Kry10 OS Software Safety Manual

Ihor Kuz - Kry10

ihor@kry10.com

Overview

Safety Overview

- Functional Safety
- Certification
- Safety Manual and SEooC

Kry10 OS (KOS)

Safety Process

- Making a System Fault Tolerant
- Maintain Safety while Handling failure
- Steps to transform a system
- Using KOS features

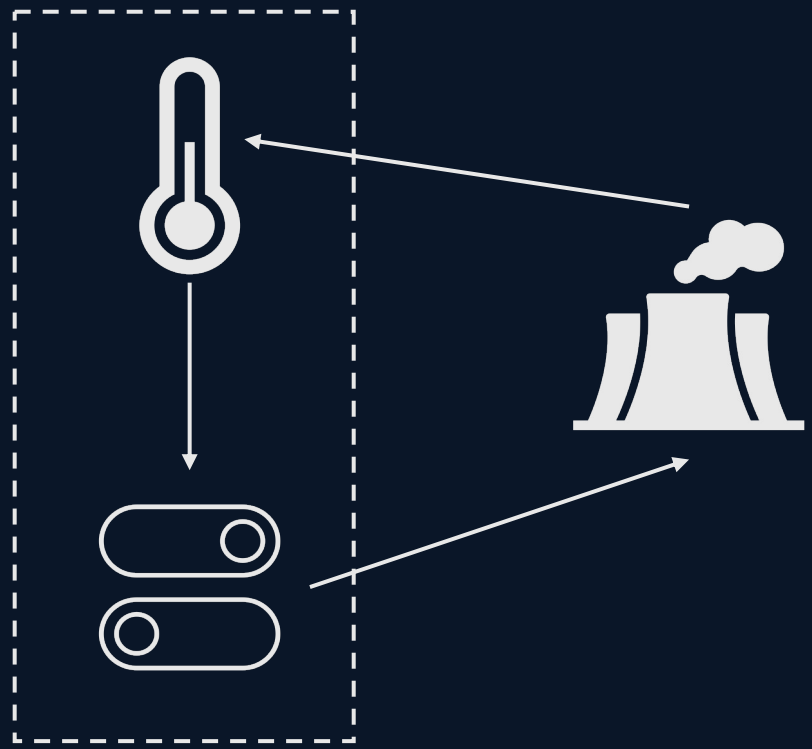
Functional Safety

Safety
- System is free from unacceptable risk of physical injury or of damage to health (direct or indirect)

Functional Safety
- Safety provided by dedicated active (sub)systems

Safety-Critical functions
- Must work correctly for the safety of the system

Quality-Managed (QM) functions
- Regular functionality
- Do not affect the safety of the system, but may be related



Example: heater safety switch

Certification

Safety sub-system must
- Work correctly
- Have Integrity
- Fail in safe/predictable way

Certification
- Confidence in reliability
- Guidelines for how to achieve it

Certification levels
- Level of Rigour, Confidence
- Mixed criticality - Challenge

Whole system certification
- System goals
- Operational conditions,
- Architecture

IEC 61508
- Functional Safety of electrical, electronic and programmable electronic (E/E/PE)
- Safety Integrity Level (SIL) 1-4

Industry-specific specialisations
- ISO 26262: automotive
  - QM, ASIL A-D
- DO-178C: aviation
  - Design assurance level (DAL) E-A
- ISO 13849: machinery
  - Performance levels (PL) a-e
- IEC 60601/62304: medical
  - Class A-C

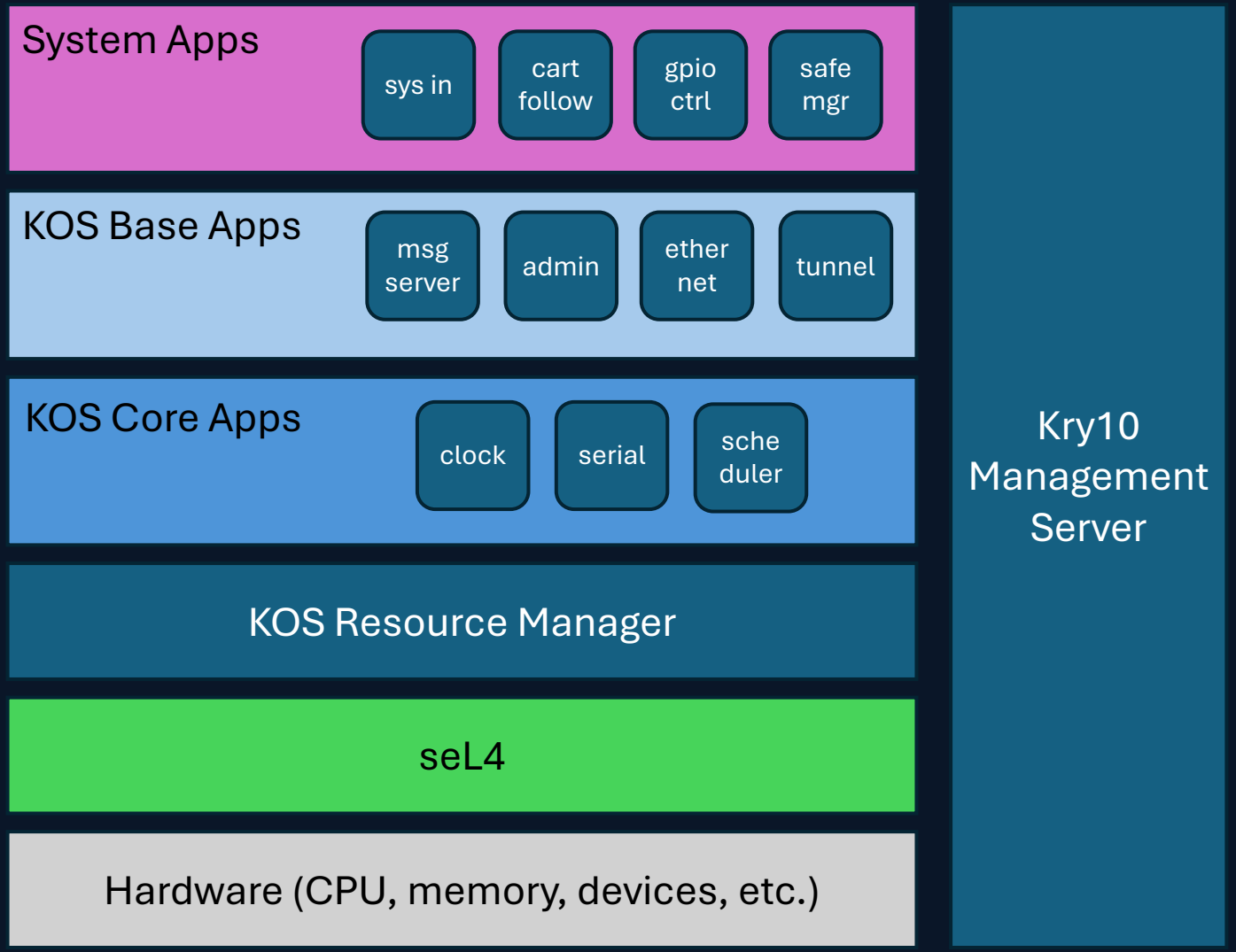
Safety Manual and SEooC

- Safety Manual
  - Instructions for safe use
  - Contract for how to maintain functional safety
  - Context assumptions
- Certification
  - Check that safety manual procedures followed
- Integration
  - Check that system context matches safety manual assumptions

- Safety Element out of Context (SEooC)
  - Component
    - hardware or software
    - for safety critical system
  - ***without*** specific final target
- Examples
  - Microcontroller
  - Operating system
  - Subsystem (e.g. brake control)
- SEooC Safety Manual
  - How to install and configure within a final product

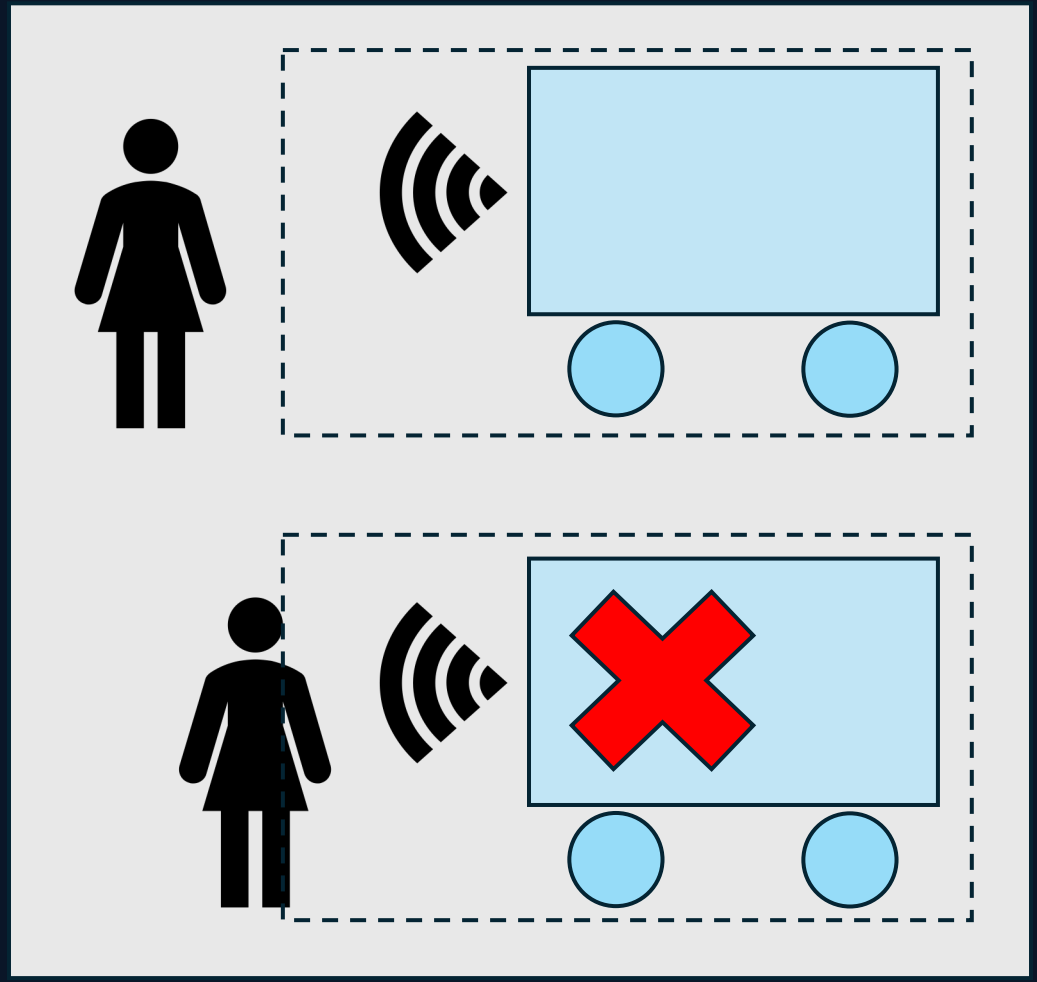
Kry10 OS (KOS)

