Safety Solutions Designer

AIX
LYON
PARIS
STRASBOURG

WWW.CLEARSY.COM

TUTORIAL

ABZ 2021

JUNE 2021

Using B to program the CLEARSY Safety Platform

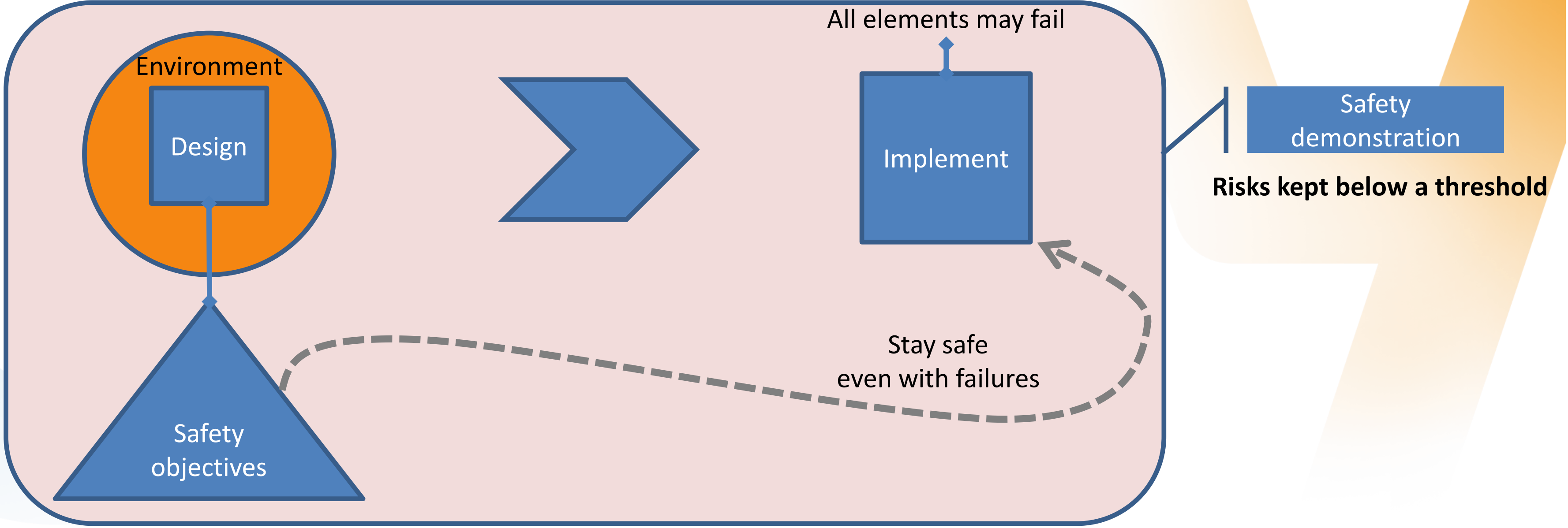
THIERRY.LECOMTE@CLEARSY.COM

PART I

INTRODUCTION

Safety, Standards & Embedded Systems

- Safety Critical Systems: systems where life is at risk
  - Errors and failures may lead to injury or death



Safety, Standards & Embedded Systems

- Trains, planes, cars, nuclear plants, etc.

- Domain specific standards and safety
  - Driverless trains able to stop [EN50126,128,129]
  - Planes can’t stop flying – availability first, human pilot to handle complex situations [DO-178]
  - Cars rely on human driver – certification not mandatory [ISO26262]
  - Nuclear plants increase # safety barriers for highest levels [CEI61513]

Safety, Standards & Embedded Systems

- Recommendations
  - Collection of industrial best practices
  - No definitive recipe to produce safe systems
  - Several standards (strongly) recommend the use of formal methods for the highest safety levels

IEC 61508: Software design and dev. (table A.2)

Technique/Measure	Ref	SIL1	SIL2	SIL3	SIL4
1 Fault detection and diagnosis	C.3.1	---	R	HR	HR
2 Error detecting and correcting codes	C.3.2	R	R	R	HR
3a Failure assertion programming	C.3.3	R	R	R	HR
3b Safety bag techniques	C.3.4	---	R	R	R
3c Diverse programming	C.3.5	R	R	R	HR
3d Recovery block	C.3.6	R	R	R	R
3e Backward recovery	C.3.7	R	R	R	R
3f Forward recovery	C.3.8	R	R	R	R
3g Re-try fault recovery mechanisms	C.3.9	R	R	R	HR
3h Memorising executed cases	C.3.10	---	R	R	HR
4 Graceful degradation	C.3.11	R	R	HR	HR
5 Artificial intelligence - fault correction	C.3.12	---	NR	NR	NR
6 Dynamic reconfiguration	C.3.13	---	NR	NR	NR
7a Structured methods including for example, JSD, MASCOT, SADT and Yourdon	C.2.1	HR	HR	HR	HR
7b Semi-formal methods	Table B.7	R	R	HR	HR
7c Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	---	R	R	HR
8 Computer-aided specification tools	B.2.4	R	R	HR	HR

a) Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

b) The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in part 2 of this standard.

