Digital Electronics Project

Arduino Uno Snake Game Project

❖ **Introduction:**

- **Project Overview:**

  This project aims to implement a classic Snake Game using Arduino Uno, a LED matrix, a 16x2 LCD display, a joystick, and jumper wires. The Snake Game is displayed on the LED matrix, and the player controls the snake's movement with the joystick. When the snake eats an apple, its length increases, and the score is updated on the LCD display.

- **Objectives:**

✓ To decide or develop a data structure to represent snake at a given instance.
✓ Integrate a LED matrix for displaying the game graphics.
✓ Implement joystick controls for snake movement.
✓ Display the score on a 16x2 LCD screen.
✓ Dynamically generate apples for the snake to consume.

❖ **Materials and Methods:**

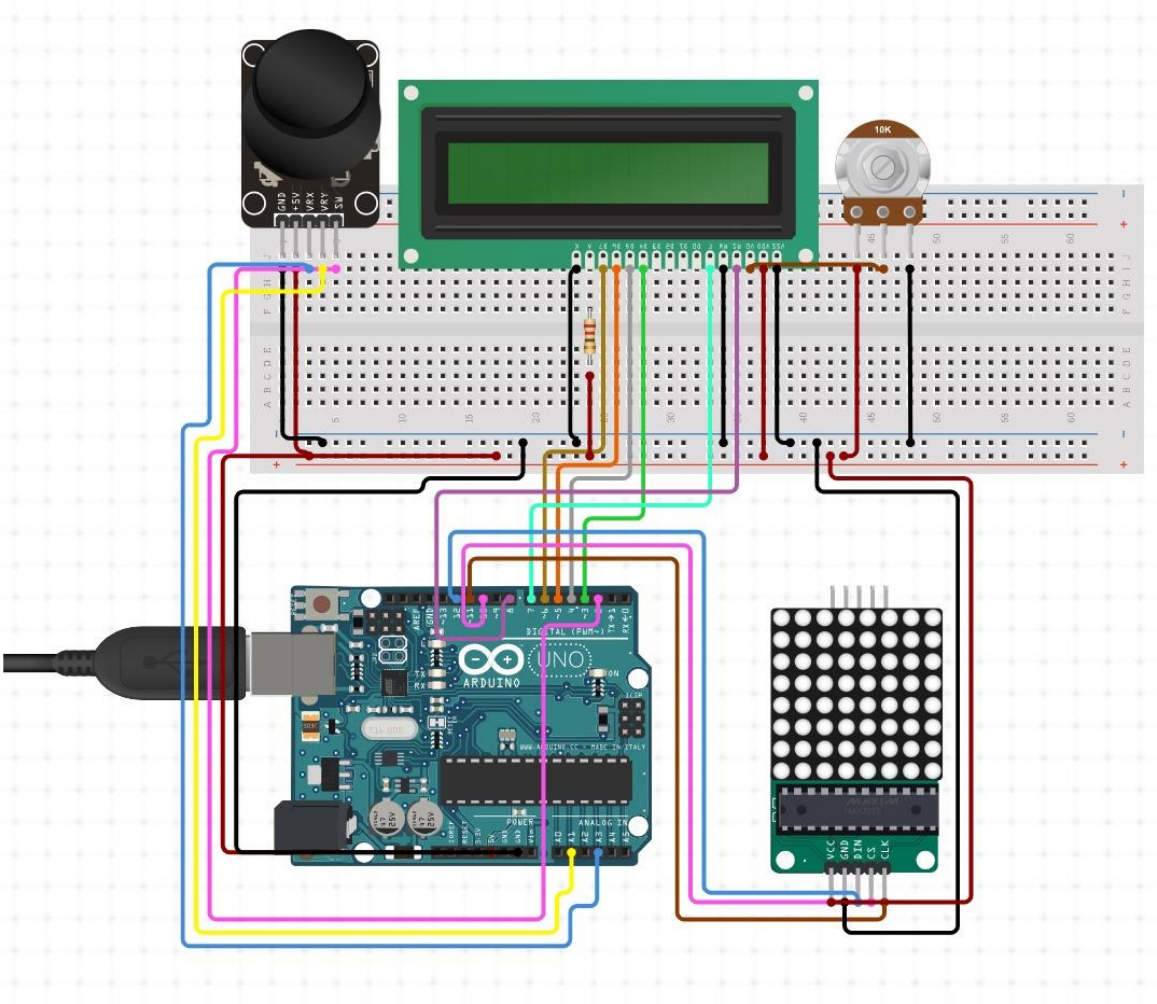
- **Hardware Components:**

1. Arduino Uno
2. LED matrix
3. 16 x 2 LCD display
4. Joystick
5. Jumper wires

- **Software:**

→ Arduino IDE for programming.
  - "LiquidCrystal.h" library for controlling LCD display.
  - "LedControl.h" library for controlling LED matrix

- **Circuit Diagram:**



❖ **Design and Implementation:**

- **Snake Game Logic**:

→ The Snake and Apple will be displayed on the LED matrix and the live score will be displayed on LCD display.

→ The joystick will act as controller for controlling the snake by providing analog inputs to Arduino board according to received data the snake will move in that direction.

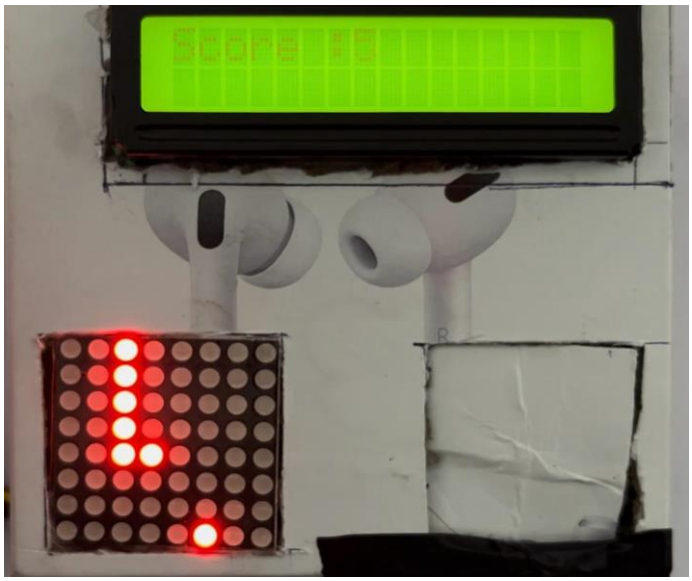
→ If snake eats the apple the score will be updated on LCD display and the length of snake will be increased.

→ If the snake is collided with its own body then game will be over.

- **User Interface:**

