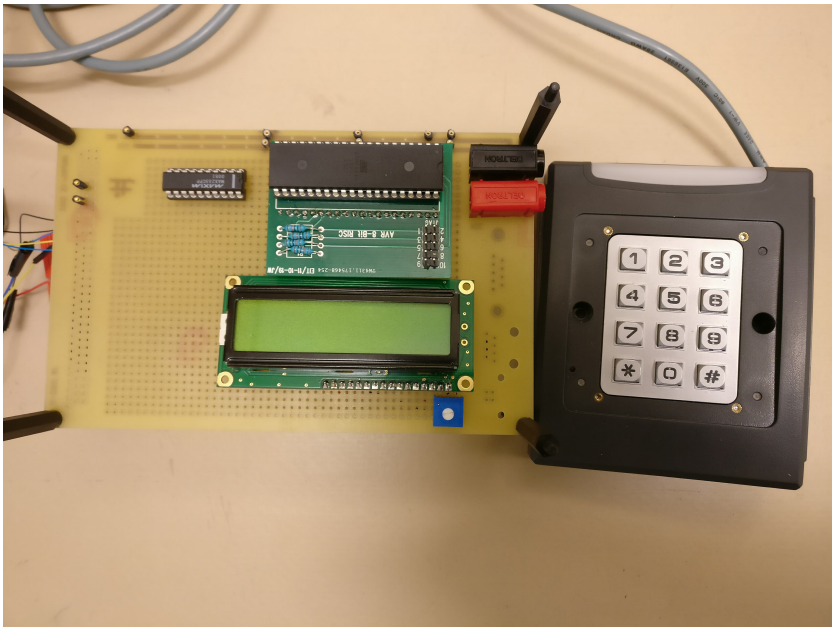
EITF11 - Digitala Projekt

Tvåfaktorslås

Lunds Universitet, Lunds Tekniska Högskola



Skriven av

Erik Mellberg, Hannes Olsson & Zack Jeppesen

22 maj 2018

Abstract

The purpose of this report is to describe and evaluate the process of designing and mounting a code lock with a card reader, in regards to both hardware and software. The process was part of the course EITF11 - Digitala Projekt at Lunds Tekniska Högskola. The project resulted in a working prototype, according to the requirement specifications.

Innehåll

1 Inledning

Tvåfaktorslåset utvecklades i samband med kursen EITF11 - Digitala projekt vid Lunds Tekniska Högskola. Vi låter institutionens formulering av kursens syfte från kurshemsidan tala för sig själv:
Syftet med kursen är att illustrera industriellt utvecklingsarbete. Målet med projektuppgiften är en prototyp för vidareutveckling med nödvändig dokumentation. Huvuddelen av kursen består i att konstruera, bygga och testa respektive konstruktion."

I kursen har vi fått testa att utifrån en idé skapa en ritning, för att sedan konstruera och programmera den slutgiltiga prototypen.

2 Produkt

2.a Produktbeskrivning

Ett kodlås kan användas för att säkerställa att något man vill hålla säkert ej kan tillträdas av obehöriga. För att maximera säkerheten har vi skapat ett lås som använder sig av tvåfaktorsautentisering i form av ett kort och en kod. I projektet har vi använt oss av våra LU-kort, men användningsområdet är inte begränsat till dessa. Prototypen är ej utrustad med någon form av mekanisk upplåsningsmekanism.

2.b Kravspecifikation

Den färdiga prototypen ska uppfylla följande krav:

- Prototypen skall läsa LU-kort och matcha kortsignaturen mot en databas av användare.
- Prototypen skall på en display tydliggöra om ett LU-kort inte är associerat med en tillåten användare
- Prototypen skall på en display tydliggöra om ett LU-kort är associerat med en tillåten användare, och begära en fyrsiffrig kod.
- Prototypen skall jämföra angiven fyrsiffrig kod mot den unika kod som är sparad för användaren vars kort är läst.

- Om rätt kod är angiven för en tillåten användare skall användaren välkomnas på en display.
- Om fel kod anges skall detta tydliggöras på skärmen, och autentiseringsförsöket avbrytas.
- Om ingen kod anges inom rimlig tid efter att ett kort har lästs skall autentiseringsförsöket avbrytas.

3 Hårdvara

3.a Komponenter

Processor
ATmega16 8-bit microcontroller med 40 pins används för att styra övriga komponenters beteende, samt för att lagra och söka databasen av tillåtna användare.

JTAG
JTAG ICE 3 används för att koppla processorn mot en dator, för programmering och debugging.

