

Group number 07 mYsynth

students’ names & student numbers

|  |  |
| --- | --- |
| Malik quamrus samawat | 281761 |
| DANIEL THOMER | 281839 |
| ALEKSEI GIMBITSKII | 281762 |
| Samaneh ammari | 281796 |
| Zhi wang | 281757 |

Project report and documentation

ELT-23056 Embedded Systems and Electronics Productization

TABLE OF CONTENTS

[TABLE OF CONTENTS 1](#_Toc8028443)

[PROJECT PLANNING 2](#_Toc8028444)

[1.1. Ideation Phase 2](#_Toc8028445)

[1.2. Microcontroller Selection 3](#_Toc8028446)

[1.3. Features 3](#_Toc8028447)

[1.3.1. Audio Stream Record and Playback 3](#_Toc8028448)

[1.3.2. Pitch Change 4](#_Toc8028449)

[1.3.3. Physical Modeling Synthesizer 5](#_Toc8028450)

[1.4. Time Planning 6](#_Toc8028451)

[PROJECT IMPLEMENTATION 7](#_Toc8028452)

[1.5. Hardware Design 7](#_Toc8028453)

[1.5.1. User Interfaces 7](#_Toc8028454)

[1.5.2. Hardware Interfaces 7](#_Toc8028455)

[1.6. Software Design 8](#_Toc8028456)

[EFFECTS LIST 9](#_Toc8028457)

[PROJECT MONETIZATION AND COMPETITION 10](#_Toc8028458)

[TESTING 11](#_Toc8028459)

[RISK MANAGEMENT 12](#_Toc8028460)

[APPENDIX 14](#_Toc8028461)

# PROJECT PLANNING

Project planning refers to everything we do to set up our project for success. It’s the process we go through to establish the steps required to define our project objectives, clarify the scope of what needs to be done and develop the task list to do it. Project planning is the process of establishing the scope and defining the objectives and steps to obtain them.

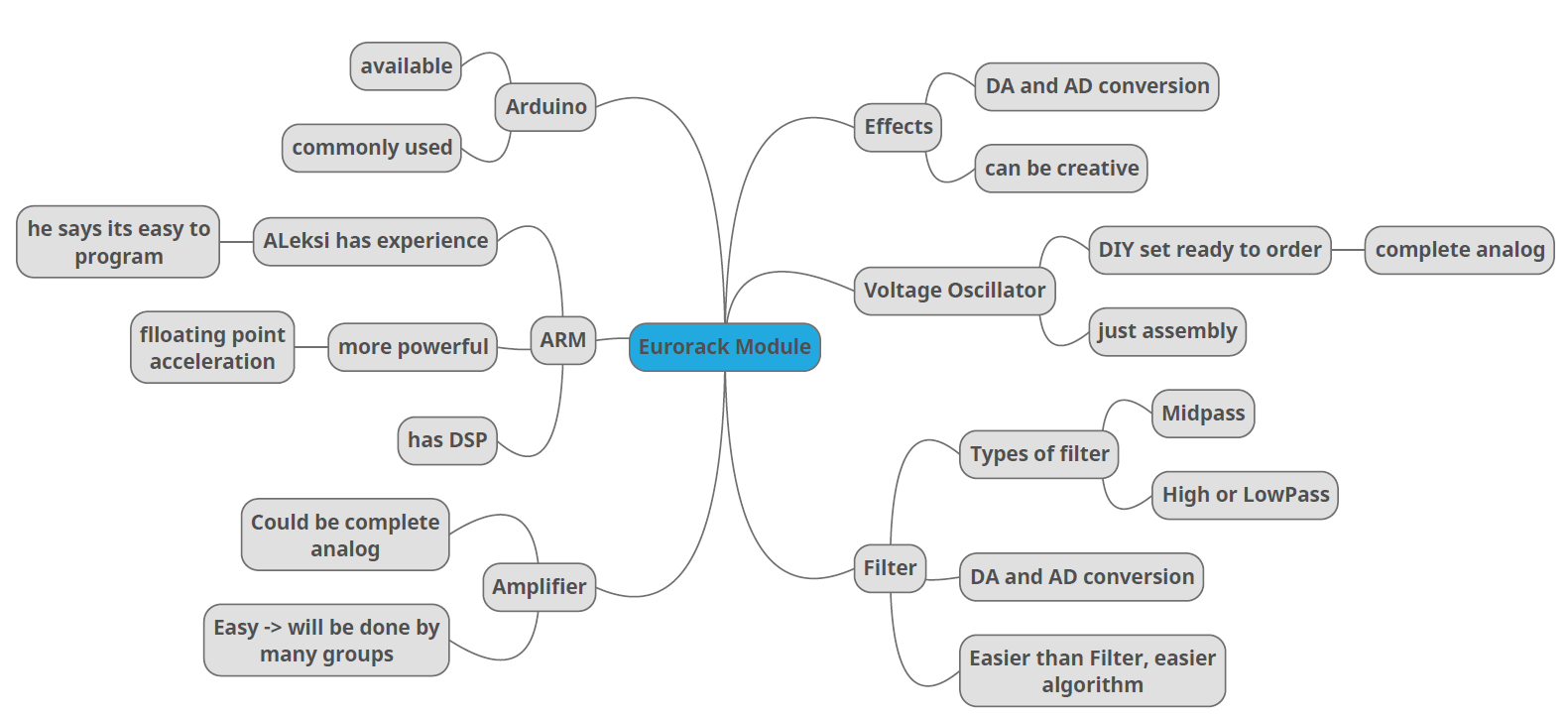
We started planning our project immediately after the first lecture. After detailed discussion on the ideas of each member, we agreed on making an effects module and implementing it as our project. After the initial decision of choosing the topic, we followed the general steps of project planning to come up with a concrete timeline for our effects module.

