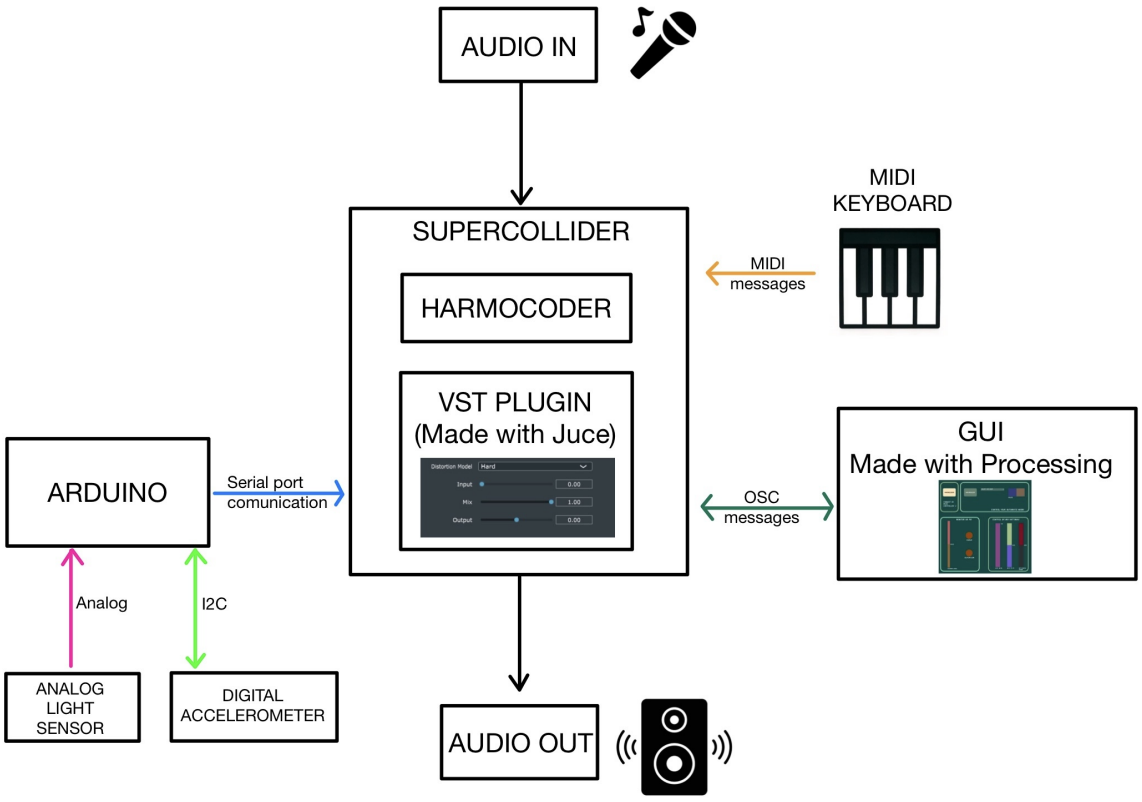
Harmooder
CMLS Project Report

Mattia Montanari id: 11007610
Flavio Ingenito id: 10865407
Giorgio Magalini id: 10990259
mattia.montanari@mail.polimi.it
flavio.ingenito@mail.polimi.it
giorgio.magalini@mail.polimi.it
Politecnico di Milano

May 27, 2024

Project Overview



Introduction

This project combines Supercollider, JUCE, Arduino, and Processing to create a musical system where the singer can expand their vocal performance using a harmonizer in a new creative and intelligent way.

A harmonizer is an audio device that adds harmonic notes to an input audio signal, producing a rich and complex musical effect. In this project, we implemented not only a classic automatic harmonizer but also a fusion between a harmonizer and a vocoder, which we called Harmocoder.

