Gang Gang Grizzlies

Brain Master

Franchescoh | Dominicoh | Kaiden | Jayden

Mentor: Robert

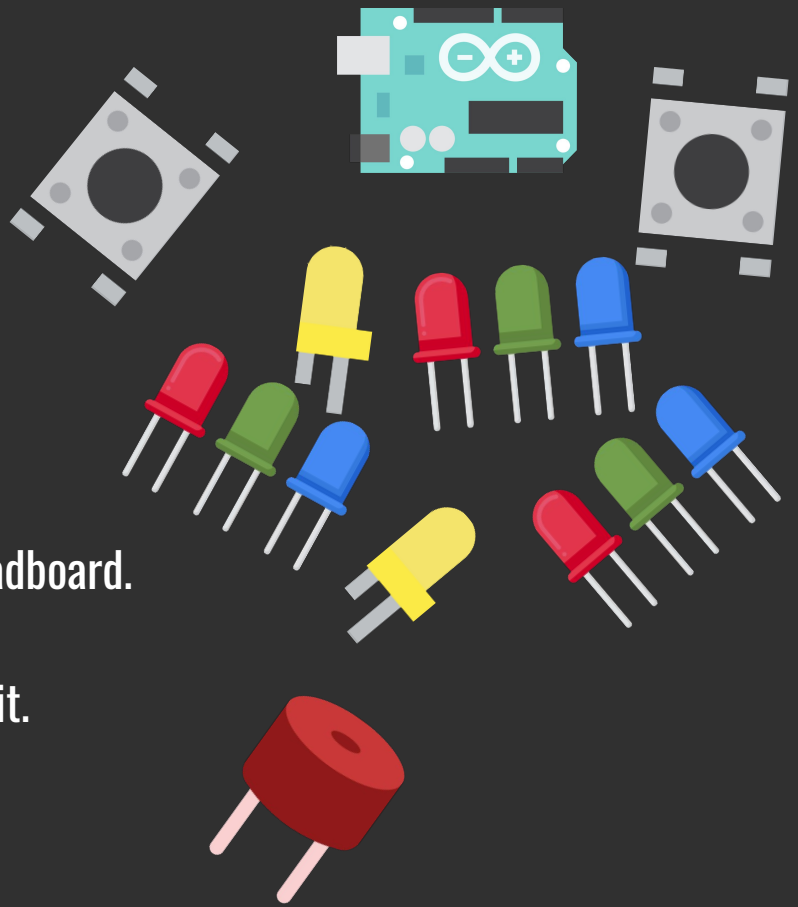
Goal

- Memory games are often used as an exercises to improve memory, concentration, and pattern recognition skills. Creating this memory game can be a way to develop and improve these abilities.
- Creating a memory game allows for innovation and creativity in game design.
- We can experiment with different features, levels, and gameplay mechanics to make their game unique and engaging.

THIS IS NOT A COPY OF SIMON Memory Game (This is just to avoid copyright issues)

Components

- FireBeetle ESP32
  - ESP32 is the brain of the game.
- 4 Small Buttons
  - Used to activate your answer.
- 4 Led Lights
  - Used to display the patterns.
- 11 Female to Female Wires
  - Used to connect the code to the Breadboard.
- 1 Breadboard
  - Lets you put all your components on it.
- 1 Buzzer
  - Used to make sounds.



Workload Division

UI Design
(Kaiden)

Random Pattern
(Jayden)

Compare User Input
and Pattern
(Franchescoh)

Read User Input
(Dominicoh)

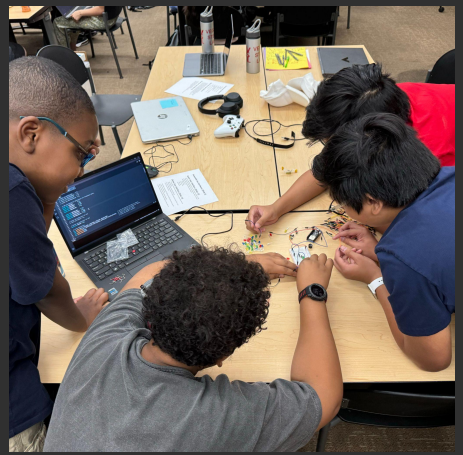
Design Steps

1. Assemble the hardware.
2. Write a function to generate and display a random pattern.
3. Write a function so that it reads player input.
4. Write a function to make it so it determines if the player has won the game.
5. Design the UI to make the game easy to play.

Assemble the Hardware

1. We connected the power and the ground from the ESP32 to the breadboard.
2. We added the LEDs and the buttons to the breadboard.
3. We grounded the LEDs and buttons to the breadboard.
4. Finally we connected the LEDs and buttons to the ESP32.



