ABSTRACT

- Event Name: Electromania
- Team name: Insignia
- Tech ID: 12758, 12759, 18600
- College Name: Indian Institute of Technology, Roorkee
- Team Leader's Name: Aditya Dhan Raj Singh
- Email ID of team leader: adityadrs@gmail.com
- Names of all the team members: Piyush Singhal, Ayush Goyal

Details

Idea

We have 8x8x8 single color led cube. We are planning to make a 3D Tetris game on it. Our blocks are going to be 2x4x4 type. We have already implemented 3D snake game on it.

Architecture of Cube

We have used 9 shift registers(74hc595) to controls each LED, and a microcontroller (Arduino Uno or Arduino Mega) will control which LED will light up.

The data sent by the microcontroller is of 72 bits. The first 62 bits correspond to each pillar, and the remaining 8 bits correspond to the level. By shifting data continuously, we can control entire 64*8=512 LEDs.

Other than the bit stream clock, latch and blank signals are also sent by the microcontroller. The latch enables the output when pulled high.

Rules

1. A level is cleared when all of the LEDs in a particular layer are turned on.
2. Control will be given to user to direct a block in any of the four directions.

NOTE: Further additions can be made as the project progresses.

Type of blocks used:

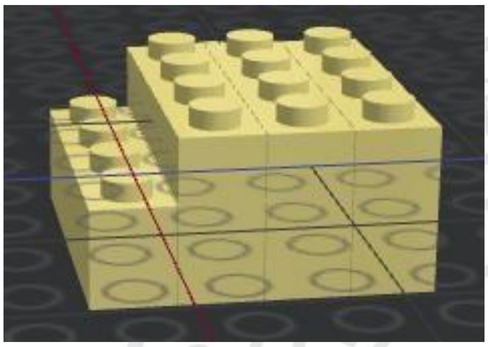
Figure 1: Block type 1

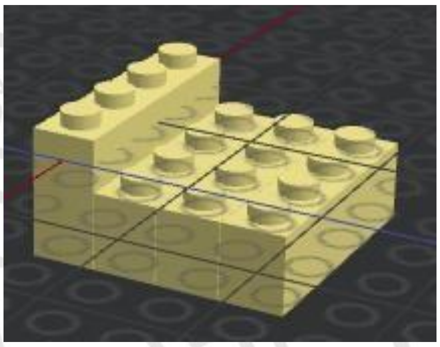
Figure 2: Block type 2

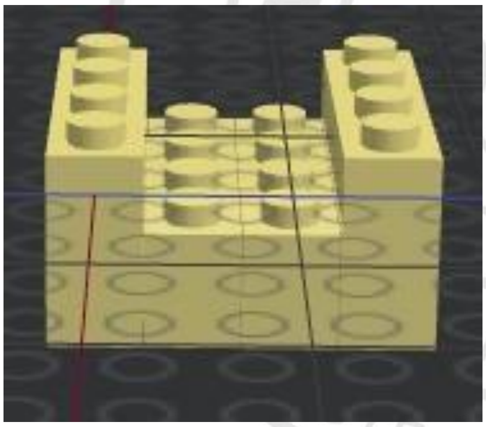
Figure 3: Block type 3

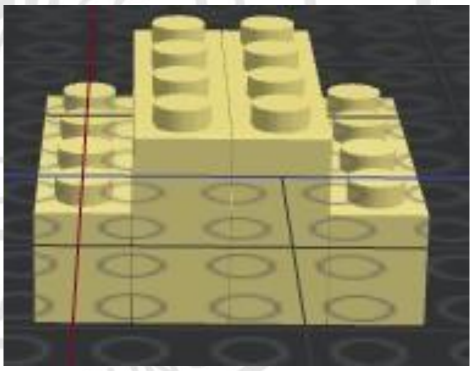
Figure 4: Block type 1

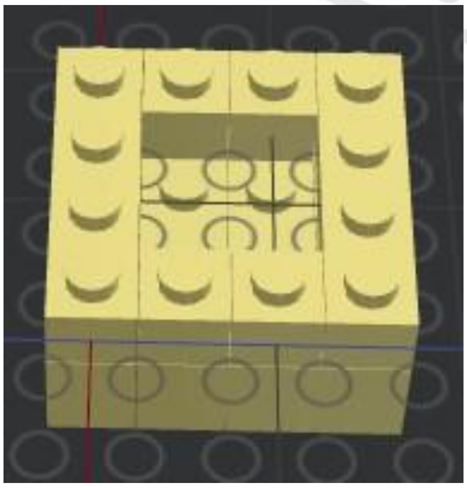
Figure 5: Block type 3

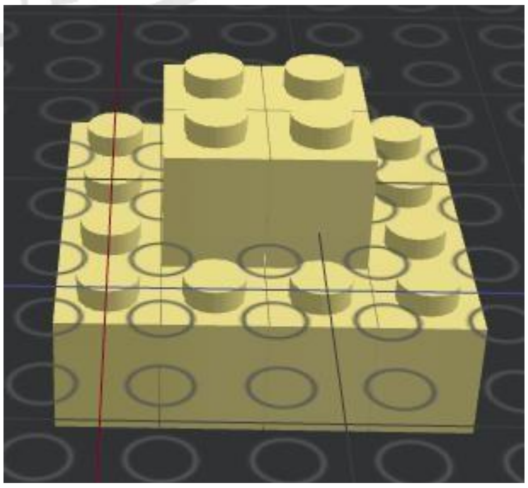
Figure 6: Block type 3

There are 512 LEDs in the cube, so when we write to level 2, column 5, row 4, that needs to be translated into a number from 0 to 511

The first level LEDs are first in the sequence, then 2nd level, then third, and so on

the (level*64) is what indexes the level's starting place, so level 0 are LEDs 0-63, level 1 are LEDs 64-127, and so on

The column counts left to right 0-7 and the row is back to front 0-7

