TEAM-18 PROJECT REPORT

VIBRANT LED CUBE

TEAM MEMBERS

ASHISH THAKUR (21NA30004)
BHAVESH MUKHEJA (21PH10008)
MANNAN THAKUR (21MT10030)
SAYAN MANDAL (21MT10049)

ACKNOWLEDGMENT

We take this opportunity to express our profound gratitude and deep regards to Prof. Sreeraj Puravankara, Prof. Koraak Sarkar, and the TAs for their exemplary guidance, monitoring, and constant encouragement throughout the course of this project. The blessing, help, and guidance given by them from time to time shall carry us a long way in the journey of life on which we are about to embark. Thank You for giving us the opportunity to learn and do something new and interesting, and learn about teamwork.

CONTENT

INTRODUCTION

A LED cube is like a LED screen, but it is special because it has a third dimension, making it 3D. Think of it as many transparent low-resolution displays. In normal displays, it is customary to try to stack the pixels as close as possible in order to make it look better, but in a cube, one must be able to see through it, and more spacing between the voxels is needed. The spacing is a trade-off between how easily the layers behind it are seen and voxel fidelity. Since it is a lot more work making a LED cube than a LED display, they are usually low resolution. The LED cube used in our project has a dimension of 8×8×8 i.e. a total of 512 cubes.

APPLICATIONS:

•The LED cube can be used for decorative purposes. Its aesthetic beauty is pleasing to the eye.

•The LED cube can be used for parties and functions. The UV distance sensor enables giving different effects at different distances, so it can be used as a party game.

•Furthermore, with advanced resources it can also be used as a display.

WORKING

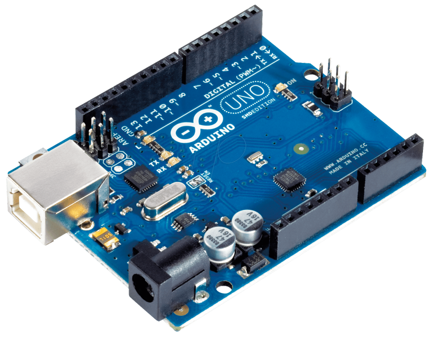
Using Arduino IDE and TINKERCAD, we have coded special effects for the cube. When the Red LED is switched on, the UV distance sensor is inactive and effects are generated at random. When the Green LED is switched on, the UV distance sensor becomes active and at different distances, different effects are generated as per the coding. The UV distance sensor works by sending out a sound wave at a frequency above human hearing capability. The effects coded on connecting the Arduino Uno to the hardware are reflected on the cube.

COMPONENTS USED IN THE PROJECT

- 1 Arduino Uno
- 1 UV Distance Sensor
- 1 Red LED
- 1 Green LED
- 2 10KΩ Resistors
- 3 Pushbuttons
- 9 74HC595 Shift Registers
- 16 2222A Transistors
- 512 Blue LEDs
- 514 220Ω Resistors
- Jumper Wires
- Prototype Board
- Soldering Kit
- Tinned Copper Wire

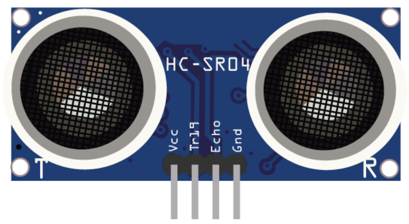
BRIEF DESCRIPTION OF THE COMPONENTS

Arduino Uno



The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino. cc. The board is equipped with sets of digital and analog input/output pins that may be interfaced to various expansion boards and other circuits

UV Distance Sensor



As the name suggests, UV Distance Sensor is a sensor that uses Ultrasound to calculate the distance between the sensor and obstacle/object.
The working principle of this module is simple. It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated. Ultrasonic sensors are a great solution for the detection of clear objects.

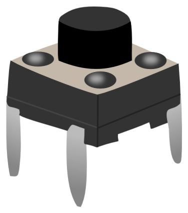
It is also ineffective to light, dust, color, smoke, etc, and can measure precisely the distance ranging from 2cm to 400cm. Hence, the UV Distance sensor was the most accurate choice for the project.
Our project uses a UV Distance sensor to show a particular effect under a particular range of distance.

- **LEDs**

A light-emitting diode or LED is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.
We have used 3 different colored LEDs in the project. Blue, Green, and Red. The Blue LEDs construct the LED Cube whereas the Red and Green LEDs are used to signify the ON and OFF of the UV Distance Sensor.

