Audio Coding Pedal Platform for Live Performances and Prototyping

Small low-budget effects pedal platform for live coding, prototyping and music performances.

Giuseppe Taormina

Master Thesis

in

Sound Design

Mentor: DI Marian Weger, PhD

FH JOANNEUM
Institut für Elektronische Musik und Akustik
Universität für Musik und darstellende Kunst Graz

Graz

Abstract

This master thesis documents the process of construction, design and development of a platform in a guitar pedal format, for live coding, prototyping, live visuals and live performances in general.
Live audio programming is an increasingly common practice, but it always requires the use of the computer or several other elements. The possibility of being able to condense everything into an instrument of reduced size almost like a normal guitar pedal could increase the practicality of this practice and allow it to be carried out in any situation. Furthermore, the possibility of always having one's own effects available, i.e. coded by oneself, in a low-cost platform and being able to modify them from the same platform using the favourite computer music language leads to the development of something new, not yet on the market.
The pedal in question is based on the Raspberry PI microcomputer and cheap DIY electronics. The components used are chosen to obtain the smallest possible dimensions of this platform, and above all to obtain a product that is as economical as possible. In addition to the processing of the incoming audio signal (guitar or any other instrument), this pedal is able to generate visuals reproduced on the screen included in it, which vary according to the audio processing and which can be projected in case of a live performance, enriching its content.

Kurzfassung

Diese Masterarbeit dokumentiert den Prozess der Konstruktion, des Designs und der Entwicklung einer Plattform im Gitarrenpedal-Format für Live Coding, Prototyping, Live Visuals und Live-Performances im Allgemeinen.
Live Coding wird immer häufiger praktiziert, erfordert jedoch zwingend den Einsatz des Computers oder mehrerer anderer Elemente. Die Möglichkeit, alles in einem Instrument mit reduzierter Größe fast wie ein normales Gitarrenpedal verdichten zu können, könnte die Praktikabilität dieser Praxis erhöhen und es ermöglichen, sie in jeder Situation durchzuführen.
Darüber hinaus ergibt sich die Möglichkeit, eigene Effekte (selbst codiert) immer verfügbar, in einer kostengünstigen Plattform zu haben und von derselben Plattform aus mit der bevorzugten Computermusiksprache modifizieren zu können; zur Entwicklung von etwas Neuem, das sich noch nicht auf der Markt befindet.
Das Pedal basiert auf dem Raspberry PI und kostengünstiger DIY-Elektronik. Die verwendeten Komponenten werden ausgewählt, um die kleinstmöglichen Abmessungen dieser Plattform und vor allem ein möglichst billiges Produkt zu erhalten. Zusätzlich zur Audioverarbeitung des eingehenden Audiosignals (Gitarre oder jedes andere Instrument) kann dieses Pedal auf dem darin enthaltenen Bildschirm wiedergegebene Visuals erzeugen, die je nach Audioverarbeitung variieren und während einer Performance projiziert werden können, um ihren Inhalt zu bereichern.

Obligatory signed declaration

I hereby confirm and declare that the present Master's thesis was composed by myself without any help from others and that the work contained herein is my own and that I have only used the specified sources and aids. The uploaded version is identical to any printed version submitted. I also confirm that I have prepared this thesis in compliance with the principles of the FH JOANNEUM Guideline for Good Scientific Practice and Prevention of Research Misconduct. I declare in particular that I have marked all content taken verbatim or in substance from third party works or my own works according to the rules of good scientific practice and that I have included clear references to all sources. The present original thesis has not been submitted to another university in Austria or abroad for the award of an academic degree in this form1 . I understand that the provision of incorrect information in this signed declaration may have legal consequences.

_____

Date, Signature

Acknowledgements

I would like to thank my supervisor Dr. Marian Weger for his guidance throughout the research and conception of this work, for encouraging me to always give my best and for making this process a significant source of learning. For the help she gave me with the 3D Model, I would like to thank my friend and 3D Artist Stefania Fiorucci.
Special thanks go to my friend and mentor Dr. Dr. Markus Mohr for making all this possible. I would also like to thank my family and all the people close to me for their support over these years.

Table of Contents

List of Figures

1. Introduction

The need to have a sound processing effect in the form of a pedal arises from the need to be able to activate, deactivate or modify parameters during a performance. In addition to the execution of the notes, taking care of the expressive side through dynamics and precision, the search for interaction with the sound parameters, capable of altering the sound spectrum in ways not possible with the typical interaction with a musical instrument, finds an answer in use of musical effects capable of altering the sound.
When playing a musical instrument, both hands are normally used and their use to interact with the desired effect could lead to poor fluidity in execution. Being able to interact with a sound effect through, for example, using your feet to click on the pedal switches, does not risk interrupting or hindering the performance.

2. Pedals

Since the guitar became an amplified instrument around 1930, to make it audible in the context of big bands, it has been tried to find ways to increase its expressiveness.

The first of these experiments led to the creation of the first guitar with an effect built into it, the Rickenbaker's Electro Vibrola Spanish Guitar, around 1938-1939 (see Fig. 1). [1]

Figure 1: [2]

It has a motor capable of moving the bridge by creating a further oscillation of the strings by slightly altering the pitch of the notes. It then generates a Vibrato.

2.1 First Pedals

The first actual effect pedal was a volume pedal, the Rowe industries DeArmond Model 600 (see Fig. 2).

Figure 2: [3]

It was designed to expand the functions of another company product, namely a Pickup (magnet capable of transducing the vibration of the strings into an electrical signal), capable of being attached to any acoustic guitar. [1] [4]

Furthermore, the same company devised a tremolo, or a pedal capable of creating a volume oscillation, the DeArmond Model 601 Tremolo Control (or Trem-Trol, Model 60 or 60A). It is the first stand-alone pedal in history (see Fig. 3). [1]

Figure 3: [5]

This pedal works by passing the signal through an electrolytic fluid with an electric motor that creates fluctuations in the signal by moving a spindle. [1]

These first pedals, however, were still too bulky and fragile to be brought to the stage. The next step took place in the 1960s, when transistors became easily available, making it possible to build lighter and cheaper pedals. [1]

In 1961, while recording Marty Robbins' "Don't Worry" at the Nashville Studios, an output transistor caused the bass signal to distort, effectively becoming the first distortion recording in a song. Studio engineers Glenn Snoddy and Revis Hobbs tried to design a circuit to replicate that sound. This led them to build the Gibson Maestro Fuzz-Tone, the first true distortion pedal in history (see Fig. 4).
It became famous thanks to its use in the song "(I Can't Get No) Satisfaction" by the Rolling Stones by guitarist Keith Richards, who was trying to emulate the sound of brass instruments. [1]

Figure 4: [1]

Several companies have therefore begun to develop similar models.

Another major breakthrough came when engineer Brad Plunkett was working to find a way to make VOX amplifier manufacturing less expensive. While trying to replace a switch with a less expensive pot, he noticed a strange fluctuation in pitch.
He then took a volume pedal from a Vox Continental organ and replaced the circuit with the one he was working on and got what we know today as Wah-sound. [1]

By pressing the pedal, we will have an accentuation of low frequencies or high frequencies, depending on how we move the potentiometer. It was called Cry Baby (see Fig. 5).

Figure 5: [1]

2.2 More Pedals

Therefore, more and more companies were born and in 1976 the Electro-Harmonix company released the Memory Man pedal, which creates signal delay lines that can be mixed with the signal itself (see Fig. 6). This was a very important moment, as guitarists finally had access during a performance to what was previously only possible in the studio. [1]

Figure 6: [1]

Pedals capable of creating various types of effects were then built in the 80s, where digital multi-effects began to spread.
For example, units comprising several programmable effects, such as the Roland GP-8, which includes 8 effects: Wah, Compressor, Overdrive, Distortion, Phaser, Equalizer, Delay and Chorus (see Fig. 7).

Figure 7: [6]

Subsequently, the first emulations of amplifiers and speakers were created, such as the Line 6 POD (see Fig. 8). [7]

Figure 8: [7]

In addition to a variety of effects, it also allows to choose the type of amplifier and to save its presets. All compact und handy.
This can be used both in the studio and for live performances, reducing the size of the instrumentation and giving the possibility of obtaining sounds that are not easily obtainable with limited instrumentation.

2.3 Pedals today

The quality of these emulators (modelers) has grown enormously in recent years, creating products that can easily compete with the sounds obtainable from real amplifiers and/or normal pedals.

Currently, the most used modelers are Line 6 Helix (Fig. 9), Kemper (Fig. 10), Fractal Audio Axe FX (Fig. 11) and Neural DSP Quad Cortex (Fig. 12).

Figure 9: Line6 Helix [8]

Figure 10: Kemper [9]

Figure 11: Fractal Audio Axe FX3 [10]

Figure 12: Neural DSP Quad Cortex [11]

These are now widespread in recording studios, but also in the live sector. They are in fact replacing the traditional instrumentation, moving more and more towards the all digital signal chain.

3. Live Performances

With the development of the first effects for musical instruments and their increasing popularity, musicians and researchers began to think about using them, or similar devices previously used only in studios for sound research purposes, in live contexts.

3.1 Live Electronics

This has already started in the 1960s, simultaneously with the development in all artistic fields of the concepts of improvisation, extemporaneousness and happening (a show that includes several arts at the same time, such as a concert accompanied by visuals). [12]

What will take the name of live electronics, begins to develop. The sound of the instruments or the voice is transformed through electronic devices in real time. This practice begins in a more pioneering and experimental setting. Among the various composers, Karlheinz Stockhausen stands out with the pieces “Mikrophonie I”, “Mikrophonie II”, “Mixture” and “Mantra”. [12]

Live electronics begins to spread more and more and to welcome new electronic instruments, such as the synthesizer. It advances together with technological development, expanding its expressive possibilities. [12]

3.2 Live Coding

The spread of computers and their use in the music field has also led to the use of programming languages for musical creation, as well as the development of programming languages totally designed for sound synthesizing and processing.

Among these programming languages developed for music, we find SuperCollider, CSound and PureData. They allow sound synthesis (the generation of sound artificially), starting from a simple oscillator to then be able to create more complex sound waves, and the processing, analysis and use of samples, or musical events already recorded.

This practice has also been brought to the live field, finding its space close to live electronics.
A typical live coding music performance takes place with the projection of the programmer's screen, making one's code the visual event of the performance. [13]

This documentation deals with the creation of an instrument that can also adapt to traditional performances, happenings, live electronics and live coding.

4. Raspberry Pi

The Raspberry Pi (see Fig. 13) is a computer that was developed by Eben Upton, Pete Lomas and David Braben with the intention of having an inexpensive computer for programming and with easily accessible functions for various electronics projects .
In order to reduce costs as much as possible, it features only the essentials, namely programming environment and hardware connections for electronics projects. It runs Raspian, a modified version of Debian Linux. [14]

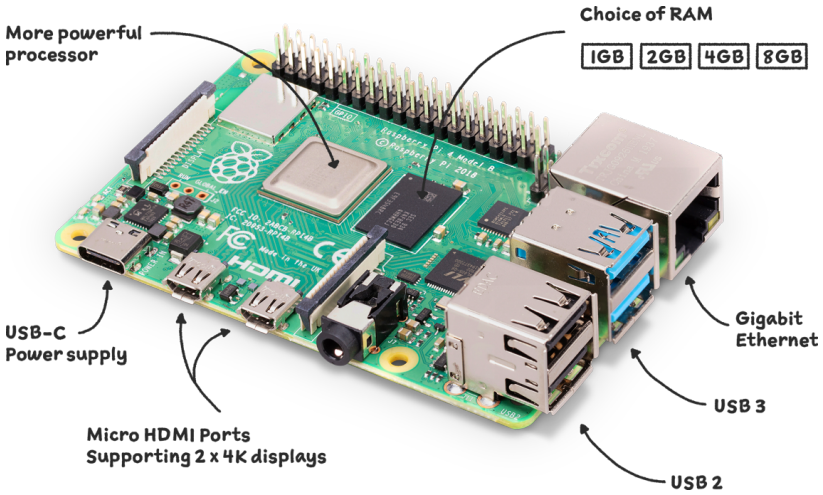

Figure 13: [15]

The photo above represents the Raspberry Pi 4. The model used is the version with four GB of RAM.
It has a USB-C input for power supply (it is preferable to use the original power supply), two HDMI ports from which the audio connection is also made (if a TV or screen with speakers is connected in general), a 3.5 mm output audio, four USB ports (USB 3 and USB 2) and one input for the ethernet cable.
It also has 40 pins (type of electrical contacts) to which various hardware components can be connected for the most varied electronics projects.

All these built-in functions, the ease of use and its low cost (between €60 and €100) have determined its use in this project.

4.2 Existing Works

Raspberry Pi based guitar effects have already been developed, and some of them, which were taken as inspiration for this work, are the Pedal-Pi [16], NeuralPi [17] and Hyllis' do-it-yourself touchscreen MultiFX [18] [19].

Pedal-Pi is a programmable guitar pedal from ElectroSmash (Fig 14). [20] It uses the Raspberry Pi ZERO Boards and the effects are coded using the programming language C. It is quite basic as it only has a push button for on/off, two potentiometers and a switch to adjust the parameters. It is therefore necessary to program the desired effect first, and then it will be possible to interact with it. Like a typical guitar pedal, it features only one input and one output.

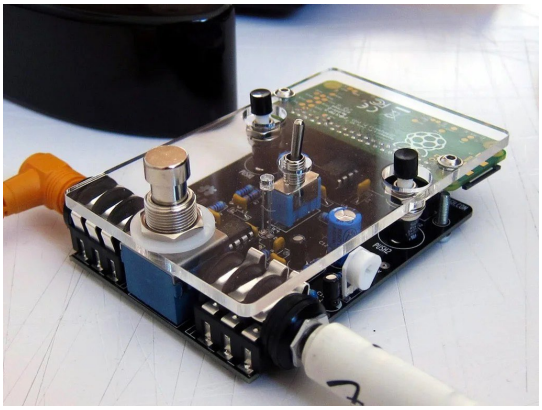

Figure 14: [16]

NeuralPi is the name of the open source software written by Keith Bloemer [21], designed to be used with Raspberry Pi (see Fig.15). It is written in the JUCE framework [22] and uses neural networks to model existing effects.
The computer used to run the software is the Raspberry Pi 4b with the low latency audio operating system Elk Audio OS [23] and the HiFi Berry ADC + DAC as audio interface.

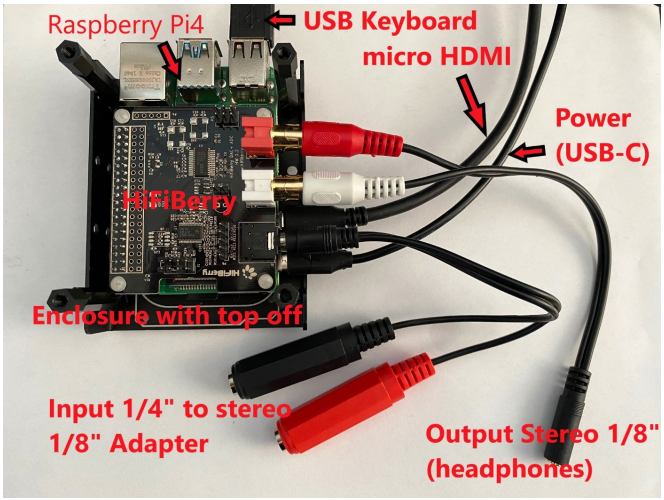

Figure 15: [17]

The latest is the **DIY Touchscreen Pedal** developed by Hyllis (see Fig. 16). [19]
It uses Raspberry Pi and rotary encoder, as well as a Behringer U-Control UCA222 audio interface, and is also made with cheap DIY electronics.

The software used is written by the author in the C programming language.

Figure 16: [18]

This is closest to to the pedal object of this thesis, due to screen and hardware control, but its use is limited to written software and doesn't have enough features.

5. Building the Pedal

5.1 Idea

The idea behind the design of this pedal is to have, on an aesthetic level, a similar product to the more modern modelers mentioned above (NeuralDSP QuadCortex, Line6 Helix, Kemper, etc.), but at the same time reduced dimensions, making it easily transportable and not bulky, and above all at the lowest possible cost.

The most important and distinctive feature of this pedal is the possibility of having access to various programming languages, being able to connect our instrument directly to this computer in guitar pedal format and to manipulate the programmed sound as if our code were the circuit of this pedal, controlled by the potentiometers installed on it.
Any effect can be programmed in any programming language (if compatible with the Raspberry Pi Linux operating system) and easily integrate it into an effects chain, or go directly to the Mixer to listen to the final result.

It must be accessible from any type of signal, instrument (hi-impedance) or line (low impedance), as well as provide the possibility of separating the input signal, in order to have a dry signal that can in case be mixed with the signal processed by the Pi (wet signal).
An effects loop must be inserted in the dry path, consisted of an i
Input (send) and an output (return) loop, through which an external pedal can be connected to process the dry signal. This can be very useful in case of physical modeling, to constantly have direct control with the hardware to be modeled.

Trying to keep the budget as low as possible, most hardware parts are homemade, being able to reduce the cost to only the necessary materials. Obviously, the choice almost always fell on DIY, being able to reduce the cost to only the necessary materials.

How does the most basic pedal work? It has a 6.3 mm jack input and output, a push button to activate it and, in the best of cases, also a potentiometer capable of adjusting a parameter. The pedal of this thesis must have some more features, which make it unique and innovative in this environment.

5.2 Characteristics

The first considerations regarding this work led to the collection of the following characteristics:

- It must be as economical as possible. (This led to the choice of using the Raspberry Pi and DIY electronics).

- It must be small, but solid at the same time, ready to be used on stage during live performances,

- A built-in screen is required for live programming and visual FX.
- It must be possible to program directly on the pedal in any programming language. (This can be obtained, for example, by connecting a mouse and an external keyboard using the USB inputs already present in the Raspberry Pi.)

- It can be controlled remotely from any USB-MIDI or OSC controller (using LAN or WLAN).

- Send/return sockets to connect other pedals in the dry path. This is useful for example for physical modeling purposes. (To compare the original pedal with the emulation being programmed).

- Switch bypass. If the pedal is not powered, the signal passes the pedal unaffected.

- The dry and wet signals must be mixed in analog domain, with the possibility of controlling their volume via 2 separate potentiometers.

- Everything is powered by a single power supply, namely the official Raspberry Pi one.

5.3 Hardware

Once the aforementioned points were taken into consideration, it was possible to reduce the hardware part necessary for the first draft to the following:

- 2 mono jack socket 6.3 mm per inputs (left and right), to connect both mono and stereo sources.

- 2 mono jack socket 6.3 mm outputs (left and right).

- 2 stereo jack socket 6.3 mm for Send / Return.

- 3 rotary encoder, for managing parameters or activating other functions.

- 2 potentiometers for dry and wet.

- 1 button/switch (for on / off).

- 1 touch screen.

- 1 Raspberry Pi 4, which USB, WLAN, POWER and HDMI must remain accessible from the custom-built case.

The size of the enclosure will then be adjusted accordingly, in order to have all these features in the smallest possible space.

6. Signal Chain (Building the Pedal)

The signal chain adopted allows to divide the input signal to separate the wet signal from the dry in analog domain.

Below is a draft of the first signal chain, which involved the splitting of the two inputs (left and right) and their combination in the final mixer, to then be levelled and continue to the stereo output.

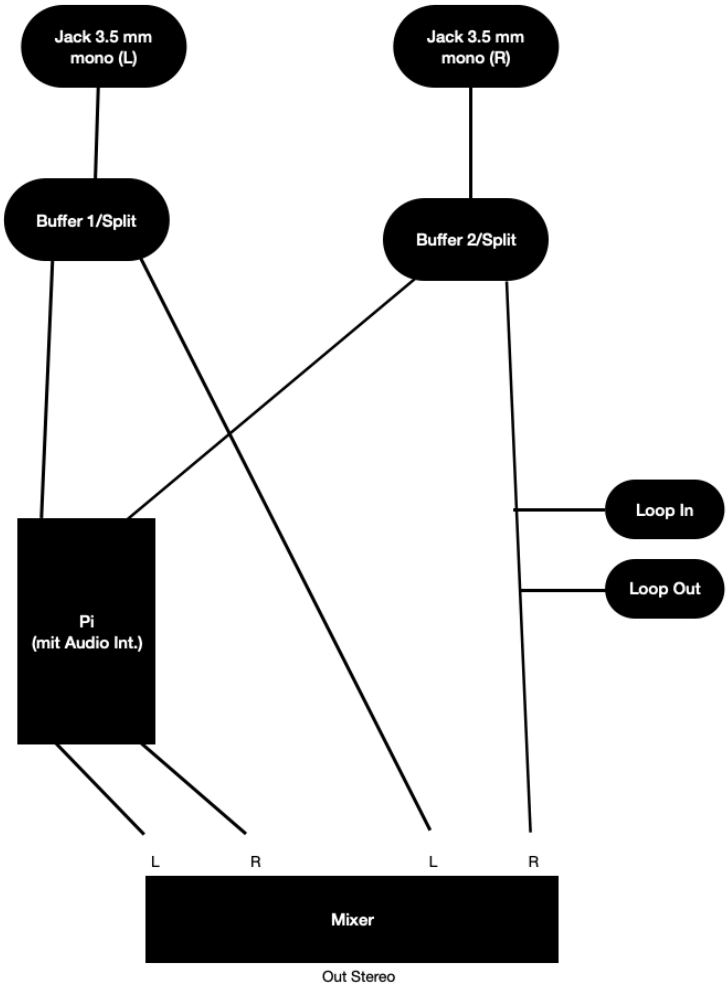

Figure 17

In this first step, the loop FX was considered mono for convenience, making it possible to use it only in the dry path of the right input.

6.1 Switch True Bypass

One of the key points of this project is having a 100% true bypass pedal.
A pedal is true bypass when, if the pedal is off, the input signal goes directly to the output, ensuring the integrity of the signal, without any alteration. Furthermore, if the pedal does not receive power, the input signal will be directly sent to the output.

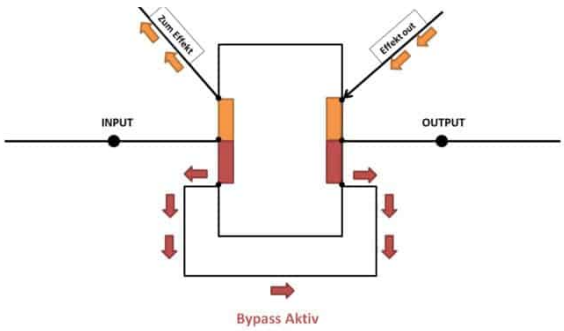

Figure 18: [24]

Not all pedals are true bypass, as this causes a typical switching noise every time the pedal is activated and, the presence of a true bypass, excludes the presence of a buffer, which reduces any noise and slightly alters the signal in input, but avoids signal loss if very long audio cables or multiple effects in chain are used. [24]

In this case, however, the buffers are necessary to be able to split the signal. The solution was therefore to insert this true bypass system even before the Buffers.

Typically, true bypass is obtained with the 3PDT (3 Poles Double Throw) switch.

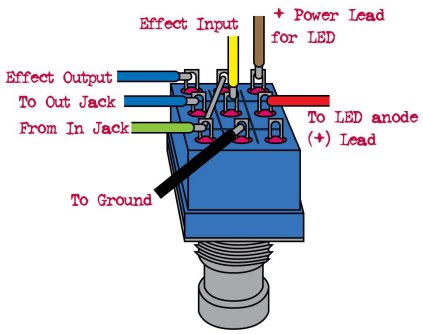

Figure 19: [25]

However, soldering directly on the pins is not very convenient (in case of future changes) and the price is around €4/5.
A more convenient and cheap solution is to use relays.

6.1.1 Relay

A relay is a switch that uses the electrical signal to control an electromagnet capable of connecting or disconnecting two circuits. [26]
It consists of various parts, an electromagnet built by winding a copper coil on a metal core, switching points (contacts), a mechanically moving contact and a spring. The contacts are three and are called normally open (NO), normally closed (NC) and common (COM). [26]

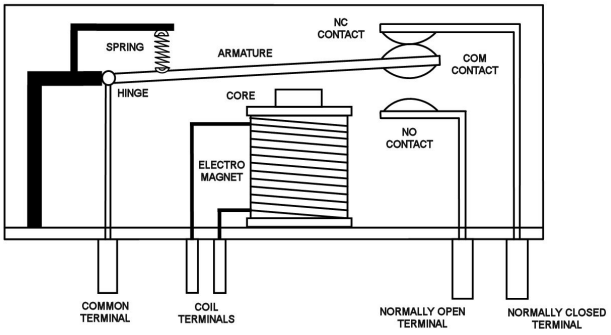

Figure 20: [27]

The image above represents the state of the relay when it is not powered. Common contact (COM) is connected to NC contact.
On the other hand, when the relay receives current, the electromagnet receives voltage and the current flowing in the coil produces magnetic energy in the core which lowers the armature connecting the COM contact with normally open (NO). [27]

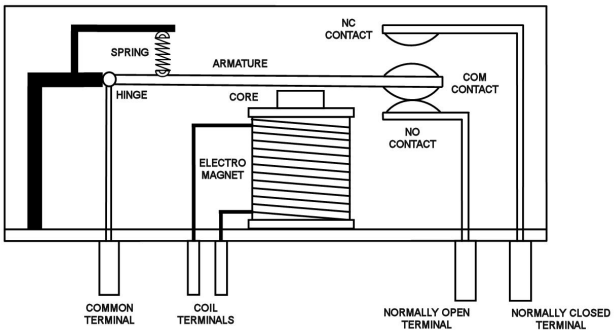

Figure 21: [27]

For this work, the Intertec Relay TD-5 was the best choice. The choice fell on it as it needs 5V current, which is directly obtainable from the Raspberry Pi and is also stereo. Being stereo, it can be used to simultaneously switch the direction of both input signals (left and right) and send them to output, keeping them separate.

