팀 프로젝트 계획서

IOT 프로그래밍 - 팀 프로젝트

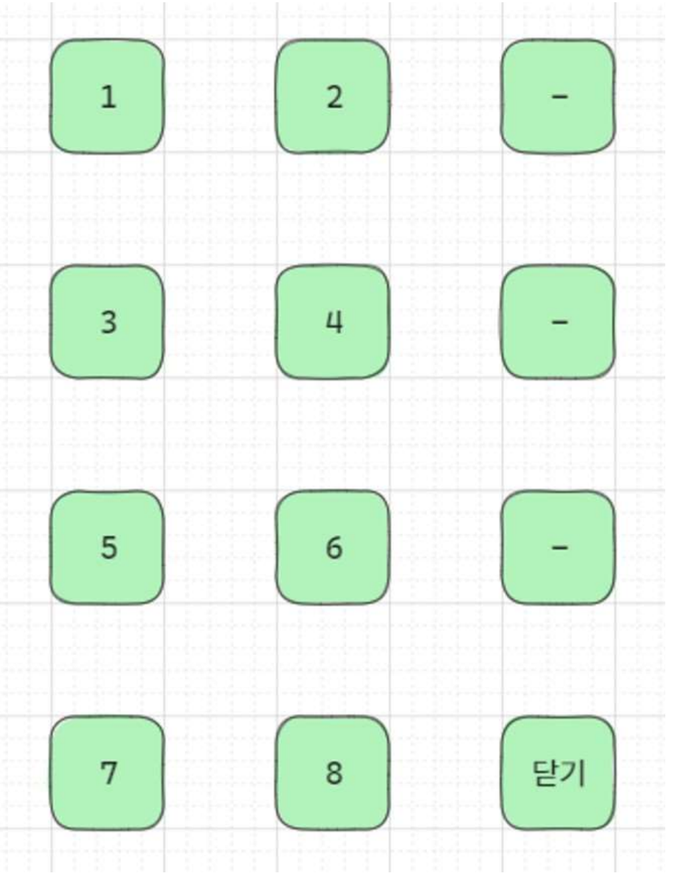
7조

2019250011 김정연 2019250010 김정목 2019250035 이동현 2020810059 이태주

기존내용 및 변경내용

(화물용)엘리베이터

- 사용할 장치 : Dot matrix, Tact switch, Character LCD
- 조건
  - 엘리베이터는 한 층당 여러 명 이용 가능
  - 총 8층
  - 우선순위목록 필요
- 내부 입력(tact switch 좌상단부터)
  - 입력 층수 === 현재 층수
  - 입력 층수 > 현재 층수
  - 입력 층수 < 현재 층수
- **외부 입력(레버 변화 감지)**
  - 엘리베이터 버튼 -> 호출 레버(화물용 엘리베이터)
- 이동(1초에 1층씩)
  - 이동 중 외부 입력 검사 -> [목적지 ~ 외부 입력을 받은 층수 ~ 현재 층수] 판단
  - 외부 입력에 따른 목적지 배열에 추가됨



Flow

- 이동(1초에 1층씩)
  - 이동 중 외부 입력 검사 -> [목적지 ~ 외부 입력을 받은 층수 ~ 현재 층수] 판단
  - 외부 입력에 따른 목적지 배열에 추가됨

참고한 Flow

엘레베이터의 상태

멈춰 있음
움직이는 중
버튼들의 상태

안에서 누름
각 층에서 누름(위/아래)

[코드1]
눌려있는 버튼이 없는 상태에서는 엘레베이터는 새로 눌리는 첫 버튼의 층으로 최종 도착지가 결정되고 그에 따라 상승 또는 하강 상태로 바뀐다.

[코드2]
엘레베이터가 이미 있는 층의 버튼은 눌리지 않는다.

[코드3]
엘레베이터가 이동 중일 때 이동방향의 최종지점에 도달하기 전까지는 이동방향을 바꾸지 않는다.

