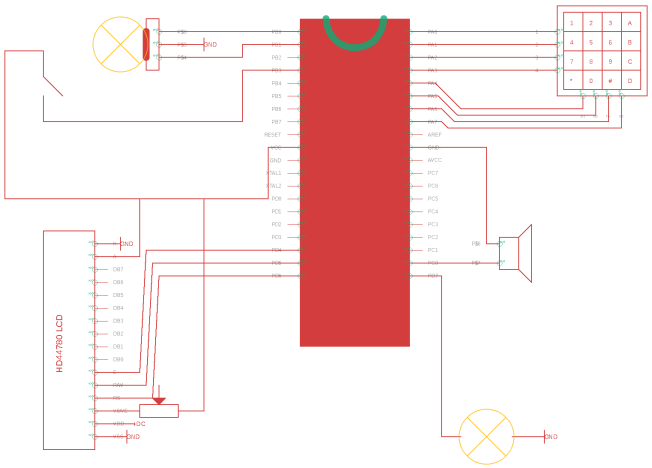


LARM

Projektrapport över larmkrets med dörrsensorer

Studenter:
Måns Alklint
Philip Johansson
Viktor Ahlén
Derin Shwan

Handledare:
Bertil Lindvall
Lars-Göran Larsson

Inledning

Bakgrund

Som projekt i kursen Digitala System, ingår delmomentet projekt som skall indela klassen i grupper om 4 och ge grupperna möjligheten att fördjupa sig och öva på att använda mikrokontrollern ATmega 1284 för att på eget bevåg, designa och konstruera en fungerande prototyp för en valfri krets.

Syfte

Syftet med projektet var att framställa en fungerande prototyp med en mikroprocessor, ATmega 1284, samt programmera den efter kravspecifikation. Detta innebär att konstruera en larmkrets som kan övervaka dörrsensorer, tolka användarinput från en knappsats och spela upp en ton när larmet utlöses.

Kravspecifikation

Syftet med kravspecifikationen är att bestämma vilka funktioner och komponenter larmet behandla enligt följande krav:

Funktioner:

- En hemlig kod som anges för att stänga av samt sätta på larmet.
    - Knappsatsen skall kunna hantera input från användaren, sådant att larmet kan larmas på och av och ställa klockan.
- En utsignal som består av en siren/ljudsignal.
    - En piezo-summer skall ge ifrån sig ljud när larmet triggas
- Larmet utlöses när en typ av sensor ger signal.
- Tiden då larmet utlösts skall visas på en display.
    - Displayen skall ge användaren de instruktioner som krävs för att administrera larmet.

Nyckelord

Larm, Krets, ATmega1284, mikrokontroller, AVR, JTAG, Knappsats, 8-bit, piezo, LCD

Keywords

Alarm, Circuit, ATmega1284, microcontroller, AVR, JTAG, Keypad, 8-bit, piezo, LCD

Terminologi

Nedan följer en lista på viktiga begrepp och dess betydelse i rapportens sammanhang:

Mikrokontroller

Mikrokontrollern i konstruktionen är en ATMEGA1284. ATMEGA1284 är en 8-bitars mikrokontroller med 44 pins varav 32 är I/O portar samt ett flashminne på 128KB.

LCD

Displayen som används i kretsen är en "Hitachi 44780 dot-matrix LCD" som kan användas med 8-bitars mikrokontroller. LCDn är bakgrundsbelyst och består av två á 28-tecken långa rader.

4x4 matrisknappsats

Knappsatsen som används är en icke-kodad matrisknappsats bestående av 16 tangenter parallellt. Tangenterna för varje rad och kolumn är anslutna genom 8 pins.

Piezo-summer

I de enklaste termerna är en piezo-summer en typ av elektronisk enhet som används för att producera en ton.

JTAG

En JTAG används för att felsöka och skriva över programmet till mikrokontrollern.

Vridpotentiometer

En vridpotentiometer (varierande resistor) är ansluten till LCDn för att kunna reglera kontrast.

