Gyrosynth

Sound Synthesis
Henrik von Coler, SS 2019

Stoltenberg, Jan 401445
Waldeck, Udo 404708

10. August 2019

1 Introduction

During Technical University of Berlin's seminar *Sound Synthesis*, led by Henrik von Coler, each group was to develop a synthesizer or digital effect using a Raspberry Pi 3.
A template and serveral examples have been provided to the students while a concrete project idea was chosen by each group independently.
This document provides information about the idea, work documentation and implementation about the project *Gyrosynth*.

Section 2 introduces the fundamental thoughts and ideas for this work, explaining why the Gyrosynth has potential to have an added value when making music with multiple persons. Section 3 explains the model in technical detail, giving an overview of the Gyrosynth's signal flow, handling and the different technologies involved. Finally, section 4 summarizes the projects, its lessons learned and foresight.

2 The idea

While making music together is a great way to spend an evening interacting with each other, there are certain limitations that prevent freedom and creativity. Especially if one doesn't explicitly meet somewhere for a music jam, odds are not very high that everyone has an instrument or knows how to get involved. Thus, often only a single musician or part of the group tends to be active.
A musical instrument that could be 'played' by multiple users at the same time therefore seems desirable in such a situation. At the same time, musical amateurs or intimidated group members, who might be uncomfortable with playing an instrument or engage musically by oneself, should be strongly encouraged to be part of this musical act.

The Gyrosynth was built pursuing these exact approaches and stands for an innovative, easy and fun way to foster active musical engagement in small groups.
As depicted in more detail in section 3, the Gyrosynth is controlled by moving and rotating a smartphone. Since the majority of people today use smartphones with built-in gyroscopes, the Gyrosynth uses this available technology to allow for an innovative approach to designing a sound by motion. Here, up to three players can choose how they would like to shape the final sound that is being emitted from the synthesizer as seen in figure 1.

There are three roles to be chosen from:

1. *Controlling the pitch of the fundamental frequency*
2. *Applying an amplitude modulation effect (AM) and changing its dry/wet ratio*
3. *Adjusting the overall volume of the sound*

Once each role has been selected, each player can connect their mobile phone and start manipulating the sound. Therefore, everyone is actively participating in the action of making music. Also, the inhibition threshold is lowered, as no prior musical knowledge is necessary and the outcome of the sound is a product of the actions of all players.

