CONTROL SYSTEMS TECHNOLOGIES
Project theme n° 1

Use the CoDeSys environment to implement the logic control of an elevator. Download the file *SingleElevator.pro* where a graphic interface and a simulator of the system (not controlled) have been already implemented.
The logic control must allow:

- the initialization of the elevator.
- the correct functioning of the elevator with an efficient management of the users calls (from both the inside and the outside of the elevator cab). Also simultaneous calls must be considered.

Note that it is not required an optimal policy for the management of the calls (e.g. the minimization of the waiting times), but it is important that the chosen policy has a minimum degree of efficiency.
The chosen policy must be accurately described in the project report.

Refer to the "Elevator description" section for a more accurate description of the system functioning requirements.

Guideline

The logic control must be developed using **SFC** language and the "**Generalized Actuator**" approach proposed during the course.
Use **Function Blocks** to organize the control program in a readable way.
It is possible to define new variables and functions for the development of the sequential control.
ST language is suggested to define SFC action/transaction, however, also the use of LD, IL and FBD is allowed.

For the presentation of the project the student must have:

- the *.pro* file compatible with *CoDeSys v.3.5* (the student can use his own laptop or can bring a Flash-Key);
- A report (3-4 pages) with the main elements of the project (solution architecture, adopted policies, choices made for meeting requirements …).

Elevator description

Refer to your personal experience for the description of the system and for the functioning requirements (elevators are commonly used systems).
Below are just some relevant notes that differs from the common experience.
Please refer to the "I/O and visualization variables description" section to having knowledge of the cited variables.

When the system becomes operative, the elevator cab is in proximity of an unknown deck. Only the first time an initialization phase must be executed:

1. Closing of the doors (checking the presence sensor);
2. Descent of the cab to the lower limit sensor with low speed;
3. Ascent of the cab to the first floor with low speed;
4. Opening of the doors;
5. Waiting for calls.

Assume the system knows a priori the number of levels.

After the initialization there is the nominal functioning. Below are the sequence of operations to serve the destination deck starting from the current one:

1. Waiting for 2sec, and closing the doors (checking the presence of people between the doors);
2. Moving the cab at low speed to the first ramp (end of ramp) signal;
3. Moving the cab with high speed until the cab reaches the ramp signal immediately before the destination deck;
4. Slowing down of the speed, and when the cab reaches the deck, stopping of the motor;
5. Waiting for 2sec and opening of the door;
6. Waiting for 5sec.

The logic control has to manage the Emergency button. If the Emergency button is pushed the control must stop the cab at the first possible floor in order to allow people to come out. The system remains stationary until the emergency button is not pressed a second time. The users calls manager has to keep working during the emergency phase.

Finally the control has to manage the LED graphical interface (when a button is pushed the corresponding LED has to turn on).

Diagnostic:
The logic control must detect and manage the deck and ramp sensors fault (we consider only these two faults). Buttons associated to variables are provided to simulate sensors faults (i.e. "Ramp#Up", "Ramp#Down", "Sensor#"). These variables cannot be accessed by the control.
For clarifications about signals meaning or the "non controlled" functioning of the system it is possible to see the code used for the simulation of the system in *SingleElevator.pro* file.

I/O and Human Machine Interface (HMI) variables description

Input variables (Sensors):

```
Open: BOOL;          (* Sensor: the door is fully open*)
Closed: BOOL;        (* Sensor: the door is fully closed*)
LimitDown: BOOL;     (* Sensor: lower limit sensor, if TRUE the elevator has reached the lower limit*)
LimitUp: BOOL;       (* Sensor: upper limit sensor, if TRUE the elevator has reached the upper limit*)
DeckSensor: BOOL;    (* Sensor: if TRUE the cab is in proximity of a deck*)
RampSensor: BOOL;    (* Sensor: if TRUE the cab is in proximity of a ramp*)
Presence :BOOL;      (* Sensor: if TRUE something is blocking the door photocell*)
```

Output Variables (Actuators):

```
Closing: BOOL;       (* Actuator: if TRUE the door starts closing*)
Opening: BOOL;       (* Actuator: the door starts opening*)
Motor_on: BOOL;      (* Actuator: if TRUE the elevator pulley motor start moving*)
Up: BOOL;            (* Actuator parameter: if TRUE the elevator moves upward, downward otherwise*)
Vel: BOOL;           (* Actuator parameter: motor speed, if TRUE high speed is selected, low speed
                      otherwise*)
```

