Arduino ERASMUS Project
Memory Game

Margout Camille

11 January 2022

Abstract

In this report, we will explain and detail how my Memory Game works. This projects have been prepared using Arduino UNO board.

Contents

Introduction **2**

1 Software and Hardware tools **3**
1.1 Main Modules . . . 3
1.2 Set Up . . . 3

2 Project Specifications - Memory game **4**
2.1 Global Idea of the project . . . 4
2.2 Libraries . . . 4
2.3 Code and Comments . . . 4
2.3.1 Variables . . . 5
2.3.2 Function Definition . . . 6
2.3.3 Pattern Blink Led . . . 9
2.3.4 Level Definition . . . 12
2.3.4.1 Level 1 . . . 12
2.3.5 Main Code . . . 16

3 Limits and Amelioration axes **21**
3.1 Limits . . . 21
3.2 Amelioration Axes . . . 21
3.2.1 Code lenght . . . 21
3.2.2 UART return . . . 21
3.2.3 Use of interrupt . . . 21

Conclusion **22**
Annexes23

Introduction

As Erasmus student at Politechnika Krakowska, I had to work on a project which could show a global work on Arduino board using Soleber workspace.

I have a personal reason to make a memory game because my grandmother is suffering from Alzaimer desease. I wanted to propose a game that help people to pratice their memory.

I also always ever like to creat game using coding tools. I pratice regurlaly project on Arduino boards and I created a "life game" in ADA language.

And today, this report is dedicated to the descritpion of "Memory Game". We will present the code, the libraries and the way of building this project.

Special thank to Paweł Król, my Arduino Prototyping teacher, who helped me during this project.

1 Software and Hardware tools

1.1 Main Modules

Figure 1: Hardware materials - VAVT1615C AVTduino LCD

I used ATMEGA328 board to make this game. Thanks to another chip, I created the memory game using buttons and LDC screen.

The main microcontroller was an Arduino UNO chip.

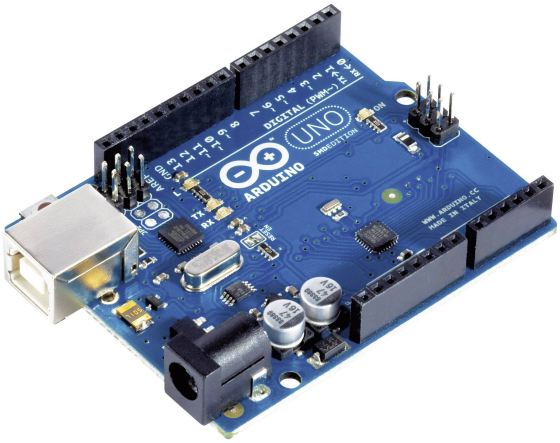

Figure 2: Hardware materials - ARDUINO UNO

1.2 Set Up

To make our Memory Game work, you have to download Sloeber software and upload $Memory_Game.cpp$ code. If your chip have already the software implemented, make sure that you power it thanks to USB cable or 5V power battery.

2 Project Specifications - Memory game

2.1 Global Idea of the project

The idea of this project is to present a memory game using Arduino UNO board.

Using the four buttons, the aim is to reproduce the blinking led pattern.

You have 3 lives and you will pass the levels if you reproduce perfectly the light pattern. If you are wrong, you lost a life. In the case of your life level is equal to 0 it is a game over.

2.2 Libraries

For memory Game project, I used this libraries :

$include"Memory_Game.h"$
$include < stdio.h >$
$include < math.h >$
$include < avr/io.h >$
$include"hd44780/HD44780.hpp"$
$include < EEPROM.h >$

```
// Do not remove the include below
#include "Memory_Game.h"
#include <stdio.h>
#include <math.h>
#include <avr/io.h>
#include "hd44780/HD44780.hpp"
#include <EEPROM.h>
```

Figure 3: Libraries

I used EEPROM library because I wanted to memorise the best score ever played in the chip even if the player reset it.

2.3 Code and Comments

I coded 10 levels of difficulty. I wanted to give 5 life to the player and if the life is equal to 0, there is a game over and the interface restart to level 1.

In each level, the player seen a blinking led pattern and have to reproduce it thanks to the four button on the right.

2.3.1 Variables

```
int i ;

#define LED1 PB5    //  LED1 definition
#define LED2 PB4    //  LED2 definition
#define LED3 PB3    //  LED2 definition
#define LED4 PB2    //  LED2 definition
#define SW1 PD0     //  BUTTON definition
#define SW2 PD1     //  BUTTON definition
#define SW3 PD2     //  BUTTON definition
#define SW4 PD3     //  BUTTON definition
#define ADCIN PC0   //  ADCIN

void ADC_Init(void);
char str[12];
int level = 1 ;
int life_nbr_level ;
int score_level ;
int best_score_ever = 0 ;
```

Figure 4: Variables

```
struct Player {
    int vie ;
    int score ;

} typedef Player ;
```

Figure 5: Player Structure

I used a player structure to have in a same time, a score and a level.

2.3.2 Function Definition

```
void start_in (int time){

    while (time > 0) {
        sprintf(str, "Start in : %d s", time);
        LCD_GoTo(1,1);
        LCD_WriteText(str);
        _delay_ms(1000);
        time = time - 1 ;

    }

    if (time == 0){
        LCD_Clear();
        LCD_GoTo(3,0);
        LCD_WriteText("Your turn");
        LCD_GoTo(6,1);
        LCD_WriteText("GO !");
    }

}
```

Figure 6: Start In

This function is to give time to the player while the code is running. It is like a ready ? Go !

```
int Minus_life (int life_nbr) {

    int new_life_score = life_nbr - 1 ;
    return new_life_score ;
}
```

Figure 7: Minus Life

This very specific function is very use-full when the user make a mistake. If yes, the function decrease the number of life of the player.

```
void Print_Game_Over_Life (){

    LCD_Clear();
    LCD_GoTo(3,0);
    LCD_WriteText("GAME OVER");

}
```

Figure 8: Game Over Life

If the number of life is equal to 0, then it is a Game Over.

```
void Print_Score (int myscore){

    LCD_GoTo(4,0);
    sprintf(str, "Score: %d", myscore);
    LCD_WriteText(str);
    _delay_ms(1000) ;

}
```

