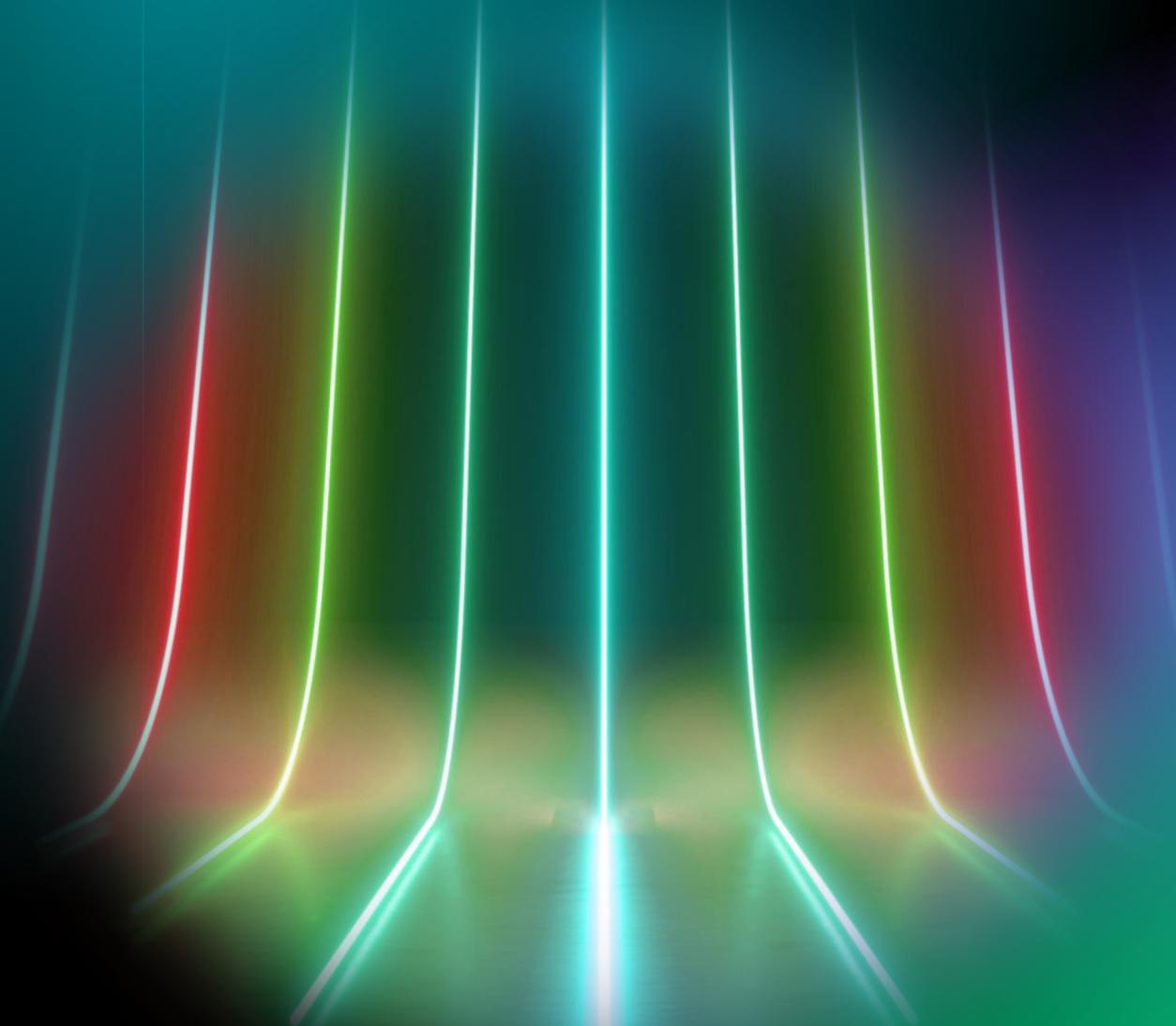


Arduino Processing - Integration Project

# Musical Lamp

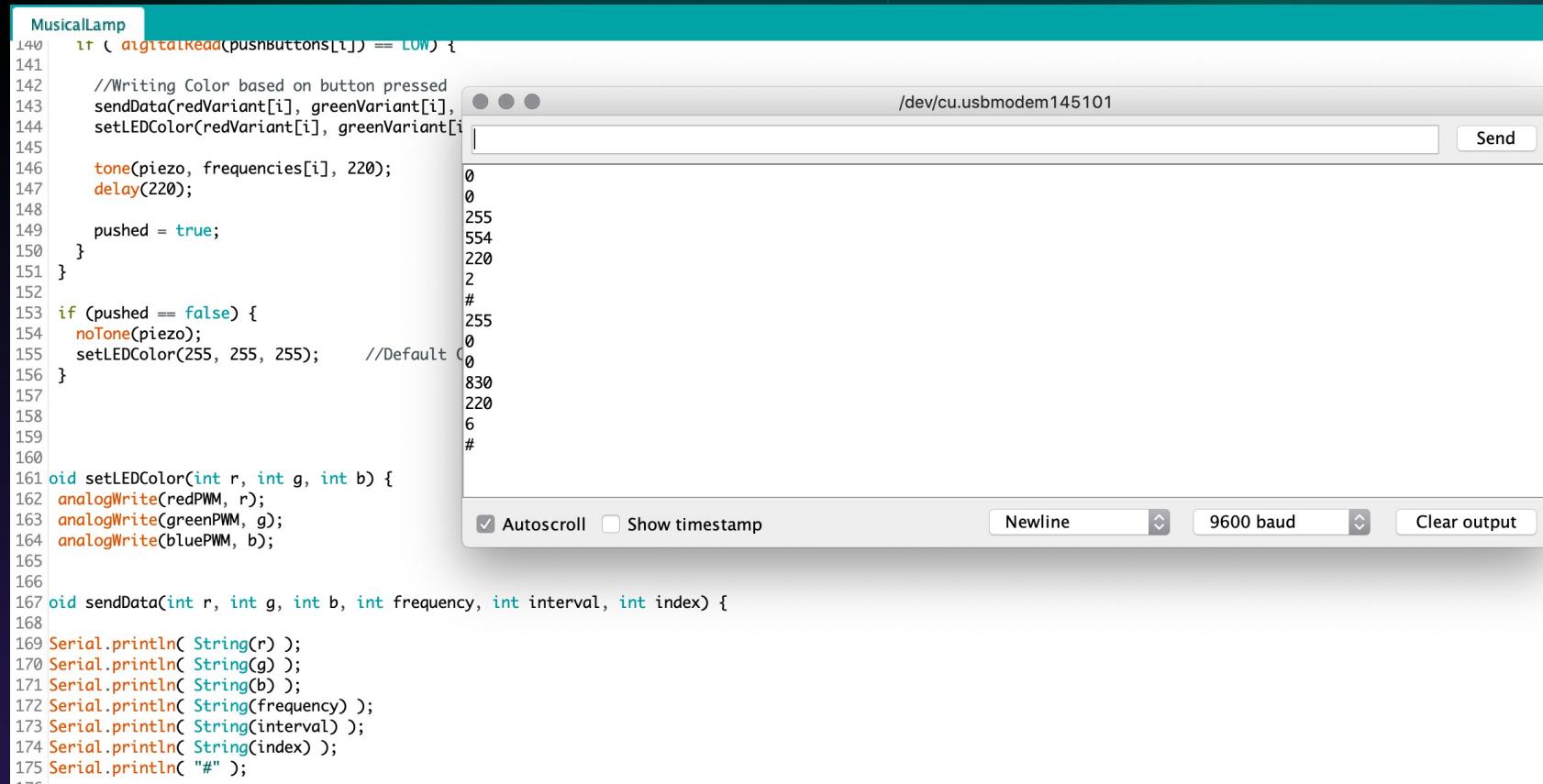
NARENDRA KONATHALA



# Serilog Data Sent

Every music note is sent with a set of values delimited by # as shown below.

Each record has R, G, B values, Frequency, Interval and index of musical note sent to processing Serial Event (Do, Re, Mi Fa, Sol, La, Ti, Do#).



