HUMAN COMPUTER INTERACTION

FINAL PROJECT REPORT

Giovannni Bindi
DINFO
Università degli Studi di Firenze
`giovanni.bindi@stud.unifi.it`

February 21, 2020

ABSTRACT

Human Computer Interaction (HCI) is a highly multidisciplinary field that focuses on the study and the design of interactions and interventions involving people and technology. In this project a link between music production and HCI has been studied: since humans have worked over millennia to develop and refine interactive musical technologies, ranging from bone flutes to synthesizers, the study of music generation in the HCI setting is therefore well suited, as evidenced by scientific publications [1] and peer-reviewed conferences in the field [2].

1 Introduction

This project's focus was to develop a simple hardware/software interface aimed at producing generative music. Generative music is a term popularized by Brian Eno [3] to describe music that is *ever-different and changing*, and that is created by a *system*. The system, in this case, is a combination of hardware and software components that interact together, with the user and with the surrounding environment in order to generate music. The music generation happens on the software side and the generation algorithm has been built using the Max/MSP [4] programming language.

2 Hardware and Software

Hardware The physical device has been realized with an Arduino Uno board (Figure 1), coupled with a small number of non expensive sensors. These sensors are responsible for capturing the "state" of the environment and for communicating their measurements to the Arduino board.

Figure 1: Arduino Uno

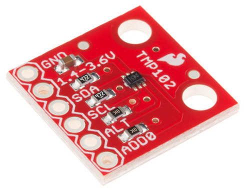


Figure 2: SparkFun TMP102

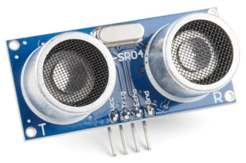


Figure 3: Ultrasonic Sensor HC-SR04

Figure 4: SparkFun Electret Microphone

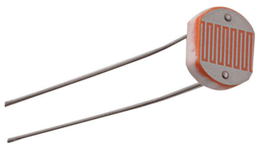


Figure 5: Photoresistor

The sensors used are:

- A SparkFun TMP102 temperature sensor (Figure 2).
- An Elecrow HC-SR04 Ultrasonic sensor (Figure 3).
- A SparkFun Electret Microphone (Figure 4).
- An Adafruit Light Dependant Resistor (Figure 5).

Their role is, respectively, to measure the **temperature** in the room, the **distance** to the closest object[1], to capture any **sound pressure variation** in the nearby and to measure the **intensity of the light** inciding on the apparatus. A soldering procedure has been operated on the SparkFun sensors in order to be able to mount them on a breadboard and connect them to the Arduino board. The connections (Figure 6) have then been made up with simple male jump wires.

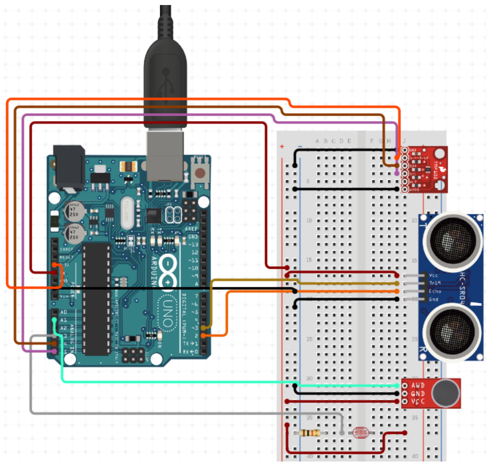


Figure 6: The connections circuit

The temperature is measured in Celsius (C) and the distance in cm. The microphone is not so sensitive and here is used only to capture an intense short variation in the sound pressure (e.g. an hand clap or a finger snap). The LDR sensor is even more basic and the light intensity is sensed through the resistance variations (i.e. in Ω).
Every piece of hardware used is certified Open Source Hardware and both the Arduino Software and the SparkFun firmware are released under Creative Commons licenses.

Software Three software components have been developed: an Arduino sketch that receives the values measured by the sensors and sends them to the computer; A Max/MSP patch[2], responsible for generating the music; A simple Max/MSP-compatible GUI (Figure 8), that is responsible for cutting off a large portion of the lower-lever patch complexity, and lets a less-experienced user play with the system.

[1]With respect to the direction pointed by the ultrasonic sensor.

[2]A *patch* (Figure 7) is the name of a Max/MSP program, made up by arranging and connecting building-blocks of objects within a visual canvas.