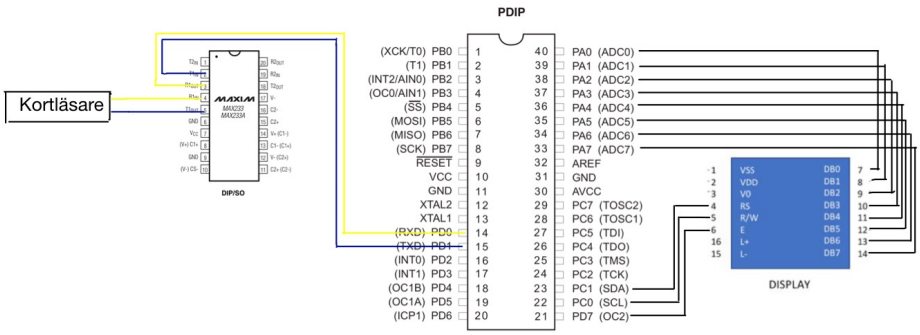
Display
SHARP Dot-Matrix GDM1602K, alfanumerisk display, vars syfte är kommunikation med användaren.

Kortläsare
DF 750k vars syfte är att ta emot inputs från användaren i form av kortläsning och knapptryck.

Variabel resistor
Kopplas mellan spänningskälla och display för att styra skärmens ljusstyrka.

RS232 driver
MAX 232 används för att överföra kortläsarens 12V signal till 5V vilket användning med ATmega16 kräver.

3.b Kopplingsschema



Figur 1: Kopplingsschema

4 Mjukvara

Mjukvaran är skriver i C med hjälp av programmet AtmelStudio 7. JTAG användes för programmering och debugging av processorn. Källkoden återfinns i Bilaga A.

Programmet drivs av en passiv, oändlig, while-loop fram tills en interrupt genereras av kortläsning eller knapptryck. När ett sådant avbrott sker undersöks den mottagna datans typ, samt programmets nuvarande status för att avgöra nästa handling enligt kravspecifikationen. Detta sker genom variabler som kontrollerar den mottagna datans längd, huruvida en tillåten användares kort har lästs och att ett autentiseringsförsök således är pågående, samt hur många knapptryck som skett under ett autentiseringsförsök.

5 Metod

5.a Planering

Det bestämdes under kursens första vecka att tillverkning av ett kodlås med display skulle ske. Eftersom att det vid kursens start återstod en hel läsperiod

innan konstruktionsarbete kunde påbörjas sköts formulering av tidsplanering och kravspecifikation upp till vårens andra läsperiod. Vid överläggningar med handledaren i samband med redovisning av idé och preliminärt kopplingsschema under andra läsperiodens första vecka beslutades det att en kortläsare skulle införas i designen. Det bestämdes strax efteråt att samtliga gruppmedlemmar skall vara delaktiga under alla utvecklingsfaser. Tidsplan och kravspecifikation togs fram.

5.b Montering

Utefter kravspecifikationen och antagna egenskaper hos den av handledaren tilldelade kortläsaren valdes övriga komponenter ut och kopplingsschemat togs fram. För att säkerställa att montering skett på korrekt vis testades varje komponents funktionalitet innan monteringsarbete fortsatte. Detta innebär att montering och programmering skedde sömlöst även under projektets tidiga faser.

Monteringens första steg var monteringen av mikrokontrollerns strömförsörjning. Spänningsmätare användes flitigt för att undvika förstörelse av hårdvaran, men på grund av mänskligt misstag förstördes en processor genom kortslutning. Nästa försök var mer framgångsrikt, och mikrokontrollern anslutning till en dator körande AtmelStudio 7 via JTAG lyckades.

Nästa steg var att montera skärmen. Detta skedde enligt kopplingsschemat och monteringen ansågs lyckad när "Hello World!"kunde visas på skärmen. Därefter skulle kortläsaren monteras. Efter att strömförsörjning av kortläsaren misslyckats upptäcktes det att kortläsaren ej, likt övrig utrustning, fungerade vid 5V, utan krävde en spänning mellan 7-24V. Överläggningar med handledare ledde till att kopplingsschemat uppdaterades med en sekundär spänningskälla, samt en komponent som konverterade kortläsarens signaler till 5V. Montering fortskred enligt det reviderade kopplingsschemat tills kommunikation mellan kortläsaren och mikrokontollern lyckades, och ansågs sedan komplett

5.c Programmering

Programmering påbörjades enligt ovan genom skapandet av simpla metoder för att skriva ut tecken på skärmen. Detta genomfördes med relativ enkelhet

med en main-metod innehållandes en oändlig while-loop. Nästa steg var kommunikation mellan mikrokontroller och kortläsare via UART. Efter åtskilliga timmar av felsökning, där varje åtgärdat misstag ledde till upptäckten av ett annat, uppnåddes kommunikation mellan komponenterna. Nästa steg var att förstå vad som skickades, vilket skulle uppnås genom att skriva ut den från kortläsaren överförda datan på displayen. På grund mycket förvirring gällande överföringsprotokollets ramförmat löstes detta genom att långsamt testa sig fram. Övergången från en konstant loop till ett interrupt-drivet program visade sig vara lösningen, och knapptryck på kortläsaren ledde sedan till att motsvarande tecken visades på skärmen.

