보안 과제( ), 일반 과제(○) / 공개(○), 비공개( )

국토교통기술촉진연구사업 제3차 연도 최종보고서

R&D / 1615007200

M2M 기반 지하공간(지하철)재난 대응 대화형 스마트 네트워크 시스템 개발 최종보고서

2017.10.20.

주관연구기관 / (주)승화기술정책연구소

공동연구기관 / 엠랩(주)

위탁연구기관 / 한국철도공사

국 토 교 통 부

국토교통과학기술진흥원

Infrastructure R&D Report

161500
7200

M2M 기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템 개발 최종보고서

2017

국토교통과학기술진흥원

국토교통부

제 출 문

국토교통과학기술진흥원장 귀하

'M2M 기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템 개발'(연구개발 기간 : 2014. 07. ~ 2017. 07.) 과제의 최종보고서를 제출합니다.

2017. 10. 20.

주관연구기관명 : (주)승화기술정책연구소 (대표자) 박 미 연 (인)

공동연구기관명 : 엠랩(주) (대표자) 홍 의 재 (인)

위탁연구기관명 : 한국철도공사 (사장직무대리) 유 재 영 (인)

주관연구기관책임자: 박 미 연
공동연구기관책임자: 고 성 기
위탁연구기관책임자: 권 세 곤

국토교통부소관 연구개발사업 운영규정 제38조에 따라 최종보고서 열람에 동의합니다.

보고서 요약서

<table>
<tr><td>과제 고유 번호</td><td>1615007200</td><td>해당 단계
연구 기간</td><td>2014.07.18.
~ 2017.07.17.</td><td>단계구분</td><td>3/3
(해당단계)/(총단계)</td></tr>
<tr><td rowspan="2">연구사업명</td><td>중사업명</td><td colspan="4">국토교통기술촉진연구사업</td></tr>
<tr><td>세부사업명</td><td colspan="4">기초원천</td></tr>
<tr><td rowspan="2">연구과제명</td><td>대과제명</td><td colspan="4">M2M 기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템 개발</td></tr>
<tr><td>세부과제명</td><td colspan="4">재난 대응 SOP 개발 및 M2M 재난 전조 대응 알고리즘 개발
스마트 N키퍼 및 재난관리 시스템 개발</td></tr>
<tr><td rowspan="2">연구책임자</td><td rowspan="2">박미연</td><td>해당단계
참여
연구원 수</td><td>총: 명
내부: 명
외부: 명</td><td>해당단계
연구개발비</td><td>정부:300,000천원
기업:100,000천원
정부 외: 천원
계:400,000천원</td></tr>
<tr><td>총 연구기간
참여
연구원 수</td><td>총: 명
내부: 명
외부: 명</td><td>총 연구개발비</td><td>정부:900,000천원
기업:300,000천원
정부 외: 천원
계:1,200,000천원</td></tr>
<tr><td>연구기관명 및
소속 부서명</td><td colspan="3">(주)승화기술정책연구소 기술연구소</td><td colspan="2">참여기업명 : 엠랩(주)</td></tr>
<tr><td>국제공동연구</td><td colspan="3">상대국명:</td><td colspan="2">상대국 연구기관명:</td></tr>
<tr><td>위탁연구</td><td colspan="3">연구기관명: 한국철도공사</td><td colspan="2">연구책임자: 권세곤</td></tr>
<tr><td colspan="4">지하공간(지하철)에서 발생 가능한 재난에 대응하기 위한 M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크(IoT) 시스템 개발을 통해 해당 시설물에서 발생하는 재난을 센서를 통해 감지하고, 승객 대피 의사결정 등 재난 대응 중심의 선제적 재난관리체계 구축 및 신속한 의사결정 지원 기술을 제시</td><td colspan="2">294 페이지</td></tr>
</table>

4. 국문 요약문

<table>
<tr><td>연구의
목적 및 내용</td><td>○ M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크(IoT) 시스템은 기존에 재난 발생 후 통신이 정지되면 마비되는 재난대응시스템의 단점을 개선하기 위해서 현장에서 센서를 관리하는 기기(스마트N키퍼) 간 네트워크 통신이 가능하도록 저전력 장거리 무선통신 모듈을 사용하여 중앙시스템(서버)이 마비되어도 각각의 스마트N키퍼 스스로 현장에서 로컬정보를 바탕으로 상황판단 및 재난대응을 할 수 있도록 대피유도 및 상황알림 기능을 자율적으로 수행할 수 있는 지능형 네트워크 시스템이다.
○ 지하시설(지하철) 재난 대응 프로세스(SOP) 개발
- 승객 정보 제공용 비주얼 E-book 제작
- 국내외 방재전문가 참여 SOP 개발 진행
○ SOP기반 재난 대응 프로세스 및 재난전조대응 M2M 알고리즘 개발
- 방재스마트N키퍼 상황인지 및 의사결정 알고리즘 개발
- 방재스마트N키퍼 상황인지 정보 공유모델 개발 및 구현
○ 스마트N키퍼 개발
- 재난 대응형 저전력 무선 센서노드 개발
- M2M기반 스마트 네트워크 관리 시스템 구성
- 재난상황 발생 이후 신속한 SOP 전개기능 개발
- 지하철 승객 피난경로 분석 및 대피 유도 및 안내 기능 개발
- 센서 정보 수집 및 효율적인 포워딩 기능 개발
○ 스마트N키퍼 테스트베드 운영
- 지하3층 이상 규모의 지하철역 선정 테스트베드 운영
- 재난 대응 대화형 웹기반 스마트 네트워크 관리시스템 개발
- 테스트베드 구축 및 현장실험</td></tr>
</table>

연구개발성과	○ 지하공간(지하철) 비상대응 매뉴얼 ○ 해외 지하공간(지하철) 비상대응 연구 사례분석 ○ 재난 전조예측을 위한 시계열 데이터분석 알고리즘 개발 ○ 재난 전조 대응 스마트N키퍼 모듈 개발 ○ 재난 전조예측 모듈 α버전 개발 ○ 재난 대응 M2M기반 SOP 모듈 β버전 개발 ○ 대피 경로 분석/대피 유도 모듈 β버전 개발 ○ 재난전조 대응 M2M 네트워크의 구축 ○ 재난 전조 대응 M2M 웹 어플리케이션 버전 1.0 개발 ○ 재난 대응 M2M 센서노드 개발 ○ 웹기반 재난 관리 시스템 버전 1.0 개발 ○ 대피 경로 안내 보조 디스플레이 모듈 개발 ○ 테스트베드 구축 및 현장시험
연구개발성과의 활용계획 (기대효과)	○ 연구를 통해 개발되는 성과물을 위탁기관으로 참여하는 철도공사에 시험 적용한 후 꾸준한 유지보수를 진행하여, 사업화 가능 수준의 제품 제시 및 사업화 과제를 통해 생산능력 및 구매처를 확보 ○ 우리나라의 경우 지하공간의 활용빈도가 어느 나라보다 높은 편이며 상대적으로 재난에 대한 위험 수준 또한 매우 높은 수준에 직면하고 있어 본 기술의 개발과 보급을 통해 재난에 대한 예방대책과 안전관리에 매우 큰 기여를 할 것으로 판단됨

핵심어 (5개 이내)	지하철시설물	재난전조예측	M2M기반	대화형 네트워크	스마트 센서

〈 SUMMARY 〉

Purpose & Contents	○ As an immediate response to emergencies in underground subway system, M2M underground rescue system will be developed based on the Internet Of Things (IoT) technology. This M2M underground rescue system can collect surrounding information through sensors and prevent undesirable damages to passengers as well as infrastructures by making prompt decision and evacuation suggestions prior to emergencies ○ **SOP development for emergencies of underground subway system** - Producing E-book for passengers about emergency manuals - Inviting safety experts in the world to develop SOP ○ **M2M algorithm development based on SOP emergency process** - Algorithm development for surrounding recognition and decision making process of safety SmartNKeeper - Development of safety SmartNKeeper in surrounding recognition information and data sharing model ○ **SmartNKeeper development** - improvement on remote sensor node feature - Network Development for smart network management system - Development of effective SOP delivering system during emergencies - Analysis on current emergency routes and development of effective evacuation system for passengers - Development of sensor information collection and forwarding methods ○ **Expending application of SmartNKeeper testbed** - Testbed operation on underground system with more than three basement levels - Development of α version for smart network communication system
Results	
Expected Contribution	○ The research product can be tested and applied to the KORAIL's system. With the constant maintenance and repairs, the quality of the product can be assessed. Through the successful test application at KORAIL, potential commercialization of the product can be proposed. Through this project, it would also be possible to collect information of potential customers in advance. ○ In Korea, usage of underground areas is relatively high, comparing to other countries. Therefore, the nation recognizes the importance of adequate safety management system for underground areas. This project will largely contribute to make a proper and effective emergency prediction and preparation program.

Keywords	Subway Structure	Disaster Sign	M2M Based	Conversation Network	Smart Sensor

〈 CONTENTS 〉

1. Outline of Research and Development Project ····· 1
 1. R & D Objectives ····· 1
 2. Necessity of research and development ····· 1
 3. R & D Scope ····· 1
2. Status of domestic and overseas technology development ····· 3
 1. Research trends in related fields at home and abroad ····· 3
 2. Patent Trends Related to Domestic and Foreign Technologies ····· 9
 3. Market Analysis and Industry Trends ····· 9
3. Research performance and achievements ····· 13
 1. Underground space (subway) emergency response manual ····· 13
 2. Case Study on Emergency Response to Overseas Underground Space (Subway) ····· 48
 3. Development of Time Series Data Analysis Algorithm for Predictive Disaster Prediction ····· 68
 4. Smart N Keeper Module for Disaster Prevention Response ····· 69
 5. Development of a Disaster Prevention / Disaster Response Module Alpha Version 72
 6. Development of M2M-based SOP module β version ····· 90
 7. Evacuation path analysis / evacuation module β development ····· 92
 8. Establishment of M2M network for disaster response ····· 102
 9. Web-based M2M management system development ····· 107
 10. Developing a sensor node ····· 117
 11. Development of M2M Web Application for Disaster Response ····· 162
 12. Development of auxiliary display module ····· 228
 13. Build a test bed and test the field ····· 253
4. Achievement goal and contribution to related field ····· 287
 1. Goal Achievement ····· 287
 2. Contribution to related fields ····· 290
5. Plan to utilize R&D achievements ····· 291
 1. Connection with Existing Fire and Disaster Systems ····· 291
 2. Railway construction disaster area drop rock detection system ····· 291
 3. Railway Crew Passenger Scoring System ····· 292
 4. Line operator alarm system (Korea Railroad Corporation) ····· 293
6. Overseas science and technology information collected during the research process 294
7. Security rating of R&D achievements ····· 295
8. Research facilities and equipment registered in the National Science and Technology Comprehensive Information System ····· 295
9. Performance of safety measures in laboratories according to performance of R&D tasks ····· 295
10. Representative Research Results of R&D Projects ····· 295

〈 목 차 〉

1. 연구개발과제의 개요 ····· 1
 1. 연구개발 목적 ····· 1
 2. 연구개발의 필요성 ····· 1
 2. 연구개발 범위 ····· 1
2. 국내외 기술 개발 현황 ····· 3
 1. 관련 분야의 국내외 연구동향 ····· 3
 2. 국내외 관련 기술 특허 동향 ····· 9
 3. 국내외 시장분석 및 산업동향 ····· 9
3. 연구 수행 내용 및 성과 ····· 13
 1. 지하공간(지하철) 비상대응 매뉴얼 ····· 13
 2. 해외 지하공간(지하철) 비상대응 연구 사례분석 ····· 48
 3. 재난 전조예측을 위한 시계열 데이터분석 알고리즘 개발 ····· 68
 4. 재난 전조 대응 스마트N키퍼 모듈 ····· 69
 5. 재난 전조예측/재난대응 모듈 알파버전 개발 ····· 72
 6. M2M기반 SOP 모듈 β버전 개발 ····· 90
 7. 대피 경로 분석/대피 유도 모듈 β버전 개발 9 ····· 2
 8. 재난전조 대응 M2M 네트워크의 구축 ····· 102
 9. 웹기반 M2M 관리 시스템 개발 ····· 107
 10. 센서노드 개발 ····· 117
 11. 재난 전조 대응 M2M 웹 어플리케이션 개발 ····· 162
 12. 보조 디스플레이 모듈 개발 ····· 228
 13. 테스트베드 구축 및 현장시험 ····· 253
4. 목표 달성도 및 관련 분야 기여도 ····· 287
 1. 목표 달성도 ····· 287
 2. 관련 분야 기여도 ····· 290
5. 연구개발성과의 활용 계획 ····· 289
 1. 기존 화재 및 재난 시스템과 연계 방안 ····· 291
 2. 철도공사 재해우려개소 낙석감지 시스템 ····· 291
 3. 철도공사 승무원 승객검표 시스템 ····· 292
 4. 선로작업자 열차접근경보 시스템 (한국철도공사) ····· 293
6. 연구 과정에서 수집한 해외 과학기술 정보 ····· 294
7. 연구개발성과의 보안등급 ····· 295
8. 국가과학기술종합정보시스템에 등록한 연구시설 · 장비 현황 ····· 296
9. 연구개발과제 수행에 따른 연구실 등의 안전 조치 이행 실적 ····· 296
10. 연구개발과제의 대표적 연구 실적 ····· 296

제1장. 연구개발과제의 개요

1. 연구개발 목적

○ 지하공간(지하철)에서 발생 가능한 재난에 대응하기 위한 M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크(IoT) 시스템 개발을 통해 해당 시설물에서 발생하는 재난을 센서를 통해 감지하고, 승객 대피 의사결정 등 재난 대응 중심의 선제적 재난관리체계 구축 및 신속한 의사결정 지원 기술을 제시

○ M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크(IoT) 시스템은 기존에 재난 발생 후 통신이 정지되면 마비되는 재난대응시스템의 단점을 개선하기 위해서 현장에서 센서를 관리하는 기기(스마트N키퍼) 간 네트워크 통신이 가능하도록 저전력 장거리 무선통신 모듈을 사용하여 중앙시스템(서버)이 마비되어도 각각의 스마트N키퍼 스스로 현장에서 로컬정보를 바탕으로 상황판단 및 재난대응을 할 수 있도록 대피유도 및 상황알림 기능을 자율적으로 수행할 수 있는 지능형 네트워크 시스템 개발

2. 연구개발의 필요성

○ 국내 재난 발생의 형태는 기후변화 및 도시화의 급속한 진행으로 인해, 예측이 불가능한 돌발적 재난의 발생이 증가하고 있으며, 일단 발생하면 상당한 규모의 인적·재산 피해가 발생하고 있다.

○ 또한, 여러 유형의 재난이 동시 다발적으로 발생하는 복합재난의 성격을 띠고 있어 상황 발생 시 이에 대한 대응 체계 구축이 매우 중요하게 대두되고 있다.

○ 하지만, 국내에서 연구를 통해 제시되고 있는 연구 성과들은 관련 센서(온도, 변위등)를 통해 재난을 감지하고 이를 중앙관제센터에서 통제하기 위한 실시간 모니터링 시스템에 국한되어 있어 감지된 재난에 대해 극한 상황 속에서도 인명을 보호하고 재난에 대응할 수 있는 최적의 재난 대응 시스템을 도입할 필요가 있다.

○ 피난대피 알고리즘 개발은 그 적용 대상이 건물과 같은 다양한 구조를 가진 유클리드 평면 혹은 유클리드 평면의 중복된 복층을 대상으로 하는데 반해서 지하철은 대피경로가 거의 선형에 가까우며, 대피 방향이 명확한 네트워크 구조에 가까운 특성을 이용해서 지하철 역사의 경우, 선형 네트워크에서 피난 경로를 선택하는 방향으로 적용을 해서 일반적인 유클리드 평면에서의 피난경로 시뮬레이션이 가지는 계산량 및 계산시간을 줄여서 실시간에 가까운 피난 유도 기능을 강화하는 쪽으로 구분해서 연구를 진행할 필요가 있다.

3. 연구개발 범위

연차	연구목표	주요연구내용
1단계	A.지하시설(지하철) 재난	①지하시설(지하철)에서 발생 가능한 재난 유형별 발

(‘14.07~’15.07)	대응 프로세스(SOP) 기초 연구	생 사례 및 위험도 영향도 분석 ②재난 유발매체와 연계된 데이터 확보 및 기술현황 분석 ③국내·외 지하시설 재난 발생 시 대응 행동 매뉴얼 및 방재 기술 개발 현황조사
	B.SOP기반 재난 대응 프로세스 및 재난전조대응 M2M 알고리즘 개발 기초 연구	①기계학습 이론 및 데이터 마이닝 이론 분야조사를 통한 적용 가능한 모델 구축 이론 연구 ②리스크 매니지먼트 방법론을 통한 change detection 기법에 관한 연구
	C.스마트N키퍼 프로토타입 개발	①스마트N키퍼의 센서, 허브, 스마트패드의 하드웨어 설계 및 프로토타입 개발 ②스마트N키퍼 운용 소프트웨어 설계 및 프로토타입 개발
2단계 (‘15.07~’16.07)	A.지하시설(지하철) 재난 대응 프로세스(SOP) 개발	①재난 유형별 발생 대응 전략 수립 ②대응 전략별 적정성 평가(전문가 및 실무자 접견) ③재난발생시 대응 행동 매뉴얼 및 방재기술 개발 현황에 대한 개선사항 연구 및 도출
	B.SOP기반 재난 대응 프로세스 및 재난전조대응 M2M 알고리즘 개발	①방재스마트N키퍼 상황인지 및 의사결정 로직 및 알고리즘 개발/구현 ②방재스마트N키퍼 상황인지 정보 공유 모델 개발 및 구현
	C.스마트N키퍼 개발	①스마트 네트워크 관리 시스템과의 통신 기능 개발 ②재난상황 발생 이후 SOP 전개 기능 개발 ③피난/대피 유도 기능 개발 ④센서 정보 수집 및 포워딩 기능 개발
3단계 (‘16.07~’17.07)	A.지하시설(지하철) 재난 대응 프로세스(SOP) 고도화	①테스트베드 운용을 통한 SOP 보완 및 매뉴얼제시
	B.지하시설(지하철) 재난 대응 알고리즘 고도화	①테스트베드 운용 및 시뮬레이션 결과를 통한 M2M 기반 대응 알고리즘의 효율성 개선
	C.스마트N키퍼 고도화	①스마트N키퍼 H/W 테스트베드 운용을 통한 사용자 편의성 및 성능개선 ②스마트N키퍼 S/W 테스트베드 운용을 통한 사용자 편의성 및 성능개선
	D.재난대응 대화형 스마트 네트워크 관리 시스	①위치 기반 상황인지를 통한 피난 정책 결정 모델 개발 및 구현

	템 개발	②센서 정보를 통합한 재난전조대응 M2M 대응모듈 개발
	D.재난대응 대화형 스마트 네트워크 관리 시스템 고도화	①재난대응 대화형 스마트 네트워크 관리 시스템 테스트베드 개발 ②테스트베드 운용을 통한 통합 관리 시스템의 현장 대응 정확도 개선

제2장. 국내외 기술 개발 현황

1. 관련 분야의 국내외 연구동향

가. 국가 연구개발사업 유사연구 실적

(1) 연구개발과제의 중복방지를 위한 조사 및 검토결과

(단위 : 억원)

부처명	사업명	과제명	예산	비 고
소방방재청	13년도 자연재해 저감기술개발사업	지능형CCTV재난활용 및 경고 · 대응기술개발	3.88	
중소기업청	중소기업기술혁신개발사업	USN기반의 지능형 환경 모니터링 및 재난감지 서비스	3.10	
미래창조과학부	차세대정보 · 컴퓨팅기술개발	사용자 적용형 MIMO HCI를 위한 지능형 양방향 피드백 인터페이스 연구	1.65	
소방방재청	인적재난안전기술 개발사업	지하 공동구 다중 센서 기반 스마트 DCV(Demand Control Ventilation) 시스템 개발	1.87	
중소기업청	첫걸음기술개발사업	무선인터넷을 이용한 종합재난 방재 장치와 그 시스템	0.99	
정보통신부	IT신성장동력 기술개발	재난구조용 웨어러블 인터페이스 기술개발	13.50	
환경부	지역환경 기술개발사업	부산지역 지하철 역사 내 실시간 환경모니터링 시스템 개발	0.40	

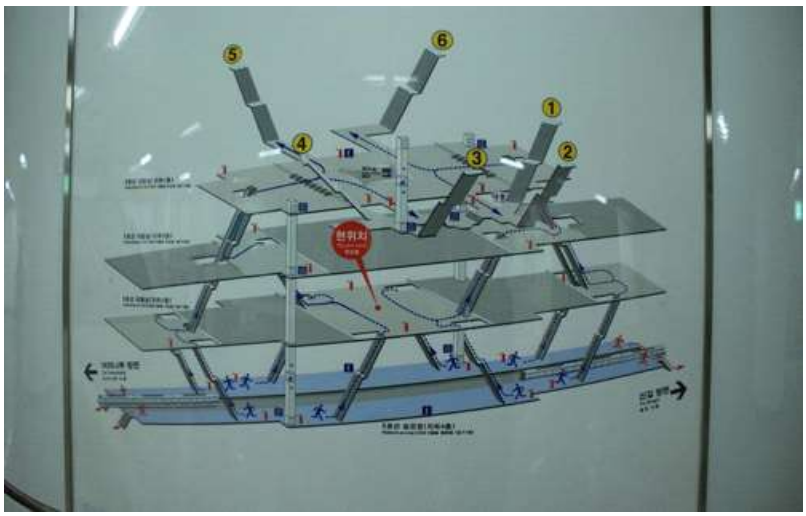
(2) 연구개발과제의 차별화 방안

① 현재 기술의 주요 문제점

○ 국내 지하철의 모니터링시스템은 몇 개의 유선센서를 이용하여 온도, 습도, 화재 감지기 혹은 특정 대기오염을 모니터링 하는 수준으로 국소지점에 대한 유선 방식은 [그림 3] 같이 공간구조가 복잡할 경우 측정지점에 따라 큰 결과치를 나타내므로 대표적인 모니터링 값을 추출하기가 어려운 문제가 있다.



[그림 3] 국내 도시철도의 방재모니터링시스템



[그림 4] 국내 지하철의 복잡한 지하 공간 구조(지하 4층)

○ 국내 지하철의 재난예방 및 대처시스템은 [그림 4]에 나타낸 것과 같이 화재감지기와 제연 팬의 운영현황을 파악할 수 있는 수신기로 이루어져 있으며, 지하 역사를 몇 개의 구역으로 구분하고, 각 구역에서 작동되는 화재감지기의 위치를 파악한 후 구역별로 급기와 배기를 실시하는 방식이다. 이는 주로 화재발생에 대한 대처를 목적으로 구성된 것으로 파악된다.



[그림 5] 화재 관련 정보 수신기

○ **[문제점1]** 유선 중심의 시스템으로 국부유선 화재감지기 센서를 설치하여 운영하므로 화재 외에 독가스살포, 지진, 생물학적 테러 등 재난 발생 시는 능동적으로 대처하기 어렵다.

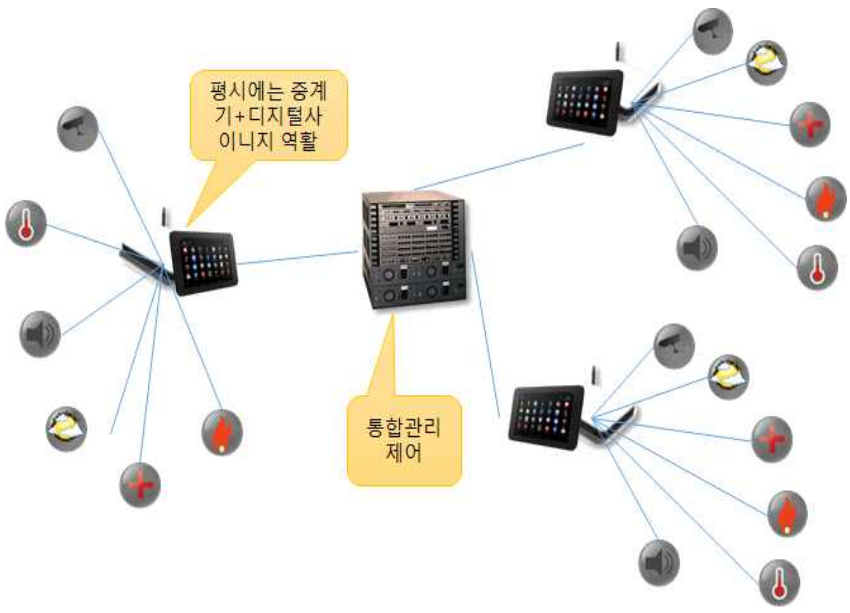
- 화재 발생 및 독가스 살포 시에 대한 대응으로 현장 확인, 승객대피, 자기보호 등의 초기 조치를 주목적으로 하며, 종합관제센터에서 119, 112 및 군부대 등에 상황 발생 위치를 신고하는 시스템으로 구성되어, 전반적으로 승무원이 직접 조치해야하는 문제가 있어 인적오류 발생의 가능성이 내포되어 있다.

○ **[문제점2]** 유·무선 기반 센서를 활용 상황파악 및 공간 내부 문제점 파악 및 상황전파가 유기적으로 이루어지고 있으나, M2M 기반 네트워크 기술의 적용과 활용은 아직 미고려 된 상황으로 기술 개발을 통한 시장성 확보 필요한 단계이다.

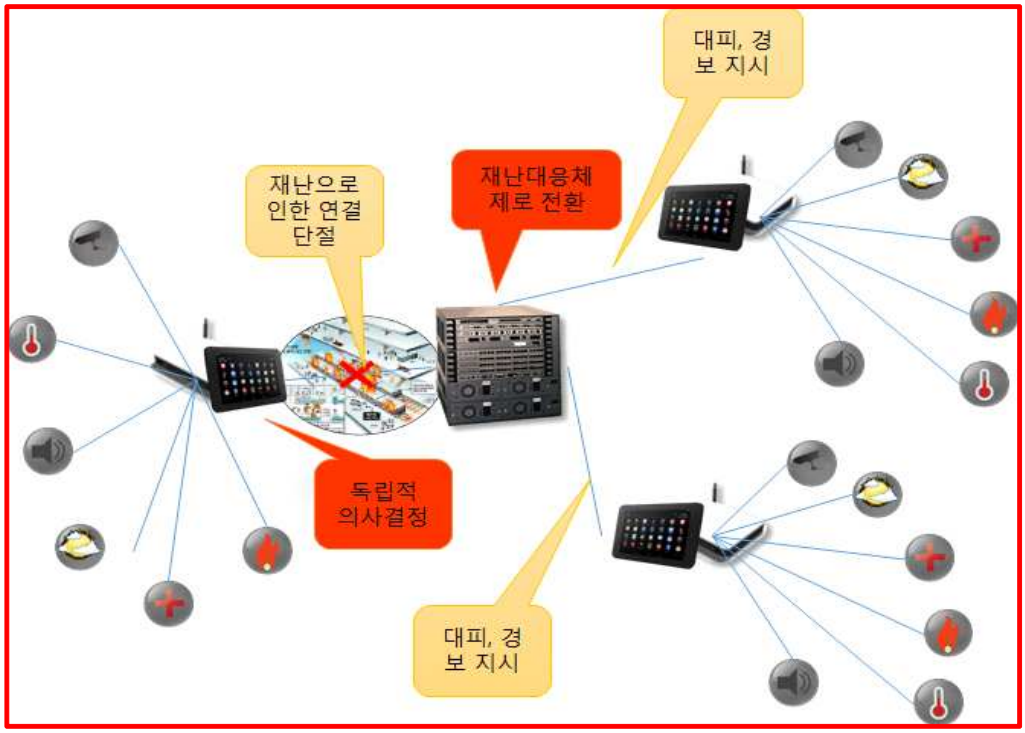
② 기존 기술과의 차별화

○ 지하철 감시 시스템과 지하공간의 재난의 대응하는 기술의 경우 대부분 CCTV와 시공 및 운영 중 설치한 계측 센서를 통한 응답결과 분석, 그리고 이를 통한 구조물의 변형 및 문제점 도출이 대부분이다. 하지만 본 과제를 통해 개발되는 시스템의 경우는 구조물 변형 데이터와 붕괴 징후, 진동, 지진 등의 데이터를 센서를 통해서 스마트 센싱 네트워크 디바이스로 수집되면, 중앙 서버에서는 데이터와 각종 파리미터를 바탕으로 재난 발생 시 대응 시니리오 및 프로세스에 대한 알고리즘을 스마트 센싱 네트워크 디바이스에 업데이트하고, 재난 발생 시 각 스마트 센싱 네트워크 디바이스는 로컬 센서들의 정보를 바탕으로 시나리오에 따라서 자율적으로 대응하는 형태로 개발될 것이다.

○ 이는 재난에 의해서 중앙 서버가 이상이 생겼을 경우, 전체 네트워크가 마비되는 현재의 시스템적 문제를 극복해줄 수 있으며, 주요 내용을 수집하는 선단에서 불필요한 데이터의 필터링 할 수 있어 중앙서버의 부하를 경감시키는 효과를 확보할 수 있고, 이는 전체 시스템 구축 및 운영에 있어서의 코스트 절감과 효율성 증대라는 효과를 가질 수 있다.



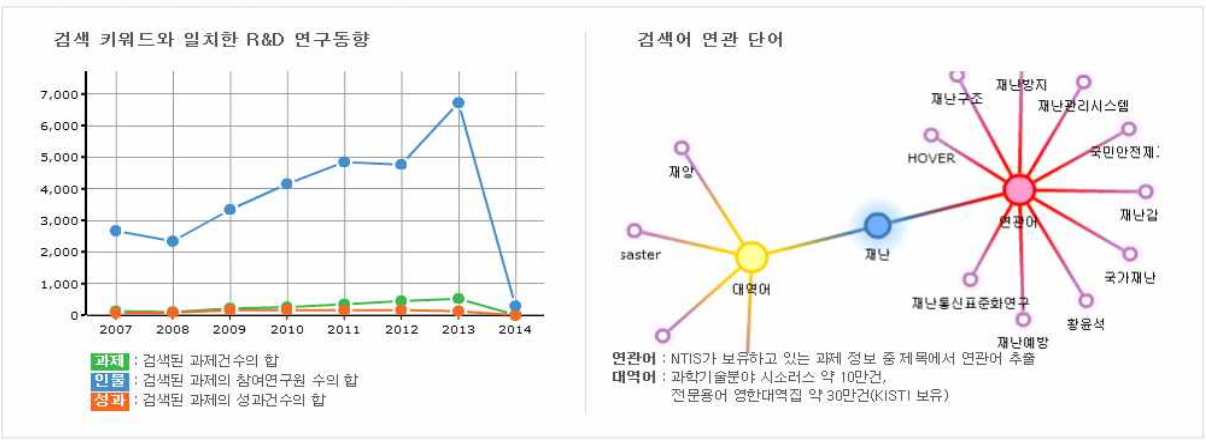
[그림 6] 평상시의 시스템 운용 예



[그림 7] 재난 발생 시의 시스템 운용 예

나. 관련분야의 국내외 연구동향 및 학술적 흐름 분석

○ 현재 재난 감시 및 관리시스템은 재난의 전조를 감시할 수 있는 센서, 위험상황을 판단 및 경보할 수 있는 조기 경보 시스템, 그리고 그 운용을 위한 기준 및 체계로 구성되고 있다. 즉, 관련 분야 대부분의 방향성이 공간정보나 센서 정보를 기반으로 재난의 감시 및 재난정보의 전달의 측면으로 발전되어 왔다. 특히, 국내의 관련분야 연구 역시 재난 예측, 경보, 재난 발생 시 정보의 수집 및 전달하는 재난 모니터링 시스템 등 IT시스템에 그 초점이 맞추어져 있으며, 통합정보·대응체계 구축에 대한 연구가 다수를 차지하고 있는 실정이다.



[그림 8] NTIS 과제 유사 과제 분석을 통한 국내 연구 동향 조사

○ 성과 논문 검색 결과 전체 : 414건, 상위 항목 중 재난관련 항목 추출
- 재난전조 감시용 센서 및 조기경보 기술에 관한 연구
- 센서 기반 유해가스 재난전조 감시 시스템을 위한 재난 대응 체계

- 3차원 BIM 기술을 활용한 수방시설의 능동형 재난관리체계 구축
- All-IP 기반의 이종 재난통신 무전기 음성 연동 방법 연구
- 미래형 재난통신 시스템의 기술동향
- 지상파 방송을 이용한 재난경보방송
- PLC 기반 인적 재난 방지 및 위치 정보 모니터링 시스템 구현
- 고층 건축물의 재난전조 모니터링의 적용성에 관한 연구

○ 지하철의 경우 다양한 전기·전자·기계장치 등과 복잡한 시설구조로 인하여 재난 발생 시 적절한 조치를 신속하게 하여야 피해의 확산을 막을 수 있을 것으로 판단된다. 특히 불특정다수인이 이용하는 공공시설물로서 현지지형에 익숙하지 않은 유동인구가 승객의 대부분을 차지하는 지하철은 피난로가 한정된 지형으로 집중되어 있고 에스컬레이터나 엘리베이터 등 승강기의 정지, 열차 진입 문제, 앞서 언급하였던 다양한 전기·전자·기계장치 등 재난 발생시 복합적인 측면들을 즉각적으로 판단 및 결정하여야 하고, 각 전문가들이 항시 사고현장에 상주하고 있지 않아 상황에 대한 판단이 미흡한 부분이 발생할 수 있으며, 보고로 인한 시간지체 및 상급자의 결정보류 등 관리자의 의사결정이 지연될 경우 막대한 피해를 초래할 수 있다. 재난 발생 및 피해확산을 방지하기 위하여 최소한의 1차 대응체계를 시스템으로 적용하여 의사결정지원을 할 수 있는 방안을 강구해야 할 것이다. 또한 이러한 대응체계를 통하여 피해확산을 방지함과 그 결과로 적절한 의사결정에 도움을 줄 수 있을 것이다.

○ 재난 발생 시 수집된 정보를 바탕으로 한 1차적인 대응체계 판단 및 초기 대응 매뉴얼을 적용한 시스템을 구축한다면 재난상황 발생 시 사고대응체계를 신속하게 전개할 수 있으며 재난에 대한 정보를 중앙관제시스템에서 수집·보고·대응방안 모색 및 결정 후 전달하는 것이 아니라 쌍방향으로 정보를 전달하여 필수적인 의사결정지원을 할 수 것이라고 판단된다.

○ 해외 선진국의 사례를 살펴보면 미국에서는 재난재해정보의 지원에 있어 재해 예방단계에서 많은 정보를 기반으로 의사결정을 위해 정확한 정보를 제공하는 전국가적 혹은 국제적 네트워크를 구축하려는 노력을 진행해 왔으나 구축·운용되고 있는 방재관련 시스템은 지진이나 홍수 등의 재해발생을 사전에 파악하기 위한 감시 및 예·경보시스템과 재해에 의한 피해예측, 피해액 산정시스템 등의 구축하였다. 통합재해관리시스템을 살펴보면 기술과 정보시스템의 급격한 발전으로 데이터에 접근할 수 있는 능력과 정보수집 능력이 크게 향상되었으며 또한 고도화된 센서의 사용, 통신 채널의 확대, 인터넷 사용 확대, GIS, GPS 및 모델링과 시뮬레이션 기술의 고도성장은 통합재해시스템을 구축하는 기반이 되었다. 이러한 시스템은 재해의 모니터링, 시뮬레이션, 사회기반 구조시설 및 위험지구, 재해 지역의 파악을 가능하게 했으며, 재해발생 전후에 자원이 부족한 지역을 대상으로 가장 효과적이고 신속하게 자원을 할당할 수 있도록 도와주는 역할을 지향하고 있으나 이는 의사결정을 위한 보다 정확한 정보수집 기능만을 수행할 뿐 기능적으로 기술의 성장이 직접적인이고 기초적인 의사결정지원 역할을 하고 있지는 않은 실정이다.

○ 일본의 경우 재난재해관리체계는 정부기구 이외에 주요 재난정보 시스템을 보유하고 있다. 일본의 방재관계 기관과 도도부현은 재해 대책을 원활하게 수행하기 위해 방재정보시스템을 구축하고 대규모 재해 시 효과적인 통신 수단이 될 수 있는 무선통신시설을

정비하는 쪽으로 정책방향을 설정해 나가고 있다. 재해대책용 무선망으로 중앙방재무선망, 소방방재무선망, 도도부현 방재행정무선망, 시정촌 방재행정무선망, 방재상호통신용 무선망 등이 있고, 각 지방의 상황에 맞는 방재시스템이 구축되어 있다. 이러한 시스템이나 체계 역시 의사결정을 위한 기초적인 자료수집이나 정보 활용 및 전달을 지원하는 역할을 하는 수준에 불과할 뿐 적극적인 의사결정지원 수단이라고 보기에는 미흡한 실정이다.

○ 각 국가별 재난대응 시스템을 살펴보면 대부분의 국가에서 재난예측이나 재난발생시의 모니터링 시스템을 확보하려 하고 있다. 그러나 이러한 시스템들은 관리자의 의사결정을 위한 다양한 정보들을 보다 정확하고 구체화하는데 지나지 않는다. 재난 발생 시에는 초기대응이 가장 중요한 핵심 요소라고 할 수 있는데 이에 핵심적이고 필수적인 초기대응 매뉴얼들을 시스템화 하여 적극적으로 기능화 할 수 있도록 하고 신속한 재난대응 체계를 구축하려는 노력이 진행되고 있으나 사회 전반적인 인식이나 노력은 아직 미흡한 실정이라고 볼 수 있을 것이다. 지하철 건축물 구조 및 전기 분야의 시설, 작동에 대한 사실적인 의미의 지능적 의사결정지원체계를 도입방안을 연구하여 관련 분야에 대한 선도적인 입지를 마련할 필요성이 있으며 성공적인 연구결과를 토대로 이를 재해발생현장에 반영하여 즉각적인 대응뿐만 아니라 피해상황 최소화 방안을 강구할 수 있을 것이다.

2. 국내외 관련 기술 특허 동향

no.	지식재산권명	지식재산권출원인	출원국/출원번호	비고
1	전력선 통신을 기반으로 하는 지능형 재난 대피 유도 시스템 및 방법	현대중공업 주식회사	한국/ 1020080104836	등록
2	양방향 재난관련 메시지 전송 시스템	(주)아이들	한국/ 1020120112789	공개
3	로봇 기반의 무선 센서 네트워크 확장 방법, 이를 위한 중계 로봇 및 무선 센서 네트워크 시스템	재단법인대구경북과학기술원	한국/ 1020090087908	등록
4	재난대응 및 멀티콘텐츠와 상호작용 정보용 디지털 사이니지 시스템 및 운용방법	목원대학교 산학협력단	한국/1020120038172	등록
5	고효율 다위처리 프로토콜을 위한 현장센서노드장치	유비텔레콤 주식회사 강릉원주대학교산학협력단	한국/1020090022281	등록
6	스마트형 소방 방재시스템	강원대학교산학협력단 (주)두원	한국/1020120084085	등록
7	ＵＳＮ/ＲＦＩＤ 태그 통합모듈	주식회사 맥스포	한국/1020050081928	등록
8	유비쿼터스 무선네트워크 시스템, 노드모듈 및 그 작동방법	주식회사 맥스포	한국/ 1020050133045	등록
7	RESCUE SIGNAL TRANSMITTER AND RESCUE SUPPORT SYSTEM	SEIKO EPSON CORP	일본/ 24001427	
8	ACCIDENT MANAGEMENT SYSTEM AND METHOD THEREFOR	KYUNGWON UNIVERSITY INDUSTRY COLLABORATION FOUNDATION	일본/23101414	

3. 국내외 시장분석 및 산업동향

○ 위기상황 감지 및 의사 전달 기술 개발을 위해 로봇, USN/RFID 태그, 유비쿼터스, 센싱, 메시지 송수신 장치, 노드모듈, 사용자 단말기, 디지털 사이니지 등 어려 가지 매체를 활용한 중계기기로써 역할을 하고 있으나 그 기능이 전달이나 감지정도의 수준이여서 중앙관리시스템과의 교신이 끊어지게 되면 진압 및 대피하는 경우는 제한이 따를 것으로 판단된다. 또한 국내 특허의 일부는 데이터를 가공하여 응급처치 요령, 상황전파, 재난대응 콘텐츠 메모리 탑재 등 위기상황에 대응이 가능하도록 중계기에 데이터가 탑재 되어있었으나, 디스플레이의 한계 및 기기간의 의사소통 기능이 미비하여 통합 재난대응 시스템으로 사용하기에 모호하다 판단된다.

○ 현재 국내에 현장조치 행동매뉴얼이 시설물별로 다양하게 존재하지만 지하시설의 경우 그 시나리오가 많아 어떤 것을 취사선택해야 할지 절차가 수립되지 않아 의사결정에 어려움을 겪고 있는 것으로 안다. 이렇듯이 행동매뉴얼을 적시 적소에 상용할 수 있도록 시스템이 마련되어야 할 것으로 관리자 뿐아니라 사물을 통한 초기대응 역시 필요하다 판단된다. 위의 표에 등록된 특허를 통하여 아래와 같이 본 과제에서 제안하는 목적 및 목표의 차이점을 기술하였다.

no.	지식재산권명	차이점	비고 (활용 고려)
1	전력선 통신을 기반으로 하는 지능형 재난 대피 유도 시스템 및 방법	중앙시스템과 사용자 단말기 간의 양방향 의사소통으로 전력선 기반(유선)한 대피경로 설정 및 비상등 점등, 음성신호를 이용한 대피유도 시스템 활용 분야 : 선박, 해양구조물, 건물내부 차이점 : 중앙시스템과 연결 단절 시 중계기 의사결정 불가능	
2	양방향 재난관련 메시지 전송 시스템	메시지 송수신 기술을 활용하여 자연재해 발생 시 사용자에게 메시지를 전송 차이점 : 자연재해 발생 시 사용하기 위한 시스템이며, 양방향 의사소통이기 보다는 메시지 전달에 그치는 한계가 있음, 또한 중계기기가 아닌 사용자 단말기라는 점에서 의사결정 지원 모호함	
3	로봇 기반의 무선 센서 네트워크 확장 방법, 이를 위한 중계 로봇 및 무선 센서 네트워크 시스템	차이점 : 로봇과 사용자 단말기기 간의 네트워크이며, 로봇은 단말기에 전달만하는 매체이므로 대화형 의사소통이 불가능 함. 또한 디스플레이 및 행동매뉴얼과는 무관함	로봇의 이동할 수 있다는 특징을 활용(공간 확장성) 넓은 지하공간에 유용
4	재난대응 및 멀티콘텐츠와 상호작용 정보용 디지털 사이니지 시스템 및 운용방법	무선센서를 이용한 디지털 사이니지 시스템으로써 사용자에게 정보를 제공하고 상황전파(재난대응 콘텐츠 메모리 탑재, 응급처지 요령 제공)를 한다는 점이 유사 차이점: 대화형 의사소통 불가능	데드 레코닝(Dead Reckoning) 위치파악 가능함

no.	지식재산권명	차이점	비고 (활용 고려)
5	고효율 다위처리 프로토콜을 위한 현장센서노드장치	무선을 사용한 중계기기(허브) 간의 대화형 네트워크로 존재 차이점 : 센서 자체적으로 의사결정 지원이 이루어지지 않고, 센서는 존재하지만 디스플레이 및 sop가 미흡하여 통합시스템이라고 하기 어려움	FRID(무선테그인식기술) 업그레이드 가능
6	스마트형 소방 방재시스템	화재 탐지설비를 통하여 의사결정을 하며, 중계기기는 센서의 역할에 그치는 한계. 차이점 : 무선연결이 아님, 센서는 존재하지만 디스플레이 및 행동매뉴얼과는 무관함	
7	USN/RFID 태그 통합모듈	센싱 개체간(중계기기 간)의 통신기능을 부여하면 양방향적 의사소통이 가능함 무선으로 이동 또한 가능하며 시설비용 절감효과 있음 차이점 : 센싱 자체의 의사결정 지원 불가능	
8	유비쿼터스 무선네트워크 시스템, 노드모듈 및 그 작동방법	유비쿼터스 네트워크에 관련된 것으로, 더 상세히는 유비쿼터스 네트워크를 구성하는 노드모듈에 관한 것 무선 네트워크를 이용하여 중계기기 간의 대화형 의사소통이 가능하는 점이 유사 차이점 : 중계기기 자체적으로 의사결정 지원이 블가능하여 초기대응이 어려움	

4. 종합결론



◦ SO : 방재 및 IT 분야 다수 보유를 통해 현재 국내 재난에 의해 발생되는 피해를 최소화하고 미

래 선진기술을 국내기술력으로 확보 할 예정이다.

◦ ST : 재난 대응 절차 및 프로세스 확보는 어려우나, 방재 및 재난 연구에 다수 실적을 가진 인력 및 인맥을 동원하여 재난관련 전문가 및 재난 기관에 자문을 구할 예정이다.

◦ WO : 빠른 기술력확보로 프로세스 구현하여 사업화 방안을 모색하고 국내 최초의 M2M 기반 재난 전조 예측 및 초기 대응 체계를 구축한다.

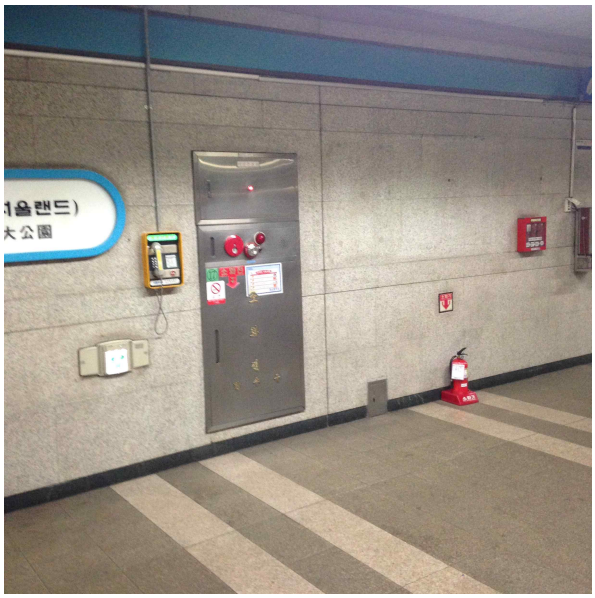
◦ WT : 빠른 기술력확보로 프로세스를 구현하여 시범 적용 문제점 극복

제3장. 연구 수행 내용 및 성과

1. 지하공간(지하철) 비상대응 매뉴얼

가. 연구 내용

본 연구 개발은 지하공간(지하철)에서 발생 가능한 재난에 대응하기 위한 M2M기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크(IoT) 시스템 개발을 통해 해당 시설물에서 발생하는 재난을 센서를 통해 감지하고, 승객 대피 의사결정 등 재난 대응 중심의 선제적 재난관리체계 구축 및 신속한 의사결정 지원 기술을 제시하는 것이다. 외와 같은 최종목표를 바탕으로 1차년 도는 국내 지하시설 재난 발생 시 대응 행동 매뉴얼 조사를 진행함으로서 지하시설(지하철) 재난 대응 프로세스(sop)기초 연구를 진행하고자 한다.
아래의 사진과 같이 본 과제의 실험을 진행하기로 한 4호선 대공원역의 경우 일반 역사와 비슷한 화재 대응 장비를 구축하고 있다.



소화기구 및 비상 전화



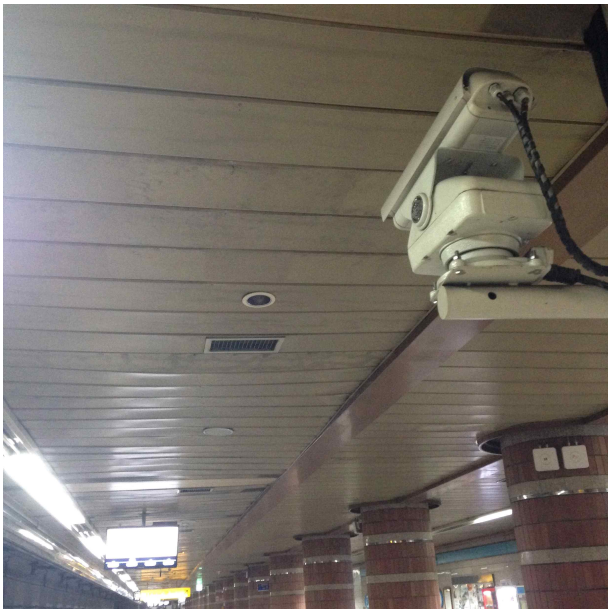
화재용 긴급 대피 마스크

하지만 4호선 대공원역의 경우 봄과 여름철과 같이 소풍, 동호회 모임 또는 마라톤 등 많은 인원이 모이는 큰 행사가 많다는 특징을 가지고 있다. 특정 기간에 많은 이용객이 모이는 상황에서 비상대응 상황에서는 화재 진압 장비가 부족 할 것으로 예상이 된다.
또한 4호선 대공원역은 아래와 같이 다수의 CCTV 및 방향 지시등을 확보하고 있다. 4호선 대공원역의 경우 평상시 CCTV를 통해 24시간 사각지대가 없이 전철역 모든 곳을 확인하고 있으며, 방향 지시등을 통해 승객이 편리하게 출구를 찾아 지하 플랫폼으로 부터 안전하게 지상으로 이동이 가능하다. 이런 장비는 화재 발생에 있어서, 화재 발생 연기가 상승하는 특징에 따라 방향 지시등 및 CCTV 연기에 의해 시양의 확보가 어려울 것으로 예상이 된다. 이러한 점에 있어 승객 및 관리자가 비상 상황에 있어서 시야 확보를 통해 CCTV 및 방향 지시등 이용하기가 쉽지 않을 것으로 짐작 할 수 있다. 위에서 나열한 지하시설(지하철) 장비 상황에서 각 지하시설(지하철) 마다의 재난 발생 시 대응 행동 매뉴얼을 정리 하고자한다. 국내의 경우 지하철의 운영 기관을 Seoul Metro와 Korail 나눌 수 있으며, 각각 기관의 비상대응 매뉴얼을 정리하며, 4호선 대공원역을 포함한 그 외의 지

하철역 비상대응 매뉴얼 정리를 진행하고자 한다.



방향 지시등



플랫폼에 설치된 CCTV

나. 기존 SOP 모델

「소방시설 설치·유지 및 안전관리에 관한 법률」제20조 및 동법시행규칙 제14조 규정에 근거하여 비상 상황에 대한 전반에 관하여 필요한 사항을 정하고 이를 실천함으로써, 지하철 화재 및 비상 상황에 있어 예방, 경계 또는 진압하여 인명과 재산을 화재로부터 적극 보호할 수 있는 제도적 장치를 마련함을 목적으로 한다.

1) 기관별 비상대응훈련 및 지하철역 자체 소방계획

□ 서울메트로

훈련방법

○ 열차 및 대합실 화재
○ 전동차 화재 발생
○ 대합실 폭발 및 화재 발생

○재난상황 대응
○유관기관 재난상황 전파
○승객대피 등 초기대응
○ 유관기관 화재 진압, 사상자 구호 등 대응

○ 시설복구 열차재계
-시설물 복구
-열차운행 재계

기관별 임무 및 역할

○ 서울메트로 조치사항
- 대내 및 유관기관 상황전파 및 긴급 구조 등 지원요청
- 안내방송 실시, 승객 긴급대피 유도, 열차사상자 구호, 화재 초기진화 등
- 서울메트로 중앙사고수습지원본부 및 현장지휘소 설치
- 운행중단에 따른 대체교통 연계 수송대책
- 사고현장 응급복구 실시 및 열차운행 재개

○ 소방서
- 긴급구조통제단 설치 운영
- 구조차, 구급차, 펌프차 출동
- 부상자 구호 및 인명구조 / 사상자 병원 후송
- 화재진압(열차, 승강장, 대합실, 출입구
- 승강장 및 대합실 잔류 승객 확인 및 점검

○ 경찰서
- 지하철역 인근 교통통제
- 화재현장 경계근무 / 피해승객 구조
- 열차방화범 진압 및 역사폭발 현장조사
- 역사순찰 및 질서유지

○ 군부대
- 역사 외부 출구 경계 근무 실시
- 승강장, 대합실 화재 / 붕괴로 인한 출구 붕괴 및 화재 사상자 구호

○ 구청
- 대피유도 및 시민홍보
- 지역 재난안전 대책본부 설치
- 통합지원본부 운영
- 응급의료소 설치 운영

훈련계획

단계	조치사항	내용	비고
훈련보고	안내방송 훈련브리핑	• 훈련안내 방송 실시 및 훈련계획 보고 역사 및 4호선 운행열차 안내방송	종합관제소 지하철역

		훈련 관계열차 정리 훈련계획 보고	승무원
1단계	상황1 발생	• 열차화재발생/열차운행중지/사상자 발생	
	상황전파 및 초기대응	• 승무원 ⇒ 종합관제소 ⇒ 지하철역 ⇒ 유관기관 등 (소방서, 구청, 경찰서, 군부대 등)	승무원 종합관제소

단계	조치사항	내용	비고
1단계	상황전파 및 초기대응	• (종합관제소) 유관기관 상황전파 일제방송, 자위소방대 출동, 장애인등 교통약자 등 사상자 구호, 승객대피유도 및 초기화재진화, 방독면 등 구호장비 지급	종합관제소
		• (승무원)안내방송 및 승객 대피유도	승무원 지하철역
		• (지하철역) 조동조치 시행 안내방송, 자위소방대출동, 장애인등 교통약자 등 사상자 구호, 승객대피유도 및 초기화재진압, 방독면 등 구호장비 지급	지하철역
	유관기관	• 소방서, 경찰서, 군부대 등 출동	유관기관
2단계	상황2 발생	• 대합실 상가 앞 폭발 화재/사상자 발생 (승강장에서 대합실 이동시 연기발생기 작동)	지하철역
	상황전파 및 초기대응	• 직원 ⇒ 역무실 ⇒ 종합관제소 ⇒ 유관기관 소방서, 구청, 경찰서, 군부대등	종합관제소 지하철역

단계	조치사항	내용	비고
2단계	상황전파 및 초기대응	• (지하철역) 조동조치 시행 안내방송, 자위소방대출동, 장애인등 교통 약자 등 사상자 구호, 승객대피유도 및 초기 화재진압, 방독면 등 구호장비 지급	지하철역
		• (종합관제소) 구간 일제방송으로 전 열차 상황 통보 및 운행통제, 전역 상황통보	종합관제소
3단계	화재진압 및 사상자구호, 현장 교통통제	• (소방서) 승강장, 화재진압 및 인명구조 (light line 설치)	소방서 119구조대
		• (경찰서) 지하철역 교통통제, 화재현장 출입통제, 경계근무(Police line 설치)	경찰서
		• (군부대) 화재 대합실 상가 인명구조, 외부출입구 경계근무 실시	군부대
		• 대피유도 및 시민홍보	구청
		• 사상자 구호 및 승객대피 유도	보건소
4단계	복구조 투입 및 훈련종료	• 화재진압/승객대피 후 관제에 상황보고 (시설 복구조 투입) 차량, 전기, 건축, 통신, 신호, 기계, 전자 등	

□ **코레일**

훈련방법

본사주관 비상대응 종합훈련·평가

본사주관 훈련평가는 불시훈련으로 시행

평가반에서 지정한 장소로 출동하여 훈련메시지에 따라 조치

중앙사고수습지원본부 운영

(중앙사고수습지원본부를 운영하지 않는 경우에는 운영상황실로 대체)

홍보문화실, 운영상황실은 조치내역을 평가반에 메일로 전송

복구용 기중기는 착발선에 전선, 출발준비 완료 신고 후 입고

사고복구는 복구계획을 수립하여 발표하는 것으로 갈음

지역본부 자체 비상대응 종합훈련・평가

지정과제에 대하여 시나리오를 작성하여 실제훈련 시행

(사고규모는 대형사고 기준(사망 10명 이상))

실제훈련 시행 전 분야별 및 유관기관과 역할분담, 시나리오 검증을 위한 토의형 모의훈련 시행

사고보고 등은 철도 교통관제센터로 하는 것을 원칙

(훈련 시행 전 해당권역 선임관제사와 협의 철저)

실제훈련을 원칙으로 하되, 대외기관 홍보・의료기관 후송 등 부득이한 사항은 자료작성 후 보고하는 것으로 갈음

훈련 장소는 열차운행에 지장 없는 곳을 선정하여 분야별로 복구훈련 시행

(차량, 선로, 전차선(전철구간에 한함), 신호분야)

고속선 관할 지역본부는 연 1회 고속유지 보수소속과 합동훈련 시행

(차량정비단, 고속철도시설・전기사무소 참여시 해당 분야 부분훈련으로 인정)

재난대응 안전한국훈련 및 을지연습 등은 정부기관 훈련방침에 따라 조정 시행

훈련계획

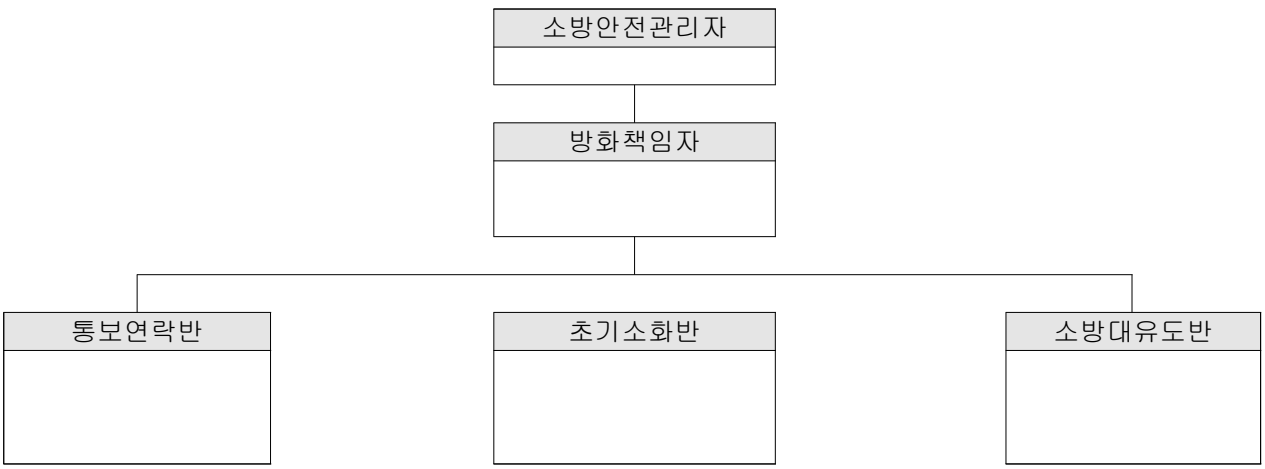
구분	설명	훈련상황	비고
화재상황 발생	6104열차 화재발생으로 지하철역에 신고	• 기관사: “6104열차 객차에서 화재가 발생 하였으니 조치하여 주시기 바람” • 지하철역: “예 알겠습니다. 현재위치?” • 기관사: “지하철역 진입 중!”	무전기
무선방호	지하철역에 진입하는 상·하행 열차 정차지시	• 역장: 무선전화기를 채널2에 맞춤고 “화재 발생 앞뒤 역간 상·하선 무선방호”	
119신고	관내소방서에 화재진압요청	• 부역장: 소방서에 화재 상황 신고, 지하철역 상행승강장 전동차에 화재가 발생전파, 지하철역 출동하시면 붉은 깃발을 들고 승강장으로 안내 • 사회복무요원 1: 붉은 깃발을 들고 (출구에 출동하여 119소방대원 유도안내)	
소방차 진입	소방차 유도 안내		
PSD출입문 개방	스크린도어 개방 소방대 화재 완전 진화	• 부역장: 전산실 PSD종합도어제어반으로 이동 • PSD조작반 스위치를 수동으로 조작 • 스위치를 이용 상·하행 수동 열림 조작	전화기
선로출입문 개방		• 부역장: PSD 경보제어반으로 이동 • 선로 출입문 스위치 열림으로 조작	전산실에서 스위치조작
게이트해제	게이트 사전 해제하여 여객을 신속히 대피	• 역무원1: 구 매표소로 달려가 게이트해제 주전원 스위치를 내린다(OFF)	구매표소에서 액션

구분	설명	훈련상황	비고
엘리베이터 정지	화재 및 정전 엘리베이터 전원 차단	• 부역장: 사회복무요원에 엘리베이터 전원 차단 지시, 엘리베이터 순차적으로 차단 후 • 출구로 이동하여 고객대피 안내, PSD 조작반 스위치를 수동으로 조작	맞이방 엘리베이터

비상방송	화재발생 대피안내방송	• 역무원1 : 휴대용 방송기를 채널#4번에 맞추고 “지하철역 승강장에 화재가 발생하였으니 고객 여러분께서는 신속히 밖으로 대피하여 주시기 바랍니다.” (상황종료시 까지 계속)	짧고 정확하게
인명구조 현장출동	화재현장에 인명구조 및 대피안내 출동	• 역장: 공기호흡기를 착용한다. • 직원1: 공기호흡기를 착용한다. • 역장, 직원1 공기호흡기를 착용하고 소화기를 들고 신속히 상행승강장으로 이동	상행승강장으로 이동
화재진압	휴대 소화기 1차 화재 진압 소화전호스로 2차 진화	• 역장: 화재객차로 접근하여 1차 소화기 발사 • 역장 소화전 호스를 꺼내고 화재객차로 이동 조준 • 역무원1에 소화전호스 개방 지시	상행승강장
사다리설피	선로로 대피한 고객을 대피	• 역장: 화재현장에서 상당거리를 두고 이동 사다리 2개설치	승강장 비상사다리 설치

□ 공덕역

소방대 조직표



화재상황별 임무

맞이방 화재시 임무

직 명	임 무
당무역장	• 열차방호 지시 • 고객대피방송 및 직원임무부여 • 공항철도, 5·6호선 상황전파 및 환승통로 통제요청 • 급보(관제센터, 안전처, 당직실, 119, 112, 인접역) • 현장출동 초동진화 및 여객 피난로 확보
로컬관제원	• 열차방호(진행중인 열차 정차통보) • 관제센터급보 및 관련기관 수시보고 • 대피방송 및 상황전파 • 운전정리
역무원	• 상·하선 게이트개방 및 맞이방 내 고객 대피유도 • 초동진화에 협력
사회복무요원	• 하선 승강장 출동, 승강장내 고객 대피유도 (피난계단, 공항철도, 5·6호선 환승통로 유도) • 초동진화에 협조

직 명	임 무
안전요원	- 상선승강장 출동, 승강장내 고객 대피유도 (피난계단, 공항철도, 5·6호선 환승통로 유도) - 초동진화에 협조
외주직원	- 맞이방 내 고객대피 유도 - 9·10번 출구 통제 - 초동진화에 협조

승강장 화재시 임무

직 명	임 무
당무역장	• 열차방호 지시 • 고객대피방송 및 직원임무부여 • 공항철도, 5·6호선 상황전파 및 환승통로 통제요청 • 급보(관제센터, 안전처, 당직실, 119, 112, 인접역) • 현장출동 초동진화 및 여객 피난로 확보
로컬관제원	• 열차방호(진행중인 열차 정차통보) • 승강장 정차열차 기관사에게 여객대피방송 및 초동진화 통보 • 관제센터급보 및 119·112통보,상황실 수시보고 • 대피방송 및 상황전파 • 운전정리
역무원	• 상·하선 게이트개방 및 맞이방 내 고객 대피유도 • 초동진화에 협력
사회복무요원	• 하선 승강장 출동, 승강장내 고객 대피유도 (피난계단, 공항철도, 5·6호선 환승통로 유도) • 초동진화에 협조
안전요원	• 상선승강장 출동, 승강장내 고객 대피유도 (피난계단, 공항철도, 5·6호선 환승통로 유도) • 초동진화에 협조
외주직원	• 맞이방 내 고객대피 유도 • 9·10번 출구 통제 • 초동진화에 협조

정차중인 열차 화재시 임무

직 명	임 무
당무역장	• 열차방호 지시 • 고객대피방송 및 직원임무부여 • 공항철도, 5·6호선 상황전파 및 환승통로 통제요청 • 급보(관제센터, 안전처, 당직실, 119, 112, 인접역) • 현장출동 초동진화 및 여객 피난로 확보
로컬관제원	• 열차방호(진행중인 열차 정차통보) • 승강장 정차열차 기관사에게 여객대피방송 및 초동진화 통보 • 관제센터급보 및 119·112통보,상황실 수시보고 • 대피방송 및 상황전파 • 운전정리
역무원	• 상·하선 게이트개방 및 맞이방 내 고객 대피유도 • 초동진화에 협력
사회복무요원	• 하선 승강장 출동, 승강장내 고객 대피유도 (피난계단, 공항철도, 5·6호선 환승통로 유도) • 초동진화에 협조
안전요원	• 상선승강장 출동, 승강장내 고객 대피유도 (피난계단, 공항철도, 5·6호선 환승통로 유도) • 초동진화에 협조
외주직원	• 맞이방 내 고객대피 유도 • 9·10번 출구 통제 • 초동진화에 협조

화재상황별 개인임무카드

당무역장 개인임무카드 (화재시)	
지하1층화재	• 화재발생 수신반 작동위치확인 • 인접선운행열차 정차조치 및 병발 사고 방지조치 • 화재상황별 안내방송실시 • 상황에 적절한 직원별 임무부여 • 게이트 비상모드 작동 및 고객대피유도 • 지하1층 9번출구 출입통제 • 사상자 구호조치
지하2층화재	• 화재발생 수신반 작동위치확인 • 인접선 운행열차 정차조치 및 병발 사고 방지조치 • 화재상황별 안내방송실시 • 상황에 적절한 직원별 임무부여 • 게이트 비상모드 작동 및 고객대피유도(9번 출구) • 지하1층 9번출구 출입통제 및 상·하행 게이트 진입통제 • 사상자 구호조치

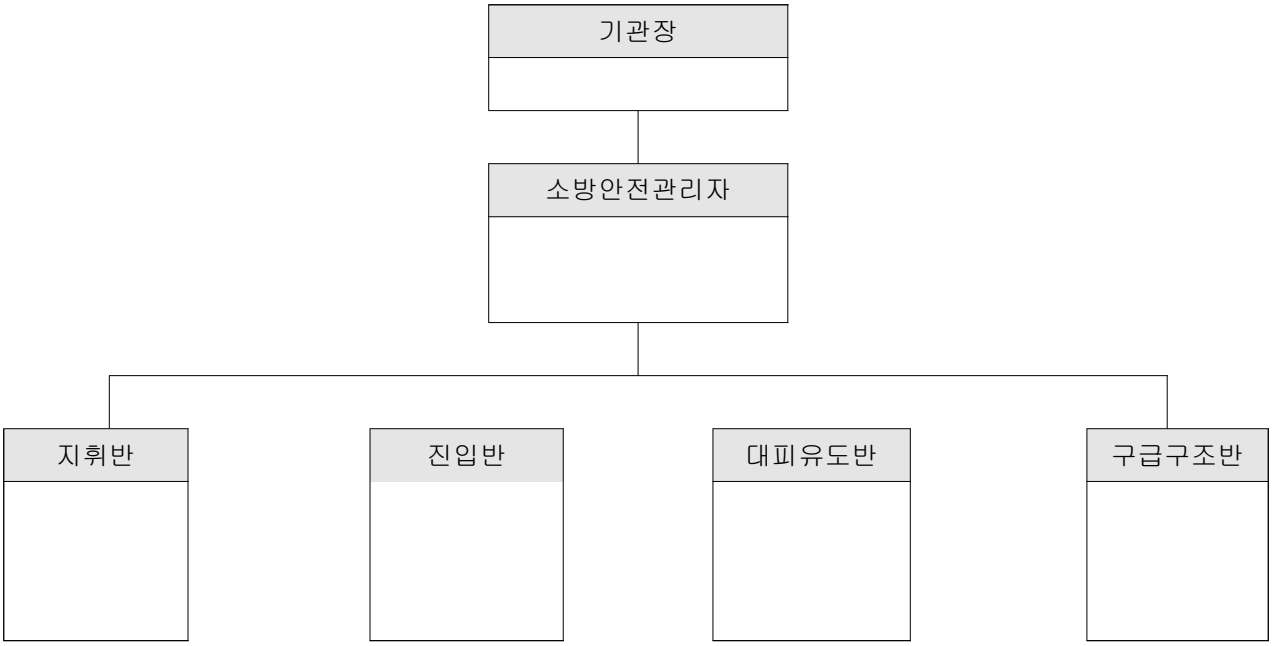
지역관제원 개인임무카드 (화재시)	
지하1층 화재	• 사고지점을 향하는 열차 긴급정차 통보 (필요시 열차방호) • 사고급보 (안전처, 관제센터, 인근역) • 긴급지원요청 (119, 철도경찰, 112, 관리역 • 지하1층 10번 출구 출입통제 및 고객대피유도
지하2층 화재	• 사고지점을 향하는 열차 긴급정차 통보 (필요시 열차방호) • 사고급보 (안전처, 관제센터, 인근역) • 긴급지원요청 (119, 철도경찰, 112, 관리역) • 지하1층 10번출구 출입통제 및 고객대피유도 (10번 출구)

역무원 개인임무카드 (화재시)	
지하1층 화재	• 최초발화지점 출동 화재현장 상황 파악 및 역장보고 (초기진압가능 하면 소화기로 진압 후 보고) • 역사지하2층 상행승강장 및 환승 통로 여객통제 • 역사 서쪽 환승통로나 승강장 양단 끝 피난계단으로 고객 대피유도 • 사상자구호조치
지하2층 화재	• 최초발화지점 출동 (초기진압가능 하면 소화기로 진압 후 보고) • 역사지하2층 상행승강장 및 환승 통로 여객통제 • 역사 서쪽 환승통로나 승강장 양단 끝 피난계단으로 고객 대피유도 • 지하3층으로 내려가 환승통로 통제 사상자구호조치

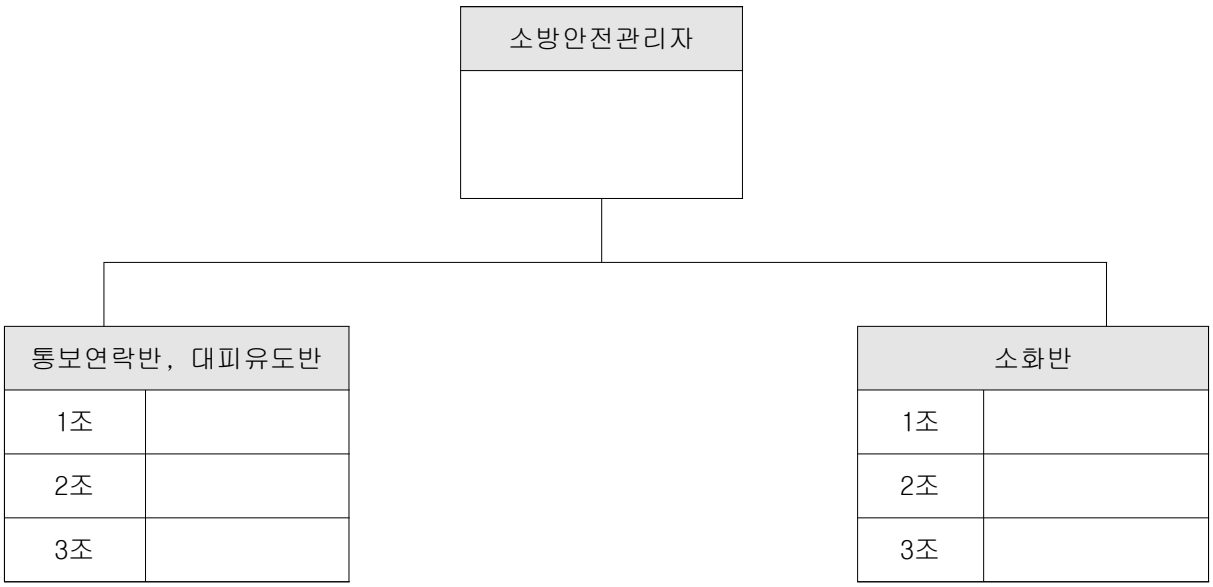
공익요원 개인임무카드 (화재시)	
지하1층 화재	• 최초발화지점 역무원과 같이 출동 (초기진압가능하면 소화기로 진압) • 피난계단이용 하행승강장으로 이동 • 역사지하2층 하행승강장 및 환승 통로 여객통제 • 역사 서쪽 환승통로나 승강장 양단 끝 피난계단으로 고객 대피유도 • 사상자구호조치
지하2층 화재	• 최초발화지점 역무원과 같이 출동 (초기진압가능하면 소화기로 진압) • 피난계단이용 하행승강장으로 이동 • 역사지하2층 하행승강장 및 환승 통로 여객통제 • 역사 서쪽 환승통로나 승강장 양단 끝 피난계단으로 고객 대피유도 • 지하3층으로 내려가 환승통로 통제 사상자구호조치

□ 죽전역

소방대 조직표 (상시)



소방대 조직표 (야간 또는 공휴일)



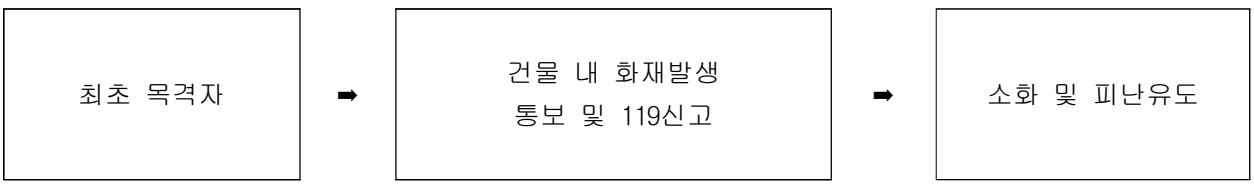
반별 임무

반별	임 무
지휘반 ( )	• 119신고, 건물 내 화재발생 통보 • 관계기관 및 관계자에게 연락
진압반(소화반) ( )	• 소화기, 옥내소화전을 사용 화재진압
대피유도반 ( )	• 근무자 등을 안전한 장소로 대피유도 • 방화문 폐쇄, 가스·위험물 등 제거 • 중요물품 안전한 장소로 이동
구조구급반 ( )	• 부상자 응급조치 • 병원 긴급후송
통보연락반 ( )	• 119 신고, 건물내 화재발생 통보 • 관계기관 및 관계자에 연락

화재상황시 해동요령 및 피난계획

화재상황시 행동요령

우리역에 근무·거주자는 평소 화재를 감지하는데 신경을 써야하며, 누구든지 화재의 발생사실을 최초로 목격하는 자는 119신고, 건물 내 화재발생 통보, 초기소화, 건물 내 거주자피난유도 등의 조치를 취하여야 한다.



화재발생 사실이 건물 내에 전파되면 평소 편성·운영 중인 자위소방대 반별로 개별임무를 즉각 수행하여야 한다.
소방안전관리자는 사업장별 실정에 적합한 세부행동요령을 별도로 정할 수 있다.

피난계획

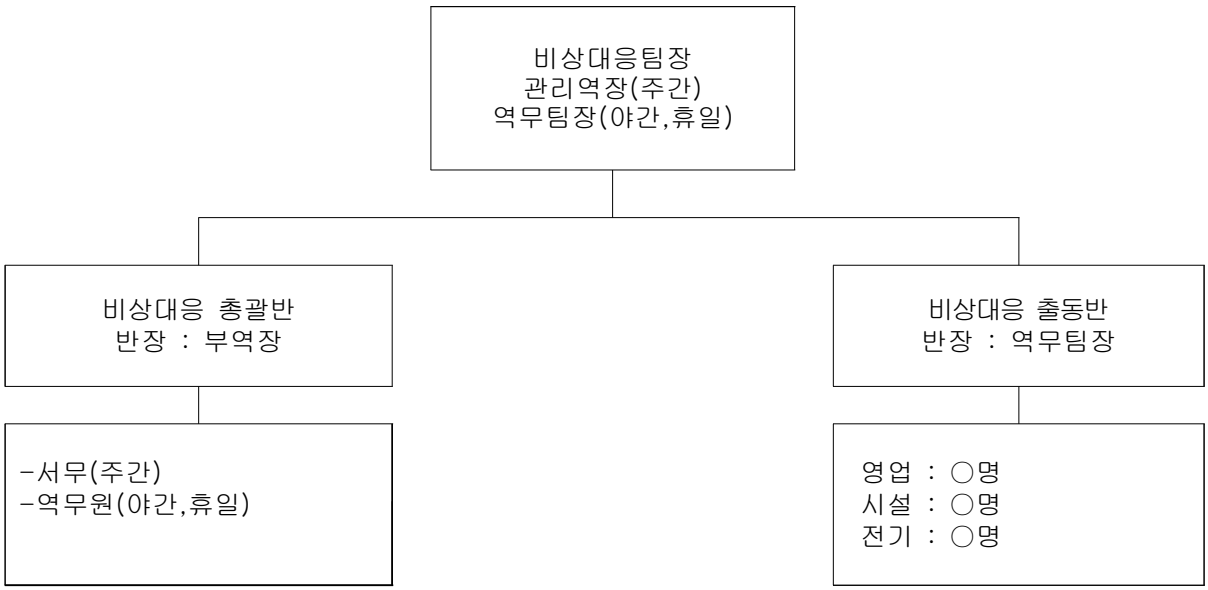
맞이방(1층)
안내방송 및 직원의 유도에 따라 신속하게 각 출구 (1번-3번)에서 역사 외부로 대피한다. 이 경우 노약자 및 장애인이 신속하게 대피할 수 있도록 각 출구 출입문 전부를 완전히

열어둔 상태로 유지한다.

승강장(2층)
1차 : 안내방송 및 직원의 유도에 따라 승강장 계단(E/S 포함)을 이용하여 맞이방 이동 후 각 출구를 통해 역사 외부로 대피한다. 이 경우 게이트 및 비상게이트를 개방하여 신속히 대피할 수 있도록 한다.
2차 : 대형화재로 1차 대피가 불리할 경우 직원의 유도에 따라 타는 곳 남쪽(보정방면) 계단을 이용하여 차량기지를 통해 역사 외부로 대피한다.
화재발생시 자위소방대 대피유도반의 「장애인 노약자 피난보조인」 으로 임무를 부여받은 자는 장애인 및 노약자의 대피유도를 최우선으로 돕는다.

□ **대공원역**

소방대 조직표



위기정보 단계별 조치사항

구 분	조치주체	조치내용
주의 (Yellow)	역(역무원)	• 접수, 보고/전파 • 역사내 임대매장 정전대비 상황 통보 • 정전 대비 장비 및 임무 확인
	전기사업소(전기처)	• 현장출동 • 상황파악 및 보고 • 예비전원 설비 집중 점검
	관제센터(전력, 운전)	• 전력상황 모니터링 강화 • 역사 자체 절전권고
	열차(운전, 승무)	• 주의단계에 따른 절전유도 안내방송
	차량기지	• 개별 전열기 사용금지 • 휴게시간 조명 절전 • 통로조명등 1/2소등
	소속 사업소 및 본부	• 개별 전열기 사용금지 • 휴게시간 조명 절전 • 통로 조명등 1/2소등
경계 (Orange)	역(역무원)	• 역사내 시설물 운행 정지 등 • 절전에 따른 안내방송 실시 • 역사내 일부 조명 차단에 따른 여객안내실시 • 취약개소 순찰 실시 • 스크린 도어 수동개방 • 비상대응 유관기관(119) 및 인접역 상황 전파
	전기사업소(전기처)	• 순차별 부하차단 1단계 : 환기설비 가동중단 2단계 : 냉방설비 가동중단 3단계 : 임대매장 전력차단 4단계 : E/V(E/S) 및 조명설비1/2차단
	관제센터(전력, 운전)	• 전력상황 모니터링 및 전 소속 상황 전파 • 관련기관 협조 및 연락체계 강화 • 이례사항 대비(열차, 역사 전달)
	열차(운전, 승무)	• 안내방송 실시 • 객실 내 격등제 시행 • 냉·난방설비 탄력적 운영
	차량기지	• 차량검수/정비 탄력적 운영 • 탄력운영에 따른 절전시행 • 불용공간 소등

구 분	조치주체	조치내용
경계 (Orange)	소속 사업소 및 본부	• 불용공간 소등 및 격등제 실시 • 승강기 운행정지 • 냉,난방설비 운행정지 • 정전대비 장비 확인 및 대응준비
심각 (Red)	역(역무원)	• 정전에 따른 안내방송 실시 • 상황전파 • E/V(E/S) 이용고객 갇힘 및 전도여부 확인 • 게이트 개방 • 승차권 발매 제한 조치 • 역사 내 질서 유지 및 안전사고 예방 • 열차대피승객 유도 지원
	전기사업소(전기처)	• 예비전력확보(비상발전기 등) • 역사내 전기실 및 현장 상황 파악, 보고
	관제 센터 (전력, 운전)	• 열차에 상황 통보 • 열차운행시격 조정 및 주의 통보 • 단·장기 정전에 따른 열차운행 지시 단기간 정전 시 차내 대기 지시 -장시간 정전 시 승객 대피 지시 (안전 조치 시행)
	열차(운전, 승무)	• 안내방송 실시 • 단·장기 정전에 따른 고객안내 • 냉·난방설비 운영중지
	차량기지	• 차량정비 일시중지(긴급차량만 정비) • 냉,난방설비 운행정지 • 사무실 조명 1/2소등
	소속 사업소 및 본부	• 실내·외등 1/2소등 • 비상전원설비 및 조명설비 등 운영

반별 임무

담당자	조 치 사 항	비 고
비상대응팀장 (팀장 : 관리역장) (역무팀장-야간,휴일)	• 비상대응팀 업무 총괄 • 승객 비상대응팀 출동지시 • 현장 상황 총괄 정리 및 각종 상황 보고 • 필요 시 대체수송(버스환승 등)방안 강구	
비상대응 총괄반 (반장 :부역장) (역무원-야간,휴일) - 서무담당 - 역무원 등	• 비상대응팀장 보좌 및 역할 대행 • 관제센터로부터 상황접수 및 전파 • 비상용품 확보 및 차량 탑재 지원 (생수,물수건,부채,랜턴,구급약 등) • 유관기관(119,의료) 지원요청 • 현장상황 파악 및 보고, 언론 등 대외기관 대응	
비상대응 출동반 (반장 : 역무팀장) (야간은 팀장 겸직) - 역무원 - 시설관리원 - 전기원 등	• 비상출동 회의 및 출동반원 안전교육 • 비상용품 승객 지급 및 사상자 구호활동 • 질서유지 및 환승 유도 안내(비상사다리, 비상계단) • 현장 접근로 차량출입 통제 및 유관기관 안내 • 현장 시설 피해 현황 파악 및 보고 • 비상대응팀장에게 활동보고(최초,중간,최종 등) (필요시 업무 변경 추가 가능)	

화재상황별 개인임무카드

직 명	임 무
당무역장(테러 발생시)	• 사고 접수 후 사고 급보 관제센터 및 안전처(야간 당직실)에 급보 안양관리역에 통보(비상대응팀 출동요청) 비상대응팀 출동까지 피해최소화를 위한 초동조치 이행 피해현황 파악 보고 인적 · 물적 피해상황을 파악하여 관리역등에 보고함. 임시복구지휘자 직 · 성명 파악보고 영업처 철)923-3204 · 경영인사처 철)923-3202 상황종료 보고 교통관제센터 및 비상대응 유관기관 상황종료 보고

직 명	임 무
역무원(테러 발생시)	• 사고 접수 후 사고 급보 국정원(111), 경찰서(112), 소방서(119) 등에 지원요청 인접역에 통보 타교통수단 이용 안내방송 실시(계속) 열차지연 안내방송 (계속) 여객 질서유지 타는 곳 승객 역사내로 대피 유도
사회복무요원 (테러 발생시)	• 역구내 여객통제 지원 기타 당무역장 지시에 따라 행동

직 명	임 무
당무역장(화재시)	• 사고 접수 후 사고 급보 관제센터 및 안전처(야간 당직실)에 급보 119, 112, 한림대병원 등에 지원요청 인접역(경마공원역, 과천역)에 통보 안양관리역에 통보(비상대응팀 출동요청) 안양전기,수원건축사업소(철:2681)에 통보 • 화재열차 도착 전 관제협의 운전관제사에게 사고현장 119구급대 출동여부 협의 사고현장 출동 시 금정역 로컬관제원에게 통보 안양관리역에 비상대응팀 출동여부 파악 119구급대(과천소방서) 사고현장 출동 협조 • 피해현황 파악 보고 인적 · 물적 피해상황을 파악하여 관리역등에 보고함. 임시복구지휘자 직·성명 파악보고 영업처 철)923-3204 · 경영인사처 철)923-3202 • 상황종료 보고 교통관제센터 및 비상대응 유관기관 상황종료 보고

직 명	임 무
역무원(화재시)	• 타교통수단 이용 안내방송 실시(계속) 열차지연 안내방송 (계속) 여객 질서유지 타는 곳 승객 역사내로 대피 유도
사회복무요원(화재시)	• 역구내 여객통제 지원 기타 당무역장 지시에 따라 행동

시간별 개인임무카드

당무역장	
10분 이내	• 사고지점을 향하는 열차 긴급정차 통보 관제센터에 사고급보, 열차방호, 신호정지 • 긴급지원요청 본부, 관리역, 철도청장, 111(테러)등 • 사상사고시 인명구조 및 고객 대피 • 초기대응팀 출동시 초기대응팀장 임무수행
30분 이내	• 관제센터(본부)에 중간보고 피해상황 및 사고현장상황 • 사상사고시 인명구조 및 고객 대피 • 비상대응팀과 연락체계 확보 • 병원안내 지원요청시 병원출동 지원
1시간 이내	• 관제센터(본부)에 중간보고 피해상황 및 사고현장상황 • 역 안내방송 시행 및 안내문게시 -열차운행정보 및 대체교통수단 안내 등
2시간 이내	• 관제센터(본부)에 중간보고 피해상황 및 사고현장상황 • 역 안내방송 시행 및 안내문게시 열차운행정보 및 대체교통수단 안내 등

역무장	
10분 이내	• 긴급지원요청 119,112, 병원, 인접역(소)등 유관기관 • 초기대응팀 출동시 당무역장 임무수행
30분 이내	• 역 안내방송 시행 및 안내문 게시 열차운행정보 및 대체교통수단 안내 등 승차권발매 중지 조치 화재시 승강기 사용중지 조치 -비상출입문 및 게이트 개방 조치
1시간 이내	• 역 안내방송 시행 및 안내문 게시 열차운행정보 및 대체교통수단 안내 등 승차권 반환 및 환불내용 안내 등
2시간 이내	• 역 안내방송 시행 및 안내문 게시 열차운행정보 및 대체교통수단 안내 등 승차권 반환 및 환불내용 안내 등

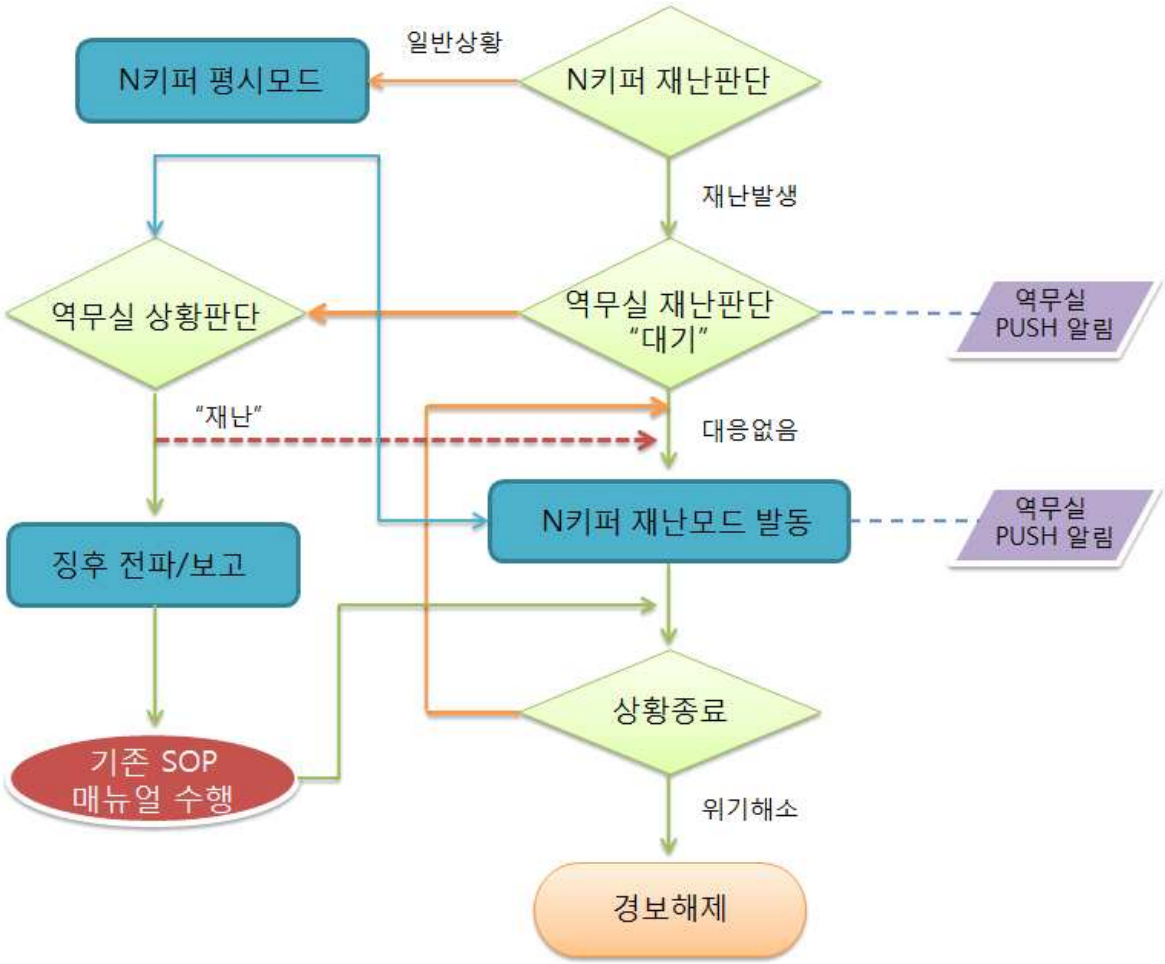
사회복무요원	
10분 이내	• 당무역장 지시에 따라 지원활동 사상자 구호 최우선, 여객대피안내 초기대응 임무수행
30분 이내	• 고객안내 및 지원업무 열차운행정보 및 대체교통수단 안내 등 119, 112 도착시 현장안내 화재시 승강기 사용중지 조치(계단안내) 비상출입문 및 게이트 개방 조치
1시간 이내	• 고객안내 및 지원업무 열차운행정도 및 대체교통수단 안내 등 고객 대피안내 및 후송지원 등
2시간 이내	• 역 안내방송 시행 및 안내문 게시 열차운행정보 및 대체교통수단 안내 등 고객 대피안내 및 후송지원 등

초기대응팀장	
10분 이내	• 사고접수 즉시 현장출동 현장도착시 관제통보 • 초기대응팀 비상연락망 가동 • 안양역 비상대응팀 출동 요청
30분 이내	• 피해상황 및 현장상황 파악 • 사상자 구호 및 승객대피 유도 • 사고현장 질서유지 • 사상자 병원후송시 동행 현황파악 및 안내(초기대응팀 파견) • 필요시 인근역장 병원안내 지원요청
1시간 이내	• 사고현장 질서유지 • 피해상황 팍 • 지역사고수습본부 인계
2시간 이내	• 지역사고수습본부 지시사항 이행

초기대응팀	
10분 이내	• 초기대응팀 출동지시 접수 • 사고현장으로 출동
30분 이내	• 사고현장 질서유지 • 사상자 구호 • 승객대피 유도 • 응급환자 병원동해
1시간 이내	• 사고현장 질서유지 • 연계버스 이용시 승객안내
2시간 이내	• 지역사고수습본부 지시사항 이해

다. 스마트N키퍼 연동 SOP 개발

재난(화재, 테러)발생 시 스마트N키퍼는 스스로 재난에 대하여 판단하여 역무실의 역무원에게 “알람”을 주게되며, 이 알람을 받은 역무원이 재난에 대한 상황판단 이후 적절한 조치를 취하여야 한다. 하지만 N키퍼의 알람에도 일정 시간(약 30초)동안 아무런 조치가 없을 경우 N키퍼는 스스로 재난모드를 발동하여 승객에게 알람 및 대피유도를 하도록 되어있으며, 자세한 흐름도는 다음과 같다.



[M2M 기반 지하공간(지하철) 재난대응 대화형 스마트 네트워크 시스템을 이용한 위기경보 발령체계도]

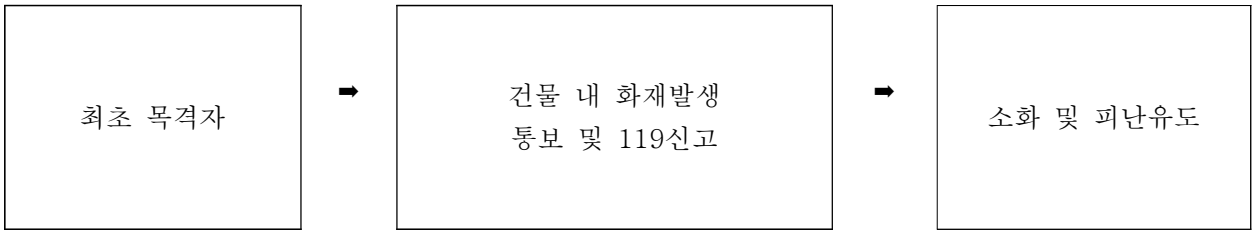
기존의 SOP에서 철도, 방재전문가와의 회의를 통해 스마트N키퍼의 역할과 기능을 고려한 SOP 베타버전을 개발하였으며 내용은 다음과 같다.

- 재난(화재,테러)상황시 행동요령 및 피난계획

□ 대공원역

- 재난(화재,테러)상황시 행동요령

대공원역에 근무・거주자는 평소 화재를 감지하는데 신경을 써야하며, 누구든지 화재의 발생사실을 최초로 목격하는 자는 119신고, 건물 내 화재발생 통보, 초기소화, 건물 내 거주자피난유도 등의 조치를 취하여야 한다. 스마트N키퍼를 통해 전달되는 재난의 경보를 감지하여 재난발생 유무를 확인하고 이에따른 조치를 취하여야 한다.



화재발생 사실이 건물 내에 전파되면 평소 편성·운영 중인 자위소방대 반별로 개별임무를 즉각 수행하여야 한다.
소방안전관리자는 사업장별 실정에 적합한 세부행동요령을 별도로 정할 수 있다.

- 피난계획

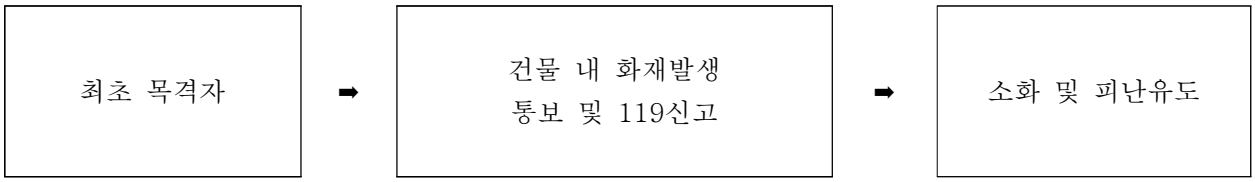
▪ 맞이방/대합실(지하 1층)
안내방송 및 스마트N키퍼(직원)의 유도에 따라 신속하게 출구 (1번-6번)를 통해 역사 외부로 대피한다. 이 경우 노약자 및 장애인이 신속하게 대피할 수 있도록 각 출구 출입문 전부를 완전히 열어둔 상태로 유지한다.

▪ 승강장(지하 2층)
1차 : 안내방송 및 스마트N키퍼(직원)의 유도에 따라 화재나 테러가 발생한 위치를 확인하고 이로부터 대피한다. 이때, 승강장 계단(E/S 포함)을 이용하여 맞이방/대합실 이동 후 각 출구를 통해 역사 외부로 대피한다. 이 경우 게이트 및 비상게이트를 개방하여 신속히 대피할 수 있도록 한다.
2차 : 대형화재로 1차 대피가 불리할 경우 직원과 스마트N키퍼의 유도에 따라 타는 곳 남쪽(과천역방면) 계단을 이용하여 차량기지를 통해 역사 외부로 대피한다.
화재발생시 자위소방대 대피유도반의 「장애인 노약자 피난보조인」 으로 임무를 부여받은 자는 장애인 및 노약자의 대피유도를 최우선으로 돕는다.

- 재난(화재,테러)상황시 행동요령 및 피난계획
□ 개포동역

- 재난(화재,테러)상황시 행동요령

개포동역에 근무·거주자는 평소 화재/테러를 감지하는데 신경을 써야하며, 누구든지 화재의 발생사실을 최초로 목격하는 자는 119신고, 건물 내 화재발생 통보, 초기소화, 건물 내 거주자피난유도 등의 조치를 취하여야 한다. 또한, 스마트N키퍼를 통해 전달되는 재난의 경보를 감지하여 재난발생 유무를 확인하고 이에따른 조치를 취하여야 한다.



화재발생 사실이 건물 내에 전파되면 평소 편성·운영 중인 자위소방대 반별로 개별임무를 즉각 수행하여야 한다.

소방안전관리자는 사업장별 실정에 적합한 세부행동요령을 별도로 정할 수 있다.

- 피난계획

▪ 맞이방/대합실(지하 1층)

안내방송 및 스마트N키퍼(직원)의 유도에 따라 신속하게 출구 (1번-8번)를 통해 역사 외부로 대피한다. 이 경우 노약자 및 장애인이 신속하게 대피할 수 있도록 각 출구 출입문 전부를 완전히 열어둔 상태로 유지한다.

▪ 중간층(지하 3층)

안내방송 및 스마트N키퍼(직원)의 유도에 따라 화재나 테러가 발생한 위치를 확인하고 이로부터 대피한다. 이때, 화재/테러가 발생한 위치에 따라 승강장 계단(E/S 포함)을 이용하여 맞이방/대합실 이동할지, 혹은 지하4층 승강장으로 이동하여 선로를 통해 대피할지 결정할 수 있도록 한다.

