**COMSATS UNIVERSITY ISLAMABAD, LAHORE CAMPUS**

**TOPIC:**

**Snake Game Using Arduino Uno**

**SUBMITTED BY:**

EMAN FATIMA (FA24-BCE-017)

JAWAD BILAL (FA24-BCE-089)

**COURSE TITLE:**

PROGRAMMING FUNDAMENTALS

**SUBMITTED TO:**

MS TABASSUM NAWAZ

**DATE OF SUBMISSION:**

12-JUNE-2025

**Introduction:**

The Snake Game is a retro-style game that tests real-time reaction and movement control. This semester project brings the Snake Game to life using an **Arduino Uno** and an **I2C LCD module**. It is controlled with four directional push buttons and rendered visually on a character-based LCD. The project demonstrates fundamental concepts of embedded systems, including digital I/O, I2C communication, game logic, and real-time interaction.

### ****Project Objectives:****

* Implement the Snake Game logic in Arduino using C/C++.
* Display the snake’s movement and fruit on an I2C 16x2 LCD.
* Use two push buttons for directional control.
* Avoid the use of external resistors by relying on Arduino’s internal pull-ups.
* Reinforce understanding of digital communication via I2C protocol.

### ****Working Principle****

* At start, the snake is displayed on the first line of the LCD as a character.
* The fruit (goal) is randomly positioned.
* Pressing buttons changes the snake’s direction.
* The snake grows each time it "eats" the fruit.
* The game ends when the snake runs into itself or moves out of bounds.

### ****Arduino Code Overview:****

The code uses the LiquidCrystal\_I2C library for displaying characters and controls button input with digitalRead().

**WORKING:**

|  |  |  |
| --- | --- | --- |
| ****Components**** | Component Coded to the Hardware | ****Code Snippet**** |
| Arduino Uno | Central microcontroller board. Controls game logic, reads input from buttons, and updates LCD via I2C. | cpp<br>void setup() { /\* Initialize components \*/ }<br>void loop() { /\* Run game logic \*/ } |
| I2C LCD Display | Connected to Arduino via ****I2C pins****: SDA → A4, SCL → A5. Displays game elements (snake, fruit, score, game over). | cpp<br>#include <Wire.h><br>#include <LiquidCrystal\_I2C.h><br>LiquidCrystal\_I2C lcd(0x27, 16, 2);<br><br>lcd.init();<br>lcd.backlight();<br>lcd.setCursor(0, 0);<br>lcd.print("Snake Game"); |
| Push Button – 1 | One leg to ****GND****, other to ****Pin 2**** on Arduino. Uses ****internal pull-up****. Reads as LOW when pressed. | cpp<br>pinMode(2, INPUT\_PULLUP);<br>if (digitalRead(2) == LOW) {<br> direction = "LEFT";<br> } |
| Push Button – 2 | One leg to ****GND****, other to ****Pin 3**** on Arduino. Also uses ****internal pull-up resistor****. Reads LOW when pressed. | cpp<br>pinMode(3, INPUT\_PULLUP);<br>if (digitalRead(3) == LOW) {<br> direction = "RIGHT";<br> } |
| Breadboard | Holds both push buttons securely and connects them to GND and digital pins via jumper wires. | No code – used for organizing the circuit. |
| Jumper Wires | Physically connect LCD and push buttons to Arduino pins through the breadboard. | No code – only for making connections. |
| Power Supply (USB) | Arduino is powered via USB. LCD and buttons draw 5V power through Arduino’s 5V and GND pins. | No code needed for power – handled by USB connection. |

