Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# Rapport: Väderstation

2019-05-20

Kurs: EITA15

Handledare: Bertil Lindvall och Lars-Göran Larsson

Grupp 14: Väderstation

Dominik Gashi

Rilind Zejnullahu

Mohammed Menim

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# Sammanfattning

Detta projekt har i huvudsak kretsat kring att i små grupper, självständigt bygga upp en hårdvara som, tillsammans med mjukvara, ska utgöra en användarvänlig produkt som uppfyller sitt ursprungliga syfte och tänkta funktioner.

Projektet tillhörande denna rapport handlade om att skapa en väderstation med funktioner som exempelvis att mäta temperatur, mäta luftfuktighet och visa tiden. Till detta krävdes kunskaper om såväl hårdvara som mjukvara. Det krävdes även en djup förståelse för hur dessa komponenter samarbetar med varandra och hur dem samarbetar med mjukvaran. Om förkunskaperna inte var tillräckliga, speglas det i arbetet då det ofta uppstår fel i något av områdena. Denna specialkunskap som inhämtats har varit erfarenheter från tidigare laborationer samt databladen för komponenterna. Det fanns även en del redskap som kunde användas för att bekräfta teorier, hypoteser eller dylikt. När all kunskap inhämtats och kopplingsschemat ritats, kunde projektet börja.

En plan gjordes upp med deadlines för när dem olika delarna, som exempelvis “virningen” av komponenterna, programmering av komponenterna och tester, skall vara klara. Sålänge schemat följdes skulle det inte uppstå problem när det gäller tidsmässigt. Däremot uppstod det andra problem som kunde handla om att komponenterna kopplats fel, felprogrammering eller komponenter som slutar fungera. Med hjälp av undersökningar och felsökningar kunde dessa problem däremot lösas utan större problem.

Projektet resulterade i en Väderstation som kan mäta temperatur, luftfuktighet samt visa klockan. Väderstationen kan även mäta den maximala och minimala temperaturen som uppmätts sen den slogs på, detsamma gäller även för luftfuktighet. Användaren kan, om önskas, ställa in maximum och minimum gränser för både temperatur och luftfuktighet. Om temperaturen eller luftfuktigheten hamnar utanför intervallet, visas en varning upp på skärmen följt av LED som lyses upp under skärmen.

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

## Nyckelord

Väderstation
Temperatur
Luftfuktighet
Maximum
Minimum
Projekt

## Abstract

This project was mainly revolved around small groups, independently building a hardware that together with software, should constitute a user-friendly product that fulfills its original purpose and intended functions.

The project related to this report was about creating a weather station with features such as measuring temperature, measuring humidity and showing time. In addition, knowledge of both hardware and software was required. It also required a deep understanding of how these components cooperate with each other and how they work with the software. If the prerequisites were not sufficient, this is reflected in the work, as errors often occur in one of the areas. This special knowledge obtained has been experiences from previous laboratory work and the data sheets for the components. There were also some tools that could be used to confirm theories, hypotheses or the like. When all knowledge was gathered and the wiring diagram was drawn, the project could begin.

A plan was made with deadlines for when the different parts, such as the wiring of the components, programming of the components and tests, should be done. As long as the schedule was followed, there would be no problems in terms of time. However, there were other problems that could be related to the components being falsely connected, errors in the programming part or components that suddenly stop working. However, by investigating, examining and troubleshooting, these problems could be solved without major problems or instability.

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

The project resulted in a weather station that can measure temperature, humidity and display the time. The weather station can also measure the maximum and minimum temperature that has been measured since it was switched on, the same applies to humidity. The user can, if desired, set maximum and minimum limits for both temperature and humidity. If the temperature or humidity falls outside the range, a warning is displayed on the screen followed by LED illuminated below the screen.

## Keywords

Weather station
Temperature
Humidity
Maximum
Minimum
Project

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# Innehållsförteckning

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# Förord

Detta projekt har tagit en positiv utveckling, från vår ursprungliga ide till färdig produkt. Under utvecklingens gång har vårt ingenjörsmässiga arbetssätt utvecklats enormt vilket vi anser vara mycket positivt och bra erfarenhet för eventuella framtida arbeten där det krävs från högskoleingenjörer att man kan analysera problemen och finna bra lösningar som håller långsiktigt och inte bara kortsiktigt. Vår förmåga att utveckla produkter genom att integrera mjukvara i hårdvara har förbättrats avsevärt då det inte liknar något vi gjort tidigare. Våra tidigare erfarenheter har bestått av att programmera färdiga kretsar genom att följa instruktioner. I detta projekt däremot var vi tvungna att själva bygga upp hårdvaran och sedan programmera den utan instruktioner eller hjälp från någon handledare. För att göra detta fick vi läsa på mycket om hur komponenterna fungerar och hur C programmering samarbetar med hårdvaran.

Under detta projekt har vi naturligtvis stött på problem som inte varit alltför hindrande för fortsättningen av projektet. När vi har kört fast har vi naturligtvis kunnat få lite vägledning från handledare men i princip hela projektet har varit “självständigt” eller samarbete inom gruppen. Tack vare våra olika styrkor och svagheter har vi kunnat dela upp projektet på ett sådant sätt att alla arbetat med det dem kan mest om men samtidigt lärt sig mycket om det dem kan mindre om, därför har alla fått möjligheten att utvecklas efter sin bästa förmåga.

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# 1 Inledning

I denna rapport sammanfattas ett projekt som utgör cirka en fjärdedel av kursen EITA15, Digitala system. Det huvudsakliga syftet med projektet var att använda tidigare kunskaper om hårdvara och mjukvara för att konstruera en produkt. Valet av produkt görs av varje grupp under projektets början. Den produkt som valdes i detta fall, och som denna rapport kommer handla om, är en tidsrelaterad väderstation. Stationen skall bland annat ha mätning av temperatur och luftfuktighet som funktion.

