BEJEWELED IN ARDUINO

DUȚU ALIN CĂLIN

CONTENTS

1 INTRODUCTION

1.1 Context

In the far reaches of the digital universe, deep within the Quantum Matrix, exists a network of energy crystals that power the very fabric of space and time. As a skilled "Crystal Architect," your mission is to align and harmonize these energy crystals to stabilize the system before it collapses into chaos.

The Bejeweled in Arduino is a project for designing and developing a Bejeweled-style game for Arduino UNO to demonstrate that even games that require computational resources can be run on less potent hardware such as microcontrollers.

1.2 Objectives

This project aims to provide valuable insights into the process of creating games for microcontrollers, exploring the unique opportunities they present as well as the technical and design challenges developers must take into account and overcome. These challenges include managing limited memory and processing power, optimizing input/output handling, designing intuitive interfaces with minimal controls, and ensuring efficient power usage. Additionally, developers must navigate constraints in graphical and sound capabilities while maintaining engaging gameplay experiences. The main objectives of this project are:

- Implement a pleasant menu interface for easy navigation
- Implement the matrix board and the gem models for gameplay
- Implement the UI interface elements for gameplay action
- Implement game-saving mechanic
- Develop a resource-efficient game that runs smoothly on the microcontroller

2 PROPOSED SOLUTION

2.1 Solution presentation

The project aims to implement a resource-efficient game resembling a puzzle game where the player needs to make a row or column match of gems to score points. The game is required to run smoothly on an Arduino UNO, a microcontroller with 32K bytes of Flash memory and 2K bytes of SRAM provided by an ATMega328p[1]. Knowing the project's purpose and complexity, the following architecture has been considered:

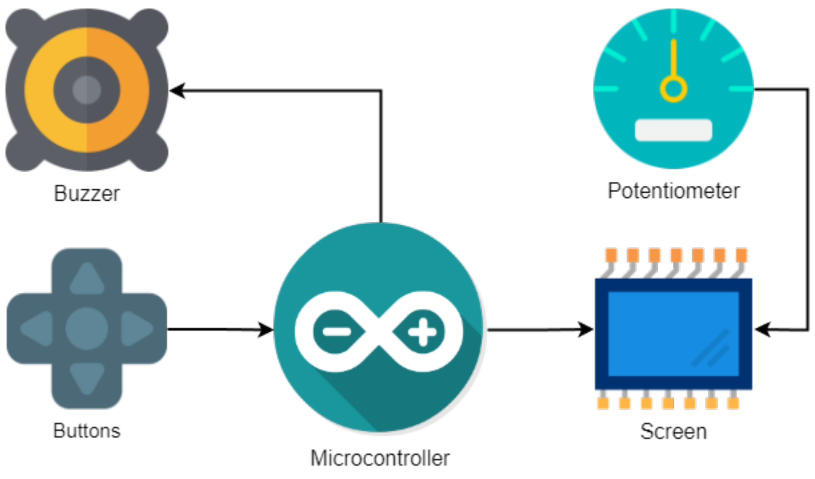

Figure 1: Architecture

There are 5 components identified in the architecture:

- The microcontroller - The main component that runs the game, registers any input, and executes it as an action that can be seen on the output
- The screen - The component that shows the game's UI and any action processed by the microcontroller
- The potentiometer - The component used to adapt the screen's brightness
- The buttons - A set of 6 buttons, 4 directional, 1 "Back" and one "OK" button, used to navigate through the main menu and do actions in the gameplay sessions
- The buzzer - A little component used to emit feedback sounds and sing when gameplay sessions are finished

The game's user interface has a main menu and a gameplay section.

The main menu can be used to select one of the available game modes, pause and continue a game, save any game session, and view the leaderboard with the highest scores ever registered.

The gameplay section is where the player will have his gaming sessions. The game's purpose is to move the gems to create row and column matches, although the main mission depends on the chosen game mode.

