NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE

NANYANG TECHNOLOGICAL UNIVERSITY

IM3080 Design and Innovation Project

AY2022/2023

Title: Cloud Tubes

Github: https://github.com/regyap/IM3080-InteractiveDisplay.git

Submitted by: Yap Hui Xin Regine

Supervisor: Mr Andy Khong

Date: 15 Nov 2022

Wang Yaoxuan (U2022429L)	Wu Zhao Yi (U2122617K)
Lim Xuan (U2020846E)	Ryan Gan (U2023880E)
Hou Bo (U2022619G)	Lim Bing Hong (U2022305K)
Loh Yi Hong (U2023022A)	Chow Zhi Xin (U2122004B)
Yap Hui Xin Regine (U2122314B)	

Table of Content

1. Project Background & Motivation

Our group was inspired by Garden of the Bay's Cloud Forest,and was interested to do an Interactive Project. Hence, we chose tubes to try to replicate the colors we see in Cloud Forest.

2. Objective

To create a light interactive installation with different features controlled by webapp

To learn the basics and principles of Arduino and Raspberry Pi and apply them to interactive devices.

Project Timeline

Our Project is loosely planned base on the tasks we have to accomplish by the end of the week, hence Trello and our common messaging platform. Telegram, were utlised to update and track our tasks.

The trello is utlised to keep track of tasks done by each team from week 4 to week 8. Our team shifted to Telegram for task assignment after week 8 as Trello was updated lesser.

Here is a summarised timeline of our Project according to our Trello tasks and Telegram messages.

	Accomplishments	Tasks
Week 1	-Project Selection	-Learn how to use Arduino -Learn how to use Raspberry Pi -Learn React **Software Team** -Design WebApp
Week 2 (Emotion Detection)	-Made Wireframe for Emotion Detector App https://www.figma.com/file/VuLFhY3DiKH40ClJPWHIUF/IM3080-Wireframe?node-id=0%3A1 -Visited Common Store to get Raspberry Pi &	**Hardware Team** -Learn how to wire -Design Structure

	other components	**Arduino Team** -Learn how to use Arduino -Learn how to use Raspberry Pi **Software Team** -Design WebApp -Learn React
Week 3 (Emotion Detection)	-Prototype of hardware display -Think of ideas for the Project -Coded Emotion Detection Model <u>Presentation #1</u> Feedback: Change idea	-Think of ideas for the Project -Modes -Games -Acquire materials **Hardware Team** -Design new structure
Week 4 (Whack A Mole)	-Meet with supervisor to discuss about idea -identify components needed -change project idea <u>Ideas</u> Features - Multiplayer red light vs green light - Scoresheet To Bump Grade Up - Mobile App Need Gameplay Design - Synchronise with Music/Beats	**Hardware Team** -Learn how to wire -Design Structure -Test sensors and LEDs on small scale **Arduino Team** -Learn how to use Arduino -Learn how to use Raspberry Pi **Software Team** -Design WebApp -Learn React -Learn Python
Week 5 (Lightshow with sensors)	-simplified and decided on different modes, design and reassigned teams	**Hardware Team** -test on small scale -get light working(re-wire) -note down strip layout **Arduino Team** -code simple codes for ripple effect(sensors+light) -code 1 minute of music with beats (2 sets of music) **Software Team**

		-Build up basic webapp with only 2 buttons
Week 6 (Lightshow with sensors)	-Coded Queue system for webapp	**Hardware Team** -soldering board for 12v and gnd connections and sensor 5v from arduino -Think of Hardware Design **Arduino Team** -work with 5v led strips to test codes **Software Team** -Complete design for webapp
Week 7	-Completed Webapp -Completed testing for 5v led for ripple effect(3 LED strips) -Completed testing for 5v led for music with beatst(3 LED strips) -Completed testing with 5V led strips with sensors -Connecting and wiring power sources <u>Presentation #2</u> Feedback:Start with wiring, work with 12V, find external power supply from Prof Chua	**Hardware Team** -start wiring tubes -work with 12V LED instead of 5V -Acquire external power supply -work on improving structure **Arduino Team** -Code Music with beats longer version -Code ripple effect with Rainbow effect -Help Hardware Team **Software Team** -Add more css and buttons -Help Hardware team
Recess	-Acquire external power supply -start wiring tubes -work with 12V LED instead of 5V -Code ripple effect -painted hardware black - added wheels	**Hardware Team** -Coil wires to make it neater **Arduino Team** -Code Music with beats longer version -Code ripple effect -Help Hardware Team **Software Team** -Add more css and buttons -Help Hardware team
Week 8	-Coil wires to make it neater(partially) -Code Music with beats longer version -Code ripple effect -Add more css and buttons	**Hardware Team** -Test and light 25 tubes -Recoil wires **Arduino Team**

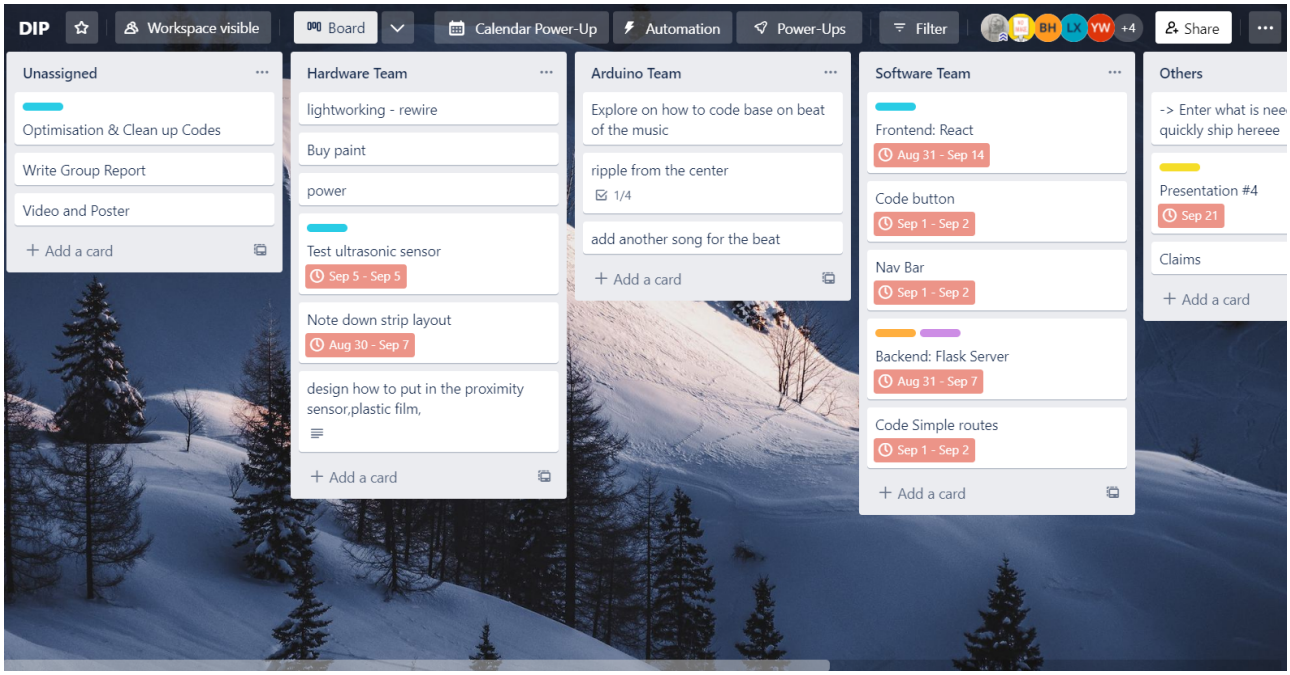
	<u>Presentation #3</u> Feedback: Hardware set up all 25 tubes, have all members help hardware. Test one effect first.	-Code Music with beats better version without too much hardcode. - Change ease in and ease out of LED -Code ripple effect with Rainbow effect -Help Hardware Team **Software Team** -Add more css -Code out functionalities like RGB bar -Help Hardware Team
Week 9	-Code Music with beats longer version -Code out functionalities like RGB bar -Add more css - Change ease in and ease out of LED -Recoil wires	**Hardware Team** -Rewire and split board into 2 sides as wire not long enough **Arduino Team** -Improve Music with beats -Code ripple effect with Rainbow effect -Help Hardware Team **Software Team** -Make webapp mobile friendly -Help Hardware Team
Week 10	-Improve Music with beats -Code ripple effect with Rainbow effect(only ripple)	**Hardware Team** -data pin not secure and solder pins **Arduino Team** -Try 1 more song for music with beats -Code ripple effect on larger scale -Help Hardware Team **Software Team** -Make webapp mobile friendly -Help Hardware Team
Week 11	-data pin not secure and solder pins -Try 1 more song for music with beats <u>Presentation #4</u> NIL.	**Hardware Team** -data pin not secure and solder pins **Arduino Team** -Code music with beats on larger scale -Code ripple effect on larger

		scale **Software Team** -Make webapp mobile friendly -Help Hardware team
Week 12	-Finish Hardware	**Group Documentations and Slides** **Hardware Team** -maintainance while arduino tests -add clouds & decorations **Arduino Team** -Test music with beats -Test ripple effect **Software Team** -look into running react and python on raspberry pi on reboot
Week 13		**Poster & Video** **Hardware Team** -maintainance while arduino tests -fix 6 sensors -add couds **Arduino Team** -Test music with beats -Test ripple effect -story mode **Software Team** -Add more modes into webapp+webpages - Change new webpages to mobile friendly -look into running react and python on raspberry pi on reboot
Week 14	-fix 6 sensors -Test music with beats -Test ripple effect -story mode -add clouds -Add more modes into webapp+webpages - Change new webpages to mobile friendly -look into running react and python on raspberry	

	pi on reboot **Poster & Video** **Group Documentations and Slides**	

Legend: November, October, September, August

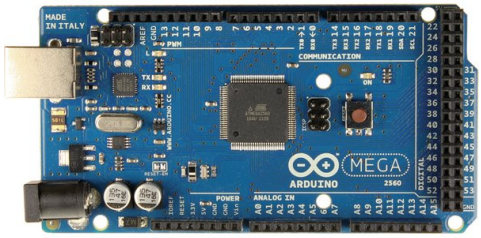
Hardware Team(3): Jasper, Ryan, Hou Bo
Arduino Team(4): Yao Xuan, Lim Xuan, Zhao Yi, Zhi Xin
Software Team(2): Bing, Regine



Picture of Trello Board- https://trello.com/b/zDXYRnwl/dip

3. Review of Literature/Technology -summary of what we found out

3.1. Hardware

Product	Description
Arduino Mega  https://store.arduino.cc/collections/boards/products/arduino-mega-2560-rev3	**a microcontroller board based on the ATmega2560**. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
Ultrasonic sensor  https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6	Sensor that can measure distance. Able to sense if user waves hand near it.
RGB LED strips	A light-emitting diode is a semiconductor device that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light is determined by the energy required for electrons to cross the band gap of the semiconductor

3.2. Arduino Code

Arduino IDE 1.8.19	The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.
Adafruit NeoPixel Library	Class that stores state and functions for interacting with Adafruit NeoPixels and compatible devices.
Arduino_JSON Library	Process JSON in your Arduino sketches.

