**화재 현장상황 IOT센서 기반 화재대피 시스템 제작**

**Nonmunjemok**

**1.서론**

2019년도 구조활동 분석 결과(소방청)에 따르면 화재는 13.1%`의 높은 비율로 전체 구조 활동 중 2위를 차지했다. 또한 최근 10년간의 화재 발생 추이를 분석한 결과 매년 평균 화재 건수는 42652건, 사망과 부상을 포함한 인명피해는 2176건으로 나타났다. 이처럼 매년 수많은 사람들이 화재로 고통받고 있다. 이를 해결하기 위한 다양한 장비들이 나와있지만, 실질적인 화재 피해는 계속되고 있다.

화재발생후 재실사들의 피난시에는 재실자의 물리적 이동 속도, 그들이 이용할 수 있는 화재와 관련된 정보 그리고 화재의 발생이후 피난을 개시하기까지의 시간손실을 줄이는 것이 인명안전을 위해 중요한 역할을 한다. [1-2]

기존의 화재시스템에서 유도등으로 표시된 대피경로가 고정되어 있기에 대피자들이 화재와 관련된 정보를 충분히 얻을 수 없으며 화재를 인지한 이후 어느정도 시간이 흐른 후 대피하는 재실자들에게는 기존의 대피경로가 안전하지 않다는 문제점을 인식하였다.

이에 따라 본 연구에서는 화재로부터 안전한 최단 대피경로를 제공하여 연기 및 화재로부터 재실자들의 이동속도 저하를 최소화시키고 신속한 화재경보와 정보를 제공하여 초기 화재 시 재실자들의 대피를 유도함으로써 최대한의 대피시간을 최우선 목표로 둔다.

**2. 관련 연구 분석**

**2-1. 통신**

본 주제와 관련해 선행된 연구들을 조사한 결과, 실시간 데이터 송수신을 위해 bluetooth 모듈을 사용하였다. [3-4] 또한 Wi-Fi 대신 지그비 통신을 이용, 혹은 NodeMCU 보드 대신 아두이노 우노를 사용하기도 하였으며[5], 본 프로젝트와 같이 Wi-Fi와 NodeMCU를 함께 사용하는 경우도 상당수 존재하였다.

또한 관련 연구들의 경우 서버와 RS-232 방식으로 통신[6]하거나, 장치 제어 시스템의 ip 주소를 사용해 디바이스 행위 메시지를 해당 디바이스 조절 장치로 전송[7]하는 방식의 통신 방식을 사용하였다.

**2-2. 알고리즘**

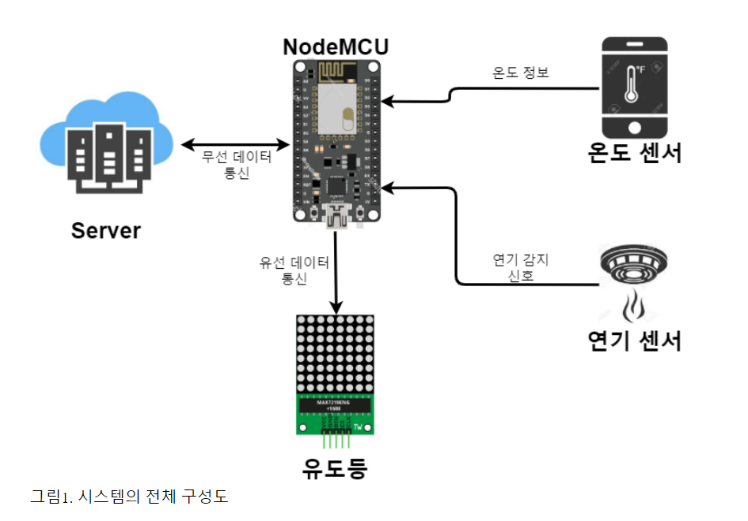
본 연구 와 비슷한 주제의 논문들의 경우 화재 시 대피경로를 탐색할 때 모든 지점에서 모든 탈출구까지의 경로(All-to-all 문제)를 구함에 Floyd-Warshall(FW) 알고리즘을 사용하였다. 이 알고리즘은 O(n^3)의 시간 복잡도를 가지므로 그리 빠르다고 볼 수 없다.[6][8]

