Digitala projekt EITF11

Larm Master

Handledare Bertil Lindvall
2014-05-14
Anton Lundqvist I14
Caroline Geelmuyden I14
Christoffer Åsberg I14

Innehållsförteckning

Inledning

Digitala projekt är en kurs där man som student ska få en bild av vad industriellt utvecklingsarbete innebär. Målet är man i grupper om två eller tre studenter ska få en fungerande prototyp som kan fortsätta utvecklas och som har all den essentiella dokumentationen. Kursen är uppdelad i en första del som består av ett antal föreläsningar och labbar. Dessa är till för att förbereda studenten inför del två som består av att bygga, konstruera och testa sin konstruktion.

Till följd av den höga stöldrisken på datorer runt om på campus valde gruppen att konstruera ett larmsystem som kan användas för att avvärja eventuella tjuvar. Larmsystemet har en rörelsesensor och en tryckknapp som illustrerar övervakning av dörrar. För att kunna manövrera prototypen används en knappsats och en LCD-display. Vid utlöst larm kommer konstruktionen att ge ifrån sig ljud- och ljussignaler i hopp om att det ska avskräcka tjuven. Tiden för utlöst larm lagras också.

Referenser

ref1 - Processor: ATmega16
ref2 - LCD-display: SHARP-Dot-Matrix
ref3 - IR-sensor: PIR-Sensor (#555–28027)
ref4 - Kristall: CMACKD 8MHZ
ref5 - Encoder: 16-knappar, MM54C922

Kravspecifikation

Användarfall

Aktivera larm

Larmsystemet aktiveras genom att trycka "A" på knappsatsen. Larmsystemet inväntar sedan på kod-inmatning som aktiverar larmet. Larmsystemet tillåter tre inkorrekta kod-inmatnings försök innan det utlöses automatiskt.

Avaktivera larm

Larmsystemet avaktiveras från aktiverat läge genom att trycka "D" på knappsatsen. Larmsystemet inväntar sedan på kod-inmatning som avaktiverar larmet.

Ändra kod

Larmsystemets aktiverings-/avaktiveringskod kan ändras genom att trycka "C" på knappsatsen. Vid start är larmsystemets kod *1234*. Larmsystemet tillåter tre inkorrekta kod-inmatnings försök innan det utlöses automatiskt.

Krav

1. Larmsystemet ska vara kopplad till en display som kan meddela om larmet är aktiverat, icke aktiverat samt tid för då larmet utlösts.
2. Tiden som visas då larmet aktiveras ska vid behov kunna kalibreras genom att ändra i mjukvaran.
3. Larmsystemet ska ha en röd lysdiod som lyser när larmet är aktiverat.
4. Larmsystemet ska ha en grön lysdiod som lyser när larmet inte är aktiverat.

5. Larmsystemet ska utlösa ett larm med hjälp av en IR-sensor som reagerar på rörelse och en knapp som reagerar på intryckning.
6. Larmsystemet ska ha en summer som sätts på och stängs av med en viss frekvens när larmet är utlöst.
7. Larmsystemet ska ha en röd lysdiod som blinkar med en viss frekvens när larmet är utlöst.
8. Larmsystemet ska ha en knappsats kopplad till en encoder.
9. Larmsystemet ska aktiveras genom att ange en fyrsiffrig kod.
10. Larmsystemet ska avaktiveras genom att ange en fyrsiffrig kod.
11. Larmsystemets aktiverings-/avaktiverings-kod ska kunna ändras.
12. Larmsystemet ska tillåta tre försök att slå in korrekt kod vid aktivering av larm och ändring av kod.