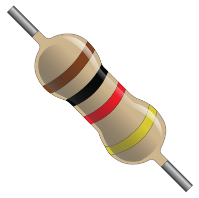
- **Pushbuttons**

A push-button or simply button is a simple switch mechanism to control some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed.
In the project, we have used 3 pushbuttons. One is to change the effects in the Cube. Second, to activate the UV Distance Sensor. Third, deactivate the UV Distance Sensor.



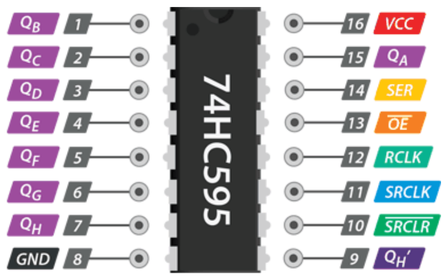
- **Resistors**

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, divide voltages, bias active elements, and terminate transmission lines, among other uses.



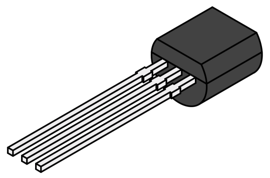
❖ 74HC595 Shift Registers

The Shift Registers are the most important components of our project. They are used to control a lot of LEDs while using only 3-4 I/O pins of the microcontroller. And this was the sole purpose why we used 16-pinned 74HC595 Shift Registers. More discussion about the same, later in the FUNCTIONING OF THE CIRCUIT section of this report.



❖ 2222A Transistors

The 2222A is a common NPN bipolar junction transistor (BJT) used for general-purpose low-power amplifying or switching applications. It is designed for low to medium current, low power, and medium voltage, and can operate at moderately high speeds.

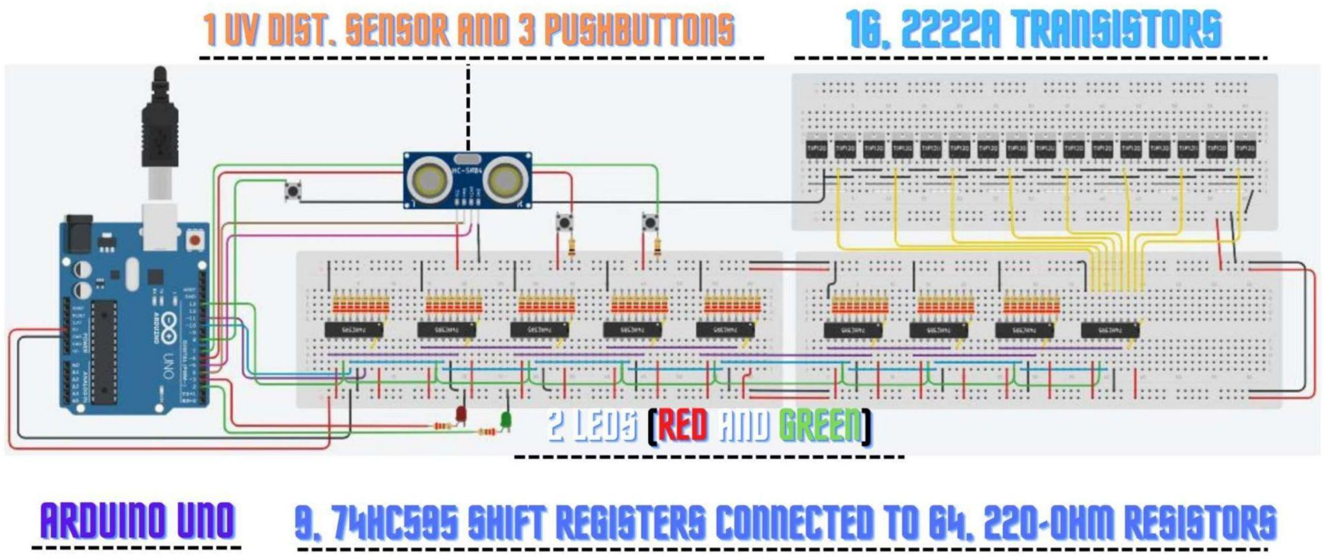


❖ Jumper Wires

A jump wire is an electrical wire, or group of them in a cable, with a connector or pins at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering. Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

