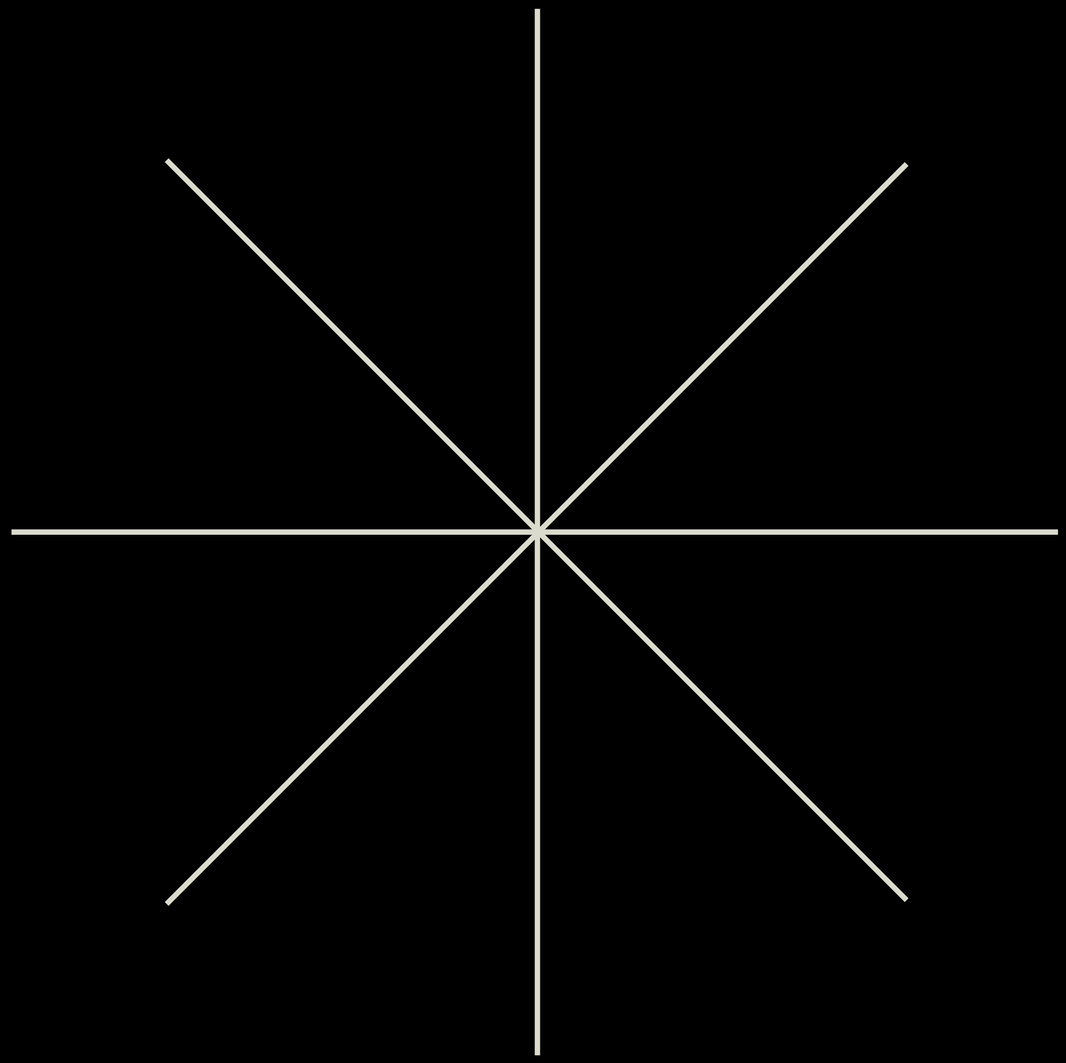
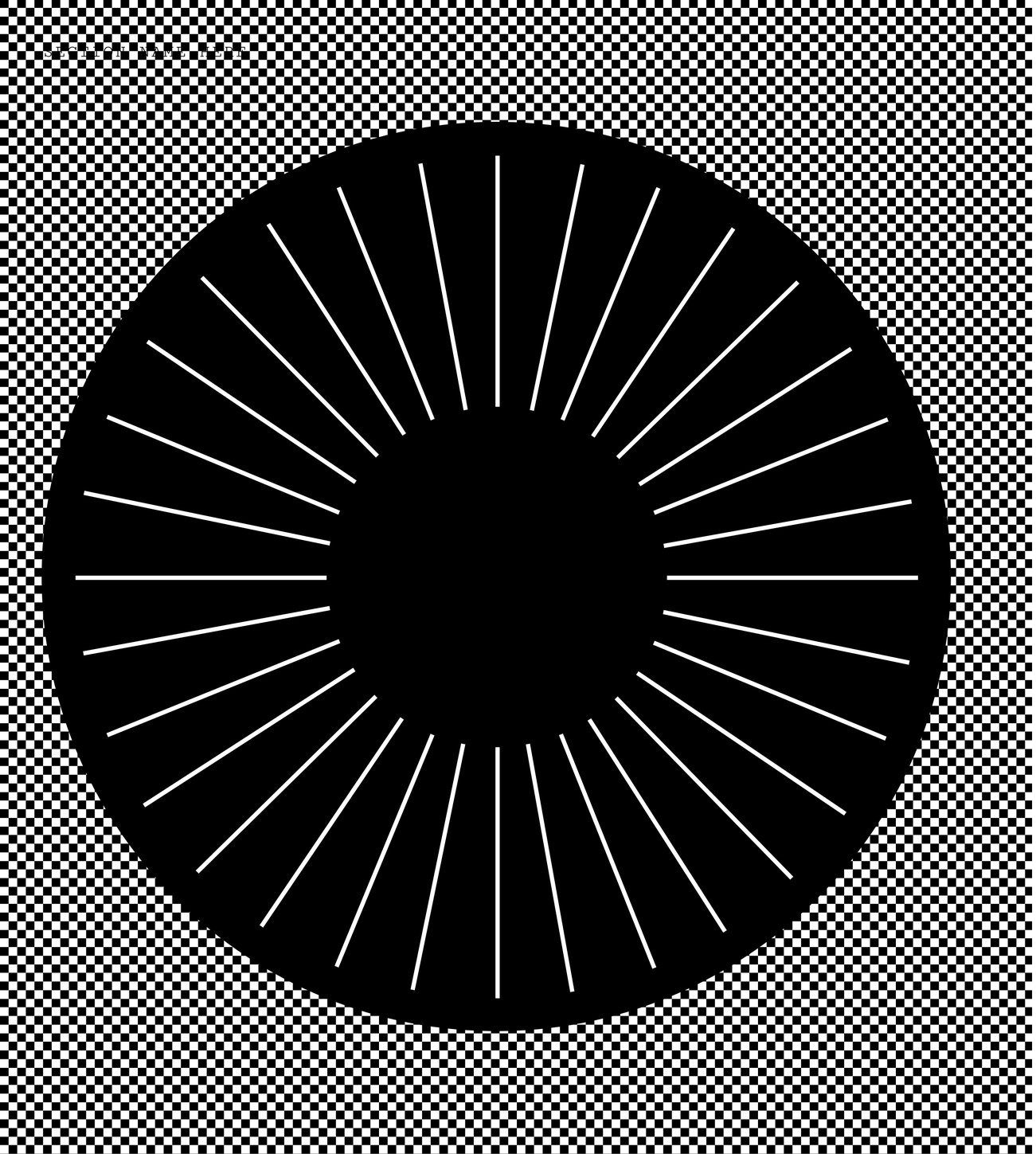
SUMMATIVE ASSESSMENT



