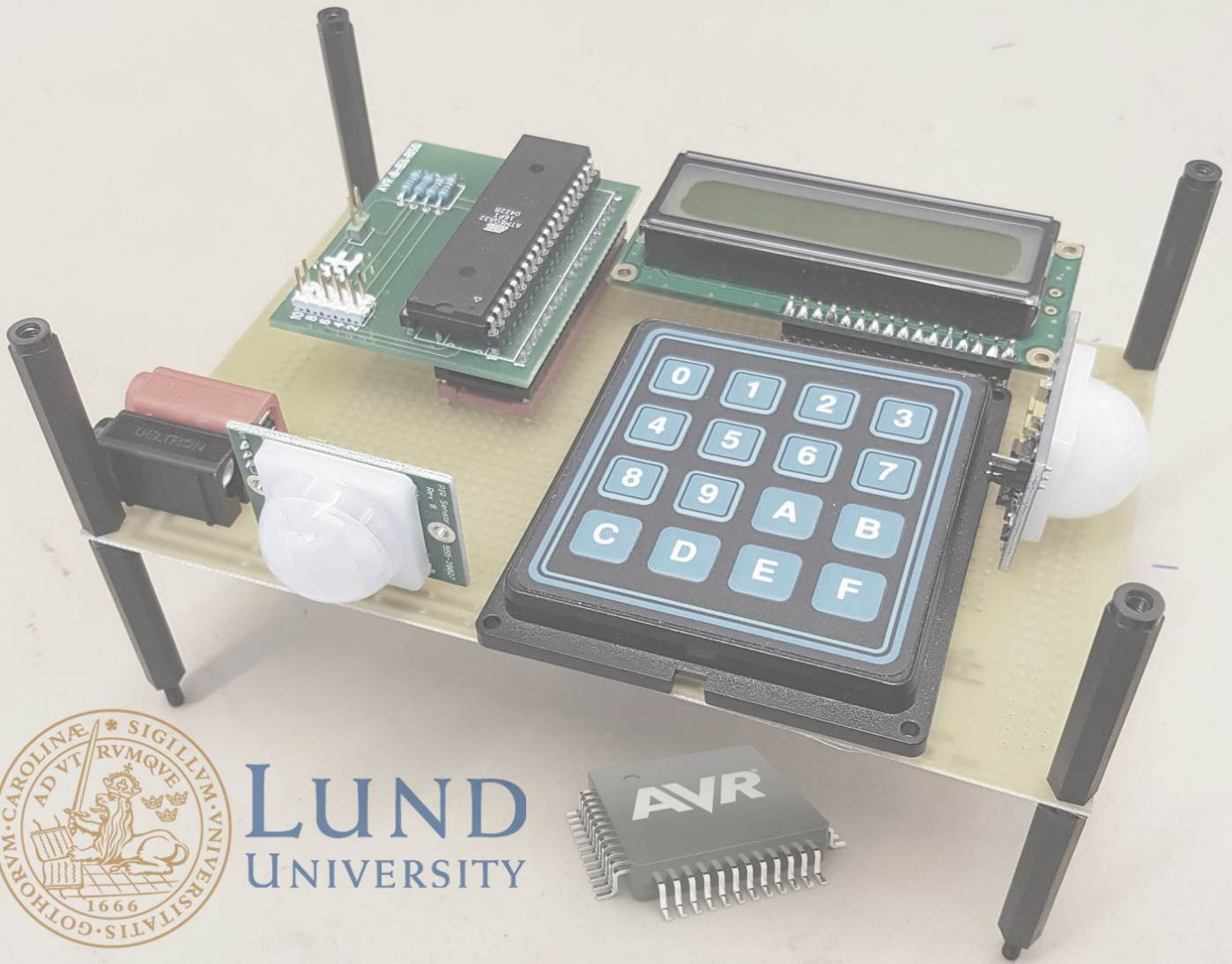


Larmanläggning



LUND
UNIVERSITY

Tor Berglund

Sebastian Svensson Bromert

Projektarbete inom Digitala Projekt EITF11
Institutionen för elektro- och informationsteknik
Handledare: Bertil Lindvall

2017-05-20

Abstract

In the project Digital Systems, Project Laboratory we were given the opportunity to freely choose a product to be developed. The microcontroller ATmega16 had a central role in the project, and it acted as an interface between all the hardware components. In this report, the process of developing a surveillance alarm is thoroughly described. The project gave the project members a deeper understanding of the connection between hardware and software in digital products.