Databladsreferenser

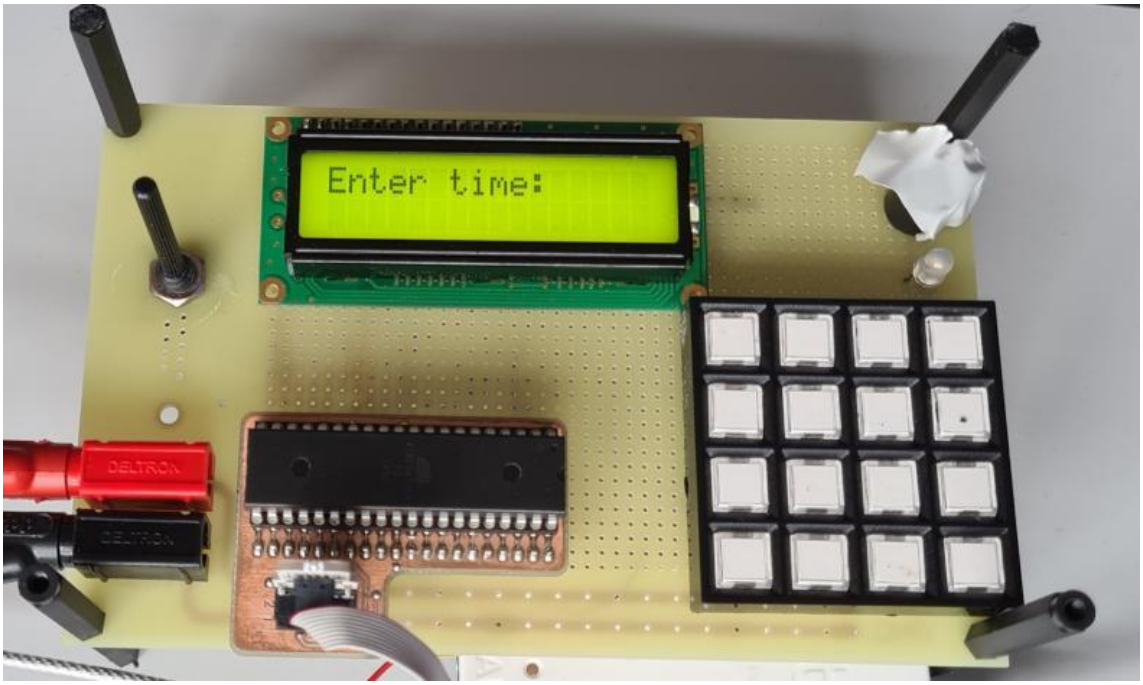
Hitachi 44780 LCD:
https://www.eit.lth.se/fileadmin/eit/courses/datablad/Display/hd44780.pdf

ATmega1284**:**
https://www.eit.lth.se/fileadmin/eit/courses/datablad/Processors/ATmega1284.pdf