## 1.1 Bakgrund

Detta projekt görs på uppdrag av Bertil Lindvall som är kursansvarig för kursen EITA15, Digitala System, och är en del av kursens examination. Som blivande högskoleingenjörer inom datateknik är en viktig del i utbildningen och eventuella framtida jobb att kunna utveckla produkter samt arbeta i grupper. I kontrast med tidigare laborationer i Digitala system, var projektet mer självgående. Detta innebar att samarbetet ökade och de allra flesta problemen fick lösas tillsammans, med ytterst lite hjälp från handledaren. En annan skillnad från laborationerna var konceptet att själva konstruera hårdvara, vilket det krävdes ytterligare undersökningar och fördjupande kunskap för att klara av.

## 1.2 Syfte

Syftet med detta projekt är, förutom att med hjälp av mjuk- och hårdvara konstruera en fungerande och användarvänlig produkt, även att träna upp det ingenjörsmässiga tänkandet när det gäller hur man tar sig an ett projekt som detta. Gällande det ingenjörsmässiga tänkandet, finns det en del punkter som tillsammans bygger upp konceptet “ingenjörsmässigt tänkande”. Dessa punkter kan exempelvis vara, samarbetsvilja, planering, frågeställning, problemlösning, arbete i grupp och hur väl användning av kunskaper görs.

Syftet med väderstationen var att ge den funktioner, utöver “basfunktioner” för en ideell väderstation, som anses vara till fördel för användaren. Dessa funktioner kan exempelvis vara möjligheten att sätta upp gränser för temperatur och luftfuktighet samt få varningar om temperaturen eller luftfuktigheten befinner sig utanför intervallet.

## 1.3 Målformulering

Målet är att skapa en väderstation som mäter temperatur, luftfuktighet samt visar tiden i timmar, minuter och sekunder. Användaren ska även kunna ställa in intervall, det vill säga över och undre gränser för både temperatur och luftfuktighet, samt få varningar om dessa gränser överskrids eller underskrids. Det användaren också ska kunna göra är att se den högsta och lägsta uppmätta temperaturen och luftfuktighet från att stationen sätts på till tidpunkten då knappen trycks ner. Möjligheten till att ändra klockan har också getts till användaren, ifall det skulle önskas att ändra tiden.

## 1.4 Problemformulering

Väderstationen består av en del komponenter, det måste undersökas hur dessa komponenter ska kopplas.

Väderstationen ska ha en klocka som anger timmar, minuter och sekunder. Klarar den interna klockan av det som ska göras eller ska en extern klocka användas? Hur ska den korrekta tiden räknas ut?

Det finns många varianter av sensorer som kan användas för att mäta temperatur och fuktighet. Behövs två sensorer eller finns det en sensor som mäter både temperatur och fuktighet?

## 1.5 Motivering av examensarbetet

Valet av detta projektarbete berodde på ett intresse först när ett par exempel på olika projekt skumlästes. Ett av de projekten var en väderstation. Den tycktes vara mest intressant. Kod mässigt så kändes det inte för svårt på grund av tidigare erfarenhet utav exempelvis kodning av sensorer och led lampor som krävdes för tidigare laborationer. Det ledde till valet av ämne för projektet.

### 1.6 Avgränsningar

Nedan finns en kravspecifikation för produkten som konstrueras i detta projekt

- Väderstationen ska kunna läsa av, kontinuerligt uppdatera och visa temperatur, luftfuktighet och tid på displayen.
- Väderstationen ska kunna varna användaren genom en text på displayen samt en LED som lyser konstant tills problemet är åtgärdat.
- Väderstationen ska kunna låta användaren ställa in maximum- och minimum-gränser.
- Väderstationen ska kunna läsa av och spara den högsta samt lägsta uppmätta temperaturen och luftfuktigheten, samt visa upp dessa värden på displayen.
- Väderstationen ska kunna presentera korrekt tid på formen Timmar : Minuter : Sekunder.

## 2 Teknisk bakgrund

### 2.1 Hårdvara

Denna väderstation kan mäta temperatur och luftfuktighet med hjälp av en sensor och visa upp mätvärdena på en display. Stationen kan även visa tiden i timmar, minuter och sekunder korrekt. Med hjälp av en 4x4 knappsats kan användaren navigera i väderstationen och göra ändringar tack vare dess inprogrammerade funktioner.

#### 2.1.1 Komponenter

- **Processor: ATmega 1284** - En högpresterande 8-bitars AVR microcontroller med bland annat läs- och skrivningsfunktioner, 32 allmänna I/O-pinnar, en realtidsräknare, 4KB EEPROM, 16KB SRAM, två USARTs, en A/D-omvandlare och ett JTAG testgränssnitt för debugging och programmering på chip. Enheten arbetar vid mellan 1,8-5,5 volt.[1]

- **Display: SHARP Dot Matrix, GDM1602K** - En 8 bitars display med inbyggd controller. Skärmen kan visa 16 tecken, inklusive markören, per rad och har två rader. Användaren har även

[1] https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega1284.pdf

möjligheten att koppla in bakgrundsbelysning till skärmen om så önskas. Displayen arbetar vid 5V.[2]

- **Sensor: DHT11** - Denna sensor kan mäta temperatur och luftfuktighet genom att använda en kapacitiv fuktighetssensor och en termistor och skickar ut en digital signal på datapinnen. Den är billig och ganska användarvänlig, däremot kräver den mycket noggrann timing. Den enda verkliga nackdelen med den här sensorn är att du bara kan få nya data från den en gång varannan sekund.[3]

