# sASSIGNMENT REPORT

---

# HARPACINO

*“Who put this thing together?!”*

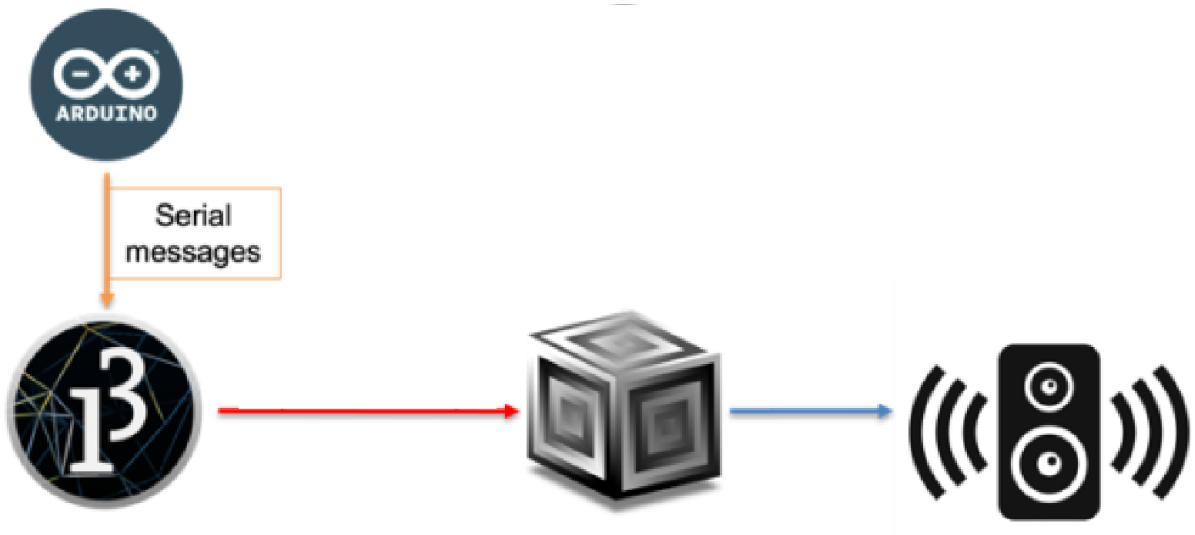
(Al Pacino, Scarface)

## General idea:

The CMS we decided to implement is composed by the following components:

- A synthesizer (Supercollider)
- A GUI (Processing)
- An input device (Arduino)

The first two elements are software components hosted on a personal computer while the third is actually a hardware device.




## Interaction design:

The *HARPACINO* has a box with six **laser beams**. The user interacts with it by blocking the light as if they were playing the strings of a harp. Each laser beam corresponds to a virtual string that the user can “pluck”.

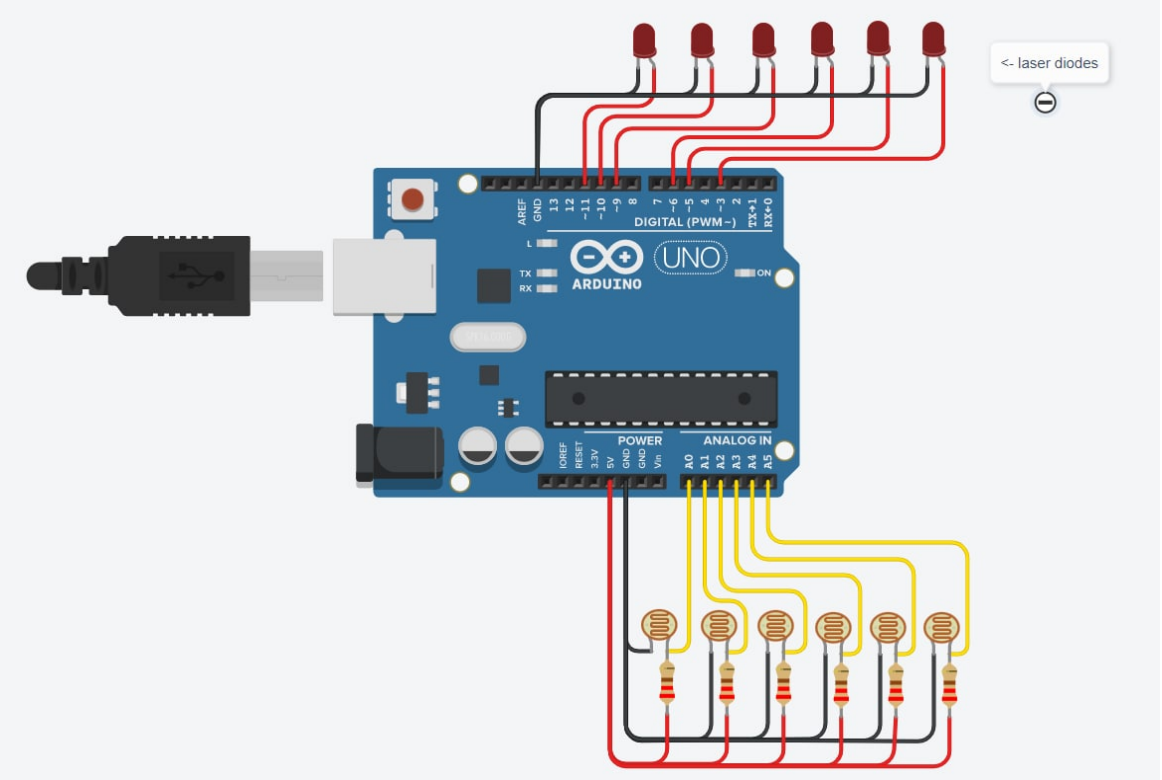
The Arduino sends a message to Processing via **Serial** when a string is played.

