DESIGN AND IMPLEMENTATION OF MICROCONTROLLER BASED MUSIC REACTIVELED CUBE

Wah Wah Myint[1], Sabal Phyu Thein[2]& Khin Mar Win[3]

Abstract

A music reactive LED cube is constructed by Arduino Microcontroller ATmega328P. The components used in this system are Arduino Uno Board, 512 LEDs, npn-transistors (2N2222), 74HC595 ICs, speaker, some resistors. The LEDs are designed into 8×8×8 cube which contains eight rows and eight columns. The output of the system shows various pattern designs of LEDs and also produces melody sound. The controlled program source code is written in C programming language. The error free source code is uploaded to the A Tmega 328P microcontroller and it controls the operation of the whole system. This system can be used for altar decoration, house and advertisement decoration.

Keywords: Arduino UNO, LEDs, Transistors (2N2222), 74HC595 ICs, Speaker

Introduction

Nowadays, LED cube is the next generation display piece of art and are adored among the electronic enthusiasts around the globe. LED cube is a set of components that can produce display of animation on the LED assembled part. The suitable software and hardware that will be involved are important aspects to take account about it. In this work, the design and construction of a music reactive 8×8×8 LED cube by Arduino microcontroller is constructed. The constructed music reactive LED cube contains eight layers. Each layer has sixty four LEDs and total eight layers have altogether five hundred and twelve LEDs. So the LED cube contains eight rows and eight columns[Chria. (2012)].The display panel of this circuit is 512 LEDs. This circuit is designed to display various pattern designs together with melody sound. Lighting and sound will be operated simultaneously. To control the functions of the cube circuit, Arduino Uno board is used in this work. The ICs used in this circuit are 74HC595 for the purpose as the layer selector and column selector. The block diagram of music reactive LED cube control system is as shown in Figure (1). The patterns are displayed on a 3D structure which is made up of copper rods. The message can be changed as per user need by rewriting the Arduino in built memory. The complete display system circuit is power supply run on 5V and 2A current which is provided externally. This unique way of displaying messages is very fantastic and eye catching therefore it is widely used in the field of advertising and toys, etc.

Construction and Operation of the System

Circuit Construction and Operation

In this circuit, Arduino microcontroller LEDs, npn transistors (2N2222), nine 74HC595 ICs, some resistors, speaker and +5V power supply are used for the circuit operation. The total combination of LED in the LED cube is 512 pieces. The arrangement of the LEDs must be carefully arranged so that the LED cube will firmly and steadier. Measuring the cathode wire of the LEDs, it's about 17 cm. The cathode wire will be connecting to the cube horizontally.

[1]Dr, Lecturer, Department of Physics, Meiktila University
[2] Lecturer, Department of Physics, Meiktila University
[3] Associate Professor, Department of Engineering Physics, Mandalay Technological University

The Anode wire is about 16 cm. The Anode wire will make all of the vertical connections. The multiplexing circuit controls all the data input from the microcontroller to the LED cube. Nine 74HC595 ICs are used in this work. There are 16 pins in 74HC595 ICs, pin 8 and 16 are connected to ground and power supply. Pin 11, 12, 14 are connected to Arduino. Pin 1, 2, 3, 4, 5, 6, 7, 15 are connected 220Ω resistors to join the Anode layer cube. Eight of ICs are connected in above way. Pin 1, 2, 3, 4, 5, 6, 7 and 15 of the last 74HC595 ICs are connected with npn (2N2222) transistors to join the cathode layer cube. The pin 10 and pin 13 of 74HC595 ICs must be held high and low for this operation.

Transistors have been used to be the switch for the LED connected to the ground. The transistors will be turned on if the data was transmitted to the transistors. The LEDs cathode leg will be attached to the ground and it also will be the controller for each level of the LED cube. The wires were made connections to each of the 8 horizontal layers and the wire down routed through the holes alongside the LED. These wires will be used with the cathode/layer control. Each row is labeled 0 to 7. This corresponds to one of each of the 74HC595 ICs. The Layer 1 wire connects to pin 1 of the cathode control header, all the way up to pin 8 connecting to layer 8's wire. The leads of the cable wire and get some solder on the bottom of the LEDs and one-by-one connect each of the 8 rows of LEDs to a connector. There are all together nine connectors, eight for LED anode connections and one for each horizontal layer's cathode of the LEDs. Since the cabling done connected all the cables to the drive board and give the system a test run. The output digital pin 1 is connected to speaker for melody sound. The circuit connection of audio system is shown in Figure (2) and the circuit diagram of 8-column connection in 8×8×8 LED cube is shown in Figure (3). After finishing verifying, the hex code of the sketch is appeared at the bottom window as shown in Figure (5). The photograph of the music reactive LED cube by Arduino Microcontroller is shown in Figure (6).

Power Supply Circuit

The power supply uses the locally available 3-terminal voltage regulator IC LM 7805. The booster transistor MJ 2955 can handle a maximum current of about 15 A but limited to about 2.0 A. During normal operation a current of about 25 mA flows through the IC, the rest current will flow through the booster transistor. The circuit diagram of power supply is shown in Figure (4).