- seL4-based OS platform
  - Static
  - Dynamic
- Manifest
- Resource Manager (root)
- Apps
  - Core apps
  - Base apps (Poukai)
- Communication
  - Channels
  - Msg server
- Management Server



Example System: Follow Cart

- Low-speed robot
  - Follows a human at a safe distance
  - Hazardous in close proximity
- Safety requirements
  - **Avoid Contact**
  - Emergency stop if person detected in safety-critical zone



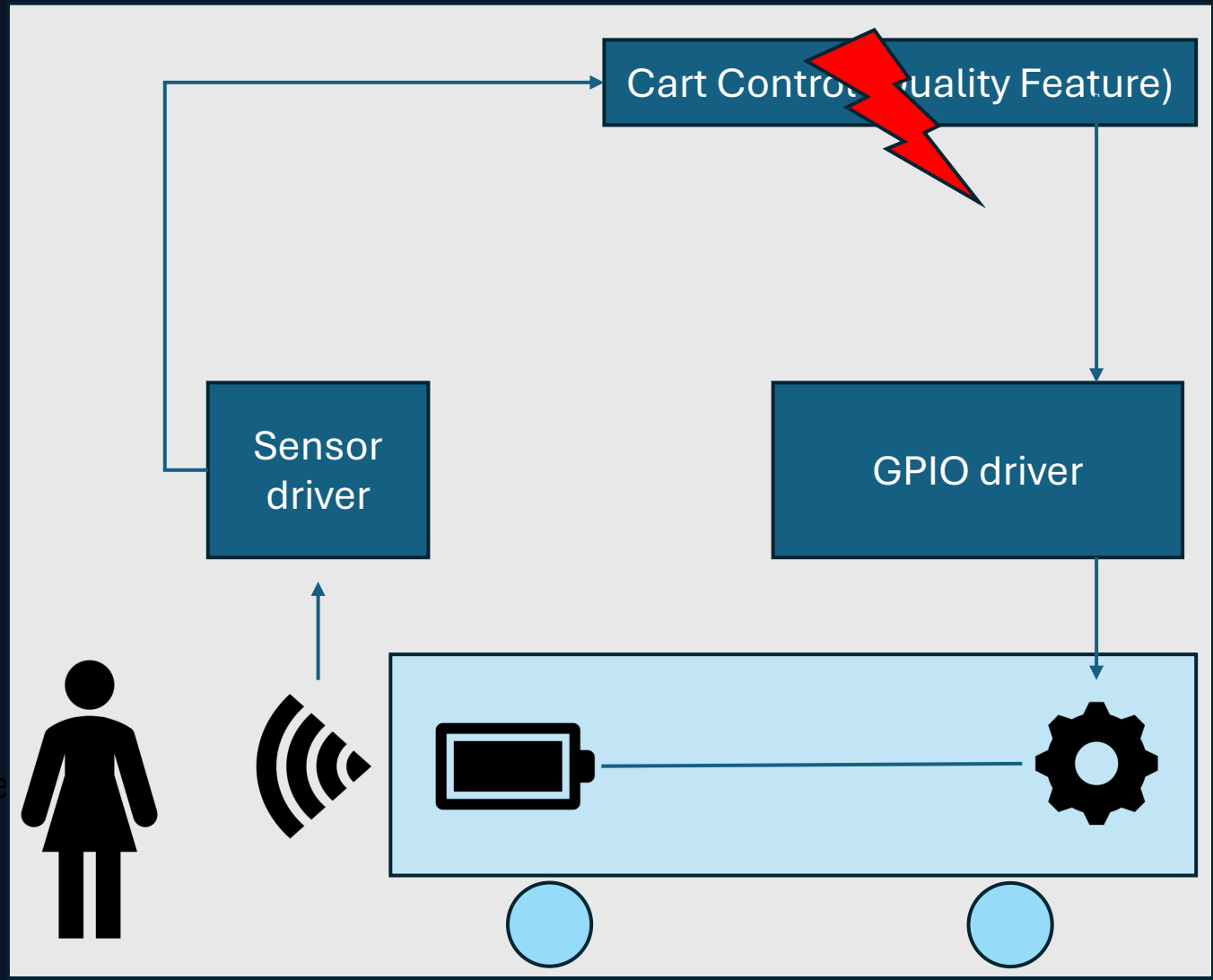
Safety Process Overview

- Define Base System
  - Quality Managed path
  - Safety Critical path
- Failure Analysis
  - Identify failure modes in safety critical path
  - Strategy to mitigate them
  - Fault model – set of faults guarded against
    - E.g. computational, memory, corruption, network, interference
- Goal
  - Internal failure does not result in safety failure
  - Maintain availability (normal functioning) as much as possible.

- Failures to Consider
  - Process failure
  - Control flow failure
  - Output failure
  - Input failure
  - Software Platform Failure
  - Hardware Failure
- Mechanisms
  - Prevention vs Detection vs Recovery
  - Freedom from Interference
  - Redundancy
  - Error detection (e.g. CRC) correction (e.g. ECC)
  - Monitoring: Timeout, Event
  - Restart: Component, System
  - Watchdog

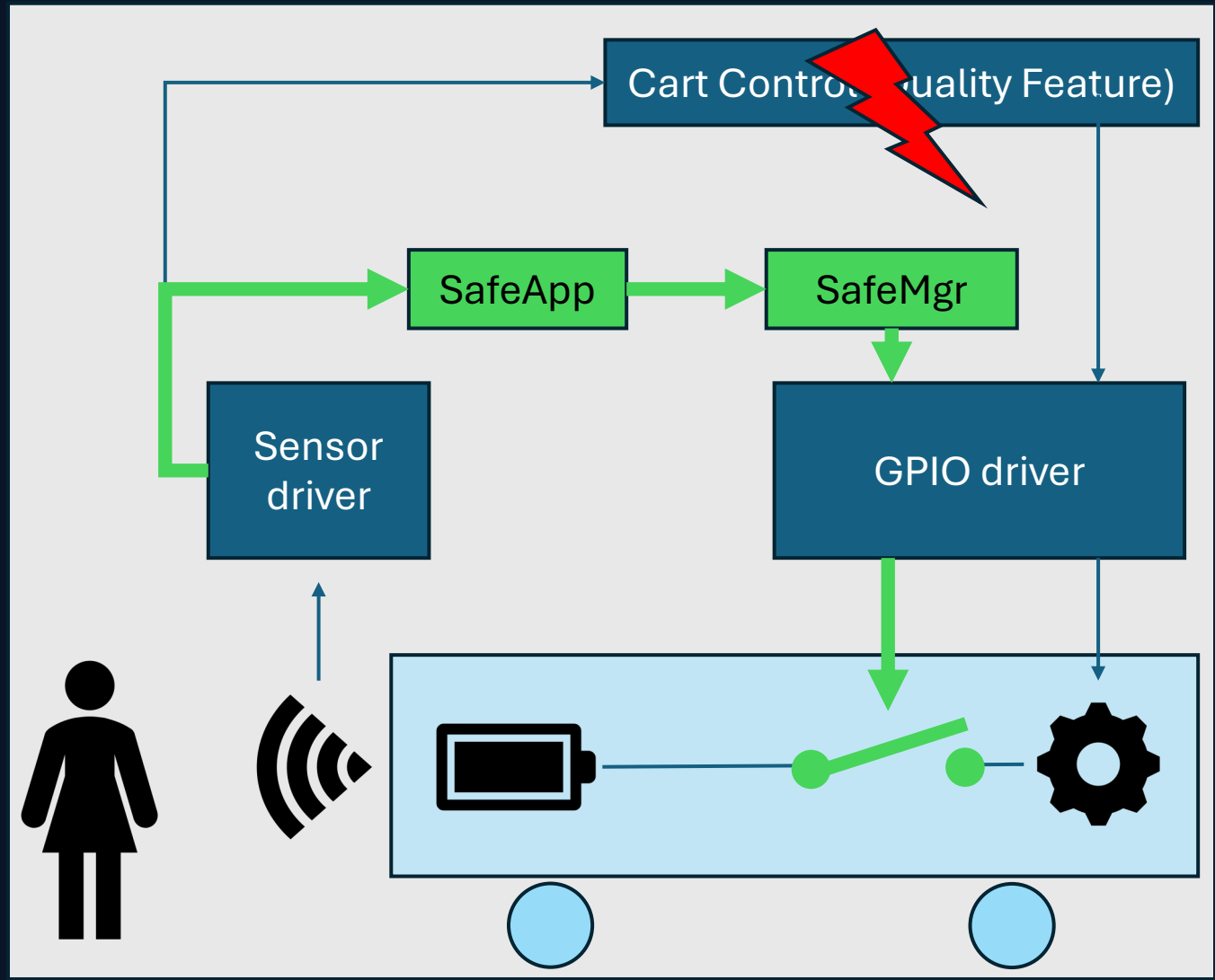
Base System with Safety Critical Path

- Base Functionality
  - Input
    - Distance sensor
  - QM functionality
    - Maintain target distance
    - Distance -> speed up/slow down
  - Output
    - Motor control
- Safety Functionality
  - Is person within critical zone?
    - In case QM functionality fails
  - Separate safety-critical path
    - SafeApp: calculate distance
    - SafeMgr: determine if distance too close
  - Disable motor power



