Digitala projekt - Grupp 12
Larmsystem

Handledare: Bertil Lindvall
Författare: Henrik Månsson D04 841125-2714 (d04tb@student.lth.se)
Tobias Lindberg D04 840425-4073(d04tl@student.lth.se)
Rikard Gustafsson D04 850427-3056 (d04rg@student.lth.se)

Inlämnad: 26 februari 2007

Innehåll

1 Inledning

Säkerhet är något som alltid intresserat oss och när vi anmälde oss till digitala projekt så var valet givet, vi ville göra ett larmsystem. Vi var dock inte intresserad av att göra ett vanligt hederligt larmsystem, vi ville ha något extra och valde då att intruducera en unik funktion i form av en knacksekvenskod. Vårat system riktar sig mer åt privatpersoner än kommersiella organisationer då vårat system ej kontaktar en larmcentral.

2 Utrustning

Vårat system består av en enchipsdator som är programmerad i C för att styra ett antal IO-enheter.

2.1 Hårdvara

Här nedan beskrivs komponenterna, för att se hur de är ihopkopplade, se appendix B.

2.1.1 Atmega16

En 8-bitars enchipsdator med bl a. 32 IO-pinnar och en 10-bit 8-kanals A/D-omvandlare. Den är programmerbar via JTAG-interface.

2.1.2 JTAG

JTAG är en modul för att koppla upp chippet mot en dator. Detta görs via USB och gör så man får total kontroll över chippet.

2.1.3 SHARP Dot-Matrix LCD Units

En två raders display med plats för 16 ASCII-tecken per rad. Den har alltså en inbyggd kontroller.

2.1.4 Knappsats

Matrisknappsats på 4x4, fyra pinnar för raderna och fyra pinnar för kolonnerna.

2.1.5 IR-sensormodul Nippon Ceramic

Rörelsesensor med öppen kollektorutgång med en räckvidden på 5 m och avkänningsvinkel på 100◦ horisontellt samt 60◦ vertikalt.

2.1.6 Magnetisk larmkontakt

Dörrsensor som leder när magneten ej är nära sensorn.

2.1.7 Högtalare

En liten enkel högtalare.

2.1.8 Kondensatormikrofonelement

En liten mikrofonenhet som genererar en våg vid ljudvibrationer.

2.2 Mjukvara

Här följer en kort beskrivning om hur mjukvaran är designad. För att se koden, se AlarmSystem.c i appendix C.

Våran mjukvara bygger på en state-struct som beskriver vilket tillstånd programmet befinner sig i. Vi har byggt upp programmet kring tre olika tillstånd; alarmerat, utlöst och avstängt tillstånd.

I alarmerat tillstånd så lyssnar vi på dörrsensorn, mikrofonen och rörelsedetektorn. Om dörrsensorn eller rörelsedetektorn blir utlöst sätts systemet i utlöst tillstånd. Upptäcker mikrofonen rätt knacksekvens så återgår systemet i avstängt tillstånd.

I utlöst tillstånd lyssnar vi på knappsatsen för inskrivning av kod. Man har obegränsat antal försök på sig att skriva koden som stänger av larmet. Då rätt kod matats in så sätts systemet i avstängt tillstånd.

I avstängt tillstånd lyssnar vi på knappsatsen för att kunna aktivera larmet, ändra känslighet på rörelsedetektorn, byta kod samt ändra knacksekvensen. När larmet aktiveras sätts systemet i alarmerat tillstånd.

3 Genomförande och resultat

Själva konstruktionen av vårat system skedde i etapper. Först kopplar vi in en ny komponent, sedan skriver vi lite testkod. När det fungerar bra integerar vi den nya funktionaliteten med det övriga systemet.

Vi började med att koppla in displayen för att börja kunna få output. Vi tyckte att detta var viktigt för debugging av övriga komponenter då vi kan skriva ut olika värden på displayen. Detta visade sig särskilt värdefullt vid senare arbete med A/D-omvandling.

