Distributed and Heterogeneous Event-based Monitoring in Smart Cyber-Physical Systems

László Balogh, István Dávid, István Ráth, Dániel Varró and **András Vörös**

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group

Overview

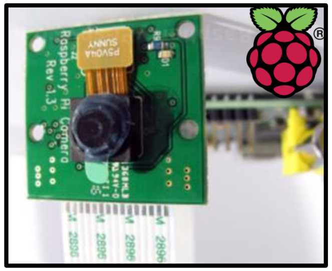
- Smart cyber-physical systems
  - Motivation: the MoDeS3 case-study
- Complex-event processing with VIATRA-CEP
- Ongoing work

MODEL-BASED DEMONSTRATOR FOR SMART AND SAFE SYSTEMS

Big Picture

- Traditional **safety-critical** systems:
  - Model-based development
  - Validation & verification
  - Code generation
  - Safety requirements

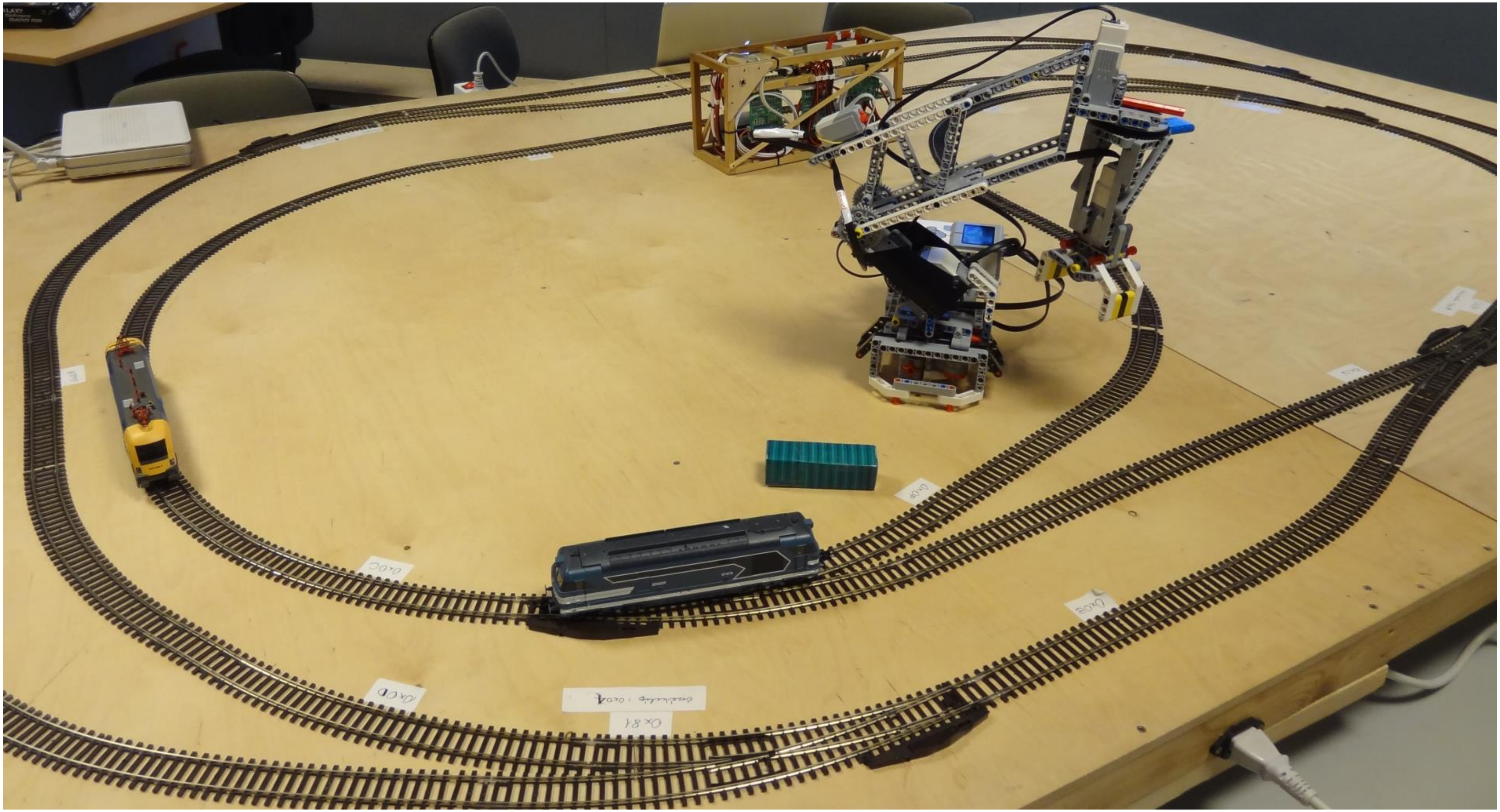
- **Cyber-physical** systems:
  - Various information sources (sensors)
  - Heterogeneous: Embedded computers & cloud computing



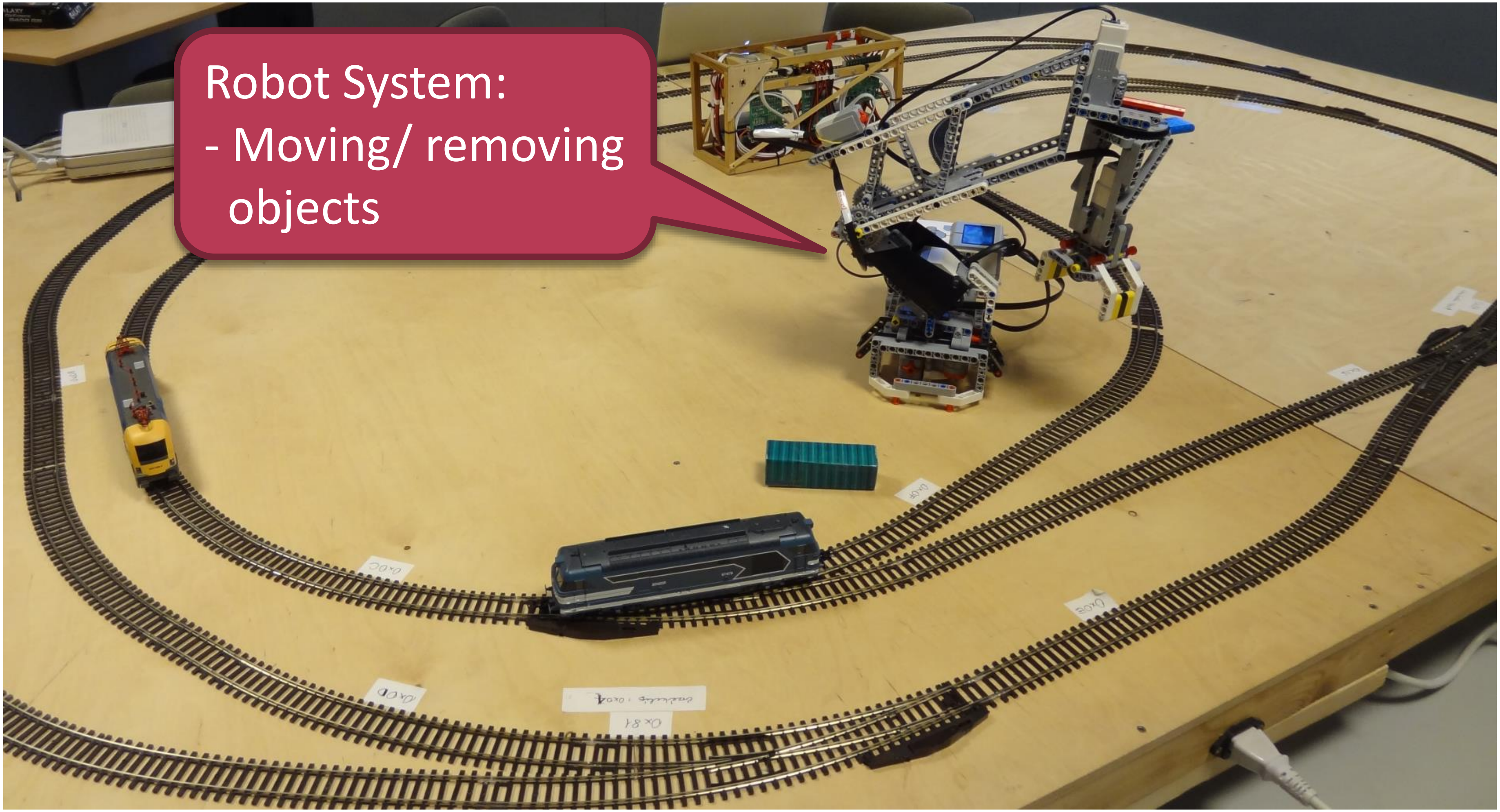
Combination of both worlds:

Development techniques used for **safety-critical systems** with **technologies** from **cyber-physical systems**

