Design with microprocessors

project documentation: *One Tap Game*

Laboratory supervisor: Attila Fuzes

Student: Petruta George-Calin

Group: 30434

Table of Contents

[1. Introduction 3](#_Toc92494396)

[2. Components 3](#_Toc92494397)

[A. LCD Display 3](#_Toc92494398)

[B. 4-Buttons module 3](#_Toc92494399)

[C. Bluetooth module 3](#_Toc92494400)

[D. Serial monitor 4](#_Toc92494401)

[3. Application versions 4](#_Toc92494402)

[A. The first version 4](#_Toc92494403)

[B. The second version 5](#_Toc92494404)

[C. The third version 5](#_Toc92494405)

[4. Schematic 6](#_Toc92494406)

[5. Program flow 7](#_Toc92494407)

[a. Start running 7](#_Toc92494408)

[b. User presses a button 7](#_Toc92494409)

[c. The user has 3 misses 7](#_Toc92494410)

[d. The user presses the reset button 7](#_Toc92494411)

[e. Running in the background at all time 7](#_Toc92494412)

[6. Bibliography 8](#_Toc92494413)

[7. Code 8](#_Toc92494414)

# Introduction

„One tap game” is just as the name suggest a game where the user has to „tap” (press) a button in order to get as many points as possible.

The main reason for such an application is the fact that attention games are becoming more and more popular, expecially those who train the mind of the player to be more receptive of the information shown/heard/seen.

Also, as a final project for design with microprocessors a major part was learning how to use and put head to head components used during the laboratory works.

This application can be used simply to pass time or to actually work on one’s attention. Depending on the version of the application, the user has to be more and more attentive.

# Components

Because the application has gone through some modifications during the implementation, this documentation will only present the latest version of it: which is also the third one

## LCD Display

The LCD Display was a no-brain chose as the game needs a way to send the user informations about what led ***(the application does not use any leds, but we will imagine it uses 4 leds)*** is lit.

This was the best way to show the user the informations he needs.

It also serves the purpose of displaying the status of the game: still running, the score or that the game has ended.

## 4-Buttons module

In order to interact with the application, the user must have the ability to send informations back to it. In our application, only 2 buttons are used, one for reseting the game and the other one to send a signal to the application.

Both these buttons are used using interrupts.

## Bluetooth module

This module is used in order to display informations about the results from the latest rounds and also the best round.

The project needed a way to display information on a remote gadget so the chose was between the wi-fi module or the bluetooth module.

After some testing, the second version of the application (the one that used the wi-fi module) showed to be extremely slow and also could not correctly handle the button pressing.

This was the main reason why the application had to go through another modification and to be designed and implemented using the bluetooth module.

The only problem with this module is the fact that depending on the way it is used, the connection might be a little bit instable (or maybe I just got unlucky when choosing the module).

## Serial monitor

This is also a big part of the application, even though it is not a physical part of it.

On the serial monitor, every event in logged and displayed:

* When a led ***(imaginative)*** is lit
* When a button is hit
* The time when the led was lit
* The moment when a button was hit
* If when the button was pressed it was a hit (the number displayed was 1) or a miss (otherwise)
* The score
* The final results
* The reset action

# Application versions

The application went through a list of modifications:

## The first version

This version was made up with only a 4 buttons module, the LCD display and the serial monitor. In this version, there were 3 buttons to be pressed and you had to press the correct one in order to get the point, otherwise that would be considered a miss.

The games ends when the player has 3 such misses. The serial monitor had the same funcitonalities, displaying the internal state of the application, or as I should call it, the game status.

## The second version

This is the first version where there’s was a try at the external communication, this time through the WI-FI module.

This version had the same functional principle of the first version with the only difference being the fact this version would display the results on a webpage.

Due to the fact that the WI-FI module was extremely laggy and that it actually introduced a lot of unnecessary bugs, the idea was fast thrown away.

This bring us to the third and latest solution and possible the last one as well.

## The third version

The third version cames with a lot of differences:

* The usage of only two buttons, of which one is a reset
* The usage of bluetooth module
* The faster refreshing of the led ***(imaginative, as I have said before)*** to be lit (4 times faster)
* Now there are 4 numbers that can appear on the LCD (1..4), instead of only 3 in the older versions

As a little fun fact: this was also the shortest version because the bluetooth module was extremely easy to configure and that 2 buttons have been deleted from existence.

# Schematic

Graphical user interface, application

Description automatically generated

A screenshot of a computer screen

Description automatically generated with medium confidenceText

Description automatically generatedThe precedent photo represent the schematic done in wowki for my application. Because I could not find any bluetooth module there, the documentation will contain some photos of its functionality taken with a phone.

As shown in the photos, the user can see the latests result and the best result.

# Program flow

The application is always running (has no stop/start status), but depending on the internal state and the game status, there are some flows that can be determined out of its functionality.

