Digitala Projekt (EITF11)
Hemlarm

Karl Nordenstorm, I-12, gem12kno

2014-05-16

Handledare: Bertil Lindvall

Sammanfattning

Den här rapporten beskriver arbetet bakom hur en prototyp för ett hemlarm har utvecklats inom ramen för kursen Digitala Projekt. I stora drag innefattade arbetet val av komponenter, ritning av kopplingsschema, hopmontering av prototyp, framställning av mjukvara samt testning. Eleverna på kursen hade viss förkunskap inom programmering, men liten eller ingen erfarenhet av att med hjälp av programmering hantera elektrotekniska komponenter. Den största utmaningen under projektet var att tillgodogöra sig nog med information för att kunna hantera komponenterna. Vid denna rapports skrivande är prototypen ännu inte färdig, men utvecklingen närmar sig slutskedet. En viktig lärdom från projektet är att det är möjligt att med relativt små förkunskaper tillverka en fungerande elektroteknisk produkt.

Innehållsförteckning

Inledning

Kursen Digitala Projekt har bland annat som mål att eleverna ska lära sig att realisera digitala system av låg och medelhög komplexitet. Just *realisera* är ett ord som beskriver vad den här kursen gått ut på. Man har haft som uppgift att konstruera en fungerande prototyp av en elektronisk produkt.

I detta projekt ska den färdiga prototypen efterlikna ett hemlarm, med en kretssensor och en IR-sensor. I ett verkligt larm skulle kretssensorn monteras mellan t.ex. ett fönster och fönsterkarmen, så att kretsen blev bruten om fönstret öppnades.

Man kan kommunicera med larmet m.h.a. en knappsats. Med denna kan man sätta på larmet, och med hjälp av en säkerhetskod kan man stänga av det. Larmet har en grön och en röd LED. Den röda lyser för att visa när larmet har gått, den gröna lyser för att visa att larmet är påslaget.

Kravspecifikation

Det första som gjordes i projektet var att fastställe en kravspecifikation. Specifikationen är långt ifrån heltäckande.

- En grön ledlampa ska alltid lysa när larmanläggningen är aktiverad, och den ska inte lysa när larmet är avaktiverat.
- Via en display ska larmanläggningen kunna ge användaren några meddelanden, så som att den informerar om att den slagits på.
- När larmanläggningen avaktiveras så ska larmet stängas av
- Larmanläggningen ska kunna avaktiveras genom att man slår in en säkerhetskod.
- Larmanläggningen ska kunna aktiveras via knappsatsen.
- Larmet ska utlösas i fall en värmedetektor registrerar en hastig temperaturförändring.
- Larmet ska utlösas om en kretsdetektors krets blir bruten.
- När larmet går ska en röd LED-lampa slås på.

Hårdvara

Processor

Valet av processor var en *ATMEGA16* . Valet fattades baserat på handledarens rekommendation. Via en JTAG överförs mjukvaran till processorn.

Display

Större delen av systemets kommunikation med användaren sker via en display. Det är via denna som användaren ges instruktioner. Displayen som används är en alfanumerisk *Dot-Matrix LCD Units*.

Knappsats

Användaren kommunicerar med systemet via en knappsats.

Encoder

Encodern känner av när användaren trycker på en av knappsatsens knappar, och meddelar CPUn vilken knapp som blev nedtryckt. Encodern som används i projektet är en *MM54C922/MM74C922 16-Key Encoder*.