1 Bakgrund	4
2 Kravspecifikation	5
2.1 Definitioner	5
2.2 Grundläggande krav	5
2.3 Användarfall	5
3 Konstruktion	6
3.1 Övergripande konstruktion	6
3.2 Blockschema	6
3.3 Komponenter	7
3.3.2 Display	7
3.3.3 Processor	7
3.3.4 Rörelsedetektorer	7
3.3.5 Knappsats	7
4 Genomförande	8
4.1 Planering	8
4.2 Förberedelse innan konstruktion	8
4.3 Konstruktion och utveckling av mjukvara	8
4.3 Implementation	8
5 Slutsats	9
6 Bilagor	10
Källkod	10

1 Bakgrund

Med den teknikutveckling som skett det senaste seklet möjliggörs digitaliserad övervakning av alltifrån kärnkraftverk, privata villor och matlagning. En av de bland privatpersoner vanligaste lösningar för övervakning av hem och hus, är en rörelsedetektor med ett fördröjt alarm. Den kan i sin tur vara kopplad till en central, som beslutar om vidare åtgärder efter att larmet har blivit utlöst. Inom ramen för kursen EITF11 Digitala Projekt har en enkel larmanordning konstruerats. Syftet med kursen är att få en grundläggande insikt hur kopplingen mellan hårdvara och mjukvara fungerar.

2 Kravspecifikation

2.1 Definitioner

Aktiverad/Avaktiverat larm anger huruvida systemet lyssnar via rörelsedetektorerna på rörelser.

Utlöst larm innebär att systemet har uppfattat en rörelse via rörelsedetektorerna, och användaren har ej angett korrekt kod inom tidsgränsen

2.2 Grundläggande krav

Krav 2.1.1 Systemet skall ha två rörelsedetektorer för att kunna uppfatta rörelser åt flera riktningar.

Krav 2.1.2 Sytemet skall ge adekvat återkoppling samt statusmeddelanden via den inkopplade skärmen.

2.3 Användarfall

AF1: Användare aktiverar larmet

Förutsättning: Systemet är strömförsörjt och larmet är avaktiverat

Framgångsrikt genomförande:

1. Användaren anger den fördefinierade fyrsiffriga koden via knappsatsen
2. Skärmen visar antal tidsenheter tills larmet aktiveras
3. Användaren lämnar rummet
4. Skärm visar meddelande "ACTIVATED"
5. Larmet är aktiverat

Undantagsfall:

1. Användaren anger fel kod
 - a. Skärm visar "W" efter koden, inväntar ny knapptryckning
 - b. Går ej vidare till (2)

AF2: Användare avaktiverar larmet

Förutsättning: Systemet är strömförsörjt och larmet är aktiverat enligt AF1

Framgångsrikt genomförande:

