Digitala Projekt(EITF40) - Larm

Handledare: Bertil Lindvall

Kristoffer Sätermark, dt08ks6

Magnus Johansson, dt08mj9

Innehåll

Sammanfattning

I detta projekt har vi byggt ett huslarm med en AVR ATMega16 mikroprocessor som central kontrollenhet. Larmet använder IR-sensorer och magnetkontakter för att detektera eventuella inbrott. I denna rapport kommer vi att presentera våra komponenter och vårt utförande av bygget.

1 Introduktion

Vi inledde vårt arbete med att hitta datablad till alla komponenterna får att se hur de fungerade och vilka ben på varje komponent vi var intresserade av. Sedan placerade vi ut komponenterna på kopplingsplattan och upptäckte snabbt att det skulle bli trångt med utrymme. Vi fick testa oss fram tills vi hittade en komponentplacering där allt fick plats och avståndet mellan relevanta ben blev kort så att kabeldragningen i sin tur också blev kort.

2 Kravspec

Det projekt vi har tänkt oss är att skapa ett larmsystem för övervakning av exempelvis en villa. Systemet är tänkt att användas och klara av enligt följande kravspecifikation:

- Larmet ska kunna hantera 1-8 digitala magnetkontakter för övervakning av tex fönster och dörrar.
- Larmet ska kunna hantera 1-2 analoga IR-sensorer för övervakning av temperatur förändringar på 5°C.
- Larmet ska kunna hantera 1-2 analoga IR-sensorer för övervakning rörelser.
- Systemet aktiveras/avaktiveras av en 4-siffrig kod knappas in på en numerisk knappsats.
- Om någon av sensorerna ger utslag larmet ska 30 sekunder ges för att skriva in en 4-siffrig kod på knappsatsen innan larmet och sirénen aktiveras.
- En sirén ska aktiveras och ljuda då larmet utlösts fram tills att användaren skrivit in en 4-siffrig kod.
- Systemet ska på en display kunna visa den tidpunkt larmet utlösts.
- Systemet ska hantera 3 lysdioder som visar systemets status för användaren.
  - En diod visar om systemet är aktiverat eller avaktiverat.
  - En diod visar om kod måste skrivas in för att inte utlösa larmet.
  - En diod visar om systemet är i test läge.
  - Är larmet utlöst blinkar samtliga dioder.
- Systemet ska kunna skicka en logg över vilken sensor som utlöst larmet och vid vilken tidpunkt till en PC.

3 Hårdvara

3.1 knappsats och decoder

Vi använde en knappstas med 16 knappar uppdelade i 4 rader och 4 kolumner. Istället för att koppla knappsatsen direkt till mikroprocessorn kopplade vi in en knappsatsdekoder. Knappsatsdekodern läser av knappsatsen med en frekvens som bestäms av de två kondensatorer som kopplas till komponeneten. När en knapptryckning registerats signalerar dekodern till processorn att data finns tillgänglig.

3.2 LCD-Display

Vi använder en alfa-numerisk display med 2 rader och 16 tecken per rad. Displayen visar relevant information om larmets tillstånd.

3.3 Processor

Vi använde en AVR ATMega16 som central enhet i vårt larm. Denna processor har en intern klocka på 8 MHz, vi valde dock att använda en extern kristall med en frekvens på 16 MHz. Processorn har 1 Kb inbyggt ramminne samt 16 Kb internt flashminne. För kommunikation med omvärlden finns det 32 I/O pinnar uppdelade på 4 portar namngivna A, B, C och D. Port A har en inbyggd A/D omvandlare för att hantera analoga signaler.

