**T.C.**

**BAHÇEŞEHİR UNIVERSITY**

**FACULTY OF ENGINEERING AND NATURAL SCIENCES**

**DEPARTMENT OF COMPUTER ENGINEERING**

**PROJECT FINAL REPORT**

**BUZZ WIRE GAME**

**CMP3010 Course Project**

**Kaan Dağlıoğlu** **2202286**

**Deniz Özmen 2203032**

**İSTANBUL, MAY 2025**

# TABLE OF CONTENTS

TABLE OF CONTENTS (right-click and select "Update Field") ii

1 OVERVIEW 1

1.1 Problem Statement and Objectives 1

1.2 Background Information 1

2 METHODOLOGY 1

2.1 Project design 1

2.2 Project components 2

3 WORK PLAN 2

3.1 Tasks and Time Line 2

3.2 Cost Proposal 2

4 REFERENCES 4

5 PHOTOS 5

6 APPENDIX 6

# OVERVIEW

## Problem Statement and Objectives

The main goal of this project is to design and implement an interactive reflex game using Arduino. The game challenges a user's hand-eye coordination and reaction speed by requiring them to guide a metallic wand along a wire path without touching it. Any contact results in a penalty indicated through sound and light. The system tracks errors and elapsed time and includes checkpoints and a start/end detection system.

## Background Information

Games involving reflex and coordination are widely used for both entertainment and therapeutic purposes. This Arduino-based system provides a low-cost and effective way to train or test reflexes.

Electromechanical games using buzzers, LEDs, and timers have long been a standard method for teaching basic electronics, providing real-time feedback and encouraging user interaction. Arduino-based systems make such projects accessible and cost-effective, offering flexibility and ease of prototyping compared to traditional embedded systems [3].

# 2 METHODOLOGY

## 2.1 Project design

The project design revolves around a wire loop game logic. The player uses a wand to navigate a curved wire path. Touch sensors mark the start and end points. A button acts as a checkpoint. The system monitors performance through an Arduino microcontroller and provides feedback via buzzer, LED, and LCD.

A simplified flow of the project:

1. The game waits for the user to press the start touch sensor.
2. Once pressed, the timer begins and the game becomes active.
3. If the player touches the wire, an error is counted with sound/light alert.
4. Pressing the checkpoint button adds score.
5. A bonus button gives a larger score bonus.
6. Pressing the end sensor stops the timer and displays final score and time.

**Technical Challenges:**

* Debouncing signals from the metal wire.
* Proper state change detection for buttons.
* Managing LCD refresh without flickering.

**Specifications and Verification:**

* LCD shows real-time score and error count
* Buzzer and LED respond to wire contact
* Start/end sensors function correctly
* Button accurately detects checkpoint and bonus events

ekran görüntüsü, çizgi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

## 2.2 Project components

This project was developed using readily accessible electronics and fabrication resources, suitable for a typical university electronics lab or maker space.

**Facilities and Equipment Used:**

* **Electronics Lab:** Soldering station, multimeter, oscilloscope (for signal integrity testing), power supply
* **Prototyping Tools:** Breadboard, jumper wires, wire cutters, USB cables
* **PC Software:** Arduino IDE for programming, Tinkercad for schematic design

**Main Components:**

* **Arduino Uno:** Acts as the central controller, processing sensor input and driving output devices. Chosen for its ease of use, abundant resources, and large community support.
* **Active Buzzer:** Provides immediate auditory feedback upon contact with the wire.
* **Red LED:** Visual cue to accompany the buzzer during an error event.
* **16x2 I2C LCD Display:** Displays the game timer, error count, and score with minimal pin usage due to I2C communication.
* **TTP223 Capacitive Touch Sensors (x2):** Used for detecting the start and end of the game. These are reliable, easy to use, and require no mechanical parts, offering better durability compared to physical buttons.
* **Push Button (Checkpoint and Bonus):** Simple digital input used to simulate game mechanics such as bonus scoring and mid-way checkpoints.
* **Resistors (10kΩ pull-down, 220Ω for LED):** Used for stable and safe circuit operation.
* **Metallic Loop and Wand:** Forms the physical game interface. The wand is wired to act as a probe, and contact with the loop completes a circuit to register errors.

**Alternative Component Considerations:**

* **Raspberry Pi vs Arduino Uno:** A Raspberry Pi could allow for a more advanced graphical interface or internet connectivity, but would increase system complexity and cost. The Arduino Uno is ideal for simple real-time sensor applications.
* **Mechanical Push Buttons vs Capacitive Touch Sensors:** Mechanical buttons are more prone to wear and bounce issues. Capacitive sensors offer higher reliability for repeated use and require less physical force to operate.
* **OLED Display vs LCD:** OLEDs provide better contrast and flexibility in size, but a 16x2 LCD with I2C interface is more cost-effective and sufficient for textual feedback.

