OPERATION

The Goofy Game for Goofy Doctors

Jacob Ander, Carl Johansson & Lovisa Dimberg

2019-05-16
Projektarbete i kursen EITF12 - Digitala Projekt

ABSTRACT

The purpose of this project was to develop a functioning prototype by constructing hardware and developing the necessary software. The end product was to be a simple construction with optional features. The group decided to develop a prototype mimicking Operation, a board game testing the players eye-hand coordination and fine motor skills.

The following report covers the development process, requirement specification and description of the final prototype developed.

INNEHÅLLSFÖRTECKNING

INLEDNING

I kursen Digitala Projekt ska studenter i grupper om två till tre utveckla en enkel prototyp för vidareutveckling. Syftet med kursen är att ge en förståelse för industriellt utvecklingsarbete genom att studenterna får konstruera, bygga och testa en konstruktion. Konstruktionen ska bestå av både hård- och mjukvara, men den exakta utformningen är upp till varje grupp att själva välja.

I samråd med handledare valde denna grupp att utveckla en simpel version av Operation, ett brädspel som testar spelarens koordination och finmotorik. Följande rapport beskriver utvecklingsarbetet från idé till färdig prototyp. Det finns även en användarmanual och kravspecifikationen som beskriver vilka krav som ställs på funktionerna hos den färdiga prototypen. Under rubrikerna Resultat och Diskussion utvärderas arbetet och den färdiga prototypen, och allra sist i appendixet återfinns det kopplingsschema och den källkod som använts.

BESKRIVNING AV PROTOTYP

Spelet går ut på att med hjälp av en pincett, på så kort tid så möjligt, plocka ur små föremål ur håligheter i spelplanen utan att nudda hålets kanter. Liksom i originalspelet Operation, som ses i bild 1, består spelet av en spelplan med tillhörande pincett, (med skillnaden att i vårt spel är spelplanen en stilren metallplatta istället för en obehaglig teckning). I tillägg finns även en skärm som visar tidtagningen för den aktuella spelrundan, och efter rundans slut en scoreboard där de tre hittills bästa tiderna med tillhörande spelarnamn är listade.

Spelet har två knappar, en spelknapp och en resetknapp. När spelaren trycker på spelknappen startas tidtagningen för nuvarande spelrunda, och spelaren får börja försöka plocka ur föremålen ur roboten. Råkar pincetten eller föremålet (om föremålet är strömförande) vidröra hålets kant adderas fem sekunder till spelarens tid. Samtidigt lyser respektive låter en lysdiod och en buzzer för att uppmärksamma spelaren på misstaget. När alla föremål är utopererade trycker spelaren på spelknappen igen, och tiden stoppas. Är tiden kortare än någon av tiderna på scoreboarden får spelaren använda spelknappen till att knappa in ett spelarnamn på tre bokstäver, och scoreboarden uppdateras.

Resetknappen kan användas till att avsluta spelrundan och till att nollställa scoreboarden. Skulle spelaren av någon anledning vilja avbryta eller börja om spelrundan i förtid, trycker denna på resetknappen. Då återgår displayen till utgångsläget och tiden jämförs inte med scoreboarden. Vill spelaren rensa scoreboarden trycker denne ned resetknappen när scoreboarden visas.

Bild 1. Originalspelet Operation

KRAVSPECIFIKATION

Den färdiga prototypen ska uppfylla följande krav:

1. När strömmen är påkopplad ska displayen visa texten “High Scores!” och scoreboarden visas. Är scoreboarden tom visas istället för scoreboarden texten “No games played yet!”.
2. När spelknappen trycks ned ska texten “Your time is:” och en tidräknaren som börjar räkna upp från noll visas.
3. Om pincetten eller ett strömförande föremål vidrör en hålkant ska:
    a. Tidräknaren hoppa 5 sekunder uppåt.
    b. Lysdioden lysa så länge som kontakten varar.

    c. Buzzern låta så länge som kontakten varar.
4. När spelknappen trycks ned en andra gång ska tidräknaren stanna, och om siffran understiger siffrorna på scoreboarden, eller om scoreboarden har tomma platser, ska texten “Write your name” och bokstaven A visas. Om resetknappen då trycks ned visas istället B, trycks den ned igen istället C och så vidare. När spelaren kommit fram till önskad bokstav trycks spelknappen ned för att låsa bokstaven, och proceduren upprepas i två bokstäver till. När ett spelarID på tre bokstäver skrivits in visas den uppdaterade scoreboarden.
5. När resetknappen trycks ned under en pågående runda ska tidräknaren nollställas, utan att tiden, även om den understiger de på scoreboarden, sätts in på denna. Skärmen ska återgå till utgångsläget där “High Score! och aktuell scoreboard visas.
6. Hålls resetknappen ned när scoreboarden visas ska tiderna raderas, och skärmen återgå till utgångsläget där “High Score! No games played yet” visas.

