**Title: Development of an Arduino-Based Wire Loop Game with LCD Interface and Audio Feedback**

**Authors:**

**Ömer Dursun 2203452 Mehmet**

**Cengizhan Kınay 2102804**

**Abstract: This report details the development and implementation of an interactive wire loop game using Arduino. The system features an LCD display for user feedback, input buttons for two players, touch-sensitive gameplay mechanics, and real-time audio output that includes the iconic DOOM E1M1 melody. The design emphasizes user engagement through visual and auditory feedback, and evaluates performance based on time and error tracking.**

**1. Introduction**

**Wire loop games are classic skill-based games that challenge players to guide a metal loop through a twisted wire path without making contact. This project modernizes the concept by integrating an Arduino microcontroller, real-time scoring, dual player support, and audio feedback. The goal was to create a dynamic and entertaining system with a clear user interface and sound effects inspired by the DOOM video game series.**

**2. Materials and Methods**

**2.1 Components Used**

|  |  |  |
| --- | --- | --- |
| **Component** | **Quantity** | **Description** |
| **Arduino UNO** | **1** | **Microcontroller platform** |
| **16x2 LCD Display (LM016L)** | **1** | **Displays game status and scores** |
| **10kΩ Potentiometer** | **1** | **Controls LCD contrast** |
| **Resistors (220Ω–470Ω)** | **1** | **Limits current for LCD backlight and LEDs** |
| **Push Buttons** | **3** | **For player selection and game reset** |
| **Buzzer (active)** | **1** | **Outputs sounds and melodies** |
| **LEDs (Red & Green)** | **2** | **Visual indicators of status and touch events** |
| **Conductive wire track** | **1** | **Physical game interface** |

**2.2 Circuit Description**

* **The LCD is connected in 4-bit mode to Arduino digital pins 2–7.**
* **A 10kΩ potentiometer is used for contrast control on pin V0 of the LCD.**
* **The touch wire, start wire, and end wire are wired to digital pins 12, 10, and 11 respectively.**
* **Player 1, Player 2, and Reset buttons are connected to analog pins A2, A1, and A0 with internal pull-ups enabled.**
* **A buzzer is connected to pin 13 for audio output.**
* **Red and green LEDs are connected to pins 8 and 9 respectively.**

**2.3 Wiring Diagram**

**TinkerCad:**

**metin, diyagram, paralel, plan içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**Proteus:diyagram, taslak, teknik çizim, plan içeren bir resim

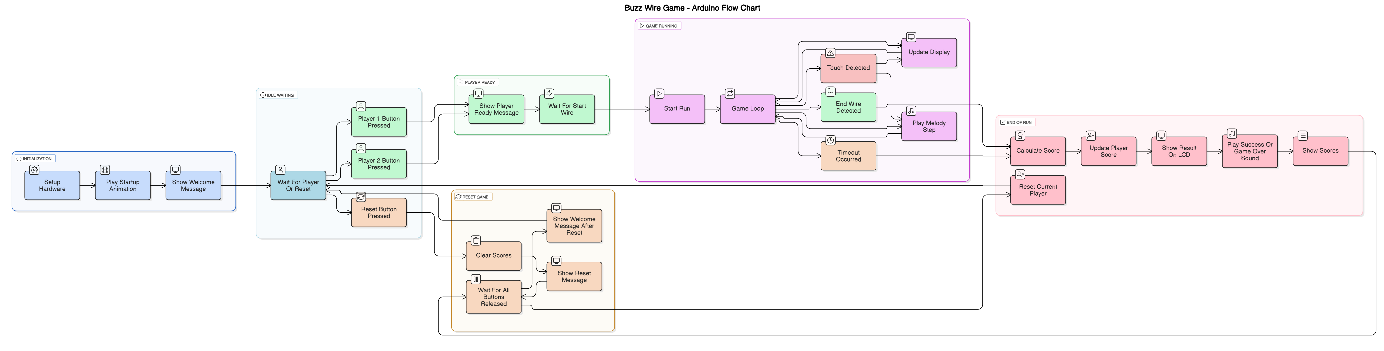
Yapay zeka tarafından oluşturulan içerik yanlış olabilir.TinkerCad Simulation**metin, ekran görüntüsü, diyagram, ekran, görüntüleme içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**To view the simulated breadboard circuit and functional connections, refer to the interactive Tinkercad project:**