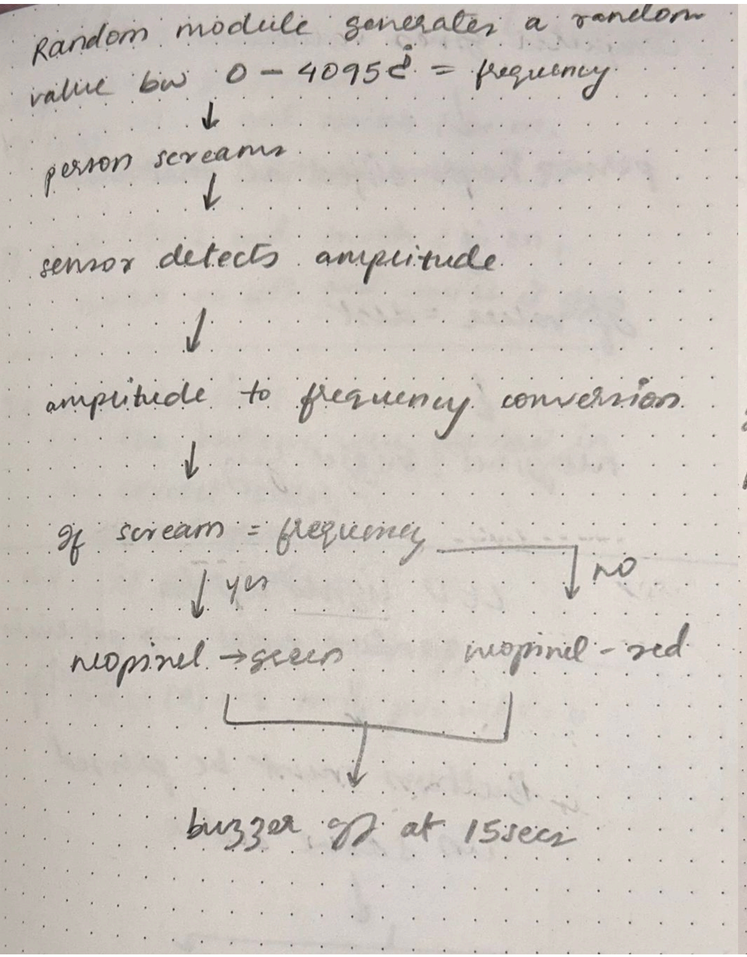
OPEN DESIGN TECHNOLOGY

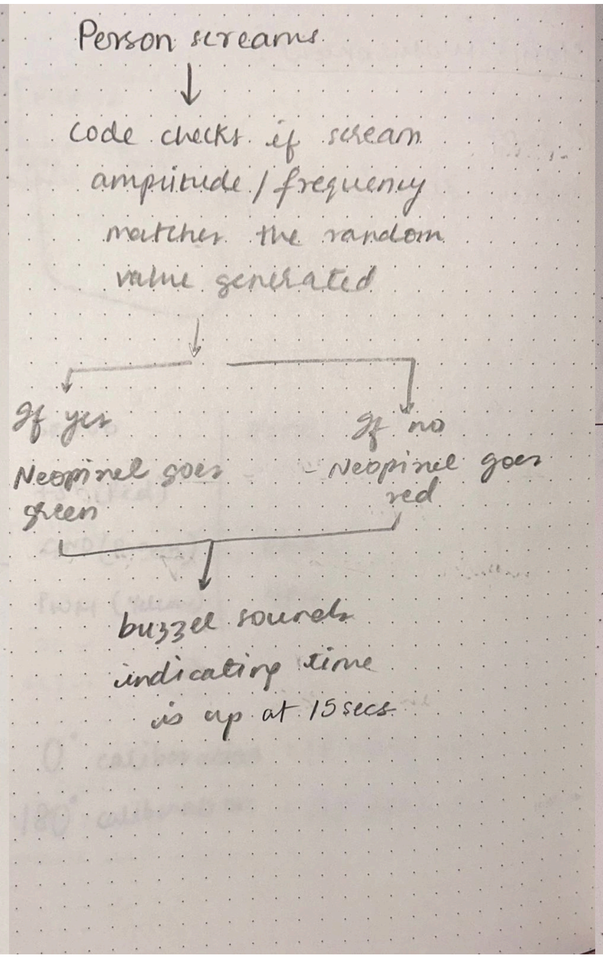
AARUSHI PURI

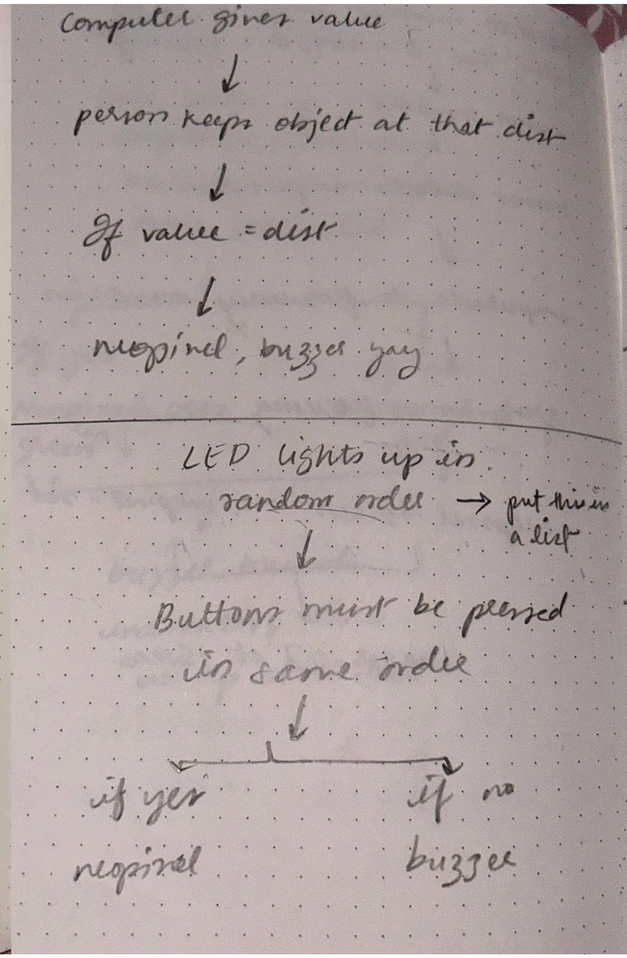


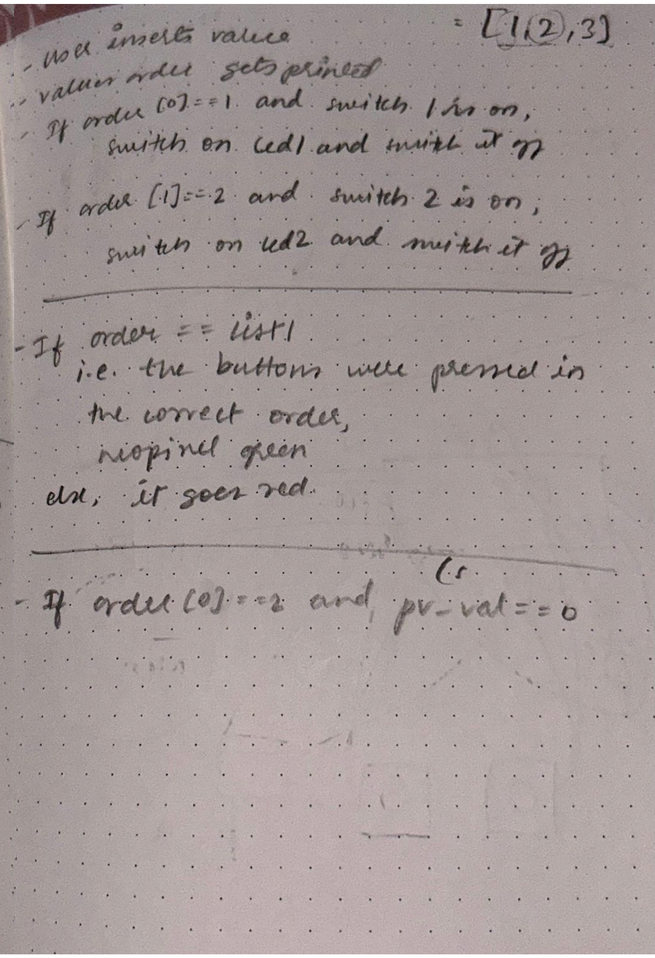
IDEATION TO PROTOTYPE

FLOWCHARTS

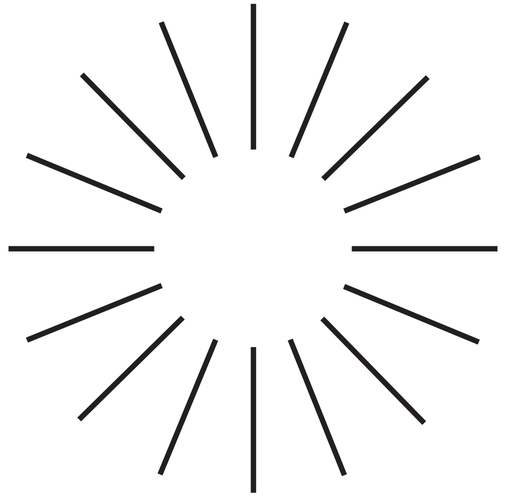

Random module generates a random value bw 0 - 4095 = frequency
person screams
sensor detects amplitude
amplitude to frequency conversion
if scream = frequency
yes
no
neopixel → green
neopixel - red
buzzer off at 15secs


Person screams
Code checks if scream amplitude / frequency matches the random value generated
if yes
if no
Neopixel goes green
Neopixel goes red
buzzer sounds indicating time is up at 15secs


Computer gives value
person keeps object at that dist
if value = dist
neopixel, buzzer jay
LED lights up in random order
put this in a list
Buttons must be pressed in same order
if yes
if no
neopixel
buzzer


= [1,2,3]
- User inserts value
- Values order sets pinned
- If order [0] == 1 and switch 1 is on, switch on led1 and switch it off
