CLEARSY
Safety Solutions Designer

AIX
LYON
PARIS
STRASBOURG

WWW.CLEARSY.COM

Hackathon
JUL2025

# Airlock
# Safety Controller

Thierry Lecomte
R&D Director

THIERRY.LECOMTE@CLEARSY.COM

PART II

Attribution 4.0 Unported (CC BY 4.0)

# Door Opening Enabler (safety)

*Modelling embedded system (SW + HW)*

- ▶ *Supervisor **independent** from functional controller*
- ▶ *System to check if a door **can** be open, based on sensors*
- ▶ *Works all the time*
- ▶ *If disagree or OFF, disables door opening*

Attribution 4.0 Unported (CC BY 4.0)

# Safety Property

- *Instead of giving a deterministic specification involving all possible cases (usually is the transcription of an algorithm)*
- *Specify what is really important – when a safety issue may arise*
- *We distinguish*
  - ***Restrictive** position: doors are closed → nothing bad could happen*
  - ***Permissive** position: doors are opening → we could have a pressure problem*
- *Formalizing safety property: **when we are moving to a permissive situation, we need to be sure that all conditions are OK***

Attribution 4.0 Unported (CC BY 4.0)

## Safety Property

- *For example*

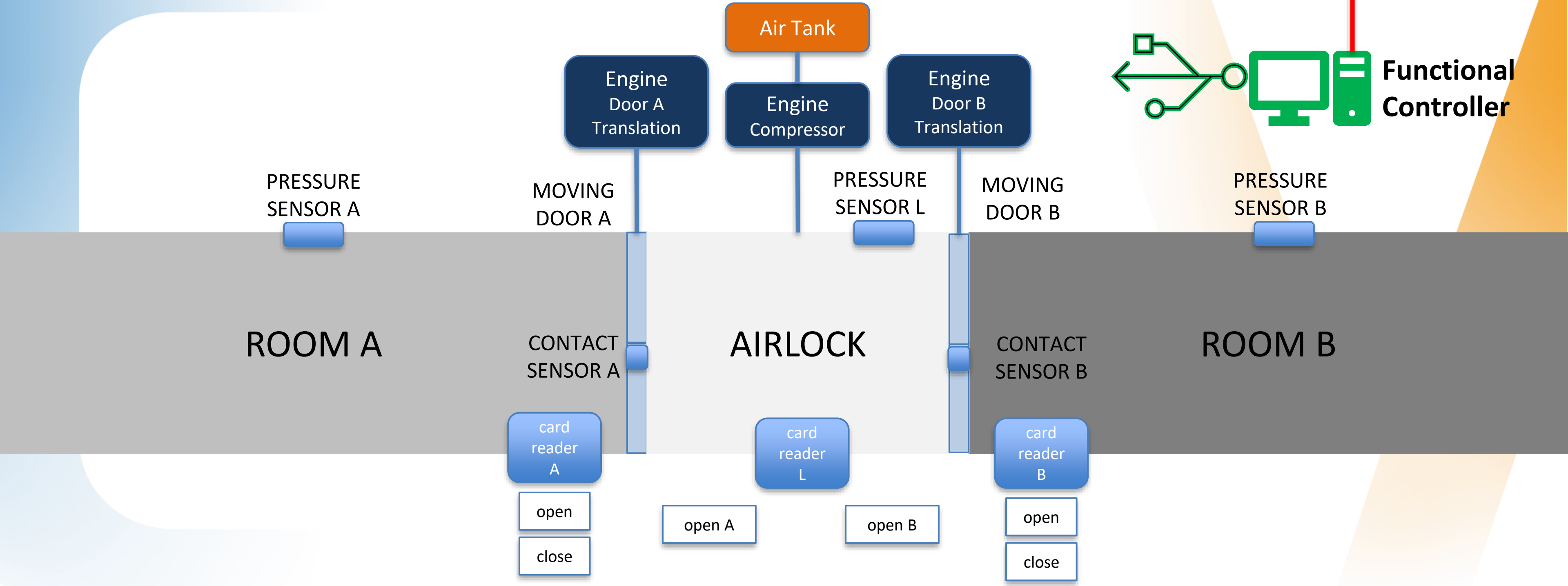
```
(door_opening=TRUE => pressure_check=TRUE &
HW_conditions=OK)
```

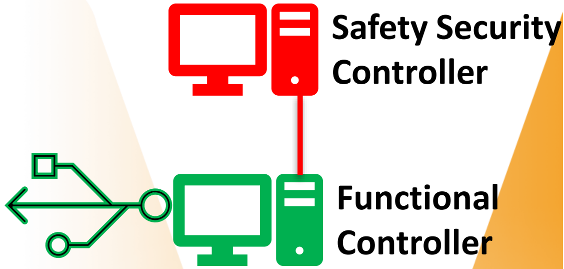
- *And a valid (but not very useful) controller implementation could be*

```
door_opening := FALSE
```

