Digitala Projekt (EITF11)
Larmanläggning

Handledare: Bertil Lindvall
2013-05-20

Caroline Brandt, I-10
Jonathan Bratel, I-10
Angelika Jansson, I-10

Sammanfattning

Kursen Digitala Projekt (I) har som syfte att skapa förståelse för hur konstruktionsarbete går till. För att uppnå detta skulle alla kursdeltagare planera och genomföra ett projekt där en fungerande prototyp med tillhörande dokumentation skulle tas fram. Denna rapport beskriver arbetsprocessen i samband med framtagningen av en larmanläggning. Rapporten beskriver dels larmets olika komponenter men även den mjukvara som tagits fram samt hur arbetsprocessen har fortlöpt. Slutligen presenteras även resultatet av detta projekt samt reflektioner över de problem som uppstått och hur prototypen skulle kunna utvecklas i framtiden.

Innehållsförteckning

1 Inledning

Detta projekt ska resultera i ett larmsystem som känner av om dörrar eller fönster öppnas då larmet är aktiverat. Som en extra säkerhet ska larmet även känna av om personer rör sig i rummet då larmet är aktiverat.

Insignaler till larmet ges av en knappsats samt en digital och en analog sensor. Knappsatsen kan användas till att aktivera eller inaktivera larmet samt till att byta kod för larmet. Utsignal till larmet består av en röd LED, en gul LED, en grön LED och en display.

Ut- och insignaler till larmet kan även skickas från en dator via en COM-port som finns inkopplad på larmet. Exempelvis kan larmet avaktiverats från datorn när larmet har gått. På datorn kan även alla kommandon som utförs på larmet följas.

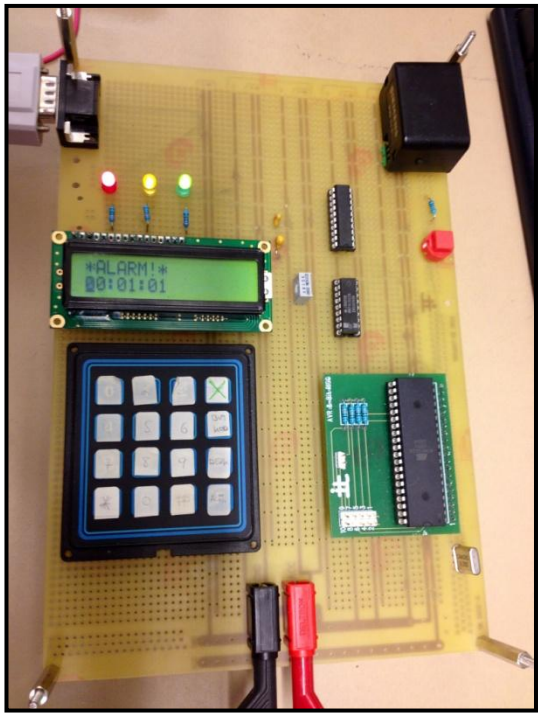

Bild 1: Larmsystemet då larmet gått

2 Kravspecifikation

- Larmet ska kunna hantera insignaler från en digital och en analog sensor.
- Anläggningen ska innehålla en knappsats för val av olika operationer samt inläsning av numerisk kod.
- Larmet ska kunna kopplas till en dator.
- Larmet ska kunna aktiveras och avaktiveras med en numerisk kod på fyra siffror.
- Efter att larmet har aktiverats tar det 10 sekunder innan larmet går på.
- Om en sensor registrerar rörelse ska koden matas in inom 10 sekunden, annars går larmet.
- Om larmet går ska larmets röda, gula och gröna LED lysa och signaler ska skickas till eventuell larmcentral.
- Om larmet går ska tiden för detta registreras och visas i displayen.
- Om larmet är aktiverat lyser larmets röda LED och texten ”AlarmON” visas på displayen.
- Om larmet avaktiveras släcks larmets alla LED och texten ”AlarmOFF” visas på displayen. Larmets gröna LED blinkar även till.
- Koden för aktivering och inaktivering ska kunna ändras.