Safety, Standards & Embedded Systems

- Safety Integrity Level
  - Level 3: 1 failure every century (10^{-7}/h)
  - Level 4: 1 failure every 10 millenia (10^{-9}/h)
- Covers hardware, software, environment
  - Specification error
  - Design error, programming error, bad compilation
  - Wrong execution, failing hardware
- Architectures for highest safety level
  - 2 processors in parallel (or more),
  - 2 independent SW dev teams, independent testing team
  - Protecting mechanisms in case of perturbation

Ex:
memory corruption,
short circuit,
drifting clock,
degrading micro-circuit

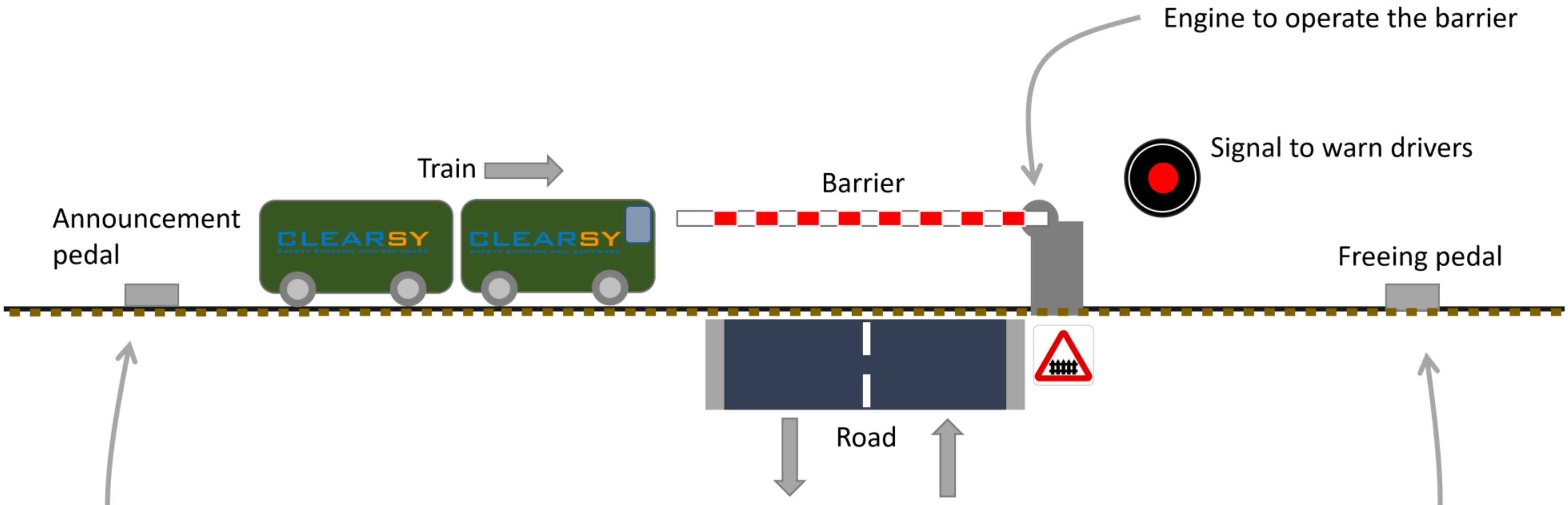
The Railway Level Crossing Example

- Safety system to prevent human being from entering when a train is approaching (main cause of accident*)
- Not intended to stop the train (road side protection and warning)
- Several different instances:
  - No barrier, single barrier, double (delayed) barriers
  - Open is safe / closed is safe

