CLEARSY
Safety Solutions Designer

AIX
LYON
PARIS
STRASBOURG

WWW.CLEARSY.COM

Hackathon
JUL2025

Airlock
Functional Controller

PART III

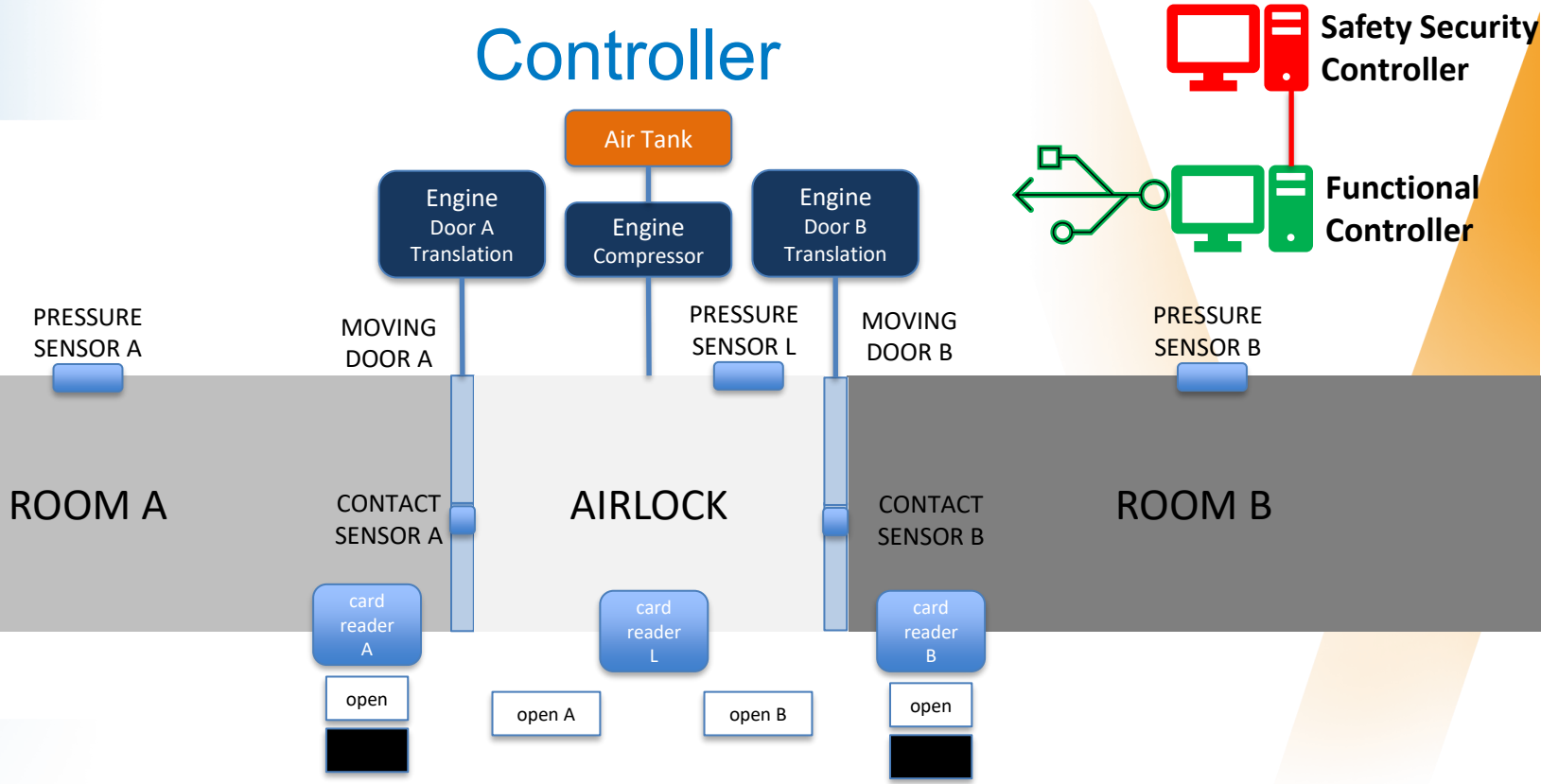
Thierry Lecomte
R&D Director

THIERRY.LECOMTE@CLEARSY.COM

Door Opening Controller

- *Open and close doors on demand*
  - *Authentication with card reader*
  - *Activation of buttons attached to an authenticated reader*
  - *Open and close doors, pressure and depressure airlock*
- *Works all the time*
- *Works independently from the safety system*

