Kösystem i MaskinerI:et

- *Lösningen till lunchkaoset i mikrovågsugnskön*

Abstract

This report describes the process of building and programming a queueing system to apply on the microwave ovens in MaskinerI:et at Lunds University, Faculty of Engineering. The idea origins from the known problems concerning the present queueing system, which mostly appear due to disorder in the queue and poor information regarding which microwave ovens are available.

The result of the project is a fully functional prototype that can be applied in real life after some improvements. Some of the assumptions that have been made do not fully correspond with reality and have to be taken into consideration during further development.

Skola: Lunds Tekniska Högskola
Kurs: Digitala projekt, EITF11
Institution: Institutionen för Elektro- och Informationsteknik
Handledare: Bertil Lindvall

Grupp: 13
Gruppmedlemmar: Elin Söderberg, Sofia Danielsson & Emma Ekström

Datum: 23 maj 2016

Innehållsförteckning

1. Inledning

1.1 Problembeskrivning

Vid Lunds Tekniska Högskola (LTH) har alla studenter lunch samtidigt, från klockan 12:00 till klockan 13:00. Ett populärt ställe att värma sin mat på är i lokalen MaskinerI:et. Då alla äter lunch samtidigt skapas ett stort tryck på de mikrovågsugnar som finns tillgängliga, vilket leder till att det skapas långa köer vid dessa. Det är då svårt att veta vem som står på tur att få värma sin mat. Dessutom är det svårt att se vilka mikrovågsugnar som är lediga, vilket beror på två saker:

1. Det finns många mikrovågsugnar, vilket gör det svåröverskådligt.
2. Flertalet studenter stänger dörren på mikron när de har värmt färdigt. Detta ger intrycket av att mikron är upptagen.

1.2 Syfte

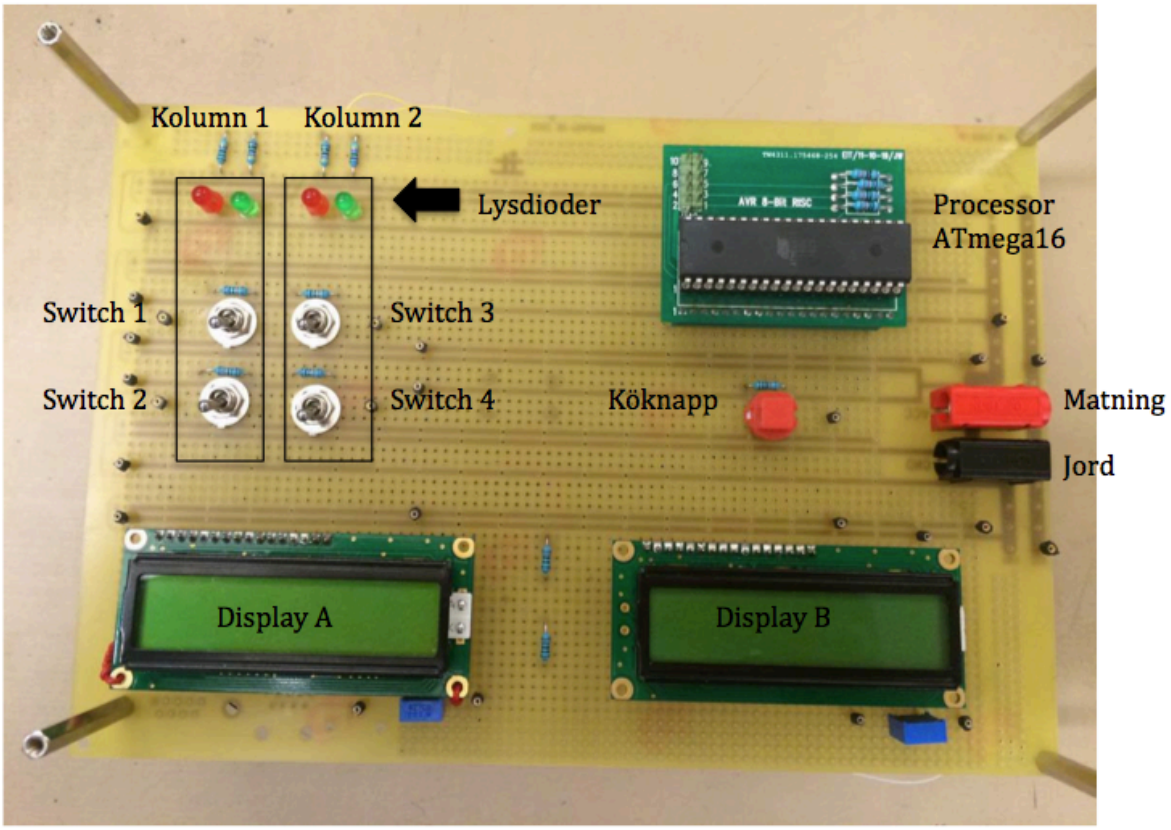
Syftet med detta arbete är att skapa en prototyp av ett kösystem som vid vidare utveckling kan appliceras på och därmed effektivisera användandet av mikrovågsugnar under lunchtid i MaskinerI:et.