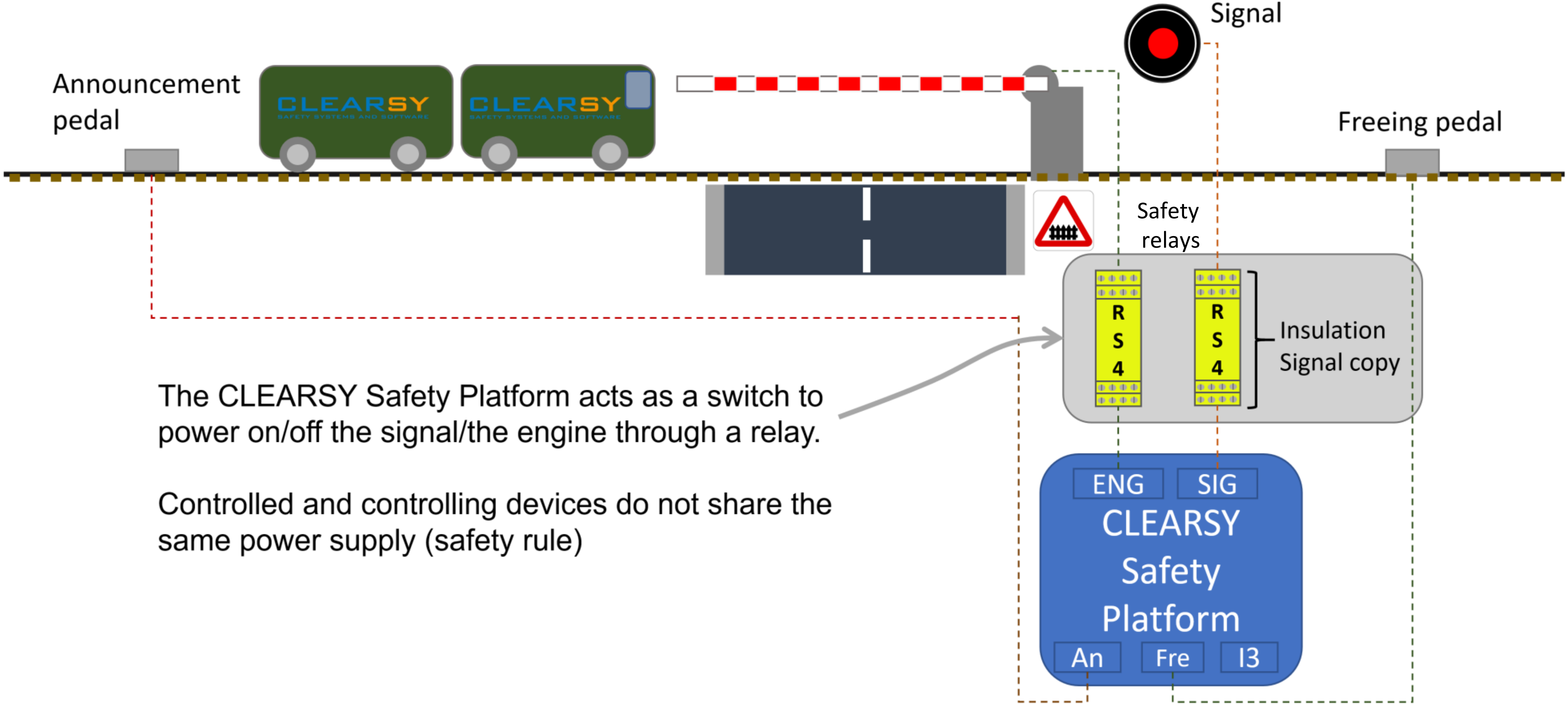
*: *Bayesian Network Modeling Applied on Railway Level Crossing Safety*
C. Liang, M. Ghazel, O. Cazier, L. Bouillaut and E. El-Koursi. RSSRail 2017 Pistoia

The Railway Crossing Example

Video « Level Crossing Accident »

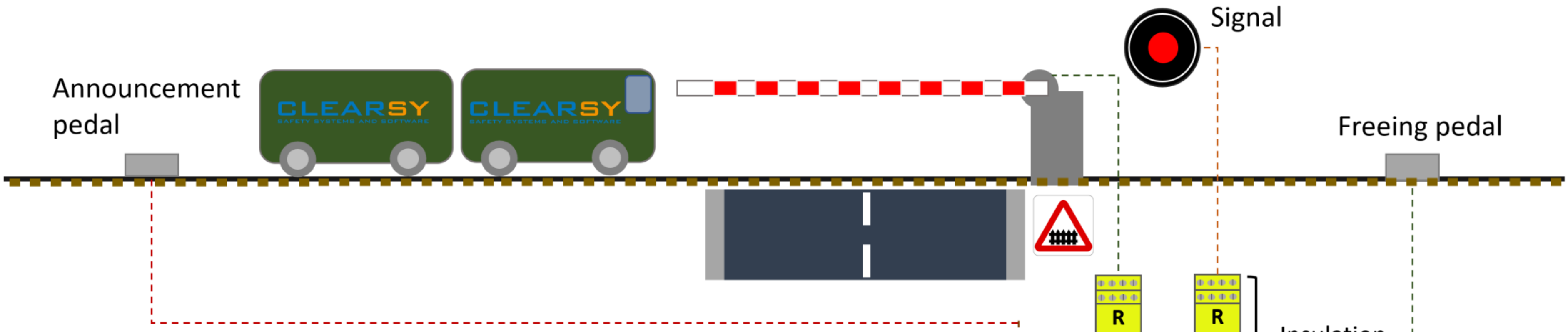


A pedal is a device that allows the detection of the passage of any rolling stock on a given track point.
The detection is either **magnetic** or **mechanical**.



