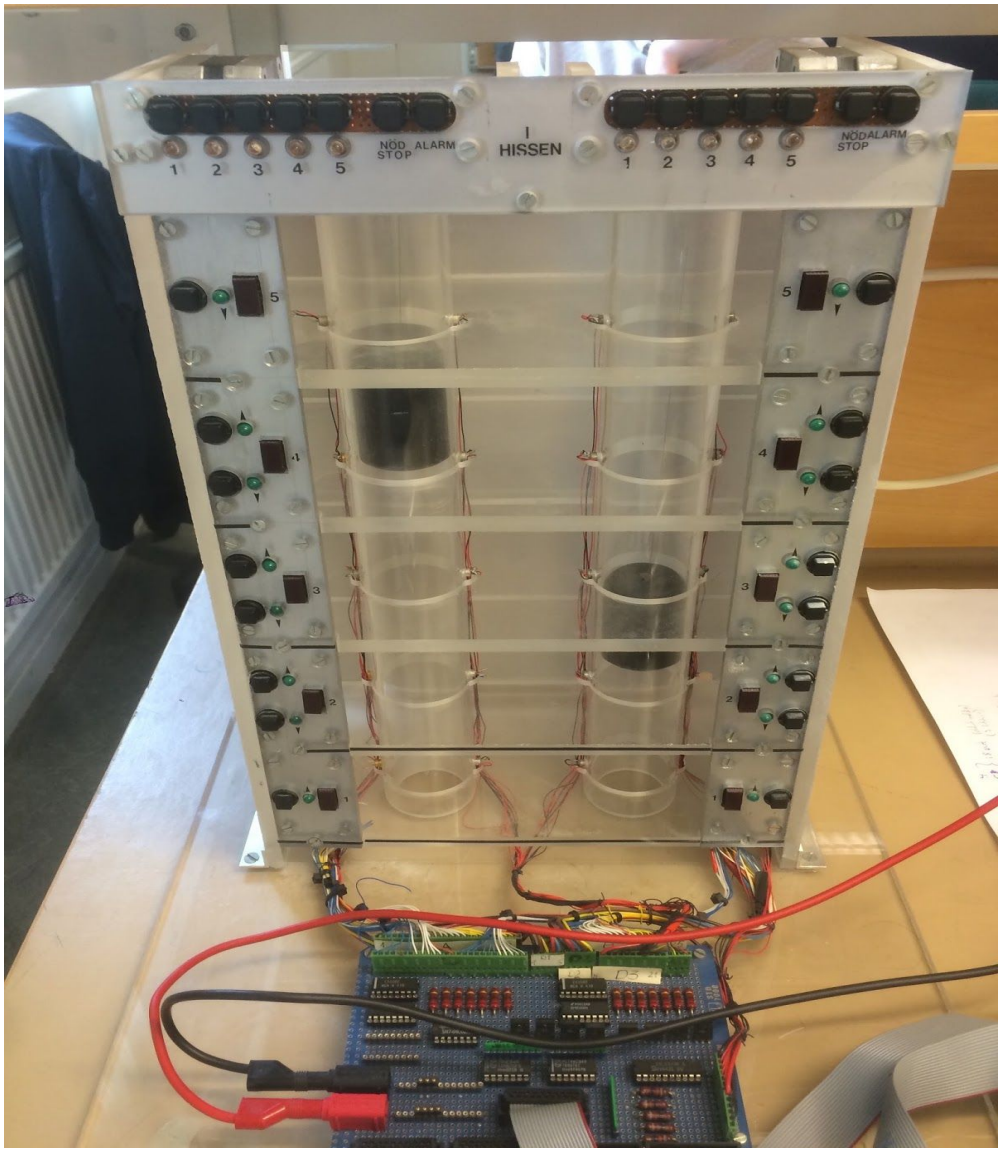
Lunds Tekniska Högskola
Elektro- och informationsteknik

Liftmaestro



Olle Gemfors, I13
Dayanand Sagar, I13
Mattias Wendler, I13

Handledare: Andreas Johansson, Bertil Lindvall

2016-05-20

Projektarbete, EITF11 Digitala projekt
Industriell ekonomi, Vårterminen 2016

Abstract

This is a project report in the course Digitala Projekt, EITF11. The purpose of the course is to get an insight in how industrial development works. Our aim with the project was to connect and code a working model of a lift. The lift hardware was provided by the course instructor.

The hardware was connected to a processor (AVR ATmega16) using five 74HC373 latches, two for the input signals and three for output signals. The processor was coded using a JTAG and Atmel Studio.

This resulted in a working lift model where the lift can serve all five floors in a logical order. Furthermore LED-lights are lit when pressed and are turned off when the lift arrives. Finally an LED-display updates and shows the current floor of the lift.