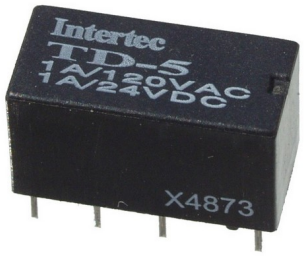

Figure 22: [28]

Here the wiring diagram corresponding to the position of the image above:

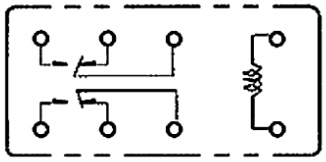

Figure 23: [29]

There are four contacts in the upper row and four in the lower one therefore, being stereo, the first line corresponds to L and the other to R channel.

The first contact (the one on the far right in Fig. 23) receive the electric current to alternate NO and NC. One row will be connected to the switch which will send voltage, the other will go to ground instead.

The input signal is then connected to COM, and the next two pins will be the two distinct outputs. When the relay does not receive current, this pin is connected to NC, this is precisely our output true bypass. When the switch is off and the Pi is not receiving current, the signal will flow in this direction. If, on the other hand, the switch is on (and obviously the Pi is powered), COM will be connected to NO, making the signal flow in this direction.

From here the signal will then go to the buffer where it will be split creating the two paths, dry and wet, which will then be combined in the final mixer.

The same thing applies to the second row, which represents the other channel.

The addition of the relay has partially modified the signal chain as showed in Fig. 24.

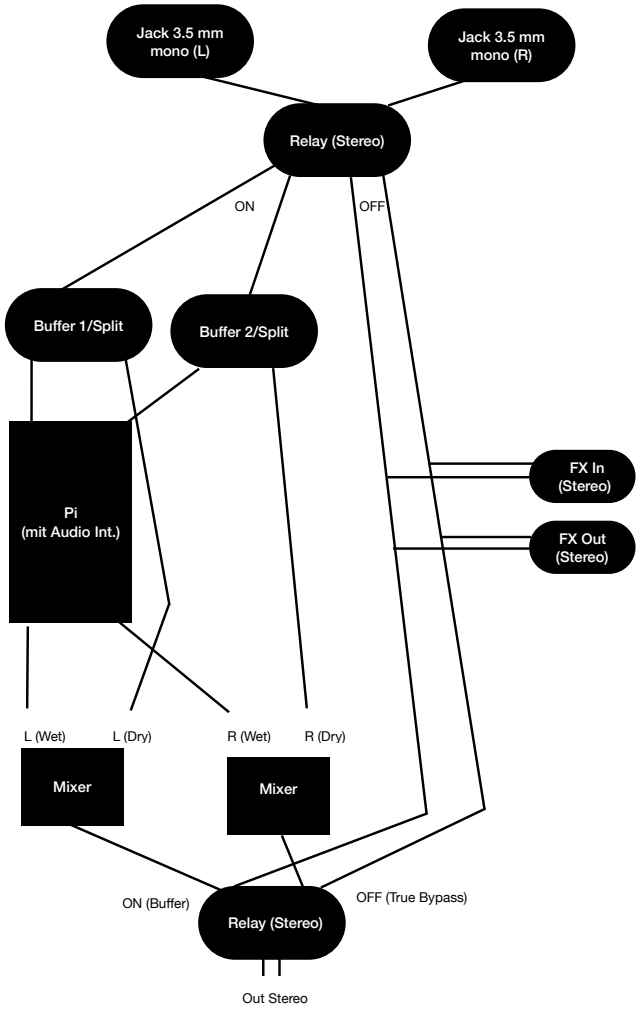

Figure 24

There are two relays, one at the beginning and one at the end, after the mixer. They are controlled by the same switch, which sends current to both relays, switching their contacts at the same time.
In this way, it is possible to exclude the final mixers and have the pedal true bypass. The signal is not affected by the pedal components in case the switch is in the off position, or the Pi does not receive power.

The final relay, also the TD-5 stereo relay, is connected to the stereo output, connecting one channel to the right output and the other to the left. The relay pins are used a little differently this time than before.

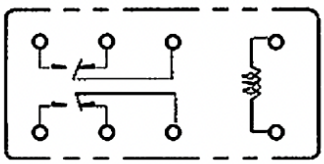

Figure 25: [29]

The first pin from the right is always connected to the switch, the other goes to ground. COM is connected with the stereo output, becoming the actual output.

In absence of current, it is connected directly to NC, which will be connected to NC pin of the initial relay, guaranteeing the passage of the signal, bypassing the pedal. NO is connected to the mixer output, and therefore receives the wet/dry signal.

As for the switch, the simplest and safest choice was to use a simple two-pole switch. If pressed and, therefore, in on mode, the circuit is completed, generating continuity between the two poles and thus making the current flow. If it is in off mode, the circuit is broken.

Figure 26: [30]

Below is the configuration adopted (made twice, for both buffers), that was developed on the open-source circuit design software DIYLC: [31]

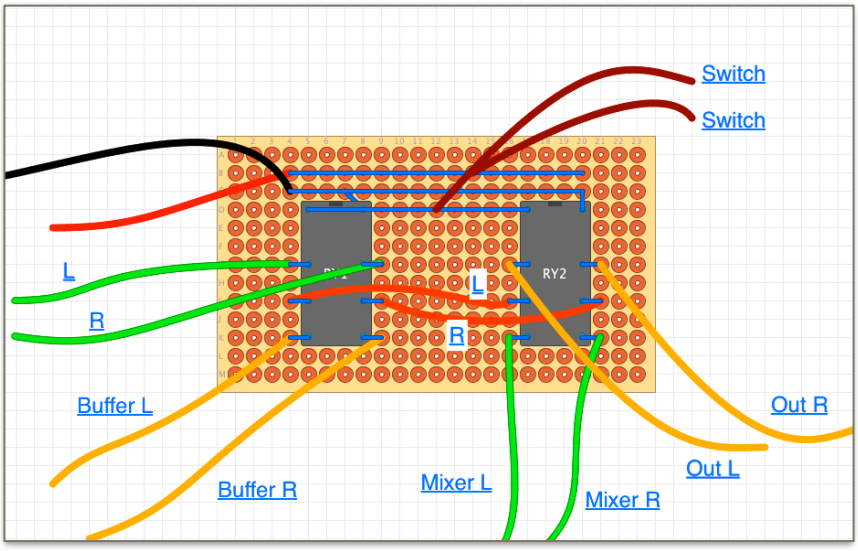

Figure 27

If the switch is off, the signal will follow in the direction that connects the two relays to each other, giving us the true bypass. If it is on, then the input signal (L and R, the green cables) will go to the buffers and, subsequently, the output of the mixer (mixer L and mixer R, green cables) will be connected directly to the general output.

The FX loop is still missing, which will be placed in the true bypass path, between the two relays. This part will be discussed later in the documentation.

This is soldered on a perfboard (see Fig. 28a), a board for permanently soldering the circuits, consisting of single holes that allow the organization of the circuit in the every desired direction.

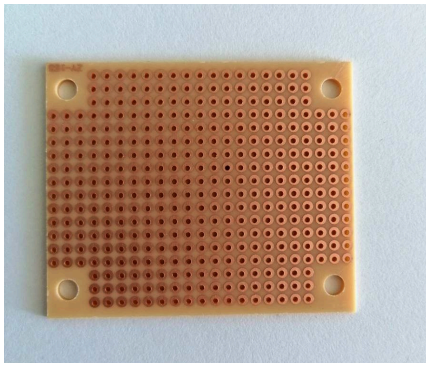

Figure 28a

Figure 28b shows the soldered circuit.

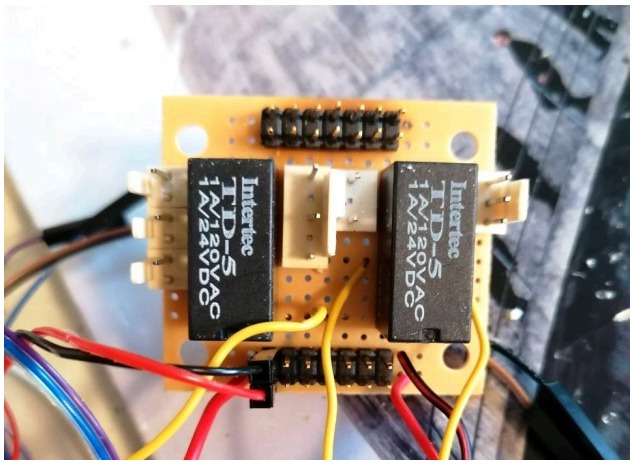

Figure 28b: The pins in the lower part are used to distribute the 5v current (and the ground) received by the Raspberry Pi. The lower ones, on the other hand, to distribute the 3.3v current or, later, 9v.

6.2 Buffer

There are 2 inputs, L and R. Both are split through a JFET Buffer.

A buffer is an amplifier that does not alter the signal gain which transforms the impedance of the incoming signal. [32]

Considering the possibility of wanting to connect, for example, a guitar to the pedal, you will have a high impedance signal at the input and, if the pedal input were at low impedance (low impedance input), there will be a loss of high frequencies. This is where the buffer comes into play, which is featured in many guitar pedal effects. It has a high impedance input and a low impedance output, transforming the input signal. This avoids loss of definition (loss of high frequencies). A typical basic buffer is the JFET (Junction Field Effect Transistor).
The schematic symbol of the JFET is shown in Fig. 29. [34]

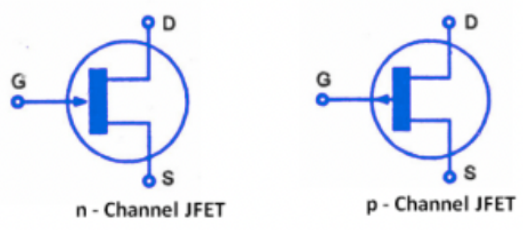

Figure 29

An example of a buffer circuit is the one proposed by AMZFX (see Fig. 30). [35]

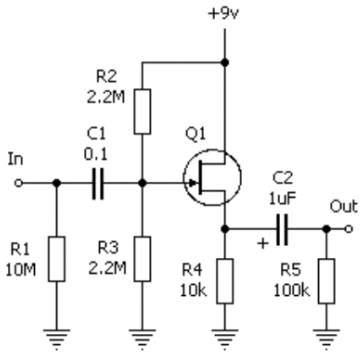

Figure 30

The circuit has a high impedance input, identifiable by the first resistance (R1) of 10 MOhm, and has a low impedance output (about 100 kOhm). As a first test, a test circuit was implemented on the breadboard, a board typically used to make prototypes by connecting electronic components together, before moving on to the actual soldering.

The JFET (Q1) used is the J201, AMZFX also suggests the use of the 2N5457 or the MPF102.

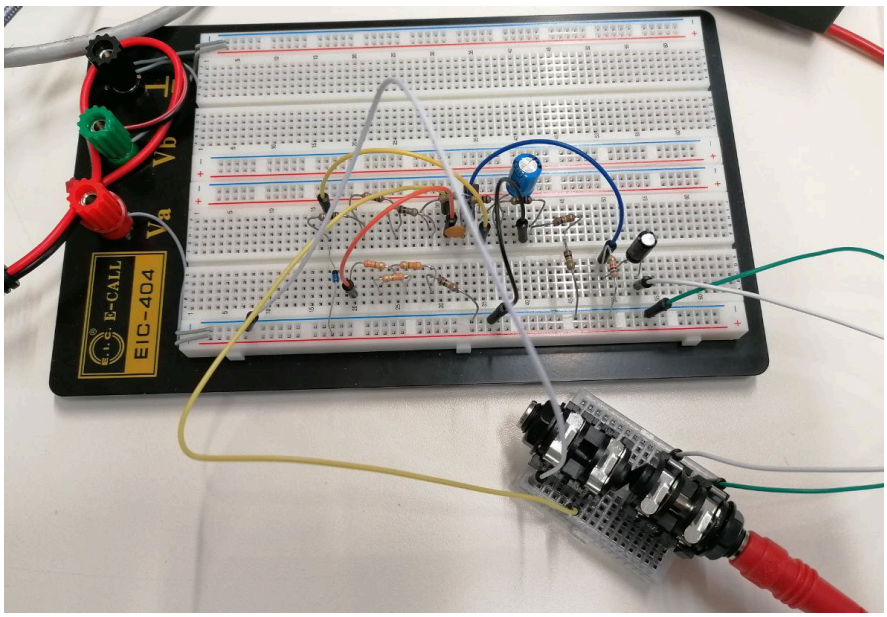

Figure 31

The prototype above was tested by connecting a guitar and comparing the output signal with the input signal (see Fig. 31). The signal is significantly improved, so the buffer fulfils its function. Once this first test was done, based on the AMZFX article [34], the circuit was expanded to be able to use the buffer not only to transform the input signal, but also to split it.
In the article, the circuit has been expanded to have the sound split in 3 but, for this work, it is only necessary to split it in 2, to separate the dry path from the wet one. The last part of the circuit was omitted (see Fig. 32) [34]

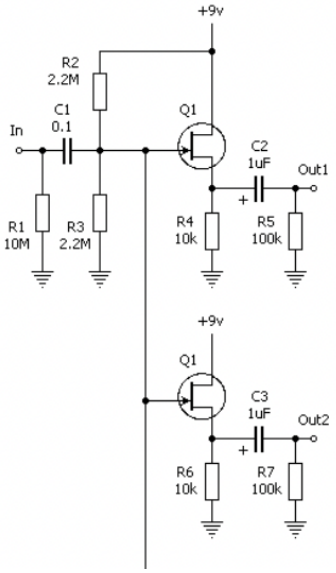

Figure 32

All this concerns only one signal, that is L or R. Therefore, two splitters are needed. Always keeping in mind the desire to obtain a pedal with the dimensions reduced to the minimum necessary, the two splitters were made as small as possible, to minimize the space occupied on the stripboard.
This is the configuration adopted (made twice, for both buffers), developed on Circuit Design DIYLC:

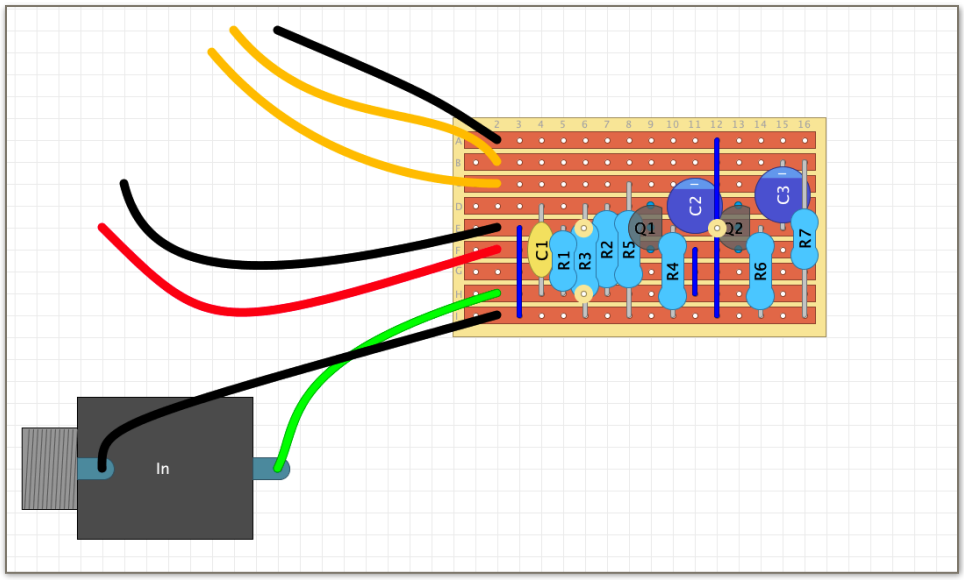

Figure 33

The green cables represent the input signals, the yellow ones the output. The red cable is the supply voltage, the black ones are the ground. The jumpers, or the blue cables, represent the bridges created to connect several lines together. The yellow circles, on the other hand, represent the cuts made in the lines to interrupt the connection.

Except for the relays, everything is soldered on stripboards, a board for permanent soldering of circuits, characterized by holes organized in horizontal lines already connected to each other.

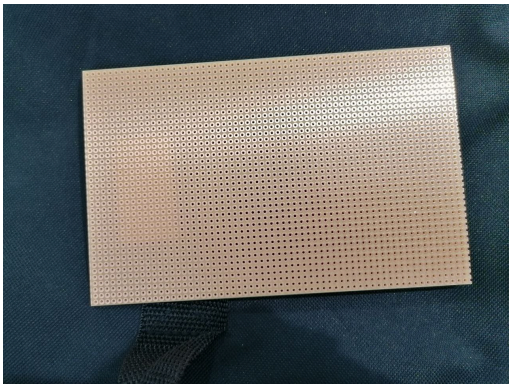

Figure 34

Figure 34 shows the stripboard used. From it, all the circuits were made by cutting it to the desired size.

Figure 35 shows the buffer soldered in the stripboards.

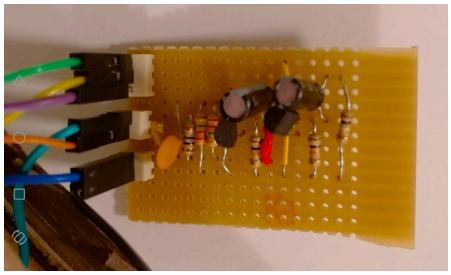

Figure 35

Then, the 6.3 mm jack inputs are connected to the buffer input pins to obtain 2 outputs. One output will go to the Raspberry Pi (wet path), the other will be the dry path, then it will go to the mixer.

6.3 Audio Input Raspberry Pi

The Raspberry Pi has several digital GPIOs (General Purpose Input Output), through which it is possible to connect sensors, potentiometers, relays and similar. But how to get audio input?
Our audio signal is analog, so if you want to use GPIOs, it would be necessary to use an ADC. The same (but inverse) would also apply to the output signal, as the dry and wet signal are mixed into analogue in the final mixer. In this case, a DAC would be needed.

At the same time, however, the Raspberry also has 4 USB inputs, which makes it possible to use an external sound card.
After various considerations and the examination of various similar works, the Behringer U-Control UCA222 audio interface was chosen.

Figure 36: [35]

It features 2 RCA Inputs and 2 RCA Outputs, A/D converters with 48 kHz sampling rate, 16 bit resolution, Optical S/PDIF Output with direct conversion and 3.5 mm headphone output with dedicated volume. Plus, it is USB powered and plug and play.
The audio quality is good enough for the goal of this work, as its use is more intended for live performances.

Given its small size (about 60 mm x 88 mm x 22 mm), the possibility of being powered via USB and its very low cost (around €20), it lends itself perfectly to its integration in this Raspberry Pi-based pedal.

To save space and avoid various clutter inside the enclosure, the RCA inputs and outputs have been unsoldered and male pins have been soldered in their place, to allow connection with the other circuits. The choice of adopting pins to connect multiple circuits together, instead of soldering the connections directly, derives from the desire to have a sort of modular system, which facilitates the implementation of the various modifications.

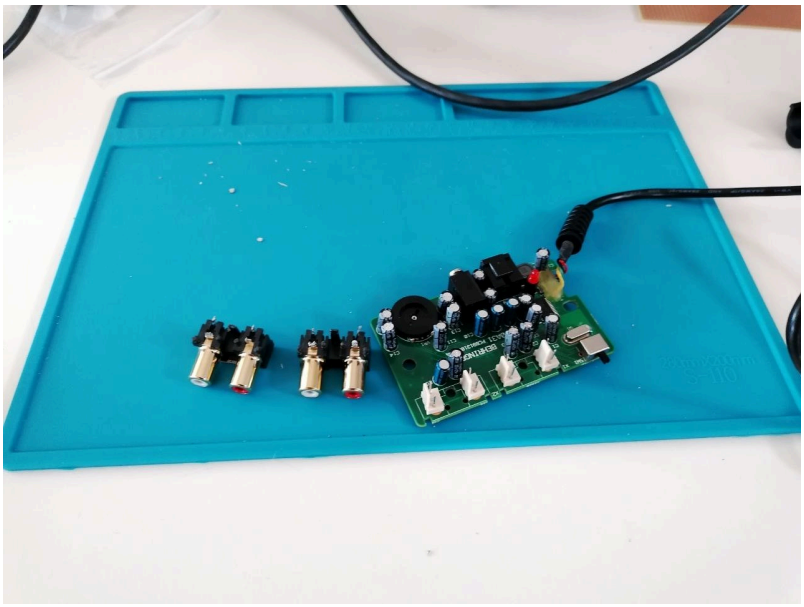

Figure 37: Audio Interface with the old RCA connectors next to it, now unsoldered. The new pins have two poles, one for ground, one for signal

Being *Plug and Play*, the sound card is connected via USB to the Raspberry Pi, where it is immediately recognized, allowing its use. It is used both for the input and for the output of the signal processed by the Pi, which again through the use of pins is subsequently connected to the final mixer, where dry and wet are mixed.

6.4 Mixer

The input signal is initially split to obtain two copies, in order to separate the path. This happens for both inputs, L and R, for a total of four signals, two dry (L and R) and two wet (L and R).

A circuit that fits the need of this work is the Mini Mixer by General Guitar Gadgets. [36] It is a 4-channel mixer with mono output, which allows to control the volume of each individual input and to have a single output for the summed signal.

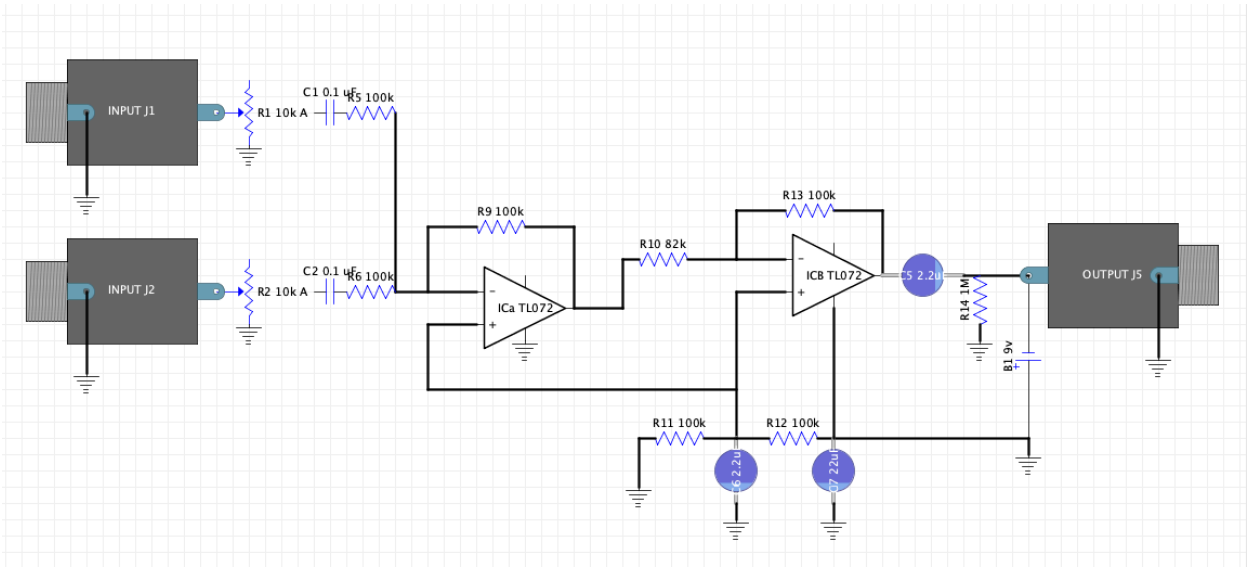

Figure 38

The circuit was adapted to mix only two channels (see Fig. 38). [37] The first test on breadboard was successful. A split stereo audio signal was sent from the computer to the two inputs, from which two oscillators with different frequencies were reproduced, one in the right channel, the other on the left. Using two potentiometers, the signals were mixed as desired.