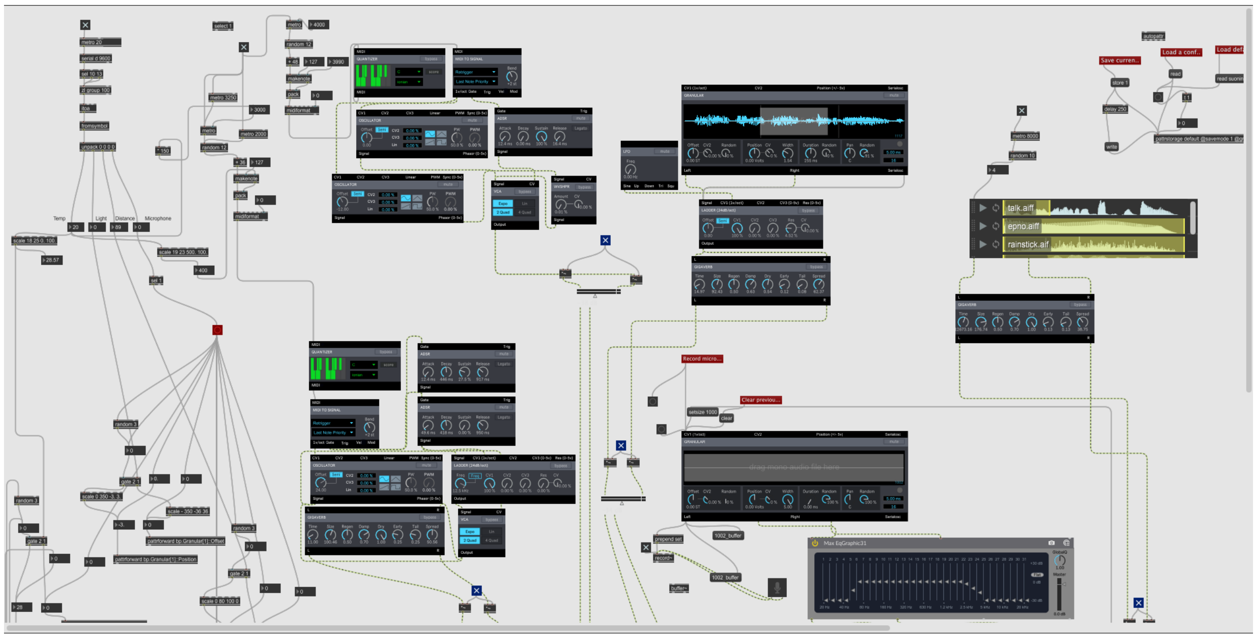
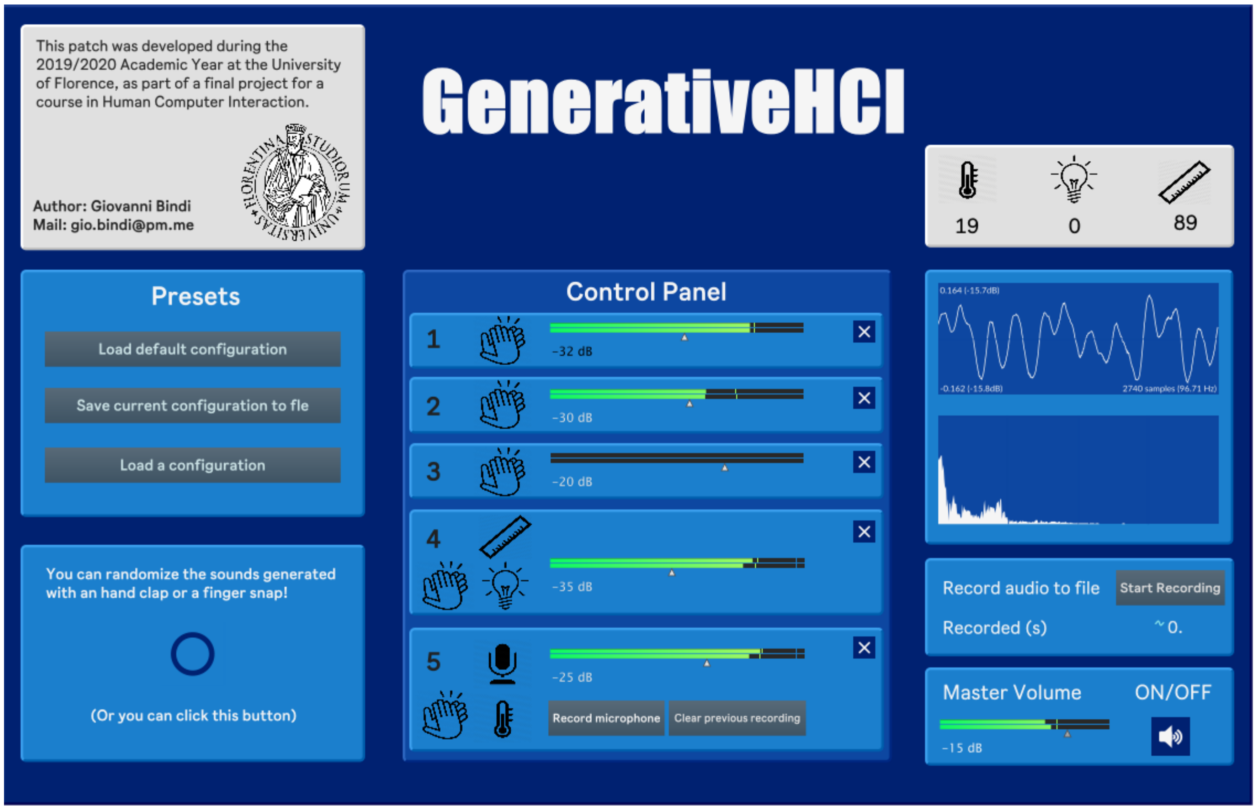