Nästa steg var att koppla in knappsatsen. Den fungerar på så sätt att vi loopar en etta på pinnarna till raderna och läser av pinnarna till kolonerna. Då en etta upptäcks så går det att identifiera vilken knapp som blivit nedtryckt.

Efter det så började vi koppla in våra sensorer. Först i kön var magnetsensorn som skulle fungera som vår dörrsensor. Den har en relativt enkel konstruktion som ger en etta ut om det magnetiska fältet kring den bryts. Därefter kopplade vi in rörelsedetektorn som gav en nolla vid rörelse och en etta annars. Möjligheten att ställa in kännsligheten på rörelsedetektorn löstes genom att använda en räknare som räknar upp när det ligger en nolla på ingången och räknar ner vid en etta. Det sker tills räknaren passerar den inställda kännslighetströskeln.

Därefter monterade vi dit högtalaren och genom en enkel funktion genererade vi en fyrkantsvåg med angiven tidslängd och frekvens. Med denna kunde vi generera ljud till knappsatsen och alarmet.

Sedan återstod bara det sista och tyngsta steget, att integrera mikrofonen. När vi hade kopplat in den så fick vi genast problem med att A/D-omvandlingen inte fungerade som vi ville. Efter många timmars problemlösning kom vi dock fram till en lösning och kunde få ut ett fint värde på displayen. Vi testade även att koppla in OP-förstärkare för att få en starkare signal från mikrofonen men stötte där på stora problem. Det slutade med att vi slopade OP-förstärkaren

helt. När mikrofonen fungerade kunde vi implementera den i koden. Lösningen blev att endast använda tre knackningar med olika intervall emellan som gick att ställa in själv. Träffar man dessa intervallen med ± 250 ms låses systemet upp.

4 Brister och kommentarer

Vi har lagt märke till två stora brister med systemet. Den allvarligaste bristen är när systemets larm låter. Problemet är att så länge en knapp är nedtryckt (dvs. man vill knappa in koden för att stänga av larmet) kommer inte larmet att låta förrän knappen släpps igen. Den andra bristen är knacksekvensen. Förutom att mikrofonen reagerar lite dåligt så är knacksekvensen begränsad till specifikt tre knackningar.

