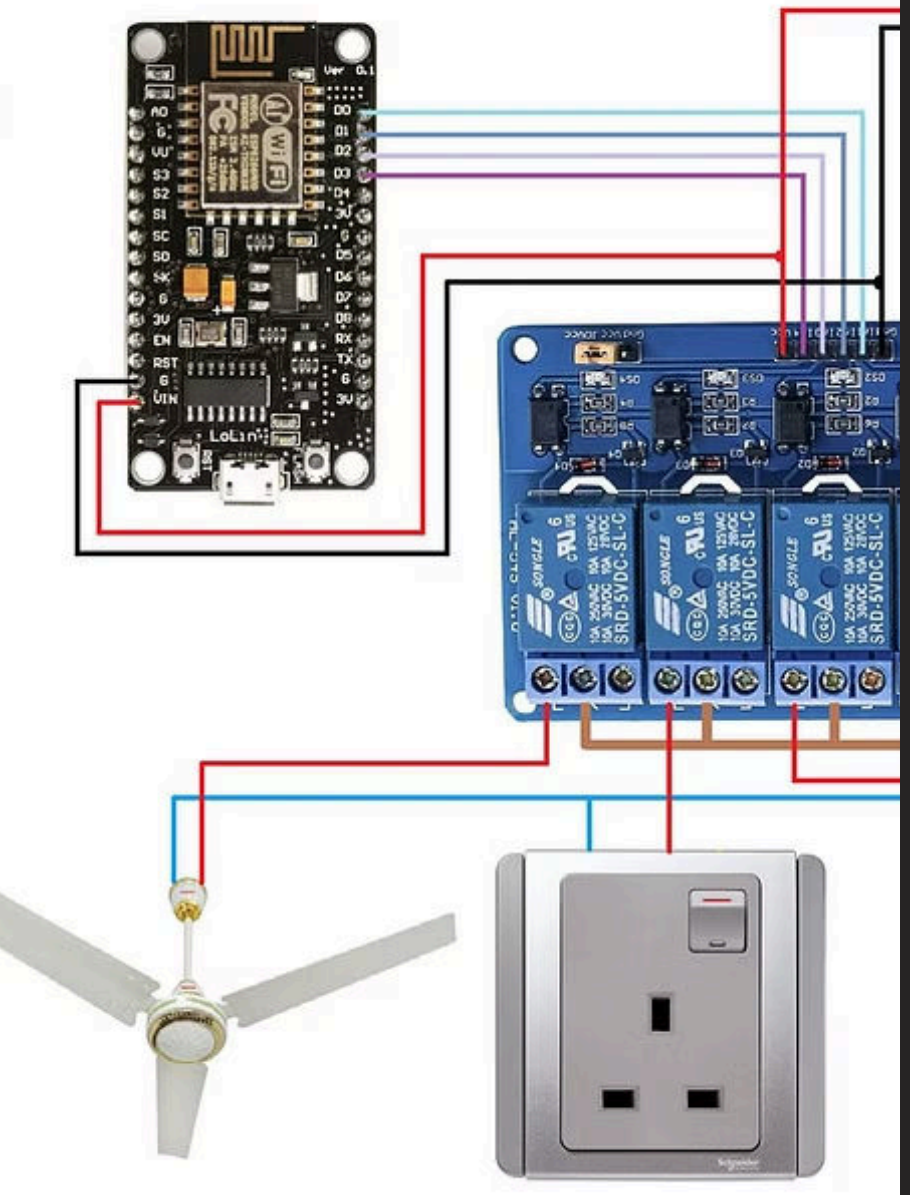


프로젝트
중간 점검

김영인 심예영 윤혜진 최현서

프로젝트 개요

1 **변경 내용**

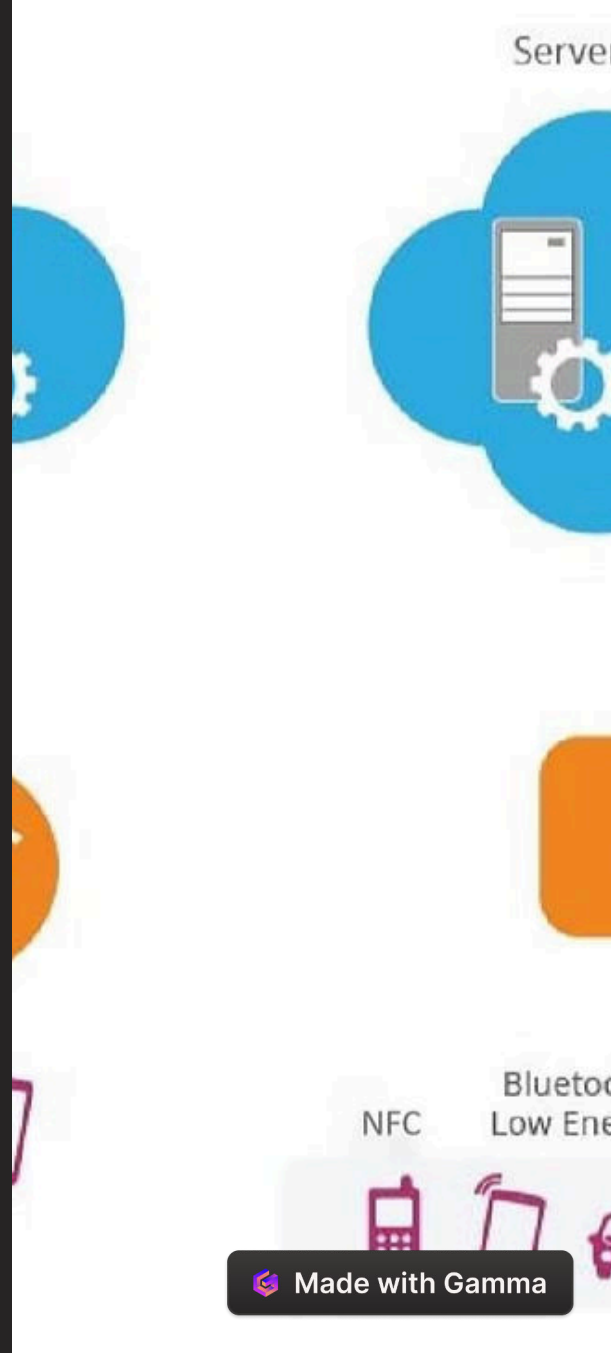
잠금 시스템을 NFC 태그에서 Tact Switch로

2 **감지 시스템**

초음파 센서를 이용해 외부 물체 감지

3 **보안 시스템**

LED 센서 코드 및 각 센서 연결 코드



변경 내용 - 잠금 시스템

기존 방식

NFC 태그를 이용한 잠금 해제 방식을 사용

새로운 방식

Tact Switch를 이용한 패스워드 입력 방식으로 변경

변경 이유

자료 부족으로 인해 방식을 변경

Tact Switch

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdbool.h>
#include <time.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/stat.h>

#define TACT "/dev/tactsw"  // 택트 스위치 장치 파일 경로

typedef struct {
    char storedPassword[5];  // 저장된 비밀번호 (최대 4자리 + null 종단 문자)
    char inputPassword[5];  // 입력된 비밀번호
    int attemptCount;  // 비밀번호 시도 횟수
    int mode;  // 모드 (0: 출근, 1: 퇴근)
    int count;  // 입력 횟수
    bool error;  // 에러 플래그
} SecurityKey;

// 시스템 초기 설정
void setup(SecurityKey *key) {
    // 초기 상태 설정
    key->attemptCount = 0;
    key->mode = 0;  // 기본 모드: 출근 모드
    key->count = 0;
    key->error = false;
    memset(key->inputPassword, 0, sizeof(key->inputPassword));
}