3 Hårdvara

3.1 Kopplingsschema

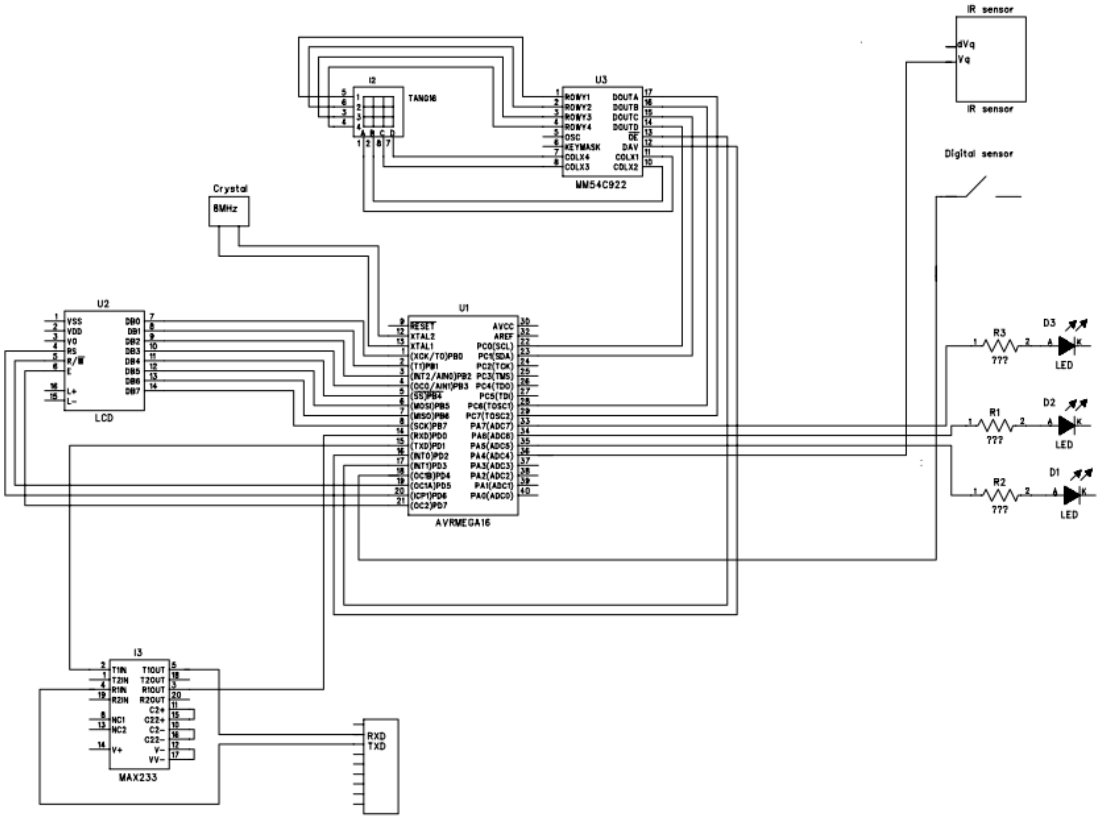

Bild 2: Kopplingsschema

3.2 Processor

En ATmega16 High-performance AVR 8-bit Microcontroller utgör grunden till hela vårt larm. Med hjälp av en JTAG har den mjukvara vi skrivit laddats över till processorn.

3.3 Knappsats

En 16-bitars knappsats som kopplats till en kodare (MM54C922/MM74C922 16-Key Encoder). Med hjälp av kodaren får varje knapp en viss kod som underlättar programmeringen av mjukvaran. Knappsatsen har en knapp för aktivering, en knapp för avaktivering samt en knapp för byte av kod. Dessutom finns knappar för siffrorna 0 – 9 vilka används för att mata in koden. Övriga knappar används ej.