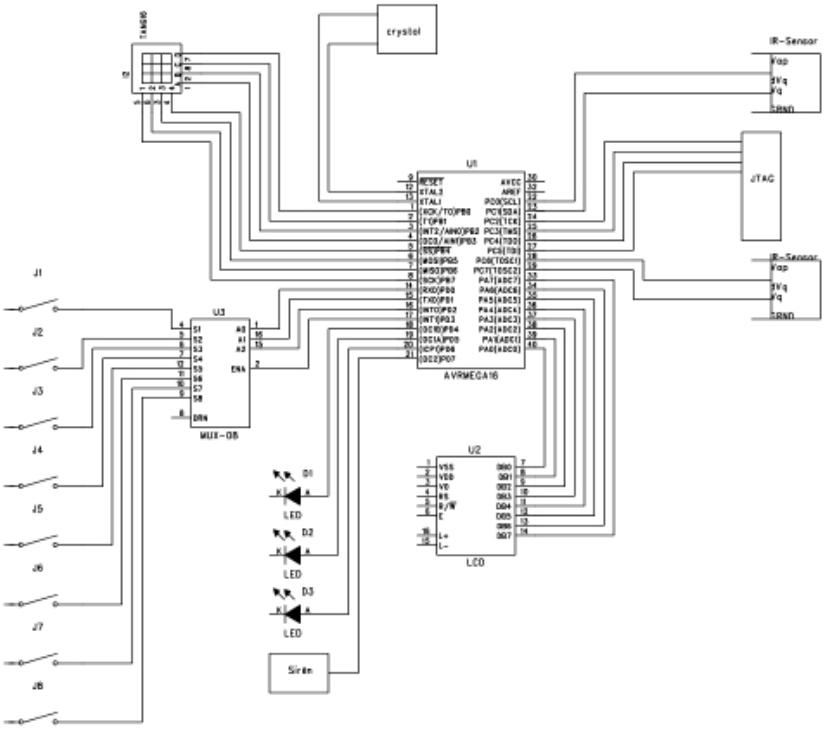
3.4 Magnetkontakter

Magnetkontakerna fungerar som en strömbrytare och leder då kontakterna är slutna.

3.5 IR sensor

IR sensor som mäter temperaturen och ger en analog utsignal mellan 0V och arbetsspänningen. Om något i mätområdet har en temperatur som avviker med mer än 5° från omgivningen ges en hög spänning, om ingenting avviker med mer än 5° ges en utspänning motsvarande halva arbetsspänningen.

4 Blockschema



Figur 1: Blockschema

5 Konstruktion

Vi valde att bygga vårt huslarm med en display för att visa information så som en klocka, vilken sensor som utlösts samt tidpunkt för detta. Tre dioder används för att visa ytterliggare information. Detta inkluderar om larmet är aktivt och om det har utlösts. För att kunna ge input till larmet använde vi oss av en knappstas med totalt 16 knappar. För att förenkla kommunikationen mellan knappsatsen och mikroprocessorn kopplade vi in en knappsatsdekoder.

Vi byggde in stöd för 8 magnetkontakter och 2 IR-sensorer. För att kunna koppla in alla 8 magnetkontakterna till mikroprocessorn använde vi oss av en 8 kanals multiplexer. Detta innebar att vi kunde minska antalet ben dedikerade till magnetkontakterna från 8 till 5. Magnetsensorerna är till för att kopplas på dörrar eller fönster och utlöses om kontakten bryts.

IR-sensorerna upptäcker temperatursignaturer som skiljer sig från omgivningens temperatur med 5°C eller mer. De ger analoga signaler vilket innebar att de bara kunde kopplas in på mikroprocessorns port A, som har en A/D omvandlare inbyggd.

6 Mjukvara

Vi skrev vårt program i C. Programmet är uppbyggt runt avbrott. Vi använder en inbyggd avbrottsfunktion som genererar avbrott med en frekvens på 61 Hz. Detta avbrott används för att kontrollera larmets klocka och timers. I programmets mainmetod finns den centrala logiken. Denna metod håller reda på i vilket tillstånd larmet befinner sig och exekverar lämpliga metoder.

Metoden sensors() läser av alla sensorerna och sätter en flagga om någon sensor utlösts. Denna metod exekveras med jämna mellanrum om larmet är i aktivt läge.

Ett externt avbrott genereras då en knapp trycks in på knappsatsen och läses in till programmet i metoden för detta avbrott.

Det interna avbrottet styr klockan i systemet för att kunna räkna sekunder.

7 Problem och slutsats

Vi byggde vårt larm efter det kretsschema vi skapat men upptäckte snabbt att vi gjort några missar, ben som var felkopplade och komponenter som saknades. Utöver detta gick konstruktionen och kopplingen av hårdvaran smidigt. I programmeringsfasen upptäckte vi en del problem, särskilt timingrelaterade.