Demonstrator

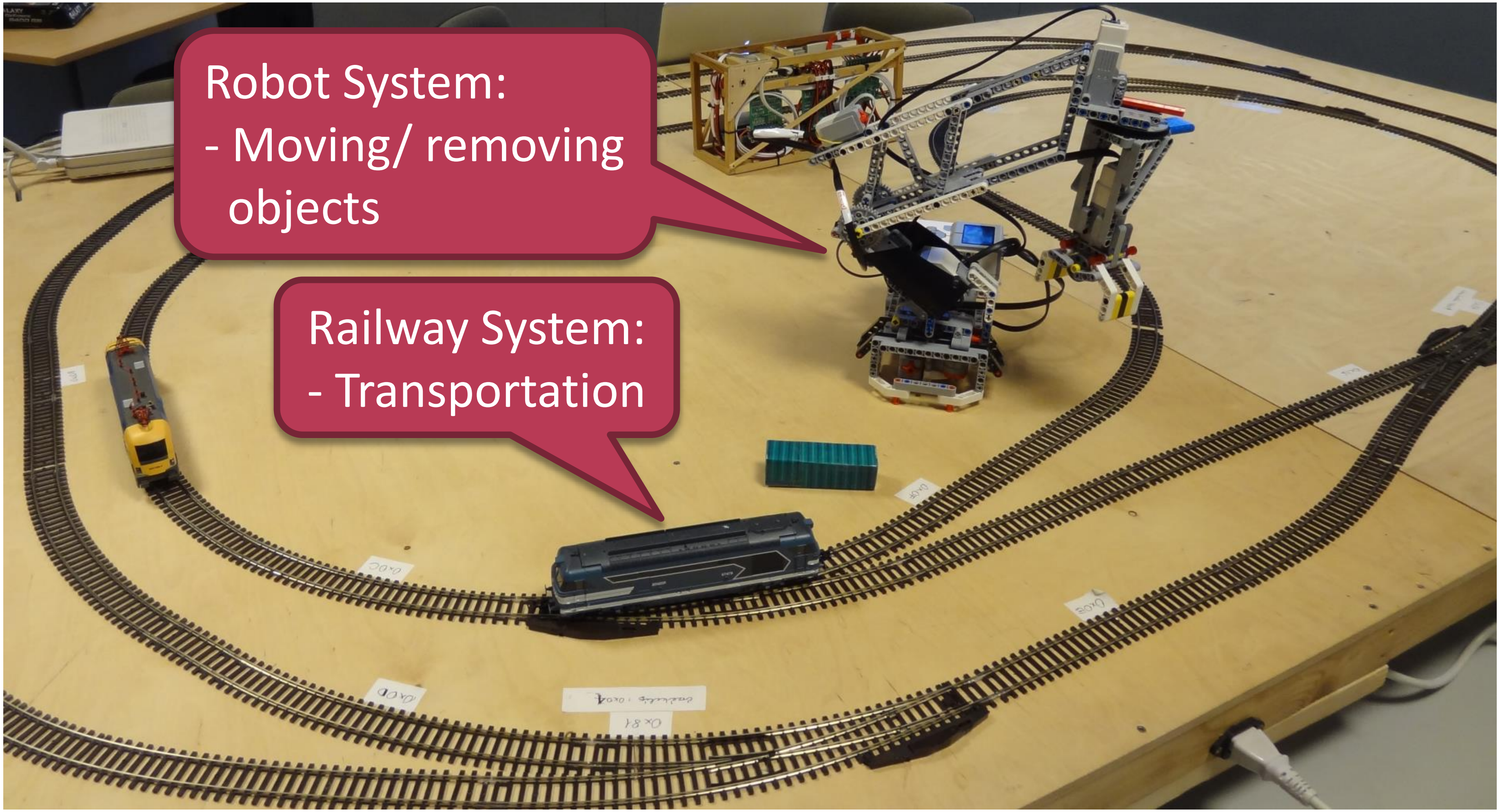


Demonstrator



Robot System:
- Moving/ removing objects

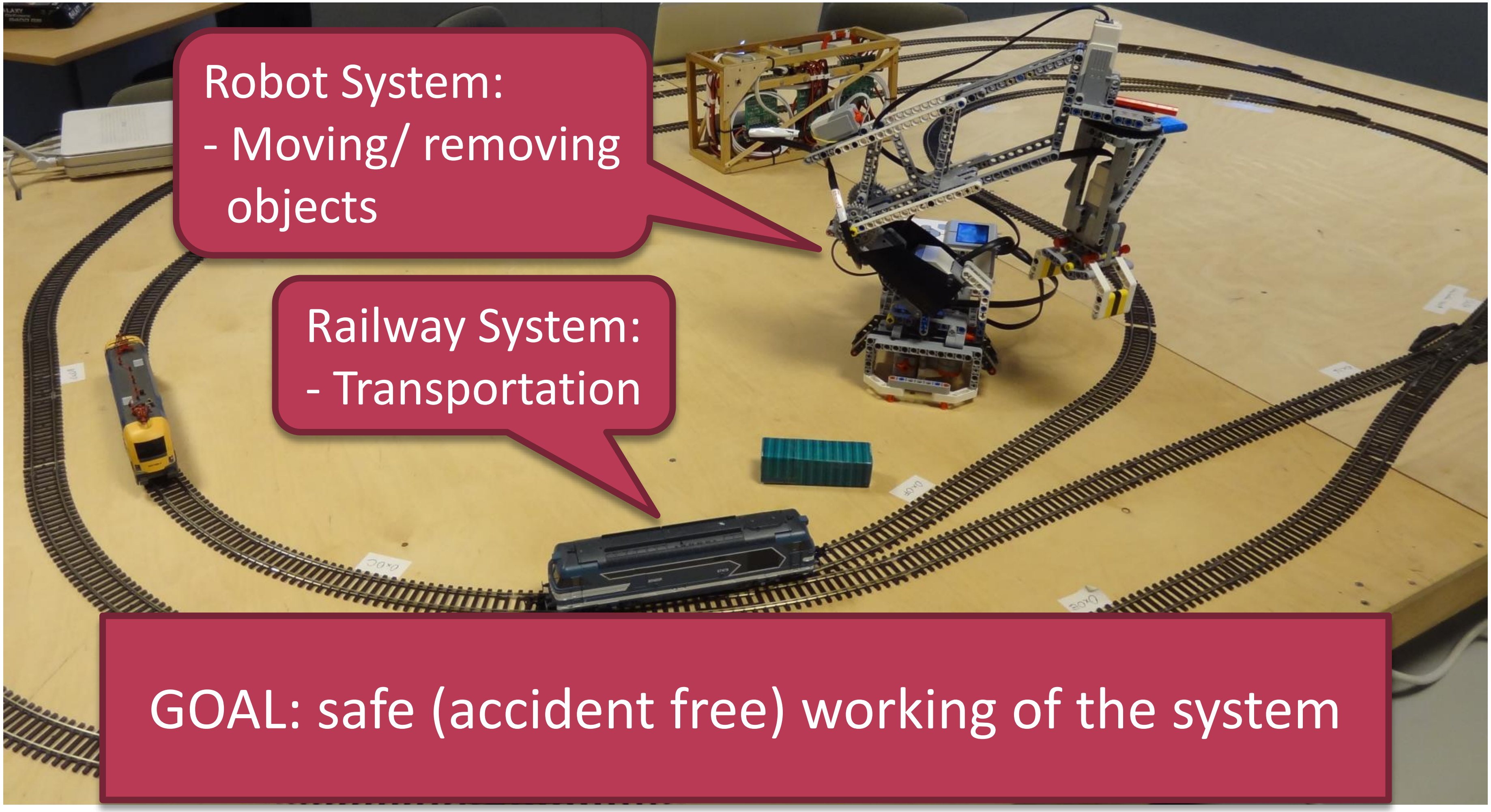
Demonstrator



Robot System:
- Moving/ removing objects

Railway System:
- Transportation

Demonstrator

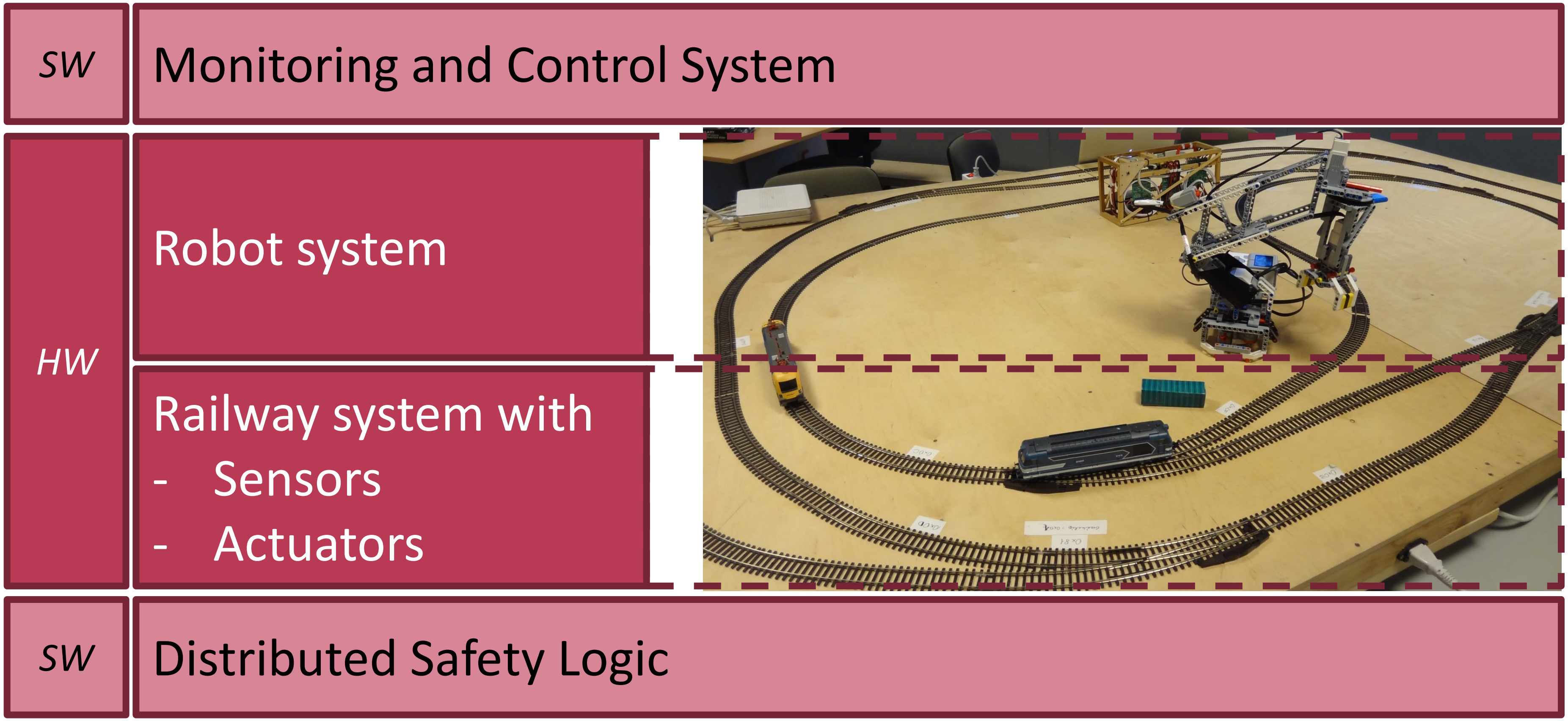


Robot System:
- Moving/ removing objects

Railway System:
- Transportation

GOAL: safe (accident free) working of the system

MoDeS3



SW

Monitoring and Control System

HW

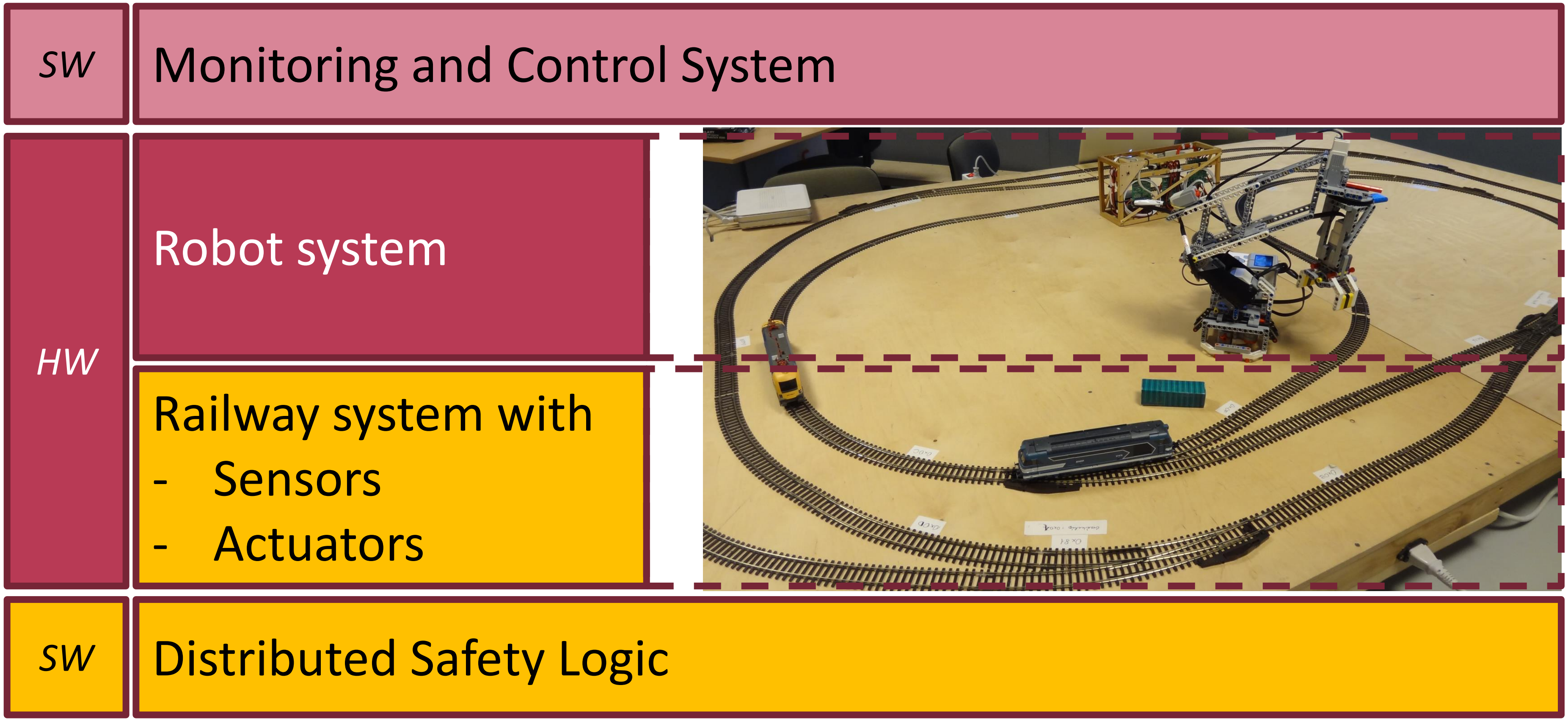
Robot system

Railway system with
- Sensors
- Actuators

SW

Distributed Safety Logic

MoDeS3



SW

Monitoring and Control System

HW

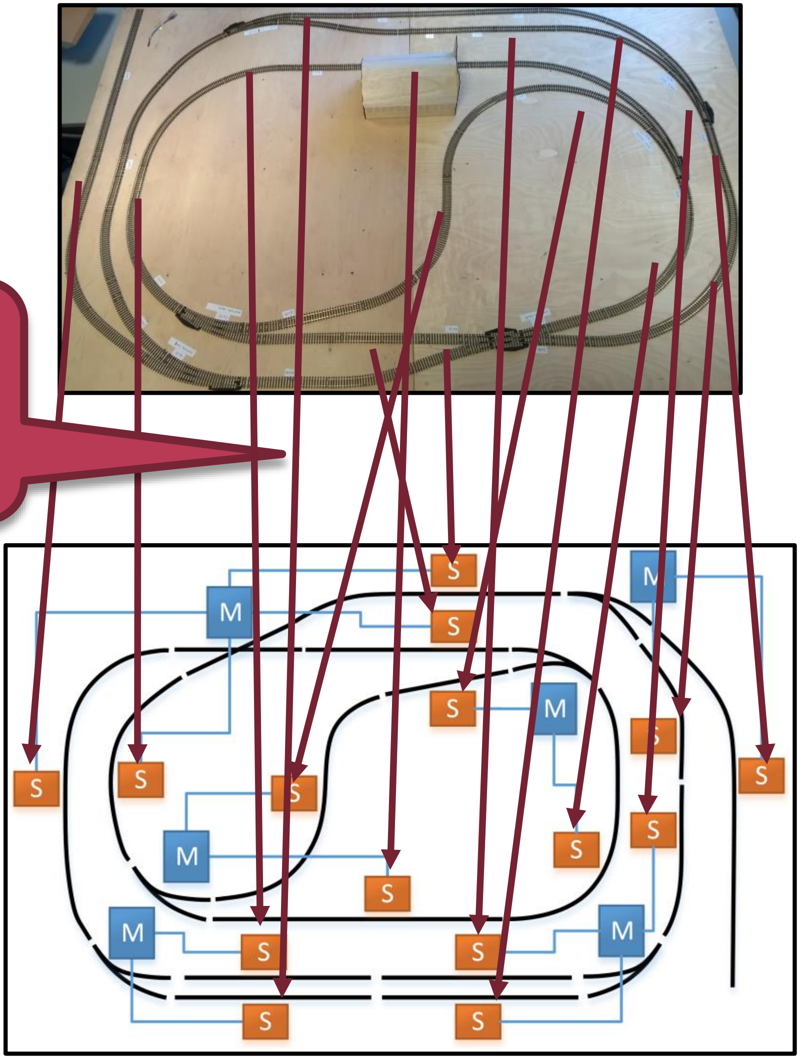
Robot system

Railway system with
- Sensors
- Actuators

SW

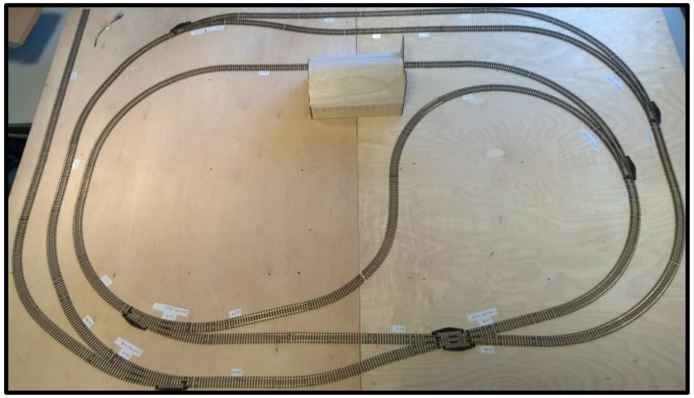
Distributed Safety Logic

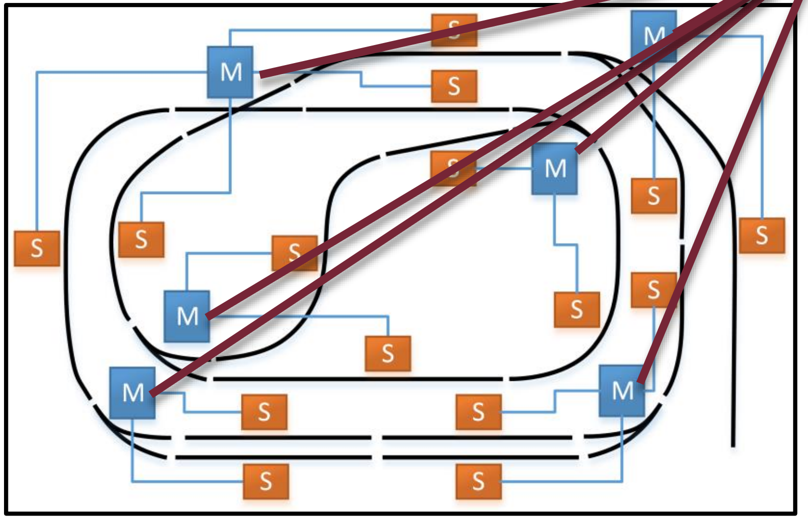
Distributed Safety Logic



15 sensors:
- Sensing the trains and estimating their locations

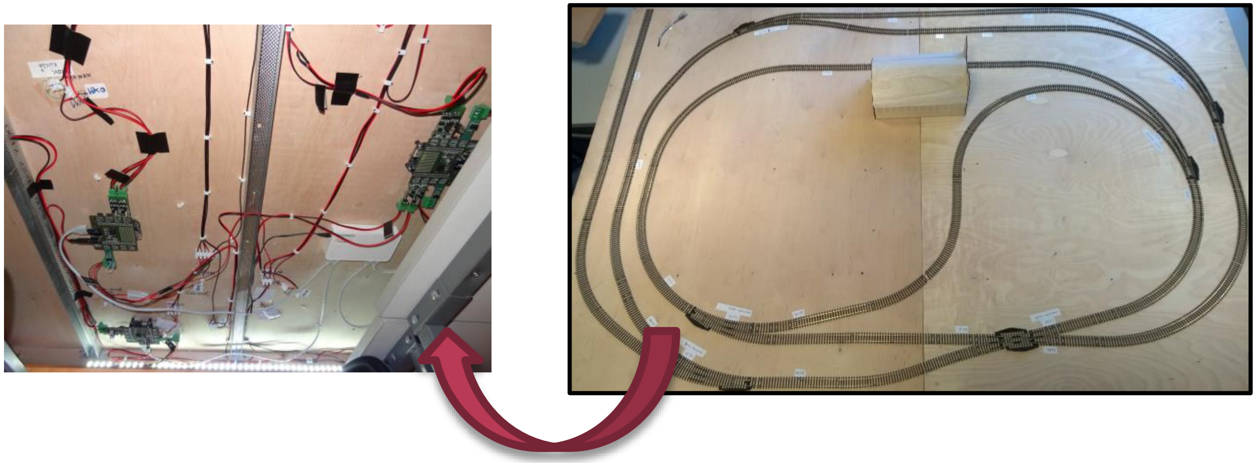
Distributed Safety Logic

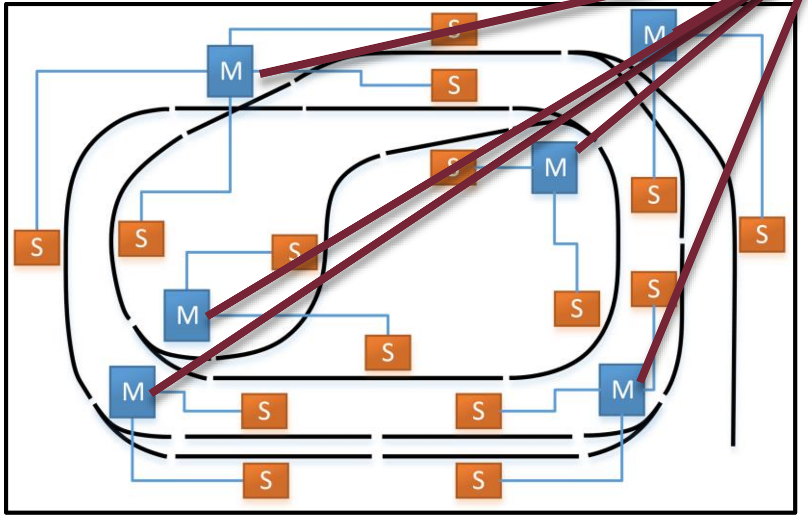




6 embedded controllers:
- Actuators

Distributed Safety Logic

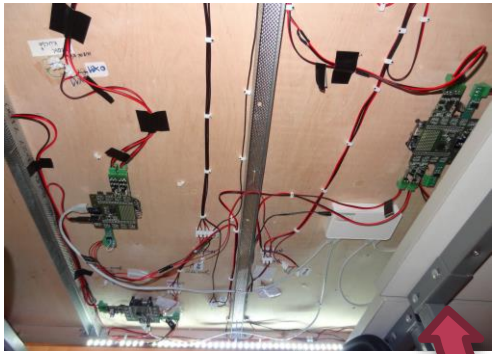




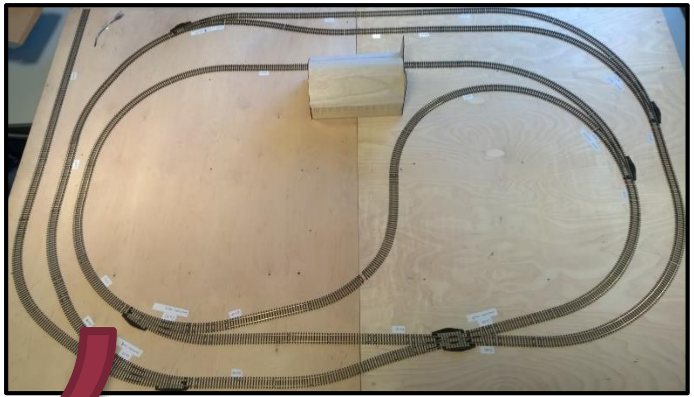
6 embedded controllers:

- Actuators

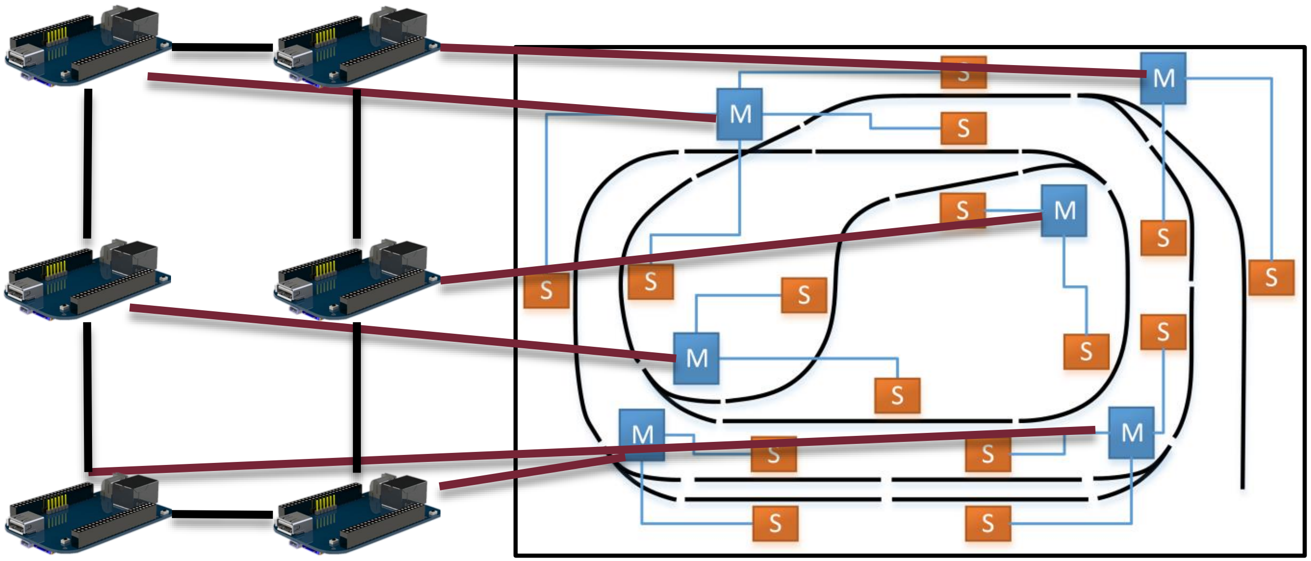
Distributed Safety Logic



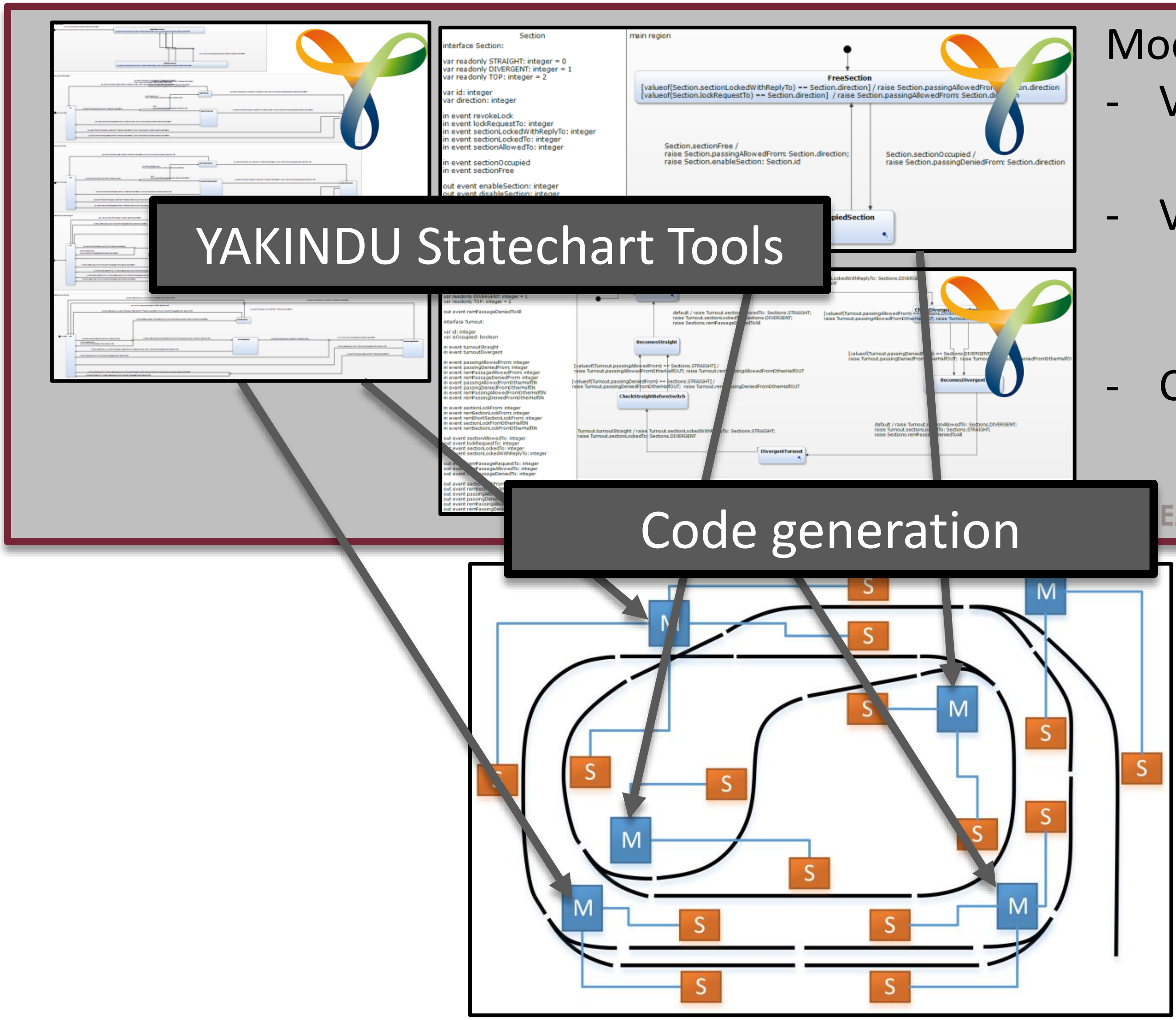
beagleboard.org



- Distributed:
  - 6 controllers
  - Communication
- Safety: prevent accidents by stopping the trains



Distributed Safety Logic



```
Section
interface Section:

var readonly STRAIGHT: integer = 0
var readonly DIVERGENT: integer = 1
var readonly TOP: integer = 2

var id: integer
var direction: integer

in event revokeLock
in event lockRequestTo: integer
in event sectionLockedWithReplyTo: integer
in event sectionLockedTo: integer
in event sectionAllowedTo: integer

in event sectionOccupied
in event sectionFree

out event enableSection: integer
out event disableSection: integer
```

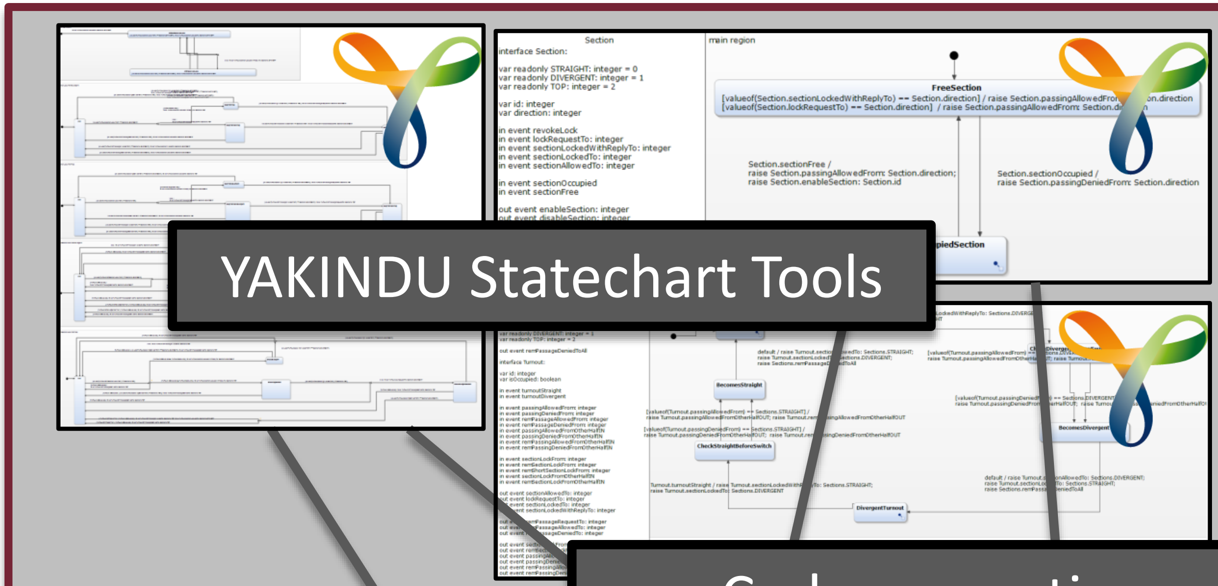
main region
FreeSection