→ At starting of the game message will be shown that Press to Start the Game.

→ While playing the game the display will show the live score.



→ At end when the game is over it will display your final score.

- **Challenges and Solutions:**

→ **Representing Snake**:- For representing snake on LED matrix we created a LinkedList data structure which gives us functionality to store the whole body of snake in a line in form of x and y coordinates. Head of LinkedList represents head of the snake and the Tail of LinkedList represents tail of the snake

```
// Creating a Class for the Nodes of Linked List
class Node {
public:
  int x, y;
  Node *next;
  Node *prev;
  Node() {
    x = y = 0;
    next = NULL;
    prev = NULL;
  }
  Node(int x, int y) {
    this->x = x;
    this->y = y;
    this->next = NULL;
    this->prev = NULL;
  }
};
// Creating a Doubly LinkedList --> Snake
class Linkedlist {
public:
  Node *head;
  Node *tail;
```

```
  Linkedlist() {
    head = NULL;
    tail = NULL;
  }
  void add(int, int);
  void addlast(int, int);
  void remove();
};

// To add a Node in the list from front
void Linkedlist::add(int x, int y) {
  Node *new_node = new Node(x, y);
  new_node->next = head;
  new_node->prev = NULL;

  if (head != NULL) {
    head->prev = new_node;
    head = new_node;
  } else {
    head = new_node;
    tail = new_node;
  }
}

// To add a Node in the list from last
void Linkedlist::addlast(int x, int y) {
  Node *new_node = new Node(x, y);
  new_node->next = NULL;
  new_node->prev = tail;
  tail->next = new_node;
  tail = tail->next;
}

//To remove a Node
void Linkedlist::remove() {
  tail = tail->prev;
  Node *tmp = tail->next;
  tail->next = tail->next->x = tail->next->y = tail->next->next = NULL;
  free(tmp);
}
```

