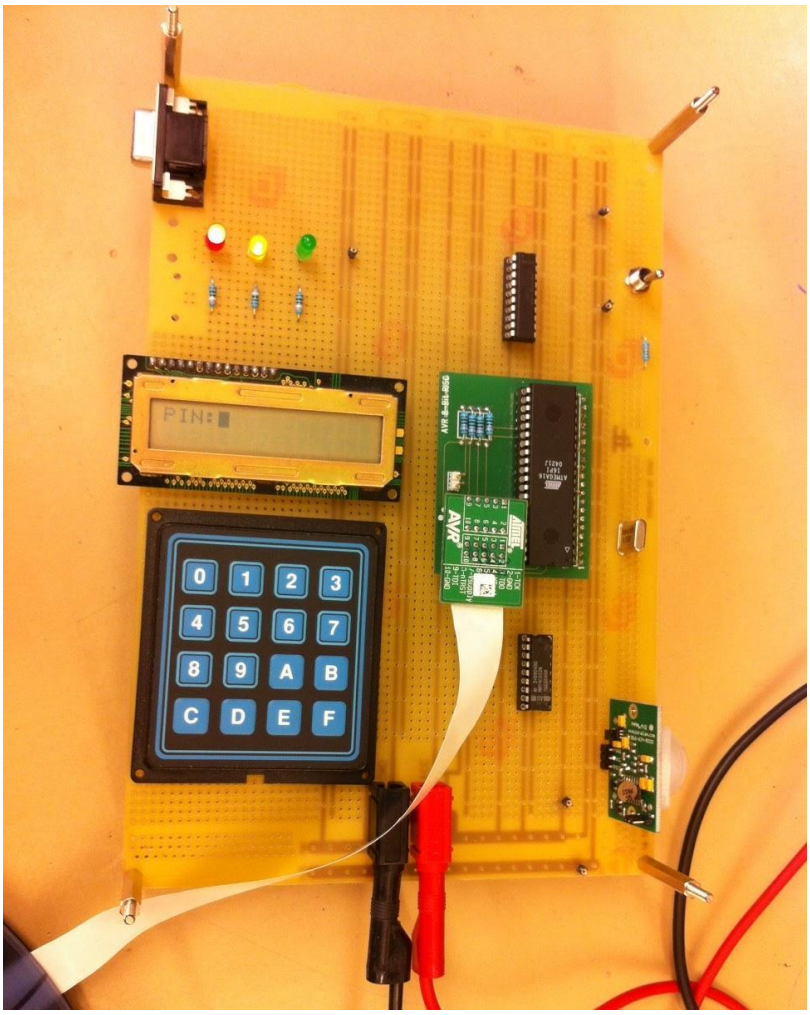


EIT

Digitala projekt EITF11

Larmanläggning

Handledare Bertil Lindvall
2014-05-14
Anna Lindberg I-11
Caroline Vitasp I-11
Eric Eliason I-10

Sammanfattning

I kursen digitala projekt (EITF11) på Lunds Tekniska Högskola genomförs ett projekt med syfte att ge förståelse för hur konstruktionsarbete går till. Denna rapport syftar ge en bild av hur projektet ”larmanläggning” projekterades, planerades och genomfördes. Rapporten beskriver arbetet från idé till färdig prototyp och innehåller förutom en utförlig beskrivning av arbetsprocessen även blockschema och källkod för den färdiga prototypen. Avslutningsvis presenteras en reflektion över arbetsprocessens svårigheter och potentiell vidareutveckling av prototypen.

Contents

2. Kravspecifikation 1

Produktbeskrivning 1

Användarscenario 2

3. Hårdvara 3

3.1 Kopplingsschema 3

3.2 Processor 3

3.3 LCD-skärm 3

3.4 Knappsats 4

3.5 Analog switch 4

3.6 IR-sensor 4

3.7 Lysdioder 4

3.8 Kristall 4

3.9 COM-port 4

4. Mjukvara 5

5. Konstruktionsprocessen 5

6. Diskussion och slutsats 6

7. Källkod 6

8. Referenser 18

1. Inledning

Digitala Projekt (EITF11) är en projektkurs med syfte att ge grundläggande förståelse för hur elektroniskt konstruktionsarbete går till. I grupper om två eller tre skall eleverna genom självständigt arbete konstruera en fungerande prototyp med tillhörande dokumentation.
Under de senaste åren har säkerhetsindustrin varit på stark frammarsch och larmsystem är mer eller mindre en självklarhet i villor och sommarstugor. Med bakgrund av detta valde gruppen att konstruera ett användarvänligt larmsystem. Systemet har förutom en IR-sensor som registrerar rörelse även vippströmbrytare som kan användas för att illustrera övervakning av dörrar och fönster. Larmet styrs genom en sifferknappsats och alfa-numerisk LCD-display.