Figure 7: Part of the music-generation patch



This patch was developed during the 2019/2020 Academic Year at the University of Florence, as part of a final project for a course in Human Computer Interaction.

Author: Giovanni Bindi
Mail: gio.bindi@pm.me

You can randomize the sounds generated with an hand clap or a finger snap!

(Or you can click this button)

Figure 8: UI generated within Max/MSP

3 Needfinding, Personas and Scenarios

Needfinding The process of needfinding has been carried out by interviewing people: in the beginning the only people interviewed were *mid-twenties males* enrolled in an *electronic music* conservatory program. These interviewees manifested interest in a **small** device (i.e. that could be carried in a backpack or in a bag) and that could be **easily** interfaced with a computer. These people usually travel a lot with their equipment and do not have strong informatics' skills.
Later on the interviewees pool had been extended to almost anyone: males and females ranging from 20 to 60 years with very different backgrounds. Two types of individuals manifested interest in this idea: people involved in various kinds of art movements (i.e. actors, dancers) and science/engineering students that were music enthusiasts as well. The first ones have little to no expertise neither in music production nor in computing and expressed the necessity of an **interface** that could be **easily comprehended** by a non-technologically-skilled user.

Personas From the previous analysis two main personas emerged:

Daniele, 26 years old, male. He is a music professional who is daily involved in electronic music production. He performs several shows during the year and frequently travels abroad. He has light electrotechnical skills and basic computing knowledge.

Giulia, 25 years old, female. She is a semi-professional contemporary dancer that performs a variable number of dance shows. These dance acts can be performed by her alone or by a whole ballet company. She has no computing knowledge and no electrotechnical skills.

Scenario 1 Daniele is planning his next electroacoustic exhibition: an audio-visual performance in a museum. He decides to embed a dynamical component into his set by exploiting the sensing capabilities of the device. He is at home modifying the core Max/MSP patch in order to suit at best his needs.

Scenario 2 Giulia is with her dance company rehearsing for the next show. They are at the theater, along with the lighting technician and the sound engineer, tweaking out the parameters of the interface, getting their act together. They want their choreography to be influenced by the music and vice-versa.

From the needfinding process and from the analysis of possible scenarios it turned out that a fundamental requirement for this device was to be **customizable**, **easy to use** and able to **reproduce** what has been previously generated.

4 Usability Test

Since Max/MSP, the audio programming language, has a paid license, it was not possible to test the system on multiple computers. The test was carried out on two laptops (one running Windows 10 and the other one MacOS[3]) where the software was already installed. Another required software is the Arduino IDE, which is responsible for sending the sensors measurements to the serial buffer (USB).

Test To test and improve the performance and the usability of this system, a test has been submitted to participants. This project targets **two different groups of people**: the first one is composed primarily of electronic music producers and performers. The second one is a mixed audience of people that are interested in music generation, although is not their main focus, and do not have strong technological skills. These two targets were tested separately, with two

[3]Max/MSP exists only on these two operating systems.

different but similar tests. Everyone used headphones.
The first test, submitted to 4 music professionals, contained 4 tasks:

1. Connect the device to the computer and load the Arduino sketch on it.
2. Open the Max/MSP interface and start playing with the device.
3. Modify the patch in order make the light intensity sensor control the master volume.
4. Save the current patch configuration to a file in order to restore it later.