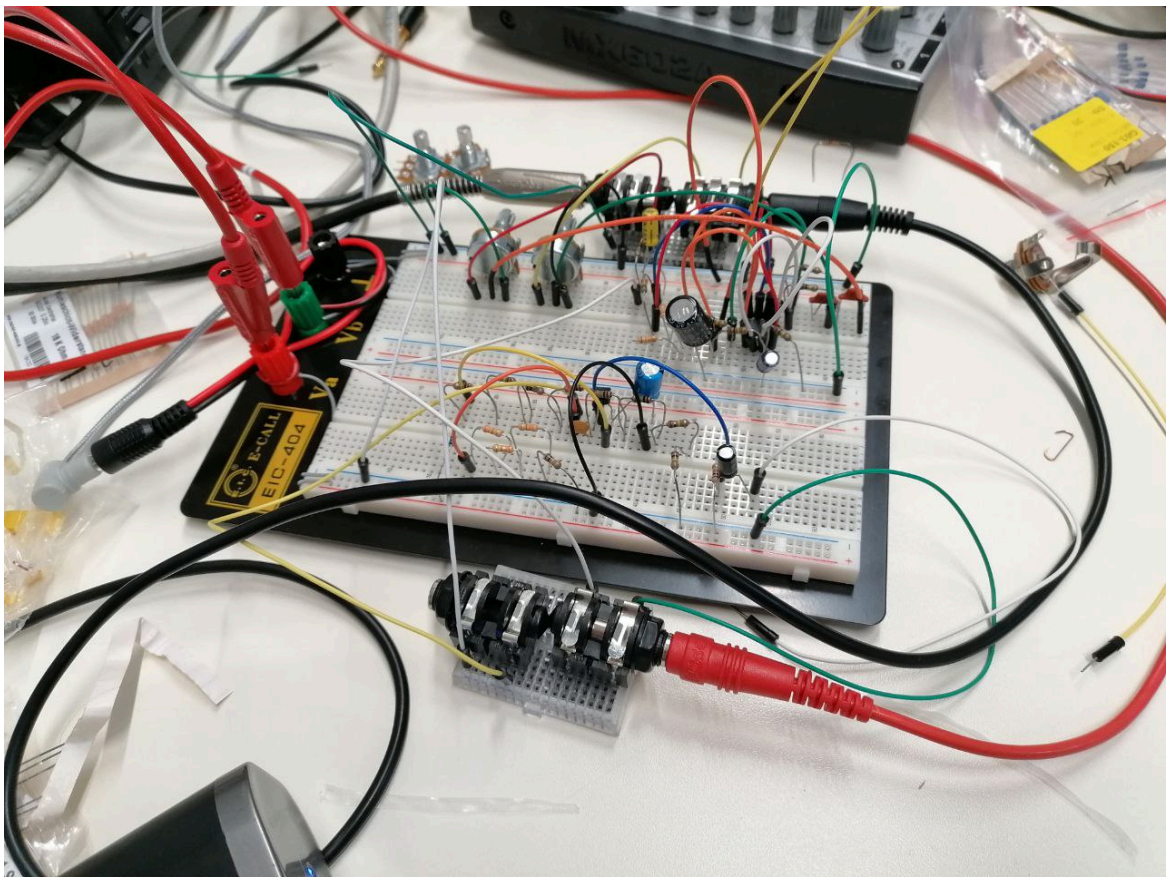

Figure 39: The first half of the BreadBoard, i.e. the lower part of the photo, represents the Mixer, the upper one is the Buffer shown previously.

However, having a total of four signals and still wanting to have a stereo output, two mixers are needed. L dry and L wet will be mixed in one mixer, R dry and R wet in the other. There are only two logarithmic stereo knobs to control both mixers. In the audio field, it is preferable to use a logarithmic scale to measure the volume of a sound, instead of a linear one, as it reflects the way in which human hearing perceives sounds.

A potentiometer is dedicated to the dry signal, o it simultaneously manages L dry from the first mixer and R dry from the second. The other potentiometer will deal with the wet signal instead. In this way, both signals can be controlled in stereo.

Figure 40 shows the arrangement of a mixer on stripboard, minimizing the dimensions as much as possible:

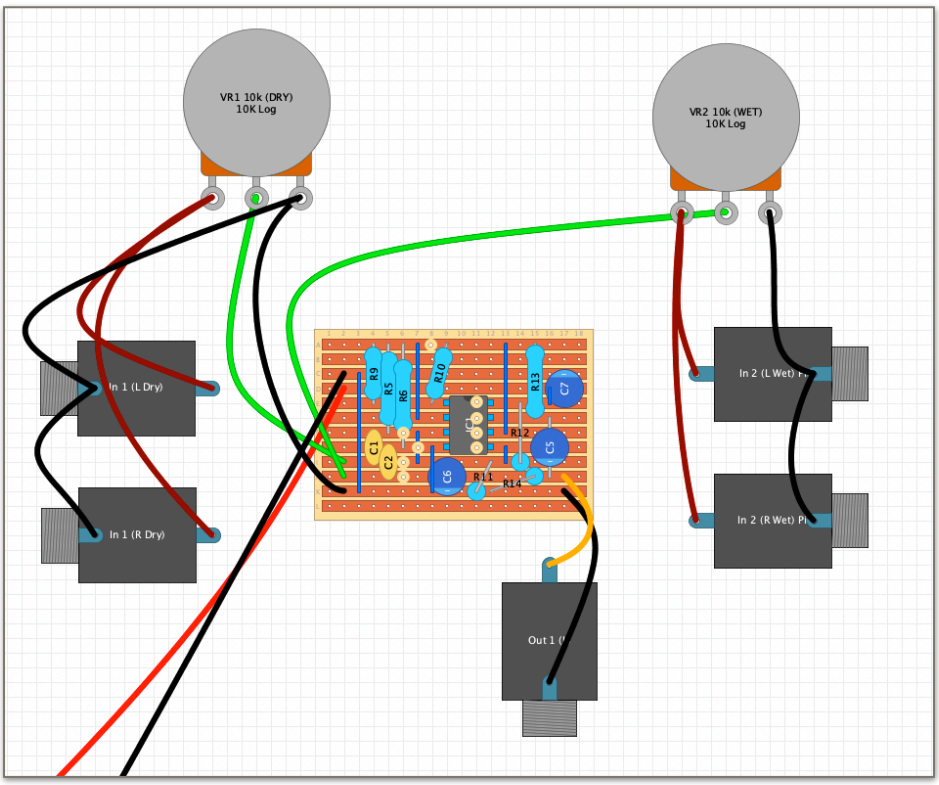

Figure 40

Both In 2 (L Wet and R Wet) are connected to the input pins of the second mixer, which has the same arrangement on the stripboard. However, the potentiometers are actually stereo one, that is, they can manage two signals and send them separately.
Figure 41 shows the mixer with all the components soldered.

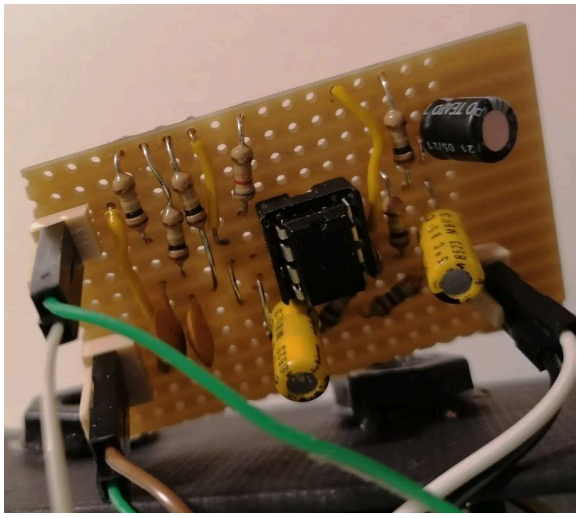

Figure 41

7. First Test

Each of these elements was tested individually when completed and, after various tests, everything worked as described previously.

The voltage for the relay comes directly from the Raspberry Pi, the other soldered parts, namely the two splitters and the two mixers, need 9V of supply voltage. From the Raspberry it is possible to have 3.3V or 5V directly so, for these first tests, the other parts were powered with an external power supply.

The first test was aimed at testing its function as true bypass, checking that, even if the pedal power supply is not connected and, therefore, the Pi is off, the signal flows from the input to the output without interruption.

Having tested this first part, the power supply was connected to the Pi. With the switch in the off position, the signal continues to flow directly to the output being still in bypass mode but, once in the on position, the signal follows another direction.

Through the use of the dedicated potentiometer, it was possible to obtain a 100% dry signal. The WET signal, on the other hand, is recognized by the Pi by selecting the Behringer sound card as the system's audio interface. On PureData the input is recognized using the object [adc ~] (analog to digital converter) and through the object [dac ~] (digital to analog converter) it is sent to the output of the sound card (see Fig. 42).

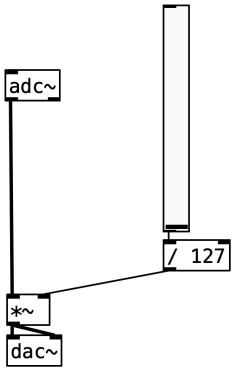

Figure 42

This signal is then sent to the potentiometer for the wet path. This part also worked as expected, introducing no noise or quality loss into the sound.

7.1 Latency Test

Having tested the functioning of the various parts, it is also helpful to measure the latency of the signal processed in the Pi.

Measuring the latency of a signal means measuring the delay in a signal caused, for example, by its passage in a device.
If the latency value is too high, it can become very difficult to play an instrument, as the sound will not be synchronized with the played notes. A latency of up to 10ms is not longer perceptible. [63]

The test took place through the use of a computer with an external audio interface, from which a signal (a sine wave) was sent to the L and R outputs. An output was connected directly to the first input of the audio interface, to make the same signal come in again, the other output, on the other hand, has been connected to the input of the Raspberry Pi. The input signal is received in Pd through the object [adc ~] and sent to the output. The output of the Raspberry Pi is connected to the other input of the audio interface. At this point, the direct signal and the same processed by the Pi return to the computer, in this way it is possible to record both and measure the distance between the two sound waves.

The measured latency corresponds to 12ms, remaining imperceptible and without affecting the functionality of the pedal.

8. 9V

As already mentioned, from the Raspberry Pi it is possible to have only two types of current through the dedicated pins, 3.3V and 5V.
The first option to obtain the 9V needed to power the two mixers and the two buffers could be to use an external power supply. This solution was adopted for the first tests, but the use of two separate power supplies was not too convenient.

The best solution would be to use only one power supply and, therefore, derive the 9V directly from the Raspberry PI. It is therefore necessary to transform the power taken from the Raspberry into 9V. This is possible through the use of a booster converter.

A boost converter stores incoming energy in an inductor and releases it at higher (step-up) or lower (step-down) voltage. [38]

So having 5V in input, it is possible to have 9V as output.
After various researches and tests, the LAOMAO step-up boost power converter XL6009 is the most suitable for this project (see Fig. 43).

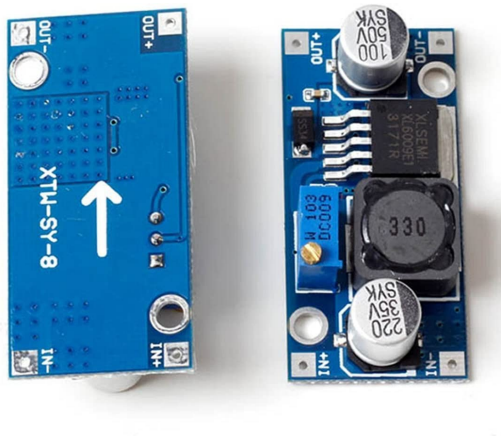

Figure 43: [39]

It is a step-up booster capable of converting an input current (in the range between 3V and 32V) into an output from 5V up to 35V. Being specifically a step-up booster, the output will always be greater than the Input.
The output is adjusted to the desired voltage using the screw on the module.

Once the pins in the two inputs and two outputs were soldered, it was possible to integrate it into the pedal while maintaining its modular characteristic, this means always having the possibility of replacing the part by simply connecting or disconnecting the cable jumpers used to connect the various parts together.
Its small size (4.4 x 2.1 x 1.2 cm) and the low price (about €4) make it perfectly integrated in this work.

9. FX Loop

The last not yet examined part of the signal chains is the one concerning the FX loop (effects loop).
It is placed in the signal path between the two relays corresponding to the active bypass, that works when there is no power supply or when the general switch is in the off position (i.e. the system including buffer, Raspberry PI, and mixer is bypassed).

Its purpose is to be able to connect one or more external effects where the original sound of the input instrument is processed, without it being affected by buffers or mixers.
This could be used in the case of physical modeling, a way to model and simulate systems made up of real physical components. It is a simplified material representation, usually on a small scale, of an object or phenomenon, for analyzing it. [40]
In this way, it is possible to directly compare the software model with the original hardware with just a switch.

The order in which the effects are positioned in a signal chain dramatically alters the sound. An FX loop is usually positioned between the preamp and the power amp and consists of a send and a return. Through the send, the signal from the preamp is sent to the input of the connected effect. Its output is connected to the return, to literally return to the chain and continue towards the power amp.

There are two types of FX loops, serial and parallel (see Fig. 44).

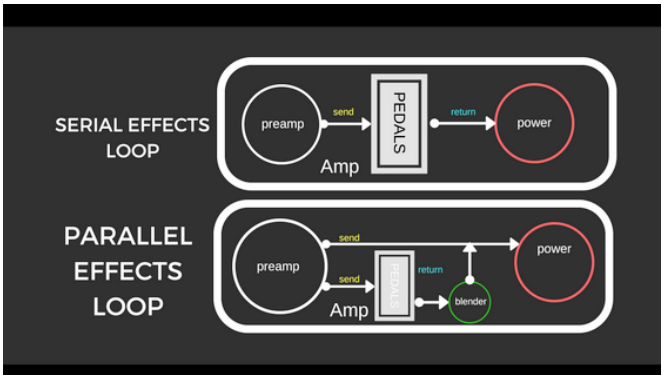

Figure 44: [41]

In most guitar amps, the FX loop is serial, meaning the entire signal from the preamp passes through the FX loop before re-entering the power amp. In the parallel one there are two signal paths which can then be combined using a control knob.
To have a direct comparison in case of physical modeling, it is sufficient to have a serial FX loop.

Having two inputs, L and R, two FX loop would be required, a send + return for L, and a send + return for R.

This implies the presence of four mono 6.3 mm jack sockets, probably increasing the size of a future case. To reduce its size while sticking to the fact of wanting to have the smallest possible product, it is possible to replace the four mono sockets with two stereo sockets. One for send (for L and R) and one for return (for L and R).

In the mono jack plug there is a tip corresponding to the contact through which the audio signal is transmitted, the rest of the plug, the sleeve, is connected to ground. The stereo plug has an additional ring for the second channel.
To connect a stereo effect, Y cables are necessary (see Fig. 45 - 46).

Figure 45

Figure 46

However, it is possible to use this FX loop with simple mono cables, to also connect effects that are not necessarily stereo.
In the case of connection with a mono cable, the signal is transmitted only through the tip of the stereo plug. The central contact is automatically connected to ground, not creating any problems.

The stereo sockets used have six connectors, three on each side. The two sides are connected together, making them three pairs of connectors. However, once the jack plug is inserted, they are no longer connected, resulting in six distinct connectors.

Figure 47 demonstrates the way in which the connection of the send and return connectors was made:

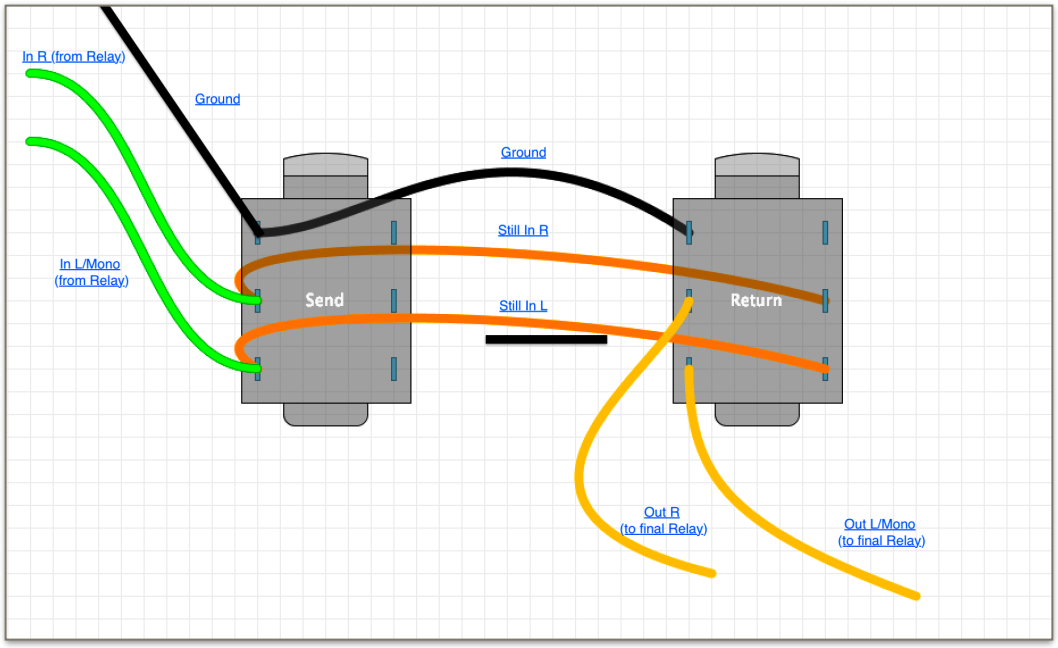

Figure 47

When no cable is connected, the signal passes from send to return and then to output, remaining unchanged.
If something is connected to the send, the signal goes into that cable and therefore to an external unit (pedals, mixer) but at the same time it continues its path towards the return and the output, guaranteeing the same input signal at the output, without any alterations.
By connecting a jack cable also to the return, the signal path of the original unaltered signal to the output is interrupted. In this way, the return will only receive the signal which, through the send, is processed by an external unit.
If only the return path is connected, the signal is interrupted and the output receives only the signal coming from the return socket.
Once the connections have been soldered, the first test took place by connecting an effect pedal in the FX loop. There was no additional noise and, as this is also true bypass, when the pedal is off the original signal is not altered at all. Figure 48 shows the soldered FX loop.

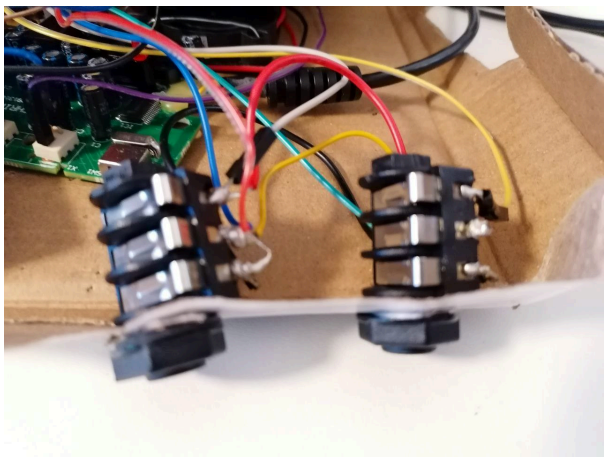

Figure 48

10. Raspberry Pi - Part II

Once the functioning of the signal chain has been tested, it is time to focus on the core of the actual sound processing, namely the Raspberry Pi.

As already mentioned at the beginning of this documentation, this pedal must have the possibility to control the patch/code (therefore the various parameters of the wet signal) through potentiometers. A patch is a code inserted into an executable program, it is the way of coding for example in PureData.
There are already two potentiometers, which control the analog mixer that mixes between the dry and wet signals. These are therefore also used main output volume.
Digital rotary encoders are used to control the code written on the Pi.

The Raspberry Pi has only digital GPIOs. Analog potentiometers would require ADC converters or even a microcontroller such as Arduino.
The rotary encoders, can be connected directly to the Pi's GPIOs, without requiring external converters. They can also rotate infinitely without limit, while most potentiometers can only rotate about 3/4 of the circle.

While potentiometers are best in situations where the exact position of the knob needs to be measured, rotary encoders are best in situations where the relative change rather than the exact position is required.
They also have an integrated push button. This function enriches the functionality of the pedal, expanding its possibilities of use.

But how to have access to the data provided by rotary encoders to be able to use them indiscriminately with any programming language for sound coding?

10.1 Python

My solution was to use the Python programming language and send the data via OSC.
Open Sound Control (OSC) is a message-based network protocol that enables communication and data exchange between software and/or hardware.
In this way the data from Python can be sent to any other program (PureData, SuperCollider and more) and be used.

11. Rotary Encoder

The rotary encoder detects the rotation and direction of the knob connected to it. This device has two internal contacts which, when the knob is turned, create and break a circuit. At each change of position, it is possible to hear a click, which indicates the passage to the next position. [48]

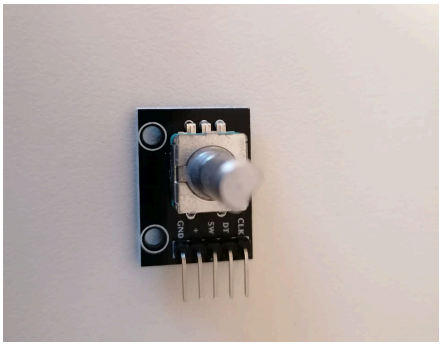

Figure 49: Rotary Encoder

11.1 Pinouts

It has five Pins (the metal ends through which it is possible to connect it to other devices and let the signal flow):

CLK, DT, SW, +, GND. [49]

CLK = This is the primary output that determines the amount of rotation. When the knob is turned one click, this output cycles through two states, it becomes high (completes the circuit) and then low (breaks it).

DT = This is the same as CLK output but with 90° phase-shift. It is used to determine the direction of rotation (clockwise or counterclockwise).

SW = Push button switch output. The voltage becomes low when the button is pressed.

+ (VCC) = The required voltage must be connected here, usually 3.3V or 5V.

GND = Ground.

11.2 How does it work?

The signal coming from CLK tells us that the knob has been turned, and through the signal coming from DT we can determine its direction.

In case of clockwise rotation, the first signal to change is CLK, followed by DT. The reverse situation occurs in the case of counterclockwise rotation.

Clockwise rotation —> Output B different from Output A (see Fig. 50).

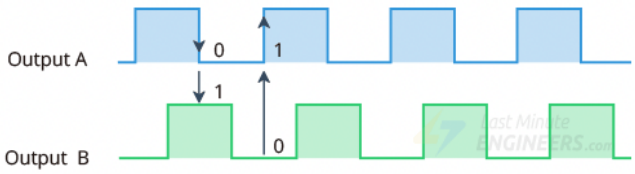

Figure 50 [49]

Counterclockwise rotation —> Output B equal to Output A (see Fig. 51).

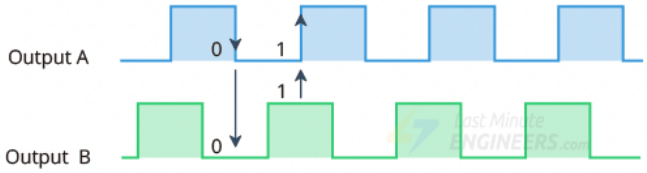

Figure 51 [49]

For the connection with Raspberry Pi, it is important to first have clear numbering, nomenclature and function of its various pins. Figure 52 shows the reference used.

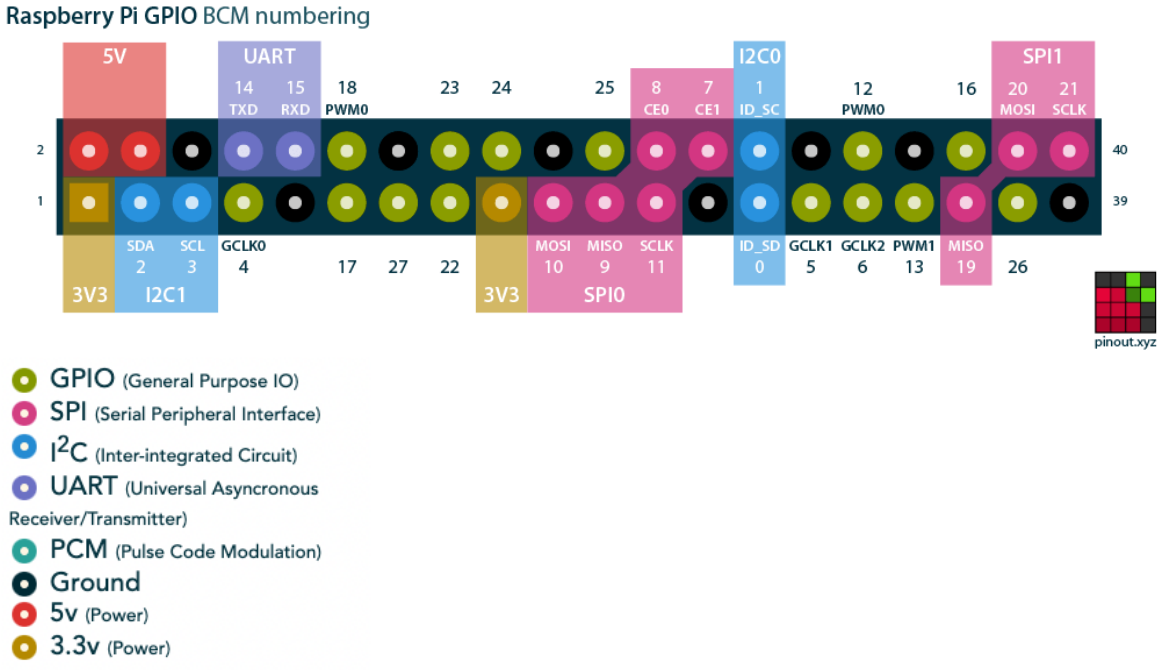

Figure 52: [50]

For designing the connections of the various parts with the Pi, the free app **circuit.io** was used, available on the website https://www.circuito.io/app?components=9443,200000. [51]

In the case of only one rotary encoder, the connection is as Fig. 53 shows. [51]

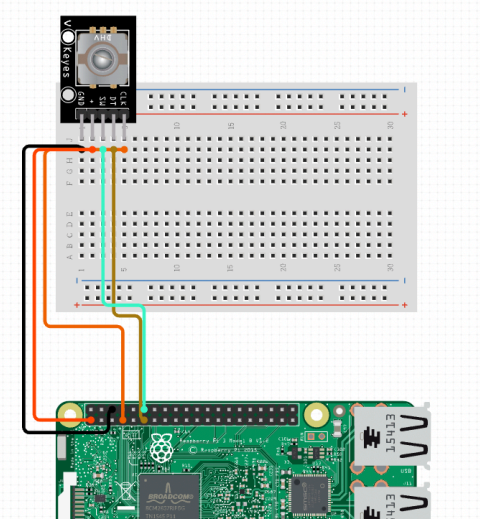

Figure 53

CLK = 4 (GPIO)
DT = 17 (GPIO)
SW = 18 (GPIO)
+ = 3v3 (the first pin of the top row)
GND = the third pin of the top row

The following is the code wrote to access the encoder data and send them via OSC:

```
from RPi import GPIO
from time import sleep

from pythonosc.udp_client import SimpleUDPClient

ip = "127.0.0.1"
port = 5720

client = SimpleUDPClient(ip, port)  # Create client

clk = 9
dt = 25
sw = 11


GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(clk, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(dt, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(sw, GPIO.IN, pull_up_down=GPIO.PUD_UP)

counter = 0
swState = 0
swLastState = 0
clkLastState = GPIO.input(clk)


while True:
        swState = 1 - GPIO.input(sw)
        clkState = GPIO.input(clk)
        dtState = GPIO.input(dt)

        if clkState != clkLastState:
                if dtState != clkState:
                        counter += 1
                else:
                        counter -= 1
                counter = max(min(counter, 127), 0) # Like MIDI, from 0 to 127
                print (counter) # Check
                client.send_message("/rotary", counter)   # Send float message
                clkLastState = clkState

        if swState != swLastState:

                print (f"Button {swState}")
                client.send_message("/button", swState)   # Send float message
                swLastState = swState
                sleep(0.01)
```

11.3 Add-On

The simple logic from above is extended for additional functionality.

