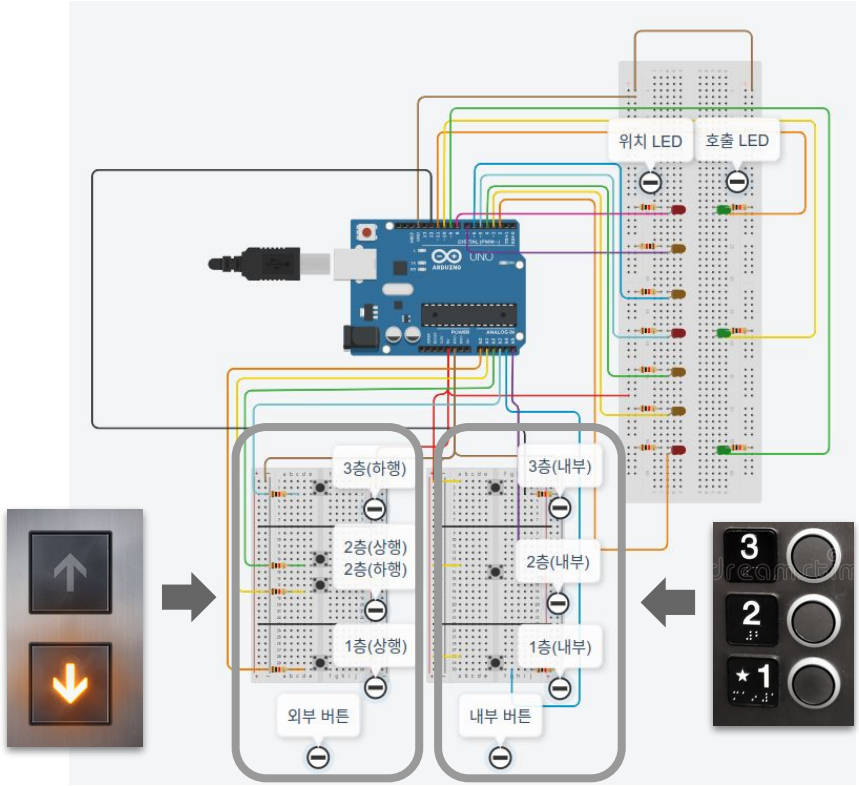
아두이노 엘리베이터 시스템

3층 엘리베이터 LED 시스템

발표자: 장진혁

1. 회로 구성

회로 구성



LED

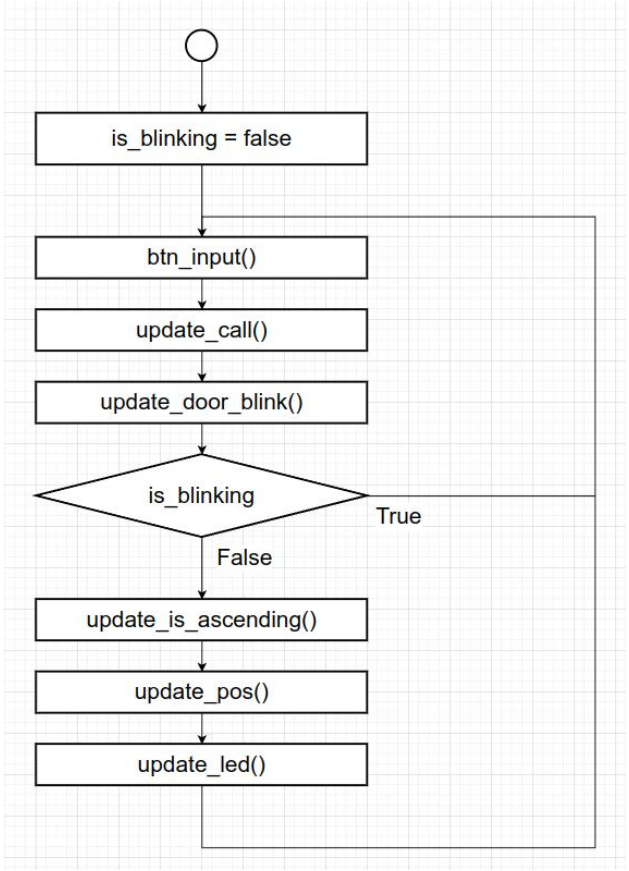
특정 층의 호출 상태를 표시하는 **호출 LED**,
엘리베이터의 현재 위치를 표시하는 **위치 LED**로 구성

버튼

외부에서 상행/하행을 선택할 수 있는 **외부버튼**,
내부에서 목적층을 선택할 수 있는 **내부버튼**으로 구성

2. 설계 특징

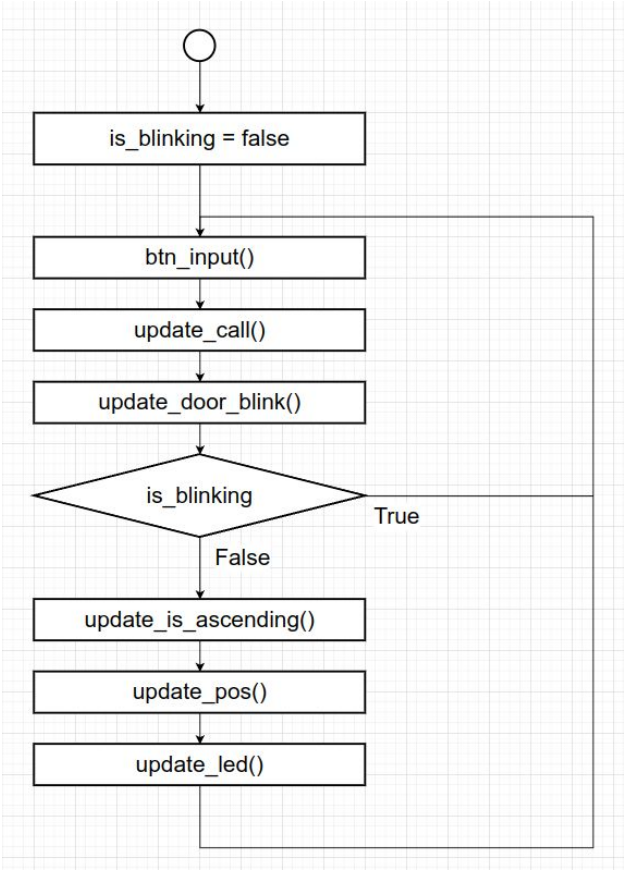
기능별 함수 분리



전체 시스템 동작을 **기능별**로 분리하여 구성함

기능별로 함수로 분리해
메인 루프는 흐름 위주로만 구성되도록 함

기능별 함수 분리



함수	설명
btn_input()	호출 등록
update_call()	호출 해제 + 문 열림 트리거
update_door_blink()	문 열림 표현을 위한 깜박임 동작
update_is_ascending()	방향 판단
update_pos()	위치 갱신
update_led()	LED 갱신

