

시스템 프로그래밍 및 실습 F124 6조

Final project Report

202322051 곽민서

202321998 김유진

202322039 노승현

202322048 최유정

CONTENTS

01. 팀 소개

02. 프로젝트 목표 및 개요

03. 시스템 구조도

04. 세부 구현 사항

05. 도전적 이슈

06. 계획 대비 완성도

07. 고찰

01. 팀 소개

팀명 : LONG TIME 노씨

팀원



곽민서 소프트웨어학과 202322051



노승현 소프트웨어학과 202322039



김유진 소프트웨어학과 2023221998



최유정 소프트웨어학과 202322048

02. 프로젝트 목표 및 개요

프로젝트 목표

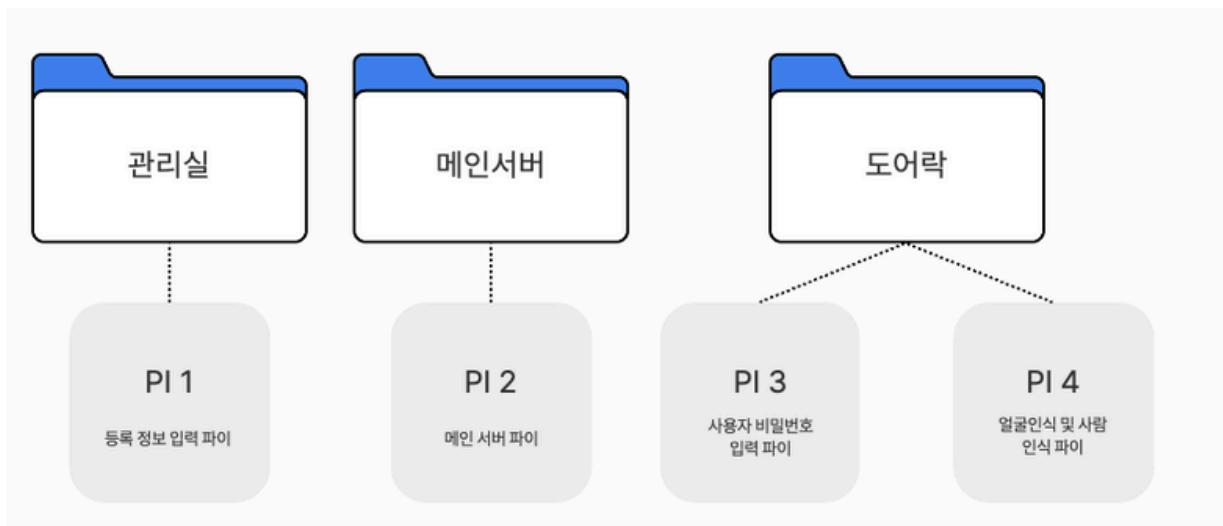
기숙사생들 출입 시의 불편함 감소
얼굴 인식과 비밀번호 확인 두 차례에 걸친 이중 보안
기숙사생과 외부인 구분

프로젝트 개요

본 프로젝트는 스마트 기숙사 도어락 통합 시스템을 구현하는 것이다.
시스템 구성은 관리실, 도어락으로 구성되어 있다.

먼저, 관리실에서는 사용자가 자신의 정보를 등록하고, 해당 정보를 데이터베이스에 저장한 후 등록된 사진을 이용해서 얼굴 인식 모델을 학습한다. 또한, 외부인 감지 시 관리실에서 알림이 울린다.

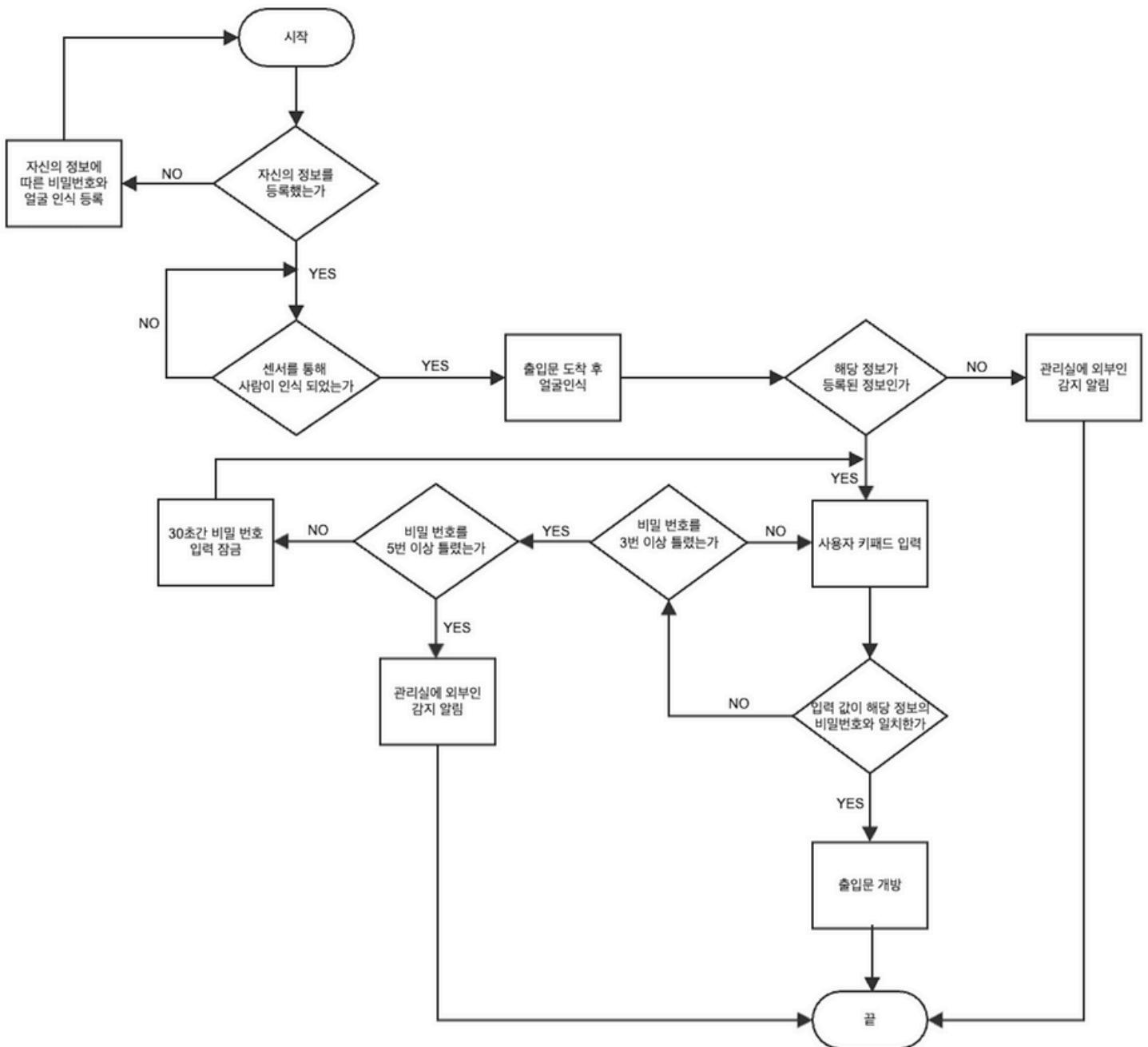
도어락 부분에서의 개요는 다음과 같다. 모션 감지되면 카메라와 학습된 얼굴 인식 모델을 사용하여 얼굴인식을 한다. 얼굴인식이 되면 데이터베이스에서 해당 학번에 따른 비밀번호와 입력된 키패드의 번호를 확인하여 일치하면 모터를 이용해 문을 개방한다. 만약 5회 틀리면 관리실에 외부인 침입 알림을 보낸다.



03. 시스템 구조도

FLOW CHART

전체 시스템 구조도는 다음과 같다.



03. 시스템 구조도

PI 1

PI 1은 키패드, 카메라 센서를 이용해 사용자로부터 입력을 받고, 받은 입력 값을 다른 PI들에게 전달한다. 버튼 입력을 이용해 눌림이 감지되면 사용자 입력이 시작되며 키패드 센서를 이용하여 정보 입력을 시행한다. 정보 입력 과정은 LCD 센서를 통해 확인할 수 있으며 카메라 모듈을 이용해 사용자를 촬영한 후 정보를 PI 2에게 전송한다.

이를 수행하기 위해 버튼 센서, LCD 센서, 키패드 센서, 카메라 모듈을 사용한다.

PI 2

PI 2는 사생 정보를 저장하는 데이터베이스 서버 역할과 각 라즈베리 파이의 요청을 처리하는 메인 서버 역할을 수행한다. 이 과정에서 발생하는 알림과 데이터베이스 테이블 출력을 시각화하고 소리로 전달한다.

이를 수행하기 위해 버튼 센서, RGB LED 센서, 피에조 부저 센서를 사용한다.

PI 3

PI 3은 사용자로부터 키패드 입력 받으며 입력 상황이 LCD를 통해 보여지고, 입력 받은 비밀 번호와 PI 2로 부터 받은 비밀 번호가 일치하면 문을 열고, 일치하지 않으면 다시 키패드를 작동시키는 도어락 역할을 수행한다.

이를 수행하기 위해, 키패드 센서, 서보 모터, LCD 센서를 사용한다.

PI 4

PI 4는 PI 1로부터 사진을 전달받은 후 인공지능을 학습 시키고, 모션감지를 통해 사진을 찍어 사용자가 등록된 사람인 경우 학번을, 등록되지 않은 사람의 경우는 외부인이라는 신호를 PI 2로 보낸다.

이를 수행하기 위해, 모션 감지 센서(PIR)와 카메라 모듈을 사용한다.

03. 시스템 구조도

PI 1

PI 1의 프로그램은 다음과 같이 작동한다.

