LUXOR 3000

Väckarklockan för en lysande start på dagen!

1. Inledning

Denna rapport beskriver ett projekt som genomförts i kursen Digitala Projekt (EITF11). Projektet har ägt rum under andra läsperioden av VT17 och givits inom teknikprofilen System- och Programvaruutveckling på programmet Industriell Ekonomi vid Lunds Tekniska Högskola. Kursens mål är att ge studenter kunskap om hur ett konstruktionsarbete och utvecklingsprojekt går till genom att låta studenterna bygga och testa sin egna digitala prototyp.

Det projekt som beskrivs i denna rapport syftade till att utveckla LUXOR 3000, en väckarklocka för den lite extra morgontrötte studenten. Klockan kan, som de flesta väckarklockor, visa den aktuella tiden och ställas in för att ringa vid ett visst klockslag. Vad som skiljer LUXOR 3000 från andra väckarklockor är att det ringande alarmet stängs av genom att man tänder lampan i sitt sovrum, allt för att ytterligare hjälpa den morgontrötte studenten att ta sig upp om dagarna. Om man mot förmodan inte vaknat ordentligt när klockans alarm ringer gör man det förhoppningsvis efter att man tänt lampan. Det en gång så mörka och mysiga sovrum man spenderat natten i känns då inte fullt så gästvänligt längre, och den numera pigge studenten kan göra sig klar för dagen och ta sig till morgonföreläsningen.

2. Kravspecifikation

Nedan beskrivs de funktionella krav och kvalitetskrav som har använts som utgångspunkt för utvecklingen av prototypen.

2.1 Funktionella krav

2.1.1 Displayen ska visa den aktuella tiden (timmar, minuter)

2.1.2 På displayen ska framgå om larmet är aktivt eller inte.

2.1.3 Displayen ska uppdateras kontinuerligt för att hela tiden visa aktuell tid.

2.1.4 Det ska finnas knappar för användaren att ställa in tid och alarm.

2.1.5 Användaren ska kunna ställa ett alarm som motsvarar den tidpunkt vid vilken alarmet ger ifrån sig ljud.

2.1.7 Tiden visas i timmar (00-24), minuter (00-59) och sekunder (00-59)

2.1.8 En ljussensor styr avstängningen av alarmet.

2.2 Kvalitetskrav

2.2.1 Vid avstängning ska det dröja maximalt 1 sekund från att väckarklockan belyses tills dess att den stängs av.

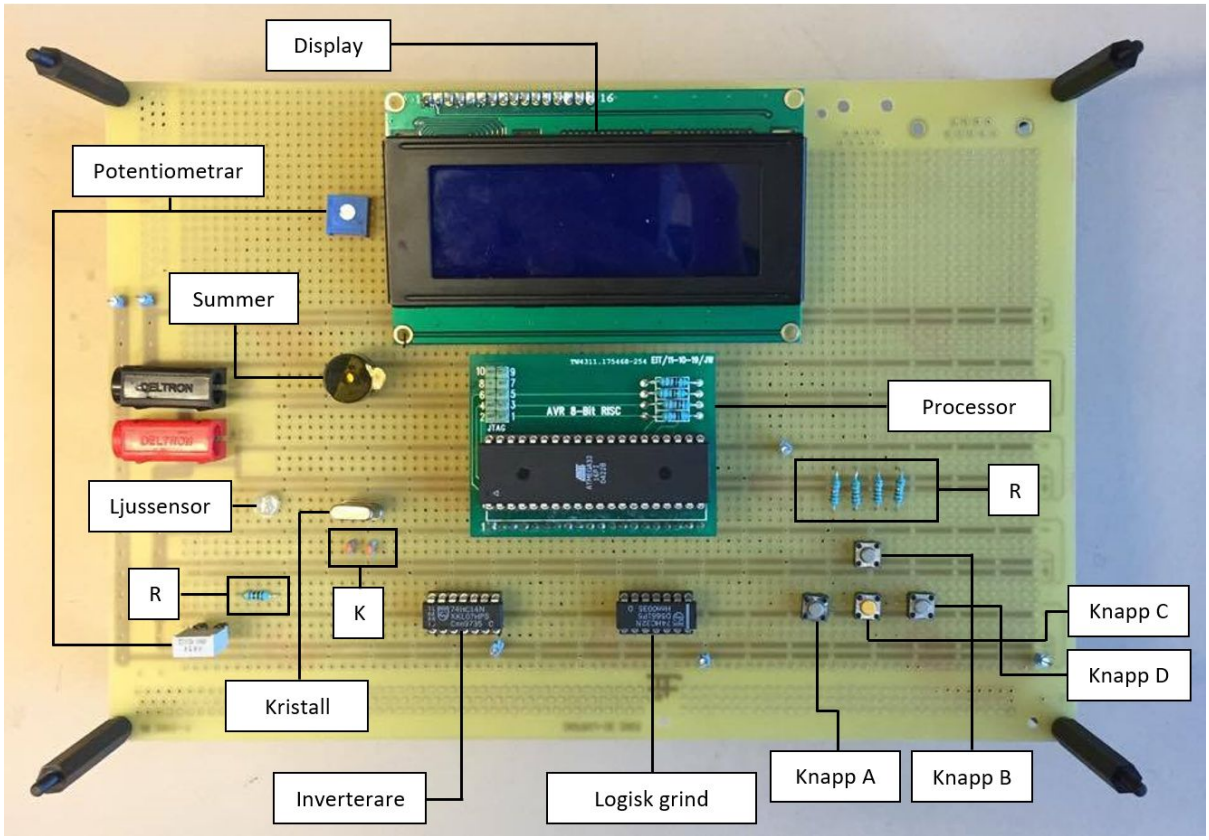
2.2.2 Klockans tideräkning ska ej avvika från den faktiska tiden med mer än en tiondels sekund per minut.