상태변수	설명
is_blinking	문 열림 상태 표현

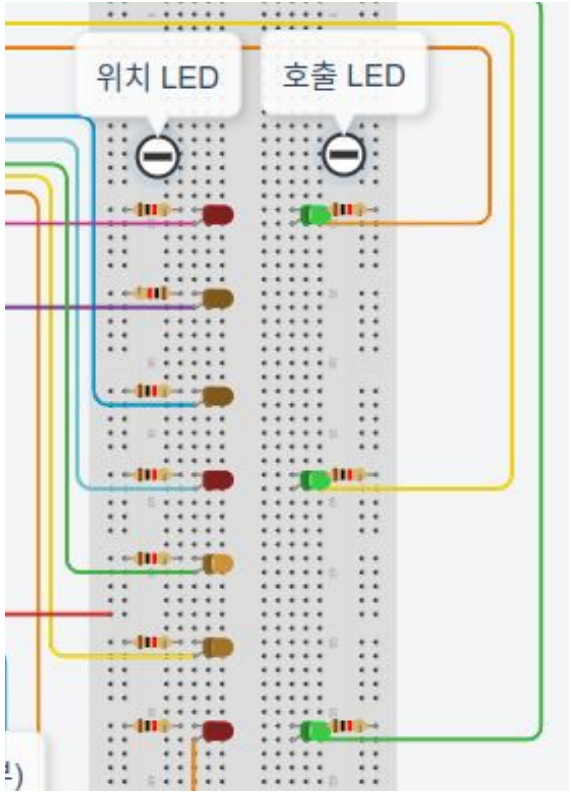
비동기 처리

동작	코드 구조
위치 이동	if (now - prev_move_time > MOVING_INTERVAL)
문 열림 LED 깜박임	if ((now / BLINK_INTERVAL) % 2) == 0)
문 열림 종료	if (now - blink_start_time < BLINK_DURATION)
LED 주기적 갱신	if (now - prev_led_time > MOVING_INTERVAL)

시스템의 모든 시간 제어를 **millis() 값(또는 차이)**로 처리

이동, 문 열림 LED 깜박임, LED 갱신 등 각각의 기능을 자신만의 주기 변수로 독립 제어

비트 플래그 기반 구조 - 상태 표현



	7	6	5	4	3	2	1	0
pos	0	0	0	0	0	**1**	0	0
up_call 🔼	0	0	0	0	0	0	0	**1**
down_call 🔽	0	**1**	0	0	**1**	0	0	0
inner_call 🧍	0	**1**	0	0	0	0	0	0
is_ascending	True							

여러 상태를 별도의 배열이나 조건문 없이
byte 변수 내에서 효율적으로 제어할 수 있어
메모리 사용과 코드 복잡도를 줄임

비트 플래그 기반 구조 - 조건 판단

예시

	7	6	5	4	3	2	1	0
pos	0	0	0	0	0	**1**	0	0
call	0	**1**	0	0	0	0	0	**1**

call = up_call | down_call | inner_call

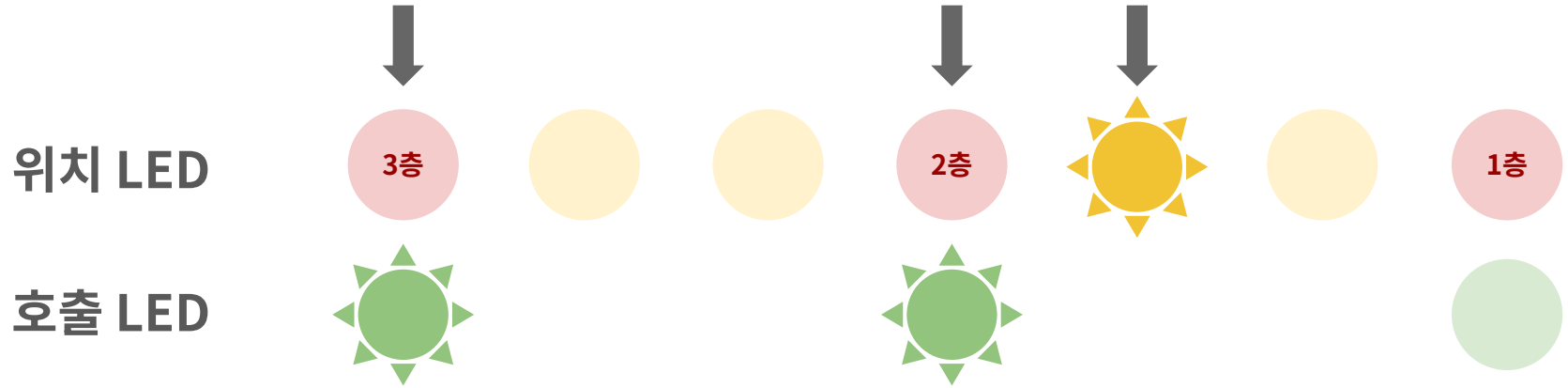
아래쪽에 호출이 존재 ↔ call % pos

위쪽에 호출이 존재 ↔ call > pos

현재 위치에 호출이 존재 ↔ call & pos

비트 플래그 기반 구조 - LED 출력

	7	6	5	4	3	2	1	0
pos	0	0	0	0	0	1	0	0
call	0	1	0	0	1	0	0	0



비트 플래그 기반 구조 - LED 출력

요소	내용
위치 LED	POS_LED_PINS[i] ← bitRead(pos, i)
호출 LED	CALL_LED_PINS[i] ← bitRead(call, 3*i)
애니메이션	pos = 0 ↔ pos_snapshot 토글만으로 구현

상태변수(pos, call)만 바꾸면 LED가 변경됨
: LED 제어를 위한 별도 로직이 없음 (단순 갱신 목적을 제외)

애니메이션도 pos 하나로 표현 가능
: LED 깜박임 애니메이션을 위한 별도 상태 변수를 만들지 않고,
기존에 존재하던 pos 값만 임시로 0과 snapshot 사이에서 토글시켜서 깜박임 구현

비트 플래그 기반 구조 - 장점

속도	**비트 연산은** 배열 인덱싱이나 조건문 분기보다 **빠름**. 상태 판단, 출력 여부 결정 등을 단일 연산으로 처리할 수 있음.
단순성	상태 표현이 **pos와 call로 통합** LED 출력이나 호출 처리 로직이 별도 조건 없이 **자동 연동**됨
확장성	코드의 구조를 그대로 유지하고 byte를 **unsigned long**으로 바꾸면 **최대 11층**까지 표현 가능하며, 시프트 레지스터를 활용하여 핀 수에 상관없이 LED를 확장 가능
일관성	호출 등록, 정지 판단, LED 제어 등 모든 기능이 비트 기반 연산으로 처리

3. 기능 설명

현실 엘리베이터와 규칙 비교

현실 엘리베이터와 규칙 비교

항목	현실 엘리베이터		본 프로젝트 구현 (User Requirement)
호출 취소	**외부**	**불가** (※ 일부 시스템에서는 가능)	**가능**
	내부	**불가**	**불가**
호출 해제 시점	**외부**	**이동 방향이 호출 방향과 같을 때**만 해제	**동일**
	내부	도착 시 **항상** 해제	**동일**
문 열림 중 입력	**가능**		**가능**
방향 전환	**현재 방향에 호출이 없으면** 반대 방향으로 전환		**동일**
호출층 도착시 동작	실제 **문 열림**		**LED 애니메이션** 처리