Attribution 4.0 Unported (CC BY 4.0)

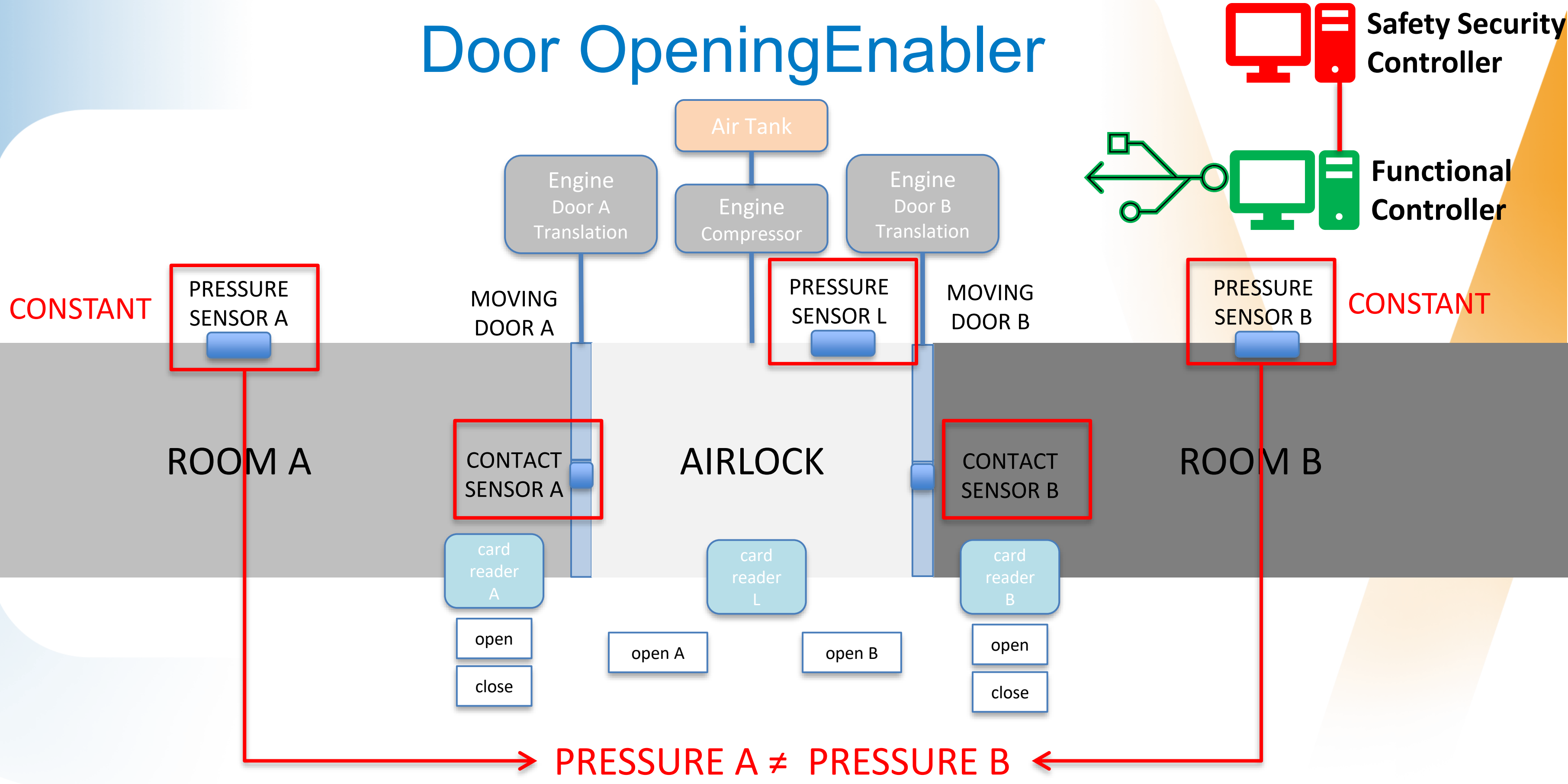
# Door OpeningEnabler





Attribution 4.0 Unported (CC BY 4.0)

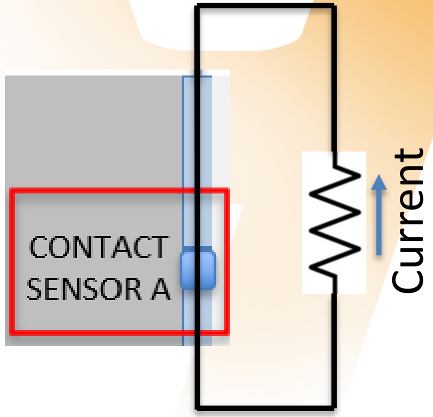
# Door OpeningEnabler



Attribution 4.0 Unported (CC BY 4.0)