HMI Input Variables

```
Emergency: BOOL;     (* Input: if TRUE the emergency button has been pressed*)

PG: BOOL;            (*internal panel variables, if TRUE a call for the corresponding floor has been
                     made*)
P1: BOOL;
P2: BOOL;
P3: BOOL;
P4: BOOL;
P5: BOOL;
P6: BOOL;
P7: BOOL;
UpDeckG: BOOL;       (*external panel variables*)
UpDeck1: BOOL;
UpDeck2: BOOL;
UpDeck3: BOOL;
UpDeck4: BOOL;
UpDeck5: BOOL;
UpDeck6: BOOL;
```

DownDeck7: BOOL;
DownDeck6: BOOL;
DownDeck5: BOOL;
DownDeck4: BOOL;
DownDeck3: BOOL;
DownDeck2: BOOL;
DownDeck1: BOOL;

HMI Output Variables

Current: INT; (*variable used to store the current deck*)
Destination: INT; (*variable used to store the destination deck*)
EmergencyVis: BOOL; (*if TRUE the emergency light is turned on*)

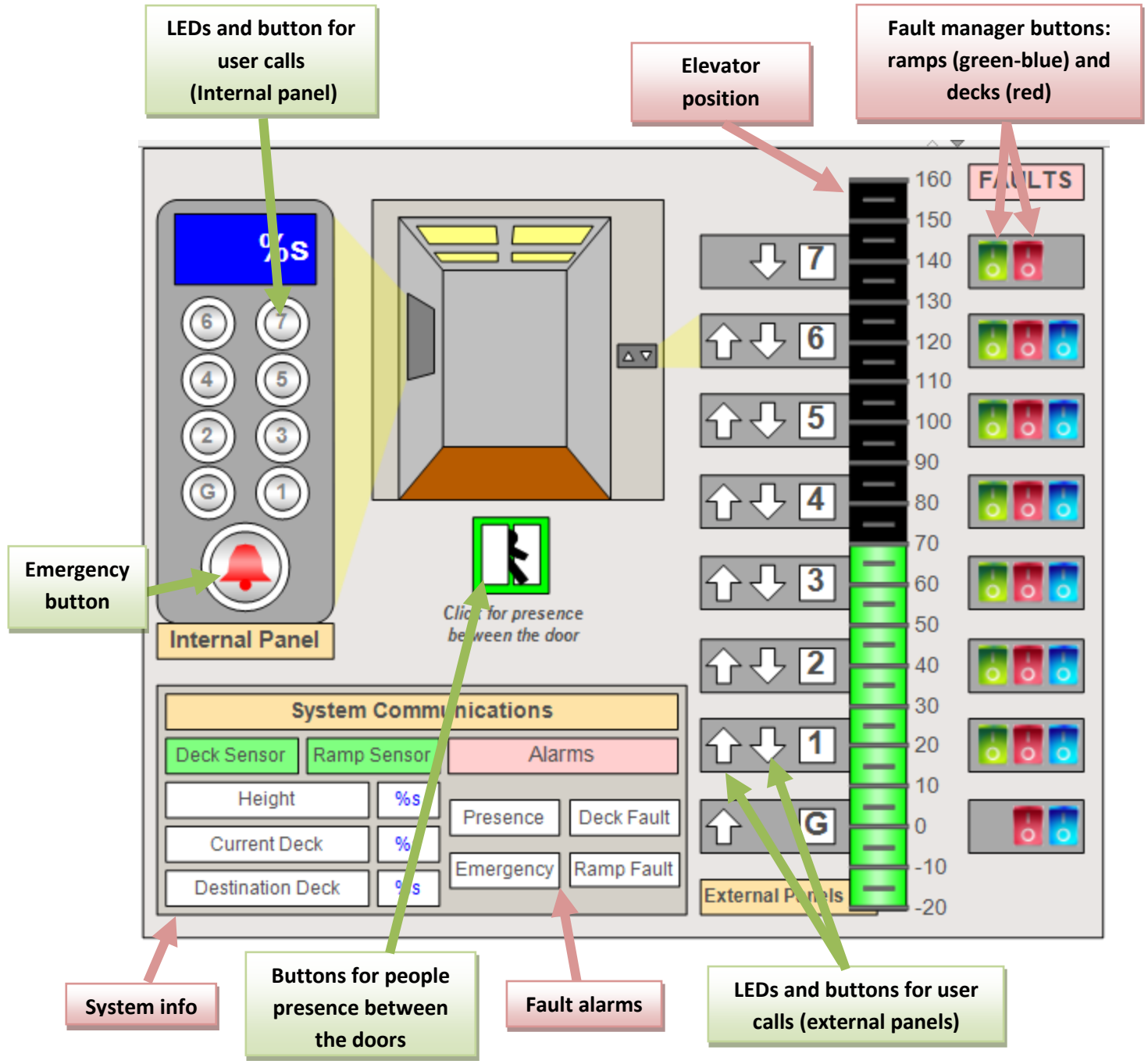
PGVis: BOOL; (*internal panel variables used to turn on the LEDs signaling the calls*)
P1Vis: BOOL;
P2Vis: BOOL;
P3Vis: BOOL;
P4Vis: BOOL;
P5Vis: BOOL;
P6Vis: BOOL;
P7Vis: BOOL;
UpDeckGVis: BOOL; (*external panel variables used to turn on the LEDs signaling the calls *)
UpDeck1Vis: BOOL;
UpDeck2Vis: BOOL;
UpDeck3Vis: BOOL;
UpDeck4Vis: BOOL;
UpDeck5Vis: BOOL;
UpDeck6Vis: BOOL;
DownDeck7Vis: BOOL;
DownDeck6Vis: BOOL;
DownDeck5Vis: BOOL;
DownDeck4Vis: BOOL;
DownDeck3Vis: BOOL;
DownDeck2Vis: BOOL;
DownDeck1Vis: BOOL;
FaultSensorAlarm: BOOL; (*if TRUE alerts the user of a deck sensor fault*)
FaultRampAlarm: BOOL; (*if TRUE alerts the user of a ramp sensor fault*)