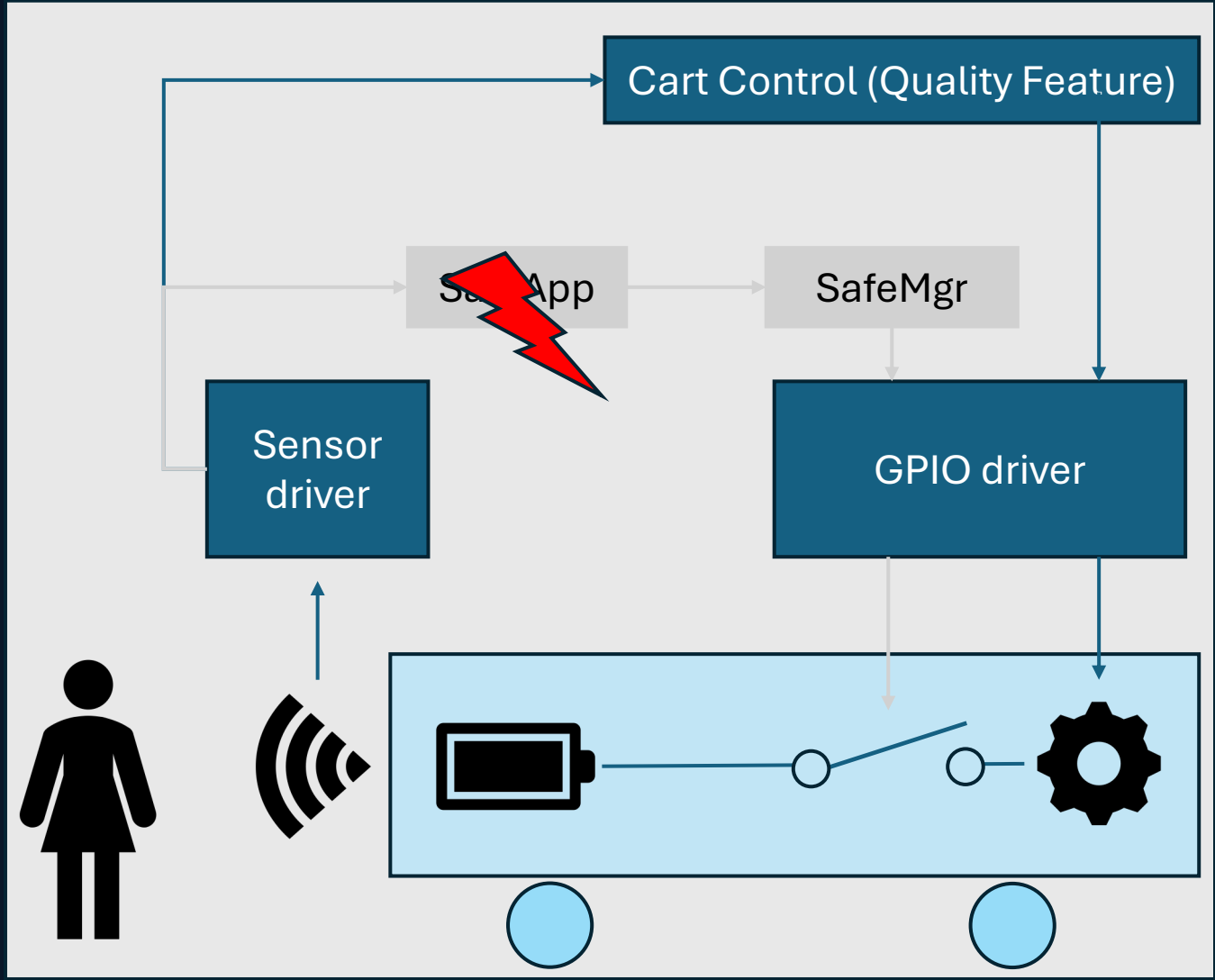
Base System with Safety Critical Path

- Base Functionality
  - Input
    - Distance sensor
  - QM functionality
    - Maintain target distance
    - Distance -> speed up/slow down
  - Output
    - Motor control and power switch
- Safety Functionality
  - Is person within critical zone?
    - In case QM functionality fails
  - Separate safety-critical path
    - SafeApp: calculate distance
    - SafeMgr: determine if distance too close
  - Disable motor power



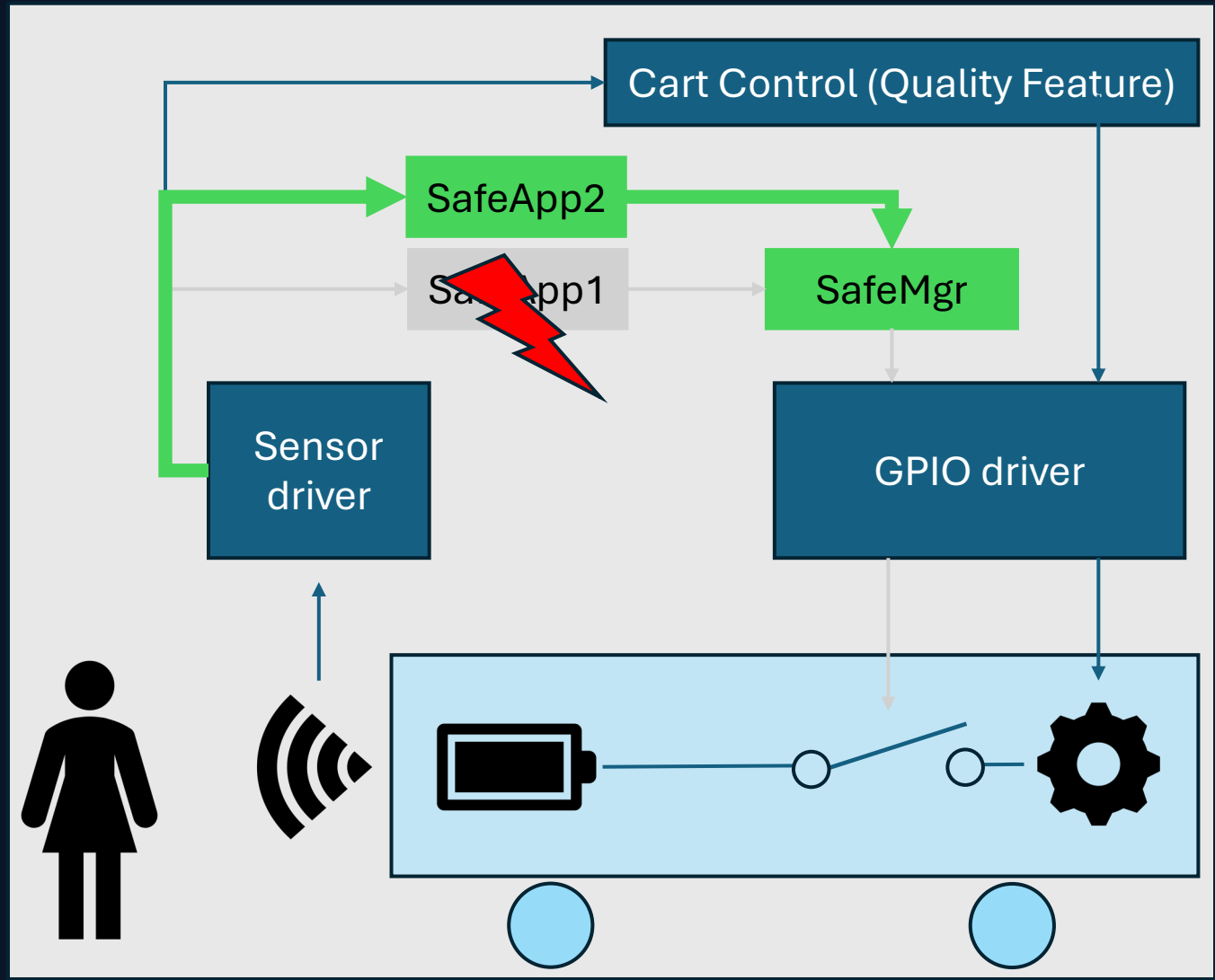
Process Failure

- Correct execution of Safety Process
- Problems
  - Memory corruption, transient faults
  - Incorrect result, Late result
  - Non-responsive, Crash
- Solutions
  - KOS FFI memory
  - ***Redundancy***
  - CRC, internal logic
  - Timeout
  - Restart
- Challenges with redundancy?
  - More software -> more failure
  - Avoid common cause failure
  - How to deal with discrepancy?