[valueof(Section.sectionLockedWithReplyTo) == Section.direction] / raise Section.passingAllowedFrom: Section.direction
[valueof(Section.lockRequestTo) == Section.direction] / raise Section.passingAllowedFrom: Section.direction
Section.sectionFree /
raise Section.passingAllowedFrom: Section.direction;
raise Section.enableSection: Section.id
Section.sectionOccupied /
raise Section.passingDeniedFrom: Section.direction
OccupiedSection

```
var readonly DIVERGENT: integer = 1
var readonly TOP: integer = 2

out event remPassageDeniedToAll

interface Turnout:

var id: integer
var isOccupied: boolean

in event turnoutStraight
in event turnoutDivergent

in event passingAllowedFrom: integer
in event passingDeniedFrom: integer
in event remPassageAllowedFrom: integer
in event remPassageDeniedFrom: integer
in event passingAllowedFromOtherHalfIN
in event passingDeniedFromOtherHalfIN
in event remPassingAllowedFromOtherHalfIN
in event remPassingDeniedFromOtherHalfIN

in event sectionLockFrom: integer
in event remSectionLockFrom: integer
in event remShortSectionLockFrom: integer
in event sectionLockFromOtherHalfIN
in event remSectionLockFromOtherHalfIN

out event sectionAllowedTo: integer
out event lockRequestTo: integer
out event sectionLockedTo: integer
out event sectionLockedWithReplyTo: integer

out event remPassageRequestTo: integer
out event remPassageAllowedTo: integer
out event remPassageDeniedTo: integer
```

Model-driven development

- Validation techniques
  - *VIATRA Query*
- Verification techniques
  - Model-transformation
  - *VIATRA*
- Code generation

Distributed Safety Logic



```
Section
interface Section:

var readonly STRAIGHT: integer = 0
var readonly DIVERGENT: integer = 1
var readonly TOP: integer = 2

var id: integer
var direction: integer

in event revokeLock
in event lockRequestTo: integer
in event sectionLockedWithReplyTo: integer
in event sectionLockedTo: integer
in event sectionAllowedTo: integer

in event sectionOccupied
in event sectionFree

out event enableSection: integer
out event disableSection: integer
```

