SIMON SAYS

Projekt i EITF11 Digitala Projekt

Oskar Strömberg I13 & Olof Svanemur I13

Lunds Tekniska Högskola Institutionen för elektro- och informationsteknik
Handledare: Bertil Lindvall

Sammanfattning

Rapporten beskriver genomförandet av ett projekt i kursen digitala projekt på LTH. Projektet består av monteringen och programmeringen av ett spel ”Simon Says”. Spelet går ut på att memorera ordningen i vilken fyra dioder har blinkat, för att sedan upprepa denna ordning i knapptryckningar. Rapporten beskriver vilken hårdvara som använts, samt den övergripande arbetsprocessen under monteringen av komponenterna och kodningen av mjukvaran. Rapporten innehåller även ett kopplingsschema och källkod i bilagor samt de lärdomar gruppen dragit av projektet.

Innehållsförteckning

1. Inledning

1.1 Bakgrund

Efter viss eftertanke valde gruppen att göra ett Simon Says spel, ett spel där dioder lyser i en viss kombination och spelaren skall trycka ner knappar i den ordningen dioderna lös. Efter konsultation med handledare fastslogs en lämplig svårighetsgrad på projektet och genomförandet inleddes.

1.2 Problemformulering

Projektet skall skapa ett mindre spel där lysdioder lyser i en slumpmässig kombination och efter dioderna lyst skall spelaren trycka ner knapparna i den ordningen dioderna lös. Om spelaren trycker ner fel knapp förlorar spelaren och då skall dioderna blinka rött. För att starta upp ett nytt spel ska spelaren kunna trycka på en knapp brevid displayen och när väl ett nytt spel inletts står det skrivet på skärmen vilken nivå spelet är på. För varje omgång spelaren klarar ökar antalet dioder som lyser med ett och snabbt blir spelet förvånansvärt svårt. Om man slår det tidigare rekordet, räknat som högsta påbörjade nivån, ska det nya rekordet synas på displayen.

2. Kravspecifikation

Kravspecifikationen ställdes upp innan projektet påbörjades för att utgöra ett ramverk för hur stort projektet skulle vara. Den har sedan anpassats under projektets gång när nya val har gjorts.

2.1 Funktionella krav

- Systemet ska ha en alfanumerisk display.
    - Displayen visar nuvarande nivå, högsta rekordet samt en meny.
- Systemet ska ha en meny-knapp som antingen startar spelet eller går tillbaka till menyn beroende på om spelet är igång eller ej.
- Systemet ska ha fyra dioder.
- Varje diod har en motsvarande knapp.

2.2 Kvalitetskrav

- Spelet ska fungera enligt följande:
    - Vid nivå 1 ska 2 blinkningar memoreras av spelaren.
        - Antalet blinkningar ökar med 1 för varje avklarad nivå.
    - Dioderna blinkar i slumpmässig ordning i början av varje nivå.

    - Spelaren ska upprepa ordningen genom att trycka ner respektive diods knapp i samma ordning som dioderna blinkade.
- Dioderna ska blinka med ett lämpligt tidsintervall.

3. Hårdvara

I bilaga 1 finns ett kopplingsschema som beskriver hur komponenterna har kopplats.

3.1 Processor

Processorn som användes är av modellen ATmega16 som har sammanlagt 40 stycken pinnar. Fyra av dessa är reserverade för koppling till JTAG. Processorn är kompatibel med JTAG och har även ett EEPROM minne där information kan lagras.

3.2 Display

En SHARP Dot-Matrix eller Alfanumerisk teckendisplay användes i projektet för att skriva ut tecken och information til spelaren.

3.3 Dioder

Fyra stycken dioder användes som kunde lysa med färgerna rött, grönt, blått eller vitt. Där vitt innebär att rött, blått och grönt lyser samtidigt.

3.4 Knappar

I projektet användes 4 stycken större röda knappar för att spelaren enkelt skulle kunna ange vilken av de fyra dioderna som blinkat. En mindre grå knapp användes för att spelaren skulle kunna starta ett nytt spel och avbryta ett pågående spel.