5 Appendix

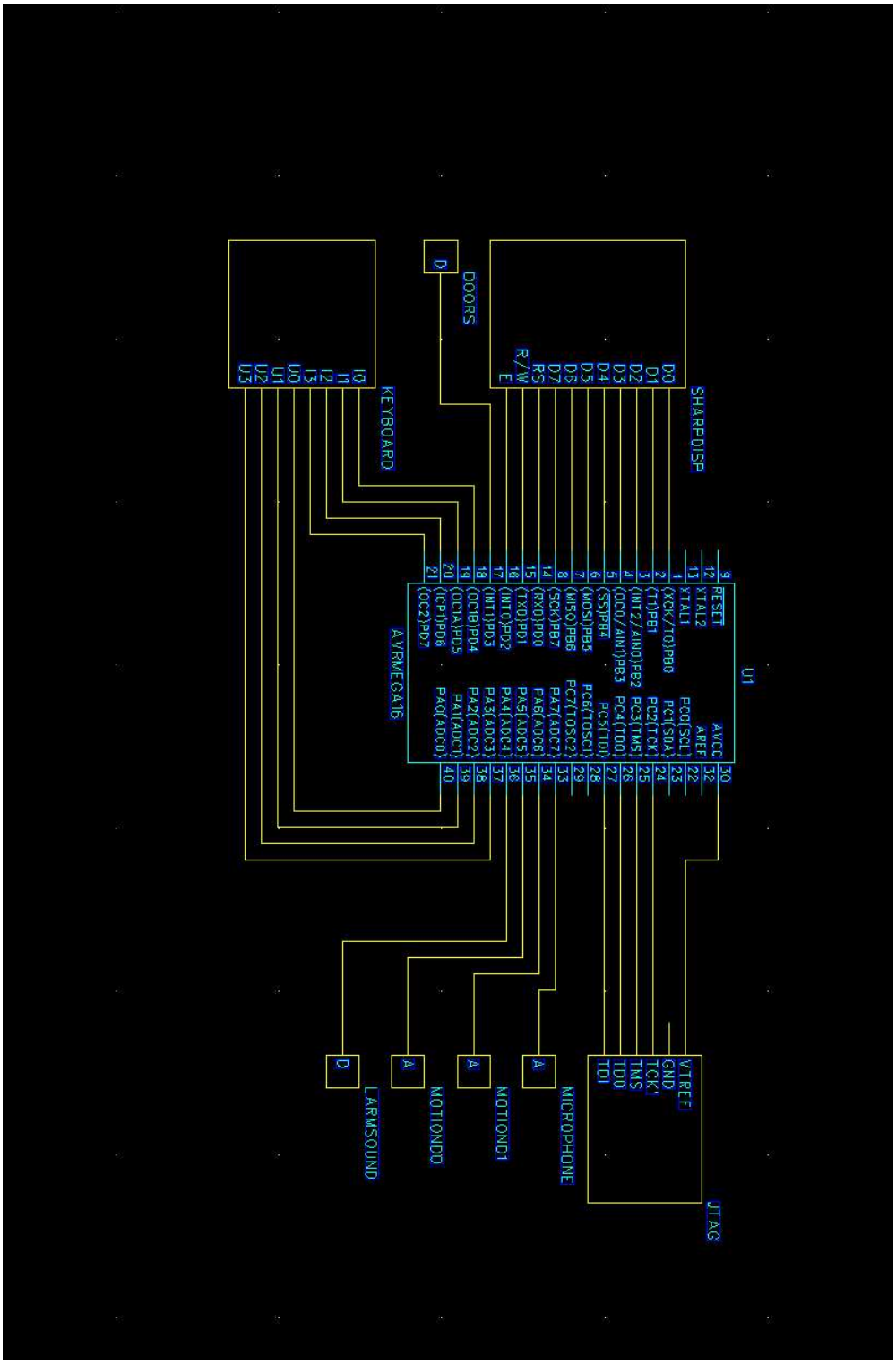
5.1 Appendix A: Kravspecifikation

Kravspecifikation

Vi vill göra ett larmsystem för att upptäcka ljudvariationer (eller rörelsevariationer) i ett rum genom att övervaka rummet med ett antal sensorer utsatta i rummet. Med detta systemet vill vi kunna övervaka dels huvudingången alltså dörren men även andra intrångsmöjlighet som t ex genom ett fönster.

- Systemet skall klara av sex analoga givare (ljudsensorer alt rörelsesensorer) och två digitala för exempelvis dörrar och fönster
- Det skall finnas en knappsats för att både sätta igång samt att stänga av larmet
- Känsligheten för de analoga sensorerna skall kunnas ställas in centralt via knappsatsen
- En liten display som visar status samt eventuella fel
- Koden ska kunna ändras via knappsatsen och displayen
- När larmet går skall en lampa blinka och ett ljud låta
- Om man slår in koden fel tre gånger i rad skall larmet gå
- Rätt kod avbryter larmet när det gått igång
- I dörren sitter en digital givare som reagerar när dörren öppnas och då har man 60 sek på sig att slå in rätt kod
- Knappsats och display skall sitta precis innanför dörren