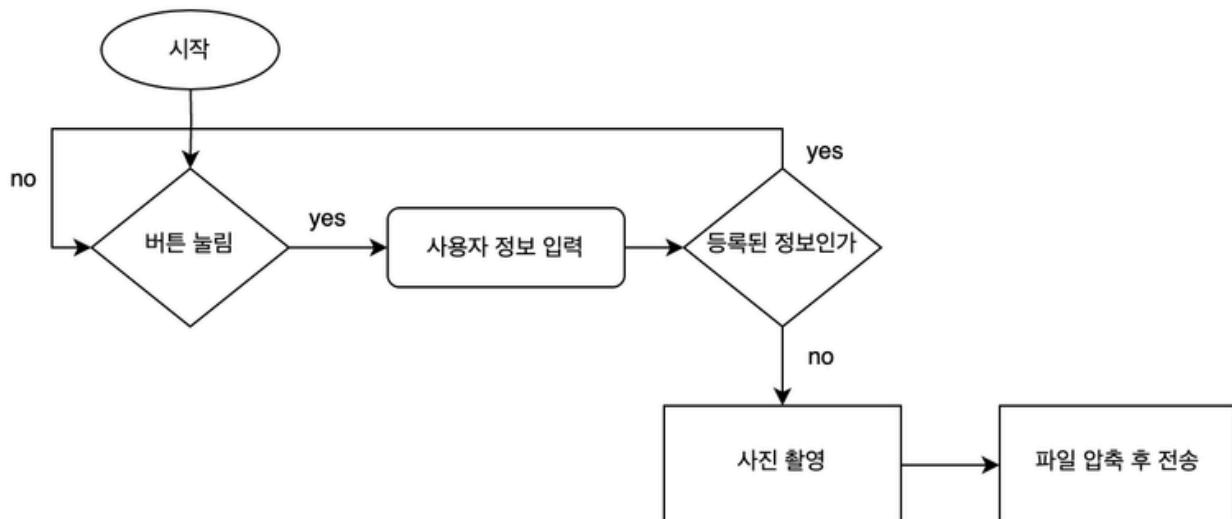
버튼 센서가 눌리면 사용자 입력이 시작되며, 버튼이 눌리기 전까지 LCD에 "No Input"이라는 문자열이 출력된다. 버튼이 눌림이 감지되면 "Enter Your Information"이 3초 동안 출력되면서 학번, 비밀번호, 방 호수를 순서대로 입력하게 된다. 이때 '*' 버튼을 누르면 백스페이스 기능이 작동하고, '#' 버튼을 누르면 입력 완료가 표시되도록 구현하였다.

또한, 학번 9자리, 비밀번호 및 방 호수 4자리를 모두 입력하지 않을 경우 "Enter Full ~"와 같은 문구가 LCD에 출력되며, 사용자가 다시 입력할 수 있도록 기능을 추가하였다.

세 가지 정보를 모두 입력받은 후, PI 2에게 중복된 학번, 즉 중복된 사용자가 존재하는지 확인하는 통신을 보내게 된다. 중복 사용자가 존재할 경우 더 이상 진행이 불가능하다는 'n'을, 존재하지 않을 경우 진행 가능하다는 'y'를 받게 된다. 'y'를 받으면 등록인의 사진 촬영이 시작되며, 50장의 사진이 촬영된다.

이미지 파일의 저장 형식은 '[학번].[사진번호]'로 하였고, '/home/mkms8/image' 디렉토리에 이미지 파일을 저장한 후 해당 디렉토리의 이미지 파일들을 압축하여 zip 파일으로 만들고 PI 4에게 소켓 통신을 통해 전송한다.

zip 파일 전송이 완료되면 프로그램은 다시 초기 상태로 돌아가 버튼 센서가 눌릴 때 까지 대기한다.



03. 시스템 구조도

PI 2 - SERVER

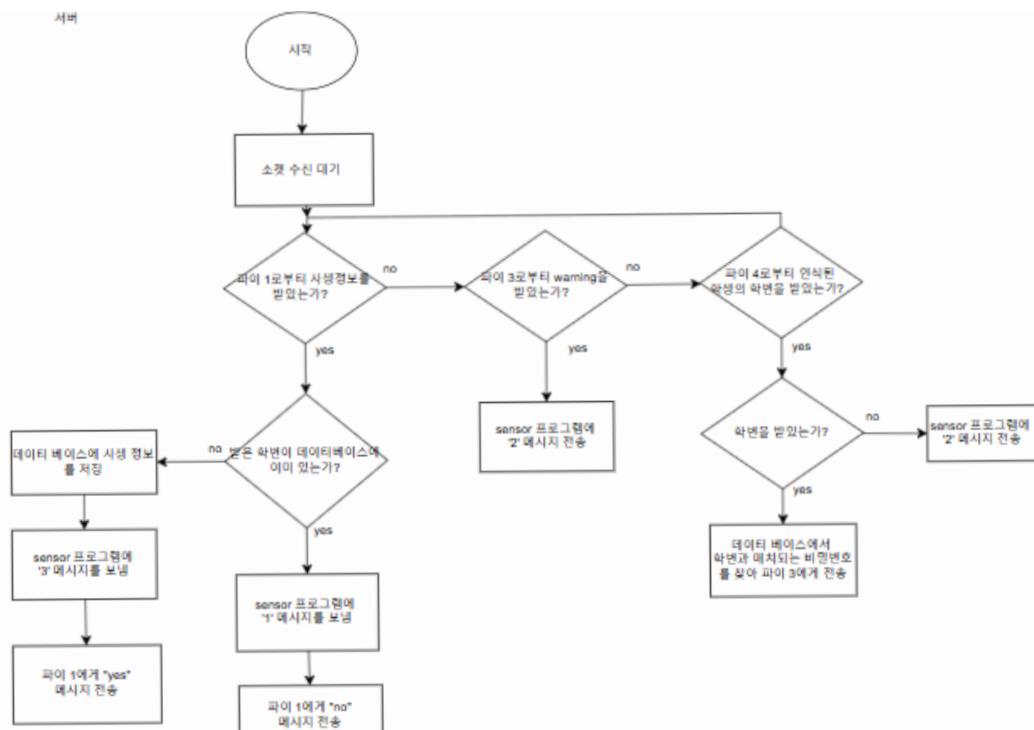
PI 2의 서버는 프로그램 시작과 동시에 클라이언트 장치(PI)로부터 데이터를 수신하기 위해 대기 상태에 들어간다. 클라이언트로부터 요청이 수신되면, 요청의 유형에 따라 다음과 같은 작업이 수행된다.

먼저, PI 1로부터 학생 정보 등록 요청이 들어온 경우, 서버는 해당 학번이 데이터베이스에 존재하는지 확인한다. 학번이 존재하지 않으면, 학생 정보를 데이터베이스에 저장하고 녹색 LED를 활성화하기 위해 sensor 프로세스에 “3” 메시지를 전송하며, 등록이 성공했음을 알리는 메시지를 전송한다. 반면, 학번이 이미 존재하는 경우에는 등록이 중복되었음을 알리는 메시지를 전송하고, 노란색 LED를 활성화하여 이를 표시하기 위해 sensor 프로세스에 “1” 메시지를 전송한다.

PI 4로부터 학생 정보 매칭 요청이 들어온 경우, 서버는 요청된 학번을 데이터베이스에서 검색한다. 만약 학번이 존재하지 않으면, 외부인이 감지된 것으로 간주하여 Piezo를 울리고 빨간색 LED를 활성화하기 위해 sensor 프로세스에 “2” 메시지를 전송하다. 학번이 존재할 경우에는 해당 학번에 매핑되는 비밀번호를 PI 3으로 전송하여 매칭이 성공적으로 이루어지도록 한다.

PI 3로부터 외부인 감지 신호가 들어온 경우, 서버는 즉시 Piezo를 울리고 빨간색 LED를 활성화하기 위해 sensor 프로세스에 “1” 메시지를 전송한다.

이 모든 요청 처리 과정에서 서버는 클라이언트와의 연결 상태를 유지하며, 처리 결과를 적절한 방식으로 반환하거나 필요한 명령을 실행하도록 한다. 시스템은 이러한 작업을 반복적으로 수행하며, 클라이언트 요청에 따라 적절히 대응한다.



03. 시스템 구조도

PI 2 - DATABASE

이 프로그램은 Raspberry Pi에서 GPIO 핀을 제어하고 MariaDB 데이터베이스와 통신하며, 사용자 입력에 따라 동작을 제어하는 시스템이다. 프로그램의 전체적인 동작은 다음과 같다.

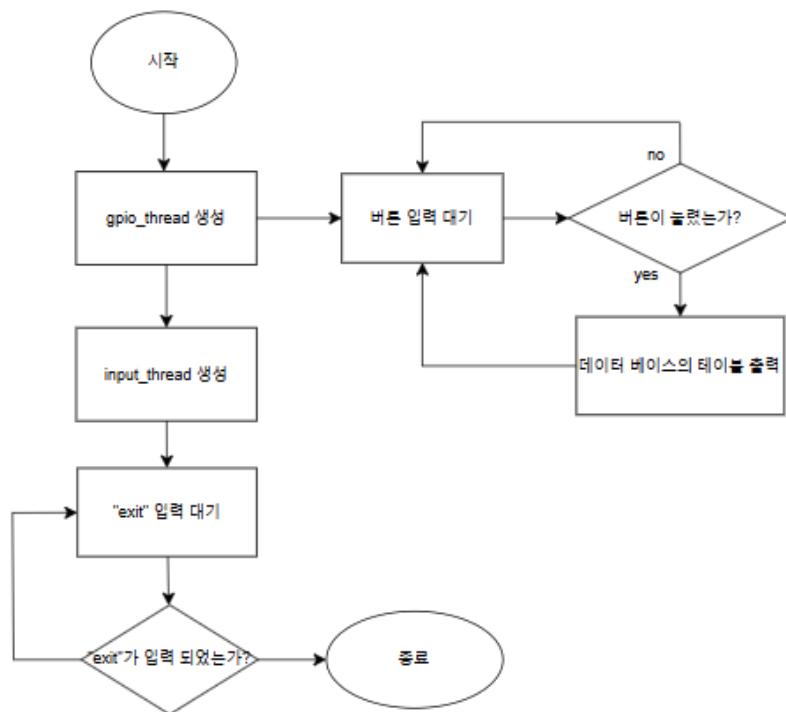
프로그램이 실행되면 먼저 MariaDB 데이터베이스와 연결을 설정한다. 데이터베이스는 localhost에 위치하며, 사용자 이름, 비밀번호, 데이터베이스 이름은 각각 설정된 값으로 초기화된다. 데이터베이스 연결이 성공하면 프로그램은 데이터 조회와 제어 작업을 수행할 준비를 완료한다.

