```
/******************************************************************************
 ******************************************************************************
 *
 * STELI  / vybraphone / server + client
 *
 * Server code: gets values (notes and volume) FROM the glockenspiel one
 * > reproduces those sounds
 * > turns on LEDs of vybraphone notes that will play in harmony
 * > measures the wind and adds reverberation according to its power
 *
 * Client code: sends values (notes and volume) TO the glockenspiel installation
 * > gets flex pins and value
 * > activates solenoids that play an actual vybraphone
 *
 ******************************************************************************
 ******************************************************************************/


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Declarations
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

// Libraries
import processing.serial.*;
import cc.arduino.*;
import krister.Ess.*;
import processing.net.*;

Envelope myEnvelope;
Arduino arduino;
Serial port;
Client client;
Server server;

String soundFiles[] = { "glockFA.wav", "glockSOL.wav", "glockLA.wav",
"glockSIB.wav", "glockDO8a.wav" };

boolean serverRunning;

// ellipses positions
int posX = 100;
int posX2 = 450;
int posY1 = 50;
int posY2 = 120;
int posY3 = 190;
int posY4 = 260;
int posY5 = 330;
int posY6 = 400;
int posY7 = 470;
int radium = 50;

// sound variables
AudioChannel[] suono = new AudioChannel[20];
Envelope envelope_lv0;
Envelope envelope_lv1;
Envelope envelope_lv2;
Envelope envelope_lv3;
Envelope envelope_lv4;

EPoint[] env_lv0 = new EPoint[3];

// notes of the instrument on this side
```

```
int DO = 0;
int RE = 1;
int MI = 2;
int FA = 3;
int SOL = 4;

// NOTES in HARMONY
// each flex activates certain LEDs (which corresponds to certain notes)
// 0 turn off the LED, 1 turns it on
int ledHarmony[][] = {
  //DO, RE, MI, FA, SOL
  {
    1, 0, 0, 1, 0        }    // flex 1 plays FA: harmony with FA, DO
  ,
  {
    1, 0, 1, 0, 1        }    // flex 2 plays SOL: harmony with MI, SOL, DO
  ,
  {
    1, 0, 0, 1, 0        }    // flex 3 plays LA: harmony with FA, DO
  ,
  {
    0, 1, 0, 1, 0        }    // flex 4 plays SIb: harmony with RE, FA
  ,
  {
    1, 0, 1, 0, 1        }    // flex 5 plays DO8a: harmony with DO, MI, SOL
};


// pins corrisponding to certain notes (so certain LED) on the breadboard
int[] ledPins = new int[13];


// FLEX VARIABLES
///////////////////////////////////////////////////////////////////////////
int flexDefault[] = {       // flex defaul positions, ordered by pin number
  528, 482, 485, 507, 490};

int numFlex = 5;
int variazioneFlex = 7;     // amount of bending variation to still calculate
the flex sensor as not bended
int maxFlex = 700;          // max bending value of the flex

// booleans to know if im playing / bending / soundDelay
boolean[] stoSuonando = new boolean[numFlex];
boolean[] stoPiegando = new boolean[numFlex];
boolean[] stoContando = new boolean[numFlex];

float[] bend = new float[numFlex];     // bending value from the flex sensor
float[] newBend = new float[numFlex];
float[] volume = new float[numFlex];
int[]   conta = new int[numFlex];      // delay counter
float   millisDelay = 10;              // delay of sound in millis


// MICROPHONE VARIABLES
//////////////////////////////////////////////////////////////////////
int micPin = 5;          // Arduino PIN that controls the microphone
int microphone;          // Microphone value sent by the arduino
int micDefault = 580;    // Default value of the microphone when there's NO wind
int micVariation = 50;   // Variation of mic default value (+ and -) when
there's NO wind

// Absolute value of the wind:
```

```
// mic sends values above and under the default value; we just need the amound
of variation
int absWind = micDefault;
int absWindLow = micDefault;

int i;
int getFlex;                // flex pin     sent from the other computer
float getVolume;            // volume value sent from the other computer
float reverberation;
int windIntensity;


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Set up
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

void setup()
{
  // GRAPHICS
/////////////////////////////////////////////////////////////////////////
  size(550, 380);
  background(0);
  smooth();
  noFill();
  strokeWeight(3);
  stroke(0,255,50);

  //draw ellipses
  ellipse(posX, posY1, radium, radium);
  ellipse(posX, posY2, radium, radium);
  ellipse(posX, posY3, radium, radium);
