LARMANLÄGGNING

Digitala Projekt, EITF11

Oskar von Knorring
Emin Karimov
Henrik Akej
Handledare: Bertil Lindvall

1. Sammanfattning

Vi har byggt ett larm vars syfte är att användas i hemmet. Larmet använder sig av rörelsedetektorer, en knappsats, en display och två LED-lampor, en grön och en röd. Larmet har programmerats i C. Vid utlöst larm skickas en signal till en dator.

Innehållsförteckning

2. Inledning

Projektet består av att bygga och programmera ett fungerande hemlarm. Larmet kommer använda sig av rörelsedetektorer, en knappsats för att styra larmet och en display för att förmedla information. Programmeringen görs i språket C.

3. Kravspecifikation

1. Den skall kunna hantera insignaler ifrån två IR-sensorer för övervakning av rörelser.
2. Vid utlöst larm skall en röd lysdiod blinka.
3. Det ska vara möjligt att manövrera med en numerisk knappsats.
4. Möjlighet att kommunicera med annan dator skall finnas.
5. För att aktivera/avaktivera larmet skall man ange en fyrsiffrig PIN-kod.
6. Systemet skall vara kopplat till en LCD-display.
7. Är larmet aktiverat skall en röd lysdiod lysa och displayen skall visa ”LARMAT”.
8. Är larmet avaktiverat skall en grön lysdiod lysa och displayen skall visa ”AVLARMAT”.

4. Kopplingsschema

Kopplingsschemat är framtaget med hjälp av PowerLogic 5.0, och hittas i bilaga 1.

5. Hårdvara

5.1 Processor

Processorn vi har använt oss av är en AVR ATmega16. Den är på åtta bitar, kan vid 16 MHz utföra upp till 16 MIPS, och har ett inbyggt flashminne på 16 kb. Processorn är kopplad till en JTAG-enhet för att kunna programmeras.

5.2 Display

För att kunna kommunicera med användaren har vi har använt en alfanumerisk display, SHARP Dot-Matrix. Den kan skriva ut två rader med respektive 16 tecken.

5.3 Knappsats

Knappsatsen har fyra rader och fyra kolumner med knappar 0-9 och A-F. Den är kopplad till processorn via en encoder.

5.4 Encoder

På grund av brist på pins på processorn har vi kopplat in en 16-knappars encoder, MM74C922. Tack vare denna minskas antalet pins till processorn från åtta till fyra stycken.

5.5 LED-lampor

För att enkelt visa när larmet är på respektive av har vi använt oss av två stycken LED-lampor, en röd och en grön. Dessa är kopplade till processorn via resistorer på 330 ohm.

5.6 Rörelsedetektorer

Vi har använt oss av två stycken PIR-Sensor (#555–28027) för att detektera rörelse. De utgör ett infrarött ljus och har en räckvidd på cirka 9 m. Dessa är kopplade direkt till processorn.

5.7 Seriell kommunikation

För att kunna kommunicera med en dator har vi använt oss av en serial, MAX233. Denna är kopplad mellan processorn och en PC serial port.

6. Mjukvara

Mjukvaran har kodats i språket C med hjälp av programmet Atmel Studio 6.1. Systemet går konstant igenom en while-loop där systemet lyssnar på input från knappsatsen, samt rörelsedetektorerna om larmet är aktiverat. För att kommunicera med datorn programmerar vi processorns USART (Universal Synchronous and Asynchronous serial Reciever and Transmitter).

7. Konstruktionsprocess

Vi började med att utvärdera vilka komponenter som krävdes för att uppnå våra krav. Därefter ritade vi upp ett kopplingsschema med hjälp av datablad för de olika komponenterna. När kopplingsschemat var färdigt lödde vi fast komponenterna på kopplingsplattan och därefter kopplade vi ihop de olika komponenterna.

När allting var färdigkopplat testade vi så att allt var korrekt inkopplat. Vi upptäckte lite problem med strömmen, samt att vi inte färdigkopplat displayen. När detta var åtgärdat började vi koda.

