공 학 사 학 위 논 문

클러스터링 기법을 이용한 스마트 엘리베이터 운영관리 애플리케이션

2016년 12월

부 경 대 학 교

컴 퓨 터 공 학 과

손 관 영　최 금 강　황 유 경

공 학 사 학 위 논 문

클러스터링 기법을 이용한 스마트 엘리베이터 운영관리 애플리케이션

지도교수 박 흥 복

이 논문을 공학사 학위논문으로 제출함.

2016년 12월

부 경 대 학 교 공 과 대 학

컴 퓨 터 공 학 과

손 관 영 최 금 강 황 유 경

목 차

그 림 목 차

표 목 차

클러스터링 기법을 이용한 스마트 엘리베이터 운영관리 애플리케이션

손 관 영　최 금 강　황 유 경

부 경 대 학 교　컴 퓨 터 공 학 과

요　　약

본 논문에서는 건물의 고층화에 따른 아파트 및 중소 규모의 빌딩들의 층수가 높아짐에 따라 사용자의 보다 향상된 엘리베이터 접근 제어와 편리성을 제공하기 위해 기존 엘리베이터 시스템을 확장하여 빅데이터 로그 분석 기법 중 클러스터링(K-평균 군집분석)을 통한 스마트 엘리베이터와 스마트 모바일 시스템을 결합한 제어 모델을 제안한다. 엘리베이터 호출 데이터를 이용한 클러스터링을 통해 엘리베이터의 최적의 위치를 찾아내고, 스마트폰을 이용하여 엘리베이터의 상태를 파악하고 제어를 할 수 있게 한다. 이를 통해 사용자의 엘리베이터 대기시간을 감소시키고 효율적인 원격제어를 통해 편리성을 향상 시킬 수 있다.

Smart elevator operation management mobile application using clustering techniques

Son Gwan-yeong Choi Geum-kang Hwang You-kyung

Department of Computer Engineering,

College of Engineering,

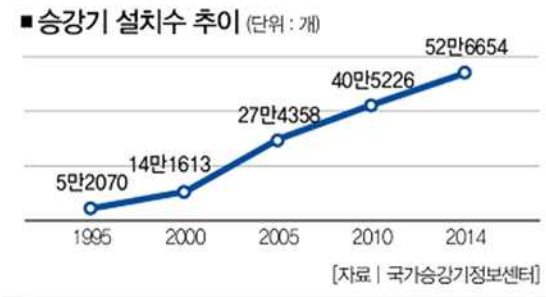
Pukyong National University

Abstract

It's being a trend that contemporary buildings are getting higher. And in line with this, we suggest the smart elevator operation management mobile application which is designed using clustering techniques to improve user's elevator access control and convenience. This clustering technique using the elevator's calling data make it possible to find its best position, and know the state of it using a smart phone as well. With this, not only the users' waiting time can be decreased, but also their convenience can be improved with the remote control system that got efficient a lot.

1. 서론

현대의 엘리베이터는 일상생활 및 산업 전반에 걸쳐 거의 모든 분야에 적용되고 있으며 중요한 수직 수평 교통수단으로서의 역할을 수행하고 있다. 우리나라는 좁은 국토와 인구 및 산업의 도시 집중화로 인해 대부분의 건축물에 엘리베이터가 설치되고 있으며 저층의 건물에서도 편의성과 경제성 건물의 디자인 측면에서 설치되고 있다. 국내의 엘리베이터 대수는 지속해서 증가하고 있으며, 최근 오피스빌딩, 주상복합, 공동주택의 보급이 급격히 증가하여 전국에 약 43만대에 이상의 엘리베이터와 에스컬레이터 등의 승강설비가 운행되고 있으며 매년 2만 5천 대 이상 신규 설치되고 있다. [1]



<그림 1> 국내 엘리베이터 설치 수 추이

(출처 : 국가 승강기 정보센터)

최근 신축되는 대부분 건축물은 고층화라는 특징을 지향하고 있으며, 이로 인하여 엘리베이터가 유일한 이동수단이 되었다. 건물의 고층화에 따른 아파트 및 중소 규모의 빌딩들의 높이가 높아짐에 따라 저속부터 고속, 쾌적함과 승차감이 우수한 고성능에 이르기까지 다양한 엘리베이터들이 보급 설치되고 있다. 이러한 이동수단을 이용한 사용자들의 다양한 요구 또한 증가하고 있다. [2]

따라서 본 논문은 더욱 향상된 사용자 접근 제어와 편리성을 제공하기 위해 기존 엘리베이터 시스템을 확장하여 클러스터링을 통한 인공지능 제어 시스템인 '스마트 엘리베이터'와, 스마트 모바일 시스템을 결합한 제어 모델을 제시한다. '스마트 엘리베이터'란 자체 내장된 소프트웨어를 통해 효율적으로 엘리베이터를 운영하고 또한 엘리베이터에 IoT(Internet of Things)를 접목함으로 인해 스마트폰으로 엘리베이터의 상태를 확인 및 제어하는 것을 말한다.

본 논문의 세부목적은 다음과 같다. 기존 연구에 따르면, 대기 시간이 120초를 초과하기 시작할 때, 사용자의 이용 만족도가 급격히 떨어지기 때문에[3] 엘리베이터가 서버로부터 클러스터링 데이터를 받아 사용자가 호출할 층수를 미리 찾아감으로써 대기시간을 감소시킨다. 그리고 사용자의 직접적인 엘리베이터 접근이 아닌 원격으로 엘리베이터 상태를 확인하고 조작할 수 있게 하여 사용자의 상황에 맞게 엘리베이터를 사용할 수 있으므로 사용자가 효율적으로 엘리베이터를 구동시킬 수 있다.

스마트 제어 시스템을 구현하기 위해서 데이터 클러스터링을 이용한다. 기존의 엘리베이터는 사용자의 호출 되는 신호가 데이터로 따로 저장되지 않으므로, 사용자 호출 데이터를 필요한 데이터로 인지하고 저장한다. 저장한 데이터는 데이터 클러스터링을 통하여 사용자가 많이 호출한 층수와 자주 사용되는 시간을 이용해 엘리베이터를 효율적으로 사용되게 한다. 애플리케이션을 통해 엘리베이터를 제어하기 위해 MCU를 사용한다. 사용자들의 엘리베이터를 이용하려는 신호를 서버가 받아 엘리베이터의 상태를 알려준다. 애플리케이션을 통한 호출이 특정 시간 이내일 때는 사용할 수 없도록 하여 호출의 남용을 막는다.

본 논문의 2장에서는 기존 엘리베이터 시스템 구조 및 환경에서의 문제점과 관련 연구에 대해 설명한다. 3장에서는 제안 시스템의 구성과 클러스터링 알고리즘에 관해 서술한다. 4장에서는 제안된 시스템 구현모델을 제시하고, 5장에서는 관련 연구와 비교하여 시스템을 분석하고, 6장에서 결론을 맺는다.