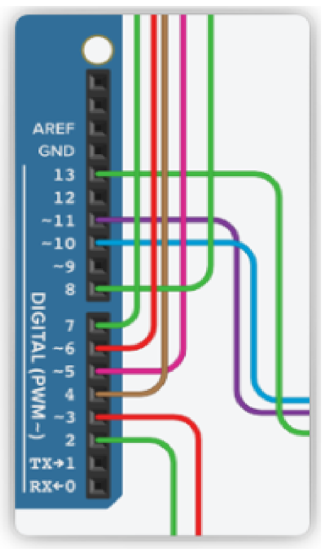
HOW DOES THE PROJECT FUNCTIONS

CIRCUIT DIAGRAM


1 UV DIST. SENSOR AND 3 PUSHBUTTONS
16. 2222A TRANSISTORS
2 LEDS (RED AND GREEN)
ARDUINO UNO
9. 74HC595 SHIFT REGISTERS CONNECTED TO 64. 220-OHM RESISTORS

74HC595 Shift Registers are the most important components of this project. The 74HC595 Shift Register takes in input signals through 3 pins which are RCLK Pin, SER Pin, and SRCLK Pin which is connected to the Arduino UNO as shown below. The 64 220-ohm resistors are connected to the 8 Shift Registers that will be connected to the LED Cube.The 9th Shift Register's 8 output pins are connected to the combination of 16 2222A Transistors. The 8x8x8 LED Cube will be connected through the end terminals of the resistors and transistors(collector terminals). Once all the connections are made, we can control 512 Blue LEDS using code.

- 🟩 Digital Pin-2 -> Green LED
- 🟥 Digital Pin-3 -> Red LED
- 🟦 Digital Pin-4 -> Trigger Pin of UV Sensor
- 🟪 Digital Pin-5 -> Echo Pin of UV Sensor
- 🟥 Digital Pin-6 -> Red Pushbutton
- 🟩 Digital Pin-7 -> Green Pushbutton
- 🟩 Digital Pin-8 -> Emitter Pushbutton
- 🟦 Digital Pin-10 -> RCLK Pin
- 🟪 Digital Pin-11 -> SER Pin
- 🟩 Digital Pin-13 -> SRCLK Pin



What's different about our LED Cube is that you can also control its effect using UV Distance Sensor if you want to. All you need to do is to Pushbutton connected to the Green Wire and Green Led will light up, signifying that the UV Sensor is now active and you can control the effects while changing your hand's (or an object's) position.

Range (cms)	**Effect**	**Description**
400-301	Rain	An effect given to the cube such that it looks like a soothing rain.
300-201	Plan Boing	The moving of a lit-up plane across the cube.
200-101	Woop-Woop	Expanding of a small cube from inside out.
100-002	Text	As the name suggests, it is an effect that shows text using LEDS

Now if you want to return to the switching mode again you can push the Pushbutton Connected to the Red Wire and the Red Led will light up signifying that the UV Sensor is now inactive. The same effects will be reflected in the Cube but without any influence on your hand (or any object).
For a better understanding of the project, you can take a look at the Circuit Schematics, and Demo Circuit.

SOFTWARE USED

❖ **TINKERCAD**

Tinkercad is a free-of-charge, online 3D modeling program that runs in a web browser. Since it became available in 2011 it has become a popular platform for creating models for 3D printing as well as an entry-level introduction to constructive solid geometry in schools.

❖ **ARDUINO IDE**

The **Arduino Integrated Development Environment** - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