Innehållsförteckning

1. Inledning

1.1 Bakgrund

Detta projekt har genomförts under våren 2016 inom ramen för kursen Digitala projekt (EITF11) som en del av teknikprofilen System- och programvaruutveckling på Industriell ekonomi.

Syftet med kursenär att ge studenterna en inblick i hur industriellt utvecklingsarbete genomförs genom att studenterna får komma på en idé, designa en modell av denna idé, bygga modellen och slutligen testa och felsöka konstruktionen.

Kursen är indelad i två delar, där den första består av föreläsningar och laborationer och den andra av självständigt arbete som ska resultera i en fungerande prototyp.

1.2 Problembeskrivning

Som projekt i denna kurs valde vi att göra ett av de projekt som fanns att tillgå i kursens uppgiftskatalog. Detta projekt använder sig av en färdig modell av en hiss som tillhandages av handledarna. Hissen skulle kopplas till en processor som sedan skulle programmeras med programmeringsspråket C för att kunna svara på indata i form av knapptryckningar på modellen som får hissen att röra sig upp och ner och stanna på önskad våning.

1.3 Avgränsningar

Hissmodellen har, som man kan se på bilden på titelsidan, två hisschakt. På grund av vår ringa erfarenhet av denna typ av arbete begränsade vi oss till att bara koppla och programmera ett av hissens schakt.

2. Kravspecifikation

- Man ska kunna välja vilken våning som man vill åka till genom att använda knapparna i hallen samt i hissen.
- Lamporna vid varje våningsplan ska tändas när en knapp trycks ner och släckas när hissen stannar vid våningsplanet.
- Den aktuella våningen som hissen befinner sig på ska visas på en LED-display.
- Hissen ska kunna ändra sin rutt under vägen om någon på ett mellanliggande hissplan trycker att han/hon vill åt samma håll som hissen är på väg.
- Hissen ska ha ett optimerat kösystem så att man inte behöver vänta på hissen alltför länge.

3. Teori

3.1 Hårdvara

Detta avsnitt beskriver de komponenter som användes för att koppla hissmodellen. För kopplingsschema, se bilaga 1.

3.1.1 Processor

AVR ATmega16, Highperformance AVR 8bit Microcontroller.
Processorn har 40 pinnar där sammanlagt 21 användes för att koppla processorn till fem latchar och delvis till de kontakter som kommer från själva hissmodellen. Dessutom användes några pinnar för att ge processorn ström, möjlggöra programmering genom JTAG och till att styra vilken latch systemet ska kommunicera med vid varje given tidpunkt.

3.1.2 Latchar

74HC373 octal transparent D-type latches with 3-state outputs
Eftersom det behövdes fler pinnar än vad processorna hade för att hantera alla in- och utsignaler var vi tvungna att använda latchar för att skapa fler kontakter för in- och output till processorn. Två latchar användes för att se till att alla insignaler kom till processorn och tre latchar användes för att se till att processorn kunde skicka all utdata som den behövde.

3.1.3 Hissmodell

Handledarna tillhandahöll hissmodellen som var färdigbyggd sedan tidigare och även redan använd vid ett par tillfällen. Hissmodellens uppbyggnad beskrevs förenklat i ett antal datablad. Det vi använde rent kopplingsmässigt från modellen var tre utkontakter som skulle kopplas till processorn och programmeras. Modellen består sedan av två schakt (där vi som tidigare nämnts valde att endast använda ett) och en liten hiss i varje schakt som drivs av varsin motor.

3.1.4 JTAG

Atmel JTAG ICE
Det verktyg som användes för att ladda och exekvera mjukvaran i processorn var en JTAG. Koden skrevs på programmeringsspråket C på dator och överfördes via en USB-sladd till JTAGen som laddade in det i processorn.

3.2 Mjukvara

Programmet skrevs i språket C i utvecklingsplattformen Atmel Studio 6.1. Programkoden byggs upp av 14 olika metoder. De skrevs gradvis en och en för att säkerhetsställa att alla funktioner i hissen fungerade innan allt sattes ihop. En del av metoderna beskrivs kort här nedan, för komplett programkod se bilaga 2.

Startupmode(); En metod för att sätta alla portar och deras in- respektive utgångar till korrekta värden så att hissmodellen blir stabil.

executeQueue(); En metod för att bestämma vilken våning hissen ska till och få den att köra dit baserat på prioritetskön.

checkTKinHall(); En metod för att läsa av knapptryckningar och tända lamporna i hallen.

CheckTKinHiss(); En metod för att läsa av knapptryckningar och tända lamporna i hissen.

CheckSensors(); En metod för att läsa av sensorerna inuti hisschaktet.

goDown(); och goUp(); Metoder för att få hissen att antingen åka nedåt eller uppåt.

dispUpdate(); En metod för att uppdatera LED-displayerna.

addToPrio();, removePrio();, removeSpecificPrio();, checkLogicinHall();, refreshPrio(); och hissUpdate(); Metoder för att behandla bland annat prioritetskön och det som hisslogiken bygger på.

