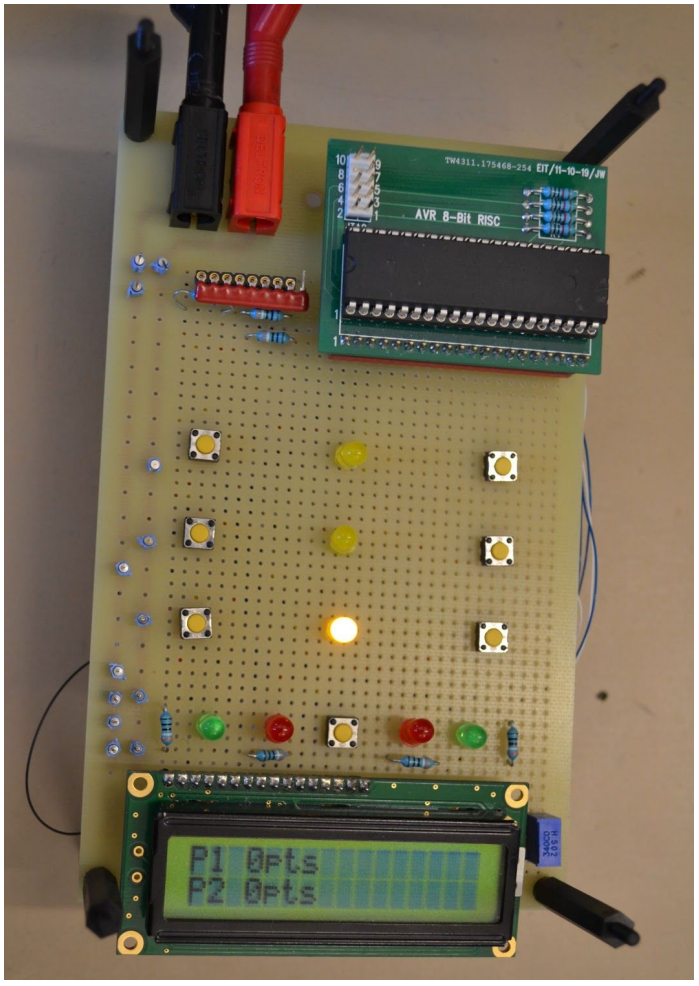
Lunds tekniska högskola
Elektro- och informationsteknik
EITF11 Digitala projekt

Nattduellen



Hugo Jernberg
Anne Lind
Emma Rydén

2018-05-15

Abstract

This report documents Nattduellen, a game that allows two opponents to compete in their ability to react the fastest. The game was developed as a project within the course Digital Systems at the Faculty of Engineering at Lund University.

The game consists of three diodes that are turned on randomly, and each opponent has three buttons corresponding to the diodes. When a diode is turned on, the player who presses their button first receives one point. The game continues until one player has three, four or five points, depending on how many rounds they have decided to play.

A crucial component in Nattduellen is the display, that shows the current score and latest reaction times. Furthermore, Nattduellen is controlled by the microcontroller ATmega16 which was programmed using C. Learning this language was one of the project's biggest challenges, but apart from this, the game was finished on time without difficulties.

Innehållsförteckning

Inledning

Nattduellen är ett spel som går ut på att reagera så snabbt som möjligt när en diod tänds. Namnet är taget från SVT:s program Mästarnas mästare i vilket Nattduellen är en tävling där två deltagare så fort som möjligt ska gripa tag i en kloss när dess lampa slocknar. Spelet är utvecklat inom kursen Digitala Projekt vid Lunds tekniska högskola.

Kravspecifikation

Spelet är konstruerat för två spelare och den spelare som först trycker på rätt knapp får ett poäng. Om den som trycker först trycker på fel knapp får istället den andra spelaren ett poäng.

Den ursprungliga kravspecifikationen innehöll följande:

- Spelet ska tre gula dioder. Dessa ska styras av processorn och i varje spelomgång ska någon av dem tändas efter en slumpmässig tid inom tre till sju sekunder.
- Spelet ska innehålla sex knappar som hör ihop med de gula dioderna, tre knappar per spelare. Det ska även finnas en sjunde knapp som startar om spelet.
- Spelet ska ha fyra ytterligare dioder, två röda och två gröna. Dessa ska visa för respektive spelare om denne tryckt på rätt eller fel knapp när en diod har tänts.
- Spelet ska ha en display som visar spelarnas poäng.
- Spelet ska mäta spelarnas reaktionstider och dessa ska skrivas ut på displayen när de har tryckt på någon knapp, oavsett om de tryckt rätt eller fel.