→ **Move the snake:-** For moving the snake we add one node or co-ordinates to start of the LinkedList or snake and will remove last co-ordinates or Node from the end of the LinkedList or snake. By this way we will able to move the snake on the LED matrix.

```
// Movement Functions
void moveDown() {
  int x = Snake->head->x;
```

```
  int y = Snake->head->y + 1;
  y %= 8;  // To avoid the points to go out of matrix
  Snake->add(x, y);
  Snake->remove();
}

void moveLeft() {
  int x = Snake->head->x + 1;
  int y = Snake->head->y;
  x %= 8;
  Snake->add(x, y);
  Snake->remove();
}

void moveRight() {
  int x = Snake->head->x + 7;
  int y = Snake->head->y;
  x %= 8;
  Snake->add(x, y);
  Snake->remove();
}

void moveUp() {
  int x = Snake->head->x;
  int y = Snake->head->y + 7;
  y %= 8;
  Snake->add(x, y);
  Snake->remove();
}
```

→ **Display the snake:-** To display the snake on LED matrix just glow the led's on LED matrix which are present at the co-ordinates that are present in LinkedList.

```
// Displaying the snake
    Node *body = Snake->head;
    while (body != NULL) {
      lc.setLed(0, body->x, body->y, true);
      body = body->next;
    }
```

→ **Taking input from joystick and controlling snake:-** For this we use the analog pins of Arduino Uno to take analog inputs from the joystick and according to input taken the code will give directionality to snake.

```
// Controlling the Joystick
void joystick() {
  int x_data = analogRead(x_pin);
```

```
  int y_data = analogRead(y_pin);

  if (x_data <= 200) {
    if (direction != 1) {
      direction = 2;
    }
  } else if (x_data >= 800) {
    if (direction != 2) {
      direction = 1;
    }
  } else if (y_data <= 200) {
    if (direction != 4) {
      direction = 3;
    }
  } else if (y_data >= 800) {
    if (direction != 3) {
      direction = 4;
    }
  }
  chooseDirection(direction);
}

void chooseDirection(int data) {
  if (data == 1) {
    moveUp();
  } else if (data == 2) {
    moveDown();
  } else if (data == 3) {
    moveLeft();
  } else {
    moveRight();
  }
}
```

→ **Generating Apple:-** For generating Apple we will use random function and will choose co-ordinates randomly. After this we will glow the apple led.

```
// Randomly Generating the Apple
  apple->x = (int)random(0, 8);
  apple->y = (int)random(0, 8);
  lc.setLed(0, apple->x, apple->y, true);
```

→ **Check whether snake had eaten the apple:-** For this we will compare the co-ordinates of the apple and the head co-ordinates of the snake. If coordinates of both are same then we will generate new apple and increase the length of snake.

```
// If Ate the Apple Increase Size and Score
```

```
void eatAppleChecker() {
  if ((Snake->head->x == apple->x) && (Snake->head->y == apple->y)) {
    lc.setLed(0, apple->x, apple->y, false);
    apple->x = (int)random(0, 8);
    apple->y = (int)random(0, 8);
    Snake->addlast(Snake->tail->x, Snake->tail->y);
    score++;
  }
}
```

→ **Check for collision:-** For checking the collision we will check that the head co-ordinates is matching with any other body co-ordinates of the snake or not. If this happens then collision is there and then game will be over.

```
// Detection of Collision
bool collision() {
  // Traverse the List and Check whether next node of Head is equal to the
body
  Node *tmp = Snake->head->next;
  int chk_x = Snake->head->x, chk_y = Snake->head->y;
  while (tmp != NULL) {
    if (tmp->x == chk_x && tmp->y == chk_y) {
      return 1;
    }
    tmp = tmp->next;
  }
  return 0;
}
```

→ **Display the Score:-** We will use a count variable to store the score and we will display the count variable value.

```
lcd.print("Score : ");
    lcd.setCursor(7, 0);
    lcd.print(score);
    lc.setLed(0, apple->x, apple->y, true);
```

❖ **Conclusion and Learnings:**

→ In conclusion, the development of the Arduino Uno Snake Game project proved to be a valuable and enlightening experience. Throughout the project, various key skills and concepts were learned and applied, contributing to a deeper understanding of both hardware and software aspects of Arduino programming.

