UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

7th Italian Workshop on Embedded Systems
(IWES 2022)

Characterizing monitoring solutions for real-time embedded applications using virtualization

Marcello Cinque, Luigi De Simone, Nicola Mazzocca, Daniele Ottaviano, Francesco Vitale

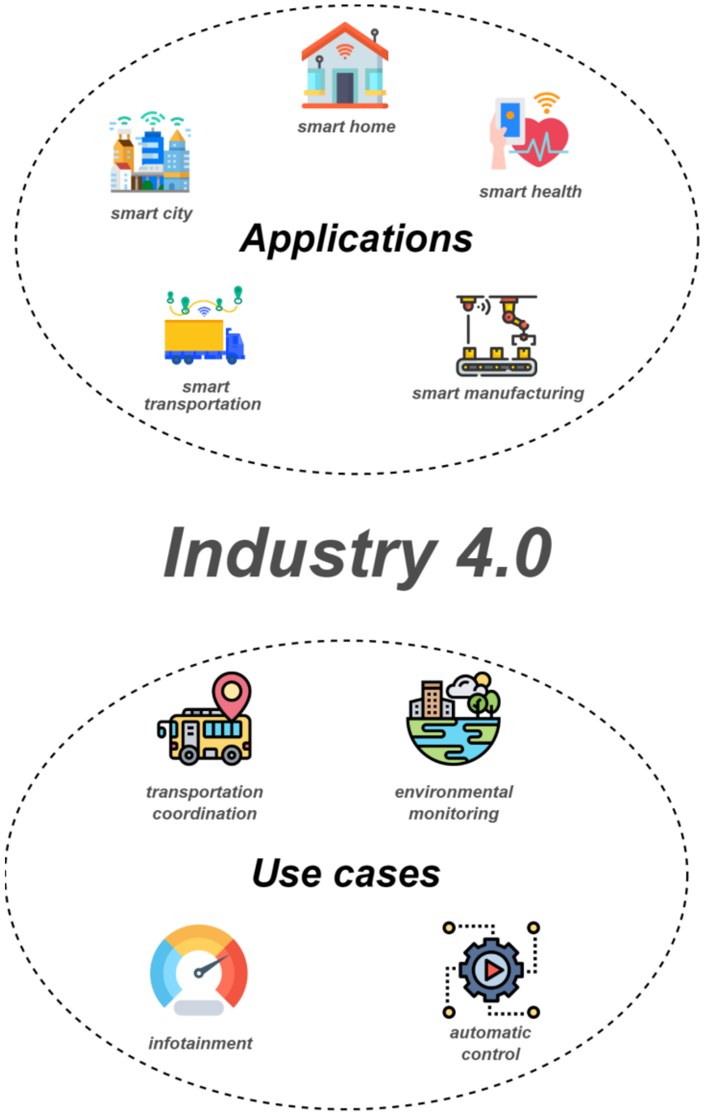
Index

- Introduction
    - Real-time industrial systems
    - Monitoring embedded applications
    - Mixed-criticality systems (MCSs)
    - Virtualizing MPSoCs for MCSs
    - Wrapping up
    - Research questions
- System development
    - Design process
    - Architectural scenarios
- Experimentation
    - Case-study
    - Goal
    - Testbed
    - Results

Real-time industrial systems

- Technological advances in embedded systems led to the development of **smart applications** for Industry 4.0 scenarios
    - Smart manufacturing
    - Smart cities
    - …

- These applications require deploying **real-time systems** managing both **critical and non-critical operations**
    - Control of industrial plants
    - Public transportation coordination and management
    - Infotainment management
    - …



Monitoring embedded applications

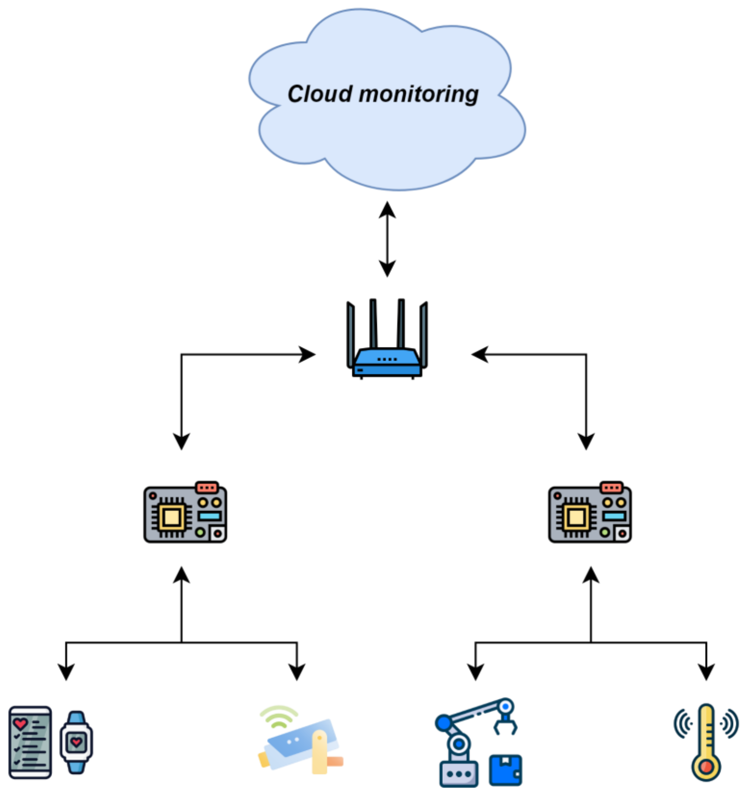
- **Monitoring** opens the opportunity for **anomaly detection** and application of **recovery actions**

