

# Infant Monitoring System

본 문서는 ‘Pi 먹는 Tux’ 팀의 유아 모니터링 시스템 프로젝트를 기반으로, 하드웨어 및 소프트웨어 구조를 개선하고 기능을 확장한 프로그램의 설계 및 구현 과정을 기술한 문서입니다.

시스템은 라즈베리파이를 중심으로 온도·습도·소리 센서를 이용하여 유아의 상태를 실시간으로 감지하고, 이상 징후 발생 시 보호자에게 경고 신호를 전달하는 기능을 수행합니다.

또한 본 문서에서는 프로그램의 모듈 구조, 스레드 설계, 인터페이스 명세, 그리고 각 기능별 테스트 및 검증 결과를 포함합니다. 이를 통해 시스템의 동작 원리와 설계 의도를 명확히 전달하는 것을 목적으로 합니다.

작성자: 장동철

작성일: 2024.10.27

소속: 아주대학교 디지털미디어학과

---

# 목차

## Chapter 1 시스템 개요서

|                             |   |
|-----------------------------|---|
| 1.1. 목적 .....               | 4 |
| 1.2. 시스템 구성 개요 .....        | 4 |
| 1.3. 시스템 동작 흐름 .....        | 4 |
| 1.3.1. 센서 초기화 및 측정 .....    | 4 |
| 1.3.2. 이상 상태 판단 .....       | 4 |
| 1.3.3. 통신 및 서버 전송 .....     | 5 |
| 1.3.4. 음성 감지 연동 .....       | 5 |
| 1.3.5. 활동 지원(모터/백색소음) ..... | 5 |
| 1.4. 주요 하드웨어 구성 .....       | 5 |
| 1.5. 소프트웨어 구조 .....         | 6 |
| 1.6. 설계 의도 및 고려사항 .....     | 6 |
| 1.7. 제약사항 및 한계 .....        | 6 |
| 1.8. 결론 .....               | 7 |

## Chapter 2 모듈 설계서

|                                |    |
|--------------------------------|----|
| 2.1. 모듈 개요 .....               | 8  |
| 2.2. 주요 기능 .....               | 8  |
| 2.2.1. 센서 초기화 및 측정 루틴 수행 ..... | 8  |
| 2.2.2. 평균값 계산 및 상태 판단 .....    | 8  |
| 2.2.3. 경고 메시지 생성 및 전송 요청 ..... | 8  |
| 2.2.4. 상태 로깅 및 디버깅 지원 .....    | 8  |
| 2.3. 모듈 구성도 .....              | 9  |
| 2.4. 데이터 구조 정의 .....           | 9  |
| 2.5. 주요 함수 설명 .....            | 9  |
| 2.6. 동작 시나리오 .....             | 10 |
| 2.6.1. 초기화 단계 .....            | 10 |
| 2.6.2. 주기적 측정 및 평가 단계 .....    | 10 |
| 2.6.3. 메시지 생성 및 송신 요청 .....    | 10 |
| 2.7. 설계 의도 및 고려사항 .....        | 10 |
| 2.8. 테스트 및 검증 결과 .....         | 10 |
| 2.9. 한계 및 개선사항 .....           | 10 |
| 2.10. 결론 .....                 | 11 |

---

## Chapter 3 인터페이스 명세서

|                               |    |
|-------------------------------|----|
| 3.1. 목적 .....                 | 12 |
| 3.2. 인터페이스 개요 .....           | 12 |
| 3.3. 모듈 간 인터페이스 정의 .....      | 12 |
| 3.4. 데이터 구조 정의 .....          | 13 |
| 3.4.1. system_message_t ..... | 13 |
| 3.4.2. sensor_data_t .....    | 13 |
| 3.4.3. queue_t .....          | 13 |
| 3.5. 통신 프로토콜 정의(TCP 기반) ..... | 13 |
| 3.5.1. 메시지 포맷 .....           | 13 |
| 3.6. 스레드 간 동기화 구조 .....       | 14 |
| 3.7. 오류 및 예외 처리 .....         | 14 |
| 3.8. 타이밍 및 전송 주기 .....        | 15 |
| 3.9. 확장 고려사항 .....            | 15 |
| 3.10. 결론 .....                | 15 |

# Chapter 1.

## 시스템 개요서

### 1.1. 목적

본 문서는 라즈베리파이 기반으로 구현된 실시간 유아 건강·안전 모니터링 시스템의 전체 구조 및 동작 원리를 기술한다. 시스템의 목적은 유아의 생체 신호 및 주변 환경을 실시간으로 측정·분석하여, 이상 상황을 조기에 감지하고 보호자에게 알림을 전달하는 것이다. 이 문서는 전체 시스템의 하드웨어·소프트웨어 구성, 주요 기능 흐름, 설계 의도 및 제약사항을 포함한다.

### 1.2. 시스템 구성 개요

본 시스템은 크게 감시(Health/Safety Monitoring), 활동 지원(Activity Support), 울음 감지(Cry Detection), 통신(Network), 서버(Server) 모듈로 구성된다.

▶ 표 1.  
시스템 주요  
모듈 및 기능

| 구분    | 주요 모듈                              | 기능  |
|-------|------------------------------------|---|
| 감시    | health_monitor.c, safety_monitor.c | 환경(온도·습도), 체온, 심박, 압력, 거리 데이터를 주기적으로 측정 및 이상 감지 |
| 활동 지원 | activity_support.c                 | 야기 울음 감지 시 진동모터 및 백색소음 스피커 동작                   |
| 울음 감지 | cry_detector.py                    | 딥러닝 기반 음성 분석을 통해 울음 여부 판단                       |
| 통신    | network.c, network.h               | TCP 기반 메시지 송수신 및 재연결 관리                         |
| 서버    | main_server.c                      | 클라이언트로부터 상태 메시지를 수신하고 로그로 기록, 알림 제어             |
| 공통    | sensors.c, common.c, config.txt    | 센서 입력 출력 제어 및 설정 관리                             |

### 1.3. 시스템 동작 흐름

#### 1.3.1. 센서 초기화 및 측정

- DHT11, IR 온도센서, 심박센서(MCP3004 ADC), 압력센서, 초음파센서를 초기화한다.
- 센서 데이터는 주기적으로 읽으며, 각각의 스레드(Health/Safety Monitor)에서 분석된다.

#### 1.3.2. 이상 상태 판단

- 환경(온도·습도) 기준: config.txt의 temp\_min/max, humidity\_min/max
- 체온 기준: body\_temp\_min/max
- 심박수 기준: heart\_rate\_\*
- 압력/거리 기준: pressure\_threshold, distance\_threshold
- 각 조건을 벗어날 경우 MSG\_ALERT 메시지를 생성하여 큐에 저장한다.

### 1.3.3. 통신 및 서버 전송

- network\_client 스레드가 큐에서 메시지를 수신하여 TCP로 서버(main\_server.c)에 전송한다.
- 네트워크 연결이 끊어질 경우 자동 재연결 루프(network\_client\_reconnect\_loop)가 동작한다.

### 1.3.4. 울음 감지 연동

- cry\_detector.py는 마이크 입력을 실시간으로 분석하여 울음 확률이 0.9 이상일 경우 서버에 상태 코드를 전송한다.
- 이때 서버는 아기 울음 상태를 감지하여 Activity Support 모듈에 알림을 전달한다.

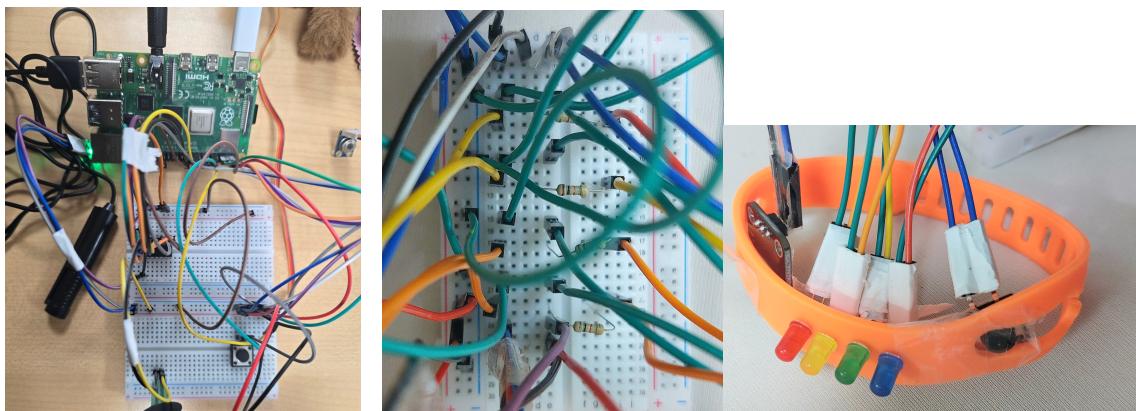
### 1.3.5. 활동 지원(모터/백색소음)

- 울음 감지 시, 화이트노이즈 재생 및 모터 진동 신호를 출력하여 아기를 안정시키도록 설계하였다.

## 1.4. 주요 하드웨어 구성

| 항목                           | 모델/인터페이스               | 용도             |
|------------------------------|------------------------|----------------|
| Raspberry Pi 4               | ARM Cortex-A72 / Linux | 시스템 제어, 스레드 실행 |
| DHT11                        | GPIO                   | 온습도 측정         |
| MLX90614 (IR Temp Sensor)    | I <sup>2</sup> C       | 체온 측정          |
| MCP3004 (ADC) + Pulse Sensor | SPI                    | 심박 측정          |
| 압력 센서                        | SPI                    | 체중/자세 감지       |
| 초음파 센서 (HC-SR04)             | GPIO                   | 거리 감지          |
| 모터 + 스피커                     | GPIO / Audio Out       | 백색소음 및 진동 출력   |
| USB 마이크                      | Audio Input            | 울음 감지 입력 장치    |

▶그림 1.  
클라이언트(좌),  
서버(중간·우)  
회로 구성



## 1.5. 소프트웨어 구조

시스템은 다음과 같은 멀티스레드 기반 구조로 동작한다.

▶표 3.  
스레드 구성  
및 주기별  
주요 기능

| 스레드명                          | 소속 모듈            | 주기     | 주요 기능             |
|-------------------------------|------------------|--------|-------------------|
| health_monitor_thread         | health_monitor   | 2ms 루프 | 온도, 습도, 체온, 심박 측정 |
| safety_monitor_thread         | safety_monitor   | 100ms  | 압력 및 거리 감시        |
| network_client_reconnect_loop | network          | 해당 없음  | TCP 연결 상태 유지      |
| cry_detector                  | cry_detector.py  | 실시간    | 음성 분석 및 울음 감지     |
| activity_support_thread       | activity_support | 이벤트 기반 | 백색소음 스피커, 모터 제어   |

## 1.6. 설계 의도 및 고려사항

- 모듈화: 각 기능을 독립적인 모듈로 분리하여 유지보수성을 확보하였다.
- 실시간성 보장: 2ms 단위의 샘플링 주기를 통해 심박 변화 등 빠른 신호에도 대응 가능하도록 설계하였다.
- 안정성 확보: 각 센서 오류 발생 시 로그를 남기고, 재시도 혹은 무시 처리를 수행하여 시스템 전체 중단을 방지하였다.
- 유연한 설정 관리: config.txt 파일로 각 임계값을 외부에서 수정 가능하게 하였다.
- 확장성: 추가 센서 및 제어 모듈을 쉽게 연동할 수 있도록 공통 인터페이스(sensors.c/h) 구조로 구현하였다.

## 1.7. 제약사항 및 한계

- 정확도: 비의료용 센서를 사용하므로 절대적인 측정 정확도는 제한적이다.
- 환경 의존성: 온습도 및 적외선 센서가 외부 온도, 조명 환경에 따라 영향을 받을 수 있다.

- 
- 모델 성능: Cry Detector는 사전 학습된 모델(baby\_cry\_model.keras)에 의존하며, 다양한 음성 데이터에 대한 일반화는 제한적이다.
  - 자원 사용량: 멀티스레드 구조로 인한 CPU 부하가 발생하므로 저사양 기기에서는 성능 저하 가능성이 있다.

## 1.8. 결론

본 시스템은 아기의 건강과 안전 상태를 센서 데이터 + 음성 데이터 기반으로 통합 감시하는 구조로 설계되었다.

각 모듈은 실제 하드웨어에서 독립적으로 동작하며, 실시간 분석 및 경보 전달 기능을 수행한다.

본 문서는 전체 시스템의 구조적 이해와 후속 모듈 설계서 작성의 기반이 된다.

---

## Chapter 2

### 모듈 설계서

#### 2.1. 모듈 개요

health\_monitor 모듈은 아기의 생체 및 환경 상태를 주기적으로 측정하고, 기준 범위를 벗어날 경우 경고 메시지를 생성하는 역할을 수행한다.

센서 데이터를 기반으로 실시간 분석을 수행하며, 결과는 system\_message\_t 구조체를 통해 상위 모듈(network 또는 main\_client)로 전달된다.

이 모듈은 시스템의 “건강 감시” 핵심 역할을 담당한다.

#### 2.2. 주요 기능

##### 2.2.1. 센서 초기화 및 측정 루틴 수행

- DHT11(온·습도), MLX90614(IR 체온), MCP3004(심박) 센서를 초기화하고 2ms 단위로 측정 루프를 수행한다.
- 각 센서의 읽기 실패 시, 로그를 기록하고 일정 횟수 재시도 후 무시 처리한다.

##### 2.2.2. 평균값 계산 및 상태 판단

- 최근 측정된 온도, 습도, 체온, 심박 데이터를 버퍼에 저장하고, 지정된 샘플 수 기준으로 평균을 계산한다.
- config.txt에 정의된 임계값(temp\_min/max, humidity\_min/max, heart\_rate\_min/max, body\_temp\_min/max)을 기준으로 상태를 판별한다.

##### 2.2.3. 경고 메시지 생성 및 전송 요청

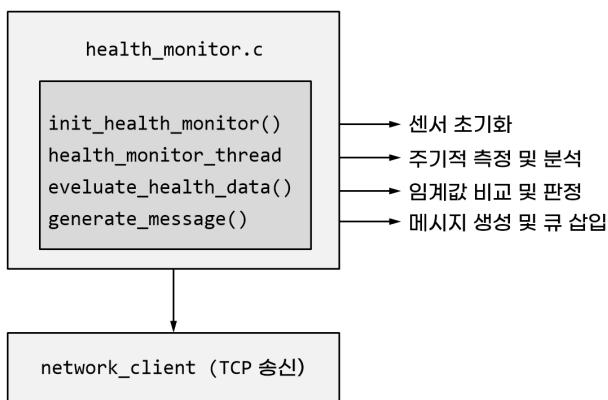
- 이상 상태가 발생하면 MSG\_ALERT 타입 메시지를 구성하여 queue\_push() 함수를 통해 송신 큐에 등록한다.
- 정상 상태일 경우 주기적으로 MSG\_STATUS 메시지를 생성하여 현재 상태를 보고한다.

##### 2.2.4. 상태 로깅 및 디버깅 지원

- 각 측정 주기마다 LOG\_INFO 매크로를 통해 콘솔 출력 및 파일 로그를 남긴다.
- 로그에는 센서별 측정값, 판정 결과, 메시지 전송 여부가 포함된다.

## 2.3. 모듈 구성도

▶그림 2.  
건강 모니터링  
모듈의 함수  
구성 및 TCP  
송신 흐름



## 2.4. 데이터 구조 정의

▶표 4.  
health\_monitor  
모듈에서  
사용되는 주요  
데이터 구조체  
정의

| 구조체명             | 정의 위치            | 주요 필드  | 설명              |
|------------------|------------------|--|-----------------|
| sensor_data_t    | common.h         | temperature, humidity, heart_rate, body_temp   | 실시간 센서 측정값 저장   |
| system_message_t | common.h         | type, timestamp, data, message_text            | 네트워크 전송용 상태 메시지 |
| threshold_t      | common.h         | min, max                                       | 임계값 관리용 구조체     |
| health_state_t   | health_monitor.h | HEALTH_NORMAL, HEALTH_WARNING, HEALTH_CRITICAL | 상태 코드 정의        |

## 2.5. 주요 함수 설명

▶표 5.  
health\_monitor  
모듈의 주요  
함수 및 역할

| 함수명  | 설명   |
|--|--|
| init_health_monitor()                                      | 센서 드라이버 초기화, 측정 스레드 생성                           |
| health_monitor_thread()                                    | 메인 루프 함수. 2ms 단위로 센서 데이터를 측정하고, 일정 주기마다 상태 평가 수행 |
| evaluate_health_data()                                     | 평균값 기반으로 현재 상태를 평가하고, 경고 조건 충족 시 메시지를 생성         |
| generate_message()   | 평가 결과를 문자열 형태로 변환하여 system_message_t 구조체 생성      |
| push_message_queue()                                       | 생성된 메시지를 송신 큐에 삽입                                |
| read_dht_sensor(), read_temp_sensor(), read_heart_sensor() | 각 센서의 데이터 읽기 및 예외 처리 담당                          |

## 2.6. 동작 시나리오

### 2.6.1. 초기화 단계

- init\_health\_monitor()에서 각 센서 초기화 수행.
- 실패 시 로그 출력 후 재시도(최대 3회).

### 2.6.2. 주기적 측정 및 평가 단계

- 스레드 내에서 2ms 단위의 루프 실행.
- 내부 카운터를 통해 5초/10초 단위 평균값 계산 및 상태 평가 수행.
- 평가 결과가 임계 범위를 벗어나면 경고 메시지 생성.

### 2.6.3. 메시지 생성 및 송신 요청

- system\_message\_t 구조체를 구성하고, queue\_push()를 통해 송신 큐에 등록.
- 네트워크 모듈에서 큐를 읽어 TCP로 서버에 전송.

## 2.7. 설계 의도 및 고려사항

- 정확성 확보: 단일 샘플이 아닌 평균값 기반 판단으로 센서 노이즈 영향을 최소화하였다.
- 비동기 처리: 센서 측정과 네트워크 송신을 별도 스레드로 분리하여 병목을 방지하였다.
- 유연한 임계값 관리: config.txt에서 값 수정만으로 측정 기준을 조정 가능하도록 하였다.
- 오류 복원력: 센서 오류 시 시스템 전체가 중단되지 않도록 독립적 예외 처리를 적용하였다.

## 2.8. 테스트 및 검증 결과

▶ 표 6.  
*health\_monitor*  
모듈 테스트 및  
검증 결과 요약

| 항목             | 테스트 조건              | 결과                  |
|----------------|---------------------|---------------------|
| 온도 30°C 이상     | 환경 시뮬레이터로 가열        | “온도 경고” 메시지 송신 확인   |
| 심박수 160 BPM 초과 | 입력 시뮬레이터 신호 주입      | 경고 메시지 정상 생성        |
| 센서 비연결 상태      | DHT11 제거            | 오류 로그 출력 및 루프 지속 확인 |
| 설정값 변경         | config.txt 수정 후 재시작 | 새로운 임계값 정상 반영       |

## 2.9. 한계 및 개선사항

- DHT11의 응답 지연으로 인한 간헐적 데이터 누락 발생 가능.
- 체온센서(MLX90614)의 측정 거리에 따른 오차 존재.
- 향후 버전에서는 이동 평균 기반 필터 적용 및 센서 보정 기능 추가 예정.

---

## 2.10. 결론

health\_monitor 모듈은 본 시스템의 핵심 감시 기능을 담당하며, 센서 데이터의 안정적 수집·평가·보고를 수행한다. 설계는 단순화되어 있으나, 모듈 분리와 임계값 기반 판단 구조 덕분에 유지보수성이 높다. 본 문서는 향후 safety\_monitor 및 activity\_support 모듈 설계 문서 작성의 기준으로 활용될 수 있다.

# Chapter 3

## 인터페이스 명세서

### 3.1. 목적

본 문서는 시스템 내 각 모듈 간, 그리고 클라이언트-서버 간의 데이터 교환 구조와 프로토콜을 정의한다.

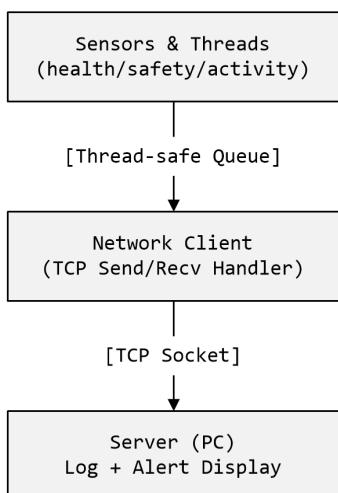
인터페이스 명세는 시스템의 일관된 데이터 전달을 보장하고, 유지보수 및 확장 시 참조 가능한 기준을 제공하기 위해 작성되었다.

### 3.2. 인터페이스 개요

전체 시스템은 클라이언트(라즈베리파이)와 서버(모니터링 장치-팔찌) 간의 TCP 통신을 기반으로 동작한다.

클라이언트 내부에서는 센서 측정 스레드, 분석 스레드, 네트워크 송신 스레드가 메시지 큐를 통해 비동기적으로 연결된다.

▶그림 3.  
시스템  
아키텍처



### 3.3. 모듈 간 인터페이스 정의

▶표 7.  
주요 모듈 간  
인터페이스  
정의

| 송신 모듈            | 수신 모듈   | 매개 방식             | 데이터 구조           | 설명              |
|------------------|---------|-------------------|------------------|-----------------|
| health_monitor   | network | Thread-safe Queue | system_message_t | 건강 상태 메시지 전송    |
| safety_monitor   | network | Thread-safe Queue | system_message_t | 안전 상태 메시지 전송    |
| activity_support | network | Thread-safe Queue | system_message_t | 지원 상태(모터/소리) 보고 |
| cry_detector.py  | network | TCP Socket        | JSON String      | 울음 감지 결과 전송     |

|             |                        |            |                             |               |
|-------------|------------------------|------------|-----------------------------|---------------|
| network     | main_server            | TCP Socket | Serialized system_message_t | 서버로 상태 보고     |
| main_server | 로컬 모듈<br>(log/display) | 함수 호출      | 문자열                         | 로그 기록 및 알람 표시 |

### 3.4. 데이터 구조 정의

#### 3.4.1. system\_message\_t

▶ 표 8.  
system\_message\_t 구조체  
필드 정의

| 필드명          | 타입            | 설명  |
|--------------|---------------|---|
| type         | int           | 메시지 유형 (MSG_STATUS, MSG_ALERT, MSG_WARNING) |
| timestamp    | char[32]      | 송신 시각 (YYYY-MM-DD HH:MM:SS)                 |
| data         | sensor_data_t | 센서 측정 데이터 (온도, 습도, 심박, 체온 등)                |
| message_text | char[256]     | 사람이 읽을 수 있는 상태 메시지 (예: “심박수 경고: 165 BPM”)   |

#### 3.4.2. sensor\_data\_t

▶ 표 9.  
sensor\_data\_t  
구조체 필드  
정의

| 필드명         | 타입    | 설명        |
|-------------|-------|-----------|
| temperature | float | 환경 온도(°C) |
| humidity    | float | 환경 습도(%)  |
| body_temp   | float | 체온(°C)    |
| heart_rate  | float | 심박수(BPM)  |

#### 3.4.3. queue\_t

▶ 표 10.  
queue\_t  
구조체 필드  
정의

| 필드명    | 타입                 | 설명            |
|--------|--------------------|---------------|
| buffer | system_message_t[] | 메시지 버퍼        |
| front  | int                | 큐의 첫 인덱스      |
| rear   | int                | 큐의 마지막 인덱스    |
| count  | int                | 현재 저장된 메시지 수  |
| mutex  | pthread_mutex_t    | 동시 접근 제어용 뮤텍스 |
| cond   | pthread_cond_t     | 큐 비/포화 상태 신호  |

### 3.5. 통신 프로토콜 정의 (TCP 기반)

#### 3.5.1. 메시지 포맷 (클라이언트 → 서버)

### <HEADER> + <BODY>

HEADER (고정 길이 8바이트)

▶ 표 11.  
TCP 통신  
메시지 헤더  
구조

| 필드명    | 크기 | 설명                  |
|--------|----|---------------------|
| type   | 4B | 메시지 타입 코드           |
| length | 4B | BODY의 전체 길이 (bytes) |

BODY (가변길이)

▶ 표 12.  
클라이언트  
송신 메시지  
예시 포맷

| 항목           | 형식  | 예示                    |
|--------------|-----|-----------------------|
| timestamp    | 문자열 | “2025-10-21 10:05:22” |
| message_text | 문자열 | “심박수 경고: 162 BPM”     |
| temperature  | 실수  | 26.3                  |
| humidity     | 실수  | 45.2                  |
| heart_rate   | 실수  | 162.0                 |

예시 송신 문자열

[TYPE=2][LEN=85]2025-10-21 10:05:22 심박수 경고: 162 BPM TEMP=26.3 HUM=45.2 HR=162.0

## 3.6. 스레드 간 동기화 구조

▶ 표 13.  
스레드 간  
동기화 방식  
및 공유 차원

| 대상 스레드   | 공유 자원     | 동기화 방식                              | 비고                        |
|--|-----------|-------------------------------------|---------------------------|
| health_monitor_thread<br>& network_client_thread | 메시지 큐     | pthread_mutex_t +<br>pthread_cond_t | 큐에 메시지 추가/삭제 시<br>Lock 적용 |
| safety_monitor_thread<br>& network_client_thread | 동일 큐      | 조건변수 기반 비동기<br>처리                   |                           |
| cry_detector.py                                  | 독립 TCP 연결 | 비동기 송신 (Lock 없음)                    |                           |

## 3.7. 오류 및 예외 처리

▶ 표 14.  
주요 오류  
원인 및 예외  
처리 방법

| 구분        | 발생 원인           | 처리 방식   |
|-----------|-----------------|---|
| TCP 연결 끊김 | 서버 종료, 네트워크 단절  | 자동 재연결 루프 수행<br>(network_client_reconnect_loop) |
| 큐 포화 상태   | 송신 지연 또는 서버 미응답 | 메시지 삭제 후 Warning 로그 출력                          |
| 센서 비정상값   | 측정 실패 또는 오류     | 로그 기록 후 값 무시 처리                                 |

|           |            |                |
|-----------|------------|----------------|
| 데이터 포맷 오류 | 잘못된 문자열 수신 | 메시지 무시 후 연결 유지 |
|-----------|------------|----------------|

### 3.8. 타이밍 및 전송 주기

▶ 표 15.  
메시지 유형별  
발생 조건 및  
전송 주기

| 메시지 유형      | 발생 조건  | 전송 주기  | 비고       |
|-------------|--------|--------|----------|
| MSG_STATUS  | 정상 상태  | 10초 간격 | 상태 보고    |
| MSG_ALERT   | 임계값 초과 | 즉시     | 서버 경보 표시 |
| MSG_WARNING | 잠재적 이상 | 5초     | 모니터링용    |

### 3.9. 확장 고려사항

- 새로운 센서 추가 시 sensor\_data\_t 구조체에 필드 확장 후 동일 포맷으로 전송 가능
- 메시지 포맷은 JSON 직렬화 방식으로 변경해도 무관 (현재는 단순 텍스트)
- 다중 클라이언트 확장 시 서버는 select() 기반 소켓 처리로 병렬 수신 가능

### 3.10. 결론

본 인터페이스 명세서는 클라이언트-서버 간 및 내부 스레드 간 데이터 교환 구조를 정의한다. TCP 기반의 단순 구조를 유지하면서도, 멀티스레드 환경에서의 안전한 데이터 전달을 보장한다. 향후 센서 종류 및 전송 포맷 확장 시, 본 문서를 기준으로 수정 및 버전 관리를 수행한다.