## Ideation Phase

Ideation is the process of forming ideas and concepts. It means creating new ideas to solve specific problems. Brainstorming is the generation of ideas in a face to face mode. Ideation phase is always when brainstorming takes place, and it is one of the most effective ways to explore new opportunities. The ideation phase is all about diverging and generating as many ideas as possible and to narrow things down and select the best approaches to take further and then to implement one good plan finally and to act and obtain the expected results.

When facilitated in a successful way, Ideation is an exciting process. The goal is to generate a large number of ideas — ideas that potentially inspire newer, better ideas — that the team can then cut down into the best, most practical and innovative ones.

While going through our ideation phase and brainstorming, we came up with the following mind map for our project:



## Microcontroller Selection

For construction and implementation of our effects module, we had two options of microcontroller – Raspberry Pi, Arduino and ARM (using mbed libraries).

Con: performance, effort, doesn´t run Linux – harder to find libraries, more programming effort to get basic function running.

Problems with ARM Cortex-M MCU is that it is too slow CPU.There is no audio processing libraries. Moreover, it is Complex to develop and debug.

On the other hand, Raspberry Pi has a powerfull CPU and very well documented open source audio libraries. It uses ALSA and PulseAudio to abstract from audio hardware.

After a lot of discussion, it was decided to use Raspberry Pi. It is easy to use, not very expensive and yet it has great potential for expendability.

## Features

Our Module to be created shall be fit-able inside a working Eurorack System and provide easy to use simple effects for a modular Synthesizer. The MySynth Effects Module will offer the following functions:

The module features include as follows:

### Audio Stream Record and Playback

* **Description and Priority**

When user doesn’t apply any sound effects, the system should be able to playback the sound stream from the audio stream recording interface.

**Priority:** High

* **Stimulus/Response Sequences**

The audio stream from the recording interfaces will be captured by the audio codec on the HW board. The sound stream will be digitalized by the audio codec and transmit to the SoC through I2S, the I2S controller will generate the interrupt to the driver. The driver will then start the DMA controller and read the data from the FIFO of the I2C controller to the system main memory. Then the driver will notice the upper layer software.

The upper layer software will talk to the driver to playback the data received. The driver will start the DMA controller to move the data to the FIFO of I2S controller. Once the FIFO of the I2S controller is full, the data will be transmitted to the audio codec. The audio code will turn the values into analog signal for the playback interfaces in the panel.

* **Functional Requirements**

**REQ-1:** The audio input jack in the panel. A standard 3.5-mm audio input jack is needed for the audio input signal from the eurorack.