This mix of low-cost components and accessible lab equipment made the project feasible to build, test, and iterate on within a short time frame, while offering potential for future enhancements using more advanced platforms if needed.

# 3 WORK PLAN

## 3.1 Tasks and Time Line

The project consists of multiple sequential and interdependent tasks:

* The process began with research and planning, where similar wire loop reflex games were studied, and a detailed component list was created.
* After this, all required electronic components such as sensors, buzzer, and the Arduino Uno were sourced.
* The next phase involved wiring the components onto a breadboard and designing the circuit layout. This step included configuring the touch sensors and LCD.
* Following this, Arduino code was written to manage the sensors, detect errors, count points, and update the LCD screen accordingly.
* Once the code and circuit were in place, testing and integration were carried out to ensure all modules worked together as intended.
* Finally, performance testing, improvements, and documentation were completed.

Table . The project Gantt chart.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | WEEK 1 | | | | | | |
| TASK LIST | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **1. Research and Design** |  |  |  |  |  |  |  |
| 1.1 Research |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.2 Component Sourcing |  |  |  |  |  |  |  |
| **2. Construction and Coding** |  |  |  |  |  |  |  |
| 2.2 Circuit Design |  |  |  |  |  |  |  |
| 2.3 Arduino Proggramming |  |  |  |  |  |  |  |
| 2.4 Integration & Debugging |  |  |  |  |  |  |  |
| **3. Testing and Documentation** |  |  |  |  |  |  |  |
| 3.1 Testing and Evaluation |  |  |  |  |  |  |  |
| 3.2 Presentation |  |  |  |  |  |  |  |
| 3.3 Final Raport |  |  |  |  |  |  |  |

## 3.2 Cost Proposal

List the cost of each component and any other costs.

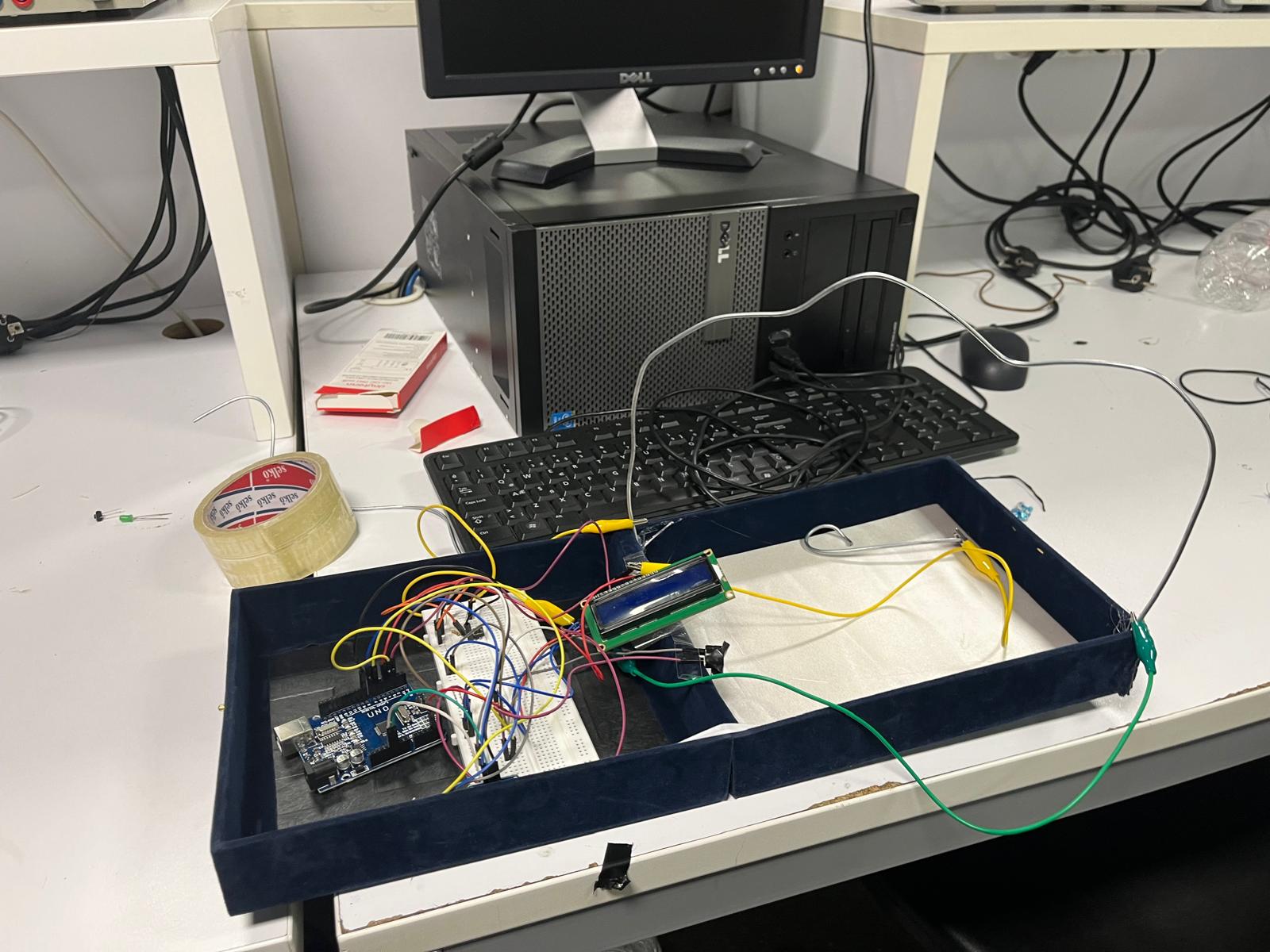
Table . Components and their estimated costs.

