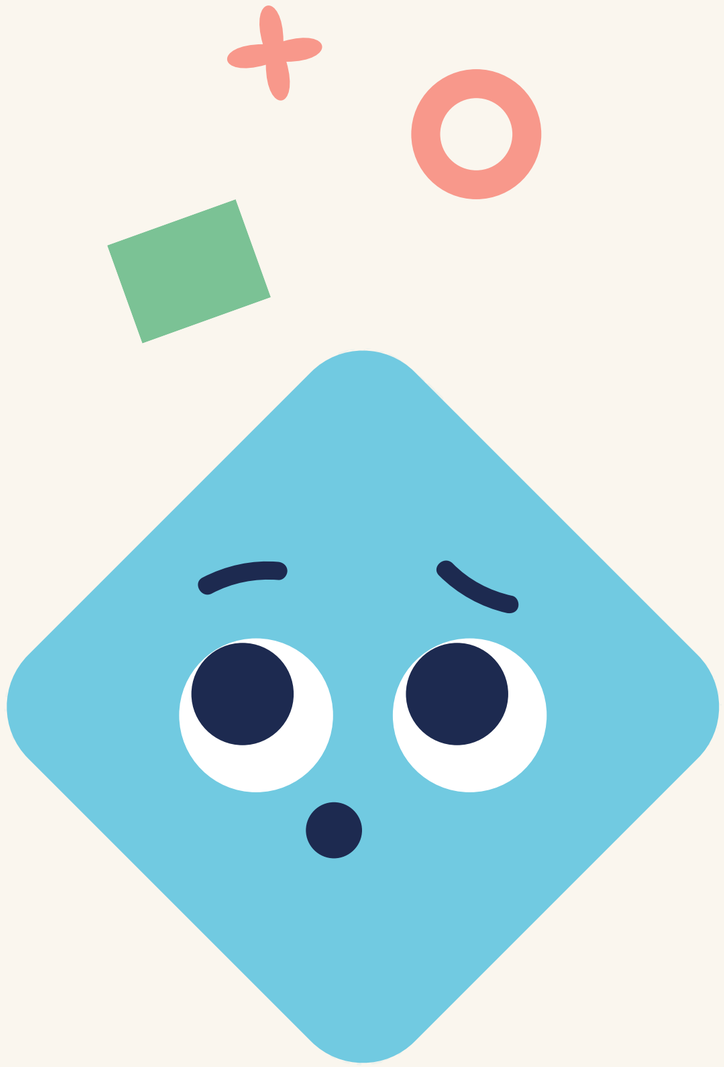
Let's Play!