- If order [1] == 2 and switch 2 is on, switch on led2 and switch it off
- If order == list1 i.e. the buttons were pressed in the correct order, neopixel green
else, it goes red
- If order [0] == 2 and pr_val == 0

THE JOURNEY



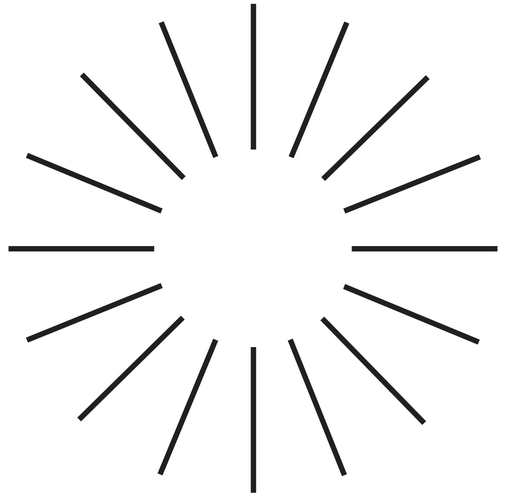
AT FIRST, WE WANTED TO MAKE A BASKETBALL GAME, THEN A FREQUENCY CHECKING GAME (THE MATH WAS TOO COMPLICATED) AND THEN FINALLY WE LANDED ON MAKING A **MEMORY GAME.**

THE DIFFICULTY IN MAKING THE FREQUENCY CHECKING CODE WAS THAT THE SOUND DETECTOR SENSOR MEASURES SOUND IN AMPLITUDE AND THEN TO CONVERT THAT TO FREQUENCY WE WOULD NEED AN ADC - ANALOG TO DIGITAL CONVERTOR WHICH WAS NOT PRESENT IN OUR SENSOR AND WE WERE NOT CONFIDENT ABOUT BEING ABLE TO USE IT WITH OUR ESP32.

WE KNEW WE WANTED TO DO MAKE A CODE THAT VALIDATES SOMETHING PROVES SOMETHING WRONG. HENCE, WE DECIDED TO MAKE A MEMORY GAME.

WITH THE HELP OF FLOWCHARTS WE EXAMINED WHAT WE NEEDED TO DO AND UNDERSTOOD THE FLOW OF OUR CODE.

