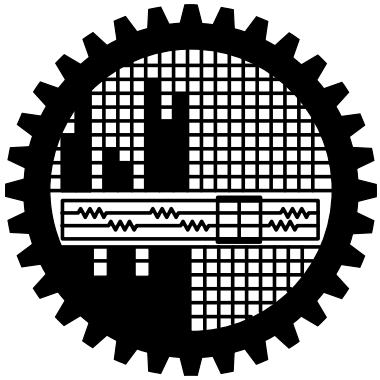

CSE316 Microprocessors, Microcontrollers, and Embedded Systems Sessional : Term Project

Tetris

Asif Ajrof (1705092), Fatima Nawmi (1705093), and

Supervised by
Md. Masum Mushfiq
Lecturer

Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology

July 2021

1 Motivation

Remaking the popular arcade game Tetris in a simple platform using microcontroller and LED matrices.

2 Components

The main components of our project are:

1. ATmega32
2. 8x8 LED Dot Matrix (common cathod)
3. 16x2 LCD Screen
4. Thumb Joystick
5. Buzzer

Some other components that were also used:

- Resistor
- Inductor
- Capacitor
- Wire
- Potentiometer
- USBasp Programmer

3 Short Description of Each Module

We have several different modules in this project. Here is a short description summarizing what each module does individually.

3.1 Main Console

The main gaming console is made of two 8x8 LED Dot Matrix, so we have a 16x8 display for the gameplay.

3.2 Pieces

We have kept all the 7 different pieces of Tetris game (I,J,L,T,S,Z,O). The pieces are generated randomly, and for each game the pieces are random as well.

We show the next piece to come in the LCD screen.For this we have ensured communication between two ATmega32.

3.3 Movement

The thumb joystick is used for four possible movement, left, right, down and rotate (up move of joystick). Also, for starting the game an up move is needed.

3.4 Score Update

When a row is filled the score is updated in the LCD screen and the filled row is removed while the rows above come down to fill up the empty row. When four consecutive rows are filled at the same time, as a bonus double points are added.

3.5 Game Speed

After certain time intervals, the game increases its speed. The speed resets when a new game is started again.

3.6 Sound

The buzzer makes a sound when there's a score update, a new game starting and when the game is over.

3.7 Game over and New game

When there is no more place for a new piece to enter, the game is over.
For a new game to begin after the last game is over,the display shows a play button and an up movement is needed.

4 Algorithm used in gameplay

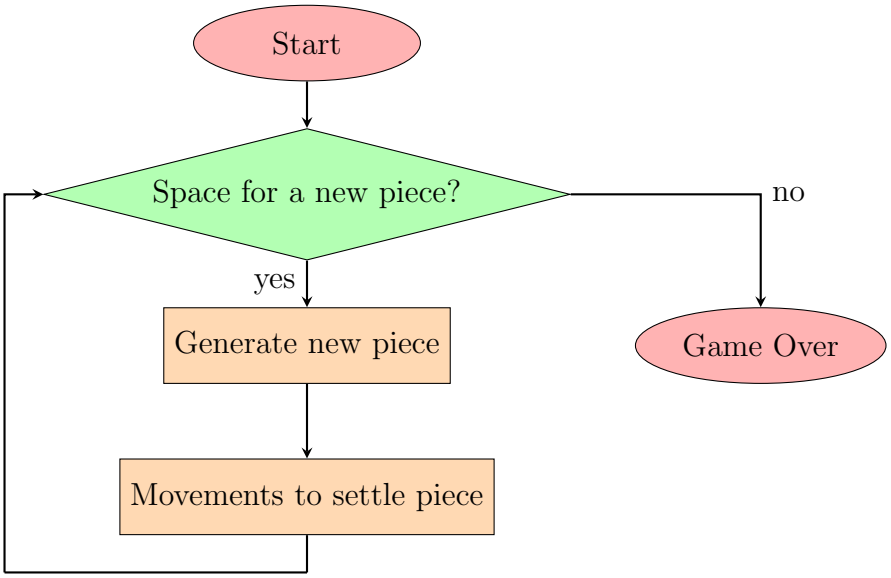

Figure 1: Algorithm

5 Methods used

5.1 UART

For having a simplex data transfer we used UART subsystem. We used 9600 bps baud rate for data transmission.

5.2 ADC

We have used the ADC chip for collecting analogue movement data from the thumb joystick and converted it to digital for our piece movements.

6 Circuit Diagram

We have used two ATmega32 microcontrollers.
The first one is connected with the main console, i.e., two LED dot matrices, buzzer and the thumb joystick. Also it sends data to the second one via TXD and RXD pins using UART communication. The second ATmega32 is connected with a LCD screen.