- **Cloud-based monitoring** leads to **several disadvantages**
  - **Network partitioning**
  - **Detection latency**

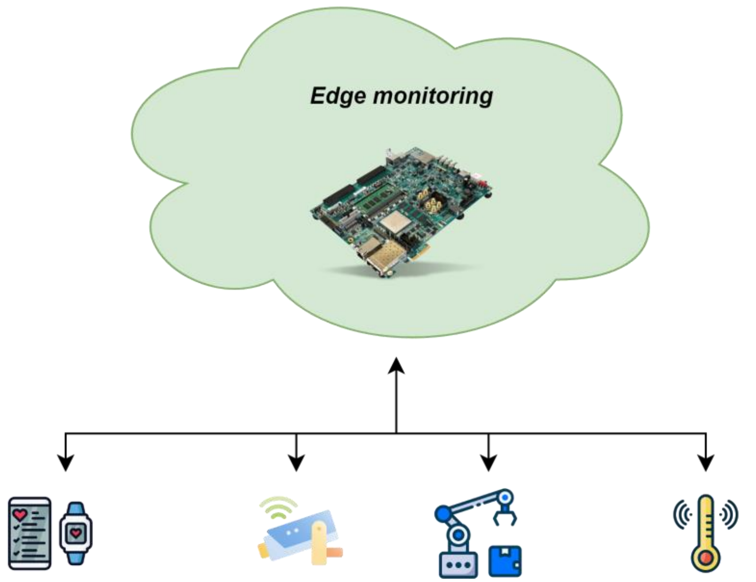
- A **paradigm shift** to **edge-based monitoring** on embedded platforms may be the solution
  - Existing literature lacks the evaluation of edge-based monitoring of real-time applications
  - **Can we meet non-functional requirements considering this paradigm shift?**