THE INITIAL LOGIC



1. OUR LOGIC WAS THAT USING RANDOM MODULE, WE WOULD GENERATE A RANDOM SEQUENCE FOR THE LEDS TO LIGHT UP IN.

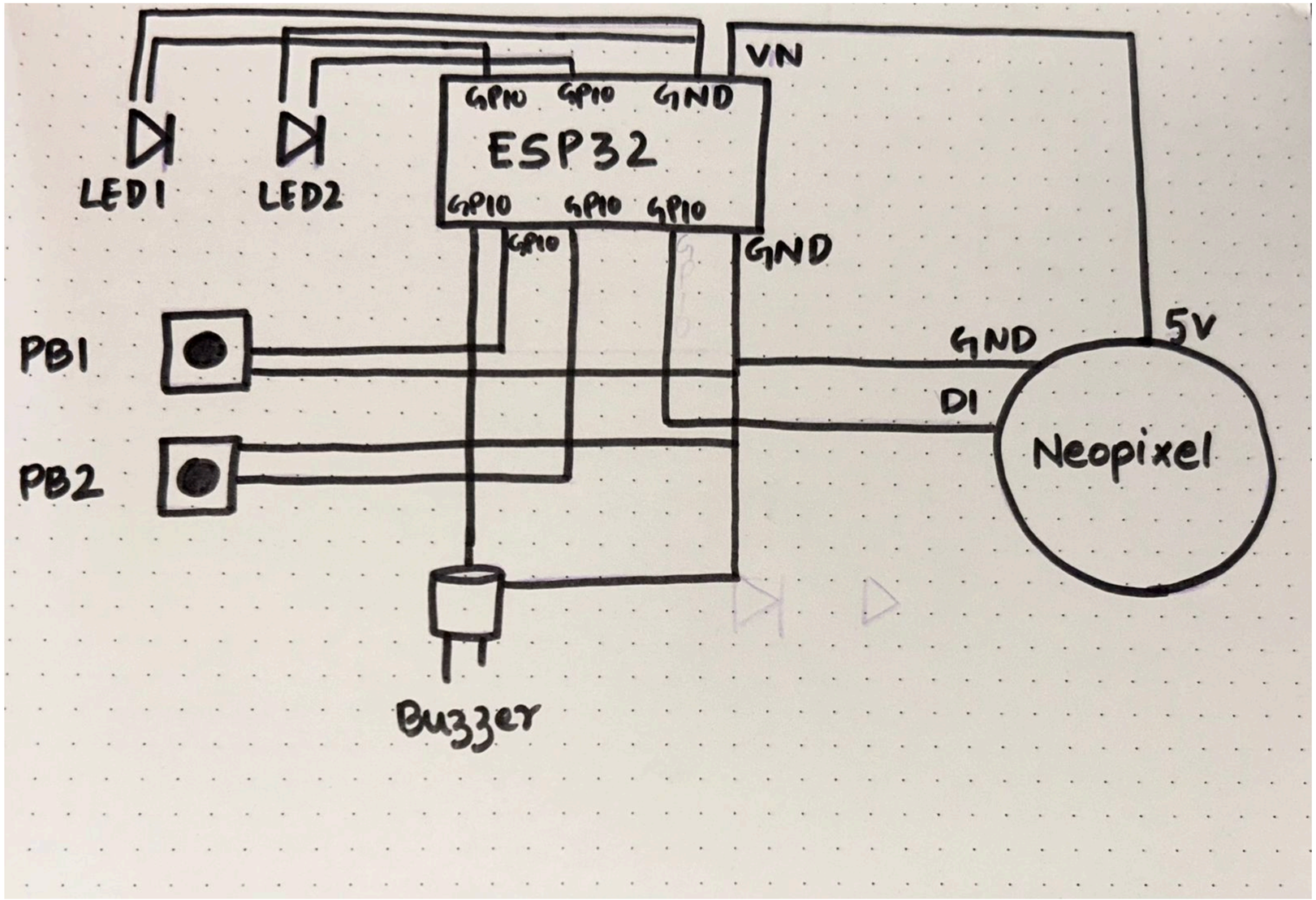
↓

2. PUSH BUTTONS WOULD BE PRESSED IN THE SAME ORDER

↓

3. NEOPIXEL LIGHTS UP AND BUZZER SOUNDS

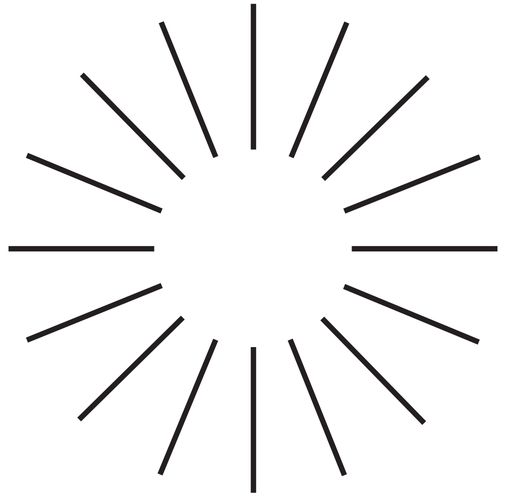
CIRCUIT DIAGRAM



THE CODE

```
from machine import Pin
import time
import random
import neopixel

pb = Pin(18, Pin.IN, Pin.PULL_UP)
p = Pin(32, Pin.IN, Pin.PULL_UP)
ir=Pin(26,Pin.In,Pin.PULL_UP)
neo=neopixel.NeoPixel(Pin(14),16)
astro = Pin(25, Pin.OUT)
astro1 = Pin(33, Pin.OUT)
buzz = Pin(22, Pin.OUT)

# led = []
# inputbutton = []
length = 3
ir_val=ir.value()
if ir_val==0:
    while True:

        led = []

        for i in range(length):
            led.append(random.choice([1,2]))

        print("sequence:", led)

        for i in range(length):
            if led[i] == 1:
                astro.on()
                time.sleep(0.4)
                astro.off()
                time.sleep(0.3)
            if led[i] == 2:
                astro1.on()
                time.sleep(0.4)
                astro1.off()
                time.sleep(0.3)

inputbutton = []

for i in range(length):

    while True:


        if pb.value() == 0:
            inputbutton.append(1)
            astro.on()
            time.sleep(0.2)
            astro.off()
            time.sleep(0.2)
            print("Pressed: 1")
            time.sleep(0.3)
            break

        if p.value() == 0:
            inputbutton.append(2)
            astro1.on()
            time.sleep(0.2)
            astro1.off()
            time.sleep(0.2)
            print("Pressed: 2")
            time.sleep(0.3)
            break

if inputbutton == led:
    print("Correct")
    time.sleep(0.1)
    buzz.on()
    time.sleep(0.1)
    buzz.off()
    time.sleep(0.1)
    buzz.on()
    time.sleep(0.1)
    buzz.off()
    for i in range(0,16):
        neo[i]=(0,255,0)
        neo.write()
        time.sleep(0.1)
    for i in range(0,16):
        neo[i]=(0,0,0)
    neo.write()
    time.sleep(0.7)
#             neo[i]=(0,0,0)
#             neo.write()
#             time.sleep(0.3)
else:
    print("Wrong")
    buzz.on()
    time.sleep(0.5)
    buzz.off()
    time.sleep(0.1)
    for i in range(0,16):
        neo[i]=(255,0,0)
    neo.write()
    time.sleep(0.7)
    for i in range(0,16):
        neo[i]=(0,0,0)
    neo.write()
    time.sleep(0.7)
#             neo[i]=(0,0,0)
#             neo.write()
#             time.sleep(0.3)

time.sleep(2)
```