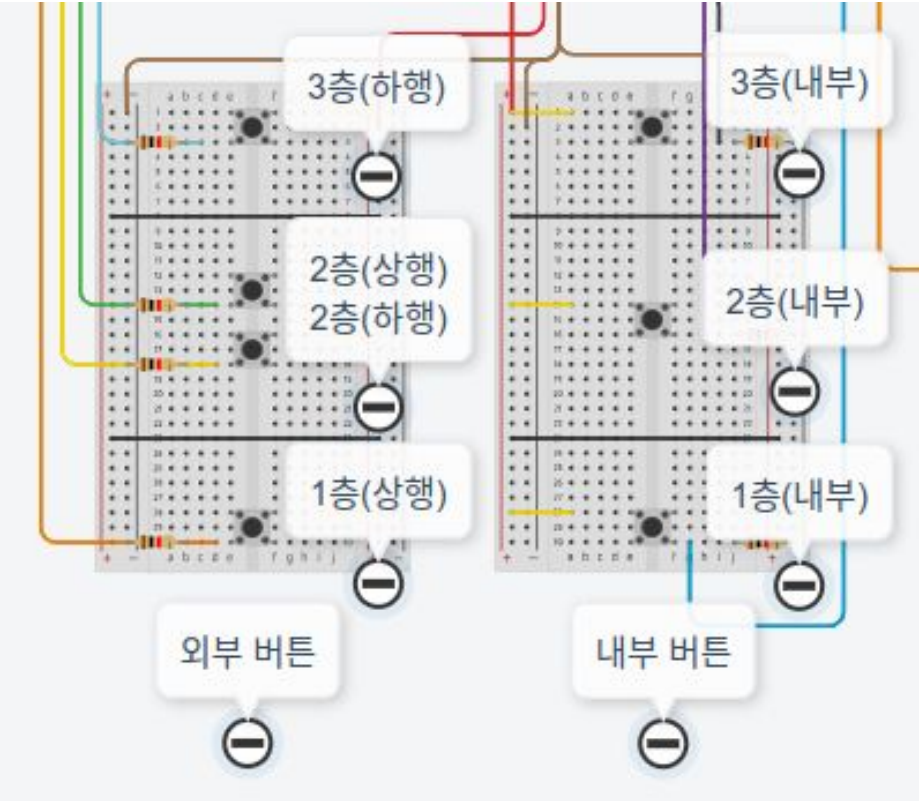
전체 기능

전체 기능

#	분류	System Requirements
1	**호출 등록 및 사용자 취소 처리**	1-1. 외부 및 내부버튼으로 호출을 등록하는 기능 1-2. 외부 버튼을 다시 눌러 호출을 취소하는 기능
2	**LED를 통한 상태 출력**	2-1. 현재 위치에 따라 위치 LED를 표시하는 기능 2-2. 호출 상태에 따라 호출 LED를 표시하는 기능
3	**방향 판단**	3-1. 위치 및 호출 상태를 기반으로 이동 방향을 판단하는 기능 3-2. 최상층/최하층 도달 시 자동으로 방향을 전환하는 기능
4	**이동 및 정지 판단**	4-1. 다음 이동 주기에 이동 여부를 판단하는 기능 4-2. 정규층이 아닌 층(층 사이)에서는 정지하지 않는 기능
5	**호출 해제**	5-1. 도착한 층에서 호출을 해제하는 기능
6	**문 열림 애니메이션**	6-1. 호출 처리 후 문이 열리는 동작을 표현하는 기능

1) 호출 등록 및 사용자 취소 처리

1-1) 외부 및 내부버튼으로 호출을 등록하는 기능

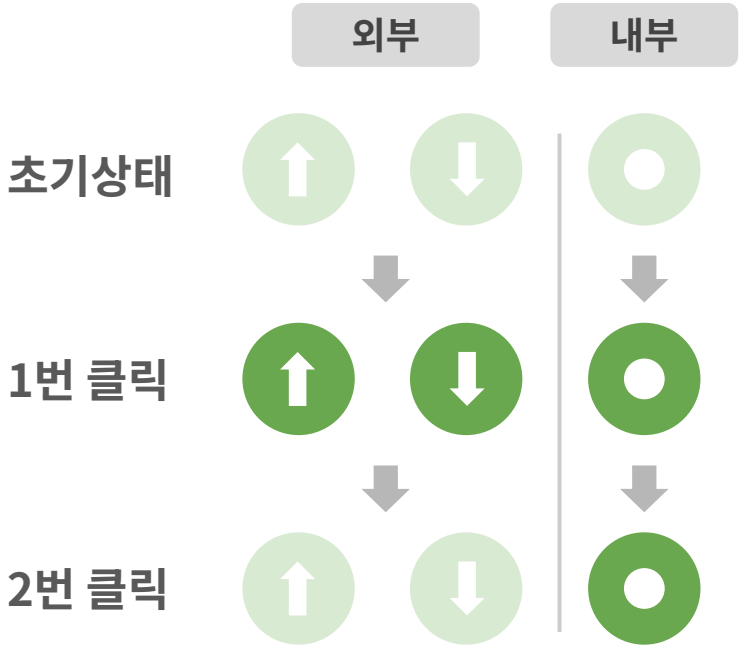


각 층의 **외부 호출 버튼(상행/하행)**을 누르면 엘리베이터가 해당 방향의 호출을 등록한다.

내부 버튼을 누르면 해당 층으로 이동 요청이 등록된다.

여러 호출을 동시에 등록할 수 있다.

1-2) 외부 버튼을 다시 눌러 호출을 취소하는 기능

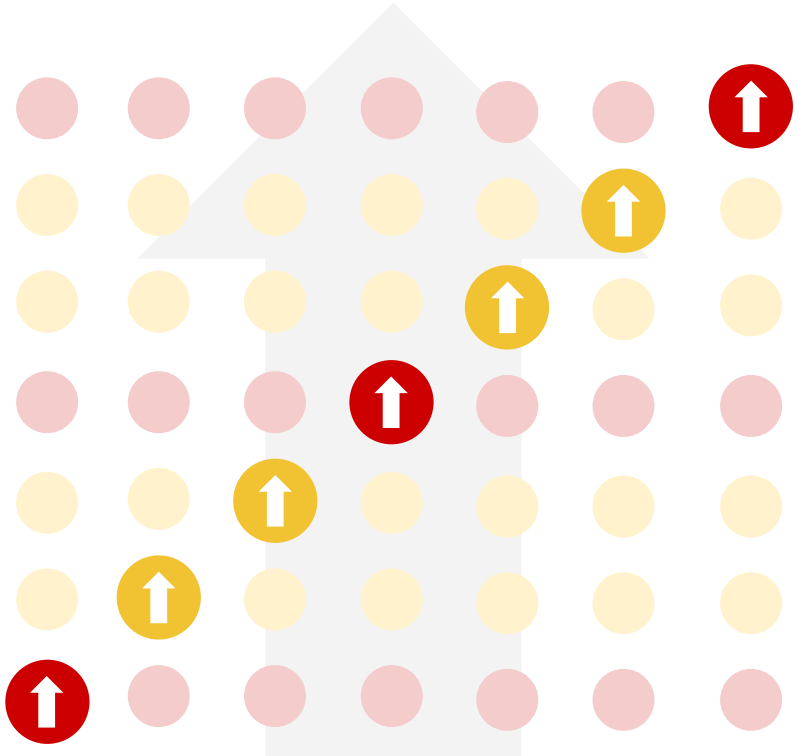


외부 호출 버튼을 다시 누르면 해당 호출이 취소된다.

내부 호출은 한 번 등록하면 취소할 수 없으며, 도착 시 자동으로 해제된다.