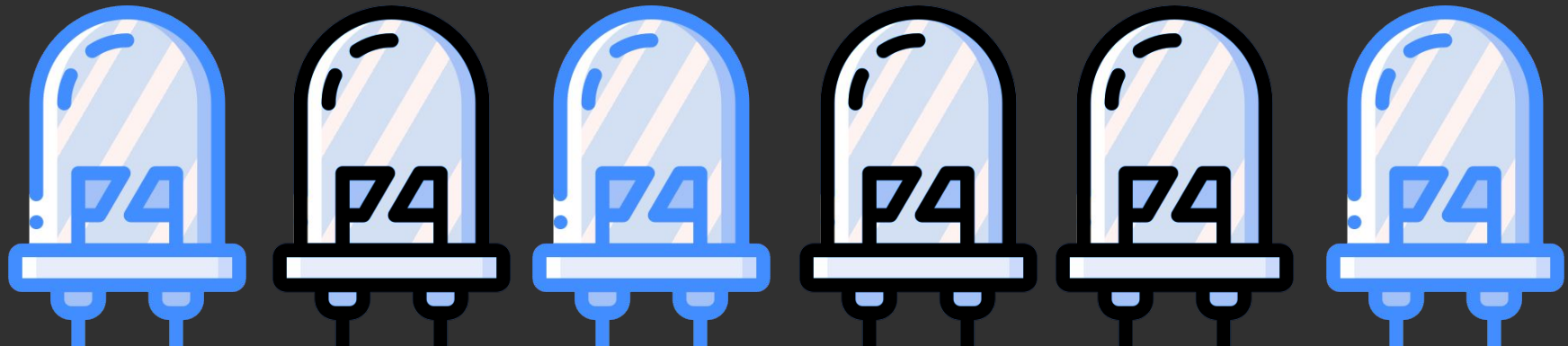




Assemble the hardware (Slide 6)
Game User Interface (Slide 13)
Game User Interface (Slide 14)

Random Pattern

- We can code the simon says board so that the electrons and array can light up a random LED. They will randomly choose to light up a random LED.

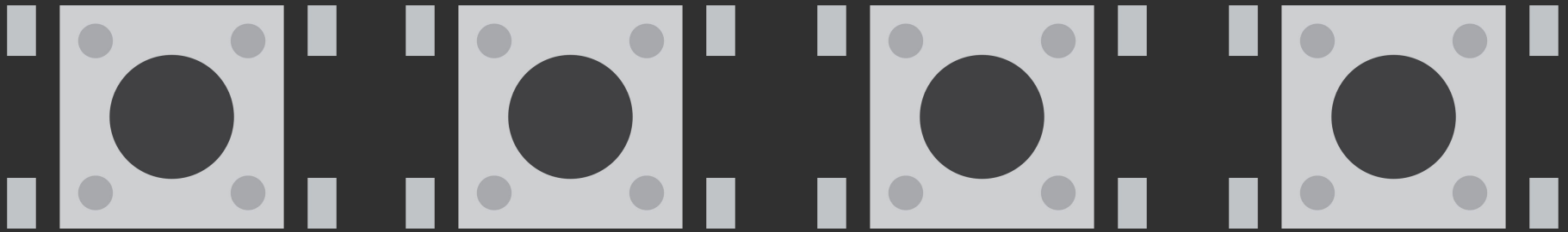


Random Pattern

```
void createPattern() {
  randomSeed(analogRead(1));
  for (int i = 0; i < gameLength; i++) {
    gamePattern[i] = random(1, 5);
  }
}
```

Read User Input

- The gamer’s input shows if they got the pattern right or wrong.
- The gamer's input directly drives the gameplay, as they must accurately mimic the sequence presented by the game.
- Correct inputs signal progress and success, while incorrect inputs indicate game over.



Read User Input

```
if (digitalRead(buttonPins[j]) == LOW) {
  digitalWrite(ledPins[j], LOW);
}
```

Compare Pattern and Input

- An array is a group of similar elements or data items of the same type collected at a contiguous memory location.
- We then compare the button pin pushed against the pin in the pattern array.
- If the values do not match we return false.

