

Home Assistant 자동화를 위한 시각 편집 환경 구현*

서미현^{1*}, 문현아¹, 최광훈¹, 박승찬², 창병모³
전남대학교¹, 이글루시큐리티², 숙명여자대학교³

{seomihyeon,hamoon72,kwanghoon.choi}@jnu.ac.kr¹, seungchan.park@igloo.co.kr², chang@cs.sookmyung.ac.kr³

Implementation of a Visual Editing Environment for Home Assistant Automation

Mi-Hyeon Seo^{1*}, Hyeon-Ah Moon¹, Kwanghoon Choi¹, Seungchan Park², Byeong-Mo Chang³
Chonnam National University¹, IGLoo Corp², Sookmyung Women's University³

{seomihyeon,hamoon72,kwanghoon.choi}@jnu.ac.kr¹, seungchan.park@igloo.co.kr², chang@cs.sookmyung.ac.kr³

요약

본 논문은 스마트홈 플랫폼 Home Assistant의 자동화 설정 언어인 YAML을 대상으로, 시각적 블록 편집과 텍스트 코드 간의 양방향 변환을 지원하는 편집 시스템을 설계·구현하였다. 기존 Home Assistant 자동화는 YAML 문법을 직접 작성해야 하므로 비전문가가 접근하기 어렵다. 본 연구는 Smart Block 기반 설계를 확장하여 Home Assistant 구조에 맞춘 Blockly 기반 시각 편집기(UI)를 개발하였으며, 이를 통해 사용자가 블록을 이용해 자동화를 구성하거나 기존 YAML을 불러와 블록 형태로 재편집한 뒤 다시 코드로 저장할 수 있도록 하였다. 검증 결과, 블록을 구성해 생성한 자동화의 변환 정확도는 100%, 공식 문서 예제를 블록으로 복원한 뒤 재생성한 YAML의 정확도는 80%로 나타났다. 이는 일부 미구현 블록으로 인한 제한에서 기인한 것으로, 전반적으로 시스템의 변환 안정성과 실용성을 확인하였다. 제안된 시스템은 SmartThings 전용이었던 기존 Smart Block의 단방향 구조를 Home Assistant 환경에 맞춘 양방향 편집 모델로 확장한 사례로, 향후 블록 범위 확장과 사용자 실험을 통해 비전문가 친화적 자동화 도구로 발전할 가능성을 제시한다.

1. 서론

스마트홈 기술의 확산과 함께 사용자가 직접 자동화 규칙을 정의하는 홈 자동화(Automation) 수요가 증가하고 있다. Home Assistant는 대표적인 오픈소스 기반 플랫폼으로[1], GitHub Octoverse 보고서에 따르면 2023년에 이어 2024년에도 가장 많은 기여자가 활동하는 오픈소스 프로젝트 상위권에 올랐다[2]. Home Assistant에서는 텍스트 기반 YAML 구성 파일을 통해 자동화를 작성한다. 그러나 YAML은 들여쓰기 오류와 복잡한 조건 중첩으로 인해 비전문가가 사용하기 어렵다. 이는 스마트 홈 자동화를 보다 쉽게 생성 및 관리할 수 있는 시각적 환경의 필요성이 커지는 점을 보여준다.

이전 Smart Block 연구는 SmartThings에서 동작하는 SmartApp(Groovy) 코드를 자동 생성하기 위한 시각적 블록 인터페이스를 제공하며, ECA(Event-Condition-Action) 블록 기반 자동화를 구현할 수 있다[4].

본 연구의 목표는 다음과 같다. 첫째, Home Assistant 자동화 구조에 맞춘 전용 Smart Block 인터페이스를 설계한다. 둘째, 기존 Home Assistant YAML 텍스트 코드를 블록 구성으로 복원하는 기능을 제공한다. 셋째, 실제 YAML 예제 30개를 통해 변환 정확성과 효용성을 평가한다.

2. 관련 연구

2.1 Home Assistant 자동화 구조

Home Assistant 자동화는 trigger-condition-action의 3계층

구조를 따른다.

```
- alias: "Light Control"
  trigger:
    - platform: state
      entity_id: light.living_room
      to: "on"
  condition:
    - condition: time
      after: "22:00:00"
  action:
    - service: light.turn_off
      target:
        entity_id: light.kitchen
```

2.2 Smart Block 연구

Smart Block은 삼성 SmartThings용 ECA 시각 언어로, Blockly[3]를 활용하여 IoT 규칙을 구성하는 연구다[4]. Smart Block은 Groovy 기반 SmartApp 코드를 생성하는 단방향 구조(one-way)이며, 기존 코드의 시각적 편집 기능은 제공하지 않았다. 본 연구는 Smart Block의 개념을 참고하되, SmartThings가 아닌 Home Assistant의 YAML 자동화 구조에 맞춘 블록 시스템을 새롭게 설계하였다.