Conclusion

The music reactive LED cube is successfully constructed. The constructed system contains 512 LEDs and each LED current limits is 25 mA. To operate this music reactive LED cube, 2 A current limit power supply is required. This power supply system is also constructed by the use of the regulator LM7805 IC and pnp power transistor MJ 2955. The LED cube is constructed by 8-row (layers) and 8-column. Various patterns of displays are produced by controlling the column and layer selection. To do this work, 74HC595 ICs are used as layer and column selectors. The various pattern designs together with melody sound are also produced by C programming source codes and this source codes are uploaded and then the system start to run. The display pattern can be 3-D graphics LED display, various fantastic pattern of blinking LEDs, light decoration and sound based on the Arduino source codes. This circuit of this work is very useful for altar decoration, home decoration, advertising and toys, etc.

Acknowledgement

The authors would like to acknowledge Professor DrKhinKhin Win, Head of Department of Physics, University of Yangon for her kind permission to carry out this work. The authors also would like to thank Professor DrHlaHlaThan (HOD) and Professor DrHla Win, Department of Physics, Meiktila University, for their kind permission to carry out this work.

References

Brain, M., Evans. (2007). "Arduino Programming Notebook". California: USA. Retrived December 16, 2016, from https://www.apress.com/us/book

Chria. (2012). "Building the 8×8×8 LED Cube Template". Retrived October 12,2016, fromhttps://www.pyroelectro.com/projects/8×8×8_led_cuve/parts.html

Githyp.com/ Anopmm/LED-CUBE. 8×8×8https://electronics.stackexchange.com/questions/20697/code-for-ripple-effect-of-an-8×8×8-led cube arduino

Jolli, C. (November 16, 2010). "Led Cube 8×8×8"._Retrived December 17, 2016, from https://www.instructables.com / id/LED Cube_8×8×8

Muhammad Faris, B. A. (January, 2015). "Hardware Design and Construction:Controlling the LED Cube Patterns with Arduino Input Signal".University TEKNOLOGI Malaysia.Retrived October 18,2016, from www.led-cube pattern/parts.html

