Väderstation med ‘seglarindikator‘

Markus Carlsson, ine15mca Johan Harrysson, jo0287ha-s
Philip Mårtensson Qvistberg, ph8105qv-s

Handledare: Bertil Lindvall & Christoffer Cederberg

22 maj 2019

Abstract

The purpose of the project was to illustrate how developing work can be done in the industry by developing a prototype sailing indicator, a type of weather station, with capability of indicating whether it is a good idea to get out on the water with a sailing boat. The result is based on temperature and wind speed which the product can show independently as well. With the help from tutors, classmates and many hours checking data sheets the project resulted in a successful prototype with a simple interface.

Innehåll

1 Syfte

Syftet med denna rapport är att beskriva och dokumentera det arbete som har utförts i samband med att konstruera en elektronisk väderstation med inbyggd funktion som bedömer om det är seglarväder, alltså en seglarindikator. Denna process och arbetssätts syfte är vidare att uppfylla kursen Digitala Projekt, EITF12, övergripande syfte att illustrera industriellt utvecklingsarbete genom att bland annat söka upp och tillgodogöra sig relevant information för att kunna realisera digitala system av låg och medelhög komplexitet.

2 Produkt

Den produkt gruppen bestämde sig för att konstruera blev seglarindikator, det vill säga en väderstation med specifik funktionalitet att ange om det är en bra dag att segla eller om det är bättre att hålla sig på land. Genom att analysera temperatur och vindhastighet ges ett resultat i om det är för varmt, för kallt, för blåsigt, inte tillräckligt med vind eller om det är helt perfekt väderförhållanden för att ta sig ut på de sju haven, eller närmsta dypöl för en trevlig seglats. Utöver denna funktion ska produkten även kunna fungera som en traditionell väderstation och visa tidpunkt, temperatur och vindhastighet.

2.1 Kravspecifikation

- Produkten ska ha ett enkelt men användarvänligt interface bestående av en alfanumerisk display, tre knappar och två LED:er.
- Produkten ska kunna visa aktuell tidpunkt.
- Produkten ska kunna visa aktuell temperatur.
- Produkten ska kunna visa aktuell vindhastighet.
- Produkten ska kunna ange om det är seglarväder eller ej. (Seglarväder definieras som temperatur mellan x och y samt vindhastighet mellan 4 och 26 grader Celsius samt vindstyrka mellan 2 och 7 meter per sekund).
- Knapparna ska användas för att visa temperatur, vindhastighet eller seglarindikation på displayen. Tid visas alltid.
- Den ena lysdioden ska lysa då seglarväder råder och den andra då det ej är seglarväder.

2.2 Komponenter

- 1x Atmel ATmega16. 8-bitars mikrokontroller med 40 pins baserad på AVR RISC-arkitektur. Innehåller bland annat 3 inbyggda timrar, 16k programmerbart flash-minne samt AD- och DA-omvandlare. Detta är "hjärnan" i väderstationen, som samlar in och processar information från vind- och temperaturgivarna, räknar ut om det råder seglarförhållanden, samt skriver ut text på displayen.
- 1x Texas Instruments LM335. Analog temperaturgivare med Kelvinskala. Används för att mäta temperaturen.
- 1x Xiamen Ocular GDM1602K. Dot-matrix alfanumerisk LCD display med 2x16 tecken. Används för att kommunicera information om rådande väderförhållanden till användaren.
- 1x Vindsensor i form av en vindsnurra som genererar en puls för varje varv som snurran snurrat. Används för att mäta vindstyrkan.
- 1x Microchip MCP7940M. Realtidsklocka och kalender (RTCC) som kommunicerar via I2C-protokollet. Används för att hålla reda på nuvarande datum och tid, samt för att kunna räkna ut medelvärden med jämna mellanrum.
- 1x grön lysdiod (LED). Används för att indikera goda seglarförhållanden.
- 1x gul lysdiod (LED). Används för att indikera dåliga seglarförhållanden
- 1x 32,768 kHz kristall. Används för att RTC:n ska kunna hålla tiden.
- 3x knappar. Används för att navigera i användargränssnittet samt styra seglarindikatorn.
- 1x vridpotentiometer. Används för att justera displayens kontrast.
- 2x 100 nF foliekondensatorer.
- 1x spole
- diverse motstånd

2.3 Kopplingsschema

Se bilaga 1.

