Redundancy Management

I'm not dead!
'Ere, he says he's not dead.
Yes he is.
I'm not.
He isn't.
Well, he will be soon, he's very ill.
I'm getting better!

— Monty Python

These tutorials are a simplified introduction, and are not sufficient on their own to achieve system safety. You are responsible for the safety of your system.
Is Your Redundancy Working?

- **Anti-Patterns for Redundancy:**
  - Unsafe because double-spending redundancy
  - No between-mission redundancy diagnostics
  - Low test coverage on redundant components

- **Redundant components help reliability**
  - But, what happens when a component breaks?
    - Need to gracefully curtail current mission
    - Prohibit additional missions until repaired
  - Reliability assumes perfection at mission start
    - Untested redundancy undermines reliability

Bellingham WA, June 1999: Gasoline spill & fire kills 3 due to improper management of SCADA redundancy

https://goo.gl/oBNIiw
Use of Redundancy: Availability
- Hot Standby takes over upon failure
- Assumes somehow you detect failure
  - For low criticality systems, perhaps it’s OK to miss some failures; have human trigger failover

Even if only one component breaks at a time...
- Single computer can fail “active” (dangerous)
- Self-test cannot find all faults
- Single component is unsafe for SIL 3,4

Use of Redundancy: Fault Detection
- 2-of-2 used for fault detection
Fail Operational Approaches

- Can’t double-spend redundancy!
  - Need 2 components to detect a failure
  - **PLUS** more components to operate after failure

- Triplex modular redundancy (2-of-3)
  - Three copies of subsystem and voter
  - But ... voter can be single point of failure!

- Dual 2-of-2
  - Two copies of subsystem for availability
  - Each subsystem is 2-of-2 to provide fault detection
Hybrid of Low SIL Doer and High SIL 2-of-2 checker

- Single Low SIL primary
  - Provides normal functionality
- 2-of-2 High SIL checker
  - Shuts down if primary unsafe
  - Shuts down if cross-check fails

Common building blocks:
- 2-of-2 for fault detection
- Doer/Checker for fault isolation
- Hot standby for fail operational
Reliability math assumes all redundancy working
- On-line diagnostics: self-test at start of mission
  - Example: IEC 60730 self-test library
- Off-line diagnostics: “Proof test”
  - Example: exercise an elevator safety limit switch

Latent undetected faults
- Undetectable faults lead to coincident failures
  - 2-of-2 doesn’t work if both fail the same way!
- Run-time detection: frequent health cross-checks
  - Scrub state, e.g., compare RAM values
  - Swap active units periodically to self-test
- Off-line detection: enforce periodic proof tests
  - Self-test or require diagnostic to resume operation
What happens when component fails?
- Some redundancy is for fault detection
- Other redundancy is for availability
- Plan how to detect & survive failures

Diagnostic coverage matters
- Pre-mission test; cross-checks; proof tests
- Minimize potential for latent faults

Pitfalls:
- Don’t double-spend your redundancy (detect & failover are different)
- Look for common-mode failures (e.g., software updates)