3.4 LCD-skärm

En LCD-skärm (SHARP Dot-Matrix , LCD Units Alfanumerisk teckendisplay) som används för kommunikation mellan larmet och användaren. Här skrivs status för larmet och uppmaningar till användaren ut då input efterfrågas.

3.5 Magnetkontakt

En switch används för att symbolisera ingångsdörren till rummet. Ett tryck på knappen motsvarar att dörren öppnas.

3.6 Lysdioder
Tre stycken lysdioder (en röd, en gul och en grön) används som komplement till LCD-skärmen, för att kommunicera i vilket läge larmet befinner sig.

3.7 IR-sensor
En IR-detektor (Siemens PID 11T) används för att registrera rörelser i rummet. Uppfattad rörelse leder till en ändring i spänningen som skickas till processorn.

3.8 COM-port
Via en COM-port kan larmet kommunicera med en extern dator som skulle kunna fungera som en larmcentral. COM-porten kopplas via en MAX233 till processorn eftersom COM-porten har en spänning på -12V till +12V medan processorn endast har en spänning på 0V till +5V.

3.9 Kristall
En CMACKD 8MHz kristall används för att få en mer exakt klocka hos processorn. Detta är nödvändigt när processorn ska synkroniseras till en dator via COM-porten. Utan kristallen kan data förvrängas vid överföringen.

4 Mjukvara
Programmet är uppbyggt så att det är en huvudloop som körs kontinuerligt fram till dess att någon typ av avbrott sker. I denna loop läses sensorerna av och härifrån styrs även larmets LED-lampor utifrån vilket tillstånd larmet befinner sig i. Vissa utskrifter på LCD-displayen initieras även från huvudloopen.

Avbrott sker då någon av knappsatsens tangenter trycks ner. När detta inträffar går programmet in i en metod som tar reda på vilken knapp som har tryckts ner och skickar sedan denna information vidare till en metod som initierar det kommando som ska göras.

En annan typ av avbrott som använts vid programmeringen är ett tidsavbrott. Detta används för att tiden ska räknas upp under hela den period som larmet är aktiverat. Uppräkning av tiden är nödvändigt dels eftersom tiden då larmet går ska registreras och eftersom larmet ska gå först 10 sek efter att någon av sensorerna reagerat.

5 Arbetsprocessen

Arbetet inleddes med att en kravspecifikation för den färdiga produkten togs fram. Utefter denna skrevs en specifikation över vilken hårdvara som skulle komma att behövas och i samband med detta ritades även en första grov skiss på ett kopplingsschema upp. När detta var gjort utvecklades kopplingsschemat så att det tydligt beskrev hur alla komponenter skulle kopplas till processorn samt till varandra. Detta låg sedan till grund för all koppling som skedde vid framtagningen av vår prototyp.

När prototypen var klar påbörjades testningen av de olika komponenterna för att säkerställa att allt fungerade som det skulle. Denna testning gjordes med hjälp av programmet AVR Studio vilket även användes vid framtagningen av mjukvaran. I detta skede upptäcktes missar som gjorts under den inledande kopplingen och detta fick då åtgärdas.

Själva kodningen gjordes i språket C. Under kodningens gång hämtades information om språkets uppbyggnad främst från boken *The C Programming Language* (Kernighan & Ritchie) men även från olika hjälpsajter på internet.

6 Slutsats

Projektet resulterade i ett larm som dels känner av ifall dörren till ett rum öppnas och dels känner av rörelser i rummet. Om någon av larmets sensorer triggas kommer larmet att gå efter 10 sekunder om inte rätt kod matas in. Funktioner för att byta kod samt att kunna kommunicera med en annan dator har även lagts till. Alla krav som ställdes på produkten i det inledande skedet av arbetsprocessen har uppnåtts och ytterligare funktioner har även lagts till.

Svårigheter som uppkommit längs arbetets gång har främst berott på en ovana och okunskap inom ämnet elektronik. Förståelsen för hur arbetsprocessen inom området elektronik går till har ökats och den förkärlek för datablad som tidigare var helt obefintlig blomstrar nu mer än någonsin.

