Audio effects library
Requirements specification

Kacper Harezga, Ewa Kobiela, Jan Laskowski
Krzysztof Sobczyk, Grzegorz Machura

10.04.2021

Contents

1 Project constraints

1.1 Project requirements document

Business Case

- We need to prepare a team project connected with our field of studies in order to pass the course Team and Preengineering project taught by Mgr. Inż Jakub Klikowski
- The idea of the project is provided by the Dolby Company with which we work on this project
- It will equip us and help expand the knowledge about digital signal processing and programming skills

- The product may replace manual coding repeated for each project and create ready-to-implement solution
- The project is requested by Dolby company, thus we will implement our solutions to the base program provided by the company
- We will use and expand our knowledge acquired in the university and private projects in the field of programming in Python and DSP

Project Objectives

- Create the audio effects library which allows the user to apply a selected effect at the input audio file and generate it as an output.
- It needs to be able to read input file and create an output file
- It needs to be intuitive for the users
- It will be implemented in Python language, version 3.6
- The management of the project will be performed with the use of Kanban board

- The project will be expanding an exampolary solution provided by Dolby - it will implement audio effect processing to the existing basic application
- Implemented effects will include time-domain processing (delay, FIR filters etc.) and frequency-domain processing (FIR filter, band equalizer etc.). It is expected to provide only a few filters of a type.
- The base application will be connected with audio effects functions in order to create complete signal path.
- No database will be created during this project. Input and output files will be read from and saved to predefined directory on local drive.

Major Deliverables

- Implementation of the ability to add time-domain effects
  - Functions for each of chosen filters (delay, FIR etc.)
- Implementation of the ability to add frequency-domain effects
  - Functions for each of chosen filters (FIR, band equalizer etc.)
- Complete signal path for audio processing
  - Functionality of reading input file and saving the output file
  - Calling the appropriate function to add audio effect chosen by the user to the input file and return it as an output file
  - Adding user interface of the program
- Program tested and potential bugs eliminated
  - Tests defined
  - Tests executed
  - Potential bugs and crashes solved

Roles & Responsibilities

- Project manager - Kacper Harężga
  - responsible for contacting main stakeholders (Dolby and Mgr. Inż. Jakub Klikowski) to discuss details of the project and ask for feedback
- The management team - Kacper Harężga and Ewa Kobiela
  - responsible for dividing tasks and creating documentation of this project
- The project team consists of five students (Kacper Harężga, Ewa Kobiela, Jan Laskowski, Krzysztof Sobczyk and Grzegorz Machura) who will implement the solution and test created versions of the project
  - Each member of project team will be responsible for creating a filter implementing a specific audio effect
  - Members will share the results with the team and stakeholders
  - In the second part of implementation the solution members will create the full signal path and test the application

Stakeholders

- The first primary stakeholder is Mgr. Inż. Jakub Klikowski, who runs the course Team and Preengineering Project and grades the project at the end of the semester
- The second primary stakeholder is Dolby Company with the representative of Paweł Jaroch, which requested the project at the Team Project Conference
- Secondary stakeholders include students in our group of Team and Preengineering Project course, who will be concerned about the results

Assumptions	• Project should be finished by the end of the semester, including development process, testing stage and documentation writing • We do not expect to bear any money cost

1.2 Mandated constraints

It is anticipated to create a single version of product which will contain all functionalities described in this document. Additionally, the development process will be divided in two increments, after which Dolby Laboratories Inc. will have opportunity to provide feedback and point possible improvement of the project. First increment will end once time domain filters will be implemented. The scope of second increment will expand the functionality of the application with frequency domain processing filters and simple interface to test or present possibilities of the system.

Programming language used in the project was chosen by the stakeholder - which is Dolby Laboratories Inc. - to be Python. Due to the need to use specific libraries for this language we agreed on version 3.6 of Python. Tools used in development process was chosen to consist of PyCharm by JetBrains as IDE and for GIT for version control.

The product shall use a simple pipeline, which will process an input sound file into an output sound file, adding a chosen effect in the process. The system will not work on real-time mode, thus the input file must be prepared by the client ahead. The library will be used by the client in post-production of sound, music or movies, thus additional real-time frequency will not be required. All files generated as output signals must be audible, saved in the sound file type. We do not assume any reverting functionality, i. e. removing effect from the output signal. It is important to allow user to choose the appropriate effect(s) for their application, thus the choosing dialogue must be clear and user-friendly.

