ABSTRACT

The following report compiles the process of constructing a prototype called The Visual Eggtimer. The Visual Eggtimer is an eggtimer which is supposed to facilitate the boiling of a perfect egg, according to the user's preferences (soft, medium or hard boiled). The timer starts when the water reaches a temperature of 100°C. Continuously, during the boiling process, the user is provided with updates about the coagulation of the egg on a LCD-display and by three LEDs. In accordance with different time settings, for various degrees of coagulation, messages and instructions are shown on the display. The LEDs are lightened gradually, to visualise the process even more. First a specification of requirements was formulated. From this a strategy for the integration between soft- and hardware was made. When the construction of the soft- and hardware was completed, tests and modifications were made on the prototype till the desired result was received.

THE VISUAL EGGTIMER

Skola: Lunds Tekniska Högskola
Institution: Elektro- och informationsteknik
Kurs: Digitala projekt (EITF11)

Projekttyp: Rapportsammanfattning
Handledare: Bertil Lindvall
Studenter: Amanda Österling, Emmy Malmqvist, Hanna Krokström
Datum: 25 maj 2016

1. Inledning

Denna rapport är en del i ett projekt i kursen Digitala projekt, EITF11, vid Lunds Tekniska Högskola. Prototypen som skapats är en visuell äggklocka, The Visual Eggtimer, med målet att underlätta kokning av ägg med olika koaguleringsgrad. Äggklockan är avgränsad till att endast ge visuella signaler.

Syftet med projektet är att ge ökad förståelse för hur konstruktionsarbete fungerar, genom utformning av en enklare digital prototyp. Framtagandet av prototypen har innefattat planering av prototypen, i form av en kravspecifikation, skapande av hårdvaran respektive mjukvaran samt en integrering av dessa.

Rapporten inkluderar kravspecifikation, teori om använd hård- och mjukvara, tillvägagångssätt samt resultat med tillhörande diskussion och slutsats. Rapporten avslutas med referenslista och appendix med källkod och kopplingsschema.

2. Kravspecifikation

The Visual Eggtimer är en äggklocka med avsikt att underlätta äggkokning genom visuella hjälpmedel. Produktidén bygger på att en timer startas då vattentemperaturen i en kastrull når sin kokpunkt 100 °C. Användaren skall sedan löpande förses med konsultativa anvisningar samt meddelanden och visuella signaler om hur långt ägget kommit i sin koagulering.

The Visual Eggtimer ska ha en reset-knapp, som startar om processen, det vill säga både tidtagning, meddelanden och signaler. Äggklockan ska även ha en timer-knapp, som möjliggör åskådning av aktuell koktid. Prototypen ska även bestå av en LCD-display, tre LED (grön, gul, röd), en timer samt en temperatursensor.

Äggklockans cykel kan beskrivas i följande faser:

> *Uppstartsfas*. Då vattnet inte ännu nått sin kokpunkt. På displayen visas texten: "TEMP HAS NOT REACHED 100 DEG".
>
> *Fas 0.* Från att vattnet nått sin kokpunkt till att ägget anses vara löskokt. På displayen visas texten: "TEMP HAS REACHED 100 DEG".
>
> *Fas 1.* Då vattnet kokat i 4 minuter anses ägget vara löskokt. Grön LED tänds och på displayen visas texten: "YOUR EGG IS SOFT-BOILED".
>
> *Fas 2.* Då vattnet kokat i 6 minuter anses ägget vara mellankokt. Även gul LED tänds och på displayen visas texten: "YOUR EGG IS MED-BOILED".
>
> *Fas 3.* Då vattnet kokat i 10 minuter anses ägget vara hårdkokt. Även röd LED tänds och på displayen visas texten: "YOUR EGG IS HARD-BOILED".

Följande konsultativa anvisningar används för att underlätta äggkokningen:

- *Användaren startar the Visual Eggtimer.* Följande meddelande visas på displayen: "WELCOME".

- *Temperaturen sjunker under kokpunkten efter att rätt temperatur har nåtts.* Följande meddelande visas på displayen: "INCREASE THE TEMP".

- *Timer-knappen trycks in.* Följande meddelande visas på displayen: "TIME : *min:sec*"

Således skall följande gälla för den färdiga prototypen:

- En temperatursensor skall mäta aktuell temperatur.
- En tidtagning skall initieras när temperaturen nått 100 °C.
- Tre stycken LED skall tändas vid förutbestämda tidpunkter.
- En LCD-display skall visa tidtagarurets aktuella tid samt visa förutbestämda textmeddelanden, under bestämda tidsintervall och vid bestämda händelser.
- En timer-knapp skall leda till att aktuell koktid visas på displayen under 6 sekunder och därefter återgår displayen till tidigare textmeddelande.
- En reset-knappen skall starta om processen.

3. Teori

3.1 Hårdvara

Detta avsnitt beskriver vilka komponenter som använts i projektets hårdvara.

3.1.1 Kopplingsschema

Ett kopplingsschema är en schematisk översikt som visar hur hårdvarukomponenterna är sammankopplade. Ett kopplingsschema för The Visual Eggtimer, skapat i PowerLogic, återfinns i *Appendix 1 Kopplingsschema*. Kopplingsschemat togs fram som ett inledande steg i projektet och prototypen skapades utifrån detta. Vissa problem uppstod i samband med testning av hårdvaran, varpå korrigeringar gjordes i hårdvaran och därefter uppdaterades kopplingsschemat.