3.3. Software

Libraries & Versions Used

React	**Python**
"antd": "^4.23.1", "axios": "^0.27.2", "bootstrap": "^5.2.1", "os": "^0.1.2", "react": "^18.2.0", "react-bootstrap": "^2.5.0", "react-dom": "^18.2.0", "react-router-dom": "^6.4.0", "react-scripts": "5.0.1", "socket.io-client": "^4.5.2", "web-vitals": "^2.1.4"	flask_socketio==5.3.1 flask_cors==3.0.10 flask_sqlalchemy==2.5.1 pyserial==3.5 gevent-websocket==0.10.1

4. Design and Implementation

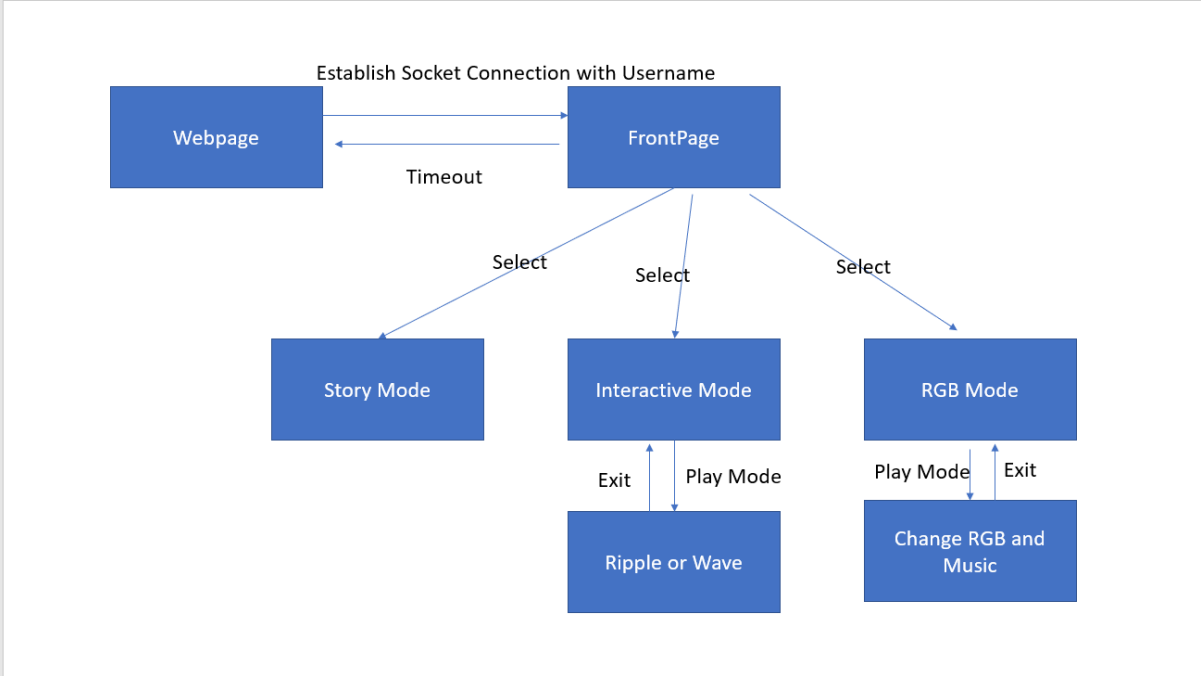
4.1. Design Consideration / Choice of components - Jasper

- Engagement: What can we do for an interactive art installation
  - Using sensors to allow user to interact with the LED lights
  - Selected Ultrasonic sensors as the point of contact with the user

- Engagement: How to allow multiple users to interact with the art installation
  - Installed 6 ultrasonic sensors at different side of the installation to trigger either a ripple or wave effect

- Safety: Wiring and electrical components
  - Installed ‘T-shape’ wire junctions to arrange and organize wires for safety hazard.

4.2. Final Design (with block diagrams) (For software, use the proper Software Engineering Diagram, such as Use case diagram, Sequence diagram, etc.)

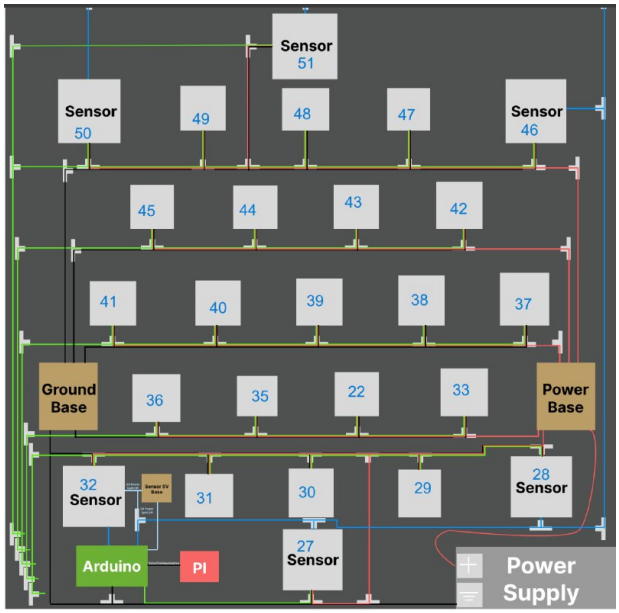


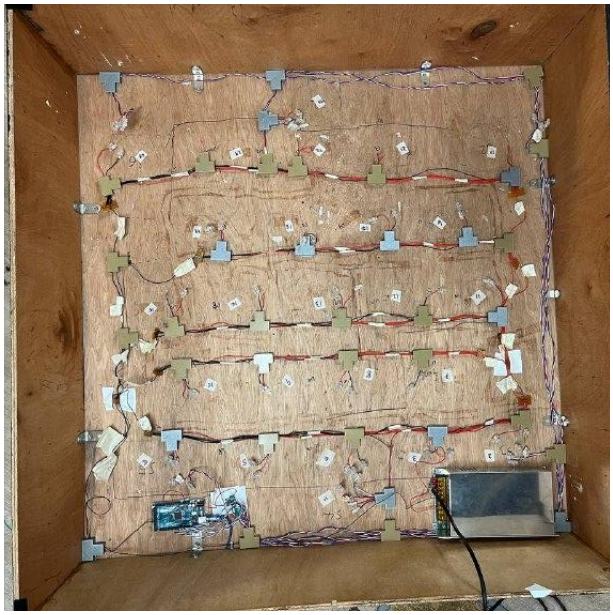
4.2.1. Software

4.3. Implementation (with photos)

Final product

Hardware





Component	Quantity	Purpose
Power Supply Unit (PSU)	1	Supply all LED strips with power
Raspberry Pi 4	1	Host the frontend and backend server. Provides data to Arduino Mega through serial communication
Arduino Mega	1	Power and takes input from the ultrasonic sensors and send signals to the LED strips
Ultrasonic Sensor	6	To measure the distance of nearby objects to trigger the LED strips based on proximity
12V LED Strips (Various Lengths)	25	To display light patterns
3D Printed Junctions	50	Wire management
Perfboard	3	For ground, 12V power and 5V power
Transparent Tubes (Various	25	To house LED strips and wiring

Lengths)		
1000 Microfarad Capacitors	25	Buffer sudden changes in current drawn by strip
475 Ohm Resistor	25	To take power spikes into account for LED data pin

Explanation

The circuitry is based on modularisation where each tube is considered as a separate component. This makes the tubes easy to remove for maintenance or future development. Assembly is also made easier using this technique.

Wires from each row of tubes are soldered to a small 2x2cm Perfboard junction which is connected to the main junction which connects all the Perfboards from each row.

All data pins are connected to a header pin strip to ensure it stays secure to the arduino.

All modular tubes are connected to the circuitry using Screw Terminal Wire Connectors to ensure it is secure.

Capacitors were connected to each LED tube to buffer sudden changes in the current drawn by the strip. As we have a large power source and a lot of pixels, resistors were also placed on each LED data pin to take power spikes into account.

Power supply was integrated into the circuit by connecting the live, neutral and ground wire from a wall socket plug to it. Followed by connecting the ground of the power supply to the arduino and the power to a Perfboard to connect to all the 12V LEDs.

Communication between Frontend and Backend

The communication between the Client (implemented using React) and Server (implemented using Flask) is done through WebSocket. React is connected to the backend using the following code.

```
#import io from "socket.io-client";
import { SERVER_IP } from "../Constant";


const socketOBJ = io(SERVER_IP, { transports: ["websocket"] });
export default socketOBJ;
```

Websocket is implemented through SocketIO library which is inspired from the Node.js EventEmitter. Either the client or server is able to emit events or register listeners. Below is an example of an interaction between the client and server through WebSocket.

```
//Frontend code emitting Event to backend
function onExit(e) {
    e.preventDefault();
    const data = { id: socketOBJ.id };
    socketOBJ.emit("exitQueue");
    navigate("/");
}
```

```
#Backend Code registering listener
@socket.on('exitQueue')
def exitQueue():
    Queue.query.filter_by(id=request.sid).delete()
    db.session.commit()
    print(request.sid + " exited queue")
```

Communication between Backend and Arduino

Currently, the server is hosted using the Raspberry Pi. Therefore, serial communication is used for communication from Backend to Arduino.

```
#Code to send data to Arduino through Serial Communication
@socket.on('buttonPressed')
def buttonPressed(data):
    text = data
    ser.write(text.encode("utf-8"))
    print(data + " transmited from frontend")
```

```
//Code to parse data from Backend
if (Serial.available() > 0) {
    payload = JSON.parse(Serial.readStringUntil('\n'));
    mode = payload["mode"];
}
```

Arduino Codes & Explanation

Import Code Library to Arduino

- Adafruit NeoPixel Library
- Arduino_JSON Library

```
#include <Adafruit_NeoPixel.h>
#include <Arduino_JSON.h>
#ifdef __AVR__
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif
```

Define variables

```
// declare light strips
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN1, NEO_BRG + NEO_KHZ800), strip3(LED_COUNT,
LED_PIN2, NEO_BRG + NEO_KHZ800);

int brightness = 250;
uint32_t color = strip.Color(0, 0, 255);
int timeInterval = 300;

//For multithreading
long lastTime = 0;

// for beats
int beatIndex = 0;

//for sensors
int maxDistance = 30;
int triggeredNo = 0;

// passing JSON data from raspberry pi to arduino
JSONVar payload;
String mode;
```

The setup() function called once after each powerup or reset of the Arduino board. Use it to initialize variables, pin modes, start using libraries, etc. in our case it's used to initialize the LED strips, ultrasonic sensor trigger and echo pin.

```
void setup() {
  // put your setup code here, to run once:
  strip.begin();
  strip.show();
  strip.setBrightness(brightness);

  strip3.begin();
  strip3.show();
  strip3.setBrightness(brightness);

  Serial.begin(9600);

  pinMode(echoPin1, INPUT);
  pinMode(echoPin2, INPUT);
  pinMode(echoPin3, INPUT);
  pinMode(echoPin4, INPUT);
  pinMode(echoPin5, INPUT);
  pinMode(echoPin6, INPUT);

  pinMode(trigPin1, OUTPUT);
```

```
  pinMode(trigPin2, OUTPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(trigPin4, OUTPUT);
  pinMode(trigPin5, OUTPUT);
  pinMode(trigPin6, OUTPUT);
}
```