3.5 OR-grind

För att hantera knapptrycken användes OR-grindar för att notera när en knapp trycktes ned, som initierar en interrupt i mjukvaran.

4. Mjukvara

Programmeringen av spelet gjordes i programmeringsspråket C, drivet av programmet Atmel Studio 6.2. Detta program användes för att kunna utföra debug av koden och tillsammans med JTAG se vad som skedde i hårdvaran. JTAG användes för att ladda över koden till processorn och även exekvera den, detta möjliggjorde felsökningar och stegvisa tester.

Koden består av flertalet metoder för att utföra blinkningar av de olika dioderna i olika färger samt diverse kombinationer av blinkningar. Det finns även hantering av det pågående spelet, knapptryckningar och skärmen. Koden i sin helhet finns i bilaga 2.

5. Genomförande

5.1 Planering

I projektets förstadie krävdes en planering för att avgöra vad som skulle göras. Val av projekt funderades över under flera veckor för att slutligen falla på detta spel lagom till att projektet skulle starta. För att sedan skapa en stadig bas att utgå ifrån gjordes en kravspecifikation som beskrev systemet, en lista över de komponenter som skulle användas samt en ritning över hur de skulle kopplas.

5.2 Montering

Under monteringen skapades först en approximativ bild av var samtliga komponenter skulle placeras för att spelet skulle bli praktiskt att spela och samtidigt se estetiskt bra ut. När det fastslagits var alla komponenter skulle sitta lödades de viktigaste komponenterna fast. Sedan lödades stift fast på plattan och dessa kopplades senare under projektet till jord eller spänning för att på så sätt underlätta tråddragningen. Stiften användes även som knytpunkter i de fall när flera komponenter behövde kopplas samman.

Monteringen fortsatte med att koppla in dioderna och tillhörande resistorer för att försöka få dessa att fungera. Sedan kopplades knapparna in och den tillhörande logiken till knapparna trådades till processorn. Parallellt med inkopplingen av dioderna programmerades flera av de kombinationer som dioderna skulle kunna blinka i. Slutligen kopplades display ihop med processorn, när det framstod som att alla trådar var korrekt dragna påbörjades programmeringen av displayen.

5.3 Programmering

Programmeringen var en successiv process av lärande hur C-programmering och processorn fungerade. Den första metoden blev framslumpningen av vilken ordning dioderna skulle lysa, vilket kunde ha sparats till senare eftersom att metoden inte kunde testas med dioderna. Därefter programmerades metoder för att blinka dioderna i olika färger och i olika ordningar. Detta var ett viktigt steg för att det möjliggjorde test som visade att dioderna fungerade. Även metoder för knapphantering skapades och varje knapp fick output i form av att motsvarande diod blinkade för att visa att trycket hade registrerats. Detta följdes av koden för att hantera spelet, hantera displayen och till sist att spara det högsta rekordet på processorns långa minne.

5.4 Motgångar

Under monteringen av komponenten behövdes den ursprungliga ritningen omformas flera gånger allt eftersom ändringar genomfördes. Den största motgången under monteringen var när gruppen konstaterade att med de dåvarande komponenterna behövdes fler pinnar på processorn än vad fanns tillgängligt. Efter konsultation med handledare byttes displayen ut till en alfanumeriska teckendisplay, vilket krävde åtskillig omdragning av trådar.

Under projektets gång stötte gruppen på fler motgångar. Bland annat körde projektet fast upprepade gånger och felsökningen kunde vara en omfattande process, då det ibland var svårt att identifiera vad som var problemet, kodens syntax eller logiken i koden. Det största problemet under kodningen var programmeringen av den alfanumeriska teckendisplayen, då behövde displayen ställas in på att ta emot 4 bitar åt gången istället för 8 bitar åt gången. Det tog en tid innan den alfanumeriska teckendisplayen började fungera och skriva ut de bokstäver som skulle skrivas ut. Till stor del berodde detta på ett missförstånd i hur kommandon till displayen skulle exekveras, men efter extra läsning i databladet löstes detta.