3.1.2 Processor

Processorn som används är ATmega 16 AVR 8-bit Microcontroller. Denna har totalt 40 pinnar uppdelade i A-, B-, C- samt D-portar. 32 pinnar är I/O-pinnar och 8 pinnar på port C är reserverade för ett JTAG-interface. Med hjälp av interfacet JTAG kan processorn

programmeras i programspråket C i en dator. Port A är ingång till A/D-konverteraren. Komponenternas koppling återfinns i *Appendix 1 Kopplingsschema*.

En timer skapas genom att i programkoden deklarera en variabel för tidsräkning och utnyttja processorns funktion för tidsavbrott. Processorns 16-bitars timer/counter används. Efter programmerade inställningar sker ett avbrott drygt varje halv sekund, varpå variabeln för tidsräkning adderas med denna tid (0,524288 sekund). Vid varje avbrott sker även temperaturmätning.

Som matningsspänning används 5 V och vald frekvens är 8 MHz.

3.1.3. LCD-display

Den LCD-display som används är SHARP Dot-Matrix, vilken är en LCD Units Alfanumerisk teckendisplay. Displayen valdes på grund av att den kan hantera både siffror och bokstäver. Varje tecken som skrivs ut på displayen sänds som 8 bitar från processorn och programkoder programmeras för att de bestämda meddelandena ska visas.

LCD-displayen används för att kunna informera användaren om kaoguleringsgrad, kokningstid samt rådgivande anvisningar.

3.1.4 LED

Tre olika LEDs (grön, gul och röd) används för att signalera hur långt ägget kommit i sin koagulering, enligt faserna beskrivna i *2. Kravspecifikation*. Varje enskild LED kan maximalt klara en ström på 10 mA, varför en resistor kopplas till respektive LED, för att begränsa strömmen.

3.1.5 Temperatursensor

Den temperatursensor som används är LM335, Kelvin temperature sensor. Temperatursensorn kopplas till processorns A-port och utnyttjar A/D-konverteraren. Som spänningsreferens används processorns pinne AREF, vilken är kopplad till 5 V.

Temperatursensorn ger en råspänning som är proportionell mot temperaturen i Kelvin, där en grad motsvarar 10 mV. Den aktuella temperaturen kan därmed, med hjälp av programkoder, beräknas av processorn och därefter omvandlas till Celsius.

3.1.6 Knappar

Två knappar, en reset-knapp och en timer-knapp, används till äggklockan med uppgift att fungera enligt beskrivning i *2. Kravspecifikation*. Knapparna fungerar likt strömbrytare där en spänning skickas till kopplad port då nedtryckning sker.

Reset-knappen är kopplad till port A4 och timer-knappen är kopplad till port A5. De är dessutom gemensamt kopplade till port D2 (INT0) via en grind, vilket innebär att ett avbrott uppstår då någon utav de två knapparna trycks in. Vid ett avbrott undersöker processorn

vilken av knapparna som tryckts in och åtar åtgärder därefter. Vad som sker då knapparna trycks in bestäms av processorns programmering.

3.1.7 Resistorer

Resistorer används för att begränsa och kontrollera strömmen till några av de använda hårdvarukomponenterna. Följande resistorer används:

- Resistorer på 500 ohm till respektive LED.
- Resistor på 6000 ohm till temperatursensorn.
- Resistorer på 10 000 ohm till respektive knapp.

3.1.8 Trimmer

För att kunna reglera kontrasten på displayen används en *Bourns Trimpot 3386* trimmer. Trimmern kopplas till displayens tredje ben. Genom att manuellt vrida på den vita skivan ökas och minskas displayens kontrast.

3.1.9 Grind

Vid sammankopplingen av de två knapparna nyttjas en OR-grind. Grindens syfte är att generera samma avbrott oavsett om det är reset-knappen eller timer-knappen som trycks ner.

3.2 Mjukvara

Prototypens mjukvara är konstruerad i AtmelStudio och skriven i programmeringsspråket C, fullständig källkod återfinns i *Appendix 2. Källkod*. Huvudmetoden inleds med en uppstartsfas där nödvändiga startinställningar sker. Detta innebär att definiera vilka av processorns portar som ska vara in- respektive utsignal, möjliggöra avbrott, ge deklarerade variabler ett startvärde, släcka lampor samt ställa in grundinställningar för LCD-display, temperatursensor respektive timer.

Därefter går huvudmetoden in i en evig while-loop som avbryts med en viss frekvens, drygt 0,5 sekund. Vid avbrottet avbryts huvudloopen och en metod som uppdaterar temperaturen anropas. Första gången temperaturen når över 100 °C initieras tidtagningen. Därefter uppdateras även tidtagningen för varje avbrott.

While-loopen fortsätter därefter med en if-sats som utförs då temperaturen är under 100 °C, och inte heller tidigare nått 100 °C. Då temperaturen når 100 °C eller över utförs istället efterföljande if-sats. Beroende på aktuell tid utförs däri olika if-satser som anropar metoder för att visa äggets aktuella koaguleringsgrad. Detta innebär att meddelande visas på LCD-displayen och LEDs tänds.

