Documentation
Mini Games using Arduino

Fabian Stiewe

Academic Year 2024-2025

Contents

1 Introduction **2**

2 Project Description **2**

2.1 General Framework . . . 2

2.2 TicTacToe . . . 2

2.2.1 Additional components . . . 2

2.2.2 Code . . . 3

2.2.3 Programming Guide . . . 3

2.3 Simon Says . . . 4

2.3.1 Additional components . . . 4

2.3.2 Code . . . 5

2.4 Programming Guide . . . 5

2.4.1 Programming Guide . . . 5

2.5 Any other game . . . 6

2.5.1 Additional components . . . 6

3 Circuit Diagram **7**

4 Circuit Blocks **8**

1 Introduction

This document is a collection of all the information needed for the project documentation. It is a summary of the project and includes all the necessary information for the project.

The file titled "Project_From.pdf" provides more general information about the school, year, learning objectives, and other relevant details, filling in the gaps in this document.

2 Project Description

My project is about creating different "MiniGames" with the Arduino Uno. The games are simple and fun to play. The games are implemented using the Arduino Uno and the components mentioned below. The games are TicTacToe, SimonSays, and any other game the students come up with.

2.1 General Framework

In all projects only the following components are mandatory, depending on the project the students choose, additional components may be needed (*see 2.5*):

- Arduino Uno
- Breadboard
- Jumper Wires

2.2 TicTacToe

TicTacToe is a two player game, where the players take turns to place their symbol on the board. The board is a 3x3 grid. The first player to get three of their symbols in a row, column or diagonal wins the game. The game ends in a draw if all the fields are filled and no player has three of their symbols in a row, column or diagonal.

2.2.1 Additional components

For my implementation of the TicTacToe game I used the following additional components:

- LED
  - Green LED
  - Red LED
  - Yellow LED
  - (Blue LED)
- Buzzer
- OLED Display
- Keypad
- Resistors (220Ω)
- Diverse cables

You can see the circuit diagram in Figure 1 and the circuit blocks in Figure 2.

2.2.2 Code

The sketch operates in the following manner:

- Initially, you **initialize** the *OLED Display*, *Keypad*, *LEDs*, and *Buzzer* within the `setup()` function.
- Subsequently, you create various **variables** for the game, such as the `board`, the `current player`, the `winner`, and so on.

Firstly, you draw a **Menu screen** to introduce the user and prompt them to **press `1` to commence the game**. Upon pressing `1`, the game commences, the **GameStatus** undergoes a change, and the user is permitted to play. During the game:

- The **board** is drawn, and it also displays the standings.
- The user can **press a button** from 1 to 9 on the keypad to **place their symbol on the board**.
- The game **verifies the validity of the move**, **updates the board**, and **checks if the current player has achieved victory**.

If the current player has **won**, the game concludes, and the **winner is displayed**. If the game ends in a **draw**, the game concludes, and the **draw is displayed**. If neither of these conditions is met, the **game continues**, and the **player's turn changes**.

2.2.3 Programming Guide

1. Initialize the *OLED Display*
   - Free E-Book for OLED Display documentation
2. Initialize the *Keypad*
   - Free E-Book for Keypad documentation
3. Initialize the *Buzzer*
4. Initialize the *LEDs* (Green, Red, Yellow, Blue)
5. Inside the `setup()` function, initialize the *OLED Display*, *Keypad*, *Buzzer*, and *LEDs*.
6. Create an `enum` for different game states.
7. Create the following variables for the game:
   - startingGame - Melody, note-duration and length of the melody
   - player**0** - move-note, move-note-duration
   - player**1** - move-note, move-note-duration
   - win - Melody, note-duration and length of the melody
   - draw - Melody, note-duration and length of the melody
   - bool - if the win or draw melody should be played
   - int - current player, player**0** score, player**1** score, current move
   - char - board[3][3][2]
     - 3x3 board
     - Each field is " ", "X" or "O"
   - helper variables for the OLED Display for a more fluent gameplay - bool boardDrawn, bool setupGame, int counterLastMove, int counterBoardDrawnFinal
8. Create the different loops for the game, for the different game states.
9. Function to loop the Menu

10. Create a function to draw the Menu
11. Create a function to get the user input inside the menu
12. Function to loop the Games
13. Create a function to draw the TicTacToe board
14. Create a function to check if the current player has won or if the game ended in a draw
    - Check winner by checking each row, col and diagonal
    - Check if the board is full for draw
    - **Note:** Create one function for Winner checking and one function for Draw checking
    - If either of these conditions is met, the game concludes, the winner or draw is displayed and the game status changes
15. Create a function to get the user input inside the game
    - Get a number from the keypad
    - Check if the number is between 1 and 9
    - Check if the field is empty
    - Update the board (using a helper function)
    - Increase the number of moves
    - Play the move sound for the current player
    - Switch the current player
16. Create a function to update the board
17. Create a function to display the current standings
18. Create a function to play the melody
19. Create a function to draw the game over screen
20. Create a function to get the user input inside the game over screen
21. After the game reset everything except:
    - player**0** score, player**1** score
    - current player

2.3 Simon Says

SimonSays is a game where the player has to repeat a sequence of colors and sounds. The sequence starts with one color and its corresponding sound and gets longer each round by 1. The player has to repeat the sequence correctly to continue to the next round. If the player makes a mistake the game ends.