A first idea was to receive the data provided by the previous Python code and use them directly in the audio programming languages to obtain new functions or improve the current ones.
The main idea, however, is to have a pedal that is suitable for everyone and for every programming language available on the Pi, so some of these extended features were coded directly in Python, to be able to send the data via OSC and have them immediately ready for use.

11.3.1 Counter according to the rotation speed

The first function considered very useful and which enriches and, above all, facilitates its use is the increase (or decrease) in the counter proportional to the rotation speed of the knob.

The faster the knob is turned, the greater the absolute difference between a value and its next one will be. The slower it is turned, the smaller this difference will be.

The code from above is expanded as follows. While the counter increment was set constant to 1 before, it is now controlled by rotation speed.:

```
while True:
        swState = 1 - GPIO.input(sw)
        clkState = GPIO.input(clk)
         dtState = GPIO.input(dt)
        now = time.monotonic()

         if clkState != clkLastState:
         clkPeriod = now-clkLastTime # measure time since last change
        increment = explin(clkPeriod, 0.005, 0.25, 4, 1) # set increment per tick
#depending on time period between ticks


                clkLastTime = now


        if dtState != clkState:
                counter += increment
        else:
                        counter -= increment

        counter = min(max(round(counter),0), 127)  # round and clip value

        print (counter)

        client.send_message("/rotary", counter)   # Send float message
        clkLastState = clkState
```

New functions must be defined to be able to use them later in the while loop. Defining a function means declaring exactly what the function does and specifying if and where there are variables.
Two functions are here defined: clip() and explin(). [62]

———

clip():

```
def clip(inVal, outMin, outMax):
    return min(max(inVal, outMin), outMax)
```

———

It delimits a value at the two extremes, thus having a minimum and maximum limit.
min() takes the minimum value of both arguments.
max() takes the maximum value of both arguments.

———

explin():

```
def explin(inVal, inMin, inMax, outMin, outMax, clipType='minmax'):
   outVal = ( (math.log(clip(inVal, inMin, inMax)/inMin)) / (math.log(inMax/inMin))
) * (outMax-outMin) + outMin
   return float(outVal)
```

———

This function is used to have a mapping between exponential input and linear output data. It is adopted from SuperCollider.

In the while loop, a new variable *now* is defined, which corresponds to the function *time.monotonic()*, which measures the current time.

clkPeriod measures the time elapsed since the last change. Turning very slowly or after a long time, this variable will be very large, while turning fast it will be very small.

This is where our mapping takes place, where our limits are 0.005s (the peak speed, i.e., smallest time difference between two triggers) and 0.25s (the maximum distance between *clkPeriod* and *clkLastTime*). The output values will be mapped between 1 and 4.
If the time distance between two consecutive counter values is equal to 0.005s or less, the increment will be 4. If it is equal to 0.25s or greater, the increment will be 1. The values located between these two extremes are mapped logarithmically.

11.3.2 Hold

A second function particularly useful is the use of the button, or swState, to recognize if it is pressed for a certain period of time, by sending a message that gives access to a new function, namely the hold function.

The hold function can be useful if you want to trigger a new function of the programmed patch. For example, in programming a delay, it might be useful to have a hold function that allows you to switch to a tap tempo mode, through which by pressing the rotary encoder at regular intervals according to the desired time, it is possible to obtain the new delay time. The same hold function can be used to exit this mode.
In this way, it is possible to have access to more functions with the same switch.

The logic used for programming this part of code is as follows:

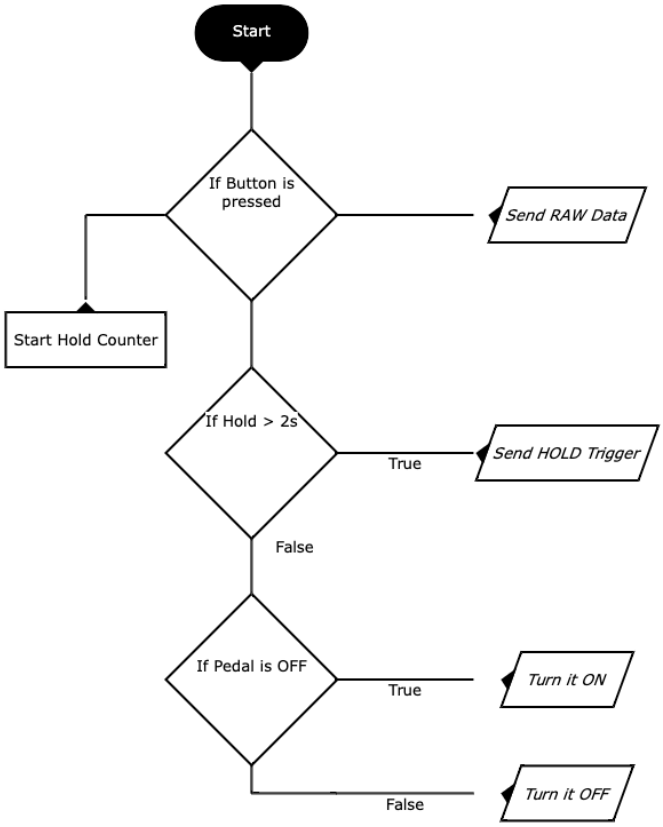

Figure 54

As soon as the button is pressed, the data is sent via OSC so that it can be used directly (raw) if necessary, avoiding sticking to the next part of the code.

At the same time, the hold counter starts measuring how long the encoder is pressed. If it is pressed for more than 2 seconds, a trigger will be sent to indicate hold mode.

Otherwise, the status of the pedal will be checked. If the pedal is currently turned off (off mode), it will be turned on (on mode). If turned on, it will be turned off.
This is useful for activating and deactivating effects with the same switch.

Here the code:

———

```
while True:
    swState = 1 - GPIO.input(sw)
    now = time.monotonic()
    holdDur = 2
    pedalMode = 1

    if swState != swLastState:


        holdStart = now #remember time
        holdStarted = True

        if swState:

            #if pedal is off
            if not pedalState and pedalMode == 1:
                #turn it on
                pedalState = True

            #if pedal is on
            else:
                pedalState = False

        else:

            #abort measuring hold time
            holdStarted = False

    if swState == swLastState:
        #if value is 1 (pressed)
        if swState:
            #if pedal is currently held, but didn't switch mode yet
            if not holdState:
                if holdStarted:
                    #check if pedal is already held long enough
                    if now >= (holdStart + holdDur):
                        pedalState = not pedalState
```

```
            holdState = True
            holdStarted = False


    else:
        if holdStarted:
            #check if pedal is already held long enough
            if now >= (holdStart + holdDur):
                pedalState = not pedalState
                holdState = False
                holdStarted = False

swLastState = swState
```

———————————————————————————————————————

The full code can be found in a Git repository. [42]

In the final code, the three rotary encoders have been defined as functions [r_1 (), r_2 (), r_3 ()] and through the library *threading* it is possible to manage the three simultaneous threads and making them run at the same time.

12. Display

Another important part of this project is the video aspect.

During the programming process of the Pi it is possible to connect it to another computer via LAN cable and, with a VNC viewer program, display the graphic part of the Pi on the screen of the other connected computer.

For programming in a context different from a live performance or for quick use, this is, but in other contexts it becomes a cumbersome process.
It is possible to connect a screen directly to the Raspberry Pi, via the HDMI inputs or the DSI input (see Fig. 55 - 56).

Figure 55: DSI cable

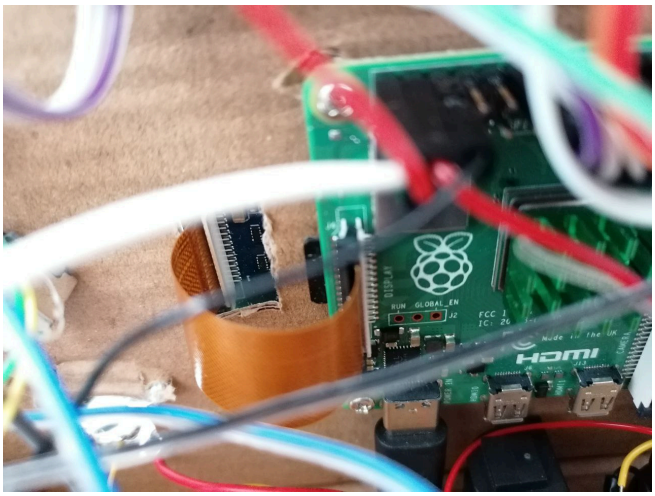

Figure 56: DSI connection to Pi

It is more convenient to adopt a screen that can be connected via the DSI port, as in addition to having an excellent performance, making everything flawless, it does not require an external power supply.

The screen adopted is the Waveshare DSI LCD touch display.
Its 4.3” size in my opinion perfect for this job, not too big, not too small, with a resolution of 800x480 Pixels (see Fig. 57-58).

Figure 57: Front [53]

Figure 58: Back [53]

The four innermost holes are designed to fix the Raspberry Pi (see Fig. 59).

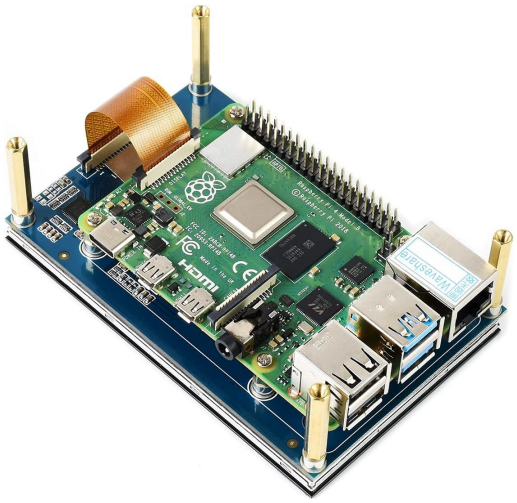

Figure 59: [53]

This solution is very convenient, as the GPIOs are easily accessible and only the unused surface of the Raspberry is covered.

Furthermore, it is a capacitive touch screen, which makes it suitable for performances in which previously programmed effects are used, as it will be sufficient to simply open the desired programs and start own codes.
Connecting a mouse via USB is possible if preferred over the touchscreen.

In a programming or live coding context, however, it is necessary to connect an external USB keyboard. It is possible to connect another screen via HDMI, which can also be used in live context to project the code or any kinds of visuals.

13. Audio in Pi

Once all the hardware has been completed and its operation tested, we can finally focus on audio processing.

For audio processing, PureData was chosen, but the pedal can be used with any programming language or DAW that can be installed on the Raspberry Pi.
PureData (Pd) is an open-source visual programming language, developed by Miller Puckette. [54]

13.1 First Audio Patch

As the first patch, also to better test the functionality of the pedal, a delay was recreated, a typical effect for obtaining a delay of the input signal.

The input signal must be written in a delay line, it will then be read at an indicated time distance and then sent to the output. This is possible using the [delwrite ~] and [delread4 ~] objects. The original signal and the delayed signal can then be mixed as desired by also having a dry/wet control (see Fig. 60).

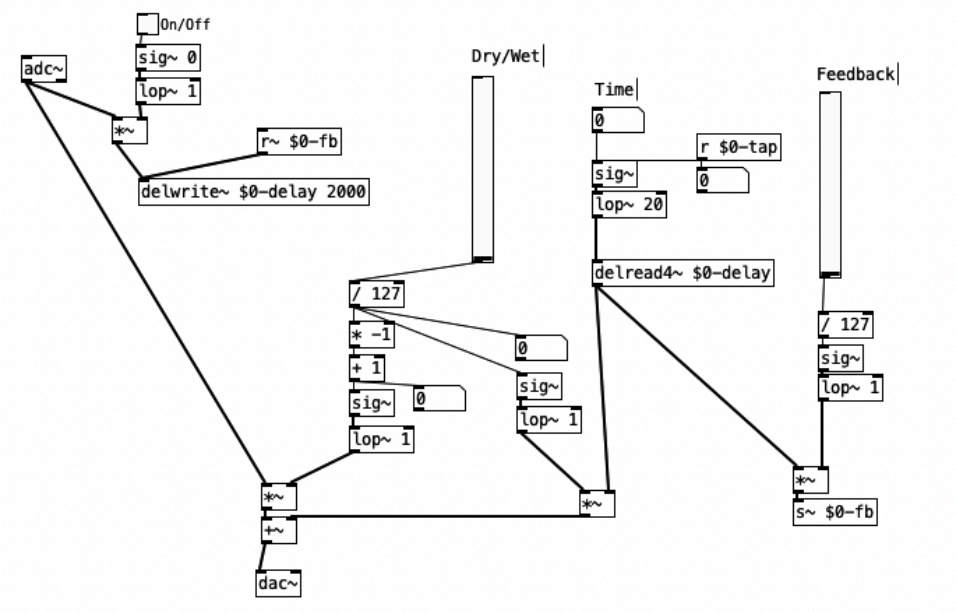

Figure 60

Here a control for the feedback was added, that is the re-entry of the delayed signal, which generates a signal tail until it fades, decreasing in its intensity with each repetition.

13.2 PD + OSC

Once this patch was programmed, some parameters were linked to the rotary encoders.

To access rotary encoder data, the OSC messages sent by Python are read. This is possible in PureData by using the [netreceive], [oscparse], [list] and [route] objects.
Through [netreceive] we receive the OSC data, with the other objects we decode them and make them usable in Pd (see Fig. 61).

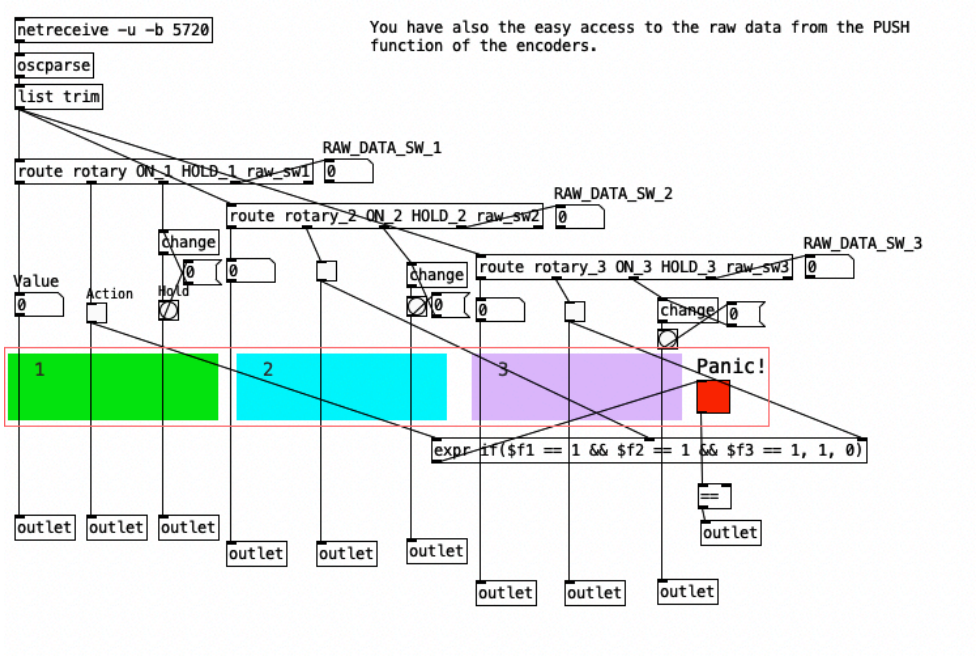

Figure 61

In [netreceive] the same port written for OSC messages in Python is declared and through [route] we have access to the various messages.

Another added function is to be able to have a sort of panic button (red toggle in the picture) that can be activated by pressing all three rotary encoders simultaneously, turning DSP audio off. The audio processing will then be stopped.

Figure 62 shows the patch where the rotary encoders are also used.

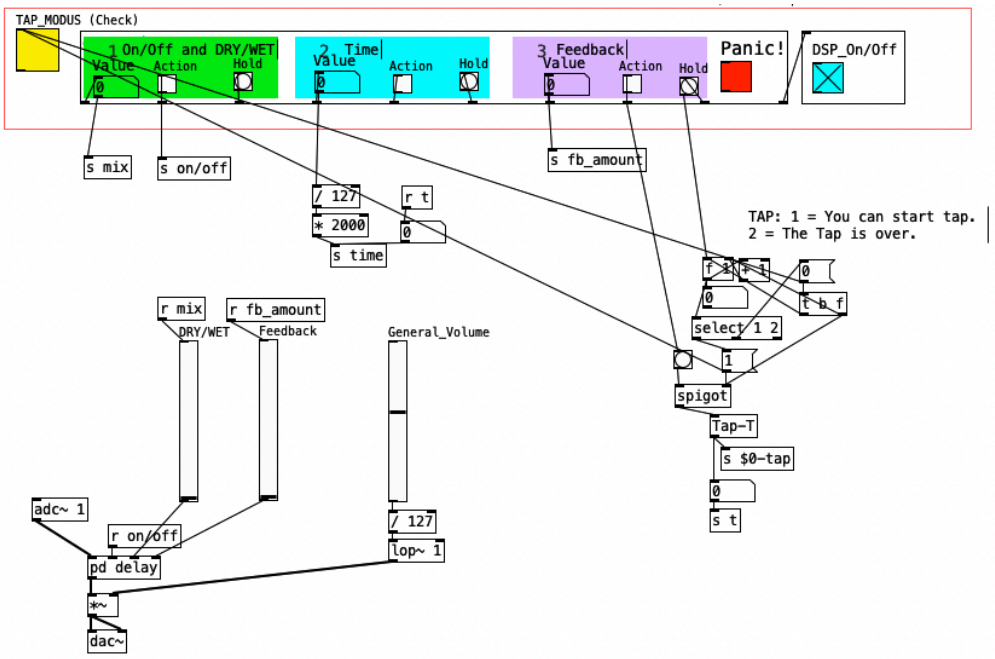

Figure 62

The first rotary controls the dry/wet mix and, through the push function, turns the delay on and off.
The second rotary instead controls the time, that is, the delay time in ms.
The third rotary controls the feedback level and also uses the hold function.
Through the hold function it is possible to switch to tap-tempo mode, that is, once the hold function is active, it is possible to press the third rotary encoder with regular frequency to translate this repetition into ms and set the delay. To exit this mode, simply hold the rotary encoder for more than 2 seconds, sending the trigger hold again. When not in hold mode, the push function of the third rotary encoder has no function.

13.3 More Patching

After having tested the correct functioning of the patch with the three rotary, more effects were coded to also to use the various functions of each rotary.

In addition to the delay, an FM synthesis was added. FM (Frequency Modulation) synthesis is a type of synthesis where the frequency of the carrier oscillator is basically modulated by another oscillator. [55]

In the FM synthesis, there are at least three parameters that can be set, and they correspond to:

Modulator Frequency (fm)

FM Index (or Amount, Δf)
Carrier Frequency (fc)

The idea is to obtain the frequency played with the instrument connected to the pedal (in this case electric guitar) and use it as carrier frequency and set the other two parameters using the rotary encoders.
This is possible in PureData through the [sigmund ~] object, which analyzes the input signal and makes an estimation of the general pitch.

Figure 63 shows the subpatch (patch in the patch).

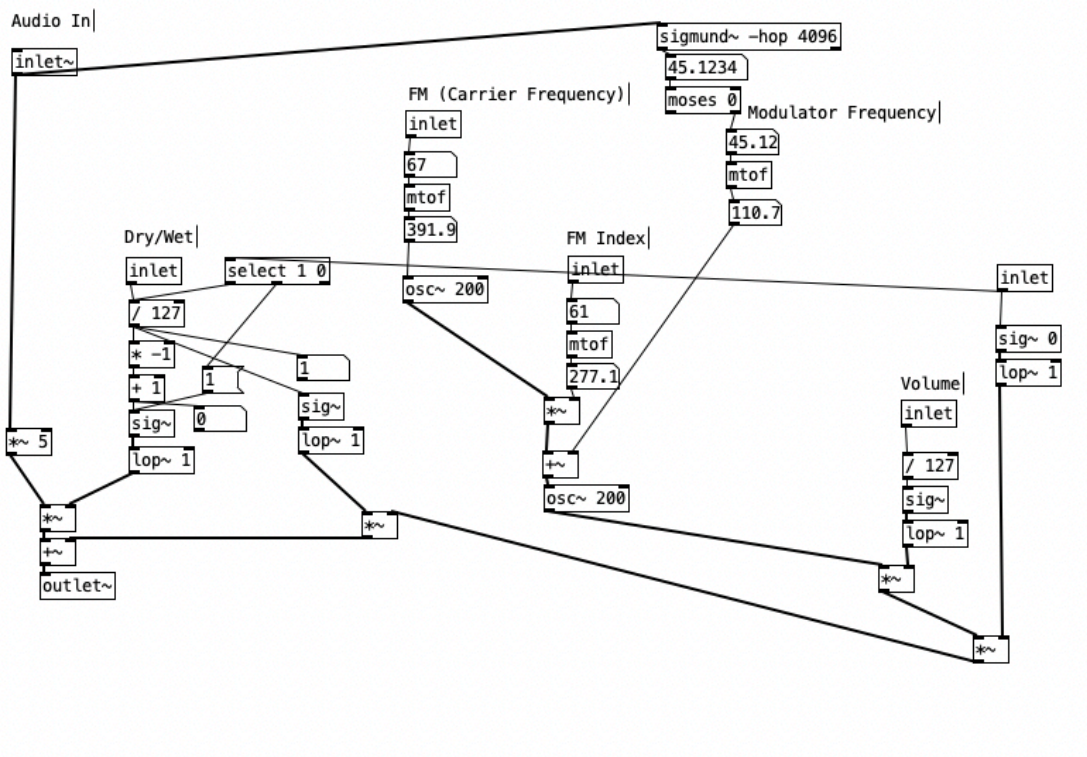

Figure 63

This time all the features of each rotary were used, having access to many patch parameters with the potentiometers.

The first rotary encoder will adjust the dry/wet values and will be used to turn on the delay. However, once the hold mode is activated, the rotary encoder values will be used to adjust the delay time. To return to the dry/wet setting, just exit the hold mode by pressing and holding the encoder again.

The second rotary will turn on the FM (as if it were an FM pedal), and will manage the parameters of fc and, in hold mode, of Δf.

The third rotary keeps the previous tap tempo functionality in hold mode, but its values will be used to adjust the dry/wet of the FM and, in hold mode, the delay feedback.

Being a patch already finished and in any case complex to display on the Raspberry Pi screen, a GUI was programmed.
A GUI is a Graphical User Interface, where only the components necessary to interact with a system are displayed, making it easily accessible (see Fig. 64).

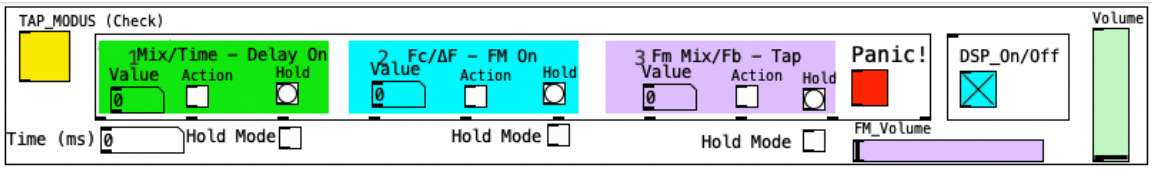

Figure 64: (GUI)

This is, for example, what could be seen during a possible live use of this patch. The various colours serve to differentiate the various sections and give an immediate link on where to act to change the parameters. It has been programmed in a small size also to be positioned at the edge of the screen and leave space for any visuals.