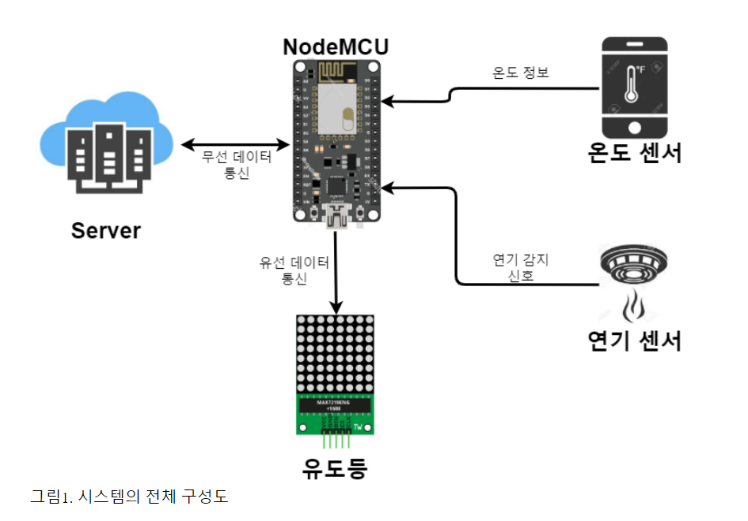
또한 화재가 발생하여 새로운 경로를 탐색할 경우 발화부 및 연기이동에 대한 신호에 대해 위험도에 따라 가중치를 부여한 후 다시 FW를 사용하거나[6] 위험도에 따라 가중치를 나누어 새로운 가중치를 부여한후 FW를 사용하였다[8]

**3. 제작한 시스템 회로 구성**

본 연구에서 제작한 화재대피시스템은 [그림1]과 같다. 화재대피시스템의 전체 구성도는 [그림2]와

같으며 총 3단계로 이루어져 있다.

[그림1]

[그림2]

**3-1. Data 획득**

수집 단계에서 사용되는 센서는 연기센서와 온도센서가 있다. 연기센서는 반도체식, 접촉연소식, SnO2 가스 센서로 나뉘는데, 본 연구에서는 응답속도와 감도가 좋은 SnO2 가스센서를 사용한다. SnO2 가스센서는 전원을 인가해주면서 전류를 계속 공급해준다. 전류로 인해 백금 열선에 열이 발생하고 공기가 흡착되게 되면서 전류의 흐름을 막고 저항값이 증가하게 된다. 이 상태에서 가스가 센서로 유입되면서 공기와 섞이게 되면, 전류 흐름이 풀어 지게 되고, 저항 값이 작아지면서 전류가 센서로 잘 흐르게 된다. 이러한 저항값 변화로 가스의 농도를 측정한다. 평상시 연기 센서의 디지털값은 High(1)를 유지하다가 연기가 검출된다면 LOW(0)로 바뀐다. 이 값은 MCU로 전달된다.

온도 센서는 접촉식, 비접촉식 온도센서로 나뉜다. 본 연구에서는 접촉식 온도센서인 RTD 온도센서를 이용한다. RTD 센서는 온도가 변함에 따라 저항 수치가 변하는 저항을 포함하는 온도 센서이다. 다양한 온도에서의 저항수치 데이터가 정리되어 있는 순물질로 만들어져 물질의 온도가 변함에 따라 예측 가능한 변화를 보이기에 수치에 대한 데이터로 온도를 확인할 수 있다. 실시간으로 온도 센서의 디지털 값을 MCU로 전달한다.

