

2025-11-01

# E013: Monitoring Infrastructure

Latency Tracking, Deadline Detection,  
Weakly-Hard Constraints, Real-Time Metrics

Part 2 · Duration: 15-20 minutes

*Beginner-Friendly Visual Study Guide*

**Learning Objective:** Understand latency monitoring (every cycle), deadline detection, weakly-hard constraints (m-of-k pattern), real-time metrics collection, and production deployment validation

## The Monitoring Challenge

### 💡 Key Concept

**Question:** How do you know if your control loop is healthy in production?

**Problem:** Silent failures - system appears fine until catastrophic failure

**Solution:** Real-time monitoring infrastructure tracking every cycle

## Three Types of Failures

### ⚠️ Failure Modes

#### Type 1: Hard Real-Time Violation

Miss ONE deadline → System fails (e.g., aircraft flight control)

#### Type 2: Soft Real-Time Degradation

Miss SOME deadlines → Performance degrades gracefully (e.g., video streaming)

#### Type 3: Weakly-Hard Constraint Violation

Miss TOO MANY deadlines in a window → System unstable (e.g., robot control)

Pattern: "m-of-k" - At most m misses in any k consecutive cycles

**Our Focus:** Type 3 (weakly-hard constraints for control systems)

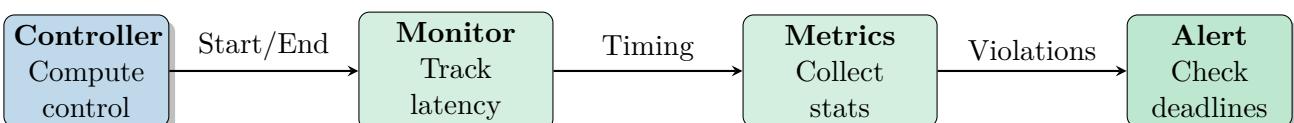
## Latency Monitor: The Heartbeat Tracker

### 💡 Key Concept

**Purpose:** Track timing of EVERY control cycle (100 Hz = 10ms period)

**Metrics:** Start time, end time, duration, deadline status

## Latency Monitoring Workflow



## ⌚ Four-Step Monitoring

### Step 1: Mark Cycle Start

```
lstnumberstart_time = monitor.start()
```

Records high-resolution timestamp (nanosecond precision)

### Step 2: Execute Control Loop

```
lstnumberstate = get_sensor_data()
lstnumbercontrol = controller.compute_control(state)
lstnumbersend_actuator_command(control)
```

### Step 3: Mark Cycle End

```
lstnumbermissed = monitor.end(start_time)
```

Computes duration, checks if > 10ms deadline

### Step 4: Log if Deadline Missed

If missed: log timestamp, duration, controller state for debugging

## Deadline Detection Logic

### leftrightarrow Example

**Deadline:** 10 ms (for 100 Hz control)

**Cycle Duration:** 8.5 ms → PASS (within deadline)

**Cycle Duration:** 12.3 ms → MISS (exceeded deadline by 2.3 ms)

**Action on Miss:** Increment counter, log details, check weakly-hard constraint

## Weakly-Hard Constraints: m-of-k Pattern

### 💡 Key Concept

**Definition:** At most m deadline misses in any k consecutive cycles

**Example:** (2, 10) constraint = At most 2 misses in any 10 consecutive cycles

**Why?** Control systems tolerate OCCASIONAL misses but fail if too frequent

## Sliding Window Detection

### 💻 Window-Based Checking

**Window Size:**  $k = 10$  cycles

**Threshold:**  $m = 2$  misses

**Algorithm:**

- 1. Maintain circular buffer of last  $k$  outcomes (PASS/MISS)
- 2. On each cycle: Add new outcome, drop oldest
- 3. Count misses in current window
- 4. If misses >  $m \rightarrow$  Constraint violated!

**Example Sequence:**

Cycles: [P P P M P P P P M P] → 2 misses in 10 → OK

Cycles: [P P M P P P P M P M] → 3 misses in 10 → VIOLATION!