1.3 Avgränsningar

Vid utformningen av prototypen, som är ett resultat av detta arbete, har ingen hänsyn tagits till estetik och hållbarhet.

2. Produktbeskrivning

Produkten består av ett kösystem som ska kunna appliceras på lunchkön vid mikrovågsugnarna i MaskinerI:et. I modellen används ett system av fyra mikrovågsugnar, uppdelade på två kolumner med två mikrovågsugnar i varje kolumn. I prototypen representeras varje mikrovågsugn av en switch. En switch som är på symboliserar att mikrovågsugnen är igång och vice versa. När alla mikrovågsugnar i en kolumn är upptagna lyser en röd diod vid denna kolumn. Om det finns någon ledig mikrovågsugn i kolumnen lyser en grön diod.



Figur 2.1 Fotografi av prototypen med beskrivning av komponenter.

Kösystemet är tänkt att användas på följande vis:

1. En student kommer till kön och trycker på köknappen.
2. På Display B visas studentens könummer som representerar dennes plats i kön.
3. Då en mikrovågsugn blir ledig visas könumret som tillhör den student som står först i kön och således har rätt att använda den lediga mikrovågsugnen på Display A.
4. Studenten sätter sin mat i den lediga mikrovågsugnen som då blir upptagen.
5. När mikrovågsugnen är färdig hämtar studenten sin mat och mikrovågsugnen blir ledig.

2.1 Antaganden

Antagande 1. Studenterna i kön uppför sig rationellt. Detta innebär t.ex. att varje student som anländer till kön trycker på köknappen för att få ett könummer och sedan ställer in sin matlåda i en mikrovågsugn när det givna könumret visas på Display A.

Antagande 2. Två mikrovågsugnar sätts aldrig på exakt samtidigt.

Antagande 3. Två mikrovågsugnar stängs aldrig av exakt samtidigt.

Antagande 4. När en persons könummer visas på Display A tar denne en ledig mikrovågsugn som då blir upptagen. När en persons mat är färdigvärmd hämtar denne sin mat. Båda dessa händelser antas ske instantant.

3. Hårdvara

Processor, ATmega16
ATmega 16 8-bit Microcontroller processor med 40 ben.

Switchar
Systemet innehåller fyra switchar som representerar mikrovågsugnar.

Displays
Två stycken SHARP Dot-Matrix displays. På dessa visas det könummer som erhålls då man ställer sig i kön och på den andra visas numret till den näst på tur att värma.

Dioder
Två röda dioder och två gröna dioder, som signalerar om det finns lediga mikrovågsugnar eller ej.