2. 관련 연구

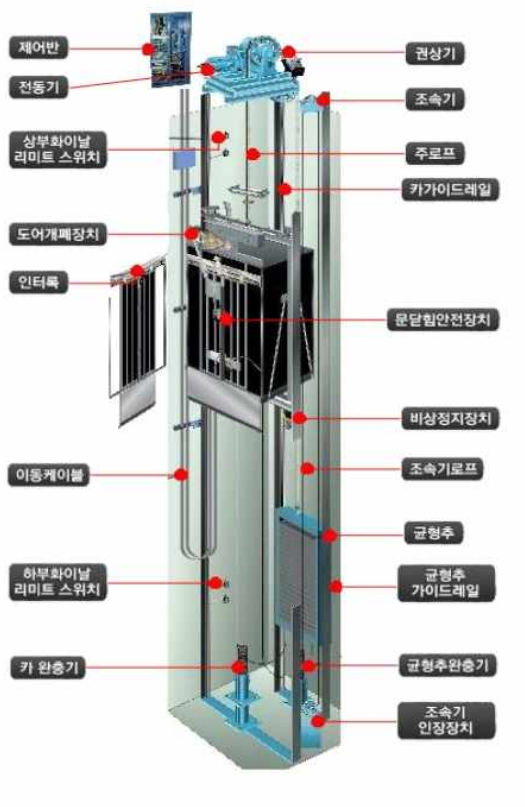
2.1 엘리베이터 시스템

2.2.1 엘리베이터의 정의

일반적으로 엘리베이터라고 하면 승강기로 통칭하고 있으나 엘리베이터와 승강기는 차이가 있다고 본다. "승강기"라 함은 건축물, 기타 공작물에 부탁 되어 일정한 승강로를 통하여 사람이나 화물을 운반하는 데 사용되는 시설로서 엘리베이터, 에스컬레이터 등 산업자원부령으로 정하는 것을 말한다. 따라서 승강기는 더 광범위한 의미를 포함한다고 볼 수 있다. [5]

2.2.2 엘리베이터 구성

엘리베이터를 구성하는 요소로 전동기, 전자 브레이크, 트랙션머신, 제어장치, 조속기, 자동 착상 장치, 완충기, 카, 전동 문 닫힘 장치, 균형추 등 다양한 요소들을 포함하고 있으며 이들은 제어장치를 통해 시스템을 제어한다. 엘리베이터 내부의 제어장치는 CAN(Controller Area Network) 통신을 통해 카 내부와 홀을 통해 데이터정보를 주고받는다. 엘리베이터 구조는 그림 2와 같다. 엘리베이터 시스템에서 사용되는 CAN은 Bosch사에서 차량제어를 위한 통신 시스템으로 개발되었으나, 산업용 분산제어 분야에서 원거리 통신, 네트워크의 단순화, 배선 비용 절감 및 설치의 용이성 등에 의해 활용도가 높다. [6]



<그림 2> 엘리베이터 구성도[7]

2.2.2 엘리베이터 시스템 연구 분야

엘리베이터의 수가 증가함에 따라 엘리베이터 시스템이 다양한 분야로 진화되고 있다. 기존의 엘리베이터 관련 연구와 개발은 대부분 엘리베이터의 효율적인 제어 및 그룹관리 등을 통한 엘리베이터 운행에 관한 연구나 엘리베이터 감시 및 통합 관련 연구들이 집중됐으며, IT 융합형 엘리베이터 모델에 대한 연구가 거의 이루어지지 않고 있다. 최근 들어 사용자의 편리성을 향상하기 위해 엘리베이터 원격 제어에 대한 연구가 진행되고 있다. [6]

그러나 기존의 연구는 사용자가 모바일 기기를 사용하여 엘리베이터 시스템에 접근하여 제어하더라도 근본적으로는 기존과 같은 사용자가 직접 엘리베이

터를 호출하여 사용한다는 점에 문제가 있다. 사용자가 최대의 편리를 얻을 수 없고 근본적인 문제를 해결하지 못하여 대기시간 차이가 크게 나지 않는다. 에너지 효율 면에서도 이는 좋지 않은 문제다. 호출의 남용문제도 해결하지 못하였다.

그리고 제한적인 조건에서 엘리베이터 운영에는 장대기 발생을 극소화하는 작업에 대해 부족함이 있다. 고정된 평가항목만을 사용하는 종래의 제어방식에는 제어목적을 종합적으로 평가할 수 없으며 수시로 변화하는 다양한 교통상황을 포괄로 다룰 수 없으므로 고도의 효율성을 보장할 수 없다.[8] 이러한 효율성을 극대화하기 위해 시스템의 불확실한 상태변화와 애매한 선택구조에 대처할 수 있도록 제어 시스템을 구축하고자 하는 연구 또한 이루어지고 있지만, 참고문헌 9에서 제시하고 있는 연구는 기존 엘리베이터 시스템을 단순화하여 실제로 쓰이게 되면 일어날 무한대기 문제, 사용자가 많아질 경우 효율성 문제가 있어 스마트한 엘리베이터 설계의 모델로는 미비하다.

2.2 Clustering

2.2.1 클러스터링 개념

클러스터링(clustering)이란 주어진 데이터 집합을 서로 유사성을 가지는 몇 개의 클러스터로 구분해 나가는 과정으로, 하나의 클러스터에 속하는 데이터 점들 간에는 서로 다른 클러스터 내의 점들과는 구분되는 유사성을 갖게 된다.[4]

데이터마이닝(data mining)은 많은 양의 데이터 속에서 쉽게 드러나지 않는 유용한 정보를 찾아내는 과정이다. 데이터마이닝은 대용량(massive)의 관측 가능한 데이터를 기반으로 숨겨진 지식, 기대하지 못했던 패턴, 새로운 법칙과 관계를 발견하고 이를 바탕으로 의사결정 등을 위한 정보로 활용하는 것이다. 데이터 마이닝 기법으로는 군집분석(cluster analysis), 연결 분석(link analysis),

판별 분석(neural network) 등의 분석 기법이 있다. [10]

데이터 마이닝에서 클러스터링 방법은 기존의 통계, 기계학습, 패턴인식에서 쓰이던 방법에 부가적으로 데이터베이스 지향적인 제약 사항들(제한된 메모리 양, I/O 시간 최소화 등)을 첨가한 것으로서, 최근의 멀티미디어 데이터와 같이 혼합되고 다양한 다차원 데이터를 효율적으로 분류해 나가기 위한 방안으로 연구되고 있다. [4]

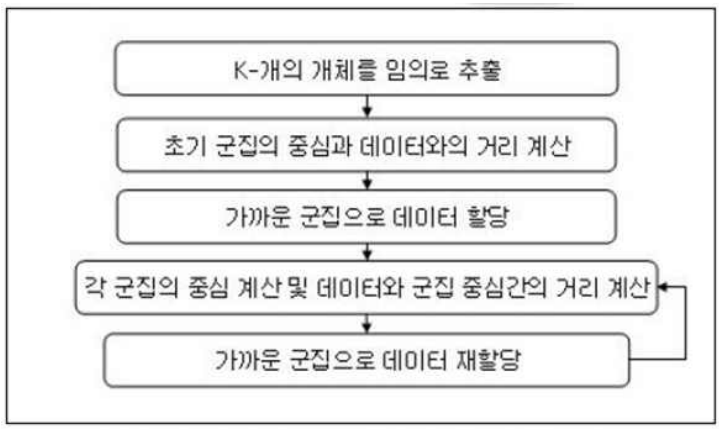
2.2.2 k-평균 군집분석(Cluster analysis)

군집분석은 다양한 특성을 가진 수많은 데이터를 비슷한 성질의 데이터끼리 묶어주는 데이터마이닝의 한 기법으로서 군집의 수 혹은 군집의 구조에 대한 가정이 없으며, 오직 데이터들 사이의 상사성 또는 비상사성(거리)에 의하여 군집을 형성하고, 형성된 군집의 특성을 파악하여 군집들 사이의 관계를 분석하는 기법이다. 군집분석을 수행하기 위한 자료를 정리한 다음에는 묶이는 각 데이터들 간의 유사성 또는 비유사성의 정도를 측정하는 기준척도가 필요하다. 두 개체 사이의 거리의 종류에는 유클리드 거리, 유클리드 제곱 거리, Mahalanobis 거리, Minkowski 거리 등이 있으나 일반적으로 아래식과 같이 정의되는 유클리드 거리를 많이 사용한다.

$$d_{ij} = \sqrt{(X_i - X_j)'(X_i - X_j)} \qquad <식\ 1>$$

여기서 d_{ij}는 유클리드 거리를 의미하고 X_i, X_j는 개체를 의미한다.