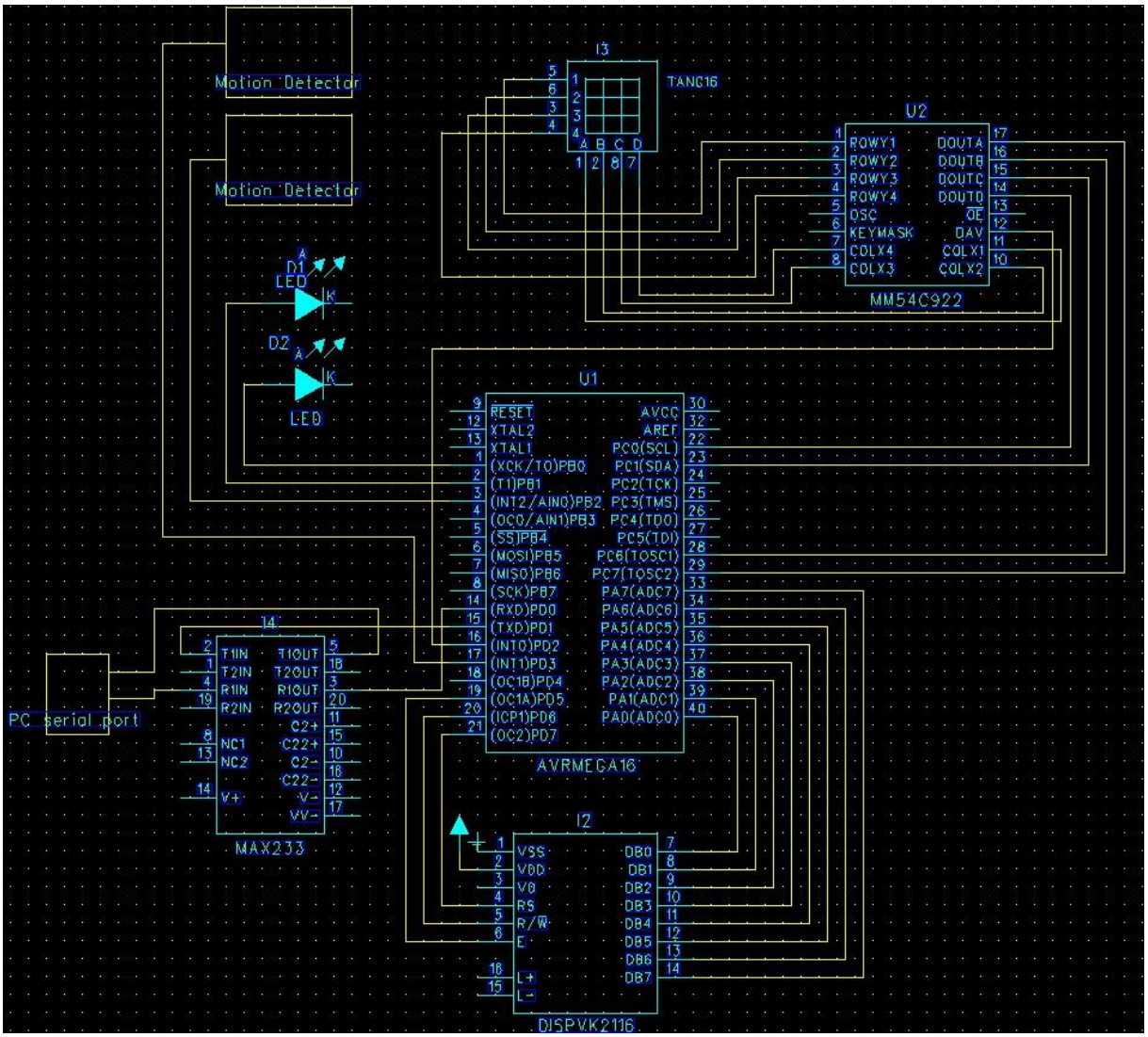
Den största utmaningen var att skapa en metod som byter värde på en enskild pin. Detta löstes med hjälp av bitvis operationer. Ett annat krångligt moment var att ställa in USART:en. Men med utförlig forskning i processor handboken lyckades vi lösa detta. Det sista stora problemet i koden var att få knappsatsen att fungera korrekt. Med användandet av en interrupt-funktion och encoderns inbyggda data-available funktion löstes detta utan större problem.