5.2 Appendix B: Kopplingsschema



5.3 Appendix C: AlarmSystem.c

```
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/eeprom.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>

#define LCD_RS 0
#define LCD_RW 1
#define LCD_E 2

struct state_t{
        bool armed;
        bool triggered;
        char password[5];
        int motion_threshold;
        int zerocounter;
        int micstate;
        int micstate_times[2];
};

typedef struct state_t state_mode;

static int time;
static bool do_alarm;

//LCD_getaddress reads the address counter and busy flag. For the address
only, mask off bit7 of the return
//value.
char LCD_getaddr(void)
{
        //make var for the return value
        char address;
        //PortD is input
        DDRB = 0;
        //RW high, strobe enable
        PORTD |= ((1<<LCD_RW)|(1<<LCD_E));
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        //while E is high, get data from LCD
        address = PINB;
        //reset RW to low, E low (for strobe)
        PORTD &= ~((1<<LCD_RW)|(1<<LCD_E));
        //return address and busy flag
```

```
        return address;
}

//LCD_wait reads the address counter (which contains the busy flag) and
loops until the busy flag is cleared.
void LCD_wait(void)
{
//get address and busy flag
//and loop until busy flag cleared
while((LCD_getaddr() & 0x80) == 0x80){}
}

/*LCD_command works EXaCTLY like LCD_putchar, but takes RS low for accessing
the command reg
see LCD_putchar for details on the code*/
void LCD_command(char command)
{
        DDRB = 0xFF;
        PORTB = command;
        PORTD &= ~((1<<LCD_RS)|(1<<LCD_RW)|(1<<LCD_E));
        PORTD |= (1<<LCD_E);
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        asm volatile ("nop");
        PORTD &= ~(1<<LCD_E);
        DDRB = 0;
}

/*LCD_init initialises the LCD with the following paramters:
8 bit mode, 5*7 font, 2 lines (also for 4 lines)
auto-inc cursor after write and read
cursor and didsplay on, cursor blinking.
*/
void LCD_init(void)
{
        //setup the LCD control signals on PortC
        DDRD |= ((1<<LCD_RS)|(1<<LCD_RW)|(1<<LCD_E));
        PORTD = 0x00;
        //if called right after power-up, we'll have to wait a bit (fine-tune for
faster execution)
        _delay_loop_2(0xFFFF);
        //tell the LCD that it's used in 8-bit mode 3 times, each with a delay
inbetween.
        LCD_command(0x30);
        _delay_loop_2(0xFFFF);
```

```
        LCD_command(0x30);
        _delay_loop_2(0xFFFF);
        LCD_command(0x30);
        _delay_loop_2(0xFFFF);
        //now: 8 bit interface, 5*7 font, 2 lines.
        LCD_command(0x38);
        //wait until command finished
        LCD_wait();
        //display on, cursor on (blinking)
        LCD_command(0x0F);
        LCD_wait();
        //now clear the display, cursor home
        LCD_command(0x01);
        LCD_wait();
        //cursor auto-inc
        LCD_command(0x06);
}

void LCD_putchar(char char_value)
{
        DDRB = 0xFF;
        PORTB = char_value;

        PORTD = 0b00000000;
        PORTD = 0b00000101;
        _delay_loop_2(0xFF);
        PORTD = 0b00000000;

        DDRB = 0;
}

void LCD_putword(char theword[])
{
        int i = 0;
        int j = 0;

        while (theword[i] != '\0'){
                j = i;
                if (theword[i] == '\n'){
                        while (j < 40){
                                LCD_putchar(' ');
                                j++;
                        }
                        i++;
                }
                LCD_putchar(theword[i]);
                i++;
        }
}
```

```
void LCD_clear()
{
        LCD_wait();
        LCD_command(0x01);
        LCD_wait();
}

void check_password(state_mode *state)
{
        int i;
        char passzeros[5];
        passzeros[0] = '\0';

        for(i = 0; i<(4-strlen(state->password)); i++){
                strcat(passzeros, "0");
        }

        strcat(passzeros, state->password);

        strcpy(state->password, passzeros);

}

void speaker_sound(int delay, int soundlen)
{
        int i = 0;
        DDRa = 0xFF;

        while (i < soundlen){
                if (PINa == 0b00000000){
                        PORTa = 0b00010000;
                        _delay_loop_2(delay);
                } else {
                        PORTa = 0b00000000;
                        _delay_loop_2(delay);
                }
                i++;
        }
        DDRa = 0b11011111;
}

char wait_button_pressed()
{
        int portd_save = PORTD;
        DDRa = 0xFF;

        //Vanta tills ingen knapp ar ned tryckt
        PORTD = 0b11110000;
        while(PINa != 0b00000000) {}
```

```
while(1) {

//Vanta 20 sek innan larmet gar
        if (get_timer()>20000 && (get_timer()%1000) ==0  && do_alarm)
                speaker_sound(0xFFF, 100);

        //Check through row 1
        PORTD = 0b00010000;
        if (PINa == 0b00000001) {
                speaker_sound(0xFF, 200);
                return '0';
        } else if (PINa == 0b00000010) {
                return '4';
        } else if (PINa == 0b00000100) {
                return '8';
        } else if (PINa == 0b00001000) {
                return 'L';
        }

        //Check through row 2
        PORTD = 0b00100000;
        if (PINa == 0b00000001) {
                return '1';
        } else if (PINa == 0b00000010) {
                return '5';
        } else if (PINa == 0b00000100) {
                return '9';
        } else if (PINa == 0b00001000) {
                return 'R';
        }

        //Check through row 3
        PORTD = 0b01000000;
        if (PINa == 0b00000001) {
