channels. The same was done for channel SIDE with the expression (L-R)/2 returning the average difference. The necessary dump functions for each new channel were also added similar to the functions of channels 0 and 1. The file **wav\_hist.cpp** was updated to detect the new channels and to properly dump them.

Additionally, the histogram bins meant for each different sample were replaced with coarser bins, that gather together 2, 4, 8, . . . 2K values.

The output is as follows:

* input
* output

## Exercise 2

For this exercise, we must implement a program named **wav\_quant.cpp** to reduce the number of bits used to represent a given audio sample, in other words, to perform uniform scalar quantization.

The output is as follows:

* input
* output

## Exercise 3

For this exercise, we must implement a program named **wav\_quant.cpp** to reduce the number of bits used to represent a given audio sample, in other words, to perform uniform scalar quantization.

The output is as follows:

* input
* output

## Exercise 4

For this exercise, we must implement a program named **wav\_quant.cpp** to reduce the number of bits used to represent a given audio sample, in other words, to perform uniform scalar quantization.

The output is as follows:

* input
* output

# Part 2

## Exercise 5

For this exercise, the idea is to modify **wav\_hist.h** and **wav\_hist.cpp** so that it can also output the histogram of the average of the channels (channel MID) and the average difference of the channels (channel SIDE).

The output is as follows:

* input
* output

## Exercise 6

For this exercise, the idea is to modify **wav\_hist.h** and **wav\_hist.cpp** so that it can also output the histogram of the average of the channels (channel MID) and the average difference of the channels (channel SIDE).

The output is as follows:

* input
* output

# Part 3

## Exercise 7

For this exercise, the idea is to modify **wav\_hist.h** and **wav\_hist.cpp** so that it can also output the histogram of the average of the channels (channel MID) and the average difference of the channels (channel SIDE).

The output is as follows:

* input
* output