**Snake game**

Abdullah Tariq   
*Faculty of Computer Science*  
2022032  
u2022032@giki.edu.pk

Hasan Abbas   
*Faculty of Computer Science*   
2022204  
u2022204@giki.edu.pk

Muhammad Murtaza   
*Faculty of Computer Science*   
2022402  
u2022402@giki.edu.pk

***Abstract*—** The main objective of this project is to showcase the capability of sequential circuits to accomplish complex tasks, such as recreating the classic snake game using only fixed logic. This report details the components and designs of each sub-circuit simulated in Proteus, along with their hardware implementation. Key components utilized in this project include shift registers, multiplexers, demultiplexers, adders, and counters.

***Keywords*— *sequential circuits, snake game, fixed logic, hardware implementation, shift registers, multiplexers, demultiplexers, adders, counters.***

# **INTRODUCTION**

Creating a snake game typically involves using a microcontroller interfaced with a display. However, this project aimed to challenge conventional methods by implementing the same game logic using only fundamental non-programmable integrated circuits (ICs). By forgoing the use of a microcontroller, the project delved into the realm of low-level digital design, showcasing the potential for creative solutions within constraints. This approach not only emphasizes resourcefulness but also highlights the foundational principles of digital electronics, making it an intriguing endeavor for exploring the possibilities of minimalist hardware design.

Furthermore, the project served as an educational endeavor, demonstrating how complex functionalities can be achieved through the careful orchestration of basic building blocks in digital electronics. This experience not only enhanced technical skills but also fostered a deeper appreciation for the underlying principles of computer hardware. Overall, the project stands as a testament to the ingenuity and adaptability of non-programmable ICs in recreating a beloved classic game, while also serving as an educational platform for exploring the fundamentals of digital design.

**METHODOLOGY**

The methodology for this project involved a ground-up approach, where every aspect was developed from scratch. Only the operations of integrated circuits (ICs) were referenced throughout the process. The project began with conceptualization, followed by component selection based on compatibility and functionality. Circuitdesigns were created, simulated, and then physically implemented on breadboards.

# **PROJECT OVERVIEW**

The complete designed is based upon 4 sub-circuits:

* Input Direction
* Movement Processing
* Length Calculation
* Output (4x4 display)

Each sub-circuit was initially designed and simulated using non-programmable ICs. However, due to time and resource constraints, the complete project could not be implemented solely with fixed logic. Only the movement processing was accomplished through fixed logic, while other sub-circuits were outsourced to an Arduino Uno. Additionally, Parallel-In-Serial-Out (PISO) and Serial-In-Parallel-Out (SIPO) registers were utilized to facilitate data movement between the non-programmable ICs and the Arduino, enabling efficient data processing.

##### **INPUT**

## **Fixed Logic Design in Simulation**

## A. Components used

* Toggle switches (Up, Down, Left, Right)
* D-latches (7474)
* 4-input OR Gate (4072)

## B. Working

The input control design for the snake game enables users to direct the movement of the snake within the game environment. Four toggle switches represent directional inputs: Up, Down, Left, and Right. Each toggle switch corresponds to a specific direction, enabling users to select their desired direction of movement.

