|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 캡스톤디자인 회의록 (회의비 사용 내역서) | | | | | | | | | | |
| 회의내용 | 프로젝트 주제 선정을 위한 팀 내 논의 | | | | | | | | | |
| **교과목명** | 컴퓨터 종합설계1 | | | | | | **담당교수** | | 정진우 | |
| **과제명** | IoT기반 엘리베이터 스마트 관리 시스템 | | | | | | **팀명** | | CS1415 | |
| **회의일자** | 20.09.01(월) | | **회의시간** | 18:00 ~ 20:00 | | | **회의장소** | | *Webex video conferencing* | |
| 회의록 | ※ 회의록은 최소 10줄 이상, 최대한 구체적으로 작성할 것 (수기 작성X, 한글파일로 작성)  ※ 부족할 경우 별지 활용 가능  [1주차] - 진행 프로젝트에 대한 소개 및 알고리즘 설명  프로젝트 소개   * IoT 기반 엘리베이터 스마트 관리 시스템은 기존 엘리베이터 관리 및 할당 시스템의 한계점을 극복하고 효율적인 운행을 위한 목표를 가진 주제이다. * 기존의 엘리베이터 운행은 실시간적인 요소가 존재합니다. 하지만 운행에 대한 실시간 정보만 존재하고 실제 대기하는 인원에 대한 정보는 활용하지 않는다. * 따라서, 본 프로젝트에서는 IoT 기기를 활용하여 실제 엘리베이터의 대기 인원과 엘리베이터 내부의 인원 정보들을 수집 및 활용하고, 이를 Genetic Algorthm을 활용하여 각 상황에서 가장 최적의 운행과 할당을 할 수 있도록 하는 것을 목표로 한다..   Genetic algorithm 사용  이유:   * 현실 세계에 사용하기에는 Rule-based 방식에는 너무 많은 변수가 존재하고 경우의 수가 존재하여 한계가 있다 * 다양한 경우에 대해 랜덤하게 시도하고 그로 부터 학습하여 다양한 경우에도 적용이 가능하다.   기존의 genetic algorithm:   * 단순 특정 층에서 call이 올 경우 엘리베이터 할당에 대해서 효율을 높이는 것을 목표로 하였다.   한계:   * 엘리베이터가 꽉 차있는 경우 멈추면 안되고, 같은 층에도 대기열이 길면 여러 대를 나눠서 할당해야 한다. 이에 대한 고려가 없다   수정방법:   * 단순 hall call 여부를 0과 1로 표현하던 기존의 방식이 아닌 엘리베이터 내의 인원 수, 대기열의 수를 정수값으로 받아서 각 승강기의 capacity를 고려하며 할당한다.   상수:   * number of floor * number of cars * Passive time * Inter floor time   변수:   * Hall call floors * Car floors * Car destination floor * number of people in each elevator   알고리즘 설명:   * 총 n층이 존재한다고 한다면 1층 부터 n-1층 까지 올라가는 call과 n층부터 2층까지 내려가는 call을 2n -2 열을 가지는 배열에 표현한다. * 각 행은 엘리베이터를 표현하여 엘리베이터가 특정 hall call에 반응하지 않으면 0이 들어가고 반응한다면 태울 수 사람수로 표현 * 배열에 데이터를 기반으로 도착 시간을 예측하여 가장 빠르게 도착하는 엘리베이터만 다음 세대로 넘겨준다. * 이를 반복한다   [2주차] - 피드백 반영을 통한 프로젝트의 당위성 및 차별성  주제 선정 배경   * 신축 건물이 늘어가면서 엘리베이터와 엘리베이터를 사용하는 사람들의 수가 증가한다. * 하지만 엘리베이터의 효율성 측면에서는 적게 고려한다. * 따라서 사용자 입장과 전력 측면에서 효율적인 엘리베이터 알고리즘 설계가 필요하다.   