HÅRDVARA

Processor	ATmega16, en 8-bitars microcontroller med 40 pinnar. Hur komponenterna är kopplade till processorn kan ses i kopplingsschemat i bilaga 1.
Display	LCD display, 4 raders alfanumerisk display av typen Sharp Dot Matrix, GDM1602K-serien.
Lysdiod	1 st, lyser när spelplanen och pincetten bildar en sluten krets vid kontakt.
Buzzer	1 st, låter när spelplanen och pincetten bildar en sluten krets vid kontakt.
Resistorer	4 st
Knappar	2 stycken, en knapptryckning registreras genom att spänningen är 0V när den är nedtryckt och 5V annars.
JTAG	Används för att föra över mjukvaran till processorn och för att felsöka koden.
Kondensatorer	2 st
Potentiometer	1 st, anpassar kontrasten på displayen

Spelplan	Bestående av en metallplatta med träbotten samt en pincett kopplad till metallplattan.

MJUKVARA

Källkoden är utvecklad i Atmel Studio 7 i programmeringsspråket C. Den är bifogad i bilaga 2. Kopplingsschemat är gjort i Eagle och är bifogat i bilaga 1.

GENOMFÖRANDE

KONSTRUKTION AV HÅRDVARA

Efter att ha bestämt hur spelet ska vara utformat så gjordes ett kopplingsschema utifrån de komponenter vi behövde för att spelet skulle fungera som vi ville. Komponenterna monterades en efter en på konstruktionen och testades efter hand genom en strömmätare och enklare kod med hjälp av JTAG. Vid felkoppling eller komponenter som inte fungerade som vi ville så justerade vi detta tills konstruktionen fungerade tillfredsställande.

PROGRAMMERING AV MJUKVARA

Då ingen av oss i projektet hade erfarenhet av C så gick programmeringsmetoden ut på att googla, fråga handledare och att kolla hur tidigare projekt har löst liknande problem som vi stod inför.

Vårt största problem var att göra en scoreboard. Denna utmaning löstes genom att lagra spelarnamn och speltid i en struct som vi sorterade med qsort för att scoreboarden ständigt ska vara uppdaterad och visa den bästa (lägsta) tiden först.

Mindre problem var att vi kollade i fel datablad och vi hade snurrat sladdarna fel eller på ett opraktiskt sätt.

RESULTAT

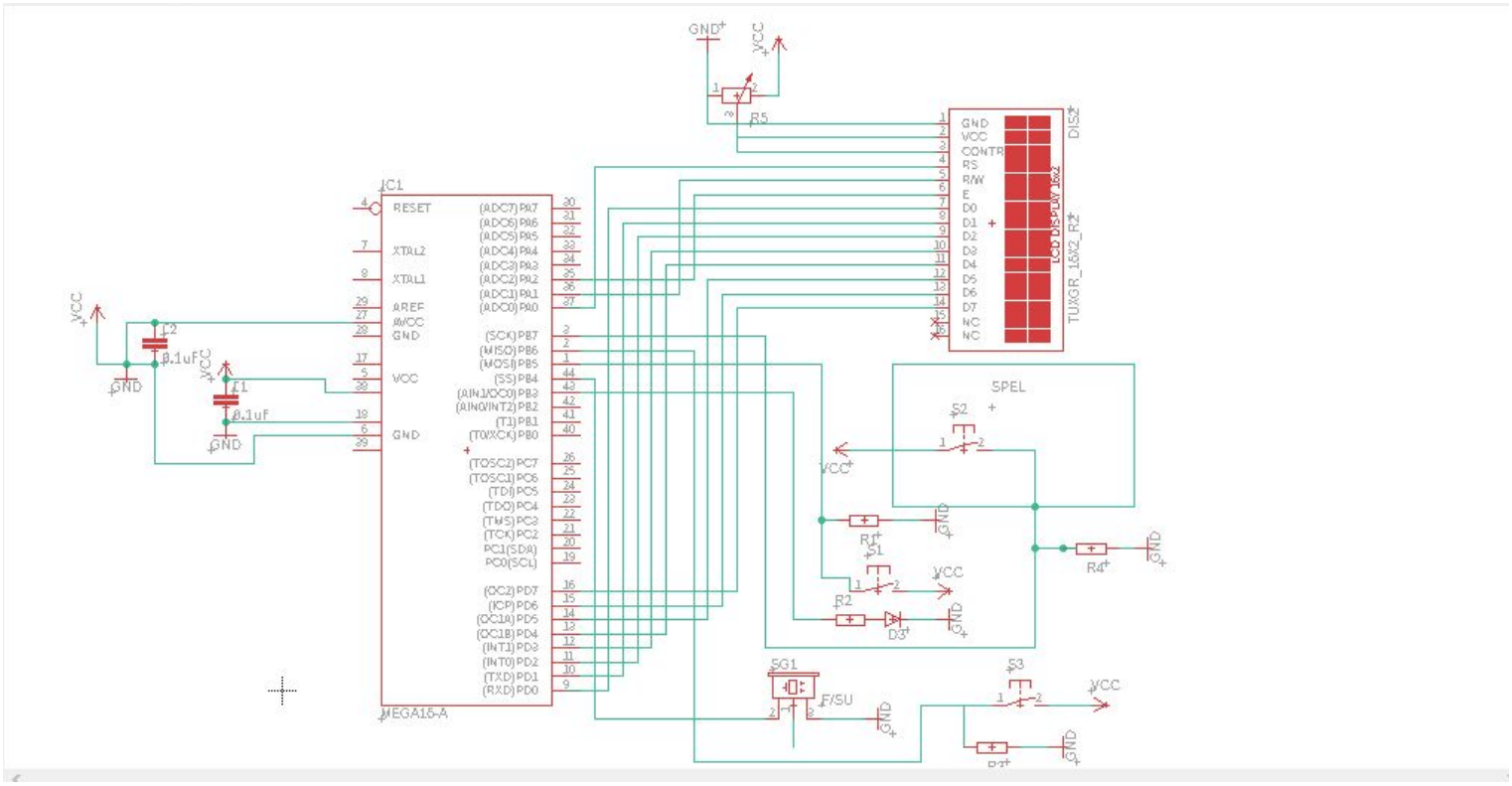
Projektet blev lyckat och resulterade i ett spel som fungerar enligt kravspecifikationen ovan.