4. Utförande

4.1 Planering

Arbetet började med att vi fick klart för oss idén kring vilken vi ville arbeta. Inspiration hämtades från kursens uppgiftskatalog och vad vi skulle göra bestämdes. Därefter skrevs en kravspecifikation och vi bestämde vilka komponenter som skulle behövas. Slutligen skapades ett kopplingsschema.

4.2 Konstruktion

Efter att kopplingsschemat blev godkänt av handledarna kunde komponenterna hämtas ut. Därefter kopplades allt enligt kopplingsschemat.

4.3 Programmering

När allt var kopplat och klart skrevs programkoden för hur hissen skulle fungera och reagera på interaktionerna på hissmodellen.

4.4 Test och felsökning

Slutligen testades allting så att alla kopplingar och programmeringen gjorts korrekt så att hissmodellen betedde sig enligt kravspecifikationen.

5. Resultat

Resultatet blev en i princip väl fungerande hissmodell. Vid optimala förhållanden, det vill säga när inga störningar ligger på systemet, fungerar modellen mycket tillfredsställande. Då kan hissen besöka alla våningar i en logisk och rättvis ordningsföljd. Om hissen är påväg uppåt och passerar en våning där någon tryckt på uppåtknappen stannar hissen. Däremot om hissen passerar en våning där någon tryckt på nedåtknappen och hissen är på väg uppåt så stannar hissen inte. Dessutom tänds och släcks alla lysdioder som de ska och displayen uppdateras till att visa rätt våning. Det finns en bugg som inträffar om en knapp i hallen trycks ned på samma våning som hissen har stannat på vilket resulterar i att systemet kraschar. Tyvärr händer det också mer ofta än sällan att hårdvaran i hissmodellen fallerar. Detta gör att fel våning ibland registreras i systemet och hissen antingen stannar på fel ställe eller fortsätter upp eller ned i all oändlighet.

6. Diskussion och slutsats

Projektet har överlag flutit på bra och varje vecka har nya framsteg gjorts. I början var det svårt att sätta sig in i och tolka all ny information. När vi väl förstod hur vi skulle gå tillväga gick själva kopplingen smärtfritt och den har i efterhand inte heller gett oss några problem. Vårt största misstag var att vi i inledningsvis gav modellen 12V istället för 5V vilket brände en del komponenter. Förutom det var all input och output var nästan korrekt kopplad från början vilket underlättade det senare arbetet.

Programmeringsfasen var det som var mest tidskrävande och vållade flest problem. Dels för att det var ett nytt programmeringsspråk vilket gav en del logiska fel. Det var viktigt att håla reda på vilka portar som skulle göras tillgängliga och vilka latchar som skulle läsas. Dels var programmeringen även svår för att hårdvaran i sig inte är fri från störningar till följd av sin stigande ålder. Detta gjorde att trots att vi både kopplat och programmerat rätt betedde sig inte modellen som den skulle. Det går då och då strömmar i hisskretsen som ger ettor där det borde vara nollor. Något vi har försökt skydda oss mot genom att koppla och koda bort problemet. Det som gäckade oss allra mest var att den tredje sensorn i hissmodellen då och då ger en etta trots att hissen inte bryter sensorn, detta gör att systemet tror att hissen är på en annan våning än vad den egentligen är.

Utöver ovan nämnda problem händer också att hissen beter sig annorlunda från gång till gång trots att koden är exakt likadan. Det har helt enkelt stundvis varit ett väldigt frustrerande och oberäkneligt projekt. Vi tror att det mesta har att göra med hissmodellens ålder då vi drabbats av icke fungerande komponenter, dåliga lödningar och lösa sladdar. Vi kan också konstatera att även om vår hisslogik är bra är den inte perfekt. Det kan uppstå situationer som inte är optimala ur ett effektiveringsperspektiv. Hade vi tagit projektet vidare hade vi fokuserat på att förfina programkoden och optimera hisslogiken.