2.3 기존 Smart Block과 본 연구의 비교

본 연구는 Smart Block의 ECA 기반 구조는 유지하되 Home Assistant 자동화 체계에 적합한 독립적 블록 모델을 구축한다는 점에서 차별성을 갖는다.

*이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-정보보호핵심원천기술개발사업의 지원을 받아 수행된 연구임(RS-2025-25394739)임. 본 연구는 한국인터넷진흥원(KISA)-정보보안 특성화대학 지원사업의 지원을 받아 수행된 연구임.

항목	Smart Block	본 연구
대상 플랫폼	SmartThings	Home Assistant
자동화	SmartApp (Groovy)	Automation (YAML)
표현 방식		
기본 모델	ECA 기반 블록	ECA 기반 블록
코드 생성	Groovy 코드 생성	YAML 코드 생성
역변환 기능	지원되지 않음	지원함

표1. 이전 연구와 본 연구의 비교

3. 시스템 설계 및 구현

시스템은 Home Assistant 자동화 YAML과 시각적 블록 간 양방향 변환을 지원하도록 설계되었다.

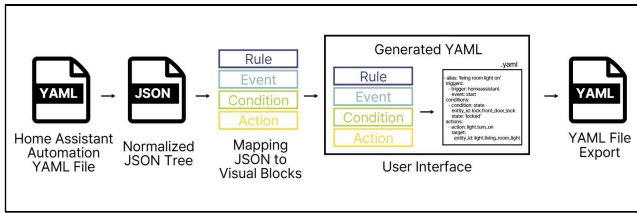


그림 1. 시스템 구조도

그림 1은 Home Assistant 자동화 YAML 파일을 정규화하여 JSON으로 변환한 후, Rule-Event-Condition-Action 구조의 시각 블록으로 매핑하고, 편집된 블록을 다시 YAML 코드로 직렬화하여 Export하는 양방향 데이터 흐름을 나타낸다.

3.1 Home Assistant 전용 스마트 블록 구현

본 연구에서는 기존 SmartThings용 Smart Block을 기반으로, Home Assistant의 자동화 구조에 최적화된 스마트블록을 새롭게 설계하였다.

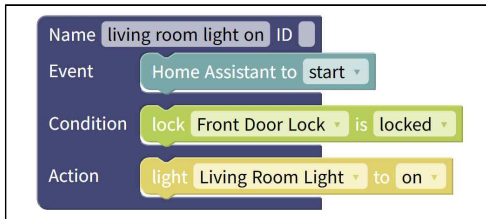


그림 2. Home Assistant 전용 스마트블록 예시

그림 2는 본 시스템에서 구현한 Home Assistant 전용 스마트블록을 이용해 구성한 자동화 예시를 보여준다. 이 예시는 Home Assistant가 시작할 때, 현관문이 잠겨 있는 경우 거실 조명을 켜는 자동화를 시각적으로 표현한 블록이다. 각 블록은 Home Assistant 자동화의 기본 구성요소인 Event, Condition, Action의 3계층 구조로 구성되며, 자동화 규칙의 실행 흐름을 직관적으로 이해할 수 있도록 설계되었다.

이 블록 구조는 Smart Block의 ECA(Event-Condition-Action) 개념을 계승하면서도, Home Assistant의 YAML 문법(trigger, condition, action)과 일대일 매핑이 가능하도록 설계되었다. Event 블록은 시스템 시작 이벤트를 트리거로 설정하고, Condition 블록은 Front Door Lock 엔티티의 상태가 locked일 때만 실행되도록 조건을 구성한다. Action 블록은 Living Room Light 엔티티에 대해 on 서비스를 호출하여 조명을 점등하도록 한다.

또한 블록의 속성값(entity_id, state, service 등)은 Home Assistant 내 실제 엔티티 목록과 연동되어 자동 완성(dropdown) 방식으로 제공되기 때문에, 사용자는 엔티티 이름이

나 서비스 호출 문법을 정확히 기억하지 않아도 자동화를 손쉽게 구성할 수 있다.

따라서 사용자는 오타자나 문법적 실수 없이 자동화를 정의할 수 있으며, 정의된 블록은 이후 YAML 생성기(yamlGenerator)를 통해 유효한 Home Assistant 자동화 YAML 코드로 직렬화된다.