### ****Code:****

### **#include <Wire.h>**

### **#include <LiquidCrystal\_I2C.h>**

### **LiquidCrystal\_I2C lcd(0x27, 16, 2); // Use 0x3F if needed**

### **// Snake game variables**

### **int snakeX[100], snakeY[100], length = 1;**

### **int foodX, foodY;**

### **int dir = 0; // 0=Right, 1=Up, 2=Left, 3=Down**

### **int score = 0;**

### **// Joystick settings**

### **#define JOY\_X A0**

### **#define JOY\_Y A1**

### **const int centerX = 280; // Measured center for your joystick**

### **const int centerY = 280;**

### **const int DEADZONE = 80; // Adjust this to reduce unwanted movement**

### **void setup() {**

### **lcd.init();**

### **lcd.backlight();**

### **lcd.print("Snake Game!");**

### **delay(1000);**

### **lcd.clear();**

### **snakeX[0] = 0;**

### **snakeY[0] = 0;**

### **spawnFood();**

### **}**

### **void loop() {**

### **readJoystick();**

### **moveSnake();**

### **checkCollision();**

### **drawGame();**

### **delay(300); // Adjust speed as needed**

### **}**

### **void readJoystick() {**

### **int xVal = analogRead(JOY\_X);**

### **int yVal = analogRead(JOY\_Y);**

### **int dx = xVal - centerX;**

### **int dy = yVal - centerY;**

### **if (abs(dx) > abs(dy)) {**

### **if (dx > DEADZONE && dir != 2) dir = 0; // Right**

### **else if (dx < -DEADZONE && dir != 0) dir = 2; // Left**

### **} else {**

### **if (dy > DEADZONE && dir != 1) dir = 3; // Down**

### **else if (dy < -DEADZONE && dir != 3) dir = 1; // Up**

### **}**

### **}**

### **void moveSnake() {**

### **// Shift snake body**

### **for (int i = length - 1; i > 0; i--) {**

### **snakeX[i] = snakeX[i - 1];**

### **snakeY[i] = snakeY[i - 1];**

### **}**

### **// Move head**

### **if (dir == 0) snakeX[0]++;**

### **else if (dir == 1) snakeY[0]--;**

### **else if (dir == 2) snakeX[0]--;**

### **else if (dir == 3) snakeY[0]++;**

### **}**

### **void spawnFood() {**

### **foodX = random(0, 16);**

### **foodY = random(0, 2);**

### **}**

### **void checkCollision() {**

### **// Check wall collision**

### **if (snakeX[0] < 0 || snakeX[0] >= 16 || snakeY[0] < 0 || snakeY[0] >= 2) {**

### **gameOver();**

### **}**

### **// Check self-collision**

### **for (int i = 1; i < length; i++) {**

### **if (snakeX[0] == snakeX[i] && snakeY[0] == snakeY[i]) {**

### **gameOver();**

### **}**

### **}**

### **// Check food collision**

### **if (snakeX[0] == foodX && snakeY[0] == foodY) {**

### **length++;**

### **score++;**

### **spawnFood();**

### **}**

### **}**

### **void drawGame() {**

### **lcd.clear();**

### **lcd.setCursor(foodX, foodY);**

### **lcd.print("\*");**

### **for (int i = 0; i < length; i++) {**

### **lcd.setCursor(snakeX[i], snakeY[i]);**

### **lcd.print("O");**

### **}**

### **lcd.setCursor(0, 1);**

### **lcd.print("Score: ");**

### **lcd.print(score);**

### **}**

### **void gameOver() {**

### **lcd.clear();**

### **lcd.print("Game Over!");**

### **lcd.setCursor(0, 1);**

### **lcd.print("Score: ");**

### **lcd.print(score);**

### **while (1); // Freeze**

### **}**

### ****Circuit Design:****

### 

### ****Challenges and Limitations:****

* **LCD Size:** Only 16x2 or 20x4 characters are available, limiting gameplay space.
* **Speed Control:** LCD refresh can be slow; optimizing game speed is necessary.
* **No Graphics:** Only characters, so symbols like #, \*, or custom characters are used to draw the snake and fruit.
* **Limited Growth:** Snake length is constrained by display size.

### ****Results and Output:****

The snake game runs smoothly on the LCD. The player can control movement using buttons, and the fruit reappears randomly on the screen. Upon collision, the game stops, and a "Game Over" message is displayed. The use of internal pull-ups worked well and reduced the need for external components.

### ****Future Improvements:****

* **Upgrade to OLED or TFT display** for pixel-based graphics.
* **Use buzzer for game sound effects.**
* **Add score tracking** and difficulty levels.
* **Implement restart/reset feature without rebooting Arduino.**
* **Store high score in EEPROM.**

### ****Conclusion:****

This project effectively demonstrates how an embedded system can be used to create interactive games using limited hardware. The Snake Game with an Arduino and I2C LCD is a fun and educational project that strengthens both hardware interfacing and programming skills. Avoiding resistors simplified the design and showed how internal features of microcontrollers can be leveraged