2) LED를 통한 상태 출력

2-1) 현재 위치에 따라 위치 LED를 표시하는 기능

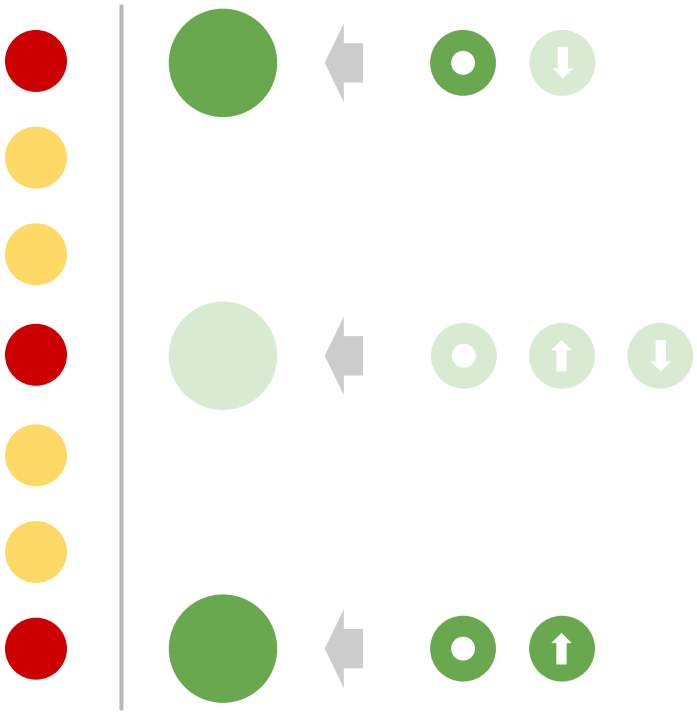


엘리베이터의 위치에 해당하는
위치 LED가 켜진다.

엘리베이터가 이동 중일 때는
현재 위치에 따라 LED 상태가 변한다.

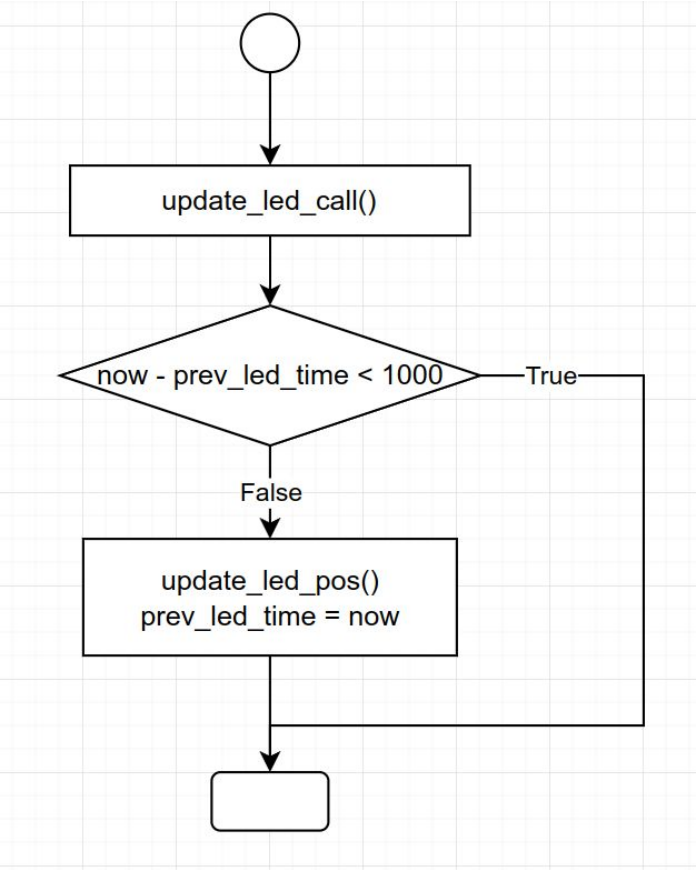
위치 LED는 **자동으로 갱신**된다.

2-2) 호출 상태에 따라 호출 LED를 표시하는 기능



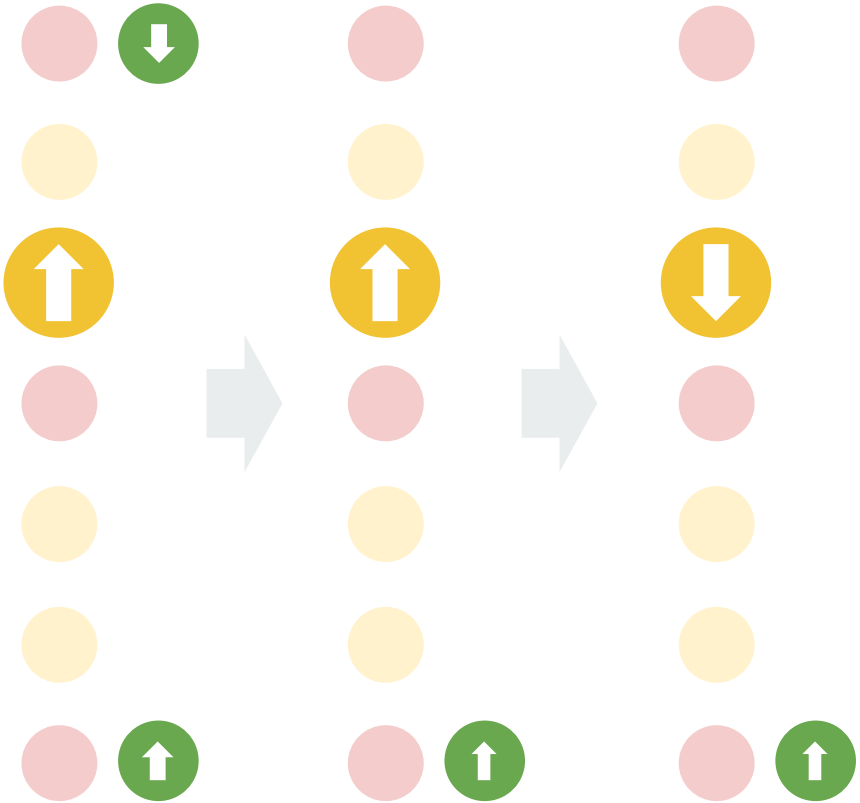
내부 호출, 상행 호출, 하행 호출이
모두 해제되어야
해당 층의 호출 LED가 꺼진다.

LED 갱신을 담당하는 함수 update_led()