När arbetet gick bättre än förväntat utökades kraven med:

- Innan varje spel ska displayen visa vilken spelare som är spelare 1 respektive 2.
- Innan varje spel ska man kunna välja om spelet ska vara bäst av 5, 7 eller 9 spelomgångar.
- Om ingen spelare har reagerat när tre sekunder har passerat släcks dioden och spelomgången börjar om. Ingen av spelarna tilldelas någon poäng, och spelomgången räknas inte.

Teori

För att få spelets elektronik och logik att fungera behövde dess hårdvara och mjukvara utvecklas och kombineras. Mjukvaran skrevs i ett effektivt och maskinnära programspråk, nämligen C, och den utvecklades i programmet Atmel Studio 7. Den fullständiga källkoden återfinns i bilaga 2 och under rubriken Mjukvara finns en kortare beskrivning av dess huvudsakliga beståndsdelar. Under rubriken Hårdvara finns en beskrivning av spelets elektronik.

Hårdvara

Spelet består av ett antal elektriska komponenter. Hur dessa kopplats visas i kopplingsschemat i bilaga 1 och i referensförteckningen finns relevanta datablad.

Processor

Konstruktionens processor är en AVR ATmega 16, som är en 8-bitars microcontroller. Den har fyra portar med totalt 40 pinnar.

JTAG

JTAG användes som den fysiska länken mellan datorn där mjukvaran programmerades och själva spelet. Mjukvaran utvecklades i Atmel Studio 7.

Display

Spelet har en display av typen GDM 1602K. Det är en display med två rader där totalt 32 tecken kan visas.

Potentiometer

För att justera ljusstyrkan på displayen används ett variabelt motstånd av typen Bourns 3386.

Knappar

Spelet har totalt 7 knappar, tre till varje spelare och en som fungerar som en omstartsknapp.

Dioder

Totalt 7 dioder ingår i konstruktionen. Av dessa är tre gula, och utgör själva spelet. Sedan används två röda och två gröna dioder, som visar för respektive spelare om denne tryckt på rätt eller fel knapp efter att en gul diod tänts.

Resistorer

Resistorer används för att få rätt spänning över dioderna. Dessutom används ett multimotstånd till knapparna.

Mjukvara

Mjukvaran utgår från en main-metod som ligger sist i källkoden. Denna metod börjar med att definiera vilka av processorns pinnar som ska vara output och vilka som ska vara input. Eftersom alla pinnar som används i port B och D går till displayen respektive lysdioderna sätts dessa till output, och detsamma gäller de pinnar som används i port C. Spelets knappar finns kopplade till port A, så dessa sätts till input. Därefter följer ett metodanrop som ser till att programmets avbrottsrutin fungerar som den ska, vilket innebär att en global variabel som håller koll på tiden räknas upp med ett varje millisekund. Detta görs med hjälp av processorns inbyggda timer. Efter det säkerställs det att skärmen är tömd, innan det skrivs ut ett välkomstmeddelande med texten “Click reset to start”.

Huvuddelen av programmet ligger därefter inuti en oändlig loop, som endast bryts när spänningen kopplas från. Inuti denna väntar programmet först på att användaren ska klicka på reset-knappen för att starta spelet, innan information om vem som är spelare 1 respektive

spelare 2 skrivs ut. Sedan får användaren välja hur många spelomgångar som ska spelas, och efter det börjar spelet. När någon spelare når den poäng som krävs för att vinna meddelas detta på displayen, innan nästa genomlöpning av programmets oändliga loop sätter igång och programmet väntar på att reset-knappen ska tryckas ner ännu en gång.

Genomförande

Planeringen inleddes ett par månader innan själva bygget genomfördes. I ett tidigt skede bestämdes vilken sorts prototyp som skulle göras och valet föll på ett spel. En övergripande planering av hur spelet skulle fungera och utformas gjordes också tidigt. Precis innan monteringen inleddes gjordes ett kopplingsschema med alla komponenter inritade.

Montering

Monteringen inleddes med att planera var komponenterna skulle sitta i förhållande till varandra. Sedan löddes processorn och displayen fast, och deras kopplingar till varandra samt till jord och spänning sattes på plats. Efter detta kopplades dioder, motstånd och knappar in. Samtliga kopplingar är gjorda genom att vira med en virpistol. Det sista som gjordes var att koppla på spänningen och kontrollera att alla pinnar, på såväl processor och display som dioder och knappar, hade rätt spänning. Då upptäcktes till exempel att multimotståndet var trasigt och det fick bytas mot ett nytt.