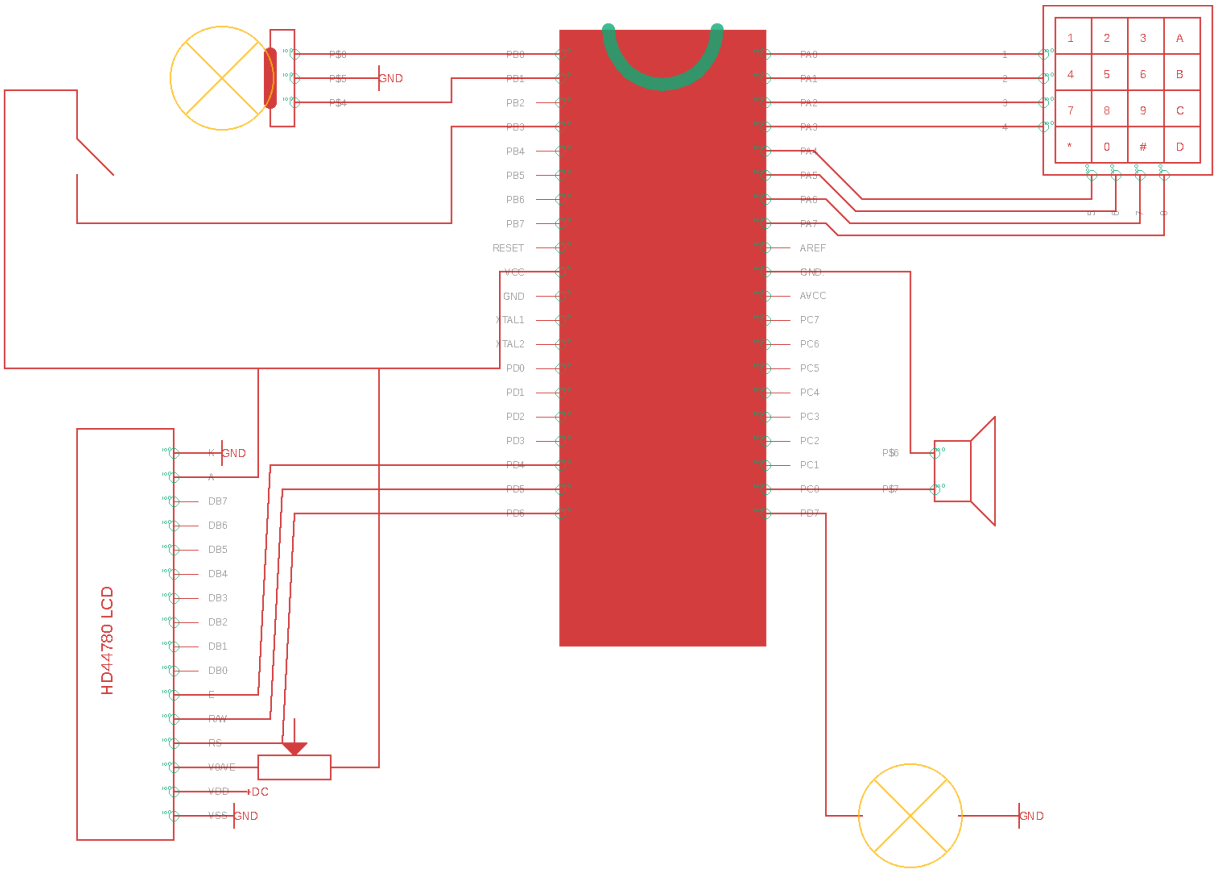
Grayhill 83BB1-001 4x4 matrix keypad:
https://www.masterelectronics.com/productfiles/mf-gh/keypads_83.pdf

Prototyp

Detta avsnitt innehåller larmkretsens kretsschema samt bild av prototyp.



Figur 1: Bild som visar larmkretsprototypen i sitt startläge.



Figur 2: Kretsschema över larmkretsprototypen.

Sammanfattning

I detta arbete har en larmkrets konstruerats med en dörrsensor (i form av en strömbrytare), en 4x4 matrisknappsats, LCD, piezo-summer samt lysdioder. För detta användes mikrokontrollern ATmega 1284 som programmerats för att kunna styra kretsen och få input samt output från dess respektive in- och utportar. Även ett detaljerat kretsschema designades, enligt arbetets instruktion för att förenkla själva konstruerandet av kretsen. Koden skrevs i språket C och omfattar kommunikation med LCD för att skriva ut text, läsa av knappsatsen där bland annat tid, larmets lösenkod samt av- och påsättning av larmet kan ske.