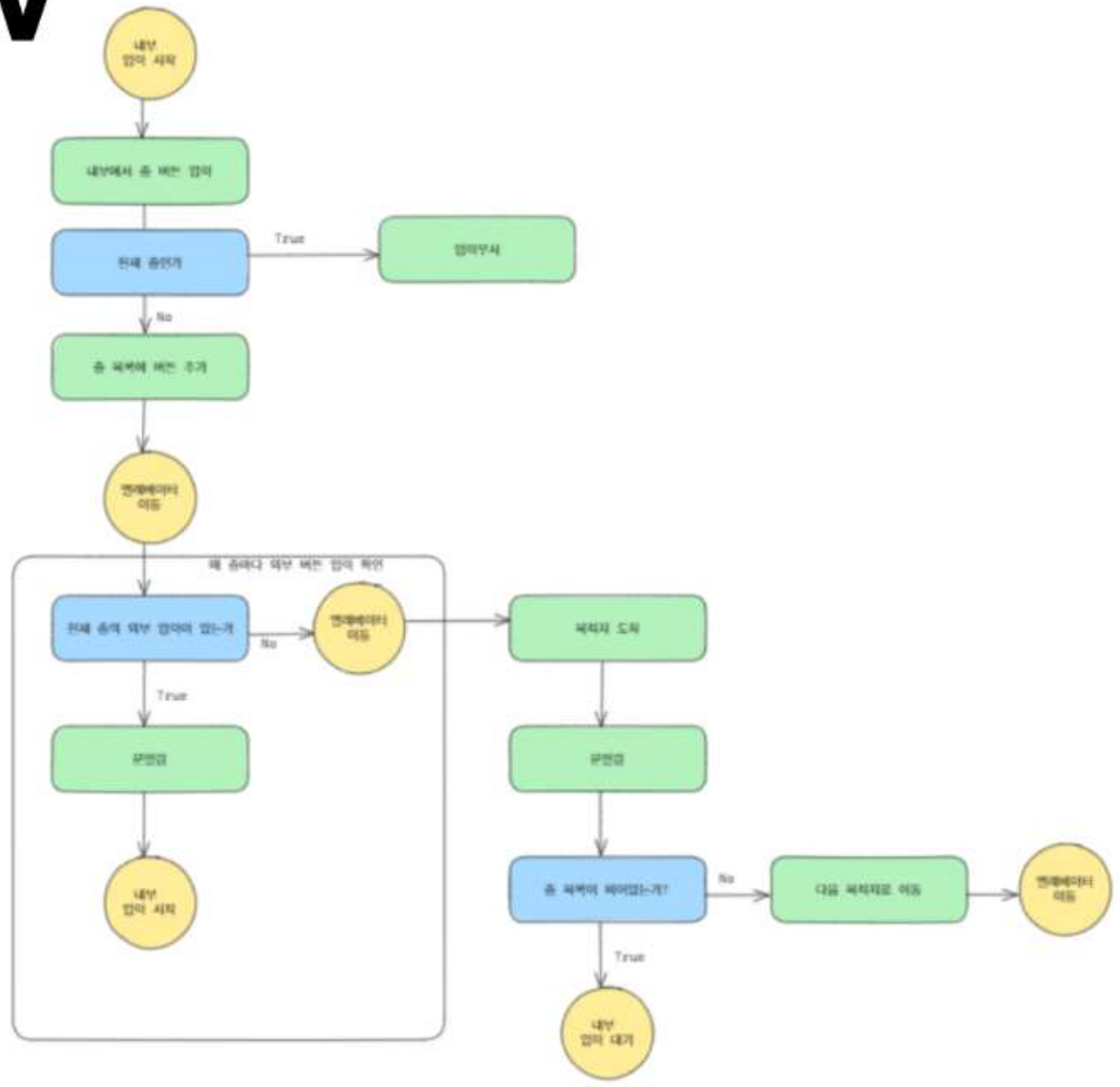
(엘레베이터가 특정 방향의 최종 목적지에 도달하면 다른 눌려있는 버튼이 하나 이상 있을 시 이동 방향을 바꾸고, 그 방향의 가장 끝점으로 최종 목적지를 변경한다.)

[코드4]
엘레베이터가 이동 중 버튼이 눌린 층이 있을 때 엘레베이터의 이동 방향이 다를 때는 정지하지 않는다.(방향이 같거나 중립이라면 정지한다.)