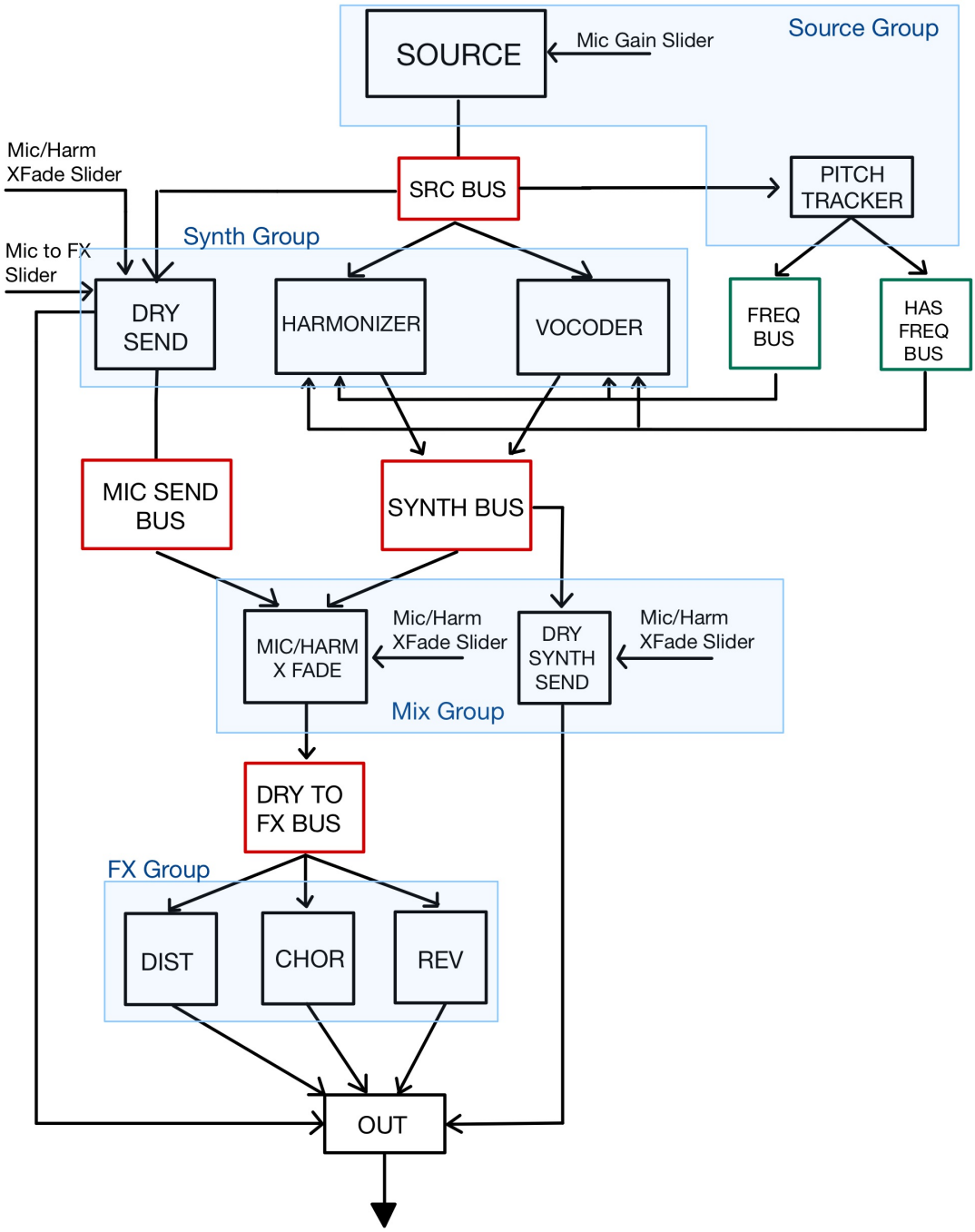
In the automatic harmonizer, the harmonization is fixed: given the key of the scale, each input note of the singer is dynamically used as a root note to build the chord. In contrast, the Harmocoder allows the singer to choose how to harmonize their voice by simultaneously playing a MIDI keyboard. Each note sung by the artist is pitched according to the keys pressed at that specific instant. While this working principle is similar to a vocoder, the main difference is that a vocoder uses the audio input to shape the formant of what is played with the MIDI keyboard, whereas our Harmocoder keeps the same harmonic content of the input voice but pitches it up or down.

To provide even more flexibility to the artist, we added the possibility to process the output sound of the harmonizer with effects applied and removed based on the singer's gestures and the changing light in the performance environment. We achieved this using two Arduino sensors: a digital 3D accelerometer and an analog gray-scale light intensity sensor. The accelerometer, attached to a glove worn by the singer, maps horizontal and vertical movements to turn on or off a distortion plugin (implemented with Juce) and a chorus effect (realized in Supercollider), respectively. The light sensor dynamically changes the dry/wet parameter based on the light intensity on stage.