Skulle ytterligare förbättringar till produkten genomföras kan det vara lämpligt att uppgradera kopplingen mellan larmet och den externa datorn så att denna kan göras trådlös vilket gör larmet mer effektivt och verklighetstroget. Ytterligare en implementeringsförbättring som skulle göra larmet mer effektivt och säkert är att byta ut den knapp som inbrottstjuven i nuläget behöver trycka på för att utlösa larmet mot en magnetkontakt som kan monteras i dörröppningen.

7 Kod

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

#define red                     PA5
#define yellow                  PA6
#define green   PA7

unsigned int activated = 0;
unsigned int alarmTrigged = 0;
unsigned int deactivated = 1;
unsigned int alarmReady = 0;
unsigned int alarmReadySec = 0;
unsigned int actByPin = 0;
unsigned int timer = 0;
unsigned int expectPin = 0;
unsigned int sek = 0, min = 0, h = 0;
unsigned int changingCode = 0;

unsigned char pinCode[4] = {'1', '2', '3', '4'};
unsigned char pinCodeTry[4];
unsigned int charNbr = 0;
unsigned int newCharNbr = 0;
unsigned int pinCounter = 0;
unsigned int irTimer = 0;
int timeVect[8];

volatile uint8_t count;

void insertCode(char letter);
unsigned short int compare();
void printWrongPin();
void printActivated();
void printExpectPin();
void printAlarm();
char getButton(unsigned short code);
void clearScreen();
void cursorHome();
void printScreen(char ch);
void setScreen(char com);
void setPin(char port, char pin, char state);
void enableKeypadInterrupt();
void enableTimeInterrupt();
void setLed(char led, char state);
void checkSwitch();
void nextLine();
void printAlarmOff();
void readSensor();
void showTime(int hour, int minute, int second);
void displayNum(int nu);
void printON();
void readSensors();
```

```
void showTime(int hour, int minute, int second);
void alternatives();
void threeStrikes();
void printAlreadyOFF();
void printAlreadyON();

uint16_t adcResult = 0;

void USARTInit() {

  UBRRL = 0xCF; //Set Baud rate
  UBRRH = 0x00;

  UCSRC |= (1<<UCSZ1) |(1 << UCSZ0) | (1<<URSEL); //Set frame format (char size 8)
  UCSRB = UCSRB | (1<<RXEN) | (1<<TXEN); //Enable receiver and transmitter
}

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
}

void main(void){
                DDRD = 0xFF;
                DDRB = 0xFF;
                DDRA = 0xEF;
                setScreen(0b00001111); // Screen ON
                setScreen(0b00111000); // Set function
                clearScreen();
                cursorHome();

                enableKeypadInterrupt();
                enableTimeInterrupt();

                USARTInit();

                while(1) {
                                DDRD = 0x00;
                                DDRD = 0xEF;
                                ADCSRA |= (1 << ADEN);

                                if(activated && !changingCode) {
                                                if (actByPin && timer < 2) {
                                                                setLed(yellow, 1);
                                                                printActivated();
                                                }
                                                if (timer >= 10){
                                                                setLed(yellow, 0);
                                                                timer = 0;
                                                                actByPin = 0;
```

```
                                                setLed(red, 1);
                                                printON();
                                        }
                                        else if(alarmTrigged) {
                                                setLed(red, 1);
                                                setLed(yellow,1);
                                                setLed(green, 1);
                                                clearScreen();
                                                cursorHome();
                                                _delay_ms(200);
                                                printAlarm();
                                                _delay_ms(200);
                                                showTime(h, min, sek);
                                                h, min, sek = 0;
                                                _delay_ms(2000);
                                                alarmTrigged = 0;