2.3 Begränsningar

- Användaren kan bara ställa in ett (1) alarm, och inte flera.
- Systemet är begränsat till det närmaste dygnet, dvs. användaren kan inte ställa in en alarmtid mer än 24 timmar framåt i tiden

3. Hårdvara

Nedan listas samtliga hårdvarukomponenter som använts vid genomförandet av projektet. Se kopplingsschema (avsnitt 6.7) för referens.



Figur 1. Schematisk bild som beskriver väckarklockans olika komponenter. Kondensatorer är utmärkta med "K", och resistorer med "R".

3.1 Processor

En processor av modellen ATmega32 används till den här prototypen. Processorn har fyra portar (A, B, C och D) med åtta pinnar var. Pinnarna kan användas som I/O-pinnar eller med för respektive pin speciella funktioner, exempelvis avbrott. Port A används för databussen till displayen. Pin noll till två på port B används till pins för RS, R/W och E på displayen, medan pin tre är kopplad till summern. Port C pin två till fem är kopplade till JTAG. Speciellt med port D är att pin två och tre kan användas för avbrott, vilket i detta projekt används både till alarm och inställningsfunktioner. Pin två (INT0) används för ljussensorfuntionen. Pin tre (INT1) kopplas till en logisk grind för att sköta avbrott relaterade till de fyra knappar som är kopplade till pin fyra till sju på port D. Processorns pin 12 och 13 är XTAL1 och XTAL2 vilka kopplas till en kristall. Utöver dessa används två pins till jord (GND) och en till VCC.

Prototypen använder 16-bitars timer för att beräkna tiden. Denna kallas TIMER1 och är inbyggd i processorn.

3.2 Display

För att visa både aktuell tid och inställd alarmtid på displayen har en Sharp Dot Matrix LCD med fyra rader använts. Displayen får information från processorn genom databuss in på pin DB0 till DB7. Pin RS styr inställning för instruktion eller data, pin R/W styr om displayen ska läsa eller skriva och pin E styr när exekvering sker. Pin 15 och 16 styr displayens bakgrundsbelysning. Genom att koppla VSS, VDD och V0 till en potentiometer kan displayens kontrast justeras.

3.3 Potentiometer

Potentiometer eller variabelt motstånd används för att justera motstånd inom ett intervall. I denna prototyp används två potentiometrar. En potentiometer kopplas till displayen för att justera kontrasten och den andra används för att variera känsligheten hos ljussensorn.