DISKUSSION

Trots för kursen vanlig inledande svårigheter med att komma igång med programmering i C har projektet fortlöpt relativt smidigt. Det praktiska arbetet med att konstruera hårdvaran var ett ovant och välkommet inslag i studierna för oss tre I-studenter.

Vi bedömer att vår initiala målsättningen med prototypen låg på en lagom nivå, så att projektet blev utmanande utan att kännas övermäktigt. Vi är överlag nöjda med vårt projekt, och finner själva spelet mycket underhållande!

BILAGA 1: KOPPLINGSSCHEMA



BILAGA 2: KÄLLKOD

```
/*
 * OperationProjekt.c
 */*
 * OperationProjekt.c
 *
 * Created: 2019-04-15 15:38:45
 * Author : ja1503an-s
 */
#define F_CPU 8000000UL
#include <util/delay.h>
#include <stdio.h>
#include <avr/io.h>
#include <time.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#include <string.h>

#define BUTTON1_PRESSED PINB & (1<<PB5) //Button is pressed
#define BUTTON2_PRESSED PINB & (1<<PB6) //Button is pressed
#define TWEEZERS PINB & (1<<PB7)

#define B_H PORTA |=(1<<PA3);
#define B_L PORTA &= ~(1<<PA3);

#define C_H PORTA |=(1<<PA4);
#define C_L PORTA &= ~(1<<PA4);

#define D_H PORTA |=(1<<PA5);
#define D_L PORTA &= ~(1<<PA5);

#define E_H PORTA |=(1<<PA2);
#define E_L PORTA &= ~(1<<PA2);

#define RS_H  PORTA |=(1<<PA0);
#define  RS_L PORTA &= ~(1<<PA0);

#define RW_H PORTA |=(1<<PA1);
#define RW_L PORTA &= ~(1<<PA1);

void clear_display(void);
```

```
void write_string(char string[]);
void init(void);
void buzzer(void);
void redDiod(void);
void timer(void);
void timeStart(void);
void timeEnd(void);
void penalty(void);
void displayOn(void);
void functionSet(void);
void displayOff(void);
void write_cmd(char c);
void write_char(char c);
void read_cmd(char c);
void write_string(char string[]);
void time_keep(void);
void timer0_init(void);
void button1(void);
void setDDRAM(char c);
void initVals(void);
void penalty(void);
void resetVals(void);
void highScore(void);
void button2(void);
void resetScoreboard(void);
void button1ForLetters(void);
void chooseLetter(void);
void button2ForLetters(void);
void readEeprom(void);
void writeToEeprom(void);

ISR(TIMER0_OVF_vect);
uint8_t  EEMEM EEPROMString[3];
int  EEMEM EEPROMINT;
char iForLetter;
uint8_t timerOverflowCount=0,sec;
uint8_t start,end,totalTime,penaltyTime;
int timerstart =1;
char result[5];
char scoreVal[5];
char highscoreToString[5];
int b1 = 0;
int n = 0;
```

```
int tioT;
int currentTime;
int penaltyDelay;
int highScore1;
int iForChoosing;

char newName[4];
typedef int bool;
#define true 1
#define false 0

bool pressedB1,notPressedB1,pressedB2,
scoreWritten,gamePlayed,gameReset,choosingLetters,pressedB2L,pressedB1L;


typedef struct{
	int playerScore;
	char player[3];
}Highscores;

Highscores arr[4];
int int_cmp (const void * a, const void * b)
{
	Highscores *scoreA = (Highscores *)a;
	Highscores *scoreB = (Highscores *)b;
	return ( scoreA->playerScore - scoreB->playerScore );
}
Highscores EEMEM EEStruct;
int main(void)
{

	TCNT0=0x00;
	TCCR0 = (1<<CS02);
	init();
	displayOn();
	functionSet();
	clear_display();
	timer0_init();
	sei();
	initVals();
	readEeprom();
	while(1)
	{
```

```
while(BUTTON1_PRESSED){
        button1();
        timeStart();
        //buzzer();
        redDiod();
        clear_display();
        resetVals();
        setDDRAM(0b10000000);
        choosingLetters = true;
        write_string("Your time is: ");
}
        pressedB1 = false;

        /*************************************************************************/
        /* Spelet körs                                                           */
        /*************************************************************************/
        while (n == 1)
        {
                while (BUTTON1_PRESSED)
                {
                        button1();

                }