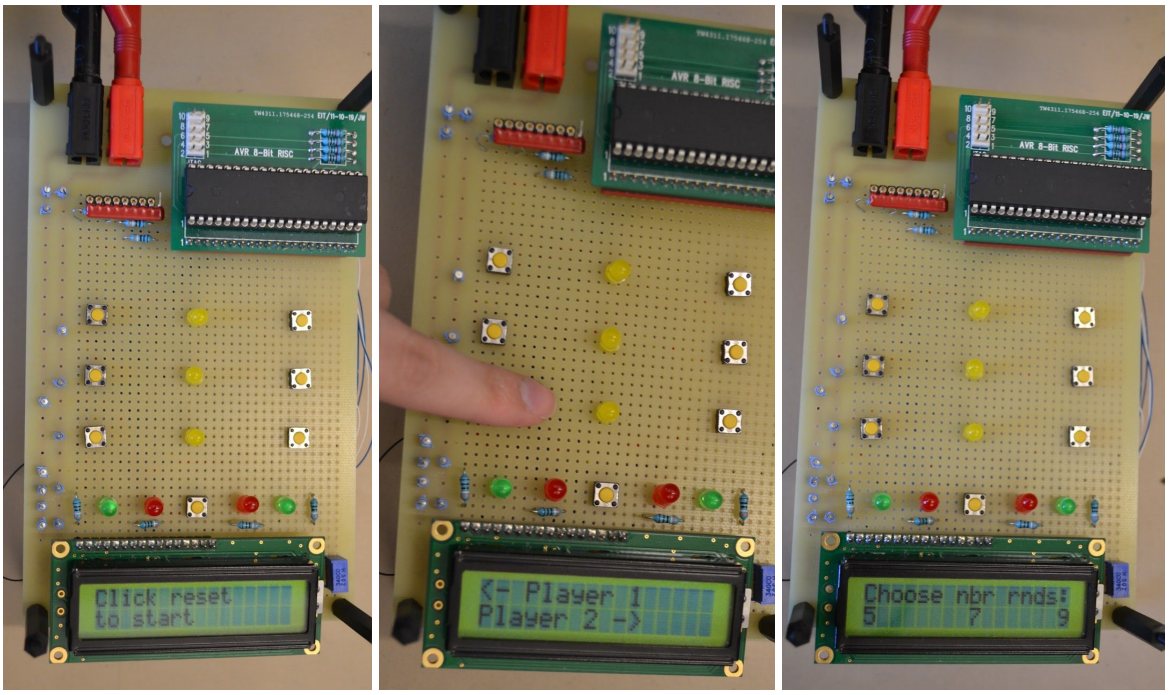
Programmering

Processorn programmerades med hjälp av Atmel Studio 7. I början gjordes mycket som till slut inte hamnade i programmet för att kontrollera att kopplingarna var korrekta, till exempel testutskrifter på displayen, tändning och släckning av dioder samt att få något att ske på displayen när en knapp tryckts ned.

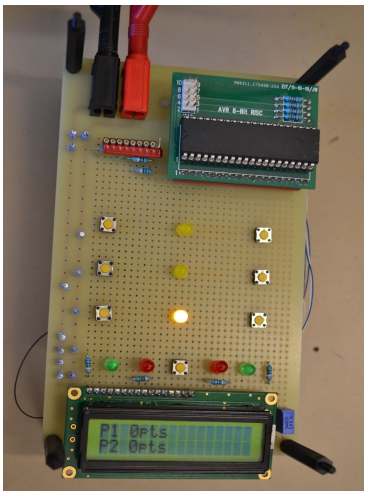
Efter detta inleddes arbetet med själva spelet. Först utvecklades en version utan tidtagning, och när denna fungerade som den skulle utökades programmet med tidtagning samt funktioner som att kunna välja hur många spelomgångar som ska köras i varje spel.

Resultat

Den färdiga prototypen visas längst till vänster i figur 1, när spänningen precis har kopplats in och det första välkomstmeddelandet har skrivits ut. Därefter har knappen reset trycks ner. Hur de efterföljande utskrifterna ser ut på displayen visas i mitten och till höger i figur 1.

