# Quality & Validation Strategy

Embedded Thermal Control System – Wearable/IoT Device

# 1. System Overview

This subsystem manages thermal control in a wearable or IoT device by dynamically adjusting fan speed based on temperature readings. It includes:  
  
- MCU: Executes firmware control logic  
- Temperature Sensor (I2C): Periodically reports ambient temperature  
- PWM Fan with Tachometer Feedback: Receives PWM signal from MCU; returns RPM via tacho pin  
  
The firmware implements a control loop that continuously monitors temperature and adjusts fan speed to maintain a stable target temperature, optimizing for user comfort, device longevity, and performance.

# 2. Firmware Control Loop

## High-Level Logic

1. Read temperature from the I2C sensor at defined intervals (e.g., every 500 ms)  
2. Compare current temperature with target threshold  
3. Compute appropriate PWM duty cycle based on control logic (e.g., PID or simple linear scaling)  
4. Output PWM signal to fan  
5. Read tacho feedback to verify fan is running at expected RPM  
6. Log / respond to discrepancies (e.g., tolerance breach, sensor error)

## Expected Behavior

- Temperature deviations should trigger proportional fan speed changes  
- RPM should track closely to target value (within tolerance)  
- In the event of a hardware fault (fan stall, sensor failure), the system must raise alerts or enter safe mode

# 3. Quality Objectives

|  |  |
| --- | --- |
| Objective | Description |
| Accuracy | Maintain temperature within ±1°C of setpoint |
| Responsiveness | Adjust fan PWM within 1 second of temperature change |
| Reliability | Detect fan or sensor faults within 3 cycles |
| Testability | Provide accessible debug/log outputs for validation and troubleshooting |
| Efficiency | Minimize unnecessary fan speed increases to save power and reduce noise |

# 4. Validation Strategy

## Unit Testing

- Validate conversion logic from temperature to PWM duty cycle  
- Confirm debounce and filtering logic for noisy tacho readings  
- Check boundary values and exception handling in firmware functions

## Integration Testing

- Test fan RPM response to PWM duty changes  
- Validate I2C communication under various voltage and load conditions  
- Cross-check tacho feedback against PWM commands in real time

## Edge Case Testing

- Sensor disconnection or noise: Ensure fallback/alert behavior triggers  
- Fan stall or low-RPM: Validate detection and safety shutdown logic  
- Out-of-range temperature: Confirm max PWM output without overshooting  
- High ambient heat: Assess system behavior under sustained thermal load

# 5. Tools & Equipment

|  |  |
| --- | --- |
| Tool | Purpose |
| Thermal test chamber | Simulate varied heat profiles |
| Mock sensors | Test sensor failure modes and invalid inputs |
| Python test framework | Execute test cases and analyze logs |
| Serial terminal / CLI | Interact with device and inject test scenarios |
| Logic analyzer | Verify PWM and tacho signal timing accuracy |

# 6. Metrics & Reporting

|  |  |
| --- | --- |
| Metric | Target Value / Notes |
| RPM tolerance | ±5% from expected at given PWM level |
| Temperature stabilization | Reach target ±1°C within 30s under load |
| Sensor error rate | <0.5% over 24-hour continuous run |
| Fault detection latency | <2s from fault to detection |
| Log frequency for anomalies | No missed logs during simulated failure events |