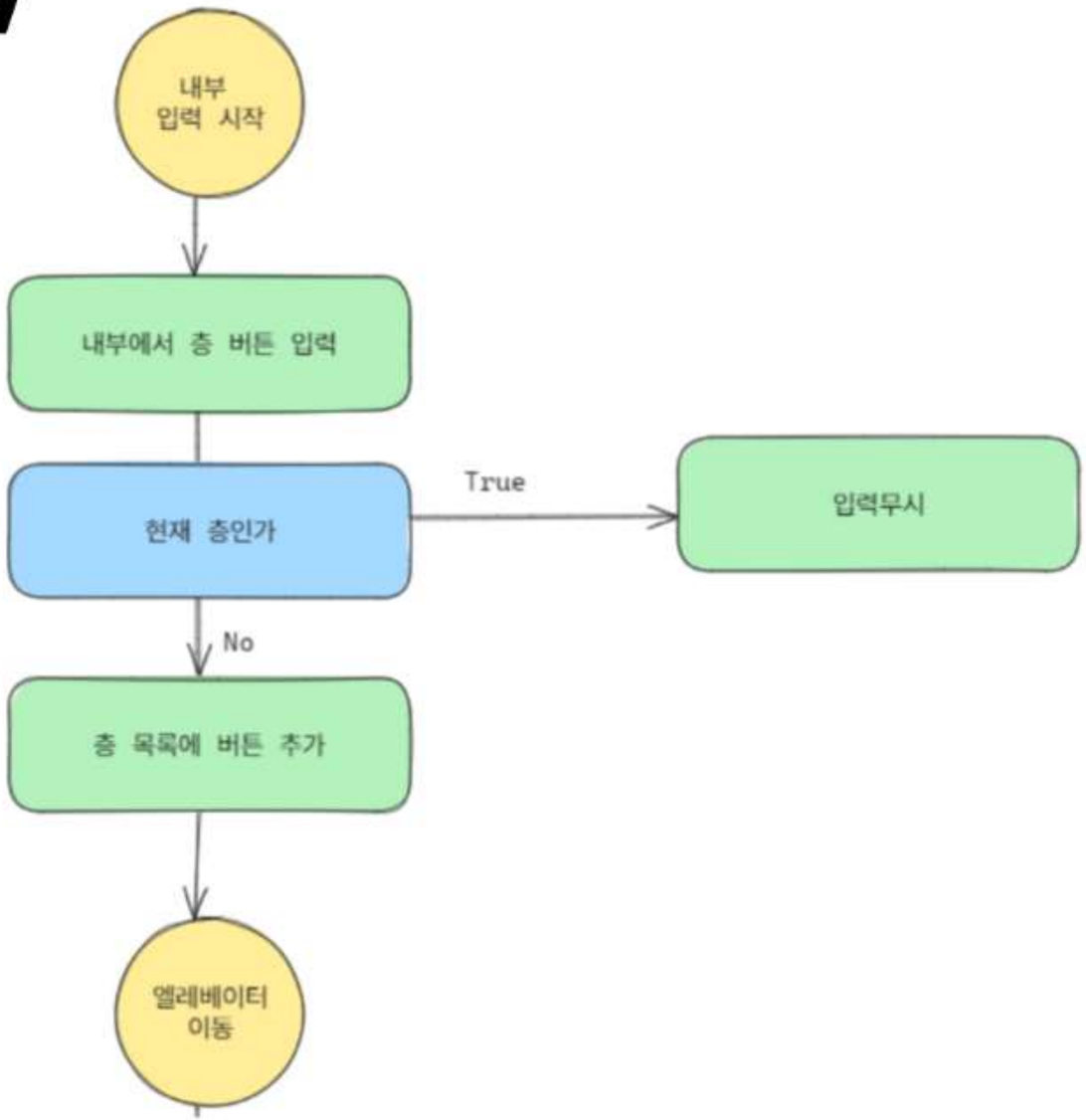
[코드5]
엘레베이터가 이동 중일 때 이동방향의 최종 목적지보다 멀리 있는 층의 버튼이 눌리면 최종 목적지를 그리로 수정한다.