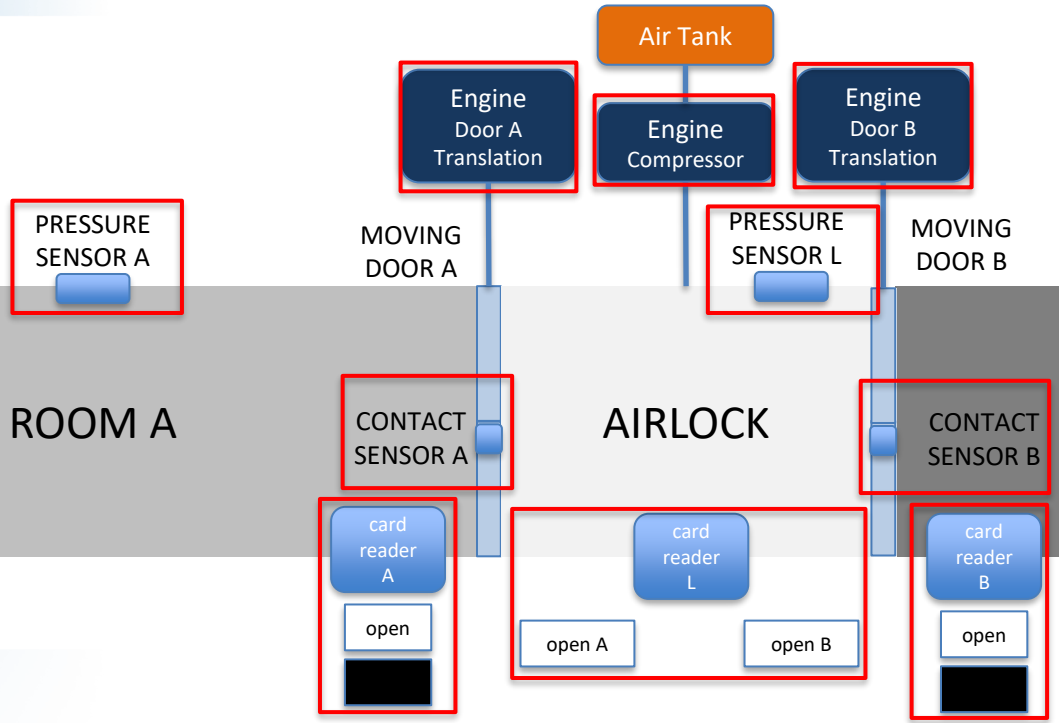
Controller

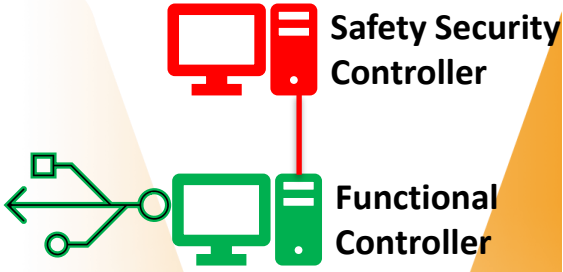


design decision: we remove the **close** buttons to save money

The

Controller





Hypotheses

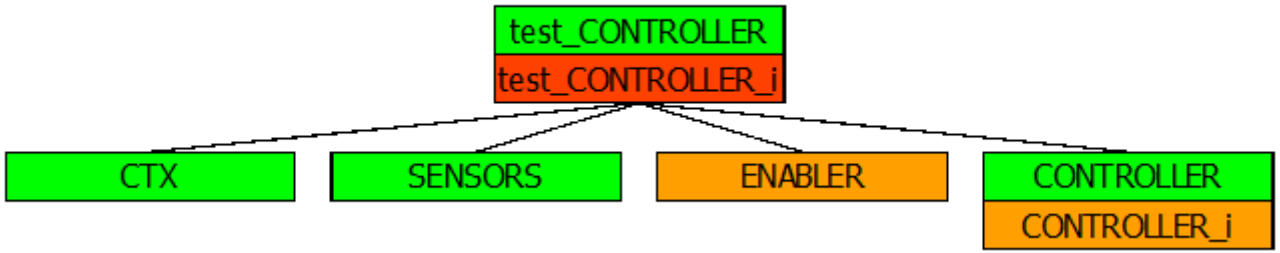
- *We cannot detect people and interpret their will.*
- *We react to interactions with card reader and open/close buttons*
- *When nothing happen, the system automatically close the open door*
- *Buttons are considered only when authentication completed (you cannot authenticate and activate buttons during the same cycle)*
- *We treat race conditions (several persons trying to authenticate at the “same time”)*
  - *No memory, first person to authenticate wins. If several authentications in the same cycle, there is a predefined order, base on safety (maybe A if A is space/vaccum)*
  - *A card reader is authenticated as long as the open/close action (if any) is not yet completed*
  - *There is some (reinforced) window to check if someone is in the airlock and some lights on wall to indicate if the airlock is “busy” or not (like toilets in a plane)*
- *We do not manage access restrictions (one person could open door A but not door B)*

Hypotheses

- *We do not consider manual mechanisms procedure to operate the airlock in case something goes wrong*
- *We do not consider attacks to the system (like someone drilling a hole in any of the doors)*
- *We do not consider timing and delays*
  - *Actions are initiated and are supposed to terminate: always possible top move doors, increase and decrease pressure – no problem of capacity*
  - *There is no delay considered after authentication – if you want to stay forever in the airlock, it is your choice*
  - *We do not specify or implement HW drivers to finely control doors and pressuring systems*

