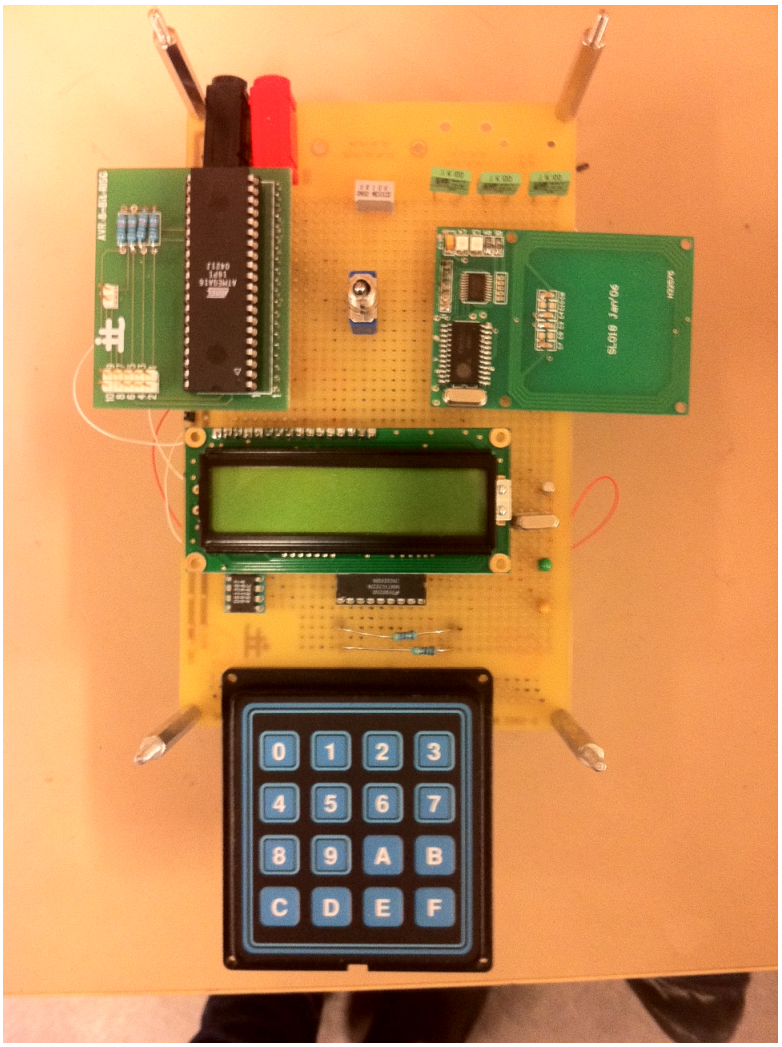
EITF40 - RFID-terminal

Joakim Marculescu (dt08jm6) , Eric Johansson (dt08ej6)

2012-02-28



Contents

1 Introduktion

Syftet med projektet i kursen var att utifrån en egen idé konstruera en hårdvaruprototyp samt implementera mjukvara så att dessa interagerar på ett korrekt sätt, vilket skulle leda till en fungerande slutprodukt. Syftet var att få en ökad förståelse för kretskonstruktion samt programmering i programspråket C.

Projektet som utfördes gällde en RFID-terminal vilket påminner om de kortläsare vilka finns tillgängliga i E-huset på Lunds Tekniska Högskola. RFID-läsaren känner av en RFID-tag som är kopplad till ett specifikt kort vilken har tillgivits en särskild pinkod. Detta simulerar en kortläsare som är kopplad till ett dörrlås där rätt tag och pinkod måste skrivas in för att dörrlåset skall öppnas.

2 Kravspecifikation

Det ställdes krav på prototypen och dess funktioner.

- Det skall finnas två olika lägen för systemet, en för registrering av ny användare och en för igenkänning av registrerad användare.
- Korrekt visuell feedback från AVR till display.
- Korrekt avläsning av ID-tag på kort av RFID-läsaren.
- Korrekt avläsning av knappsatsen.
- Ett fungerande mjuk- och hårdvarusystem som interagerar med varandra på ett korrekt sätt för att simulera ett verklighetstroget kortavläsarsystem.
- Optimerad implementation av kod sådan att knapptryckningar med kort tidsintervall inte orsakar problem.
- Vid läsning av icke-registrerad användare skall systemet neka till att knappa in personlig kod.
- Vid läsning av registrerad användare skall användaren få möjlighet att knappa in personlig kod.
- Användare skall inte kunna knappa in personlig kod innan läsning av ID-tag skett.
- Maximal tid för intryckning av personlig kod skall vara 10 sek.
- Den personliga koden får endast bestå av 4 siffror.