This means that if you had level 0, row 0, the bottom back row would count from 0-7,

So if you looked down on the cube, and only looked at the bottom level

00	01	02	03	04	05	06	07
08	09	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Then, if you incremented the level, the top right of the grid above would start at 64

The reason for doing this, is so you don't have to memorize a number for each LED, allowing you to use level, row, column

Now, what about the divide by 8 in there?

Well, we have 8 bits per byte, and we have 64 bytes in memory for all 512 bits needed for each LED, so

We divide the number we just found by 8, and take the integ7er of it, so we know which byte, that bit is located

Confused? That’s ok, let's take an example, if we wanted to write to the LED to the last LED in the cube, we would write a 7, 7, 7 giving (7*64)+(7*8)=7 = 511, which is right, but now let's divide it by 8, 511/8 = 63.875, and take the int of it so, we get 63, this is the last byte in the array, which is right since this is the last LED

This next variable is the same thing as before, but here we don't divide by 8, so we get the LED number 0-511

*int wholebyte=(level*64)+(row*8)+column;*

This is 4 bit color resolution, so each color contains x4 64 byte arrays, explanation below:

```
  bitWrite(blue0[whichbyte], wholebyte-(8*whichbyte), bitRead(blue, 0));
  bitWrite(blue1[whichbyte], wholebyte-(8*whichbyte), bitRead(blue, 1));
  bitWrite(blue2[whichbyte], wholebyte-(8*whichbyte), bitRead(blue, 2));
  bitWrite(blue3[whichbyte], wholebyte-(8*whichbyte), bitRead(blue, 3));
```

Are you now more confused? You shouldn't be! It's starting to make sense now. Notice how each line is a *bitWrite*, which is, *bitWrite(the byte you want to write to, the bit of the byte to write, and the 0 or 1 you want to write)*. This means that the '*whichbyte*' is the byte from 0-63 in which the bit corresponding to the LED from 0-511. Is making sense now why we did that? Taking a value from 0-511 and converting it to a value from 0-63, since each LED represents a bit in an array of 64 bytes. Then next line is which bit *'wholebyte-(8*whichbyte)*'