För att minska omfattningen av arbetet till rimlig nivå används endast en strömbrytare för att illustrera dörrar och fönster, samt endast en IR-sensor. Detta anser vi inte påverkar våra kunskaper i området utan endast minskar skalan av ett riktigt hemmalarm.

2. Kravspecifikation

Den ska kunna övervaka 1 digital givare och 1 analog givare och kunna larma enligt användarscenariona nedan. Tiden som har gått sedan larmet sattes på skall kunna avläsas när larmet utlösts. Larmanläggningen manövreras med en knappsats och ger utskrifter på en display. Systemet loggar vid vilken tidpunkt larmet triggas. Larmpersonal kan deaktivera larmet genom ett visst kommando efterföljt av korrekt kod. Larmanläggningen sätts på och slås av med en av användaren bestämd “hemlig” fyrsiffrig kod (utskrifter av PIN-koden på displayen illustreras med ‘*’). Det är möjligt att byta koden i avslaget läge.

De dioder som finns är en grön, en gul och en röd. I avslaget läge ska den gröna dioden lysa, i läget “alarmReady” (vilket innebär att larmet snart sätts på) och läget “personEntered” (vilket innebär att någon kommit in och larmet snart går av) lyser den gula dioden, och då larmet är på och ingen är hemma lyser den röda dioden. Sirén kommer inte att användas men kan lätt läggas till vid behov, istället kommer alla LEDer lysa för att illustrera att sirénen ljuder när larmet har gått.

Produktbeskrivning

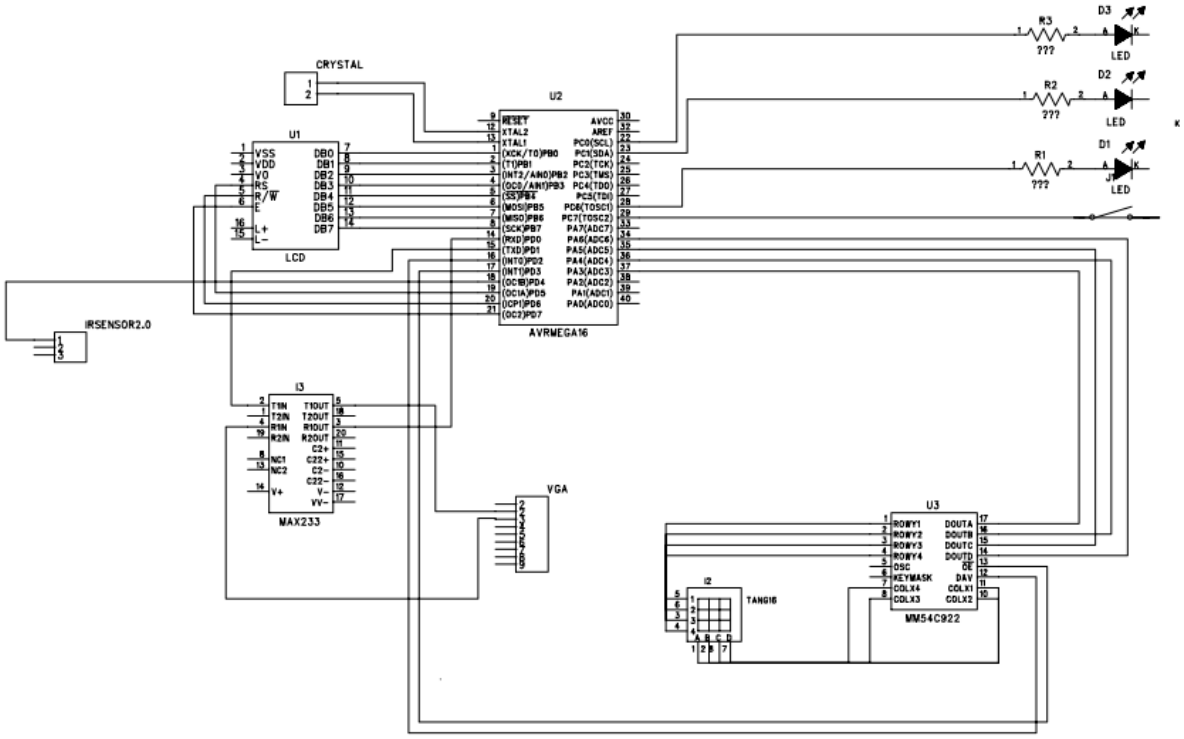
- 1 st digital rörelsesensor (för att registrera rörelser inomhus)
- 1 st analog switch (för att illustrera dörrar och fönster)
- 1 st knappsats med siffrorna 0-9 (samt ytterligare knappar för extra-funktioner och kommandon)
- 1 st digital alfa-numerisk display (för att informera om larmets status och interagera med användaren)
- 3 st dioder (1 st grön, 1 st gul, 1 st röd, för att tydligt och intuitivt visa larmets läge)