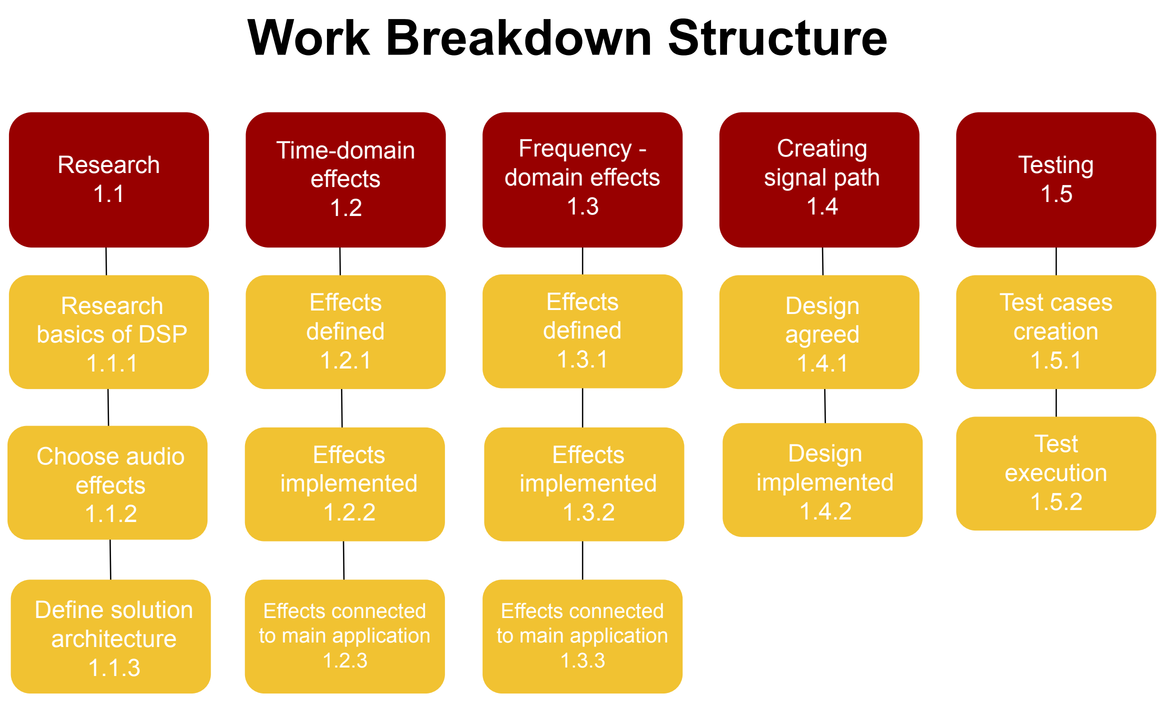
The designed system might be a part of a bigger system. To achieve that effect, the external application must support file reading or/writing. The input files must be placed in a predefined directory required by Audio Effect Library's pipeline and the output files will be saved to another predefined directory, thus the external application must be able to read and/or write files with sound types of extensions to/from this directory.

The Audio Effect Library system could be used on personal computers in the form of scripts using and IDE. In this form the application would be stand-alone, however it would require from the potential user a set of IT skills and installing additional software such as IDE to run the application. The second way to implement the system is to use it on a microcontroller as a part of a bigger sound system.

Scheduled constrains of this project is set to the end of a summer semester 2020-2021, i. e. 22.06.2021, which is a critical data of finishing all the work. In case of any unexpected adversities that would make it impossible to finish all planned functionalities by this date, the project would be released in the current state, as the deadline is final. Such a deadline is required by the university and the need for grading the project and finishing the course. However, the plan assumes finishing the process of development by the end of May to reserve time for testing procedures and applying feedback or performance investigation of the system.

Budget of this project do not assume any expenses that the project team would take, besides energy consumed by the devices and maintenance costs of them. We intend to use only the software with open licences, such as GIT, or provided by the university, such as PyCharm IDE.

1.3 Work breakdown structure


Work Breakdown Structure
Research 1.1
Research basics of DSP 1.1.1
Choose audio effects 1.1.2
Define solution architecture 1.1.3
Time-domain effects 1.2
Effects defined 1.2.1
Effects implemented 1.2.2
Effects connected to main application 1.2.3
Frequency - domain effects 1.3
Effects defined 1.3.1
Effects implemented 1.3.2
Effects connected to main application 1.3.3
Creating signal path 1.4
Design agreed 1.4.1
Design implemented 1.4.2
Testing 1.5
Test cases creation 1.5.1
Test execution 1.5.2

1.4 Naming conventions and terminology

In order to avoid misunderstanding of scope of each work package from provided work breakdown structure, we present below the dictionary of the convention and terminology.

