LUNDS
UNIVERSITET

Larm

Kurs: EITA15 Grupp 20

Handledare: Bertil Lindvall & Lars-Göran Larsson.

David Baker
Marcus Nyberg Carlsson
Pontus Lindh

Abstract

The final part of the course EITA15 at LTH involves creating a working prototype of a product utilizing an Atmega1284 microcontroller and programming in C. Our group considered several different products and designs before settling on a motion activated alarm system. Our design incorporates a PIR motion detector, an LCD display, a 4x4 keypad and a buzzer all connected to the Atmega1284 microcontroller via a breadboard. If the system is armed and motion is detected, the alarm will sound. To turn off the alarm it is necessary to input a 4 digit code on the keypad to disarm the system. The motion sensor is attached through an extension cable for easier placement. Throughout the course of the project our group members deepened their understanding of how electrical components interact with each other, as well as how the C programming language interfaces with hardware in the real world.

Innehållsförteckning

Inledning

Detta projekt är det sista momentet av kursen Digitala System, som sammanfattar kursens teoretiska och praktiska moment. Kursen har lärt ut grunderna i ämnena digitala kretsar och datorteknik, där studenter har fått använda kunskapen praktiskt i olika labbar där studenter har använt processorn Atmega1284 och programmeringsspråket C. Handledare för projektet är Bertil Lindvall och Lars-Göran Larsson.

Syfte

Förberedelse av praktiskt arbete som ingenjör inom en projektgrupp vars mål är att bygga en fungerande hårdvara och sedan programmera mjukvaran i den samt ge en inblick i hur rapporter skrivs.

Målformulering

Projektet har som mål att konstruera en prototyp av ett funktionellt larm. Prototypen ska startas med att fråga efter ett 4-nummerat lösenord som användaren skriver in. Larmet bör göra ett högt ljud ifall fel lösenord skrivs in eller ifall den infraröda-sensorn skickar signal om aktivitet. Användaren stänger av ljudet genom att skriva in det korrekta lösenordet vilket sätter larmet i dvala, för att återstarta larmet måste ett nytt lösenord skrivas in.

Problemformulering

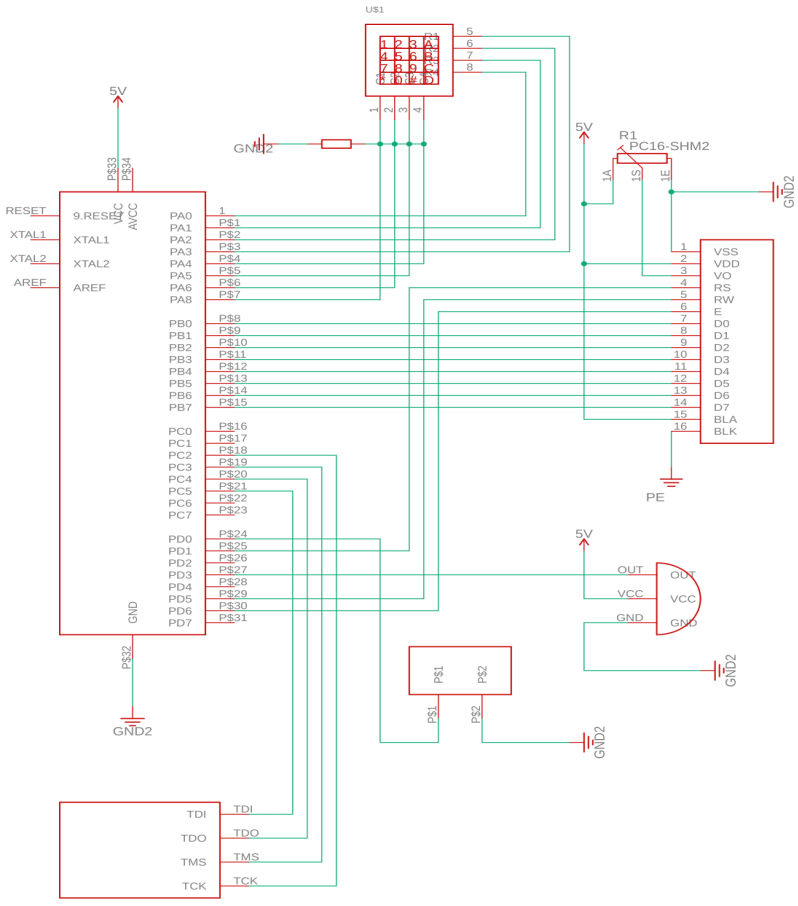
- Skriva en rapport som ger insyn i projektet.
- Ska vara användarvänligt.
- Sensorn ska känna av rörelse och skicka signal till processorn.
- Keypad ska skicka signal till processorn om vilken tangent som blir tryck.
- LCD-skärm ska ge relevant information till användaren som skickas från processorn.
- Programmering av processor med lågnivå-språket C.
- Ett kretsschema som visar hur prototypen är kopplad.