Användarscenario

- Larmet sätts igång (från avslaget läge) 10 sekunder efter det att rätt kod slagits
    - Om fel kod slås i avslaget läge händer inget (oändligt antal ggr)
    - Under dena tid ska sensorn kunna registrera rörelse samt dörrar kunna öppnas utan att något händer
- Om dörren öppnas när larmet är på har man 10 sekunder på sig att slå rätt kod
    - Om fel kod slås 3 ggr ska larmet gå av direkt
- Om IR-sensorn registrerar rörelse när larmet är på har man 10 sekunder på sig att slå rätt kod
    - Om fel kod slås 3 ggr ska larmet gå av direkt
    - Sensorerna registrerar temperaturförändringar på 5 °C
- Alla gånger då larmet går av skickas ett meddelande till larmcentral med tidsstämpel.
- Larmet ska vara hårdvarumässigt förberett för kommunikation med en dator för att kunna logga önskad information.

3. Hårdvara

3.1 Kopplingsschema



3.2 Processor

Processorn och grunden till larmet är en ATmega16 High-performance AVR 8-bit Microcontroller. Processorns hårdvara programmeras i C med hjälp av programmet AVR Studio 6 och överförs till processorn med hjälp av en JTAG.

3.3 LCD-skärm

Användaren interagerar med larmet med hjälp av en SANYO alfa-numerisk teckendisplay. På displayen skrivs status på larmet (ex aktiverat/avaktiverat) samt uppmaningar till användaren (ex skriv in pin-kod) ut.

3.4 Knappsats

Larmet har en knappsats med 16 knappar fördelade på fyra rader och fyra kolumner. Knappsatsen är ansluten till en knappsatsdekoder som läser av knappsatsen med en frekvens bestämd av två kondensatorer. När en knapp trycks in signalerar dekodern till processorn att det finns tillgänglig data, och den läser av.

3.5 Analog switch

Som symbol för dörr och/eller fönster används en analog switch. I ena läget har switchen kontakt med låg spänning och i andra läget med hög spänning, som skickas till processorn direkt. Dörren anses vara stängd då låg signal skickas och öppen då hög signal skickas.

3.6 IR-sensor

IR-sensorn mäter temperaturen i rummet och registrerar avvikelser från denna som rörelser. Dvs om något i rummet har en temperatur som avviker med mer än ett visst värde från omgivningen ger sensorn utslag och skickar en digital signal till processorn.

3.7 Lysdioder

Till larmet är tre lysdioder kopplade, en röd, en gul och en grön. Dessa dioder används som komplement till LCD-skärmen för att kommunicera vilket läge larmet befinner sig i till användaren. I avslaget läge ska den gröna dioden lysa, i läget “alarmReady” (vilket innebär att larmet snart sätts på) och läget “personEntered” (vilket innebär att någon kommit in och larmet snart går av) lyser den gula dioden, och då larmet är på och ingen är hemma lyser den röda dioden.