3.4 Knappar

Knapparna används för att ställa in tiden, sätta alarm och öka timmar respektive minuter. Knapparna fungerar på samma sätt som strömbrytare, vilket innebär att så fort en knapp är nedtryckt sluts kretsen och en binär etta skickas till en av processorns portar. För att koppla knapparna till ett avbrott i processorn kopplas de både till I/O-pinnar på processorn samt till en logisk grind.

- Knapp A: Ställa in tid.
- Knapp B: Öka timmar.
- Knapp C: Ställa in alarm.
- Knapp D: Öka minuter.

3.5 Kondensatorer

Kondensatorer är elektroniska komponenter som kan lagra elektrisk laddning. I projektet kopplas två kondensatorer mellan kristallen och jord.

3.6 Logisk grind

För att kunna använda avbrott kopplas en logisk grind mellan processorn (INT1) och de fyra knapparna. Den logiska grinden samlar knapparnas signaler till en gemensam signal. När någon av knapparna trycks ned genereras ett externt avbrott genom pin INT1, varpå processorn undersöker vilken knapp det är som är nedtryckt.

3.7 Resistorer

Resistorer eller motstånd innebär hinder för elektronernas rörelse. I väckarklockan ansluts en resistor på 100 k□ till kopplingen mellan knapp och processor. Resistorerna kopplas till jord och används för att processorn alltid ska få en digital signal (1:a eller 0:a) och undvika att bli utan signal. Ytterligare en resistor på 470 k□ återfinns mellan ljussensorn och dess potentiometer. Syftet med detta motstånd är att förhindra att ljussensorn går sönder om potentiometerns motstånd sätts för lågt.

3.8 Summer

En summer är en komponent som genererar ljud vid binär etta. Den kopplas direkt till processorn samt till jord.

3.9 Kristall

En kristall har använts som extern frekvenskälla vid beräkning av tiden. Processorn har också möjlighet att beräkna tiden utan att en kristall används, men kristallen är mer pålitlig som frekvenskälla och ger stabilare frekvensoutput även vid temperaturförändringar. Kristallens frekvens är 16 MHz.

3.10 Ljussensor

En ljussensor av typen AMS302T har använts för väckarklockans avstängningsfunktion. Ljussensorn är analog men kan göras digital med hjälp av en inverterare. Genom att variera motståndet i en potentiometer kalibrerades ljussensorn för att inte reagera på belysningen i rummet utan endast på direkt belysning.

3.11 Inverterare

Mellan lysdioden och processorn användes två inverterare i form av en Hex Schmitt-trigger inverterare. Den första inverteraren användes för att göra om ljussensorns analoga signal till digital signal. Den första inverteraren resulterade i att en binär etta skickades när det var mörkt, vilket skulle aktivera det externa avbrottet INT0 i processorn. För att det externa avbrottet skulle aktiveras när det är ljust istället användes ytterligare en inverterare.

4. Mjukvara

Mjukvaran utvecklades med programspråket C i programmet Atmel Studio 7.0.

4.1 Uppstart

Vid uppstart körs huvudprogrammet och funktionen start() exekveras som först sätter portarna till ut- eller insignaler. Det anpassar också de externa avbrottsrutinerna, beräkningen av tiden genom funktionen timerSetup() och förbereder displayen. Efter start funktionen går programmet in i main funtionen som loopar över if-satser som avgör vilken vy som ska visas på displayen. Main-loopen och avbrott från knapparna ger möjligheten att styra väckarklockan.

4.2 Timer

För tidtagning används en av processorns interna timerfunktioner i kombination med en extern 16 MHz kristall som frekvenskälla. Den interna timerfunktionen som använder 16 bitar för beräkning av tid (Timer1) konfigurerades sedan för att efter ett visst antal uppräkningar markera att en sekund förflutit. Det åstadkoms genom konfiguration av registrern TCCR1B, TCNT1, OCR1A, och TIMSK.

Med denna konfiguration kunde tideräkning kodas med hjälp av parvisa integers för sekunder, minuter och timmar.

4.3 Summer

Summern kontrolleras med två funktioner summerOn() och summerOff(). Dessa sätter pinnen som summern är kopplad till 1 (på) eller 0 (av).