▪ 승강장(지하 4층)

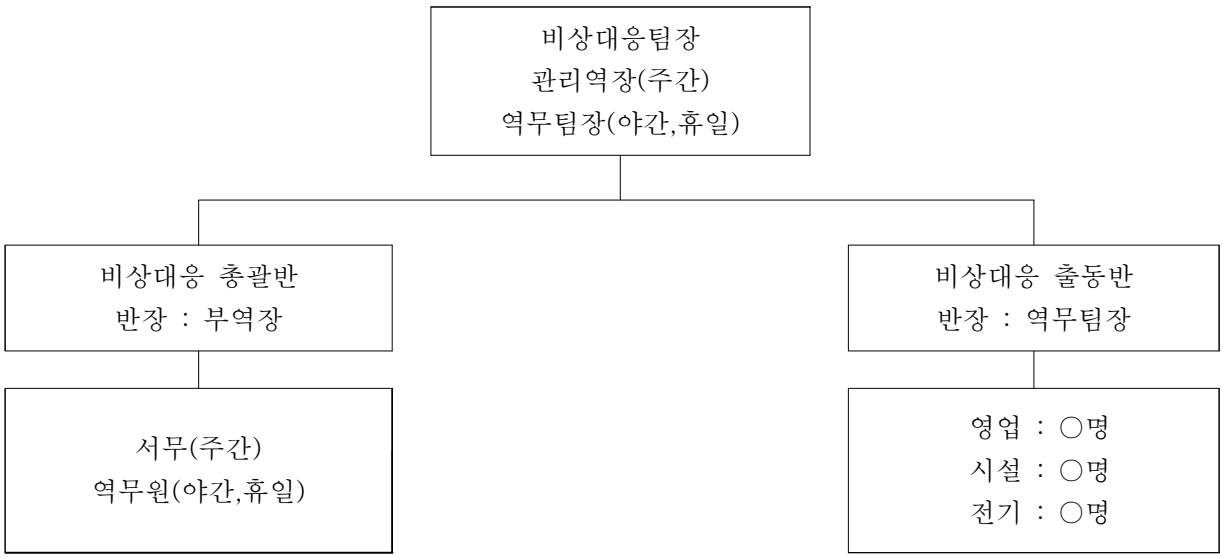
1차 : 안내방송 및 스마트N키퍼(직원)의 유도에 따라 화재나 테러가 발생한 위치를 확인하고 이로부터 대피한다. 이때, 승강장 계단(E/S 포함)을 이용하여 맞이방/대합실 이동 후 각 출구를 통해 역사 외부로 대피한다. 이 경우 게이트 및 비상게이트를 개방하여 신속히 대피할 수 있도록 한다.

2차 : 대형화재로 1차 대피가 불리할 경우 직원과 스마트N키퍼의 유도에 따라 타는 곳 동쪽(대모산입구역방면) 계단을 이용하여 차량기지를 통해 역사 외부로 대피한다.

화재발생시 자위소방대 대피유도반의 「장애인 노약자 피난보조인」 으로 임무를 부여받은 자는 장애인 및 노약자의 대피유도를 최우선으로 돕는다.

소방대 조직표

□ 대공원역/개포동역



- 위기정보 단계별 조치사항

구 분	조치주체	조치내용
주의 (Yellow)	역(역무원)	• 접수, 보고/전파 • 역사내 임대매장 정전대비 상황 통보 • 정전 대비 장비 및 임무 확인
	전기사업소(전기처)	• 현장출동 • 상황파악 및 보고 • 예비전원 설비 집중 점검
	관제센터(전력, 운전)	• 전력상황 모니터링 강화 • 역사 자체 절전권고
	열차(운전, 승무)	• 주의단계에 따른 절전유도 안내방송
	차량기지	• 개별 전열기 사용금지 • 휴게시간 조명 절전 • 통로조명등 1/2소등
	소속 사업소 및 본부	• 개별 전열기 사용금지 • 휴게시간 조명 절전 • 통로 조명등 1/2소등
경계 (Orange)	역(역무원)	• 역사내 시설물 운행 정지 등 • 절전에 따른 안내방송 실시 • 역사내 일부 조명 차단에 따른 여객안내실시 • 취약개소 순찰 실시 • 스크린 도어 수동개방 • 비상대응 유관기관(119) 및 인접역 상황 전파
	전기사업소(전기처)	• 순차별 부하차단

구 분	조치주체	조치내용
		1단계 : 환기설비 가동중단 2단계 : 냉방설비 가동중단 3단계 : 임대매장 전력차단 4단계 : E/V(E/S) 및 조명설비1/2차단
	관제센터(전력, 운전)	• 전력상황 모니터링 및 전 소속 상황 전파 • 관련기관 협조 및 연락체계 강화 • 이례사항 대비(열차, 역사 전달)
	열차(운전, 승무)	• 안내방송 실시 • 객실 내 격등제 시행 • 냉·난방설비 탄력적 운영
	차량기지	• 차량검수/정비 탄력적 운영 • 탄력운영에 따른 절전시행 • 불용공간 소등
	소속 사업소 및 본부	• 불용공간 소등 및 격등제 실시 • 승강기 운행정지 • 냉,난방설비 운행정지 • 정전대비 장비 확인 및 대응준비
심각 (Red)	역(역무원)	• 정전에 따른 안내방송 실시 • 상황전파 • E/V(E/S) 이용고객 갇힘 및 전도여부 확인 • 게이트 개방 • 승차권 발매 제한 조치 • 역사 내 질서 유지 및 안전사고 예방 • 열차대피승객 유도 지원
	전기사업소(전기처)	• 예비전력확보(비상발전기 등) • 역사내 전기실 및 현장 상황 파악, 보고
	관제 센터 (전력, 운전)	• 열차에 상황 통보 • 열차운행시격 조정 및 주의 통보 • 단·장기 정전에 따른 열차운행 지시 단기간 정전 시 차내 대기 지시 -장시간 정전 시 승객 대피 지시 (안전 조치 시행)
	열차(운전, 승무)	• 안내방송 실시 • 단·장기 정전에 따른 고객안내 • 냉·난방설비 운영중지

구 분	조치주체	조치내용
[redacted]	차량기지	• 차량정비 일시중지(긴급차량만 정비) • 냉,난방설비 운행정지 • 사무실 조명 1/2소등
	소속 사업소 및 본부	• 실내·외등 1/2소등 • 비상전원설비 및 조명설비 등 운영

- 반별 임무

담당자	조 치 사 항	비 고
비상대응팀장 (팀장 : 관리역장) (역무팀장-야간,휴일)	• 비상대응팀 업무 총괄 • 승객 비상대응팀 출동지시 • 현장 상황 총괄 정리 및 각종 상황 보고 • 필요 시 대체수송(버스환승 등)방안 강구	
비상대응 총괄반 (반장 :부역장) (역무원-야간,휴일) - 서무담당 - 역무원 등	• 비상대응팀장 보좌 및 역할 대행 • 관제센터로부터 상황접수 및 전파 • 비상용품 확보 및 차량 탑재 지원 (생수,물수건,부채,랜턴,구급약 등) • 유관기관(119,의료) 지원요청 • 현장상황 파악 및 보고, 언론 등 대외기관 대응	
비상대응 출동반 (반장 : 역무팀장) (야간은 팀장 겸직) - 역무원 - 시설관리원 - 전기원 등	• 비상출동 회의 및 출동반원 안전교육 • 비상용품 승객 지급 및 사상자 구호활동 • 질서유지 및 환승 유도 안내(비상사다리, 비상계단) • 현장 접근로 차량출입 통제 및 유관기관 안내 • 현장 시설 피해 현황 파악 및 보고 • 비상대응팀장에게 활동보고(최초,중간,최종 등) (필요시 업무 변경 추가 가능)	

- 재난상황별 개인임무카드

직 명	임 무
당무역장 (테러 발생시)	• 사고 접수 후 사고 급보 관제센터 및 안전처(야간 당직실)에 급보 안양관리역/수서역에 통보(비상대응팀 출동요청) 비상대응팀 출동까지 피해최소화를 위한 초동조치 이행 피해현황 파악 보고 인적·물적 피해상황을 파악하여 관리역등에 보고함. 임시복구지휘자 직·성명 파악보고 교통관제센터 및 비상대응 유관기관 상황종료 보고 • 스마트N키퍼 테러 상황 주시 및 “알람”발생 시 상황대응
역무원 (테러 발생시)	• 사고 접수 후 사고 급보 국정원(111), 경찰서(112), 소방서(119) 등에 지원요청 인접역에 통보 타교통수단 이용 안내방송 실시(계속) 열차지연 안내방송 (계속) 여객 질서유지 타는 곳 승객 역사내로 대피 유도 • 스마트N키퍼 테러 상황 주시 및 “알람”발생 시 상황대응
사회복무요원 (테러 발생시)	• 역구내 여객통제 지원 기타 당무역장 지시에 따라 행동

직 명	임 무
당무역장(화재시) 대공원역/개포동역	• 사고 접수 후 사고 급보 관제센터 및 안전처(야간 당직실)에 급보 119, 112, 한림대병원/삼성서울병원 등에 지원요청 인접역(경마공원역, 과천역/대치역, 구룡역)에 통보 안양관리역/수서역에 통보(비상대응팀 출동요청) 안양전기,수원건축사업소/ 수도권동부본부에 통보 **• 스마트N키퍼 화재발생 상황 주시 및 “알람”발생 시 상황 대응** • 화재열차 도착 전 관제협의 운전관제사에게 사고현장 119구급대 출동여부 협의 사고현장 출동 시 로컬관제원에게 통보 안양관리역/수서역에 비상대응팀 출동여부 파악 119구급대(과천소방서/수서119안전센터) 사고현장 출동 협조 • 피해현황 파악 보고 인적·물적 피해상황을 파악하여 관리역등에 보고함. 임시복구지휘자 직·성명 파악보고 • 상황종료 보고 교통관제센터 및 비상대응 유관기관 상황종료 보고
역무원(화재시)	• 타교통수단 이용 안내방송 실시(계속) 열차지연 안내방송 (계속) 여객 질서유지 타는 곳 승객 역사내로 대피 유도 **• 스마트N키퍼 화재발생 상황 주시 및 “알람”발생 시 상황 대응**
사회복무요원 (화재시)	• 역구내 여객통제 지원 기타 당무역장 지시에 따라 행동

- 시간별 개인임무카드

당무역장	
2분 이내	• **스마트N키퍼 재난모드 전환, 유도 대피로 확인/조정**
10분 이내	• 사고지점을 향하는 열차 긴급정차 통보 관제센터에 사고급보, 열차방호, 신호정지 • 긴급지원요청 본부, 관리역, 철도청장, 111(테러)등 • 사상사고시 인명구조 및 고객 대피 • 초기대응팀 출동시 초기대응팀장 임무수행
30분 이내	• 관제센터(본부)에 중간보고 피해상황 및 사고현장상황 • 사상사고시 인명구조 및 고객 대피 • 비상대응팀과 연락체계 확보 • 병원안내 지원요청시 병원출동 지원
1시간 이내	• 관제센터(본부)에 중간보고 피해상황 및 사고현장상황 • 역 안내방송 시행 및 안내문게시 • 열차운행정보 및 대체교통수단 안내 등
2시간 이내	• 관제센터(본부)에 중간보고 피해상황 및 사고현장상황 • 역 안내방송 시행 및 안내문게시 열차운행정보 및 대체교통수단 안내 등

역무장	
2분 이내	• **스마트N키퍼 재난모드 전환, 유도 대피로 확인/조정**
10분 이내	• 긴급지원요청 119,112, 병원, 인접역(소)등 유관기관 • 초기대응팀 출동시 당무역장 임무수행
30분 이내	• 역 안내방송 시행 및 안내문 게시 열차운행정보 및 대체교통수단 안내 등 승차권발매 중지 조치 화재시 승강기 사용중지 조치 • 비상출입문 및 게이트 개방 조치
1시간 이내	• 역 안내방송 시행 및 안내문 게시 열차운행정보 및 대체교통수단 안내 등 승차권 반환 및 환불내용 안내 등
2시간 이내	• 역 안내방송 시행 및 안내문 게시 열차운행정보 및 대체교통수단 안내 등 승차권 반환 및 환불내용 안내 등

사회복무요원	
2분 이내	• **스마트N키퍼 재난모드 전환, 유도 대피로 확인/조정**
10분 이내	• 당무역장 지시에 따라 지원활동 사상자 구호 최우선, 여객대피안내 초기대응 임무수행
30분 이내	• 고객안내 및 지원업무 열차운행정보 및 대체교통수단 안내 등 119, 112 도착시 현장안내 화재시 승강기 사용중지 조치(계단안내) 비상출입문 및 게이트 개방 조치
1시간 이내	• 고객안내 및 지원업무 열차운행정도 및 대체교통수단 안내 등 고객 대피안내 및 후송지원 등
2시간 이내	• 역 안내방송 시행 및 안내문 게시 열차운행정보 및 대체교통수단 안내 등 고객 대피안내 및 후송지원 등

초기대응팀장	
2분 이내	• **스마트N키퍼 재난모드 전환, 유도 대피로 확인/조정**
10분 이내	• 사고접수 즉시 현장출동 현장도착시 관제통보 • 초기대응팀 비상연락망 가동 • 비상대응팀 출동 요청
30분 이내	• 피해상황 및 현장상황 파악 • 사상자 구호 및 승객대피 유도 • 사고현장 질서유지 • 사상자 병원후송시 동행 현황파악 및 안내(초기대응팀 파견) • 필요시 인근역장 병원안내 지원요청
1시간 이내	• 사고현장 질서유지 • 피해상황 팍 • 지역사고수습본부 인계
2시간 이내	• 지역사고수습본부 지시사항 이행

초기대응팀	
2분 이내	• **스마트N키퍼 재난모드 전환, 유도 대피로 확인/조정**
10분 이내	• 초기대응팀 출동지시 접수 • 사고현장으로 출동
30분 이내	• 사고현장 질서유지 • 사상자 구호 • 승객대피 유도 • 응급환자 병원동해
1시간 이내	• 사고현장 질서유지 • 연계버스 이용시 승객안내
2시간 이내	• 지역사고수습본부 지시사항 이해

2. 해외 지하공간(지하철) 비상대응 연구 사례분석

1) 내용

철도 시설물에서는 드물지만 화재가 발생을 하고 있습니다. 2003년 대구 도시철도 1호선 중앙로역에서 방화로 인하여 대형 참사가 발생하였습니다. 이와 같이 치명적인 잠재력을 가지고 있기 때문에 지하시설(지하철)에서의 안전한 탈출 및 올바른 인도 신도 신호등에 관한 연구가 필요하다. 해외 지하공간(지하철) 비상대응 연구 분석에서는 University of Lund에서 진행한 실험 및 실제 현장에서 사용되는 내용을 위주로 분석을 진행 할 예정이며, 연구 분야는 크게 표지판 그리고 플래싱 라이트 2 가지 부분으로 분류할 수 있다.

가) 정보 표지판 내용

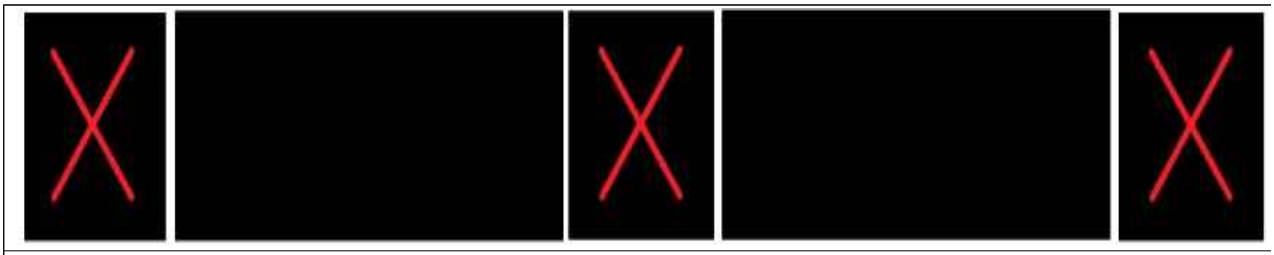
주위 환경의 이해는 의사 결정을 위해 필요한 기본적인 사항이다. 이러한 생각은 환경의 전체적인 지각과 깊은 연관을 가지고 있다. 표지판은 이러한 개념 정리에 있어서 승객 또는 이용자에게 환경을 인식하는 능력 향상에 많은 도움을 준다. 이와 같이 표지판은 일반적인 의미 의도에서처럼 긴급 상황에서도 승객 또는 이용자에게 많은 도움을 준다.

표지판 : 내용, 표지판의 크기, 색상 그리고 그림 문자의 사용 또는 라이트
비상 출구 정문 배열 : 색상
음향 시스템 : 음성 메시지 또는 위험 경고 신호
본 장에서는 교통 정보 표지판, 가변 정보판에 대해서 분석 할 예정이며 이는 실시간 정보를 터널 이용자에게 제공되는 기술이다. 이런 기술은 패널을 사용하여 다양한 정보를 터널의 사용자에게 전달 할 목적으로 사용된다. 이러한 기술을 바탕으로 정보가 제공 되었을 때 정보를 제공받는 터널 사용자의 행동 유도성에 관해서도 정리 할 예정이다. 행동 유도성은, 탈출 시스템을 분석하여 탈출 방법의 보다 효과적으로 파악할 수 있다. 행동 유도성은 아래와 같이 크게 4가지 분야로 나눌 수 있다.

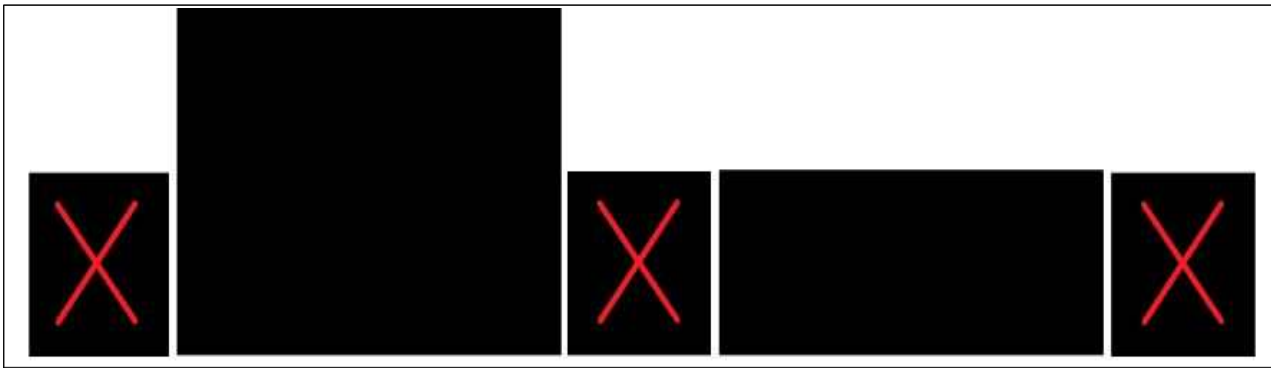
감각 행동 유도성 : 느낌 또는 관찰
인지 행동 유도성 : 이해
물리적 행동 유도성 : 어떤 행동을 하거나 사용
기능 행동 유도성 : 목표 달성을 위해

□ 교통 정보 표지판

교통 정보 표지판의 경우 크기에 따라 정보 이용자에서 혼선 및 정확하게 정보를 전달하기 힘든 점이 있다. 아래의 사진처럼 같은 크기의 패널을 사용하거나 또는 응급 정보의 제공에 있어서는 사이즈를 변경하였다.



동일 크기의 패널



응급 패널 사이즈를 확대

[정보 표지판의 도식 표현. 두가지 패널에 있어서 위에 패널의 경우 동일한 크기, 아래의 경우 응급대피 패널의 크기를 확대]

이러한 패널 크기에 따라서 정보 이용자의 시선을 사로잡을 뿐 아니라 색상에 의해서도 정보 제공 이용자에게 보다 효과적으로 다가갈 수 있다. 색상의 차이에 따라서 운전자에게 보다 효과적으로 정보를 제공 할 수 있다. 색상의 선택은 다른 타입의 문자 또는 그림문자 보다 더 효율적인 효과를 가지고 있다. 이는 좀 더 쉽게 인식할 수 있다는 의미이지만 문자나 숫자 없이는 효과적일 수가 없다는 것을 의미하며 실험에 의하면 지역적인 차이를 두기도 한다. 미국이나 중국의 경우 황색 또는 녹색이 빨간색에 비해 정보 이용자들에게 빠르고 정확하게 인지되는 것으로 확인이 되었다.

또한 패널에서 제공되는 메시지 정보의 양과 메시지의 길이 또는 크기 모양과 함축 정도가 중요한 요소이다. 또한 정보가 노출되는 시간에 따라서 정보의 제공의 적당한 정도를 확인 할 수가 있다. 2개 프레임의 메시지가 정보 제공이 되는 경우 한 프레임당 2초 또는 4초가 가장 적당한 것으로 현장 연구에서 결론 되었다.
하나의 패널에서는 3가지의 정보는 아래와 같은 3가지의 정보는 포함하고 있다.
문제(터널에서 어떤 사고가 발생)
어떤 지점에서 사고가 발생
사고에 따른 운전자의 행동 방법
위와 같은 3가지의 정보가 메시지로 패널을 통해서 운전자 또는 정보 이용자에게 제공이 되어야 한다.
2줄 응급 메시지
Line 1: “STANNA MOTORN” (Swedish) - “STOP ENGINE” (English)
Line 2: “UTRYM TUNNELN” (Swedish) - “LEAVE TUNNEL” (English)

1줄 응급 메시지
Line 2: “UTRYM TUNNELN” (Swedish) - “LEAVE TUNNEL” (English)



[정보 표지판의 두가지 예시, 황색의 2줄 메시지 및 흰색의 1줄 메시지]

패널 언어 사용에 있어서 좀 더 명확한 표현을 필요로 한다. 예를 들어 응급 상황에 있어서 “터널 밖으로 나가시오” 보다는 “터널에서 탈출 하시오” 라는 표현이 더 정확하다. 메시지 단어의 사이즈의 경우 명확하게 보이는 크기를 선택하여야 하며 적절한 분석 없이는 단어 사이즈의 변경을 권고하지 않는다.

아래의 사진과 같은 그림문자가 교통 정보 표지판에 사용된다.

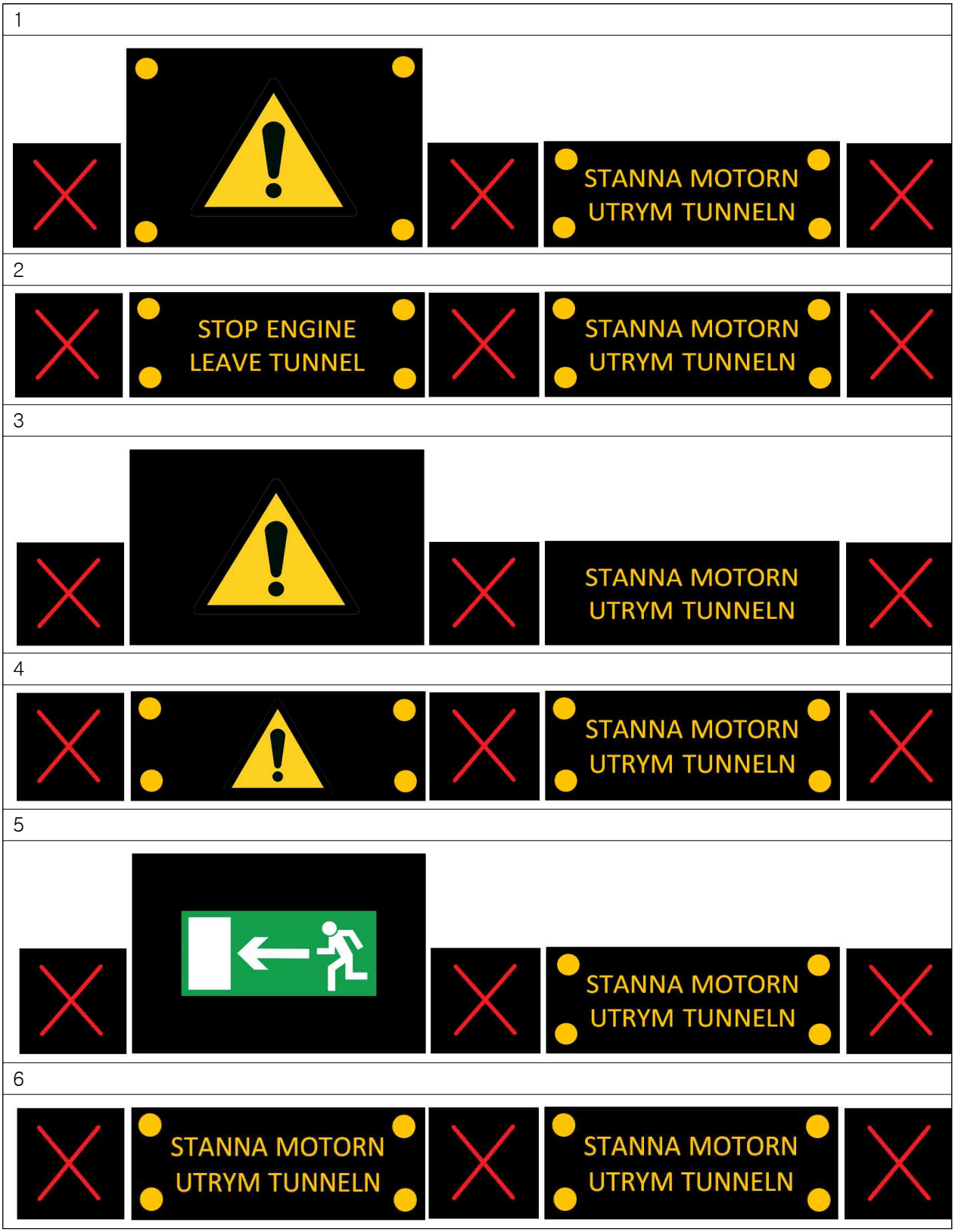


[교통 정보에 사용되는 그림문자]

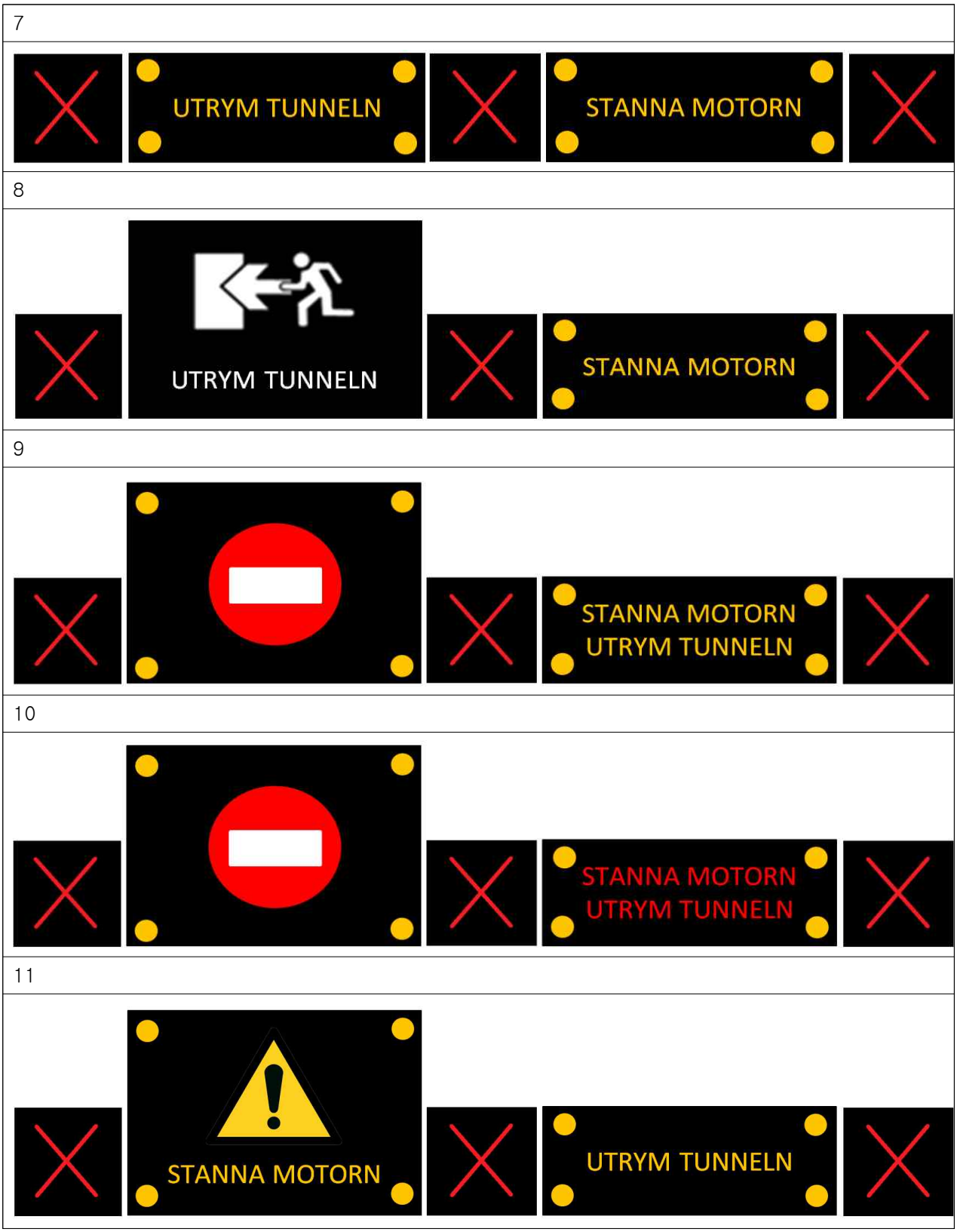
위에 종합한 정보를 바탕으로 여러 가지 방법으로 조합을 진행하여 실험을 진행한다.
-패널의 사이즈는 240x170 cm or 240x90 cm, 가장 작은 패널의 사이즈는 90x90 cm
-색상 배치
-메세지 디자인
-플래싱 라이트 효과
-그림문자

[교통 정보 표지판 실험에 선정된 조합]

No.	색상	메시지 (패널당 줄수)	플래싱 라이트 (패널당)	그림문자	패널 크기
1	검색/황색	0+2	4+4	경고	240x170 & 240x90
2	검정/황색	2+2	4+4	/	240x90 & 240x90
3	검정/황색	0+2	0+0	경고	240x170 & 240x90
4	검정/녹색/황색	0+2	2+2	경고	240x90 & 240x90
5	검정/황색	1+1	4+4	/	240x90 & 240x90
6	검정/황색	2+2	4+4	/	240x90 & 240x90
7	검정/황색	0+2	0+4	비상구	240x170 & 240x90
8	검정/흰색/황색	1+1	0+4	비상구	240x170 & 240x90
9	검정/빨간/황색	0+2	4+4	출입금지	240x170 & 240x90
10	검정/빨간	0+2	4+4	출입금지	240x170 & 240x90
11	검정/황색	1+1	4+4	경고	240x170 & 240x90



[교통 정보 표지판 실험 예비 리스트 (1-6)]



[교통 정보 표지판 실험 예비 리스트 (7-11)]

감각 행동 유도성

감각 행동 유도성은 교통 정보 표지판에 의해 운전자가 얼마나 집중하여 제공된 정보를 이용하는지에 따라서 결정된다. 이는 패널 위치, 크기, 색상 및 기호와 그 특징에 큰 영향을 받는다. 황색 색상의 글씨는 다른 색상에 비해서 집중력 및 반응 속도를 높이는 효과를 가지고 있으며, 빨간색 또는 흰색은 감각 행동 유도성을 낮추는 효과를 가지고 있다.

또한 플래싱 라이트는 운전자의 주의를 높이는 효과를 가지고 있는 것으로 확인 되었다. 비상 출구의 사이즈는 240x170 cm 크기가 감각 행동 유도성 증가에 도움을 주는 것으로 나타났다. 리뷰에서와 같이 1 줄과 2줄 메시지의 효과를 비교하면 2 줄 메시지가 1 줄 메시지에 비해서 더 빨리 읽혀지는 것으로 확인 되었다. 감각 행동 유도성의 경우 운전자가 짧은 시간에 인지하고 대응하기 때문에 교통 정보 표지판이 매우 크게 영향을 미치는 것으로 확인 되었다.

[감각 행동 유도성의 증가 또는 감소 기여 요인 (1-4)]

No.	감각 행동 유도성
1	+ 경고 사인은 TIS를 인지/찾기 쉽게 함 + 경고 사인은 TIS를 구별하기 쉽게 함 + 8개 황색 섬광등은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 구별하기 쉽게 함
2	+ 8개 황색 섬광등은 TIS를 인지/찾기 쉽게 함
3	+ 경고 사인은 TIS를 인지/찾기 쉽게 함 + 경고 사인은 TIS를 구별하기 쉽게 함 + 하나의 큰 패널은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 구별하기 쉽게 함
4	+ 경고 사인은 TIS를 인지/찾기 쉽게 함 + 경고 사인은 TIS를 구별하기 쉽게 함 + 8개 황색 섬광등은 TIS를 인지/찾기 쉽게 함 - 작아진 그림 문자는 TIS를 인지/찾기 어렵게 함 - 작아진 그림 문자는 TIS를 구별하기 어렵게 함

[감각 행동 유도성의 증가 또는 감소 기여 요인 (5-9)]

No.	감각 행동 유도성
5	+ 비상구 그림문자는 TIS를 인지/찾기 쉽게 함 + 비상구 그림문자는 TIS를 구별하기 쉽게 함 + 하나의 패널에 섬광등은 패널을 인지/찾기 쉽게 함 - 하나의 패널에 섬광등이 있어 다른 패널을 인지/찾기 어렵게 함 + 하나의 큰 패널은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 구별하기 쉽게 함
6	+ 8개의 황색 섬광등은 TIS를 인지/찾기 쉽게 함
7	+ 8개의 황색 섬광등은 TIS를 인지/찾기 쉽게 함 - 한줄 문자는 문자를 인지/찾기 어렵게 함 - 한줄 문자는 문자를 구별하기 어렵게 함
8	+ 비상구 그림문자는 TIS를 인지/찾기 쉽게 함 + 비상구 그림문자는 TIS를 구별하기 쉽게 함 + 하나의 패널의 섬광등은 패널을 인식/찾기 쉽게 함 - 하나의 패널에만 섬광등이 있어 다른 패널을 인식/찾기 어렵게 함 + 하나의 큰 패널을 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널을 TIS를 구별하기 쉽게 함 - 흰색으로 쓰여진 글자는 TIS를 인지/찾기 어렵게 함 - 흰색으로 쓰여진 글자는 TIS를 구별하기 어렵게 함 - 작아진 그림문자는 TIS를 인지/찾기 어렵게 함 - 작아진 그림문자는 TIS를 구별하기 어렵게 함 - 한줄의 문자는 문자를 인지/찾기 어렵게 함 - 한줄의 문자는 문자를 구별하기 어렵게 함
9	+ “출입금지” 사인은 TIS를 인지/찾기 쉽게 함 + “출입금지” 사인은 TIS를 구별하기 쉽게 함 + 8개의 황색 섬광등은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 구별하기 쉽게 함

[감각 행동 유도성의 증가 또는 감소 기여 요인 (10-11)]

No.	감각 행동 유도성
10	+ “출입금지” 사인은 TIS를 인지/찾기 쉽게 함 + “출입금지” 사인은 TIS를 구별하기 쉽게 함 + 8개의 황색 섬광등은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 구별하기 쉽게 함 - 적색의 문자는 TIS를 인지/찾기 어렵게 함 - 적색의 문자는 TIS를 구별하게 어렵게 함
11	+ 경고 사인은 TIS를 인지/찾기 쉽게 함 + 경고 사인은 TIS를 구별하기 쉽게 함 + 8개의 황색 섬광등은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 인지/찾기 쉽게 함 + 하나의 큰 패널은 TIS를 구별하기 쉽게 함 - 한줄의 문자는 문자를 인지/찾기 어렵게 함 - 한줄의 문자는 문자를 구별하기 어렵게 함 - 작아진 그림문자는 TIS를 인지/찾기 어렵게 함 - 작아진 그림문자는 TIS를 구별하기 어렵게 함.

인지 행동 유도성

교통 정보 표지판의 경우 운전자가 이해할 수 있는 정보는 제공하는 패널의 효율과 관련이 된다. 텍스트, 그림 기호, 플래싱 라이트 및 메시지 디자인과 조합에 따라서 달라집니다. 교통 정보 표지판 5,8 그리고 9번의 경우 운전자에게 있어서 그림문자의 색상이 달라 혼란을 주는 것으로 확인 되었다. 위와 같은 표지판은 사용하지 않는 것이 좋으나. 부정적인 효과를 그림 문자 사용을 통해 균형을 맞출 수 있는 지를 확인하기 위해 실험은 필요로 한다. 특이한 색상의 그림 문자는 행동 유도성을 감소시키며 흰색의 글자 역시 감소의 효과를 가지고 있다. 또한 황색 또는 빨간색상의 비교를 통해 두 색상은 운전자로 하여금 비상상황 및 긴급상황을 인지 못 할 수도 있다.

[인지 행동 유도성의 증가 또는 감소 기여 요인 (1-7)]

No.	인지 행동 유도성
1	+ 익숙한 그림문자는 경고를 더 잘 전달함 + 그림문자와 섬광등의 조합은 메시지의 이해를 향상시킴 + 8개의 황색 섬광등은 경고를 더 잘 전달함
2	+ 스웨덴어를 못하는 사람들에게 더 나은 메시지 이해 + 8개의 황색 섬광등은 경고를 더 잘 전달함 - 두 패널의 다른 언어 메시지는 영어를 못하는 사람들에게 혼란을 일으킴
3	+ 익숙한 그림문자는 경고를 더 잘 전달함 + 그림문자와 섬광등의 조합은 메시지의 이해를 향상시킴
4	+ 익숙한 그림문자는 경고를 더 잘 전달함 + 그림문자와 섬광등의 조합은 메시지의 이해를 향상시킴 + 8개의 황색 섬광등은 경고를 더 잘 전달함
5	+ 익숙한 그림문자는 경고를 더 잘 전달함 + 그림문자와 섬광등의 조합은 메시지의 이해를 향상시킴 + 8개의 황색 섬광등은 경고를 더 잘 전달함 + 녹색은 안전하다는 메시지를 전달하는데 익숙한 색 - 하나의 패널에만 있는 섬광등은 메시지 이해를 감소시킴 - 패널에 두 가지의 다른 색상 (녹색 그림문자 와 황색 문자)은 혼란은 일으킴 - 다른 두 종류의 정보를 패널에 쓰는 것 (안전과 경고)은 혼란을 일으킴
6	+ 8개의 황색 섬광등은 경고를 더 잘 전달함 + 2개의 다른 패널에서 같은 경고 메시지를 반복하는 것은 경고를 더 잘 전달함
7	+ 8개의 황색 섬광등은 경고를 더 잘 전달함 - 2개의 패널에 다른 문자는 혼란을 일으킴

[인지 행동 유도성의 증가 또는 감소 기여 요인 (8-11)]

No.	인지 행동 유도성
8	+ 그림문자와 섬광등의 조합은 메시지의 이해를 향상시킴 + 8개의 황색 섬광등은 경고를 더 잘 전달함 - 하나의 패널에만 있는 섬광등은 메시지 이해를 감소시킴 - 흰색은 안전 메시지를 전달하기에 익숙하지 않은 색이고 메시지 이해를 감소 - 두 패널의 다른 두 문자는 혼란을 일으킴 - 패널에 두 가지 다른색의 사용 (흰색의 그림문자와 황색색의 문자)은 혼란을 일으킴 - 두 다른 종류의 정보를 패널에 사용 (안전과 경고)하는 것은 혼란을 일으킴
9	+ 8개의 황색 섬광등은 경고를 더 잘 전달함 + 익숙한 그림문자는 위험을 더 잘 전달함 - 두 개의 다른 정보를 전달하는 그림문자와 문자의 조합 (위험과 경고)은 혼란을 일으킴 - “출입금지” 사인의 사용은 터널을 나가라고 하는 메시지와 모순됨 - 두 패널의 다른 색상 사용 (문자나 사인은 적색이고 섬광등은 황색)은 혼란을 일으킴 - 패널에서 두 가지 색상의 사용 (문자 또는 사인은 적색이고 섬광등은 황색)은 혼란을 일으킴
10	+ 8개의 황색 섬광등은 경고를 더 잘 전달함 + 익숙한 그림문자는 위험을 더 잘 전달함 - 두 개의 다른 정보를 전달하는 그림문자와 문자의 조합 (위험과 경고)은 혼란을 일으킴 - “출입금지” 사인의 사용은 터널을 나가라고 하는 메시지와 모순됨 - 각각의 패널의 두 가지 다른 색상을 사용하는 것 (문자나 사인은 적색이고 섬광등은 황색)은 혼란을 일으킴
11	+ 익숙한 그림문자는 경고를 더 잘 전달함 + 그림문자와 섬광등의 조합은 메시지의 이해를 향상시킴 + 8개의 황색 섬광등은 경고를 더 잘 전달함 - 경고 그림문자의 위치와 터널을 나가라는 문자를 왼쪽에 배치하는 것은 메시지의 이해를 감소시킴 (운전자는 패널을 왼쪽에서 오른쪽으로 읽기 때문에)

물리적 행동 유도성

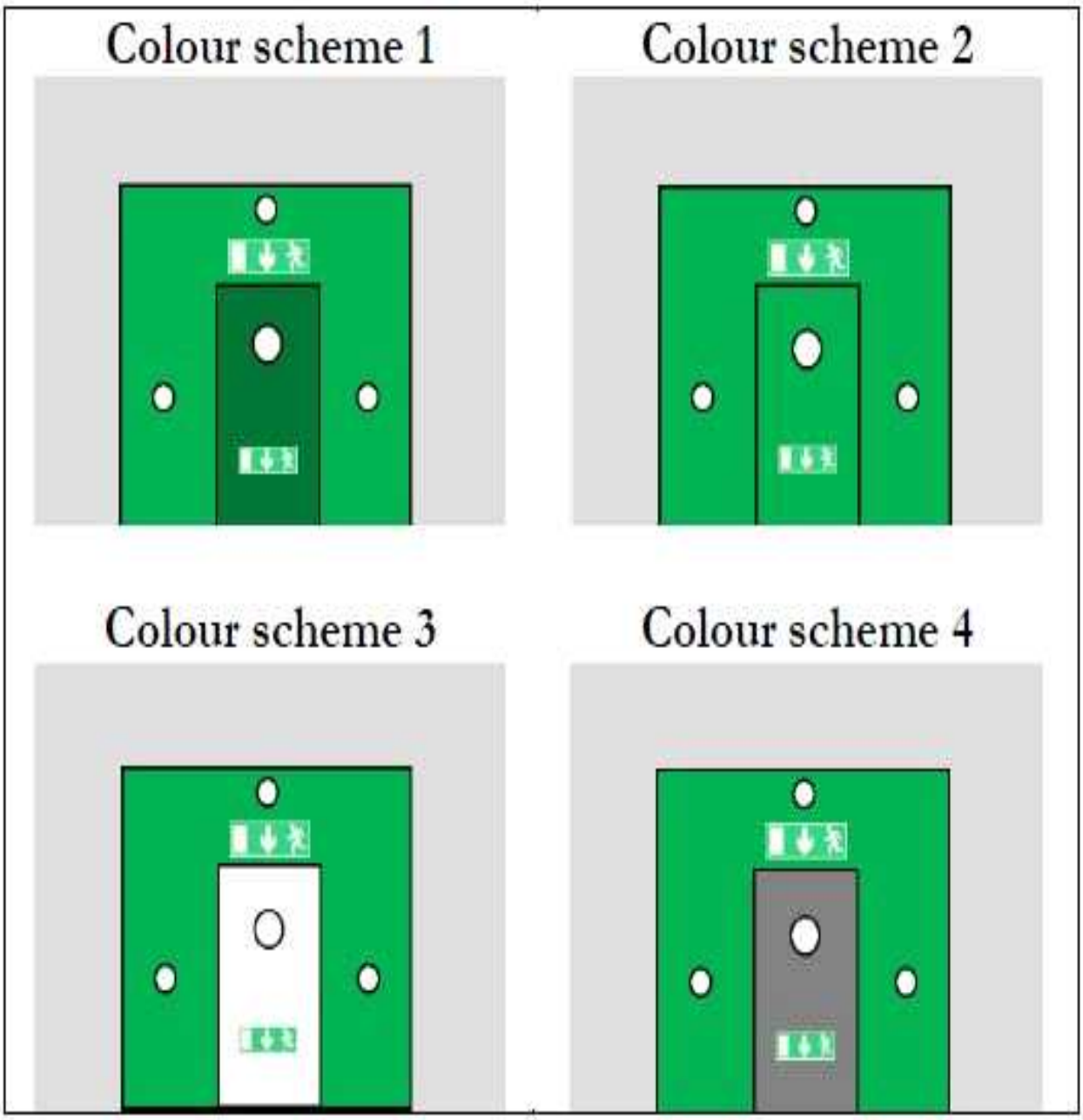
물리적 행동 유도성은 각자의 물리적인 지원이기 때문에 교통 정보 표지판과 물리적인 상호작용을 필요로 하지 않는다. 이러한 이유로 교통 정보 표지판의 분석에 있어서 물리적 행동 유도성은 해당 사항이 없다.

기능 행동 유도성

기능 행동 유도성은 모든 운전자가 터널에서의 목표와 연관이 되어 있다. 터널 사용자의 목표는 안전하게 극복하는 것이며 모든 시스템은 이를 고려하여 설계되어 있다. 이러한 이유로 기능 행동 유도성은 인지 및 감각 행동 유도성에 영향을 준다고 주장하고 있다. 이러한 요소는 감각 그리고 인지 행동 유도성의 증가 또는 감소 기여 요인의 표에서도 확인 할 수 있다. 기능 행동 유도성의 높은 점수는 다른 행동 유도성의 조합에 으로부터 결과를 확인 할 수 있다. 만약 다른 행동 유도성의 실패가 나타난다면, 기능 행동 유도성의 결과 역시 낮거나 실패로 이어질 것이다.

□ 비상 출구 색상 선택

비상 출구는 운전자가 찾기 쉽게 만들어져야 한다. 이 목표를 달성하기 위해서는 다른 유형의 행동 유도성이 고려되어야 한다. 예를 들면 색상 선택을 통해서 인지 행동 유도성을 강화하여야 한다. 색상은 인지 행동 유도성을 향상시켜, 비상 탈출구가 더 잘 보이는데 도움을 준다. 색상은 항상 인지 행동 유도성을 높여 제공 정보와 일치하도록 설계되어야 한다. 일반적으로 비상 탈출구의 색상은 녹색이며, 인지 행동 유도성을 높이기 위해서는 비상 탈출구 정문과 터널 벽면의 색상을 고려하는 것도 좋은 방법이다.
비상 탈출구의 추천된 색상은 “안전 녹색“ ”안전 녹색보다 어두운 녹색“, 흰색 또는 회색입니다. 다음 그림에서는 비상 탈출구 정문의 색상을 비교하고 있습니다. 주변 환경의 색상은 콘크리트 표준 색상인 회색이다.



[비상 탈출구로 가능한 4가지 색상]

감각 행동 유도성

적절한 색의 대비는 감각 행동 유도성을 높이기 위해 제공 되어야 한다. 비상 탈출구는 다른 2가지 색상에 비해서 뚜렷하게 구별되어야 한다. 아래 표와 같이 여러 가지 조합을 통해 감각 행동 유도성에 영향을 미치는 요소를 찾아 낼 수 있다. 고려될 변수는 색상의 대비 및 문의 색상이다.

[감각 행동 유도성의 증가 또는 감소 기여 요인 (1-4)]

No.	감각 행동 유도성
1	+ 문과 정면의 색상 대비로 문을 찾기가 쉬움 + 문과 정면을 색상의 대비로 문을 쉽게 구별 할 수 있음
2	- 문과 정면의 색상 대비가 부족하여 문을 찾기가 어려움 - 문과 정면을 색상 대비가 부족하여 문을 구별하기 없음
3	+ 문과 정면의 색상 대비로 문을 찾기가 쉬움 + 문과 정면을 색상의 대비로 문을 쉽게 구별 할 수 있음 - 문의 창문과 문의 색상 대비가 부족하여 문을 구별하기 어려움
4	+ 문과 정면의 색상 대비로 문을 찾기가 쉬움 + 문과 정면을 색상의 대비로 문을 쉽게 구별 할 수 있음

인지 행동 유도성

녹색은 일반적으로 안전과 연관성을 가지고 있다. 따라서 녹색의 사용은 회색 및 흰색 색상의 사용보다 더 높은 인지 행동 유도성을 발생시킨다. 아래 표에서 인지 행동 유도성의 요소를 확인 할 수 있다.

[인지 행동 유도성의 증가 또는 감소 기여 요인 (1-4)]

No.	감각 행동 유도성
1	+ 문과 정면의 색상 대비로 문을 찾기가 쉬움 + 문과 정면을 색상의 대비로 문을 쉽게 구별 할 수 있음
2	- 문과 정면의 색상 대비가 부족하여 문을 찾기가 어려움 - 문과 정면을 색상 대비가 부족하여 문을 구별하기 없음
3	+ 문과 정면의 색상 대비로 문을 찾기가 쉬움 + 문과 정면을 색상의 대비로 문을 쉽게 구별 할 수 있음 - 문의 창문과 문의 색상 대비가 부족하여 문을 구별하기 어려움
4	+ 문과 정면의 색상 대비로 문을 찾기가 쉬움 + 문과 정면을 색상의 대비로 문을 쉽게 구별 할 수 있음

물리적 행동 유도성

모든 비상구 설계에 동일한 것으로 가정이 되기 때문에 물리적인 행동 유도성을 고려되지 않으며, 문의 물리적 요소의 변경을 고려되지 않는다.

기능 행동 유도성

기능 행동 유도성은 모든 운전자가 터널에서의 목표와 연관이 되어 있다. 터널 사용자의 목표는 안전하게 극복하는 것이며 모든 시스템은 이를 고려하여 설계되어 있다. 이러한 이유로 기능 행동 유도성은 인지 및 감각 행동 유도성에 영향을 준다고 주장하고 있다. 이러한 요소는 감각 그리고 인지 행동 유도성의 증가 또는 감소 기여 요인의 표에서도 확인 할 수 있다. 기능 행동 유도성의 높은 점수는 다른 행동 유도성의 조합으로부터 결과를 확인 할 수 있다. 만약 다른 행동 유도성의 실패가 나타난다면, 기능 행동 유도성의 결과 역시 낮거나 실패로 이어진다.

[행동 유도성 이론을 사용하여 정한 비상구 색상 순위]

No.	감각 행동	인지 행동	물리적	기능적
1	높음	높음	해당 없음	높음
2	낮음	높음	해당 없음	낮음/중간
3	중간	낮음	해당 없음	낮음/중간
4	높음	낮음	해당 없음	중간

결론

하나의 패널에는 텍스트와 다른 하나의 패널에는 그림 문자의 사용을 권장한다.
“비상 출입구” 문구 사용은 경고 문구 보다 더 선호한다.
패널에 사용되는 문자 색상은 황색을 권장한다.
패널에 경고 메시지와 플래싱 라이트를 권장한다.
패널의 크기는 증가하는 것을 권장한다.
표지에 작성될 정보의 양은 작게 유지하여야 한다.
표지 정보는 적적하게 분리되어야 한다.

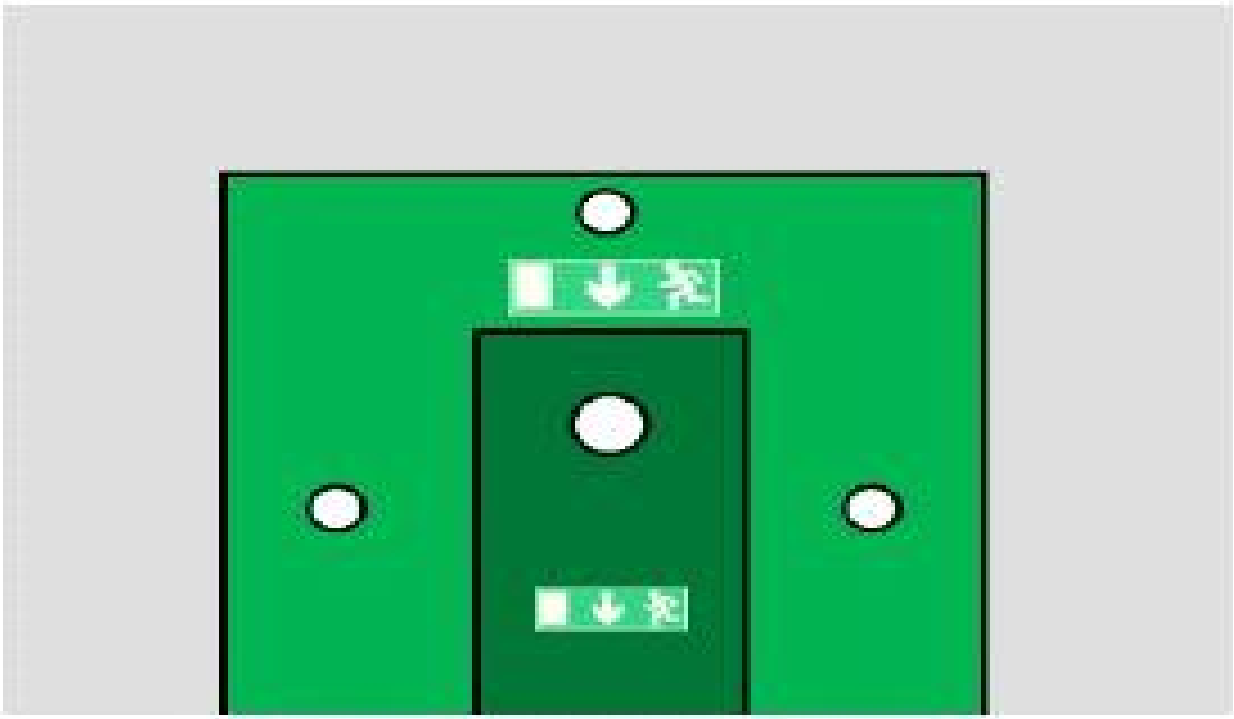
아래 그림은 교통 정보 표지판으로 권장된 디자인이다.



[교통 정보 표지판으로 권장된 디자인 (6번 디자인)]

교통 정보 표지판으로 권장된 이유는 아래와 같다.
-직사각형의 텍스트 패널 및 정사각형의 그림문자 패널 사용
-녹색 색상의 비상 탈출 그림문자 사용
-확대된 패널의 사이즈
-황색 색상의 메시지
-플래싱 라이트 사용으로 집중력 향상
-적당한 양의 글자 정보 및 2 줄 텍스트

비상 탈출구 설계로 권장된 색상은 “안전 녹색보다 어두운 녹색”이다. 비상 탈출구로 가능한 색상의 배열 중에서 1번 디자인 이였으며 감각 인지 기능적 모든 부분에서 높은 행동 유도성을 가졌다. 색상의 배열은 아래와 같다.



[비상 탈출구 색상으로 권장된 디자인 (1번 디자인)]

1) 플래싱 라이트

□ 내용

이동 방법 시스템의 안전 수준을 높일 수 있는 방법은 터널에서의 적절한 경로를 촉진시켜 최종 대피 시간 및 위험을 감소하는 것이다. 이동 방법 시스템의 효과는 이용자가 비상 탈출구를 사용함에 따라 반영된다. 촉진 시키는 방법에는 여러 가지가 있다. 그 중에서도 이번 장에서는 라이트를 이용한 방법에 중점을 두고자 한다. 라이트의 사용은 비상구 사용을 위해 좋은 방법으로 보고되고 있다. 플래싱 라이트의 사용은 터널안에서 좋은 방법이며 비상 탈출구 선택에 있어서 불확실성을 감소시키는 역할을 한다. 플래싱 라이트의 사용은 권장 되며 플래싱 라이트 특징은 조사되었으며, 터널에서의 대피로부터 플래싱 라이트의 특성의 체계적인 평가가 필요하다.

플래싱 라이트 실험

아래 조건을 확인하기 위해 실험을 진행하였다.
- 라이트의 색상
- 플래싱 비율
- 라이트 유형
라이트의 배열 및 수량

실험을 위해 라이트의 색상은 녹색, 흰색 그리고 파란색을 고려하였다. 일반적으로 흰색은 정보를 전달하기 위해 사용지만, 녹색은 안전 메시지를 전달하기 위해서 가장 많이 사용되는 색상이다. 파란색은 이전 연구에서 응급 신호에 효과적일 수 있다고 발표 되었으며 이전 연구에서는 1Hz 동작하는 상황에서 비상구 사용이 효과 적인 것으로 나타났다. 체계적으로 조사를 진행하기 위해서 4 Hz 부터 0.25 Hz 까지 다양한 속도를 설정하여 이상적인 레이트를 조사하고자 한다.
또한 라이트 유형에 따른 비교를 진행하였다. 스트로브 조명과 더블 스트로브 현재 터널에서 비상 조명으로 사용이 되고 있으니, 갑자기 방전되는 단점을 가지고 있다. 기존의 백열 램프와 비교하여 LED 등은 긴 수명과 높은 전기 효율을 가지고 있다.
이외에 전구의 수에 따른 실험을 진행하고자 한다. 전구의 경우 경제적인 요소의 영향 등을 고려하여 최대의 전등 수를 반영한다.
이와 같은 실험 조건을 바탕으로 효율적인 비상대응을 찾기 위해서는 행동 유도성의 분석도 필요로 한다. 행동 유도성 분석은 아래와 같이 4가지로 나눌 수 있다.

감각 행동 유도성 : 느낌 또는 관찰
인지 행동 유도성 : 이해
물리적 행동 유도성 : 어떤 행동을 하거나 사용
기능 행동 유도성 : 목표 달성을 위해

감각 행동 유도성

비상 탈출 시스템이 의도한대로 작동하는지 이용자에 의해 보거나 듣는 것이 가장 먼저 이루어진다. 예로 비상 탈출구의 색상을 주위와 비슷한 색상 또는 패턴을 선택하여 이용자에게 있어 혼란을 준다. 이러한 실수를 줄이고 쉽게 이용자가 이용하도록 도와주는 것이다.

인지 행동 유도성

비상 탈출 시스템의 이해를 돕는 것이 목적이다. 색상은 비상 탈출 시스템의 이해에 큰 영향을 미친다. 예를 들어 녹색은 안전한 이동을 의미한다. 인지 행동 유도성은 특정 디자인에 의해 제공되며, 상황의 전후 사정에 의해서 영향을 받을 수 있다.

물리적 행동 유도성

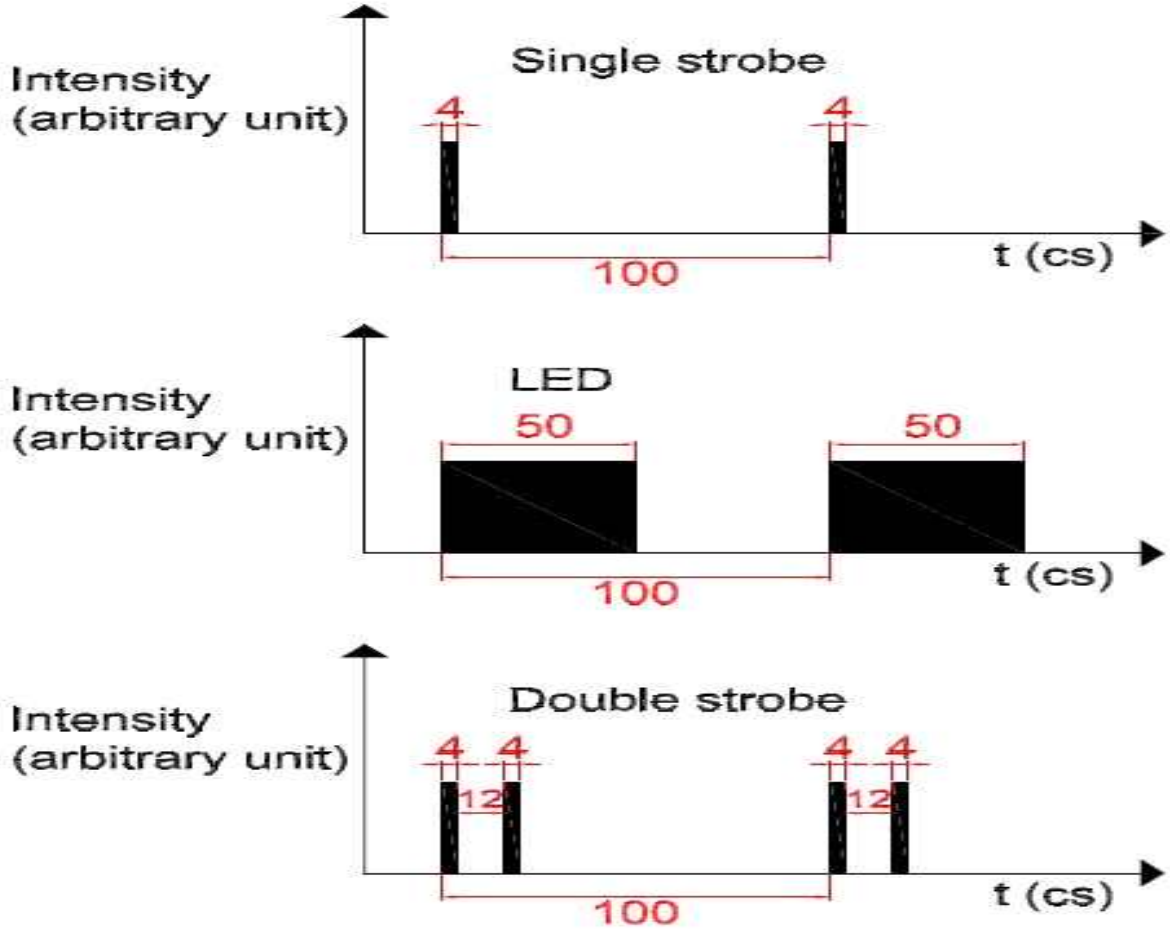
물리적인 행동 유도성은 가장 기본적인 사항 중에 하나이다. 하지만 비상 탈출구에 있어 라이트 사용에 대한 고려는 감각, 인지 그리고 기능적인 면에서만 고려가 된다. 이러한 이유로 물리적 행동 유도성은 고려사항이 아니다.

기능 행동 유도성

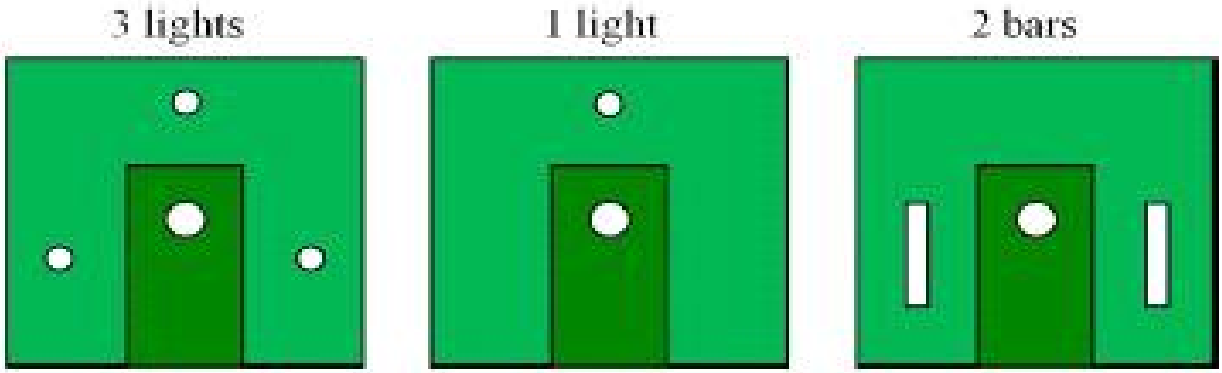
기능 행동 유도성은 이용자가 원하는 목적을 달성하고, 인지, 감각 그리고 물리적 행동 유도성의 조합의 최종 결과이다.

[실험 변수 및 구성 요소]

변수	설치 구성
색상	녹색 흰색 파란색
플래싱 비율	0.25Hz 1Hz 4Hz
라이트 유형 (라이트 종류의 도식)	스트로브 LED 더블 스트로브
라이트 배열 (라이트 배열의 도식)	2 바 3 라이트 (사이드 2, 위 1) 1 라이트



[라이트 유형]



[3가지 종류의 라이트 배열]

실험은 실험 변수 및 구성 요소 에서와 같이 여러 가지 조합을 통해서 진행하였으며 자세한 조합은 아래와 같다.

[실험 변수 조합]

No	변수	색상	플래싱 비율	라이트 유형	라이트 배열
RS	색상	녹색	1	LED	라이트 3
C1		흰색	1	LED	라이트 3
C2		파란색	1	LED	라이트 3
FR1	플래싱 비율	녹색	4	LED	라이트 3
RS		녹색	1	LED	라이트 3
FR2		녹색	0.25	LED	라이트 3
TL1	라이트 유형	녹색	1	스트로브	라이트 3
RS		녹색	1	LED	라이트 3
TL2		녹색	1	더블 스트로브	라이트 3
LP1	배열 및 위치	녹색	1	LED	바형태 2
RS		녹색	1	LED	라이트 3
LP2		녹색	1	LED	라이트 1
NO		라이트 없음			

결론

최신 경향은 분명 비상 탈출구에 라이트를 설치하는 것이며 이는 비상 탈출구 사용에 매우 긍정적인 영향을 미친다. 이러한 이유 때문에 점멸된 상황에서의 실험은 다른 실험에 비해서 가장 낮은 점수를 받았다. 플래싱 비율은 1Hz와 4Hz의 성능이 0.25Hz 보다 좋은 것으로 나타났다. 플래싱 비율이 낮은, 0.25Hz의 경우 이용자에 있어서 상황의 긴박성을 받아들이지 않는 것으로 나타났다. 감각과 인지 행동 유도성이 낮게 나타났다.
라이트 유형 선택에 있어서 크게 2가지로 나뉘었으며 LED 또는 스트로브 유형으로 나누어 실험을 진행하였다. 비교 실험 결과 LED등이 스트로브 및 더블 스트로브에 비해 성능이 더 좋은 것으로 결론이 도출 되었다. 이러한 결론은 터널의 비상 디자인 설치 및 비용에 상당한 영향을 미칠 수 있다.
이와 달리 라이트 배열은 터널 탈출에 있어서 많은 영향을 주지 않는 것으로 나타났지만 1개의 라이트를 설치하여 실험한 경우 감각 행동 유도성이 낮게 도출 되었다. 단, 인지 행동 유도성 부분에 있어서는 통계적인 차이가 없는 것으로 나타났다.

기존 현장 대응 메뉴얼은 재난대응에 있어서 재난 발생과 유관기관과의 연락위주의 현장 대응이 위주로 되어 있어 초기대응 이후의 중앙사고수습본부 (중수본) 이후의 재난 대응으로 이루어져있다. 그러나, 본 시스템은 화재와 화학테러의 두가지 재난 시나리오를 선

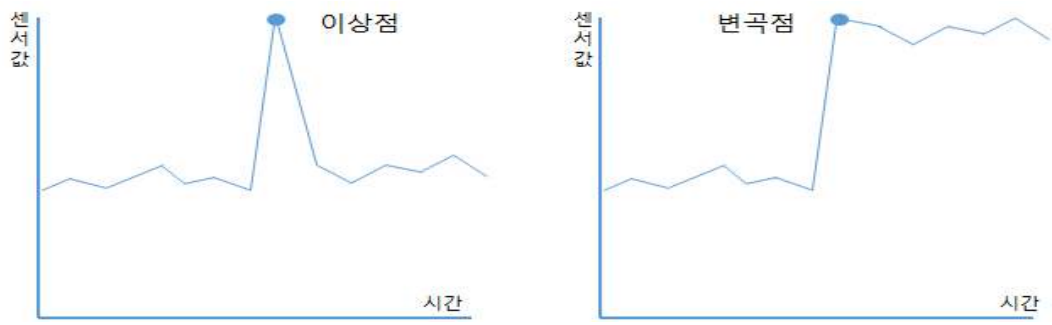
정하여 재난 발생 위치를 기반으로 승객들의 대피를 우선적으로 수행하는 초기대응을 위한 시스템으로 역무원을 기존 현장 대응 메뉴얼을 그대로 수행함과 동시에 승객들의 초기대응에 있어 적극적으로 개입하여 대피경로를 안내하는 시스템으로 차별화가 되어있다.

따라서 실험의 결과에 따라 아래와 같이 터널 비상 탈출구는 설치하는 것이 권고 된다.
-녹색 또는 흰색 라이트가 파란색 라이트 보다 더 효율적이다.
-플래싱 비율은 1Hz 에서 4Hz
-LED가 스트로브에 비해 더 효율적이다.
-라이트 배열에 있어서 3가지 방법으로 실험을 진행 하였지만 별다른 차이점을 발견하지 못하였다.
SOP기반 재난 대응 프로세스 및 재난전조대응 M2M 알고리즘 개발 기초 연구

재난전조 및 대응을 위해서 재난 전조예측을 위한 시계열 데이터분석 알고리즘 개발 및 피난유도를 위한 피난 경로 분석 및 유도 알고리즘 개발에 관한 연구를 수행하여 다음과 같은 성과를 얻었으며 세부 내용은 별책에 기술하였다.

3. 재난 전조예측을 위한 시계열 데이터분석 알고리즘 개발

재난 전조예측을 위한 시계열 데이터분석 알고리즘 개발은 재난전조에 대한 정의를 일상적인 데이터 범위를 벗어나는 센서 데이터가 일정기간 이상 지속적으로 수신이 되는 경우로 정의하였다. 재난전조를 예측함에 있어서 가장 중요한 점으로 오작동 및 이상에 대한 필터링과 재난전조 예측의 빠르기로 설정하고, 이에 대한 연구를 진행하였으며, 오작동 및 이상에 대한 필터링은 센서의 성능, 설치 위치 등의 차이에 의해서 같은 제품이라도 다른 패턴을 보여줄 수 있기 때문에 개별 센서에서 생성되는 데이터에 기반한 룰 생성이 가장 적합하며, 이를 위해서 센서에서 발생하는 시계열 데이터에서 일시적인 이상상태와 재난 발생으로 인한 데이터의 변화의 시작점 즉 변곡점을 구분하는 알고리즘을 개발하고 테스트 데이터를 이용한 실험을 수행하였다.



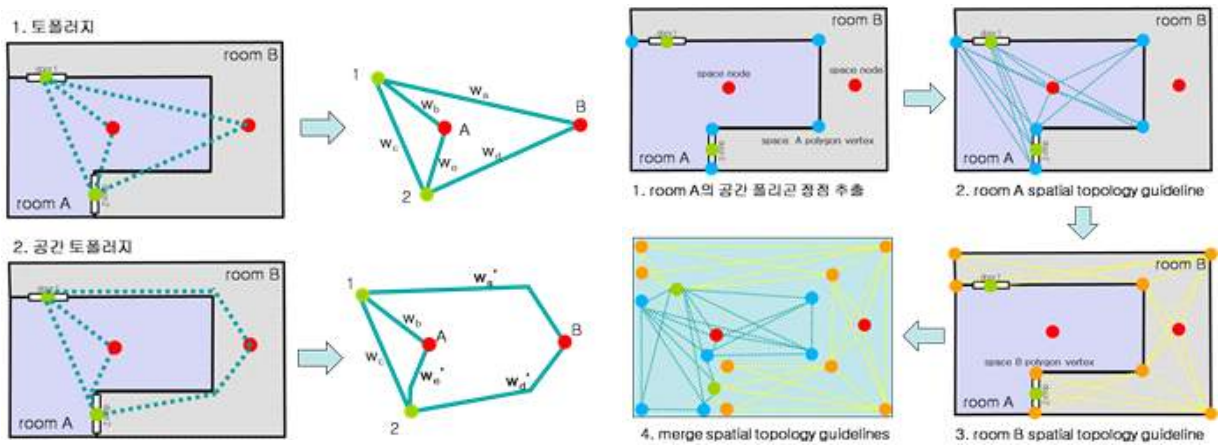
[센싱데이터의 이상점과 변곡점의 차이]

프로트타입에 대한 실험결과 센싱 시계열 데이터를 정규분포로 가정을 하고 확률밀도함수로 이상치를 검출가능하며 확률에 따른 검출성공률에 차이가 발생하는 결과를 얻었다. 확률밀도함수가 0.8정도에서 95%의 정확도를 얻었으며, 재난발생 후 20번째 이내의 시간대에서 60%이상이 검출되는 결과를 도출하였다. 이는 센싱데이터가 초당 1회 수집되는 것으로 가정을 하면 재난발생 후 20초 이내에 60%정도의 재난을 검출할 수 있다는 의미

이나, 본 연구는 샘플데이터에 대해서 제한적인 상황에서 진행된 실험이기 때문에 이번 연차년도에서는 전조감지 가능성에 대한 기본적인 알고리즘을 달성했다는데 그 의의가 있으며 향후 연구를 통해서 실데이터에 적합한 개선된 알고리즘 개발을 수행하였다.

1) 피난유도를 위한 피난 경로 분석 및 유도 알고리즘 개발

스마트 N키퍼가 피난을 유도하기 위해서는 각각 설치된 위치에서 어느 경로를 통해서 유도할 것인가에 대한 의사결정이 필요하며, 이러한 의사결정을 위해서는 지하철의 전체 공간에 대한 정보 및 특히 대피경로에 대한 정보가 필요하다. 대피 경로에 대한 정보에는 경로뿐만이 아니라 각 경로를 이용해서 대피하는데 필요한 시간 정보도 필요한데, 시간 정보를 이용해서 각 경로 간의 대피 시간 비교 및 대안제시를 해야 하는 경우도 발생하기 때문이다. 이를 위해서 지하철공간을 피난 경로 및 피난 시간을 도출할 수 있도록 그래프 구조로 변환하는 연구를 수행하였으며, 캐드 데이터에서 피난경로 토폴로지를 도출하는 알고리즘을 개발하였다. 또한, 대상 공간의 혼잡정도 등 피난유도에 영향을 주는 정보를 피난경로 토폴로지에 반영하기 위해서 그래프 이론에 관한 기초연구를 수행하였다.



①토폴로지와 공간 토폴로지의비교 ②공간토폴로지 작성 방법

[피난유도 공간 토폴로지 작성 방법]

4. 재난 전조 대응 스마트N키퍼 모듈

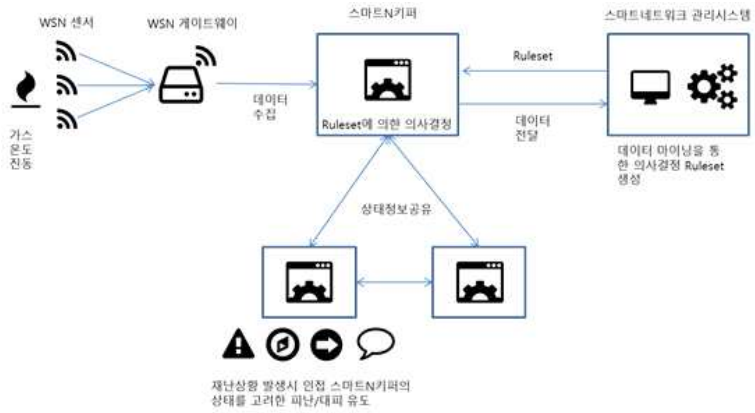
1) 시스템 구성 기획

가) 시스템 전체 개요

스마트N키퍼는 지하공간(지하철) 시설물에 설치된 센서를 통해 여러가지 환경정보 데이터를 수집하여 승객 대피 의사결정 을 지원한다.

관리자의 의사결정에 따라 재난의 경우 스마트N키퍼는 재난모드로 전환이 되어 이용객의 대피를 유도한다.

스마트N키퍼는 자신의 상태를 다른 스마트N키퍼와 공유하여 이용객에게 정확한 대피로를 안내한다.

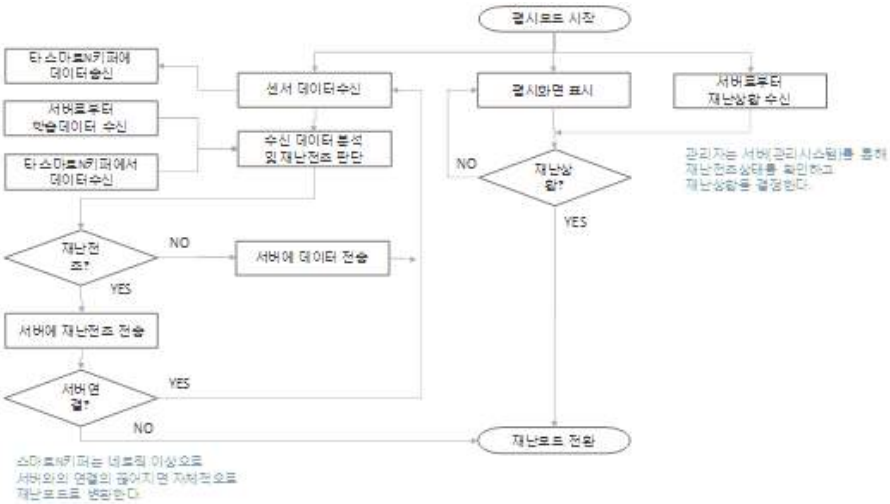


[시스템 전체 개요]

2) 평시모드 작동 시나리오

스마트N키퍼의 평시모드는 센서 데이터를 수집하여 분석한후 데이터를 서버의 의사결정 학습 모듈에 전달하여 분석하도록 하며, 분석된 데이터를 수신받아 업데이트한다.

UI는 재난 시 대피요령에 대한 안내, 평시 안전관련 수칙등의 정보를 순차적으로 제공한다.

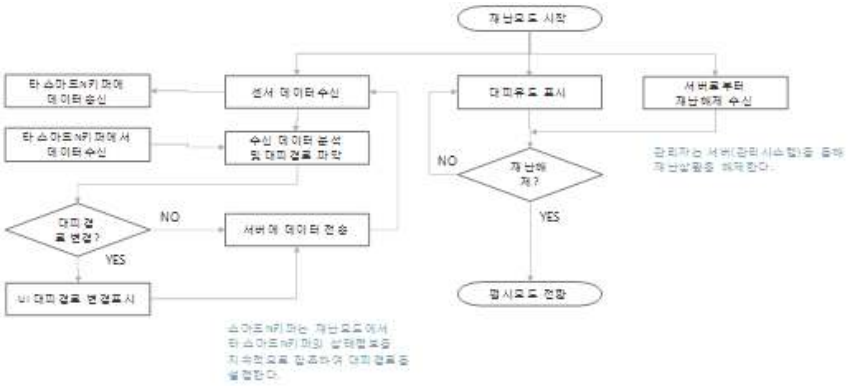


[시스템 평시 모드 작동 시나리오]

3) 재난모드 작동 시나리오

스마트N키퍼의 재난모드는 서버로부터 재난발생상황을 수신한 상태이며 UI가 재난모드로 변경이 되어 이용객에게 현재 발생한 재난의 종류와 대피로의 경로를 기본으로 표시한다.

사용자의 선택에 따라 대피방향, 재난발생지점, 최적대피경로 및 안전대피방법을 표시한다. 서버로부터 재난해제가 수신될 때까지 유지된다.

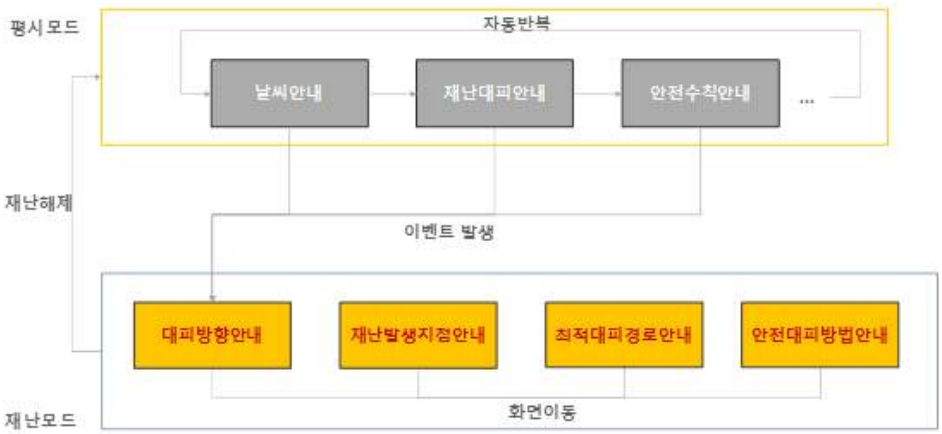


<그림 재난모드 작동 시나리오>

4) 스마트 N 키퍼 UI 구성

가) 개요

스마트 N키퍼는 중앙 관리 시스템과 네트워크상에서 연결이 되고 있을 경우에는 관리하는 각종 센서의 정보를 중앙 관리 시스템으로 전달하고 이용객에게 각종 정보를 공유하는 디지털 사이니지(Digital Sinage)로의 역할을 하는 평시모드와 중앙 관리 시스템에서 전달받은 SOP를 지속적으로 업데이트하다가 재난전조감지 및 재난 발생 시 관할 구역에 있는 이용객을 안전하게 대피할 수 있는 SOP 가이드로서 역할을 하기 위한 재난모드가 있다.



[스마트 N 키퍼 구성]

나) 평시모드

스마트 N키퍼의 평시모드에서는 재난시 대피요령에 대한 안내, 평시 안전관련 수칙등의 정보를 순차적으로 제공한다.







<그림 평시모드 화면 예시>

다) 재난모드

스마트 N키퍼의 재난모드에서는 현재 발생한 재난의 종류와 대피로의 경로를 기본으로 표시하고 사용자의 선택에 따라 대피방향, 재난발생지점, 최적대피경로 및 안전대피방법을 표시한다.

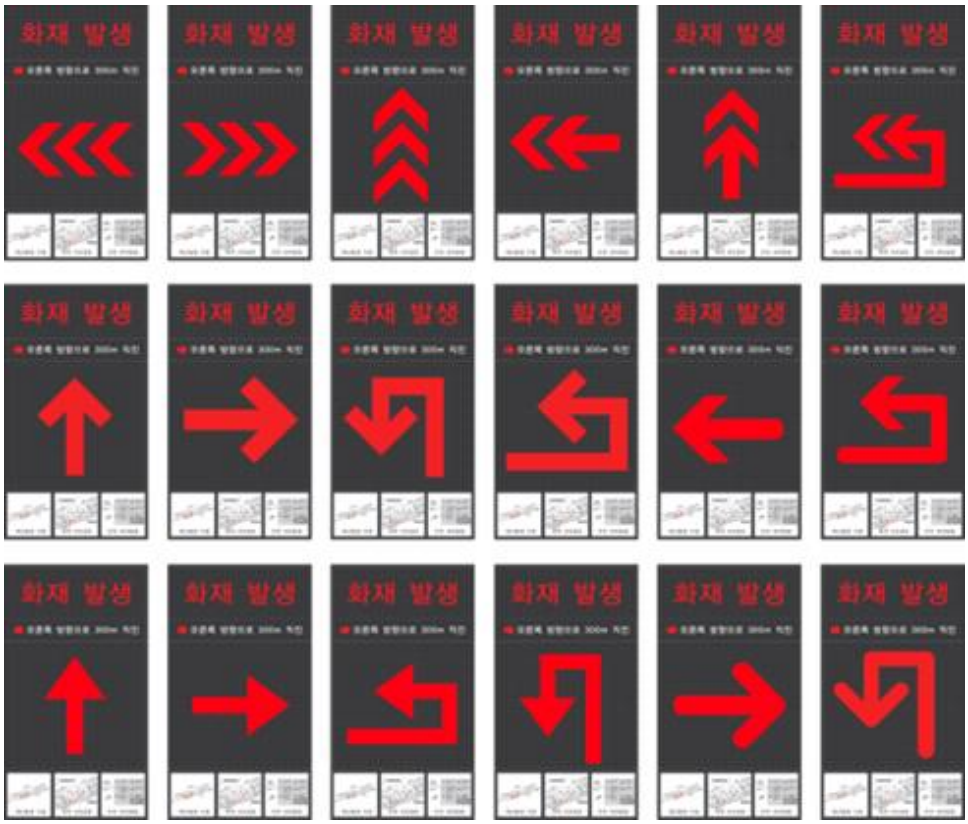


<그림 재난모드 화면 예시>

라) 방향표지예시

아래와 같이 다양한 방향을 표지할 수 있도록 다양한 방향표지 화살표를 디자인해서 시스템에 내장시킨다.

재난 발생 시 발동 시나리오에 따라서 각 키퍼들은 설정된 방향표지 화살표를 전면에 띄워서 피난유도를 수행하게 된다.



<그림 방향 표지 예시>

5. 재난전조 예측/재난 대응 모듈 α버전 개발

1) 개요

- 재난전조예측 및 재난대응 모듈은 센서로부터 취득한 센싱 데이터에 대해서 재난발생 유무를 판단하고 기능을 수행한다.
- 각 개별 센서 별로 센싱 데이터의 기존 데이터에 데이터 마이닝의 기법 중 하나인 의사결정나무(decision tree) 알고리즘을 사용하였다.

가) 데이터 마이닝 개요

① 데이터 마이닝이란

데이터마이닝은 일반적으로 대량의(꼭 대량이 아니어도 됨) 복잡한 데이터로부터 묵시적이고 잠재적인 '알려지지 않은 정보' 즉 특별한 경향을 찾아내는 일련의 과정이다.
또한 데이터 마이닝이란 '대량의 데이터로부터 새롭고 의미있는 정보를 추출해 의사결정에 활용하는 작업'이다. 이런 면에서 볼 때 '채굴하다'라는 의미의 마이닝은 매우 적절한 용어이다. 다이아몬드를 만들기 위해서는 흙과 잡석을 파헤치고 제거하듯이 데이터베이스에서도 숨겨져 있는 정보를 찾기 위해 많은 양의 데이터 부스러기를 제거하는 작업을 해야 하기 때문이다.

② 데이터마이닝과 알고리즘

데이터 마이닝에 있어서 중요한 점은 알고리즘의 이해에 있다. 이것은 모델의 선택이나 샘플링에 있어서 중요한 문제가 된다.
데이터 마이닝에서는 알고리즘으로써 통계적 수법과 기계 학습 분야의 다양한 알고리즘을 데이터 속에 숨어 있는 패턴을 인식하는데 사용한다.
데이터마이닝과 패턴 인식의 차이는 패턴인식은 데이터 속에 숨어 있는 패턴을 수치적으로 인식하는 것을 목표로 하나, 데이터 마이닝은 이렇게 발견된 패턴이 어떤 의미를 가지고 있는지에 대한 설명가능성을 연구하는 분야이다.

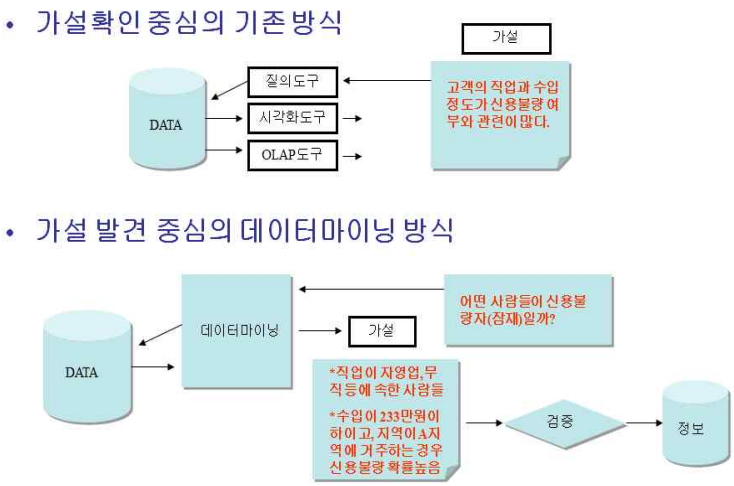
③ 가설확인중심의 기존 통계해석 방식과 데이터마이닝의 차이

통계해석은 가설의 증명이고, 데이터 마이닝은 문제제기와 그 문제의 원인 및 구조분석에 있다.
통계해석은 가설을 설정하고 이것을 증명하기 위한 데이터 수집 및 분석함, 증명하고자 하는 가설이 데이터에 영향을 받지 않고, 증명이 되면 그 가설은 채용이 된다. 하지만 데이터 마이닝에서는 분석하고자 하는 데이터가 먼저 존재하고, 그 데이터에 무엇이 들어 있는지는 모르기 때문에 가설설정과 같은 과정은 필요가 없는 특징이 있다.
데이터의 산더미에서 문제점을 발견하고 그 문제점의 원인 및 구조를 찾아내는 것이 데이터 마이닝이다.

④ 데이터마이닝의 기원

데이터마이닝은 통계학과 기계학습(machine learning: 인공지능으로도 알려짐)으로 알려진 두 학문분야의 합류점에서 존재한다.
데이터를 탐색하고 모델을 구축하는 다양한 기법들은 통계학분야에서 오랫동안 존재해 왔다. 예를 들어 여기에는 선 형 회귀분석, 로지스틱 회귀분석, 판별분석, 주성분 분석 등이 포함된다.
그러나 충분한 데이터와 계산능력을 가진 데이터마이닝의 응용분야에서는 이러한 고전적인 통계학의 핵심원리(계산이 어렵고 데이터가 희소하다는 것)가 적용되지 않는다.
이러한 이유로 해서 Daryl Pregibon은 데이터마이닝을 "규모와 속도의 통계학 (statistics at scale and speed)"으로 묘사하고 있다(Pregibon, 1999). 이를 좀더 확장 한 개념은 "규

모, 속도 및 단순성의 통계학(statistics at scale, speed, and simplicity)" 이다.
이 경우에 단순성이란 알고리즘의 단순성뿐만 아니라 추론논리의 단순성을 의미한다. 전통적인 통계환경하에서는 데이터가 희소하기 때문에 추정치를 계산하고 추정치가 얼마나 신뢰할 만한가를 결정하는 데 동일한 표본이 사용된 다. 그 결과, 추론을 위해 사용되는 신뢰구간과 가설검정에 대한 논리는 대부분 의 경우에 이해하기가 쉽지 않으며, 또한 이러한 한계점들은 잘 인식되지 못하고 있다.
이와는 반대로 하나의 표본으로 모델을 적합시킨 후 다른 표본으로 모델의 성과를 평가하는 데이터마이닝의 이론적 구조는 이해하기가 쉽다.
오늘날 컴퓨터 과학은 우리에게 의사결정나무(decision tree) 또는 신경망(neural networks)과 같은 기계학습기법(machine learning techniques)을 제공한다.
기계학습 기법은 강력한 계산능력에 의존하며 전통적인 통계모형보다는 덜 구조화되어 있다. 아울러 현재 성장하고 있는 데이터베이스관리분야 또한 부분적으로 이 기법 들의 영역에 속한다.
전통적인 통계학은 추론(하나의 패턴 또는 흥미로운 결과가 우연히 발생하였는지를 결정하는 것)에 초점을 두고 있지만 데이터마이닝은 그렇지 않다. 통계학과 비교 할 때, 데이터마이닝은 다양한 방식으로 대량의 데이터 집합을 다루기 때문에 추론에서 요구하는 것처럼 엄격한 제약을 둘 필요가 없다. 그러나, 이로 인해서 데이터마이닝에 대한 접근은 일반적으로 과적합화(overfitting)의 위험성에 쉽게 노출된다. 즉, 하나의 모형이 모형개발에 사용된 표본데이터에 너무 가깝게 적합화되면 이 모형은 데이터가 갖는 구조적 특성뿐만 아니라 우연적 특수성을 모두 반영하게 되어 과적합화 현상이 발생하게 된다.
공학적인 관점에서 이러한 모형은 신호(signal)가 아닌, 잡음(noise)을 적합시킨 모형이 되고 만다.



<그림 기존방식과 데이터마이닝의 문제 접근 방식의 차이>

2) 데이터 마이닝의 주요 용어

데이터마이닝은 태생적으로 여러 분야가 결합된 분야이기 때문에 실무자들은 종종 동일한 의미에 대해 다양한 용어들을 사용한다.
예를 들어 기계학습(인공지능)분야에서는 예측되는 변수에 대해 출력변수 또는 목표변수라는 용어를 사용 하고, 통계학에서는 이를 종속변수 또는 반응변수라고 부른다.
본 연구에서는 아래와 같은 의미로 용어를 사용하였다.

알고리즘(algorithm)

- 특정 데이터마이닝 기법, 예를 들어 분류나무, 판별분석 등을 실행하기 위해 사용되는 특정 절차를 의미한다.

속성(attribute)

- 보통 X 로 표기되며, 특성, 예측변수, 입력변수, 독립변수, 또는 데이터베이스 관점에서 필드(field)라고도 한다.

사례(case)

- 고객, 거래 등의 측정치를 갖는 분석의 단위로서 관찰치, 레코드, 패턴, 또는 행(row)이라고도 한다(각 행은 레코드를, 각 열은 변수를 의미한다).

신뢰도(confidence)

- 이 용어는 다음과 같이 2가지 의미를 갖는다.
- 첫째, “A 와 B 가 구매된다면 C 또한 구매될 것이다.” 라는 연관성규칙에서 사용되며, 이때 신뢰도는 A 와 B 가 구매된다는 조건하에서 C 가 구매될 조건부 확률을 말한다.
- 둘째, 신뢰도는 또한 통계학의 신뢰구간(confidence interval)과 같이 좀더 넓은 의미로 쓰인다. 다른 표본과는 반대되는 표본을 선택함으로 인해 발생하는 추정치의 오차 정도를 말한다.

종속변수(dependent variable)

- 보통 Y 로 표기되며, 지도학습으로 예측되는 변수이다. 또한 반응변수, 출력변수, 목표변수, 또는 성과변수라고도 한다.

추정(estimation)

- 연속형 출력변수의 가치를 예측하는 것을 말하며, 예측 (prediction)이라고도 한다.

특성(feature)

- 보통 X로 표기되며, 예측변수, 입력변수, 독립변수, 또는 데이터 베이스 관점에서 필드라고도 한다.

예비용 표본(holdout sample)

- 모형을 적합 시키는데 사용되지 않고 모형의 성과를 평가하는데 사용되는 데이터 표본이다. 이 책에서는 예비용 표본이라는 용어 대신에 평가용 집합(validation set)이라는 용어를 사용하거나 검증용 집합 (test set)이라는 용어를 사용한다.

입력변수(input variable)

- 보통 X로 표기되며, 특성, 예측변수, 독립변수, 또는 데이터베이스 관점에서 필드라고도 한다.

모형(model)

- 데이터 집합에 적용되어 파라미터(parameter)를 결정하는 알고리즘을 말한다(많은 알고리즘은 사용자가 조정할 수 있는 파라미터를 가짐).

관찰치(observation)

- 고객, 거래 등의 측정치를 갖는 분석의 단위로서 사례, 레코드, 패턴, 또는 행(row)이라고도 한다(각 행은 레코드를, 각 열은 변수를 의미함).

성과변수(outcome variable)

- 보통 Y로 표기되며, 지도학습으로 예측되는 변수이다. 또한 종속변수, 반응변수, 출력변수, 또는 목표변수라고도 한다.

출력변수(output variable)

- 보통 Y로 표기되며, 지도학습으로 예측되는 변수 이다. 또한 종속변수, 반응변수, 목표

변수, 또는 성과변수라고도 한다.

P(A|B)

- 사건 B가 발생한다는 조건하에서 사건 A가 발생할 조건부 확률을 의미한다.

패턴(pattern)

- 하나의 관찰치에 대한 측정치의 집합을 말한다. 예를 들어 한 사람의 키, 몸무게 및 나이를 묶어서 하나의 패턴으로 정의한다.

예측(prediction)

- 연속형 출력변수의 가치를 예측하는 것을 말하며, 추정(estimation)이라고도 한다.

예측변수(predictor)

- 보통 X로 표기되며, 특성, 입력변수, 독립변수, 또는 데이터베이스 관점에서 필드(field)라고도 한다.

레코드(record)

- 고객, 거래 등의 측정치를 갖는 분석의 단위로서 사례, 관찰치, 패턴, 또는 행(row)이라고도 한다(각 행은 레코드를, 각 열은 변수를 의미함).

반응변수(response variable)

- 보통 Y로 표기되며, 지도학습으로 예측되는 변수이다. 또한 종속변수, 출력변수, 목표변수, 또는 성과변수라고도 한다.

점수(score)

- 예측 값 또는 예측집단을 나타낸다. 새로운 데이터에 점수를 매기는 것은 학습용 데이터를 가지고 개발된 모형을 이용하여 새로운 데이터에 출력 값을 예측하는 것을 의미한다.

성공집단(success class)

- 이진 집단변수에서 중요한 집단을 말한다. 예를 들어 구매/비구매의 성과변수에서 구매자 집단이 성공집단에 해당한다.

지도학습(supervised learning)

- 주요 종속변수가 알려져 있는 레코드에 알고리즘(로지스틱 회귀분석, 회귀나무 등)을 적용하는 과정을 말한다. 알고리즘은 출력변수가 알려져 있지 않은 신규 레코드를 가지고 출력변수 값을 예측하는 방법을 학습한다.

검증용 데이터(test data)(또는 검증용 집합)

- 최종 선택모형이 새로운 데이터 에 대하여 얼마나 좋은 성과를 갖는지를 평가하기 위해서 모형구축 및 모형선택과정이 끝난 후에만 사용되는 데이터를 말한다.

학습용 데이터(training data)(또는 학습용 집합)

- 모형을 적합화 시키기 위해 사용되는 데이터를 말한다.

자율학습(unsupervised learning)

- 주요 출력 값을 예측하기보다는 데이터 자체의 특성에 대한 학습을 시도, 분석한다(예를 들어 군집들의 소속여부 등).

평가용 데이터(validation data)(또는 평가용 집합)

- 모형이 얼마나 잘 적합화 되었는지 평가하고, 일부 모형들을 조정하며, 구축된 모형들 중에서 가장 좋은 모형을 선택하기 위한 목적으로 사용되는 데이터를 말한다.

변수(variable)

- 입력변수(X)와 출력변수(Y)를 모두 포함하는 레코드의 측정치를 말한다.

3) 데이터 마이닝 응용 사례

① 적용사례 개요

○ 데이터 마이닝은 다양한 산업에서 활용되고 있다. 특히 금융권에서는 예전부터 활용을 하고 있다.

○ 한 지불 결제 처리 업체는 실시간으로 10억 개 이상의 거래 중에서 숨겨진 사기를 탐지함으로써 손실 금액을 월 100만 달러 가량 줄이고 있다.

○ 어느 자동차 보험업체는 상세한 지역 관련 데이터를 사용해 심각한 기상 악화가 비즈니스에 미치는 영향을 모델링하고 보험 청구로 인한 손실 금액을 예측한다.

○ 한 제조업체는 차량 텔레매틱스에서 생성하는 데이터를 사용해 운영 지표 상의 패턴을 발견하고 선제적인 정비를 유도 한다.

○ 사기 방지는, 1억 5,000만 개의 디지털 월릿을 통해 연간 2,000억 달러 이상의 결제를 처리하는 페이이다.

○ 페이팔(PayPal)은, 온라인 결제업계의 선두 주자다. 이정도 규모에서는 사기 비율이 낮다 해도 그 비용은 상당 규모에 이른다. 창업 초기에는 월별 사기 피해 금액이 1,000만 달러에 이르렀다. 페이 팔은 이 문제를 해결하기 위해 최고의 연구원들로 팀을 꾸렸고 이 팀은 최신 머신러닝 기법을 사용해 사기성 결제를 실시간 으로 식별하는 모델을 구축했다.

○ 타겟팅 디지털 디스플레이는, 광고 기술 기업 디스틸러리(Dstillery)가 머신러닝을 사용해 바라이즌(Veriszon), 윌리엄스-소노마(Williams-Sonoma)와 같은 기업이 실시간 입찰 플랫폼에서 타겟 디지털 디스플레이 광고를 진행하도록 한다. 디스틸러리는 개인의 브라우징 내역, 방문, 클릭 및 구매에 대해 수집된 데이터를 사용해 한 번에 수백 개의 광고 캠페인을 처리하며 초당 수천 건의 예측을 실행한다. 덕분에 디스틸러리는 투자대비 최적의 결과를 얻기 위한 타겟 광고에서 인간 마케터보다 훨씬 더 좋은 성과를 내고 있다.

○ 콘텐츠 추천은, 컴캐스트(Comcast)가 X1 인터랙티브 TV 서비스 고객을 위해 각 고객의 이전 시청 습관을 기반으로 한 실시간으로 개인 맞춤화 된 콘텐츠를 추천한다. 컴캐스트가 운영하는 머신러닝은 수십억 개의 내역 기록을 사용해 각 고객별로 고유한 취향 프로필을 작성한 다음. 공통적인 취향을 가진 고객을 클러스터로 묶는다. 그런 후 각 고객 클러스터를 대상으로 가장 인기있는 콘텐츠를 실시간으로 추적, 표시해 고객이 현재 인기있는 콘텐츠를 볼 수 있도록 한다. 더 정확한 추천으로 사용률을 높이고 고객 만족도도 높아진다.