프로그램은 두 개의 스레드를 생성하여 작업을 수행한다. 첫 번째 스레드는 GPIO 핀의 상태를 지속적으로 확인하는 역할을 한다. 이 스레드는 출력 핀에 신호를 보내고, 입력 핀에서 값을 읽는다. 입력 핀이 HIGH 상태로 감지되면, 데이터베이스에서 학생 정보를 조회하는 작업이 수행된다. 조회된 데이터는 프로그램 실행 중 콘솔에 출력된다.

두 번째 스레드는 사용자 입력을 처리한다. 사용자로부터 "exit"이라는 명령이 입력되면, 프로그램을 종료하기 위한 신호를 설정한다. 이 신호를 통해 다른 스레드와 메인 프로그램은 종료 과정을 시작한다.

프로그램 종료 시, GPIO 핀은 시스템에서 해제되고 데이터베이스 연결이 닫힌다. 모든 작업이 안전하게 종료되면 프로그램은 종료 메시지를 출력하며 마무리된다.

이 프로그램은 GPIO 핀 제어와 데이터베이스 작업을 병렬로 처리하며, 사용자 입력을 통해 시스템을 제어할 수 있는 구조를 갖추고 있다.



03. 시스템 구조도

PI 2 - SENSOR

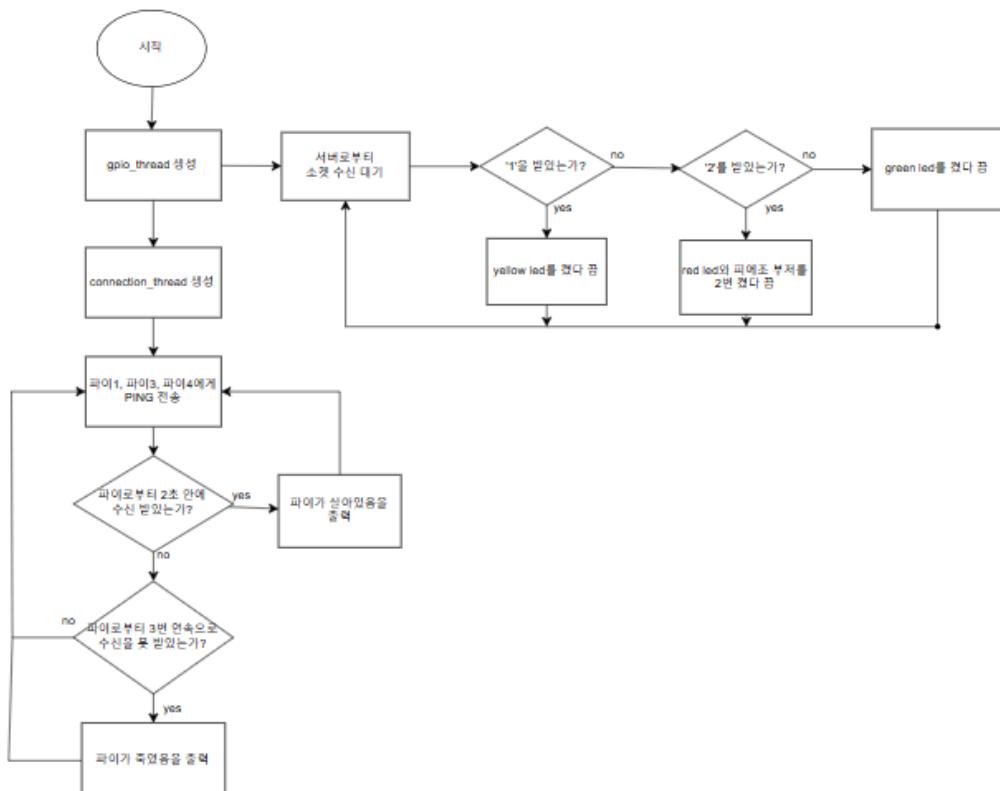
이 프로그램은 Raspberry Pi에서 GPIO 핀 제어, 네트워크를 통한 장치 간 통신, 사용자 입력에 따른 시스템 제어를 수행한다.

먼저, GPIO 제어를 담당하는 스레드는 초기화 과정에서 빨간색, 초록색, 파란색 LED 핀과 부저 핀을 시스템에 등록한다. 이후 출력 핀으로 설정하고 TCP 서버를 실행한다. 이 서버는 Java 서버로부터 메시지를 수신한다. 메시지의 내용에 따라 초록색 LED를 켜거나, 빨간색 LED와 부저를 활성화하거나, 초록색과 빨간색 LED를 동시에 켜는 동작을 수행한다.

한편, 두 번째 스레드는 네트워크의 다른 장치들과 UDP를 통해 통신한다. 프로그램이 시작되면 UDP 소켓이 생성되고, 대상 장치들에게 "PING" 메시지를 주기적으로 전송한다. 장치로부터 응답이 있을 경우, 해당 장치가 정상적으로 작동하고 있음을 기록한다. 반대로 응답이 없으면 누락 횟수를 증가시키고, 누락 횟수가 일정 기준을 초과하면 해당 장치가 비활성화 상태임을 기록한다. 만약 비활성화된 장치가 다시 응답을 보내면 정상 상태로 복구된다.

세 번째 스레드는 사용자 입력을 처리한다. 실행 중 사용자가 "exit"이라는 명령을 입력하면 프로그램이 종료 상태로 전환된다. 이 명령은 프로그램의 실행 플래그를 종료 상태로 변경하여 모든 스레드가 안전하게 종료될 수 있도록 신호를 전달한다.

프로그램이 종료 상태로 전환되면 각 스레드는 작업을 중단하고 종료 과정을 시작한다. GPIO 핀은 해제되고 네트워크 소켓은 닫히며, 모든 리소스가 안전하게 정리된다. 모든 스레드가 종료되면 프로그램은 종료 메시지를 출력하며 동작을 마친다.



03. 시스템 구조도

PI 3

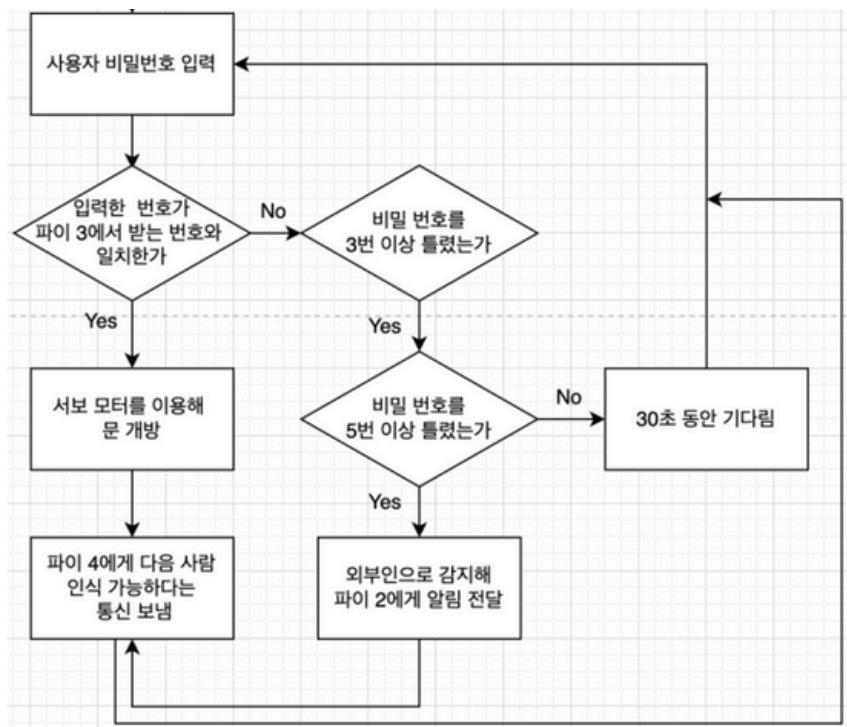
PI 3은 키패드로부터 비밀번호 입력 받고 입력을 LCD에 출력한다. 데이터베이스에 저장된 비밀번호와 키패드로 입력 받은 번호가 일치하는지에 따라 문을 개방하고, 비밀 번호 틀린 횟수에 따라 각기 다른 대응 방안을 적용한다.

먼저, PI 2로부터 인식된 사람에 대한 비밀 번호를 받으면 사용자가 비밀 번호를 입력할 수 있다. 이 때, 사용자가 입력하는 비밀 번호 자리에 따라 LCD에 *이 출력되고, 사용자가 #을 누르면 키패드 입력이 종료 되었음을 의미한다. 비밀 번호 입력이 종료되면 파이 2로부터 받은 비밀 번호와 비교해 일치하면 서보 보터를 이용해 문을 연다.

만약, 비밀번호가 일치하지 않는다면 사용자로부터 다시 입력을 받는다. 비밀 번호가 3, 4번 째 일치하지 않는 경우에는 각각 30초 씩 대기한 후 다시 키패드 입력이 가능하고, 5번째 일치하지 않으면 외부인으로 인식해 PI 2에게 외부인 감지에 대해 알림을 보낸다.