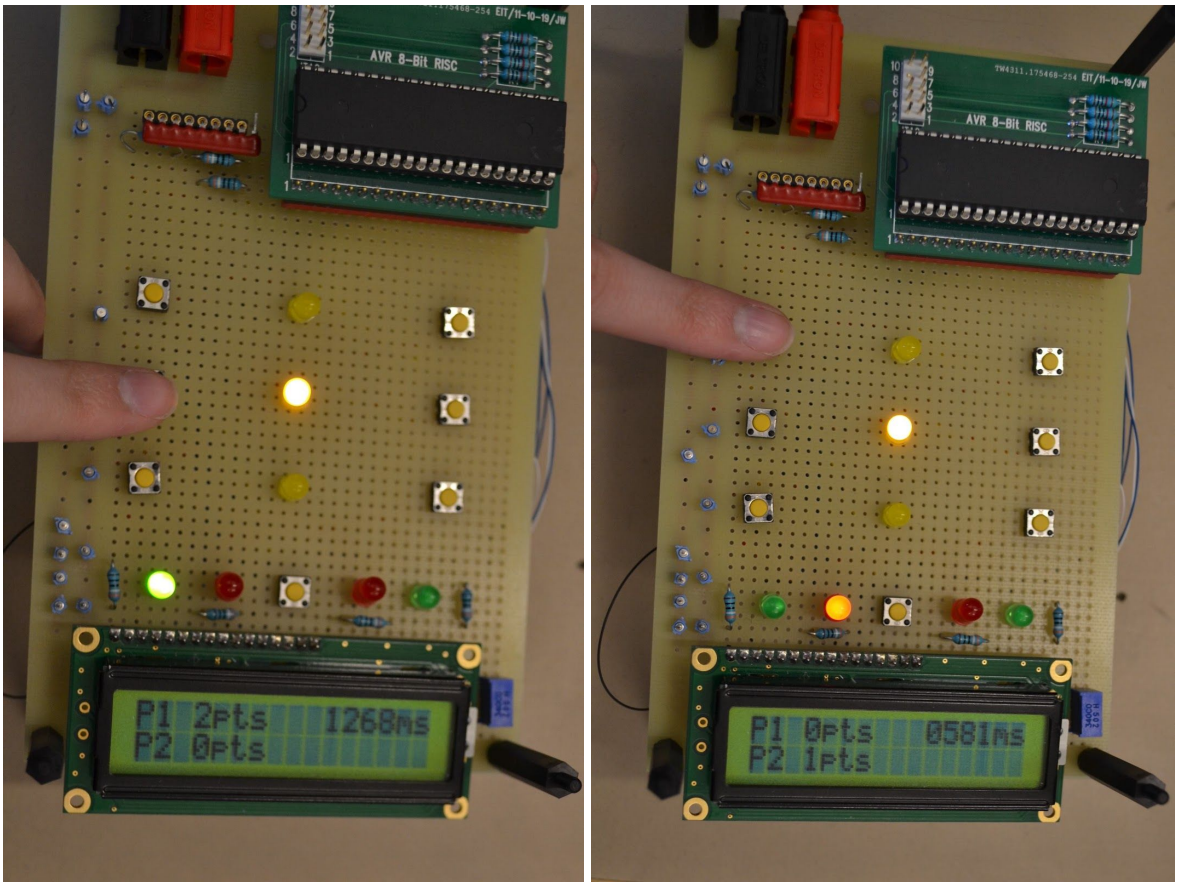


Figur 1: Nattduellens initiala utskrifter.



Spelet uppfyller samtliga krav som angavs i kravspecifikationen tidigare i rapporten. De gula lysdioderna tänds slumpmässigt som man kan se i figur 2, och spelet reagerar på ett lämpligt sätt när man trycker på rätt respektive fel knapp vilket man kan se till vänster respektive höger i figur 3. De gröna och röda dioderna tänds när de ska och displayen visar den information som behövs. Spelet avslutas när någon spelare har vunnit mer än hälften av det valda antalet spelomgångar.

Figur 2: Hur spelet ser ut när den första gula lysdioden har tänts.



Figur 3: Till vänster har spelare 1 tryckt på rätt knapp och tilldelats ett poäng. Spelarens gröna lysdiod lyser och reaktionstiden har skrivits ut. Till höger har spelare 1 tryckt på fel knapp och spelare 2 har tilldelats ett poäng. Reaktionstiden har skrivits ut, men nu lyser istället den röda lysdioden.

Diskussion

Arbetet med detta projekt har varit väldigt lärorikt, då ingen av gruppmedlemmarna gjort något liknande tidigare. Vi har under projektets gång fått både praktisk kunskap, som att koppla samman komponenter och löda, och teoretisk kunskap, som att läsa datablad och att programmera i C. Som tidigare nämnt flöt arbetet på bättre än förväntat och vi kunde därför ägna tid åt att programmera in extra funktioner som inte ingick i den ursprungliga kravspecifikationen.