14. Video

14.1 Processing

Processing is an open source programming language developed by Ben Fry and Casey Reas in 2001. [61]

The possibilities are obviously limitless, but at the moment the focus is on creating visuals that react to the sound source processed in the Raspberry Pi, creating something that reflects the waveform generated.

14.2 OSC

Trying to analyze the waveform directly in Processing through external libraries (minim library), it was not possible to process the sound output from the Raspberry Pi, but only the input one.

Since the sound is, in this case, processed in PureData, it can communicate with Processing via OSC. Through OSC, however, it is not possible to send audio stream, but only messages. The audio analysis must therefore be done in PureData.

14.2.1 Pd analysis

It is possible to analyze the audio processed in PureData with the object [fiddle ~] (see Fig. 65).

It analyzes the incoming signal and gives a general estimate of pitch and amplitude. It works through FFT (Fast Fourier Transform). The FFT is an algorithm that allows the decomposition of a complex sound into its sinusoidal components. This allows a more accurate analysis of the sound by obtaining different parameters.

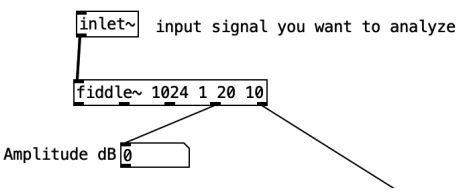

Figure 65: [fiddle~]

Ten sinusoidal components are more than sufficient for the type of visuals to be developed in Processing.

14.2.2 Pd to Processing

Here is the entire patch in which the sound analysis takes place and data is sent to Processing via OSC.

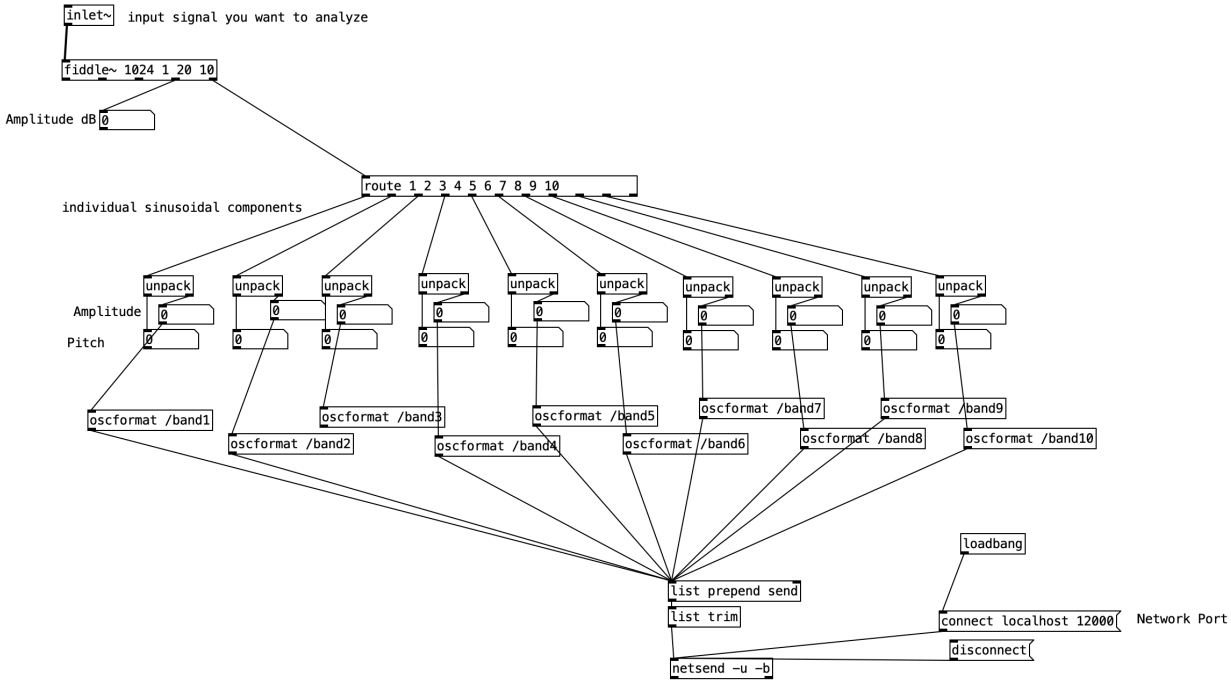

Figure 66: Pd patch with [fiddle~] e OSC

The amplitude data of the various components are sent to Processing.

14.3 Graphic Test/Code

In order to receive (and send) data via OSC in Processing it is necessary to install the external library *oscP5*.

The idea is to create figures that vary in size depending on the sound processed on PureData, creating something that resembles a waveform. Therefore, some ellipses were programmed that correspond to the various sinusoidal components and that change dimensions according to their amplitudes.

Additionally, their colour changes randomly to increase their singularity, creating more engaging images.

The code is very simple, but it is only a basis for developing more complex sound-controlled interfaces. It can be found in a Git repository. [43]

Unfortunately, Processing does not manage a list of messages, so it was necessary to send them individually from PureData and receive them individually to be used separately.

These ten components create a list which is continuously updated, creating an engaging representation of the waveform (see Fig. 67).

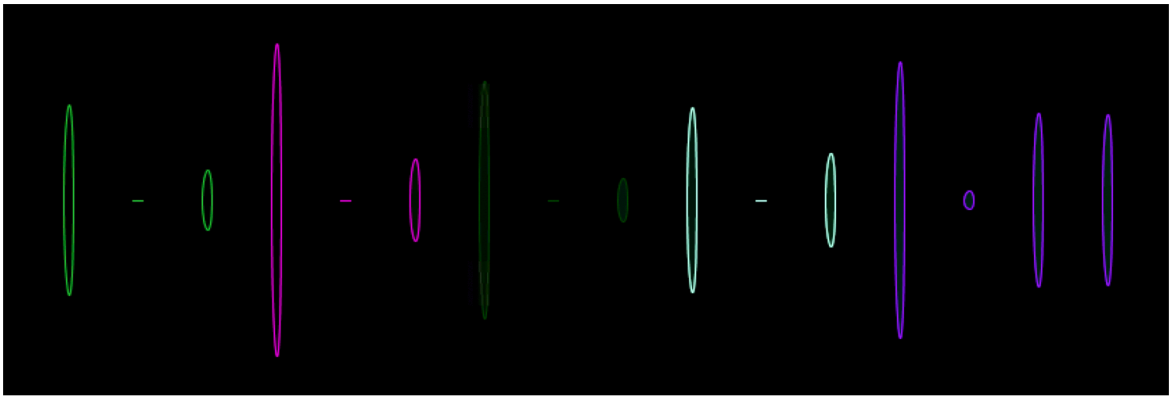

Figure 67: Processing Visuals

Figure 68 shows what can be seen on the Pi screen during a performance.

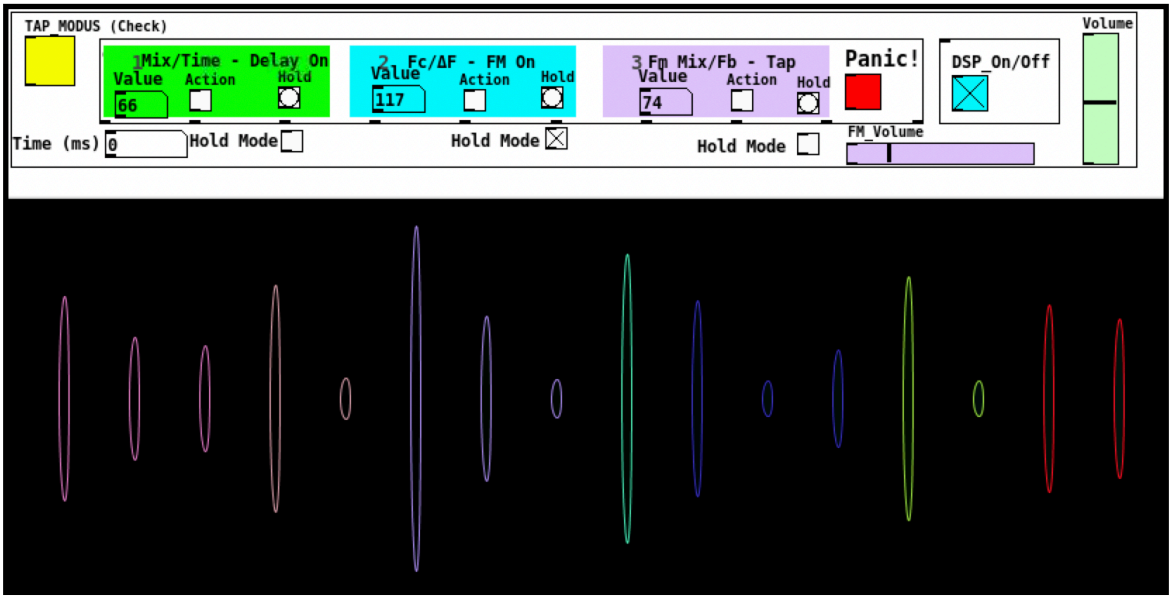

Figure 68: Pd + Visuals

At the top is the GUI developed on PureData, where you can keep the parameters under control, at the bottom instead, their visual representation. It is obviously possible to view

only the Processing window in full screen, but it could be convenient to have the sound patch under control.
Again, this is just a basis, both in terms of graphic and sound processing, just to give an example of what is possible to do with the realization of this pedal.

15. Enclosure

Last piece of this work is the enclosure. It determines its ease of use and its robustness over time.

The idea is to have a case that minimizes the size of the pedal and at the same time makes all its functions accessible. It must allow it to be used both from the floor and from the desktop, in order to adapt to all situations. Considering also the dimensions of the internal circuits and the cables that connect them, a first draft of the enclosure was drawn.

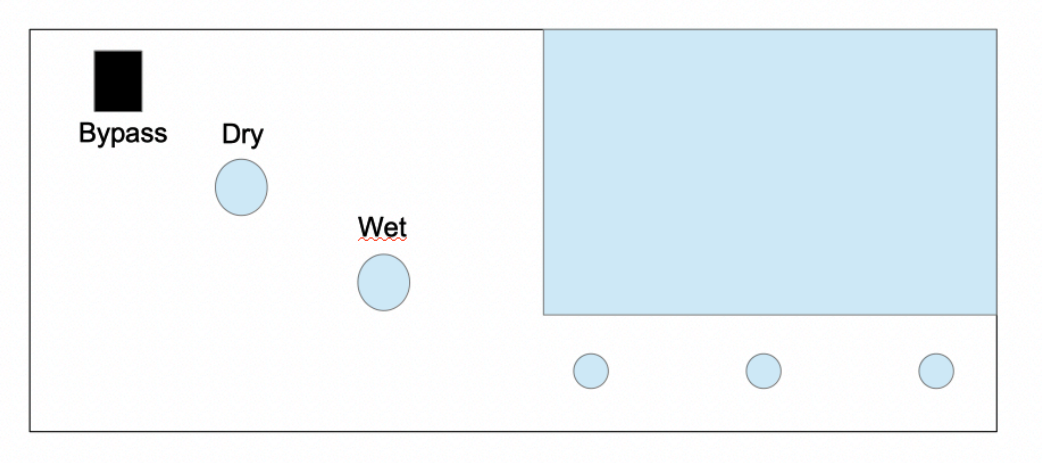

Figure 69

The rotary encoders are positioned directly under the screen to increase usability and to facilitate the interaction with the code/patch. Having a direct correspondence, their use becomes natural and easily associated.

In the opposite corner is located the bypass switch to create contrast and detachment with the other controls, so as to avoid inadvertently pressing it during a performance.

Dry and wet are positioned along the line that connects the bypass switch to the third rotary encoder to create symmetry and to give a sort of continuity to the design. Wet is closer to the rotary and the screen, creating an indirect link with the Pi, to associate it with sound processing. Dry is located closer to the switch to symbolize its affinity for the raw signal.

On the right will be positioned the two inputs, on the left the two outputs, and in the front (but on the left, not close to the screen) the FX loop. The screen is positioned in the corner to make the Raspberry Pi inputs accessible.

All of this has been designed to maximize its usability, visually appealing and functional design.

It is also important to insert a fan in the enclosure to cool the Raspberry Pi. Figure 70 shows the used fan, chosen because it is possible to power it via 5V, obtainable from the Pi.

Figure 70: [56]

The two cables are just soldiered in the same circuit of the relays, which also uses 5V. It is integrated into the case just below the Pi, creating the space needed for air to flow.

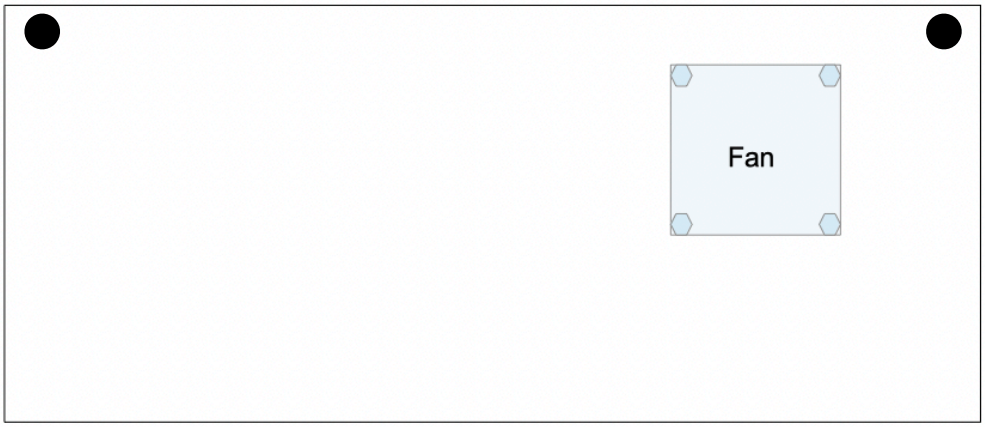

Figure 71

Considering the presence of the fan in the back of the enclosure, two feet could be added to slightly tilt the pedal to facilitate air passage (see Fig. 71).
Here is a first realization, carried out by modifying a robust cardboard box (see Fig. 72 - 76).

Figure 72: Up

Figure 73: Front (Send and Return)
From PI (Power, HDMI)

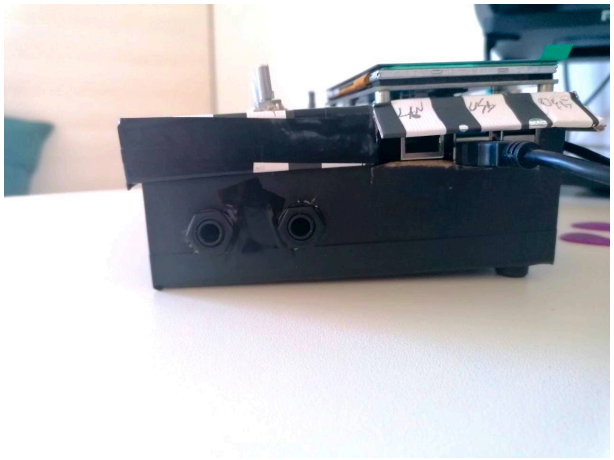

Figure 74: Right (Input L and R)
From PI (USB, LAN)

Figure 75: Left (Output L and R)

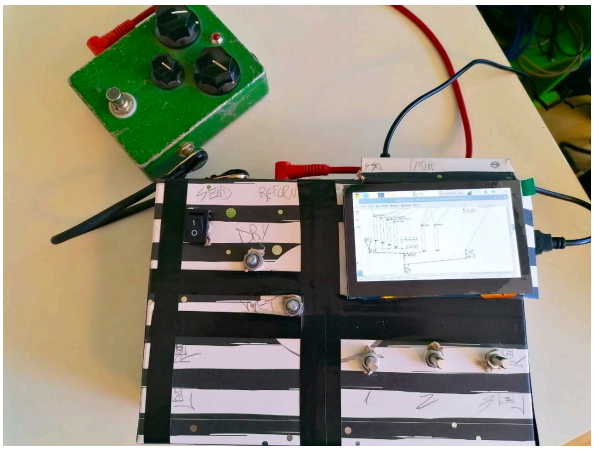

Figure 76: External pedal in FX Loop,
PureData

Later a 3D model was developed using the cloud-based 3D modeling platform Fusion 360 (see Fig. 77 - 82). [64]

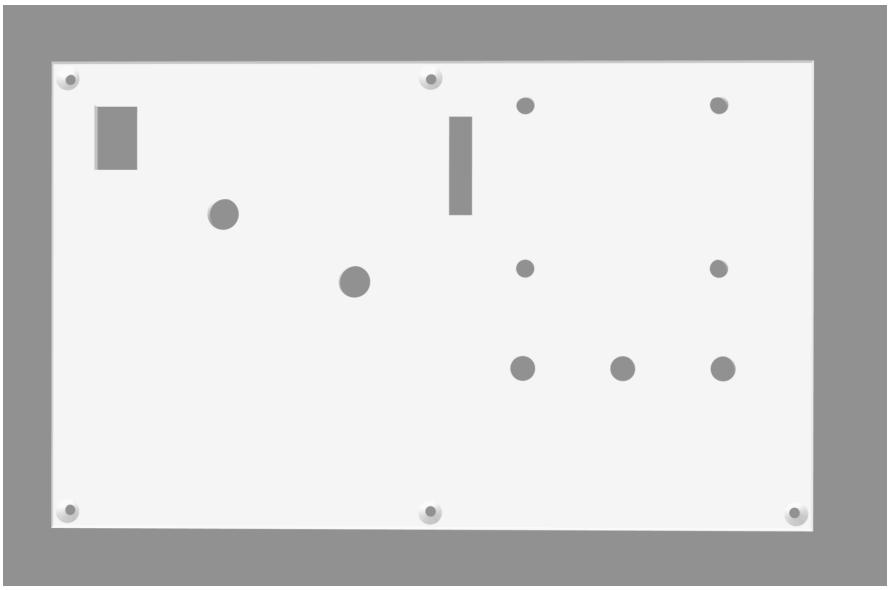

Figure 77: Cover (with holes also for screws)

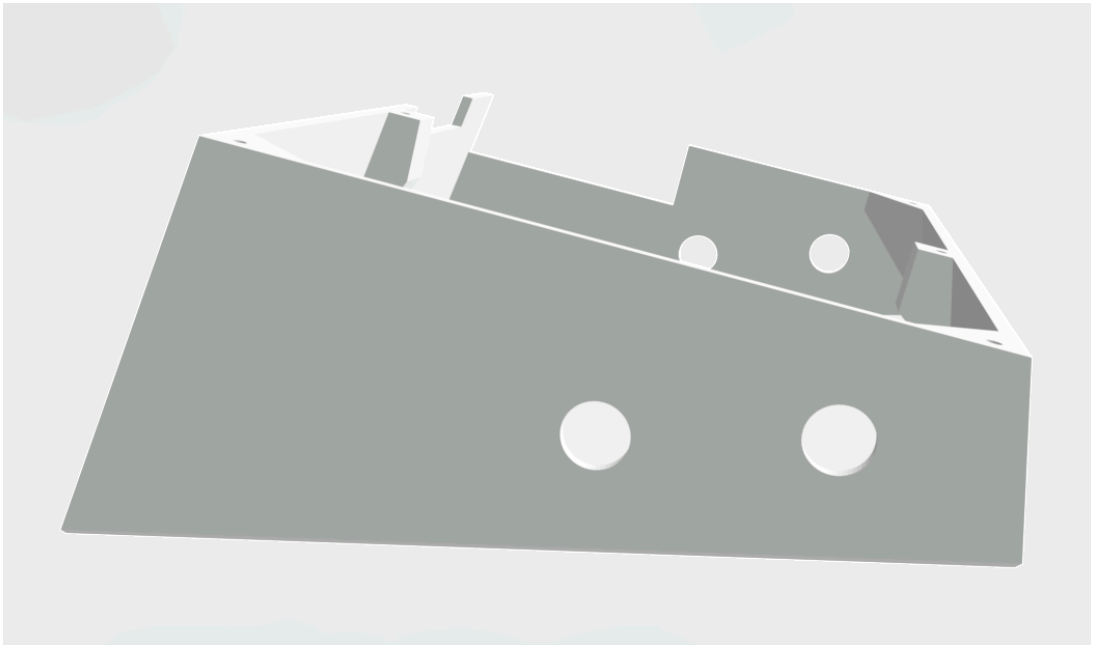

Figure 78: Side L

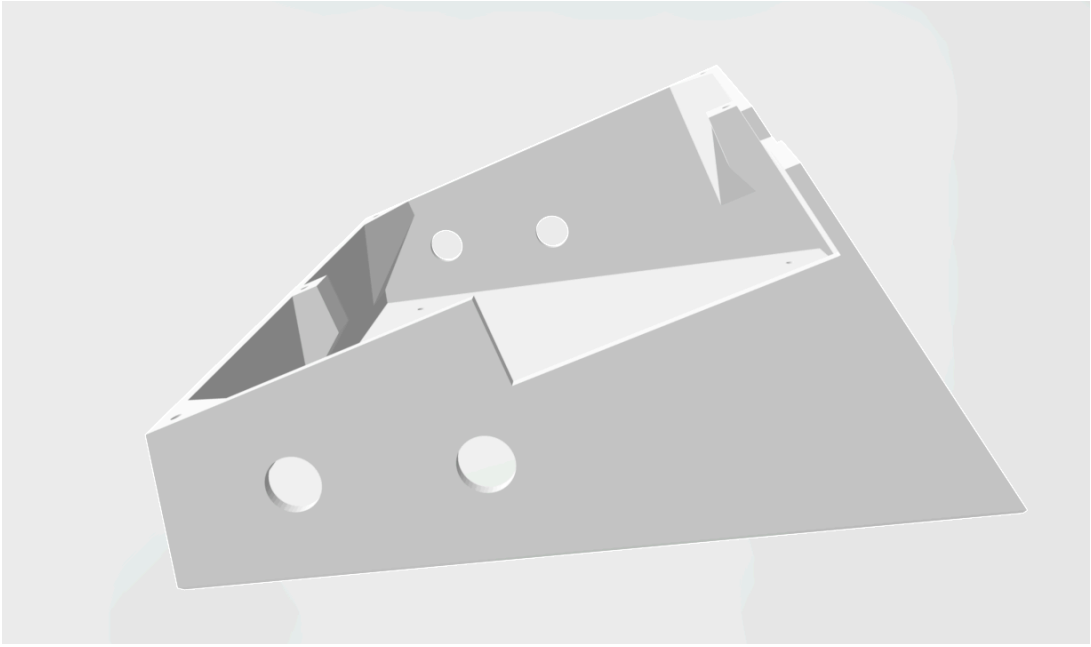

Figure 79: Side R

Figure 80: Front (with extra hole for USB)

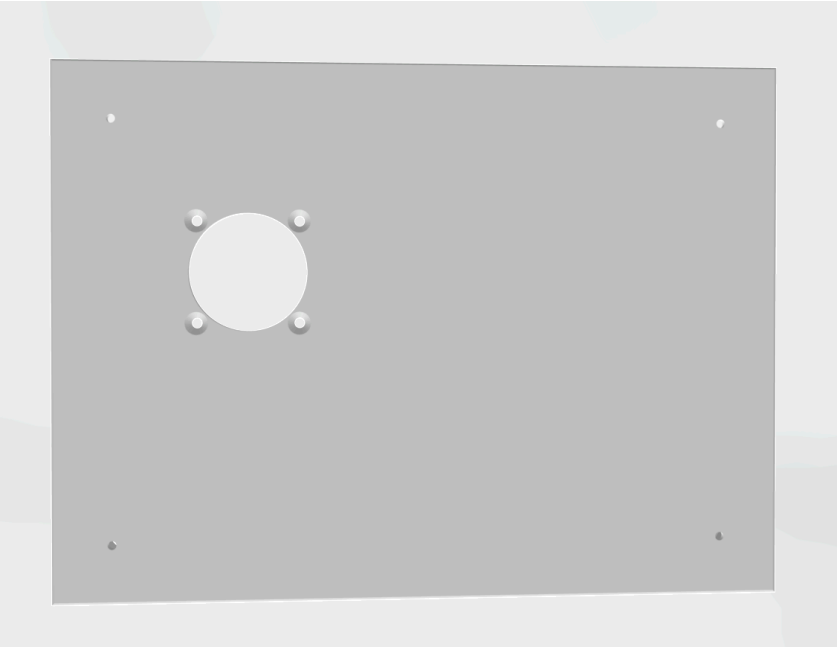

Figure 81: Back (with holes for 4 feet)

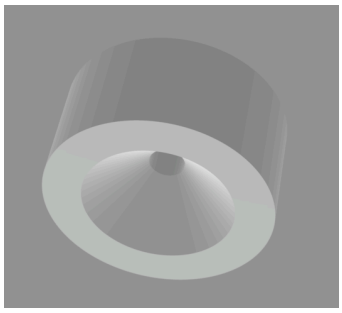

Figure 82: Feet (with holes for screws)

It is similar to the first draft but has an inclined shape, making it even more suitable for a desktop function, improving accessibility to the various controls and facilitating interaction with the screen. It also has an extra hole for the USB cable, so that it can reach the Pi exting the enclosure. This can be attached to the front after use using velcro.
The upper part is the only one that can be unscrewed to access the enclosure where all the components will be positioned.

The material chosen for the printing was ABS (Acrylonitrile Butadiene Syrene), for its lightness and consistency. Figures 83-88 show some photos of the final pedal.