- https://velog.io/@dongduu/%EC%8A%A4%ED%83%80%ED%8A%B8%EC%97%85%ED%8A%B8%EB%9E%99-%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%94%EB%93%9C-%EA%B0%9C%EB%B0%9C%EA%B3%BC%EC%A0%95-1-4

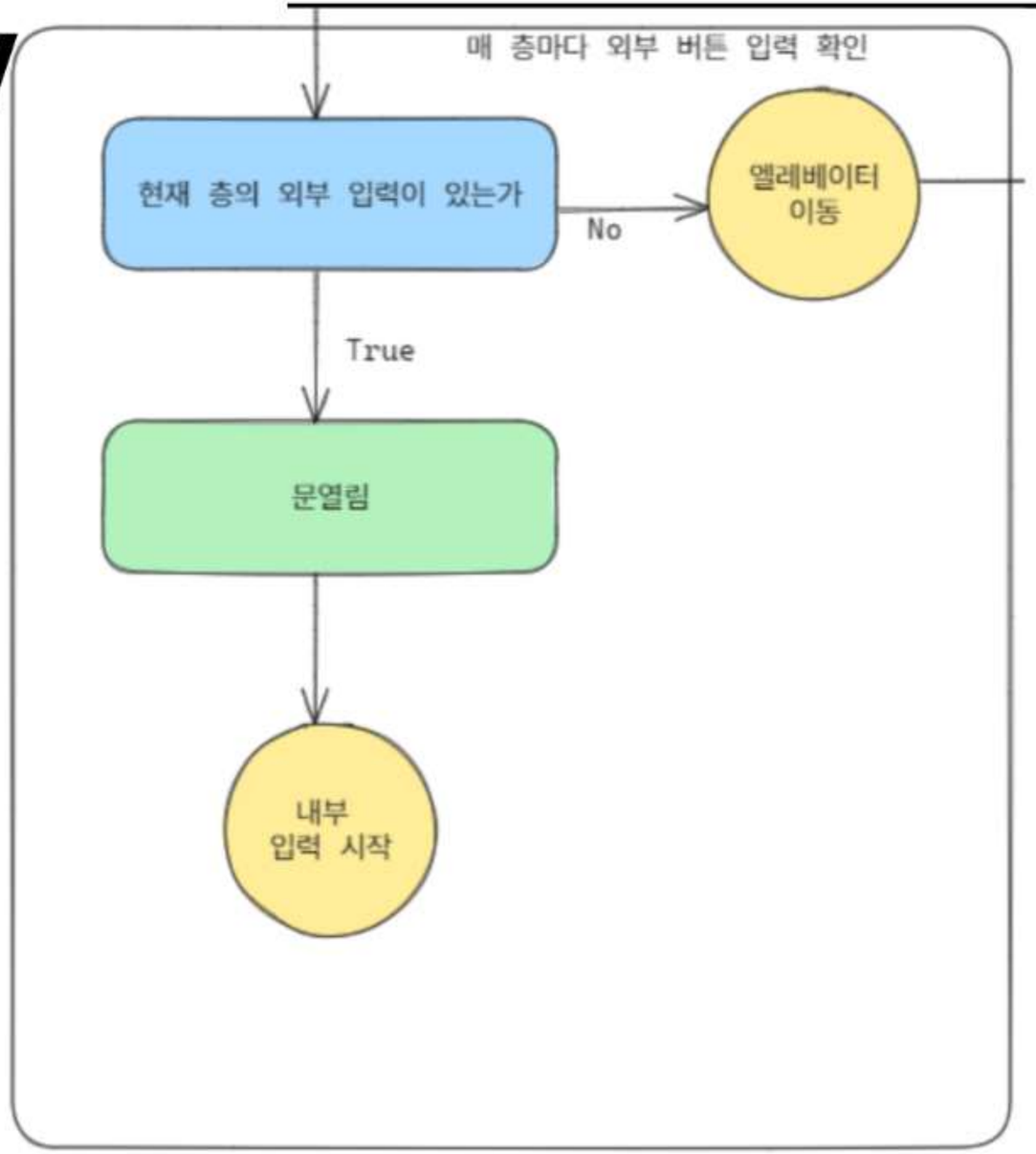
변경된 Flow



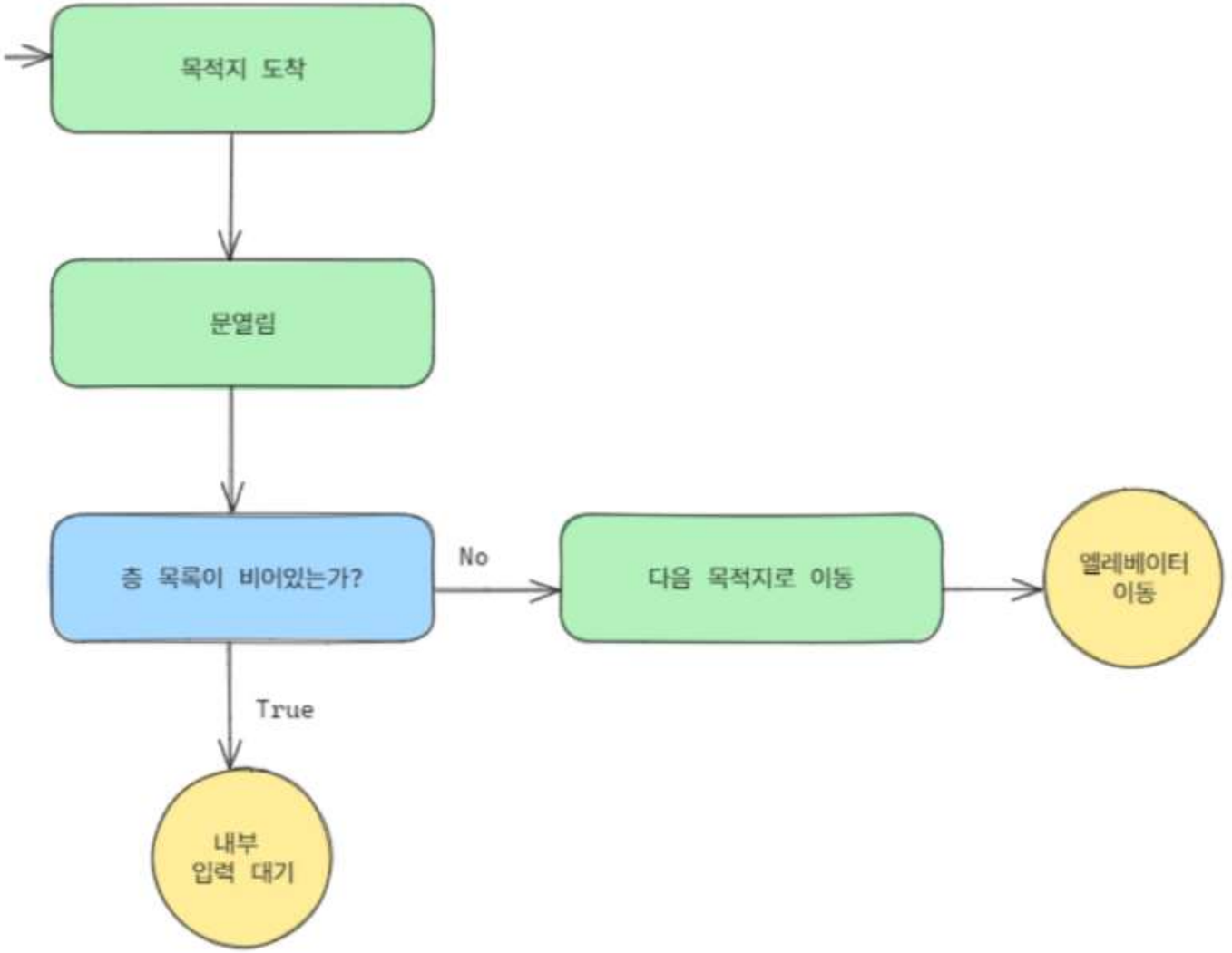
변경된 Flow



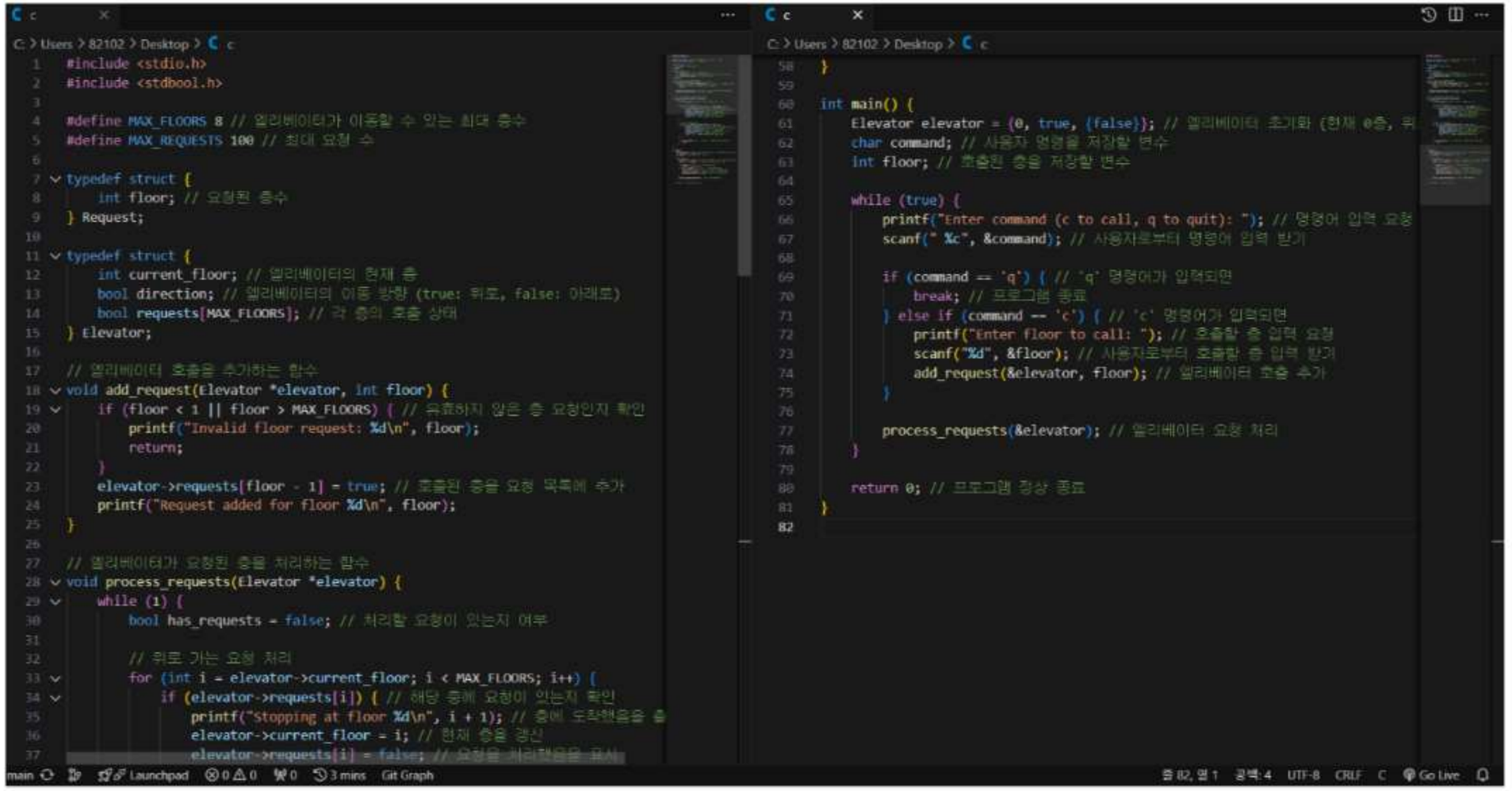
변경된 Flow



변경된 Flow



임시 code



```c
#include <stdio.h>
#include <stdbool.h>

#define MAX_FLOORS 8 // 엘리베이터가 이동할 수 있는 최대 층수
#define MAX_REQUESTS 100 // 최대 요청 수

typedef struct {
    int floor; // 요청된 층수
} Request;

typedef struct {
    int current_floor; // 엘리베이터의 현재 층
    bool direction; // 엘리베이터의 이동 방향 (true: 위로, false: 아래로)
    bool requests[MAX_FLOORS]; // 각 층의 호출 상태
} Elevator;

// 엘리베이터 호출을 추가하는 함수
void add_request(Elevator *elevator, int floor) {
    if (floor < 1 || floor > MAX_FLOORS) { // 유효하지 않은 층 요청인지 확인
        printf("Invalid floor request: %d\n", floor);
        return;
    }
    elevator->requests[floor - 1] = true; // 호출된 층을 요청 목록에 추가
    printf("Request added for floor %d\n", floor);
}

// 엘리베이터가 요청된 층을 처리하는 함수
void process_requests(Elevator *elevator) {
    while (1) {
        bool has_requests = false; // 처리할 요청이 있는지 여부

