1. 문서 개요

1.1 목적

본 문서는 ESP32-C3 기반 환경 센서 모니터링 시스템의 기능 및 인터페이스를 정의한 표준 명세서이다.
센서 노드(ESP32), MQTT 브로커, 웹 대시보드(index.html) 간의 동작 및 데이터 형식을 명확히 하는 것을 목적으로 한다.

1.2 범위

- 하드웨어
    - ESP32-C3
    - SCD4x 센서 (CO_2 / 온도 / 습도)
    - 상태 LED, AP 버튼
- 네트워크
    - Wi-Fi STA/AP 모드
    - MQTT 브로커 (HiveMQ Demo: broker.hivemq.com)
- 소프트웨어
    - ESP32 펌웨어 파일: `ESP32_SCD_LED_AP_Reset_Table_Graph_Mem.ino`
    - 웹 대시보드 파일: `index.html` (PC에서 로컬 실행)

---

2. 시스템 구성

2.1 전체 구조

1. 센서 노드 (ESP32-C3)
    - SCD4x 센서에서 CO_2, 온도, 습도 데이터를 주기적으로 측정
    - 일사량 값은 현재 랜덤 정수값으로 생성 (향후 실제 센서로 대체 가능)
    - 측정 데이터 및 상태, 알람, 히스토리 정보를 MQTT 브로커로 송신
2. MQTT 브로커
    - 서버: `broker.hivemq.com`
    - 역할: ESP32와 웹 대시보드 간의 메시지 중계
3. 웹 대시보드 (index.html)
    - PC에서 로컬로 HTML 파일을 브라우저로 실행
    - HiveMQ WebSocket (`wss://broker.hivemq.com:8884/mqtt`)을 통해 MQTT 연결
    - 실시간 상태 표시, 1분 평균 히스토리(테이블/그래프), 알람 설정, LED/RESET 제어 기능 제공

---

3. 하드웨어 및 네트워크 설정

3.1 하드웨어 구성

- MCU: ESP32-C3
- 센서: SCD4x (CO_2, Temperature, Humidity)
- LED: 상태 및 통신 표시용 (디지털 출력, 예: PIN 10)
- 버튼: AP 모드 진입용 (디지털 입력, 예: PIN 9, 3초 이상 눌림 감지)

- 전원: USB 또는 외부 5V

3.2 기본 네트워크 설정

- 기본 STA 모드 Wi-Fi 설정
    - SSID: `Backhome`
    - Password: `1700note`
- MQTT 브로커 설정
    - Host: `broker.hivemq.com`
    - Port (ESP32): `1883` (TCP)
    - WebSocket (웹 대시보드): `wss://broker.hivemq.com:8884/mqtt`

---

4. ESP32 펌웨어 기능 명세

4.1 부팅 및 동작 모드

1. AP 버튼 체크(부팅 시)
    - 전원 인가 후 약 3초 동안 AP 버튼(PIN 9) 상태를 확인
    - 3초 동안 LOW 상태 유지 시: AP 모드로 진입
    - 그렇지 않을 경우: STA 모드로 동작
2. STA 모드
    - 저장된 Wi-Fi SSID/비밀번호(Preferences)로 라우터 접속 시도
    - 접속 성공 시:
        - MQTT 브로커에 연결
        - 센서 측정 및 상태 전송 시작

- 접속 실패 시:
    - 펌웨어 로직에 따라 재시도 또는 AP 모드로 전환

3. AP 모드
    - SoftAP 개설 (예: SSID `ESP32_Config`, 고정 Password)
    - 내장 WebServer로 Wi-Fi 설정 페이지 제공
    - 사용자가 웹 페이지를 통해 SSID/비밀번호를 입력 → Preferences에 저장 → 재부팅 후 STA 모드로 재시도

4.2 센서 측정 및 **5**초 주기 전송

1. 측정 주기
    - 기준 주기: 5초
    - `millis()` 기반 타이머를 사용하여 정확히 5초마다 측정 수행
2. 측정 항목
    - CO_2 (ppm): SCD4x 센서
    - 온도 (°C): SCD4x 센서
    - 습도 (%): SCD4x 센서
    - 일사량: 정수 랜덤값 (예: 0 ~ 800, 현재 모의 값)
    - RSSI (dBm): `WiFi.RSSI()`
    - LED 상태: 현재 LED 핀의 On/Off 상태
3. MQTT 전송 포맷 (`esp32c3/status`)
    - 토픽: `esp32c3/status`
    - 전송 주기: 5초마다
    - retain: `false` (실시간 상태용)

JSON 예시:

```
{
  "type": "status",
  "temp": 24.75,
  "hum": 43.5,
  "co2": 930,
  "solar": 256,
  "rssi": -61,
  "led": false
}
```

        -

4. 통신 시 LED 표시
    - MQTT 송신 시 LED를 약 80ms 동안 반전하여 Blink
    - Blink 후 원래 상태로 복귀 (통신 activity 표시)

4.3 1분 평균 히스토리 관리

1. 집계 방식
    - 내부 집계 구조체(MinuteAgg)를 사용하여 1분 동안의 샘플을 누적
        - tempSum, humSum, co2Sum, solarSum
        - rssiSum, sampleCount, rssiCnt
        - lastLed (마지막 LED 상태), startMs (집계 시작 시각)
2. 집계 처리
    - 5초마다 측정된 status 값을 MinuteAgg에 누적
    - 1분이 경과할 때마다:
        - 누적값 / sampleCount로 평균 계산
        - 평균 결과를 HistRecord 구조체에 저장
3. 히스토리 버퍼

- 최대 저장 개수: 60개 (최근 60분)
- 초과 시 FIFO 방식으로 가장 오래된 레코드를 삭제 (Rolling Buffer)
- HistRecord 필드:
  - idx (순번)
  - temp, hum, co2, solar, rssi, led
  - valid (유효 여부)

4. NVS(Preferences) 저장
   - 일정 조건(예: 새 히스토리 추가 시)마다 전체 히스토리를 JSON 배열로 직렬화
   - Preferences의 `"history"` 네임스페이스에 String 형태로 저장
   - 재부팅 시:
     - JSON을 역직렬화하여 historyBuf를 복원
     - historyCount, historyNextIdx 재계산
5. 히스토리 MQTT 전송 (`esp32c3/history`)
   - 브라우저의 요청 또는 MQTT 재연결 시, 저장된 히스토리 전송
   - 토픽: `esp32c3/history` (또는 세분화된 `esp32c3/history/...` 구조)
   - retain: `true` (새로 접속한 클라이언트가 최신 히스토리를 즉시 수신하도록 하기 위함)

JSON 예시:

```
{
  "type": "history",
  "data": [
    {
      "idx": 0,
      "temp": 24.7,
      "hum": 43.1,
      "co2": 930,
      "solar": 300,
```

```
      "rssi": -60,
      "led": false
    }
    // ... 최대 60개
  ]
}
```

4.4 알람 설정 관리

1. 알람 범위 구조 (AlarmRange)
    - 항목별 구조체: temp, hum, co2, solar 각각 사용
    - 필드:
        1. `hasLow` (bool), `low` (float)
        2. `hasHigh` (bool), `high` (float)
    - 하한/상한 둘 다 옵션이며, 사용하지 않을 경우 `hasLow` 또는 `hasHigh`를 false로 설정
2. 저장 위치 (Preferences)
    - 네임스페이스: `"alarm"`
    - 키 예시:
        1. `"temp_low"`, `"temp_high"`
        2. `"hum_low"`, `"hum_high"`
        3. `"co2_low"`, `"co2_high"`
        4. `"solar_low"`, `"solar_high"`
3. MQTT 인터페이스
    - 토픽: `esp32c3/config/alarm`

웹 → ESP32 수신 JSON 포맷:

```
{
  "type": "alarm_config",
  "temp":  { "low": 10, "high": 30 },
  "hum":   { "low": 20, "high": 80 },
  "co2":   { "low": 400, "high": 1000 },
  "solar": { "low": 10, "high": 500 }
}
```

-
- 처리 순서:
    1. JSON 파싱 후 AlarmRange 구조체에 반영
    2. Preferences에 저장
    3. 현재 알람 설정을 동일 토픽(`esp32c3/config/alarm`)으로 다시 Publish
       (브라우저와 설정 동기화를 위함)
- ESP32 → 웹 전송 시 retain: `true`
  (새로 연결된 웹 대시보드가 즉시 설정값을 수신)

4.5 제어 기능 **(LED /** 소프트 리셋**)**

1. LED 제어
    - 구독 토픽: `esp32c3/led`
    - Payload:
        1. `"LED_ON"` → 상태 LED HIGH
        2. `"LED_OFF"` → 상태 LED LOW
2. 소프트 리셋
    - 구독 토픽: `esp32c3/reset`
    - Payload: `"SOFT_RESET"`

- 동작:
    1. LED를 짧은 주기로 여러 번(예: 5회) 빠르게 깜빡임
    2. `ESP.restart()` 호출하여 소프트웨어 리셋 수행

4.6 MQTT 재연결 및 예외 처리

1. MQTT 연결 실패 시
    - 연결 실패 로그 출력
    - 일정 시간(예: 2초) 지연 후 재연결 시도 루프
    - 연결 성공 시:
        - 필요한 토픽(led, reset, config/alarm, history/get 등) 재구독
        - 알람 설정 및 히스토리 재전송
2. Wi-Fi 끊김 처리
    - STA 모드에서 Wi-Fi 상태 감시
    - 끊긴 경우 재연결 시도
    - 상황에 따라 AP 모드 진입 로직 적용 가능

---

5. 웹 대시보드 (index.html) 기능 명세

5.1 MQTT 연결

- WebSocket 주소: `wss://broker.hivemq.com:8884/mqtt`
- MQTT 클라이언트:
    - 랜덤 또는 접두어 기반 클라이언트 ID 사용

- 구독 토픽:
    - `esp32c3/status`
    - `esp32c3/config/alarm`
    - `esp32c3/history/#`
- Publish 토픽:
    - `esp32c3/led`
    - `esp32c3/reset`
    - `esp32c3/config/alarm`
    - 필요 시 `esp32c3/history/get`

5.2 실시간 상태 표시

1. 상태 카드
    - 표시 항목:
        - 온도, 습도, CO_2, 일사량
        - LED 상태 (ON/OFF)
        - RSSI
        - 장비 온라인 여부 (최근 status 수신 시간 기준)
    - 알람 처리:
        - 특정 항목 값이 알람 범위를 벗어날 경우 카드 또는 값에 강조 색(예: 빨간색) 적용
2. MQTT 상태 표시
    - 상단 영역에서 다음 정보 표시:

        - MQTT 연결 여부 (아이콘 및 텍스트)
        - 마지막 수신 시간
        - 브로커 호스트 정보 등

5.3 히스토리 테이블 및 그래프

1. 히스토리 데이터 구조
    - 자바스크립트 배열 `historyBuckets[]` 사용
    - 최대 60개 레코드 저장
    - 각 레코드 항목:
        - 시간(분 기반), temp, hum, co2, solar, rssi, led, index 등
2. 테이블 모드
    - 1분 평균 데이터를 행 단위로 표시
    - 열 구성:
        - 시간(HH:MM)
        - 온도
        - 습도
        - CO_2
        - 일사량
        - LED 상태
        - RSSI
    - 최신 데이터가 상단에 표시
3. 그래프 모드
    - Chart.js 등을 사용한 라인 그래프

- 그래프 종류:
  - 온도 그래프
  - 습도 그래프
  - CO_2 그래프
  - 일사량 그래프
- X축: 시간(HH:MM), Y축: 각 항목 단위

4. 브라우저 캐시 (옵션)
   - `localStorage`를 사용하여 수신한 히스토리 캐시 저장
   - 페이지 새로고침 시 캐시를 먼저 로드하여 동일 데이터 표시
   - ESP32 재부팅 전까지는 MQTT retained 메시지와 캐시를 조합해 일관된 히스토리 제공 가능

5.4 제어 및 설정 **UI**

1. LED 제어
   - ON/OFF 버튼 제공
   - 클릭 시 `esp32c3/led` 토픽에 `"LED_ON"` 또는 `"LED_OFF"` 전송
   - 전송 결과를 UI 로그에 표시
2. 소프트 리셋
   - RESET 버튼 제공
   - 클릭 시 확인 팝업 표시 후 `"SOFT_RESET"` 메시지 전송
3. 알람 설정
   - 센서 카드 클릭 시 알람 설정 모달 창 표시
   - 각 항목(온도, 습도, CO_2, 일사량)에 대해 하한/상한 입력 가능
   - 저장 버튼 클릭 시 `esp32c3/config/alarm` 토픽에 JSON 전송

- ESP32에서 재전송한 알람 설정을 수신하여 UI와 동기화

5.5 로그 관리

- MQTT 송수신 이벤트를 하단 로그 영역에 출력
- 로그 최대 개수를 제한(예: 300~400줄)하여 메모리 과사용 방지
- 로그 영역 표시/숨김 토글 버튼 제공

---

6. MQTT 토픽 및 메시지 포맷 요약

구분	토픽	방향	설명	**retain**
상태 전송	`esp32c3/status`	ESP32 → 웹	5초 주기 센서/LED/RSSI 상태	false
히스토리	`esp32c3/history` (또는 하위)	ESP32 → 웹	1분 평균 이력 (최대 60개)	true
알람 설정	`esp32c3/config/alarm`	웹 ↔ ESP32	알람 범위 설정 및 동기화	true
LED 제어	`esp32c3/led`	웹 → ESP32	`"LED_ON"` / `"LED_OFF"` 제어	false
소프트 리셋	`esp32c3/reset`	웹 → ESP32	`"SOFT_RESET"` 요청	false
히스토리 요청	`esp32c3/history/get` (옵션)	웹 → ESP32	히스토리 재전송 요청	false

---

7. 향후 확장 항목 (참고)

1. MQTT 또는 HTTP 기반 펌웨어 OTA 업데이트 기능 추가