The loop() function loops consecutively, allowing the program to change and respond. Allowing Arduino board to be actively control, in our codes it uses if/else statement to allow different function to be called.

```
void loop() {
  if (Serial.available() > 0) {
    payload = JSON.parse(Serial.readStringUntil('\n'));
    mode = payload["mode"];
  }

  long elapsedTime = millis() - lastTime;
  lastTime = lastTime + elapsedTime;

  if (mode == "bread") {
    uint32_t color = strip.Color(int(payload["red"]), int(payload["green"]),
                                 int(payload["blue"]));
    breadColor(elapsedTime);
  }
  else if (mode == "sunflower") {
    uint32_t color = strip.Color(int(payload["red"]), int(payload["green"]),
                                 int(payload["blue"]));
    sunflowerColor(elapsedTime);
  }

  else if (mode == "ripple") {
    bool sensor1Triggered = SensorIsTriggered(trigPin1, echoPin1);
    bool sensor2Triggered = SensorIsTriggered(trigPin2, echoPin2);
    bool sensor3Triggered = SensorIsTriggered(trigPin3, echoPin3);
    bool sensor4Triggered = SensorIsTriggered(trigPin4, echoPin4);
    bool sensor5Triggered = SensorIsTriggered(trigPin5, echoPin5);
    bool sensor6Triggered = SensorIsTriggered(trigPin6, echoPin6);

    if (triggeredNo > 1) {
      CenterRipple(elapsedTime);
    } else if (triggeredNo == 1) {
      if (sensor1Triggered) {
        DirectionRipple1(elapsedTime);
      }
      if (sensor2Triggered) {
        DirectionRipple2(elapsedTime);
      }
      if (sensor3Triggered) {
        DirectionRipple3(elapsedTime);
      }
      if (sensor4Triggered) {
        DirectionRipple4(elapsedTime);
      }
      if (sensor5Triggered) {
        DirectionRipple5(elapsedTime);
      }
      if (sensor6Triggered) {
        DirectionRipple6(elapsedTime);
      }
```

```
    }
  }
  else if (mode == "storymode") {
    storymode(elapsedTime);
  }
  else if (mode == "rgb") {
    uint32_t color = strip.Color(int(payload["red"]),
                     int(payload["green"]), int(payload["blue"]));
    strip.setPin(2);
    strip.fill(color);
    strip.setPin(3);
    strip.fill(color);
    strip.setPin(4);
    strip.fill(color);
  }
}
```

Raise effect:

Raise function (breadColor & sunflowerColor function are similar): Light up one pixel at a time to give the raising effect when ledTime is equal to the beats in the array. Allowing a 700ms time to raise and 300ms to lower. If ledTime is not equals or less than beat it will clear, reset pixel to zero and increase beatIndex.

```
void breadColor(long elapsedTime, uint32_t color) {
  static long ledTime = 0;
  static long raiseTime = 0;
  static long pixel = 0;

  ledTime += elapsedTime;

  if (ledTime >= breadBeat[beatIndex] - 400) {
    raiseTime = ledTime - breadBeat[beatIndex];
    if (raiseTime <= 1000) {
      if (raiseTime < 700) {
        setAllTubes_Beats(color, pixel);
        pixel++;
      }
      else {
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
      }

      strip.show();
      strip3.show();
    } else {
      strip.clear();
      strip.show();
      strip3.clear();
      strip3.show();
      pixel = 0;
      beatIndex += 1;
    }
  }
  else {
    strip.clear();
    strip.show();
    strip3.clear();
    strip3.show();
```

```
  }
}
```

Ripple Effect:

TurnOn & TurnOff functions: get pin number of the strip and set brightness/color for that strip

```
void TurnOn(int pin) {
  strip.setPin(pin);
  strip.fill(color);
  strip.show();

}

void TurnOff(int pin) {
  strip.setPin(pin);
  strip.clear();
  strip.show();
}
```

SensorIsTriggered function: return Boolean based on the distance between object and sensor

```
bool SensorIsTriggered(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  unsigned duration = pulseIn(echoPin, HIGH);
  Serial.println(duration);

  int distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance <= maxDistance) {
    triggeredNo += 1;
    return true;
  } else {
    return false;
  }
}
```

CenterRipple function (DirectionRipple functions are similar): elapsedTime is passed from loop function and used to determine which tubes are supposed to be turned on/off. For each “wave”, the brightness and color are slightly different.

```
void CenterRipple(long elapsedTime) {
  static long ledTime = 0;
```

```
  ledTime += elapsedTime;
  static int counter = 0;
  //  Serial.println(counter);
  if (ledTime < timeInterval) {
    brightness = 255; //Brightness decreases for each ripple wave
    color = strip.Color(0, 0, 255); //Color changes for each ripple wave
    TurnOn(LED_PIN13);
  };
  if (ledTime >= timeInterval && ledTime < timeInterval * 2) {
    brightness = 235;
    color = strip.Color(0, 50, 235);
    TurnOff(LED_PIN13);

    TurnOn(LED_PIN8);
    TurnOn(LED_PIN9);
    TurnOn(LED_PIN12);
    TurnOn(LED_PIN14);
    TurnOn(LED_PIN17);
    TurnOn(LED_PIN18);
  };
```

Story Mode function:

Storymode function: stoory mode has been splited up into five sub function for easier debugging, each sub function will cycle throught the codes for 3mins(180s).

sky_storymode(): lights up all tubes with varying color of blue simulating day time sky.

lightning_storymode(): flashes of white light to simulate lightning, with “sky” turning a darker blue everytime the lightning flash.

rain_storymode(): one pixel lighting up as it moves down the tube, to simulate the falling rain. With tubes lighting up odd or even pixels so the rain will look random.

sunshine_storymode(): lights up all tubes with varying color of yellow simulating the sun coming out again, after the rain.

rainbow_storymode(): lights up an array of rainbow color in a diagonal pattern to simulate rainbow after it rains.

```
static unsigned long storymodeTime = 0;
// one cycle : 180s (180000ms)
// each sub-mode : 36s (36000ms)
void storymode(long elapsedTime) {
  storymodeTime += elapsedTime;

  if (storymodeTime < 36000) {
    sky_storymode();
  } else if (storymodeTime < 47000) {
    lightning_storymode();
  } else if (storymodeTime < 83000) {
    rain_storymode();
  } else if (storymodeTime < 119000) {
    sunshine_storymode();
  } else if (storymodeTime < 155000) {
```

```
    rainbow_storymode();
  } else {
    storymodeTime = 0;
  }
  Serial.println("OUT : " + String(storymodeTime));
}
```

Code of beat(raise), ripple & story mode function can be found in GitHub

```
https://github.com/regyap/IM3080-InteractiveDisplay/blob/main/hardware/Main/Main.ino
```

Software codes

React

-Used websocket connection

Raspberry

-Used bash and autostart scripts

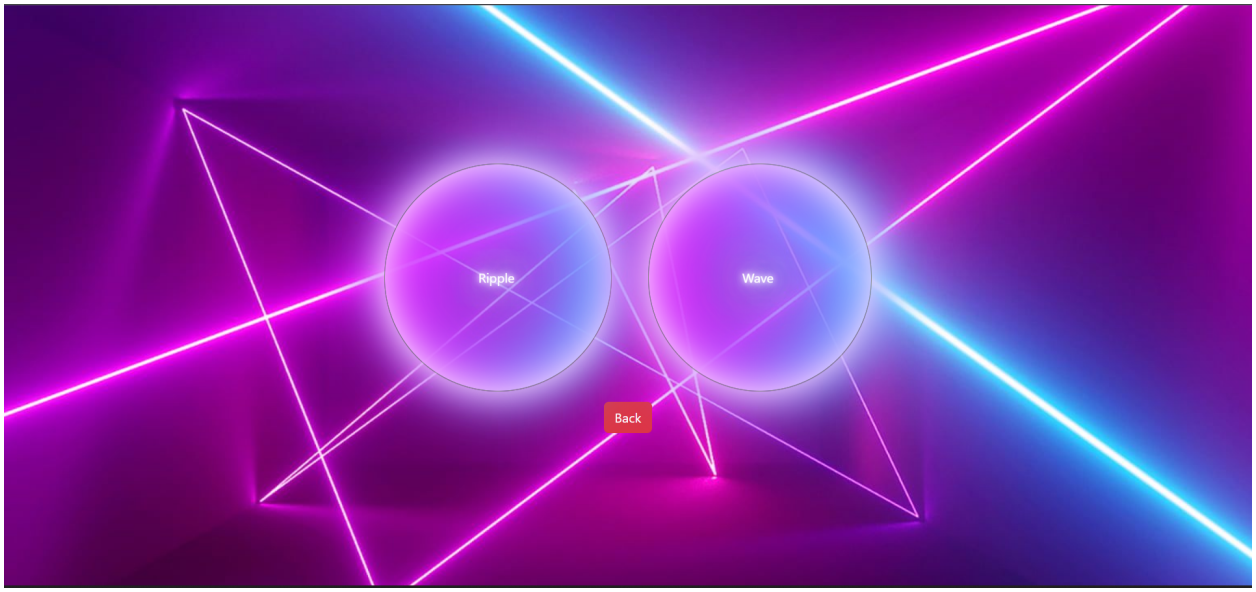
Python

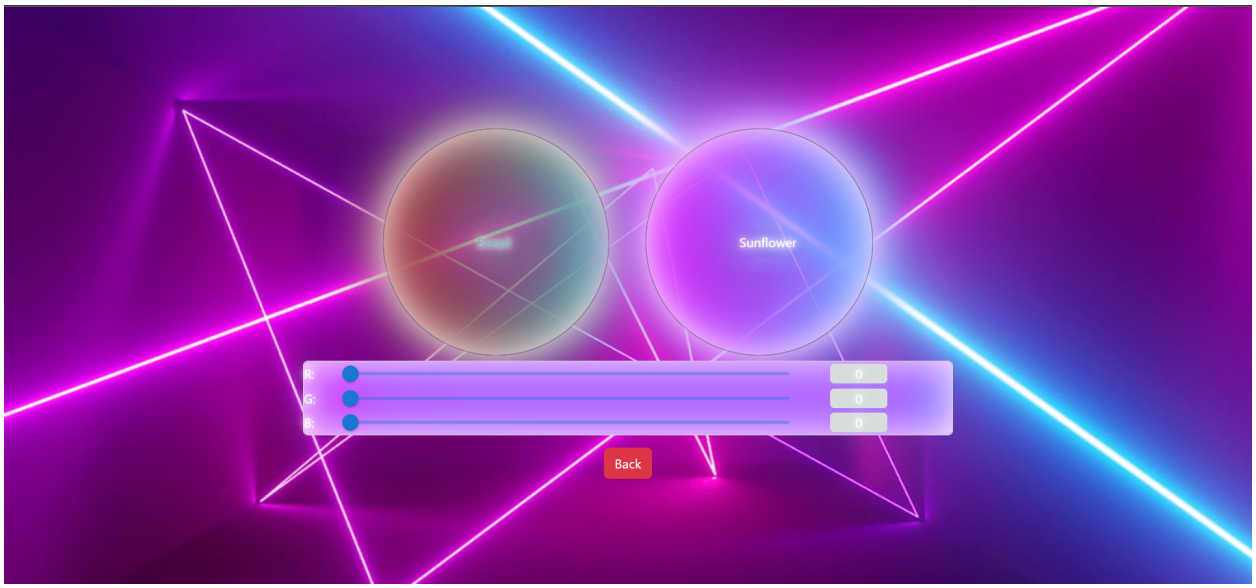
-Used backend to connect to react script