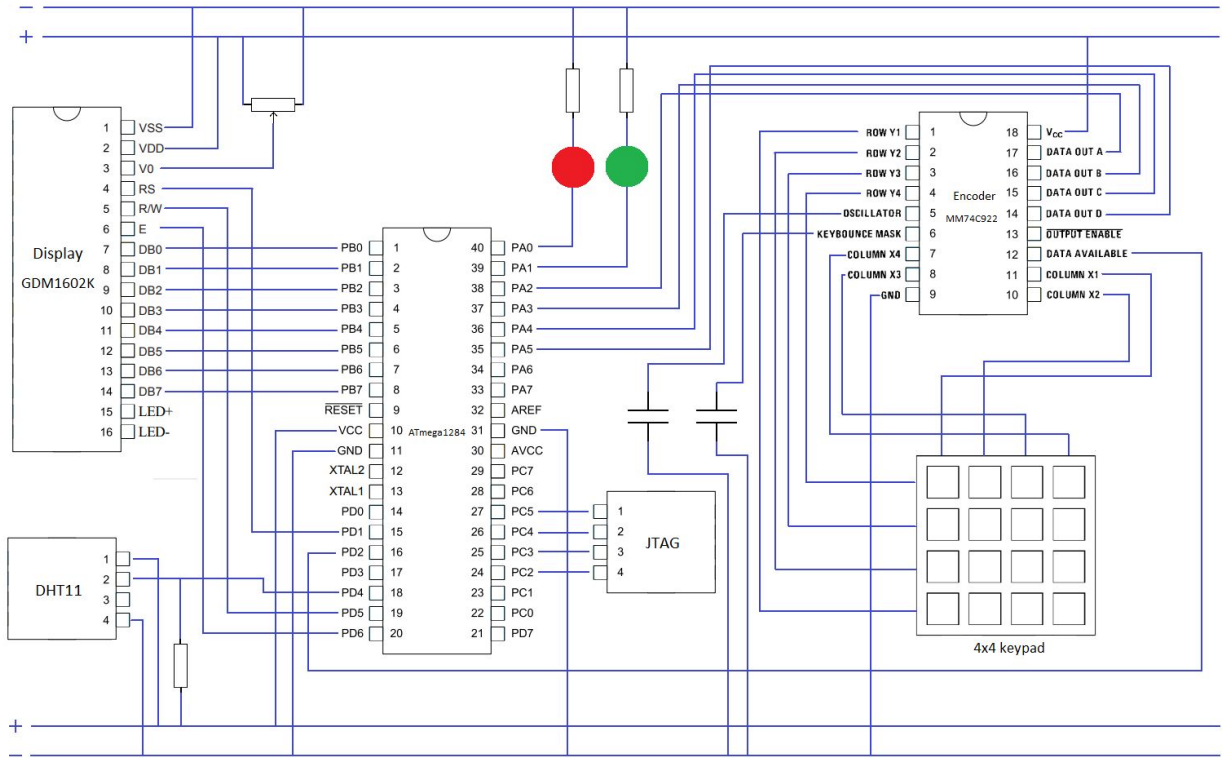
- **Keypad: 16-Key + Encoder, MM74C922** - Keypad där tangentbordsskanningen kan implementeras av antingen en extern klocka eller extern kondensator. Encodern har också "on-chip pull-up" som tillåter switchar med motstånd på upp till 50 kΩ att användas. Den inre debounce-kretsen behöver bara en enda extern kondensator. "Tillgänglig data"- signalen går till en hög nivå när en knapp trycks ner. När knappen släpps återgår signalen till en låg nivå.[4]

- **JTAG** - Används för att kunna ladda mjukvara i processorn, samt för att debugga koden.

Utöver dessa komponenter användes även både en grön och en röd LED, en potentiometer, kondensatorer och resistorer för att bygga upp hårdvaran.

[2] https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf
[3] https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf
[4] https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/Other/MM54C922.pdf

### 2.1.2 Kopplingsschema




## 2.2 Mjukvara

Mjukvaran skrevs i programmeringsspråket C, koden skrevs i utvecklingsmiljön Atmel Studio 7. För att kunna ladda mjukvaran i processorn och för att kunna debugga koden användes JTAG. För att skriva koden, användes inga speciella APIs. Den fullständiga koden finns i Appendix B.

# 3 Metod

Utförandet av projektet har varit en lång och komplicerad process som har krävt en massa planering för att kunna klara av projektet i tid. Först och främst så krävdes ett kopplingsschema över hur väderstationen ska konstrueras. Där så bestämdes alla komponenter som behövdes för väderstationen.Vidare så behövdes också resistorer, kondensatorer och en potentiometer sättas på rätt plats bland alla komponenter vilket gjorde det komplicerat. Där så var databladen för de olika komponenterna till stor hjälp. När schemat var färdig så var det dags att sätta igång och bygga ihop hårdvaran enligt kopplingsschemat. Angående hårdvaran så tog det lång tid innan den byggdes upp helt och hållet. Det berodde på en del felkopplingar och misstag i kopplingsschemat.

När det kommer till mjukvaran så delades programmeringen upp för att utnyttja förkunskaper så effektivt som möjligt, vilket har varit en huvudprincip genom hela projektarbetet. Tack vare denna strategi kunde samarbetet stärkas för att gemensamt lösa problem som uppstod. Problemen som stötts på har till en viss del kunnat lösas med hjälp av tidigare erfarenhet från bland annat tidigare kurser och laborationer. Alltför stora eller avancerade problem har kunnat lösas med hjälp av vägledning från handledare eller internet.

Skrivandet av mjukvaran har för det mesta skett i labbsalen. Genom att skriva mjukvaran i labbsalen har produkten blivit uppdaterad kontinuerligt och ofta, vilket visade sig vara fördelaktigt tidsmässigt. Hårdvaran har däremot helt och hållet byggts upp i labbsalen.

## 3.1 Källkritik

Endast datablad för de olika komponenterna användes. Databladen kom direkt från tillverkarna av komponenterna, vilket gör de trovärdiga. Utöver dessa källor kom mycket hjälp från tidigare erfarenheter från tidigare laborationer där trovärdigheten anses vara hög då metoderna testats för att se om dem fungerar. Hjälpen från handledarna har ansetts trovärdiga då de besitter flera års erfarenhet inom ämnet och arbetat med bland annat väderstationer i flera år.

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# 4 Analys

Under projektets gång så har ett par problem uppstått. Ett mindre problem har varit att lista ut hur portarna som en komponent har skulle anslutas till en annan komponents portar samt vilken ström och spänning varje komponent behövde. Det krävdes mycket sökande ute på nätet samt eget testande med hjälp av ett kopplingsdäck.