LEARNINGS AND PAIN POINTS



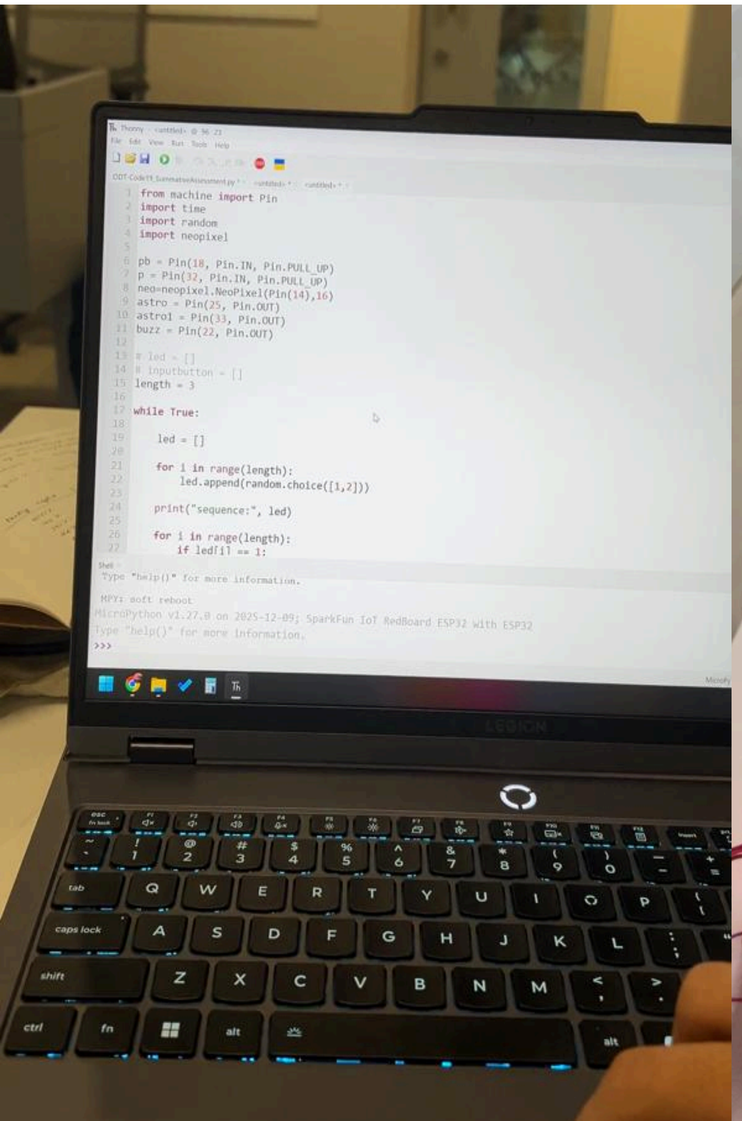
PAIN POINTS :

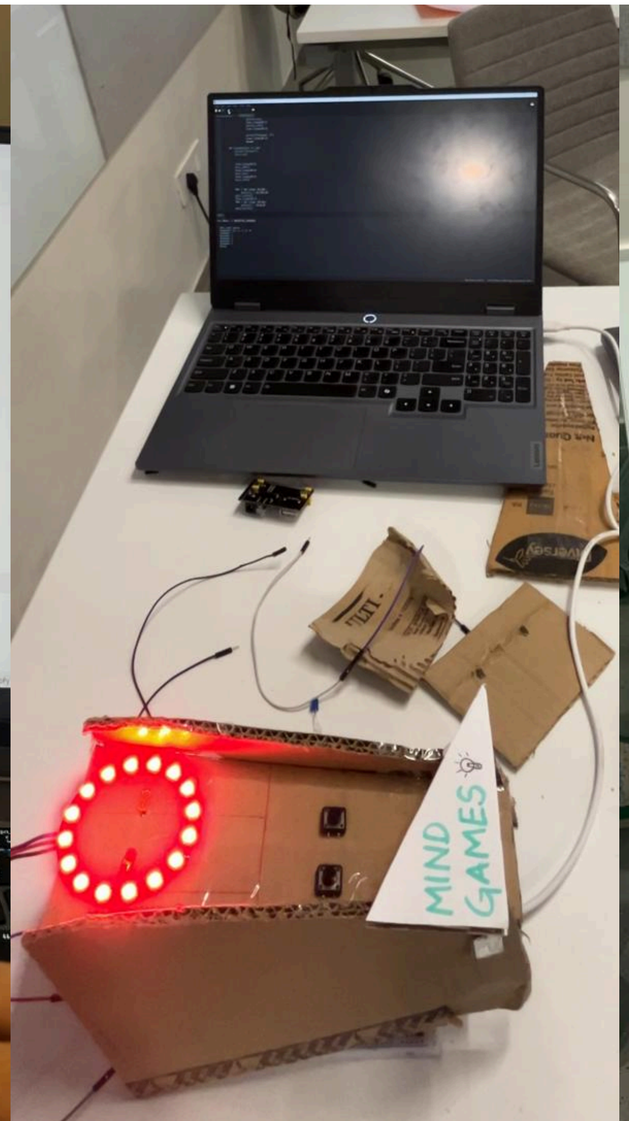
- OUR COMPONENTS STARTED MALFUNCTIONING AT DIFFERENT POINTS IN TIME.
- IN TERMS OF OUR CODE, WE/I HAD TO ITERATE AND UNDERSTAND HOW TO USE MULTIPLE LISTS TO COMPARE DIFFERENT SETS OF DATA FROM DIFFERENT COMPONENTS.
- I.E: IT WAS A LEARNING EXPERIENCE TO UNDERSTAND THAT WE NEEDED TWO DIFFERENT ‘WHILE TRUE’ LOOPS WITH TWO SEPARATE EMPTY LISTS TO COMPARE DATA FROM THE RANDOM LED SEQUENCE AND THE PUSH BUTTONS.