엘리베이터 발전방향 추세:   * 엘리베이터 교통량에 대한 인공지능 분석을 통해 예상 운행시간 및 대기시간을 예측하여 빠르게 엘리베이터 탑승이 가능하도록 발전하고 있다.   기 논문과의 차별성   * Genetic algorithm을 사용한 기존 논문의 성과   - 각 목표 층에 대한 호출의 균등한 승강기 할당  - 대기 시간의 감축  - 승강기 Stop 수 감축  - 승강기 전체 이동 시간의 감축  개선 사항:   * 승강기 최적 할당 측면 : 실시간 요소들에 대한 데이터를 추가적으로 도입하고 학습을 진행하여 발생할 수 있는 모든 상황에 대한 최적의 해결책을 도출할 수 있도록 모델을 생성 및 학습시킬 수 있도록 한다. * 승강기 효율 측면 : IoT 기기 및 Object Detection을 활용하여 시스템 파라미터에 number of clients, car capacity 와 같은 내부 인원 수 계수에 영향을 미치는 파라미터들을 추가하여 알고리즘에 적용할 것.   기대 효과:   * 사람들이 실시간으로 최적의 엘리베이터를 할당받아 각 개인의 시간을 절약할수 있음 * 엘리베이터의 정차 횟수를 감소시킴으로써 에너지 효율을 높임 * 이산화탄소 배출을 감소시켜 환경적인 측면에서의 이점 확보   [3주차] - 기존 프로젝트 주제의 방향성 수정 및 새로운 프로젝트 제안  방향성 수정 : 기존 프로젝트의 피드백을 반영하여 주제 수정   * 엘리베이터 실시간 모니터링 시스템   - 기존에 상용화 되어있는 실시간 버스의 위치 및 혼잡도를 제공하는  서비스에서 착안하여 엘리베이터에 적용할 것을 제안    - 건물 내의 각 엘리베이터에 대한 내부 인원, 방향, 중간 호출 수 등을 제공  - 목표 : 원격으로 엘리베이터 실시간 상태를 모니터링 및 특정 엘리베이터 호출  기능을 제공하여 사용자의 엘리베이터 대기시간을 최소화  -문제점:  - 원격 호출에 대한 신빙성  - 확립되지 않은 인프라  - 분산되지 않은 엘리베이터 환경  - 해결 방안:  - 호출에 대한 응답 여부 판단 (비콘) 후 제재  - 오프라인을 통한 호출 제공  - 엘리베이터 호수기 별 정보 제공 : 현재 위치 , 방향, 중간 호출 수, 엘리베이터  내부 무게(%)    새로운 프로젝트 제안   * 분산화 결제 시스템 : 혼잡한 시간 식당 및 술집과 같은 요식업 종의 계산 과정이 비효율적 → 분산화 결제 시스템 착안 * 매장 내 테이블에 결제할 수 있는 리더기 마련 * 통신 API를 통해 메인 포스기와 각 테이블의 포스기 결제 요청 및 응답 * 메인 포스기와 결제 API 간 결제 요청 및 응답     [4주차] - 새로운 프로젝트 주제의 방향성 수정   * 새로운 주제에 대한 피드백에 따른 방향성 수정 * 지문인식 기반 간편 결제 시스템   -결제에 필요한 work-flow를 간소화하기 위해 고안  -지문이라는 바이오 정보와 클라우드를 융합하여 결제 시스템에 융합  - 사용 시나리오:  1. 은행에서 계좌 및 카드에 대한 지문 정보 등록  2. 