IR-sensor

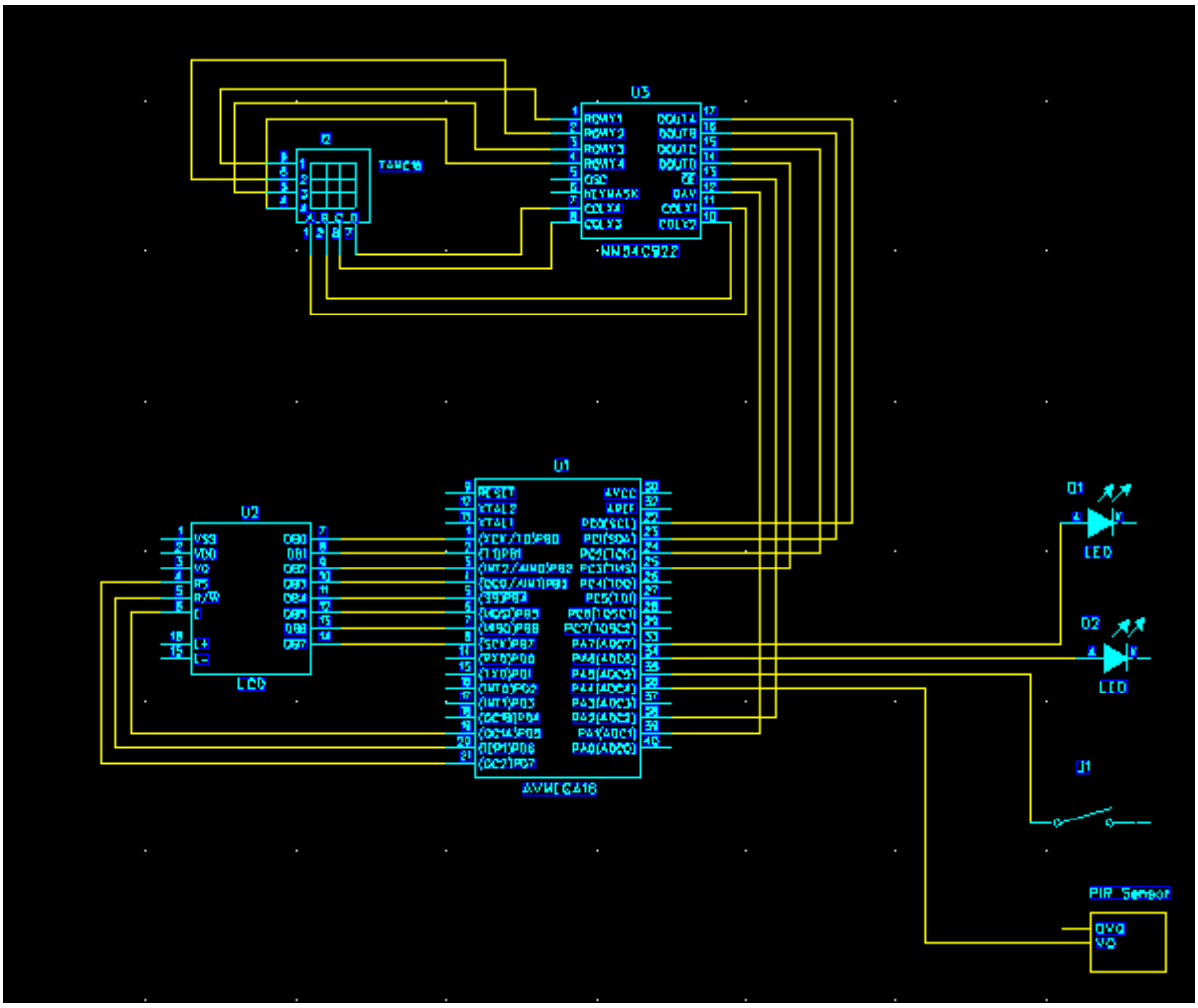
En IR-sensor används för att övervaka om någon rör sig framför sensorn. Då sensorn känner en temperaturförändring sänder den en signal till CPUn. IR-sensorn som används är en *PIR Sensor (#555-28027)*. Valet av IR-sensor baserades på en rekommendation från handledaren.

Övriga komponenter

Utöver de ovan nämnda komponenterna, så användes en rad mindre avancerade komponenter. Däribland två LED-lampor vilka används för att meddela användaren vilket tillstånd larmet befinner sig i. Den kretssensor som används är en krets, dvs. en kabel som bryts manuellt. Kondensatorer och motstånd används för att hantera de övriga komponenterna. Valet av dessa utgick från övriga komponenters behov.