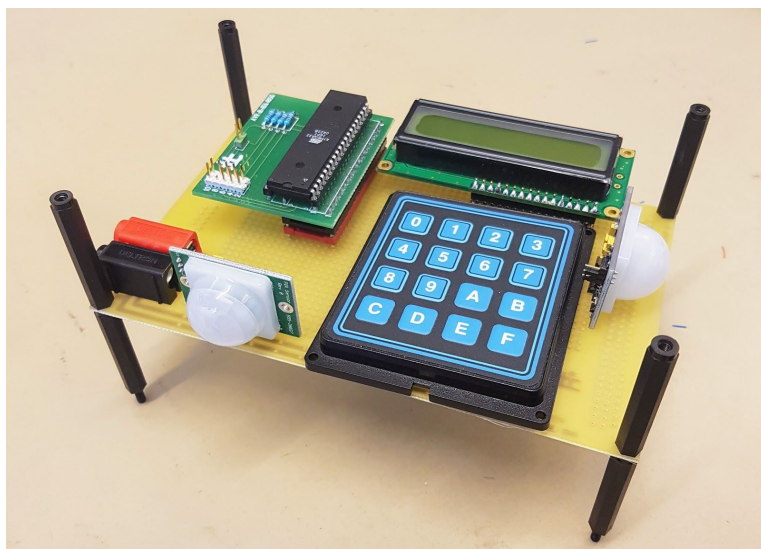
1. Användaren genomför en rörelse framför en rörelsedetektor
2. Display visar antal tidsenheter tills larmet utlöses
3. Användaren anger den fördefinierade fyrsiffriga koden via knappsatsen innan larmet utlöses
4. Skärmen visar meddelande "DEACTIVATED"
5. Systemet är redo för interaktion enligt AF1

Undantagsfall:

3. Användaren anger fel kod
 - c. Skärm visar "W" efter koden, inväntar ny knapptryckning
 - d. Om användare ej hinner ange korrekt kod inom den fördefinierade tiden enligt (2):
 - i. Larmet utlöses
 - ii. Skärmen visar meddelande "INTRUSION!"
 - iii. Systemet är redo för en ny interaktion enligt AF1

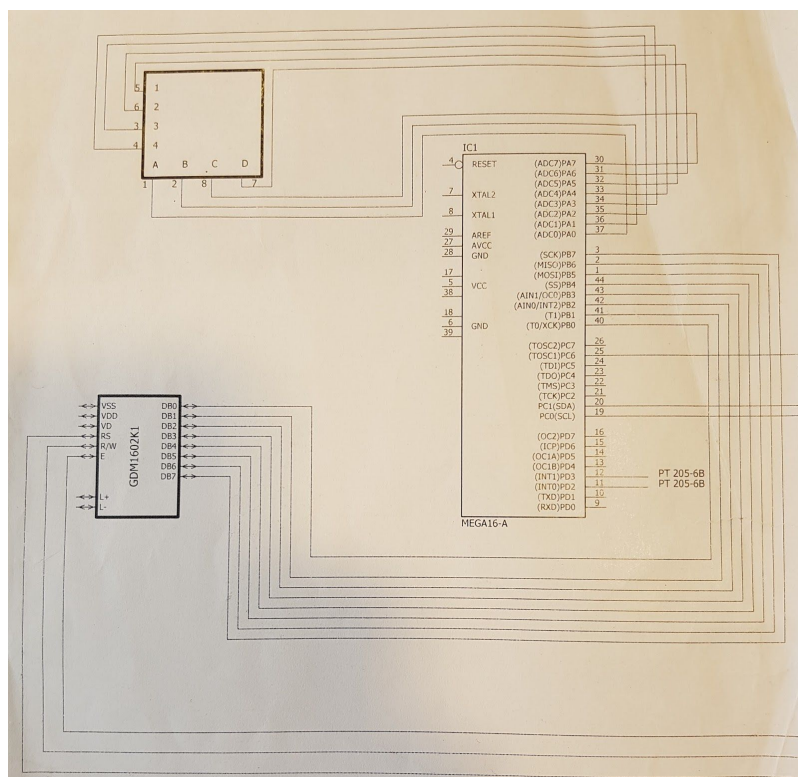
3 Konstruktion

3.1 Övergripande konstruktion



Systemet utgörs utav en skärm (2 rader), en hexadecimal knappsats, två rörelsedetektorer, en processor samt strömförsörjningskomponenter.

3.2 Blockschema



3.3 Komponenter

3.3.2 Display

Som display används Xiamen Ocular GDM1602K Dot-Matrix Alfanumerisk teckendisplay.

3.3.3 Processor

Som display används ATmega16 High-performance AVR 8-bit Microcontroller

3.3.4 Rörelsedetektorer

I systemet använder två stycken rörelsedetektorer av modellen Parallax PIR Motion Sensor Rev B

3.3.5 Knappsats

Systemet använder en knappsats av okänt fabrikat, med hexadecimal inmatning uppdelat på fyra rader och fyra kolumner.