Figure 9: Print Score

As same as print life, print score help the player to know the actual score.

```
int Print_Time_Out (int life_lvl){

        LCD_Clear();
        LCD_GoTo(3,0);
        LCD_WriteText("Time Out");

        int penality = Minus_life(life_lvl) ;
        return penality ;

}

int Timer_to_play (int time) {

    int penality = 0 ;

        while ((time > 0)) {
            _delay_ms(1000);
            time = time - 1 ;
            }

        if (time == 0){
            return penality ;

        }
}
```

Figure 10: Timer

I wanted to use timer to manage playing time. For example, if the player do not touch any button during 10 second, then he would loose a life. But I did not have the time for that. I will implement it after.

```
void Your_turn (){

    LCD_Clear();
    LCD_GoTo(4,0);
    LCD_WriteText("Your Turn");
    LCD_GoTo(1,1);
    LCD_WriteText("S1=LED4 S4=LED1");
    _delay_ms(100) ;
}

void My_turn (){

    LCD_Clear();
    LCD_GoTo(5,0);
    LCD_WriteText("My Turn");
    LCD_GoTo(1,1);
    LCD_WriteText("S1=LED4 S4=LED1");
    _delay_ms(1000) ;
}
```

Figure 11: Your and My Turn

MyTurn is the time when the microcontroller shows the pattern and YourTurn is the time for the player to reproduce the pattern.

```
void EEPROM_best_score (int ascore) {

    int value ;
    EEPROM.get(0,value) ;

    if (ascore > value){
        EEPROM.put(0, ascore);
    }
}
```

Figure 12: EEPROM Best Score

I was very proud of this function that worked perfectly. I wanted to give the best score ever done even if the player reset the board. I succeeded using this function and another call after in the code.

2.3.3 Pattern Blink Led

These functions were used to make the led blinking according to each level. In a next step, I will use random blinking and more and more levels.

```
void pattern_blink_led1 (int* tab) {


    _delay_ms(1000);
    PORTB ^= (1 << PORTB2);
    _delay_ms(700);
    PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
    _delay_ms(700);


}


void pattern_blink_led2 (int* tab) {


    pattern_blink_led1(light_partern1) ;
    PORTB ^= (1 << PORTB3);
    _delay_ms(700);
    PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
    _delay_ms(700);


}
void pattern_blink_led3 (int* tab) {


    pattern_blink_led2(light_partern2) ;
    PORTB ^= (1 << PORTB5);
    _delay_ms(700);
    PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
    _delay_ms(700);


}
```

Figure 13: Pattern of level 1, 2 and 3

```
void pattern_blink_led4 (int* tab) {


    pattern_blink_led3(light_partern3) ;
    PORTB ^= (1 << PORTB5);
    _delay_ms(700);
    PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
    _delay_ms(700);

}



void pattern_blink_led5 (int* tab) {


    pattern_blink_led4(light_partern4) ;
    PORTB ^= (1 << PORTB4);
    _delay_ms(700);
    PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
    _delay_ms(700);


}
void pattern_blink_led6 (int* tab) {


    pattern_blink_led3(light_partern3) ;
    pattern_blink_led3(light_partern3) ;
    _delay_ms(700);

}
```

Figure 14: Pattern of level 4, 5 and 6

```
void pattern_blink_led7 () {


    pattern_blink_led4(light_partern4) ;
    pattern_blink_led3(light_partern3) ;
    _delay_ms(700);

}
void pattern_blink_led8 () {


    pattern_blink_led2(light_partern1) ;
    pattern_blink_led5(light_partern5) ;
    pattern_blink_led1(light_partern1) ;
    _delay_ms(700);

}
void pattern_blink_led9 () {


    pattern_blink_led1(light_partern1) ;
    pattern_blink_led8() ;
    _delay_ms(700);

}
```

Figure 15: Pattern of level 7, 8 and 9

```
void pattern_blink_led10 () {


    pattern_blink_led2(light_partern2) ;
    pattern_blink_led3(light_partern3) ;
    pattern_blink_led5(light_partern5) ;
    _delay_ms(700);

}
```

Figure 16: Pattern of level 10

I did not used random blinking but I will implement it in a futur version of this code.

2.3.4 Level Definition

2.3.4.1 Level 1

```
Player Level_1 (int timer_select, int life_nbr, int score){

    Player player_level1 ;

    //int level1_life_level = life_nbr;

    bool status = true ;
    // int time_status = Timer_to_play(timer_select);
    int counter = 0;

    // blinking led template
    My_turn();
    pattern_blink_led1(light_partern1) ;
    _delay_ms(1000) ;
    Your_turn();
    _delay_ms(500) ;
    LCD_Clear();
    LCD_GoTo(5,0);
    LCD_WriteText("LEVEL 1");
    Print_Life(life_nbr_level) ;



    // your turn

while ( (counter <=0) and (life_nbr >  0) ) {

    // check pattern
    // To check a level, you have to increment a counter to 3, 5, 8, 10.. depending on
    // light_parternX size

    // we know the size of level 1 : 3
    // user should press button S1 -> S2 -> S4



        if (!(PIND & 1<<PIND3) and counter == 0) {

            PORTB ^= (1 << PORTB2);
            _delay_ms(1000);
            PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

            counter ++ ;
            LCD_Clear();
            LCD_GoTo(5,0);
            LCD_WriteText("LEVEL 1");
            LCD_GoTo(7,1);
            LCD_WriteText("*");
            _delay_ms(100) ;
            LCD_Clear();
            LCD_GoTo(0,0);
            LCD_WriteText("LEVEL 1 - Success");
            _delay_ms(1500) ;
            score++ ;
        }

        else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter == 0){
            life_nbr = Minus_life(life_nbr) ;
            _delay_ms(500) ;
        }
```

Figure 17: Code - Level 1

```
                if (score == 1){

                    counter++ ;
                }



    }


    LCD_Clear();
    Print_Score(score);
    Print_Life(life_nbr) ;
    _delay_ms(1000);

    player_level1.vie = life_nbr ;
    player_level1.score = score ;

    return player_level1 ;


}
```

Figure 18: [END] Code - Level 1

All the level function are belt quite the same. We first make the pattern blinking and then, give the player the opportunity the reproduce it. If he succeed, we save the score and the life level in player structure. I used LCDWriteText to print the Life level, score level or Game over in case of Game Over.