Ett större problem har däremot varit att hårdvaran slutade fungera strax innan den skulle redovisas för handledaren. Strax efter problemet uppstod kunde mjukvaran uteslutas som ursprung till problemet då den inte ändrats innan problemet uppstod. Någon minut innan hårdvaran slutade svara så testades sensorn. Slutsatsen drogs att problemet borde ligga inom hårdvaran, framförallt sensorn som precis testats. De olika komponenterna testades var för sig med hjälp av debuggern i programmet Atmel Studio 7.0 vilket i slutändan visade var problemet låg. Hypotesen om att felet kan ligga i sensorn stämde och risken var att den förstörts. För att lösa detta problem ersattes komponenten med en ny och fungerande komponent och därefter fungerade produkten som den ska

I början tänktes två analoga sensor användas för mätningen av temperatur och luftfuktighet. Efter en diskussion med handledaren, upptäcktes det att luftfuktighet sensorn är ganska dålig. Handledaren gav tips om att använda en sensor som mäter både temperatur och fuktighet, som dessutom inte är analog.

För att få kunna visa tiden, användes den interna klockan i processorn. Då det ansågs inte vara nödvändigt med en extern klocka eftersom en intern klocka räknar ut tiden med tillräckligt bra precision. Dessutom så är klockan inte huvudfunktionen i projektet. Tiden räknas ut genom att ha ett avbrott som sker varje sekund, då kommer variabeln som håller koll på sekunderna att öka med ett. Sedan kontrolleras det om timmarna, minuterna eller sekunderna ska ändras.

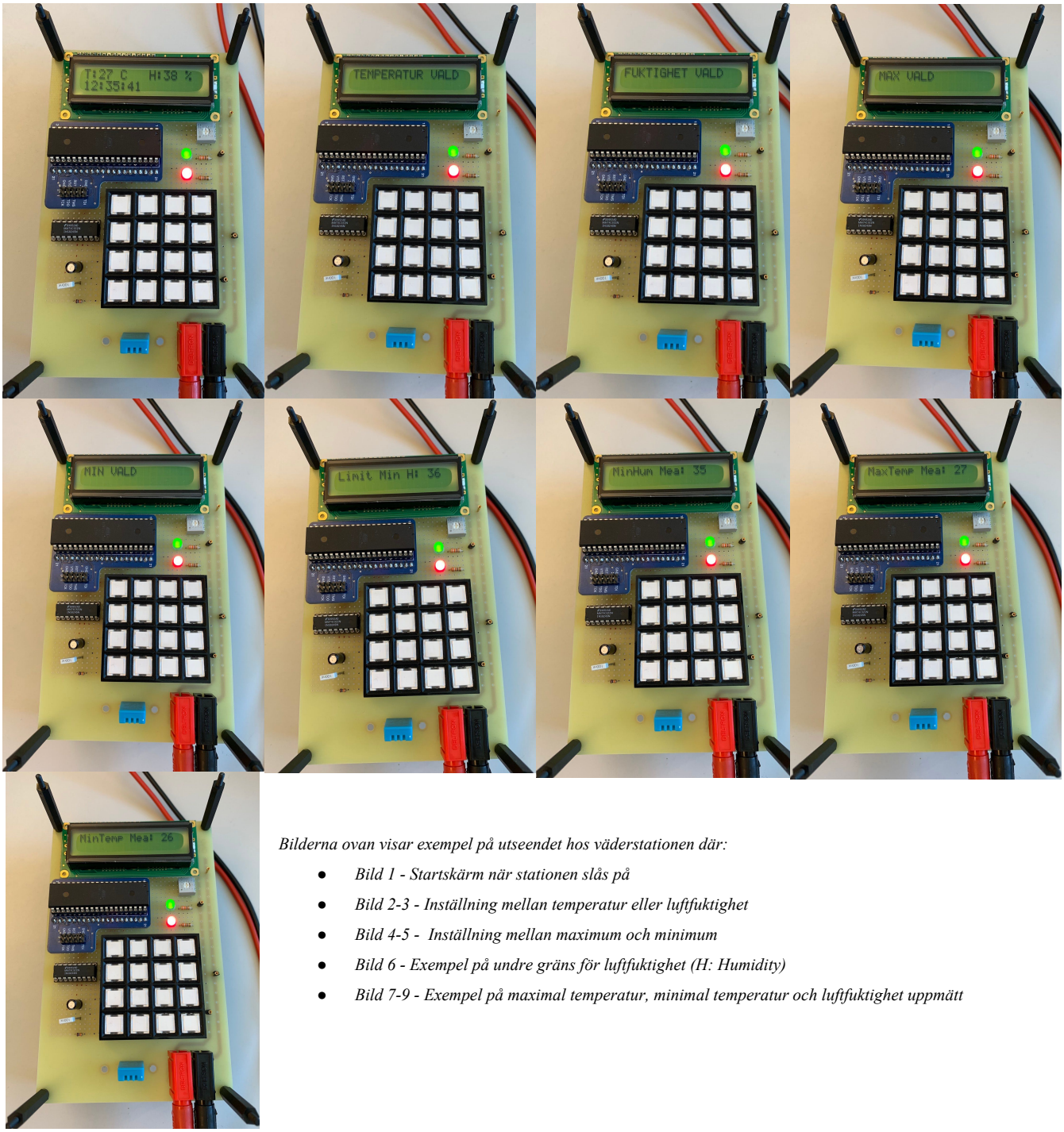
Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# 5 Resultat

Resultatet av detta projekt blev en väderstation uppbyggd av en ATmega 1284, GDM1602K, DHT11, MM74C922, tre resistorer, två kondensatorer samt en potentiometer.