Vissa delmoment har varit svårare än andra, till exempel att lära sig att programmera i C och särskilt att programmera direkt till en processor. Dessutom hade vi en del problem med kopplingen mellan datorn och processorn när vår JTAG inte alltid var helt samarbetsvillig. Detta skapade en hel del frustration och ett antal timmar har lagts på att försöka få kontakt mellan dator och processor. En annan stor utmaning var att sätta sig in i hur processorns inbyggda timer fungerar, som vi använde till tidtagningen.

Sammanfattningsvis tycker vi att detta var ett roligt projekt och vi är nöjda med vår prototyp. Vidareutvecklingar av den skulle kunna vara att spara resultat från matcher och skapa en lista över de bästa eller att ha ett läge där det endast är en spelare som tävlar mot sig själv genom att spara sin snabbaste reaktionstid och försöka slå den.

Referensförteckning

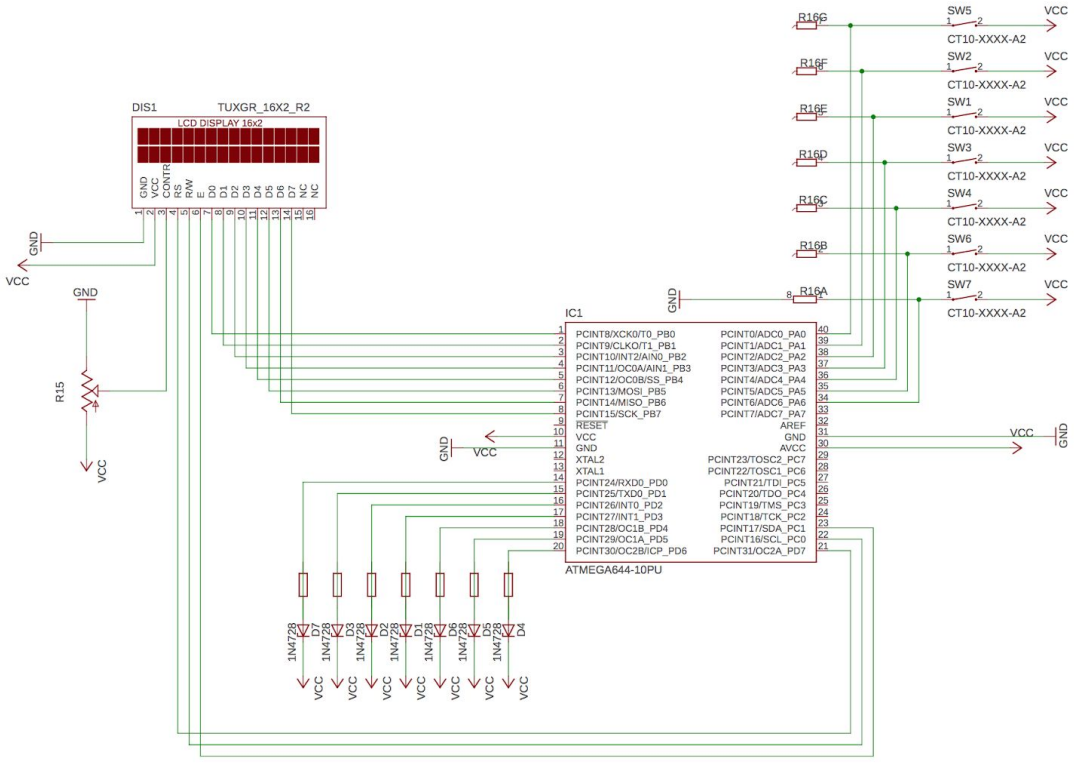
Datablad till processorn ATmega16:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf
Hämtad 2018-03-26

Datablad till displayen GDM 1602K:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf
Hämtad 2018-03-26

Datablad till potentiometern Bourns 3386:
https://www.bourns.com/pdfs/3386.pdf
Hämtad 2018-03-26

Bilaga 1: Kopplingsschema

I figur 4 nedan finns en schematisk bild över hur prototypens komponenter är sammankopplade. Hela konstruktionen utgår från processorn, som totalt har fyra portar, port A-D. Till port A är spelets sju knappar kopplade, samt ett multimotstånd som är gemensamt för samtliga knappar. Till port B är displayen kopplad. Port C har några pinnar som har lämnats orörda, eftersom dessa reserverades för den JTAG som användes för att kommunicera med datorn som mjukvaran utvecklades på. Några av pinnarna som tillhör port C har dessutom använts för att koppla in displayen. Slutligen är prototypens samtliga lysdioder kopplade till processorns port D.