Abstract

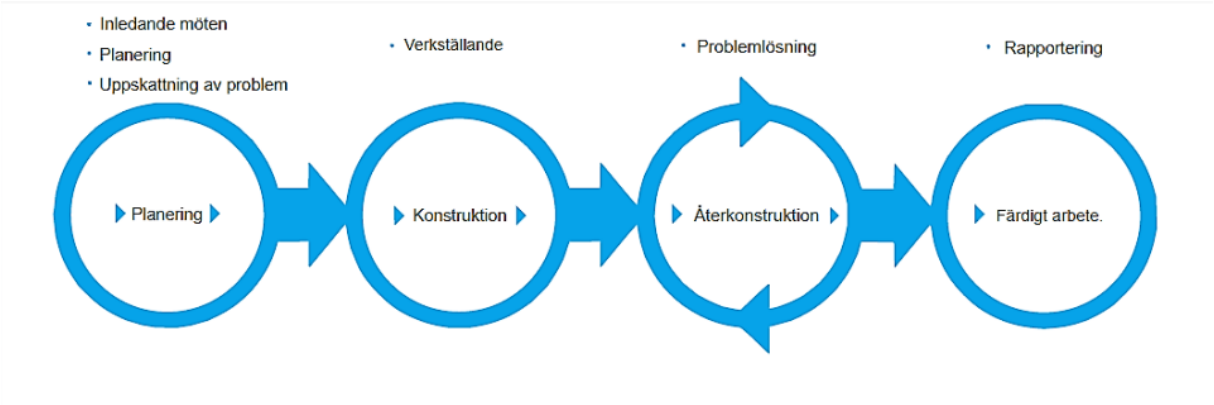
In this group assignment a home alarm system was created, using the ATmega 1284 microcontroller, electronic switches on doors, piezo summers, LCD, LEDs, and a keypad. It can show the last time since the alarm was last triggered and by using a LED, it can show the current alarm status. The alarm can also be turned on and off using a passcode that can be entered using the keypad.

Innehållsförteckning

Metod

Arbetet med examensarbetet har skett relativt planeringsändligt och ömsesidigt mellan gruppens fyra medlemmar. Det upplägg som detta arbete omfattar består i runda drag av:

1. Planering. När arbetsförberedelserna skedde, beslutades det samtidigt om vad för typ av krets som skulle konstrueras samt vad kretsens funktioner, i detta fall ett larm, skulle bestå av
2. Verkställande; då kretsen designades och byggdes enligt kravspecifikationen.
3. Problemlösning. Problem uppstår under konstruktion och dessa måste lösas.
4. Rapportering. Denna projektrapport skrevs för att beskriva arbetets process, kretsens funktion samt eftersom det är en obligatorisk del i momentet.



Figur 3: Bild som beskriver arbetsprocessens upplägg.

Denna arbetsprocess är en något förenklad rekonstruktion hur ingenjörsarbete normalt sker, eftersom upplägg och problemuppskattning är nödvändigt för att uppskatta arbetets svårighetsgrad och metod. Efter konstruktion följer nästan alltid ett problemlösningssteg eftersom problem ofta dyker upp längs arbetets förlopp.

Efter det första mötet har gruppens medlemmar haft en öppen kommunikation med varandra genom hela arbetsprocessen och har haft särskilt ansvar för vissa delar av arbetet. Det har funnits olika sätt för att kunna programmera en viss egenskap i larmkretsen men gruppens medlemmar enades att snarare försöka använda så mycket som möjligt av ATmega1284 inbyggda funktioner snarare än att använda flera separata komponenter. Information om hur vissa funktioner kunde programmeras fanns lättillgänglig på internet i form av datablad och bibliotek, så kallade “Libraries”, som innehåller en del färdiga standardrutiner för att göra omvandlingar mellan exempelvis textsträngar, tal eller karaktärer.

Kretsen designades först i Eagle av alla gruppens medlemmar efter det att kravspecifikationen var formulerad. Därefter har alla gruppens medlemmar ömsom kodat och ömsom kopplat komponenter, vilket har bestått av bland annat lödande eller enbart ett sätt att utprova kretsens funktioner innan fastlödning, som i då skett på ett så kallat ”BreadBoard”.

Kodandet av larmkretsen har alltså skett jämsides kopplandet av kretsens alla komponenter. Det första som lades upp var själva kodskelett som definierade funktioner. Funktioner såsom att kretsen skall spela ett ljud när larmet går definierades i detta steg, varpå pseudokod och kommentarer fick illustrera vad som skulle ske.