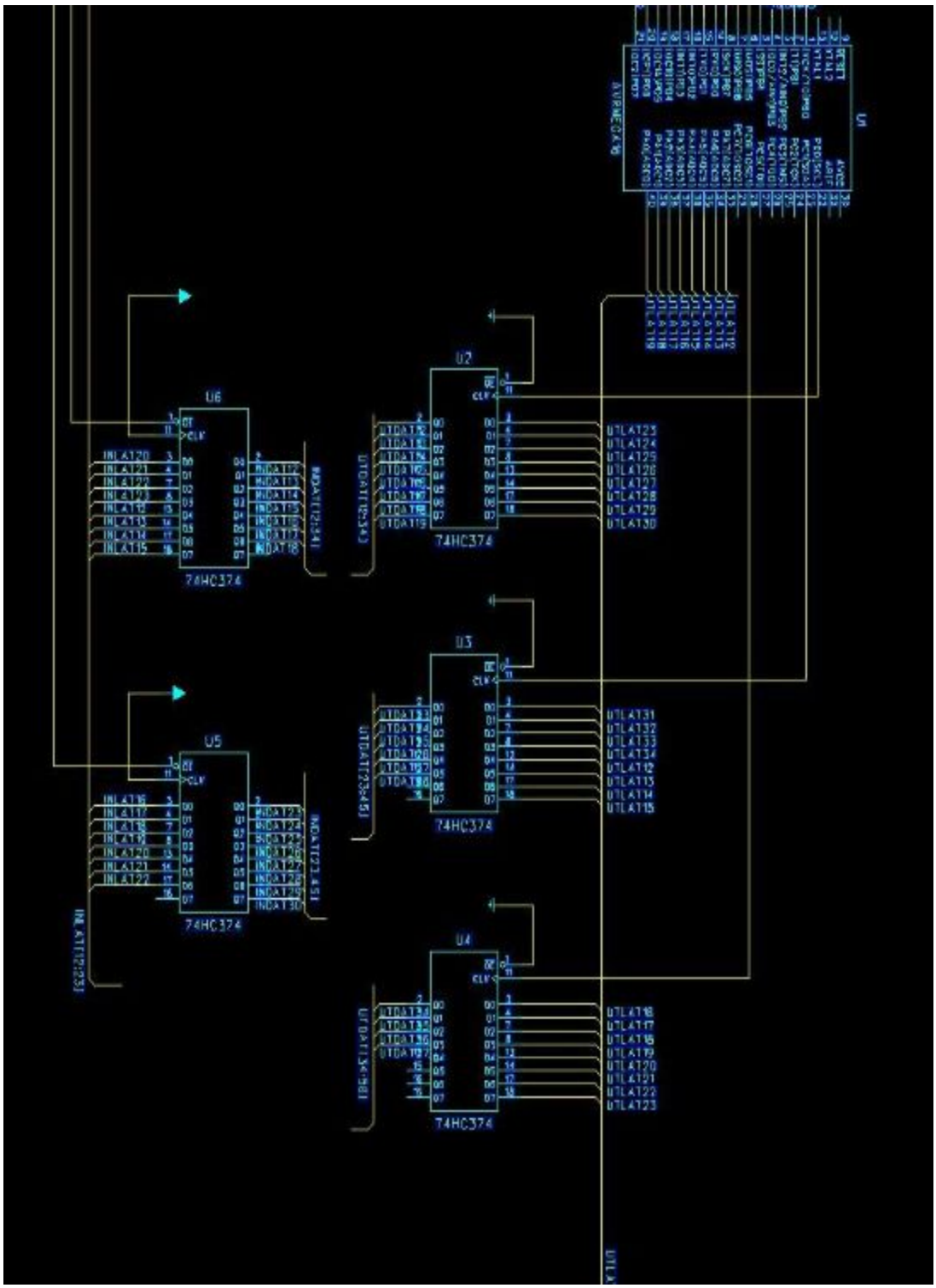
7. Källförteckning

Digitala projekt, Atmel, 8-bit AVR Microcontroller with 16K byte In-system programmable flash, ATmega16, ATmega16L