마지막으로, 한 사람이 키패드를 사용하는 동안 다른 사람이 PIR에 의해 모션 감지되면 파이의 흐름이 안 맞기 때문에 한 사람이 키패드 입력, 문 개방 또는 외부인 감지와 같이 모든 로직이 끝나고 나면 모션 감지가 될 수 있도록 모든 로직이 끝나고 나면 PI 4에게 소켓 통신을 통해 알림을 보낸다.

PI 3은 PI 2와 파이 연결 유무를 알리기 위해 통신한다. PI 2가 PING 메세지를 보내면 PONG 메세지를 통해 응답을 보내 메인 서버가 파이 연결 유무를 감지하도록 한다. 파이 연결 유무 주기적으로 알리기 위한 파이 연결 유무 통신과 메인 로직은 각각 병렬로 실행해야하기 때문에 스레드를 사용한다.



03. 시스템 구조도

PI 4

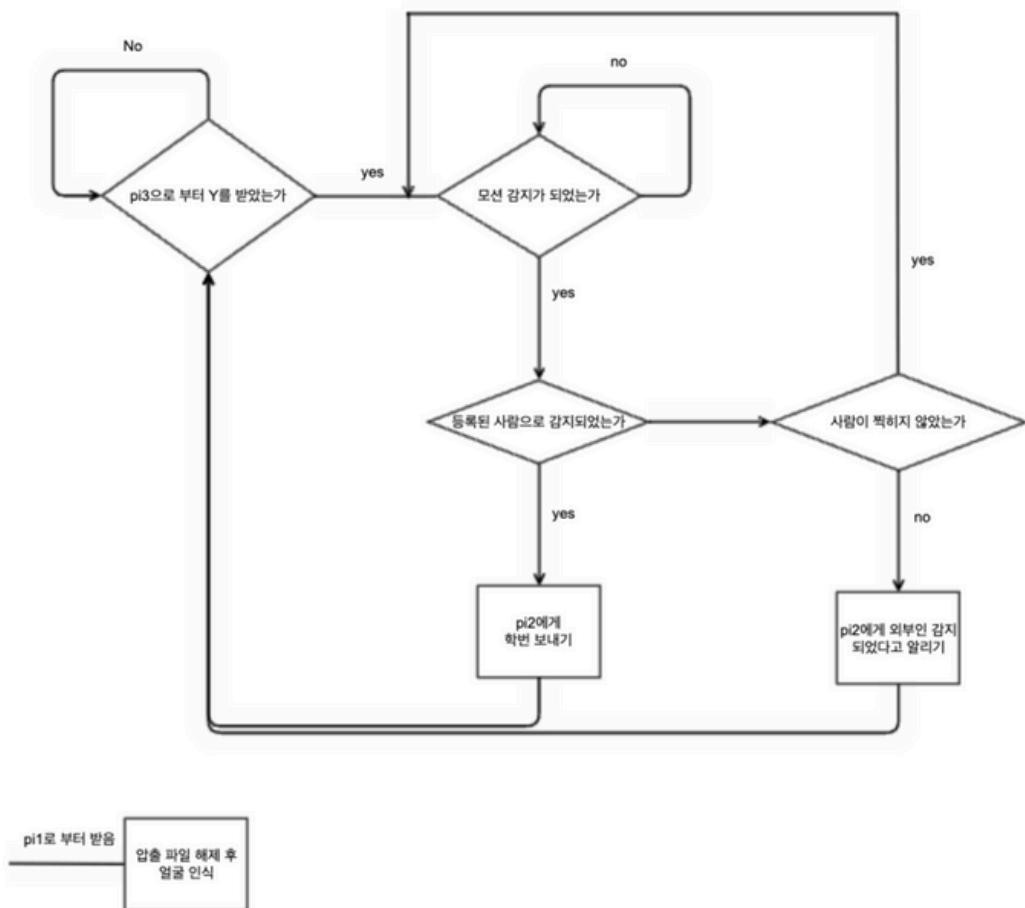
PI 4 는 인공지능을 활용하여 사람을 인식하는 역할을 담당한다.

PI 1로부터 소켓 통신을 통해 압축된 사진 파일을 받으면 해당 파일을 압축해제 하여 인공지능을 학습을 진행한다. 또한, PI 3에서부터 모션 감지를 해도 된다는 신호를 받으면 모션을 감지하여 사진을 찍는다.

사진을 찍은 후 해당 사진에 사람이 없다면 다시 촬영을 하고, 사람이 인지되었을 때에는 등록된 사람과 외부인으로 구분하여 PI 2에게 전송을 한다.

또한 사람이 비밀번호를 입력하는 도중 물체를 감지했을 때 다른 사람이 인식되는 경우 잘못된 비밀번호가 전달되는 경우가 있을수도 있고, 비밀번호 입력 횟수가 초기화가 될 수도 있기에 PI 3으로 부터 다시 모션 감지를 해도 된다는 신호를 받았을 때와 외부인으로 감지된 경우에만 다시 모션 감지를 하도록 하였다.

PI 2와 ping, pong 신호를 주고 받으며 연결 상태를 확인하였고, 핑 신호와 사람 인식 결과를 각각 동시에 전달해야했으므로 스레드를 활용하여 이를 처리했다.



04. 세부 구현 사항

PI 1

이 프로그램은 키패드를 통해 사용자로부터 정보를 입력받고, GPIO 및 I2C를 활용하여 LCD 디스플레이와 상호작용하며, 네트워크를 통해 다른 파이(PI 2, PI 4)와 통신하고 카메라를 사용하여 이미지를 촬영 및 전송하는 과정을 수행한다.

1. 초기 설정 및 GPIO 구성

- GPIO 핀 설정
 - ROWS와 COLS 배열을 통해 키패드의 행과 열을 제어할 GPIO 핀을 정의한다.
 - POUT2는 출력용, PIN은 입력용 GPIO로 설정한다.
- I2C 설정
 - LCD 디스플레이를 제어하기 위해 I2C 버스(/dev/i2c-1)를 열고, LCD 주소 (0x27)를 설정한다.
- 버퍼 초기화
 - 학번(sn), 비밀번호(pw), 방 번호(rn)를 저장할 배열과 카운터를 초기화한다.

2. 사용자 입력 처리 (input_thread)

- 키패드 입력
 - 키패드를 통해 학번, 비밀번호, 방 번호를 순차적으로 입력 받는다.
 - 입력 완료 시 '#' 키를 눌러 다음 단계로 진행한다.
 - '*' 키를 눌러 마지막 입력을 삭제(백스페이스 기능)한다.
- LCD 디스플레이
 - 입력 단계별로 LCD에 현재 입력 상태를 표시한다.
 - 입력 완료 또는 오류 시 적절한 메시지를 출력한다.
- 데이터 전송 및 응답 처리
 - 최종 입력된 데이터를 final 배열에 저장한 후 PI 2로 전송한다.
 - PI 2로부터 y 응답 시 사진 촬영을 진행하고 zip파일을 PI 4로 전송한다.
 - n 응답 시 재입력을 요청한다.

3. GPIO 제어 함수

- GPIO Export/Unexport
 - 특정 GPIO핀을 사용하기 위해 /sys/class/gpio/export 파일을 통해 활성화한다.
 - 프로그램 종료 시 /sys/class/gpio/unexport를 통해 핀을 비활성화한다.
- GPIO 방향 설정
 - 입력(IN) 또는 출력(OUT) 방향을 설정한다.
- GPIO 읽기/쓰기
 - 버튼 입력을 읽거나 LED 등 출력을 제어하기 위한 함수들

PI 1

4. 키패드 제어

- 행(Row) 핀 연결
 - 키패드의 R1~R4 핀을 라즈베리 파이 GPIO 핀에 연결한다.
 - GPIO 핀은 출력 모드로 설정하여 각 행을 활성화하거나 비활성화한다,
- 열(Column) 핀 연결
 - 키패드의 C1~C4 핀을 라즈베리 파이 GPIO 핀에 연결한다,
 - GPIO 핀은 입력 모드로 설정하고 기본 상태를 LOW로 유지한다,
 - 버튼이 눌리면 행에서 활성화된 신호가 열로 전달되어 HIGH 상태를 감지한다.
- 특정 행 핀(R1~R4)을 HIGH로 설정하여 활성화한다.
 - 열 핀(C1~C4)을 주기적으로 확인하여 버튼 눌림 여부를 감지한다.

5. LCD 제어 함수

- I2C 통신을 통한 LCD 제어
 - I2C 주소 설정
 - LCD의 고유 I2C 주소(I2C_ADDR)를 사용하여 특정 장치와 통신한다.
 - 라즈베리 파이는 /dev/i2c-1 디바이스 파일을 통해 LCD에 데이터를 전송한다.
 - 명령 및 데이터 전송
 - 명령은 lcd_byte()를 통해 I2C 버스를 사용해 전송되며, LCD의 동작을 설정한다.
 - lcd_string() 함수에서 센서 데이터를 변환하여 LCD의 특정 라인에 출력한다.
- 초기화 및 명령 전송
 - lcd_init(): LCD를 초기화한다.
 - lcd_byte(), lcd_toggle_enable(): 데이터와 명령을 LCD로 전송한다.
- 문자열 출력 및 클리어
 - lcd_string(): 특정 라인에 문자열을 출력한다.
 - lcd_clear(): LCD 화면을 지운다.

6. 카메라 제어 및 이미지 처리

- 사진 촬영
 - libcamera-jpeg 명령어를 사용하여 지정된 경로에 사진을 촬영하고 저장한다.
 - 총 50장의 사진을 촬영하여 저장한다.
 - 이미지 파일을 저장할 때 이미지의 이름은 '[학번].[촬영횟수]'로 저장한다.
- 이미지 압축 및 전송
 - 촬영된 사진들을 zip 명령어로 압축하여 images.zip 파일을 생성하고 PI 4로 전송한다.