3.2 사용자 인터페이스 (UI)

사용자는 블록을 조합하여 규칙을 구성할 수 있다.



그림 3. 사용자 인터페이스

그림 3은 시스템의 사용자 인터페이스를 나타낸다. 우측 블록 영역에서는 사용자가 블록을 조합하여 자동화 규칙을 구성할 수 있으며, 좌측 코드 영역에는 이에 대응하는 YAML 코드가 실시간으로 생성된다. 또한 사용자는 Import/Export 기능을 통해 YAML 파일을 읽어오거나 저장할 수 있다.

3.3 스마트블록 코드를 Home Assistant 자동화 YAML 텍스트 코드로 변환

본 절에서는 Blockly 기반 스마트블록으로 구성된 자동화 규칙을 Home Assistant 자동화 YAML 코드로 직렬화하는 과정을 다룬다. 기존 Smart Block의 단방향 코드 생성 구조를 확장하여, Home Assistant의 trigger-condition-action 문법에 맞는 변환 구조를 설계하였다. YAML 생성기(yamlGenerator)는 각 블록의 속성을 자동화 요소와 일대일로 매핑하며, Home Assistant Automation에 따라 2칸 들여쓰기와 하이픈 리스트를 사용하고 불필요한 기본값은 생략하도록 하였다.

이벤트 블록은 시스템 시작, 상태 변화, 센서, 시간 등 트리거를 trigger절로 변환하고, 조건 블록은 논리 구조(and, or, not)를 유지한 채 condition: state 또는 numeric_state 구문으로 직렬화된다. 액션 블록은 도메인별(light, lock, 등) 서비스 호출을 action: <domain>.<service> 형식으로 변환하며, UI의 if-then-else 구조는 Home Assistant의 choose-default 구문으로 대응된다. 이 과정을 통해 사용자가 구성한 블록은 Home Assistant 자동화 문법에 맞게 정규화된 YAML로 생성된다.

3.4 ECA 기반 스마트 블록

본 연구의 스마트블록은 IoT 자동화의 핵심 모델인 ECA(Event-Condition-Action) 구조를 기반으로 설계되었다. ECA 모델은 특정 이벤트(Event)가 발생하면 조건(Condition)을 평가하고, 조건이 충족될 때 지정된 행동(Action)을 수행하는 규칙 체계로, IoT 자동화 언어의 표준적 설계 패턴으로 널리 사용된다 [4][5].

이 개념은 IOTA: A Calculus for Internet of Things Automation[5]에서 제시한 이론적 자동화 모델과 동일한 원리를 따르며, Home Assistant의 자동화 구조('trigger-condition-action')도 ECA 모델에 기반한다[1]. 기존 Smart Block[4]은 ECA 개념을 SmartThings 플랫폼의 Groovy 코드 생성기로 구현

하여, 이벤트 감지 → 조건 판별 → 행동 실행의 절차를 시각적 블록으로 표현하였다. 그러나 단방향 코드 생성만 지원했기 때문에 작성된 코드를 다시 블록으로 편집하거나 수정하는 것은 불가능하였다. 본 연구에서는 이 구조를 계승하며 Home Assistant의 YAML 문법에 맞추어 ECA 기반 스마트블록을 재설계하였다.

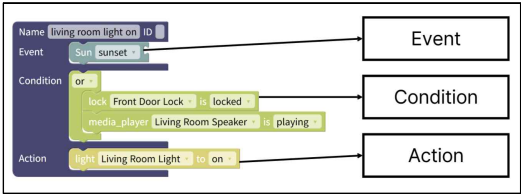


그림 4. ECA 기반 스마트블록 구조

3.5 Home Assistant 자동화 YAML 텍스트 코드를 스마트 블록 코드로 변환

본 절에서는 Home Assistant 자동화 YAML 파일을 시각적 스마트블록 형태로 변환하는 Import 경로를 설명한다.