  ellipse(posX, posY4, radium, radium);
  ellipse(posX, posY5, radium, radium);
  ellipse(posX2, posY1, radium, radium);
  ellipse(posX2, posY2, radium, radium);
  ellipse(posX2, posY3, radium, radium);
  ellipse(posX2, posY4, radium, radium);
  ellipse(posX2, posY5, radium, radium);

  frameRate(30);

  // SERVER / CLIENT
///////////////////////////////////////////////////////////////////
  // Starts a server on port 10002
  // Please note: only clients need to know a target IP address, the server
  // just listens to a given port
  server = new Server(this, 10002);
  //client = new Client(this, "172.16.248.227", 10002);
  serverRunning = true;

  println("server starting");

  // LED PINS
//////////////////////////////////////////////////////////////////////////
  // which LED's pin correspond to wich note
  ledPins[DO] = 11;
  ledPins[RE] = 10;
  ledPins[MI] = 9;
  ledPins[FA] =  8;
  ledPins[SOL] = 7;
  // ARDUINO
/////////////////////////////////////////////////////////////////////////
  arduino = new Arduino(this, Arduino.list()[0], 115200);
  arduino.pinMode(0, Arduino.INPUT);

  for (int i = 3; i <= 12; i++){
    arduino.pinMode(i, Arduino.OUTPUT);
  }

  // SOUNDS
/////////////////////////////////////////////////////////////////////////
  //load sounds
  Ess.start(this);

  suono[0] = new AudioChannel("glockFA.wav");
  suono[1] = new AudioChannel("glockSOL.wav");
  suono[2] = new AudioChannel("glockLA.wav");
  suono[3] = new AudioChannel("glockSIB.wav");
  suono[4] = new AudioChannel("glockDO8a.wav");
  suono[10] = new AudioChannel("vibDO.wav");
  suono[11] = new AudioChannel("vibRE.wav");
  suono[12] = new AudioChannel("vibMI.wav");
  suono[13] = new AudioChannel("vibFA.wav");
  suono[14] = new AudioChannel("vibSOL.wav");

  // ENVELOPE
/////////////////////////////////////////////////////////////////////////
  // sets 4 evelops effects with different intensity according to the wind
  // EPoint(time, amplitude);

  // wind = 0
  //EPoint[] env_lv0 = new EPoint[3];
  env_lv0[0] = new EPoint(0,1);
  env_lv0[1] = new EPoint(0,3);
  env_lv0[2] = new EPoint(3,0);
  envelope_lv0 = new Envelope(env_lv0);
}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Draw
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

void draw() {

  // SERVER
/////////////////////////////////////////////////////////////////////////
  if(client != null) {
    if (client.available() > 0) {
      String message = client.readString();
      println(message);
    }

    if(serverRunning == true) {
      //check for the next client in line with a new message
      Client thisClient = server.available();

      // is there a client?
      if(thisClient != null) {
        // check to see if there is a message from the client
        if(thisClient.available() > 0) {
          // read in the message
          String message = thisClient.readString();

          if(message.length() > 8) {
            getFlex = int(message.substring(0,1));
            getVolume = float(message.substring(7));

            // plays the sample sounds of given note and volume when a message
is received
            playSounds(getFlex, getVolume);
          }

          println(message);

          // this is a very simple server - it's write method it just broadcasts
to
          // every client connected
          server.write(message);
        }
      }
    }

    activeFlex();
    microphoneListener();
    mouseOver();
  }
}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Active Flex
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

void activeFlex(){
  for(i=0; i<numFlex; i++) {

    // FLEX IN DEFAULT POSITION
    // reset sounds, bending value and delay counter
    if(arduino.analogRead(i) > flexDefault[i]-variazioneFlex &&
arduino.analogRead(i) < flexDefault[i]+variazioneFlex) {
      stoSuonando[i] = false;
      stoPiegando[i] = false;
      stoContando[i] = false;
      conta[i] = 0;
      bend[i] = 0;
      println(i+": SONO FERMO --- " +arduino.analogRead(i));
    }

    // DELAY COUNTER
    // counts till millisDelay value, then stops
    if(stoContando[i] = true) {
      if(conta[i] < millisDelay) conta[i]++;
      else {
        stoContando[i] = false;
      }
    }

    // FLEX BENDED
    // detects that it's active, and starts counting
    if(arduino.analogRead(i) <= flexDefault[i]-variazioneFlex ||
arduino.analogRead(i) >= flexDefault[i]+variazioneFlex) {
      stoPiegando[i] = true;
      if(conta[i] < millisDelay) stoContando[i] = true;