I give you picture of level 1 and all the code is given after.

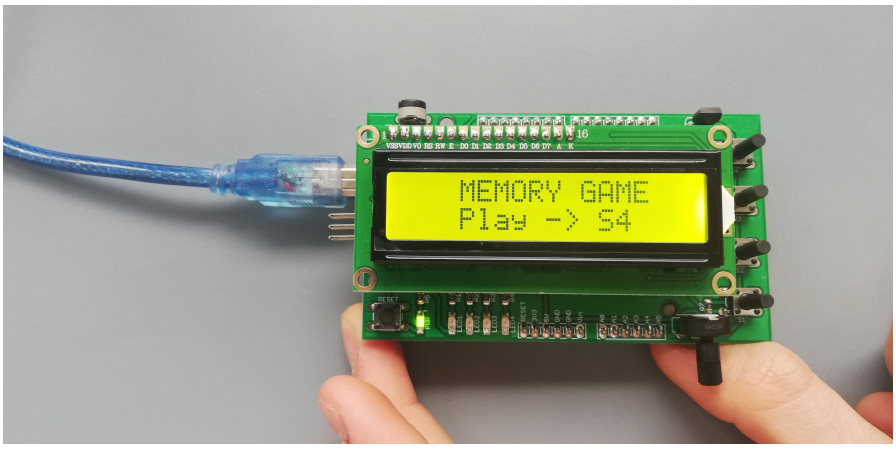

Figure 19: Overview - Level 1

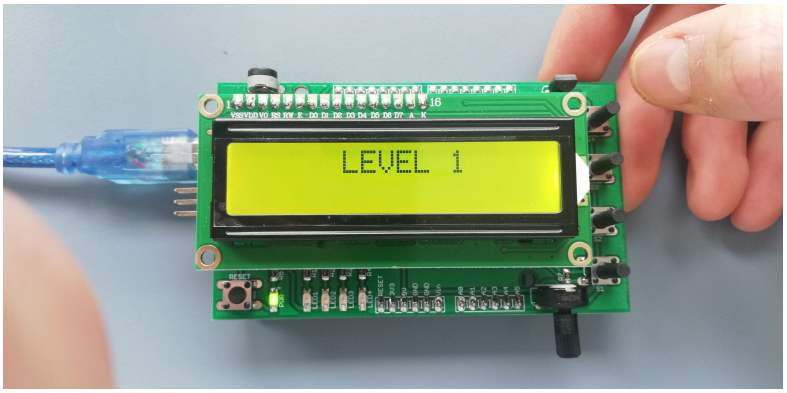

Figure 20: Overview - Level 1

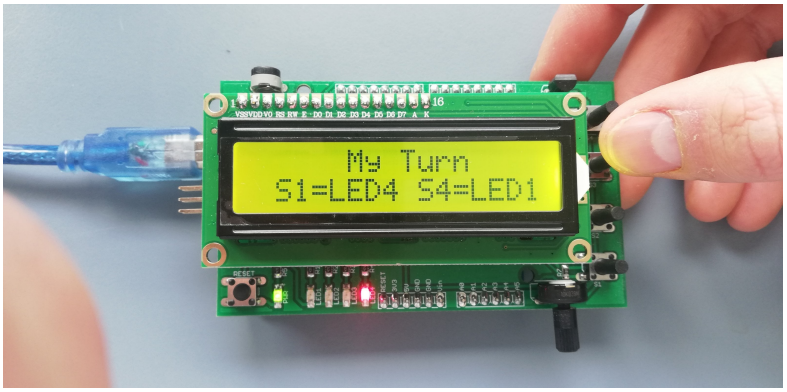

Figure 21: Overview - Level 1

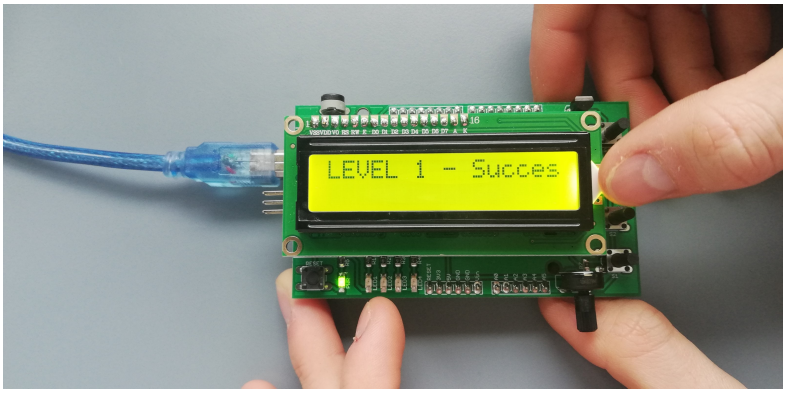

Figure 22: Overview - Level 1

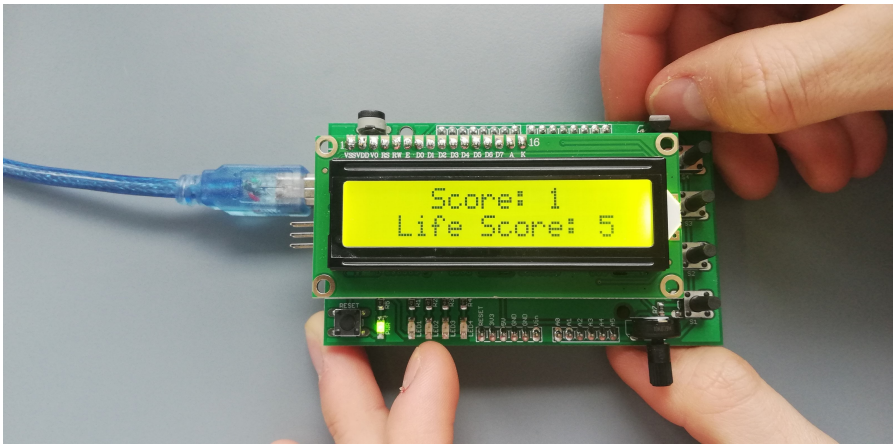

Figure 23: Overview - Level 1

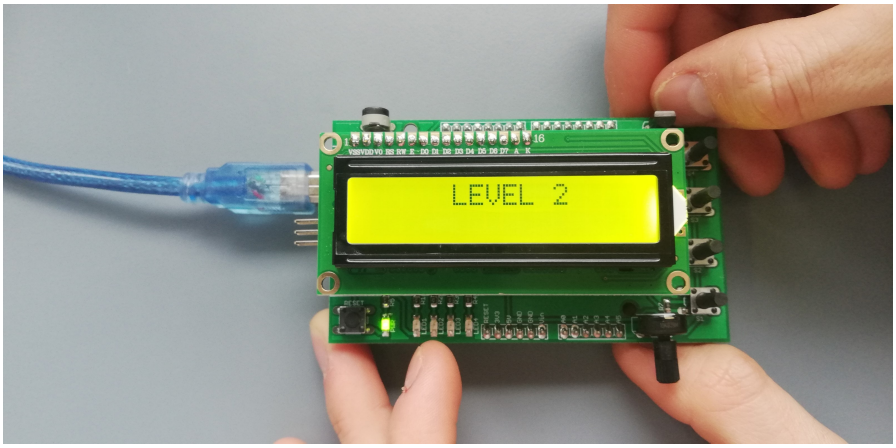

Figure 24: Overview - Access to Level 2

2.3.5 Main Code

In the main code, the player in invited to press S4 button to start the game. If he press the wrong button, then a timer of 5 seconds is set and we have another time a choice of pressing S4.

Then, the player have a first information : the best score even done on the board.

The Memory game is beginning, level by level.

```
int main(void)
{



    // IO configuration
    DDRB  |= (1<<LED1) | (1<<LED2) | (1<<LED3) | (1<<LED4);     // configuration of LED pins as outputs
    DDRC &=~ (1<<ADCIN);             // ADC pin as input

    // leds controller
    DDRB |= ((1 << DDB2) | (1 << DDB3) | (1 << DDB4) | (1 << DDB5));  // Direction of port line (1 - output)
    PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5);   // Output register (1 - do not light)

    // buttons controller
    DDRD &= ~(1<<DDD0) | ~(1<<DDD1) | ~(1<<DDD2) | ~(1<<DDD3);        // Direction of port line (0 - input)


    // screen initialization
        LCD_Initalize();


bool status_game = true  ;


    EEPROM.get(0,best_score_ever) ;
    Player play1 ;
```

Figure 25: Main Code - Initialization

```
while (1) {

    life_nbr_level = 5 ;
    score_level = 0 ;

while ((PIND & 1<<PIND0)){


    LCD_Clear();
    LCD_GoTo(3,0);
    LCD_WriteText("MEMORY GAME");
    LCD_GoTo(3,1);
    LCD_WriteText("Play -> S4");
    _delay_ms(500);

    if (!(PIND & 1<<PIND1) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND3)){
        LCD_Clear();
        LCD_GoTo(0,0);
        LCD_WriteText("Waiting for user");
        start_in(5) ;

        }

    }

if (!(PIND & 1<<PIND0)){

    LCD_Clear();
    LCD_GoTo(2,0);
    LCD_WriteText("Best Score");
    LCD_GoTo(2,1);
    sprintf(str, "%d", best_score_ever);
    LCD_WriteText(str);
    _delay_ms(1500) ;
```

Figure 26: Main Code - Start and EEPROM Best Score Display

```
while (score_level >= 0){

    if (score_level == 0) {


        LCD_Clear();
        LCD_GoTo(5,0);
        LCD_WriteText("LEVEL 1");
        _delay_ms(2000) ;
        //start_in(3) ;
        play1 = Level_1(4000, life_nbr_level, score_level) ;

        life_nbr_level = play1.vie ;
        score_level = play1.score ;
        EEPROM_best_score(score_level) ;