Till väderstationer har även en del funktioner implementerats, som mätning av temperatur och luftfuktighet  samt visning av tiden på displayen. Användaren kan välja att ställa in maximum och minimum gränser för både temperatur och luftfuktighet, varvid en röd eller grön lampa tänds då gränserna överskrids. Om maximum-gränsen för temperaturen  och/eller luftfuktigheten överskrids, tänds den röda lampan. Om minimumgränsen för temperaturen och/eller luftfuktigheten underskrids, tänds den gröna lampan. I samband med att varningslamporna tänds dyker även en varningstext upp på skärmen som talar om vilken gräns som överskridits eller underskridits och om det gäller temperatur eller luftfuktighet. Utöver dessa funktioner kan användaren även se den maximala uppmätta temperaturen och luftfuktigheten från att stationen sätts på till den tidpunkt då knappen trycks ner. På samma sätt kan användaren även kolla minimala uppmätta temperatur och luftfuktighet. Projektet har även resulterat i en mycket enkel HTML-baserad hemsida som sammanfattar produkten och dess huvudfunktioner.

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019




*Bilderna ovan visar exempel på utseendet hos väderstationen där:*

- *Bild 1 - Startskärm när stationen slås på*
- *Bild 2-3 - Inställning mellan temperatur eller luftfuktighet*
- *Bild 4-5 - Inställning mellan maximum och minimum*
- *Bild 6 - Exempel på undre gräns för luftfuktighet (H: Humidity)*
- *Bild 7-9 - Exempel på maximal temperatur, minimal temperatur och luftfuktighet uppmätt*

# 6 Slutsats

Resultatet av detta projekt har varit mycket tillfredsställande. Alla funktioner som önskades kunde implementeras. Dessutom blev projektet klart fortare än ursprungligen planerat och tänkt, därför har mer tid till att testa produkten spenderats, vilket i sin tur leder till ett resultat med färre fel eller problem. Med detta projekt har det inte funnits en direkt frågeställning mer än “går det att bygga en station med dem funktioner som önskas”. Den frågeställningen har besvarats med ett ja - det går att bygga en väderstation från grunden. Processen från tanke till färdig produkt har gått utan större problem eller hinder och det har definitivt skett en utveckling i det ingenjörsmässiga tänkandet och arbetandet i allmänhet vilket är en erfarenhet som följer med hela livet, detta med tanke på den begränsade erfarenheten från tidigare.

## 6.1 Framtida utvecklingsmöjligheter

Klockan som används i väderstationen kör inte när den är avstängd. Vilket innebär att klockan måste ställas in varje gång väderstationen sätts på. En extern klocka hade kunnats implementeras som funkar även när väderstationen är avstängd, så att användaren slipper ställa in klockan varje gång. Med en klocka som kör dygnet runt kan nya funktioner tilläggas. En av dessa är att man kan mäta medeltemperaturen över ett dygn. Hade även kunnat lägga till en kalender som håller reda på vad medeltemperaturen var varje dag och så vidare. Displayen som användes kunde bara visa 32 tecken i taget. Med en annan typ av display hade grafer av hur temperaturen varierar under dagen kunnats ritas.

Det tänkta användningsområdet har varit i miljöer där det är viktigt att temperatur och luftfuktigheten är reglerad och under kontroll. Men om ovanstående funktioner implementeras hade den kunnat fungera i miljöer som vanliga hem eller skolor och dylikt.

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# 7 Terminologi

“Basfunktioner för en ideell väderstation” - Väderstation vars funktioner är att visa temperatur, luftfuktighet och tid.

“Ingenjörsmässigt tänkande/ arbetssätt” - Förmågan hitta långsiktigt hållbara lösningar till problem som dyker upp samt använda nytänkande till att förbättra ex. En produkt.

# 8 Källförteckning

## Datablad:

“DHT11 Humidity & Temperature Sensor, ” *Mouser.com* [online] Tillgänglig: https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf.[Hämtad: 2019-05-20]

“MM54C922/MM74C922 16-Key Encoder,” *eit.lth.se,* 2019-04-09. [online] Tillgänglig: https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/Other/MM54C922.pdf [Hämtad: 2019-05-20]

“GDM1602K,”*eit.lth.se,* 1993-07. [online] Tillgänlig: https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf [Hämtad: 2019-05-20]

“ATmega1284,” *Microchip.com,* 2016-10. [online] Tillgänglig: https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega1284.pdf [Hämtad: 2019-05-20]

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

# 9 Appendix

## 9.1 Appendix A

Manual för knappsats.

1: Starta/släck displayen. Visar nuvarande temperatur, fuktighet och tid(klocka).

2: Välj om man vill justera timmar eller minuter. Alterneras mellan valen varje gång man trycker på knappen.

3: Gör så att man kan justera klockan, tryck på knappen när du har justerat klart.

4: Välj om man vill justera temperaturen eller fuktigheten. Alterneras mellan valen varje gång man trycker på knappen.

5: Öka gränsen/tiden med +1

6: Öka gränsen/tiden med +5

7:  Öka gränsen/tiden med +10

8: Välj om man vill justera max- eller mingränserna. Alterneras mellan valen varje gång man trycker på knappen.

9: Sänk gränsen/tiden med -1

10: Sänk gränsen/tiden med -5

11: Sänk gränsen/tiden med -10

12: Visa värdet på gränsen man har valt(max/min och temperatur/fuktighet)

13: Visar Maximala uppmätta fuktigheten

14: Visar Minimala uppmätta fuktigheten

15: Visar Maximala uppmätta temperaturen

16: Visar Minimala uppmätta temperaturen

## 9.2 Appendix B

```
#define F_CPU 8000000UL

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

#define RW 5
#define Enable 6
#define RS 1
#define DA 2
#define DHT11 4

volatile char val;
volatile int timer;
volatile int newChoice;
volatile int seconds = 5;
volatile int minutes = 33;
volatile int hours = 12;
volatile uint8_t data = 0;
volatile uint8_t humidity_H;
volatile uint8_t humidity_D;
volatile uint8_t temp_H;
volatile uint8_t temp_D;
volatile uint8_t checkSum;
volatile uint8_t toggleTH;
volatile uint8_t toggleMM;
volatile int limitMaxTemp = 40;
volatile int limitMinTemp = 0;
```