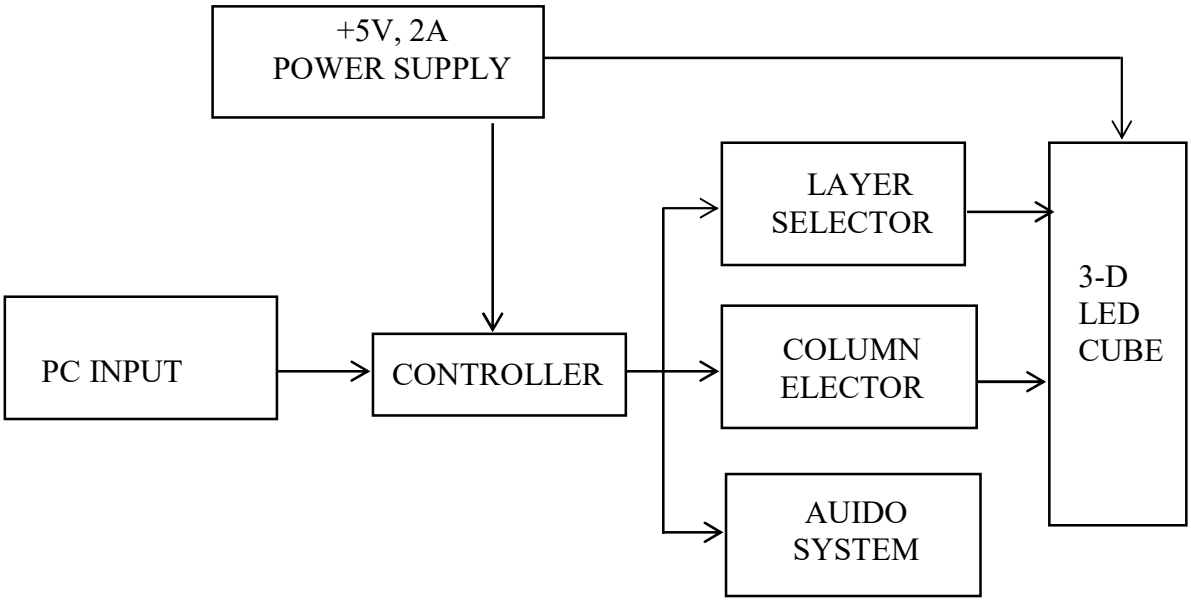

Figure 1 Block Diagram of Music Reactive LED Cube Control System

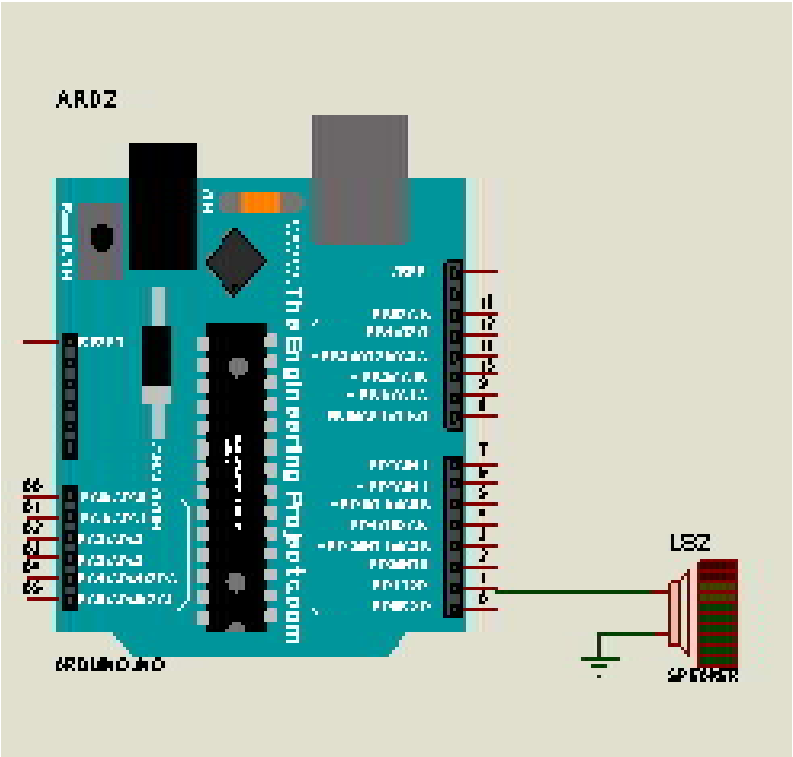

Figure 2 Circuit Diagram of Audio System

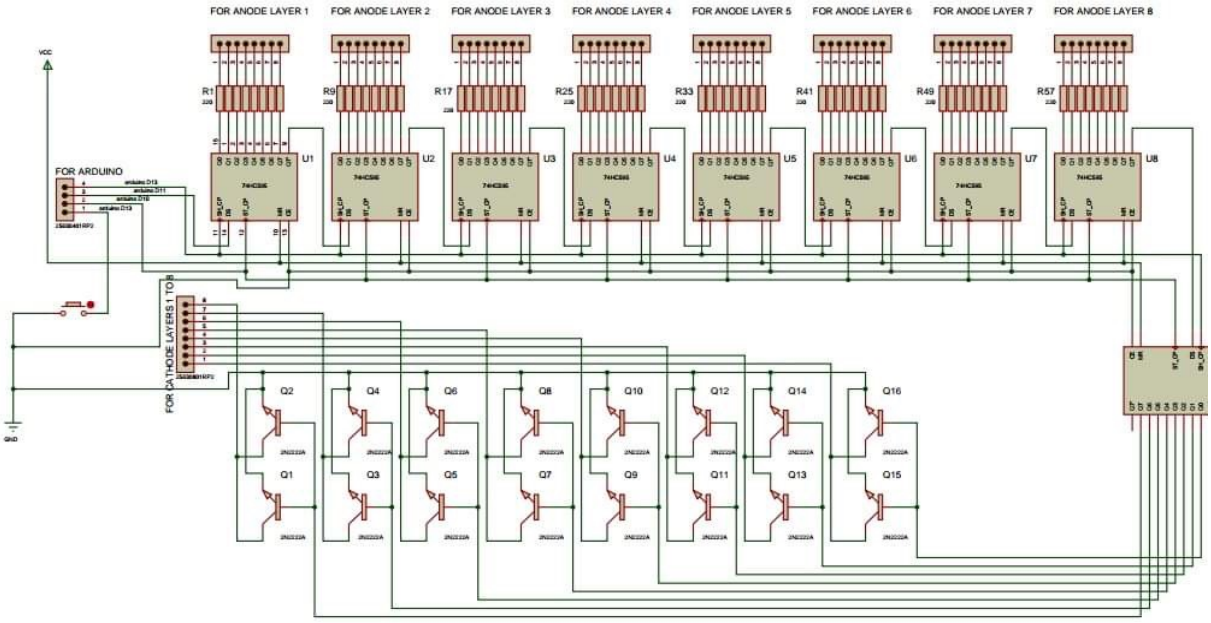

Figure 3 The circuit diagram of 8-column connection in 8×8×8 LED cube