      //println(i+": MI HAI PIEGATOOO -> " +arduino.analogRead(i)+" -
"+bend[i]+" --- c="+conta[i]);
    }

    // Calculates max bending achieved before the delay counter stops
    if(stoPiegando[i] == true && stoContando[i] == true) {
      newBend[i] = arduino.analogRead(i);
      if(newBend[i] > bend[i]) bend[i] = newBend[i];
    }

    // println(bend);

    // PLAY SOUND
    if(stoSuonando[i] == false && stoContando[i] == false) {
      if(bend[i] > 0) {
        stoSuonando[i] = true;
        volume[i] = map(bend[i], flexDefault[i], maxFlex, 0, 3);
        client.write(i + " and " + volume[i]);
        suono[i].volume(volume[i]);

        if (suono[i+10] != null){
          if (suono[i+10].state==Ess.PLAYING) {
            suono[i+10].stop();
          }
        }
        suono[i+10].play();
        //println(i+": PLAY! :D -- volume = "+bend[i]);
        conta[i] = 0;
      }
    }
  }
}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Microphone Listener
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

void microphoneListener() {

    microphone = arduino.analogRead(micPin);    // gets microphone value

    /*if(microphone > absWind) absWind = microphone;
      if(microphone < absWindLow) absWindLow = microphone;  */

    // if the wind is blowing
    if(microphone > micDefault + micVariation || microphone < micDefault -
micVariation) {
      absWind = abs(microphone - micDefault);
    } else {
      absWind = 0;
    }

    //println("mic --> "+microphone+" --- abs: "+absWind);

}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Mouse over  and Key Pressed
```

```
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

void mouseOver(){

  //first ellipse
  if (((posX < mouseX) && (mouseX < (posX + radium))) && ((posY1 < mouseY) &&
(mouseY < (posY1 + radium)))){
    playSounds(0, random(1,1));
  }


  //second ellipse
  if (((posX < mouseX) && (mouseX < (posX + radium))) && ((posY2 < mouseY) &&
(mouseY < (posY2 + radium)))){
  playSounds(1, random(1,1));
  }
  //third ellipse
  if (((posX < mouseX) && (mouseX < (posX + radium))) && ((posY3 < mouseY) &&
(mouseY < (posY3 + radium)))){
    playSounds(2, random(1,1));
  }
  //fourth ellipse
  if (((posX < mouseX) && (mouseX < (posX + radium))) && ((posY4 < mouseY) &&
(mouseY < (posY4 + radium)))){
    playSounds(3, random(1,1));
  }
  //fifth ellipse
  if (((posX < mouseX) && (mouseX < (posX + radium))) && ((posY5 < mouseY) &&
(mouseY < (posY5 + radium)))){
    playSounds(4, random(1,1));
  }


}

void keyPressed() {
  if(keyCode==UP) {
    playSounds(1,1);
  }
  if(keyCode==DOWN) {
    playSounds(2,1);
  }
  if(keyCode==LEFT) {
    playSounds(3,1);
  }
  if(keyCode==RIGHT) {
    playSounds(4,1);
  }
  if (key == 'c'){
    startClient();
  }
}

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Play Sounds
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

void playSounds(int whichSound, float whatVolume){

  // REVERBERATION
  // uses the amount of wind blowing: the more wind, the more riverberd
  reverberation = map(absWind, 0, 200, -0.2, 2);
  if(reverberation < 0) reverberation = 0;

  // if it was playing, it stops to allow the same sample to start again
  if (suono[whichSound] != null){
    if (suono[whichSound].state==Ess.PLAYING) {
      suono[whichSound].stop();
    }
  }

  suono[whichSound] = new AudioChannel(soundFiles[whichSound]);

  if(reverberation > 0.2) {
    env_lv0[0] = new EPoint(0,0);
  } else {
    env_lv0[0] = new EPoint(0,1);
  }
  env_lv0[1] = new EPoint(reverberation,3);
  envelope_lv0.points = env_lv0;

  envelope_lv0.filter(suono[whichSound]);

  suono[whichSound].volume(whatVolume);      // sets volume
  suono[whichSound].play();                  // play sounds
  println("PLAY -- reverber: "+reverberation+" -- volume: "+whatVolume);


  // LEDs
  // controls all the LEDs: if they are in harmony with the played not, they
turn on,
  // otherwise they turn off
  for(i=0; i<numFlex; i++) {

    if(ledHarmony[whichSound][i] == 0) {
      arduino.digitalWrite(ledPins[i], Arduino.HIGH);
      //println("LED: "+i+" --> ledPins["+i+"]");
    }
    else {
      arduino.digitalWrite(ledPins[i], Arduino.LOW);
    }
  }

}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// Start Client
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

void startClient(){

  //create a new client
  //this takes an IP address (localhost means connect to myself)
  //and a port number
  client = new Client(this, "172.16.248.227", 10002);
  println("client starting");
  client.write("ciao Vale");
}
```