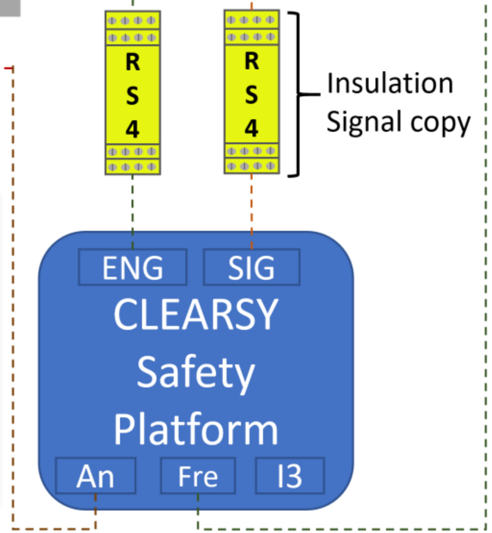
The CLEARSY Safety Platform acts as a switch to power on/off the signal/the engine through a relay.

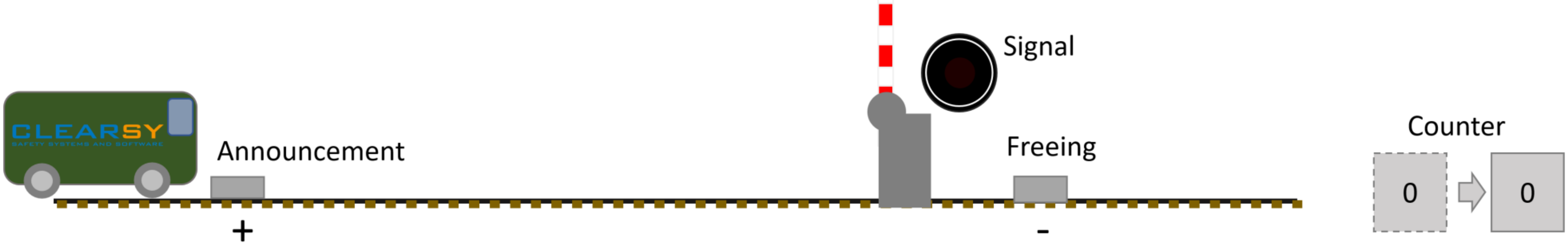
Controlled and controlling devices do not share the same power supply (safety rule)