2.3.1 Additional components

For my implementation of the TicTacToe game I used the following additional components:

- LED
  - Green LED
  - Red LED
  - Yellow LED
  - (Blue LED)
- Buzzer

- OLED Display
- Keypad
- Resistors (220Ω)
- Diverse cables

You can see the circuit diagram in Figure 1 and the circuit blocks in Figure 2.

2.3.2 Code

The sketch operates in the following manner:

- Initially, you **initialize** the *OLED Display*, *Keypad*, *LEDs*, and *Buzzer* within the `setup()` function.
- Subsequently, you create various **variables** for the game, such as the `sequence`, the `player sequence`, the `current round`, and so on.

Firstly, you draw a **Menu screen** to introduce the user and prompt them to **press `1` to commence the game**. Upon pressing `1`, the game commences, the **GameStatus** undergoes a change, and the user is permitted to play. During the game:

- The game **displays** to “WATCH” and **plays the sequence**.
- The user has to **repeat the sequence** by **pressing the corresponding buttons** on the keypad, which are from 0 to 3.
- The game **verifies the correctness of the sequence**.
- If the sequence is correct, the game **goes into the round round** and randomly creates a new sequence, which is longer by 1.
- If the sequence is incorrect, the game **ends**, and the **correct sequence** will be displayed.

2.4 Programming Guide

2.4.1 Programming Guide

1. Initialize the *OLED Display*
   - Free E-Book for OLED Display documentation
2. Initialize the *Keypad*
   - Free E-Book for Keypad documentation
3. Initialize the *Buzzer*
4. Initialize the *LEDs* (Green, Red, Yellow, Blue)
5. Inside the `setup()` function, initialize the *OLED Display*, *Keypad*, *Buzzer*, and *LEDs*.
6. Create an `enum` for different game states.
7. Create the following variables for the game:
   - level - current level
   - charLightSequence[] - sequence of lights
   - Helper variables for the OLED Display for a more fluent gameplay - int counterShowLightShow, int counterAskForUserInput, int counterShowCorrect *(Those may not be necessary!)*
8. Create the different loops for the game, for the different game states.
9. Starting with the loopPhaseInstructions

- Display the name of the game
- Ask for any key to start the game
- Create a function to get the user input, to start the game

10. Create a function to loop lightShow
    - Write on the OLED Display "WATCH"
    - Wait for a short time, so WATCH is correctly displayed
    - Play the sequence of lights and corresponding sounds
      - Get a random number between 0 and 3
      - Store this number so the user input can be compared to it
      - Turn on the corresponding LED for a short time and play the corresponding sound
      - Delay between the lights and sounds
      - Change the game phase to user input
11. Create a function to loop user input
    - Write on the OLED Display "REPEAT" (or equivalent)
    - Wait for a short time, so REPEAT is correctly displayed
    - Get the user input
      - Get a number from the keypad
      - Check if the number is between 0 and 3, if not, ignore the input and wait for the next input **This is not an error!**
      - If the number is between 0 and 3
        - Check if the user input is correct
        - Do this by comparing the user input with the light sequence
        - Play the corresponding sound and turn on the corresponding LED for a short time
      - If all inputs are correct, change the game phase to correct
        - Show a success message
        - Wait a short time
        - Increase the level
        - Go back to the lightShow phase
      - If the input is incorrect, change the game phase to wrong input
        - Show a failure message
        - Show the correct sequence
        - Wait a short time
        - Ask for any key to start the game again

2.5 Any other game

Students are encouraged to come up with their own game ideas and implement them. The game should be simple and fun to play. The game should also be implemented using the Arduino Uno and the components mentioned above.

2.5.1 Additional components

Students are free to use any additional components they wish, but it's recommended to consult the teacher first so that the teacher can provide guidance and assistance if necessary.