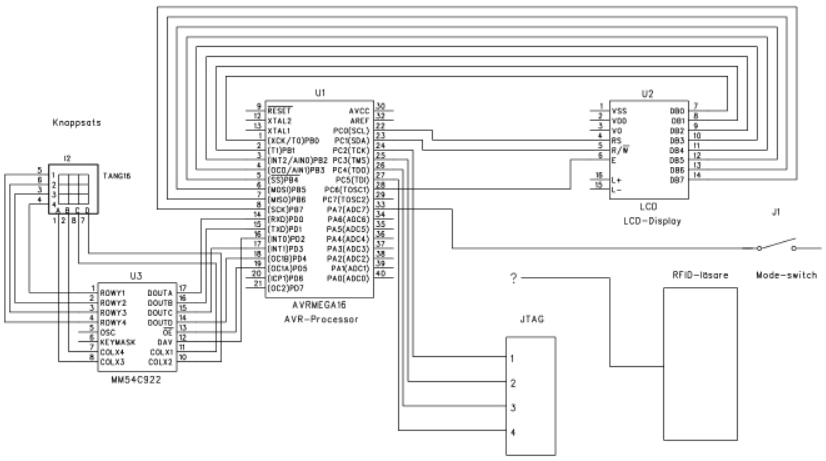


Figure 1: Blockschema för originalprototypen

3 Prototypen

3.1 Hårdvara

RFID-läsare (SL018)

RFID-läsare med stöd för tagarna Mifare 1K, 4K och Ultralight. RFID-läsaren har 5 pinnar:

- TagSta
- SDA
- SCL
- VCC
- GND

TagSta (Tag Status) - Indikerar att ett kort är inom avläsningsavstånd genom att ge en nolla som output, annars en etta.

SDA (Serial Data Line) - Pinnen som används för att skriva och läsa data.

SCL (Serial Clock Line) - Används för att tala om när I2C moduler skall läsa och skriva på SDA.

VCC - Spänning.

GND - Jord.

SL018 kommunicerar med andra enheter genom kommunikationsprotokollet I2C och kan på så sätt utföra diverse read- och writeoperationer.

AVR ATMega16

AVR ATMega16 är en processor som består av 32 programmerbara I/O pinnar. Processorn tillhandahåller ett inbyggt flashminne på 16KB samt ett 1KB internt minne. För att kommunicera med processorn användes en JTAG-enhet och programmet AVR Studio 4.

LCD-display

För RFID-terminalen användes en LCD-display med möjlighet att skriva ut 16x2 tecken. Bakgrundsbelysning fanns att använda men ansågs inte nödvändig för detta projekt. För inmatning av tecken används ASCII standarden.

Knappsats (Grayhill 88BB2-072)

Knappsatsen vilken användes var en Grayhill med siffrorna 0-9 och bokstäverna A-F. I detta projektet hanteras endast de numeriska knapparna. Informationen mellan knappsatsen och processorn skickades med hjälp av en avkodare.

Avkodare

Användes för att summera knappsatsens 8 pinnar in, till 4 pinnar ut från avkodaren och in till processorn.

8kHz kristall

För att generera en tidskontinuerlig klocka till processorn användes en 8kHz kristall. Detta användes för att kunna generera en jämn signal på I2C-bussen.

Övriga komponenter

Det användes även 3 kondensatorer till processorn, displayen och RFID-läsaren. Det användes även 2 stycken 4,7kOhm pull-up resistorer till I2C-bussen.