PI 1

7. 네트워크 통신

- PI 2와의 통신
 - 데이터 전송 (`send_to_pi2`) : TCP 소켓을 통해 PI 2의 지정된 포트로 `final` 배열 값을 전송한다.
 - 응답 수신 (`receive_from_pi2`): PI 2의 응답을 TCP 소켓을 통해 수신하고, 응답(y 또는 n)을 반환한다.
- PI 4와의 통신 (`send_to_pi4`)
 - TCP 소켓을 통해 압축된 이미지 파일(`images.zip`)을 PI 4로 전송한다.
- Ping 처리 (`ping_thread`)
 - UDP 소켓을 통해 PI 2에게 PING 메시지를 수신하고, PONG 응답을 전송한다.

8. 멀티스레딩 처리

- 쓰레드 생성
 - `ping_thread`: PI 2에게 PING 메시지를 주기적으로 전송하고 응답을 수신한다.
 - `input_thread`: 사용자 입력 처리, 이미지 촬영 및 전송을 실행한다.
 - `gpio_thread`: GPIO 상태 모니터링과 프로그램 종료 시 GPIO를 정리한다.
 - `exit_thread`: 사용자로부터 'exit' 입력을 받아 프로그램을 종료한다.
- 쓰레드 동기화
 - `atomic_int running` 변수를 통해 프로그램의 실행 상태를 관리하고, 모든 쓰레드가 이를 참조하여 종료 조건을 확인한다

9. 함수별 세부 설명

- GPIO 관련 함수들
 - `GPIOExport`, `GPIOUnexport`, `GPIODirection`, `GPIORead`, `GPIOWrite`: GPIO 핀의 설정 및 제어를 위한 기본 함수들이다.
 - `gpio_export`, `gpio_unexport`, `gpio_direction`, `gpio_write`, `gpio_read`: 유사한 기능을 수행하는 보조 함수들로 코드의 중복을 줄인다.
- LCD 관련 함수들
 - LCD 디스플레이 초기화 및 데이터 전송을 위한 함수들이다.
- 키패드 입력 함수들
 - `read_keypad`: 키패드의 현재 입력 상태를 읽어 눌린 키를 반환한다.
 - `student_num`, `password`, `room_num`: 각각 학번, 비밀번호, 방 번호를 입력받는 함수들이다.
 - `final_result`: 입력받은 데이터를 특정 형식으로 포맷팅하여 `final` 배열에 저장한다.
- 통신 함수들
 - `send_to_pi2`, `receive_from_pi2`, `send_to_pi4`: PI 2와 PI 4와의 데이터 전송 및 수신을 담당한다.

04. 세부 구현 사항

PI 2 - SERVER

Server 프로그램은 Java로 작성된 서버 애플리케이션으로, 클라이언트 요청을 처리하며 데이터베이스와 상호작용하고 외부 장치와의 통신을 관리하는 역할을 한다.

1. 클라이언트 요청 수신 및 처리

- 소켓 생성 및 연결 대기
 - 서버는 포트 12345에서 TCP 소켓을 열어 클라이언트 요청을 대기한다. 클라이언트가 연결되면 요청을 처리하기 위해 RequestHandler 스레드가 생성된다.
- 스레드 풀 관리
 - 서버는 최대 10개의 동시 요청을 처리하기 위해 고정된 크기의 스레드 풀을 사용한다.

2. 요청 처리 (RequestHandler)

- 학생 정보 등록 (r로 시작)
 - 학생 정보를 데이터베이스에 저장한다. 이미 등록된 학번인지 확인한 뒤
 - 학번이 없으면 새로 저장하고, "녹색 LED"를 키기 위한 신호를 보낸다.
 - 학번이 중복되면 "중복" 메시지를 전송하고, "노란색 LED"를 키기 위한 신호를 보낸다.
- 학생 정보 매칭 (m로 시작)
 - 학번을 조회하여 매칭 결과를 반환한다.
 - 매칭 실패 시 "빨간색 LED와 부저 활성화" 신호를 보낸다.
 - 매칭 성공 시 학번에 해당하는 정보를 다른 PI로 전송한다.
- 외부인 감지 (w로 시작)
 - 외부인이 감지되었음을 나타내기 위해 "빨간색 LED와 부저 활성화" 신호를 보낸다.

3. 데이터베이스 관리 (Database)

- Singleton 패턴
 - 하나의 인스턴스로 데이터베이스 연결을 관리한다.
- 데이터베이스 초기화
 - 프로그램 실행 시 데이터베이스와 테이블이 없으면 자동으로 생성한다.
- 학생 정보 관리
 - 학번을 기준으로 학생 정보를 저장하거나 조회한다.

4. 외부 장치 통신 (Controller)

- SensorController
 - LED와 부저 제어 신호를 외부 장치로 전송한다.
- DuplicateController
 - 학번 중복 여부를 확인하는 신호를 전송한다.
- MatchController
 - 학번 매칭 결과를 전송한다.

PI 2 - DATABASE

이 프로그램은 MariaDB와 상호작용하며 사용자 입력을 처리하는 시스템이다.

1. MariaDB와의 연결

- 데이터베이스와 연결을 설정
 - localhost에서 실행되는 MariaDB에 연결한다.
 - 사용자 이름은 root, 비밀번호는 7179, 데이터베이스 이름은 dormitory_db
- 연결 성공 시 데이터베이스 쿼리를 실행할 준비 완료

2. GPIO 핀 초기화 및 제어

- GPIO 핀 설정
 - GPIO 핀 20번(PIN)은 입력 핀으로 설정한다.
 - GPIO 핀 21번(POUT2)은 출력 핀으로 설정하고 활성화(HIGH) 상태로 유지한다.
- GPIO 핀 읽기 및 쓰기
 - GPIO 핀 20번의 값을 읽어 입력 상태를 확인한다.
 - 핀이 활성화(HIGH) 상태로 변경되면 데이터베이스 쿼리를 실행하여 학생 정보를 조회한다.

3. 데이터베이스 쿼리 실행

- execute_query 함수
 - 쿼리를 실행하고 결과를 가져온다.
 - 학생 정보를 포함한 데이터베이스의 결과를 콘솔에 출력한다.

4. 사용자 입력 처리

- 사용자로부터 "exit" 명령을 입력받아 프로그램 종료
 - "exit" 명령이 입력되면 프로그램 실행 상태를 변경한다.
 - 모든 스레드가 안전하게 종료되도록 신호를 보낸다.

5. 멀티스레딩

- 프로그램은 두 개의 스레드로 구성되어 작업을 병렬적으로 수행
 - GPIO 제어 스레드
 - GPIO 핀의 상태를 읽고 데이터베이스 쿼리를 실행한다.
 - 사용자 입력 처리 스레드
 - 사용자 입력을 처리하여 종료 명령을 감지한다.

6. 프로그램 종료

- 프로그램 종료 시
 - GPIO 핀 해제 및 데이터베이스 연결 종료한다.
 - 모든 스레드가 종료된 뒤 리소스 정리한다.
 - 종료 메시지 출력 후 프로그램 종료한다.

7. 무한 반복 처리

- 프로그램은 종료 명령이 입력되기 전까지 GPIO 핀 상태를 지속적으로 확인하고 데이터베이스 쿼리를 실행
- 종료 명령이 입력되면 반복 처리가 중단되고 프로그램이 종료

PI 2 - SENSOR

이 프로그램은 GPIO 핀을 제어하고 UDP/TCP 네트워크 통신을 통해 장치 상태를 모니터링하며, 외부 명령에 따라 LED 및 부저를 동작시키는 기능을 수행한다.

1. GPIO 핀 초기화 및 제어

- GPIO 핀 설정
 - 빨간색(RED), 초록색(GREEN), 파란색(BLUE) LED 핀과 부저(BUZZER_PIN) 핀을 초기화하고 출력 모드로 설정한다.
 - GPIO 핀의 값을 설정하여 LED와 부저의 동작 제어한다.
- LED 동작
 - 특정 조건에 따라 LED를 켜고 일정 시간 후 끈다.
 - YELLOW 상태에서는 두 개의 LED를 동시에 켜는 동작 수행한다.

2. UDP를 통한 PING 메시지 전송 및 응답 처리

- PING 메시지 전송
 - 지정된 IP 주소 배열(targets)에 포함된 장치들에게 UDP를 통해 주기적으로 PING 메시지를 전송한다.
- 응답 수신 및 상태 업데이트
 - PONG 메시지를 수신하면 해당 장치의 상태를 활성(alive)으로 업데이트한다.
 - 일정 횟수 이상 응답이 누락된 장치를 "비활성(dead)" 상태로 간주하고 경고 메시지 출력한다.

3. TCP를 통한 명령 수신 및 처리

- TCP 서버 설정
 - 지정된 포트(PORT)에서 TCP 소켓을 열어 클라이언트 연결을 대기한다.
- 명령 처리
 - 클라이언트로부터 명령 메시지를 수신하여 동작 수행한다.
 - "1" 메시지: 노란색 LED 동작한다.
 - "2" 메시지: 빨간색 LED와 부저 동작한다.
 - 기본 메시지: 초록색 LED 동작한다.

4. 멀티스레딩을 활용한 병렬 처리

- 프로그램은 세 개의 스레드로 구성되어 독립적으로 작업을 수행
 - GPIO 제어 스레드
 - GPIO 핀을 초기화하고 TCP 명령을 처리하여 LED 및 부저 동작 수행한다.
 - UDP 연결 관리 스레드
 - PING 메시지를 전송하고 응답을 처리하여 장치 상태를 모니터링한다.
 - 사용자 입력 처리 스레드
 - 사용자로부터 명령을 입력받아 프로그램 종료 또는 기타 작업 수행한다.

5. 프로그램 종료

- 프로그램 종료 시
 - 모든 GPIO 핀을 비활성화하여 리소스를 정리한다.
 - 소켓 연결을 닫고 모든 스레드가 종료된 뒤 프로그램이 안전하게 종료한다.