기준 척도를 정한 후에는 실제로 대상들을 군집화를 해나가야 한다. 비계층적 군집화 방법에서 가장 일반적으로 사용되는 k-평균 군집분석은 데이터를 k개의 군집으로 임의로 부할을 하여 군집의 평균을 대푯값으로 분할해 나가는 방법으로 데이터들을 유사성을 바탕으로 재배치를 하는 방법이다. MacQueen(1967)이 제안한 k-평균군집분석 알고리즘은 그림 3과 같다.[12]

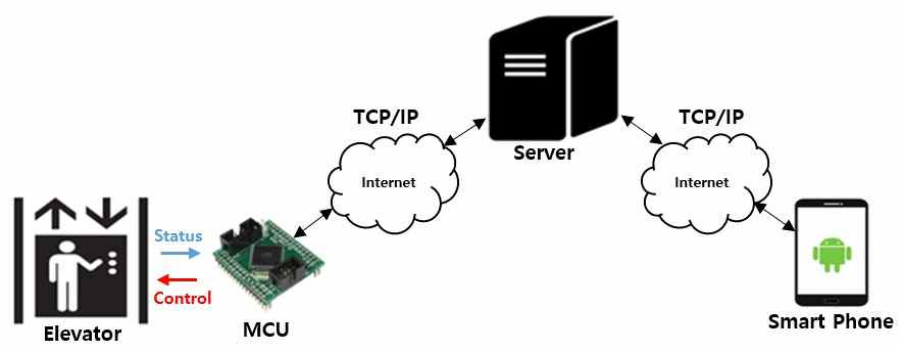


<그림 3> k-평균 군집분석 수행단계

3. 시스템 설계

3.1 제안 시스템의 구성

본 논문에서 제시한 엘리베이터 시스템은 그림 4와 같이 클라이언트(스마트폰), 서버, 엘리베이터 제어 모듈(MCU)로, 크게 3분류로 나눌 수 있다.



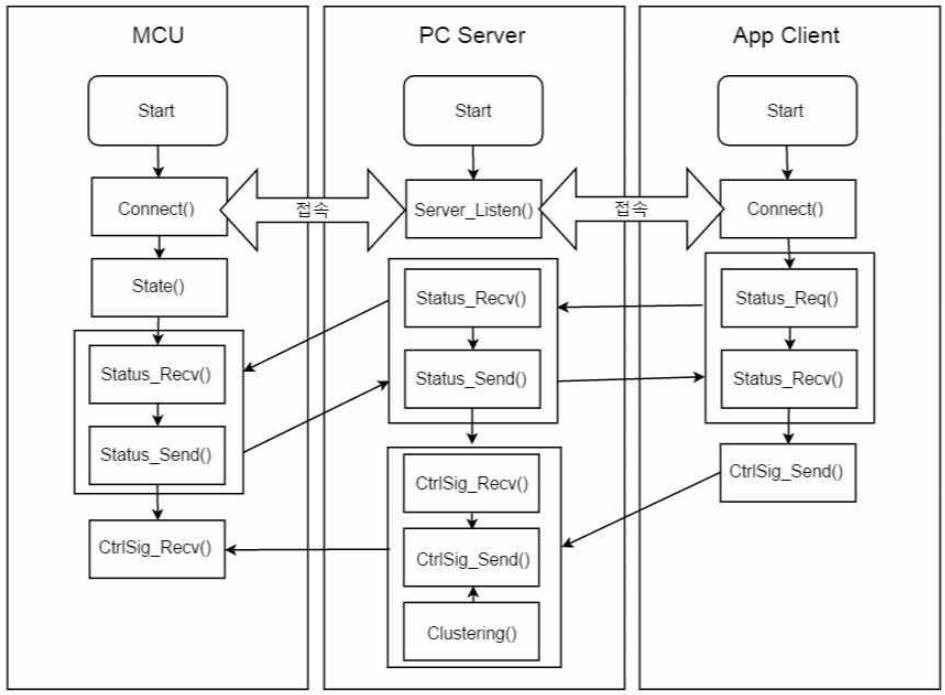
<그림 4> 원격 제어 시스템 구성도

클라이언트(스마트폰)가 엘리베이터를 이용하기 위해서 안드로이드 앱을 통해 서버에게 신호를 보내면 TCP/IP 프로토콜을 사용하여 신호가 서버에게 전달되고 서버는 받은 신호를 엘리베이터 제어모듈에 전송하게 된다. 이렇게 하면 클라이언트들은 스마트폰 앱을 통해서 엘리베이터의 현재 위치와 상태를 실시간으로 알 수 있고, 엘리베이터를 앞에서 누르지 않고도 앱을 통하여 엘리베이터 제어 시스템에 접속함으로써 원격제어할 수 있다. 엘리베이터 제어모듈은 실시간 위치와 상태를 실시간으로 서버에게 전송해 준다.

본 논문을 통해 구현하고자 하는 스마트 엘리베이터 운영관리 애플리케이션을 구현하기 위해 설계한 흐름도는 그림 5와 같다. 연산 및 중계담당을 하는 PC Server와 제어담당을 하는 MCU(Micro Controller Unit)는 항상 연결되어

있고, 인터페이스 역할을 하는 App Client는 요청이 있을 때마다 서버와 연결하게 된다.

MCU는 엘리베이터의 상태정보를 State() 함수를 통해 가지고 있으며 이를 PC Server에게 Status_Send() 함수를 통해 주기적으로 전송하게 된다. PC Server는 MCU에게 Status_Send() 함수를 통해 전송되는 제어정보를 CtrlSig_Recv() 함수를 통해 PC Server 저장장치에 기록해 두었다가 Clustering() 함수를 통해 나온 결괏값으로 MCU에게 CtrlSig_Send() 함수를 통하여 제어 신호를 보내 자동으로 제어하게 된다.



<그림 5> 시스템 흐름도

App Cient가 PC Server에게 Status_Req()함수를 통하여 엘리베이터의 상태를 요청하게 되면 PC Server는 가지고 있던 엘리베이터 상태정보를

Status_Send() 함수를 통하여 App Client에게 보내게 된다. App Client는 이렇게 받은 엘리베이터 상태정보를 바탕으로 CtrlSig_Send() 함수를 통해 서버에게 엘리베이터 제어 신호를 보내게 되고 서버는 이를 MCU에게 전달하여 App Client가 엘리베이터를 제어하게 된다.

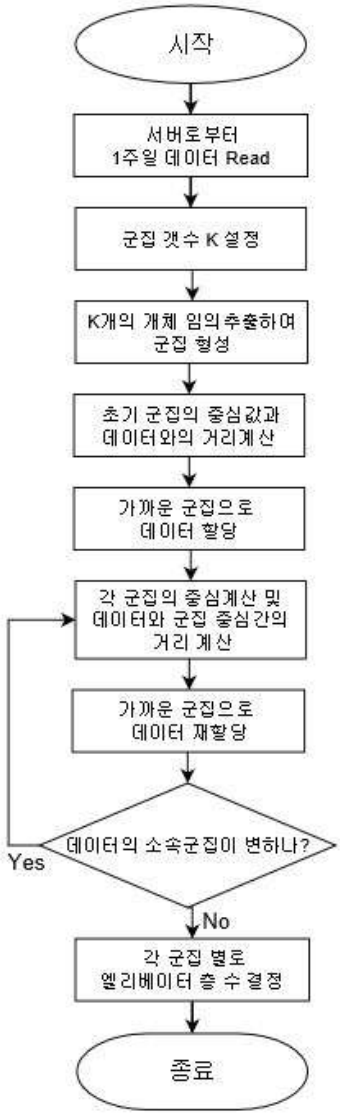
참고문헌 13은 단순히 앱과 서버 그리고 제어모듈 간의 상태정보만 주고받지만 본 논문의 흐름도는 상태정보를 주고받을 뿐만 아니라 서버가 오고 가는 제어 신호를 저장하여 자체 클러스터링을 통해 엘리베이터를 제어한다는 점에 차이가 있다.