När koden var korrekt och larmet fungerade som det skulle byggde vi hemsidan.

8. Slutsats

Larmet uppfyller de funktioner som vi ville ha då vi startade projektet. Det är fullt möjligt att använda detta som ett hemlarm, men larmet är primitivt på många olika sätt. PIN-koden bestäms i mjukvaran, och går därför ej att ändra manuellt via terminalen. Knappsatsen fungerar inte optimalt då den vid vissa tillfällen skriver en siffra dubbelt. I övrigt är där inga märkbara problem med larmet.

Bilaga 1 – Kopplingsschema



Bilaga 2 – C Kod

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

int larm_status; //0 = off, 1 = on, 2 = utlöst
int pincode[4] = {4, 5, 6, 7};
int pressed_keys[4] = {0, 0, 0, 0};

int main(void) {

            disp_start();
            alarmOff();
            USART_Init();
            sei();
            INT_Init();

            while(1) {
                        while (larm_status == 2) {
                                    _delay_ms(200);
                                    redLED_ON();
                                    _delay_ms(200);
                                    redLED_OFF();
                        }
    }
}

void change_pin(char port, char pin, char val) { //port = A-D, pin = 0-7, val = 1/0
            char temp = 1 << pin; //Skapar en temp-variabel med 1 på samma plats som
pin
            if (port == 'A') {
                         temp &= PORTA; //Multiplicerar temp med PORTA
                         if (temp && !val) {
                                     PORTA ^= temp; //Ändrar från 1 -> 0
                         }
                         if (temp == 0 && val) {
                                     temp = 1 << pin; //Återger temp på ursprungliga
formen
                                     PORTA ^= temp; //Ändrar från 0 -> 1
                         }
            } else if (port == 'B') {
                        temp &= PORTB; //Multiplicerar temp med PORTB
                        if (temp && !val) {
                                    PORTB ^= temp; //Ändrar från 1 -> 0
                        }
                        if (temp == 0 && val) {
                                    temp = 1 << pin; //Återger temp på ursprungliga
formen
                                    PORTB ^= temp; //Ändrar från 0 -> 1
                        }
            } else if (port == 'C') {
                        temp &= PORTC; //Multiplicerar temp med PORTC
                        if (temp && !val) {
                                    PORTC ^= temp; //Ändrar från 1 -> 0
                        }
                        if (temp == 0 && val) {
                                    temp = 1 << pin; //Återger temp på ursprungliga
formen
                                    PORTC ^= temp; //Ändrar från 0 -> 1
                        }
            } else if (port == 'D') {
```

```
                        temp &= PORTD; //Multiplicerar temp med PORTD
                        if (temp && !val) {
                                PORTD ^= temp; //Ändrar från 1 -> 0
                        }
                        if (temp == 0 && val) {
                                temp = 1 << pin; //Återger temp på ursprungliga
formen
                                PORTD ^= temp; //Ändrar från 0 -> 1
                        }
                }
}

//Interrupt

INT_Init() {
                MCUCR = 0x0F;
                GICR = 0xE0;
                MCUCSR = 0x40;
}

ISR(INT0_vect) {
                pin_entry(keyset_pressed());
}

ISR(INT1_vect) { //Övre
                if (larm_status == 1) {
                        alarmActivated();
                }
}

ISR(INT2_vect) { //Undre
                if (larm_status == 1) {
                        alarmActivated();
                }
}