6. 무한 반복 처리

- 프로그램은 종료 명령이 입력되기 전까지 PING 메시지 전송, TCP 명령 수신 및 처리, 사용자 입력 대기 등의 작업을 계속 수행

04. 세부 구현 사항

PI 3

이 프로그램은 GPIO 핀을 제어하고, 네트워크를 통해 데이터를 송수신하며, 사용자 입력을 처리하여 센서와 관련된 여러 작업을 수행한다.

1. 네트워크 파일 수신

- PI 2 통신
 - 비밀번호 통신
 - 서버는 포트 `23232`에서 소켓을 열고 클라이언트(PI 2)로부터의 연결 요청을 대기한다.
 - 클라이언트가 연결되면, 전송된 비밀번호 데이터를 읽어와 저장한다.
 - 외부인 알림 통신
 - 서버는 포트 `12345`에서 소켓을 열고 외부인 감지 시 PI 2에게 외부인 알림을 전송한다.
 - PING 통신
 - 프로그램은 소켓을 생성하여 포트 `54321`에서 PING 요청을 대기한다. PI 2로부터 PING 메시지가 수신되면, 즉시 PONG 메시지로 응답한다.
- PI 4 통신
 - 프로그램은 소켓을 생성하여 포트 `34343`으로 PI 4에게 Y 메세지 전송한다. 모션 감지를 다시 실행해도 된다는 의미로 전송한다.

2. LCD 제어

- I2C 통신을 통한 LCD 제어
 - I2C 주소 설정
 - LCD의 고유 I2C 주소(I2C_ADDR)를 사용하여 특정 장치와 통신한다.
 - 라즈베리 파이는 /dev/i2c-1 디바이스 파일을 통해 LCD에 데이터를 전송한다.
 - 명령 및 데이터 전송
 - 명령은 lcd_byte()를 통해 I2C 버스를 사용해 전송되며, LCD의 동작을 설정한다.
 - lcd_string() 함수에서 센서 데이터를 변환하여 LCD의 특정 라인에 출력한다.
- 초기화 및 명령 전송
 - lcd_init(): LCD를 초기화한다.
 - lcd_byte(), lcd_toggle_enable(): 데이터와 명령을 LCD로 전송한다.
- 문자열 출력 및 클리어
 - lcd_string(): 특정 라인에 문자열을 출력한다.
 - lcd_clear(): LCD 화면을 지운다.

3. 키패드 제어

- 행(Row) 핀 연결
 - GPIO 핀은 출력 모드로 설정하여 각 행을 활성화하거나 비활성화한다.
- 열(Column) 핀 연결
 - GPIO 핀은 입력 모드로 설정하고 풀다운 저항을 사용하여 기본 상태를 LOW로 유지한다.
 - 버튼이 눌리면 행에서 활성화된 신호가 열로 전달되어 HIGH 상태를 감지한다.
- 특정 행 핀(R1~R4)을 HIGH로 설정하여 활성화한다.
 - 열 핀(C1~C4)을 주기적으로 확인하여 버튼 눌림 여부를 감지한다.

PI 3

4. 모터 제어

- 모터는 PWM 신호를 통해 제어되며, 비밀번호가 일치하면 모터가 작동하여 잠금을 해제한다.
- PWM 채널을 활성화하고, 듀티 사이클을 조정하여 잠금 해제 및 복구 동작을 수행한다.
- PWM 채널 활성화
 - PWMExport (int pwmnum)
 - 사용하려는 PWM 채널을 활성화한다.
 - 해당 채널 번호(pwmnum)를 /sys/class/pwm/pwmchip0/export에 쓰면, PWM 채널이 시스템에 등록된다.
- PWM 활성화
 - PwMEnable (int pwmnum)
 - 지정된 PWM 채널을 활성화한다.
 - /sys/class/pwm/pwmchip0/pwm0/enable 파일에 1을 기록하여 PWM 출력을 시작한다.
- PWM 주기 설정
 - PwMWritePeriod (int pwmnum, int value)
 - PWM 신호의 주기를 설정한다.
 - /sys/class/pwm/pwmchip0/pwm0/period에 주기 값을 기록하여 적용한다.
- PWM 듀티 사이클 설정
 - PwMWriteDutyCycle (int pwmnum, int value)
 - PWM 신호의 듀티 사이클을 설정함.
 - 듀티 사이클은 PWM 주기 중 HIGH 상태를 유지하는 시간의 비율로, 모터의 속도와 힘을 조절한다.
 - /sys/class/pwm/pwmchip0/pwm0/duty_cycle에 듀티 사이클 값을 기록하여 설정한다.

5. 비밀번호 처리

- 비밀번호 입력 처리
 - 사용자는 키패드 입력을 통해 비밀번호를 입력한다. 입력된 비밀번호는 PI 2로부터 수신된 비밀번호와 비교된다.
 - 비밀번호가 일치하면 모터를 작동시켜 잠금을 해제하고, PI 4로 모션 감지 활성화 메시지를 전송한다.
 - 비밀번호가 일치하지 않을 경우, 최대 5번까지 재입력이 가능하다. 틀린 횟수에 따라 다음과 같은 작업이 수행된다
 - 3번 또는 4번 틀린 경우 30초 대기 후 재입력이 가능하다.
 - 5번 틀린 경우 외부인으로 간주하고 PI 2와 PI 4에 알림 메시지를 전송한다.
 - 키패드에서 입력된 비밀번호는 실시간으로 LCD에 출력된다. 사용자가 입력을 종료하면 (# 버튼) 입력된 비밀번호가 검증에 사용된다.

6. 스레드 기본 처리

- 프로그램은 세 개의 스레드로 구성되어 병렬로 작업을 수행한다.
 - 센서 및 비밀번호 검증 스레드: 센서와 키패드, LCD, 모터를 제어하고 비밀번호 검증을 처리한다.
 - PING 메시지 처리 스레드: 네트워크를 통해 PING 요청을 수신하고 응답한다.
 - 종료 명령 처리 스레드: 사용자 입력을 처리하여 종료 명령을 감지하고 프로그램을 안전하게 종료한다.

04. 세부 구현 사항

PI 4

`rec_img.c`는 네트워크를 통해 파일(압축된 이미지 데이터)을 수신, 저장, 압축 해제한 뒤 Python 스크립트를 실행하는 과정을 수행한다.

1. 네트워크 파일 수신

- PI 1
 - 포트 14141에서 TCP 소켓을 열어 클라이언트(PI 1)의 연결을 대기한다.
 - 클라이언트 연결이 수립되면 전송받은 데이터를 파일 (`/home/ruby/Desktop/case/dataset/image.zip`)로 저장한다.
 - 파일 수신: 데이터를 `BUFFER_SIZE`(1024바이트) 단위로 읽어 ZIP 파일로 작성한다.

2. 압축 해제 및 기존 데이터 삭제

- 기존 데이터 삭제:
 - `rm -rf` 명령어를 사용하여 압축 해제 경로 (`/home/ruby/Desktop/case/dataset/extracted`) 내 모든 파일 삭제한다.
 - 사진을 전송 받을 때마다 인공지능을 학습하므로 한번 학습했던 사진들을 다시 학습시키면 처음 사진에 대한 정확도만 높아질 수 있기 때문에 삭제한다.
- 압축 해제:
 - `unzip -o` 명령어를 사용해 수신된 ZIP 파일을 지정된 경로로 압축 해제. 이미 존재하는 파일은 덮어씌운다.

3. Python 스크립트 실행

- 압축 해제된 데이터를 기반으로 Python 스크립트 (`/home/ruby/Desktop/raspi_dataset+learning.py`)를 실행하여 AI 학습 작업 수행한다.

4. 무한 반복 처리

- 메인 함수에서 `receive_file_and_extract` 함수를 무한 루프(`while(1)`)로 실행하여 지속적인 파일 수신 및 처리가 가능하도록 설계한다.

PI 4

pir.c는 여러 개의 스레드를 사용하여 사진 촬영 및 모션 감지, 파일 전송 및 서버 통신을 관리한다.

1. 네트워크 송수신

- PI 2 (communicate_with_server)
 - 얼굴 인식 결과 송신
 - 포트 12345에서 TCP 소켓을 열어 얼굴 인식 후 얻은 학번은 PI 2에 전달한다.
 - PING 통신
 - 프로그램은 소켓을 생성하여 포트 `54321`에서 PING 요청을 대기한다. PI 2로부터 PING 메시지가 수신되면, 즉시 PONG 메시지로 응답한다.
- PI 3 (receive_message_on_port)
 - 포트 34343에서 소켓을 열어 PI 3으로 부터 Y 메세지를 받았을 때 모션 감지 하도록 수행한다.

2. 상수 정의 및 초기 설정

- SERVER_ADDRESS: 서버 주소 (PI 2의 IP 주소).
- MAIN_SERVER_PORT: PI 2와 통신하는 포트 번호.
- LISTEN_PORT: PI 3와의 통신 포트.
- BUFFER_SIZE: 데이터 버퍼 크기.
- SEND_FILE_PATH, TMP_FILE_PATH: 파일 경로.
- PIR_PIN: GPIO 핀 번호 (여기서는 PIR 센서).
- ERROR_PORT: PI 2와 ping-pong을 주고받는 포트.

3. GPIO 제어 함수들

- GPIOExport, GPIOUnexport: GPIO 핀을 활성화/비활성화한다.
- GPIODirection: GPIO 핀의 입출력을 설정한다.
- GPIORead: GPIO 핀의 값을 읽는다.

4. PIR 센서 제어

- PIR 센서 핀 설정
 - GPIOExport()와 GPIODirection() 함수로 PIR 센서의 GPIO 핀을 설정한다.
 - GPIOExport(PIR_PIN)은 지정된 GPIO 핀을 시스템에 등록하는 과정이다.
 - GPIODirection(PIR_PIN, IN)은 PIR 센서의 핀을 입력 모드로 설정하여 센서의 출력 신호(모션 감지 여부)를 읽을 수 있도록 한다.
- PIR 센서의 동작
 - GPIORead(PIR_PIN)로 PIR 센서에서 출력되는 신호를 읽는다.
 - LOW 신호: 모션이 감지되지 않음.
 - HIGH 신호: 모션이 감지됨.