3.2 클러스터링 알고리즘

스마트 제어 시스템을 구현하기 위해서 데이터 클러스터링을 이용한다. 시간대별로 엘리베이터 호출의 빈도를 분석하여 호출의 빈도가 높은 시간대에 가장 효율적인 층수에 엘리베이터가 자동으로 위치한다. 데이터 클러스터링을 통해 데이터를 분류하기 위해 K-평균 군집 분석을 사용한다.

K-평균 군집분석을 하기 위한 데이터는 일주일간의 엘리베이터 시간대별 사용 데이터를 바탕으로 수행하게 된다. 최대한 사용자의 사용 시간대와 엘리베이터의 동작을 일치시키기 위해서 K-평균 군집분석은 서버에 의해 평일 데이터와 주말 데이터로 나뉘어 매주 한 번씩 수행되며 1개월 동안 데이터를 축적하며 계속해서 군집을 수정해 나간다. 그 이후의 시간이 지남에 따라 1개월이 지난 데이터는 자동으로 삭제된다.

다음의 그림 6은 K-평균 군집분석을 위한 알고리즘이다.



<그림 6> K-평균 군집분석 알고리즘

먼저 서버에 저장된 일주일간의 데이터를 읽어와 K-평균 군집분석에서 가장 먼저 설정해 줘야 할 K개의 군집을 정한다. 그리고 K개의 데이터를 임의로 추출하여 군집을 형성하고 초기 군집과의 거리를 시간 간격의 차를 이용하여 계산하고 가까운 군집으로 데이터를 할당한다. 데이터가 할당되면 초기 군집이 잘

못 형성됨을 보완하기 위해 각 군집의 중심 데이터를 다시 계산하고 데이터를 재할당 한다. 이것을 군집이 변경되지 않을 때까지 계속해서 반복 수행한다. 군집이 더 이상 변경되지 않으면 K-평균 군집분석이 완료되었다고 판단하고 엘리베이터는 시간대별로 형성된 각 군집의 최소 시간과 최대시간 사이를 빈도수가 많은 시간대로 알게 되고 그 군집의 평균 층수로 엘리베이터가 움직이게 된다.

위 클러스터링 알고리즘을 가지고 데이터값들을 18개, 즉 n=18로 가정하였을 경우 클러스터링을 진행할 경우 K값은 다음과 같다.

$$k = \sqrt{\frac{n}{2}} \qquad \text{<식 2>}$$

식 2를 이용하여 k값을 구하면 k = $\sqrt{18/2}$ = 3.

따라서 k 값이 3이 된다. 이를 이용한 클러스터링 과정은 아래 그림 7과 같다.

n=18	k=3.랜덤값		평균값	재형성	평균값	재형성
7:14:00 AM(1->40)	9:20	7:14	8:53	7:14	8:53	7:14
8:40:00 AM(1->40)		8:40		8:40	1	8:40
8:45:00 AM(1->30)		8:45		8:45		8:45
9:10:00 AM(1->20)		9:10		9:10		9:10
9:20:00 AM(1->25)		9:20		9:20		9:20
10:10:00 AM(1->30)	10:10	10:10		10:10		10:10
11:42:00 AM(38->40)		11:42	12:27	11:42	12:27	11:42
11:58:00 AM(40->38)		11:58		11:58	38	11:58
12:05:00 PM(40->3)		12:05		12:05		12:05
12:11:00 PM(30->5)		12:11		12:11		12:11
2:20:00 PM(40->1)		14:20		14:20		14:20
4:13:00 PM(29->3)	18:00	16:13	18:12	16:13	18:12	16:13
5:49:00 PM(35->4)		17:49		17:49	34	17:49
5:52:00 PM(33->1)		17:52		17:52		17:52
6:00:00 PM(30->1)		18:00		18:00		18:00
6:30:00 PM(40->1)		18:30		18:30		18:30
6:40:00 PM(36->20)		18:40		18:40		18:40
8:20:00 PM(35->1)		20:20		20:20		20:20

<그림 7> 클러스터링 적용 과정

① k-개의 개체를 임의로 추출

18개의 데이터 AM7:14~ PM8:20를 가지고 클러스터링을 적용 시켜보면 처음에 임의로 값을 3가지 정하게 된다. 9:20 , 10:10 , 18:00이 임의의 값으로 추출되었다.

② 초기 군집 중심과 데이터와의 거리 계산 후 가까운 군집으로 데이터 할당

각 군집의 평균을 구한다, 각 군집별로 데이터가 하나이므로 그 데이터가 평균이 된다. 평균을 중심값으로 한다. 9:20과 10:10 중앙값을 구한다. (9:20 + 10:10) /2 = 9:45 중앙값과 데이터 값의 거리를 계산하여 다시 할당한다. 7:14, 8:40, 8:45, 9:10, 9:20 은 중앙값 9:45와 가까움으로 9:20분의 군집으로 할당한다. 이와 같이 10:10의 군집과 18:00의 군집을 구한다.

③ 군집의 중심값이 변하지 않을 때 까지 재할당.

9:20군집의 평균값을 구한다(= 8:37), 10:10군집의 평균값을 구한다(=12:15), 18:00군집의 평균값을 구한다(=18:12) 앞의 ①② 과정을 반복한다.

④ 완료

평균값이 변하지 않고, 데이터의 이동이 없다면 클러스터링이 완료된 것이다.

4. 시스템 구현

4.1 시스템 환경

스마트 엘리베이터 시스템은 H/W와 S/W로 구성된다. H/W 및 S/W 서버, 엘리베이터 제어기, 클라이언트로 구성되어 있다. 스마트 엘리베이터 시스템의 인터페이스를 구현하기 위한 시스템 환경은 다음 표 1과 같다. 하드웨어부분에서 서버는 PC로 네트워크와 유선으로 연결되어 있고 DB서버, 데이터 수집, 클러스터링 연산을 수행한다. 엘리베이터 제어기는 아두이노를 사용하여 엘리베이터를 제어한다. 클라이언트는 스마트폰을 통해서 엘리베이터 사용자가 사용한다.

소프트웨어 부분에서 서버는 java로 구현하고 DB는 MySQL을 사용하여 서버와 연동한다. 엘리베이터 제어기는 아두이노에서 엘리베이터를 제어하는 프로그램을 수행한다. 클라이언트는 엘리베이터 제어 앱을 통해서 제어신호를 엘리베이터에게 보낸다.