Om aktuell temperatur sjunker under 100 °C anropas metoder för konsultativa anvisningar. För att textmeddelanden, för respektive fas, ska visas konstant under utsatt tid används

variabler som markörer, så kallade flaggor, för att se till att dessa metoder inte anropas på nytt varje loop.

Vid knapptryckning sker ett avbrott, där huvudmetoden avbryts och metoder som kontrollerar vilken knapp som tryckts in anropas. While-loopen avslutas med två if-satser där huvudmetoden kontrollerar ifall variablerna som representerar någon av knapparna är lika med 1. Om så är fallet, anropas metoder för att vidta rätt åtgärder och därefter nollställs variabelns värde.

4. Genomförande

4.1 Planering

Arbetet inleddes med idéspåning som mynnade ut i en lista med önskvärda funktioner för produkten, vilka sedan sammanställdes i en kravspecifikation. Efter informationssökning kring de olika hårdvarukomponenternas funktioner skapades ett kopplingsschema i Power Logic och utifrån detta påbörjades hårdvarukonstruktionen.

4.2 Hårdvarukonstruktion

Gruppen tilldelades komponenter, enligt *Appendix 1 Kopplingsschema*, och verktyg för att kunna påbörja konstruktionen. Sammankopplingen av hårdvarukomponenterna gjordes enligt kopplingsschemat och med vägledning från handledaren. Därefter testades och felsöktes hårdvaran med hjälp av multimeter, logik prob samt enklare programmeringsrader med hjälp av debugger-funktionen i AtmelStudio. Nödvändiga konstruktionsändringar gjordes tills önskvärt beteende påvisades.

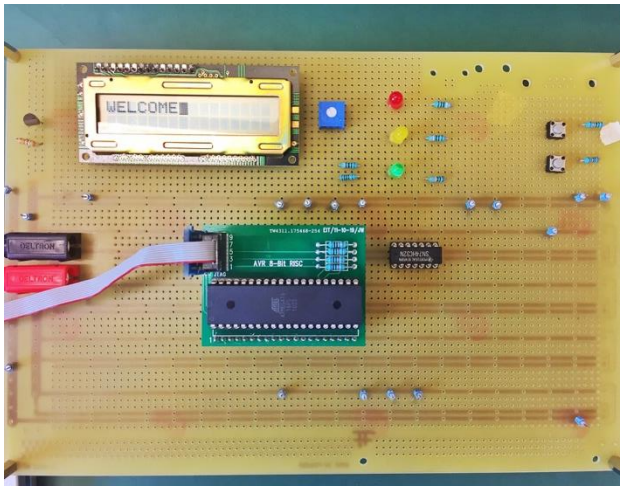
4.3 Mjukvarukonstruktion

Mjukvaran har programmerats med programspråket C i AtmelStudio. Testning av mjukvaran har genomförts löpande, genom koppling av hårdvaran till mjukvaran med hjälp av en JTAG. Modifieringar gjordes successivt för att nå önskvärt resultat.

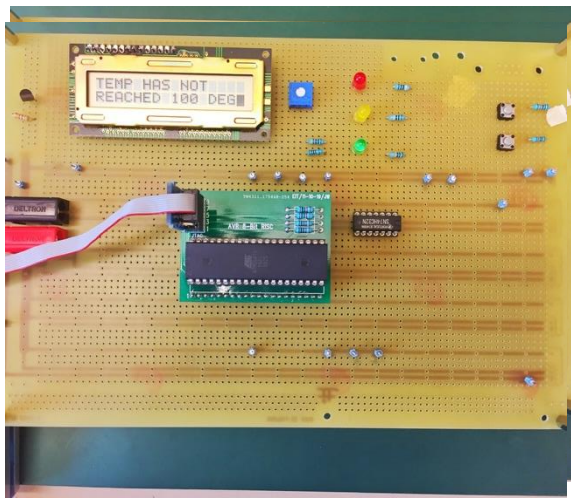
5. Resultat

Projektet resulterade i en välfungerande prototyp med funktioner i enlighet med kravspecifikationen. Observera att vissa modifikationer, avseende temperatur och tid, har gjorts i källkoden för att enkelt kunna utföra tester av prototypen. Se *Appendix 2 Källkod.*

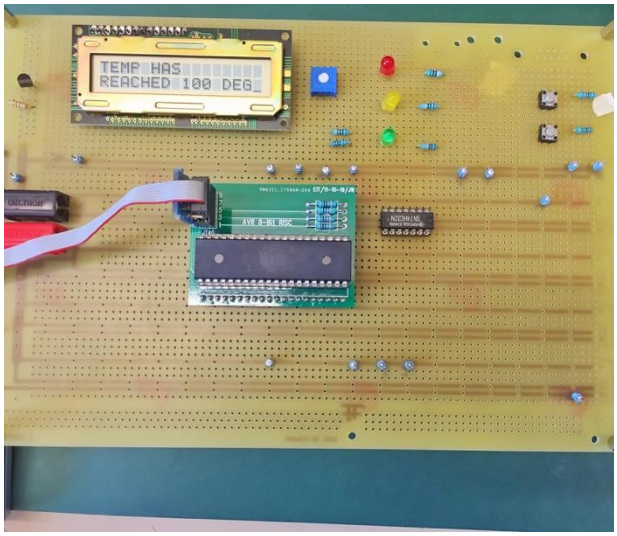
Nedan följer en presentation av prototypen, enligt beskrivning i *2. Kravspecifikation*.