                return '2';
        } else if (PINa == 0b00000010) {
                return '6';
        } else if (PINa == 0b00000100) {
                return 'Y';
        } else if (PINa == 0b00001000) {
                return 'U';
        }

        //Check through row 4
        PORTD = 0b10000000;
        if (PINa == 0b00000001) {
                return '3';
        } else if (PINa == 0b00000010) {
                return '7';
        } else if (PINa == 0b00000100) {
```

```
                        return 'N';
                } else if (PINa == 0b00001000) {
                        return 'D';
                }
        }
        PORTD = portd_save;
        DDRa = 0b11011111;
}

int aDC_read(unsigned char channel)
{
        int value;

        aDMUX = 0x60 | channel;
        aDCSRa=aDCSRa|0x80; //enable adc
        aDCSRa=aDCSRa|0x40; //start conversion
        while((aDCSRa&0x10)==0); //wait untill conversion is done
        value=aDCL;   //Read 8 low bits first (important)
        value|=(int)aDCH << 8; //read 2 high bits and shift into top byte

        return value;
}



void LCD_putint(int value){
        char buf[20];

        itoa(value, buf, 10);
        LCD_putword(buf);
}

void time_delay(int delay_ms)
{
        int delay = delay_ms;

        //Konfigurera Timer1
        TCNT1 = 0;
        TCCR1a = 0x00;
        TCCR1B = 0b00001101; //Borja om vid Output Compare, systemklockan/8
        while(delay>=0){
                if(TCNT1>=8) {
                        delay = delay - 1;
                        TCNT1 = 0;
                }
        }
}

void start_timer()
{
```

```
        time = 0;
        TCNT1 = 0;
        TCCR1a = 0x00;
        TCCR1B = 0b00001101; //Borja om vid Output Compare, systemklockan/8
}

int get_timer()
{
        time += TCNT1/8;
        TCNT1 = TCNT1%8;

        return time;
}

bool check_mic(){
        int value;
        value = aDC_read(0b00111);
        if (value > 0)
                return true;
        else
                return false;
}

void poll_while_armed(state_mode *state)
{
        //Poll motion
        if ((PINa & 0b00100000) == 0){
                state->zerocounter++;
        } else if (state->zerocounter > 0) {
                state->zerocounter=state->zerocounter - 1;
        }

        if (state->zerocounter > state->motion_threshold*400) {
                LCD_clear();
                LCD_putword("Motion detected!\nCode: ");
                state->triggered = true;
                state->armed = false;
                state->zerocounter = 0;
        }

        //Poll mic
        if (check_mic()){
                int thetime;
                if(state->micstate == 0){
                        start_timer();
                        state->micstate = 1;
                        _delay_loop_2(0xFFFF);
                }
                else if(state->micstate == 1){
                        thetime = get_timer();
```

```
                        if (thetime > state->micstate_times[0] - 250 && thetime <
state->micstate_times[0] + 250){
                                start_timer();
                                state->micstate = 2;
                                _delay_loop_2(0xFFFF);
                        }
                        else {
                                state->micstate = 0;
                                _delay_loop_2(0xFFFF);
                        }
                }
                else if(state->micstate == 2){
                        thetime = get_timer();
                        if (thetime > state->micstate_times[1] - 250 && thetime <
state->micstate_times[1] + 250){
                                //Korrekt mic-kod!
                                LCD_clear();
                                LCD_putword("Correct code!\nWelcome home!");
                                speaker_sound(0xFF, 200);
                                speaker_sound(0xFF, 200);
                                speaker_sound(0xFFF, 200);
                                //Delay sa att man hinner se status
                                time_delay(2000);
                                LCD_clear();
                                state->triggered = false;
                                state->armed = false;

                                state->micstate = 0;
                        }
                        else {
                                state->micstate = 0;
                                _delay_loop_2(0xFFFF);
                        }
                }
        } else if (get_timer() > 5000) {
                state->micstate = 0;
        }

        //Poll door
        if (PIND == 0b00000000) {
                LCD_clear();
                LCD_putword("Door opened!\nCode: ");
                state->triggered = true;
                state->armed = false;
        }
}

void poll_while_disarmed(state_mode *state)
{
        int i;
```

```
char c;
char entered[5];

//Poll keypad
c = wait_button_pressed();
speaker_sound(0xFF, 200);

if (c == 'Y') { //Y = andra kod
        LCD_clear();
        LCD_putword("Changing code\nOld code: ");
        for(i = 0;i<4;i++) { //Tryck in den gamla koden
                c = wait_button_pressed();
                if(isalpha(c)==0){
                        entered[i] = c;
                        entered[i+1] = '\0';
                        LCD_putchar('*');
                        speaker_sound(0xFF, 200);
                }
                else
                        --i;
        }
        if (strcmp(entered, state->password) == 0) {
                LCD_clear();