3) 방향 판단

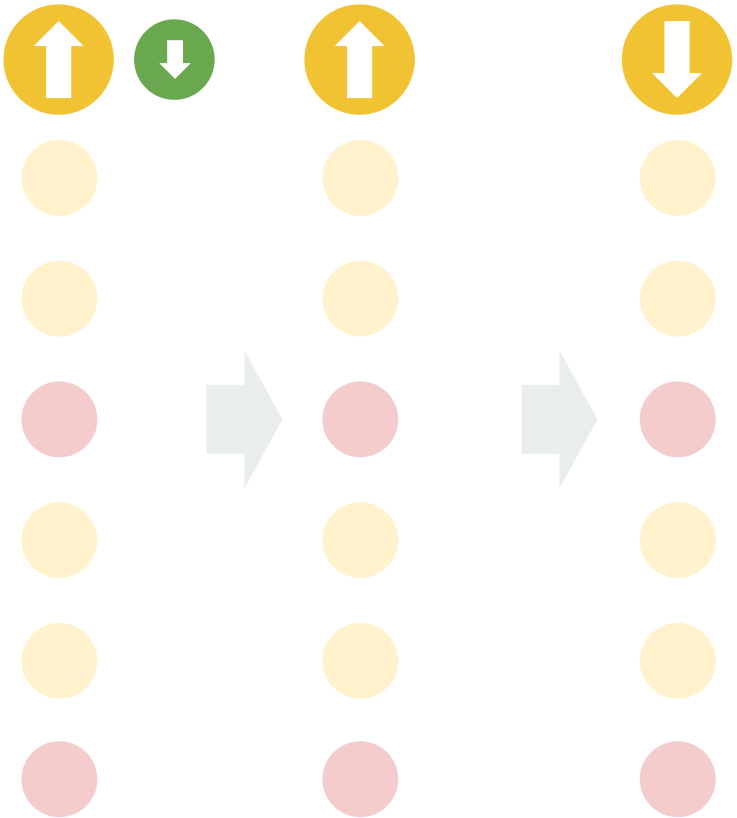
3-1) 위치 및 호출 상태를 기반으로 이동 방향을 판단하는 기능



현재 이동 방향을 기준으로
그 방향에 호출이 남아 있는지 판단

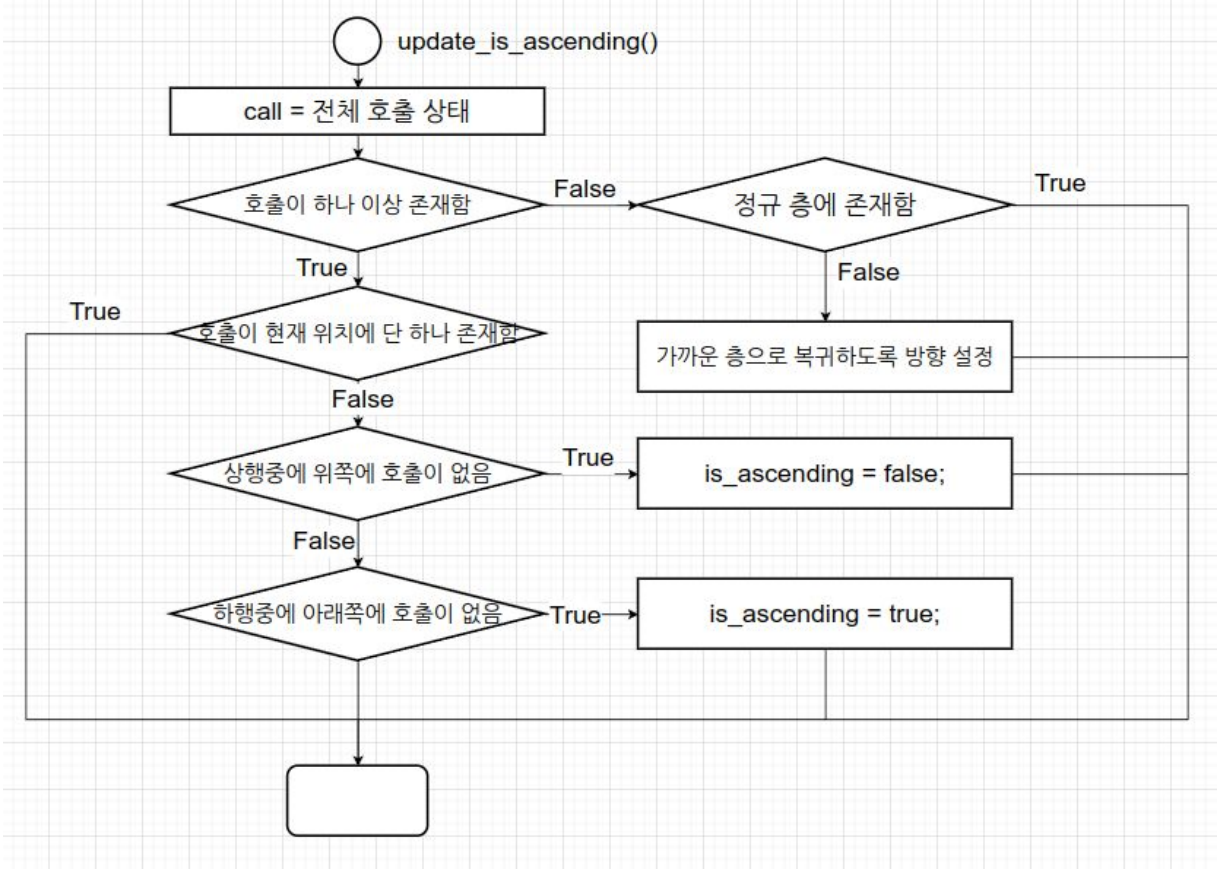
호출이 있으면 해당 방향을 유지하며 이동을 지속
호출이 없다면 방향을 전환하여 남은 호출을 처리함

3-2) 최상층/최하층 도달 시 자동으로 방향을 전환하는 기능



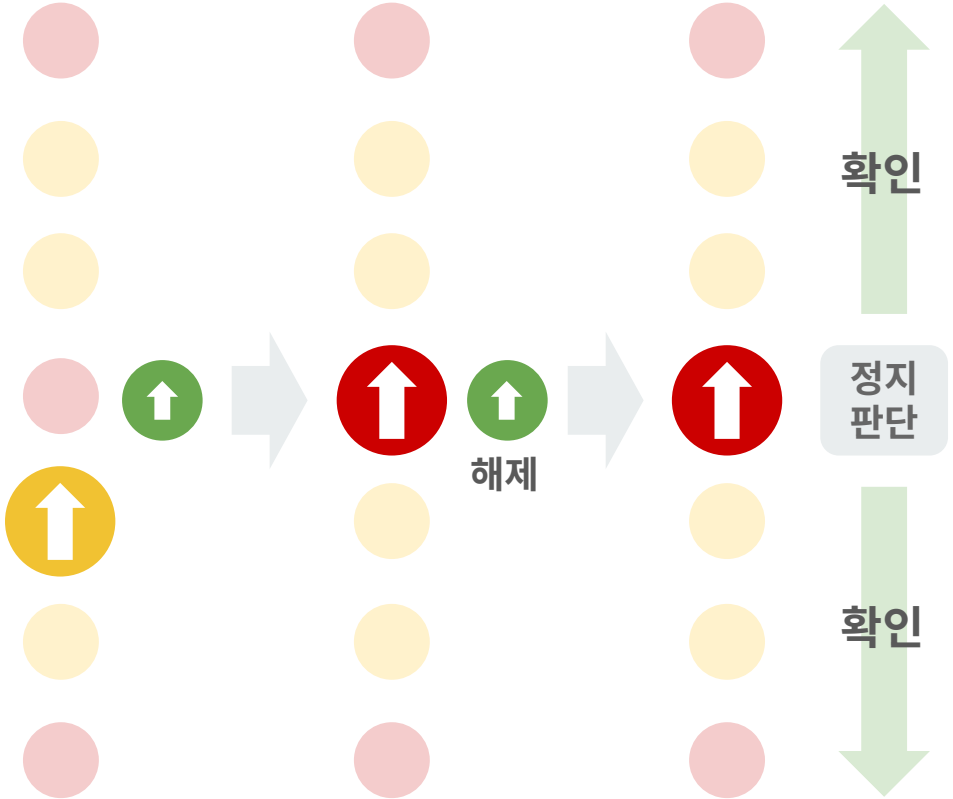
최상층/최하층 도달시
반대 방향 호출 유무와 관계 없이
무조건 방향을 전환한다.