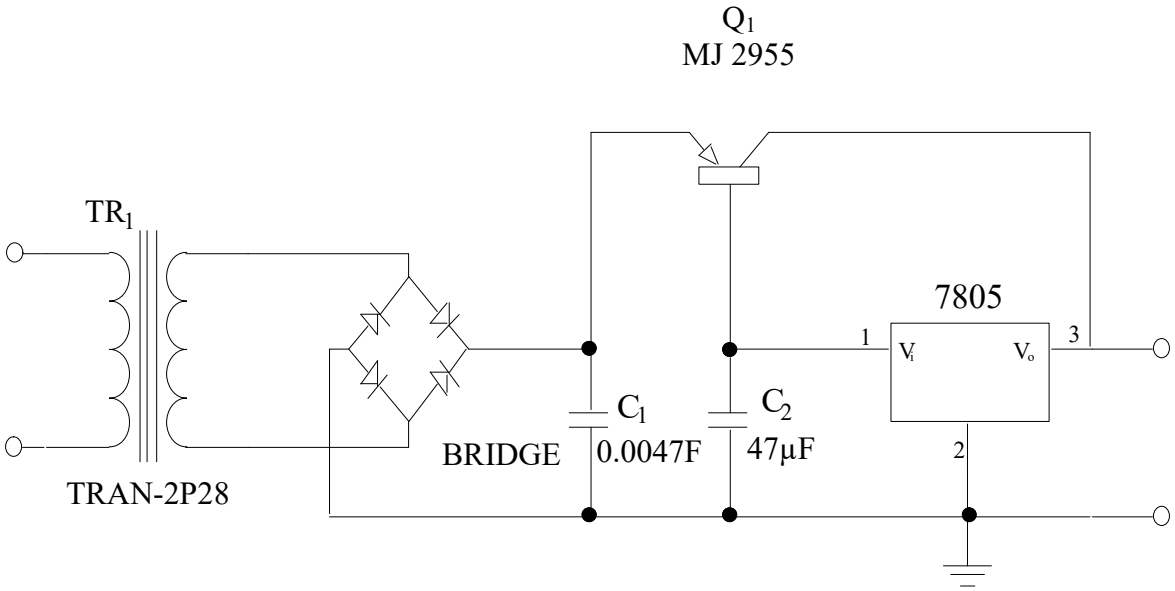

Figure 4 The circuit diagram of power supply

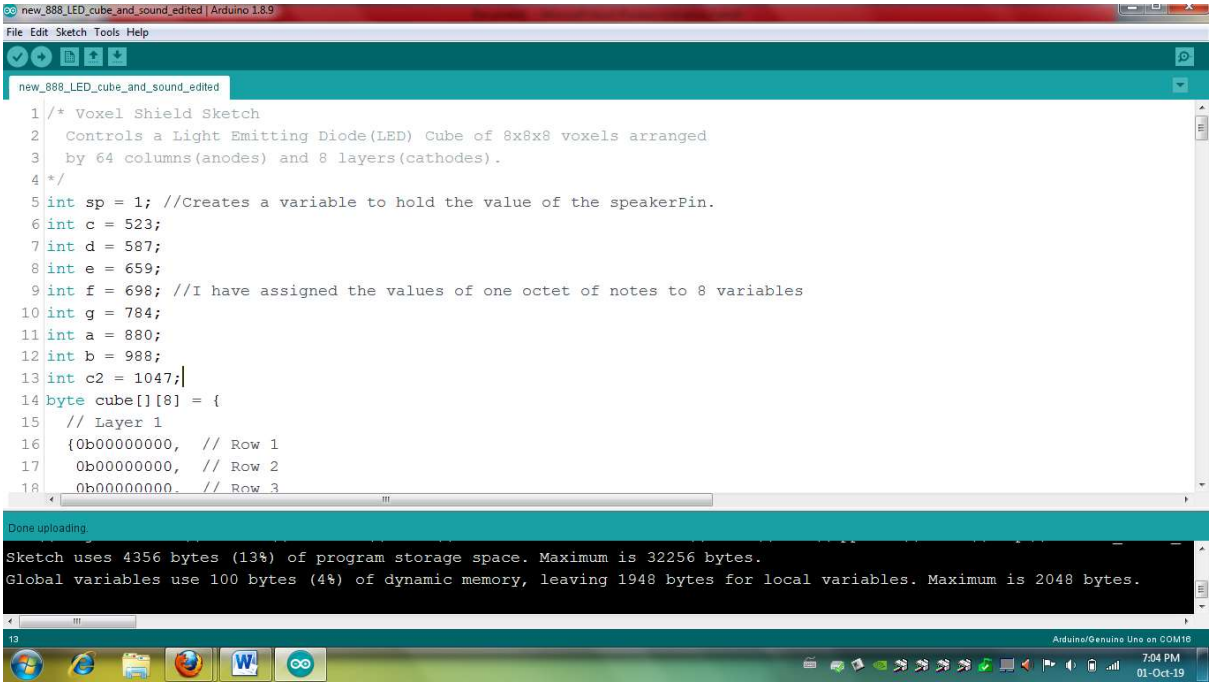

Figure 5 The photograph of the error free Arduino sketch showing .hex file after successfully verifying case