4 Genomförande

4.1 Planering

Innan projektets start fördes en diskussion om vilket typ av projekt som skulle genomföras, och vad slutprodukten skulle bli. Sedan upprättades en kravspecifikation som har ej har varit föremål för större förändringar under arbetets gång. Slutligen upprättades ett tidsschema som tog hänsyn till faktorer härrörande från andra kurser och sociala åligganden.

4.2 Förberedelse innan konstruktion

Innan arbetet med konstruktionen kunde påbörjas, utfördes en grundlig inventering av de komponenter som fanns till förfogande. Arbetet delades upp, där en gruppmedlem ansvarade för hårdvaran, och den andra för mjukvaran. Sedermera upprättades ett blockschema av hårdvaruansvarig. Blockschemat fungerade som ritning under konstruktionen, samt som hjälpmedel till mjukvaruansvarig för att kunna utveckla psuedokod.

4.3 Konstruktion och utveckling av mjukvara

Med ett komplett blockschema kunde mjukvaruansvarig fortskrida med att utveckla källkoden. Utvecklingen skedde i Sublime Text på Mac OSX samt Atmel Studio 7.0 på Windows. Hårdvaruansvarig installerade komponenterna enligt blockschemat, och testade med hjälp av elektroniska verktyg att komponenterna var rätt installerade.

4.3 Implementation

Vid implementation och test blev det tydligt att exempelvis lösningen att använda funktionen `_delay_ms(millisecons)` ej var förenligt med att kunna ange koder via knappatsen simultant. Således skedde en revidering av källkoden, där användandet av `interrupts` infördes. Vidare visade test på ett nyckfullt beteende på vissa komponenter, som kunde rektifieras medelst `software debouncing` i ett enkelt utförande (tidsfördröjning). I projektets slutskede kunde test av samtliga användarfall genomföras utan problem med ena rörelsedetektorn. Den andra rörelsedetektorn påvisade ett beteende som ej var förutsägbart, vilket med stor sannolikhet kan härledas till ett hårdvarufel.

5 Slutsats

Projektet var en stor utmaning för projektgruppen, ty kunskapen om kopplingen mellan hårdvara och mjukvara var högst begränsad. En stor del av tiden för felsökning lades på att läsa och tyda databladen för komponenterna. Det hade underlättat att inkorporera information från databladen i ett tidigare skede under projektets gång, men det skulle erfordra tidigare erfarenhet av användande av datablad.

Något annat som blev uppenbart under projektet gång var att det finns ett stort egenvärde i att koden skall vara enkel att läsa för att underlätta felsökning. Efter varje "milstöple" i implementationen (exempelvis att en komponent fungerar som ursprungligen tänkt) sparades en säkerhetskopia av koden, och sedan förenklades den med bibehållen funktion. Detta resulterade i en kod som är förhållandevis logisk och enkel att läsa.

6 Bilagor

Källkod

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

char alarm_active=0;
char sensor1=0;
char sensor2=0;
char correct_input=0;
int pin[4] = {1,2,3,4};
int input[4]= {0,0,0,0};
int temp=0;
char val;
int number;
int got_input=0;
int digit_input=0;
int larm_triggered=0;
int vectorCount2=0;
int right_pin=0;
int sec=0;

int main(){
    DDRA = 0xE7;
    PORTA = 0x00;
    DDRC = 0xFF;
    DDRD = 0x00;
    DDRB = 0xFF;
    PORTB = 0xF0;

    disp_init();
    disp_home();
    disp_clear();

    // Configure PORTA as output
    // timer WGM mode 5 = CTC, divide 256 prescaler
    TCCR1B = (1<<CS11);

    // setup for compare interrupts
    TIMSK = (1<<TOIE1);

    TCNT1=20000;
```

```
//DDRD = 1<<PD2;          // Set PD2 as input (Using for interupt INT0)
//PORTD = 1<<PD2;        // Enable PD2 pull-up resistor
```