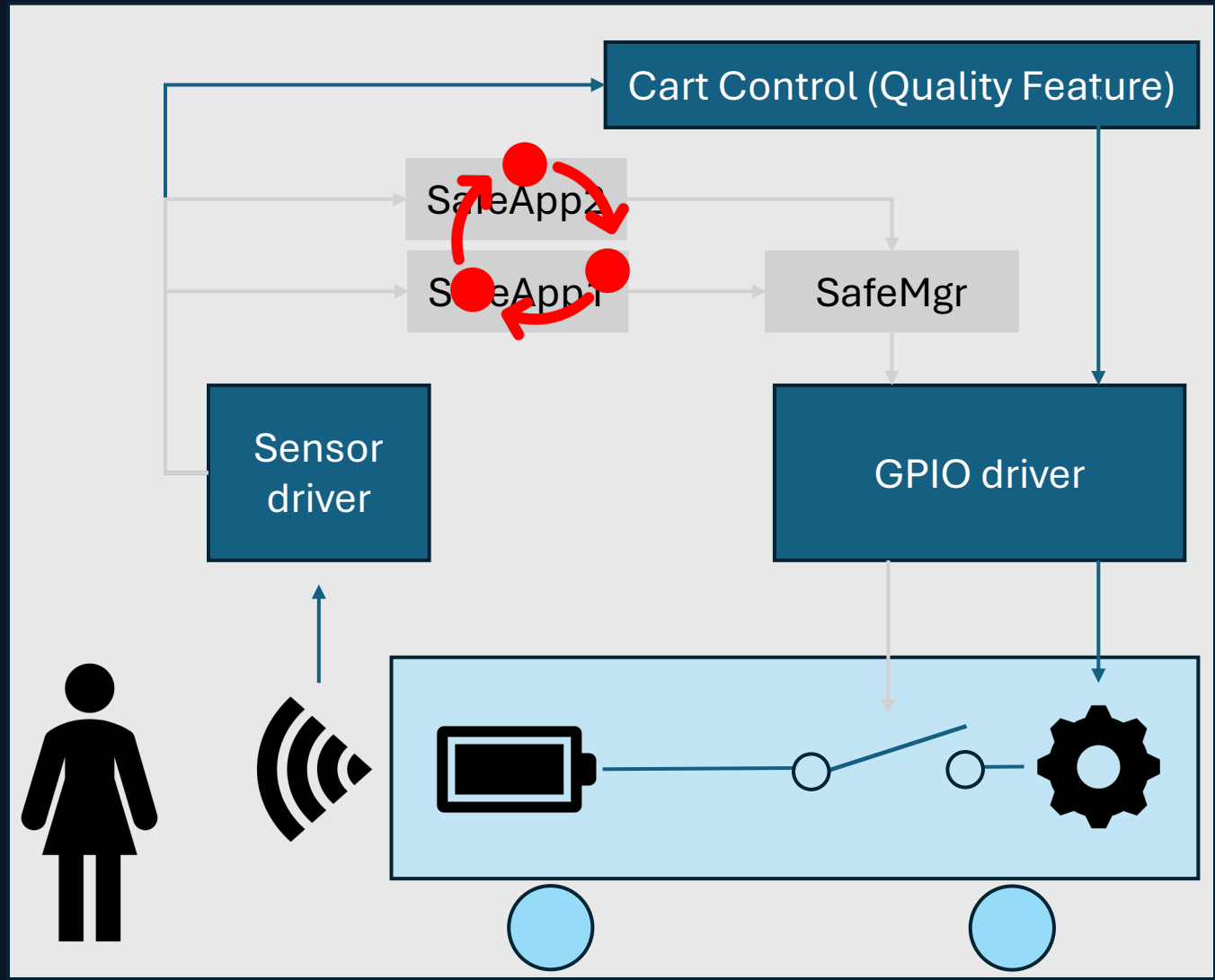
Process Failure

- Correct execution of Safety Process
- Problems
  - Memory corruption, transient faults
  - Incorrect result, Late result
  - Non-responsive, Crash
- Solutions
  - KOS FFI memory
  - ***Redundancy***
  - CRC, internal logic
  - Timeout
  - Restart
- Challenges with redundancy?
  - More software -> more failure
  - Avoid common cause failure
  - How to deal with discrepancy?



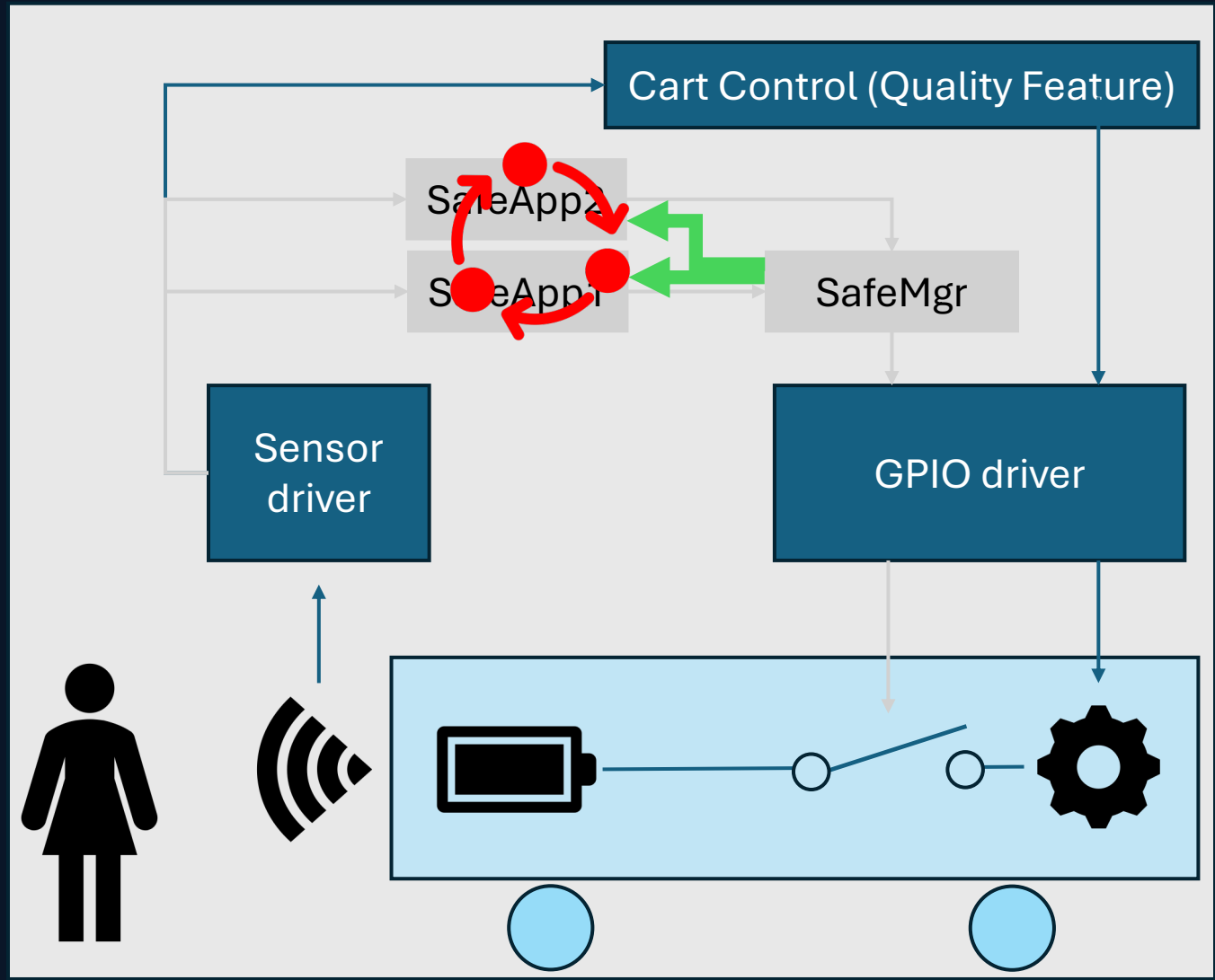
Control Flow Failure

- Execution over multiple processes
- Problems?
  - Incorrect order
  - Incorrect timing
  - Incorrect messaging (content, destination)
- Solutions?
  - ***Event monitoring***
  - Timeout
  - Error detection
  - ***Flow control (synchronization)***