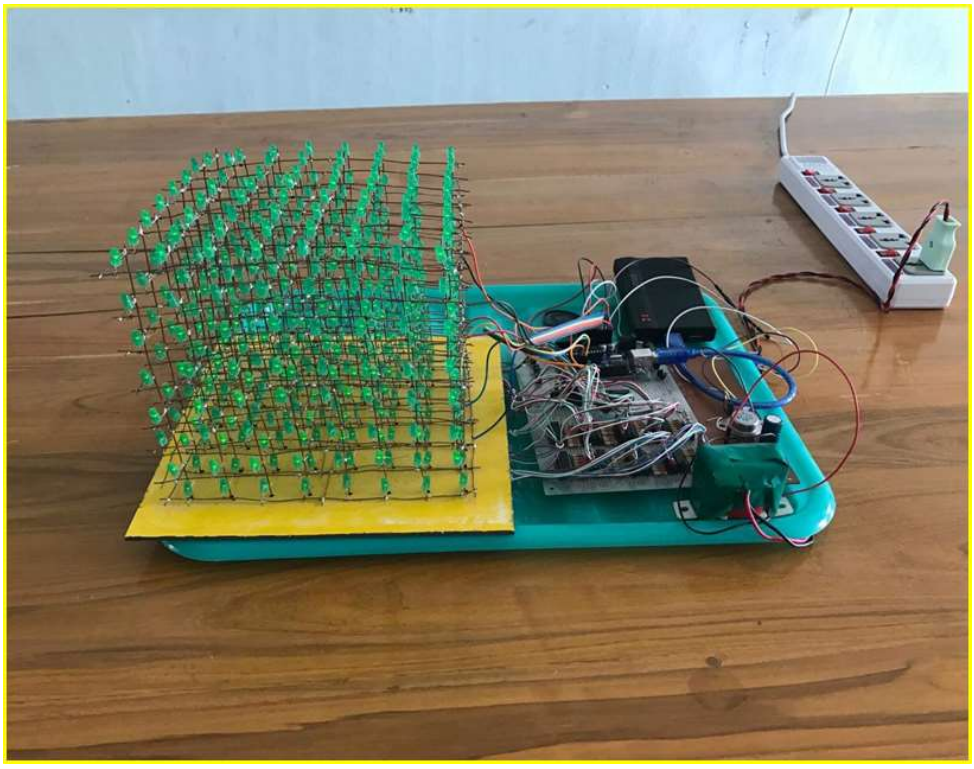

Figure 6 The photograph of the music reactive LED cube by Arduino Microcontroller