Figur 4: Nattduellens kopplingsschema.

Bilaga 2: Källkod

```
#include <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>


char nextRow = 0x40;            //Displayens adress till första tecknet på andra raden
char p1Points = 0b00110000;
char p2Points = 0b00110000;
int nbrOfGames = 0;             //Antal spelade omgångar
int selection = 0;              //Valt antal spelomgångar
int t = 0;

char diode1 = 0b00000001;
char diode2 = 0b00000010;
char diode3 = 0b00000100;
char green1 = 0b00001000;
char red1 = 0b00010000;
char green2 = 0b00100000;
char red2 = 0b01000000;

char p1k1 = 0b00000010;         //Spelare 1, knapp 1
char p1k2 = 0b00000100;
char p1k3 = 0b00001000;
char p2k1 = 0b00010000;
char p2k2 = 0b00100000;
char p2k3 = 0b01000000;

void setRSHigh()
{
      PORTD = PORTD | 0b10000000;
}

void setRSLow()
{
      PORTD = PORTD & 0b01111111;
}

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

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

void setEHigh()
{
      PORTC = PORTC | 0b00000010;
```

```
}

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

/*Skickar det som just nu står i PORTB till displayen */
void write()
{
	setELow();
	_delay_ms(2);
	setEHigh();
}

/*Flyttar positionen på displayen*/
void setAddress(char c)
{
	setRSLow();
	setRWLow();
	setEHigh();
	_delay_ms(2);
	PORTB = 0b10000000 + c;
	write();
}

/*Programmerar displayen och skriver ut "Click reset to start"*/
void setUpDisplay()
{
	setRSLow();
	setRWLow();
	setEHigh();
	PORTB = 0b00111000;
	write();
	PORTB = 0b00001100;
	write();

	setRSHigh();
	PORTB = 0b01000011; //C
	write();
	PORTB = 0b01101100; //l
	write();
	PORTB = 0b01101001; //i
	write();
	PORTB = 0b01100011; //c
	write();
	PORTB = 0b01101011; //k
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b01100101; //e
	write();
```

```
	PORTB = 0b01110100; //t
	write();

	setAddress(nextRow);
	setRSHigh();
	PORTB = 0b01110100; //t
	write();
	PORTB = 0b01101111; //o
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b01110100; //t
	write();
	PORTB = 0b01100001; //a
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b01110100; //t
	write();
}

/*Suddar allt som står på displayen*/
void clearDisplay()
{
	setRSLow();
	setRWLow();
	setEHigh();
	_delay_ms(2);
	PORTB = 0b00000001;
	write();
}

/*Visar vilken spelare som är vilken*/
void introduction()
{
	clearDisplay();
	setAddress(0x00);
	setRSHigh();
	PORTB = 0b00111100; //<
	write();
	PORTB = 0b00101101; //-
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01010000; //P
	write();
	PORTB = 0b01101100; //l
	write();
	PORTB = 0b01100001; //a
	write();
	PORTB = 0b01111001; //y
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110010; //r
	write();
```

```
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00110001; //1
	write();

	setAddress(nextRow);
	setRSHigh();
	PORTB = 0b01010000; //P
	write();
	PORTB = 0b01101100; //l
	write();
	PORTB = 0b01100001; //a
	write();
	PORTB = 0b01111001; //y
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00110010; //2
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00101101; //-
	write();
	PORTB = 0b00111110; //>
	write();
}

/*Nollställer spelarnas poäng och antalet spelade omgångar och skriver ut
	P1 0pts
	P2 0pts
 */
void newGame()
{
	clearDisplay();
	setRSHigh();
	PORTB = 0b01010000; //P
	write();
	PORTB = 0b00110001; //1
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00110000; //0
	write();
	PORTB = 0b01110000; // p
	write();
	PORTB = 0b01110100; //t
	write();
	PORTB = 0b01110011; //s
	write();

	setAddress(nextRow);
	setRSHigh();
	PORTB = 0b01010000; //P
	write();
```

```
	PORTB = 0b00110010; //2
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00110000; //0
	write();
	PORTB = 0b01110000; //p
	write();
	PORTB = 0b01110100; //t
	write();
	PORTB = 0b01110011; //s
	write();
	nbrOfGames = 0;
	p1Points = 0b00110000;
	p2Points = 0b00110000;
}

/*Skriver ut att spelare 1 vann*/
void playerWins(int k)
{
	clearDisplay();
	setRSHigh();
	PORTB = 0b01010000; //P
	write();

	if (k == 1)
	{
		PORTB = 0b00110001; //1
		write();
	}
	if (k == 2)
	{
		PORTB = 0b00110010; //2