Monitoring embedded applications

Cloud-based monitoring

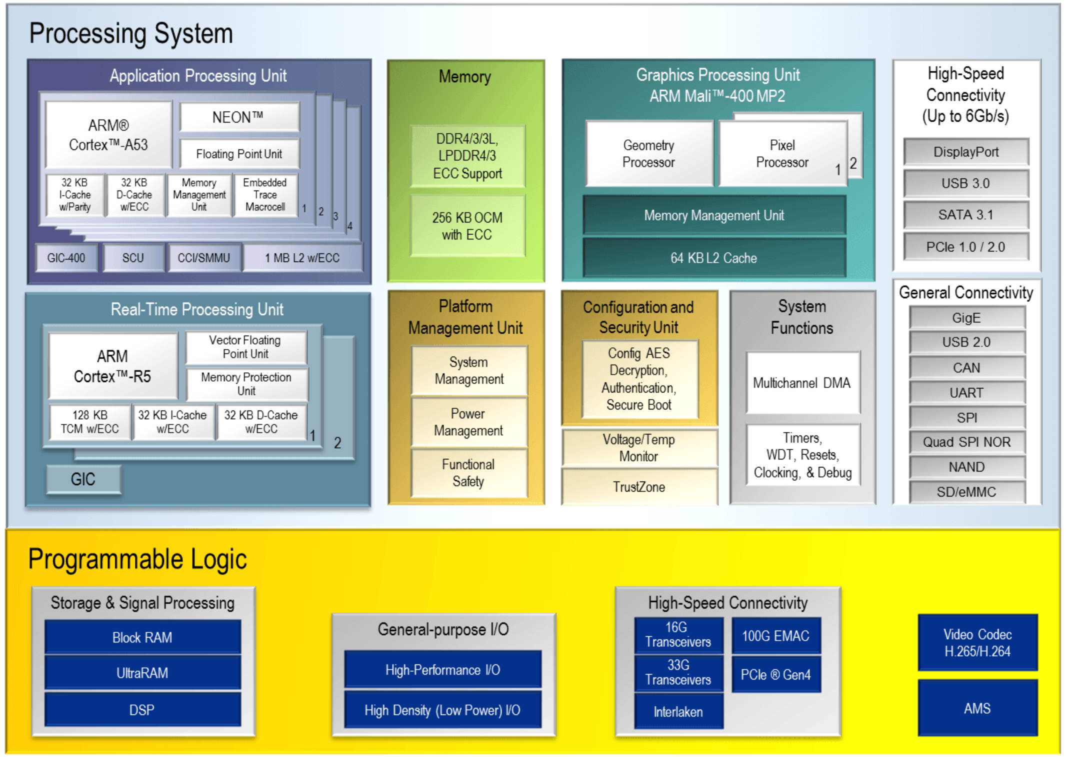


Edge-based monitoring



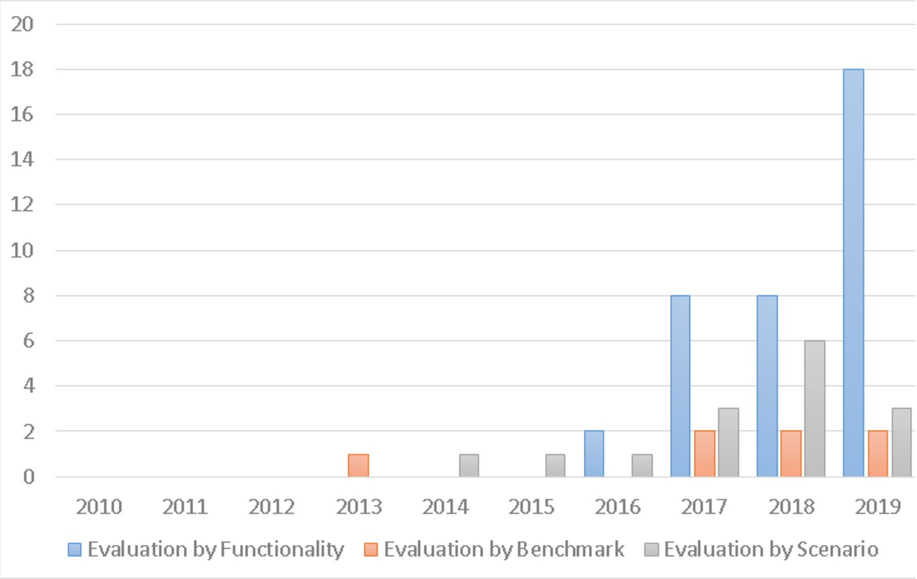
Mixed-criticality systems (MCSs)

- Developing industrial systems leads to several **non-functional requirements**
  - **Performance scalability**
  - **Dependability** attributes (safety, security, reliability)
  - **Timeliness**
  - **Mixed-criticality**
  - …
- **Multiprocessor System-on-Chips** (MPSoCs) properties allow deploying industrial systems meeting their non-functional requirements
  - **Multiprocessor parallel architectures**
  - **Dedicated accelerators** (GPU, FPGA, RPU)
  - **Hardware support for virtualization**
  - …



Virtualizing MPSoCs for MCSs

- Virtualization is considered the most affordable solution to problems related to **space, weight, power, and cost (SWaP-C)**
- Virtualization is used to deal with MPSoCs and IoT issues:
  - **Device heterogeneity**
  - **Environment variety**
  - **Management complexity**
  - **Scalability**
  - **Dependability**
- Hypervisors in embedded real-time systems can guarantee:
  - **Hardware consolidation**
  - **Legacy software migration**
  - **Reduced development cost and time to market**
  - **Mixed-criticality (e.g., RTOS and modern commodity OSes)**

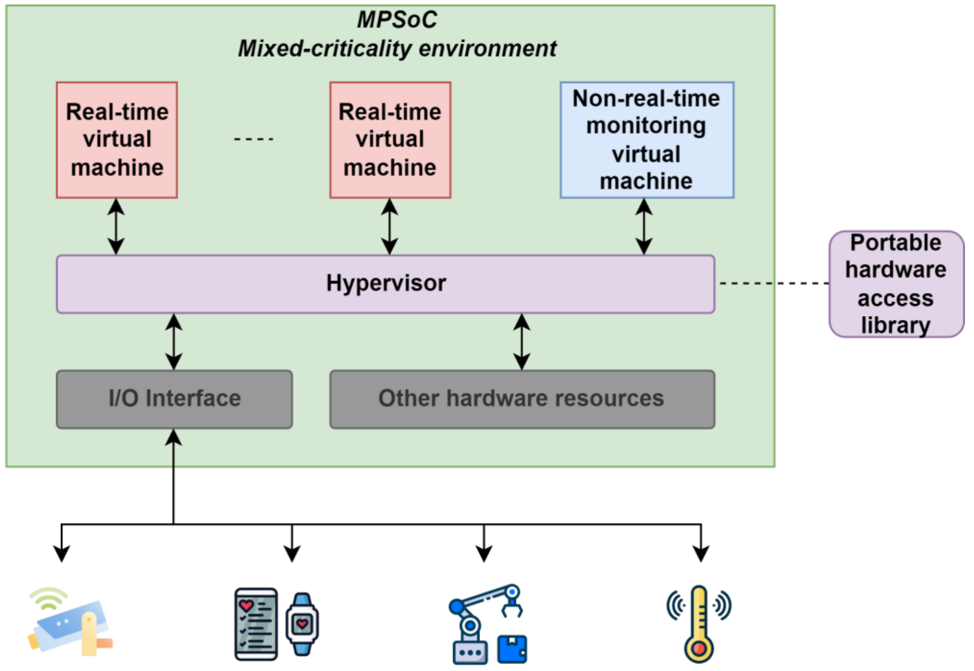


Recently, the number of papers leveraging on virtualization to deploy IoT services is continuously growing