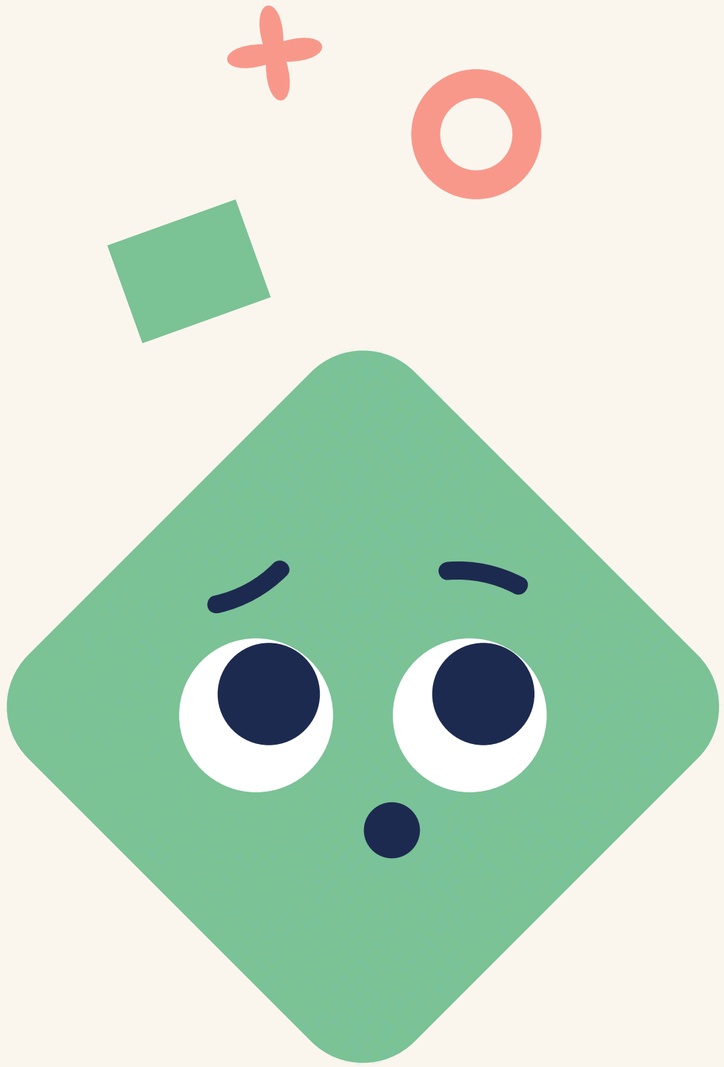
THE CLICKER

Varanasi Manasa & Sana Jain



Why This?

This project was inspired by a mobile reflex game I play a lot. It is one of those simple games where the rules are easy, but the pressure builds up quickly. You just have to react at the right moment, and if you hesitate or rush, you lose. I really enjoy that kind of tension where your focus is the only thing that matters.
While playing it, I started thinking about how interesting it would be to bring that same feeling into the real world. Instead of tapping a screen, I wanted players to physically react to something happening in front of them. Using sensors and buttons makes it more intense because your whole body is involved, not just your thumb.
The life system and scoring came from the way mobile games keep you hooked. One mistake is fine, but repeated mistakes have consequences. That creates pressure without making it unfair. I wanted to recreate that balance so the game feels competitive but still fun.
This project is basically my way of turning a simple mobile reflex game into a physical interactive experience using hardware and embedded systems. It takes something I casually play on my phone and rebuilds it into something tangible and competitive.

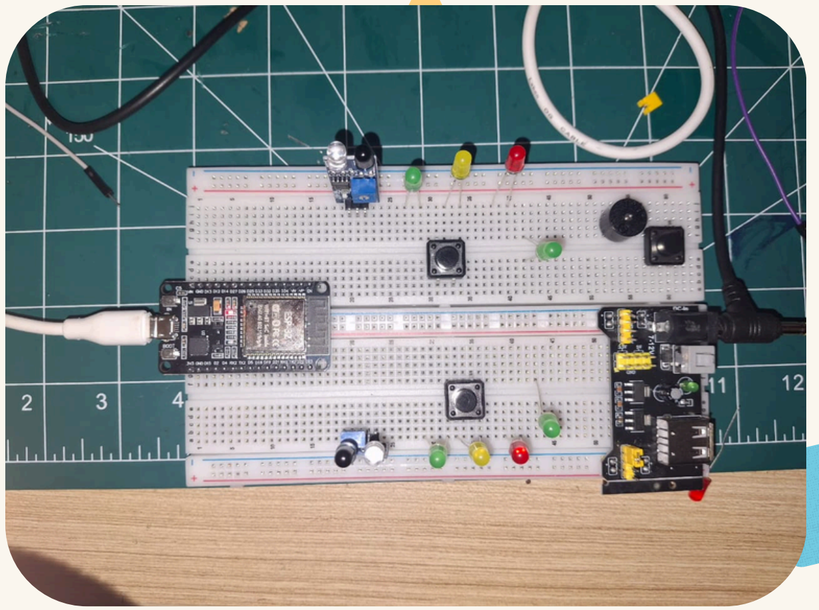


How to Play?

- Press the Start button to begin the game.
- When the ball crosses the first sensor, the round is activated.
- When it crosses the second sensor, the reaction time starts.
- Both players must press their button as fast as possible.
- You have only 1 second to press.
- The first player to press within 1 second gets 1 point.
- The LED strip shows the score for both players.
- If no one presses in time, no point is given.
- The first player to reach 8 points wins.
- Click the button to reset!

Technical Feasibility

The project was designed using components that are easily available and compatible with ESP32.
All sensors and buttons operate on simple digital input signals, reducing system complexity.
The reaction time (1 second) was implemented using non-blocking timing logic to ensure accurate performance.
Power requirements were kept minimal so the entire system could run safely on a standard 5V setup.