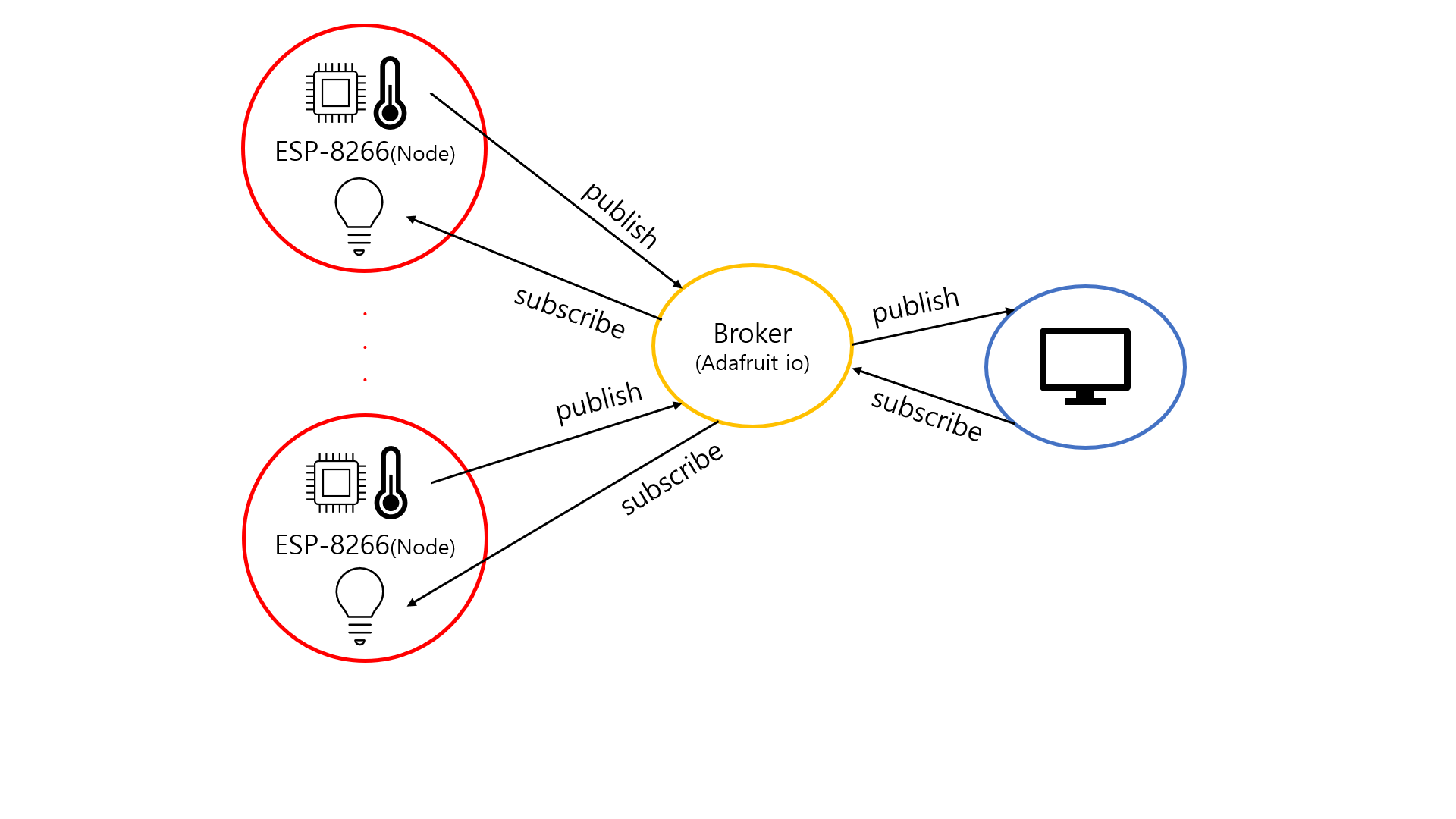
**3-2. 시스템 Data 처리**

**MQTT 프로토콜**

Eclipse mosquitto에 따르면, MQTT는 publish/subscribe 모델을 사용한 경량화 프로토콜로, 제한된 네트워크 환경에서도 효율적인 메시지 전달을 할 수 있다. 국제 표준 경량화 프로토콜으로써 MQTT는 IoT 메시징에 적합하다는 평을 받고 있으며, 실제로 저전력 센서나 모바일 기기 등에 유용하게 사용되고 있다. [9-10]

이 publish/subscribe 모델을 기반으로, 각 MQTT client들은 broker에게 메시지를 전달 혹은 수신하는 간단한 방식으로 양방향 통신 수행할 수 있다.

MQTT 클라이언트들의 중재 역할을 하는 MQTT 브로커의 종류는 mosquitto, HiveMQ, RabbitMQ 등 다양하다. 본 연구에서는 python과 esp8266을 지원하는 Adafruit io 플랫폼을 채택하였다.