                LCD_putword("Changing code\nNew code: ");
                for(i = 0;i<4;i++) { //Tryck in den nya koden
                        c = wait_button_pressed();
                        if(isalpha(c)==0){
                                state->password[i] = c;
                                state->password[i+1] = '\0';
                                LCD_putchar('*');
                                speaker_sound(0xFF, 200);
                        }
                        else
                                --i;
                }
                LCD_clear();
                LCD_putword("Code changed!");
                eeprom_write_word(0xF0, atoi(state->password));
        } else {
                LCD_clear();
                LCD_putword("Wrong code!");
        }
        //Delay sa att man hinner se status
        time_delay(2000);
        LCD_clear();
        LCD_putword("Standby mode");

} else if (c == 'N') { //N = Skriv losenord for att alarmera systemet
        LCD_clear();
        LCD_putword("arming system\nCode: ");
```

```
        for(i = 0;i<4;i++) { //Tryck in koden
                c = wait_button_pressed();
                if(isalpha(c)==0){
                        entered[i] = c;
                        entered[i+1] = '\0';
                        LCD_putchar('*');
                        speaker_sound(0xFF, 200);
                }
                else
                        i--;
        }
        if (strcmp(entered, state->password) == 0) {
                LCD_clear();
                //Vanta 30 sec innan systemet blir alarmerat
                for(i = 30; i>0; i--) {
                        LCD_clear();
                        LCD_putword("alarm on in ");
                        LCD_putint(i);
                        LCD_putword(" s");
                        time_delay(1000);
        }
        state->triggered = false;
        state->armed = true;
        state->micstate = 0;
        state->zerocounter = 0;
        LCD_clear();
        LCD_putword("armed");
        } else {
                LCD_clear();
                LCD_putword("Wrong Code!");
                //Delay sa att man hinner se status
                time_delay(2000);
                LCD_clear();
                LCD_putword("Standby mode");
        }

} else if (c == 'D') {

        LCD_clear();
        LCD_putword("Current sense: ");
        LCD_putint(state->motion_threshold);
        LCD_putword("                        New sense: ");
        c = wait_button_pressed();
        while(isalpha(c)!=0){
                c = wait_button_pressed();
        }
        LCD_clear();
        LCD_putword("New sense: ");
        char temp[2];
        temp[0]=c;
```

```
                temp[1]='\0';
                state->motion_threshold = atoi(temp);
                LCD_putint(state->motion_threshold);
                eeprom_write_word(0x05, state->motion_threshold);

                //Delay sa att man hinner se status
                time_delay(2000);
                LCD_clear();
                LCD_putword("Standby mode");
        } else if(c == 'U') {
                int time1, time2;
                LCD_clear();
                LCD_putword("Knock once to\nstart timing");

                while(!check_mic()){}

                _delay_loop_2(0xFFFF);
                _delay_loop_2(0xFFFF);
                _delay_loop_2(0xFFFF);
                start_timer();
                LCD_clear();
                LCD_putword("Timing started..");

                while(!check_mic()){}

                time1 = get_timer();
                LCD_clear();
                LCD_putword("1st time: ");
                LCD_putint(time1);
                start_timer();

                while(!check_mic()){}

                time2 = get_timer();
                LCD_clear();
                LCD_putword("2nd time: ");
                LCD_putint(time2);

                eeprom_write_word(0x07, time1);
                eeprom_write_word(0x09, time2);
                state->micstate_times[0] = time1;
                state->micstate_times[1] = time2;

                //Delay sa att man hinner se status
                time_delay(2000);
                LCD_clear();
                LCD_putword("Standby mode");
        }
}
```

```
void poll_while_triggered(state_mode* state)
{
        int i;
        char c;
        char entered[5];

        for(i = 0;i<4;i++) {
                c = wait_button_pressed();

                if(isalpha(c)==0){
                        entered[i] = c;
                        entered[i+1] = '\0';
                        LCD_putchar('*');
                        speaker_sound(0xFF, 200);
                }
                else
                        i--;
        }
        if (strcmp(entered, state->password) == 0) {
                LCD_clear();

                time=0;//Stanger av larmet
                TCNT1=0;
                TCCR1B = 0b00001000;
                do_alarm = false;

                LCD_putword("Correct code!\nWelcome home!");

                //Delay sa att man hinner se status
                time_delay(2000);
                LCD_clear();
                state->triggered = false;
                state->armed = false;

        } else {
                LCD_clear();
                LCD_putword("Wrong code!\nCode: ");
        }
        //Lat larmet ga igang efter 20 sec
        if (get_timer()>20000 && (get_timer()%1000) ==0  && do_alarm) {
                speaker_sound(0xFFF, 100);
        }
}

void init_system(state_mode *state)
{
        //Har initieras alla pinnar
        DDRD = 0xFF;
        DDRa = 0b11011111;
```