4.4 Display

Vid uppstart ställs grundinställningarna för displayen in. Processor jobbar snabbare än displayen, vilket leder till att processorn måste vänta mellan kommandon som skickas till displayen. Detta görs genom att använda funktionen busy() som kontrollerar när displayen är redo för att ta emot ny information. Det flesta av funktionerna använder samma mönster för att skicka data till displayen. Därför användes commandToDisplay(char commandB, char commandA) för att effektivisera koden och minska risken för fel.

För att skriva ut text på displayen användes funktionen writeString() som delar upp bokstäverna var för sig och skriver ut dessa på skärmen.

4.5 Knappar

Då ett externt avbrott sker kan inga fler externa avbrott genereras samtidigt. Alltså kan man inte vara inne i time setting vyn (som kunde vara ett knappavbrott) och öka timmarna (ett annat avbortt). Därför användes en kombination ett antal globala variabler som antingen kunde vara 0/1 och nya knapptryck. Dessa kombinationer har koll på vilken vy som ska visas på displayen, när minutknappen och timknappen används och när alarmet ska stängas av eller sluta ringa.

Knapparna är kopplade till en logisk grind som i sin tur är kopplad till INT1. Så fort en knapp trycks ned genereras ett externt avbrott via INT1. Varje enskild knapp är också ansluten till egna pinnar på D-porten för att särskiljas.

4.6 Ljussensor

Ljussenorn styrs med funktionerna lightSensorOn() och lightSensorOff(). Dessa ställer in om sensorn ska ta emot signaler (av) eller skicka signaler till processorn (på). Ljussensorn är endast på då alarmet ringer.

Då ljussensorn belyses och alarmet ringer genereras ett avbrott via INT0 som stänger av alarmet.

4.7 Testning

För att testa och felsöka koden användes en JTAG. Testningen har genomförts i faser under projektets gång. Först testades displayen för att underlätta fortsatt testning och felsökning. Därefter testades timerfunktionen och klockan, följt av summern, ljussensorn och knapparna.

5.Utvecklingspotential

1. Gör det möjligt att öka och minska minuter och timmar istället för endast öka.
2. Gör det möjligt att ha datum så man kan ställa alarmet längre fram i tiden än 24h.

6. Instruktionsmanual

6.1 Uppstart

Koppla in en en strömkälla på 5V.

6.2 Ställ in tiden

1. Tyck på Knapp A.
2. Tyck på Knapp B för att öka antal timmar.
3. Tryck på Knapp C för att öka antalet minuter.
4. Tryck på Knapp A igen.

6.3 Ställ in alarm

1. Tyck på Knapp D.
2. Tyck på Knapp B för att öka antal timmar.
3. Tryck på Knapp C för att öka antalet minuter.
4. Tryck på Knapp D igen.
5. Nu är alarmet på och inställt till önskade tiden.