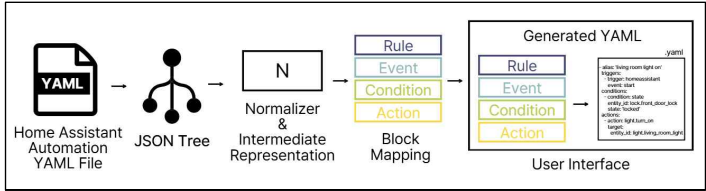


그림 5. YAML → 블록 변환 구조

그림 5는 YAML 파일이 블록 구조로 재구성되는 과정을 보여준다. 시스템은 YAML 코드를 파싱하여 JSON 트리로 변환하고, 정규화기(Normalizer)를 통해 값을 일관된 형식으로 표준화한다. 정규화된 데이터는 중간 표현(IR)로 저장된 후, 각 요소(trigger, condition, action)가 대응되는 블록으로 자동 매핑된다. 이 과정에서 블록의 속성(entity_id, state, service, target)은 JSON의 값으로 채워지며, 결과적으로 사용자는 기존 YAML 자동화를 블록 편집기 내에서 시각적으로 확인하고 수정할 수 있다.

4. 검증

본 장에서는 제안한 시스템의 양방향 변환 정확성과 호환성을 검증하기 위해 두 가지 검증을 수행하였다.

4.1 검증1 - 블록 기반 양방향 변환 검증

첫 번째 검증은 사전 정의의 엔티티 집합을 기반으로 15개의 대표적인 YAML 자동화 시나리오를 선정하여 제안 시스템 내에서 직접 블록을 생성한 후, (1)블록 → (2) YAML 변환, (3) YAML → (4) 블록 복원, (5) 다시 YAML 코드로 내보내는 과정을 수행하여 (1)과 (3)의 블록 구조 일치, (2)와 (4)의 코드 의미 동일성을 검증한다.

구분	전체 시나리오	일치 시나리오	일치율(100%)
블록 구조 동일성	15	15	100.0%
코드 의미 동일성	15	15	100.0%

표2. 블록 기반 양방향 변환 검증 요약

4.2 검증 2 - 레거시 HA YAML 코드 호환성 검증

두 번째 검증에서는 Home Assistant 공식 문서[1]의 예제를 기반으로 15개의 대표적인 YAML 자동화 코드를 선정·가공하여 (예: Motion → Living Light On, Window Open & Power > 1.0 → Notify), (1) 원본 YAML → (2) 블록 변환 → (3) 다시 YAML 생성 과정을 수행하였다. 이때 (1)과 (3)의 코드 구조 및 의미가 동일하지 비교하였다.

구분	전체 시나리오	일치 시나리오	일치율
코드 구조 및 의미 동일성	15	12	80.0%

표3. 레거시 HA YAML 코드 호환성 검증

아직 zone entity, 템플릿(template) 기반 표현, 특수 조건을 담당하는 일부 블록이 구현되지 않았기 때문에 총 3개의 실패 케이스가 발생했다.

5. 결론

본 연구에서는 Home Assistant 자동화의 YAML 편집 복잡성을 완화하기 위해, Smart Block을 확장한 Blockly 기반 시각 편집 시스템을 설계·구현하였다. 제안한 시스템은 Event-Condition-Action(ECA) 구조를 시각 블록으로 구성하고, YAML 코드와의 양방향 변환을 지원하여 비전문가도 손쉽게 자동화를 작성·수정할 수 있도록 한다. 검증 결과, 사용자가 생성한 블록과 재생성된 YAML 코드 간의 구조 및 의미가 대부분 일치하였으며, 일부 시나리오에서는 미구현 블록으로 인해 완전한 변환이 이루어지지 않았다. 향후 연구에서는 Home Assistant 자동화를 포괄할 수 있도록 블록 종류를 확장하고, 실제 환경에서의 사용자 실험을 통해 시스템의 실효성과 완성도를 높일 예정이다.

본 연구에서 구현된 시스템은 GitHub에 공개되어 있다.

https://github.com/seomihyeons/HA_smartblock

참고문헌

[1] Home Assistant Documentation, “Automations and Templates,” Home Assistant Docs, 2024. Available: <https://www.home-assistant.io/docs/automation/>

[2] GitHub, “Celebrating the GitHub Awards 2024 recipients,” <https://github.blog/news-insights/company-news/celebrating-the-github-awards-2024-recipients/>

[3] Google Developers, “Blockly: Visual Programming Editor,” Google Developer Resources, 2023. Available: <https://developers.google.com/blockly>

[4] Bak, N., Chang, B.-M., and Choi, K., “Smart Block: A visual block language and its programming environment for IoT,” Journal of Computer Languages, vol. 58, pp. 100968, 2020.

[5] Ciatto, G., Calvaresi, D., Mariani, S., and Omicini, A., “IOTA: A Calculus for Internet of Things Automation,” Journal of Logical and Algebraic Methods in Programming, vol. 119, pp. 100633, 2021. DOI: 10.1016/j.jlamp.2020.100633.