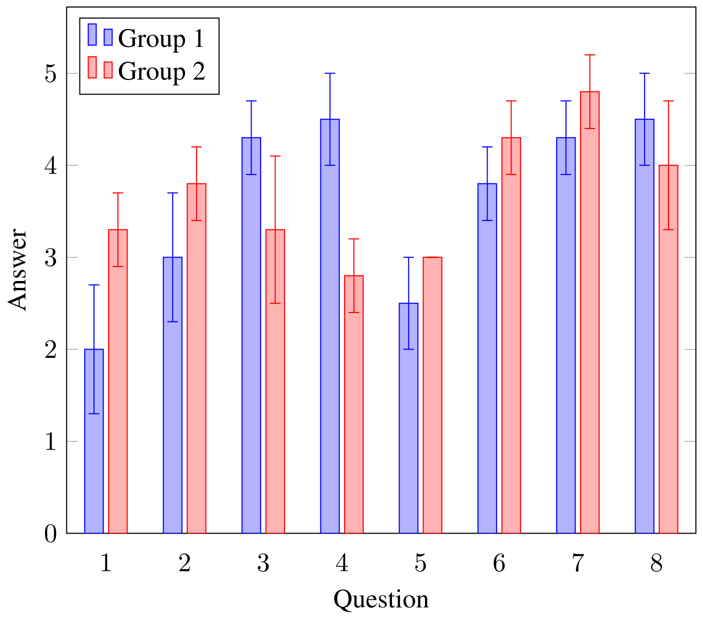
The second test, submitted to a a dancer, two computer scientists and a kindergarten teacher, contained 4 tasks as well. This second test was partially moderated since the majority of the audience needed technical support.

1. Connect the device to the computer and load the Arduino sketch on it.
2. Open the Max/MSP interface and start playing with the device.
3. Tweak the interface: turn on/off elements, use microphone recordings, randomize the music generation.
4. Save the audio produced to a file.

Results After performing the tasks, eight Single Ease Questions have been submitted to the users: the answers to these questions are encoded on a 5-point Likert scale, where 1 encodes the user strongly disagreeing with the question and 5 a strong agreement. The median, 3, encodes neutrality.

N	**Question**	μ	σ
1	Task completion required too much effort.	2.6	0.9
2	The Arduino device is accurate.	3.4	0.7
3	The Arduino device is easy to use.	3.8	0.8
4	I can immediately understand what I can control in the UI.	3.6	1.
5	I like the visual appeal of the UI.	2.8	0.4
6	I like the sounds I generated.	4	0.5
7	I learned to use the device better after a little while.	4.5	0.5
8	Overall, I am satisfied with this system.	4.3	0.7

As we can see fom the table above the audience responded positively to the proposed system. The highest rating has been reached on question 7, meaning that both the interface and the underlying patch are quite easy to learn after a short trial. Question number 8 gives a positive feedback as well, indicating that both the groups agree on the overall functioning of the system.
The lowest value with was reached in number 5, indicating that the proposed User Interface was slightly worse than sufficient.
In the following bar plot are shown the results differentiated by group type. The first group (*Group 1*) is the one composed by musicians and the second one is the mixed group. We can deduce that *Group 1* people tend to feel at ease with Arduino and are able to interpret the contents of the UI at eyesight, while *Group 2* found difficulties in items identifications. On question number 6 the second group seemed more satisfied with the sounds they were able to generate, with respect to the first one.



5 Conclusions & Furthur Work

In this project a simple music generation environment has been developed. An hardware device was built, along with the relative software, in order to exploit the connection between Human Computer Interaction and music production.

A discussion with users followed the usability test. All the musicians expressed their **approval** on the system, highlighting its capabilities and affirming their interest in the project. One of them even decided to *embed it in his live improvisation setup*. The second group was less confident with the system: the kindergarten teacher expressed her interest on using this device in a *teaching environment*. The dancer instead found that this system was too *raw* to be used in a professional setting, since the measurements outputted by the sensors were too inaccurate. All of them agreed on the fact that this system required to perform too many steps before being able to be used correctly. They also pointed out the fact that the Max/MSP programming language has a proprietary license that they would not have bought.

In order to enhance the performance of the system, a new set of **more accurate sensors** could be mounted on the Arduino board. The patch developed in this work is very easily extendable to new inputs, as this is meant to be a constant work-in-progress. One future development could also be a porting to **Pure Data** [5], another audio/visual programming language that, instead, is Free and Open Source Software.

References

[1] Simon Holland, Tom Mudd, Katie Wilkie-Mckenna, Andrew McPherson, and Marcelo M. Wanderley. *New Directions in Music and Human-Computer Interaction*. Springer Publishing Company, Incorporated, 1st edition, 2019.

[2] NIME. The International Conference on New Interfaces for Musical Expression. `https://www.nime.org/`, 2020.

[3] Brian Eno. *A Year with Swollen Appendices*. Faber and Faber, 1996.

[4] Cycling '74. Max/MSP. `https://cycling74.com/products/max/`, 2018. [Stable release].

[5] Patcher. Pure Data. `http://puredata.info/`, 2018. [Stable release].