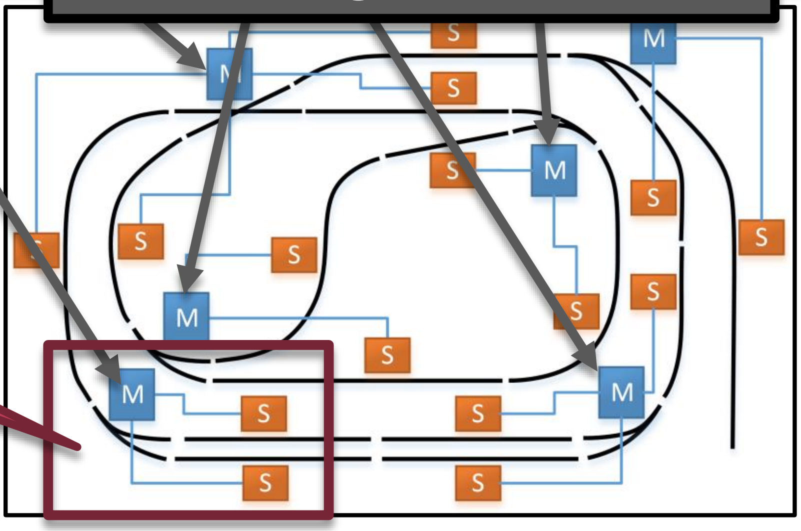
```
main region

FreeSection
[valueof(Section.sectionLockedWithReplyTo) == Section.direction] / raise Section.passingAllowedFrom: Section.direction
[valueof(Section.lockRequestTo) == Section.direction] / raise Section.passingAllowedFrom: Section.direction

Section.sectionFree /
raise Section.passingAllowedFrom: Section.direction;
raise Section.enableSection: Section.id

Section.sectionOccupied /
raise Section.passingDeniedFrom: Section.direction

OccupiedSection
```

YAKINDU Statechart Tools

Model-driven development

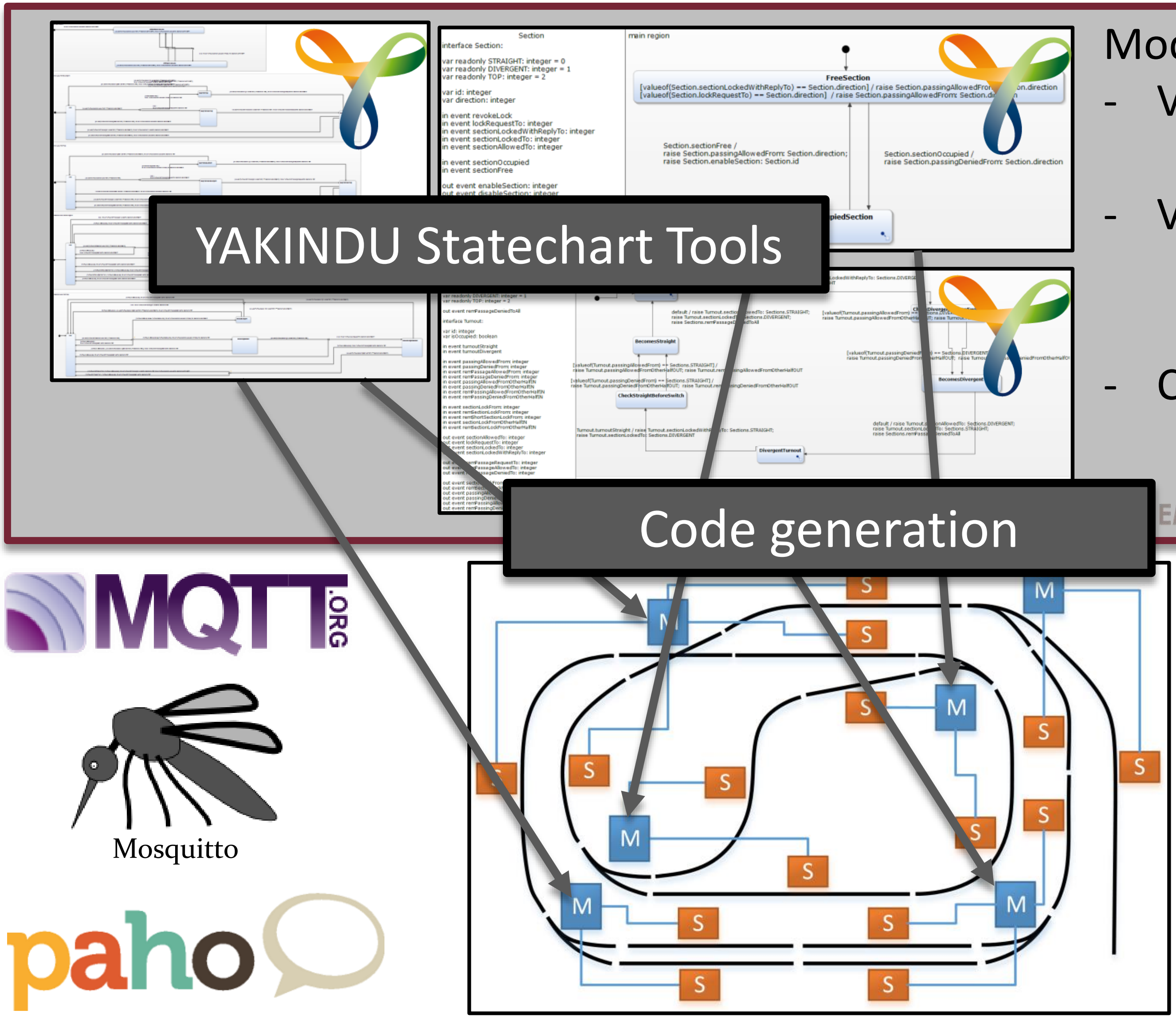
- Validation techniques
  - *VIATRA Query*
- Verification techniques
  - Model-transformation
  - *VIATRA*
- Code generation

Code generation



Each track section is controlled by a dedicated BBB

Distributed Safety Logic



```
Section
interface Section:

var readonly STRAIGHT: integer = 0
var readonly DIVERGENT: integer = 1
var readonly TOP: integer = 2

var id: integer
var direction: integer

in event revokeLock
in event lockRequestTo: integer
in event sectionLockedWithReplyTo: integer
in event sectionLockedTo: integer
in event sectionAllowedTo: integer

in event sectionOccupied
in event sectionFree

out event enableSection: integer
out event disableSection: integer
```

main region
FreeSection
[valueof(Section.sectionLockedWithReplyTo) == Section.direction] / raise Section.passingAllowedFrom: Section.direction
[valueof(Section.lockRequestTo) == Section.direction] / raise Section.passingAllowedFrom: Section.direction
Section.sectionFree / raise Section.passingAllowedFrom: Section.direction; raise Section.enableSection: Section.id
Section.sectionOccupied / raise Section.passingDeniedFrom: Section.direction
OccupiedSection
BecomesStraight
CheckStraightBeforeSwitch
DivergentTurnout
BecomesDivergent
YAKINDU Statechart Tools
Code generation
M
S

Model-driven development

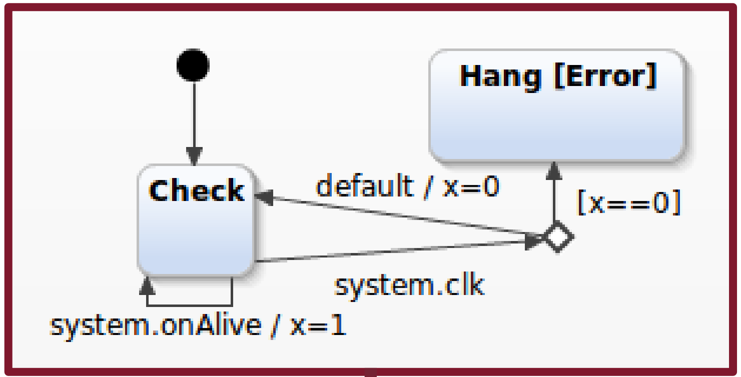
- Validation techniques
  - *VIATRA Query*
- Verification techniques
  - Model-transformation
  - *VIATRA*
- Code generation

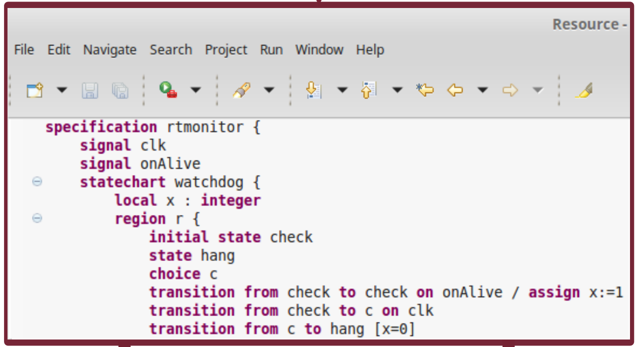
- IoT technologies for communication
- MQTT
  - Eclipse Paho
  - Mosquitto

Mosquitto

Component Level Runtime Verification

- Formal specification language: statechart
  - Hierarchical
  - Timed
  - Parametric
- Runtime monitor generation
- Formal semantics
  - Analysis





```
specification rtmonitor {
    signal clk
    signal onAlive
    statechart watchdog {
        local x : integer
        region r {
            initial state check
            state hang
            choice c
            transition from check to check on onAlive / assign x:=1
            transition from check to c on clk
            transition from c to hang [x=0]
```

Formal model

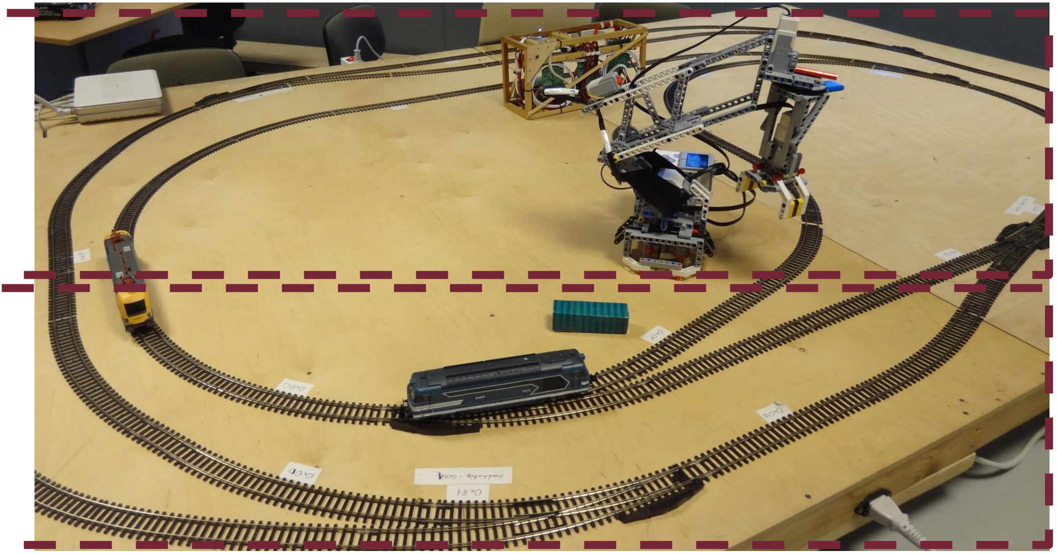
C++ Monitor

MoDeS3

SW | Monitoring and Control System

HW | Robot system

Railway system with
- Sensors
- Actuators



SW | Distributed Safety Logic

Monitoring and Control System

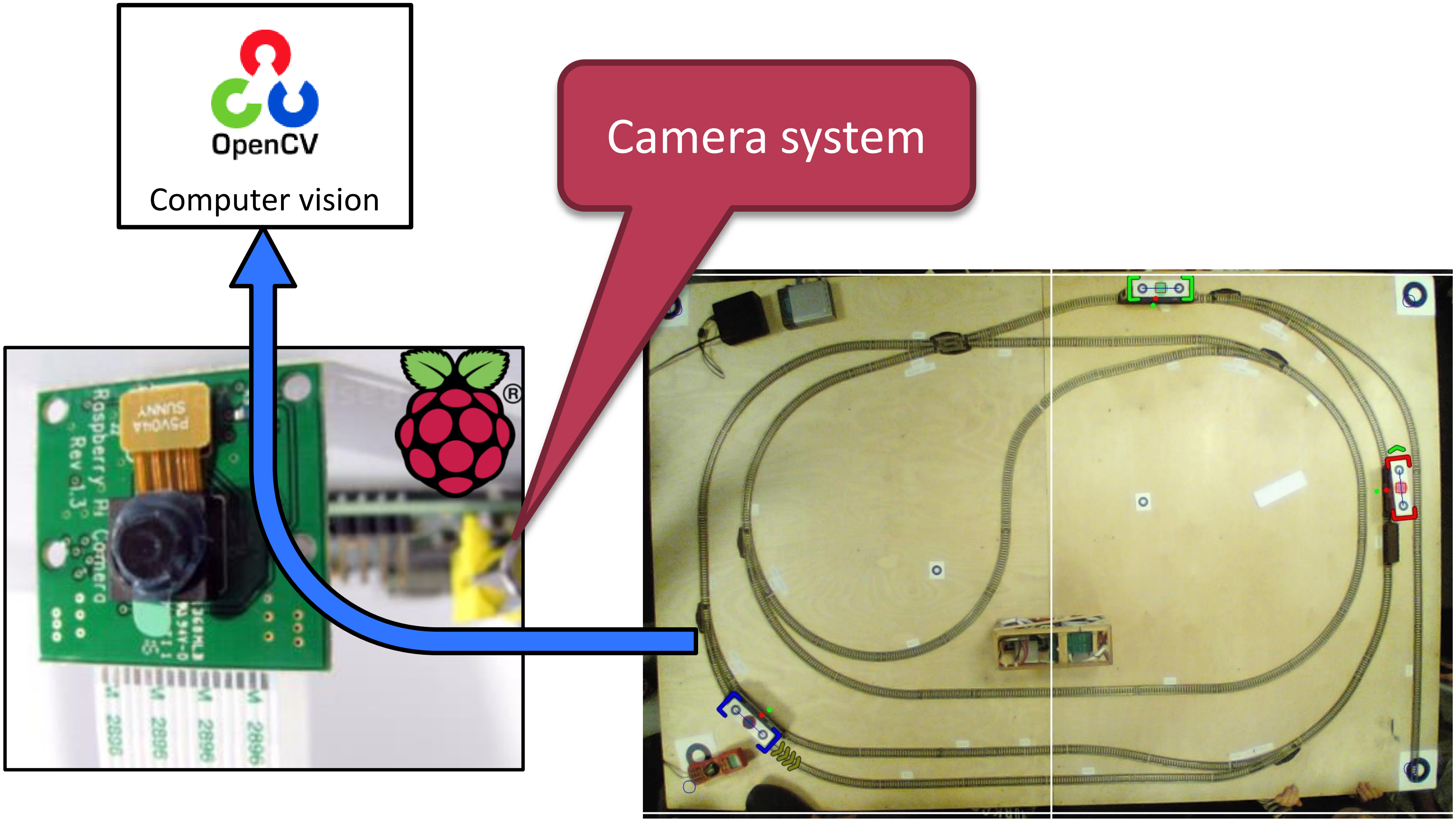
- Additional level of safety – high level monitoring