Hårdvara

Kopplingsschema

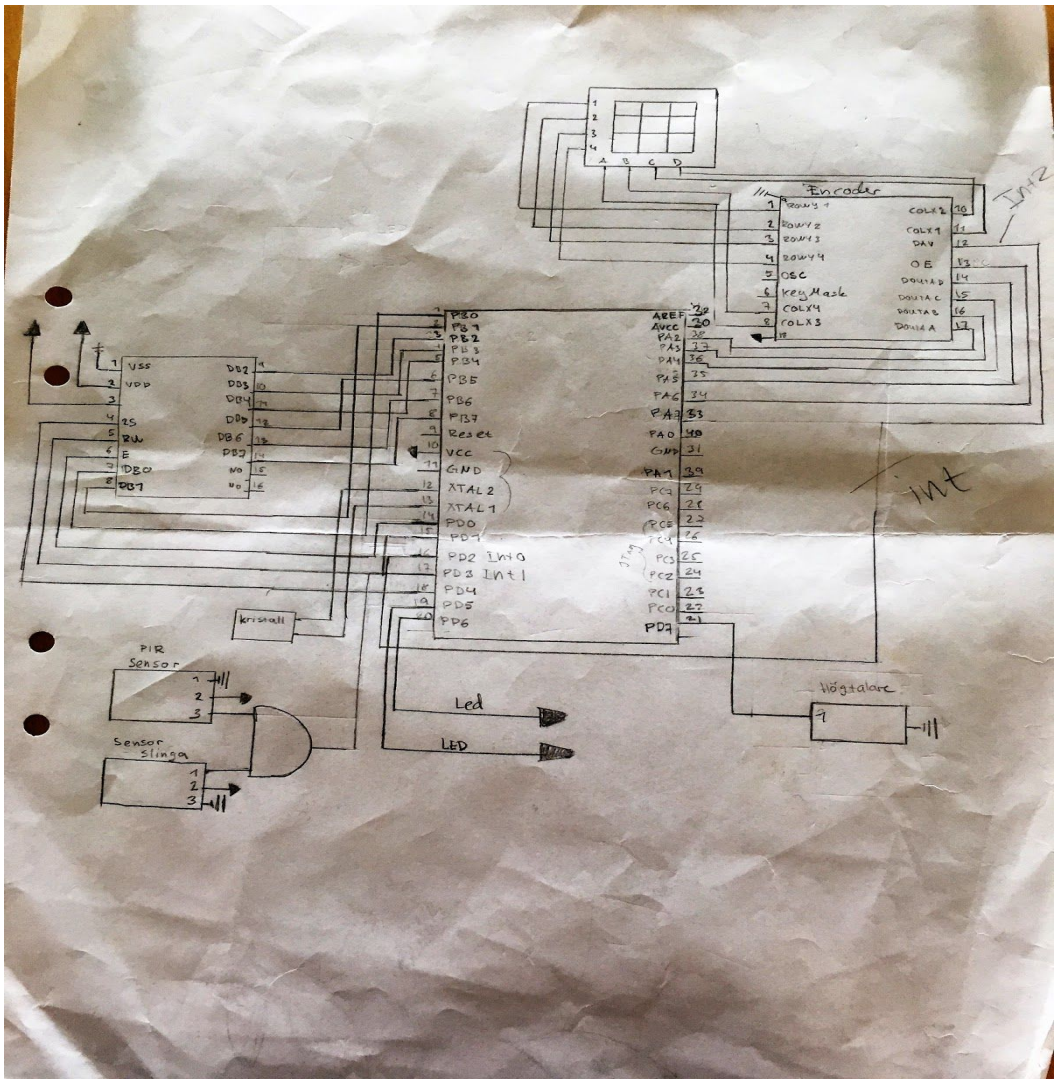

Bild 1. Ritning av kopplingsschemat för Larm Master

Processor

Som processor användes ATmega32 High-performance AVR 8-bit Microcontroller. Processorn har programmerats med hjälp av språket C i programmet AVR Studio 6. För att överföra programmeringskoden från datorn till processorn användes en JTAG. Se [ref1] för datablad.

LCD-skärm

Som display för systemet användes en SHARP-Dot-Matrix. Denna display har två rader med utrymme för 16 tecken på vardera rad. Dess funktioner i detta system är att meddela om larmet är på eller av, ge instruktioner till användaren, ge visuell återkoppling när koden skrivs in och visa tid för utlöst larm. Se [ref2] för datablad.

Knappsats

Som knappsats användes en enkel variant innehållandes siffrorna 1-9 och bokstäverna A-F. Siffrorna används för att skriva in pinkod medan bokstäverna används för att ge olika kommandon. Exempel på kommandon är aktivera larm, avaktivera larm och byta kod.

Tryckknapp

För att illustrera en dörr som öppnas används en tryckknapp. När man trycker ner knappen går signalen in till processorn från låg till hög och larmet utlöses.

IR-sensor

För att detektera rörelse i rummet används en PIR-Sensor (#555–28027). Sensorn känner av ändringarna i de infraröda nivåerna som avges av de omgivande objekten i rummet. När rörelse är detekterad sänder sensorn en hög signal in till processorn och larmet utlöses. Se [ref3] för datablad.