Virtualizing MPSoCs for MCSs

Portability

- The **virtualization layer** providing portable hardware access **must not impact real-time guarantees**



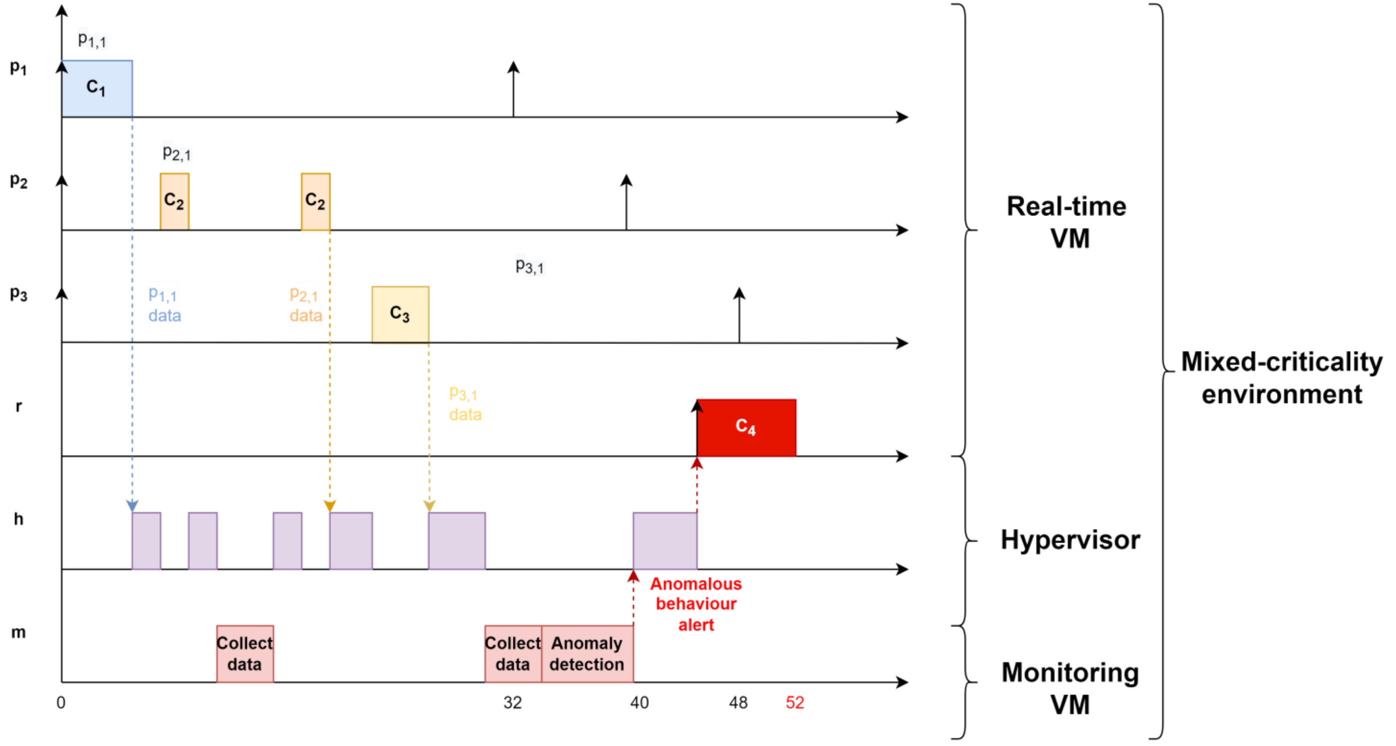
Virtualizing MPSoCs for MCSs

Predictability

- **Edge-based monitoring reduces the time required for detecting anomalous** behaviour, but may introduce **non-predictable interferences** due to the orchestration of the environment by the hypervisor

$$TS_{RT\_VM} = \{P, A\}$$
$$P = \{p_1, p_2, p_3\}$$
$$T(p_1) = 32, T(p_2) = 40, T(p_3) = 48$$
$$A = \{r\}$$