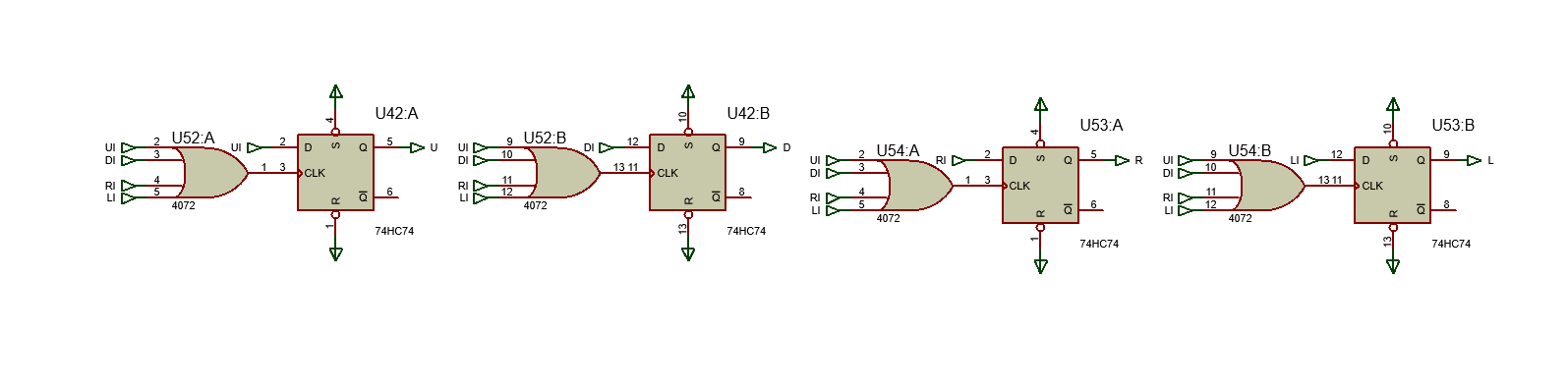
Four D-latches store the state of the direction. Ideally, only one of the latches will be activated while the others remain inactive. The outputs of these latches are connected to the movement calculation circuit, which determines the coordinates of the snake's head in the next game loop.

When a switch is pressed, all the latches enter transparent mode as the switch output transitions from low to high. Each switch output is connected to the D input of a latch, activating the corresponding direction while deactivating the others.

*C. Simulation Results:*

The input control design was successfully simulated in Proteus, accurately capturing user input and controlling the movement of the snake within the game environment.

*D. Proteus Schematic of input*



## **Hardware Implementation (with Arduino)**

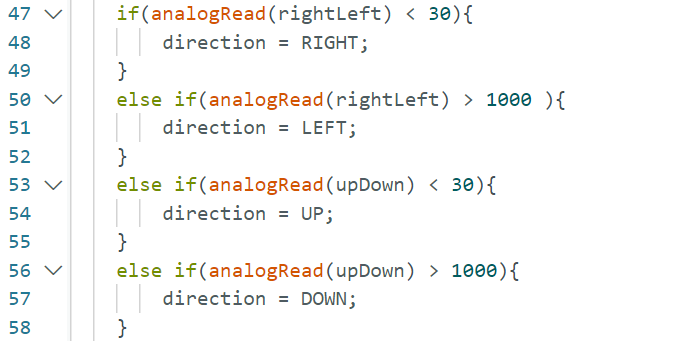
## A. Components used

* Arduino Uno
* Joystick Module
* De-multiplexer (SN74LS139)

*B. Working*

Each direction is assigned a number (0 for right, 1 for left, 2 for up, 3 for down). A variable named direction stores the current direction. This variable is updated only after moving the joystick (connected with pins A0 and A1) to the extreme of the desired direction. The variable is converted to binary and written to pins 11 and 12 of the Arduino. These pins are connected with a de-multiplexer (de-mux) to convert the binary number into 4 direction states.

*C. Code*

**

**MOVEMENT PROCESSING**

## A. Components used

* 4-bit Universal Shift Registers (SN74LS95)
* 4x1 Multiplexers (74LS153)
* 4-bit Full Adders (74LS83)
* 2x1 Multiplexers (74LS157)
* 1x4 De-Multiplexer (74LS139)
* 2-bit Counters (4520)

*B. Working*

The processing module implements eight Universal Shift registers (4 for each dimension) in Parallel-In-Parallel-Out (PISO) configuration to store the four 2-dimentional coordinates of each block of the snake.

In order to update these coordinates, two 2-bit circular counters, pointing towards the latest-updated (head) and oldest-updated (tail) block of the snake, respectively.