3 Circuit Diagram

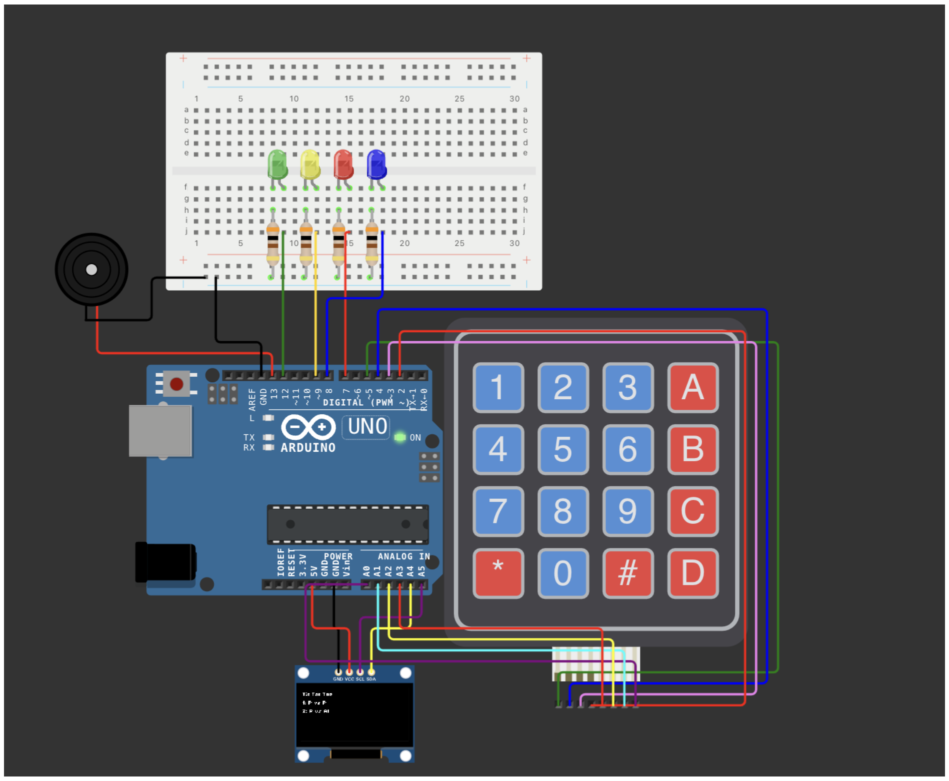

Figure 1: Circuit Diagram created using Wokwi

4 Circuit Blocks

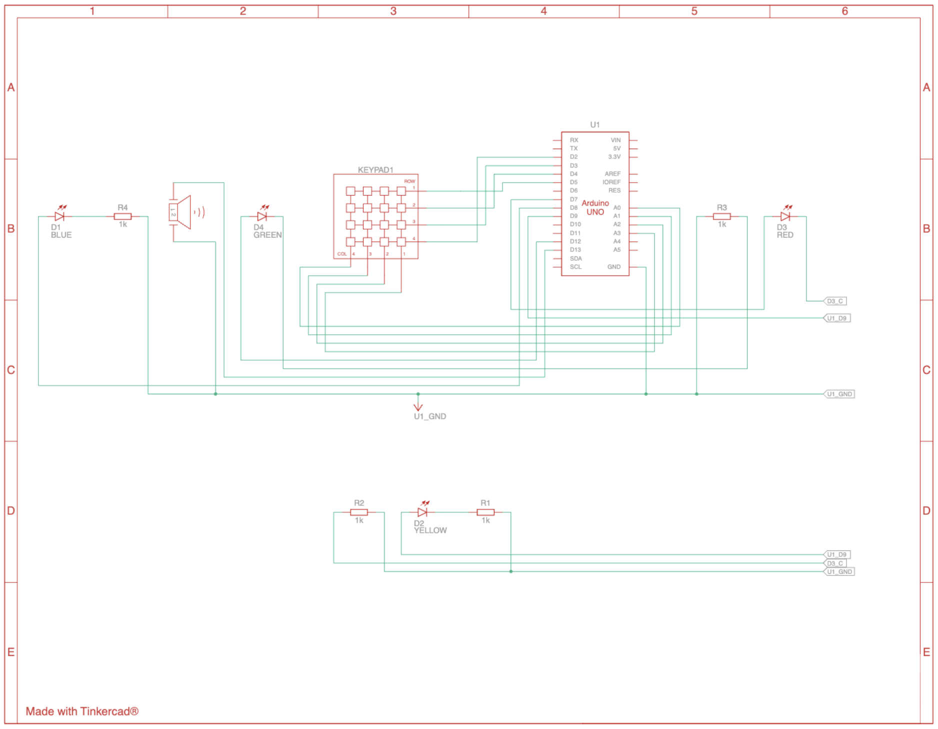

Figure 2: Circuit Blocks created using Tinkercad

Note: This is unfortunately not complete, because the OLED Display is not available in Tinkercad.