        if (life_nbr_level <= 0){
            score_level = -1 ;
        }

    }
```

Figure 27: Main Code - Level Display and Game

```
if (score_level == 1) {

// enter level 2

    LCD_Clear();
    LCD_GoTo(5,0);
    LCD_WriteText("LEVEL 2");
    _delay_ms(2000) ;
    //start_in(3) ;
    play1 = Level_2(4000, life_nbr_level, score_level) ;

    life_nbr_level = play1.vie ;
    score_level = play1.score ;
    EEPROM_best_score(score_level) ;

    if (life_nbr_level <= 0){
        score_level = -1 ;
    }
}
```

Figure 28: Main Code - Level Display and Game

```
if (score_level == 2) {

        LCD_Clear();
        LCD_GoTo(5,0);
        LCD_WriteText("LEVEL 3");
        _delay_ms(2000) ;
        //start_in(3) ;
        play1 = Level_3(4000, life_nbr_level, score_level) ;

        life_nbr_level = play1.vie ;
        score_level = play1.score ;
        EEPROM_best_score(score_level) ;

        if (life_nbr_level <= 0){
            score_level = -1 ;

        }

    }
```

Figure 29: Main Code - Level Display and Game

You can find the other level in Annexe A. The call is the same but we change the conditions to access or not to the next level.

```
if (score_level == 9) {

    LCD_Clear();
    LCD_GoTo(5,0);
    LCD_WriteText("LEVEL 10");
    _delay_ms(2000) ;
    //start_in(3) ;
    play1 = Level_10(4000, life_nbr_level, score_level) ;

    life_nbr_level = play1.vie ;
    score_level = play1.score ;
    EEPROM_best_score(score_level) ;

    if (life_nbr_level <= 0){
        score_level = -1 ;
    }


}

if (score_level == 10) {

    LCD_Clear();
    LCD_GoTo(4,0);
    LCD_WriteText("SUCCESS !");
    LCD_GoTo(0,1);
    LCD_WriteText("CONGRATULATION");
    _delay_ms(2000) ;
    score_level = 0 ;


}
```

Figure 30: Main Code - Success Level 10

3 Limits and Amelioration axes

3.1 Limits

The first limit I see on my project is the easy way to code and so, the easy way to play. I did not used random function to make the led blinking. The consequence is that if the player play five times, he or she will know the patterns by heart and will not play.

I wish to focus me again on this project and that is why I bought the chip too. I would like to go further.

3.2 Amelioration Axes

3.2.1 Code lenght

I did the project very fast at the beginning but the mistake was the I did not used beautiful function to easily code the function.

So the amelioration could be that I could use more featured function.

3.2.2 UART return

I also could used UART protocol to for example, send the score to the computer. I did not want to implement that because I have the habit to power the board with battery and not connect it to the computer.