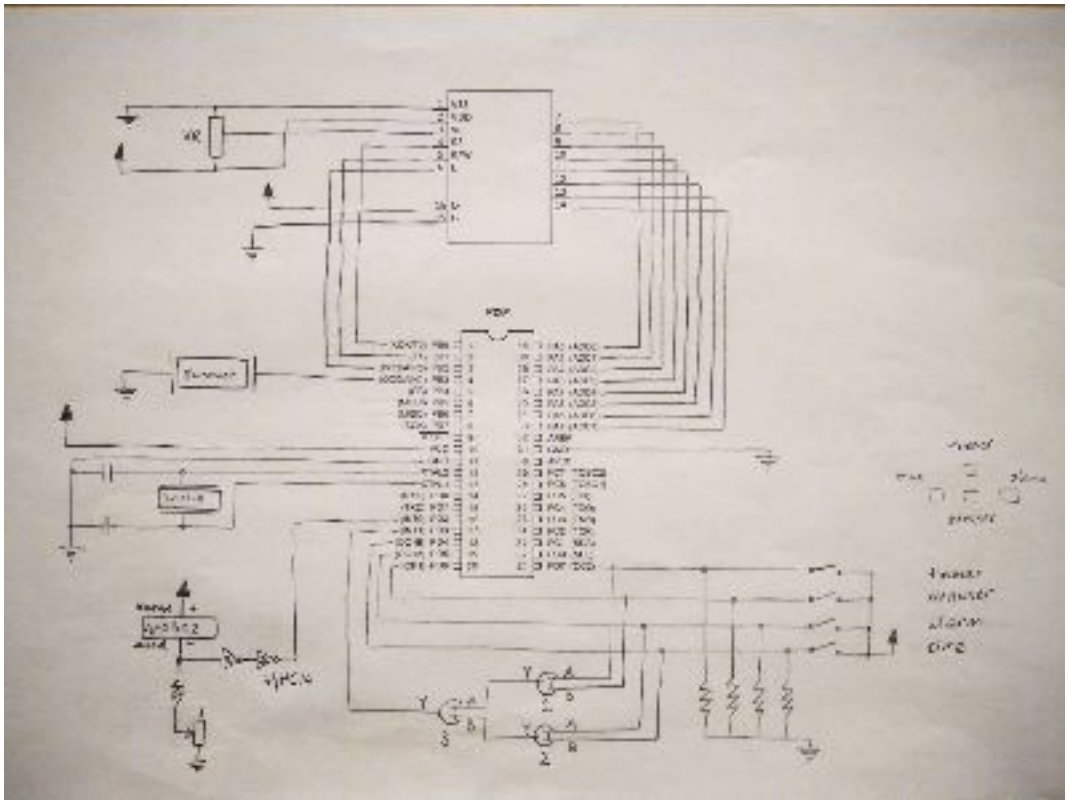
6.4 Avaktivera alarmet

1. Tryck på Knapp A
2. Nu är alarmet avaktiverat.

6.6 Stäng av summern

1. Lys med en stark lampa på ljussensorn, alternativt tryck på Knapp A.
2. Nu är summern avstängd och alarmet avaktiverat

6.7 Kopplingsschema



Figur 2. Kopplingsschema.