|  |  |
| --- | --- |
| **Arduino Uno** | **230** |
| Buzzer | 20 |
| Jumper cables and other materials | - |
| LCD with I2C | **120 + 30** |
| Touch Sensor (TTP 223) | 25 x 2 |
| Push Button | 15 |
| Resistors | **-** |
| Crocodile Cables | - |
| Breadboard | - |
| **TOTAL** | **465** |

# 4 REFERENCES

* Arduino.cc Documentation – <https://docs.arduino.cc/tutorials/uno-rev3/intro-to-board/>
* LiquidCrystal\_I2C Library Documentation – <https://github.com/johnrickman/LiquidCrystal_I2C>
* 16x2 LCD with I2C Interface Tutorial – <https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/>
* Wire Loop Reflex Game (PicoBricks Reference) – <https://docs.picobricks.com/tr/latest/projeler/buzz%20wire%20game.html>
* Playful Technology: Arduino Reflex Game Example – <https://www.youtube.com/watch?v=nG2siafMz_c&t=42s&ab_channel=PlayfulTechnology>
* TTP223 Touch Sensor Datasheet – <https://components101.com/sensors/ttp223-touch-sensor>
* Random Nerd Tutorials: Arduino I2C LCD Guide – <https://randomnerdtutorials.com/arduino-display-temperature-humidity-sensor-lcd/>
* CircuitDigest: Arduino Reflex Game Tutorial – <https://circuitdigest.com/microcontroller-projects/arduino-based-wire-loop-game>
* Tinkercad for Circuit Simulation and Schematic Design – <https://www.tinkercad.com/>

# 5 PHOTOS



# 6 APPENDIX

**Code Explanation:**  
 The Arduino code controls the reflex wire loop game by coordinating sensor inputs, output responses, score tracking, and user feedback. It begins by initializing hardware components including the buzzer, LED, LCD, and various input pins connected to touch sensors and buttons. When the game starts (triggered by touching the start sensor), the Arduino resets scores, errors, and timer values.

The game logic is implemented in the loop() function. This function continuously monitors for wire contact through the errorPin. When contact is detected, the buzzer sounds and LED blinks briefly, and an error is registered. To prevent the error from being counted repeatedly, the system waits for the wire to be released before continuing.

Two key interactive elements include a checkpoint button and a bonus button. The checkpoint button can only be activated once during a game, and adds 5 points to the score. The bonus button gives an additional 25 points and can be triggered any time during gameplay. Both buttons use state-tracking to detect the transition from HIGH to LOW, avoiding multiple triggers from the same press.

The system also tracks the game's progress. The game ends either when the player reaches 100 points (displaying a congratulatory message), or when they touch the end sensor, at which point the game stops and shows the final score and total time on the LCD screen. Throughout the game, the LCD provides live updates of elapsed time, current score, and error count.

The code is written to be responsive, efficient, and user-friendly. It avoids redundant LCD updates to prevent flickering and uses internal timers (via millis()) to track gameplay duration without pausing the system. This real-time handling ensures that user inputs and feedback are processed immediately, providing a smooth and interactive experience.