CODE

```
#include <SPI.h>

#define X_axis 0
#define Y_axis 1
#define Z_axis 2

#define POSITION_X 0
#define POSITION_Z 2
#define POSITION_Y 4

#define NEG_X 1
#define NEG_Z 3
#define NEG_Y 5

#define BTN 4

#define TOTAL 4
#define RAIN 0
#define PLANE_BOING 1
#define WOOP_WOOP 2
#define TEXT 3


#define RAIN_TIME 200
#define PLANE_BOING_TIME 300
#define WOOP_WOOP_TIME 350
#define TEXT_TIME 300


int greenButton = 0;
int redButton = 0;

const int greenLed = 2;
const int redLed = 3;
const int trigPin = 4;
const int echoPin = 5;
int duration = 0;
int distance = 0;


uint8_t characters[10][8] = {
  {0x3C, 0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x3C}, //0
  {0x10, 0x18, 0x14, 0x10, 0x10, 0x10, 0x10, 0x3C}, //1
  {0x3C, 0x42, 0x40, 0x40, 0x3C, 0x02, 0x02, 0x7E}, //2
  {0x3C, 0x40, 0x40, 0x3C, 0x40, 0x40, 0x42, 0x3C}, //3
  {0x22, 0x22, 0x22, 0x22, 0x7E, 0x20, 0x20, 0x20}, //4
  {0x7E, 0x02, 0x02, 0x3E, 0x40, 0x40, 0x42, 0x3C}, //5
  {0x3C, 0x02, 0x02, 0x3E, 0x42, 0x42, 0x42, 0x3C}, //6
  {0x3C, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40}, //7
  {0x3C, 0x42, 0x42, 0x3C, 0x42, 0x42, 0x42, 0x3C}, //8
  {0x3C, 0x42, 0x42, 0x42, 0x3C, 0x40, 0x40, 0x3C}, //9
};


uint8_t CubeData[8][8];        // LEDs Data (One Bit Per LED) => [y][z] = x //
uint8_t CurrentEffect;         // Store The Number Of Current Effect //
uint8_t Blinker;               // Counter For Blinker LED (Front LED) //
uint8_t CurrentStep;           // Output Step (Y Axis) //

uint64_t RandomSeed;

uint8_t cube[8][8];
uint8_t currentEffect;

uint16_t timer;

uint64_t randomTimer;

bool loading;

void setup() {

  pinMode(6, INPUT);
   pinMode(7, INPUT);
  pinMode(greenLed, OUTPUT);
    pinMode(redLed, OUTPUT);
      pinMode(trigPin , OUTPUT);
pinMode(echoPin , INPUT);

loading = true;
randomTimer = 0;
currentEffect = RAIN;

SPI.begin();
SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE0));

pinMode(BTN, INPUT_PULLUP);

//Serial.begin(9600);

randomSeed(analogRead(0));

}

void loop() {

  digitalWrite(greenLed, LOW);
  digitalWrite(redLed, HIGH);

  randomTimer++;

  if (digitalRead(BTN) == LOW) {
    clearCube();
    loading = true;
    timer = 0;
    currentEffect++;
    if (currentEffect == TOTAL) {
      currentEffect = 0;
    }
    randomSeed(randomTimer);
    randomTimer = 0;
    delay(500);

  }
  switch (currentEffect) {
    case RAIN: rain(); break;
    case PLANE_BOING: planeBoing(); break;
    case WOOP_WOOP: woopWoop(); break;
    case TEXT: text("0123456789", 10); break;
    default: rain();
  }

  renderCube();

  greenButton = digitalRead(2);

if (greenButton == HIGH){

  digitalWrite(redLed,LOW);
  digitalWrite(greenLed, HIGH);

  while(1){
    digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);  // in microseconds
  distance = duration/58.2;           // in cms

  if(distance <=400 && distance >300){
rain();
  }
   else if(distance <=300 && distance >200){
  planeBoing();
  }
   else if(distance <=200 && distance >100){
  woopWoop();
  }
   else if(distance <=100){
  text("0123456789", 10);
  }
```

```
    redButton = digitalRead(3);
    if(redButton == HIGH)
      break;
    }

  }

}

void renderCube() {
  for (uint8_t i = 0; i < 8; i++) {
    digitalWrite(SS, LOW);
    SPI.transfer(0x01 << i);
    for (uint8_t j = 0; j < 8; j++) {
      SPI.transfer(cube[i][j]);
    }
    digitalWrite(SS, HIGH);
  }
}

void rain() {
  if (loading) {
    clearCube();
    loading = false;
  }
  timer++;
  if (timer > RAIN_TIME) {
    timer = 0;
    shift(NEG_Y);
    uint8_t numDrops = random(0, 5);
    for (uint8_t i = 0; i < numDrops; i++) {
      setVoxel(random(0, 8), 7, random(0, 8));
    }
  }
}

uint8_t planePosition = 0;
uint8_t planeDirection = 0;
bool looped = false;
```

```
void planeBoing() {
  if (loading) {
    clearCube();
    uint8_t axis = random(0, 3);
    planePosition = random(0, 2) * 7;
    setPlane(axis, planePosition);
    if (axis == X_axis) {
      if (planePosition == 0) {
        planeDirection = POSITION_X;
      } else {
        planeDirection = NEG_X;
      }
    } else if (axis == Y_axis) {
      if (planePosition == 0) {
        planeDirection = POSITION_Y;
      } else {
        planeDirection = NEG_Y;
      }
    } else if (axis == Z_axis) {
      if (planePosition == 0) {
        planeDirection = POSITION_Z;
      } else {
        planeDirection = NEG_Z;
      }
    }
    timer = 0;
    looped = false;
    loading = false;
  }

  timer++;
  if (timer > PLANE_BOING_TIME) {