		write();
	}
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110111; //w
	write();
	PORTB = 0b01101001; //i
	write();
	PORTB = 0b01101110; //n
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b00100001; //!
	write();

	setAddress(nextRow);
	setRSHigh();
	PORTB = 0b01001110; //N
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110111; //w
	write();
	PORTB = 0b00111010; //:
	write();
```

```
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01000011; //C
	write();
	PORTB = 0b01101100; //l
	write();
	PORTB = 0b01101001; //i
	write();
	PORTB = 0b01100011; //c
	write();
	PORTB = 0b01101011; //k
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110100; //t
	write();
}

/*Ger spelare 1 en poäng och ökar poängen på displayen*/
void p1Point()
{
	char place = 0x03;
	setAddress(place);
	p1Points++;
	PORTB = p1Points;
	setRSHigh();
	write();
}

/*Ger spelare 2 en poäng och ökar poängen på displayen*/
void p2Point()
{
	char place = 0x43;
	setAddress(place);
	p2Points++;
	PORTB = p2Points;
	setRSHigh();
	write();
}

/*Hjälpmetod för att skriva ut en siffra på displayen*/
void printNbr(int k)
{
	PORTB = 0b00110000 + k;
	write();
}

/*Skriver ut reaktionstiden för spelare 1*/
void playerTime(int ms, int player)
{
```

```
	char place;
	if (player == 1)
	{
		place = 0x0A;
	}
	if (player == 2)
	{
		place = 0x4A;
	}
	setAddress(place);
	setRSHigh();
	int k1 = ms / 1000;
	printNbr(k1);
	int k2 = (ms / 100) % 10;
	printNbr(k2);
	int k3 = (ms / 10) % 10;
	printNbr(k3);
	int k4 = ms % 10;
	printNbr(k4);
	PORTB = 0b01101101; //m
	write();
	PORTB = 0b01110011; //s
	write();
}

/*Slumpar fram vilken diod som ska tändas*/
int randDiode()
{
	#define RAND_MAX = 2;
	return (rand() + 1);
}

/*Slumpar fram hur lång tid det ska ta innan dioden tänds*/
int randTime()
{
	#define RAND_MAX = 4000;
	return rand();
}

/*Tar bort reaktionstiderna från displayen*/
void clearTime()
{
	char place = 0x0A;
	setAddress(place);
	setRSHigh();
	for(int i = 0; i < 6; i++){
		PORTB = 0b10000000;
		write();
	}
	place = 0x4A;
	setAddress(place);
	setRSHigh();
	for(int i = 0; i < 6; i++){
		PORTB = 0b10000000;
		write();
	}
}
```

```
/*En spelomgång där en diod tänds*/
void gameRound()
{
	int b = randTime() / 10;
	_delay_ms(b + 3000);
	int a = randDiode() / 10000;
	if (a == 0)
	{
		PORTD = 0b00000001;
	}
	if (a == 1)
	{
		PORTD = 0b00000010;
	}
	if (a == 2)
	{
		PORTD = 0b00000100;
	}
	clearTime();
	t = 0;

	int p1Pressed = 0;
	int p2Pressed = 0;
	while(1)
	{
		if (p1Pressed == 0 && (PINA & p1k1) == p1k1)
		{
			p1Pressed++;
			int t1 = t;
			playerTime(t1, 1);
			if (a == 0)
			{
				PORTD = PORTD | green1;
				if(p1Pressed > p2Pressed)
				{
					p1Point();
				}
			}
			else
			{
				PORTD = PORTD | red1;
				if(p1Pressed > p2Pressed)
				{
					p2Point();
				}
			}
		}
		if (p1Pressed == 0 && (PINA & p1k2) == p1k2)
		{
			p1Pressed++;
			int t1 = t;
			playerTime(t1, 1);
			if (a == 1)
			{
				PORTD = PORTD | green1;
				if(p1Pressed > p2Pressed)
				{
					p1Point();
```

```
            }
        }
        else
        {
            PORTD = PORTD | red1;
            if(p1Pressed > p2Pressed)
            {
                p2Point();
            }
        }
    }
    if (p1Pressed == 0 && (PINA & p1k3) == p1k3)
    {
        p1Pressed++;
        int t1 = t;
        playerTime(t1, 1);
        if (a == 2)
        {
            PORTD = PORTD | green1;
            if(p1Pressed > p2Pressed)
            {
                p1Point();
            }
        }
        else
        {
            PORTD = PORTD | red1;
            if(p1Pressed > p2Pressed)
            {
                p2Point();
            }
        }
    }
    if (p2Pressed == 0 && (PINA & p2k1) == p2k1)
    {
        p2Pressed++;
        int t2 = t;
        playerTime(t2, 2);
        if (a == 0)
        {
            PORTD = PORTD | green2;
            if(p1Pressed < p2Pressed)
            {
                p2Point();
            }
        }
        else
        {
            PORTD = PORTD | red2;
            if(p1Pressed < p2Pressed)
            {
                p1Point();
            }