Datablad, 74HC373 Octal 3-state NonInverting D-latch

8. Appendix

Bilaga 1, Kopplingsschema



Bilaga 2, programkoden

```
/*
 * hissen.c
 *
 * Created: 2016-04-26 11:44:02
 *  Author: digpi17
 */



#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <avr/delay.h>



int alarm = 0;
int stop = 1;
int inMotion = 0;
int goingUp = 0;
int goingDown = 0;
int currentFloor = 0;
int previousFloor = 0;
int targetFloor = 0;
int hissQ[5] = {0, 0, 0, 0, 0};
int queueDown[4] = {0, 0 ,0 ,0};
```

```
int queueUp[4] = {0, 0, 0, 0};

int sensorVect[5] = {0, 0, 0, 0, 0};

int prio[8] = {0,0,0,0,0,0,0,0};

int direction;

int waitTime;

volatile uint8_t tot_overflow;


char LedHall = 0;

char LedHiss = 0;

char sensorInput = 0;

char hallInput = 0;

char hissInput = 0;

int start = 1;

int nodStop = 0;


char temp = 0;

char disp = 0;


void Startupmode();

void checkTKinHall();

void executeQueue();

void CheckTKinHiss();

void CheckSensors();

void goDown();

void goUp();
```

```
void dispUpdate();

void addToPrio();

void removePrio();

void removeSpecificPrio();

void checkLogicinHall();

void refreshPrio();

void hissUpdate();


int main(void)

{

        Startupmode();

        currentFloor = 4;

        //targetFloor = 5;

   while(start)

   {


                checkTKinHall();

                CheckTKinHiss();

                while(nodStop){


                }

                executeQueue();

                hissUpdate();

                while(waitTime){

                        CheckTKinHiss();
```

```
                checkTKinHall();
                refreshPrio();

            }
            TCCR1B = 0;
            cli();
            //start = 1;

    }
}

void timer1_init(void){
    TCCR1B |= (1 << CS11);
    TCNT1 = 0;
    TIMSK |= (1 << TOIE1);
    sei();
    tot_overflow = 0;
}

ISR(TIMER1_OVF_vect){
    tot_overflow++;
    if(tot_overflow >= 7){
        waitTime = 0;
        tot_overflow = 0;
    }
```

```
}


void checkTKinHall(void){
        DDRB = 0b11100000; //Sätt PB5 och PB6 och PB7 som output
        PORTB = 0b10100000; // sätt PB6 som output 1 och PB5 som output 0. Port D läser nu från hallen.
        hallInput = PIND;


        LedHall = hallInput | LedHall;
        PORTC = 0b00000001;
        PORTA = LedHall;




        if((hallInput & 0b00000001) != 0){
                queueUp[0] = 1;
                addToPrio(1);
        }


        if((hallInput & 0b00000010) != 0){
                queueDown[0] = 1;
                addToPrio(2);



        }
```

```
if((hallInput & 0b00000100) != 0){
        queueUp[1] = 1;
        addToPrio(3);


}

if((hallInput & 0b00001000) != 0){
        queueDown[1] = 1;
        addToPrio(4);
}

if((hallInput & 0b00010000) != 0){
        queueUp[2] = 1;
        addToPrio(5);

}

if((hallInput & 0b00100000) != 0){
        queueDown[2] = 1;
        addToPrio(6);

}
```

```
        if((hallInput & 0b01000000) != 0){
                queueUp[3] = 1;
                addToPrio(7);

        }

        if((hallInput & 0b10000000) != 0){
                addToPrio(8);
                queueDown[3] = 1;

        }

        PORTB = 0b11100000;
        PORTC = 0b00000000;

}

void executeQueue(void){
        if(inMotion == 0 ){ //&& Något som ser till att denna inte körs om vi läser från hallen istället -> remove körs för ofta//
                int x = prio[0];
                switch (x)
                {

                        case 1:
```

```
targetFloor = 1;

direction = 1;

//removePrio(); // borde kanske ta bort denna funktionen, istället gör vi så att
```
den först tas ur prion när detta faktiskt är uppnått. Alltså tas prion bort då vi stannar på första våningen. På så vis kan vi inte lyckas fylla på med en 1a i prioque direkt.

```
break;


case 2:

targetFloor = 2;

direction = 0;

//removePrio();

break;


case 3:

targetFloor = 2;

direction = 1;

//removePrio();

break;


case 4:

targetFloor = 3;

direction = 0;

//removePrio();

break;


case 5:
```

```
                targetFloor = 3;
                direction = 1;
                //removePrio();
                break;

                case 6:
                targetFloor = 4;
                direction = 0;
                //removePrio();
                break;

                case 7:
                targetFloor = 4;
                direction = 1;
                //removePrio();
                break;

                case 8:
                targetFloor = 5;
                direction = 0;
                //removePrio();
                break;
        }

if(currentFloor > targetFloor && targetFloor != 0){
```

```
                stop = 1;
                goDown();
        }else if( targetFloor > currentFloor && targetFloor != 0){
                stop = 1;
                goUp();
        }

        }

}

void checkLogicinHall(void){

        if(goingDown == 1 && inMotion == 1){
                int i = 0;
                int v;
                for(i = 0; i < 8; i = i +1){
                        v = prio[i];
                        switch(v){