6. Diskussion och slutsats

Under projektets gång har gruppen fått ett helhetstänk kring hur man kan bygga upp ett fungerande spel från ide till slutprodukt. Det har varit mycket lärorikt att arbeta metodiskt med elektroniska komponenter och få en liten bild av hur man kan bygga en prototyp. Gruppen känner sig nöjd med spelet och tycker att det fungerar som förväntat.

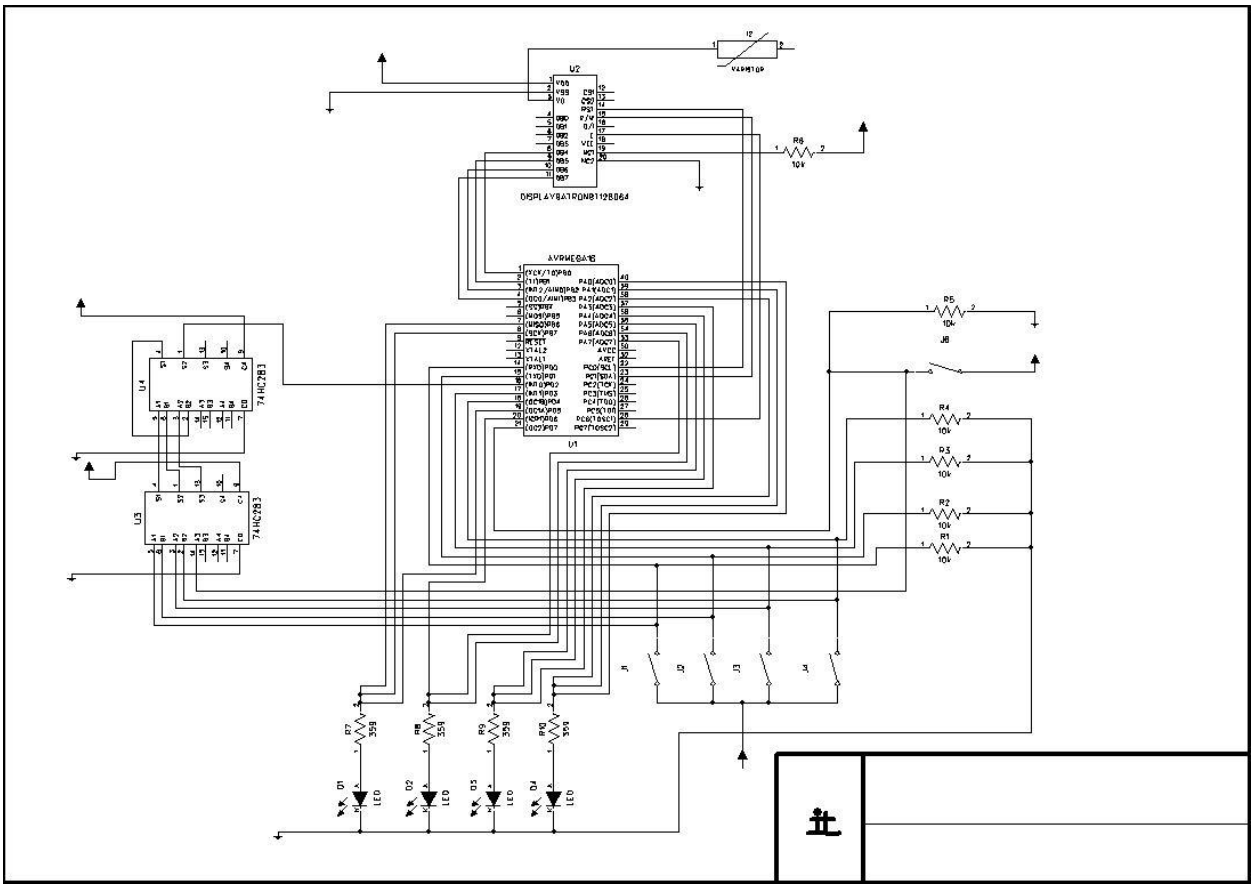
6.1 Lärdomar

Det är väldigt viktigt att ha en klar och tydlig kommunikation, både inom gruppen och med handledaren, eftersom att det är lätt att man missförstår varandra. Missförstår man varandra kan det leda till att man får komponenter som inte är lämpliga i sammanhanget. Exempelvis kunde inte den första displayen skicka 4 bitar åt gången, vilket vårt projekt krävde för att antalet pinnar på processorn ej skulle ta slut. Hade gruppen varit tydligare i kommunikationen kring vad projektet skulle åstadkomma hade detta problemet kunnat undvikas. Den andra lärdomen är att man bör läsa instruktioner (databladen) väldigt noga och göra exakt som de säger. Det går inte att på ett ungefär följa instruktioner i databladen.