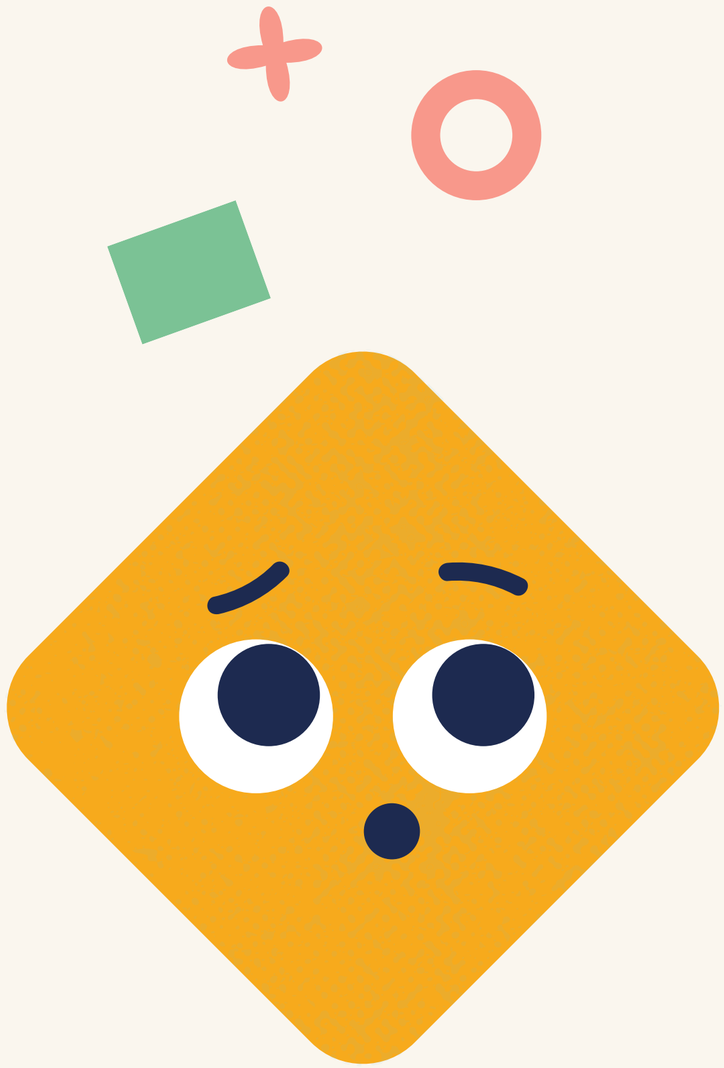
Our Components Used

Round 1

Let’s Go!