3.2.3 Use of interrupt

The next amelioration axe will be the use of interrupt. I was afraid of using them because some of them are reset by the button so it was complicated and I left the idea very fast. But I will do another project using it I think.

Conclusion

The conclusion of this project is very positive for me because it helped me to better understand the use of register and pins of ATMEGA328p.

It was an ambitious project and I can see its limits today but I think I will implement and improve it in a next future. I spend many many hours on this project. I really enjoyed it, so, thank you for your support and your help.

It was really important for me because the main motivation was to think about how help people with memory disease.

The most important conclusion of this project is that it is not finished. I can see more and more implementation to do but I wanted to have a working game more than a beautiful code. I will do many modification because I really like Arduino and I practice it in my free time. So, this is not an end and it have been a please to work with registers. I did not know neither Sloeber work space so thank you, and, I hope to give you a version 2 !

Annexes

Annexe A Memory Game - Main Code

```
/*
*
* CAMILLE MARGOUT
* ARDUINO PROJECT
*   MEMORY GAME
*    11/01/2022
*
*
*/



// Do not remove the include below
#include "Memory_Game.h"
#include <stdio.h>
#include <math.h>
#include <avr/io.h>
#include "hd44780/HD44780.hpp"
#include <EEPROM.h>


int i ;

#define LED1 PB5     //  LED1 definition
#define LED2 PB4     //  LED2 definition
#define LED3 PB3     //  LED2 definition
#define LED4 PB2     //  LED2 definition
#define SW1 PD0          //  BUTTON definition
#define SW2 PD1          //  BUTTON definition
#define SW3 PD2          //  BUTTON definition
#define SW4 PD3          //  BUTTON definition
#define ADCIN PC0    //  ADCIN

void ADC_Init(void);
char str[12];
int level = 1 ;
int life_nbr_level ;
int score_level ;
int best_score_ever = 0 ;

struct Player {
int vie ;
int score ;
```

```
} typedef Player ;


int light_partern1[1] = {PORTB2} ;
int light_partern2[2] = {PORTB2,PORTB3} ;
int light_partern3[3] = {PORTB2,PORTB3, PORTB5} ;
int light_partern4[4] = {PORTB2,PORTB3, PORTB5, PORTB5} ;
int light_partern5[5] = {PORTB2,PORTB3, PORTB5, PORTB5, PORTB4} ;

void start_in (int time){

while (time > 0) {
sprintf(str, "Start in : %d s", time);
LCD_GoTo(1,1);
LCD_WriteText(str);
_delay_ms(1000);
time = time - 1 ;

}

if (time == 0){
LCD_Clear();
LCD_GoTo(3,0);
LCD_WriteText("Your turn");
LCD_GoTo(6,1);
LCD_WriteText("GO !");
}

}


int Minus_life (int life_nbr) {

int new_life_score = life_nbr - 1 ;
return new_life_score ;
}


void Print_Game_Over_Life (){

LCD_Clear();
LCD_GoTo(3,0);
LCD_WriteText("GAME OVER");

}


void Print_Life (int life_nbr){
```

```
LCD_GoTo(2,1);
sprintf(str, "Life Score: %d", life_nbr);
LCD_WriteText(str);
_delay_ms(1000) ;


}

void Print_Score (int myscore){

LCD_GoTo(4,0);
sprintf(str, "Score: %d", myscore);
LCD_WriteText(str);
_delay_ms(1000) ;

}

int Print_Time_Out (int life_lvl){

LCD_Clear();
LCD_GoTo(3,0);
LCD_WriteText("Time Out");

int penality = Minus_life(life_lvl) ;
return penality ;

}

int Timer_to_play (int time) {

int penality = 0 ;

while ((time > 0)) {
_delay_ms(1000);
time = time - 1 ;
}

if (time == 0){
return penality ;

}
}



void Your_turn (){

LCD_Clear();
LCD_GoTo(4,0);
```

```
LCD_WriteText("Your Turn");
LCD_GoTo(1,1);
LCD_WriteText("S1=LED4 S4=LED1");
_delay_ms(100) ;
}

void My_turn (){

LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("My Turn");
LCD_GoTo(1,1);
LCD_WriteText("S1=LED4 S4=LED1");
_delay_ms(1000) ;
}


void EEPROM_best_score (int ascore) {

int value ;
EEPROM.get(0,value) ;

if (ascore > value){
EEPROM.put(0, ascore);
}
}


void pattern_blink_led1 (int* tab) {


_delay_ms(1000);
PORTB ^= (1 << PORTB2);
_delay_ms(700);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
_delay_ms(700);


}


void pattern_blink_led2 (int* tab) {


pattern_blink_led1(light_partern1) ;
PORTB ^= (1 << PORTB3);
_delay_ms(700);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
_delay_ms(700);
```

```
}

void pattern_blink_led3 (int* tab) {


pattern_blink_led2(light_partern2) ;
PORTB ^= (1 << PORTB5);
_delay_ms(700);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
_delay_ms(700);


}



void pattern_blink_led4 (int* tab) {


pattern_blink_led3(light_partern3) ;
PORTB ^= (1 << PORTB5);
_delay_ms(700);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
_delay_ms(700);

}



void pattern_blink_led5 (int* tab) {


pattern_blink_led4(light_partern4) ;
PORTB ^= (1 << PORTB4);
_delay_ms(700);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
_delay_ms(700);


}
void pattern_blink_led6 (int* tab) {


pattern_blink_led3(light_partern3) ;
pattern_blink_led3(light_partern3) ;
_delay_ms(700);
```

```
}

void pattern_blink_led7 () {


pattern_blink_led4(light_partern4) ;
pattern_blink_led3(light_partern3) ;
_delay_ms(700);

}

void pattern_blink_led8 () {


pattern_blink_led2(light_partern1) ;
pattern_blink_led5(light_partern5) ;
pattern_blink_led1(light_partern1) ;
_delay_ms(700);

}

void pattern_blink_led9 () {


pattern_blink_led1(light_partern1) ;
pattern_blink_led8() ;
_delay_ms(700);

}

void pattern_blink_led10 () {


pattern_blink_led2(light_partern2) ;
pattern_blink_led3(light_partern3) ;
pattern_blink_led5(light_partern5) ;
_delay_ms(700);

}

Player Level_1 (int timer_select, int life_nbr, int score){

Player player_level1 ;

//int level1_life_level = life_nbr;

bool status = true ;
// int time_status = Timer_to_play(timer_select);
int counter = 0;
```

```
// blinking led template
My_turn();
pattern_blink_led1(light_partern1) ;
_delay_ms(1000) ;
Your_turn();
_delay_ms(500) ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 1");
Print_Life(life_nbr_level) ;



// your turn



while ( (counter <=0) and (life_nbr >  0) ) {

// check pattern
// To check a level, you have to increment a counter to 3, 5, 8, 10.. depending on
// light_parternX size

// we know the size of level 1 : 3
// user should press button S1 -> S2 -> S4




if (!(PIND & 1<<PIND3) and counter == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 1");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 1 - Success");
_delay_ms(1500) ;
score++ ;
}
```

```
else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (score == 1){

counter++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level1.vie = life_nbr ;
player_level1.score = score ;

return player_level1 ;


}


Player Level_2 (int timer_select, int life_nbr, int score){

Player player_level2 ;

//int level1_life_level = life_nbr;

bool status = true ;
// int time_status = Timer_to_play(timer_select);
int counter = 0;

// blinking led template
My_turn();
pattern_blink_led2(light_partern2) ;
_delay_ms(1000) ;
Your_turn();
```

```
_delay_ms(500) ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 2");
Print_Life(life_nbr_level) ;



// your turn



while ( (counter <=1) and (life_nbr >  0) ) {

// check pattern
// To check a level, you have to increment a counter to 3, 5, 8, 10.. depending on
// light_parternX size

// we know the size of level 1 : 3