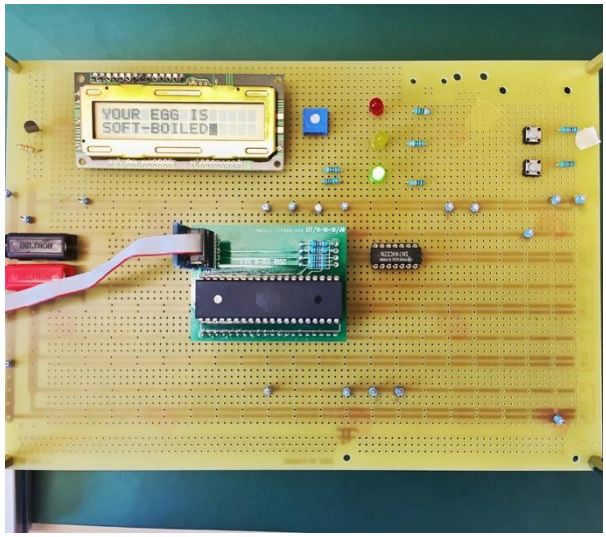
Figur 1. När användaren startar The Visual Eggtimer skrivs meddelandet "WELCOME". Ingen LED lyser.



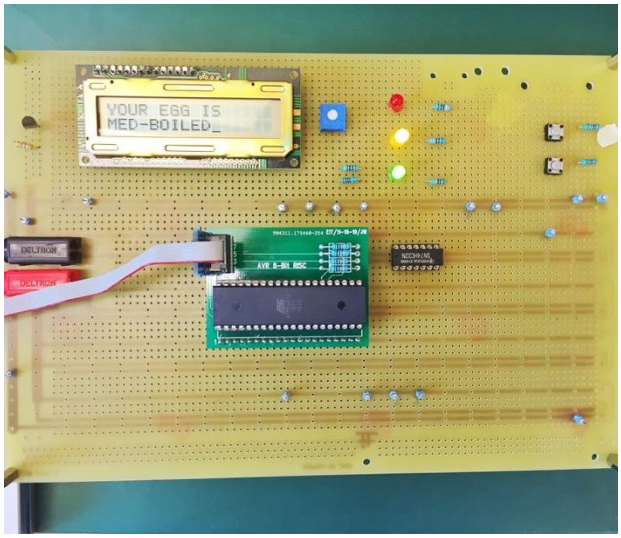
Figur 2 Uppstartsfasen. På displayen visas meddelandet TEMP HAS NOT REACHED 100 DEG. Ingen LED lyser.



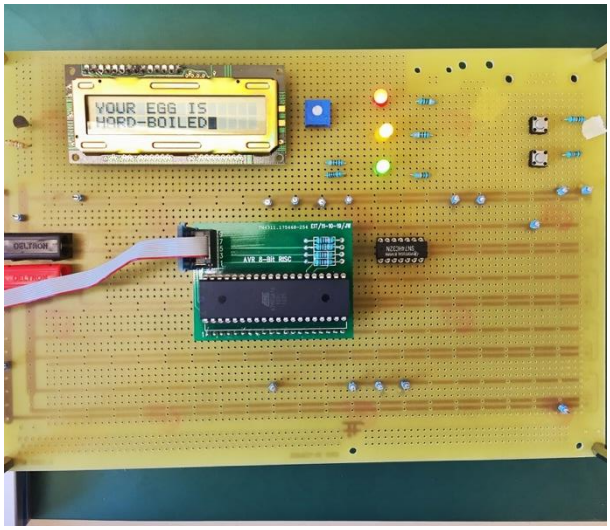
Figur 3. Fas 0. På displayen visas meddelandet "TEMP HAS REACHED 100 DEG". Ingen LED lyser.



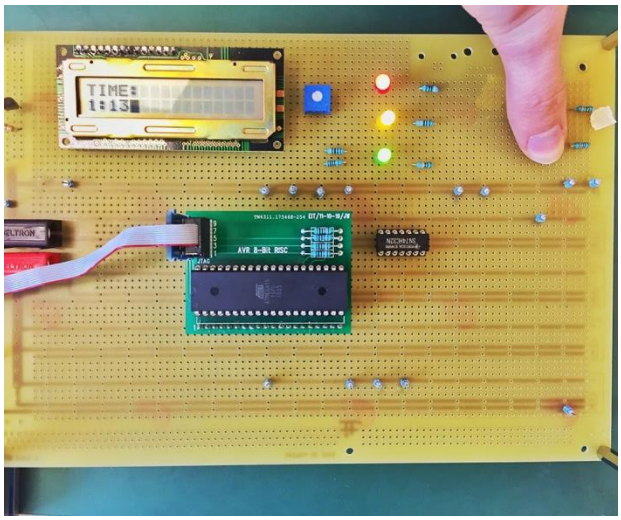
Figur 4 Fas 1. På displayen visas meddelandet "YOUR EGG IS SOFT-BOILED". Grön LED lyser.



