1 Implementations

1.1 elevator io

```
#include "elevator_io.h"


void io_resetAllButtonLights(){
        int floor;
        for(floor=0;floor<N_FLOORS;floor++){
                elev_set_button_lamp(BUTTON_COMMAND, floor,0);
                if(floor!=0)
                        elev_set_button_lamp(BUTTON_CALL_DOWN, floor,0);
                if(floor<(N_FLOORS-1))
                        elev_set_button_lamp(BUTTON_CALL_UP, floor,0);
        }
}
void io_resetStopLight(){
        elev_set_stop_lamp(0);
}
void io_resetFloorLightsOnTemporaryStop(floor_t floor, direction_t direction){
        elev_set_button_lamp(BUTTON_COMMAND, floor,0);
        if(direction==UP)
                elev_set_button_lamp(BUTTON_CALL_UP, floor,0);
        else if(direction==DOWN)
                elev_set_button_lamp(BUTTON_CALL_DOWN, floor,0);
}
void io_closeDoor(){
        elev_set_door_open_lamp(0);
}
void io_resetButtonLight(buttonType_t button, floor_t floor){
        elev_set_button_lamp(button,floor,0);
}
void io_setStopLight(){
        elev_set_stop_lamp(1);
}
void io_setButtonLight(buttonType_t button, floor_t floor){
        elev_set_button_lamp(button,floor,1);
}
/*void io_setFloorCallLight(floor_t floor, direction_t direction){
        if(direction==UP)
                elev_set_button_lamp(BUTTON_CALL_UP, floor,1);
        else if(direction==DOWN)
                elev_set_button_lamp(BUTTON_CALL_DOWN, floor,1);
}*/
/*void io_setCommandLight(floor_t floor){
        elev_set_button_lamp(BUTTON_COMMAND, floor,1);
}*/
void io_setFloorIndicator(floor_t floor){
        elev_set_floor_indicator(floor);
}
void io_openDoor(){
        elev_set_door_open_lamp(1);
}

void io_startMotor(direction_t direction){
        elev_set_speed(300*direction);
}
void io_stopMotor(){
        elev_set_speed(0);
}
int io_elevatorIsObstructed(){
        return elev_get_obstruction_signal();
}
int io_elevatorIsAtFloor(){
        return elev_get_floor_sensor_signal();
}
int io_elevatorIsInFloor(){
        return elev_get_floor_sensor_signal()+1;
}
int io_getCurrentFloor(){
        return elev_get_floor_sensor_signal();
}
```

1.2 elevator ui

```
#include "elevator_ctrl.h"
#include "elev.h"
#include "elevator_sm.h"

void ui_checkStop(){
        if(elev_get_stop_signal()){
                sm_handleEvent(STOP_PRESSED);
        }
}

void ui_checkButtons(){
        elev_button_type_t buttonType;
        floor_t floor;

        buttonType = BUTTON_CALL_UP;
        for(floor=0; floor<=2; floor++){
                if(elev_get_button_signal(buttonType, floor)){
                        ctrl_addOrderToList(buttonType,floor);
                }
        }

        buttonType = BUTTON_CALL_DOWN;
        for(floor=1; floor<=3; floor++){
                if(elev_get_button_signal(buttonType, floor)){
                        ctrl_addOrderToList(buttonType,floor);
                }
        }

        buttonType = BUTTON_COMMAND;
        for(floor=0; floor<=3; floor++){
                if(elev_get_button_signal(buttonType, floor)){
                        ctrl_addOrderToList(buttonType,floor);
                }
        }
}
```

1.3 elevator sm

```
#include "elevator_sm.h"
#include "elevator_ctrl.h"
#include <stdio.h>

static state_t state = IDLE;

struct state_action_pair_t{
        state_t nextState;
        int (*guard)();
        void (*action)();
};

/*nextState, guard, action */
struct state_action_pair_t stateTable[NUMBEROFSTATES][NUMBEROFEVENTS] = {
/*                      NEW_DESTINATION                                         FL
/*EXECUTING_ORDER*/     {{EXECUTING_ORDER, NULL, NULL},                         {T
/*TEMPORARY_STOP*/      {{EXECUTING_ORDER, NULL, ctrl_handleDestination},       {T
/*IDLE*/                {{EXECUTING_ORDER, NULL, ctrl_handleDestination},       {I
/*EMERGENCY_STOP*/      {{EXECUTING_ORDER, NULL, ctrl_handleDestination},       {E
};

void sm_handleEvent(event_t event){
        struct state_action_pair_t transition = stateTable[state][event];
        if(transition.guard == NULL || transition.guard()){
                if(transition.action != NULL){
                        transition.action();
                }
                state = transition.nextState;
        }
}
```