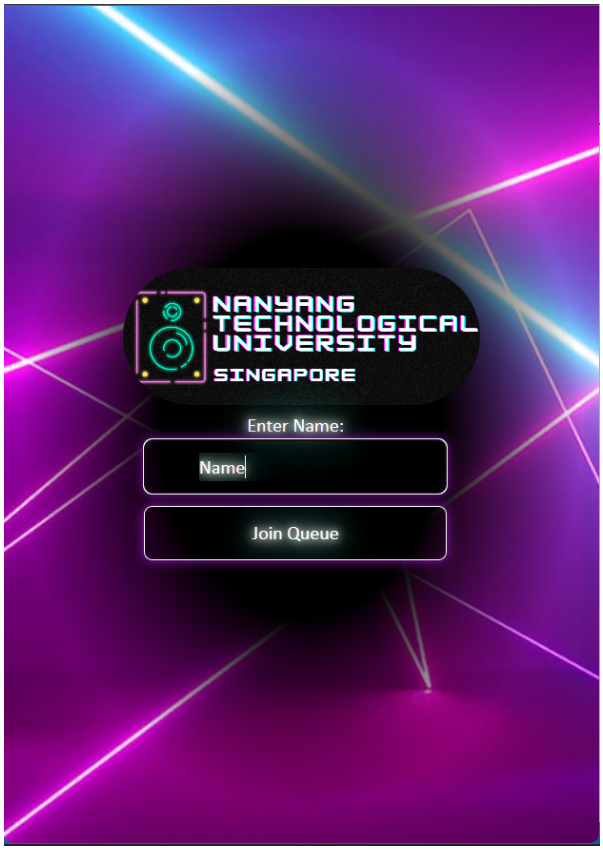
CSS

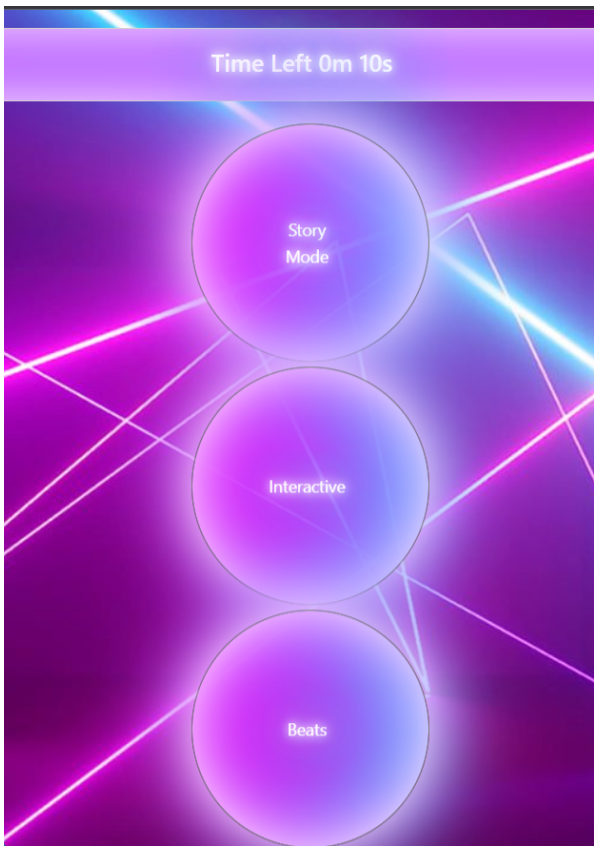
- It is coded in neon theme to match the light theme of the project
- Used Media query to make webpage available as a mobile application
- Utilised hover, animations, transitions and activate to achieve current effect

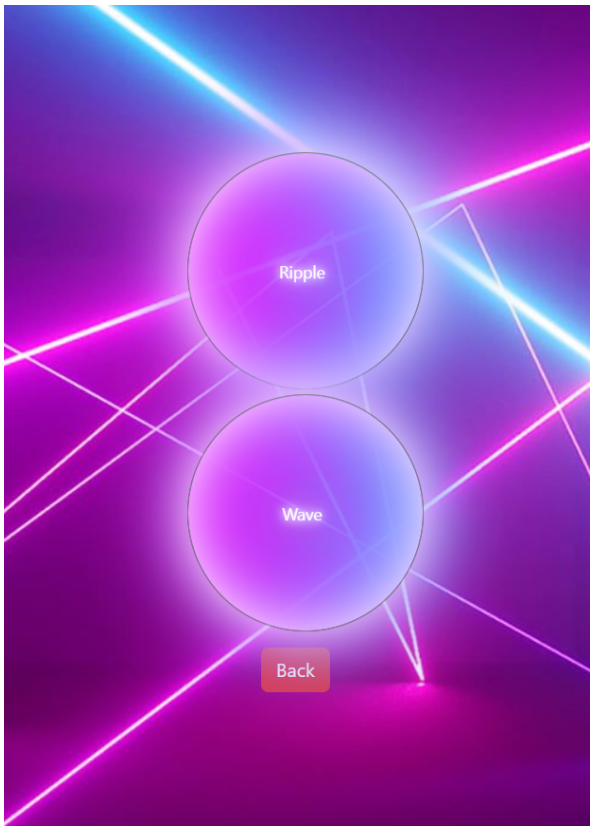
- Picture of the Final Webpage

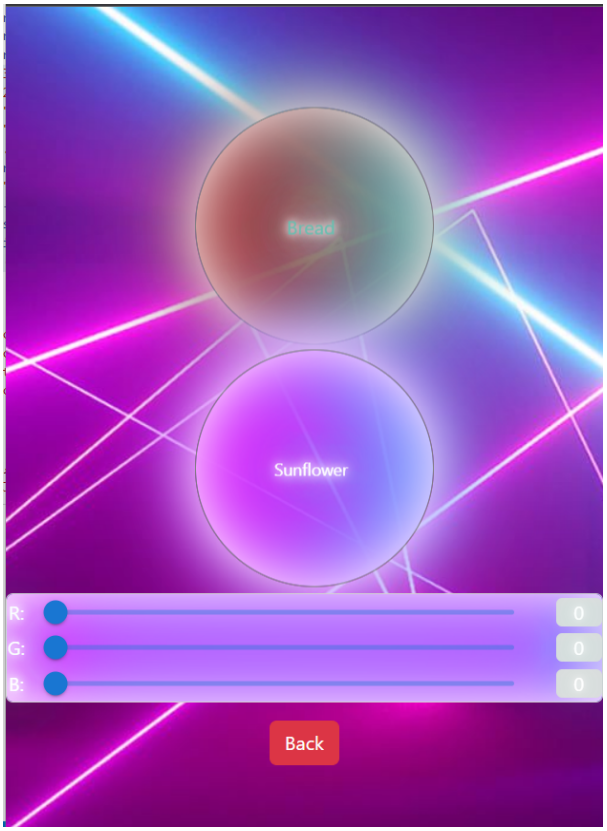

Ripple
Wave
Back


Frost
Sunflower
R:
G:
B:
0
0
0
Back


NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE
Enter Name:
Name
Join Queue


Time Left 0m 10s
Story
Mode
Interactive
Beats


Ripple
Wave
Back


Bread
Sunflower
R:
0
G:
0
B:
0
Back

4.4. Challenges

Acquiring Materials

1. Have to check common store and make a trip to Sim Lim

   There is a lot of moving about to get components. There was a shortage of time as we started on an idea later on other groups, hence we had to quickly buy from simlim despite the far distance.

Hardware

1. How to fix and clean up the wires so that they won't easily tangled or drop off:

   Used the 3D printer to make some ‘T-shape’ wire junctions to arrange and fix the wire. The wires were divided into several groups in horizontal or vertical orientation and were tightly fastened on the floor board.

2. How to arrange the wires to make sure every tubes can reach the power supply or signal pins:

   Used the other kind of junctions that have screws to extend the wire and also double the signal/power. Used an outside breadboard to clean up the wire.

Arduino

1. How to turn on several LED strips simultaneously and keep them on for a short time:

   We first used delay function to keep the lights on for a short while. However, it was not ideal because delay would pause the whole program. Therefore, to achieve multithreading, we defined variables to keep track of the elapsed time and controlled the functions using if statements.

2. How to get the LED strip to light up different patterns at the same time.

   Initially, we coded it such that all strip was using the same declared name “strip” which conflicted with the pattern we wanted. We added another declared names “strip” and “strip3” to allow different strips to run different pattern at the same time.

Software

1. How to implement a system where one user can use the web application without any interference from other users.

   We wanted to create a web application that is able to control the light display. However, we foresee a situation whereby there may be many users controlling the light display at once which may cause problems. To resolve this issue, we implemented a queue management system. When entering the software, the user will be placed onto the queue. Every user will only have a set amount of time to interact with the light display using the software. When the time is up, the user will lose control of the light display and the next user will gain control.

5. Conclusion and Recommendation

5.1. Conclusion

During the whole project, we are divided into 3 sub-team, e.t. Arduino team(4 persons), in charge of arduino coding; Software team(2 persons), in charge of webapp development; Hardware team(3 persons), in charge of hardware design and installation. For project management, we used trello/google document to note down the meeting content, used Github to share the code and used jira to track different works.

We finally made the ideal interactive installation—a LED art piece made of 25 tubes, LED strips, a wooden base and sensors. With software control, the piece can exhibit different features in 3 modes(Music beat mode, story mode, interactive mode).

In music beat mode, users can first adjust the color they want by changing the RGB value. The LED light will beat with the rhythm of the music.

In story mode, the LED light will change according to the characteristics of the natural weather (Blue sky, thunder & lightning, rain, rainbow).

In interactive mode, we have 2 features— ripple and wave. The 6 sensors on the outside of the tubes will detect the distance between the users and the fixture. If it is detected close enough for any of the sensors, the ripple mode will make a

light ripple from the center. Wave mode will make a light wave starting from the tube with the sensor instead.

In the whole process, we have faced challenges like the no proper circuit to the power supply, chaotic wiring, weak connection, test failure and so on. By applying electric engineering knowledge and problem solving skills, we were able to overcome the difficulties.

During the development journey, we have learned how to combine the software and hardware. Theoretical engineer knowledge is applied in practice. Arduino coding and circuit design become more familiar to us.

5.2. Recommendation for Future Works

We can add more additions to the project such as more interactive elements to the tubes.
Future groups can explore how they can incorporate more modes and different types of sensors to make interaction more fun.

Appendices:

1. (For project with hardware expenses) Bill of Materials (BOM) – “All spending” AND “for the project prototype”

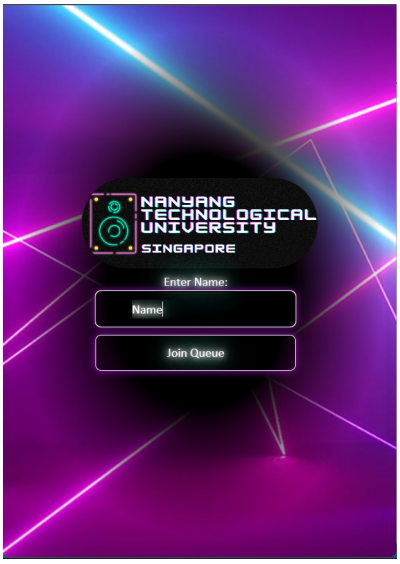
S/N	Item	Quantity	Unit Cost	Total Cost
1	Aluminium Foil	2	$2.40	$4.80
2	Sponge	2	$1.40	$2.80
3	Glue	1	$6	$6
4	Spray Paint	1	$4	$4
5	Styrofoam Board	1	$4.20	$4.20
6	Flat Head	2	$1	$2
7	L bracket	2	$2.20	$4.40
8	Washer	2	$0.80	$1.60

9	Wheels	6	$4.15	$24.90
10	Ultrasonic Sensor HC-SR04	6	$6	$36
11	Art Paper	1 set of 8 pcs	$12.10	$12.10
12	LED	1 set of 3 strips	$41.66	$41.66
13	Double-Sided Tape	1 set of 2 rolls	$1.05	$1.05
14	Tracing Paper	14	$1.20	$16.80
15	6A Connector	10	$0.90	$9
16	Soft Foam	1	$2.80	$2.80
17	Plastic File	1	$1.80	$1.80
18	Cotton Balls	5	$1.15	$5.75
			Total Cost:	$181.66