Avslutningsvis bör man vara väldigt metodisk under genomförandet av arbetet för att på så sätt sakteligen utesluta felkällor. Det som framförallt tar tid och energi under arbetet är när man kör helt fast och inte vet vad som är fel. Om man kontinuerligt konstaterar vilka delar av komponenter som fungerar och vilken del av koden som fungerar kan man underlätta felsökningen markant. Under projektet försökte gruppen applicera denna principen och kunde på så sätt helt utesluta fel under monteringen och därav fokusera på vad som kunde potentiellt vara fel i programmeringen. Extra glad blir man när komponeneterna plötsligt fungerar efter lunchen.

Bilagor

Bilaga 1 - Kopplingsschema



Bilaga 2 - Källkod

```
/*
 * Kod_0.c
 *
 * Created: 2016-04-05 09:22:33
 *  Author: Oskar Strömberg & Olof Svanemur
 */

#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>

//Permament datalagring
```

```
uint8_t highscore = 1;
#define HIGHSCORE_ADDRESS 0x00

//Temporär datalagring
int orderNbr = 0;
int level =  0;
int orderSize = 1;
int order[100];
char success = 1;

//Program
char gameRunning = 0;
int click = 0;
int buttonClicks = 0;

void SetEHigh(){
	PORTC = PORTC | 0b01000000;
}

void SetELow(){
	PORTC = PORTC & 0b10111111;
}

void SetRSHigh(){
	PORTC = PORTC | 0b00000001;
}

void SetRSLow(){
	PORTC = PORTC & 0b11111110;
}

void SetRWHigh(){
	PORTC = PORTC | 0b00000010;
}

void SetRWLow(){
	PORTC = PORTC & 0b11111101;
}

void SetAllHigh(){
	SetEHigh();
	SetRSHigh();
	SetRWHigh();
}

void SetAllLow(){
	SetELow();
	SetRSLow();
	SetRWLow();
}
```

```
//Set up för displayen
void SetUpDisplay(){

	char c = 0b00000010;
	_delay_ms(15);
	ExecuteCmd(c);
	c = 0b00101000; //0b00101000
	_delay_ms(5);
	ExecuteCmd(c);


}

//Tömmer skärmen
void ClearDisplay(){
	ExecuteCmd(0b00000001);
	_delay_ms(5);
	_delay_ms(5);
}

//Startar skärmen, går till första menyn med visat start av spel och nuvarande rekord
void StartDisplay(){
	char c = 0b00001111;
	ExecuteCmd(c);
	ClearDisplay();
}

//Stänger av skärmen
void StopDisplay(){
	ClearDisplay();
	ExecuteCmd(0b00001000);
}

//Skicka in kommando som ska utföras
void ExecuteCmd(char c){
	char c1 = (PORTB & 0xF0)|((c>>4)&0x0F); //Övre fyra bitarna
	WriteCmd(c1);
	char c2 = (PORTB & 0xF0)|(c&0x0F); //Nedre fyra bitarna
	WriteCmd(c2);
}

//Skriv kommando för att ställa in displayen.
void WriteCmd(char c){
	SetRSHigh();
	SetRWHigh();
	SetELow();
	_delay_ms(1);
	SetRSLow();
	SetRWLow();
```

```
	_delay_ms(1);
	SetEHigh();
	_delay_ms(1);
	PORTB = c;
	SetELow();
	_delay_ms(1);
	SetRSHigh();
	SetRWHigh();
	_delay_ms(1);
	SetEHigh();

}

//Skriv data till RAM, för att skriva på displayen.
void WriteRam(char c){
	SetAllHigh();
	PORTB = c;
	SetRWLow();
	_delay_ms(1);
	SetELow();
	_delay_ms(1);
	SetEHigh();
	SetAllHigh();
}

//Skriv tecken på displayen.
void WriteChar(char c){
	char c1 = (PORTB & 0xF0)|((c>>4) & 0x0F);
	WriteRam(c1);
	char c2 = (PORTB & 0xF0)|(c & 0x0F);
	WriteRam(c2);
}

//Slumpar fram ordningen som dioderna ska blinka samt vilken färg som ska blinka
void Randomize()
{
	int place = 0;
	while(place < level){
		order[place] = rand() % 16;
		place++;
	}
}

void FirstBlueBlink(){
	PORTA = 0b00000001;
	_delay_ms(1000);
	PORTA = 0b00000000;
	_delay_ms(500);