<표 1> 인터페이스 구현 시스템 환경

	구분	항목	설명
h/w	서버	PC	DB 서버 데이터 수집, 연산
	엘리베이터 제어기 (MCU)	아두이노	엘리베이터 제어
	클라이언트	스마트폰	엘리베이터 사용자
s/w	서버	이클립스 (Java) MYSQL	DB 서버 데이터 수집, 연산
	엘리베이터 제어기 (MCU)	스케치 (C)	엘리베이터 제어
	클라이언트	안드로이드 스튜디오 (java)	엘리베이터 제어 앱

4.2 클러스터링 알고리즘 구현

구현에 앞서 엘리베이터 호출 데이터를 저장할 테이블을 구성한다. Mysql을 사용하여 'clus' 데이터베이스 공간을 만들고 'datas' 라는 테이블을 생성한다. call_day는 엘리베이터를 호출한 날짜, call_time은 엘리베이터를 호출한 시간, call_floor은 엘리베이터를 호출한 층, is_weekend는 평일과 주말을 구분할 구분자 역할을 한다. 그림 8은 임의로 구성한 데이터베이스를 출력했을 때의 화면이다.

```
mysql> select * from clus.datas;
+------------+----------+------------+------------+
| call_day   | call_time | call_floor | is_weekend |
+------------+----------+------------+------------+
| 2016-09-08 | 07:14:00  |          1 |          1 |
| 2016-09-08 | 08:40:00  |          1 |          1 |
| 2016-09-08 | 08:45:00  |          1 |          1 |
| 2016-09-08 | 09:10:00  |          1 |          1 |
| 2016-09-08 | 09:20:00  |          1 |          1 |
| 2016-09-08 | 10:10:00  |          1 |          1 |
| 2016-09-08 | 11:42:00  |         38 |          1 |
| 2016-09-08 | 11:58:00  |         40 |          1 |
| 2016-09-08 | 12:05:00  |         40 |          1 |
| 2016-09-08 | 12:11:00  |         30 |          1 |
| 2016-09-08 | 14:20:00  |         40 |          1 |
| 2016-09-08 | 16:13:00  |         29 |          1 |
| 2016-09-08 | 17:49:00  |         35 |          1 |
| 2016-09-08 | 17:52:00  |         33 |          1 |
| 2016-09-08 | 18:00:00  |         30 |          1 |
| 2016-09-08 | 18:30:00  |         40 |          1 |
| 2016-09-08 | 18:40:00  |         36 |          1 |
| 2016-09-08 | 20:20:00  |         35 |          1 |
+------------+----------+------------+------------+
18 rows in set (0.00 sec)
```

<그림 8> 엘리베이터 데이터베이스

다음 그림 9는 서버의 클러스터링 알고리즘 중 각 군집의 중심을 계산하고 데이터와 군집 중심 간의 거리를 계산하여 가까운 군집으로 데이터를 계속 재

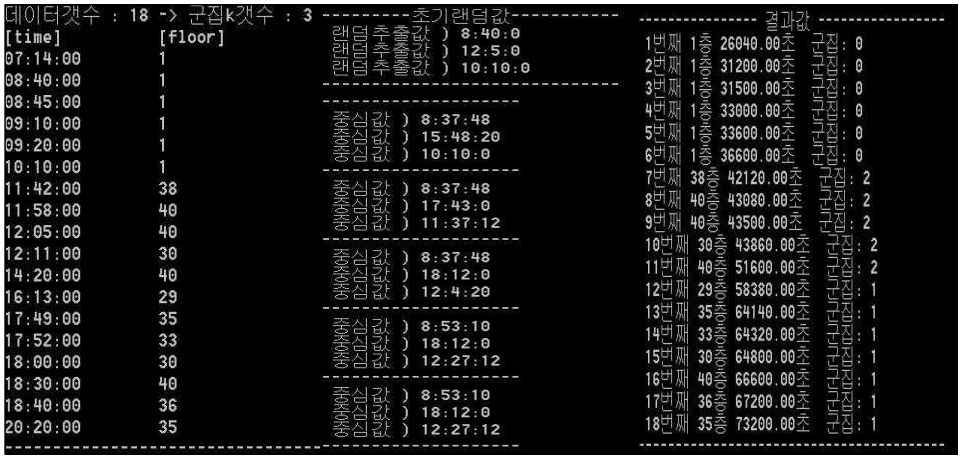
할당 해 나가는 전체 코드의 일부이다. 변수 dc 는 데이터의 개수, kc는 군집의 개수이다. 올바른 군집으로의 재 할당이 계속되다가 군집의 중심값 변화가 없을 때 반복문을 탈출한다.

```
//while 탈출키 signal
signal = 1;
while (signal){

    for (i = 0; i<kc; i++) {    //중심값을 0으로 초기화
        center[i].time = 0;
        center[i].floor = 0;
        count_Group[i] = 0;
    }

    for (i = 0; i<dc; i++) {    //각각의 모든 데이터에 대하여
        for (j = 0; j<kc; j++) {    //군집의 중심과 데이터와의 거리 계산
            tmp_distance = sqrt(pow(k[j].time - datas[i].time, 2));    //거리 계산 식
            distance[j][i] = tmp_distance;    //거리값 최신화
        }
    }

    for (i = 0; i<dc; i++) {    //각각의 모든 데이터에 대하여
        min = distance[0][i];  //하나의 데이터와 첫번째 군집과의 거리를 최소화
        min_j = 0; //데이터가 속한 군집 설정
        for (j = 1; j<kc; j++) {  //두번째 군집에 대하여
            if (min > distance[j][i]) { //첫번째 군집과 두번째 군집간의 거리 비교
                min = distance[j][i];//두번째 군집이 더 가깝다면 최소값 변경
                min_j = j;  //데이터를 두번째 군집으로 설정
            }
        }
        center[min_j].time = center[min_j].time + datas[i].time;  //시간 중심값 누적
        count_Group[min_j]++;  //군집안에 들어있는 데이터 갯수 증가
    }
```

```
    same_count = 0; //소속군집의 변화를 알기위한 변수 선언
    for (i = 0; i<kc; i++) {
        if (count_Group[i] != 0) { //군집안의 데이터의 개수가 0 이 아니면
            if (center[i].time / count_Group[i] == k[i].time)
            //현재 계산된 군집별 중심값이 기존의 군집별 중심값(K배열)과 같다면
                same_count++; //군집이 바뀌지 않았다는 것을 나타내는 변수값 증가
            k[i].time = center[i].time / count_Group[i];
        }
        if (same_count == kc) { //모든 데이터의 소속군집의 변화가 없으면
            signal = 0; //무한루프를 빠져나감
        }
        printf(" 중심값 ) ");
        char * str = chartime(k[i].time);
                        //chartime은 숫자값을 datetime 형식으로 바꿔주는 함수
        printf("%s\n", str);
    }
}// while 루프끝
```

<그림 9> 클러스터링 코드

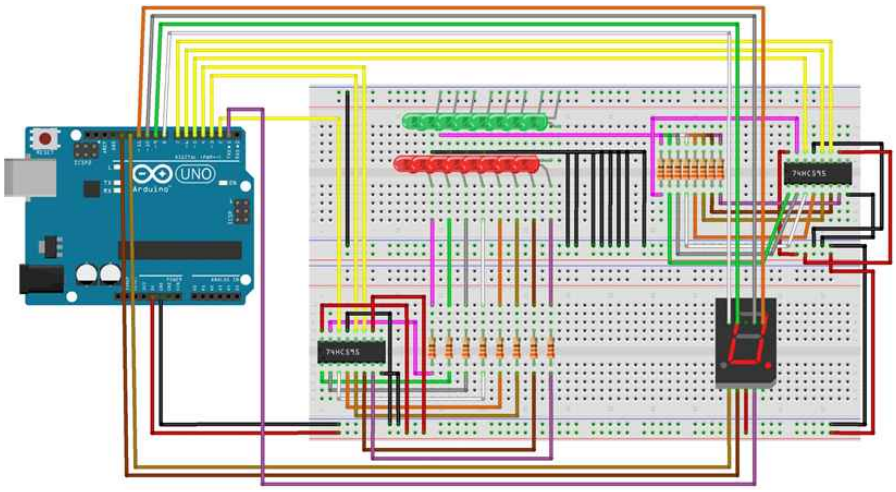
코드설명은 코드의 주석으로 대체한다. 그림 10은 구현할 알고리즘을 실행했을 경우의 결과이다. 미리 구성된 데이터베이스에서 데이터를 받아와 구조체에 저장하여 그 데이터를 시간별 군집으로 나눈다. 올바른 군집으로의 재할당이 계속 이뤄지면서, 중심값의 변화가 없으면 올바른 군집으로 구성되었다고 본다. 마지막에 데이터별로 어떤 군집에 속해있는지 소속군집과 그 군집의 평균 층수를 보여준다.