// 비밀번호 확인
void checkPassword(SecurityKey *key) {
    // 입력된 비밀번호가 저장된 비밀번호와 일치하면 메시지 출력
    if (strcmp(key->inputPassword, key->storedPassword) == 0) {
        printf("Password is correct!\n");
    } else {
        // 일치하지 않으면 시도 횟수 증가
        key->attemptCount++;
        printf("Incorrect password. Attempt count: %d\n", key->attemptCount);
    }
    // 입력된 비밀번호 초기화
    memset(key->inputPassword, 0, sizeof(key->inputPassword));
}

// 키패드 입력 받기
char getKeypadInput(SecurityKey *key) {
    unsigned char b;
    int tactswFd = open(TACT, O_RDONLY);
    if (tactswFd < 0) {
        perror("tact device error");
        return '\0';
    }
    read(tactswFd, &b, sizeof(b));
    close(tactswFd);
    // 택트 스위치 입력에 따라 반환 값 결정
    switch (b) {
    case 1: return '1';
    case 2: return '2';
    case 4: return '3';
    case 8: return '4';
    case 16: return '5';
    case 32: return '6';
    case 64: return '7';
    case 128: return '8';
    case 256: return '9';
    case 512: return '0';
    case 1024: return '*';
    case 2048: return '#';
    default: return '\0';
    }
}
```

```c
// 출근/퇴근 모드 처리 및 비밀번호 확인
void handleKeyPress(SecurityKey *key) {
    char keyInput = getKeypadInput(key);
    if (keyInput) {
        // '*' 키를 누르면 출근 모드로 변경
        if (keyInput == '*') {
            key->mode = 0;  // 출근 모드로 변경
            printf("Mode changed to Work Mode.\n");
        }
        // '#' 키를 누르면 퇴근 모드로 변경
        else if (keyInput == '#') {
            key->mode = 1;  // 퇴근 모드로 변경
            printf("Mode changed to Home Mode.\n");
        }
        // 그 외의 키 입력을 비밀번호로 추가
        else {
            size_t len = strlen(key->inputPassword);
            if (len < sizeof(key->inputPassword) - 1) {
                key->inputPassword[len] = keyInput;
                key->inputPassword[len + 1] = '\0';
                key->count++;
                if (key->count >= 5) {
                    // 입력 횟수 초과 시 에러 설정 및 초기화
                    key->error = true;
                    key->count = 0;
                    printf("Error: Too many inputs. Error flag set.\n");
                }
                // 5번 입력 시 비밀번호 확인
                if (key->count == 5) {
                    checkPassword(key);
                }
            }
        }
    }
}

int main() {
    // SecurityKey 구조체 초기화
    SecurityKey key = { "1234", "", 0, 0, 0, false };
    setup(&key); // 시스템 초기 설정