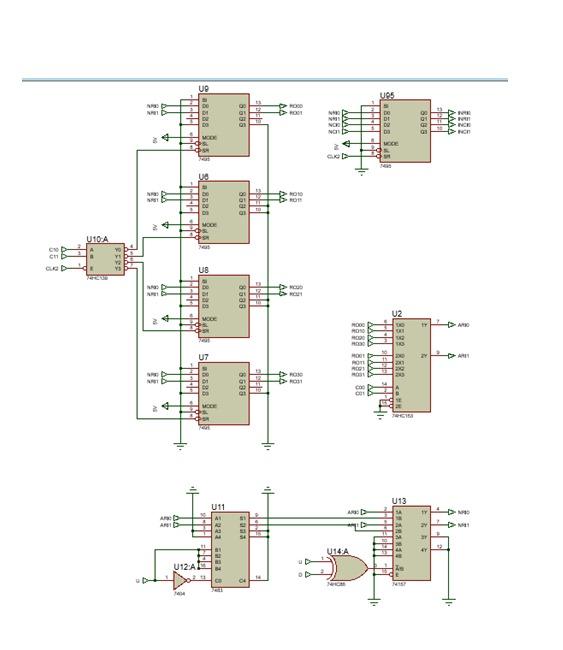
The value of the head block is retrieved using four 4x1 multiplexers, with their input lines connected to the most significant bit (MSB) and least significant bit (LSB) of four rows and four column shift registers, and their select lines connected with the head counter. This retrieved value is then processed by two 4-bit adders and two 2x1 multiplexers (one for each dimension), which adjust the value based on the input direction. The updated value is then distributed to all shift registers, with the new row value sent to the four row shift registers and the new column value sent to the four column shift registers.

The select lines of two 1x4 (one for each dimension) de-mux is connected with the tail counter. The output lines are connected with the left-clock of each register, and the enable line is connected with a store clock. As the next frame occurs, the falling edge of the store clock is passed to the register pointed to by the tail counter, updating its value and effectively shifting the snake’s body.

For example, if the snake is represented as "123\_" and it moves right, the new configuration becomes "\_231", indicating that the head has moved one step to the right. The tail becomes the new head, and the snake has effectively moved.

The counters are updated 50ms after the falling edge of the update clock in order to avoid any unpredictable behavior.

*B. Processing Schematic on Proteus.*



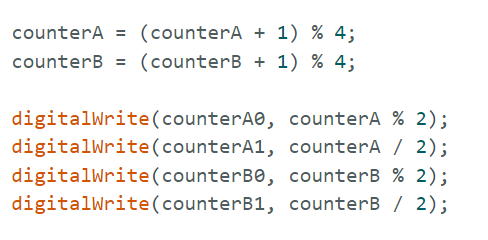
This is for a single dimension.

*C. Hardware Implementation*

This part of the circuit was implemented, with the exceptions of counters and clocks, which were assigned to the Arduino. This approach was taken to avoid any synchronization issues with the Input and Output modules.

**

*D. Code*

**

###### LENGTH CALCULATION

## **Fixed Logic Design in Simulation**

## A. Components used

* D-Latches (7474)
* 1x4 De-Multiplexer (74LS139)
* 2-bit Counters (4520)
* 4-bit Comparator (7485)
* AND gates (7408)
* OR gates (7432)
* NOT gates (7404)

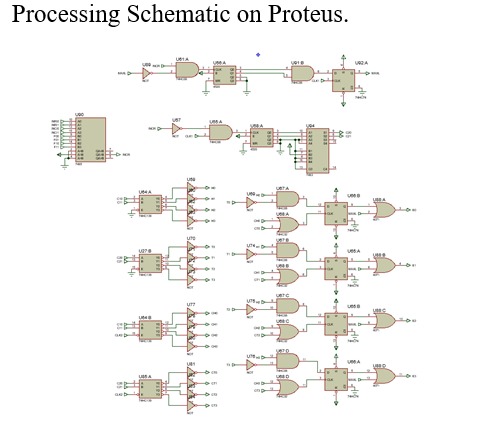
*B. Working*

To vary the length of the snake, four D-latches corresponding to the four coordinates are used, with each latch indicating whether the corresponding block will be visible or not. Additionally, a third counter, referred to as the virtual tail counter (distinct from the tail counter in the previous section), is established.

Initially, the virtual tail counter is set to be two steps behind the head counter. The latch is set whenever the head counter points towards the block, whereas it is reset whenever the virtual tail counter points towards it. The difference between these counters determines the length of the snake, which is initially set to 2.

When a fruit is detected using a comparator, the virtual tail counter is not incremented for the next pulse, effectively increasing the difference between the two counters and thus increasing the length of the snake.

*C. Schematic*



***Hardware Implementation***

This part of the circuit was not implemented due to the design being in its initial stages and not being deemed sufficiently reliable. Additionally, time constraints played a significant role in the decision to forego this aspect of the game.