**REQ-2**: The audio output jack in the panel.

**REQ-3:** Audio codec. The audio codec on the board will be responsible for capturing and playback the analog audio stream.

**REQ-4:** I2S controller. The I2S controller in the SoC is responsible for transmitting the digitalized values between SoC and audio codec.

**REQ-5:** I2S controller driver. The I2S driver will be responsible for managing the I2S controller

**REQ-6:** I2C controller. The I2C controller will be responsible for configuring the audio codec.

**REQ-7:** I2C controller driver. The I2C controller will be responsible for managing the I2C controller.

**REQ-8:** Upper layer software. The upper layer software will control and manage the audio data

### Pitch Change

* **Description and Priority**

Provide the function of changing the pitch of sound to implement several funny sound effects for the users. Most funny sound effects are based on pitch change technology, like the Funny Tomcat.

**Priority:** high

* **Stimulus/Response Sequences**

**Option 1:** A tuner to adjust the pitch. The user will turn the tuner back and forth to change the pitch of the sound. When the user turns the tuner, the electrical level of the tuner will be changed. The electrical level of the tuner will be sampled by an ADC channel inside the SoC. The ADC channel will output the value of the electrical level. The user-space application will read the value and change the pitch according to it.

**Option 2:** Two buttons to adjust the pitch. One button will be responsible for higher the pitch and another button will be responsible for lower the pitch. When user presses a button, which is connected to a GPIO pin, an interrupt from the GPIO pin will be generated. The user-space application will read the events and adjust the pitch based on the type of the event. (Higher the pitch or lower the pitch)

* **Functional Requirements**

As the description above, the user input from the HW would be a tuner or two buttons.

SoC receives the external user input

**REQ-9** The tuner. The tuner will output the electrical level. The output of the tuner will be connected an ADC channel of the SoC.

**REQ-10** Two buttons. The two buttons will output the signal of “high” or “low”. The outputs of the two different pins will be connected to two different pins of the GPIO of the SoC.

Change the pitch based on the external user input

After the SoC received the input, the software will receive the input from the SoC and change the pitch according to it.

**REQ-11** The kernel drivers of ADC and GPIO. The driver will receive the interrupts from ADC or GPIO controller and read the values from it, then expose the values to the upper layer software.

**REQ-12** The upper layer software will read the values and the events and change the pitch based on the input from the kernel drivers.

### Physical Modeling Synthesizer

* **Description and Priority**

Physical modelling synthesis refers to sound synthesis methods in which the waveform of the sound to be generated is computed using a mathematical model, a set of equations and algorithms to simulate a physical source of sound, usually a musical instrument.