Lysdioder

För att ge användaren (och tjuven) en visuell bild av larmet status används två stycken lysdioder. Den ena dioden är grön och den andra är röd. När larmet är avaktiverat lyser den gröna dioden. När larmet är aktiverat lyser den röda och när larmet är utlöst blinkar den röda.

Summer

För att skrämma tjuven och ge omgivningen en notis om vad som pågår används en summer. När larmet utlöses ger summern ifrån sig en ljudlig signal.

Kristall

För att processorn ska få en mer precis klocka används en kristall. Kristallen är en CMACKD 8MHZ vilket innebär att den svänger åtta miljoner gånger per sekund. Se [ref4] för datablad.

Encoder

För att omvandla signalerna från knappsatsen till processorn används en 16-knappars encoder av modellen MM54C922. Se [ref5] för datablad.

Mjukvara

Mjukvaran till larmsystemet är konstruerad kring en main metod som består av en evig while-loop som endast bryts av tre stycken externa avbrott. Dessa avbrott sker då: knappsatsen används, IR-sensorn registrerar rörelse eller analogen switchen pressas ner, och efter ungefär varje sekund för att simulera tidsåtgång.

Den eviga while-loopen har fyra huvudsatser. Den första satsen kollar ifall larmets status är 2, som representerar ett utlöst larm, och ifall den är utlöst, anropas en metod som triggar summer, röd lysdiod etc. Resterande satser i while-loopen är för aktivering, avaktivering och ändring av kod. Dessa satser exekveras då avbrott från knappsatsen sker från val: "A", "D" eller "C".

Konstruktionsprocessen

Enligt kravspecifikationen valdes lämpliga komponenter till processorn. Komponenterna kopplades till varandra och processorn enligt beskrivningar i datablad.

För att minimera risken kring fel i hårdvaran senare i projektet testades varje komponent var för sig innan vi gick vidare. Detta gjordes med hjälp av logikpennan och funktionen i AVR Studio 6 där man manuellt kunde justera pinnarnas datariktning och läge.

När alla komponenter och kopplingar testats, påbörjades utvecklingen av själva mjukvaran för larmsystemet. Mjukvaran utvecklades som tidigare nämnt i avsnitt *Mjukvara* kring en while-loop som koden skrivs efter. Detta testades kontinuerligt med hjälp av debuggern tills en färdig produkt var klarställd.

Diskussion och slutsats

Arbetet med projektet har varit som en berg- och dalbana. Inför varje ny del i konstruktionsprocessen kände sig gruppen vilsen och oförmögen att ta sig vidare på egen hand. När man med hjälp på traven från handledare lyckades ta sig över den första tröskeln flöt dock arbetet på i en sakta mak. Med noggrann granskning av diverse datablad har gruppen gemensamt kunnat klura ut hur kopplingsschemat skulle ritas, sladdarna kopplas och hur programmeringen skulle skrivas. Ett av våra stora problem har varit att få de externa interrupten att fungera som de ska. Detta tog oss ända in i slutet att få rätt på, men med en säker hand och ett icke trytande tålamod rodde vi Larm Master i land.

En av svårigheterna med projektet var att behålla en kontinuitet i arbetet då det ofta endast var två av tre gruppmedlemmar som jobbade samtidigt. Vi är dock glada att vi varit tre i gruppen vilket lett till att varje gruppmedlem kunnat ha ett fokus på en speciell del, till exempel läsa datablad eller vara kreativ med koden.

Vi är nöjda med vår prototyp och arbetet generellt. Det har gett oss en insikt i hur denna typen av arbete kan gå till i industrin och det har även gett oss en respekt för all den avancerade teknik som omger oss i vardagen.