    timer = 0;
    shift(planeDirection);
    if (planeDirection % 2 == 0) {
      planePosition++;
      if (planePosition == 7) {
        if (looped) {
          loading = true;
        } else {
          planeDirection++;
          looped = true;
        }
      }
    } else {
      planePosition--;
      if (planePosition == 0) {
        if (looped) {
          loading = true;
        } else {
          planeDirection--;
          looped = true;
        }
      }
    }
  }
}

uint8_t selX = 0;
uint8_t selY = 0;
uint8_t selZ = 0;
uint8_t sendDirection = 0;
bool sending = false;



uint8_t cubeSize = 0;
bool cubeExpanding = true;

void woopWoop() {
  if (loading) {
    clearCube();
    cubeSize = 2;
    cubeExpanding = true;
    loading = false;
  }

  timer++;
  if (timer > WOOP_WOOP_TIME) {
    timer = 0;
    if (cubeExpanding) {
      cubeSize += 2;
      if (cubeSize == 8) {
        cubeExpanding = false;
      }
    } else {
      cubeSize -= 2;
      if (cubeSize == 2) {
        cubeExpanding = true;
      }
    }
    clearCube();
    drawCube(4 - cubeSize / 2, 4 - cubeSize / 2, 4 - cubeSize / 2, cubeSize);
  }
}

uint8_t xPos;
uint8_t yPos;
uint8_t zPos;


bool glowing;
uint16_t glowCount = 0;



uint8_t charCounter = 0;
uint8_t charPosition = 0;

void text(char string[], uint8_t len) {
  if (loading) {
    clearCube();
    charPosition = -1;
    charCounter = 0;
    loading = false;
  }
  timer++;
  if (timer > TEXT_TIME) {
    timer = 0;
    shift(NEG_Z);
    charPosition++;

    if (charPosition == 7) {
      charCounter++;
      if (charCounter > len - 1) {
        charCounter = 0;
      }
      charPosition = 0;
    }

    if (charPosition == 0) {
      for (uint8_t i = 0; i < 8; i++) {
        cube[i][0] = characters[string[charCounter] - '0'][i];
      }
    }
  }
}



void setVoxel(uint8_t x, uint8_t y, uint8_t z) {
  cube[7 - y][7 - z] |= (0x01 << x);
}

void clearVoxel(uint8_t x, uint8_t y, uint8_t z) {
  cube[7 - y][7 - z] ^= (0x01 << x);
}

bool getVoxel(uint8_t x, uint8_t y, uint8_t z) {
  return (cube[7 - y][7 - z] & (0x01 << x)) == (0x01 << x);
}

void setPlane(uint8_t axis, uint8_t i) {
  for (uint8_t j = 0; j < 8; j++) {
    for (uint8_t k = 0; k < 8; k++) {
      if (axis == X_axis) {
        setVoxel(i, j, k);
      } else if (axis == Y_axis) {
        setVoxel(j, i, k);
      } else if (axis == Z_axis) {
        setVoxel(j, k, i);
      }
    }
  }
}

void shift(uint8_t dir) {

  if (dir == POSITION_X) {
    for (uint8_t y = 0; y < 8; y++) {
      for (uint8_t z = 0; z < 8; z++) {
        cube[y][z] = cube[y][z] << 1;
      }
    }
  } else if (dir == NEG_X) {
    for (uint8_t y = 0; y < 8; y++) {
      for (uint8_t z = 0; z < 8; z++) {
        cube[y][z] = cube[y][z] >> 1;
      }
    }
  } else if (dir == POSITION_Y) {
    for (uint8_t y = 1; y < 8; y++) {
      for (uint8_t z = 0; z < 8; z++) {
        cube[y - 1][z] = cube[y][z];
      }
    }
    for (uint8_t i = 0; i < 8; i++) {
      cube[7][i] = 0;
    }
  } else if (dir == NEG_Y) {
    for (uint8_t y = 7; y > 0; y--) {
      for (uint8_t z = 0; z < 8; z++) {
        cube[y][z] = cube[y - 1][z];
      }
    }
    for (uint8_t i = 0; i < 8; i++) {
      cube[0][i] = 0;
    }
  } else if (dir == POSITION_Z) {
    for (uint8_t y = 0; y < 8; y++) {
      for (uint8_t z = 1; z < 8; z++) {
        cube[y][z - 1] = cube[y][z];
      }
    }
    for (uint8_t i = 0; i < 8; i++) {
      cube[i][7] = 0;
    }
  } else if (dir == NEG_Z) {
    for (uint8_t y = 0; y < 8; y++) {
      for (uint8_t z = 7; z > 0; z--) {
        cube[y][z] = cube[y][z - 1];
      }
    }
    for (uint8_t i = 0; i < 8; i++) {
      cube[i][0] = 0;
    }
  }
}

void drawCube(uint8_t x, uint8_t y, uint8_t z, uint8_t s) {
  for (uint8_t i = 0; i < s; i++) {
    setVoxel(x, y + i, z);
    setVoxel(x + i, y, z);
    setVoxel(x, y, z + i);
    setVoxel(x + s - 1, y + i, z + s - 1);
    setVoxel(x + i, y + s - 1, z + s - 1);
    setVoxel(x + s - 1, y + s - 1, z + i);
    setVoxel(x + s - 1, y + i, z);
    setVoxel(x, y + i, z + s - 1);
    setVoxel(x + i, y + s - 1, z);
    setVoxel(x + i, y, z + s - 1);
    setVoxel(x + s - 1, y, z + i);
    setVoxel(x, y + s - 1, z + i);
  }
}

void lightCube() {
  for (uint8_t i = 0; i < 8; i++) {
    for (uint8_t j = 0; j < 8; j++) {
      cube[i][j] = 0xFF;
    }
  }
}

void clearCube() {
  for (uint8_t i = 0; i < 8; i++) {
    for (uint8_t j = 0; j < 8; j++) {
      cube[i][j] = 0;
    }
  }
}.
```