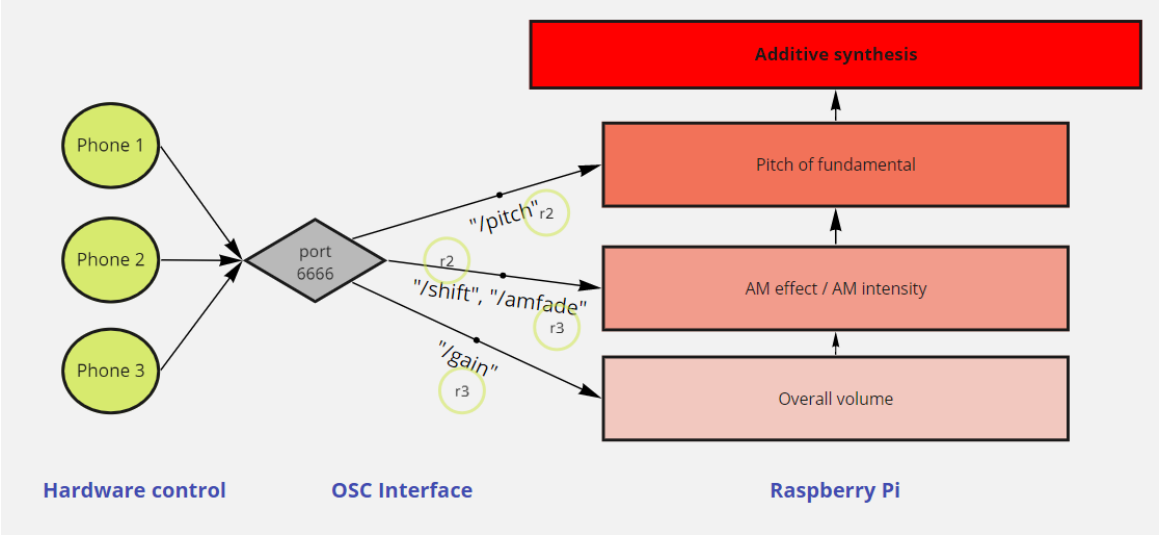


Figure 1: System overview

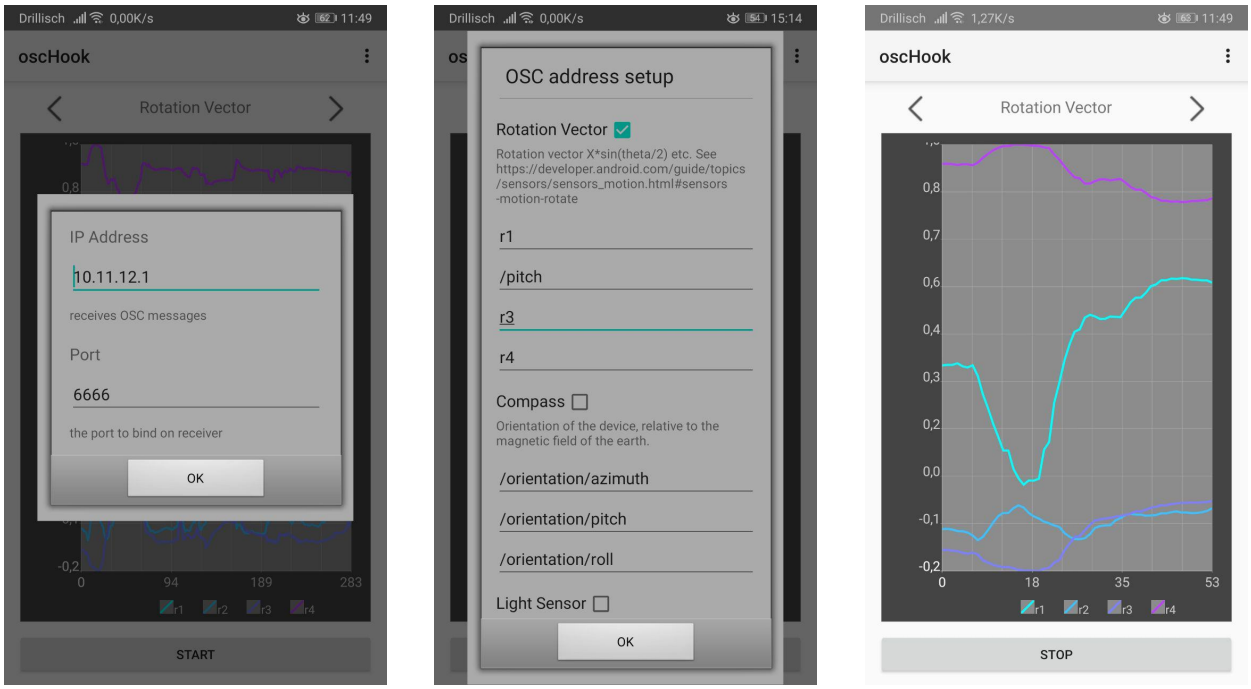
The creators hope that the Gyrosynth will be tried out in areas like public parks, at home and other get-togethers, inspiring also non-musicians and groups to allow for a fun musical experience and thus connect socially.

3 Work documentation

The basic hardware and software used in this project are a Raspberry Pi 3 with Raspbian OS (based on Debian Linux distribution), a Behringer UCA222 stereo USB-Audiointerface and a smartphone with an application of choice that can send OSC messages. These are used to change parameters of the synthesizer in real time.
The Android app tested is "oscHook" (https://play.google.com/store/apps/details?id=com.hollyhook.oscHookhl=en). OSC apps send OSC messages over a network connection to the Raspberry. While oscHook reads out smartphone sensor data to measure for example phone orientation in 3D space, with other apps the user can set up buttons and sliders to change control values. The respective app needs to be set up with a port and the correct OSC strings or addresses (e.g., /pitch) as seen in figure 2b. The required IP address is that of the Raspberry and the OSC port and addresses are configured in the "config.yml" file. After turning on the Raspberry PI, while the audio interface is connected to it, and connecting the smartphone to it's wireless network, the Gyrosynth should start and be controllable.

This project is programmed in C++ and consists of six subscripts plus header files. "main" is as usual the the entry point to the program and needs the already mentioned "config.yml" file to initialize the "YamlMan" object. YamlMan has the purpose of configuration by assigning the chosen strings of the OSC addresses to their respective paths in the program, so that the values of the OSC messages are assigned to the correct variables. The "Gyrosynth" class, of which an instance is initialized in "main", represents the runtime loop of the synthesizer. In its process function the generated samples are written into the output buffer, so that the here used audio engine "JACK" (http://jackaudio.org/files/docs/html/index.html) can pass them on to the audio output. For this to work an instance of the class "oscman" is required that handles the OSC communication between phone and synthesizer and the adjustment of the output values of the used android app. For instance, when using the "Rotation Vector" menu of oscHook as seen in figure 2b, the sent floating point values range from -1 to 1. Because negative values make no sense regarding frequencies, the absolute value is calculated and the value range is adjusted to fit every use case. This results in the fact, that after 180°phone rotation in each of the three room dimensions the value output is mirrored.

In this project, four variables are controlled by OSC as seen in figure 1. "/pitch" changes the fundamental frequency of the basic sound, that is created by an instance of the class "sinusoid". Regarding the amplitude modultaion effect (AM), "/shift" changes the frequency of the AM of the base sound in a way so that this effect ranges from a tremolo at low frequencies to an aliasing effect at frequencies, that are perceived as tones (above 16 Hz). The waveform of the AM is created by an instance of the class "SquareWave", which corresponds to the name of the class, so that base sound and AM effect at higher frequencies are easier to distinguish from another. To control the percentage of the AM effect that is mixed with the base sound, "/amfade" is used. Finally "/gain" functions as an overall volume control. For a better understanding, please have a look at the code in the git repository (https://gitlab.tubit.tu-berlin.de/u.waldeck/Gyrosynth.git).



(a) setup of IP address and OSC port (b) setup of OSC addresses (c) GUI of oscHook

Figure 2: setup of Android app oscHook

4 Conclusion

Because of the the spatial interaction with the smartphone, sound creation with Gyrosynth is an intuitive and expressive act. It is an inspiring tool to create unusual and mostly monophonic sounds alone or in a group. Since the creation of new and unknown sounds was one of the goals of the project, we didn't focus on musicality, but on creativity. Because this project is open source as requested by the seminar and no expensive hardware is necessary, Gyrosynth is easily available to interested people, who want to try something new. Challenges for our first audio programming project have been all over the programming infrastructure of course starting with C++ syntax. But we never the less created an interesting prototype and have learned a lot. Matters that still need to be fixed are the slow refreshrate of oscHook of 100ms. First of all this hurts the directness of the interaction with the sound, secondly it also generates sound clicks by value jumps, when users move their phone to fast. It has not been tested how many OSC messages can be sent simultaneously with one running "oscman" instance. To extend the insturment more effects can be added in the future like lowpass filters, distortion or noise generators so that more users can interact with the same device.