<그림 10> 클러스터링 알고리즘 구현 결과

4.3 하드웨어

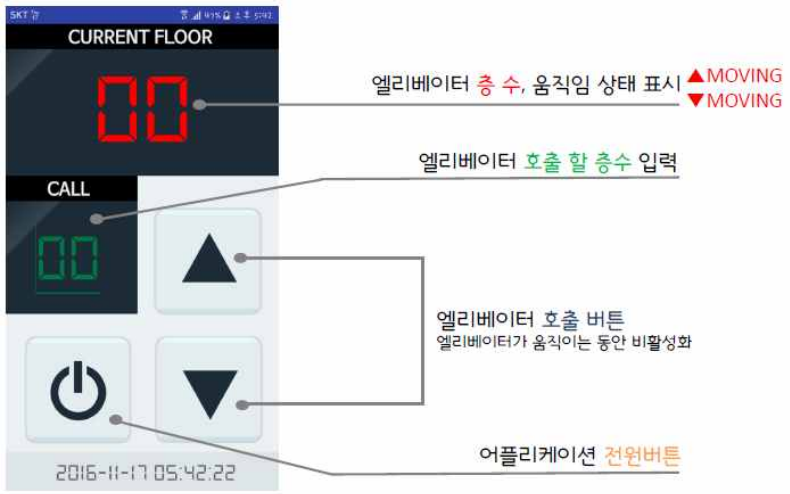
엘리베이터를 표현하기 위한 하드웨어를 설계한다. 서버와의 통신을 통해 아두이노로 LED와 7-segment를 제어한다. 아두이노의 한정된 출력개수로 16개의 LED를 사용하고 엘리베이터 층 수를 확인하기 위해 7-segment와 Shift Register를 이용한다. LED는 엘리베이터의 움직임을 표시하고, 7-segment로 엘리베이터의 층 수를 나타낸다. 7-segment 1개에 출력핀이 7개가 쓰이고, LED가 16개 출력핀을 사용하므로 Shift Register 2개를 이용하여 출력핀 3개를 이용한다. 총 10개의 출력핀으로 엘리베이터 층 수를 표시한다. 회로도는 그림 11과 같다.



<그림 11> 회로도

4.4 애플리케이션

엘리베이터 애플리케이션의 화면은 그림 12와 같다. TextView로 엘리베이터의 상태를 받아와 표시하고, EditText와 Button을 이용하여 엘리베이터 호출 층수를 입력하고 제어신호를 보낸다.



<그림 12> 애플리케이션 초기 화면과 설명

4.5 서버

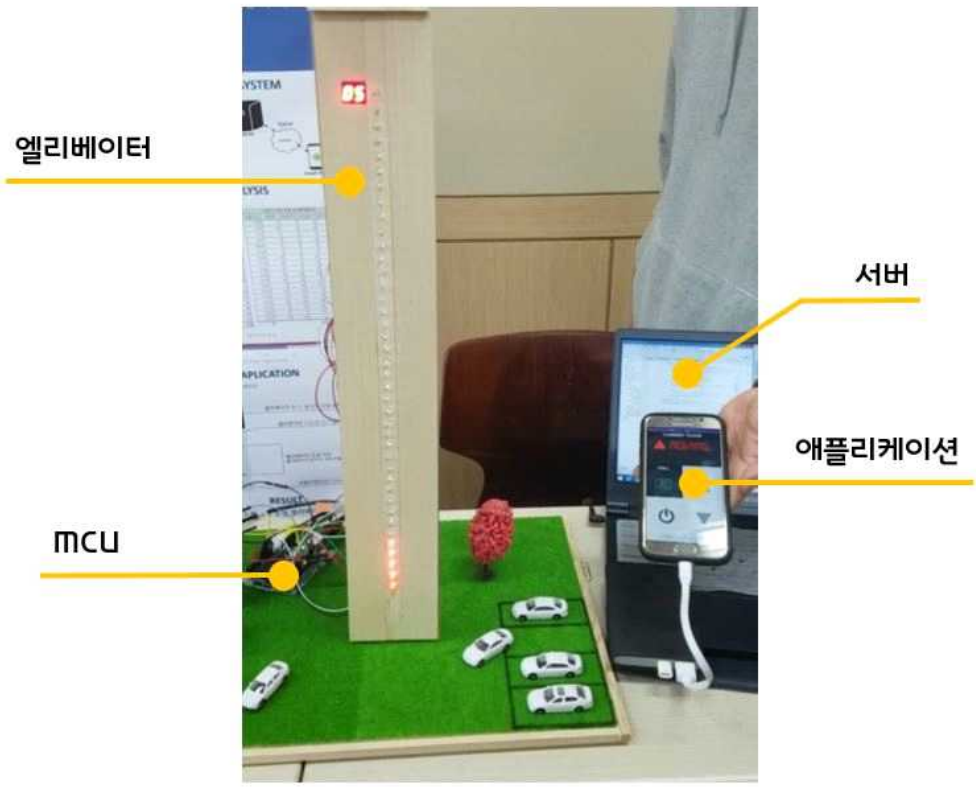
서버에는 총 2가지 프로그램이 실행된다. 하나는 C언어로 작성된 클러스터링 프로그램이고, 나머지 하나는 자바언어로 작성된 통신 프로그램이다. 클러스터링 프로그램은 저장된 데이터를 바탕으로 매주 실행되어 클러스터링 결과 테이블을 구성한다. 통신 프로그램은 클라이언트가 접속하여 아두이노의 상태를 받아오고 제어 신호를 전달하게 하고, 클러스터링 된 결과 테이블 값의 시간을 비교하여 아두이노로 알맞은 제어 신호를 보내며 클러스터링을 위해 사용자의 호출 데이터를 데이터베이스에 저장시킨다. 여러 사용자의 호출을 처리하는 부분은 엘리베이터 제어반이 담당하는 부분으로, 본 논문에서는 엘리베이터 제어반에 호출 신호를 전달하는 것까지만 구현하도록 한다. 다음 그림 13은 서버에서 클러스터링 된 결과를 받아 시각적으로 볼 수 있도록 문자열로 출력한 그림이다.

```
드라이버 검색 성공!!
My-SQL 접속 성공!!
-------------------------------------
58380~73200 -> 34층
26040~36600 -> 1층
42120~51600 -> 38층
-------------------------------------
현재 시간은 17:50:30(64230)이므로 클러스터링 된 층은 "34" 입니다.
```

<그림 13> 서버의 클러스터링 결과 호출

4.6 하드웨어 구현결과

다음 그림 14는 실제 구현한 하드웨어의 결과이다.



<그림 14> 하드웨어 구현 결과

5. 시스템 분석

5.1 분석 방법

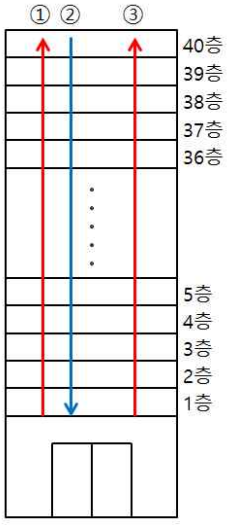
기존 엘리베이터를 사용 한 효율과 본 논문에서 제시한 k-평균 군집분석 알고리즘을 적용하여 각각 대기 시간을 예측한다. 대기 시간 계산은 다음과 같이 한다.