Källkod

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define red_diode PD6
#define green_diode PD5
#define E PD0
#define RW PD1
#define RS PD4
unsigned int password = 1;
unsigned int count = 0;
unsigned int larm_status = 0;
unsigned int alarmactive = 0;
unsigned int data_available;
unsigned int pinCounter = 0;
unsigned char pinCode[4] = {'1', '2', '3', '4'};
unsigned char pinCodeTry[4];
unsigned char val;
unsigned int charnbr = 0;
unsigned int new_input = 0;
unsigned int sec = 0;
unsigned int min = 15;
unsigned int hour = 10;
unsigned int clock = 49911;
int timevect[8];

                char read_keypad(char code){
                        if(code == 0b00000000){
                                return '0';
                                }else if(code==0b00000100){
                                return '1';
                                }else if(code==0b00001000){
                                return '2';
                                }else if(code==0b00001100){
                                return '3';
                                }else if(code==0b00010000){
                                return '4';
                                }else if(code==0b00010100){
                                return '5';
                                }else if(code==0b00011000){
                                return'6';
                                }else if(code==0b00011100){
                                return '7';
                                }else if(code==0b00100000){
```

```
                                return '8';
                                }else if(code==0b00100100){
                                return '9';
                                }else if(code==0b00101000){
                                return 'A';
                                }else if(code==0b00101100){
                                return 'B';
                                }else if(code==0b00110000){
                                return 'C';
                                }else if(code==0b00110100){
                                return 'D';
                                }else if(code==0b00111000){
                                return 'E';
                                }else if(code==0b00111100){
                                return 'F';
                        }

                }
int main(void)
{
        INIT();
        _delay_ms(5);
        val=0xff;
        _delay_ms(5);
        start_Display();
        _delay_ms(5);
        set_Display();
        _delay_ms(5);
        clear_display();
        _delay_ms(5);
        write_welcome();

        while (1)
        {


                if(larm_status == 2){
                        alarmtriggered();
                }

                while(new_input == 1){
                        char input = read_keypad(val);


                        if(input == 'A' ){
```

```
        int pincount = 0;
        while(pincount < 3){
                clear_display();
                write_activatelarm();
                new_input = 0;
                for(int k =0; k<4; k++){

                        while(new_input == 0){
                        }

                        char pin = read_keypad(val);
                        pinCodeTry[k] = pin;
                        write_char('*');
                        new_input = 0;
                }
                int wrong = 0;
                for(int i=0; i<4;i++){
                        if(pinCodeTry[i] !=pinCode[i]){
                        wrong = 1;
                        }
                }
                if(wrong == 0){
                        clear_display();
                        alarm_ON();
                        new_input = 0;
                        pincount = 3;
                } else {
                        pincount++;
                        write_wrongcode();
                        if(pincount == 3){
                                clear_display();
                                larm_status = 2;
                                new_input =0;
                        }
                }
        }
}
if(input == 'D'){
        int pincount = 0;
        while(pincount < 3){
                clear_display();
                write_deactivatelarm();
                new_input = 0;
                for(int k =0; k<4; k++){

                                while(new_input == 0){
```

```
                                        }

                                        char pin = read_keypad(val);
                                        pinCodeTry[k] = pin;
                                        write_char('*');
                                new_input = 0;
                                }
                                int wrong = 0;
                                for(int i=0; i<4;i++){
                                        if(pinCodeTry[i] !=pinCode[i]){
                                                wrong = 1;
                                        }
                                }
                                if(wrong == 0){
                                        clear_display();
                                        alarm_OFF();
                                        new_input = 0;
                                        pincount = 3;
                                        } else {
                                        pincount++;
                                        write_wrongcode();
                                        _delay_ms(5000);
                                        if(pincount == 3){
                                                clear_display();
                                                larm_status = 2;
                                                new_input =0;
                                        }
                                }
                        }
                }

                if(input == 'C'){
                        int pincount = 0;
                        while(pincount < 3){
                        new_input = 0;
                                clear_display();
                                write_oldcode();
                                for(int k =0; k<4; k++){

                                                while(new_input == 0){
                                                }

                                        char pin = read_keypad(val);
                                        pinCodeTry[k] = pin;
                                        write_char('*');
                                        new_input = 0;
```

```
                                        }
                                        int wrong = 0;

                                        for(int i=0; i<4;i++){
                                                if(pinCodeTry[i] !=pinCode[i]){
                                                        wrong = 1;
                                                }
                                        }
                                        if(wrong == 0){
                                                clear_display();
                                                write_newcode();
                                                new_input = 0;
                                                for(int i=0;i<4;i++){

                                                        while(new_input == 0){
                                                        }


                                                        char newPin = read_keypad(val);
                                                        pinCode[i] = newPin;
                                                        write_char(newPin);
                                                        new_input = 0;
                                                }
                                                new_input = 0;
                                                pincount = 3;
                                                clear_display();
                                        } else {
                                                pincount++;
                                                write_wrongcode();
                                                _delay_ms(5000);
                                                if(pincount == 3){
                                                        clear_display();
                                                        larm_status = 2;
                                                        new_input =0;
                                                }
                                        }
                                }
                        }
                }
        }
}