        // 위로 가는 요청 처리
        for (int i = elevator->current_floor; i < MAX_FLOORS; i++) {
            if (elevator->requests[i]) { // 해당 층에 요청이 있는지 확인
                printf("Stopping at floor %d\n", i + 1); // 층에 도착했음을 출
                elevator->current_floor = i; // 현재 층을 갱신
                elevator->requests[i] = false; // 요청을 처리했음을 표시
```

```c
}

int main() {
    Elevator elevator = {0, true, {false}}; // 엘리베이터 초기화 (현재 0층, 위
    char command; // 사용자 명령을 저장할 변수
    int floor; // 호출된 층을 저장할 변수

    while (true) {
        printf("Enter command (c to call, q to quit): "); // 명령어 입력 요청
        scanf(" %c", &command); // 사용자로부터 명령어 입력 받기

        if (command == 'q') { // 'q' 명령어가 입력되면
            break; // 프로그램 종료
        } else if (command == 'c') { // 'c' 명령어가 입력되면
            printf("Enter floor to call: "); // 호출할 층 입력 요청
            scanf("%d", &floor); // 사용자로부터 호출할 층 입력 받기
            add_request(&elevator, floor); // 엘리베이터 호출 추가
        }

        process_requests(&elevator); // 엘리베이터 요청 처리
    }

    return 0; // 프로그램 정상 종료
}
```

2주차 진행 상황

엘리베이터 알고리즘 임시 코드 → 코드 개선

Dot Matrix, Tact Switch, Character LCD 및 어댑터 코드, 명령어 작성

Code 개선