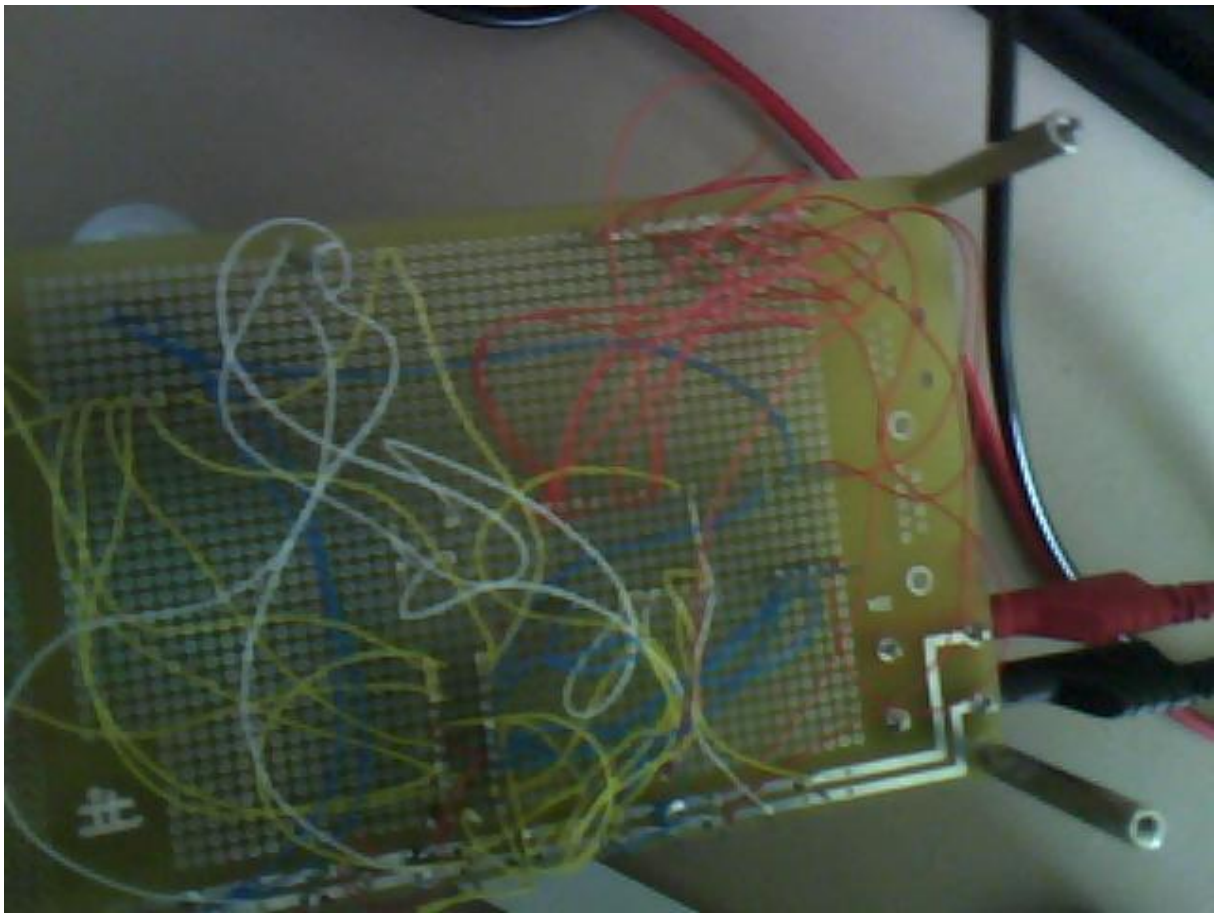
Kopplingsschema

Kopplingsschemat i figur 1 fastställdes inför monteringen. I efterhand gjordes två förändringar kabeln från processorn till *Data Available* på knappsatsen respektive kabeln från processorn till pinnen för utsignalen från IR-sensorn, flyttades till PD2, respektive PD3 på processorn. Detta för att kunna använda interrupts.



Figur 1 Kopplingsschema

Figur 2 Bild av prototyp



Figur 3 Bild av prototypens undersida

Mjukvara

I det här projektet gjordes en preliminär variant av mjukvaran mycket tidigt, som en del av processen att läsa in sig i hur komponenterna fungerar. Allt eftersom så har mjukvaran ändrats, men det har aldrig gjorts någon omskrivning från scratch. På grund av den här arbetsproessen så har den ingen genomtänkt design.

Mjukvaran är i skrivande stund helt färdig. Den senaste varianten av källkoden finns i Bilaga A.

Problem under arbetet

Det största problemet har varit att förstå hur komponenterna fungerar. Man hade datablad över produkterna till sitt förfogande, men många av dem är för tunglästa och för långa för att man ska kunna ta till sig texten direkt. Jag handskades med det här problemet genom att hitta exempel på internet över hur andra har hanterat komponenterna i programmering. Efter att ha sett praktiska exempel blev det lättare att identifiera vilka delar av koden som var viktiga att läsa.

Jag hade en något olämplig arbetsprocess, som började med att jag hemifrån skrev kod och läste på om komponenterna. Jag borde istället ha arbetat med en komponent i taget, och gå igenom alla stegen från påläsningen till testning. Med en sådan process hade det varit lättare att hålla koll på

En rad praktiska problem uppstod, däribland att en display inte fungerade och att jag under monteringen lät kablarna bli så långa att de ofta trasslar ihop sig (Se figur 3). Vid de ändringar som gjordes efter den huvudsakliga monteringen orsakade detta att det var svårt att komma åt med lödkolv på kretskortet. Ett ytterligare problem detta orsakade var att det var arbetsamt att kontrolera att kablarna var kopplade korrekt.