Motivering

- Detta projekt är gruppens första försök till att designa en sådan prototyp, således valdes ett relativt enkelt koncept att realisera. På så sätt får våra gruppmedlemmar möjlighet att utveckla vår kunskap om grunderna för hårdvaran och mjukvaran som tillsammans utgör så mycket av vår moderna värld.
- Under projektets gång har gruppen utvecklat en stor variation bland olika specifika kunskaper, så som hur komponenter initialiseras i C programmering på AVR mikrokontroller och hur stora knappsatser och tangentbord hanteras samt läses av på ett begränsat antal pins på keypaden.
- Larm är en produkt som är aktuell och användbar för många hushåll alternativt företag.

Teknisk Bakgrund

I denna del av rapporten kommer gruppen att fördjupa sig i de olika komponenter som använts för att bygga ihop projektet. De delar som ingår i projektet har tillhandahållits av Lunds Tekniska Högskola i Helsingborg.



2023-05-19 13:49 /Users/davidbaker/Documents/EAGLE/projects/Larm/Larm.sch (Sheet: 1/1)

Figur 1 Kretsschema som visar hur prototypens komponenter är sammankopplade.

Hårdvara

Mikroprocessor: ATMEGA 1284

Atmega1284 är en mikrokontroller som bland annat kan användas för att styra elektriska enheter. Den har 128 kB flashminne samt 16 kB sRam. Den har 40 olika pins. Mjukvaran kommer att programmeras i Atmel Studio

med programmeringsspråket C som kommer laddas över till processorn i JTAG.

Rörelsedetektor Pir-Sensor

Sensorn använder sig av infrarött ljus för att kunna detektera rörelser i sin omgivning. PIR står för Passive InfraRed, sensorn är alltså helt passiv. Den upptäcker rörelse genom att passivt mäta temperaturförändringar i det infraröda spektrumet; eftersom alla objekt och varelser utstrålar och reflekterar värme i form av elektromagnetisk strålning. Den här typen av strålning är osynlig för mänskliga ögat, eftersom den befinner sig utanför det synbara spektrumet, men med elektronisk utrustning är det möjligt att mäta. En PIR-sensor skickar alltså en elektrisk signal när den upptäcker en temperaturförändring, då det indikerar rörelse. Mikrokontrollern noterar signalen och sätter igång buzzern.

Skärm 16x2 GDM1602k

16x2 LCD-display är en mindre skärm som består av två rader som har plats för 16 tecken vardera. Det är en alfanumerisk lcd-skärm som kan använda sig av siffrorna mellan 0-9 samt bokstäverna A-Z. Den har även bakgrundsljus.

Summer (Buzzer)

Summern är en liten komponent som genererar ett skarpt ljud när spänning förs igenom den.

Potentiometer

Potentiometern är kopplad till lcd skärmen och fungerar som en resistor. Genom att man skruvar på denna justeras kontrasten på skärmen.

Knappsats 83BB1-001

Knappsatsen som valdes är en programmeringsbar 4x4 med 8 pins som kopplas till kretsen. När knapptryckning sker så bryts utsignalerna som skickas från processorn, denna data sparas som ett bitmönster och står för den tryckta knappens kolumn. När strömmen som flödar till en kolumn avbryts så skickas en signal till processorn, detta möjliggöra att både rader och kolumner kan kodas.