Bland annat tog det ganska mycket tid att initiera displayen på rätt sätt och sedan skriva till den, detta berodde mycket på konstruktionsfel när blockschemat ritades.

Kristallens ben var till en början anslutna genom virning, detta visade sig efter en lång tids felsökning ge för dålig kontakt som gav upphov till mycket märkligt beteende hos processorn. När problemet var upptäckt löstes det dock enkelt genom att enbart löda fast benen istället för att vira.

Utöver dessa problem gick projektet bra och vi har lärt oss mycket om framförallt hårdvaran, men även programmeringsdelen. Det var för oss första gången vi dels konstruerat, dels byggt en prototyp av detta slag så många misstag kan läggas på kontot för nybörjare. Det till trots fick vi vårat larm färdigt till slut.

8 C-kod

```
#include <avr/io.h>
#define F_CPU 16000000UL  // 16 MHz
#include <util/delay.h>
#include <avr/interrupt.h>

char kod[8];
char x[4] = { '0', '1', '2', '3' } ;
int code_sec = 0; // sekunder sen sensor utlosts
int triggered = 0;//sensor utlost

int code_input = 0; // sekunder sen kod borjade skrivas in
int code_entered = 0; // kod inlasning paborjad

int vec1[9];

char larmOn = 0;                //larmet har utlosts
char larm_activated = 0;      // larmet aktiverat
int sensor = 0;


void set_pin(char port, char pin, char state);

volatile uint8_t count;
int checkTemp=0;
int sec=0, min=0, h=0;
int larm_sec, larm_min, larm_h;
int selection=0;
int clockReset=0;
int check=0;

uint16_t adc_result = 0;
int ir_sec = 0;

void write_code(char number){
     if(kod[4] == 0){
          kod[0] = number;
          kod[4] = 1;
     } else if(kod[5] == 0){
          kod[1] = number;
          kod[5] = 1;
     } else if(kod[6] == 0){
          kod[2] = number;
          kod[6] = 1;
     }else{
          kod[3] = number;
          if(kod[0] == x[0] && kod[1] == x[1] && kod[2] == x[2] && kod[3] == x[3]){
               larm_activated ^= 1;
               larmOn = 0;
```

```
          }
     }
}

void write_cmd(char val){
     PORTB=val;
     _delay_ms(5);
     set_pin('D', PD5,0);      //RW
     set_pin('D', PD6, 0);     //RS
     _delay_ms(5);
     set_pin('D',PD7, 1);     //Enable
     _delay_ms(5);
     set_pin('D',PD7, 0);     //Enable
}


void disp_writeCh(char val){
     PORTB=val;
     set_pin('D', PD5,0);     //RW
     set_pin('D', PD6, 1);     //RS
     set_pin('D',PD7, 1);     //Enable
     _delay_ms(5);
     set_pin('D',PD7, 0);     //Enable
}

void disp_clear() {
     write_cmd(0x01); //clear disp
     write_cmd(0x38); //func set
}

void disp_init() {
     disp_clear();
     write_cmd(0x0f); //disp on
     write_cmd(0x06); //entry mode set
}

void disp_home(){
     write_cmd(0x03);
}

void disp_secondLine() {
     write_cmd(0xC0);
}

void display_num(int nu){
     if(nu<10){
          disp_writeCh('0');
          itoa(nu,vec1,10);
          disp_writeCh(vec1[0]);
     }
```

```
     else {
          int num1=nu/10;
          itoa(num1,vec1,10);
          disp_writeCh(vec1[0]);
          int num2=nu%10;
          num1=itoa(num2,vec1,10);
          disp_writeCh(vec1[0]);
     }
}

void disp_time(int hour, int minute, int second){
     disp_clear();
     disp_home();
     disp_writeCh('T');
     disp_writeCh('i');
     disp_writeCh('m');
     disp_writeCh('e');
     disp_writeCh(':');
     disp_writeCh(' ');
     disp_writeCh(' ');
     disp_writeCh(' ');
     disp_secondLine();

     display_num(hour);
     disp_writeCh(':');
     display_num(minute);
     disp_writeCh(':');
     display_num(second);
}

void set_pin(char port, char pin, char state){
     char set = 1 << pin;
     if(port == 'A'){
          set &= PORTA;
          if(set && !state){ //andra fran 1 -> 0
               PORTA ^= set;