키오스크 및 단말기에서 주문 확인 후 지문으로 간편한 결제  - work-flow:     * 기 시스템과의 차별점   - 생체 인증 기반 결제의 간편함  - 취약계층의 보다 간편한 사용  - 기대 효과:  - 생체인식 및 암호화 중요성 증대에 따른 관련 핀테크 산업 육성 촉진  - 정부의 신분증 모바일화 와의 일원화를 통한 신분 증명과 간편 결제의 융합  - 개인 결제 정보의 수집을 통해 추가적인 빅데이터 산업 육성 촉진  - 생체 인증을 통해 결제수단의 간편화를 이루어 사용자의 편의를 증대  - 사용자 편의에 따른 소비 문화 촉진과 내수 경제 활성화  [5주차] - 피드백에 따른 기존 주제 개발 추진   * 프로젝트에 대한 개선점   - 엘리베이터 내부 혼잡도에 대한 고려  엘리베이터에 탑재되어 있는 무게센서를 통해 승강기에 대한 진입, 진출자를 계수  →  계수된 인원에 대해서는 진행방향 내에 있는 목표층들에 대한 확률을 부여  →  현 시점의 진입 및 진출자에 대한 계수와 미래의 진입 및 진출자에 대한 예상이 가능  - 엘리베이터 외부 혼잡도에 대한 고려    서로 다른 연구에서 대부분 비슷한 주기로 엘리베이터를 사용함  → 상식적으로도 출근시간, 퇴근시간, 점심 시간에 사람이 몰림  반복되는 패턴이라면 특정 시간에 위치별 탑승, 하차 인원 예측 가능   1. 과거 정보를 기반으로 만들어진 Look up table을 사용하여 특정 시간 외부 혼잡도 예측 2. 실제 값을 반영하여 Look up table 지속적인 업데이트    1. 8시 24분에 1층에서 엘리베이터가 열렸는데 20명이 탐   8시~9시 1층 탑승 = (13.2 \* 현재까지 관측 일수 + 20)/ (현재까지 관측 일수 + 1)  - 최적의 엘리베이터에 대한 추천    - 위의 엘리베이터 내/외부 혼잡도를 고려하여 사용자에게 원거리에서 엘리베이터의 혼잡도 및 예상도착 시간을 제공해주고, 나아가 탑승할 엘리베이터를 추천, 호출  [6주차] - 프로젝트에 대한 세부 사항 설정   * 개선 사항   - 회사 건물과 주택의 전반적인 엘리베이터 사용량 유사성 존재(건물의 특성은 큰 영향이 없음)  →  But, 주중과 주말에 대한 약간의 차이 존재(요일별 특성은 영향이 있음)  →  주중과 주말에 대한 차별화 필요   * 내부 혼잡도 계산 방법 개선 사항 * 상기 센서 모듈(100)은 엘리베이터 입구지역 바닥에 설치되어 해당지역에 존재하는 물체의 중량을 측정한다 * 상기 센서 모듈(100)은 엘리베이터 입구지역 바닥에 설치되어 해당지역에 존재하는 물체의 중량을 측정한다   →  무게 정보 서버 전송 및 이를 기반으로 내부 인원 파악 가능  추가적인 고려 사항   * 사람 외의 물체, 연령,성별 등의 정보를 고려하지 않았을 시의 문제 * 무게 정보의 인원수로의 변환 방법 * 외부 혼잡도 설정 방안 * 실제 사용처에서 특정 기간(1~2달) 사용하여 값을 구한 후 실사용 * 학교 데이터 이용(이용 트래픽 및 인원 데이터(CCTV)) * 시간 당 엘리베이터 호출 횟수에 매핑   eg) 시간당 호출 횟수가 100회라면 호출마다 만원일 가능성이 큼  시간당 호출 횟수가 1회라면 호출마다 적은 인원이 사용할 가능성이 큼  [7주차] - 복잡도 계산 방식에 대한 결정 사항   * 내부 혼잡도 계산 방식의 결정(CCTV 기반) * Deep Learning 기반의 Detecting을 통해 탑승객에 대해 Head Counting 실시 * → 계수된 인원에 대해 평균 성인의 부피 및 무게치 적용을 통해 내부 혼잡도 계산      * 외부 혼잡도 계산 방식의 결정 * 과거 정보를 