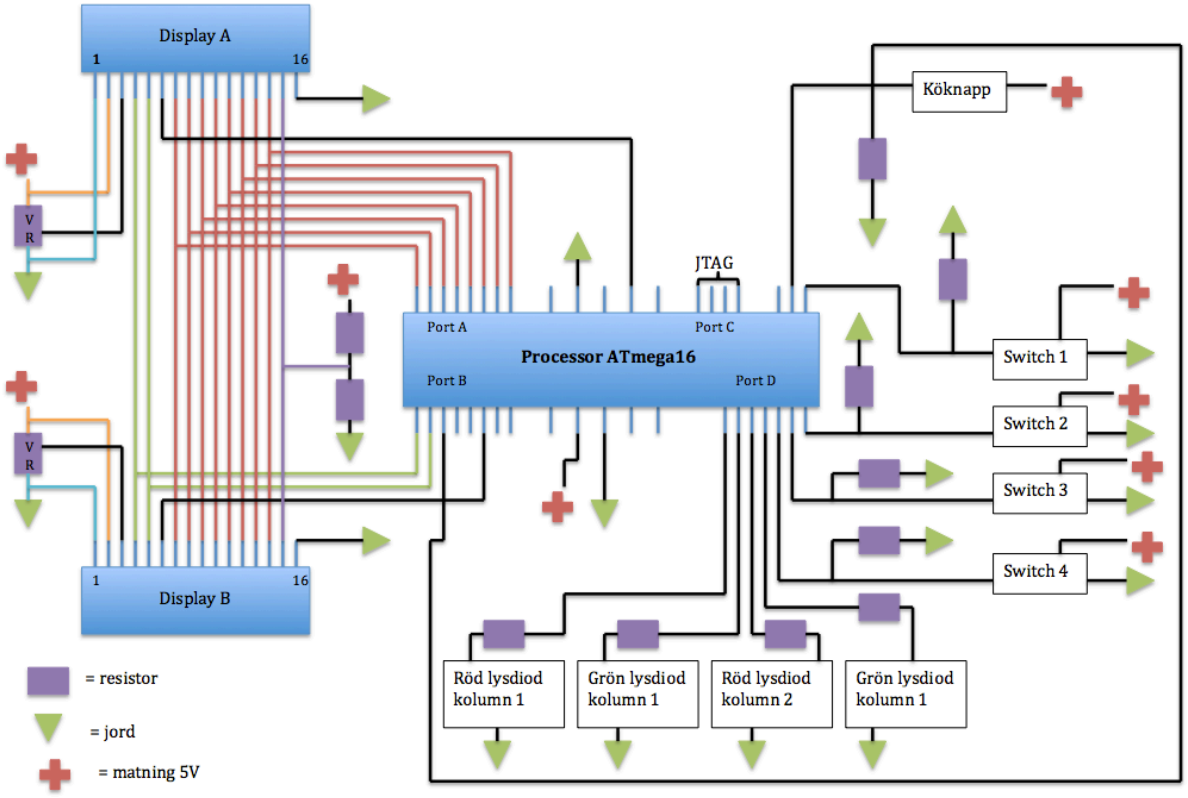
Sladdar
Vanlig kopplingstråd.

Bottenplatta
Här fästs alla komponenter.

Köknapp
En knapp användes, som vid intryckning ska generera nästa könummer, som visas på Display B.

Resistorer
Totalt användes åtta resistorer med olika resistans samt två variabla resistorer.

3.1 Kopplingsschema



Figur 3.1 Kopplingsschema

4. Kravspecifikation

Krav 1. När man trycker på köknappen ska nästa nummer i ordningen visas på Display A.

Krav 2. När alla mikrovågsugnar i en kolumn är upptagna ska den röda lampan vid denna kolumn lysa och den gröna vara släckt.

Krav 3. När det finns minst en ledig mikrovågsugn i en kolumn ska den gröna lampan vid denna kolumn lysa och den röda vara släckt.