// user should press button S1 -> S2 -> S4




if (!(PIND & 1<<PIND3) and counter == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 2");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (!(PIND & 1<<PIND2) and counter == 1){

PORTB ^= (1 << PORTB3);
```

```
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 2");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 2 - Success");
_delay_ms(1500) ;
score++ ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}



if (score == 2){

counter++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level2.vie = life_nbr ;
player_level2.score = score ;

return player_level2 ;
```

```
}


Player Level_3 (int timer_select, int life_nbr, int score){

Player player_level3 ;

//int level1_life_level = life_nbr;

bool status = true ;
// int time_status = Timer_to_play(timer_select);
int counter = 0;

// blinking led template
My_turn();
pattern_blink_led3(light_partern3) ;
_delay_ms(1000) ;
Your_turn();
_delay_ms(500) ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 3");
Print_Life(life_nbr_level) ;



// your turn



while ( (counter <=2) and (life_nbr >  0) ) {

// check pattern
// To check a level, you have to increment a counter to 3, 5, 8, 10.. depending on
// light_parternX size

// we know the size of level 1 : 3
// user should press button S1 -> S2 -> S4




if (!(PIND & 1<<PIND3) and counter == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
```

```
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 3");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (!(PIND & 1<<PIND2) and counter == 1){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 3");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}

if (!(PIND & 1<<PIND0) and counter == 2){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 3");
LCD_GoTo(9,1);
LCD_WriteText("*");
```

```
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 3 - Success");
_delay_ms(1500) ;
score++ ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}

if (score == 3){

counter++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level3.vie = life_nbr ;
player_level3.score = score ;

return player_level3 ;


}


Player Level_4 (int timer_select, int life_nbr, int score){


Player player_level4 ;


bool status = true ;
// int time_status = Timer_to_play(timer_select);
int counter2 = 0;
```

```
// blinking led template
My_turn();
pattern_blink_led4(light_partern4) ;
_delay_ms(1000) ;
Your_turn();
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 4");
Print_Life(life_nbr) ;



// your turn



while ( (counter2 <= 3) and (life_nbr >  0) ) {

// check pattern
// To check a level, you have to increment a counter to 3, 5, 8, 10.. depending on
// light_parternX size

// we know the size of level 1 : 3
// user should press button S1 -> S2 -> S4

if (!(PIND & 1<<PIND3) and counter2 == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter2 ++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 4");
LCD_GoTo(6,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}



if (!(PIND & 1<<PIND2) and counter2 == 1){
```

```
PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter2 ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 4");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter2 == 2){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter2 ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 4");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter2 == 3){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
```

```
counter2 ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 4");
LCD_GoTo(9,1);
LCD_WriteText("*");
_delay_ms(100) ;
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 4 - Success");
_delay_ms(1500) ;
score++ ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}



if (score == 4){

counter2++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level4.score = score ;
player_level4.vie = life_nbr ;
return player_level4 ;


}
```

```
Player Level_5 (int timer_select, int life_nbr, int score){


Player player_level5 ;


bool status = true ;
// int time_status = Timer_to_play(timer_select);
int counter2 = 0;

// blinking led template
My_turn();
pattern_blink_led5(light_partern5) ;
_delay_ms(1000) ;
Your_turn();
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 5");
Print_Life(life_nbr) ;



// your turn



while ( (counter2 <= 4) and (life_nbr >  0) ) {

// check pattern
// To check a level, you have to increment a counter to 3, 5, 8, 10.. depending on
// light_parternX size

// we know the size of level 1 : 3
// user should press button S1 -> S2 -> S4

if (!(PIND & 1<<PIND3) and counter2 == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter2 ++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 5");
LCD_GoTo(5,1);
LCD_WriteText("*");
```

```
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}



if (!(PIND & 1<<PIND2) and counter2 == 1){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter2 ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 5");
LCD_GoTo(6,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter2 == 2){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter2 ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 5");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter
```

```
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter2 == 3){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter2 ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 5");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}


if (!(PIND & 1<<PIND1) and counter2 == 4){

PORTB ^= (1 << PORTB4);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter2 ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 5");
LCD_GoTo(9,1);
LCD_WriteText("*");
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 5 - Success");
_delay_ms(1500) ;
score++ ;
```

```
}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}

if (score == 5){

counter2++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level5.score = score ;
player_level5.vie = life_nbr ;
return player_level5 ;


}


Player Level_6 (int timer_select, int life_nbr, int score){


Player player_level6 ;


bool status = true ;
// int time_status = Timer_to_play(timer_select);
int counter = 0;

// blinking led template
My_turn();
pattern_blink_led6(light_partern3) ;
_delay_ms(1000) ;
Your_turn();
LCD_Clear();
LCD_GoTo(5,0);
```

```
LCD_WriteText("LEVEL 6");
Print_Life(life_nbr) ;



// your turn



while ( (counter <= 5) and (life_nbr >  0) ) {

// check pattern
// To check a level, you have to increment a counter to 3, 5, 8, 10.. depending on
// light_parternX size

// we know the size of level 1 : 3
// user should press button S1 -> S2 -> S4

if (!(PIND & 1<<PIND3) and counter == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 6");
LCD_GoTo(5,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (!(PIND & 1<<PIND2) and counter == 1){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
```

```
LCD_WriteText("LEVEL 6");
LCD_GoTo(6,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}

if (!(PIND & 1<<PIND0) and counter == 2){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 6");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}