        ISR(TIMER1_OVF_vect) {
                        sec++;
                        if(sec == 60){
                                min++;
```

```
                        sec = 0;
                        if(min == 60){
                                hour++;
                                min = 0;
                                if(hour == 24){
                                        hour = 0;
                                }
                        }
                }
        }

        ISR(INT0_vect){
                val = PINA & 0b00111100;
                new_input = 1;
        }

        ISR(INT1_vect){
                if(larm_status==1){
                        larm_status = 2;
                }
        }




        void INIT(){ // Sätter initialtillstånden
                DDRB = 0b11111111; //vi sätter databussen ut
                DDRA = 0b00000000; //vi sätter a-porten in
                DDRD = 0b11110011; //vi sätter RS. RW, E ut
                DDRC = 0b00000000; // C-porten in
                PORTD = 0b00001100; //vi sätter RS till 0 och E till 0 även interupt 1
                GICR = 0b11000000;
                MCUCR = 0b0001111;
                TCCR1B =0b00000101; //inställning för klockan (vet ej om bit 6 ska vara 0
eller 1)
                TCNT1 = clock;
                TIMSK = 0b00000100; // Ska sätta hur ofta interuptet skjer, hur??
                sei();


        }

        void display_int(int time){
                int time1 = time/10;
                itoa(time1, timevect, 10);
                write_char(timevect[0]);
```

```
	int time2 = time%10;
	time1 = itoa(time2, timevect, 10);
	write_char(timevect[0]);
}

void showTime(int hour, int min, int sec){
	display_int(hour);
	write_char(':');
	display_int(min);
	write_char(':');
	display_int(sec);

}

void clear_display(){
	_delay_ms(10);

	PORTB = 0b00000001;
	PORTD |= _BV(E);
	PORTD &= ~_BV(E);
	_delay_ms(10);
}
void change_curser(){
	char location = 0x40;
	if(location < 0b01111111){
	PORTB = 0b10000000 | location;
	PORTD |= _BV(E);
	PORTD &= ~_BV(E);
	}
	}

void set_Display(){
	PORTB = 0b00111100; // Ska sista två vara nollor? - i datablad
	PORTD |= _BV(E);
	PORTD &= ~_BV(E);
}

void start_Display(){ // Används även denna för off? Behövs off?
	PORTB = 0b00001111;
	PORTD |= _BV(E); // Sätter E till 1
	PORTD &= ~_BV(E); // Sätter E till 0
}

void write_char(char ch){
	_delay_ms(5);
	PORTD |= _BV(RS); // Sätter RS hög
```

```
	_delay_ms(5);
	PORTB = ch;
	_delay_ms(5);
	PORTD |= _BV(E); // Sätter E hög
	_delay_ms(5);
	PORTD &= ~_BV(E); // Sätter E låg
	_delay_ms(5);
	PORTD &= ~_BV(RS); // Sätter rs låg
}

void write_int(int i){
	_delay_ms(5);
	PORTD |= _BV(RS); // Sätter RS hög
	_delay_ms(5);
	PORTB = i;
	_delay_ms(5);
	PORTD |= _BV(E); // Sätter E hög
	_delay_ms(5);
	PORTD &= ~_BV(E); // Sätter E låg
	_delay_ms(5);
	PORTD &= ~_BV(RS); // Sätter rs låg
}

void write_cmd(char cmd){
	_delay_ms(5);
	PORTB = cmd;
	PORTD &= ~ _BV(PD4); // Sätter RS låg
	PORTD &= ~ _BV(PD0); // Sätter E låg
	PORTD |= _BV(PD4); // Sätter RS hög
	PORTD |= _BV(PD0); // Sätter E hög
}

void write_AlarmON(){
	clear_display();
	_delay_ms(5);
	write_char('A');
	write_char('L');
	write_char('A');
	write_char('R');
	write_char('M');
	write_char(' ');
	write_char('O');
	write_char('N');
}

void write_AlarmOFF(){
```

```
	clear_display();
	_delay_ms(5);
	write_char('A');
	write_char('L');
	write_char('A');
	write_char('R');
	write_char('M');
	write_char(' ');
	write_char('O');
	write_char('F');
	write_char('F');
}

void write_wrongcode(){
	clear_display();
	_delay_ms(5);
	write_char('W');
	write_char('R');
	write_char('O');
	write_char('N');
	write_char('G');
	write_char(' ');
	write_char('C');
	write_char('O');
	write_char('D');
	write_char('E');
}



	void write_oldcode(){
		clear_display();
		_delay_ms(5);
		write_char('O');
		write_char('L');
		write_char('D');
		write_char(' ');
		write_char('C');
		write_char('O');
		write_char('D');
		write_char('E');
		write_char(':');
		change_curser();
	}
	void write_newcode(){
		clear_display();
```