○ 자동차 품질 개선은, 재규어 랜드로버(Jaguar Land Rover)의 신형 차량에는 60개의 온보드 컴퓨터가 탑재되며 이 컴퓨터는 2만개 이상의 매트릭스를 기준으로 매일 1.5GB의 데이터를 생성한다. 재규어 랜드로버 엔지니어들은 머신러닝을 사용해 이 데이터에서 고객이 차량을 실제로 어떻게 다루는지를 파악해낸다. 이렇게 얻은 정확한 사용 데이터를 통해 설계자는 부품 고장과 잠재적 안전 위험을 예측할 수 있다. 이는 예상되는 조건에 따라 적절히 차량을 엔지니어링하는데 도움이 된다.

○ 유망 잠재 고객에 대한 집중은, 마케터들이 최적의 판매와 마케팅 기회, 그리고 최적의 제품을 판단하기 위한 도구로 '구매성향(propensity to buy)' 모델을 사용한다. 라우터부터 케이블 TV 박스에 이르기까지 방대한 제품을 보유한 시스코(Cisco)의 마케팅 분석팀은 몇 시간 만에 6만 개의 모델을 교육시키고 1억 6,000만 명의 잠재 고객을 확보했다. 이 팀은 의사결정 트리부터 그래디언트 부스티드머신까지 다양한 기법을 테스트함으로써 모델의 정확도를 대폭 개선했다. 이는 판매량 증가, 무익한 판매 전화 감소, 영업 담당자들의 만족도 향상으로 이어진다.

○ 미디어 최적화는, NBC 유니버설(NBC Universal)은 국제 케이블 TV 배포를 위해 수백 테라바이트 용량의 미디어 파일을 저장한다. 또한 전세계 고객을 대상으로 한 배포를 지원하기 위한 효율적인 온라인 리소스 관리가 필요하다. NBC 유니버설은 머신러닝을 사용해 척도의 조합을 기반으로 각 항목의 미래 수요를 예측한다. 이런 예측을 기반으로 수요가 낮을 것으로 예상되는 미디어를 저렴한 오프라인 스토리지로 옮긴다. 머신러닝을 통한 예측은 파일 수명과 같은 하나의 척도를 기반한 임의 규칙에 비해 훨씬 더 효과적이다. 결과적으로 NBC 유니버설은 고객 만족도를 그대로 유지하면서 전체 스토리지 비용을 줄이고 있다.

○ 의료 보건 서비스 개선은, 병원에게 있어 환자의 재입실은 심각한 문제다. 환자의 건강과 복지도 문제지만 의료 보험 공단과 민간 보험사가 재입실 비율이 높은 병원에 불이익을 주기 때문이다. 따라서 향후 건강한 상태를 유지할 가능성이 충분히 높은 환자만 퇴원시키는 역량이 병원의 재무에 큰 영향을 미치게 된다. 캐롤리나 헬스케어 시스템(Carolinas Healthcare System, CHS)은 머신러닝을 사용해서 환자의 위험 점수를 계산하고 병원 사례 관리자는 이를 바탕으로 퇴원 결정을 내린다. 이 시스템은 각 사례의 위험과 복잡성에 따라 환자에 우선 순위를 부여함으로써 간호사와 사례 관리자의 능률을 높여준다. 그 결과 CHS는 재입실 비율을 21%에서 14%로 낮췄다.

② 이상감지(Fraud detection)

데이터 마이닝의 중요한 응용중의 하나는 예외적인 데이터를 찾는 것이다.

이동전화, 신용카드 등은 이러한 예외적인 데이터의 감지를 통하여 불법적 이용의 가성을 알 수 있다.

미국에서는 불법적인 이동전화 사용으로 인하여 매년 수 백만불의 손해를 이동통신회사들이 입고 있다. 따라서 불법적인 이동전화의 사용은 상대편 이동전화의 등록번호를 입수하여 이 번호를 자신의 이동전화에 입력함으로써 가능하다.

이 때는, 전화사용료는 등록번호의 소유주에게 청구되므로 이러한 불법적 이동전화 사용을 탐지하기 위한 간단한 방법으로는, 같은 번호에서 두 개의 서로 다른 번호로 통화 중이거나, 또는 두 개의 연속된 전화사용 내역에서 시간간격과 전화를 사용한 장소를 비교하여 시간에 비하여 두개의 전화사용 장소가 너무 먼 경우에 불법사용으로 간주할 수 있다.

좀 더 발전된 방법으로는 이동전화 사용자의 평소의 전화이용 패턴을 추정하여 이 패턴을 심하게 벗어나는 통화를 하는 경우 불법적인 이동전화 사용으로 의심할 수 있다.

신용카드의 경우는 크게 두 가지의 불법사용이 있다. 첫째는 신용카드의 분실과 이에 따른 불법 이용이고, 둘째는 신용카드번호와 비밀번호만을 불법적으로 사용하여 인터넷이나 전화를 통한 물품구매를 한 경우가 있다.

첫 번째의 경우에는 그리 큰 문제가 아니다. 대부분의 신용카드를 분실한 사람은 아주 신속하게 신고하기 때문이다.

두 번째 유형의 불법사용은 신용카드 소지자조차도 불법적 사용을 모르는데 그 문제의 심각성이 있다. 이 경우에는 신용카드 소지자의 과거 구매형태를 파악하여 이 형태를 크게 벗어나는 거래인 경우에 불법거래를 의심할 수 있다.

구매형태라는 단어는 상당히 모호한 단어이다. 이렇게 모호하게 정의된 문제를 데이터와 연관된 잘 정의된 문제로 다시 정립하는 것도 데이터 마이닝의 한 부분이다.

신용카드 불법사용을 탐지하기 위해서는 많은 회사들이 신경망을 이용한 데이터 마이닝 기법을 사용한다.

③ 신용도조회(Credit Scoring)

신용등급은 은행에 대출을 하거나 신용카드의 사용한도를 정할 때 고객의 파산위험을 측정하는 방법이다.

가중 되는 금융권의 경쟁은 은행이나 기타 모든 금융회사들이 고객확보에 총력전을 펴고 있다.

고객확보와 파산위험관리는 서로 상호 배치된다. 따라서, 신용등급의 결정은 너무 관대하지도 않게(반대의 경우, 잦은 파산으로 인한 손실이 큼), 또한 너무 엄격 하지도 않게(너무 엄격하면 고객을 다른 금융회사로 뺏길 수 있음) 정해야 하며, 이러한 신용등급의 결정에 데이터 마이닝 방법이 사용된다.

과거에는 금융전문가에 의해 신용등급이 결정되어 왔으나, 현재는 데이터 마이닝을 이용하여 주관적인 견해를 배제하고 데이터를 바탕으로 신용등급을 결정하는 시스템을 많이 사용한다.

④ 고객 유지(Customer Retention)

은행들은 그들의 고객과 되도록 이면 장기간에 걸쳐 거래하기를 원한다. 만약, 오래된 고객이 다른 은행으로 옮기면, 새로운 고객을 유치한다 하여도 은행으로서는 많은 손해를 본다.

그 이유는 일반적으로 오래 거래한 고객들은 신용도 좋고 금전적으로도 여유롭기 때문이다.

이자율의 변동에 그리 민감하지 않으며, 새로운 고객도 가끔 소개 시켜준다. 또한 새로운 고객에게 주는 특혜를 주지 않아도 된다. 이러한 이유로 볼 때, 고객관리는 성공하는 사업으로 가는데 있어서의 중요한 요소가 된다.

우수한 고객을 오래 유치하기 위해서는 은행 측에서 은행을 떠난 고객과 은행과 오래 거래하는 고객과의 차이를 알아야한다. 이러한 차이를 인식한 후에는 현재의 고객 중 단시일 내에 은행을 떠날 가능성이 높은 집단을 추려서 집중적인 홍보와 선전을 함으로써 고객이탈을 예방할 수 있다.

고객 간의 속성의 차이를 알기 위해서는 고객의 과거 거래내역 자료를 바탕으로 데이터 마이닝 기법을 사용한다.

⑤ 회사홍보(Customer profiling)

회사의 제품을 고객에게 알리는 방법은 크게 두 가지가 있다.

불특정 다수를 상대로 신문, 텔레비전, 라디오를 통하여 알리는 대중마케팅(mass marketing)방법과 알리고자 하는 내용에 특별히 관심이 있는 집단을 찾아서 이 집단에게 집중적으로 광고를 하는 직접마케팅(direct marketing)이 있다.

현재와 같이 수많은 제품이 쏟아지고 시장에서의 경쟁이 점점 치열해지는 상황에서는 대중마케팅방법은 그 효력을 점점 상실하고 있다.

수많은 광고를 접하는 고객은 심지어 광고내용은 기억하나 제품은 기억하지 못하는 현상도 자주 일어나게 된다. 자료에 의하면 대중매체를 이용한 광고에 대한 고객의 반응정도(광고가 구매로 이어짐)가 점점 미약해지고 있다.

이와는 대조적으로 직접마케팅은 소수의 구매의사가 높은 집단을 상대로 광고를 함으로 광고비 절약은 물론 광고의 효과도 대중매체를 이용한 광고와는 비교할 수 없을 정도로 높다.

요즘 범람하는 광고용 우편물은 바로 이러한 직접 마케팅의 한 예다.

최근에는 전자우편을 이용한 직접마케팅 기법도 많이 쓰인다.

현재, 엄청난 양의 고객정보가 데이터베이스화 되어 있다. 예로, 인터넷 특정사이트에 회원으로 가입할 때 제공하는 여러 가지 개인 신상자료는 소중한 고객자료가 된다.
이렇게 모여진 거대한 양의 고객 데이터베이스를 데이터 마이닝 기법을 이용하여 분석하면 직접마케팅에 필요한 특정한 고객집단을 찾아 낼 수 있다.
예를 들면, 최근에 등산용 배낭을 구입한 고객이 캠핑용 텐트를 구입할 확률이 높음을 데이터 마이닝을 통하여 발견한 후, 배낭을 구입한 고객을 상대로 새로운 텐트를 집중적으로 광고함으로써 큰 매출액 증가를 이룩할 수 있다.

4) 데이터 마이닝 기법들

데이터 마이닝은 크게 지도학습과 비지도 학습의 2가지로 분류할 수 있다. 지도학습은 목적변수가 존재하는 경우이며, 비지도 학습은 데이터마이닝이 존재하지 않는 경우의 두 가지로 나눌 수 있다.

① 지도학습과 비지도 학습 분류

지도 학습

○ 일반화 선형 모델(Generalized linear models, GLM)

- 선형 회귀(linear regression)의 발전된 형태로, 다양한 가능성 분산과 연결 함수를 지원해 분석가가 더 효과적으로 데이터를 모델링할 수 있도록 한다. 그리드 탐색(grid search)으로 강화된 GLM은 전통적인 통계와 가장 발전된 머신러닝의 조합이다.

○ 의사결정 트리(Decision trees)

- 모집단을 대상 변수에 대해 동질적인 더 작은 조각으로 점진적으로 분할하는 규칙 집합을 학습하는 자율 학습 방법이다.

- 의사결정나무는 데이터의 속성들을 나무구조를 이용하여 서로 비슷한 것들로 데이터를 나누어 간다. 특정한 데이터 항목을 분류하기 위하여는 그 데이터를 가장 잘 분류하는 변수부터 단계적으로 나누어 가며, 마지막에 주어진 의사결정나무는 데이터의 여러 속성들을 분류하는 규칙집합으로 해석될 수 있다. 신경망과는 달리 의사결정나무는 그 구조의 단순성과 변수 상호간의 관계를 볼 수 있는 장점에 의하여 현재 널리 쓰이고 있다.

랜덤 포레스트(Random forests)

널리 사용되는 총체적 학습 방법으로, 다수의 의사결정 트리를 학습한 다음, 트리 전반에 걸친 평균을 구해 예측을 산출한다. 이 평균 프로세스는 일반화 가능한 솔루션을 제공하며 데이터의 불규칙 잡음(random noise)을 걸러내는 효과가 있다.

○ 그래디언트 부스팅 머신(Gradient boosting machine, GBM):

- 의사결정 트리의 시퀀스 교육을 통해 예측 모델을 생성하는 방법으로, 연속되는 트리가 이전 트리의 예측 오류를 수정해 나간다.

○ 딥 러닝(Deep learning)

- 데이터의 고수준 패턴을 복합적인 다계층 네트워크로 모델링하는 방법으로, 문제를 모델링하는 가장 일반적인 방법이며 머신러닝의 가장 어려운 문제를 해결할 잠재력을 지녔다.

○ 순차패턴 (Sequential patterns)

- 동시에 구매될 가능성이 큰 상품 군을 찾아내는 연관성측정(Associations)에서 시간이라는 개념이 포함되어 순차적인 구매 가능성이 큰 상품 군을 찾아내는 것
- 이벤트나 행동의 시간적인 순서를 나타내는 규칙이다. 예를 들면, "새 컴퓨터를 구입한 사람들 중 25%는 그 다음달에 레이저프린터를 구입할 것이다", "휴대폰은 구입한 40대 남성고객이 3개월 이내에 다시 핸즈프리를 구입하고, 12개월 이내에 배터리를 구입하는 경향이 있다." 와 같은 연관 규칙을 찾아내는 것이다.
- 순차적 패턴발견에서의 연관 규칙 A→B 은 "상품 A가 구매되면 일정시간의 경과한 다음에는 상품 B가 구매된다."라고 해석된다.
- 순차적 패턴발견은 구매의 순서가 고려되어 상품 간의 연관성이 측정되고 이의 정도에 따라 유용한 연관 규칙을 찾는 기법이다.
- 위의 연관성측정에서의 데이터 형태에서 각각의 고객으로부터 발생한 구매의 시점에 대한 정보가 포함되어져야 한다.

비지도 학습

○ 클러스터링(Clustering)

- 개체를 다수의 메트릭스에서 상호 유사한 세그먼트 또는 클러스터로 그룹화하는 기법으로, 고객 세분화가 클러스터링의 실제 예다. 클러스터링 알고리즘은 무척 다양한데, 가장 널리 사용되는 것이 k-평균(k-means)이다.

○ 비정상 탐지(Anomaly detection)

- 예상치 못한 이벤트 또는 결과를 식별하는 프로세스. 보안, 사기 등의 분야에서는 모든 거래를 철저하게 조사하기란 불가능하므로 가장 비일반적인 거래에 체계적으로 플래그를 지정해야 한다. 앞서 지도 러닝에서 언급한 기법인 딥 러닝도 비정상 탐지에 사용할 수 있다.

○ 차원 축소(Dimension reduction)

- 고려 대상 변수의 수를 줄이는 프로세스. 조직이 더 많은 데이터를 캡처할수록 예측에 사용 가능한 예측 변수(또는 특성)의 수도 급격히 증가한다. 특정 문제에 대해 가치 있는 정보를 제공하는 데이터를 식별하는 것만 해도 상당한 작업이다. 주성분 요소 분석(Principal components analysis, PCA)은 일련의 원시 특성을 평가해 이를 상호 독립적인 인덱스로 축소한다.

5) 의사결정나무 알고리즘

① 개요

결정 트리 학습법(decision tree learning)은 어떤 항목에 대한 관측값과 목표값을 연결시켜주는 예측 모델로써 결정 트리를 사용한다.

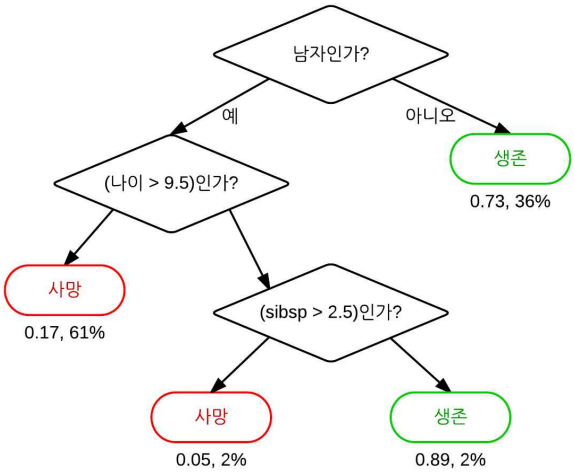
- 결정 트리(decision tree)는 의사 결정 규칙과 그 결과들을 트리 구조 형태로 도식화한 의사 결정 지원 도구의 일종임
- 결정 트리는 운용 과학, 그 중에서도 의사 결정 분석에서 목표에 가장 가까운 결과를 낼 수 있는 전략을 찾기 위해 주로 사용된다.
- 지도학습(분류 및 예측)의 데이터 마이닝 기법이다.
- 적용 결과에 의해서 If-then으로 표현되는 규칙이 생성된다.

의사결정나무 방법론은 의사결정에서 나무의 가지를 가지고 목표와 상황과의 상호 관련

성을 나타내어 최종적인 의사결정을 하는 불확실한 상황 하의 의사결정 분석 방법을 이야기한다.

의사결정자의 통제 아래 있는 결정 노드(decision nodes)와 의사결정자의 통제 밖에 있는 기회 노드(chance nodes)로 구성된 의사결정 문제의 논리적 구조를 이용해 의사결정을 하는 방법을 말한다.

의사결정자는 각 기회 노드에서 초래될 불확실한 사상(事象)들과 그것이 가져올 모든 가능한 결과에 대해 확률을 부여하는 방식으로 최적 대안을 선택하게 된다. 이 때 부여되는 확률은 과거의 경험적 데이터, 의사결정자들의 주관적 판단, 전문가의 견해 등을 토대로 여러 가지 기법과 절차를 혼합적으로 사용해 얻게 된다.



[의사결정나무 예]

② 의사결정나무의 적용

의사결정나무는 예측력과 해석력이 필요한 상황에 사용한다.

예측력이 필요한 경우

○ 예를 들어서 CRM에서 홍보책자 발송회사가 기대집단의 사람들이 가장 많은 반응을 보일 고객 유치방안을 위한 예측에 사용된다.

해석력이 필요한 경우

○ 많은 분야에서 결정을 내리게 된 이유를 설명하는데 활용된다.

○ 예를 들어 은행의 대출심사 결과 부적격 판정이 나온 경우 고객에게 부적격 이유를 설명해야 하는 경우 등이 있다.

③ 의사결정나무의 장점

- 이해하기 쉬운 규칙을 생성시켜 준다.
- 분류작업이 용이하다.
- 연속형변수와 범주형변수를 모두 다 취급할 수 있다.
- 가장 좋은 변수를 명확히 알아낸다. (SAS E-Miner의 Variable selection 노드에서 카이제곱 통계량을 이용한 변수 선택은 바로 CHAID를 이용한 것이다.)
- 이상치에 덜 민감하다.
- 모형의 가정 (선형성, 등분산성 등)이 필요 없다. 즉, 비모수적 모형이다.

④ 의사결정나무의 단점

- 목표변수가 연속형인 회귀모형에서는 그 예측력이 떨어진다.
- 나무가 너무 깊은 경우에는 예측력의 저하뿐 아니라 해석도 하기가 쉽지 않다.

- 계산량이 많을 수 있다.
- 비사각영역에서 문제가 있다.
- 결과가 불안정하다.
- 선형성 또는 주효과의 결여가 발생할 수 있다.

⑤ 의사결정나무 형성과정

의사결정 나무는 성장 - 가지치기 - 타당성 평가 - 해석 및 예측의 순으로 진행된다.

나무의 성장(growing)

○ 각 노드에서 적절한 최적의 분리규칙을 찾아서 나무를 성장시킨다.

○ 정지규칙을 만족하면 성장을 중단한다.

가지치기(pruning)

○ 분류오류를 크게 할 위험이 높거나 부적절한 추론규칙을 가지고 있는 가지를 제거

○ 또한 불필요한 가지를 제거

타당성 평가

○ 이익도표(gain chart)나 위험도표(risk chart) 또는 검증용 자료의 사용 또는 교차타당성(cross validation) 등을 이용하여 의사결정나무를 평가한다.

해석 및 예측

○ 구축된 나무모형을 해석하고 예측모형을 설정한다

⑥ 목적변수의 종류에 따른 의사결정나무의 분류

목적 변수가 명목 변수인 경우에 분류트리라고 하고, 목적변수가 수인 경우를 회귀트리라고 한다.

분류트리

○ 잎은 클래스 라벨을 나타내고, 가지는 클래스 라벨과 관련있는 특징들의 논리곱을 나타낸다.

회귀트리

○ 평균을 기준으로 평균보다 높은 그룹과 낮은 그룹으로 분할하는 분할조건을 탐색하는 등 적절한 분할 조건을 찾는다.

⑦ 알고리즘에 따른 분류

CART

C4.5

CHAID

CART, CHAID, C4.5의 의사결정나무 알고리즘들은 분할조건을 계산하는 방법과 어떻게 가지치기를 멈추어 가장 적절한 크기의 나무를 최종모형으로 결정하는 지에 대한 방법이 다르다.

⑧ C4.5

호주의 연구원 J.Ross Quinlan에 의하여 개발되었다.

초기버전은 ID 3 (Iterative Dichotomizer 3)로 1986년에 개발되었는데 CART와는 다르게 각 마디에서 다지분리 (multiple split)가 가능하다.

또한 범주형 입력변수에 대해서는 범주의 수만큼 분리가 일어나고, 불순도함수로 엔트로피 지수를 사용한다. 가지치기를 사용할 때 학습자료를 사용한다.

C4.5를 언급할 때 따라다니는 지수는 엔트로피(Entropy)이다. CART와 마찬가지로 C4.5

도 마디의 순수함을 재는데 비트(bit)개념을 이용한다.
비트 개념을 다음의 예로 이해할 수 있는데, 만약 8개의 카테고리로 이루어진 마디가 있다면 그 각각의 카테고리를 표시할 비트는 총 log28 = 3 개가 필요하게 된다. (000은 제1 카테고리, 001은 제2 카테고리,…. ,111은 제8 카테고리로 표현 가능하다)
그 마디에 가지가 쳐져서 4개의 카테고리가 있는 자식마디가 생기면 그 자식마디는 log24 = 2 개의 비트만 있으면 표현이 가능하다. 결국 가지가 쳐지면서 3개가 필요했던 비트가 2개로 줄어들어 엔트로피 1의 이득을 본 것이다. C4.5에서는 이런 비트 개념이 수리적으로 보다 더 정교하게 지수화된다.

⑨ CHAID
CHAID는 CHI-SQUARE AUTOMATIC INTERACTION DETECTION의 약자이다.
CHAID는 Chi-squared Automatic Interaction Detection의 준말로서 1975년 J. A. Hartigan이 발표했고 1963년 J. A. Morgan과 N. A. Souquist에 의해 서술된 AID의 후신이다.
CHAID는 가지치기를 하지 않고 나무를 적당한 크기에서 성장을 중지시킨다.
입력변수가 반드시 범주형이어야하고 불순도함수로는 카이제곱통계량을 사용한다.
CHAID에서는 분할표를 바탕으로 Q값을 계산한다면 CART에서는 Gini 지수를 이용한다.
Gini 지수는 마디(node)의 순수함의 정도를 나타내는 척도인데, 어느 한쪽의 카테고리로만 구성되어 있는 마디일수록 순수하다고 판단한다. 어느 한쪽의 카테고리가 지배적으로 차지하고 있는 마디일수록 즉, 순수할수록 Gini 지수의 값은 작아진다.
의사결정나무는 Gini 지수가 줄어지는 방향으로 움직이는 것이며 Gini 지수의 값을 가장 많이 줄여주는 변수가 영향을 가장 많이 끼치는 변수가 된다. 그리고 이 변수를 기준으로 가지는 쳐진다.

6) 데이터 마이닝 툴
① 개요
많은 툴들이 데이터 마이닝 기능을 지원하고 있으며, 상용프로그램과 오픈소스의 무료 프로그램 들이 공개되어 있다.
상용프로그램으로는 매트랩(matlab)이 유명하며, 오픈소스로는 파이썬, R, weka 등이 있다.
오픈소스는 따로 비용을 지불하지 않고, 프로그램을 인터넷에서 다운로드 받아서 바로 사용할 수 있다.

② MATLAB
MATLAB은 Mathwork회사의 제품으로, 수치 해석, 행렬 연산, 신호 처리 및 간편한 그래픽 기능 등을 통합하여 고성능의 수치 계산 및 결과의 가시화 기능을 제공하는 인터프리터 프로그램이다.
MATLAB은 그 이름이 말하듯이 행렬 또는 벡터를 기본 자료로 사용하여 기능을 수행하는 계산 환경을 제공한다.
MATLAB은 기본적으로 행렬 자료를 다루기 때문에 차원화(dimensioning)가 필요하지 않으며 통상적인 프로그래밍 언어들을 사용하여 프로그램을 작성하지 않고도 쉽게 수치 계산을 수행할 수 있다.
MATLAB의 가장 큰 특징은 M-file을 사용함으로써 특정한 해를 구하는데 필요한 응용 프로그램들을 손쉽게 작성할 수 있다는 점이다.

M-file이란 매크로 파일로서 해석기(interpreter)방식으로 수행되며 사용자가 직접 작성할 수 있는 프로그램이다. 기본적인 내부 명령들뿐만 아니라 다른 M-file들로 불러서 사용할 수 있으며, 특정한 문제를 풀기 위하여 사용자가 직접 손쉽게 M-file을 작성하여 사용할 수 있다는 점이 가장 큰 특징이다.
도구상자(toolbox)란 이러한 M-file들을 응용 분야별로 모아 놓은 것으로서, 그 종류로는 신호처리, 제어계의 설계, 동적계의 모사, 신경회로망 등이 있다. 도구상자에 이미 만들어진 여러 함수들을 이용함으로써 쉽게 공학 문제들을 해석할 수 있다.

③ WEKA
웨카(WEKA)란 자바(Java) 언어로 개발된 오픈 소스 데이터 마이닝 프로그램이다. 이는 뉴질랜드의 와이카토 대학의 '이안 위튼' 교수팀에 의해 개발 되어지고 있는 프로그램으로 지식분석용 와이카토 환경이라는 뜻의 "Waikato Environmet for Knowledge Analysis"의 앞 글자를 따서 이름 지어졌다. 이것은 프로그램 전체에 자바(Java) 언어로 된 소스 코드를 제공하는데 여기서 자바는 완성도가 높고 웹과의 연동성이 좋으며 보안이 뛰어난 장점을 가진다.
웨카(WEKA)는 다양한 분석 알고리즘들을 제공하여 유용하게 사용될 수 있다. 구체적으로 데이터 전처리, 분류, 클러스터링, 연관 규칙, 시각화 등 데이터 마이닝 분석 알고리즘을 포함하고 있다.
자바(Java)는 썬 마이크로시스템즈의 제임스 고슬링(James Gosling)과 다른 연구원들이 개발한 객체 지향적 프로그래밍 언어이며, 오라클에서 무료로 제공하고 있다. 처음에는 가전제품 내에 탑재해 동작하는 프로그램을 위해 개발했지만 현재 웹 애플리케이션 개발에 가장 많이 사용하는 언어 가운데 하나이고, 모바일 기기용 소프트웨어 개발에도 널리 사용하고 있다. 이러한 자바는 문법적인 특성이 C 언어와 비슷하다.
자바를 다른 컴파일언어와 구분 짓는 가장 큰 특징은 컴파일 된 코드가 플랫폼이 독립적이라는 점이다. 자바 컴파일러는 자바 언어로 작성된 프로그램을 바이트 코드라는 특수한 바이너리 형태로 변환한다. 이러한 바이트 코드를 실행하기 위해서는 JVM(자바 가상 머신, Java Virtual Machine)이라는 특수한 가상 머신이 필요한데, 이 가상 머신은 자바 바이트 코드를 어느 플랫폼에서나 동일한 형태로 실행시킨다. 때문에 자바로 개발된 프로그램은 CPU나 운영 체제의 종류에 관계없이 JVM을 설치할 수 있는 시스템에서는 어디서나 실행할 수 있으며, 이 점이 웹 애플리케이션의 특성과 맞아 떨어져 폭발적인 인기를 끌게 되었다.

④ 파이썬
파이썬(Python)은 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어이다. 귀도는 파이썬이라는 이름을 자신이 좋아하는 코미디 쇼인 "몬티 파이썬의 날아다니는 서커스(Monty Python's Flying Circus)"에서 따왔다고 한다.
파이썬의 사전적인 의미는 고대 신화에 나오는 파르나소스 산의 동굴에 살던 큰 뱀을 뜻하며, 아폴로 신이 델파이에서 파이썬을 퇴치했다는 이야기가 전해지고 있다. 대부분의 파이썬 책 표지와 아이콘이 뱀 모양으로 그려져 있는 이유가 여기에 있다.
파이썬은 우리나라에서는 아직 대중적으로 사용되고 있지 않지만 외국에서는 교육 목적뿐 아니라 실무에서도 많이 사용되고 있다. 그 대표적인 예가 바로 구글인데, 구글에서 만들어진 소프트웨어의 50% 이상이 파이썬으로 만들어졌다는 이야기도 있다. 이외에도 많이 알려진 예 중 몇가지로 파일 동기화 서비스인 드롭박스(Dropbox), 쉽고 빠르게 웹

개발을 할 수 있도록 도와주는 프레임워크인 장고(Django) 등이 있다.
또한 파이썬 프로그램은 공동 작업과 유지 보수가 매우 쉽고 편하기 때문에 이미 다른 언어로 작성된 많은 프로그램과 모듈들이 파이썬으로 재구성되고 있다. 국내에서도 그 가치를 인정받아 사용자층이 더욱 넓어지고 있고, 파이썬을 이용해 프로그램을 개발하는 기업체들 또한 늘어 가고 있는 추세이다.

⑤ R
R은 통계 컴퓨팅 및 그래픽을 위한 언어와 환경이다. 이것은 벨 연구소(이전에 AT & T, 지금은 루슨트 테크놀로지)의 존 챔버스(John Chambers)와 동료들이 개발한 언어 및 환경인 S와 유사한 GNU 프로젝트이다. R은 S의 다른 구현으로 생각될 수 있다. 몇 가지 중요한 차이가 있지만, S용으로 작성된 많은 코드는 R에서 변경없이 실행 할 수 있다.
R 시스템은 미국 AT&T사의 Bell 연구소에서 개발한 S언어를 기반으로, 통계분석 모듈 및 다양한 그래픽 도구로 이루어져 있다.
초기 R버전은 오클랜드 대학의 Ross Ihaka와 Robert Gentleman이 발전시켰고 전 세계의 R 개발 팀에 의해 꾸준히 확장, 발전되는 중이다.
R은 다양한 통계 (선형 및 비선형 모델링, 고전 통계적 테스트, 시계열 분석, 분류, 클러스터링, ...) 및 그래픽 기술을 제공하고, 확장성이 뛰어나다. S언어는 종종 통계방법론 연구를 위한 선택 수단이고, R은 그 활동에 참여하기 위한 오픈 소스 경로를 제공한다.
R시스템은 S언어를 기반으로 한 통계분석 시스템인 S-PLUS와 달리 누구나 자유롭게 쓸 수 있으며 GNU규약 하에 소스가 개방되어 꾸준히 발전하고 있다. R언어는 객체 지향 프로그램 언어로서 일련의 데이터 처리 및 분석 작업을 대화형으로 처리할 수 있다. 또한 뛰어난 그래픽 기능을 가지고 있어서 그래프를 이용한 자료 분석 작업도 매우 쉽게 할 수 있다.
R의 강점 중 하나는 손쉽게 필요한 곳에 수학기호와 수식을 포함한, 잘 디자인된 출판물 수준의 plot을 생산한다는 것이다. 그래픽 기본값에 작은 디자인 선택을 해야하는 세심한 주의가 필요하지만, 사용자가 모든 제어를 할 수 있다.
R은 소스 코드 형태로 자유 소프트웨어 재단(Free Software Foundation)의 GNU GPL(General Public License)에 의거하여 자유 소프트웨어로 사용할 수 있다. R은 다양한 종류의 UNIX 플랫폼과 유사 시스템(FreeBSD와 Linux 등), 윈도우와 맥 OS에서 컴파일되고 실행된다.

⑥ 개발언어 선택
다양한 개발 환경에 대한 비교를 수행한 결과 성능면에서는 크게 차이가 없으나 시스템이 파이썬 언어로 이루어져 있는 바, 파이썬 언어로 개발을 하는 것이 가장 효율적일 것으로 판단되었다.
이에 개발 언어를 파이썬으로 선택하였다.

7) 의사결정나무 알고리즘과 재난전조예측
① 개요
다양한 검토를 걸쳐서 지하철 재난전조 예측을 위해서 데이터마이닝 기법 중 의사결정나무 알고리즘을 이용해서 실시간 센싱 데이터를 이용한 재난전조예측모델을 작성하기로 했다.
재난전조예측모델은 일정기간 축적된 센싱데이터를 가지고 기계학습을 통해서 예측 모델

을 작성한 후, 실시간 센싱 데이터에서 모델에 적용하기 위한 정보를 가공 추출한 후에 모델에 적용하여 정상, 이상, 재난 판정을 받는 프로세스를 실시간으로 반복한다.

② 파이썬 - 의사결정나무 개발 설정

파이썬에서 의사결정나무 알고리즘을 구현하기 위해서 scikit-learn 모듈을 사용하였다. 이 모듈은 아래와 같은 모듈의 추가 설치를 요구한다.

- ○ pandas
- ○ matplotlib

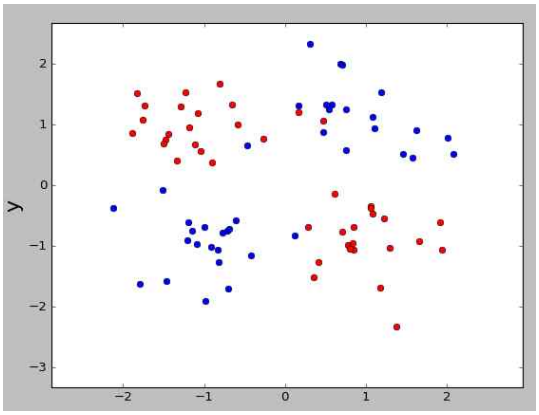
③ 샘플 데이터를 이용한 scikit-learn 테스트

아래의 두개의 입력변수 x, y에 출력변수는 2 클래스(0,1)의 샘플 데이터를 이용해서 의사결정나무를 테스트하였다.

x	y	class
-2.12158	-0.36567	0
-1.79727	-1.61852	0
-0.71739	-0.73818	0
-0.83066	-1.05879	0
-1.14532	-0.75062	0
-1.19392	-0.60669	0
-1.51019	-0.07176	0
-1.20439	-0.89437	0
-1.08969	-0.95712	0
0.120336	-0.82279	0
-0.61349	-0.57486	0
-0.42223	-1.15616	0
-1.46094	-1.56919	0
-0.97911	-1.90755	0
-0.81742	-1.26213	0
-0.69772	-1.69373	0
-0.90938	-1.01461	0
-0.68479	-0.71548	0
-1.0011	-0.67721	0
-0.77379	-0.78149	0
0.67965	2.009711	0
1.08799	1.128682	0

[샘플 데이터]

샘플데이터는 다음과 같은 분포를 하고 있다. 붉은 색과 파란색은 각각의 클래스를 나타내고 있으며, x, y 축은 입력데이터의 값이다.



[샘플 데이터의 분포]

의사결정나무 구현을 위해서 다음과 같은 코드를 작성하였다.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import tree


#preprocess
df = pd.read_csv('xor_simple.csv')

# pylab inline
def x_y_species(df, species_name):
    x_array = df[df['class'] == species_name]['x'].values
    y_array = df[df['class'] == species_name]['y'].values
    return x_array, y_array

# plot sample data
def data_plot(df):
x_array, y_array = x_y_species(df, 0)
plt.plot(x_array, y_array, 'bo')

x_array, y_array = x_y_species(df, 1)
plt.plot(x_array, y_array, 'ro')
plt.xlabel('x', fontsize=20)
plt.ylabel('y', fontsize=20)
plt.axis("tight", fontsize=20)
x_min, x_max = x_array.min() - 1, x_array.max() + 1
y_min, y_max = y_array.min() - 1, y_array.max() + 1
plt.xlim((x_min, x_max))
plt.ylim((y_min, y_max))
plt.show()

#data_plot(df)

# test decision tree
data_array = df[['x', 'y']].values
class_array = df['class'].values

clf = tree.DecisionTreeClassifier()
clf = clf.fit(data_array, class_array)

result = clf.predict([[2., 1.], [1., -0.5]])
print(result)

from sklearn.externals.six import StringIO

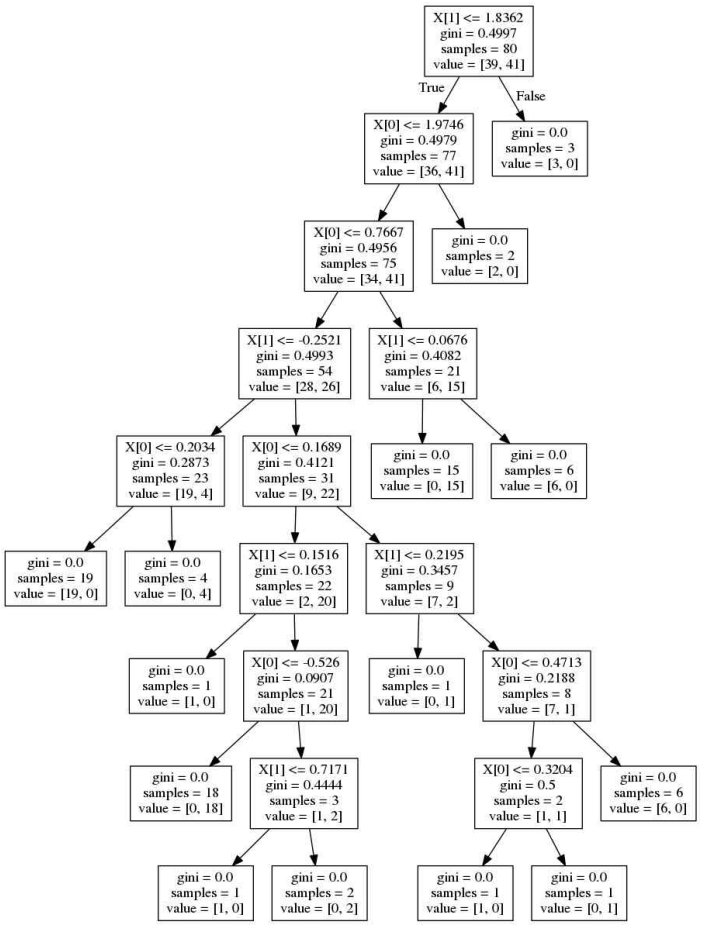
with open("xor_simple.dot",'w') as f:
f = tree.export_graphviz(clf, out_file=f)
#f.close()

# transform ruleset to human-readable format

def transform_ruleset(inputFile):
import os
#from graphviz import Source
file = open(inputFile,'r')
text=file.read()
return text

ruleset_text = transform_ruleset('xor_simple.dot')
```

프로그램 실행 결과 다음과 같은 의사결정나무를 도출하였다.



[테스트 결과 의사결정나무]

위의 의사결정나무는 다음과 같은 코드로 출력되며, 이 결과를 가지고 시스템에서 룰셋 생성에 활용할 수 있다.

```
digraph Tree {
node [shape=box] ;
0 [label="X[1] <= 1.8362₩ngini = 0.4997₩nsamples = 80₩nvalue = [39, 41]"] ;
1 [label="X[0] <= 1.9746₩ngini = 0.4979₩nsamples = 77₩nvalue = [36, 41]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[0] <= 0.7667₩ngini = 0.4956₩nsamples = 75₩nvalue = [34, 41]"] ;
1 -> 2 ;
3 [label="X[1] <= -0.2521₩ngini = 0.4993₩nsamples = 54₩nvalue = [28, 26]"] ;
2 -> 3 ;
4 [label="X[0] <= 0.2034₩ngini = 0.2873₩nsamples = 23₩nvalue = [19, 4]"] ;
3 -> 4 ;
5 [label="gini = 0.0₩nsamples = 19₩nvalue = [19, 0]"] ;
4 -> 5 ;
6 [label="gini = 0.0₩nsamples = 4₩nvalue = [0, 4]"] ;
4 -> 6 ;
7 [label="X[0] <= 0.1689₩ngini = 0.4121₩nsamples = 31₩nvalue = [9, 22]"] ;
3 -> 7 ;
8 [label="X[1] <= 0.1516₩ngini = 0.1653₩nsamples = 22₩nvalue = [2, 20]"] ;
7 -> 8 ;
9 [label="gini = 0.0₩nsamples = 1₩nvalue = [1, 0]"] ;
8 -> 9 ;
10 [label="X[0] <= -0.526₩ngini = 0.0907₩nsamples = 21₩nvalue = [1, 20]"] ;
8 -> 10 ;
11 [label="gini = 0.0₩nsamples = 18₩nvalue = [0, 18]"] ;
10 -> 11 ;
12 [label="X[1] <= 0.7171₩ngini = 0.4444₩nsamples = 3₩nvalue = [1, 2]"] ;
10 -> 12 ;
13 [label="gini = 0.0₩nsamples = 1₩nvalue = [1, 0]"] ;
12 -> 13 ;
```

```
14 [label="gini = 0.0₩nsamples = 2₩nvalue = [0, 2]"] ;
12 -> 14 ;
15 [label="X[1] <= 0.2195₩ngini = 0.3457₩nsamples = 9₩nvalue = [7, 2]"] ;
7 -> 15 ;
16 [label="gini = 0.0₩nsamples = 1₩nvalue = [0, 1]"] ;
15 -> 16 ;
17 [label="X[0] <= 0.4713₩ngini = 0.2188₩nsamples = 8₩nvalue = [7, 1]"] ;
15 -> 17 ;
18 [label="X[0] <= 0.3204₩ngini = 0.5₩nsamples = 2₩nvalue = [1, 1]"] ;
17 -> 18 ;
19 [label="gini = 0.0₩nsamples = 1₩nvalue = [1, 0]"] ;
18 -> 19 ;
20 [label="gini = 0.0₩nsamples = 1₩nvalue = [0, 1]"] ;
18 -> 20 ;
21 [label="gini = 0.0₩nsamples = 6₩nvalue = [6, 0]"] ;
17 -> 21 ;
22 [label="X[1] <= 0.0676₩ngini = 0.4082₩nsamples = 21₩nvalue = [6, 15]"] ;
2 -> 22 ;
23 [label="gini = 0.0₩nsamples = 15₩nvalue = [0, 15]"] ;
22 -> 23 ;
24 [label="gini = 0.0₩nsamples = 6₩nvalue = [6, 0]"] ;
22 -> 24 ;
25 [label="gini = 0.0₩nsamples = 2₩nvalue = [2, 0]"] ;
1 -> 25 ;
26 [label="gini = 0.0₩nsamples = 3₩nvalue = [3, 0]"] ;
0 -> 26 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

6. M2M 기반 SOP 모듈 β버전 개발

1) 개요

데이터 마이닝 기술에 바탕한 패턴인식 기능을 통해서 센싱 데이터의 실시간 해석 기능에 의한 재난전조 예측 기능 개발을 목표로 한다.

이상데이터에 대한 자동 필터링 기능과 더불어 센싱 디바이스의 설치환경을 고려한 상황인지 기능을 통해서 안정적이며, 정확한 재난 상황 판단이 가능한 M2M알고리즘 개발을 목표로 한다.

개별 센서 데이터의 상태에 기반한 재난 전조 예측 알고리즘으로 센서 데이터의 학습을 바탕으로 재난 종류와 재난 발생 위치를 예측한다.

기존의 결정론적인 재난 판단 시스템, 즉 일정한 단순 파라미터에 의해서 재난 상황을 판단하는 재난 감지 모델의 오작동 및 정확도의 한계를 극복하는 재난 상황 판단 모델을 개발함으로써 보다 향상된 방재 시스템을 개발을 지향한다.

2) 성능목표

- 기존기술 수준

○ 기존의 재난 감지 시스템은 임계치에 기반한 단순 로직에 의한 재난 상황 기능에 의해서 재난을 감지한다.

○ 센서 이상 등에 의한 이상치에 대해서 오작동의 가능성이 크다.

- 개발기술 수준

○ 데이터 마이닝 기술에 바탕한 패턴인식 기능을 통해서 센싱 데이터의 실시간 해석 기능에 의한 재난전조 예측 기능 개발을 목표로 한다.

○ 이상데이터에 대한 자동 필터링 기능과 더불어 센싱 디바이스의 설치환경을 고려한 상황인지 기능을 통해서 안정적이며, 정확한 재난 상황 판단이 가능한 M2M알고리즘 개발을 목표로 한다.

3) 개발내용

파이썬의 scikit-learn 모듈을 통해서 개발한 의사결정나무에 아래 센서 애뮬레이터 데이터를 적용해서 의사결정나무를 도출하였다.

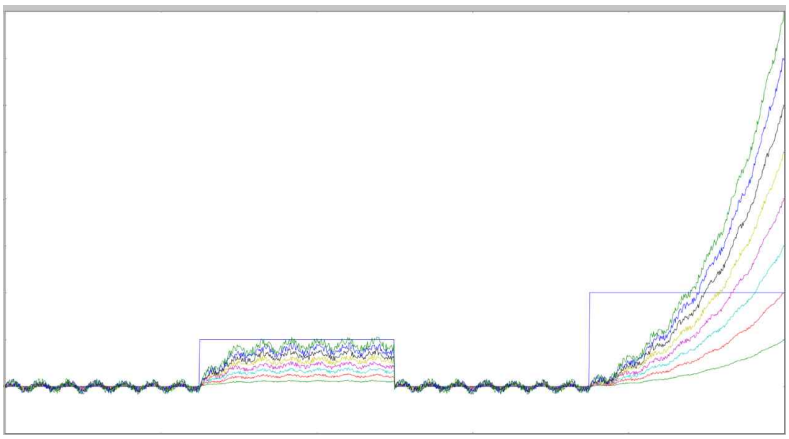
또한 SOP 모듈에 사용자가 설정한 룰셋도 적용될 수 있도록 사용자 지정 룰셋입력 기능을 개발하였다.

전체적으로는 사용자 지정 룰셋이 먼저 적용되고, 이후로 의사결정나무 알고리즘에 의해서 도출된 룰셋이 적용된다.

4) 센서 에뮬레이션

데이터를 계측하기 전에 에뮬레이션 기능을 개발하여 실 데이터 대신 사용하였다.

데이터는 정상- 이상-정상-재난구간의 3개 패턴의 애뮬레이션 데이터가 반복적으로 발생하도록 설정하였으며, 데이터 패턴은 아래와 같다.



[센서 에뮬레이션 데이터]

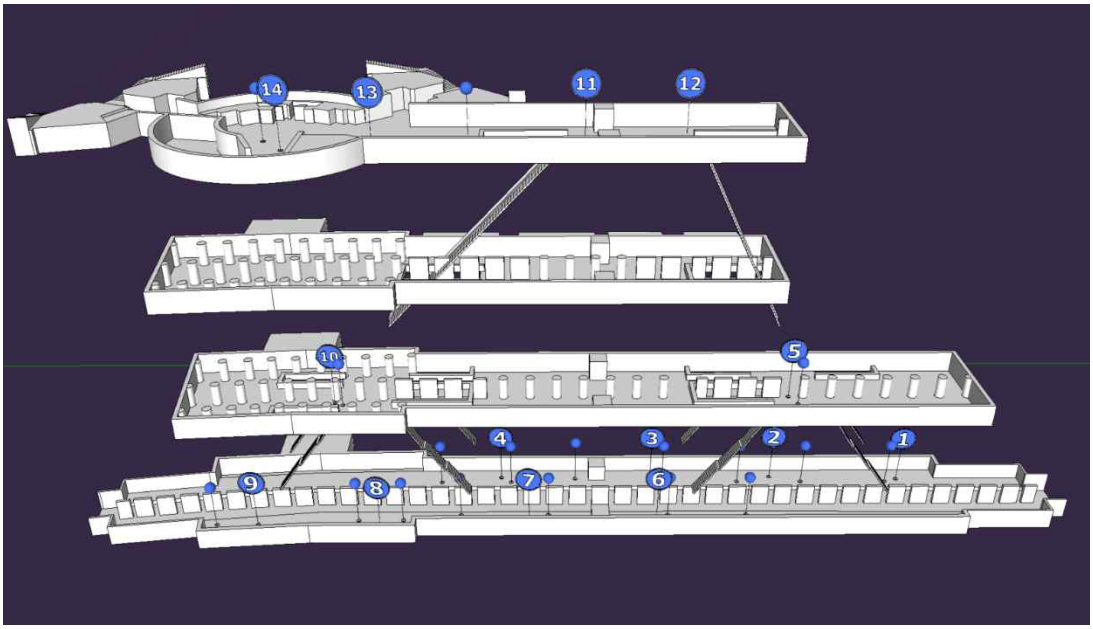
총 18개의 센서가 들어가 있으며 세부 정보는 다음과 같다.

- ○ 최종적으로 sensor Data가 리턴값
- ○ sensor 데이터의 구조는 18 * 9 * 1000의 삼중리스트임
- ○ 18 - 센서를 나타냄
- ○ 9 - 데이터 종류를 나타냄(첫번째가 상태(0,1,2)이고 두번째 이후가 데이터임)
- ○ 1000 는 데이터 길이를 나타냄

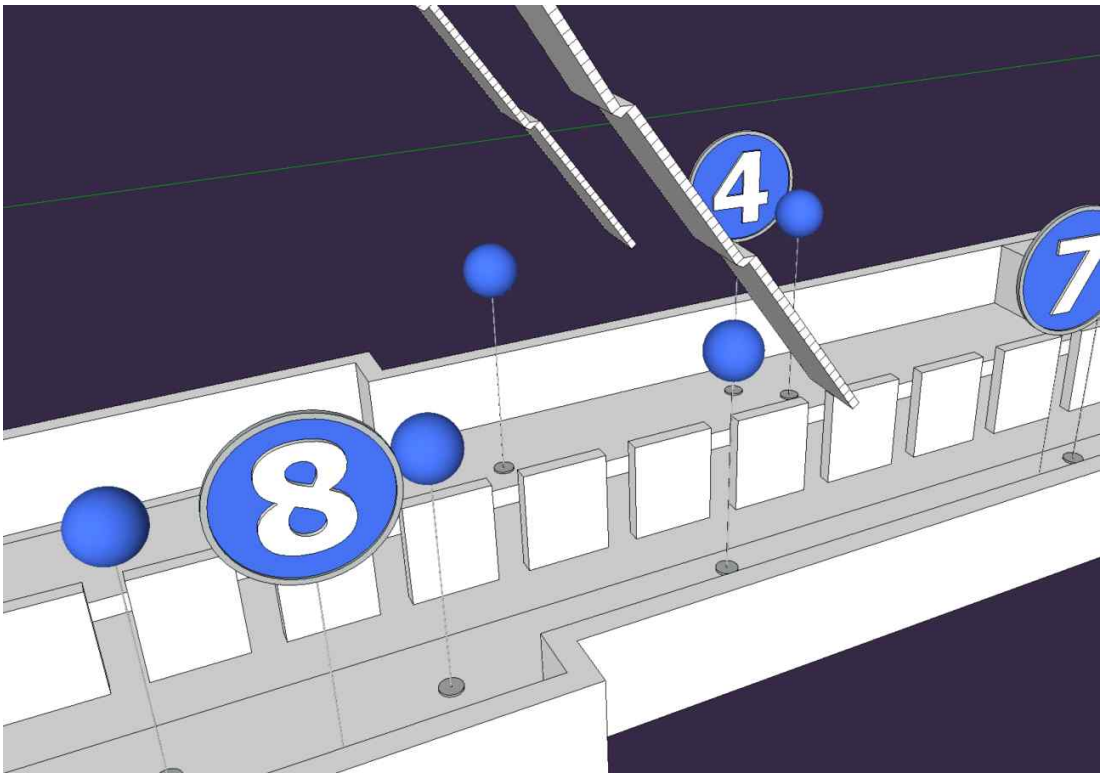
5) 상황전파 및 키퍼 인지

재난전조 예측 모듈에 의해서 재난상황인지가 이루어지면 마스터는 각 센서에 현재 상황을 전파하고 발동 시나리오를 통보한다.

이를 통해서 각 키퍼들은 각각의 위치에서 자신이 수행하여야할 SOP를 시나리오에 따라서 선택해서 수행하게 된다.



[시스템 내 키퍼와 센서 위치 표시 POI]



[시스템 내 키퍼와 센서 위치 표시 상세]

7. 대피 경로 분석/대피 유도 모듈 β버전 개발

1) 개요

재난 발생 시 M2M 기반 재난대응 SOP에 따라서 최적 피난 경로 설정 및 피난 유도를 위한 그래프 기반 피난 유도 알고리즘 및 프로그램 개발을 수행하였다.

지하철 역사의 공간 정보를 기반으로 위험도를 최소화하는 피난 경로 산출 기능을 구현하였다.

2) 성능목표

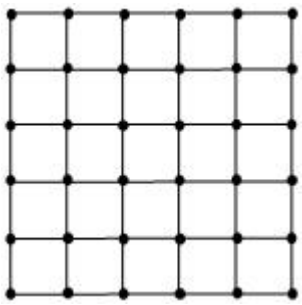
○ 기존기술 수준

기존 지하철 공간 피난 대피는 피난 공간에 대한 정량적 분석 기반이 아닌, 경험적 판단에 의한 경로 설정이 대부분이였다.
이러한 경험적 경로 설정에 대한 객관적 근거 제시가 부족했으며, 최적 경로인지에 대한 분석 기법이 부재하였다.

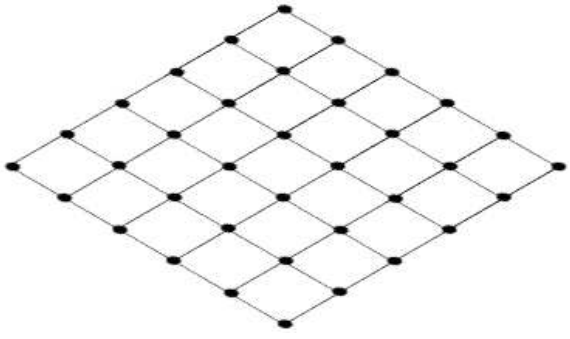
○ 개발기술 수준
지하철 역사의 최적피난경로설정을 위한 정량적 공간분석기능 개발 및 공간분석 결과 기반 피난 유도 경로 표지 SW모듈 개발을 목표로 하였다.

3) 공간의 그래프화

지하철 역사에 재난 발생 시 최적의 경로로 피난을 유도해야 하나, 최적의 피난 경로를 정량적으로 구하는 방법은 그렇게 많지 않다.
일반적인 방법으로는 건물 내 피난 시뮬레이션 SW를 사용하는 방법이 있으나, 이는 대상 공간의 크기에 따라서 많은 소요시간이 걸릴 수 있다.
이에 본 연구에서는 시뮬레이션 이외에 그래프 이론(graph theory)를 이용해서 최적 피난 경로를 찾을 수 있는 알고리즘을 개발하였다.
다음의 단계를 거쳐서 공간은 그래프화를 진행한다.
우선 단순 격자 그래프를 이용해서 평면을 매핑할 수 있다. 대상 공간 전체를 격자 그래프로 치환하는 과정을 거쳐서 유클리드 평면이 그래프 평면이 된다.
각 노드들은 공간 상에서의 지점을 나타내며, 각 간선(edge)는 공간의 연결성을 나타낸다.

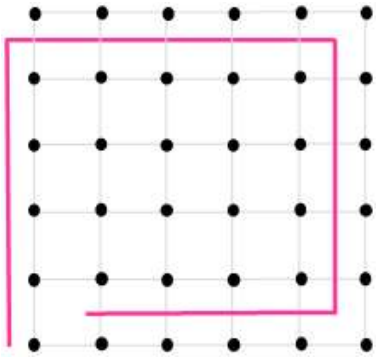

[격자형 평면 그래프]

이러한 평면 그래프는 수직으로 중첩함으로써 평면에서 공간으로 확장할 수 있는데, 아래 그림과 같이 공간에서 평면그래프로 이해될 수 있다.


[공간에서의 평면 그래프]

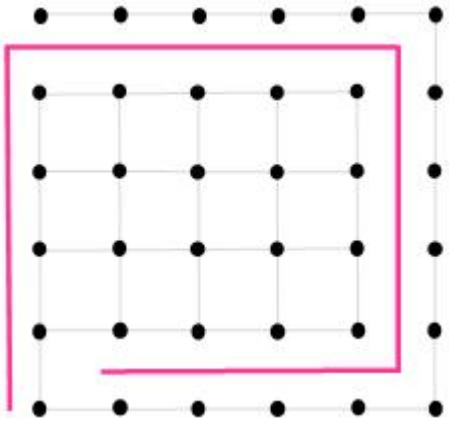
실제 공간에서는 벽체가 존재하며 아래 그림과 같이 벽체가 존재하면 그 벽체가 공간의

연결을 끊게 된다.



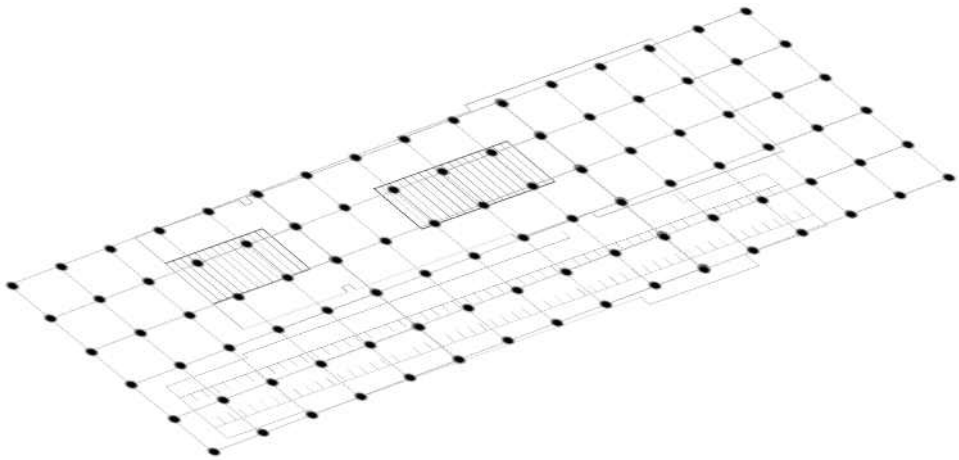
[벽체가 공간의 연결을 끊는 예시]

벽체가 지나는 부분의 간선을 삭제함으로써 공간 그래프도 원 공간과 같은 연결 구조를 가지게 된다.



[공간을 그래프로 변환한 결과 그래프]

이러한 과정을 일반적인 공간 도면에 적용하면 아래와 같은 결과가 얻어진다.



[공간 도면과 그래프 데이터의 매칭]

4) Minimum Node Weight Spanning Tree Algorithm

가) 개요

제안 알고리즘은 그래프 구조의 각 노드에서 시작점(sink)로 이동하기 위한 최소거리 및 경로를 노드의 웨이트를 이용해서 산출할 수 있도록 한 node-weighted spanning tree이다.

spanning tree(신장 트리)의 일종인데 신장트리는 노드 V 에 대해서 에지(E)의 개수가 |V| - 1인 트리이다.

너비우선탐색(BFS), 깊이우선탐색(DFS)와 유사하게 각 노드를 탐색하면서 트리를 구현했는데 기본적으로 BFS의 알고리즘을 차용하며, 중복적으로 각 노드의 웨이트를 비교해서 최소 웨이트를 산출하는 방식이다.

입력 데이터 형식은 인접 리스트 형태이다.

나) 타 알고리즘과 비교

MST와의 차이 : 도로의 관점에서 본다면 기존 도시 도로망에서 MST(minimum spanning tree)는 각 수요처를 연결하는 최소도로길이를 구하는 관점에서 볼 수 있고, MNWST는 각 지점에서 특정 지역으로 최단거리이동하기 위한 주요도로를 지정하는 것으로 이해를 하면 된다.

BFS와의 차이점 : DFS는 특정 노드를 큐에 넣은 후에 다시 확인을 하지 않으나 본 알고리즘은 최소 가중치를 구해야하기 때문에 반복적으로 계산을 수행한다.

다) 알고리즘

제안 알고리즘은 다음과 같다.

```
MNWST(G,s)
for each v in V
Weight[v] <- |V| + 1;
Weight[s] <- 0;
# 각 노드별로 가중치를 구하고, 부모 노드를 구함
if all Weight < |V| + 1 #모든 노드가 가중치를 부여 받을 때까지
for each (v1, v2) in E   # [1]
if weight[v1] is not equal to |V|+1
tmp <- Weight[v1] + 1;
if Weight[v2] > tmp
    Weight[v2] <- tmp
    Parent[v2] <- v1   # v2의 부모 노드로 v1을 설정
    for each v in Parent
        E_mnwst <- (Parent(v), v) # MNWST 를 구함
```

G : 그래프, V : 노드 집합 , E : 간선 집합, E_mnwst : MNWST 의 간선집합

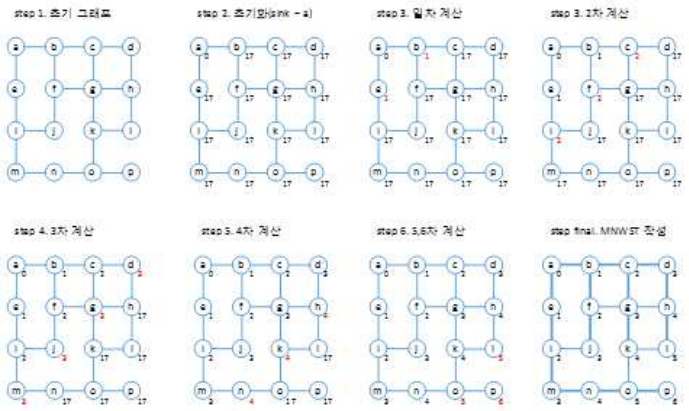
s : 시작점, sink, 대피소

본 알고리즘은 모든 노드가 연결되어 있는 그래프(connected graph)를 상정하고 있으며, 만약 연결되어 있지 않은 그래프인 경우, 코드에서는 [1]에서 인덱스를 두어서 한번 루틴이 돌때 변경사항이 있는지를 확인하는 프로세스가 포함되어야 함

connected graph에서는 프로세스가 종료시 모든 노드가 |V| +1 보다 작은 값을 부여받게 됨

각 노드의 자식 노드 수는 Parent[]에서 각 노드별로 구할 수 있음

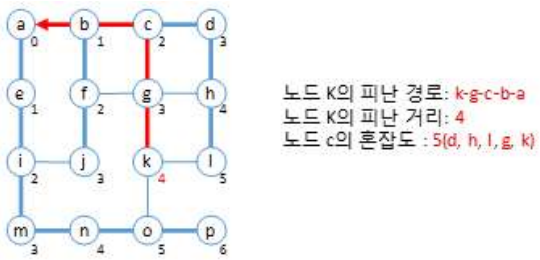
라) 알고리즘 도식 해설



[MNWST]

step1은 공간의 그래프화에 의해서 생성된 초기 그래프이다.
step2는 노드 a를 탈출처(sink)로 하고 나머지 노드는 전체 노드 수+1의 값을 부여한다.
step3에서는 노드 a의 인접 노드에 대해서 노드 a의 값인 0에 1을 플러스한 값을 부여한다.
step6까지 값을 부여받지 못한 즉 노드값이 17인 노드에 대해서 인접 노드의 가장 낮은 수에 대해서 플러스 1을 한 값을 새로운 노드로 부여하는 프로세스를 반복적으로 수행한다.
step final에서는 자신의 값을 결정한 인접노드를 부모노드로 하는 에지만을 남겨두면 MNWST가 작성이 된다.

마) MNWST의 특징
모든 노드에서 최단 경로 검색이 가능하다. 각 노드의 값들은 MNWST에서 탈출처인 싱크(sink)로의 최단거리를 나타낸다.
또한 필요에 따라서 각 노드가 이동경로로 이용되는 빈도로서 노드의 혼잡도는 각 노드의 자식 노드 수로서 구할 수 있다.



[MNWST의 특징]

바) MNWST 공간 적용
그래프 데이터 생성

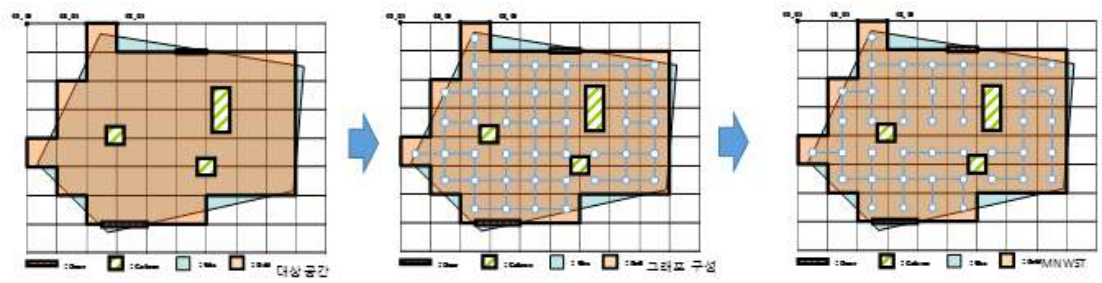
- 대상 공간을 그리드로 나누고 문이나 기둥에 의해서 단절되는 부분이외를 그래프로 작성
  단절이 되지 않은 노드끼리만 연결을 하고 대상 사이트 내부에 대해서 그래프를 생성한다.

대피처 설정

- 대피처를 하나 이상 설정 한 후에 알고리즘에 의해서 각 노드의 웨이트를 계산한다.

MNWST 계산

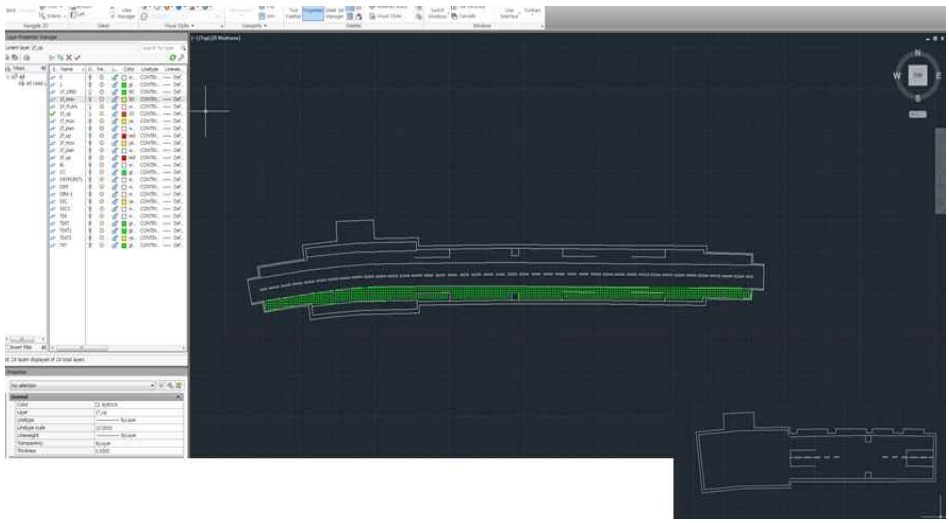
- 알고리즘에 의해서 각 노드의 가중치 및 스패닝 트리가 산출된다.



<그림 공간데이터의 MNWST 적용 예>

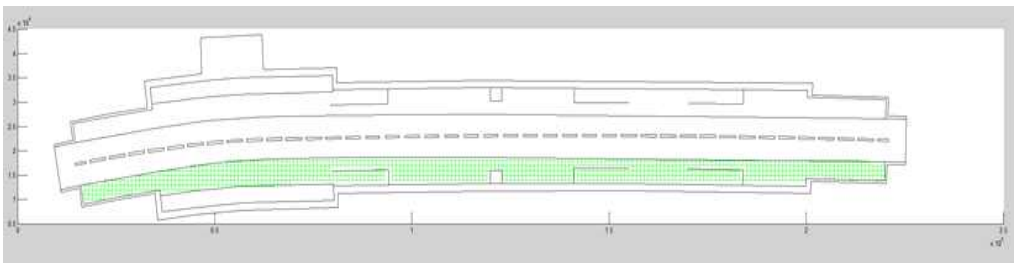
사) 개포동역 MNWST 적용

이 작업을 위해서 우선 한국철도공사에서 제공한 캐드도면에서 필요한 정보만을 추출하였다.



[한국철도공사의 캐드 파일]

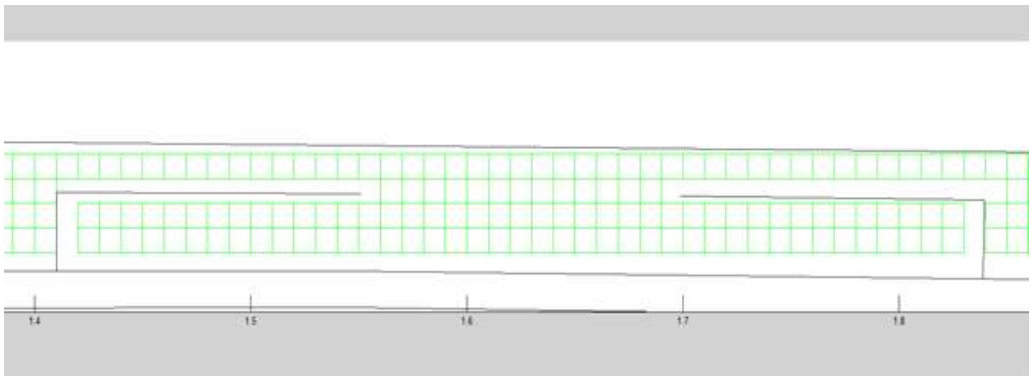
원 데이터의 많은 레이어 중에서 벽체 및 기둥 등의 바닥면을 결정하는 레이어와 계단부만을 추출한 후에 여기에 공간의 그래프화를 거쳐서 다음과 같은 공간 그래프를 작성하였다.



[공간 그래프]

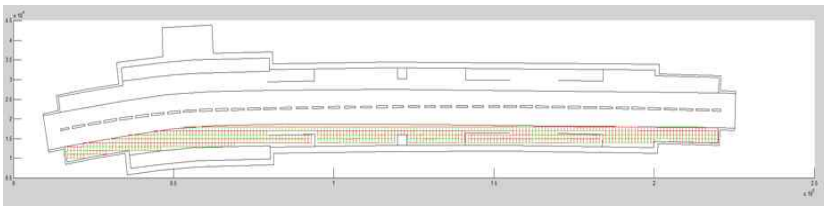
공간 그래프 중 계단부를 확대해보면 아래 그림과 같이 벽체가 있는 부분의 간선은 전부

삭제된 것을 알 수 있다.

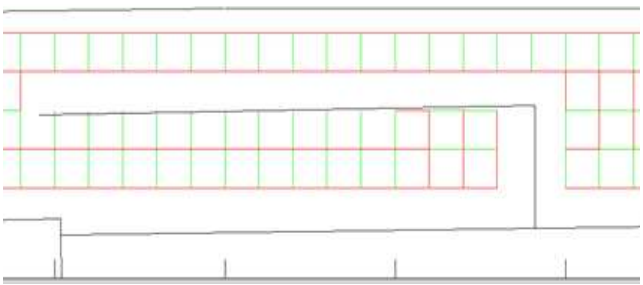


[공간 그래프 중 계단부 확대]

공간 그래프에 대해서 MNWST를 적용하여 다음의 결과를 얻었다.



[개포동역의 공간그래프(녹색) + MNWST(붉은색)]



[개포동역 공간그래프 및 MNWST 계단부 확대]

아) MNWST 알고리즘의 활용

방재

○ 공간을 그리드화 한 그래프 공간 구조에서 각 노드에서 대피처까지의 최단 경로를 그래프를 이용해서 도출해서 피난 경로 산출에 활용가능하다.

○ 모든 노드에 대한 피난 경로 계산을 통해서 각 노드의 사용 빈도를 구함으로써 피난 동선 밀도 계산에 필요하다.

○ 사용빈도는 에지의 사용회수를 구하거나 타임라인 별로 각 노드에서의 피난 에이전트 수를 구함으로서 간이 피난 시뮬레이션을 수행하는 것과 동일한 효과가 발생한다.

재난전조 예측시스템

○ 주어진 공간 내에서 대피처(sink)로 향하는 경로를 찾는 알고리즘이다.

○ 일반적으로는 최단경로 탐색을 이용하면 되나, 고정적인 평면(그래프 구조)에서 대피처(sink)가 정해진 상태이면 모든 출발지(노드)에서 대피처까지의 경로를 케이스바이케이스로 계산하는 것이 아니라, 모든 노드에서 최단경로를 위해서 이동가능한 다음 목적

지를 계산 한 그래프 즉 대피처까지의 거리를 속성으로 가지는 그래프를 작성한다.

○ 일정 시간대에 정체가 높아지면 다른 대안 선정 기능도 고려할 수 있다.

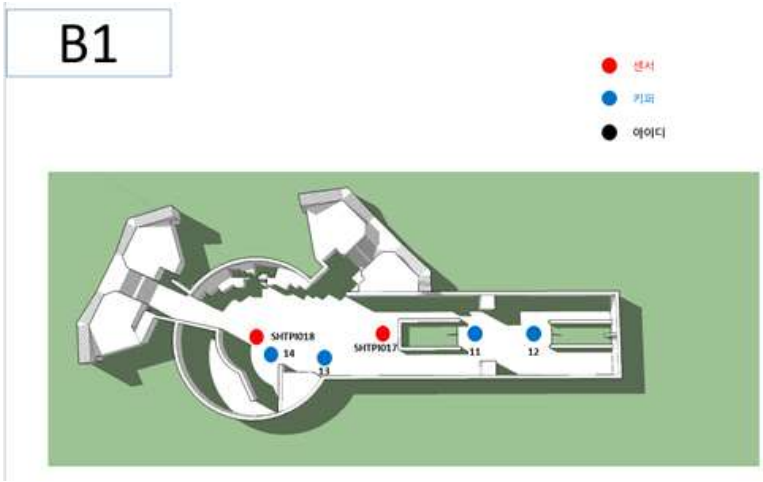
4) 대피유도 모듈 개발

가) 개요

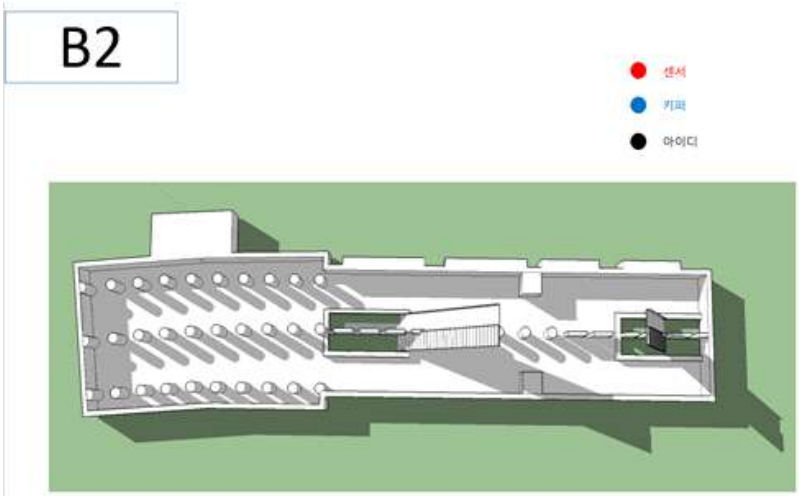
지하철 개포동역에는 키퍼 14개와 센서 디바이스 18개가 설치되었다.

지하철 개포동역의 피난유도 모듈은 이들 배치 현황정보를 바탕으로 각 구역 혹은 센서별로 대응 방법을 설정해서 피난을 수행하도록 한다.

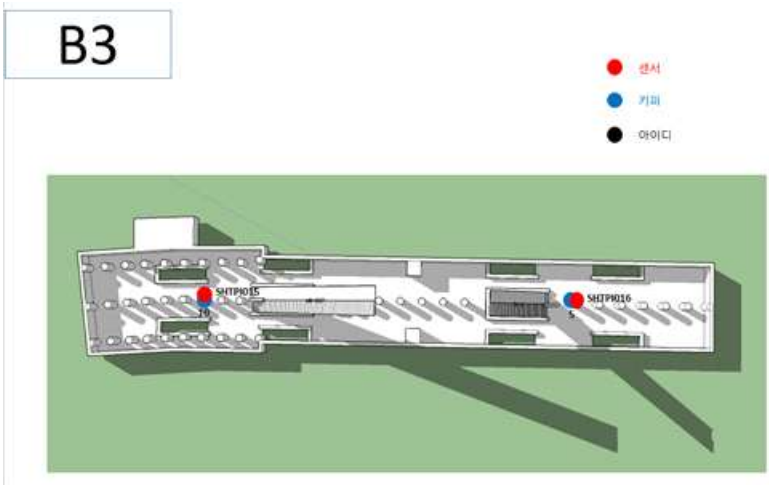
개포동역에서는 센서를 중심으로 피난 계획을 수립한다. 즉, 이상이 발생한 센서가 위치한 지점에 화재나 가스 유출 등의 재난이 발생한 것으로 간주해서 피난유도를 수행하도록 계획하였다.



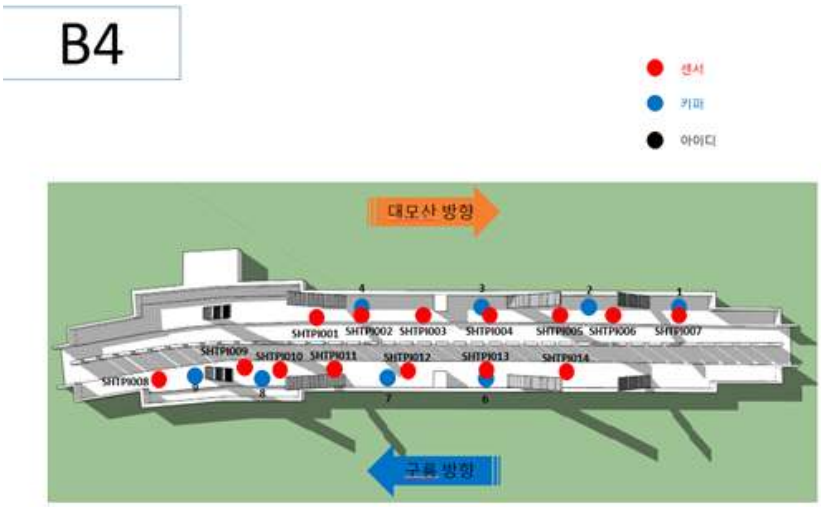
[개포동역사 지하 1층 센서 및 키퍼 배치 현황]



[개포동역사 지하 2층 센서 및 키퍼 배치 현황]



[개포동역사 지하 3층 센서 및 키퍼 배치 현황]



[개포동역사 플랫폼 센서 및 키퍼 배치 현황]

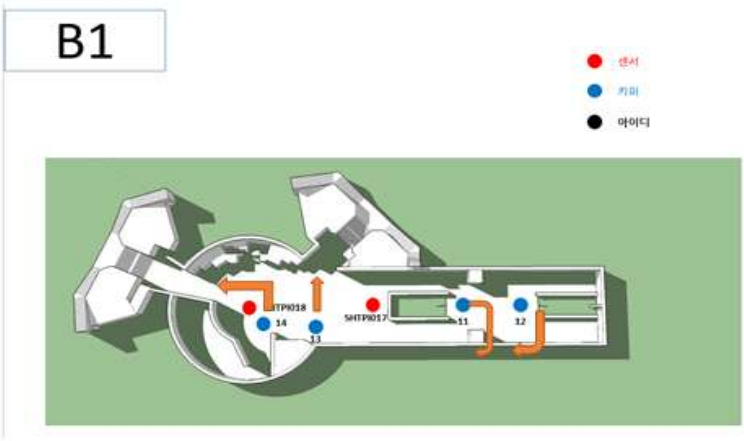
나) 피난유도 설정 방법

기본적으로 피난유도 SOP가 필요하며, 각 재난 별로 각 센서(혹은 존)에서 재난 발생 시 대피처가 어디가 되는지에 대한 결정이 필요하고 대피처가 결정이 되면 각 키퍼 별로 피난 유도 방향이 결정된다.

예를 들어서 지하 2층의 어떤 센서에서 어떤 재난 발생 시 지하 2층 이상은 대피처가 지하 1층의 몇 번 출구(혹은 복수)가 되고, 지하 4층은 터널로 이동하는 시나리오의 성립이 가능하다.

재난 종류 및 재난 발생 층수에 따라서 대피처가 달라질 수 있으며, 피난 유도 방향이 결정되면 각 키퍼 별로 피난 방향 지시 설정을 수행하게 된다.

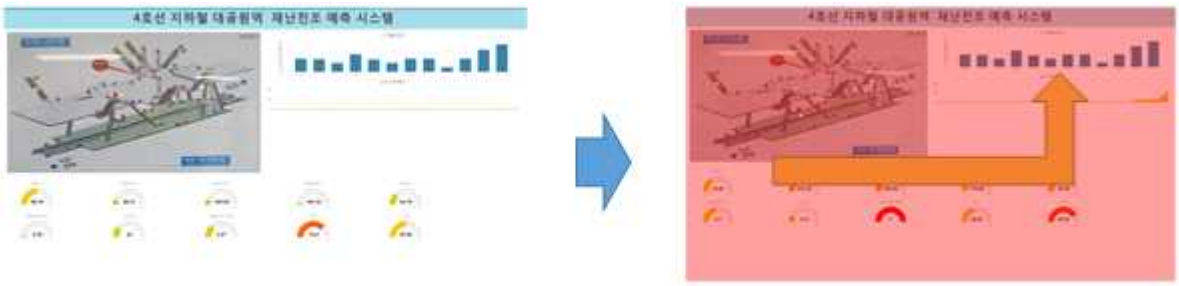
다음 그림에서는 지하 2층에서 화재 발생 시 키퍼 11, 12, 13, 14는 각 화살표 방향으로 피난방향 지시 화살표가 표시되도록 설정하는 예이다.



[피난 유도 방향 설정 예시]

다) 피난유도 SOP의 표출

피난유도 방향을 마스터에서 설정하고 저장을 하면 설정이 키퍼에 전파되어 동기화된다. 이 설정은 재난이 발생 시 키퍼의 화면에서 아래 그림과 같이 표출된다.



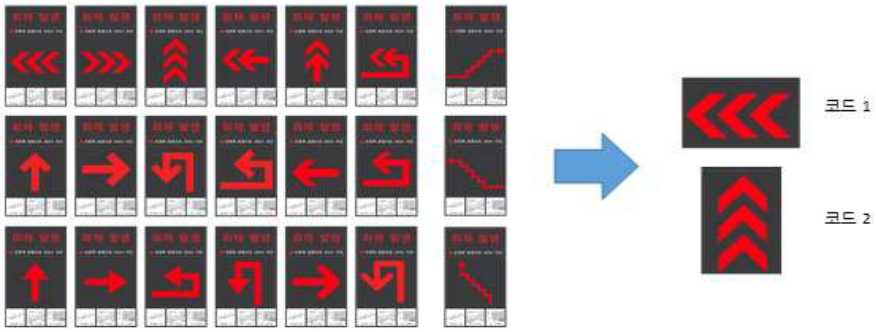
[키퍼에서의 피난 방향 지시 화살표 표시 화면]

여기서 중요한 것은 공간데이터에 피난유도 알고리즘을 적용한 결과의 피난 유도 방향과 키퍼에서 보여주는 방향에는 차이가 있다는 점이다.

피난 유도 알고리즘에서 나타내는 방향은 평면상에서의 방향이며, 키퍼에서 나타내는 방향은 키퍼가 설치된 입면에서 나타내는 방향이기 때문에 다를 수 있다.

이에 키퍼에서 나타내는 방향을 설정하기 위해서는 실제 키퍼가 설치된 장소의 입면에서 보는 방향이 되기 때문에 각 키퍼 별로 방향을 설치하기 위해서는 현장에서 키퍼가 설치된 위치를 확인한 후에 방향을 설정해야 한다.

화살표는 시스템에서 코드화되어서 각 시나리오별로 키퍼와 코드가 매칭되어서 저장된다.



시나리오	키퍼	코드
senaro1	keeper01	1
senaro1	keeper02	3
...	....	..
senaro18	keeper14	4

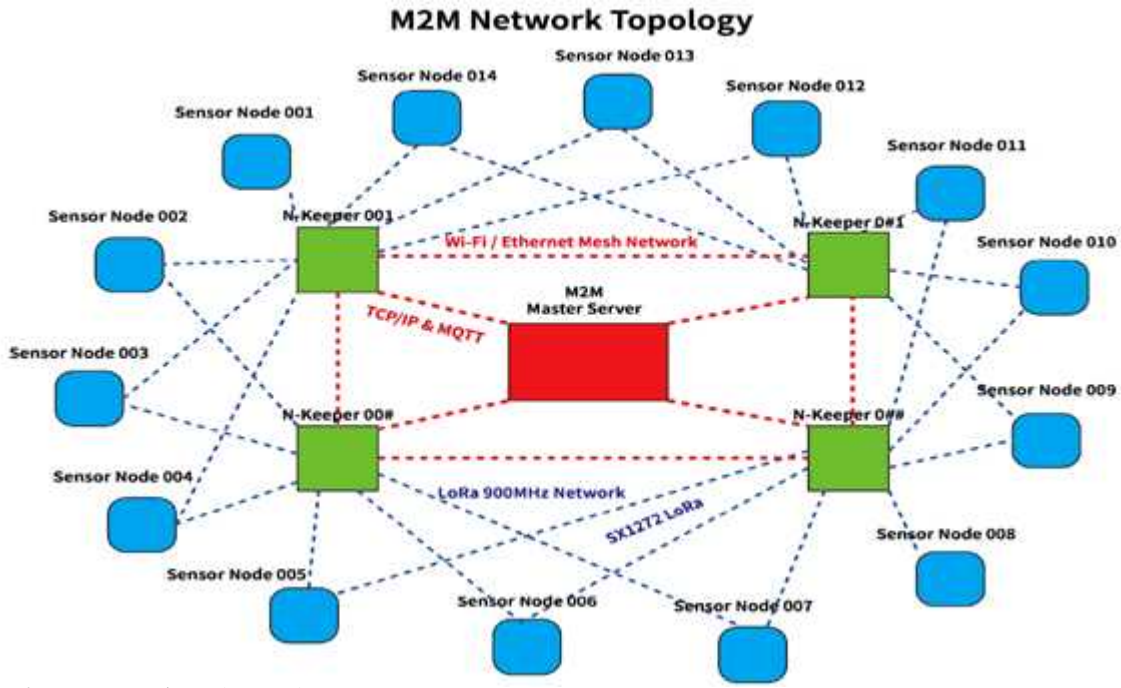
시나리오	재난종류	감지센서	대피처
senaro1	화재	Shtpi001	1
senaro2	가스유출	shtpi017	1,3

[피난유도 화살표와 코드화]

8. 재난전조 대응 M2M 네트워크의 구축

가. M2M 네트워크 구성

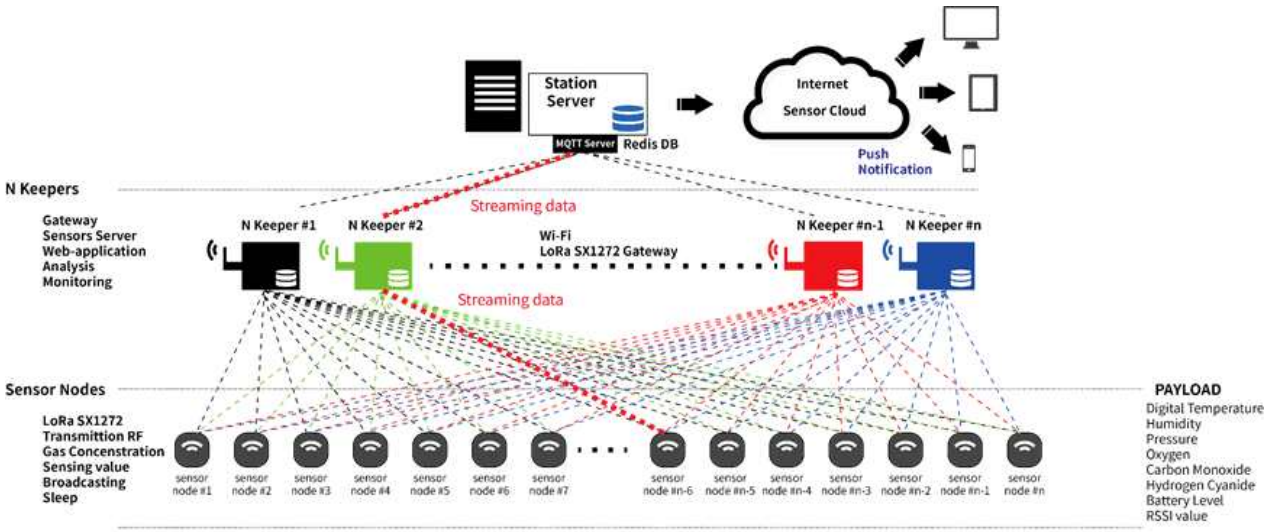
재난전조 대응 M2M 네트워크는 아래그림과 같다. 스마트N키퍼는 센서노드와 N키퍼 그리고 마스터 서버로 구성되어진다. 센서노드는 N키퍼와 900MHz 대역 LoRa SX1272 네트워크로 LoRa프로토콜로 데이터를 주고받는다. 이때 센서노드는 여러대의 N키퍼를 도착 주소로 갖게 되며 주변 N키퍼가 많기 때문에 센서노드의 네트워크 hop은 N키퍼가 MQTT 프로토콜을 사용하여 대신한다. 따라서 아래 그림과 같이 N키퍼의 통신에서 센서노드의 중복 데이터 수집하게 되고, 센서노드에서 데이터가 도달하지 않는 네트워크의 사각지대가 사라지게 된다. 마스터 서버는 센서노드의 RPC시각 데이터로 중복성 처리를 하는 알고리즘을 탑재한다.



나. M2M 네트워크 계층도

○ 재난 전조 대응 M2M 네트워크는 아래 그림과 같다. 센서노드와 스마트N키퍼 그리고 역무실의 마스터 서버로 구성되어진다. 각 레이어에서 통신하며 각 계층마다 역할은

다음과 같다.



○ 센서노드 레이어 : 센서노드는 주로 센싱작업과 데이터 송신작업을 한다. 각각의 가스센서의 concentration을 수행하며, 전력 절감을 위하 슬립모드를 수행한다. 페이로드로는 온도, 습도, 압력, 산소량, 일산화탄소량, 사인화수소, 이산화질소량, 배터리량, 그리고 통신상태를 송신한다. 페이로드는 데이터량을 절감하기 위하여 binary형태로 송신하며 프레임 데이터 구조는 다음과 같다.

HEADER							PAYLOAD								
<=>	0x00	0x17	0x74F94515	NODE_001	#	0x00	ID	Byte 1	Byte 2	ID	Byte 1	Byte 2	ID	Byte 1	Byte 2
A	B	C	E	F	D	G	Sensor 1			Sensor 2			Sensor 3		

A	Start Delimiter [3 Bytes]
B	Frame Type Byte [1Byte]
C	Number of Bytes [1Byte]
D	Separator [1Byte]
E	Serial ID [4Byte]
F	Node ID [variable]
G	Frame sequence [1Byte]

○ N키퍼 레이어 : 센서노드의 게이트웨이 역할을 하며 데이터 파싱과 동시에 raw 데이터를 바이너리 형태로 저장한다. 또한 파싱된 데이터를 각각의 N키퍼 DB에 저장하며 역무실에 있는 마스터 서버와 통신한다. 서버에서 각 N키퍼에 데이터 요구시 JSON형태로 데이터를 제공한다.

○ 마스터 서버(Station server) 레이어 : N키퍼에서 MQTT로 전송한 데이터를 마스터 서버의 DB에 저장한다. 최초의 데이터는 Redis에 저장되며, 누적된 콜드데이터는 mongodb로 이전한다. 마스터 서버는 M2M LTE 라우터를 통해 인터넷과 연결되며, 본사 사무실 서버에 구축된 센서 클라우드로 데이터를 송신할 예정이다. 이를 통해 역무실 담당자나 관리자는 마스터 서버를 통해 데이터를 모니터링 할 수 있으며, Push 알림이나 SMS등 다양한 서비스를 제공할 수 있다.

나. 재난 전조 대응 N키퍼 모듈 네트워크 구성

1) 개요

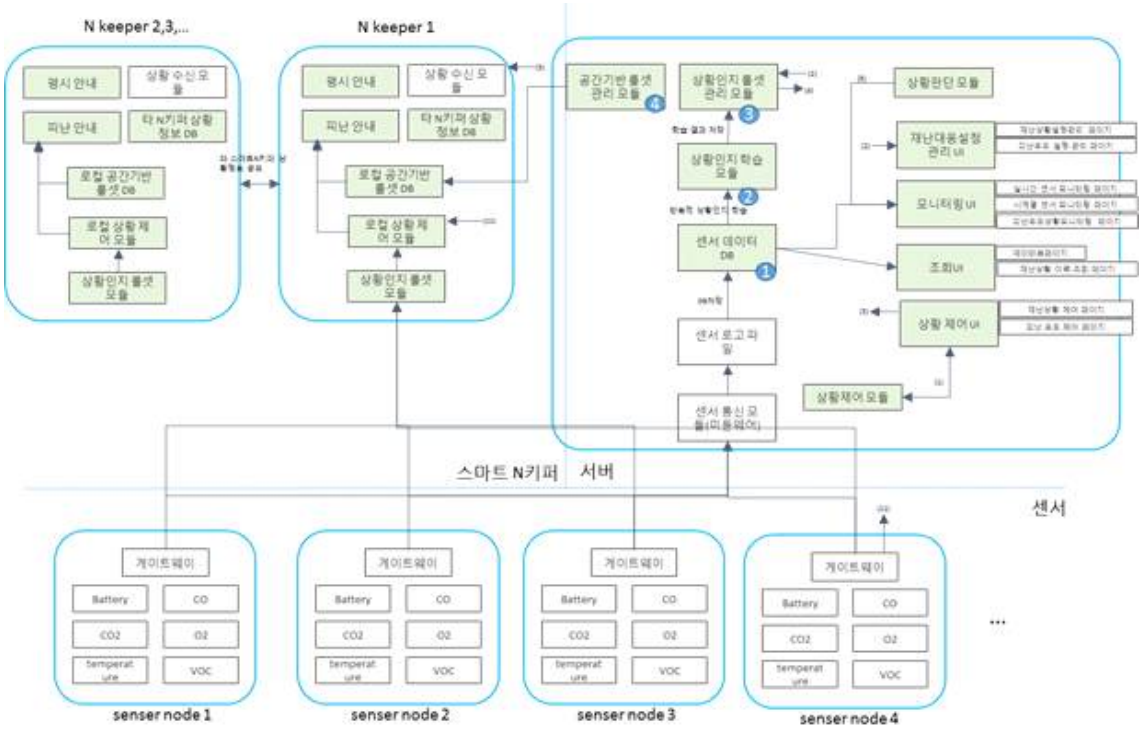
○ 시스템은 구성도에서 우상단은 마스터의 기능을 나타내며, 좌상단은 키퍼의 기능을 나타낸다.

○ 복수의 스마트N키퍼는 TCP/IP로 연결되고 센서노드는 LoRa네트워크로 연결된 센서 네트워크들이 각 키퍼 및 마스터의 센서 데이터 수용모듈을 통해서 각 머신에 저장이 된다.

○ 저장된 데이터는 JSON 형식으로 각 머신에서 제공이 된다.

○ 키퍼 및 마스터는 각 SW가 설치된 머신의 로컬호스트를 통해서 JSON형식으로 센싱 데이터를 제공받아 자체DB에 저장을 하며 사용을 한다.

○ 키퍼 및 마스터는 Ajax로 통신을 하며 필요한 정보를 교환한다.



2) 주요 서비스 내용

○ 전체 시스템은 역무실에 있는 관리 시스템인 마스터와 역사에 있는 키퍼로 이루어지며, 각 시스템의 서비스 내용은 다음과 같다.

가) 마스터

마스터는 센서 미들웨어에서 올라온 정보들을 취합하면서 실시간으로 모니터링을 하면서, 센서 데이터의 조건에 의해서 정상, 이상, 재난이라는3가지 상태를 실시간으로 상황인지 예측을 한다.

상황인지 예측 결과가 이상이나 재난을 나타내게 되면 시스템은 관리자에게 이 정보를 표출하고 관리자의 대응을 요청한다. 관리자는 일정한 시간 이내에 시스템의 요청에 따라서 상황을 인지한 후 정상, 이상, 재난의3가지 상태를 선택하게 된다.

정상으로 판단된 경우, 시스템은 정상모드로 돌아가며, 이전 상황을 로그에 기록하고, 이 상황을 룰셋으로 저장한 후 동일한 조건에서 상황을 발생시키지 않도록 한다.

이상이나 재난으로 관리자가 판단한 경우, 시스템은 이상 및 재난 상황발생을 수행하며 SOP에 의해서 이후 재난 대응 프로세스를 진행한다.

이상 및 재난 상황 종료 후 시스템은 해당 정보를 로그로 기록하여 추후 분석에 사용할 수 있도록 한다.

나) 키퍼

키퍼는 평상 시에는 센서 미들웨어의 정보를 실시간으로 모니터링하면서 상황인지를 수행하고, 이상 및 재난 발생 판단 시 서버와 연동하여 상황인지의 정확성을 검토한다.

서버에서 동일한 상황인지를 한 경우, 서버의 다음 대응을 기다리며, 서버에서 응답이 없는 경우(연결이 되지 않는 경우)에 일정 시간을 대기한 후에 자율적으로 상황전파 및 피

난유도를 수행한다.
피난유도 시 재난 발생 위치에 따른 사전 입력 시나리오에 따라서 승객을 정해진 경로로 유도한다
평상시에는 시스템 사용법 및 알람에 대한 대응방안 등을 안내한다.

3) 개발 환경 및 운용환경
가) 개발환경
○ 모든 개발은 윈도우 7 및 우분투 16.04의 환경에서 파이썬 3.5.3 및 장고 1.6.4의 환경에서 개발되었다.
○ 서비스는 웹으로 표출되며, 프론트엔드(웹페이지)는 우분투 16.04 환경의 파이어폭스을 대상으로 개발하였다.

나) 운용환경
운용환경은 파이썬 3.5.3과 장고 1.6.4가 돌아가는 환경이면 OS에 상관없이 돌아갈 수 있다.
일부 기능의 경우 우분투의 bash script를 사용하고 있기 때문에 가능한 우분투 환경에서 사용할 것을 추천한다.
현재 운용중인 우분투 OS 는 키퍼는 15.04버전, 마스터는 14.10버전이다.
웹브라우저로는 파이어폭스를 사용하였다.

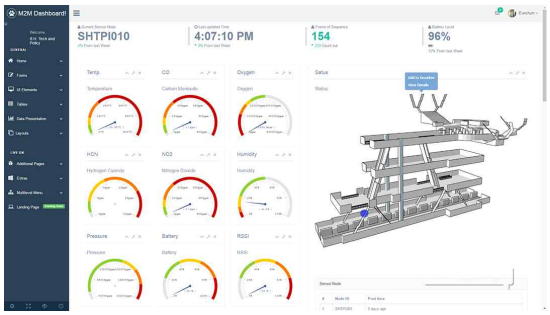
4) 용어집

본 시스템 및 보고서에서 사용하는 주요 용어는 다음과 같다.