    while (1) {
        handleKeyPress(&key); // 키 입력 처리
        if (key.error) {
            // 에러가 발생하면 처리
            printf("Error occurred. Resetting...\n");
            key.error = false; // 에러 플래그 초기화
            sleep(1); // 간단한 지연
        }
    }
    return 0;
}
```

감지 시스템 - 초음파 센서

물체와의 거리를 감지하는 센서

초음파를 발생시켜 물체에 닿아 반사된 신호를 감지하여 거리를 측정

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/types.h>
#include <asm/io.h>
#include <linux/interrupt.h>
#include <linux/irq.h>
#include <linux/delay.h>
#include <linux/time.h>

// GPIO 관련 상수 정의
#define GPIO_BASE   0xE0200000 // GPIO 베이스 주소
#define GPH0CON     (0xC00 / 4) // GPIO 핀 컨트롤 레지스터 오프셋
#define GPH0DAT     (0xC04 / 4) // GPIO 데이터 레지스터 오프셋
#define TRIG        14 // 트리거 핀 번호
#define ECHO        16 // 에코 핀 번호
#define THRESHOLD   50 // 거리 임계값 (50cm)
#define IRQ_NUM     10 // 인터럽트 번호

// 전역 변수 선언
static volatile u32 *gpio_base = 0x0; // GPIO 베이스 주소의 포인터
static struct timespec64 before; // 트리거 시점의 시간 저장
static int ultrasonic_signal = 0; // 초음파 센서 신호 저장 변수

// GPIO 데이터 레지스터 설정 함수
static void gpio_data(int gpio, int bit, int value) {
    u32 data = 0x0;
    data = gpio_base[gpio]; // 현재 GPIO 데이터 읽기
    data &= ~(1 << bit); // 해당 비트 클리어
    data |= (value << bit); // 새로운 값 설정
    gpio_base[gpio] = data; // 데이터 레지스터에 값 쓰기
}

// GPIO 컨트롤 레지스터 설정 함수
static void gpio_config(int gpio, int bit, int value) {
    u32 data = 0x0;
    data = gpio_base[gpio]; // 현재 GPIO 컨트롤 레지스터 읽기
    data &= (~(0xF << (bit * 4))); // 해당 비트 클리어
    data |= (value << (bit * 4)); // 새로운 값 설정
    gpio_base[gpio] = data; // 컨트롤 레지스터에 값 쓰기
}

// 인터럽트 핸들러 함수
static irqreturn_t int_interrupt(int irq, void *dev_id) {
    struct timespec64 after;
    ktime_get_real_ts64(&after); // 현재 시간 저장

    // 시간 차이를 마이크로초 단위로 계산
    long duration = (after.tv_sec - before.tv_sec) * 1000000 + (after.tv_nsec - before.tv_nsec) / 1000;
    long distance = duration / 58; // 거리를 cm 단위로 계산

    printk("%ldcm\n", distance); // 거리 출력
    if (distance <= THRESHOLD) { // 거리가 임계값 이하일 경우
        printk("Object detected within 50cm\n");
        ultrasonic_signal = 1; // 신호 변수 설정
    } else { // 거리가 임계값 이상일 경우
        printk("No object detected within 50cm\n");
        ultrasonic_signal = 0; // 신호 변수 클리어
    }
    return IRQ_HANDLED; // 인터럽트 처리 완료
}

// 인터럽트 등록 함수
static int register_itrp(void) {
    if (request_irq(IRQ_NUM, int_interrupt, IRQF_TRIGGER_FALLING, "ultrasonic_sensor", NULL)) {
        return -EINVAL; // 인터럽트 등록 실패 시 에러 반환
    }
    return 0; // 인터럽트 등록 성공
}

// GPIO 초기화 함수
static void gpio_init(void) {
    gpio_config(GPH0CON, TRIG, 1); // 트리거 핀을 출력으로 설정
    gpio_config(GPH0CON, ECHO, 0xF); // 에코 핀을 입력으로 설정
}

// 초음파 펄스 송신 함수
static void send_pulse(void) {
    gpio_data(GPH0DAT, TRIG, 1); // 트리거 핀 HIGH 설정
    udelay(10); // 10 마이크로초 대기
    gpio_data(GPH0DAT, TRIG, 0); // 트리거 핀 LOW 설정
    ktime_get_real_ts64(&before); // 현재 시간 저장
}

// 모듈 초기화 함수
static int __init ultrasonic_init(void) {
    printk(KERN_INFO "Ultrasonic sensor module loaded\n"); // 모듈 로드 메시지 출력
    gpio_base = (u32 *)ioremap(GPIO_BASE, PAGE_SIZE); // GPIO 메모리 매핑
    if (!gpio_base) { // 매핑 실패 시
        printk(KERN_ERR "Failed to remap IO memory\n");
        return -ENOMEM; // 메모리 부족 에러 반환
    }
    gpio_init(); // GPIO 초기화
    if (register_itrp()) { // 인터럽트 등록
        printk(KERN_ERR "Failed to register interrupt\n");
        iounmap((void *)gpio_base); // 매핑 해제
        return -EINVAL; // 에러 반환
    }
    send_pulse(); // 초기 펄스 송신
    return 0; // 모듈 초기화 성공
}

// 모듈 종료 함수
static void __exit ultrasonic_exit(void) {
    if (gpio_base) { // GPIO 베이스 주소가 설정되어 있을 경우
        iounmap((void *)gpio_base); // 매핑 해제
    }
    free_irq(IRQ_NUM, NULL); // 인터럽트 해제
    printk(KERN_INFO "Ultrasonic sensor module unloaded\n"); // 모듈 언로드 메시지 출력
}

// 모듈 초기화 및 종료 함수 등록
module_init(ultrasonic_init);
module_exit(ultrasonic_exit);

// 모듈 메타데이터
MODULE_LICENSE("GPL");
MODULE_AUTHOR("IOT Team 6");
MODULE_DESCRIPTION("Ultrasonic Sensor Driver");
```