###### OUTPUT (DISPLAY)

## **Fixed Logic Design in Simulation**

## A. Components used

* 1x4 De-Multiplexers (SN74LS139)
* 2 Input Nor Gates (4007)
* 4 Input Or Gates (4027)

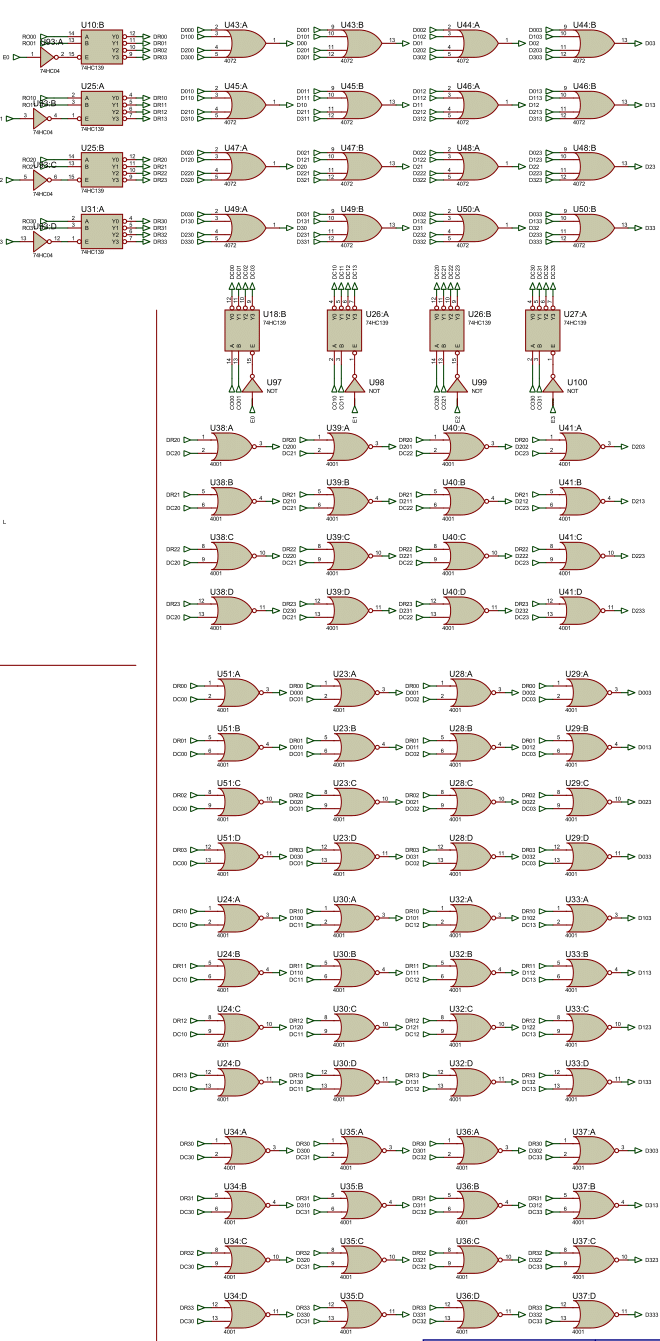
## Working:

The display module of the snake game project processes and presents the game environment to the user in real-time. It involves converting the coordinates of the snake blocks into a 4x4 display matrix.

To display a single block of the snake, its row and column values are input into two demultiplexers. Each output of the row demultiplexer is combined with each output of the column demultiplexer using NOR gates instead of AND gates, given the demultiplexers' active-high logic (according to Boolean’s law). This combination forms a 4 by 4 matrix with 16 bits, where one bit is active, indicating the coordinate's position based on the active inputs of the NOR gates. The input lines of both demultiplexers share a common connection with the complement (NOT) of the corresponding enable latch, determining whether the block is hidden or visible.

This process is repeated for each of the four blocks, and the outputs of each matrix are then ORed together to create the final 4 by 4-bit matrix. This matrix is then connected to the corresponding LEDs, displaying the snake block.