Krav 4. När en switch slås på ska systemet förstå att den mikrovågsugn som denna switch symboliserar är upptagen.

Krav 5. När en switch slås av ska systemet förstå att den mikrovågsugn som denna switch symboliserar är ledig.

Krav 6. När en mikrovågsugn blir ledig ska nämaste könummer i turordningen visas på Display B.

Krav 7. När könumrena når 200 så börjar räkningen om från början på 1.

Krav 8. Om man trycker på köknappen då man är först i kön och det finns lediga mikrovågsugnar ska könumret direkt visas på Display A.

5. Genomförande

Arbetet inleddes med att rita ett kopplingsschema, se figur 3.1. Detta för att underlätta det framtida kopplingsarbetet. När detta var gjort på ett korrekt sätt erhölls verktygslådan, som bland annat innehöll diverse användbara mätinstrument, kopplingstrådar och verktyg. Vid sidan av denna erhölls även komponenterna enligt avsnitt 2.1. Efter detta påbörjades arbetet med att koppla enligt kopplingsschemat. De större komponenterna, så som displayer och processor lödades fast vid plattan.

Då alla komponenter kopplats ihop påbörjades arbetet med att koda programmet till processorn. Till en början skrevs kod för att kontrollera att dioderna lyser då processorn skickar ut en etta på de ben som är kopplade till dioderna. Detta gjordes för att se till att kopplingen hade utförts rätt. Efter detta kontrollerades det att switcharna hade kopplats rätt genom att använda logikpennan och hålla den mot det ben som var kopplat till respektive switch. Då logikpennan visade låg spänning när switchen var av och hög spänning då switchen var på, kunde det konstateras att kopplingen var korrekt. Arbetet fortsattes sedan med att koda programmet för att få kösystemet att fungera som avsett. Detta innebar mycket läsning i databladen om hur de olika komponenterna fungerar och hur de ska kodas för att erhålla önskat resultat.

5.1 Tillstånd

För att underlätta kodningen ritades ett tillståndsdiagram som visas i figur 5.1. Cirklarna i tillståndsdiagrammet beskriver de olika tillstånden som de två kolumnerna med mikrovågsugnar i systemet kan befinna sig i. Pilarna mellan cirklarna beskriver hur tillståndet kan ändras på ett ögonblick. Varje pil representerar ett av fallen nedan. För varje fall beskrivs vad som ska hända i systemet när detta fall inträffar.

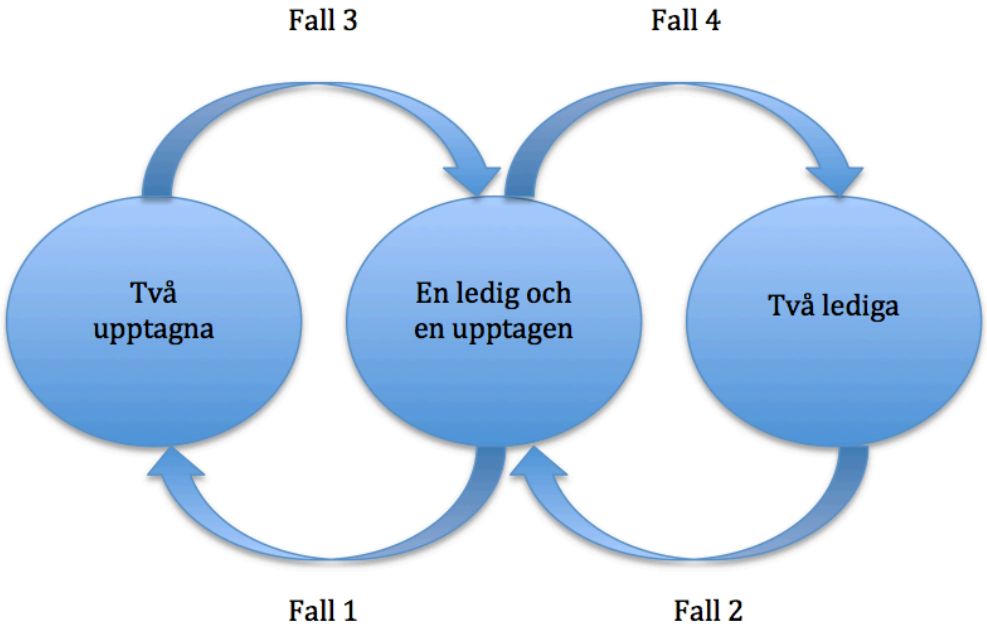
I beskrivningen av fallen används följande beteckningar:
FirstNbr = könumret som tillhör den som står först i kön, visas på Display A.
LastNbr = könumret som tillhör den som står sist i kön, visas på Display B.