```
//DDRD = 1<<PD3;          // Set PD2 as input (Using for interupt INT0)
//PORTD = 1<<PD3;        // Enable PD2 pull-up resistor
```

```
MCUCR = 0b00001111;
```

```
// start the interrupts
sei();
```

```
while(1) {
```

```
    larm_triggered=0;
    if (alarm_active==0) {
```

```
        listenForPin();
```

```
        if (right_pin==1) {
            right_pin=0;
            input[0]=0;
            input[1]=0;
            input[2]=0;
            input[3]=0;
```

```
            startLeaveWarning();
```

```
        }
```

```
    }
```

```
while(alarm_active==1) {
```

```
    if (larm_triggered==1) {
```

```
        startArriveWarning();
```

```

        }
    }

}

ISR(TIMER1_OVF_vect) {
    // XOR PORTA with 0x02 to toggle the LSB
    sec++;
}

// timer0 overflow
ISR(INT0_vect) {
    _delay_ms(500);
    if (alarm_active==1) {

        larm_triggered=1;
    }
}

ISR(INT1_vect) {
    _delay_ms(500);
    if (alarm_active==1){

        larm_triggered=1;
    }
}

void startLeaveWarning() {
    disp_clear();
    int timer = 10;
    int i =10;
    int temp_sec = sec;
    int temp_sec2=0;

    while(sec<=temp_sec+10) {

```

```

        if (temp_sec2 != sec) {
            temp_sec2=sec;
            disp_clear();
            disp_writeCh('A');
            disp_writeCh('C');
            disp_writeCh('T');
            disp_writeCh('I');
            disp_writeCh('V');
            disp_writeCh('A');
            disp_writeCh('T');
            disp_writeCh('I');
            disp_writeCh('N');
            disp_writeCh('G');
            disp_writeCh(':');
            disp_writeCh(' ');
            disp_writeNum(10-(sec-temp_sec));

            if (sec-temp_sec ==10) {

                larm_triggered=0;
                writeActivated();
                _delay_ms(2000);
                disp_clear();
                alarm_active=1;
                GICR = (1<<INT1 | 1<<INT0);
            }

        }

    }

}

void startArriveWarning() {

    int temp_sec = sec;
    int temp_sec2=0;

    GICR = (0<<INT1 | 0<<INT0);
    while(sec<=temp_sec+10) {
        listenForPin();
    }
}

```

```

if (temp_sec2 != sec) {
    temp_sec2=sec;

    if (right_pin==0 && larm_triggered==1) {

        disp_home();
        disp_writeCh('A');
        disp_writeCh('L');
        disp_writeCh('A');
        disp_writeCh('R');
        disp_writeCh('M');
        disp_writeCh(' ');
        disp_writeCh('I');
        disp_writeCh('N');
        disp_writeCh(':');
        disp_writeCh(' ');
        disp_writeNum(10-(sec-temp_sec));

        if (sec-temp_sec==10) {
            writeIntrusion();
            _delay_ms(1000);
            disp_clear();
            alarm_active=0;
            larm_triggered=0;

        }
    }
    else {
        right_pin=0;
        input[0]=0;
        input[1]=0;
        input[2]=0;
        input[3]=0;
        writeDeactivated();
        _delay_ms(5000);
        disp_clear();
        alarm_active=0;
        larm_triggered=0;
    }
}
}

void writeActivated(){

```

```

disp_clear();
disp_writeCh('A');
disp_writeCh('C');
disp_writeCh('T');
disp_writeCh('I');
disp_writeCh('V');
disp_writeCh('A');
disp_writeCh('T');
disp_writeCh('E');
disp_writeCh('D');
}
//Skriver ut INACTIVATED på skärmen
void writeDeactivated(){
disp_home();
disp_writeCh('D');
disp_writeCh('E');
disp_writeCh('A');
disp_writeCh('C');
disp_writeCh('T');
disp_writeCh('I');
disp_writeCh('V');
disp_writeCh('A');
disp_writeCh('T');
disp_writeCh('E');
disp_writeCh('D');
}