Utvecklingsmiljö Atmel Studio 7

Larmets processor programmerades i Atmel Studio med programmeringsspråket C.

Metod

Efter att gruppen bestämde sig för att göra ett larm så plockades komponenter i skolans labbsal som ansågs passa till gruppens projekt. Så fort lämpliga komponenter valts ritades ett kopplingsschema som visades till labbhandledare för att godkännas. Efter att kopplingsschemat blivit godkänt så kvitterades alla komponenter ut. Nästa steg var att koppla ihop alla delar utifrån det designade kopplingsschemat. Efter en och annan felkoppling så lärde gruppen sig tricket att kontrollera de olika komponenterna efterhand som de kopplades in i kretsen. En del hjälp från handledare togs också emot under de utsatta projekttiderna.

När hårdvaran kopplats korrekt så var nästa steg mjukvaran, själva programmeringen. Processorn programmerades i Atmel Studio med programmeringsspråket C. Programmeringen strukturerades genom att skapa olika sektioner i koden där varje sektion tillhörde en komponent. Inledningsvis programmerades LCD skärmen sedan summern och slutligen keypaden och PIR sensorn. Slutligen användes alla sektionerna i main metoden.

Källkritik

Källorna i projektet har varit datablad utdelat av labbhandledare. De var trovärdiga, p.g.a. LTH har hög trovärdighet samt har databladen fungerat praktiskt när gruppen har använt dem.

Analys

För att kunna programmera olika önskvärda instruktioner som enheten ska utföra behövdes en processor och gruppen valde en Atmega1284, gruppmedlemmarna har arbetat med denna tidigare. Gruppen valde också en infraröd sensor (Pir sensor) som mäter infraröd strålning, denna var liten till storleken och hade önskvärda egenskaper för projektet. Det fanns ett antal olika Lcd-skärmar att välja mellan men gruppen valde en av de större som hade bakgrundsbelysning, denna funktion ansågs vara viktig för att larmet skulle vara användbart vid mörkare ljusförhållanden. Tangentbordet valdes därför att storleken passade och det var enkelt att fästa på enheten, det fanns även möjlighet att lyfta på knapparna och lägga dit lappar med siffror. Gruppen valde även att löda fast så mycket som möjligt för att göra larmet så tåligt som möjligt.

Resultat

Kopplingen av prototypen fungerar enligt projektets mål som skrivs i Målformulering i kapitel 1. Atmega 1284 är central i prototypen då den var kopplad till LCD-skärmen, keypaden, summern och PIR-sensorn. Resultatet blev som önskat i målformulering då varje komponent skickade korrekta signaler.

Koden till prototypen som den är skriven i Appendix under Källkod fungerade enligt Målformulering i kapitel 1. LCD-skärmen var programmerad att ge information till användaren beroende på vad som efterfrågas. Processorn var kodad att läsa av rätt tangent från prototypens keypad och PIR-sensorn skickade en signal till processorn när rörelse upptäcktes. Processorn var programmerad för att hantera lösenord och veta när larmet skulle göra ljud.

Slutsats

Gruppen anser att larmet uppfyller de krav och förväntningar som ställdes när projektet påbörjades. Enheten reagerar ifall sensorn upptäcker rörelse i sin omgivning och meddelar detta till processorn som sätter igång buzzern. För att stänga av buzzern så måste rätt kod slås in via knappsatsen annars fortsätter den att tjuta. Lcd-skärmen har bidragit till att användare enkelt kan använda larmet. Resultatet visade att studenterna fullföljde de krav som krävdes för att kunna konstruera en prototyp av ett larm.

Framtida utvecklingsmöjligheter

Designen i dagsläget är inte optimal och hade kunnat förbättras avsevärt med tex en 3d-printare. Istället för att använda en kod för att stänga av larmet hade man kunnat använda en NFC-tag tex för att göra den ännu mer användarvänlig. Om man hade utvecklat denna prototypen till en färdig produkt så hade man fäst alla komponenter på ett säkrare sätt, tex använt sig av lödning på alla kopplingar.

