

# 15. 종합 아키텍처 설계

## 15.1 펌웨어 구조

1	Application
2	└─ Tasks (FreeRTOS)
3	│   └─ RTCTask      : 주기적 알람 스케줄링 및 시간 관리
4	│   └─ SensorTask   : 초음파, 수위, 무게, ToF 등 센서 통합 측정
5	│   └─ ControlTask  : 밸브, 펌프 등 액추에이터 제어
6	│   └─ DisplayTask   : OLED / UART 상태 출력
7	│   └─ CommTask     : BLE, Wi-Fi, RS485 통신 관리
8	│
9	└─ Sensor Drivers
10	│   └─ Ultrasonic    : TRIG/ECHO, 거리 계산
11	│   └─ HX711        : 24bit ADC, 보정 및 필터링
12	│   └─ Water Sensor  : 20단 정전식, Threshold 감지
13	│   └─ ToF Sensor    : I²C 기반 거리, VCSEL 파라미터 조정
14	│
15	└─ HAL wrappers
16	│   └─ GPIO / EXTI   : 인터럽트 처리 및 입력 감지
17	│   └─ ADC / DMA     : 고속 샘플링 및 필터링
18	│   └─ I2C / SPI     : 센서 통신 및 충돌 방지
19	│   └─ TIM / PWM     : 펄스 측정 및 주기 제어
20	│
21	└─ RTC / Sleep Controller
22	│   └─ LSE Clock Init
23	│   └─ Alarm / Wake-up 관리
24	│   └─ Tickless Idle / STOP 모드 복귀
25	│
26	└─ Logging / Control Modules
27	│   └─ PID / Fail-Safe 제어
28	│   └─ Calibration (Offset, Scale, EEPROM 저장)
29	│   └─ FATFS / CSV 로깅
30	│   └─ MQTT / Modbus / Grafana 연동
31	

## 15.2 하드웨어 매핑

센서	인터페이스	포트
HC-SR04	GPIO + TIM	PB9 / PB8
HX711	GPIO	PA1 / PA2
VL53L0X	I²C1	PB6 / PB7
Water Sensor	I²C2	PB10 / PB11
Voltage	ADC1	PA0

센서	인터페이스	포트
RTC	LSE 32.768kHz	PC14 / PC15
Pump Relay	GPIO	PB12
OLED	I <sup>2</sup> C1 (공유)	PB6 / PB7

구분	기능	MCU 핀	인터페이스	비고
초음파 센서 (HC-SR04)	TRIG 신호 출력	PB8	GPIO Output	10μs 펄스 발생
	ECHO 신호 입력	PB9	TIM4 CH4 (Input Capture)	거리 계산용 타이머
무게 센서 (HX711)	DT (Data)	PA1	GPIO Input	24bit ADC 데이터 수신
	SCK (Clock)	PA0	GPIO Output	데이터 클록 생성
정전식 수위센서 (20 단)	Level Input[0..19]	PC0~PC19	GPIO Input	각 단별 Threshold 감지
	Multiplexer Control	PB0~PB3	GPIO Output	선택형 스캔 구조
ToF 센서 (VL53L0X 등)	SDA / SCL	PB7 / PB6	I <sup>2</sup> C1	Dual Address 충돌 방지
	XSHUT	PA8	GPIO Output	개별 Enable 제어
RTC (Real-Time Clock)	LSE 32.768 kHz	PC14 / PC15	OSC32_IN / OUT	백업 배터리 유지
OLED 디스플레이 (SSD1306)	SDA / SCL	PB9 / PB8	I <sup>2</sup> C2	128×32 표시용
UART 통신 (로그 / BLE)	TX / RX	PA9 / PA10	USART1	<code>printf()</code> 기반 디버깅
RS485 통신 (Modbus)	TX / RX / DE	PB10 / PB11 / PB12	USART3	Half-duplex 구성
펌프 제어 (릴레이 / MOSFET)	Control	PB1	GPIO Output	PWM 제어 가능
밸브 제어	Control	PB2	GPIO Output	수위 제어용
EEPROM / Flash 저장	Internal Flash	—	HAL Flash API	Calibration 데이터 저장
SD 카드 (FATFS)	CS / SCK / MISO / MOSI	PA4 / PA5 / PA6 / PA7	SPI1	CSV 로깅