Därefter inledde gruppen med att se till att displayen fungerade. Efter att ha justerat strömstyrka och spänning, insåg gruppmedlemmarna att displayen behövde en resistor som beroende dess resistans, kunde definiera displayens kontrast.

Efter att displayen kunde visa klockslag och andra texter på ett korrekt sätt, programmerades matrisknappsatsen. Detta görs genom en tvådimensionell lista i koden som för varje rad har lika många kolumner. Genom en tvådimensionell lista var det även lättare att illustrera vilken knapp på knappsatsen som stod för vilket tecken.

Efter knappsatsen fungerade utvecklades tidtagningsfunktionen. Denna går ut på att kunna räkna antalet sekunder sedan då användaren utförde den första installationen av kretsen, bestående av att skriva in vad nuvarande exakta klockslag är på knappsatsen.

Efter det att larmkretsen hade tidtagningsmöjligheter var det relativt lätt att inse hur resterande kod skulle struktureras och indelas för att kretsen skall ge ifrån ljudsignal då larmet går, skriva ut klockslaget för detta ögonblick på skärmen, samt kunna avlarmas med en kod.

För detta programmerades kretsens förmåga att spela en ton på en piezo-summer samt en strömbrytare som i denna krets fall, illustrerar en sensor för en dörr. Mikroprocessorn programmerades in så att i de fall som denna strömbrytare påverkas så utlöses larmet.

Resultat

Resultatet av projektet är ett fungerande larm som programmerats rätt enligt kravspecifikation och gruppmedlemmarnas kunskap, som samtidigt fungerar tillräckligt stabilt för att vara fungerande prototyp.

Arbetsprocessen har alltså skett enligt upplägget och de avsteg från processen som skett har varit obetydliga och endast för att lösa smärre problem i källkoden när denna inte fungerat korrekt.

Larmkretsen startas genom att strömmen slås på och en tid ställs. Därefter utlöses larmet om dörrsensorn påverkas genom exempelvis beröring, varpå en ljudsignal spelas. Samtidigt skrivs tiden då larmet utlösts på displayen och därefter ges 3 försök att skriva in den rätta koden och således återställa larmet. Efter 3 felaktiga försök att skriva in koden hamnar kretsen i ett läge utan återvändo, där ljudsignalen fortsätter till dess att strömmen bryts.

Analys

Detta avsnitt hanterar arbetsförloppets problem och dess respektive lösningar.

Problem

De största utmaningarna har varit att få klockan att fungera utan separata komponenter och använda den interna oscillatorn, det vill säga ATmega1284s klockfrekvens. Detta behövdes för att kunna mäta tid utan en extern oscillator/kvartsklocka. Lösningen var att använda den interna oscillatorn i ATmega1284, det vill säga dess klockfrekvens på 16 MHz. Detta behövdes för att larmet skall kunna ange vilken tid det löst ut och kunna visa detta för användaren.

4x4-matrisknappsatsen var också svår att programmera, eftersom det knappt fanns någon information i databladen om vilka pins som skulle användas för att läsa av rader respektive kolumner. Lösningen var att mäta på knappsatsen och själva markera ut vilka pins som är rad respektive kolumn.

Vi har även mött motgångar som har med programspråket C att göra, eftersom C är ett idag gammalt språk där problem ofta uppstår i exempelvis hur omvandling mellan datatyper skall ske. Lyckligtvis har det funnits mycket dokumentation online som underlättat.

Dessutom har själva utvecklingsmiljön varit svårhanterlig, eftersom vi har använt oss av en gammal version av Atmel Studio (idag kallad Microchip studio[1]) där kompatibilitet och stabilitet inte varit helt felfritt när mikrokontrollern programmerats och felsökts. Lösningen var att använda en nyare version av programmet på en annan dator.

[1] (Microchip Technology Inc, n.d.)