Research basics of DSP 1.1.1	● Familiarizing with chosen aspects of the given literature: ○ "Guide to DSP" by Steven W. Smith; "Think DSP" by Allen B. Downey ● Getting basic knowledge of digital signal processing principle and discuss the ways of implementing them in Python code
Choose audio effects 1.1.2	● Decide on which and how many filters will be implemented, both time- and frequency-domains ● List them and divide among group members
Define solution architecture 1.1.3	● Decide the software architecture of the solution ● Define file and class structure ● Create version control repository and common starting point
Effects defined 1.2.1 & 1.3.1	● Name audio effects implemented at this stage ● Define its principle of operation, required input (if special preprocessing is needed) and expected output
Effects implemented 1.2.2 & 1.3.2	● Create class for each audio effect ● Define methods of processing the input signal to the output results ● Refactor code to meet common architecture agreement
Effects connected to main application 1.2.3 & 1.3.3	● Create instance of a class representing each audio effect ● Provide the possibility to call the class processing methods from main function of the application
Design agreed 1.4.1	● The form of user interface is chosen as the most efficient one for the current stage of work, schedule realization and other significant factors ● Design the user interface allowing user to choose an audio effect out of implemented ones
Design implemented 1.4.2	● Implemented the chosen solution of user interface
Test cases creation 1.5.1	● Choose the areas that needs to be tested before closing the project ● Create test cases for the project functionality
Test execution 1.5.2	● Execute created test cases ● List potentially found bugs and prioritize them ● Solve the relevant bugs and repeat test execution proving completeness of the project