APPENDIX

Program Listing

```
/* Design and construction and music reactive LED cube*/
/* Voxel Shield Sketch
  Controls a Light Emitting Diode(LED) Cube of 8x8x8 voxels arranged
by 64 columns(anodes) and 8 layers(cathodes).
*/
intsp = 1; //Creates a variable to hold the value of the speakerPin.
int c = 523;
int d = 587;
int e = 659;
int f = 698; //I have assigned the values of one octet of notes to 8 variables
int g = 784;
int a = 880;
int b = 988;
int c2 = 1047;
byte cube[][8] = {
  // Layer 1
  {0b00000000,  // Row 1
   0b00000000,  // Row 2
   0b00000000,  // Row 3
   0b00000000,  // Row 4
   0b00000000,  // Row 5
   0b00000000,  // Row 6
   0b00000000,  // Row 7
   0b00000000}, // Row 8
  // Layer 2
  {0b00000000,  // Row 1
   0b00000000,  // Etc ...
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000},
  // Layer 3
  {0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000},
  // Layer 4
  {0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000},
  // Layer 5
  {0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
   0b00000000,
```

```
  0b00000000,
  0b00000000,
  0b00000000},
 // Layer 6
 {0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000},
 // Layer 7
 {0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000},
 // Layer 8
 {0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000}
};
// Shift Register pin assignments
intdataPin    = 11;
intclockPin    = 13;
intlatchPin    = 10;
intmasterClear = 4;
// Cube indicies
introwIndex   = 0;       // used for shifting out the data
intlayerIndex = 0;
void all(){
  cube[0][0] = 255; cube[1][0] = 255; cube[2][0] = 255; cube[3][0] = 255; cube[4][0] = 255; cube[5][0] = 255;
cube[6][0] = 255; cube[7][0] = 255;
  cube[0][1] = 255; cube[1][1] = 255; cube[2][1] = 255; cube[3][1] = 255; cube[4][1] = 255; cube[5][1] = 255;
cube[6][1] = 255; cube[7][1] = 255;
  cube[0][2] = 255; cube[1][2] = 255; cube[2][2] = 255; cube[3][2] = 255; cube[4][2] = 255; cube[5][2] = 255;
cube[6][2] = 255; cube[7][2] = 255;
  cube[0][3] = 255; cube[1][3] = 255; cube[2][3] = 255; cube[3][3] = 255; cube[4][3] = 255; cube[5][3] = 255;
cube[6][3] = 255; cube[7][3] = 255;
  cube[0][4] = 255; cube[1][4] = 255; cube[2][4] = 255; cube[3][4] = 255; cube[4][4] = 255; cube[5][4] = 255;
cube[6][4] = 255; cube[7][4] = 255;
  cube[0][5] = 255; cube[1][5] = 255; cube[2][5] = 255; cube[3][5] = 255; cube[4][5] = 255; cube[5][5] = 255;
cube[6][5] = 255; cube[7][5] = 255;
  cube[0][6] = 255; cube[1][6] = 255; cube[2][6] = 255; cube[3][6] = 255; cube[4][6] = 255; cube[5][6] = 255;
cube[6][6] = 255; cube[7][6] = 255;
  cube[0][7] = 255; cube[1][7] = 255; cube[2][7] = 255; cube[3][7] = 255; cube[4][7] = 255; cube[5][7] = 255;
cube[6][7] = 255; cube[7][7] = 255;
}
void none(){
  cube[0][0] = 0; cube[1][0] = 0; cube[2][0] = 0; cube[3][0] = 0; cube[4][0] = 0; cube[5][0] = 0; cube[6][0] = 0;
cube[7][0] = 0;
```

```
  cube[0][1] = 0; cube[1][1] = 0; cube[2][1] = 0; cube[3][1] = 0; cube[4][1] = 0; cube[5][1] = 0; cube[6][1] = 0;
cube[7][1] = 0;
  cube[0][2] = 0; cube[1][2] = 0; cube[2][2] = 0; cube[3][2] = 0; cube[4][2] = 0; cube[5][2] = 0; cube[6][2] = 0;
cube[7][2] = 0;
  cube[0][3] = 0; cube[1][3] = 0; cube[2][3] = 0; cube[3][3] = 0; cube[4][3] = 0; cube[5][3] = 0; cube[6][3] = 0;
cube[7][3] = 0;
  cube[0][4] = 0; cube[1][4] = 0; cube[2][4] = 0; cube[3][4] = 0; cube[4][4] = 0; cube[5][4] = 0; cube[6][4] = 0;
cube[7][4] = 0;
  cube[0][5] = 0; cube[1][5] = 0; cube[2][5] = 0; cube[3][5] = 0; cube[4][5] = 0; cube[5][5] = 0; cube[6][5] = 0;
cube[7][5] = 0;
  cube[0][6] = 0; cube[1][6] = 0; cube[2][6] = 0; cube[3][6] = 0; cube[4][6] = 0; cube[5][6] = 0; cube[6][6] = 0;
cube[7][6] = 0;
  cube[0][7] = 0; cube[1][7] = 0; cube[2][7] = 0; cube[3][7] = 0; cube[4][7] = 0; cube[5][7] = 0; cube[6][7] = 0;
cube[7][7] = 0;
}
void box(){
  cube[0][0] = 0b11111111; cube[1][0] = 0b10000001; cube[2][0] = 0b10000001; cube[3][0] = 0b10000001;
cube[4][0] = 0b10000001; cube[5][0] = 0b10000001; cube[6][0] = 0b10000001; cube[7][0] = 0b11111111;
  cube[0][1] = 0b10000001; cube[1][1] = 0b00000000; cube[2][1] = 0b00000000; cube[3][1] = 0b00000000;
cube[4][1] = 0b00000000; cube[5][1] = 0b00000000; cube[6][1] = 0b00000000; cube[7][1] = 0b10000001;