3 IMPLEMENTATION DETAILS

3.1 Hardware

Even for a game like Bejeweled which itself is a complex game to make, the hardware required is minimal and affordable. The following list enumerates all the hardware components used in the project:

- 1 x Arduino Uno
- 1 x Breadboard
- 1 x 84x48 resolution Nokia 5510 LCD display[2]
- 1 x Buzzer
- 1 x potentiometer
- 6 x button
- 4 x 10K resistors
- 2 x 1k resistors
- 1 x 100Ω resistor
- Wires for the connections between components

The assembling process of the project is quite easy and appealing and the hardware schematic demonstrates how simple it is to build:

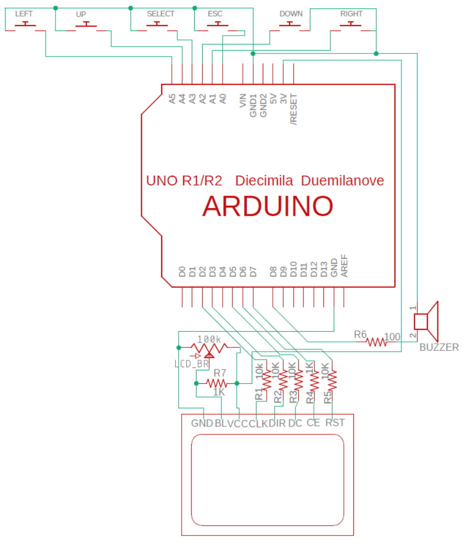

Figure 2: Hardware schematic

3.2 Software

The software side of the project, on the other hand, holds up most of the project's complexity because multiple challenges had to be dealt with. From a developer's standpoint, the functionalities can be split into 4 categories:

- Main Menu development
- Gameplay development
- Leaderboard development
- Exit development

3.2.1 Main Menu development

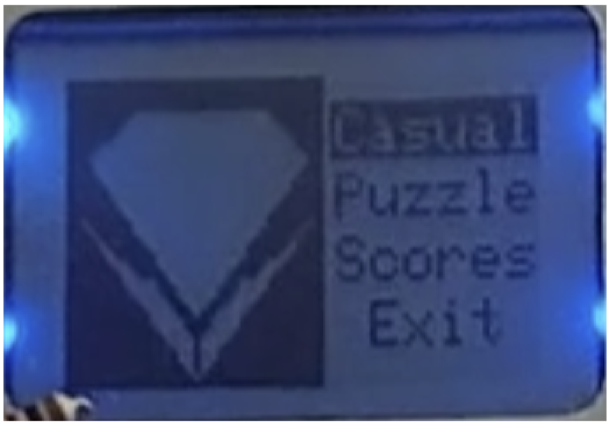

Figure 3: Main menu

The previous figure depicts the main menu, which has on the left side a picture with the game's representative gem converted into an 8-bit bitmap format. On the right side, there are 4 options that the player can select:

- Casual Mode
- Puzzle Mode
- Leaderboard
- Exit

3.2.2 Gameplay development

When selecting the Casual or the Puzzle game mode, a new game mode menu appears, which is related only to the chosen game mode. This game menu can be easily identified by the previously mentioned image which is split evenly on the sides of the screen and by the following options which now appear in the middle of the screen:

- New Game - Starts a new game
- Continue - Continues a saved game
- Back - Moves back to the main menu

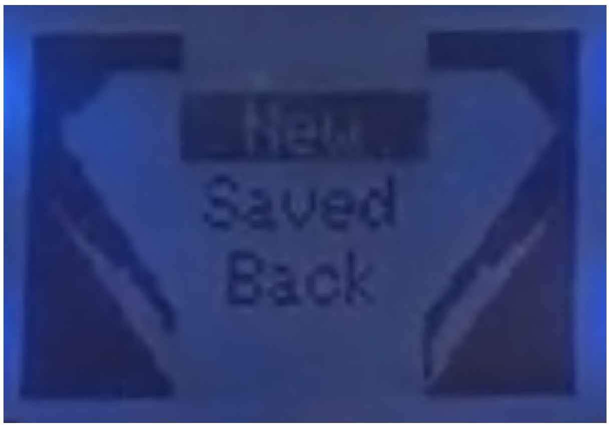

Figure 4: Game menu

Gameplay interface

The gameplay interface is quite the same in any of the game modes. On the left side of the screen, there is the accumulated score for that round and a thin but long line from the top to the bottom of the screen which shortens as time passes, that is the timer and when the line disappears, an Arduino interruption which acts as a listener ends the game and the results are shown on the screen. The timer can be seen only in Casual games, while in Puzzle mode, it is changed to a simple grid line.