OpenCV

Computer vision

Monitoring and Control System

- Additional level of safety – high level monitoring

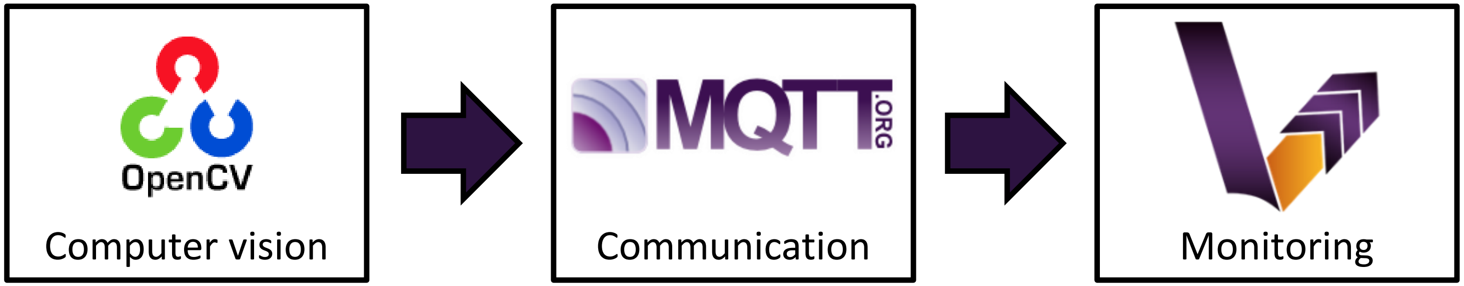


Computer vision

Camera system

Monitoring and Control System

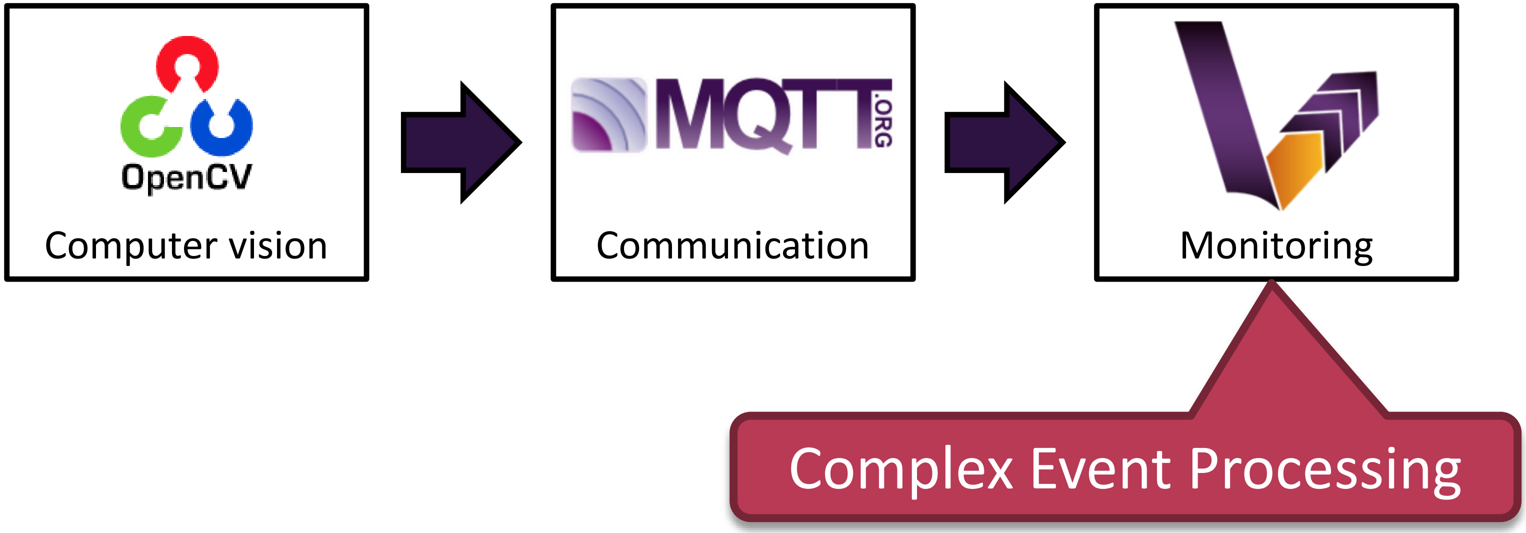
- Additional level of safety – high level monitoring



Data stream

Monitoring and Control System

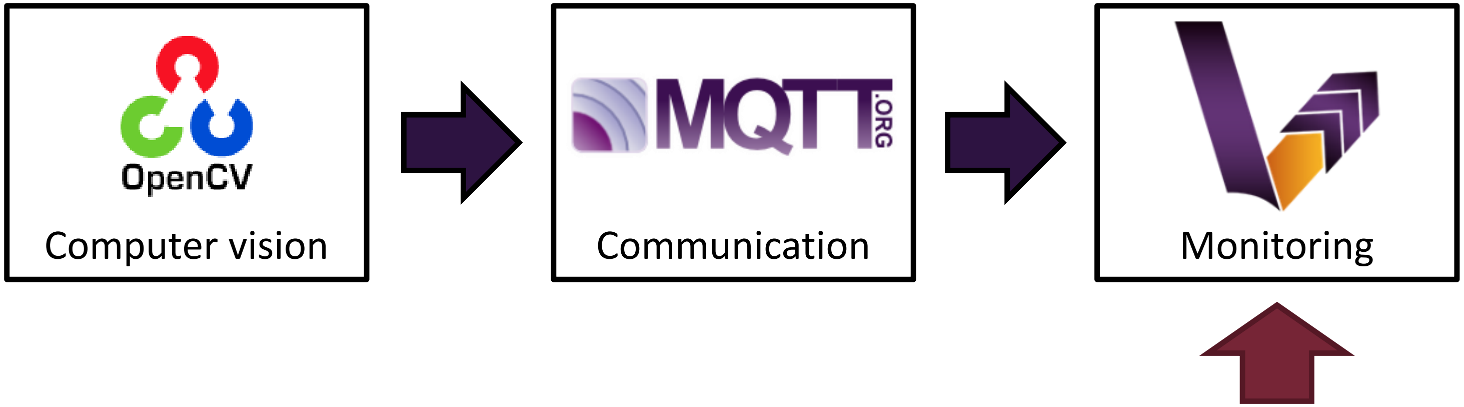
- Additional level of safety – high level monitoring

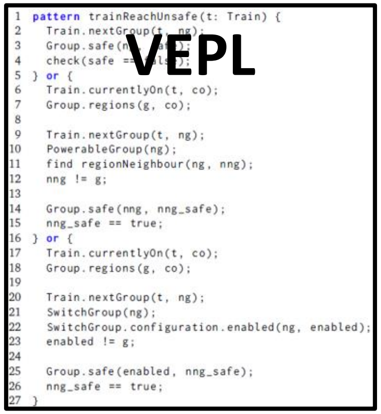


Complex Event Processing

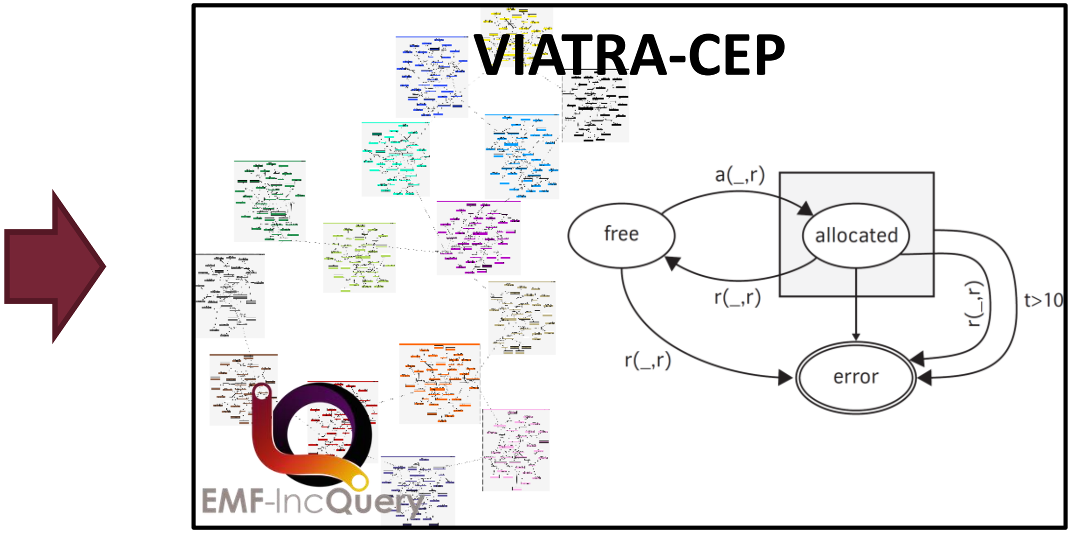
Monitoring and Control System

- Additional level of safety – high level monitoring



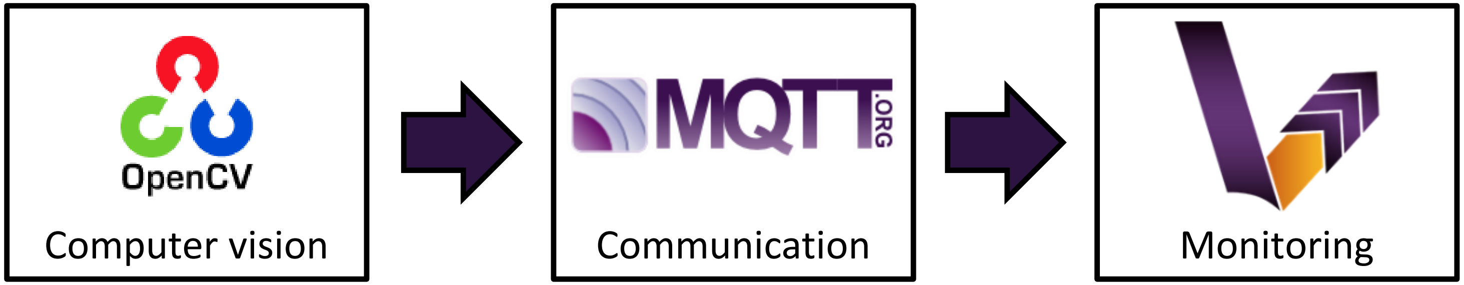


```
pattern trainReachUnsafe(t: Train) {
  Train.nextGroup(t, ng);
  Group.safe(ng, safe);
  check(safe == false);
} or {
  Train.currentlyOn(t, co);
  Group.regions(g, co);

  Train.nextGroup(t, ng);
  PowerableGroup(ng);
  find regionNeighbour(ng, nng);
  nng != g;

  Group.safe(nng, nng_safe);
  nng_safe == true;
} or {
  Train.currentlyOn(t, co);
  Group.regions(g, co);

  Train.nextGroup(t, ng);
  SwitchGroup(ng);
  SwitchGroup.configuration.enabled(ng, enabled);
  enabled != g;

  Group.safe(enabled, nng_safe);
  nng_safe == true;
}
```

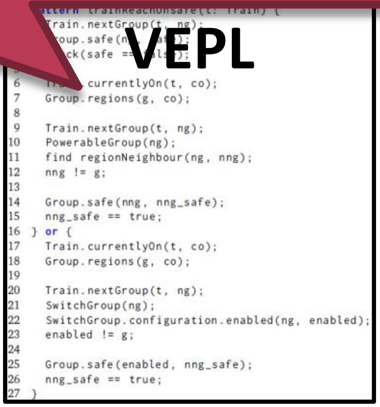


Monitoring and Control System

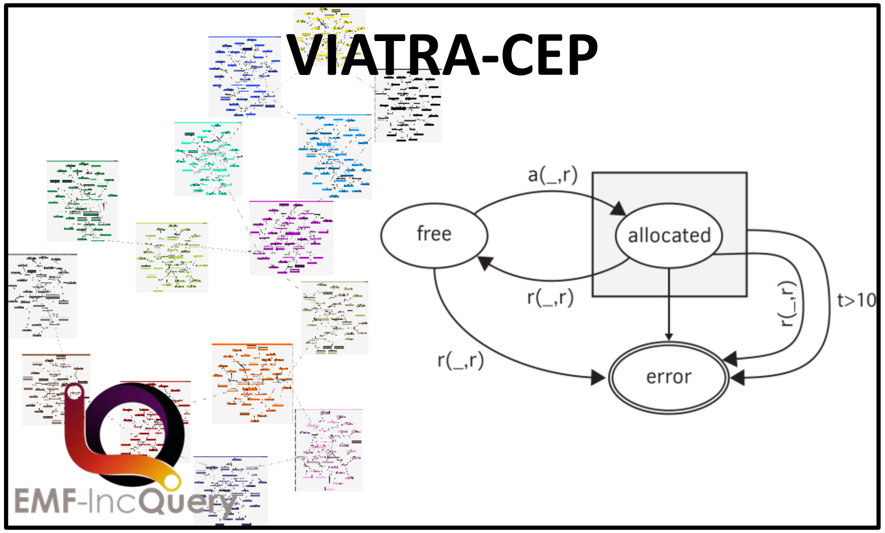
- Additional level of safety – high level monitoring



Shut down the system in case of dangerous situation

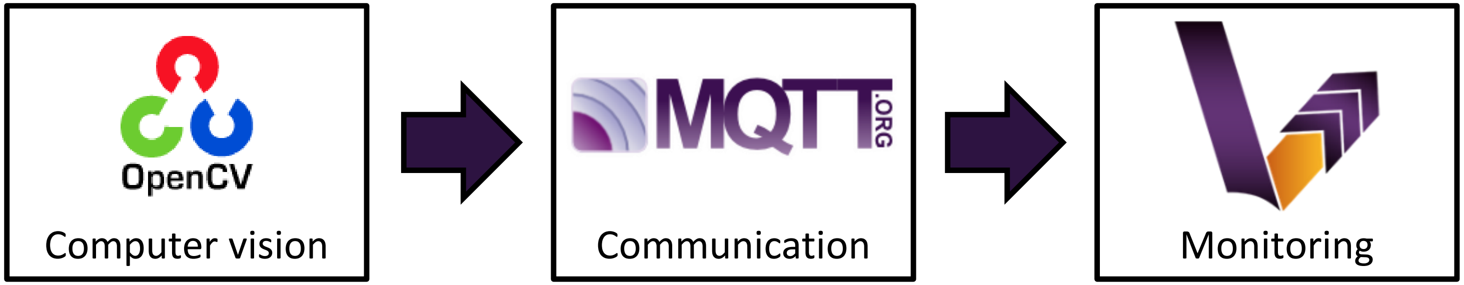


```
attern trainReachUnsafe(t: Train) {
 Train.nextGroup(t, ng);
 roup.safe(n     );
   ck(safe ==  ls );

     .currentlyOn(t, co);
 Group.regions(g, co);

 Train.nextGroup(t, ng);
 PowerableGroup(ng);
 find regionNeighbour(ng, nng);
 nng != g;

 Group.safe(nng, nng_safe);
 nng_safe == true;
} or {
 Train.currentlyOn(t, co);
 Group.regions(g, co);

 Train.nextGroup(t, ng);
 SwitchGroup(ng);
 SwitchGroup.configuration.enabled(ng, enabled);
 enabled != g;

 Group.safe(enabled, nng_safe);
 nng_safe == true;
}
```