방향 판단을 담당하는 함수 update_is_ascending()



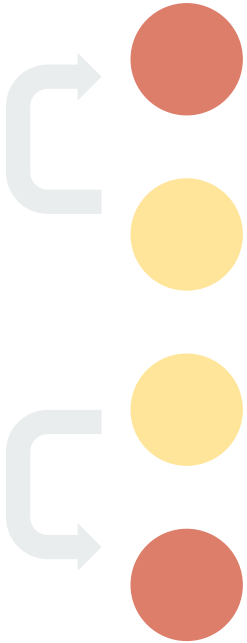
4) 이동 및 정지 판단

4-1) 다음 이동 주기에 이동 여부를 판단하는 기능



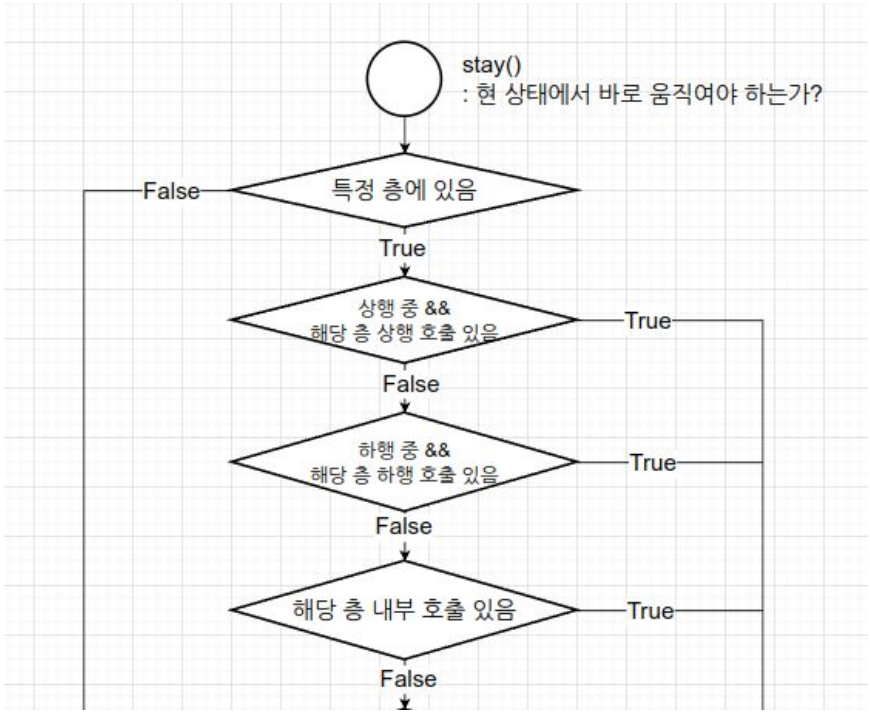
현재 위치와 **호출 상태**를 고려하여
다음 이동 주기에 이동해야 할지 여부를 결정
: stay() 함수를 통해 구현

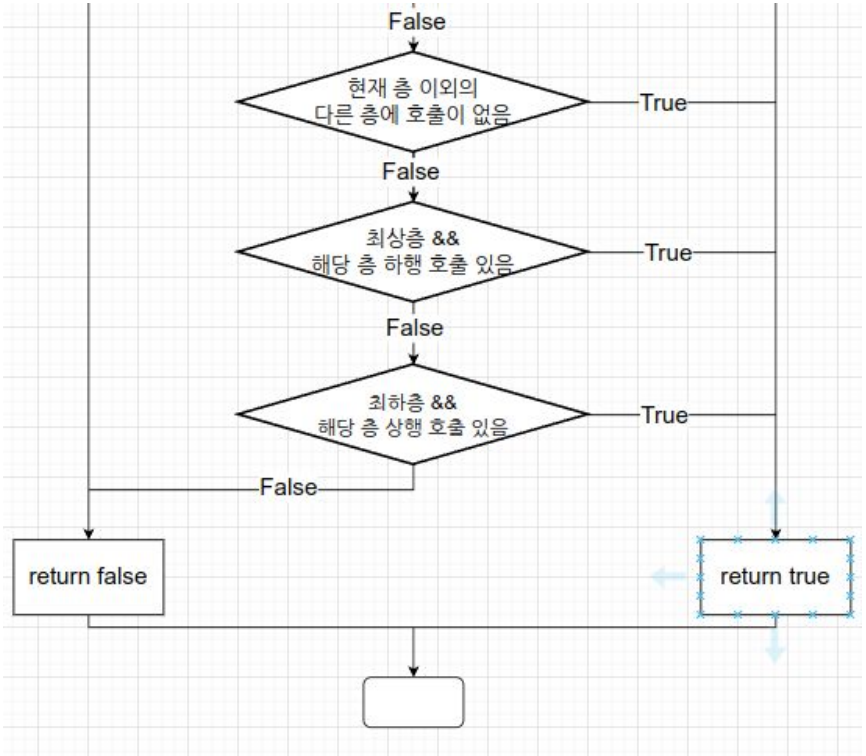
4-2) 정규층이 아닌 층(층 사이)에서는 정지하지 않는 기능



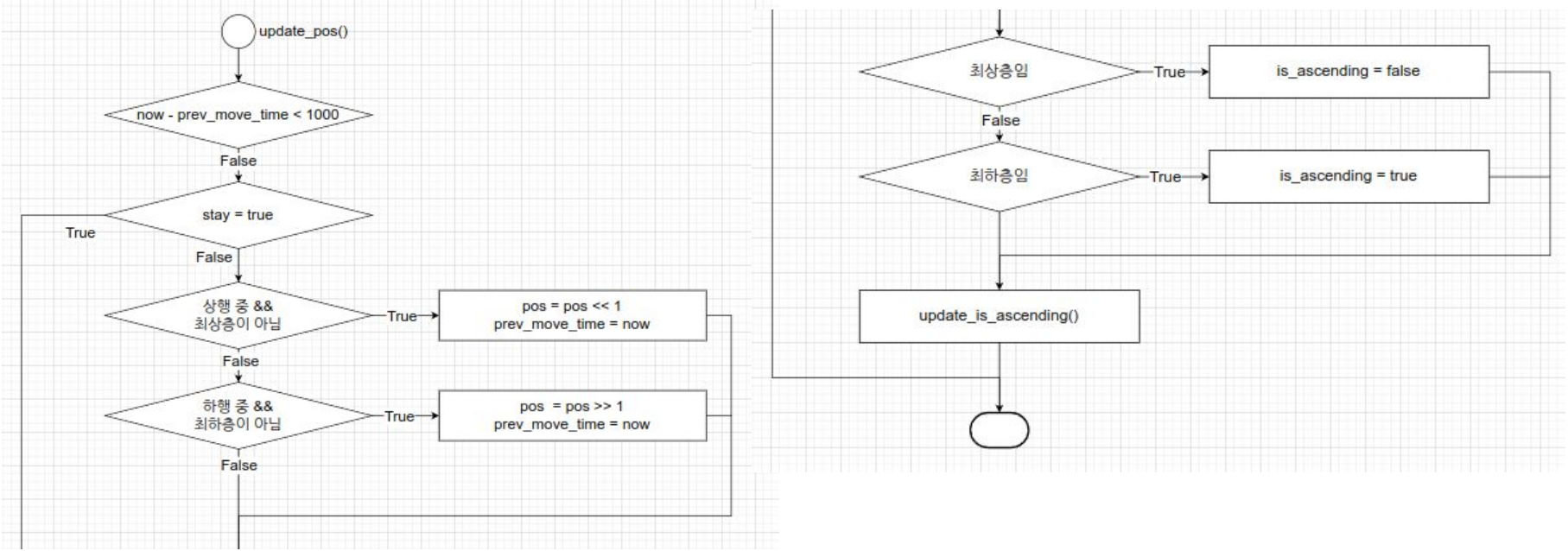
호출이 모두 사라진 경우에도,
현재 위치 기준 가장 가까운 층까지 이동한 뒤 정지
: 층 사이에서 정지하지 않음.