Your turn

- *Develop a B model of this control function [8 pt]*
  - *many sensors to read*
  - *Definition of objective (based on card reader activated and button pushed)*
  - *Definition of actions (based on objective and current sensors, this is the next action to perform in this order: close door, pressure, open door)*
  - *safety properties – they are going to be linked in a combining component where the enabler will be linked with the controller*



```
MACHINE
    CTX
SETS
    PRESSURES = {
    };
    OBJECTIVES = {
        OBJ_OPEN_DOOR_A,
        OBJ_OPEN_DOOR_B,
        OBJ_NONE
        };
    ACTIONS = {
        NONE,
        TRANSLATE_OPEN_DOOR_A,
        TRANSLATE_OPEN_DOOR_B,
        TRANSLATE_CLOSE_DOOR_A,
        TRANSLATE_CLOSE_DOOR_B,
        ADAPT_PRESSURE_L_TO_A,
        ADAPT_PRESSURE_L_TO_B
    };
    AUTHENTICATED = {
        AUTHENTICATED_A, // successful authentication on card reader A
        AUTHENTICATED_L, // successful authentication on card reader L
        AUTHENTICATED_B, // successful authentication on card reader B
        AUTHENTICATED_NONE  // no active authentication
    }
```

```
MACHINE
    SENSORS
SEES
    CTX
CONCRETE_VARIABLES
    /* P2 */
    pressure_sensor_l,
    contact_sensor_a,
    contact_sensor_b,

    /* P3 */
    card_reader_a,
    card_reader_l,
    card_reader_b,

    button_room_a_open_
    button_room_l_open_a,
    button_room_l_open_b,
    button_room_b_open_b
```

```
INVARIANT
    pressure_sensor_l : PRESSURES &
    contact_sensor_a : BOOL & // TRUE means door closed
    contact_sensor_b : BOOL & // TRUE means door closed

    card_reader_a : BOOL & // TRUE means card reader activated this cycle
    card_reader_l : BOOL & // TRUE means card reader activated this cycle
    card_reader_b : BOOL & // TRUE means bcard reader activated this cycle

    button_room_a_open_a : BOOL & // TRUE means button pushed this cycle
    button_room_l_open_a : BOOL & // TRUE means button pushed this cycle
    button_room_l_open_b : BOOL & // TRUE means button pushed this cycle
    button_room_b_open_b : BOOL  // TRUE means button pushed this cycle
```

```
update_sensors_states =
BEGIN
    pressure_sensor_l,
    contact_sensor_a,
    contact_sensor_b: (
        pressure_sensor_l : PRESSURES &
        contact_sensor_a : BOOL &
        contact_sensor_b : BOOL &
        (not(pressure_sensor_l = PRESSURE_A) => contact_sensor_a = TRUE) &
        (not(pressure_sensor_l = PRESSURE_B) => contact_sensor_b = TRUE)
    ) ||
    card_reader_a :: BOOL ||
    card_reader_l :: BOOL ||
    card_reader_b :: BOOL ||

    button_room_a_open_a :: BOOL ||
    button_room_l_open_a :: BOOL ||
    button_room_l_open_b :: BOOL ||
    button_room_b_open_b :: BOOL
END
```

```
MACHINE
    CONTROLLER
SEES
    CTX, SENSORS, ENABLER
CONCRETE_VARIABLES
    current_action, /* action to be performed by the system */
    current_authentication,
    current_objective /* user's objective based on button pushed and card reader */
INVARIANT
    current_action : ACTIONS &
    current_authentication: AUTHENTICATED &
    current_objective : OBJECTIVES
INITIALISATION
    current_action := NONE ||
    current_authentication := AUTHENTICATED_NONE ||
    current_objective := OBJ_NONE
```

```
process_readers =
BEGIN
    current_authentication: (
        current_authentication: AUTHENTICATED &
        (not(current_authentication = AUTHENTICATED_NONE) =>
            current_authentication$0 :{AUTHENTICATED_NONE, current_authentication})
    )
END;

control =
BEGIN
    current_action,
    current_objective :(
        current_action: ACTIONS &
        current_objective : OBJECTIVES &

        (current_action = TRANSLATE_OPEN_DOOR_A =>
            pressure_sensor_1 = PRESSURE_A & contact_sensor_b = TRUE
        )
        /* to complete */
    )
END
```

Your turn

- Complete the postcondition of the function control
  - Keep focused on the main aspects
- We are going to implement the logic tomorrow … be prepared for it !

Your turn

```
MACHINE
    test_CONTROLLER
OPERATIONS
    test_control = skip
END
```

- Execute 10 steps with ProB [2 pts]
  - Save the probtrace file

```
IMPLEMENTATION test_CONTROLLER_i
REFINES test_CONTROLLER

IMPORTS CTX, SENSORS, ENABLER, CONTROLLER
OPERATIONS
    test_control =
    BEGIN
        update_sensors_states;
        process_readers;
        control;
        compute_enabling
    END
END
```

Your turn

- ***Optional**: do you see a more efficient way to manage the sequence of actions to perform, to complete a scenario? [3 pt]*