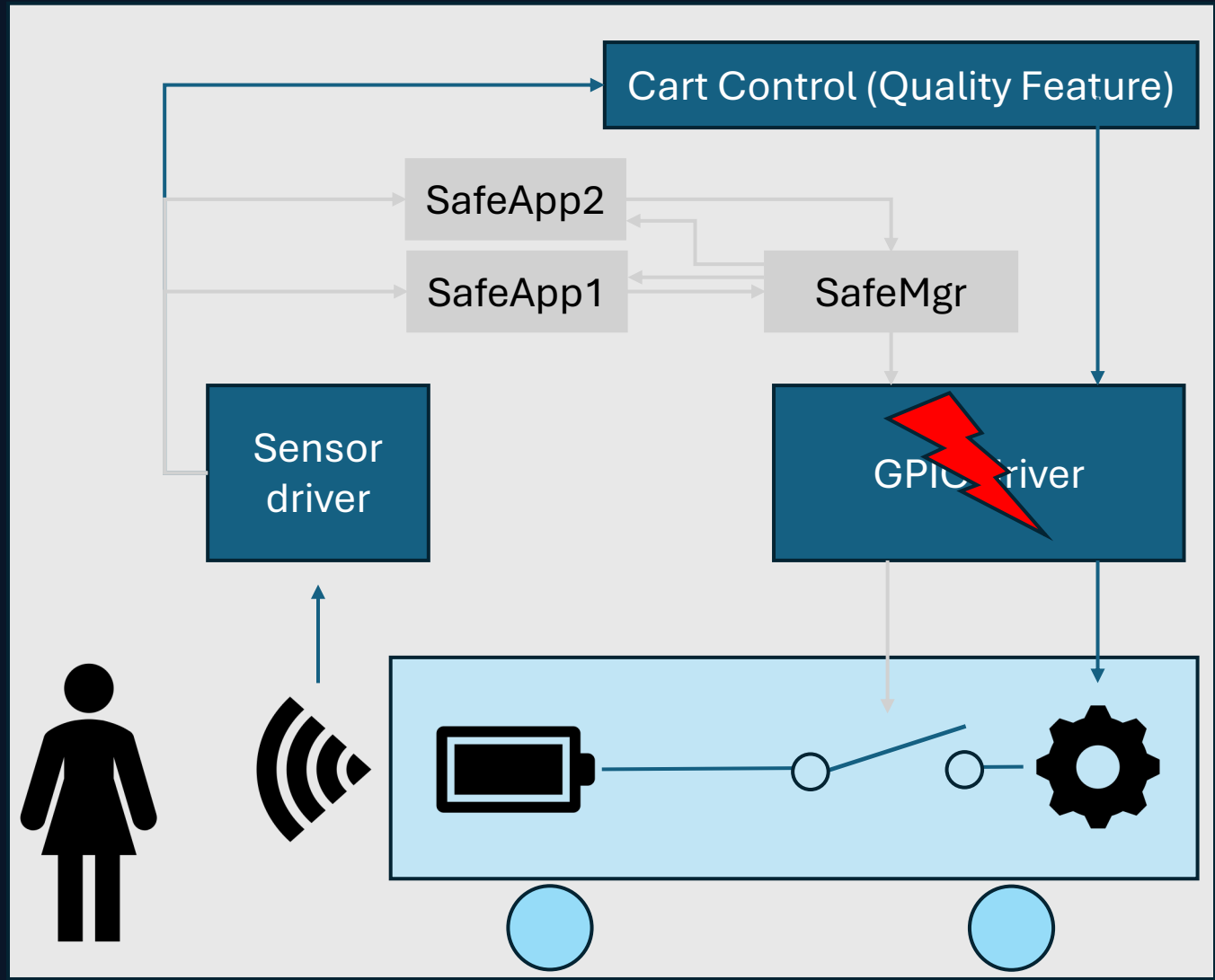
Control Flow Failure

- Execution over multiple processes
- Problems?
  - Incorrect order
  - Incorrect timing
  - Incorrect messaging (content, destination)
- Solutions?
  - ***Event monitoring***
  - Timeout
  - Error detection
  - ***Flow control (synchronization)***



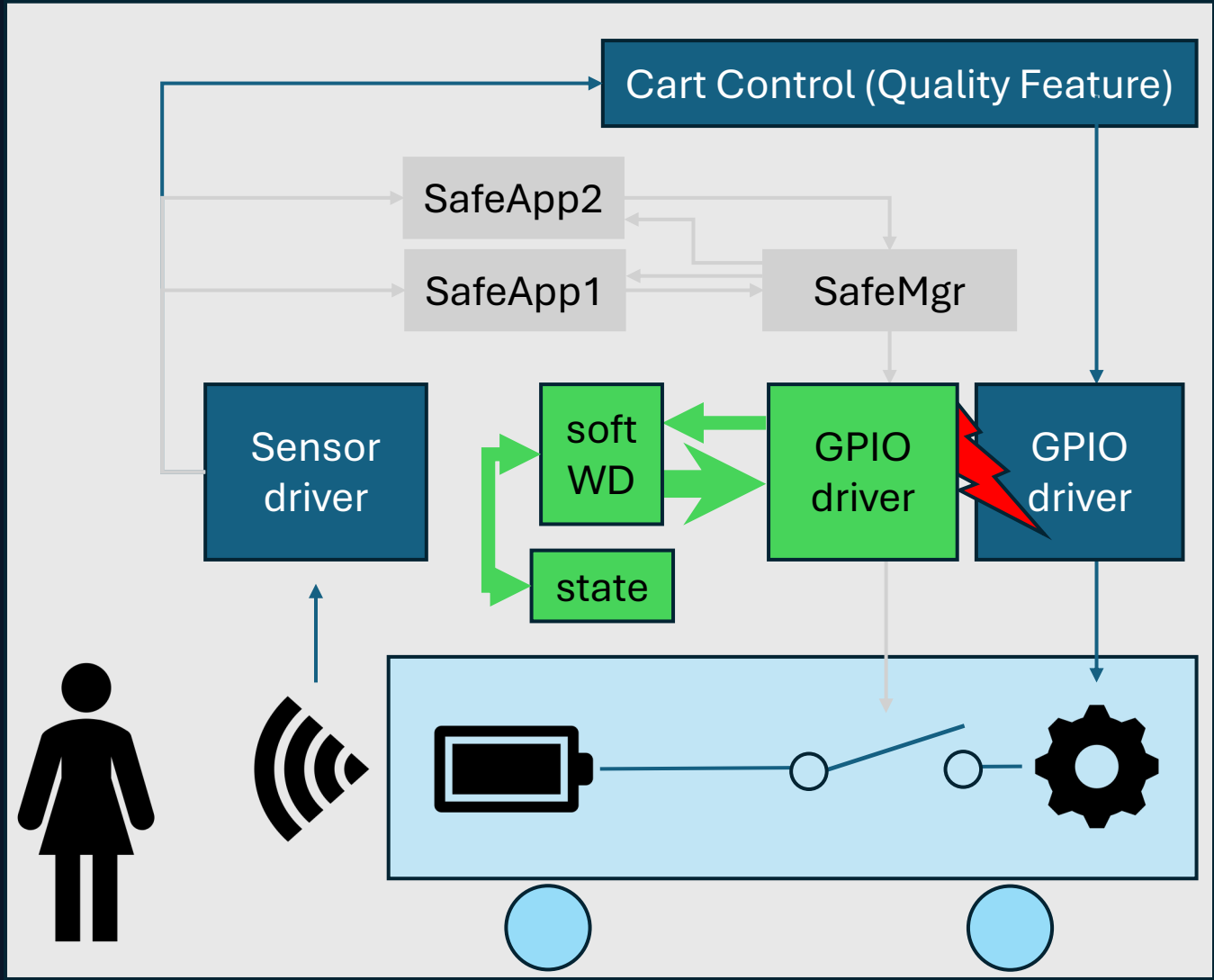
Output Failure

- Ensure safety output is correct
- Problems
  - Output trigger -> no output
  - No output trigger -> output
  - Process failure: driver
  - Control flow failure: trigger lost, spurious trigger
- Solutions
  - Redundancy
  - ***Watchdog***
  - ***Restart***
- Challenges with restart
  - Save state