                                        }
                                        readSensors();
                                }
                                else if (deactivated && !changingCode) {
                                        deactivated = 0;
                                        alarmTrigged = 0;
                                        activated = 0;
                                        alarmReadySec = 0;
                                        timer = 0;
                                        clearScreen();
                                        cursorHome();
                                        _delay_ms(200);
                                        printAlarmOff();
                                        setLed(red, 0);
                                        setLed(yellow, 0);
                                        setLed(green, 1);
                                        _delay_ms(1000);
                                        setLed(green, 0);
                                        clearScreen();
                                        _delay_ms(100);
                                }
                        }

}
ISR(INT0_vect) { // Interrupts when there is data available from the keypad

                        cli();

                        DDRC = 0x00;
                        PORTC = 0x00;

                        unsigned short int code = PINC;
                        char letter = getButton(code);
                        _delay_ms(200);
                        if(letter == 'q') { // change code
                                        printOld();
                                        printExpectPin();
                                        nextLine();
                                        changingCode = 1;
                        } else if(letter == 'r') { // deactivate
                                        if(activated) {
                                                printExpectPin();
```

```
                        } else {
                                printAlreadyOFF();
                        }
                } else if(letter == 's') { // activate
                        if(!activated) {
                                printExpectPin();
                        } else {
                                printAlreadyON();
                        }
                } else {
                        insertCode(letter); // any of the numbers 0 to 9
                        if(charNbr == 4) {
                                charNbr = 0;
                                nextLine();
                                if(changingCode == 2) {
                                        changePin();
                                        changingCode = 0;
                                        clearScreen();
                                        cursorHome();
                                        _delay_ms(100);
                                        printChanged();
                                } else {
                                        alternatives();
                                }
                        }
                }

                sei();
}
void changePin() {
                for(unsigned int i = 0; i<=3; i++) {
                        pinCode[i] = pinCodeTry[i];
                }
}
void insertCode(char letter) {
                pinCodeTry[charNbr] = letter;
                printScreen(letter);
                charNbr++;
}
void alternatives() { // Investigates if the alarm should be activated or deactivated
                if(changingCode == 1) {
                        if(compare()) {
                                clearScreen();
                                cursorHome();
                                _delay_ms(50);
                                printNew();
                                printExpectPin();
                                nextLine();
                                changingCode = 2;
                        } else {
                                printWrongPin();
                                changingCode = 0;
                                _delay_ms(1000);
                                clearScreen();
                                cursorHome();
                                _delay_ms(50);
                        }
                } else {
```

```
                        if(compare() && activated) { //If the PIN is correct and the alarm is
activated -> Deactivate
                                deactivated = 1;
                                activated = 0;
                                alarmReady = 0;
                        }
                        else if(compare() && !activated) { // If the PIN is correct and the alarm isn't
activated -> Activate
                                activated = 1;
                                actByPin = 1;
                        }
                        else {
                                printWrongPin();
                                _delay_ms(2000);
                                clearScreen();
                                cursorHome();
                                threeStrikes();
                        }
                }
}
void threeStrikes() { // Wrong PIN-code three times triggers the alarm
                if (activated) {
                        pinCounter++;
                }
                if (pinCounter == 3) {
                        alarmTrigged = 1;
                        pinCounter = 0;
                }
}
unsigned short int compare() {
                for(unsigned short int k = 0; k < 4; k++) {
                        if(pinCode[k] != pinCodeTry[k]) {
                                return 0;
                        }
                }
                return 1;
}
void printChanged() {
                printScreen('C');
                printScreen('h');
                printScreen('a');
                printScreen('n');
                printScreen('g');
                printScreen('e');
                printScreen('d');
                nextLine();
                printScreen('P');
                printScreen('I');
                printScreen('N');
                printScreen(' ');
                _delay_ms(1500);
                printScreen(' ');
                printScreen('O');
                printScreen('K');
                _delay_ms(1000);
                nextLine();
                clearScreen();
                cursorHome();
}
```