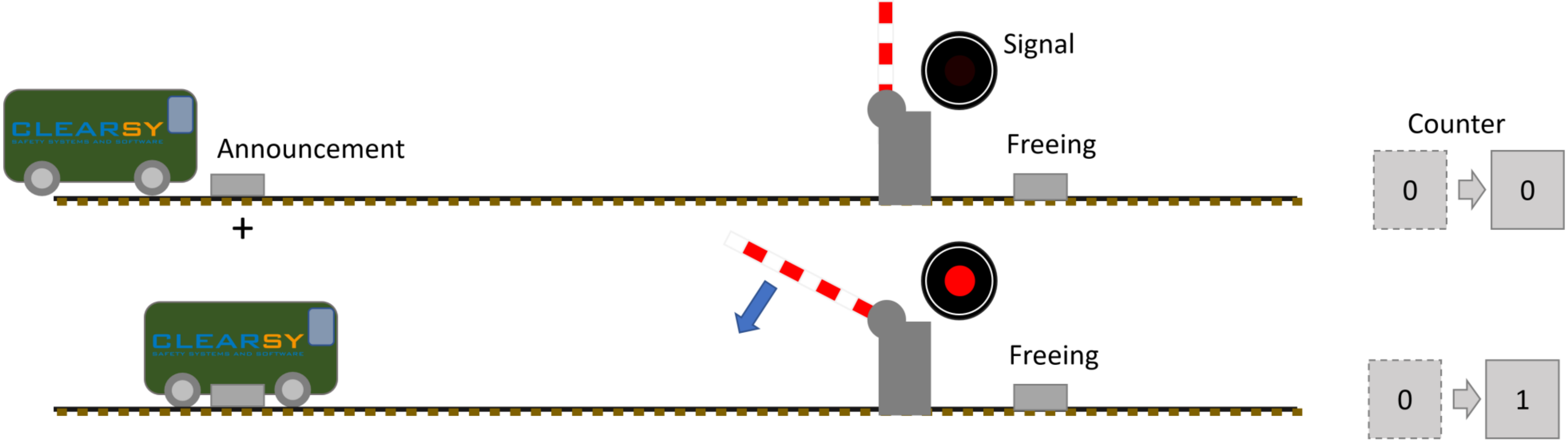


Sequence of events

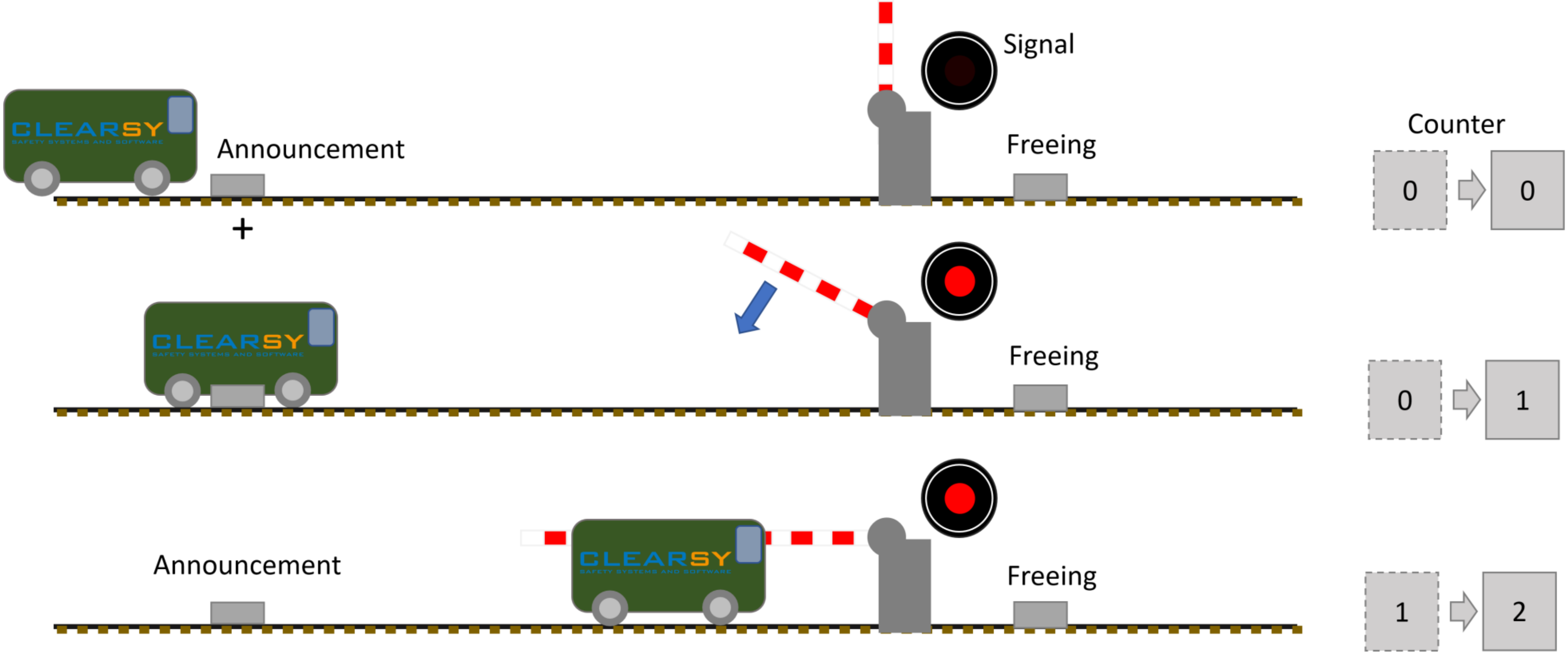
- An axle passes over the announcement pedal: the red light flashes, the barriers close
- Each time an axle passes over the pedal, the counter is incremented
- Each time an axle passes over the freeing pedal, the axle counter is decremented
- After the last axle of the train has passed (axle counter at 0): the red light go out and the barriers open