  cube[0][2] = 0b10000001; cube[1][2] = 0b00000000; cube[2][2] = 0b00000000; cube[3][2] = 0b00000000;
cube[4][2] = 0b00000000; cube[5][2] = 0b00000000; cube[6][2] = 0b00000000; cube[7][2] = 0b10000001;
  cube[0][3] = 0b10000001; cube[1][3] = 0b00000000; cube[2][3] = 0b00000000; cube[3][3] = 0b00000000;
cube[4][3] = 0b00000000; cube[5][3] = 0b00000000; cube[6][3] = 0b00000000; cube[7][3] = 0b10000001;
  cube[0][4] = 0b10000001; cube[1][4] = 0b00000000; cube[2][4] = 0b00000000; cube[3][4] = 0b00000000;
cube[4][4] = 0b00000000; cube[5][4] = 0b00000000; cube[6][4] = 0b00000000; cube[7][4] = 0b10000001;
  cube[0][5] = 0b10000001; cube[1][5] = 0b00000000; cube[2][5] = 0b00000000; cube[3][5] = 0b00000000;
cube[4][5] = 0b00000000; cube[5][5] = 0b00000000; cube[6][5] = 0b00000000; cube[7][5] = 0b10000001;
  cube[0][6] = 0b10000001; cube[1][6] = 0b00000000; cube[2][6] = 0b00000000; cube[3][6] = 0b00000000;
cube[4][6] = 0b00000000; cube[5][6] = 0b00000000; cube[6][6] = 0b00000000; cube[7][6] = 0b10000001;
  cube[0][7] = 0b11111111; cube[1][7] = 0b10000001; cube[2][7] = 0b10000001; cube[3][7] = 0b10000001;
cube[4][7] = 0b10000001; cube[5][7] = 0b10000001; cube[6][7] = 0b10000001; cube[7][7] = 0b11111111;
}
void box2(){
  cube[0][0] = 0b00000000; cube[1][0] = 0b00000000; cube[2][0] = 0b00000000; cube[3][0] = 0b00000000;
cube[4][0] = 0b00000000; cube[5][0] = 0b00000000; cube[6][0] = 0b00000000; cube[7][0] = 0b00000000;
  cube[0][1] = 0b00000000; cube[1][1] = 0b01111110; cube[2][1] = 0b01000010; cube[3][1] = 0b01000010;
cube[4][1] = 0b01000010; cube[5][1] = 0b01000010; cube[6][1] = 0b01111110; cube[7][1] = 0b00000000;
  cube[0][2] = 0b00000000; cube[1][2] = 0b01000010; cube[2][2] = 0b00000000; cube[3][2] = 0b00000000;
cube[4][2] = 0b00000000; cube[5][2] = 0b00000000; cube[6][2] = 0b01000010; cube[7][2] = 0b00000000;
  cube[0][3] = 0b00000000; cube[1][3] = 0b01000010; cube[2][3] = 0b00000000; cube[3][3] = 0b00000000;
cube[4][3] = 0b00000000; cube[5][3] = 0b00000000; cube[6][3] = 0b01000010; cube[7][3] = 0b00000000;
  cube[0][4] = 0b00000000; cube[1][4] = 0b01000010; cube[2][4] = 0b00000000; cube[3][4] = 0b00000000;
cube[4][4] = 0b00000000; cube[5][4] = 0b00000000; cube[6][4] = 0b01000010; cube[7][4] = 0b00000000;
  cube[0][5] = 0b00000000; cube[1][5] = 0b01000010; cube[2][5] = 0b00000000; cube[3][5] = 0b00000000;
cube[4][5] = 0b00000000; cube[5][5] = 0b00000000; cube[6][5] = 0b01000010; cube[7][5] = 0b00000000;
  cube[0][6] = 0b00000000; cube[1][6] = 0b01111110; cube[2][6] = 0b01000010; cube[3][6] = 0b01000010;
cube[4][6] = 0b01000010; cube[5][6] = 0b01000010; cube[6][6] = 0b01111110; cube[7][6] = 0b00000000;
  cube[0][7] = 0b00000000; cube[1][7] = 0b00000000; cube[2][7] = 0b00000000; cube[3][7] = 0b00000000;
cube[4][7] = 0b00000000; cube[5][7] = 0b00000000; cube[6][7] = 0b00000000; cube[7][7] = 0b00000000;
}
void box3(){
  cube[0][0] = 0b00000000; cube[1][0] = 0b00000000; cube[2][0] = 0b00000000; cube[3][0] = 0b00000000;
cube[4][0] = 0b00000000; cube[5][0] = 0b00000000; cube[6][0] = 0b00000000; cube[7][0] = 0b00000000;
  cube[0][1] = 0b00000000; cube[1][1] = 0b00000000; cube[2][1] = 0b00000000; cube[3][1] = 0b00000000;
cube[4][1] = 0b00000000; cube[5][1] = 0b00000000; cube[6][1] = 0b00000000; cube[7][1] = 0b00000000;
  cube[0][2] = 0b00000000; cube[1][2] = 0b00000000; cube[2][2] = 0b00111100; cube[3][2] = 0b00100100;
cube[4][2] = 0b00100100; cube[5][2] = 0b00111100; cube[6][2] = 0b00000000; cube[7][2] = 0b00000000;
  cube[0][3] = 0b00000000; cube[1][3] = 0b00000000; cube[2][3] = 0b00100100; cube[3][3] = 0b00000000;
cube[4][3] = 0b00000000; cube[5][3] = 0b00100100; cube[6][3] = 0b00000000; cube[7][3] = 0b00000000;
```

```
  cube[0][4] = 0b00000000; cube[1][4] = 0b00000000; cube[2][4] = 0b00100100; cube[3][4] = 0b00000000;
cube[4][4] = 0b00000000; cube[5][4] = 0b00100100; cube[6][4] = 0b00000000; cube[7][4] = 0b00000000;
  cube[0][5] = 0b00000000; cube[1][5] = 0b00000000; cube[2][5] = 0b00111100; cube[3][5] = 0b00100100;
cube[4][5] = 0b00100100; cube[5][5] = 0b00111100; cube[6][5] = 0b00000000; cube[7][5] = 0b00000000;
  cube[0][6] = 0b00000000; cube[1][6] = 0b00000000; cube[2][6] = 0b00000000; cube[3][6] = 0b00000000;