```
void printAlreadyOFF() {
                clearScreen();
                cursorHome();
                _delay_ms(50);
                printScreen('A');
                printScreen('l');
                printScreen('r');
                printScreen('e');
                printScreen('a');
                printScreen('d');
                printScreen('y');
                nextLine();
                printScreen('O');
                printScreen('F');
                printScreen('F');
                _delay_ms(1000);
                clearScreen();
                cursorHome();
                _delay_ms(50);
}
void printAlreadyON() {
                clearScreen();
                cursorHome();
                _delay_ms(50);
                printScreen('A');
                printScreen('l');
                printScreen('r');
                printScreen('e');
                printScreen('a');
                printScreen('d');
                printScreen('y');
                nextLine();
                printScreen('O');
                printScreen('N');
                nextLine();
                _delay_ms(1000);
                clearScreen();
                cursorHome();
                _delay_ms(50);
}
void printOld() {
                printScreen('O');
                printScreen('L');
                printScreen('D');
}
void printNew() {
                printScreen('N');
                printScreen('E');
                printScreen('W');

}
void printWrongPin() {
                printScreen('W');
                printScreen('r');
                printScreen('o');
                printScreen('n');
                printScreen('g');
                printScreen('P');
                printScreen('I');
```

```
		printScreen('N');
		nextLine();
}
void printON() {
		clearScreen();
		cursorHome();
		_delay_ms(200);
		printScreen('A');
		printScreen('l');
		printScreen('a');
		printScreen('r');
		printScreen('m');
		printScreen('O');
		printScreen('N');
		_delay_ms(2000);
		nextLine();
		clearScreen();
		cursorHome();
}
void printExpectPin() {
		printScreen('P');
		printScreen('I');
		printScreen('N');
		printScreen(':');
}
void printAlarm() {
		nextLine();
		clearScreen();
		cursorHome();
		_delay_ms(100);
		printScreen('*');
		printScreen('A');
		printScreen('L');
		printScreen('A');
		printScreen('R');
		printScreen('M');
		printScreen('!');
		printScreen('*');
		_delay_ms(2000);
		nextLine();
		_delay_ms(50);
}
void printAlarmOff(){
		printScreen('A');
		printScreen('l');
		printScreen('a');
		printScreen('r');
		printScreen('m');
		printScreen('O');
		printScreen('F');
		printScreen('F');
		_delay_ms(2000);
		nextLine();
}
void printActivated() {
		printScreen('A');
		printScreen('C');
		printScreen('T');
		printScreen('I');
```

```
		printScreen('V');
		printScreen('A');
		printScreen('T');
		printScreen('E');
		_delay_ms(2000);
		nextLine();
}
uint16_t readADC() {
		ADMUX |= (1 << MUX2);  // Set ADC Channel to 4 (Used by the IR Detector)
		ADCSRA |= (1 << ADSC); // Starts a single conversion
		while(!(ADCSRA & (1 << ADIF))); // Wait until the conversion is finished
		ADCSRA |= (1 << ADIF);
		return (ADC);
}
ISR(TIMER0_OVF_vect) { // Interrupts every 1/33 second
		count ++;
		if(count == 33) {		// Enters this loop once every second
			if(activated) {
				sek++;
			}
			if(sek == 60) {
				min++;
				sek = 0;
				if(min == 60) {
					h++;
					min = 0;
					if(h == 24) {
						h = 0;
					}
				}
			}
			if(adcResult > 35000) { // When voltage > 3,5V the timer starts counting up
				irTimer++;
			} else {
				irTimer = 0;
			}
			if(irTimer >= 1) { // When the timer exceeds 1 second the alarm is triggerd
				alarmReady = 1;
				irTimer = 0;
			}
			if(alarmReady) {
				setLed(yellow, 1);
				alarmReadySec++;
				if(alarmReadySec < 2) {
					printActivated();
					printExpectPin();
				}
				if(UCSRA & (1 << RXC)) {
					char data = usartRead();
					insertCode(data);
					if(charNbr == 4) {
						charNbr = 0;
						nextLine();
						alternatives();
					}
				}
				if(alarmReadySec >= 10) {
					alarmReady = 0;
					alarmTrigged = 1;
```