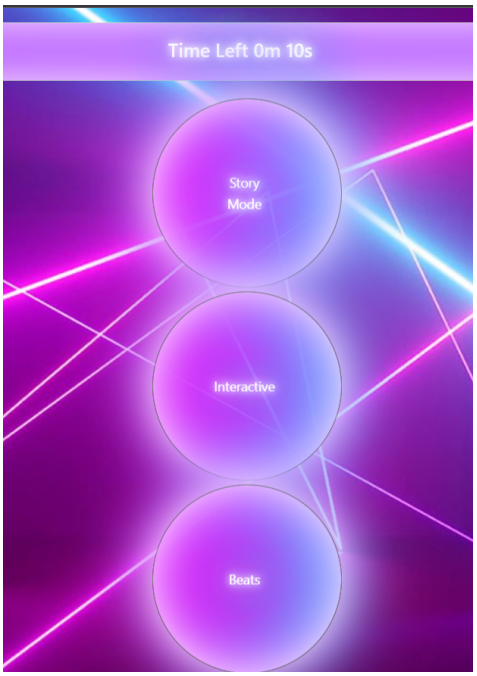
2. Design Diagrams (e.g., detailed circuit diagrams, software engineering diagrams)
    - Refer to 4.3 implementation

3. User Guide

3.1. User approach the User Interface and type in their name to join the queue (as shown in the photo below)



3.2. When it's the user's turn, the timer for their turn will start to countdown, they may choose between three various modes, including interactive mode, story mode, and music beat mode.
(as shown in the photo below)

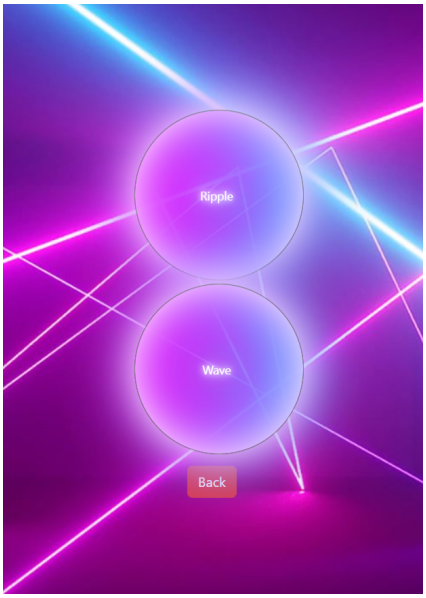


3.2.1. **[Story Mode]** A light show will be triggered, the LED light will alter in story mode in accordance with the features of the natural climate (Blue sky, thunder & lightning, rain, rainbow).

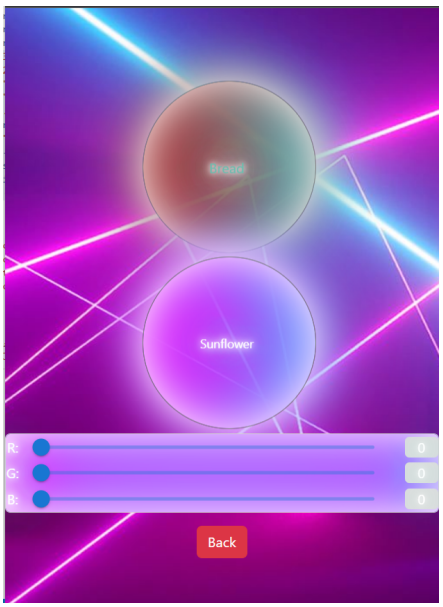
3.2.2. **[Interactive Mode]** There are 2 features in the Interactive Mode, the 6 sensors on the tubes will measure the user's distance from the fixture.

[Wave Feature] Create a light wave beginning with the sensor-equipped tube.

[Ripple Feature] Create a gentle ripple from the center to the outer tubes.



3.2.3. **[Music Beat Mode]** There are 2 music beats in this mode, users can first change the RGB value of the desired color and music beats in Music Beat mode. The LED light will pulse in time with the music.



3.2.4. Once the timer is up, the user's turn will be completed

4. How-To Guides

4.1. How to start python

Cd to folder for backend

activate.bat

cd ../..

pip install -r requirements.txt

Python simpleback.py

4.2. How to start react

Cd to folder for frontend

Npm start

5. Source Code

5.1. Arduino

Import lib

```
#include <Adafruit_NeoPixel.h>
#include <Arduino_JSON.h>
#ifdef __AVR__
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif
```

Declare variables

```
//LED pins [FILL IN THE PIN NO!!]
#define LED_PIN1 27
#define LED_PIN2 28
#define LED_PIN3 29
#define LED_PIN4 30
#define LED_PIN5 31
#define LED_PIN6 32
#define LED_PIN7 33
#define LED_PIN8 22
#define LED_PIN9 35
#define LED_PIN10 36
#define LED_PIN11 37
#define LED_PIN12 38
#define LED_PIN13 39
#define LED_PIN14 40
#define LED_PIN15 41
#define LED_PIN16 42
#define LED_PIN17 43
```

```
#define LED_PIN18 44
#define LED_PIN19 45
#define LED_PIN20 46
#define LED_PIN21 47
#define LED_PIN22 48
#define LED_PIN23 49
#define LED_PIN24 50
#define LED_PIN25 51

//Max led pixels number [FILL IN MAX LED PIXELS NO!!!]
#define LED_COUNT 50

//Ultrasonic sensor pins [FILL IN PIN NO!!!]
#define trigPin1 6
#define trigPin2 9
#define trigPin3 11
#define trigPin4 13
#define trigPin5 24
#define trigPin6 53

#define echoPin1 5
#define echoPin2 8
#define echoPin3 10
#define echoPin4 12
#define echoPin5 23
#define echoPin6 52

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN1, NEO_BRG + NEO_KHZ800), strip3(LED_COUNT,
LED_PIN2, NEO_BRG + NEO_KHZ800);

int brightness = 250;
uint32_t color = strip.Color(0, 0, 255);
int timeInterval = 300;

//For multithreading
long lastTime = 0;

// for beats
int beatIndex = 0;

//for sensors
int maxDistance = 30;
int triggeredNo = 0;

// beat array
long breadBeat[] = {17, 484, 810, 2683, 4180, 4738, 6003, 6359, 6721, 7952, 8488,
                    8809, 10204, 10698, 11459, 13676, 14699, 15996, 16768, 18746,
                    20710, 22732, 23948, 24740, 26720, 27457, 29703, 30743, 31993,
                    32771, 34720, 36717, 38087, 38702, 40486, 40796, 42714, 43461,
                    44281, 45696, 46721, 47996, 48778, 50719, 52715, 54094, 54729,
                    55946, 56719, 58734, 59457, 60227, 61719, 63985, 64789, 66706,
                    68703, 70102, 70739, 71958, 72726, 74709, 75455, 76235, 77706,
                    80682, 82678, 84685, 86694, 88692, 90674, 91495, 92213, 93699,
                    94738, 95745, 96477, 96780, 97082, 98677, 100680, 102682, 103943,
                    104470, 106687, 107498, 108237, 109702, 110692, 112674, 113444,
                    114672, 116679, 117460, 118699, 119945, 120696, 121470, 122671,
                    124225, 124695, 125463, 126687, 127979, 128670, 129438, 130667,
                    132689, 133460, 34685, 135948, 136674, 137471, 138670, 140246,
                    140685, 141458, 142686, 143996, 144682, 145456, 146672, 148687,
```

```
                    149472, 150693, 152704, 154682
                   };
long sunflowerBeat[] = {15, 728, 1459, 2207, 2966, 3671, 4467, 4850, 5217, 5973, 6679,
                        7489, 7861, 8968, 9672, 10463, 10850, 11954, 12664,13490,13850,
                        14233, 14966, 15656, 16474, 16856, 17979, 18649, 19483, 19844,
                        20218, 20978, 21673, 22469, 22854, 23221, 23978, 24659, 25475,
                        25859, 26956, 27657, 28480, 28829, 29229, 29979, 30649, 31464,
                        31844, 32239, 32967, 33668, 34480, 34843, 35982, 36671, 37489,
                        37848, 38211, 38980, 39638, 40467, 40855, 41964, 42656, 43476,
                        43853, 44229, 44966, 45674, 46479, 46837, 47222, 47982, 48655,
                        49482, 49844, 51865, 52523, 53230, 54755, 55294, 55952, 56688,
                        57791, 58151, 58512, 59201, 60748, 61283, 61983, 63903, 65244,
                        66738, 67302, 67969, 68654, 69748, 70168, 71234, 72730, 73285,
                        73950, 74964, 75669, 76469, 76852, 77209, 77976, 78664, 79451,
                        79849, 80970, 81662, 82501, 82838, 83963, 84682, 85476, 85833,
                        86219, 86973, 87650, 88457, 88842, 89966, 90673, 91472, 91855,
                        92210, 92966, 93678, 94481, 94853, 95222, 95973, 96654, 97477,
                        97842, 98964, 99660, 100468, 100852, 101232, 101971, 102655,
                        103456, 103843, 104983, 105661, 106467, 106865, 107967, 108670,
                        109476, 109847, 110216, 110962, 111648, 112486, 112848, 113982,
                        114658, 115484, 115868, 116220, 116961, 117666, 118483, 118829,
                        119981, 120657, 121483, 121846, 123889, 124506, 125253, 126768,
                        127300, 127948, 129872
                       };
```

Setup()

```
 void setup() {
  // put your setup code here, to run once:
  strip.begin();
  strip.show();
  strip.setBrightness(brightness);

  strip3.begin();
  strip3.show();
  strip3.setBrightness(brightness);

  Serial.begin(9600);
  pinMode(echoPin1, INPUT);
  pinMode(echoPin2, INPUT);
  pinMode(echoPin3, INPUT);
  pinMode(echoPin4, INPUT);
  pinMode(echoPin5, INPUT);
  pinMode(echoPin6, INPUT);
  pinMode(trigPin1, OUTPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(trigPin4, OUTPUT);
  pinMode(trigPin5, OUTPUT);
  pinMode(trigPin6, OUTPUT);
}
```

Loop()

```
// from rasberry pi to arduino
JSONVar payload = JSON.parse("{mode: 'bread', red: 255, green: 0, blue: 0}");
String mode = "ripple";

void loop() {
  if (Serial.available() > 0) {
    payload = JSON.parse(Serial.readStringUntil('\n'));
    mode = payload["mode"];
  }

  long elapsedTime = millis() - lastTime;
  lastTime = lastTime + elapsedTime;

  if (mode == "bread") {
    uint32_t color = strip.Color(int(payload["red"]), int(payload["green"]),
int(payload["blue"]));
    breadColor(elapsedTime);
  }  else if (mode == "sunflower") {
    uint32_t color = strip.Color(int(payload["red"]), int(payload["green"]),
int(payload["blue"]));
    sunflowerColor(elapsedTime);
  }  else if (mode == "ripple") {
    // ripple with sensor
    bool sensor1Triggered = SensorIsTriggered(trigPin1, echoPin1);
    bool sensor2Triggered = SensorIsTriggered(trigPin2, echoPin2);
    bool sensor3Triggered = SensorIsTriggered(trigPin3, echoPin3);
    bool sensor4Triggered = SensorIsTriggered(trigPin4, echoPin4);
    bool sensor5Triggered = SensorIsTriggered(trigPin5, echoPin5);
    bool sensor6Triggered = SensorIsTriggered(trigPin6, echoPin6);

    if (triggeredNo > 1) {
      CenterRipple(elapsedTime);
    } else if (triggeredNo == 1) {
      if (sensor1Triggered) {
        DirectionRipple1(elapsedTime);
      }
      if (sensor2Triggered) {
        DirectionRipple2(elapsedTime);
      }
      if (sensor3Triggered) {
        DirectionRipple3(elapsedTime);
      }
      if (sensor4Triggered) {
        DirectionRipple4(elapsedTime);
      }
      if (sensor5Triggered) {
        DirectionRipple5(elapsedTime);
      }
      if (sensor6Triggered) {
        DirectionRipple6(elapsedTime);
      }
    }
  } else if (mode == "storyMode") {
    storymode(elapsedTime);
    Serial.println("storyMode");
  } else if (mode == "rgb") {
    uint32_t color = strip.Color(int(payload["red"]), int(payload["green"]),
int(payload["blue"]));
    strip.setPin(2);
```

```
    strip.fill(color);
    strip.setPin(3);
    strip.fill(color);
    strip.setPin(4);
    strip.fill(color);
  } else if (mode == "test") {
    for (int i = 22; i <= 51; i++) {
      strip.setPin(i);
      strip.fill(strip.Color(255, 0, 0));
      strip.show();
    }
  }
}
```