The main idea of our project is to allow the artist to significantly extend the horizon of their performance with minimal effort. Therefore, we adopted a simple GUI, avoiding the need for extensive parameter adjustments and allowing the artist to focus on singing and harmonizing with the keyboard.

Supercollider

Graph of the Server Architecture



Pitch Shifter

Pitch shifting is typically done by decreasing or increasing the read speed of multiple delayed reading heads, which signals can be overlapped by tapering and mixing them together to avoid clicking noises due to the crossing between the reading and the writing pointers. This rudimentary method can cause some comb filtering; if the delay is too large, it becomes perceivable and unpleasant. To avoid artifacts, a granular approach is preferred. By randomly varying the length of the windows and the overlapping between the delayed windows by small amounts, artifacts can be avoided. This is the approach of the Supercollider UGen called *PitchShifter*.

The *PitchShift* UGen is a time-domain granular pitch shifter. Grains have a triangular amplitude envelope and an overlap of 4:1, using linear interpolation of the buffer.

Pitch Tracker

Fast and responsive pitch tracking in Supercollider is hard to obtain. We found that using the *Tartini* UGen gives the best possible results. Tartini UGen is an alternative pitch follower that uses autocorrelation like Pitch UGen, but with an adapted method, calculated via FFT. It calculates a modified autocorrelation function following the paper: Philip McLeod and Geoff Wyvill (2005) "A Smarter Way to Find Pitch", ICMC Proceedings; 138-141.

Further testing revealed that the control-rate signal containing the pitch information is slightly delayed from the actual pitch of the input signal. So, we delayed the incoming signal by a small amount before the pitch shifting.

VST Plugin in Supercollider

To apply our distortion plugin, we used the *VSTPlugin* Supercollider extension. This extension allows us to use our plugin as an insert to modify the sound produced in Supercollider while remaining within the Supercollider environment.

Chorus

We implemented chorus in Supercollider by creating a set of delays with slightly modulated delay times for the left and right channels to widen the stereo image.

Reverb

We used Supercollider's *GVerb* UGen, a two-channel reverb UGen based on the "GVerb" LADSPA effect by Juhana Sadeharju. We adjusted the settings to our taste and applied filtering to remove some high and low frequencies, making the resonance more pleasant.

Communication

Supercollider receives serial data from Arduino and OSC messages from Processing, sending feedback to Processing via OSC for visual monitoring of the effect parameters.

Distortion Effect

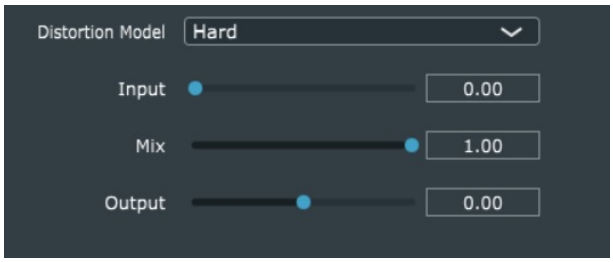


Figure 1: Distortion VST GUI

We implemented three types of distortion:

1. Hard
2. Soft
3. Saturation

The *Hard* distortion clips the audio at a certain threshold.

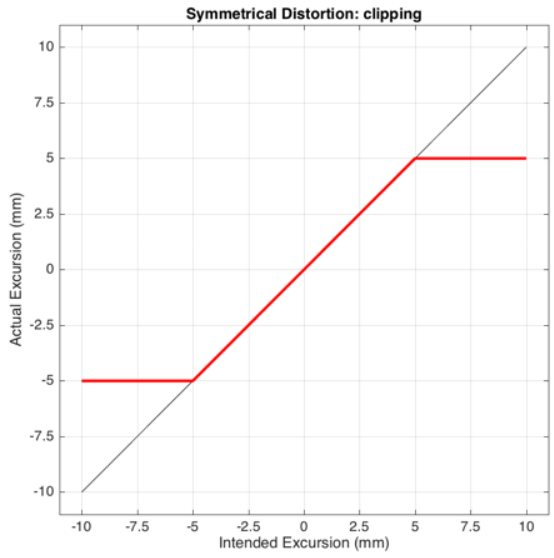


Figure 2: Hard Clipping Distortion Transfer Function

The *Soft* distortion is implemented by applying a hyperbolic tangent function. This function is inherently non-linear, resulting in a distorted output. However, compared to hard distortion, it produces a smoother and more gradual alteration of the signal.