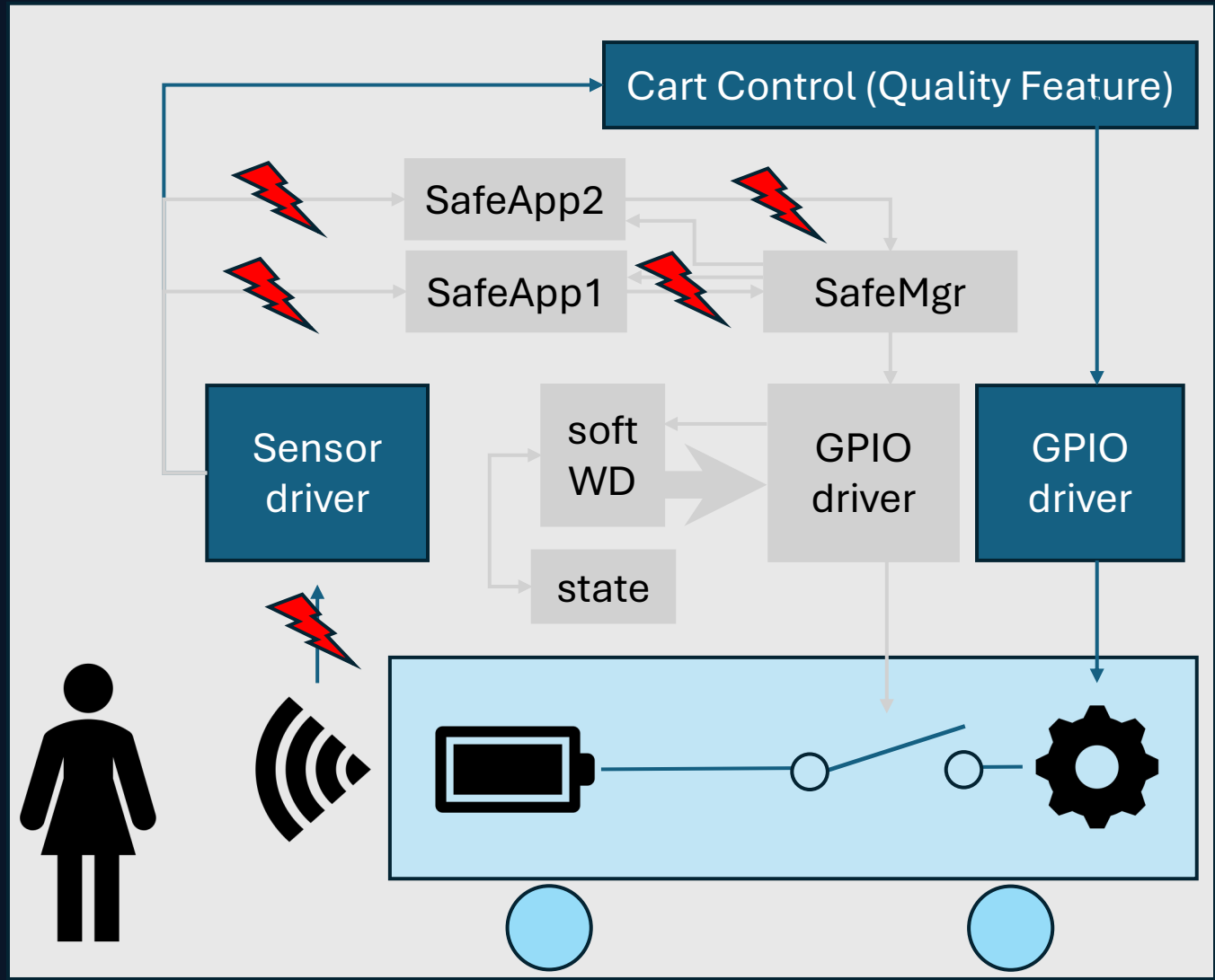
Output Failure

- Ensure safety output is correct
- Problems
  - Output trigger -> no output
  - No output trigger -> output
  - Process failure: driver
  - Control flow failure: trigger lost, spurious trigger
- Solutions
  - Redundancy
  - ***Watchdog***
  - ***Restart***
- Challenges with restart
  - Save state



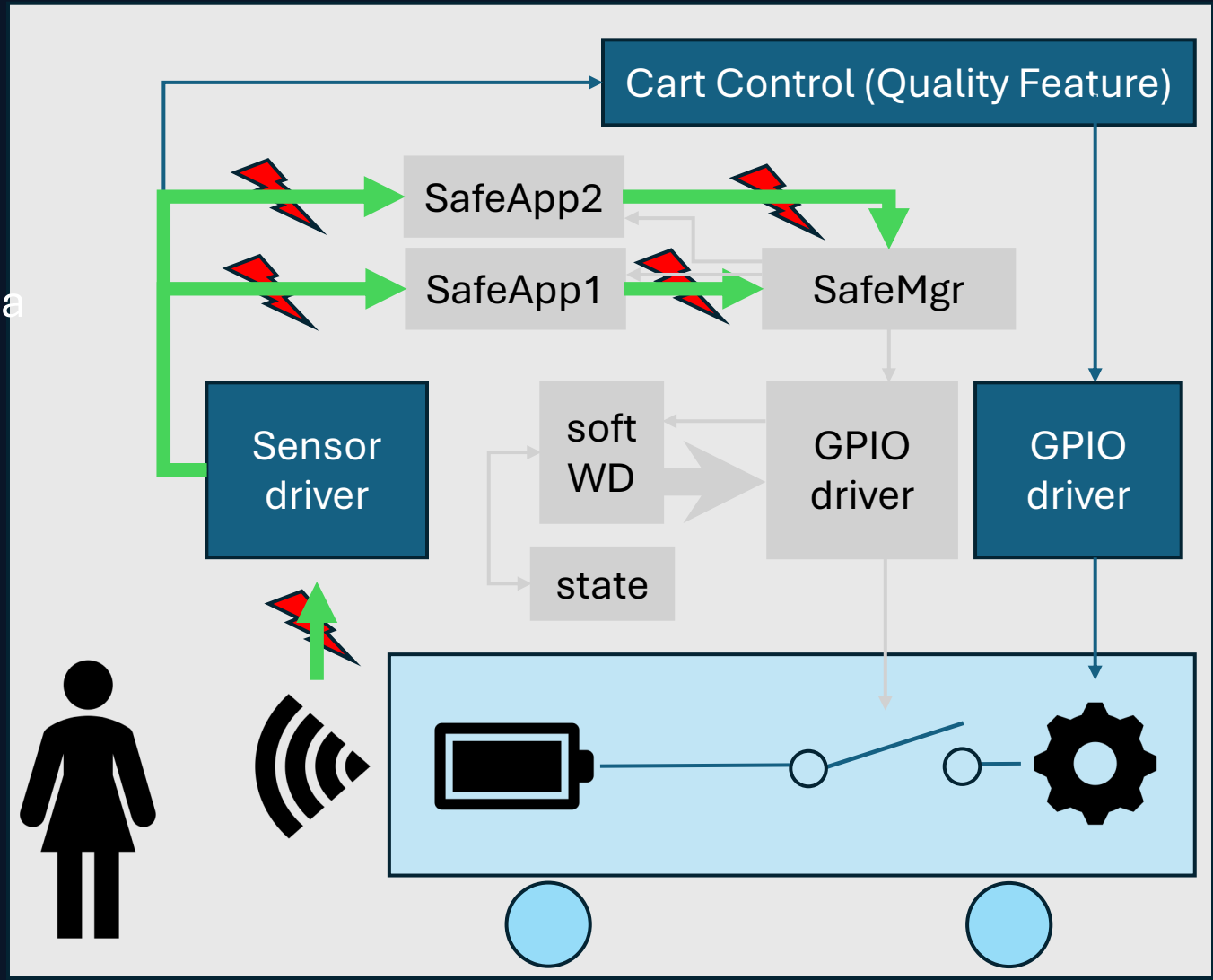
Input Failure

- Validity of incoming values
- Problems
  - Process failure: driver
  - Communication failure: corrupted data
  - Hardware failure
- Solutions
  - ***Error detection (CRC, parity, seq)***
  - Redundancy
  - Error correction
  - ***Discard invalid input***
  - Restart
- Challenges with error detection
  - Limited
  - Detection vs correction



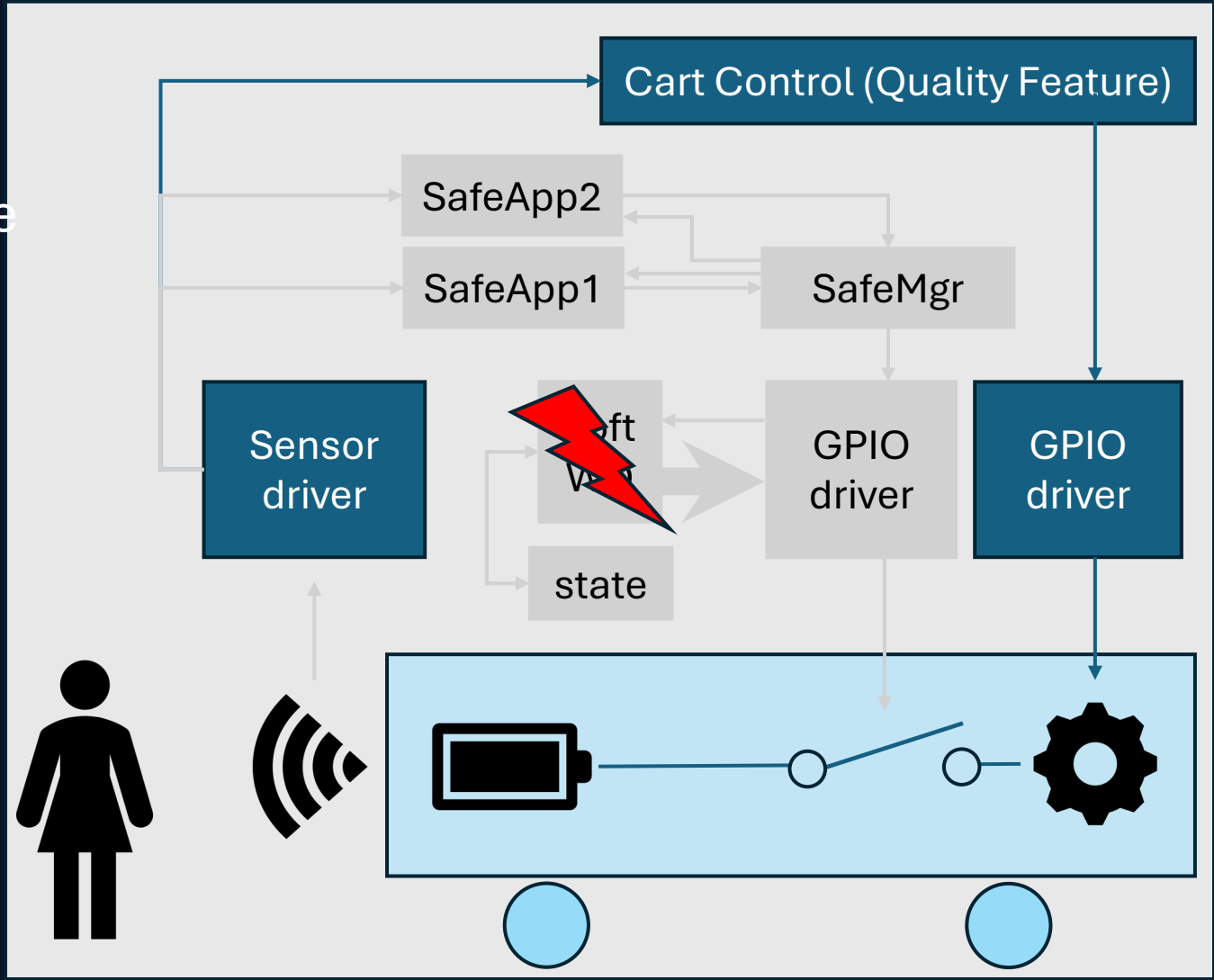
Input Failure

- Validity of incoming values
- Problems
  - Process failure: driver
  - Communication failure: corrupted data
  - Hardware failure
- Solutions
  - ***Error detection (CRC, parity, seq)***
  - Redundancy
  - Error correction
  - ***Discard invalid input***
  - Restart
- Challenges with error detection
  - Limited
  - Detection vs correction