Story Mode: Main function

```
static unsigned long storymodeTime = 0;

void storymode(long elapsedTime) {

  storymodeTime += elapsedTime;

  if (storymodeTime < 36000) {
    sky_storymode();

  } else if (storymodeTime < 47000) {
    lightning_storymode();

  } else if (storymodeTime < 83000) {
    rain_storymode();

  } else if (storymodeTime < 98000) {
    // sunshine
    sunshine_storymode();

  } else if (storymodeTime < 134000) {
    // rainbow
    rainbow_storymode();
  } else {
    storymodeTime = 0;
  }
  Serial.println("OUT : " + String(storymodeTime));

}
```

Story Mode: Blue sky

```
void sky_storymode() {
  static boolean darken = true;
  static int skyIndex = 98;

  if (darken)
  {
    skyIndex -= 6;
    if (skyIndex <= 2) {
      darken = false;
    }
  } else {
    skyIndex += 6;
    if (skyIndex >= 254) {
      darken = true;
    }
  }
  setAllTubes(strip.Color(0, skyIndex, 255));
}
```

Story Mode: Lightning

```
void lightning_storymode() {

  if (storymodeTime > 37000 && storymodeTime < 37300) {
    setAllTubes(strip.Color(255, 255, 255));                // lightning
  } else if (storymodeTime < 40000) {
    strip.setBrightness(210);
    setAllTubes(strip.Color(0, 32, 127));                   // dark blue sky
  } else if (storymodeTime < 40300) {
    strip.setBrightness(brightness);
    setAllTubes(strip.Color(255, 255, 255));                // lightning
  } else if (storymodeTime < 41000) {
    strip.setBrightness(10);
    setAllTubes(strip.Color(16, 0, 63));                    // dark blue sky
  } else if (storymodeTime < 41300) {
    strip.setBrightness(brightness);
    setAllTubes(strip.Color(255, 255, 255));                // lightning
  } else if (storymodeTime < 41600) {
    strip.setBrightness(10);
    setAllTubes(strip.Color(16, 0, 63));                    // dark blue sky
  } else if (storymodeTime < 42000) {
    strip.setBrightness(brightness);
    setAllTubes(strip.Color(255, 255, 255));                // lightning
  } else {
    setAllTubes(strip.Color(0, 0, 0));                      // no color
  }
}
```

Story Mode: Rain

```
void rain_storymode() {
  const int maxpixels = 30;

  static int pixelCount = 1;
```

```
  for (int i = (maxpixels / 2) - 1; i >= 0; i--) {
    if ((i + pixelCount) % 4 == 0 || (i + pixelCount) % 4 == 1) {
      setAllTubes_odd(strip.Color(0, 0, 0));
      setAllTubes_pixel_odd(strip.Color(255, 255, 255), i);
      setAllTubes_pixel_odd(strip.Color(255, 255, 255), i + (maxpixels / 2));
    } else if ((i + pixelCount) % 4 == 2 || (i + pixelCount) % 4 == 4) {
      setAllTubes_even(strip.Color(0, 0, 0));
      setAllTubes_pixel_even(strip.Color(255, 255, 255), i);
      setAllTubes_pixel_even(strip.Color(255, 255, 255), i + (maxpixels / 2));
    }
  }
  // increment
  if (pixelCount == 4) {
    pixelCount = 1;
  } else {
    pixelCount++;
  }
}
```

Story Mode: Sunshine

```
void sunshine_storymode() {

  static boolean begining = true;
  static int loopCount = 1;
  int maxpixels = 30;

  // flow down
  if (begining == true) {
    for (int i = maxpixels ; i >= 0 ; i--) {
      setAllTubes_pixel(strip.Color(255, 204, 0), i);
    }
    begining = false;
  }

  static boolean darken = true;
  static int sunIndex = 254;

  if (darken) {
    sunIndex -= 3;          // darken
    if (sunIndex <= 204) {
      darken = false;
    }
  } else {                  // lighten
    sunIndex += 3;
    if (sunIndex >= 254) {
      darken = true;
    }
  }
  setAllTubes(strip.Color(255, sunIndex, 0));
  loopCount++;
}
```

Story Mode: Rainbow

```
void rainbow_storymode() {

  static int colorCount = 1;

  if (colorCount / 8 == 1) {
    setRow1Tubes(strip.Color(255, 0, 0));
    setRow2Tubes(strip.Color(255, 64, 0));
    setRow3Tubes(strip.Color(255, 128, 0));
    setRow4Tubes(strip.Color(255, 255, 0));
    setRow5Tubes(strip.Color(0, 255, 0));
    setRow6Tubes(strip.Color(0, 64, 255));
    setRow7Tubes(strip.Color(64, 0, 255));
  } else if (colorCount / 8 == 2) {
    setRow1Tubes(strip.Color(64, 0, 255));
    setRow2Tubes(strip.Color(255, 0, 0));
    setRow3Tubes(strip.Color(255, 64, 0));
    setRow4Tubes(strip.Color(255, 128, 0));
    setRow5Tubes(strip.Color(255, 255, 0));
    setRow6Tubes(strip.Color(0, 255, 0));
    setRow7Tubes(strip.Color(0, 64, 255));
  } else if (colorCount / 8 == 3) {
    setRow1Tubes(strip.Color(0, 64, 255));
    setRow2Tubes(strip.Color(64, 0, 255));
    setRow3Tubes(strip.Color(255, 0, 0));
    setRow4Tubes(strip.Color(255, 64, 0));
    setRow5Tubes(strip.Color(255, 128, 0));
    setRow6Tubes(strip.Color(255, 255, 0));
    setRow7Tubes(strip.Color(0, 255, 0));
  } else if (colorCount / 8 == 4) {
    setRow1Tubes(strip.Color(0, 255, 0));
    setRow2Tubes(strip.Color(0, 64, 255));
    setRow3Tubes(strip.Color(64, 0, 255));
    setRow4Tubes(strip.Color(255, 0, 0));
    setRow5Tubes(strip.Color(255, 64, 0));
    setRow6Tubes(strip.Color(255, 128, 0));
    setRow7Tubes(strip.Color(255, 255, 0));
  } else if (colorCount / 8 == 5) {
    setRow1Tubes(strip.Color(255, 255, 0));
    setRow2Tubes(strip.Color(0, 255, 0));
    setRow3Tubes(strip.Color(0, 64, 255));
    setRow4Tubes(strip.Color(64, 0, 255));
    setRow5Tubes(strip.Color(255, 0, 0));
    setRow6Tubes(strip.Color(255, 64, 0));
    setRow7Tubes(strip.Color(255, 128, 0));
  } else if (colorCount / 8 == 6) {
    setRow1Tubes(strip.Color(255, 128, 0));
    setRow2Tubes(strip.Color(255, 255, 0));
    setRow3Tubes(strip.Color(0, 255, 0));
    setRow4Tubes(strip.Color(0, 64, 255));
    setRow5Tubes(strip.Color(64, 0, 255));
    setRow6Tubes(strip.Color(255, 0, 0));
    setRow7Tubes(strip.Color(255, 64, 0));
  } else if (colorCount / 8 == 7) {
    setRow1Tubes(strip.Color(255, 64, 0));
    setRow2Tubes(strip.Color(255, 128, 0));
    setRow3Tubes(strip.Color(255, 255, 0));
    setRow4Tubes(strip.Color(0, 255, 0));
    setRow5Tubes(strip.Color(0, 64, 255));
    setRow6Tubes(strip.Color(64, 0, 255));
```

```
    setRow7Tubes(strip.Color(255, 0, 0));
  }

  // increment
  if (colorCount == 56) {
    colorCount = 1;
  } else {
    colorCount++;
  }
}
```

Raise: bread

```
void breadColor(long elapsedTime, uint32_t color) {
  static long ledTime = 0;
  static long raiseTime = 0;
  static long pixel = 0;

  ledTime += elapsedTime;

  if (ledTime >= breadBeat[beatIndex] - 400) {
    raiseTime = ledTime - breadBeat[beatIndex];
    if (raiseTime <= 1000) {
      if (raiseTime < 700) {
        setAllTubes_Beats(color, pixel);
        pixel++;
        setAllTubes_Beats(color, pixel);
        pixel++;
        setAllTubes_Beats(color, pixel);
        pixel++;
        setAllTubes_Beats(color, pixel);
        pixel++;
        setAllTubes_Beats(color, pixel);
        pixel++;
      } else {
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
      }
      strip.show();
      strip3.show();
    } else {
      strip.clear();
      strip.show();
      strip3.clear();
      strip3.show();
      pixel = 0;
      beatIndex += 1;
    }
  }
  else {
```

```
    strip.clear();
    strip.show();
    strip3.clear();
    strip3.show();
  }
}
```

Raise: sunflower

```
void sunflowerColor(long elapsedTime, uint32_t color) {
  static long ledTime = 0;
  static long raiseTime = 0;
  static long pixel = 0;

  ledTime += elapsedTime;

  if (ledTime >= sunflowerBeat[beatIndex] - 400) {
    raiseTime = ledTime - sunflowerBeat[beatIndex];
    if (raiseTime <= 1000) {
      if (raiseTime < 700) {
        setAllTubes_Beats(color, pixel);
        pixel++;
        setAllTubes_Beats(color, pixel);
        pixel++;
        setAllTubes_Beats(color, pixel);
        pixel++;
        setAllTubes_Beats(color, pixel);
        pixel++;
        setAllTubes_Beats(color, pixel);
        pixel++;
      } else {
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
        setAllTubes_Beats(strip.Color(0, 0, 0), pixel);
        pixel--;
      }
      strip.show();
      strip3.show();
    } else {
      strip.clear();
      strip.show();
      strip3.clear();
      strip3.show();
      pixel = 0;
      beatIndex += 1;
    }
  } else {
    strip.clear();
    strip.show();
    strip3.clear();
    strip3.show();
  }
```

```
}
```