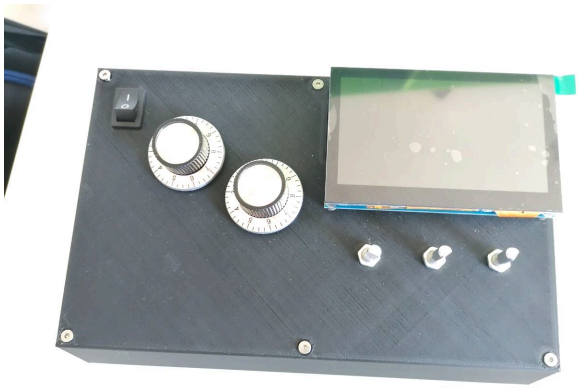

Figure 83: Print Up

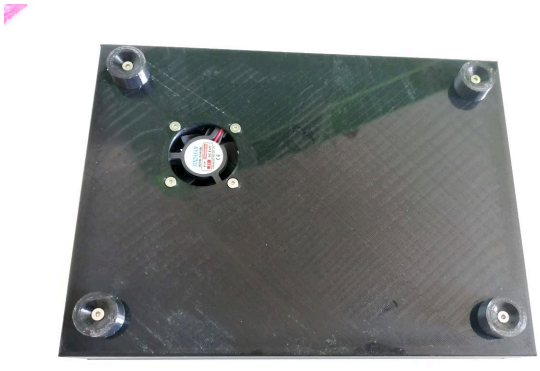

Figure 84: Print Back

Figure 85: Print Left

Figure 86: Print Right

Figure 87: Print Front

Figure 88: Print Front 2

16. Preparation for User Study

Since the pedal is ready to be used, a user study was carried out, in order to obtain feedback on its various aspects, necessary to make any future changes or improvements.

Some PureData patches were programmed to give an idea of what can be achieved through this device.

16.1 Patching

Four Patches have been prepared, from the most basic to the most complex that take advantage of more and more features of the pedal.

16.1.1 Patch 000

The first patch only provides for the amplification of the input signal, with the possibility to adjust the output volume (see Fig. 89).

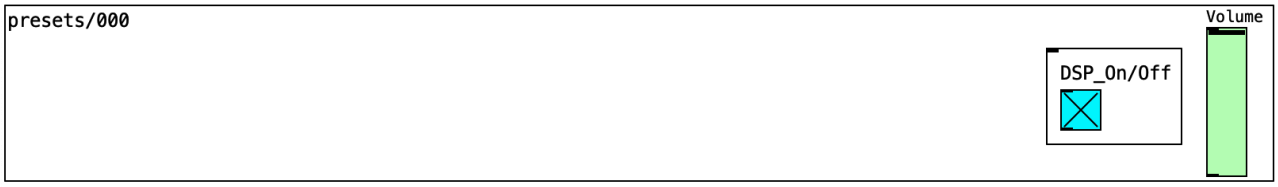

Figure 89: Preset 000

16.1.2 Patch 001

The second patch begins to be more complex. A vibrato has been programmed, for which it is possible to control the speed and depth parameters by means of the two rotary encoders (see Fig. 90).

The first rotary encoder will activate and deactivate the effect by pressing it.
By pressing the third rotary encoder, on the other hand, it is possible to activate a fuzz of which, again through the same encoder, it is possible to adjust the amount of gain.

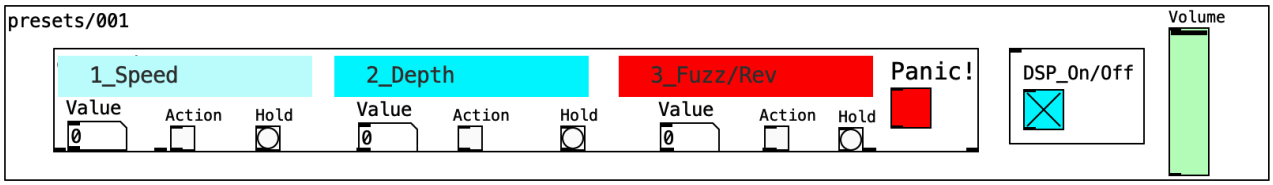

Figure 90: Preset 001

16.1.3 Patch 002

With the third Patch, the hold function begins to be used (see Fig. 91).
The first rotary encoder activates and deactivates a step-vibrato based on the patch developed by Pierre Massat in his blog Guitar Extended. [65]
It is a trapezoid modulator which, playing only one note, reproduces three different ones, creating a sort of arpeggio.
Through the first two rotary, it is possible to control the speed and depth parameters.
Instead, using the hold function activates the mode here called "Swirl!", That is an automation of the two parameters that starts from a minimum value and grows to a maximum value and then goes back, continuing in a loop until the mode is active.
The third rotary encoder activates and deactivates a reverb, of which it is possible to adjust the dry/wet and, through the hold function, it is possible to adjust the decay.

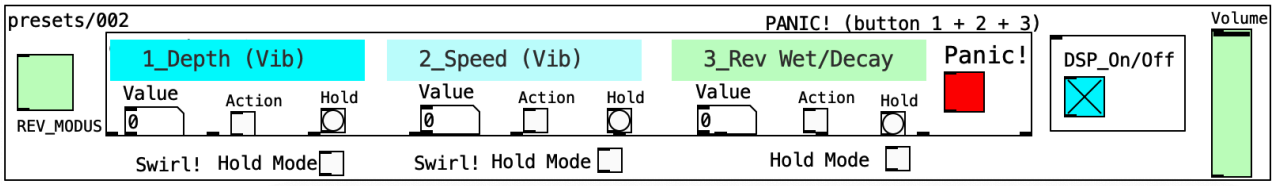

Figure 91: Preset 002

16.1.4 Patch 003

The latest patch is the same used as a test of the various functions in the chapter 13.3.
It is the most complex, as it uses all the functions of the rotary encoders plus the touch screen to adjust some parameters (see Fig. 92).

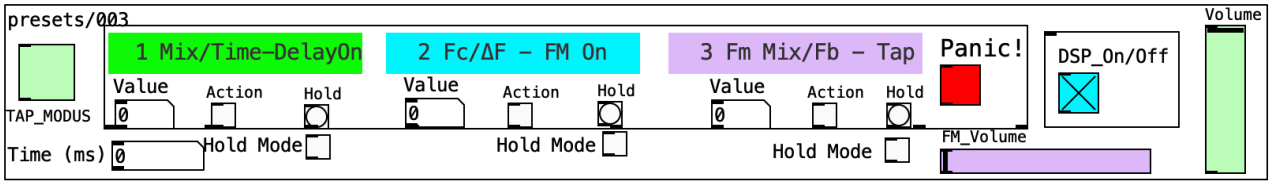

Figure 92: Preset 003

16.2 Presets (Dynamic Patching)

Almost every digital or multi-effect pedal allows the user to select presets. This functionality has therefore been recreated in PureData, to allow simple switching between presets without having to manually open and close the various program windows each time.

This was achieved through so-called dynamic patching, that is, to use Pd messages for automatically editing the patch itself.

Presets, stored as self-contained Pd patches are thereby dynamically loaded into the main patch by selecting the respective preset number (see Fig. 93).

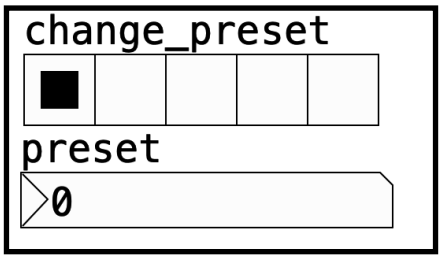

Figure 93: Preset change

When changing presets, the DSP is turned off and on to avoid additional noise.

16.3 Start everything with the Pi (Python, Pd)

It is convenient to start the python code and the main GUI for selecting the presets in Pd with the booting of the Pi, in order to have everything easily and quickly accessible for each use.

The python code (for rotary encoders) runs on boot via CronTab. At each boot of the Pi will be started the service Cron, which allows executing scheduled commands. [66]

To autostart PureData, the procedure is slightly different. Here, the autostart functionality of the LXDE-pi window manager is used. [67]

16.4 More visuals

Since there are four audio patches, four video patches for the graphics were coded as well. All of them react to sound.

16.4.1 Sketch 000

The first one used is the one described in the chapter 14.3.

16.4.2 Sketch 001

This code generates white bubbles on a black screen, which float upwards (see Fig. 94). They get bigger and start shaking according to the dynamics of the sound.

Figure 94: Visual 1

16.4.3 Sketch 002

On a grey background, coloured stars are generated in different points and of different sizes depending on the sound frequencies (see Fig. 95).

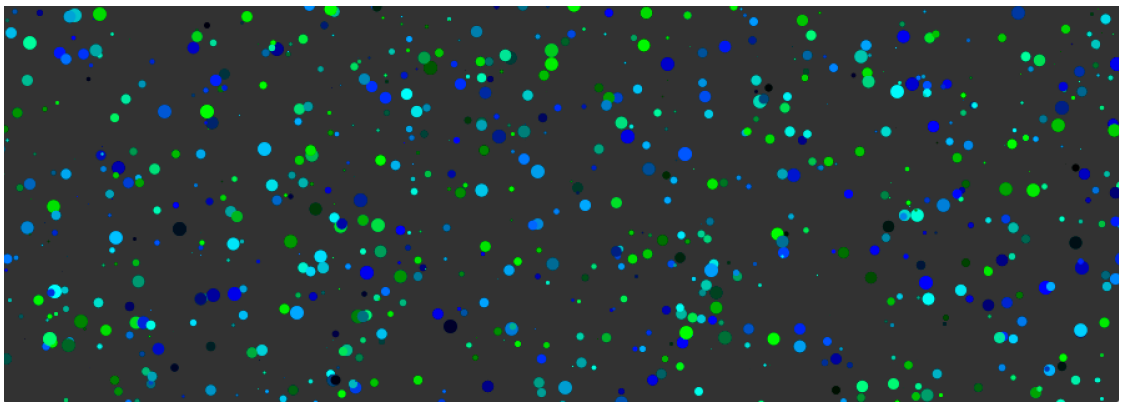

Figure 95: Visual 2

16.4.4 Sketch 003

The last code generates a smoke effect that follows the sounds (see Fig. 96). It randomly moves according to the frequency and to the dynamic of the sound.

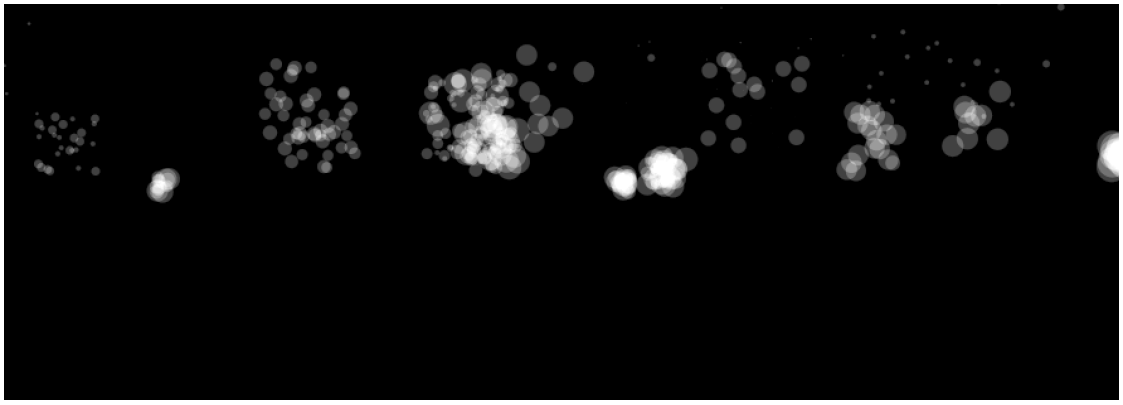

Figure 96: Visual 3

16.5 Preset visuals

Since the PureData patch through which the various presets are started at boot, it would be quite practical to use it to change the coded visuals as well.

In pure data, through the object [shell], part the external library *ggee*, it is possible to send executable commands directly to the terminal on Linux.
In processing it is possible to export the codes as applications, in this way it is possible to start the sketches directly from the terminal, without having to open the code and run.

Using a bit of the same logic to change audio presets, visuals are also alternated (see Fig. 97). Through the object [shell] the command *killall -java* is first sent to close all open java applications (the processing applications) before starting a new one. This way, there will only be one visual open at a time.

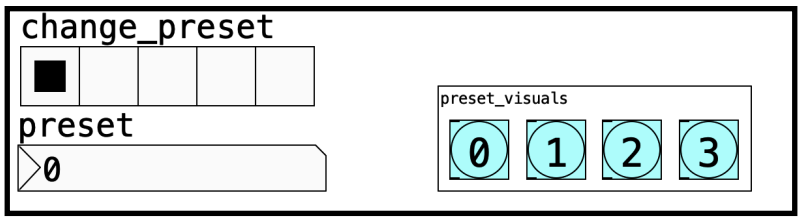

Figure 97: Presets GUI

The dimensions of this window are already set in Processing (810 x 320), but unfortunately it is not possible to set the position of the sketch once started.
Wanting to have the PureData GUI at the top and the visuals window at the bottom, it would be very practical to make sure that they are always started in the same position. In PureData this is easy, as the window is launched in the last saved position.

In Processing, on the other hand, this is done via the terminal with the command *wmctrol*, also window manager control, which allows to move a window that is already open, as well as adjust its size and decide whether to bring it to the foreground or to the background.

This command is saved in a shell script, so that it can be executed from the terminal using [shell]. It runs 2.5s after the sketch is started, which is the time the window takes to open. Another bang is sent to execute this command 2s after the first, so that the position of the window can be adjusted in case the sketch takes longer to start.
Figure 98 shows an example of full GUI.

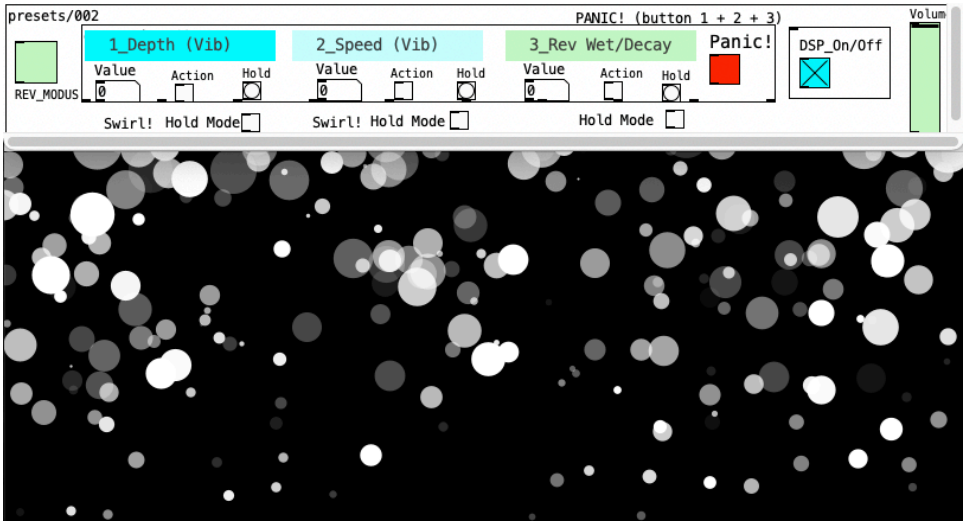

Figure 98: Full GUI

16.6 Patch for Physical Modeling

The last part of the user test concerns physical modeling.
Using the patch developed during the physical modeling course in winter semester 2021, where the famous overdrive Boss DS-1 was modeled, another preset was prepared that also allows the use of rotary encoders.

Its purpose is to make the user compare, via the bypass switch, the patch with the real pedal connected in the FX loop. With this comparison is it possible to directly make changes to the patch, being able to readily compare it with the real model.

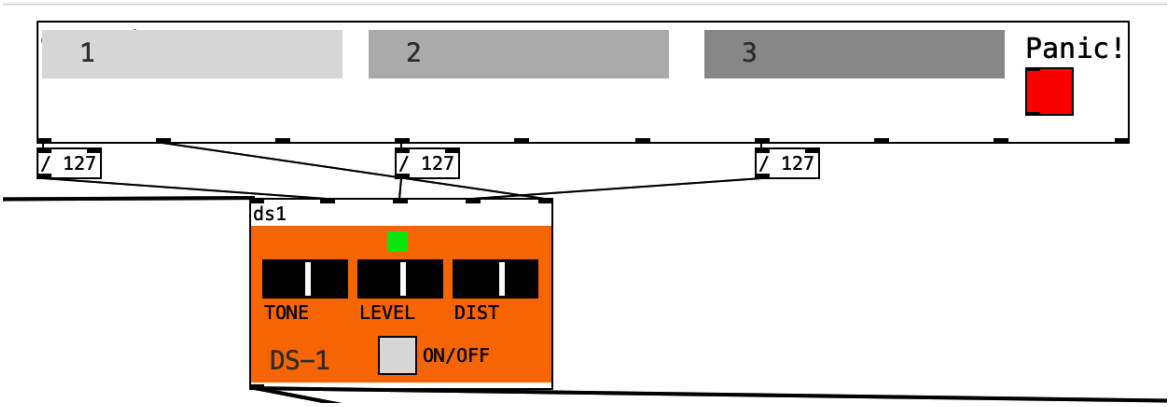

Figure 99: Patch Boss DS-1

17. User Study

The patches described in the previous sections are designed to demonstrate a little bit of all the various pedal functions and give the participants of a user study an overview of what can be achieved with this device.
A questionnaire was prepared to be filled out after the test. The questions focus a little on the user's background (playing skills, number of effects owned) and then on the various features of the pedal, both from the audio aspect and from the design aspect.
At the end, it is asked what price is reasonable for this pedal, if there are any missing features, and if something needs to be changed or improved.
The evaluation was carried out with 10 people, with different playing and programming skills recruited from colleagues and personal acquaintances.

17.1 Results

As shown in Fig. 100, half of the participants judged themselves at least as advanced guitar players. Only one had no guitar playing skills at all.

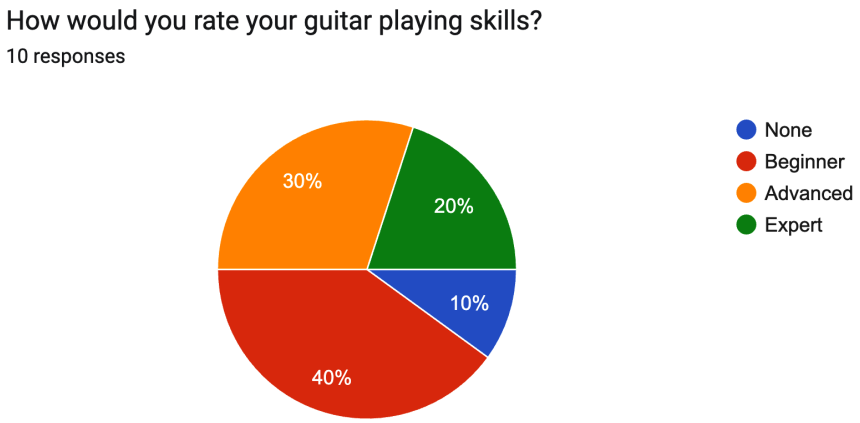

Figure 100: Evaluation Question 1

Concerning programming skills it was the other way around (see Fig. 101), with only one expert and 3 participants with no knowledge at all

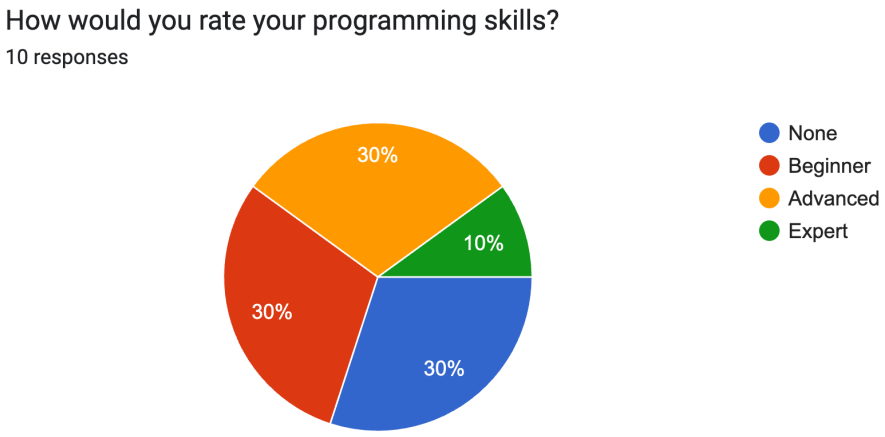

Figure 101: Evaluation Question 2

As shown in Fig. 102, 7 participants own at least one effect pedal.

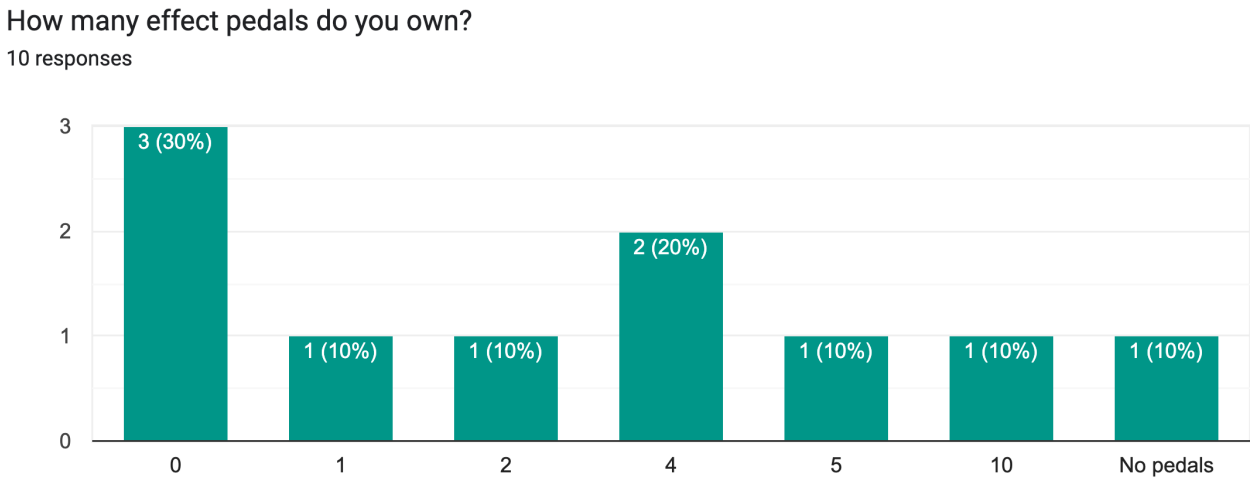

Figure 102: Evaluation Question 3

The participants were also asked how it felt when playing with the pedal. 70% gave 6 of 7 possible points (see Fig. 103), two gave 5, one even 7 points.

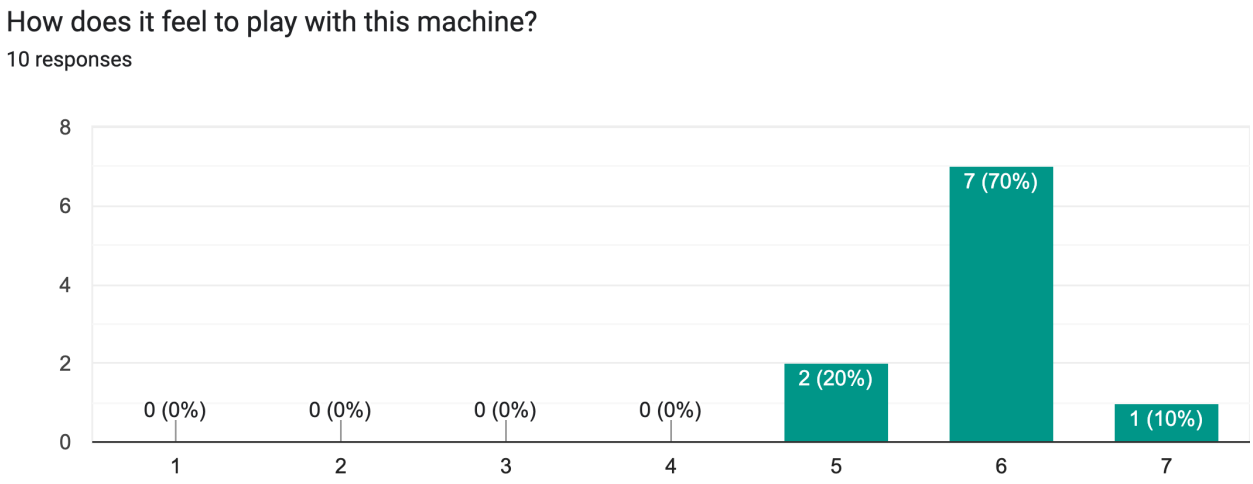

Figure 103: Evaluation Question 4

The visual impression of the device was rated with 7 by 50% of the participants, 6 by 50% and 5 by 10% (see Fig. 104).