$$TS_{M\_VM} = \{m\}$$

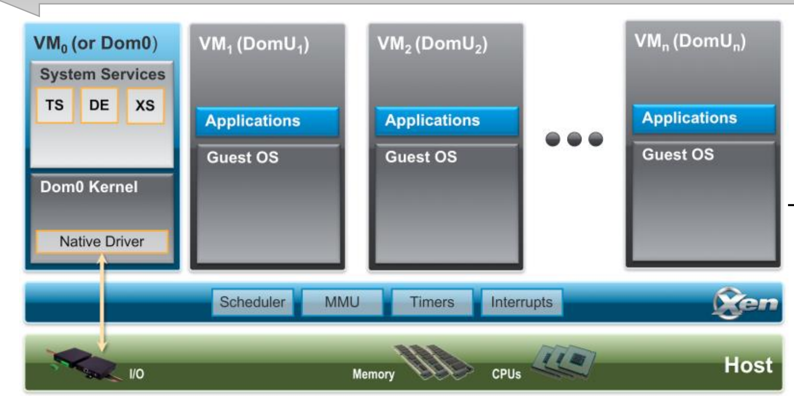


Virtualizing MPSoCs for MCSs

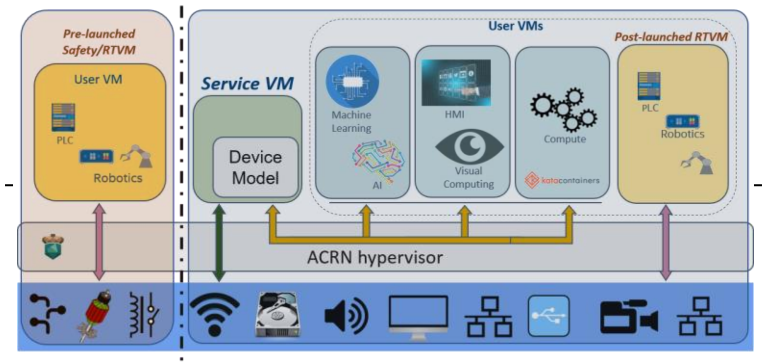
Hypervisor

- Choosing the right hypervisor can have limited impact on predictability while guaranteeing applications portability
  - Isolation mechanisms
  - Inter-VM communication
  - I/O access

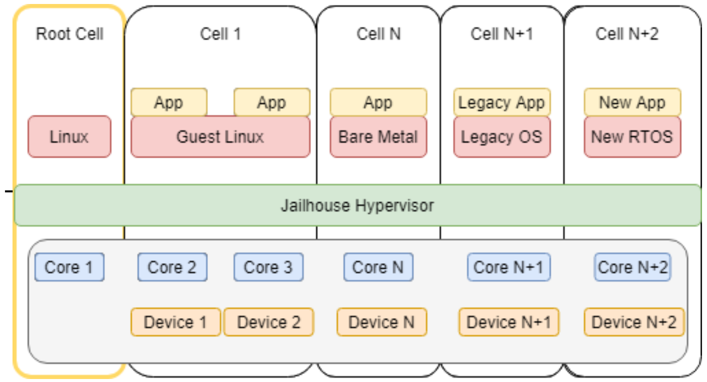
flexibility



XEN Hypervisor



ACRN Hypervisor



Jailhouse Hypervisor

Isolation of resources

Wrapping up

- **Virtualized MPSoCs can integrate monitoring and application-specific software leading to a fault-tolerant design**
  - The resulting MCS has several **advantages** and **drawbacks**

<table>
<tr><th>Advantage</th><th>Drawback</th><th>Solution</th></tr>
<tr><td>Portability</td><td rowspan="2">Predictability issues due to resource contention</td><td rowspan="2">Apply sound development methodologies and architectural design choices</td></tr>
<tr><td>Lower detection latency</td></tr>
<tr><td>Independent applications certification</td><td>Faulty hypervisors</td><td>Use certified hypervisors</td></tr>
<tr><td>No network partitioning</td><td>Resource constraints</td><td>Select appropriate MPSoCs for the applications to deploy</td></tr>
</table>

Research questions

- We came up with the following research question:

RQ: Can we integrate monitoring for real-time embedded applications through MPSoCs meeting their non-functional requirements?

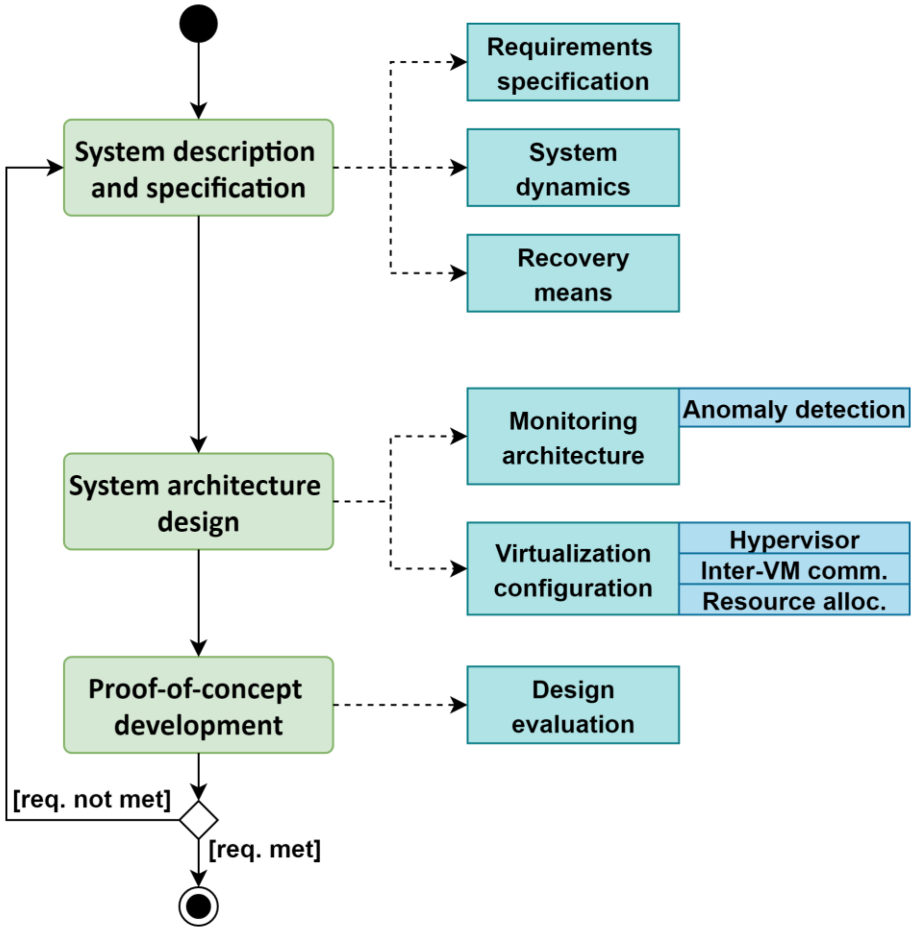
- We split it into:

RQ1: What design process should be followed for integrating monitoring in real-time embedded applications?

RQ2: Which architectural scenarios lead to the best performance and predictability trade-off in virtualized MPSoCs?

Design process

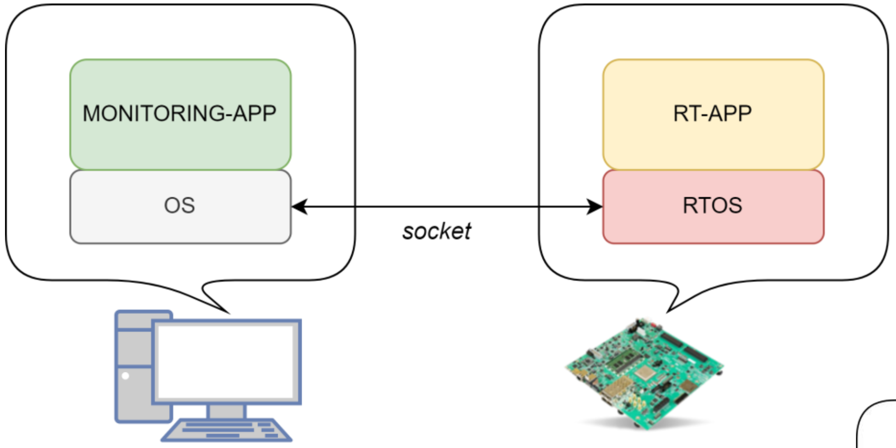
- We propose a **design process** to address RQ1
- The process is split in several activities
  - **System description and specification**
    - **Requirements specification**
    - **System dynamics**
    - **Recovery means**
  - **System architecture design**
    - **Monitoring architecture**
      - Anomaly detection
    - **Virtualization configuration**
      - **Hypervisor**
      - **Inter-VM communication**
      - **Resource allocation**
  - Proof-of-concept development
    - **Design evaluation**



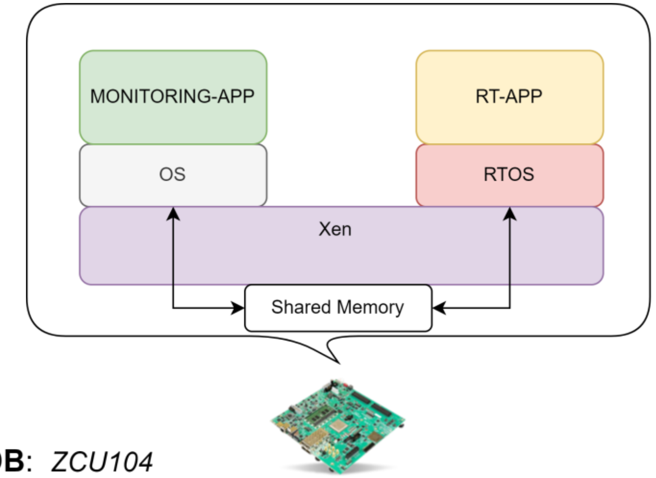
Architectural scenarios

- An architectural scenario is characterized by **several factors**
- Development Board (**DB**)
- Monitoring Deployment (**MD**)
- Virtualization (**V**)
  - Hypervisor (**H**)
  - Virtual machines scheduling (**SCHED**)
- Communication Technique (**CT**)
  - Socket (**SCKT**)
  - Message Queue (**MQ**)
  - Shared Memory (**SHM**)

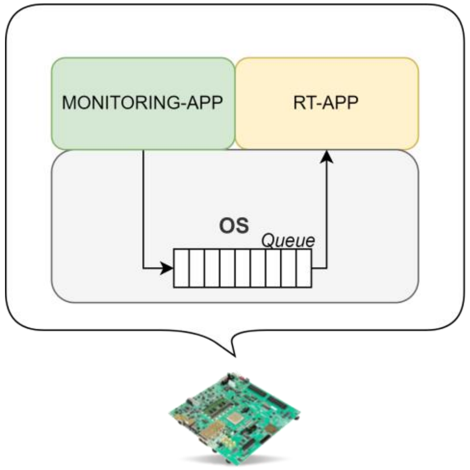
Architectural scenarios (2/3)