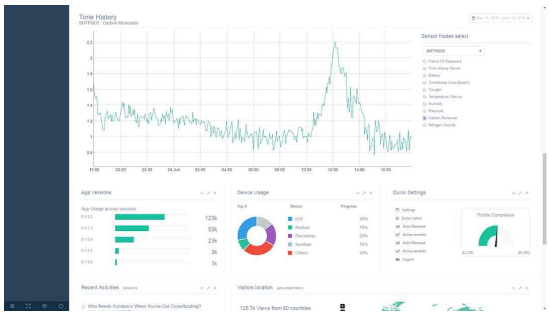
용어	정의
키퍼	제안서에서는 스마트N키퍼라고 정의하며, 줄여서 키퍼라고 함. 지하철 역사에 설치되는 단말기로서 센서 데이터를 수신하며, 센서 데이터에서 재난 여부를 판단하고, 상황전파를 하는 역할을 한다. 이를 위해서 디스플레이부도 포함하고 있으며, 네트워크에 연결되어 있으며, 네트워크 연결에 문제가 발생할 경우, 독자적으로 판단함
마스터	중앙의 관리서버를 지칭하며, 키퍼의 상태 점검 및 판단 모듈 업데이트 기능을 수행하며, 센서데이터를 수집, 분석 등을 수행하며, 재난 발생시에는 관제 제어의 역할을 하는 시스템
전파	상황이나 정보를 다른 사람이나 시스템에 전달
SOP	표준대응절차(Standard Operation Process)
상태정보	정상, 이상, 재난의 3가지 상태를 상태정보라 함
(재난)상태판단	센서 데이터와 재난상황조건을 바탕으로 현재의 상태가 정상, 이상, 재난 중 어떤 상태인지를 판단하는 행위
상태전환	시스템에서 상태정보를 한 상태에서 다른 상태로 변경하는 행위
존	재난 전조 시스템의 대상영역인 플랫폼 및 지하 공간을 일정한 구역 별로 나눈 단위 구역을 지칭함. 존 별로 하나이상의 키퍼가 설치되며 키퍼는 해당 존의 승객에 대해서 재난 발생 시 정보 표출 및 피난 유도 역할을 수행함
상황발생	시스템이나 관리자가 이상이나 재난이라고 상황을 판단하여 해당되는 SOP를 발생시키는 행위를 의미함
서비스지역	재난전조 시스템이 서비스를 대상으로 하는 전체 지역을 지칭
거리맵	서비스지역의 공간정보를 바탕으로 대상지역을 그래프구조로 변환해서 만든 맵으로 도로 네트워크와 유사한 네트워크 구조로 구성되며,
재난상황조건	이상과 재난의 상태정보로 상태전환을 하기 위한 조건으로 사용자가 센서 데이터의 값의 범위 및 위험도누적시간 등의 조건을 의미함
위험도누적시간	상황인지 룰셋에 의해서 이상 및 방재로 판별된 타임이 연속적으로 누적되는 시간. 위험도누적시간이 일정범위를 넘어서면 이상이나 재난으로 상태전환을 한다.
상황인지룰셋 민감도	새로 설정한 혹은 업데이트한 상황인지 룰셋이 수집된 과거 센서 데이터에 대해서 얼마나 자주, 오랜 시간동안 이상 혹은 재난이라고 상황인지를 했는지를 나타내는 조건
재난발령자	재난이라고 판단을 하고 시스템을 재난모드로 상태전환을 하는 액터를 지칭함, 재난발령자로는 관리자와 마스터 시스템과 키퍼가 있으며, 의사결정 우선순위는 관리자>마스터>키퍼 순이며, 상위 의사결정권자의 의사결정이 일정시간 내에 이루어지지 않으면, 권한이 다음 권한자로 넘어간다.
방재액티비티	마스터나 키퍼에서 재난 모드에서 방재를 위해서 수행한 모든 활동
센서한계값	사용자가 설정하는 센서의 경계값(threshold)로서 이 값을 넘어가면 이상 및 재난 상태로 판단하는 값이며, 지속성 여부에 따라서 이상과 재난을 판단한다
센서한계시간	센서한계값이 지속되는 시간

9. 웹기반 M2M 관리 시스템 개발

○ 재난 전조 대응 M2M 관리자 웹 어플리케이션은 아래 그림(a)와 같다. 최상단의 화면에서 현재 들어오는 데이터의 센서노드 번호와 측정된 시간 그리고 센서노드 프레임 카운트 그리고 배터리양을 보여준다. 왼쪽에는 센서노드에서 도달한 값 총 9가지의 센서값(온도, 일산화탄소량, 산소량, 시안화수소, 이산화질소, 습도, 대기압, 배터리양, 통신상태)를 게이지레벨로 출력한다. 오른쪽은 지하철역사에서 센서노드의 위치를 깜박이며 그 위치에서는 센서값을 한눈에 알 수 있도록 구성하였다. 총 18대의 센서노드의 값이 들어오면서 화면은 계속 변경되며, 현재 재난 상태와 센서 상태를 인식할 수 있다. 그림(b)와 그림(c)는 센서노드 별로 센서데이터를 선택하여 과거의 데이터 이력을 확인할 수 있다. 그림(b)는 단기 저장 그래프이며, 그림(c)는 장기 저장된 그래프이다. 데이터는 모두 고속처리를 위해 1~2일치 값을 인메모리(in-memory) 데이터베이스인 Redis에서 저장되고 표시한다. 그림(d)는 모든 센서노드의 데이터가 도착한 시간을 표시한다.



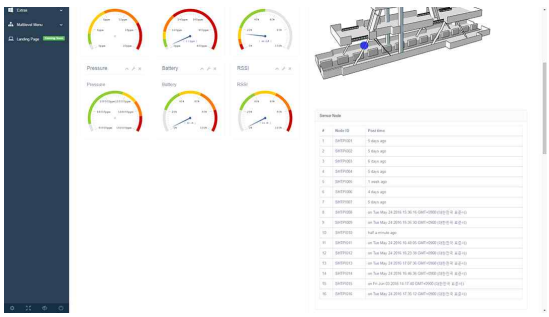
(a) 스트리밍 게이지 차트 / 위치 알림



(b) 센서노드별 단기 시간이력 조회



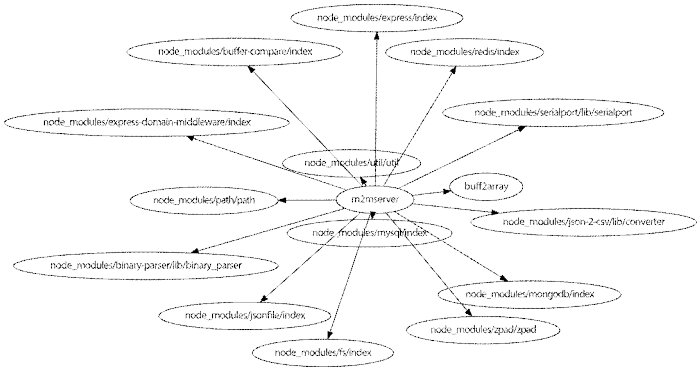
(c) 센서노드별 장기 시간이력 조회



(d) 센서노드 업데이트 상태

가. M2M 웹 어플리케이션 구성

1) 데이터 통신 / 센서노드 관리자 웹 어플리케이션 (m2mserver.js) 구성도



2) 패키지 구성

패키지 및 의존성 관리 파일	**package.json**

```
{
  "name": "m2mserver",
  "version": "1.0.0",
  "description": "M2M project administrator",
  "main": "m2mserver.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "sudo node m2mserver.js"
  },
  "repository": {
    "type": " git",
    "url": "https://github.com/haykin/m2m-server-main.git"
  },
  "keywords": [
    "shtpi",
    "m2mserver",
    "node"
  ],
  "author": "Eunchurn Park",
  "license": "ISC",
  "dependencies": {
    "binary-parser": "^1.1.5",
    "buffer": "^4.3.0",
    "buffer-compare": "^1.1.1",
    "coap": "^0.15.0",
    "debug": "^2.2.0",
    "dygraphs": "^1.1.1",
    "ejs": "^2.4.1",
    "ejs-locals": "^1.0.2",
    "express": "^3.21.2",
    "express-domain-middleware": "^0.1.0",
    "fs": "0.0.2",
    "grunt": "^1.0.1",
    "json-2-csv": "^2.0.22",
    "jsonfile": "^2.2.3",
    "mongodb": "^2.1.21",
    "mqtt": "^1.8.0",
    "mysql": "^2.10.2",
    "path": "^0.12.7",
    "redis": "^2.6.0-1",
    "serialport": "^2.1.2",
    "socket.io": "^1.4.5",
    "url": "^0.11.0",
    "util": "^0.10.3",
    "zpad": "^0.5.0"
  },
  "devDependencies": {
    "grunt": "^1.0.1",
    "grunt-contrib-concat": "^1.0.1",
    "grunt-contrib-jshint": "^1.0.0",
    "grunt-contrib-uglify": "^1.0.1"
  }
}
```

3) 데이터구조 및 클래스

Data Structure & Classes

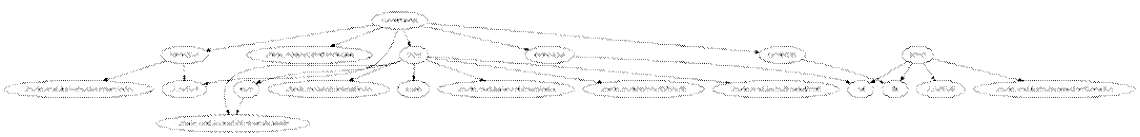
+SensorName.k : string = TimeStampUnix
+SensorTag.k : string = SENSORTST
+SensorUnit.k : string = EpochTime
+SensorValue.k : variable = TimeStamp
+TimeStamp : variable = undefined
+Value : variable = undefined
+buffersize : number = 128
+client : variable = undefined
+count : number = 0
+database : string = MeshliumDB
+file : string = serialconfigure
+host : string = 1921680251
+host : string = localhost
+i : number = 0
+i : variable = undefined
+ind : number = 20
+ind : variable
+k : number = 0
+latesttime : string
+latesttime : variable
+password : string = libelium2007
+password : string = qkrdmscjs
+queryscript : variable
+querystring : variable
+startindex.count : variable = i
+user : string = root
Parser : object
SensorName : Array
SensorTag : Array
SensorUnit : Array
SensorValue : Array
SerialPort : object
args : Array
dataset : object
listOfNode : Array
mongoClient : object
mongoServer : object
nowarray : Array
obj : object
output : Array
parsedJSON : Array
parseddata : Array
queryscript : Array
redisArgs : object
redisArgs.%5B%5D : object
sensorData : Array
sensorData.i.sensorName : object
sensorTimes : object
sensorUpdateInfo : Array
startindex : Array
status : object
temp_data : object
uptodate : Array

+DataStruct(serialID, waspmoteID, NOB, FOS, TST, name, tag, value, unit)
+FrameSequence(19, 20)
+FrameType(4, 5)
+LT()
+NumberOfBytes(5, 6)
+Separator(18, 19)
+SerialID()
+StartDelimiter()
+TimeStamp(0)
+Value(NULL)
+WaspmoteID()
+app()
+args(NULL)
+baudRate(baudRate)
+buf(,)
+client(NULL)
+comName(comName)
+csv(sensorData,)
+dataBits(dataBits)
+dateStart(dateStart)
+dateStop(dateStop)
+dbid(dbid)
+flowControl(flowControl)
+id_wasp(id_wasp)
+index(qdata, indicator)
+indicator()
+nowVal()
+nowarray(,)
+parity(parity)
+parsedJSON(result)
+parsedJSON(NULL)
+qarray(qdata)
+qdata(buffersize)
+redisArgs(NULL)
+redisClient()
+sensor(sensor)
+sensorData.i(NULL)
+sensorID(ind,)
+sensorName(sensorName)
+sensorNode(sensorNode)
+serialConf(file)
+serialPort(, , true,)
+server(,)
+serverTimeStamp()
+startDataId(nStart)
+stopBits(stopBits)
+stopDataId(nStop)
+timeStamp()
+uptodate(NULL)
+wstream()

4) 의존성 패키지 구성도

가) 통신 모듈 (serialport)

통신 모듈 'serialport'는 'node.js'에서 시리얼포트를 제어할 수 있는 모듈이다. OS는 Linux, MacOS, Windows 및 RaspberryPi등 다양한 환경에서 사용가능하며 모두 콜백함수 처리하여 비동기식 웹어플리케이션을 구성할 수 있다.



패키지 및 의존성 관리 파일	package.json

```
{
  "name": "serialport",
  "version": "2.1.2",
  "description": "Node.js package to access serial ports. Welcome your robotic javascript overlords. Better yet, program them!",
  "author": {
    "name": "Chris Williams",
    "email": "voodootikigod@gmail.com",
    "url": "http://www.voodootikigod.com"
  },
```

```
  "binary": {
    "module_name": "serialport",
    "module_path": "build/{configuration}/",
    "host": "https://github.com/voodootikigod/node-serialport/releases/download/2.1.2"
  },
  "main": "./lib/serialport",
  "repository": {
    "type": "git",
    "url": "git://github.com/voodootikigod/node-serialport.git"
  },
  "keywords": [
    "serialport",
    "nodebots",
    "johnny-five"
  ],
  "maintainers": [
    {
      "name": "jjrosent",
      "email": "jakerosenthal@gmail.com"
    },
    {
      "name": "reconbot",
      "email": "wizard@roborooter.com"
    },
    {
      "name": "voodootikigod",
      "email": "voodootikigod@gmail.com"
    }
  ],
  "dependencies": {
    "bindings": "1.2.1",
    "bluebird": "^3.3.5",
    "debug": "^2.1.1",
    "nan": "~2.2.1",
    "node-pre-gyp": "^0.6.26",
    "node-pre-gyp-github": "^1.1.0",
    "object.assign": "^4.0.3",
    "optimist": "~0.6.1",
    "sf": "0.1.7"
  },
  "devDependencies": {
    "chai": "^3.5.0",
    "chai-subset": "^1.2.2",
    "eslint-config-standard": "^5.1.0",
    "eslint-plugin-promise": "^1.1.0",
    "eslint-plugin-standard": "^1.3.2",
    "grunt": "0.4.5",
    "grunt-cli": "1.1.0",
    "grunt-mocha-test": "*",
    "gruntify-eslint": "^2.0.0",
    "mocha": "^2.4.5",
    "sandboxed-module": "^2.0.3",
    "sinon": "^1.17.3",
    "sinon-chai": "^2.8.0"
  },
  "engines": {
    "node": ">= 0.10.0"
  },
  "bin": {
    "serialportlist": "./bin/serialportList.js",
    "serialportterm": "./bin/serialportTerminal.js"
  },
  "license": "MIT",
  "scripts": {
    "install": "node-pre-gyp install --fallback-to-build",
    "rebuild": "npm rebuild && node-pre-gyp rebuild",
    "stress": "mocha --no-timeouts test/arduinoTest/stress.js",
    "test": "grunt --verbose"
  },
  "gypfile": true,
  "gitHead": "ac82f855e5a107ae8535fac9270236a9d06754ae",
  "bugs": {
    "url": "https://github.com/voodootikigod/node-serialport/issues"
  },
  "homepage": "https://github.com/voodootikigod/node-serialport#readme",
  "bundleDependencies": [
    "node-pre-gyp",
    "node-pre-gyp-github"
  ],
  "_id": "serialport@2.1.2",
  "_shasum": "493af176ac59043e7da5f2d7978fa30d1a8cb353",
  "_from": "serialport@>=2.1.2 <3.0.0",
  "_npmVersion": "3.8.6",
  "_nodeVersion": "5.11.0",
  "_npmUser": {
    "name": "reconbot",
    "email": "wizard@roborooter.com"
  },
  "dist": {
    "shasum": "493af176ac59043e7da5f2d7978fa30d1a8cb353",
    "tarball": "https://registry.npmjs.org/serialport/-/serialport-2.1.2.tgz"
  },
  "_npmOperationalInternal": {
    "host": "packages-12-west.internal.npmjs.com",
```

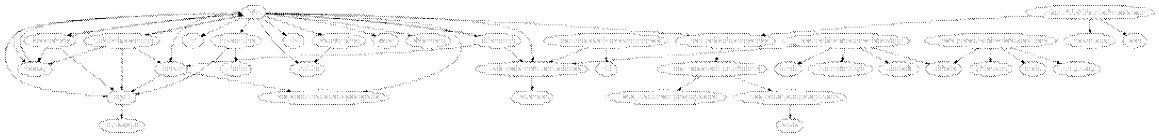
```
    "tmp": "tmp/serialport-2.1.2.tgz_1461557022672_0.7833052794449031"
  },
  "directories": {},
  "_resolved": "https://registry.npmjs.org/serialport/-/serialport-2.1.2.tgz",
  "readme": "ERROR: No README data found!"
}
```

나) Redis 모듈

레디스(Redis)는 오픈 소스이며 네트워크화, 인메모리 데이터베이스, 선택적 영구성 키-값 데이터 저장소이다. 레디스의 개발은 2013년 5월부터 Pivotal의 지원을 받고 있으며[1], 그 전에는 VMWare의 지원을 받았다.[2][3] DB-Engines.com의 월간 랭킹에 따르면, 레디스는 가장 인기 있는 키-값 저장소이다.[4]

성능은 memcached에 버금가면서 다양한 데이터 구조체를 지원함으로써 Message Queue, Shared memory, Remote directory 용도로 사용할 수 있으며, 이런 이유로 Instagram, Line 메신저 서비스, StackOverFlow등 여러 소셜 서비스에 널리 사용되고 있다. 본 과제에서는 실시간 데이터 표시와 응급신호의 처리 및 빠른 상황 처리를 위해 적용하였다.

본 과제에서는 아래와 같이 'node.js' 기반 'Redis' 클라이언트 모듈을 적용하였다.



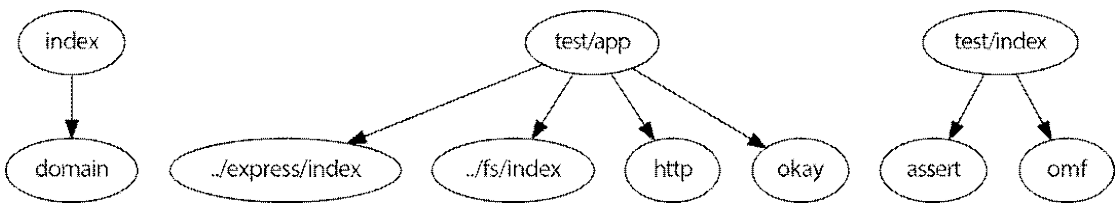
패키지 및 의존성 관리 파일	package.json

```
{
  "name": "redis",
  "version": "2.6.0-2",
  "description": "Redis client library",
  "keywords": [
    "database",
    "redis",
    "transaction",
    "pipelining",
    "performance",
    "queue",
    "nodejs",
    "pubsub",
    "backpressure"
  ],
  "author": {
    "name": "Matt Ranney",
    "email": "mjr@ranney.com"
  },
  "license": "MIT",
  "main": "./index.js",
  "scripts": {
    "coveralls": "nyc report --reporter=text-lcov | coveralls",
    "coverage": "nyc report --reporter=html",
    "benchmark": "node benchmarks/multi_bench.js",
    "test": "nyc --cache mocha ./test/*.js ./test/commands/*.js --timeout=8000",
    "posttest": "eslint . --fix"
  },
  "dependencies": {
    "double-ended-queue": "^2.1.0-0",
    "redis-commands": "^1.2.0",
    "redis-parser": "^1.3.0"
  },
  "engines": {
    "node": ">=0.10.0"
  },
  "devDependencies": {
    "bluebird": "^3.0.2",
    "coveralls": "^2.11.2",
    "intercept-stdout": "~0.1.2",
    "eslint": "^2.5.0",
    "metrics": "^0.1.9",
    "mocha": "^2.3.2",
    "nyc": "^6.0.0",
    "tcp-port-used": "^0.1.2",
    "uuid": "^2.0.1",
    "win-spawn": "^2.0.0"
  },
  "repository": {
```

```
    "type": "git",
    "url": "git://github.com/NodeRedis/node_redis.git"
  },
  "bugs": {
    "url": "https://github.com/NodeRedis/node_redis/issues"
  },
  "homepage": "https://github.com/NodeRedis/node_redis",
  "directories": {
    "example": "examples",
    "test": "test"
  },
  "gitHead": "a1755b91fb7ace72f79e3a42aeaf79374cd6023c",
  "_id": "redis@2.6.0-2",
  "_shasum": "808493eb168711bd36afdb1ac1ced22d19554779",
  "_from": "redis@>=2.6.0-1 <3.0.0",
  "_npmVersion": "2.15.1",
  "_nodeVersion": "4.4.3",
  "_npmUser": {
    "name": "bridgear",
    "email": "ruben@bridgewater.de"
  },
  "dist": {
    "shasum": "808493eb168711bd36afdb1ac1ced22d19554779",
    "tarball": "https://registry.npmjs.org/redis/-/redis-2.6.0-2.tgz"
  },
  "maintainers": [
    {
      "name": "bcoe",
      "email": "ben@npmjs.com"
    },
    {
      "name": "bridgear",
      "email": "ruben@bridgewater.de"
    },
    {
      "name": "bryce",
      "email": "bryce@ravenwall.com"
    },
    {
      "name": "mjr",
      "email": "mjr@ranney.com"
    }
  ],
  "_npmOperationalInternal": {
    "host": "packages-12-west.internal.npmjs.com",
    "tmp": "tmp/redis-2.6.0-2.tgz_1461897861582_0.2723832947667688"
  },
  "_resolved": "https://registry.npmjs.org/redis/-/redis-2.6.0-2.tgz",
  "readme": "ERROR: No README data found!"
}
```


다) Express 미들웨어 API

'express-domain-middleware'는 'express'웹서버 에러 핸들러 미들웨어이다. 시리얼포트 구성이나, 통신에러 및 DB접속에러가 발생할 경우 웹서버를 종료 시키지 않고 재가동 시킨다. 이는 비동기식 함수처리에 있어 시스템 입출력이나 이벤트 루프에서 에러가 발생할 경우 'throw'를 바로 사용할 수 없는 경우에 발생한다. 이때 도메인 API를 사용하여 에러를 처리할 수 있고, 도메인 에러 핸들러를 사용할 수 있다. 'express-domain-middleware'는 'node.js'의 웹서버 'express'에서 사용할 수 있는 대표적인 에러 핸들러이다.



패키지 및 의존성 관리 파일	package.json

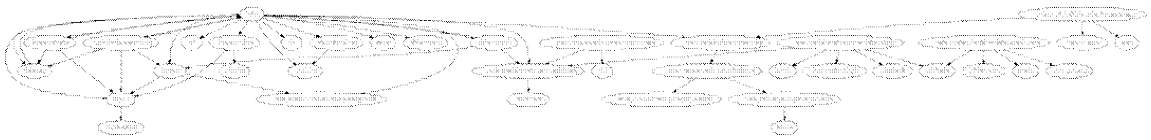

```
{
  "name": "express-domain-middleware",
  "version": "0.1.0",
  "description": "wrap express request/response with node domains",
  "main": "index.js",
```

```
  "directories": {
    "test": "test"
  },
  "scripts": {
    "test": "npm install express && mocha"
  },
  "repository": {
    "type": "git",
    "url": "git://github.com/brianc/node-domain-middleware.git"
  },
  "keywords": [
    "domain",
    "express",
    "middleware"
  ],
  "author": {
    "name": "Brian M. Carlson"
  },
  "license": "MIT",
  "devDependencies": {
    "omf": "~0.3.0",
    "okay": "~0.2.0",
    "mocha": "~1.8.2",
    "express": "~3.2.4"
  },
  "readme": "# express-domain-middleware₩n₩n[![Build Status](https://travis-ci.org/brianc/node-domain-middleware.png?branch=master)](https://travis-ci.org/brianc/node-domain-middleware)₩n₩nHi! Are you using `process.uncaughtException` in your express apps to keep them running?₩n₩nOR...Do you just let your process crash on any unhandled exception and restart it?₩n₩nDo you find it hard to pass ₩"request.id₩" to 8 nested database calls so you can keep a context of what request you're working on?₩n₩nHow do you associate log entries with a specific request?  Passing the request around everywhere again?₩n₩nDomains can help.₩n₩n## USE DOMAINS₩n₩nFirst, [read this](http://nodejs.org/api/domain.html)₩n₩nSecond, realize once you enable domains `process.domain` will give you the active domain.₩n₩nThird, use this middleware to bind each request/response pair to its own domain.₩n₩n```js₩nexpress.use(require('express-domain-middleware'));₩n```₩n₩n### express-domain-middleware api₩n₩n#### var domainMiddleware = require('express-domain-middleware');₩n₩n#### domainMiddleware = function(req, res, next) ₩n₩nExports a function matching the signature of express middleware.  Binds incoming request & response from express to a new domain.  Assigns the domain a unique id by calling `domainMiddleware.id(req)`.₩n₩nIf the domain emits an error event, `domainMiddleware` will call `next(err)` with the error event from the domain.  This allows existing express specific error handling middleware to₩nfunction as if the error was hanlded by your application code.  Allow me to demonstrate with an example:₩n₩n```js₩n///old-school₩napp.get('/error', function(req, res, next) {₩n db.query('SELECT happiness()', function(err, rows) {₩n if(err) return next(err); ₩n fs.readFile('alskdjflasd', function(err, contents) {₩n if(err) return next(err);₩n process.nextTick(function() {₩n throw new Error(₩"congratulations, your node process crashed and the user request disconnected in a jarring way₩");₩n });₩n });₩n })₩n});₩n```₩n₩n₩nnow with less crashing...₩n₩n₩n```js₩n//with domain-middleware₩napp.use(require('express-domain-middleware'));₩napp.use(app.router);₩napp.use(function errorHandler(err, req, res, next) {₩n  console.log('error on request %d %s %s: %j', process.domain.id, req.method, req.url, err);₩n  res.send(500, ₩"Something bad happened. :(₩");₩n if(err.domain) {₩n  //you should think about gracefully stopping & respawning your server₩n //since an unhandled error might put your application into an unknown state₩n }₩n});₩napp.get('/error', function(req, res, next) {₩n  db.query('SELECT happiness()', process.domain.intercept(function(rows) {₩n  fs.readFile('asldkfjasdf', process.domain.intercept(function(contents) {₩n process.nextTick(process.domain.intercept(function() {₩n  throw new Error(₩"The individual request will be passed to the express error handler, and your application will keep running.₩");₩n }));₩n }));₩n}));₩n});₩n```₩n₩n₩nI have to recommend using [okay](https://github.com/brianc/node-okay) to gracefully fallback in the absence of domains.  Plus..it's terse.  Go, code golf!₩n₩n```js₩nvar ok = require('okay');₩napp.use(require('express-domain-middleware'));₩napp.use(app.router);₩napp.use(function errorHandler(err, req, res, next) {₩n  console.log('error on request %d %s %s: %j', process.domain.id, req.method, req.url, err);₩n  res.send(500, ₩"Something bad happened. :(₩");₩n});₩napp.get('/error', function(req, res, next) {₩n  db.query('SELECT happiness()', ok(next, function(rows) {₩n  fs.readFile('asldkfjasdf', ok(next, function(contents) {₩n process.nextTick(ok(next, function() {₩n  throw new Error(₩"The individual request will be passed to the express error handler, and your application will keep running.₩");₩n  }));₩n  }));₩n }));₩n});₩n```₩n₩n## license₩nMIT₩n",
  "readmeFilename": "README.md",
  "_id": "express-domain-middleware@0.1.0",
  "dist": {
    "shasum": "36731b7c1901284fbf4fb5a62b0e7b0457d8e8c5",
    "tarball": "http://registry.npmjs.org/express-domain-middleware/-/express-domain-middleware-0.1.0.tgz"
  },
  "_from": "express-domain-middleware@>=0.1.0 <0.2.0",
  "_npmVersion": "1.2.17",
  "_npmUser": {
    "name": "brianc",
    "email": "brian.m.carlson@gmail.com"
  },
  "maintainers": [
    {
      "name": "brianc",
      "email": "brian.m.carlson@gmail.com"
    }
  ],
  "_shasum": "36731b7c1901284fbf4fb5a62b0e7b0457d8e8c5",
  "_resolved": "https://registry.npmjs.org/express-domain-middleware/-/express-domain-middleware-0.1.0.tgz",
  "bugs": {
    "url": "https://github.com/brianc/node-domain-middleware/issues"
```

```
  },
  "homepage": "https://github.com/brianc/node-domain-middleware#readme"
}
```

라) MQTT 서버/클라이언트

MQTT는 경량의 Publish/Subscribe(Pub/Sub) 메시징 프로토콜이다. M2M(machine-to-machine)와 IoT(Internet of things)에서의 사용하려고 만들었다. IoT를 위해서 낮은 전력, 낮은 대역폭 환경에서도 사용할 수 있도록 설계됐다. 본 과제에서는 N키퍼간의 통신을 위해 사용되며 각 N키퍼에서 받은 센서노드 데이터를 MQTT브로커를 통해 N키퍼간에 공유하며 마스터 서버에 저장하게 된다.



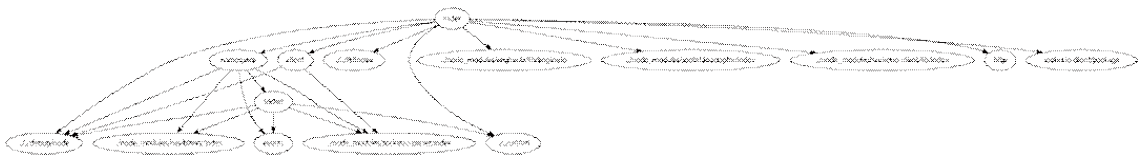
패키지 및 의존성 관리 파일	package.json

```
{
  "name": "mqtt",
  "description": "A library for the MQTT protocol",
  "version": "1.10.0",
  "contributors": [
    {
      "name": "Adam Rudd",
      "email": "adamvrr@gmail.com"
    },
    {
      "name": "Matteo Collina",
      "email": "matteo.collina@gmail.com",
      "url": "https://github.com/mcollina"
    }
  ],
  "keywords": [
    "mqtt",
    "publish/subscribe",
    "publish",
    "subscribe"
  ],
  "license": "MIT",
  "repository": {
    "type": "git",
    "url": "git://github.com/mqttjs/MQTT.js.git"
  },
  "main": "mqtt.js",
  "scripts": {
    "test": "mocha",
    "pretest": "npm run check-style",
    "check-style:eslint": "eslint mqtt.js lib test test/browser test/helpers",
    "check-style:jshint": "jshint mqtt.js lib test test/browser test/helpers",
    "check-style:jscs": "jscs mqtt.js lib test test/browser test/helpers",
    "check-style": "npm run check-style:jshint && npm run check-style:jscs && npm run
check-style:eslint",
    "browser-test": "zuul --server test/browser/server.js --local --open test/browser/test.js",
    "sauce-test": "zuul --server test/browser/server.js --tunnel ngrok -- test/browser/test.js"
  },
  "pre-commit": [
    "test"
  ],
  "bin": {
    "mqtt_pub": "./bin/pub.js",
    "mqtt_sub": "./bin/sub.js",
    "mqtt": "./mqtt.js"
  },
  "engines": {
    "node": ">=0.8.6"
  },
  "browser": {
    "./mqtt.js": "./lib/connect/index.js",
    "fs": false,
    "tls": false,
    "net": false
  },
  "dependencies": {
    "commist": "^1.0.0",
    "concat-stream": "^1.4.7",
    "end-of-stream": "^1.1.0",
```

```
    "help-me": "^0.1.0",
    "inherits": "^2.0.1",
    "minimist": "^1.1.0",
    "mqtt-connection": "^2.0.0",
    "mqtt-packet": "^3.2.0",
    "pump": "^1.0.1",
    "readable-stream": "~1.0.2",
    "reinterval": "^1.0.1",
    "split2": "^2.0.1",
    "websocket-stream": "^3.0.1",
    "xtend": "^4.0.0"
  },
  "devDependencies": {
    "browserify": "^11.0.1",
    "eslint": "^1.3.1",
    "jscs": "^2.1.1",
    "jshint": "2.7.0",
    "mocha": "*",
    "pre-commit": "1.1.1",
    "should": "*",
    "sinon": "~1.10.0",
    "through2": "^0.6.3",
    "uglify": "^0.1.1",
    "ws": "^0.8.0",
    "zuul": "^3.4.0",
    "zuul-ngrok": "github:gnandretta/zuul-ngrok#upgrade-ngrok"
  },
  "gitHead": "a62351b440e693a73ab0b4f00aa79ec2c7ea2b6f",
  "bugs": {
    "url": "https://github.com/mqttjs/MQTT.js/issues"
  },
  "homepage": "https://github.com/mqttjs/MQTT.js#readme",
  "_id": "mqtt@1.10.0",
  "_shasum": "0e82ea9bb3cde6b385cae0c2c728a6921d338c7c",
  "_from": "mqtt@>=1.8.0 <2.0.0",
  "_npmVersion": "2.15.0",
  "_nodeVersion": "4.4.2",
  "_npmUser": {
    "name": "matteo.collina",
    "email": "hello@matteocollina.com"
  },
  "dist": {
    "shasum": "0e82ea9bb3cde6b385cae0c2c728a6921d338c7c",
    "tarball": "https://registry.npmjs.org/mqtt/-/mqtt-1.10.0.tgz"
  },
  "maintainers": [
    {
      "name": "adamvr",
      "email": "adam.rudd@uqconnect.edu.au"
    },
    {
      "name": "matteo.collina",
      "email": "hello@matteocollina.com"
    }
  ],
  "_npmOperationalInternal": {
    "host": "packages-16-east.internal.npmjs.com",
    "tmp": "tmp/mqtt-1.10.0.tgz_1461768142810_0.8647426271345466"
  },
  "directories": {},
  "_resolved": "https://registry.npmjs.org/mqtt/-/mqtt-1.10.0.tgz"
}
```

마) Socket.io 모듈

WebSocket은 웹 서버와 웹 브라우저가 지속적으로 연결된 TCP 라인을 통해 실시간으로 데이터를 주고 받을 수 있도록 하는 HTML5의 새로운 사양이다. 따라서 WebSocket을 이용하면 일반적인 TCP소켓과 같이 연결지향 양방향 전이중 통신이 가능하다. 이와 같은 특징으로 웹에서도 채팅이나 게임, 실시간 주식 차트와 같은 실시간이 요구되는 응용 프로그램의 개발을 한층 효과적으로 구현할 수 있게 되었다. 본 과제에서는 Socket.io를 node에서 구현하여 각 N키퍼간의 브라우저로 양방향 통신을 구현한다.



패키지 및 의존성 관리 파일	package.json

```
{
  "name": "socket.io",
  "version": "1.4.6",
```

```
  "description": "node.js realtime framework server",
  "keywords": [
    "realtime",
    "framework",
    "websocket",
    "tcp",
    "events",
    "socket",
    "io"
  ],
  "main": "./lib/index",
  "license": "MIT",
  "repository": {
    "type": "git",
    "url": "git://github.com/Automattic/socket.io.git"
  },
  "scripts": {
    "test": "mocha --reporter dot --slow 200ms --bail"
  },
  "dependencies": {
    "engine.io": "1.6.9",
    "socket.io-parser": "2.2.6",
    "socket.io-client": "1.4.6",
    "socket.io-adapter": "0.4.0",
    "has-binary": "0.1.7",
    "debug": "2.2.0"
  },
  "devDependencies": {
    "expect.js": "0.3.1",
    "istanbul": "0.4.1",
    "mocha": "2.3.4",
    "superagent": "1.6.1",
    "supertest": "1.1.0",
    "zuul-ngrok": "3.2.0"
  },
  "contributors": [
    {
      "name": "Guillermo Rauch",
      "email": "rauchg@gmail.com"
    },
    {
      "name": "Arnout Kazemier",
      "email": "info@3rd-eden.com"
    },
    {
      "name": "Vladimir Dronnikov",
      "email": "dronnikov@gmail.com"
    },
    {
      "name": "Einar Otto Stangvik",
      "email": "einaros@gmail.com"
    }
  ],
  "gitHead": "dcca01f5a42b49e6fe68abefaabe5291e834c5d0",
  "bugs": {
    "url": "https://github.com/Automattic/socket.io/issues"
  },
  "homepage": "https://github.com/Automattic/socket.io#readme",
  "_id": "socket.io@1.4.6",
  "_shasum": "0dddc2cb8fb9b66fc928604f13f6aa910254cc1c",
  "_from": "socket.io@>=1.4.5 <2.0.0",
  "_npmVersion": "2.15.0",
  "_nodeVersion": "4.4.2",
  "_npmUser": {
    "name": "rauchg",
    "email": "rauchg@gmail.com"
  },
  "dist": {
    "shasum": "0dddc2cb8fb9b66fc928604f13f6aa910254cc1c",
    "tarball": "https://registry.npmjs.org/socket.io/-/socket.io-1.4.6.tgz"
  },
  "maintainers": [
    {
      "name": "rauchg",
      "email": "rauchg@gmail.com"
    }
  ],
  "_npmOperationalInternal": {
    "host": "packages-16-east.internal.npmjs.com",
    "tmp": "tmp/socket.io-1.4.6.tgz_1462247882801_0.6891660823021084"
  },
  "directories": {},
  "_resolved": "https://registry.npmjs.org/socket.io/-/socket.io-1.4.6.tgz"
}
```

10. 센서노드 개발

가. 센서보드 구성

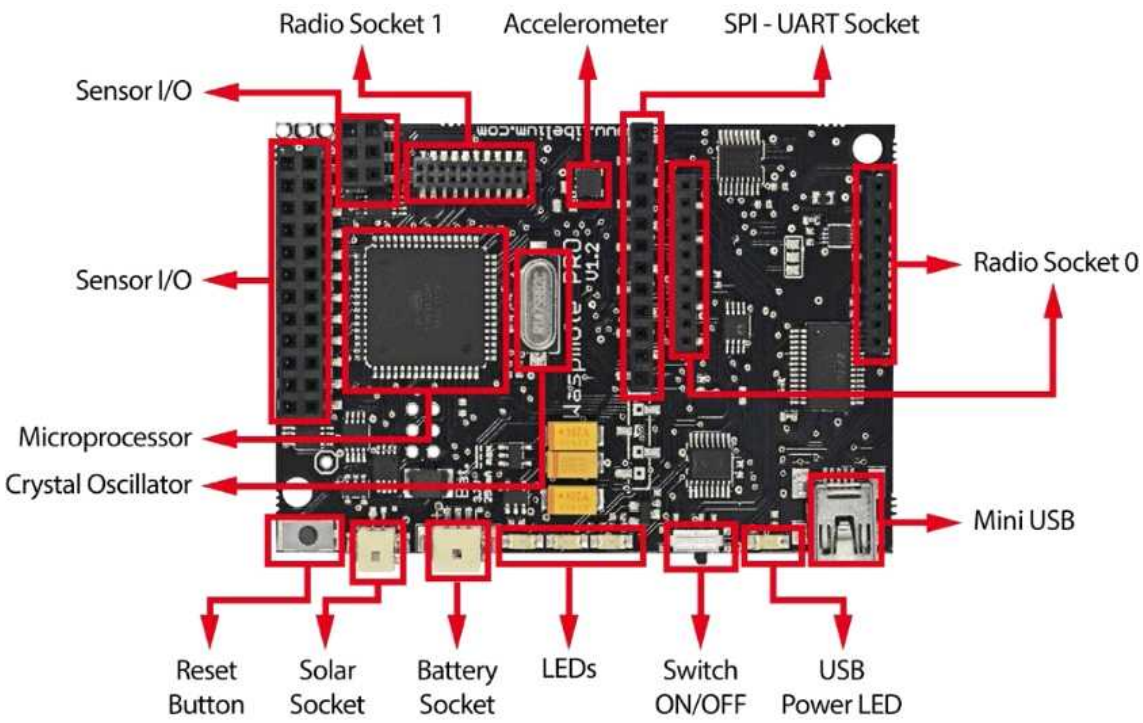
1) Waspmote

Libelium사의 Waspmote는 Aduino의 reference board에 센서들을 처리하기 위한 산업용 특화 제품으로 Mainboard에 다양한 종류의 센서 및 센서보드를 설치하여 다양한 분야에 사용이 가능하며 통신모듈의 범용성이 높아 필요한 통신거리에 따라 통신모듈의 커스터마이징이 가능한 장점을 갖고 있다.

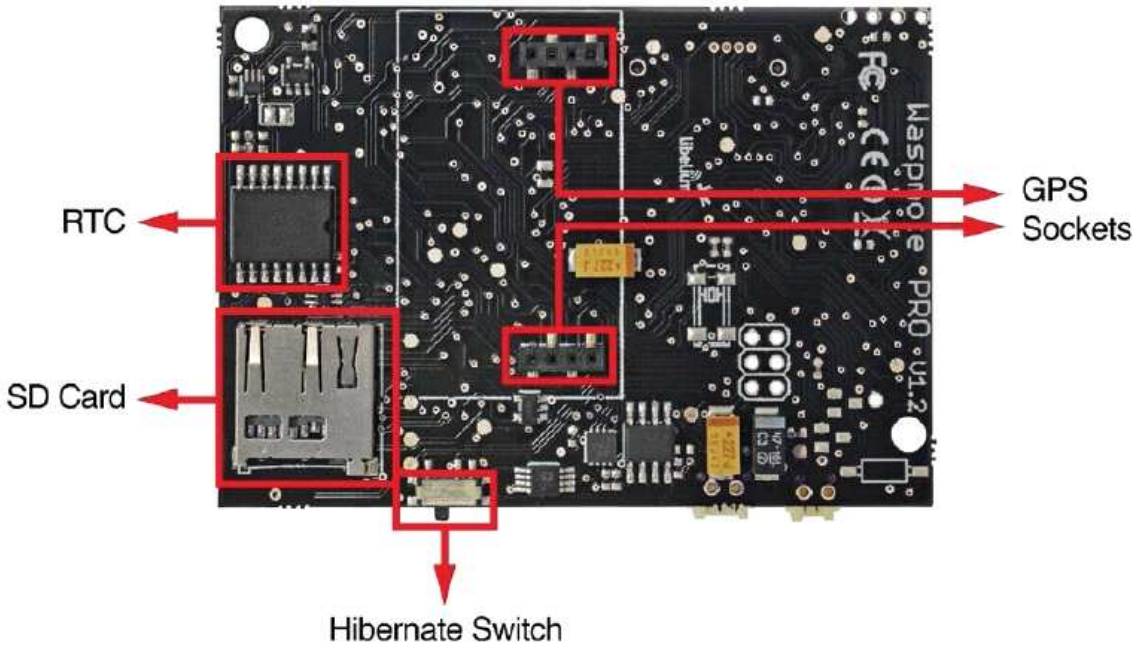
Waspmote의 자세한 Spec은 아래와 같다.

WSN Node Mainboard

- Microcontroller : ATmega1281 14.7456MHz
- Weight : 20g
- Dimensions : 73.5 x 51.0 x 13.0mm
- Consumption : 15mA(On), 55µA(Sleep), 0.07µA(Hibernate)
- Battery : 3.3V ~ 4.2V
- Storage : 128KB(Flash), 2GB(SD Card)
- Certification : CE(Europe), FCC(USA), IC(Canada)



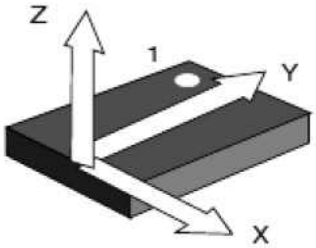
[Mainboard 상부]



[Mainboard 하부]

또한 진동과 가속도를 측정할 수 있는 3축 가속도계가 내장되어 있으며 Spec은 다음과 같다.

* WSN 노드용 메인보드에 내장
- Measurement range : ±2g, ±4g, ±8g
- Low power sampling : 0.5Hz/1Hz/2Hz/5Hz/10Hz
- Normal power sampling : 50Hz/100Hz/400Hz/1000Hz



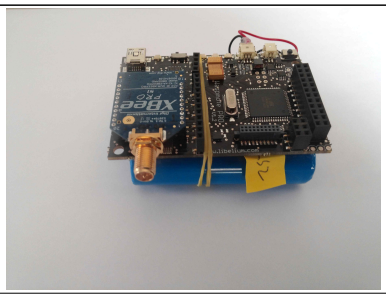
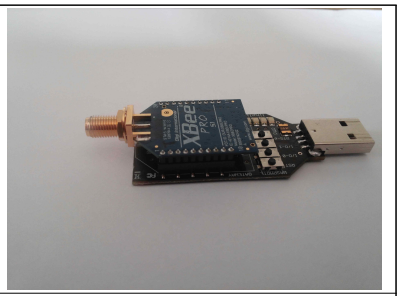
2) Digimesh PRO

XBee-PRO Digimesh Module은 전 세계적으로 효율적인 활용을 할 수 있는 2.4GHz Transceiver와 함께 쉽게 Mesh 네트워크를 구성 할 수 있다. XBee-Pro 모듈은 Sleeping 라우터와 복잡한 네트워크 운영과 같은 진보된 네트워크 특징을 가지는 반면, Mesh네트워크를 간소화하기 위해 peer-to-peer(P2P) 구조를 가진다. 이러한 기능은 배터리의 수명 기간을 늘여주며 네트워크의 안전성을 제공한다.

XBee-PRO모듈의 주요특징 및 사용 Spec을 정리한 표는 다음과 같다.

[XBee-PRO 통신모듈의 주요특징]

(1) Price-to-Performance Value	(2) XBee 모델 디자인에서 각인된 혁신으로 인해 표준ZigBee모듈의 범위보다 2~3배 더 길다. 또한, XBee모듈은 사용하기 쉬우며, 결론적으로 Data 시스템 개발의 비용을 상당히 줄여준다.
(3) Low Power Consumption	(4) XBee 모듈은 적은 파워를 요구한다.(XBee에서는45mA Tx 전류, XBee-PRO에서는 270mA Tx 전류)게다가, Sleep모드는 10uA 이하에 달하는 전력강하를 지원한다.
(5) Receiver Sensitivity	(6) Wireless 연결에 있어서 상당한 통신범위와 신뢰성을 지원한다. TX 파워나 RX 감도에서 얻은 6dB에 대해 OEMs & Integrators는 Wireless 연결범위를 두배로 할 수 있다. XBee모듈은 우수한 RX 감도를 통한 대규모적인 범위 이득으로 인해 고가 모듈 이상의 결과를 낸다.
(7) Worldwide Acceptance	(8) FCC(U.S.A), IC(Canada), ETSI(Europe)

		
XBee-PRO	Waspmote 장착 모습	USB Gateway 장착 모습

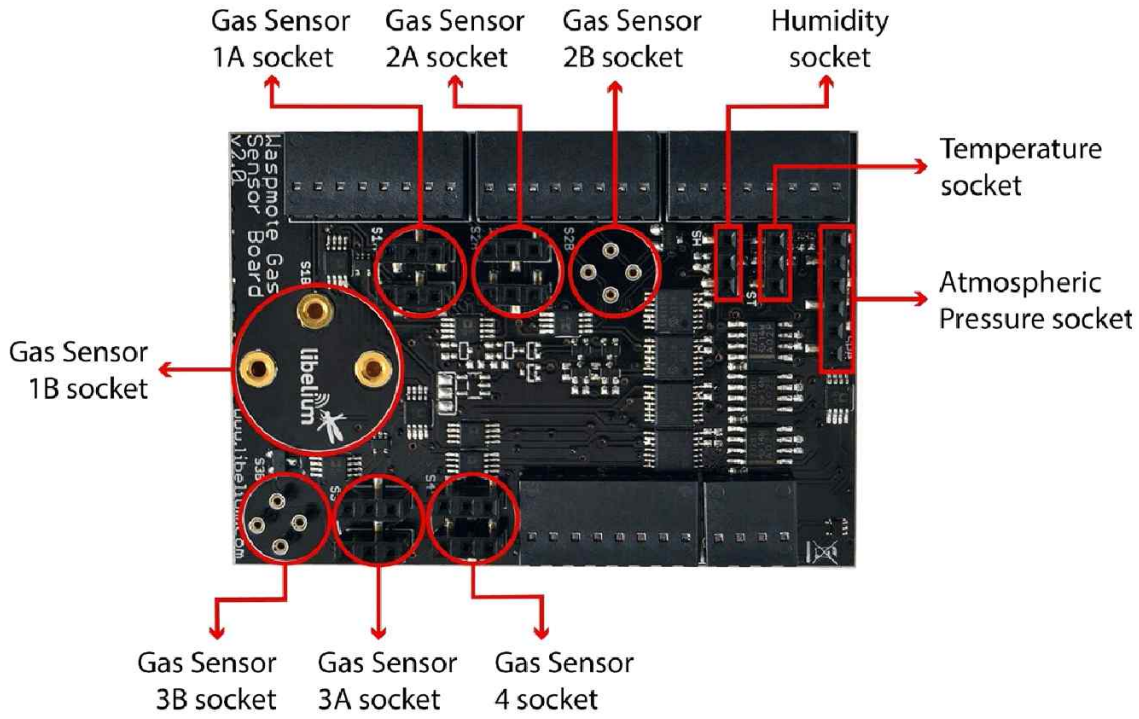
[XBee-PRO 통신모듈의 사용 Spec]

Specification		XBee-PRO
Performance	Indoor/Urban Range	up to 300ft.(100m)
	Outdoor RF line-of-sight Range	up to 1mile(1.6Km)
	Transmit Power Output	60 mW(18dBm), 100 mW EIRP
	RF Data Rate	250,000 bps
	Receiver Sensitivity	-100 dBm(1% PER)
Power Requirements	Supply Voltage	2.8-3.4 V
	Transmit Current(typical)	270 mA @ 3.3V
	Receive Current(typical)	55 mA @ 3.3V
	Power-down Current	<10 uA
General	Frequency	ISM 2.4 GHz
	Dimensions	0.960" X 1.297" (2.438cm X 3.294cm)
	Operating Temperature	-40 to 85℃(industrial)
	Antenna Options	U.FL Connector, Chip Antenna, Whip Antenna
Networking and Security	Supported Network Topologies	Point-to-Point, Point-to-Multipoint, Peer-to-Peer and Mesh
	Number of Channels	13 Direct Sequence Channels (software selectable)
	Filtration Options	Channel, PAN ID & Source/Destination Address
Agency Approvals	FCC Part 15.247	OUR-XBee
	Industry Cananda(IC)	4214A-XBee PRO
	Europe(CE)	ETSI(10 mW max Tx output)

3) 가스센서보드

본 연구에서 사용할 가스센서보드는 O2, CO, CO2, VOC, CH4, H2, O3등 14가지의 가스 및 대기상태를 탐지할 수 있는 센서보드이다. 또한 최대 7가지의 가스센서를 동시에 장착하여 감지할 수 있다.

가스센서보드가 탐지할 수 있는 가스의 종류 및 가스보드의 Hardware는 다음과 같다.



[가스센서보드]

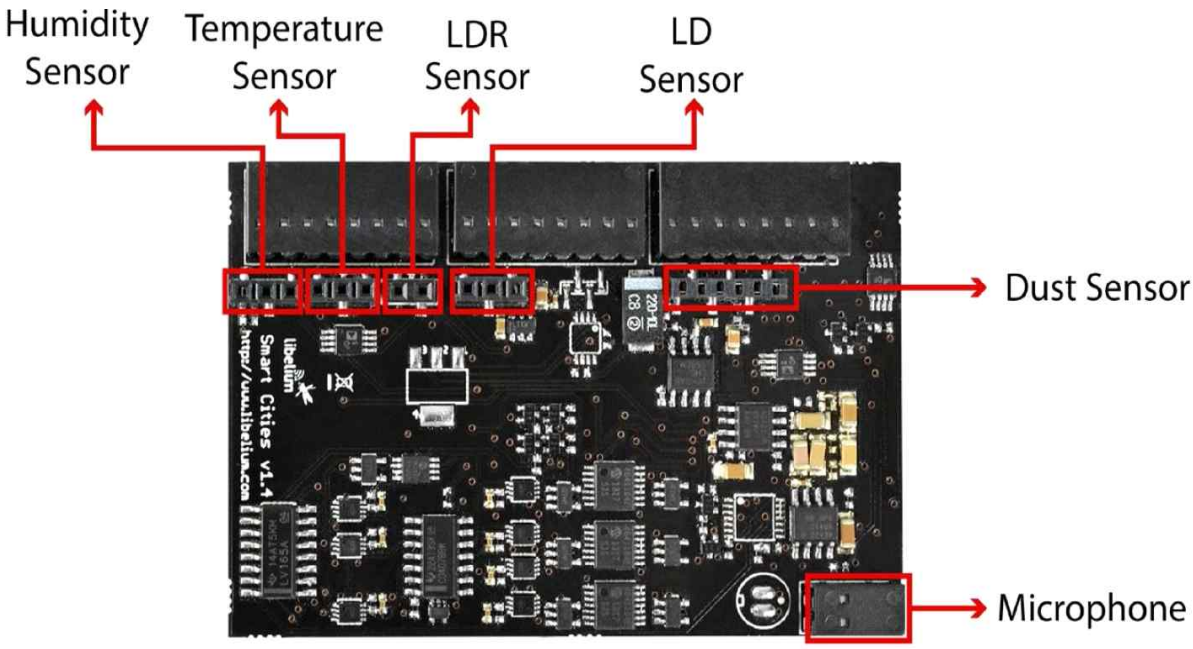
- Weight : 20g
- Dimensions : 73.5 x 51.0 x 13.0mm
- Board power : 3.3V, 5.0V (cont. 200mA, peak 400mA)
- Sensor power : 5.0V
- Gas sensors : Carbon Monoxide - CO
  Carbon Dioxide - CO2
  Molecular Oxygen - O2
  Methane - CH4
  Molecular Hydrogen - H2
  Ammonia - NH3
  Volatile Organic Compounds (VOC's)
  Hydrogen Sulphide - H2S
  Nitrogen Dioxide - NO2
  Ozone - O3
  Hydrocarbons
  Isobutane - C4H10
  Ethanol - CH3CH2OH

[가스센서보드의 Spec 및 Calibration 유무]

Parameter	Range	Calibration	Consumption
Temperature	-40 to 125℃	Not calibrated ±2℃ (range 0 ~ 70 ℃), ±4℃ (range -40 ~ 125 ℃)	Some ㎃ @ 3V3
Humidity	0 to 100% RH	Not calibrated <±4% RH (at 25℃, range 30~80% RH) <±6% RH (range 00~100% RH)	Some ㎃ @ 5V
Pressure	15 to 115 kPa	Not calibrated <±1.5%V (range 0~85℃)	7 ㎃ @ 5V
Carbon Monoxide CO	30 to 1000ppm	Not calibrated	3 ㎃ @ 5V
Carbon Dioxide CO_2	350 to 10000ppm	Not calibrated	50 ㎃ @ 5V
Molecular Oxygen O_2	0 to 30%	Not calibrated	0 ㎂
Ozone O_3	0.010 to 1 ppm	Not calibrated	34 ㎃
Nitric Dioxide NO_2	0.05 to 5 ppm	Not calibrated	26 ㎃
Ammonia NH_3	10 to 100	Not calibrated	12 ㎃ @5V
Methane and other combustible gases CH_4	500 to 10000 ppm	Not calibrated	61 ㎃ @5V
Molecular Hydrogen H_2	500 to 10000 ppm	Not calibrated	61 ㎃ @5V 46 ㎃ @5V
Hydrogen Sulfide H_2S	0.1 to 3 ppm	Not calibrated	61 ㎃ @5V
Isobutane C_4H_{10}	50 to 5000 ppm	Not calibrated	61 ㎃ @5V
Ethanol CH_3CH_2OH	1 to 30 ppm 50 to 5000 ppm	Not calibrated	61 ㎃ @5V
Volatile Organic Compounds (VOC)	30 to 400 ppm	Not calibrated	32 ㎃

4) 조도/소음 센서보드

방화나 누전, 폭발과 같은 재난 발생 시 신속한 초기 대응을 위한 조도/소음 센서보드의 자세한 Spec은 다음과 같다.



[조도/소음 센서보드]

- Weight : 20g
- Dimensions : 73.5 x 51.0 x 13.0mm
- Board power : 3.3V, 5.0V (cont. 200mA, peak 400mA)
- Sensor power : 5.0V
- Sensors : Microphone, LDR, etc.

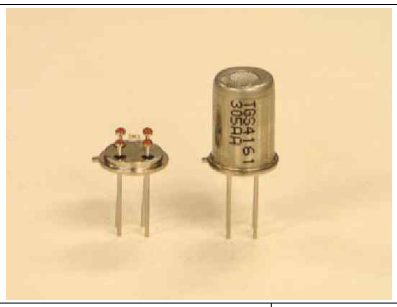
5) Sensor

□ CO2센서

일반사항(기준 범위)

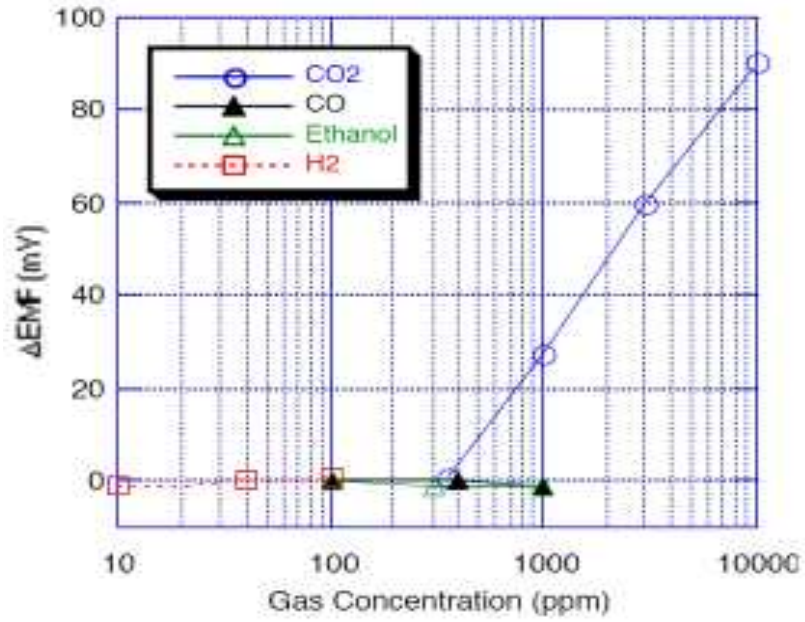
일반적으로 CO2의 실내환경기준치 범위는 1000ppm 이하 이며 CO2농도가 1000~2000ppm 일 경우 졸림을 느끼는 등 컨디션 변화가 나타날 수 있다. 또한, 공기중 CO2농도가 3000ppm 이상일 경우 두통, 현기증 등의 인체에 악영향을 끼칠 우려가 있다. 본 연구에서 사용한 CO2 센서는 Figaro社의 TGS 4161를 사용하였다. TGS 4161는 작은 크기와 저전력 소모, 긴 사용수명으로 경제적인 장점을 가지고 있다.

CO2센서의 자세한 Spec과 Hardware는 다음과 같다.

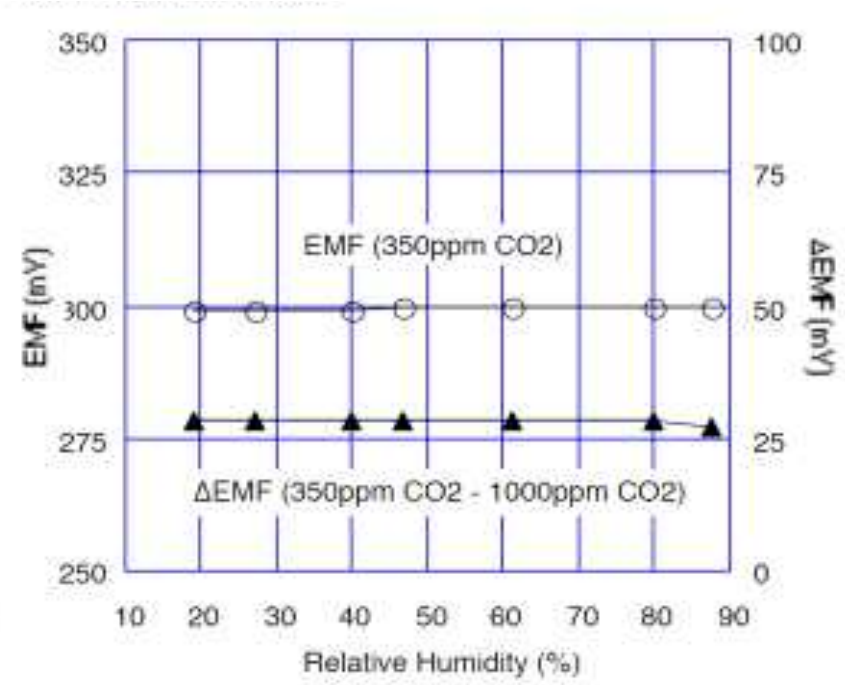
<table>
<tr><td colspan="5">  </td></tr>
<tr><td colspan="3">(1) Typical Detection range</td><td colspan="2">(2) 350~10,000 ppm</td></tr>
<tr><td rowspan="6">(3) Electronic characteristic</td><td>(4) Heater resistance</td><td>(5) RH</td><td colspan="2">(6) 70±7Ω at room temp.</td></tr>
<tr><td>(7) Heater current</td><td>(8) IH</td><td colspan="2">(9) approx. 50㎃</td></tr>
<tr><td>(10) Heater power (11) consumption</td><td>(12) PH</td><td colspan="2">(13) approx. 250㎃</td></tr>
<tr><td>(14) Electromotive force</td><td>(15) EMF</td><td colspan="2">(16) 220~490㎷ in 350ppm CO2</td></tr>
<tr><td>(17) Sensitivity</td><td>(18) △EMF</td><td>(19) 44~72㎷</td><td>(20) EMF(350ppm CO2)
(21) EMF(3500ppm CO2)</td></tr>
<tr><td>(22) Heater voltage</td><td>(23) VH</td><td colspan="2">(24) 5.0±0.2V (DC)</td></tr>
<tr><td rowspan="2">(25) Sensor characteristic</td><td colspan="2">(26) Response time</td><td colspan="2">(27) approx. 1.5 min
(28) (to 90% of final △EMF value)</td></tr>
<tr><td colspan="2">(29) Measurement accuracy</td><td colspan="2">(30) approx.±20% at 1,000ppm CO2</td></tr>
<tr><td colspan="3">(31) Operation conditions</td><td colspan="2">(32) -10~50℃, 5~95%RH</td></tr>
<tr><td colspan="3">(33) Storage conditions</td><td colspan="2">(34) -20~60℃, 5~90%RH</td></tr>
<tr><td rowspan="3">(35) Standard test conditions</td><td colspan="2">(36) Test gas condition</td><td colspan="2">(37) CO2 in air
(38) at 20±2℃, 65±5%RH</td></tr>
<tr><td colspan="2">(39) Circuit condition</td><td colspan="2">(40) VH=5.0±0.05V DC</td></tr>
<tr><td colspan="2">(41) conditioning period before test</td><td colspan="2">(42) 12hours or longer</td></tr>
</table>

Calibration

TGS 4161센서의 민감도 특성 및 습도와의 관계는 다음의 그래프와 같으며 그래프에서 도출한 관계식을 CO2감지 Programing 코드에 적용하였다.



[CO_2 센서의 민감도 특성]



[CO_2 센서의 습도와의 영향]

[CO_2 센서 변수 선언 Code]

```
//CO2 sensor
int    CO2Gain=1;
float  CO2Volt;
float  CO2ppm;
float  EMF1;//EMF1 in 350ppm  CO2 (mV)
float  EMF2;//EMF2 in listed gas concetration
float  dEMF;//dEMF=EMF1-EMF2
char   stringCO2ppm[10];
char   stringCO2Volt[10];
char   temper[10];
```

[CO_2 센서 Calibration Code]

```
// CO2 sensor TGS4161, Response time = 90s
// CO2 sensor can not be used simultaneously with O2 sensor
SensorGasv20.readValue(SENS_CO2);
CO2Volt=0;
for(int i=0; i<10; i++)
{ CO2Volt=CO2Volt+SensorGasv20.readValue(SENS_CO2);
delay(50); }
CO2Volt=CO2Volt/10;
    EMF1=316.399;//mV
EMF2=CO2Volt*1000;//change as mV
dEMF=EMF1-EMF2;

if(dEMF<30)
{
CO2ppm=350.*exp(0.0349941*dEMF);
}
else if(30<=dEMF && dEMF<60)
{
      CO2ppm=333.333*exp(0.0366204*dEMF);
}
else if(60<=dEMF)
{
CO2ppm=270.*exp(0.0401324*dEMF);
}

Utils.float2String(CO2Volt, stringCO2Volt, 2);
Utils.float2String(CO2ppm, stringCO2ppm, 2);
if(CO2ppm >=1000){
timeDelay=1000;
}
else{ timeDelay=10000; }
```

□ O2센서

일반사항

일반적으로 O2 결핍 규제는 우리나라와 일본의 경우 18%이하 이고, 미국의 경우 19.5%로 규정하고 있다. 또한 산소농도가 8~10%일 경우에 실신 및 구토가 일어날 수 있으며 4~6% 산소농도에서는 40초 내로 혼수상태 및 사망에 이르기도 한다. 본 연구에서 사용한 O2 센서는 Figaro社의 SK-25F를 사용하였다. SK-25F는 안정적이고 신뢰성 높은 결과값 도출이 가능하고, 다른 산소센서와는 달리 추가적인 전력공급 없이 작동가능하며, warmup시간이 필요없다는 장점을 가지고 있다.

O2센서의 자세한 Spec과 Hardware는 다음과 같다.

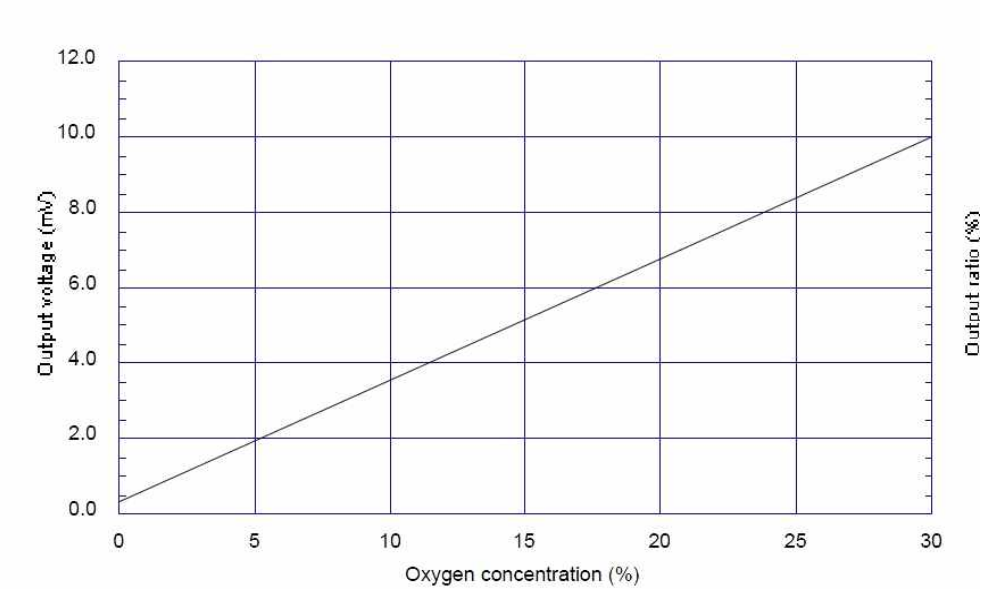


(43) Measurement range		(44) 0~30% O2
(45) Accuracy (Note 1)		(46) ±1% full scale
(47) Operating conditions	(48) Atmospheric pressure	(49) 1013 hPa±20%
	(50) Temperature	(51) -10~50℃
	(52) Relative humidity	(53) 0~99% RH (54) (no condensation)
(55) Response time (90%) (Note 2)		(56) ≤15 seconds
(57) Initial output voltage under standard test conditions		(58) 5.5~8.5 mV
(59) Standard test (60) conditions	(61) Atmospheric pressure	(62) 1013±5 hPa
	(63) Temperature	(64) 20°±1℃
	(65) Relative humidity	(66) 60±5% RH
(67) Life expectancy at 20℃ in normal air (Note 3)		(68) approx. 3 years

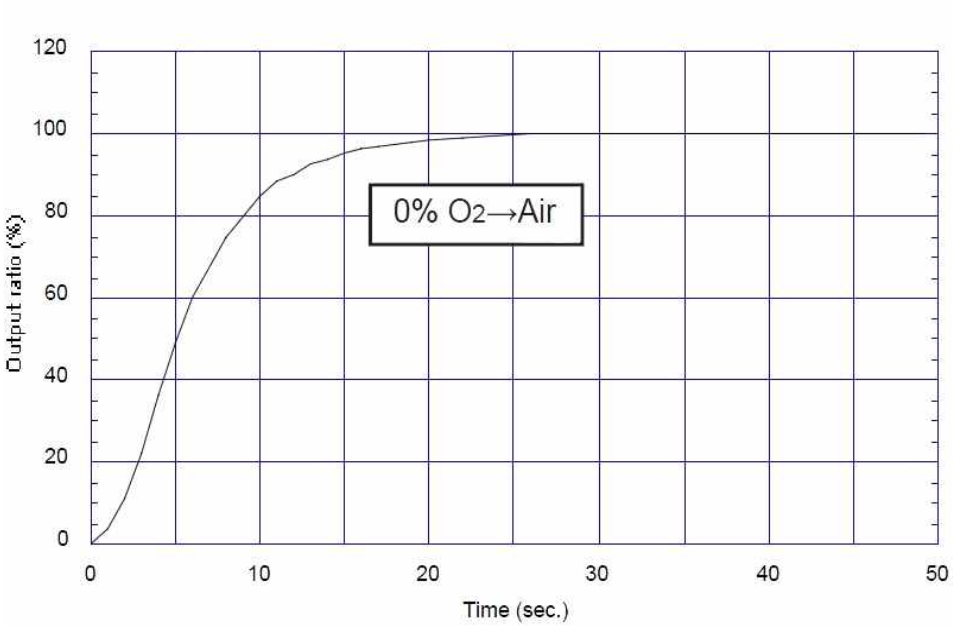
Notes:
1) 산소의 측정치가 2~30%인 경우, 0~30% 사이에서의 정확성은 오차범위 1%이내여야 한다.
2) 센서는 공기순환이 200~300㎖/min 보다 큰 상황에서 사용되어야 한다.
3) 정상적인 공기온도 20℃에서의 수명은 센서의 사용기간의 원래수명의 60%까지 떨어진다.

Calibration

SK-25F 센서의 민감도 특성 및 일반적인 상황에서의 응답시간 관계는 다음의 그래프와 같으며 이 관계식에서 도출한 Calibration 값을 O2감지 Programing에 적용하였다.



[O_2 센서의 민감도 특성]



[O_2 센서의 일반적인 응답시간]

[O_2 센서 변수 선언 Code]

```
//O2 sensor
int   O2Gain=1000;
float O2Volt;
float O2percent;
// O2 sensor SK-25 sensitivity characteristics (0%,0.3354mV)~(30%,10mV) results errors
// The following code in Libelium example shows good results
int   O2calibrationConcentration[3] = {0,0,0}; // percent
float O2calibrationOutput[3] = {0.3354,0,0}; // O2 Sensor SK-25 mV value
char  stringO2percent[10];
char  stringO2Volt[10];
```

[O_2 센서 Calibration Code]

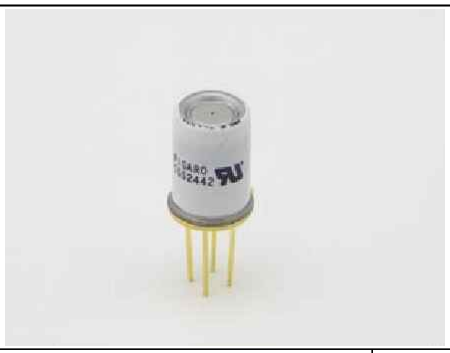
```
// O2 sensor SK-25, Response time = 15s
SensorGasv20.readValue(SENS_O2);
O2Volt = 0;
    for(int i=0; i<10; i++){
      O2Volt = O2Volt + SensorGasv20.readValue(SENS_O2);
      delay(50);
    }
    O2Volt = O2Volt/10;
O2percent = SensorGasv20.calculateConcentration(O2calibrationConcentration,O2calibrationOutput,O2Volt);
Utils.float2String(O2Volt, stringO2Volt, 2);
Utils.float2String(O2percent, stringO2percent, 2);
if(O2percent <=18){
timeDelay=1000;
}
else{
timeDelay=10000;
}
```

□ CO센서

일반사항

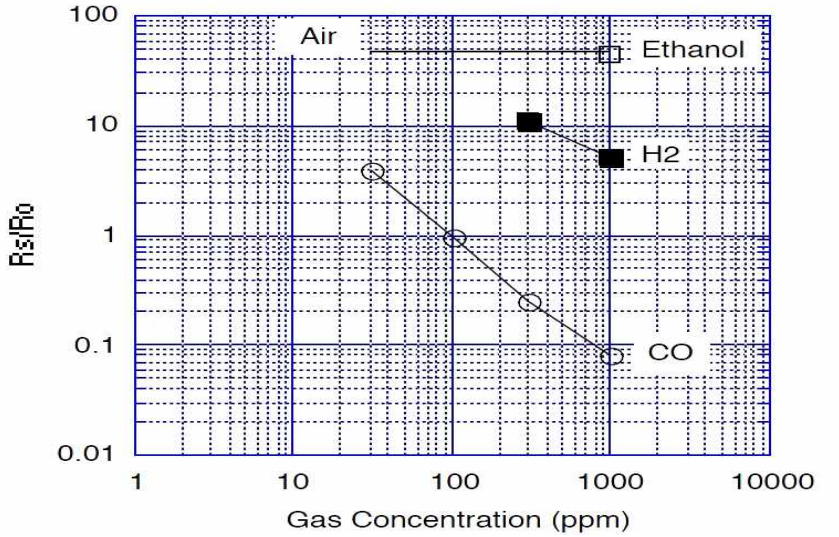
일반적으로 CO의 기준범위는 1시간 지속 노출 시 허용농도는 25ppm 이하이며 800ppm이상일 경우 2시간 내 의식을 잃고 사망할 가능성이 있으며 3,200ppm이상일 경우 30분 이내로 사망할 수 있다. 본 연구에서 사용한 CO 센서는 Figaro社의 TGS 2442를 사용하였다. TGS 2442는 크기가 작으며 알코올 및 습도의 영향이 적다. 또한, 저전력 소모와 긴 사용수명으로 경제적인 장점을 가지고 있다.

CO센서의 자세한 Spec과 Hardware는 다음과 같다.

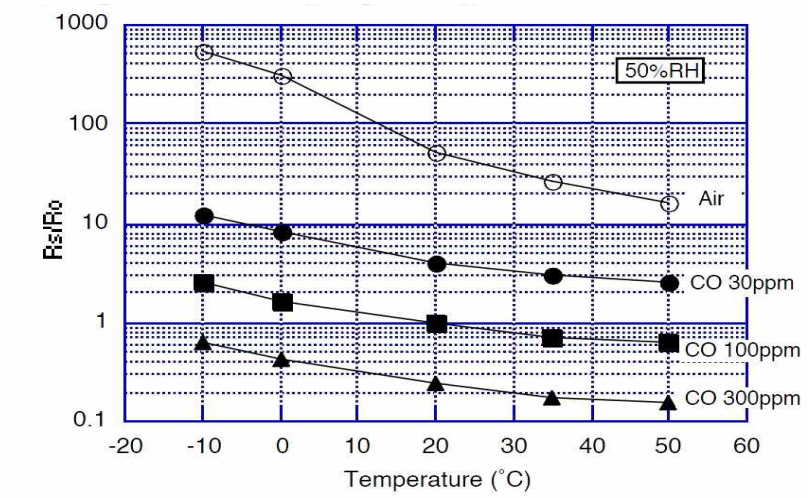
<table>
<tr><td colspan="4">  </td></tr>
<tr><td colspan="2">(69) Typical detection range</td><td colspan="2">(70) 30 ~ 1000 ppm</td></tr>
<tr><td rowspan="3">(71) Standard circuit conditions</td><td>(72) Heater voltage cycle</td><td>(73) VH</td><td>(74) VHH=4.8V±0.2V DC, 14ms
(75) VHL=0.0, 986ms</td></tr>
<tr><td>(76) Circuit voltage cycle</td><td>(77) VC</td><td>(78) VC=0V for 995ms,
(79) VC=5.0V±0.2V DC for 5ms</td></tr>
<tr><td>(80) Load resistance</td><td>(81) RL</td><td>(82) variable (≥10kΩ)</td></tr>
<tr><td rowspan="4">(83) Electrical characteristics under standard test conditions</td><td>(84) Heater resistance</td><td>(85) RH</td><td>(86) 17±2.5Ω at room temp.</td></tr>
<tr><td>(87) Heater current</td><td>(88) IH</td><td>(89) approx. 203mA
(90) (in case of VHH)</td></tr>
<tr><td>(91) Heater power
(92) consumption</td><td>(93) PH</td><td>(94) approx. 14mW (ave.)</td></tr>
<tr><td>(95) Sensor resistance</td><td>(96) RS</td><td>(97) 13.3kΩ~133kΩ in 100ppm of
(98) carbon monoxide</td></tr>
<tr><td>(99) Sensitivity
(100) (change ratio of RS)</td><td>(101) β</td><td>(102) 0.13~0.31</td></tr>
<tr><td rowspan="3">(103) Standard test conditions</td><td colspan="2">(104) Test gas conditions</td><td>(105) Carbon monoxide in air
(106) at 20±2℃, 65±5%RH</td></tr>
<tr><td colspan="2">(107) Circuit conditions</td><td>(108) Same as Std. Circuit Condition (above)</td></tr>
<tr><td colspan="2">(109) Conditioning period before test</td><td>(110) 2 days or more</td></tr>
</table>

Calibration

TGS 2442 센서의 민감도 특성 및 온도, 습도와의 관계는 다음의 그래프와 같으며 이 관계식에서 도출한 Calibration 값을 CO감지 Programing에 적용하였다.



[CO 센서의 민감도 특성]



[CO 센서의 온도, 습도와의 영향]

[CO 센서 변수 선언 Code]

```
//CO sensor
int    COGain=1;
int    CORLkOhm=100;
float  COVolt;
float  CORskOhm;
float  COppm;
int    COCalibrationConcentration[3] = {30,100,300};// ppm
float  COCalibrationOutput[3];// CO Sensor TGS2442 Rs/Ro value
char   stringCOVolt[10];
char   stringCOppm[10];
```

[CO 센서 Calibration Code]

```
// CO sensor TGS 2442, Response time = 1s
// If use with O2 sensor, insert CO sensor to SOCKET4
SensorGasv20.readValue(SENS_SOCKET4CO);
COVolt = SensorGasv20.readValue(SENS_SOCKET4CO);
CORskOhm = SensorGasv20.calculateResistance(SENS_SOCKET4CO, COVolt, COGain, CORLkOhm);
// Following equation is calculate by Mathematica
if(TMP <0){
  COCalibrationOutput[0]=15.0 - 0.7*(10.0 + TMP);
  COCalibrationOutput[1]=2.5 - 0.07*(10.0 + TMP);
COCalibrationOutput[2]=0.6 - 0.02*(10.0 + TMP);
    }
else if(0<= TMP && TMP <10){
  COCalibrationOutput[0]=8.0 - 0.20*(0.0 + TMP);
COCalibrationOutput[1]=1.8 - 0.04*(0.0 + TMP);
COCalibrationOutput[2]=0.4 - 0.01*(0.0 + TMP);
}
else if(10<= TMP && TMP <20){
COCalibrationOutput[0]=6.0 - 0.200*(-10.0 + TMP);
COCalibrationOutput[1]=1.4 - 0.040*(-10.0 + TMP);
COCalibrationOutput[2]=0.3 - 0.005*(-10.0 + TMP);
}
else if(20<= TMP && TMP <30){
COCalibrationOutput[0]=4.00 - 0.06*(-20.0 + TMP);
COCalibrationOutput[1]=1.00 - 0.02*(-20.0 + TMP);
COCalibrationOutput[2]=0.25 -0.005*(-20.0 + TMP);
}
else if(30<= TMP && TMP <35){
COCalibrationOutput[0]=3.4 - 0.080*(-30.0 + TMP);
COCalibrationOutput[1]=0.8 - 0.020*(-30.0 + TMP);
COCalibrationOutput[2]=0.2 - 0.004*(-30.0 + TMP);
}
else if(35<= TMP){
COCalibrationOutput[0]=3.00 - 0.03333330*(-35.0 + TMP);
COCalibrationOutput[1]=0.70 - 0.00666667*(-35.0 + TMP);
COCalibrationOutput[2]=0.18 - 0.00200000*(-35.0 + TMP);
}
COppm = SensorGasv20.calculateConcentration(COCalibrationConcentration,COCalibrationOutput,CORskOhm);
Utils.float2String(COVolt, stringCOVolt, 2);
Utils.float2String(COppm, stringCOppm, 2);
if(COppm >=25){
timeDelay=1000;
}
```

□ 조도, 소음, 온도센서

일반사항

본 연구에서 사용된 LDR(조도), 소음 및 온도의 경우 Waspmote API에 Calibration이 되어있어 추가적인 코딩없이 해당하는 Class를 호출하여 사용하였으며 각 센서의 자세한 Spec은 다음과 같다.

LDR(조도측정)

- Resistance in darkness : 20MΩ
- Resistance in light(10lux) : 5~20kΩ
- Spectral range : 400~700nm
- Min. consumption : 0μA



Microphone(소음측정)

- Sensitivity : -35±4dB
- Impedance : <2.2kΩ

- Directivity : Omni
- Frequency : 20Hz ~ 20kHz
- Supply voltage : +3V(std.), +10V(max.)
- Max. current consumption : 0.5mA
- Sensitivity reduction : -3dB at 1.5V
- Max. sound pressure level : 114.5±10dBSPL
- S/N ratio : 60dB
- Noise Level : 26±1dBA
- Stage measurement range : 50~100dBA

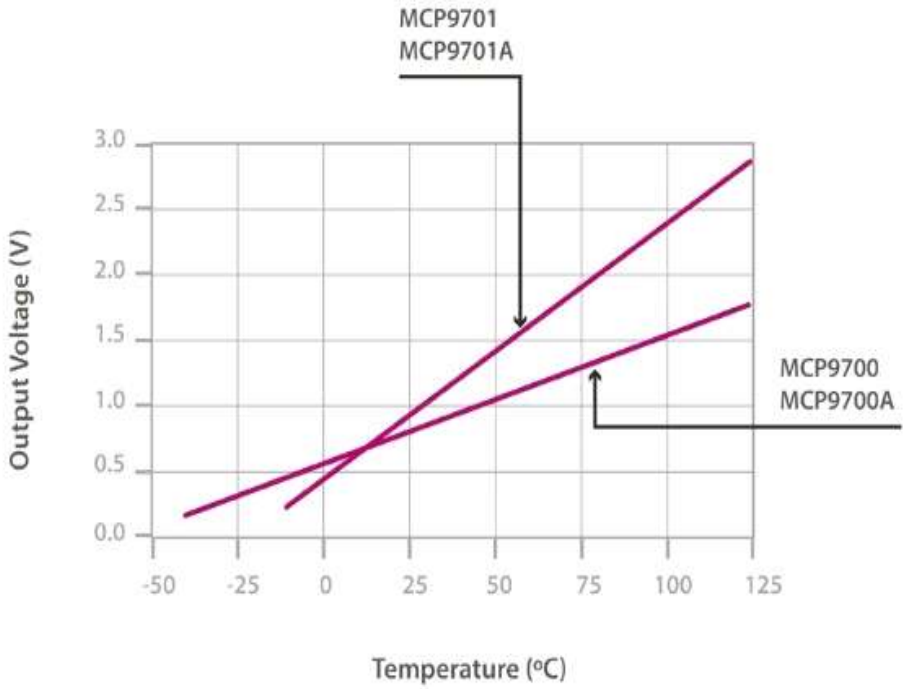


Temperature(온도측정)

- Measurement range : -40~125℃
- Output voltage(0℃) : 500mV
- Sensitivity : 100mV/℃
- Accuracy : ±2℃(0~+70℃), ±4℃(-40~+125℃)
- Supply voltage : 2.3~5.5V
- Response time : 1.65s
- Consumption : 6μA(typically), 12μA(Max.)



사용하는 온도센서의 종류에 따라 출력되는 전압의 변환값이 다르며 기본적으로MCP9700A센서로 설정되어 있어 추가적인 변환작업을 실시하지 않았다.



[온도센서 종류에 따른 온도측정값 변화 그래프]

□ VOC센서

일반사항

휘발성유기화합물(VOC) 가스의 탐지를 위해 SGX社의 MiCS-5524 센서를 사용하였으며 자세한 Sepc은 다음과 한다.

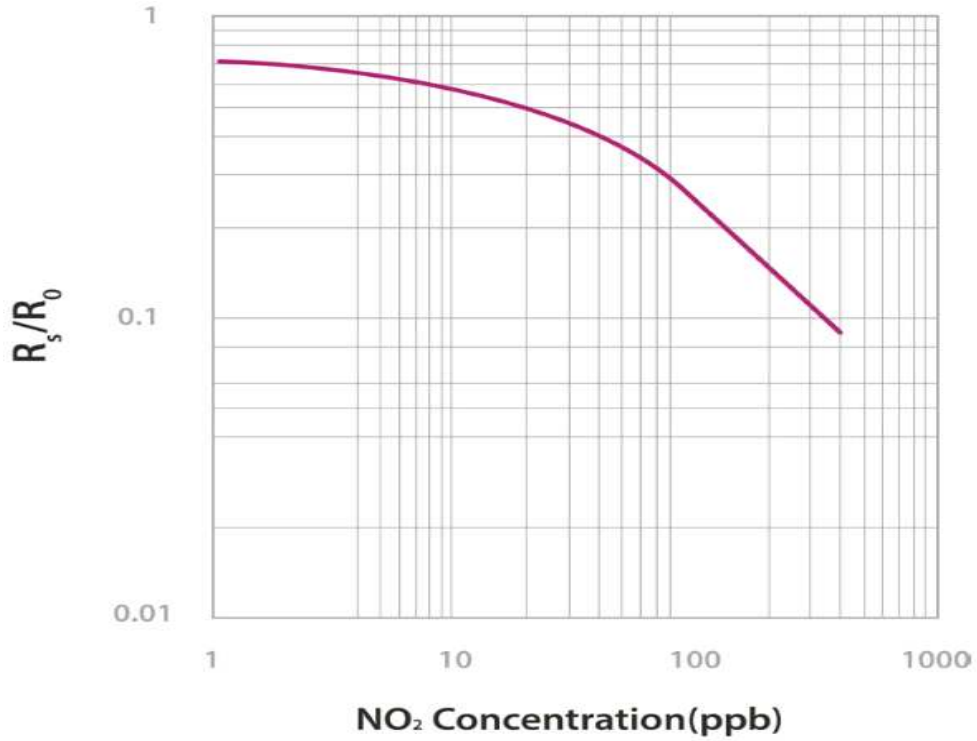
- Detecting Gases : CO, Hydrocarbons, Volatile Organic Compounds
- Measurement range : 30~400ppm

- Air resistance : 100~1500kΩ
- Sensitivity : 1.8 ~ 6
- Supply voltage : 2.1 ~ 5V DC
- Response time : 30s
- Average consumption : 32mA



Calibration

MiCS-5524 센서의 Rs/R0 값과 NO2와의 관계 그래프와 Programing에 적용된 값은 다음과 같다.



[R_s/R_0 값과 NO_2와의 관계 그래프]

[VOC 센서 변수 선언 및 Calibration Code]

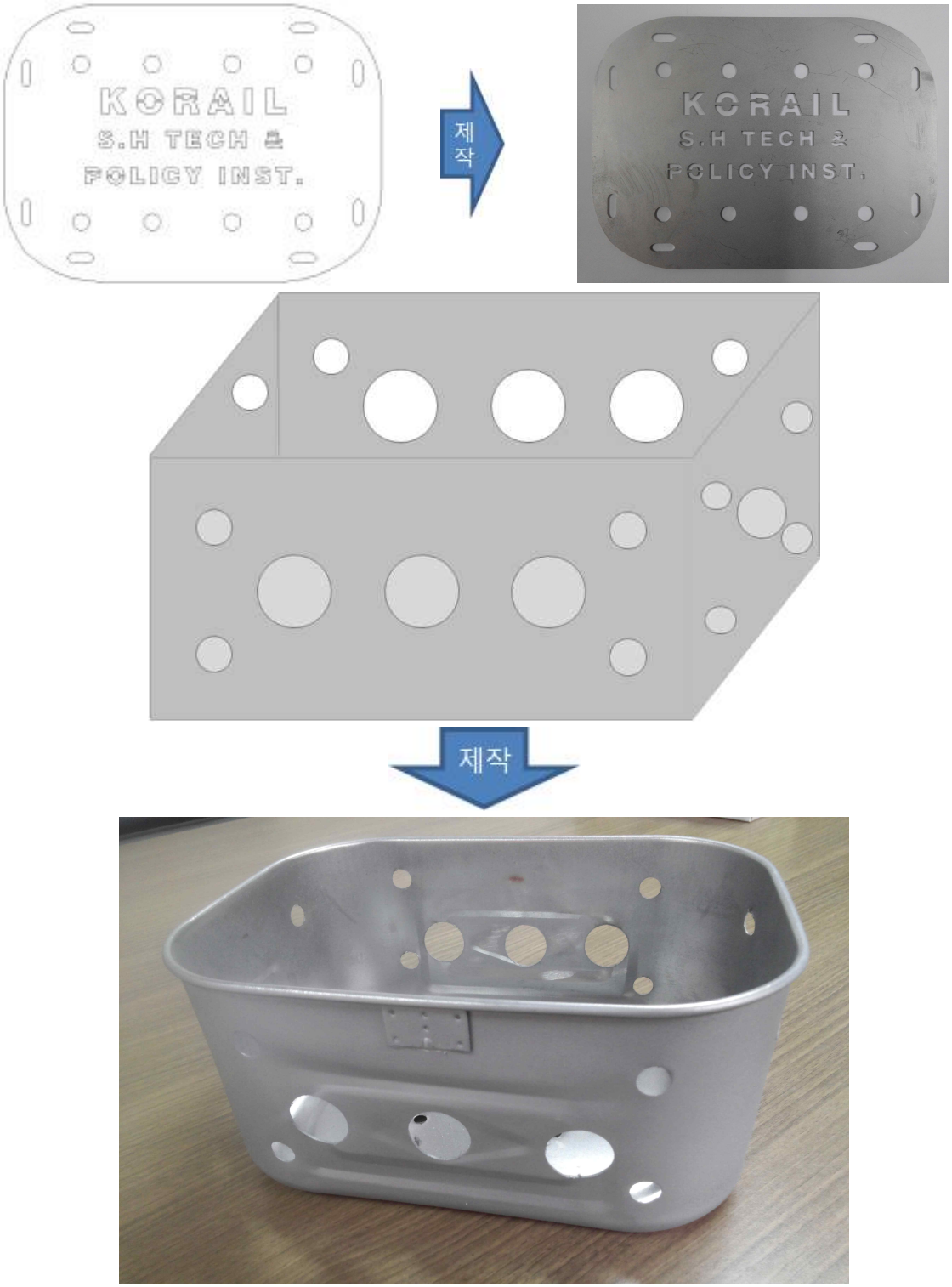
```
//VOC sensor
int    VOCGain=1;
int    VOCRLkOhm=20;
float VOCVolt;
char   stringVOC[10];


// VOC sensor MiCS-5524, Response time = 30s
SensorGasv20.readValue(SENS_SOCKET2B);
 VOCVolt = SensorGasv20.readValue(SENS_SOCKET2B);
Utils.float2String(VOCVolt, stringVOC, 2);
if(VOCVolt >=0.5){
timeDelay=1000;
}
else{
timeDelay=10000;
}
```

6) Waspmote Case제작

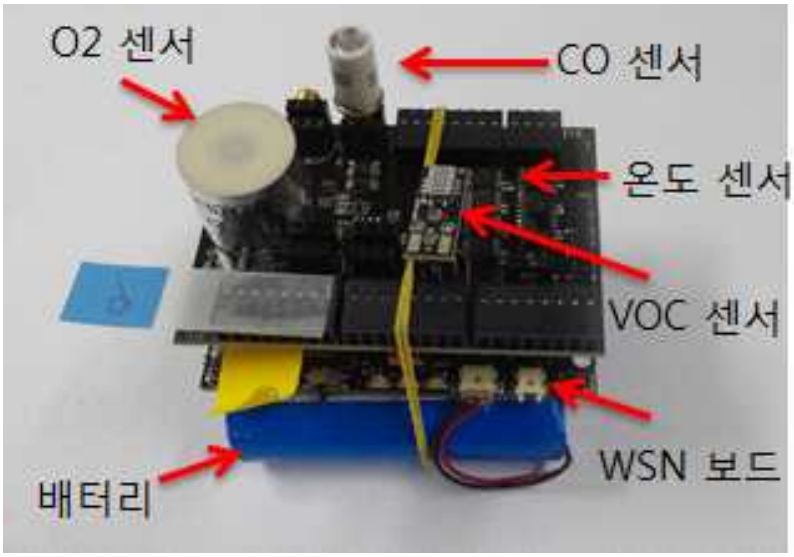
Waspmote Case의 가장 중요한 포인트는 ① 화재 발생 시에도 지속적으로 스마트N키퍼와의 통신을 유지하여 재난정보를 전송하기 위해서 뛰어난 내화성능을 가져야 하며 ②원활한 가스탐지를 위한 통풍성능을 가져야 한다는 것이다. 이러한 점에서 본 연구에서는 Stainless재질의 공기순환용 구멍을 뚫은 Case를 제작하였으며 내열 부식 억제용 Stainless 코팅제이용하여 Case의 표면처리를

하였다.

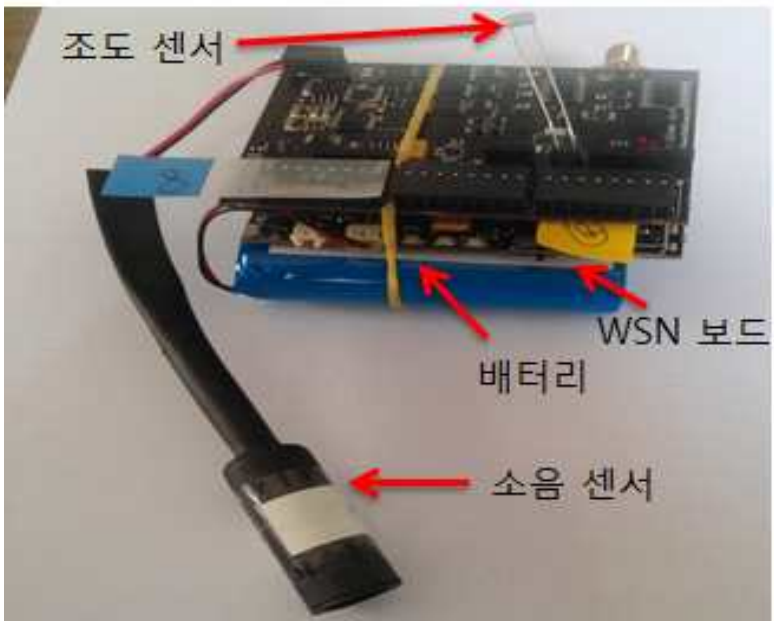


7) Node 제작

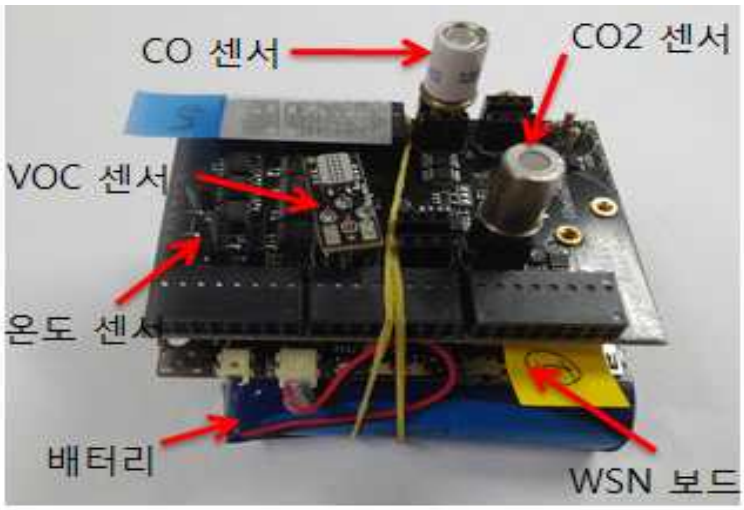
가스센서보드와 조도/소음센서보드를 이용하여 감지하고자 하는 가스와 용도에 따라 3가지 Type의 센서Node를 제작하였으며 각각 사용된 센서의 종류의 자세한 사항은 아래 그림과 같다.



[가스 보드 센서(O_2)]



[조도/소음 보드 센서]



[가스 보드 센서(CO_2)]

[센서보드 종류별 사용 센서]

(1) NO	(2) 조도	(3) 소음	(4) CO	(5) CO2	(6) O2	(7) Temp	(8) VOC	(9) 비고
(10) 1	(11) -	(12) -	(13) √	(14) -	(15) √	(16) √	(17) √	(18) O2 센서보드
(19) 2	(20) -	(21) -	(22) √	(23) √	(24)	(25) √	(26) -	(27) CO2 센서보드
(28) 3	(29) √	(30) √	(31) -	(32) -	(33) -	(34) -	(35) -	(36) 조도/소음 센서보드
(37) 4	(38) -	(39) -	(40) √	(41) √	(42) -	(43) √	(44) -	(45) CO2 센서보드
(46) 5	(47) -	(48) -	(49) √	(50) -	(51) √	(52) √	(53) √	(54) O2 센서보드
(55) 6	(56) -	(57) -	(58) √	(59) -	(60) √	(61) √	(62) √	(63) O2 센서보드
(64) 7	(65) -	(66) -	(67) √	(68) √	(69) -	(70) √	(71) -	(72) CO2 센서보드
(73) 8	(74) √	(75) √	(76) -	(77) -	(78) -	(79) -	(80) -	(81) 조도/소음센서보드
(82) 9	(83) -	(84) -	(85) √	(86) -	(87) √	(88) √	(89) √	(90) O2 센서보드
(91) 10	(92) -	(93) -	(94) √	(95) -	(96) √	(97) √	(98) √	(99) O2 센서보드
(100) 11	(101) -	(102) -	(103) √	(104) -	(105) √	(106) √	(107) √	(108) O2 센서보드
(109) 12	(110) √	(111) √	(112) -	(113) -	(114) -	(115) -	(116) -	(117) 조도/소음센서보드
(118) 13	(119) -	(120) -	(121) -	(122) -	(123) -	(124) -	(125) -	(126) Router(사당방향)
(127) 14	(128) -	(129) -	(130) -	(131) -	(132) -	(133) -	(134) -	(135) Router(사당방향)
(136) 15	(137) -	(138) -	(139) -	(140) -	(141) -	(142) -	(143) -	(144) Router(평촌방향)
(145) 16	(146) -	(147) -	(148) -	(149) -	(150) -	(151) -	(152) -	(153) Router(평촌방향)

나. NODE 설정 및 USB Gateway 설정

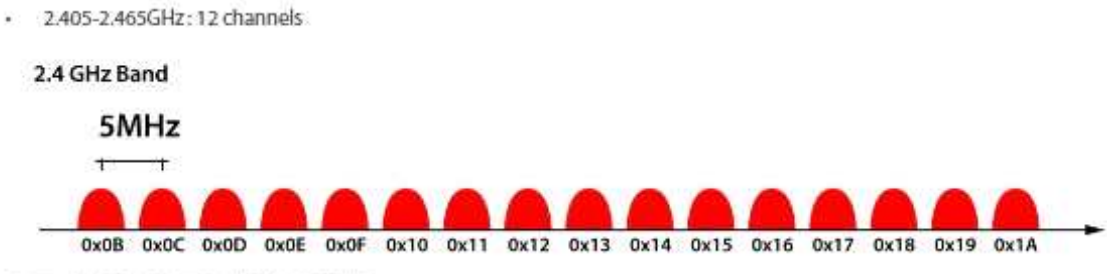
1) 센서Node 프로그램밍

센서Node가 감지하는 가스 및 조도, 온도 등에 따라 각각 다른 Programing Code를 작성하여 임베디드 하였다.