The right side of the screen is covered by the matrix board of gems, the place where the game is played. The board itself is a 46x60 bitmap and contains the gems to be matched to score points. The gems are a 6x8 bitmap representation of the original gems from the original Bejeweled game[3].

Gameplay

To make a combination the player has to move the selector to a certain gem, press the "OK" button to select it and move the gem using the directional buttons to set a 3+ match. Based on the number of gems matched, the score increases.

When making one combination, multiple things are happening behind the scenes. First, after the combination has been executed, the score is increased, the matched gems disappear from the board, the gems from the above are moved to the bottom, and the empty spaces from the top of the board will be filled with random gems so that the player won't ever remain with no moves to make. After all the gems are on the board, it is checked whether there are additional matches after new gems have been generated, in which case the cycle repeats until no matches are found.

The algorithm used for generating gems so that the previously mentioned condition is met is a form of backtracking that iterates through each possible combination of gems and chooses the first one that meets the required condition.

There are 2 game modes the player can choose, each one offering its challenge. The Puzzle mode focuses on strategy and tactical thinking, requiring the player to carefully plan each move to make all the gems from the board disappear, while the Casual mode emphasizes fast-paced action, testing the player's ability to make as many combinations as possible in a short amount of time. Regardless of the choice, both modes are designed to provide an engaging and immersive experience, tailored to different playstyles.

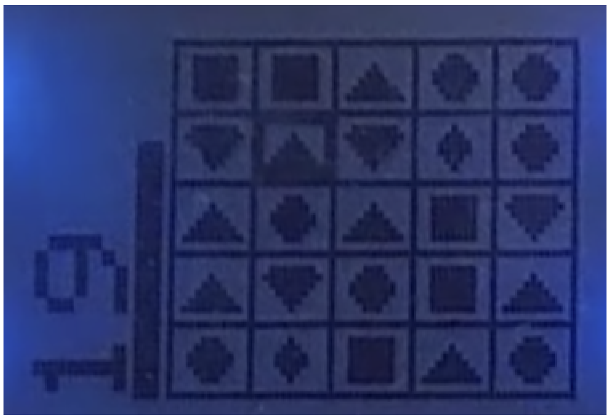

Figure 5: Casual gameplay

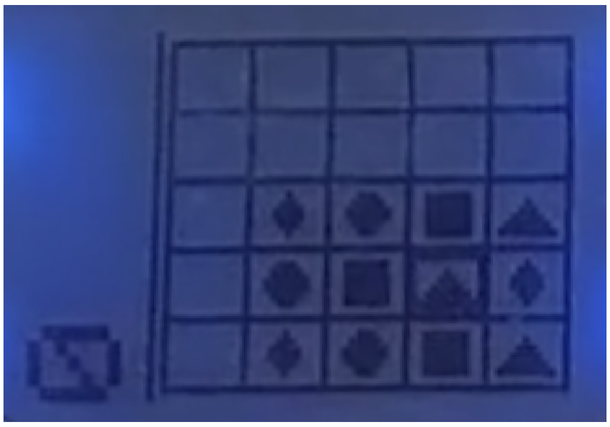

Figure 6: Puzzle gameplay

Game Submenu

The game submenu can be accessed by pressing the "Back" button while playing. This submenu serves either to set a game to pause and resume a bit later, save the game in Arduino's memory, or exit without saving.