Fall 1: går från en ledig och en upptagen till två upptagna
- Röd diod i motsvarande kolumn tänds
- Grön diod i motsvarande kolumn släcks

Fall 2: går från två lediga till en ledig och en upptagen
- Grön diod i motsvarande kolumn fortsätter vara tänd
- Röd diod i motsvarande kolumn fortsätter vara släckt

Fall 3: går från två upptagna till en ledig och en upptagen
- Grön diod i motsvarande kolumn tänds
- Röd diod i motsvarande kolumn släcks
- *FirstNbr* ökar med ett (sålänge det inte är större än eller lika med *LastNbr*)

Fall 4: går från en upptagen och en ledig till två lediga
- Grön diod i motsvarande kolumn fortsätter vara tänd
- Röd diod i motsvarande kolumn fortsätter vara släckt
- *FirstNbr* ökar med ett (sålänge det inte är större än eller lika med *LastNbr*)



Figur 5.1 Tillståndsdiagram

6. Mjukvara

Programmet är skrivet på programspråket C, i programmet Atmel Studio 6.2. För att kunna läsa över programmet från datorn till processorn användes en J-TAG.

6.1 Interrupts

Då köknappen trycks in genereras ett interrupt som bryter while-loopen i programmet. Det genereras även interna interrupts av timern i processorn med en fjärdedels sekunds mellanrum. Vid dessa interrupts undersöks om en mikrovågsugn har ändrat status, det vill säga blivit ledig eller blivit upptagen. Detta avgörs genom att se om signalerna från de olika switcharna är en nolla eller en etta, och jämföra med tidigare värde.

6.2 Metoder

Följande metoder finns i programmet:

clearDisplay1() rensar display A från tidigare tecken och ställer cursorn längst till vänster

clearDisplay2() rensar display B från tidigare tecken och ställer cursorn längst till vänster

showOnDisplay1(char nbr) beräknar den binära representationen av den symbol som ska visas på Display A och anropar writeData1 med denna representation som inparameter

showOnDisplay2(char nbr) beräknar den binära representationen av den symbol som ska visas på Display B och anropar writeData2 med denna representation som inparameter

writeCommand1(char c) ger kommandon till Display A

writeCommand1(char c) ger kommandon till Display A

writeData1(char c) ger data till Display A

writeData2(char c) ger data till Display B

7. Resultat

Detta projekt har resulterat i en fullt duglig prototyp av ett kösystem som uppfyller alla ovan nämnda krav samt som vid vidare utveckling kan appliceras på MaskinerI:ets mikrovågsugnsköer.

8. Diskussion

För att prototypen ska kunna applierceras på MaskinerI:et mikrovågsugnsköer så krävs det att samtliga antaganden under *2.1 Antaganden* är uppfyllda.

Enligt *Antagande 1* är en förutsättning för att lösningen ska fungera i verkligheten att alla ankommande studenter trycker på köknappen och därmed hamnar sist i kön samt att de sedan håller sig till de könummer de blivit tilldelade. Detta är inte fullt realistiskt i alla situationer och därför kan det inte garanteras att *Antagande 1* uppfylls i verkligheten.