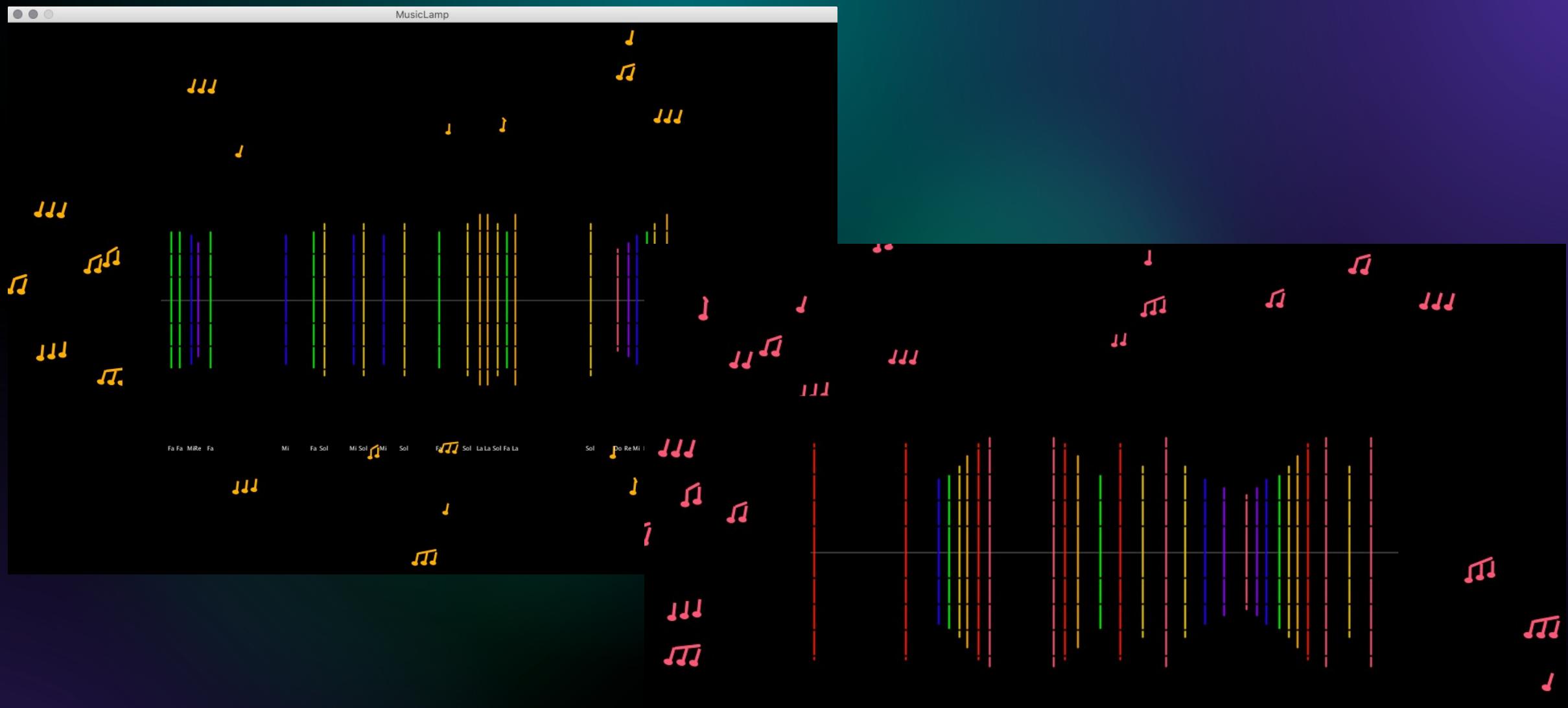
The screenshot shows a serial monitor window titled "MusicalLamp". The code in the editor is as follows:

```
140 if (digitalRead(pushButtons[i]) == LOW) {
141
142     //Writing Color based on button pressed
143     sendData(redVariant[i], greenVariant[i],
144     setLEDColor(redVariant[i], greenVariant[i]
145
146     tone(piezo, frequencies[i], 220);
147     delay(220);
148
149     pushed = true;
150 }
151 }
152
153 if (pushed == false) {
154     noTone(piezo);
155     setLEDColor(255, 255, 255);    //Default C
156 }
157
158
159
160
161 void setLEDColor(int r, int g, int b) {
162     analogWrite(redPWM, r);
163     analogWrite(greenPWM, g);
164     analogWrite(bluePWM, b);
165
166
167 void sendData(int r, int g, int b, int frequency, int interval, int index) {
168
169 Serial.println( String(r) );
170 Serial.println( String(g) );
171 Serial.println( String(b) );
172 Serial.println( String(frequency) );
173 Serial.println( String(interval) );
174 Serial.println( String(index) );
175 Serial.println( "#" );
```