*C. Schematic* 

**

## **Hardware Implementation (with Arduino)**

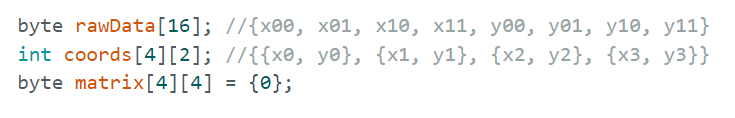
## A. Components used

* Universal 4-bit Shift Registers (74LS95)
* Arduino Uno
* 4 x 4 LED matrix

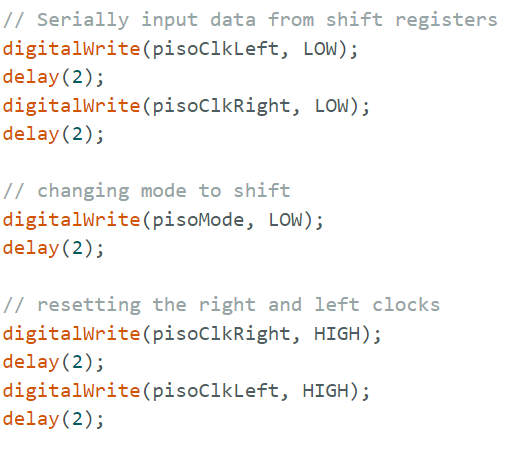
*B. Working*

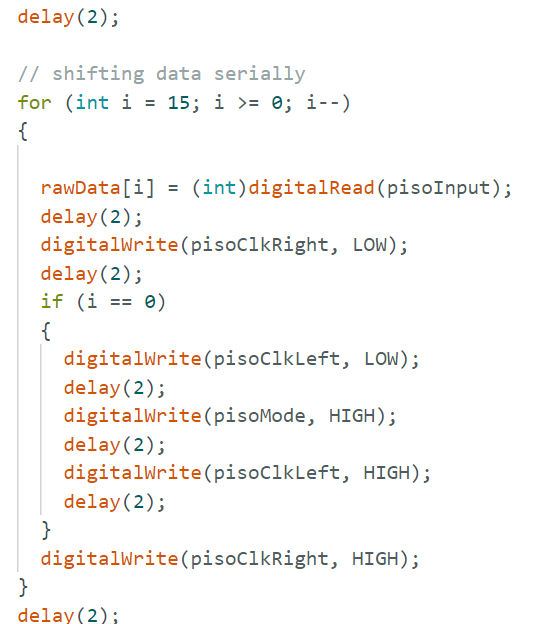
4 shift registers for PISO, as well as SIPO, are daisy-chained together to move data from the process module into Arduino and from Arduino to display LEDs. The clocks of both the circuits are connected with Arduino. The following 5 algorithms are run

* + - 1. Initialization

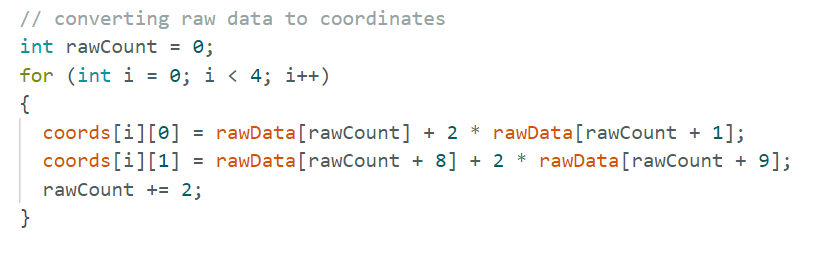


* + - 1. Shifting raw bits from PISO

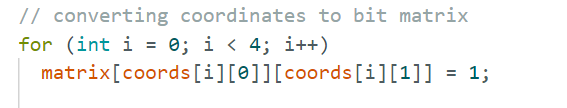




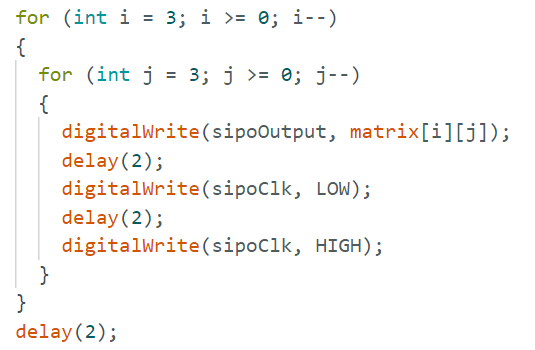
* + - 1. Converting the raw bits to a coordinate array