**Nodemcu**

NodeMCU는 오픈 소스 IoT 플랫폼으로 와이파이 무선 통신이 지원되는 MCU 개발 보드이다. 본연구에서는 NodeMCU v1.0을 기본 보드로 선택하여 ESP8266 와이파이 모듈을 이용해 네트워크 기능을 구현하였다. NodeMCU는 아두이노 IDE 개발환경에서 프로그래밍을 하여 코드를 업로드 할 수 있다. 업로드 후, Power source와 연결돼 있다면 PC와의 직접적인 연결 없이 무선으로 센서의 데이터를 보내는 것이 가능하다. 본 프로젝트에서는 NodeMCU하나 당 한 노드의 가스센서, 열센서, dot matrix 각각을 연결해 자원을 최대한 절약하였다. 각 NodeMCU에서 획득한 데이터를 MQTT 브로커인 adafruit io로 publish 혹은 susbcribe해 추가적인 작업이 수행될 수 있도록 하였다.

**Adafruit io**

Adafruit io는 인터넷에 연결된 여러 센서들의 값을 저장, 보관, 추적하고 요청에 따라 사용할 수 있도록 돕는 플랫폼이다. Mqtt용 서버와 데이터 베이스 기능을 제공하기 때문에 사용자가 이를 직접 구축할 필요가 없다. 센서에서 획득한 데이터는 미리 설정해 둔 feed에 시간 별로 저장된다. Adafruit IO에서는 feed를 무제한으로 제공하여 데이터의 저장을 용이하게 한다. 모든 feed는 고유의 id를 가지고 있기 때문에 id를 활용해 원하는 노드와 1대1 연결하는 것이 가능하다.

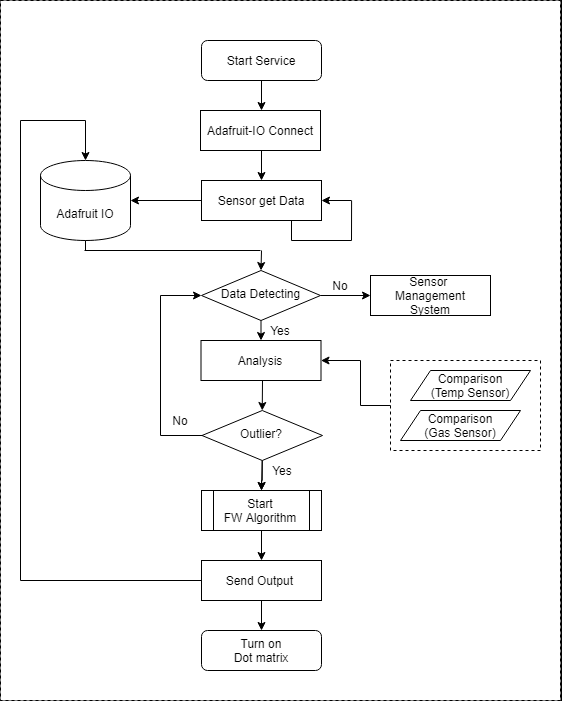
Feed의 데이터를 시각화하기 위해 Adafruit에서 제공하는 dash board를 사용할 수 있다. Dash board에 존재하는 widget을 이용해 feed의 데이터를 차트, 슬라이더, 버튼과 같은 다양한 형태로 나타낼 수 있다. 그리고 언제 어디서든 web을 이용해 데이터 저장현황을 확인할 수 있다.

본 연구에서 MQTT 플랫폼으로 Adafruit io를 채택하게 된 이유로, Adafruit io의 가장 큰 강점은 자체 라이브러리를 지원하는 것이다. 라이브러리에서 제공하는 함수를 이용해 feed의 데이터를 손쉽게 저장, 삭제 및 관리하는 것이 가능하다. 해당 라이브러리는 esp8266과 python 모두를 지원하기 때문에 사용자가 직관적으로 코드를 설계하는 것을 돕는다.

**동작원리**

건물 내부에 단위 면적 당 가스센서, 열 센서, dot matrix를 각각 설치한다. MQTT에서 클라이언트들에 해당하는 열 센서와 가스 센서는 주변 환경을 지속적으로 감지하며 감지된 데이터를 MQTT 브로커인 Adafruit IO로 publish해 해당하는 feed에 값을 저장한다. 이 때 알고리즘과 시스템을 실행시키는 서버 또한 MQTT의 클라이언트로 기능하게 된다. 가스 센서, 열 센서, dot matrix에 대한 feed는 각 노드 별로 하나씩 연결되어, 실제로 모든 센서 수만큼의 feed가 존재하게 된다.