구분	기능	MCU 핀	인터페이스	비고
전원 / 보호회로	TVS / Fuse / LC Filter	—	—	전원 안정화 및 ESD 대응
디버깅 포트	SWDIO / SWCLK	PA13 / PA14	ST-LINK	실시간 디버깅 지원

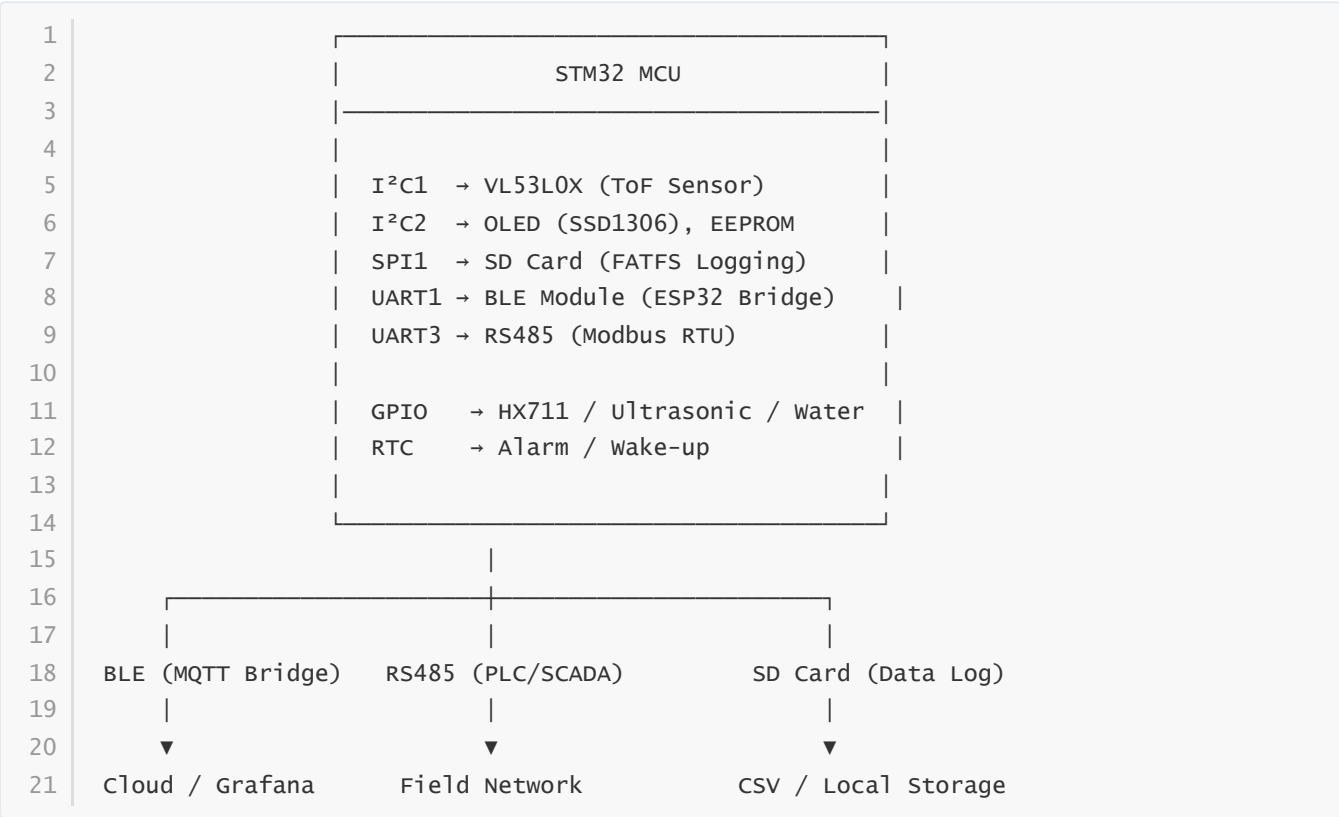
설계 포인트

- I<sup>2</sup>C Address 충돌 방지: ToF 센서 다중 연결 시 XSHUT 핀으로 순차 초기화
- 저전력 설계: 센서별 전원 제어 (MOSFET 스위칭) + STOP 모드 RTC Wake-up
- 공통 GND / 전원라인 필터링: 아날로그 / 디지털 노이즈 분리
- 보정 및 로그 저장: Flash Page Write 또는 SD 로그 병행 지원

15.3 통신 구조

시스템의 통신 구조는 내부 센서 버스(I<sup>2</sup>C, SPI), 외부 통신(UART, RS485, BLE), 및 데이터 로깅 인터페이스(SD/FATFS)로 구성된다. 각 계층은 비동기적으로 동작하며 FreeRTOS Task 간 Queue 또는 Message Buffer로 데이터가 교환된다.

1. 전체 통신 블록 다이어그램



2. 내부 통신 구조

인터페이스	대상	주소 / 채널	데이터 방향	주요 기능
I <sup>2</sup> C1	VL53L0X (ToF)	0x29 / 0x30	Master → Slave	거리 측정
I <sup>2</sup> C2	SSD1306 / EEPROM	0x3C / 0x50	양방향	OLED 표시 / 설정 저장
SPI1	SD Card	CS=PA4	Master → Slave	FATFS 로깅
GPIO	HX711 / HC-SR04 / WaterLevel	—	Mixed	센서 제어 및 입력 감지

- I<sup>2</sup>C: 센서 및 표시 장치 간 저속 버스 (100~400kHz)
- SPI: 대용량 데이터 전송용 (최대 수 MHz)
- GPIO: 타이밍 제어 및 단발 신호 측정에 사용

3. 외부 통신 구조

인터페이스	프로토콜	주요 목적	속도	비고
UART1	BLE (ESP32 MQTT 브릿지)	실시간 데이터 전송 / 원격 모니터링	115200bps	JSON 포맷
UART3	RS485 (Modbus RTU)	산업용 PLC/SCADA 연동	9600~57600bps	Half-Duplex
USB (Optional)	Virtual COM	개발용 디버그	—	ST-LINK 통합
RTC Alarm + Wake-up	내부 이벤트	저전력 복귀 트리거	—	시스템 타이밍 기준

4. 데이터 흐름 요약

1	[센서 버스] → I <sup>2</sup> C / GPIO / SPI
2	↓
3	[MCU 내부 처리]
4	↓
5	[출력 채널]
6	├ BLE (MQTT)
7	├ RS485 (Modbus)
8	├ SD Card (FATFS)
9	└ OLED / UART 로그

- 센서 데이터 수집: 주기적으로 I<sup>2</sup>C/SPI 버스에서 값 읽기
- Task 간 전달: FreeRTOS Queue를 통해 DisplayTask, ControlTask로 전달
- 외부 송신: BLE → MQTT Broker, RS485 → Field Device
- 로컬 저장: FATFS를 통한 CSV 로그 기록

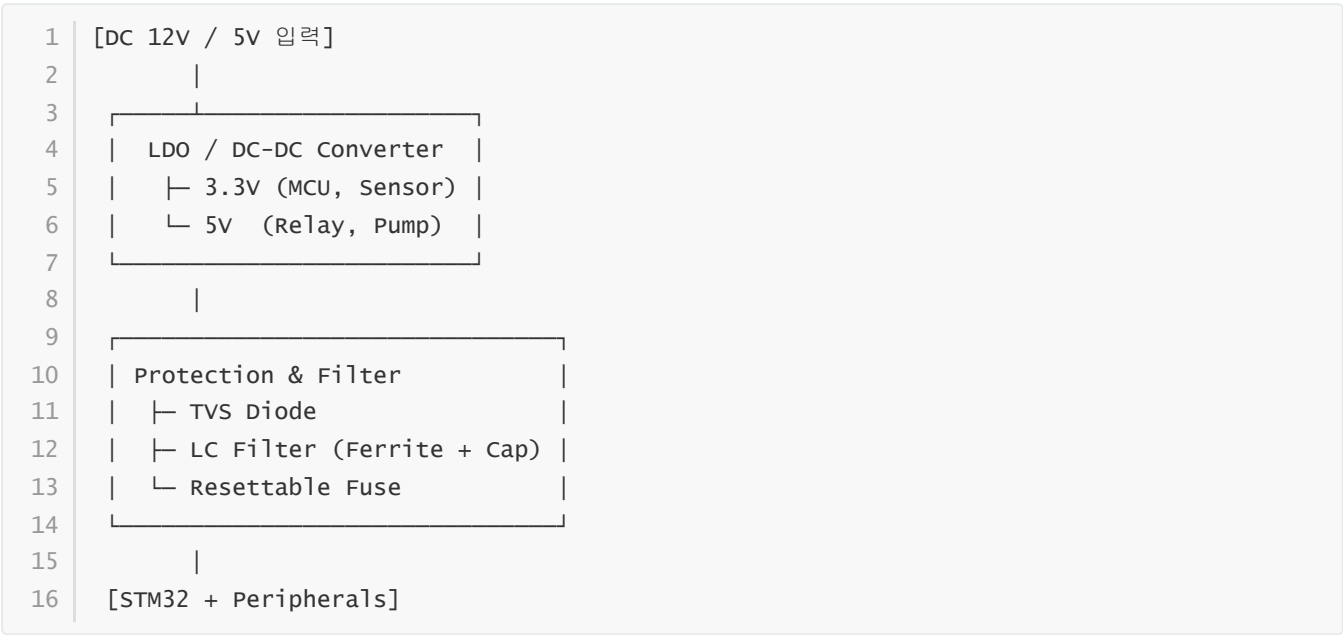
## 5. 설계 고려사항

- **버스 충돌 방지:** I<sup>2</sup>C 듀얼 센서(XSHUT) 시퀀스 초기화
- **우선순위 제어:** UART DMA 전송 시 Task 전환 최소화
- **전력 관리:** 통신 모듈 Sleep 제어 (BLE, RS485 Transceiver)
- **데이터 무결성:** CRC / Checksum 기반 오류 검출
- **통신 확장성:** BLE → Wi-Fi 또는 LTE 모듈 교체 가능

## 15.4 전원 및 저전력 설계

STM32 기반 시스템의 안정적 전원 공급과 저전력 동작을 위해 전원 라인 설계, LSE 기반 RTC 유지, 센서 단위 전력 차단, FreeRTOS Tickless Idle, STOP 모드 복귀 절차를 통합적으로 구성한다.

### 1. 전원 구조 개요



- **3.3V 라인:** MCU, OLED, I<sup>2</sup>C 센서, EEPROM, BLE 등
- **5V 라인:** 릴레이, 모터, 초음파 트리거 구동부
- **GND 분리:** 디지털 GND / 아날로그 GND 분리 후 단일점 접지

### 2. LSE 기반 RTC 및 백업 전원

구성 요소	핀	역할	비고
LSE Crystal	PC14 / PC15	32.768 kHz 기준 클럭	RTC 정확도 향상
Backup Battery	VBAT 핀	RTC 유지 전원	CR2032 또는 슈퍼커패시터
Power Domain	VDD / VBAT	메인전원 / 백업전원	자동 스위칭

- LSE 구동 후 HAL\_RTC\_Init() 시 LSEDriveConfig 최적화 필요
- VBAT 전원은 1.65V 이상 유지 시 RTC 동작 지속

### 3. 저전력 동작 단계

모드	소비전류	동작 블록	특징
RUN 모드	5~20 mA	CPU + 모든 Peripherals	정상 동작
SLEEP 모드	수 mA	CPU OFF / Peripherals ON	__WFI() 진입
STOP 모드	수십 µA	RTC + SRAM 유지	LSE 기반 알람 Wake-up
STANDBY 모드	수 µA 이하	RTC 단독 유지	전원 재부팅 필요

#### 주요 API

- HAL\_PWR\_EnterSLEEPMode(PWR\_MAINREGULATOR\_ON, PWR\_SLEEPENTRY\_WFI)
- HAL\_PWR\_EnterSTOPMode(PWR\_LOWPOWERREGULATOR\_ON, PWR\_STOPENTRY\_WFI)
- HAL\_PWR\_EnterSTANDBYMode()

### 4. FreeRTOS Tickless Idle 적용

- 동작 개요:  
시스템이 Idle Task 상태에 진입하면 SysTick을 일시 정지하고, RTC 또는 LPTIM을 기반으로 Sleep 기간을 계산 후 자동 복귀한다.
- CubeMX 설정 경로:  
FreeRTOS → Config Parameters → Enable Tickless Idle = True
- Tickless Idle 과정
  1. 모든 Task가 Blocked 상태 → Idle Hook 진입
  2. eTaskConfirmsSleepModeStatus() 호출 → Sleep 가능 판정
  3. SysTick 정지 후 STOP 모드 진입
  4. RTC Alarm 또는 외부 인터럽트 발생 시 복귀

### 5. 센서 단위 전력 차단

센서	제어핀	제어 방식	저전력 효과
VL53L0X	XSHUT (PA8)	LOW → 전원 차단	I2C 버스 절전
HX711	SCK (PA0)	Clock 정지	내부 ADC Sleep
OLED (SSD1306)	RST (PB4)	Sleep Command	표시 오프
초음파 센서	TRIG OFF	내부 회로 차단	대기전류 제거

→ 필요 Task 시작 전 Enable, 종료 후 Disable

## 6. 복귀 절차 (RTC Alarm → Wake-up)

1. RTC Alarm 설정
- HAL\_RTC\_SetAlarm\_IT() 호출
  - 알람 발생 시 RTC\_Alarm\_IRQHandler() 트리거
2. STOP 모드 진입
- HAL\_PWR\_EnterSTOPMode() 실행
  - LSE 및 RTC 유지
3. 복귀 시퀀스
- 클럭 재설정 (SystemClock\_Config())
  - 알람 인터럽트 → RTCTask Notify
  - 센서 초기화 → 측정 루틴 수행

## 7. 전력 최적화 요약

항목	방법	기대 효과
MCU 클럭 스케일링	PLL → HSI 전환	동작전류 20~30% 감소
주변장치 Sleep 제어	GPIO로 전원 스위칭	센서 전류 절감
FreeRTOS Tickless Idle	Idle 시 SysTick 정지	CPU Sleep 비율 향상
RTC Wake-up 제어	STOP 모드 복귀 타이밍 제어	장시간 배터리 구동
전원라인 필터링	LC / TVS 구성	노이즈로 인한 재기동 방지

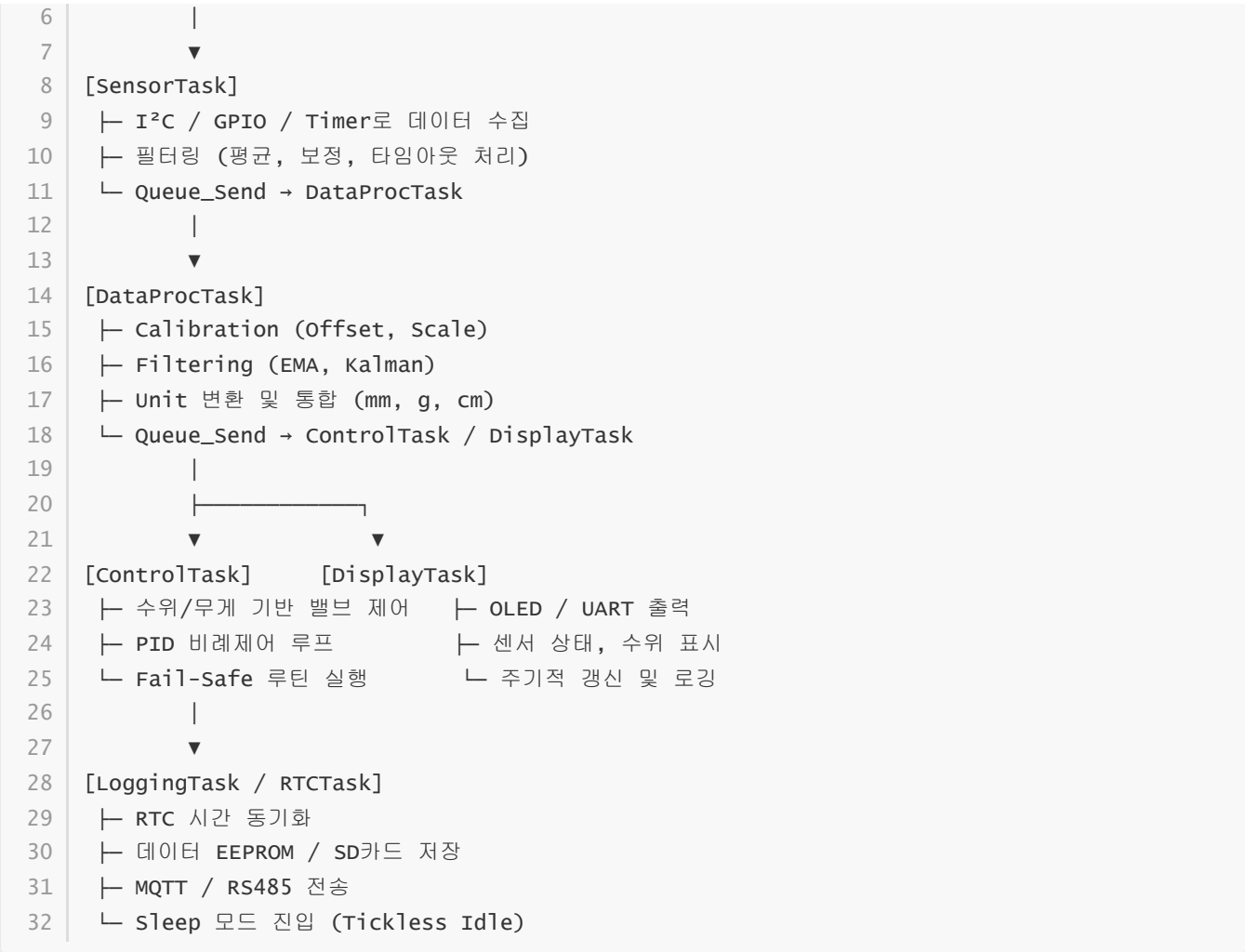
## 15.5 데이터 흐름

STM32 기반 계측 제어 시스템의 전체 데이터 흐름은 **센서 입력** → **데이터 처리** → **제어 및 표시** → **저장/통신**의 4단계로 구성된다.

이 구조는 FreeRTOS 멀티태스킹 환경에서 병렬적으로 수행되며, 각 Task 간에는 **Queue, Semaphore, EventFlag**로 동기화된다.

### 1. 전체 데이터 경로 개요

1	[센서 입력부]
2	├─ HC-SR04 (거리)
3	├─ HX711 (무게)
4	├─ VL53L0X (ToF)
5	└─ Capacitive Water Sensor (수위)



## 2. 센서 입력 단계

센서	인터페이스	주요 처리	출력 변수
HC-SR04	GPIO + Timer Input Capture	Echo 펄스폭 측정 → 거리 계산	distance_cm
HX711	GPIO Bit-bang	24bit ADC 읽기 → 무게 변환	weight_kg
VL53L0X	I²C	ToF 데이터 획득 → 거리 보정	tof_mm
Water Sensor	I²C	정전식 수위 단 탐색 → 수위 변환	level_mm

- 각 센서는 `SensorTask` 에서 주기적으로 스캔
- ADC/I²C 통신은 DMA 기반 비차단 모드로 수행
- 측정 실패 시 Retry + Timeout 처리

## 3. 데이터 처리 단계

### 1. 보정(Calibration):

EEPROM에 저장된 Offset / Scale 값을 로드하여  
센서 Raw 값을 실측 단위로 변환

```
1 | real_value = (raw - offset) * scale
```

### 2. 필터링:

- Moving Average (N=8)
- Exponential Smoothing ( $\alpha=0.3$ )
- Kalman Filter (Q, R 값 튜닝 기반)

### 3. 통합:

- 초음파와 수위센서 평균 → 신뢰도 기반 융합
- 무게 + 수위 연동 → 밀도 계산 (옵션)

### 4. 출력 데이터 구조체 예시:

```
1 | typedef struct {  
2 |     float distance_cm;  
3 |     float level_mm;  
4 |     float weight_kg;  
5 |     float temperature_C;  
6 |     uint32_t timestamp;  
7 | } SensorData_t;
```

## 4. 제어 및 표시 단계

### • ControlTask:

- 목표 수위와 현재 수위를 비교
- 오차에 비례하여 릴레이 또는 MOSFET 제어
- 예:

```
1 | error = target_level - level_mm;  
2 | control = Kp * error;  
3 | if (control > threshold) Pump_ON();  
4 | else Pump_OFF();
```

- 긴급 조건 (Fault, Timeout 등) 발생 시 Fail-Safe 모드 진입

### • DisplayTask:

- UART로 실시간 상태 로깅
- OLED(SSD1306)에 주요 정보 표시
- RTC 시각 및 시스템 상태 동시 출력

5. 저장 및 통신 단계

모듈	기능	인터페이스
EEPROM (24C02)	보정 데이터 저장	I <sup>2</sup> C
SD Card (FATFS)	센서 로그 CSV 저장	SPI
ESP32 MQTT Bridge	IoT 서버 전송	UART
RS485 (Modbus RTU)	산업 장비 연동	UART2

- RTCTask는 실시간 타임스탬프 제공
- LoggingTask는 FreeRTOS Queue를 통해 주기적 데이터 수집 및 기록

6. 저전력 및 슬립 루틴

- 모든 센서 데이터 전송 완료 후 SensorTask 정지
- DisplayTask 업데이트 종료 → OLED Sleep 명령 전송
- RTC 알람 설정 (HAL\_RTC\_SetAlarm\_IT())
- HAL\_PWR\_EnterSTOPMode() 진입
- 알람 발생 시 Wake-up → 클럭 재설정 → 측정 재개

7. 데이터 흐름 요약

단계	주요 동작	대표 함수	출력
Sensor	센서값 읽기	HAL_I2C_Mem_Read, HAL_GPIO_ReadPin	Raw Data
Processing	보정 + 필터링	Calibrate(), Filter_Update()	정제 데이터
Control	제어 루프 실행	PID_Update(), Pump_Control()	제어신호
Display	시각화	OLED_Print(), printf()	상태 표시
Logging	저장/통신	f_write(), MQTT_Publish()	데이터 기록
Sleep	전력 절감	HAL_PWR_EnterSTOPMode()	저전력 유지

이 구조는 전체 시스템이 센서 데이터 중심으로 순환하며, FreeRTOS Task 간 Queue를 통해 완전 비동기 동작하도록 설계되어 있다.

## 15.6 확장 및 유지보수

본 절에서는 STM32 기반 계측 제어 시스템을 장기적으로 운용하거나, 기능을 확장할 때 고려해야 할 **모듈화 설계**, **유지보수 절차**, **펌웨어/하드웨어 확장성**을 다룬다.

목표는 시스템이 현장 환경 변화, 센서 교체, 통신 확장, 펌웨어 업데이트 등에 쉽게 대응할 수 있도록 하는 것이다.

### 1. 모듈화 및 계층 구조 유지

#### ● 계층적 구조 설계 원칙

1	Application Layer
2	└─ Control Logic / User Interface
3	└─ Communication (BLE, RS485, MQTT)
4	└─ Logging / Data Processing
5	Driver Layer
6	└─ HAL Wrappers (GPIO, ADC, I2C, SPI)
7	└─ Sensor Drivers (Ultrasonic, HX711, ToF)
8	└─ Peripheral Handlers (RTC, DMA, Timer)
9	Hardware Layer
10	└─ STM32 MCU + Power
11	└─ Sensor / Actuator Board

- **Application Layer**는 구체적인 하드웨어에 의존하지 않음
- **Driver Layer**는 HAL 함수만 사용하고 직접 레지스터 접근 최소화
- **Hardware Layer** 변경 시, Driver Layer만 수정해 전체 시스템 유지

### 2. 신규 센서 및 통신 모듈 추가

확장 대상	인터페이스	추가 위치	주요 수정
수질 센서 (pH, EC)	ADC / UART	SensorTask	Sensor_Init(), Sensor_Read()
Wi-Fi (ESP8266 / ESP32)	UART	CommTask	Comm_Init(), MQTT_Handler()
LoRa 통신 모듈	SPI / UART	CommTask	LoRa_Send(), LoRa_Recv()
유량 센서 / 밸브 제어	Timer Input / GPIO	ControlTask	Flow_Calc(), Valve_Control()

- Task 구조를 그대로 유지하고 **Queue 기반 메시지 전달**로 확장성 확보
- 센서 추가 시 `typedef struct SensorData_t` 확장 → 기존 로직 영향 최소화

### 3. 펌웨어 유지보수 정책

#### 1. 버전 관리 (Git 기반)

- `/Core/Src`, `/Drivers`, `/Config` 디렉토리 단위로 커밋
- 주요 변경 시 `CHANGELOG.md` 기록
- 태그 예시:

```
1 | v1.0.0 - 기본 계측 및 제어 동작
2 | v1.1.0 - EEPROM 보정 기능 추가
3 | v1.2.0 - BLE 연동 및 데이터 로그 확장
```

#### 2. 빌드 설정 자동화 (CubeIDE + Makefile)

- 빌드 프로필 분리 (`Debug`, `Release`, `LowPower`)
- Debug 모드: `UART printf`, `Live watch` 활성화
- Release 모드: `assert` 제거, 최적화 플래그 `-O2`

#### 3. 유닛 테스트 및 시뮬레이션

- PC 기반 시뮬레이션 코드 (CMock, Unity) 활용
- 주요 함수별 입력-출력 검증 (`Filter_Update()`, `PID_Update()`)

### 4. 하드웨어 변경 대응

변경 항목	영향	대응 방법
MCU 교체 (F103 → F303)	레지스터 차이, 클럭 설정	HAL CubeMX 재생성 후 핀맵 재배치
전원회로 변경	ADC 참조 전압 변동	<code>ADC_Calibration()</code> 재실행
센서 보드 교체	I2C 주소 변경	<code>i2c_address.h</code> 수정
외부 스토리지 교체 (SD ↔ EEPROM)	파일시스템 변경	<code>fatfs.c</code> , <code>eprom.c</code> 분리 설계

### 5. 오류 대응 및 진단 절차

#### 1. Watchdog 활성화 (IWDG, WWDG)

- 주요 Task 내 `Kick()` 루틴 삽입
- 비정상 루프 발생 시 자동 리셋

#### 2. UART Debug Port

- 실시간 로그(`printf()`)로 오류 코드 출력
- 예: `[ERROR][I2C][Sensor Read Timeout]`

#### 3. LED / OLED 상태 표시

- LED Blinking Code로 Fault 분류
- OLED에 "Sensor Fault", "Comm Fail" 등 출력

#### 4. RTC 기반 Fault Log

- EEPROM / SD에 Timestamp + ErrorCode 저장
- 향후 유지보수 시 오류 이력 분석

## 6. 펌웨어 업데이트 (Bootloader 구조)

- 내장 Bootloader (USART / USB DFU) 또는 사용자 정의 Bootloader (Flash 영역 분할) 적용 가능
- 구조 예시:

```
1 | [0x0800_0000] Bootloader (16KB)
2 | [0x0800_4000] Application (Main FW)
3 | [0x0803_FFFF] Config / Log
```

- UART 또는 BLE를 통한 펌웨어 교체 시 CRC32 검증 수행

## 7. 장기 운용 시 유지보수 체크리스트

점검 항목	주기	설명
센서 오프셋 재보정	3~6개월	환경 온도/습도 영향
RTC 오차 보정	1년	DS1307 등 $\pm 2\text{ppm}$ 확인
EEPROM 쓰기 횟수 관리	주기적	100k cycle 한계 고려
배터리 / 전원 회로 점검	6개월	전압 드롭 및 ESD 확인
펌웨어 버전 / 로그 백업	필요 시	Git / SD 카드 기록

## 8. 향후 기능 확장 방향

- AI 기반 이상 수위 탐지 (TinyML)
- BLE → Wi-Fi / LTE 전환 (클라우드 직접 업로드)
- 멀티센서 네트워크 구성 (RS485 Daisy Chain)
- 원격 OTA 업데이트 (ESP32 Gateway)
- Grafana + InfluxDB 연동을 통한 실시간 분석 강화

### ✅ 요약:

본 시스템은 계층적 구조, 모듈화된 드라이버, 큐 기반 FreeRTOS 설계 덕분에 하드웨어 교체나 기능 확장 시에도 **핵심 로직의 재사용성과 안정성**을 유지할 수 있다. 정기적인 유지보수와 로그 기반 진단 체계를 통해 **현장 신뢰성을 장기 확보**한다.