}
```

```
void FirstGreenBlink(){
	PORTA = 0b00000010;
	_delay_ms(1000);
	PORTA = 0b00000000;
	_delay_ms(500);
}

void FirstRedBlink(){
	PORTA = 0b00000100;
	_delay_ms(1000);
	PORTA = 0b00000000;
	_delay_ms(500);
}

void FirstWhiteBlink(){
	PORTA = 0b00000111;
	_delay_ms(1000);
	PORTA = 0b00000000;
	_delay_ms(500);
}

void SecondBlueBlink(){
	PORTA = 0b00001000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	_delay_ms(500);
}

void SecondGreenBlink(){
	PORTA = 0b00010000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	_delay_ms(500);
}

void SecondRedBlink(){
	PORTA = 0b00100000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	_delay_ms(500);
}

void SecondWhiteBlink(){
	PORTA = 0b00111000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	_delay_ms(500);
}

void ThirdBlueBlink(){
```

```
        PORTA = 0b01000000;
        _delay_ms(1000);
        PORTA = 0b00000000;
        _delay_ms(500);
}

void ThirdGreenBlink(){
        PORTA = 0b10000000;
        _delay_ms(1000);
        PORTA = 0b00000000;
        _delay_ms(500);
}

void ThirdRedBlink(){
        PORTD = 0b01000000;
        _delay_ms(1000);
        PORTD = 0b00000000;
        _delay_ms(500);
}

void ThirdWhiteBlink(){
        PORTA = 0b11000000;
        PORTD = 0b01000000;
        _delay_ms(1000);
        PORTA = 0b00000000;
        PORTD = 0b00000000;
        _delay_ms(500);
}

void FourthBlueBlink(){
        PORTD = 0b00100000;
        _delay_ms(1000);
        PORTD = 0b00000000;
        _delay_ms(500);
}

void FourthGreenBlink(){
        PORTB = 0b10000000;
        _delay_ms(1000);
        PORTB = 0b00000000;
        _delay_ms(500);
}

void FourthRedBlink(){
        PORTB = 0b01000000;
        _delay_ms(1000);
        PORTB = 0b00000000;
        _delay_ms(500);
}
```

```
void FourthWhiteBlink(){
	PORTB = 0b11000000;
	PORTD = 0b00100000;
	_delay_ms(1000);
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(500);
}

void AllBlink(){
	PORTA = 0b00010001;
	PORTB = 0b11000000;
	PORTD = 0b01100000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(100);
	PORTA = 0b11100010;
	PORTB = 0b00000000;
	PORTD = 0b01100000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(100);
	PORTA = 0b01111100;
	PORTB = 0b10000000;
	PORTD = 0b00000000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(100);
	PORTA = 0b10001111;
	PORTB = 0b01000000;
	PORTD = 0b00000000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(100);
}

void AllBlueBlink(){
	PORTA = 0b01001001;
	PORTB = 0b00000000;
	PORTD = 0b00100000;
	_delay_ms(1000);
	PORTA = 0b00000000;
```

```
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(100);
}

void AllGreenBlink(){
	PORTA = 0b10010010;
	PORTB = 0b10000000;
	PORTD = 0b00000000;
	_delay_ms(500);
	PORTA = 0b00000000;
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(100);
}

void AllRedBlink(){
	PORTA = 0b00100100;
	PORTB = 0b01000000;
	PORTD = 0b01000000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(100);
}

//Blinkar alla dioder vitt en gång
void AllWhiteBlink(){
	PORTA = 0b11111111;
	PORTB = 0b11000000;
	PORTD = 0b01100000;
	_delay_ms(1000);
	PORTA = 0b00000000;
	PORTB = 0b00000000;
	PORTD = 0b00000000;
	_delay_ms(100);
}

//Blinkar motsvarande diod grönt en gång