7. Källkod

```
#include <avr/io.h>
#include <avr/interrupt.h>
//importerar string och funktionerna som strlen
#include <string.h>
#include <util/delay.h>

int h1 = 0; // h1h2:m1m2:s1s2 den aktiva tiden för klockan
int h2 = 0;
int m1 = 0;
int m2 = 0;
int s1 = 0;
int s2 = 0;
int s2Old = 0;

int setH1 = 0; //tid för att sätta klockan
int setH2 = 0;
int setM1 = 0;
int setM2 = 0;

int ah1 = 0; // ah1ah2:am1am2:as1as2 tiden för alarmet
int ah2 = 0;
int am1 = 0;
int am2 = 0;

int alarmOnOff = 0; // har koll på om alarmet är på eller av. = 1 på, = 0 av.
int alarmRinging = 0; // 0 - alarmet ringer ingte, 1 - alarmet ringer
int settingTime = 0; // 0 - inget, 1 - sätter tid
int settingAlarm = 0;  // 0 - inget, 1 - sätter alarm
int settingButton = 0; //0 - loppar inte över checkButtons, 1- loppar över checkButtons (för att för settings vyerna att stanna.


/*Får ut de första 5 siffrorna är för PORTB (som inte påverkar displayen).
*Viktigt då vi inte vill att dessa ska ändras då vi ändrar på PB0-PB2 (de tre sista).
*Det kallas bit masking.
*/
char bitMaskPortB(){
	//sparar vad den nuvarande port B är.
	char currentPortB = PORTB;

	//Använder en AND operation för att få ut vilka av de första fem pin som är high och low.
	char bFirstFive = currentPortB & 0b11111000;

	//Returnerar abcde000
	return bFirstFive;
}

/*Väntar på att displayens busy flag är av. Alltså om BF = 0 (tredje från vänster)*/
void busy(){
	//Sätter alla pins på port A till input (alltså att de kommer att ta emot info till displayen)
	DDRA = 0b00000000;

	//Skapa en bit mask av port b
	char bFirstFive = bitMaskPortB();

	//Sätter port b att man vill läsa busy flag, sätter E till low
	PORTB = 0b00000010| bFirstFive;

	//Sätter på funktionen för att läsa data från RAM (DDRAM) och E till high
```

```
		PORTB = 0b00000110| bFirstFive;

		//Tar emot infromation från displayen
		char info = PINA;

		//Jämför busy flag om BF = 1 så kommer checkFlag = 0b10000000 annars 0b00000000.
		char checkFlag = info & 0b10000000;

		//Upprepar de tidigare stegen i en lopp tills BF = 0
		while(checkFlag == 0b10000000){
			PORTB = 0b00000010| bFirstFive;
			PORTB = 0b00000110| bFirstFive;
			info = PINA;
			checkFlag = info & 0b10000000;
		}
		//Sätter alla pins i port A till output.
		DDRA = 0b11111111;
}

/*Metod för att skriva ett kommando till displayen*/
void commandToDisplay(char commandB, char commandA){
		//Väntar tills BF = 0
		busy();

		//Sätter E i high (PB2 = 1) + RS & R/W
		char bFirstFive = bitMaskPortB();

		PORTB = commandB | bFirstFive;

		//Sätter DB7-DB0
		PORTA = commandA;

		//Sätter E i low
		PORTB = 0b00000000 | bFirstFive;

		//Sätter port A till 0 (ej viktigt)
		PORTA = 0b00000000;

}

/*Sätt function set*/
void functionSet(){
		//0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
		//0b00111000: Sätter function set till 8-bit, 2 rader, 5*8, de sista två siffrorna spelar ingen roll om 1 eller 0.
		commandToDisplay(0b00000100, 0b00111000);
}

/*Sätt på displayen*/
void displayOn(){
		//0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
		//0b00001100: Sätt på displayen
		commandToDisplay(0b00000100, 0b00001100);
}

/*Metod för att clear display.*/
void clearDisplay(){
		//0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
		//0b00000001: Clear display
		commandToDisplay(0b00000100, 0b00000001);
}

/*Metoden för att skriva en char.*/
```

```
void writeChar(char character){
	//0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 1,)
	commandToDisplay(0b00000101, character);
}

/*Skriver ut siffror då writeChar metoden inte kan göra det med direkta siffor.*/
void writeNumber(int nbr){
	switch (nbr)
	{
		case 1:
		writeChar(0x31);
		break;

		case 2:
		writeChar(0x32);
		break;

		case 3:
		writeChar(0x33);
		break;

		case 4:
		writeChar(0x34);
		break;

		case 5:
		writeChar(0x35);
		break;

		case 6:
		writeChar(0x36);
		break;

		case 7:
		writeChar(0x37);
		break;

		case 8:
		writeChar(0x38);
		break;

		case 9:
		writeChar(0x39);
		break;

		case 0:
		writeChar(0x30);
		break;
	}


}

/*Metoden för att skriva en string.*/
void writeString(char string[]){
	//loop över string (i c är det en vektor)
	int length = strlen(string);
	for(int c = 0; c < length; c++){
		//Skriver ut på displayen varje tecken från string
		writeChar(string[c]);
	}
}
```

```
/*Sätt på markören*/
void cursorOn(){
	//0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
	//0b00001100: Sätt på markören
	commandToDisplay(0b00000100, 0b00001111);
}

/*Stäng av markören */
void cursorOff(){
	//0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
	//0b00001100: Stänger av markören
	commandToDisplay(0b00000100, 0b00001100);
}

/*Flytta markören till bestämd plats.*/
void moveCursor(char location){
	if(location<0b01111111){
		//0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