void writeIntrusion(){
disp_clear();
disp_writeCh('I');
disp_writeCh('N');
disp_writeCh('T');
disp_writeCh('R');
disp_writeCh('U');
disp_writeCh('S');
disp_writeCh('I');
disp_writeCh('O');
disp_writeCh('N');
disp_writeCh('!');
}

int rightPinCode() {
if (input[0] == pin[0] && input[1] == pin[1] && input[2] == pin[2] &&
input[3] == pin[3]) {
write_cmd(0xC5);
disp_writeCh('R');
}
}

```

```

        return 1;
    } else {
        write_cmd(0xC5);
        disp_writeCh('W');
        disp_clear();
        return 0;
    }
}

void listenForPin(){
    _delay_ms(250);
    digit_input = checkButton();

    if (digit_input == 10) {

        return;
    }
    if (digit_input == 9) {

        vectorCount2 = 0;
    }
    if (vectorCount2 == 0) {
        disp_secondLine();
        disp_writeNum(digit_input);
        disp_home();
        input[0] = digit_input;
        vectorCount2 = 1;
    } else if (vectorCount2 == 1) {
        write_cmd(0xC1);
        disp_writeNum(digit_input);
        disp_home();
        input[1] = digit_input;
        vectorCount2 = 2;
    } else if (vectorCount2 == 2) {
        write_cmd(0xC2);
        disp_writeNum(digit_input);
        disp_home();
        input[2] = digit_input;
        vectorCount2 = 3;
    } else if (vectorCount2 == 3) {
        write_cmd(0xC3);
        disp_writeNum(digit_input);
        disp_writeCh('*');
        _delay_ms(200);
        disp_home();
        input[3] = digit_input;
    }
}

```



```

        if (rightPinCode()==1) {

            right_pin=1;
            vectorCount2 = 0;
        }
        else {
            right_pin=0;
            vectorCount2=0;
        }

    }
}

void set_pin(char port, char pin, char state){
    char set = 1 << pin;
    if(port == 'A'){
        set &= PORTA;
        if(set && !state){ //ändra från 1 -> 0
            PORTA ^= set;
        }
        if(set == 0 && state){ //ändra från 0 -> 1
            set = 1 << pin;
            PORTA ^= set;
        }
    }
    else if(port == 'B'){
        set &= PORTB;
        if(set && !state){ //ändra från 1 -> 0
            PORTB ^= set;
        }
        if(set == 0 && state){ //ändra från 0 -> 1
            set = 1 << pin;
            PORTB ^= set;
        }
    }
    else if(port == 'C'){
        set &= PORTC;
        if(set && !state){ //ändra från 1 -> 0
            PORTC ^= set;
        }
        if(set == 0 && state){ //ändra från 0 -> 1

```

```

        set = 1 << pin;
        PORTC ^= set;
    }
} else if(port == 'D'){
    set &= PORTD;
    if(set && !state){ //ändra från 1 -> 0
        PORTD ^= set;
    }
    if(set == 0 && state){ //ändra från 0 -> 1
        set = 1 << pin;
        PORTD ^= set;
    }
}
}
}

void write_cmd(char val) {
    PORTB=val;
    _delay_ms(5);
    set_pin('C', PC6, 0); //E går till låg
    set_pin('C', PC1, 0); //RW
    set_pin('C', PC0, 0); //RS väntar på kommando
    _delay_ms(5);
    set_pin('C', PC6, 1); //E går till hög
    _delay_ms(5);
    set_pin('C', PC6, 0); //E går till låg
}

//Tillkallas när vi vill skriva ut en bokstav/siffra/tecken
void disp_writeCh(char val) {
    PORTB=val;
    set_pin('C', PC6, 0); //E går till låg
    set_pin('C', PC1, 0); //RW
    set_pin('C', PC0, 1); //RS visar på skärm
    _delay_ms(5);
    set_pin('C', PC6, 1); //E går till hög
    _delay_ms(5);
    set_pin('C', PC6, 0); //E går till låg
}