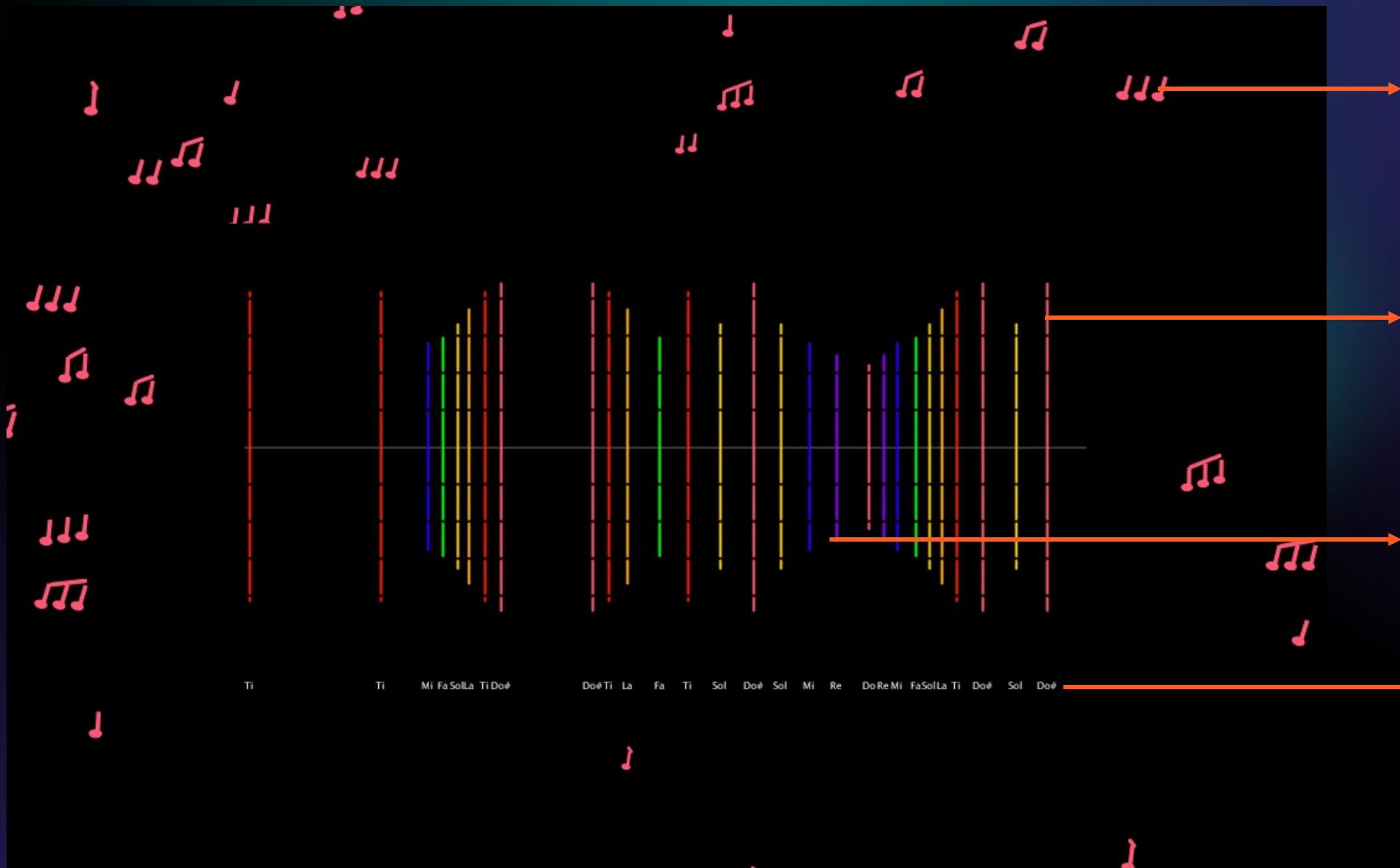
The serial monitor displays the following data:

Line	Data
0	0
1	0
2	255
3	554
4	220
5	2
6	#
7	255
8	0
9	0
10	830
11	220
12	6
13	#

Below the monitor are controls: "Autoscroll" (checked), "Show timestamp" (unchecked), "Newline" dropdown, "9600 baud" dropdown, and "Clear output" button.