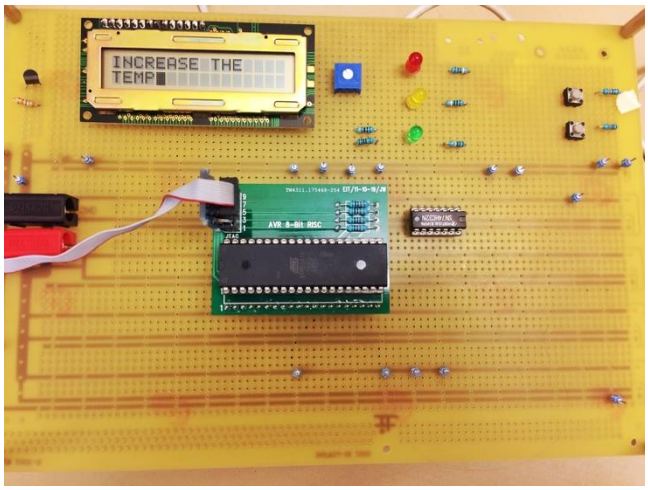
Figur 5 Fas 2. På displayen visas meddelandet "YOUR EGG IS MED-BOILED". Grön och gul LED lyser.



Figur 6 Fas 3. På displayen visas meddelandet "YOUR EGG IS HARD-BOILED". Grön, gul och röd LED lyser.



Figur 7 När timer-knappen tryckts in visas den aktuella tiden på displayen.



Figur 8. När temperaturen sjunker under kokpunkten visas meddelandet " INCREASE THE TEMP" på displayen.

6. Diskussion och slutsats

Framtagningen av äggklockan har varit både lärorik, spännande och uppmuntrat till kreativitet och skapande. Inför projektet saknade gruppen grundläggande kunskaper kring konstruktionsarbete samt programmering i C, varpå processens uppstart tog lång tid. Fokus lades därför inledningsvis på att skapa en grundförståelse för arbetsområdet. Under arbetets gång har kontinuerlig feedback getts från vår handledare och vid projektets utgång har ökad förståelse för både hård- och mjukvarukonstruktion inom ämnet fåtts. Olika delmoment har upplevts olika svåra. Den största utmaningen ansåg vi vara temperatursensorn, då det var flera delmoment som skulle stämma överens för att uppnå önskvärd funktionalitet.

Under arbetets gång har insikters fåtts, varpå det första utkastet av kravspecifikationen modifierats. Bland annat har på- och av-knappar ersatts av reset- och timer-knappar, detta på grund av det faktum att aggklockan automatiskt startas då strömmen slås på. Istället för den påtänkta ljudsignalen har fler konslutativa anvisningar skapats för att öka den visuella upplevelsen, därav namnbytet ”The Ultimate Eggtimer” till ”The Visual Eggtimer”

Arbetsbelastningen har legat på en rimlig nivå under läsperiodens gång, då konstruktionsarbetet inleddes i god tid. Vi är nöjda med vår slutprodukt och känner att projektet i sin helhet har varit givande och utvecklande. Ytterligare utvecklingspotential finns för produkten genom att lägga till funktioner som ljud och vibration.

7. Referenslista

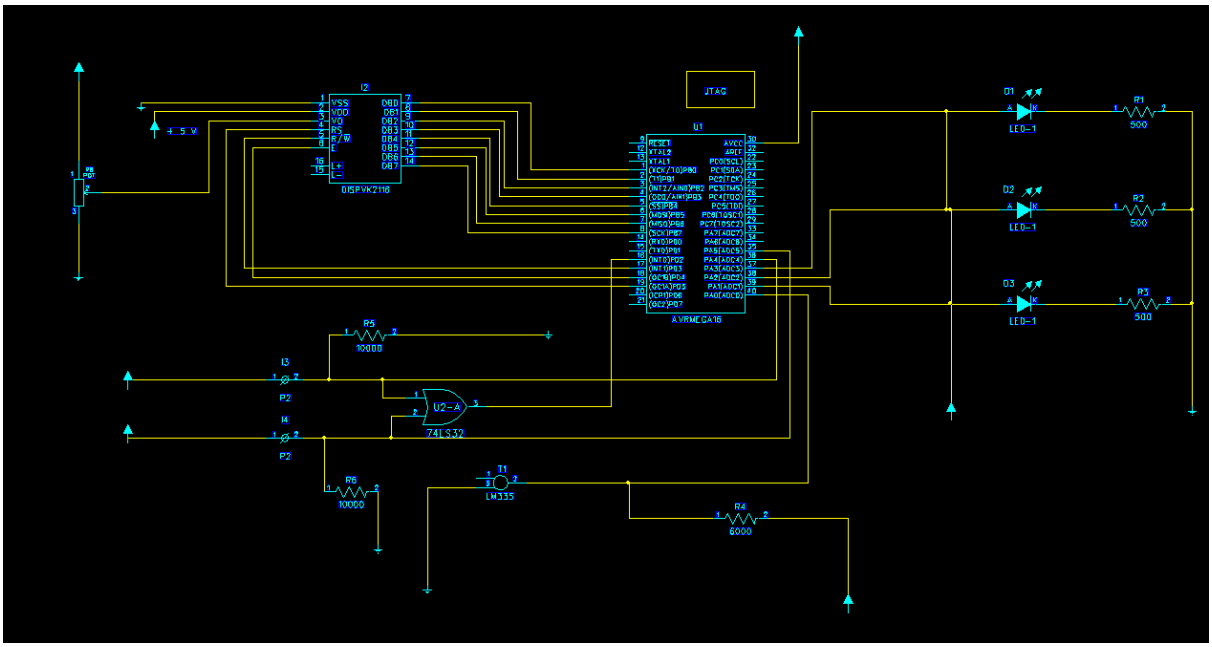
Databladet AVR- ATmega16 High Performance AVR 8-bit Microcontroller (Complete)
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf
Datum: 2016-05-17