Slutsats

I det här projektet har jag hittills lyckats komma en bra bit mot en fungerande prototyp. Under arbetets gång så har det varit utmanande att hitta all relevant information för hur komponenterna ska hanteras, bland den uppsjö av information som finns tillgänglig. Jag har fått praktiska lärdommar som att kablar bör hållas så korta som möjligt, för att undvika att de trasslar ihop sig.

Bilaga A Preliminär kod

```
/*
* LarmArbetsDok.c
*
* Created: 2014-05-13 10:17:48
* Author: digpi15
*/
#include <avr/io.h>
#include <avr/interrupt.h> //
#include <util/delay.h>
#include <avr/portpins.h> // ?? Kommer inte ihåg vad den gör
//Attributes
int monitorMode = 0; // 0 = not active, 1 = active
int soundAlarm = 0; // 0 = not active, 1 = active
char input; // används som input
char inputCode[4]; //
char deactivationCode[4];
int cursor; // Den cursor som avgör var I inputCode man skriver
char inputToChar(int);
/// Metoder
//void //TEMPwriteString(char inputString[]); // Jag vet inte varför
man måste deklarera
allt på det här sättet
void LCDSetup();
void CPUSetup();
void delay(int);
void main(void)
{
// Assigning values
delay(200);
cursor = 1; // Den cursor som avgör var I inputCode man skriver
inputCode[1] = 'x';
inputCode[2] = 'x';
inputCode[3] = 'x';
```

```
inputCode[4] = 'x';
deactivationCode[1] = '0';
deactivationCode[2] = '4';
deactivationCode[3] = 'A';
deactivationCode[4] = 'B';
CPUSetup();
LCDSetup();
// Skriv "Click 0 to activate"
writeChar(0b01000011);//c
writeChar(0b01001100);//l
writeChar(0b01001001);//i
writeChar(0b01000011);//c
writeChar(0b01001011);//k
writeChar(0b00100000);//_ k
writeChar(0b00110000);//0
writeChar(0b00100000);//_ _
writeChar(0b01010100);//t _
writeChar(0b01011111);//o 1
writeChar(0b00100000);//_ 3
writeChar(0b00110001);//a t
writeChar(0b00110011);//c i
writeChar(0b01010100);//t v
writeChar(0b01001001);//i
writeChar(0b01010110);//v
writeChar(0b00110001);//a
writeChar(0b01010100);//t
writeChar(0b00000101);//e
while(1){
}
}
/*while(1)
{
;
if(input = '1')
{
monitorMode = 1;
//TEMPwriteString("Monitor Mode activated");
}
// Loopen där övervakning sker, och i vilken larmet kan
avaktiveras
while (monitorMode)
{
while (soundAlarm)
{ // Den här loopen stängs via ett interrupt
setPin("LED_RED", 1);
//TEMPwriteString("Alert");
}
}
}
```

```
}*/
//-----------------------------------------------------------------
--------------------
---
//CPUSetup Jag vill säga till portarna vad de är till för på något
sätt
//-----------------------------------------------------------------
--------------------
---
void CPUSetup(){
DDRA = 0xC5;//11000100; // OK
DDRB = 0xFF; // 11111111 // OK
DDRC = 0x00; // OK
DDRD = 0xE0; //11100000; // OK
MCUCR |= 1<<ISC01 | 1<<ISC00 | 0<<ISC11 | 1<<ISC10;
//http://www.avr-tutorials.com/interrupts/avr-external-interrupt-
cprogramming
GICR |= 1<<INT0 |1<<INT1;
sei();
}
//-----------------------------------------------------------------
--------------------
---------------
// LCDSetup // Se sida 15 LCD
KLAR
//-----------------------------------------------------------------
--------------------
---------------
void LCDSetup()
{
PORTD = 0x20;
//e 1
PORTB = 0x38;
//Function set
PORTD = 0x00;
//E 0
PORTD = 0x20;
//E 1
delay(10);
//Wait
PORTD = 0x00;
//E 0
PORTD = 0x20;
//E 1
delay(10);
PORTD = 0x00;
//e 0
PORTD = 0x20;
//e 1
```

```
delay(10);
PORTD = 0x00;
//e 0
PORTD = 0x20;
//e 1
delay(10);
PORTD = 0x08; //Display off
PORTD = 0x00;
//e 0
PORTD = 0x20;
//e 1
delay(1);
PORTB = 0x01; //Display clear
PORTD = 0x00;
//e 0
PORTD = 0x20;
//e 1
delay(4);
PORTB = 0x06; //Entry mode set
PORTD = 0x00;
//e 0
PORTD = 0x20;
//e 1
delay(1);
PORTB = 0x0D; //Display on, curser off,
blink on
PORTD = 0x00;
//e 0
PORTD = 0x20;
//e 1
delay(1);
}
//---------------------------------------------------------------------
----------------------
---------------
//LCDCursor Denna hanterar cursorn. Flyttar den till 0