이 서버에서는 모든 feed의 최근 데이터 값을 지속적으로 검사한다. 관리자가 설정한 경고 범위의 값과 비교해 이상치가 발생하게 된다면 해당 노드 정보를 알고리즘에 전달한다. 노드 정보를 받은 알고리즘을 이용하여 각 노드에서 갈 수 있는 방향 정보를 얻어온 후, 이 정보를 알려주기 위해 다시 해당 노드들의 dot matrix feed에 정보를 전송한다.



각 NodeMCU 보드에는 아두이노 IDE를 통해 작성한 코드를 업로드한다. Wi-Fi 연결이 정상적으로 완료되면, Adafruit io 라이브러리를 사용하여 Smoke feed, Fire feed, LED feed에 해당하는 객체를 생성하고 MQTT 브로커와 연결한다. 이 때 LED 센서는 subscribe 역할을 하여 LED feed와 연결된 객체에 어떠한 값이 들어오면 함수를 실행해 값을 확인하고 dot matrix에 해당하는 방향을 표시한다. 온도, 연기 감지 센서는 publish 역할을 하여 실시간으로 온도 및 가스를 감지해 일정 주기마다 해당하는 Fire feed, Smoke feed에 값을 보내준다. 관리자가 정한 임계값에 따라 이상 수치가 나오면 0을 전송, 정상 수치가 나오면 1을 전송하는 방식으로 이루어지며, 센서가 고장이 나는 경우도 0을 전송하게 되어 혼란을 방지하고자 하였다. 코드가 업로드 된 보드는 power만 공급해주면 pc와 연결되지 않아도 작동할 수 있다. 이와 같은 방법으로 단위 면적당 배치된 노드(NodeMCU)들이 센서 값들을 감지해 실시간으로 데이터를 송수신하게 된다.

서버에서는 Adafruit io 라이브러리를 이용하여 client 객체를 생성한 후, 각각의 feed와 연결한다. 반복적으로 모든 노드들의 Smoke feed, Fire feed 값을 subscribe하며 대기하다가, 이상치 발견시 해당 노드 번호를 알고리즘에게 제공하여 알고리즘을 실행시킨다. 알고리즘 수행 후, 방향 지시의 변화가 필요한 노드에게 방향을 알려주기 위해 해당하는 노드의 LED feed에게 데이터를 publish한다. 이와 같은 방식을 이용하면, 각 보드와 서버에서는 불필요한 코드를 없애 보다 간결고 가독성 좋게 통신을 수행할 수 있다

**3-3. 대피유도등 표시제어**

기존의 비상 유도등의 문제점은 비상구 표지판의 픽토그램의 방향이 한정적이라는 것이다. 본 연구의 비상 유도등은 최적의 경로로 비상구를 향해 탈출하는 길을 상하좌우로 표현하여 재실자(대피자)에게 보다 안전한 탈출을 유도할 수 있다. 비상 유도등으로는 Pin 수를 적게 필요로 하는 MAX7219 Chip을 활용한 8X8 LED Matrix를 사용하였다.

**4. 제안한 알고리즘**

**4-1. 화재 발생 전 유도등 제어 알고리즘**

대피경로 탐색을 위한 All-to-some문제해결을 위해 All 대비 Some의 개수가 적을 경우 All-to-all FW 알고리즘보다도 One-to-all Djikstra 알고리즘을 반복하여 사용하는 것이 훨씬 효과적이다. [11]

따라서 본 연구에서는 One-to-all Djikstra 알고리즘을 이용하여 One을 하나의 출구, all을 모든 노드로 두고 모든 출구들에 대해 출구 개수(M)번 알고리즘을 반복하여[그림3] 각 지점별 출구까지의 최소 거리(비용)를 노드의 distance로 저장한후 각 노드와 연결된 노드들 중 가장 작은 distance를 가지는 노드로 향하도록 유도등을 제어하였다. 여기서 소방법상 화재 감지센서 및 유도등은 20m를 간격으로 설치되어야 하기에[12] 노드간 edge의 weight는 1로 모두 같게 설정하였다.

N = 노드 개수, M = 출구 개수 (N>>M)

For( i=0; i<N; i++){

For( j=0; j<M; j++){

For( k=0; k<M; k++){

}

}

}

[그림3]

**4-2. 화재 발생 후 유도등 제어 알고리즘**

화재발생시 대피자들의 심리상 화재 유도등이 탈출 인구 밀도에 의해 변화되었다고 해서 유도등을 따르는 것이 아닌 대다수의 사람들이 이동하는 방향을 선택하기에 밀집도에 따른 유도 등 변화는 효과가 없다고 판단하였다.