Databladet LM335- Kelvin temperature sensor
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Sensors/lm335.pdf
Datum: 2016-05-17

Databladet Sharp Dot-Matrix - LCD Units Alfanumerisk teckendisplay
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf
Datum: 2016-05-17

Appendix

1. Kopplingsschema



2. Källkod

```
/*
 * Eggtimer_v1.c
 *
 * Created: 2016-04-21 12:00:51
 *  Author: digpi05
 */

#include <avr/io.h> //kan skriva betäckningar som i databladet
#include <util/delay.h>
#include <avr/interrupt.h>


#define F_CPU 8000000UL   // 8 MHz konstant som definierar frekvensen

//------------------------------VARIABLES--------------------------------------

int buttonReset = 0;
int buttonTime = 0;
int celsius;
double rawVoltage;
double millivolts;
double counter; //tidräkning
int min;
int sec;
int sec1;
int sec2;

//flaggor
int first0;
int first1;
int first2;
int first3;
int first4;
int reached;

//------------------------------METHODS----------------------------------------

void setUp();
void setUpLCD();
void setPin(char port, char pin, char state);
int checkButton();
void enableButtonInterrupt();

void phase1();
void phase2();
void phase3();

void LCDcmd(char cmd);
void LCDwrt(char ch);
void LCDclear();
void backOnTrack();

void messageWelcome();
void messageunder100();
void messagePhase1();
void messagePhase2();
void messagePhase3();
void messageButton2();
void messagesunk();
```

```
void setUpTemp();
int readTemp();

//----------------------------------------SETUP-----------------------------------

void setUp() {
            DDRA=0b00001110;         //1 är utsignal, 0 är insignal eller används inte
            DDRB=0b11111111;
            DDRC=0b00000000;
            DDRD=0b00111000;

            enableButtonInterrupt();
            sei(); //enables interuptions

            counter=0; //tidräkning=0

            first0=0;
            first1=0;
            first2=0;
            first3=0;
            first4=0;
            reached=0;

            setPin('A', PA1, 0); //grön lampa släcks
            setPin('A', PA2, 0); //gul lampa släcks
            setPin('A', PA3, 0); //röd lampa släcks

            setUpLCD();
            setUpTemp();
            setUpTimer();
            celsius=readTemp(); //läser av temperaturen
}

void setUpLCD() {
            LCDclear();
            LCDcmd(0x38); //functional set
            LCDcmd(0x0F); //display on
            LCDcmd(0x01);
            LCDcmd(0x03); //entry mode set
            messageWelcome();
}


//--------------------------------HELP-METHODS-----------------------------------
---

void setPin(char port, char pin, char state) {
            char set = 1 << pin; // bit shifts -> set = shifting 1 to the left by pin
bits -> sätter aktuell pin till 1
            if (port == 'A') {
                        set &= PORTA; //bitwise AND -> kräver att båda är 1+1=1,
1+0=0. läs på datorn
                        if(set && !state) { // om set=1 och state=0
                                    PORTA ^= set; // ändrar set eller state från 1 ->
0? Vi sätter PORTA till set. bitwise exclusive or -> 0+0=0, 1+0=1, 1+1=0
                        }
                        if(set==0 && state) { // om set=0 och state = 1 LIKA MED
NOLL???
                                    set = 1 << pin;          //sätter aktuell pin
till 1
                                    PORTA^=set;  //ändrar state från 0 -> 1
                        }
```

```
	} else if (port == 'B') {
			set &= PORTB;
			if(set && !state) {
				PORTB ^= set;
			}
			if(set == 0 && state) {
				set = 1 << pin;
				PORTB^=set;
			}
	} else if (port == 'C') {
			set &= PORTC;
			if(set && !state) {
				PORTC ^= set;
			}
			if(set == 0 && state) {
				set = 1 << pin;
				PORTC^=set;
			}
	} else if (port == 'D') {
				set &= PORTD;
				if(set && !state) {
				PORTD ^= set;
				}
				if(set == 0 && state) {
				set = 1 << pin;

				PORTD^=set;
				}
			}
	}

	//--------BUTTONS-------

	int checkButton() {
			char check = PINA & 0b00110000; //kollar PA4 och PA5, alla
andra kommer bli noll
			if(check == 16) { //om reser var intryckt (2^4)
				return 1;

			} else if (check == 32) { //om buttonTime var intryckt (2^5)
				return 2;
			}
	}


	void enableButtonInterrupt() {
			MCUCR = 0b00000011; //s 66, detta är för INT0
			SREG=0x82;
			GICR=(1<<INT0);
	}

	ISR(INT0_vect) { //Globalt button-interrupt.
			_delay_ms(1);
			int a = checkButton();
			if(a==1) {
				buttonReset=1;
			} else if (a==2) {
				buttonTime=1;
			}
	}
```

```
//---------------------------------------PHASES-----------------------------------
---------------

void phase1() {
            setPin('A', PA1, 1);
}

void phase2() {
            setPin('A', PA1, 1);
            setPin('A', PA2, 1);
}

void phase3() {
            setPin('A', PA1, 1);
            setPin('A', PA2, 1);
            setPin('A', PA3, 1);
}



            //-------------------------LCD--------------------------------------
---

void LCDcmd(char cmd) {
            _delay_ms(50);
            setPin('D', PD3,1); //R/W=1
            setPin('D',PD5,1); //RS=1
            setPin('D', PD4,1); //E=1
            PORTB=cmd;
            setPin('D', PD3,0);//R/W=0
            setPin('D',PD5,0); //RS=0

            setPin('D', PD4,0);//E=0 data läses av