PI 4

5. 모션 감지 및 동작

- 모션이 감지되면 (`GPIORead(PIR_PIN) == HIGH`), 카메라를 통해 사진을 촬영하고, 촬영된 이미지를 저장한 후, Python 스크립트를 실행하여 특정 처리를 진행한다.
 - PIR 센서를 통해 모션을 감지하고, 모션이 감지되면 카메라를 실행하여 사진을 촬영한다.
 - 사진을 찍은 후 Python 스크립트를 실행하여 인식된 사람 정보를 처리하고, 이를 서버에 전송한다.

6. 스레드 함수들

- `gpio_thread`: PI 3의 포트에서 메시지를 수신하고, 적절한 작업을 처리한다.
- `input_thread`: 사용자 입력을 받는 스레드다. "exit"를 입력하면 프로그램을 종료한다.
- `ping_thread`: PI 2와 ping-pong 메시지를 주고받는 스레드다. UDP 소켓을 사용하여 PI 2에서 보내는 PING 메시지를 수신하고 PONG 메시지를 반환한다.

04. 세부 구현 사항

AI

< 등록된 사진으로 인공지능 모델 학습 - raspi_dataset+learning.py >

1. 필요한 라이브러리 import

- cv2, numpy, PIL, os, random 라이브러리 import한다.
- face_cascade: OpenCV 얼굴 인식 모델
- recognizer: LBPH 얼굴 인식기 초기화
- input_dir, output_dir, save_model_dir: 이미지 저장 및 모델 디렉토리 설정한다.

2. 디렉토리 설정 및 초기화

- output_dir 디렉토리 존재 여부 확인 후, 없으면 생성한다.
- 이전 학습 데이터 제거한다. (output_dir 내 모든 파일 삭제)
- 새로운 데이터셋을 준비하기 위해 output_dir 초기화한다.

3. 이미지 증강 함수 정의

- augment_image(): 이미지 증강을 위한 함수
 - 회전, 수평 반전, 밝기 조정 등 랜덤 변환한다.
 - 원본 이미지를 증강하여 학습 데이터 확장한다.

4. 이미지 파일 처리 및 얼굴 인식

- input_dir 내 이미지 파일들을 처리한다.
 - .png, .jpg, .jpeg 파일만 처리한다.
- 이미지 읽기 및 그레이스케일 변환한다.
- 얼굴 인식은 Haar Cascade 모델로 얼굴 감지한다.
- 가장 큰 얼굴을 추출하고, 얼굴 이미지만 잘라내어 output_dir에 저장한다.
- 증강된 이미지도 생성하여 output_dir에 .aug 접미사로 저장한다.

5. 학습용 데이터 준비

- get_images_and_labels() 함수로 학습 데이터 준비
 - output_dir 내 모든 이미지 파일 읽어들여, 그레이스케일로 변환한다.
 - 파일명에서 학번을 추출하여 학습 데이터로 사용한다.

6. 학습 및 모델 저장

- recognizer.train()을 사용하여 얼굴 이미지와 학번 정보를 학습한다.
- 학습된 모델을 trainer.yml 파일로 저장한다.
- 학습된 모델을 save_model_dir에 저장한다.

7. 결과 출력

- 모델 학습 완료 후, 성공 메시지 출력한다.
- 학습된 모델이 save_model_dir에 저장되었음을 출력한다,

04. 세부 구현 사항

AI

< 학습된 인공지능을 통해 얼굴 인식 - predict.py >

1. 학습된 모델 로드

- recognizer.read(): 학습된 얼굴 인식 모델을 지정된 경로(/Users/kim-yujin/Desktop/save/trainer.yml)에서 읽어온다. 해당 모델은 이전에 학습된 데이터를 바탕으로 얼굴 인식 예측을 수행한다.

2. 예측할 이미지 파일 찾기

- predict_dir: 예측할 이미지들이 저장된 디렉토리 경로이다.
- image_files: 디렉토리에서 .png/.jpg./jpeg 파일들만 필터링하여 리스트에 저장한다.
- 가장 최근에 수정된 파일을 선택하기 위해 max() 함수를 사용하여 파일들의 수정 시간을 기준으로 최신 파일을 찾는다.

3. 이미지 파일 로드 및 확인

- 파일이 존재하는지 확인한 후, 해당 파일 경로로 이미지를 읽어온다. 이미지가 제대로 로드되지 않은 경우 오류 메시지를 출력한다.

4. 그레이스케일 변환 및 얼굴 인식

- 이미지를 그레이스케일로 변환한 후, Haar Cascade를 사용하여 얼굴을 감지한다.
 - detectMultiScale(): 이미지에서 얼굴을 감지하는 함수로, 여러 인자들을 설정하여 얼굴의 크기 및 감지 민감도를 조정한다.
- 감지된 얼굴에 대해 사각형을 그려 얼굴 위치를 표시한다.

5. 얼굴 인식 및 학번 예측

- recognizer.predict(): 얼굴 영역에 대해 예측을 수행하고, 예측된 학번과 유사도를 반환한다.
 - 유사도가 100보다 작으면 인식된 사람과 유사한 학번이 있다는 것을 의미한다.
 - 유사도 값에 따라 예측된 학번과 유사도(정확도)를 이미지에 표시한다.
- 유사도가 100 이상인 경우, id는 "unknown"으로 설정하고 유사도를 100에서 빼서 출력 한다.

6. 결과 출력

- cv2.putText(): 얼굴 인식 결과인 학번과 유사도 값을 이미지에 텍스트로 표시한다.
- cv2.imshow(): 인식된 얼굴과 예측된 학번 및 유사도 정보를 포함한 이미지를 화면에 표시한다.

7. 프로그램 종료 및 자원 정리

- cv2.waitKey(): 키보드 입력을 대기하며, 프로그램 종료 조건을 처리한다.
- cv2.destroyAllWindows(): OpenCV 창을 닫고 자원을 반납하여 종료한다.

04. 세부 구현 사항

DATABASE

1. 데이터베이스 초기화 및 연결

- Java와 C 코드 모두 MySQL/MariaDB와 연결하기 위해 데이터베이스 초기화와 연결 작업을 수행한다.
- Java에서는 JDBC를 사용하여 데이터베이스에 연결한다. 이 과정에서 MySQL JDBC 드라이버를 로드하고, 데이터베이스 URL, 사용자 이름, 비밀번호를 제공하여 연결을 설정한다. 연결이 성공하면 성공 메시지를 출력하고, 실패 시 예외를 처리하여 오류 내용을 출력한다.
- C에서는 MariaDB C API를 사용하여 연결 객체를 초기화한 후, 데이터베이스 호스트 주소, 사용자 이름, 비밀번호, 데이터베이스 이름을 설정하여 연결을 시도한다. 연결이 성공하면 성공 메시지를 출력하며, 실패 시 오류 메시지를 출력한다.

2. 데이터베이스 존재 여부 확인 및 생성

- Java 코드에서 데이터베이스가 존재하지 않을 경우, SHOW DATABASES 명령을 통해 데이터베이스가 있는지 확인한다. 데이터베이스가 존재하지 않는 경우 CREATE DATABASE 명령을 실행하여 새로운 데이터베이스를 생성한다. 이미 존재하는 경우 별도의 생성을 하지 않고 넘어간다.

3. 테이블 생성

- Java 코드에서는 CREATE TABLE 명령을 사용하여 students라는 테이블을 생성한다. 이 테이블에는 학생의 ID, 비밀번호, 방 번호를 저장할 수 있는 세 가지 필드가 정의되어 있다. 테이블이 이미 존재할 경우 에러를 방지하기 위해 IF NOT EXISTS 조건을 사용하여 중복 생성이 방지된다.
- 테이블 생성이 성공하면 성공 메시지를 출력한다. 실패 시 SQL 예외 처리를 통해 오류 내용을 출력한다.

4. 학생 정보 저장

- Java에서는 학생 정보를 데이터베이스에 저장하기 위해 INSERT INTO SQL 명령을 사용한다. 학번, 비밀번호, 방 번호를 파라미터로 받아 SQL 쿼리를 실행한다. 이를 통해 새로운 학생 정보를 데이터베이스에 추가할 수 있다. 예외가 발생하면 SQL 에러를 출력한다.

5. 학생 정보 조회

- Java에서 특정 학번에 대한 정보를 조회하기 위해 SELECT 명령을 사용한다. 학번을 기준으로 데이터베이스를 검색하고, 결과값으로 저장된 비밀번호를 반환한다. 학번이 존재하지 않을 경우 null을 반환하도록 처리되어 있다.

6. 데이터베이스와 GPIO 동작 연계

- C 코드에서는 GPIO 입력값을 감지하여 특정 조건이 만족되면 데이터베이스에서 정보를 검색한다. 이를 통해 외부 센서나 입력 신호를 기반으로 데이터베이스와의 상호작용을 수행한다.

7. 데이터베이스 연결 종료

- Java에서는 프로그램 종료 시 close() 메서드를 호출하여 데이터베이스 연결을 종료한다. 이를 통해 리소스를 해제하고 데이터베이스 연결의 누수를 방지한다.
- C에서도 mysql_close()를 호출하여 MariaDB 연결을 종료하며, 프로그램이 정상적으로 종료되도록 설계되어 있다.

05. 도전적 이슈

사진 전송 시 용량 문제

처음에는 PI 1에서 인공지능으로 학습한 YML 파일을 소켓 통신을 통해 PI 4로 전송하는 방안이 계획되었다. PI 1에서는 OpenCV를 사용하여 촬영한 이미지 파일들을 학습시켜 YML 파일로 생성한 후, 이를 PI 4로 전송하려 하였다. 그러나 YML 파일을 주고받는 과정에서 파일이 손상되는 문제가 발생하였다. 이에 따라 이미지 파일을 직접 전송하는 방식으로 변경되었다.