This is simply taking the LED's value of 0-511 and subtracting it from the BYTE its bit was located in times 8. Think about it, byte 63 will contain LEDs from 504 to 511, so if you took 505-(8*63), you get a 1, meaning that, LED number 505 is located in bit 1 of byte 63 in the array

Is that it? No, you still have to do the *bitRead* of the brightness 0-15 you are trying to write, if you wrote a 15 to RED, all 4 arrays for that LED would have a 1 for that bit, meaning it will be on 100%. This is why the four arrays read 0-4 of the value entered in for RED, GREEN, and BLUE. Hopefully this all makes some sense?

Methodology

The making of the Circuit

First of all circuit layout for the two set of PCBs(printed circuit boards) were made from the schematic diagrams. One of the set would act as a base for the cube and will also contain the anode power supply circuit and would work in conjunction with the second set of boards that would control the 64 cathodes and therefor enable multiplexing.

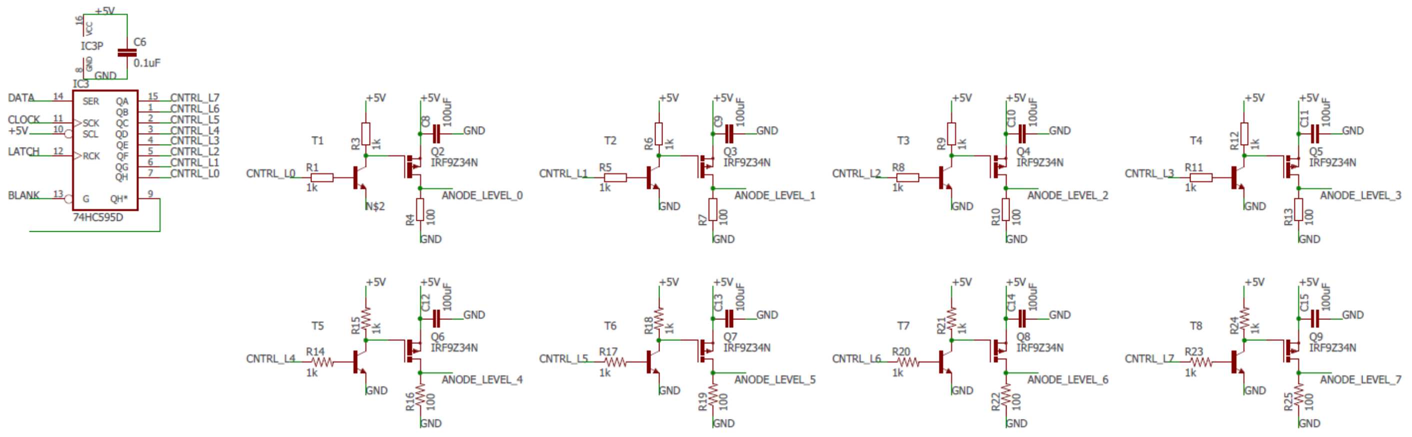

Figure 1: Anode control part of the schematics

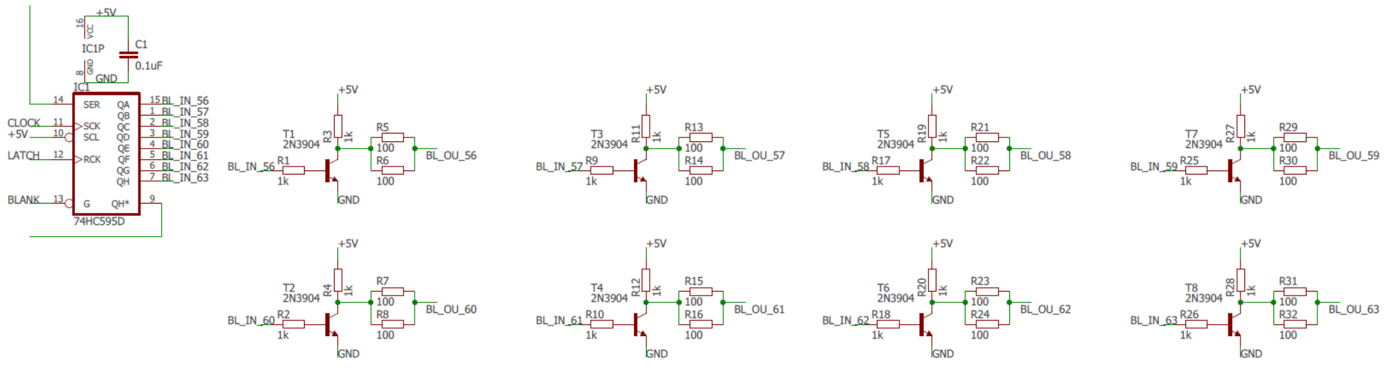

Figure 2: Cathode control part of the schematics

The complete schematic diagram is given below in figure 3.