void GreenBlink(int button){
	switch(button){
		case 0:
		FirstGreenBlink();
		break;

		case 1:
		SecondGreenBlink();
		break;
```

```
                case 2:
                ThirdGreenBlink();
                break;

                case 3:
                FourthGreenBlink();
                break;

                case 4:
                AllGreenBlink();
                break;
        }
}

//Blinkar motsvarande diod rött en gång
void RedBlink(int button){
        switch(button){
                case 0:
                FirstRedBlink();
                break;

                case 1:
                SecondRedBlink();
                break;

                case 2:
                ThirdRedBlink();
                break;

                case 3:
                FourthRedBlink();
                break;

                case 4:
                AllRedBlink();
                break;
        }
}

//5 röda blinkningar med alla dioder vid förlust
void LoseBlink(){
        int i = 0;
        while(i < 5){
                AllRedBlink();
                i++;
        }
}

//5 gröna blinkningar med alla dioder vid vunnen nivå
void WinBlink(){
```

```
	int i = 0;
	while(i < 5){
		AllGreenBlink();
		i++;
	}
}

//Blinkar den framslumpade sekvensen
void BlinkSequence(){
	int i = 0;
	orderNbr = 0;
	while(i < orderSize){
		i++;
		switch(order[orderNbr]){
			case 0:
			FirstBlueBlink();
			break;

			case 1:
			FirstGreenBlink();
			break;

			case 2:
			FirstRedBlink();
			break;

			case 3:
			FirstWhiteBlink();
			break;

			case 4:
			SecondBlueBlink();
			break;

			case 5:
			SecondGreenBlink();
			break;

			case 6:
			SecondRedBlink();
			break;

			case 7:
			SecondWhiteBlink();
			break;

			case 8:
			ThirdBlueBlink();
			break;
```

```
                        case 9:
                        ThirdGreenBlink();
                        break;

                        case 10:
                        ThirdRedBlink();
                        break;

                        case 11:
                        ThirdWhiteBlink();
                        break;

                        case 12:
                        FourthBlueBlink();
                        break;

                        case 13:
                        FourthGreenBlink();
                        break;

                        case 14:
                        FourthRedBlink();
                        break;

                        case 15:
                        FourthWhiteBlink();
                        break;
                }
                orderNbr++;
        }
}

//Jämför tryckt knapp med vilken diod som lystes upp på nuvarande plats.
void ButtonCompare(int button){
        buttonClicks++;
        int diodNbr = order[orderNbr] / 4;
        if(button != diodNbr){
                RedBlink(button);
                success = 0;
                LoseBlink();
        } else {
                GreenBlink(button);
        }
        orderNbr++;
}

//Nollställer nivån inför nytt spel.
void SetUpGame(){
        gameRunning = 1;
        success = 1;
```

```
	orderNbr = 0;
	orderSize = 1;
	level = 0;
}

//Loop som väntar medans spelaren trycker på knapparna, avbryter då fel knapp trycks
// eller hela sekvensen har tryckts korrekt