- **DB**: *ZCU104*
- **MD**: *OFF_BOARD*
- **V**:
- **V_ON**: *NO*
- **CT**: *SCKT*

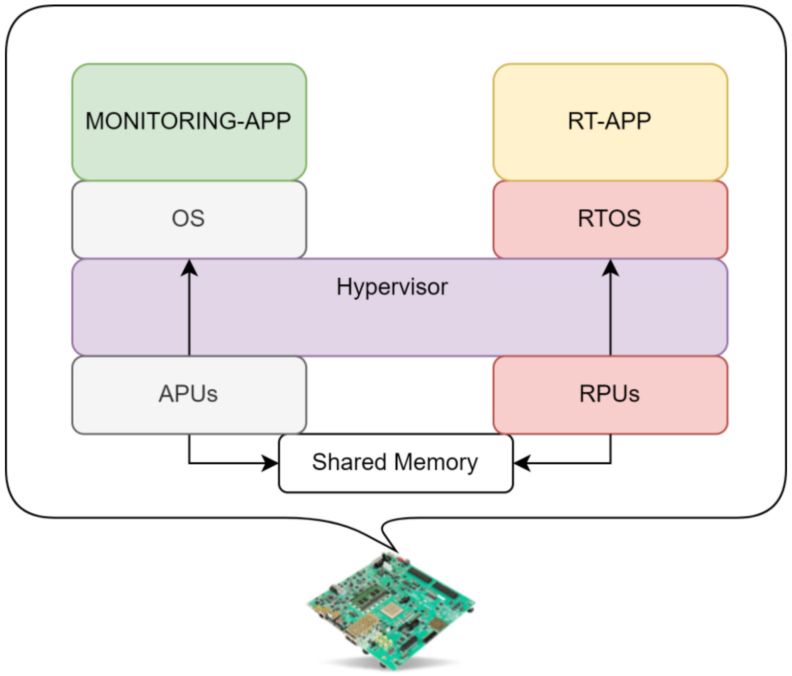


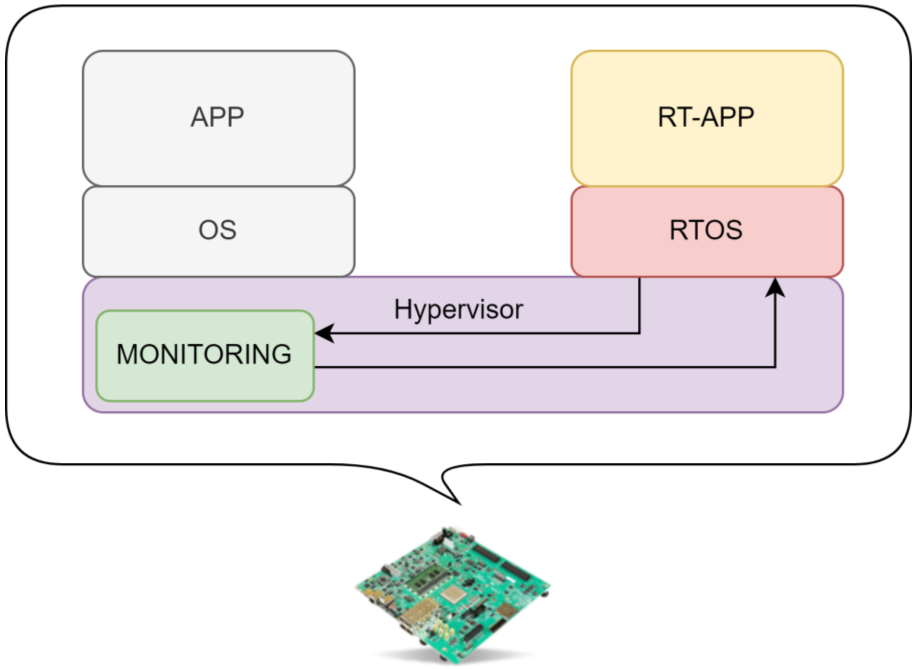
- **DB**: *ZCU104*
- **MD**: *ON_BOARD*
- **V**:
- **V_ON**: *YES*
- **H**: *Xen*
- **SCHED**: *RTDS*
- **CT**: *SHM*