<표 2> 대기 시간 계산 변수

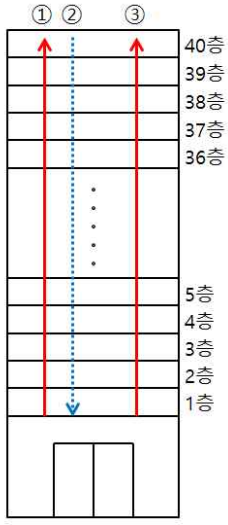
변수	설명
k	1층을 지나가는데 걸리는 시간
x	호출 목적지 층수
t	총 걸리는 시간

$$t = x * k \qquad \text{<식 3>}$$

간단한 예시로, 40층 엘리베이터가 있을 때 클러스터링한 결과가 7시 10분에 1층으로 이동하도록 설계가 되어있고, 붐비는 시간대인 7시에 사용자가 이용을 하고 그 다음사람이 7시 15분에 이용을 하였을 때의 대기시간을 예측한다. 이용은 아래 그림 15, 그림 16과 같다.



<그림 15> 기존의 엘리베이터 시스템



<그림 16> 클러스터링 알고리즘 적용 후의 엘리베이터 시스템

그림 15 기존 엘리베이터 시스템을 이용한 경우, ① 7시에 1층에서 엘리베이터 호출을 하여 40층으로 이동을 하고, ② 7시 15분에 그 다음 이용자가 1층에서 엘리베이터 호출을 하여(1층 ->40층), 40층에 있던 엘리베이터가 다시 1층으로 내려온다. ③ 엘리베이터가 40층으로 이동을 한다. 이 경우 식 3을 이용한 대기시간 예측은 다음과 같다.

<표 3> 기존 엘리베이터 시스템 대기시간 예측

k= 2초라 가정	
①	40 * 2 = 80
②	40 * 2 = 80
③	40 * 2 = 80
총 시간: 80 + 80 + 80 = 240초.	

그림 16 클러스터링 알고리즘 시스템을 이용한 경우, ① 7시에 1층에서 엘리베이터 호출을 하여 40층으로 이동을 하고, ② 7시 12분에 클러스터링을 통하여 1층으로 내려오게 된다. ③ 7시 15분에 그 다음 이용자가 1층에서 엘리베이터 호출을 하여(1층 ->40층),엘리베이터가 40층으로 이동을 한다. 이 경우 식(1)을 이용한 대기시간 예측은 다음과 같다.

<표 4> 본 논문의 엘리베이터 시스템 대기시간 예측

k= 2초라 가정	
①	40 * 2 = 80
②	0
③	40 * 2 = 80
총 시간: 80 + 80 = 160초.	

기존 엘리베이터 시스템을 이용하면 ②과정이 들어가 80초를 대기하게 되지만 엘리베이터 클러스터링을 통한 시스템을 이용하게 되면 ②과정이 7시 12분에 자동으로 1층으로 내려오게 되어있어 80초를 대기하지 않아도 된다. 이렇게 클러스터링을 한번 이용하였을 때 80초를 절약 할 수 있는데 클러스터링 K값을 높이게 되면 이와 같은 현상은 더욱 많이 이루어져 많은 시간을 절약 할 수 있을 것으로 예상된다.

5.2 데이터 분석

위와 같은 방법을 이용하여 실제 데이터에 적용해서 클러스터링을 이용한 엘리베이터 운영이 얼마만큼 대기시간을 감소시키는지 분석한다.

첫 번째로 그림 17의 데이터를 이용하여 클러스터링 전 후의 대기시간을 비교 분석한다. 이 때 엘리베이터의 초기 위치는 1층으로 가정한다.

클러스터링 적용하지 않은 엘리베이터				
호출시간	호출층	최종위치	이동층수	대기시간(초)
7:14	1	40	0	0
8:40	1	40	39	78
8:45	1	30	39	78
9:10	1	20	29	58
9:20	1	25	19	38
10:10	1	30	24	48
11:42	38	40	8	16
11:58	40	38	0	0
12:05	40	3	2	4
12:11	30	5	27	54
14:20	40	1	35	70
16:13	29	3	28	56
17:49	35	4	32	64
17:52	33	1	29	58
18:00	30	1	29	58
18:30	40	1	39	78
18:40	36	20	35	70
20:20	35	1	15	30
계			429	858

클러스터링 적용 한 엘리베이터				
호출시간	호출층	최종위치	이동층수	대기시간(초)
7:14	1	1	0	0
8:40	1	1	0	0
8:45	1	1	0	0
9:10	1	1	0	0
9:20	1	1	0	0
10:10	1	1	0	0
11:42	38	38	37	74
11:58	40	38	2	4
12:05	40	38	2	4
12:11	30	38	8	16
14:20	40	38	2	4
16:13	29	34	9	18
17:49	35	34	1	2
17:52	33	34	1	2
18:00	30	34	4	8
18:30	40	34	6	12
18:40	36	34	2	4
20:20	35	34	1	2
계			75	150

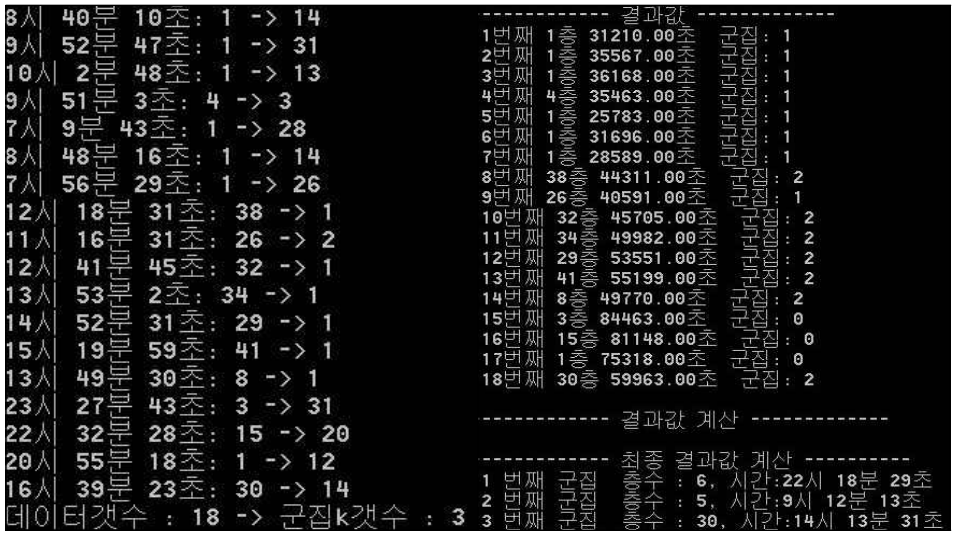
<그림 17> 그림 7에 대한 대기시간 계산

그림 17을 보면 클러스터링을 적용하지 않았을 경우 총 대기시간은 858초이며 클러스터링을 적용한 경우 총 대기시간은 150초이므로 대기시간이 약 82.5% 감소함을 알 수 있다. 하지만 이것은 동일한 데이터에 대해 클러스터링을 적용한 결과이며 본 논문에서 클러스터링을 적용할 때는 과거의 클러스터링을 미래의 데이터에 적용하는 것이므로 위의 분석은 클러스터링 자체가 대기시간을 감소할 수 있음을 확인해 주는 분석이다.

두 번째로 과거의 데이터를 클러스터링하여 미래의 데이터에 적용해 클러스터링 전 후의 대기시간을 분석한다.

이 때 데이터를 랜덤 추출하는데, 실제 엘리베이터 호출 데이터와 비슷하도록 추출할 때 가중치를 두어 추출하였다. 예를 들면 오전시간대는 출근하는 사람이 많기 때문에 1층 호출이 많은데 데이터를 랜덤 추출할 때 호출 층수가 1층이 많이 나오도록 가중치를 주었다.

이전의 분석과 마찬가지로 엘리베이터의 초기 위치는 1층으로 가정한다.