* + - 1. Converting the coordinate array to bit matrix



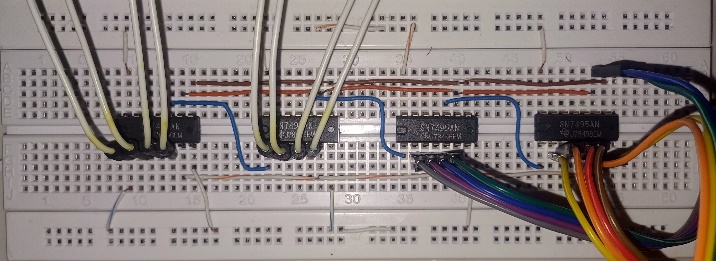
* + - 1. Shifting the bit matrix to SIPO

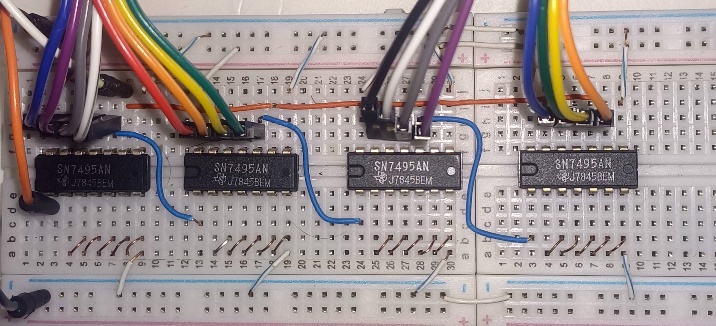


The outputs of the SIPO circuits are connected with 16 LEDs arranged in a 4 by 4 mesh, representing the display

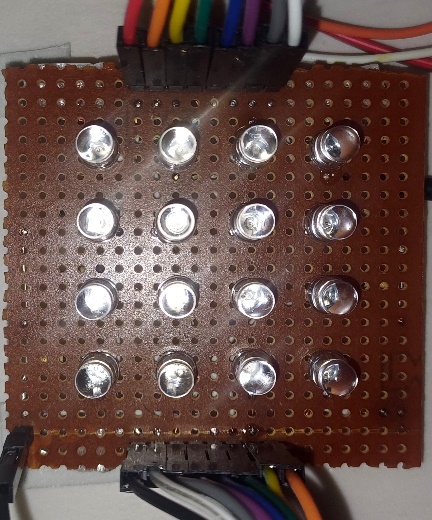
*C. Images*

PISO:



SIPO:   


DISPLAY:



**COMPONENTS**

**Shift Registers**: Shift registers are the backbone of this project. They are used for storing the coordinates of the snake blocks as well as to expand the number of output pins available for driving the LED matrix. They allow for efficient control of multiple LEDs with fewer Arduino pins.

**Multiplexer/Demultiplexer:** Multiplexers and demultiplexers facilitate the selection and routing of signals between various components, achieving effective read/write operations on the shift registers

**Adders:** Adders and comparators play a role in arithmetic and logical operations within the circuitry, enabling the manipulation and comparison of data to facilitate gameplay logic.