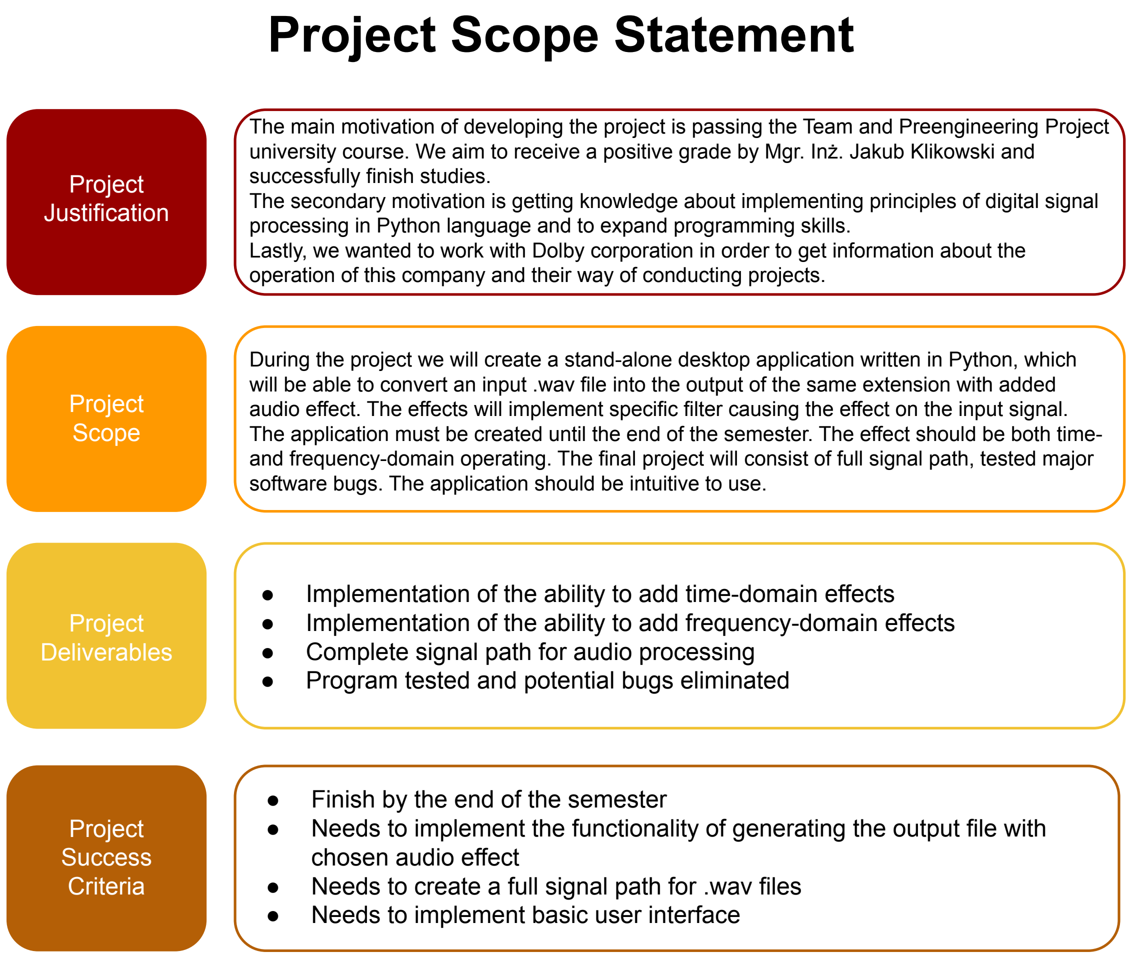
1.5 Relevant facts and assumptions

An exemplary pipeline is provided by Dolby Laboratories Inc. That consists of a set of scripts processing an input signal by reducing each audio frame by a half. The team's task is to extend this simple application to allow wider range of audio processing effects. The assumption is not to write the main function from scratch, but to extend the one provided by Dolby company. We will build on basis on that and expand the project with feature files.

Another relevant assumption is that no money cost is predicted. The software used in this project will contain free licence programs (such as Git version control system) or with licence provided by university (such as PyCharm IDE by JetBrains).

Moreover, the non-exceeding deadline for finishing the project is 13.06.2021 which is the end of current university semester.

2 Functional Requirements



Project Scope Statement

Project Justification	The main motivation of developing the project is passing the Team and Preengineering Project university course. We aim to receive a positive grade by Mgr. Inż. Jakub Klikowski and successfully finish studies. The secondary motivation is getting knowledge about implementing principles of digital signal processing in Python language and to expand programming skills. Lastly, we wanted to work with Dolby corporation in order to get information about the operation of this company and their way of conducting projects.
Project Scope	During the project we will create a stand-alone desktop application written in Python, which will be able to convert an input .wav file into the output of the same extension with added audio effect. The effects will implement specific filter causing the effect on the input signal. The application must be created until the end of the semester. The effect should be both time- and frequency-domain operating. The final project will consist of full signal path, tested major software bugs. The application should be intuitive to use.
Project Deliverables	• Implementation of the ability to add time-domain effects • Implementation of the ability to add frequency-domain effects • Complete signal path for audio processing • Program tested and potential bugs eliminated
Project Success Criteria	• Finish by the end of the semester • Needs to implement the functionality of generating the output file with chosen audio effect • Needs to create a full signal path for .wav files • Needs to implement basic user interface

2.1 Scope of the work

The workload in this project consists mainly of creating the product with constraints agreed with the stakeholders. The product should be fully functional with complete sognal path, ability to read/write files and add an audio effect chosen by the user to the main signal. The application must be fully tested and potentially relevant bugs should be eleiminated before finishing the project.

Moveover, the project scope will include achieveing the ecceptance of the project by the stakeholders, both Dolby and University side. After receiving feedback for each step we will implement the comments and improvements. Moreover, after the stage of testing all potential bugs should be closed.

Besides implementing the technical solutions, the docummentation of this project will be written. It will consist several stages and feedback to them will be included in the final form of project docummentation.

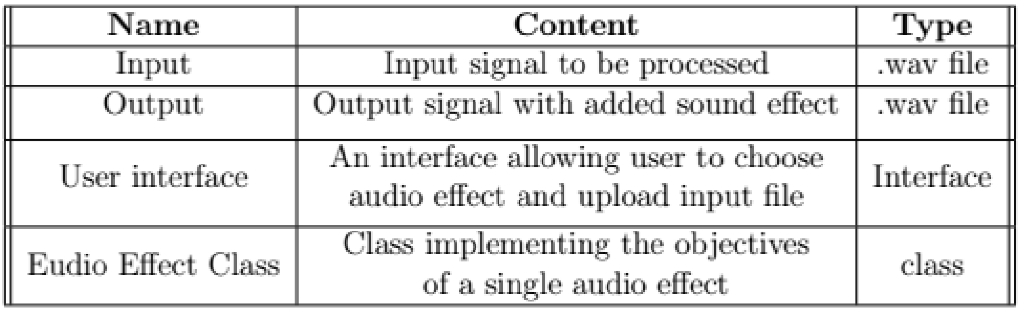
Finally, we will collect knowledge gained in this project. All personal experiences will be collected and discussed, thus strong and weak points of technical solution, project management and any other concerned field will be known and we will sum up the benefits of this project duration.

