Politecnico di Milano
Computer Music: Languages and Systems
A.Y. 2022/2023
HW3

DESIGN AND IMPLEMENTATION OF A COMPUTER MUSIC SYSTEM

Emma Coletta 214391
Federico Ferreri 221557
Lorenzo Previati 222058

PURPOSE OF THE FINAL SYSTEM:

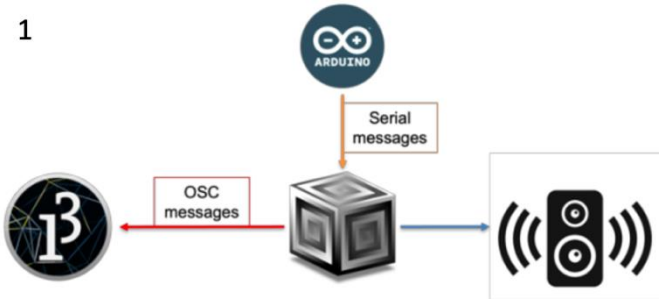
frequency modulation of a synthesizer controlled by a gray scale photosensitive sensor.

Needed tools:

- interaction unit system;
- computer music unit;
- graphical feedback unit.

For the implementation of our music system, we chose to use the first among the suggested possible schemes, involving the Arduino Uno (as our *interaction music system*), SuperCollider (as our *computer music unit*) and the exchange of OSC messages.

1



Using Arduino UNO as our control device, we share information to SuperCollider with serial messages providing the feedback events to Processing (our *graphical feedback unit*) with OSC messages.

SYSTEM DESCRIPTION

Tools:

- digital push button
- analog gray scale sensor
- Arduino UNO board

Connected to the Arduino, the digital push button has the simple task of activating or stopping our synthesizer from playing. If the numerical value representing the activation state passed on to Supercollider is equal to 0, the synthesizer is refrained from playing, otherwise the program then works on frequency modulation.

The gray scale sensor for Arduino is a composition of a photocell and an integrated white LED on board. We exploited its ability to measure the intensity of light from black to white to decide on which frequency our synthesizer would play. So, depending on the amount of light caught by the sensor, frequency modulation happens, resulting on the change of pitch of the playing synth.

Arduino board settings:

- digital push button: charged at 3.3V;
- analog gray scale sensor: charged at 5V.

Value range caught by the gray scale sensor: from 0 (when deactivated) to 1024 (max. brightness condition).

Frequency range for the frequency modulation: 440Hz to 880Hz.

Notes on SuperCollider code

We open the communication with the Serial Port of the computer to which the Arduino board is connected.

With the help of a routine, we store in an array the values caught by the sensor (luminosity condition) transmitted by the Arduino.

We then proceeded to define a Synth object, initialized at a frequency of 0Hz (not playing). As our last step, we created a control routine that sets the frequency for the playing synth.

The following portion of code sets the frequency for the playing synth, mapping between the range of values caught by the light sensor (from 0 to 1024) and a set value for the frequency of a note from central A to A one octave above it.

```
var noteFunction = { arg x; var y;
    y= case
    {x>0 && x<78.69}{440}
    {x>78.69 && x<157.38}{466.16}
    {x>157.38 && x<236.07}{493.88}
    {x>236.07 && x<314.76}{523.25}
    {x>314.76 && x<393.45}{554.37}
    {x>393.45 && x<462.14}{587.33}
    {x>462.14 && x<550.83}{622.25}
    {x>550.83 && x<629.52}{659.25}
    {x>629.52 && x<708.21}{698.46}
    {x>708.21 && x<786.9}{739.99}
    {x>786.9 && x<865.59}{783.99}
    {x>865.59 && x<944.28}{830.61}
    {x>944.28 && x<1024}{880};
};
```

Notes on Processing code

In the Processing code a visual representation of sound is created using a noise animation.

The oscEvent() function is the event handler for OSC messages. It is called whenever an OSC message is received. It checks for specific OSC address patterns ("/play" and "/stop") and extracts the frequency value of the synth from the message, if it matches the address pattern and the typetag "f", and stores it in the x variable.

```
void oscEvent(OscMessage msg) {
  if(msg.checkAddrPattern("/play")==true) {
    if (msg.checkTypetag("f")) { // Check if the message contains a single float value
      x = msg.get(0).floatValue(); // Get the float value from the message
      print(x);
    }
  }
  if(msg.checkAddrPattern("/stop")==true) { // If the sound is stopped
     x=0.00;
  }
}
```

The setup() function is called once at the beginning of the program. It sets up the size of the canvas, initializes the oscP5 object, and creates a font.

The draw() function is called repeatedly in a loop, and it handles the animation and drawing of the visual representation of sound.

The variables used to draw the animation are the following:

- yoff: A variable used to control the vertical movement of the noise animation.
- xoffIncrement and yoffIncrement: The amount by which the xoff and yoff variables are incremented in each frame of the animation.
- level1 and level2: Variables used to determine the range of y-values for the shape being drawn.

The code checks the value of x to determine the frequency of the sound being played. Based on the frequency value, the level1 and level2 variables are set to determine the range of y-values for the shape being drawn. Each frequency corresponds to a different range of y-values.
If x is equal to 0, it means the sound is stopped, and specific visual elements are drawn to indicate this state.
If x is not equal to 0, the sound is playing, and different visual elements are drawn to indicate this state.