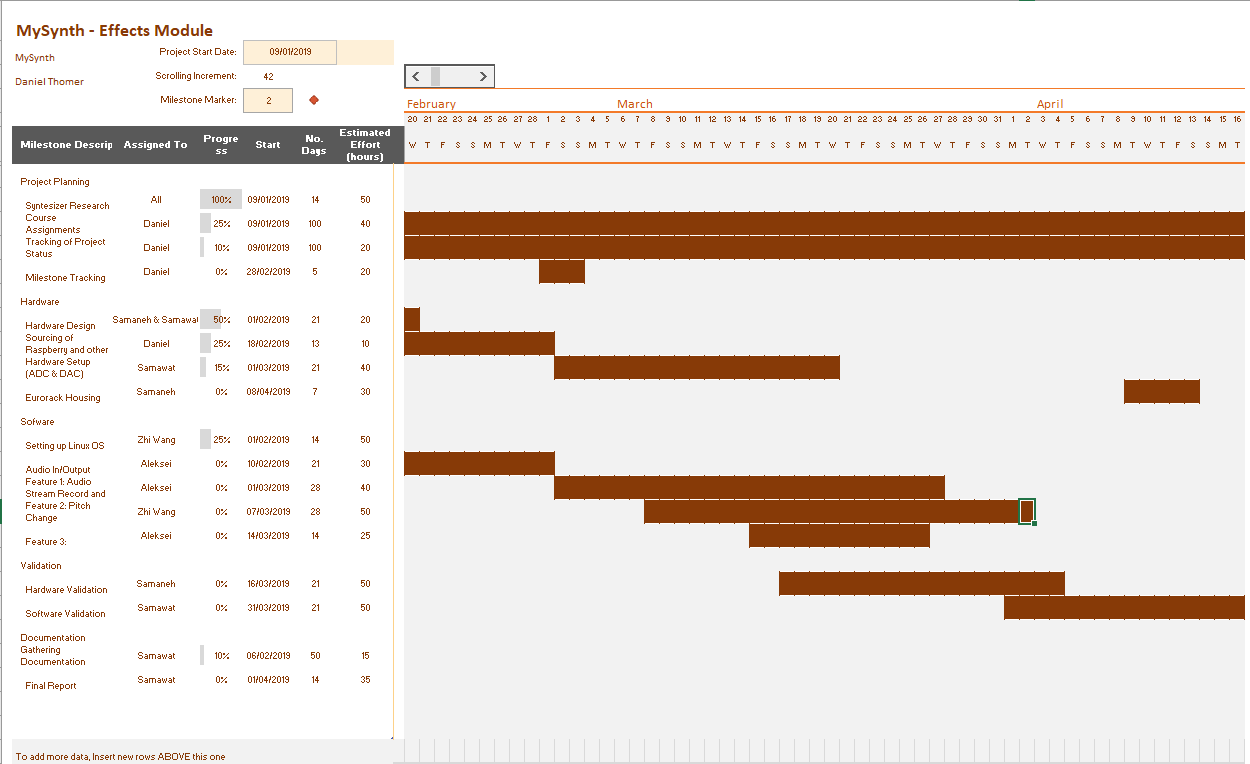
**Priority:** Medium

* **Stimulus/Response Sequences**

When a button of the keyboard is pressed by the user. The key code will be scanned by the keyboard IC. The keyboard IC will generate the interrupt to the SoC. The keyboard driver will read the key code from the buffer of the keyboard IC through I2C bus. The key code will be reported to the upper layer software. The upper layer software will figure out the note by the mapping table between key code and notes. With the notes, the physical modelling synthesizer library will generate the wave of specific instruments. The upper layer software will playback the wave to the I2S controller driver, the I2C controller driver will start the DMA and move the wave data to the FIFO of I2S controller. The I2S controller will transform the wave data into analog signals and play them through the analog output.

## Time Planning

The project’s time planning was done by a Gantt chart and we tried to follow the schedule accordingly. The Gantt chart is as follows:



The details clear version of the time schedule is attached in the appendix.

# PROJECT IMPLEMENTATION

## Hardware Design

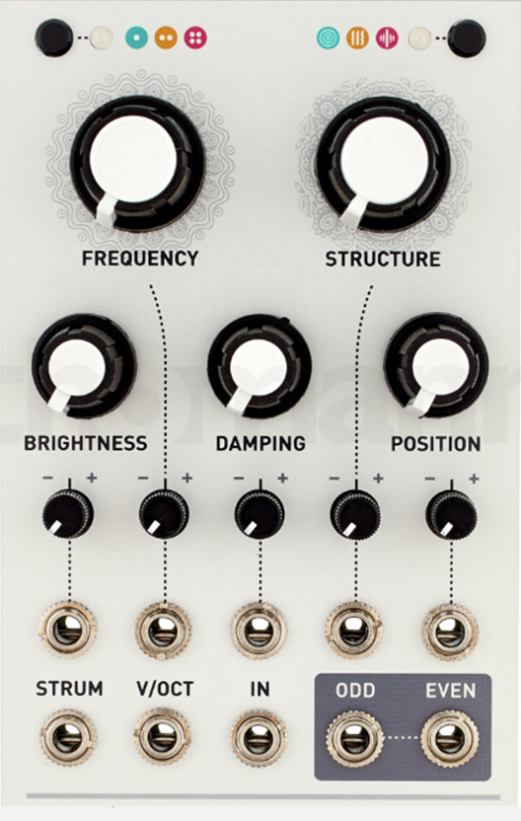
### User Interfaces

The User Interface shall consist of:

* 1 Knob to control effects
* 4 Buttons to control effects
* We use the LCD display contained in the shield

The User Interface should be visually similar to the picture shown on the right.