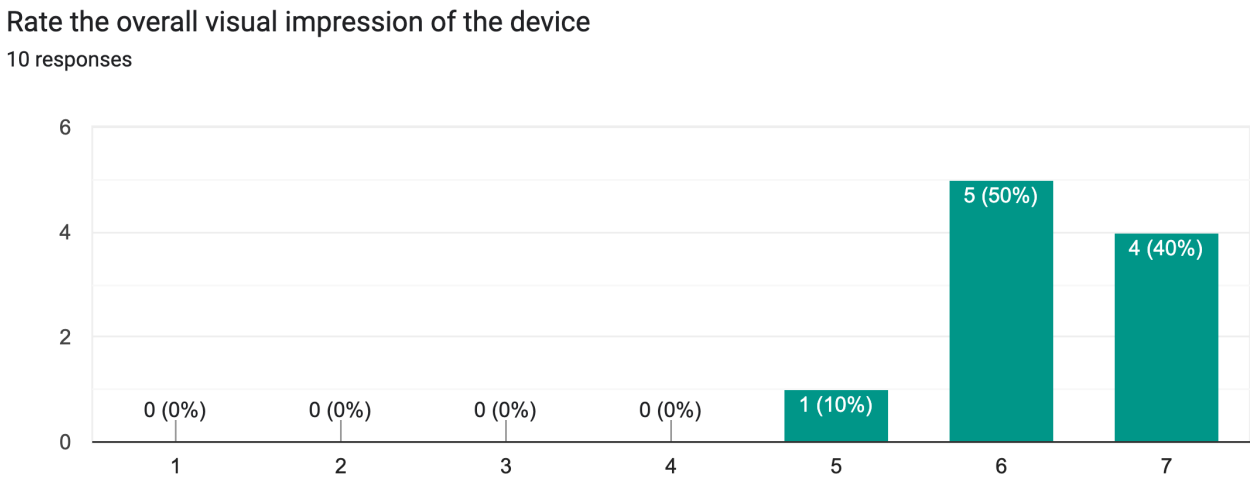

Figure 104: Evaluation Question 5

Figure 105 shows how the haptic experience was rated. 40 % gave 7, 30% 6, 20% 5 and 10% 4.

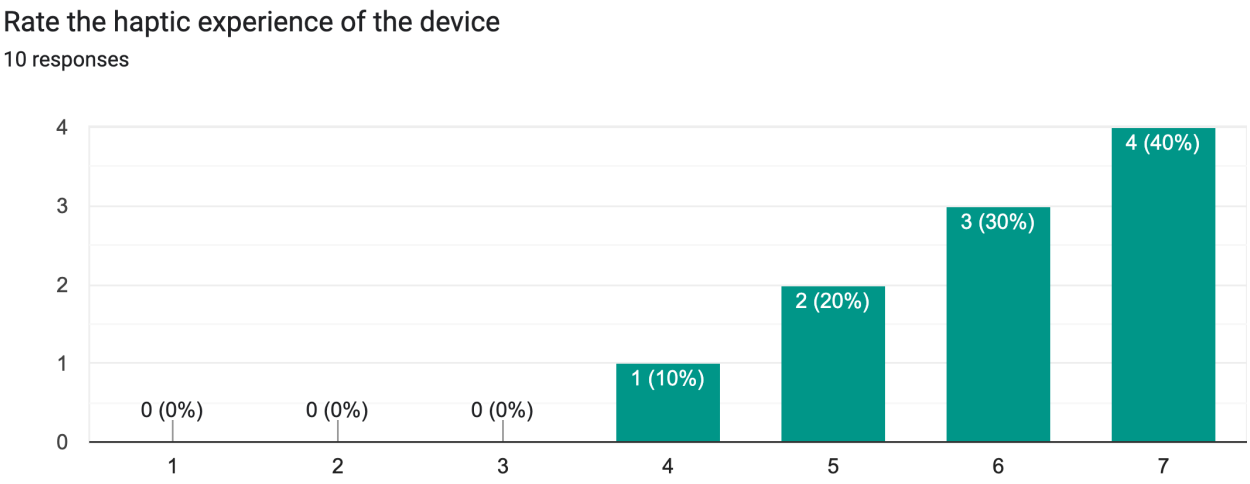

Figure 105: Evaluation Question 6

As shown in Fig. 106, 1 participant perceived some latency. Half of them did not perceive it at all, and 4 of them gave at 2.

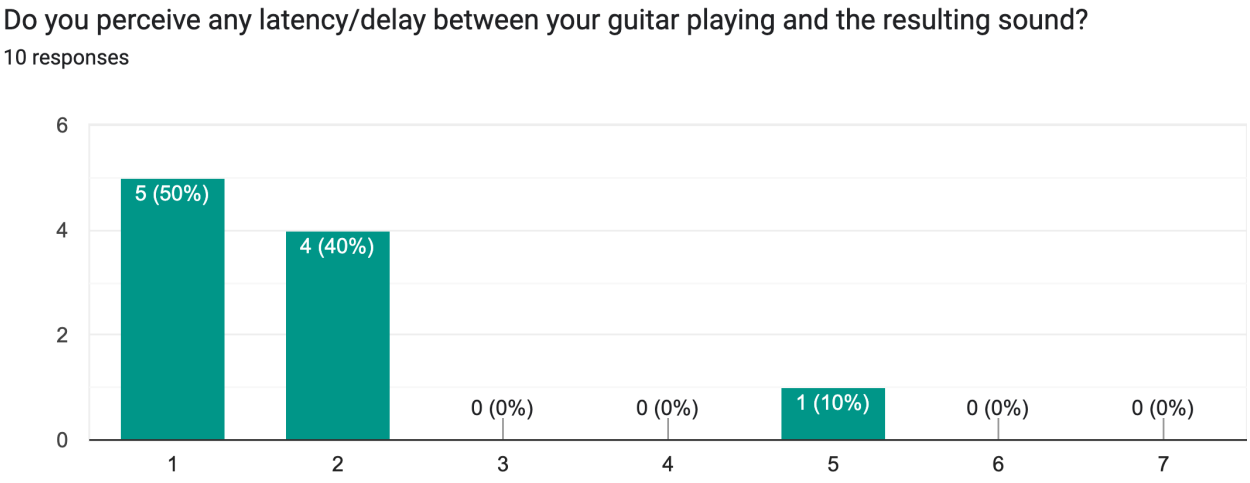

Figure 106: Evaluation Question 7

Figure 107 shows if the participants would use the pedal for a live performance. 4 of them gave 7, 1 gave 6, 3 gave 5 and 2 gave 4.

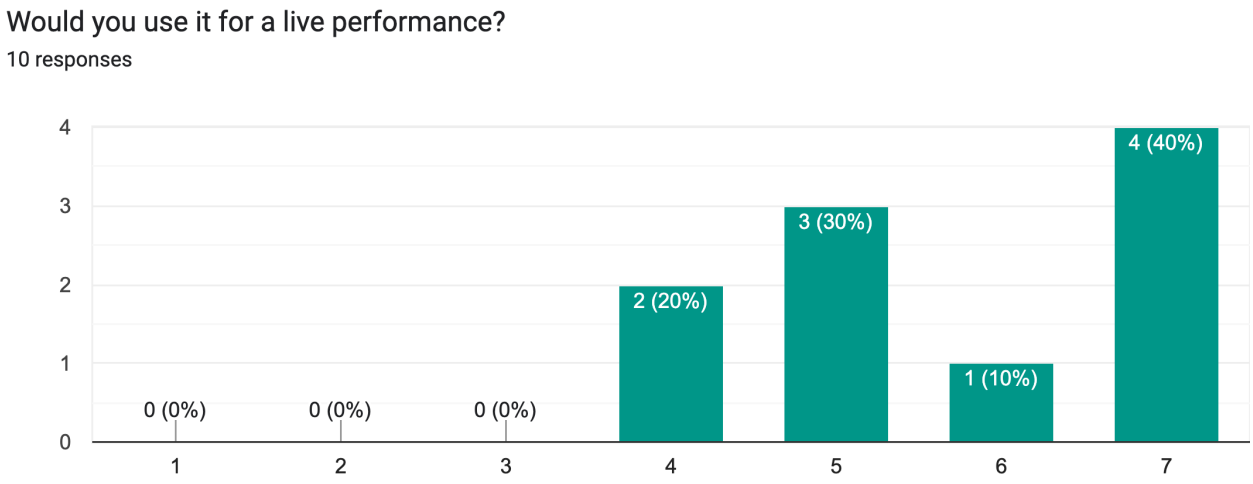

Figure 107: Evaluation Question 8

The participants were also asked if they would use it as their only live gear. 50% gave 5, 20% 4, 10% 6, 10% 3, 10% 2.

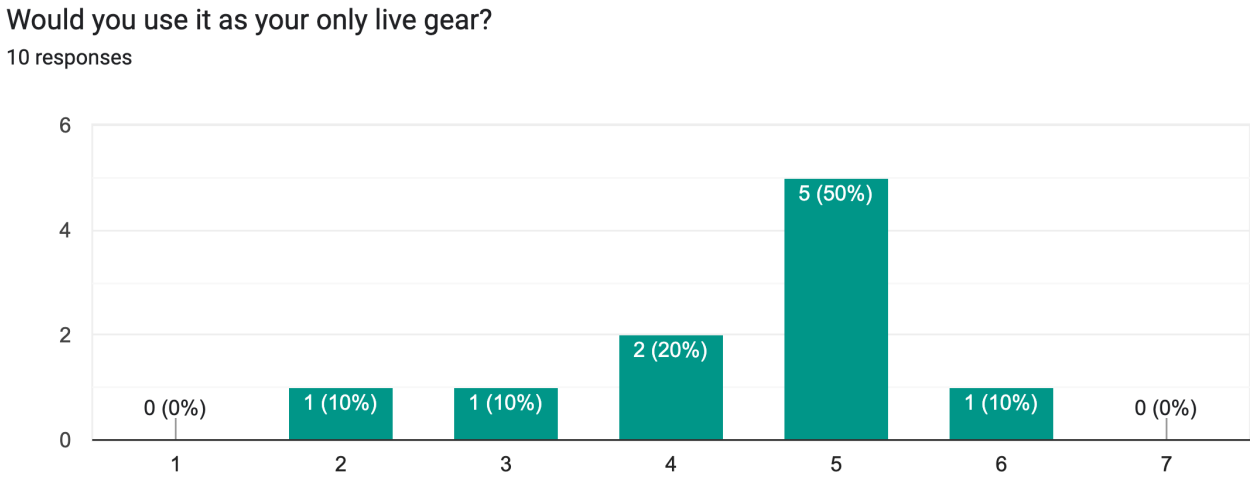

Figure 108: Evaluation Question 9

Figure109 shows that 50% of the participants would indeed integrate it in their live gear. 30% of them gave 6, 10% 5 and 10% 1.

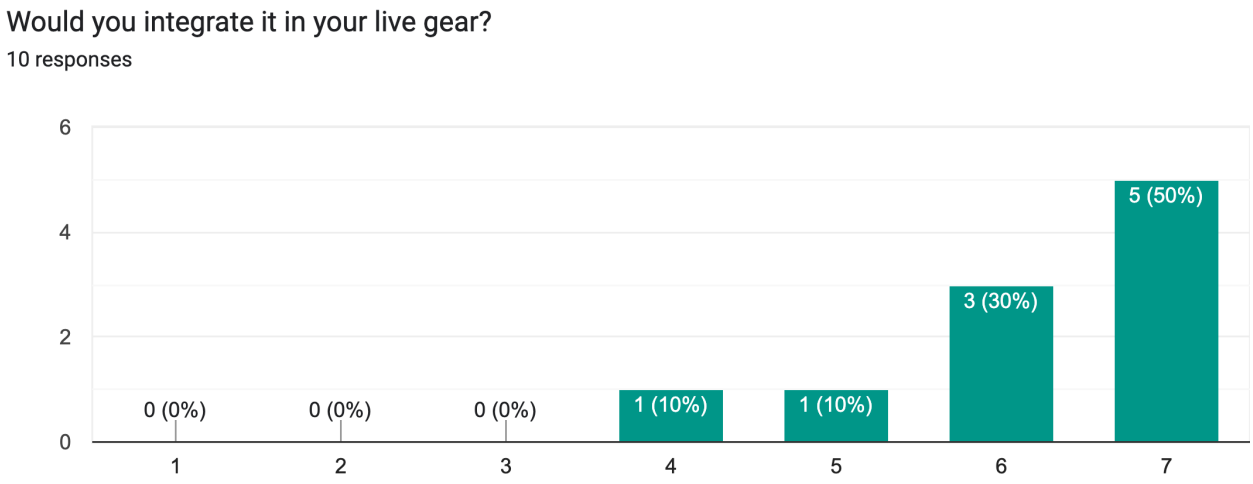

Figure 109: Evaluation Question 10

They were also asked if they would use it to program their own effects and if this device would make them want to learn how to code them. 50% gave 7 of 7 possible points, 40% 6 and 10% 5 (see Fig. 110).

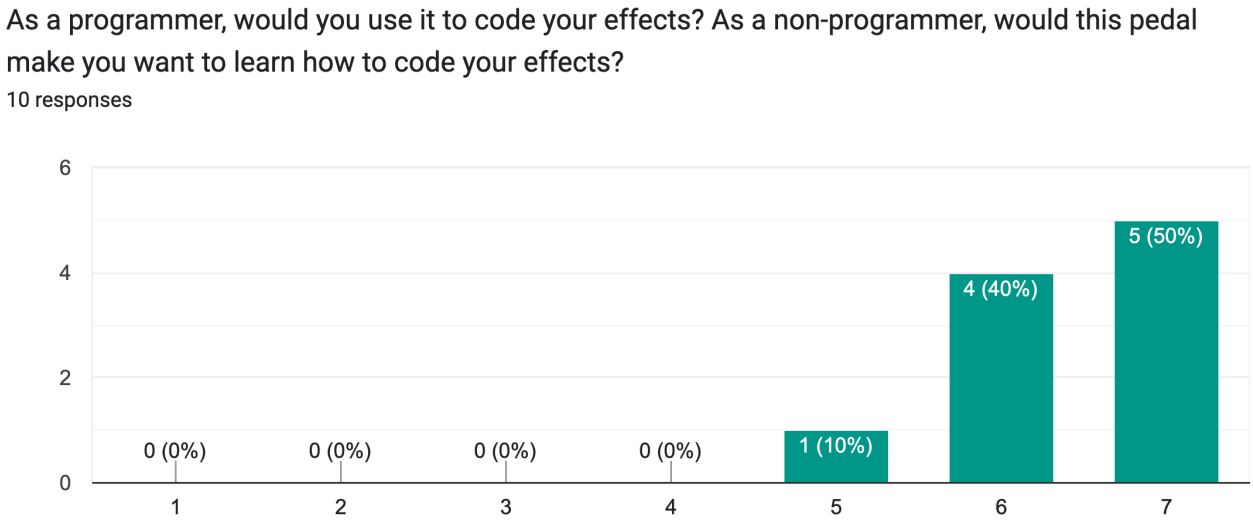

Figure 110: Evaluation Question 11

Figure 111 shows that 40% of the participants find it a good solution to avoid computers or other devices on stage giving 7 out of 7. 30% gave 6, 30% gave 3.

Figure 111 shows that 40% of the participants find it a good solution to avoid computers or other devices on stage, giving 7 out of 7. 30% gave 6, 30% gave 3.

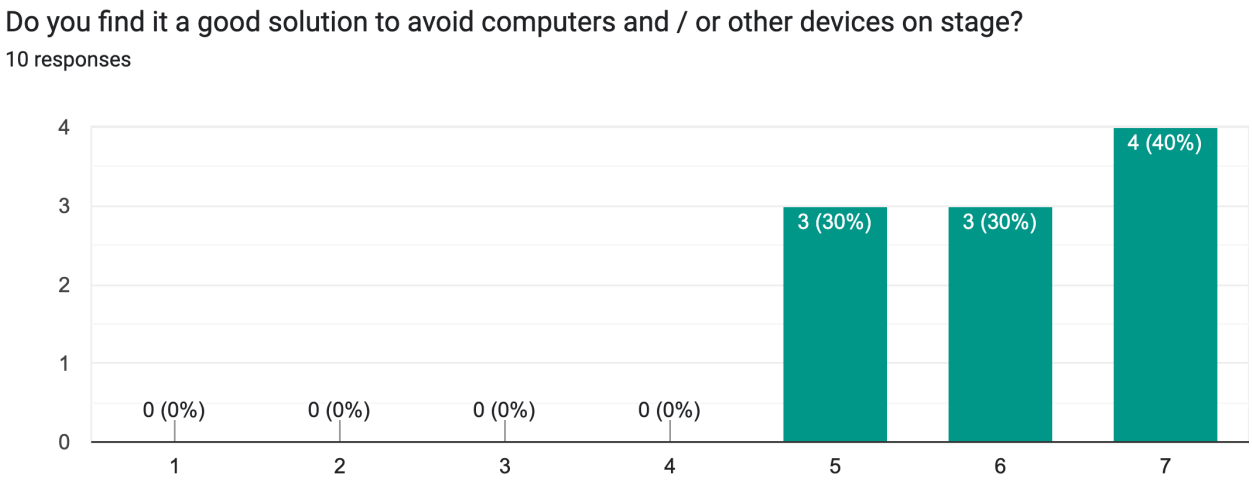

Figure 111: Evaluation Question 12

The arrangement of the potentiometers was rated 7 by 60% of the participants, 6 by 30%, 5 by 10% (see Fig.112).

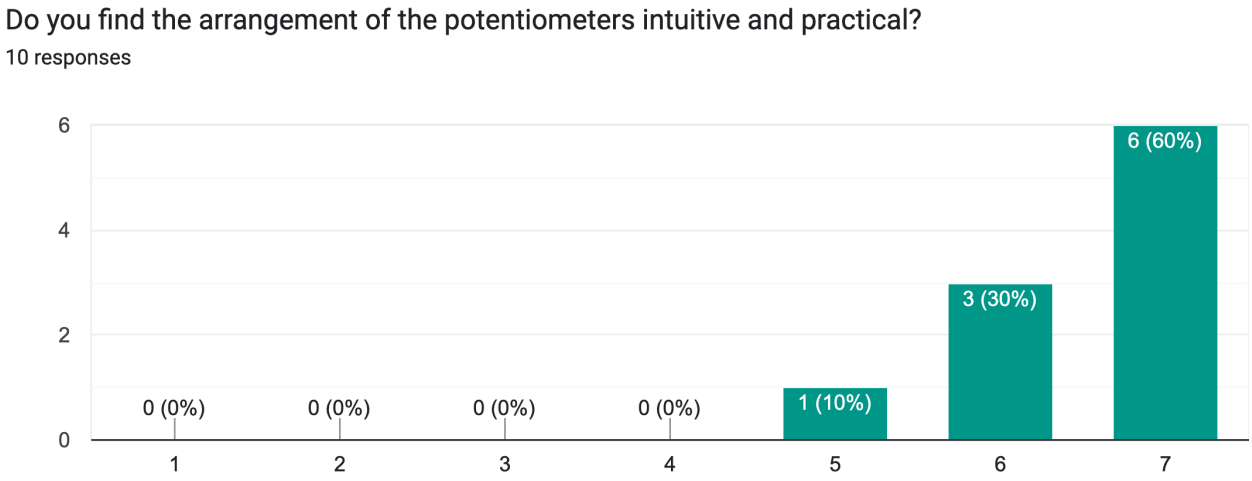

Figure 112: Evaluation Question 13

As shown in Fig. 113, inputs and outputs arrangement was rated 7 by 70% of the participants, 6 by 20%, 5 by 10%.

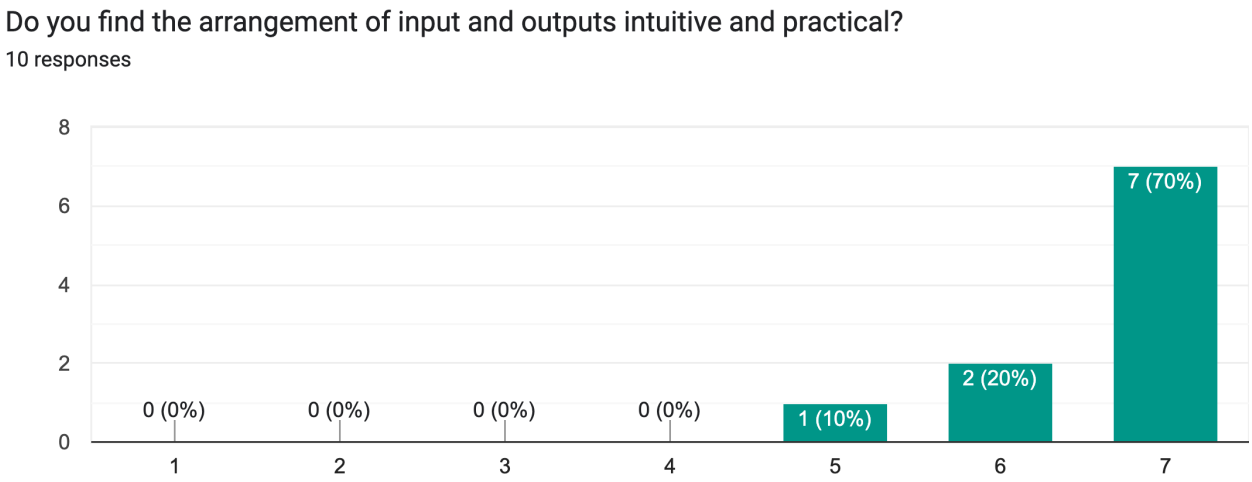

Figure 113: Evaluation Question 14

They were also asked if there are enough hardware controls. Figure 114 shows that 30% gave 7 out of 7, 40% 4, 20% 5 and 10% 1.

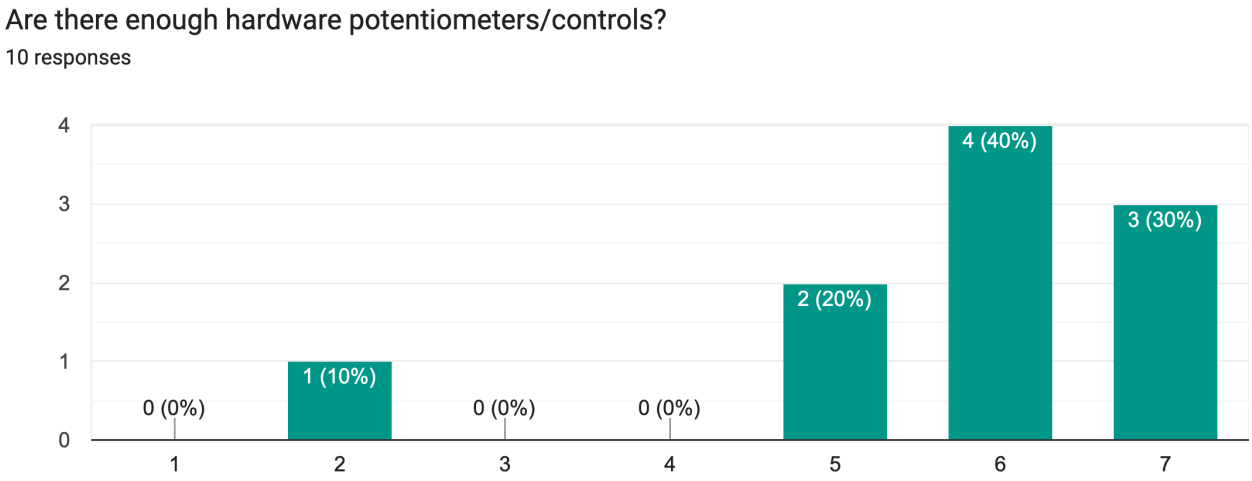

Figure 114: Evaluation Question 15

Figure 115 shows that 40% of the participants find the visuals to really make the pedal more interesting, giving 7 out of 7. 30% gave 5, 20% gave 6, 10% gave1.

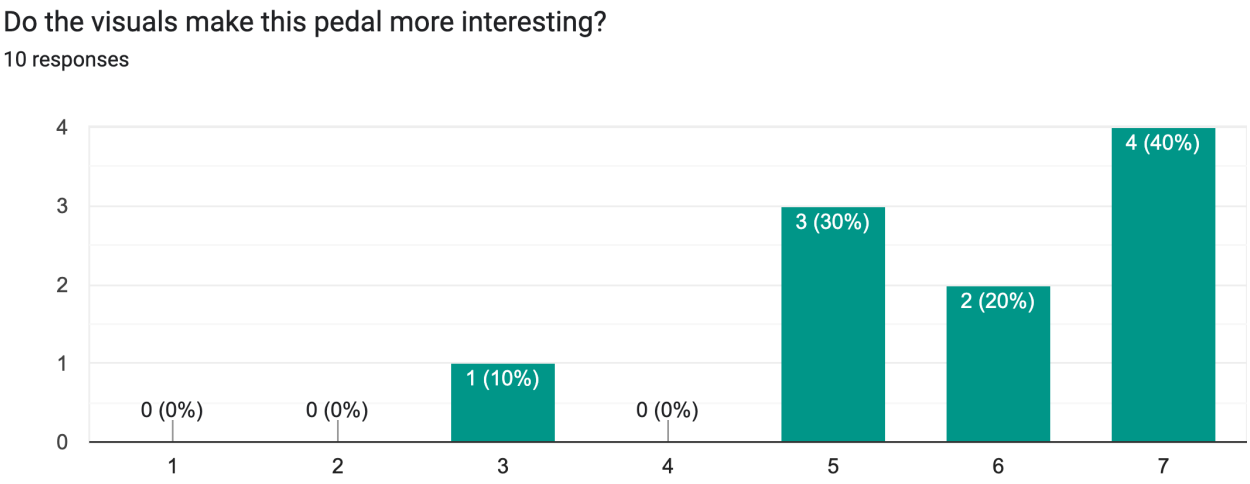

Figure 115: Evaluation Question 16

The screen size fits for 90% of the candidates, but for the 10% it should be larger (see Fig. 116 - 117).