        }
    }
    if (p2Pressed == 0 && (PINA & p2k2) == p2k2)
    {
        p2Pressed++;
```

```
                int t2 = t;
                playerTime(t2, 2);
                if (a == 1)
                {
                        PORTD = PORTD | green2;
                        if(p1Pressed < p2Pressed)
                        {
                                p2Point();
                        }
                }
                else
                {
                        PORTD = PORTD | red2;
                        if(p1Pressed < p2Pressed)
                        {
                                p1Point();
                        }
                }
        }
        if (p2Pressed == 0 && (PINA & p2k3) == p2k3)
        {
                p2Pressed++;
                int t2 = t;
                playerTime(t2, 2);
                if (a == 2)
                {
                        PORTD = PORTD | green2;
                        if(p1Pressed < p2Pressed)
                        {
                                p2Point();
                        }
                }
                else
                {
                        PORTD = PORTD | red2;
                        if(p1Pressed < p2Pressed)
                        {
                                p1Point();
                        }
                }
        }
        if (p1Pressed == 1 && p2Pressed == 1)
        {
                _delay_ms(2500);
                PORTD = 0x00;
                break;
        }
        if (t > 3000)
        {
                PORTD = 0x00;
                nbrOfGames--;
                break;
        }
    }
    nbrOfGames++;
}

/*Spelarna får välja hur många omgångar spelet ska pågå i*/
```

```
void chooseRounds() {
	clearDisplay();
	setRSHigh();
	PORTB = 0b01000011; // C
	write();
	PORTB = 0b01101000; //h
	write();
	PORTB = 0b01101111; //o
	write();
	PORTB = 0b01101111; //o
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01101110; //n
	write();
	PORTB = 0b01100010; //b
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b01101110; //n
	write();
	PORTB = 0b01100100; //d
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b00111010; //:
	write();

	char place = 0x40;
	setAddress(place);
	setRSHigh();
	PORTB = 0b00110101; //5
	write();
	place = 0x48;
	setAddress(place);
	setRSHigh();
	PORTB = 0b00110111; //7
	write();
	place = 0x4F;
	setAddress(place);
	setRSHigh();
	PORTB = 0b00111001; //9
	write();

	while(1)
	{
		char temp = PINA & p1k3;
		if (temp == p1k3)
		{
			selection = 5;
```

```
                newGame();
                break;
            }
            temp = PINA & 0x01;
            if (temp == 0x01)
            {
                selection = 7;
                newGame();
                break;
            }
            temp = PINA & p2k3;
            if (temp == p2k3)
            {
                selection = 9;
                newGame();
                break;
            }
        }
}

/*Sätter timern till att göra avbrott varje millisekund*/
void setTimer()
{
    TCCR1A = 0;
    TCCR1B = 0;
    TCNT1 = 0;
    OCR1A = 999;
    TCCR1B |= (1 << WGM12);
    TCCR1B |= (1 << CS11);
    TIMSK |= (1 << OCIE1A);
    sei();
}

/*Ökar t med 1 vid varje avbrott*/
ISR(TIMER1_COMPA_vect)
{
    t++;
}

int main(void)
{
    DDRD = 0xFF;
    DDRA = 0x00;
    DDRB = 0xFF;
    DDRC = 0x03;

    setTimer();

    clearDisplay();
    setUpDisplay();
    srand(time(NULL));

    while(1)
    {
        while(1)
        {
            char temp = PINA & 0x01;
            if (temp == 0x01)
```

```
            {
                newGame();
                break;
            }
        }

        introduction();
        _delay_ms(3000);
        chooseRounds();

        while(nbrOfGames < selection)
        {
            gameRound();
            if (p1Points == 0b00110000 + selection / 2 + 1)
            {
                playerWins(1);
                break;
            }
            if (p2Points == 0b00110000 + selection / 2 + 1)
            {
                playerWins(2);
                break;
            }
        }
    }
}
```