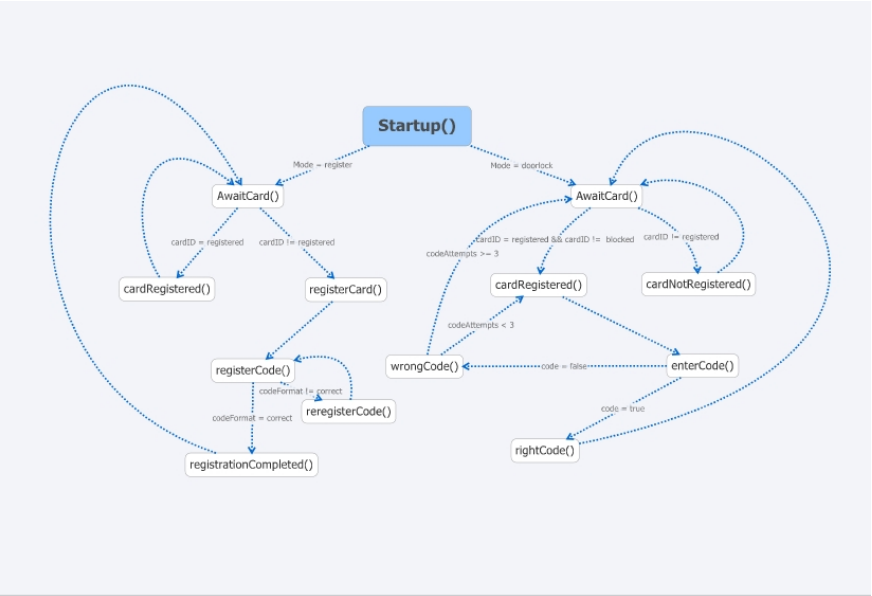


Figure 2: Flödesschema över programmet

3.2 Mjukvara

Implementeringen av koden gjordes i programspråket C och skrevs i AVR Studio 4. Det fanns två olika lägen till prototypen, ett läge för registrering av ny användare med särskild tag och unik pinkod och ett läge för att kontrollera en redan registrerad användare och hantera dennes access till applikationen vilken RFID-läsaren var kopplad till.

Då läget är inställt på registrering kommer processen till en while-loop (AwaitCard()), där den väntar på att ett kort skall hållas vid RFID-läsaren. Efter att kortets ID har lästs in jämför programmet det med de redan registrerade kort som finns lagrade. Om kortet redan är registrerat går programmet till AwaitCard(), annars vidare till en process där användaren skall skriva in sin personliga kod vilken skall vara sammanknuten med kortet. Då användaren har skrivit in sin personliga kod på rätt format så sparas denna och är sammanknuten till RFID-kortets ID. Då lagringen är klar återgår programmet till AwaitCard().

Läget där programmet kollar efter redan registerade kort börjar på samma sätt som för registreringsläget, dvs den ligger och väntar på att kort skall hållas vid RFID-läsaren i en while-loop (AwaitCard()). Om kortet som avläses inte är registerat återgår programmet till AwaitCard(). Om det är registrerat så går den till ett läge (enterCode()), där den väntar på att användaren skall skriva in en pinkod. Om användaren skriver in en felaktig pinkod går den vidare till läget wrongCode() och sedan tillbaka till enterCode(). Efter tre felaktiga intryckningar av pinkoden nekas användaren access och programmet återgår till AwaitCard(). Om rätt pinkod trycks in visas "Welcome" på displayen och access ges till användaren och programmet återgår till AwaitCard().

4 Utförande

Projektet startade genom att komma fram till vad för prototyp som skulle skapas vilket blev en RFID-terminal. Detta valdes för att det ansågs vara intressant att lära sig hur sådana fungerar och det ansågs vara ett rimligt projekt att ta sig an. Efter detta skapades en kravspecifikation för prototypen med olika krav som kunde ställas och som implementeringen skulle utgå ifrån.

När kravspecifikationen ansågs vara klar och godkänd gjordes ett blockschema och flödesschema för prototypens konstruktion samt kodimplementering. Detta för att på ett lättare sätt kunna koppla ihop de olika enheterna på kretskortet och för att få en överblick över kommunikationen mellan dessa.

Därefter konstruerades prototypen och alla komponenter löddes på kretskortet och kablarna drogs mellan enheterna, till stor del utifrån det framställda blockschemat.

När kontruktionen var gjord testades de olika komponenterna var för sig. Testningen skedde genom att testkod skrevs för varje enhet och logikpenna användes för att verifiera värdena på diverse pinnar. Det första testet avsåg LCD-displayen, vilket valdes för att lättare kunna verifiera att andra enheter fungerade. När displayen var testad och fungerade som tänkt testades knappsatsen. Med hjälp av displayen kunde det enkelt verifieras att knappsatsen fungerade på det sätt som var tänkt och när detta var klart testades till slut RFID-läsaren.

Efter att testningen var klar skrevs resterande kod för att framställa ett program vilket skötte interaktionen mellan de olika enheterna.

5 Diskussion

Under projektets gång stötte vi på många problem. Det första riktiga problemet var att få LCD-displayen att fungera. Vi lärde oss efter en stund att det var bättre att sköta diverse portar och pinnar manuellt. På så sätt fick vi en bättre förståelse för hur den skulle programmeras för att fungera som vi ville. Då vi med denna kunskap fortfarande inte fick den att fungera ordentligt diskuterade vi med handledaren och kom fram till att det kunde vara ett hårdvaruproblem. Vi bytte LCD-display och fick då allting att fungera.