### Hardware Interfaces

The module shall have one analog Input:

* Audio Input
* Voltage Range: 0V -5V

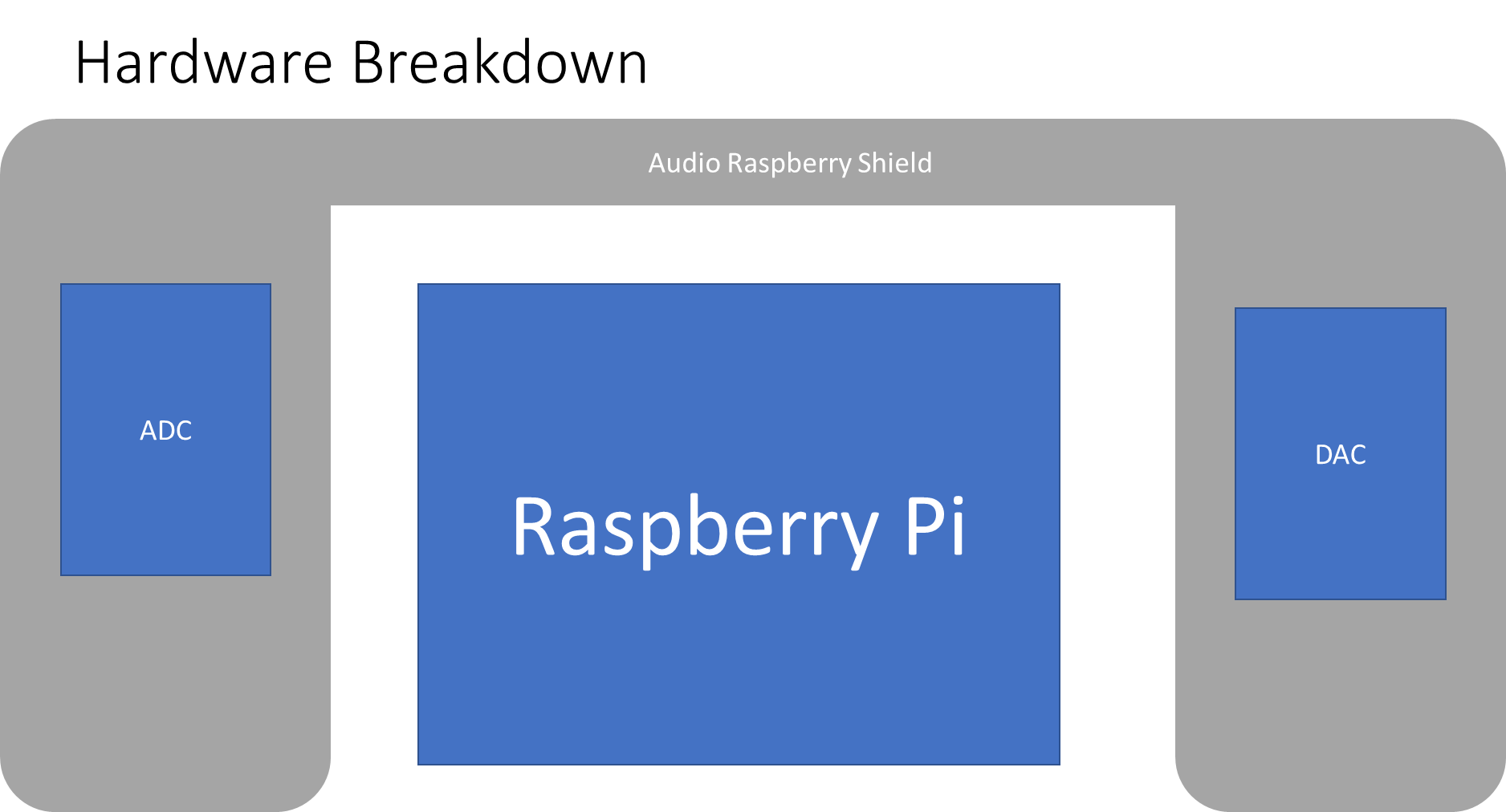
The module shall have one analog Output:

* Audio Output
* Voltage Range: 0V -5V

The Input voltage shall be transformed using a ADC to be used in Software.