void CheckButtons(){
	orderNbr = 0;
	buttonClicks = 0;
	while(success && (buttonClicks <= level)){
		_delay_ms(10);
	}
}

//Startar en ny spelomgång
void Play(){
	SetUpGame();
	AllBlink();
	while(success){
		_delay_ms(500);
		orderNbr = 0;
		level++;
		orderSize++;
		SetPlayDisplay();
		_delay_ms(2000);
		Randomize();
		BlinkSequence();
		CheckButtons();
		if(success){
			WinBlink();
		}
	}
	gameRunning = 0;
	if(level > highscore){
		highscore = level;
		eeprom_write_word((uint8_t*)HIGHSCORE_ADDRESS,highscore);
	}
	level = 0;
	SetMenuDisplay();
}

//Går till menyn.
void Menu(){
	gameRunning = 0;
	SetMenuDisplay();

}

//Skriver ut Avsluta
```

```
void WriteAvsluta(char c){
	ClearDisplay();
	ExecuteCmd(c); //0b10000001
	WriteChar('A'); //A
	WriteChar('v'); //v
	WriteChar('s'); //s
	WriteChar('l'); //l
	WriteChar('u'); //u
	WriteChar('t'); //t
	WriteChar('a'); //a
}

//Skriver ut Start
void WriteStart(){
	ClearDisplay();
	WriteChar('S');
	WriteChar('t');
	WriteChar('a');
	WriteChar('r');
	WriteChar('t');
}

//Skriver ut rekord
void WriteHighscore(char k){
	ExecuteCmd(k); //0b10000011, 0x94
	WriteChar('R');
	WriteChar('e');
	WriteChar('k');
	WriteChar('o');
	WriteChar('r');
	WriteChar('d');
	WriteChar(':');

	//Beräkning av ascii för rekord
	int i = highscore/10;
	char c;
	if(i > 0){
		char b = 48 + i;
		WriteChar(b);
		int rest = highscore%10;
		c = 48 + rest;
	} else {
		c = 48+highscore;
	}

	WriteChar(c);
}

//Skriver ut nivå
void WriteLevel(char k){
```

```
	ExecuteCmd(k); //0b10000001, 0xD4
	WriteChar('N');
	WriteChar('i');
	WriteChar('v');
	WriteChar('a');
	WriteChar(':');

	//Beräkning av ascii för nivå
	int i = level/10;
	char c;
	if(i > 0){
		char b = 48 + i;
		WriteChar(b);
		int rest = level%10;
		c = 48 + rest;
	} else {
		c = 48+level;
	}

	WriteChar(c);
}

//Visar Display för spelomgång
void SetPlayDisplay(){
	WriteAvsluta(0x82);
	WriteHighscore(0x96);
	WriteLevel(0xD6);
}

//Visar menyns display
void SetMenuDisplay(){
	WriteStart();
	WriteHighscore(0x94);
	WriteLevel(0xD4);
}

//Sparar highscore på processorns permanenta minne. //Överflödig, ta bort!
void SaveHighscore(){
	//Sida 20, datablad processor
}

void SetDataDirection(){
	DDRA = 0b11111111;
	DDRB = 0b11001111;
	DDRC = 0b01000011;
	DDRD = 0b01100000;
}

int main(void)
{
```

```
	SetDataDirection();
	SetUpDisplay();

	ClearDisplay();
	StartDisplay();

	PIND = 0b00000000;
	DDRD = 0b01100000;
	GICR = 0b01000000;
	MCUCR = 0b00000011;
	sei();

	highscore = eeprom_read_word((uint8_t*)HIGHSCORE_ADDRESS);
	if (highscore==0xFFFF){
		highscore=0;
	}

	Menu();

	while(1)
	{

	}
}

ISR(INT0_vect){

	if(click == 1){
		return;
	}

	click = 1;
	cli();

	_delay_ms(50);

	switch(PIND){
		case 0b00010100:
			if(gameRunning){
				ButtonCompare(0);
			}

			break;

		case 0b00001100:
			if(gameRunning){
				ButtonCompare(1);
			}

			break;
```

```
		case 0b00000110:
			if(gameRunning){
				ButtonCompare(2);
			}

		break;

		case 0b00000101:
			if(gameRunning){
				ButtonCompare(3);

			}

		break;

		case 0b10000100:
			AllBlueBlink();
			if(gameRunning){
				sei();
				click = 0;
				success = 0;
			} else {
				sei();
				click = 0;
				Play();
			}

		break;
	}

	sei();

	click = 0;
}
```