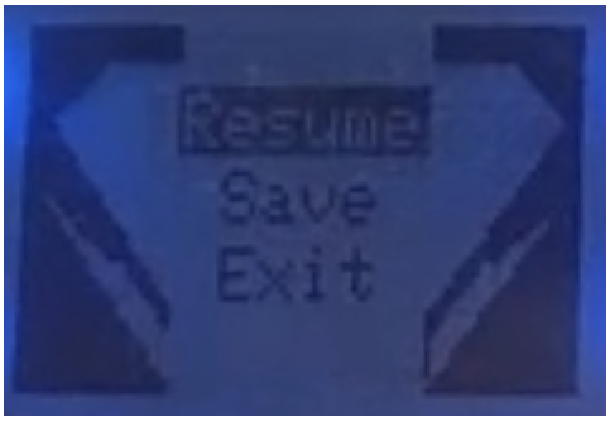

Figure 7: Gamemode submenu

3.2.3 Leaderboard development

The Leaderboard shows the top scores for each game mode. Switching from one leaderboard to the other is done by the left and right buttons from the console controls.

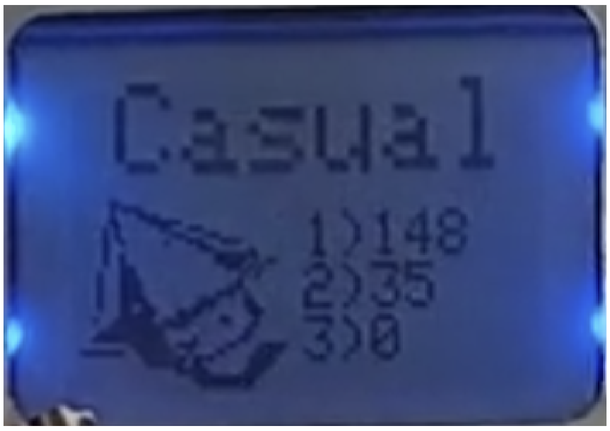

Figure 8: Casual Leaderboard

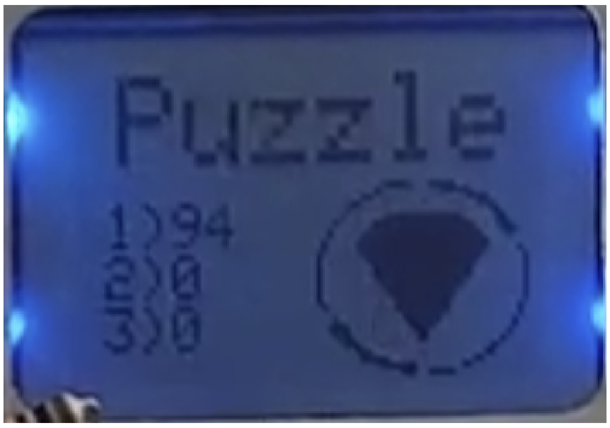

Figure 9: Puzzle Leaderboard

3.2.4 Exit development

The exit option has multiple functionalities and it is mandatory to use when exiting the game. When saving a game or registering a new high score, this data is stored in the microcontroller's memory only.

When a player is ready to quit the game, he can persist its saves and high scores by selecting the exit option which saves the data in Arduino's EEPROM memory, a persistent memory that will be read when the game opens.

3.3 Testing

Multiple demos have been conducted for testing the implemented functionalities and can be found publicly at the following links:

- Demo Casual Mode - `https://youtu.be/6oMhK3js9Dg`
- Demo Puzzle Mode - `https://youtu.be/KmDn7bPvKUg`
- Demo Game saving - `https://youtu.be/WtNOwS99JUc`
- Demo Leaderboard and Exit - `https://youtu.be/LWpidVnRRmQ`

4 CONCLUSIONS

In conclusion, this program aims to provide valuable insights into the process of creating games for microcontrollers. This project explored the unique opportunities the software in combination with the hardware offering a different experience in playing video games as well as present the technical and design challenges developers must take into account and overcome the limitations such as storage, memory and conflicts of development library.

BIBLIOGRAPHY

[1] Atmel. ATMega328p datasheet. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. Latest access: 22.09.2024.

[2] ETT. Nokia 5510 display datasheet. http://masserv.utcluj.ro/~florind/cursuri/ISM/_ISM_Lab/User_Manual_ET_LCD5110.pdf. Latest access: 22.09.2024.

[3] PopCap games. Bejeweled 2.

[4] Duțu Alin Călin. OCW documentation website. https://ocw.cs.pub.ro/courses/pm/prj2022/ndrogeanu/bejeweled_in_arduino. Latest access: 22.09.2024.