Källor

ATMEGA1284 Atmel Corporation. San Jose, USA, 2016. Tillgänglig: https://www.eit.lth.se/fileadmin/eit/courses/datablad/Processors/ATmega1284.pdf (Hämtad 2023-05-19).

Pir Sensor (#555-28027) Parallax Inc. Rocklin, Kalifornien, USA, 2014. Tillgänglig: https://www.eit.lth.se/fileadmin/eit/courses/datablad/Lawicell/555-28027-PIR-Sensor-Prodcut -Doc-v2.2.pdf (Hämtad 2023-05-19).]

Källkod

```
/*
 * Created: 2023-05-17 15:20:05
 */

#include <avr/io.h>
#define F_CPU 1000000UL                                 // Frequency of clock
#include <avr/delay.h>                  // used for DELAYS
#include <string.h>                                     // string?
#include <stdio.h>                                      // ???
#include <signal.h>                                     // used for INTERRUPTS


/* Global variables*/
short col = 3;                                          // column of keypad
short row = 3;                                                  // row for keypad
short btn = 0;                                          // determines if a key on the keypad is
pressed or not
short nr = 0;                                           // goes from 0 to 3
short alarm = 0;                                        // if LOW buzzer shouldn't make
sound, if HIGH buzzer will make sound

char input[4] ;                                                 // four key inputs on
keypad

/* 4x4 keyboard */
char keys[4][4] =
{
                { 'A','0','A','A'},
                { '7','8','9','A'},
                { '4','5','6','A'},
                { '1','2','3','A'}
};

/*************************************************************************/
/* FUNCTIONS - BUZZER                                                    */
/*************************************************************************/
/* Manual function to turn on buzzer */
void tjut_on(){
                alarm = 1;
                _delay_ms(1);
                PORTD = 0x01;
}
/* Manual function to turn off buzzer*/
void tjut_off(){
                alarm = 0;
                _delay_ms(1);
                PORTD = 0x00;
}
/* Function that temporarily turns on buzzer*/
void tjut()
{
                _delay_ms(1);
                PORTD = 0x01;
```

```
                _delay_ms(2000);
                PORTD = 0x00;
}
/*************************************************************************/
/* FUNCTIONS - KEYPAD                                                    */
/*************************************************************************/
/* Function for initializing relevant registers and having correct settings on LCD-screen */
void lcd_init(){
                DDRB = 0xFF;
                DDRD = 0b01100011;

                _delay_ms(1);
                PORTB = 0x30;                                   // function set: 8 bit bus
mode with MPU
                d_enter();

                _delay_ms(1);
                PORTB = 0b00001111;             // initializes cursor
                d_enter();

                lcd_clear();                                    // clears screen in case anything
earlier was written on it
}
/* Function used to write a single character on LCD-screen */
void d_write(unsigned char c){
                if(alarm > 0){                  // alarm is ON
                                PORTD = 0b00000011;
                                PORTB = c;
                                PORTD = 0b01000011;
                                _delay_ms(100);
                                PORTD = 0b00000011;
                }
                else{                                           // alarm is OFF

                                PORTD = 0b00000010;             // write high
                                PORTB = c;                                      // portb =
char
                                PORTD = 0b01000010;             // write high and enable
high
                                _delay_ms(100);
                                PORTD = 0b00000010;             // write high and enable
low

                }
}

/* Function for lcd to commit to data-based changes */
void d_enter(){
                _delay_ms(1);
                if(alarm > 0){                                                  // is an
alarm going off
                                PORTD = 0b01000001;
                                _delay_ms(1);
                                PORTD = 0b00000001;
```

```
		}
		else{
			PORTD = 0b01000000;
			_delay_ms(1);
			PORTD = 0b00000000;
		}
}

/* Function for printing a text onto lcd-screen */
void lcd_msg(unsigned char text[]){
		for(short i = 0; i < strlen(text); i++)
			d_write(text[i]);
}
/* Function for clearing lcd-screen from any character */
void lcd_clear(){
		_delay_ms(1);
		PORTB = 0b00000001;
		d_enter();
}
/*************************************************************************/
/* FUNCTIONS - KEYPAD                                                    */
/*************************************************************************/
/* Function for initializing relevant registers*/
void keypad_init(){
		DDRA = 0b00111100;
		PORTA= 0b11000011;
}
/* Function for determine which key is pressed */
void get_key(){
		// local variables
		int arr[4] = { 0xCB, 0xC7, 0xE3, 0xD3 };
		int slask;