Nästa problem uppkom när vi skulle testa knappsatsen. Vid testningen såg vi att en knapptryckning genererade två interrupts när den skulle generera endast ett. Detta åtgärdades med hjälp utav två kondensatorer.

Det sista, stora problemet uppstod när vi skulle få igång kommunikationen mellan processorn och RFID-läsaren med hjälp utav I2C protokollet. Till en början var vi inte bekanta med I2C protokollet och hade därför väldigt svårt att få RFID-läsaren att kommunicera på rätt sätt. Efter att ha läst på ordentligt ansåg vi oss ha tillräcklig förståelse för hur protkollet fungerade och började implementera kod för att initiera en kommunikation och få en röd LED-lampa att börja lysa på RFIDn. Då detta inte fungerade diskuterades problemet med handledaren och slutsatsen blev att vi skulle titta på signalerna vilka genererades av processorn med hjälp av ett oscilloskop. Med oscilloskopet såg vi att signalerna vilka vi skickade till RFIDn låg tidsmässigt rätt. Slutligen analyserade vi statusregistret för RFID-kommunikationen och såg att RFID-läsaren skickade ett 'Not Acknowledge' på adressen för RFIDn. Här skulle egentligen

ett 'Acknowledge' ha skickats för att kommunikationen skulle fungera.

Än en gång diskuterade vi våran situation med handledaren och det resulterade i att vi kopplade in en annan hårdvaruenhet som använde sig av I2C protokollet. När detta gjordes fungerade kommunikationen med den enheten och slutsatsen kunde dras att det var ett hårdvarufel, RFID-läsaren fungerade inte korrekt.

Slutprototypen blev inte som vi hade tänkt oss från början då projektet gick ut på att skapa en RFID-terminal och när den viktiga komponenten, RFID-läsaren inte fungerade, kunde inte heller kravspecifikationen uppfyllas till fullo. Med mer tid hade vi kunnat byta RFID-läsaren och få projektet att bli som vi ville men detta kunde i vårat fall inte göras p.g.a att det inte fanns någon mer RFID-läsare att tillgå.

6 Resultat och lärdomar

Resultatet av vårat projekt blev en prototyp vilket mer liknade en vanlig portkodsterminal med visuell feedback. RFID-läsaren används för att skicka en signal till processorn att en pinkod skall skrivas in. Tanken var att varje kort skulle kunna tilldelas en egen unik pinkod men då kommunikationen inte fungerade var inte detta möjligt.

Det vi har lärt oss från detta projektet är framförallt hur I2C protokollet fungerar. En annan viktig lärdom vi fått ut av detta projektet är att felsöka då det har varit en väsentlig del under processens gång. Vi har lärt oss programmet AVR Studio samt hur man använder sig utav JTAGen. Det har varit nyttigt att skapa ett blockschema och sedan få koppla ihop alla enheter utifrån detta och sedan kunna verifiera att allting är rätt kopplat. Vi har tagit lärdom av att det är svårt att tidsestimera ett projekt då vi haft begränsad kunskap om de verktyg och komponenter som använts genom projektets gång.