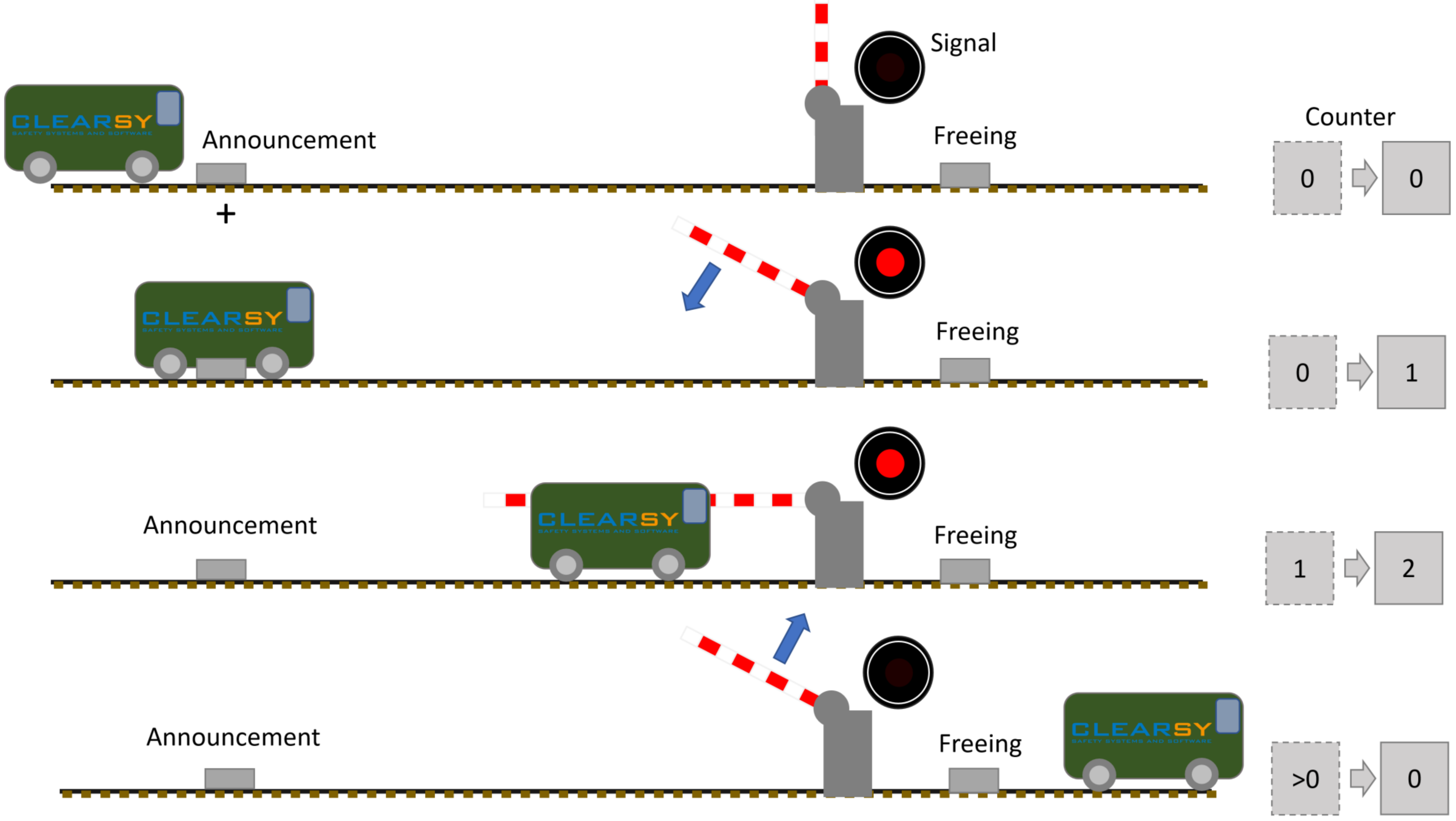



Signal
CLEARSY
Announcement
Freeing
Counter
0
0
+
-



Signal
Announcement
Freeing
Counter
0
0
+
Freeing
0
1


Signal
Announcement
Freeing
Counter
0
0
+
Freeing
0
1
Announcement
Freeing
1
2


Signal
Announcement
Freeing
Counter
0
0
+
Freeing
0
1
Announcement
Freeing
1
2
Announcement
Freeing
>0
0

The Railway Level Crossing Example

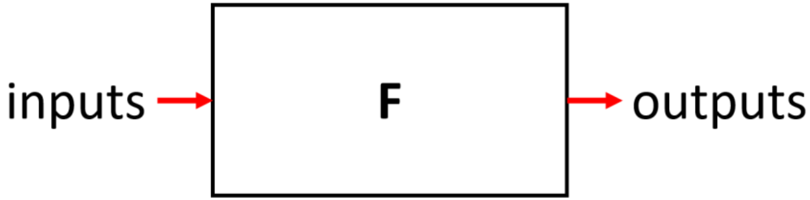
- [**Decision**] closed is safe – in case of « problems », no car is allowed to enter the level crossing
- Consequences:
  - The barrier engine is used to keep the barrier open
  - In case of power shortage, gravity is used to lower the barrier
  - In case of failing controller, the outputs ENG & SIG are OFF

    the relay should be guaranteed not to provide energy in case of absence of command

The Railway Level Crossing Example

- [**Question**] What to do in case of « negative counter » (more train axles leaving the zone than entering)?
- [**Question**] What to do in case of a train between the pedals when the controller is switched on?
- [**Question**] In case of power shortage, is it a problem to have the barrier going down while the signal is not on?
- [**Question**] What if someone blocked the barrier, preventing it to go down?
- [**Question**] Where to install the announcement pedal? (distance from Xing)
- The global picture:
  - contains the system, its environment, exploitation procedures, traffic, maintenance, human behaviour, etc.
  - relates to hypotheses made to restrict situations being considered

What is a Safety Computer?



F == (read inputs, compute, set outputs)*

F could harm / kill people

- Is a computer able
  - to check if able to execute F properly (✓ or ✗)
  - to adapt accordingly

What are ✓ and ✗ ?

✗ if
- memory corrupted (data, program, registers)
- incorrect computation
- incorrect timing reference
- incorrect output physical status

✓ if not(✗)

✓/✗ involve both hardware and software

History

R&D

CLEARSY Safety Platform building blocks certified

- 2017: platform screen door control system Sao Paulo, SIL4, CERTIFER
- 2017: platform screen door control system Stockholm, SIL3, Bureau Veritas
- 2019: vital remote I/O system, SIL4, Bureau Veritas

10 years of prior experience developing SIL4 systems with PLCs

Development

Invention of **CLEARSY Safety platform for Industry**

- 2021: Core safety computer certified SIL4, CERTIFER

2016 2017 2018 2019 2020 2021

R&D

LCHIP (Low Cost High Integrity Platform)

Collaborative Project with SNCF

Invention of **CLEARSY Safety platform for Education**

ABZ 2021

Tutorials

Brazil, Canada, France, Italy, Portugal, UK

today

Courses

CLEARSY Safety Platform for Education released

France, Italy

What is the CLEARSY Safety Platform?