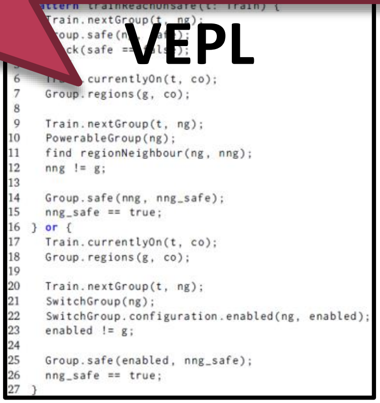
Monitoring and Control System

- Additional level of safety – high level monitoring

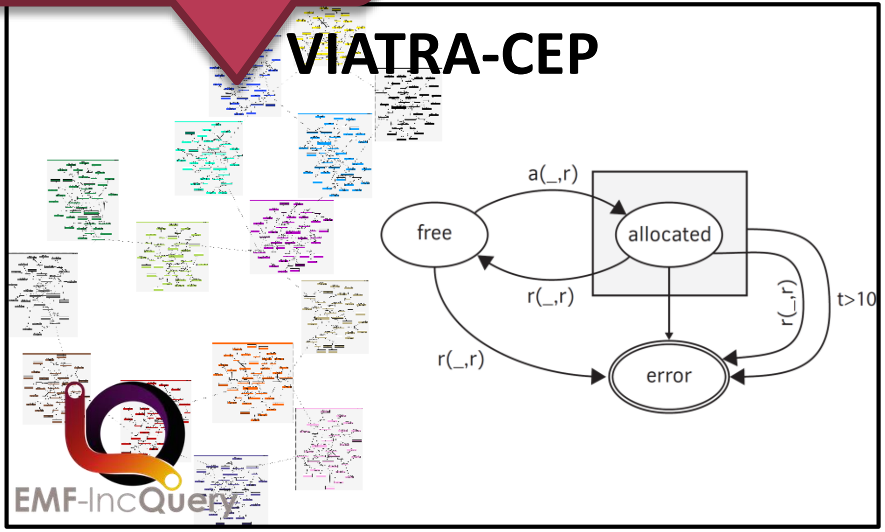


Shut down the system in case of dangerous situation

Monitoring logic

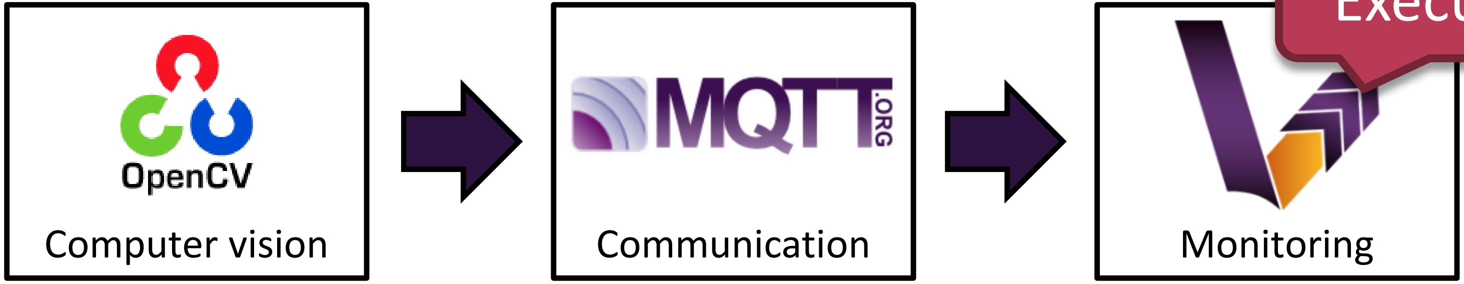


```
attern trainReachUnsafe(t: Train) {
Train.nextGroup(t, ng);
roup.safe(n
ck(safe == f

Tr  .currentlyOn(t, co);
  Group.regions(g, co);

  Train.nextGroup(t, ng);
  PowerableGroup(ng);
  find regionNeighbour(ng, nng);
  nng != g;

  Group.safe(nng, nng_safe);
  nng_safe == true;
} or {
  Train.currentlyOn(t, co);
  Group.regions(g, co);

  Train.nextGroup(t, ng);
  SwitchGroup(ng);
  SwitchGroup.configuration.enabled(ng, enabled);
  enabled != g;

  Group.safe(enabled, nng_safe);
  nng_safe == true;
}
```



Monitoring and Control System

- Additional level of safety – high level monitoring



Computer vision

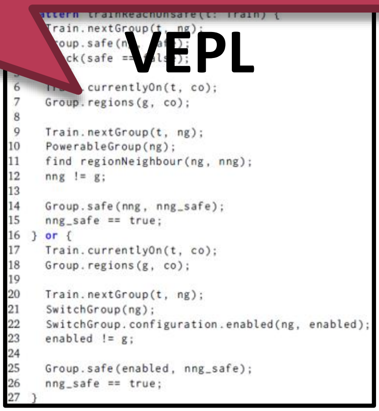
Communication

Monitoring

Execution

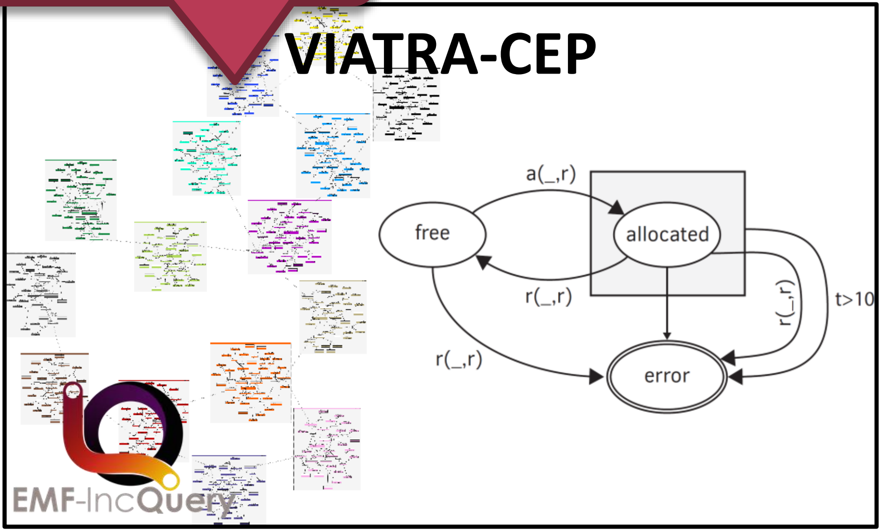
Shut down the system in case of dangerous situation

Monitoring logic



VEPL

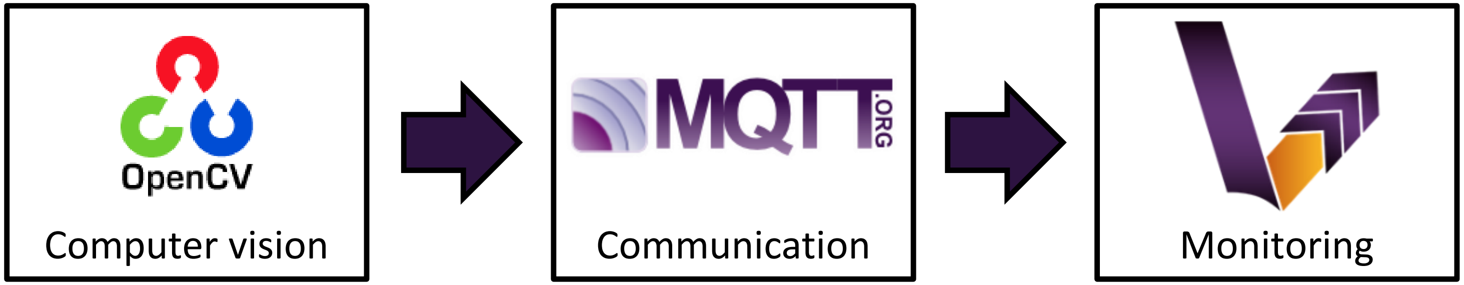
```
attern trainReachUnsafe(t: Train) {
  Train.nextGroup(t, ng);
  roup.safe(n   afe);
    ck(safe ==   ls  );

  Tr   currentlyOn(t, co);
  Group.regions(g, co);

  Train.nextGroup(t, ng);
  PowerableGroup(ng);
  find regionNeighbour(ng, nng);
  nng != g;

  Group.safe(nng, nng_safe);
  nng_safe == true;
} or {
  Train.currentlyOn(t, co);
  Group.regions(g, co);

  Train.nextGroup(t, ng);
  SwitchGroup(ng);
  SwitchGroup.configuration.enabled(ng, enabled);
  enabled != g;

  Group.safe(enabled, nng_safe);
  nng_safe == true;
}
```