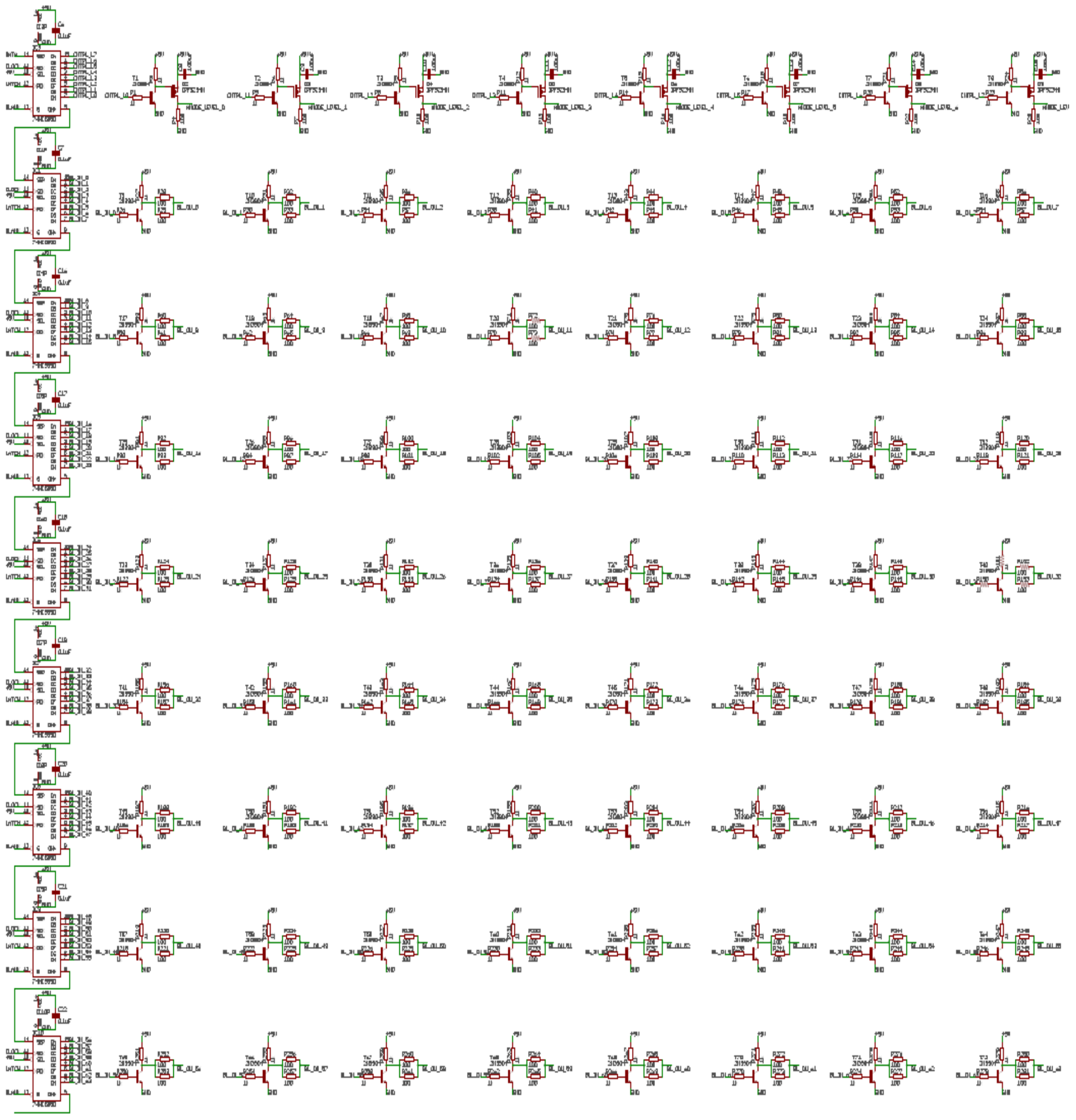

Figure 3: Complete schematics

The making of the 3D LED matrix

Anodes of all the LEDs were bent at a 90^{o} angle to fit the stencil made by drilling holes in a piece of wood. The stencil contained 8 holes so as to hold 8 LEDs, and thereby allowing to make linear array of 8 LEDs by soldering the bent anode on one LED to the anode of the next LED. 64 such linear arrays were made, each having 8 LEDs.

Now 8 such arrays were arranged in a stencil made with the help of paper pins. These 8 arrays were solder by connecting their cathodes using tin wire in a vertical sense so as to make a slice, NOT A LAYER because if the linear arrays are soldered in horizontal sense, they would have 64 cathodes that would have to be connected to the next layer