                pressedB1 = false;



                if (TWEEZERS)
                {
                        redDiod();
                        buzzer();
                        penalty();
                }

                currentTime = sec - start + penaltyTime;
                itoa(currentTime,result,10);
                setDDRAM(0xc0);
                write_char(result[0]);

                if (result[1]!=0)
                {
                        setDDRAM(0xc1);
```

```
                write_char(result[1]);
            }
            if (result[2]!=0)
            {
                setDDRAM(0xc2);
                write_char(result[2]);
            }
            scoreWritten = false;
            gamePlayed = true;
                while (BUTTON2_PRESSED)
                {
                    button2();
                }
                pressedB2 = false;
        }

        if(gamePlayed == true && gameReset == false){
            timeEnd();
            gamePlayed = false;
        }

        if(scoreWritten == false){
            clear_display();

            highScore();
            scoreWritten = true;
        }

        while (BUTTON2_PRESSED)
                {
                    resetScoreboard();
                }
                pressedB2 = false;
    }
}

void init(){
    DDRA = 0b11111111; //Port A output
    _delay_ms(2);
    DDRB = 0b11101111; //Port B biderictional for 0-2, output for 3-4 and input for 5
        _delay_ms(2);
    DDRB = 0xFF; //Port B output
        _delay_ms(2);
```

```
	DDRD = 0xFF; //Port D output
		_delay_ms(2);
	 DDRB |= 1<<PB3;
		_delay_ms(2);
	 DDRB |= 1<<PB4;
		_delay_ms(2);
	 DDRB &= ~(1<<PB5); //ändrade från DDRD till DDRB
		_delay_ms(2);
}

void initVals(){
	pressedB1 = true;
	notPressedB1 = true;
	scoreWritten = false;
	penaltyDelay = -6;
	highScore1 = 10000;
	gamePlayed = false;
	pressedB2 = true;
	gameReset = false;
	choosingLetters = true;
	pressedB2L =true;
	pressedB1L = true;

	strcpy(arr[0].player,"NUL");
	arr[0].playerScore = 255;

	strcpy(arr[1].player,"NUL");
	arr[1].playerScore = 255;

	strcpy(arr[2].player,"NUL");
	arr[2].playerScore = 255;

	strcpy(arr[3].player,"NUL");
	arr[3].playerScore = 255;
	iForLetter = 0x65;
}
void buzzer(){
	PORTB ^= 1<<PB4;

}

void redDiod(){
	PORTB ^= 1<<PB3;
```

```
}

void highScore(){
        setDDRAM(0x80);
        write_string("Highscores!");
        qsort(arr, 4, sizeof(Highscores),int_cmp);
        if (arr[0].playerScore == 255)
        {
        setDDRAM(0xc0);
         write_string("No games played yet!");
        }else{


        setDDRAM(0xC0);
        itoa(arr[0].playerScore,scoreVal,10);
        write_char(arr[0].player[0]);
        write_char(arr[0].player[1]);
        write_char(arr[0].player[2]);
        write_char(0x20);
        write_string(scoreVal);
        if (arr[1].playerScore!= 255)
        {
        setDDRAM(0x94);
        itoa(arr[1].playerScore,scoreVal,10);
        write_char(arr[1].player[0]);
        write_char(arr[1].player[1]);
        write_char(arr[1].player[2]);
        write_char(0x20);
        write_string(scoreVal);
        }
        if (arr[2].playerScore!=255)
        {
        setDDRAM(0xD4);
        itoa(arr[2].playerScore,scoreVal,10);
        write_char(arr[2].player[0]);
        write_char(arr[2].player[1]);
        write_char(arr[2].player[2]);
        write_char(0x20);
        write_string(scoreVal);
        }

        }
        writeToEeprom();
```

```
}
void readEeprom(){
	for (int j=0;j<3;j++)
	{
	for (int i= 0; i<3;i++)
	{
		arr[j].player[i] = eeprom_read_byte((uint8_t*) i+(j*3));
	}
	}

	for (int i = 0;i<3;i++)
	{
		arr[i].playerScore = eeprom_read_byte((uint8_t*) i+9);
	}


}
void writeToEeprom(){
	int j = 0;
	for(int i= 0; i<3;i++)
	{
		eeprom_write_byte ((uint8_t*) 0+j*3, arr[i].player[0]);