3.8 Kristall

För att få en mer exakt klocka hos processorn används en kristall av typen CMACKD 8MHZ, som med andra ord svänger 8 miljoner gånger per sekund. Detta är för att processorn på ett effektivt sätt ska kunna kommunicera med en dator via en COM-port som finns monterad på larmet.

3.9 COM-port

Via en COM-port som sitter monterad på larmet kan det i framtiden kommunicera med en dator som i sin tur fungerar som en larmcentral. Detta stöds inte mjukvarumässigt för att kund lättare ska kunna välja själv vilken information som ska kunna utbytas mellan larmet och datorn. COM-porten är kopplad via en MAX233 till processorn för att på så sätt få rätt spänning.

4. Mjukvara

Larmets mjukvara är uppbyggd kring en huvudloop som kontinuerligt körs fram till att ett avbrott genereras. I huvudloopen aktiveras larmets LED-lampor beroende på larmets tillstånd. I denna loop läses larmets två sensorer (Switch och IR-sensor) kontinuerligt av samt att utskrift av vissa meddelande sker.

Avbrott genereras på två sätt, det första är då någon av knappsatsens knappar trycks ner. När så sker körs en metod för att identifiera vilken av knapparna som tryckts när och därefter skicka vidare en exekveringsorder med vilket kommando som skall utföras. Den andra typen av avbrott är ett så kallat tidsavbrott. Denna typ av avbrott är nödvändig dels eftersom tiden då sensorerna aktiverats ska registreras och dels för att larmet ska aktiveras först 10 sekunder efter det att sensorerna aktiverats.

5. Konstruktionsprocessen

Projektet inleddes med att en kravspecifikation formulerades. Denna diskuterades sedan med handledaren och reviderades för att projektet skulle rymmas inom tidsramen för kursen. Därefter ritades ett kopplingsschema i enlighet med komponenternas datablad varefter prototypens beståndsdelar konstruerades och sammankopplades. När hårdvaran konstruerats och lötts samman kopplades larmets processor till datorn med hjälp av en JTAG. Felsökning genomfördes i programmet AVR Studio 6 med hjälp av JTAGen och det kontrollerades att komponenterna fått rätt spänning och att de reagerade som det var tänkt då de fick hög respektive låg spänning.

LCD-displayen var den första komponenten att programmeras, vilket visade sig vara problematiskt, dels på grund av hårdvarufel och dels genom att gruppen hade vissa svårigheter att få korrekt utskrift på skärmen. För att systemet inte skulle reagera för snabbt och skriva över text skrevs kod för att systemet enbart skulle reagera om displayens ”Busy Flag” tillät detta. Knappsatsen kopplades därefter till dekodern och kod skrevs för att processorn skulle skriva ut rätt siffra på skärmen då aktuell knapp trycktes in. Därefter skrevs avbrottskoden, exempelvis vilken metod som aktiveras då rätt pin-kod slagits in följt av ett ”enter-tryck” eller vilken metod som exekveras när någon av larmets sensorer aktiveras.

Först efter detta skrevs det första riktiga “main-programmet” till larmet, där vi kopplade ihop sensorer, LCD och knappsats mm. Här finputsades sedan egenskaper som att PIN-koden illustreras med ‘*’, dioder sätts av och på vid rätt tillfällen, tiden löper rätt mm. Tillslut var allt klart!

6. Diskussion och slutsats

När hårdvaran till larmet kopplats färdigt testades det med en logikpenna så att allting var korrekt kopplat. Det visade sig då att ett par komponenter felkopplats samtidigt som det behövde adderas ytterligare några för att underlätta den kommande kodningen. Många oförutsedda problem uppstod, såsom felaktiga inserts till olika komponenter samt en felaktig processor. Felkopplingen kunde ha förutsatts, men alla problem som har stötts på har lett till viktiga lärdomar!

Det största problemet med kodningen visade sig vara att få korrekt utskrift på LCD-displayen och gruppen fick ägna många timmar för att korrekt initiera denna och programmera koden för ”busy flag”. Därefter fick mycket tid läggas på att förstå upplägget av programmeringsspråket och läsa datablad för att korrekt sätta upp larmet och få det att agera som önskat.