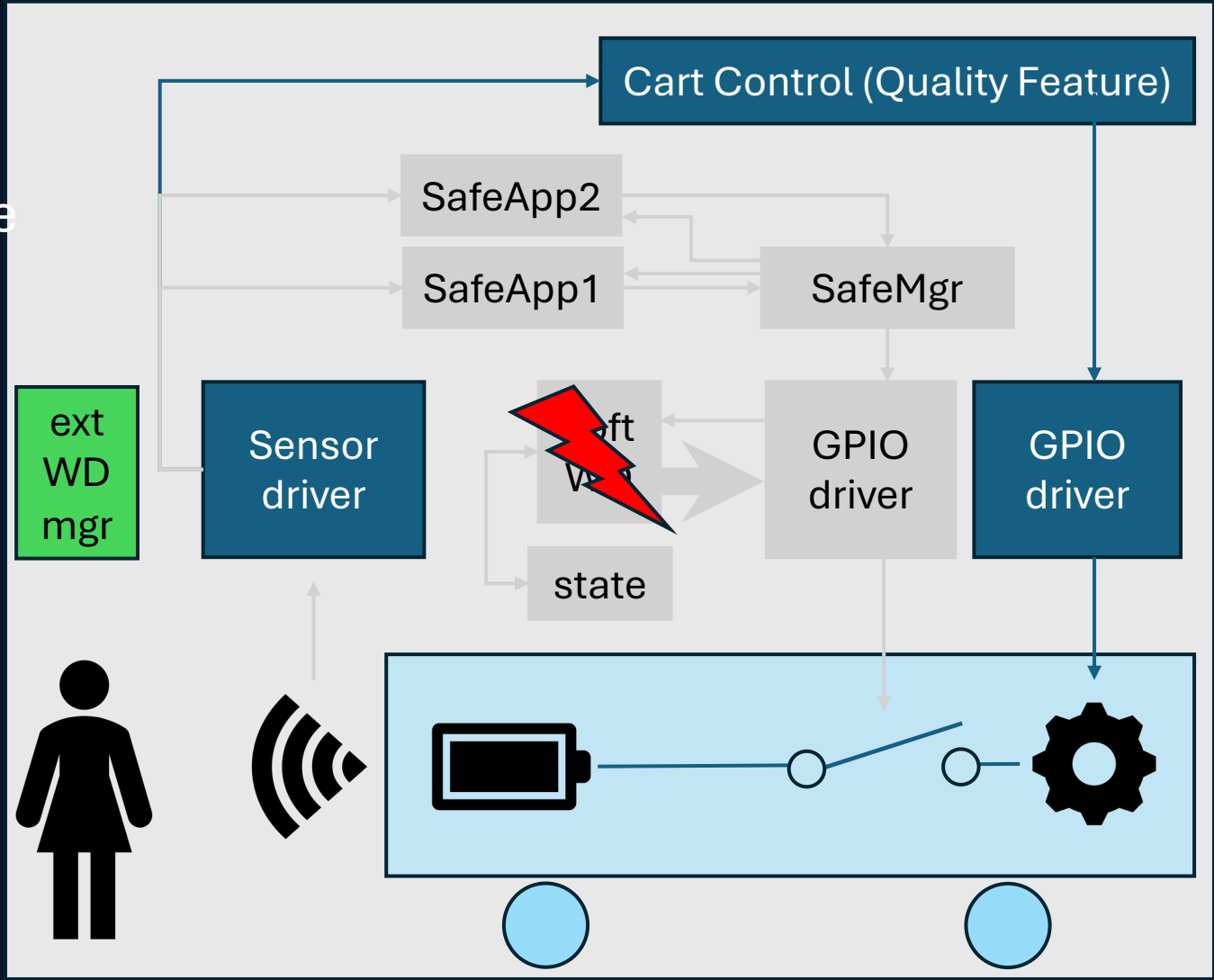
Software Platform Failure

- OS or firmware failure
  - safety functions rely on software platform to run
- Problems
  - Safety processes don’t run
  - No IPC
  - Process corruption
- Solutions
  - ***Watchdog***
  - Restart



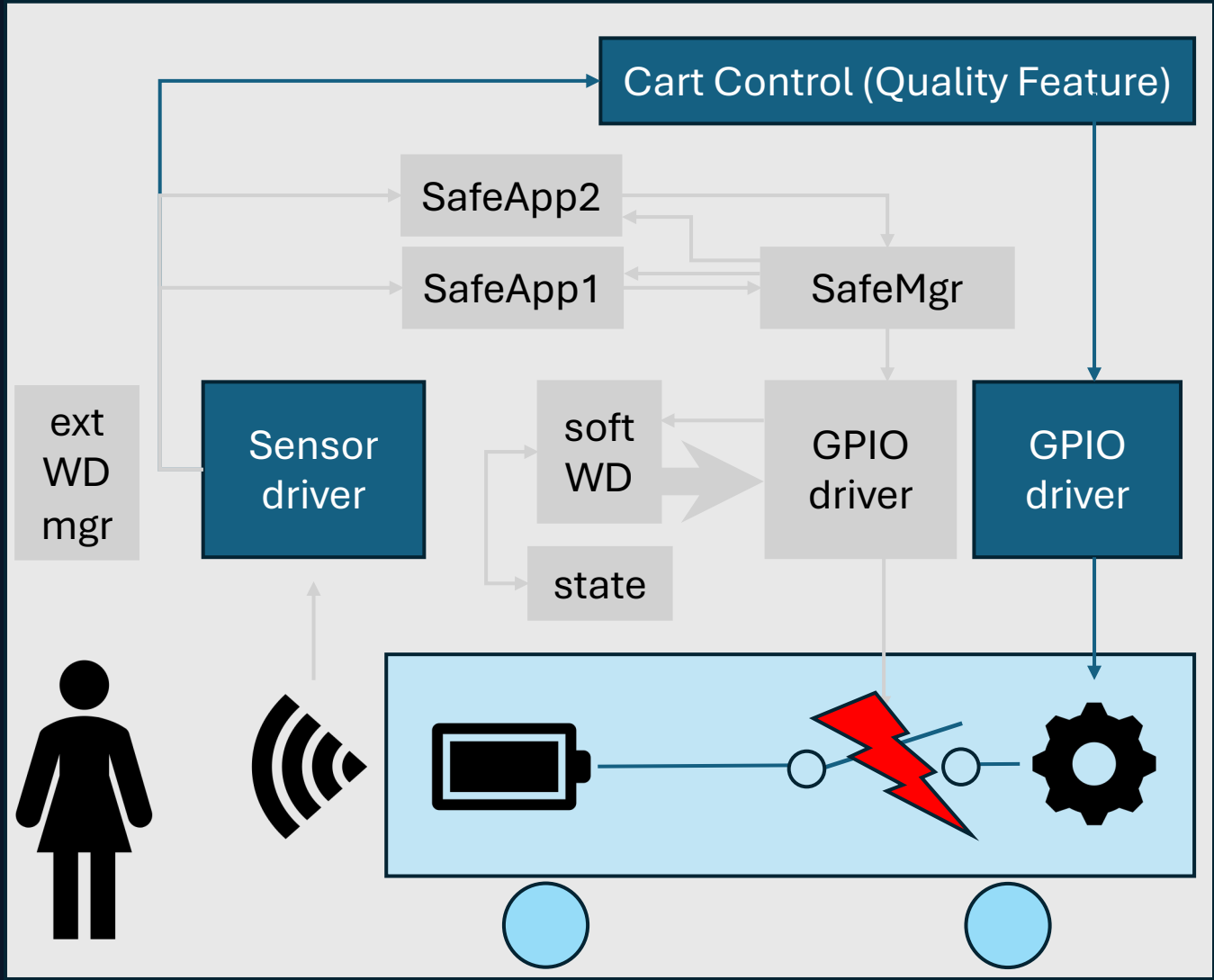
Software Platform Failure

- OS or firmware failure
  - safety functions rely on software platform to run
- Problems
  - Safety processes don’t run
  - No IPC
  - Process corruption
- Solutions
  - ***Watchdog***
  - Restart