Internal Variables (Only for simulation)

```
InitialDeck: INT:=1;          (* Only for simulation: variable used to define the initial deck*)
Init: BOOL := FALSE;          (* Only for simulation: used to execute the instruction at the first cycle*)
Height: INT;                  (* Only for simulation: elevator height*)
DeckRest: INT;                (* Only for simulation: division rest for generating deck sensor*)
Memory: INT;
RampRest: INT;                (* Only for simulation: division rest for generating the ramp position*)
MemoryR: INT;
Door: INT;                    (* Only for simulation: to visualize the closing of the door*)
Displacement: INT;            (* Only for simulation: elevator displacement, dependent on the motor speed*)

(*Only for simulation: variables used to simulate the fault on deck sensors*)
SensorG: BOOL;      (* if TRUE the magnetic info for deck G is lost and the deck sensor cannot become TRUE*)
Sensor1: BOOL;      (* if TRUE the magnetic info for deck 1 is lost and the deck sensor cannot become TRUE*)
Sensor2: BOOL;
Sensor3: BOOL;
Sensor4: BOOL;
Sensor5: BOOL;
Sensor6: BOOL;
Sensor7: BOOL;
RampGUp: BOOL;                (*Only for simulation: variables used to simulate the fault on ramp sensors*)
Ramp1Down: BOOL;
Ramp1Up:BOOL;
Ramp2Down: BOOL;
Ramp2Up:BOOL;
Ramp3Down: BOOL;
Ramp3Up:BOOL;
Ramp4Down: BOOL;
Ramp4Up:BOOL;
Ramp5Down: BOOL;
Ramp5Up:BOOL;
Ramp6Down: BOOL;
Ramp6Up:BOOL;
Ramp7Down: BOOL;
SensorGOK: BOOL;
Sensor1OK: BOOL;
Sensor2OK: BOOL;
Sensor6OK: BOOL;
Sensor3OK: BOOL;
Sensor4OK: BOOL;
Sensor5OK: BOOL;
Sensor7OK: BOOL;
```

RampGUpOK: BOOL;
Ramp1DownOK: BOOL;
Ramp1UpOK: BOOL;
Ramp2DownOK: BOOL;
Ramp2UpOK: BOOL;
Ramp3DownOK: BOOL;
Ramp3UpOK: BOOL;
Ramp4DownOK: BOOL;
Ramp4UpOK: BOOL;
Ramp5DownOK: BOOL;
Ramp5UpOK: BOOL;
Ramp6DownOK: BOOL;
Ramp6UpOK: BOOL;
Ramp7DownOK: BOOL;
Ramp7UpOK: BOOL;


LEDs and button for user calls (Internal panel)
Elevator position
Fault manager buttons: ramps (green-blue) and decks (red)
%s
6
7
4
5
2
3
G
1
Internal Panel
Emergency button
Click for presence between the door
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
-10
-20
7
6
5
4
3
2
1
G
FAULTS
System Communications
Deck Sensor
Ramp Sensor
Alarms
Height
%s
Current Deck
%
Destination Deck
%s
Presence
Deck Fault
Emergency
Ramp Fault
External Panels
System info
Buttons for people presence between the doors
Fault alarms
LEDs and buttons for user calls (external panels)