정지 판단을 담당하는 함수 stay()



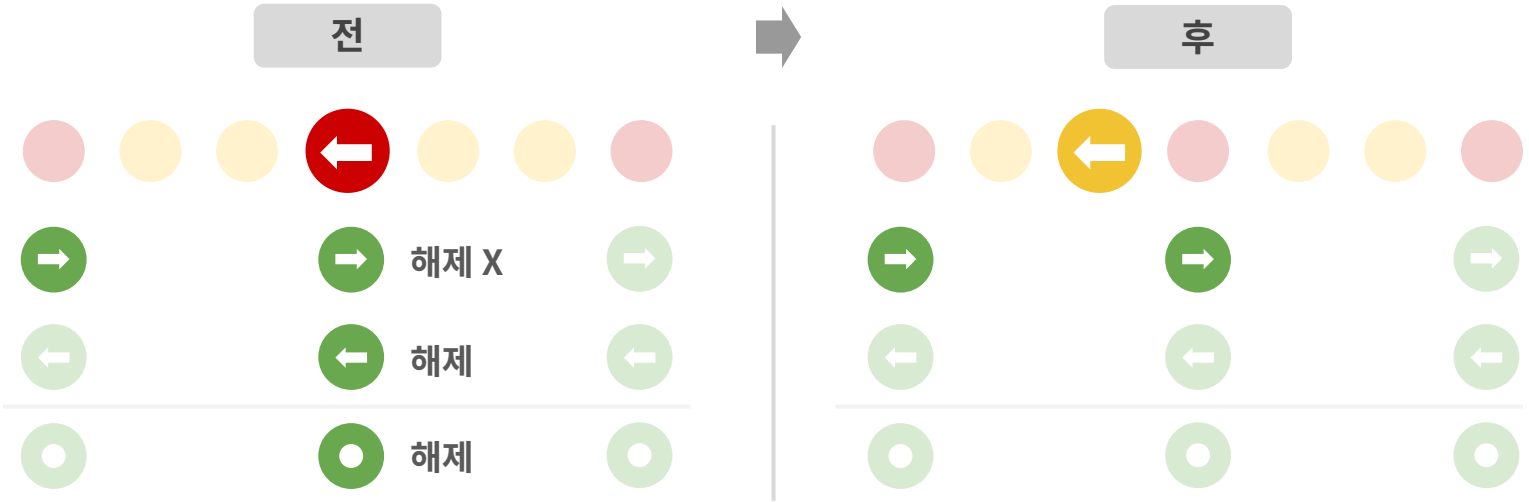


이동을 담당하는 함수 update_pos()



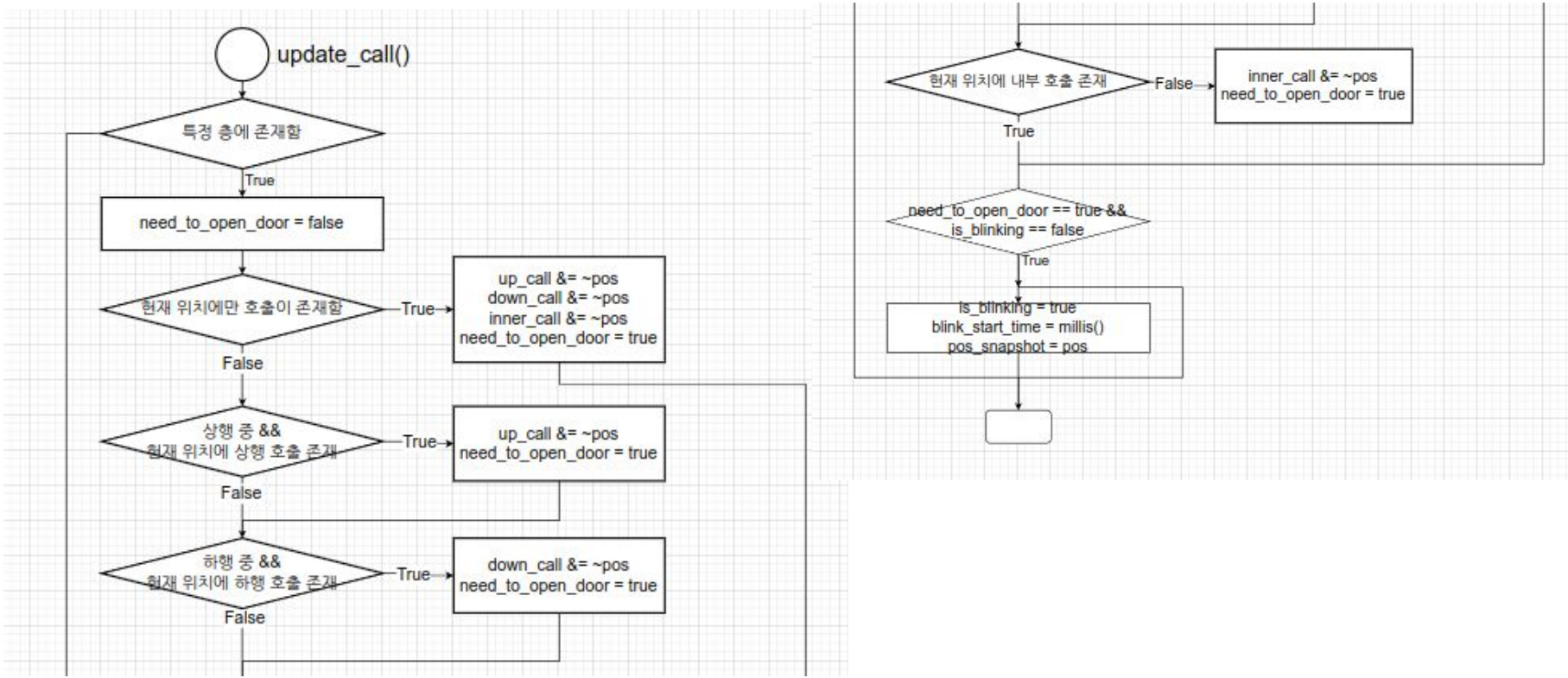
5) 호출 해제

5-1) 도착한 층에서 호출을 해제하는 기능



- **외부 호출**은 현재 이동 방향과 일치할 경우에만 해제된다.
- **내부 호출**은 도착 시 항상 해제된다.