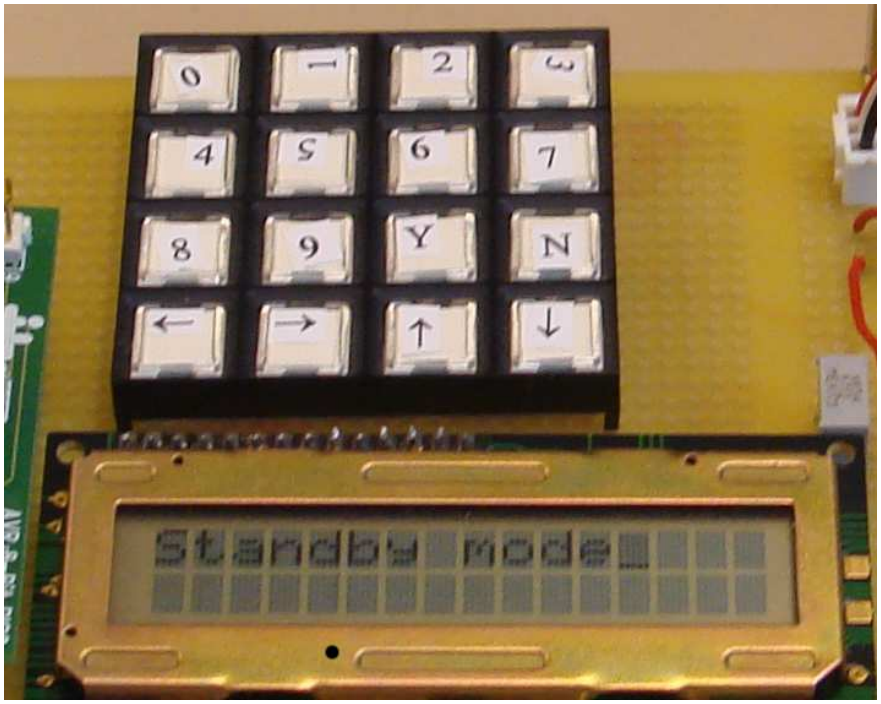
```
        //Displayen initsieras
        LCD_init();

        //Initsiering av state_mode
        if ((int)eeprom_read_word(0xF0) < 0) {
                eeprom_write_word(0xF0, 1234);
        }

        if ((int)eeprom_read_word(0x05) < 0) {
                eeprom_write_word(0x05, 5);
        }

        if ((int)eeprom_read_word(0x07) < 0) {
                eeprom_write_word(0x07, 1000);
        }

        if ((int)eeprom_read_word(0x09) < 0) {
                eeprom_write_word(0x09, 1000);
        }

        state->armed = false;
        state->triggered = false;
        state->zerocounter = 0;
        state->micstate = 0;
        state->motion_threshold = eeprom_read_word(0x05);
        state->micstate_times[0] = eeprom_read_word(0x07);
        state->micstate_times[1] = eeprom_read_word(0x09);
        itoa(eeprom_read_word(0xF0), state->password, 10);

        check_password(state);

        do_alarm = false;
}

int main()
{
        state_mode * state;
        state = (state_mode*)malloc(sizeof(state_mode));

        init_system(state);

        while(1) {

                while(state->armed) {
                        poll_while_armed(state);

                }

                if (!state->armed && !state->triggered) {
```

```
                        LCD_putword("Standby mode");

                        while(!state->armed && !state->triggered) {

                                poll_while_disarmed(state);

                        }

                }

                if (state->triggered) {

                        speaker_sound(0xaaa, 300);

                        start_timer();
                        do_alarm = true;
                        while(state->triggered) {

                                poll_while_triggered(state);

                        }
                }
        }
}
```