기반으로 만들어진 Look up table을 사용하여 특정 시간 외부 혼잡도 예측 * 실제 값을 반영하여 Look up table 지속적인 업데이트   + 8시 24분에 1층에서 엘리베이터가 열렸는데 20명이 탐   8시~9시 1층 탑승 = (13.2 \* 현재까지 관측 일수 + 20)/ (현재까지 관측 일수 + 1)  외부 혼잡도 적용 이전, 초기화 작업을 위해 1달 간의 데이터 수집 기간 필요  → 데이터 수집 이후, 외부 혼잡도 적용 실시  [8주차] - 테스트 및 수정,보완을 위한 시뮬레이터 개발 및 클래스 다이어그램   * 시뮬레이터 개발      * Baseline : 2019 카카오 Blind 테스트 2 Simulator   + 유저들의 이동 정보 (출발 층, 도착 층, 순서) 추가   + 다양한 환경에 대한 구현 * 클래스 다이어그램 * 구현 계획   테스트 환경 개선 (1주)   * 엘리베이터 4대 이상 허용 (코드 수정) * 층수 설정 허용 (코드 수정) * 데이터 수집 및 추가 (sampling)   Back-end (3주)   * Controller, Scheduler 세분화 (코드 수정) * Traffic Server 구현   + Outer Traffic controller   + Inner Traffic controller     - Counting Human (CCTV + a)   Front-end (2주)   * 사용별 WorkFlow 세분화 및 구체화 * 서버와 통신에 필요한 data 및 request 구체화 * 디자인 개선 * UI/UX 최적화   [9주차] - 내부 혼잡도의 최선책 및 차선책 강구   * 내부 혼잡도의 최선책 * Deep learning 기반의 Head counting * 내부 혼잡도의 차선책 * 하중센서(130)을 이용하여 엘리베이터에 탑승한 인원 또는 다른 물체의 전체 중량을 측정하고, 엘리베이터가 운행될 수 있는 안전 중량 기준보다 엘리베이터 내부 중량이 적은 경우, 카메라(110) 또는 거리측정센서(120)가 조합된 수단을 이용하여 탑승 인원을 측정하거나, 엘리베이터 내부 가용공간 면적을 측정하여, 엘리베이터 외부 대기자가 몇 명이나 탑승할 수 있는지 판단할 수 도 있다. * Park. S. 엘리베이터 탑승 정보를 안내하는 시스템 및 엘리베이터 탑승 정보를 표시하는 터치 패널. PCT/KR2011/003038(2011)   → 내부 탑승 인원을 파악해야하는 것이 과제 중의 하나인 거의 모든 특허에서 내부 카메라를 이용한 내부 인원 계수가 최선책으로 가정되고 사용되고 있다.   * 차선책으로 엘리베이터 거리측정센서/광센서 와 하중센서를 조합하는 방식으로 기술되어 있다.   <본 과제의 내부혼잡도 계산에 대한 차선책>  센서를 설치할 수 있다고 가정하였을 때에,  엘리베이터 내부 바닥면에 위치한 하중 센서와  엘리베이터 내부 천장면에 적외선 거리측정 센서를 설치하여  조합된 결과로 내부 탑승 인원을 산정할 것이다.  [10주차] - 모듈 개발 및 구현 현황   * 시뮬레이터 * 엘리베이터 최대 탑승 인원은 새로운 엘리베이터 객체 한대를 생성할 때 마다 파라미터로 max passengers를 정해줄 수 있는 부분이여서 딱히 수정이 필요없음. * go 언어는 동적 배열 형태인 slice를 제공, cars 라는 길이와 용량이 0인 엘리베이터 데이터 타입 슬라이스를 생성하고 carCount 만큼 cars 안에 엘리베이터 객체를 슬라이스에 추가하는 형태 * 따라서 simulator.py 에서 count를 바꾸어서 실행해보았는데 start api 에서 400이 날아와서 start requests 를 보낼 때 api.go 에서 start api 에 걸려있는 에러 핸들링 부분을 확인함 count, 즉 엘리베이터가 4대 이상일 경우 400이 날아오는 것을 확인하고 이 부분 코드 제거 * 웹 스크립트 * html 코드로 엘리베이터의 대수는 6대까지 추가 * 건물이 25층을 넘는 경우는 거의 없을 것 같아 25층으로 유지 * go 코드 전체 파악하고 이해하고 수정하는 데에 시간을 많이 사용해서 자바 스크립트 부분은 수정하지 못함. * 자바 스크립트 코드 수정해서 역할에 맞게 움직이게 한 뒤, 테스트 케이스 추가하여 돌려볼 예정   [11주차] - 내부혼잡도 및 외부혼잡도 구현   * Inner traffic   * Outer traffic    [12주차] - 유튜브 발표/데모 기획   * 프로젝트 개요 * 20세기 말부터 현재까지도 엘리베이터의 설치량 및 사용량은 꾸준히 증가하고 있다. 건물이 더욱 높아지고 엘리베이터가 분산됨에 따라 같은 목적지를 가지고 있다고 하더라도 어떤 엘리베이터를 타는지에 따라 도착 시간이 천차만별이다.   -->   * 특정 건물 내의 분산된 각 엘리베이터에 대한 혼잡도 및 예상 대기 시간을 제공해줌으로써 사용자에게 효율적인 엘리베이터 선택을 유도할 수 있다. * 프로젝트 추진 내용 * 1. CCTV 출입 인원 계수 * 엘리베이터 내부의 CCTV 영상 데이터를 활용하여 실시간으로 출입 인원을 계수  각 인원에 id를 부여하여 인덱싱 * 2. 내부 혼잡도 계산 * 엘리베이터 내부 출입 인원에 따라서 내부 혼잡도를 계산  엘리베이터가 이동하며 인원이 타고 내릴 때 마다 남은 인원의 하차 예측 확률을 계속해서 재설정 * 3. 외부 혼잡도 계산 * 특정 시간대의 특정 층에서 몇 명의 인원이 호출 요청을 이루고 탑승을 하는 지에 대한 과거의 데이터의 축적을 활용하여 Lookup table 을 만들어주어 예상 대기 시간 제공에 활용 * 4. Simulator * 엘리베이터의 현 위치와 외부 호출에 대한 정보를 실시간으로 출력할 수 있는 시뮬레이터 구현 내부/외부 혼잡도를 시뮬레이터에 적용 * 5. Application * 사용자가 내/외부 혼잡도 기준으로 어플리케이션 단에서 예상 도착 시간, 내부 인원을 제공 * 6. work - flow * 기대 효과 * 엘리베이터 사용자에게 대기 시간 및 혼잡도를 알려주어 효율적인 엘리베이터 선택을 유도한다.  --> 결과적으로 기존 엘리베이터 사용자들이 비효율적인 엘리베이터 탑승으로 인해 낭비되었던 시간을 줄여주어 사용자의 편의성을 개선하였다.   [13주차] - 최종 전시회 발표 기획   * 프로젝트 개요 * 우리나라에 승강기가 70만대이고 우리나라는 연간 설치 대수가 전 세계에서 3위인 것을 알 수 있다. 또한 엘리베이터가 늘어남에 따라 엘리베이터는 더욱 고도화된 방향으로 발전하고 있다. 이처럼 20세기 말부터 현재까지도 엘리베이터의 설치량 및 사용량은 꾸준히 증가하고 있다. * 엘리베이터가 많아짐에 따라 사용자에 대한 편의성을 개선하는 방향으로도 연구가 활발함. 편의성이라고 하면 가장 먼저 떠오르는 것이 엘리베이터 대기 시간의 감소이다. 