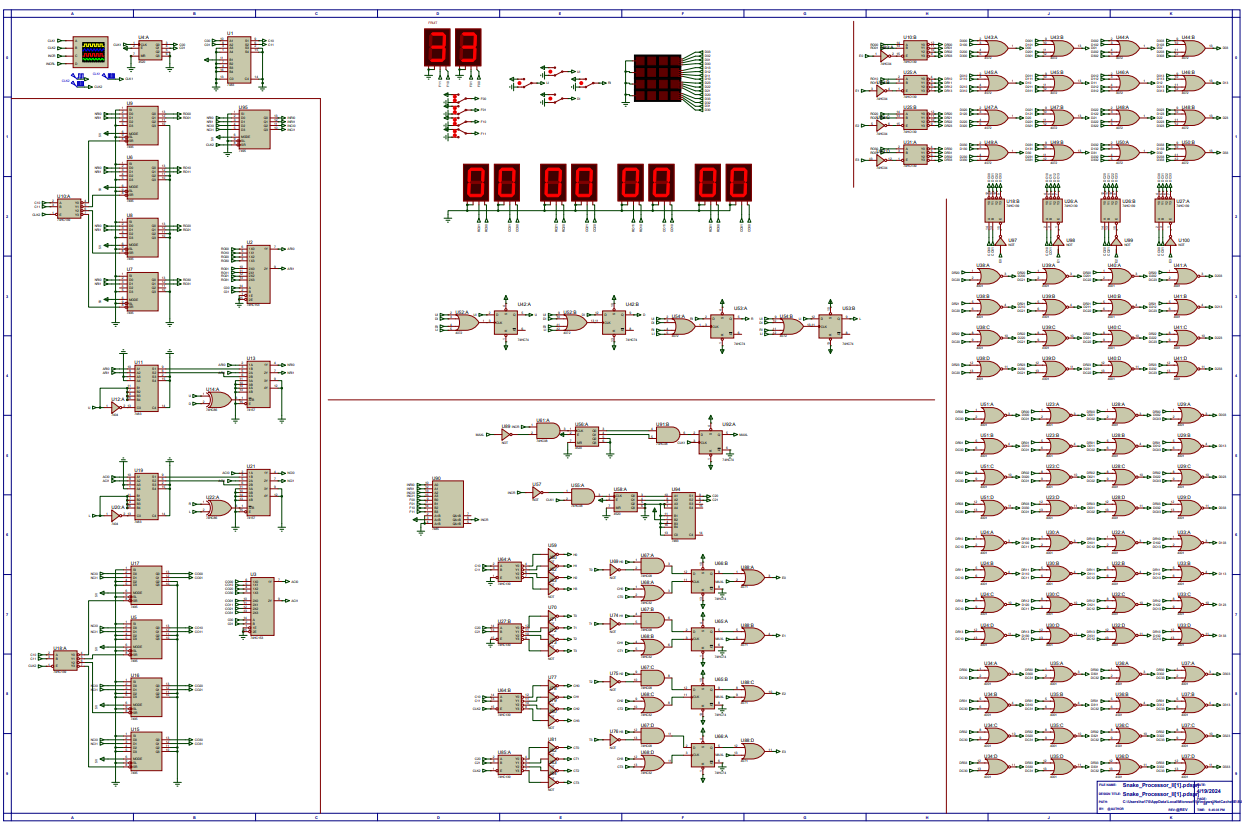
**Arduino Uno:** The Arduino Uno serves as the primary microcontroller for the project, providing the necessary processing power and control capabilities.

**Breadboard:** A standard breadboard serves as the platform for integrating and connecting the various hardware components, providing a convenient and flexible layout for circuitry.