		eeprom_write_byte ((uint8_t*) 1+j*3, arr[i].player[1]);
		eeprom_write_byte ((uint8_t*) 2+j*3 , arr[i].player[2]);
		j++;
	}

	for (int i=0; i<3;i++)
	{
		eeprom_write_byte((uint8_t*) 9+i, arr[i].playerScore);
	}

}
void button1(){
	if (pressedB1 == false)
	{

	if (b1==1)
	{
		b1=0;
	}else{
		b1 = 1;
	}
```

```
                switch(b1){
                        case 0: n = 0 ;
                        break;
                        case 1: n = 1 ;
                        break;
                }
        }
        pressedB1 = true;
}

void button2(){
        if(pressedB2 == false){
        if(n == 0){
                highScore1 = 10000;
        }else{
                n = 0;
        }
        }
        b1 =0;
        pressedB2 = true;
        gameReset = true;
}

// initialize timer, interrupt and variable
void timer0_init()
{
        // set up timer with prescaler = 256
        TCCR0 |= (1 << CS02);

        // initialize counter
        TCNT0 = 0;

        // enable overflow interrupt
        TIMSK |= (1 << TOIE0);

        // enable global interrupts


        // initialize overflow counter variable
        timerOverflowCount = 0;
}
void resetVals(){
        penaltyDelay = -6;
```

```
	penaltyTime = 0;
	gameReset = false;

}
ISR(TIMER0_OVF_vect)
{
	// keep a track of number of overflows
	timerOverflowCount++;
	if (timerOverflowCount >= 122)
	{
		timerOverflowCount = 0;
		sec++;
	}
}


void timeStart(){
	start = sec;
}
void timeEnd(){

		highScore1=currentTime;
		arr[3].playerScore = highScore1;
		clear_display();
		setDDRAM(0x80);
		write_string("Write your name");
		if(arr[2].playerScore>arr[3].playerScore){
			chooseLetter();
		}
		strcpy(arr[3].player,newName);
}

void chooseLetter(){

	iForChoosing = 0;
	iForLetter = 0x41;
	while (choosingLetters == true)
	{
		if(BUTTON2_PRESSED){
		while(BUTTON2_PRESSED){
		button2ForLetters();
	}
		}
```

```
                if(BUTTON1_PRESSED){
                while(BUTTON1_PRESSED){
                button1ForLetters();
        }
                }
        newName[iForChoosing]=iForLetter;
        if (iForChoosing == 0)
        {
                setDDRAM(0xc0);
                write_char(newName[iForChoosing]);
        }else if (iForChoosing == 1)
        {
                setDDRAM(0xc1);
                write_char(newName[iForChoosing]);
        }else if (iForChoosing == 2)
        {
                setDDRAM(0xc2);
                write_char(newName[iForChoosing]);
        }else{
                choosingLetters = false;
        }
        pressedB1L = false;
        pressedB2L = false;
        }


}

void resetScoreboard(){
        strcpy(arr[0].player,"NUL");
        arr[0].playerScore = 255;

        strcpy(arr[1].player,"NUL");
        arr[1].playerScore = 255;

        strcpy(arr[2].player,"NUL");
        arr[2].playerScore = 255;

        strcpy(arr[3].player,"NUL");
        arr[3].playerScore = 255;
        scoreWritten = false;
}
```

```
void button1ForLetters(){
	if(pressedB1L == false){
	iForChoosing++;
	}
	pressedB1L = true;
	iForLetter = 0x41;
}

void button2ForLetters(){
		if(pressedB2L == false){
		if (iForLetter<0x5A)
		{
			iForLetter++;
		}else{
		iForLetter = 0x41;
	   }


	}
	pressedB2L = true;
}
void penalty(){
	if (penaltyDelay+6 < currentTime)
	{
		penaltyTime = penaltyTime +5;
		penaltyDelay = currentTime;
	}
}

void displayOn(){
	write_cmd(0b00001100);
	_delay_ms(10);
}

void functionSet(){
	write_cmd(0b00111100);
	_delay_us(60);
}



void displayOff(){
	write_cmd(0b00001000);
```

```
}

void setDDRAM(char c){
	write_cmd(c);
	_delay_us(60);
}

void write_cmd(char c){
	RS_L;
	RW_L;
	E_H;
	PORTD = c;
	E_L;
	E_H;
}

void write_char(char c){
	RS_H;
	RW_L;
	E_H;
	PORTD =c;
	E_L;
	E_H;
	_delay_us(60);
}

void read_cmd(char c){
	RS_H;
	RW_H;
	E_H;
	PORTD = c;
	E_L;
	E_H;
}

void write_string(char string[]){