- **DB**: *ZCU104*
- **MD**: *ON_BOARD*
- **V**:
- **V_ON**: *NO*
- **CT**: *MQ*

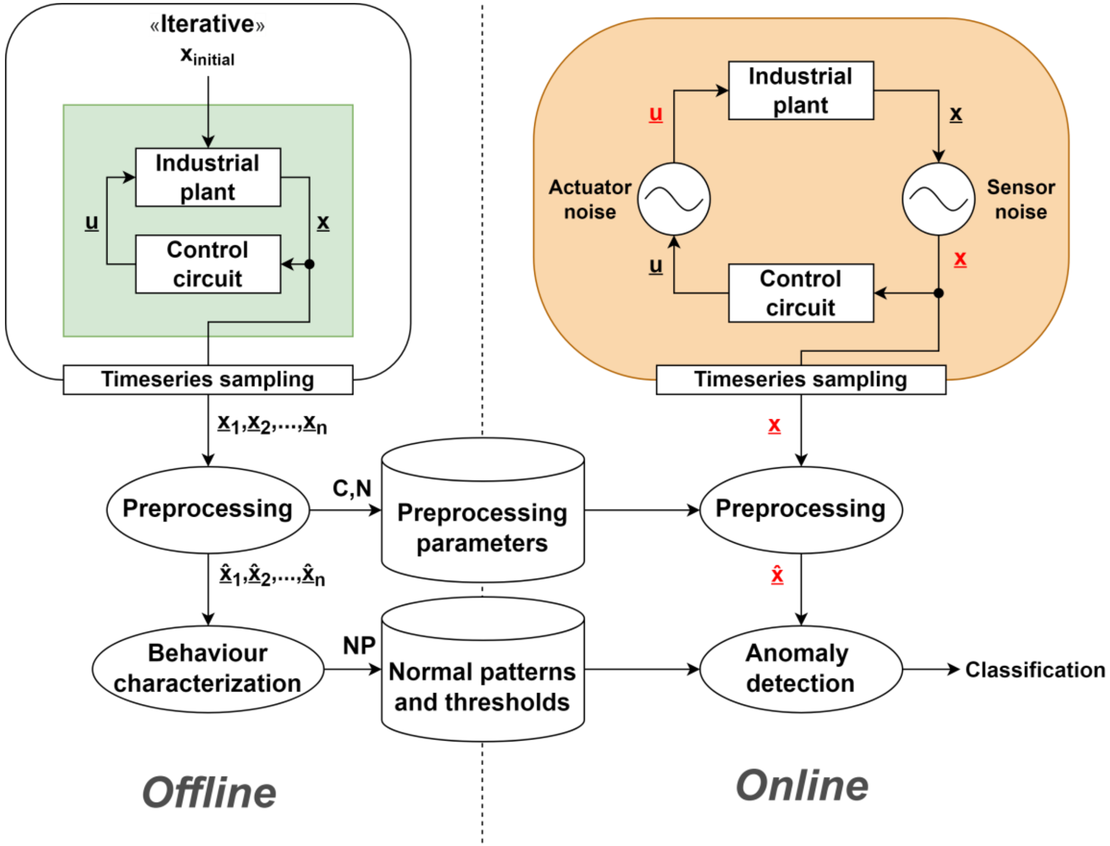
Architectural scenarios (3/3)





Case study

- Our target case study is the **control of critical industrial plants**, a typical Industry 4.0 use
- **Developing robust control require modelling external noise and uncertain plant dynamics**
- **Monitoring plant parameters may highlight anomalous behaviour** due to unmodelled noise and disturbance
- We propose the **integration of monitoring routines** for detection of anomalous plant behaviour **through edge-based monitoring**



Goal

- The **goal** of our experimentation **is addressing RQ2**, i.e., **showing different architectural scenarios lead to changes in detection performance and system predictability**

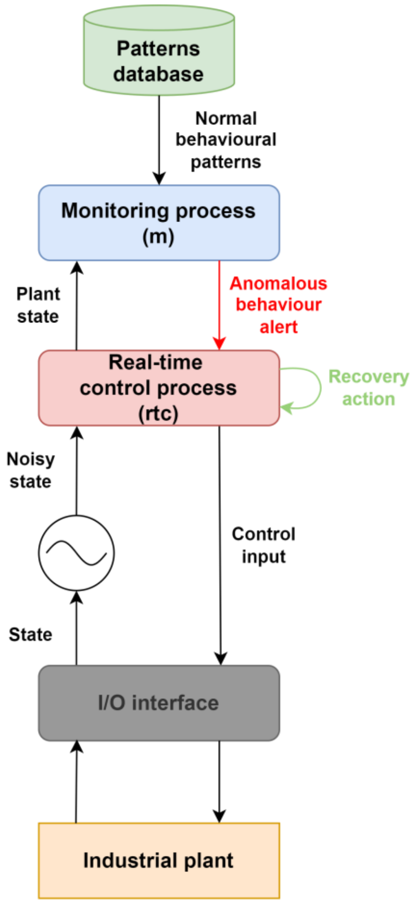
- Our experiments are designed around factors and response variables which made us split RQ2 in two further questions:

RQ2_1: Do different architectural scenarios influence system predictability?

RQ2_2: Which architectural scenario offers the best detection performance-system predictability trade-off?

Testbed

Logical architecture



Factors

- Sampling Frequency (SF) *(fixed)*
- Window Size (WS) *(fixed)*
- Architectural Scenario x (ASx)

Response variables

- $avg_{m,ASx}$
- $avg_{rtc,ASx}$
- $dev_{rtc,ASx}$

We establish as baseline the response variables linked to the real-time control process on a raw reference real-time embedded environment (we label this architectural scenario as AS0)

- $avg_{rtc,AS0}$
- $dev_{rtc,AS0}$

Thank you!

Any questions?