<https://youtu.be/l7PAFI PYLes>



Musical Symbols created using basic shapes.

Its color changes based on the lamp light

Magnitude of each line represent the Frequency (Hz) of sound

Its color changes based on the lamp light

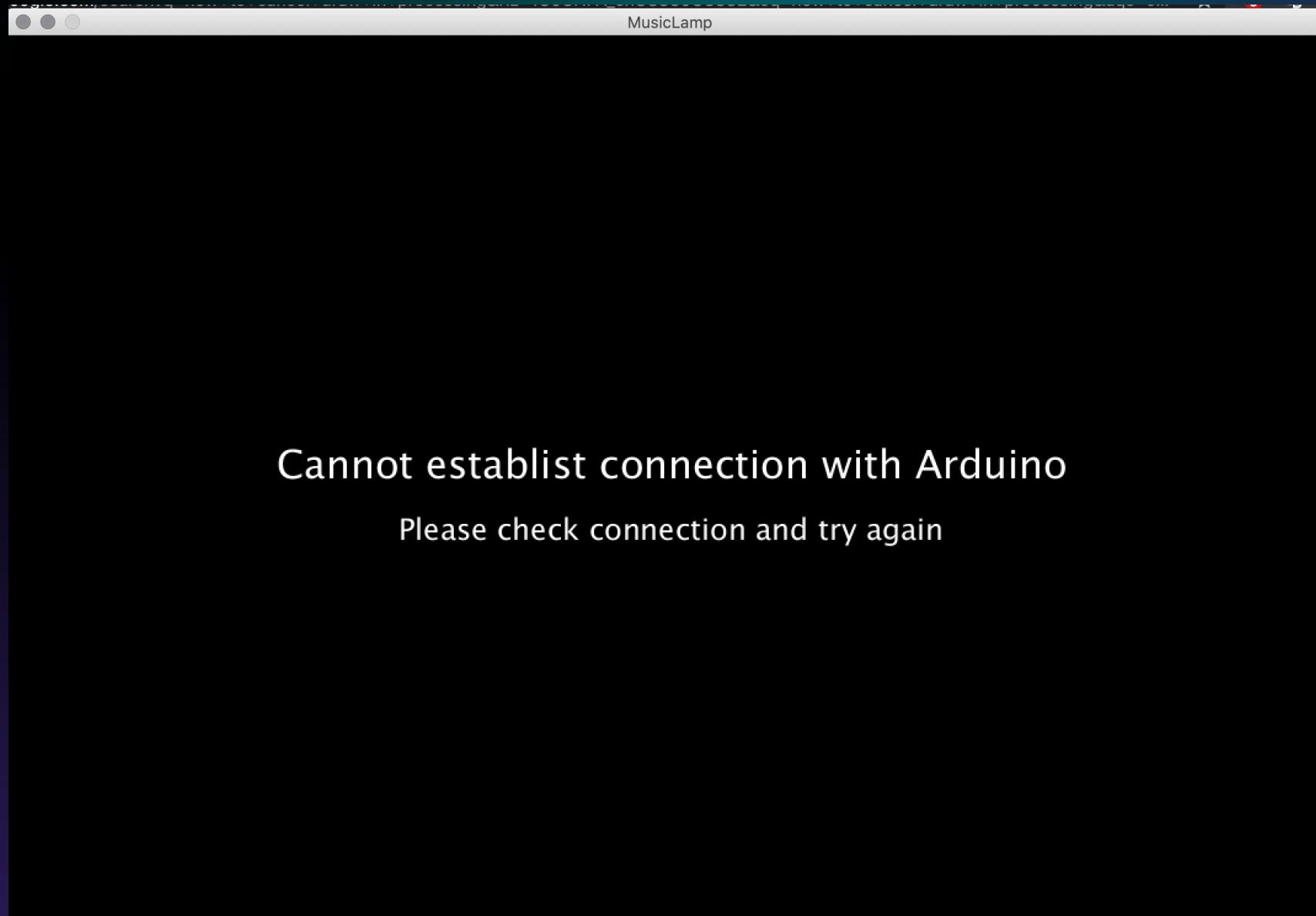
Time based graph , kind of sound wave. Gaps in between shows pause in music.

Music note value is written in the bottom for correlation.

Empty State is displayed when no data is transmitted for idle time of 10 secs.