- Safety computer implementing the verification ✓/✗
  - Save time and allow « less expert engineers» to develop
  - Programmed in B to obtain defect-free software

- No magic:
  - support is provided once software specification is available
  - if system study / design is incorrect, safety is not ensured
  - electronics skills required for safe interfacing with external world
  - F is not safe just because a safety computer is used

« execute the right F and execute the F right »

Main Principles

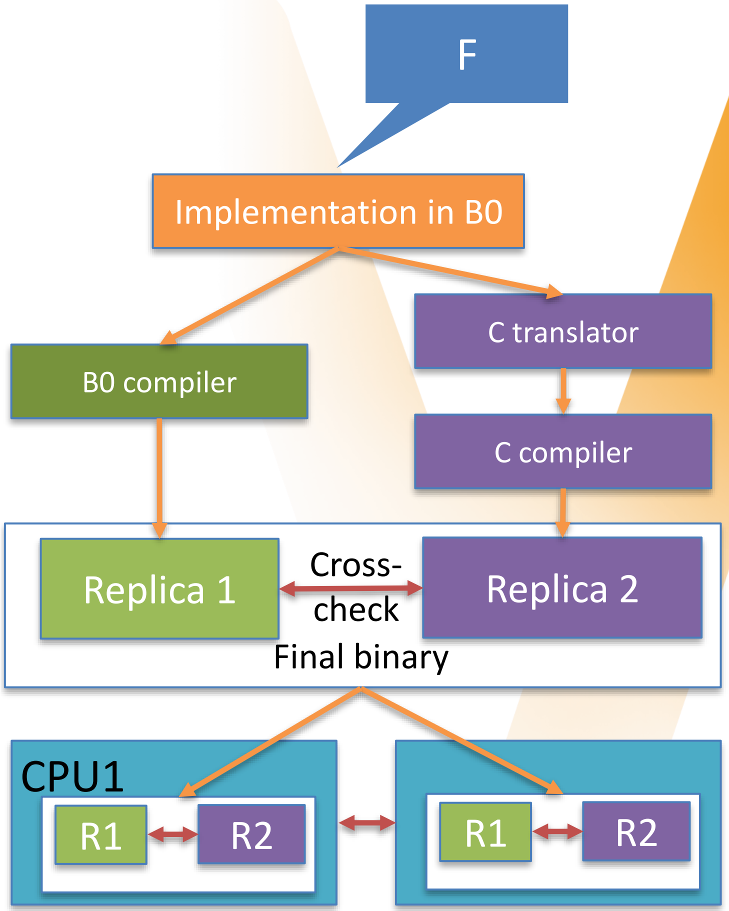
- F is a proved B formal model
- F code is generated automatically from model

- 2 identical microcontrollers to execute several diverse instances of F
  - PIC32MX from Microchip (80 DMIPS)

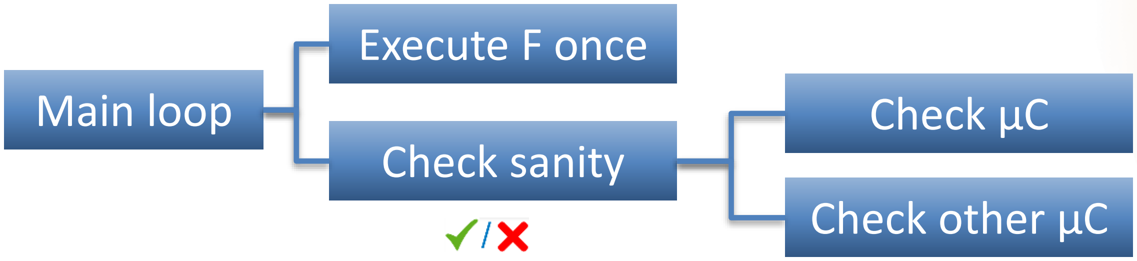
- Continuous behavioral verification

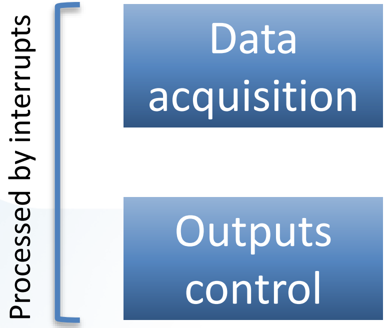
Architecture

- B0 as Internal Representation
  - F fully developed in B (preferred)
  - B0 translated from a DSL
  - Handwritten B0
- Composition of the final binary
  - The execution of the toolchain builds two independent binaries called replica.
- Binary diversification
  - Each replica is built by an independent and diverse compiler from the same formal model
  - **The software is written only once.** No need for two independent software design teams.
- Runtime verification
  - Both replicas are executed in sequence with the same input data. The **comparison of the output data of each replica** detects discrepancies and random failures during runtime.
  - Cross-checking mechanisms between CPU1 and CPU2 mitigate remaining failure modes.