	int i = 0;
	while(string[i] !='\0'){
		write_char(string[i]);
		i++;
	}
}
```

```
void time_keep(){

}

void clear_display(){
        write_cmd(0b00000001);
        _delay_ms(4);
}



 * Created: 2019-04-15 15:38:45
 * Author : ja1503an-s
 */
#define F_CPU 8000000UL
#include <util/delay.h>
#include <stdio.h>
#include <avr/io.h>
#include <time.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#include <string.h>

#define BUTTON1_PRESSED PINB & (1<<PB5) //Button is pressed
#define BUTTON2_PRESSED PINB & (1<<PB6) //Button is pressed
#define TWEEZERS PINB & (1<<PB7)

#define B_H PORTA |=(1<<PA3);
#define B_L PORTA &= ~(1<<PA3);

#define C_H PORTA |=(1<<PA4);
#define C_L PORTA &= ~(1<<PA4);

#define D_H PORTA |=(1<<PA5);
#define D_L PORTA &= ~(1<<PA5);

#define E_H PORTA |=(1<<PA2);
#define E_L PORTA &= ~(1<<PA2);

#define RS_H  PORTA |=(1<<PA0);
#define  RS_L PORTA &= ~(1<<PA0);
```

```
#define RW_H PORTA |=(1<<PA1);
#define RW_L PORTA &= ~(1<<PA1);

void clear_display(void);
void write_string(char string[]);
void init(void);
void buzzer(void);
void redDiod(void);
void timer(void);
void timeStart(void);
void timeEnd(void);
void penalty(void);
void displayOn(void);
void functionSet(void);
void displayOff(void);
void write_cmd(char c);
void write_char(char c);
void read_cmd(char c);
void write_string(char string[]);
void time_keep(void);
void timer0_init(void);
void button1(void);
void setDDRAM(char c);
void initVals(void);
void penalty(void);
void resetVals(void);
void highScore(void);
void button2(void);
void resetScoreboard(void);
void button1ForLetters(void);
void chooseLetter(void);
void button2ForLetters(void);
void readEeprom(void);
void writeToEeprom(void);

ISR(TIMER0_OVF_vect);
uint8_t  EEMEM EEPROMString[3];
int  EEMEM EEPROMINT;
char iForLetter;
uint8_t timerOverflowCount=0,sec;
uint8_t start,end,totalTime,penaltyTime;
int timerstart =1;
char result[5];
```

```
char scoreVal[5];
char highscoreToString[5];
int b1 = 0;
int n = 0;
int tioT;
int currentTime;
int penaltyDelay;
int highScore1;
int iForChoosing;

char newName[4];
typedef int bool;
#define true 1
#define false 0

bool pressedB1,notPressedB1,pressedB2,
scoreWritten,gamePlayed,gameReset,choosingLetters,pressedB2L,pressedB1L;


typedef struct{
	int playerScore;
	char player[3];
}Highscores;

Highscores arr[4];
int int_cmp (const void * a, const void * b)
{
	Highscores *scoreA = (Highscores *)a;
	Highscores *scoreB = (Highscores *)b;
	return ( scoreA->playerScore - scoreB->playerScore );
}
Highscores EEMEM EEStruct;
int main(void)
{

	TCNT0=0x00;
	TCCR0 = (1<<CS02);
	init();
	displayOn();
	functionSet();
	clear_display();
	timer0_init();
	sei();
```

```
initVals();
readEeprom();
while(1)
{

while(BUTTON1_PRESSED){
        button1();
        timeStart();
        //buzzer();
        redDiod();
        clear_display();
        resetVals();
        setDDRAM(0b10000000);
        choosingLetters = true;
        write_string("Your time is: ");
}
        pressedB1 = false;

        /*****************************************************************************/
        /* Spelet körs                                                               */
        /*****************************************************************************/
        while (n == 1)
        {
                while (BUTTON1_PRESSED)
                {
                        button1();

                }
                pressedB1 = false;



                if (TWEEZERS)
                {
                        redDiod();
                        buzzer();
                        penalty();
                }

                currentTime = sec - start + penaltyTime;
                itoa(currentTime,result,10);
                setDDRAM(0xc0);
                write_char(result[0]);
```

```
                if (result[1]!=0)