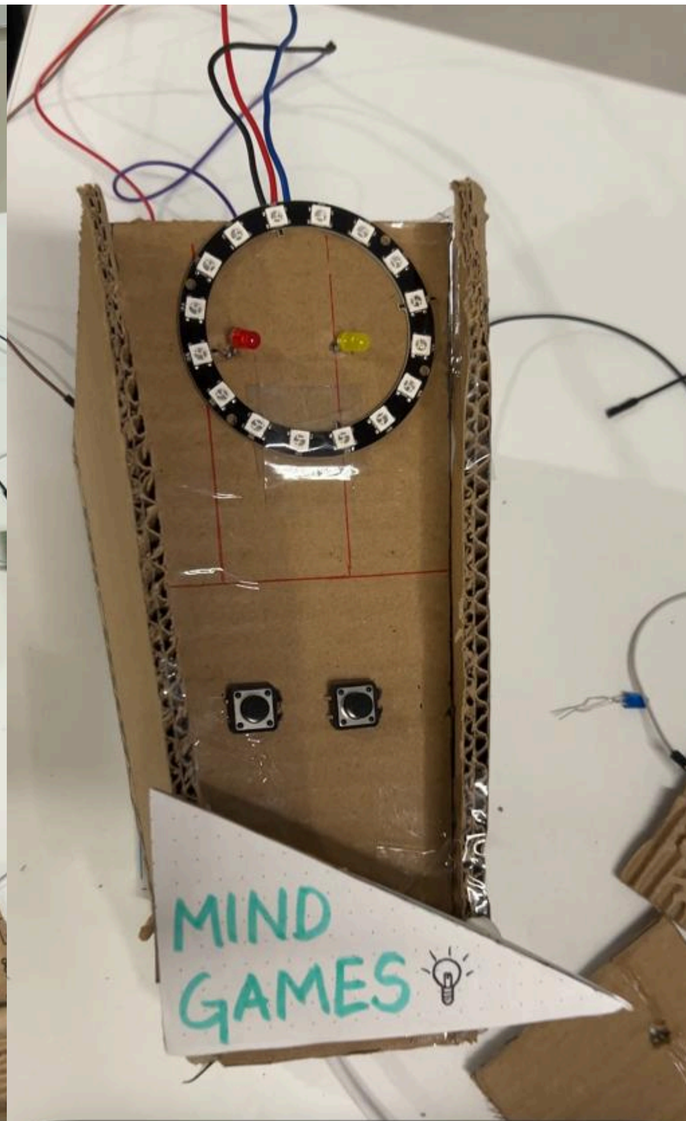
LEARNINGS :

- I LEARNT HOW TO COMBINE ALL THESE COMPONENTS
- I LEARNT HOW IMPORTANT “TIME.SLEEP()” IS BECAUSE THAT WAS THE REASON OUR BULBS WERE NOT LIGHTING UP MORE OFTEN THAN NOT.
- I ALSO LEARNT HOW TO POWER THE NEOPIXEL FROM OUR LAPTOPS BECAUSE OUR EXTERNAL POWER SOURCE STOPPED WORKING PROPERLY.