```cpp
#include <iostream>
#include <set>
#include <vector>
#include <thread>
#include <chrono>

enum ElevatorState {
    // 엘리베이터 상태(멈춤, 움직임-위/아래, 문-열림/닫힘)
    STOPPED,
    MOVING_UP,
    MOVING_DOWN,
    DOOR_OPEN,
    DOOR_CLOSED
};

class Elevator {
public:
    Elevator(int totalFloors) : currentFloor(1), state(STOPPED), totalFloors(totalFloors) {}

    // 엘리베이터 외부 호출
    void callElevator(int floor) {

        // 1~8층 사이의 층에서 호출했을 때
        if (floor >= 1 && floor <= totalFloors) {

            // 엘리베이터를 호출한 층과 엘리베이터의 현재 층이 일치할 때
            if (floor == currentFloor) {

                // 문열림
                openDoor();

                // 움직일 층을 고르라는 문구가 뜸
                std::cout << "Please select the floor to move: ";

                // 입력할 층을 입력받음
                int selectedFloor;
                std::cin >> selectedFloor;

                // 선택한 층이 1~8층 사이에 있고, 엘리베이터의 현재 층과 다를 때
                if (selectedFloor >= 1 && selectedFloor <= totalFloors && selectedFloor != currentFloor) {

                    // 내부 층 입력 함수로 입력 층을 넘겨줌
                    selectFloor(selectedFloor);
                }

            // 엘리베이터를 호출한 층과 엘리베이터의 현재 층이 일치하지 않을 때