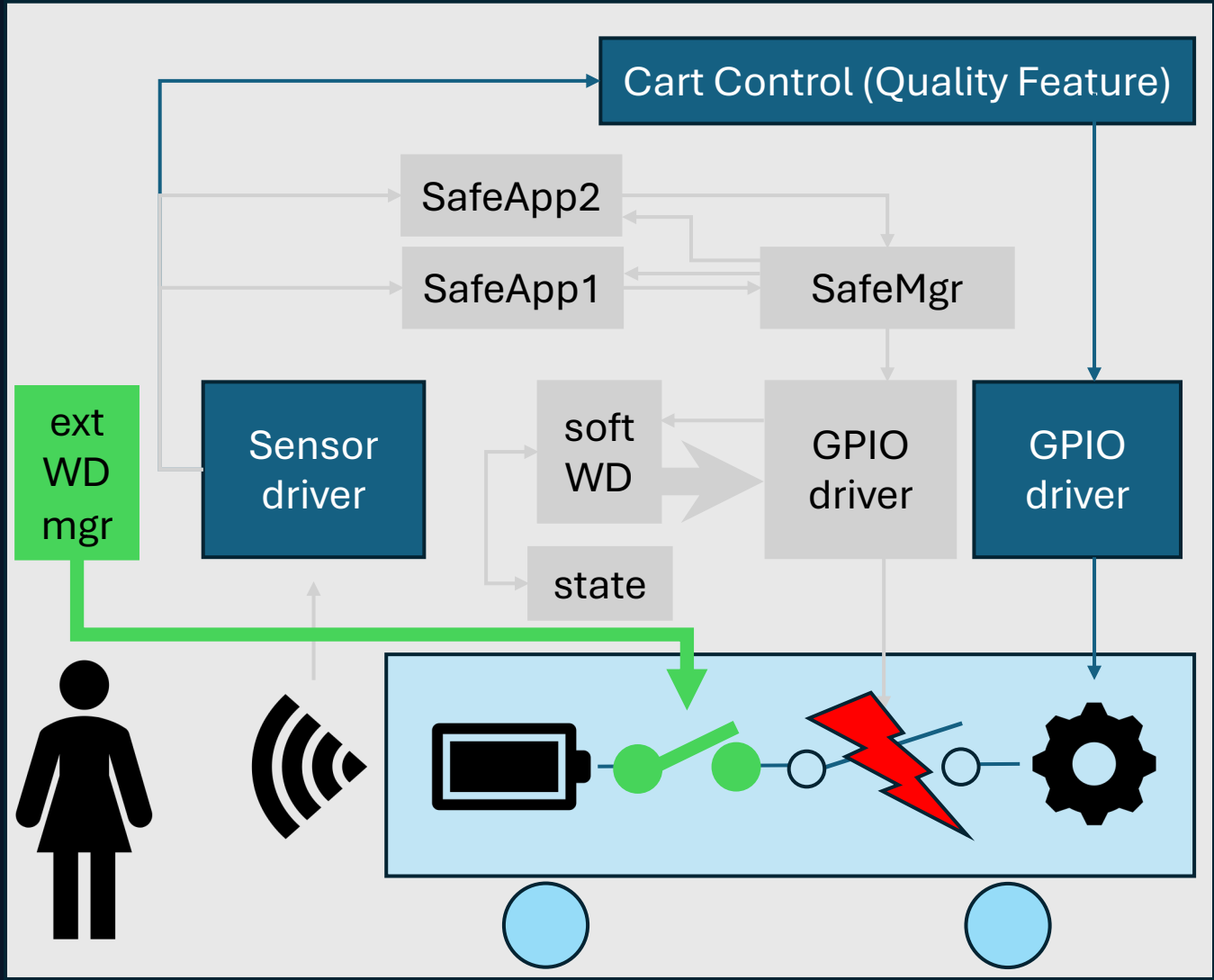
Hardware Failure

- Common cause failure
  - Redundant path -> single hardware switch
  - Redundant software on same CPU
- Problems
  - Hardware failure
- Solutions
  - ***Hardware redundancy***
  - Watchdog

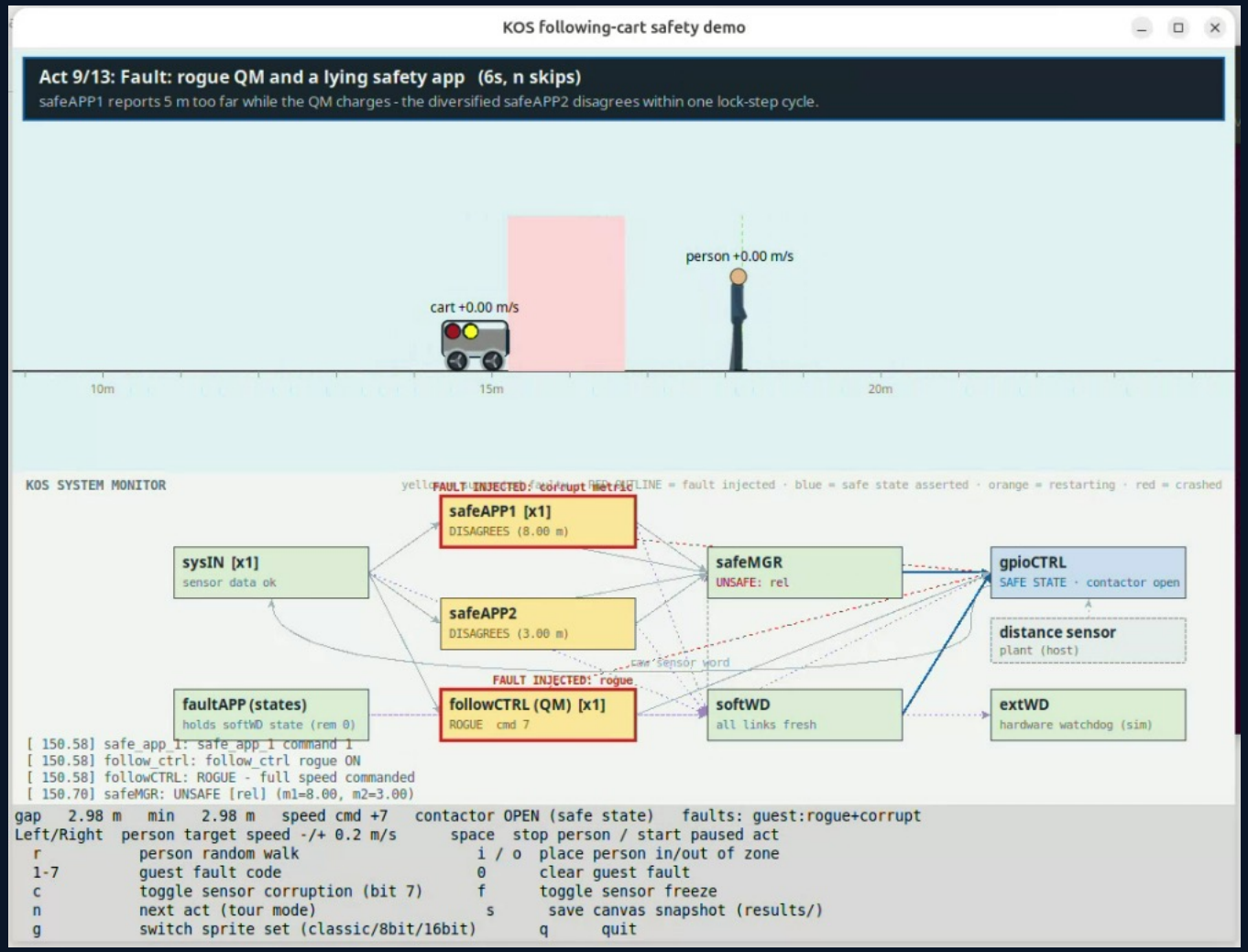


Hardware Failure

- Common cause failure
  - Redundant path -> single hardware switch
  - Redundant software on same CPU
- Problems
  - Hardware failure
- Solutions
  - ***Hardware redundancy***
  - Watchdog



Demo


KOS following-cart safety demo
Act 9/13: Fault: rogue QM and a lying safety app (6s, n skips)
safeAPP1 reports 5 m too far while the QM charges - the diversified safeAPP2 disagrees within one lock-step cycle.
person +0.00 m/s
cart +0.00 m/s
10m
15m
20m
KOS SYSTEM MONITOR
FAULT INJECTED: corrupt metric
safeAPP1 [x1]
DISAGREES (8.00 m)
sysIN [x1]
sensor data ok
safeMGR
UNSAFE: rel
gpioCTRL
SAFE STATE · contactor open
safeAPP2
DISAGREES (3.00 m)
distance sensor
plant (host)
can sensor word
FAULT INJECTED: rogue
faultAPP (states)
holds softWD state (rem 0)
followCTRL (QM) [x1]
ROGUE cmd 7
softWD
all links fresh
extWD
hardware watchdog (sim)
[150.58] safe_app_1: safe_app_1 command 1
[150.58] follow_ctrl: follow_ctrl rogue ON
[150.58] followCTRL: ROGUE - full speed commanded
[150.70] safeMGR: UNSAFE [rel] (m1=8.00, m2=3.00)
gap 2.98 m min 2.98 m speed cmd +7 contactor OPEN (safe state) faults: guest:rogue+corrupt
Left/Right person target speed -/+ 0.2 m/s space stop person / start paused act
r person random walk i / o place person in/out of zone
1-7 guest fault code 0 clear guest fault
c toggle sensor corruption (bit 7) f toggle sensor freeze
n next act (tour mode) s save canvas snapshot (results/)
g switch sprite set (classic/8bit/16bit) q quit

Summary

- Functional Safety
  - Certification
  - SEooC
  - Safety Manual
- Safety Process
  - Consider failures
  - Apply mechanisms
- Example: Follow Cart
  - KOS system
  - Improve fault tolerance

- Future Work
  - Evaluation and Certification of Safety Manual
  - Fail operational
  - Multicore
  - Real-time scheduling