The strings correspond to the notes of the musical scale chosen by the user in the

Processing interface. The notes are then sent to Supercollider (via **OSC protocol**) to be synthesized.

## Arduino (hardware):

We decided to use an Arduino Uno for the task. The circuit is pretty simple as it involves six photoresistors connected in a **voltage divider** fashion. These are then connected to the Arduino's six analog inputs. Six laser diodes are also connected to the digital pins. In particular, the PWM capable ones (Pulse Width Modulation is used in order to limit power consumption and to prevent the diodes from breaking down). The laser diodes point to the photoresistors.




## Arduino (software):

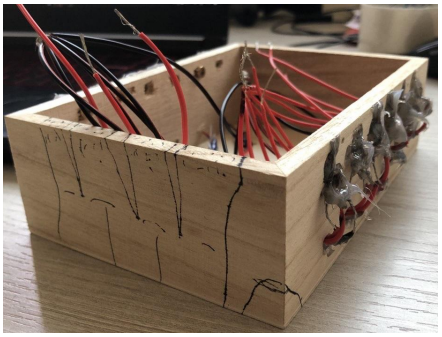
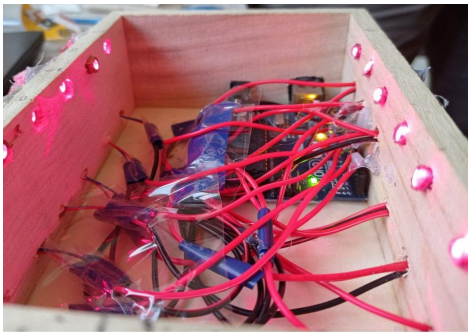
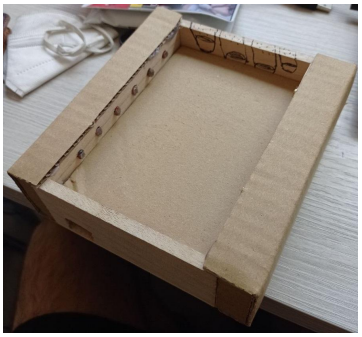
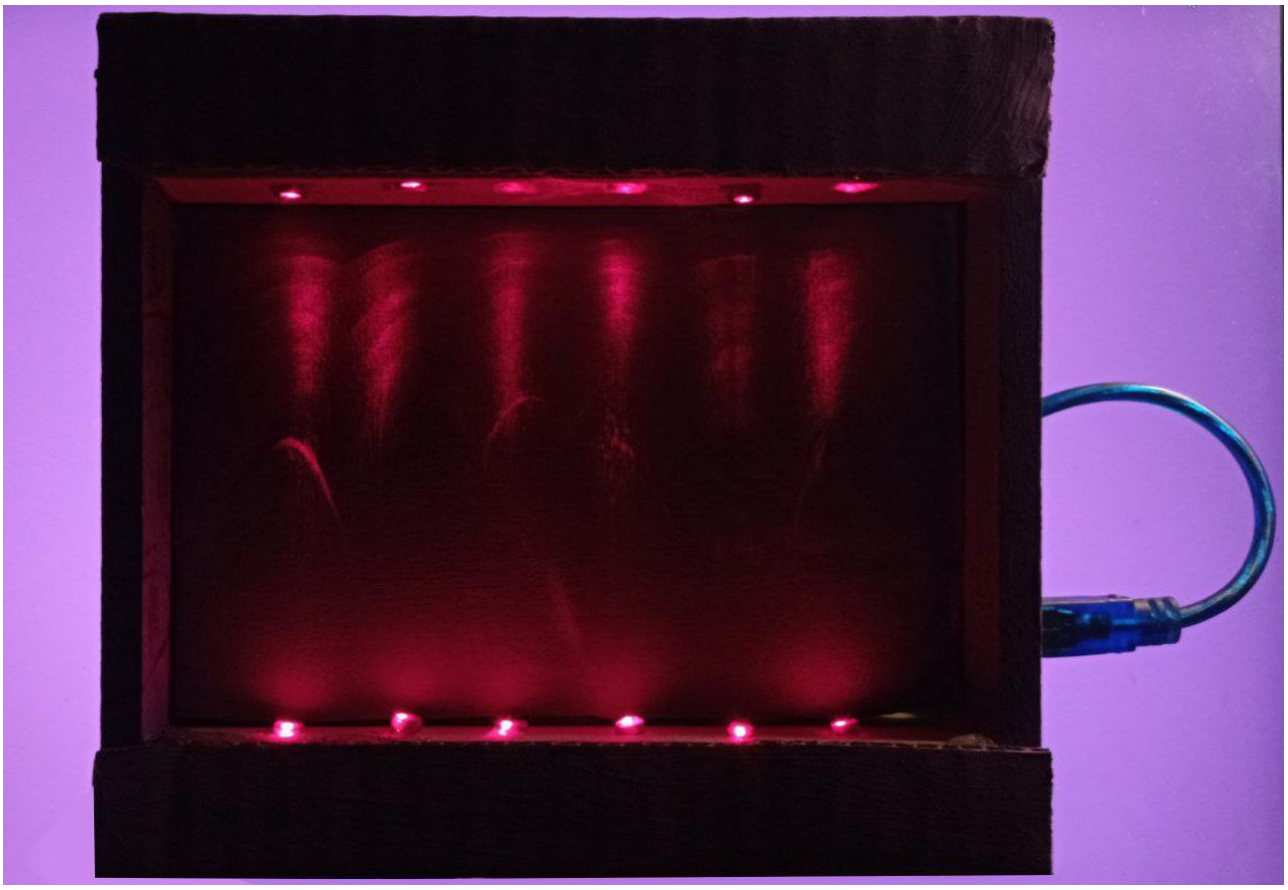
The Arduino reads from the analog pins (which return a value in the 0-1023 range) continuously and checks if the value is under a certain **threshold**. In our code we set this threshold to 280 but it is mostly arbitrary. When the previously mentioned condition is met, the Arduino sends the index of the corresponding "laser string" over the Serial protocol, to be read by the Processing sketch.

To implement some kind of "**debouncing**" and avoid sending the same message more than once, we use an array that keeps track of the state of the laser beams,

and only sends a Serial message if it has not been sent already. When the value returns above the threshold the state is reset.

## Hardware build:

A combination of disparate but also desperate techniques was used to build the *HARPACINO*, which can essentially be described as a wooden box, with holes drilled with a screwdriver, covered in a **mess** of hot glue finally hidden behind some cardboard.

**Processing-Supercollider Communication:**

This was brought about through the **OSC** library in processing and OSCdef function of Supercollider, which enabled us to send messages including the frequency of a single tone to Supercollider.

To do so, we needed to install and import oscP5 and netP5 in process and use the local host address as a path for communication.

On the supercollider end, we receive the message containing the frequency of the note that was received by processing from the arduino and sent to Supercollider.

**Synthesizer: HarpacinoSound:**

Harpacino sound is a synthesizer which was written using the mix of a simple SinOsc function with a particular envelope with a DynKlank which is a bank of resonators to which different values were tested until the right ones were at hand. As you will hear in the presentation this song has got a nice twang which in our opinion was totally appropriate for a laser harp.

## Processing:

The GUI and all business logic of the application have been implemented on the Processing side.