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

```
volatile int limitMaxHum = 40;
volatile int limitMinHum = 0;
volatile int maxTemp;
volatile int minTemp;
volatile int maxHum;
volatile int minHum;
volatile int tempBool = 0;

volatile int limitBool = 0;
volatile int limitBool2 = 0;
volatile int highSelect = 0;
volatile int minSelect = 0;

volatile int toggleClk = 0;
volatile int toggleClkHM = 0;
volatile int onOrOff = 0;

void SendCmd(unsigned char command) { //Send commands to display
        PORTD |= (1<<Enable);
        PORTD &= ~ (1 << RW | 1 << RS);
        PORTB = command;
        _delay_ms(5);
        PORTD &= ~ (1<<Enable);
        PORTB = 0;
}

void SendData(unsigned char data) { //Send data to display(Display characters)
        PORTD |= (1<<Enable);
        PORTD &= ~ (1 << RW);
        PORTD |=   (1 << RS);
        PORTB = data;
        _delay_ms(5);
```

```
	PORTD &= ~ (1<<Enable);
	PORTB = 0;
}

void clearDisplay(){
	SendCmd(0x01);
	_delay_ms(5);
}

void displayOn(){
	SendCmd(0b00001100);
	clearDisplay();
	SendCmd(0x3C);
}

void displayOff(){
	SendCmd(0b00001000);
}

void SendText(char text[]){ //Display text
	int i = 0;
	while(text[i] != '\0'){
		SendData(text[i]);
		i++;
	}
}

void SendNbr(int nbr){  //Display numbers
	char talet[sizeof(nbr) * 4 + 1];
	sprintf(talet, "%d", nbr);
	SendText(talet);
}
```