                                case 2:
                                if(currentFloor > 2 && targetFloor < 2){
                                        targetFloor = 2;
                                }
                                break;
```

```
                    case 4:
                    if(currentFloor > 3 && targetFloor < 3){
                        targetFloor = 3;
                    }
                    break;

                    case 6:
                    if(currentFloor > 4 && targetFloor < 4){
                        targetFloor = 4;
                    }
                    break;
                }

            }

}

if(goingUp == 1 && inMotion){
            int i = 0;
            int v;
            for(i = 0; i < 8; i = i +1){
                v = prio[i];
                switch(v){
                    case 3:
                    if(currentFloor < 2 && targetFloor > 2){
```

```
                    targetFloor = 2;
                }
                break;

                case 5:
                if(currentFloor < 3 && targetFloor > 3){
                    targetFloor = 3;
                }
                break;

                case 7:
                if(currentFloor < 4 && targetFloor > 4){
                    targetFloor = 4;
                }
                break;
            }
        }
    }
}


void CheckTKinHiss(void){
    DDRB = 0b11100000;
    PORTB = 0b11000000;
    hissInput = PIND;
```

```
hissInput = hissInput << 3;
hissInput = hissInput >> 3;

LedHiss = hissInput | LedHiss;
PORTC = 0b00000010;
PORTA = LedHiss;

hissInput = PIND;

if((hissInput & 0b00000001) != 0){
        hissQ[0] = 1;
}

if((hissInput & 0b00000010) != 0){
        hissQ[1] = 1;
}

if((hissInput & 0b00000100) != 0){
        hissQ[2] = 1;
}

if((hissInput & 0b00001000) != 0){
        hissQ[3] = 1;
}
```

```
        if((hissInput & 0b00010000) != 0){
                hissQ[4] = 1;
        }

        if((hissInput & 0b00100000) != 0){
                nodStop = 1;
                stop = 0;
        }

        if((hallInput & 0b01000000) != 0){
                alarm = 1;
        }

        PORTB = 0b11100000;
        PORTC = 0b00000000;
}

void CheckSensors(void){
        DDRB = 0b11100000;
        sensorInput = PINB;

        char temp;
        temp = sensorInput << 3;
        sensorInput = temp >> 3;
```

```
if((sensorInput & 0b00000001 ) != 0){
        sensorVect[0] = 1;
}else{
        sensorVect[0] = 0;
}

if((sensorInput & 0b00000010) != 0 && goingUp == 1){
        sensorVect[1] = 1;
        if(previousFloor == 2 || previousFloor == 0){
        previousFloor = 1;
        }
}else if((sensorInput & 0b00000010) != 0 && goingDown == 1){
        sensorVect[1] = 1;
        if(previousFloor == 1 || previousFloor == 3 || previousFloor == 0){
        previousFloor = 2;
        }
}else{
        sensorVect[1] = 0;
}

if((sensorInput & 0b00000100) != 0 && goingUp == 1){
        sensorVect[2] = 1;
        if(previousFloor == 1 || previousFloor == 3 || previousFloor == 0){
```

```
            previousFloor = 2;
            }
    }else if((sensorInput & 0b00000100) != 0 && goingDown == 1){
            sensorVect[2] = 1;
            if(previousFloor == 4 || previousFloor == 2 || previousFloor == 0){
            previousFloor = 3;
            }
    }else{
            sensorVect[2] = 0;
    }

    if((sensorInput & 0b00001000) != 0 && goingUp == 1){
            sensorVect[3] = 1;
            if(previousFloor == 2 || previousFloor == 4 || previousFloor == 0){
            previousFloor = 3;
            }
    }else if((sensorInput & 0b00001000) != 0 && goingDown == 1){
            sensorVect[3] = 1;
            if(previousFloor == 5 || previousFloor == 3 || previousFloor == 0){
            previousFloor = 4;
            }
    }else{
            sensorVect[3] = 0;
    }
```

```
if((sensorInput & 0b00010000) != 0 && goingUp == 1){
        sensorVect[4] = 1;
        if(previousFloor == 3 || previousFloor == 0){
        previousFloor = 4;
        }
}else if((sensorInput & 0b00010000) != 0 && goingDown == 1){
        sensorVect[4] = 1;
        if(previousFloor == 4 || previousFloor == 0){
        previousFloor = 5;
        }

}else{
        sensorVect[4] = 0;
}

if(sensorVect[1] == 0 && goingUp == 1 && previousFloor == 1 && currentFloor == 1){
        currentFloor = 2;
}else if( sensorVect[1] == 0 && goingDown == 1 && previousFloor == 2 && currentFloor ==
2){
        currentFloor = 1;
}

if(sensorVect[2] == 0 && goingUp == 1 && previousFloor == 2 && currentFloor == 2){
        currentFloor = 3;
```

```
        }else if( sensorVect[2] == 0 && goingDown == 1 && previousFloor == 3 && currentFloor ==
3){
                currentFloor = 2;
        }

        if(sensorVect[3] == 0 && goingUp == 1 && previousFloor == 3 && currentFloor == 3){
                currentFloor = 4;
        }else if( sensorVect[3] == 0 && goingDown == 1 && previousFloor == 4 && currentFloor ==
4){
                currentFloor = 3;
        }