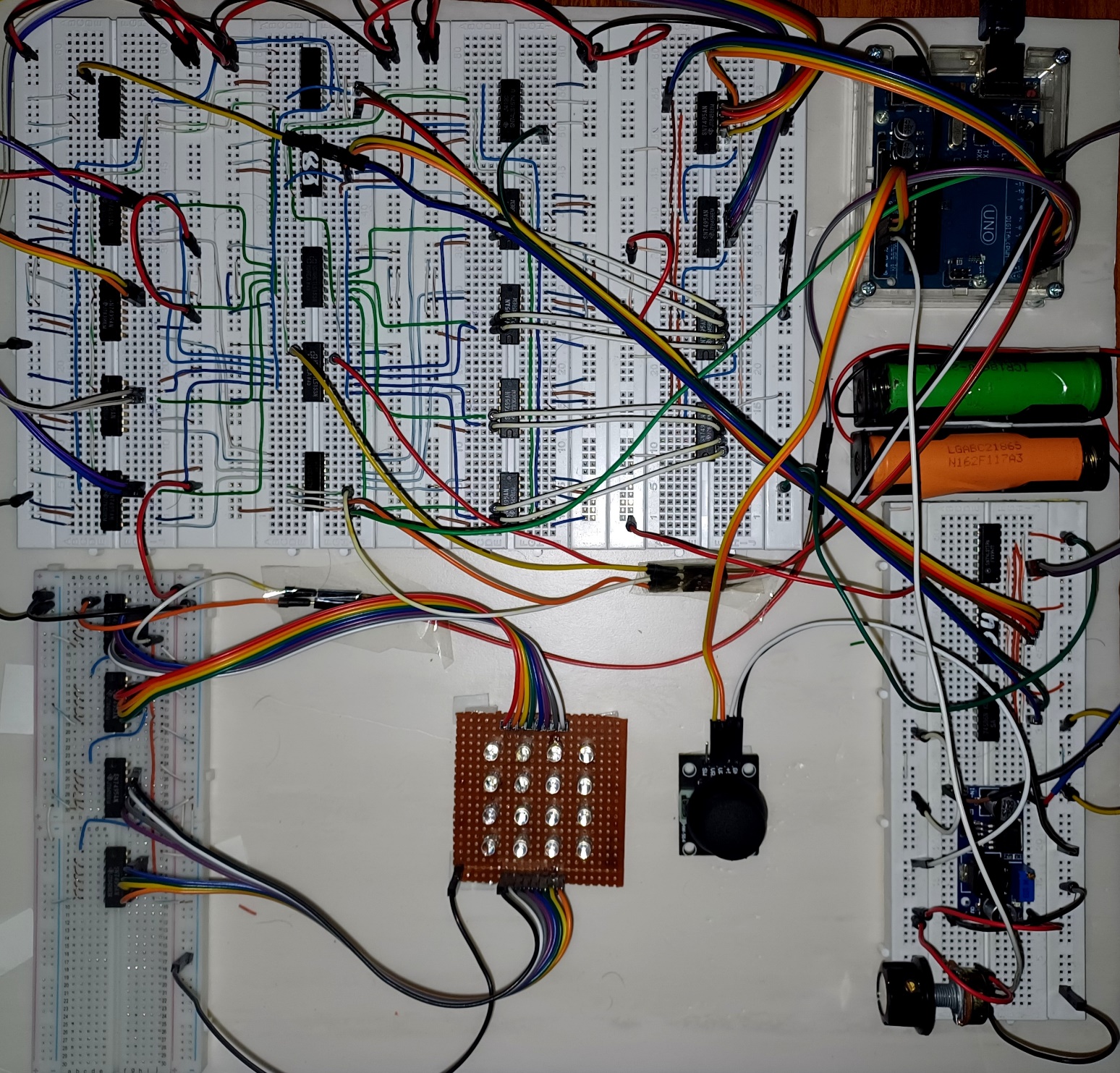
**Joystick Module:** The joystick module enables user input by detecting movement along both the X and Y axes. It consists of potentiometers that translate physical movement into analog signals.

**4x4 LED Matrix:** The 4x4 LED matrix serves as the display interface for the snake game, providing visual feedback to the user regarding the snake's position and game state.

**Wiring and Connectors:** Various wiring components such as jumper wires, connectors, and terminal blocks are employed to establish connections between the different hardware elements and ensure reliable operation.

**PROTEUS SCHEMATIC**

**IMPLEMENTATION**

****

# **REFERENCES**

* 4-BIT BINARY FULL ADDER WITH FAST CARRY. (n.d.).
* DM74LS153 Dual 1-of-4 Line Data Selectors/Multiplexers. (n.d.). Retrieved from https://eeshop.unl.edu/pdf/DM74LS153.pdf
* SN54F157A, SN74F157A SELECTORS/MULTIPLEXERS DATA 1-LINE TO 2-LINE QUADRUPLE. (1993). 8. Retrieved from https://www.ti.com/lit/ds/symlink/sn74f157a.pdf?ts=1714735510391&ref\_url=https%253A%252F%252Fwww.google.com%252F
* SN7495A Datasheet(PDF) 4 Page - Texas Instruments. (1988). Retrieved from https://html.alldatasheet.com/html-pdf/27427/TI/SN7495A/21/1/SN7495A.html
* SN74LS139 Dual1-of-4Decoder/Demultiplexer. (n.d.). Retrieved from https://pdf1.alldatasheet.com/datasheet-pdf/view/12626/ONSEMI/74139.html

**PROJECT LINK**

https://github.com/AbdullahTariqCS/Hardware-Snake-Game