보안 시스템 - LED 센서 코드

LED 센서

초음파 센서의 신호를 받고 발광합니다.

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/ioctl.h>
#include<sys/stat.h>

#define led "/dev/led" // led 장치 불러오기
#define dip "/dev/dipsw"

int main() {
    int dev, dip_d;
    unsigned char c, data, ultrasonic_signal = 0; // 초음파 신호를 0으로 초기화

    // 장치 파일 열기
    dev = open(led, O_RDWR); // LED 장치 파일 열기
    dip_d = open(dip, O_RDWR); // DIP 스위치 장치 파일 열기

    if (dev < 0) {
        printf("Can't open LED.\n");
        exit(0); // 장치를 열지 못한 경우 예외 처리 후 종료
    }

    if (dip_d < 0) {
        printf("Can't open DIP switch.\n");
        close(dev);
        exit(0); // DIP 스위치 장치를 열지 못한 경우 예외 처리 후 종료
    }

    while (1) {
        data = 0xff;

        // 초음파 신호가 없는 상황을 시뮬레이션하기 위해 ultrasonic_signal을 0으로 설정
        ultrasonic_signal = 0;

        // 초음파 신호가 있으면 LED 켜기
        if (ultrasonic_signal) {
            data = 0x00; // 모든 LED 켜기

            // DIP 스위치 신호 읽기
            read(dip_d, &c, sizeof(c));

            // DIP 스위치가 ON인 경우 해당 LED 끄기
            if (c & 0x01) data |= 0x01;
            if (c & 0x02) data |= 0x02;
            if (c & 0x04) data |= 0x04;
            if (c & 0x08) data |= 0x08;
            if (c & 0x10) data |= 0x10;
            if (c & 0x20) data |= 0x20;
            if (c & 0x40) data |= 0x40;
            if (c & 0x80) data |= 0x80;

            // LED 장치에 데이터 쓰기
            write(dev, &data, sizeof(unsigned char));
        }