		for(int i = 0; i < 4; i++){
				PORTA = arr[i];

				slask = PINA & 0b11000011;

				if(slask == 0x0002){
					_delay_ms(1000);
					col = 0;
					row = i;
					break;
				}
				if(slask == 0x0001){
					_delay_ms(1000);
					col = 1;
					row = i;
					break;
				}
				if(slask == 0x0040){
					_delay_ms(1000);
					col = 2;
					row = i;
```

```
                                        break;
                                }
                                if(slask == 0x0080){
                                        _delay_ms(1000);
                                        col = 3;
                                        row = i;
                                        break;
                                }

                        }
}
/* Function for changing global variable input */
void keypad_input(){

        if(keys[row][col] != 'A' ){
                if(btn == 0){

                        d_write(keys[row][col]);

                        input[nr] = keys[row][col];
                        nr++;

                        _delay_ms(10000);
                        btn = 1;
                }

        } else btn = 0;
}
/*************************************************************************/
/* FUNCTION - MAIN                                                       */
/*************************************************************************/
int main(void)
{
        lcd_init();

        keypad_init();
        char password[4] = { 'x','x','x','x' };
        lcd_msg("Register a Password:");

        int e;

   while (1)
   {
                row = 3, col = 3;

                get_key();

                keypad_input();

                // If correct key is pressed, save the input and print onto screen
                if(keys[row][col] != 'A'){
                        if(btn == 0){

                                d_write(keys[row][col]);
```

```
                                        input[nr] = keys[row][col];
                                        nr++;

                                        _delay_ms(10000);
                                        btn = 1;
                                }

                        } else btn = 0;

                        // When 4 numbers have been input, match corresponding values
depending on state.
                        if(nr == 4){
                                if(password[0] == 'x' && password[1] == 'x' &&
                                                        password[2] == 'x' &&
password[3] == 'x' ){
                                        password[0] = input[0];
                                        password[1] = input[1];
                                        password[2] = input[2];
                                        password[3] = input[3];
                                        lcd_clear();
                                        lcd_msg("Password registered.");
                                        _delay_ms(5000);
                                        lcd_clear();
                                        lcd_msg("Alarm turned on.");
                                        _delay_ms(5000);
                                        lcd_clear();
                                        lcd_msg("Password:");
                                }
                                else if(input[0] == password[0] && input[1] ==
password[1] &&
                                                                        input[2] ==
password[2] && input[3] == password[3] ){
                                        lcd_clear();
                                        password[0] = 'x';
                                        password[1] = 'x';
                                        password[2] = 'x';
                                        password[3] = 'x';

                                        alarm = 0;

                                        lcd_clear();
                                        lcd_msg("Password was correct.");
                                        _delay_ms(5000);

                                        lcd_clear();
                                        lcd_msg("Alarm turned off.");
                                        _delay_ms(5000);

                                        lcd_clear();
                                        lcd_msg("Register a Password:");
                                }
                                else{
                                        lcd_clear();
```

```
                                        lcd_msg("Criminal scum!");
                                        _delay_ms(10000);
                                        alarm = 1;

                                        _delay_ms(2000);
                                        lcd_clear();
                                        lcd_msg("Password:");
                                }

                                nr = 0;
                        }

                        // If password has been set
                        if(password[3] != 'x')
                        {
                                e = PIND;


                                if(e != 0x002){
                                        alarm = 1;
                                }

                        }

                }

}
```