3 Metod

Arbetet inleddes med att bestämma seglarindikatorns omfattning och vilka funktioner som skulle implementeras. Detta resulterade till slut i en kravspecifikation. Efter detta bestämdes vilka huvudsakliga komponenter som skulle användas och hur dessa, på ett logiskt plan, skulle sammankopplas med hjälp av ett blockschema.

Utifrån blockschemat togs ett kopplingschema fram med alla komponenter som skulle krävas och godkändes av handledaren. Konstruktionsarbetet inleddes och i enlighet med kopplingsschemat fästes komponenter på kortet genom att lödas fast. Dessa kopplades samman med tråd som virades på komponenternas ben med virpistol. Arbetet skedde etappvis där processor, LED:er och knappar först monterades, vidare monterades display, temperatursensor, RTC och till sist vindsensor. Mellan varje etapp testades kortet med hjälp av JTag för att lättare hitta hårdvarufel som kunnat uppstå i konstruktionsfasen.

Mjukvaran, som skrevs i Atmel Studio 7.0 i språket C, konstruerades till en början etappvis i takt med att kortet fylldes med komponenter men den större delen av arbetet tog fart då hårdvaran var slutkonstruerad och de mer avancerade komponenterna, som RTC och Vindsensor, skulle inkorporeras i systemet med sina respektive funktioner. Slutligen när alla komponenterna var för sig gav korrekt information skapades ett användarvänligt interface som sammanställde informationen. Datablad för respektive komponent inspekterades grundligt för att rätt information skulle skickas och tas emot av processorn.

4 Resultat

Arbetet resulterade i en seglarindikator som uppfyllde de krav på funktionalitet som vi satt upp enligt kravspecifikationen i subsection2.1. Ett misstag som upptäcktes var att korrekt färgkonvention ej följts då sladdarna virades för att koppla ihop komponenterna. Svart sladd till GND, vit sladd till VCC och gul sladd för data. Då sladdarna var konsekvent kopplade åtgärdades det ej.