//Tillkallas när vi vill skriva nummer
void disp_writeNum(int number) {
    if (number < 10) {
        if (number == 0) {
            disp_writeCh('0');
        }
        if (number == 1) {
            disp_writeCh('1');
        }
    }
}

```

```

        if (number == 2) {
            disp_writeCh('2');
        }
        if (number == 3) {
            disp_writeCh('3');
        }
        if (number == 4) {
            disp_writeCh('4');
        }
        if (number == 5) {
            disp_writeCh('5');
        }
        if (number == 6) {
            disp_writeCh('6');
        }
        if (number == 7) {
            disp_writeCh('7');
        }
        if (number == 8) {
            disp_writeCh('8');
        }
        if (number == 9) {
            disp_writeCh('9');
        }
    } else {
        int num1 = number/10;
        disp_writeNum(num1);
        int num2 = number%10;
        disp_writeNum(num2);
    }
}

void disp_clear() {
    write_cmd(0x01); // clear display
    _delay_ms(5);
    write_cmd(0x38); //functions set
    _delay_ms(5);
}

void disp_init() {

    write_cmd(0x0F); //display on
    _delay_ms(1);
    write_cmd(0x06); //Entry mode set
    _delay_ms(1);

}

```

```

void disp_home(){
    write_cmd(0x03); //flyttar markören hem
    _delay_ms(5);
}
void disp_secondLine() {
    write_cmd(0xC0); //Byter rad
    _delay_ms(5);
}

```

```

int checkRow() {
    DDRA = 0x0F;
    PORTA = 0xF0;
    val = PINA & 0xF0;
    if (val == 0xE0) {
        return checkCol(10);
    }
    if (val == 0xD0) {
        return checkCol(20);
    }
    if (val == 0xB0) {
        return checkCol(30);
    }
    if (val == 0x70) {
        return checkCol(40);
    }
    return 0;
}
//Läser av vilken kolumn som trycks in på PIN-kodsterminalen

```

```

int checkCol(int x) {
    set_pin('A', PA0, 0);
    set_pin('B', PA1, 0);
    set_pin('B', PA2, 0);
    set_pin('B', PA3, 1);
    val = PINA & 0xF0;
    if (val == 0xF0) {
        return (x + 1);
    }
    set_pin('A', PA0, 0);
    set_pin('A', PA1, 0);
    set_pin('A', PA2, 1);
    set_pin('A', PA3, 0);
    val = PINA & 0xF0;
    if (val == 0xF0) {
        return (x + 2);
    }
}

```

```

    }
    set_pin('A', PA0, 0);
    set_pin('A', PA1, 1);
    set_pin('A', PA2, 0);
    set_pin('A', PA3, 0);
    val = PINA & 0xF0;
    if (val == 0xF0) {
        return (x + 3);
    }
    set_pin('A', PA0, 1);
    set_pin('A', PA1, 0);
    set_pin('A', PA2, 0);
    set_pin('A', PA3, 0);
    val = PINA & 0xF0;
    if (val == 0xF0) {
        return (x + 4);
    }
}
int checkButton() {
    int button = checkRow();
    if (button == 11) {
        return 0;
    }
    if (button == 12) {
        return 8;
    }
    if (button == 13) {
        return 4;
    }
    if (button == 14) {
        return 0;
    }
    if (button == 21) {
        return 4;
    }
    if (button == 22) {
        return 9;
    }
    if (button == 23) {
        return 5;
    }
    if (button == 24) {
        return 1;
    }
    if (button == 31) {
        return 8;
    }
}

```

```
}  
if (button == 32) {  
    return 10;  
}  
if (button == 33) {  
    return 6;  
}  
if (button == 34) {  
    return 2;  
}  
if (button == 41) {///?  
    return 7;  
}  
if (button == 42) {//B  
    return 7;  
}  
if (button == 43) {  
    return 7;  
}  
if (button == 44) {  
    return 3;  
}  
return 10;
```

```
}
```