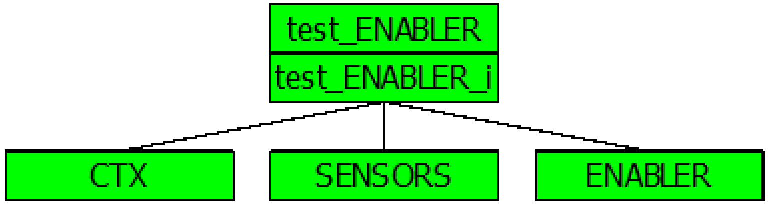
# Hypotheses

- ***PRESSURE_A*** *sufficiently different from* ***PRESSURE_B*** *to injure people in case of "illegal" opening*
- ***pressure_a*** *and* ***pressure_b*** *always return the same value and are ignored (constants)*
- ***contact_a*** *and* ***contact_b*** *are "almost perfect" and do not generate "illegal" "closed" signal*
- *We are not sensing people, where they are in the airlock, if they are wearing equipment (suit) compatible with pressure in the other room (could be done with RFID tag on the suit and proper interface)*



# Your turn

- *Develop a B model of this enabling function [8 pts]*
  - *3 sensors to read (pressure_l, contact_a, contact_b*
  - *2 enabling variables (enable_door_a, enable_door_b)*
  - *safety properties*



Attribution 4.0 Unported (CC BY 4.0)

# Your turn

- *CTX model*

We do not model the exact value of the pressure

- **PRESSURE_A** means sufficiently close to the pressure in room A to avoid injury
- **PRESSURE_OTHER** means sufficiently different from PRESSURE_A and B to ensure injuries

```
MACHINE
    CTX
SETS
    PRESSURES = {
        PRESSURE_A,
        PRESSURE_B,
        PRESSURE_OTHER
    }

END
```

Attribution 4.0 Unported (CC BY 4.0)

# Your turn

- ▶ *SENSORS model*

```
MACHINE
    SENSORS
SEES
    CTX
CONCRETE_VARIABLES
    pressure_sensor_l,
    contact_sensor_a,
    contact_sensor_b
INVARIANT
    pressure_sensor_l : PRESSURES &
    contact_sensor_a : BOOL & // TRUE means door closed
    contact_sensor_b : BOOL   // TRUE means door closed
INITIALISATION
OPERATIONS
    update_sensors_states =
    BEGIN
        pressure_sensor_l,
        contact_sensor_a,
        contact_sensor_b: (
        )
    END
END
```

Any values in their domain
but a door open has an impact on
the pressure in the airlock

Attribution 4.0 Unported (CC BY 4.0)

# Your turn

- *ENABLER model*

The safety property of the airlock (what do we want to ensure all the time)

The safety property of the control of each enabling variable

```
MACHINE
    ENABLER
SEES
    CTX, SENSORS
CONCRETE_VARIABLES
    enable_door_a,
    enable_door_b
INVARIANT
    enable_door_a : BOOL &
    enable_door_b : BOOL &

INITIALISATION
    enable_door_a := FALSE ||
    enable_door_b := FALSE
OPERATIONS
    compute_enabling =
    PRE
    THEN
        enable_door_a,
        enable_door_b :(
        )
    END
END
```

Attribution 4.0 Unported (CC BY 4.0)

# Your turn

- *Check the enabling with ProB [2 pts]*
  - *run the operation 10 times*
  - Save the probtrace file with the value of oks

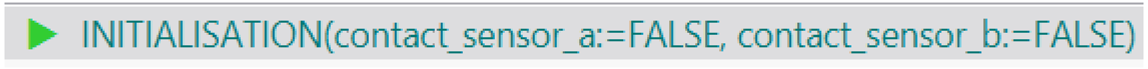
```
MACHINE
    test_ENABLER
OPERATIONS
    test_compute_enabling = skip
END
```

```
IMPLEMENTATION test_ENABLER_i
REFINES test_ENABLER
IMPORTS CTX, SENSORS, ENABLER
OPERATIONS
    test_compute_enabling =
    BEGIN
        update_sensors_states;
        compute_enabling
    END
 END
```

Attribution 4.0 Unported (CC BY 4.0)

# Your turn

- ***Optional**: what happens if we chose the INITIALISATION with both doors open ? [1 pt]*



```
▶ INITIALISATION(contact_sensor_a:=FALSE, contact_sensor_b:=FALSE)
```

- ***Optional**: in the operation test_compute_enabling,what happens if we forgot to update the sensors (we forget to call update_sensors_states) ? [1 pt]*
- ***Optional**: how would you simply ensure a correct (i.e. proved) sequencing of the two operations update_sensors_states and* compute_enabling *? [3 pt]*