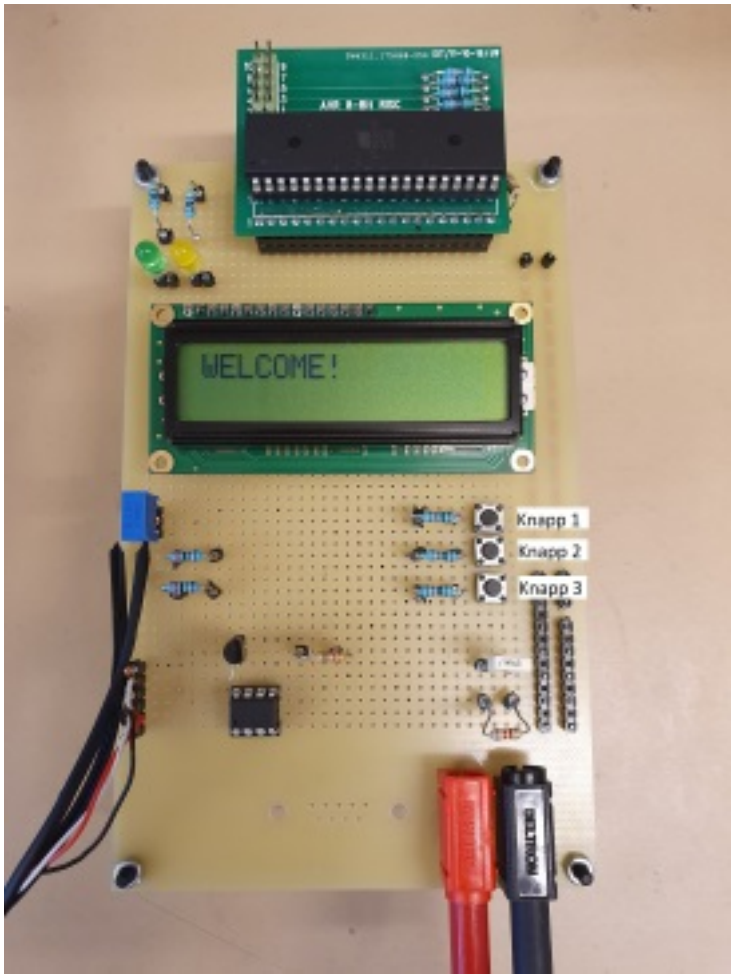
4.1 Användarmanual

Då seglarindikatorn startas välkomnas användaren av systemet och genom att trycka på de tre olika knapparna kan olika information visas på displayen. Knapparnas position kan ses i figur figure1.

- Knapp 1 ger aktuell tid, temperatur och vind

- Knapp 2 ger medeltemperatur och medelvind
- Knapp 3 ger information om det råder seglarförhållanden eller ej
- De två LED:er, en grön och en gul ger information om det råder seglarförhållanden eller ej.
- Grön LED innebär att seglarförhållanden råder.
- Gul LED innebär att seglarförhållanden ej råder.

Figur 1: Seglarstationen direkt efter uppstart. Knapparnas positioner utmärkta.



5 Diskussion

5.1 Utmaningar

De utmaningar gruppen stötte på grundades till stor del på den begränsade erfarenheten av liknande projekt och att kunskaperna inom ellära och C-programmering var knapp.

Hårdvarumässigt monterades kristallen på RTC felaktigt, för långt från RTC och med koppling till grund vilket den ej skulle ha. Detta åtgärdades med hjälp av handledare efter lång felsökning från gruppens sida. Vid montering av vindsensor behövde även ett redan upptaget ben på processorn (INT0) användas vilket resulterade i att displayen till viss del kopplas om och kopplingsschemat uppdaterades. Dessutom behövdes ett "pull down"-motstånd för att vindsensor skulle kunna ge de impulser som önskades till processorn.

När mjukvaran utvecklades uppstod flera svårigheter. Dels var de komponenterna vi arbetade med som exempelvis vår RTC och displayen svårprogrammerade och dels hade vi som grupp få erfarenheter inom C-programmering. En utmaning som vi tidigt identifierade var hur vi skriva mjukvaran så att displayen skulle fungera. Det gick att konstatera att all information fanns tillgänglig i tillhörande datablad men att det inte var så enkelt att omsätta till mjukvara. Detta var den första komponenten som vi skrev mjukvaran till på egen hand vilket gjorde att vi fortfarande hade bristfällig erfarenhet att C-programmering.

En ytterligare utmaning var att få vår RTC att fungera. Trots de hårdvarufel som identifierades var vår RTC svårprogrammerad då den kommunicerade genom ett protokoll som heter I2C. För att lösa utmaningen användes inspiration från en färdigskriven c-fil med I2C-protokollet efter godkännande av handledarna. Detta löste endast delar av utmaningen då det fortfarande fanns svårigheter i att få vår RTC att fungera. Detta löstes dock med hjälp av databladet och många timmars arbete.

Övriga utmaningar vi stötte på i vår mjukvaruutveckling var av mindre karaktär. Dessa löstes oftast genom datablad eller handledning. Några av dessa mindre utmaningar var hur vi skulle arbeta med interrupts, hur en av de timers som var inbyggda i processorn fungerade och hur man får AD-omvandlingen att fungera.