**FULL CODE:**

**#include <Wire.h>**

**#include <LiquidCrystal\_I2C.h>**

**LiquidCrystal\_I2C lcd(0x27, 16, 2);**

**const int errorPin = 2;**

**const int buzzerPin = 3;**

**const int ledPin = 4;**

**const int startPin = 6;**

**const int endPin = 7;**

**const int checkpointPin = 5;**

**const int bonusButtonPin = 8;**

**int score = 0;**

**int errors = 0;**

**bool checkpointPassed = false;**

**unsigned long startTime;**

**bool gameStarted = false;**

**bool shownReadyText = false;**

**bool bonusButtonPrevState = HIGH;**

**bool checkpointPrevState = HIGH;**

**void setup() {**

**Serial.begin(9600);**

**pinMode(errorPin, INPUT\_PULLUP);**

**pinMode(buzzerPin, OUTPUT);**

**pinMode(ledPin, OUTPUT);**

**pinMode(startPin, INPUT);**

**pinMode(endPin, INPUT);**

**pinMode(checkpointPin, INPUT\_PULLUP);**

**pinMode(bonusButtonPin, INPUT\_PULLUP);**

**lcd.init();**

**lcd.backlight();**

**lcd.setCursor(0, 0);**

**lcd.print("Refleks Oyunu");**

**delay(2000);**

**lcd.clear();**

**}**

**void loop() {**

**unsigned long elapsed = (millis() - startTime) / 1000;**

**if (!gameStarted && !shownReadyText) {**

**lcd.clear();**

**lcd.setCursor(0, 0);**

**lcd.print("Bas ve Basla");**

**shownReadyText = true;**

**}**

**if (!gameStarted && digitalRead(startPin) == HIGH) {**

**gameStarted = true;**

**shownReadyText = false;**

**score = 0;**

**errors = 0;**

**checkpointPassed = false;**

**startTime = millis();**

**bonusButtonPrevState = digitalRead(bonusButtonPin);**

**checkpointPrevState = digitalRead(checkpointPin);**

**lcd.clear();**

**lcd.setCursor(0, 0);**

**lcd.print("Oyun Basladi!");**

**delay(1000);**

**lcd.clear();**

**}**

**if (gameStarted && score >= 100) {**

**lcd.clear();**

**lcd.setCursor(0, 0);**

**lcd.print("Tebrikler!");**

**lcd.setCursor(0, 1);**

**lcd.print("Skor: ");**

**lcd.print(score);**

**delay(4000);**

**gameStarted = false;**

**lcd.clear();**

**return;**

**}**

**if (gameStarted && digitalRead(endPin) == HIGH) {**

**unsigned long totalTime = (millis() - startTime) / 1000;**

**lcd.clear();**

**lcd.setCursor(0, 0);**

**lcd.print("Oyun Bitti!");**

**lcd.setCursor(0, 1);**

**lcd.print("S:");**

**lcd.print(score);**

**lcd.print(" H:");**

**lcd.print(errors);**

**delay(3000);**

**lcd.clear();**

**lcd.setCursor(0, 0);**

**lcd.print("Sure: ");**

**lcd.print(totalTime);**

**lcd.print(" sn");**

**lcd.setCursor(0, 1);**

**lcd.print("Skor: ");**

**lcd.print(score);**

**delay(4000);**

**gameStarted = false;**

**lcd.clear();**

**return;**

**}**

**if (gameStarted) {**

**if (digitalRead(errorPin) == LOW) {**

**errors++;**

**digitalWrite(ledPin, HIGH);**

**tone(buzzerPin, 1000, 200);**

**delay(200);**

**digitalWrite(ledPin, LOW);**

**while (digitalRead(errorPin) == LOW);**

**}**

**bool checkpointState = digitalRead(checkpointPin);**

**if (!checkpointPassed && checkpointPrevState == HIGH && checkpointState == LOW) {**

**checkpointPassed = true;**

**score += 5;**

**lcd.clear();**

**lcd.setCursor(0, 0);**

**lcd.print("Checkpoint!");**

**lcd.setCursor(0, 1);**

**lcd.print("Skor +5");**

**delay(1500);**

**lcd.clear();**

**}**

**checkpointPrevState = checkpointState;**

**bool bonusState = digitalRead(bonusButtonPin);**

**if (bonusButtonPrevState == HIGH && bonusState == LOW) {**

**score += 25;**

**lcd.clear();**

**lcd.setCursor(0, 0);**

**lcd.print("BONUS!");**

**lcd.setCursor(0, 1);**

**lcd.print("Skor +25");**

**delay(1500);**

**lcd.clear();**

**}**

**bonusButtonPrevState = bonusState;**

**lcd.setCursor(0, 0);**

**lcd.print("Sure: ");**

**lcd.print((millis() - startTime) / 1000);**

**lcd.print(" sn ");**

**lcd.setCursor(0, 1);**

**lcd.print("P:");**

**lcd.print(score);**

**lcd.print(" H:");**

**lcd.print(errors);**

**}**

**delay(100);**

**}**