cube[4][6] = 0b00000000; cube[5][6] = 0b00000000; cube[6][6] = 0b00000000; cube[7][6] = 0b00000000;
  cube[0][7] = 0b00000000; cube[1][7] = 0b00000000; cube[2][7] = 0b00000000; cube[3][7] = 0b00000000;
cube[4][7] = 0b00000000; cube[5][7] = 0b00000000; cube[6][7] = 0b00000000; cube[7][7] = 0b00000000;
}
void box4(){
  cube[0][0] = 0b00000000; cube[1][0] = 0b00000000; cube[2][0] = 0b00000000; cube[3][0] = 0b00000000;
cube[4][0] = 0b00000000; cube[5][0] = 0b00000000; cube[6][0] = 0b00000000; cube[7][0] = 0b00000000;
  cube[0][1] = 0b00000000; cube[1][1] = 0b00000000; cube[2][1] = 0b00000000; cube[3][1] = 0b00000000;
cube[4][1] = 0b00000000; cube[5][1] = 0b00000000; cube[6][1] = 0b00000000; cube[7][1] = 0b00000000;
  cube[0][2] = 0b00000000; cube[1][2] = 0b00000000; cube[2][2] = 0b00000000; cube[3][2] = 0b00000000;
cube[4][2] = 0b00000000; cube[5][2] = 0b00000000; cube[6][2] = 0b00000000; cube[7][2] = 0b00000000;
  cube[0][3] = 0b00000000; cube[1][3] = 0b00000000; cube[2][3] = 0b00000000; cube[3][3] = 0b00011000;
cube[4][3] = 0b00011000; cube[5][3] = 0b00000000; cube[6][3] = 0b00000000; cube[7][3] = 0b00000000;
  cube[0][4] = 0b00000000; cube[1][4] = 0b00000000; cube[2][4] = 0b00000000; cube[3][4] = 0b00011000;
cube[4][4] = 0b00011000; cube[5][4] = 0b00000000; cube[6][4] = 0b00000000; cube[7][4] = 0b00000000;
  cube[0][5] = 0b00000000; cube[1][5] = 0b00000000; cube[2][5] = 0b00000000; cube[3][5] = 0b00000000;
cube[4][5] = 0b00000000; cube[5][5] = 0b00000000; cube[6][5] = 0b00000000; cube[7][5] = 0b00000000;
  cube[0][6] = 0b00000000; cube[1][6] = 0b00000000; cube[2][6] = 0b00000000; cube[3][6] = 0b00000000;
cube[4][6] = 0b00000000; cube[5][6] = 0b00000000; cube[6][6] = 0b00000000; cube[7][6] = 0b00000000;
  cube[0][7] = 0b00000000; cube[1][7] = 0b00000000; cube[2][7] = 0b00000000; cube[3][7] = 0b00000000;
cube[4][7] = 0b00000000; cube[5][7] = 0b00000000; cube[6][7] = 0b00000000; cube[7][7] = 0b00000000;
}
void draw(){
  // Repeat p many times before moving on to next frame
  // change what p is less than; more for slower annimation, less for faster
for(int p = 0; p < 10; p++){
   // Reset to first layer
layerIndex = 0;
for(int q = 0; q < 8; q++){
    // Reset to row 1
rowIndex = 0;
    // Make shift registers accept new data
digitalWrite(latchPin, LOW);
    // Shift out the layer data first
shiftOut(dataPin,clockPin,MSBFIRST,0b00000001<<layerIndex);
for(int t = 0; t < 8; t++){
      // Shift out the rows of data
shiftOut(dataPin,clockPin,MSBFIRST,cube[layerIndex][rowIndex]);
      // Move onto the next row then repeat
sound();
rowIndex++;
    }
    // Move onto the next layer and repeat
layerIndex++;
digitalWrite(latchPin, HIGH);
   }
  }
}
void setup()
{
pinMode(sp,OUTPUT); // This code runs only once. It assigns pin 1as output, and this is where we'll connect out
speaker.
randomSeed(analogRead(A0)); //This creates a random 'seed'. An initial point for the randomization to start.
//The value of this seed is from Analog input 0, which is empty. Hence it generates a random noise value.
  // Setup the pin modes
```

```
pinMode(dataPin,     OUTPUT);
pinMode(clockPin,    OUTPUT);
pinMode(latchPin,    OUTPUT);
pinMode(masterClear, OUTPUT);
  // Clear the shift registers
digitalWrite(masterClear, LOW);
delay(10);
digitalWrite(masterClear, HIGH);
}
void sound()
{
int notes[] = {c,d,e,g,a,c2,random(33,102),random(200,4978)}; //Creates an array of all notes used; A sort of dictionary which is indexed by numbers 0 to 7
  //I added 2 random notes here to add some spice to the tonality. The notes I chose belong to a pentatonic scale.
inti = random(0,8); //Generates a random value between 0 & 8, the maximum and minimum indexes of the notes
delay(70); //Creates a delay of 150 ms between each note, so it can be heard distinctly.
tone (sp,notes[i],80); //And finally, the tone function generates a tone sent to pin 'sp',of a random frequency from index i.
//The length of each note chosen is 80.
}
void loop()
{ //This is the main code, and it runs repeatedly.
  //Expanding Box Animation
  // Repeat 10 times
for(int k = 0;k < 10;k++){
box();
draw();
box2();
draw();
box3();
draw();
box4();
draw();
box3();
draw();
box2();
draw();
  }
}
```