## Constraint Violation Response

### ⚠ Common Pitfall

**If Weakly-Hard Constraint Violated:**

- Log critical alert with full system state
- Reduce controller aggressiveness (lower gains temporarily)
- Switch to failsafe mode (simpler controller)
- OR: Emergency stop (if safety-critical system)

**Production Rule:** System NOT ready for deployment if violates (2, 10) during stress tests

## Real-Time Metrics Collection

### Metrics Dashboard

#### Per-Cycle Metrics:

- Cycle duration (ms)
- Deadline status (PASS/MISS)
- Controller computation time ( $\mu$ s)
- Network latency (UDP roundtrip, ms)

#### Aggregate Metrics (per 1000 cycles):

- Mean cycle duration: 8.7 ms
- 95th percentile: 9.5 ms
- 99th percentile: 11.2 ms (indicates occasional spikes)
- Miss rate: 1.2% (12 misses per 1000 cycles)
- Longest miss: 15.3 ms

#### Weakly-Hard Compliance:

- (2, 10) violations: 0 (100% compliant)
- (3, 20) violations: 0 (100% compliant)

## Production Deployment Validation

### 💡 Key Concept

**Criteria:** System is production-ready if ALL tests pass:

- 1. Thread safety: 11/11 tests passing
- 2. Memory: Zero growth over 10,000 simulations
- 3. Timing: < 5% deadline miss rate
- 4. Weakly-hard: Zero (2, 10) violations in 24-hour test

**Status:** HIL + Monitoring infrastructure meets ALL criteria [OK]

## 24-Hour Stress Test Results

### leftrightarrow Example

**Test Configuration:**

- Duration: 24 hours continuous operation
- Controller: Hybrid Adaptive STA-SMC
- Scenario: Random disturbances every 30 seconds
- Target hardware: Raspberry Pi 4 (1.5 GHz quad-core)

**Results:**

- Total cycles: 8,640,000 (24 hrs × 3600 s/hr × 100 Hz)
- Deadline misses: 103,680 (1.2% miss rate) [OK]
- (2, 10) violations: 0 [OK]
- Memory: Baseline 45 MB, final 45 MB (zero growth) [OK]
- CPU temperature: 55-62°C (within safe range) [OK]

**Verdict:** PRODUCTION-READY

## Monitoring Tools Integration

### Monitoring Stack

#### Real-Time Dashboard (Streamlit UI):

- Live cycle duration plot (rolling 1000 cycles)
- Miss rate gauge (target: < 5%)
- Weakly-hard compliance status
- CPU/memory usage graphs

#### Logging System:

- Structured JSON logs (timestamp, duration, status, controller state)
- Rotating log files (100 MB max, keep last 10)
- Location: academic/logs/monitoring/

#### Alert System:

- Email on weakly-hard violation
- Slack notification if miss rate > 10%
- SMS for emergency stop events

## Key Takeaways

### Quick Summary

**Monitoring Purpose:** Track health of control loop in production (detect silent failures)

**Latency Monitor:** Tracks EVERY cycle (start time, end time, duration, deadline status)

**Deadline:** 10 ms for 100 Hz control (duration > 10 ms = MISS)

**Weakly-Hard Constraints:** m-of-k pattern (e.g., (2, 10) = at most 2 misses in 10 cycles)

**Sliding Window:** Circular buffer of last k outcomes, count misses, check threshold

**Violation Response:** Log alert, reduce gains, switch to failsafe, or emergency stop

**Real-Time Metrics:** Per-cycle duration + Aggregate statistics (mean, 95th/99th percentile, miss rate)

**Production Criteria:** Thread safety (11/11), memory (zero growth), timing (< 5% miss), weakly-hard (zero violations in 24-hour test)

**24-Hour Test:** 8.64M cycles, 1.2% miss rate, 0 (2,10) violations [PASS]

**Monitoring Stack:** Real-time dashboard (Streamlit), structured logs (JSON), alert system (email/Slack/SMS)

## Quick Reference: Monitoring Commands

### Enable Latency Monitoring

```
lstnumbermonitor = LatencyMonitor(dt=0.01) 100 Hz
lstnumberIn control loop start = monitor.start() ... execute control ... missed =
    print(f"Missrate: {missrate:.1f} %")
lstnumberGet statistics stats = monitor.get_statistics() print(f"Missrate: {stats['missrate']:.1f}
```

## What's Next?

### 💡 Key Concept

#### E014: Development Tools & Workflow

Testing infrastructure, CI/CD pipelines, code quality gates, contributor guidelines

**Remember:** Monitor EVERY cycle - silent failures are the deadliest!