또한 화재 발생 후 모든 구역의 유도등 방향이 바뀌는 것이 아니기 때문에 최적 경로를 위한 all to some 알고리즘을 반복 수행하는 것은 비효율적이다. 이 문제점 해결을 위해 본 연구에서는 불이 난 뒤 유도등 방향이 바뀔 필요가 있는 노드만 탐색하여 유도등 방향을 수정하는 알고리즘을 고안하여 사용하였다. [그림4]은 본 연구에서 사용된 알고리즘의 Pseudocode이다.

Q : Nodes to Explore

Fire search(){

if cur node's adjacent node is not prev node

Q <- adjacent node

M <- min(adjacent node distance - cur node distance)

if cur direction is not prev direction

Mark in M direction

else

Q -> Delete cur node

if Q is not empty

Fire search()

}

[그림4]

**5. 기대효과**

본 연구에서 개발한 시스템은 MQTT를 통해 연기와 화재 감지기에서 실시간으로 수집한 정보를 받아 탈출 방향을 안내하는 시스템이다. 기존 시스템과는 달리, 무선 IoT 기반 시스템을 도입함으로써 배선 공사와 고비용의 문제들을 해결하고, 보다 간결한 방식으로 수집된 센서 데이터들을 전송할 수 있을 것이라 판단한다.

또한, 본 알고리즘은 탈출구까지의 최단 거리만을 고려하는 기존의 고정된 비상등 경로와는 달리, 최단 거리와 위험도를 함께 고려해 동적으로 경로를 도출해낸다. 빠르게 변화하는 화재 상황에 맞추어 효과적인 탈출 경로를 기대할 수 있다. 따라서 탈출 시 발생하는 불과 연기가 퍼져 이동하게 되더라도 갇힌 사람에게 탈출 방향을 실시간으로 제시함으로써 이에 의한 인명 피해를 최소화할 수 있을 것으로 기대한다.

**6. 결론**

**참고문헌**

[1] 최민지 외 3, “화재 시 재실자 행동의 상호 작용을 고려한 건물 피난 행태분석”, 한국건설관리학회 논문집, Vol.14 No.6 pp 49-61, 2013,11

[2] 박재성, “건축물 화재 시 피난개시 과정의 주요 특성 및 요소에 관한 연구 – 일본 등 해외 화재사례로-“, 한국화재소방학회 논문지 Vol.26 No.2 통권 93호 pp. 59-68, 2012

[3] 성현진, ‘시설물 화재시 실시간 모니터링을 위한 u-방재프로세스 개발’, 부산대학교 대학원, 2011

[4] 한정철, ‘블루투스 저에너지 기기를 활용한 실내 재난 대피 시스템의 설계 및 구현’, 고려대학교 대학원, 2017]

[5] 최문기, ‘IoT센서의 공간 빅데이터를 활용한 실내 대피 경로 제공 모델 연구’, 경희대학교 대학원, 2018

[6] 김동욱 외 6, “유도등 제어시스템의 개발” 조명⬝전기설비학회 논문지, 제23권, 제6호, PP.. 52-58, 2009,6

[7] 김용우 외 3, “무선센서 네트워크 기반의 화재 대피유도연구”

[8] 이기연 외, “최적 비상대피로 유도를 위한 방향성 유도 알고리즘”, 조명⬝전기설비학회 논문지, 제23권, 제10호, PP.. 115-120, 2009,6

[9] Raphael Cohn and Richard Coppen. OASIS Message Queuing Telemetry Transport (MQTT) TC[Internet]. Available: https://www.oasis-open.org/committees/tc\_home.php?wgab brev =mqtt

[10] M. Prihodko, “Energy Consumption in Location Sharing Protocols for Android Applications,” Master's thesis, Linkoopings Univers

ity, Sweden, 2012.

[11] 김병인 외 1, “도로 네트워크에서의 Some-to-some 최단경로 생성 알고리즘 성능비교”, 대한산업공학회 춘계 학술대회논문집, pp 1-6, 2006

[12] “유도등 및 유도표지의 화재안전기준(NFSC 303)[시행 2017.7.26.] [소방청고시 제2017-1호, 2017. 7. 26., 타법개정]