5.2 Lärdomar

De främsta lärdomarna från detta projekt kan sammanfattas följande punkter:

- Ett bra förarbete och en god design spar mycket tid senare i projektet

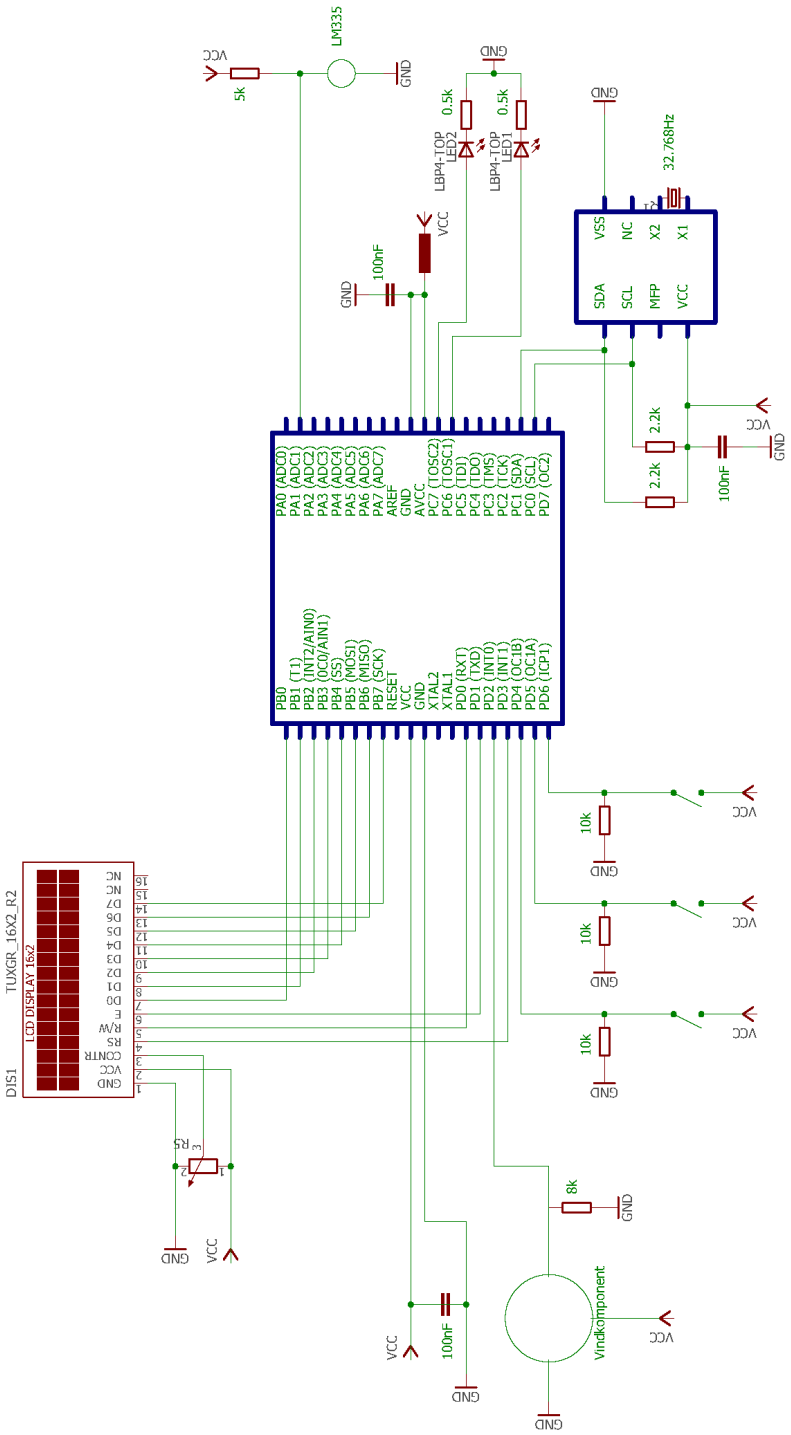
- Genom att etappvis testa sitt system kan man tidigt upptäcka och åtgärda fel
- Kunna sovra och ta fram relevant information ur ett datablad
- Större förståelse för hur hårdvaruutveckling sker i praktiken

5.3 Utvecklingspotential

Med de nyvunna kunskaper gruppen besitter efter utfört projekt fanns grund för en kortare reflektion över vad som skulle kunna förbättras med produkten. Det gruppen kom fram till var att exempelvis skulle följande förbättringar kunna utföras.