            setPin('D', PD3,1); //R/W=1
            setPin('D',PD5,1); //RS=1
            setPin('D', PD4,1); //E=1
            }

void LCDwrt(char ch) { // RS alltid 1
            _delay_ms(50);
            setPin('D', PD3,1); //R/W=1
            setPin('D',PD5,1); //RS=1
            setPin('D', PD4,1); //E=1
            PORTB=ch;
            setPin('D', PD3,0); //R/W=0

            setPin('D', PD4,0); //E=0 data läses av

            setPin('D', PD3,1); //R/W=1
            setPin('D', PD4,1); //E=1
}

void LCDclear() {
            LCDcmd(0x02); //cursor home
            for(int a=0;a<40;a++) {
                        LCDwrt(' ');
            }
            LCDcmd(0xC0);
            for(int a=0;a<20;a++) {
                        LCDwrt(' ');
            }
```

```
            LCDcmd(0x02);
}

//-----------------------------------MESSAGES-------------------------------------

void messageWelcome() {
            LCDwrt('W');
            LCDwrt('E');
            LCDwrt('L');
            LCDwrt('C');
            LCDwrt('O');
            LCDwrt('M');
            LCDwrt('E');
            _delay_ms(20000);
            LCDcmd(0x01); //clear display
}

void messageunder100() {
            LCDclear();
            LCDwrt('T');
            LCDwrt('E');
            LCDwrt('M');
            LCDwrt('P');
            LCDwrt(' ');
            LCDwrt('H');
            LCDwrt('A');
            LCDwrt('S');
            LCDwrt(' ');
            LCDwrt('N');
            LCDwrt('O');
            LCDwrt('T');
            LCDwrt(' ');
            LCDcmd(0xC0);
            LCDwrt('R');
            LCDwrt('E');
            LCDwrt('A');
            LCDwrt('C');
            LCDwrt('H');
            LCDwrt('E');
            LCDwrt('D');
            LCDwrt(' ');
            LCDwrt('1');
            LCDwrt('0');
            LCDwrt('0');
            LCDwrt(' ');
            LCDwrt('D');
            LCDwrt('E');
            LCDwrt('G');
            LCDcmd(0x00);
            }

void messageOver100() {
            LCDclear();
            LCDwrt('T');
            LCDwrt('E');
            LCDwrt('M');
            LCDwrt('P');
            LCDwrt(' ');
            LCDwrt('H');
            LCDwrt('A');
            LCDwrt('S');
            LCDcmd(0xC0);
```

```
            LCDwrt('R');
            LCDwrt('E');
            LCDwrt('A');
            LCDwrt('C');
            LCDwrt('H');
            LCDwrt('E');
            LCDwrt('D');
            LCDwrt(' ');
            LCDwrt('1');
            LCDwrt('0');
            LCDwrt('0');
            LCDwrt(' ');
            LCDwrt('D');
            LCDwrt('E');
            LCDwrt('G');
            LCDcmd(0x00);
            }

void messagePhase1() {
            LCDclear();
            LCDwrt('Y');
            LCDwrt('O');
            LCDwrt('U');
            LCDwrt('R');
            LCDwrt(' ');
            LCDwrt('E');
            LCDwrt('G');
            LCDwrt('G');
            LCDwrt(' ');
            LCDwrt('I');
            LCDwrt('S');
            LCDcmd(0xC0);
            LCDwrt('S');
            LCDwrt('O');
            LCDwrt('F');
            LCDwrt('T');
            LCDwrt('-');
            LCDwrt('B');
            LCDwrt('O');
            LCDwrt('I');
            LCDwrt('L');
            LCDwrt('E');
            LCDwrt('D');
            LCDcmd(0x00);
}

void messagePhase2() {
            LCDclear();
            LCDwrt('Y');
            LCDwrt('O');
            LCDwrt('U');
            LCDwrt('R');
            LCDwrt(' ');
            LCDwrt('E');
            LCDwrt('G');
            LCDwrt('G');
            LCDwrt(' ');
            LCDwrt('I');
            LCDwrt('S');
            LCDcmd(0xC0);
            LCDwrt('M');
            LCDwrt('E');
```

```
            LCDwrt('D');
            LCDwrt('-');
            LCDwrt('B');
            LCDwrt('O');
            LCDwrt('I');
            LCDwrt('L');
            LCDwrt('E');
            LCDwrt('D');
            LCDcmd(0x00);
}

void messagePhase3() {
            LCDclear();
            LCDwrt('Y');
            LCDwrt('O');
            LCDwrt('U');
            LCDwrt('R');
            LCDwrt(' ');
            LCDwrt('E');
            LCDwrt('G');
            LCDwrt('G');
            LCDwrt(' ');
            LCDwrt('I');
            LCDwrt('S');
            LCDwrt(' ');
            LCDcmd(0xC0);
            LCDwrt('H');
            LCDwrt('A');
            LCDwrt('R');
            LCDwrt('D');
            LCDwrt('-');
            LCDwrt('B');
            LCDwrt('O');
            LCDwrt('I');
            LCDwrt('L');
            LCDwrt('E');
            LCDwrt('D');
            LCDcmd(0x00);
}
void messagesunk(){
            LCDclear();
            LCDwrt('I');
            LCDwrt('N');
            LCDwrt('C');
            LCDwrt('R');
            LCDwrt('E');
            LCDwrt('A');
            LCDwrt('S');
            LCDwrt('E');
            LCDwrt(' ');
            LCDwrt('T');
            LCDwrt('H');
            LCDwrt('E');
            LCDcmd(0xC0);
            LCDwrt('T');
            LCDwrt('E');
            LCDwrt('M');
            LCDwrt('P');
            LCDcmd(0x00);
}
```

```
void messageButton2(){
            LCDclear();
            min = (int) counter/60;
            sec= (int) counter%60;