Slutsats

Slutligen, lyckades gruppen konstruera en larmkrets enligt kravspecifikation med funktionalitet som dels härmar modernare sofistikerade larmsystem och som dels med väldigt få komponenter, är tillräckligt avancerad för att vara tillfredsställande som slutresultat i projektmomentet för kursen Digitala System. Samtidigt kan gruppens medlemmar poängtera att kretsen delvis låts bygga på konceptet avskräckning med hjälp av en larmsignal när en dörr öppnas och ett inbrott sker, eftersom den av oss bedöms alldeles för osäker för att kunna användas som primär källa för en anläggnings säkerhet.

Larmkretsens samhällsnytta

Denna krets fyller sitt syfte och skulle kunna användas i exempelvis bilar eller andra små anläggningar för att förhindra inbrott och stöld. En slutsats är att kretsen har en viss samhällsnytta även om den är långt lika säker som andra larmtyper, eftersom den endast har dörrsensorer.

Referenslista

[1] Microchip Technology Inc. (n.d.). *Microchip Studio for AVR® and SAM Devices*. Hämtad från Microchip Technology Inc-webbplats.

Referenskritik

Källan vi använt kommer från ett amerikanskt företag som tillverkar mikrokontroller, och integrerade kretsar och som idag står för den uppdaterade versionen av Atmel Studio. Vi bedömer därmed denna källa som trovärdig.