Error Handling, when no port is available or Arduino not responding.



```
1 import processing.serial.*;
2 import java.util.LinkedList;
3 import java.util.Queue;
4 import java.time.*;
5
6 Serial arduinoPort;
7 final String portStart = "/dev/cu.usbmodem";
8
9 Queue<NoteData> queue = new LinkedList(); // Used to store the note
frequencies using 1st in 1st out method
10
11
12 final int num0fNotes = 50;
13 Note[] notes = new Note[num0fNotes];
14 NoteData activeNode; //Used to change the background
color
15 int offset = 200; //Used to centered the
frequencies of notes
16 ZonedDateTime pastEventTime; //To calculate idle time of
Arduino
17 String keyNote[] = {"Do", "Re", "Mi", "Fa", "Sol", "La", "Ti", "Do#"};
18 boolean badPort = false;
19
20 void setup() {
21     //fullScreen();
22     size(1080, 720);
23     background(0);
24
25     arduinoPort = getPort();
26
27     if (arduinoPort == null) { //Error handling - No Port found
28         translate(width/2, height/2);
29         textAlign(CENTER);
30         textSize(32);
31         text("Cannot establist connection with Arduino", 0, 0);
32         textSize(24);
33         text("Please check connection and try again", 0, 50);
34         noLoop();
35         badPort = true;
36         return;
37     }
38
39     for (int i = offset; i < width-offset; i++) {
40         queue.add(new NoteData()); //Initiate an empty queue for the
length of the graph
41     }
42
43     for (int i = 0; i < num0fNotes; i++) { //Notes - Reused for performance
44         Note note = new Note();
45         notes[i] = note;
46     }
47
48 }
49
50
51 void draw() {
52     if (badPort == true) { return ;}
53     background(0);
```

```

54
55 if (pastEventTime == null || isArduinoIdle(pastEventTime)) {
56     pushMatrix();
57     fill(125);
58     translate(width/2, height/2);
59     textAlign(CENTER);
60     textSize(32);
61     text("Arduino is idle", 0, 0);           //Notify user the idle case - when
62 no data is transmitted
63     textSize(16);
64     text("Perform some interactions to transmit data", 0, 50);
65     popMatrix();
66     return;
67 }
68
69 readSerialData(arduinoPort);           //Reads data to a queue
70
71 drawBackground();
72 drawFrequencyGraph();
73 drawAxis();
74
75
76
77 void drawFrequencyGraph() {           //Draw frequencies with color codes
78     int i = offset;                  //starts from offset to the length
of the queue
79
80     for (NoteData element : queue) {
81         pushMatrix();
82         translate(0, height/2);
83
84         if (element.frequency == 0) {
85             strokeWeight(1);
86             stroke(element.noteColor, 50);
87             line(i, 0, i, 1);           //Flat Line when no sound is made
88         } else {
89             float val = map(element.frequency, 0, 1200, 0, height / 4); //Map Hz
to Height
90             strokeWeight(2);
91             stroke(element.noteColor);
92             line(i, val, i, -1 * val);
93
94             fill(255);               //In addition to color code display
the note on
95             textAlign(CENTER);       //the bottom for better data
representation
96             textSize(8);
97             text( keyNote[element.index] , i, height / 4 + 16);
98         }
99         popMatrix();
100        i++;
101    }
102 }
103
104 void drawBackground() {
105     for (int j = 0; j < num0fNotes; j++) {
106         if (activeNode == null) {
107             notes[j].draw(-1, -1);