50장의 이미지가 포함된 파일을 전송하려고 시도하였을 때, 설정한 버퍼 크기보다 파일 용량이 커져 buffer overflow가 발생함을 확인할 수 있었다. 이러한 문제를 해결하기 위해 파일 용량을 줄이기 위해 ZIP 압축 방식을 도입하였다. 이를 통해 전송 파일의 크기를 효과적으로 줄일 수 있었으며, buffer overflow 문제를 방지할 수 있었다.

파이 통신 단절

프로젝트에서는 4개의 라즈베리 파이가 서로 주기적으로 통신을 주고받으며 프로그램을 실행하도록 설계되었다. 이러한 통신 구조에서는 각 파이 간 연결 상태를 지속적으로 확인하는 것이 중요한데, 이를 위해 PING과 PONG 메시지를 주고받으며 연결 상태를 점검하는 방식을 도입했다. 메인 서버 역할을 하는 파이가 특정 주기에 따라 PING 메시지를 전송하면, 해당 메시지를 수신한 다른 파이들이 PONG 메시지를 응답으로 보내는 구조로 통신이 이루어진다.

이러한 작업을 단일 프로그램으로 처리할 경우 시스템 자원이 과도하게 소모될 우려가 있기에, 프로그램의 효율성과 자원 관리를 위해 멀티스레딩 방식을 채택하여 구현했다. 스레드를 활용함으로써 통신 확인 작업을 독립적으로 수행할 수 있었다.

05. 도전적 이슈

서로 다른 언어의 파일을 메인 함수에서 실행

프로젝트에서는 라즈베리파이를 활용하여 데이터베이스를 구축하고자 했을 때, MySQL을 이용하여 데이터베이스를 구현하려 했으나, 라즈베리파이 환경에서는 MySQL이 직접적으로 지원되지 않는 제한이 있었다. 이를 해결하기 위해 MySQL의 기능을 지원하는 MariaDB를 도입하였으며, MariaDB를 통해 데이터베이스를 설계할 수 있었다.

또한, 프로젝트에서는 C언어로 작성된 프로그램 코드에서 인공지능 관련 기능을 구현해야 하는 상황이 있었다. 그러나 인공지능 모델 및 라이브러리는 주로 Python에서 사용 가능했기 때문에, Python으로 작성된 코드를 C언어 기반 프로그램에서 실행해야 하는 문제가 발생했다. 이 문제를 해결하기 위해 수업 시간에 학습한 `system()` 함수를 활용했다. `system()` 함수를 사용하면 외부 스크립트를 호출하여 실행할 수 있는데, 이를 통해 C언어 코드에서 Python으로 작성된 인공지능 코드를 호출하고 필요한 작업을 수행할 수 있었다.

예외 사항 처리

해당 프로젝트의 시나리오는 단순히 도어락 기능을 구현하는 것을 기본 목표로 설정하였지만, 프로젝트의 완성도를 높이고 실제 상황에서 발생할 수 있는 다양한 문제를 해결하기 위해 단순한 기능 구현에 그치지 않고 더욱 복잡하고 현실적인 상황까지 고려하며 설계되었다.

먼저, 데이터베이스 관리에서 발생할 수 있는 중복 문제를 해결하기 위해 예 저장된 학번을 입력했을 시 재등록 하지 않도록 구현했다. 또한, 사람이 아닌 물체가 인식되는 경우 다시 사람을 인식해 키패드를 시작하도록 설계했다. 이 외에도 시스템 운영 중 발생할 수 있는 다양한 시나리오를 분석해 예외 시나리오도 수행할 수 있도록 했다.

06. 계획 대비 완성도

PLAN

계획 대비 완성도

본 프로젝트는 사전 계획에 따라 도어락 시스템 구현과 데이터베이스, 인공지능 구현을 성공적으로 완료했다. 또한, 추가적으로 라즈베리 파이 간 연결이 끊어졌을 때 이를 감지하는 기능도 구현했다. 한편, 사용자가 터미널에서 "exit"을 입력했을 때 프로그램이 종료되도록 하는 기능을 구현하려고 노력했으나, blocking 모드와 관련된 문제를 해결하지 못해 완전한 구현에는 이르지 못했다. 이는 향후 도전 과제로 남겨둔다.

1. 사전 계획

01. ~ 11월 17일	사생 정보 등록 기능 구현
02. ~ 11월 27일	등록된 정보를 데이터 베이스 저장 및 인공지능 모델 학습
03. ~ 12월 5일	사용자가 비밀 번호 입력, 얼굴 인증, 출입문 개방 기능, 메인 서버 구현
04. ~ 12월 10일	촉소 모형 제작 & 시뮬레이션, 최종 발표 자료 제작

2. 진행 계획

Aa 할일	Status	완료일	Aa 할일	Status	완료일
파이 1 - 키패드 센서	Done	2024/11/17	파이 3 - 키패드 센서	Done	2024/11/26
파이 1 - LCD 센서	Done	2024/11/16	파이 3 - LCD 센서	Done	2024/11/28
파이 1 - 카메라 모듈	Done	2024/11/12	파이 3 - 모터 센서	Done	2024/11/28
파이 1 - 버튼 센서	Done	2024/11/24	파이 3 → 파이 2 비밀번호 틀렸을 때 통신	Done	2024/12/05
파이 1 → 파이 4 이미지 통신	Done	2024/12/03	파이 3 → 파이 4 문열고 난 뒤 사이클 끌婊	Done	2024/12/05
파이 1 → 파이 2 학번, 비번, 방번호 통신	Done	2024/12/03	파이 3 → 파이 4 시작할 때	Done	2024/12/03
	Not started			Not started	
파이 2 - 메인 서버	Done	2024/11/28	파이 4 - 카메라 모듈	Done	2024/11/17
파이 2 - 피에조 부저	Done	2024/11/26	파이 4 - PIR 모션 센서	Done	2024/11/27
파이 2 - RGB LED 센서	Done	2024/11/27	파이 4 → OpenCV	Done	2024/11/27
파이 2 → 파이 3 학번에 따른 비번 통신	Done	2024/12/04	파이 4 → 파이 2 예측 결과 (학번) 통신	Done	2024/12/04
파이 2 - 버튼눌러서 db 쿼리보내기(추가)	Done	2024/12/04		Not started	
			인공지능	Done	2024/12/02
			데이터 베이스	Done	2024/12/01

07. 고찰

곽민서

시스템 프로그래밍 과목에서 팀으로 프로젝트를 진행하면서 협업의 이점을 깨달았습니다. 팀원들과 회의를 진행하고 역할을 나누어서 작업을 한 뒤 합치는 과정을 보는게 이전까지는 많이 해보지 않았던 경험이었기 때문에 좋은 경험이 되었다고 생각합니다. 혼자하는 프로젝트가 아니다보니 책임감도 느낄 수 있었고 저의 부족한 부분을 팀원들이 채워주어서 고마움도 느껴졌습니다. 프로젝트를 하면서 쓰레드에 대해 처음 알게 되었고 소켓 통신을 처음 사용하였는데 개인적으로도 성장할 수 있는 좋은 기회였다고 생각합니다.

김유진

이번 프로젝트를 통해 하드웨어의 특성과 팀 프로젝트 시 중요한 자세를 배울 수 있었습니다. 특히 I2C 통신을 통해 장치 간 데이터 송수신 방법을 이해하고, LCD 센서에서 커서를 제어하며 명령어와 데이터를 구분하는 중요성을 깨달았습니다. 또한 GPIO 핀을 제어하여 하드웨어를 제어하는 방법을 배우며 하드웨어와의 상호작용을 이해할 수 있었습니다. 또한, 반복적인 소켓 통신 등에서 코드를 모듈화하는 것이 중요하다는 것을 느꼈습니다. 마지막으로, 팀원 간 의견을 나누고, 서로의 역할과 책임을 명확히 이해해 책임감을 갖고 임하는 태도가 중요하다는 것을 느꼈습니다.

노승현

이번 프로젝트를 통해 데이터베이스, GPIO, socket 통신 등 다양한 기술을 통합하면서 시스템 설계와 구현의 중요성을 깨달았습니다. 각 기술을 통합하는 과정에서 mysql이 라즈베리파이에서 호환이 안 되는 의존성 문제가 있었고 UDP 통신에서 blocking 모드로 인해 무한 대기 상태가 발생하는 문제도 발생했습니다. 이 과정에서 기술 간 의존성과 통합의 복잡성을 이해했고, 설계 단계에서 명확히 사용하고자 하는 기술에 대한 이해가 중요함을 깨달았습니다. 특히, SD카드 손상으로 인해 백업해 두지 않았던 코드를 모두 다시 작성해야 하는 큰 실수를 겪으며 데이터 백업의 중요성을 뼈저리게 깨달았습니다. 이번 경험은 시스템 통합 설계와 관리의 어려움뿐 아니라 신중한 데이터 관리 습관의 필요성을 배우는 계기가 되었습니다.

최유정

이번 프로젝트를 통해 라즈베리파이와 같은 임베디드 하드웨어를 다루는 경험을 쌓을 수 있었습니다. 특히, 제한된 자원을 가진 환경에서의 개발이 일반적인 데스크톱이나 노트북을 사용하는 개발과는 상당히 다르다는 점을 체감했습니다. 하드웨어의 저장 용량과 처리 성능이 제한적이기 때문에, 구현하고자 하는 기능에 대해 사전 조사를 철저히 진행하고, 효율적인 설계를 우선시해야 한다는 교훈을 얻었습니다. 이를 통해 자원의 한계를 고려하는 사고방식과 최적화된 코드를 작성하는 능력을 기를 수 있었으며, 이러한 경험은 향후 다양한 제약 조건 속에서 개발을 진행할 때 큰 자산이 될 것이라 생각합니다.