        if(sensorVect[4] == 0 && goingUp == 1 && previousFloor == 4 && currentFloor == 4){
                currentFloor = 5;
        }else if( sensorVect[4] == 0 && goingDown == 1 && previousFloor == 5 && currentFloor ==
5){
                currentFloor = 4;
        }

        if(sensorVect[0] == 1){
                stop = 0;
        }

        if((currentFloor == targetFloor) && (sensorVect[0] == 0) && (sensorVect[1] == 0)  &&
(sensorVect[2] == 0) && (sensorVect[3] == 0) && (sensorVect[4] == 0)){
                stop = 0;
```

```
waitTime = 1;
int f = currentFloor;
switch(f){
        PORTC = 0b00000010;
        case 1:
        LedHiss = 0b11111110 & LedHiss;
        PORTA = LedHiss;
        break;

        case 2:
        LedHiss = 0b11111101 & LedHiss;
        PORTA = LedHiss;
        break;

        case 3:
        LedHiss = 0b11111011 & LedHiss;
        PORTA = LedHiss;
        break;

        case 4:
        LedHiss = 0b11110111 & LedHiss;
        PORTA = LedHiss;
        break;

        case 5:
```

```
            LedHiss = 0b11101111 & LedHiss;
            PORTA =  LedHiss;
            break;
}

PORTC = 0b00000000;
PORTA = 0b00000000;

if(goingDown == 1 && goingUp == 0){
            int f = currentFloor;
            switch(f){
                        PORTC = 0b00000001;
                        case 1:
                        LedHall = 0b11111110 & LedHall;
                        PORTA = LedHall;
                        break;

                        case 2:
                        LedHall = 0b11111101 & LedHall;
                        PORTA = LedHall;
                        break;

                        case 3:
                        LedHall = 0b11110111 & LedHall;
                        PORTA = LedHall;
```

```
                    break;


                    case 4:
                    LedHall = 0b11011111 & LedHall;
                    PORTA = LedHall;
                    break;


            }
}

if(goingDown == 0 && goingUp == 1){
            int f = currentFloor;
            switch(f){
                    PORTC = 0b00000001;
                    case 2:
                    LedHall = 0b11111011 & LedHall;
                    PORTA = LedHall;
                    break;


                    case 3:
                    LedHall = 0b11101111 & LedHall;
                    PORTA = LedHall;
                    break;


                    case 4:
```

```
                    LedHall = 0b10111111 & LedHall;
                    PORTA = LedHall;
                    break;

                    case 5:
                    LedHall = 0b01111111 & LedHall;
                    PORTA = LedHall;
                    break;
                }
            }

            PORTC = 0b00000000;
            PORTA = 0b00000000;
        }

        PORTB = 0b11100000;
        PORTC = 0b00000000;
        timer1_init();
}

void Startupmode(void){
        DDRB = 0b11100000;
        DDRC = 0b01000011;
        DDRD = 0b00000000;
        DDRA = 0b11111111;
```

```
        PORTC = 0b00000001;
        PORTA = 0b00000000;
        PORTC = 0b00000010;
        PORTA = 0b00000000;
        PORTC = 0b01000000;
        PORTA = 0b00000000;

}

void goDown(void){
        goingDown = 1;
        inMotion = 1;
        goingUp = 0;
        if(stop == 1){
        while(stop){

                PORTC = 0b01000000;

                PORTA = 0b00000110;
                _delay_ms(10);

                PORTA = 0b00001010;
                _delay_ms(10);
```

```
            PORTA = 0b00001001;
            _delay_ms(10);

            PORTA = 0b00000101;
            _delay_ms(10);

            checkTKinHall();
            CheckTKinHiss();
            CheckSensors();
            dispUpdate();
            hissUpdate();
            checkLogicinHall();

            PORTB = 0b11100000;
            PORTC = 0b00000000;
}
int i;
for(i = 0; i < 15; i = i + 1){

            PORTC = 0b01000000;

            PORTA = 0b00000110;
            _delay_ms(10);

            PORTA = 0b00001010;
```

```
                _delay_ms(10);


                PORTA = 0b00001001;
                _delay_ms(10);


                PORTA = 0b00000101;
                _delay_ms(10);


                checkTKinHall();
                CheckTKinHiss();
                dispUpdate();


                PORTB = 0b11100000;
                PORTC = 0b00000000;
        }
        }
        inMotion = 0;
        }


void goUp(void){
        goingUp = 1;
        inMotion = 1;
        goingDown = 0;
        while(stop){
```

```
PORTC = 0b01000000;

PORTA = 0b00000101;
_delay_ms(10);


PORTA = 0b00001001;
_delay_ms(10);


PORTA = 0b00001010;
_delay_ms(10);


PORTA = 0b00000110;
_delay_ms(10);

checkTKinHall();
CheckTKinHiss();
CheckSensors();
dispUpdate();
hissUpdate();
checkLogicinHall();

PORTB = 0b11100000;
```

```
            PORTC = 0b00000000;


        }
        inMotion = 0;
}

void dispUpdate(void){
        PORTC = 0b00000010;
        char temp;
        temp = LedHiss << 3;
        LedHiss = temp >> 3;



        if(currentFloor == 1){