        usleep(200000); // 200밀리초 대기
    }

    close(dip_d);
    close(dev);
    return 0;
}
```

보안 시스템, 출퇴근 상태

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <nfc/nfc.h>

#define LED_DEVICE "/dev/led" // LED 장치 경로
#define DIP_DEVICE "/dev/dipsw" // DIP 스위치 장치 경로
#define ULTRASONIC_DEVICE "/dev/ultrasonic" // 초음파 센서 장치 경로 (예시)

// 장치 파일 열기
int open_device(const char* device_path) {
    int device = open(device_path, O_RDWR);
    if (device < 0) {
        printf("Can't open %s.\n", device_path);
        exit(0);
    }
    return device;
}

// LED 상태 설정
void set_led_state(int dev, unsigned char data) {
    write(dev, &data, sizeof(unsigned char));
}

// DIP 스위치 상태 읽기
unsigned char read_dip_switch(int dip_d) {
    unsigned char c;
    read(dip_d, &c, sizeof(c));
    return c;
}

// 초음파 신호 처리 및 LED 제어
void handle_ultrasonic_signal(int ultrasonic_d, int dip_d, int led_d) {
    unsigned char c, data = 0xff;
    unsigned char ultrasonic_signal = 0;

    // 초음파 신호를 읽어오는 로직 (예시)
    // read(ultrasonic_d, &ultrasonic_signal, sizeof(ultrasonic_signal));

    // 초음파 신호가 있는 경우
    if (ultrasonic_signal) {
        data = 0x00; // 모든 LED 켜기
        c = read_dip_switch(dip_d);

        // DIP 스위치가 ON인 경우 해당 LED 끄기
        if (c & 0x01) data |= 0x01;
        if (c & 0x02) data |= 0x02;
        if (c & 0x04) data |= 0x04;
        if (c & 0x08) data |= 0x08;
        if (c & 0x10) data |= 0x10;
        if (c & 0x20) data |= 0x20;
        if (c & 0x40) data |= 0x40;
        if (c & 0x80) data |= 0x80;

        set_led_state(led_d, data);
    }
}

// 출근 및 퇴근 처리
void handle_attendance(int dip_d) {
    unsigned char c = read_dip_switch(dip_d);

    if (c & 0x01) {
        printf("출근 처리\n");
    }
    if (c & 0x02) {
        printf("퇴근 처리\n");
    }
}
```

```c
// NFC 태그 감지 처리(임시) ->Tact Switch 로 수정
void handle_nfc(nfc_device* pnd) {
    nfc_target nt;
    const nfc_modulation nmModulations[] = {
        {.nmt = NMT_ISO14443A, .nbr = NBR_106 }
    };
    const size_t szModulations = 1;
    const uint8_t uiPollNr = 1;
    const uint8_t uiPeriod = 2;

    if (nfc_initiator_poll_target(pnd, nmModulations, szModulations, uiPollNr, uiPeriod, &nt) > 0) {
        // NFC 태그 감지 처리(임시) ->Tact Switch 로 수정
        printf("NFC tag detected!\n");
    }
    else {
        // NFC 태그 감지 처리(임시) ->Tact Switch 로 수정
        printf("No NFC tag detected.\n");
    }
}

int main() {
    int led_d, dip_d, ultrasonic_d;
    nfc_device* pnd;
    nfc_context* context;

    // 장치 파일 열기
    led_d = open_device(LED_DEVICE); // LED 장치 파일 열기
    dip_d = open_device(DIP_DEVICE); // DIP 스위치 장치 파일 열기
    ultrasonic_d = open_device(ULTRASONIC_DEVICE); // 초음파 센서 장치 파일 열기

    // libnfc 초기화
    nfc_init(&context);
    if (context == NULL) {
        fprintf(stderr, "Unable to init libnfc\n");
        return EXIT_FAILURE;
    }

    // NFC 디바이스 열기
    pnd = nfc_open(context, NULL);
    if (pnd == NULL) {
        fprintf(stderr, "Unable to open NFC device\n");
        nfc_exit(context);
        return EXIT_FAILURE;
    }

    // NFC 디바이스 설정
    if (nfc_initiator_init(pnd) < 0) {
        nfc_perror(pnd, "nfc_initiator_init");
        nfc_close(pnd);
        nfc_exit(context);
        return EXIT_FAILURE;
    }

    printf("NFC reader: %s opened\n", nfc_device_get_name(pnd));

    while (1) {
        handle_ultrasonic_signal(ultrasonic_d, dip_d, led_d);
        handle_attendance(dip_d);
        handle_nfc(pnd);

        usleep(200000); // 200밀리초 대기
    }

    // 리소스 해제
    close(dip_d);
    close(led_d);
    close(ultrasonic_d);
    nfc_close(pnd);
    nfc_exit(context);

    return 0;
}
```