Enligt *Antagande 2 och 3* klarar systemet inte av att två mikrovågsugnar sätts på eller stängs av exakt samtidigt. Detta händer inte i verkligheten eftersom tiden alltid kan delas in i mindre intervall, men i fallet med kösystemet är detta en begränsing eftersom kösystemet inte kan hantera alltför små tidsintervall. Vid vidare utveckling bör detta tas i åtanke.

Tiden det tar för en student att börja värma sin mat från det att dess könummer visats på Display A försummas enligt *Antagande 4*. Detta är dock inte rimligt i verkligheten och måste därmed tas i åtanke vid vidare utveckling av systemet. Ett alternativ är att använda en timer,

så att studenten har en viss tid på sig att sätta in sin mat i en ledig mikrovågsugn. Tiden det tar för en student att hämta sin färdigvärmda mat försummas också enligt *Antagande 4*. Detta är ett mindre problem eftersom nästa student enkelt kan ta ut den icke hämtade matlådan. Man kan även i detta fall använda en timer, men det är inte fullt nödvändigt. Alternativt kan en sensor som känner matlådans tyngd installeras.

Ytterligare en aspekt som bör tas i åtanke vid vidare utveckling av kömodellen är att switcharna måste ersättas med någon form av sensorer som känner av om mikrovågsugnarna är igång.

Sammantaget innebär detta att systemet har potential att med vidare utveckling kunna appliceras på MaskinerI:ets mikrovågsugnskö under lunchtid och syftet är därmed uppnått.

9. Referenslista

Datablad Display, http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf
Datablad Processor ATmega 16,
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf

10. Appendix

```
/*
* GccApplication1.c
*
* Created: 2016-04-07 10:48:36
*  Author: digpi13
*/


#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 8000000UL  // 8 MHz
#include <util/delay.h>

int queueButton;
int switch1;     //"boolean" 0 är false och 1 är true
int switch2;
int switch3;
int switch4;
int on1;           //1 då mikro 1 är upptagen
int on2;
int on3;
int on4;


int main(void)
{
                queueButton = 0;
                switch1 = 0;
                switch2 = 0;
                switch3 = 0;
                switch4 = 0;
                on1 = 0;
                on2 = 0;
                on3 = 0;
                on4 = 0;
                int redDiod1 = 0;
                int redDiod2 = 0;
                char firstNbr = 0;              //display a, aktuellt könummer
                char lastNbr = 0;               //display b, tilldelat könummer

                DDRA = 0b11111111;          //sätter alla PA-portar till ut.
                DDRB = 0b00100011;
                DDRC = 0b10000000;
                DDRD = 0b00001111;

                PORTB |= _BV(PB0);          //RS sätts till 1
                PORTB &= ~_BV(PB1);         //R/W sätts till 0
                PORTC |= _BV(PC7);  //sätter E till 1
                PORTB |= _BV(PB5);          //E sätts till 1

                GICR = 0b00100000; //enable external interrupt INT2
                MCUCSR = 0b01000000; // interrupt sense control ISC2 s. 67
                TCCR1A = 0b00000000;
                TCCR1B = 0b00000101;        //Timer med prescalor, genererar interrupts 4 gånger per
sekund
                TCNT1H = 0b11111000;
                TCNT1L = 0b01011111; // 63583 binärt, för att få interrupt med 1/4 sekunds mellanrum
```

```
		TIMSK = 0b00000100;
		TIFR = 0b00000100;
		sei();

		PORTD = 0b00001010;			//gröna dioder lyser och röda dioder är
släckta

		writeCommand1(0b00110000); //function set display a
		writeCommand2(0b00110000); //function set display b
		writeCommand1(0b00001110); //sätter på display a
		writeCommand2(0b00001110); //sätter på display b
		clearDisplay1();
		clearDisplay2();

  while(1)
  {
				if (queueButton == 1)
				{
					if(lastNbr == 0b11001000)	//om lastNbr = 200
					{
						lastNbr = 0b00000000;
					}
					lastNbr++;
					clearDisplay2();
					showOnDisplay2(lastNbr);
					queueButton = 0;
					if((on1 == 0 || on2 == 0 || on3 == 0 || on4 == 0) && firstNbr
== lastNbr-1)
					{
						firstNbr++;
						clearDisplay1();
						showOnDisplay1(firstNbr);

					}
				}

				if(switch1 == 1 || switch2 == 1)		//kolumn 1
				{
					if(on1 == 1 && on2 == 1)	//fall 1
					{
						PORTD &= ~_BV(PD1);
		//endast grön diod kolumn 1 släcks
						PORTD |= _BV(PD0);
		//endast röd diod kolumn 1 tänds
						redDiod1 = 1;
					}

					else if(on1 != on2)		//fall 2 och 3
					{
						if(redDiod1 == 1)//Fall 3
						{
							if(firstNbr < lastNbr)
							{
								firstNbr++;

		clearDisplay1();

		showOnDisplay1(firstNbr);
							}
						}
```

```
                                        PORTD &= ~_BV(PD0);
//endast röd diod kolumn 1 släcks
                                        PORTD |= _BV(PD1);
//endast grön diod kolumn 1 tänds
                                        redDiod1 = 0;
                                }

                                else if(on1 == 0 && on2 == 0) //fall 4
                                {
                                        PORTD &= ~_BV(PD0);
//endast röd diod kolumn 1 släcks
                                        PORTD |= _BV(PD1);
//endast grön diod kolumn 1 tänds
                                        redDiod1 = 0;
                                        if(firstNbr < lastNbr)
                                        {
                                                firstNbr++;
                                                clearDisplay1();
                                                showOnDisplay1(firstNbr);
                                        }
                                }
                                switch1 = 0;
                                switch2 = 0;
                }

                if(switch3 == 1 || switch4 == 1)                //kolumn 2
                {
                        if(on3 == 1 && on4 == 1)        //fall 1
                        {
                                PORTD &= ~_BV(PD3);
//endast grön diod kolumn 2 släcks
                                PORTD |= _BV(PD2);
//endast röd diod kolumn 2 tänds
                                redDiod2 = 1;
                        }

                        else if(on3 != on4)             //fall 2 och 3
                        {
                                if(redDiod2 == 1)//Fall 3
                                {
                                        if(firstNbr < lastNbr)
                                        {
                                                firstNbr++;

clearDisplay1();

showOnDisplay1(firstNbr);
                                                }
                                }
                                PORTD &= ~_BV(PD2);
//endast röd diod kolumn 2 släcks
                                PORTD |= _BV(PD3);
//endast grön diod kolumn 2 tänds
                                redDiod2 = 0;
                        }

                        else if(on3 == 0 && on4 == 0) //fall 4