**Gui:**

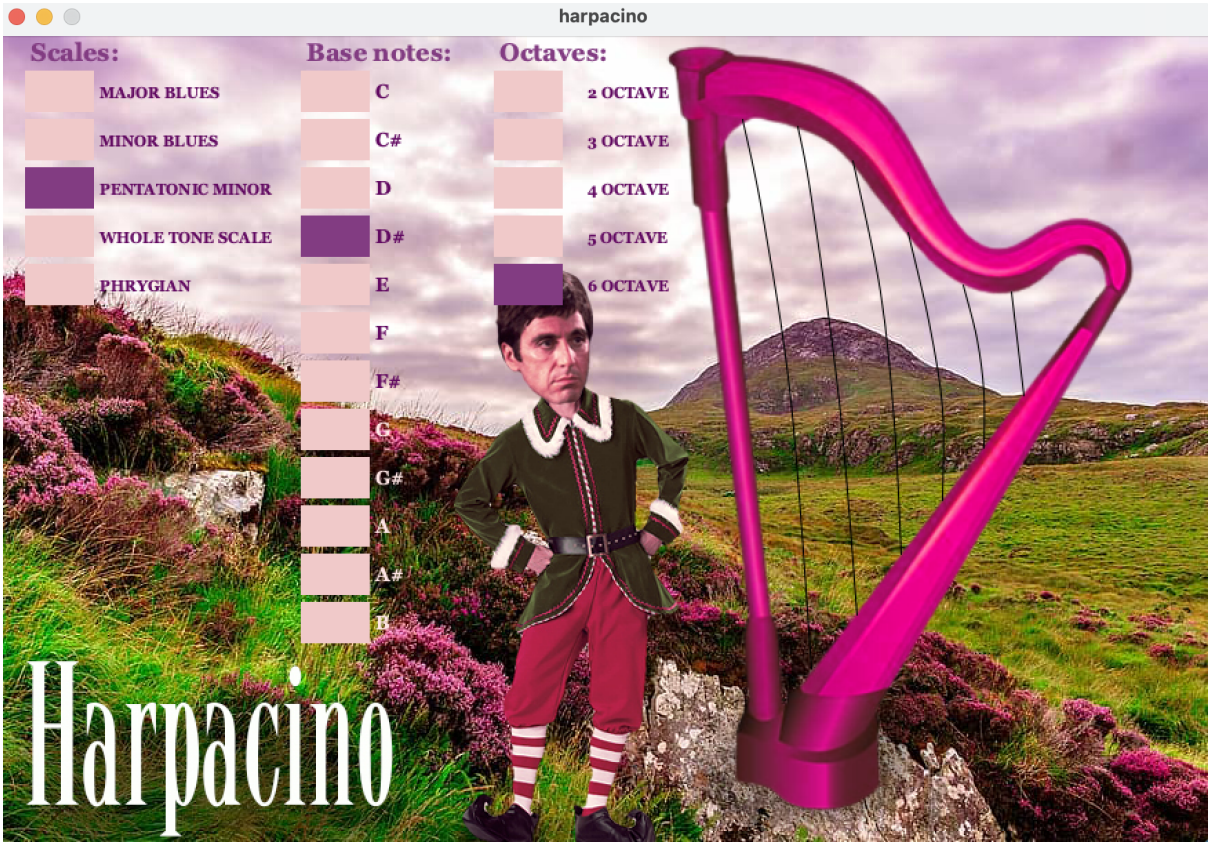
Harpacino offers several options to choose:
- Scale;
- Base note;
- Octave.

Scales have been selected according to our preferences so we decided to include the following: major and minor blues, pentatonic scale, whole note scale and phrygian mode. As well as the base note and the octave also can vary easily.

A slight animation has been added to the strings which start vibrating as the note is playing to give the sense of interaction with the instrument.

Considering implementation, controlP5 library has been used for the basic gui elements: radio buttons and text labels. The animation takes place in the draw() function. When the event ( that the specific string is plucked )  is triggered, we change the state of the particular string and it starts oscillating. After the pre-defined time ( which corresponds to the duration of the note ) the string stop oscillating. For

this purpose each string has its own timer to control how long the string should oscillate.




**Interaction logic:**

Also, the processing code is responsible for receiving messages from Arduino via serial port (basically, it receives the number of string which has been plucked), as well as for sending messages to SuperCollider via OSC protocol. Before this, the frequency has to be computed in accordance with the scale, base note and octave which the user chooses.