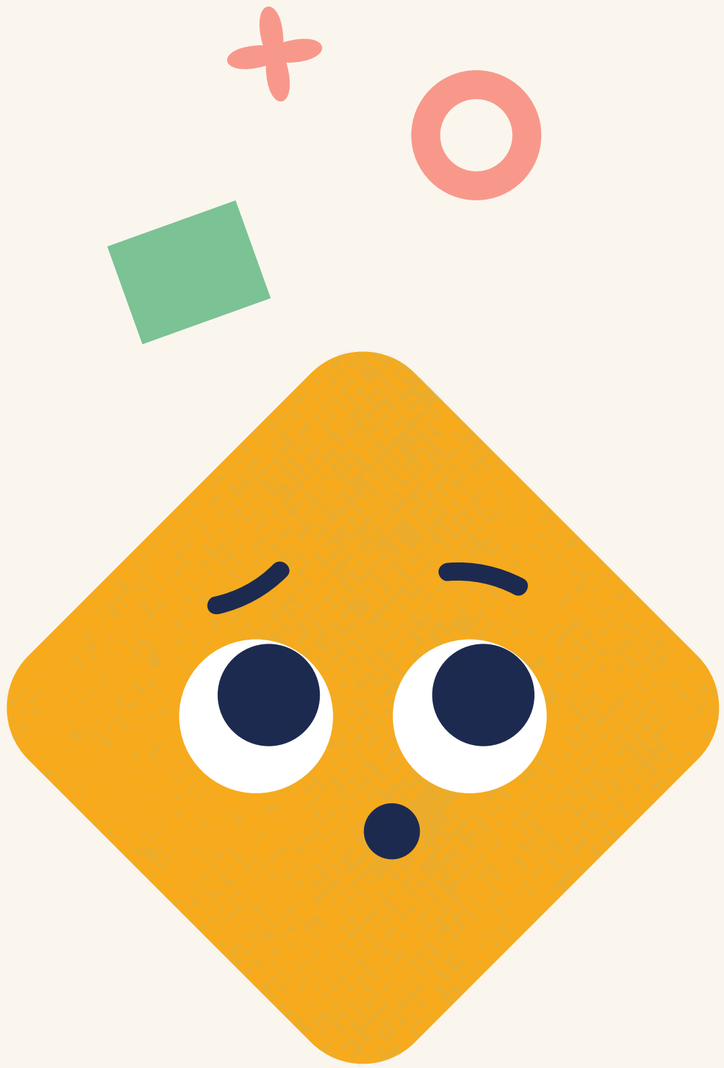
Selection of Components & Connection

Component	Purpose	GPIO Pin	Mode
Start Button	Start / Reset Game	13	Input (Pull-Up)
Player 1 Button	Player 1 Input	14	Input (Pull-Up)
Player 2 Button	Player 2 Input	27	Input (Pull-Up)
IR Sensor 1	Round Activation	35	Input
IR Sensor 2	Reaction Trigger	32	Input
NeoPixel Strip (16 LEDs)	Scoreboard Display	4	Output
Player 1 Life LED 1	Life Indicator	26	Output
Player 1 Life LED 2	Life Indicator	15	Output
Player 1 Life LED 3	Life Indicator	5	Output
Player 2 Life LED 1	Life Indicator	21	Output
Player 2 Life LED 2	Life Indicator	18	Output
Player 2 Life LED 3	Life Indicator	25	Output
Player 1 Indicator LED	Reaction Indicator	22	Output
Player 2 Indicator LED	Reaction Indicator	23	Output



Logic?

- The game works step by step, like following instructions in order.
- First, it waits for the ball to cross the first sensor.
- Then it waits for the ball to cross the second sensor.
- When that happens, a 1-second timer starts.
- During that 1 second, both players can press their buttons.
- The first player to press within that time gets a point.
- If no one presses in time, both players lose one life.
- The LED strip updates to show the new score.
- Life LEDs turn on to show how many chances are lost.
- If a player loses all 3 lives, they are out of the game.
- The Start button resets everything back to zero.