호출 해제를 담당하는 함수 update_call()



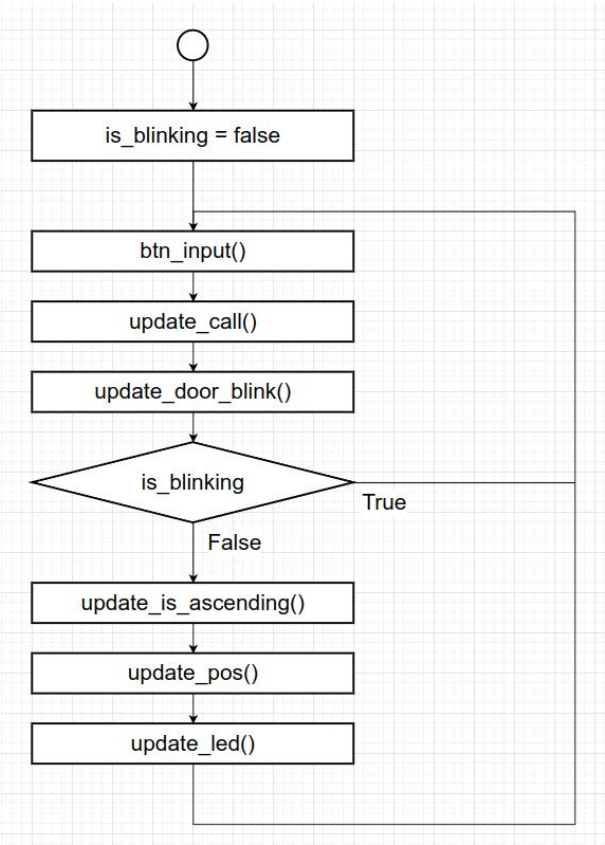
6) 문 열림 애니메이션

6-1) 호출 처리 후 문이 열리는 동작을 표현하는 기능

호출된 층에 도착하면 문이 열리는 것을 표현하기 위해 **LED가 일정 시간 깜빡인다.**

pos의 값을 미리 저장된 **pos_snapshot**과 **0**으로 주기적으로 변경하여 깜빡임을 구현한다.

6-2) 문 열림 중에도 호출 입력을 허용하는 기능

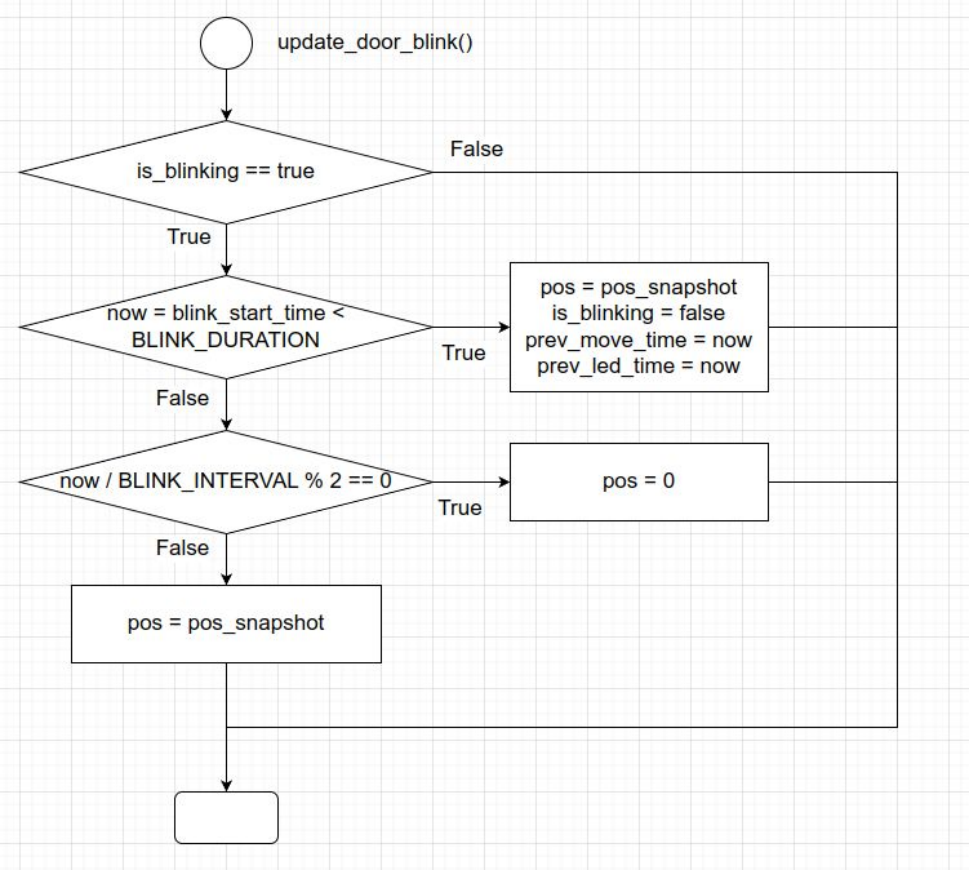


is_blinking = true

문이 열리는 동안 엘리베이터는 **이동하지 않는다**.

문이 열리는 동안 **새로운 호출 입력이나 호출 취소**를 할 수 있다.

문 열림 애니메이션을 담당하는 함수 update_door_blink()



4. 테스트

기본 동작 조건 테스트

TC_ID	시나리오	예상 결과	결과
TC_01	초기 상태 → 이후 호출 없음	정지 유지	✅
TC_02	1층에서 3층 하행 호출 → 이후 호출 없음	3층 도착 → 문 열림 → 하행 전환 → 정지 유지	✅
TC_03	3층에서 1층 상행 호출 → 이후 호출 없음	1층 도착 → 문 열림 → 상행 전환 → 정지 유지	✅
TC_04	정지 상태 → 현재 층 호출 등록	문 열림	✅
TC_05	층 사이에서 이동 중인 상태 → 모든 호출 취소	가장 가까운 층까지 이동 → 문 열림 없음	✅
TC_06	문 열림 중인 상태 → 다른 층 외부 호출 버튼 누름	문 열림 중 이동 없음 → 문 열림 후 이동	✅

입력 취소 테스트

TC_ID	시나리오	예상 결과	결과
TC_07	외부 호출 버튼 두 번 누름	호출 LED 점등 → 호출 LED 소등	✅
TC_08	내부 호출 버튼 두 번 누름	호출 LED 점등 → 호출 LED 유지 → 호출된 층 도착 → 호출 LED 소등	✅

연속 호출 테스트

TC_ID	시나리오	예상 결과	결과
TC_10	1층에서 내부 2층 호출 → 외부 1층 상행 호출	2층 도착 → 문 열림 → 하행 전환 → 1층 도착 → 문 열림	✅
TC_11	1층에서 내부 3층 호출 → 외부 2층 하행 호출	3층 도착 → 하행 전환 → 2층 도착	✅
TC_12	1층에서 외부 2층 상행 호출 → 내부 3층 호출	2층 도착 → 문 열림 → 3층 도착 → 문 열림	✅
TC_13	1층에서 내부 2층 호출 → 외부 3층 하행 호출	2층 도착 → 문 열림 → 3층 도착 → 문 열림	✅
TC_14	2층에서 외부 1층 상행 호출 → 외부 3층 하행 호출	1층 도착 → 상행 전환 → 3층 도착 → 문 열림	✅
TC_15	1층에서 내부 2층 호출 → 내부 3층 호출	2층 도착 → 문 열림 → 3층 도착 → 문 열림	✅

동시 호출 테스트

TC_ID	시나리오	예상 결과	결과
TC_16	2층에서 내부 3층 호출 → 2층 상행 + 1층 상행 동시 호출	3층 도착 → 하행 전환 → 2층 무시 → 1층 도착 → 상행 전환 → 2층 도착	✅
TC_17	3층에서 내부 2층 호출 → 외부 3층 하행 + 1층 상행 동시 호출	2층 도착 → 1층 도착 → 상행 전환→ 3층 도착	✅
TC_18	2층에서 외부 1층 상행 호출 → 외부 3층 하행 + 2층 내부 동시 호출	1층 도착 → 2층 도착 → 3층 도착 → 완료	✅