            } else {

                // 엘리베이터가 이동할 층에 추가함
                targetFloors.insert(floor);

                // 엘리베이터의 상태를 업데이트함
                updateState();
            }
        }
    }
```

```cpp
private:
    int currentFloor; // 현재 층
    ElevatorState state; // 엘리베이터의 현재 상태
    int totalFloors; // 엘리베이터의 총 층수
    std::set<int> targetFloors; // 목적지 층들의 집합

    void openDoor() {
        state = DOOR_OPEN;
        displayStatus("Door Open");
        std::this_thread::sleep_for(std::chrono::seconds(2)); // 문 열림 상태 유지
        closeDoor();
    }

    void closeDoor() {
        state = DOOR_CLOSED;
        displayStatus("Door Closed");
        std::this_thread::sleep_for(std::chrono::seconds(2)); // 문 닫힘 상태 유지
    }

    void displayStatus(const char* status) {
        std::cout << "Status: " << status << std::endl;
    }
};

int main() {
    // 총 8층의 엘리베이터 생성
    Elevator elevator(8);

    // 외부에서 엘리베이터 호출
    elevator.callElevator(3);
    elevator.callElevator(5);

    // 내부에서 층수 선택
    elevator.selectFloor(7);
    elevator.selectFloor(2);

    // 선택한 층수 취소
    elevator.selectFloor(7);

    // 엘리베이터 동작 시작
    elevator.run();

    return 0;
}
```