- Seglarindikatorn skulle kunna drivas av ett batteri och på så sätt bli mobil.
- Ett back-up batteri till RTC skulle förbättra funktionaliteten så att klockan inte behöver ställas in vid varje uppstart.
- En accelerometer skulle kunna ge seglarindikatorn funktionen att varna om att båten bör tas mot land på grund av för höga vågor.
- En summer eller enklare högtalare skulle kunna användas för att larma vid mindre önskvärda förhållanden.

Bilaga 1: Kopplingsschema



Bilaga 2: Källkod

```
/*
 * Seglarstation.c
 *
 * Created: 2019-04-17 14:56:58
 * Author : Jo0287ha-s
 */

#define F_CPU 8000000UL
#define F_SCL 100000UL

#include <avr/io.h>
#include "led_btn.h"
#include "display.h"
#include "tempSensor.h"
#include "rtcNew.h"
#include "Timer.h"
#include "WindSensor.h"
#include <avr/interrupt.h>


float temp;
float windRound;
float wind;

float avrTemp[5];
float avrWind[5];
int index;
float avrT;
float avrW;

float calcAvrWind();
float calcAvrTemp();
void system_init();
void insertValue(float value, float list[]);

int main(void)
{
system_init();
sei();
```

```
    while (1) {
if (btn1_read()){
display_clear();
setRow(1);
temp= get_temp();
printTemp(temp);
writeText("  ");
printWind(wind);
setRow(0);
printCurrentTime( getYear(), getMonth() , getDay() , getHour() ,getMin());
}
if(btn2_read()){
display_clear();
avrT=calcAvrTemp();
avrW=calcAvrWind();
writeText("AvrT: ");
printTemp(calcAvrTemp());
setRow(2);
writeText("AvrW: ");
printWind(avrW);

    }
if(btn3_read()){
display_clear();
avrT=calcAvrTemp();
avrW=calcAvrWind();
if(avrT<4){
writeText("SEGLA INTE!");
setRow(2);
writeText("Kallt! ");
printTemp(avrT);
led_set_yellow();
led_clear_green();
}

else if(avrT>26){
writeText("SEGLA INTE!");
setRow(2);
writeText("Varmt! Sola ist");
led_set_yellow();
```

```
led_clear_green();
}
else if(avrW<2){
writeText("SEGLA EJ! Ingen");
setRow(2);
writeText("vind, ");
printWind(avrW);
led_set_yellow();
led_clear_green();
}
else if(avrW>7){
writeText("SEGLA EJ! Mycket");
setRow(2);
writeText("vind, ");
printWind(avrW);
led_set_yellow();
led_clear_green();
}
else{
writeText("SEGLARDAG! Ta");
setRow(2);
writeText("solskydd! :)");
led_set_green();
led_clear_yellow();
}
}
}
}

void system_init(){
  display_init();
  displayOn();
  init_temp();
  init_WindSensor();
  Start_Timer();
  led_init();
  btn_init();
  set_RTC(1,9,0,5,1,5,1,2,0,0,0,0);
  windRound=0;
  index=0;
}
```

```
ISR(INT0_vect){
windRound++;
}

ISR(TIMER1_OVF_vect){
  wind=windRound*0.314*256*65535/F_CPU;
windRound=0;
index++;
insertValue(wind,avrWind);
insertValue(get_temp(),avrTemp);
}

void insertValue(float value, float list[]){
if(index==5){
index=0;
}
list[index]=value;
}

float calcAvrTemp(){
if(sizeof(avrTemp)){
float sum=0;
for(int i=0; i <5 ; i++){
sum+=avrTemp[i];
}
float res=sum/5;
return res;
}
else{
return 0;

}
}

float calcAvrWind(){
if(sizeof(avrWind)){
float sum=0;
for(int i=0;i<5;i++){
sum+=avrWind[i];
}
```

```
float res=sum/5;
return res;
}
else{
return 0;
}
}
```