Appendix

Programmets källkod från filen main.c:

```
#define F_CPU 16000000UL

#define TIMER1_PRESCALER (uint8_t) 8

// #define SPEAKER_PORT PORTB
// #define SPEAKER_DDR DDRB
// #define SPEAKER_PIN 7

#define KEY_PRT       PORTA
#define KEY_DDR                  DDRA
#define KEY_PIN                  PINA
#include <avr/io.h>
#include <stdarg.h>
#include <stdio.h>
#include <util/delay.h>
#include "hd44780.h"
//#include <math.h>
#include <string.h>
#include <stdint.h>
#include <avr/interrupt.h>

#include "main.h"

uint16_t pulse = 0x0FFF;
uint16_t period = 0;

uint16_t adc = 0;

uint8_t alarmOn = 0;

volatile uint8_t keyPort = 0;
volatile uint16_t code = 0;

char buffer [10];

// Time variables
volatile uint8_t hour = 0;
volatile uint8_t minute = 0;
volatile uint8_t second = 0;


// TODO change secret on runtime
const uint16_t secret = 1234;

char secretS [4] = {'1','2','3','4'};


// notes
const uint16_t A4_FREQ = 440;
const uint16_t C5_FREQ = 523;
const uint16_t D6_FREQ = 1175;

// OCR1A values
#define A4 (F_CPU / (A4_FREQ * TIMER1_PRESCALER * 2) - 1)
#define C5 (F_CPU / (C5_FREQ * TIMER1_PRESCALER * 2) - 1)
#define D6 (F_CPU / (D6_FREQ * TIMER1_PRESCALER * 2) - 1)
```

```
typedef enum {
           RED = 0b01000010,
           GREEN = 0b100,
           OFF = 0
} LED_t;



// void adc_init(void)
// {
//         ADMUX = (1<<REFS0);// | (1<<MUX3) | (1<<MUX1);     //select AVCC
as reference
//         ADCSRA = (1<<ADEN) | (1<<ADSC) | (1<<ADIE) | 7;  //
enable and pre-scaler = 128 (16MHz/128 = 125kHz)
//         DIDR0 = (1<<ADC0D);
// }

void pwm_init()
{

                        TCCR3A |= (COM3B1);
                        TCCR3B |= (1<<WGM32) | (1<<CS31);

           // Output compare 1hz
           // Interrupt every second, increment time

           //OCR1A = 15625;
           OCR1A = 7812;
           //         OCR1B = 31250;
           //         OCR1C = 46875;

           //TODO second timer for keypad timeout
           //OCR2A = 15625;

}

// void timer2_start(timer_t state) {
//
//         if (state == on);
// }

void set_pulse(uint16_t p){
           OCR3A = p;
}

void set_period(uint16_t p){
           ICR3 = p;
}

int readAdc(char chan)
{
           //ADMUX = (1 << REFS0) | (chan & 0x0f);
           ADCSRA |= (1 << ADSC);
           while (!(ADCSRA & (1 << ADIF)));
           /*return (ADCL | (ADCH << 8));*/
           return ADCW;
}

void uart_init()
{
```

```
            UCSR0B |= (1<<TXEN0);
            UCSR0C |= (1<<UCSZ00)|(1<<UCSZ01);
            UBRR0L = 1;
}

void uartt(char rec[])
{

            for (int i = 0; i < strlen(rec); i++){
                        while (!(UCSR0A & (1<<UDRE0)));
                        UDR0 = rec[i];
            }
            while (!(UCSR0A & (1<<UDRE0)));
            UDR0 = '\n';
}

void keyInt(timer_t state)
{
            if (state == on){
                        PCIFR |= (1<<PCIF2); // interrupt r PCMSK2 enabled
                        PCMSK2 |= (1<<PCINT16) | (1<<PCINT0); // interrupt 16
enabled, TODO: keypad digital pins
            }
            else {
                        PCIFR = 0;
                        PCMSK2 = 0;
            }
}

typedef enum { false, true } bool;


// void playNote(float duration, int frequency) {
//          long int i,cycles;
//          int half_period;
//          int wavelength;
//
//
//          wavelength=(int)(1/frequency)*1000;
//
//          cycles = (int)duration/wavelength;
//
//          half_period = (int)wavelength/2;
//
//          SPEAKER_DDR |= (1 << SPEAKER_PIN);
//
//
//          for (i=0;i<cycles;i++) {
//
//                      for (int foo=0; foo<half_period;foo++) {
//                                  _delay_ms(1);
//                      }
//
//                      SPEAKER_PORT |= (1 << SPEAKER_PIN);
//
//                      for (int foo=0; foo<half_period;foo++){
//                                  _delay_ms(1);
//                      }
//                      SPEAKER_PORT &= ~(1 << SPEAKER_PIN);
//          }
//
```

```
//          return;
//
// }

int main(void)
{


            //output
            DDRD = 0xff;
            DDRB = 0b01000010;
            //DDRA = 0b01000110;
            PORTB = 1;

            // Set PORT C bit 0 to an input
            DDRC = 0x00;

            // Initialize ISR and Timers
            //isr_init();
            sei();
            pwm_init();

            // Initialize LCD
            lcd_init();
            lcd_goto(0x00);

            //adc_init();
            //keyInt(on);

            // Enter correct time with keypad
            enterTime();

            // Start counting seconds
            timerStart(on);

            // Enable sensor interrupt
            pcintOn(on);

            //set_pulse(0x00FF);
            //set_period(0x000A);

    while (1)
    {
                        if (alarmOn) alarm();
                        showTime(hour, minute, second);
                        _delay_ms(65);
                        lcd_clrscr();
    }
}

void showTime(uint16_t hour, uint16_t minute, uint16_t second)
{
                                    if (hour < 10) {
                                              lcd_putc('0');
                                    }
                                    lcd_puts(itoa(hour, buffer, 10));
                                    lcd_putc(':');
                                    if (minute < 10) {