건물이 더욱 높아지고 엘리베이터가 분산됨에 따라 같은 목적지를 가지고 있다고 하더라도 어떤 엘리베이터를 타는지에 따라 도착 시간이 천차만별이다. * 프로젝트 목표 * 한 건물 내에 사용자가 선택할 수 있는 엘리베이터가 여러 대 일 때 엘리베이터의 출발 층, 목적 층을 입력 받아 특정 엘리베이터의 현재 혼잡도, 예상 도착 시간 등의 정보를 제공해줍니다. 따라서 궁극적으로 사용자에게 효율적인 엘리베이터 선택을 유도하여 사용자의 편의성을 제공하는 것이 본 프로젝트의 목표이다. * 프로젝트 추진 내용  1. CCTV : 엘리베이터 내부의 CCTV 영상 데이터를 활용하여 실시간으로 출입 인원을 계수하고, 각 인원에 id를 부여하여 인덱싱 2. 내부 혼잡도 : 내부 혼잡도란 엘리베이터 내부의 탑승 인원에 대한 혼잡의 정도를 계량화 한 것이다. 앞선 CCTV 출입 인원 계수 결과를 토대로 엘리베이터 내부 출입 인원에 따라서 내부 혼잡도를 계산한다. 엘리베이터가 이동하며 인원이 타고 내릴 때 마다 남은 인원의 하차 확률을 예측하여 계속해서 재설정해줌으로써 엘리베이터 내부 혼잡도를 계산할 수 있다. 3. 외부 혼잡도 : 외부 혼잡도란 엘리베이터 외부의 대기 인원에 대한 혼잡의 정도를 계량화 한 것이다. 외부 혼잡도의 계산은 특정 시간 대의 특정 층에서 몇 명의 인원이 호출 요청을 이루고 탑승을 하는 지에 대한 과거의 데이터의 축적을 활용하여 Lookup table 을 만들어준다. 그리고 만들어진 Lookup table을 예상 대기 시간 제공에 활용한다. 4. 시뮬레이터 : 엘리베이터의 현 위치와 외부 호출에 대한 정보를 실시간으로 출력할 수 있는 시뮬레이터를 구현하고 앞서 언급한 내부 혼잡도를 시뮬레이터에 적용하였다. 최대 층 수를 25층으로 두고 최대 엘리베이터의 수를 4대로 산정하였으며 특정 층에서 올라가거나 내려가는 콜을 눌렀을 때 어떤 엘리베이터가 가장 먼저 도착하는 지를 가시적으로 확인하였다. 5. 어플리케이션 : 시뮬레이터가 정상적으로 작동하는 것을 확인한 후, 사용자가 내/외부 혼잡도 기준으로 어플리케이션 단에서 출발 층과 목적 층을 입력하였을 때에 최종적으로 사용자에게 예상 대기 시간, 예상 내부 인원을 제공 해주었다. 6. Work flow : 사용자에게 출발 층과 목적 층에 대한 정보를 APP을 통하여 입력 받아 서버로 넘겨준다. 과거의 시간 별 탑승 데이터 기반 Lookup table을 이용한 외부 혼잡도를 예측하고, 엘리베이터 내부 CCTV object detection을 통한 내부 혼잡도 계산한다. 그리고 트래픽 서버를 통한 혼잡도 계산 및 시뮬레이터, APP 간 통신을 통해 혼잡도 및 예상 대기 시간을 APP을 통하여 사용자에게 가시적으로 제공해준다. 8. 최종 결과물 10. 결론 및 기대효과 : 엘리베이터 사용자에게 대기 시간 및 혼잡도를 알려주어 효율적인 엘리베이터 선택을 유도한다.   결과적으로 기존 엘리베이터 사용자들이 비효율적인 엘리베이터 탑승으로 인해 낭비되었던 시간을 줄여주어 사용자의 편의성을 개선하였다.   [14주차] - 전시회 보고  [15주차] - | | | | | | | | | |
|  | **소속학과** | **성 명** | | | **자필서명** | **소속학과** | | **성 명** | | **자필서명** |
|  |  | | |  |  | |  | |  |
|  |  | | |  |  | |  | |  |
|  |  | | |  |  | |  | |  |
|  |  | | |  |  | |  | |  |

|  |  |
| --- | --- |
| **회의록** |  |
|  |  |