```
        _delay_ms(5);
        write_char('N');
        write_char('E');
        write_char('W');
        write_char(' ');
        write_char('C');
        write_char('O');
        write_char('D');
        write_char('E');
        write_char(':');
        change_curser();
}
void write_welcome(){
        clear_display();
        _delay_ms(5);
        write_char('W');
        write_char('E');
        write_char('L');
        write_char('C');
        write_char('O');
        write_char('M');
        write_char('E');
        write_char(' ');
        write_char('T');
        write_char('O');
        change_curser();
        write_char('L');
        write_char('A');
        write_char('R');
        write_char('M');
        write_char(' ');
        write_char('M');
        write_char('A');
        write_char('S');
        write_char('T');
        write_char('E');
        write_char('R');
}
void write_activatelarm(){
        clear_display();
        _delay_ms(5);
        write_char('A');
        write_char('C');
        write_char('T');
        write_char('I');
        write_char('V');
```

```
        write_char('A');
        write_char('T');
        write_char('E');
        write_char(' ');
        write_char('L');
        write_char('A');
        write_char('R');
        write_char('M');
        write_char(':');
        change_curser();
}
void write_deactivatelarm(){
        clear_display();
        _delay_ms(5);
        write_char('D');
        write_char('E');
        write_char('A');
        write_char('C');
        write_char('T');
        write_char('I');
        write_char('V');
        write_char('A');
        write_char('T');
        write_char('E');
        write_char(' ');
        write_char('L');
        write_char('A');
        write_char('R');
        write_char('M');
        write_char(':');
        change_curser();
}

void triggered_at(){
        clear_display();
        _delay_ms(5);
        write_char('T');
        write_char('R');
        write_char('I');
        write_char('G');
        write_char('G');
        write_char('E');
        write_char('R');
        write_char('E');
        write_char('D');
        write_char(' ');
```

```
                write_char('A');
                write_char('T');
                write_char(':');
                change_curser();
                _delay_ms(5);
        }

// LED
    void greenDiode_ON(){
        PORTD |= _BV(green_diode);
        PORTD &= ~_BV(PD0);
        PORTD |= _BV(PD0);
    }

        void redDiode_ON(){
            PORTD |= _BV(red_diode);
            PORTD &= ~_BV(PD0);
            PORTD |= _BV(PD0);
        }

        void greenDiode_OFF(){
            PORTD &= ~ _BV(green_diode);
            PORTD &= ~_BV(PD0);
            PORTD |= _BV(PD0);
        }

        void redDiode_OFF(){
            PORTD &= ~ _BV(red_diode);
            PORTD &= ~_BV(PD0);
            PORTD |= _BV(PD0);
        }
        void summer_ON(){
            PORTD |= _BV(PD7);
            PORTD &= ~_BV(E);
            PORTD |= _BV(E);
        }
        void summer_OFF(){
            PORTD &= ~ _BV(PD7);
            PORTD &= ~_BV(PD0);
            PORTD |= _BV(PD0);
            }

    void alarm_ON(){
        larm_status = 1;
        greenDiode_OFF();
        redDiode_ON();
```

```
        write_AlarmON();
}

void alarm_OFF(){
        redDiode_OFF();
        greenDiode_ON();

        larm_status = 0;
        summer_OFF();
        write_AlarmOFF();
}

void alarmtriggered(){
        clear_display();
        triggered_at();
        showTime(hour, min,sec);
while(read_keypad(val) != 'D'){
        summer_ON();
                redDiode_ON();
                _delay_ms(1000);
                summer_OFF();
        redDiode_OFF();
                _delay_ms(1000);
        }
        if(larm_status == 2){
                summer_ON();
                redDiode_ON();
        }
}
```

Referenser

Literatur

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