Code 작성

```c
//#include <asm/ioctls.h>
#include <stdio.h>              // 입출력 관련
#include <stdlib.h>             // 문자열 변환, 메모리 관련
#include <string.h>             // 문자열 처리
#include <time.h>               // 시간 관련
#include <stdbool.h>

#include <fcntl.h>      // 타겟시스템 입출력 장치 관련
#include<sys/types.h>   // 시스템에서 사용하는 자료형 정보
//#include<sys/ioctl.h>     // 하드웨어의 제어와 상태 정보 //실제 사용시에는 주석해제
#include<sys/stat.h>        // 파일의 상태에 대한 정보
//#include <unistd.h>   // POSIX 운영체제 API에 대한 액세스 제공 //실제 사용시에는 주석해제

// Target System
#define dot "/dev/dot"      // Dot Matrix
#define tact "/dev/tactsw"  // Tact Switch
#define clcd "/dev/clcd"    // Character LCD
#define MESSAGE_NUM 18

int dot_d;
int tact_d;
int clcd_d;

int PRINT(char P[]);
void setup();

// 8종까지의 숫자 패턴 정의
// 하나의 8바이트는 하나의 도트행임
const unsigned char numbers[8][8] = {
    {0b00011000, 0b00111000, 0b00011000, 0b00011000, 0b00011000, 0b00011000, 0b01111110, 0b00000000}, // 1
    {0b00111100, 0b01100110, 0b00000110, 0b00001100, 0b00110000, 0b01100000, 0b01111110, 0b00000000}, // 2
    {0b00111100, 0b01100110, 0b00000110, 0b00011100, 0b00000110, 0b01100110, 0b00111100, 0b00000000}, // 3
    {0b00001100, 0b00011100, 0b00101100, 0b01001100, 0b01111110, 0b00001100, 0b00001100, 0b00000000}, // 4
    {0b01111110, 0b01100000, 0b01111100, 0b00000110, 0b00000110, 0b01100110, 0b00111100, 0b00000000}, // 5
    {0b00111100, 0b01100110, 0b01100000, 0b01111100, 0b01100110, 0b01100110, 0b00111100, 0b00000000}, // 6
    {0b01111110, 0b01100110, 0b00000110, 0b00001100, 0b00011000, 0b00011000, 0b00011000, 0b00000000}, // 7
    {0b00111100, 0b01100110, 0b01100110, 0b00111100, 0b01100110, 0b01100110, 0b00111100, 0b00000000}, // 8
};

char* clcd_top[MESSAGE_NUM] =
{ "press any button" };
```

```c
void setup() {
    dot_d = open(dot, O_RDWR);
    write(dot_d, &numbers[0], sizeof(numbers[0]));//sizeof 안되면 numbers[0]->numbers 시도
    PRINT(clcd_top);
    sleep(50000); //이부분은 테스트용일 이후 지울것
    close(dot_d); //숫자가 사라져버리면 지울 것
}
int PRINT(char P[]) {
    clcd_d = open(clcd, O_RDWR);
    if (clcd < 0) {
        perror("clcd 오류");
        exit(0);
    }
    write(clcd_d, P, strlen(P));
    sleep(3);
    close(clcd_d);
}
//i는 반복문용
int main() {
    int i; //반복문용
    setuo();
    dot_d = open(dot, O_RDWR);
    tact_d = open(tact, O_RDWR);
    if (dot_d < 0) {
        perror("dot 오류");
        exit(0);
    }
    if (tact_d < 0) {
        perror("tact 오류");
        exit(0);
    }
    close(tact_d);
    close(dot_d);
}
```