Dominik Gashi, Rilind Zejnullahu, Mohammed Menim
Väderstation, EITA15 - Digitala System
IDA1, Datateknik 2019

```
void initTimer() {
	TCCR1B |= (1 << WGM12); // Configure timer 1 for CTC mode
	TIMSK1 |= (1 << OCIE1A); // Enable CTC interrupt

	OCR1A = 31249;// Set CTC compare value to 1Hz at 8 MHz AVR clock, with a prescaler of 256
	TCCR1B |= (1 << CS12); // Start timer at Fcpu/256

	sei();
}

void startSensor(){
	//REQUEST
	DDRD |= (1<<DHT11); //DHT11 PIN to OUTPUT
	PORTD &= ~(1 << DHT11); //Set pin to LOW
	_delay_ms(25); //At least 18ms delay
	PORTD |= (1 << DHT11); //Set pin to HIGH

	//RESPONSE
	DDRD &= ~(1 << DHT11); //DHT11 PIN to INPUT
	while(PIND & (1<<DHT11)); //Checks DHT11 PIN, WAIT UNTIL SIGNAL GETS LOW
	while((PIND & (1<<DHT11))==0); //Checks DHT11 PIN, WAIT UNTIL SIGNAL GETS HIGH
	while(PIND & (1<<DHT11));//Checks DHT11 PIN, WAIT UNTIL SIGNAL GETS LOW

}

uint8_t receiveData(){
	for(int i = 1; i <= 8; i++){  //8 bits at a time
		while((PIND & (1<<DHT11))==0); //Wait until DHT11 is high
		_delay_us(30);
		if(PIND & (1<<DHT11)){ //If DHT11 is still high after delay, data bit is '1'
			data = (data<<1)|(0b00000001);
```

```
                }
                else { //data bit is '0'
                        data = (data<<1);
                }
                while(PIND & (1<<DHT11)); //Wait until DHT11 is low, starting to transmit new bit
        }
        return data;
}

void measure(){ //Receive value of temperature and humidity
        humidity_H = receiveData();
        humidity_D = receiveData();
        temp_H = receiveData();
        temp_D = receiveData();
        checkSum = receiveData();
}

void initPins(){
        DDRB = 0b11111111; // pins B to output
        DDRD |= (1<<Enable) | (1<<RW) | (1<<RS);
        DDRA = 0b00000011;
}

void printTH(){ //Print temperature and humidity
        startSensor();
        measure();
        if((humidity_D + humidity_H + temp_D + temp_H) != checkSum){
                //Error, wrong value
        }
        else{
                SendCmd(0b10000000);
                SendText("T:");
```

```
                SendNbr(temp_H);
                SendText(" C");
                SendCmd(0b10001001);
                SendText("H:");
                SendNbr(humidity_H);
                SendText(" %");
        }
}

void calcTime() {
        if (timer == 1) { //Timer gets the value "1" through the interrupt that occurs every second
                seconds++;
                hours %= 24;
                minutes %= 60;
                if (seconds == 60) {
                        minutes++;
                        seconds = 0;
                }
                if (minutes == 60) {
                        hours++;
                        minutes = 0;
                }
                if (seconds < 10) {
                        SendCmd(0b10101110);
                        SendNbr(0);
                        SendNbr(seconds);
                }
                if (minutes < 10) {
                        SendCmd(0b10101011);
                        SendNbr(0);
                        SendNbr(minutes);
                        SendText(":");
```

```
                }
                if (hours < 10) {
                        SendCmd(0b10101000);
                        SendNbr(0);
                        SendNbr(hours);
                        SendText(":");
                }
                if (seconds >= 10) {
                        SendCmd(0b10101110);
                        SendNbr(seconds);
                }
                if (minutes >= 10) {
                        SendCmd(0b10101011);
                        SendNbr(minutes);
                        SendText(":");
                }
                if (hours >= 10) {
                        SendCmd(0b10101000);
                        SendNbr(hours);
                        SendText(":");
                }
                printTH(); //Print temperature and humidity every second.
                timer = 0;
                SendCmd(0b10000000);
        }
}

void displayLimit(){ //Display the limits
        if(toggleTH == 1){ //Temperature selected
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Limit Temperature");
```

```
                SendCmd(0b10101000);
                SendText("Max:");
                SendNbr(limitMaxTemp);
                SendCmd(0b10110001);
                SendText("Min:");
                SendNbr(limitMinTemp);
                _delay_ms(1000);
                clearDisplay();
        }
        else if(toggleTH == 0){ //Humidity selected
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Limit Humidity");
                SendCmd(0b10101000);
                SendText("Max:");
                SendNbr(limitMaxHum);
                SendCmd(0b10110001);
                SendText(" Min:");
                SendNbr(limitMinHum);
                _delay_ms(1000);
                clearDisplay();
        }
}

void calcLimit(int amount){ //Increase or decrease the limits
        if(toggleClk == 1){ //If "Adjust time" button is pressed
                if(toggleClkHM == 1) { //Hours selected
                        if((hours + amount) < 0){
                                hours = 24 + (hours + amount);
                                seconds = -1; // Seconds++ before time is calculated in calcTime()
                        }
                        else{
```

```
                        hours+= amount;
                        seconds = -1;
                }
        }
        else if(toggleClkHM == 0) { //Minutes selected
                if((minutes + amount) < 0){
                        minutes = 60 + (minutes + amount);
                        seconds = -1;
                }
                else{
                        minutes+= amount;
                        seconds = -1;
                }
        }
}
else if(toggleTH == 1 && toggleMM == 1){ //Temperature and max limit selected
        limitMaxTemp += amount;
        clearDisplay();
        SendCmd(0b10000000);
        SendText("Limit Max T: ");
        SendNbr(limitMaxTemp);
        _delay_ms(1000);
        clearDisplay();
}
else if(toggleTH == 1 && toggleMM == 0){ //Temperature and min limit selected
        limitMinTemp += amount;
        clearDisplay();
        SendCmd(0b10000000);
        SendText("Limit Min T: ");
        SendNbr(limitMinTemp);
        _delay_ms(1000);
        clearDisplay();
```

```
        }
        else if(toggleTH == 0 && toggleMM == 1){ //Humidity and max limit selected
                limitMaxHum += amount;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Limit Max H: ");
                SendNbr(limitMaxHum);
                _delay_ms(1000);
                clearDisplay();
        }
        else if(toggleTH == 0 && toggleMM == 0){ //Humidity and min limit selected
                limitMinHum += amount;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Limit Min H: ");
                SendNbr(limitMinHum);
                _delay_ms(1000);
                clearDisplay();
        }
}

void limitCheck(){
        //A bit complicated, two LEDS to show if max/min limits of temperature/humidity are exceeded.
        //Red LED shows the state of the MAX limit, Green LED shows the state of the MIN limit.

        if(limitBool == 0){ //Code for handling MAX limit. Warn ONE time if you exceed a limit
        if(temp_H > limitMaxTemp && humidity_H > limitMaxHum){
                limitBool = 1;
                highSelect = 1; //1 = combination of temp&hum over both Max limits
                PORTA |= (1<<PINA0);
                clearDisplay();
                SendCmd(0b10000000);
```

```
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("HIGH TEMP & HUM");
                _delay_ms(1000);
                clearDisplay();
        }
        else if(temp_H > limitMaxTemp){
                limitBool = 1;
                highSelect = 2; //2 = temp over MAX limit
                PORTA |= (1<<PINA0);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("HIGH TEMP");
                _delay_ms(1000);
                clearDisplay();
        }
        else if(humidity_H > limitMaxHum){
                limitBool = 1;
                highSelect = 3; //3 = humidity over MAX limit
                PORTA |= (1<<PINA0);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("HIGH Humidity");
                _delay_ms(1000);
                clearDisplay();
        }
                else{
                        PORTA &= ~(1<<PINA0); //TURN OFF RED LIGHT
```

```
                }
}


        if(limitBool == 1) { //If MAX limit has been exceeded
                if(highSelect == 1 && temp_H < limitMaxTemp && humidity_H < limitMaxHum){
//If temp&hum was over MAX limits, but now both are under the MAX.
                        limitBool = 0; //CAN WARN AGAIN
                }
                else if(highSelect == 1 && temp_H > limitMaxTemp && humidity_H < limitMaxHum)
{