		//avnänder set DDRAM adress för att flytta markören till önskad plats
		commandToDisplay(0b00000100, 0b10000000|location);
	}
}

/*Sätter flyttar markören till första rutan på diplayen*/
void cursorHome(){
	//0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
	//0b00001100: Sätt på displayen
	commandToDisplay(0b00000100, 0b00000010);
}

//Vyn för klockan
void clockView(){
	clearDisplay();
	moveCursor(0x40);
	writeString("Time: ");
	writeNumber(h1);
	writeNumber(h2);
	writeString(":");
	writeNumber(m1);
	writeNumber(m2);
	writeString(":");
	writeNumber(s1);
	writeNumber(s2);

	moveCursor(0x14);
	writeString("Alarm:");

	if(alarmOnOff == 0){
		moveCursor(0x1B);
		writeString("OFF");


	} else{
		writeNumber(ah1);
		writeNumber(ah2);
		writeString(":");
		writeNumber(am1);
		writeNumber(am2);
```

```
		}
		cursorOff();
}

//Vyn för när man ställer in klockan
void timeSettingView(){
		int setH1 = h1;
		int setH2 = h2;
		int setM1 = m1;
		int setM2 = m2;
		clearDisplay();

		moveCursor(0x04);
		writeString("Set the Time");
		moveCursor(0x40);
		writeString("Time:");
		writeNumber(setH1);
		writeNumber(setH2);
		writeString(":");
		writeNumber(setM1);
		writeNumber(setM2);

		cursorOff();
}

//Vyn för när man sätter alarmet
void alarmSettingView(){
		clearDisplay();
		moveCursor(0x03);
		writeString("Set the Alarm");

		moveCursor(0x40);
		writeString("Alarm:");
		writeNumber(ah1);
		writeNumber(ah2);
		writeString(":");
		writeNumber(am1);
		writeNumber(am2);

		cursorOff();
}

//Vyn när almarmen går av. Vi borde även displaya tiden här.
void alarmGoesOffView(){
		clearDisplay();
		moveCursor(0x47);
		writeString("Ring!");
		cursorOff();

}

//Metod för att kolla om alarmet ska gå av
void alarmGoesOff(){
		if(h1==ah1 & h2==ah2 & m1==am1 & m2==am2 & alarmOnOff == 1){
				alarmGoesOffView();
				//summerOn();
				alarmRinging = 1;
				lightSensorOn();

		}

}
```

```
//Sätter på summer så den börjar låta
void summerOn(){
	//Behåller display inställningen samma (3 sista siffrorna)
	PORTB |=  0b00001000;

}

//Stänger av summern så den slutar låta
void summerOff(){
	//Behåller display inställningen samma (3 sista siffrorna)
	PORTB &=  0b00000111;
}

//Stänger av ljussensors så den inte reagerar på ljus
void lightSenorOff(){
	DDRD |= 0b00000100;

}

//Sätter på ljussensorn så den reagerar på ljus
void lightSenorOn(){
	DDRD |= 0b00000000;

}

//Läser av vilken knappt som har blivit tryckt
void checkButtons() {
	while(settingButton == 1){
		_delay_ms(150);
		char btn1 = (1 << PD4); //set time, vänster.
		char btn2 = (1 << PD5); //set alarm, höger
		char btn3 = (1 << PD6); //ner, sätt minuter
		char btn4 = (1 << PD7); // upp, sätt timmar samma som 0b10000000
		btn1 &= PIND; // kollar om set time knappen är 1 eller 0
		btn2 &= PIND;
		btn3 &= PIND;
		btn4 &= PIND;

		if(btn1 == 0b00010000) { //om set time är 1
			if(settingTime == 0){ //vill sätta tiden
				settingTime = 1;
				settingButton = 1;
				timeSettingView();

			} else if (settingTime == 1){ //går från att sätta tiden till vanlig
				settingTime = 0;
				settingButton = 0;

			}
		}

		if(btn2 == 0b0010000){ //om set alarm är 1
				if(alarmOnOff == 0) { //om man vill sätta alarmet

					settingAlarm = 1;
					alarmOnOff = 1;
					settingButton = 1;
					alarmSettingView();

				} else if (alarmOnOff == 1) { //alarmet är på
```

```
                                                    if(alarmRinging == 1){          // alarmet ringer och man vill
        stänga av det
                                                            alarmOnOff = 0;
                                                            summerOff();
                                                            alarmRinging = 0;
                                                            lightSenorOff();

                                                    } else if(settingAlarm == 1){ //man vill gå ur setting alarm vyn
                                                            settingAlarm == 0;

                                                    } else { //man vill stäng av alarmet (som inte ringer), settingAlarm
        = 0
                                                            alarmOnOff = 0;
                                                    }
                                                    settingButton = 0;
                                            }
                                    }


                                    if(btn3 == 0b01000000  && settingTime== 1){ //sätt minuter tid
                                                    setM2++;
                                            if (setM2 > 9){
                                                    setM1++;
                                                    setM2 = 0;
                                            }
                                    }