if (!(PIND & 1<<PIND3) and counter == 3) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 6");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
```

```
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (!(PIND & 1<<PIND2) and counter == 4){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 6");
LCD_GoTo(9,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}

if (!(PIND & 1<<PIND0) and counter == 5){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 6");
LCD_GoTo(10,1);
LCD_WriteText("*");
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 6 - Success");
_delay_ms(1500) ;
score++ ;

}
```

```
else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}


if (score == 6){

counter++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level6.score = score ;
player_level6.vie = life_nbr ;
return player_level6 ;


}


Player Level_7 (int timer_select, int life_nbr, int score){


Player player_level7 ;


bool status = true ;
int counter = 0;

// blinking led template
My_turn();
pattern_blink_led7() ;
_delay_ms(1000) ;
Your_turn();
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
Print_Life(life_nbr) ;
```

```
// your turn


while ( (counter <= 6) and (life_nbr >  0) ) {

if (!(PIND & 1<<PIND3) and counter == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
LCD_GoTo(4,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}



if (!(PIND & 1<<PIND2) and counter == 1){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
LCD_GoTo(5,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
```

```
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter == 2){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
LCD_GoTo(6,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter == 3){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}
```

```
if (!(PIND & 1<<PIND3) and counter == 4) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (!(PIND & 1<<PIND2) and counter == 5){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
LCD_GoTo(9,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}

if (!(PIND & 1<<PIND0) and counter == 6){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
```

```
counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
LCD_GoTo(10,1);
LCD_WriteText("*");
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 7 - Success");
_delay_ms(1500) ;
score++ ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (score == 7){

counter++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level7.score = score ;
player_level7.vie = life_nbr ;
return player_level7 ;


}


Player Level_8 (int timer_select, int life_nbr, int score){
```

```
Player player_level8 ;


bool status = true ;
int counter = 0;

// blinking led template
My_turn();
pattern_blink_led8() ;
_delay_ms(1000) ;
Your_turn();
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
Print_Life(life_nbr) ;



// your turn



while ( (counter <= 7) and (life_nbr >  0) ) {

if (!(PIND & 1<<PIND3) and counter == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
LCD_GoTo(4,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (!(PIND & 1<<PIND2) and counter == 1){
```

```
PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
LCD_GoTo(5,1);
LCD_WriteText("*");
_delay_ms(100) ;


}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}

if (!(PIND & 1<<PIND3) and counter == 2) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
LCD_GoTo(6,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}



if (!(PIND & 1<<PIND2) and counter == 3){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;
```

```
counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter == 4){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter == 5){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
```

```
LCD_GoTo(9,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}


if (!(PIND & 1<<PIND1) and counter == 6){

PORTB ^= (1 << PORTB4);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
LCD_GoTo(10,1);
LCD_WriteText("*");
_delay_ms(100) ;


}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}


if (!(PIND & 1<<PIND3) and counter == 7) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
LCD_GoTo(11,1);
LCD_WriteText("*");
```

```
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 8 - Success");
_delay_ms(1500) ;
score++ ;

}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}




if (score == 8){

counter++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level8.score = score ;
player_level8.vie = life_nbr ;
return player_level8 ;


}

Player Level_9 (int timer_select, int life_nbr, int score){


Player player_level9 ;


bool status = true ;
```

```
// int time_status = Timer_to_play(timer_select);
int counter = 0;

// blinking led template
My_turn();
pattern_blink_led9() ;
_delay_ms(1000) ;
Your_turn();
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
Print_Life(life_nbr) ;



// your turn



while ( (counter <= 8) and (life_nbr >  0) ) {


if (!(PIND & 1<<PIND3) and counter == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(3,1);
LCD_WriteText("*");
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}

if (!(PIND & 1<<PIND3) and counter == 1) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
```

```
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(4,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (!(PIND & 1<<PIND2) and counter == 2){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(5,1);
LCD_WriteText("*");
_delay_ms(100) ;


}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}

if (!(PIND & 1<<PIND3) and counter == 3) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(6,1);
LCD_WriteText("*");
```

```
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}



if (!(PIND & 1<<PIND2) and counter == 4){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter == 5){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter
```

```
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter == 6){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(9,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}


if (!(PIND & 1<<PIND1) and counter == 7){

PORTB ^= (1 << PORTB4);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(10,1);
LCD_WriteText("*");
_delay_ms(100) ;


}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
```

```
_delay_ms(100) ;
}


if (!(PIND & 1<<PIND3) and counter == 8) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
LCD_GoTo(11,1);
LCD_WriteText("*");
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 9 - Success");
_delay_ms(1500) ;
score++ ;

}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}


if (score == 9){

counter++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level9.score = score ;
```

```
player_level9.vie = life_nbr ;
return player_level9 ;


}


Player Level_10 (int timer_select, int life_nbr, int score){


Player player_level10 ;


bool status = true ;
// int time_status = Timer_to_play(timer_select);
int counter = 0;

// blinking led template
My_turn();
pattern_blink_led10() ;
_delay_ms(1000) ;
Your_turn();
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
Print_Life(life_nbr) ;



// your turn



while ( (counter <= 9) and (life_nbr >  0) ) {

if (!(PIND & 1<<PIND3) and counter == 0) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