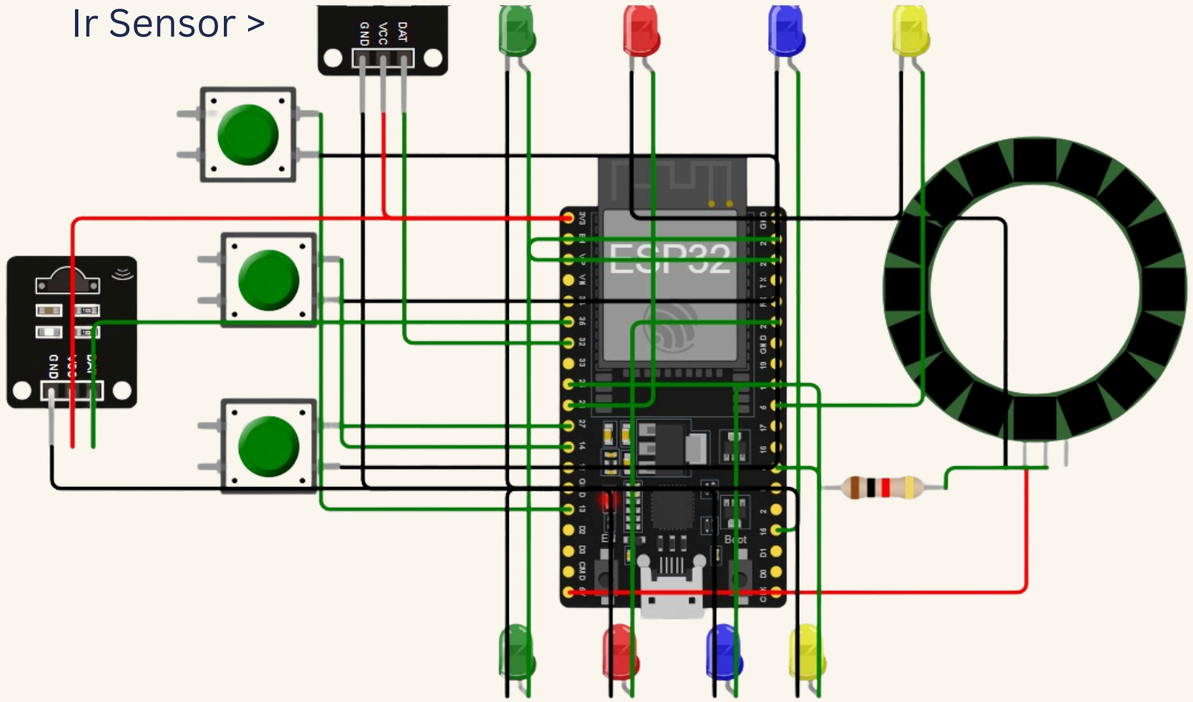


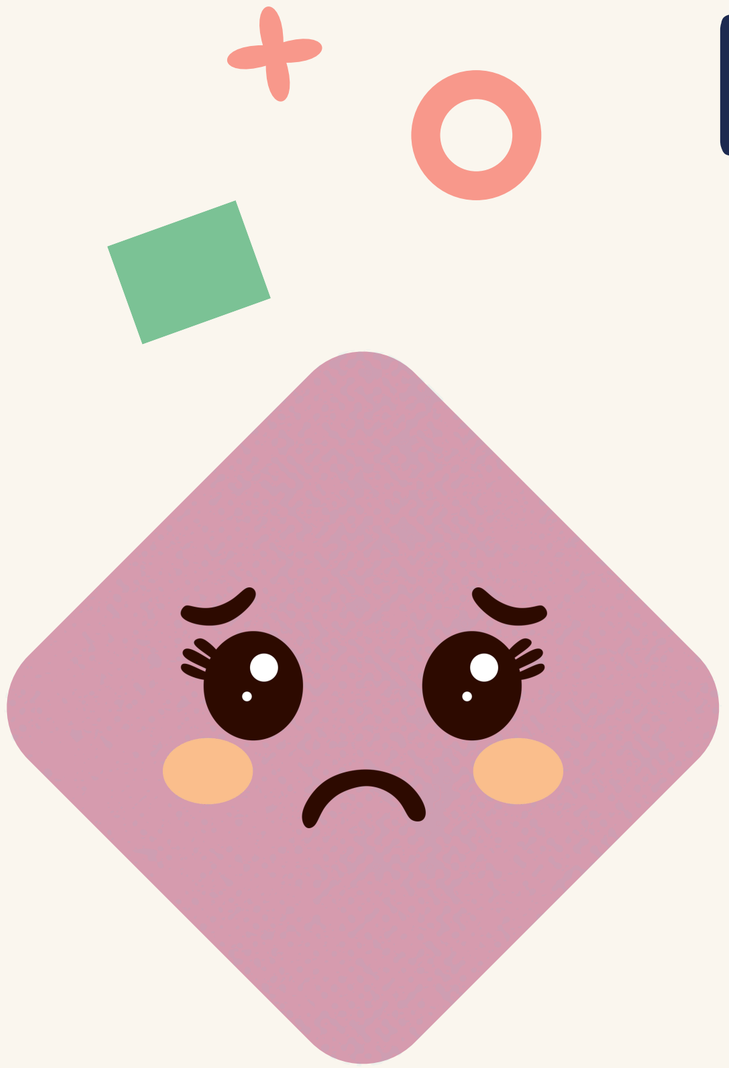
Coding Logic?

1. State-Based Flow
The game was structured as a sequence of stages:
Idle
Round Activated
Reaction Window
Score Update
This approach ensures the program runs in an organized and predictable manner.
2. Non-Blocking Timing
time.ticks_ms() was used instead of sleep() to:
Maintain real-time responsiveness
Prevent the system from freezing
Ensure accurate and fair reaction measurement
This allows the system to monitor inputs continuously during the timer.
3. First-Press Detection Logic
During the 1-second reaction window, both buttons are continuously monitored.
The first valid press increases the corresponding player's score.
Additional presses are ignored until the next round begins.
This ensures fairness and prevents multiple score increments in a single round.
4. Reset Logic
A single start button resets all game variables.
The scoreboard is cleared.
The game state is reinitialized for a fresh start.