where as in these slices there are only eight connections that need to be made. Eight such slices of 8x8 LEDs were made and there after they were stacked upon each other using spacers that can be removed and soldered the anodes. A tin wire was used to provide structural rigidity. This way, a cube of 8x8x8 LEDs was formed to get a 3D LED matrix.

After this, the LED matrix was placed and soldered on a PCB that was fabricated with lay out diagram in figure 4. The picture of the completed LED matrix after fixing on the PCB is given in figure 6 below.

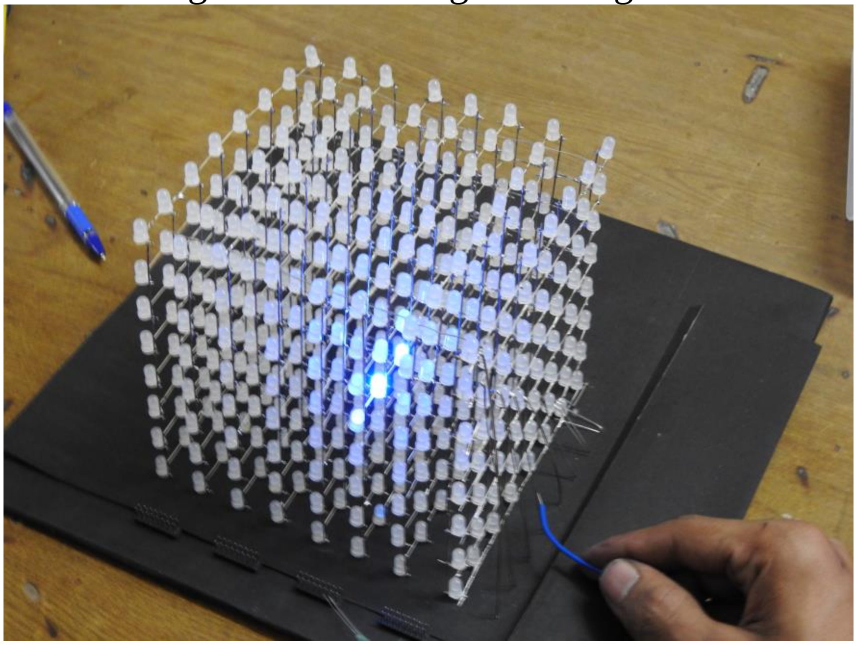

Figure 6: Completed LED matrix

The second set of PCBs connect to the PCB that holds the LED matrix via four 2x8 connectors that are place at the bottom and at the middle of the respective boards.