Komponenternas grundläggande funktionalitet var nu färdig och skapandet av det faktiska programmet påbörjades. På grund av att mängden tillåtna användare är begränsad till den utvecklande gruppen skapades databasenäv tillåtna användare direkt i programmet. Funktionaliteten hos programmet testades och justerades tills funktionen var acceptabel, varefter prototypen ansågs färdig.

6 Diskussion

Utvecklingsprocessen genomsyrades av långa sessioner utan framgång, med enstaka genombrott då och då. Bristande kunskaper inom både elektronik och maskin-nära kodning ledde till att vi ofta famlade i mörker och sakta testade oss fram. På grund av detta var det väldigt användbart att vi satt tillsammans och experimenterade, då detta gjorde brainstormingprocessen mycket effektivare.

Den mest problematiska delen av utvecklingsarbetet är utan tvekan implementationen av UART, d.v.s. kommunikationen mellan kortläsare och mikrokontroller. Avsaknaden av datablad till kortläsaren tillsammans med, i programmeringssammanhang, mycket bristande tillgång på dokumentation och exempel online ledde till att denna del av projektet blev en stor utmaning för oss.

Vi är väldigt nöjda med vår insats i kursen. Prototypen som bildats genom våra framgångar är definitivt acceptabel, men framför allt är vi nöjda med den kunskap och erfarenhet vi bildat genom alla motgångar och misstag.