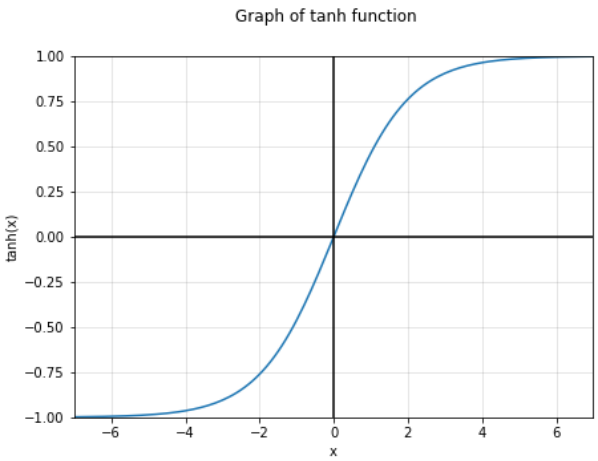


Figure 3: Soft Clipping Distortion Transfer Function (Hyperbolic Tangent Function)

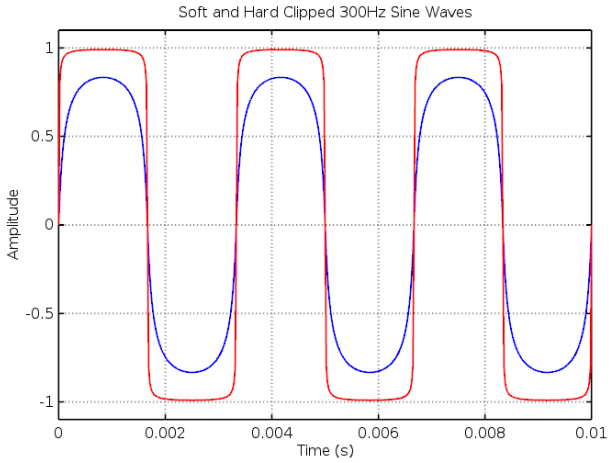


Figure 4: Hard Clip Distortion vs Soft Clip Distortion

In order to achieve smoother distortion with odd harmonics, we implemented *saturation*. Since saturation involves an asymmetric transfer function, the distorted signal emphasizes not only odd but also even harmonics. This results in a sound where the fundamental frequency and its even multiples are more pronounced, adding richness and depth to the audio. By focusing on even harmonics, the output retains a musical quality while avoiding the harsher effects of even harmonics, creating a more pleasing and characterful distortion.

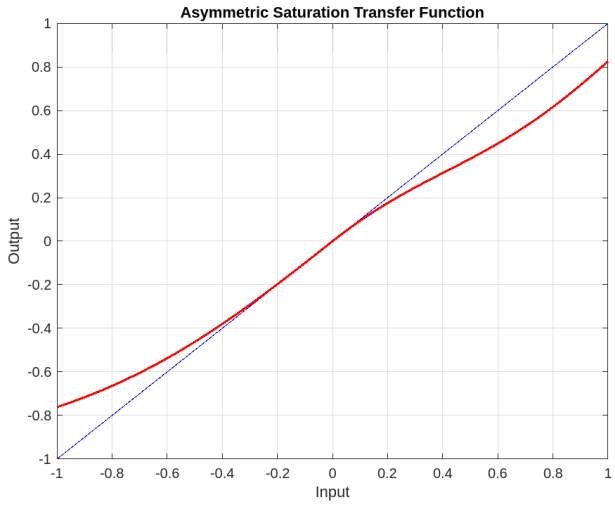


Figure 5: Asymmetric Saturation Transfer Function

Sensors

Arduino Wiring and Communication

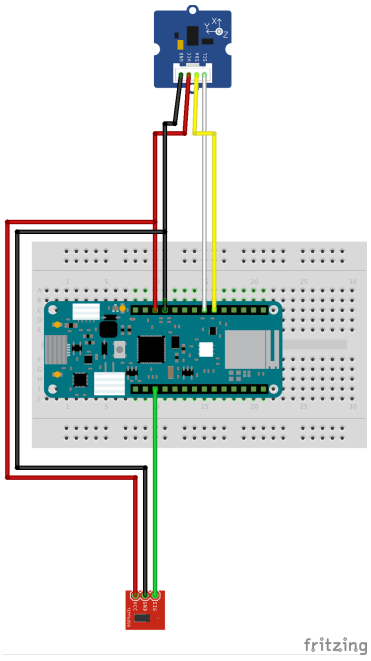


Figure 6: Arduino and Sensors Wiring

The ambient light sensor is an analog sensor, so communication with Arduino is done by simply connecting the sensor signal out to an analog input on the Arduino. The digital sensor communicates with Arduino using the IIC (or I2C) protocol, a *two-wire* protocol characterized by two signals:

1. SCL: synchronous clock
2. SDA: synchronous data

Grey Scale Ambient Light Sensor

We used an analog ambient light sensor to let the ambient light control the amount of reverberation effect. The sensor is mounted on a breakout board and connected to Ground, Vcc, and the signal output to analog input 1 of the Arduino.

Digital Accelerometer Sensor

When the device accelerates in a direction, the output signal for that specific motion is a wave with two peaks: one positive and one negative (or vice versa). We implemented a code in Arduino to recognize four kinds of hand movements based on the combinations of positive and negative directions of the y and z axes. The algorithm is a temporized double threshold trigger. If the signal crosses a positive threshold and subsequently a negative one within a specified time interval, an output trigger signal is generated. The output of this algorithm encodes the hand movement and sends it to a serial port. The sensor must be mounted on a glove with the palm facing downwards.

GUI (made in processing)

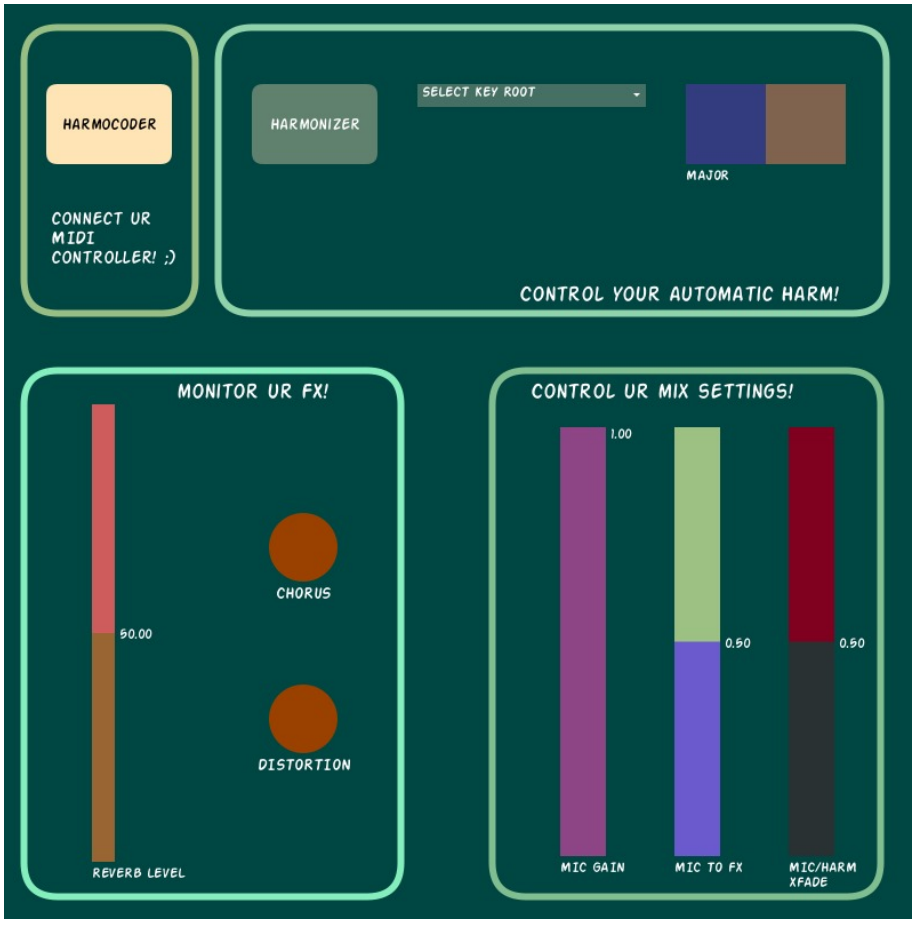


Figure 7: Multi Control Processing GUI