                        {
                                PORTD &= ~_BV(PD2);
//endast röd diod kolumn 2 släcks
```

```
                                                        PORTD |= _BV(PD3);
                //endast grön diod kolumn 2 tänds
                                                        redDiod2 = 0;
                                                        if(firstNbr < lastNbr)
                                                        {
                                                                firstNbr++;
                                                                clearDisplay1();
                                                                showOnDisplay1(firstNbr);
                                                        }
                                        }
                                        switch3 = 0;
                                        switch4 = 0;
                                }
                }



}

void showOnDisplay1(char nbr)
{
                char nbrLeft = nbr;
                char nbr1 = nbrLeft%10;
                nbrLeft = nbrLeft/10;
                char nbr2 = nbrLeft%10;
                nbrLeft = nbrLeft/10;
                char nbr3 = nbrLeft%10;

                writeData1(48 + nbr3);
                writeData1(48 + nbr2);
                writeData1(48 + nbr1);
                return;
}

void showOnDisplay2(char nbr)
{
                char nbrLeft = nbr;
                char nbr1 = nbrLeft%10;
                nbrLeft = nbrLeft/10;
                char nbr2 = nbrLeft%10;
                nbrLeft = nbrLeft/10;
                char nbr3 = nbrLeft%10;

                writeData2(48 + nbr3);
                writeData2(48 + nbr2);
                writeData2(48 + nbr1);
                return;
}

void clearDisplay1()
{
                writeCommand1(0b00000001);
                return;
}

void clearDisplay2()
{
                writeCommand2(0b00000001);
                return;
}
```

```
void writeCommand1(char c)
{
		_delay_ms(10);
		PORTA = c;
		PORTB &= ~_BV(PB0); //sätter RS till 0

		PORTC &= ~_BV(PC7); //sätter E till 0
		PORTC |= _BV(PC7);  //sätter E till 1
		PORTB |= _BV(PB0);		//sätter RS till 1

		return;
}

void writeCommand2(char c)
{
		_delay_ms(10);
		PORTA = c;
		PORTB &= ~_BV(PB0);		//sätter RS till 0

		PORTB &= ~_BV(PB5);		//sätter E till 0
		PORTB |= _BV(PB5);  //sätter E till 1
		PORTB |= _BV(PB0);		//sätter RS till 1

		return;
}

void writeData1(char c)
{
		_delay_ms(1);
		PORTA = c;

		PORTC &= ~_BV(PC7);		//E sätts till 0
		PORTC |= _BV(PC7);		//E sätts till 1
		return;
}

void writeData2(char c)
{
		_delay_ms(1);
		PORTA = c;

		PORTB &= ~_BV(PB5);		//E sätts till 0
		PORTB |= _BV(PB5);		//E sätts till 1
		return;
}

ISR(TIMER1_OVF_vect)
{
		char val = PIND;
		char val1 = val&0b10000000;
		char val2 = val&0b01000000;
		char val3 = val&0b00100000;
		char val4 = val&0b00010000;
		if((val1 != 0 && on1 == 0) || (val1 == 0 && on1 != 0))
		{
				switch1 = 1;
				switch2 = 0;
				switch3 = 0;
				switch4 = 0;
```

```
                        if (val1 == 0)
                        {
                                on1 = 0; //switch 1 är av
                        } else
                        {
                                on1 = 1; //switch 1 är på
                        }
                }
                else if((val2 != 0 && on2 == 0) || (val2 == 0 && on2 != 0))
                {
                        switch1 = 0;
                        switch2 = 1;
                        switch3 = 0;
                        switch4 = 0;
                        if (val2 == 0)
                        {
                                on2 = 0; //switch 2 är av
                        } else
                        {
                                on2 = 1; //switch 2 är på
                        }
                }
                else if((val3 != 0 && on3 == 0) || (val3 == 0 && on3 != 0))
                {
                        switch1 = 0;
                        switch2 = 0;
                        switch3 = 1;
                        switch4 = 0;
                        if (val3 == 0)
                        {
                                on3 = 0; //switch 3 är av
                        } else
                        {
                                on3 = 1; //switch 3 är på
                        }
                }
                else if((val4 != 0 && on4 == 0) || (val4 == 0 && on4 != 0))
                {
                        switch1 = 0;
                        switch2 = 0;
                        switch3 = 0;
                        switch4 = 1;
                        if (val4 == 0)
                        {
                                on4 = 0; //switch 4 är av
                        } else
                        {
                                on4 = 1; //switch 4 är på
                        }
                }
                else
                {
                        switch1 = 0;
                        switch2 = 0;
                        switch3 = 0;
                        switch4 = 0;
                }
                TCNT1H = 0b11111000;
                TCNT1L = 0b01011111;        //måste börja om och räkna vid rätt tid.
}
```

```
ISR(INT2_vect)
{
		queueButton = 1;
}
```