LCD_GoTo(3,1);
LCD_WriteText("*");
_delay_ms(100) ;
}
```

```
else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}



if (!(PIND & 1<<PIND2) and counter == 1){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
LCD_GoTo(4,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}
if (!(PIND & 1<<PIND3) and counter == 2) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
LCD_GoTo(5,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}
```

```
if (!(PIND & 1<<PIND2) and counter == 3){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
LCD_GoTo(6,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;

}

if (!(PIND & 1<<PIND0) and counter == 4){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
LCD_GoTo(7,1);
LCD_WriteText("*");
_delay_ms(100) ;


}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(500) ;
}
if (!(PIND & 1<<PIND3) and counter == 5) {

PORTB ^= (1 << PORTB2);
_delay_ms(1000);
```

```
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
LCD_GoTo(8,1);
LCD_WriteText("*");
_delay_ms(100) ;
}

else if ((!(PIND & 1<<PIND2) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}



if (!(PIND & 1<<PIND2) and counter == 6){

PORTB ^= (1 << PORTB3);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
LCD_GoTo(9,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND0)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter == 7){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
```

```
LCD_WriteText("LEVEL 10");
LCD_GoTo(10,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}

if (!(PIND & 1<<PIND0) and counter == 8){

PORTB ^= (1 << PORTB5);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 5");
LCD_GoTo(11,1);
LCD_WriteText("*");
_delay_ms(100) ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND1) or !(PIND & 1<<PIND2)) and counter

life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;

}


if (!(PIND & 1<<PIND1) and counter == 9){

PORTB ^= (1 << PORTB4);
_delay_ms(1000);
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5) ;

counter ++ ;
//LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
LCD_GoTo(12,1);
```

```
LCD_WriteText("*");
_delay_ms(100) ;
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("LEVEL 10 - Success");
_delay_ms(1500) ;
score++ ;

}

else if ((!(PIND & 1<<PIND3) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND1)) and counter
life_nbr = Minus_life(life_nbr) ;
_delay_ms(100) ;
}



if (score == 10){

counter++ ;
}




}



LCD_Clear();
Print_Score(score);
Print_Life(life_nbr) ;
_delay_ms(1000);

player_level10.score = score ;
player_level10.vie = life_nbr ;
return player_level10 ;


}


int main(void)
{



// IO configuration
DDRB  |= (1<<LED1) | (1<<LED2) | (1<<LED3) | (1<<LED4);          // configuration of L
```

```
DDRC &=~ (1<<ADCIN);                    // ADC pin as input

// leds controller
DDRB |= ((1 << DDB2) | (1 << DDB3) | (1 << DDB4) | (1 << DDB5));  // Direction of por
PORTB |= (1<<PORTB2) | (1<<PORTB3) | (1<<PORTB4) | (1<<PORTB5);    // Output register

// buttons controller
DDRD &= ~(1<<DDD0) | ~(1<<DDD1) | ~(1<<DDD2) | ~(1<<DDD3);         // Direction of por


// screen initialization
LCD_Initalize();


bool status_game = true  ;


EEPROM.get(0,best_score_ever) ;
Player play1 ;


while (1) {

life_nbr_level = 5 ;
score_level = 0 ;

while ((PIND & 1<<PIND0)){


LCD_Clear();
LCD_GoTo(3,0);
LCD_WriteText("MEMORY GAME");
LCD_GoTo(3,1);
LCD_WriteText("Play -> S4");
_delay_ms(500);

if (!(PIND & 1<<PIND1) or !(PIND & 1<<PIND2) or !(PIND & 1<<PIND3)){
LCD_Clear();
LCD_GoTo(0,0);
LCD_WriteText("Waiting for user");
start_in(5) ;

}

}

if (!(PIND & 1<<PIND0)){

LCD_Clear();
```

```
LCD_GoTo(2,0);
LCD_WriteText("Best Score");
LCD_GoTo(2,1);
sprintf(str, "%d", best_score_ever);
LCD_WriteText(str);
_delay_ms(1500) ;



while (score_level >= 0){

if (score_level == 0) {


LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 1");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_1(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
score_level = play1.score ;
EEPROM_best_score(score_level) ;


if (life_nbr_level <= 0){
score_level = -1 ;
}

}




if (score_level == 1) {

// enter level 2

LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 2");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_2(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
```

```
score_level = play1.score ;
EEPROM_best_score(score_level) ;

if (life_nbr_level <= 0){
score_level = -1 ;
}
}

if (score_level == 2) {

LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 3");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_3(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
score_level = play1.score ;
EEPROM_best_score(score_level) ;


if (life_nbr_level <= 0){
score_level = -1 ;

}

}


if (score_level == 3) {



LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 4");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_4(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
score_level = play1.score ;
EEPROM_best_score(score_level) ;


if (life_nbr_level <= 0){
score_level = -1 ;
}
```

```
}

if (score_level == 4) {

LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 5");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_5(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
score_level = play1.score ;
EEPROM_best_score(score_level) ;


if (life_nbr_level <= 0){
score_level = -1 ;
}


}




if (score_level == 5) {

LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 6");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_6(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
score_level = play1.score ;
EEPROM_best_score(score_level) ;

if (life_nbr_level <= 0){
score_level = -1 ;
}


}

if (score_level == 6) {
```

```
LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 7");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_7(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
score_level = play1.score ;
EEPROM_best_score(score_level) ;

if (life_nbr_level <= 0){
score_level = -1 ;
}


}

if (score_level == 7) {

LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 8");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_8(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
score_level = play1.score ;
EEPROM_best_score(score_level) ;

if (life_nbr_level <= 0){
score_level = -1 ;
}


}

if (score_level == 8) {

LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 9");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_9(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
```

```
score_level = play1.score ;
EEPROM_best_score(score_level) ;

if (life_nbr_level <= 0){
score_level = -1 ;
}


}

if (score_level == 9) {

LCD_Clear();
LCD_GoTo(5,0);
LCD_WriteText("LEVEL 10");
_delay_ms(2000) ;
//start_in(3) ;
play1 = Level_10(4000, life_nbr_level, score_level) ;

life_nbr_level = play1.vie ;
score_level = play1.score ;
EEPROM_best_score(score_level) ;

if (life_nbr_level <= 0){
score_level = -1 ;
}


}

if (score_level == 10) {

LCD_Clear();
LCD_GoTo(4,0);
LCD_WriteText("SUCCESS !");
LCD_GoTo(0,1);
LCD_WriteText("CONGRATULATION");
_delay_ms(2000) ;
score_level = -1 ;


}


}

LCD_Clear();
Print_Game_Over_Life() ;
_delay_ms(2000) ;
```

```
score_level = 0 ;


}


}
}
```