Ripple: CenterRipple

```
void CenterRipple(long elapsedTime) {
  static long ledTime = 0;
  ledTime += elapsedTime;
  static int counter = 0;
  if (ledTime < timeInterval) {
    brightness = 255; //Brightness decreases for each ripple wave
    color = strip.Color(0, 0, 255); //Color changes for each ripple wave
    TurnOn(LED_PIN13);
  };
  if (ledTime >= timeInterval && ledTime < timeInterval * 2) {
    brightness = 235;
    color = strip.Color(0, 50, 235);
    TurnOff(LED_PIN13);

    TurnOn(LED_PIN8);
    TurnOn(LED_PIN9);
    TurnOn(LED_PIN12);
    TurnOn(LED_PIN14);
    TurnOn(LED_PIN17);
    TurnOn(LED_PIN18);
  };
  if (ledTime >= timeInterval * 2 && ledTime < timeInterval * 3) {
    brightness = 215;
    color = strip.Color(0, 100, 215);
    TurnOff(LED_PIN8);
    TurnOff(LED_PIN9);
    TurnOff(LED_PIN12);
    TurnOff(LED_PIN14);
    TurnOff(LED_PIN17);
    TurnOff(LED_PIN18);

    TurnOn(LED_PIN3);
    TurnOn(LED_PIN4);
    TurnOn(LED_PIN5);
    TurnOn(LED_PIN7);
    TurnOn(LED_PIN10);
    TurnOn(LED_PIN11);
    TurnOn(LED_PIN15);
    TurnOn(LED_PIN16);
    TurnOn(LED_PIN19);
    TurnOn(LED_PIN21);
    TurnOn(LED_PIN22);
    TurnOn(LED_PIN23);
  };
  if (ledTime >= timeInterval * 3 && ledTime < timeInterval * 4) {
    brightness = 195;
    color = strip.Color(0, 150, 195);
    TurnOff(LED_PIN3);
    TurnOff(LED_PIN4);
    TurnOff(LED_PIN5);
    TurnOff(LED_PIN7);
```

```
    TurnOff(LED_PIN10);
    TurnOff(LED_PIN11);
    TurnOff(LED_PIN15);
    TurnOff(LED_PIN16);
    TurnOff(LED_PIN19);
    TurnOff(LED_PIN21);
    TurnOff(LED_PIN22);
    TurnOff(LED_PIN23);

    TurnOn(LED_PIN1);
    TurnOn(LED_PIN2);
    TurnOn(LED_PIN6);
    TurnOn(LED_PIN20);
    TurnOn(LED_PIN24);
    TurnOn(LED_PIN25);
  };
  if (ledTime >= timeInterval * 4) {
    TurnOff(LED_PIN1);
    TurnOff(LED_PIN2);
    TurnOff(LED_PIN6);
    TurnOff(LED_PIN20);
    TurnOff(LED_PIN24);
    TurnOff(LED_PIN25);
  }
}
```