5.4 Appendix D: Manual

Systemet startar alltid i standbymode. Knacksekvensen kan endast hantera tre knackningar med olika intervall. Default-inställningen för koden är 1234. Defaultknacksekvensen är tre knackningar med en sekunds mellanrum. För rörelsekänsligheten är defaultvärdet fem. I standbymode kan man göra följande operationer:

- **Alarmera** - Tryck på knapp "N" och ange koden. Efter 30 sekunder aktiveras larmsystemet.
- **Byta kod** - Tryck på knapp "Y" och ange den gamla koden, knappa därefter in den nya koden.
- **Ändra rörelsekänslighet** - Tryck på knapp "↓" och ange den nya känsligheten.
- **Byta knacksekvens** - Tryck på knapp "↑". Första knackningen startar timing mellan första och andra knackningen. Andra knackningen startar timingen mellan andra och sista knackningen.



När Systemet är i alarmerat tillstånd kan det låsas upp antingen genom att ange knacksekvensen eller genom att slå in koden. Knacksekvensen anger man när systemet ej är utlöst och vid korrekt sekvens kommer ett litet pip. Koden anger man när detektorerna är triggade och då har man 20 sekunder på sig att knappa in koden innan larmet går. Man kan givetvis även stänga av larmet med koden när larmet ljuder.