7 Bilaga A

```
#define F_CPU 4000000UL
#include <avr/delay.h>
#include <avr/io.h>
#include <stdlib.h>
#include <util/delay.h>
#include <avr/interrupt.h>


/*LCD function declarations */
void LCD_send_command(unsigned char cmnd);
void LCD_send_data(unsigned char data);
void LCD_init();
void LCD_goto(unsigned char y, unsigned char x);
void LCD_print(char *string);
void LCD_clear();

/*UART function declarations*/
unsigned char uart_recieve();
void uart_init();
void uart_transmit(char data);

void scanCard(char rb);

#define LCD_DATA_PORT PORTA
#define LCD_DATA_DDR DDRA
#define LCD_DATA_PIN PINA

#define LCD_RW PC0
#define LCD_RS PC1
#define LCD_E PD7

#define LCD_RW_PIN 0
#define LCD_RS_PIN 1
#define LCD_E_PIN 7

#define USART_BAUDRATE 9600
```

```
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

#define START_TIMER0  TCCR0 |= (1 << CS02)|(1 << CS00);
#define STOP_TIMER0   TCCR0 &= 0B11111000;
#define CLEAR_TIMER0  TCNT0 = 0;

#define DB_size 10 //Sets maximum number of users

/*Variables*/
int kcounter=0;
int ccounter=0;
int dataLength=0;
int currentCardID=0;
int currentPassCode=0;
int cardMatch=0;
int scannedCard=0;
int passCodes[DB_size];
int cardIDs[DB_size];

// global variable to count the number of overflows
volatile uint8_t tot_overflow;

int main(void)
{
uart_init();
LCD_init();
DB_init();
LCD_print("Scan card");
while(1){}
}

/* Interrupts */

// TIMER0 overflow interrupt service routine
// called whenever TCNT0 overflows
ISR(TIMER0_OVF_vect)
{
// keep a track of number of overflows
```

```
tot_overflow++;
if (tot_overflow>=250) {
LCD_clear();
LCD_print("Scan Card");
cardMatch=0;
STOP_TIMER0;
tot_overflow = 0;
}
}

/* Interrupt for USART Receive */
ISR(USART_RXC_vect)
{
char ReceivedByte = UDR; // Fetch the received byte value into the variable "ByteR
if (dataLength==0)
{
dataLength=(ReceivedByte);
}
else if(dataLength==3)
{
if(cardMatch == 1)
{
keyPress(ReceivedByte);
}
dataLength = 0;
}
else
{
scanCard(ReceivedByte);
}
}

void keyPress(char rb)
{
if (rb==0x00)
{
return;
}
```

```
if(cardMatch==1)
{
currentPassCode+=rb; //summerar koden, bör ersättas med en Long som jämför hexadec
kcounter++;
if(kcounter==4)
{
if (currentPassCode==passCodes[scannedCard])
{
LCD_clear();
CLEAR_TIMER0;
LCD_print("Welcome!");
}
else
{
LCD_clear();
CLEAR_TIMER0;
LCD_print("Wrong code");
}
kcounter=0;
cardMatch=0;
currentPassCode=0;
}
}
}

void scanCard(char rb)
{
currentCardID+=rb; //se keyPress...
ccounter++;
if (ccounter==dataLength)
{
for (int i = 0; i < DB_size; i++) //Går igenom listan med kort
{
if (currentCardID==cardIDs[i])
{
cardMatch=1;
scannedCard=i; //Sparar vilket kort som scannats
LCD_clear();
```

```
LCD_print("Card accepted");

// initialize timer
    timer0_init();

dataLength=0; //Resets
currentCardID=0;
ccounter=0;
return;
}
}
LCD_clear();
LCD_print("Card denied");
timer0_init();
}
}

// initialize timer, interrupt and variable
void timer0_init()
{
    // set up timer with prescaler = 1024
START_TIMER0;

// initialize counter
CLEAR_TIMER0;

// enable overflow interrupt
TIMSK |= (1 << TOIE0);

// enable global interrupts
sei();

// initialize overflow counter variable
tot_overflow = 0;
}

/* Database INIT*/
void DB_init()
```

```
{
//Zack
cardIDs[0]=708; //Se keyPress
passCodes[0]=202;

//Erik
cardIDs[1]=381;
passCodes[1]=196;
}

/* USART Functions */

//Hannes kod - 04 73 D0 79 F8 1111 1000
//Hgerda kod - 07 04 22 5D 92 F7 4D 81
//Zacks kod - 04 5E D3 98 FB 1111 1011
//zGerda kod - 07 04 2F A1 42 A2 3E 81

/* This function initiates the UART pins */
void uart_init ()
{
UBRRH=(BAUD_PRESCALE>>8);
UBRRL=BAUD_PRESCALE; //set baud rate
UCSRB|=(1<<TXEN)|(1<<RXEN); //enable receiver and transmitter
UCSRC|=(1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1); // 8bit data format
UCSRB|=(1<<RXCIE); // Enable the USART Receive Complete interrupt (USART_RXC)
sei(); // Enable the Global Interrupt Enable flag so that interrupts can be proces
}

/* function to send data */
void uart_transmit (char data)
{
while (!( UCSRA & (1<<UDRE))); //wait while register is free
LCD_send_data(data);
UDR = data; //load data in the register

}

/* function to receive data */
```

```
unsigned char uart_recieve (void)
{
while(!(UCSRA) & (1<<RXC)); //wait while data is being received
return UDR; //return 8-bit data
}


/* LCD Functions */

/* This function sends a command 'cmnd' to the LCD module*/
void LCD_send_command(unsigned char cmnd)
{
PORTC &= ~(1<<LCD_RS_PIN); //RS=0
PORTC &= ~(1<<LCD_RW_PIN); //RW=0
_delay_ms(5);
PORTD |= (1<<LCD_E_PIN); //E=1
_delay_ms(5);
PORTA = cmnd;
_delay_ms(5);
PORTD &= ~(1<<LCD_E_PIN); //E=0
}

/* This function sends the data 'data' to the LCD module*/
void LCD_send_data(unsigned char data)
{
PORTC &= ~(1<<LCD_RW_PIN); //RW=0
PORTC |= (1<<LCD_RS_PIN); //RS=1
_delay_ms(5);
PORTD |= (1<<LCD_E_PIN); //E=1
_delay_ms(5);
LCD_DATA_PORT = data;
_delay_ms(5);
PORTD &= ~(1<<LCD_E_PIN); //E=0
}

/* This function initiates the LCD pins */
void LCD_init()
{
```

```
DDRC = (1<<LCD_RW_PIN)|(1<<LCD_RS_PIN);
DDRD = (1<<LCD_E_PIN);
PORTC = 0x00;
PORTD = 0x00;
LCD_DATA_DDR = 0xFF;
LCD_DATA_PORT = 0xFF;
LCD_send_command(0x0F); //start
LCD_send_command(0x3C); //conf
LCD_send_command(0x01); //clear
}

/* Function to print a string on the LCD */
void LCD_print(char *string)
{
unsigned char i;

while(string[i]!=0)
{
LCD_send_data(string[i]);
i++;
}
}

/* Clears the LCD */
void LCD_clear(){
LCD_send_command(0x01); //clear
}

/* Moves LCD cursor */
```