Circuit Diagram

Ir Sensor >





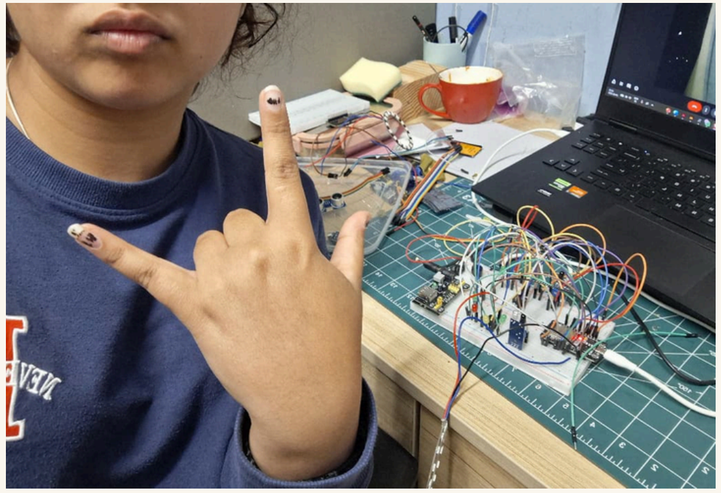
Pain Points & Error

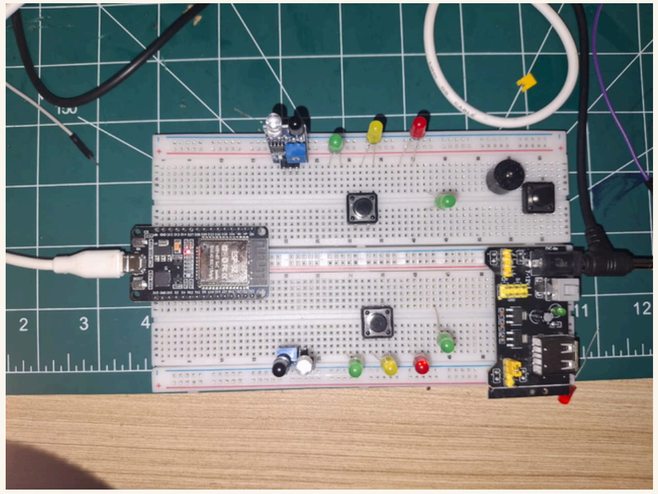
Learnings

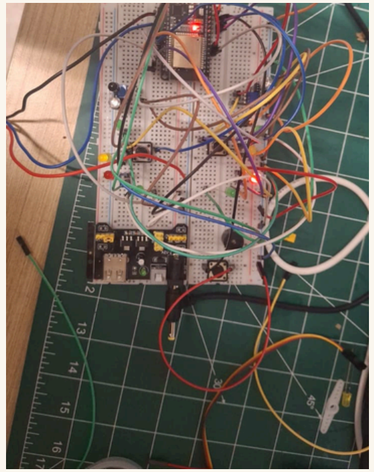
- The circuit had multiple loose connections during hardware setup.
- Due to the number of components and jumper wires, the system did not work consistently.
- Troubleshooting wiring errors took significant time.
- To ensure proper demonstration and logic validation, a simulation was implemented.
- The simulation helped verify that the coding logic was correct even when the physical circuit faced instability.