//Display

void disp_start() {
                DDRA = 0xFF; //Öppnar PORTA
                DDRB = 0x03; //Öppnar PORTB0-1, stänger resten
                DDRC = 0x00; //Stänger PORTC
                DDRD = 0xE2; //PD1, PD5-7 öppna, resten stängda
                PORTD = 0x20; //E hög
                PORTA = 0x38; //Function set
                toggle_E();
                PORTA = 0x0F; //Display ON
                toggle_E();
}

void write_cmd(char ch) {
                _delay_ms(5);
                change_pin('D', 7, 0); //RS låg
                PORTD = 0x20; //E hög
                PORTA = ch;
                toggle_E();
}

void write_data(char ch) {
                _delay_ms(5);
                change_pin('D', 7, 1); //RS hög
                change_pin('D', 5, 1); //E hög
```

```
            PORTA = ch;
            toggle_E();
}

void write_data_nbr(int nbr) {
            _delay_ms(5);
            change_pin('D', 7, 1); //RS hög
            change_pin('D', 5, 1); //E hög
            PORTA = nbr;
            toggle_E();
}

void toggle_E() {
            change_pin('D', 5, 0); //E låg
            change_pin('D', 5, 1); //E hög
}

void print_larmat() {
            write_cmd(0x01);
            write_data('L');
            write_data('A');
            write_data('R');
            write_data('M');
            write_data('A');
            write_data('T');
}

void print_avlarmat() {
            write_cmd(0x01);
            write_data('A');
            write_data('V');
            write_data('L');
            write_data('A');
            write_data('R');
            write_data('M');
            write_data('A');
            write_data('T');
}

void print_activated() {
            write_cmd(0x01);
            write_data('L');
            write_data('A');
            write_data('R');
            write_data('M');
            write_data(' ');
            write_data('A');
            write_data('K');
            write_data('T');
            write_data('I');
            write_data('V');
            write_data('E');
            write_data('R');
            write_data('A');
            write_data('T');
}

//Knappsats

int keyset_pressed() {
            char val = PINC;
            val &= 0b11000011;
```

```
		if (val == 0xC0) {
			return 0;
		}
		if (val == 0x40) {
			return 1;
		}
		if (val == 0x80) {
			return 2;
		}
		if (val == 0x00) {
			return 3;
		}
		if (val == 0xC2) {
			return 4;
		}
		if (val == 0x42) {
			return 5;
		}
		if (val == 0x82) {
			return 6;
		}
		if (val == 0x02) {
			return 7;
		}
		if (val == 0xC1) {
			return 8;
		}
		if (val == 0x41) {
			return 9;
		}
		return 10;
}

void pin_entry(int n) {
		if (n == 10) {
			return;
		}
		write_data_nbr(n);
		for (int i = 0; i < 3; i++) {
			pressed_keys[i] = pressed_keys[i+1];
		}
		pressed_keys[3] = n;
		if (pressed_keys[0] == pincode[0] && pressed_keys[1] == pincode[1] &&
pressed_keys[2] == pincode[2] && pressed_keys[3] == pincode[3]) {
			correct_pin();
		}
}

void correct_pin() {
		if (larm_status == 0) {
			print_larmat();
			alarmOn();
		} else {
			print_avlarmat();
			alarmOff();
		}
}

//LED

void greenLED_ON() {
		change_pin('B', 1, 1);
```

```
}

void redLED_ON() {
            change_pin('B', 0, 1);
}

void greenLED_OFF() {
            change_pin('B', 1, 0);
}

void redLED_OFF() {
            change_pin('B', 0, 0);
}

void alarmOn() {
            greenLED_OFF();
            redLED_ON();
            larm_status = 1;
            print_larmat();
}

void alarmOff() {
            redLED_OFF();
            greenLED_ON();
            larm_status = 0;
            print_avlarmat();
}

void alarmActivated() {
            larm_status = 2;
            USART_Transmit();
            print_activated();
}

//Serial

void USART_Init() {
            UBRRH = 0b00000000;
            UBRRL = 0b00110011; //9600 bits/sec (51)
            UCSRB = 0b00001000; //Transmitter ON
}

void USART_Transmit() {
             while ( !(UCSRA & (1<<UDRE)) ) //Transmit buffer (UDR) not ready
                        ;
            UDR = 'A';
}
```