                {
                        setDDRAM(0xc1);
                        write_char(result[1]);
                }
                if (result[2]!=0)
                {
                        setDDRAM(0xc2);
                        write_char(result[2]);
                }
                scoreWritten = false;
                gamePlayed = true;
                        while (BUTTON2_PRESSED)
                        {
                                button2();
                        }
                        pressedB2 = false;
        }

        if(gamePlayed == true && gameReset == false){
                timeEnd();
                gamePlayed = false;
        }

        if(scoreWritten == false){
                clear_display();

                highScore();
                scoreWritten = true;
        }

        while (BUTTON2_PRESSED)
                        {
                                resetScoreboard();
                        }
                        pressedB2 = false;
    }
}

void init(){
    DDRA = 0b11111111; //Port A output
    _delay_ms(2);
```

```
	DDRB = 0b11101111; //Port B biderictional for 0-2, output for 3-4 and input for 5
		_delay_ms(2);
	DDRB = 0xFF; //Port B output
		_delay_ms(2);
	DDRD = 0xFF; //Port D output
		_delay_ms(2);
	 DDRB |= 1<<PB3;
		_delay_ms(2);
	 DDRB |= 1<<PB4;
		_delay_ms(2);
	 DDRB &= ~(1<<PB5); //ändrade från DDRD till DDRB
		_delay_ms(2);
}

void initVals(){
	pressedB1 = true;
	notPressedB1 = true;
	scoreWritten = false;
	penaltyDelay = -6;
	highScore1 = 10000;
	gamePlayed = false;
	pressedB2 = true;
	gameReset = false;
	choosingLetters = true;
	pressedB2L =true;
	pressedB1L = true;

	strcpy(arr[0].player,"NUL");
	arr[0].playerScore = 255;

	strcpy(arr[1].player,"NUL");
	arr[1].playerScore = 255;

	strcpy(arr[2].player,"NUL");
	arr[2].playerScore = 255;

	strcpy(arr[3].player,"NUL");
	arr[3].playerScore = 255;
	iForLetter = 0x65;
}
void buzzer(){
	PORTB ^= 1<<PB4;
```

```
}

void redDiod(){
	PORTB ^= 1<<PB3;
}

void highScore(){
	setDDRAM(0x80);
	write_string("Highscores!");
	qsort(arr, 4, sizeof(Highscores),int_cmp);
	if (arr[0].playerScore == 255)
	{
	setDDRAM(0xc0);
	 write_string("No games played yet!");
	}else{


	setDDRAM(0xC0);
	itoa(arr[0].playerScore,scoreVal,10);
	write_char(arr[0].player[0]);
	write_char(arr[0].player[1]);
	write_char(arr[0].player[2]);
	write_char(0x20);
	write_string(scoreVal);
	if (arr[1].playerScore!= 255)
	{
	setDDRAM(0x94);
	itoa(arr[1].playerScore,scoreVal,10);
	write_char(arr[1].player[0]);
	write_char(arr[1].player[1]);
	write_char(arr[1].player[2]);
	write_char(0x20);
	write_string(scoreVal);
	}
	if (arr[2].playerScore!=255)
	{
	setDDRAM(0xD4);
	itoa(arr[2].playerScore,scoreVal,10);
	write_char(arr[2].player[0]);
	write_char(arr[2].player[1]);
	write_char(arr[2].player[2]);
	write_char(0x20);
	write_string(scoreVal);
```

```
        }

        }
        writeToEeprom();
}
void readEeprom(){
        for (int j=0;j<3;j++)
        {
        for (int i= 0; i<3;i++)
        {
                arr[j].player[i] = eeprom_read_byte((uint8_t*) i+(j*3));
        }
        }

        for (int i = 0;i<3;i++)
        {
                arr[i].playerScore = eeprom_read_byte((uint8_t*) i+9);
        }


}
void writeToEeprom(){
        int j = 0;
        for(int i= 0; i<3;i++)
        {
                eeprom_write_byte ((uint8_t*) 0+j*3, arr[i].player[0]);
                eeprom_write_byte ((uint8_t*) 1+j*3, arr[i].player[1]);
                eeprom_write_byte ((uint8_t*) 2+j*3 , arr[i].player[2]);
                j++;
        }

        for (int i=0; i<3;i++)
        {
                eeprom_write_byte((uint8_t*) 9+i, arr[i].playerScore);
        }

}
void button1(){
        if (pressedB1 == false)
        {

        if (b1==1)
        {
```

```
                b1=0;