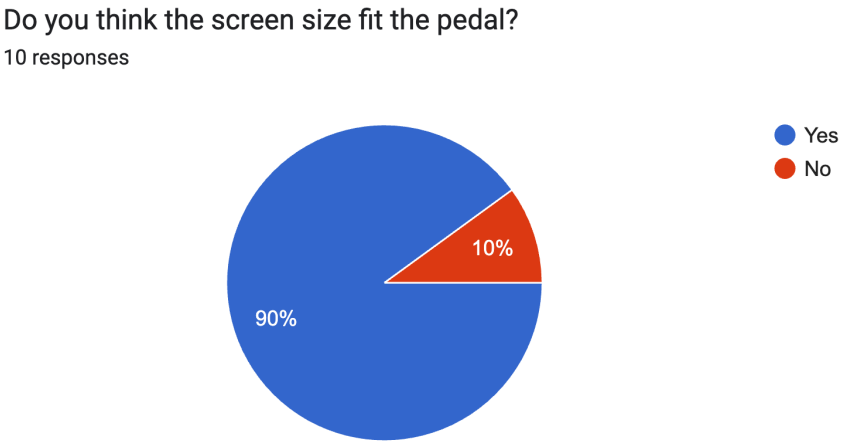

Figure 116: Evaluation Question 17

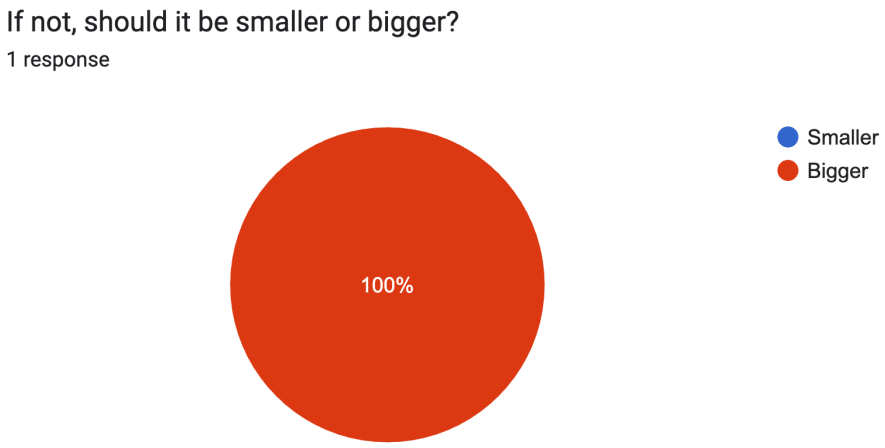

Figure 117: Evaluation Question 18

The participants were asked if they find it useful for phisical modeling. As shown in Fig. 118, 70 gave 6 out of 7 and 30% gave 7.

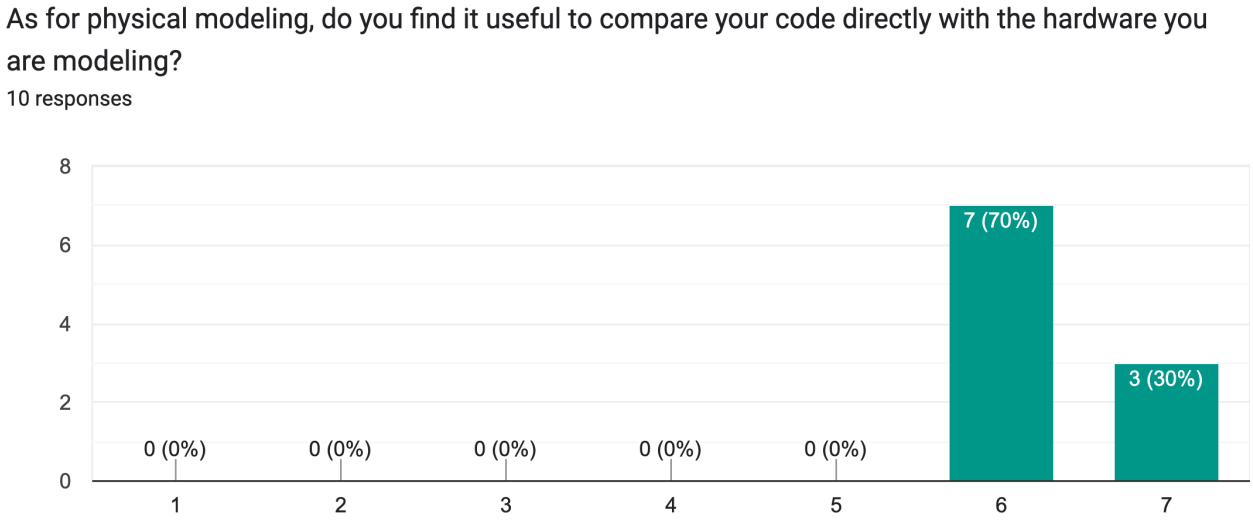

Figure 118: Evaluation Question 19

Figure 119 shows if the participants would consider making or buying this pedal. 30% gave 7 out of 7, 30% gave 6, 30% gave 5 and 10% gave 4.

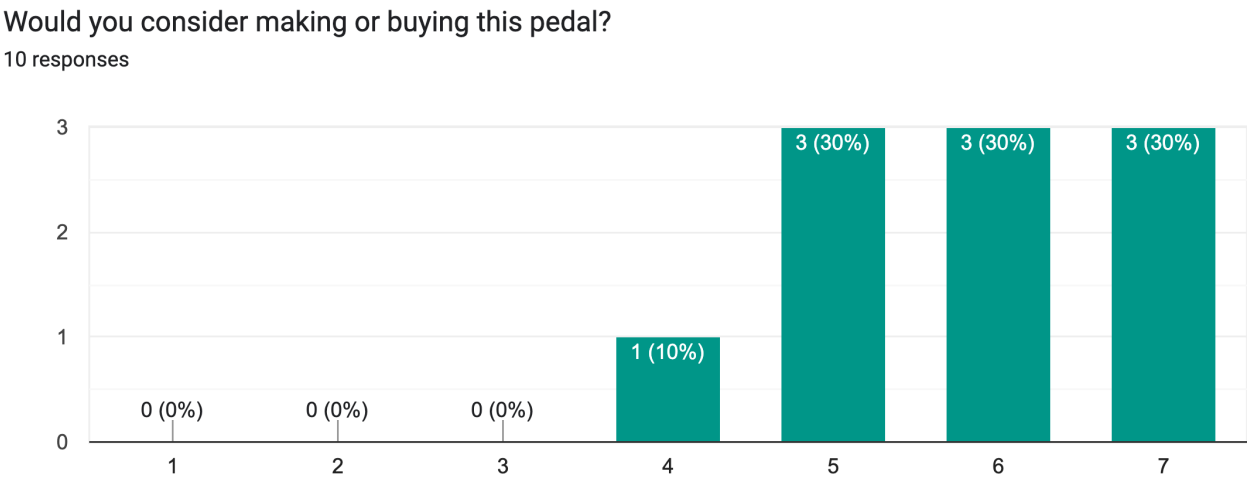

Figure 119: Evaluation Question 20

The participants were asked what price they considered reasonable for this device. 20% answered €300 and 30% could not judge it. The other answers are between €150 and €500. (see Fig. 120)

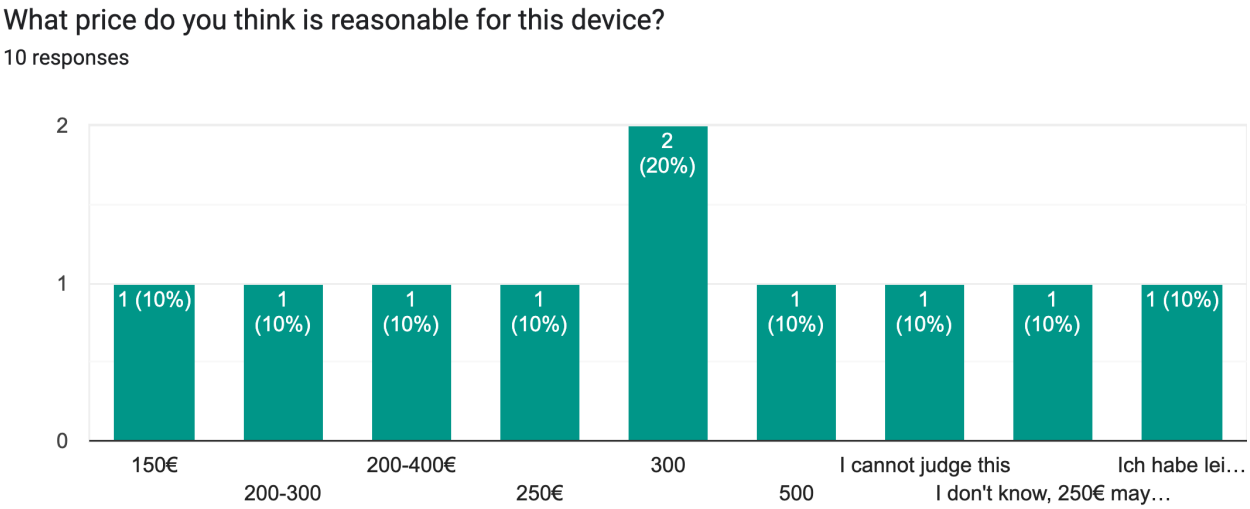

Figure 120: Evaluation Question 21

The last two questions asked are whether some essential features are missing (none are missing) or whether something could be improved. According to the results, no essential features are missing. Regarding the improvement, one participant answered that the user interface could be simplified and another participant that more presets or default effects could be added.

17.2 Discussion

The results indicate a positive response to the pedal and, the candidates having different guitar and programming skills or familiarity with similar devices, it can be concluded that it is suitable for various types of users.
The survey has asked how many pedals the participant owns to understand if the user is already comfortable using pedals. In case of not having much previous experience using pedals, it was interesting to observe how difficult it was using the device and what the general feeling of those participants was like. The result shows that the first approach does not seem so hostile at all.
Most users would integrate it in their live gear. Yet, for the users that might be playing in a more traditional way, it would be hard to just use this one device on stage. Surprisingly however, greater part of the participants have seen the possibilities and could use it live as the only device.
Some variety was observed in the haptic experience of users. This result could be taken into consideration to improve this aspect, by changing the enclosure material for example, or proceeding with lacquering and adding covers for rotary encoders.
At first the visuals were considered as just an add-on, but they turned out to be interesting and distinctive for this device. This result could lead to a further improvement on this side, improving the code and perhaps finding new solutions for more interactive sketches.
Greater part of the users did not perceive (almost) any latency. Just one of them perceived it. This means it could be considered making changes to the Raspberry Pi's operating system or trying other audio interfaces to improve its performance.
It was interesting to see the price range of the device estimated by the participants, it mostly ranged between €200 and €300. Considering the invested financial sources (see Appendix: Bill), not including the time and energetic sources used, it may be concluded, that the device would not be really profitable. This could be improved further in the future - some costs could be reduced by arranging the soldered circuits more efficiently and with not including some components, the reduction in the size of the pedal could be considered as well. At this time, it is still a low-cost solution for what it offers and the goal of making a cheap device has been achieved.
Overall, the results show that the pedal suits users' needs, but nevertheless some minor improvements have been suggested that could make the pedal into a completely professional and solid product.

18. Conclusion

During the development of this work, all possible solutions were adopted to be able to reduce the components to a minimum, both for reasons of space and money. The pedal lends itself to all the features sought at the beginning of the project, and has already been tested in various contexts.
As for the size of the pedal, the size of the enclosure is necessary to contain all the circuits and cables. The size of the circuits, as well as the cables, could be significantly reduced. For example, it would be possible to make a single circuit, in which the entire signal processing takes place, but this involves much more familiarity with circuit design and soldering. It would also be possible to unsolder the USB cable of the sound card, using a shorter one or a flat USB, to take up even less space. But it is not necessary, as the total cost would increase. Being modular, it is open to the most varied modifications simply by connecting cables and adding new circuits/components. This makes it adaptable to future needs.

The final bill was made as one of the key points of this project is to obtain the result at the lowest possible cost. The price is quite accessible and, considering all its possibilities of use, it is not possible to find something on the market that comes in the least to such a low price. For those who already own a Raspberry Pi, the cost becomes equal to that of any normal pedal currently on the market. It is obviously possible to further reduce the cost by saving on the screen and on the jack sockets, but I thought it is important not to save on the quality of these components. This ensures a longer duration over time.
The test carried out has brought quite positive results, which pave the way for future improvements that can be made before the product can confidently be placed on the market. A short video demonstration of the usage of the pedal and its function is represented in the video linked in the resources. [68]

References

[1] Mixdown. From tremolo to wah-wah: A history of the guitar effect pedal. 2021.
https://mixdownmag.com.au/features/from-tremolo-to-wah-wah-a-history-of-the-guitar-effect-pedal/

[2] Met Museum. Electro Vibrola Spanish Electric Guitar.
https://www.metmuseum.org/art/collection/search/718415

[3] Keith Holland. “DeArmond Volume Pedal”. Accessed July 20, 2022.
http://www.keithhollandguitars.com/dearmond-volume-pedal.html

[4] Dregni, Michael. Article on Vintage Guitar Magazine. “DeArmond Tremolo Control”.
https://www.vintageguitar.com/18545/dearmond-tremolo-control/

[5] TR Crandal Guitars. 1950s DeArmond Tremolo Control.
https://trcrandall.com/products/1950s-dearmond-tremolo-control

[6] Krayer, Johannes. Aritcle on Amazonia.de. “ZEITMASCHINE: ROLAND GP-8, EFFEKTPROZESSOR”. May 30, 2017.
https://www.amazona.de/zeitmaschine-roland-gp-8/

[7] White, Paul. Article on Sound on Sound. “Line 6 POD”. February, 1999.
https://www.soundonsound.com/reviews/line-6-pod

[8] Line 6. “Helix”. Accessed July 20, 2022.
https://de.line6.com/helix/helix-floor-rack.html

[9] Dimitri. Article on Amazona.de. “NEWS: KEMPER PROFILER OS8.0”. October 29, 2020.
https://www.amazona.de/news-kemper-profiler-os-8-0/

[10] Gitarre&Bass. "Fractal Audio kündigt Axe-FX III an” January 29, 2018.
https://www.gitarrebass.de/equipment/fractal-audio-kuendigt-axe-fx-iii-an/

[11] Neural DSP. “Quad Cortex”. Accessed July 20, 2022.
https://neuraldsp.com/quad-cortex

[12] Cremaschi, Andrea and Giomi, Francesco. “Rumore bianco. Introduzione alla musica digitale”. 2008.

[13] Romero, Juan & Borgeat, Patrick. Performance at TEDx Talks. “Live-Coding - programming masterly music”. Published on February 13, 2018.
https://www.youtube.com/watch?v=Ix2b_qFYfAA&ab_channel=TEDxTalks

[14] Imagine Publish Ltd. Raspberry Pi for Beginners, 2nd Edition. 2014.

[15] Raspberry Pi. “Raspberry Pi 4”. Accessed July 20, 2022.
https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[16] Electrosmash. “Pedal PI - Raspberry Pi ZERO Guitar Pedal". Accessed July 20, 2022.
www.electrosmash.com/pedal-pi

[17] Bloemer, Keith. Article published on Towards Data Science. "Neural Networks for Real-Time Audio: Raspberry-Pi Guitar Pedal”. May 25, 2021.
https://towardsdatascience.com/neural-networks-for-real-time-audio-raspberry-pi-guitar-pedal-bded4b6b7f31

[18] Hyllis. Article published on Reddit. “My first DIY pedal: touchscreen & parallel signal chain”. September 27, 2020.
https://www.reddit.com/r/diypedals/comments/j0xqk2/
my_first_diy_pedal_touchscreen_parallel_signal/

[19] Hillys. “Multi-effects guitar pedal w/ touchscreen – Parallel signal chains, modular, sound design”. Published on September 27, 2020.
https://www.youtube.com/watch?v=JoJEhhwpi9Q&ab_channel=Hyllis

[20] https://www.instructables.com/member/ElectroSmash/

[21] GuitarML. “Neutal Pi”. Accessed July 20, 2022.
https://github.com/GuitarML/NeuralPi

[22] Juce. “Juce". Accessed July 20, 2022.
https://juce.com/

[23] Elk-audio. “elk-pi". Accessed July 20, 2022,
https://github.com/elk-audio/elk-pi/releases/tag/0.9.0

[24] Delamar. “True Bypass. Was ist das?”. Accessed July 20, 2022.
https://www.delamar.de/gitarre/gitarren-effektgeraete/true-bypass-46981/

[25] Otosection. "Diy Pedal Building How To Wire A 3pdt Effects Pedal Footswitch For True Bypass”. Accessed July 20, 2022.
https://www.otosection.com/diy-pedal-building-how-to-wire-a-3pdt-effects-pedal-footswitch-for-true-bypass/

[26] Prasad, Leela. Tutorial published on Electronics Hub. “What is Relay? How it Works? Types, Applications, Testing". February 28, 2022.
https://www.electronicshub.org/what-is-relay-and-how-it-works/

[27] Glolab Corporation. “Relays, The Electromechanical amplifier”. Accessed July 19, 2022.
http://www.glolab.com/relays/relays.html

[28] Neuhold-elektronic. “Printrelais TD-5”. Accessed July 19, 2022.
https://www.neuhold-elektronik.at/catshop/popup_image.php?pID=3343

[29] Intertec Components. “Signal Relays. TD Relay”.
https://www.neuhold-elektronik.at/datenblatt/N3840.pdf

[30] Lüde Elektronic. “Wipptaster Taster Schwarz MRS-111C3”. Accessed July 20, 2022.
https://www.luedeke-elektronic.de/Wipptaster-Taster-Schwarz-MRS-111C3.html

[31] DIY Fever. “DIY Layout Creator”. Accessed July 20, 2022.
http://diy-fever.com/software/diylc/

[32] Patton, Don. Article on azcentral. “What Is a Buffer in Electronics?”. Accessed July 20, 2022.
https://yourbusiness.azcentral.com/buffer-electronics-20738.html

[33] Sasmita. Article on Electronics Post. “Explain the construction and working of a JFET . What is the difference between a JFET and a BJT ?". January 9, 2020.
https://electronicspost.com/explain-the-construction-and-working-of-a-jfet-what-is-the-difference-between-a-jfet-and-a-bjt/

[34] amzfx. “AMZ Jfet Splitter”. Accessed July 20, 2022.
https://www.muzique.com/lab/splitter.htm

[35] Thomann. “Behringer U-Control UCA222”. Accessed May 19, 2022.
https://www.thomann.de/at/behringer_ucontrol_uca_222.htm

[36] General Guitar Gadgets. “Mini Mixer”. Accessed 2 Juni, 2022.
http://www.generalguitargadgets.com/effects-projects/boosters/mini-mixer/

[37] General Guitar Gadgets. Circuit “Mini Mixer”. Accessed 2 Juni, 2022.
http://www.generalguitargadgets.com/pdf/ggg_mixer_sc.pdf

[38] Texas Instruments. “Working with Boost Converters”. Application Report SNVA731. June 2015.
https://www.ti.com/lit/an/snva731/snva731.pdf?ts=1657383928088&ref_url=https%253A%252F%252Fwww.google.com%252F

[39] Amazon. “LAOMAO Step-up boost power converter XL6009 for Arduino Raspberry DIY projects crafting". Accessed 25 Juni 2022.

https://www.amazon.de/gp/product/B00HV59922/ref=ppx_yo_dt_b_asin_image_o01_s00?ie=UTF8&psc=1

[40] Futura - Sciences. “Physical model”. Accessed 20 July, 2022. http://www.futura-sciences.us/dico/d/physics-physical-model-50003678/

[41] Graham, Josh. Article on Pedalboard Planet. “Effects Loop”. February 12, 2018. https://pedalboardplanet.com/effects-loop/

[42] https://github.com/giuseppetiyr/raspberry-pi-based-fx-pedal-rotary-encoder-code

[43] https://github.com/giuseppetiyr/raspberry-pi-based-fx-pedal-visual-1

[44] Turchet, Luca & Fischione, Carlo. Elk Audio OS: An Open Source Operating System for the Internet of Musical Things. 2011. https://www.researchgate.net/publication/350444728_Elk_Audio_OS_An_Open_Source_Operating_System_for_the_Internet_of_Musical_Things

[45] Watkiss, Stewart. Learn Electronics with Raspberry Pi: Physical Computing with Circuits, Sensors, Outputs, and Projects. Apress. Second Edition. 2020.

[46] Shaha, Samart. Learning Raspberry Pi. Packt Publishing. 2015.

[47] Python.org. "What is Python? Executive Summary”. Accessed July 20, 2022. https://www.python.org/doc/essays/blurb/

[48] The PiHut. “How to use a Rotary Encoder with the Raspberry Pi”. October 15, 2014. https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-use-a-rotary-encoder-with-the-raspberry-pi

[49] Last Minute Engineers. “How Rotary Encoder Works and Interface It with Arduino”. Accessed July 20, 2022. https://lastminuteengineers.com/rotary-encoder-arduino-tutorial/

[50] Raspberry Pi Pinout. “Pinout!". Accessed July 20, 2022. https://pinout.xyz/pinout/pin2_5v_power#

[51] https://www.circuito.io/app?components=9443,200000

[52] https://www.circuito.io/app?components=9443,200000,217614

[53] Amazon. “4.3 Inch DSI LCD Touch Display for Raspberry Pi 4B/3B+/3A+/3B/2B/B+/A+ 800×480 IPS Screen Capacitive Touchscreen Monitor MIPI-DSI Interface”. Accessed July 20, 2022.

https://www.amazon.de/gp/product/B0972Z6VPN/ref=ppx_yo_dt_b_asin_title_o09_s00?ie=UTF8&psc=1

[54] Pure Data. “Pure Data”. Accessed May 10, 2022.
https://puredata.info/

[55] Farnell, Andy. Designing Sound. The MIT Press. 2010.

[56] Technick-Passage24.de. “Xinda Miniatur Lüfter 5 V/DC 25 x 25 x 10 mm”. Accessed July 20, 2022.
https://www.technik-passage24.de/xinda-luefter-25-x-25-x-10

[57] Bloemer, Keith. Article on Nerd For Tech. “Neural Networks for Real-Time Audio: Introduction". April 26, 2021.
https://medium.com/nerd-for-tech/neural-networks-for-real-time-audio-introduction-ed5d575dc341

[58] Bilbao, Stefan. Numerical Sound Synthesis. 2007.

[59] MacConnel, Duncan & Trail, Shawn & Tzanetakis, George & Diriessen Peter & Page, Wyatt. “Reconfigurable Autonomous Novel Guitar Effects (RANGE)”. Paper presented at the Sound and Music Computing Conference 2013, SMC 2013, Stockholm, Sweden. 2013

[60] Ragland, John. “Digital Simulation and Recreation of a Vacuum Tube Guitar Amp.” Master Thesis. Auburn University, 2020.

[61] Processing. “Processing”. Accessed July 25, 2022.
https://processing.org/

[62] https://github.com/supercollider/supercollider/blob/61ad680238c965d780def2346994f6377cac6204/SCClassLibrary/Common/Audio/Demand.sc

[63] Jack, Robert & Mehrabi, Adib & Stockman, Tony & McPherson, Andrew. “Action-sound Latency and the Perceived Quality of Digital Musical Instruments: Comparing Professional Percussionists and Amateur Musicians". Music Perception: An Interdisciplinary Journal. 36. 10.1525/MP.2018.36.1.109. 2018.

[64] Autodesk. “Fusion 360”. Accessed June 22, 2022.
https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription

[65] Massat, Pierre. Article on Guitar Extended. “Step-vibrato effect for guitar with Pure Data”. January 18, 2012.
https://guitarextended.wordpress.com/2012/01/18/step-vibrato-effect-for-guitar-with-pure-data/

[66] Fromaget, Patrick. “How to Auto Start a Program on Raspberry Pi? (4 ways)”. Visited on August 02, 2022.
https://raspberrytips.com/autostart-a-program-on-boot/

[67] Reddit. “How to autostart a PD Patch on raspberry pi?”. Visited on August 02, 2022.
https://www.reddit.com/r/puredata/comments/kprgsn/
how_to_autostart_a_pd_patch_on_raspberry_pi/

[68] https://www.youtube.com/watch?v=3QIhOViTr3w&ab_channel=GiuseppeTaormina

Appendix: Bill

1x Raspberry Pi 4GB	€75,00
1x Pi Power Supply	€7,00
1x SD Card 4GB	€2,90
1x Screen	€43,90
1x Audio Interface	€21,00
4x Mono Sockets	€4,40
2x Stereo Sockets	€2,10
3x Rotary Encoder	€6,00
2x Stereo Potentiometers	€5,00
1x Switch per Bypass	€0.20
1x Fan	€2,60
2x Relays	€1,80
1x StepUp Booster Converter	€4,00
1x PerfBoard	€0,40
1x StripBoard	€3,00
4x Transistor	€1,60
2x OpAmp	€0,70
Resistors	about €2,50
Capacitors	about €2,50
Jumper Cables	about €2,00
Tot	€188,60

I want to consider the cost of the 3D model as an extra, as it is not essential for the correct functioning of the pedal. Its price can vary from €50 to €80.