Step1.	헤더파일 선언 (가스보드, Digimesh, Waspmote Frame 등)
Step2.	변수 선언 (Node ID, 각각 사용 센서 등)
Step3.	USB Gateway Mac Address 설정 → 센서Node에서 6개의 스마트N키퍼용 USB Gateway로 Packet을 전송
Step4.	통신모듈 및 센서보드 Turn on → 센서감지 및 Packet 전송
Step5.	Mesh망을 통한 무선통신 → Router간의 통신을 통해 End Device(USB Gateway)로 Packet 전송

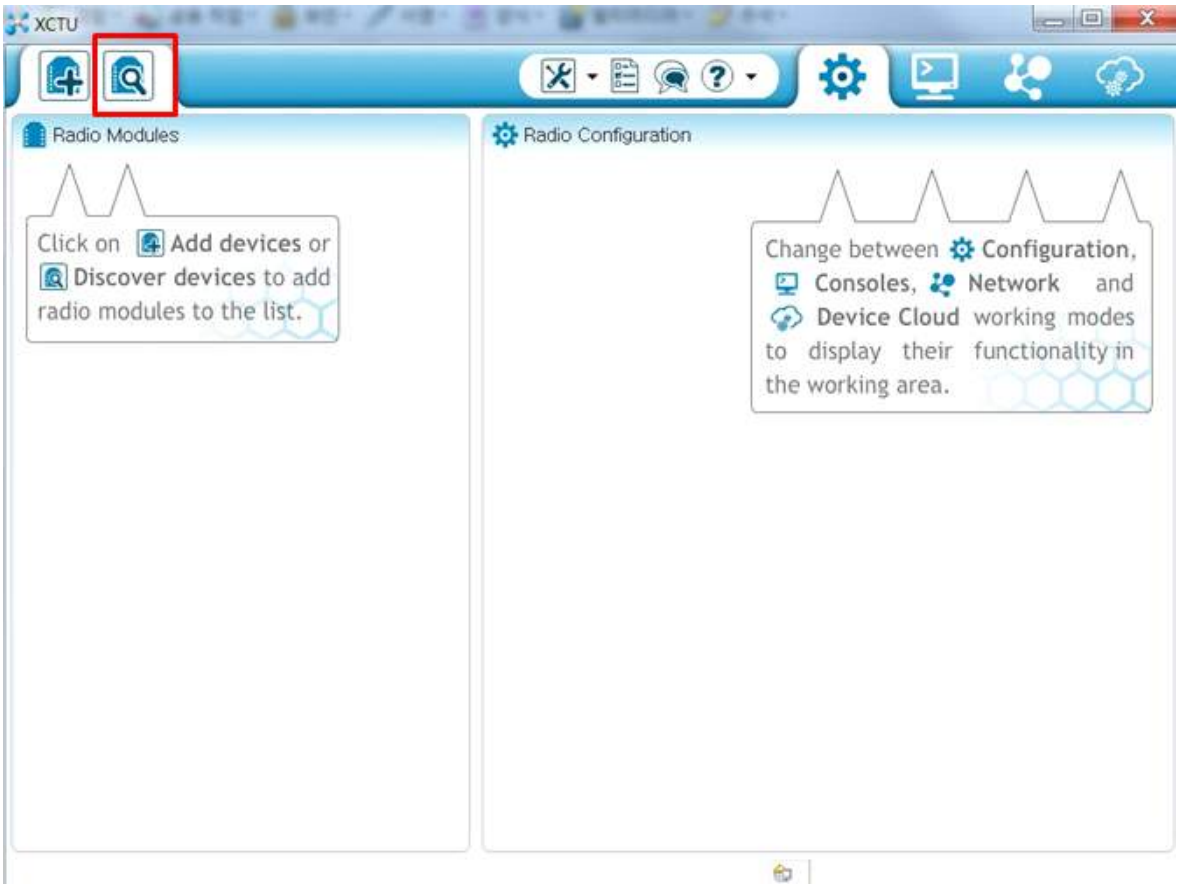
2) XCTU 설정

XBee-Pro 통신모듈을 이용한 무선네트워크를 사용하기 위해서는 모든 Node 및 Gateway의 채널과 PAN ID의 통일된 설정이 필요하다. 본 연구에서 사용하는 XBee-PRO의 주파수 영역대는 2.4GHz 이며 5MHz당 1개의 채널로 총 12개의 채널(2.405~2.465GHz)을 사용하며 자세한 내용은 다음의 그림과 같다.

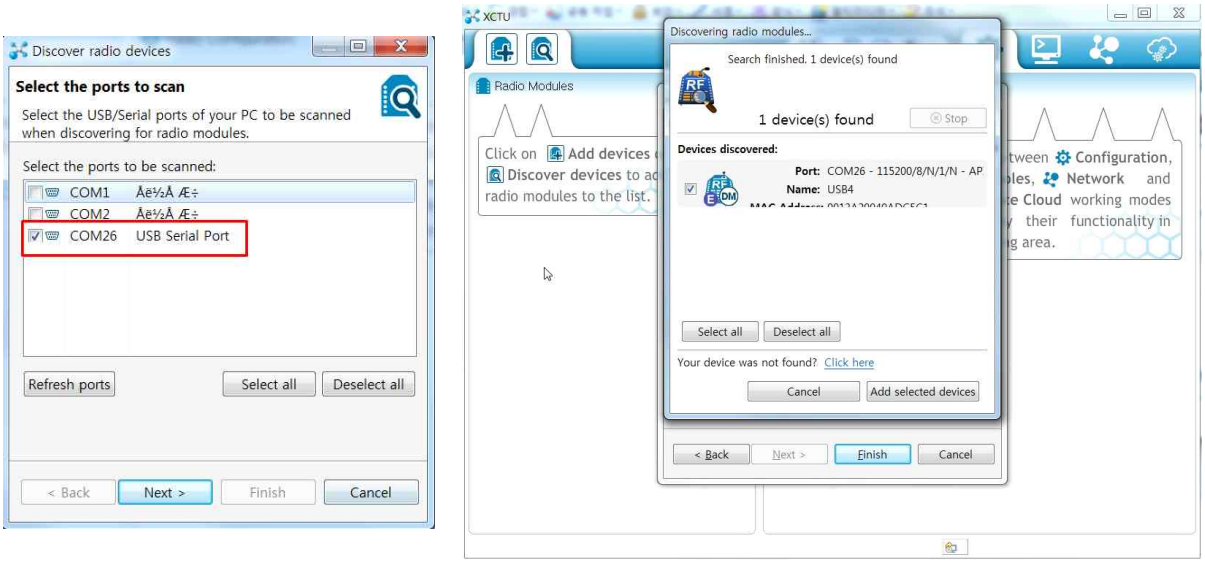


(1) Channel Number	(2) Frequency	(3) Supported by
(4) 0x0C –Channel 12	(5) 2,405 ~ 2,410 GHz	(6) PRO
(7) 0x0D –Channel 13	(8) 2,410 ~ 2,415 GHz	(9) PRO
(10) 0x0E –Channel 14	(11) 2,415 ~ 2,420 GHz	(12) PRO
(13) 0x0F –Channel 15	(14) 2,420 ~ 2,425 GHz	(15) PRO
(16) 0x10 –Channel 16	(17) 2,425 ~ 2,430 GHz	(18) PRO
(19) 0x11 –Channel 17	(20) 2,430 ~ 2,435 GHz	(21) PRO
(22) 0x12 –Channel 17	(23) 2,435 ~ 2,440 GHz	(24) PRO
(25) 0x13 –Channel 18	(26) 2,440 ~ 2,445 GHz	(27) PRO
(28) 0x14 –Channel 19	(29) 2,445 ~ 2,450 GHz	(30) PRO
(31) 0x15 –Channel 20	(32) 2,450 ~ 2,455 GHz	(33) PRO
(34) 0x16 –Channel 21	(35) 2,455 ~ 2,460 GHz	(36) PRO
(37) 0x17 –Channel 22	(38) 2,460 ~ 2,465 GHz	(39) PRO

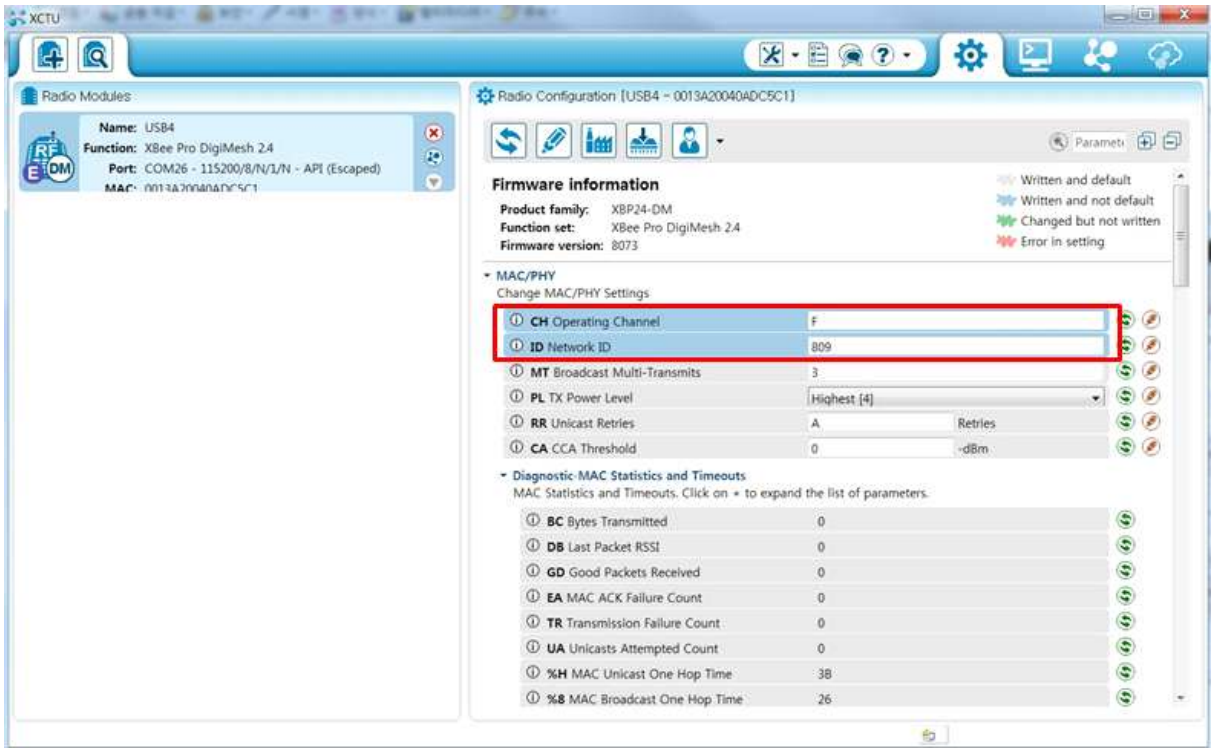
XBee-Pro의 통신모듈을 사용하기 위해서 Digi社에서 제공하는 프로그램인 XCTU를 이용하여 XBee 통신모듈의 설정을 해야하며 자세한 내용은 다음과 같다.



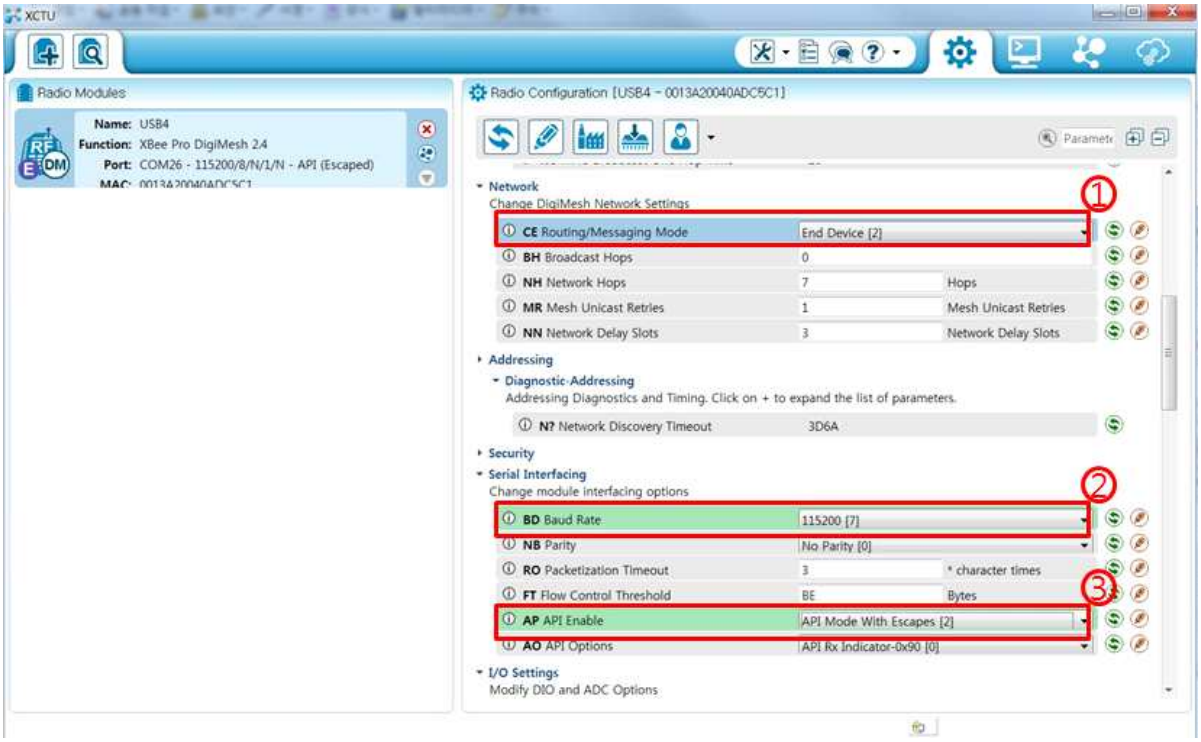
1. USB Gateway를 PC에 연결한 다음 통신모듈을 검색



2. USB Gateway의 해당하는 Port를 선택하고 검색을 하여 통신 장치를 선택



3. Node 및 스마트N키퍼에 사용되는 XBee-PRO 통신 모듈은 모두 동일한 채널과 PAN ID로 설정



4-1. XBee-PRO 통신모듈의 사용용도에 따라 CE Setting이 달라지며 센서 Node 및 Router의 경우 Standard Router로 설정하고 스마트N키퍼용 USB Gateway의 경우 End Device로 설정

4-2. Waspmote에서 송·수신하는 Date의 Baud Rate값은 115200이므로 통신모듈 설정

4-3. XBee는 AT와 API모드의 데이터를 송수신 방법을 지원하는데, AT모드는 메시지 데이터 자체를 보내고 받을 수 있으며 단순한 전송 및 수신이 가능하다. API모드는 프로그래머가 필요한 정보인 도착 주소, 패킷형태를 설정할 수 있으며 데이터를 패키지화 한다. 따라서 Node간에 정확하고 빠른 통신을 위해서는 유연성이 있는 API모드를 선택하여 설정

다. 무선 네트워크 망 설정

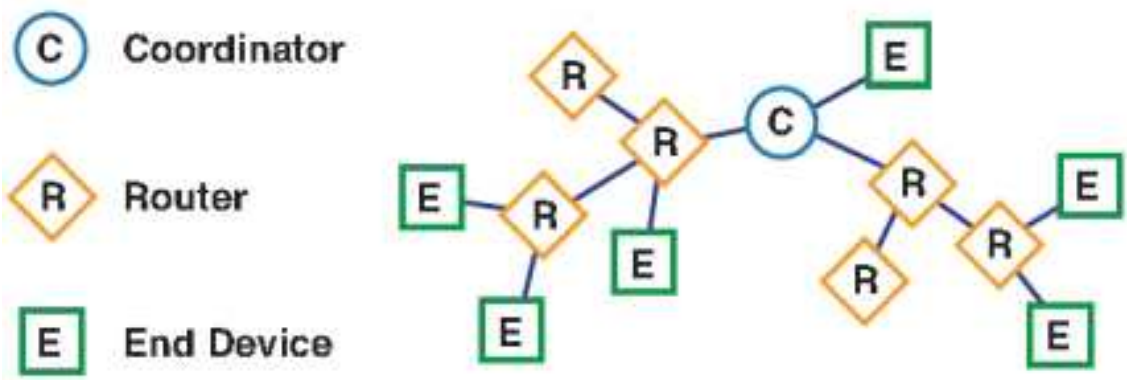
1) ZigBee 시스템과 XBee-PRO 의 차이점

□ ZigBee

ZigBee프로토콜은 3가지 형태의 Node를 가지고 있다.
① Coordinator - 각 네트워크를 구성하는데 있어서 1개의 Coordinator가 필요하다. 그리고 Security Keys를 포함한 Networking에 의한 Information은 Coordinator에 저장된다.

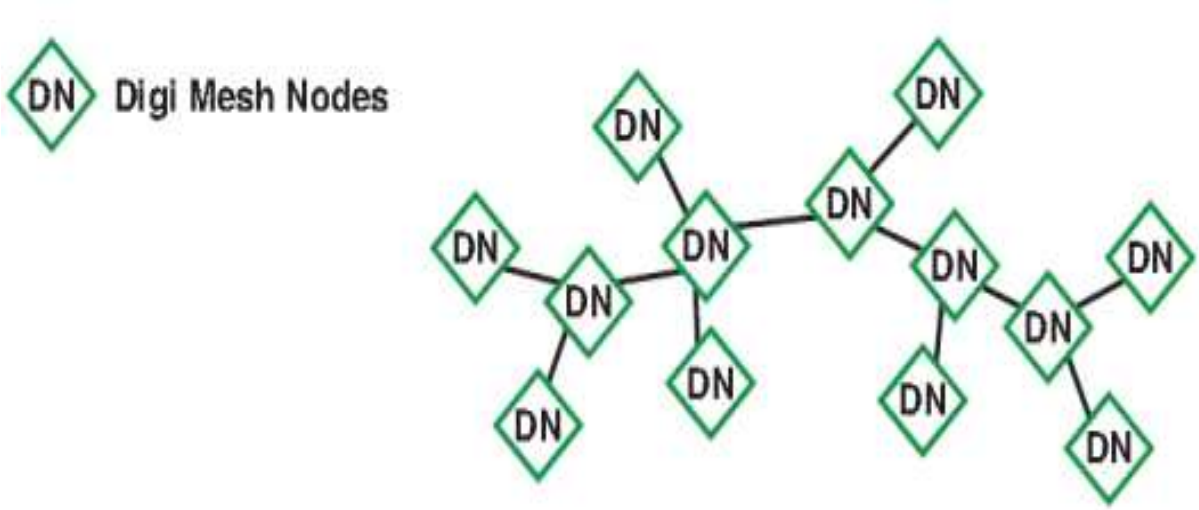
② Router - 중간 노드로써의 역할을 한다. Data를 다른 Device로 전달한다.

③ End Device - 이는 다른 노드들(Coordinator또는 Router)와 통신하며 다른 Device로 Data를 전달하지는 않는다. End Device는 Low-power/Battery-powered Devices역할을 한다. 이를 통해 비용절감의 효과를 얻을 수 있다.
이러한 ZigBee시스템은 여러 Vendor들 사이에서 정보처리 상호운용의 가능성과 함께 공개 표준화 되어 있다. 저비용으로 구현이 가능하지만, End노드들의 기능이 감소된다는 단점이 있다.
다음 그림은 ZigBee프로토콜 구성도이다.



□ DigiMesh Node(XBee-Pro)

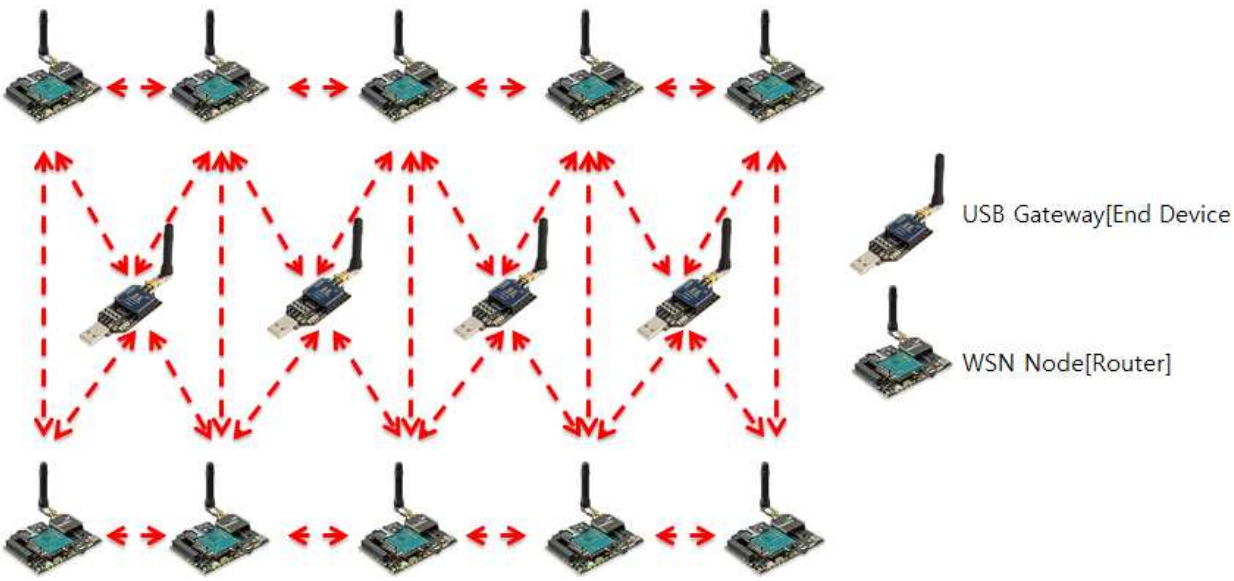
XBee DigiMesh는 ZigBee와는 다르게 한 가지 형태의 Node를 가지고 있다. HomogenousNetwork라 불리는 이 Node는 모든 Node들이 Data를 보내고 Data를 상호교체 할 수 있다. 서로의 Node들은 상위-하위관계가 아니며 모든 Node들이 ZigBee의 End Devices처럼 Low-power/Battery powered Devices 역할을 한다. XBee DigiMesh는 특히 ZigBee에서 Coordinator나 Router에서 발생하는 단일 고장점(Singlepointoffailure)을 해결한다는 장점을 가지고 있다. XBee DigiMesh를 사용하였을 경우 더 융통성있게 확장된다.
Network를 구현할 수 있으며, 단순한 Network 세팅과 간섭이나 데미지로인한 데이터 송·수신하는 Router들 간의 환경에서 향상된 신뢰성을 제공한다. 다음 그림은 XBee프로토콜 구성도이다.



□ ZigBee vs DigiMesh

ZigBee시스템은 공개 표준화되어 있기 때문에, 다른 Vendor들에 의해 만들어진 Device와 정보처리 상호운용의 가능성을 잠재적으로 제공한다. 이는 방송으로 펌웨어 업데이트를 하는 기능을 제공한다. 또한, ZigBee는 조명제어나 에너지 관리와 같은 공통된 application을 위한 프로파일이 설립되어 있다. DigiMesh는 전매 프로토콜로써 타이트한 코드공간의 제어를 허용하여 더 많은 공간을 가지는 특징이 있다. 그리고 더 넓은 통신범위와 RFdatarate을 제공하며 큰 Data블록을 보내는 application의 처리량을 향상시키기 위해 Framepayload가 일반적으로 더 크다. 또한,DigiMesh는 Networksetup이나Troubleshooting등을 향상시키는 기능을 위해 간략화 된 어드레스 처리방식을 사용한다.
다음 표는 전반적인 ZigBee시스템과 XBee시스템을 비교한 표이다.

(1)	(2) ZigBee	(3) DigiMesh(XBee-PRO)
(4) 노드타입, 장점	(5) Coordinators, Routers, End Devices로 구성. (6) 적은 기능을 가진 End Devices의 잠재적인 적은 비용.	(7) homogenous의 한가지 타입으로 구성. (8) 네트워크 확장을 위한 좋은 유연성. (9) 간단한 네트워크 셋업. (10) 간섭이나 데미지로 인한 Routers의 통신환경에서의 신뢰성 향상.
(11) Sleep 기능의 Routers, (12) 배터리 수명	(13) 오직 End Devices만 sleep 기능	(14) 모든 노드 sleep기능. 시간 (15) 동기화를 위한 gateway 나 Coordinator에서 발생하는 단일 고장점 해결.
(16) 통신 범위 옵션	(17) 대부분 ZigBee장비는 각 hop당 적어도 2마일(3.2Km)의 범위를 가짐	(18) XTendTM에 관하여 가 hop당 40마일(64Km)까지의 범위를가짐
(19) 프레임 payload, 처리량	(20) 80 bytes 까지	(21) 현 제품 사용시 256 bytes까지. (22) 큰 블록의 데이터를 보내는 어플리케이션을 위한 처리량 향상
(23) 코드 사이즈	(24) 큼.	(25) 작음
(26) 지원되는 주파수대역과 RF Date Rate	(27) 대게 2.4 GHz (250 kbps). (28) 900 MHz (40 kbps) 그리고 868 MHz (20 kbps) 넓게 이용하지는 못함.	(29) 900MHz (10, 125, 150 kbps) (30) 2.4GHz (250 kbps)
(31) 보안	(32) AES 암호화. 네트워크를 제재할 수 있으며, 다른 노드들로 부터의 참여를 예방.	(33) AES 암호화.
(34) 정보처리 상호운용 (35) 가능성	(36) vendor들 사이의 정보처리 상호운용 가능성 높음	(37) 상표 등록된 회사 (38) 정보처리 상호운용 가능성 낮음
(39) 간섭 허용 오차	(40) 직접 주파수 확산(DSSS)	(41) 900 MHz : 주파수 도약 확산 (42) (FHSS) (43) 2.4GHz : 직접 주파수 확산(DSSS)
(44) 어드레스 지정	(45) 두 가지 레이어. (46) MAC address(64 bit) (47) Network address(16 bit)	(48) 한 가지 레이어. (49) MAC address(64 bit)
(50) 유지	(51) 마켓에서 많은 탐지기와 진단 (52) 메시지 도구들 이용가능	(53) 진단문제에 도움을 주고 network 설립에 도움을 주는 (54) 간단한 어드레스 지정.

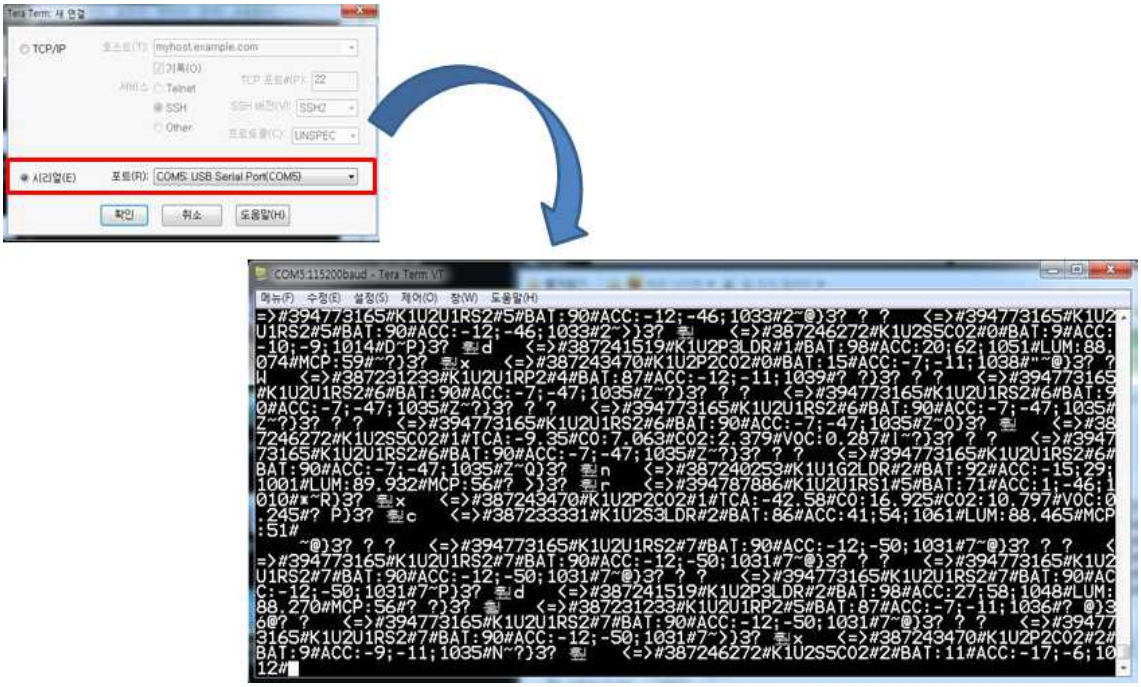


[DigiMesh topology]

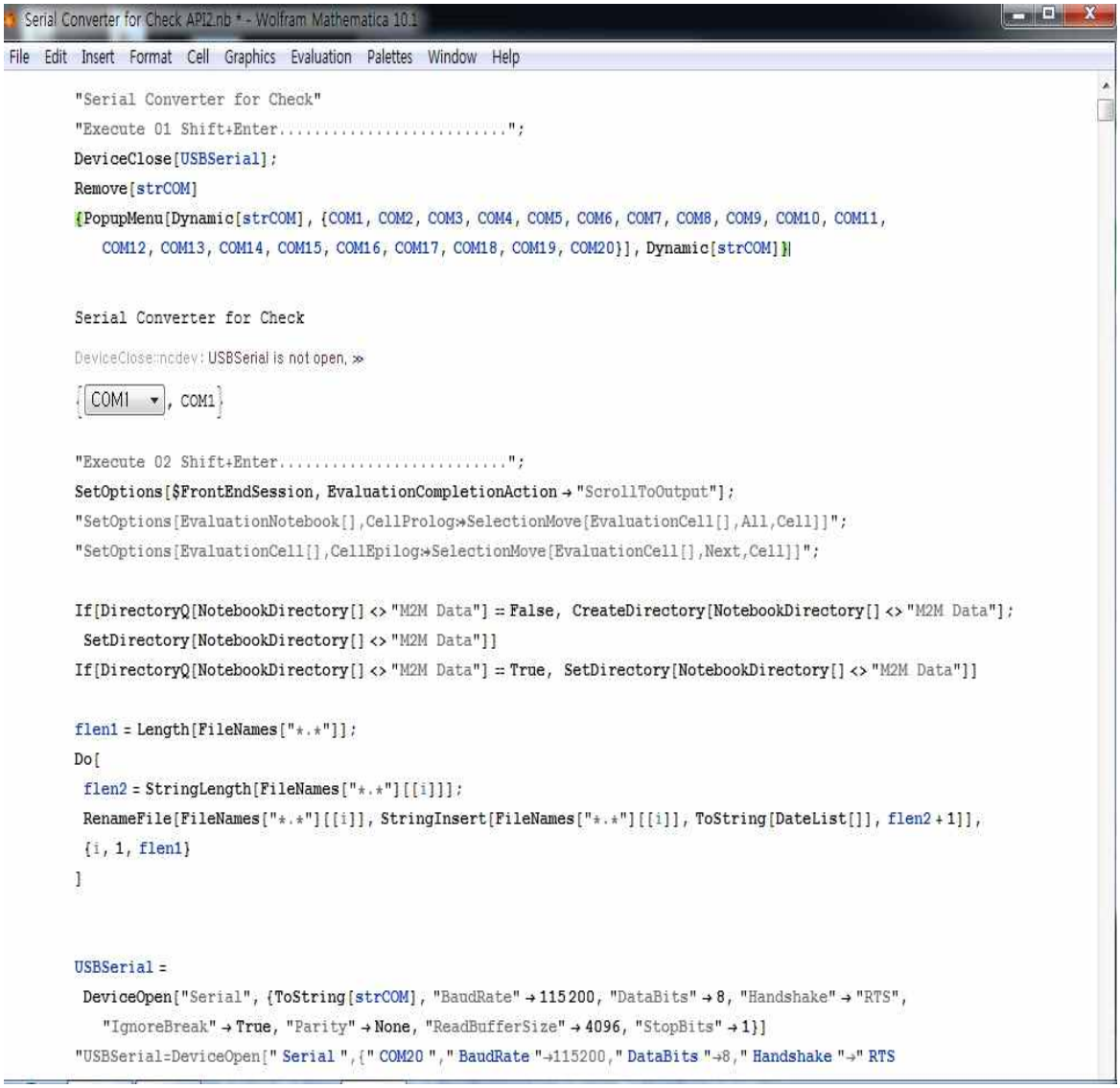
2) 시리얼 터미널

각각의 Node에서 스마트N키퍼의 USB Gateway로 송신되는 Date는 시리얼 터미널을 통해 확인할 수 있으며 Waspmote의 경우, 발신하는 Date의 Baud Rate는 115200으로 설정해야 한다. 여러 시리얼 터미널 중 본 연구에서는 Tera Term과 Mathematica 기반의 *.cdf파일을 실행하는 CDF Player Pro의 차이를 확인하고자 한다.

Tera Term는 오픈 소스 소프트웨어로서 조작이 간편하여 사용이 용이하지만 12개의 Node에서 송신되는 Date들의 값이 비순차적으로 밖에 확인할 수 없어 Date의 신뢰성과 직관성이 매우 떨어진다. 다음의 그림은 Tera Term의 설정방법 및 구동모습이다.



반면, CDF Player Pro의 경우 Node에서 들어오는 비순차적으로 들어오는 Date들을 하나로 통합하여 Gage로 표현이 가능하다. 따라서 시각화된 Gage를 통해 이용객들이 온도, 산소농도 등과 같이 센서링 되어지는 정보의 확인이 가능한 장점을 갖고 있어 현장구축에 용이할 것으로 판단하여 모든 스마트N키퍼에 CDF Player Pro의 Licence를 구입하여 설치하였다. CDF Player Pro의 코딩 및 구동모습은 아래 그림과 같다.



[CDF Player PRO 코딩 Part]

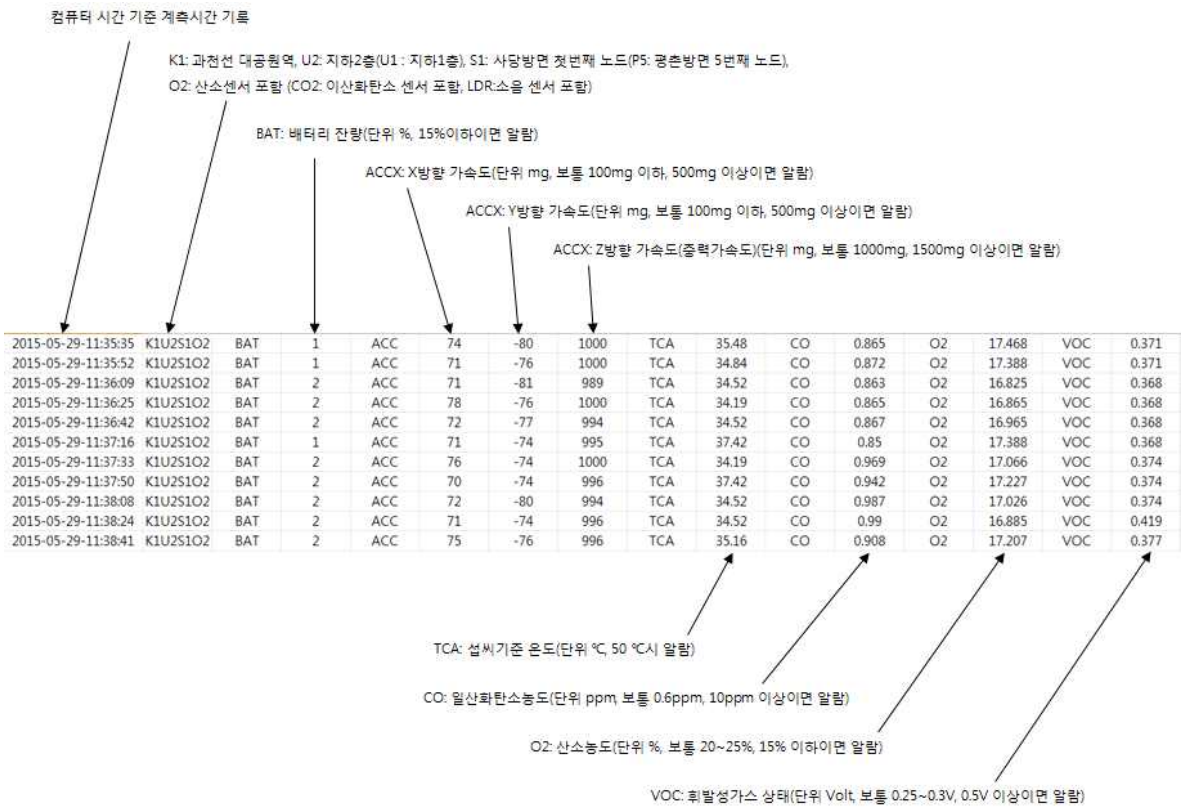


[CDF Player PRO 구동모습 01]

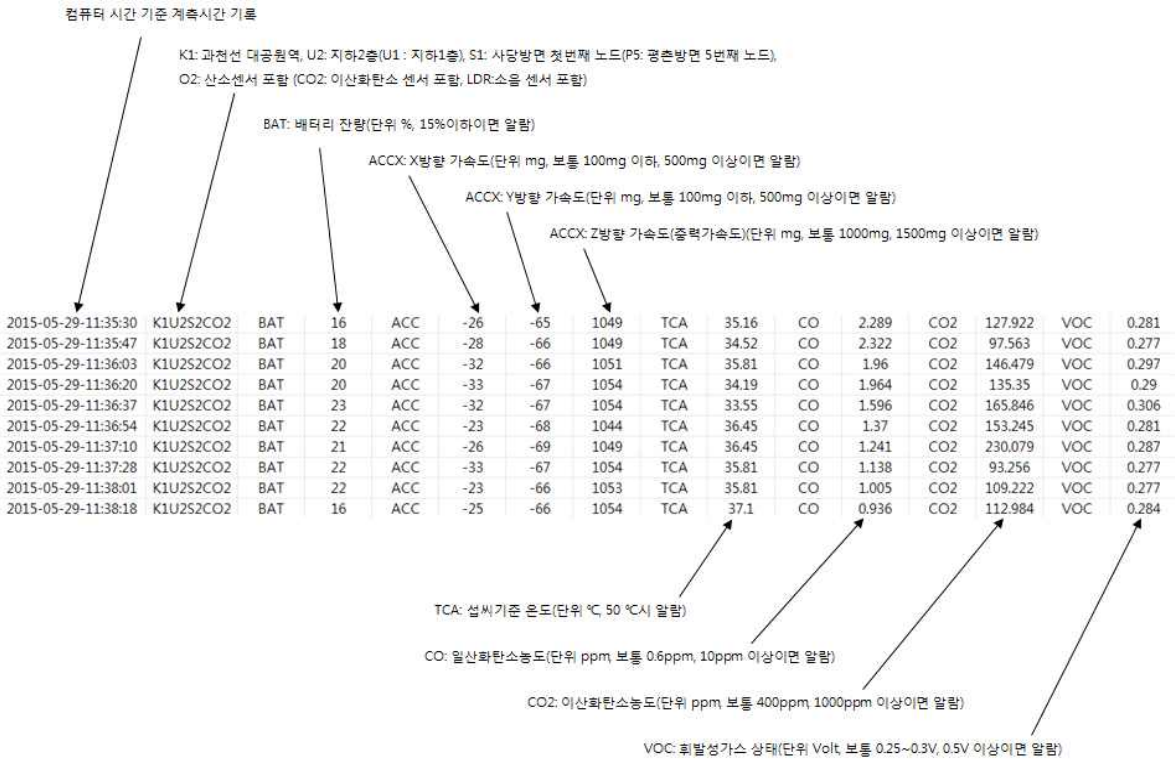


[CDF Player PRO 구동모습 02]

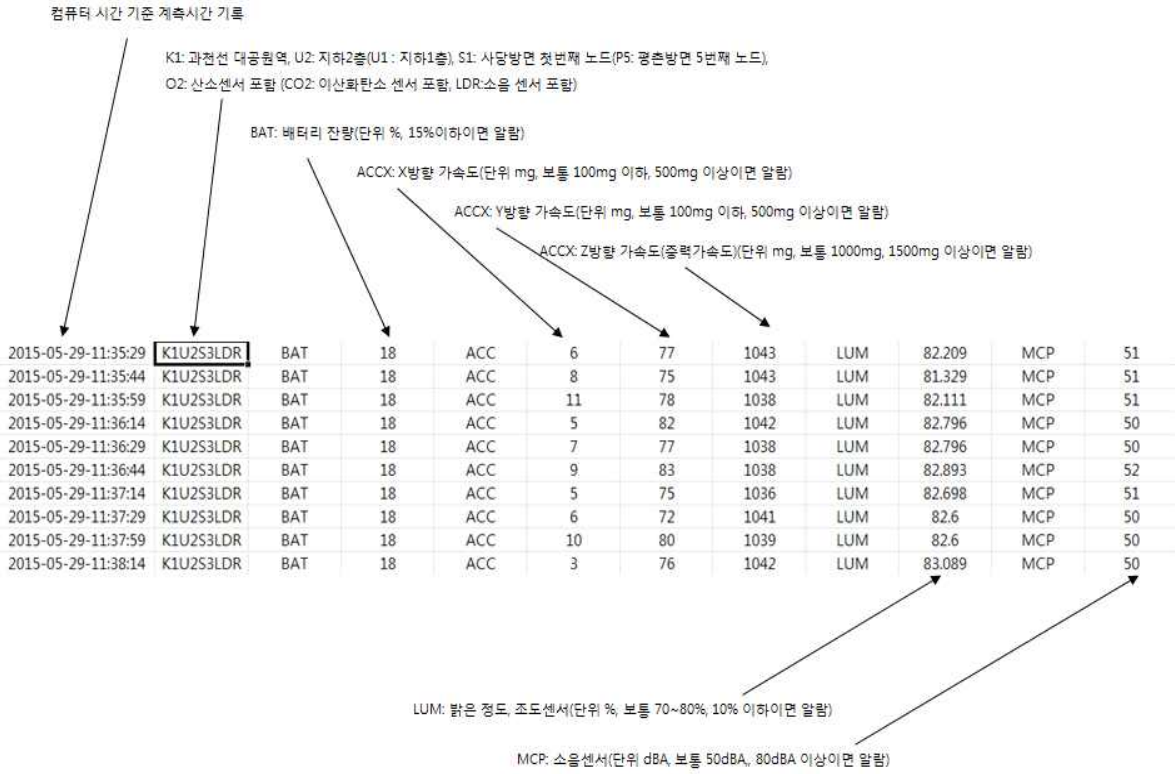
또한, 수신되는 Date들은 Node 고유 ID와 동일한 이름의 *.dat파일이 생성되며 *.dat파일에 대한 설명은 아래의 그림과 같다.



[O_2 센서보드에서 수신된 dat파일 예]



[CO_2 센서보드에서 수신된 dat파일 예]



[조도/소음 센서보드에서 수신된 dat파일 예]

라. 센서노드 파싱 모듈 API 개발

각각의 스마트N키퍼는 nodeJS 기반으로 센서노드로부터 수신된 데이터를 수집하고 저장한다. 또한 클라이언트 요청에 따라 데이터를 전송한다. 프로세스를 수행하는 의존성 모듈과 데이터 정의는 아래 코드와 같다.

[의존성 모듈 로딩 및 데이터 타입 정의]

```
var express = require('express');
var cors = require('cors');
var path = require('path');
var fs = require('fs');

var Parser = require('binary-parser').Parser;
var mysql = require('mysql');
var serial = require("serialport");
var buff = require('./buff2array');
var zpad = require('zpad');
var converter = require('json-2-csv');
var compare = require('buffer-compare');
var jsonfile = require('jsonfile');
var redis = require("redis"),
    redisClient = redis.createClient({host: '192.168.0.100', password: '.});
    redisClientLocal = redis.createClient({password: '.});
var util = require('util');
var mongoClient = require('mongodb').MongoClient;
var mongoServer = require('mongodb').Server;
redisClient.on("error", function(err) {
    console.log("Redis Error " + err);
});
redisClientLocal.on("error", function(err) {
    console.log("Redis Error " + err);
});

var app = express();
app.use(cors());
app.set('views', path.join(__dirname, 'views'));
app.use(express.methodOverride());
app.use(require('express-domain-middleware'));
app.use(express.static(path.join(__dirname, 'views')));
app.use(express.bodyParser());
app.use(app.router);

app.get('/', function(req, res) {
    res.render('pages/index.ejs');
});

var client;

function DataStruct(serialID, waspmoteID, NOB, FOS, TST, name, tag, value, unit) {
    this.SerialID = serialID;
    this.WaspmoteID = waspmoteID;
    this.NumberOfBytes = NOB;
    this.FrameOfSequence = FOS;
    this.TimeStamp = TST;
    this.SensorName = name;
    this.SensorTag = tag;
    this.SensorValue = value;
    this.SensorUnit = unit;
}
```

[시리얼 데이터 수집 및 센서별 파싱 및 DB저장]

```
var SerialPort = serial.SerialPort;
var serialPort = new SerialPort(
```

```
    serialConf.comName, {
        baudrate: serialConf.baudRate
    }, true,
    function(err) {
        if (err) {
            return console.log(err.message);
        }

        console.log('## serial open');
        //const atcommand = new
Buffer('01534554234652454 33A43485F30385F3930303B414444523A313B42573A42575F3132353B43523A43525F353B53463A534
65F31320D0A3642353004', 'hex');
        // serialPort.write(atcommand);
        // console.log(atcommand);
        // console.log('## sent LoRa 900 atcommand')

        serialPort.on('data', function(data) {

            // var json = JSON.stringify(data);
            // console.log(json);
            var buf = qdata.slice(data.length, qdata.length);
            buf.copy(qdata, 0, 0, buf.length);

            if (data.length <= buffersize) {
                data.copy(qdata, buffersize - data.length, 0, data.length);

                var index = compare(qdata, indicator);
                var qarray = buff.array(qdata);
                var count = 0;
                var startindex = [];
                var nowarray = [];

                for (i = 0; i <= qarray.length - 3; i++) {
                    if (qarray[i] == 60) {
                        if (qarray[i + 1] == 61) {
                            if (qarray[i + 2] == 62) {

                                startindex[count] = i;
                                count++;
                            }
                        }
                    }
                }
                if (count >= 2) {
                    nowarray = qarray.slice(
                        startindex[0] - 1, startindex[1]
                    );
                    qdata.fill(0, startindex[0] - 1,
                        startindex[1] - 1);

                    console.log(
                        '============================================================='
                    );

                    console.log('Raw data : ' + nowarray);
                    console.log('length : ' + nowarray.length);
                    var StartDelimiter = new Buffer(nowarray.slice(1, 4));
                    console.log('Start Delimiter : ' + StartDelimiter.toString('ascii'));
                    var FrameType = nowarray.slice(4, 5);
                    console.log('Frame Type : ' + FrameType);
                    var NumberOfBytes = nowarray.slice(5, 6);
                    console.log('Number Of Bytes : ' + NumberOfBytes);
                    var SerialID = new Buffer(nowarray.slice(6, 10));
                    var WaspmoteID = new Buffer(nowarray.slice(10, 18));
                    var Separator = nowarray.slice(18, 19);
                    console.log('Separator : ' + Separator);
                    var FrameSequence = nowarray.slice(19, 20);
                    console.log('Frame Sequence : ' + FrameSequence);
```

```
                    console.log('Serial ID : ' + SerialID.readUInt32LE(0, 4));
                    console.log('Waspmote ID : ' + WaspmoteID.toString('ascii'));
                    var ind = 20;
                    var k = 0;

                    var SensorTag = [];
                    var SensorName = [];
                    var SensorValue = [];
                    var SensorUnit = [];
                    var TimeStamp;
                    // var redisArgs = ['sensor:'+WaspmoteID.toString('ascii'),FrameSequence];
                    var serverTimeStamp = (new Date).getTime();
                    var redisArgs = { 'Frame Of Sequence': FrameSequence.toString(), 'Time Stamp Server': serverTimeStamp
};
                    var writeRedisInd = false;
                    while (ind - 5 <= NumberOfBytes) {
                        var sensorID = nowarray.slice(ind, ind + 1);
                        var Value;
                        if (sensorID == 123) {
                            SensorTag[k] = 'SENSOR_TST';
                            SensorName[k] = 'TimeStamp (Unix/Epoch)';
                            Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                            TimeStamp = Value.readUInt32LE(0);
                            SensorValue[k] = TimeStamp;
                            SensorUnit[k] = 'Epoch Time';
                            console.log('## ' + SensorName[k]);
                            console.log(SensorTag[k] + ' : ' + SensorValue[k].toString() +
                                ' ' + SensorUnit[k]);
                            redisArgs[SensorName[k]] = SensorValue[k];
                            k++;
                            ind = ind + 4;
                            writeRedisInd = true;
                        } else if (sensorID === 0) {
                            SensorTag[k] = 'SENSOR_CO';
                            SensorName[k] = 'Carbon Monoxide';
                            Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                            SensorValue[k] = Value.readFloatLE(0);
                            SensorUnit[k] = 'ppm';
                            console.log('## ' + SensorName[k]);
                            console.log(SensorTag[k] +
                                ' : ' + SensorValue[k] + ' ' +
                                SensorUnit[k]);
                            redisArgs[SensorName[k]] = SensorValue[k];
                            k++;
                            ind = ind + 4;
                            writeRedisInd = true;
                        } else if (sensorID == 1) {
                            SensorTag[k] = 'SENSOR_CO2';
                            SensorName[k] =
                                'Carbon Dioxide';
                            Value = new Buffer(nowarray
                                .slice(ind + 1, ind + 5));
                            SensorValue[k] = Value.readFloatLE(
                                0);

                            SensorUnit[k] = '%';
                            console.log('## ' + SensorName[k]);
                            console.log(SensorTag[k] +
                                ' : ' + SensorValue[k] + ' ' +
                                SensorUnit[k]);
                            redisArgs[SensorName[k]] = SensorValue[k];
                            k++;
                            ind = ind + 4;
                            writeRedisInd = true;
                        } else if (sensorID == 2) {
                            SensorTag[k] = 'SENSOR_O2';
                            SensorName[k] = 'Oxygen';
                            Value = new Buffer(nowarray
```

```
                    .slice(ind + 1, ind + 5));
                SensorValue[k] = Value.readFloatLE(
                    0) * 10000;

                SensorUnit[k] = 'ppm';
                console.log('## ' + SensorName[k]);
                console.log(SensorTag[k] +
                    ' : ' + SensorValue[k] + ' ' +
                    SensorUnit[k]);
                redisArgs[SensorName[k]] = SensorValue[k];
                k++;
                ind = ind + 4;
                writeRedisInd = true;
            } else if (sensorID == 11) {
                SensorTag[k] = 'SENSOR_VOC';
                SensorName[k] = 'Hydrocarbons';
                Value = new Buffer(nowarray
                    .slice(ind + 1, ind + 5));

                SensorValue[k] = Value.readFloatLE(
                    0);

                SensorUnit[k] = 'volt';
                console.log('## ' + SensorName[k]);
                console.log(SensorTag[k] +
                    ' : ' + SensorValue[k] + ' ' +
                    SensorUnit[k]);
                redisArgs[SensorName[k]] = SensorValue[k];
                k++;
                ind = ind + 4;
                writeRedisInd = true;
            } else if (sensorID == 12) {
                SensorTag[k] = 'SENSOR_TCA';
                SensorName[k] = 'Temperature Celcius';
                Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                SensorValue[k] = Value.readFloatLE(0);

                SensorUnit[k] = 'degree';
                console.log('## ' + SensorName[k]);
                console.log(SensorTag[k] +
                    ' : ' + SensorValue[k] + ' ' +
                    SensorUnit[k]);
                redisArgs[SensorName[k]] = SensorValue[k];
                k++;
                ind = ind + 4;
                writeRedisInd = true;
            } else if (sensorID == 129) {
                SensorTag[k] = 'SENSOR_GP_CO';
                SensorName[k] = 'Carbon Monoxide';
                Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                SensorValue[k] = Value.readFloatLE(0);

                SensorUnit[k] = 'ppm';
                console.log('## ' + SensorName[k]);
                console.log(SensorTag[k] +
                    ' : ' + SensorValue[k] + ' ' +
                    SensorUnit[k]);
                redisArgs[SensorName[k]] = SensorValue[k];
                k++;
                ind = ind + 4;
                writeRedisInd = true;
            } else if (sensorID == 134) {
                SensorTag[k] = 'SENSOR_GP_HCN';
                SensorName[k] = 'Hydrogen Cyanide';
                Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                SensorValue[k] = Value.readFloatLE(0);

                SensorUnit[k] = 'ppm';
```

```
                    console.log('## ' + SensorName[k]);
                    console.log(SensorTag[k] +
                        ' : ' + SensorValue[k] + ' ' +
                        SensorUnit[k]);
                    redisArgs[SensorName[k]] = SensorValue[k];
                    k++;
                    ind = ind + 4;
                    writeRedisInd = true;
                } else if (sensorID == 137) {
                    SensorTag[k] = 'SENSOR_GP_NO2';
                    SensorName[k] = 'Nitrogen Dioxide';
                    Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                    SensorValue[k] = Value.readFloatLE(0);

                    SensorUnit[k] = 'ppm';
                    console.log('## ' + SensorName[k]);
                    console.log(SensorTag[k] +
                        ' : ' + SensorValue[k] + ' ' +
                        SensorUnit[k]);
                    redisArgs[SensorName[k]] = SensorValue[k];
                    k++;
                    ind = ind + 4;
                    writeRedisInd = true;
                } else if (sensorID == 138) {
                    SensorTag[k] = 'SENSOR_GP_O2';
                    SensorName[k] = 'Oxygen';
                    Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                    SensorValue[k] = Value.readFloatLE(0);

                    SensorUnit[k] = 'ppm';
                    console.log('## ' + SensorName[k]);
                    console.log(SensorTag[k] +
                        ' : ' + SensorValue[k] + ' ' +
                        SensorUnit[k]);
                    redisArgs[SensorName[k]] = SensorValue[k];
                    k++;
                    ind = ind + 4;
                    writeRedisInd = true;
                } else if (sensorID == 144) {
                    SensorTag[k] = 'SENSOR_GP_TC';
                    SensorName[k] = 'Temperature Celcius';
                    Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                    SensorValue[k] = Value.readFloatLE(0);

                    SensorUnit[k] = 'ºC';
                    console.log('## ' + SensorName[k]);
                    console.log(SensorTag[k] +
                        ' : ' + SensorValue[k] + ' ' +
                        SensorUnit[k]);
                    redisArgs[SensorName[k]] = SensorValue[k];
                    k++;
                    ind = ind + 4;
                    writeRedisInd = true;
                } else if (sensorID == 146) {
                    SensorTag[k] = 'SENSOR_GP_HUM';
                    SensorName[k] = 'Humidity';
                    Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
                    SensorValue[k] = Value.readFloatLE(0);

                    SensorUnit[k] = '%RH';
                    console.log('## ' + SensorName[k]);
                    console.log(SensorTag[k] +
                        ' : ' + SensorValue[k] + ' ' +
                        SensorUnit[k]);
                    redisArgs[SensorName[k]] = SensorValue[k];
                    k++;
                    ind = ind + 4;
                    writeRedisInd = true;
```

```
        } else if (sensorID == 147) {
            SensorTag[k] = 'SENSOR_GP_PRES';
            SensorName[k] = 'Pressure';
            Value = new Buffer(nowarray.slice(ind + 1, ind + 5));
            SensorValue[k] = Value.readFloatLE(0);

            SensorUnit[k] = 'Pa';
            console.log('## ' + SensorName[k]);
            console.log(SensorTag[k] +
                ' : ' + SensorValue[k] + ' ' +
                SensorUnit[k]);
            redisArgs[SensorName[k]] = SensorValue[k].toString();
            k++;
            ind = ind + 4;
            writeRedisInd = true;
        } else if (sensorID == 52) {
            SensorTag[k] = 'SENSOR_BAT';
            SensorName[k] = 'Battery';
            Value = new Buffer(nowarray.slice(ind + 1, ind + 2));
            SensorValue[k] = Value.readInt8(0);
            SensorUnit[k] = '%';
            console.log('## ' + SensorName[k]);
            console.log(SensorTag[k] +
                ' : ' + SensorValue[k].toString() +
                ' ' + SensorUnit[k]);
            redisArgs[SensorName[k]] = SensorValue[k];
            k++;
            ind = ind + 1;
            writeRedisInd = true;
        } else if (sensorID == 60) {
            SensorTag[k] = 'SENSOR_GMT';
            SensorName[k] = 'TimeSyncID';
            Value = new Buffer(nowarray.slice(ind + 1, ind + 2));
            console.log('Time Synchronization ID : ' + Value.readInt8(0));
            var date = new Date().YYYYMMDDHHMMSS();
            var SOH = new Buffer("01", "hex");
            var CRLF = new Buffer("0D0A", "hex");
            var sMessage = "DATA#" + Value.readInt8(0) + "#ASCII#" + date;
            var message = new Buffer(sMessage);
            var CRCString = crc.crc16modbus(message).toString(16);
            var CRC = new Buffer(CRCString.toUpperCase());
            var EOT = new Buffer("04", "hex");
            var bMessage = Buffer.concat([SOH, message, CRLF, CRC, EOT]);
            setTimeout(function() {
                serialPort.write(bMessage);
                console.log('Time Sync message sent to ID' + Value.readInt8(0));
            }, 1000);
            redisClient.set('timesync:' + WaspmoteID.toString('ascii'),sMessage);
            // redisClientLocal.set('timesync:' + WaspmoteID.toString('ascii'),sMessage);
            k++;
            ind = ind + 1;
            writeRedisInd = false;
        }
        ind++;

    }
    if (writeRedisInd) {
        var output = new DataStruct(SerialID.readUInt32LE(0, 4),
            WaspmoteID.toString('ascii'),
            NumberOfBytes,
            FrameSequence,
            TimeStamp,
            SensorName,
            SensorTag,
            SensorValue,
            SensorUnit);
```

```
                        wstream.write(data);
                        var dataset = output;
                        if (dataset) {
                            var args = ['sensor:' + dataset.WaspmoteID, dataset.TimeStamp.toString(),
JSON.stringify(redisArgs)];
                            console.log('sensor:' + dataset.WaspmoteID, dataset.TimeStamp, dataset.TimeStamp);
                            var isScore = redisClient.zcount('sensor:' + dataset.WaspmoteID, dataset.TimeStamp,
dataset.TimeStamp);
                            console.log(isScore);

                            redisClient.zadd(args, function(err, response) {
                                if (err) {
                                    console.log(err);
                                }
                            });
                            // redisClientLocal.zadd(args, function(err, response) {
                            //     if (err) {
                            //         console.log(err);
                            //     }
                            //     args = [];
                            //     redisArgs = {};
                            // });


                            //     console.log(dataset.SerialID);
                            //     redisClient.hmset(dataset.WaspmoteID, dataset, redis.print);
                        }
                    }
                }


            } else {
                console.log('data.length : ' +
                    data.length);
                console.log('qdata.length : ' +
                    qdata.length);
            }


        });

    });
```

[웹서버 가동 및 센서데이터 쿼리 API]

```
app.all('/parseddata', function(request, response) {
    response.send(output[output.length - 1]);
});

app.all('/keys', function(request, response) {
    redisClient.keys('sensor:*', function(err, keys) {
        keys.sort();
        var sensorUpdateInfo = [];
        var i = 0;
        keys.forEach(function(val, I) {
            redisClient.zrange(keys[I], -1, -1, function(err, result) {
                i++;
                if (err) { console.log(err) } else {
                    var parsedJSON = JSON.parse(result);
                    var sensorTimes = parsedJSON["Time Stamp Server"];
                    var temp_data = { 'sensorNode': keys[I], 'updatedTime': sensorTimes };
                    sensorUpdateInfo.push(temp_data);
                }

                if (i == keys.length) {
                    response.send(sensorUpdateInfo);
                }
```

```
            });
        });
    });
});

app.all('/data', function(request, response) {
    var sensorNode = request.param('sensorNode');
    if (request.param('nStart')) {
        var startDataId = request.param('nStart');

    } else {
        var startDataId = 0 }
    if (request.param('nStop')) {
        var stopDataId = request.param('nStop');

    } else {
        var stopDataId = -1 }
    redisClient.zrange('sensor:' + sensorNode, startDataId, stopDataId, function(err, result) { // ''withscores''
        var parsedJSON = [];
        for (var i = 0; i < result.length; i++) {
            parsedJSON[i] = JSON.parse(result[i]);
        }
        response.send(parsedJSON);
    });
});

app.all('/databydate', function(request, response) {
    var sensorNode = request.param('sensorNode');
    var dateStart = request.param('dateStart');
    var dateStop = request.param('dateStop');
    var sensorName = request.param('sensorName');
    var sensorData = [];
    redisClient.zrangebyscore('sensor:' + sensorNode, dateStart, dateStop, function(err, result) {
        if (err) throw err;
        var parsedJSON = [];
        for (var i = 0; i < result.length; i++) {
            parsedJSON[i] = JSON.parse(result[i]);
            var timeStamp = new Date(parsedJSON[i]["Time Stamp Server"]);
            sensorData[i] = { 'time': timeStamp };
            sensorData[i][sensorName] = parsedJSON[i][sensorName];
        }
        var csv = converter.json2csv(sensorData, function(err, csv) {
            if (err) throw err;
            response.send(csv);
        });
    });
});

app.all('/dbname', function(request, response) {
redisClient.get("server", function (err, reply) {
    response.send({'server': 'http://' + reply});
});
});

app.get('/port', function(request, response) {
    serialPort.list(function(err, ports) {
        if (err) {
            console.log(err);
        } else {
            response.send(ports);
        }
    });
});

app.all('/chart', function(request, response) {
    var id_wasp = request.param('id_wasp');
    var sensor = request.param('sensor');
```

```
    var latesttime = '';
    for (var i in uptodate) {
        if (id_wasp == uptodate[i].id_wasp) {
            console.log(uptodate[i].timestamp);
            var LT = new Date(uptodate[i].timestamp);
            latesttime = "'" + LT.getFullYear() +
                "-" + zpad(LT.getMonth() + 1) +
                "-" + zpad(LT.getDate()) +
                "T" + zpad(LT.getHours()) +
                ":" + zpad(LT.getMinutes()) +
                ":" + zpad(LT.getSeconds()) +
                "." + (LT.getMilliseconds() / 1000).toFixed(3).slice(2, 5) +
                "Z'";
            console.log(latesttime);
            break;
        } else {
            latesttime = "(SELECT timestamp from sensorParser WHERE id_wasp='" + id_wasp + "' and sensor='" + sensor + "'
ORDER BY timestamp DESC limit 1)";
        }

    }
    var querystring = "SELECT timestamp, value, frame_number FROM sensorParser WHERE id_wasp='" + id_wasp + "' AND
sensor='" + sensor + "' AND timestamp > (SELECT DATE_ADD(" + latesttime + ", INTERVAL -1 day));";
    console.log(querystring);
    client.query(querystring, function(error, data) {
        if (error) {
            console.log(error);
        } else {
            response.send(data);
        }
    });
});

app.all('/sensorinfo', function(request, response) {
    var sensor = request.param('sensor');
    client.query('SELECT units, name FROM sensors WHERE id_ascii=?', [sensor], function(error, data) {
        if (error) {
            console.log(error);
        } else {
            response.send(data);
        }
    });
});

app.get('/bynode', function(request, response) {
    client.query('SELECT id_wasp FROM sensorParser GROUP BY id_wasp', function(error, data) {
        if (error) {
            console.log(error);
        } else {
            response.send(data);
        }
    });
});

app.get('/bysensor', function(request, response) {
    client.query('SELECT sensor FROM sensorParser GROUP BY sensor', function(error, data) {
        if (error) {
            console.log(error);
        } else {
            response.send(data);
        }

    });
});

var listOfNode = [];
var uptodate = [];
app.get('/uptodate', function(request, response) {
```

```
    client.query('SELECT id_wasp FROM sensorParser GROUP BY id_wasp', function(error, data) {
        if (error) {
            console.log(error);
        } else {
            // console.log(data);
            var queryscript = [];
            uptodate = [];
            for (var i in data) {
                queryscript = "SELECT id_wasp, timestamp FROM sensorParser where id_wasp='" + data[i].id_wasp + "' ORDER BY
timestamp DESC LIMIT 1;";
                client.query(queryscript, function(error, result) {
                    uptodate.push(result[0]);
                });
            };
        }
    });
    response.send(uptodate);
});

/*
app.get('/chart/:id_wasp/:sensor', function(request, response) {
    var id_wasp = request.param('id_wasp');
    var sensor = request.param('sensor');
    client.query('SELECT timestamp, value, frame_number FROM sensorParser WHERE id_wasp=? AND sensor=?', [
        id_wasp, sensor
    ], function(error, data) {
        if (error) {
            console.log(error);
        } else {
            response.send(data);
        }
    });
});
*/
app.set('port', process.env.PORT || 8000);

var server = app.listen(app.get('port'), function() {
    console.log('Express server listening on port ' + server.address().port);
});
```

[자동실행 스크립트 (Debian Linux]

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:             shtech
# Required-Start:     $local_fs $network
# Required-Stop:       $local_fs
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description: m2mserver
# Description:         s.h. Tech & policy
### END INIT INFO
export PATH=$PATH:/usr/bin
export NODE_PATH=$NODE_PATH:/usr/lib/node_modules

case "$1" in
    start)
        echo "Starting M2M Server"
        (cd /home/shtpi/m2m-server-main/ && sudo forever start m2mserver.js);; #실행되는 위치 변경
    stop)
        echo "Stopping M2M Server"
        (sudo forever stop /home/shtpi/m2m-server-main/m2mserver.js);; #실행되는 위치 변경
    *)
        echo "Usage: /etc/init.d/m2mserver {start|stop}"
        exit 1
```

```
        ;;
esac

exit 0
```

1) 센서 데이터 쿼리 API

본 연구에서는 빠른 응답이 중요하기 때문에 무선센서노드의 모든 데이터는 최초 Redis에 저장이 되며, 오래된 데이터는 mongodb로 이전하고 이전한 Redis의 데이터는 삭제된다. 이는 필요한 응급 데이터는 hot data로 유지하며 데이터 마이닝을 위한 cold data를 mongodb로 이전시킨다. 데이터를 쿼리하는 방법은 다음과 같다.

① GET : Recent data

최근 1개 데이터 /data?sensorNode= + $nodeId + &nStart=-1&nStop=-1
최근 10개 데이터 /data?sensorNode= + $nodeId + &nStart=-10&nStop=-1
최초 1개 데이터 /data?sensorNode= + $nodeId + &nStart=0&nStop=0
최초 10개 데이터 /data?sensorNode= + $nodeId + &nStart=0&nStop=9

- sensorNode=$nodeId : 스트링 변수 SHTPI001에서 SHTPI016를 나타낸다.

데이터를 수집할 때 마다 현장 역무실의 Master 서버에서는 승화기술정책연구소 내부에 있는 서버로 push 한다.

```
ex) http://shtpiserver.iptime.org/data?sensorNode=SHTPI010&nStart=-1&nStop=-1 (본사외부접속의 경우)
ex) http://192.168.0.100/data?sensorNode=SHTPI010&nStart=-1&nStop=-1 (역무실의 경우)
```

② GET : Time history
센서들의 시간이력 데이터는 아래와 같이 불러온다.
리턴값은 csv(comma separated value)이다.
첫번째 열은 시간 두번째 열은 데이터.

```
databydate?sensorNode=' + $nodeId + '&dateStart=' + $startTimestamp + '&dateStop=' + $endTimestamp + '&sensorName=' + $sensorId
```

sensorNode=$nodeId : 스트링 변수 SHTPI001에서 SHTPI016를 나타낸다.
sensorName=$sensorId : 스트링 변수 "센서 이름" 여기서 센서이름은 최근 1개 데이터에서 JSON 키값을 참고한다(Object.keys(JSON)).
dateStart=$startTimestamp : 스트링 변수, 불러올 데이터의 시작 타임스탬프이다. 최근 1개의 데이터에서 JSON키 "Time Stamp Server"값에서 86400000값을 빼면 1day 전이 된다(24h x 60m x 60s x 1000ms)
dateStop=$endTimestamp : 스트링 변수, 불러올 데이터의 마지막 타임스탬프이다. 위와 같은 방법으로 불러온다. $startTimestamp는 $startTimestamp보다 작아야 함

```
ex)
http://shtpiserver.iptime.org/databydate?sensorNode=SHTPI001&dateStart=1464173160316&dateStop=1464259560316&sensorName="Oxygen" (회사 외부접속의 경우)
 ex)
http://192.168.1.18/databydate?sensorNode=SHTPI001&dateStart=1464173160316&dateStop=1464259560316&sensorName="Carbon
```

Monoxide" (역무실의 경우)

nkeeper001 : 192.168.0.114 nkeeper010 : 192.168.0.111 nkeeper012 : 192.168.0.98 nkeeper015 : 192.168.0.81

2) 센서노드 시각동기화

기존 1차년도 테스트베드 센서의 고도화를 위하여 Digimesh 용 센서노드의 시각동기화 (Time Synchronize) 작업을 아래 그림과 같이 수행하였다.

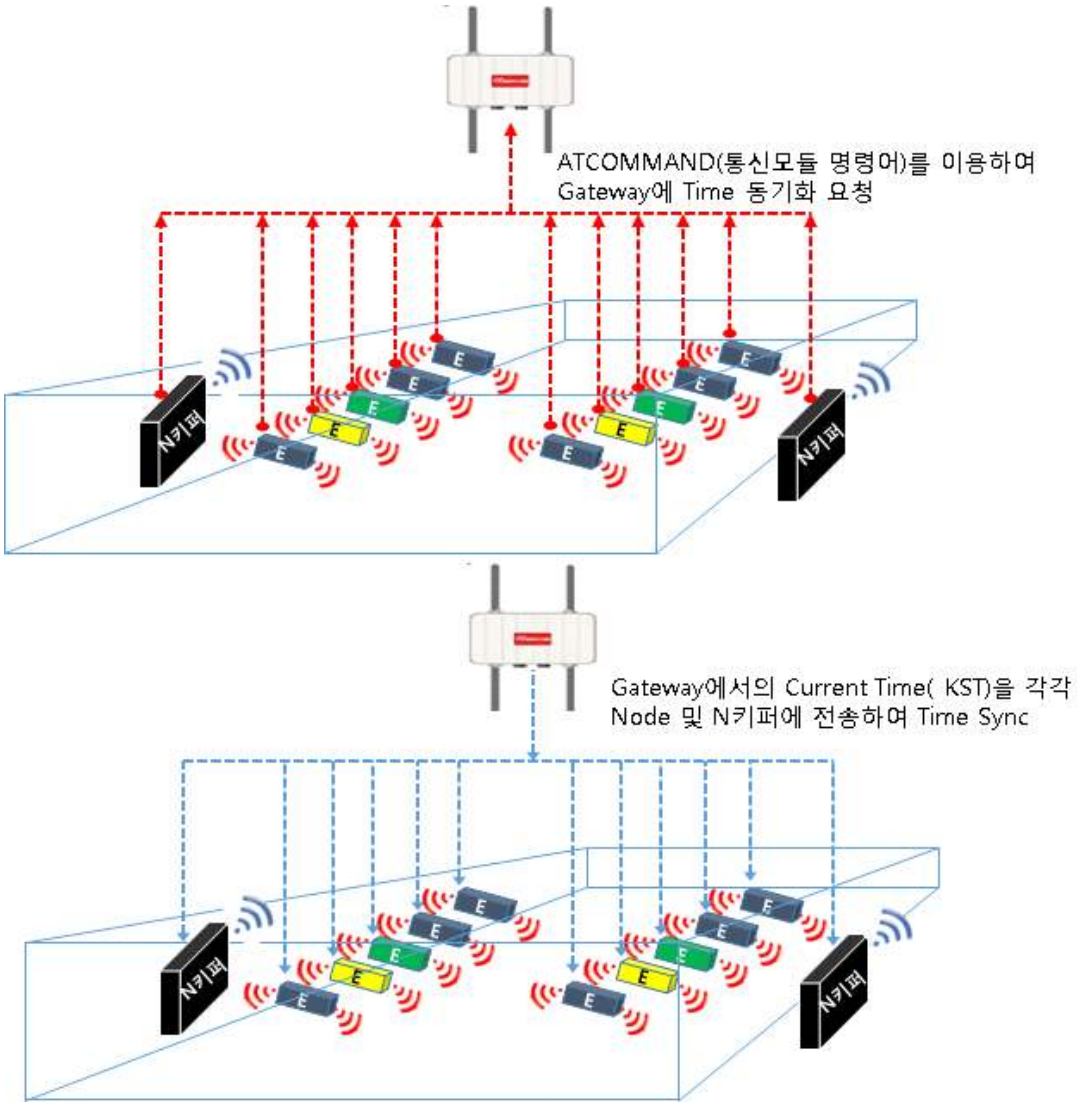


그림. 센서노드 시각동기화 구성도

- 센서보드가 갖고있는 RTC(Real Time Clock)기능을 이용하여 실제 KST시간으로 센서노드의 시각동기화를 통해 데이터가 전송된 시간이 아닌 측정한 시간의 데이터를 수집하여 데이터의 신뢰성을 높이고, 재난 혹은 센서 값들의 위치에 따른 위상결과를 확인이 가능하다.

```
uint8_t WaspXBeeCore::setRTCfromMeshlium(char* address, char * myAddress)
{
    uint16_t length = 0;
    bool status = false;
    char buffer[100];
    char timestamp[30];
    int year, yearH;
    int month;
    int date;
    int hour;
    int minute;
    int second;
    packetXBee packet;
    int retries;

    uint8_t request[45];
    char myMACAddressHex[16];

    // define the correct response length
    //(19B for something like "20150203144622+0000")
    uint16_t PATTERN_LENGTH = 19;

    // clear buffers
    memset( buffer, 0x00, sizeof(buffer) );
    memset( timestamp, 0x00, sizeof(timestamp) );
    memset( &packet, 0x00, sizeof(packet) );

    memset( myMACAddressHex, 0x00, sizeof(myMACAddressHex));
    memset( request, 0x00, sizeof(request) );
    request[0] = 0x3C; request[1] = 0x3D; request[2] = 0x3E; request[3] = 0x9B; request[4] = 0x01; request[5] = 0x23;
    request[6] = 0x33; request[7] = 0x38; request[8] = 0x37; request[9] = 0x32; request[10] = 0x36; request[11] = 0x34;
    request[12] = 0x35; request[13] = 0x33; request[14] = 0x39; request[15] = 0x23;
    request[16] = 0x6E; request[17] = 0x6F; request[18] = 0x64; request[19] = 0x65; request[20] = 0x31; request[21] = 0x23;
    request[22] = 0x30; request[23] = 0x23;
    request[24] = 0x4D; request[25] = 0x41; request[26] = 0x43; request[27] = 0x3A;

    for(int i=0, j=0; i<16; i++, j++)
    {
        sprintf(&myMACAddressHex[i], "%X", myAddress[j] & 0xf0);
        i++;
        sprintf(&myMACAddressHex[i], "%X", myAddress[j] & 0x0f);
    }
    for(int i=28, j=0; i<44; i++, j++)
    {
        request[i] = myMACAddressHex[j];
    }
    request[44] = 0x23;

    /*
    uint8_t request[] = {   0x3C, 0x3D, 0x3E, 0x9B, 0x01, 0x23,
                            0x33, 0x38, 0x37, 0x32, 0x36, 0x34,
                            0x35, 0x33, 0x39, 0x23,
                            0x6E, 0x6F, 0x64, 0x65, 0x31, 0x23,
                            0x30, 0x23,
                            0x4D, 0x41, 0x43, 0x3A,
                            0x30, 0x30, 0x31, 0x33, 0x41, 0x32, 0x30, 0x30,
                            0x34, 0x30, 0x43, 0x35, 0x35, 0x36, 0x35, 0x44,
                            0x23
                        };
```

3) 통신모듈 β버젼 제작

○ 스마트N키퍼와 각 모듈간의 통신 사이에 DB저장 및 게이트웨이 구성.

○ 스마트N키퍼가 데이터분석 및 피난 유도 알고리즘을 분석함과 동시에 DB를 구축하고 있지만 게이트웨이 DB모듈을 중간 단계로 두어 안정성을 도모

○ 스마트N키퍼간의 통신을 위해 각 스마트N키퍼에 AP를 구성

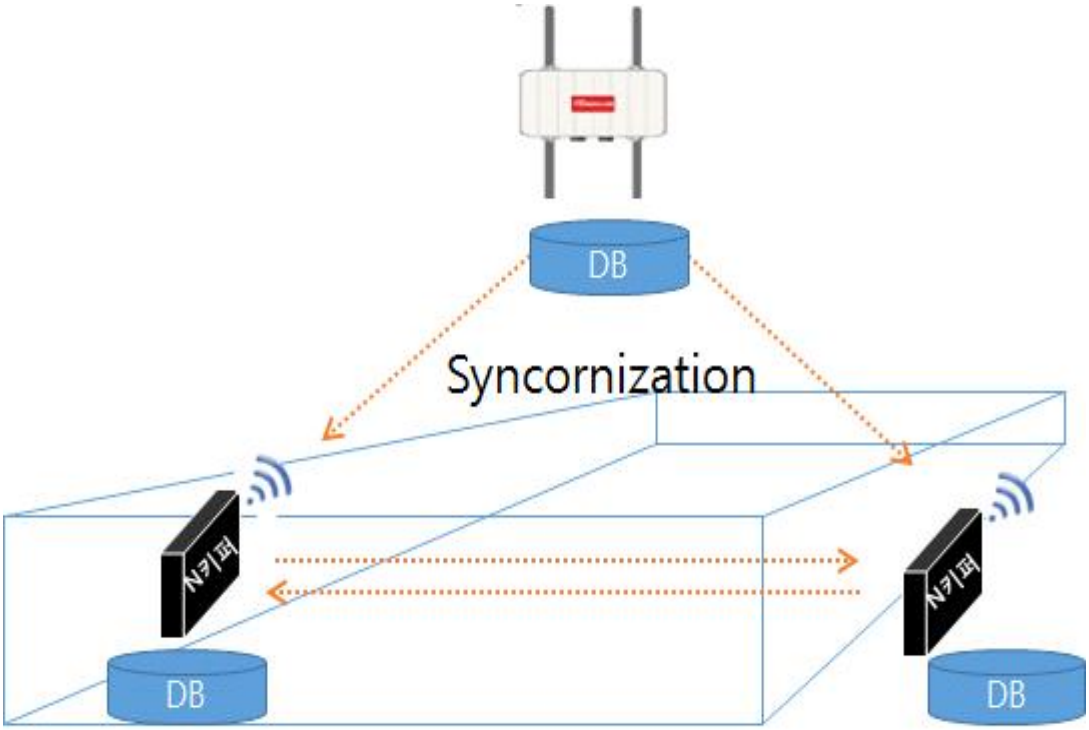


그림. 통신모듈 β버젼 구성도

마스터 측에서 시각동기화가 작동하는 루틴은 센서노드에서 수신된 패킷을 통해 해당 센서노드가 시각동기화를 위한 수신 대기임을 인식하고, 마스터의 시간을 센서노드를 향해 command 신호로 보내게 된다. LoRa SX1272의 경우 하프듀플렉스로 작동하기 때문에 센서노드는 정기적으로 Timeout 을 걸어주며 마스터에 시각데이터를 요청한다. 여기서 command 신호의 유효성 검증을 위해 CRC modbus 알고리즘을 사용한다.

[마스터의 시각동기화 코드]