```

```

108 }else { //Highlight the note which is active
109     in the background
110     notes[j].draw(activeNode.index, activeNode.noteColor);
111 }
112
113 pushMatrix(); //Block middle area for Graph
114 translate(0, height/2);
115 noStroke();
116 fill(0);
117 rect(offset - 50, -1 * height / 4, width - 2 * (offset - 50), height / 2);
118 popMatrix();
119 }
120
121 void drawAxis() { //Frequencies - Y Axis
122     for (int i = -1200 ; i <= 1200; i = i + 200) {
123         if (i == 0) continue;
124         pushMatrix();
125         textAlign(CENTER);
126         textSize(8);
127         translate(0 ,height/2);
128         stroke(0);
129         strokeWeight(2);
130         int y = (int) map(i, -1200, 1200, -1 * height / 4, height / 4);
131         line(offset, y, width - offset, y);
132         popMatrix();
133     }
134 }
135
136 void serialEvent(Serial p) { //Used for calculating idle time of
Arduino
137     pastEventTime = ZonedDateTime.now();
138 }
139
140 boolean isArduinoIdle(ZonedDateTime oldTime) {
141     Duration duration = Duration.between(oldTime, ZonedDateTime.now());
142     if (duration.getSeconds() > 10) { //Idle timeout set to 10 sec...
143         return true; //Returns idle if no data is
transmitted for 10Secs
144     }
145     return false;
146 }
147
148 void readSerialData(Serial port){ //Function to read each set of Serial
data trasmitted from Arduino
149
150     String inBuffer = port.readStringUntil('#');
151     NoteData data = null;
152     if (inBuffer != null) {
153         data = new NoteData(inBuffer.trim());
154         activeNode = data;
155     } else {
156         data = new NoteData(); //No sound
157     }
158     queue.poll(); //Removes the 1st element from queue
159     queue.add(data); //Add new data to the end.
160 }
161
162

```

```

163 Serial getPort() { //Gets the Arduino port and creates a
164     listener
165     String[] portList = Serial.list();
166     for (int i = 0; i < portList.length; i++) {
167         if (portList[i].startsWith(portStart)) {
168             return new Serial(this, portList[i], 9600);
169         }
170     }
171     return null;
172 }
173 class NoteData { //Data structure to hold each note data
174     received from Arduino
175     public color noteColor;
176     public int frequency;
177     public int interval;
178     public int index;
179     public NoteData() { //Empty Note - default constructor
180         noteColor = color(255); //When no music or data transmitted
181     }
182     public NoteData(String rawText) { //override constructor
183         try {
184             String arr[] = rawText.split("\\n");
185             this.noteColor = color(Integer.parseInt(arr[0].trim()),
186             Integer.parseInt(arr[1].trim()), Integer.parseInt(arr[2].trim()));
187             this.frequency = Integer.parseInt(arr[3].trim());
188             this.interval = Integer.parseInt(arr[4].trim());
189             this.index = Integer.parseInt(arr[5].trim());
190         } catch (Exception e) {
191             println(e);
192         }
193     }
194 }
195 public class Note { //Data to represent background
196     public float x;
197     public float y;
198     int size = 0;
199     float distance;
200     float angle = 0;
201     final float boundRadius = width /2;
202     int rindex = (int) (Math.random() * 8);
203     float inclination = (float) Math.random() * TWO_PI / 20;
204     float scale = 0.1;
205
206     int type = (int) (Math.random() * 2 + 1);
207     int type1 = (int) (Math.random() * 3 + 1);
208
209     public Note() { //Random point on the canvas
210         angle = (float) Math.random() * TWO_PI;
211         distance = (float) Math.sqrt(Math.random()) * boundRadius + 200;
212         setX(); setY();
213     }
214
215     private void setX() {
216         x = (float) (distance * cos( (float) angle));
217     }
218

```

```
219
220     private void setY() {
221         y = (float) (distance * sin( (float) angle));
222     }
223
224     private void move() { //For 3D feel
225         distance = distance + 2;
226         size++;
227         if (distance > boundRadius + 200) {
228             distance = (float) Math.sqrt(Math.random()) * boundRadius+ 200;
229             scale = 0.1;
230         }
231         size = (int) ((distance) * 0.04);
232         scale = distance / 1000;
233
234         if ( size < 4 ) { size = 4 ;};
235         setX(); setY();
236     }
237
238     void draw(int index, color noteColor) {
239         move(); //Move the object
240         towards the circumference
241         pushMatrix(); //This function
242         draws random notes.
243         fill(noteColor);
244         translate(width/2, height/2);
245         scale(scale);
246         rotate(inclination);
247         noStroke();
248
249         for (int i = 0; i < type1; i++) {
250             rect(x + 20 * i, y - i * 5, 5, 25); //# string
251             ellipse(x + 20 * i, y - i * 5 + 25, 15, 10);
252
253             if (type > 1 && i == type1 -1) {
254                 strokeWeight(5);
255                 stroke(noteColor);
256                 line(x + 5, y, x + 20 * (i), y - 5 * (i +1)); //With Bar
257             }
258         }
259
260
261     }
262 }
```