After the Effects described in Chapter 4 have been applied by Software, a DAC shall be used to create the analog Output.

A schematic of the HW design is as follows:



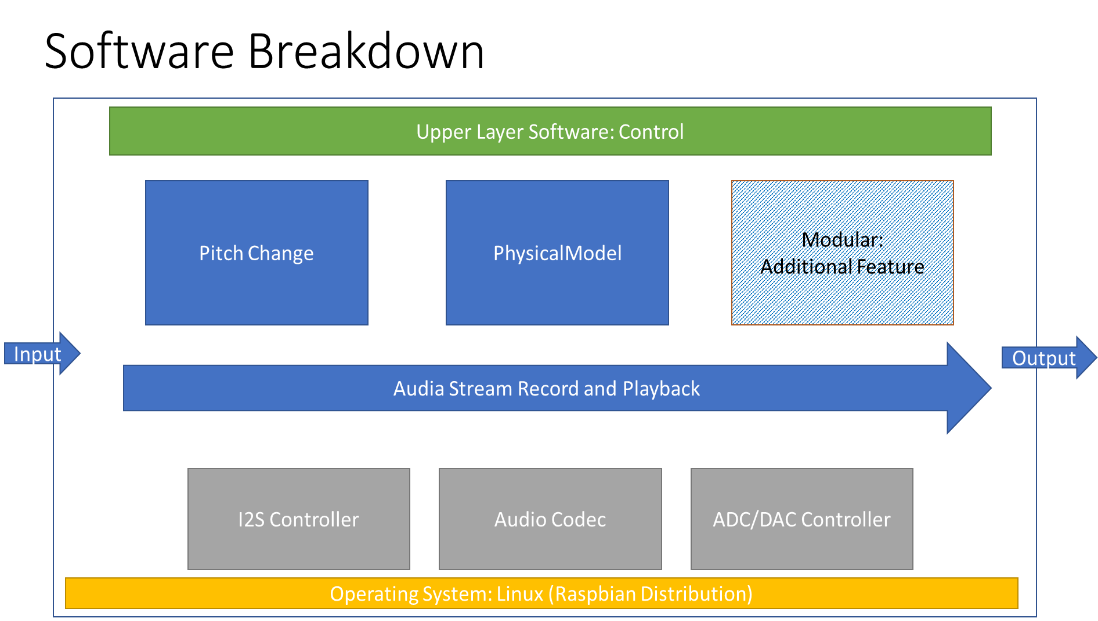
The Frame was designed according to the mechanical specification, which was provided by Jukka Vanhala. Solid work software was used here. It was then printed on wooden material by laser cut. First version was tested on the Euro rack which was not compatible. after that some changes were made and second version was suitable.

## Software Design

The Software and Hardware shall work within a Eurorack system as specified by Doepfer. For implementing the software, ALSA (**Advanced Linux Sound Architecture**) was used. It   
allows to abstarct from audio hardware. The module will work with any soundcard, if it has Linux driver.

We used The Synthesis ToolKit in C++ (STK)**.** There is an open source audio processing library (https://ccrma.stanford.edu/software/stk/), well documented, very easy to integrate to the project as Linux library.

The schematic is as follows:



# EFFECTS LIST

The final list of effects implemented in out module are as follows:

* Frequency filters (three low-pass filters, two band-pass filters and one high-pass filter).
* Delay blocks ("echo" effect).
* Modulators (one "vibrato" effect which multiply incoming audio signal and sinusoidal wave)
* Non-linear blocks ("distortion" effect)

By combining these 4 basic blocks every effect can be implemented, we can have in real synthesizer. Here we demonstrate that our program is capable of doing everything what is may be required by end user.

# PROJECT MONETIZATION AND COMPETITION

The MySynth module is designed to be very modular. This is it´s big advantage over the competition. All products offered by competitors have a fixed set of effects that they offer. That makes it very hard for beginners or even advanced users to decide which module to buy because they will be unsure which effects they really need and there might not be a single module offering everything that they need for different music projects.

This problem can be solved by the modular approach of MySynth. The idea is that for the final product the users buy only a base set, including the Hardware and some basic effects. Then they can easily upgrade to more effects by buying software from us. A similar approach is used by many game companies already and proofed to be very effective earning more money in the end than by just selling a single product.