**Wire Loop Buzz Game Simulation: https://www.tinkercad.com/things/9Tp2xygvUrM-wire-loop-buzz-game-with-scoreboard?sharecode=whqO1mfj3Hv7q8OD0pEfH\_AgG6kN3L8t2KjAp4YVpUQ**

**2.4 Code Flowchart**

******

**3. Software Design**

**3.1 Overview The Arduino sketch utilizes the LiquidCrystal library to manage the LCD. It employs a state machine to control game flow across idle, running, and finished states. Timers track gameplay duration and melody playback.**

**3.2 Key Functions Explained**

* **setup(): Initializes the pins, sets up the LCD, and shows a welcome message.**
* **loop(): Contains the main game logic. Waits for player input, handles gameplay, and determines outcomes.**
* **startRun(): Initializes game parameters like time and touch count, sets gameRunning to true, and starts the DOOM melody.**
* **handleTouch(): Increments the touch count, triggers the red LED, and plays a short buzzer tone.**
* **endRun(): Called when the player reaches the end or timeout. Calculates score, updates player stats, and shows feedback.**
* **resetGame(): Clears all player scores and resets the game state to initial.**
* **showScores(): Displays the accumulated scores of both players.**

**3.3 Scoring Algorithm**

**int calculateScore(unsigned long elapsedTime) {**

**int elapsedSeconds = elapsedTime / 1000;**

**int timePenalty = (elapsedSeconds > 20) ? (elapsedSeconds - 20) \* 2 : 0;**

**int penalty = touchCount \* penaltyPerTouch;**

**int score = maxScore - (timePenalty + penalty);**

**return score < 0 ? 0 : score;**

**}**

* **Every touch deducts 5 points.**
* **For every second past after 20seconds, 2 points are deducted.**
* **Maximum score is 100.**

**3.4 Audio Integration - DOOM E1M1 Melody**

**The melody from the DOOM game (E1M1: "At Doom's Gate") is recreated using the tone() function and a note matrix:**

**const int Notes[8][12] = { ... } // Frequency table across octaves**

* **Each note is played via playNote(octave, note, duration).**
* **playE1M1MelodyStep() runs one note at a time during the game loop.**
* **doomStep tracks which note is currently being played.**

**This implementation was inspired by the open-source DOOM music project on Arduino by VBStudio [4], where note frequencies and sequence structure for DOOM’s iconic riff were adapted to run using Arduino’s tone() and delay() functions. The adaptation avoids memory-heavy MIDI libraries and instead plays one note at a time in a loop, synchronized with game execution.**

**Supporting Functions:**

* **noteDoomBase(): Plays the recurring E-note pulse.**
* **playGameOverSound(): Plays a sequence when time runs out.**
* **playSuccessSound(): Signals a successful run.**

**4. Results and Discussion**

**The system successfully implements a playable two-player wire loop game. The LCD provides clear feedback, and the scoring system rewards both speed and precision. Touch events are reliably detected. The integrated DOOM music makes the game more immersive and engaging.**

**The system could be improved by adding:**

* **Persistent storage (e.g., SD card)**
* **Real-time score export via Bluetooth**
* **Difficulty modes or time settings**

**5. Conclusion**

**This project demonstrates a full integration of Arduino hardware and software for interactive gaming. By combining real-time inputs, visual feedback, and iconic music, it offers a compelling user experience. The code structure is modular and extensible, allowing future enhancements in gameplay complexity or connectivity.**

**6. License**

**This project is released under the MIT License.**

**MIT License**

**References**

1. **DOOM Wiki: E1M1 Hangar. https://doom.fandom.com/wiki/E1M1:\_Hangar\_(Doom)**
2. **Last Minute Engineers: LCD with Arduino. https://lastminuteengineers.com/arduino-lcd-tutorial/**
3. **Bobby Prince, Composer of DOOM (1993) Soundtrack**
4. **VBStudio. "Playing DOOM on an Arduino." https://vbstudio.hu/en/blog/20190330-Playing-DOOM-on-an-Arduino**