Ripple: DirectionRipple

```
void DirectionRipple1(long elapsedTime) {
  static long ledTime = 0;
  ledTime += elapsedTime;

  if (ledTime <= timeInterval) {
    brightness = 255;
    color = strip.Color(0, 0, 255); //Color changes for each ripple wave
    TurnOn(LED_PIN1);
  }  if (ledTime >= timeInterval && ledTime < timeInterval * 2) {
    brightness = 240;
    color = strip.Color(0, 40, 235);
    TurnOff(LED_PIN1);

    TurnOn(LED_PIN2);
    TurnOn(LED_PIN3);
    TurnOn(LED_PIN4);
    TurnOn(LED_PIN5);
    TurnOn(LED_PIN6);
  }  if (ledTime >= timeInterval * 2 && ledTime < timeInterval * 3) {
    brightness = 225;
    color = strip.Color(0, 80, 215);
    TurnOff(LED_PIN2);
    TurnOff(LED_PIN3);
    TurnOff(LED_PIN4);
    TurnOff(LED_PIN5);
    TurnOff(LED_PIN6);

    TurnOn(LED_PIN7);
    TurnOn(LED_PIN8);
    TurnOn(LED_PIN9);
    TurnOn(LED_PIN10);
  }  if (ledTime >= timeInterval * 3 && ledTime < timeInterval * 4) {
```

```
    brightness = 210;
    color = strip.Color(0, 120, 195);
    TurnOff(LED_PIN7);
    TurnOff(LED_PIN8);
    TurnOff(LED_PIN9);
    TurnOff(LED_PIN10);

    TurnOn(LED_PIN11);
    TurnOn(LED_PIN12);
    TurnOn(LED_PIN13);
    TurnOn(LED_PIN14);
    TurnOn(LED_PIN15);
  }  if (ledTime >= timeInterval * 4 && ledTime < timeInterval * 5) {
    brightness = 195;
    color = strip.Color(0, 160, 175);
    TurnOff(LED_PIN11);
    TurnOff(LED_PIN12);
    TurnOff(LED_PIN13);
    TurnOff(LED_PIN14);
    TurnOff(LED_PIN15);

    TurnOn(LED_PIN16);
    TurnOn(LED_PIN17);
    TurnOn(LED_PIN18);
    TurnOn(LED_PIN19);
  }  if (ledTime >= timeInterval * 5 && ledTime < timeInterval * 6) {
    brightness = 180;
    color = strip.Color(0, 200, 155);
    TurnOff(LED_PIN16);
    TurnOff(LED_PIN17);
    TurnOff(LED_PIN18);
    TurnOff(LED_PIN19);

    TurnOn(LED_PIN20);
    TurnOn(LED_PIN21);
    TurnOn(LED_PIN22);
    TurnOn(LED_PIN23);
    TurnOn(LED_PIN24);
  }  if (ledTime >= timeInterval * 6 && ledTime < timeInterval * 7) {
    brightness = 165;
    color = strip.Color(0, 240, 135);
    TurnOff(LED_PIN20);
    TurnOff(LED_PIN21);
    TurnOff(LED_PIN22);
    TurnOff(LED_PIN23);
    TurnOff(LED_PIN24);

    TurnOn(LED_PIN25);
  }  if (ledTime >= timeInterval * 7) {
    TurnOff(LED_PIN25);
    ledTime -= timeInterval * 7;
  }
}

void DirectionRipple2(long elapsedTime) {
  static long ledTime = 0;
  ledTime += elapsedTime;
  if (ledTime <= timeInterval) {
    brightness = 255;
    color = strip.Color(0, 0, 255);
    TurnOn(LED_PIN2);
  }  if (ledTime >= timeInterval && ledTime < timeInterval * 2) {
    brightness = 240;
    color = strip.Color(0, 40, 235);
    TurnOff(LED_PIN2);

    TurnOn(LED_PIN1);
```

```
  TurnOn(LED_PIN3);
  TurnOn(LED_PIN7);
  TurnOn(LED_PIN11);
} if (ledTime >= timeInterval * 2 && ledTime < timeInterval * 3) {
  brightness = 225;
  color = strip.Color(0, 80, 215);
  TurnOff(LED_PIN1);
  TurnOff(LED_PIN3);
  TurnOff(LED_PIN7);
  TurnOff(LED_PIN11);

  TurnOn(LED_PIN4);
  TurnOn(LED_PIN8);
  TurnOn(LED_PIN12);
  TurnOn(LED_PIN16);
  TurnOn(LED_PIN20);
} if (ledTime >= timeInterval * 3 && ledTime < timeInterval * 4) {
  brightness = 210;
  color = strip.Color(0, 120, 195);
  TurnOff(LED_PIN4);
  TurnOff(LED_PIN8);
  TurnOff(LED_PIN12);
  TurnOff(LED_PIN16);
  TurnOff(LED_PIN20);

  TurnOn(LED_PIN5);
  TurnOn(LED_PIN9);
  TurnOn(LED_PIN13);
  TurnOn(LED_PIN17);
  TurnOn(LED_PIN21);
} if (ledTime >= timeInterval * 4 && ledTime < timeInterval * 5) {
  brightness = 195;
  color = strip.Color(0, 160, 175);
  TurnOff(LED_PIN5);
  TurnOff(LED_PIN9);
  TurnOff(LED_PIN13);
  TurnOff(LED_PIN17);
  TurnOff(LED_PIN21);

  TurnOn(LED_PIN6);
  TurnOn(LED_PIN10);
  TurnOn(LED_PIN14);
  TurnOn(LED_PIN18);
  TurnOn(LED_PIN22);
} if (ledTime >= timeInterval * 5 && ledTime < timeInterval * 6) {
  brightness = 180;
  color = strip.Color(0, 200, 155);
  TurnOff(LED_PIN6);
  TurnOff(LED_PIN10);
  TurnOff(LED_PIN14);
  TurnOff(LED_PIN18);
  TurnOff(LED_PIN22);

  TurnOn(LED_PIN15);
  TurnOn(LED_PIN19);
  TurnOn(LED_PIN23);
  TurnOn(LED_PIN25);
} if (ledTime >= timeInterval * 6 && ledTime < timeInterval * 7) {
  brightness = 165;
  color = strip.Color(0, 240, 135);
  TurnOff(LED_PIN15);
  TurnOff(LED_PIN19);
  TurnOff(LED_PIN23);
  TurnOff(LED_PIN25);

  TurnOn(LED_PIN24);
} if (ledTime >= timeInterval * 7) {
```

```
    TurnOff(LED_PIN24);
    ledTime -= timeInterval * 7;
  }
}

void DirectionRipple3(long elapsedTime) {
  static long ledTime = 0;
  ledTime += elapsedTime;

  if (ledTime <= timeInterval) {
    brightness = 255;
    color = strip.Color(0, 0, 255);
    TurnOn(LED_PIN6);
  }  if (ledTime >= timeInterval && ledTime < timeInterval * 2) {
    brightness = 240;
    color = strip.Color(0, 40, 235);
    TurnOff(LED_PIN6);

    TurnOn(LED_PIN1);
    TurnOn(LED_PIN5);
    TurnOn(LED_PIN10);
    TurnOn(LED_PIN15);
  }  if (ledTime >= timeInterval * 2 && ledTime < timeInterval * 3) {
    brightness = 225;
    color = strip.Color(0, 80, 215);
    TurnOff(LED_PIN1);
    TurnOff(LED_PIN5);
    TurnOff(LED_PIN10);
    TurnOff(LED_PIN15);

    TurnOn(LED_PIN4);
    TurnOn(LED_PIN9);
    TurnOn(LED_PIN14);
    TurnOn(LED_PIN19);
    TurnOn(LED_PIN24);
  }  if (ledTime >= timeInterval * 3 && ledTime < timeInterval * 4) {
    brightness = 210;
    color = strip.Color(0, 120, 195);
    TurnOff(LED_PIN4);
    TurnOff(LED_PIN9);
    TurnOff(LED_PIN14);
    TurnOff(LED_PIN19);
    TurnOff(LED_PIN24);

    TurnOn(LED_PIN3);
    TurnOn(LED_PIN8);
    TurnOn(LED_PIN13);
    TurnOn(LED_PIN18);
    TurnOn(LED_PIN23);
  }  if (ledTime >= timeInterval * 4 && ledTime < timeInterval * 5) {
    brightness = 195;
    color = strip.Color(0, 160, 175);
    TurnOff(LED_PIN3);
    TurnOff(LED_PIN8);
    TurnOff(LED_PIN13);
    TurnOff(LED_PIN18);
    TurnOff(LED_PIN23);

    TurnOn(LED_PIN2);
    TurnOn(LED_PIN7);
    TurnOn(LED_PIN12);
    TurnOn(LED_PIN17);
    TurnOn(LED_PIN22);
  }  if (ledTime >= timeInterval * 5 && ledTime < timeInterval * 6) {
    brightness = 180;
    color = strip.Color(0, 200, 155);
    TurnOff(LED_PIN2);
```

```
    TurnOff(LED_PIN7);
    TurnOff(LED_PIN12);
    TurnOff(LED_PIN17);
    TurnOff(LED_PIN22);

    TurnOn(LED_PIN11);
    TurnOn(LED_PIN16);
    TurnOn(LED_PIN21);
    TurnOn(LED_PIN25);
  }  if (ledTime >= timeInterval * 6 && ledTime < timeInterval * 7) {
    brightness = 165;
    color = strip.Color(0, 240, 135);
    TurnOff(LED_PIN11);
    TurnOff(LED_PIN16);
    TurnOff(LED_PIN21);
    TurnOff(LED_PIN25);

    TurnOn(LED_PIN20);
  }  if (ledTime >= timeInterval * 7) {
    TurnOff(LED_PIN20);
    ledTime -= timeInterval * 7;
  }
}

void DirectionRipple4(long elapsedTime) {
  static long ledTime = 0;
  ledTime += elapsedTime;

  if (ledTime <= timeInterval) {
    brightness = 255;
    color = strip.Color(0, 0, 255);
    TurnOn(LED_PIN20);
  }  if (ledTime >= timeInterval && ledTime < timeInterval * 2) {
    brightness = 240;
    color = strip.Color(0, 40, 235);
    TurnOff(LED_PIN20);

    TurnOn(LED_PIN11);
    TurnOn(LED_PIN16);
    TurnOn(LED_PIN21);
    TurnOn(LED_PIN25);
  }  if (ledTime >= timeInterval * 2 && ledTime < timeInterval * 3) {
    brightness = 225;
    color = strip.Color(0, 80, 215);
    TurnOff(LED_PIN11);
    TurnOff(LED_PIN16);
    TurnOff(LED_PIN21);
    TurnOff(LED_PIN25);

    TurnOn(LED_PIN2);
    TurnOn(LED_PIN7);
    TurnOn(LED_PIN12);
    TurnOn(LED_PIN17);
    TurnOn(LED_PIN22);
  }  if (ledTime >= timeInterval * 3 && ledTime < timeInterval * 4) {
    brightness = 210;
    color = strip.Color(0, 120, 195);
    TurnOff(LED_PIN2);
    TurnOff(LED_PIN7);
    TurnOff(LED_PIN12);
    TurnOff(LED_PIN17);
    TurnOff(LED_PIN22);

    TurnOn(LED_PIN3);
    TurnOn(LED_PIN8);
    TurnOn(LED_PIN13);
    TurnOn(LED_PIN18);
```

```
    TurnOn(LED_PIN23);
  }  if (ledTime >= timeInterval * 4 && ledTime < timeInterval * 5) {
    brightness = 195;
    color = strip.Color(0, 160, 175);
    TurnOff(LED_PIN3);
    TurnOff(LED_PIN8);
    TurnOff(LED_PIN13);
    TurnOff(LED_PIN18);
    TurnOff(LED_PIN23);

    TurnOn(LED_PIN4);
    TurnOn(LED_PIN9);
    TurnOn(LED_PIN14);
    TurnOn(LED_PIN19);
    TurnOn(LED_PIN24);
  }  if (ledTime >= timeInterval * 5 && ledTime < timeInterval * 6) {
    brightness = 180;
    color = strip.Color(0, 200, 155);
    TurnOff(LED_PIN4);
    TurnOff(LED_PIN9);
    TurnOff(LED_PIN14);
    TurnOff(LED_PIN19);
    TurnOff(LED_PIN24);

    TurnOn(LED_PIN1);
    TurnOn(LED_PIN5);
    TurnOn(LED_PIN10);
    TurnOn(LED_PIN15);
  }  if (ledTime >= timeInterval * 6 && ledTime < timeInterval * 7) {
    brightness = 165;
    color = strip.Color(0, 240, 135);
    TurnOff(LED_PIN1);
    TurnOff(LED_PIN5);
    TurnOff(LED_PIN10);
    TurnOff(LED_PIN15);

    TurnOn(LED_PIN6);
  }  if (ledTime >= timeInterval * 7) {
    TurnOff(LED_PIN6);
    ledTime -= timeInterval * 7;
  }
}----

void DirectionRipple5(long elapsedTime) {
  static long ledTime = 0;
  ledTime += elapsedTime;

  static int counter = 0;
  //  Serial.println(counter);
  if (ledTime <= timeInterval) {
    brightness = 255;
    color = strip.Color(0, 0, 255);
    TurnOn(LED_PIN24);
  }  if (ledTime >= timeInterval && ledTime < timeInterval * 2) {
    brightness = 240;
    color = strip.Color(0, 40, 235);
    TurnOff(LED_PIN24);

    TurnOn(LED_PIN15);
    TurnOn(LED_PIN19);
    TurnOn(LED_PIN23);
    TurnOn(LED_PIN25);
  }  if (ledTime >= timeInterval * 2 && ledTime < timeInterval * 3) {
    brightness = 225;
    color = strip.Color(0, 80, 215);
    TurnOff(LED_PIN15);
    TurnOff(LED_PIN19);
```

```
    TurnOff(LED_PIN23);
    TurnOff(LED_PIN25);

    TurnOn(LED_PIN6);
    TurnOn(LED_PIN10);
    TurnOn(LED_PIN14);
    TurnOn(LED_PIN18);
    TurnOn(LED_PIN22);
  }  if (ledTime >= timeInterval * 3 && ledTime < timeInterval * 4) {
    brightness = 210;
    color = strip.Color(0, 120, 195);
    TurnOff(LED_PIN6);
    TurnOff(LED_PIN10);
    TurnOff(LED_PIN14);
    TurnOff(LED_PIN18);
    TurnOff(LED_PIN22);

    TurnOn(LED_PIN5);
    TurnOn(LED_PIN9);
    TurnOn(LED_PIN13);
    TurnOn(LED_PIN17);
    TurnOn(LED_PIN21);
  }  if (ledTime >= timeInterval * 4 && ledTime < timeInterval * 5) {
    brightness = 195;
    color = strip.Color(0, 160, 175);
    TurnOff(LED_PIN5);
    TurnOff(LED_PIN9);
    TurnOff(LED_PIN13);
    TurnOff(LED_PIN17);
    TurnOff(LED_PIN21);

    TurnOn(LED_PIN4);
    TurnOn(LED_PIN8);
    TurnOn(LED_PIN12);
    TurnOn(LED_PIN16);
    TurnOn(LED_PIN20);
  }  if (ledTime >= timeInterval * 5 && ledTime < timeInterval * 6) {
    brightness = 180;
    color = strip.Color(0, 200, 155);
    TurnOff(LED_PIN4);
    TurnOff(LED_PIN8);
    TurnOff(LED_PIN12);
    TurnOff(LED_PIN16);
    TurnOff(LED_PIN20);

    TurnOn(LED_PIN1);
    TurnOn(LED_PIN3);
    TurnOn(LED_PIN7);
    TurnOn(LED_PIN11);
  }  if (ledTime >= timeInterval * 6 && ledTime < timeInterval * 7) {
    brightness = 165;
    color = strip.Color(0, 240, 135);
    TurnOff(LED_PIN1);
    TurnOff(LED_PIN3);
    TurnOff(LED_PIN7);
    TurnOff(LED_PIN11);

    TurnOn(LED_PIN2);
  }  if (ledTime >= timeInterval * 7) {
    TurnOff(LED_PIN2);
    ledTime -= timeInterval * 7;
  }
}

void DirectionRipple6(long elapsedTime) {
  static long ledTime = 0;
  ledTime += elapsedTime;
```

```
if (ledTime <= timeInterval) {
  brightness = 255;
  color = strip.Color(0, 0, 255);
  TurnOn(LED_PIN25);
}  if (ledTime >= timeInterval && ledTime < timeInterval * 2) {
  brightness = 240;
  color = strip.Color(0, 40, 235);
  TurnOff(LED_PIN25);

  TurnOn(LED_PIN20);
  TurnOn(LED_PIN21);
  TurnOn(LED_PIN22);
  TurnOn(LED_PIN23);
  TurnOn(LED_PIN24);
}  if (ledTime >= timeInterval * 2 && ledTime < timeInterval * 3) {
  brightness = 225;
  color = strip.Color(0, 80, 215);
  TurnOff(LED_PIN20);
  TurnOff(LED_PIN21);
  TurnOff(LED_PIN22);
  TurnOff(LED_PIN23);
  TurnOff(LED_PIN24);

  TurnOn(LED_PIN16);
  TurnOn(LED_PIN17);
  TurnOn(LED_PIN18);
  TurnOn(LED_PIN19);
}  if (ledTime >= timeInterval * 3 && ledTime < timeInterval * 4) {
  brightness = 210;
  color = strip.Color(0, 120, 195);
  TurnOff(LED_PIN16);
  TurnOff(LED_PIN17);
  TurnOff(LED_PIN18);
  TurnOff(LED_PIN19);

  TurnOn(LED_PIN11);
  TurnOn(LED_PIN12);
  TurnOn(LED_PIN13);
  TurnOn(LED_PIN14);
  TurnOn(LED_PIN15);
}  if (ledTime >= timeInterval * 4 && ledTime < timeInterval * 5) {
  brightness = 195;
  color = strip.Color(0, 160, 175);
  TurnOff(LED_PIN11);
  TurnOff(LED_PIN12);
  TurnOff(LED_PIN13);
  TurnOff(LED_PIN14);
  TurnOff(LED_PIN15);

  TurnOn(LED_PIN7);
  TurnOn(LED_PIN8);
  TurnOn(LED_PIN9);
  TurnOn(LED_PIN10);
}  if (ledTime >= timeInterval * 5 && ledTime < timeInterval * 6) {
  brightness = 180;
  color = strip.Color(0, 200, 155);
  TurnOff(LED_PIN7);
  TurnOff(LED_PIN8);
  TurnOff(LED_PIN9);
  TurnOff(LED_PIN10);

  TurnOn(LED_PIN2);
  TurnOn(LED_PIN3);
  TurnOn(LED_PIN4);
  TurnOn(LED_PIN5);
  TurnOn(LED_PIN6);
```

```
  }  if (ledTime >= timeInterval * 6 && ledTime < timeInterval * 7) {
    brightness = 165;
    color = strip.Color(0, 240, 135);
    TurnOff(LED_PIN2);
    TurnOff(LED_PIN3);
    TurnOff(LED_PIN4);
    TurnOff(LED_PIN5);
    TurnOff(LED_PIN6);

    TurnOn(LED_PIN1);
  }  if (ledTime >= timeInterval * 7) {
    TurnOff(LED_PIN1);
    ledTime -= timeInterval * 7;
  }
}
```

Ripple: SensorIsTriggered

```
bool SensorIsTriggered(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  unsigned duration = pulseIn(echoPin, HIGH);
  Serial.println(duration);

  int distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance <= maxDistance) {
    triggeredNo += 1;
    return true;
  } else {
    return false;
  }
}
```

Ripple: TurnOn & TurnOff

```
void TurnOn(int pin) {
  strip.setPin(pin);
  strip.fill(color);
  strip.show();

}

void TurnOff(int pin) {
  strip.setPin(pin);
  strip.clear();
  strip.show();
```

```
}
```

Github link:
https://github.com/regyap/IM3080-InteractiveDisplay/blob/main/hardware/Main/Main.ino

References

Hardware:

Arduino:

https://adafruit.github.io/Adafruit_NeoPixel/html/class_adafruit___neo_pixel.html#aac61dab69181d4554be8cdde40e4bce6
https://youtu.be/6YUVsCSIbb8
https://github.com/arduino-libraries/Arduino_JSON
https://forum.arduino.cc/t/command-to-run-2-or-more-led-strips-simultaneously/596625/4

Software: