스마트축사 반응형 환경제어를 위한 RS485 Modbus RTU 통합 인터페이스 설계

전진효, 이명훈*

*국립순천대학교

jjhyo94@gmail.com, *leemh777@scnu.ac.kr

Design of an Integrated RS485 Modbus RTU Interface for Responsive Environmental Control in Smart Livestock Barns

Jin Hyo Jeon, Meong Hun Lee*

*Sunchon National Univ.

요 약

스마트축사 환경에서는 기침음과 같은 개체군 이상 징후 및 설비 이상음 등 시간 민감도가 높은 이벤트에 대해 환기·분무·경보 등의 환경제어가 즉시 반응하는 반응형 제어 요구가 증가하고 있다. 그러나 현장에서 널리 사용되는 RS485 기반 Modbus RTU 환경에서는 이벤트 전달 및 제어 연계를 위한 공통 레지스터 맵 정의가 부족하여, 이기종 장비 간 상호운용성 확보에 한계가 있다. 본 논문은 반응형 환경제어 구현을 위해 RS485/Modbus RTU 기반의 이벤트/트리거 레지스터 맵을 정의한다. 제안 방식은 기존 표준의 고정 주소 체계와 상태·제어권·경보 코드 구조를 유지하면서 Reserved 성격의 주소 구간을 활용해 확장성을 확보하고, opid 기반 갱신 규칙을 적용하여 트리거 중복 수행을 완화한다. 제안된 레지스터 맵은 통합제어기가 이벤트를 일관된 방식으로 해석하고 제어 동작으로 연계할 수 있는 최소 공통 인터페이스로 활용될 수 있다.

Ⅰ. 서 론

스마트축사는 환기, 급이·급수, 냉난방, 분무 등 다양한 설비가 통합제어기와 연동되어 운영되며, 축사 내부 상태 변화에 따라 적시에 제어가 수행되는 것이 생산성과 안전성에 직결된다. 최근에는 돼지의 기침음과 같은 개체군 이상 징후 또는 설비 이상음을 포함한 음향 기반 이벤트를 활용하여, 이상 상황을 조기에 탐지하고 환경제어를 즉시 트리거하는 반응형 환경제어 요구가 증가하고 있다[1][2][3].

현장 통신 환경에서는 RS485 기반 Modbus RTU가 널리 사용되나, 반응형 제어를 실제로 구현하기 위해서는 이벤트 정보를 제어기로 전달하고, 이를 제어 명령으로 연계할 수 있는 공통 레지스터 맵(데이터 모델) 정의가 선행되어야 한다[4]. 기존 표준안은 통합제어기(마스터) - 기기(슬레이브) 구조에서 레지스터 맵 확보를 자동등록 방식 또는 디폴트 레지스터 맵 방식으로 구분하고, 노드 정보 및 디바이스 코드 등 고정 주소 영역을 포함한 디폴트 레지스터 맵 구조를 제시한다. 또한 상태(status)·제어권(control)·경보(alert) 코드 체계와 제어 명령의 일관성 확보를 위한 opid 기반 처리 규칙을 제공하며, Reserved 구간을 통해 기능 확장이 가능하도록 구성되어 있다.

본 논문은 기존 표준 구조를 기반으로, 스마트축사 반응형 환경제어를 위한 RS485 Modbus RTU 레지스터 맵을 정의하는 것을 목표로 한다. 구체적으로, 음향 이벤트의 발생 정보를 전달하기 위한 이벤트 상태 레지스터 블록, 이벤트에 따른 제어 연계를 지원하는 트리거 레지스터 블록 설계, 기존 고정 주소 영역 및 코드 체계와의 정합성을 유지하는 주소 배치 원칙을 제시한다. 제안된 레지스터 맵은 현장 적용 시 통합제어기가 이벤트를 일관된 방식으로 해석하고 제어 동작으로 연결할 수 있는 최소 공통 인터페이스로 활용될 수 있을 것으로 기대된다[5].

Ⅱ. 반응형 환경제어를 위한 레지스터 맵 설계 요구사항

스마트축사 반응형 환경제어는 이벤트 발생을 빠르게 인지하고, 이를 제어 동작으로 연결하는 과정이 핵심이다. 현장에서는 RS485 기반 Modbus RTU가 널리 활용되나, 장비별 구현 차이로 인해 이벤트 정보 전달 방식과 제어 연계 방식이 일관되지 않은 문제가 발생한다. 따라서 본 연구는 반응형 제어 구현에 필요한 최소 정보를 Modbus RTU 레지스터로 정형화하고, 통합제어기가 이를 동일한 규칙으로 해석할 수 있도록 레지스터 맵 설계 요구사항을 정리한다.

첫째, 통합제어기가 폴링 방식으로도 이벤트 발생 여부를 즉시 판별할 수 있도록 이벤트 유무 및 갱신을 나타내는 최소 상태 필드를 제공한다. 둘째, 기존 표준 구조와의 정합성을 확보하기 위해 고정 주소 영역과 코드 체계(status/control/alert) 및 제어 명령 ID(opid) 기반 처리 개념을 유지한다. 셋째, 기존 레지스터 정의와의 충돌을 방지하기 위해 신규 기능(이벤트/트리거)은 Reserved 또는 신규 블록으로 분리 배치하여 후방 호환성과 확장성을 동시에 고려한다. 넷째, 제조사별 해석 차이를 최소화하기 위해 심각도/신뢰도 등 핵심 값은 정규화 스케일과 데이터 타입을 명확히 규정한다. 마지막으로, 이벤트 반복 보고 및 통신 재전송 상황에서도 제어가 중복 수행되지 않도록 트리거 처리에는 opid 기반 갱신 규칙을 적용할 수 있도록 설계한다.

이러한 설계 원칙을 바탕으로, 다음 장에서는 이벤트 감지 결과를 전달하기 위한 레지스터 블록과 환경제어 연계를 위한 트리거 레지스터 블록을 정의하고, 주소 배치 및 opid 기반 동작 규칙을 제시한다.

Ⅲ. 이벤트/트리거 레지스터 맵 정의

기존 RS485 Modbus RTU 기반 인터페이스 표준은 노드 식별 및 장비 구성을 확인하기 위한 고정 주소 영역(노드 정보, 디바이스 코드 목록 등)과 상태 정보 및 제어 정보 영역을 구분하여 정의하고, 일부 구간을 Reserved로 두어 확장 가능성을 확보한다. 또한 제어 명령은 operation과

제어 명령 ID(opid) 기반으로 구성되어, opid 갱신을 통해 명령 활성화를 판별할 수 있도록 설계되어 있다. 본 논문은 이러한 구조를 유지한 상태에서, 반응형 환경제어에 필요한 기능을 이벤트 상태 전달 레지스터 블록과 트리거(제어 연계) 레지스터 블록으로 정의한다. 특히 기존 정의와의 충돌을 최소화하기 위해, 제안 블록은 표준에서 확장 용도로 남겨둔 Reserved 성격의 구간을 우선 활용하는 방식으로 주소를 배치한다.

표 1. 이벤트 상태 레지스터 블록

레지스터 주소	필드	타입	설명
470	event_flag	uint16	0: 이벤트 없음, 1: 신규 이벤트 존재
471	event_code	uint16	이벤트 유형 코드
472	severity	uint16	이벤트 심각도(0~100 정규화)
473	confidence	uint16	이벤트 신뢰도(0~100 정규화)
474	event_count	uint16	관측 윈도우 내 누적 발생 횟수
475-476	event_ts	uint32	마지막 이벤트 발생 시각
477	window_sec	uint16	관측 윈도우 길이(초)
478	channel_id	uint16	음향 채널/센서 ID
479-499	reserved	-	확장 Reserved

표 1은 통합제어기가 폴링 방식으로도 이벤트 발생을 빠르게 판별할 수 있도록, 이벤트 유무(event_flag)와 최소 메타(event_code, severity, confidence, event_ts)를 포함한다. 이벤트 값의 정규화(0~100)는 제조사별 해석 차이를 줄이기 위한 것으로, 추후 이벤트 유형이 확장되더라도 event_code 및 reserved 구간을 통해 확장 가능하도록 설계한다.

표 2. 트리거(제어 연계) 레지스터 블록

레지스터 주소	필드	타입	설명
650	operation	uint16	제어 동작 코드(ON/OFF 등)
651	target_dev	uint16	대상 디바이스 코드 (표준의 디바이스 코드 체계 기반)
652	action_level	uint16	제어 강도(0~100)
653-654	duration_sec	uint32	제어 지속시간(초)
655	control	uint16	제어권 상태 (LOCAL/REMOTE/MANUAL 등 표준 코드 체계 준용)
656	opid	uint16	트리거 명령 ID (갱신 시 신규 트리거로 처리)
657-699	reserved	-	확장 reserved

표 2는 이벤트에 따른 반응형 환경제어를 “제어 연계 트리거” 형태로 전달하기 위한 블록이다. 본 논문은 표준의 제어 처리 개념을 준용하여, operation과 opid를 함께 사용하고 opid의 갱신을 신규 트리거의 판별 기준으로 채택한다. 이를 통해 동일 이벤트의 반복 보고 또는 통신 재전송 상황에서도 트리거 중복 실행을 완화할 수 있다. 또한 target_dev는 표준의 디바이스 코드 체계와 연동되도록 하여, 통합제어기가 대상 장비를 일관된 방식으로 지정할 수 있도록 한다.

반응형 환경제어의 기본 동작은 다음과 같다. 통합제어기는 상태 정보 조회 과정에서 event_flag를 확인하고, event_flag가 1이면 이벤트 상세 필드를 읽어 이벤트를 해석한다. 이벤트가 제어 조건을 만족할 경우, 통합제어기는 트리거 블록에 operation 및 제어 파라미터를 기록한 후 opid를 이전 값과 다르게 갱신한다. 노드는 opid 변화에 따라 트리거의 유효성을 판단하고, 제어권(control) 값은 표준의 코드 체계를 준용하여 원격 제어 가능 여부를 일관되게 처리한다.

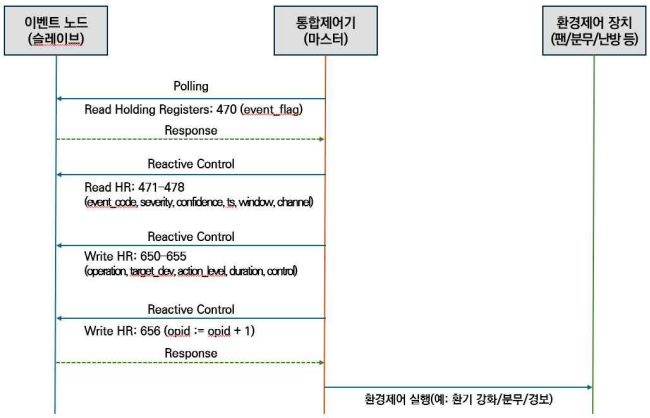

그림 1. 이벤트 폴링 및 opid 갱신을 통한 반응형 환경제어 시퀀스

그림 1은 상기 절차를 Modbus RTU 레지스터 접근 순서(470, 471 - 478, 650 - 655, 656)로 도시한 것이다.

Ⅳ. 결론

본 논문은 스마트축사에서 음향 기반 이벤트 등 시간 민감도가 높은 상황에 즉시 대응하기 위한 반응형 환경제어 구현을 목표로, RS485 Modbus RTU 기반의 이벤트/트리거 레지스터 맵을 정의하였다. 제안한 레지스터 맵은 기존 표준의 고정 주소 체계 및 상태·제어권·경보 코드 체계와의 정합성을 유지하면서, Reserved 성격의 주소 구간을 활용하여 신규 기능을 확장하는 방식으로 설계하였다. 또한 제어 연계 과정에서 opid 기반 갱신 규칙을 적용함으로써 트리거의 중복 수행을 완화하고, 통합제어기 관점에서 일관된 해석·연동이 가능하도록 구성하였다.

향후에는 제안한 레지스터 맵을 실제 축사 환경의 이기종 설비(환기, 분무, 냉난방 등) 및 다양한 이벤트 유형에 적용하여, 이벤트 발생부터 제어 반응까지의 지연 특성, 안정성, 운용 편의성을 중심으로 현장 실증을 수행할 필요가 있다. 또한 레지스터 기반 이벤트 표현을 상위 플랫폼 연동(게이트웨이/데이터 모델)까지 확장함으로써, 스마트축사 전반의 반응형 제어 서비스 고도화를 위한 기반으로 활용될 수 있을 것이다.

ACKNOWLEDGMENT

“본 연구는 2025년도 전라남도 재원으로 전남인재평생교육진흥원의 지원을 받아 수행되었습니다.”

참 고 문 헌

[1] Y. Chung, S. Oh, J. Lee, and D. Park, “Automatic Detection and Recognition of Pig Wasting Diseases Using Sound Data in Audio Surveillance Systems,” Sensors (Basel), vol. 13, no. 10, pp. 12929 - 12942, 2013, doi: 10.3390/s131012929.

[2] H. Chae, J. Lee, J. Kim, S. Lee, J. Lee, Y. Chung, and D. Park, “Novel Method for Detecting Coughing Pigs with Audio-Visual Multimodality for Smart Agriculture Monitoring,” Sensors, vol. 24, no. 22, 7232, 2024, doi: 10.3390/s24227232.

[3] X. Wang, Y. Yin, X. Dai, W. Shen, S. Kou, and B. Dai, “Automatic detection of continuous pig cough in a complex piggery environment,” Biosystems Engineering, vol. 238, pp. 78 - 88, 2024, doi: 10.1016/j.biosystemseng.2024.01.002.

[4] S. Elamanov, H. Son, B. Flynn, S. K. Yoo, N. Dilshad, and J. Song, “Interworking between Modbus and internet of things platform for industrial services,” Digital Communications and Networks, vol. 10, no. 2, pp. 461 - 471, 2024, doi: 10.1016/j.dcan.2022.09.013.

[5] I. Ungurean, “Integrating Fieldbus and Data-Centric Middleware: An STM32 Modbus Master Gateway for DDS-Based IIoT Systems,” Technologies, vol. 13, no. 11, 526, 2025, doi: 10.3390/technologies13110526.