På det stora hela gick projektet mycket bra trots att samtliga gruppmedlemmar för första gången arbetade med en konstruktion av detta slag, samt praktiskt använda programmeringsspråket C. Gruppen är nöjda med projektet i allmänhet. Det mest värdefulla anses vara insikten i liknande konstruktionsarbeten samt med att ha fått nya programmeringskunskaper.

7. Källkod

```
#include <avr/io.h>
#include <util/delay.h>
#define USART_BAUDRATE 9600
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)
#include <avr/interrupt.h>
#define sei()
#define cli()
#define F_CPU 8000000UL


#define LCD_DATA     PORTB
#define LCD_E  PD7   // PORTD.7
#define LCD_WR PD6   //PORTD.6
#define LCD_RS PD5   //PORTD.5

#define LED_GREEN PC0
#define LED_YELLOW PC1
#define LED_RED PC6

#define KEY_DOA PA3
#define KEY_DOB PA4
#define KEY_DOC PA5
#define KEY_DOD PA6

unsigned int activated = 0;
unsigned int changingCode = 0;
unsigned int timerIn = 0;
```

```
unsigned int alarmTriggered = 0;
unsigned int deactivated = 1;
unsigned int timerOut = 0;
unsigned int alarmReady = 0;
unsigned int sec = 0, min = 0, h = 0;
unsigned int charNbr = 0;
unsigned int pinCounter = 0;
unsigned int expectPin = 0;
unsigned int personEntered = 0;
unsigned int burglary = 0;
unsigned int printOnce = 0;

int timeVect[8];
volatile uint8_t count;

unsigned char pinCode[4] = {'1', '3', '3', '7'};
unsigned char pinCodeTry[4];


char read_keypad(char code) {
	if (code == 0b00000000){
		return '0';
		}else if(code==0b00001000){
			return '1';
		}else if(code==0b00010000){
			return '2';
		}else if(code==0b00011000){
			return '3';
		}else if(code==0b00100000){
			return '4';
		}else if(code==0b00101000){
			return '5';
		}else if(code==0b00110000){
			return '6';
		}else if(code==0b00111000){
			return '7';
		}else if(code==0b01000000){
			return '8';
		}else if(code==0b01001000){
			return '9';
		}else if(code==0b01010000){
			return 'A';
		}else if(code==0b01011000){
			return 'B';
		}else if(code==0b01100000){
			return 'C';
		}else if(code==0b01101000){
			return 'D';
		}else if(code==0b01110000){
			return 'E';
		}else if (code==0b01111000){
			return 'F';
		}
	}

char usartRead() {

	while(!(UCSRA & (1<<RXC))); //Wait until data is available
```

```
	return UDR;
}

void usartWrite(char data) {
	while(!(UCSRA & (1<<UDRE))); //Wait until the buffer is empty
	UCSRA = UCSRA | (1<<UDRE); // Set UDRE = 1
	UDR = data;
	UCSRA = UCSRA & ~(1<<UDRE); //Set UDRE = 0
}

int main(void) {
	set_Init();
	check_BF();
	PORTB = 0b00000001; //Clear display
	toggle_E();
	check_BF();
	keypad_setup();
	enableKeypadInterrupt();
	enableTimeInterrupt();
	showTime(h, min, sec);
	next_Line();

	while(1) {
		check_Switch();
		check_IRsensor();


			if (!activated && !burglary) {
				set_pin('C', LED_GREEN, 1);
			}
			if (alarmReady && timerOut == 1) { //När man slår pinkoden för att
sätta igång larmet
				set_pin('C', LED_GREEN, 0);
				set_pin('C', LED_YELLOW, 1);
				//printAlarmReady();
			}
			if (timerOut >= 10){ //När det har gått 10 sekunder efter man slagit
pinkoden sätts larmet igång (hoppas man gått ut!!!)
				set_pin('C', LED_GREEN, 0);
				set_pin('C', LED_YELLOW, 0);
				set_pin('C', LED_RED, 1);
				write_alarm_on();
				activated = 1;
				alarmReady = 0;
				timerOut = 0;
			}
			if (personEntered && timerIn ==1 ) {
				set_pin('C', LED_YELLOW, 1);
				if (printOnce){
					printExpectPin();
					printOnce=0;
				}

			}
			if (alarmTriggered)  {
				set_pin('C', LED_RED, 1);
				set_pin('C', LED_YELLOW, 1);
```

```
                        set_pin('C', LED_GREEN, 1);
                        if (printOnce){
                                print_Alarm();
                                showTime(h, min, sec);
                                printOnce=0;
                        }
                        burglary = 1;
                        h, min, sec = 0;
                        activated = 0;
                        alarmTriggered = 0;
                }
        //}
        }
}

ISR(INT0_vect) { // Global Interrupt (Känner av och läser av om knappar trycks)

        cli();
