WIRE MAZE GAME USING ARDUINO

GROUP MEMBERS:

Nihar Ranjan jena – 22EC01001

Vikram Patil – 22EC01008

Aim of the experiment:

To design and implement a wire maze game that detects user contact with the maze wire and keeps track of the number of hits uisng an arduino.

Objectives:

- Develop a circuit that detects the contact of the key with maze.
- Display the total hit count of the user on LCD screen.
- If the counct exceeds the limit then the game is over.
- Alert the user with a buzzer and LED indicator upon contact.
- Provide a way for the user to reset the game and start again.

Working principle:

1. **Initialization:** Pin 3 is configured as **INPUT_PULLUP** to keep the maze wire at a HIGH state.Pin 2 is set for the **reset button**.Red and Green LEDs (pins 5 and 4), and the buzzer (pin 6) are set as outputs.The **hit count** is initialized to zero, and the **game lost flag** is set to false.

2. **Maze Wiring and Contact Detection:** One end of the maze wire is connected to pin 3, with the user operating a key to touch the maze.When the key touches the maze wire, a LOW signal is sent to pin 3.Arduino increments the **hit count**, activates the **Red LED**, and sounds the **buzzer**.

3. Game Logic: If the hit count exceeds the limit:The **game lost flag** is set."**YOU LOST**" is displayed on the LCD.A continuous buzzer tone signals the loss.

4. **Reset Mechanism:** The **reset button** on pin 2 resets the game.When pressed, a HIGH signal is sent, resetting the hit count and clearing the game lost flag.

5. Auditory and Visual Feedback: Buzzer provides feedback on contact and game loss events.**Red LED** signals hit detection, while the **Green LED** shows no contact.A **White LED** can indicate idle status when no contact is made.

6. **User Interaction and Game Play:**The **LCD** displays the **hit count** and updates in real time.The system uses **debounce delays** to ensure smooth gameplay and user interaction.

CONNECTIONS:

1. LCD with I2C Module (16x2 LCD): VCC → Arduino **5V**, **GND** → Arduino **GND**, **SDA** → Arduino **A4** (Data line for I2C communication), **SCL** → Arduino **A5** (Clock line for I2C communication)

2. Buzzer: Positive Terminal (+) → Arduino **Pin 6, Negative Terminal (-)** → **GND**

3. LEDs: Green LED: Anode (long leg) → Arduino **Pin 4, Cathode (short leg)** →**GND** (through a 220Ω resistor).

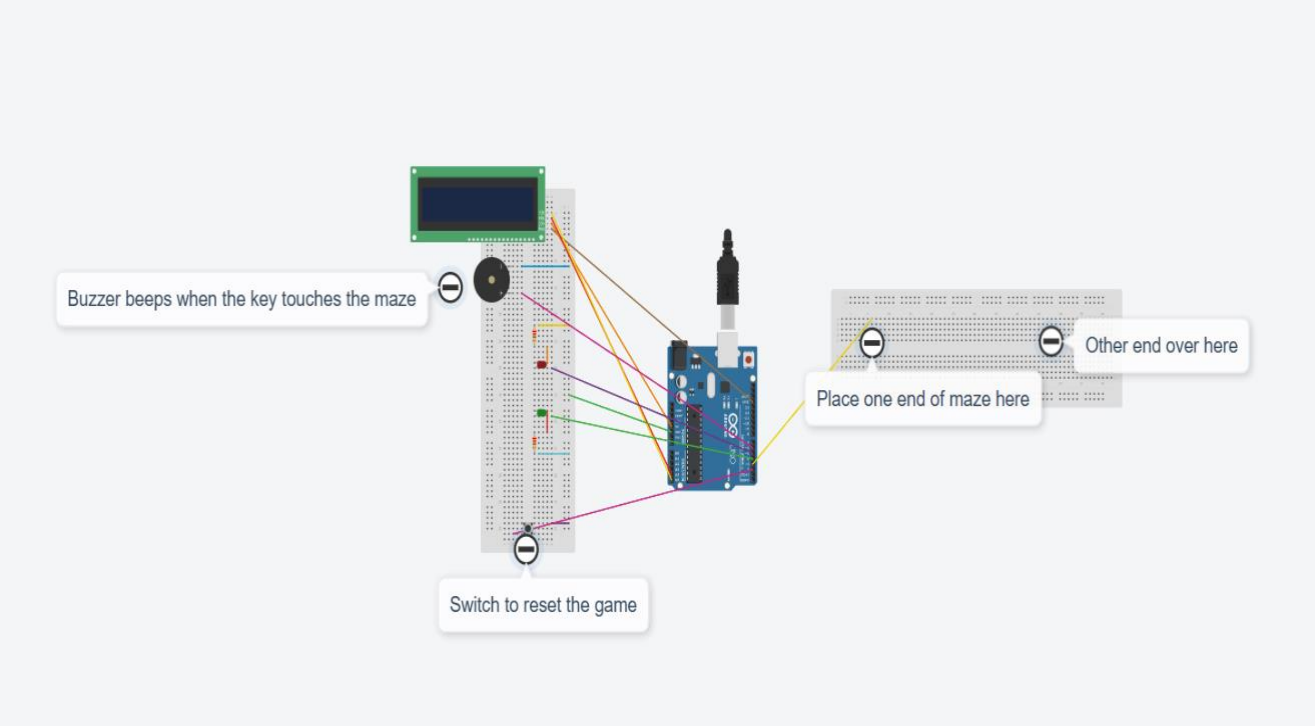
Red LED: Anode (long leg) → Arduino **Pin 5, Cathode (short leg)** → **GND** (through a 220Ω resistor).

4. Wire Maze Input (Game Input): One end of the maze wire → Arduino **Pin 3,** The other end of the maze wire left open.

Pull-up Configuration for Pin 3: Use the internal pull-up resistor by declaring pinMode (wireMazePin, INPUT_PULLUP) in the code. No external resistor is needed unless you face issues with sensitivity, in which case you can add a 10kΩ resistor between **Pin 3** and **5V**.

5. Reset Button: One terminal → Arduino **Pin 2.**, **Other terminal** → **GND**.,Configure the internal pull-up resistor in the code with pinMode(resetButtonPin, INPUT_PULLUP).

CIRCUIT DIAGRAM:



Learnings Outcomes:

** We gained practical experience with the fundamentals of serial communication and successfully implemented it during this experiment.

** We explored various Arduino IDE commands to perform specific tasks and developed a better understanding of the code flow in the Arduino environment.

** We learned about the functionality and working principles of different components, including I2C modules, LCD displays, switches, and buzzers.

** We explored the integration of visual and auditory feedback mechanisms using LEDs and a buzzer to enhance user interaction.

** We gained insights into implementing conditional logic for real-time decision-making, such as detecting hits and resetting the game state.