```
SensorTag[k] = 'SENSOR_GMT';
SensorName[k] = 'TimeSyncID';
Value = new Buffer(nowarray.slice(ind + 1, ind + 2));
console.log('Time Synchronization ID : ' + Value.readInt8(0));
var date = new Date().YYYYMMDDHHMMSS();
var SOH = new Buffer("01", "hex");
var CRLF = new Buffer("0D0A", "hex");
var sMessage = "DATA#" + Value.readInt8(0) + "#ASCII#" + date;
var message = new Buffer(sMessage);
var CRCString = crc.crc16modbus(message).toString(16);
var CRC = new Buffer(CRCString.toUpperCase());
var EOT = new Buffer("04", "hex");
var bMessage = Buffer.concat([SOH, message, CRLF, CRC, EOT]);
setTimeout(function() {
    serialPort.write(bMessage);
    console.log('Time Sync message sent to ID' + Value.readInt8(0));
}, 1000);
redisClient.set('timesync:' + WaspmoteID.toString('ascii'),sMessage);
k++;
ind = ind + 1;
writeRedisInd = false;
```

11. 재난 전조 대응 M2M 웹 어플리케이션 개발

가. 개요

현장에 설치된 안내 시스템인 N키퍼의 상태관리 및 각 키퍼 별 피난 경로 설정 기능 개발을 수행하였다.
센서 데이터 기반 재난 발생 전조 예측 룰셋 입력 및 관리 기능 개발을 수행하였다.
재난 발생 상황 전파 기능 개발을 수행하였다.

나. 성능목표

기존기술 수준

- 기존 관리 시스템은 현장에 설치된 센서의 기기적 상태 이상여부와 작동 여부에 대한 관리 기능 수준에서 기능이 한정되었다.

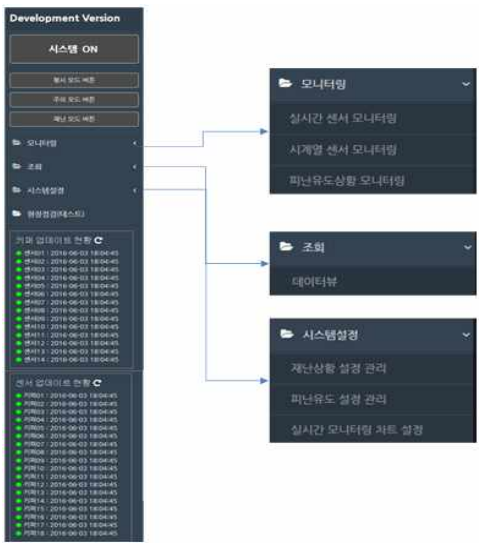
개발기술 수준

- 데이터 마이닝 기반 현장 재난 대응 시스템의 실시간 상황인지 결과 모니터링 및 가시화 기능 개발을 목표로 하였다.
- 재난전조 예측을 위한 현장 데이터 기반 재난전조 룰셋 학습기능 및 클라이언트에 전파 기능 등, IoT 대응 재난 전조 예측 기능 개발을 목표로 하였다.

다. 관리 시스템 기능

관리 시스템은 다음과 같은 기능을 수행한다.

- 실시간 센서 데이터 모니터링
- 이상/재난상황 판단
- 센서 데이터 조회
- SOP 시나리오 등록/삭제/수정
- 관리 설정



<그림 마스터 메뉴>

라. 실시간 센서 모니터링 기능 개발

관리 시스템에서는 모든 센서들의 실시간 데이터를 모니터링한다.
모든 데이터를 적절한 범위 내로 표준화한 후에 정상범위와 이상범위의 데이터를 시각적으로 쉽게 판별할 수 있도록 한다.



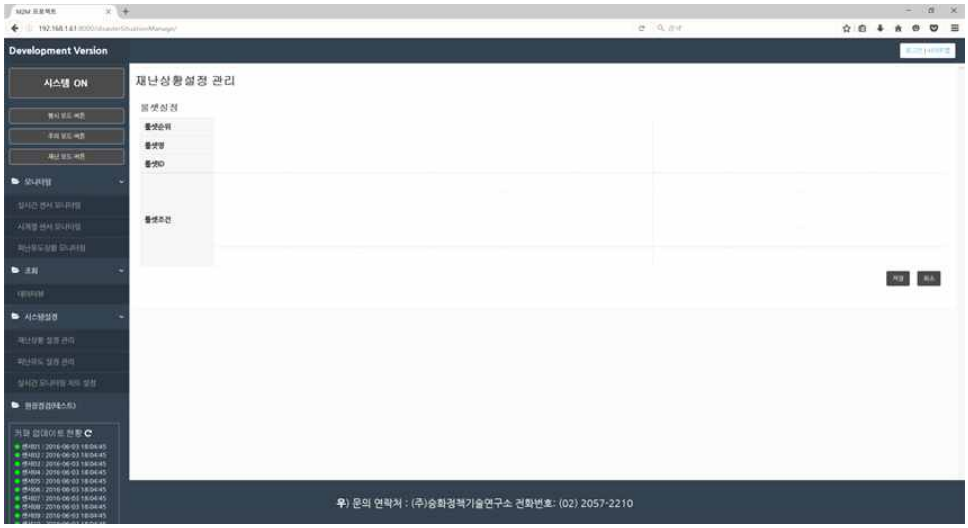
[실시간 센서 모니터링 페이지]



[실시간 모니터링 차트 설정]

마. 이상/재난상황 판단

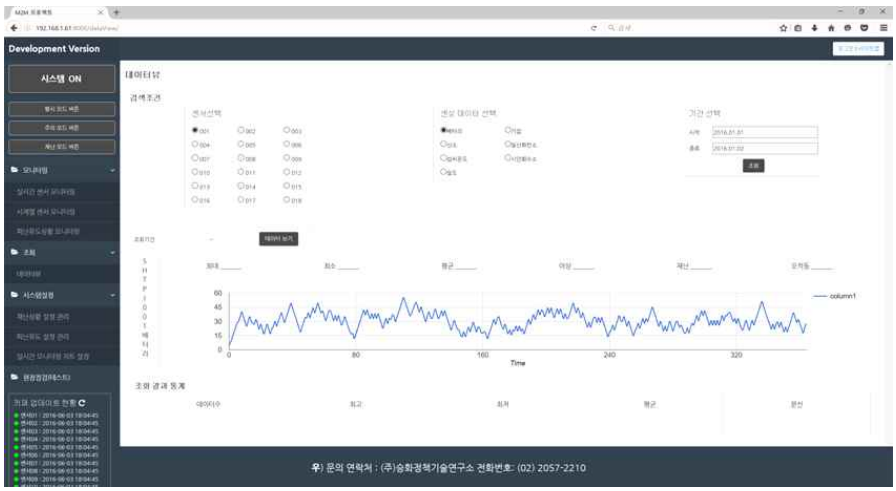
이상 및 재난상황 판단은 사용자가 등록한 룰셋 및 재난전조 예측 모델에서 도출된 룰셋에 의해서 이루어진다. 실시간 데이터가 룰셋에 해당이 되면 시스템은 관리자에게 상황을 알리고, 재난상황을 전파할 것인지에 대한 의사결정을 요구한다.



<그림 이상/재난 판단 룰셋 관리 페이지>

바. 센서 데이터 조회

관리 시스템에서는 과거의 센서 시계열 데이터에 대한 조회 기능을 제공한다.
특정 센서 및 특정 데이터에 대해서 기간을 지정하게 되면 해당 데이터가 시계열 차트로 제공된다.



<그림 시계열 데이터 조회 페이지>

사. SOP 시나리오 등록/삭제/수정

SOP 시나리오 관리 시스템에서는 지하철 SOP를 시스템에 등록하거나 삭제 수정하는 기능을 제공한다.
각 시나리오에서는 재난이 감지된 센서와 재난 종류를 입력하고 각 시나리오 별로 SOP 전략을 기록하고 각 키퍼별로 피난 유도 방향을 지정한다.



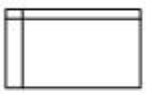
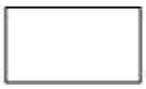
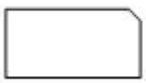
<그림 시나리오 등록/삭제/수정 페이지>

아. DFD 분석

1) 개요

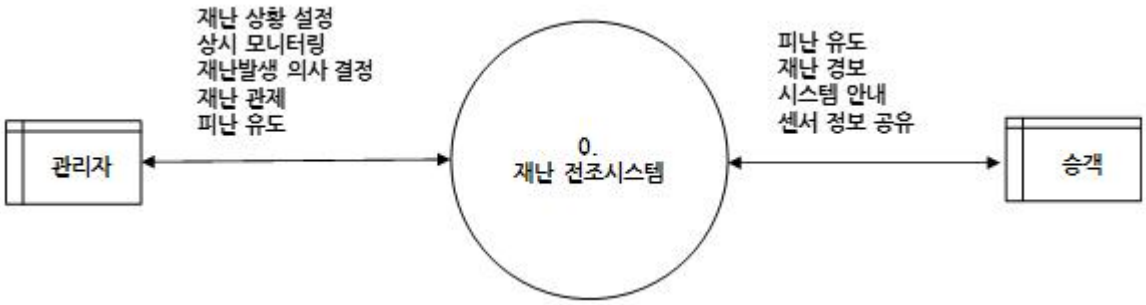
○ 3차년도 새로운 사용자 및 전문가 요구사항에 맞추어 기존 시스템을 업데이트하기 위해서 변경된 부분에 대해서 새롭게 DFD를 분석하여 시스템 추가 요청 사항을 반영함

2) 기호 설명

도형	설명
	DFD 문서 참고
	외부 시스템
	주체
	DB
	데이터 흐름

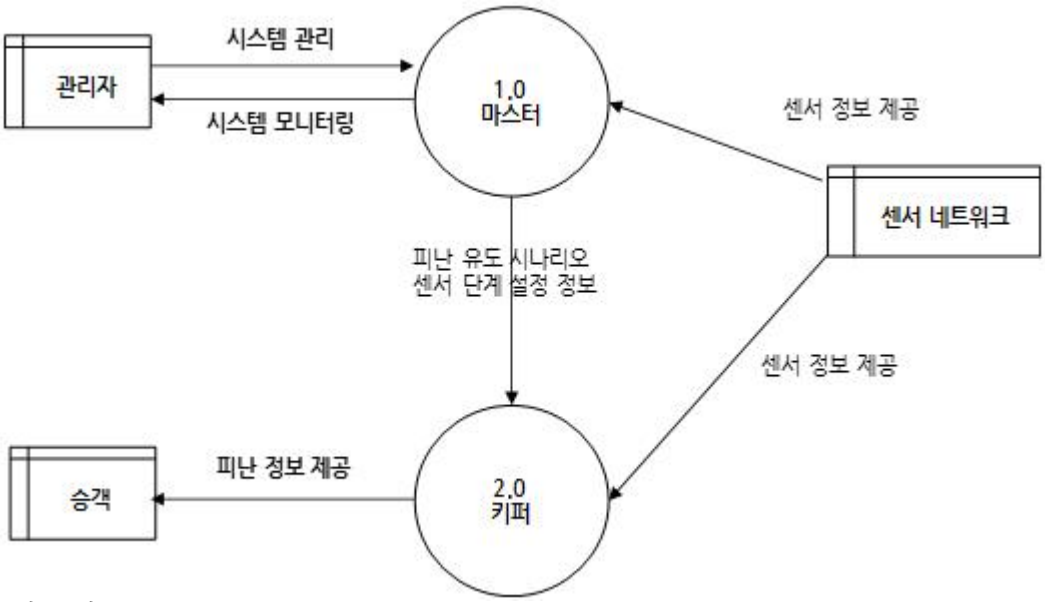
3) 0 전체 시스템

- 전체 시스템은 관리자가 재난 전조 시스템을 통해서 재난 정보를 제공하면 승객은 그 정보에 의해서 피난을 수행한다.



4) 0 재난 전조 서브 시스템

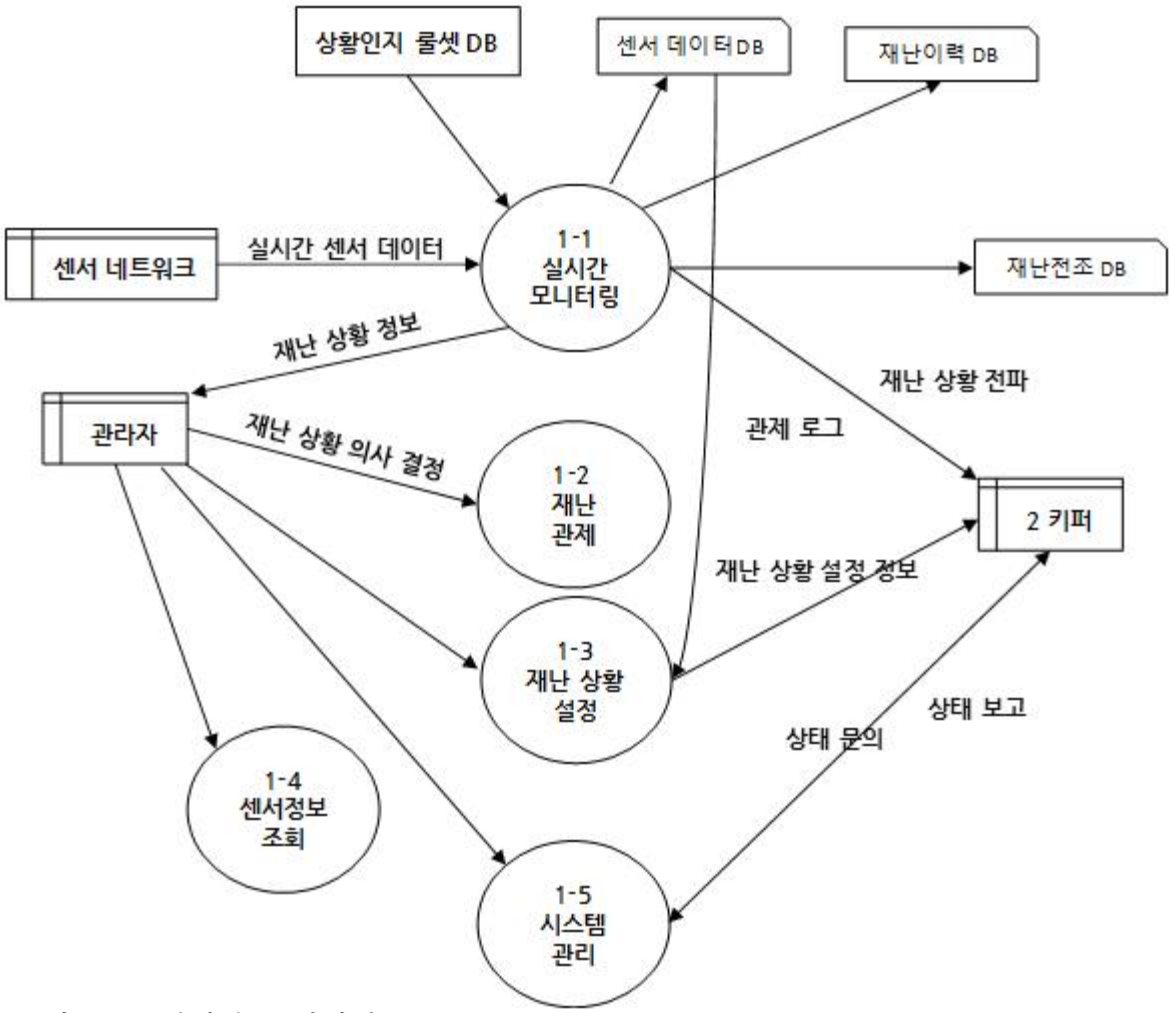
- 시스템은 센서 데이터에서 전달되는 센서정보를 실시간으로 시스템에 전달한다.
- 마스터는 전달된 정보를 실시간으로 분석해서 재난전조를 감지하여 재난 발생 시 사용자에게 재난 경보를 전달한다.
- 키퍼는 평시모드에서는 센서 데이터를 표출하며, 재난 발생 시 마스터로부터 재난 정보를 전달 받아서 재난 경보를 수행한다.



5) 1 마스터

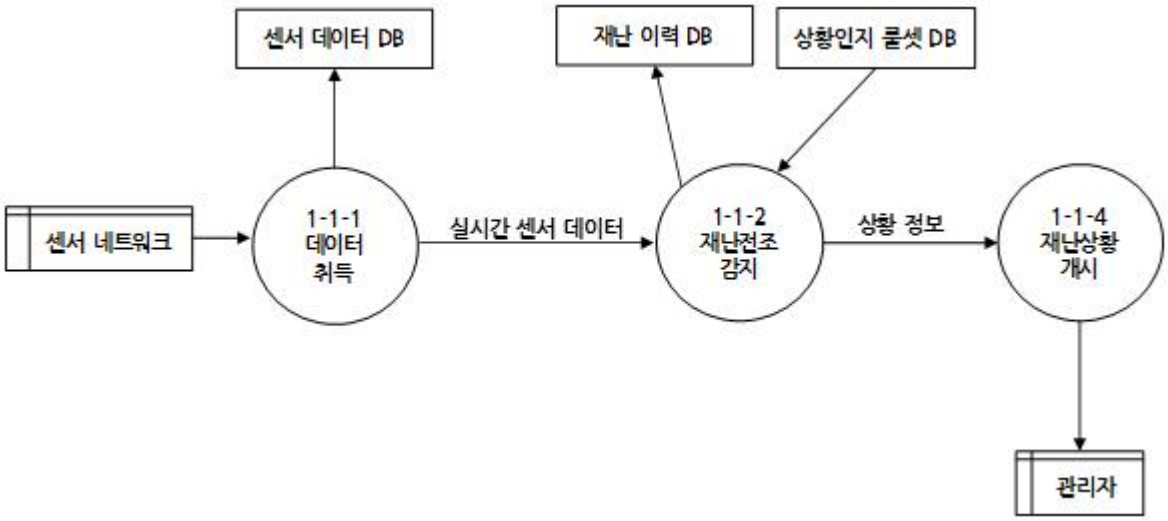
마스터의 주요 기능은 다음과 같다. 재난 조건에 대한 설정하며, 재난 발생 여부를 상시 모니터링 한다.

또한 재난 발생 시 피난 유도를 관제한다.



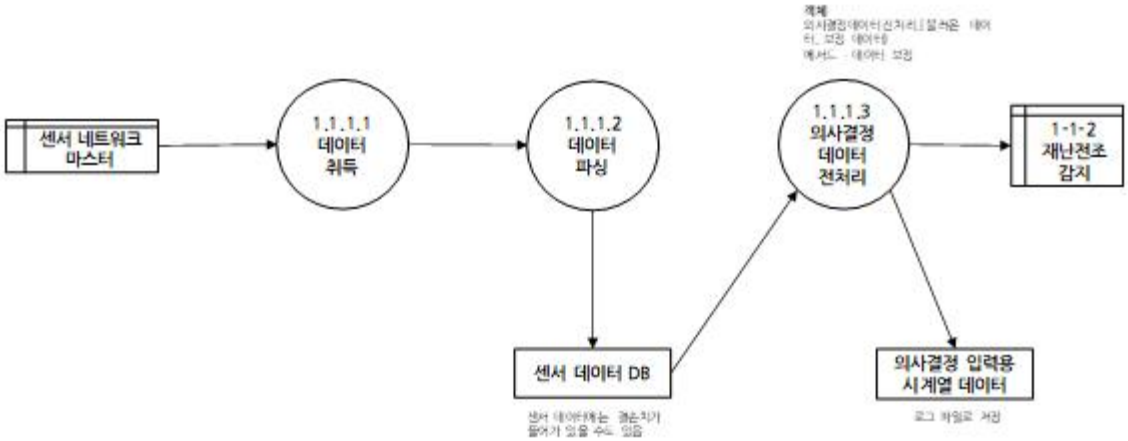
가) 1-1 실시간 모니터링

- 센서네트워크에서 전달도는 센서 정보를 실시간으로 마스터에 저장하고, 이 데이터를 재난전조 감지 룰셋과 비교해서 재난 전조를 모니터링한다.
- 센서 데이터가 재난 범위에 들어가면, 마스터는 일정 기간 스택을 감시하다가 스택이 지정된 값을 넘어서게 되면 재난상황을 개시하며, 관리자에게 카운트다운을 수행한다.



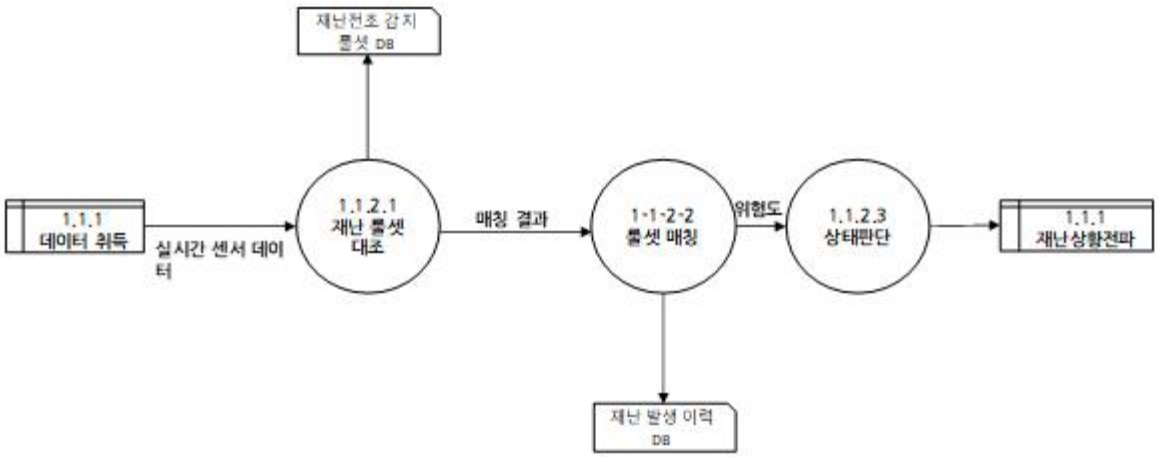
(1) 1-1-1 데이터 취득

- 센서 데이터 서버에서 마스터 및 키퍼는 센서 정보를 5초에 한번씩 전달받는다.
- 마스터에서는 이 센서 데이터를 DB에 저장하고, 키퍼는 정보 표출에만 사용한다.



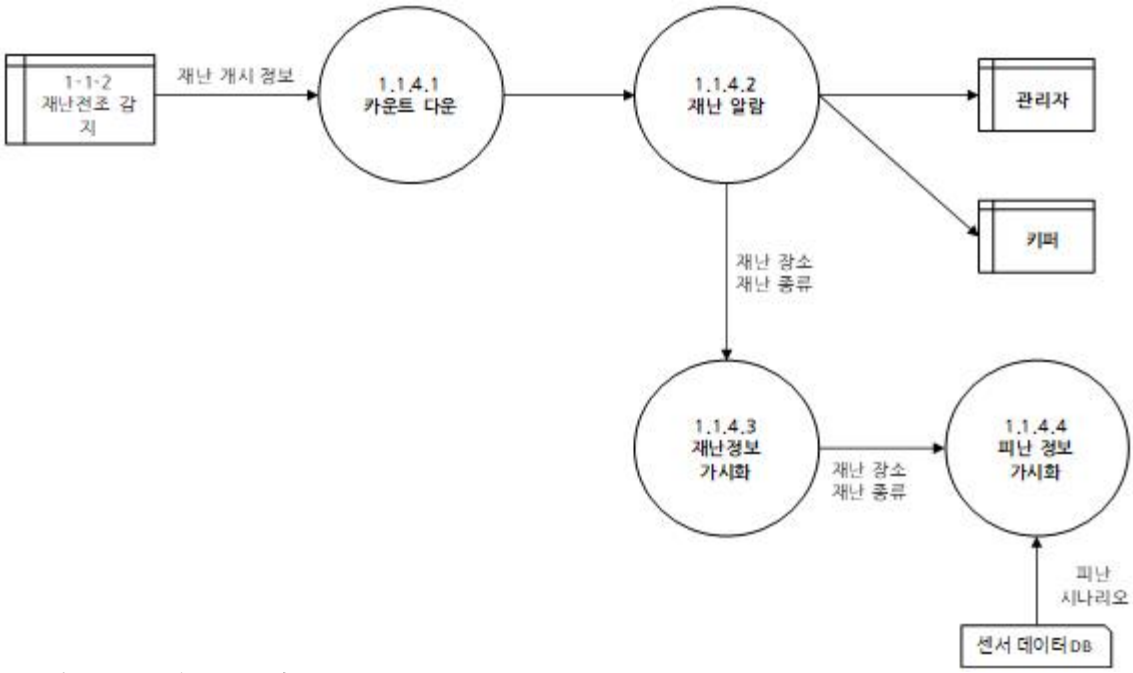
(2) 1-1-2 재난전조감지

- 센서마스터에서 전달된 센서 데이터를 실시간으로 재난 룰셋과 대조하면서 재난 전조의 발생 여부를 모니터링한다.
- 데이터가 재난 상황에 매칭이 되는 경우 한번 매칭이 될 때마다 스택이 쌓이며, 시스템에서 정한 스택을 초월하면 재난 모드를 발동시킨다.



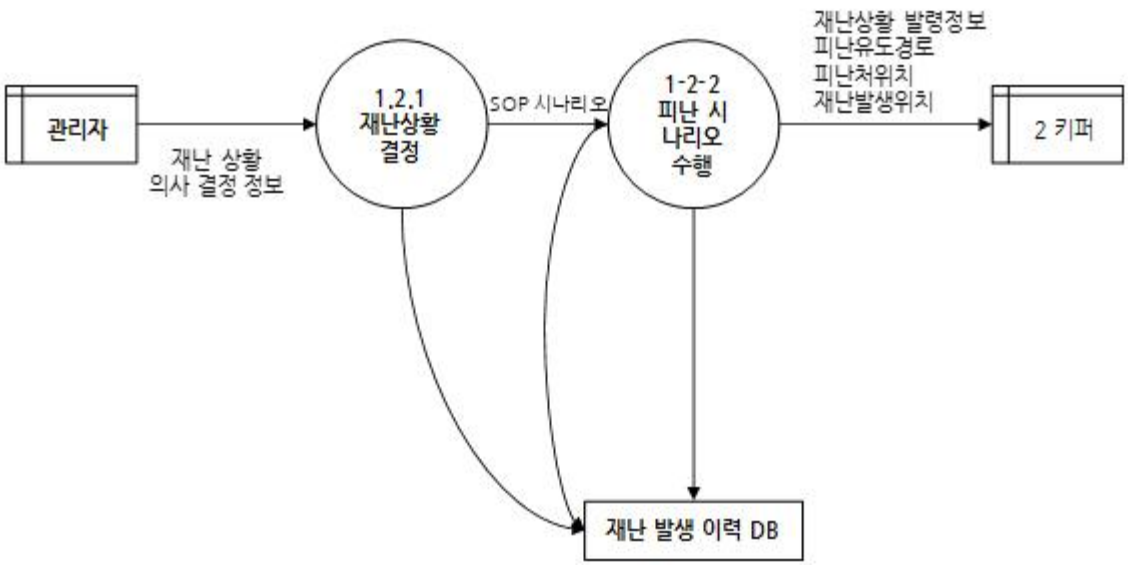
(3) 1-1-4 재난상황 개시

- 재난 전조가 감지되면 30초동안 카운트다운이 수행되고, 이때 역무원이 정지를 하지 않는 경우, 재난 모드가 발동한다.
- 재난 모드가 발동하면 재난 상황정보가 키퍼에게 전달되고, 키퍼는 재난모드로 돌입하면서, 이용객에게 피난 유도를 지시한다.



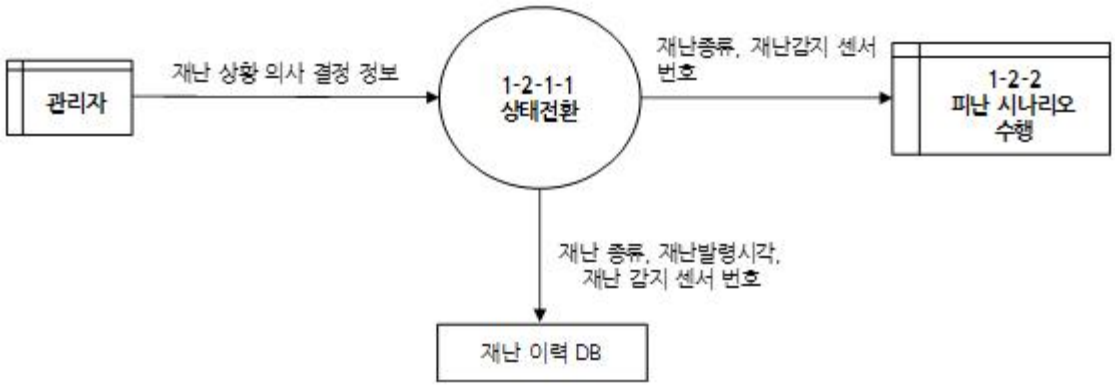
나) 1-2 재난 관제

- 재난이 발생하면 센서 번호와 재난 종류 정보에 의해서 사전에 저장된 피난 유도 시나리오에 따라서 각 키퍼 별로 피난 방향이 지시된다.
- 키퍼는 피난 방향을 화면에 표시하여 승객의 피난을 돕는다.



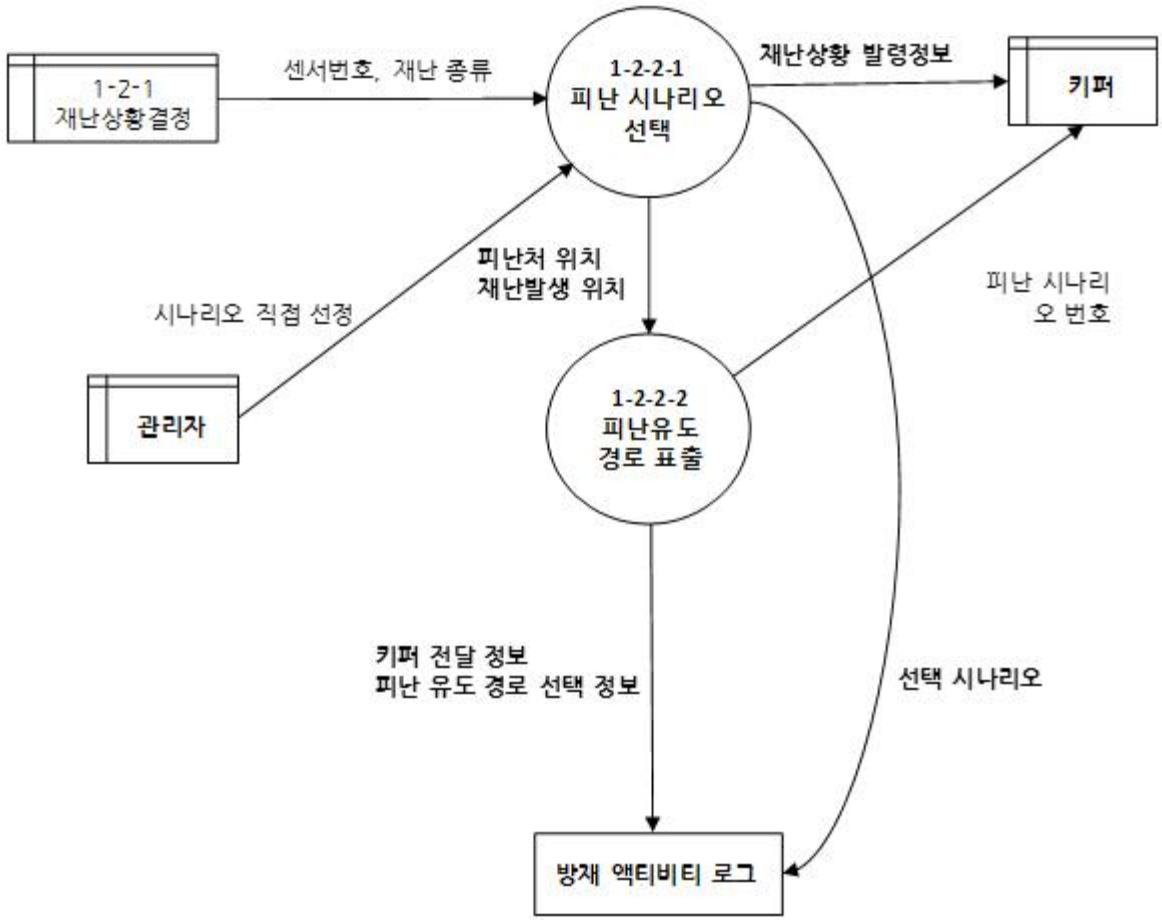
(1) 1-2-1 재난상황 결정

- 재난 상황이 결정되면 재난 종류 및 재난 발생위치 정보가 피난 시나리오 수행 모듈에 전달된다
- 재난 발생에 관련된 정보가 재난 이력 DB에 저장된다.



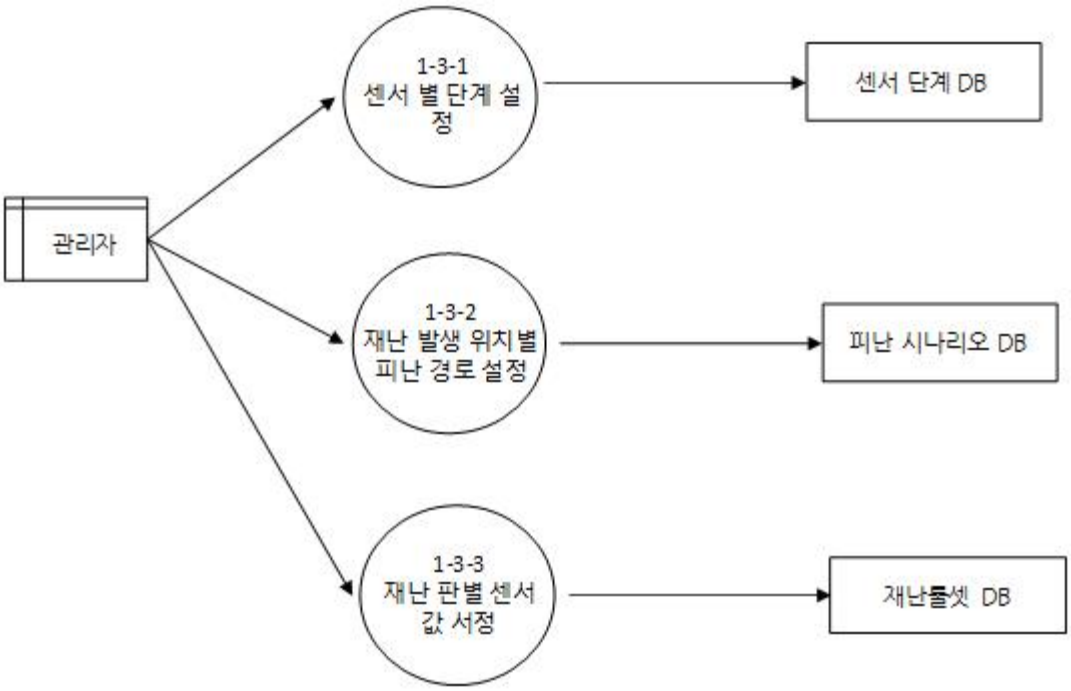
(2) 1-2-2 피난 시나리오 수행

- 마스터에서 재난 감지 센서 번호와 재난 종류 정보를 바탕으로 피난 시나리오를 선택하고 이를 키퍼에 전달한다.
- 키퍼에서는 사전에 등록된 피난 경로를 표출하여 이용객의 피난을 유도한다.



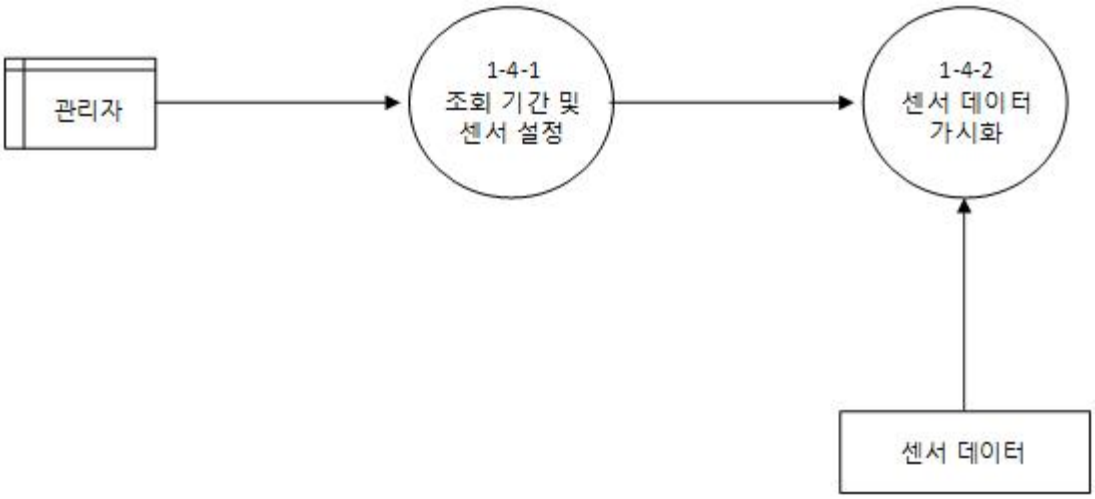
다) 1-3 재난 상황 설정

- 마스터에서는 각 센서 디바이스의 센서 종류 별로 5단계에 걸쳐서 값을 설정하여 센서 상태를표시한다.
- 관리자는 각 재난 감시 센서 별로 피난 방향을 설정하는 시나리오를 시스템에 입력한다.
- 각 센서 별 센서 데이터에 대해서 재난 범위를 설정하여, 실시간 센서 데이터에 대한 재난 전조 여부를 모니터링하는데 사용한다.



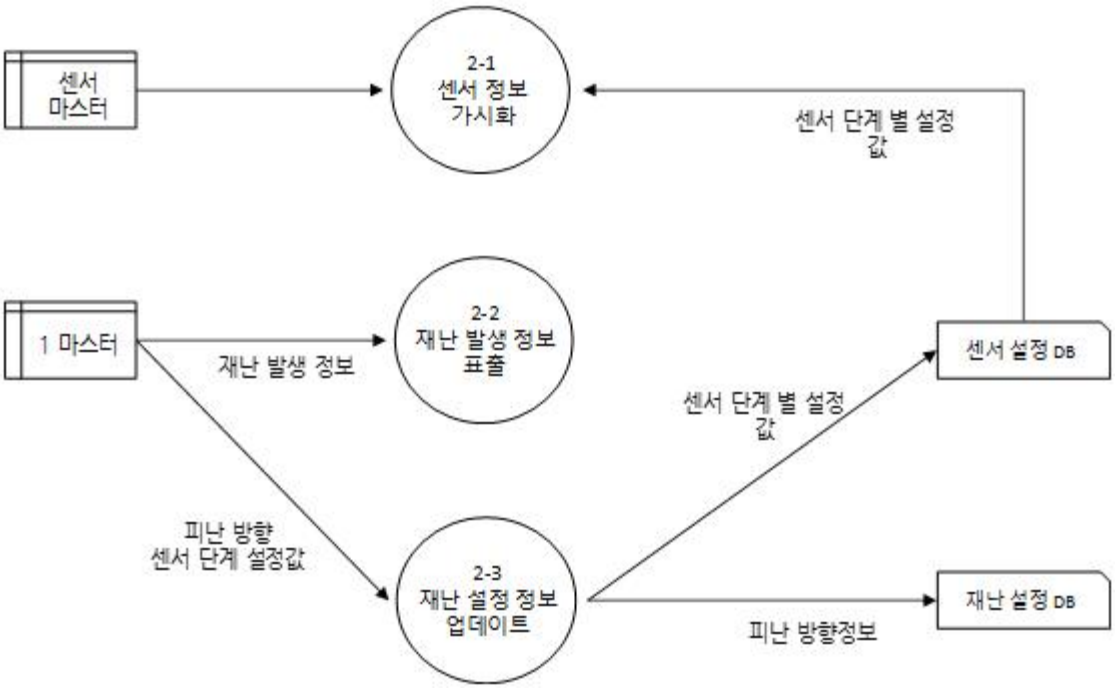
라) 1-4 센서정보 조회

- 센서 데이터는 마스터에 저장되며, 관리자가 기간을 설정해서 과거 센서 데이터를 조회할 수 있다.



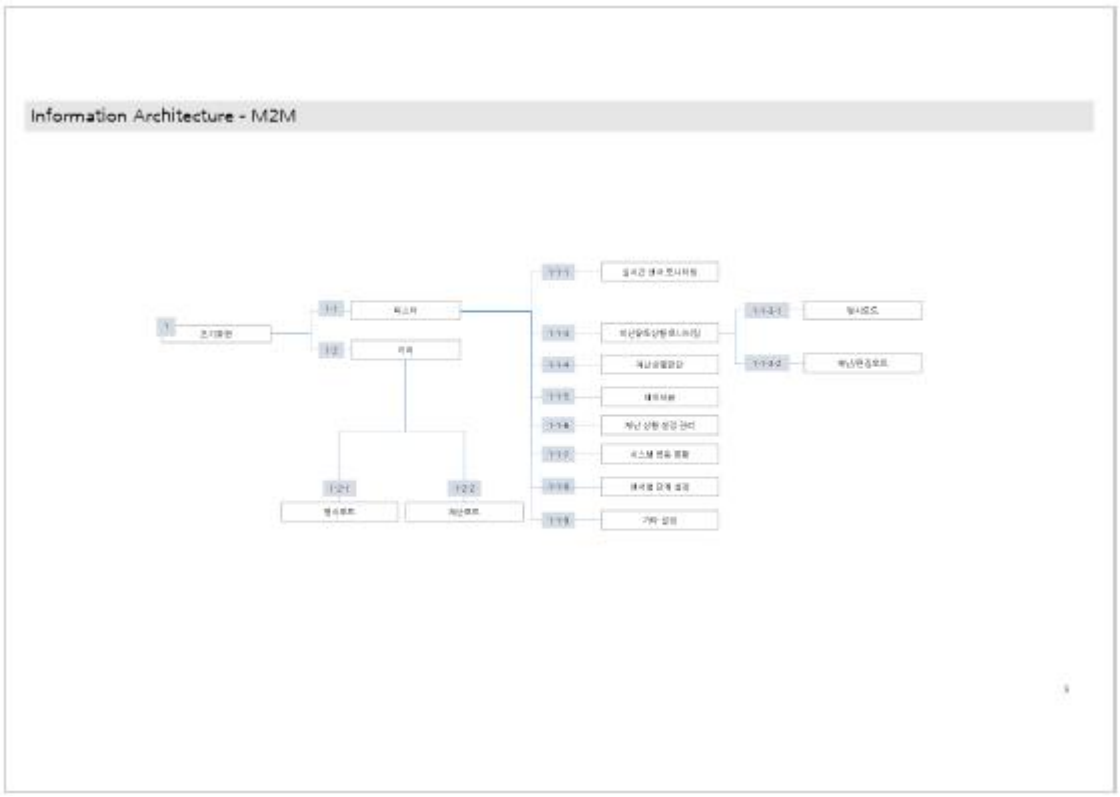
6) 2 키퍼

- 키퍼에서는 센서 마스터에서 전달되는 센서 정보를 가시화하며, 마스터에서 재난상황을 전달하면 재난 모들로 전환하며, 피난유도 정보를 이용객에게 전달한다.
- 마스터에서 피난유도 및 센서 별 단계 설정 정보가 평시에 전파되면 그 정보를 재난 설정 DB와 센서 설정DB에 저장하고 시스템 운영에 활용한다.



자. 4-1 UI 기획 개요

- 2차년도의 기능 우선 개발에서 발생한 디자인 문제 및 개발 후 다양한 현업의 의견을 반영해서 디자인안을 업데이트 하였음.
- 전체 시스템 구조를 아래 IA(Information Architecture)와 같이 개편한 후에 각 페이지 별로 기획 및 디자인을 새로 진행하였음.



차. 4-2 UI 기획 및 디자인

- 기획 안과 디자인 안 및 실제 개발 완료 후의 캡쳐 안을 순서대로 나열하였음.

초기화면

1

초기화면 화면기획서

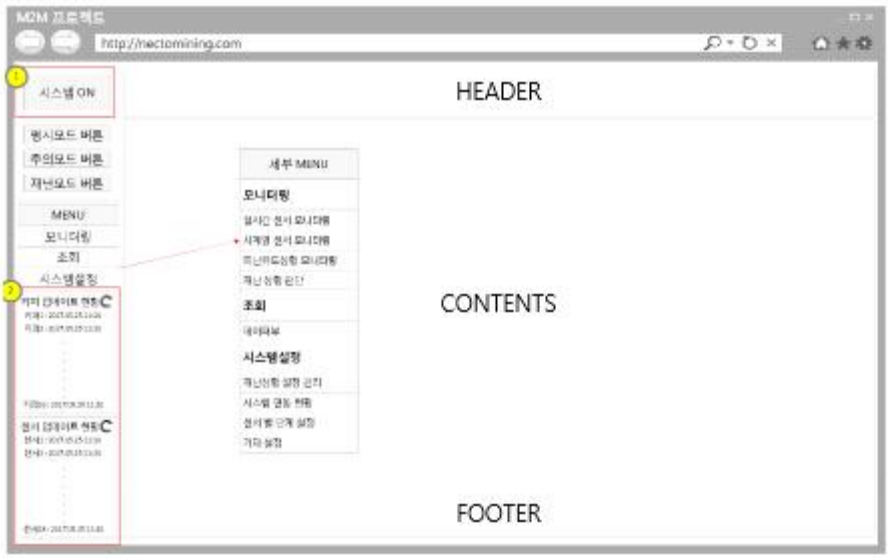


설명
1. 마스터 버튼을 클릭하면 마스터 웹사이트로 연결됨
2. 키퍼 버튼을 클릭하면 키퍼의 디스플레이모드로 연결됨

초기화면 화면캡처



베이스 화면기획서



베이스 화면설계서



설명
1. 마스터에 공통으로 사용되는 템 페이지
2. 왼쪽 메뉴를 기본으로 함

베이스 화면캡처

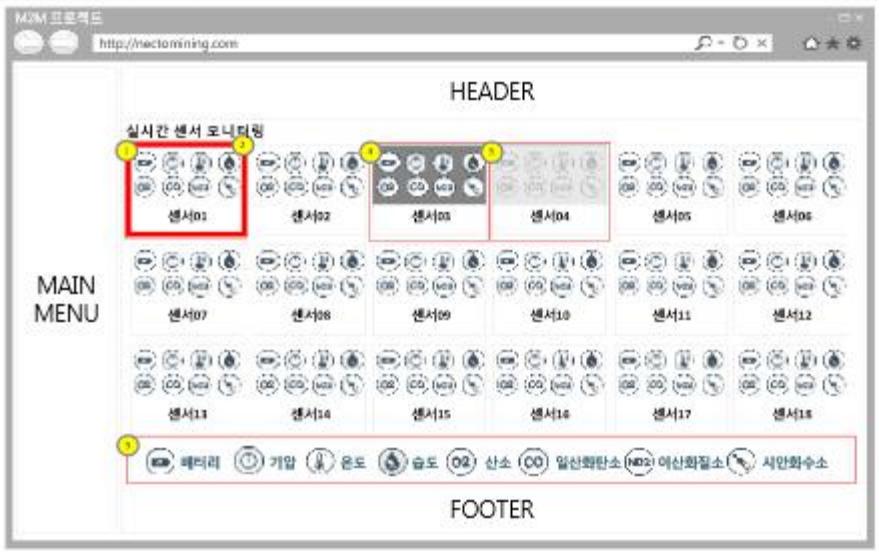


마스터

1-1

실시간 센서 모니터링

화면기획서



설명

1. 각 18개 센서에 대한 8개의 센서데이터 요소가 키퍼의 센서상태가시화와 동일하게 아이콘으로 5단계로 나타남 (정상, 주의, 위험 모두 각 5단계씩)
2. 센서별로 하나라도 이상이거나 주의인경우 센서 박스 테두리가 빨간색으로 변하도록 함
3. 센서 아이콘 범례
4. 이상상태설정 페이지에서 지정한 시간보다 오래된 센서 데이터일시 바탕색을 회색으로 함.
5. 해당 센서에 데이터가 하나도 없을 경우 옅은 회색 처리함.

실시간 센서 모니터링

화면설계서



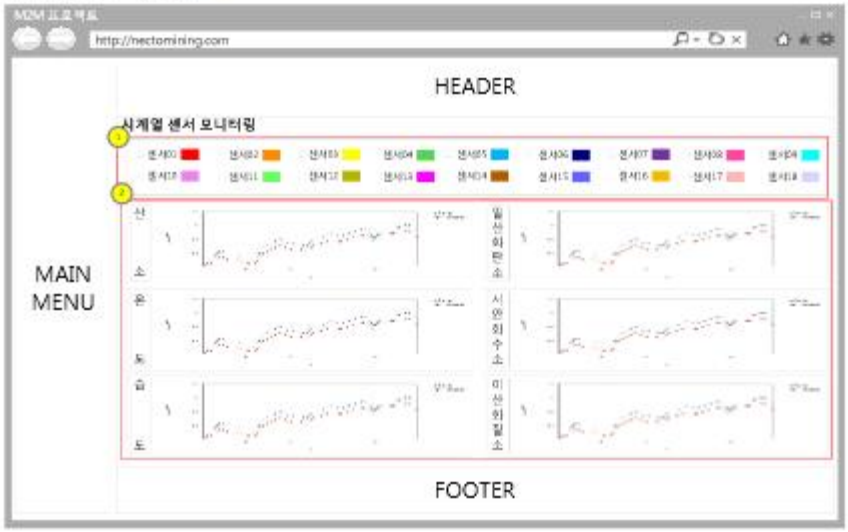
설명
1. UI/UX 디자인이 잘 되도록 함.
2. 재난 발생의 원인이 되는 센서는 붉은색으로 표현함

실시간 센서 모니터링

화면캡처



시계열 센서 모니터링 화면기획서

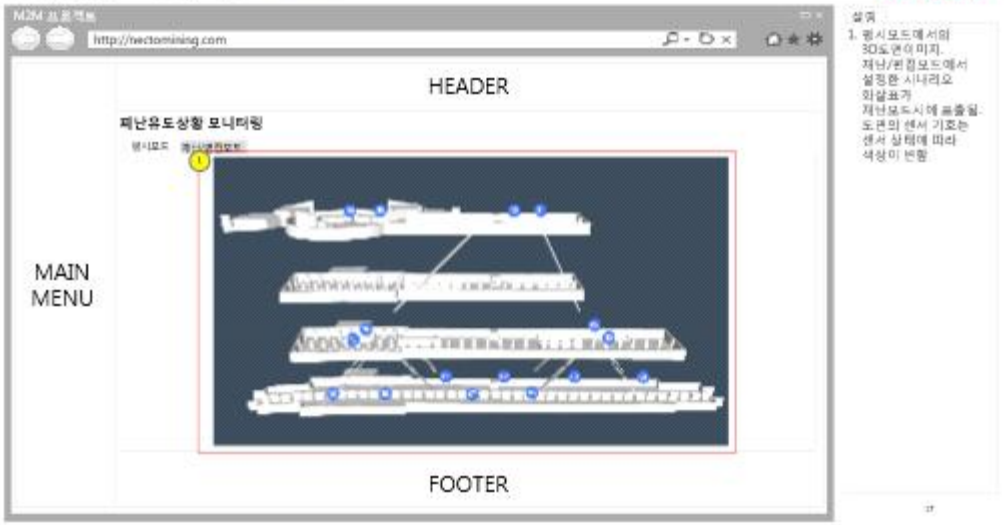


설명
1. 센서 등록.
체크박스에서 원하는 데이터의 센서를 체크하면 그 센서의 데이터만 볼 수 있음 (개발완료 후 협의하여 작업할 것)
2. CH-GL-16 Line Chart - 6개의 센서 데이터 표소. 각 데이터에는 18개의 센서 데이터가 있음 (선이 18개) 시간의 디폴트 값 1일.

시계열 센서 모니터링 화면캡쳐



피난유도상황 모니터링 - 평시모드 화면기획서



피난유도상황 모니터링 – 재난/편집모드 화면기획서



피난유도상황 모니터링 - 평시모드

화면설계서



설명
1. 재난이 발생한 센서는 붉은 색으로 표시해서 구분이 되도록 함
2. 평시의 센서는 회색으로 표시함

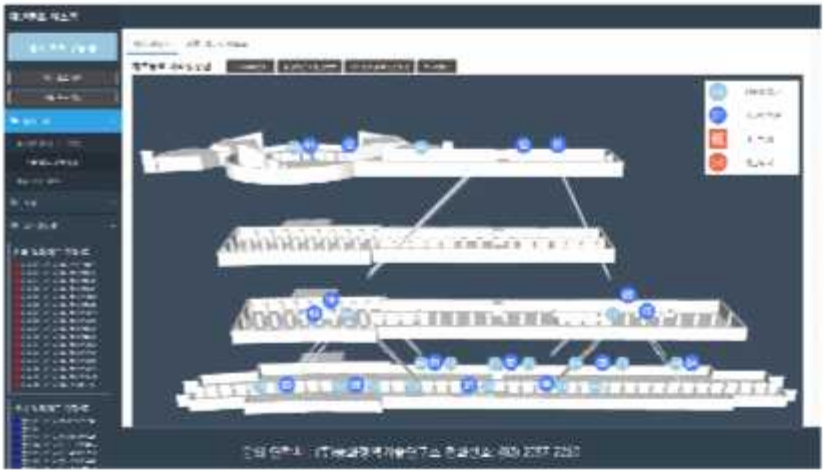
피난유도상황 모니터링 - 재난/편집모드

화면설계서



설명
1. 각 센서 별로 재난 종류와 피난 방향을 설정할 수 있도록 함
2. 대피전략을 간단하게 입력할 수 있도록 텍스트박스가 추가됨

피난유도상황 모니터링 - 평시모드 화면캡쳐



피난유도상황 모니터링 – 재난/편집모드 화면캡처



재난 상황 판단 (V02)

화면설계서



설계
1. 스택이 쌓임에 따라서 원의 크기가 커지도록 디자인함
2. 센서하고 연결이 단절된 경우에는 블루색, 센서에 이상이 있는 경우에는 주황색으로 표시해서 인지도를 높임
3. 각 센서 칸 옆에 데이관을 넣어서 인지도를 높임

재난 상황 판단

화면캡쳐



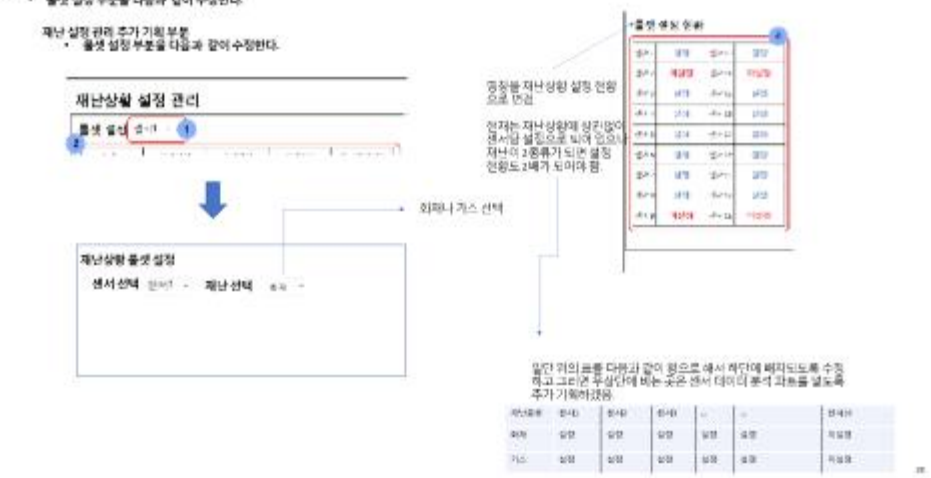
데이터뷰 화면기획서



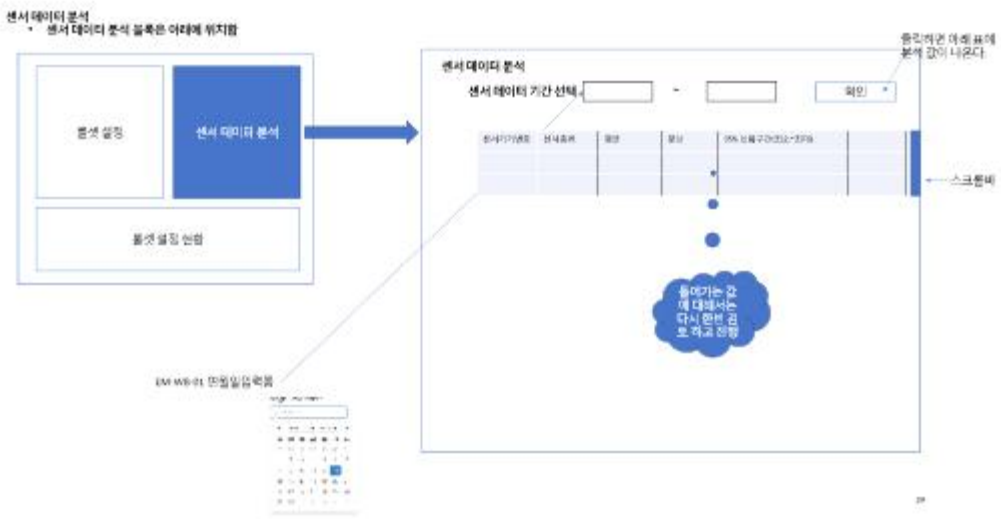
데이터뷰 화면캡처



재난 상황 조건 설정 화면기획서



재난 상황 조건 설정 화면기획서



재난 상황 조건 설정

화면캡처



시스템 연동 현황

화면기획서



설명

1. 마스터가 설정한 재난유도 시나리오, 센서별 단계설정, 키피루의 설정이 키피에 연결된 여부 현황을 보고자 하는 것임

2. 재난유도상황 모니터링의 재난/원킨 모드에서 설정한 화재 또는 미난에 대해 설정된 키피별 미난 시나리오가 키피로 전파된 성공 여부를 ○, X로 나타낸다.(전파 : ○, 실패 : X)

3. 센서 별 단계 설정 페이지에서 설정한 각 센서별 5단계의 경계 값이 키피에게 전파된 여부를 나타낸다.(전파 : ○, 미전파 : X)

4. 기타설정 페이지에서 설정한 키피 루터 설정의 내용이 키피에 연결된 여부를 나타낸다.(전파 : ○, 미전파 : X)

5. 재 전파 버튼을 클릭하면, 전체 키피에 대해서 [시나리오], [센서별 단계 설정], [키피 루터 설정]을 다시 전파하게 된다

시스템 연동 현황

화면기획서



시스템 연동 현황

화면기획서



시스템 연동 현황

화면설계서



설명
1. 데이블 방식으로 시스템 연동현황을 표시함.
2. 연동 정보에 문제가 있는 경우에는 적전과를 알 수 있도록함.

시스템 연동 현황

화면설계서



설명
1. 데이블 방식으로 시스템 연동현황을 표시함.
2. 연동 정보에 문제가 있는 경우에는 적전과를 알 수 있도록함.

시스템 연동 현황

화면캡쳐



센서별 단계 설정

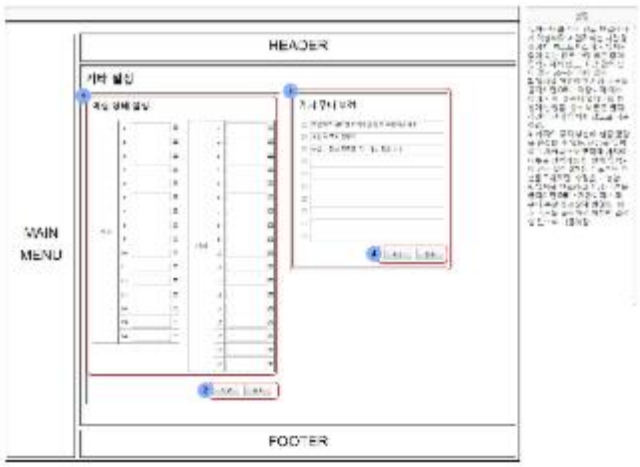
화면기획서



센서별 단계 설정 화면캡쳐



기타 설정 화면기획서



기타 설정

화면캡쳐



키퍼

1-2

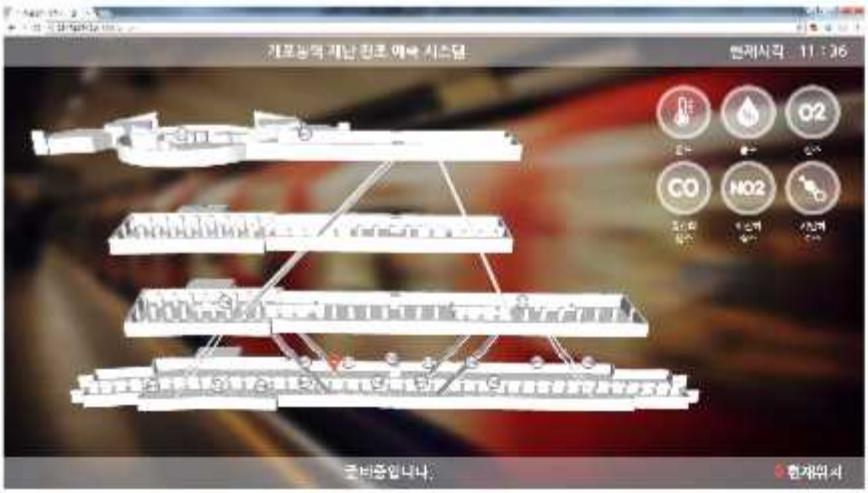
평시모드

화면설계서



평시모드 - 센서모니터링

화면캡처



평시모드 – eBook 모드 화면캡처



재난모드 화면기획서



재난모드(화재발생1) 화면설계서



재난모드 (화재발생2) 화면설계서



재난모드 (화학테러발생1) 화면설계서



재난모드 (화학테러발생2) 화면설계서



재난모드(화재발생1) 화면캡쳐



재난모드 (화재발생2) 화면캡처



재난모드 (화학테러발생1) 화면캡처



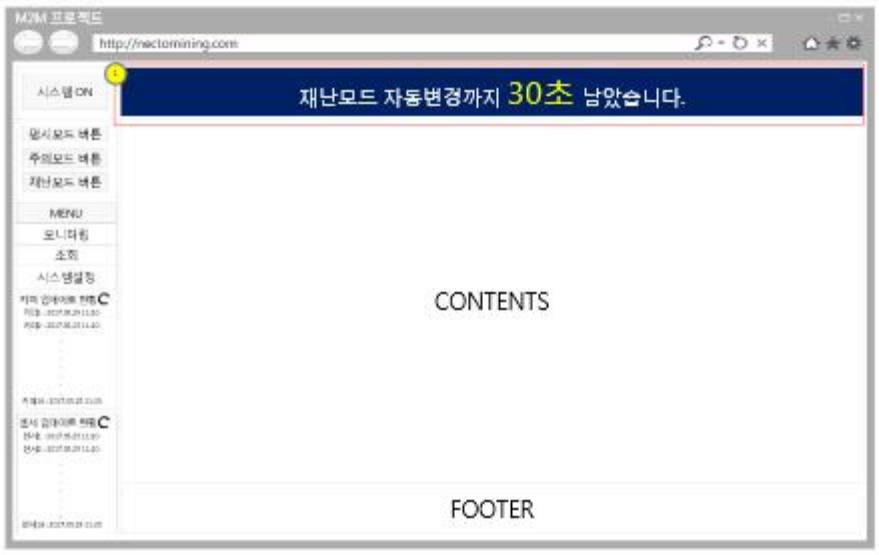
재난모드 (화학테러발생2) 화면캡처



기능모음

카운트다운

화면기획서



설명

1. 노란색 초 숫자 문구가 점멸거리게 함
센서에 재난이 감지되는데 한동안 아무 액션이 없을 경우, 30초부터 카운트다운을 시작하고 키퍼 및 대스터에서는 주의모드가 발동됨. 카운트다운이 끝나면 재난모드로 자동변경됨. 평시모드에는 해당 텍스트를 띄우지 않게 함. (재난이 감지될 경우에만 띄움)

카운트다운 화면설계서



설명
1. 재난 감지 후 30초 동안 카운트 다운이 수행됨

카운트다운 화면캡쳐



모드변경 가시화 - 재난 모드 화면캡처



센서아이콘 - 디폴트 화면설계서



센서아이콘 - 안전 화면설계서

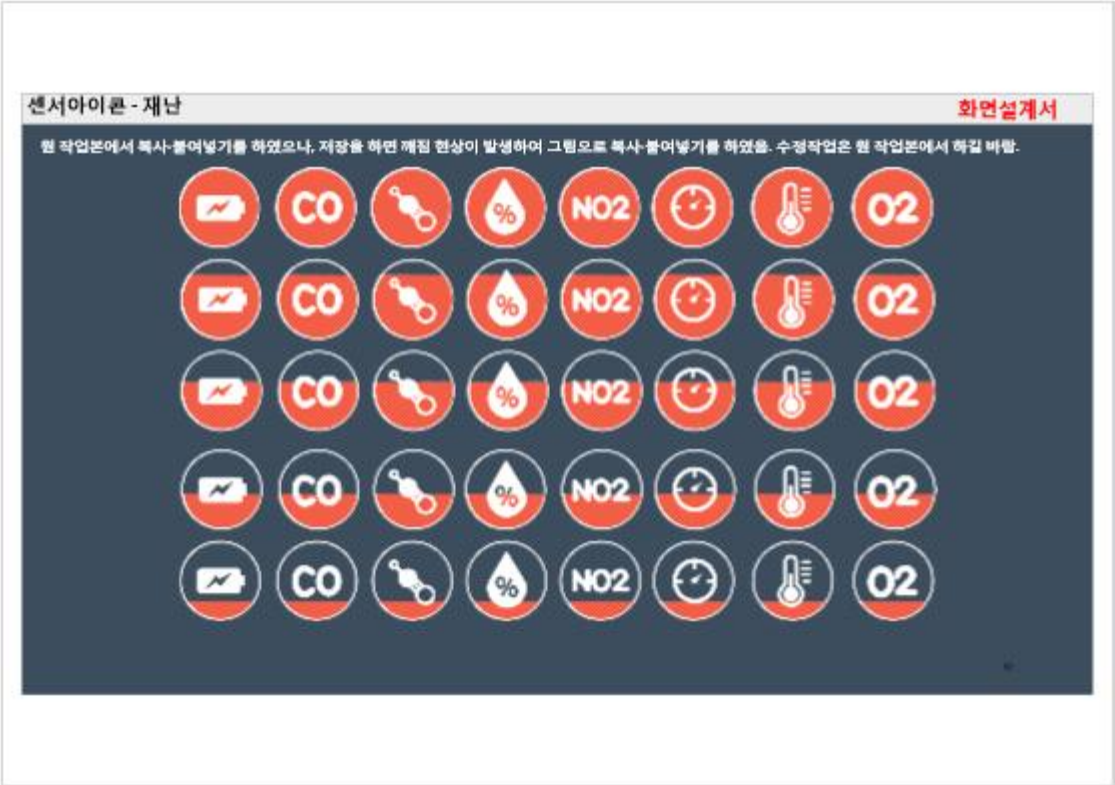
원 작업본에서 복사-붙여넣기를 하였으나, 저장을 하면 깨짐 현상이 발생하여 그림으로 복사-붙여넣기를 하였음. 수정작업은 원 작업본에서 하길 바람.



센서아이콘 - 주의 화면설계서

원 작업본에서 복사-붙여넣기를 하였으나, 저장을 하면 깨짐 현상이 발생하여 그림으로 복사-붙여넣기를 하였음. 수정작업은 원 작업본에서 하길 바람.





라. 4-3 피난 유도 화살표 디자인

개요

- 키퍼 및 마스터에 사용하는 피난 유도 아이콘은 아래와 같이 디자인되었음

화살표 디자인

	코드네임	설명
	Left	왼쪽 직진 유도
	Right	오른쪽 직진 유도
	Up	앞으로 직진 유도
	Down	뒤로 직진 유도
	RS	오른쪽-왼쪽 코너 유도
	LS	왼쪽-오른쪽 코너 유도
	Uturn	직진-왼쪽-왼쪽 유턴 코너 유도
	Rstairs	오른쪽 계단으로 올라가도록 유도

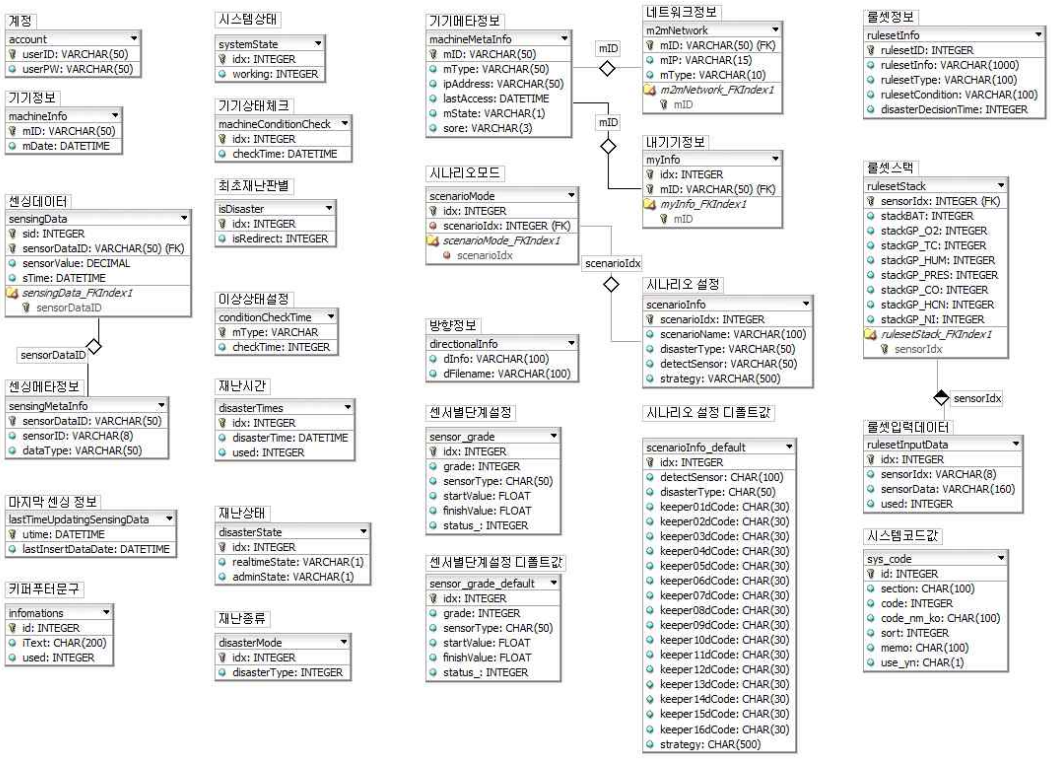
	Lstairs	왼쪽 계단으로 올라가도록 유도
	LLstairs	왼쪽 직진-왼쪽 계단으로 유도
	LRstairs	왼쪽 직진-오른쪽 계단으로 유도
	RRstairs	오른쪽 직진-오른쪽 계단으로 유도
	RLstairs	오른쪽 직진-왼쪽 계단으로 유도
	RSLstairs	오른쪽-왼쪽 코너 후 왼쪽 계단으로 유도
	LSRstairs	왼쪽-오른쪽 코너 후 오른쪽 계단으로 유도
	UpRstairs	앞으로 직진-오른쪽 계단으로 유도
	UpLstairs	앞으로 직진-왼쪽 계단으로 유도
	DownRstairs	뒤로 직진-오른쪽 계단으로 유도
	DownLstairs	뒤로 직진-왼쪽 계단으로 유도

타. 5-1시스템 개발 개요

개요

- 2차년도의 개발 이후 성능 테스트 및 현장 점검을 통해서 보완이 필요한 부분

마스터의 DB 구조



파. 5-2 주요 기능 별 개발 내용

1) 5-2-1 시나리오 전파 기능 개발 내용 작성

시나리오 작성 관련 수정사항

- 기존 방식
  - 시나리오를 직접 작성 후 관리자가 시나리오를 선택하여 발동시킴
- 변경 후
  - 모든 시나리오에 대해 설정한 뒤 재난발생시 미리 설정해둔 시나리오가 발동함.
  - 기본적으로 모든 시나리오에 대해 디폴트 설정이 있고 관리자가 시나리오를 설정을 할시 우선순위는 관리자가 설정한 값임

시나리오 전파 기능

- 프로그램 설치시 디폴트 시나리오는 프로그램내의 DB에 기본값이 있음.
- 마스터에서 키퍼로 시나리오 내용을 전파할시 DB에 등록된 모든 키퍼IP에 대하여 관리자가 저장한 시나리오만 get방식으로 전파함.
- 파라메터
  - 센서명
  - 재난종류
  - 키퍼화살표정보
  - SOP
- 시나리오순서는 센서번호 순으로 전파하며 UI는 시스템설정-시스템연동현황페이지-SOP 대피경로전파현황에 있음.

- 

2) 5-2-2 센서 데이터를 가져오는 기능 내용 작성

DB에서 센서데이터를 가져오는 방법

- 데이터 저장
  - 장고 라이브러리인 django-crontab 를 사용하여 센서 데이터 서버에서 센서 데이터를 읽어오는 스케줄러를 작성함.
  - 최근 센서 데이터를 읽어와 중복을 제거 하고 DB에 저장함.
  - 마스터에서 사용함.
- 최근 데이터 로딩
  - DB에 저장된 센서 데이터에 대해 시간순으로 정렬을 하여 마지막 데이터를 받아옴.
  - 각 센서별로 계산을 하여 json 데이터로 리턴함.
- 코드

```
def get_sensingDataFromDB():
        ret={}
        sensingData_rows = sensingData.objects.all()
        sensorName = ["-BAT" ,"-GP_CO","-GP_HUM","-GP_NI","-GP_O2","-GP_PRES","-GP_TC","-GP_HCN",]
        sensorCount = 18
        for sIdx in range(1,sensorCount+1):
                subIdx = ("0"+ str(sIdx)) if sIdx <10 else str(sIdx)
                sensor = "SHTPI0"+subIdx
                try:
                        sData= sensingData_rows.filter(sensorDataID=sensor+"-BAT").latest('sid')
                except ObjectDoesNotExist:
                        sData=None
                if sData:
                        rows = sensingData_rows.filter(sTime=sData.sTime)
                        tmp = {"data": 0 ,"sTime" :0 }
                        for sType in sensorName:
                                ret[sensor+sType]=tmp
                        for row in rows:
                                ret[row.sensorDataID] = {"data" : round(float(row.sensorValue),4) , "sTime" :
                                str(row.sTime)}
                else:
                        tmp = {"data": 0 ,"sTime" : "" }
                        for sType in sensorName:
                                ret[sensor+sType]=tmp
        return ret
```

URL을 통해 센서데이터를 가져오는 방법

- 센서데이터서버에 저장된 최근 데이터1건을 받아오며 데이터 저장은 없음
- 키퍼에서 사용함.
- 센서 데이터 서버IP는 외부주소(223.171.32.11)가 디폴트값으로 입력되어있으며 내부설정시 localhost로 수정을 하면 속도가 상승됨
  - settings.py 파일의 SHOST 변수 수정
- 코드

```
def get_sensingDataFromURL():
        startIdx = -1
        endIdx = -1
        sensorCount = 18
        list_sensors = ['BAT', 'GP_NI' ,'GP_TC', 'GP_CO', 'GP_O2', 'GP_PRES', 'GP_HUM', 'GP_HCN']
        retData = {}
        for sidx in range(1,sensorCount+1):
                id_wasp = "" if sidx>9 else "0"
                sensorName = 'SHTPI0%s%s' %(id_wasp ,sidx)
                dict_data = readSensorDataToJson(sidx,startIdx,endIdx)
                if dict_data:
                        timeStamp = dict_data[0]["Time Stamp Server"]/1000
                        sTime = datetime.fromtimestamp(timeStamp).strftime("%Y-%m-%d %H:%M:%S")
```

```
                        else:
                                sTime = 0
                        retData[sensorName+"-"+list_sensors[0]]=addSensorData("Battery", dict_data ,sensorName, list_sensors[0],
                        sTime)
                        retData[sensorName+"-"+list_sensors[1]]=addSensorData("Nitrogen Dioxide", dict_data ,sensorName,
                        list_sensors[1], sTime)
                        retData[sensorName+"-"+list_sensors[2]]=addSensorData("Temperature Celcius", dict_data ,sensorName,
                        list_sensors[2], sTime)
                        retData[sensorName+"-"+list_sensors[3]]=addSensorData("Carbon Monoxide", dict_data ,sensorName,
                        list_sensors[3], sTime)
                        retData[sensorName+"-"+list_sensors[4]]=addSensorData("Oxygen", dict_data ,sensorName,
                        list_sensors[4], sTime)
                        retData[sensorName+"-"+list_sensors[5]]=addSensorData("Pressure", dict_data ,sensorName,
                        list_sensors[5], sTime)
                        retData[sensorName+"-"+list_sensors[6]]=addSensorData("Humidity", dict_data ,sensorName,
                        list_sensors[6], sTime)
                        retData[sensorName+"-"+list_sensors[7]]=addSensorData("Hydrogen Cyanide", dict_data ,sensorName,
                        list_sensors[7], sTime)
                return retData

        def readSensorDataToJson(sIdx,startIdx,endIdx):
                host = settings.SHOST
                id_wasp = ""
                if sIdx<10:
                        id_wasp = "0"
                sensorNode = "SHTPI0"+id_wasp+str(sIdx)
                nStart = startIdx
                nStop = endIdx
                url = host + "/data?sensorNode="+sensorNode +"&nStart="+str(nStart)+"&nStop="+str(nStop)
                response = urllib.request.urlopen(url)
                content = response.read()
                data = json.loads(content.decode("utf8"))
                return data

django-crontab
```

- 리눅스에서만 사용가능하며 리눅스의 crontab 기능을 장고에서 연동하여 사용할수 있는 라이브러리.
- 장고프로그램에서 사용하기 위한 설정
  - settings.py 파일

    INSTALLED_APPS = (

        ...

        'django_crontab',

)

    CRONJOBS = [

        ('* * * * *', 'crons.cron.job_name')

    ]

- 반복시간은 리눅스 스케줄러 기능인 crontab과 설정은 동일함.
  - * * * * * 수행할 명령어
  - 각 자리는 "분시일월요"를 뜻함
  - 분 (0 - 59)
  - 시 (0 -23)
  - 일 (1 -31)
  - 월 (1 -12)
  - 요일 (0 - 6) (0:일요일,1:월요일..., 6:토요일)
  - * * * * * 이렇게 설정하면 매 1분마다 작업 수행을 뜻함.
- 장고에서 django-crontab 조회/추가/제거
  - 조회 : python3 manage.py crontab show
  - 추가 : python3 manage.py crontab add
  - 제거 : python3 manage.py crontab remove
  - 콘솔로 프로그램 Root 위치에서 위의 명령어를 실행

3) 5-2-3 키퍼 평시모드에서 풋터 메세지 디스플레이 기능 개발 내용 정리

메세지 데이터 관리

- 키퍼의 풋터에 디스플레이 될 텍스트는 마스터의 시스템설정-기타설정-키퍼푸터설정란에서 내용을 작성함.
- 사용할 텍스트와 미사용 텍스트는 관리할수 있으며 3개 이하로 작성시에는 자동으로 다음의 텍스트가 DB에 저장되면서 최소3개 이상의 텍스트가 입력됨.
  - 자동으로 입력되는 첫번째 텍스트 : 안전하고 편리한 지하철을 많이 이용해주세요.
  - 자동으로 입력되는 두번째 텍스트 : 오늘 하루도 안전히
  - 자동으로 입력되는 세번째 텍스트 : 항상 시민의 안전을 감시하고 있습니다.
- 설정에서 저장한 메세지는 키퍼로 전파해야지만 동기화가 됨
- 코드

```
def setText(request):
        defaultText = [
                        "안전하고 편리한 지하철을 많이 이용해주세요.",
                        "오늘 하루도 안전히",
                        "항상 시민의 안전을 감시하고 있습니다."
        ]
        for idx in range(0,10):
                checkbox_text = request.POST.get("checkbox_text"+str(idx))
                text= request.POST.get("text"+str(idx))
                if text:
                        created = infomations.objects.update_or_create(
                        id = idx+1,
                        defaults = {
                                "iText" : text,
                                "used" : 1 if checkbox_text else 0,
                                }
                        )
                else:
                        if idx<3:
                                text = defaultText[idx]
                                created = infomations.objects.update_or_create(
                                        id = idx+1,
                                        defaults = {
                                                "iText" : text,
                                                "used" : 1 if checkbox_text else 0,
                                        }
                                )
                        else:
                                infomations.objects.filter(id=(idx+1)).delete()

        retMsg="저장되었습니다."
        return HttpResponse('<script>alert("'+retMsg+'");window.history.back();</script>')
```

데이터 로딩

- 키퍼는 로컬DB의 infomations 테이블의 used 컬럼이 1인 값을 읽어와 해당 텍스트를 배열에 담아 프론트엔드로 넘김.
- 10초에 한번씩 업데이트
- DB의 infomations 테이블 스키마
  - iText
    - 문자열
    - max_length 200
    - 메세지 텍스트
  - used
    - 정수형
    - default 0
    - 사용여부(0 , 1) (0 : 미사용, 1 : 사용)
- 코드

```
def updateText(request):
        ret = []
        infomations_rows = infomations.objects.filter(used=1)
        if infomations_rows:
                for row in infomations_rows:
                        ret.append(row.iText)
        else:
                ret = ["준비중입니다."]
        return HttpResponse(json.dumps(ret))
```

데이터 디스플레이

- 프론트엔드처리. javascript로 작성.
- 디폴트값으로 "준비중입니다." 텍스트를 디스플레이함.
- 데이터로 로딩 기능에 의해 넘어온 배열의 텍스트(들)을 5초에 한번씩 순차적으로 표기함.
- 코드

```
function displayText(){
        var tag = $("#bottom_position>.page-toolbar-title");
        if(infomations.length==0){
                $(tag).text("준비중입니다.");
                infomations_idx = 0;
        }else{
                if(infomations.length<infomations_idx){
                        infomations_idx = 0;
                }
                $(tag).text(infomations[infomations_idx]);
                infomations_idx++;
        }
}
```

4) 5-2-4 센서별 단계 설정 기능 내용 정리

단계 설정값 관리

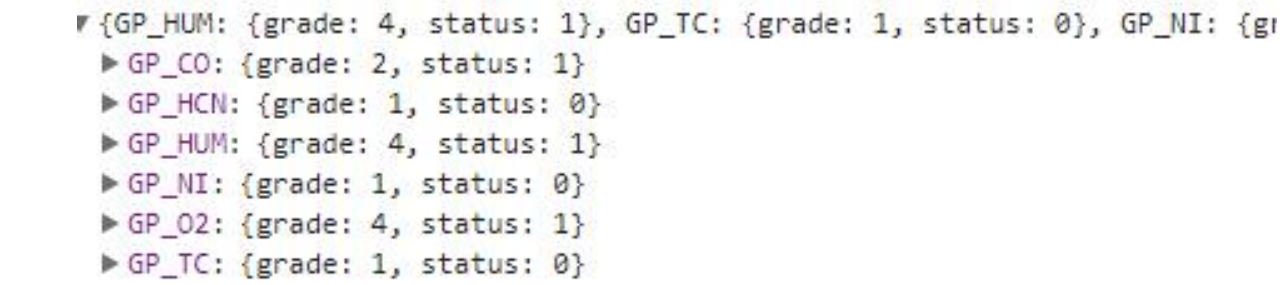
- 프로그램 설치시 디폴트 설정값이 있으며 관리자가 값을 설정 할 경우 우선순위는 관리자가 설정한 값임
- 각 센서별로 5단계의 경계값 설정은 마스터의 시스템설정-센서별 단계 설정 페이지에서 값을 저장함.
- 값을 저장시 히스토리는 관리하지 않으며 기존의 데이터에 저장된 값을 업데이트함.
- 설정에서 저장한 메세지는 키퍼로 전파해야지만 동기화가 됨

데이터 로딩

- DB의 sensor_grade_default 와 sensor_grade 테이블 스키마는 동일하며 아래와 같음.
    - idx
        - index, primary key
    - grade
        - 정수형
        - 단계값 (1 - 5)
    - sensorType
        - 문자열
        - 센서종류이며 정의는 sys_code 테이블을 참조. (GP_O2, GP_CO,GP_HCN,GP_TC,GP_HUM,GP_NI)
    - startValue
        - 실수형
        - 범위값의 시작값이며 범위의 표기는 이상임
        - 범위가 1이상 5미만일 경우 1을 기록함.
    - finishValue
        - 실수형
        - 범위값의 끝값이며 범위의 표기는 미만임
        - 범위가 1이상 5미만일 경우 5를 기록함.
    - status
        - 정수형
        - 상태값인 0 - 2 의 값을 가지며 정의는 sys_code 테이블을 참조. (0 : 안전 ,

1 : 주의 , 2 : 위험)

- 6개의 센서데이터를 입력 받아 계산 후 각 센서별로 단계값과 상태값을 json 데이터로 리턴함.



  - grade : 단계 , status : 상태값
- 센서별로 단계를 계산하는 코드

```
def setSensorGrade(lst):
        ret = {}
        list_sensorType = ["GP_O2","GP_CO","GP_HCN","GP_TC","GP_HUM","GP_NI"]
        if len(lst)==len(list_sensorType):
                idx = 0
                if len(sensor_grade.objects.all())==0:
                        sensor_grade_rows = sensor_grade_default.objects.all()
                else:
                        sensor_grade_rows = sensor_grade.objects.all()
                for sList in list_sensorType:
                        if not lst[idx]=="":
                                rows = sensor_grade_rows.filter(sensorType=sList)
                                for row in rows:
                                        if row.startValue == row.finishValue:
                                                if (lst[idx] if not sList=="GP_O2" else (lst[idx]/10000)) == row.startValue:
                                                        ret[sList] = {"grade" : row.grade, "status" : row.status}
                                                        break
                                        else:
                                                if (lst[idx] if not sList=="GP_O2" else (lst[idx]/10000))>=row.startValue and
                                                (lst[idx] if not sList=="GP_O2" else (lst[idx]/10000))<row.finishValue:
                                                        ret[sList] = {"grade" : row.grade, "status" : row.status}
                                                        break
                                if not sList in ret:
                                        ret[sList] = None
                        else:
                                ret[sList] = None
                        idx += 1
        return ret
```

5) 5-2-5 분기룰을 적용시키는 기능 내용 작성

룰셋값 관리

- 센서별로 룰셋값을 입력하며 마스터의 시스템설정-재난상황조건설정페이지에서 내용을 작성함.
- 기존에는 하나의 룰셋값만 설정하였는데 센서별로 환경이 다를수가 있어 각각 룰셋을 입력할수 있게 하였음.
- 마스터 기능

테이블 스키마

- rulesetInfo 스키마
  - idx
    - index, primary key
  - rulesetInfo
    - 문자열
    - 룰셋설명
  - rulesetType
    - 문자열
    - 룰셋타입
  - rulesetCondition
    - 문자열
    - 룰셋조건

- disasterDecisionTime
  - 정수형
  - 재난판별시간
- mID
  - 문자열
  - 센서ID (SHTPI + 001,002 등의 3자리 숫자)
- rulesetID
  - 정수형
  - 룰셋ID

스택관리

- 센서데이터가 DB에 입력되면 룰셋조건에 의해 참이면 센서별로 스택이 증가함.
- 스택정보는 DB에 저장됨.

판별

- 프론트엔트에서 저장된 스택값을 배열로 리턴받음.
- 스택 상태는 마스터의 모니터링-재난상황판단페이지의 재난 발생 판단 정보에서 확인이 가능하며 증가된 스택을 계산하여 재난 판별 시간보다 더 높게 되면 재난 모드 프로세스가 작동함.
  - 스택 계산 기능은 5초에 한번씩 실행되어 스택x5초로 하여 재난판별시간과 비교를 함.
- 코드

```
def adjustRuleset(inputData,ruleset):
        if len(inputData)<8:
                return
        choices = dict(
                BAT = inputData[0],
                GP_O2 = inputData[1],
                GP_TC = inputData[2],
                GP_HUM = inputData[3],
                GP_PRES = inputData[4],
                GP_CO = inputData[5],
                GP_HCN = inputData[6],
                GP_NI = inputData[7],
        )
        # initialize output
        output = [1, 1, 1, 1, 1, 1, 1, 1]
        # for idxRule in range(0, 8):
        for idxRule in range(0, len(ruleset)):
                ruleItem = ruleset[idxRule][0]
                ruleUnit = ruleset[idxRule][1]
                inputVal = choices[ruleItem]
                rlt = 0
                exec("%s = %s" % (ruleItem, inputVal))
                if inputVal is None:
                        continue
                if eval(ruleUnit):
                        rlt = 2
                if (not inputVal == None) or (inputVal != -1):
                        if ruleItem == "BAT":
                                output[0] = rlt
                        if ruleItem == "GP_O2":
                                output[1] = rlt
                        if ruleItem == "GP_TC":
                                output[2] = rlt
                        if ruleItem == "GP_HUM":
                                output[3] = rlt
                        if ruleItem == "GP_PRES":
                                output[4] = rlt
                        if ruleItem == "GP_CO":
                                output[5] = rlt
                        if ruleItem == "GP_HCN":
                                output[6] = rlt
                        if ruleItem == "GP_NI":
                                output[7] = rlt
        return output
```

하. 6 센서 데이터를 이용한 재난전조 감지

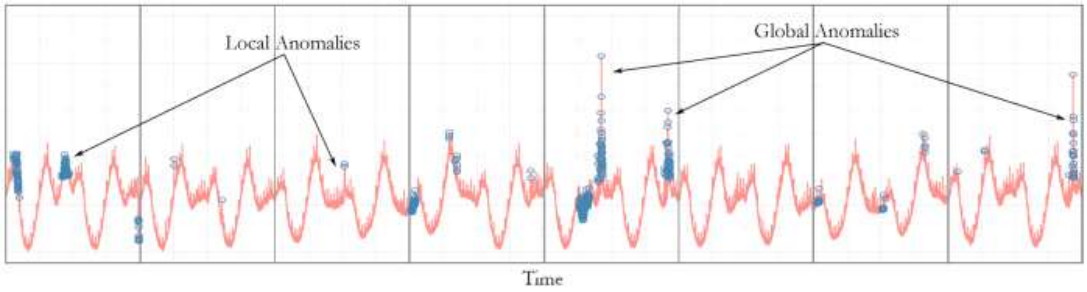
개요

- 센서에서 올라오는 실시간 데이터를 이용해서 재난 전조를 조기에 감지하면서, 센서의

일시적인 오작동을 필터링하는 기능을 수행하는 이상전조 예측 모델을 생성하고 지속적인 센서 데이터를 이용한 학습을 통해서 안정적인 재난전조 예측 기술을 개발하고자 함

센서 데이터의 개요

- 아래 그림과 같이 일정한 범위의 값의 시계열 데이터에 정상상황과 이상상황의 데이터가 혼재되어서 들어오는 경우, 이상상황을 판별해 내는 방법론 총칭



센서 데이터 정상 범위 설정

- 기존의 센서 데이터를 각 센서 종류별로 시간별로 정리 후에 값의 범위를 설정해서 정상범위를 선정한다.
- 마할로코프 거리 3을 정상범위로 설정해서 전체 데이터에 대해서 99%가 정상범위가 되도록 센서 값의 범위를 잡음
- 단 한번의 임계치 초과로 화재로 판명되면 곤란하므로, 이상 상황 설정은 임계치를 연속적으로 몇번 이상 넘었을 때 혹은 임계치를 넘은 상태를 플러스1, 임계치를 내려가면 마이너스1로 했을 때 총 누진 가중치가 일정값 이상을 넘어갔을 때로 설정을 하도록 함.

입력데이터

- 각 센서 디바이스의 센서 종류 별 시계열 데이터를 시스템에서 지정한 기간 동안 DB에 쿼리를 던져서 리턴을 받은 다음에 csv 파일로 저장해서 분석에 활용함

입력 데이터 파싱

- csv 모듈을 이용해서 데이터를 읽어들임.
  - csv 파일 형태는 아래와 같음
    - 1,SHTPI001-BAT,94,2017-01-24 17:33:14
    - 2,SHTPI001-GP_NI,0.00E+00,2017-01-24 17:33:14
    - 3,SHTPI001-GP_TC,16.40999985,2017-01-24 17:33:14
  - 여기서 필요한 정보는 2열의 센서 정보, 3열의 값 4열의 시간임
  - 아래 파이썬 코드를 이용해서 읽어들임

```
import csv
csv_file = open("input.csv", 'r')
reader = csv.reader(csv_file)
mtxData = [];
for row in reader:
mtxData.append(row)
```

  - 다음과 같은 리스트 형태로 읽어들여짐

```
In [4]: mtxData[0]
Out[4]: ['1', 'SHTPI001-BAT', '94', '2017-01-24 17:33:14']

In [5]: mtxData[0][0]
Out[5]: '1'
```

- 자세히 보면 이중 리스트 형태임을 알 수 있음

- 중요하진 않지만 일단 필요없는 첫번째 요소를 없애도록 함.
  - 가장 단순한 방법인 for-loop를 이용해서 새 변수에 필요한 것만 담도록 함
  - 아래와 같이 새변수 newMtxData에 필요변수를 담음

```
# remove first element form mtxData because it is unnecessary.
#
newMtxData = []
countData = len(mtxData)
for i in range(0, countData):
row = mtxData[i]
pureRow = row[1:4]
newMtxData.append(pureRow)
```

```
In [17]: newMtxData[0:2]
Out[17]:
[['SHTPI001-BAT', '94', '2017-01-24 17:33:14'],
 ['SHTPI001-GP_NI', '0.00E+00', '2017-01-24 17:33:14']]
```

센서 별 데이터 추출

- 센서리스트 도출을 위한 그룹추출

  http://stackoverflow.com/questions/7961363/removing-duplicates-in-lists

  - 파이썬의 set 함수를 이용해서 그룹 추출
  - List()로 다시 리스트 변수로 변환
- 센서 별로 데이터를 아래와 같은 형식으로 리스트 모음으로 정의함.

  #각 센서별로 데이터 모으기

```
sensorListCount = len(sensorGroup)
for i in range(0,sensorListCount):
sensorDataList.append([])
sensorDataList[i].append(sensorGroup[i])
for i in range(0,sensorListCount):
sensorName1 = sensorDataList[i][0]
for j in range(0,dataCount):
sensorName2 = sensorData[j][0]
if sensorName1 == sensorName2:
sensorDataList[i].append(sensorData[j][1])

W
 ['SHTPI011-GP_HUM',
  ['2017', '01', '24', '16', '29', '31', '21.27441406'],
  ['2017', '01', '24', '16', '37', '37', '21.06933594'],
  ['2017', '01', '24', '16', '41', '20', '21.03613281'],
  ['2017', '01', '24', '16', '15', '48', '21.08398438'],
  ['2017', '01', '24', '16', '48', '03', '20.94433594'],
  ['2017', '01', '24', '16', '50', '07', '21.06054688'],
  ['2017', '01', '24', '16', '51', '06', '21.08300781'],
  ['2017', '01', '24', '16', '56', '26', '21.03027344'],
  ['2017', '01', '24', '16', '57', '15', '20.98828125'],
  ['2017', '01', '24', '16', '57', '34', '20.98828125'],
  ['2017', '01', '24', '17', '01', '36', '20.98828125'],
  ['2017', '01', '24', '17', '01', '50', '21.04296875'],
  ['2017', '01', '24', '17', '02', '14', '21.04296875'],
  ['2017', '01', '24', '17', '03', '43', '21.01074219'],
  ['2017', '01', '24', '17', '04', '38', '21.05371094'],
  ['2017', '01', '24', '17', '13', '03', '21.21777344'],
  ['2017', '01', '24', '17', '14', '24', '21.13085938'],
  ['2017', '01', '24', '17', '16', '20', '21.26074219'],
  ['2017', '01', '24', '17', '16', '32', '21.26074219'],
  ['2017', '01', '24', '17', '17', '29', '21.25'],
  ['2017', '01', '24', '17', '21', '18', '21.31542969'],
  ['2017', '01', '24', '16', '20', '20', '20.99609375'],
  ['2017', '01', '24', '16', '20', '28', '20.99609375'],
  ['2017', '01', '24', '16', '22', '58', '21.015625']]]
```

정상범위 추출

- 센서 데이터에 마할라노비스 거리를 적용한 정상 범위 선정
- 마할라노비스 거리가 3이하의 범위를 각 센서에 대해서 정상범위로 설정함

마할노비스 거리

- 정의
  - 균과의 거리가 표준편차의 몇 배인지를 나타내는 값
- 공식
  - (실측값-평균값)/표준편차
- 의의
  - 어떤 값이 얼마나 일어나기 힘들 값이지, 또는 얼마나 이상한 값인지를 수치화하는 방법
- 용도
  - 가짜 데이터와 진짜 데이터의 구분용도 즉 outlier나 이상검출에 활용됨
- 적용 예
  - 하루 교통량 평균이 20대이고 표준편차가 3대인 어느 날 차가 26대가 지나갔다고 하자. 자로 잰 평균과의 거리는 6이다. 그런데, Mahalanobis distance로는 (26 - 20)/3 = 2이다. 즉, 표준적인 편차의 2배 정도의 오차가 있는 값이라는 것이다.
- 참고
  - 우리가 고등학교 수학에서 (확률과 통계 파트) 표준정규분포로 바꾼 후에 z 값을 구하는 것이 바로 Mahalanobis distance를 구하는 과정이다. 그리고 우리가 통상적으로 사용하는 자로 잰 거리는 유클리디언 거리(Euclidean distance)라고 부른다.

개발 언어

- 파이썬3 버전으로 함수 구현

코드 리뷰

- 마할라노비스 거리 계산

```
# 각 데이터의 거리 구하기
for j in range(0, oneSensorDataCount):
if oneSensorStd != 0:
tmp = abs(oneSensorData[j] - oneSensorMean)/oneSensorStd
else:
tmp = 0
mahaDist.append(tmp)


        # 센서 데이터 전처리를 통한 리스트 형식의 입력 데이터 추출

SHTPI016-GP_HUM (0.82536582606899533, 0.93293280102662091)
SHTPI008-GP_TC (0.25543427692709286, 1.3904658775058838)
SHTPI004-GP_NI 0
SHTPI017-GP_TC (0.73291079208932419, 0.75057813406229623)
SHTPI015-GP_O2 (0.51412997437930441, 1.2394026650297478)
SHTPI015-GP_TC (0.33405723253991348, 1.2523890687296011)
SHTPI011-GP_PRES (0.46535327349870059, 1.0355344843530219)
SHTPI014-GP_TC (0.75026101561533398, 1.1474346088715683)
SHTPI004-GP_PRES (0.4774362936549979, 1.4071111508496499)
SHTPI008-GP_NI (0.27883998571097746, 1.3859856404695314)
SHTPI013-GP_PRES (0.60930002640913117, 1.0747010749530235)
SHTPI005-GP_HUM (0.82379432349193649, 0.85788587030024521)
SHTPI015-GP_CO (0.21585049904225262, 1.2454842901862562)
SHTPI011-GP_HUM (0.59695292666015176, 1.0722011921664203)
SHTPI006-GP_NI 0
SHTPI008-GP_O2 (0.31704405033157357, 1.3777870081865951)
SHTPI012-GP_NI 0
SHTPI016-GP_PRES (0.83951240332810362, 0.94208092884986527)
SHTPI009-GP_HCN 0
SHTPI001-GP_HCN (0.23031932905436575, 0.29203094114947797)
SHTPI005-GP_TC (0.78027149260138162, 0.81812826070318878)
SHTPI005-GP_HCN 0
SHTPI007-BAT (0.25950914232245681, 1.1002293946356478)
SHTPI007-GP_HUM (0.73441717451311839, 1.137733364134728)
SHTPI006-GP_HCN (0.74517952360276063, 0.78962600975050623)
SHTPI004-GP_TC (0.36477992570699802, 1.4699466558399528)
SHTPI012-GP_HCN (0.076721913698214711, 0.14142582871701256)
SHTPI012-GP_PRES (0.79109821631915345, 0.82925441593825588)
SHTPI016-GP_TC (0.89052387730435778, 0.97259414910520725)
SHTPI008-GP_CO (0.29397721118132336, 1.3828693307687678)
SHTPI005-GP_O2 (0.58395945170857722, 0.63391847046783922)
```

```
SHTPI017-GP_PRES (0.13096878594363617, 0.1570377321189633)
SHTPI018-GP_CO (0.43661021916828358, 1.0814216525143965)
SHTPI015-GP_HUM (0.55389904796717671, 1.2323135183323037)
SHTPI004-GP_HUM (-0.058366255369664555, 1.6403453022576557)
SHTPI015-GP_NI 0
SHTPI010-GP_NI 0
SHTPI015-GP_PRES (0.13268762711852311, 1.2338534257907596)
SHTPI005-GP_NI 0
SHTPI009-BAT (0.93103576898558882, 0.95475020245801567)
SHTPI012-GP_HUM (0.8048951209413675, 0.84183574126596827)
SHTPI018-GP_TC (0.56198512199285267, 1.1074032632274033)
SHTPI003-GP_O2 (0.095029929326116203, 1.5901972987477562)
SHTPI013-GP_CO (0.58839147892450649, 1.0703653229692636)
SHTPI013-GP_HUM (0.51552831556710821, 1.0515526620815649)
SHTPI007-GP_TC (0.66727746589624626, 1.1482218055562381)
SHTPI018-GP_PRES (0.43066090704693233, 1.0798111160857142)
SHTPI006-GP_O2 (0.41121914336659576, 0.47339106623890054)
SHTPI013-GP_NI 0
```

출력

- 각 센서데이터 별로 마할라노비스 거리가 나오고 값이 3이하면 정상, 3이상이면 비정상의 정보를 출력함

소스코드

```
# read csv file and save to 2 demension list
import csv
import numpy as np
import scipy.stats as st
#import matplotlib.pyplot as plt
csv_file = open("input.csv", 'r')
reader = csv.reader(csv_file)
mtxData = []
for row in reader:
mtxData.append(row)
# remove first element form mtxData because it is unnecessary.
#
newMtxData = []
countData = len(mtxData)
for i in range(0, countData):
row = mtxData[i]
pureRow = row[1:4]
newMtxData.append(pureRow)
sensorData = []
#시간 데이터를 단위 요소로 분해함
dataCount = len(newMtxData)
for i in range(0,dataCount):
sensorName = newMtxData[i][0]
sensorValue = newMtxData[i][1]
if sensorValue == '0.00E+00':
sensorValue = 0
else:
sensorValue = float(sensorValue)
tmpTimes = newMtxData[i][2]
tmpItems = tmpTimes.split()
tmpDate = tmpItems[0]
tmpTime = tmpItems[1]
tmpDateItems = tmpDate.split('-')
tmpTimeItems = tmpTime.split(':')
tmpDateItems.extend(tmpTimeItems)
tmpDateItems.append(sensorValue)
sensorData.append([])
sensorData[i].append(sensorName)
sensorData[i].append(tmpDateItems)
# 센서별로 데이터 수집하기
sensorGroup = []
for i in range(0,dataCount):
sensorGroup.append(sensorData[i][0])
# 센서명 추출
sensorGroup = list(set(sensorGroup))
sensorDataList = []
#각 센서별로 데이터 모으기
sensorListCount = len(sensorGroup)
for i in range(0,sensorListCount):
sensorDataList.append([])
sensorDataList[i].append(sensorGroup[i])
for i in range(0,sensorListCount):
sensorName1 = sensorDataList[i][0]
for j in range(0,dataCount):
sensorName2 = sensorData[j][0]
if sensorName1 == sensorName2:
sensorDataList[i].append(sensorData[j][1])
## 각 센서별로 정규분포를 만들고, ~~거래가 일정 값 이상 되는 범위를 찾는다.
#센서별 평균값을 구하기
sensorStat = []
for i in range(0,sensorListCount):
tmp = sensorDataList[i]
tmpCount = len(tmp)
tmpData = []
for j in range(1,tmpCount):
tmpData.append(tmp[j][6])
sensorStat.append([])
sensorStat[i].append(tmp[0])
sensorStat[i].append(len(tmpData))
sensorStat[i].append(np.mean(tmpData))
sensorStat[i].append(np.std(tmpData))
sensorStat[i].append(tmpData)
# 각 센서 데이터별 마할라노비스 거리 구하기
```