VIATRA-CEP

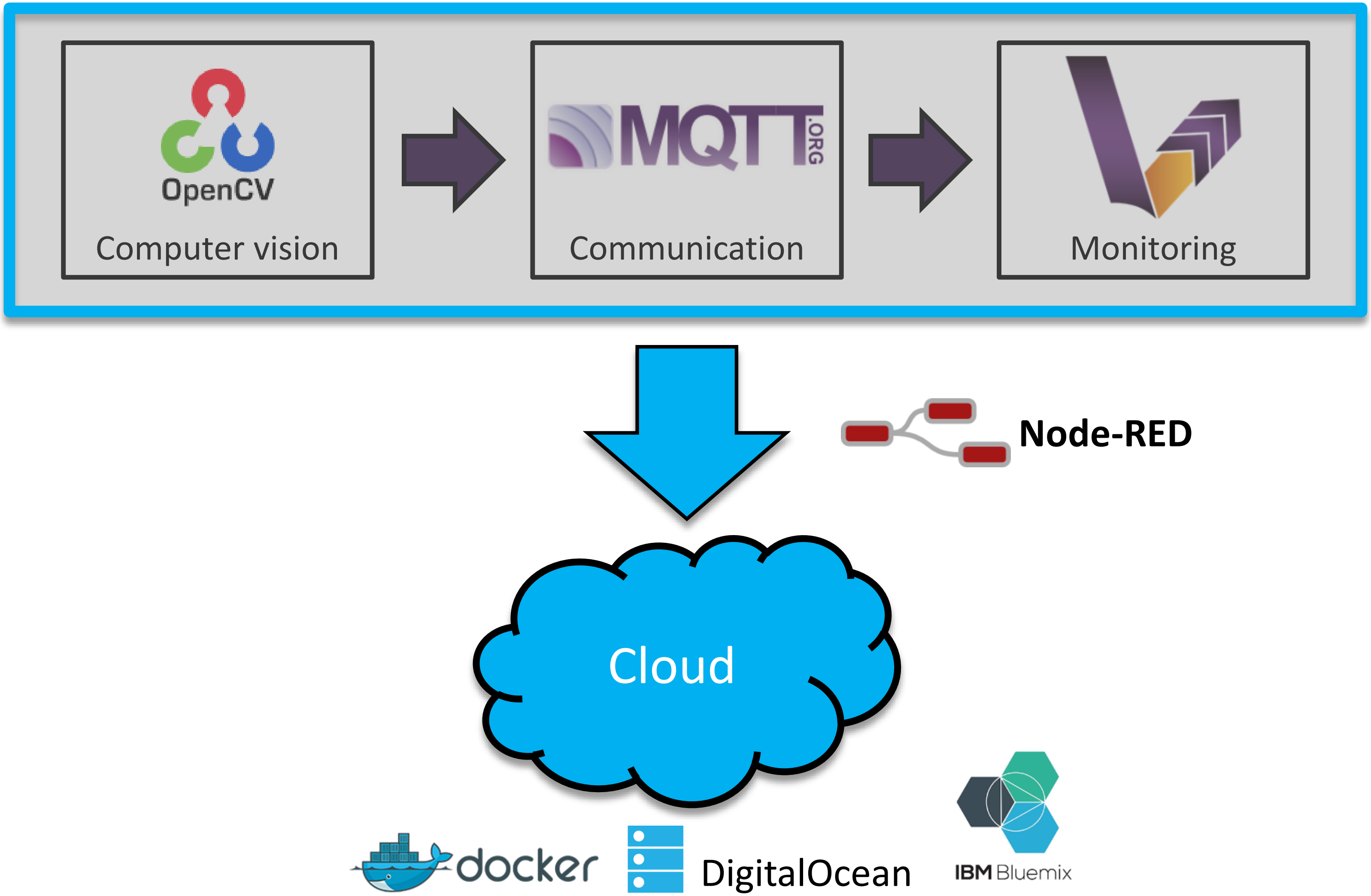
Monitoring and Control System

- Additional level of safety – high level monitoring



Monitoring and Control System

- Additional level of safety – high level monitoring



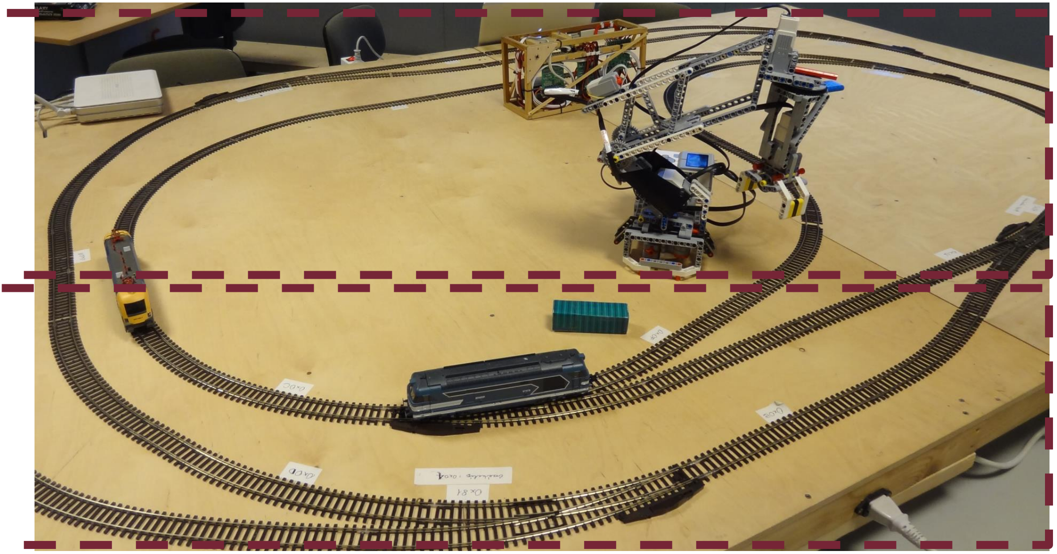
MoDeS3

SW Monitoring and Control System

HW

Robot system

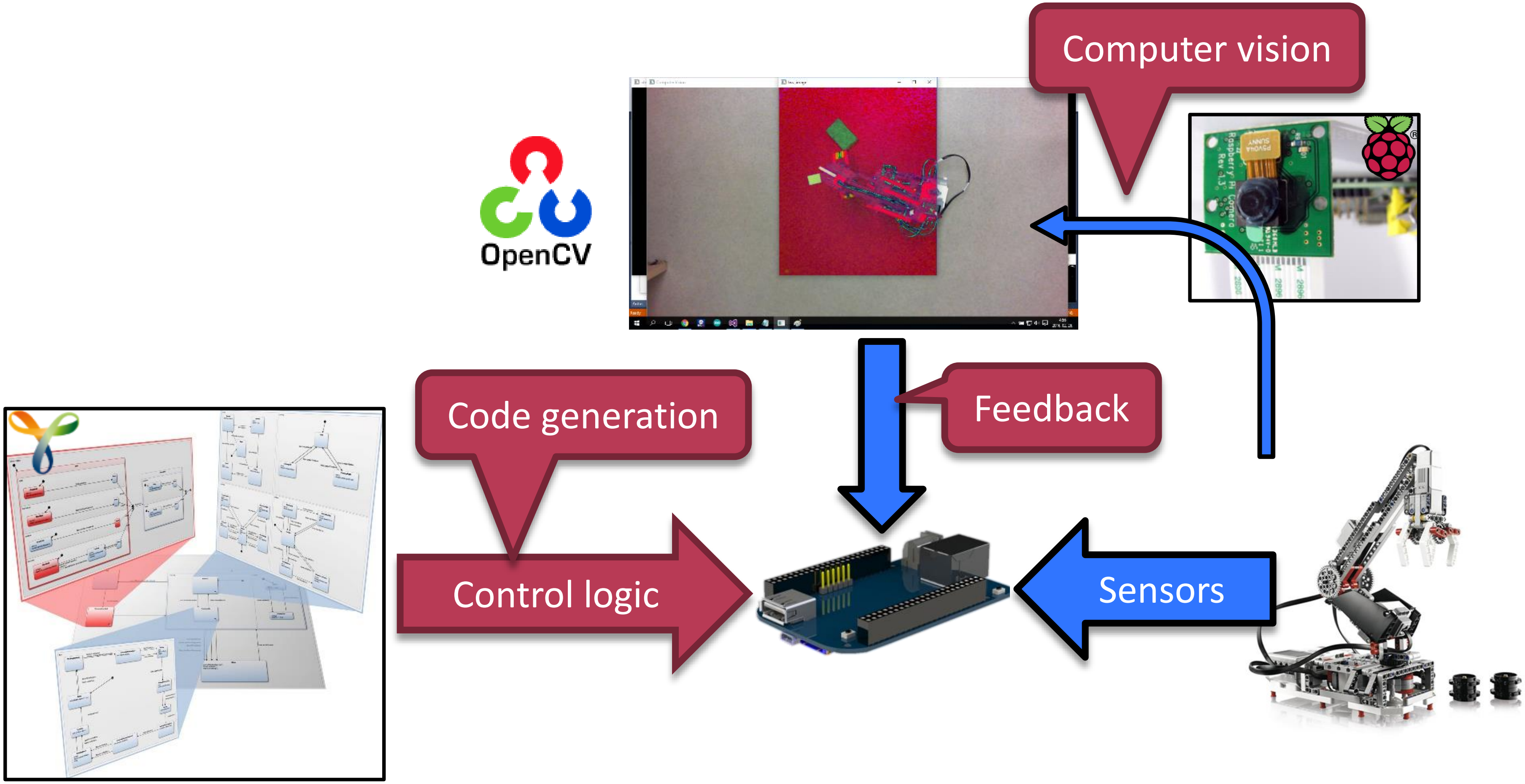
Railway system with
- Sensors
- Actuators



SW Distributed Safety Logic

Robot System

- Goal: Moving/removing objects from the trains
  - Place onto other train/place onto the ground



Computer vision

Code generation

Feedback

Control logic

Sensors

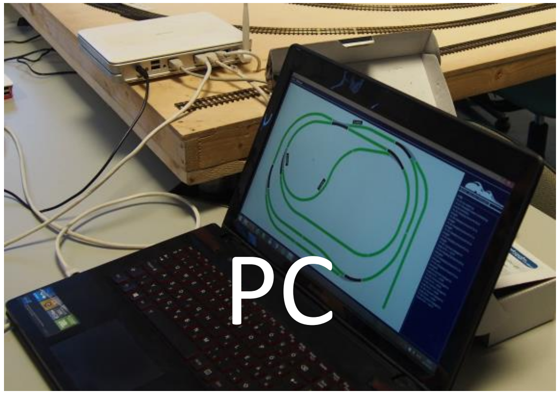
Heterogeneous platform

Cloud infrastructure

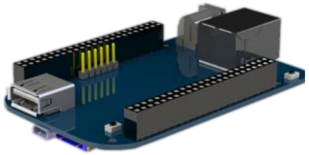
Cloud

Infinite resources

Personal computer



Application processor



Real-time unit

Fast response

Technologies

VIATRA-CEP

Mosquitto

Node-RED

YAKINDU
BUILT ON eclipse
itemis
open source
EMF-IncQuery
python
OpenCV
Maven
OpenModelica
MQTT.ORG
debian
Xtend
A language made for Java developers.
IBM Bluemix
beagleboard.org
Xtext
docker
paho

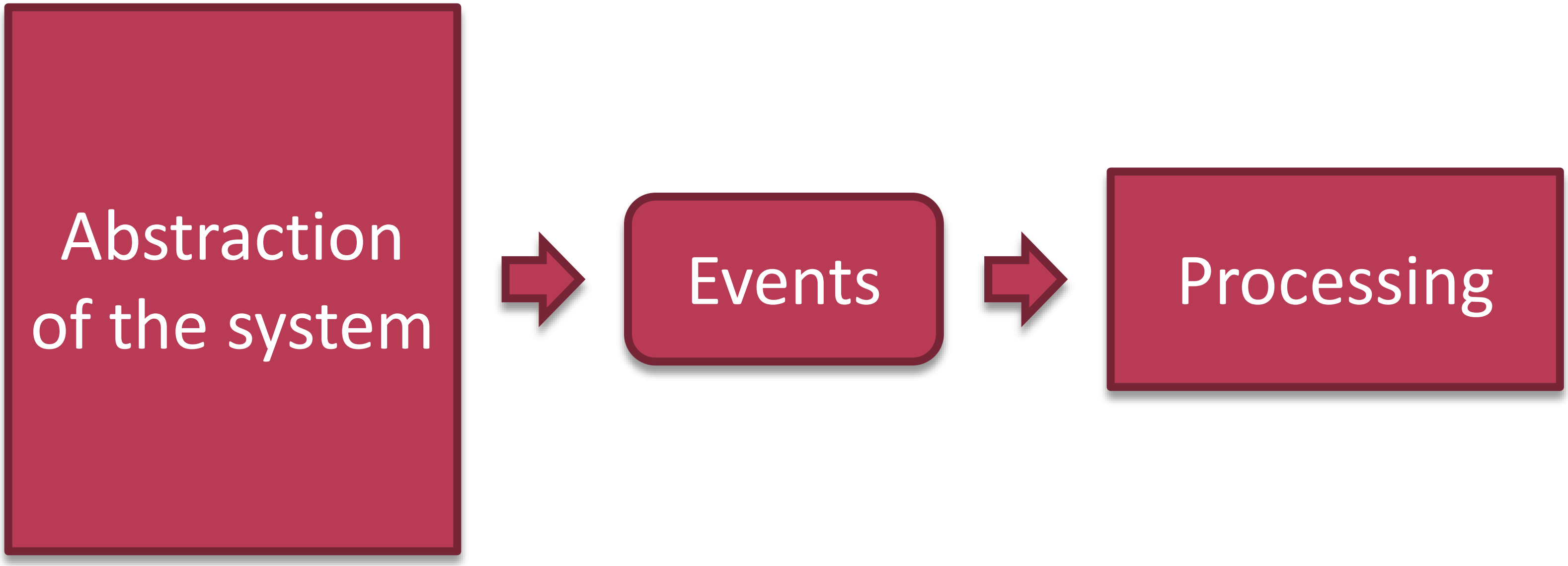
Summary

- Goal: case-study for smart CPS
- Combine various techniques from the domains of
  - Cyber-physical systems
  - Safety-critical systems

SYSTEM LEVEL MONITORING FRAMEWORK: VIATRA-CEP

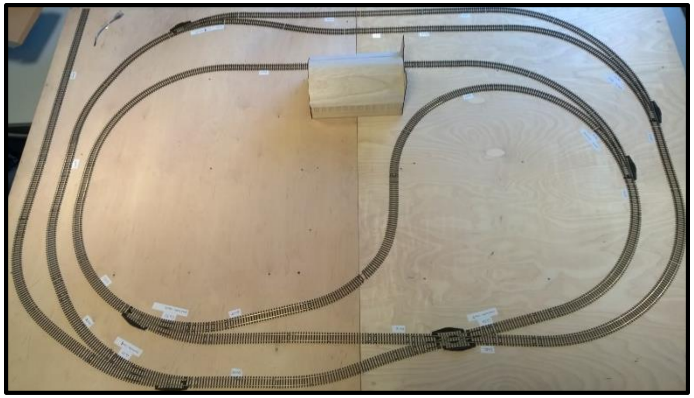
System Level Runtime Verification

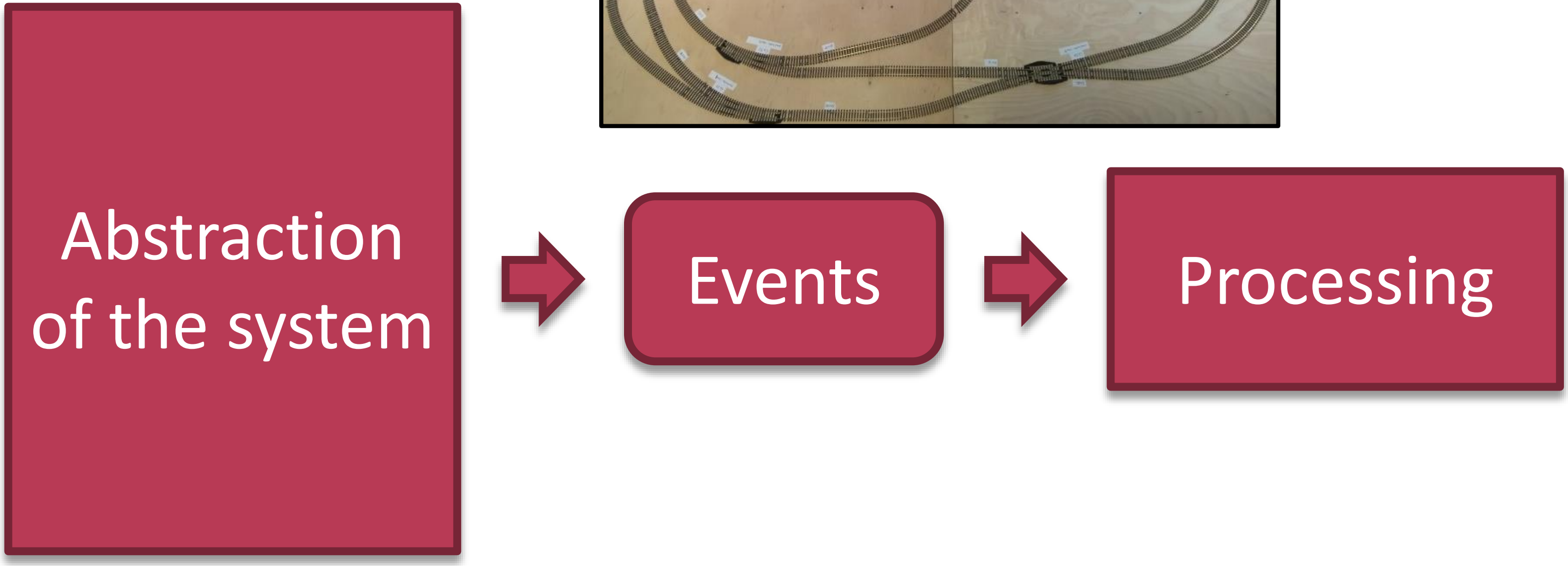
- VIATRA - CEP



System Level Runtime Verification

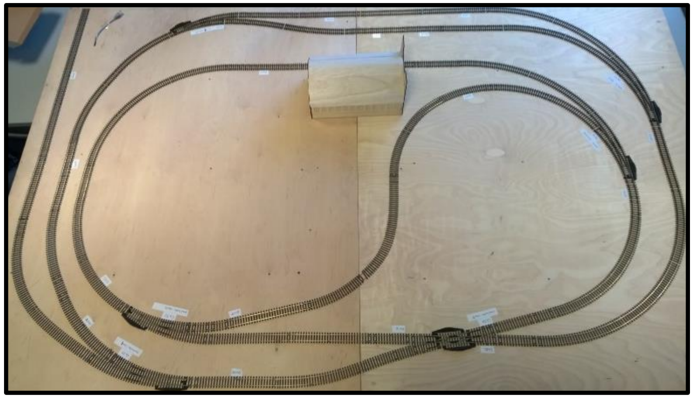
- VIATRA - CEP

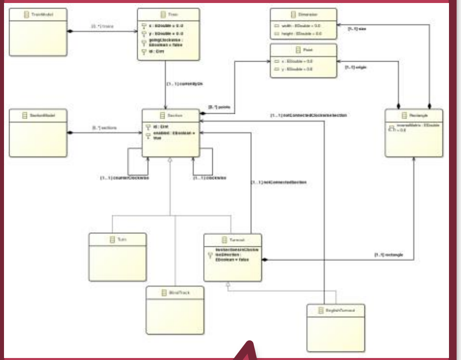




System Level Runtime Verification

- VIATRA - CEP





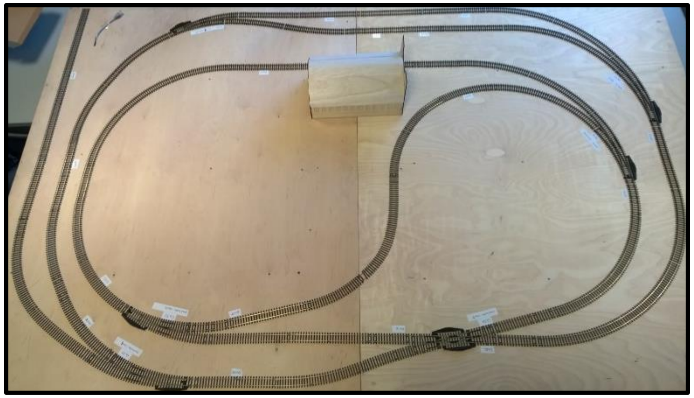
EMF metamodel:
- Elements and possible relations/connections