## Start running

* The application is running
* The details about the score, the misses and which led it lit are displayed
* A timer is used as an **internal interrupt** to chose which led to be lit (randomly)

## User presses a button

* Depending on the button, there are certain possibilities:
  + Each button is implementing a function related to an external interrupt.
  + If the number displayed on the led is one, then the score gets increased
  + If the number is not one, then the number misses are increased

## The user has 3 misses

* The application is stopped, it is displayed a “Game over” message and the score
* Also, the bluetooth module will display information about the last rounds and the best result

## The user presses the reset button

* The moment the user presses this button, the game is reseted. If the game was not over, the game is simply put in its initial state where the score and the misses were both 0
* The game starts again
* This, once again is an external interrupt caused by the pressing of a button

## Running in the background at all time

* In the serial monitor every single modification on the internal state of the application is displayed
* The serial monitor contains all the information regarding the game status

# Bibliography

1. <https://users.utcluj.ro/~rdanescu/>
2. <https://users.utcluj.ro/~negrum/index.php/home/design-with-microprocessors/>
3. <https://wokwi.com/arduino/new?template=arduino-mega>
4. <https://www.arduino.cc/>
5. <https://create.arduino.cc/projecthub/imjeffparedes/add-wifi-to-arduino-uno-663b9e>

# Code

#include <LiquidCrystal.h>

#include <TimerOne.h>

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

volatile int ledNumber = 0;

volatile int badHits = 0;

volatile int timeAtBlink = 0;

volatile int score = 0;

volatile int gameOver = 0;

volatile int finalScore = -1;

volatile int printedResult = 0;

int result[2000];

int counter;

void setup()

{

lcd.begin(16, 2);

counter = 0;

pinMode(20 , INPUT);

pinMode(21 , INPUT);

digitalWrite(20, HIGH);

digitalWrite(21, HIGH);

attachInterrupt(digitalPinToInterrupt(20), functieLed1, FALLING);

attachInterrupt(digitalPinToInterrupt(21), restartJoc, FALLING);

Timer1.initialize(500000);

Timer1.attachInterrupt(displayLed);

printLCDInitialStatus();

Serial.begin(9600);

Serial1.begin(9600);

Serial.println("The game has started!");

pinMode(LED\_BUILTIN, OUTPUT);

digitalWrite(LED\_BUILTIN, LOW);

}

void printLCDInitialStatus()

{

lcd.setCursor(0, 0);

lcd.print("Score:");

lcd.setCursor(7, 1);

lcd.print("Bad hit:");

lcd.setCursor(0, 1);

lcd.print("led:");

}

void loop()

{

if (badHits >= 3)

{

Timer1.stop();

gameOver = 1;

if (finalScore == -1)

{

finalScore = score;

}

if (printedResult == 0)

{

Serial.println("The game is over!");

Serial.print("The score: ");

Serial.println(finalScore);

result[++counter] = finalScore;

printedResult = 1;

int maxim = -1;

for (int i = 1; i <= counter; i++)

{

if (result[i] > maxim)

{

maxim = result[i];

}

}

Serial1.write("The best score is: \n");

Serial1.println(maxim);

Serial1.write("The last results are: \n");

for (int i = counter; i >= 1; i--)

{

Serial1.println(result[i]);

}

Serial1.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Game over!");

lcd.setCursor(0, 1);

lcd.print("Score: ");

lcd.setCursor(7, 1);

lcd.print(finalScore);

delay(1000);

}

}

void functieLed1()

{

Serial.println();

int badHit = 0;

if (gameOver == 0)

{

int timeAtHit = millis();

Serial.print("Time at hit: ");

Serial.print(timeAtHit);

Serial.print(", time at blink: ");

Serial.print(timeAtBlink);

Serial.println();

Serial.println("The button 1 was pressed!");

if ((timeAtHit - timeAtBlink) / 1000 > 1)

{

badHit = 1;

badHits++;

}

else

{

if (ledNumber != 1)

{

badHit = 1;

badHits++;

}

else

{

score++;

}

}

}

if (badHit)

{

Serial.print("That was a bad hit: \nThe active led: ");

Serial.print(ledNumber);

Serial.print(", Button pressed: ");

Serial.println("1");

}

else

{

Serial.print("That was a good hit: \nThe active led: ");

Serial.print(ledNumber);

Serial.print(", Button pressed: ");

Serial.println("1");

}

Serial.println();

}

void restartJoc()

{

Serial.print("\n\n\n");

Serial.println("The game was restarted");

Timer1.start();

lcd.clear();

badHits = 0;

score = 0;

gameOver = 0;

finalScore = -1;

printedResult = 0;

printLCDInitialStatus();

}

void displayLed()

{

ledNumber = random(4) + 1;

Serial.print("The led ");

Serial.print(ledNumber);

Serial.println(" was activated");

lcd.setCursor(4, 1);

lcd.print(ledNumber);

lcd.setCursor(15, 1);

lcd.print(badHits);

lcd.setCursor(6, 0);

lcd.print(score);

timeAtBlink = millis();

}