ERRORS/PROBLEMS FACED

- The size of the cube i.e. 8×8×8 512 LEDs caused a slight issue. With only one team member taking care of the hardware, the tasks of soldering, assembling, and testing all the LEDs became quite a tedious task.

- We had initially planned to code the cube on the beats of a song. However, we were unable to do so because there was an issue in distinguishing respective beats. So it was being tough to correspond a particular beat to a particular effect.

SOLUTIONS

- Despite the size being an issue, in the end, all the connections and testing were done successfully by our team members. The size of the cube eventually adds to its aesthetic beauty.

- Though we weren't able to code effects on beats, by using the UV distance sensor we more than made up for our above hindrance. The sensor not only codes different effects at different corresponding distances but also adds the utility of the cube as a party game.

WORK DIVISION:

- ***Ashish Thakur*:**
  - Assembled the hardware components
  - Completed testing and running of the LED cube
  - Video Editing

- ***Bhavesh Mukheja*:**
  - Made PPT for all the weeks
  - Designed the Circuits in TINKERCAD and prepared Circuit Schematics.
  - Integrated the UV distance sensor code with the effects code
  - Contributed to Final Project Report

- ***Mannan Thakur*:**
  - Wrote the code for the effects and UV distance sensor for the LED cube
  - Made PPT for week 4
  - Contributed to Final Project Report

- ***Sayan Mandal*:**
  - Writing the code for the effects and UV distance sensor for the LED cube
  - Making of PPT for week 1
  - Contributed to Final Project Report

FUTURE WORK/ALTERNATIVES

Future work we are considering for this project is a larger resolution cube. Currently, the dimension of the cube is 4x4x4. The amount of animation and light patterns is very limited due to the dimension limitation. By making a larger resolution cube, the display will have a higher resolution, and thus be possible for displaying ASCII characters and more sophisticated animations. Next, we are also considering adding more sensors to the LED cube, to make the animations more interactive. For example, if the LED cube consists of multiple IR range finders, then the animation can respond to the user's motion in more than one direction. Lastly, animation design software should be developed to come with the cube. Then this allows users to design animations without having to write any code.

CONCLUSION

A 3D LED cube can be used for displaying various patterns or structures in a 3-dimensional plane with help of LED arrays and microcontrollers. The resolution of the image being displayed can be improved by increasing the voxels i.e the number of LEDs used is to be increased. A smaller cube also can be built using a 4x4x4 configuration for simulation purposes. For this project, we have successfully finished and fulfilled all the requirements and verification proposed in the design review. The construction of the cube allows good viewing of all the LEDs in the cube making for a great visual presentation of the cube. On the software side, we have successfully designed three interactive animations. The successful completion of every task demonstrated the potential of mechatronic systems and a positive group dynamic.

REFERENCES

- Google
- YouTube

SOME IMPORTANT LINKS

- Link for Circuit Schematics, Demo Circuit, and Demonstration Video.
- Link for the YouTube Video.