TEMPERATURE MONITORING SYSTEM – SOFTWARE ARCHITECTURE OVERVIEW

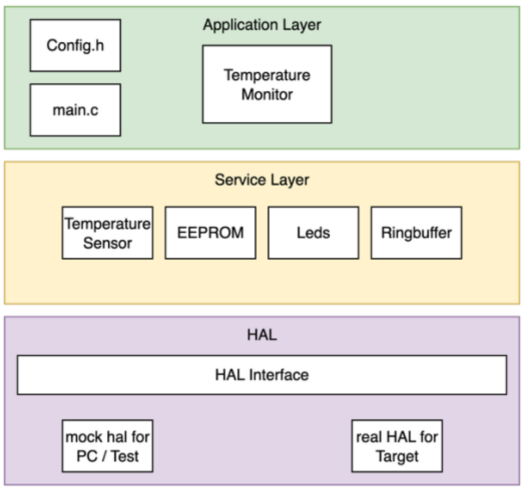
1. PURPOSE AND SCOPE

This document describes the architecture of the embedded temperature monitoring system. The software provides temperature measurement, status indication, and configurable behavior across both PC-simulation and target hardware builds.

The architecture follows a layered and modular design with strict separation between hardware-independent logic and hardware access through a unified HAL (Hardware Abstraction Layer). All functional modules use Dependency Injection for clear interfaces and testability.

---

2. ARCHITECTURE



Layer	Description	Main Components
Application Layer	High-level system control and execution flow. Initializes and coordinates all modules.	main.c, TemperatureMonitor
Service / Logic Layer	Core logic and data flow. Each module encapsulates a single responsibility.	TemperatureSensor, Eeprom, Led, RingBuffer
HAL Layer	Provides unified hardware access functions. Build-time selectable between mock and real hardware.	hal_if.h, mock_hal.c, real_hal.c
Hardware / Mocks	Physical interfaces or software simulations.	EEPROM, ADC, GPIO

3. Module Overview

Temperature Monitor

- Consumes raw samples from the ring buffer.
- Converts to Celsius via the TemperatureSensor.
- Classifies temperature into states:
  - TEMP_STATUS_INIT – all LEDs ON (no valid data yet)
  - TEMP_STATUS_NORMAL – below warning threshold
  - TEMP_STATUS_WARNING – above 85 °C
  - TEMP_STATUS_CRITICAL – above 105 °C or below 5 °C
- Drives three LEDs (green, yellow, red) to represent the current state.
- Provides TemperatureMonitor_Poll() for non-blocking updates.

Temperature Sensor

- Converts ADC raw values to Celsius using a hardware-specific scaling factor. Was choosen so we don't create convert functions, so in this case it is more elegant.
- Factor derived from system_config.hw_revision read from EEPROM.
- Provides an ISR-safe sampling function (TemperatureSensor_IsrSample()) that pushes raw data to the RingBuffer.
- Provides a non-ISR conversion function for debugging purposes only. Not safe with ISR

EEPROM

- Reads system configuration (system_config_t) from non-volatile memory.
- In mock HAL provides accurate EEPROM behaviour by filling data with 0xFF.
- Uses injected I2C read function (hal_i2c_read) to access memory.

LED

- Abstracts LED control using an injected GPIO write function.
- Interface: Led_Init(gpio_set_fn, pin) and Led_Set(on).
- Keeps API minimal to ensure fast response and low coupling.
- Active High logic is not included in this, but can be added easily.

Ring Buffer

- Lock-free single-producer/ single-consumer buffer between ISR and main loop.
- Power-of-two size with bitmask wrap for constant-time operations and almost no jitter
- Provides standard functions as Push for ISR use only, and Pop for the consumer.
- Used by TemperatureSensor ISR and TemperatureMonitor main loop.
- Timestamp is not added because of simplicity, but can be added by modification of the buffer entry container
- Overflow handled in a simple way, if consumer is to slow and buffer is full, new values will be dropped. Decition made because assumed that more important is to have constant time behaviour, and miss some values.

HAL Interface

- Common interface used by all modules:
  - hal_i2c_read() – EEPROM access
  - hal_adc_read() – ADC sample acquisition
  - hal_gpio_set() – LED output control
  - hal_adc_irq_handler() – optional simulated ADC interrupt entry point
- **mock_hal.c** provides PC-based emulation (prints to console).
- **real_hal.c** provides empty or hardware-specific implementations.

---

4. Data and Control Flow

1. **Startup Sequence**
   1. main.c initializes all components.
   2. System_init() loads configuration (hw_revision, serial number) and initializes the ring buffer.
   3. TemperatureSensor_Init() sets scaling factor based on revision.
   4. Initialization of the LEDs
   5. TemperatureMonitor_Init() wires sensor, ring buffer, and LEDs.
   6. System enters monitoring loop.

2. **Runtime Operation**
   - **ISR Context:**
     hal_adc_irq_handler() → TemperatureSensor_IsrSample()
     → pushes raw ADC value into RingBuffer.
   - **Main Loop:**
     TemperatureMonitor_Poll() drains buffer, converts readings,
     classifies temperature, prints debug output, and updates LEDs.

---

5. Design Principles

- **Dependency Injection:**
  All hardware interactions are injected as function pointers at initialization.
  This enables seamless mocking and testing on a PC build.
- **Encapsulation and Opaque Handles:**
  Each module exposes an opaque handle type (typedef struct Module Module;) to hide internal data structures, ensuring modularity and ABI stability.
- **Deterministic and Real-Time Safe:**
  The ISR path executes in constant time with integer arithmetic only minimal amount of branches
  All floating-point and print operations occur in the main loop.
- **Build-Time Flexibility:**
  Controlled by CMake option USE_MOCK_HAL:
  - ON → Uses mock_hal.c for PC testing.
  - OFF → Uses real_hal.c for target hardware.

- **Portability:**
  Code is ISO C11 compliant and hardware-agnostic at logic level.
  Platform-specific code is isolated within the HAL.

---

6. Known Issues

Despite a clean modular structure, several areas require attention before production readiness:

1. **Missing Unit Tests**
   - No automated tests currently validate functional correctness.
   - The system compiles and runs manually, but integration or regression testing is not in place.
   - **Planned action:** introduce a GoogleTest or Unity-based test suite for TemperatureSensor, Eeprom, RingBuffer, and TemperatureMonitor.
2. **Non-Encapsulated Print Statements**
   - Debug printf() calls are scattered through modules and mock HAL.
   - These violate encapsulation and prevent silent operation in production.
   - **Planned action:** introduce a configurable logging interface (e.g., Log_Info(), Log_Error()) with compile-time enable/disable.
3. **Limited Error Handling**
   - Several functions return generic STATUS_EIO without detailed cause.
   - **Planned action:** extend status_t enum and propagate richer error codes up to the monitor.
4. **ISR Simulation and Timing**
   - The PC mock environment triggers ADC sampling manually; timing jitter is not emulated.
   - **Planned action:** add periodic ISR simulation via a timer thread.
5. **Missing CI/CD**
   - No GitHub actions are included.
   - **Planned action:** Add automatic build and test inside GitHub, connect with unit tests.
6. **Thread Safety**
   - The ring buffer assumes a single producer and consumer.
   - **Planned action:** document this explicitly and add runtime assertions.