        char code = PINA & 0b01111000;
        char input = read_keypad(code);

        if(input == 'B') { // deactivate
                if(activated) {
                        expectPin=1;
                        printExpectPin();
                } else {
                        printAlreadyOFF();
                }
        }else if(input == 'A') { // activate
                if(!activated) {
                        expectPin=1;
                        printExpectPin();
                } else {
                        printAlreadyON();
                }
        } else if(input == 'F') { // change code
                expectPin=1;
                print_Old_Pin();
                changingCode = 1;
        } else if(input == 'C') {
                clear_display();
        } else if(input == 'D') {
        expectPin=1;
        clear_display();
        printTurnOffSiren();
        }
        else if(expectPin) {
                insertCode(input); // any of the numbers 0 to 9
                if(charNbr == 4) {
                        charNbr = 0;
                        expectPin = 0;
                        next_Line();
                        if(changingCode == 2) {
                                change_Pin();
                                changingCode = 0;
                                printChanged();
                        } else {
                        check_Mode();
```

```
                        }
                }
        }
        sei();
}

void check_Mode() { // Determines if the alarm should be activated or deactivated
        if(changingCode == 1) {
                if(compare()) {
                        print_Enter_New();
                        changingCode = 2;
                        expectPin=1;
                } else {
                        printWrongPin();
                        changingCode = 0;
                }
        } else {
                if(compare() && personEntered) { //If the PIN is correct and the alarm is
(in personEntered-mode) activated -> Deactivate
                        activated = 0;
                        print_Alarm_Off();
                        set_pin('C', LED_RED, 0);
                        set_pin('C', LED_YELLOW, 0);
                        set_pin('C', LED_GREEN, 0);
                        personEntered = 0;
                } else if(compare() && !activated && burglary) {
                        set_pin('C', LED_GREEN, 0);
                        set_pin('C', LED_RED, 0);
                        set_pin('C', LED_YELLOW, 0);
                        burglary = 0;
                }
                else if(compare() && !activated) { // If the PIN is correct and the alarm
isn't activated -> Activate
                        printAlarmReady();
                        alarmReady = 1;
                } else if(!compare() && personEntered) {
                        printWrongPin();
                        check_Tries();
                } else if(!compare()) {
                        printWrongPin();
                }
        }
}

int compare() { // Compares the entered code with the correct code (and returns a
"boolean")
        for(unsigned short int k = 0; k < 4; k++) {
                if(pinCode[k] != pinCodeTry[k]) {
                        return 0;
                }
        }
        return 1;
}

void check_Tries() { // Wrong PIN-code three times triggers the alarm
        if (activated) {
                pinCounter++;
        }
```

```
	if (pinCounter == 3) {
		alarmTriggered = 1;
		printOnce=1;
		pinCounter = 0;
	}
}

void insertCode(char letter) {
	pinCodeTry[charNbr] = letter;
	write_Data('*');
	charNbr++;
}

void change_Pin() {
	for(unsigned int i = 0; i<=3; i++) {
	pinCode[i] = pinCodeTry[i];
	}
}

void check_Switch(){
	if (activated) {
		char set = PINC & 0b10000000;
		if(set == 0b10000000){
			personEntered = 1;
			printOnce=1;
		}
	}
}

void check_IRsensor() {
	if (activated) {
	char set = PIND & 0b00010000;
		if(set == 0b00010000){
			personEntered = 1;
			printOnce=1;
		}
	}
}

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