1.4 elevator ctrl

```
#include "elevator_io.h"
#include "elevator_ctrl.h"
#include "elevator_sm.h"
```

```
#include "elevator_ui.h"
#include <stdio.h>

static floor_t floor;
static 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 ctrl_initiateElevator(){
        if(io_elevatorIsInFloor()){
                floor=io_getCurrentFloor();
                return;
        }
        else{
                direction=UP;
                io_startMotor(direction);
                while(!io_elevatorIsInFloor()){

                }
                io_stopMotor();
                floor=io_getCurrentFloor();
        }
}
void ctrl_checkSensor(){
        if(io_elevatorIsInFloor()){
                floor=io_getCurrentFloor();
                printf("Etasje: %d\n",floor);
                sm_handleEvent(FLOOR_REACHED);
        }
}
/*
floorHasOrder()
og
noObstruction()
er guards for FSM
*/
int ctrl_floorHasOrder(){
        return (destinationMatrix[direction][floor] || destinationMatrix[COMMAND][
}
int ctrl_elevatorObstructed(){
        return io_elevatorIsObstructed();
}
/*
addOrderToList()
er en del av elevator-klassen
*/
```

```
void ctrl_addOrderToList(elev_button_type_t button, floor_t floor){
        destinationMatrix[button][floor]=1;
        io_setButtonLight(button, floor);
        sm_handleEvent(NEW_DESTINATION);
}
/*
handleStop()
handleEmergencyStop()
handleDestination()
kalles av tilstandsmaskinen ved hhv ankomst etasje, n dstopp og avgang etasje
*/
void ctrl_handleStop(){
        ctrl_setLightsAtElevatorStop();
        clock_t startTime=clock();
        clock_t stopTime=clock();
        while( ((stopTime-startTime)/CLOCKS_PER_SEC) < 3){
                ui_checkButtons();
                if(ctrl_elevatorObstructed())
                        startTime=stopTime;
                stopTime=clock();
        }
        io_closeDoor();
        sm_handleEvent(NEW_DESTINATION);
}
void ctrl_handleEmergencyStop(){
        io_setStopLight();
        io_stopMotor();
        io_resetAllButtonLights();
        ctrl_clearDestinationMatrix();
}
void ctrl_handleDestination(){
        io_resetStopLight();
        printf("Retning: %d\n",direction);
        if(ctrl_checkOrderInThisDirection()){
                printf("Motor burde startes\n");
                io_startMotor(direction);
        }else if(ctrl_checkOrderInOtherDirection()){
                io_startMotor((-1)*direction);
                direction = (-1)*direction;
        }
}
int ctrl_checkOrderInThisDirection(){
        int keepPreviousDirection=0;/*heisen g r andre vei hvis ikke den f r or
        if(direction==DOWN)
                keepPreviousDirection=ctrl_checkLowerFloorsForOrders();
        else
                keepPreviousDirection=ctrl_checkUpperFloorsForOrders();

        return keepPreviousDirection;
}
```

```
int ctrl_checkOrderInOtherDirection(){
        int changeDirection=0;/* eisen skal ikke endre retning dersom den ikke h
        if(direction==DOWN)
                changeDirection=ctrl_checkUpperFloorsForOrders();
        else
                changeDirection=ctrl_checkLowerFloorsForOrders();
        return changeDirection;
}
int ctrl_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 ctrl_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;
}
void ctrl_clearDestinationMatrix(){
        int i,k;
        for(i=0;i<NUMBEROFBUTTONTYPES;i++){
                for(k=0;k<NUMBEROFFLOORS;k++){
                        destinationMatrix[i][k]=0;
                }
        }
}
void ctrl_setLightsAtElevatorStop(){
        io_openDoor();
        io_resetButtonLight(BUTTON_COMMAND, floor);
        if(floor!=FIRST&&direction==DOWN)
                io_resetButtonLight(BUTTON_CALL_DOWN, floor);
        if(floor!=FOURTH&&direction==UP)
                io_resetButtonLight(BUTTON_CALL_UP, floor);
}
```