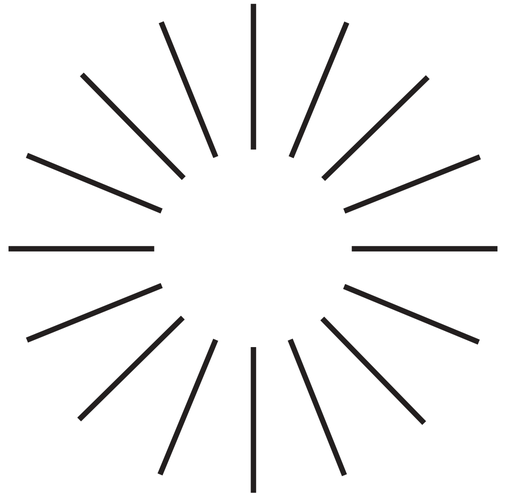
PROCESS DOCUMENTATION





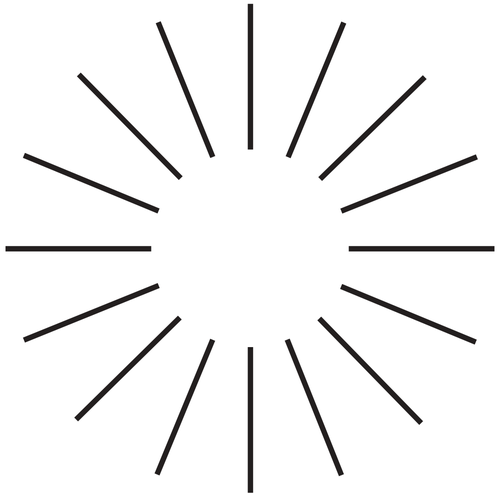


INDIVIDUAL CONTRIBUTION



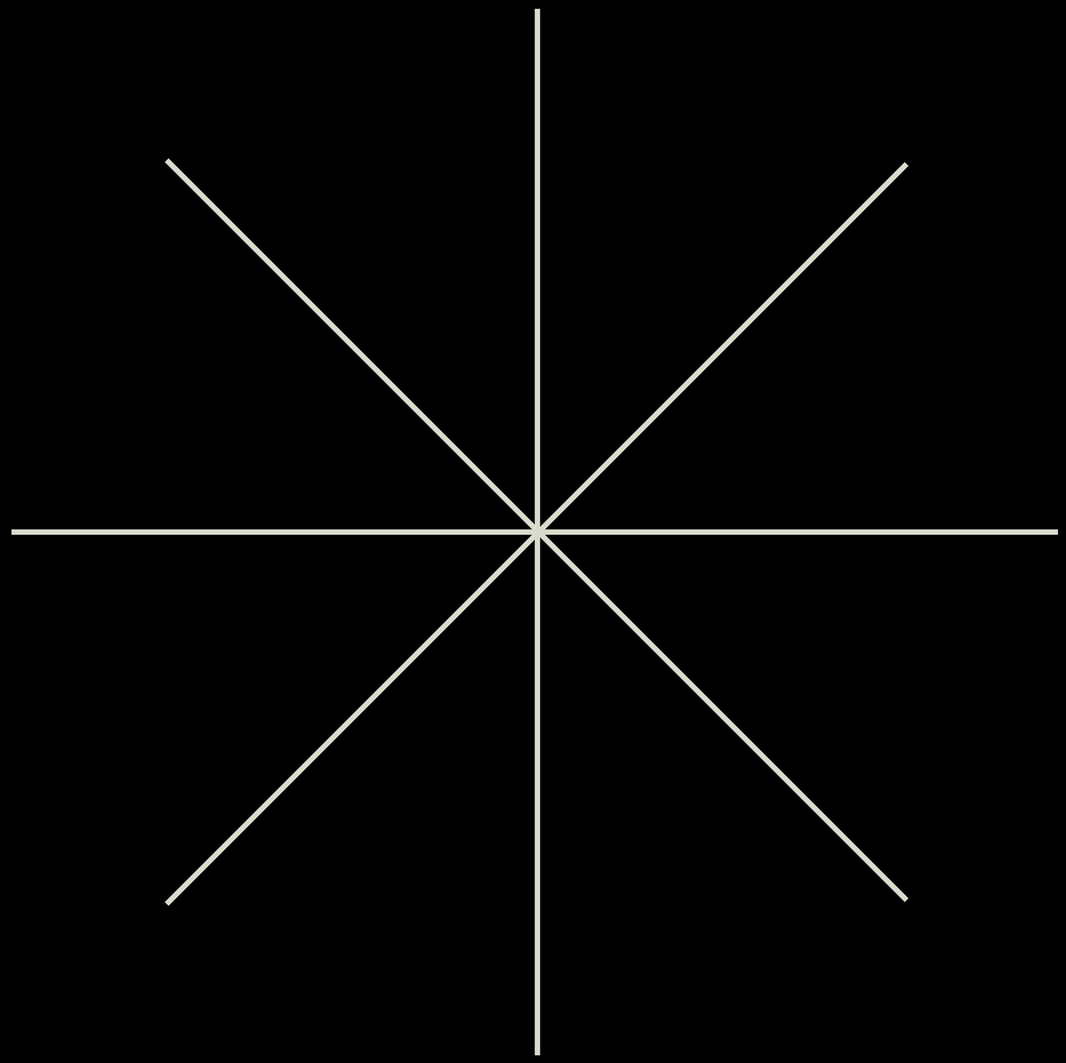
- LOGIC OF THE CODE
- CODING THE BASIC FRAMEWORK
- RANDOM MODULE INITIATION
- COMPARISON OF DIFFERENT DATA USING TWO LISTS
- INITIATION OF LEDS
- INTEGRATING PUSH BUTTONS
- COMBINING CODES AND STREAMLINING FLOW OF CODE

ADDITIONAL INFORMATION



I REALLY LIKED DOING THIS ASSIGNMENT TODAY NOT ONLY BECAUSE IT WAS ABOUT MAKING A GAME BUT BECAUSE IT WAS SUCH A LEARNING EXPERIENCE. I'M REALLY GRATEFUL FOR NAYAN, HIS TEACHING TECHNIQUES AND PATIENCE. IT HAS BEEN A GREAT LEARNING ENVIRONMENT.
I PARTICULARLY ENJOY OPEN DESIGN TECHNOLOGY BECAUSE ITT CHALLENGES ME TO THINK DIFFERENTLY AND ALWAYS FIND AN EASIER WAY TO DO THINGS AND ALSO BECAUSE I GET TO DO IT WITH MY FRIEND.

SUMMATIVE ASSESSMENT



THANKYOU

AARUSHI PURI