```
                        PORTB ^= set;
                }
        }
        else if(port == 'C'){
                set &= PORTC;
                if(set && !state){ //Set PIN
                        PORTC ^= set;
                }
                if(set == 0 && state){ //Clear PIN
                        set = 1 << pin;
                        PORTC ^= set;
                }
        }
        else if(port == 'D'){
                set &= PORTD;
                if(set && !state){ //Set PIN
                        PORTD ^= set;
                }
                if(set == 0 && state){ //Clear PIN
                        set = 1 << pin;
                        PORTD ^= set;
                }
        }
}

void next_Line() {
        PORTD &= 0b10011111;
        PORTB = 0b10101000;
        toggle_E();
}


void toggle_E(){
        PORTD &= 0b01111111; //Toggle  0x7F
        _delay_ms(10);
        PORTD |= 0b10000000; //Toggle  0x80
        _delay_ms(10);
}


char read_Cmd() { //Not used presently
        PORTD |= 0b01000000; //RW = 1
        DDRB = 0b00000000; //DDRB till in
        toggle_E();
        char val = PORTB;
        DDRB = 0b11111111; //DDRB ut
        PORTD &= 0b10111111; //RW = 0
        return(val);
}

void check_BF() {
        char slask = 1;
        _delay_ms(10);
        while (slask != 0) {
                slask = read_Cmd() & 0b10000000;
        }
}
```

```
void clear_display() {
	PORTB = 0b00000001;
	toggle_E();
}

ISR(TIMER0_OVF_vect) { //We made this work but timer1 can also be used.
	count++;
	if(count == 4) { /// Enters this loop once every second
		if(activated) {
			sec++;
		}
		if(sec == 60) {
			min++;
			sec = 0;
			if(min == 60) {
				h++;
				min = 0;
				if(h == 24) {
					h = 0;
				}
			}
		}
		if (personEntered) {
			timerIn++;
			if(timerIn >= 10) {
				personEntered = 0;
				alarmTriggered = 1;
				printOnce=1;
				timerIn = 0;
			}
		}
		if (alarmReady) {
			timerOut++;
		}
		count = 0;
	}

}
void enableKeypadInterrupt() {
	MCUCR = (1 << ISC01) | (1 << ISC00);
	SREG = 0x82;
	GICR = (1 << INT0);
	sei();
}

void enableTimeInterrupt() {
 TCCR0 |= (1 << CS02) | (1 << CS00); /*| (0 << CS01)*/ // Prescaler factor 1024
 TIMSK |= (1 << TOIE0) | (1<<TOIE1);// Enable overflow interrupt
 TCNT0 = 0x00;
 count = 0;
 sei();
}

void set_Init() {
	PORTD = 0b10000000; //set E till 1
	DDRA = 0x00;
	DDRB = 0xFF; //0b11111111 DDRB ut
	DDRD = 0b11101000; //0xE0;  7,6,5,3 ut
```

```
	PORTB = 0x0F; //ON/OFF 0b00001111
	check_BF();
	toggle_E();
	check_BF();
	PORTB = 0x38; //Function set 0b00111000
	check_BF();
	toggle_E();
	PORTB = 0b00000110; //Entry mode set 0x06
	toggle_E();
	DDRC = 0b01111111;
	sei();
}

//write_cmd används för att skriva kommandon till lcdn, hur den ska bete sig.
void write_cmd(char ch) {
	PORTB = ch;
	PORTD &= 011111111; //RS = 0
	toggle_E();
}

void write_Data(char ch) {
	DDRB = 0b11111111;
	PORTB = ch;
	PORTD |= 0b00100000;
	toggle_E();
	PORTD &= 0b11011111; //sätt R/W till 0
}

void keypad_setup() {
	DDRA = 0b00000000;
}

void showTime(int hour, int minute, int second){
	display_Num(hour);
	write_Data(':');
	display_Num(minute);
	write_Data(':');
	display_Num(second);
	next_Line();
}
void display_Num(int time){
	int time1=time/10;
	itoa(time1,timeVect,10);
	write_Data(timeVect[0]);
	int time2=time%10;
	time1=itoa(time2,timeVect,10);
	write_Data(timeVect[0]);
}

void print_Alarm() {
	clear_display();
	write_Data('*');
	write_Data('A');
	write_Data('L');
	write_Data('A');
	write_Data('R');
	write_Data('M');
	write_Data('!');
```

