```
#include "elev.h"
#include "elevator.h"

static enum floor_t floor;
static enum direction_t direction;
static int destinationMatrix[NUMBEROFBUTTONTYPES][NUMBEROFFLOORS]={
                        /*1        2        3        4*/
/*CALL_UP*/{              0,       0,       0,       0},
/*CALL_DOWN*/{            0,       0,       0,       0},
/*COMMAND*/{              0,       0,       0,       0}
};


void checkSensor(){
        if(inFloor()){
                floor=elev_get_floor_sensor_signal();
                handleEvent(FLOOR_REACHED);
        }
}
/*
floorHasOrder()
og
noObstruction()
er guards for FSM
*/
int floorHasOrder(){
        return (destinationMatrix[direction][floor] || destionationMatrix[COMMAND]
}
int elevatorObstructed(){
        return elev_get_obstruction_signal();
}
/*
addOrderToList()
er en del av elevator-klassen
*/
void addOrderToList(enum elev_button_type_t button, enum floor_t floor){
        destinationMatrix[button][floor]=1;
}
/*
handleStop()
handleEmergencyStop()
handleDestination()
kalles av tilstandsmaskinen ved hhv ankomst etasje, n dstopp og avgang etasje
*/
void handleStop(){
        elev_set_door_open_lamp(1);
        clock_t startTime=clock();
        clock_t stopTime=clock();
        while( ((stopTime-startTime)/CLOCKS_PER_SEC) < 3){
                checkButtons();
```

```
                if(elevatorObstructed())
                        startTime=stopTime;
                stopTime=clock();
        }
        elev_set_door_lamp(0);
        handleEvent(NEW_DESTINATION);
}
void handleEmergencyStop(){
        elev_set_stop_lamp(1);
        elev_set_speed(0);
        clearDestinationMatrix();
}
void handleDestination(){
        elev_set_stop_lamp(0);
        if(checkOrderInThisDirection())
                elev_set_speed(300*dir);
        else if(checkOrderInOtherDirection())
                elev_set_speed(300*(-1)*dir);
}
int checkOrderInThisDirection(){
        int keepPreviousDirection=0;/*heisen g r andre vei hvis ikke den f r ord
        if(direction==DOWN)
                keepPreviousDirection=checkLowerFloorsForOrders();
        else
                checkUpperFloorsForOrders();
        return keepPreviousDirection;
}
int checkOrderInOtherDirection(){
        int changeDirection=0;/* eisen skal ikke endre retning dersom den ikke h
        if(direction==DOWN)
                changeDirection=checkUpperFloorsForOrders();
        else
                checkLowerFloorsForOrders();
        return changeDirection;
}
int checkLowerFloorsForOrders(){
        int i,k;
        for(i=0;i<floor;i++){
                for(k=0;k<NUMBEROFBUTTONTYPES;k++){
                        if(destinationMatrix[i][k]==1)
                                return 1;
                }/* end k loop*/
        }/*end i loop*/
        return 0;
}
int checkUpperFloorsForOrders(){
        int i,k;
        for(i=floor+1;i<NUMBEROFFLOORS;i++){
                for(k=0;k<NUMBEROFBUTTONTYPES;k++){
                        if(destinationMatrix[i][k]==1)
```

```
                                    return 1;
                            }/*end k loop*/
                    }/*end i loop*/
                    return 0;
            }
```