Appendix

Main.c

```
#include <avr/io.h>
#include "lcd.h"
#include <avr/interrupt.h>
#define F_CPU 1000000UL  // 1 MHz
#include <util/delay.h>
#include <inttypes.h>

uint8_t temp;
int PINS, pw0, pw1, pw2, pw3,  enterPW, pwCounter,
entpw0,entpw1,entpw2,entpw3, enabled;
volatile int cardIN;

ISR(INT0_vect)
{
        if(enterPW == 1)
        {
                _delay_ms(50);
                temp = PIND;
                temp &= 0x33;
                switch(temp)
                {
                case 0x00 :
                        PINS = 0;
                        disp_writeCh('*');
                break;

                case 0x01 :
                        PINS = 1;
                        disp_writeCh('*');
                break;

                case 0x02 :
                        PINS = 2;
                        disp_writeCh('*');
                break;

                case 0x03 :
                        PINS = 3;
                        disp_writeCh('*');
                break;

                case 0x10 :
                        PINS = 4;
                        disp_writeCh('*');
                break;
```

```
case 0x11 :
        PINS = 5;
        disp_writeCh('*');
break;

case 0x12 :
        PINS = 6;
        disp_writeCh('*');
break;

case 0x13 :
        PINS = 7;
        disp_writeCh('*');
break;

case 0x20 :
        PINS = 8;
        disp_writeCh('*');
break;

case 0x21 :
        PINS = 9;
        disp_writeCh('*');
break;
}
switch(pwCounter)
{
case 0:
        entpw0 = PINS;
break;

case 1:
        entpw1 = PINS;
break;

case 2:
        entpw2 = PINS;
break;

case 3:
        entpw3 = PINS;
break;
}
pwCounter++;
if(pwCounter == 4)
{
        int result = pw_match(entpw0,entpw1,entpw2,entpw3);
        if(result == 1)
        {
```

```
                                disp_Welcome();
                                _delay_ms(50000);
                                _delay_ms(1000);
                                _delay_ms(1000);
                        }
                        else
                        {
                                disp_wrongPIN();
                                _delay_ms(50000);
                                _delay_ms(1000);
                                _delay_ms(1000);
                        }
                        cardIN = 0;
                        pwCounter = 0;
                        entPW = 0;
                }
        }
        _delay_ms(500);

}

int main()
{
        cli();                          //enable interrupt
        DDRC = 0xff;
        DDRB = 0xff;
        DDRA = 0xff;
        DDRD = 0x00;
        MCUCR = 0x03;
        GICR = 0x40;
        SREG = 0x80;
        temp=0x00;
        PINS = 0x00;
        cardIN = 0;
        enterPW = 0;
        pwCounter = 0;
        pw0 = 0;
        pw1 = 1;
        pw2 = 2;
        pw3 = 3;

        sei();                          //disable interrupt


        disp_init();
        disp_home();
        disp_hello();

        while(1)
        {
```

```
                disp_home();
                disp_hello();
                while(cardIN != 1)
                {
                        checkforCardIN();
                        _delay_ms(500);

                }
                disp_writePIN();
                disp_moveBotRow();
                enterPW = 1;
                while(cardIN != 0);
        }
}

void
checkforCardIN()
{
        temp = PIND;
        temp &= 0x80;
        if(temp == 0x00)
        {
                cardIN = 1;
        }
}

int
pw_match(int p0,int p1,int p2, int p3)
{
        if(p0 == pw0, p1==pw1, p2==pw2, p3==pw3)
        {
                return 1;
        }
        return 0;
}
```

lcd.c

```
#include <avr/io.h>
#include "lcd.h"
#include <avr/interrupt.h>
#include <util/delay.h>

void write_cmd(char val) {
        _delay_ms(5);
        PORTB=val;
        PORTA=0x00;
        PORTA=0x01;
        PORTA=0x00;
}

void disp_writeCh(char val) {
        _delay_ms(5);
        PORTB=val;
        PORTA=0x04;
        PORTA=0x05;
        PORTA=0x04;
}

void disp_clear() {
        _delay_ms(10);
        write_cmd(0x38); //function set
        _delay_ms(10);
        write_cmd(0x01); //clear display
        _delay_ms(10);
}

void disp_init() {

        disp_clear();
        _delay_ms(5);
        write_cmd(0x0f); //display on
        _delay_ms(5);
        write_cmd(0x06); //entry mode set
}


void disp_home(){
        write_cmd(0x03);
}

void disp_writePIN()
{

disp_clear();
```

```
_delay_ms(10);
disp_writeCh('E');
disp_writeCh('n');
disp_writeCh('t');
disp_writeCh('e');
disp_writeCh('r');
disp_writeCh(' ');
disp_writeCh('p');
disp_writeCh('i');
disp_writeCh('n');
disp_writeCh('c');
disp_writeCh('o');
disp_writeCh('d');
disp_writeCh('e');
disp_writeCh(':');
}

void disp_hello()
{
        disp_clear();
        _delay_ms(10);
        disp_writeCh('H');
        disp_writeCh('e');
        disp_writeCh('l');
        disp_writeCh('l');
        disp_writeCh('o');
        disp_writeCh('!');
        _delay_ms(10);
}

void disp_moveBotRow()
{
        disp_home();
        _delay_ms(10);
        for(int i = 0; i < 40; i++)
        {
        write_cmd(0x14);
        _delay_ms(10);
        }

}

void disp_Welcome()
{
        disp_clear();
        _delay_ms(10);
        disp_writeCh('W');
        disp_writeCh('e');
        disp_writeCh('l');
        disp_writeCh('c');
```

```
        disp_writeCh('o');
        disp_writeCh('m');
        disp_writeCh('e');
        disp_writeCh('!');
        _delay_ms(10);
}

void disp_wrongPIN()
{
        disp_clear();
        _delay_ms(10);
        disp_writeCh('W');
        disp_writeCh('r');
        disp_writeCh('o');
        disp_writeCh('n');
        disp_writeCh('g');
        disp_writeCh(' ');
        disp_writeCh('c');
        disp_writeCh('o');
        disp_writeCh('d');
        disp_writeCh('e');
        _delay_ms(10);
}
```