<그림 18> 첫 번째 추출한 랜덤 데이터 클러스터링

클러스터링 적용하지 않은 엘리베이터				
호출시간	호출층	최종위치	이동층수	대기시간(초)
7:13	1	20	0	0
7:45	39	37	19	38
7:54	1	17	36	72
9:50	1	4	16	32
10:01	1	19	3	6
10:26	1	23	18	36
10:38	26	14	3	6
11:34	30	1	16	32
11:38	35	1	34	68
12:45	39	1	38	76
13:26	24	1	23	46
13:55	5	1	4	8
15:04	23	1	22	44
15:54	21	1	20	40
16:22	14	35	13	26
17:25	6	38	29	58
17:35	15	1	23	46
18:13	18	35	17	34
계			334	668

클러스터링 적용 한 엘리베이터				
호출시간	호출층	최종위치	이동층수	대기시간(초)
7:13	1	5	0	0
7:45	39	5	34	68
7:54	1	5	4	8
9:50	1	5	4	8
10:01	1	5	4	8
10:26	1	5	4	8
10:38	26	5	21	42
11:34	30	5	25	50
11:38	35	1	30	60
12:45	39	30	38	76
13:26	24	30	6	12
13:55	5	30	25	50
15:04	23	30	7	14
15:54	21	30	9	18
16:22	14	30	16	32
17:25	6	30	24	48
17:35	15	1	15	30
18:13	18	35	17	34
계			283	566

<그림 19> 두 번째 추출한 랜덤데이터 대기시간 계산

클러스터링 적용하지 않은 엘리베이터				
호출시간	호출층	최종위치	이동층수	대기시간(초)
7:47	1	23	0	0
8:43	1	6	22	44
9:31	1	6	5	10
9:49	1	40	5	10
9:52	1	12	39	78
9:56	1	29	11	22
10:26	1	35	28	56
11:25	24	1	11	22
11:47	3	1	2	4
11:55	18	1	17	34
12:42	18	1	17	34
14:08	30	23	29	58
14:33	3	1	20	40
14:45	2	1	1	2
16:36	4	35	3	6
17:52	26	34	9	18
19:45	16	17	18	36
21:39	13	6	4	8
계			241	482

클러스터링 적용 한 엘리베이터				
호출시간	호출층	최종위치	이동층수	대기시간(초)
7:47	1	5	0	0
8:43	1	5	4	8
9:31	1	5	4	8
9:49	1	5	4	8
9:52	1	5	4	8
9:56	1	5	4	8
10:26	1	5	4	8
11:25	24	1	19	38
11:47	3	1	2	4
11:55	18	30	17	34
12:42	18	30	12	24
14:08	30	30	0	0
14:33	3	30	27	54
14:45	2	30	28	56
16:36	4	30	26	52
17:52	26	34	4	8
19:45	16	6	18	36
21:39	13	6	7	14
계			184	368

<그림 20> 세 번째 추출한 랜덤데이터 대기시간 계산

그림 18을 보면 첫 번째로 랜덤 데이터를 추출하여 이 데이터는 분석하지 않고 클러스터링을 한다. 그리고 이 클러스터링 데이터를 가지고 두 번째, 세 번째로 랜덤 추출한 데이터에 적용한다. 그림 19와 그림 20을 보면 각각 약 15.3%, 23.7% 대기시간이 감소함을 알 수 있다.

이를 통해 규칙성이 있는 엘리베이터 호출과 같은 자료에서 과거의 데이터를 클러스터링 하여 미래의 데이터에 적용시키면 대기시간을 감소시킬 수 있다는 결과를 얻을 수 있다.

6. 결론

본 논문에서는 보다 향상된 사용자 접근 제어와 편리성을 제공하기 위해 기존 엘리베이터 시스템을 확장하여 클러스터링을 기반으로 한 엘리베이터 운영과, 스마트 모바일 시스템과 결합한 제어 모델을 제안하였다.

엘리베이터에 MCU를 부착함으로써 스마트폰에서 엘리베이터를 원격으로 제어하고, 데이터 클러스터링을 이용해 시간 별로 엘리베이터의 최적의 위치를 찾음으로써 근본적으로 사용자의 대기시간을 감소시키고 편리성을 추구한다.

본 논문에서 구현한 엘리베이터 시스템과 기존의 방법과의 차이점은 다음과 같다. 먼저, 지금 사용하고 있는 엘리베이터는 사용자가 엘리베이터 앞에 가서 직접 버튼을 눌러야만 호출이 가능하다. 이는 사용자의 대기시간을 더 늘릴 수 있고 엘리베이터의 상태를 즉각적으로 확인 할 수 없다는 단점이 있다. 본 논문에서 구현한 스마트폰 앱을 통해 엘리베이터의 상태를 확인하고 원하는 층으로 호출이 가능함으로서 사용자의 대기시간이 감소하고 편리성이 추구된다. 엘리베이터를 이용하지 않는 사용자의 무분별한 호출 반복을 방지 하기 위해 사용시간이 특정 시간 이내일 때는 엘리베이터 호출을 할 수 없도록 제한을 추가하였다. 그리고 엘리베이터가 작동 중이지 않을 때는 이전의 데이터를 클러스터링 분석을 통해 최적의 위치를 탐색하고 그 위치로 이동한다. 과거 호출되었던 데이터를 분석하여 미리 엘리베이터가 그 위치로 가 있기 때문에 사용자는 엘리베이터 호출 후 대기 시간을 줄이고 만족감이 높아진다.

따라서, 스마트 엘리베이터 제어 시스템을 통해서 대기시간과 편리성 향상을 기대하며, 이 시스템이 다른 장치에도 적용되어 스마트한 모델이 많이 창출 될 수 있도록 기대하는 바이다.

참고문헌

[1] 김태석, 2012, “효율적인 엘리베이터 안전관리를 위한 스마트 디바이스 설계 및 구현에 관한 연구”, pp.1

[2] 전종백, 2013, “이더넷을 이용한 CAN 기반 엘리베이터 원격 모니터링 시스템 구현에 관한 연구”, pp.1

[3] 이용석 외3, “한국 산업경영시스템 학회 학술대회”, “인공지능을 이용한 엘리베이터 대기시간 최소화 시스템 개발”, 한양대학교, 산업경영공학과, pp.356

[4] 김미라, 2003, “Fuzzy C-Means 클러스터링을 이용한 웹 로그 분석 기법“, 이화여자대학교 과학기술대학원, pp.1-2

[5] 연태철, 백용남, 이후곤, 2008, "승강기기능사", 성안당

[6] 김운용, “스마트 모바일 엘리베이터 제어 시스템 구축”, 한국컴퓨터정보학회 학술대회 논문집. pp209

[7] 황재헌, 2016.2, “승강기 안전관리체계의 개선 방안에관한 연구”,서울과학기술대학교 철도전문대학원, pp6

[8] Don Choi, Hee-Chul Park and Kwang-Bang Woo, Apr. 1994, “A Study on Predictive Fuzzy Control Algorithm for Elevator Group Supervisory System,” Journal of the Korean Institute of Electrical Engineers, Vol. 43, No. 4, pp. 627~637

[9] 김진성, 임장춘, 우영운, 이임건, 박충식, 2011.1, “퍼지 추론 기법을 이용한 지능형 엘리베이터 시스템” ,한국컴퓨터정보학회 학술발표논문집 19(1), 317-320 (4 page)

[10] 박희청, 조광현, 2006, “데이터 마이닝에서의 군집화 기법 비교”, 창원대학교, 한국자료분석학회, 585-596

[11] 황진욱, 오해석, 2010, “전력소모 감소를 위한 스마트폰 홈 네트워크 관리

시스템 설계", 한국멀티미디어학회 추계학술발표대회 논문집

[12] 박주봉, 신승중, 2014, "IoT 기반 효율적 관리를 위한 엘리베이터 시스템 제안", 한국인터넷방송통신학회

[13] 이주희, 이상훈, 이지영, 2015, "라즈베리파이b+를 이용한 주차공간 관리 시스템 구현", 한국통신학회 2015년도 추계종합학술발표회, p322-324