```
        write_Data('*');
        next_Line();

}

void print_Activated() {
        clear_display();
        write_Data('A');
        write_Data('C');
        write_Data('T');
        write_Data('I');
        write_Data('V');
        write_Data('A');
        write_Data('T');
        write_Data('E');
        write_Data('D');
        next_Line();
}

void printAlreadyON() {
        clear_display();
        write_Data('A');
        write_Data('l');
        write_Data('r');
        write_Data('e');
        write_Data('a');
        write_Data('d');
        write_Data('y');
        write_Data(' ');
        write_Data('O');
        write_Data('n');
        next_Line();
        _delay_ms(1000);
        clear_display();
}

void printAlreadyOFF() {
        clear_display();
        write_Data('A');
        write_Data('l');
        write_Data('r');
        write_Data('e');
        write_Data('a');
        write_Data('d');
        write_Data('y');
        write_Data(' ');
        write_Data('O');
        write_Data('f');
        write_Data('f');
        next_Line();
        _delay_ms(1000);
        clear_display();
}

void printExpectPin() {
        clear_display();
        write_Data('P');
        write_Data('I');
```

```
	write_Data('N');
	write_Data(':');
}

void printWrongPin() {
	clear_display();
	write_Data('W');
	write_Data('r');
	write_Data('o');
	write_Data('n');
	write_Data('g');
	write_Data(' ');
	write_Data('P');
	write_Data('I');
	write_Data('N');
	next_Line();
	_delay_ms(1000);
	clear_display();
}

void write_alarm_on() {
	clear_display();
	write_Data('A');
	write_Data('l');
	write_Data('a');
	write_Data('r');
	write_Data('m');
	write_Data(' ');
	write_Data('O');
	write_Data('n');
	next_Line();
}

void print_Alarm_Off(){
	clear_display();
	write_Data('A');
	write_Data('l');
	write_Data('a');
	write_Data('r');
	write_Data('m');
	write_Data(' ');
	write_Data('O');
	write_Data('F');
	write_Data('F');
	next_Line();
	_delay_ms(1000);
	clear_display();
}

void print_Old_Pin() {
	clear_display();
	write_Data('O');
	write_Data('l');
	write_Data('d');
	write_Data(' ');
	write_Data('P');
	write_Data('I');
	write_Data('N');
```

```
	write_Data(':');
}

void print_Enter_New() {
	clear_display();
	write_Data('E');
	write_Data('n');
	write_Data('t');
	write_Data('e');
	write_Data('r');
	write_Data(' ');
	write_Data('N');
	write_Data('E');
	write_Data('W');
	write_Data(':');
}

void printChanged() {
	clear_display();
	write_Data('P');
	write_Data('I');
	write_Data('N');
	write_Data(' ');
	write_Data('C');
	write_Data('h');
	write_Data('a');
	write_Data('n');
	write_Data('g');
	write_Data('e');
	write_Data('d');
	next_Line();
	_delay_ms(1500);
	clear_display();
}

void printAlarmReady() {
	clear_display();
	write_Data('A');
	write_Data('l');
	write_Data('a');
	write_Data('r');
	write_Data('m');
	write_Data(' ');
	write_Data('R');
	write_Data('e');
	write_Data('a');
	write_Data('d');
	write_Data('y');
	next_Line();
}

void printTurnOffSiren() {
	clear_display();
	write_Data('T');
	write_Data('u');
	write_Data('r');
	write_Data('n');
	write_Data(' ');
```

```
        write_Data('O');
        write_Data('f');
        write_Data('f');
        write_Data(' ');
        write_Data('S');
        write_Data('i');
        write_Data('r');
        write_Data('e');
        write_Data('n');
        next_Line();
}
```

8. Referenser

Datablad:

- ATMEL – 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash, Atmega16
- 555-28027-PIR-Sensor-Prodcut-Doc-v2.2
- 27899-4x4-Matrix-Membrane-Keypad-v1.2
- CRYSTAL
- LCD
- MAX220-MAX249
- mm74c922_3_fair

The C Programming Language Second Edition – Brian W. Kernighan, Dennis M. Ritchie