          }
          if(set == 0 && state){ //andra fran 0 -> 1
               set = 1 << pin;
               PORTA ^= set;
          }
     }
     else if(port == 'B'){
          set &= PORTB;
          if(set && !state){ //andra fran 1 -> 0
               PORTB ^= set;
          }
          if(set == 0 && state){ //andra fran 0 -> 1
               set = 1 << pin;
               PORTB ^= set;
```

```
        }
    }
    else if(port == 'C'){
        set &= PORTC;
        if(set && !state){ //andra fran 1 -> 0
            PORTC ^= set;
        }
        if(set == 0 && state){ //andra fran 0 -> 1
            set = 1 << pin;
            PORTC ^= set;
        }
    }
    else if(port == 'D'){
        set &= PORTD;
        if(set && !state){ //andra fran 1 -> 0
            PORTD ^= set;
        }
        if(set == 0 && state){ //andra fran 0 -> 1
            set = 1 << pin;
            PORTD ^= set;
        }
    }


}



ISR(TIMER0_OVF_vect) {  //kors vid varje interrupt
    count++;
    //En sekund har gatt
    if(count==61){
        //oka sekund
        sec++;
        if(adc_result > 550){
            ir_sec++;
        }else{
            ir_sec = 0;
        }

        if(ir_sec >= 5){
            triggered = 1;
            sensor = 9;
            set_pin('A', PA1, 1);
        }
        if(triggered){
            code_sec++;
            if(code_sec == 10){
                larmOn=1;
```

```
                    larm_sec = sec;
                    larm_min = min;
                    larm_h = h;
               }
          }
          if (sec == 60) {
               sec = 0;
               min++;
               if (min == 60) {
                    min = 0;
                    h++;
                    if (h == 24) {
                         h = 0;
                    }
               }
          }
         count=0;
     }
}

ISR(INT0_vect)
{
     cli();
     PORTB = 0x00;
     set_pin('D', PD7, 0);
     DDRB = 0x00;
     set_pin('D', PD3, 0);
     _delay_ms(300);
     char number = PINB;

     _delay_ms(30);
     set_pin('D', PD3, 1);
     DDRB = 0xFF;
     _delay_ms(30);

     disp_clear();
     disp_home();

     number &= 0x0F;

     if(number == 0x00){
          write_code('0');
          disp_writeCh('0');
     }
     else if(number == 0x01){
          write_code('8');
          disp_writeCh('8');
     }
     else if(number == 0x02){
          write_code('4');
```

```
     disp_writeCh('4');
}
else if(number == 0x03){
     disp_writeCh('C');
     kod[4] = 0;
     kod[5] = 0;
     kod[6] = 0;
     kod[7] = 0;
}
else if(number == 0x04){
     write_code('2');
     disp_writeCh('2');
}
else if(number == 0x05){
     disp_writeCh('A');
}
else if(number == 0x06){
     write_code('6');
     disp_writeCh('6');
}
else if(number == 0x07){
     disp_writeCh('E');
}
else if(number == 0x08){
     write_code('1');
     disp_writeCh('1');
}
else if(number == 0x09){
     write_code('9');
     disp_writeCh('9');
}
else if(number == 0x0A){
     write_code('5');
     disp_writeCh('5');
}
else if(number == 0x0B){
     disp_writeCh('D');
}
else if(number == 0x0C){
     write_code('3');
     disp_writeCh('3');
}
else if(number == 0x0D){
     disp_writeCh('B');
}
else if(number == 0x0E){
     write_code('7');
     disp_writeCh('7');
}
else if(number == 0x0F){
```

```
            disp_writeCh('F');
        }

        disp_writeCh(' ');

        disp_writeCh(kod[0]);
        disp_writeCh(kod[1]);
        disp_writeCh(kod[2]);
        disp_writeCh(kod[3]);

        disp_secondLine();

        disp_writeCh(x[0]);
        disp_writeCh(x[1]);
        disp_writeCh(x[2]);
        disp_writeCh(x[3]);


        code_entered = 1;
        sei();
}

void interupt_init(){