            LedHiss = LedHiss | 0b00100000;
            PORTA = LedHiss;
        }

        if(currentFloor == 2){
            LedHiss = LedHiss | 0b01000000;
            PORTA = LedHiss;
        }

        if(currentFloor == 3){
            LedHiss = LedHiss | 0b01100000;
```

```
            PORTA = LedHiss;
        }

        if(currentFloor == 4){
            LedHiss = LedHiss | 0b10000000;
            PORTA = LedHiss;
        }

        if(currentFloor == 5){
            LedHiss = LedHiss | 0b10100000;
            PORTA = LedHiss;
        }

        PORTB = 0b11100000;
        PORTC = 0b00000000;
        PORTA = 0b00000000;
}

void addToPrio(int x){

int count;
int prioValue;
for(count = 0; count < 8; count = count + 1){
        prioValue = prio[count];
        if(prioValue == x){
```

```
                count = 8;
        }
        if(prioValue == 0){
                prio[count] = x;
                count = 8;
        }
}


}

void removePrio(void){
int temp[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int t = 0;
int l;
for(l = 1; l < 8; l = l + 1){
        t = prio[l];
        temp[l-1] = t;
}

int j;
for(j = 0; j < 8; j = j + 1){
        t = temp[j];
        prio[j] = t;
}
```

```
}

void removeSpecificPrio(int pos){
	int i;
	int tempValue;
	int tempV[8] = {0,0,0,0,0,0,0,0};
	for(i = 0; i < pos; i = i +1){
		tempValue = prio[i];
		tempV[i] = tempValue;
	}

	for(i = pos + 1; i < 8; i = i + 1){
		tempValue = prio[i];
		tempV[i-1] = tempValue;
	}

	for(i = 0; i < 8; i = i + 1){
		tempValue = tempV[i];
		prio[i] = tempValue;
	}

}

void test(void){
	currentFloor = 2;
```

```
        targetFloor = 5;
        goUp();
        _delay_ms(500);
        stop = 1;
        currentFloor = 5;
        targetFloor = 1;
        goDown();
        _delay_ms(500);
        stop = 1;
        currentFloor = 1;
        targetFloor = 2;
        goUp();
        _delay_ms(500);
        stop = 1;
        currentFloor = 2;
        targetFloor = 4;
        goUp();
        _delay_ms(500);
        stop = 1;
        currentFloor = 4;
        targetFloor = 2;
        goDown();
}

void refreshPrio(void){
```

```
int i;
int e;
for(i = 0; i < 8; i = i + 1){
        e = prio[i];
        if(goingDown == 1){
                switch(e){
                        case 1:
                        if(currentFloor == 1){
                                removeSpecificPrio(i);
                                hissQ[0] = 0;
                        }
                        break;

                        case 2:
                        if(currentFloor == 2){
                                removeSpecificPrio(i);
                                hissQ[1] = 0;
                        }
                        break;

                        case 4:
                        if(currentFloor == 3){
                                removeSpecificPrio(i);
                                hissQ[2] = 0;
                        }
```

```
                    break;


                    case 6:
                    if(currentFloor == 4){
                    removeSpecificPrio(i);
                    hissQ[3] = 0;
                    }
                    break;
            }
}

if(goingUp == 1){
            switch(e){
                    case 3:
                    if(currentFloor == 2){
                            removeSpecificPrio(i);
                            hissQ[1] = 0;
                    }
                    break;


                    case 5:
                    if(currentFloor == 3){
                            removeSpecificPrio(i);
                            hissQ[2] = 0;
                    }
```

```
                                        break;


                                        case 7:
                                        if(currentFloor == 4){
                                                removeSpecificPrio(i);
                                                hissQ[3] = 0;
                                        }
                                        break;


                                        case 8:
                                        if(currentFloor == 5){
                                                removeSpecificPrio(i);
                                                hissQ[4] = 0;
                                        }
                                        break;
                                }
                        }
}
}


void hissUpdate(void){
        if(hissQ[0] == 1 && currentFloor > 1){
                addToPrio(1);
        }
```

```
if(hissQ[1] == 1 && currentFloor > 2){
        addToPrio(2);
}

if(hissQ[1] == 1 && currentFloor < 2){
        addToPrio(3);
}

if(hissQ[2] == 1 && currentFloor > 3){
        addToPrio(4);
}

if(hissQ[2] == 1 && currentFloor < 3){
        addToPrio(5);
}

if(hissQ[3] == 1 && currentFloor > 4){
        addToPrio(6);
}

if(hissQ[3] == 1 && currentFloor < 4){
        addToPrio(7);
}

if(hissQ[4] == 1 && currentFloor < 5){
```

```
                addToPrio(8);
        }
}
```