            sec1=sec/10;
            sec2=sec%10;
            LCDwrt('T');
            LCDwrt('I');
            LCDwrt('M');
            LCDwrt('E');
            LCDwrt(':');
            LCDcmd(0xC0);
            intToChar(min);
            LCDwrt(':');
            intToChar(sec1);
            intToChar(sec2);
            LCDcmd(0x00);
}

void intToChar(int number)
{
            if(number == 0){
                        LCDwrt(0x30);
            }
            if(number == 1){
                        LCDwrt(0x31);
            }
            if(number == 2){
                        LCDwrt(0x32);
            }
            if(number == 3){
                        LCDwrt(0x33);
            }
            if(number == 4){
                        LCDwrt(0x34);
            }
            if(number == 5){
                        LCDwrt(0x35);
            }
            if(number == 6){
                        LCDwrt(0x36);
            }
            if(number == 7){
                        LCDwrt(0x37);
            }
            if(number == 8){
                        LCDwrt(0x38);
            }
            if(number == 9){
                        LCDwrt(0x39);
            }
}

void backOnTrack() {
            if(first3==1) {
                        messagePhase3();
                        } else if (first2==1) {
                        messagePhase2();
                        } else if (first1==1) {
                        messagePhase1();
                        } else if(first4==1) {
                        messageOver100();
```

```
                        } else {
                        messageunder100();
             }
}


//----------------------------TEMP-----------------------------------------

void setUpTemp() {
            ADMUX=0b01000000;
            ADCSRA=0b11000011; //delat med 8. s 217
}

int readTemp() {
            rawVoltage=ADC;
            millivolts=(rawVoltage/1024)*5000;
            celsius=(int)((millivolts/10)-273);
            ADCSRA=0b11000011;
            return celsius;
}

//-------------------------------------TIME----------------------------

void setUpTimer() { //16 bit timer
            TCCR1A=0b00000000;
            TCCR1B=0b00000011; // Prescaler =64
            TCNT1H=0xFF;
            TCNT1L=0xFF;
            TIMSK=0x04; //enable overflow interrupt
            TIFR=0x04; //enable overflow interrupt
}

ISR(TIMER1_OVF_vect) { //timer för att mäta temperaturen och räkna tid.
65536/(8Mhz/64)=0,524288 s.
            celsius=readTemp();
            if(celsius>=22 || reached==1) { //ska vara 100 celsius eller någon gång
nått 100 celsius.
                        counter=counter+0.524288;
            }
}


//-----------------------------------MAIN----------------------------

int main(void)
{
setUp();

while(1)    {
            if(celsius<22 && reached==0) { // ska vara 100 celsius

                                    if(first0==0) {
                                                LCDclear();
                                                messageunder100();;
                                    }
                                    first0=1;
                                    }

            if(celsius>=22 || reached==1) { // ska vara 100 celsius
                        reached = 1;
                        if(celsius<22) { // ska vara 100 celsius
                                    messagesunk();
```

```
                                        _delay_ms(8000);
                                        backOnTrack();
                                }
                                if(first4==0) {
                                        messageOver100();
                                }
                                first4=1;

                                if(counter>=30 && counter < 45 ) { //4-6 min -> 240-360 sek

                                        phase1();
                                        if(first1==0) {
                                                messagePhase1();
                                        }
                                        first1=1;
                                }
                                else if (counter >= 45 && counter < 60) { //6-10 min -> 360-
600 sek
                                        phase2();
                                        if(first2==0) {

                                                messagePhase2();
                                        }
                                        first2=1;
                                } else if(counter>=60) { // > 10 min -> 600 sek
                                        phase3();
                                        if(first3==0) {

                                                messagePhase3();
                                        }
                                        first3=1;
                                }

                }

                if(buttonReset==1) {
                                setUp();
                                buttonReset=0;
                }
                if(buttonTime==1) {
                                messageButton2();
                                _delay_ms(6000);
                                buttonTime=0;
                                backOnTrack();
                }

}

}
```