Main Principles: software





- No underlying operating system
- Main loop is executed as fasy as possible
- Binary made of
  - Replicated code: subject to 2oo2 verification
  - Non-replicated code

Main Principles: software

Legend
- to develop
- to complement
- developped or generated

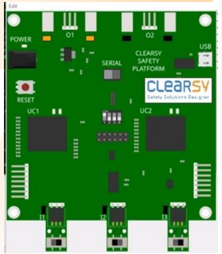
Main loop
- Execute F once
- Check sanity
  - Check µC
  - Check other µC

✓/✗

Processed by interrupts
- Data acquisition
- Outputs control

CLEARSY Safety Platform for Education
- F as a B model only (replicated code)
- Most verifications implemented
- Cost effective hardware interface
- Cannot be used for real life safety app

IDE + board
3 inputs, 2 outputs

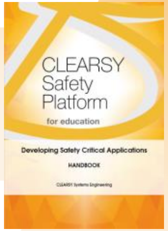


IDE + board emulator
3 inputs, 2 outputs

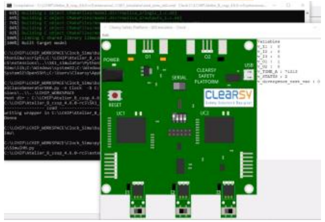
CLEARSY Safety Platform for Education

- Easy to set-up, program, and experiment with
  - Specialized Atelier B, push-button compilation toolchain & upload
  - Triggered by switches and arduino-based schematics
  - Up to 3 digital inputs and 2 digital outputs
  - Board software emulator for virtual modelling only

- Programming handbook available with examples
  - Introduction to specification and programming in B
  - Combinatorial and synchronous examples
  - https://github.com/CLEARSY/CSSP-Programming-Handbook



- Software emulator
  - Specialized Atelier B, board graphical animation
  - https://github.com/CLEARSY/tutorial-ABZ-2021/ section « Atelier CLEARSY Safety Platform »



Main Principles: software

Legend

- to develop
- to complement
- developped or generated

Main loop

Execute F once

Check sanity

✓ / ✗

Check µC

Check other µC

Processed by interrupts

Data acquisition

Outputs control

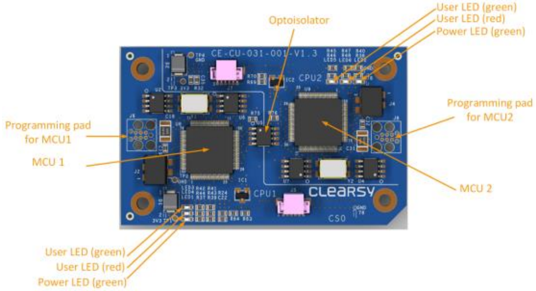
CLEARSY Safety Platform for Industry

- B and C used for sequential and interrupted code
- More adaptable to specific needs
- All required verifications implemented
- I/O hosted on a motherboard to develop

compilation toolchain + core computer + SK motherboard 32 inputs, 32 outputs

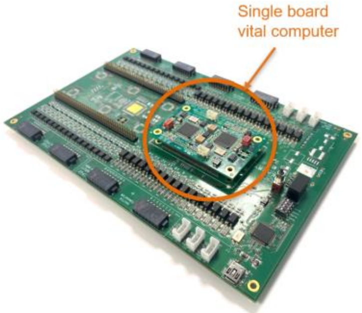
CLEARSY Safety Platform for Industry

- **Core Computer CS_0**
  - Smartcard format, power consumption < 2W
  - Precompiled binary objects library CSPlib (boot, BSP, drivers)
  - Docker-based toolchain
  - More versatile than a PLC (Programmable Logic Controller)
  - No limit in term of kind and count of interfaces
  - Can be seen and integrated like a component
  - SIL4 design ready (**SIL4 certificate**, 2021, Certifer)
  - Safety functions fully transparent to the user



CS_0 Core Computer

- **Requires a motherboard for power and I/O**
  - Starter kit for PoC (format A5)
  - Mini USB plug for power and debug interfaces (2x serial ports)
  - 32x digital inputs, 32x digital outputs
  - 2,54mm pitch row header pins for extension and interface prototyping
  - Extra plug for I²C or UART



Starter kit motherboard with CS_0 plugged

Before jumping to part 2 and 3

- Live demonstration of the level crossing controller
- CLEARSY Safety Platform for Education
  - Switch-based interface
  - Modelling, proof, code generation and compilation, upload, execution
  - With the real board, with the emulated board
- CLEARSY Safety Platform for Industry
  - 3D printed, button-based interface
  - Overview of the installation
  - Focus on some bits of the software and formal model (+ metrics)
  - Testing nominal and dysfunctional behaviour