 //If temp&hum was over MAX limits, but now humidity is under the MAX
                        limitBool = 0;
                }
                else if(highSelect == 1 && temp_H < limitMaxTemp && humidity_H > limitMaxHum)
{
//If temp&hum was over MAX limits, but now temp is under the MAX
                        limitBool = 0;
                }
                else if(highSelect == 2 && temp_H < limitMaxTemp) {
 //If temp was over MAX limit, but now temp is under the MAX
                        limitBool = 0;
                }
                else if(highSelect == 2 && temp_H > limitMaxTemp && humidity_H > limitMaxHum)
{
//If temp was over MAX limit, but now humidity is also over MAX
                        limitBool = 0;
                }
                else if(highSelect == 3 && humidity_H < limitMaxHum){
 //If humidity was over MAX limit, but now humidity is under MAX
                        limitBool = 0;
                }
```

```
                else if(highSelect == 3 && humidity_H > limitMaxHum && temp_H  >
limitMaxTemp){
//If humidity was over MAX limit, but now temp is also over MAX
                        limitBool = 0;
                }
        }

        if(limitBool2 == 0){ //Code for handling MIN limit. Warn ONE time if you exceed a limit
        if(temp_H < limitMinTemp && humidity_H < limitMinHum){
                limitBool2 = 1;
                minSelect = 1;
                PORTA |= (1<<PINA1);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("LOW TEMP & HUM");
                _delay_ms(1000);
                clearDisplay();
        }
        else if(temp_H < limitMinTemp){
                limitBool2 = 1;
                minSelect = 2;
                PORTA |= (1<<PINA1);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("LOW TEMP");
                _delay_ms(1000);
                clearDisplay();
        }
```

```
        else if(humidity_H < limitMinHum){
                limitBool2 = 1;
                minSelect = 3;
                PORTA |= (1<<PINA1);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("LOW HUMIDITY");
                _delay_ms(1000);
                clearDisplay();
        }
        else{
                PORTA &= ~(1<<PINA1); //Turn OFF Green light
        }
        }
                if(limitBool2 == 1) {
                        if(minSelect == 1 && temp_H > limitMinTemp && humidity_H >
limitMinHum){
                                limitBool2 = 0;
                        }
                        else if(minSelect == 1 && temp_H > limitMinTemp && humidity_H <
limitMinHum) {
                                limitBool2 = 0;
                        }
                        else if(minSelect == 1 && temp_H < limitMinTemp && humidity_H >
limitMinHum) {
                                limitBool2 = 0;
                        }
                        else if(minSelect == 2 && temp_H > limitMinTemp){
                                limitBool2 = 0;
                        }
```

```
                        else if(minSelect == 2 && temp_H < limitMinTemp && humidity_H <
limitMinHum) {
                                limitBool2 = 0;
                        }
                        else if(minSelect == 3 && humidity_H > limitMinHum){
                                limitBool2 = 0;
                        }
                        else if(minSelect == 3 && humidity_H < limitMinHum && temp_H <
limitMinTemp){
                                limitBool2 = 0;
                        }
                }
}

void calcTempHum(uint8_t temp, uint8_t hum){
//Calculate the highest & lowest temperature and humidity measured
        if(tempBool == 0 && temp != 0){ //First measured temp and hum gets set as max and min
                tempBool = 1;
                maxTemp = temp;
                minTemp = temp;
                maxHum = hum;
                minHum = hum;
        }
        if(temp > maxTemp){
                maxTemp = temp;
        }
        if(temp < minTemp){
                minTemp = temp;
        }
        if(hum > maxHum){
                maxHum = hum;
        }
```

```
		if(hum < minHum){
			minHum = hum;
		}
}

ISR(TIMER1_COMPA_vect) //Interrupt every second
{
		timer = 1;

}

ISR(INT0_vect) //Keypad interrupt
{
		val= PINA & 0b11110000;
		newChoice = 1;
}

void keyConfig()
{
		EICRA |= (1 << ISC00) | (1 << ISC01);
		EIMSK |= (1 << INT0);
		DDRD &= ~(1 << DA);
}

int main(void) {
		initPins();
		displayOff();
		sei();
		initTimer();
		keyConfig();

		while (1) {
```

```
calcTime();
limitCheck();
calcTempHum(temp_H, humidity_H);
if (newChoice == 1) {
    if (val == 0xc0) { //1 DISPLAY ON
        if (onOrOff == 0) {
            onOrOff++;
            displayOn();
            } else {
            onOrOff--;
            displayOff();
        }
    }
    if (val == 0x40) { // 2 //Toggle between hours and minutes(clock)
        if (toggleClkHM == 0) {
            toggleClkHM++;
            clearDisplay();
            SendCmd(0b10000000);
            SendText("Timmar vald");
            _delay_ms(1000);
            clearDisplay();
            } else {
            toggleClkHM--;
            clearDisplay();
            SendCmd(0b10000000);
            SendText("Minuter vald");
            _delay_ms(1000);
            clearDisplay();
        }

    }
```

```
if (val == 0x80) { // 3 //Press to be able to change the clock, press again when
done.
        if (toggleClk == 0) {
                toggleClk++;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Justera klockan");
                _delay_ms(1000);
                clearDisplay();
                } else {
                toggleClk--;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Klar!");
                _delay_ms(1000);
                clearDisplay();
        }
}
if (val == 0x00) { // 4 Toggle between Temperature and Humidity
        if (toggleTH == 0) {
                toggleTH++;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("TEMPERATUR VALD ");
                _delay_ms(1000);
                clearDisplay();
                } else {
                toggleTH--;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("FUKTIGHET VALD ");
                _delay_ms(1000);
```

```
                clearDisplay();
            }
        }
        if (val == 0xe0) { // 5 Increase limit by +1
            calcLimit(1);
        }
        if (val == 0x60) { // 6 Increase limit by +5
            calcLimit(5);
        }
        if (val == 0xa0) { // 7 Increase Limit by +10
            calcLimit(10);
        }
        if (val == 0x20) { // 8 Toggle between Max Limit and Min Limit
            if (toggleMM == 0) {
                toggleMM++;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("MAX VALD ");
                _delay_ms(1000);
                clearDisplay();
                } else {
                toggleMM--;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("MIN VALD ");
                _delay_ms(1000);
                clearDisplay();
            }
        }
        if (val == 0xd0) { // 9 Decrease limit by -1
            calcLimit(-1);
        }
```

```
if (val == 0x50) { // 10 Decrease limit by -5
        calcLimit(-5);
}
if (val == 0x90) { // 11 Decrease limit by -10
        calcLimit(-10);
}
if (val == 0x10) { // 12 Show Max/Min limit without changing it
        displayLimit();
}
if (val == 0xf0) { // 13 Show the highest measured humidity
        clearDisplay();
        SendCmd(0b10000000);
        SendText("MaxHum Measured:");
        SendCmd(0b10101000);
        SendNbr(maxHum);
        SendText(" %");
        _delay_ms(1000);
        clearDisplay();
}
if (val == 0x70) { // 14 Show the lowest measured humidity
        clearDisplay();
        SendCmd(0b10000000);
        SendText("MinHum Measured");
        SendCmd(0b10101000);
        SendNbr(minHum);
        SendText(" %");
        _delay_ms(1000);
        clearDisplay();
}
if (val == 0xb0) { // 15 Show the highest measured temperature
        clearDisplay();
        SendCmd(0b10000000);
```

```
                                SendText("MaxTemp Measured");
                                SendCmd(0b10101000);
                                SendNbr(maxTemp);
                                SendText(" C");
                                _delay_ms(1000);
                                clearDisplay();
                        }
                        if (val == 0x30) { // 16 Show the lowest measured temperature
                                clearDisplay();
                                SendCmd(0b10000000);
                                SendText("MinTemp Measured");
                                SendCmd(0b10101000);
                                SendNbr(minTemp);
                                SendText(" C");
                                _delay_ms(1000);
                                clearDisplay();
                        }
                        newChoice = 0;
                }
        }
}
```