                                    if(btn4 == 0b10000000 && settingTime== 1){ //sätter timmar tid

                                                            setH2++;
                                                            if (setH2 > 9){
                                                                    setH1++;
                                                                    setH2 = 0;
                                                            }
                                    }

                                    if(btn3 == 0b01000000 && settingAlarm == 1) { //alarm minuter
                                                    am2++;
                                            if (am2 > 9){
                                                    am1++;
                                                    am2 = 0;
                                            }
                                    }
                                    if(btn4 == 0b10000000 && settingAlarm == 1){ //alarm timmar
                                                    ah2++;
                                            if (ah2 > 9){
                                                    ah1++;
                                                    ah2 = 0;
                                            }
                                    }
                                    _delay_ms(150);
                                    }
}



/*Fixar med alla funktioner.*/
void start(){

        //Sätt vilka håll pins ska vara (DDR*)
        DDRA = 0xFF; // display
```

```
	DDRB = 0xFF; // E, R/w, RS (display) + summer
	DDRC = 0x00; //används inte av oss
	DDRD = 0x00; // knappar + ljussensor

	//Avbrottsrutiner
	GICR |= 1<<INT1 | 1<<INT0;
	MCUCR |= 1<<ISC11 | 1<<ISC10 | 1<<ISC01 | 1<<ISC00;


	//Fixar med timern
	timerSetup();

	//Sätter alarm till av
	alarmOnOff = 0;

	//Förbereder displayen
	functionSet();
	clearDisplay();
	displayOn();
	cursorOn();
	cursorHome();

}

void timerSetup()
{
	TCCR1B |= (1 << WGM12 | 1 << CS10 | 1 << CS12);  //Timer/Counter Control Register 1B. Sets up timer with
prescaler = 1024 and CTC enabled. 8 bits.
	TCNT1 = 0x00; // Initialize counter
	OCR1A = 15625; // Output Compare Register 1A. Set the CTC compare value.  16 bits
	TIMSK |= ((1 << OCIE1A));  // Timer Interrupt Mask Register, Output Compare a match Interrupt 1A. Enables the
CTC interrupt. Vi använder CTC, så processorn ska ta emot signal när räknaren matchar.
	sei();  // enable Global interrupts.
}

ISR(TIMER1_COMPA_vect) { //Funktion som körs varje gång ett avbrott genereras   //External events trigger an interrupt — the
normal control flow is interrupted and an Interrupt Service Routine (ISR) is called.
	//TIFR = 0x01; //Interrupt request har genererats. (Behövs detta?)
	if (s2<9) {
		s2Old = s2;
		s2++;
	}
	else {
		s2=0;
		if (s1<5) {
			s1++;
		}
		else {
			s1=0;
			if (m2<9) {
				m2++;
			}
			else {
				m2=0;
				if (m1<5) {
					m1++;
				}
				else {
					m1=0;
					if (h1<2) {
						if(h2<9) {
							h2++;
```

```
                                                }
                                                else {
                                                        h2=0;
                                                        h1++;
                                                }
                                        }
                                        else {
                                                if (h2<3) {
                                                        h2++;
                                                        }else {
                                                        h1=0;
                                                        h2=0;
                                                        m1=0;
                                                        m2=0;
                                                        s1=0;
                                                        s2=0;
                                                }
                                        }
                                }
                        }
                }
        }
}

ISR(INT1_vect) {
        GIFR = 1<<INTF1;
        _delay_ms(100);
        checkButtons(); //kallar på knappmetoden
        _delay_ms(100);
}

ISR(INT0_vect) {
        GIFR = 1<<INTF0; //aktiveras när man lyser på ljussensorn
        _delay_ms(100);
                summerOff();
                alarmRinging = 0;
                lightSenorOff();
        _delay_ms(100);
}


int main(void)
{
        start();
        clockView();

        while (1)
        {
                //_delay_ms(10);


                if(s2 != s2Old & alarmRinging == 0){

                        clockView();
                        alarmGoesOff();
                }


        }
}
```