- **Learning from Joystick Input:**

→ The implementation of joystick controls for snake movement was a crucial learning point. Understanding how to read input from the joystick not only enhanced proficiency in working with analog sensors but also provided insights into translating physical interactions into meaningful actions within the digital realm.

- **LCD and LED Matrix Integration:**

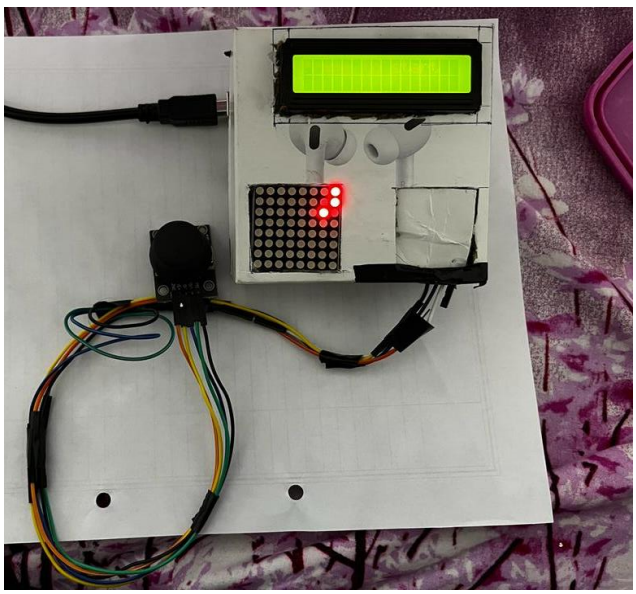
→ The integration of both a 16x2 LCD display and a LED matrix was a significant aspect of the project. This allowed for a multi-faceted user interface, where the LCD displayed essential game information such as the score, and the LED matrix provided a visually engaging representation of the Snake Game. Learning to manage two distinct display technologies concurrently was both challenging and rewarding.

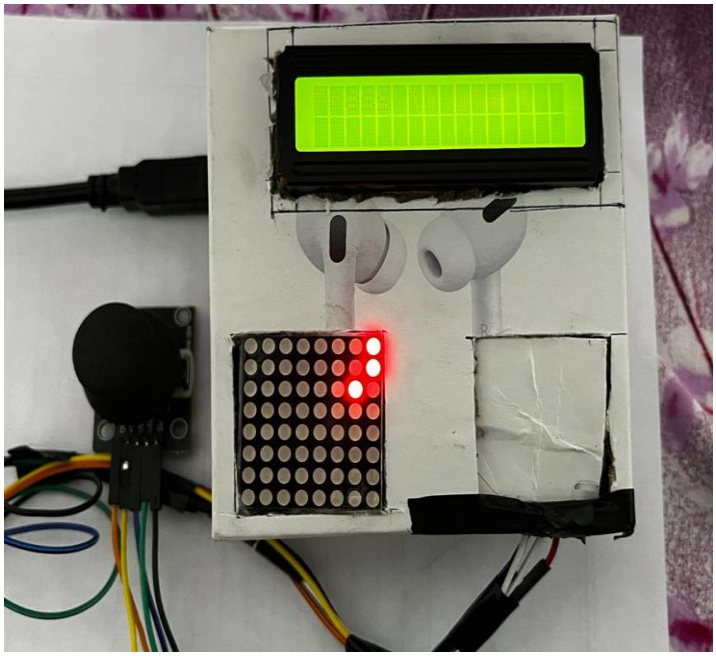
- **Troubleshooting and Error Handling:**

→ The project presented opportunities to tackle errors in code and troubleshoot hardware issues. Debugging became a critical skill as challenges arose, ranging from issues with snake movement logic to ensuring proper connections between components. These problem-solving experiences were instrumental in developing a resilient approach to programming and hardware implementation.

- **Appendices:**

→ Some photos of our project.





Thank You