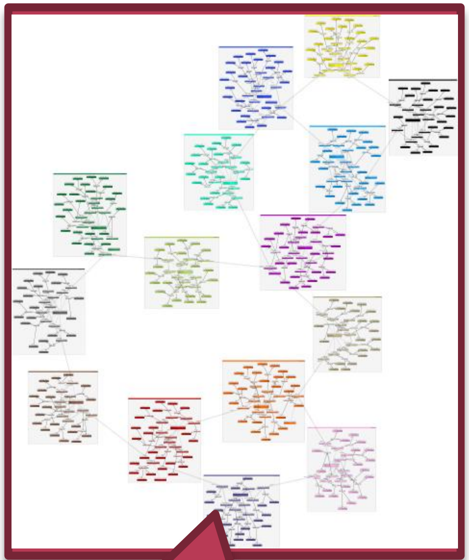
Events

Processing

System Level Runtime Verification

- VIATRA - CEP





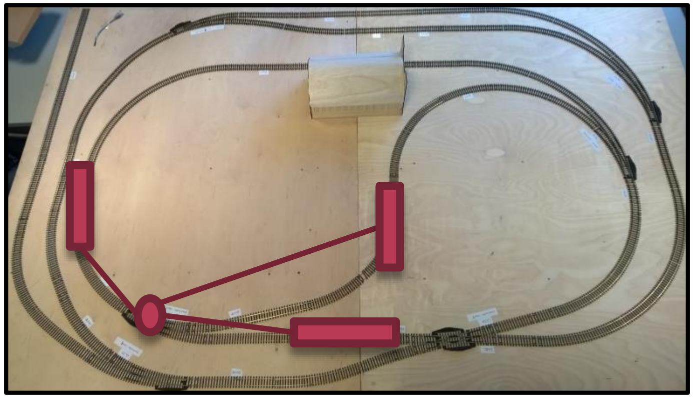
Instance model

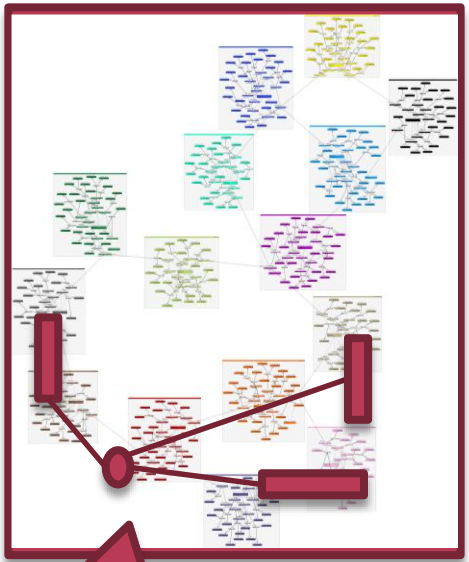
Events

Processing

System Level Runtime Verification

- VIATRA - CEP





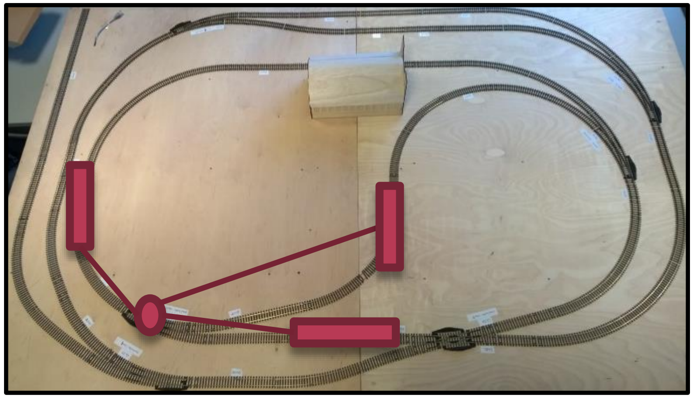
Graph pattern matching

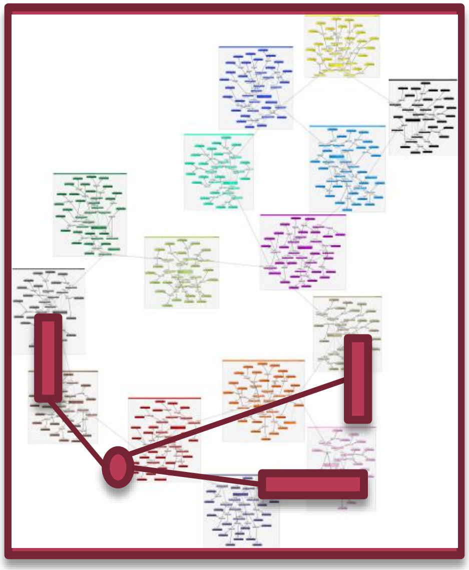
Events

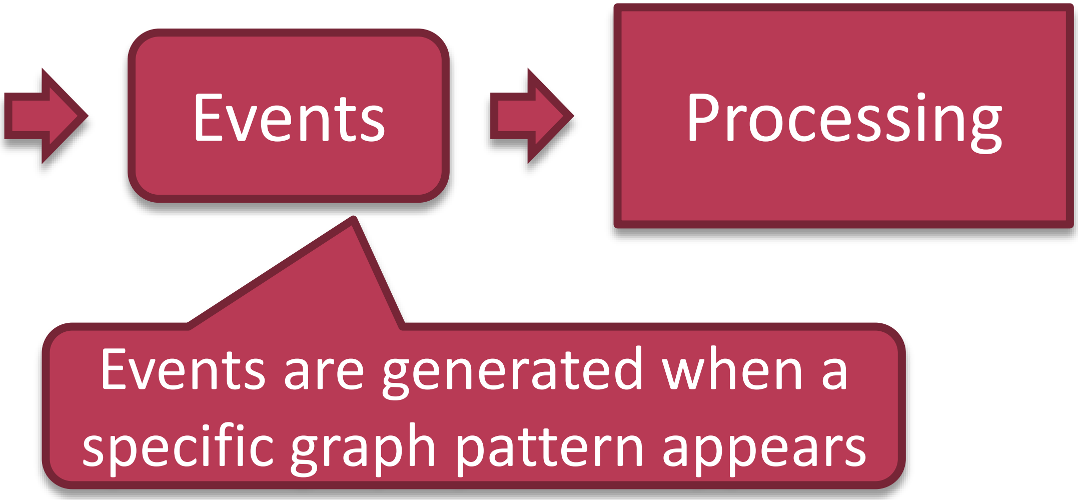
Processing

System Level Runtime Verification

- VIATRA - CEP







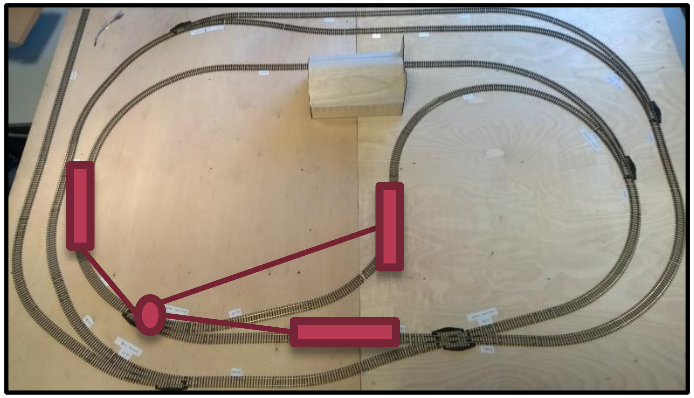
Events

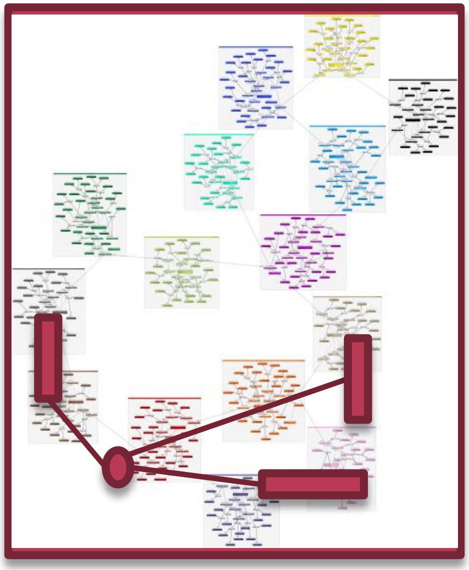
Processing

Events are generated when a specific graph pattern appears

System Level Runtime Verification

- VIATRA - CEP

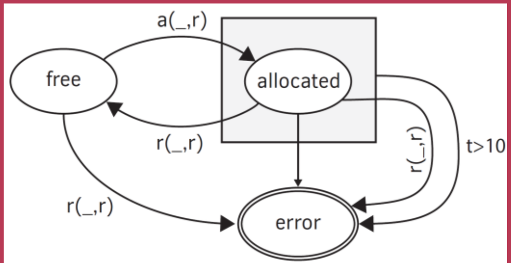




Events

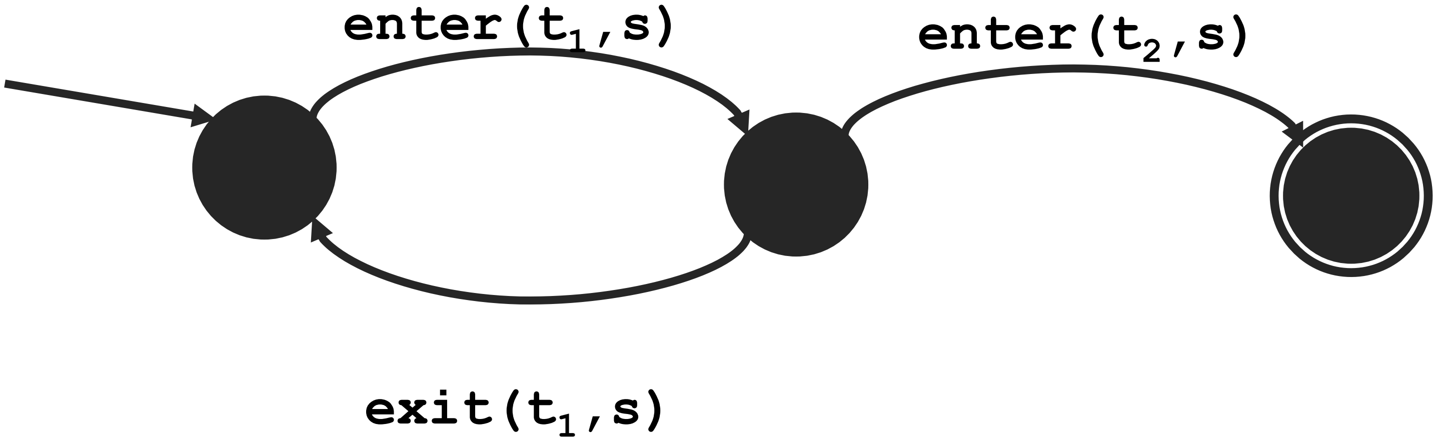
Processing

Automaton „consumes” the events



Investigation of languages

- Parametric Timed Regular Expression
- Parametric Timed Event Automaton
  - Based on Parametric Event Automaton
- Example:
  - Two trains should not enter the same section
  - **`enter(`t_1`,s)->NOT(exit(`t_1`,s)){*}->enter(`t_2`,s)`**

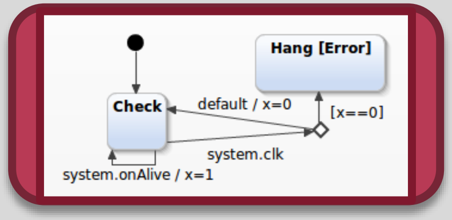


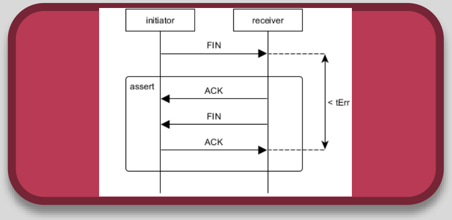
Investigation of languages

- Parametric Timed Regular Expression
- Parametric Timed Event Automaton
  - Based on Parametric Event Automaton
- Questions:
  - Timed-automaton determinization
    - Needed to run the monitor on embedded devices

Future Goals

Engineering languages





enter(t_1,s)->NOT
(exit(t_1,s)){*}
->enter(t_2,s)

Common formal intermediate representation

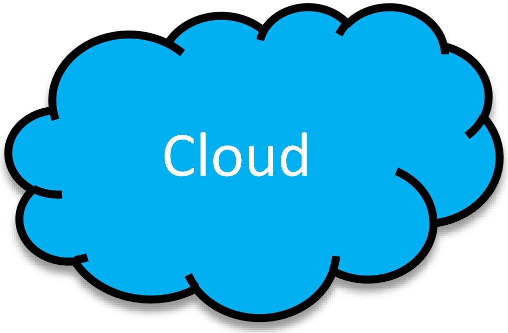
Embedded (C++) monitor

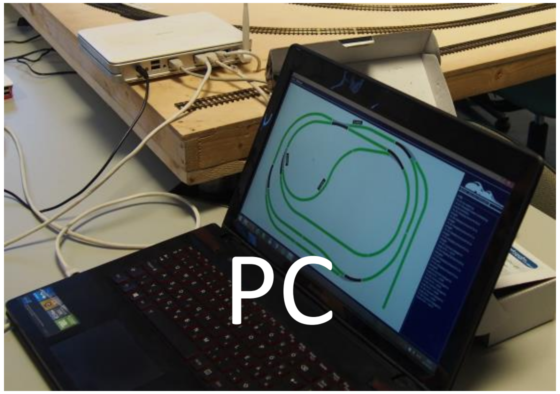
Rule System

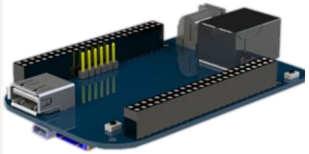
Runtime verification components

Automated deployment

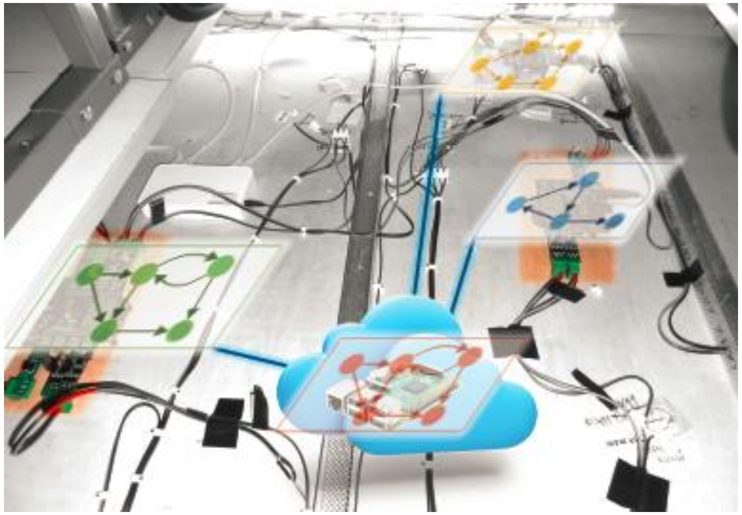
Platform model







Runtime specification
(high level language)



Summary

- CPS demonstrator: MoDeS3
- VIATRA – CEP: ongoing developments
  - Development of the automaton formalism
  - Determinization
  - Automatic deployment/monitor synthesis

Acknowledgements

- Application in research:
  - Fault Tolerant Systems Research Group
  - MTA-BME Research Group on Cyber-Physical Systems
  - Reconfigurable ROS-based Resilient Reasoning Robotic Cooperating Systems (R5-COP)
- Industrial sponsors:
  - IncQuery Labs Ltd.
  - Quanopt Ltd.
  - Ericsson