                                              lcd_putc('0');
                                    }
                                    lcd_puts(itoa(minute,buffer, 10));
```

```
                                lcd_putc(':');
                                if (second < 10) {
                                        lcd_putc('0');
                                }
                                lcd_puts(itoa(second,buffer, 10));
}

void enterTime() {

        char hourArray [3] = {'0','0','\n'};
        char minuteArray [3] = {'0','0', '\n'};

        lcd_clrscr();
        lcd_goto(0x00);
        lcd_puts("Enter time:");
        lcd_goto(0x40);

        //while(1){
        for (int i = 0; i<2; i++) {
                hourArray[i] = keyfind();
                lcd_putc(hourArray[i]);

        }


        for (int i = 0; i<2; i++) {
                minuteArray[i] = keyfind();
                lcd_putc(minuteArray[i]);

        }
                                //break;
        //}
        hour = atoi(hourArray);
        minute = atoi(minuteArray);
        lcd_clrscr();

}


// set LED
void setLED(LED_t state) {
        PORTB = state;
}

void isr_init()
{
//      PCIFR |= (1<<PCIF2) | (1<<PCIF1); // interrupt r PCMSK2 enabled
//      PCICR |= (1<<PCIE1);
//      PCMSK2 |= (1<<PCINT16) | (1<<PCINT0); // interrupt 16 enabled,
TODO: keypad digital pins
//      PCMSK1 |= (1<<PCINT8);
        sei();
}

void pcintOn(timer_t state)
{
        if (state == on) {

        PCIFR |= (1<<PCIF1); // interrupt r PCMSK2 enabled
        PCICR |= (1<<PCIE1);
        PCMSK1 |= (1<<PCINT8);
```

```
            PORTB |= 1;
            }
            else {
                        PCIFR = 0;
                        PCICR = 0;
                        PCMSK1 = 0;
            }
}

void alarm()
{
            uint8_t tries = 3;

            uint8_t hourc = hour;
            uint8_t minutec = minute;
            uint8_t secondc = second;

            PORTB = 0b11000010;
            //PORTB |= 64;

            setLED(RED);

            char code [4] = {'0','0','0','0'};


            uint16_t intCode = 0;

            while (alarmOn && tries)
            {
                                    lcd_clrscr();
                                    lcd_goto(0x00);
                                    lcd_puts("Alarm Trigd!");
                                    lcd_goto(0x40);
                                    showTime(hourc, minutec, secondc);




                                    for (int i =0; i<4; i++) {
                                                code[i] = keyfind();
                                                if (!i) {
                                                            lcd_goto(0x40);
                                                            lcd_puts("
");
                                                            lcd_goto(0x40);
                                                            }
                                                lcd_putc('*');

                                    }

                                    intCode = atoi(code);


                                    if (intCode == secret)
                                    {
                                                alarmOn = 0;
```

```
                                        lcd_clrscr();
                                        lcd_goto(0x00);
                                        lcd_puts("Alarm disarmed!");
                                        PORTB &= 31;
                                        setLED(GREEN);
                                        _delay_ms(2000);
                                        setLED(OFF);
                                        break;
                                }
                                tries--;

            }

            while (!tries) {
                        lcd_clrscr();
                        lcd_puts("COPS CALLED >:(");
                        _delay_ms(100);
                        //playNote(400, 400);
            }
            pcintOn(on);

}



ISR(PCINT1_vect)
{

            alarmOn = 1;

            pcintOn(off);

}
// ISR(PCINT2_vect)
// {
//  lcd_putc('&');
//
//   alarmOn = true;
// }
// ISR(PCINT3_vect)
// {
//  lcd_putc('&');
//
//   alarmOn = true;
// }

void timerStart(timer_t state)
{
            if (state == on)
            {
                        TCCR1B |= (1<<CS10) | (1<<CS12);
                        TIMSK1 |= (1<<OCIE1A);
            }

            else
            {
                        TCCR1B = 0;
                        TCNT1 = 0;
            }
```

```
}

// TODO timer 2
void timer2start(timer_t state)
{
            if (state == on)
            {
                        TCCR2B |= (1<<CS10) | (1<<CS12);
                        TIMSK2 |= (1<<OCIE2A);
            }
            else
            {
                        TCCR2B = 0;
                        TCNT2 = 0;
            }
}

ISR(TIMER1_COMPA_vect)
{
            TCNT1 = TCNT1 - OCR1A;

            second++;
            if (second > 59) {
                        second = 0;
                        minute++;
            }
            if (minute > 59) {
                        minute = 0;
                        hour++;
            }
            if (hour > 23){
                        hour = 0;
            }
}

// TODO Timer 2
ISR(TIMER2_COMPA_vect)
{

            TCNT2 = 0;
}
```