        ADMUX=(1<<REFS0);                              // For Aref=AVcc;
        ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0); //Rrescalar div factor =128

        TCCR0|=(1<<CS02)|(1<<CS00);          // Prescaler = FCPU/1024
        TIMSK|=(1<<TOIE0);                    //Overflow Interrupt Enable

        GICR = 1<<INT0;                            // Enable INT0
        MCUCR = 1<<ISC01 | 1<<ISC00;     // Trigger INT0 on rising edge

           //Init raknare
        TCNT0=0;
        count=0;

        sei();
}

uint16_t ReadADC(uint8_t ch)
{
   //Select ADC Channel ch must be 0-7
   ch=ch&0b00000111;
   ADMUX|=ch;

   //Start Single conversion
   ADCSRA|=(1<<ADSC);

   //Wait for conversion to complete
   while(!(ADCSRA & (1<<ADIF)));
```

```
   //Clear ADIF by writing one to it
   ADCSRA|=(1<<ADIF);

   return(ADC);
}

void sensors(){
          char sensor1, sensor2;

          set_pin('C', PC0,0);
          set_pin('C', PC1,0);
          set_pin('C', PC6,0);

          sensor1 = PIND;
          sensor1 &= 0x01;

          if(sensor1 == 0 && sensor == 0){
               triggered = 1;
               set_pin('A', PA1, 1);
               sensor = 1;
          }

          set_pin('C', PC0,1);
          set_pin('C', PC1,0);
          set_pin('C', PC6,0);

          sensor2 = PIND;
          sensor2 &= 0x01;

          if(sensor2 == 0 && sensor == 0){
               triggered = 1;
               set_pin('A', PA1, 1);
               sensor = 2;
          }
          adc_result=ReadADC(7);           // Read Analog value from channel-0

}


int main(void)
{
     kod[4] = 0;
     kod[5] = 0;
     kod[6] = 0;
     kod[7] = 0;

     DDRA = 0x7F;
     DDRB = 0xFF;
     DDRC = 0xFF;
```

```
DDRD = 0xFA;

//Enable the multiplexer
set_pin('C', PC7,0);

set_pin('D', PD2,0);
set_pin('D', PD3,1);

interupt_init();

PORTA = 0x00;

disp_init();
disp_home();

while(1){

     if(larmOn == 1 && larm_activated){
          disp_clear();
          disp_home();

          if(sensor < 9){
               disp_writeCh('M');
               disp_writeCh('a');
               disp_writeCh('g');
               disp_writeCh('n');
               disp_writeCh('e');
               disp_writeCh('t');

               disp_writeCh(' ');

               if(sensor == 1){
                    disp_writeCh('1');
               }else if(sensor == 2){
                    disp_writeCh('2');
               }
          }else{
               disp_writeCh('I');
               disp_writeCh('R');
          }
          disp_secondLine();

          display_num(larm_h);
          disp_writeCh(':');
          display_num(larm_min);
          disp_writeCh(':');
          display_num(larm_sec);


          set_pin('A', PA1, 1);
```

```
                set_pin('A', PA2, 1);
                set_pin('A', PA0, 1);

                _delay_ms(50);

                set_pin('A', PA0, 0);
                set_pin('A', PA1, 0);
                set_pin('A', PA2, 0);
            }else if(larm_activated){
                set_pin('A', PA0,1);
                disp_clear();
                disp_home();
                disp_writeCh('L');
                disp_writeCh('a');
                disp_writeCh('r');
                disp_writeCh('m');
                disp_writeCh(' ');
                disp_writeCh('A');
                disp_writeCh('c');
                disp_writeCh('t');

                sensors();

            }else{
                disp_time(h,min,sec);
                set_pin('A', PA0,0);
                set_pin('A', PA1,0);
                set_pin('A', PA2,0);
                larmOn = 0;
                triggered = 0;
                code_sec = 0;
                sensor = 0;
            }
            _delay_ms(500);

        }
        return 0;
}
```