```
#
for i in range(0,sensorListCount):
mahaDist = []
oneSensorMean = sensorStat[i][2]
oneSensorStd = sensorStat[i][3]
oneSensorData = sensorStat[i][-1]
oneSensorDataCount = len(oneSensorData)
# 각 데이터의 거리 구하기
for j in range(0, oneSensorDataCount):
if oneSensorStd != 0:
tmp = abs(oneSensorData[j] - oneSensorMean)/oneSensorStd
else:
tmp = 0
mahaDist.append(tmp)
if np.mean(mahaDist) == 0:
mahaDistInterval = 0
#print(i,sensorStat[i][0])
else:
# 구해낸 마할노비스의 거리 집합에 대해서 신뢰성 95%의 신뢰 구간을 구한다.
mahaDistInterval = st.t.interval(0.95, len(mahaDist)-1, loc = np.mean(mahaDist), scale = st.sem(mahaDist))
sensorStat[i].append(mahaDist)
sensorStat[i].append(mahaDistInterval)
print(sensorStat[i][0],mahaDistInterval)
print('over')
```

가-1. 재난 룰셋 기준값 산정

1) 적정 센서 데이터 설정

개요

- 센서데이터를 상태로 키퍼에 가시화 하기 위해서는 적절한 통계 기준치가 필요함.
- 따라서, 통계기준치를 서치한 결과를 아래 정리하였음.

산소

- 적정 산소 농도 18% 이상 ~ 23.5% 미만(18000 ~ 23500 ppm)
- 18% 미만인 경우 산소결핍 발생(18000ppm)
- 16% 이하 : 빈맥, 빈호흡, 구토 두통 등의 증상 발생(16000ppm)
- 10% 이하 : 의식상실, 경련, 혈압강하, 서맥을 초래(10000ppm)
- 산소의 농도가 30% 이상일 경우 발화가 쉽게 발생
    - 마찰에 의한 정전기 등의 발생으로 화재가 발생할 위험이 높음
- 출처 : 한국산업안전보건공단 밀폐공간 작업안전 6, 7, 9 페이지

일산화탄소

퍼센트	증 상
0~10	몹시 숨이 차다
10~20	앞머리가 당기는 것 같고 약간의 두통
20~30	머리를 두드리는 것 같은 두통
30~40	심한 두통, 구역질, 구토, 허탈에 빠짐
40~50	모든 증상이 증가됨, 맥박이 빠르고 호흡증가
50~70	혼수, 간헐적인 호흡, 사망

- 출처 : 한국산업안전보건공단 밀폐공간 작업안전 35, 36 페이지

시안화수소

- 허용한계 TLV-TWA : 10ppm
- 20ppm 이하 : 두통, 어지럼증, 구토
- 50ppm 이상 : 호흡곤란, 심장 두근거림, 마비, 의식불명, 호흡정지, 사망
- 135ppm : 30분 노출시 사망
- 275ppm : 즉사
- 출처 : http://kisga.or.kr/?action=SiteBoard&iBrdContNo=20&iBrdNo=10&module=Board&sBrdContRe=0&sMode=VIEW_FORM

온도

- 실내 적정온도 15.6 ~ 20
- 공기조화를 실시하는 실내의 권장온도
- 겨울철 18도, 여름철 26도
- 고온
  - 40도 이상일 경우 주의(열경련 발생 가능)
  - 50도 이상일 경우 위험(심장마비 발생 가능)
- 저온
  - 영하 10~25도 이하일 경우 주의(저체온증 발생가능)
  - 영하 25~45도 이하일 경우 위험(동상 위험)
  - 영하 45도 미만(사망 위험)

습도

- 30%미만 , 80%이상 나쁨
- 40% ~ 70% 쾌적

온도/습도 상관관계

- 15℃ → 70%정도
- 18~20℃ → 60%
- 21~23℃ → 50%

- 24℃ 이상 → 40%

기압

- 실내표준기압 1013.25mbar = 1 기압 = 1013.25 hPa = 101325 Pa
- 기 상 청 http://web.kma.go.kr/communication/encyclopedia/list.jsp?encyc_id=683&page=9&schLang=eng&schGroup=&schType=&schText=S
- 실내에서의 기압 변동은 강한 바람의 유입이 있지 않으면 기압의 변동은 거의 없음
-

이산화질소

- 안전 : 0.05ppm 이하
- 주의 : 0.05ppm 초과 ~ 0.1ppm이하
- 위험 : 0.1ppm 초과
- 출처 : http://www.healthyhome.kr/Contents/Sub2_3.aspx

2) 센서 데이터 설정

- 리서치 결과에 의해서 다음과 같이 센서 데이터의 값을 5단계에 의해서 설정함

	안전	주의	위험
산소	18%(18,000ppm) 이상 ~ 23%(23,000ppm) 미만 (적정 산소 농도)	16%(16,000ppm) 초과 ~ 18%(18,000ppm) 미만 (산소결핍 발생)	1%(1,000ppm) 이상 ~ 16%(16,000ppm) 이하 또는 30%(30,000ppm) 이상 (빈맥, 빈호흡, 구토, 두통->더 나아가 의식상실, 경련, 혈압강하, 서맥 초래) (30% 이상인경우 발화 쉽게 발생 - 마찰에 의한 정전기 등)
일산화탄소	0ppm 이하	0ppm 초과 ~ 10ppm 이하	10ppm 초과
시안화수소	10ppm 이하	10ppm 초과 ~ 20ppm 이하	20ppm 초과
온도	여름철 26도 ±2도 겨울철 18도 ±2도	고온 : 40도 이상(열경련 발생 가능) 저온 : 영하 10~25도(저체온증 발생 가능)	고온 : 50도 이상(심장마비 발생 가능) 저온 : 영하25~45도(동상의 위험) 영하 45도 미만(사망의 위험)
습도	40% 이상 ~ 70% 이하	31%~39% 또는 71%~79%	30% 미만 또는 80% 이상
기압	101325pa	실내에서의 기압 변동은 강한 바람의 유입이 있지 않은 이상 거의 없기 때문에 주의 부분의 기준을 정하기 힘듬	실내에서의 기압 변동은 강한 바람의 유입이 있지 않은 이상 거의 없기 때문에 위험 부분의 기준을 정하기 힘듬
이산화질소	0.05ppm 이하	0.05ppm초과 ~ 0.1ppm 이하	0.1ppm 초과

매뉴얼 사용법

- 별첨의 동영상 매뉴얼 파일에는 각 페이지에 기능별 동영상이 들어가 있음
- 사용법은 조회하려는 기능 페이지에서 동영상을 플레이해서 확인하면 됨

01 마스터_시스템온오프기능



02 마스터_평시모드버튼기능



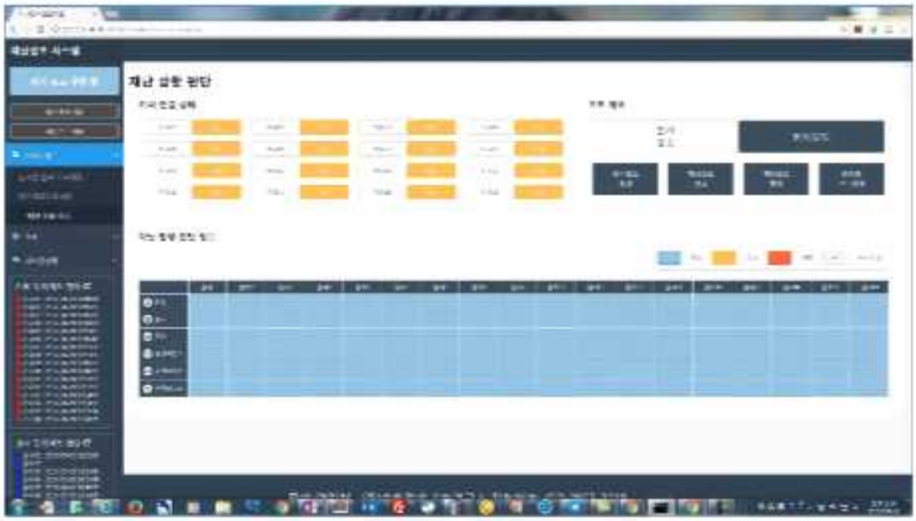
03 마스터_재난모드버튼기능



04 마스터_기기업데이트현황기능



05 마스터_재난발생시_페이지이동_카운트다운



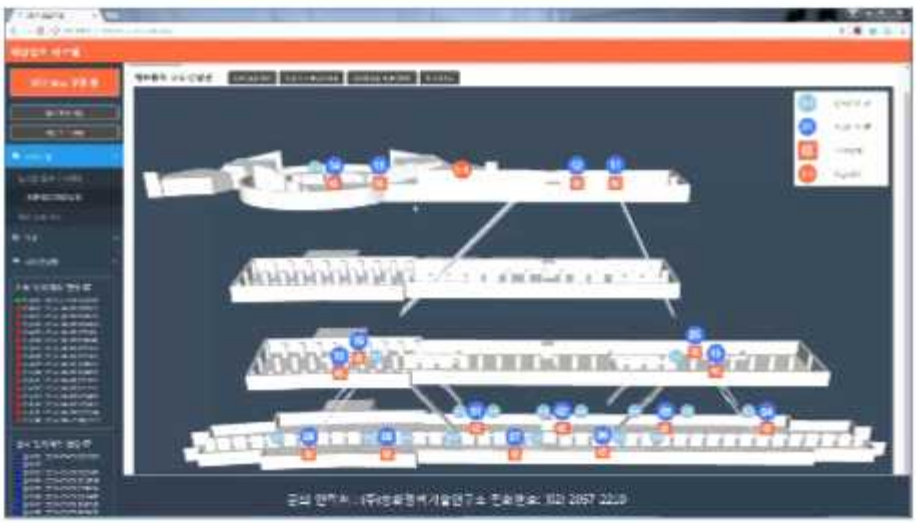
06 마스터_실시간센서모니터링페이지



07 마스터_최초재난센서변경기능



08 마스터_SOP대피경로설정페이지_모달기능



09 마스터_SOP대피설정모드



10 마스터_재난상황판단페이지_모드제어기능



11 마스터_재난발생판단정보스택기능



12 마스터_데이터뷰조회기능



13 마스터_데이터뷰조회리스트삭제기능



14 마스터_재난상황조건설정_룰셋설정기능



15 마스터_재난상황조건설정_센서데이터분석



16 마스터_센서별단계설정_저장기능



17 마스터_기타설정_이상상태설정기능



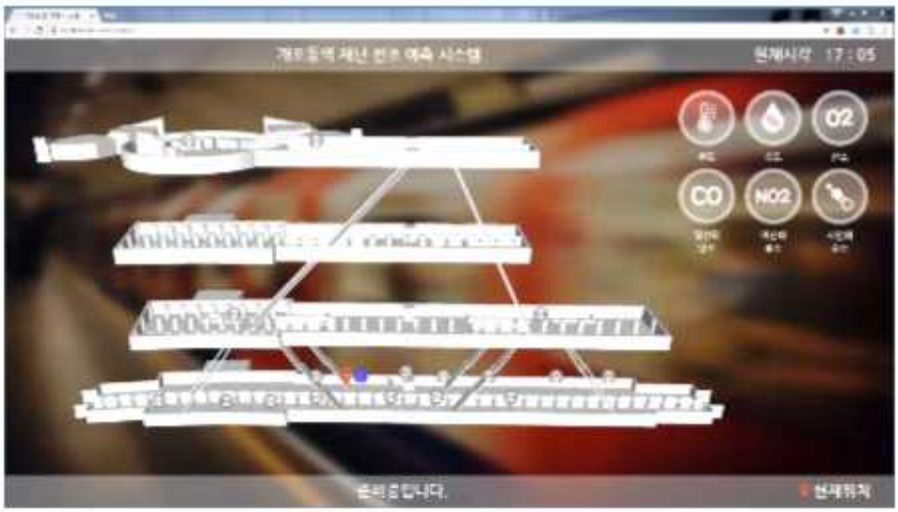
18 마스터_기타설정_키퍼루터설정기능



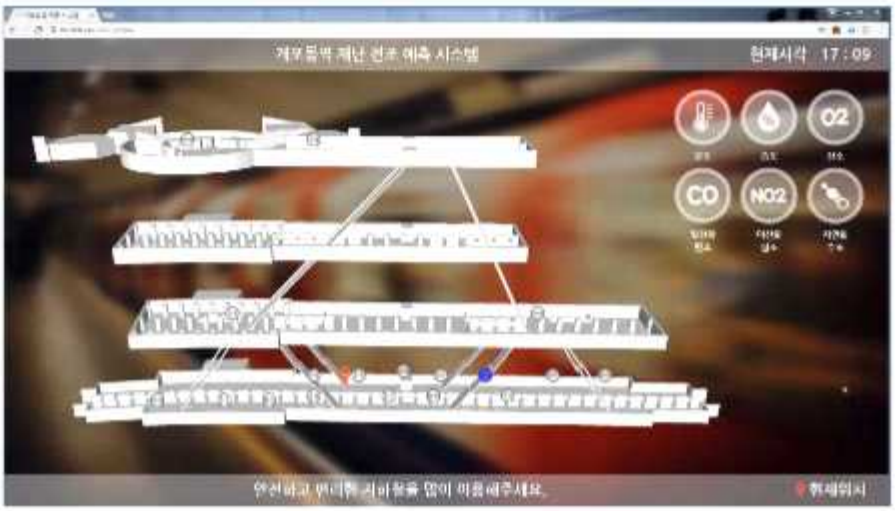
19 마스터_시스템연동현황_설정전파기능



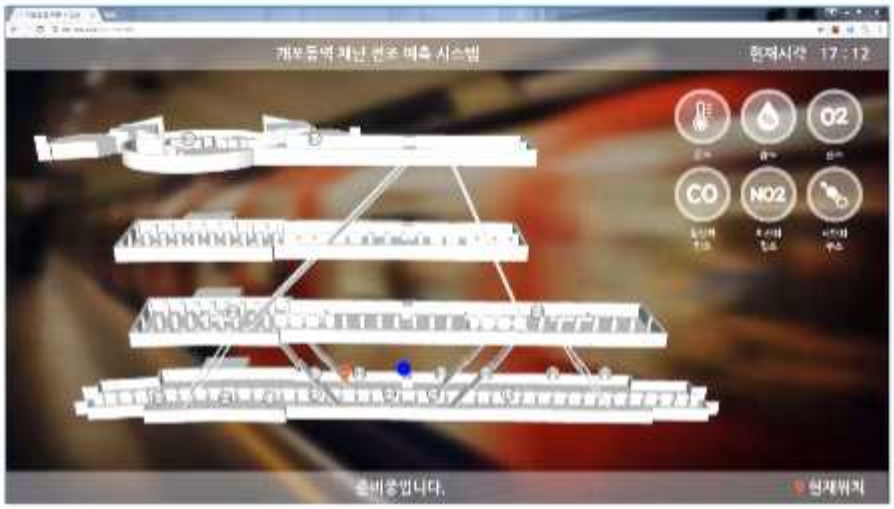
20 키퍼_평시모드_센서모니터링



21 키퍼_평시모드_매뉴얼디스플레이



22 키퍼_재난모드_4분할화면



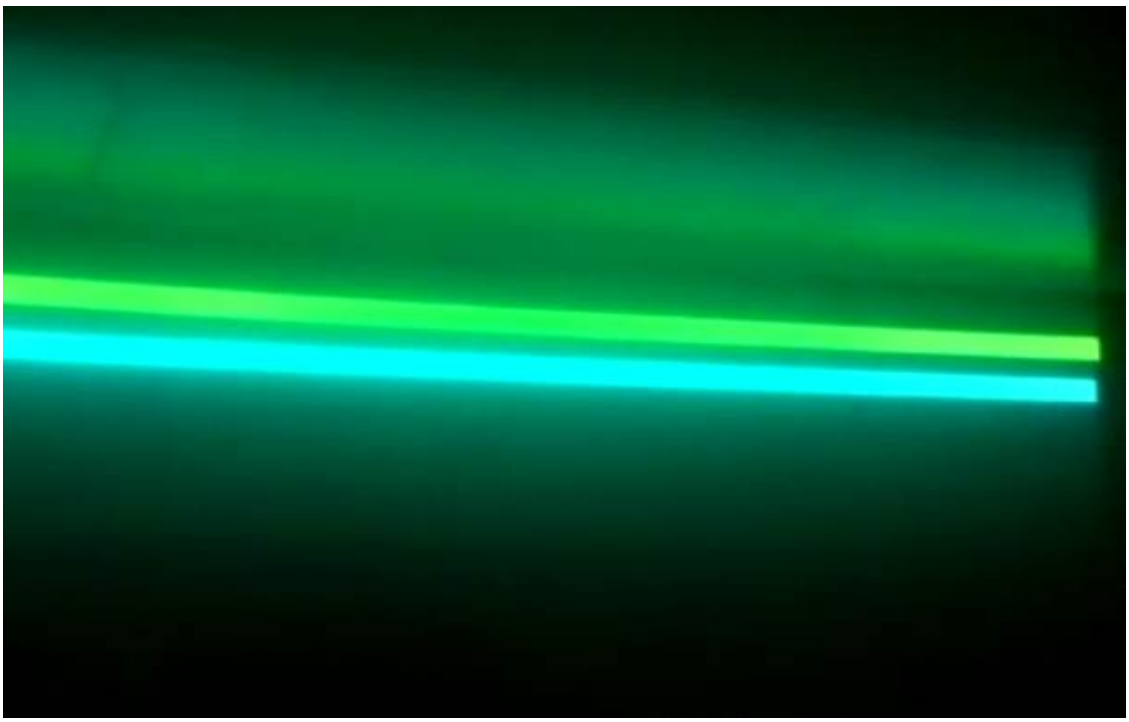
12. 보조 디스플레이 모듈 개발

가. 일반사항

아래 그림과 같이 현재 설치 된 야광타입의 바닥재로는 화재로 인해 연기가 가득찬 지하철역 안에서 식별하기가 어렵고 비상구 표시도 빛의 산란에 의해 식별이 쉽지 않다. 또한 스마트N키퍼 또한 42인치 LCD 모니터를 사용하기 때문에 산란에 의해 알아보기 어려울 수 있으며, 2차년도 전문가 평가에서 승객들의 시야는 아래를 향하고 있는 점과 연기와 암전상황에서 시인성을 확보할 수 있는 보조 디스플레이의 필요성이 대두되었다.
따라서 이러한 화재로 인한 연기와 정전으로 인한 어두운 지하철 내에서 대피경로의 시인성을 확보해 줄 수 있는 EL패널을 활용한 보조 디스플레이 모듈을 개발 및 제작하였다.



지하 구조물인 지하철에서 화재로 인해 연기가 가득차면 시야가 가려져 인간은 지광본능에 의해 밝은 방향으로 도피하려 한다. 그 과정에서 추종 본능으로 인해 화재 발생시 최초로 행동을 개시한 사람을 따라 전체가 움직이는 본능적인 움직임을 보이는데 최초 행동한 사람이 피난유도등을 통해 지광본능으로 피난의 올바른길을 피난 유도가 필요하다.



색의 시인성

색이 알아보기 쉬운 성질을 시인성이라고 한다. 멀리서도 대상의 존재를 쉽게 인식 할수 있는 색일수록 시인성이 높다. 대상의 존재가 인지될 때까지의 시간이나 거리로 그 정도를 나타낸다.



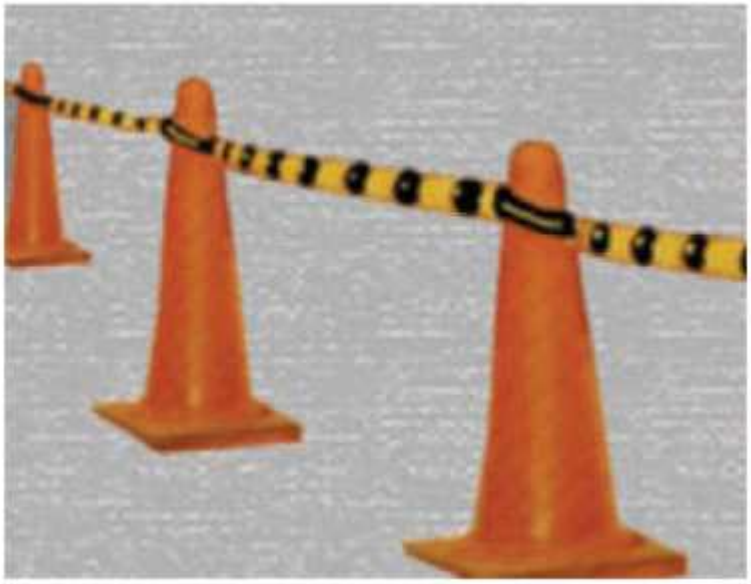
일반적으로 문자나 숫자를 인지하는 '가독성'과 형태 등을 인지하는 '명시성'을 포함한다. 시인성에는 배경색과 대상색과의 명도차이가 가장 큰 영향을 미친다. 배경색과의 명도차이가 크면 클수록 시인성이 높아져, 멀리서도 그 존재를 쉽게 알아 볼 수 있다. 시인성에 색상이나 채도는 그다지 큰 영향을 미치지 않는다. 색상이나 채도의 차이를 크게 두어도 명도차이가 없으면 시인성은 높아지지 않는다.

명도차이를 크게 둔 검정과 흰색이나 노란색의 조합은 시인성이 높으나, 흔히 눈에 잘 뛰는 색이라고 하는 노란색도 흰색과 조합하면 명도차이가 작아 시인성이 매우 낮아지는 것을 알수 있다. 또 색상차이는 크게 두었으나 명도의 차이가 작은 빨간색과 녹색의 조합 및 채도차이는 크게 두었으나 명도의 차이가 작은 회색과 파산색의 조합은 명도 차이를 크게 둔 경우만큼 시인성이 높아지지 않는다는 것을 알 수 있다.

색의 유목성

색에는 눈길을 끌기 쉬운 색과 그렇지 않은 색이 있다. 눈길을 끌기 쉬운 색을 유목성이 높은 색이라고 한다. 유목성은 사람의 심리적인 영향도 있어, 익숙하지 않은 색이나 반대로 너무 익숙한 색, 좋다고 느끼는 색에 대해서는 눈길이 끌리기 쉽지만 일반적으로 명도 단계에서는 고명도, 채도 단계에서는 난색계의 고채도색, 예를 들면 선명한 빨강이나 오렌지, 노랑 등이 사람의 눈길을 끌기 쉬운 유목성이 높은 색이다. 위험을 알리는 표시나 간판, 공업제품의 중요한 스위치 등에 이러한 색들이 많이 사용된다. 단 시인성과 마

찬가지로 배경색과의 색상차이가 작은 색은 유목성도 떨어지게 되므로 고려하지 않으면 안 된다.



도로안내표시에 사용되고 있는 선명한 주황색은 유목성이 높아 멀리서도 쉽게 인지할 수 있다.

일반 채색과 EL 디스플레이 비교



[형광등 간판과 EL 디스플레이 밝기 비율]



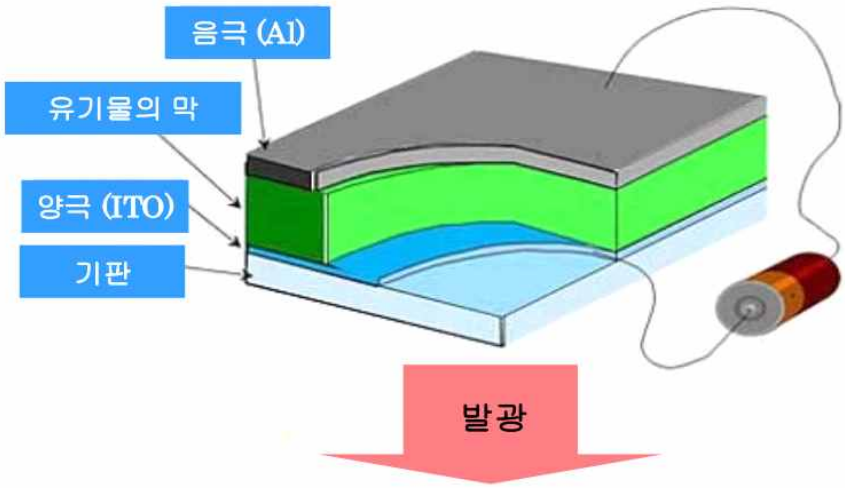
[밝은 곳에서 EL 디스플레이 밝기 비율]

나. EL Display 종류

6) 유기 EL

○ 유기 EL은 발광층이 2개의 전극 사이에 끼어 있는 sandwich 구조이다. 그러나 발광층에서 나오는 광을 밖으로 내보내기 위해서는 한 쪽 전극에 투명한 재료를 사용해야 한다. 일반적으로 양극(anode)에는 ITO(indium tinoxide)럴, 음극에는 Al 등을 사용하고 있다.

○ 유기 EL display에 사용되는 유기분자는 저분자와 고분자 2가지로 나눌수 있으나 처음으로 발광원리가 발견된 것은 저분자였다. 따라서 나중 에 발견된 고분자보다 저분자인 유기 EL에 대한 연구가 먼저 추진되었다. 그러나 최근에는 고분자에 대한 연구개발이 활발하게 진행되어 저분자, 고분자 사이에 우열을 가리지 못하게 되고 있다.





○ 유기 EL display는 EL(electroluminescence)이라 하여 거의 발열하지 않고 전기를 광으로 변환하는 현상을 이용하고 있다. LED는 p형 반도체와 n형 반도체를 접합하여 전기를 운반하는 carrier가 존재하고 있으나 유기 EL의 유기분자에는 carrier가 거의 존재하지 않는다. 따라서 발광시키기 위해서는 양극과 음극에 각각 정공과 전자를 주입해야 한다. 이때 유의할 점은 높은 휘도, 높은 효율의 소자에는 발광층에 p와 n형의 carrier를 균형 있게 주입하는 것이 중요하다.

○ 발광할 유기층은 단층에서 다층의 것까지 있으며 경우에 따라서는 각 층의 역할이 다르다. 또한 발광층에 이용되는 유기분자는 저분자와 고분자 2가지로 대별된다. 유기층 전체는 수백 nm 이하로서 대단히 얇은 것이 특징이다.

- 저분자와 고분자의 발광 mechanism 등은 그리 큰 차이는 없다. 오히려 상품적인 측면, 제조방법 등에서 차이가 난다.

	저분자	고분자
ETL(elctron transport layer)	Alq3	-
EML(emissive layer)	Alq3NPB	PPV
HTL(hole transfer layer)		PEDOT

[유기 EL 재료의 예]

주: Alq3는 aluminato-tris-8-hydroxyquinolate

NPB는 N,N-di(naphthalene-1-yl)-N, N-diphenyl-benzidene

PPV는 poly(phenylene vinylene)

PEDOT는 poly(ethylenedioxy)thiophene

- 저분자의 경우에는 분자를 진공상태에서 승화시켜 유리기판에 막을 증착시키는 방법으로 제조하고 있다. 이때의 온도는 분자가 기체가 되도록 고온이라야 하지만 유기분자가 분해되는 고온(250-450°C)은 피해야한다. 그 후에 냉각한 유리기판에 막을 형성하지만 이는 단분자막 level로서 정확히 제어할 필요가 있다. 또한 증착할 때에는 유리기판과 금속 mask 사이에 발생하는 열팽창율의 차이로 인해 대형이 될수록 성막에얼룩이가 발생하기 쉬워 저분자 유기 EL에서는 대형화가 어려운 것으로 알려져 있다.
- 한편 고분자의 경우에는 액체에 녹일 수가 있다. 그 때문에 roll과 roll 사이의 sheet를 통해 coating할 수 있어 제조원가를 낮출 수 있다. 또한 유리 이외에도 plastic sheet를 기판으로 할 수 있어 flexible display도 가능하다. 그 이외에도 고분자 제조방법에는 ink-jet printer 법 등이 있다.

○ 유기 EL 디스플레이는 자발광형의 FPD로서 광시야각, 고속응답, 박형경량, 고정밀도와 원리상 압도적으로 뛰어난 성능을 가지고 있지만, 성장이 현저한 LCD와 본격적으로 경합 하려면 해결해야 과제도 적지 않다. 특히 유기 EL 디스플레이의 휘도수명 특성은 유기 EL 디스플레이 사업의 운명을 좌우하는 최대의 핵심기술로 자리잡고 있다.
유기 EL 디스플레이의 휘도수명 특성은 유기 EL 재료, 유기 EL 박막적층 구조, 유기 EL 제조 프로세스에 크게 의존하기 때문에, 이들을 종합적으로 최적화하는 높은 기술력이 필요하게 된다.

가) Active와 Passive 구 조

○ 패시브 타입은 표시영역을 양극과 음극에 의한 단순 매트릭스로 구성되어 있고, 음극과 양극이 교차되는 부분이 발광 가능함. Row 라인, 즉 음극이 선택되었을 때에만 점등하는 Dut y 구동이며, 또한 구동용 드라이버IC는 TCP나 COG에 의해 外裝할 필요가 있음.
○ 이에 대해, 액티브 매트릭스는 표시영역의 각 화소에 스위칭용 TFT를 배치하고, 非선택시에도 점등 가능한 St a t ic 구동이 가능함. 유기 EL은 전류구동이기 때문에 비교적 커다란 전류를 흘릴 수 있는 TFT가 필요하기 때문에, 액티브 타입에는 이동도가 높은 저온 p- S i TFT기술이 채용되고 있음. 저온 p- S i TFT는 값싼 유리기판으로 만들 수 있고, 또 주변 드라이브회로를 내장하는 것도 가능하기 때문에, 콤팩트한 디스플레이 제작이 가능함.

나) 두 타 입의 비교

○ 유기 EL 소자는 응답성이 높고, 잔광특성이 없기 때문에, 패시브 타입과 같은 Dut y 구동 디스플레이에서는 필요한 휘도의 Row 라인 몇 배가1개의 Row라인에 필요한 휘도가 됨 . 예를 들면, 디스플레이 평균 휘도가 200cd/ m2, Row 라인수 480개라 하면, 각 화소에는 200cd/ m2×280개=96000cd/ m2라는 고휘도가 필요하게 되어, 유기 EL 재료의 수명을 현저히 저하시킴과 더불어 구동전압 · 전류의 상승을 초래하여, 그에 따른 배선의 전압강하 문제로 , 고휘도 · 고정밀화에는 제약이 발생함.
○ 또한, 이 고휘도는 플리커 주파수의 상승을 초래할 것으로 생각됨 . 통상의 TV 신호인 경우에는 필터 주파수는 60Hz 인데, 이 순간적인 고휘도에 의해 필드 주파수를 올리기 위한 변환회로가 필요하게 됨 .

○ 현재 상황으로서는 가능한 한 필요화소 휘도를 낮게 억제하기 위해, 화면을 상하로 분할 구동함으로써 Row 라인수 240개 정도의 디스플레이가 실현되고 있음.

○ 이에 대해, 액티브 타입은 상시 발광하고 있기 때문에, 이와 같은 제약은 생기지 않음. 저전압구동이 가능하고, 저소비전력화로 유리하며 또한 위에서 기술한 것과 같이 저온 p-Si TFT 기술을 이용하여 주변 드라이버를 내장함으로써 , 좁은 틀을 가능하게 하여 소형화에 유리함.

7) 유기 EL 디스플레이의 장점

유기 EL 디스플레이는 LCD와 비교하여 아래와 같은 우수한 특징을 가지고 있음.

- 고휘도, 자기발광
- 밝고, 선명한 표시
- 넓은 시야각
- 백라이트가 불필요함에 따른 저소비전력
- 백라이트가 불필요함에 따른 박형 · 경량화
- 직류 저전압구동
- 잡음없음, 저소비전력
- 고응답속도
- 원활한 동화상 재생
- 유기막에 의한 발광
- 넓은 사용온도 범위

3) 유기 EL의 어플리케이션

○ 유기 EL 디스플레이는 LCD와 비교한 이들 우위성은 앞으로 평면 디스플레이 시장에 참여해 나아갈 유기 EL 디스플레이에 있어 커다란 경쟁력이 되는데, 이 우위성을 어떠한 용도에 응용하여 살려 나가는가 하는 것이 매우 커다란 과제임.

○ 패시브타입은 소형 · 저정밀분야라면 싼값으로 제조할 수 있고, 유기EL 디스플레이의 특징을 살린 응용(휴대전화 , PDA, 차량용 오디오 디스플레이 등) 이 가능할 것으로 보임. 또한 액티브타입은 고화질 · 고정밀, 박형 · 경량으로 DSC, DVC, 차량 자동항법장치용 디스플레이, PC용모니터, 디지털 TV 등에 응용이 가능할 것으로 보임.

다. 대피경로 표시에 있어 EL Display의 특징 (LED와의 차별성)

○ 발광효율이 낮고 수명이 짧은 백열전구와 신경계통에 해를 주는 수은(Hg)을 사용하는 형광램프는 '발광다이오드(LED, Light Emitting Diode)'와 면 전체가 발광하는 '유기EL(OLED,Organic LED)' 등 수명이 길고 전력소비가 적은 조명기기로 착실히 바뀌고있다.

○ 피난구유도등의 경우도 형광등 back-light를 사용하다가 LED back-light 를 사용한 제품이 출시되어 좀 더 저전력으로 표시를 하고 있다.

피난구유도등의 예



LED가 발광효율에서 백열전구나 형광램프를 상회하고 일반적으로 조명기구에서 나오는 광은 사람 눈에 보이는 광뿐 아니라 사람눈에 보이지 않는 자외선도 섞여 있다. 그러나 LED는 가시광선 범위 내로서 열선이나 자외선을 거의 방출하지 않는다
조명기구로서 LED와 유기 EL의 장점을 들어 보면 다음과 같다.

○ LED
장점
- 백열등이나 형광램프에 비해 수명이 훨씬 길다.
- 시인성(視認性)이 좋아 옥내외를 막론하고 폭넓게 사용할 수 있다.
- 기구를 소형화할 수 있어 조명기구로서 설계가 자유롭다.
- 작은 전력(LED는 같은 밝기의 백열전구와 비해 소비전력이 약 86%절감)으로서도 점등이 가능하고 환경에 친밀하다.
- 자외선을 거의 방출하지 않아 방충효과가 크고 조광이 자유롭다.
단점
- LED는 점광원(点光源)이기 때문에 눈이 부시고 조명 design의 자유도가 적다는 제약이 있다.

○ 유기 EL
- 면의 전체가 발광하기 때문에 눈부심이 없다.
- 면의 모양을 자유롭게 선택할 수 있어 design의 자유도가 많다.
- 유기물 분자구조의 조합이 무한해 이에 따라 발색을 달리할 수 있다.
- 매우 가볍고 얇으며(발광층의 두께가 0.1μm 이하)로서 발열량이 미소하여 광을 확산하기 위한 기구가 불필요하여 poster와 같은 조명에도 적합하다.
- 발광면을 곡면으로 만들 수 있어 모양이 flexible하게 된다.
단점 유기 EL은 “Edison 이래의 발명”이라 할 정도로 혁신성이 큰 반면 현재로서는 발광효율이 낮으며 휘도를 높이면 수명이 짧아진다.

○ LED를 대신하여 EL 디스플레이를 사용하여 피난구유도등을 만들 경우
- LED는 간접조명이고 EL은 직접 조명이다.
- LED는 점광원이고 밝기때문에 연기속에서 빛의 산란이 심하여 시인성이 떨어진다. EL의 경우 휘도가 낮아서 빛의 산란 현상이 심하지 않아 글자나 모양의 식별이 LED에 비해 우수하다.
- LED조명 기구가 아닌 벽면이나 바닥에 설치할 경우 제작이 힘들지만 EL디스플레

이의 경우 곡면으로 제작이 가능하다

- LED 부품도 얇게 제작이 가능하나 EL의 경우 두께가 훨씬 얇기 때문에 여러 구조물에 표시용으로 제작이 가능하다.

- LED의 경우 Driver가 꾸준히 개발되어 회로 개발을 다양한 방법으로 개발이 가능하다. 하지만 EL의 경우 Driver의 개발이 아직 초기이고 구동전압이 AC100V인 단점이 있어 Driver개발이 어려운 단점이 있다.

다. 부품별 기능

1) 전원부

일반 산업용 SMPS를 사용하고 있으며, 부하에 맞춰 용량을 선택해야한다. 1차 디자인 시안 규격으로 Power supply 용량을 산정하였다. DC12V를 선정하였다. 하지만 새로운 2차 디자인 시안이 나오면서 Power supply 규격을 DC15V/350W로 선정하였다.

첫번째로는 DC12V를 사용할 경우 전선의 굵기가 커져서 실제 배선을 했을 경우 안전에 문제가 있다는 제조사의 의견이 반영되어 DC15V/350W를 선택하였다.

두번째는 용량이 큰 인버터의 경우 15V로 제작된 것을 사용하기 위해 제조업체 의견을 반영하여 DC15V/350W로 변경하여 진행하였다.

SMPS

출력 DC15V/350W

입력 110V/220V 60hrz



2) 인버터부

SMPS 를 전원으로 사용하며 , 트랜스포머를 활용하여 전압을 상승시킨다 .

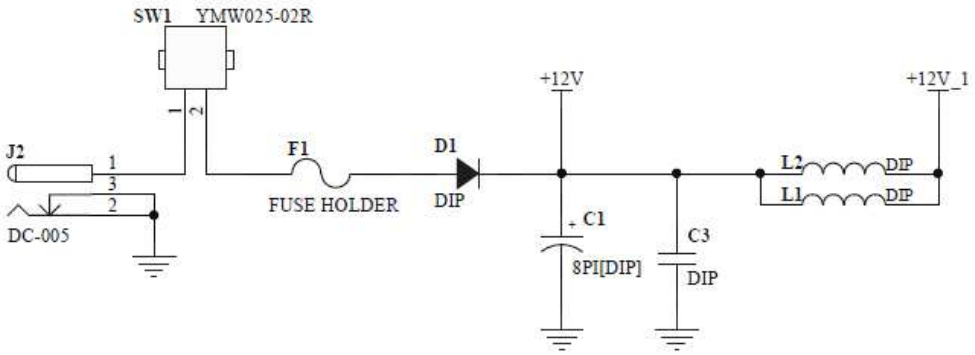
자력발진을 통하여 주파수를 만들어준다 . 이 때 부하의 용량이 주파수에 영향을 주게 되므로 개발시 주의가 필요하다.



① 인버터 전원 입력부

○ DC JACK 을 이용하여 전원입력을 시키며 , SW1 의 ON/OFF 스위치를 구비해 필요시에는 OFF시킬 수 있도록한다.

○ 시스템의 안전을 위하여 Fuse 를 사용하며 일정전류 이상의 이상전류가 흐르지 못하도록 해준다.



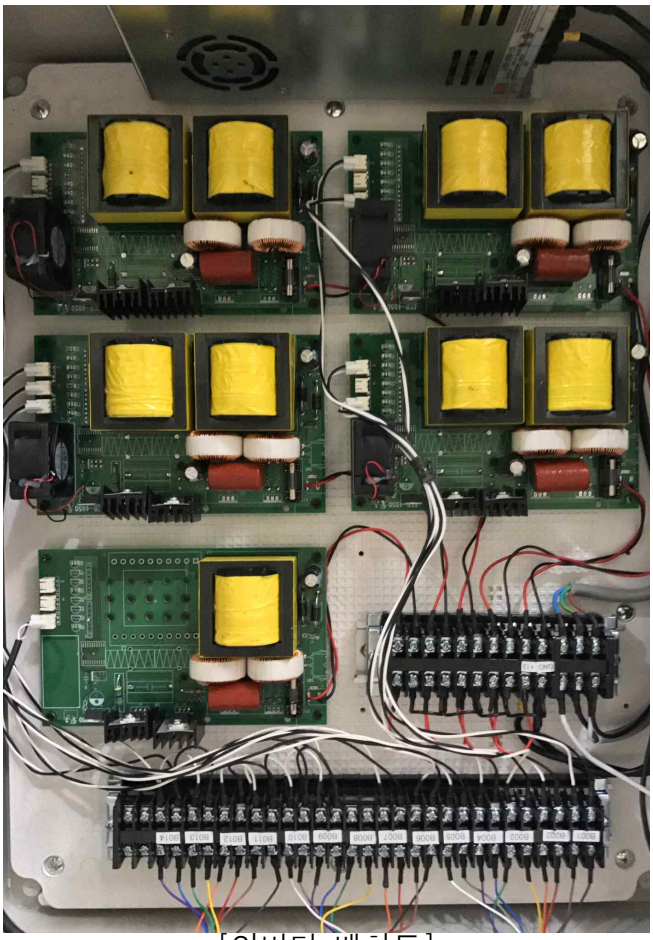
[인버터 전원 입력]

○ 스마트인버터

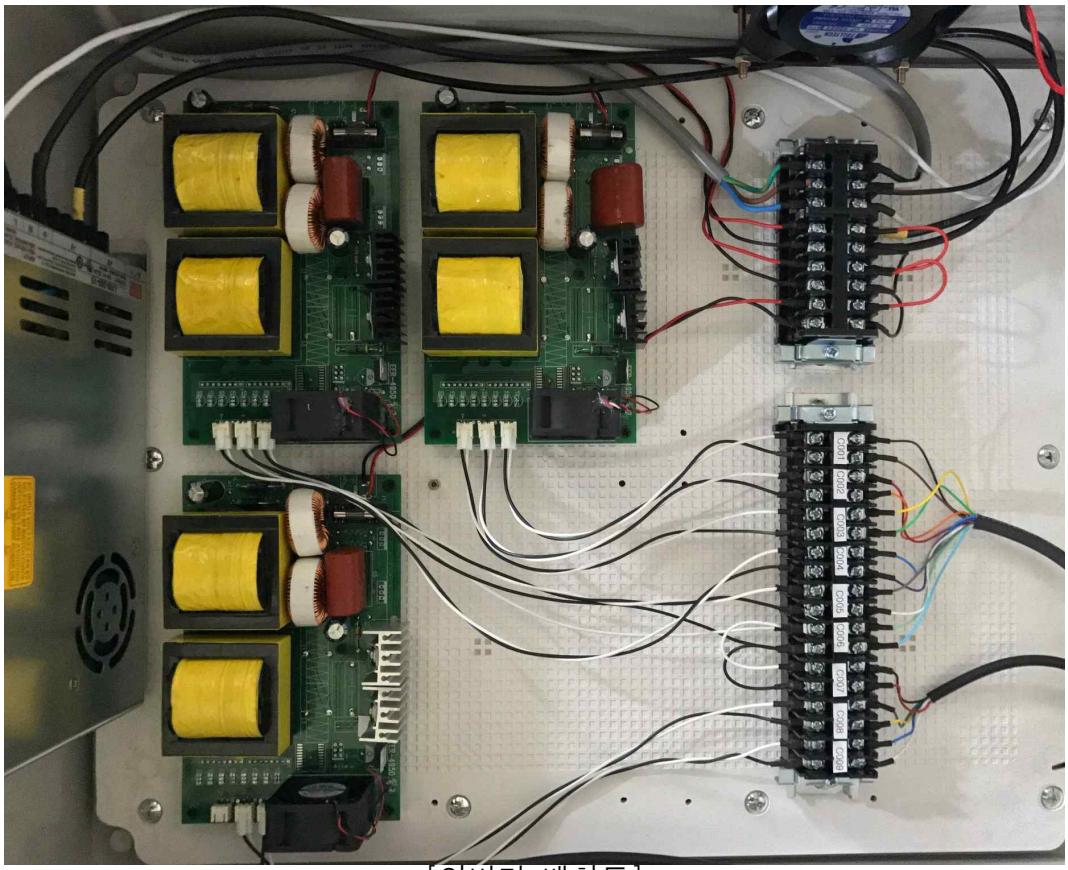
○ 출력 120V / 500hrz

○ 입력 DC15V

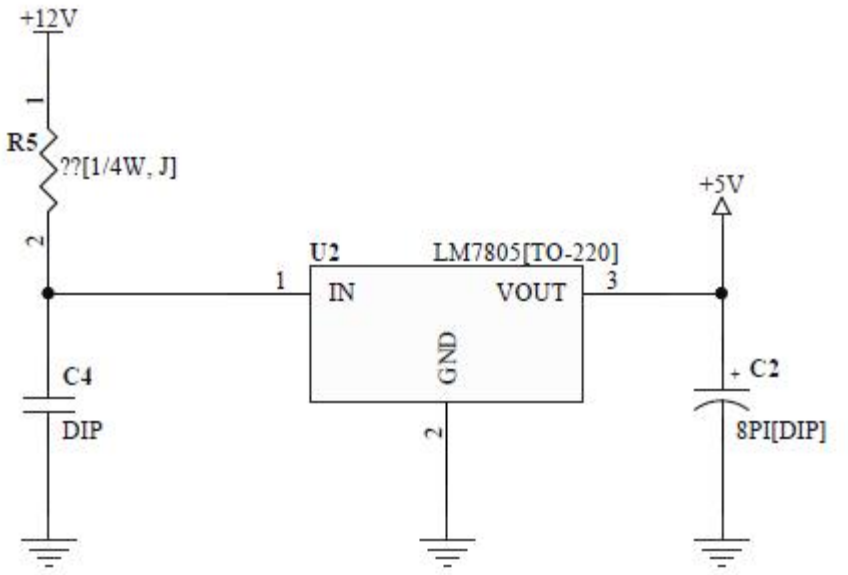
○ 각 50W



[인버터 배치도]



[인버터 배치도]

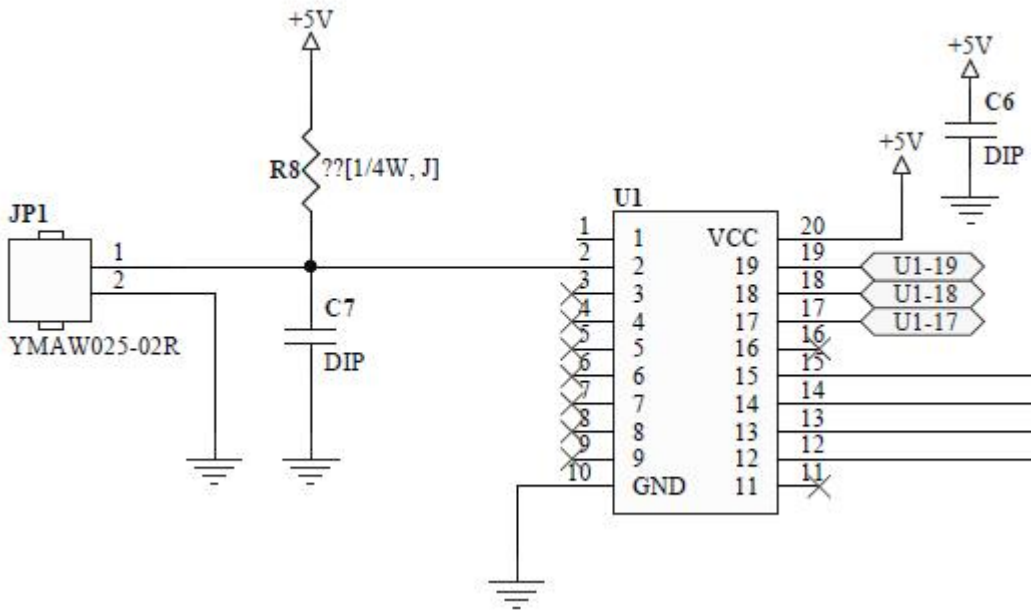


[디지털 전원부]

디지털 회로의 전원을 위하여 DC5V 를 생성한다.

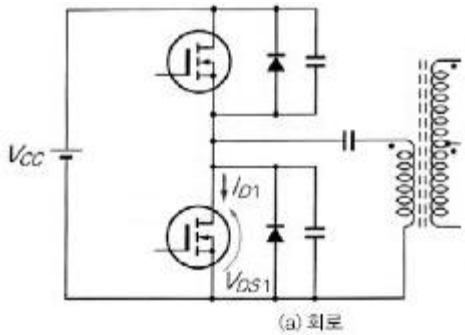
② 채널 제어부

마이크로프로세서 (MG86FE508) 를 이용하여 각 채널의 ON/OFF 를 제어하도록 한다 . 본 연구에서는 RF 수신모듈의 데이터를 프로세서가 분석하여 제어하도록 하였다 .



[채널제어부]

③ 공진형 로이어 컨버터

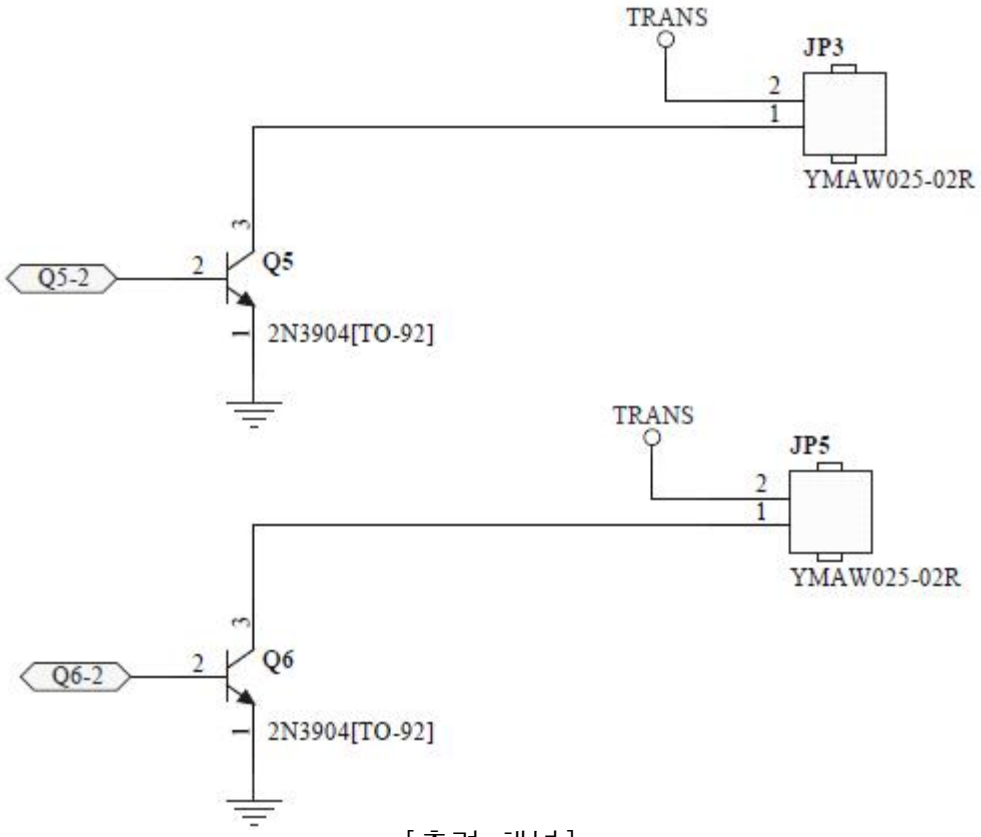


[공진형 로이어 컨버터]

DC12V~15V 전압을 인가하여 공진형 로이어 컨버터를 통해 EL 에 인가될 전원을 생성시킨다 .

④ 출력 채널 구동

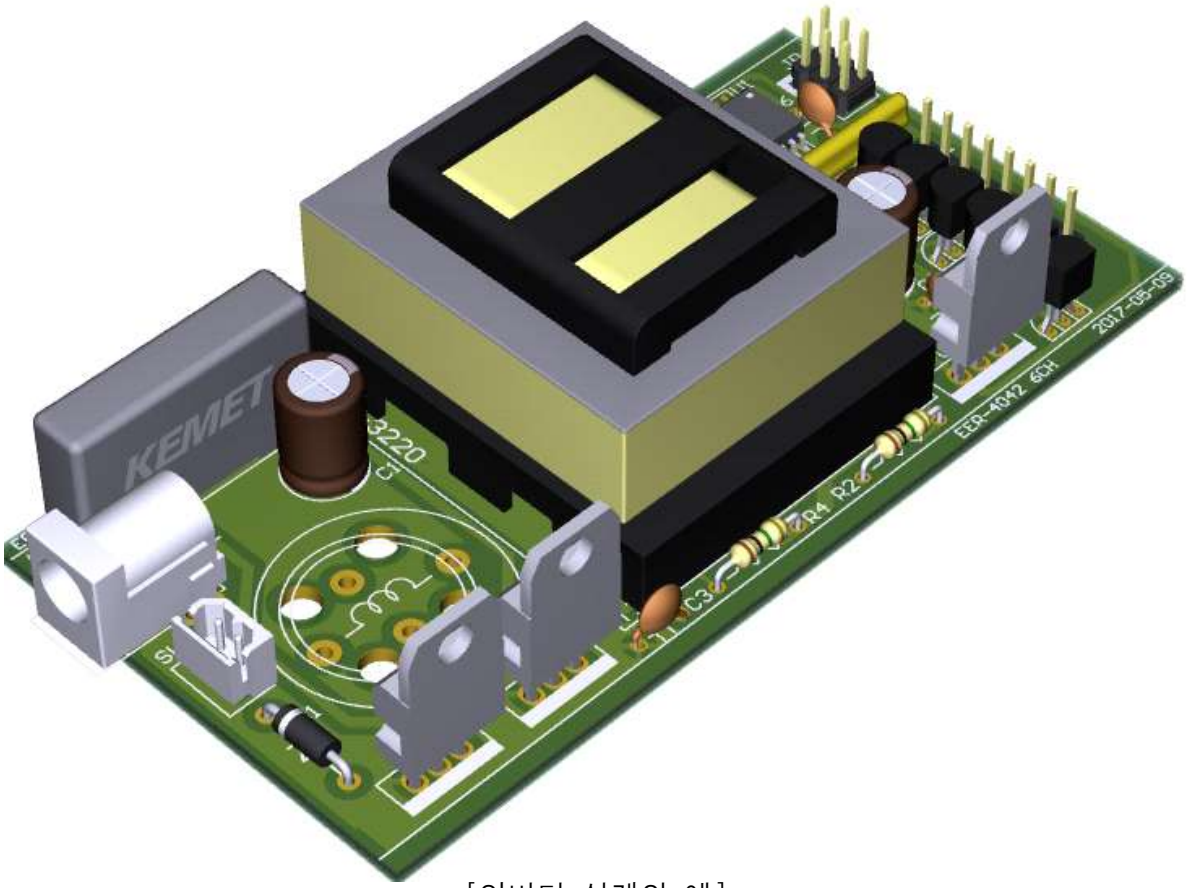
트랜스포머를 통해 발생된 파형을 각 EL에 인가를 시키고 , 트랜지스터를 ON/OFF 시킴으로써 발광이 될 수 있도록 한다.



[출력 채널]

⑤ 인버터 설계의 예

위의 그림과 같이 인버터는 트랜스포머와 인덕터 및 캐패시터를 이용한 자려발진 회로들로 구성이 된다.



[인버터 설계의 예]

3) 통신모듈



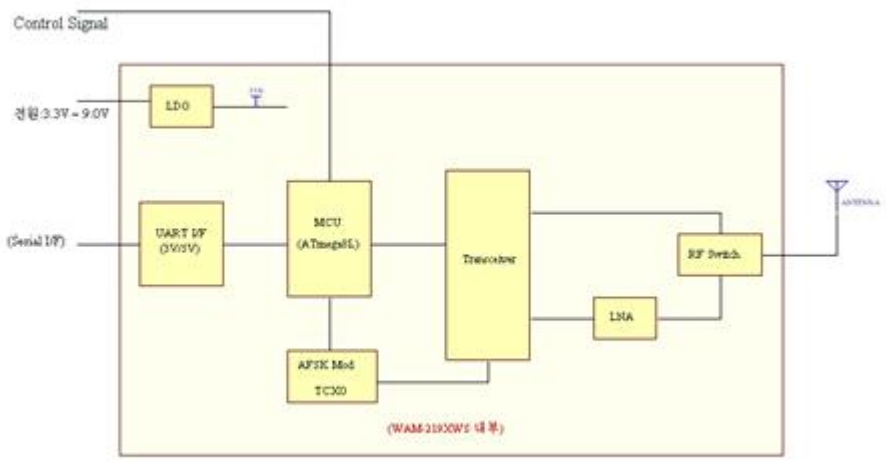
[통신을 위한 RF 모듈]

① 통신 모듈의 특징

- 데이터전송용특정소출력무선기기 VHF(219MHz) 대역 10 채널사용
- Digital AFSK(MSK) 변복조방식

- 중계기모드
- 1:1, 1:N, N:N 네트워크구성가능
- Master/Slave 모드
- Custom Cod e 부여방식 - 혼신방지 , 데이터보안이나안전시스템에활용
- HDLC 프로토콜내장
- UART 통신 3V/5V I/F 가능 , 1200 ~ 38400bps
- TCXO(온도보상크리스탈) 사용
- 모듈전원 On/Off 단자지원 자동모드전환기능동작모드 (상태) 확인용 LED: 2EA 장착
- Narrow Band (Occupied bandwidth : < 8.5KHz, 변조 < ±2.5KHz) 송신버퍼 : 512Byte,
- 수신버퍼: 256Byte
- 단일 3.3V~ 9.0V 전원
- 송신모드 : < 50mA 수신모드 : < 40mA
- Half Duplex Communication
- RF Data Rate : 1200 bps
- Dimension: 33.02mm x 58.42mm (connector 제외)
- 송신출력: 10mW (10dBm)
- 최대출력: 20mW(13dBm)
- TX : Deviation: < ±2.5kHz 인접채널전력 (ACP): > 50dBc
- TX : 2 차 , 3 차하모닉 : > 50dBc
- @219.0000MHz
- @219.0000MHz
- RX : 에러검출 / 보정기능내장 (출력 2 배증가효과)
- RX : 수신감도 : -125dBm (Typ)
- RX : RSSI 출력지원
- 용도 : 원격검침 및 제어 , 방범 및 화재 경보시스템 , 출입문관리 / 통제

② 블럭도 및 핀 설명



[통신 모듈 블록도]

WAM-219XWS 모델의권장사용전원은 3.3V~9.0V(Low-Dropout Regulator 가내장) 이며 , 시리얼데이터통신 (UART) 은 3V 나 5V 통신이가능하도록되어있고 , 데이터처리및 제어용 MCU, 무선송 / 수신 IC, 저잡음증폭기 LNA 와 RF Switch 등으로구성이되어있다 .

No	Name	I/O	Description	연결	
				필수	선택
1	BS2	I[1]	시리얼 Data Rate 설정		
2	BS1	I[1]	시리얼 Data Rate 설정		
3	RxD	I	UART 통신 관련		
4	TxD	O	UART 통신 관련		
5	TxPU	I	UART 통신 관련		
6	RSSI	O	수신모드시 RSSI출력		
7	bRst	I	Reset(Low Active)신호 *3.0V 100Kohm Pull-Up 이 되어 있다.		
8	GND	Ground	Ground		
9	V+	Power	3.3V~9.0V Power(내부LDO)		
10	On/Off	I	모듈전원On/Off제어신호 *V+에 4.7Kohm Pull-Up이 되어 있다.		
11	NC		No Internal connection		
12	NC		No Internal connection		
13	EEPR/DIP	I[1]			
14	CS1	I[1]	RF 채널 설정용		
15	CS2	I[1]	RF 채널 설정용		
16	CS3	I[1]	RF 채널 설정용		
17	CS4	I[1]	RF 채널 설정용		
18	bCfg	I	1) GND 연결→ 설정모드 동작 2) Open → 송/수신 모드 동작 * 3.0V에 100Kohm Pull-Up 이 되어 있다.		
19	RfRxd(test)	O	테스트 용도로 사용		
20	RfClk(test)	O	테스트 용도로 사용		
※ 사용하지 않는 핀은 Open 상태로 놓으면 된다 . ※ 전원 (2 line) 과 UART 통신 (3 line) 연결만으로 데이터 통신이 가능하다 . ※ (1): 모듈 내부의 전원 (3.0V) 에 20~50Kohm 의 Pull-Up 이 되어 있다 . ※ (2) : 중계기로 이용 시 전원 (2 Line) 연결만으로 사용 가능					

전원입력 (V+) 은 3.3V~9.0V 사용을 권장한다 . 모듈 내부에서는 3.0V 출력의 Low-Dropout Regulator LP2985 를 사용하여 모듈 내에서는 3.0V 를 사용하고 있다 . Pin10(On/Off) 는 LDO 의 출력을 제어하는 신호로 모듈 내에서 R1(4.7Kohm) 이 V+ 에 연결이 되어 있다 . On/Off 핀 (Turn-On Time: <40ms) 을 사용하여 외부에서 전원을 On/Off 시키는 용도로 사용이 가능하며 , 건전지를 사용하는 Application 에서 유용하게 사용 할 수 있다 .

WAM-219XWS 모듈의 통신 (UART) 라인 TxD 와 RxD 는 3~5V 인터페이스가 가능하도록 설계가 되어 있다 . TxD 출력은 Open Drain 출력에 Pin5(TxPU:3/5.0V) 이 모듈 내부의 Pull-up 저항 (R2:4.7Kohm) 에 연결이 되어 있으며 , R3(510ohm) 저항은 사용자의 실수로 잘못 연결한 경우에 모듈을 보호하기위한 저항이다 . RxD 라인의 최대 입력

전압 (SN74LVC2G07 규격참조)
은 5.5V 이며 , R1 저항 100K 는 RxD 가 open 된 경우 입력이 High 가 되도록 TxPU 단자에 연결이 되어있다.
WAM-219XWS 모듈의 RF Frame 은 시작과 끝이 Flag(0x7E) 로 되어있는 HDLC 프로토콜을사용하고 있으며 , 1:1, 1:N(Master-Slave), N:N 무선네트워크 사용시 RP 값과 DA 값을 고정해서 사용하는 경우에는 Slave 형 (Transparent) 으로 , RP 값과 DA 값을 자주 변경하면서 사용하는 경우에 는 Master 형으로 사용하면 편리하다 . 무선네트워크 구성시 Master 형과 Slave 형을 혼용해서 사용도 가능하고 , Slave 형으로 사용 시 별도의 프로토콜을 사용하지 않아도 사용이 가능하며 , Master 형으로 사용 시는 모듈과 간단한 SLIP(RFC-1055) 프로토콜을 사용해서 통신을 해야 한다.

Flag	Custom Code	RP	DA	User Data	CRC-16	Flag
8 byte (0x7E)	2 byte (0xXXXX)	1 byte	1 byte	1 ~ 128 byte	2 byte	1 ~2 byte (0x7E)

데이터를 , 수신 모듈에서는 1)DA 필드 값이 자신의 ID 와 같은 경우 2)DA 필드 값이 “255”
(Broadi ngAddress) 인 경우에만 수신을 할 수 있다 .
유저 데이터 필드는 사용자가 주고 받는 실제 데이터로 1~128 byte 전송이 된다 . 송 / 수신 데이터의 버퍼는 송신 512Byte, 수신 256Byte 로 되어 있다 . 연속해서 512 Byte 이상의 데이터를 전송하면 버퍼에 모두 넣을 수 없는 경우에는 뒷 부분 데이터는 전송하지 못하는 경우가 발생할 수 있다 . 실제 사용 가능 최대 데이터 사이즈는 480 ~ 3000Byte 이다 . 왜냐하면 데이터가 입력되는 동안에 RF 출력으로 데이터가 출력되고 버퍼가 비워지게 되어있어 UART 속도에 따라서 실제 전송 가능 최대 길이는 차이가 있다 . 중계기를 사용하는 경우에는 데이터 사이즈를 128Byte 이하로 사용해야 한다.
CRC-16 필드는 통신하는 과정에서 데이터의 이상유무를 확인하는 용도로 사용한다.

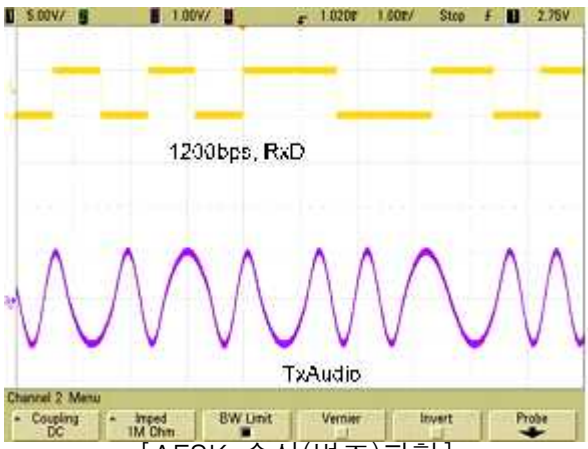
③ WAM-219XWS의 동작모드
WAM-219XWS의 동작은 2 개의 LED의 상태로 동작모드를 알 수 있으며, 모드에 따라 LED의 상태는 아래의 표와 같다.

	녹색 LED	적색 LED	설명
초기화 모드	Off	Off	1) POR(Power On Reset) 2) bRst 단자(Pin: 7)이용 Reset 시 모듈 Reset 후 모듈을 초기화 시키는 모드
설정모드	On	On	EEPROM 에 설정 값을 Read/Write 하는 모드
일반/수신모드	On	Off	데이터를 수신중인 상태
중계/수신모드	Off	On	
일반/송신모드	Off	On	데이터를 송신중인 상태
중계/송신모드	On	Off	
모듈의 중계기/송수신기 동작 확인 방법			
데이터 송수신이 없는 상태	적색 LED: On => 중계기로 동작 중(중계/수신 모드)		
	녹색 LED: On => 일반적인 송수신기로 동작 중(일반/수신 모드)		

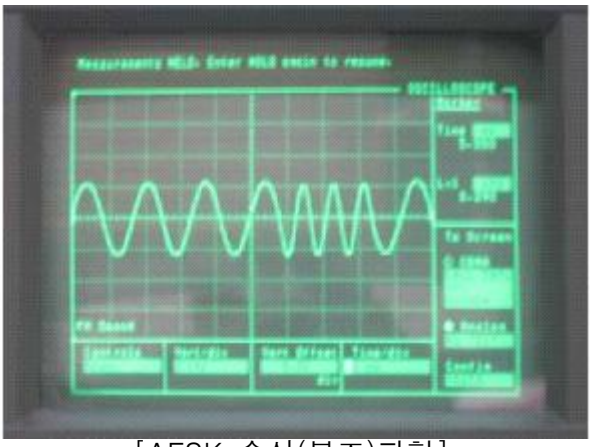
[WAM-219XWS 의 동작모드]

④ AFSK 변/복조파형 및 Transmit Spectrums

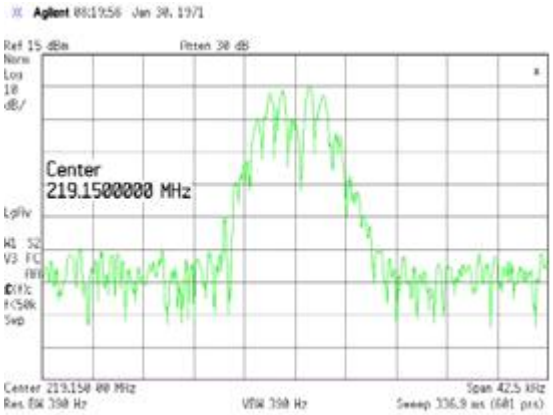
AFSK 변조는 송신 데이터 (RxD Pin) 를 송신기에서 FM 변조를 위한 Audio 신호를 생성시켜주는 부분으로 1200Hz(Data Low) 와 600Hz(Data High) 의 사인파형을 생성하여 FM 변조가되어 RF 로 출력된다 . FSK 모뎀 보다는 AFSK 모뎀이 점유 대역이 좁고 , 노이즈 특성이 우수하며 데이터의 전송률이 높다.



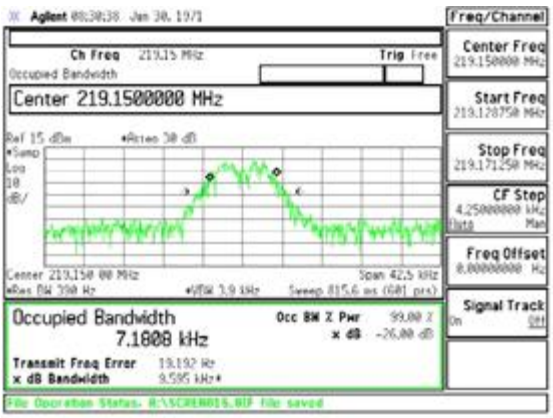
[AFSK 송신(변조)파형]



[AFSK 수신(복조)파형]



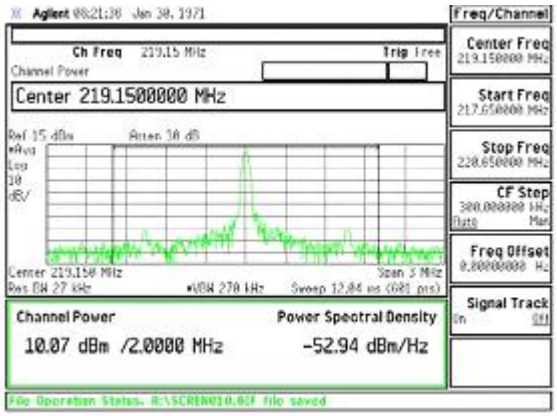
[AFSK 변조 스펙트럼]



[Occupied Bandwidth]



[ADJ Channel Power]



[Channel Power]

라. 통신프로토콜

1) 안테나

무선 통신에서 무엇보다 중요한 것이 안테나이다 . 성능과 크기 및 설치 용이성 등을 고려해서 제품을 사용하면 된다.

본 연구에서는 EWA-219H 로 1/2 파장의 휩 안테나를 사용하였다 .

2) N 키퍼용 통신 모듈 확장

그림 15 N 키퍼용 송신 모듈

N 키퍼용의 송신 모듈은 EL 패널부에 사용된 것과 동일한 RF 모듈을 USB 로 제어할 수 있도록 제작한 별도의 통신 모듈을 사용한다 . FT232RL 칩을 사용하여 PC 의 USB Serial Port로 사용한다.

3) 사용한 통신 프로토콜

과제 결과물의 테스트용으로 아래와 같은 조건으로 설정하여 통신 모듈에 적용하였다.

통신속도 : 9600bps, 8bit, N, 1

통신프로토콜

	STX	CMD	CNT	DATA	DATA	DATA	CR	LF	ETX
OFF	0x02	0x16	0x05	0x01	0x00	0x03	0x0D	0x0A	0x03
1ch ON	0x02	0x16	0x05	0x02	0x00	0x01	0x0D	0x0A	0x03
2ch ON	0x02	0x16	0x05	0x02	0x00	0x02	0x0D	0x0A	0x03
Flicker	0x02	0x16	0x05	0x02	0x00	0x03	0x0D	0x0A	0x03

STX : Start frame
CMD : RF Mod ule 의 HEX Data 전송 Command
CNT : Data 3byte + CR + LF 의 데이터 바이트 수
DATA : 3byte, 가변 가능
CR/LF : 데이터의 끝ETX : End Frame
현재는 테스트 프로토콜이며 , 향후 필요시 EX-OR Check Sum Byte 가 추가될 수 있음
Flicker 기능 : 1ch, 2ch 이 서로 번갈아가면서 ON/OFF 됨 . 시간은 ON 5 초 , OFF 5 초

4) 송수신-Slave 모드 실행
Master Check 버튼이 해제 상태가 되면 송수신-Slave 통신 상태로 전환이 되며 화면 상단이 아래와 같이 변한다. 테스트 프로그램에서는 송수신-Slave 통신 상태로 전환이 되면, 데이터 통신에서 많이 사용하고 있는 패킷의 시작은 STX(0x02), 끝은 ETX(0x03)을 사용하고, 패킷의 길이는 Info Length 에 따라서 변하는 프로토콜을 사용하였다.

STX(0x02)	Frame Type	Info Length	Info	ETX(0x03)
1 Byte	1 Byte	1 Byte	0 ~ n Byte	1 Byte

Slave-Frame 구조(STX-ETX Frame)

5) 송수신-Master 모드 실행
송수신-Master 상태가 되면, 설정모드 그룹박스 내의 사용자의 입력 값 RP 와 DA 값을 STX-ETX Frame 앞에 붙여 모듈과 SLIP(RFC-1055) 통신 프로토콜을 사용해서 통신을 한다.
※ SLIP 프로토콜은 RFC-1055 문서나 소스코드 참조

RP	DA	STX – ETX Frame(Table 3)
1 Byte	1 Byte	n Byte

Master-Frame 구조

6) N키퍼의 EL 디스플레이 모듈 제어

통신프로토콜을 통해 EL 디스플레이의 상태를 제어한다. N키퍼에서 재난모드가 발동이 되면 아래와 같은 코드로 EL 디스플레이 모듈로 상태를 송신하게 된다.

[EL display 제어 코드]

```
var SerialPort = require('serialport');
var http = require('http');
var port = new SerialPort('/dev/ttyUSB1', {
    baudRate: 9600
});
var elOff = new Buffer([0x02, 0x16, 0x05, 0x01, 0x00, 0x03, 0x0D, 0x0A, 0x03]);
var elOn1ch = new Buffer([0x02, 0x16, 0x05, 0x02, 0x00, 0x01, 0x0D, 0x0A, 0x03]);
var elOn2ch = new Buffer([0x02, 0x16, 0x05, 0x02, 0x00, 0x02, 0x0D, 0x0A, 0x03]);
var elFlicker = new Buffer([0x02, 0x16, 0x05, 0x02, 0x00, 0x03, 0x0D, 0x0A, 0x03]);

port.on('open', function() {
    setInterval(function() {
        http.get({
            hostname: 'localhost',
            port: 80,
            path: '/keys',
            agent: false
        }, (res) => {
            res.on('data', function(data) {
                var jsonData = JSON.parse(data.toString());
                console.log(jsonData[1].sensorNode)

                port.write(elOff, function(err) {
                    if (err) {
                        return console.log('Error on write: ', err.message);
                    }
                    console.log('message written');
                });
            });
        });
    }, 2000);

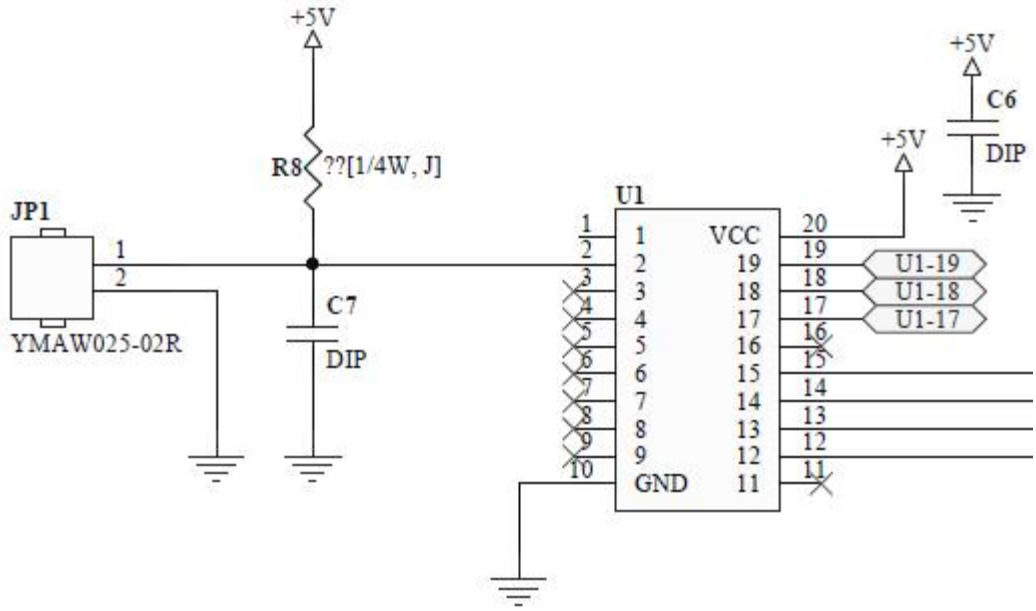
});