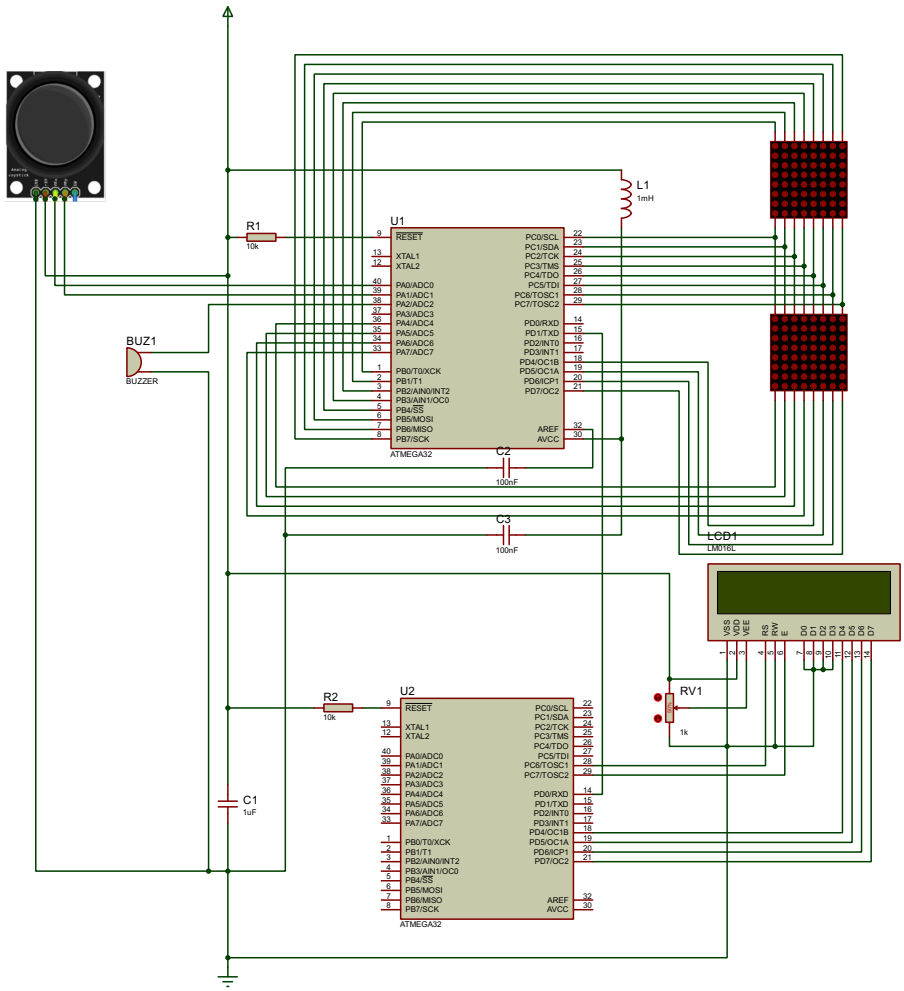

Figure 2: Detailed Pin Diagram

7 Challenges

- We noticed that while using UART serial communication the data transfer was delayed by few seconds, for fixing this we tried using a larger Baud rate for a larger bit per second, but after some trial and error even with a larger Baud rate there was some delay, hence when the score gets updated or game over or a new game there is a slight delay before the LCD gets updated.

- We faced a problem in our main display LED matrices, every next row was emitting very dim light under the actual positions. We figured since we are multiplexing the rows and columns, and we are changing the row values first and then changing the column values, as a result for a very brief time the columns are getting previous values for an

updated row value, hence the dim emission of light is occurring. We fixed this problem by clearing the values of columns after a short delay in each iteration.

- While implementing the movements a delay is needed because without delay a single press would trigger multiple moves. But a delay function for every movement pauses all the other functionalities. So, instead we kept a variable, that keeps count of the button presses and ensures one movement for one press.

- While designing our main circuit we first wanted to use one ATmega32 with the help of decoders, for all our operations. We tried simulating this with Proteus, but we noticed because of using decoder the entire process was becoming very slow. So we excluded this idea from the designing phase and designed with two ATmega32 which also enabled us to learn about a serial communication method.

- Even though we needed simplex serial communication between the two microcontrollers we found the if we did not enable the RXC and TXC of UCSRA registers in both ATmega32 code it was not working in hardware but it was working in simulation. Hence we used duplex mode for implementing simplex communication.

- LCD shows garbage values if there is a slight change in voltage, after reconnecting the voltage source this problem goes away.

- Implementing all the logistics of the game into hardware was very challenging. Implementing the same task in simulator was widely different than implementing it in hardware.