2.2 Business data model

The Audio Effects Library is not be using any additional database. The input and output files are stored in project directories and can be modified directly in the file explorer. The application downloads file with a given name from the folder ”input” and after processing the signal it writes a file containing new data into the folder ”output”, using the same filename as of the input file. This allows user to easily recognize the processed data in the directory. In case of processing the same input file for the second time, the output file is overwrite with the new type of processing. The input signal may be either stereo (for example music) or mono (for example speech) type of signal.

2.3 Data dictionary

Dictionary of data names used in the project is presented in the table below.



Name	Content	Type
Input	Input signal to be processed	.wav file
Output	Output signal with added sound effect	.wav file
User interface	An interface allowing user to choose audio effect and upload input file	Interface
Eudio Effect Class	Class implementing the objectives of a single audio effect	class

2.4 Scope of the product

The product consists of a stand-alone desktop application implementing a functionality of adding chosen audio effect to the input .wav file. The application will generate an output .wav file being a result of addition of the input signal and audio effect. The set of possible effects is finite and will be displeyed to the user. The list will contain only the effect being in the scope of this project.

The deliverable consist of Python scripts bein a prototype of the application and specific place noted for transfering input files and receiving output files. There are no database implemented. The packet handled to stakeholders will contain also docummentation of the project, description of the prototype including short manual for instalation and usage.

2.5 Functional requirements

The main functionality is defined as converting input files into outputs with added chosen audio effect. The success criteria of this functionality is that the full signal path is created and user is allowed to choose an effect from the list of implemented ones. User will know where to put the input file and where to find the output file. Both of them should be in .wav format.

3 Non-functional Requirements

3.1 Look and feel requirements

The product will not implement any physical features, thus feel requirements does not apply.

Feel requirements are defined as a simple user interface application. User should get the information where to place the input file, which format is supported, where to find the output file. The user should be able to recognize the place to choose an effect to be implemented and be able to use it. The apperance requirements such as colours or brandings are not defined and may be choosen freely.

3.2 Usability and humanity requirements

The user should be able to use this product quiclky when installed on the computer. No preparation should be required for the user, if they come with a .wav file with signal to be processed. The casual user should only remember to have the correct format of the file, as the product will not support any other file types.

The application should generate a minimum amount of errors and the output files should be possible to be recognized which effect was used when creating the specific file. The product can help the user to avoid making mistakes when implementing the audio effect independently. The effects avaliable in the product will be strictly defined and missunderstandings of the process of digital signal processing by the user will be decreased. The product shall be used by people with basic understand of English and computer skills, with no advanced knowledge about digital signal processing on electroacoustics.

3.3 Performance requirements

The speed of operation of the product shall be less than few minutes. The response of the system to the user decision to process the input signal shall be immidiate. However, the time of processing the signal, aspecially large files is allowed to take up to seconds or minutes. We set a treshold for the system to generate the output file to maximum of five minutes for lagre files for 90 percent of the interrogations. The product shall be avaliable to operate at any time, after the previous processing has ended.

3.4 Operational and environmental requirements

The product shall work on the last releases of Windows operating systems. We do not support any other systems than Windows at this point. We will test the functionality of the product on Windows 10.

Our product shall be distributed as a ZIP file. The installation and operating intruction shall be delivered with the product files to allow user safely and properly installation and usage of the application.

We do not predict to maintain the product after release and acceptance by the Dolby company and University party. Only one release is planned.

3.5 Maintainability and support requirements

The maintenance shall be done only by the application creators. We do not support any changes done by the user, which include adding, deleting or changing any source files. The user is allowed to save files in a given directory at any time. The product is intended to be used by private users in the Windows operating systems.

3.6 Security requirements

Only creators of the application shall access and change source files. The product shall prevent incorrect file formats to being processed in order to avoid crashes of the application. The product is not intended to store or process any personal or sensitive data, thus no data security mechanisms will e provided. The concern of processing copyright files such as songs etc. is in the user's responsibility. The product creators does not take reponisbility for using the product to process copyright files.

3.7 Cultural and legal requirements

The product shall not be offensive to any social group. The product shall not be used for illegal purposes and with copyright data.