KLAR
//---------------------------------------------------------------------
----------------------
---------------
void LCDCursor()
{
screenMode(0x02);//0000000010); // Sida 17 Sista tecknet är dont
care
}
//---------------------------------------------------------------------
----------------------
---------------
//Whipe the screen // sida 16 LCD
```

```
KLAR
//-----------------------------------------------------------------
---------------------
---------------
void whipeScreen()
{
screenMode (0x01);//0000000001); // Sida 17
}
void writeChar(char input){ // Måste jag inte deklarera vad för
vektor det är?
/// Write char see CG RAM/DD RAM Data Write page 19
PORTD = 0xA0;//setPin("LCD_RS", 1);setPin("LCD_R/W", 0);
setPin("LCD_E", 1);
PORTB = input; // Notera input ska vara i HEXA
delay(10);
PORTD = 0x00;//setPin("LCD_E", 0); Skulle kunna vara 0x80
//TEMP//TEMPdelay_ms(100);
}
//-----------------------------------------------------------------
---------------------
---------------
// ScreenMode Ändrar LCDns läge KLAR
//-----------------------------------------------------------------
---------------------
---------------
void screenMode(char mode) {
PORTD = 0x20;//setPin("LCD_R/W", 0); setPin("LCD_RS", 0);
setPin("LCD_E", 1);
PORTB = mode; // Skickar informationen om läget på port B
PORTD = 0x00; //setPin("LCD_E", 0);
delay(100);
PORTD = 0x20;//setPin("LCD_E", 1);
}
//-----------------------------------------------------------------
---------------------
---------------
// Lyssna efter input från IR SENSOR KLAR
//-----------------------------------------------------------------
-------------
ISR(INT1_vect) // Värme
{
if (monitorMode = 1){
soundLarm = 1;
}
}
//-----------------------------------------------------------------
---------------------
---------------
```

```
// Lyssna efter input från keypad Den här metoden hanterar
kodinsättningar
//-----------------------------------------------------------------
--------------------
--------------
ISR(INT0_vect)
{
unsigned short int inputs = PINC; //
char c = inputToChar(inputs); //
if(c == '0')
{ // Turn on monitorMode if it was off
if (! monitorMode)
{
monitorMode = 1;
}
}
else if (c != '0')
{
if (monitorMode)
{
inputCode[cursor] ='c';
writeChar('c');
//LCDCursor('+');
cursor ++;
if (cursor = 5)
{ // Kolla om koden är rätt
if (inputCode == deactivationCode)
{
monitorMode = 0;
soundAlarm = 0; // Här
stängs soundAlarm av.
//TEMPwriteString("Correct
Code");
}
else {
cursor = 1;
//TEMPwriteString("Incorrect Code");
//TEMP//TEMPdelay_ms(1000);
whipeScreen();
}
}
}
}
}
//-----------------------------------------------------------------
--------------------
--------------
//Interpret input from keypad KLAR
```

```
//-----------------------------------------------------------------
--------------------
--------------
char inputToChar(int input){ // Här så har jag redan fått
maskinkoden som input. I Main
Loop så ska jag koda //code. Input sker till PINC0 ,1 , 2 och 3.
// Ska jag skriva binärt eller decimalt? Se sida 4 MMC för tabellen
switch(input) {
case 0xC0: //1100 0000
return '0';
break;
case 0xC1: //1100 0001
return '4';
break;
case 0xC3: //0010
return 'A';
break; // Notera hoppet
case 0xC2: //0100
return 'B';
break;
}
return 'n'; // N = No input. Denna har ingen riktig funktion
}
//---------
// Delay KLAR
//---------
void delay(int i){
int k;
for (k=0;k<i;k++)
{
_delay_loop_2;
}
}
```

Referenser

Datablad

Databladen är tillgängliga via kursens hemsida.

- 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash
- 555-28027-PIR-Sensor-Prodcut-Doc-v2.2
- MM54C922 16-Key Encoder MM54C923 20-Key Encoder
- Dot-Matrix LCD Units (with built-in controllers) User's Manual

Internetkällor

AVR Tutorials. http://www.avr-tutorials.com

Stack Overflow. http://stackoverflow.com/

Wikipedia. *C (Programming language)*
http://en.wikipedia.org/wiki/C_%28programming_language%29