8. 소프트웨어/펌웨어 설계

8.1 전체 제어 로직 구조

본 장비는 **양측 컨베이어 + 펌프/배관 + 저장 탱크 + 주행부 + 센서 네트워크 + UI + 원격통신 + BMS**로 이루어져 있으며,
전체 제어 로직은 다음과 같은 **모듈형/상태머신 기반 구조**로 설계한다.

---

1) 상위 구조 개요(Top-Level Architecture)

전체 로직은 아래 5개의 대분류 모듈로 구성된다.

1. **상태 관리(State Machine Manager)**
2. **컨베이어·흡입·유입 모듈(Infeed & Conveyor Control)**
3. **폐유 저장/처리 모듈(Oil Handling & Tank System)**
4. **주행 및 플랫폼 이동 제어(Drive Control)**
5. **안전·보호·검사 모듈(Safety Manager)**
6. **UI/통신 모듈(User Interface & Connectivity)**
7. **전원/배터리 관리(Power & BMS Control)**

---

2) 메인 루프 구조(Main Loop Architecture)

① RTOS 기반 구조(권장: FreeRTOS)

```
Main()
 ├─ initHardware()
 ├─ initTasks()
 └─ vTaskStartScheduler()

<RTOS Tasks>

1) StateManagerTask (100ms)
2) ConveyorTask (20ms)
3) PumpTask (20ms)
4) SensorAcquisitionTask (50~100ms)
5) DriveControlTask (10~20ms)
6) SafetyMonitorTask (10ms)
7) UItask (100ms)
8) CommunicationTask (500ms)
9) BMSTask (1s)
```

---

3) 전체 시스템 상태(Top-Level States)

1. IDLE (대기)

- 모든 구동계 OFF
- UI는 Ready 표시
- 컨베이어/펌프 부하 검사만 수행
- 배터리 상태 모니터링 유지

2. PRE-CHECK (사전 점검)

- 탱크 잔량 체크
- 필터 막힘센서/압력센서 확인
- 컨베이어 양쪽 토크 체크
- 모터/펌프 이상 여부 판독
- 문제가 없을 때만 RUN 가능

3. RUNNING (수거 동작)

- 좌/우 컨베이어 가동
- 유입 트레이 → 1차 필터 → 유압/배관 → 저장탱크로 이송
- 실시간 센서 기반 유입량 제어
- 탱크 레벨 상승 속도를 모니터링
- 이물질 감지 시 정지/역회전

4. FULL (저장탱크 만수)

- 펌프 감속 또는 정지
- 컨베이어 대기 상태
- UI 및 원격에서 “탱크 만수" 알림
- 배출 또는 교체 필요

5. FAULT (고장 회피 모드)

- 즉시 컨베이어·펌프·모터 중지
- BMS 보호 개입 시도
- UI 경고 + 원격 경고
- 원인에 따라 자동 복귀 or 강제 Reset 필요

---

4) 주요 제어 모듈 상세 구조

(1) 컨베이어 제어(Conveyor Control Module)

A. 입력 요소

- 좌/우 모터 전류센서 (부하 감지)
- 회전속도(Encoder)
- 이물질 감지(압력 증가/IR센서)
- 경사각 센서

B. 제어 로직

1. 속도 PID 제어
2. 좌우 동시운전 Sync Mode
3. 이물질 감지 → 즉시 정지 + 역회전
4. 유입량이 과다하면 속도 제한
5. 경사 보정(높낮이 조절된 상태에서도 일정 유입 유지)

---

(2) 펌프/배관 제어(Pump & Piping Module)

A. 입력 요소

- 펌프 전류
- 유량센서
- 배관 압력센서
- 탱크 레벨

B. 제어 로직

1. 목표 유량 유지 위한 속도 제어
2. 압력 급상승(막힘) 시 펌프 정지
3. 탱크 만수 근접 시 펌프 감속
4. 배출 모드 전환 시 배관 스위칭
5. 유입량 실시간 매칭(컨베이어와 연동)

---

(3) 탱크 시스템 제어(Tank Handling Module)

A. 입력 요소

- 초음파/부력/차압식 레벨 센서
- 온도센서
- 히터/냉각기 상태
- 탱크 압력(옵션)

B. 제어 로직

1. 레벨 기반 탱크 만수 판단
2. 점도 유지 위한 히터/쿨러 PID 제어
3. 침전탱크는 시간 기반 Settling Profile 운용
4. 온도 과상승 시 히터 차단
5. 배출 모드 제어(드레인 밸브)

---

(4) 주행/프레임 제어(Drive Control Module)

- 안정적인 이동을 위한 속도 제어
- 부하 증가 시 자동 감속
- 경사각 센서 기반 기울기 보정
- 모터 토크 과다 시 경고 및 주행 제한

(도로주행 없음 → 실내/실외 저속 주행만)

---

(5) 안전 관리(Safety Manager)

항상 실시간으로 동작하는 가장 높은 우선순위 Task.

감시 항목

- 비상정지 스위치
- 배터리 과전류/과열
- 컨베이어 응답 없음
- 펌프 압력 과상승
- 탱크 오버플로
- 통신 두절
- MCU 온도 상승

행동

- 즉각 동작 차단
- FAULT 상태로 전환
- UI 및 원격에 에러코드 전송

---

(6) 센서 획득 모듈(Sensor Acquisition)

- ADC 센서(온도·압력·전류) 필터링
- 레벨 센서 보정
- 유량센서 펄스 계산

- 컨베이어 모터 RPM 측정
- 다중센서 평균값/이상값 제거

---

(7) UI/통신 모듈(UI & Connectivity)

UI 표시 요소

- 컨베이어 상태
- 펌프 속도
- 탱크 레벨
- 점도 유지 온도
- 배터리 SoC
- Fault 코드

통신 방식

- BLE 단거리
- Wi-Fi 현장 모니터링
- LTE/5G 원격 알람(옵션)

---

(8) 전원·BMS 모듈(Power/BMS Control)

- SoC 기반 운전 시간 계산
- 출력 가능한 최대 부하 제어
- 충전/방전 가능 여부 제어
- BMS 경고 발생 시 속도 제한 또는 긴급 정지

---

5) 전체 데이터 흐름(Data Flow)

```
[센서] → SensorTask → StateManager → 각 제어모듈
                   ↑               ↓
            SafetyTask ← BMS/전원 ← BMSTask
                   ↓
                UI/통신
```

- SafetyTask는 모든 데이터 스트림을 가로지르는 **Parallel Monitor** 역할
- StateManager는 **전체 동작 상태의 뇌(Brain)** 역할

---

6) 예시: RUNNING 상태에서의 실시간 흐름

1. SensorTask
   → 유량/압력/레벨/온도/모터부하 읽음
2. ConveyorTask
   → 양측 모터 속도 동기, 이물질 체크
3. PumpTask
   → 유입량과 유량 매칭 제어
4. StateManager
   → 조건 충족 여부 판단
   → FULL/FAULT 여부 판정
5. SafetyTask
   → 급상승 신호 감지 시 즉시 STOP
6. UI
   → 실시간 그래프/수치 업데이트
7. BMS
   → 배터리 상태에 따라 최대 출력 제한

8.2 컨베이어 속도 제어 소프트 스타트 알고리즘

냉장고 크기의 이동식 폐식용유 수거 장비에서 컨베이어는 **폐유·부유물·이물질이 혼합된 medium**을 다루므로,
기계적 스트레스와 초기 부하를 최소화하기 위해 **Soft-Start(점진 가속)** 방식이 필수적이다.

본 절에서는 **모터 구동 시 순간 토크 급상승을 방지하고, 이물질과 과부하 발생을 조기에 탐지**하는 소프트 스타트 알고리즘을 정의한다.

---

1) 설계 목적

1. **모터·기어·컨베이어 구조에 가해지는 Shock Load 감소**
2. **전류 돌입(Inrush Current) 최소화 → 배터리 보호**
3. **첫 구동 시 점도 높은 폐식용유 때문에 발생하는 초기 부하 완화**
4. **토크 급증을 상시 감지하여 이물질/막힘 조기 차단**
5. **좌/우 컨베이어가 동시에 기동할 때 속도 동기(Sync Start) 유지**

---

2) 요구 파라미터 정의

파라미터	설명
V_target (RPM or %)	정상 운전 목표 속도
V_current	현재 모터 속도
accel_rate (RPM/s 또는 %/s)	속도 증가량(초당 증가 비율)
startup_time_min	최소 가속 시간

파라미터	설명
startup_profile	선형/지수/단계형 프로파일
I_limit_start	기동 단계에서 허용 전류 한계
torque_limit	이물질 감지/막힘 판단 기준

---

3) 소프트 스타트 방식 선택

① 선형 Soft-Start (Linear Ramp-up) – 권장

가장 안정적이며 폐유 점도 조건에 잘 맞음.

```
V(t) = V_target * (t / T_ramp)     (0 ≤ t ≤ T_ramp)
```

② S-Curve(Jerk-Limited) – 고급형

기계적 충격이 가장 적음, 고가 기어모터용.

③ Step Ramp(계단식 가속) – 배터리 절약형

특정 점도 환경(겨울철 등)에서 유리.

---

4) 알고리즘 동작 흐름 (Pseudo Flow)

```
StartSoftStart():
    read I_current
    if I_current > I_limit_start:
        delay(50ms) and retry
        if 계속 과전류 → FAULT

    V_current = 0
    t = 0

    while V_current < V_target:
        V_current += accel_rate * dt
        setMotorPWM(V_current)
        read I_current, torque

        if I_current > I_limit_start or torque > torque_limit:
            StopMotor()
            EnterBlockMode()
            return FAIL

        wait(dt)   // 10~20ms

    return OK
```

---

5) 전체 소프트 스타트 State Machine

```
[INIT]
   ↓
[CHECK_LOAD]
   - 모터 정지 상태 전류 확인
   - 기어/컨베이어 물리적 걸림 확인
   ↓
[RAMP_UP]
   - 선형/지수 프로파일로 PWM 증가
   - 부하·토크 상시 감시
   ↓
[HOLD_STABLE]
   - 목표속도 근처 도달
   - ±5% PID 정착
   ↓
[RUN]
   - 정상운전
```

---

6) 부하 기반 Adaptive Soft-Start(지능형 가속 보정)

폐식용유는 온도/점도에 따라 부하가 급변하므로,

가속 과정에서 실시간 부하를 감지하여 **자동으로 팀핑(tempering) 속도를 조절**하는 로직을 추가한다.

```
if I_current > 0.7 * I_limit_start:
    accel_rate = accel_rate * 0.5    // 가속 절반으로
elif I_current < 0.3 * I_limit_start:
    accel_rate = accel_rate * 1.2    // 여유 부하 → 가속 증가
```

장점

- 점도 높은 겨울철 폐유에서도 안정적 기동
- 무부하 상태에서는 불필요한 지연 없이 빠르게 목표 속도 도달

---

7) 양측 컨베이어 동기(Start Sync) 제어

좌/우를 동시에 기동할 경우 속도 편차가 발생하면
 유입 흐름이 한쪽으로 치우칠 수 있으므로 다음 제어를 적용한다.

A. 시작 오프셋 보정

```
V_left(t)  = Ramp(t)
V_right(t) = Ramp(t) * sync_coeff
```

B. 실시간 엔코더 비교

```
if abs(rpm_left - rpm_right) > Δrpm_limit:
    느린쪽 accel_rate 증가
    빠른쪽 accel_rate 감소
```

C. 기동 순서

1. 두 모터 모두 PRE-CHECK
2. 부하 정상 확인 후
3. 동시에 RAMP_UP 진입
4. PID로 동기 유지

---

8) 예외 처리(오류/Fault 대응)

(1) 과전류/토크 급등

- 즉시 PWM 0 → HARD STOP
- 역회전 0.5초(이물질 토출 목적)
- 예외 카운트 증가
- 3회 이상 발생 → FAULT 상태 전환

(2) 기동 시간 초과

```
if t > startup_time_limit and V_current < 0.8*V_target:
    startup_fail
```

(3) 좌/우 속도 편차 과도

- 2초 이상 지속될 경우 장비 경보
- 한쪽 모터를 보호 정지

---

9) RTOS Task 구조

```
ConveyorTask (주기 20ms)
  ├─ readSensors()
  ├─ if state == RAMP_UP:
  |      softStartStep()
  ├─ else if state == RUN:
  |      speedControlPID()
  ├─ updateMotorPWM()
  └─ safetyCheck()
```

PWM 업데이트는 20ms(50Hz) 이하의 속도로 변경하면
 대부분의 BLDC/브러시 모터에서 안정적인 가속이 가능함.

---

10) 파라미터 튜닝 가이드라인

항목	권장값
accel_rate	5–15%/100ms
dt (제어 주기)	10–20ms
I_limit_start	정격전류의 1.5배
torque_limit	설비별 실측 기반 설정
startup_time_limit	3–5초

---

11) 최종 요약

컨베이어 소프트 스타트 알고리즘은
선형 램프 + 부하 기반 가속 조절 + 실시간 이물질 감지 + 양측 동기화
구조로 설계되며,

- 기어 스트레스 감소
- 배터리 돌입 전류 방지
- 이물질 걸림 조기 차단
- 좌우 수거 균형 유지

를 동시에 달성한다.

8.3 유입량 기반 자동 속도 보정

아래는 **실사용 가능한 제어 설계 문서 수준**으로 정리한 "유입량 기반 자동 속도 보정" 알고리즘이다.
목표는 **컨베이어(입력)와 펌프(이송)를 실시간 유입량(또는 중량) 신호에 맞춰 자동으로 보정**해 탱크 과충전·과부하·필터막힘을 예방하고 전체 공정을 안정화하는 것이다. 그대로 펌웨어에 넣을 수 있는 구조(입력/출력, 필터링, 제어 루프, 예외 처리, RTOS task 배치, 파라미터 권장치 등)를 포함한다.

---

1) 제어 목적 요약

- 실시간 유입량(또는 중량 변화)을 기준으로 컨베이어 속도와 펌프 RPM을 **자동 보정**
- 탱크 레벨·필터 상태·펌프 부하를 고려해 **과충전·과부하 방지**
- 점도·온도 변화에 따른 유입 변화에 적응적 대응
- 좌/우 컨베이어 동기성 유지

---

2) 입력(센서) · 출력(액추에이터)

입력

- 유량계(기어형) 펄스 또는 RS485 유량(L/min) `flow_meas`
- 중량(로드셀) `weight` (보조/대체)

- 탱크 레벨 `level%` (초음파/차압/플로트 조합)
- 펌프 토출 압력 `p_out`
- 펌프 전류 `I_pump`
- 컨베이어 전류/토크 `I_conv_L`, `I_conv_R`
- 온도 `T` (점도 보정용)
- IMU 기울기 `tilt` (비정상 조건 시 보수)

출력

- 좌/우 컨베이어 목표 속도 `V_conv_L_cmd`, `V_conv_R_cmd` (PWM 또는 RPM)
- 펌프 속도/토출 제어 `V_pump_cmd` (PWM or freq)
- 알람/상태 플래그, UI 표시

---

3) 제어 구조(블록 레벨)

1. 센서 전처리 → 노이즈 제거 + 보정(온도 보정 등)
2. 유입량 추정기(flow estimator)
    - 기본: 유량계 직접값
    - 보조: 로드셀 Δweight/Δt → 유량 환산(밀도 보정)
    - 두 값 융합(가중평균, 신뢰도 기반)
3. 상위(목표) 로직: 목표 유입속도(`flow_setpoint`) 산출
    - 사용자/운영 정책, 탱크 잔량, 후단 처리능력 기준
4. 오차 계산: `err = flow_setpoint - flow_meas`
5. Cascade 제어
    - 외부 루프: 유량 PID → 생성되는 `V_conv_ref` (피드백→컨베이어)
    - 내부 루프: 컨베이어/Pump 속도 PID → 실제 PWM
6. 안전/제한: 압력, 전류, 레벨 등으로 제약 적용(soft-limit/hard-stop)
7. 동기화: 좌/우 컨베이어 보정(동기 PID 또는 master-slave)

---

4) 필터링·추정기 구체화

4.1 유량 필터

- 펄스→초당 펄스수 `pulse/s` → `flow_meas_raw = pulse/s × K_pulse`
- 필터: 이동평균(윈도우 N=5) 또는 1차 저역통과 (τ = 0.5–1.0 s)
- 노이즈/스파이크 제거: median filter + spike reject (Δ>thr → 보간)

4.2 중량 기반 유량 환산 (보조)

- `flow_from_weight = (weight_now - weight_prev) / Δt / ρ` (L/s)
- 밀도 ρ는 온도 보정 표 또는 실측값으로 보정

4.3 융합 로직

- 신뢰도 산정: 유량계 상태(에러, 펄스 유무), 로드셀 노이즈 수준
- `flow_est = w1*flow_meas_filtered + w2*flow_from_weight` (w1+w2=1)
    - 기본 w1=0.8, w2=0.2 (필요시 적응형 변경)

---

5) Cascade 제어 설계 (권장)

외부 루프 (유량 루프)

- 목표: `flow_setpoint`를 추종
- 제어기: PID (Kp_f, Ki_f, Kd_f)
- 출력: `conv_speed_ref` (unit: %PWM or RPM)

Pseudo:

```
err_f = flow_setpoint - flow_est
conv_speed_ref = PID_flow.update(err_f)
conv_speed_ref = saturate(conv_speed_ref, conv_min, conv_max)
```

내부 루프 (컨베이어 속도 루프)

- 목표: `conv_speed_ref`를 실제 컨베이어 속도/전류로 추종
- 제어기: PI 또는 PID (Kp_c, Ki_c)
- 보호: current-limit 기반 토크 리미트 (I_limit_conv)

Pseudo:

```
err_c = conv_speed_ref - conv_speed_meas
pwm_cmd = PID_conv.update(err_c)
if I_conv > I_limit_conv: pwm_cmd = pwm_cmd * safety_factor
apply_pwm(pwm_cmd)
```

펌프 제어 (동기 또는 보조)

- 펌프는 보통 컨베이어와 매칭(비례)되거나, 저장 탱크 레벨에 따라 독립 제어
- 간단한 규칙:
    - `V_pump_cmd = K_ff * conv_speed_ref + K_fb * (flow_setpoint - flow_est)`
    - K_ff: feedforward 계수(예: 펌프 유량 특성 보상)

---

6) 안전 및 제한 로직 (우선순위)

- **레벨 상한(예: 95%) → 즉시 컨베이어 속도 감소(50%) → 펌프 정지 → PRE-HIGH 알람**
- **압력 초과(>P_max) or 펌프 전류 초과 → 펌프 정지 + 컨베이어 감속/정지**
- **컨베이어 전류 초과 → 해당 축 정지 → 역회전 0.5s → 재시도** (오류 카운트 누적 시 FAULT)
- **유량 센서 이상(무펄스) → 중량 기반 모드 전환 또는 안전 정지**
- **IMU tilt>θ_limit → 즉시 주행 제한, 컨베이어 정지**

Hard-stop 우선(하드웨어 E-Stop, Float HIGH 등).

---

7) 알고리즘 Pseudocode (단순화)

```
// 주기: CONTROL_DT (e.g., 100ms)
void ControlTask() {
  flow_raw = read_flow_sensor();
  flow_filtered = lowpass(flow_raw);

  weight = read_loadcell();
  flow_w = (weight - weight_prev)/dt / density;
  weight_prev = weight;

  flow_est = fuse(flow_filtered, flow_w);

  // compute desired setpoint (could be fixed or adapt by tank level)
  flow_set = determine_flow_setpoint(level, op_mode);

  // flow outer PID
  conv_ref = pid_flow.update(flow_set - flow_est);

  // apply anti-windup and limits
  conv_ref = clamp(conv_ref, conv_min, conv_max);

  // conveyor inner PID / current limiter
  conv_pwm = pid_conv.update(conv_ref - conv_meas);
  if (I_conv > I_limit) conv_pwm *= (I_limit / I_conv);

  // pump feedforward + feedback
  pump_pwm = K_ff*conv_ref + K_fb*(flow_set - flow_est);
  pump_pwm = clamp(pump_pwm, pump_min, pump_max);

  // safety overrides
  if (level > LEVEL_HIGH) { conv_pwm *= 0.2; pump_pwm = 0; trigger_alarm(); }
  if (p_out > P_MAX) { pump_pwm = 0; conv_pwm = conv_pwm * 0.5; }

  setMotorPWM(conv_pwm);
  setPumpPWM(pump_pwm);
}
```

---

8) RTOS Task 배치 제안

- `SensorTask` (주기 50–100 ms): 모든 센서 샘플링 + 전처리
- `ControlTask` (주기 100 ms): 위 Cascade 제어 수행
- `SafetyTask` (주기 20–50 ms): 최고 우선순위, 즉시 차단 신호 처리
- `Comm/UI Task` (주기 200–1000 ms): 상태 송신, UI 업데이트
- `LoggingTask` (주기 1 s)

---

9) 파라미터 권장값(초기값)

- 제어 주기 `CONTROL_DT` = 100 ms
- 유량 PID: Kp_f = 0.6, Ki_f = 0.3, Kd_f = 0.0 (단위에 따라 조정)
- 컨베이어 PID: Kp_c = 0.5, Ki_c = 0.2
- I_limit_conv = 1.2 × 정격전류
- conv_min = 10% (idle), conv_max = 100%
- pump_min = 20%, pump_max = 90%
- flow_filter τ = 0.8 s (1차 LPF)
- 융합 가중치 w1=0.8(유량계), w2=0.2(로드셀)

> 실제 파라미터는 시스템 관성·모터 특성·유량센서 K값에 따라 현장튜닝 필요

---

10) 적응형 보정(온도·점도 보정)

- 온도 `T`를 읽어 밀도 ρ(T)·점도 보정 인자 `alpha(T)` 적용
- `flow_setpoint` 또는 PID gain에 온도 기반 이득 보정 적용
  - 낮은 T(점도↑) → 가속률 줄임, I_limit 완화

---

11) 실패 모드 & 페일오버 전략

- **유량 센서 실패(무응답)**
  - 자동: 로드셀 기반 유량 사용(임시) → 경고 로그 → 리셋 시도
  - 복구 불가 → 안전 정지
- **로드셀 실패**
  - 유량계 우선 사용
- **센서 튀는 값**
  - Spike reject → 이전 정상값 유지 → 로그
- **연속 재시도 실패(예: 3회 역회전 후 정상화 없음)**
  - FAULT 상태 → 비상알림 및 서비스 모드

---

12) 튜닝 및 검증 절차

1. 센서(유량계/로드셀/압력) 캘리브레이션
2. 무부하 컨베이어/펌프 응답 실험 → conv_ref ↔ pwm mapping 작성
3. Closed-loop PID 튜닝 (Ziegler–Nichols 또는 수동 튜닝)
4. 부하(점도 높은 폐유) 시나리오 테스트 → I_limit/torque_limit 조정
5. 동기성 테스트(좌/우 컨베이어) → Δrpm_limit 설정
6. 비상시나리오 테스트(E-Stop, 센서 단선, 레벨 만수) 검증

---

13) 운영 시나리오 예시

시나리오 A — 정상 유입 증가

- flow_est > flow_set → PID 명령으로 conv_ref 감소 → pump_pwm 보정 → 안정화

시나리오 B — 필터 막힘 발생

- p_out 상승 + flow 감소 + I_pump 증가 → SafetyTask 우선 펌프 감속/정지, conv 속도 감소, 알람

시나리오 C — 유량센서 단선

- 즉시 로드셀 모드 전환, flow_set 70%로 축소 운전 → 유지보수 안내

마무리 요약

- 핵심은 **유량 기반 외부 루프 + 컨베이어/펌프의 내부 루프(캐스케이드)** 방식.
- 필터링·융합(유량계+로드셀)+온도 보정으로 신뢰도 향상.
- SafetyTask가 모든 제어권을 덮어쓰는 구조로 설계해야 함.
- 실제 파라미터는 현장 시험으로 튜닝 필요 — 문서의 권장값은 출발점으로 사용.

8.4 저장 탱크 레벨 기반 자동 중지

본 항목은 **유입 시스템의 최종 안전 방어선**으로서, 저장 탱크 레벨(초음파/차압/플로트 복합) 값을 활용해 **과충전 방지 및 비상 정지**를 수행하는 제어 설계 문서 수준으로 정리하였다.

---

1) 제어 목적

- 탱크 **과충전(overfill) 방지**
- 펌프/컨베이어 작동 시 **레벨 변화 추적**
- **정상/주의/위험 3단계** 레벨 정책 반영
- 하드웨어 플로트 스위치를 통한 **Fail-safe 이중화**

---

2) 주요 입력/출력

입력

- 레벨 센서 1 (초음파) `level_Ultrasonic`
- 레벨 센서 2 (차압 or 플로트) `level_DP`
- 레벨 속도 변화율 `d(level)/dt`
- 탱크 용량 `V_capacity`

출력

- 펌프 정지/감속 명령
- 컨베이어 정지/감속 명령
- UI 경고/경보
- 리모트 알람 전송

---

3) 상태 정의(Level-Zone)

상태명	레벨 % 조건	동작 정책	비고
NORMAL	0% ~ 85%	정상 운전	여유 영역
PRE-HIGH	85% ~ 95%	유입 감속 · 주의 알림	완충 영역
HIGH/STOP	95% ~ 100%	**즉시 유입 중지** · 경보	안전 차단
EMERGENCY	플로트 HIGH 또는 센서 오류	**즉시 모든 구동 차단**	Fail-safe

※ 수치는 장비 사양에 따라 조정 가능
→ 권장: PRE-HIGH=85% / HIGH=95%

---

4) 제어 알고리즘 개요

```
레벨 측정 → 필터링 → 영역 판별(NORMAL / PRE-HIGH / HIGH / EMERGENCY)
    ├─ NORMAL → 속도 정상 유지
    ├─ PRE-HIGH → 펌프 감속 / 컨베이어 감속 / 경고 표시
    ├─ HIGH → 펌프 OFF / 컨베이어 OFF / 경보 송신
    └─ EMERGENCY → 전체 OFF (Fail-safe)
```

---

5) 안전 로직 동작 규칙

A. PRE-HIGH (85~95%)

- 펌프 속도 **50% 감속**
- 컨베이어 속도 **70% 감속**
- UI: “만수 주의(예정)” 점멸
- 원격: Warning 메시지 송신

B. HIGH (≥95%)

- **펌프 즉시 정지**
- **컨베이어 즉시 정지**
- BLE/Wi-Fi/LTE 경보 송신
- 조작자 승인(또는 수위 복귀) 전까지 자동 복귀 금지

C. EMERGENCY 조건

- 하드 플로트 HIGH
- 센서 불일치(Δ>10%) 지속 3초
- 레벨 값 급상승 (d(level)/dt > 임계치)
  → 즉시 **E-Stop 처리**
  → 릴레이 차단(HW 레벨)

---

6) 필터링 & 신뢰도 평가

항목	방법
센서 노이즈 제거	3~5포인트 이동평균
측정 불일치	Δ=
급락/급상승	Spike reject + 상태 Freeze

센서 신뢰도 `R` 기반 융합:

```
level_est = R*Ultrasonic + (1-R)*DP
```

※ R은 평상시 0.7~0.9, 이상 시 0.3~0.5 자동 변화

---

7) 상태도(State Machine)

NORMAL
⬇️ (level ≥ PRE-HIGH%)
PRE-HIGH
⬇️ (level ≥ HIGH%)
HIGH/STOP
⬅️ (level ≤ NORMAL% - hysteresis)
⬆️ (플로트 HIGH → EMERGENCY)

EMERGENCY → 조작자 수동 해제 전 복귀 불가

- 히스테리시스 권장: 5%
  → 경계치 진동으로 인한 ON/OFF 반복 방지

---

8) Pseudocode 예시

```
void TankLevelControlTask() {

  level = get_level_estimated(); // filtered + fused value

  if (float_switch_high || sensor_fault) {
    stopPump();
    stopConveyor();
    raiseEmergencyAlarm();
    return;
  }

  if (level >= HIGH_LEVEL) {
    stopPump();
    stopConveyor();
    raiseHighAlarm();
    state = HIGH_STOP;
  }
  else if (level >= PREHIGH_LEVEL) {
    setPumpSpeed(0.5 * nominalSpeed);
    setConveyorSpeed(0.7 * nominalSpeed);
    showWarning("Tank almost full");
    state = PRE_HIGH;
  }
  else {
    // NORMAL
    resumeNominalSpeeds();
    state = NORMAL;
  }
}
```

---

9) 이중 보호 계층(Fail-Safe)

계층	수단	특징
소프트웨어	레벨 기반 감속/정지	환경값 추종
하드웨어	**플로트 HIGH + 릴레이 차단**	전원 자체 차단
구조적	오버플로 배출구(수동/자동)	최후 수단

안전 설계 원칙:

> 소프트웨어는 편의를 담당, **안전을 책임지는 것은 하드웨어**

---

10) 검증 및 테스트 절차

1. 센서 캘리브레이션(레벨-용량 대응)

2. 유입 속도별 HIGH 도달 시간 측정
3. 동작 검증 시나리오:
    - 정상 → PRE-HIGH → HIGH 전이
    - 급유입(펌프 과속 상태)에서 HIGH 감지
    - 센서 오류/단선 → EMERGENCY 처리 확인
4. 하드 플로트 성능검증:
    - 릴레이 차단 즉시 전원 off 확인
5. UI/원격 알림 지연 확인(≤1초 목표)

---

11) UI/알람 표시 가이드

상태	UI 표시	원격 알람	핸들링
NORMAL	녹색 상태	None	-
PRE-HIGH	노란색 점멸	Warning	속도 자동 감속
HIGH/STOP	빨강 점멸	Alarm	운전 재개 금지
EMERGENCY	빨강 고정	Critical Alarm	수동 리셋 필요

✨ 요약

- 레벨 기반 자동 중지는 **운전 마지노선의 마지막 안전 기제**
- PRE-HIGH → 감속, HIGH → 즉시 중지, EMERGENCY → 하드차단
- **센서 융합 + 히스테리시스 + 하드웨어 플로트**가 핵심

8.5 펌프 PWM 제어

본 항목은 **폐식용유 펌프(기어 펌프·로터리 펌프·다이어프램 펌프 등)**을 PWM 방식으로 정밀 제어하기 위한 소프트웨어/하드웨어 설계 기준을 기술한 문서이다.
폐식용유의 **점도 변화·온도 변화·유입량 변화**에 대응하여 안정적인 흡입/토출을 유지하는 것이 목표이다.

---

1) 설계 목적

- 펌프의 토출량(L/h)을 **PWM 듀티(Duty %)와 RPM의 함수**로 제어
- 점도 변화(20~80°C)에 따른 **펌핑 부하 증가/감소에 자동 대응**
- **부하 감지(전류량 상승)** → 자동 감속 또는 안전정지
- 소프트 스타트/스톱 구현
- 저장 탱크 레벨과 연동하여 자동 정지

---

2) PWM 제어 대상

- **DC 기어펌프**
  - PWM 직접 인가 가능
  - 12V/24V 권장
- **BLDC 펌프(3상)**
  - PWM은 FOC 제어기의 Input으로만 사용
  - 본 문서는 DC 기어펌프 기준 설명
- **다이어프램 펌프**
  - 펌브 사이클이 중단되면 압력이 급상승하므로
  - PWM 듀티 최소값을 **20~30% 이상으로 제한**

---

3) PWM 제어 주파수

주파수	특징	권장 펌프
400–800 Hz	소음 발생 가능	저가형 DC펌프
20–25 kHz (권장)	가청대역 이상 → 매우 조용	**기어펌프·다이어프램 펌프 최적**

권장값: 20kHz

---

4) 기본 제어 파라미터

- `PWM_min = 20%` (기동 불가 방지)
- `PWM_max = 95%` (과전압·과열 방지)
- `PWM_soft_start_time = 1.0~2.0 sec`
- `I_limit` (과부하 전류값, 예: 6A)

---

5) 핵심 제어 알고리즘 구조

```
입력: 목표 유량(Q_target), 저장탱크 레벨, 전류 센서, 온도, 점도 추정값
출력: PWM Duty
```

PWM 결정 로직:

```
1) Soft-Start 구간
2) 점도 보정
3) 부하 보정(전류)
4) 탱크 레벨 안전 제약
5) 최종 PWM 결정
```

---

6) Soft-Start 알고리즘

펌프를 갑자기 0→100% PWM으로 올리면
 기름 점도 때문에 기동 실패 또는 과전류 발생 가능.

```
start_pwm = PWM_min
for t < soft_start_time:
    PWM = map(t, 0 → soft_start_time, start_pwm → PWM_target)
```

점도 높을 때는 soft-start 시간 자동 증가:

```
if viscosity_high:
    soft_start_time = 2~3 sec
```

---

7) 점도 기반 PWM 보정

폐식용유는 **온도·점도에 따라 부하가 크게 변함**
 점도는 직접 측정하지 않고 **유입 온도(T)로 간접 추정**.

예시 모델:

```
viscosity_factor = clamp( 1.0 + k * (25°C - OilTemp), 1.0, 1.5 )
PWM_viscosity = PWM_base * viscosity_factor
```

- k값 일반적으로 0.015~0.03
- 10°C 유입의 경우 → 펌핑 부하 약 15~30% 증가

---

8) 부하(전류) 기반 실시간 보정

모터 전류 상승 → 펌프 부하 증가 의미.

```
if I_meas > I_limit:
    PWM = PWM - Δ
    if I_meas 계속 증가:
         stopPump()   // 막힘 또는 내부 이상
```

- Δ: 2~5% 단위 PWM 저감
- Stall 지속시간 0.5~1.0초 → 긴급 정지

---

9) 저장 탱크 레벨 제한과 연동

레벨 구간별 제어:

- **NORMAL (0~85%)**
    - PWM 정상제어
- **PRE-HIGH (85~95%)**

    - PWM 50% 감소
- **HIGH (≥95%)**
    - PWM=0 (즉시 정지)
- **EMERGENCY**
    - 릴레이 차단, 전원 OFF

---

10) PWM 제어 흐름도

```
[센서 수집]
      ↓
[Soft Start 적용]
      ↓
[온도 → 점도 보정]
      ↓
[전류값 기반 부하 보정]
      ↓
[레벨 상태 기반 제한]
      ↓
[최종 PWM 출력]
```

---

11) Pseudocode 예시

```
void PumpPWMControlTask() {

    float pwm = PWM_base;

    // 1. Soft Start
    if (state == PUMP_STARTING) {
        pwm = softStart(PWM_target);
    }

    // 2. Temperature/Viscosity Compensation
    float viscosity_factor = 1.0 + k * max(0, (25 - oilTemp));
    pwm *= viscosity_factor;

    // 3. Current-based Load Control
    if (I_meas > I_limit) {
        pwm -= 0.05; // reduce by 5%
        if (I_meas > I_critical) {
            stopPump();
            return;
        }
    }

    // 4. Level Protection
    if (level >= HIGH_LEVEL) {
        stopPump();
        return;
```

```
    } else if (level >= PREHIGH_LEVEL) {
        pwm *= 0.5;  // reduce output
    }

    // 5. Apply Boundaries
    pwm = clamp(pwm, PWM_min, PWM_max);

    // Output to hardware
    setPWM(pwm);
}
```

---

12) PWM 제어 안정성 기준

- PWM 신호는 **하드웨어 타이머 기반**
- MCU는 32bit 타이머(STM32) 권장
- MOSFET 드라이버는 **게이트 충전량(Qg) 큰 제품** 추천
- **스위칭 손실 줄이기 위해 Dead-time 보정 필요**

---

13) 시험·검증 항목

시험	내용
기동성 시험	Soft-start가 실제 부하에서 문제 없는지
점도 변화 시험	10°C, 20°C, 30°C, 40°C 조건에서 RPM 비교
Stall 시험	토출 라인 막음 → 과전류 정지 확인
레벨 연동 시험	PREHIGH/HIGH 구간 속도 감속/정지 확인
장시간 운전	2~6시간 연속 펌핑 테스트

요약

- PWM은 펌프의 **RPM·유량·부하 제어의 핵심**
- Soft-start + 점도 보정 + 전류 기반 부하 보호 + 레벨 연동이 필수
- 저장 탱크 상태와 완전히 통합된 제어 구조 필요

8.6 배터리/전류 모니터링

개요

이 항목은 이동형 폐식용유 수거 장치의 **전력 안정성**, **안전성(BMS 연동)**, **잔량 기반 운전 제어**, **배터리 수명 확보**, **과전류 보호**를 목표로 하는 배터리/전류 모니터링 설계를 다룹니다.

---

① 배터리 전압·전류 측정 구조

● 전압 측정(ADC 기반)

- MCU의 ADC 입력을 사용하여 배터리 전압을 실시간 측정
- 고전압 배터리(12V·24V)는 반드시 **전압 분배(저항 2~3개)** 후 입력
- 예: ESP32/STM32 ADC 입력 0~3.3V
  - 24V → (R1=100k, R2=10k) → 3.3V 이하로 변환
- 배터리 소모 패턴(전압 대비 % 테이블)은 **화학 종류(Li-ion, LiFePO4)**에 따라 다름

● 전류 측정(Shunt or Hall Sensor)

전류 측정 방식은 2개 가능:

1) 샌스 저항(Shunt Resistor + INA류 증폭기)

- 1~10 mΩ 정도의 저저항을 배터리 음극에 직렬 삽입
- INA219 / INA226 / INA238 등
  - 전압 + 전류 + 전력 자동 계산
  - I²C로 간단히 읽기 가능
- 장점: 정확, 비용 저렴
- 단점: 발열, 고전류(>20A)엔 비추천

2) 홀 센서(Ampere Sensor, 비접촉)

- ACS712, ACS758, WCS1800 등
- 장점: **절연**, 발열 없음, 고전류(30~200A) 측정 가능
- 단점: 정확도는 INA 대비 약간 떨어짐

> 이동식 폐식용유 처리 장치처럼 **모터+펌프+컨베이어가 동시에 구동**하는 구조는
> **홀센서 기반 전류 측정이 더 적합**.

---

② BMS와 MCU 통신 구조

BMS의 기본 역할

- 셀 밸런싱
- 과충전·과방전 보호
- 과전류 차단
- 셀 전압 모니터링
- 온도 모니터링
- 충방전 차단 MOSFET 제어

MCU가 읽어야 할 값

1. 배터리 팩 전압
2. 실시간 전류
3. SoC(State of Charge, 잔량 %)
4. SoH(수명)
5. 온도
6. 알람(OVP, UVP, OTP, OCP 플래그)

BMS 모델 선택 기준

- UART / CAN / I²C 통신 지원
- LiFePO4 4S~8S 지원
- 충방전 20~60A급
- UART 프로토콜 문서 제공 여부(중요!)

권장 통신 방식

- 산업용 또는 혹사 조건 → **CAN 통신 BMS (최고 안정성)**
- 저비용 가벼운 모델 → **UART 기반 Daly/JK BMS**

---

③ MCU 전력 소비 감시 로직

● 기본 로직

1. 전압 < 20% → “저전력 경고” 출력
2. 전압 < 10% → 펌프 속도 자동 저감
3. 전압 < 5% → 시스템 자동 Shutdown
4. 전류 급증(>30A 등) → 모터 정지 + 오류 표시
5. 펌프 역방향 전류 감지 → 밸브 문제 판단

● 충전 시 로직

- 충전 중 과전압 예측
- 충전 전류 감소 시점 판단
- 황변된 폐식용유 기반 시스템이라면 외부 충전기와 통합 시
  BMS 상태를 UI로 피드백해야 함

---

④ 센서·전력부 배선 레이아웃

● 배터리 → BMS → 부하(모터/펌프) 구조

- **GND 루프 방지**
- 고전류 라인은 굵은 전선(8~12AWG)
- 신호선(I²C/UART)은 모터 라인에서 멀리 배치
- Hall 센서는 **배터리 +라인**에 설치
- Shunt 센서는 **배터리 -라인**에 설치

---

⑤ 방수·방진 고려(실외 장치 전제)

- KV 커넥터·XT60/XT90 방수 타입
- 배터리 팩은 IP54~IP65
- 케이블 글랜드로 유입부 마감
- 센서 보드는 실리콘 코팅(Conformal Coating)

---

⑥ UI 화면에 표시할 데이터

필수 표시

- 배터리 %
- 실시간 전류 (A)
- 실시간 전압 (V)
- 소비 전력 (W)
- 남은 사용 가능 시간(재계산 값)
- BMS 상태(OK/OCP/OTP/UVLO 등)

추가 가능

- 펌프 구동 중 전류 패턴 그래프
- 모터 부하 예측
- 배터리 수명(싸이클 카운트 기반)

---

⑦ 전체 시스템에서의 역할 정리

기능	목적	방식
배터리 전압 측정	잔량 계산	ADC + 분압 회로
전류 측정	부하 감지/보호	홀 센서 또는 INA219
BMS 연동	안전·보호	CAN/UART

기능	목적	방식
전력 제한 로직	과부하 방지	PWM 감소, 모터 정지
UI 피드백	사용자 안내	OLED/LCD 표시

8.7 OTA 펌웨어 업데이트

개요

OTA(Over-The-Air) 업데이트 기능은 현장에서 장비를 분해하지 않고도 **무선으로 MCU 펌웨어를 갱신**할 수 있게 해줍니다. 이동형 폐식용유 수거 장치는 실사용 환경이 거칠고 유지보수 주기가 길기 때문에 OTA 기능이 **유지보수 비용 절감**, **버그 패치 속도 향상**, **기능 추가 편의성**을 크게 높입니다.

아래 내용은 ESP32(ESP-IDF/Arduino) 또는 STM32 + 외장 통신 모듈(LTE/WiFi/BLE) 기준으로 정리한 OTA 설계 가이드입니다.

---

① OTA 방식 선택 (BLE / Wi-Fi / LTE)

1) BLE OTA

- 장점: 낮은 전력, 스마트폰 앱에서 간편
- 단점: 속도 느림(수백 KB~1MB+ 펌웨어는 오래 걸림)
- 모바일 유지보수 인력이 다닐 때 유용

2) Wi-Fi OTA

- 장점: 속도 빠름, HTTPS 쉽게 적용
- 단점: Wi-Fi 환경 요구
- 장비에 Wi-Fi 핫스팟(테더링) 걸어도 가능

3) LTE OTA (MQTT/HTTPS)

- 장점: 현장 네트워크 영향 없음, 가장 안정적
- 단점: 비용 높음
- 대규모 운영 차량/장비 관리에 최적

> 장치가 외부에서 이동하며 사용되므로
> **BLE(간단 업데이트) + Wi-Fi(Large Update)** 조합이 가장 현실적.

---

② OTA 펌웨어 분할 구조

듀얼 파티션 구조 (권장)

- 파티션 A: 현재 실행 중인 펌웨어
- 파티션 B: 업데이트 받을 새로운 펌웨어
- OTA 후 검증에 성공하면 A←→B 스왑

이 방식은 업데이트 도중 전원 꺼져도 **장비 벽돌화 방지**.

필수 파티션 구성 예시(ESP32)

- **bootloader**
- **nvs**
- **otadata**
- **app0** (현재 실행)
- **app1** (OTA 다운로드용)
- **spiffs** (설정값 저장)

STM32라면

- **Bank1/Bank2 듀얼 이미지 구조 + Bootloader**
- 또는 외장 플래시(QSPI/NOR)에 OTA 이미지 저장 후 교체 방식을 사용.

---

③ OTA 과정 전체 시퀀스

Step 1: 업데이트 요청 확인

- BLE 명령: `"OTA_START"`
- Wi-Fi/LTE: 서버에서 “새 버전 존재” 메시지

Step 2: 버전 체크

- 현재 버전 vs 서버 최신 버전 비교
- 다르면 다운로드 실행

Step 3: 이미지 다운

- BLE: 스마트폰 앱 → MCU로 바이너리 전송
- Wi-Fi/LTE: HTTPS GET으로 직접 다운로드

Step 4: 무결성 검증

- CRC32
- SHA-256 (ESP-IDF 기본 지원)
- 파일 크기 확인

Step 5: 플래시 영역 기록

- 파티션 B(app1)에 기록
- 기록 후 재검증

Step 6: OTA 스위치

- 부트로더에 "app1으로 부팅" 메시지 기록
- 재부팅

Step 7: 정상 부팅 검증

- 부팅 후 Self-test
- 이상 없으면 새 이미지 고정
- 오류 발생 시 자동 롤백(app0로 복귀)

---

④ 보안 요소(필수)

● 펌웨어 암호화

- ESP32 → AES-256 기반 OTA 암호화 지원
- STM32 → AES-XTS 또는 Secure Boot + Secure Firmware Upgrade(SBSFU)

● 펌웨어 서명(Signature)

- RSA/ECDSA 서명
- 부트로더에서 서명 확인
- 서명 틀리면 업데이트 거부 → 보안 강화

● HTTPS/TLS

- Wi-Fi/LTE OTA 시 필수
- 서버 인증서 갱신 관리 필요

---

⑤ OTA 진행 중 시스템 보호

● 절대 금지

- OTA 중 펌프/모터 구동
- OTA 중 전원 변동

● 보호 로직

- OTA 시작하면:
  - 모든 모터 OFF
  - 센서 읽기 중단
  - 전력 상태 검사(전압 < 20%이면 OTA 거부)
  - 사용자 UI에 “업데이트 진행 중...” 표시

● OTA 도중 오류 대비

- 통신 끊김
- CRC 불일치
- 플래시 쓰기 실패

→ 자동 롤백 가능하도록 파티션/부트로더 설계 필수

---

⑥ OTA UI/상태 표시 예시

● OLED/LCD에 표시

- "OTA Downloading... XX%"
- "Verifying..."
- "Installing..."
- "Done. Rebooting."

● BLE 앱 상태 표시

- 진행률
- 예상 시간
- 완료 후 자동 재접속

---

⑦ OTA 서버 설계(옵션)

간단 서버

- GitHub Releases + HTTPS URL
- ESP32는 URL 직접 접근 가능

운영용 상위 서버

- AWS S3 + CloudFront
- Firebase Hosting
- MQTT Broker(OTA 메시지 전송)

대규모 확장 대비

- 장치별 버전 관리
- OTA 실패 로그
- OTA 성공률 대시보드
- 전압 조건 검사 자동화

---

⑧ 이 프로젝트에 맞는 최적 OTA 구조 추천

유지보수 인력이 스마트폰 기반이라면

- **BLE OTA + Wi-Fi OTA 병행**
- BLE OTA: 급히 현장에서 버그 수정
- Wi-Fi OTA: 대규모 기능 업데이트

완전 자동화 운영 원한다면

- LTE 모듈 + HTTPS OTA
- 서버에서 강제 업데이트 Push 가능

8.8 에러 로그 & 진단 시스템

개요

폐식용유 수거 장비는 **컨베이어, 펌프, 탱크, 센서, 주행 모듈, 배터리 시스템** 등 여러 서브시스템으로 구성되어 있으며, 실사용 환경(기름, 이물질, 온도 변화)이 거칠기 때문에 **고장 모니터링, 에러 로깅, 진단 체계**가 매우 중요합니다.
이 장에서는 장비의 신뢰성을 높이기 위한 **에러 감지·분류·저장·전송·자가진단(Self-test)** 구조를 상세히 정의합니다.

---

① 에러 분류 체계(에러 코드 체계)

1) 레벨 분류

- **Level 0: Info**
  - 일반 상태 메시지 (시작, 종료, 세척 모드 진입 등)
- **Level 1: Warning(경고)**
  - 즉시 고장은 아니지만 동작 저하 가능
  - 예: 점도 과다로 모터 부하 증가, 레벨 센서 잡음
- **Level 2: Error(오류)**
  - 일부 기능 제한 필요
  - 예: 펌프 과열, 컨베이어 정지 감지
- **Level 3: Critical(치명적 고장)**
  - 시스템 즉시 정지 필요
  - 예: 배터리 과전류, 탱크 오버플로, 펌프 역회전 감지

2) 모듈별 에러 코드

- **1000번대: 컨베이어**
  - 1101: 좌측 모터 과전류
  - 1102: 우측 모터 과부하
  - 1103: 컨베이어 벨트 속도 불일치
  - 1104: Jam(이물질 끼임) 감지

- **2000번대: 펌프/배관**
  - 2101: 펌프 압력 과다
  - 2102: 펌프 유량 부족
  - 2103: 역류 밸브 고착
  - 2104: 배관 누유 감지
- **3000번대: 탱크/센서**
  - 3101: 레벨 센서 불일치(중복 센서 값 mismatch)
  - 3102: 초음파 센서 Timeout
  - 3103: 온도 센서 과열
  - 3104: 오버플로 위험 감지
- **4000번대: 주행 시스템**
  - 4101: 주행 모터 과전류
  - 4102: ESC 오류
  - 4103: 속도 센서 고장
- **5000번대: 전원/BMS**
  - 5101: 저전압
  - 5102: 과전류
  - 5103: 셀 밸런싱 오류
  - 5104: 온도 과열
- **6000번대: 통신/OTA**
  - 6101: BLE 연결 불안정
  - 6102: Wi-Fi 연결 실패
  - 6103: OTA 파일 무결성 실패

---

② 에러 감지 방법

1) 전류 기반 감지

- 컨베이어 모터 전류 > 기준값
  → Jam 또는 과부하 판단
- 펌프 전류 감소
  → 공회전, 유입량 부족 추정

2) 속도·압력 기반 감지

- 모터 RPM/엔코더 값 vs 목표 속도 비교
- 펌프 압력 변화 그래프 분석
- 갑작스런 압력 상승 → 막힘 가능성

3) 센서 신호 검증

- 레벨 센서 이중화:
  - 초음파 + 부력식 + 차압식 중 2개 조합
- 센서 신호 outlier 자동 제거
- 값 변화 속도(d/dt) 기반 이상 감지

4) 전원 시스템 모니터링

- 배터리 전압 변동률
- 과전류 → 즉시 펌프/모터 정지
- 온도 급변 → BMS 경고

---

③ 로그 저장 구조

1) 로그 저장 방식

- **MCU 내부 Flash (순환 버퍼 Ring Buffer)**
- **외장 SPI Flash (권장)**
  - 용량 1~4MB로 넉넉히 사용
  - 저장 형식: JSON/MessagePack

예)

```
{
  "ts": 1735678123,
  "code": 1103,
  "level": 2,
  "msg": "Conveyor Left Speed Mismatch",
  "v": { "target": 120, "actual": 84 }
}
```

2) 로그 저장 조건

- Error Level 2 이상 무조건 저장
- Warning은 최근 50개만 저장
- Critical 발생 시 즉시 플래그 설정(재부팅 후 리포트)

---

④ 실시간 진단(Real-time Diagnostics)

1) 주기적 헬스 체크 Task

- 100ms 주기: 모터/펌프 전류 검사
- 500ms 주기: 센서 값 검증
- 1초 주기: 탱크 레벨 확인

- 5초 주기: 온도/배터리 상태 진단

2) 패턴 분석 기반 진단

- 펌프 압력 패턴 변화
- 컨베이어 속도 흔들림
- 배터리 전압 드리프트

MCU 자원 여유 있으면 Kalman Filter 또는 EWMA 적용.

---

⑤ 사용자 알림(Warning → Error → Critical 단계 처리)

단계	동작
Warning	UI 표시, BLE 앱 푸시, 동작 유지
Error	관련 모듈 감속 또는 정지, 로그 기록
Critical	전체 시스템 정지, 경고음, 재가동 금지

---

⑥ 원격 진단 기능(선택)

BLE

- 최근 50개 로그 조회
- 실시간 전류/압력/레벨 값 보기

Wi-Fi/LTE

- MQTT 기반 클라우드로 에러 업로드
- OTA 실패 로그 자동 전송
- 센서 이상 패턴 서버 분석 가능

---

⑦ Self-Test(자가진단) 기능

장비 전원을 켜면 자동 점검:

- 컨베이어 모터 단시간 구동 → 속도/전류 검사
- 펌프 무부하 구동 → 유량/압력 확인
- 레벨 센서 반응 검사
- 온도 센서 정상 범위 확인
- 배터리 BMS 오류 체크
- 전원 인가 후 5초 이내 완료

이후 정상 조건이면 READY 상태 진입.

---

⑧ 유지보수 모드

정비 인력이 다음 기능을 사용할 수 있음:

- 센서 Raw 값 실시간 모니터링
- 모터/펌프 단독 구동 테스트
- 로그 백업/삭제
- 펌프 역류 테스트
- 컨베이어 정렬 확인 모드