# TESTING

The module was tested for various parameters based on the test plan (attached in the appendix). Few photos of the testing being done is as follows:

A close up of a computer

Description automatically generated A close up of a computer

Description automatically generated

A close up of a computer

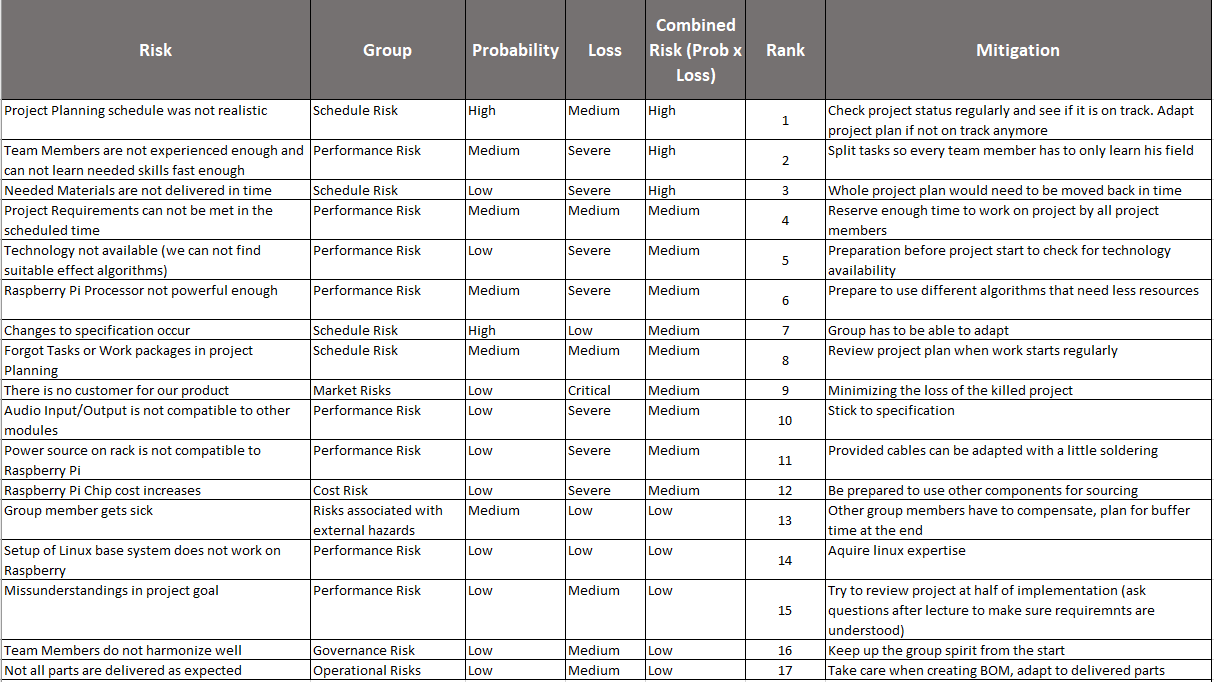
Description automatically generated

# RISK MANAGEMENT

For our synthesizer, we forecasted few possible risk factors and considered their probability of occurring. Accordingly, we calculated the category of loss caused by the occurrence of each of those risks and ways to mitigate them. The risks were classified into few groups such as – schedule risk, performance risk, market risk, cost risk, government risk, operation risk, risk associated with external hazards etc.

The probability of each risk was measured in a scale of low, medium and high. The risks were ranked and placed accordingly in a chart form and ways to mitigate the risks were discussed and decided.

The summary of our findings is given in the following chart:



The details clear version of the time schedule is attached in the appendix.

# APPENDIX

All Appendices are in the submitted ZIP File with this report:

* Ammari\_ELT-23056\_Logbook.xlsx
* Gimbitskii\_ELT-23056\_Logbook.xlsx
* Samawat\_ELT-23056\_Logbook.xlsx
* Thomer\_ELT-23056\_Logbook.xlsx
* Wang\_ELT-23056\_Logbook.xlsx
* Created Code
  + MySynth.cpp
  + Input.py
  + The full Code can be found on <https://github.com/AlekseiGimbitskii/MySynth_ELT-23056>