```
                                                alarmReadySec = 0;
                                                expectPin = 0;
                                        }
                                }
                                if(actByPin) {
                                        timer++;
                                }
                                count = 0;
                        }
}
char getButton(unsigned short code) {

                switch(code) {

                case 0:
                                return '1';
                                break;

                case 128:
                                return '2';
                                break;

                case 64:
                                return '3';
                                break;

                case 2:
                                return '4';
                                break;

                case 130:
                                return '5';
                                break;

                case 66:
                                return '6';
                                break;

                case 1:
                                return '7';
                                break;

                case 129:
                                return '8';
                                break;

                case 65:
                                return '9';
                                break;

                case 131:
                                return '0';
                                break;

                case 194:
                                return 'q'; //change code
                                break;

                case 193:
```

```
                        clearScreen();
                        cursorHome();
                        return 'r'; // deactivate
                        break;

                case 195:
                        clearScreen();
                        cursorHome();
                        return 's'; // activate
                        break;
                }
                return '-';
}
void clearScreen() {
                setScreen(0b00000001);
}
void cursorHome() {
                setScreen(0b00000010);
}
void nextLine() {
                setScreen(0b11000000);
                usartWrite(0b00001010);
                usartWrite(0b00001101);
}
void printScreen(char ch) { /
                usartWrite(ch);
                setPin('D', PD5, 0); // R/W = 0
                setPin('D', PD6, 1); // RS = 1
                setPin('D', PD7, 1); // E = 1
                PORTB = ch;                                     // Sets the data
                setPin('D', PD7, 0); // E = 0
                setPin('D', PD7, 1); // E = 1
}
void setScreen(char com) {
                setPin('D', PD5, 0); // R/W = 0
                setPin('D', PD6, 0); // RS = 0
                setPin('D', PD7, 1); // E = 1
                PORTB = com;                    // Sets the data
                setPin('D', PD7, 0); // E = 0
                setPin('D', PD7, 1); // E = 1
}
void setPin(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 enableKeypadInterrupt() {
                MCUCR = (1 << ISC01) | (1 << ISC00);
                SREG = 0x82;
                GICR = (1 << INT0);
                sei();
}
void enableTimeInterrupt() {
                TCCR0 |= (1 << CS02) | (1 << CS00); // Prescaler factor 1024
                TIMSK |= (1 << TOIE0); // Enable overflow interrupt
                TCNT0 = 0;
                count = 0;

                ADMUX = (1 << ADLAR) | (1 << REFS0) | (1 << MUX2);
                ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
                GICR = 1 << INT0; // Enable INT0
                MCUCR = 1 << ISC01 | 1 << ISC00; // Trigger INT0 on rising edge
                sei();
}
void setLed(char led, char state) {
                setPin('A', led, state);
}
void readSensors(){
                char sensor1;
                adcResult = readADC(); // reads the voltage from the IR detector
                sensor1 = PIND;
                sensor1 &= 0x10;
                if(sensor1 == 0) {
                                alarmReady = 1;
                }
}
void showTime(int hour, int minute, int second){
                displayNum(hour);
                printScreen(':');
                displayNum(minute);
```

```
                printScreen(':');
                displayNum(second);
                nextLine();
}
void displayNum(int nu){
                int num1=nu/10;
                itoa(num1,timeVect,10);
                printScreen(timeVect[0]);
                int num2=nu%10;
                num1=itoa(num2,timeVect,10);
                printScreen(timeVect[0]);
}
```

8 Referenser

- Kernighan & Ritchie. *The C Programming Language*. Bell Telephone Laboratories, Incorporated. New Jersey, 1988.
- Datablad LCD-display: http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf
- Datablad ATmega16: http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf
- Datablad Keypad Encoder: http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/Other/MM54C922.pdf
- Datablad Max233: http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/Communication/max232-233.pdf
- Datablad IR-sensor. *Passive IR Detector (PID11),* Siemens. (Från Bertil Lindvall)