        }else{
                b1 = 1;
        }
                switch(b1){
                        case 0: n = 0 ;
                        break;
                        case 1: n = 1 ;
                        break;
                }
        }
        pressedB1 = true;
}

void button2(){
        if(pressedB2 == false){
        if(n == 0){
                highScore1 = 10000;
        }else{
                n = 0;
        }
        }
        b1 =0;
        pressedB2 = true;
        gameReset = true;
}

// initialize timer, interrupt and variable
void timer0_init()
{
        // set up timer with prescaler = 256
        TCCR0 |= (1 << CS02);

        // initialize counter
        TCNT0 = 0;

        // enable overflow interrupt
        TIMSK |= (1 << TOIE0);

        // enable global interrupts


        // initialize overflow counter variable
```

```
	timerOverflowCount = 0;
}
void resetVals(){
	penaltyDelay = -6;
	penaltyTime = 0;
	gameReset = false;

}
ISR(TIMER0_OVF_vect)
{
	// keep a track of number of overflows
	timerOverflowCount++;
	if (timerOverflowCount >= 122)
	{
		timerOverflowCount = 0;
		sec++;
	}
}


void timeStart(){
	start = sec;
}
void timeEnd(){

		highScore1=currentTime;
		arr[3].playerScore = highScore1;
		clear_display();
		setDDRAM(0x80);
		write_string("Write your name");
		if(arr[2].playerScore>arr[3].playerScore){
			chooseLetter();
		}
		strcpy(arr[3].player,newName);
}

void chooseLetter(){

	iForChoosing = 0;
	iForLetter = 0x41;
	while (choosingLetters == true)
	{
		if(BUTTON2_PRESSED){
```

```
                while(BUTTON2_PRESSED){
                button2ForLetters();
        }
                }
                if(BUTTON1_PRESSED){
                while(BUTTON1_PRESSED){
                button1ForLetters();
        }
                }
        newName[iForChoosing]=iForLetter;
        if (iForChoosing == 0)
        {
                setDDRAM(0xc0);
                write_char(newName[iForChoosing]);
        }else if (iForChoosing == 1)
        {
                setDDRAM(0xc1);
                write_char(newName[iForChoosing]);
        }else if (iForChoosing == 2)
        {
                setDDRAM(0xc2);
                write_char(newName[iForChoosing]);
        }else{
                choosingLetters = false;
        }
        pressedB1L = false;
        pressedB2L = false;
        }


}

void resetScoreboard(){
        strcpy(arr[0].player,"NUL");
        arr[0].playerScore = 255;

        strcpy(arr[1].player,"NUL");
        arr[1].playerScore = 255;

        strcpy(arr[2].player,"NUL");
        arr[2].playerScore = 255;

        strcpy(arr[3].player,"NUL");
```

```
	arr[3].playerScore = 255;
	scoreWritten = false;
}

void button1ForLetters(){
	if(pressedB1L == false){
	iForChoosing++;
	}
	pressedB1L = true;
	iForLetter = 0x41;
}

void button2ForLetters(){
		if(pressedB2L == false){
		if (iForLetter<0x5A)
		{
			iForLetter++;
		}else{
		iForLetter = 0x41;
	   }


	}
	pressedB2L = true;
}
void penalty(){
	if (penaltyDelay+6 < currentTime)
	{
		penaltyTime = penaltyTime +5;
		penaltyDelay = currentTime;
	}
}

void displayOn(){
	write_cmd(0b00001100);
	_delay_ms(10);
}

void functionSet(){
	write_cmd(0b00111100);
	_delay_us(60);
}
```

```
void displayOff(){
	write_cmd(0b00001000);

}

void setDDRAM(char c){
	write_cmd(c);
	_delay_us(60);
}

void write_cmd(char c){
	RS_L;
	RW_L;
	E_H;
	PORTD = c;
	E_L;
	E_H;
}

void write_char(char c){
	RS_H;
	RW_L;
	E_H;
	PORTD =c;
	E_L;
	E_H;
	_delay_us(60);
}

void read_cmd(char c){
	RS_H;
	RW_H;
	E_H;
	PORTD = c;
	E_L;
	E_H;
}

void write_string(char string[]){
	int i = 0;
	while(string[i] !='\0'){
```

```
                write_char(string[i]);
                i++;
        }
}

void time_keep(){

}

void clear_display(){
        write_cmd(0b00000001);
        _delay_ms(4);
}
```