// open errors will be emitted as an error event
port.on('error', function(err) {
    console.log('Error: ', err.message);
})
```

마. 제어부

1) 채널 제어부

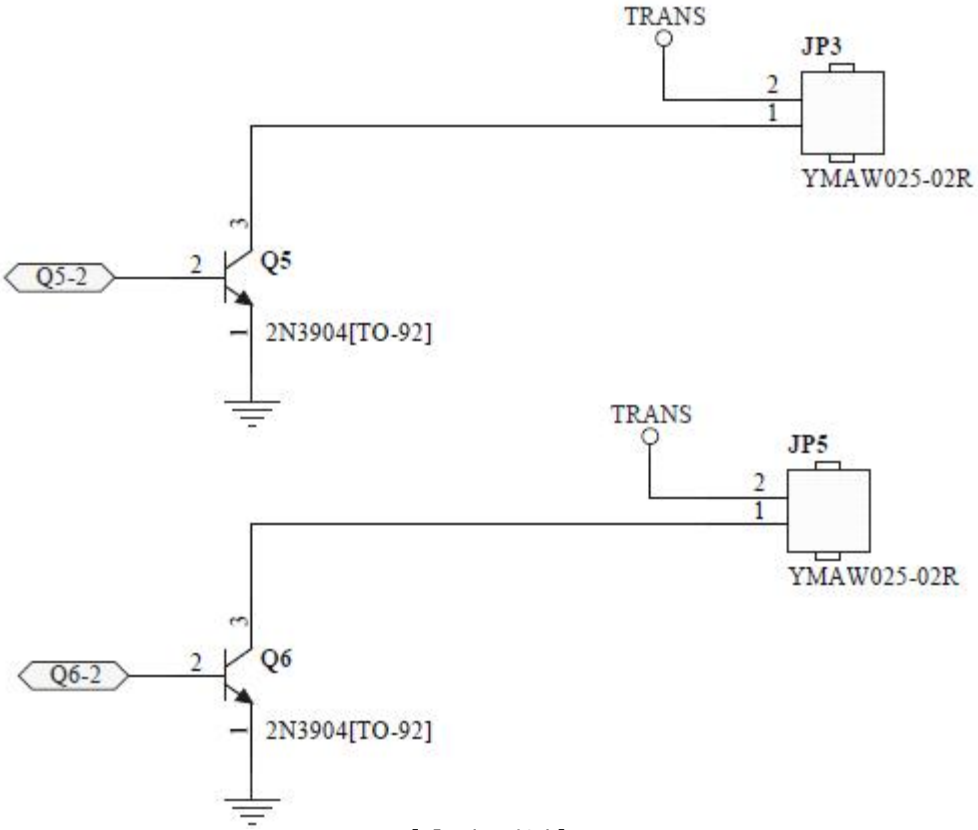
마이크로프로세서 (MG86FE508) 를 이용하여 각 채널의 ON/OFF 를 제어하도록 한다. 본 연구에서는 RF 수신모듈의 데이터를 프로세서가 분석하여 제어하도록 하였다.



[채널제어부]

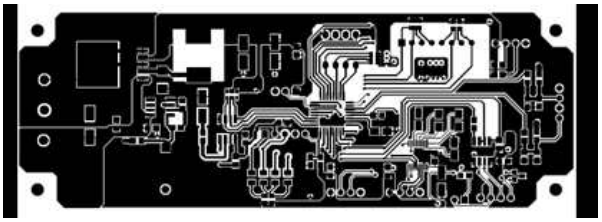
2) 출력 채널 구동

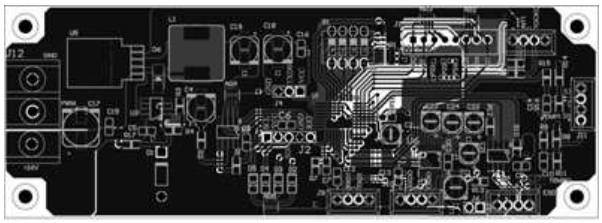
트랜스포머를 통해 발생된 파형을 각 EL 에 인가를 시키고 , 트랜지스터를 ON/OFF 시킴으로 서 발광이 될 수 있도록 한다.

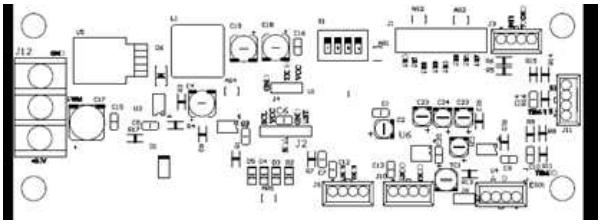


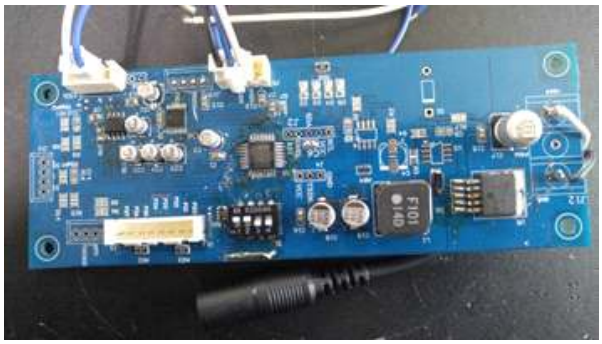
[출력 채널]

바. EL 디스플레이 패널 제어용 보드 가버데이터









사. 최종 프로토타입 설계

1) 시안 디자인

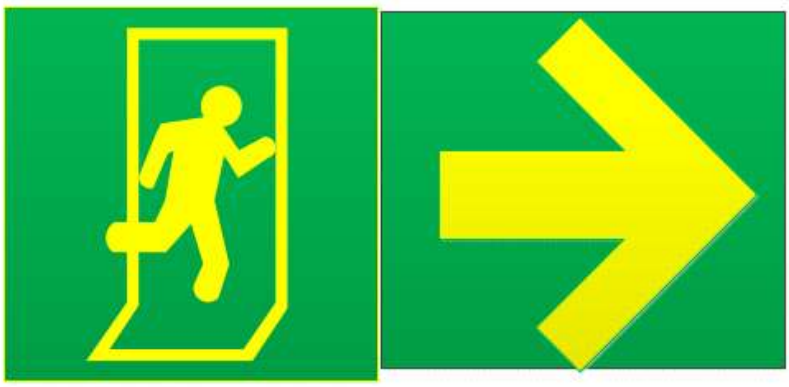
가) 라인 화살표



나) 벽면 대피로 표시



벽면용



[계단앞 벽면용]

· 시안


화살표 방향으로
이동하세요!

· 발광시


화살표 방향으로
이동하세요!

다) 원형 기둥


화재 발생
신속히 대피하세요
젖은 수건 등으로 코와 입을 막고
낮은 자세 로 대피하세요
반드시 계단을 이용하세요
화학 테러 발생
신속히 대피하세요
방독면 이나 젖은 수건
피부 노출
승화연구소
Modu engineering

13. 테스트베드 구축 및 현장시험

1) 1차년도 테스트베드 구축

가) 현장시험 필요성

당초 연구계획서상에는 테스트베드 현장설치 및 운용은 3차년도에 진행하기로 되어 있었다. 하지만, 충실한 연구성과 도출을 위해 철도공사와의 협의를 거쳐 1차년도 부터 현장시험을 진행하기로 계획을 변경하였다. 현장시험은 연구과제가 1차, 2차, 3차년도로 진행될수록, 보다 복잡하고 규모가 큰 지하철 역사들을 대상을 진행하여 많은 데이터베이스 축적과 실제 현장구축을 하면서 얻을 수 있는 보완점을 통해 더 좋은 연구결과가 도출될 것을 기대한다.

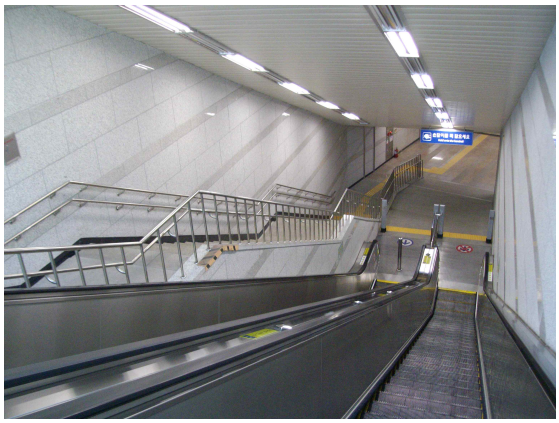
나) 1차년도 시험장소 선정

1차년도에 현장 시험을 실시하는 지하철역은 지하철 플랫폼에 스크린 도어가 설치되지 않은 곳을 선정하고자 하였다. 그 이유는 무선 센서링 네트워크에 대한 스크린 도어의 영향을 배제한 조건하에서 스마트N키퍼 시제품을 테스트하기 위한 것으로 2, 3차년도에는 보다 정교한 무선 센서링 네트워크의 구성을 통해 보다 복잡하고, 스크린 도어가 설치되어 있는 지하철역을 선정하기로 하였다.

또한, 현장 시험의 원활한 수행을 위해선 하루 평균 지하철 이용고객수가 적은 지하철역을 선정하는 것이 좋은데, 그 이유는 현장구축 및 운영적인 측면에서의 장점과 이용고객들의 불편을 최소화하기 위함이다. 이러한 점을 고려하였을 때 철도공사에서 직영으로 관리하는 과천선 대공원역을 현장시험장소로 선정하였으며 과천선 대공원역의 주요장소는 다음과 같다.



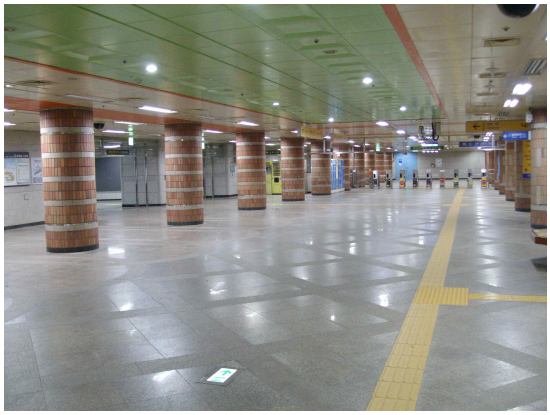
[대공원역 3번 출구]



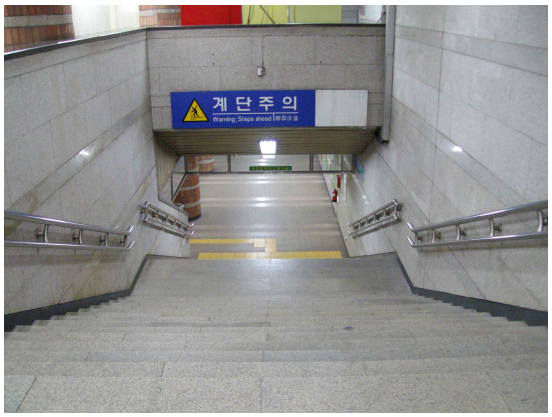
[3번 출구-지하1층 연결통로]



[지하1층 통로 전경 01]



[지하1층 통로 전경 02]



[사당방면 지하2층 연결계단]



[사당방면 승강장 플랫폼]



[안산방면 지하2층 연결계단]

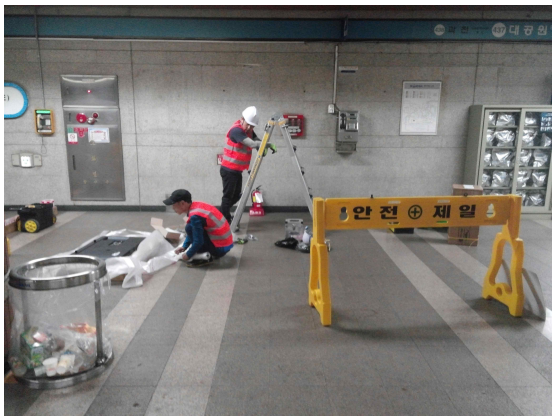


[안산방면 승강장 플랫폼]

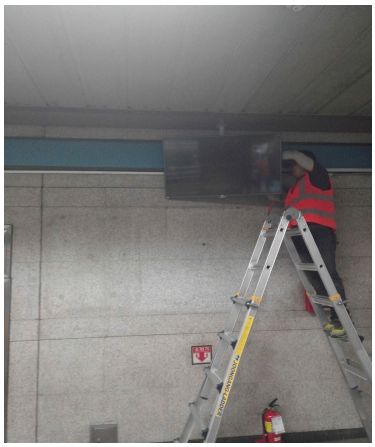
다) 작업기간 및 작업시간

작업기간의 경우 2015. 5. 18. ~ 5. 19. 2일간 Node와 스마트N키퍼용 디스플레이를 설치하였으며, 중·고등학교의 소풍시기인 점을 감안하여 안전사고 예방 및 이용객의 불편을 최소화하고자 이용객이 적은 시간인 목요일과 금요일 오후 6시 이후에 작업을 실시하였다. (철도공사 감독관 입회하에 시험장비 설치 및 테스트)

현장 작업사진은 다음과 같다.



[스마트N키퍼용 디스플레이어 설치 전경 01]



[스마트N키퍼용 디스플레이어 설치 전경 02]



[스마트N키퍼용 디스플레이어 설치 전경 03]



[스마트N키퍼용 디스플레이어 설치 전경 04]



[USB Gateway 설치 전경]

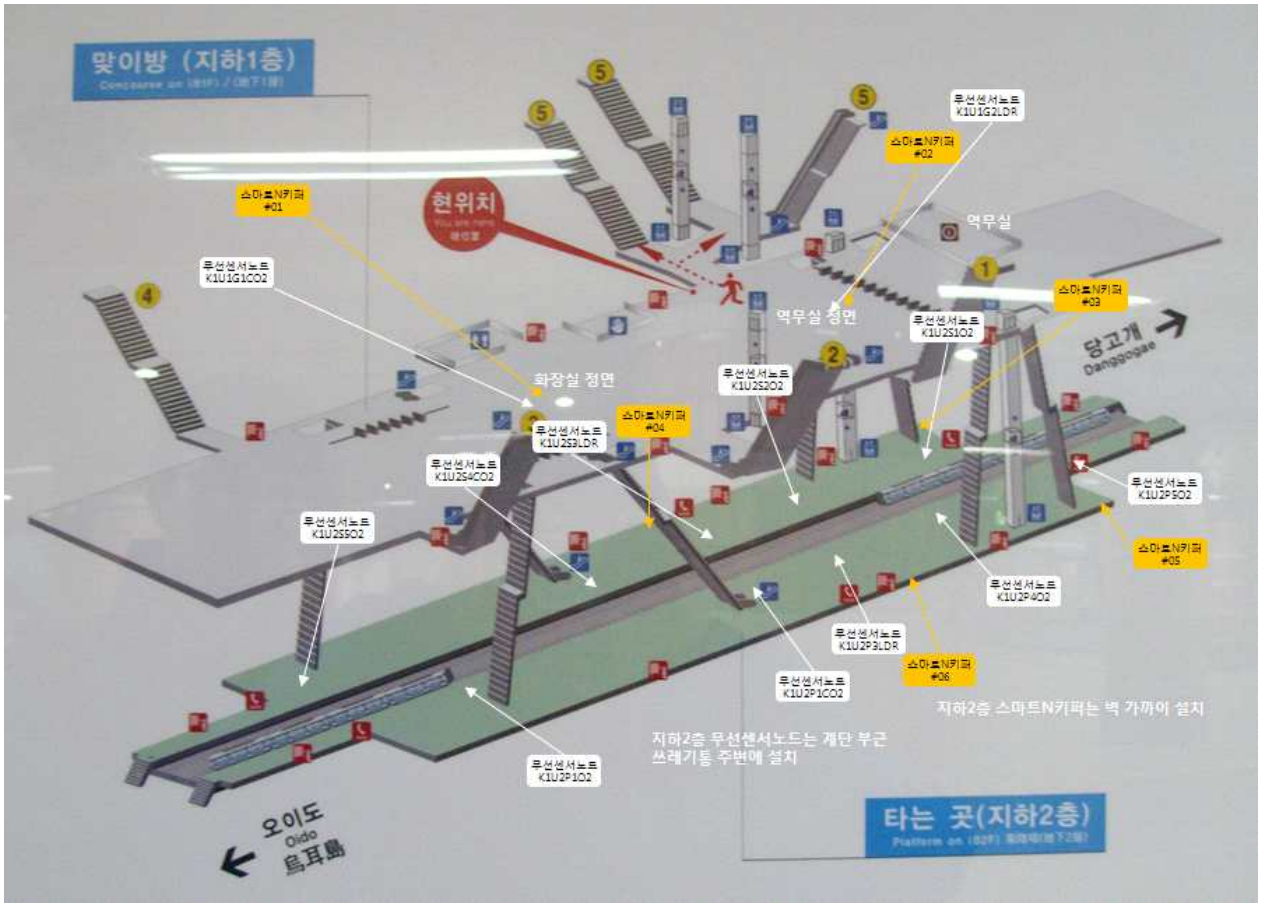


[Node 설치 전경]

라) 시험장비 설치위치

가스탐지용 Node의 경우 역과 역 사이에서 화재 또는 재난 발생 시 유해가스 유입을 조기에 감지하고 대응 할 수 있도록 양방향 플랫폼의 양쪽 끝부분에 설치하고 지하1층엔 이용객이 가장 많이 이용하는 2번 출구에 설치하였다.

또한, 조도/소음탐지용 Node의 경우 소음이 가장 많이 발생하는 지하1, 2층 의 중앙위치에 설치하였다. 이번 현장실험장소인 과선천 대공원역의 M2M 장비 배치도는 다음 그림과 같다.



[과천선 대공원역 M2M 장비 배치도]

마) 스마트N키퍼(α Version) 설치 및 구축

일반적인 방재대응 시스템은 운영자 혹은 초기 설정에 의해서 이후 프로세스가 진행되는데, 최근의 일련의 사고에서 알 수 있듯이, 운영자 대응 미숙, 정보의 비대칭성에 의해서 피해규모가 커지는 사례가 많이 발생하고 있다.

이에 본 연구에서는 스마트N키퍼라고 하는 분산자율형 재난대응 디바이스(α Version)를 재난대응시스템의 로컬(현장)에 설치하였다.

스마트N키퍼의 목표 및 기대효과로는 디지털 사이니지의 역할을 하면서 재난감지 센서들 간의 Mesh네트워크를 구성하여 Node간의 정보를 공유함으로써 초기대응 준비를 할 수 있도록 하며, 정보의 비대칭성에 의한 피해를 최소화하는 것이다.

스마트N키퍼에 사용되는 구성장비 중 디스플레이 패널은 스피커내장의 광시야각 기능을 가진 제품으로 선정하였으며, 패널 프레임에 부착하여 스마트N키퍼의 중추 역할을 수행하는 Mini PC의 자세한 Spec은 다음과 같다.



제조회사		LG(42LX530H)
화면규격	화면크기	106cm / 42인치
	해상도	1920 x 1080, 풀HD
	응답속도	1 ㎳
	시야각	178° 광시야각
화질	패널	IPS 패널
	화면스캔	60 Hz
	선명도	CPI 120 Hz
음향	스피커출력	10W + 10W 버츄얼 서라운드
단자	입출력	HDMI x 2, USB x 1, PC입력(D-Sub) x 1, RS-232C x 1, RF안테나 x 1, 컴포넌트&컴포지트 x 1

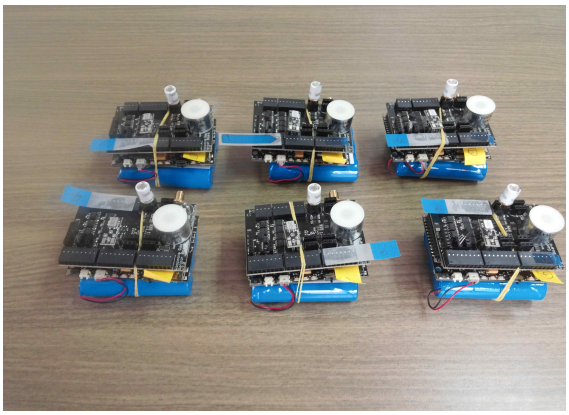
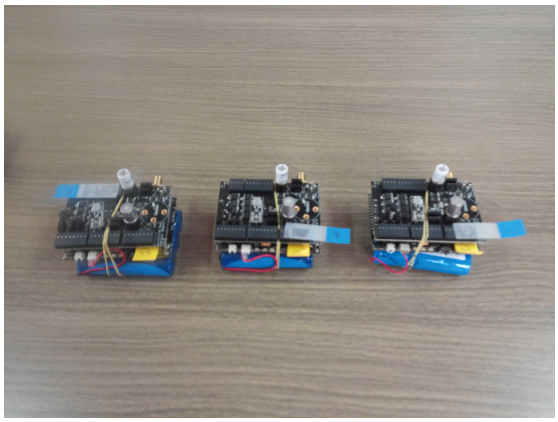
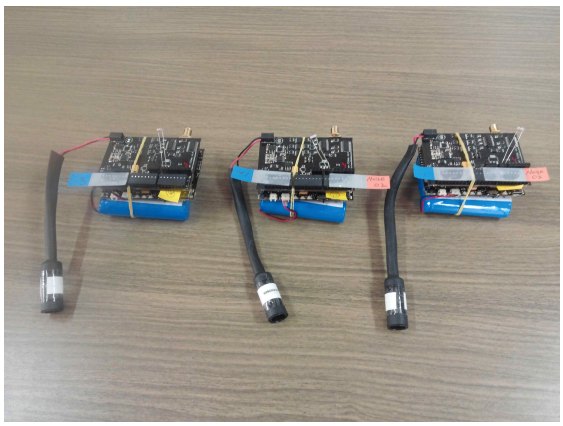
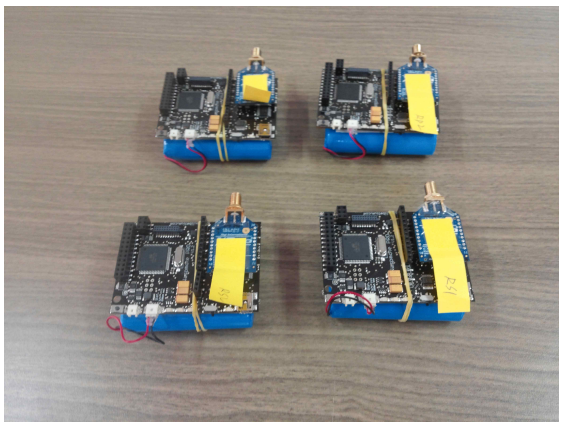


제조회사	Intel(NUC5i3RYH)
크기	115mm x 111mm x 48.7mm
프로세서	5세대 인텔 Core i3-5010U 프로세서
메모리	듀얼 채널 DDR3L SODIMM
시야각	178° 광시야각
네트워크	인텔 Wireless-AC7265 기본 내장 (IEEE 802.11ac, 블루투스 4.0, 인텔무선 WIDI)
입출력	Mini HDMI x 1, USB 3.0포트 x 4 (전면, 후면 각가 2개) IR 적외선 센서(전면 패널)

바) 가스/조도/소음/온도/가속도센서 설치

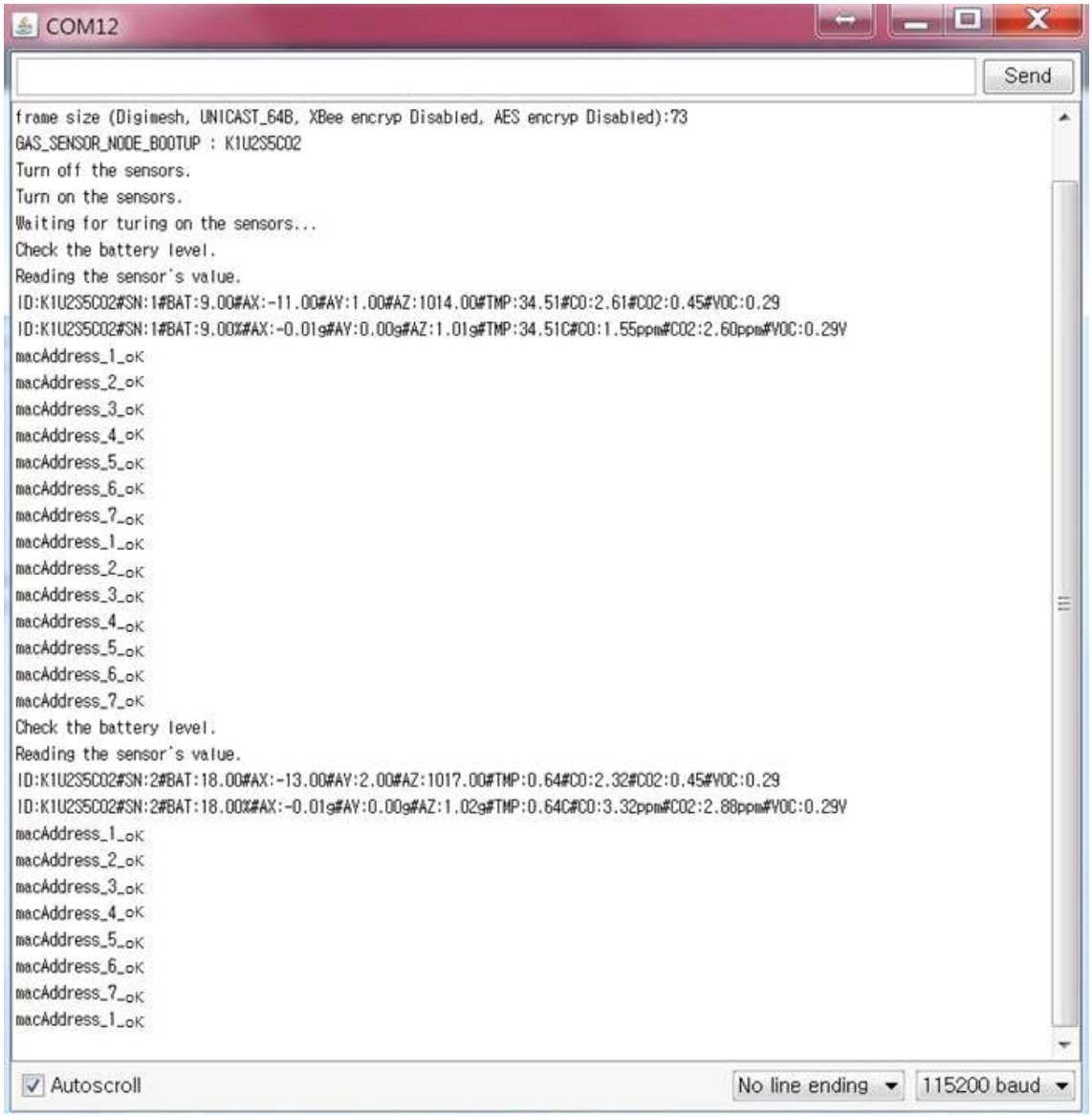
□ Step 01 : WSN 메인보드와 가스센서보드 및 조도감지보드 결합

CO2감지 Node 3ea, O2감지 Node 6ea, 소음/조도감지 Node 3ea를 제작하여 미리 선정한 위치에 각각 설치하여 스마트N키퍼와 Mesh 네트워크를 구성하고 센서의 정상작동을 확인하고자 한다.

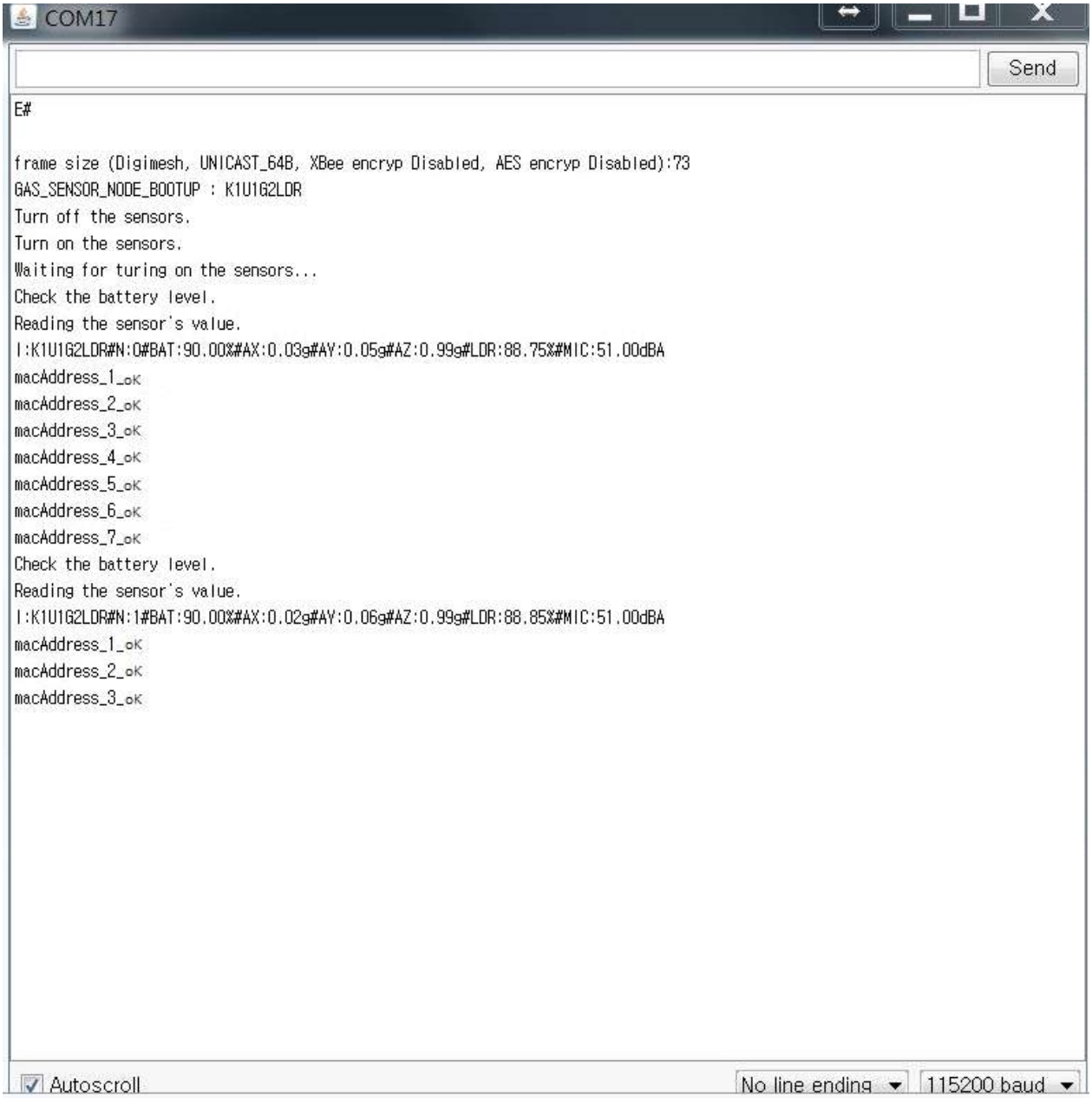
	
O_2감지 센서보드 6EA	CO_2감지 센서보드 3EA
	
조도/소음 센서보드 3EA	Router 4EA

□ Step 02 : 센서 Node용 메인보드 프로그래밍

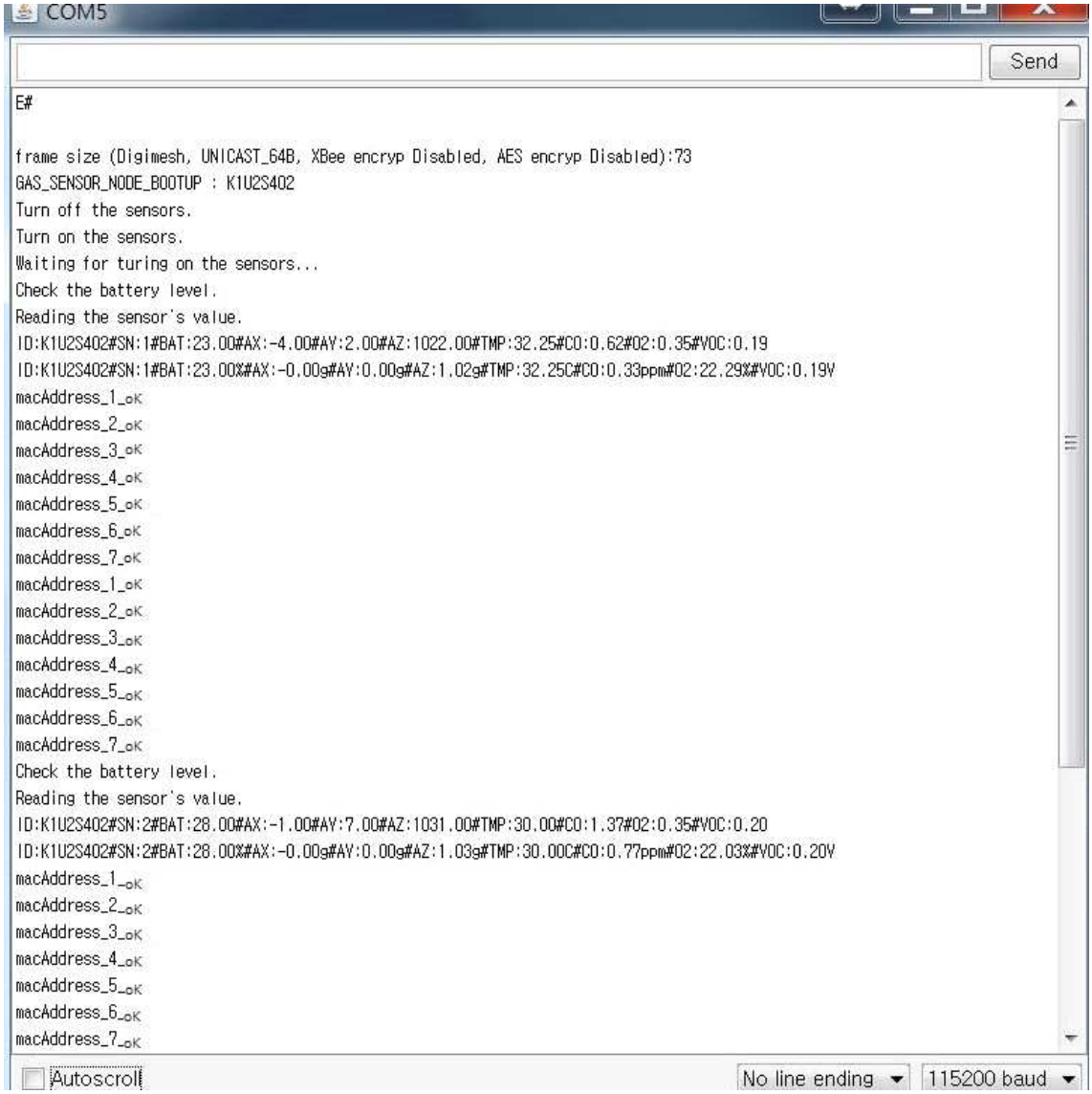
각각의 센서 Node의 용도에 맞도록 프로그래밍을 하여 임베디드 하였으며 시리얼 모니터를 통해 나오는 Output은 다음과 같다.



[CO_2 센서보드 시리얼모니터 Output]



[조도/소음 센서보드 시리얼모니터 Output]



[O_2 센서보드 시리얼모니터 Output]

사) 무선 센서링 메쉬 네트워크 테스트

□ Step 01 : WSN 노드용 메인보드 프로그래밍

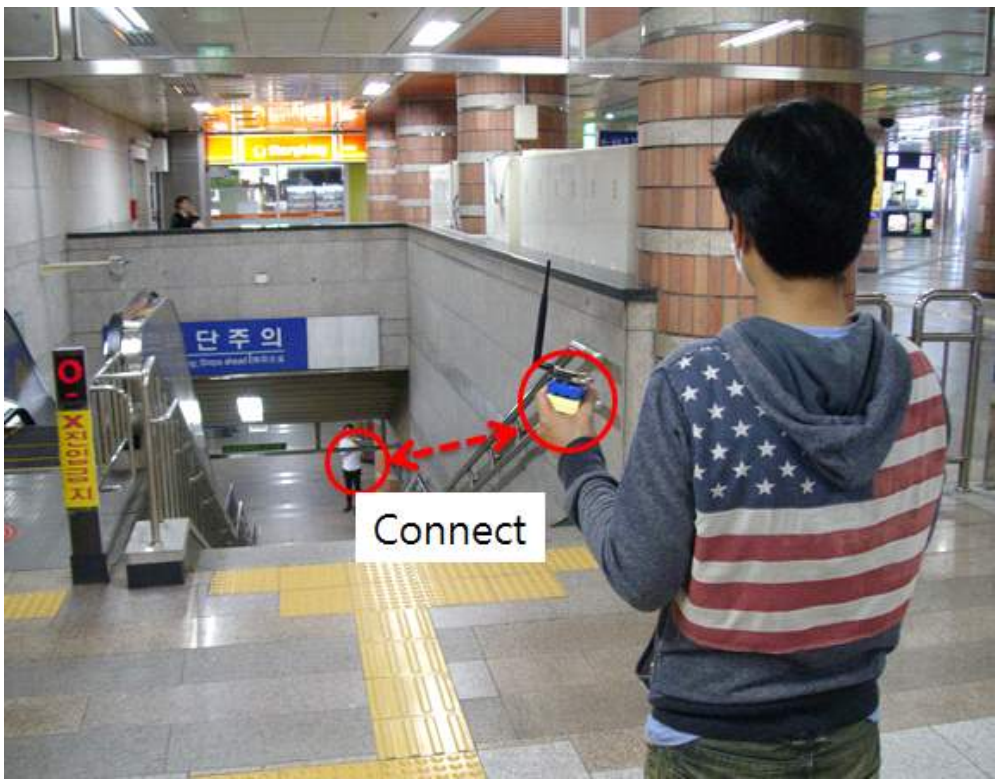
XCTU를 이용하여 Node 및 USB Gateway의 무선통신 모듈을 설정해야 하며 자세한 내용은 XCTU설정 장에서 기술하였다.

□ Step 02 : 대공원역 내부의 공간적 특성 분석

지하2층의 10개의 Node와 지하1층의 2개의 Node의 Mesh 네트워크망 구축을 위해 Router를 연결계단에 2개씩 설치하였다.



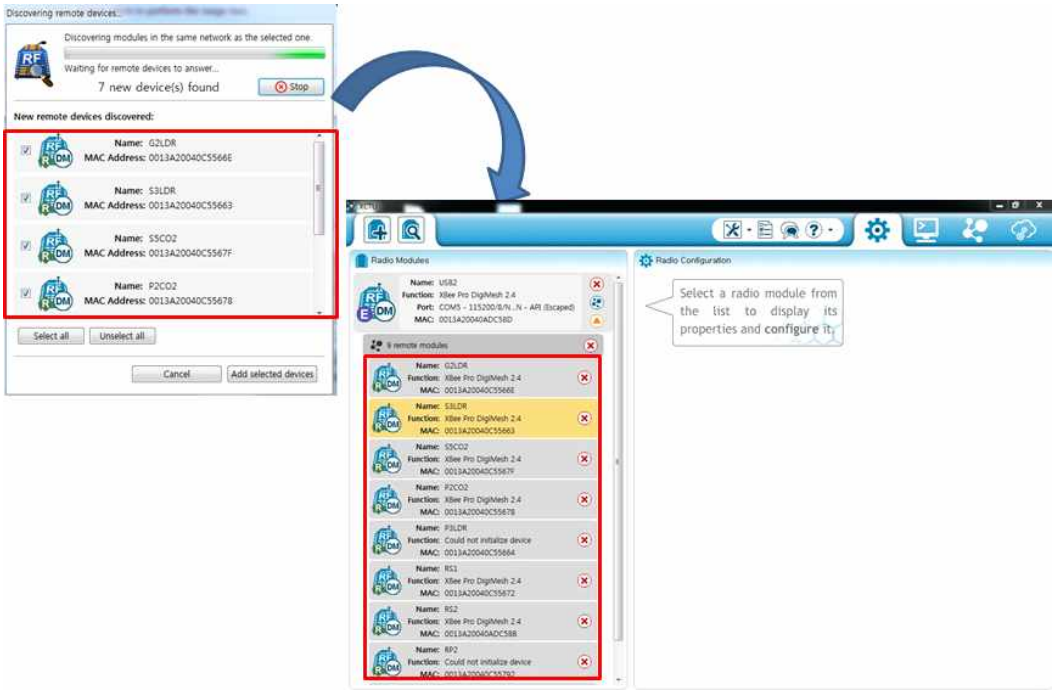
[지하1층과 지하2층 연결 Router 위치 확인 01]



[지하1층과 지하2층 연결 Router 위치 확인 02]

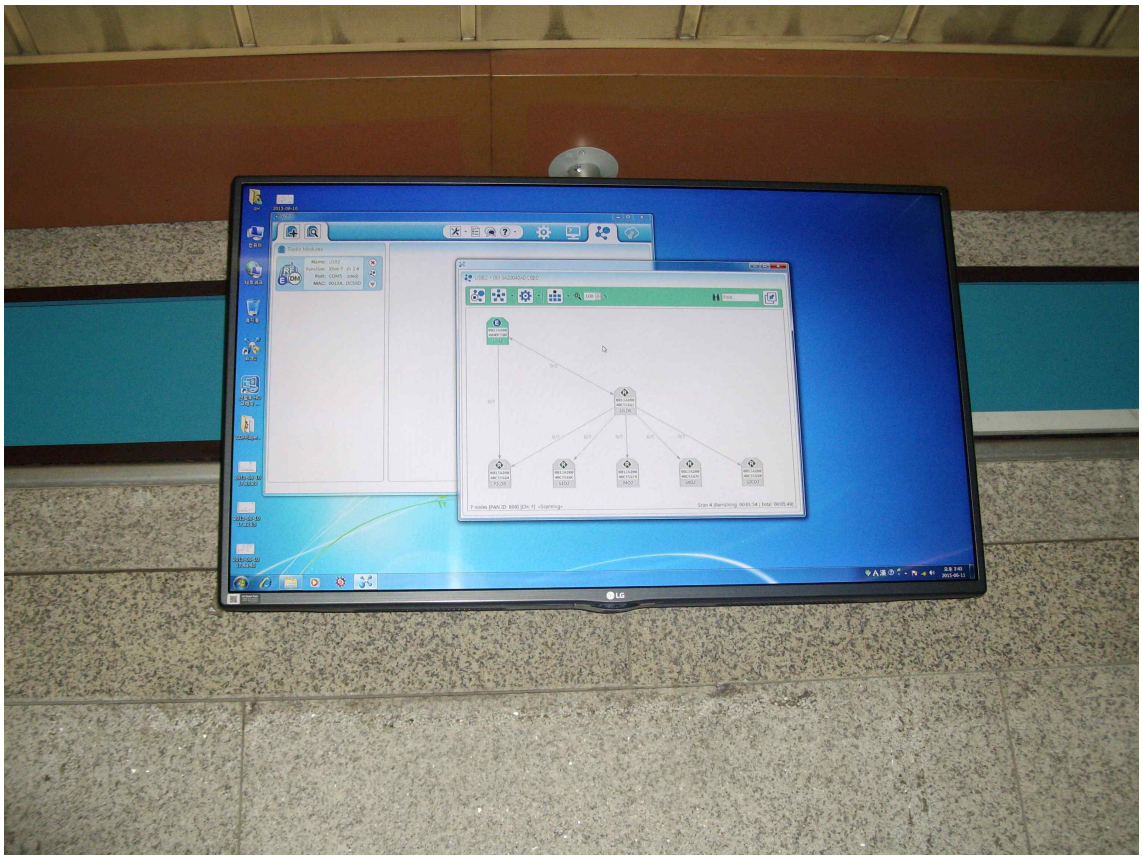
□ Step 03 : XCTU를 이용한 Mesh 네트워크망 확인

XCTU를 이용하여 USB Gateway와 Mesh 네트워크를 이루는 Node의 무선통신 모듈을 확인할 수 있으며 실제 프로그램의 구동모습은 아래의 그림과 같다.

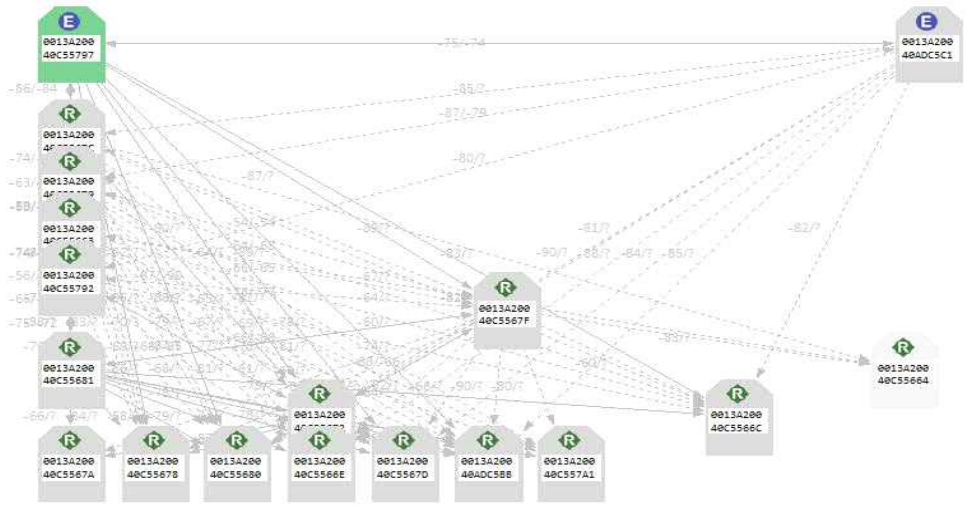


□ Step 04 : Mesh 네트워크 정상작동 테스트

각각의 센서 Node들과 End Device인 USB Gateway가 Mesh 네트워크망을 정상적으로 구축하고 연결이 되는지 확인한다. XCTU를 이용한 Mesh네트워크 내의 데이터 송수신 그래프는 다음 그림과 같다.



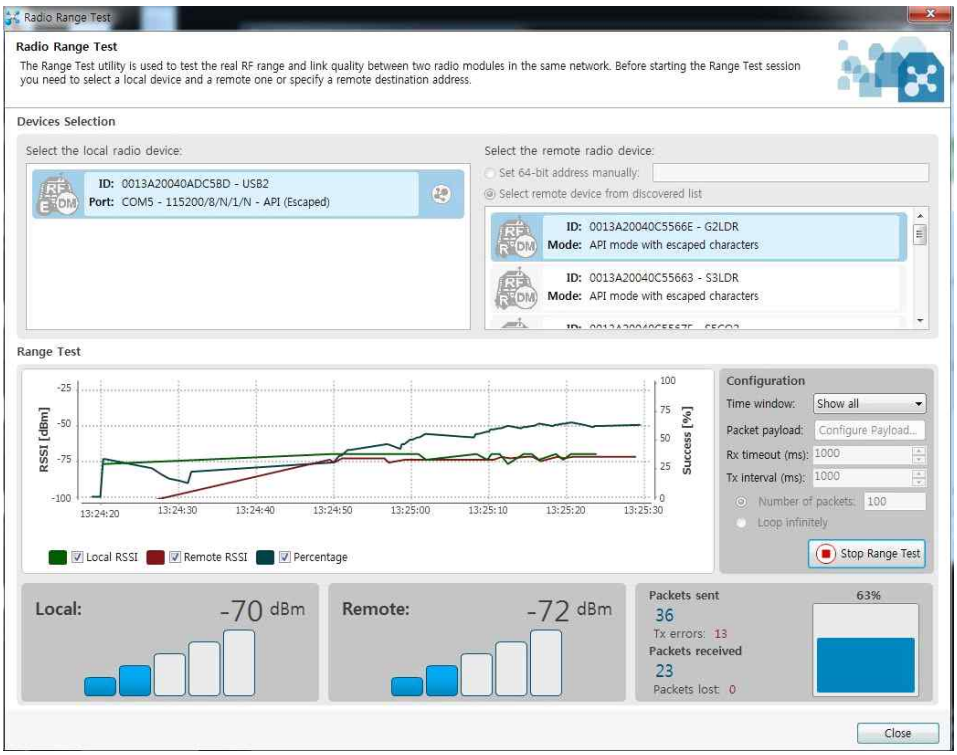
[M2M장비간 Mesh 네트워크망 Date 송·수신 그래프 01]



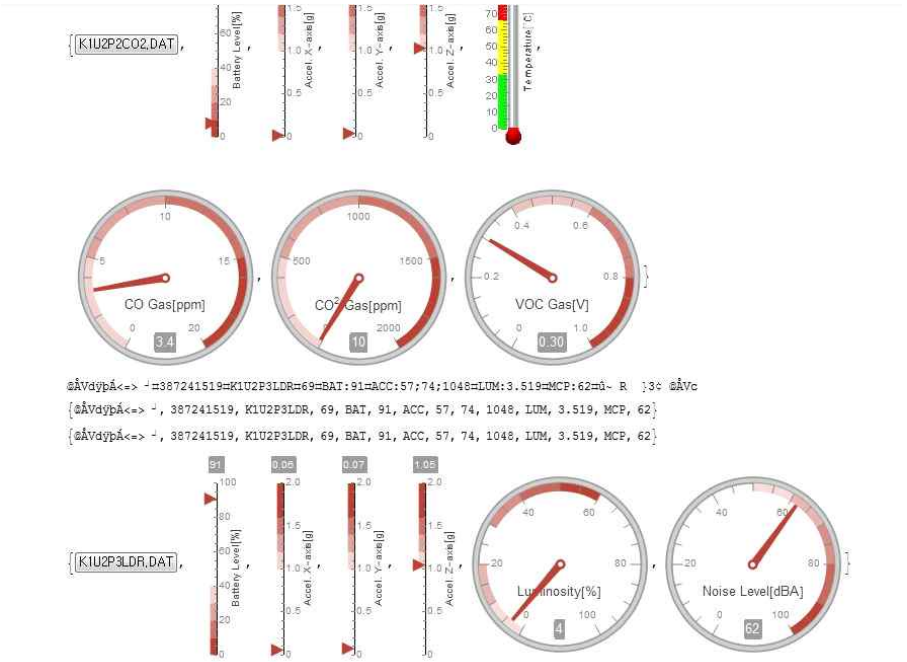
[M2M장비간 Mesh 네트워크망 Date 송·수신 그래프 02]

□ Step 05 : 통신감도 및 시리얼 터미널 수신 Date 확인

Mesh 네트워크 내의 Node의 통신감도를 확인하고 최종적으로 시리얼 터미널인 CDF Player PRO를 이용하여 수신된 Date를 확인하였다.



[USB Gateway 무선통신 감도 확인]



[CDF Player PRO 출력 값 01]



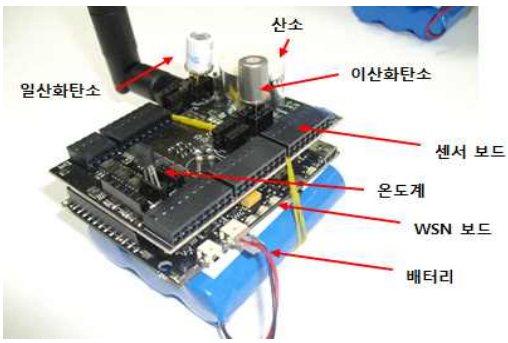
[CDF Player PRO 출력 값 02]



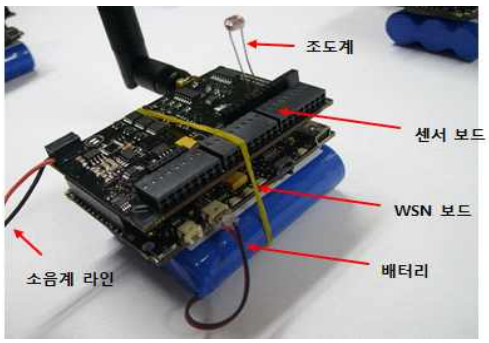
[CDF Player PRO 출력 값 03]

2) 1차 테스트베드 추진경과

- 1차 테스트베드인 대공원역의 경우 지하 2층 구조의 비교적 간단한 구조이며, 가스탐지 센서와 소음(조도)센서의 두종류의 센서보드에 대한 무선통신 및 현장 테스트를 실시하였으며 주요 내용은 다음과 같음.



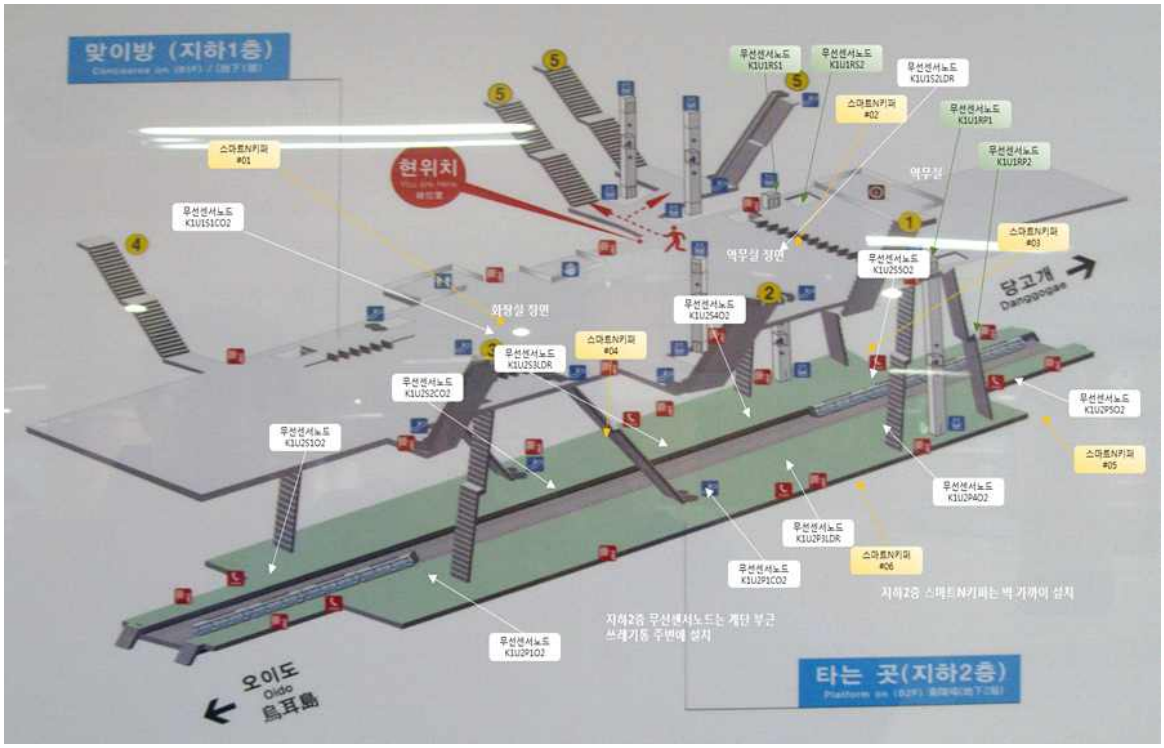
[가스탐지 센서]



[소음탐지 센서]

[센서보드 종류별 사용 센서]

NO	조도	소음	CO	CO2	O2	Temp	VOC	비고
1	-	-	√	-	√	√	√	O2 센서보드
2	-	-	√	√		√	-	CO2 센서보드
3	√	√	-	-	-	-	-	조도/소음 센서보드
4	-	-	√	√	-	√	-	CO2 센서보드
5	-	-	√	-	√	√	√	O2 센서보드
6	-	-	√	-	√	√	√	O2 센서보드
7	-	-	√	√	-	√	-	CO2 센서보드
8	√	√	-	-	-	-	-	조도/소음센서보드
9	-	-	√	-	√	√	√	O2 센서보드
10	-	-	√	-	√	√	√	O2 센서보드
11	-	-	√	-	√	√	√	O2 센서보드
12	√	√	-	-	-	-	-	조도/소음센서보드
13	-	-	-	-	-	-	-	Router(사당방향)
14	-	-	-	-	-	-	-	Router(사당방향)
15	-	-	-	-	-	-	-	Router(평촌방향)
16	-	-	-	-	-	-	-	Router(평촌방향)



[대공원역 센서노드 및 스마트N키퍼 설치현황]

- 16개의 센서보드와 6개의 스마트N키퍼 간의 Digimesh를 통한 무선네트워크 통신을 구축하여 스마트N키퍼 베타버전 구동 검증.

[스마트N키퍼 재난전도 예측 프로그램 구동모습]

		
평시	재난감지	재난확대

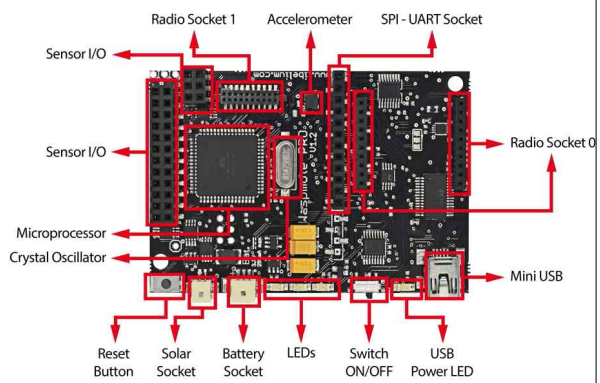
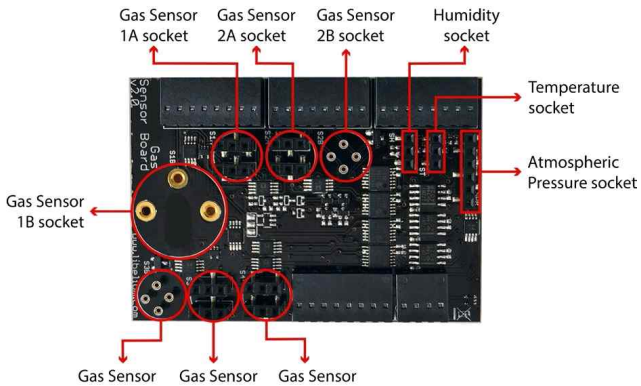
가) 실험목표

- 1차 테스트베드에서 어려움을 겪었던 ①무선통신네트워크의 개선과 ②배터리소모량을 감소하기 위한 코드수정 및 ③가스센서노드의 방수 케이스 및 센서의 탈부착이 용이한 센서프로브 개발 등과 같은 전반적인 소프트웨어 및 하드웨어의 고도화 테스트와 현장 데이터 수집을 통한 대피경로 분석 알고리즘, 사용자(관리자) UI 등에 대한 현장 적용 테스트

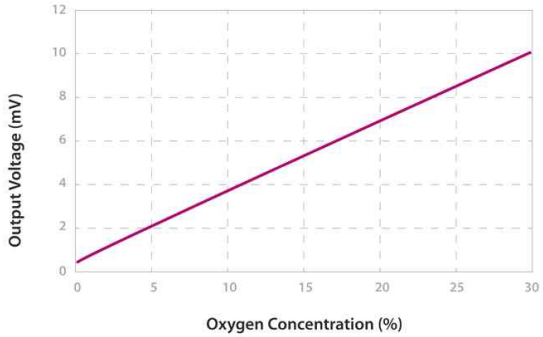
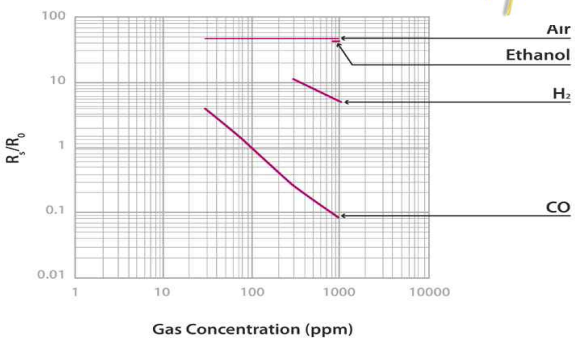
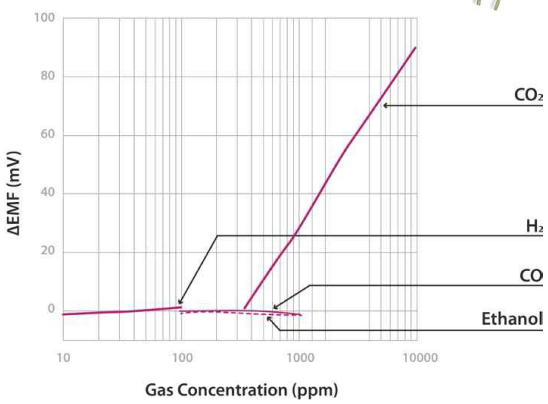
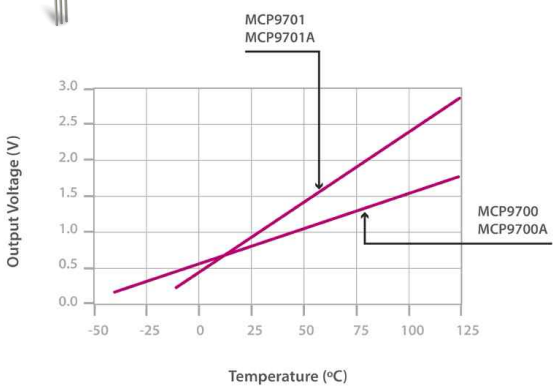
나) 설치장비

- 기존 사용하는 가스센서의 정확성과 신뢰도를 높이며, 추후 사업화 과정에서 유지관리비용을 저감하기 위하여 Calibrated가 된 가스센서를 기준으로 기존의 가스센서(비용절감)를 보정하여 테스트를 수행

- 무선통신모듈의 경우, 기존 1차년도 테스트베드 구축 시 센서노드들과 스마트N키퍼 간 양방향 통신을 하는 Digimesh로 네트워크를 구성하였으나 철근콘크리트 구조의 지하철역은 무선 통신의 간섭 및 방해가 많아 서브 1GHz의 저주파 장거리 통신을 사용하기로 함

가스 센서 메인보드	가스센서 보드
WSN Node Mainboard - Microcontroller : ATmega1281 14.7456MHz - Weight : 20g - Dimensions : 73.5 x 51.0 x 13.0mm - Consumption : 15mA(On), 55μA(Sleep), 0.07μA(Hibernate) - Battery : 3.3V ~ 4.2V - Storage : 128KB(Flash), 2GB(SD Card) - Certification : CE(Europe), FCC(USA), IC(Canada)	Gases Sensor Board - Weight 20g - Dimensions : 73.5 x 51.0 x 13.0mm - Board power : 3.3V, 5.0V (cont. 200mA, peak 400mA) - Sensor power : 5.0V - Gas sensors : O_2, CO, CO_2, VOC, CH_4, H_2, O_3 etc.
 [Mainboard 상부 이미지]	 [가스 탐지용 센서 보드]

무선통신모듈(LoRa)	게이트웨이
LoRa 5 dBi - Protocol : LoRa - Frequency : 860~1050 MHz - 158 dB maximum link budget - +20 dBm - 100 mW constant RF output vs. V supply - +14 dBm high efficiency PA - Programmable bit rate up to 300 kbps - High sensitivity: down to -138.5 dBm - Bullet-proof front end: IIP3 = -12 dBm with FSK - 100 dB blocking immunity - Low RX current of 10 mA, 200 nA register retention - Fully integrated synthesizer with a resolution of 61 Hz - Built-in bit synchronizer for clock recovery - Sync word recognition - Preamble detection - 115 dB+ Dynamic Range RSSI - Automatic RF Sense with ultra-fast AFC - Packet engine up to 64 bytes with CRC - Built-in temperature sensor and low battery indicator	Digimesh PRO AP - Processor : 500MHz(x86) - RAM : 256MB(DDR) - HDD : 8GB - Consumption : Normal 270mA, High 450mA - Weight : 1.2Kg - Dimensions : 210 x 175 x 50mm - Response : 60s - Power : 5W(18V) - Power source : POE(Power Over Ethernet) - Power supply : AC 220V - System : Debian GNU/Linux - Protocol : OLSR Mesh Communication/MadWiFi - Security : Authentication WEP, WPA-PSK, HTTPS, SSH - Certification : CE(Europe), FCC(USA) 

O_2 Sensor(산소측정)-Not calibrated	CO Sensor(일산화탄소측정)-Not calibrated
O_2 Sensor(산소측정)-Not calibrated - Measurement range : 0~30% - Output range : 0~10mV - Initial voltage : 5.5~8.8mV - Response time : 15s - Consumption : 0μA  	CO Sensor(일산화탄소측정)-Not calibrated - Measurement range : 30~1000ppm - Resistance at 100ppm : 13.3~133kΩ - Sensitivity : 0.13 ~ 0.31 - Supply voltage : 5V±0.2V DC - Response time : 1s - Min. Load resistance : 10kΩ - Average consumption : 3mA  
CO_2 Sensor(이산화탄소 측정)-Not calibrated	Temperature(온도측정)-Not calibrated
CO_2 Sensor(이산화탄소 측정)-Not calibrated - Measurement range : 350~10000ppm - Voltage at 350ppm : 220~490mV - Sensitivity : 44 ~ 72mV - Supply volatage : 5V±0.2V DC - Response time : 1.5m - Average consumption : 50mA  	Temperature(온도측정)-Not calibrated  

O_2 Sensor(산소측정)-Calibrated	NO_2 Sensor(이산화질소측정)-Calibrated
Performance Characteristics - Nominal Range: 0 to 30 Vol.% - Maximum Overload: 90 Vol.% - Long Term Output Drift: < 2% signal/3 months - Response Time (T90): ≤ 30 seconds - Sensitivity: 1.66 ± 0.238 nA/ppm - Accuracy: as good as ± 0.1 % (ideal conditions) 	Performance Characteristics - Nominal Range: 0 to 20 ppm - Maximum Overload: 250 ppm - Long Term Output Drift: < 2% signal/month - Response Time (T90): ≤ 30 seconds - Sensitivity: 600 ± 150 nA/ppm - Accuracy: as good as ±0.1 ppm* (ideal conditions) 
HCN Sensor(이산화탄소 측정)-Calibrated	Temperature, Humidity(온도,습도 측정)-Calibrated
Performance Characteristics - Nominal Range: 0 to 50 ppm - Maximum Overload: 100 ppm - Long Term Output Drift: < 2% signal/month - Response Time (T90): ≤ 120 seconds - Sensitivity: 100 ± 20 nA/ppm - Accuracy: as good as ±0.2 ppm* (ideal conditions) 	Temperature sensor: - Operational range: -40 ~ +85 ºC - Full accuracy range: 0 ~ +65 ºC - Accuracy: ±1 ºC (range 0 ºC ~ +65 ºC) - Response time: 1.65 seconds (63% response from +30 to +125 °C). - Typical consumption: 1 µA measuring Humidity sensor: - Measurement range: 0 ~ 100% of Relative - Humidity (for temperatures < 0 °C and > 60 °C see figure below) - Accuracy: < ±3% RH (at 25 ºC, range 20 ~ 80%) - Hysteresis: ±1% RH - Operating temperature: -40 ~ +85 ºC - Response time (63% of step 90% to 0% or 0% to 90%): 1 second - Typical consumption: 1.8 µA measuring - Maximum consumption: 2.8 µA measuring

3) 2차 테스트베드 실험장소

가) 장　　소 : 철도공사 직영 분당선 개포동역
　테스트베드시험운영기간 : 설치이후 약 4개월간

나) 선정사유 :

- 1차년도 테스트베드 결과를 바탕으로 대심도 지하공간(지하 1~4층)에서 발생하는 다양한 유형의 재난을 첨단센서를 통해 감지하고 신속한 고객 대피 의사결정 지원 기술개발에 필요한 재난감지 장비 시제품의 네트워크 통신 및 작동상태와 센서의 성능시험을 시행하고자 함. → 다양한 지하 공간 특성을 고려한 통합관리지원체계 개발을 목적으로 대심도 추가 공간을 확보 및 실험

다) 설치 작업
- 전기인입/배선/브라켓 설치 : 2016년 03월 28일 ~ 2016년 03월 29일
- 스마트N키퍼/스마트N키퍼용 모니터/WSN센서노드 설치(네트워크 연결 테스트용) : 2016년 03월 30일

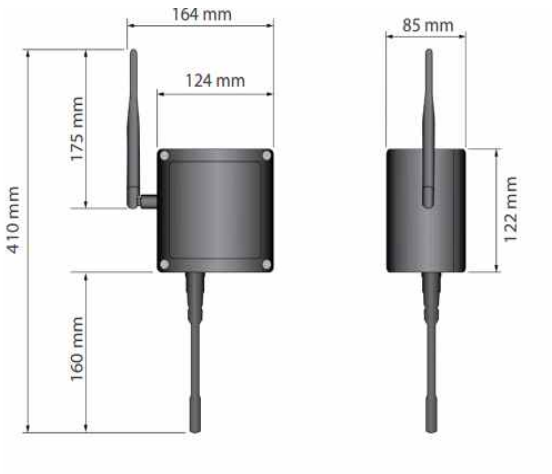
라) M2M 장비 설치방안

가. WSN 노드 설치방안
- WSN 노드 전용 방수 및 화재에 강한 케이스 구매 및 센서의 탈부착이 가능하도록 센서프로브 장착

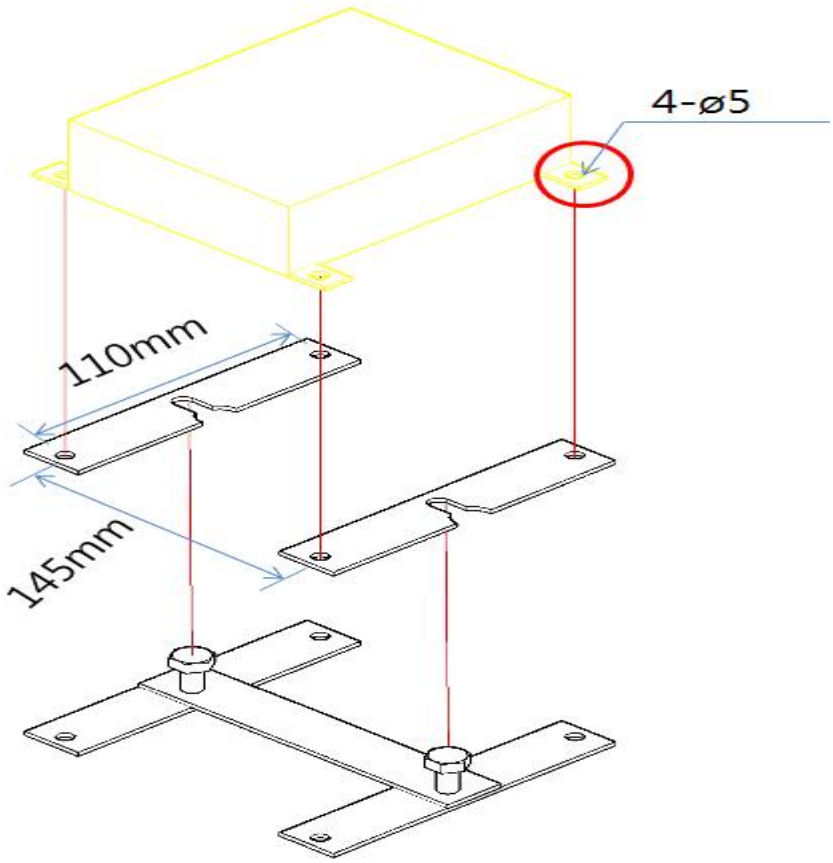


[센서프로브 탈부착 모습]

- 가스가 감지되는 가스프로브의 높이 위치를 사람의 키높이 위치에 설치하여 선지적 정보제공을 위한 빠른 가스감지를 하고자 함

Material: polycarbonate • Sealing: polyurethane • Cover screws: stainless steel • Ingress protection: IP65 • Impact resistance: IK08 • Rated insulation voltage AC: 690 V • Rated insulation voltage DC: 1000 V • Heavy metals-free: Yes • Weatherproof: true - nach UL 746 C • Ambient temperature (min.): -10 °C • Ambient temperature (max.): 50 °C • Approximated weight: 800 g	

- 추후 테스트베드 기간 이후 센서노드의 철거가 용이함과 테스트베드의 벽면의 피해를 최소화하기 위하여 브라켓 제작



[센서노드 설치용 브라켓 도면]

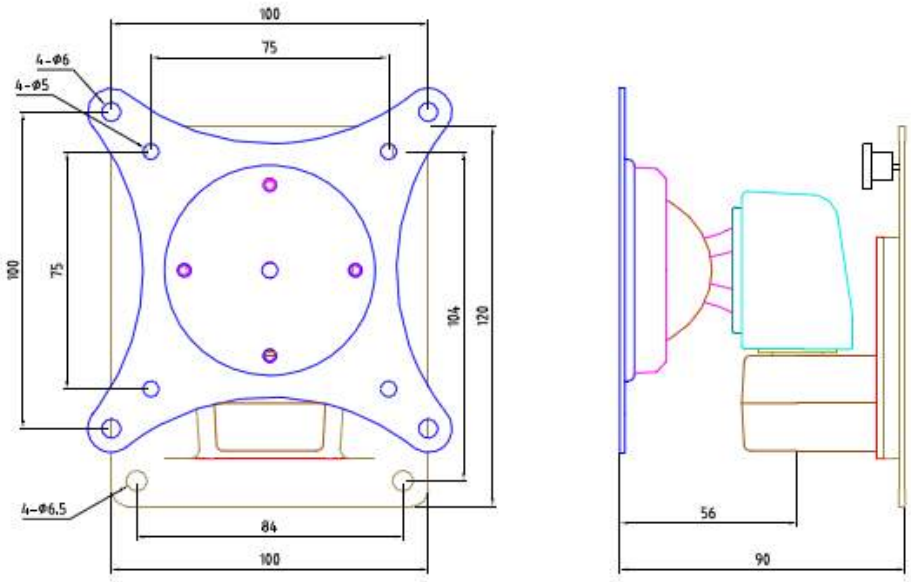
마) 스마트N키퍼 설치방안

- 32inch 스피커 내장형 모니터와 LoRa 통신 모듈을 장착한 미니PC를 설치하여 스마트N키퍼 구성



[스마트 N키퍼 설치 예(모니터+미니PC 구성)]

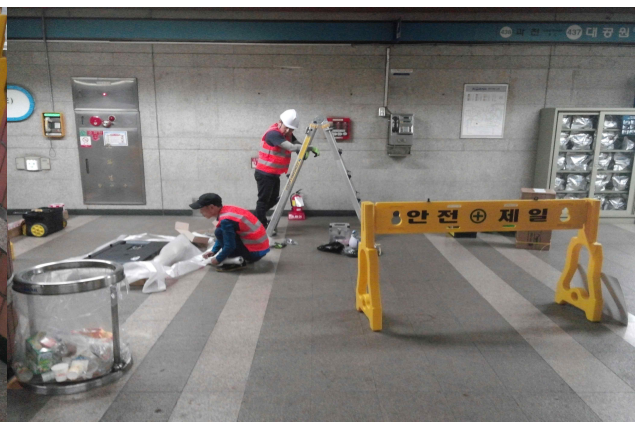
- 스마트N키퍼용 모니터는 벽면에에 월브라켓을 설치하고 모니터 베사마운트에 고정하여 설치하고자 하며, 작업중의 안전을 위하여 작업자의 안전장비 착용 및 안전펜스 설치



[모니터 설치용 브라켓]



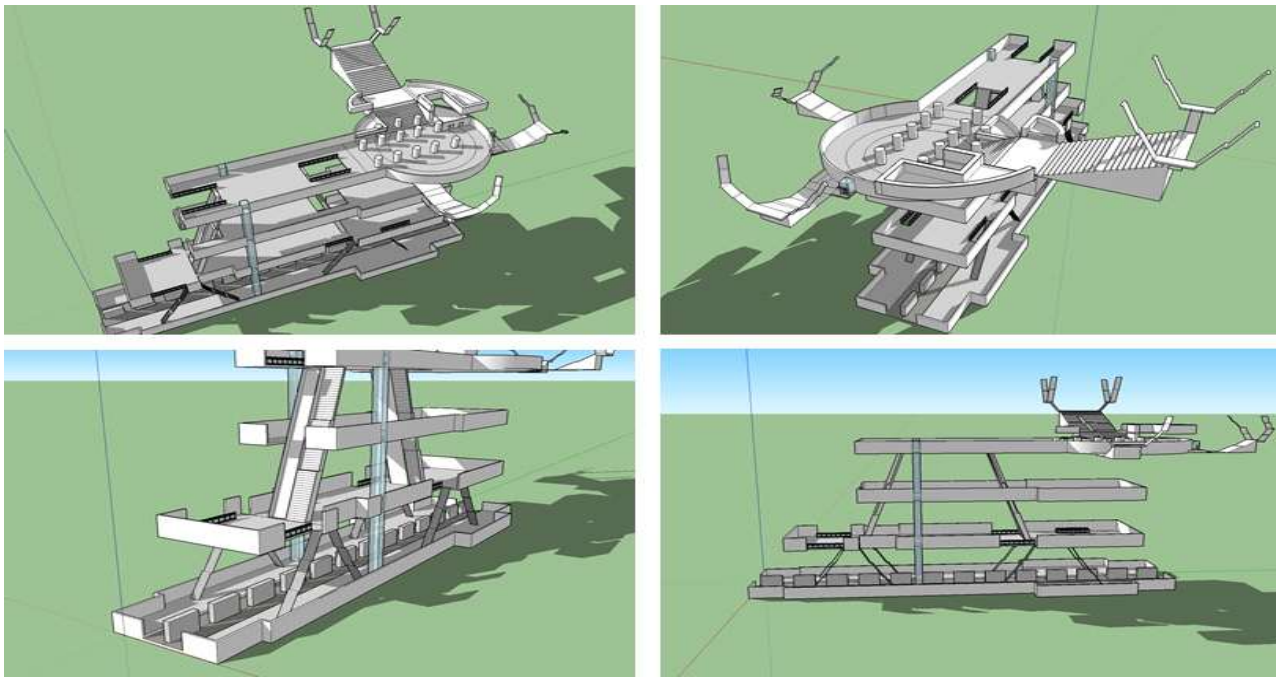
[안전펜스 설치 작업 현황]



[안전펜스 설치 작업 현황]

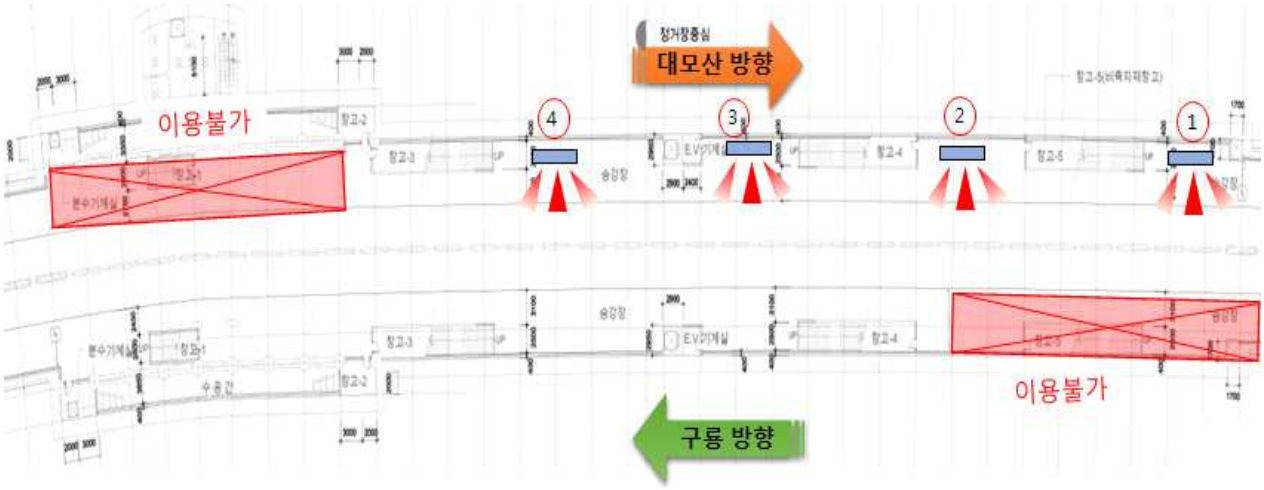
- 개포동역 역사에서의 원활한 무선 데이터통신을 위하여 최대한 LOS(Line Of Sight)를 고려하여 N키퍼의 위치를 선정하였으며, 특히 승강장의 경우 재난발생 시 승객들의 대피경로를 유도하기 위하여 동선이 나뉘는 위치마다 N키퍼를 설치 계획

위치	구조/용도	N키퍼 설치 개수	비고
지하 1층	대합실, 개찰구	3개	
지하 2층	기계실/계단	-	지하1층에서 3층까지 에스컬레이터/계단으로 연결됨
지하 3층	공조실	4개	
지하 4층	승강장	8개	
계		15개	



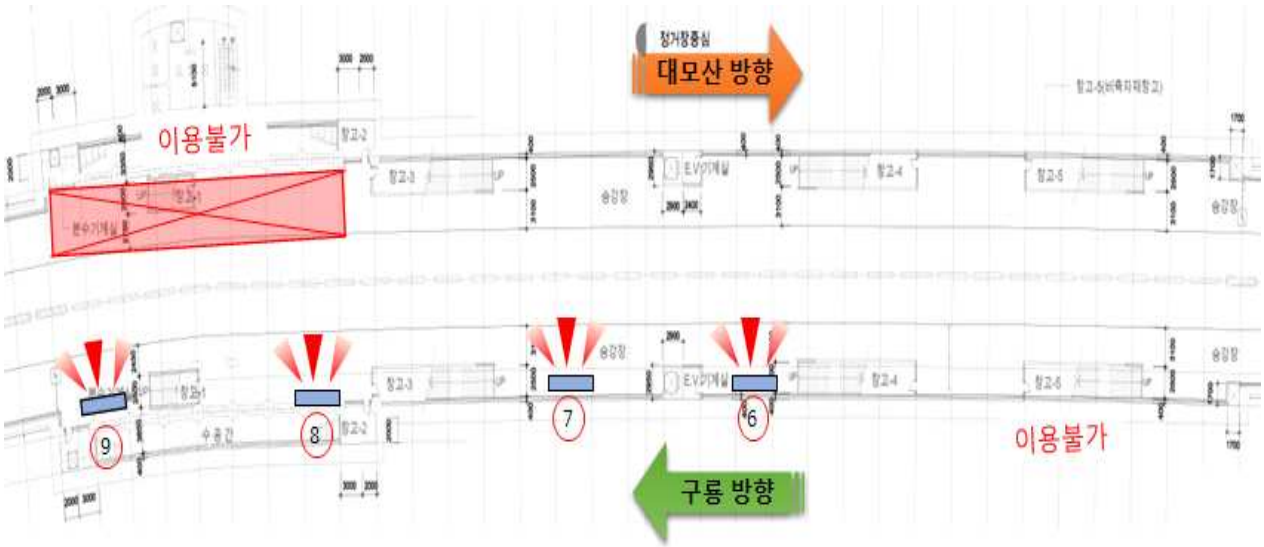
[개포동역 3D 모델링]

- 개포동역 지하4층 승강장(대모산 방향)의 스마트N키퍼 설치예정위치는 다음과 같다.





- 개포동역 지하4층 승강장(구룡 방향)의 스마트N키퍼 설치예정위치는 다음과 같다.



6



7



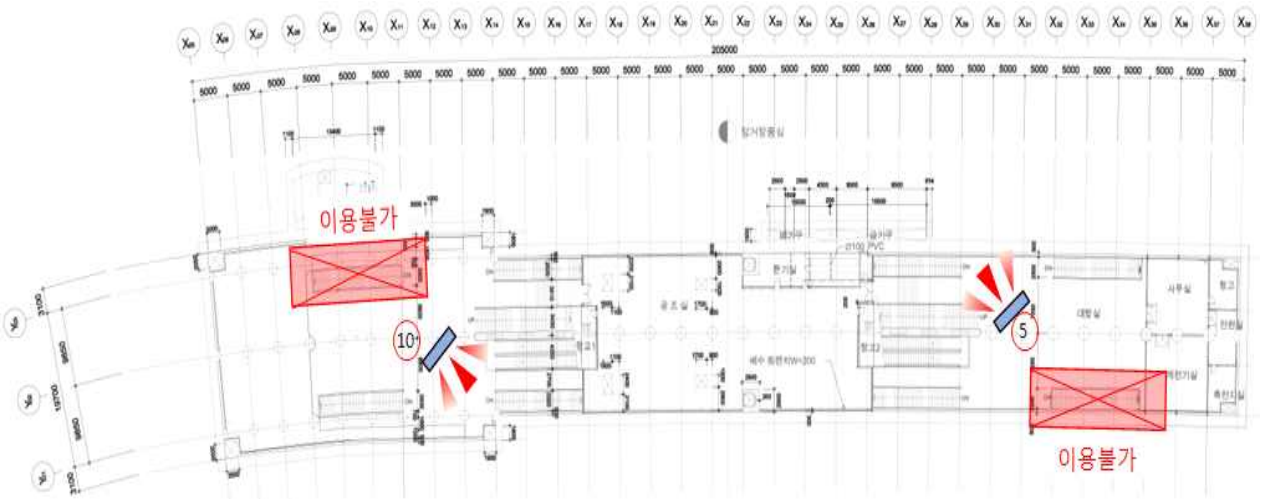
9



8



- 개포동역 지하3층 계단위치의 스마트N키퍼 설치예정위치는 다음과 같다.



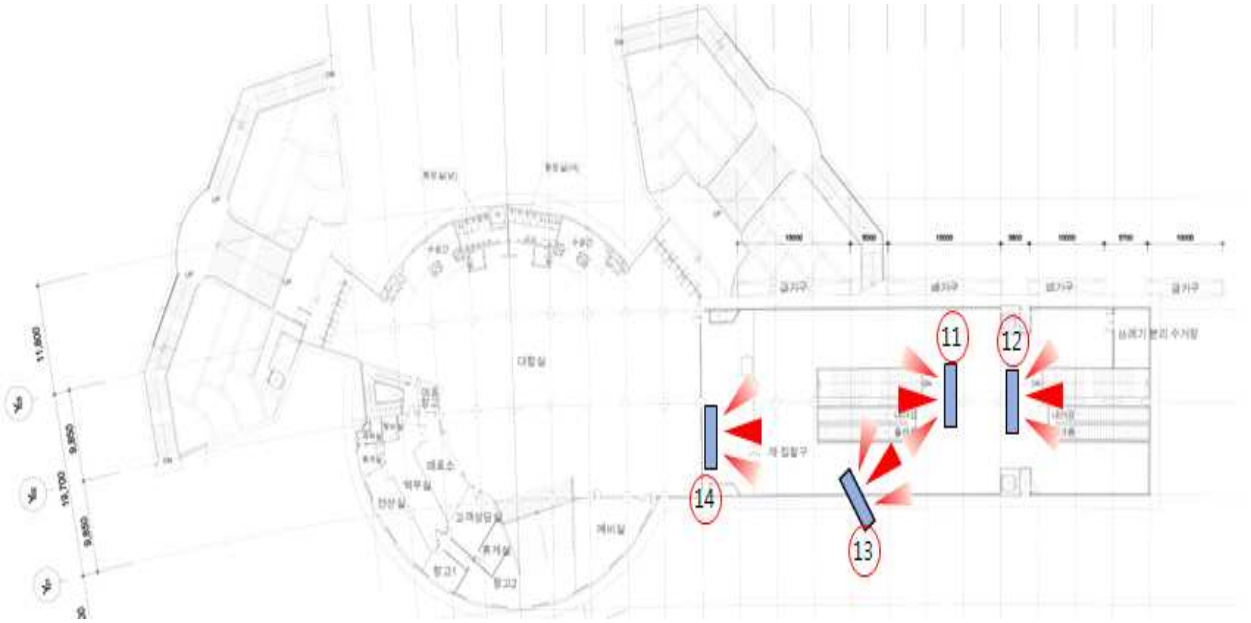
5



10



- 개포동역 지하1층 개찰구 및 대합실의 스마트N키퍼 설치예정위치는 다음과 같다.



11



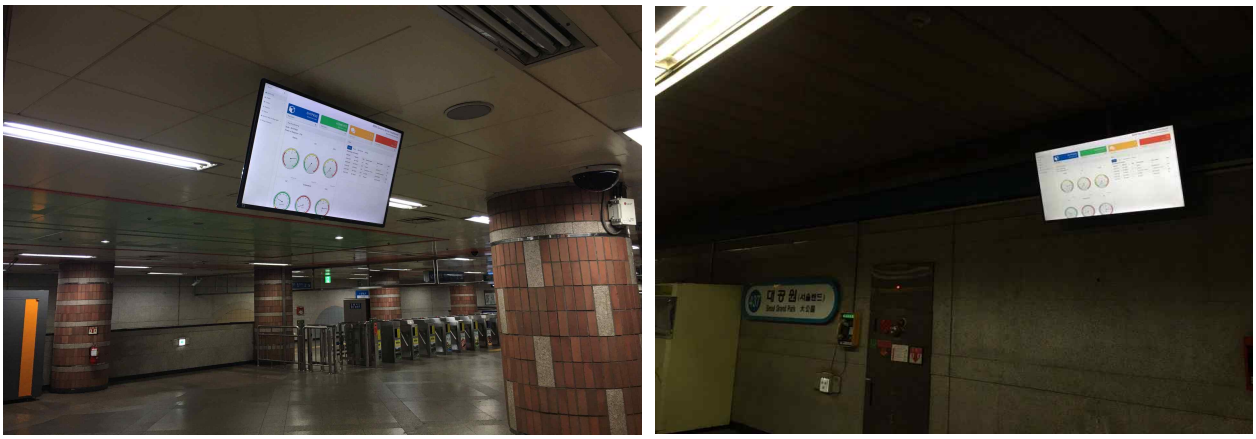
12





테스트베드 시스템 구동

- 평시 데이터 수집과 상황인지모듈의 학습, 저전력모드 테스트를 위하여 1차년도 테스트베드인 대공원역과 2차년도 테스트베드인 개포동역 두 곳에 N키퍼와 센서노드를 설치하여 테스트베드를 시범운영하고 있다. 또한, 각각의 테스트베드에서 수집되는 데이터는 웹기반으로 개발된 관리모듈을 통하여 실시간으로 모니터링이 가능하다. 실제 테스트베드에서 운영되는 시스템 구동모습은 다음과 같다.



[대공원역 시스템 구동모습]



[개포동역 시스템 구동모습]

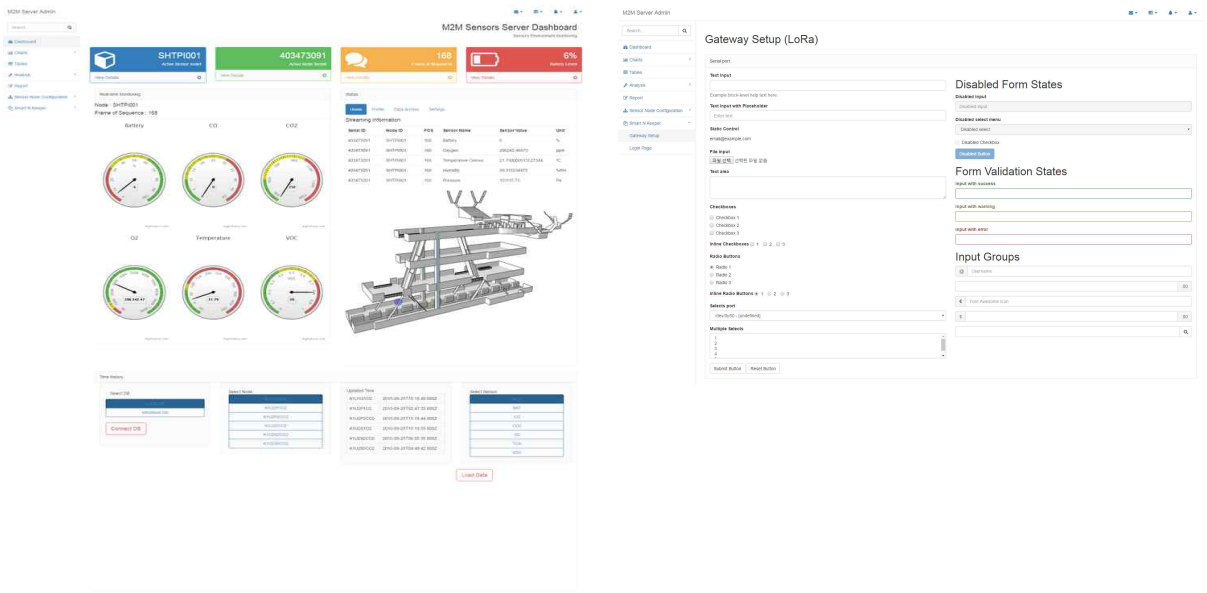
- 개포동의 경우 센서기능 및 방열/방습 기능이 추가된 새로운 센서노드 케이스를 제작하여 설치하였으며, 대공원의 경우 기존 센서노드를 사용하여 시범운영을 하였으며 시범운영 모습은 다음과 같다.



[개포동역 센서 설치모습]



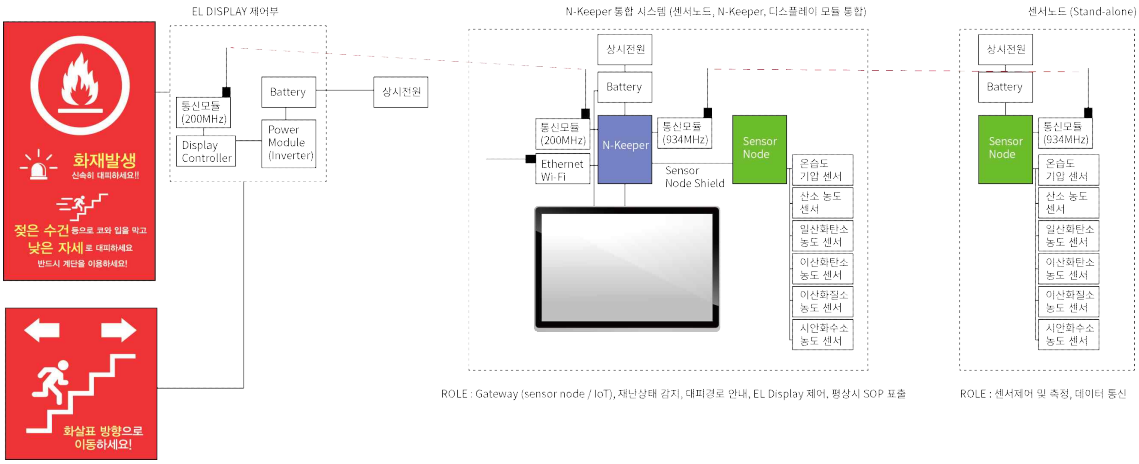
[대공원역 센서 설치모습]



[DB접속 및 스마트N키퍼 Gateway 설정화면]

4) 3차 테스트베드 화재 실험

① 3차년도에서 2차년도에 구축된 테스트베드 개포동역을 중심으로 고도화 작업을 수행하였다. 보조 EL 디스플레이 모듈과 통신모듈을 API에 포함시키고 대피경로 방향에 따른 EL 디스플레이를 제어하였다. 아래 그림과 같이 센서노드와 스마트 N키퍼 그리고 EL 디스플레이 제어를 통한 통합 테스트를 진행하였다.



② 중형 EL 디스플레이 패널



③ 원기둥 부착형 EL 디스플레이 패널



화재 대피 안내문



화학 테러 대피 안내문

④ 비상구형 및 테이프형 EL 디스플레이 설치



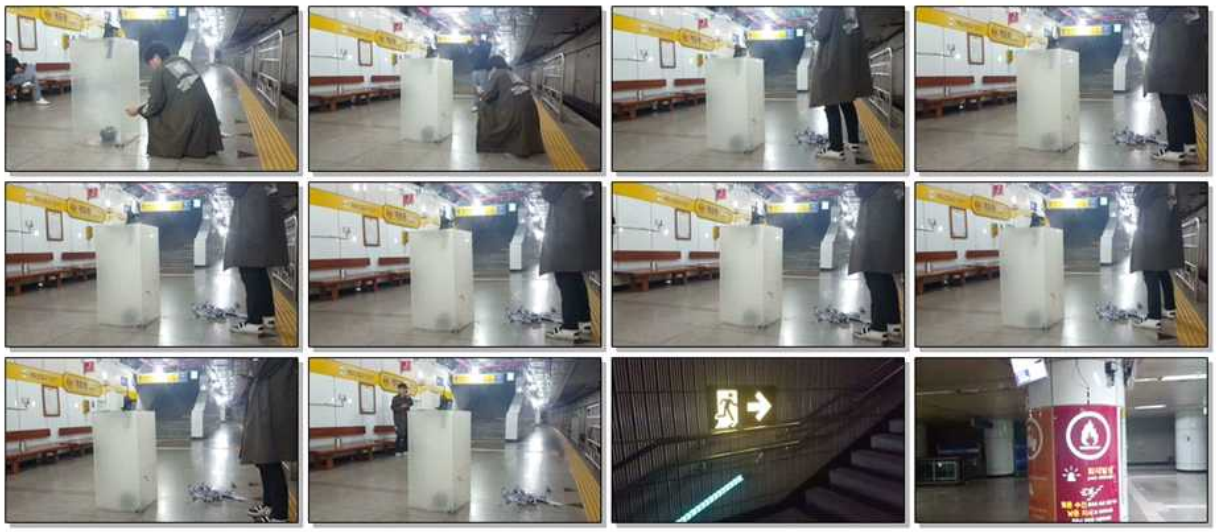
비상구형 EL 디스플레이 패널



계단 하부 테이프형 방향지시 EL 디스플레이 패널

⑤ 화재테스트 결과

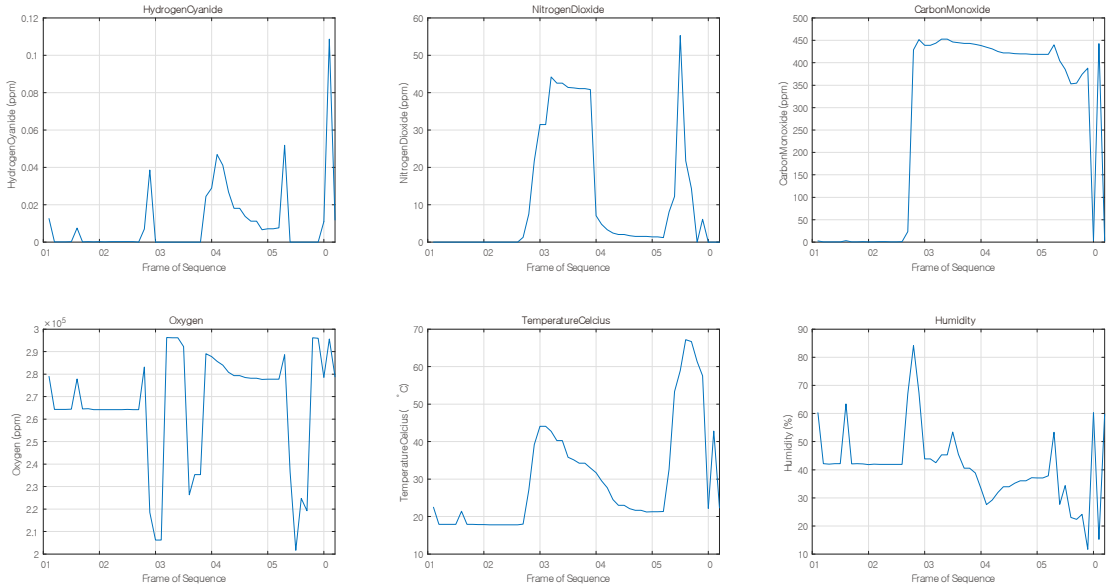
- 화재테스트는 2017년 4월 21일 02:33 영업 종료후 개포동역에서 실시하였다.
- 센서노드에서 화재를 발생시키고 알고리즘 및 룰셋이 작동하여 1분 30초만에 화재인식이 되었다.
- 역무실 응답 대기 30초 후 재난모드로 자동전환하여 대피경로 안내를 표시하였다.



[화재테스트 영상 촬영 스냅샷]

① 화재 실험 센서노드 (SHTPI001) 데이터

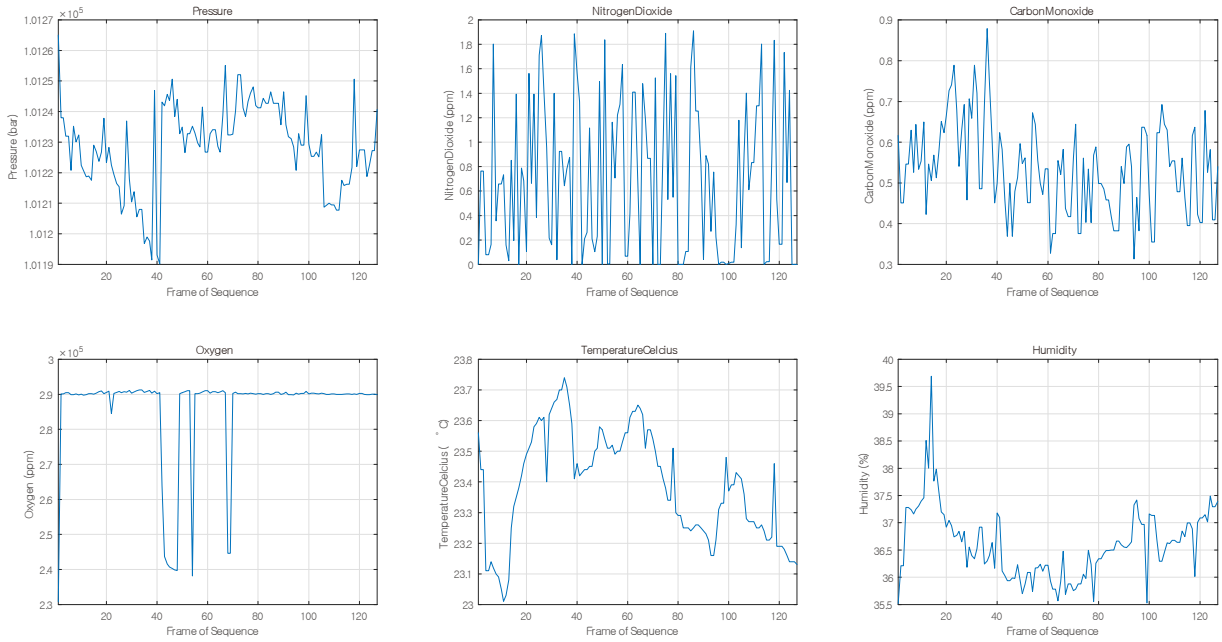
화재 발생시, HCN은 약간 증가하는 양상을 보이나, 이산화질소(NO_2)와 일산화탄소(CO)는 급격히 증가함을 알 수 있다. 또한 산소농도는 초기에 급격히 떨어지며, 온도는 섭씨 70도 가까이 증가한다.



[센서노드 SHTPI001 화재테스트 결과]

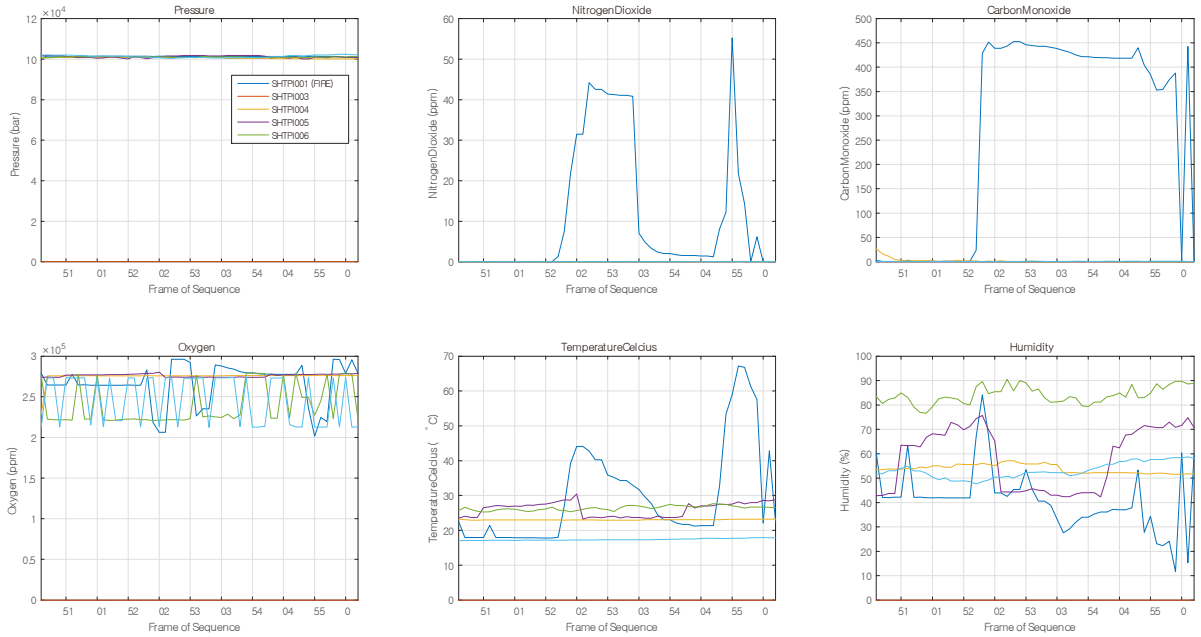
② 평상시 센서노드 (SHTPI013) 데이터

장기 계측 결과 평상시에는 이산화질소와 일산화탄소등 기준치를 못미치는 낮은 값을 유지하고 있다.



③ 화재 실험 센서노드 (SHTPI001) 데이터의 평시 데이터와 비교

평시 모드인 다른 센서노드에 비해 화재 실험 당시 센서노드(SHTPI001)은 이산화질소(NO_2)와 일산화탄소(CO)는 급격히 증가함을 보인다. 산소농도는 오차범위 이내이고, 온도는 화재를 분별할 수 있으나 그 차이가 크지 않다. 따라서 이산화질소와 일산화탄소의 룰셋정의에 의하여 결정론적 알고리즘으로도 화재 탐지가 가능하다고 판단된다.



[센서노드 SHTPI001 화재 감지된 센서와 평시 모드의 센서데이터 비교]

제4장. 목표 달성도 및 관련 분야 기여도

1. 목표 달성도

가. 1차년도

번호	세 부 연 구 목 표	달 성 내 용	달성도(%)
1	지하시설(지하철) 재난 대응 프로세스 (SOP) 기초 연구	▪ 스웨덴 룬드 대학 소방안전공학과 및 Lynn Field사 현지방문 및 자료수집 - 지하철 화재상황에 대한 각종 실험자료 - 지하철 재난대응 장비 개발현황 ▪ 지하철 재난유형별 사례 조사 및 분석 - 탈선, 충돌, 화재, 테러, 폭발 사례 등 ▪ 지하철 안전시설 조사 및 분석 - 역사별 소방계획, 방재시설 등 ▪ 역사별 재난대응매뉴얼 수집 및 분석 - 양재역, 공덕역, 죽전역, 대공원역 ▪ SOP 매뉴얼 e-book 제작 - 지하철 재난유형 사례집, 안전시설 사례집, 재난대응 매뉴얼 e-book 제작	100 %
2	SOP기반 재난 대응 프로세스 및 재난전조대응 M2M 알고리즘 개발 기초 연구	▪ 재난전조 감지 및 재난대응 시스템 관련 논문 2편 국내 학술지 게재 - 한국방재안전학회 논문집(`14.12, `15.06) ▪ 스마트N키퍼 재난대응 프로세스 처리를 위한 의사결정 기본 알고리즘 개발	100 %
3	스마트N키퍼 프로토타입 개발	▪ 스마트N키퍼 H/W Prototype 개발 - 가스(CO, CO2, O2, VOC)/음향/조도/가속도 계측이 가능한 무선 노드 - 지하공간에 산재된 무선 센서링 노드 및 게이트웨이를 연결할 수 있는 독립적인 무선 메쉬 네트워크 - USB Gateway, 미니PC, 액정패널, 스피커로 구성된 정보제공 장치 ▪ 스마트N키퍼 S/W Prototype 개발 - 무선 센서링 노드용 임베디드 프로그램 - 무선 메쉬 네트워크용 제어용 프로그램 - 시리얼 데이터 모니터링용 미들웨어 - 의사결정 지원용 미들웨어 - 승객 유도용 스마트N키퍼 인터페이스	100 %
4	스마트N키퍼 프로토타입 테스트베드 운용 (추가 성과)	▪ 지하철 4호선 대공원역 테스트베드 구축 - 무선 센서링 노드 16개소 설치 - 스마트N키퍼 6개소 설치 - 지하 1, 2층 연결 메쉬 네트워크 구성 - 가스/음향/조도/가속도 센서링 테스트 - 스마트N키퍼 인터페이스 테스트	100 %

나. 2차년도

번호	세부연구목표	달성내용	달성도(%)
1	M2M 기반 지하시설(지하철) 재난대응 프로세스(SOP) 개발	SOP ▪ 기존 지하철역에서 사용하는 재난대응 매뉴얼에서 본 과제의 성과물(스마트N키퍼)을 이용한 재난대응 매뉴얼 보고서 제시 ▪ 싱가포르 SMRT 방문 : 무인 열차 운영에 필요한 재난 대응 방법 공유 및 홍보 E-Book ▪ 이용객들이 스마트N키퍼를 잘 활용할 수 있도록 직관적인스마트N키퍼 활용방법(매뉴얼)안내용 E-Book 제작 및 시스템 탑재	100%
2	SOP기반 재난 대응 프로세스 및 재난전조대응 M2M 알고리즘 개발	▪ 가스센서의 무선네트워크 기반 긴급대피유도 시스템 개발의 주제로 2016년 한국도시철도학회 춘계학술대회 논문게재 및 오랄발표 ▪ 재난전조 예측/재난대응 모듈의 개체-관계 다이어그램(Entity-Relationship Diagram) 작성	100%
3	스마트N키퍼 개발	무선 센서링 노드 성능개선 ▪ 당해연도에 추가된 테스트베드에 센서 신뢰성 및 테러에 대응할 수 있는 센서노드 설치 및 운영 ▪ 특허등록, 제 10-1569723 호 다점송출식 다중이용시설 가스 탐지 방법 통신모듈 개선 ▪ IoT 및 정보통신 전문가(교수)의 자문/회의를 통해 무선네트워크 구성 및 통신모듈 선정 ▪ 스마트N키퍼, 센서노드의 무선통신 망 구축 SOP모듈 개발 ▪ 스마트N키퍼를 활용한 재난대응매뉴얼(SOP)을 모듈화 하여 이용객들이 재난발생 시 활용할 수 있도록 디스플레이 ▪ 개별 센서 데이터의 상태에 기반한 재난 전조 예측 알고리즘 개발 피난경로 분석/대피 유도 모듈개발 ▪ 정량적 공간분석기능 개발 및 공간분석 결과 기반 최적 피난경로 분석 알고리즘 개발 ▪ 마스터(서버)의 시나리오 모드 추가를 통한 SOP 기반 재난대응(대피경로 유도) 검증	100%
4	스마트N키퍼 테스트베드 대상 확대	▪ 지하 4층의 개포동역의 테스트베드 선정 ▪ 테스트베드 스마트N키퍼 및 센서노드 설치 ▪ 테스트베드 시범 운영 ▪ 재난상황과 발생위치에 따른 시나리오구성 ▪ 설정된 시나리오에 대한 재난모드 테스트 및 검증 ▪ 협업환경 활성 및 버전관리를 위한 GIT 서버 운영 ▪ Middleware 수정 및 고도화/ 센서노드 파싱 모듈화 ▪ 각 센서의 측정 데이터 저장 및 관리 ▪ Redis 및 Monggo DB를 활용한 데이터 리스폰스 속도 향상	100%

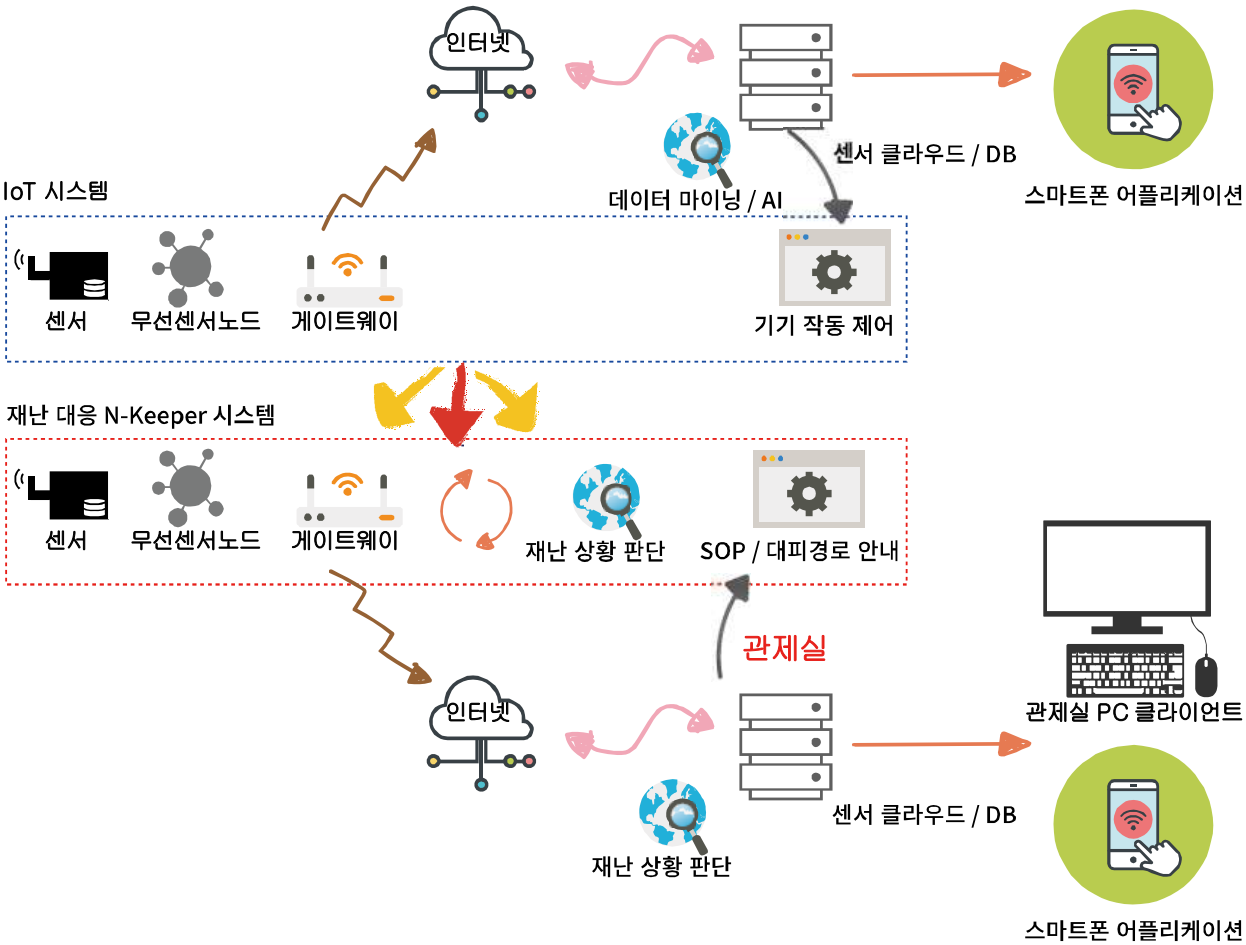
다. 3차년도

번호	세부연구목표	달성내용	달성도(%)
1	지하시설(지하철)재난대응 프로세스 (SOP) 매뉴얼 고도화	SOP ▪ 본 과제에서 개발된 재난 대응 SOP프로그램을 통해 시나리오별로 스마트N키퍼의 대응 방법이 포함된 SOP를 직접 표출하여 직관적으로 기존 SOP와 함께 적용이 가능 E-Book ▪ 이용객들이 스마트N키퍼를 잘 활용할 수 있도록 직관적인 스마트N키퍼 활용방법(매뉴얼)안내용 E-Book 제작 및 시스템 탑재 UI업그레이드	100%
2	SOP기반 재난대응 프로세스 및 재난전조 대응 M2M S/W 고도화	▪ 화재테스트를 통한 재난 대응 모듈 룰셋 구성 및 업데이트 ▪ 방재전문가 시연을 통한 상황인지 모듈 및 역무실 역무원을 위한 SOP모듈의 고도화 확인 ▪ 소프트웨어 저작권 등록 ▪ 방재안전학회 논문 게재 예정	100%
3	스마트N키퍼 개발 고도화	무선 센서링 노드 성능개선 ▪ 자동 캘리브레이션 센서를 사용하여 장기 계측 확인 ▪ Half duplex 시각 동기화 구현, (마스터/N키퍼) 통신모듈 개선 ▪ 저전력 장거리 통신으로 인해 음역지역 해소 ▪ 스마트N키퍼, 센서노드의 무선통신 망 구축 SOP모듈 개발 ▪ 스마트N키퍼를 활용한 재난대응매뉴얼(SOP)을 모듈화 하여 역무원 및 승객들이 재난발생 시 활용할 수 있도록 디스플레이 및 제어 ▪ 개별 센서 데이터의 상태에 기반한 재난 전조 예측 알고리즘 고도화 피난경로 분석/대피 유도 모듈개발 ▪ 정량적 공간분석기능 개발 및 공간분석 결과 기반 최적 피난경로 분석 알고리즘 고도화 ▪ 마스터(서버)의 시나리오 모드 및 재난 예측 룰셋 기능 추가를 통한 SOP기반 재난대응(대피경로 유도) 검증	100%
4	스마트N키퍼 테스트베드 운영 고도화	▪ 지하 4층의 개포동역의 테스트베드 고도화 ▪ 테스트베드 스마트N키퍼 및 센서노드 설치 ▪ 테스트베드 역무실 시범 운영 ▪ 재난상황과 발생위치에 따른 시나리오별 대응 방안 도시 ▪ 실제 화재 테스트를 통한 재난모드 테스트 및 검증 ▪ 직관적인 UI를 통한 웹 관리 시스템 업그레이드 ▪ Middleware 수정 및 고도화/ 센서노드 파싱 모듈화 / 장기 계측 안정화 ▪ 각 센서의 측정 데이터 저장 및 관리의 모듈화	100%

2. 관련 분야 기여도

○ 본 과제는 지능형 재난 대응 시스템으로 기존의 IT기술과 방재기술을 조합하고 최적화 하여, 방재에 특화된 M2M 및 IoT 시스템을 구축하였다.

○ 본 과제는 아래의 그림과 같이 기존의 IoT 요소기술에서 방재에 특화된 요소기술을 조합하고 개발하여 재난 대응형 IoT 기술에 기여하였다.



○ 본 과제에서 개발한 시스템은 기존 화재 및 재난 시스템과 연계가 가능하며, 열차 운행에 있어 재해우려개소 낙석 감지 시스템과, 선로 작업자 안전을 위한 열차접근경보 시스템등과 같은 방재 시스템에 개발된 M2M요소기술을 적용하였으며 더 나아가 방재관련 요소기술에 기여할 것이라 사료된다.

1. 기존 화재 및 재난 시스템과 연계 방안

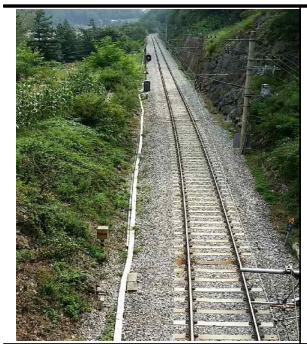
- 기존의 P형,R형 및 GR형 자동화재탐지설비와 연계하여 시스템을 구축이 가능하다. 사업화 진행할 시 현장과 협의 후 진행될 예정임.



- 추후 기존의 P형, R형 및 GR형 자동화재탐지설비와 연계하여 M2M 재난 대응 시스템을 통합해야 하며, 이는 기존 방재시스템과 연동할 수 있는 항목에 대하여 협의 후 진행할 수 있다.

2. 철도공사 재해우려개소 낙석감지 시스템 (적용완료)

- 영상감지시스템과 M2M을 연동하여 낙석재해가 발생할 경우 즉시 기관사와 관제센터로 경고를 하고, 의사결정 지원하는 시스템으로 본 연구과제의 요소기술을 이용하여 개발하였음.



현장전경

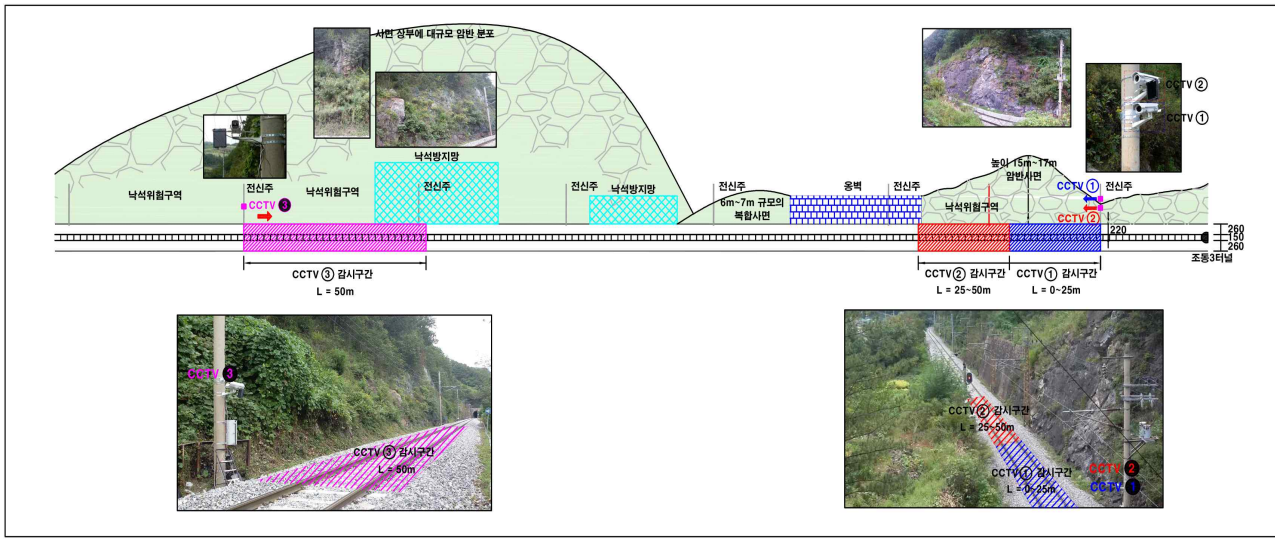


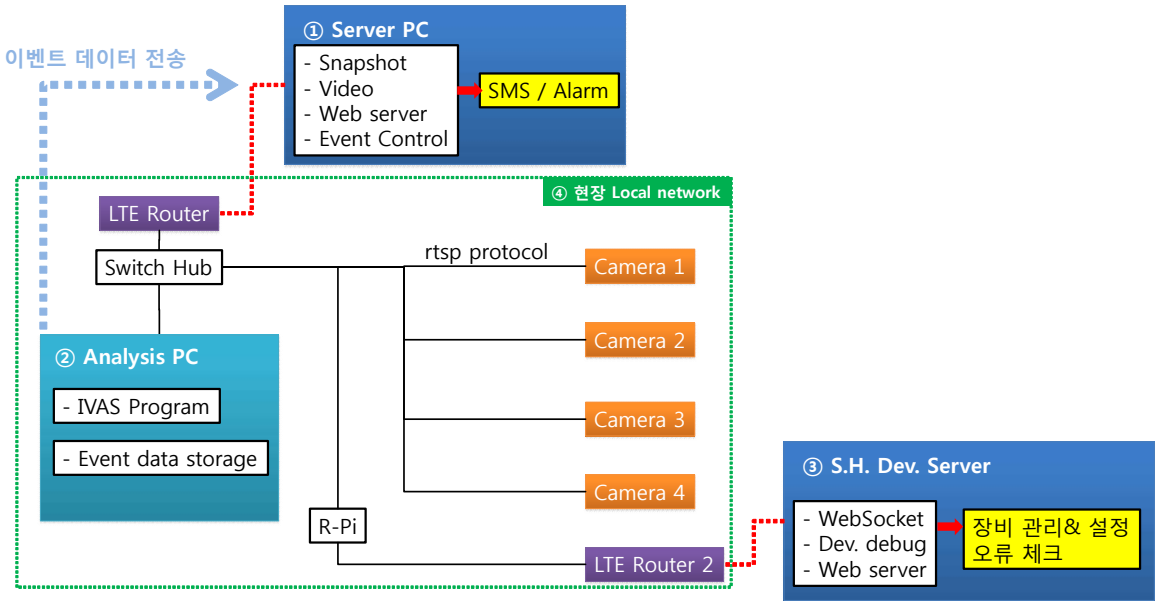
case 4 cctv 주야간 전경





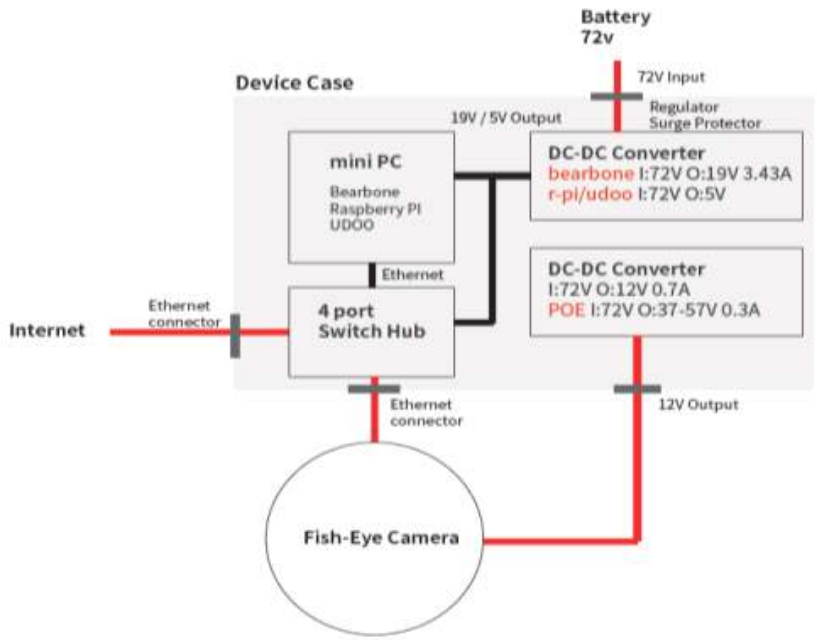
성능검증 재료

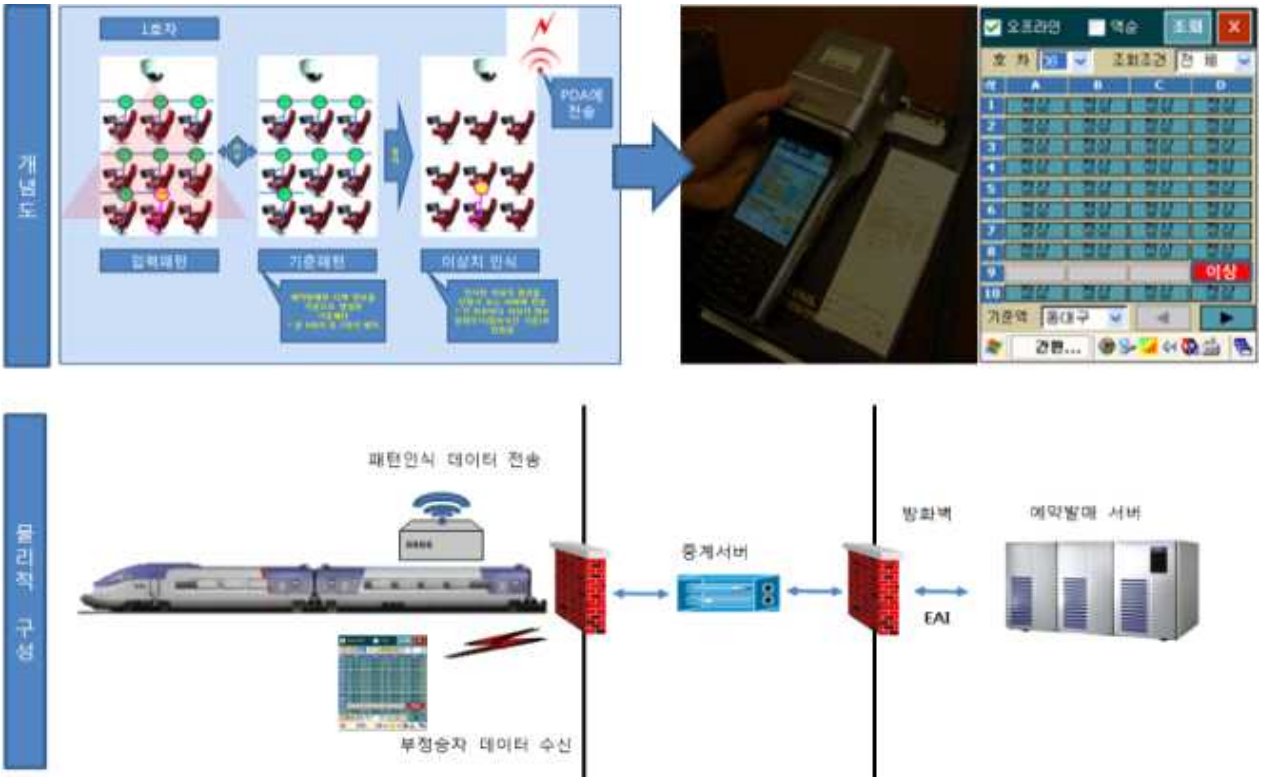




3. 철도공사 승무원 승객검표 시스템 (적용완료)

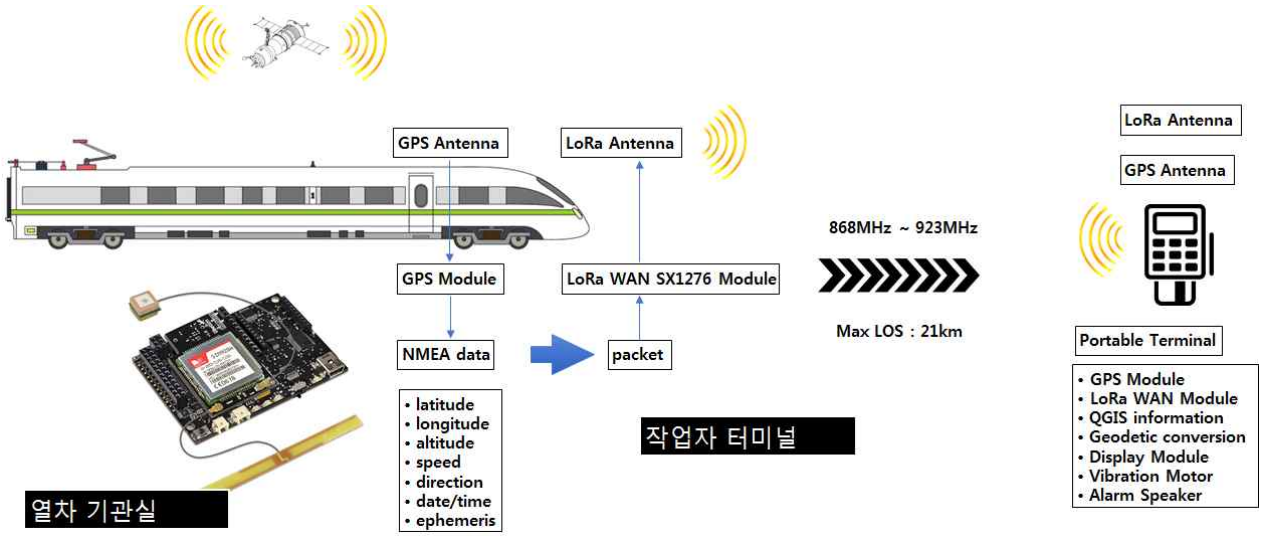
- M2M 시스템의 요소기술을 적용함으로써 영상감지 시스템과 연동 가능성 확보





4. 선로작업자 열차접근경보 시스템 (한국철도공사)

- 본 과제에서 적용한 저전력 장거리 통신망 LoRa에 착안하여 위탁연구기관인 한국철도공사에서 작업자 안전을 위한 M2M 기반 접근경보시스템을 개발 중이다.



제6장. 연구 과정에서 수집한 해외 과학기술 정보

1.딥러닝 기술 수집 및 싱가포르 SMRT 지하철 승객 안전 관련 회의

월/일	시 각	방문기관	수 집 내 용
4/26	13:00~18:00	SMRT	- 싱가포르 도시철도 재난관리 현황 자료 조사 (지하철 안전관리, 재난대응 매뉴얼 등)
4/27	10:00~16:00	RE:WORK	- 국제딥러닝회의(ELEVENTH GLOBAL DEEP LEARNING SUMMIT SINGAPORE) 참관 - 패턴인식/이미지탐색 관련 최근 기술동향 자료수집
4/28	10:00~16:00	RE:WORK	- 국제딥러닝회의(ELEVENTH GLOBAL DEEP LEARNING SUMMIT SINGAPORE) 참관

* RE:WORK DEEP LEARNING SUMMIT

- M2M 기반 지하공간(지하철) 재난대응 대화형 네트워크 시스템 개발」연구과제 수행 관련 화재 및 테러에 대응에 인공지능을 적용할 수 있는 요소기술을 개발에 활용 가능하며 추후 사업화 과제 추진시 지하철 역사의 역무원 및 각 유관기관에 상황 전파 대응할 수 있는 SOP에 적용 가능성 검토

  ○재난상황 판단 및 대피경로 안내에 적용 가능한 딥러닝 요소기술 조합 방법 검토

  ○CuDNN 및 TensorFlow 등 요소기술을 영상 시스템을 적용하여 재난상황 판단과 최적 대피경로 안내에 활용

* SMRT 방문

- 국가R&D 과제로 개발중인 지하철 재난대응 시스템 설명 및 자문

○개발중인 지하철 재난대응 시스템 SMRT 적용 필요성 인식, 향후 코레일 시범운영 결과 공유 시 SMRT 내부 검토 및 심의 후 적용 답변

○화재, 독가스 등 센서의 정밀도가 본 개발 시스템의 핵심으로 센서의 성능 및 상태를 측정할 수 있는 방안에 대해 면밀한 검토가 필요함

○재난발생 시 고객은 시각에 의존하여 역사를 탈출하기 때문에 대피경로 안내 UI는 최대한 단순하며 시인성 있게 개발할 필요성이 있음

- 지하철 재난(화재 등) 발생시 조치현황 의견수렴

○지하철 화재 등 재난발생 시 관제통보 및 열차제어, 고객대피 순으로 비상조치 시행토록 SOP 운영 중 ※ SOP를 요청하였으나 내부 보안사항으로 검토 후 추후 공유(Email) 답변

○1개역 기준 1조에 2명 근무 시행으로 골든타임 확보 문제가 우려되나 이에 대한 대책이 없은 실정임.

○지하철내 화재의 경우 객차내 설치된 CCTV를 관제 및 역에서 확인하여 비상조치 시행(무인운전 시행으로 모든 객차에 2개의 CCTV 설치 운영 중)

- SMRT 관제실 견학 : 시설은 전반적으로 국내와 유사하나, 운행열차별, 객차별 실시간 CCTV 영상 모니터링 시스템 운영(보안시설로 사진촬영 불가)

* 출장 성과

- 지하철 역사 재난 대응에 대한 싱가포르의 상용기술 상황 파악
- 싱가포르 SMRT 관계자에 지하철 M2M 재난대응시스템에 대한 프리젠테이션 및 홍보
- 무인 운행중인 SMRT에 적합한 지하철 M2M 재난대응시스템에 대하여 설명 및 추후 KORAIL협조하에 SMRT에 적용 가능한지 검토
- Deep Learning 최신 기술 동향 파악
- RE:WORK에 전시된 소형 이더넷 8포트, 1TB 메모리에 Deep Learning을 적용가능한

NVDIA Jetson TX1/TX2 탑재한 보드 성능 탐색

* 활용 계획

- Deep Learning 기술을 각종 재난 대응 시스템에 적용
- 대피경로 안내에 있어 혼잡상황을 판단하여 대피경로 분산
- 열차 운행에 있어 컴퓨터 비전 시스템과 Deep Learning 기술을 도입하여 운행 안전 확보
- 난 대응 영상시스템에 있어 DAG-RNN을 활용하여 움직이는 물체를 픽셀레벨로 분리하여 상황판단
- SMRT와 지속적 우호관계 유지를 통한 상호 기술 정보 공유





제7장. 연구개발성과의 보안등급

- 해당사항 없음

제8장. 국가과학기술종합정보시스템에 등록한 연구시설·장비 현황

- 해당사항 없음

제9장. 연구개발과제 수행에 따른 연구실 등의 안전 조치 이행 실적

- 해당사항 없음

제10장. 연구개발과제의 대표적 연구 실적

번호	구분 (논문/ 특허/ 기타)	논문명/특허명/기타	소속 기관명	역할	논문 게재지/ 특허 등록 국가	영향력 지수	논문 게재일 /특허 등록일	사사 여부 (단독 사사 또는 중복 사사)	특기 사항 (SCI 여부/인용 횟수 등)
1	논문	피난 경로 분석 및 유도 알고리즘 개발에 관한 연구	㈜승화기술정책연구소 / 한국철도공사	주관	한국방재안전학회	-	2015.06.30	-	-
2	논문	사물인터넷 기반 지하철 역사공간 재난대응 시스템 개발에 관한 연구	㈜승화기술정책연구소 / 한국철도공사	주관	한국방재안전학회	-	2016.09.08	-	-
3	논문	지하철 재난 전조 예측 모델 개발	㈜승화기술정책연구소 / 한국철도공사	주관	한국방재안전학회	-	2017.09.30	-	-
4	특허	재난전조 및 대피경로 안내 SOP 프로그램	㈜승화기술정책연구소 / 한국철도공사	주관	한국	-	2015.03.31	-	-
5	특허	다점 송출식 다중이용시설 가스 탐지 방법	㈜승화기술정책연구소 / 한국철도공사	주관	한국	-	2015.11.11	-	-

주 의

1. 이 최종보고서는 국토교통부에서 시행한 국토교통기술촉진연구사업의 연구보고서입니다.

2. 이 최종보고서 내용을 발표하는 때에는 반드시 국토교통부에서 시행한 사업의 연구개발성과임을 밝혀야 합니다.

3. 국가과학기술 기밀 유지에 필요한 내용은 대외적으로 발표 또는 공개하여서는 안 됩니다.