Compare Pattern and Input

```
if (j + 1 != gamePattern[inputIndex])
  digitalWrite(ledPins[j], HIGH);
  return false;
}
```

Game User Interface

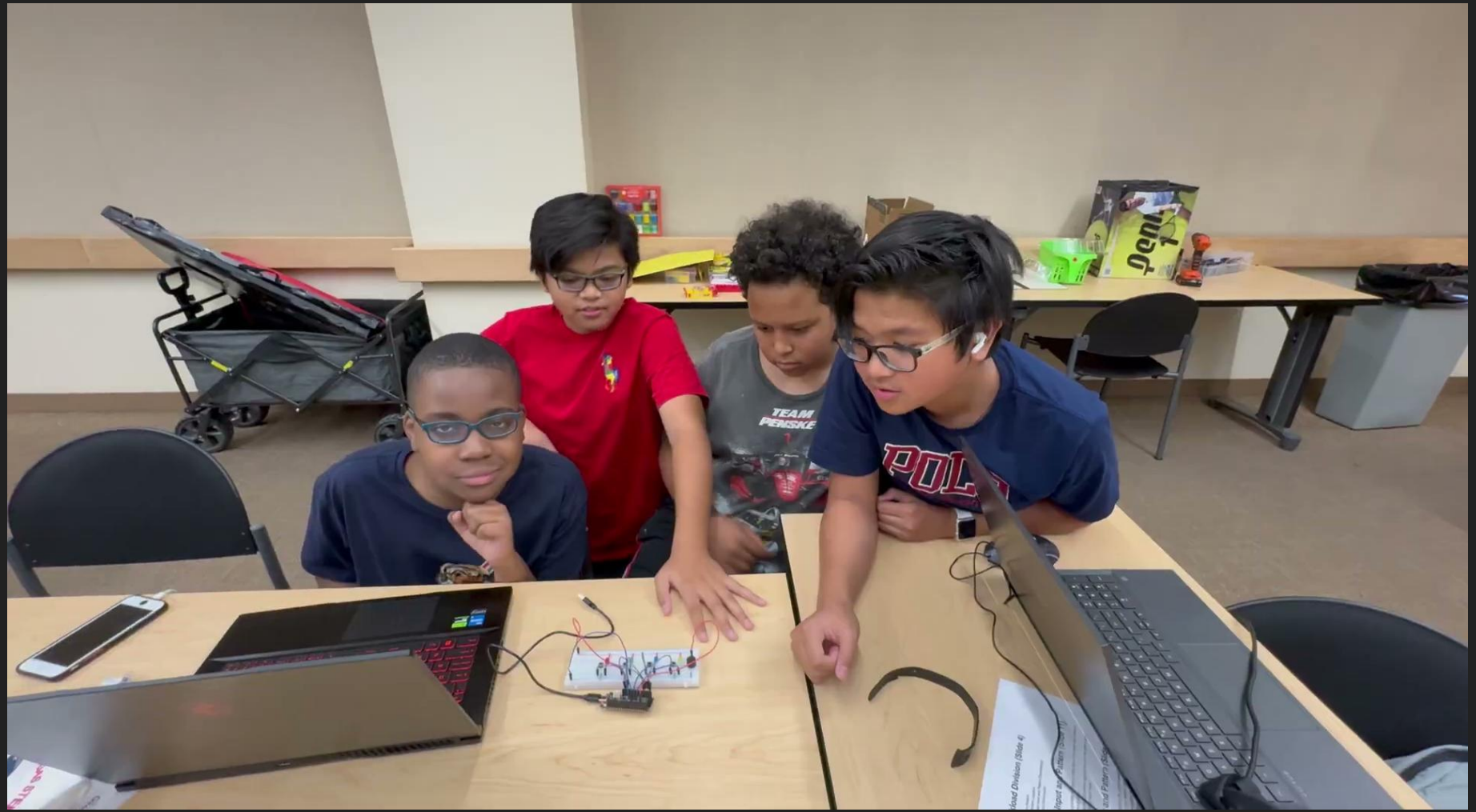
- Game user interface also known as UI is very important for games. UI helps players navigate, find information, and accomplish goals. Without UI players would be lost and would not want to play the game.
- We will create UI for:
  - Start of Game
  - Different Levels
  - Win or Lose
  - End of Game

Game User Interface

```
void startGame() {
  Serial.println("Get ready!");
  for (int i = 0; i < 4; i++) {
    digitalWrite(ledPins[i], LOW);
    delay(250);
    playTone(NOTE_C5, 250);
    digitalWrite(ledPins[i], HIGH);
  }
  delay(500);
}
```

```
void gameOver() {
  for (int i = 0; i < 4; i++) {
    digitalWrite(ledPins[i], LOW);
  }
  playTone(NOTE_A4, 200);
  playTone(NOTE_G4, 200);
  playTone(NOTE_F4, 200);
  playTone(NOTE_E4, 200);
  playTone(NOTE_D4, 200);

  delay(1000);
  for (int i = 0; i < 4; i++) {
    digitalWrite(ledPins[i], HIGH);
  }
}
```


TEAM PENSKE

Thank You