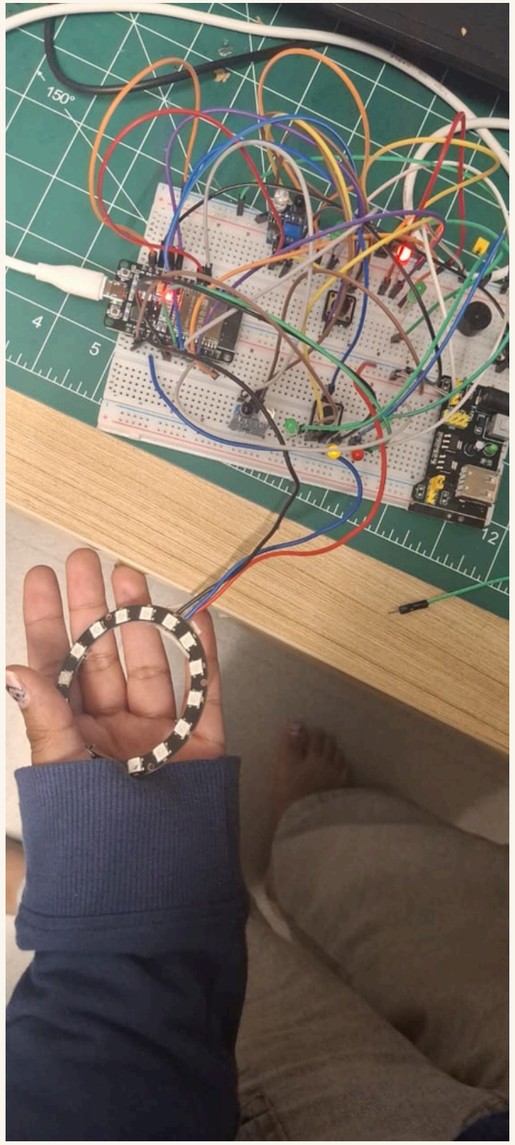
Personal Contributions

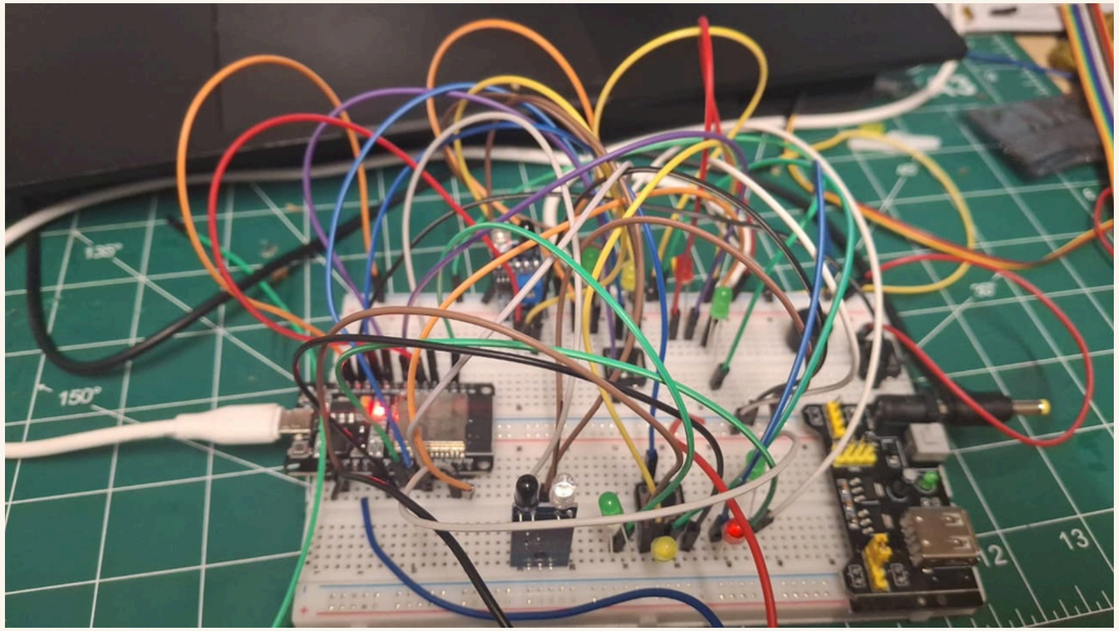
You Got This!











<untitled> * | CODEEE.py * | <untitled> * | <untitled> * | <untitled> * | <untitled> * | <untitled> *

```
1  from machine import Pin
2  import neopixel
3  import time
4
5  # -------- BUTTONS --------
6  start_btn = Pin(13, Pin.IN, Pin.PULL_UP)
7  p1_btn = Pin(14, Pin.IN, Pin.PULL_UP)
8  p2_btn = Pin(27, Pin.IN, Pin.PULL_UP)
9
10 # -------- IR SENSORS --------
11 ir1 = Pin(35, Pin.IN)
12 ir2 = Pin(32, Pin.IN)
13
14 # -------- NEOPIXEL --------
15 pixels = neopixel.NeoPixel(Pin(4), 16)
16
17 # -------- PLAYER 1 LIFE LEDS --------
18 p1_life1 = Pin(26, Pin.OUT)
19 p1_life2 = Pin(15, Pin.OUT)
20 p1_life3 = Pin(5, Pin.OUT)
```

Shell

```
- use Ctrl+C to interrupt current work;
- reset the device and try again;
- check connection properties;
- make sure the device has suitable MicroPython / CircuitPyt
- make sure the device is not in bootloader mode.
```

Listing Contribution

- Designed the complete game logic and system flow.
- Wrote the entire program for the project.
- Assembled and completed the full circuit connections.
- Created the circuit diagram for documentation.
- Developed a simulation model to validate functionality.
- Produced the final demonstration video of the working system.

Thank You For Playing!

See You Next Time!